Administration Guide

Amazon AppStream 2.0

Amazon AppStream 2.0: Administration Guide

Table of Contents

What Is Amazon AppStream 2.0? ... 1
Features ... 1
Key Concepts ... 3
How to Get Started ... 5
Accessing AppStream 2.0 ... 6
Setting Up ... 7
Sign up for an AWS account ... 7
Create an administrative user ... 7
Get Started: Set Up With Sample Applications ... 9
Step 1: Set Up a Sample Stack, Choose an Image, and Configure a Fleet ... 9
Step 2: Provide Access to Users ... 17
Resources ... 18
Networking and Access ... 20
Internet Access ... 20
VPC Requirements ... 22
VPC Setup Recommendations ... 22
Configure a VPC with Private Subnets and a NAT Gateway ... 24
Configure a VPC with a Public Subnet ... 33
Use the Default VPC and Public Subnet ... 36
Amazon S3 VPC Endpoints ... 39
AppStream 2.0 Connections to Your VPC ... 40
Network Interfaces ... 40
Management Network Interface IP Address Range and Ports ... 40
Customer Network Interface Ports ... 41
User Connections to AppStream 2.0 ... 42
Bandwidth Recommendations ... 43
IP Address and Port Requirements ... 44
Allowed Domains ... 45
Image Builders ... 47
Launch an Image Builder ... 48
Connect to an Image Builder ... 51
AppStream 2.0 Console (Web Connection) ... 52
Streaming URL (AppStream 2.0 Client or Web Connection) ... 53
Image Builder Actions ... 55

Instance Metadata for Image Builders ... 56
Install AMD Driver on Graphics Design Instances ... 56
Base Image and Managed Image Update Release Notes ... 58
Images ... 74
Default Settings and Application Launch Performance ... 74
Creating Default Application and Windows Settings ... 75
Optimizing the Launch Performance of Your Applications ... 75
Manage AppStream 2.0 Agent Versions ... 76
Create an Image That Always Uses the Latest Version of the AppStream 2.0 Agent ... 77
Create an Image That Uses a Specific Version of the AppStream 2.0 Agent ... 78
Create an Image That Uses a Newer Version of the AppStream 2.0 Agent ... 79
Agent Release Notes ... 80
Tutorial: Create a Custom Image by Using the Console ... 106
Step 1: Install Applications on the Image Builder ... 107
Step 2: Create an AppStream 2.0 Application Catalog ... 108
Step 3: Create Default Application and Windows Settings ... 109
Step 4: Test Applications ... 110
Step 5: Optimize Applications ... 111
Step 6: Finish Creating Your Image ... 112
Step 7 (Optional): Tag and Copy an Image ... 113
Step 8: Clean Up ... 115
Administer Your Images ... 115
Delete a Private Image ... 116
Copy an Image That You Own to Another AWS Region ... 116
Share an Image That You Own With Another AWS Account ... 117
Stop Sharing an Image That You Own ... 119
Keep Your AppStream 2.0 Image Up-to-Date ... 119
Windows Update and Antivirus Software on AppStream 2.0 ... 123
Programmatically Create a New Image ... 126
Create Your AppStream 2.0 Image Programmatically ... 126
Creating Default Application and Windows Settings with the Image Assistant CLI operations ... 127
Optimizing the Launch Performance of Your Applications with the Image Assistant CLI Operations ... 127
Process Overview for Programmatically Creating an AppStream 2.0 Image ... 128
Image Assistant CLI Operations for Creating and Managing Your AppStream 2.0 Image ... 129

Create Your Linux-Based Images ... 140
Creating Default Application Settings for Your Users ... 140
Creating Default Environment Variables for Your Linux Users ... 144
Optimizing the Launch Performance of Your Linux Applications ... 145
Creating Session Scripts ... 146
Using the Image Assistant CLI Tool for Linux ... 147
Enabling and Disabling Webcam Support ... 147
Tutorial: Create a Custom Linux-Based Image ... 148
Tutorial: Enable Japanese Support ... 155
Use Session Scripts to Manage Your Users' Streaming Experience ... 159
Run Scripts Before Streaming Sessions Begin ... 160
Run Scripts After Streaming Sessions End ... 164
Create and Specify Session Scripts ... 166
Session Scripts Configuration File ... 168
Using Windows PowerShell Files ... 171
Logging Session Script Output ... 171
Use Storage Connectors with Session Scripts ... 171
Enable Amazon S3 Bucket Storage for Session Script Logs ... 173
Use Session Scripts on Multi-Session Fleets ... 175
Applications Manager ... 178
App Blocks ... 178
Custom App Blocks ... 179
AppStream 2.0 App Blocks ... 188
Unsupported Applications ... 199
App Block Builder ... 201
Create an App Block Builder ... 201
Connect to an App Block Builder ... 204
App Block Builder Actions ... 206
Applications ... 207
Store Application Icon, Setup Script, Session Script, and VHD in an S3 Bucket ... 208
Amazon S3 Bucket Permissions ... 209
Associate Applications to Elastic Fleets ... 210
Additional Resources ... 211
Fleets and Stacks ... 213
Fleet Type ... 213
Session Context ... 214

Using Session Context to Pass Parameters to a Streaming Application ... 215
Fleet Types ... 217
Always-On and On-Demand Fleets ... 218
Elastic Fleets ... 218
Instance Families ... 219
Create a Fleet and Stack ... 221
Create a Fleet ... 222
Create a Stack ... 228
Provide Access to Users ... 234
Clean Up Resources ... 234
Customize a Fleet ... 235
Persist Environment Variables ... 236
Set Default File Associations for Your Users ... 240
Disable Internet Explorer Enhanced Security Configuration ... 243
Change the Default Internet Explorer Home Page for Users' Streaming Sessions ... 245
User and Instance Metadata for AppStream 2.0 Fleets ... 249
Update a Fleet ... 251
Update a fleet with a new image ... 251
Manage applications associated to an Elastic fleet ... 252
Fleet Auto Scaling ... 253
Scaling Concepts ... 254
Managing Fleet Scaling Using the Console ... 257
Managing Fleet Scaling Using the AWS CLI ... 258
Additional Resources ... 266
Multi-Session Recommendations ... 267
User Authentication ... 270
AppStream 2.0 User Pools ... 270
User Pool End User Experience ... 270
User Pool Administration ... 272
SAML 2.0 Integration ... 276
Example Authentication Workflow ... 276
Setting Up SAML ... 278
AppStream 2.0 Integration with SAML 2.0 ... 292
Using Active Directory ... 295
Active Directory Domains ... 296
Before You Begin ... 298

Active Directory Domain Environment ... 298
Domain-Joined AppStream 2.0 Streaming Instances ... 299
Group Policy Settings ... 300
Smart Card Authentication ... 300
Tutorial: Setting Up ... 301
Step 1: Create a Directory Config Object ... 301
Step 2: Create an Image by Using a Domain-Joined Image Builder ... 302
Step 3: Create a Domain-Joined Fleet ... 303
Step 4: Configure SAML 2.0 ... 304
Certificate-Based Authentication ... 305
Prerequisites ... 305
Enable Certificate-based Authentication ... 309
Manage Certificate-based Authentication ... 309
Administration ... 310
Granting Permissions to Create and Manage Active Directory Computer Objects ... 311
Finding the Organizational Unit Distinguished Name ... 312
Granting Local Administrator Rights on Image Builders ... 313
Updating the Service Account Used for Joining the Domain ... 315
Locking the Streaming Session When the User is Idle ... 316
Editing the Directory Configuration ... 317
Deleting a Directory Configuration ... 318
Configuring AppStream 2.0 to Use Domain Trusts ... 318
Managing AppStream 2.0 Computer Objects in Active Directory ... 319
More Info ... 320
Add Your Custom Branding to AppStream 2.0 ... 321
Custom Branding Options ... 321
Adding Your Custom Branding to AppStream 2.0 ... 323
Specifying a Custom Redirect URL and Feedback URL ... 325
Previewing Your Custom Branding Changes ... 325
Color Theme Palettes ... 325
Red ... 325
Light Blue ... 326
Blue ... 327
Pink ... 328
Embed AppStream 2.0 Streaming Sessions ... 330
Prerequisites ... 330

Recommendations and Usage Considerations ... 331
Step 1: Specify a Host Domain for Embedded AppStream 2.0 Streaming Sessions ... 332
Step 2: Create a Streaming URL for User Authentication ... 332
Configuration Requirements for Using Custom Domains ... 333
Step 3: Download the Embedded AppStream 2.0 Files ... 334
Step 4. Configure Your Website for Integration with AppStream 2.0 ... 335
Import the appstream-embed JavaScript File ... 335
Initialize and Configure the `AppStream.Embed` Interface Object ... 336
Examples for Hiding Items in the AppStream 2.0 User Interface ... 337
Constants, Functions, and Events for Embedded AppStream 2.0 Streaming Sessions ... 339
Working with `HIDDEN_ELEMENTS` ... 339
Functions for the `AppStream.Embed` Object ... 340
Events for Embedded AppStream 2.0 Streaming Sessions ... 342
Examples for Adding Event Listeners and Ending an Embedded AppStream 2.0 Streaming Session ... 345
Administer Persistent Storage ... 347
Administer Home Folders ... 347
Files and Directories Associated with Compute-Intensive Applications ... 348
Enable Home Folders for Your AppStream 2.0 Users ... 349
Administer Your Home Folders ... 350
Administer Google Drive ... 356
Enable Google Drive for Your AppStream 2.0 Users ... 357
Disable Google Drive for Your AppStream 2.0 Users ... 361
Administer OneDrive for Business ... 362
Enable OneDrive for Your AppStream 2.0 Users ... 363
Disable OneDrive for Your AppStream 2.0 Users ... 364
Enable Application Settings Persistence for Your Users ... 366
How Application Settings Persistence Works ... 366
Enabling Application Settings Persistence ... 369
Prerequisites for Enabling Application Settings Persistence ... 369
Best Practices for Application Settings Persistence ... 369
How to Enable Application Settings Persistence ... 370
Administer the VHDs for Your Users' Application Settings ... 370
Amazon S3 Bucket Storage ... 371
Reset a User's Application Settings ... 373
Enable Amazon S3 Object Versioning and Revert a User's Application Settings ... 373

Increase the Size of the Application Settings VHD ... 375
Enable Regional Settings for Your Users ... 377
Configure Default Regional Settings for Your Users ... 377
Specify a Default Time Zone ... 378
Specify a Default Display Language ... 382
Specify a Default System Locale ... 384
Specify a Default User Locale ... 386
Specify a Default Input Method ... 387
Special Considerations for Application Settings Persistence ... 389
Special Considerations for Japanese Language Settings ... 390
Enable Your Users to Configure Their Regional Settings ... 391
Supported Locales ... 392
Enable Regional Settings for Your AppStream 2.0 Users ... 393
Manage Application Entitlements ... 395
Attribute-Based Application Entitlements ... 395
Create Application Entitlements ... 395
SAML 2.0 Multi-Stack Application Catalog ... 398
Dynamic Application Framework ... 398
Example API Operations WorkFlow ... 399
Use the Dynamic Application Framework ... 401
Enable and Test Dynamic App Providers ... 407
Additional Resources ... 410
Provide User Access ... 412
Supported Features ... 412
Provide Access Through a Web Browser ... 417
Requirements and Features ... 417
Configure a Connection Method for Your Users ... 420
Provide Access Through the Client ... 422
Requirements and Features ... 423
Install and Configure the AppStream 2.0 Client ... 435
Client Release Notes ... 463
Tagging Your Resources ... 478
Tagging Basics ... 478
Tag Restrictions ... 479
Adding Tags during Resource Creation in the AppStream 2.0 Console ... 480
Adding, Editing, and Deleting Tags for Existing Resources in the AppStream 2.0 Console ... 480

Working with Tags by Using the AppStream 2.0 API, an AWS SDK, or AWS CLI 481
Monitoring and Reporting 484
Monitoring Resources 484
Viewing Fleet Usage Using the Console 484
Viewing Instance and Session Performance Metrics Using the Console 485
AppStream 2.0 Metrics and Dimensions 486
Usage Reports 493
Enable Usage Reports 494
Usage Reports Fields 496
Create Custom Reports 503
Logging AppStream 2.0 API Calls 511
AppStream 2.0 Information in CloudTrail 512
Example: AppStream 2.0 Log File Entries 514
Security 517
Data Protection 518
Encryption at Rest 518
Encryption in Transit 519
Administrator Controls 520
Application Access 520
Identity and Access Management 523
Network Access 524
Access to AppStream 2.0 Resources 524
Access to Application Auto Scaling 538
Access to the S3 Bucket for Home Folders and Application Settings Persistence 540
Access to Applications and Scripts on Streaming Instances 542
Logging and Monitoring 547
Compliance Validation 548
Resilience 549
Infrastructure Security 550
Network Isolation 551
Isolation on Physical Hosts 551
Controlling Network Traffic 551
Interface VPC Endpoints 552
FIPS Endpoints 558
Security Groups 559
Update Management 561

Confused Deputy Prevention 561
Example: AppStream 2.0 service role cross-service confused deputy prevention 562
Example: AppStream 2.0 fleet machine role cross-service confused deputy prevention 563
Example: AppStream 2.0 Elastic fleets session script Amazon S3 bucket policy cross-service confused deputy prevention 565
Example: AppStream 2.0 Application Amazon S3 bucket policy cross-service confused deputy prevention 566
Troubleshooting 570
General Troubleshooting 570
SAML federation is not working. The user is not authorized to view AppStream 2.0 applications. 571
After federating from an ADFS portal, my streaming session doesn't start. I am getting the error "Sorry connection went down". 571
I get an invalid redirect URI error. 571
My image builders and fleets never reach the running state. My DNS servers are in a Simple AD directory. 571
I've enabled application settings persistence for my users, but their persistent application settings aren't being saved or loaded. 572
I've enabled application settings persistence for my users, but for certain streaming applications, my users’ passwords aren’t persisting across sessions. 572
Google Chrome data is filling the VHD file that contains my users' persistent application settings. This is preventing their settings from persisting. How can I manage the Chrome profile? 573
I set up a custom domain for my embedded AppStream 2.0 streaming sessions, but my AppStream 2.0 streaming URLs aren't redirecting to my custom domain. 573
I launched an app on a smartcard-enabled AppStream 2.0 fleet, and there are a limited number of certificates (or none) available to the app for authentication. 574
The Certification Propagation service isn't starting on my smartcard-enabled AppStream 2.0 fleet. 575
Troubleshooting Image Builders 577
I cannot connect to the internet from my image builder. 578
When I tried installing my application, I see an error that the operating system version is not supported. 578
I want to use a Windows PowerShell script to open my applications. 578
I want to make ClickOnce applications available to users. 579

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. 579
When I switched between admin and test modes, I saw a request for a password. I don't know how to get a password. 580
I get an error when I add my installed application. 580
I accidentally quit a background service on the image builder and got disconnected. I am now unable to connect to that image builder. 580
The application fails to launch in test mode. 580
The application could not connect to a network resource in my VPC. 580
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. 581
My application is missing a command line parameter when launching. 581
I am unable to use my image with a fleet after installing an antivirus application. 581
My image creation failed. 581
The Image Assistant `create-image` operation failed with an error message that access to the PrewarmManifest.txt is denied 582
Troubleshooting Fleets 582
I tried to increase my fleet capacity, but the update isn't taking effect. 582
My applications won't work correctly unless I use the Internet Explorer defaults. How do I restore the Internet Explorer default settings? 584
I need to persist environment variables across my fleet instances. 585
I want to change the default Internet Explorer home page for my users. 585
When my users end a streaming session and then start a new one, they see a message that says no streaming resources are available. 586
Troubleshooting Active Directory 586
My image builders and fleet instances are stuck in the PENDING state. 586
My users aren't able to log in with the SAML application. 587
My fleet instances work for one user but don't cycle correctly. 587
My user Group Policy objects aren't being successfully applied. 587
My AppStream 2.0 streaming instances aren't joining the Active Directory domain. 588
User login is taking a long time to complete on a domain-joined streaming session. 589
My users can't access a domain resource in a domain-joined streaming session, but they can access the resource from a domain-joined image builder. 589
My users receive the error “Certificate-Based Authentication not available” and are prompted to enter their domain password. Or users receive the error “Disconnected from session” when they are starting a session enabled with certificate-based authentication. . 590

Troubleshooting AppStream 2.0 User Issues 591
My users' AppStream 2.0 client installations fail, and they're getting a message stating that .NET Framework 4.6 is required. 592
My users' USB driver installations fail when they install the AppStream 2.0 client, and now they can't use their USB devices with AppStream 2.0. 592
My AppStream 2.0 client users are getting disconnected from their AppStream 2.0 session after every 60 minutes. 593
My users can’t copy and paste between their local device and their streaming session. 594
Some keyboard shortcuts aren’t working for users during their streaming sessions. 595
My users' drawing tablets are not working with the streaming applications I deployed. 596
The Japanese language input method doesn't work for my users during their streaming sessions 597
My user sees an error about reaching the max number of streaming sessions when they try to launch an application from the application catalog. 599
My user sees a black screen or the desktop, and their application doesn’t launch on an Elastic fleet. No error appears. 600
Troubleshooting Persistent Storage Issues 600
My stack's home folders aren't working correctly. 601
My users can't access their home folder directory from one of our applications. 601
I removed or replaced a file in a user’s home folder in Amazon S3, but my users don’t see the changes in their home folder on the fleet instance during their streaming sessions. 601
Persistent storage isn't performing as expected. My users' files are taking longer than expected to save to persistent storage. 602
My users are getting errors that files are already in use when their files are not in use. 603
When a folder contains thousands of files, AppStream 2.0 might take a long time to display the list of files. 604
Troubleshooting Notification Codes 604
Active Directory Internal Service 604
Active Directory Domain Join 605
Image Internal Service 608
Session Provisioning 609
Quotas 610
Guidance for AppStream 2.0 Users 616
Access Methods and Clients 616
Web Browser Access 616
Client for Windows 629

File Storage Options 653
Use Home Folders 653
Use Google Drive 655
Use OneDrive for Business 659
Regional Settings 663
Extension SDK Developer Guide 664
Prerequisites 664
Third-party vendor extensions 665
Document History 666
Earlier Updates 687

What Is Amazon AppStream 2.0?

Amazon AppStream 2.0 is a fully managed application streaming service that provides users with instant access to their desktop applications from anywhere. AppStream 2.0 manages the AWS resources required to host and run your applications, scales automatically, and provides access to your users on demand. AppStream 2.0 provides users access to the applications they need on the device of their choice, with a responsive, fluid user experience that is indistinguishable from natively installed applications.

With AppStream 2.0, you can easily add your existing desktop applications to AWS and enable your users to instantly stream them. Windows users can use either the AppStream 2.0 client or an HTML5-capable web browser for application streaming. You can maintain a single version of each of your applications, 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 AppStream 2.0.

Features

Using Amazon AppStream 2.0 provides the following advantages:

Access desktop applications securely from any supported device

Your desktop applications can be accessed securely through an HTML5-capable web browser on Windows and Linux PCs, Macs, Chromebooks, iPads, and Android tablets. Or, for supported versions of Windows, the AppStream 2.0 client can be used for application streaming.

Secure applications and data

Applications and data remain on AWS — only encrypted pixels are streamed to 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 virtual private cloud (VPC), and

you can use Amazon VPC security features to control access. This enables you to isolate your applications and deliver them in a secure way.

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 the AWS region that is closest to your users, and start streaming with no incremental capital investment.

Integrate with your IT environment

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 have in AWS. This reduces the movement of data between AWS and your environment and provides a faster user experience.

Integrate with your existing Microsoft Active Directory environment. This enables you to use existing Active Directory governance, user experience, and security policies with your streaming applications.

Configure 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.

Choose the fleet type that meets your needs

These are the types of fleets:

- Always-On — Streaming instances run all the time, even when no users are streaming applications and desktops. Streaming instances must be provisioned before a user is able to stream. The number of streaming instances provisioned is managed through auto scaling rules. For more information, see the section called “Fleet Auto Scaling”.

  When your users choose their application or desktop, they will start streaming instantly. You are charged the running instance fee for all streaming instances, even when no users are streaming.
- On-Demand — Streaming instances run only when users are streaming applications and desktops. Streaming instances not yet assigned to users are in a stopped state. Streaming instances must be provisioned before a user is able to stream. The number of streaming

instances provisioned is managed through auto scaling rules. For more information, see the section called "Fleet Auto Scaling".

When your users choose their application or desktop, they will start streaming after a 1-2 minute wait. You are charged a lower stopped instance fee for streaming instances that are not yet assigned to users, and the running instance fee for streaming instances that are assigned to users.

- Elastic — The pool of streaming instances is managed by AppStream 2.0. When your users select their application or desktop to launch, they will start streaming after the app block has been downloaded and mounted to a streaming instance.

  You are charged the running instance fee for Elastic fleet streaming instances only for the duration of the streaming session, in seconds.

For more information, see Amazon AppStream 2.0 Pricing.

Key Concepts

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

application

An *application* contains the information necessary to launch the application that you want to stream to your users. An application is associated with the resource that contains the files necessary to launch the application, such as an app block or image.

app block

An *app block* contains the application files that you want to stream to your users, and the details necessary to configure it.

app block builder

An *app block builder* is a virtual machine that you use to create an app block. You can launch and connect to an app block builder by using the AppStream 2.0 console. After you connect to an appblock builder, you can install your application(s). App block builder packages your app contents, uploads it to an Amazon S3 bucket, and completes app block creation.

image builder

An *image builder* is a virtual machine that you use to create an image. You can launch and connect to an image builder by using the AppStream 2.0 console. After you connect to an image

builder, you can install, add, and test your applications, and then use the image builder to create an image. You can launch new image builders by using private images that you own.

image

An *image* contains applications that you can stream to your users, and default system and application settings to enable your users to get started with their applications quickly. AWS provides base images that you can use to create image builders to then create images that include your own applications. After you create an image, you can't change it. To add other applications, update existing applications, or change image settings, you must create a new image. You can copy your images to other AWS Regions or share them with other AWS accounts in the same Region. your users, and default system and application settings to enable your users to get started with their applications quickly.

fleet

A *fleet* consists of fleet instances (also known as streaming instances) that run the applications and desktops that you specify.

stack

A *stack* consists of an associated fleet, user access policies, and storage configurations. You set up a stack to start streaming applications to users.

streaming instance

A *streaming instance* (also known as a fleet instance) is an EC2 instance that is made available to a single user for application streaming. After the user’s session completes, the instance is terminated by EC2.

user pool

Use the *user pool* to manage users and their assigned stacks.

auto scaling rules

Auto scaling rules are schedule-based and usage-based policies that you can apply to an Always-On or On-Demand fleet to automatically manage the number of streaming instances available for users to stream from.

multi-session

A *multi-session* fleet allows you to provision more than one user session on a single fleet instance. The underlying infrastructure resources are shared across all of the user sessions.

> **Note**
>
> Multi-session is available only on Always-on and On-demand fleets powered by a Windows operations system. Multi-session is not available on Elastic fleets or the Linux operating system.
>
> Make sure you are using latest AppStream 2.0 images for multi-session fleets. To keep your images are up-to-date, see the section called "Keep Your AppStream 2.0 Image Up-to-Date". For details on supported images and AppStream 2.0 agent versions for multi-session, see the section called "Base Image and Managed Image Update Release Notes".

How to Get Started

If you are using AppStream 2.0 for the first time, you can use the **Try it Now** feature or follow the Get Started with Amazon AppStream 2.0: Set Up With Sample Applications tutorial (both are available in the AppStream 2.0 console).

- **Try It Now** provides you with a free trial experience that allows you to easily start desktop applications from your desktop browser.
- The Getting Started tutorial enables you to set up application streaming by using sample applications or your own applications. If you decide to start by using sample applications, you can always add your own applications later.

  For more information about these two options, see Amazon AppStream 2.0 FAQs.

When you use the service for the first time, AppStream 2.0 creates an AWS Identity and Access Management (IAM) role to create and manage AppStream 2.0 resources on your behalf.

To use the Try It Now feature

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 Getting Started 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.

Accessing AppStream 2.0

You can work with AppStream 2.0 using any of the following interfaces:

AWS Management Console

The console is a browser-based interface to manage AppStream 2.0 resources. For more information, see Get Started with Amazon AppStream 2.0: Set Up With Sample Applications.

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. 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, 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.

Sign up for an AWS account

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

To sign up for an AWS account

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

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

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

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

Create an administrative user

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

Secure your AWS account root user

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

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

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

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

Create an administrative user

1. Enable IAM Identity Center.

   For instructions, see Enabling AWS IAM Identity Center in the *AWS IAM Identity Center User Guide*.
2. In IAM Identity Center, grant administrative access to an administrative user.

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

Sign in as the administrative user

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

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

Get Started with Amazon AppStream 2.0: Set Up With Sample Applications

To stream your applications, Amazon AppStream 2.0 requires an environment that includes a fleet that is associated with a stack, and at least one application image. This tutorial describes how to configure a sample AppStream 2.0 environment for application streaming and give users access to that stream.

> **Note**
>
> For additional guidance in learning how to get started with AppStream 2.0, see the Amazon AppStream 2.0 Getting Started Guide. This guide describes how to install and configure two applications, perform foundational administrative tasks using the AppStream 2.0 console, and provision an Amazon Virtual Private Cloud by using a provided AWS CloudFormation template.

Tasks

- Step 1: Set Up a Sample Stack, Choose an Image, and Configure a Fleet
- Step 2: Provide Access to Users
- Resources

Step 1: Set Up a Sample Stack, Choose an Image, and Configure a Fleet

Before you can stream your applications, you need to set up a stack, choose an image that has applications installed, and configure a fleet. In this step, you use a template to help simplify these tasks.

To set up a sample stack, choose an image, and configure a fleet

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. Choose **Get Started** if you are new to the console, or **Quick Links** from the left navigation menu. Choose **Set up with sample apps**.

3. For **Step 1: Stack Details**, keep the default stack name or enter your own. Optionally, you can do the following:

   - **Display name** — Enter a name to display for the stack (maximum of 100 characters).
   - **Description**— Keep the default description or enter your own (maximum of 256 characters).
   - **Redirect URL** — Specify a URL to which users are redirected after their streaming sessions end.
   - **Feedback URL** — Specify a URL to which users are redirected after they click the **Send Feedback** link to submit feedback about their application streaming experience. If you do not specify a URL, this link is not displayed.
   - **Streaming Protocol Preference** — Specify the streaming protocol you'd like your stack to prefer, UDP or TCP. UDP is currently only supported in the Windows native client. For more information, see System Requirements and Feature Support (AppStream 2.0 Client).
   - **Tags** — Choose **Add Tag**, and type the key and value for the tag. To add more tags, repeat this step as needed. For more information, see Tagging Your Amazon AppStream 2.0 Resources.
   - **VPC Endpoints (Advanced)** — You can create a private link, which is an interface VPC endpoint (interface endpoint), in your virtual private cloud (VPC). To start creating the interface endpoint, select **Create VPC Endpoint**. Selecting this link opens the VPC console. To finish creating the endpoint, follow steps 3 through 6 in *To create an interface endpoint*, in Creating and Streaming from Interface VPC Endpoints.

     After you create the interface endpoint, you can use it to keep streaming traffic within your VPC.
   - **Embed AppStream 2.0 (Optional)** — To embed an AppStream 2.0 streaming session in a webpage, specify the domain to host the embedded streaming session. Embedded streaming sessions are only supported over HTTPS [TCP port 443].

     > **Note**
     >
     > You must meet prerequisites and perform additional steps to configure embedded AppStream 2.0 streaming sessions. For more information, see Embed AppStream 2.0 Streaming Sessions.

4. Choose **Next**.

5. For **Step 2: Choose Image**, a sample image is already selected. The image contains pre-installed open-source applications for evaluation purposes. Choose **Next**.
6. For **Step 3: Configure Fleet**, we recommend that you keep any default values that are provided. You can change most of these values after fleet creation.

   - **Choose instance type** — Choose the instance type that matches the performance requirements of your applications. All streaming instances in your fleet launch with the instance type that you select. For more information, see AppStream 2.0 Instance Families.
   - **Fleet type** — Choose the fleet type that suits your use case. The fleet type determines its immediate availability and how you pay for it.
   - **Maximum session duration in minutes** — Choose the maximum amount of time that a streaming session can remain active. If users are still connected to a streaming instance five minutes before this limit is reached, they are prompted to save any open documents before being disconnected. After this time elapses, the instance is terminated and replaced by a new instance.
   - **Disconnect timeout in minutes** — Choose the amount of time that a streaming session should remain active after users disconnect. If users try to reconnect to the streaming instance after a disconnection or network interruption within this time interval, they are connected to the previous session. Otherwise, they are connected to a new session with a new instance. If you associate a stack with a fleet for which a redirect URL is specified, after users' streaming sessions end, the users are redirected to that URL.

     If a user ends the session by choosing **End Session** on the streaming session toolbar, the disconnect timeout does not apply. Instead, the user is prompted to save any open documents, and then immediately disconnected from the streaming instance.
   - **Idle disconnect timeout in minutes** — Choose the amount of time that users can be idle (inactive) before they are disconnected from their streaming session and the **Disconnect timeout in minutes** time interval begins. Users are notified before they are disconnected due to inactivity. If they try to reconnect to the streaming session before the time interval specified in **Disconnect timeout in minutes** has elapsed, they are connected to their previous session. Otherwise, they are connected to a new session with a new streaming instance. Setting this value to 0 disables it. When this value is disabled, users are not disconnected due to inactivity.

> **Note**
>
> Users are considered idle when they stop providing keyboard or mouse input during their streaming session. File uploads and downloads, audio in, audio out, and pixels changing do not qualify as user activity. If users continue to be idle after the time interval in **Idle disconnect timeout in minutes** elapses, they are disconnected.

- **Multiple user sessions** — Choose this option if you want to provision multiple user sessions on a single instance. By default, every unique user session is served by an instance (single-session).

> **Note**
>
> Multi-session is available only on Always-on and On-demand fleets powered by a Windows operations system. Multi-session is not available on Elastic fleets or the Linux operating system.
> Make sure you are using latest AppStream 2.0 images for multi-session fleets. To keep your images are up-to-date, see the section called "Keep Your AppStream 2.0 Image Up-to-Date". For details on supported images and AppStream 2.0 agent versions for multi-session, see the section called "Base Image and Managed Image Update Release Notes".

- **Maximum sessions per instance** — Maximum number of user sessions on an instance. You must choose this value based on your end users' application performance needs. You can also adjust the maximum sessions per instance for a fleet after it is provisioned. In that case, the existing user sessions and instances will not be impacted, but the fleet will become consistent with the new value of maximum sessions per instance. The value must be between 2 and 50. Before setting this value for your fleet, see the section called "Multi-Session Recommendations".
- **Minimum capacity** — Choose a minimum number of instances for your fleet based on the minimum number of expected concurrent users. Every unique user session is served by an instance. For example, to have your stack support 100 concurrent users during low demand, specify a minimum capacity of 100. This ensures that 100 instances are running even if there are fewer than 100 users.
- **Maximum capacity** — Choose a maximum number of instances for your fleet based on the maximum number of expected concurrent users. Every unique user session is served by an

instance. For example, to have your stack support 500 concurrent users during high demand, specify a maximum capacity of 500. This ensures that up to 500 instances can be created on demand.

- **Minimum user sessions for fleet** — Choose a minimum number of user sessions for your fleet based on the minimum number of expected concurrent users. For example, to have your stack support 100 concurrent users during low demand, specify a minimum capacity of 100. This ensures that 100 user sessions are available even if there are fewer than 100 users.
- **Maximum user sessions for fleet** — Choose a maximum number of user sessions for your fleet based on the maximum number of expected concurrent users. For example, to have your stack support 500 concurrent users during high demand, specify a maximum capacity of 500. This ensures that up to 500 user sessions can be provisioned on demand.

> **Note**
>
> For a single-session fleet, one instance will be launched of every user session. However, for multi-session, the number of running instances depends on the maximum sessions per instance. You must provide the capacity in terms of user sessions. The service will decide how many instances are required based on your fleet type (multi-session or single-session) and maximum sessions per instance.

- **Scaling details** — Specify the scaling policies that AppStream 2.0 uses to increase and decrease the capacity of your fleet. Note that the size of your fleet is limited by the minimum and maximum capacity that you specified. For more information, see Fleet Auto Scaling for Amazon AppStream 2.0.
- **IAM role (Advanced)** — When you apply an IAM role from your account to an AppStream 2.0 fleet instance, you can make AWS API requests from the fleet instance without manually managing AWS credentials. To apply an IAM role, do either of the following:
  - To use an existing IAM role in your Amazon Web Services account, choose the role that you want to use from the **IAM role** list. The role must must be accessible from the fleet instance. For more information, see Configuring an Existing IAM Role to Use With AppStream 2.0 Streaming Instances.
  - To create a new IAM role, choose **Create new IAM role** and follow the steps in How to Create an IAM Role to Use With AppStream 2.0 Streaming Instances.

7. Choose **Next**.

8. For **Step 4: Configure Network**, a default VPC is provided. This VPC includes a default public subnet in each Availability Zone and an internet gateway that is attached to your VPC. The VPC also includes a default security group. To use the default VPC configuration, do the following:

   - Keep the **Default Internet Access** check box selected.

     When **Default Internet Access** is enabled, a maximum of 100 fleet instances is supported. If your deployment must support more than 100 concurrent users, use the NAT gateway configuration instead.

   - For **VPC**, keep the default VPC selected for your AWS Region.

     The default VPC name uses the following format: `vpc-`*`vpc-id`* `(No_default_value_Name)`.

   - For **Subnet 1** and **Subnet 2**, keep the default public subnets selected.

     The default subnet names use the following format: `subnet-`*`subnet-id`* `|` `(`*`IPv4 CIDR block`*`) | Default in` *`availability-zone`*.

   - For **Security groups**, keep the default security group selected.

     The default security group name uses the following format: `sg-`*`security-group-id`*`-default`.

9. For **Step 5: Enable Storage**, choose one or more of the following, then choose **Next**.

   > **Note**
   >
   > Google Drive and OneDrive options are currently not available for multi-session fleets.

   - **Enable Home Folders** — By default, this setting is enabled. Keep the default setting. For information about requirements for enabling home folders, see Enable Home Folders for Your AppStream 2.0 Users.
   - **Enable Google Drive** — Optionally, you can enable users to link their Google Drive for G Suite account to AppStream 2.0. You can enable Google Drive for accounts in G Suite domains only, not for personal Gmail accounts. For information about requirements for enabling Google Drive, see Enable Google Drive for Your AppStream 2.0 Users.
   - **Enable OneDrive** — Optionally, you can enable users to link their OneDrive for Business account to AppStream 2.0. You can enable OneDrive for accounts in OneDrive domains only,

not for personal accounts. For information about requirements for enabling OneDrive, see Enable OneDrive for Your AppStream 2.0 Users.

10. For **Step 6: User Settings**, configure the following settings. When you're done, choose **Review**:

Clipboard, file transfer, print to local device, and authentication permissions options:

> **Note**
>
> **Print to local device** and **Smart card sign in for Active Directory** are currently not available for multi-session fleets.

- **Clipboard** — By default, users can copy and paste data between their local device and streaming applications. You can limit Clipboard options so that users can paste data to their remote streaming session only or copy data to their local device only. You can also disable Clipboard options entirely. Users can still copy and paste between applications in their streaming session. You can choose **Copy to local device character limit** or **Paste to remote session character limit** or both to limit the amount of data that users can copy or paste when using the clipboard, either in or out of their AppStream 2.0 streaming session. The value can be between 1 and 20,971,520 (20 MB), and defaults to the maximum value when unspecified.
- **File transfer** — By default, users can upload and download files between their local device and streaming session. You can limit file transfer options so that users can upload files to their streaming session only or download files to their local device only. You can also disable file transfer entirely.

> **Important**
>
> If your users require AppStream 2.0 file system redirection to access local drives and folders during their streaming sessions, you must enable both file upload and download. To use file system redirection, your users must have AppStream 2.0 client version 1.0.480 or later installed. For more information, see Enable File System Redirection for Your AppStream 2.0 Users.

- **Print to local device** — By default, users can print to their local device from within a streaming application. When they choose **Print** in the application, they can download a .pdf

file that they can print to a local printer. You can disable this option to prevent users from printing to a local device.

- **Password sign in for Active Directory** — Users can enter their Active Directory domain password to sign in to an AppStream 2.0 streaming instance that is joined to an Active Directory domain.

  You can also enable **Smart card sign in for Active Directory**. At least one authentication method must be enabled.

- **Smart card sign in for Active Directory** — Users can use a smart card reader and smart card connected to their local computer to sign in to an AppStream 2.0 streaming instance that is joined to an Active Directory domain.

  You can also enable **Password sign in for Active Directory**. At least one authentication method must be enabled.

> **Note**
>
> **Clipboard, file transfer, and print to local device settings** — These settings control only whether users can use AppStream 2.0 data transfer features. If your image provides access to a browser, network printer, or other remote resource, your users might be able to transfer data to or from their streaming session in other ways. **Authentication settings** — These settings control only the authentication method that can be used for Windows sign in to an AppStream 2.0 streaming instance (fleet or image builder). They do not control the authentication method that can be used for in-session authentication, after a user signs in to a streaming instance. For information about configuration requirements for using smart cards for Windows sign in and in-session authentication, see Smart Cards.

Application settings persistence options:

- **Enable Application Settings Persistence** — Users' application customizations and Windows settings are automatically saved after each streaming session and applied during the next session. These settings are saved to an Amazon Simple Storage Service (Amazon S3) bucket in your account, within the AWS Region in which application settings persistence is enabled.
- **Settings Group** — The settings group determines which saved application settings are used for a streaming session from this stack. If the same settings group is applied to another

stack, both stacks use the same application settings. By default, the settings group value is the name of the stack.

> **Note**
>
> For information about requirements for enabling and administering application settings persistence, see Enable Application Settings Persistence for Your AppStream 2.0 Users.

11. For **Step 7: Review**, confirm the details for the stack. To change the configuration for any section, choose **Edit** and make the needed changes. After you finish reviewing the configuration details, choose **Create**.
12. In the pricing acknowledgement dialog box, select the acknowledgement check box, and choose **Create**.
13. After the service sets up resources, the **Stacks** page appears. The status of your new stack appears as **Active** when it is ready to use.

Step 2: Provide Access to Users

After you create a stack with an associated fleet, you can provide access to users through the AppStream 2.0 user pool, SAML 2.0 [single sign-on (SSO)], or the AppStream 2.0 API. For more information, see User Pool Administration and Amazon AppStream 2.0 Integration with SAML 2.0.

> **Note**
>
> Users in the AppStream 2.0 user pool can't be assigned to stacks with fleets that are joined to an Active Directory domain.

For this getting started exercise, you can use the AppStream 2.0 user pool. This access method enables you to create and manage users by using a permanent login portal URL. To quickly test application streaming without setting up users, complete the following steps to create a temporary URL, also known as a streaming URL.

To provide access to users with a temporary URL

1. In the navigation pane, choose **Fleets**.

2. In the list of fleets, choose the fleet that is associated with the stack for which you want to create a streaming URL. Verify that the status of the fleet is **Running**.
3. In the navigation pane, choose **Stacks**. Select the stack, and then choose **Actions**, **Create Streaming URL**.
4. For **User id**, type the user ID. Choose an expiration time, which determines how long the generated URL is valid.
5. To view the user ID and URL, choose **Get URL**.
6. To copy the link to the clipboard, choose **Copy Link**.

After you provide your users with access to AppStream 2.0, they can start AppStream 2.0 streaming sessions. If you provide access through the AppStream 2.0 user pool, they must use a web browser for streaming sessions.

If you plan to use SAML 2.0 [single sign-on (SSO)] or the AppStream 2.0 API to provide access to your users, you can make the AppStream 2.0 client available to them. The AppStream 2.0 client is a native application that is designed for users who require additional functionality during their AppStream 2.0 streaming sessions. For more information, see Provide Access Through the AppStream 2.0 Client for Windows.

Resources

For more information, see the following:

- Learn how to use the AppStream 2.0 image builder to add your own applications and create images that you can stream to your users. For more information, see Tutorial: Create a Custom AppStream 2.0 Image by Using the AppStream 2.0 Console.
- Provide persistent storage for your session users by using AppStream 2.0 home folders, Google Drive, and OneDrive. For more information, see Enable and Administer Persistent Storage for Your AppStream 2.0 Users.
- Integrate your AppStream 2.0 streaming resources with your Microsoft Active Directory environment. For more information, see Using Active Directory with AppStream 2.0.
- Control who has access to your AppStream 2.0 streaming instances. For more information, see Identity and Access Management for Amazon AppStream 2.0, AppStream 2.0 User Pools and Amazon AppStream 2.0 Integration with SAML 2.0.
- Monitor your AppStream 2.0 resources by using Amazon CloudWatch. For more information, see AppStream 2.0 Metrics and Dimensions .

- Troubleshoot your AppStream 2.0 streaming experience. For more information, see Troubleshooting.

Networking and Access for Amazon AppStream 2.0

The following topics provide information about enabling users to connect to AppStream 2.0 streaming instances (fleet instances) and enabling your AppStream 2.0 fleets, image builders, and app block builders to access network resources and the internet.

Contents

- Internet Access
- Configure a VPC for AppStream 2.0
- Using Amazon S3 VPC Endpoints for AppStream 2.0 Features
- Amazon AppStream 2.0 Connections to Your VPC
- User Connections to Amazon AppStream 2.0

Internet Access

If your fleets, app block builders, and image builders require internet access, you can enable internet access in several ways. When you choose a method for enabling internet access, consider the number of users your deployment must support and your deployment goals. For example:

- If your deployment must support more than 100 concurrent users, configure a VPC with private subnets and a NAT gateway.
- If your deployment supports fewer than 100 concurrent users, you can configure a new or existing VPC with a public subnet.
- If your deployment supports fewer than 100 concurrent users and you are new to AppStream 2.0 and want to get started using the service, you can use the default VPC, public subnet, and security group.

The following sections provide more information about each of these deployment options.

- Configure a VPC with Private Subnets and a NAT Gateway (recommended) — With this configuration, you launch your fleets, app block builders, and image builders in a private subnet and configure a NAT gateway in a public subnet in your VPC. Your streaming instances are assigned a private IP address that is not directly accessible from the internet.

In addition, unlike configurations that use the **Default Internet Access** option for enabling internet access, the NAT configuration is not limited to 100 fleet instances. If your deployment must support more than 100 concurrent users, use this configuration.

You can create and configure a new VPC to use with a NAT gateway, or add a NAT gateway to an existing VPC.

- Configure a New or Existing VPC with a Public Subnet — With this configuration, you launch your fleets, app block builders, and image builders in a public subnet and enable **Default Internet Access**. When you enable this option, AppStream 2.0 uses the internet gateway in your Amazon VPC public subnet to provide the internet connection. Your streaming instances are assigned a public IP address that is directly accessible from the internet. You can create a new VPC or configure an existing one for this purpose.

  > **Note**
  >
  > When **Default Internet Access** is enabled, a maximum of 100 fleet instances is supported. If your deployment must support more than 100 concurrent users, use the NAT gateway configuration instead.

- Use the Default VPC, Public Subnet, and Security Group — If you are new to AppStream 2.0 and want to get started using the service, you can launch your fleets, app block builders, and image builders in a default public subnet and enable **Default Internet Access**. When you enable this option, AppStream 2.0 uses the internet gateway in your Amazon VPC public subnet to provide the internet connection. Your streaming instances are assigned a public IP address that is directly accessible from the internet.

  Default VPCs are available in Amazon Web Services accounts created after 2013-12-04.

  The default VPC includes a default public subnet in each Availability Zone and an internet gateway that is attached to your VPC. The VPC also includes a default security group.

  > **Note**
  >
  > When **Default Internet Access** is enabled, a maximum of 100 fleet instances is supported. If your deployment must support more than 100 concurrent users, use the NAT gateway configuration instead.

Configure a VPC for AppStream 2.0

When you set up AppStream 2.0, you must specify the virtual private cloud (VPC) and at least one subnet in which to launch your fleet instances and image builders. A VPC is a virtual network in your own logically isolated area within the Amazon Web Services Cloud. A subnet is a range of IP addresses in your VPC.

When you configure your VPC for AppStream 2.0, you can specify either public or private subnets, or a mix of both types of subnets. A public subnet has direct access to the internet through an internet gateway. A private subnet, which doesn't have a route to an internet gateway, requires a Network Address Translation (NAT) gateway or NAT instance to provide access to the internet.

Contents

- VPC Setup Recommendations
- Configure a VPC with Private Subnets and a NAT Gateway
- Configure a New or Existing VPC with a Public Subnet
- Use the Default VPC, Public Subnet, and Security Group

VPC Setup Recommendations

When you create a fleet, or launch an image builder or app block builder, you specify the VPC and one or more subnets to use. You can provide additional access control to your VPC by specifying security groups.

The following recommendations can help you configure your VPC more effectively and securely. In addition, they can help you configure an environment that supports effective fleet scaling. With effective fleet scaling, you can meet current and anticipated AppStream 2.0 user demand, while avoiding unecessary resource usage and associated costs.

Overall VPC Configuration

- Make sure that your VPC configuration can support your fleet scaling needs.

  As you develop your plan for fleet scaling, keep in mind that one user requires one fleet instance. Therefore, the size of your fleet determines the number of users who can stream concurrently. For this reason, for each instance type that you plan to use, make sure that the number of fleet instances that your VPC can support is greater than the number of anticipated concurrent users for the same instance type.

- Make sure that your AppStream 2.0 account quotas (also referred to as limits) are sufficient to support your anticipated demand. To request a quota increase, you can use the Service Quotas console at https://console.aws.amazon.com/servicequotas/. For information about default AppStream 2.0 quotas, see Amazon AppStream 2.0 Service Quotas.
- If you plan to provide your streaming instances (fleet instances, app block builder, or image builders) with access to the internet, we recommend that you configure a VPC with two private subnets for your streaming instances and a NAT gateway in a public subnet.

  The NAT gateway lets the streaming instances in your private subnets connect to the internet or other AWS services. However, it prevents the internet from initiating a connection with those instances. In addition, unlike configurations that use the **Default Internet Access** option for enabling internet access, the NAT configuration supports more than 100 fleet instances. For more information, see Configure a VPC with Private Subnets and a NAT Gateway.

Elastic Network Interfaces

- AppStream 2.0 creates as many elastic network interfaces (network interfaces) as the maximum desired capacity of your fleet. By default, the limit for network interfaces per Region is 5000.

  When planning capacity for very large deployments, for example, thousands of streaming instances, consider the number of EC2 instances that are also used in the same Region.

Subnets

- If you are configuring more than one private subnet for your VPC, configure each in a different Availability Zone. Doing so increases fault tolerance and can help prevent insufficient capacity errors. If you use two subnets in the same AZ, you might run out of IP addresses, because AppStream 2.0 will not use the second subnet.
- Make sure that the network resources required for your applications are accessible through both of your private subnets.
- Configure each of your private subnets with a subnet mask that allows for enough client IP addresses to account for the maximum number of expected concurrent users. In addition, allow for additional IP addresses to account for anticipated growth. For more information, see VPC and Subnet Sizing for IPv4.

- If you are using a VPC with NAT, configure at least one public subnet with a NAT Gateway for internet access, preferably two. Configure the public subnets in the same Availability Zones where your private subnets reside.

  To enhance fault tolerance and reduce the chance of insufficient capacity errors for large AppStream 2.0 fleet deployments, consider extending your VPC configuration into a third Availability Zone. Include a private subnet, public subnet, and NAT gateway in this additional Availability Zone.

  To configure more than two Availability Zones, you can use the UpdateFleet API action or the update-fleet AWS CLI command.

Security Groups

- Use security groups to provide additional access control to your VPC.

  Security groups that belong to your VPC let you control the network traffic between AppStream 2.0 streaming instances and network resources required by applications. These resources may include other AWS services such as Amazon RDS or Amazon FSx, license servers, database servers, file servers, and application servers.
- Make sure that the security groups provide access to the network resources that your applications require.

  For more information about configuring security groups for AppStream 2.0, see Security Groups in Amazon AppStream 2.0. For general information about security groups, see Security Groups for Your VPC in the *Amazon VPC User Guide*.

Configure a VPC with Private Subnets and a NAT Gateway

If you plan to provide your streaming instances (fleet instances, app block builders, and image builders) with access to the internet, we recommend that you configure a VPC with two private subnets for your streaming instances and a NAT gateway in a public subnet. You can create and configure a new VPC to use with a NAT gateway, or add a NAT gateway to an existing VPC. For additional VPC configuration recommendations, see VPC Setup Recommendations.

The NAT gateway lets the streaming instances in your private subnets connect to the internet or other AWS services, but prevents the internet from initiating a connection with those instances. In addition, unlike configurations that use the **Default Internet Access** option for enabling internet

access for AppStream 2.0 streaming instances, this configuration is not limited to 100 fleet instances.

For information about using NAT Gateways and this configuration, see NAT Gateways and VPC with Public and Private Subnets (NAT) in the *Amazon VPC User Guide*.

Contents

- Create and Configure a New VPC
- Add a NAT Gateway to an Existing VPC
- Enable Internet Access for Your Fleet, Image Builder, or App Block Builder

Create and Configure a New VPC

This topic describes how to use the VPC wizard to create a VPC with a public subnet and one private subnet. As part of this process, the wizard creates an internet gateway and a NAT gateway. It also creates a custom route table associated with the public subnet and updates the main route table associated with the private subnet. The NAT gateway is automatically created in the public subnet of your VPC.

After you use the wizard to create the initial VPC configuration, you'll add a second private subnet. For more information about this configuration, see VPC with Public and Private Subnets (NAT) in the *Amazon VPC User Guide*.

> **Note**
>
> If you already have a VPC, complete the steps in Add a NAT Gateway to an Existing VPC instead.

Contents

- Step 1: Allocate an Elastic IP Address
- Step 2: Create a New VPC
- Step 3: Add a Second Private Subnet
- Step 4: Verify and Name Your Subnet Route Tables

Step 1: Allocate an Elastic IP Address

Before you create your VPC, you must allocate an Elastic IP address in your AppStream 2.0 Region. You must first allocate an Elastic IP address for use in your VPC, and then associate it with your NAT gateway. For more information, see Elastic IP Addresses in the *Amazon VPC User Guide*.

> **Note**
>
> Charges may apply to Elastic IP addresses that you use. For more information, see Elastic IP Addresses on the Amazon EC2 pricing page.

Complete the following steps if you don't already have an Elastic IP address. If you want to use an existing Elastic IP address, verify that it's not currently associated with another instance or network interface.

To allocate an Elastic IP address

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, under **Network & Security**, choose **Elastic IPs**.
3. Choose **Allocate New Address**, and then choose **Allocate**.
4. Note the Elastic IP address.
5. In the upper right of the **Elastic IPs** pane, click the X icon to close the pane.

Step 2: Create a New VPC

Complete the following steps to create a new VPC with a public subnet and one private subnet.

To create a new VPC

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **VPC Dashboard**.
3. Choose **Launch VPC Wizard**.
4. In **Step 1: Select a VPC Configuration**, choose **VPC with Public and Private Subnets**, and then choose **Select**.
5. In **Step 2: VPC with Public and Private Subnets**, configure the VPC as follows:

- For **IPv4 CIDR block**, specify an IPv4 CIDR block for the VPC.
- For **IPv6 CIDR block**, keep the default value, **No IPv6 CIDR Block**.
- For **VPC name**, type a unique name for the VPC.

6. Configure the public subnet as follows:

   - For **Public subnet's IPv4 CIDR**, specify the CIDR block for the subnet.
   - For **Availability Zone**, keep the default value, **No Preference**.
   - For **Public subnet name**, type a name for the subnet; for example, `AppStream2 Public Subnet`.

7. Configure the first private subnet as follows:

   - For **Private subnet's IPv4 CIDR**, specify the CIDR block for the subnet. Make a note of the value that you specify.
   - For **Availability Zone**, select a specific zone and make a note of the zone that you select.
   - For **Private subnet name**, type a name for the subnet; for example, `AppStream2 Private Subnet1`.
   - For the remaining fields, where applicable, keep the default values.

8. For **Elastic IP Allocation ID**, click in the text box and select the value that corresponds to the Elastic IP address that you created. This address is assigned to the NAT gateway. If you don't have an Elastic IP address, create one by using the Amazon VPC console at https://console.aws.amazon.com/vpc/.

9. For **Service endpoints**, if an Amazon S3 endpoint is required for your environment, specify one. An S3 endpoint is required to provide users with access to home folders or to enable application settings persistence for your users in a private network.

   To specify an Amazon S3 endpoint, do the following:

   a. Choose **Add Endpoint**.

   b. For **Service**, select the entry in the list that ends with "s3" (the `com.amazonaws.`*`region`*`.s3` entry that corresponds to the Region in which the VPC is being created).

   c. For **Subnet**, choose **Private subnet**.

   d. For **Policy**, keep the default value, **Full Access**.

10. For **Enable DNS hostnames**, keep the default value, **Yes**.

11. For **Hardware tenancy**, keep the default value, **Default**.
12. Choose **Create VPC**.
13. Note that it takes several minutes to set up your VPC. After the VPC is created, choose **OK**.

Step 3: Add a Second Private Subnet

In the previous step (Step 2: Create a New VPC), you created a VPC with one public subnet and one private subnet. Perform the following steps to add a second private subnet. We recommend that you add a second private subnet in a different Availability Zone than your first private subnet.

1. In the navigation pane, choose **Subnets**.
2. Select the first private subnet that you created in the previous step. On the **Description** tab, below the list of subnets, make a note of the Availability Zone for this subnet.
3. On the upper left of the subnets pane, choose **Create Subnet**.
4. For **Name tag**, type a name for the private subnet; for example, `AppStream2 Private Subnet2`.
5. For **VPC**, select the VPC that you created in the previous step.
6. For **Availability Zone**, select an Availability Zone other than the one you are using for your first private subnet. Selecting a different Availability Zone increases fault tolerance and helps prevent insufficient capacity errors.
7. For **IPv4 CIDR block**, specify a unique CIDR block range for the new subnet. For example, if your first private subnet has an IPv4 CIDR block range of `10.0.1.0/24`, you could specify a CIDR block range of `10.0.2.0/24` for the new private subnet.
8. Choose **Create**.
9. After your subnet is created, choose **Close**.

Step 4: Verify and Name Your Subnet Route Tables

After you've created and configured your VPC, complete the following steps to specify a name for your route tables, and to verify that:

- The route table associated with the subnet in which your NAT gateway resides includes a route that points internet traffic to an internet gateway. This ensures that your NAT gateway can access the internet.

- The route tables associated with your private subnets are configured to point internet traffic to the NAT gateway. This enables the streaming instances in your private subnets to communicate with the internet.

1. In the navigation pane, choose **Subnets**, and select the public subnet that you created; for example, `AppStream 2.0 Public Subnet`.

   a. On the **Route Table** tab, choose the ID of the route table; for example, `rtb-12345678`.

   b. Select the route table. Under **Name**, choose the edit icon (the pencil), and type a name (for example, `appstream2-public-routetable`), and then select the check mark to save the name.

   c. With the public route table still selected, on the **Routes** tab, verify that there is one route for local traffic and another route that sends all other traffic to the internet gateway for the VPC. The following table describes these two routes:

Destination	Target	Description
Public subnet IPv4 CIDR Block (for example, 10.0.0/20)	Local	All traffic from the resources destined for IPv4 addresses within the public subnet IPv4 CIDR block is routed locally within the VPC.
Traffic destined to all other IPv4 addresses (for example, 0.0.0.0/0	Outbound (`igw-`*`ID`*)	Traffic destined for all other IPv4 addresses is routed to the internet gateway (identified by `igw-`*`ID`*) that was created by the VPC Wizard.

2. In the navigation pane, choose **Subnets**, and select the first private subnet that you created (for example, `AppStream2 Private Subnet1`).

   a. On the **Route Table** tab, choose the ID of the route table.

   b. Select the route table. Under **Name**, choose the edit icon (the pencil), and enter a name (for example, `appstream2-private-routetable`), and then choose the check mark to save the name.

   c. On the **Routes** tab, verify that the route table includes the following routes:

Destination	Target	Description
Public subnet IPv4 CIDR Block (for example, 10.0.0/20)	Local	All traffic from the resources destined for IPv4 addresses within the public subnet IPv4 CIDR block is routed locally within the VPC.
Traffic destined to all other IPv4 addresses (for example, 0.0.0.0/0	Outbound (`nat-`*`ID`*)	Traffic destined for all other IPv4 addresses is routed to the NAT gateway (identified by `nat-`*`ID`*).
Traffic destined for S3 buckets (applicable if you specified an S3 endpoint) [`pl-`*`ID`* (`com.amazonaws.`*`region`*`.s3`)]	Storage (`vpce-`*`ID`*)	Traffic destined for S3 buckets is routed to the S3 endpoint (identified by `vpce-`*`ID`*).

3. In the navigation pane, choose **Subnets**, and select the second private subnet that you created (for example, `AppStream2 Private Subnet2`).
4. On the **Route Table** tab, verify that the route table is the private route table (for example, `appstream2-private-routetable`). If the route table is different, choose **Edit** and select this route table.

Next Steps

To enable your fleet instances, app block builders, and image builders to access the internet, complete the steps in Enable Internet Access for Your Fleet, Image Builder, or App Block Builder.

Add a NAT Gateway to an Existing VPC

If you have already configured a VPC, complete the following steps to add a NAT gateway to your VPC. If you need to create a new VPC, see Create and Configure a New VPC.

To add a NAT gateway to an existing VPC

1. To create your NAT gateway, complete the steps in Creating a NAT Gateway in the *Amazon VPC User Guide*.
2. Verify that your VPC has at least one private subnet. We recommend that you specify two private subnets from different Availability Zones for high availability and fault tolerance. For information about how to create a second private subnet, see Step 3: Add a Second Private Subnet.
3. Update the route table associated with one or more of your private subnets to point internet-bound traffic to the NAT gateway. This enables the streaming instances in your private subnets to communicate with the internet. To do so, complete the steps in Updating Your Route Table in the *Amazon VPC User Guide*.

Next Steps

To enable your fleet instances, app block builders, and image builders to access the internet, complete the steps in Enable Internet Access for Your Fleet, Image Builder, or App Block Builder.

Enable Internet Access for Your Fleet, Image Builder, or App Block Builder

After your NAT gateway is available on a VPC, you can enable internet access for your fleet, image builder, and app block builder.

Enable Internet Access for Your Fleet

You can enable internet access either when you create the fleet or later.

To enable internet access at fleet creation

1. Complete the steps in Create a Fleet up to **Step 4: Configure Network**.
2. Choose a VPC with a NAT gateway.
3. If the subnet fields are empty, select a private subnet for **Subnet 1** and, optionally, another private subnet for **Subnet 2**. If you don't already have a private subnet in your VPC, you may need to create a second private subnet.
4. Continue with the steps in Create a Fleet.

To enable internet access after fleet creation by using a NAT gateway

1. In the navigation pane, choose **Fleets**.
2. Select a fleet and verify 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, optionally, another private subnet for **Subnet 2**. If you don't already have a private subnet in your VPC, you may need to create a second private subnet.
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.

Enable Internet Access for Your Image Builder

If you plan to enable internet access for your image builder, you must do so when you create the image builder.

To enable internet access for an image builder

1. Complete the steps in Launch an Image Builder to Install and Configure Streaming Applications, up to **Step 3: Configure Network**.
2. Choose the VPC with a NAT gateway.
3. If **Subnet** is empty, select a subnet.
4. Continue with the steps in Launch an Image Builder to Install and Configure Streaming Applications.

Enable Internet Access for Your App Block Builder

If you plan to enable internet access for your app block builder, you must do so when you create the app block builder.

To enable internet access for an app block builder

1. Complete the steps in the section called “Create an App Block Builder” up to **Step 2: Configure Network**.
2. Choose the VPC with a NAT gateway.

3. If **Subnet** is empty, select a subnet.
4. Continue with the steps in the section called "Create an App Block Builder".

Configure a New or Existing VPC with a Public Subnet

If you created your Amazon Web Services account after 2013-12-04, you have a default VPC in each AWS Region that includes default public subnets. However, you may want to create your own nondefault VPC or configure an existing VPC to use with AppStream 2.0. This topic describes how to configure a nondefault VPC and public subnet to use with AppStream 2.0.

After you configure your VPC and public subnet, you can provide your streaming instances (fleet instances and image builders) with access to the internet by enabling the **Default Internet Access** option. When you enable this option, 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 public subnet. An Elastic IP address is a public IPv4 address that is reachable from the internet. For this reason, we recommend that you instead use a NAT gateway to provide internet access to your AppStream 2.0 instances. In addition, when **Default Internet Access** is enabled, a maximum of 100 fleet instances is supported. If your deployment must support more than 100 concurrent users, use the NAT gateway configuration instead.

For more information, see the steps in Configure a VPC with Private Subnets and a NAT Gateway. For additional VPC configuration recommendations, see VPC Setup Recommendations.

Contents

- Step 1: Configure a VPC with a Public Subnet
- Step 2: Enable Default Internet Access Your Fleet, Image Builder, or App Block Builder

Step 1: Configure a VPC with a Public Subnet

You can configure your own non-default VPC with a public subnet by using either of the following methods:

- Create a New VPC with a Single Public Subnet
- Configure an Existing VPC

Create a New VPC with a Single Public Subnet

When you use the VPC wizard to create a new VPC, the wizard creates an internet gateway and a custom route table that is associated with the public subnet. The route table routes all traffic destined for an address outside the VPC to the internet gateway. For more information about this configuration, see VPC with a Single Public Subnet in the *Amazon VPC User Guide*.

1. Complete the steps in Step 1: Create the VPC in the *Amazon VPC User Guide* to create your VPC.
2. To enable your fleet instances and image builders to access the internet, complete the steps in Step 2: Enable Default Internet Access Your Fleet, Image Builder, or App Block Builder.

Configure an Existing VPC

If you want to use an existing VPC that does not have a public subnet, you can add a new public subnet. In addition to a public subnet, you must also have an internet gateway attached to your VPC and a route table that routes all traffic destined for an address outside the VPC to the internet gateway. To configure these components, complete the following steps.

1. To add a public subnet, complete the steps in Creating a Subnet in Your VPC. Use the existing VPC that you plan to use with AppStream 2.0.

   If your VPC is configured to support IPv6 addressing, the **IPv6 CIDR block** list displays. Select **Don't assign Ipv6**.
2. To create and attach an internet gateway to your VPC, complete the steps in Creating and Attaching an Internet Gateway.
3. To configure your subnet to route internet traffic through the internet gateway, complete the steps in Creating a Custom Route Table. In step 5, for **Destination**, use IPv4 format (`0.0.0.0/0`).
4. To enable your fleet instances and image builders to access the internet, complete the steps in Step 2: Enable Default Internet Access Your Fleet, Image Builder, or App Block Builder.

Step 2: Enable Default Internet Access Your Fleet, Image Builder, or App Block Builder

After you configure a VPC that has a public subnet, you can enable the **Default Internet Access** option for your fleet and image builder.

Enable Default Internet Access for a Fleet

You can enable the **Default Internet Access** option when you create the fleet, or later.

> **Note**
>
> For fleet instances that have the **Default Internet Access** option enabled, the limit is 100.

To enable internet access at fleet creation

1. Complete the steps in Create a Fleet up to **Step 4: Configure Network**.
2. Select the **Default Internet Access** check box.
3. If the subnet fields are empty, select a subnet for **Subnet 1** and, optionally, **Subnet 2**.
4. Continue with the steps in Create a Fleet.

To enable internet access after fleet creation

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

You can test internet connectivity by starting your fleet, creating a stack, associating the fleet with a stack, and browsing the internet within a streaming session for stack. For more information, see Create an AppStream 2.0 Fleet and Stack.

Enable Default Internet Access for an Image Builder

After you configure a VPC that has a public subnet, you can enable the **Default Internet Access** option for your image builder. You can do so when you create the image builder.

To enable internet access for an image builder

1. Complete the steps in Launch an Image Builder to Install and Configure Streaming Applications up to **Step 3: Configure Network**.
2. Select the **Default Internet Access** check box.

3. If **Subnet 1** is empty, select a subnet.
4. Continue with the steps in Launch an Image Builder to Install and Configure Streaming Applications.

Enable Default Internet Access for an App Block Builder

After you configure a VPC that has a public subnet, you can enable the **Default Internet Access** option for your app block builder. You can do so when you create the app block builder.

To enable internet access for an app block builder

1. Follow the steps in the section called "Create an App Block Builder", up to **Step 2: Configure Network**.
2. Select the **Default Internet Access** check box.
3. If **Subnet** is empty, select a subnet.
4. Continue with the steps in the section called "Create an App Block Builder".

Use the Default VPC, Public Subnet, and Security Group

Your Amazon Web Services account, if it was created after 2013-12-04, has a default VPC in each AWS Region. The default VPC includes a default public subnet in each Availability Zone and an internet gateway that is attached to your VPC. The VPC also includes a default security group. If you are new to AppStream 2.0 and want to get started using the service, you can keep the default VPC and security group selected when you create a fleet, create an app block builder, or launch an image builder. Then, you can select at least one default subnet.

> **Note**
>
> If your Amazon Web Services account was created before 2013-12-04, you must create a new VPC or configure an existing one to use with AppStream 2.0. We recommend that you manually configure a VPC with two private subnets for your fleets, app block builders, and image builders and a NAT gateway in a public subnet. For more information, see Configure a VPC with Private Subnets and a NAT Gateway. Alternatively, you can configure a non-default VPC with a public subnet. For more information, see Configure a New or Existing VPC with a Public Subnet.

To use the default VPC, subnet, and security group for a fleet

1. Complete the steps in Create a Fleet up to **Step 4: Configure Network**.
2. In **Step 4: Configure Network**, do the following:

   - To enable your fleet instances to access the internet, select the **Default Internet Access** check box.

     > **Note**
     >
     > For fleet instances that have the **Default Internet Access** option enabled, the limit is 100.

   - For **VPC**, choose the default VPC for your AWS Region.

     The default VPC name uses the following format: `vpc-`*`vpc-id`* `(No_default_value_Name)`.

   - For **Subnet 1**, choose a default public subnet and make a note of the Availability Zone.

     The default subnet names use the following format: `subnet-`*`subnet-id`* `|` `(`*`IPv4 CIDR block`*`) | Default in` *`availability-zone`*.

   - Optionally, for **Subnet 2**, choose a default subnet in a different Availability Zone.
   - For **Security groups**, select the default security group.

     The default security group name uses the following format: `sg-`*`security-group-id`*`-default`

3. Continue with the steps in Create a Fleet.

Complete the following steps to use the default VPC, subnet, and security group for an image builder.

To use the default VPC, subnet, and security group for an image builder

1. Follow the steps in Launch an Image Builder to Install and Configure Streaming Applications up to **Step 3: Configure Network**.
2. In **Step 4: Configure Network**, do the following:

- To enable your image builder to access the internet, select the **Default Internet Access** check box.
- For **VPC**, choose the default VPC for your AWS Region.

  The default VPC name uses the following format: `vpc-`*`vpc-id`* `(No_default_value_Name)`.
- For **Subnet 1**, choose a default public subnet.

  The default subnet names use the following format: `subnet-`*`subnet-id`* `|` `(`*`IPv4 CIDR block`*`) | Default in` *`availability-zone`*.
- For **Security groups**, select the default security group.

  The default security group name uses the following format: `sg-`*`security-group-id`*`-default`

3. Continue with the steps in Launch an Image Builder to Install and Configure Streaming Applications.

Complete the following steps to use the default VPC, subnet, and security group for an app block builder.

To use the default VPC, subnet, and security group for an app block builder

1. Follow the steps in the section called "Create an App Block Builder", up to **Step 2: Configure Network**.
2. In **Step 2: Configure Network**, do the following:

   - To enable your image builder to access the internet, select the **Default Internet Access** check box.
   - For **VPC**, choose the default VPC for your AWS Region.

     The default VPC name uses the following format: `vpc-`*`vpc-id`* `(No_default_value_Name)`.
   - For **Subnet 1**, choose a default public subnet.

     The default subnet names use the following format: `subnet-`*`subnet-id`* `|` `(`*`IPv4 CIDR block`*`) | Default in` *`availability-zone`*.

- For **Security groups**, select the default security group.

  The default security group name uses the following format: `sg-`*`security-group-id`*`-default`

3. Continue with the steps in the section called "Create an App Block Builder".

Using Amazon S3 VPC Endpoints for AppStream 2.0 Features

When you enable Application Settings Persistence or Home folders on a stack, AppStream 2.0 uses the VPC you specify for your fleet to provide access to Amazon Simple Storage Service (Amazon S3) buckets. For Elastic fleets, AppStream 2.0 will use the VPC to access the Amazon S3 bucket containing applications assigned to the fleet's app block. To enable AppStream 2.0 access to your private S3 endpoint, attach the following custom policy 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-S3-buckets",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:sts::account-id-without-hyphens:assumed-role/
AmazonAppStreamServiceAccess/AppStream2.0"
            },
            "Action": [
                "s3:ListBucket",
                "s3:GetObject",
                "s3:PutObject",
                "s3:DeleteObject",
                "s3:GetObjectVersion",
                "s3:DeleteObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::appstream2-36fb080bb8-*",
                "arn:aws:s3:::appstream-app-settings-*",
                "arn:aws:s3:::appstream-logs-*"
            ]
        },
```

```
        {
            "Sid": "Allow-AppStream-ElasticFleetstoRetrieveObjects",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::bucket-with-application-or-app-block-objects/*",
            "Condition": {
                "StringEquals": {
                    "aws:PrincipalServiceName": "appstream.amazonaws.com"
                }
            }
        }
    ]
}
```

Amazon AppStream 2.0 Connections to Your VPC

To enable AppStream 2.0 connectivity to network resources and the internet, configure your streaming instances as follows.

Network Interfaces

Each AppStream 2.0 streaming instance has the following network interfaces:

- The customer network interface provides connectivity to the resources within your VPC, as well as the internet, and is used to join the streaming instance to your directory.
- The management network interface is connected to a secure AppStream 2.0 management network. It is used for interactive streaming of the streaming instance to a user's device, and to allow AppStream 2.0 to manage the streaming instance.

AppStream 2.0 selects the IP address for the management network interface from the following private IP address range: 198.19.0.0/16. Do not use this range for your VPC CIDR or peer your VPC with another VPC with this range, as this might create a conflict and cause streaming instances to be unreachable. Also, do not modify or delete any of the network interfaces attached to a streaming instance, as this might also cause the streaming instance to become unreachable.

Management Network Interface IP Address Range and Ports

The management network interface IP address range is 198.19.0.0/16. The following ports must be open on the management network interface of all streaming instances:

- Inbound TCP on port 8300. This is used for establishment of the streaming connection.
- Inbound TCP on ports 8000 and 8443. These are used for management of the streaming instance by AppStream 2.0.
- Inbound UDP on port 8300. This is used for establishment of the streaming connection over UDP.

Limit the inbound range on the management network interface to 198.19.0.0/16.

Under normal circumstances, AppStream 2.0 correctly configures these ports for your streaming instances. If any security or firewall software is installed on a streaming instance that blocks any of these ports, the streaming instance may not function correctly or may be unreachable.

Do not disable IPv6. If you disable IPv6, AppStream 2.0 will not function correctly. For information about configuring IPv6 for Windows, see Guidance for configuring IPv6 in Windows for advanced users.

> **Note**
>
> AppStream 2.0 relies on the DNS servers within your VPC to return a non-existent domain (NXDOMAIN) response for local domain names that don't exist. This enables the AppStream 2.0-managed network interface to communicate with the management servers.
> When you create a directory with Simple AD, AWS Directory Service creates two domain controllers that also function as DNS servers on your behalf. Because the domain controllers don't provide the NXDOMAIN response, they can't be used with AppStream 2.0.

Customer Network Interface Ports

- For internet connectivity, the following ports must be open to all destinations. If you are using a modified or custom security group, you need to add the required rules manually. For more information, see Security Group Rules in the *Amazon VPC User Guide*.
  - TCP 80 (HTTP)
  - TCP 443 (HTTPS)
  - UDP 8433
- If you join your streaming instances to a directory, the following ports must be open between your AppStream 2.0 VPC and your directory controllers.
  - TCP/UDP 53 - DNS

- TCP/UDP 88 - Kerberos authentication
- UDP 123 - NTP
- TCP 135 - RPC
- UDP 137-138 - Netlogon
- TCP 139 - Netlogon
- TCP/UDP 389 - LDAP
- TCP/UDP 445 - SMB
- TCP 1024-65535 - Dynamic ports for RPC

  For a complete list of ports, see Active Directory and Active Directory Domain Services Port Requirements in the Microsoft documentation.

- All streaming instances require that port 80 (HTTP) be open to IP address `169.254.169.254` to allow access to the EC2 metadata service. The IP address range `169.254.0.0/16` is reserved for AppStream 2.0 service usage for management traffic. Failure to exclude this range might result in streaming issues.

User Connections to Amazon AppStream 2.0

Users can connect to AppStream 2.0 streaming instances through the default public internet endpoint, or by using an interface VPC endpoint (interface endpoint) that you create in your virtual private cloud (VPC). For more information, see Creating and Streaming from Interface VPC Endpoints.

By default, AppStream 2.0 is configured to route streaming connections over the public internet. Internet connectivity is required to authenticate users and deliver the web assets that AppStream 2.0 requires to function. To allow this traffic, you must allow the domains listed in Allowed Domains.

> **Note**
>
> For user authentication, AppStream 2.0 supports user pools, Security Assertion Markup Language 2.0 (SAML 2.0), and the CreateStreamingURL API action. For more information, see User Authentication.

The following topics provide information about how to enable user connections to AppStream 2.0.

Contents

- Bandwidth Recommendations
- IP Address and Port Requirements for AppStream 2.0 User Devices
- Allowed Domains

Bandwidth Recommendations

To optimize the performance of AppStream 2.0, make sure that your network bandwidth and latency can sustain your users' needs.

AppStream 2.0 uses NICE Desktop Cloud Visualization (DCV) to enable your users to securely access and stream your applications over varying network conditions. To help reduce bandwidth consumption, NICE DCV uses H.264-based video compression and encoding. During streaming sessions, the visual output of applications is compressed and streamed to your users as an AES-256 encrypted pixel stream over HTTPS. After the stream is received, it is decrypted and output to your users' local screen. When your users interact with their streaming applications, the NICE DCV protocol captures their input and sends it back to their streaming applications over HTTPS.

Network conditions are constantly measured during this process and information is sent back to AppStream 2.0. AppStream 2.0 dynamically responds to changing network conditions by changing the video and audio encoding in real time to produce a high-quality stream for a wide variety of applications and network conditions.

The recommended bandwidth and latency for AppStream 2.0 streaming sessions depends on the workload. For example, a user who works with graphic-intensive applications to perform computer-aided design tasks will require more bandwidth and lower latency than a user who works with business productivity applications to write documents.

The following table provides guidance on the recommended network bandwidth and latency for AppStream 2.0 streaming sessions based on common workloads.

For each workload, the bandwidth recommendation is based on what an individual user might require at a specific point in time. The recommendation does not reflect the bandwidth required for sustained throughput. When only a few pixels change on the screen during a streaming session, the sustained throughput is much lower. Although users who have less bandwidth available can still stream their applications, the frame rate or image quality may not be optimal.

Workload	Description	Bandwidth recommended per user	Recommended maximum roundtrip latency
Line of business applications	Document writing applications, database analysis utilities	2 Mbps	< 150 ms
Graphics applications	Computer-aided design and modeling applications, photo and video editing	5 Mbps	< 100 ms
High fidelity	High-fidelity datasets or maps across multiple monitors	10 Mbps	< 50 ms

IP Address and Port Requirements for AppStream 2.0 User Devices

AppStream 2.0 users' devices require outbound access on port 443 (TCP) and port 8433 (UDP) when using the internet endpoints, and if you are using DNS servers for domain name resolution, port 53 (UDP).

- Port 443 is used for HTTPS communication between AppStream 2.0 users' devices and streaming instances when using the internet endpoints. Typically, when end users browse the web during streaming sessions, the web browser randomly selects a source port in the high range for streaming traffic. You must ensure that return traffic to this port is allowed.
- Port 8433 is used for UDP HTTPS communication between AppStream 2.0 users' devices and streaming instances when using the internet endpoints. This is currently only supported in the Windows native client. UDP is not supported if you are using VPC endpoints.

> **Note**
>
> Streaming through interface VPC endpoints requires additional ports. For more information, see Creating and Streaming from Interface VPC Endpoints.

- Port 53 is used for communication between AppStream 2.0 users' devices and your DNS servers. The port must be open to the IP addresses for your DNS servers so that public domain names can be resolved. This port is optional if you are not using DNS servers for domain name resolution.

Allowed Domains

For AppStream 2.0 users to access streaming instances, you must allow the following domain on the network from which users initiate access to the streaming instances.

- Session Gateway: *.amazonappstream.com

One or more of the following domains must be allowed to enable user authentication. You must allow the domains and subdomains that correspond to the Regions where AppStream 2.0 is deployed.

Region	Domain
US East (N. Virginia)	*.appstream2.us-east-1.aws.amazon.com
US East (Ohio)	*.appstream2.us-east-2.aws.amazon.com
US West (Oregon)	*.appstream2.us-west-2.aws.amazon.com
Asia Pacific (Mumbai)	*.appstream2.ap-south-1.aws.amazon.com
Asia Pacific (Seoul)	*.appstream2.ap-northeast-2.aws.amazon.com
Asia Pacific (Singapore)	*.appstream2.ap-southeast-1.aws.amazon.com
Asia Pacific (Sydney)	*.appstream2.ap-southeast-2.aws.amazon.com
Asia Pacific (Tokyo)	*.appstream2.ap-northeast-1.aws.amazon.com

Region	Domain
Canada (Central)	*.appstream2.ca-central-1.aws.amazon.com
Europe (Frankfurt)	*.appstream2.eu-central-1.aws.amazon.com
Europe (London)	*.appstream2.eu-west-2.aws.amazon.com
Europe (Ireland)	*.appstream2.eu-west-1.aws.amazon.com
AWS GovCloud (US-East)	*.appstream2.us-gov-east-1.amazonaws-us-gov.com
AWS GovCloud (US-West)	*.appstream2.us-gov-west-1.amazonaws-us-gov.com
South America (São Paulo)	*.appstream2.sa-east-1.aws.amazon.com

> **Note**
>
> If your users use a network proxy to access streaming instances, disable any proxy caching for the user auth domains in the table and the session gateway, *.amazonappstream.com.

AWS publishes its current IP address ranges, including the ranges that the Session Gateway and CloudFront domains may resolve to, in JSON format. For information about how to download the .json file and view the current ranges, see AWS IP Address Ranges in the Amazon Web Services General Reference. Or, if you are using AWS Tools for Windows PowerShell, you can access the same information by using the `Get-AWSPublicIpAddressRange` cmdlet. For more information, see Querying the Public IP Address Ranges for AWS.

For AppStream 2.0 users that are accessing Elastic fleets, you must allow access to the domain for the Amazon Simple Storage Service (S3) bucket that contains the application icon.

> **Note**
>
> If your S3 bucket has a "." character in the name, the domain used is https://s3.<AWS Region>.amazonaws.com. If your S3 bucket does not have a "." character in the name, the domain used is https://<*bucket name*>.s3.<*AWS Region*>.amazonaws.com.

Image Builders

Amazon AppStream 2.0 uses EC2 instances to stream applications. You launch instances from base images, called *image builders*, which AppStream 2.0 provides. To create your own custom image, you connect to an image builder instance, install and configure your applications for streaming, and then create your image by creating a snapshot of the image builder instance.

When you launch an image builder, you choose:

- An instance type — AppStream 2.0 provides different instance types with various compute, memory, and graphics configurations. The instance type must align with the instance family you need. For more information, see AppStream 2.0 Instance Families.
- An operating system — AppStream 2.0 provides the following operating systems:
  - Windows Server 2012 R2
  - Windows Server 2016 Base
  - Windows Server 2019 Base
  - Windows Server 2022 Base
  - Amazon Linux 2
- The subnet and security groups to use — Make sure that the subnet and security groups provide access to the network resources that your applications require. Typical network resources required by applications may include licensing servers, database servers, file servers, and application servers.

Contents

- Launch an Image Builder to Install and Configure Streaming Applications
- Connect to an Image Builder
- Image Builder Actions
- Instance Metadata for AppStream 2.0 Image Builders
- Install AMD Driver on Graphics Design Instances
- AppStream 2.0 Base Image and Managed Image Update Release Notes

Launch an Image Builder to Install and Configure Streaming Applications

To install and configure applications to stream to your users, you start by launching an image builder instance as described in the following procedure.

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

To launch an image builder

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. You can launch the image builder in the following ways:
   - If a welcome screen appears displaying two options (**Try it now** and **Get started**), choose **Get started**, **Custom set up**.

     For information about these two options, see Amazon AppStream 2.0 FAQs.
   - If a welcome screen does not appear, choose **Quick links** in the left navigation pane, then **Custom set up**.
   - Alternatively, choose **Images** in the left navigation pane, then the **Image Builder** tab, **Launch Image Builder**.
3. For **Step 1: Choose Image**, choose a base image. If you are launching the image builder for the first time, you can use one of the latest base images released by AWS (selected by default). For a list of the latest versions of base images released by AWS, see AppStream 2.0 Base Image and Managed Image Update Release Notes. If you have already created images, or you want to update applications in an existing image, you can select one of your existing images. Be sure to select an image that aligns with the instance family that you need. For more information, see AppStream 2.0 Instance Families.

   Choose **Next**.
4. For **Step 2: Configure Image Builder**, configure the image builder by doing the following:
   - **Name**: Type a unique name identifier for the image builder.

- **Display name (optional)**: Type a name to display for the image builder (maximum of 100 characters).
- **Tags (optional)**: Choose **Add Tag**, and type the key and value for the tag. To add more tags, repeat this step. For more information, see Tagging Your Amazon AppStream 2.0 Resources.
- **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. For more information, see AppStream 2.0 Instance Families.
- **VPC Endpoints (Advanced)**: You can create an interface VPC endpoint (interface endpoint), in your virtual private cloud (VPC). To start creating the interface endpoint, select **Create VPC Endpoint**. Selecting this link opens the VPC console. To finish creating the endpoint, follow steps 3 through 6 in *To create an interface endpoint*, in Creating and Streaming from Interface VPC Endpoints.

  After you create the interface endpoint, you can use it to keep streaming traffic within your VPC.
- **AppStream 2.0 Agent**: This section displays only if you are not using the latest base image from AWS or a custom image that uses the latest version of the agent.

  The AppStream 2.0 agent software runs on your streaming instances, enabling your users to connect to and stream their applications. Starting December 7, 2017, your streaming instances can be automatically updated with the latest AppStream 2.0 agent software. This capability helps to ensure that your image builder includes the latest features, performance improvements, and security updates that are available from AWS.

  You can enable automatic updates of the AppStream 2.0 agent by creating a new image from any base image published by AWS on or after December 7, 2017. If the image that you are launching your image builder from doesn't use the latest version of the AppStream 2.0 agent, we recommend that you select the option to launch your image builder with the latest agent.
- **IAM role (Advanced)**: When you apply an IAM role from your account to an AppStream 2.0 image builder, you can make AWS API requests from the image builder instance without manually managing AWS credentials. To apply an IAM role to the image builder, do either of the following:
  - To use an existing IAM role in your Amazon Web Services account, choose the role that you want to use from the **IAM role** list. The role must must be accessible from the

image builder. For more information, see Configuring an Existing IAM Role to Use With AppStream 2.0 Streaming Instances.

- To create a new IAM role, choose **Create new IAM role** and follow the steps in How to Create an IAM Role to Use With AppStream 2.0 Streaming Instances.

5. Choose **Next**.
6. For **Step 3: Configure Network**, do the following:

   - To add internet access for the image builder in a VPC with a public subnet, choose **Default Internet Access**. If you are providing internet access by using a NAT gateway, leave **Default Internet Access** unselected. For more information, see Internet Access.
   - For **VPC** and **Subnet 1**, choose a VPC and at least one subnet. For increased fault tolerance, we recommend that you choose two subnets in different Availability Zones. For more information, see Configure a VPC with Private Subnets and a NAT Gateway.

     If you don't have your own VPC and subnet, you can use the default VPC or create your own. To create your own, choose the **Create a new VPC** and **Create new subnet** links to create them. Choosing these links opens the Amazon VPC console. After you create your VPC and subnets, return to the AppStream 2.0 console and choose the refresh icon to the left of the **Create a new VPC** and **Create new subnet** links to display them in the list. For more information, see Configure a VPC for AppStream 2.0.
   - For **Security group(s)**, choose up to five security groups to associate with this image builder. If you don't have your own security group and you don't want to use the default security group, choose the **Create new security group** link to create one. After you create your subnets in the Amazon VPC console, return to the AppStream 2.0 console and choose the refresh icon to the left of the **Create new security group** link to display them in the list. For more information, see Security Groups in Amazon AppStream 2.0.
7. For **Active Directory Domain (Optional)**, expand this section to choose the Active Directory configuration and organizational unit in which to place your streaming instance computer objects. Ensure that the selected network access settings enable DNS resolvability and communication with your directory. For more information, see Using Active Directory with AppStream 2.0.
8. Choose **Review** and confirm the details for the image builder. To change the configuration for any section, choose **Edit** and make the needed changes.
9. After you finish reviewing the configuration details, choose **Launch**.

> **Note**
>
> If an error message notifies you that you don't have sufficient limits (quotas) to create the image builder, submit a limit increase request through the Service Quotas console at https://console.aws.amazon.com/servicequotas/. For more information, see Requesting a quota increase in the *Service Quotas User Guide*.

10. During the image builder creation process, the status of the image builder displays as **Pending** while AppStream 2.0 prepares the necessary resources. Click the **Refresh** icon periodically to update the image builder status. After the status changes to **Running**, the image builder is ready to use and you can create a custom image.

Next Steps

Next, install and configure your applications for streaming, and then create an image by creating a snapshot of the image builder instance. For more information, see Tutorial: Create a Custom AppStream 2.0 Image by Using the AppStream 2.0 Console.

Connect to an Image Builder

You can connect to an image builder by doing either of the following:

- Using the AppStream 2.0 console (for web connections only)
- Creating a streaming URL (for web or AppStream 2.0 client connections)

> **Note**
>
> If the image builder that you want to connect to is joined to an Active Directory domain and your organization requires smart card sign in, you must create a streaming URL and use the AppStream 2.0 client for the connection.

Contents

- AppStream 2.0 Console (Web Connection)
- Streaming URL (AppStream 2.0 Client or Web Connection)

AppStream 2.0 Console (Web Connection)

To use the AppStream 2.0 console to connect to an image builder through a web browser, complete the following steps.

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. In the left navigation pane, choose **Images**, **Image Builder**.
3. In the list of image builders, choose the image builder to which you want to connect. Verify that the status of the image builder is **Running**, and choose **Connect**.

   For this step to work, you might need to configure your browser to allow pop-ups from https://stream.<aws-region>.amazonappstream.com/.
4. Log in to the image builder by doing either of the following:

   - If your image builder is Windows-based and not joined to an Active Directory domain, on the **Local User** tab, choose one of the following:
     - **Administrator** — Choose **Administrator** to install your applications on the image builder and create an image, or to perform any other tasks that require local administrator permissions.
     - **Template User** — Choose **Template User** to create default application and Windows settings.
     - **Test User** — Choose **Test User** to open your applications and verify their settings.
   - If your image builder is Windows-based and joined to an Active Directory domain and you require access to resources that are managed by Active Directory to install your applications, choose the **Directory User** tab, and enter the credentials for a domain account that has local administrator permissions on the image builder.

     > **Note**
     >
     > Smart card sign in isn't supported for connections through a web browser. Instead, you must create a streaming URL and use the AppStream 2.0 client. For information about smart card sign in, see Smart Cards.
   - If your image builder is Linux-based, you are automatically logged in as

     the **ImageBuilderAdmin** user in the Amazon Linux GNOME desktop and have root admin privileges.

Streaming URL (AppStream 2.0 Client or Web Connection)

You can create a streaming URL to connect to an image builder through a web browser or the AppStream 2.0 client. Unlike a streaming URL that you create to enable user access to a fleet instance, which is valid for a maximum of seven days, by default, a streaming URL that you create to access an image builder expires after one hour. To set a different expiration time, you must generate the streaming URL by using the CreateStreamingURL API action.

If the image builder that you want to connect to is joined to an Active Directory domain and your organization requires smart card sign in, you must create a streaming URL and use the AppStream 2.0 client for the connection.

> **Note**
>
> Native application mode is not supported for AppStream 2.0 client connections to image builders. If you use the AppStream 2.0 client to connect to an image builder and the **Start in native application mode** check box is selected, an AppStream 2.0 error notification displays, stating that your session was switched to classic mode.

You can create a streaming URL in any of the following ways:

- AppStream 2.0 console
- The CreateStreamingURL API action
- The create-streaming-url AWS CLI command

To create a streaming URL and connect to the image builder by using the AppStream 2.0 console, complete the steps in the following procedure.

To create a streaming URL and connect to the image builder by using the AppStream 2.0 console

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. In the navigation pane, choose **Images**, **Image Builder**.
3. In the list of image builders, choose the image builder to which you want to connect. Verify that the status of the image builder is **Running**.
4. Choose **Actions**, **Create streaming URL**.

5. Do one of the following:

   - To save the streaming URL to connect to the image builder later, choose **Copy Link** to copy the URL, then save it to an accessible location.
   - To connect to the image builder through the AppStream 2.0 client, choose **Launch in Client**. When you choose this option, the AppStream 2.0 client sign-in page is prepopulated with the streaming URL.
   - To connect to the image builder through a web browser, choose **Launch in Browser**. When you choose this option, a web browser opens with the address bar prepopulated with the streaming URL.

6. After you create the streaming URL and connect to the image builder, log in to the image builder by doing either of the following:

   - If your image builder is Windows-based and not joined to an Active Directory domain, on the **Local User** tab, choose one of the following:
     - **Administrator** — Choose **Administrator** to install your applications on the image builder and create an image, or to perform any other tasks that require local administrator permissions.
     - **Template User** — Choose **Template User** to create default application and Windows settings.
     - **Test User** — Choose **Test User** to open your applications and verify their settings.
   - If your image builder is Windows-based and joined to an Active Directory domain and you require access to resources that are managed by Active Directory to install your applications, choose the **Directory User** tab, and enter the credentials for a domain account that has local administrator permissions on the image builder.

     > **Note**
     >
     > If you're using the AppStream 2.0 client, you can enter either your Active Directory domain password and choose **Password sign in**, or select **Choose a smart card** and provide your smart card PIN when prompted.
     > If you're using a web browser, you must enter your Active Directory domain password. Smart card sign in is supported only for AppStream 2.0 client connections to streaming instances.

   - If your image builder is Linux-based, you are automatically logged in as

the **ImageBuilderAdmin** user in the Amazon Linux GNOME desktop and have root admin privileges.

Image Builder Actions

You can perform the following actions 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**
- **Updating**

- **Pending Qualification**

Instance Metadata for AppStream 2.0 Image Builders

AppStream 2.0 image builder instances have instance metadata available through Windows environment variables. You can use the following environment variables in your applications and scripts to modify your environment based on the image builder instance details.

Environment Variable	Context	Description
AppStream_Image_Arn	Machine	The ARN of the image that was used to create the streaming instance.
AppStream_Instance_Type	Machine	The instance type of the streaming instance. For example, **`stream.standard.medium`**.
AppStream_Resource_Type	Machine	The type of AppStream 2.0 resource. The value is either **`fleet`** or **`imagebuilder`**.
AppStream_Resource_Name	Machine	The name of the image builder.

On Linux image builders, environment variables are exported through the script at **/etc/profile.d/appstream_system_vars.sh**. To access the environment variables, you can explicitly source this file in your application.

Install AMD Driver on Graphics Design Instances

If you need to update the AMD driver on your Windows Image Builder that is using a Graphics Design instance, you can either use the latest AppStream 2.0 Graphics Design base images, or download the AMD driver and install it on your Image Builder. If you need to update the AMD driver for an existing image of the Graphics Design instance family, you can use managed AppStream 2.0 image updates. For more information, see the section called "Update an Image by Using Managed AppStream 2.0 Image Updates".

The AMD driver download is available to AWS customers only. By downloading, you agree to use the downloaded software only to build images for use with AppStream 2.0 Graphics Design instances using AMD FirePro S7150x2 Server GPU hardware. Upon installation of the software, you are bound by the terms of the AMD Software End User License Agreements.

The latest AMD driver version for Graphics Design instances is version 24.20.13028.5012.

Before you begin, make sure that you meet the following prerequisites:

- Configure default credentials for the AWS Tools for Windows PowerShell on your Windows instance. For more information, see Getting Started with the AWS Tools for Windows PowerShell.
- IAM users must have the permissions granted by the **AmazonS3ReadOnlyAccess** policy.

To install the AMD driver on your Image Builder

1. Connect to your Windows Image Builder instance and open a PowerShell window as an Administrator.
2. Download the drivers from Amazon S3 to your desktop using the following PowerShell commands:

```
$Bucket = "appstream2-driver-patches"
$LocalPath = "$home\Desktop\AMD"
$Objects = Get-S3Object -BucketName $Bucket -Region us-east-1
foreach ($Object in $Objects) {
   $LocalFileName = $Object.Key
   if ($LocalFileName -ne '' -and $Object.Size -ne 0) {
     $LocalFilePath = Join-Path $LocalPath $LocalFileName
     Copy-S3Object -BucketName $Bucket -Key $Object.Key -LocalFile $LocalFilePath -
Region us-east-1
   }
 }
```

3. Unzip the downloaded driver file and run the installer using the following PowerShell commands:

```
Expand-Archive $LocalFilePath -DestinationPath $home\Desktop -Verbose
$Driverdir = Get-ChildItem $home\Desktop\ -Directory -Filter "*210426a-366782C*"
```

```
Write-Host $Driverdir
pnputil /add-driver $home\Desktop\$Driverdir\Packages\Drivers\Display\WT6A_INF
\*inf /install
```

4. Follow the instructions to install the driver and reboot your instance as required.
5. To verify that the GPU is working properly, check **Device Manager**. You should see **AMD MxGPU** listed as a display adapter, with the latest driver version.

AppStream 2.0 Base Image and Managed Image Update Release Notes

Amazon AppStream 2.0 provides base images to help you create images that include your own applications. Base images are Amazon Machine Images (AMIs) that contain software configurations specific to the operating system. For AppStream 2.0, each base image includes the AppStream 2.0 agent and the latest version of one of the following operating systems:

- Windows Server 2012 R2 — Available on the following image types: Base, Graphics Design, Graphics G4dn, Graphics Pro, and Sample apps

> **Note**
>
> Public base images powered by Windows Server 2012 R2 will no longer be available from AWS after 10/10/2023 due to Microsoft End-of-Support for Windows Server 2012 R2.

- Windows Server 2016 Base — Available on the following image types: Base, Graphics Design, Graphics G4dn, Graphics Pro, and Graphics G5
- Windows Server 2019 Base — Available on the following image types: Base, Graphics Design, Graphics G4dn, Graphics Pro, and Graphics G5
- Windows Server 2022 Base — Available on the following image types: Base, Graphics G4dn, and Graphics G5
- Amazon Linux 2 – Available on the following image types: Base, Graphics G4dn, and Graphics Pro

After you create your own image that includes your own applications, you are responsible for installing and maintaining the updates for the operating system, your applications, and their dependencies. AppStream 2.0 provides an automated way to update your image using managed

AppStream 2.0 image updates. With managed image updates, you select the image that you want to update. AppStream 2.0 creates an image builder in the same AWS account and Region to install the updates and create the new image. After the new image is created, you can test it on a pre-production fleet before updating your production fleets or sharing the image with other AWS accounts. For more information, see "Keep Your AppStream 2.0 Image Up-to-Date" in Administer Your Amazon AppStream 2.0 Images.

For information about the latest AppStream 2.0 agent, see AppStream 2.0 Agent Release Notes.

The following table lists the latest released images.

Image type	Image name
Base	• AppStream-WinServer2016-01-26-2024 • AppStream-WinServer2019-01-26-2024 • AppStream-WinServer2022-01-26-2024 • AppStream-AmazonLinux2-11-13-2023
Graphics Design	• AppStream-Graphics-Design-WinServer2016-01-26-2024 • AppStream-Graphics-Design-WinServer2019-01-26-2024
Graphics G4dn	• AppStream-Graphics-G4dn-WinServer2016-01-26-2024 • AppStream-Graphics-G4dn-WinServer2019-01-26-2024 • AppStream-Graphics-G4dn-WinServer2022-01-26-2024 • AppStream-Graphics-G4dn-AmazonLinux2-11-13-2023
Graphics G5	• AppStream-Graphics-G5-WinServer2016-01-26-2024 • AppStream-Graphics-G5-WinServer2019-01-26-2024 • AppStream-Graphics-G5-WinServer2022-01-26-2024 • AppStream-Graphics-G5-AmazonLinux2-11-13-2023

Image type	Image name
Graphics Pro	• AppStream-Graphics-Pro-WinServer2016-01-26-2024 • AppStream-Graphics-Pro-WinServer2019-01-26-2024 • AppStream-Graphics-Pro-AmazonLinux2-11-13-2023
Sample apps	Amazon-AppStream2-Sample-Image-03-29-2023 For information about how to access this base image, see Get Started with Amazon AppStream 2.0: Set Up With Sample Applications.

The latest Windows base images were released on January 26, 2024, and the latest managed AppStream 2.0 image updates for the Windows platform were released on January 26, 2024. The following table lists the software components for the latest released base images and the components that are available if you update your image using managed image updates. If the version is marked “latest”, the current stable software component available from the vendor will be installed. If the version is marked “not included”, managed image updates is not managing the component and the version will not be changed when you update your image.

The following table lists the software components for the latest released Windows images.

Software component	Latest base images (January 26, 2024)	Managed AppStream 2.0 image updates (January 26, 2024)
Amazon AWS (AvsCamera) Driver	1.0.0.6	1.0.0.6
Amazon CloudWatch Agent	1.4.37887	Not included
SSM Agent	3.2.1705.0	3.2.1798.0
NICE DCV Virtual Display	NICE DCV Virtual Display 1.4.78.0	NICE DCV Virtual Display 1.4.78.0

Software component	Latest base images (January 26, 2024)	Managed AppStream 2.0 image updates (January 26, 2024)
AMD Driver for Graphics Design instances	24.20.13028.7002	24.20.13028.7002
AppStream 2.0 Agent	01-17-2024	01-17-2024
AWS Command Line Interface (AWS CLI)	1.32.29 (Windows Server 2016/2019) 2.15.15.0 (Windows Server 2022)	Not included
Firefox	122.0 (Windows Server 2016/2019)	Not included
Microsoft Message Queuing (MSMQ)	Installed with Windows Server	Installed with Windows Server
NVIDIA Graphics Driver for Graphics Pro, G4dn, and G5 instances	537.70 (Windows Server 2019/2022) 512.78 (Windows Server 2016)	537.70 (Windows Server 2019/2022) 512.78 (Windows Server 2016)
Process monitor	3.95	Latest
Quality Windows Audio/Video Experience (qWAVE)	Installed with Windows Server	Installed with Windows Server
Visual C++ redistributable packages	Microsoft Visual C++ 2013 Redistributable (x64) - 12.0.40664.0	Microsoft Visual C++ 2013 Redistributable (x64) - 12.0.40664.0
Windows Server updates	Base image updates as of January 2024	Latest
WinSCard Filter Driver	1.0.19.0	1.0.19.0

The latest Linux base images were released on November 13, 2023. The following table lists the software components for the latest released Linux base images.

Software component	Latest base images (June 11, 2023)
AWS Command Line Interface (AWS CLI)	1.18.147-1
Amazon CloudWatch Agent	1.300028.1-1
SSM Agent	3.2.1798.0-1
NICE DCV Server AppStream	2023.1.16167-17
Cloud-init	19.3-46
AL2 Kernel	4.14.327-246.539
NVIDIA Graphics Driver for Graphics Pro, G4dn, and G5 instances	535.104.05
Cuda Version	12.2

Important

The following public images are deprecated and therefore no longer available from AWS:

- Windows images released before November 2023
- Linux images released before June 2023
- Images for the Graphics Desktop instance family

If you want to use an image for a multi-session fleet, the image must meet the following conditions:

- The image must be created from a base image released on or after June 12, 2023. Or, the image must be updated by using managed AppStream 2.0 image updates released on or after September 6, 2023. For more information, see the section called "Update an Image by Using Managed AppStream 2.0 Image Updates".

- The AppStream 2.0 agent release version must be 09-06-2023 or later. For more information, see the section called "Manage AppStream 2.0 Agent Versions".
- If you have updated your image using Managed AppStream 2.0 Image updates, then the AppStream 2.0 agent release version is not applicable. Your image must be updated using a Managed Image Update released on or after September 6, 2023. For more information, see the section called "Update an Image by Using Managed AppStream 2.0 Image Updates".

The following table describes all released base images.

Release	Platform	Image	Changes
01-26-2024	Windows	• Base • Graphics Design • Graphics G4dn • Graphics Pro • Graphics G5	• Includes Microsoft security updates up to January 2024
12-11-2023	Windows	• Base • Graphics G4dn • Graphics G5	• Add support for Windows Server 2022
11-13-2023	Windows	• Base • Graphics Design • Graphics G4dn • Graphics Pro • Graphics G5	• Includes Microsoft security updates up to November 2023
11-13-2023	Amazon Linux 2	• Base • Graphics G4dn • Graphics Pro • Graphics G5	• Updated Linux to version 2.0.20231101.0. For more information, see Amazon Linux 2.0.20231101.0 release notes.

Release	Platform	Image	Changes
06-12-2023	Windows	• Base • Graphics Design • Graphics G4dn • Graphics Pro	• Includes Microsoft security updates up to June 2023
06-11-2023	Amazon Linux 2	• Base • Graphics G4dn • Graphics Pro	• Updated Linux to version 2.0.20230530.0. For more information, see Amazon Linux 2 2.0.20230530.0 release notes.
03-29-2023	Windows	• Base • Graphics Design • Graphics G4dn • Graphics Pro	• Includes Microsoft security updates up to February 2023
03-15-2023	Amazon Linux 2	• Base • Graphics G4dn • Graphics Pro	• Updated Linux to version 2.0.20220805.0. For more information, see Amazon Linux 2 2.0.20230221.0 release notes. • Improves Webcam experience • Resolves an issue that prevents AppStream 2.0 fleet instances from provisioning when the system cryptography is set to use FIPS-compliant algorithms
10-05-2022	Windows	• Base • Graphics Design • Graphics G4dn • Graphics Pro	• Includes Microsoft security updates up to September 13, 2022

Release	Platform	Image	Changes
09-21-2022	Amazon Linux 2	• Base • Graphics G4dn • Graphics Pro	• Updated Linux to version 2.0.20220805.0. For more information, see Amazon Linux 2.0.20220805.0 release notes. • Includes Image Assistant GUI • Includes webcam support
09-14-2022	Amazon Linux 2	• Graphics G4dn • Graphics Pro	• Includes NVIDIA Graphics Driver (510.85.02)
09-01-2022	Windows	• Graphics G4dn • Graphics Pro	• Includes NVIDIA Graphics Driver (473.47) for Windows Server 2012 R2 • Includes NVIDIA Graphics Driver (512.78) for Windows Server 2016 and Windows Server 2019
07-12-2022	Windows	• Base • Graphics Design • Graphics G4dn • Graphics Pro	• Includes Microsoft security updates up to June 14, 2022 • Includes latest AMD Driver (24.20.13028.7002) for Graphics Design instances for Windows Server 2016 and Windows Server 2019 • Includes latest NVIDIA Graphics Driver (472.98) for Graphics Pro and G4dn instances for Windows Server 2012R2 • Includes latest NVIDIA Graphics Driver (511.65) for Graphics Pro and G4dn instances for Windows Server 2016 and Windows Server 2019 • Includes CloudWatch Agent 1.3.50742 • Includes SSM Agent 3.1.1575.0 • Includes AWS Command Line Interface (AWS CLI) 1.23.11

Release	Platform	Image	Changes
06-20-2022	Amazon Linux 2	• Base • Graphics G4dn • Graphics Pro	• Updated Linux to version 2.0.20220426.0. For more information, see Amazon Linux 2.0.20220426.0 release notes.
03-03-2022	Windows	• Base • Graphics Design • Graphics G4dn • Graphics Pro	• Includes Microsoft security updates up to January 11, 2022
02-18-2022	Amazon Linux 2	• Base • Graphics G4dn • Graphics Pro	• Updated Linux to version 2.0.20211223.0. For more information, see Amazon Linux 2.0.20211223.0 release notes. • Latest Linux base images
11-19-2021	Amazon Linux 2	• Base • Graphics G4dn • Graphics Pro	• Latest Linux base images, including blank screen fixes on small instance types
11-15-2021	Amazon Linux 2	• Base • Graphics G4dn • Graphics Pro	• Linux base images
10-08-2021	Windows	• Base • Graphics Design • Graphics G4dn • Graphics Pro • Sample apps	• Includes Microsoft security updates up to September 15, 2021 • AWS Tools for PowerShell updated to version 3.15.1398

Release	Platform	Image	Changes
07-19-2021	Windows	• Base • Graphics Design • Graphics G4dn • Graphics Pro	• Includes Microsoft Windows updates up to July 13, 2021
06-01-2021	Windows	• Base • Graphics Design • Graphics G4dn • Graphics Pro	• Includes Microsoft Windows updates up to April 14, 2021 • Includes AMD driver version 24.20.130 28.5012 for Graphics Design instances
12-28-2020	Windows	• Base • Graphics Design • Graphics G4dn • Graphics Pro	• Includes a driver that adds support for using smart cards. Smart cards can be used for Windows sign in, Active Directory-joined streaming instances , and in-session authentication for streaming applications • Includes Microsoft Windows updates up to December 9, 2020 • Includes AWS CLI version 1.18.138 • Includes NVIDIA Graphics Driver version 451.48 for Graphics Pro and Graphics G4dn instances
07-16-2020	Windows	• Base • Graphics Design • Graphics G4dn • Graphics Pro	• Includes Microsoft Windows updates up to June 9, 2020 • Includes AWS CLI version 1.18.86 • Includes NVIDIA Graphics Driver version 441.66 for Graphics Pro instances

Release	Platform	Image	Changes
04-22-2020	Windows	• Base (Windows Server 2019) • Graphics Design (Windows Server 2019) • Graphics G4dn (Windows Server 2019) • Graphics Pro (Windows Server 2019)	• Includes Microsoft Windows updates up to March 10, 2020 • Includes AWS CLI version 1.18.21 • Includes NVIDIA Graphics Driver version 441.66 for Graphics Pro instances
03-18-2020	Windows	• Base • Graphics Design • Graphics Pro	• Includes Microsoft Windows updates up to February 11, 2020 • Includes AWS CLI version 1.17.5 • Includes NVIDIA Graphics Driver version 412.16 for Graphics Pro instances
03-16-2020	Windows	• Graphics G4dn	• Adds support for Graphics G4dn instances based on the EC2 G4dn family (Windows Server 2012 R2) • Includes Microsoft Windows updates up to February 11, 2020 • Includes AWS CLI version 1.17.5
03-05-2020	Windows	• Graphics G4dn	• Adds support for Graphics G4dn instances based on the EC2 G4dn family (Windows Server 2016 and Windows Server 2019) • Includes Microsoft Windows updates up to February 11, 2020 • Includes AWS CLI version 1.17.5

Release	Platform	Image	Changes
01-13-2020	Windows	• Graphics Design	• Adds support for Windows Server 2019, with Microsoft Windows updates up to November 12, 2019
12-12-2019	Windows	• Base • Graphics Design • Graphics Pro	• Includes Microsoft Windows updates up to November 12, 2019 • Includes AWS CLI version 1.16.284 • Includes a new version of the SSM Agent (v2.3.760.0), which resolves an issue that prevented streaming instances from being provisioned
09-18-2019	Windows	• Base • Graphics Design • Graphics Pro	• Includes Microsoft Windows updates up to August 13, 2019 for all Base and Graphics Pro instances and for Graphics Design Windows Server 2012 R2. Graphics Design Windows Server 2016 instances already include this version. • Includes AWS CLI version 1.16.222 for all Base and Graphics Pro instances and Graphics Design Windows Server 2012 R2. Graphics Design Windows Server 2016 instances already include this version. • Includes a fix to prevent Windows Defender from being enabled by default on Windows Server 2016 and Windows Server 2019 image builder instances. For more information, see Windows Update and Antivirus Software on AppStream 2.0.

Release	Platform	Image	Changes
09-05-2019	Windows	• Graphics Design	• Adds support for Windows Server 2016 • Includes Microsoft Windows updates up to August 13, 2019 • Includes AWS CLI version 1.16.222 • Includes AMD Driver version 24.20.130 28.3002 for Graphics Design instances (compatible with Windows Server 2016)
06-24-2019	Windows	• Base • Graphics Pro	• Adds support for Windows Server 2016 and Windows Server 2019
05-28-2019	Windows	• Base • Graphics Design • Graphics Pro	• Includes Microsoft Windows updates up to May 14, 2019
04-29-2019	Windows	• Base • Graphics Design • Graphics Pro	• Includes Microsoft Windows updates up to April 20, 2019 • Includes AWS CLI version 1.16.126 • Includes NVIDIA Graphics Driver 412.16 for Graphics Pro instances
01-22-2019	Windows	• Base • Graphics Design • Graphics Pro	• Includes Microsoft Windows updates up to December 10, 2018 • Includes AWS CLI version 1.16.84 • Includes NVIDIA Graphics Driver version 391.58 for Graphics Pro instances
06-12-2018	Windows	• Base • Graphics Design • Graphics Desktop • Graphics Pro	• Includes Microsoft Windows updates up to May 9, 2018 • Includes Windows PowerShell 5.1

Release	Platform	Image	Changes
05-02-2018	Windows	• Base • Graphics Design • Graphics Desktop • Graphics Pro	• Includes Microsoft Windows updates up to April 10, 2018 • Adds the following language packs: Japanese, Korean, Portuguese (Brazil), Thai, Chinese (Simplified), Chinese (Traditional)
03-19-2018	Windows	• Base • Graphics Design • Graphics Desktop • Graphics Pro	• Includes Microsoft Windows updates up to February 23, 2018 • Includes the following language packs: German, French, Italian, Spanish, Dutch • Resolves intermittent issues with using Microsoft Visio and Microsoft Project applications during streaming sessions
01-24-2018	Windows	• Base • Graphics Design • Graphics Desktop • Graphics Pro	• Includes Microsoft Windows updates up to January 5, 2018 • Includes Microsoft Windows updates for the Spectre and Meltdown vulnerabi lities • Enables a default profile to be created on image builders and used for the AWS Command Line Interface (CLI) during streaming sessions
01-01-2018	Windows	• Base • Graphics Design • Graphics Desktop • Graphics Pro	• Resolves an issue with connectivity to AppStream 2.0 instances

Release	Platform	Image	Changes
12-07-2017	Windows	• Base • Graphics Design • Graphics Desktop • Graphics Pro	• Includes Microsoft Windows updates up to November 19, 2017 • Adds support for managed AppStream 2.0 agent updates
11-13-2017	Windows	• Base	• Resolves an issue with Microsoft Office 365 applications not working during streaming sessions • Includes Microsoft Windows updates up to October 11, 2017
09-05-2017	Windows	• Base • Graphics Design • Graphics Desktop • Graphics Pro	• New Graphics Design instance family • Support for On-Demand fleets • Updated approach for session context • Includes Microsoft Windows updates up to August 9, 2017 • Resolves an intermittent issue with applications not coming to the foreground • Resolves an intermittent issue with applications not appearing in tile view
07-25-2017	Windows	• Graphics Desktop • Graphics Pro	• New Graphics Desktop and Graphics Pro instance families • Adds support for 2 K resolution
07-24-2017	Windows	• Base	• Includes Microsoft Windows updates up to July 13, 2017 • Adds support for Microsoft Active Directory domains

Release	Platform	Image	Changes
06-20-2017	Windows	• Base • Sample apps	• Optimizes application launch performan ce • Resolves an issue with applications not displaying in tile view • Resolves an issue with applications displaying in tile view only • Resolves an issue with applications displaying multiple times in tile view • Resolves an issue with recently launched application windows not appearing in the foreground • Resolves an issue with page margins when printing
05-18-2017	Windows	• Base • Sample apps	• Adds support for AppStream 2.0 home folders • Includes Microsoft Windows updates up to May 16, 2017 • Resolves an intermittent network issue that affects internet connections from streaming instances • Resolves an issue with application tiles not functioning correctly

Images

You can create Amazon AppStream 2.0 images that contain applications you can stream to your users and default system and application settings to enable your users to get started with those applications quickly. However, after you create an image, you can't change it. To add other applications, update existing applications, or change image settings, you must start and reconnect to the image builder that you used to create the image. If you deleted that image builder, launch a new image builder that is based on your image. Then make your changes and create a new image. For more information, see Launch an Image Builder to Install and Configure Streaming Applications and Tutorial: Create a Custom AppStream 2.0 Image by Using the AppStream 2.0 Console.

Images that are available to you are listed in the **Image Registry** in the AppStream 2.0 console. They are categorized as public, private, or shared. You can use any of these image types to launch an image builder and set up an AppStream 2.0 fleet. Shared images are owned by other Amazon Web Services accounts and shared with you. Permissions set on images that are shared with you may limit what you can do with those images. For more information, see Administer Your Amazon AppStream 2.0 Images.

Contents

- Default Application and Windows Settings and Application Launch Performance
- Manage AppStream 2.0 Agent Versions
- AppStream 2.0 Agent Release Notes
- Tutorial: Create a Custom AppStream 2.0 Image by Using the AppStream 2.0 Console
- Administer Your Amazon AppStream 2.0 Images
- Create Your AppStream 2.0 Image Programmatically by Using the Image Assistant CLI Operations
- Create Your Linux-Based Images
- Use Session Scripts to Manage Your AppStream 2.0 Users' Streaming Experience

Default Application and Windows Settings and Application Launch Performance

You can create default application and Windows settings to enable your users to get started with their applications quickly, so that they won't need to create or configure the settings themselves.

AppStream 2.0 optimizes the launch performance of your applications for your users' streaming sessions. To ensure that all of the required files are included in this process, you may need to manually add certain files and folders to the optimization manifest.

Contents

- Creating Default Application and Windows Settings for Your AppStream 2.0 Users
- Optimizing the Launch Performance of Your Applications

Creating Default Application and Windows Settings for Your AppStream 2.0 Users

Application customizations and Windows settings that are saved to the Windows user profile folder or the user registry hive can be set as defaults. When you save the default settings by using the **Template User** in Image Assistant, AppStream 2.0 replaces the Windows default user profile with the profile that you configure. The Windows default user profile is then used to create the initial settings for users in the fleet instance. If the application or Windows settings that you configure don't work in the fleet, confirm that they are saved in the Windows user profile. For more information, see Step 3: Create Default Application and Windows Settings in Tutorial: Create a Custom AppStream 2.0 Image by Using the AppStream 2.0 Console.

Default settings that you can create and configure include:

- Application preferences, including a browser home page, toolbar customizations, and security settings.
- Application data settings, including browser bookmarks and connection profiles.
- Windows experience settings, including displaying file name extensions and hidden folders.

Additionally, you can modify or disable Internet Explorer security settings, such as Enhanced Security Configuration (ESC). For more information, see Disable Internet Explorer Enhanced Security Configuration.

Optimizing the Launch Performance of Your Applications

When you create an image, AppStream 2.0 requires that you optimize the launch performance of your applications for your users' streaming sessions. When your applications are opened during this process, make sure that they use the initial components required by your users. Doing so

ensures that these components are captured by the optimization process. In some cases, not all of the files required for the optimizations are detected. Examples of such files would be plug-ins or components that aren't opened in the image builder. To ensure that all of the files needed for your application are captured, you can include them in the optimization manifest. Adding files to the optimization manifest may increase the time it takes for fleet instances to be created and made available for users. Doing so, however, reduces the time it takes for the application to be launched the first time on the fleet instance.

To optimize all the files in a folder, open PowerShell and use the following PowerShell command:

```
dir -path "C:\Path\To\Folder\To\Optimize" -Recurse -ErrorAction SilentlyContinue |
 %{$_.FullName} | Out-File "C:\ProgramData\Amazon\Photon\Prewarm\PrewarmManifest.txt" -
encoding UTF8 -append
```

By default, Image Assistant replaces the application optimization manifest each time the Image Assistant **Optimize** step runs. You must run the PowerShell command to optimize all files in a folder:

- Each time after the **Optimize** step runs.
- Before you choose **Disconnect and create image** on the Image Assistant **Review** page.

Alternatively, you can specify the optimization manifest on a per-application basis by using the Image Assistant command line interface (CLI) operations. When you specify the optimization manifest by using the Image Assistant CLI operations, AppStream 2.0 merges the specified application optimization manifest with the files identified by the Image Assistant **Optimize** step. For more information, see Create Your AppStream 2.0 Image Programmatically by Using the Image Assistant CLI Operations.

Manage AppStream 2.0 Agent Versions

The AppStream 2.0 agent is software that runs on your streaming instances and enables users to stream applications. When you create a new image, the **Always use latest agent version** option is selected by default. When this option is selected, new image builders or fleet instances that are launched from your image always use the latest AppStream 2.0 agent version. You might want to control agent updates to ensure compatibility with your software or to qualify the updated environment before you deploy it for your end users.

The following procedures describe how to manage AppStream 2.0 agent versions.

Contents

- Create an Image That Always Uses the Latest Version of the AppStream 2.0 Agent
- Create an Image That Uses a Specific Version of the AppStream 2.0 Agent
- Create an Image That Uses a Newer Version of the AppStream 2.0 Agent

Create an Image That Always Uses the Latest Version of the AppStream 2.0 Agent

When your images are configured to always use the latest AppStream 2.0 agent version, your streaming instances are automatically updated with the latest features, performance improvements, and security updates that are available from AWS when a new agent version is released.

> **Note**
>
> In some cases, a new AppStream 2.0 agent version might conflict with your software. We recommend that you qualify the new AppStream 2.0 agent version before deploying it to your production fleets.

To create an image that always uses the latest version of the AppStream 2.0 agent

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. Do either of the following:
   - If you have an image builder that you want to use to create the image, start the image builder and then connect to it. If the image builder is not running the latest version of the AppStream 2.0 agent, you are prompted to choose whether to start the image builder with the latest agent. Make sure that this option is selected, choose **Start**, and then connect to the image builder.
   - If you do not have an image builder that you want to use to create the image, launch a new image builder. In **Step 1: Choose Image**, choose an AWS base image or a custom image. In **Step 2: Configure Image Builder**, if the image that you choose is not running the latest version of the AppStream 2.0 agent, the **AppStream 2.0** section displays. In the **Agent version** list, select the latest agent version. Complete the remaining steps to create the

image builder, and then connect to it. For more information, see Launch an Image Builder to Install and Configure Streaming Applications.

3. On the image builder desktop, open Image Assistant and follow the steps to create your new image. For the **Configure Image** step, make sure that **Always use the latest agent version** is selected. For more information, see Tutorial: Create a Custom AppStream 2.0 Image by Using the AppStream 2.0 Console.

   If you decide later to not always use the latest version of the AppStream 2.0 agent, you must create a new image and clear that option.

4. Create a new fleet or modify an existing one. When you configure the fleet, select the new image that you created. For more information, see Create an AppStream 2.0 Fleet and Stack.
5. Create a new stack or modify an existing one and associate it with your fleet.

Create an Image That Uses a Specific Version of the AppStream 2.0 Agent

You may want to control AppStream 2.0 agent updates rather than always using the latest version so that you can test for compatibility first. To ensure that the version of the AppStream 2.0 agent you use is compatible with your streaming applications, you can create an image that uses a specific version of the agent software. Then perform your qualification tests in a separate fleet before deploying to your production fleet.

When you create the image, make sure that the **Always use latest agent version** option is not selected. Doing so pins your image to the version of the AppStream 2.0 agent that you selected when you launched the image builder, rather than always using the latest version. After you finish your qualification tests, you can update your production fleet with the image.

To create an image that uses a specific version of the AppStream 2.0 agent

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. Do either of the following:

   - If you have an image builder that you want to use to create the image, start the image builder and then connect to it.
   - If you do not have an image builder that you want to use to create the image, launch a new image builder. In **Step 1: Choose Image**, choose an AWS base image or a custom image. In **Step 2: Configure Image Builder**, if the image that you choose is not running the latest

version of the AppStream 2.0 agent, the **AppStream 2.0** section displays. In the **Agent version** list, do not select the latest agent version. Complete the remaining steps to create the image builder, and then connect to it. For more information, see Launch an Image Builder to Install and Configure Streaming Applications.

3. On the image builder desktop, open Image Assistant and follow the steps to create your new image. For the **Configure Image** step in Image Assistant, make sure that **Always use the latest agent version** is not selected. For more information, see Tutorial: Create a Custom AppStream 2.0 Image by Using the AppStream 2.0 Console.

   If you decide later to always use the latest version of the AppStream 2.0 agent, you must create a new image and select that option.
4. Create a new fleet or modify an existing one. When you configure the fleet, select the new image that you created. For more information, see Create an AppStream 2.0 Fleet and Stack.
5. Create a new stack or modify an existing one and associate it with your fleet.
6. Connect to your fleet and test your applications for compatibility.

Create an Image That Uses a Newer Version of the AppStream 2.0 Agent

If you pin your image to a specific AppStream 2.0 agent version, you must update to a newer version by creating a new image. This approach lets you test each agent update for compatibility first, and then update your fleet incrementally.

When you create the image, make sure that the **Always use latest agent version** option is not selected. After you create your image, perform your qualification tests in a separate fleet before deploying to your production fleet. After you finish your qualification tests, you can update your production fleet with the image.

To create an image that uses a newer version of the AppStream 2.0 agent

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. Do either of the following:

   - If you have an image builder that you want to use to create the image, start the image builder and then connect to it. If the image builder is not running the latest version of the AppStream 2.0 agent, you are prompted to choose whether to start the image builder with the latest agent. Make sure that this option is selected, choose **Start**, and then connect to the image builder.

- If you do not have an image builder that you want to use to create the image, launch a new image builder. In **Step 1: Choose Image**, choose an AWS base image or a custom image. In **Step 2: Configure Image Builder**, if the image that you choose is not running the latest version of the AppStream 2.0 agent, the **AppStream 2.0** section displays. In the **Agent version** list, select the latest agent version. Complete the remaining steps to create the image builder, and then connect to it. For more information, see Launch an Image Builder to Install and Configure Streaming Applications.

3. On the image builder desktop, open Image Assistant and follow the steps to create your new image. For the **Configure Image** step in Image Assistant, make sure that **Always use the latest agent version** is not selected. For more information, see Tutorial: Create a Custom AppStream 2.0 Image by Using the AppStream 2.0 Console.

   If you decide later to always use the latest version of the AppStream 2.0 agent, you must create a new image and select that option.

4. Create a new fleet or modify an existing one. When you configure the fleet, select the new image that you created. For more information, see Create an AppStream 2.0 Fleet and Stack.
5. Create a new stack or modify an existing one and associate it with your fleet.
6. Connect to your fleet and test your applications for compatibility.

AppStream 2.0 Agent Release Notes

The Amazon AppStream 2.0 agent software runs on your streaming instances, enabling end users to connect to and start their streaming applications. Starting December 7, 2017, your streaming instances can be automatically updated with the latest features, performance improvements, and security updates that are available from AWS. Before December 7, 2017, agent updates were included with new base image releases.

To use the latest AppStream 2.0 agent software, you need to rebuild your images by using new base images published by AWS on or after December 7, 2017. When you do this, the option to enable automatic updates of the agent is selected by default in the Image Assistant. We recommend that you leave this option selected so that any new image builder or fleet instance that is launched from your image always uses the latest version of the agent. For more information, see Tutorial: Create a Custom AppStream 2.0 Image by Using the AppStream 2.0 Console.

The following table describes the latest updates that are available in released versions of the AppStream 2.0 agent for Windows instances.

Amazon AppStream 2.0 agent version	Changes
01-17-2024	• Adds support for audio out on multi-session fleets • Adds support for session scripts on multi-session fleets • Improves provisioning resilience on multi-session fleets
12-07-2023	• Adds support for Windows Server 2022 • Improves streaming performance on Windows Server 2019 • Adds AWS CLI v2 support • Adds keyboard support to switch between applicati ons • Solves an issue with certificate-based authentication when Windows session is locked • Note: Windows Server 2012 R2 reached end of support on October 10th, 2023. For better streaming experience support, upgrade to Windows Server 2016, Windows Server 2019, or Windows Server 2022.
09-06-2023	• Adds support for multi-session fleets • Improves instance and session provisioning • Resolves an issue with copy/paste functionality • Requires the following software components: • Microsoft .NET Framework Runtime — 4.7.2
05-30-2023	• Improves instance provisioning resilience
05-08-2023	• Resolves an issue with a shutdown warning in fleet instances for Windows 2016 and Windows 2012 R2
04-13-2023	• Resolves an issue with the streaming session being stuck in a connecting state

Amazon AppStream 2.0 agent version	Changes
03-21-2023	• Resolves an issue with application freezing • Resolves an issue with physical smartcard authentic ation failure • Resolves an issue with home folders not working with FIPS enabled on Windows • Improves instance provisioning resilience • Improves performance with physical smartcard logon time for Windows Server 2019
10-13-2022	• Improves performance with agents • Resolves issue with DCV physical smartcard
06-20-2022	• Adds backwards compatibility for the USB string filter file location on old images • Improves instance provisioning resilience • Improves session connection reliability
03-14-2022	• Resolves an issue with regional settings not updating

02-21-2022	• Resolves issue with Microsoft OneDrive copying larger files
	• Improves agent robustness on small instance types
	• Works with the following software components. For more information, see the section called "Base Image and Managed Image Update Release Notes".
	  • Amazon SSM Agent — 3.0.1295.0
	  • Amazon WDDM Hook Driver — 1.0.0.56 (Windows Server 2012 R2)
	  • NICE DCV Virtual Display — 1.0.34.0 (Windows Server 2016/2019)
	  • EC2Config service (Windows Server 2012 R2 only) — 4.9.4500

Amazon AppStream 2.0 agent version	Changes

12-20-2021	• Resolves an issue with the mouse disappearing when using the native client
	• Resolves an issue of storage unmount time for session termination
	• Resolves issues with system crashes on Graphics instances running Windows Server 2016
	• Added support for Windows Server instances when system cryptography group policy is enabled. For more information, see System cryptography.
	• Added the ability to toggle file system caching
	• Works with the following software components. For more information, see the section called "Base Image and Managed Image Update Release Notes".
	  • Amazon SSM Agent — 3.0.1295.0
	  • Amazon WDDM Hook Driver — 1.0.0.56 (Windows Server 2012 R2)
	  • NICE DCV Virtual Display — 1.0.34.0 (Windows Server 2016/2019)
	  • EC2Config service (Windows Server 2012 R2 only) — 4.9.4500

Amazon AppStream 2.0 agent version	Changes

10-19-2021	• Resolves an issue preventing users from streaming when the Microsoft Windows printer service is disabled
	• Resolves an issue where language pack installation doesn't complete successfully
	• Resolves an issue with the S3 Home Folder where folders and files are being changed to all uppercase
	• Works with the following software components. For more information, see the section called "Base Image and Managed Image Update Release Notes".
	  • Amazon SSM Agent — 3.0.1295.0
	  • Amazon WDDM Hook Driver — 1.0.0.56 (Windows Server 2012 R2)
	  • NICE DCV Virtual Display — 1.0.34.0 (Windows Server 2016/2019)
	  • EC2Config service (Windows Server 2012 R2 only) — 4.9.4500

Amazon AppStream 2.0 agent version	Changes

08-02-2021	• Updated USB driver to include important fixes
	• Resolves an issue where the customer's local machine's caps lock state becomes out of sync with the caps lock state of the remote machine
	• Works with the following software components. For more information, see the section called "Base Image and Managed Image Update Release Notes".
	  • Amazon SSM Agent — 3.0.1295.0
	  • Amazon WDDM Hook Driver — 1.0.0.56 (Windows Server 2012 R2)
	  • NICE DCV Virtual Display — 1.0.34.0 (Windows Server 2016/2019)
	  • EC2Config service (Windows Server 2012 R2 only) — 4.9.4419.0

07-01-2021	• Incremental agent release for Managed Image Updates. For more information, see Update an Image by Using Managed AppStream 2.0 Image Updates. • Includes changes from the 06-25-2021 agent.
06-25-2021	• Resolved various networking issues • Resolved an issue where local group policies were overridden • Resolved an issue where files were failing to be created if they were contained in parent directori es that did not exist after attempting to fetch from OneDrive and Google cloud storage • Resolved an issue where session scripts failed to run at the end of a session • Added support for webcam redirection in the web client

Amazon AppStream 2.0 agent version	Changes
05-17-2021	• Enables real-time audio-video (AV) feature by default • Fixes an output of the Image Assistant CLI command to be valid JSON • Fixes an issue causing instance provisioning failures due to internal timeouts • Amazon SSM Agent, Amazon WDDM Hook Driver, and EC2Config service versions remain the same as the previous agent version release

03-04-2021	• Resolves issues with smart card authentication that cause connection failures. The connection failures occur when users close and reopen a streaming session multiple times
	• Resolves an issue that causes right-click menu items in Microsoft Office applications to be unavailable
	• Resolves an issue that causes multiple storage connector processes for OneDrive and Google Drive to appear in Task Manager
	• Resolves an issue that prevents files larger than 2 GB from being downloaded from Google Drive
	• Resolves an intermittent issue that causes provision ing delays for AppStream 2.0 fleet instances that are joined to a Microsoft Active Directory domain
	• Works with these software components:
	  • Amazon SSM Agent — 3.0.431.0
	  • Amazon WDDM Hook Driver — 1.0.0.56
	  • EC2Config service (Windows Server 2012 R2 only) — 4.9.4279.0

Amazon AppStream 2.0 agent version	Changes

12-17-2020	• Resolves an issue that causes the application settings persistence VHD file to not download to the AppStream 2.0 fleet streaming instance
	• Resolves an issue that causes local printer redirecti on to stop working during AppStream 2.0 streaming sessions. This issue might occur when Microsoft KB4571694 is installed on the AppStream 2.0 image builder or fleet streaming instance
	• Resolves an issue that causes the Image Assistant `update-default-profile` command line interface (CLI) operation to return an error when attempting to reference a local Microsoft Windows user as the source for the default user profile
	• Resolves an issue that prevents AppStream 2.0 fleet instances from provisioning when the system cryptography is set to use FIPS-compliant algorithms
	• Resolves an issue that prevents icons from appearing on users' local computer taskbar during streaming sessions in native application mode
	• Adds support for files shared by Microsoft SharePoin t to the OneDrive for Business persistent storage connector
	• Works with these software components:
	  • Amazon SSM Agent — 2.3.1319.0
	  • Amazon WDDM Hook Driver — 1.0.0.56
	  • EC2Config service (Windows Server 2012 R2 only) — 4.9.4222.0

Amazon AppStream 2.0 agent version	Changes

01-04-2021	• Adds support for using a smart card for Windows sign in to Active Directory-joined streaming instances and in-session authentication for streaming applications
	• Works with these software components:
	  • Amazon SSM Agent — 2.3.1319.0
	  • Amazon WDDM Hook Driver — 1.0.0.56
	  • EC2Config service (Windows Server 2012 R2 only) — 4.9.4222.0

Amazon AppStream 2.0 agent version	Changes

12-17-2020	• Resolves an issue that causes the application settings persistence VHD file to not download to the AppStream 2.0 fleet streaming instance
	• Resolves an issue that causes local printer redirecti on to stop working during AppStream 2.0 streaming sessions. This issue might occur when Microsoft KB4571694 is installed on the AppStream 2.0 image builder or fleet streaming instance
	• Resolves an issue that causes the Image Assistant `update-default-profile` command line interface (CLI) operation to return an error when attempting to reference a local Microsoft Windows user as the source for the default user profile
	• Resolves an issue that prevents AppStream 2.0 fleet instances from provisioning when the system cryptography is set to use FIPS-compliant algorithms
	• Resolves an issue that prevents icons from appearing on users' local computer taskbar during streaming sessions in native application mode
	• Adds support for files shared by Microsoft SharePoin t to the OneDrive for Business persistent storage connector
	• Works with these software components:
	  • Amazon SSM Agent — 2.3.1319.0
	  • Amazon WDDM Hook Driver — 1.0.0.56
	  • EC2Config service (Windows Server 2012 R2 only) — 4.9.4222.0

Amazon AppStream 2.0 agent version	Changes

10-08-2020	• Resolves an issue that causes users to receive an internal error notification when they connect to AppStream 2.0 streaming sessions
	• Resolves an issue that causes intermittent copy and paste failures during AppStream 2.0 streaming sessions
	• Resolves an issue that causes application icons to not appear on the taskbar during AppStream 2.0 streaming sessions in native application mode
	• Resolves an issue that causes the application catalog to appear empty when users reconnect to AppStream 2.0 after an idle disconnect
	• Improves the download speed between AppStream 2.0 home folders and AppStream 2.0 fleet instances
	• Works with these software components:
	  • Amazon SSM Agent — 3.0.161.0
	  • Amazon WDDM Hook Driver — 1.0.0.56
	  • EC2Config service (Windows Server 2012R2 only) — 4.9.4222.0

Amazon AppStream 2.0 agent version	Changes

09-01-2020	• Resolves an issue that causes Graphics Design instances to not display the correct resolution
	• Resolves an issue that causes a white screen when using the AppStream 2.0 client in native application mode to stream Microsoft Remote Desktop
	• Resolves an issue that causes a streaming applicati on to freeze when minimized. This issue occurs when using the AppStream 2.0 client in native application mode
	• Works with these software components:
	  • Amazon SSM Agent — 2.3.1319.0
	  • Amazon WDDM Hook Driver — 1.0.0.56
	  • EC2Config service (Windows Server 2012R2 only) — 4.9.4222.0

07-30-2020	• Adds support for printer redirection to the AppStream 2.0 client for Windows
	• Resolves an issue that causes file downloads for files greater than 5 GB to stop, and then fail
	• Improves clipboard performance when using Microsoft Office 2016 plug-ins
	• Works with these software components:
	  • Amazon SSM Agent — 2.3.1319.0
	  • Amazon WDDM Hook Driver — 1.0.0.56
	  • EC2Config service (Windows Server 2012R2 only) — 4.9.4222.0

Amazon AppStream 2.0 agent version	Changes

05-27-2020	• Resolves an issue that prevents some applications from being resized, moved, or maximized when users stream in native application mode using the AppStream 2.0 client for Windows
	• Resolves an intermittent issue with downloading utility software. The issue may prevent image builders and fleet instances from being provisioned
	• Resolves an intermittent issue with certain language settings that may prevent image builders and fleet instances from being provisioned
	• Works with these software components:
	  • Amazon SSM Agent — 2.3.701.0
	  • Amazon WDDM Hook Driver — 1.0.0.56
	  • EC2Config service (Windows Server 2012R2 only) — 4.9.3519.0

04-20-2020	• Resolves an issue that causes streaming sessions to fail when session scripts are run
	• Improves performance when IAM roles are used
	• Works with these software components:
	  • Amazon SSM Agent — 2.3.701.0
	  • Amazon WDDM Hook Driver — 1.0.0.56
	  • EC2Config service (Windows Server 2012R2 only) — 4.9.3519.0

Amazon AppStream 2.0 agent version	Changes

02-19-2020	• Adds support for native application mode. For more information, see Native Application Mode
	• Adds support for the **Desktop** stream view
	• Improves interprocess communication between AppStream 2.0 components
	• Resolves an issue that caused streaming instances to fail to be provisioned
	• Works with these software components:
	  • Amazon SSM Agent — 2.3.701.0
	  • Amazon WDDM Hook Driver — 1.0.0.56
	  • EC2Config service (Windows Server 2012R2 only) — 4.9.3519.0

01-13-2020	• For persistent storage with Google Drive for G Suite, *Team Drives* have been renamed to *Shared Drives*
	• Resolves an issue that causes slow provisioning for streaming instances in Active Directory environments that have many users
	• Resolves an issue with accessing applications from the application switcher when the fleet user is an administrator
	• Works with these software components:
	  • Amazon SSM Agent — 2.3.701.0
	  • Amazon WDDM Hook Driver — 1.0.0.56
	  • EC2Config service (Windows Server 2012R2 only) — 4.9.3519.0

Amazon AppStream 2.0 agent version	Changes

11-13-2019	• AppStream 2.0 assemblies are now signed, including executables and installer packages
	• Works with these software components:
	  • Amazon SSM Agent — 2.3.701.0
	  • Amazon WDDM Hook Driver — 1.0.0.56
	  • EC2Config service — 4.9.3519.0

10-08-2019	• Modifies the AppStream 2.0 storage connector to no longer bypass the system proxy server
	• Works with these software components:
	  • Amazon SSM Agent — 2.3.701.0
	  • Amazon WDDM Hook Driver — 1.0.0.56
	  • EC2Config service — 4.9.3519.0

09-23-2019	• Resolves an issue that occurs when launching applicati ons that start child processes
	• Resolves an issue with directory traversal
	• Resolves an issue that causes the AppStream 2.0 agent to stop functioning, which prevents interaction with applications
	• Works with these software components:
	  • Amazon SSM Agent — 2.3.701.0
	  • Amazon WDDM Hook Driver — 1.0.0.56
	  • EC2Config service — 4.9.3519.0

Amazon AppStream 2.0 agent version	Changes

09-03-2019	• Adds support for applying IAM roles to AppStream 2.0 streaming instances. For more information, see Using an IAM Role to Grant Permissions to Applications and Scripts Running on AppStream 2.0 Streaming Instances
	• Adds support for specifying tags when creating AppStream 2.0 images programmatically with a command line interface
	• Modifies the AppStream 2.0 storage connector to bypass the system proxy server when mounting storage
	• Resolves an issue that prevented .lnk files from being specified in Image Assistant
	• Works with these software components:
	  • Amazon SSM Agent — 2.3.612.0
	  • Amazon WDDM Hook Driver — 1.0.0.56
	  • EC2Config service — 4.9.3429

08-08-2019	• Adds support for AppStream 2.0 file system redirection. For more information, see Enable File System Redirection for Your AppStream 2.0 Users
	• Adds support for three new locales: English-United Kingdom (en-GB), English-Canada (en-CA), and English-Australia (en-AU)
	• Works with these software components:
	  • Amazon SSM Agent — 2.3.612.0
	  • Amazon WDDM Hook Driver — 1.0.0.56
	  • EC2Config service — 4.9.3429

Amazon AppStream 2.0 agent version	Changes

07-26-2019	• Adds support for creating and managing AppStream 2.0 images programmatically with a command line interface. For more information, see Create Your AppStream 2.0 Image Programmatically by Using the Image Assistant CLI Operations.
	• Image creation is no longer blocked when automatic Windows updates are enabled on an image builder. However, a message notifies administrators that automatic Windows updates will be disabled on the fleet in this case (that is, automatic Windows updates won't be enabled on fleet instances).
	• Disables Windows updates when a fleet instance starts
	• Users in the Administrators group are no longer disabled when an image builder instance starts
	• Users in the Administrators group are now disabled rather than deleted when an image builder instance starts
	• Resolves an issue that prevents the streaming resolution from resizing when network connections change
	• Resolves a race condition that prevents the streaming resolution from resizing when application settings persistence is enabled
	• Works with these software components:
	  • Amazon SSM Agent — 2.3.612.0
	  • Amazon WDDM Hook Driver — 1.0.0.56
	  • EC2Config service — 4.9.3429

Amazon AppStream 2.0 agent version	Changes

06-19-2019	• Adds support for Windows Server 2016 and Windows Server 2019 base images
	• AppStream 2.0 session scripts are now terminated after the configured timeout is exceeded
	• Resolves an issue where streaming instances may not be provisioned if the locale is changed
	• Includes a change to block image creation when automatic Windows updates are enabled on an image builder
	• Resolves an issue where streaming instances may take a long time to stop if the storage connector mount fails
	• Works with these software components:
	  • Amazon SSM Agent — 2.3.612.0
	  • Amazon WDDM Hook Driver — 1.0.0.56
	  • EC2Config service — 4.9.3429

Amazon AppStream 2.0 agent version	Changes

05-07-2019	• Adds support for subscribing to AppStream 2.0 usage reports. For more information, see AppStream 2.0 Usage Reports.
	• Adds support for configuring the amount of time that users can be idle (inactive) before they are disconnec ted from their streaming session. For more informati on, see "Create a Fleet" in Create an AppStream 2.0 Fleet and Stack.
	• Resolves an issue with using Amazon S3 buckets for home folder and application settings persistence with an Amazon S3 virtual private gateway
	• Includes a change to block image creation when automatic Windows updates are enabled on an image builder
	• Resolves an issue with persistent storage drives (home folders, OneDrive, and Google Drive) intermittently disappearing from the **My Files** dialog box
	• Works with these software components:
	  • Amazon SSM Agent — 2.3.542.0
	  • Amazon WDDM Hook Driver — 1.0.0.56
	  • EC2Config service — 4.9.3289

04-02-2019	• Resolves an issue with session scripts and storage connector mounting
	• Resolves a minor issue with instance provisioning
	• Works with these software components:
	  • Amazon SSM Agent — 2.3.344.0
	  • Amazon WDDM Hook Driver — 1.0.0.56
	  • EC2Config service — 4.9.3067

Amazon AppStream 2.0 agent version	Changes

03-07-2019	• Adds support for gestures on touch-enabled iPads, Android tablets, and Windows devices
	• Resolves an issue with switching users in an image builder instance
	• Resolves an intermittent issue with instance reservati ons
	• Works with these software components:
	  • Amazon SSM Agent — 2.3.344.0
	  • Amazon WDDM Hook Driver — 1.0.0.56
	  • EC2Config service — 4.9.3067

01-22-2019	• Adds support for using on-instance session scripts to run your own custom scripts when specific events occur in users' streaming sessions
	• Adds support for adding tags to the following AppStream 2.0 resource types during resource creation: image builders, images, fleets, and stacks
	• Includes a fix removing storage connector log files from the application settings persistence Virtual Hard Disk (VHD) file
	• Prevents image creation when the display language is changed from English and the AWS Command Line Interface (AWS CLI) version is earlier than 1.16.36. For more information, see "Special Considerations for Japanese Language Settings" in Configure Default Regional Settings for Your AppStream 2.0 Users.
	• Works with these software components:
	  • Amazon SSM Agent — 2.3.344.0
	  • Amazon WDDM Hook Driver — 1.0.0.56
	  • EC2Config service — 4.9.3067

Amazon AppStream 2.0 agent version	Changes

01-08-2019	• Improves the instance provisioning time for base images dated 01-08-2019
	• Works with these software components:
	  • Amazon SSM Agent — 2.3.344.0
	  • Amazon WDDM Hook Driver — 1.0.0.56
	  • EC2Config service — 4.9.3067

12-19-2018	• Resolves an issue with dynamic applications not being added to the application catalog
	• Works with these software components:
	  • Amazon SSM Agent — 2.2.619.0
	  • Amazon WDDM Hook Driver — 1.0.0.56
	  • EC2Config service — 4.9.2644

12-17-2018	• The AppStream 2.0 client now supports a multiple-monitor experience for streaming instances that use a Graphics Design instance type
	• Resolves an issue with the temporary drive being visible on fleet instances that use a Graphics Desktop or Memory Optimized instance type
	• Works with these software components:
	  • Amazon SSM Agent — 2.2.619.0
	  • Amazon WDDM Hook Driver — 1.0.0.56
	  • EC2Config service — 4.9.2644

Amazon AppStream 2.0 agent version	Changes

12-04-2018	• Adds support for using a Japanese keyboard with web clients that run on Windows
	• Adds support for using the AppStream 2.0 dynamic application framework APIs to build a dynamic app provider
	• Resolves an issue with streaming the same session concurrently on multiple tabs or browsers
	• Includes a fix to make home folders, Google Drive, and OneDrive read-only until mounting is completed
	• Improves the mount time for home folders that are stored on fleet instances connected to an Amazon S3 VPC endpoint
	• Works with these software components:
	  • Amazon SSM Agent — 2.2.619.0
	  • Amazon WDDM Hook Driver — 1.0.0.56
	  • EC2Config service — 4.9.2644

11-14-2018	• Adds support for launching streaming sessions using the AppStream 2.0 Windows client
	• Resolves an issue with opening applications that use environment variables for the fleet user name
	• Works with these software components:
	  • Amazon SSM Agent — 2.2.619.0
	  • Amazon WDDM Hook Driver — 1.0.0.56
	  • EC2Config service — 4.9.2644

Amazon AppStream 2.0 agent version	Changes

10-30-2018	• Resolves an issue with mounting home folders that are larger than 1 GB when application settings persistence is enabled
	• Resolves an issue with image creation when IPv6 is disabled
	• Session information is now provided as environment variables within streaming instances
	• Works with these software components:
	  • Amazon SSM Agent — 2.2.619.0
	  • Amazon WDDM Hook Driver — 1.0.0.56
	  • EC2Config service — 4.9.2644

10-24-2018	• Includes a fix to display more than 1,000 files in the Amazon S3 home folders directory
	• Works with these software components:
	  • Amazon SSM Agent — 2.2.619.0
	  • Amazon WDDM Hook Driver — 1.0.0.56
	  • EC2Config service — 4.9.2644

10-01-2018	• Improves the performance of application settings persistence
	• Includes a fix to unhide all drives on a fleet instance, except Drive C and Drive D, during user streaming sessions that are launched from the instance
	• Resolves an issue with accessing minimized applicati on subwindows from the application switcher
	• Works with these software components:
	  • Amazon SSM Agent — 2.2.619.0
	  • Amazon WDDM Hook Driver — 1.0.0.56
	  • EC2Config service — 4.9.2644

Amazon AppStream 2.0 agent version	Changes

08-29-2018	• Adds support for application settings persistence
	• Resolves an issue with copying and pasting large amounts of data between applications within an AppStream 2.0 streaming session
	• Resolves an issue with accessing unresponsive applications from the application switcher
	• Works with these software components:
	  • Amazon SSM Agent — 2.2.619.0
	  • Amazon WDDM Hook Driver — 1.0.0.56
	  • EC2Config service — 4.9.2644

07-26-2018	• Adds support for OneDrive persistent storage
	• Resolves an issue with saving Visio files to home folders and Google Drive
	• Works with these software components:
	  • Amazon SSM Agent — 2.2.619.0
	  • Amazon WDDM Hook Driver — 1.0.0.56
	  • EC2Config service — 4.9.2644

06-19-2018	• Resolves an issue with optimizing images for applicati on launch
	• Works with these software components:
	  • Amazon SSM Agent — 2.2.619.0
	  • Amazon WDDM Hook Driver — 1.0.0.56
	  • EC2Config service — 4.9.2644

Amazon AppStream 2.0 agent version	Changes

06-06-2018	• Adds support for regional settings and default application and Windows settings
	• Works with these software components:
	  • Amazon SSM Agent — 2.2.619.0
	  • Amazon WDDM Hook Driver — 1.0.0.56
	  • EC2Config service — 4.9.2644

05-31-2018	• Adds support for Google Drive persistent storage
	• Works with these software components:
	  • Amazon SSM Agent — 2.2.392.0
	  • Amazon WDDM Hook Driver — 1.0.0.56
	  • EC2Config service — 4.9.2586

05-21-2018	• Adds support for administrative controls for data transfer
	• Adds support for the Safari browser on macOS X
	• Works with these software components:
	  • Amazon SSM Agent — 2.2.392.0
	  • Amazon WDDM Hook Driver — 1.0.0.56
	  • EC2Config service — 4.9.2586

03-19-2018	• Resolves an issue with minimizing the application window in certain environments
	• Works with these software components:
	  • Amazon SSM Agent — 2.2.160.0
	  • Amazon WDDM Hook Driver — 1.0.0.56
	  • EC2Config service — 4.9.2400.0

Amazon AppStream 2.0 agent version	Changes

01-24-2018	• Resolves an issue with the Alt Graph key not working on certain keyboard layouts
	• Works with these software components:
	  • Amazon SSM Agent — 2.2.93.0
	  • Amazon WDDM Hook Driver — 1.0.0.50
	  • EC2Config service — 4.9.2262.0

12-07-2017	• Resolves issues with using ALT key combinations
	• Resolves an issue with file uploads from local computers to streaming sessions
	• Works with these software components:
	  • Amazon SSM Agent — 2.2.93.0
	  • Amazon WDDM Hook Driver — 1.0.0.21
	  • EC2Config service — 4.9.2218.0

The following table describes the latest updates that are available in released versions of the AppStream 2.0 agent for Linux instances.

Amazon AppStream 2.0 agent version	Changes
11-13-2023	• Updated Linux to version 2.0.20231020.1. For more information, see Amazon Linux 2.0.20231020.1 release notes.
06-11-2023	• No agent updates
03-15-2023	• Improves webcam support • Resolves an issue that prevents AppStream 2.0 fleet instances from provisioning when the system cryptography is set to use FIPS-compliant algorithms

Amazon AppStream 2.0 agent version	Changes
09-21-2022	• Supports webcam • Supports Image Assistant GUI
11-19-2021	• Resolves blank screen issues on small instance types
11-15-2021	• Supports Linux instances

Tutorial: Create a Custom AppStream 2.0 Image by Using the AppStream 2.0 Console

This tutorial describes how to create AppStream 2.0 images that are based on Microsoft Windows Server operating systems. If you want to create custom images that are based on Amazon Linux 2 operating system, see the section called "Tutorial: Create a Custom Linux-Based Image".

In this tutorial, you will learn how to create a custom Amazon AppStream 2.0 image that contains applications you can stream to your users, and default application and Windows settings to enable your users to get started with their applications quickly. To complete this tutorial, you must already have an image builder. If you don't have an image builder, see Launch an Image Builder to Install and Configure Streaming Applications.

> **Important**
>
> This tutorial includes information that applies to the latest base image release. For more information, see AppStream 2.0 Base Image and Managed Image Update Release Notes.

Contents

- Step 1: Install Applications on the Image Builder
- Step 2: Create an AppStream 2.0 Application Catalog
- Step 3: Create Default Application and Windows Settings
- Step 4: Test Applications
- Step 5: Optimize Applications

- Step 6: Finish Creating Your Image
- Step 7 (Optional): Tag and Copy an Image
- Step 8: Clean Up

Step 1: Install Applications on the Image Builder

In this step, you connect an image builder and install your applications on the image builder.

> **Important**
>
> To complete this step, you must log into the image builder with the local **Administrator** account or a domain account that has local administrator permissions. Do not rename or delete the local built-in **Administrator** account. If you rename or delete this account, the image builder will not start and image creation will fail.

To install applications on the image builder

1. Connect to the image builder by doing either of the following:

   - Use the AppStream 2.0 console (for web connections only)
   - Create a streaming URL (for web or AppStream 2.0 client connections)

   > **Note**
   >
   > If the image builder that you want to connect to is joined to an Active Directory domain and your organization requires smart card sign in, you must create a streaming URL and use the AppStream 2.0 client for the connection. For information about smart card sign in, see Smart Cards.

2. Install applications from an application website or other download source. Install the applications you want before proceeding to the next step.

   > **Note**
   >
   > Download and install applications only from sites that you trust.

If an application requires the Windows operating system to restart, let it do so. Before the operating system restarts, you are disconnected from your image builder. After the restart is complete, connect to the image builder again, then finish installing the application.

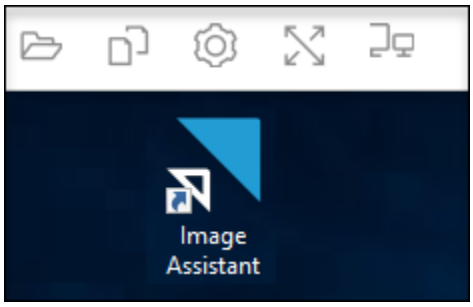
Step 2: Create an AppStream 2.0 Application Catalog

In this step, create an AppStream 2.0 application catalog by specifying applications (*.exe*), batch scripts (*.bat*), and application shortcuts (*.lnk*) for your image. For each application that you plan to stream, you can specify the name, display name, executable file to launch, and icon to display. If you choose an application shortcut, these values are prepopulated for you.

> **Important**
>
> To complete this step, you must be logged into the image builder with the local **Administrator** account or a domain account that has local administrator permissions.

To create an AppStream 2.0 application catalog

1. From the image builder desktop, open Image Assistant. Image Assistant guides you through the image creation process.



2. In **1. Add Apps**, choose **+ Add App**, and navigate to the location of the application, script, or shortcut to add. Choose **Open**.
3. In the **App Launch Settings** dialog box, keep or change the default settings for **Name**, **Display Name**, and **Icon Path**. Optionally, you can specify launch parameters (additional arguments passed to the application when it is launched) and a working directory for the application. When you're done, choose **Save**.

The **Display Name** and **Icon Path** settings determine how your application name and icon appear in the application catalog. The catalog displays to users when they sign in to an AppStream 2.0 streaming session.

4. Repeat steps 2 and 3 for each application in Image Assistant and confirm that the applications appear on the **Add Apps** tab. When you're done, choose **Next** to continue using Image Assistant to create your image.

Step 3: Create Default Application and Windows Settings

In this step, you create default application and Windows settings for your AppStream 2.0 users. Doing this enables your users to get started with applications quickly during their AppStream 2.0 streaming sessions, without the need to create or configure these settings themselves.

> **Important**
>
> To complete this step, you must be logged into the image builder with the local **Template User** account or a domain user account that does not have local administrator permissions.

To create default application and Windows settings for your users

1. In Image Assistant, in **2. Configure Apps**, choose **Switch user**. This disconnects you from the current session and displays the login menu.
2. Do either of the following:
   - If your image builder is not joined to an Active Directory domain, on the **Local User** tab, choose **Template User**. This account enables you to create your default application and Windows settings.
   - If your image builder is joined to an Active Directory domain, choose **Directory User**, and log in as a domain user that does not have local administrator permissions.
3. From the image builder desktop, open Image Assistant, which displays the applications that you added when you created the application catalog.
4. Choose the application for which you want to create default application settings.
5. After the application opens, create these settings as needed.
6. When you're done, close the application, and return to Image Assistant.

7. If you specified more than one application in Image Assistant, repeat steps 4 through 6 for each application as needed.
8. If you want default Windows settings, create them now. When you're done, return to Image Assistant.
9. Choose **Switch user** and log in with the same account that you used to create the application catalog (an account that has local administrator permissions).
10. In Image Assistant, in **2. Configure Apps**, do either of the following:

    - If your image builder is not joined to an Active Directory domain, choose **Save settings**.
    - If your image builder is joined to an Active Directory domain, in the **Choose which user settings to copy** list, choose the same account that you used to log into the image builder when you created the default application and Windows settings, then choose **Save settings**.

      The **Choose which settings to copy** list displays any account that currently has settings saved on the image builder.
11. When you're done, choose **Next** to continue creating your image.

Step 4: Test Applications

In this step, verify that the applications you've added open correctly and perform as expected. To do so, start a new Windows session as a user who has the same permissions as your users.

> **Important**
>
> To complete this step, you must log in to the image builder with the **Test User** account or a domain account that does not have local administrator permissions.

To test your applications

1. In Image Assistant, in **3. Test**, do either of the following:

    - If your image builder is not joined to an Active Directory domain, choose **Switch user**.
    - If your image builder is joined to an Active Directory domain, you require a domain account to test your applications, and the user already has settings on the image builder, you must reset the application settings for that user. To do so, select the user from the **User to reset** list, and choose **Reset**. When you're done, choose **Switch user**.

> **Note**
>
> If your image builder is new and no users have settings on the image builder, the list does not display any users.

2. Choose the user to use for testing by doing either of the following:
   - If your image builder is not joined to an Active Directory domain, choose **Test User**. This account enables you to test your applications by using the same policies and permissions as your users.
   - If your image builder is joined to an Active Directory domain, choose **Directory User**, specify the credentials for a domain account that does not have local administrator permissions, then choose **Log in**.
3. From the image builder desktop, open Image Assistant, which displays the applications that you specified when you created the application catalog.
4. Choose the application that you want to test, to confirm that it opens correctly and that any default application settings you created are applied.
5. After the application opens, test it as needed. When you're done, close the application and return to Image Assistant.
6. If you specified more than one application in Image Assistant, repeat steps 4 and 5 to test each application as needed.
7. When you're done, choose **Switch user**, then do either of the following:
   - If your image builder is not joined to an Active Directory domain, on the **Local User** tab, choose **Administrator**.
   - If your image builder is joined to an Active Directory domain and you logged in as a domain user with local administrator permissions to specify applications in Image Assistant, log in as that user.
8. Choose **Next** to continue creating your image.

Step 5: Optimize Applications

In this step, Image Assistant opens your applications one after another, identifies their launch dependencies, and performs optimizations to ensure that applications launch quickly. These are required steps that are performed on all applications in the list.

To optimize your applications

1. In Image Assistant, in **4. Optimize**, choose **Launch**.
2. AppStream 2.0 automatically launches the first application in your list. When the application completely starts, provide any required input to perform the first run experience for the application. For example, a web browser may prompt you to import settings before it is completely up and running.
3. After you complete the first run experience and verify that the application performs as expected, choose **Continue**. If you added more than one application to your image, each application opens automatically. Repeat this step for each application as needed, leaving all applications running.
4. When you're done, the next tab in Image Assistant, **5. Configure Image**, automatically displays.

Step 6: Finish Creating Your Image

In this step, choose an image name and finish creating your image.

To create the image

1. Type a unique image name, and an optional image display name and description. The image name cannot begin with "Amazon," "AWS," or "AppStream."

   You can also add one or more tags to the image. To do so, choose **Add Tag**, and type the key and value for the tag. To add more tags, repeat this step. For more information, see Tagging Your Amazon AppStream 2.0 Resources. When you're done, choose **Next**.

   > **Note**
   >
   > If you choose a base image that is published by AWS on or after December 7, 2017, the option **Always use the latest agent version** appears, selected by default. We recommend that you leave this option selected so that streaming instances that are launched from the image always use the latest version of the agent. If you disable this option, you can't enable it again after you finish creating the image. For information about the latest release of the AppStream 2.0 agent, see AppStream 2.0 Agent Release Notes.

2. In **6. Review**, verify the image details. To make changes, choose **Previous** to navigate to the appropriate Image Assistant tab, make your changes, and then proceed through the steps in Image Assistant as needed.
3. After you finish reviewing the image details, choose **Disconnect and Create Image**.
4. The remote session disconnects within a few moments. When the **Lost Connectivity** message appears, close the browser tab. While the image is created, the image builder status appears as **Snapshotting**. You cannot connect to the image builder until this process finishes.
5. Return to the console and navigate to **Images**, **Image Registry**. Verify that your new image appears in the list.

   While your image is being created, the image status in the image registry of the console appears as **Pending** and you cannot connect to it.
6. Choose the **Refresh** icon periodically to update the status. After your image is created, the image status changes to **Available** and the image builder is automatically stopped.

   To continue creating images, start the image builder and connect to it from the console, or create a new image builder.

> **Note**
>
> After you create your image, you are responsible for maintaining updates for the Windows operating system. To do so, you can use managed AppStream 2.0 image updates. You are also responsible for maintaining updates for your applications and their dependencies. For more information, see Keep Your AppStream 2.0 Image Up-to-Date.
> To add other applications, update existing applications, or change image settings, you must start and reconnect to the image builder that you used to create the image. Or, if you deleted that image builder, launch a new image builder that is based on your image. Then, make your changes and create a new image.

Step 7 (Optional): Tag and Copy an Image

You can add one or more tags to an image during image creation or after you create an image. You can also copy the image within the same Region or to a new Region within the same Amazon Web Services account. Copying a source image results in an identical but distinct destination image.

AWS does not copy any user-defined tags, however. Also, you can only copy custom images that you create, not the base images that are provided by AWS.

> **Note**
>
> You can copy up to two images at the same time to a destination. If the destination to which you are copying an image is at the image limit, you receive an error. To copy the image in this case, you must first remove images from the destination. After the destination is below the image quota (also referred to as limit), initiate the image copy from the source Region. For more information, see Amazon AppStream 2.0 Service Quotas.

To add tags to an existing image

1. In the navigation pane, choose **Images**, **Image Registry**.
2. In the image list, select the image to which you want to add tags.
3. Choose **Tags**, choose **Add/Edit Tags**, choose **Add Tag**, specify the key and value for the tag, and then choose **Save**.

For more information, see Tagging Your Amazon AppStream 2.0 Resources.

To copy an image

Copying an image across geographically diverse regions enables you to stream applications from multiple regions based on the same image. By streaming your applications in closer proximity to your users, you can improve your users' experience streaming applications with AppStream 2.0.

1. In the navigation pane, choose **Images**, **Image Registry**.
2. In the image list, select the image that you want to copy.
3. Choose **Actions**, **Copy**.
4. In the **Copy Image** dialog box, specify the following information, and then choose **Copy Image:**
   - For **Destination region**, choose the region to which to copy the new image.
   - For **Name**, specify a name that the image will have when it is copied to the destination.
   - For **Description** (optional), specify a description that the image will have when it is copied to the destination.

5. To check on the progress of the copy operation, return to the console and navigate to **Images**, **Image Registry**. Use the navigation bar to switch to the destination region (if applicable), and confirm that your new image appears in the list of images.

   The new image first appears with a status of **Copying** in the image registry of your console. After the image is successfully created, the status of the image changes to **Available**, which means that you can use the image to launch a stack and stream your applications.

Step 8: 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**.

Administer Your Amazon AppStream 2.0 Images

Available images are listed in the **Image Registry** in the AppStream 2.0 console, and categorized by visibility as follows:

- **Public** — Base images that are owned and made available by AWS. Base images include the latest Windows operating system and the AppStream 2.0 agent software. You can use these base images to create new images that include your own applications. For information about the base images released by AWS, see AppStream 2.0 Base Image and Managed Image Update Release Notes.
- **Private** — Images that you create and own, and that you have not shared with other AWS accounts.
- **Shared with others** — Images that you create and own, and that you have shared with one or more AWS accounts in the same AWS Region. When you share an image with another AWS account, you can specify whether the image can be used for an image builder (to create a new image), for a fleet, or both.

- **Shared with me** — Images that are created and owned by another AWS account in the same AWS Region, and that are shared with your AWS account. Depending on the permissions that the owner provided when sharing the image with your account, you can use this image for image builders, for fleets, or both.

Contents

- Delete a Private Image
- Copy an Image That You Own to Another AWS Region
- Share an Image That You Own With Another AWS Account
- Stop Sharing an Image That You Own
- Keep Your AppStream 2.0 Image Up-to-Date
- Windows Update and Antivirus Software on AppStream 2.0
- Programmatically Create a New Image

Delete a Private Image

You can delete your private images when you no longer need them. You can't delete an image that is used by fleets or shared with other AWS accounts. To delete an image that is used by fleets or shared, you must first remove the image from any fleets and remove all image sharing permissions. After you delete an image, you can't recover it.

To delete a private image

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. In the navigation pane, choose **Images**, **Image Registry**.
3. In the image list, select the private image you want to delete.
4. Choose **Actions**, **Delete**, then choose **Delete** again.

   The image is removed from the image registry and deleted.

Copy an Image That You Own to Another AWS Region

You can copy images that you own to another AWS Region. Using the same image across different AWS Regions can help simplify global deployments of your applications on AppStream 2.0. By

deploying your applications in the AWS Regions that are geographically closest to your users, you can help provide your users with a more responsive experience.

To copy an image that you own to another AWS Region

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. In the navigation pane, choose **Images**, **Image Registry**.
3. In the image list, select the image that you want to copy to another AWS Region.
4. Choose **Actions**, **Copy**.
5. In the **Copy image** dialog box, in **Destination region**, select the AWS Region that you want to copy the image to.
6. Type a unique name and optionally, a description for the image in **Destination region**.
7. Choose **Copy Image**.

Share an Image That You Own With Another AWS Account

AppStream 2.0 images are a regional resource, so you can share an image that you own with other AWS accounts within the same AWS Region. Doing so can be helpful in several different scenarios. For example, if you separate your development and production resources by using different AWS accounts, you can create an image by using your development account. Then you can share the image with your production account. If your organization is an independent software vendor (ISV), you can share optimized images with your customers. Optimized images that have the required applications already installed and configured let your customers get started with your applications quickly, so that they won't need to install and configure those applications themselves.

When you share an image with another AWS account, you specify whether the destination account can use the image in a fleet or create new images by creating an image builder. You continue to own images that you share. This way, you can add, change, or remove permissions as needed for your shared images.

If you share an image with an account and grant the account fleet permissions, the shared image can be used to create or update fleets in that account. If you remove these permissions later, the account can no longer use the image. For fleets in the account that use the shared image, the desired capacity is set to 0, which prevents new fleet instances from being created. Existing sessions continue until the streaming session ends. For new fleet instances to be created, the fleet in that account must be updated with a valid image.

If you share an image with an account and grant the account image builder permissions, the shared image can be used to create image builders and images in that account. If you remove these permissions later, image builders and images that were created from your image are not affected.

> **Important**
>
> After you share an image with an account, you can't control image builders or images in the account that are created from your image. For this reason, grant image builder permissions to an account only if you want to enable the account to make a copy of your image, and retain access to the copy after you stop sharing your image.

To share an image that you own with another AWS account

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. In the navigation pane, choose **Images**, **Image Registry**.
3. In the image list, select the image that you want to share.
4. Choose **Actions**, **Share**.
5. In the **Share image** dialog box, choose **Add account**.
6. Type the 12-digit AWS account ID of the account that you want to share the image with, and then select whether the account can do one or both of the following:

   - Use the image to launch an image builder, if you want to create a new image.
   - Use the image with a fleet.

   To remove an account from the list of accounts that the image is shared with, in the row for the account you want to remove, choose the X icon to the right of the **Use for fleet** option.
7. To share the image with more AWS accounts, repeat step 6 for each account that you want to share the image with.
8. Choose **Share Image**.

To add or update image sharing permissions for an image that you own

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. In the navigation pane, choose **Images**, **Image Registry**.

3. In the image list, select the image that you want to change the permissions for.
4. Below the image list, choose the **Permissions** tab for the image you selected, then choose **Edit**.
5. In the **Edit image permissions** dialog box, select or clear one or both of the following image sharing options as needed for one or more AWS accounts. If you clear both options for an account, the image is no longer shared with that account.

   - Use the image to launch an image builder, if you want to create a new image.
   - Use the image with a fleet.

   To remove an account from the list of accounts that the image is shared with, in the row for the account you want to remove, choose the X icon to the right of the **Use for fleet** option.
6. To edit image sharing permissions for more AWS accounts, repeat step 5 for each account you want to update permissions for.
7. Choose **Update image sharing permissions**.

Stop Sharing an Image That You Own

Follow these steps to stop sharing an image that you own with any other AWS account.

To stop sharing an image that you own with any other AWS account

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. In the navigation pane, choose **Images**, **Image Registry**.
3. In the image list, select the image that you want to change the permissions for.
4. Below the image list, choose the **Permissions** tab for the image you selected, then choose **Edit**.
5. In the **Edit image permissions** dialog box, in the row for all AWS accounts that the image is shared with, choose the X icon to the right of the **Use for fleet** option.
6. Choose **Update image sharing permissions**.

Keep Your AppStream 2.0 Image Up-to-Date

You can keep your AppStream 2.0 image up-to-date by doing either of the following:

- Update an Image by Using Managed AppStream 2.0 Image Updates – This update method provides the latest Windows operating system updates and driver updates, and the latest AppStream 2.0 agent software.
- Update the AppStream 2.0 Agent Software by Using Managed AppStream 2.0 Agent Versions – This update method provides the latest AppStream 2.0 agent software.

Update an Image by Using Managed AppStream 2.0 Image Updates

AppStream 2.0 provides an automated way to update your image with the latest Windows operating system updates, driver updates, and AppStream 2.0 agent software. With managed AppStream 2.0 image updates, you select the image that you want to update. AppStream 2.0 creates an image builder in the same AWS account and Region to install the updates and create the new image. After the new image is created, you can test it on a pre-production fleet before updating your production fleets or sharing the image with other AWS accounts.

> **Note**
>
> After your new image is created, you're responsible for maintaining updates for the Windows operating system. To do so, you can continue using managed AppStream 2.0 image updates.
>
> You're also responsible for maintaining your applications and their dependencies. To add other applications, update existing applications, or change image settings, you must start and reconnect to the image builder that you used to create the image. Or, if you deleted that image builder, launch a new image builder that is based on your image. Then, make your changes and create a new image.

Prerequisites

The following are prerequisites and considerations for working with managed image updates.

- Make sure that your AppStream 2.0 account quotas (also referred to as limits) are sufficient to support the creation of a new image builder and a new image. To request a quota increase, you can use the Service Quotas console at https://console.aws.amazon.com/servicequotas/. For information about default AppStream 2.0 quotas, see Amazon AppStream 2.0 Service Quotas.
- You must own the image that you update. You can't update an image that is shared with you.

- When AppStream 2.0 creates an image builder to install the latest Windows operating system updates, driver updates, and AppStream 2.0 agent software, and creates the new image, you're charged for the image builder instance while it's updating.
- Supported images must be created from a base image released on 2017-07-24T00:00:00Z or later.
- English and Japanese are supported display languages. For more information, see Specify a Default Display Language.
- Use the latest version of SSM Agent. For version information, see the section called "Base Image and Managed Image Update Release Notes". For installation information, see Manually install SSM Agent on EC2 instances for Windows Server.

How to Update an Image by Using Managed AppStream 2.0 Image Updates

To update an AppStream 2.0 image with the latest patches, driver updates, and AppStream 2.0 agent software, perform the following steps.

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. In the navigation pane, choose **Images**, **Image Registry**.
3. In the image list, choose the image that you want to update. Verify that the status of the image is **Available**.
4. Choose **Actions**, **Update**.
5. In the **Update image** dialog box, do the following:
   - For **New image name**, enter an image name that is unique within the AWS account and Region. The image name can't begin with "Amazon," "AWS," or "AppStream."
   - For **New image display name**, you can optionally enter a name to display for the image.
   - For **New image description**, you can optionally provide a description for the image.
   - For **Tags**, you can choose **Add Tag**, and type the key and value for the tag. To add more tags, repeat this step. For more information, see Tagging Your Amazon AppStream 2.0 Resources.
6. Choose **Update image**.

   If your current image is already up to date, a message notifies you.
7. In the navigation pane, choose **Images**, and then choose **Image Builder**.
8. In the list of image builders, verify that a new image builder appears in the **Updating** state. The name of the image builder includes a random 10-digit suffix.

The image builder is the smallest size in the instance family that you chose for the new image in step 5. No subnet is specified because the image builder is not attached to your virtual private cloud (VPC).

9. Choose **Image Registry** and verify that your new image appears in the list.

   While your image is being created, the image status in the image registry of the console appears as **Creating**.

10. After your image is created, AppStream 2.0 performs a qualification process to verify that the image works as expected.

    During this time, the image builder, which is also used for this process, appears in the **Image Builder** list with a status of **Pending Qualification**.

11. After the qualification process successfully completes, a **Success** message appears at the top of the console and the image status in the image registry appears as **Available**.

    In addition, the image builder that AppStream 2.0 created is deleted automatically.

    > **Note**
    >
    > Depending on the volume of Windows operating system updates, it might take several hours for an image update to complete. If an issue prevents the image from being updated, a red icon with an exclamation point appears next to the image name, and the image status in the image registry appears as **Failed**. If this occurs, select the image, choose the **Notifications** tab, and review any error notifications. For more information, see the information in the Image Internal Service section of the documentation for troubleshooting notification codes.
    > If the qualification process is not successful, the image builder that AppStream 2.0 created is still deleted automatically.

12. After AppStream 2.0 creates the new image, test the image on a pre-production fleet. After you verify that your applications work as expected, update your production fleet with the new image.

Update the AppStream 2.0 Agent Software by Using Managed AppStream 2.0 Agent Versions

AppStream 2.0 provides an automated way to update your image builder with newer AppStream 2.0 agent software. Doing so enables you to create a new image whenever a new version of the agent is released. You can then test the image before updating your production fleets. For more information about how to manage the AppStream 2.0 agent software, see Manage AppStream 2.0 Agent Versions.

> **Note**
>
> You're responsible for installing and maintaining the updates for the Windows operating system, your applications, and their dependencies.

To keep your AppStream 2.0 image updated with the latest Windows operating system updates, do one of the following:

- Install your applications on the latest base image each time a new image is released.
- Install the updates for the Windows operating system, your applications, and their dependencies on an existing image builder.
- Install the updates for the Windows operating system, your applications, and their dependencies on a new image builder from an existing image.

After you create a new image with the latest Windows operating system, applications and their dependencies, and the AppStream 2.0 agent software, test the image on a development fleet. After you verify that your applications work as expected, update your production fleet with the new image.

Windows Update and Antivirus Software on AppStream 2.0

AppStream 2.0 streaming instances are non-persistent. When a user streaming session ends, AppStream 2.0 terminates the instance used by the session and, depending on your scaling policies, provisions a new instance to replace it in your fleet. All fleet instances are provisioned from the same image. Because images cannot be changed once created, all fleet instances used in user streaming sessions have only the Windows and application updates that were installed on the underlying image when the image was created. In addition, because a fleet instance used

for a streaming session terminates at the end of the session, any updates made to Windows or to applications on the instance during the streaming session will not persist to future sessions by the same user or other users.

> **Note**
>
> If you enabled application settings persistence for your stack, AppStream 2.0 persists Windows and application configuration changes made by a user to future sessions for the same user if those configuration changes are stored in the user's Windows profile. However, the application settings persistence feature persists only Windows and application configuration settings. It does not persist software updates to Windows or applications on the streaming instance.

For these reasons, AppStream 2.0 takes the following approach to Windows Update and antivirus software on AppStream 2.0 instances.

Windows Update

Windows Update is not enabled by default on AppStream 2.0 base images. If you enable Windows Update on an image builder and then try to create an image, Image Assistant displays a warning and disables Windows Update during the image creation process. To ensure that your fleet instances have the latest Windows updates installed, we recommend that you install Windows updates on your image builder, create a new image, and update your fleet with the new image on a regular basis.

Antivirus Software

If you choose to install antivirus software on your image, we recommend that you do not enable automatic updates for the antivirus software. Otherwise, the antivirus software may attempt to update itself with the latest definition files or other updates during user sessions. This may affect performance. In addition, any updates made to the antivirus software will not persist beyond the current user session. To ensure that your fleet instances always have the latest antivirus updates, we recommend that you do either of the following:

- Update your image builder and create a new image on a regular basis (for example, by using the Image Assistant CLI operations).
- Use an antivirus application that delegates scanning or other operations to an always-up-to-date external server.

> **Note**
>
> Even if you do not enable automatic updates for your antivirus software, the antivirus software may perform hard drive scans or other operations that may impact the performance of your fleet instances during user sessions.

AppStream 2.0 Windows Server 2012 R2 base images do not include any antivirus software. On AppStream 2.0 Windows Server 2016 and Windows Server 2019 base images published on or after September 10, 2019, Windows Defender is not enabled by default. On AppStream 2.0 Windows Server 2016 and Windows Server 2019 base images published on June 24, 2019, Windows Defender is enabled by default.

To enable Windows Defender manually

If Windows Defender is not enabled on your base image, you can enable it manually. To do so, complete the following steps.

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. In the left navigation pane, choose **Images**, **Image Builder**.
3. Choose the image builder on which to enable Windows Defender, verify that it is in the **Running** state, and choose **Connect**.
4. Log in to the image builder with the local **Administrator** account or with a domain account that has local administrator permissions.
5. Open Registry Editor.
6. Navigate to the following location in the registry: **HKLM\SOFTWARE\Policies\Microsoft\Windows Defender\DisableAntiSpyware**.
7. To edit this registry key, double-click it, or right-click the registry key, and choose **Modify**.
8. In the **Edit DWORD (32-bit) Value** dialog box, in **Value data**, change **1** to **0**.
9. Choose **OK**.
10. Close Registry Editor.
11. Open the Microsoft Management Console (MMC) **Services** snap-in (`services.msc`).
12. In the list of services, do either of the following:

    - Right-click **Windows Defender Antivirus Service**, and choose **Start**.

- Double-click **Windows Defender Antivirus Service**, choose **Start** in the properties dialog box, and then choose **OK**.

13. Close the **Services** snap-in.

Programmatically Create a New Image

You can create AppStream 2.0 images programmatically by connecting to an image builder and using the Image Assistant command line interface (CLI) operations. For more information, see Create Your AppStream 2.0 Image Programmatically by Using the Image Assistant CLI Operations.

Create Your AppStream 2.0 Image Programmatically by Using the Image Assistant CLI Operations

You can create Amazon AppStream 2.0 images by connecting to an image builder and using the Image Assistant graphical user interface (GUI) or command line interface (CLI) operations. The Image Assistant CLI operations provide functionality that is similar to the Image Assistant GUI. With these operations, you can programmatically do the following:

- Manage the applications that are included in an image.
- Save, update, and reset default application settings.
- Enable or disable the AppStream 2.0 dynamic application framework.
- Specify tags.
- Create an image.

You can use these operations to integrate AppStream 2.0 image creation with your continuous integration or deployment software development process.

To work with the Image Assistant CLI operations, use the command line shell of your choice on an image builder. For example, you can use the Windows command prompt or PowerShell.

> **Note**
>
> The image builder must use a version of the AppStream 2.0 agent that is released on or after July 26, 2019. If you don't have an image builder, you must create one. For more information, see Launch an Image Builder to Install and Configure Streaming Applications.

Contents

- Creating Default Application and Windows Settings with the Image Assistant CLI operations
- Optimizing the Launch Performance of Your Applications with the Image Assistant CLI Operations
- Process Overview for Programmatically Creating an AppStream 2.0 Image
- Image Assistant CLI Operations for Creating and Managing Your AppStream 2.0 Image

Creating Default Application and Windows Settings with the Image Assistant CLI operations

You can create default application and Windows settings so that your users can get started with their applications quickly. When you create these settings, AppStream 2.0 replaces the Windows default user profile with the profile that you configure. The Windows default user profile is then used to create the initial settings for users in the fleet instance. If you create these settings by using the Image Assistant CLI operations, your application installer, or the automation, should modify the Windows default user profile directly.

To overwrite the Windows default user profile with that of another Windows user, you can also use the Image Assistant `update-default-profile` CLI operation.

For more information about configuring default application and Windows settings, see *Creating Default Application and Windows Settings for Your AppStream 2.0 Users* in Default Application and Windows Settings and Application Launch Performance.

Optimizing the Launch Performance of Your Applications with the Image Assistant CLI Operations

AppStream 2.0 lets you optimize the launch performance of your applications for your users' streaming sessions. When you do so by using the Image Assistant CLI operations, you can specify the files to optimize for your application launch. Adding files to the application optimization manifest reduces the time that it takes for the application to launch for the first time on a new fleet instance. However, this also increases the time that it takes for the fleet instances to be made available to users. The optimization manifest is a line-delimited text file that is per application.

> **Note**
>
> If you onboard application optimization manifests by using both the Image Assistant CLI operations and the Image Assistant GUI, the manifests are merged.

Following is an example of an application optimization manifest file:

```
C:\Program Files (x86)\Notepad++\autoCompletion
C:\Program Files (x86)\Notepad++\localization
C:\Program Files (x86)\Notepad++\plugins
C:\Program Files (x86)\Notepad++\themes
C:\Program Files (x86)\Notepad++\updater
C:\Program Files (x86)\Notepad++\userDefineLangs
C:\Program Files (x86)\Notepad++\change.log
C:\Program Files (x86)\Notepad++\config.xml
C:\Program Files (x86)\Notepad++\contextMenu.xml
C:\Program Files (x86)\Notepad++\doLocalConf.xml
C:\Program Files (x86)\Notepad++\functionList.xml
C:\Program Files (x86)\Notepad++\langs.model.xml
C:\Program Files (x86)\Notepad++\license.txt
C:\Program Files (x86)\Notepad++\notepad++.exe
C:\Program Files (x86)\Notepad++\readme.txt
C:\Program Files (x86)\Notepad++\SciLexer.dll
C:\Program Files (x86)\Notepad++\shortcuts.xml
C:\Program Files (x86)\Notepad++\stylers.model.xml
```

For more information about optimizing the launch performance of your applications, see *Optimizing the Launch Performance of Your Applications* in Default Application and Windows Settings and Application Launch Performance.

Process Overview for Programmatically Creating an AppStream 2.0 Image

You can use the Image Assistant CLI operations with your application installation automation to create a fully programmatic AppStream 2.0 image creation workflow. After your application installation automation completes, but before the image is created, use the Image Assistant CLI operations to specify the following:

- The executable files that your users can launch

- The optimization manifests for your applications
- Other AppStream 2.0 image metadata

The following high-level overview describes the process for programmatically creating an AppStream 2.0 image.

1. Use your application installation automation to install the required applications on your image builder. This installation may include applications that your users will launch, any dependencies, and background applications.
2. Determine the files and folders to optimize.
3. If applicable, use the Image Assistant `add-application` CLI operation to specify the application metadata and optimization manifest for the AppStream 2.0 image.
4. To specify additional applications for the AppStream 2.0 image, repeat steps 1 through 3 for each application as needed.
5. If applicable, use the Image Assistant `update-default-profile` CLI operation to overwrite the default Windows profile and create default application and Windows settings for your users.
6. Use the Image Assistant `create-image` CLI operation to create the image.

Image Assistant CLI Operations for Creating and Managing Your AppStream 2.0 Image

This section describes the Image Assistant CLI operations that you can use to create and manage your AppStream 2.0 image.

On Windows image builders, the executable file that includes the command line interface is located at: C:\Program Files\Amazon\Photon\ConsoleImageBuilder\Image-Assistant.exe. For your convenience, this executable file is included in the Windows PATH variable. This lets you call the Image Assistant CLI operations without specifying the absolute path to the executable file. To call these operations, type the **image-assistant.exe** command.

On Linux image builders, the image assistant tool is located at /usr/local/appstream/image-assistant/AppStreamImageAssistant, with a symbolic link at /bin/AppStreamImageAssistant.

`help` operation

Retrieves a list of all Image Assistant CLI operations. For each operation in the list, a description and usage syntax is provided. To display help for a specific operation, type the name of the operation and specify the **--help** parameter. For example:

```
add-application --help
```

Synopsis

```
help
```

Output

Prints to standard out the list of available operations with a description of their function.

`add-application` operation

Adds the application to the application list for AppStream 2.0 users. Applications in this list are included in the application catalog. The application catalog displays to users when they sign in to an AppStream 2.0 streaming session.

> **Note**
>
> If you need to make changes to an application configuration, remove the application and add the application with the updated settings.

Synopsis

```
add-application
--name <value>
--absolute-app-path <value>
[--display-name <value>]
[--absolute-icon-path <value>]
[--working-directory <value>]
[--launch-parameters <""-escaped value>]
[--absolute-manifest-path <value>]
```

Options

`--name` (string)

A unique name for the application. The maximum length is 256 characters. You can add up to 50 applications. You cannot use whitespace characters.

`--absolute-app-path` (string)

The absolute path to the executable file, batch file, or script for the application. The path must point to a valid file.

`--display-name` (string)

The name to display for the application in the application catalog. If you don't specify a display name, AppStream 2.0 creates a name that is derived from the executable file name. The name is created without the file extension and with underscores in place of spaces. The maximum length is 256 characters.

`--absolute-icon-path` (string)

The absolute path to the icon for the application. The path must point to a valid icon file that is one of the following types: .jpg, .png, or .bmp. The maximum dimensions are: 256 px x 256 px. If you don't specify a path, the default icon for the executable file is used, if available. If a default icon is not available for the executable file, a default AppStream 2.0 application icon is used.

`--working-directory` (string)

The initial working directory for the application when the application is launched.

`--absolute-manifest-path` (string)

The path to a new line-delimited text file. The file specifies the absolute paths of the files to optimize before the fleet instance is made available to a user for streaming. The path must point to a valid text file.

Message output

Exit code	Message printed to standard out	Description
0	{"status": 0, "message": "Success"}	The application was added successfully.

Exit code	Message printed to standard out	Description
1	{"status": 1, "message": "Administrator privileges are required to perform this operation"}	Administrator privileges are required to complete the operation.
1	{"status": 1, "message": "Unable to add more than 50 apps to the catalog."}	The application could not be added because the maximum number of applications that can be added to the AppStream 2.0 application catalog is 50.
1	{"status": 1, "message": "Name is not unique"}	An application with that name already exists in the AppStream 2.0 application catalog.
1	{"status": 1, "message": "File not found (absolute-app-path)"}	The file that was specified for `absolute-app-path` could not be found.
1	{"status": 1, "message": "Unsupported file extension"}	The `Absolute-app-path` parameter only supports the following file types: .exe and .bat.
1	{"status": 1, "message": "Directory not found (working-directory)"	The directory that was specified for `working-directory` could not be found.
1	{"status": 1, "message": "Optimization-manifest not found: <filename>"}	The file that was specified for `optimization-manifest` could not be found.
1	{"status": 1, "message": "File not found: <filename>"}	A file that was specified within the optimization manifest could not be found.

Exit code	Message printed to standard out	Description
255	{"status": 255, "message": <error message>}	An unexpected error occurred. Try the request again. If the error persists, contact AWS Support for assistance. For more information, see AWS Support Center.

`remove-application` operation

Removes an application from the application list for the AppStream 2.0 image. The application is not uninstalled or modified, but users will not be able to launch it from the AppStream 2.0 application catalog.

Synopsis

```
remove-application
--name <value>
```

Options

`--name` (string)

The unique identifier of the application to remove.

Message output

Exit code	Message printed to standard out	Description
0	{"status": 0, "message": "Success"}	The application was removed successfully.
1	{"status": 1, "message": "Administrator privileges are required to perform this operation"}	Administrator privileges are required to complete the operation.

Exit code	Message printed to standard out	Description
1	{"status": 1, "message": "App not found"}	The application that was specified could not be found in the AppStream 2.0 application catalog.
255	{"status": 255, "message": <error message>}	An unexpected error occurred. Try the request again. If the error persists, contact AWS Support for assistance. For more information, see AWS Support Center.

`list-applications` operation

Lists all of the applications that are specified in the application catalog.

Synopsis

```
list-applications
```

Message output

Exit code	Message printed to standard out	Description
0	{"status": 0, "message" : "Success", "applicat ions": [{..app1.. }, { ..app2.. }]}	List of applications in the AppStream 2.0 applicati on catalog.
255	{"status": 255, "message": <error message>}	An unexpected error occurred. Try the request again. If the error persists, contact AWS Support for assistance. For more information, see AWS Support Center.

`update-default-profile` operation

Copies the specified Windows user's profile to the Windows default user profile. New users who stream inherit the settings stored in the specified profile.

> **Note**
>
> This operation is not supported by the Linux image assistant CLI tool.

Synopsis

```
update-default-profile
[--profile <value>]
```

Options

`--profile` (string)

The name of the user whose Windows profile will be copied to the Windows default user profile. Use the following format for the name:

"<domain>\<username>"

If your image builder isn't joined to a Microsoft Active Directory domain, enter a period "." for the domain. If you don't specify a user, the AppStream 2.0 Template User account is used.

Message output

Exit code	Message printed to standard out	Description
0	{"status": 0, "message": "Success"}	The user settings were successfully copied to the default Windows profile.
1	{"status": 1, "message": "Administrator privilege s are required to	Administrator privileges are required to complete the operation.

Exit code	Message printed to standard out	Description
	perform this operation "}	
1	{"status": 1, "message" : "Unable to copy file or folder: <path>. <reason>"}	The user settings could not be copied because a file or folder was unavailable.
1	{"status": 1, "message": "Cannot copy a domain user when not joined to a domain""}	A Microsoft Active Directory domain user was specified, but the image builder is not joined to an Active Directory domain.
255	{"status": 255, "message": <error message>}	An unexpected error occurred. Try the request again. If the error persists, contact AWS Support for assistance. For more information, see AWS Support Center.

`reset-user-profile` operation

Deletes the Windows user profile for the specified user.

> **Note**
>
> This operation is not supported by the Linux image assistant CLI tool.

Synopsis

```
reset-user-profile
[--profile <value>]
```

Options

`--profile` (string)

The name of the Windows user whose Windows profile will be deleted. Use the following format for the name:

"<domain>\<username>"

If your image builder isn't joined to a Microsoft Active Directory domain, enter a period "." for the domain.

Message output

Exit code	Message printed to standard out	Description
0	{"status": 0, "message": "Success"}	The specified user settings were deleted successfully.
1	{"status": 1, "message": "Administrator privileges are required to perform this operation"}	Administrator privileges are required to complete the operation.
1	{"status": 1, "message": "Unable to copy file or folder: <path>. <reason>"}	The user settings could not be reset because a file or folder was unavailable.
1	{"status": 1, "message": "Cannot copy a domain user when not joined to a domain""}	A Microsoft Active Directory domain user was specified, but the image builder is not joined to an Active Directory domain.
255	{"status": 255, "message": <error message>}	An unexpected error occurred. Try the request again. If the error persists, contact AWS Support for assistance. For more information, see AWS Support Center.

`create-image` operation

Starts the image creation workflow, resulting in an AppStream 2.0 image that can be used for AppStream 2.0 fleets.

Synopsis

```
create-image
--name <value>
[--description <value>]
[--display-name <value>]
[--enable-dynamic-app-catalog] | [--no-enable-dynamic-app-catalog]
[--use-latest-agent-version] | [--no-use-latest-agent-version]
[--tags <value>]
[--dry-run]
```

Options

`--name` (string)

The name for the AppStream 2.0 image. The name must be unique within the Amazon Web Services account and AWS Region. The maximum length is 100 characters. Allowed characters are:

a-z, A-Z, 0-9, underscores (_), hyphens (-), and periods (.)

The image name cannot start with any of the following prefixes: 'aws', 'appstream', and 'amazon'. These prefixes are reserved for AWS use.

`--description` (string)

The description to display for the image. The maximum length is 256 characters.

`--display-name` (string)

The name to display for the image. The maximum length is 256 characters.

`--enable-dynamic-app-catalog` | `--no-enable-dynamic-app-catalog`

Enables or disables support for the AppStream 2.0 dynamic application framework. If you don't specify either parameter, support for the dynamic application framework is not enabled.

The dynamic application framework provides operations within an AppStream 2.0 streaming instance that you can use to build a dynamic app provider. Dynamic app providers can use

these operations to modify the catalog of applications that your users can access in real time. For more information, see Use the AppStream 2.0 Dynamic Application Framework to Build a Dynamic App Provider.

`--use-latest-agent-version | --no-use-latest-agent-version`

Specifies whether to pin the image to the version of the AppStream 2.0 agent that is currently installed, or to always use the latest agent version. If you don't specify either parameter, the image is pinned to the version of the AppStream 2.0 agent that is currently installed. For more information, see Manage AppStream 2.0 Agent Versions.

`--tags` (string)

The tags to associate with the image. A tag is a key-value pair. Use the following format:

--tags "mykey" "myval" "mykey2" "myval2"

For more information about tags, see Tagging Your Amazon AppStream 2.0 Resources.

`--dry-run` (string)

Performs validation without creating the image. Use this command to identify whether your image has any issues before you create it.

Message output

Exit code	Message printed to standard out	Description
0	{"status": 0, "message": "Success"}	The workflow to create the image was initiated successfully.
1	{"status": 1, "message": "Administrator privileges are required to perform this operation "}	Administrator privileges are required to complete the operation.
1	{"status": 1, "message" : "An image with the	An image with the specified name already exists in the Amazon Web Services account.

Exit code	Message printed to standard out	Description
	given name already exists"}	
1	{"status": 1, "message": "Invalid value (tags)"}	The specified tags are not valid.
255	{"status": 255, "message": <error message>}	An unexpected error occurred. Try the request again. If the error persists, contact AWS Support for assistance. For more information, see AWS Support Center.

Create Your Linux-Based Images

You can create Linux-based Amazon AppStream 2.0 images by connecting to a Linux image builder, installing the applications you need, creating default application settings and environment variables, and using a command line interface (CLI) tool or Image Assistant (GUI) tool to add these applications to the application catalog. To open the GUI tool, search for **Image Assistant** in the list of applications.

Contents

- Creating Default Application Settings for Your Users
- Creating Default Environment Variables for Your Linux Users
- Optimizing the Launch Performance of Your Linux Applications
- Creating Session Scripts
- Using the Image Assistant CLI Tool for Linux
- Enabling and Disabling Webcam Support
- Tutorial: Create a Custom Linux-Based AppStream 2.0 Image
- Tutorial: Enable Japanese Support for Your Linux Images

Creating Default Application Settings for Your Users

Contents

- Step 1: Install Linux Applications on the Image Builder
- Step 2: Create a TemplateUser Account
- Step 3: Create Default Application Settings
- Step 4: Save Default Application Settings
- Step 5: Test Default Application Settings (optional)
- Step 6: Clean Up

Step 1: Install Linux Applications on the Image Builder

In this step, you connect a Linux image builder and install your applications on the image builder.

To install applications on the image builder

1. Connect to the image builder by doing either of the following:

   - Use the AppStream 2.0 console (for web connections only)
   - Create a streaming URL (for web or AppStream 2.0 client connections)

   > **Note**
   >
   > You will be logged in as an ImageBuilderAdmin user to the Amazon Linux GNOME desktop and have root admin privileges.

2. Install the applications that you need. For example, to install a Chromium browser from a public yum repo, first open the Terminal application, then run the following command:

   [ImageBuilderAdmin]$ sudo yum update && sudo yum install chromium.x86_64

Step 2: Create a TemplateUser Account

In this step, you create a TemplateUser account, which creates default application settings for your streaming users.

To create a TemplateUser Account

1. Create a TemplateUser account that has no root permissions. For example, in a Terminal window, run the following commands to create TemplateUser on the image builder:

```
[ImageBuilderAdmin]$ sudo useradd -m TemplateUser

[ImageBuilderAdmin]$ echo -e '<password>\npassword>\n' | sudo passwd TemplateUser
```

2. Switch to the TemplateUser account:

```
[ImageBuilderAdmin]$ su - TemplateUser
```

Step 3: Create Default Application Settings

In this step, you create default application settings for your AppStream 2.0 users. Doing this enables your users to get started with applications quickly during their AppStream 2.0 streaming sessions, without the need to create or configure these settings themselves.

To create default application settings for your users

1. Launch the application that you want to create the default settings for. For example, in a Terminal window, run following command to launch Chromium browser:

```
[TemplateUser]$ chromium-browser
```

2. Configure the settings of the application. For example, set the home page of the Chromium browser as `https://aws.amazon.com`.
3. Close the applications.
4. Log out:

```
[TemplateUser]$ logout
```

Step 4: Save Default Application Settings

In this step, you copy the default application settings you added to the **/etc/skel/** directory, and make them available to your streaming users.

To save default application settings

- Run the following command in a Terminal window to copy the default application settings for your streaming users:

```
[ImageBuilderAdmin]$ sudo cp -r -f /home/TemplateUser/. /etc/skel
```

Step 5: Test Default Application Settings (optional)

In this step, verify that the applications you've added run correctly, and the default application settings work as expected.

To test your applications and default settings on an image builder

1. Create a test user that has no root permissions. For example, in a **Terminal** window, run the following commands to create **test-user** on the image builder:

   [ImageBuilderAdmin]$ sudo useradd -m test-user

   [ImageBuilderAdmin]$ echo -e '*password*>\n<*password*>\n' | sudo passwd test-user

2. Switch to the test user:

   [ImageBuilderAdmin]$ su - test-user

3. Launch the application (e.g., Chromium) as the test user:

   [test-user]$ /usr/bin/chromium-browser

4. Verify that the default settings are available for the test user (e.g., the Chromium home page is https://aws.amazon.com/).

5. Log out:

   [TestUser]$ logout

Step 6: Clean Up

Finally, your last step is to clean up.

To clean up

1. Delete TemplateUser:

   [ImageBuilderAdmin]$ sudo killall -u TemplateUser

   [ImageBuilderAdmin]$ sudo userdel -r TemplateUser

2. Delete TestUser (not required if you skipped step 5):

   [ImageBuilderAdmin]$ sudo killall -u TestUser

ImageBuilderAdmin]$ sudo userdel -r TestUser

Creating Default Environment Variables for Your Linux Users

You can create environment variables on a Linux Image Builder instance. Creating environment variables makes them available on streaming instances created from that image.

> **Note**
>
> On Linux fleet instances, environment variables set using the Image Assistant (GUI) tool and the default system environment variables are exported through the /etc/profile.d/appstream_system_vars.sh script. To access these environment variables, you must explicitly source the /etc/profile.d/appstream_system_vars.sh script in your applications.

To create environment variables for your users

1. If the folder `/etc/profile` doesn't exist, run the following command to create it:

   [ImageBuilderAdmin]$ sudo mkdir -p /etc/profile.d

2. To create a new shell script file (for example, my-environment.sh) in this folder, run the following command:

   [ImageBuilderAdmin]$ vim my-environment.sh

3. On first line of the script file, add the following content:

   #!/bin/sh

4. For each subsequent line, add an **export** command to set the environment variables for your image. The following example adds `$HOME/bin` to the `PATH` variable:

   export PATH="$HOME/bin:$PATH"

5. Press the **Esc** key to return to command mode in vim, then run the following command to save your script and exit vim:

   :x

6. Run the following command to allow the script to run as a program:

   [ImageBuilderAdmin]$ chmod +x my-environment.sh

Optimizing the Launch Performance of Your Linux Applications

If you are using the Image Assistant GUI tool, the tool optimizes launch performance for your applications automatically.

If you are using the Image Assistant CLI, use the following steps to optimize launch performance manually. When you create and add files to an application optimization manifest, the application will launch faster when first started on a new fleet instance. However, this also increases the time that it takes for the fleet instances to be made available to users. The optimization manifest is one line-delimited text file for every application.

You can create a manifest file (such as <*your-app*>-manifest.txt) either manually or by following with the steps below.

To create a manifest file

1. Make sure that the application that you are trying to optimize is launched and running.
2. From a terminal in the Linux image builder, run the following command:

   ps -ef | grep <*application-process-name*>
3. Search for the smallest PID number from the last step's output. This is the PID for the root parent process of the application.
4. Keep the application running and make sure to use the initial components required by your users. This ensures that these components are captured by the optimization process.
5. Create a script file (e.g., `~/getfilestool.sh`) with the following content:

   ```
   #!/bin/bash
   ## usage getfilestool.sh $pid
   lsof -p $(pstree -p $1 | grep -o '([0-9]\+)' | grep -o '[0-9]\+' | tr '\012' ,)|
   grep REG | sed -n '1!p' | awk '{print $9}'|awk 'NF'
   ```
6. Make sure that the file can be run with the following command:

   [ImageBuilderAdmin]$ chmod u+x ~/getfilestool.sh
7. Run the following command to capture all of the running files from the root parent process found in step 3, and save it to a temporary manifest file.

   [ImageBuilderAdmin]$ sudo ~/getfilestool.sh <*root-parent-pid*> > /tmp/<y*our-app*>-manifest.txt

8. Verify the content of the optimization manifest, which is a line-delimited text file for every application.

You can specify the optimization manifest on a per-application basis by using the Image Assistant command line interface (CLI) tool. For more information, see the section called "Using the Image Assistant CLI Tool for Linux".

Creating Session Scripts

AppStream 2.0 provides on-instance session scripts on both Windows- and Linux-based streaming instances. For more information about session scripts, see the section called "Use Session Scripts to Manage Your Users' Streaming Experience".

Session scripts are specified within an AppStream 2.0 image. To locate the session scripts configuration file on a Linux instance, navigate to `/opt/appstream/SessionScripts/config.json`. The following code is a sample `config.json` file that specifies a session start script named "`test-session-start`" and a session end script named "`test-session-stop`" together with their runtime parameters. Make sure that the scripts referenced in `config.json` have run permissions and a command interpreter is defined (for example, #!/bin/bash).

```
{
     "SessionStart": {
          "Executables": [
               {
                    "Context": "system",
                    "Filename": "/opt/appstream/SessionScripts/test-session-start",
                    "Arguments": "arg1",
                    "S3LogEnabled": true
               }
          ],
          "WaitingTime": 30
     },
     "SessionTermination": {
          "Executables": [
               {
                    "Context": "system",
                    "Filename": "/opt/appstream/SessionScripts/test-session-stop",
                    "Arguments": "arg2",
                    "S3LogEnabled": true
               }
          ],
```

```
            "WaitingTime": 30
        }
}
```

Using the Image Assistant CLI Tool for Linux

On a Linux-based image builder, you can use the Image Assistant CLI tool **AppStreamImageAssistant** to create and manage your AppStream 2.0 image. The tool is located at `/usr/local/appstream/image-assistant/AppStreamImageAssistant` with a symbolic link at `/bin/AppStreamImageAssistant`. This CLI tool for Linux supports many of the same operations as the Image Assistant CLI tool for Windows. For more information on these operations, see the section called "Image Assistant CLI Operations for Creating and Managing Your AppStream 2.0 Image".

Enabling and Disabling Webcam Support

AppStream 2.0 supports real-time audio-video (AV) by redirecting local webcam video input to AppStream 2.0 streaming sessions. This capability enables your users to use their local webcam for video and audio conferencing within an AppStream 2.0 streaming session. With real-time AV and support for real-time audio, your users can collaborate by using familiar video and audio conferencing applications without having to leave their AppStream 2.0 streaming session.

To use this feature, you must use a Linux AppStream 2.0 image that uses a Linux AppStream 2.0 agent released on or after September 21, 2022.

The real-time AV feature is enabled by default for Linux streaming sessions. To configure webcam permissions for your users on a Linux image builder, create `/etc/appstream/appstream.conf` and add the following contents:

> **Note**
>
> Specify **1** to enable webcam, or **0** to disable webcam.

```
[webcam]
permission = 1
```

Tutorial: Create a Custom Linux-Based AppStream 2.0 Image

This tutorial describes how to create a custom Linux-based Amazon AppStream 2.0 image that contains applications which you can stream to your users.

Contents

- Step 1: Install Linux Applications on the Image Builder
- Step 2: Generate Application Optimization Manifest File
- Step 3: Create an AppStream 2.0 Application Catalog
- Step 4: Create Default Application Settings and Environment Variables
- Step 5: Test Applications and Settings
- Step 6: Finish Creating Your Image
- Step 7 (Optional): Tag and Copy an Image
- Step 8: Clean Up

Step 1: Install Linux Applications on the Image Builder

In this step, you connect a Linux image builder and install your applications on the image builder.

To install applications on the image builder

1. Connect to the image builder by doing either of the following:

   - Use the AppStream 2.0 console (for web connections only)
   - Create a streaming URL (for web or AppStream 2.0 client connections)

   > **Note**
   >
   > You will be logged in as an ImageBuilderAdmin user to the Amazon Linux GNOME desktop and have root admin privileges.

2. Install the applications that you need. For example, to install a Chromium browser from a public yum repo, first open the Terminal application, then run the following command:

   [ImageBuilderAdmin]$ sudo yum update && sudo yum install chromium.x86_64

> **Note**
>
> Download and install applications only from sites that you trust.

Step 2: Generate Application Optimization Manifest File

In this step, you generate a manifest file for each application you installed in step 1.

To generate a manifest file for optimizing the launch performance of an application

1. Make sure the application (e.g., Chromium) that you are trying to optimize is launched and running.
2. In a Terminal window, run the following command to list processes related to the application:

   [ImageBuilderAdmin]$ ps -ef | grep chromium

3. Find the root parent PID from the output of the command above. The following is an example output, and the root parent PID is 16712:

   Example

   ```
   [ImageBuilderAdmin]$ ps -ef | grep chromium

   ImageBu+ 16712 4128 0 Aug26 ? 00:00:44 /usr/lib64/chromium- browser/chromium-
   browser --enable-plugins --enable-extensions -- enable-user- scripts --enable-
   printing --enable-gpu-rasterization -- enable-sync --auto-ssl- client-auth

   ImageBu+ 16726 16712 0 Aug26 ? 00:00:00 /usr/lib64/chromium- browser/chromium-
   browser --type=zygote --no-zygote-sandbox ImageBu+ 16727 16712 0 Aug26 ? 00:00:00 /
   usr/lib64/chromium- browser/chromium- browser --type=zygote

   ImageBu+ 16731 16727 0 Aug26 ? 00:00:00 /usr/lib64/chromium- browser/chromium-
   browser --type=zygot
   ```

4. Keep the application running and make sure to use the initial components required by your users. This ensures that these components are captured by the optimization process.
5. Create script file (e.g., ~/`getfilestool.sh`) with the following content:

   ```
   #!/bin/bash
   ```

```
## usage getfilestool.sh $pid
lsof -p $(pstree -p $1 | grep -o '([0-9]\+)' | grep -o '[0-9]\+' | tr '\012' ,)|
grep REG | sed -n '1!p' | awk '{print $9}'|awk 'NF'
```

6. Verify that the file can be run by running the following command:

   [ImageBuilderAdmin]$ chmod u+x ~/getfilestool.sh

7. Run the following command to capture all running files from the root parent process found in step 3 above, and save it to a temporary manifest file:

   [ImageBuilderAdmin]$ sudo ~/getfilestool.sh 16712 > /tmp/chromium-manifest.txt

8. Verify the content of the optimization manifest, which is a line-delimited text file for each application.

Step 3: Create an AppStream 2.0 Application Catalog

In this step, you use the CLI tool `AppStreamImageAssistant` on the image builder to create an AppStream 2.0 application catalog by specifying applications for your image. For each application that you plan to stream, you can specify the name, display name, executable file to launch, and icon to display.

To create an AppStream 2.0 application catalog

1. From the image builder desktop, open **Terminal** either from the side panel or by opening the app grid.
2. Run **AppStreamImageAssistant --help** to see the list of available commands. You will use these commands to add applications and create an Image.
3. Run the following command to add an installed application (e.g., Chromium) to the application list for AppStream 2.0 users:

   ```
   AppStreamImageAssistant add-application \
    --name Chromium \
    --absolute-app-path /usr/lib64/chromium-browser/chromium-browser \
    --display-name Chromium \
    --absolute-icon-path /usr/share/icons/hicolor/256x256/apps/chromium-browser.png \
    --absolute-manifest-path /tmp/chromium-manifest.txt
   ```

   Alternatively, run the following command:

```
AppStreamImageAssistant add-application \
 --name="Chromium" \
 --absolute-app-path="/usr/lib64/chromium-browser/chromium-browser" \
 --display-name="Chromium" \
 --absolute-icon-path="/usr/share/icons/hicolor/256x256/apps/chromium-browser.png"
 \
 --absolute-manifest-path="/tmp/chromium-manifest.txt"
```

4. To add more applications, repeat step 3 for each additional application.
5. To see the list of applications that have been added in the catalog, along with metadata like icon paths and launch parameters, run the following command:

   AppStreamImageAssistant list-applications

6. To remove applications from the catalog, run the following command:

   AppStreamImageAssistant remove-application –-name *application_name*

Step 4: Create Default Application Settings and Environment Variables

In this step, you create default application settings and environment variables for your AppStream 2.0 users. Doing this enables your users to get started with applications quickly during their AppStream 2.0 streaming sessions, without the need to create or configure these settings themselves.

To create default application and environment variables for your users

1. Launch the application that you want create the default settings for. For example, in a Terminal window, run following command to launch Chromium browser:

   [ImageBuilderAdmin]$ chromium-browser

2. Configure the settings of the application. For example, set the home page of the Chromium browser as **`https://aws.amazon.com`**.
3. Run the following commands to copy the configuration for Chromium to **/etc/skel**:

   [ImageBuilderAdmin]$ sudo mkdir /etc/skel/.config

   [ImageBuilderAdmin]$ sudo cp -R ~/.config/chromium /etc/skel/.config

4. Set the environment variables and add it to the script file. For example, run the following commands:

   [ImageBuilderAdmin]$ echo "export *FOO*=*BAR*" | sudo tee -a /etc/profile.d/myenvvars.sh

   [ImageBuilderAdmin]$ sudo chmod +x /etc/profile.d/myenvvars.sh

Step 5: Test Applications and Settings

In this step, verify that the applications you've added run correctly, and the default application settings and environment variables work as expected.

To test your applications and default settings on an image builder

1. Create a test user that has no root permissions. For example, in a **Terminal** window, run the following commands to create **test-user** on the image builder:

   [ImageBuilderAdmin]$ sudo useradd -m test-user

   [ImageBuilderAdmin]$ echo -e 'Pa55w0rdas2!!!\nPa55w0rdas2!!!\n' | sudo passwd test-user

2. Switch to the test user:

   [ImageBuilderAdmin]$ su - test-user

3. Launch the application (e.g., Chromium) as the test user:

   [test-user]$ /usr/bin/chromium-browser

4. Verify that the default settings are available for the test user (e.g., the Chromium home page is https://aws.amazon.com/).
5. Verify that the environment variables are available for the test user. For example, run the following command:

   [test-user]$ echo $*FOO*

   This command should display the output ***BAR*** in the terminal.

6. Run the following commands to delete the test user before creating an image from this image builder:

   # logout test user

[test-user]$ logout

 # kill test user's running processes

[ImageBuilderAdmin]$ sudo killall -u test-user

 # delete user

[ImageBuilderAdmin]$ sudo userdel -r test-user

Step 6: Finish Creating Your Image

In this step, choose an image name and finish creating your image.

To create the image

1. In a **Terminal** window, create an image from your Image Builder by running **AppStreamImageAssistant create-image**. This image contains your installed and registered applications, plus any session scripts and default application settings that you have configured.

   To see the list of available options, run **AppStreamImageAssistant create-image --help**. For more information, see the **create-image** operation in the section called "Create Your AppStream 2.0 Image Programmatically".
2. The remote session disconnects after a few moments. When the **Lost Connectivity** message appears, close the browser tab. While the image is created, the image builder status appears as **Snapshotting**. You cannot connect to the image builder until this process finishes.
3. Return to the console and navigate to **Images**, **Image Registry**. Verify that your new image appears in the list.

   While your image is being created, the image status in the image registry of the console appears as **Pending**. You can't connect to images that are in a **Pending** status.
4. Choose the **Refresh** icon to update the status. After your image is created, the image status changes to **Available** and the image builder is automatically stopped.

   To continue creating images, start the image builder and connect to it from the console, or create a new image builder.

Step 7 (Optional): Tag and Copy an Image

You can add one or more tags to an image during image creation or after you create an image. You can also copy the image within the same Region or to a new Region within the same Amazon Web Services account. Copying a source image results in an identical but distinct destination image. AWS does not copy any user-defined tags, however. Also, you can only copy custom images that you create, not the base images that are provided by AWS.

> **Note**
>
> You can copy up to two images at the same time to a destination. If the destination to which you are copying an image is at the image limit, you receive an error. To copy the image in this case, you must first remove images from the destination. After the destination is below the image quota (also referred to as limit), initiate the image copy from the source Region. For more information, see Amazon AppStream 2.0 Service Quotas.

To add tags to an existing image

1. In the navigation pane, choose **Images**, **Image Registry**.
2. In the image list, select the image to which you want to add tags.
3. Choose **Tags**, choose **Add/Edit Tags**, and then choose **Add Tag**. Specify the key and value for the tag, and then choose **Save**.

For more information, see Tagging Your Amazon AppStream 2.0 Resources.

To copy an image

Copying an image across geographically diverse regions enables you to stream applications from multiple regions based on the same image. By streaming your applications in closer proximity to your users, you can improve your users' experience streaming applications with AppStream 2.0.

1. In the navigation pane, choose **Images**, **Image Registry**.
2. In the image list, select the image that you want to copy.
3. Choose **Actions**, **Copy**.
4. In the **Copy Image** dialog box, specify the following information, and then choose **Copy Image**:

- For **Destination region**, choose the region to which to copy the new image.
- For **Name**, specify a name that the image will have when it is copied to the destination.
- For **Description** (optional), specify a description that the image will have when it is copied to the destination.

5. To check on the progress of the copy operation, return to the console and navigate to **Images**, **Image Registry**. Use the navigation bar to switch to the destination region (if applicable), and confirm that your new image appears in the list of images.

   The new image first appears with a status of **Copying** in the image registry of your console. After the image is successfully created, the status of the image changes to **Available**, which means that you can use the image to launch a stack and stream your applications.

Step 8: Clean Up

Finally, you can 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**.

Tutorial: Enable Japanese Support for Your Linux Images

This tutorial describes how to enable Japanese support for a Linux image. This enables applications on the image to display Japanese characters, and streaming users to use the Japanese input method in the streaming sessions from the image.

Contents

- Step 1: Install Japanese Font and Input Method
- Step 2: Set the System Time Zone
- Step 3: Set the System Locale and Display Language
- Step 4: Configure the Input Methods

- Step 5: Set the Keyboard Layout
- Step 6: Verify on Image Builder
- Step 7: Create the Image

Step 1: Install Japanese Font and Input Method

In this step, you connect a Linux image builder and install the font and input method packages of your choice.

To install Japanese font and input method

1. Connect to the image builder by doing either of the following:

   - Use the AppStream 2.0 console (for web connections only)
   - Create a streaming URL (for web or AppStream 2.0 client connections)

   > **Note**
   >
   > You will be logged in as an ImageBuilderAdmin user to the Amazon Linux GNOME desktop and have root admin privileges.

2. Install the font and input method that you need. To do this, open the Terminal application, then run the following commands:

   sudo yum install vlgothic-p-fonts.noarch

   sudo yum install ibus-kkc.x86_64

Step 2: Set the System Time Zone

To set the system time zone, run the following command:

sudo timedatectl set-timezone "Asia/Tokyo"

Step 3: Set the System Locale and Display Language

To set the system locale and display language, run the following commands.

To set the system locale and display language

1. Update the `cloud-init config` file by running the command **sudo vim /etc/cloud/cloud.cfg**, and change **locale** to **locale: ja_JP.utf8**, then save and close the file.
2. Update the system settings by running **sudo localectl set-locale LANG=ja_JP.utf8**.
3. Update the Gnome shell settings by running **sudo gsettings set org.gnome.system.locale region "ja_JP.utf8"**.

Step 4: Configure the Input Methods

Configure the input methods for the application you want to add to the image. For more information about how install an application, generate a manifest file, and create default settings, see Tutorial: Create a Custom Linux-Based AppStream 2.0 Image. In this step, we assume that you've already installed the application Firefox, which is located at `/usr/local/firefox/firefox`.

To configure the input methods

1. Create a script by running the command **sudo vim /usr/local/bin/update-input-method.sh**, and add the following content to the script:

```
#!/bin/bash

function start_process()
{
    command=$1
    process_name=$2

    process_count=$(pgrep $process_name -c)
    echo "$(date) current $process_name count: $process_count"
    while [ $process_count -lt 1 ]
    do
        echo "$(date) starting $process_name"
        eval $command
        sleep 1
        process_count=$(pgrep $process_name -c)
    done
    echo "$(date) $process_name started"
}
```

```
start_process "ibus-daemon --xim &" "ibus-daemon"
start_process "/usr/libexec/ibus-engine-kkc --ibus &" "ibus-engine-kkc"

gsettings set org.gnome.desktop.input-sources sources "[('ibus','kkc'), ('xkb',
 'us')]"
gsettings set org.gnome.desktop.wm.keybindings switch-input-source
 "['<Control>space']"
gsettings set org.gnome.desktop.wm.keybindings switch-input-source-backward
 "['<Shift><Control>space']"

echo "$(date) updated input source and switch shortcut"
```

In the script above, the first input source (‘ibus’, ‘kkc’) is the default input method. You can change the default input method by changing the order of input sources. In addition, “Control +Space” and “Shift+Control+Space” are specified as shortcut key combinations for switching between input methods. You can specify your own key combinations that your users can use to switch input methods during streaming sessions.

2. Create the script for launching the application (Firefox) that you will add to the image. To do this, run the command s**udo vim /usr/local/bin/firefox-jp.sh**, then add following content to the script:

```
#!/bin/bash

/usr/local/bin/update-input-method.sh > /var/tmp/update-input-method.log 2>&1 &

/usr/local/firefox/firefox &
```

3. Add run permission to both scripts by running following the commands:

   sudo chmod +x /usr/local/bin/update-input-method.sh

   sudo chmod +x /usr/local/bin/firefox-jp.sh

4. If you already created the optimization manifest file for the application run the following commands to add the application launch script to the application catalog:

```
sudo AppStreamImageAssistant add-application \
--name firefox \
--absolute-app-path /usr/local/bin/firefox-jp.sh \
```

```
--display-name firefox \
--absolute-icon-path /usr/local/firefox/browser/chrome/icons/default/default128.png
 \
--absolute-manifest-path /tmp/firefox-manifest.txt
```

Alternatively, you can also configure the input methods by adding the script update-input-method.sh as a separate application to the application catalog for the image. During streaming sessions, your users can launch this application to enable Japanese input, and switch between input methods with specified shortcut keys within the same session.

Step 5: Set the Keyboard Layout

Set the keyboard layout to match the keyboards your users will use during streaming sessions. You can use the command **localectl list-keymaps** to list all the available keymaps, and use the command **sudo localectl set-keymap jp106** to set the keymap to the Japanese keyboard of 106 keys, for example.

Step 6: Verify on Image Builder

To verify on image builder, first reboot the image builder by running the command **sudo shutdown -r now**. After reboot, connect to the image builder again, and verify that everything, including time zone, locale, language, and input method, works as expected.

Step 7: Create the Image

Create the image on the image builder. For more information, see Tutorial: Create a Custom Linux-Based AppStream 2.0 Image. Make sure to create default application settings, including the regional settings you just configured. For more information, see "Creating Default Application Settings for Your Users" in Create Your Linux-Based Images.

All of the Linux fleet instances created from this image will have the same default time zone, locale, language, and input method settings that you configured for the image.

Use Session Scripts to Manage Your AppStream 2.0 Users' Streaming Experience

AppStream 2.0 provides on-instance session scripts. You can use these scripts to run your own custom scripts when specific events occur in users' streaming sessions. For example, you can use

custom scripts to prepare your AppStream 2.0 environment before your users' streaming sessions begin. You can also use custom scripts to clean up streaming instances after users complete their streaming sessions.

Session scripts are specified within an AppStream 2.0 image. These scripts are run within the user context or the system context. If your session scripts use the standard out to write information, error, or debugging messaging, these can be optionally saved to an Amazon S3 bucket within your Amazon Web Services account.

Contents

- Run Scripts Before Streaming Sessions Begin
- Run Scripts After Streaming Sessions End
- Create and Specify Session Scripts
- Session Scripts Configuration File
- Using Windows PowerShell Files
- Logging Session Script Output
- Use Storage Connectors with Session Scripts
- Enable Amazon S3 Bucket Storage for Session Script Logs
- Use Session Scripts on Multi-Session Fleets

Run Scripts Before Streaming Sessions Begin

You can configure your scripts to run for a maximum of 60 seconds before your users' applications launch and their streaming sessions begin. Doing so enables you to customize the AppStream 2.0 environment before users start streaming their applications. When the session scripts run, a loading spinner displays for your users. When your scripts complete successfully or the maximum waiting time elapses, your users' streaming session will begin. If your scripts don't complete successfully, an error message displays for your users. However, your users are not prevented from using their streaming session.

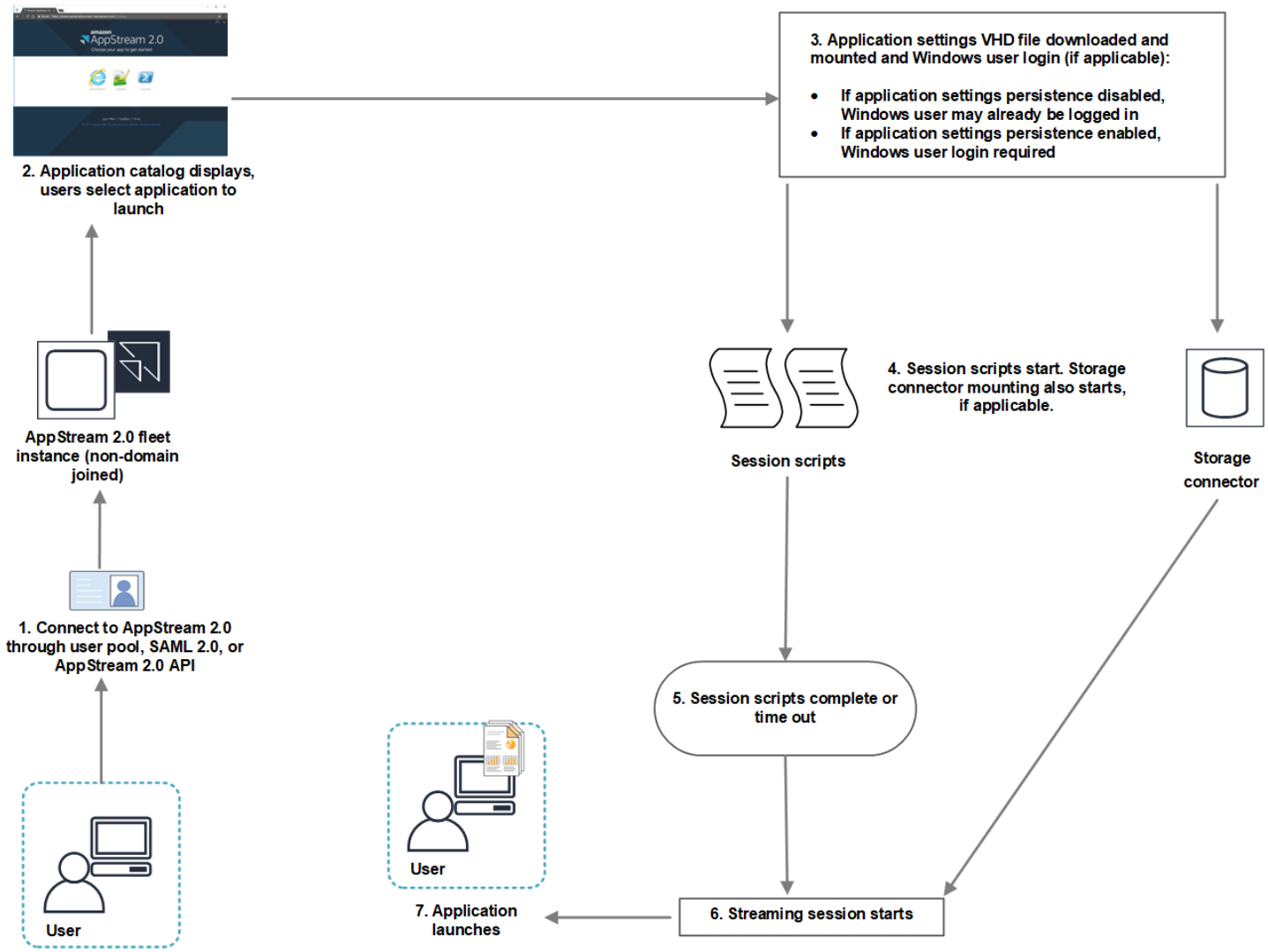
When you specify a file name on a Windows instance, you must use a double backslash. For example:

C:\\Scripts\\Myscript.bat

If you don't use a double backslash, an error displays to notify you that the .json file is incorrectly formatted.

> **Note**
>
> When your scripts complete successfully, they must return a value of 0. If your scripts return a value other than 0, AppStream 2.0 displays the error message to the user.

When you run scripts before streaming sessions begin and the AppStream 2.0 dynamic application framework is not enabled, the following process occurs:



1. Your users connect to an AppStream 2.0 fleet instance that is not domain-joined. They connect by using one of the following access methods:
   - AppStream 2.0 user pool
   - SAML 2.0

- AppStream 2.0 API

2. The application catalog displays in the AppStream 2.0 portal, and your users choose an application to launch.
3. One of the following occurs:
   - If application settings persistence is enabled for your users, the application settings Virtual Hard Disk (VHD) file that stores your users' customizations and Windows settings is downloaded and mounted. Windows user login is required in this case.

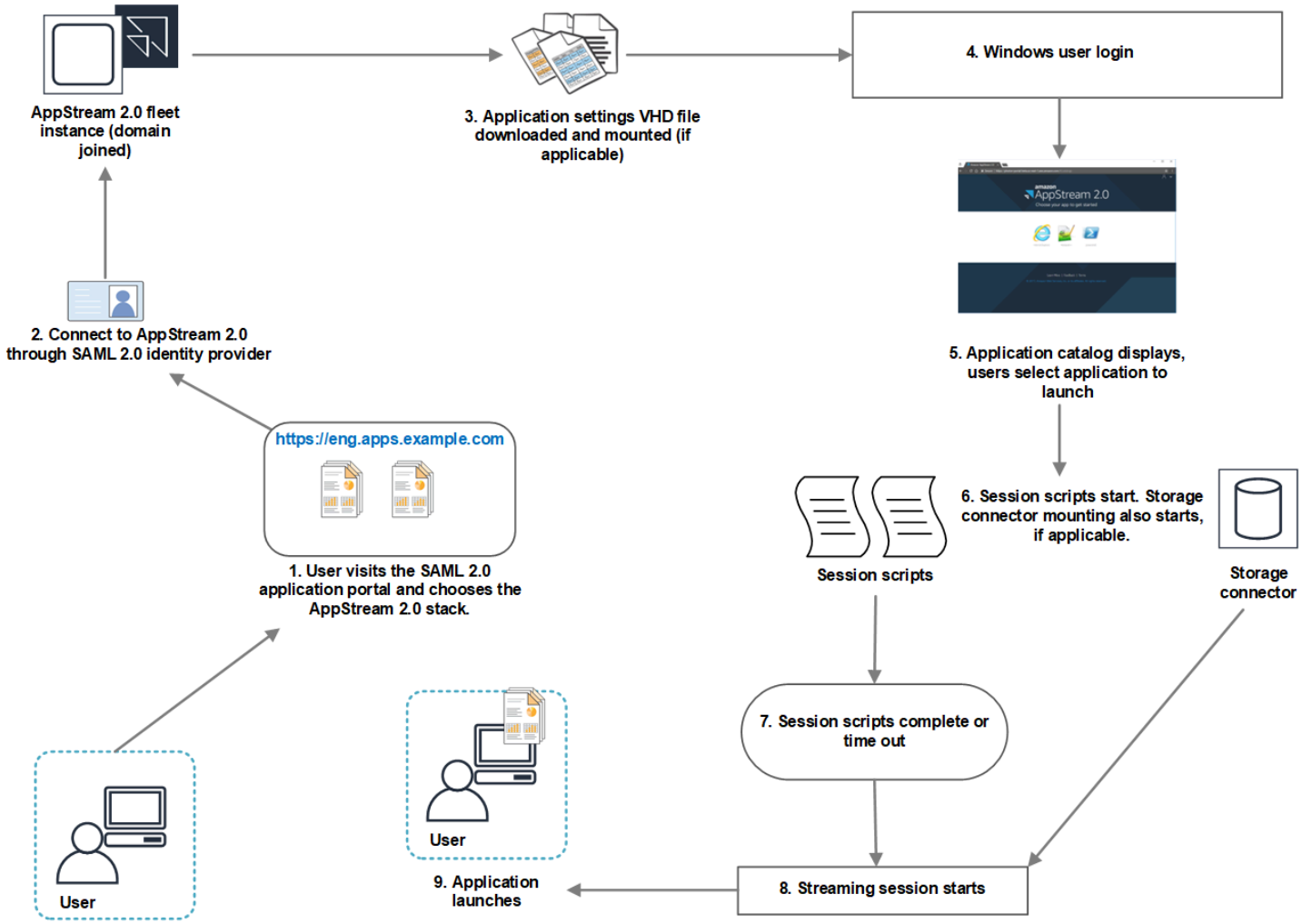
     For information about application settings persistence, see Enable Application Settings Persistence for Your AppStream 2.0 Users.
   - If application settings persistence is not enabled, the Windows user is already logged in.
4. Your session scripts start. If persistent storage is enabled for your users, storage connector mounting also starts. For information about persistent storage, see Enable and Administer Persistent Storage for Your AppStream 2.0 Users.

   > **Note**
   >
   > The storage connector mount doesn't need to complete for the streaming session to start. If the session scripts complete before the storage connector mount completes, the streaming session starts.
   > For information about monitoring the mount status of storage connectors, see Use Storage Connectors with Session Scripts.

5. Your session scripts complete or time out.
6. The users' streaming session starts.
7. The application that your users chose launches.

For information about the AppStream 2.0 dynamic application framework, see Use the AppStream 2.0 Dynamic Application Framework to Build a Dynamic App Provider.

When you run scripts before streaming sessions begin and the AppStream 2.0 dynamic application framework is enabled, the following process occurs:



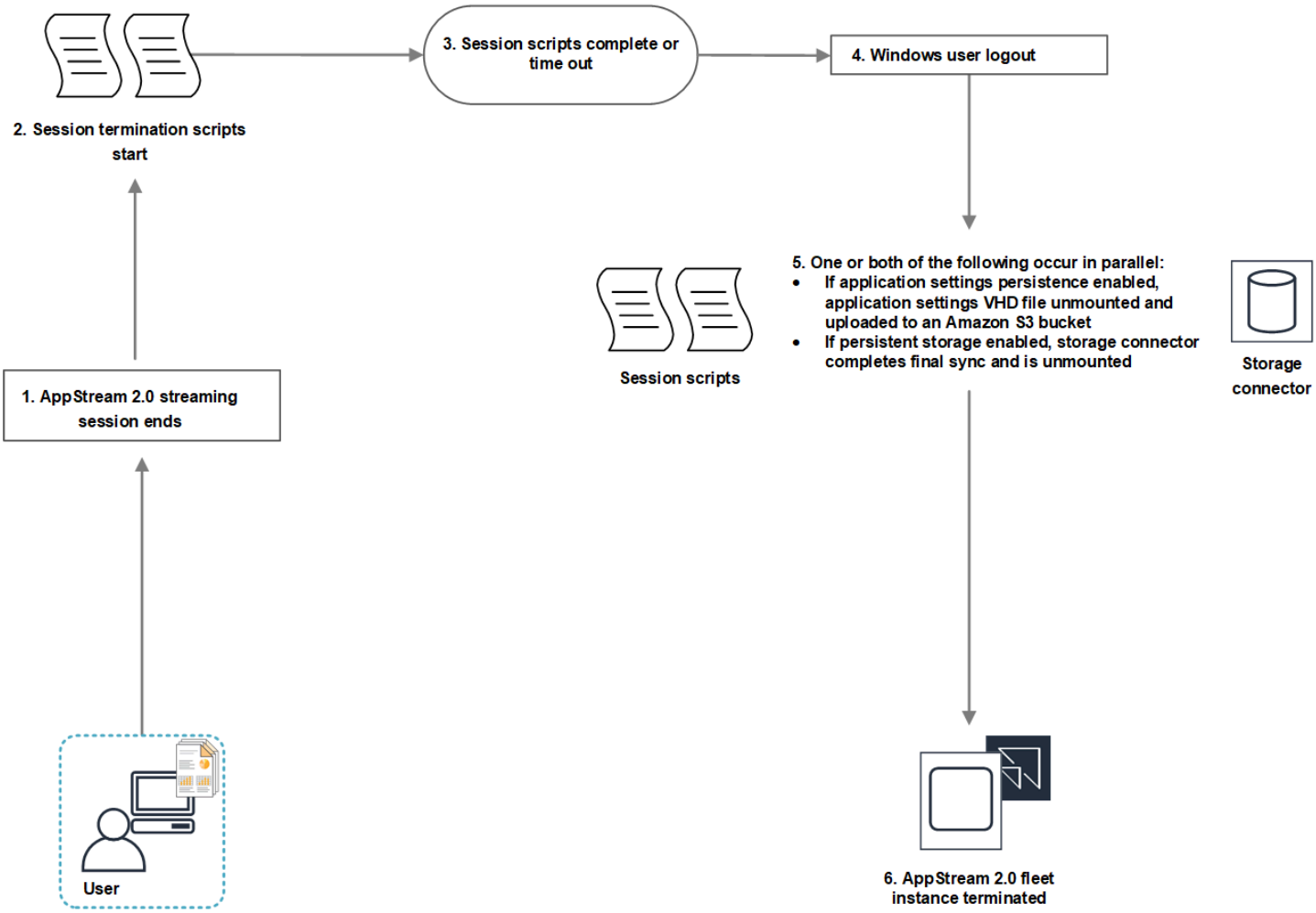
1. Your users visit the SAML 2.0 application portal for your organization, and they choose the AppStream 2.0 stack.
2. They connect to an AppStream 2.0 fleet instance that is domain-joined.
3. If application settings persistence is enabled for your users, the application settings VHD file that stores your users' customizations and Windows settings is downloaded and mounted.
4. Windows user logon occurs.
5. The application catalog displays in the AppStream 2.0 portal and your users choose an application to launch.
6. Your session scripts start. If persistent storage is enabled for your users, storage connector mounting also starts.

> **Note**
>
> The storage connector mount doesn't need to complete for the streaming session to start. If the session scripts complete before the storage connector mount completes, the streaming session starts.
> For information about monitoring the mount status of storage connectors, see Use Storage Connectors with Session Scripts.

7. Your session scripts complete or time out.
8. The users' streaming session starts.
9. The application that your users chose launches.

Run Scripts After Streaming Sessions End

You can also configure your scripts to run after users' streaming sessions end. For example, you can run a script when users select **End Session** from the AppStream 2.0 toolbar, or when they reach the maximum allowed duration for the session. You can also use these session scripts to clean up your AppStream 2.0 environment before a streaming instance is terminated. For example, you can use scripts to release file locks or upload log files. When you run scripts after streaming sessions end, the following process occurs:



1. Your users' AppStream 2.0 streaming session ends.
2. Your session termination scripts start.
3. The session termination scripts complete or time out.
4. Windows user logout occurs.
5. One or both of the following occur in parallel, if applicable:
   - If application settings persistence is enabled for your users, the application settings VHD file that stores your users' customizations and Windows settings is unmounted and uploaded to an Amazon S3 bucket in your account.
   - If persistent storage is enabled for your users, the storage connector completes a final synchronization and is unmounted.
6. The fleet instance is terminated.

Create and Specify Session Scripts

You can configure and specify session scripts for Always-on, On-demand, and Elastic fleets.

To configure and specify session scripts for Always-on and On-demand fleets

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. In the navigation pane, choose **Images**, **Image Builder**.
3. Choose an image builder that is in the **Running** state, and choose **Connect**.
4. When prompted, choose **Administrator**.
5. Navigate to `C:\AppStream\SessionScripts`, and open the `config.json` configuration file.

   For information about session script parameters, see Session Scripts Configuration File.
6. After you finish making your changes, save and close the `config.json` file.
7. On the image builder desktop, open **Image Assistant**.
8. (Optional) Specify any additional applications that you want to include in the image.
9. Follow the necessary steps in Image Assistant to finish creating your image.

   If the session scripts configuration can't be validated (for example, if the .json file is not correctly formatted), you are notified when you choose **Disconnect and create image**.

   > **Note**
   >
   > To locate the session scripts configuration file for Linux-based image builders, navigate to `/opt/appstream/SessionScripts/config.json`.

To configure and specify session scripts for Elastic fleets

1. Create a zip file that contains the session scripts and config.json file. The scripts files will be copied to the following locations. You must use these locations for your config.json.

   - For Windows, use `C:\AppStream\SessionScripts\`*`SessionScript`*.
   - For Linux, use `/opt/appstream/SessionScripts/`*`SessionScript`*.

> **Note**
>
> In order to run the session script files, make sure that the .zip file only contains the session scripts and `config.json` files, and not the containing folder. For more information, see Session Scripts Configuration File.

2. Upload the zip file to an Amazon S3 bucket in your account.

> **Note**
>
> Your VPC must provide access to the Amazon S3 bucket. For more information, see Using Amazon S3 VPC Endpoints for AppStream 2.0 Features.
> You must have your S3 bucket and AppStream 2.0 fleet in the same AWS Region.
> You must have IAM permissions to perform the `S3:GetObject` action on the session scripts object in the Amazon S3 bucket. To learn more about storing the session scripts in an Amazon S3 bucket, see Store Application Icon, Setup Script, Session Script, and VHD in an S3 Bucket.

3. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
4. In the navigation pane, choose **Fleets**.
5. Choose an Elastic fleet that you want to update, and then choose **View Details**.
6. On the **Session scripts settings** tab, choose **Edit**.
7. For **Session scripts object in S3**, either enter the S3 URI that represents the session scripts object, or choose **Browse S3** to navigate to your S3 buckets and find the session scripts object.
8. After you finish making your changes, choose **Save Changes**.
9. At this point, session scripts are available for all fleet instances launched.

> **Note**
>
> You can also configure the session scripts when you create a new Elastic fleet.

Session Scripts Configuration File

To locate the session scripts configuration file in a Windows instance, navigate to C:\AppStream\SessionScripts\config.json. On a Linux instance, navigate to /opt/appstream/SessionScripts/config.json. The file is formatted as follows.

> **Note**
>
> The configuration file is in .json format. Verify that any text you type in this file is in valid .json format.

```
{
  "SessionStart": {
    "executables": [
      {
        "context": "system",
        "filename": "",
        "arguments": "",
        "s3LogEnabled": true
      },
      {
        "context": "user",
        "filename": "",
        "arguments": "",
        "s3LogEnabled": true
      }
    ],
    "waitingTime": 30
  },
  "SessionTermination": {
    "executables": [
      {
        "context": "system",
        "filename": "",
        "arguments": "",
        "s3LogEnabled": true
      },
      {
        "context": "user",
        "filename": "",
        "arguments": "",
```

```
        "s3LogEnabled": true
      }
    ],
    "waitingTime": 30
  }
}
```

You can use the following parameters in the session scripts configuration file.

SessionStart/SessionTermination

The session scripts to run in the appropriate session event based on the name of the object.

Type: String

Required: No

Allowed values: `SessionStart`, **`SessionTermination`**

WaitingTime

The maximum duration of the session scripts in seconds.

Type: Integer

Required: No

Constraints: The maximum duration is 60 seconds. If the session scripts don't complete within this duration, they will be stopped. If you require a script to continue running, launch it as a separate process.

Executables

The details for the session scripts to run.

Type: String

Required: Yes

Constraints: The maximum number of scripts that can run per session event is 2 (one for the user context, one for the system context).

Context

The context in which to run the session script.

Type: String

Required: Yes

Allowed values: `user`, `system`

`Filename`

The full path to the session script to run. If this parameter is not specified, the session script is not run.

Type: String

Required: No

Constraints: The maximum length for the file name and full path is 1,000 characters.

Allowed values: `.bat`, `.exe`, `.sh`

> **Note**
>
> You can also use Windows PowerShell files. For more information, see Using Windows PowerShell Files.

`Arguments`

The arguments for your session script or executable file.

Type: String

Required: No

Length constraints: The maximum length is 1,000 characters.

`S3LogEnabled`

When the value for this parameter is set to **`True`**, an S3 bucket is created within your Amazon Web Services account to store the logs created by the session script. By default, this value is set to **`True`**. For more information, see the *Logging Session Script Output* section later in this topic.

Type: Boolean

Required: No

Allowed values: `True`, `False`

Using Windows PowerShell Files

To use Windows PowerShell files, specify the full path to the PowerShell file in the **`filename`** parameter:

```
"filename":
"C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe",
```

Then specify your session script in the **`arguments`** parameter:

```
"arguments": "-File \"C:\\path\\to\\session\\script.ps1\"",
```

Finally, verify that the PowerShell Execution Policy allows your PowerShell file to run.

Logging Session Script Output

When this option is enabled in the configuration file, AppStream 2.0 automatically captures the output from the session script that is written to the standard out. This output is uploaded to an Amazon S3 bucket in your account. You can review the log files for troubleshooting or debugging purposes.

> **Note**
>
> The log files are uploaded when the session script returns a value, or the value set in **`WaitingTime`** has elapsed, whichever comes first.

Use Storage Connectors with Session Scripts

When AppStream 2.0 storage connectors are enabled, they begin mounting when the session start scripts run. If your script relies on the storage connectors being mounted, you can wait for the connectors to be available. AppStream 2.0 maintains the mount status of the storage connectors in the Windows registry on Windows instances, at the following key:

HKEY_LOCAL_MACHINE\SOFTWARE\Amazon\AppStream\Storage\<provided user name>\<Storage connector>

The registry key values are as follows:

- Provided user name — The user ID provided through the access mode. The access modes and value for each mode are as follows:
  - User Pool — The email address for the user
  - Streaming URL — The UserID
  - SAML — The NameID. If the user name includes a slash (for example, a domain user's SAMAccountName), the slash is replaced by a "-" character.
- Storage connector — The connector for the persistent storage option that is enabled for the user. The storage connector values are as follows:
  - HomeFolder
  - GoogleDrive
  - OneDrive

Each storage connector registry key contains a **MountStatus** DWORD value. The following table lists the possible values for **MountStatus**.

> **Note**
>
> To view these registry keys, you must have Microsoft .NET Framework version 4.7.2 or later installed on your image.

Value	Description
0	Storage connector not be enabled for this user
1	Storage connector mounting is in progress
2	Storage connector mounted successfully
3	Storage connector mounting failed
4	Storage connector mounting is enabled, but not mounted yet

On Linux instances, you can check the home folder mount status by looking at the value of appstream_home_folder_mount_status in the file ~/.config/appstream-home-folder/appstream-home-folder-mount-status.

Value	Description
True	Home folder is mounted successfully
False	Home folder is not mounted yet

Enable Amazon S3 Bucket Storage for Session Script Logs

When you enable Amazon S3 logging in your session script configuration, AppStream 2.0 captures standard output from your session script. The output is periodically uploaded to an S3 bucket within your Amazon Web Services account. For every AWS Region, AppStream 2.0 creates a bucket in your account that is unique to your account and the Region.

You do not need to perform any configuration tasks to manage these S3 buckets. They are fully managed by the AppStream 2.0 service. The log files that are stored in each bucket are encrypted in transit using Amazon S3's SSL endpoints and at rest using Amazon S3-managed encryption keys. The buckets are named in a specific format as follows:

```
appstream-logs-region-code-account-id-without-hyphens-random-identifier
```

region-code

This is the AWS Region code in which the stack is created with Amazon S3 bucket storage enabled for session script logs.

account-id-without-hyphens

Your Amazon Web Services account identifier. The random ID ensures that there is no conflict with other buckets in that Region. The first part of the bucket name, `appstream-logs`, does not change across accounts or Regions.

For example, if you specify session scripts in an image in the US West (Oregon) Region (us-west-2) on account number 123456789012, AppStream 2.0 creates an Amazon S3 bucket within your

account in that Region with the name shown. Only an administrator with sufficient permissions can delete this bucket.

```
appstream-logs-us-west-2-1234567890123-abcdefg
```

Disabling session scripts does not delete any log files stored in the S3 bucket. To permanently delete log files, you or another administrator with adequate permissions must do so by using the Amazon S3 console or API. 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 Application Settings Persistence* in Identity and Access Management for Amazon AppStream 2.0.

When session scripts are enabled, a unique folder is created for each streaming session that is started.

The path for the folder where the log files are stored in the S3 bucket in your account uses the following structure:

```
bucket-name/stack-name/fleet-name/access-mode/user-id-SHA-256-hash/session-id/
SessionScriptsLogs/session-event
```

bucket-name

The name of the S3 bucket in which the session scripts are stored. The name format is described earlier in this section.

stack-name

The name of the stack the session came from.

fleet-name

The name of the fleet the session script is running on.

access-mode

The identity method of the user: `custom` for the AppStream 2.0 API or CLI, `federated` for SAML, and `userpool` for users in the user pool.

user-id-SHA-256-hash

The user-specific folder name. This name is created using a lowercase SHA-256 hash hexadecimal string generated from the user identifier.

session-id

The identifier of the user's streaming session. Each user streaming session generates a unique ID.

session-event

The event that generated the session script log. The event values are: `SessionStart` and `SessionTermination`.

The following example folder structure applies to a streaming session started from the test-stack and test-fleet. The session uses the API of user ID `testuser@mydomain.com`, from an AWS account ID of `123456789012`, and the settings group `test-stack` in the US West (Oregon) Region (us-west-2):

```
appstream-logs-us-west-2-1234567890123-abcdefg/test-stack/test-fleet/custom/
a0bcb1da11f480d9b5b3e90f91243143eac04cfccfbdc777e740fab628a1cd13/05yd1391-4805-3da6-
f498-76f5x6746016/SessionScriptsLogs/SessionStart/
```

This example folder structure contains one log file for a user context session start script, and one log file for a system context session start script, if applicable.

Use Session Scripts on Multi-Session Fleets

When using session scripts on multi-session fleets, there are additional requirements and considerations to ensure optimal performance and security.

Requirements

On a single-session fleet, for a given instance, the **SessionStart** and **SessionTermination** hooks are guaranteed to run only one time. This is because there is a 1:1 mapping of sessions to instances. When using multi-session fleets, there is an N:M mapping of sessions to instances, where each session runs its own **SessionStart** and **SessionTermination** hook. This means that the **SessionStart** and **SessionTermination** hooks can be run many times on a given instance, and in many different orderings. For the best experience, the following should be true of your session scripts when used on multi-session fleets:

- Scripts are idempotent.

When an action has already been performed, scripts should handle more than one execution on the same instance with graceful handling.

- Scripts are independent.

  Because scripts run per session, if one session is running **SessionTermination** while another is running **SessionStart**, they should not interfere with each other, or with the experience of other sessions.

- Scripts are performant.

  On multi-session instances, multiple sessions can be provisioned concurrently. This means that there can be multiple concurrent executions of the session scripts. Scripts should be efficient, not consume excessive resources, and not impact the experience of other users on the instance or the stability of the sessions.

Many of these requirements can be met by keeping session script logic focused on the specific user session for which the script is running.

Security Considerations

AppStream 2.0 images should not be configured to allow write permission to session script files by any users. This introduces a critical attack vector for malicious users, where they could modify script files. These files could then be run as SYSTEM or another user, depending on your configuration.

> **Important**
>
> It is your responsibility to make sure that your AppStream 2.0 images are configured securely. This is especially important for multi-session instances, where multiple users are using the same instance. If images are not configured securely, there is a security risk for all users of that instance.

The following should be true of your images and session scripts files:

- Users do not have permission to modify session script files.
- Users do not have permission to modify the session script config.json. Default behavior on the image restricts access to administrators.

Session scripts executables should be stored in a secure location where they are safe from modification at runtime.

If the service detects that a session script executable has been modified, it will fail any subsequent executions of that hook on that instance, upload log files to Amazon S3 (if Amazon S3 logging is enabled), and you will see the following message:

The session script was not executed because the executable was modified after instance provisioning. Execution was skipped for security.

If your use case requires modifying the session script executable at run time (for example, if you point to an EXE file which is modified by an automatic update process at runtime), this will fail the above checks. In this case, use a script to redirect execution to your modified executable. Leave the script unmodified at runtime when the service performs security checks.

If your session script files are excessively large (more than 100 MB), this can cause delays in instance and session provisioning, and the security checks will take additional time (depending on instance type and available resources). If your use case requires large session scripts, consider using smaller scripts to redirect execution. This will improve instance and session provisioning experiences.

Note that the service is only checking the executable defined in the session scripts config.json, and this is only a fallback/best effort mechanism. It is your responsibility to ensure that all code paths in session scripts executables are secure and cannot be modified by end users.

Applications Manager

When using an Elastic fleet, you can create app blocks and applications. *App blocks* represent a virtual hard disk (VHD) that is stored within an Amazon S3 bucket within your account that contains the application files and binaries necessary to launch the applications that your users will use. *Applications* contain the details necessary to launch your application after the VHD has been mounted. The following sections describe how to create and manage these resources.

Contents

- App Blocks
- App Block Builder
- Applications
- Store Application Icon, Setup Script, Session Script, and VHD in an S3 Bucket
- Associate Applications to Elastic Fleets
- Additional Resources

App Blocks

App blocks represent a virtual hard disk (VHD) that is stored within an Amazon S3 bucket within your account that contains the application files and binaries necessary to launch the applications your users will use. App blocks also include the setup script that informs the operating system how to handle the VHD file.

App blocks support two different types of packaging:

- Custom - Choose this option to create your application package (VHD) manually. For more information, see the section called “Custom App Blocks”.
- AppStream 2.0 - Choose this recommended option to create your application package using app block builder. For more information, see the section called “AppStream 2.0 App Blocks”.

Contents

- Custom App Blocks
- AppStream 2.0 App Blocks
- Unsupported Applications

Custom App Blocks

Elastic fleet streaming instances utilize applications that are installed on virtual hard disk (VHD) files stored within an Amazon S3 bucket in your account. App blocks with custom packaging gives you the flexibility to create your own VHD file, and upload it to an Amazon S3 bucket within your account.

Contents

- Create the VHD
- Create the Setup Script for the VHD
- Create a Custom App Block
- Update the App Block, VHD, and Setup Script

Create the VHD

A VHD is a single file that when mounted to the operating system is treated like a hard disk. The VHD can be mounted as a drive letter, to a folder path, or both. When the VHD is mounted, you can treat it as you would any other hard disk, including installing your application or copying files to it that your user will need.

To create the app block, you will need to create the VHD, install your applications to it, then detach it. Once detached you can test your VHD on another PC, an EC2 instance, or an AppStream 2.0 image builder to validate the applications work as expected. Once completed, upload to an Amazon S3 bucket in your account and create the app block.

> **Note**
>
> This page describes using a VHD to deliver your application; however, the AppStream 2.0 streaming instance will download any object from Amazon S3. The object you store in Amazon S3 can also be a zip file, application installer, or the application executable itself. You can use the setup script to configure it correctly on the streaming instance before a user launches their application.
>
> The AppStream 2.0 streaming instance waits up to 90 seconds for the VHD to complete downloading before the setup script runs. If the VHD does not complete downloading within this duration, the download stops, and the setup script will not run.
>
> We recommend a maximum size of 1.5 gigabyte for the VHD. You might be able to reduce the size of the VHD by compressing. You must use the setup script to decompress it before

mounting it, because the file needs to be fully downloaded from Amazon S3 before it can be mounted and the application is launched. Larger VHDs increase the time it takes for the application to launch and the streaming session to begin.

To create a VHD for Microsoft Windows

1. From a Windows PC or Windows Amazon Elastic Compute Cloud (Amazon EC2) instance, open a command prompt with administrative privileges.
2. Launch the Microsoft **diskpart** utility by entering the following command:

   diskpart
3. Create the unformatted and uninitialized VHD file by entering the following command, where *<maximum file size>* is the size of the VHD file, in MB:

   create vdisk file=C:\path\to\new\file.vhdx maximum=*<maximum file size>* type=expandable
4. Select the newly created VHD by entering the following command:

   select vdisk file=C:\path\to\new\file.vhdx
5. Attach the newly created VHD by entering the following command:

   attach vdisk
6. Initialize the newly created VHD by entering the following command:

   convert mbr
7. Create the primary partition spanning the entire VHD by entering the following command:

   create partition primary
8. Format the newly created partition by entering the following command:

   format fs=ntfs quick
9. You can mount your newly created VHD to an unused drive letter, a folder path on the root volume, or both.

   To mount a drive letter, enter: **assign letter=*<unused drive letter>***

   To mount a folder, enter: **assign mount=*C:\path\to\empty\folder\to\mount***

> **Note**
>
> To mount to a folder path, the folder must already exist and must be empty.

10. You can now install your application to the VHD, using either the drive letter or the folder mount path chosen in step 9.

After you finish installing your application(s) to the VHD, you need to detach it before you can safely upload it to an Amazon S3 bucket.

To detach a VHD for Microsoft Windows

1. Launch the Microsoft diskpart utility by entering the following command:

   diskpart

2. Select the VHD by entering the following command:

   select vdisk file=*C:\path\to\new\file.vhdx*

3. Detach the VHD by entering the following command:

   detach vdisk

4. The VHD has now been detached, and can be tested on another Windows PC, Amazon EC2 instance, or an AppStream 2.0 image builder.

To create a VHD for Linux

1. From an Amazon Linux 2 EC2 instance, Amazon Linux 2 AppStream 2.0 image builder, or Amazon Linux 2 WorkSpaces, open a terminal session.
2. Create the unformatted and uninitialized VHD file:

   dd if=/dev/zero of=*<name of file>* bs=*<size of VHD>* count=1

3. Add a file system to the created VHD by entering the following command:

   sudo mkfs -t ext4 *<name of file>*

> **Note**
>
> You might see a message stating that the file is not a block special device. You can select to proceed anyway.

4. Create an empty folder to use for the mount point by entering the following command:

   sudo mkdir ***/path/to/mount/point***

5. Mount the newly created VHD to a file system path by running the following command:

   sudo mount -t auto -o loop ***<name of file> /path/to/mount/point***

6. You can now install your application to the VHD using the folder mount path chosen in step 4.

> **Note**
>
> The default permissions for files and folders created on the VHD can prevent non-administrator users from launching applications or reading files. Validate the permissions and change them, if necessary.

After you finish installing your application(s) to the VHD, you need to detach it before you can safely upload it to an Amazon S3 bucket.

To detach a VHD for Linux

1. Open a terminal session, and enter the following command:

   sudo umount ***/path/to/mount/point***

2. The VHD has now been detached, and can be tested on another Amazon Linux 2 Amazon EC2 instance, Amazon Linux 2 AppStream 2.0 image builder, or Amazon Linux 2 WorkSpaces.

Create the Setup Script for the VHD

AppStream 2.0 uses a setup script that you provide to mount the VHD before the application launches. You can also use the setup script to complete other tasks required to make your application work. For example, you can configure registry keys, register DLLs, manage pre-requisites, or modify the user profile from the setup script. AppStream 2.0 provides script examples

that you can use to mount your VHD. You will need to modify these scripts for your VHD and application needs.

> **Note**
>
> Setup scripts aren't required for app blocks with AppStream 2.0 packaging. However, you can provide optional post-setup scripts to customize application installation.

Use the following links to download the example scripts:

- Amazon Linux 2 bash script
- Microsoft Windows Powershell script

> **Note**
>
> AppStream 2.0 and the Microsoft Windows operating system reserve drive letters A through E. Don't mount VHDs or network shares to these drive letters.

AppStream 2.0 downloads the setup script and VHD to a directory on the fleet streaming instance, then runs the setup script. The setup script runs on the operating system with full administrator rights. The setup script runs in the `SYSTEM` context on Microsoft Windows, and as the `root` user on Amazon Linux 2.

File system location for the VHD and setup script:

- Amazon Linux 2:

  `/opt/appstream/AppBlocks/`*`appblock-name`*`/`

  `appblock-name`

  The name of the app block that the VHD and setup script correspond to.

- Microsoft Windows:

  `C:\AppStream\AppBlocks\`*`appblock-name`*`\`

  `appblock-name`

  The name of the app block that the VHD and setup script correspond to.

AppStream 2.0 maintains the file name as they are on the object. For example, if your app block is named `MyApps`, with a VHD named `apps.vhd` and setup script named `mount-apps.ps1`, then the full path on a Windows streaming instance is:

- VHD

  `C:\AppStream\AppBlocks\MyApps\apps.vhd`

- Setup script

  `C:\AppStream\AppBlocks\MyApps\mount-apps.ps1`

AppStream 2.0 captures the standard error and standard output from your setup script when it runs on a fleet streaming instance and uploads the output to an Amazon S3 bucket within your account. You can use these logs to identify and resolve issues you may have with your setup script. The buckets are named in a specific format as follows:

```
appstream-logs-region-code-account-id-without-hyphens-random-identifier
```

`region-code`

This is the AWS Region code in which the elastic fleet is created within.

`account-id-without-hyphens`

Your AWS account identifier. The random ID ensures that there is no conflict with other buckets in that Region. The first part of the bucket name, appstream-logs, does not change across accounts or Regions.

For example, if you create an elastic fleet in the US West (Oregon) Region (us-west-2) on account number 123456789012, AppStream 2.0 creates an Amazon S3 bucket within your account in that Region with the name shown. Only an administrator with sufficient permissions can delete this bucket.

```
appstream-logs-us-west-2-1234567890123-abcdefg
```

The path for the folder where the log files are stored in the S3 bucket in your account uses the following structure:

```
bucket-name/fleet-name/instance-id/appblock-name/
```

`bucket-name`

The name of the Amazon S3 bucket in which the setup script logs are stored. The name format is described earlier in this section.

`Instance-id`

The unique identifier for the streaming instance that the setup script ran on

`appblock-name`

The name of the appblock that the setup script corresponds to.

The following example folder structure applies to a streaming session started from `test-fleet`. The session is from an AWS account ID of 123456789012, and appblock name is testappblock in the US West (Oregon) Region (us-west-2):

```
appstream-logs-us-west-2-1234567890123-abcdefg/test-fleet/
i-084427ab4a1cff7f5/testappblock/
```

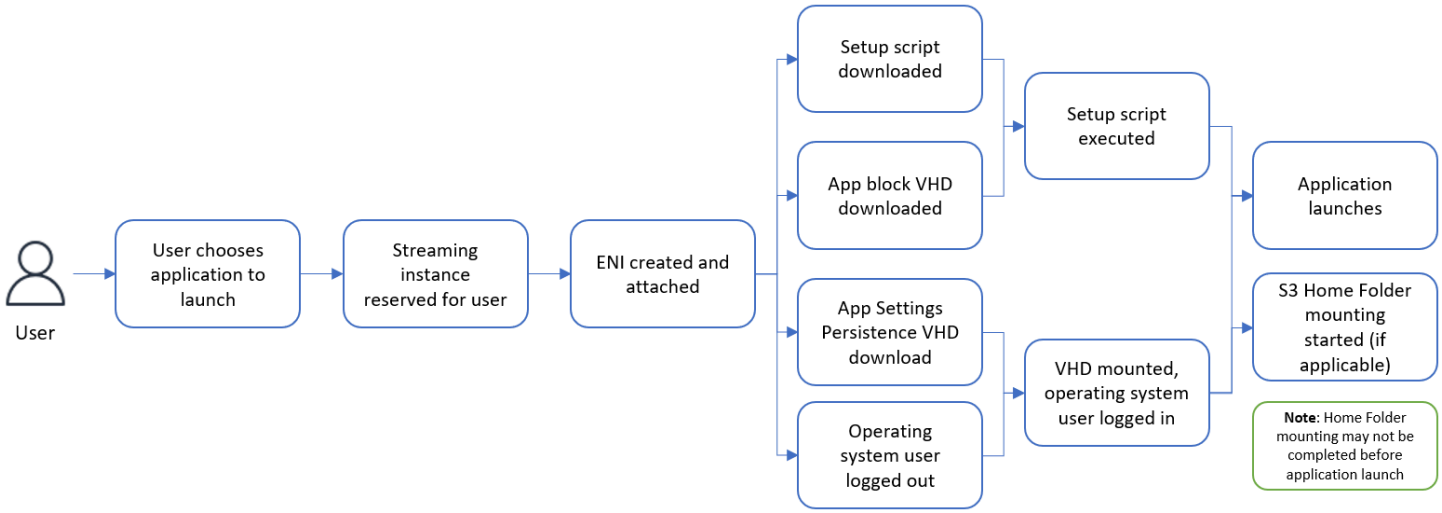
This example folder structure contains one log file for the standard output, and one log file for the standard error.

App block setup script execution

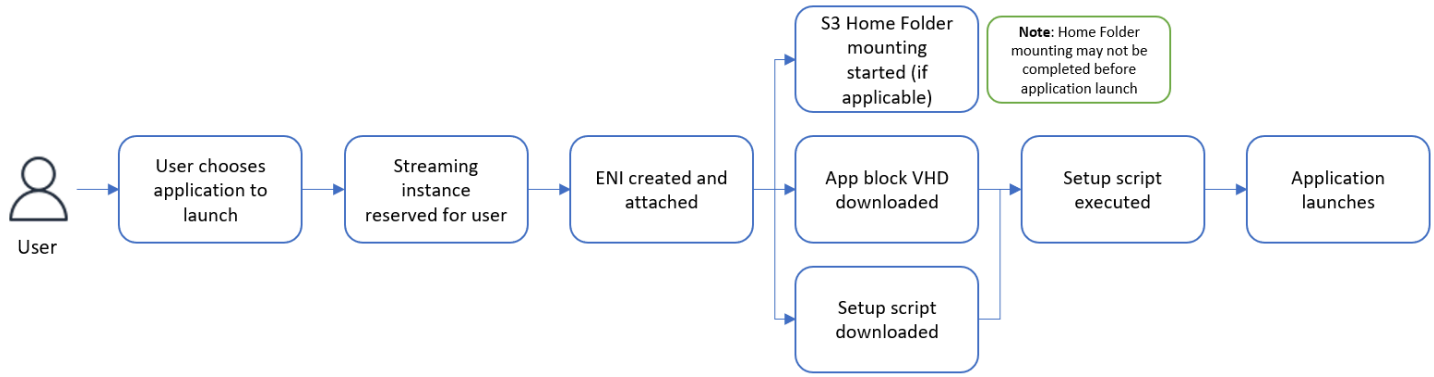
The following diagrams indicate where in the process the setup script runs. The run order is dependent upon whether Application Settings Persistence is enabled on the stack associated with the elastic fleet.

> **Note**
>
> AppStream 2.0 uses your VPC details to download the VHD and setup script from the Amazon S3 bucket. Your VPC must provide access to the Amazon S3 bucket. For more information, see Using Amazon S3 VPC Endpoints for AppStream 2.0 Features.

Application Settings Persistence is enabled:



Application Settings Persistence is disabled:



Create a Custom App Block

You can use the AppStream 2.0 console to create the app block resource once you have your VHD and setup script created and uploaded to an S3 bucket in your AWS account. To learn more about storing the VHD and setup script in an Amazon S3 bucket, see the section called "Store Application Icon, Setup Script, Session Script, and VHD in an S3 Bucket".

> **Note**
>
> You must have IAM permissions to perform the `S3:GetObject` action on the VHD and setup script objects in the Amazon S3 bucket to create the app block resource.

To create the app block resource

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.

2. From the left-hand navigation menu, choose **Applications**, **App block**, and **Create app block**.
3. For app block packaging, select **Custom**.
4. For **App block details**, type a unique name identifier for the app block. Optionally, you can also specify the following:
   - **Display name** – A friendly name for the app block.
   - **Description** – A description for the app block.
5. For **Virtual hard disk object in S3** under **Script settings**, either enter the S3 URI that represents the VHD object, or choose **Browse S3** to navigate to your S3 buckets and find the VHD object.
6. For **Setup script object in S3** under **Script settings**, either enter the S3 URI that represents the setup script object, or choose **Browse S3** to navigate to your S3 buckets and find the setup script object.
7. For **Setup script executable** under **Script settings**, enter the executable necessary for your setup script.

   > **Note**
   >
   > If your setup script can execute directly, enter the filename of the setup script. If your setup script relies on another executable (for example, Microsoft PowerShell) to execute, enter the path to that executable.
   > Path to Microsoft PowerShell on Microsoft Windows:
   > `C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe`

8. Optionally, for **Setup script executable arguments** under **Script settings**, enter in the arguments that need to be provided to the setup script executable to execute your setup script.

   > **Note**
   >
   > If you are using a Microsoft PowerShell script, you must specify the "-File" parameter with the name of your setup script as an executable argument. Additionally, ensure that the Execution Policy allows your script to be run. To learn more, see about_Execution_Policies and What is PowerShell?.

9. For **Execution duration in seconds** under **Script settings**, enter the timeout duration for your setup script.

> **Note**
>
> The execution duration in seconds is how long AppStream 2.0 waits for the setup script to run before continuing. If your setup script doesn't complete within this duration, an error is displayed to your user and the application will attempt to launch. The setup script is terminated after the execution duration has elapsed.

10. (Optional) For **Tags**, create tags for the app block resource
11. Review the information that you entered, and choose **Create**.
12. If your app block was created successfully, you see a success message at the top of the console. If an error occurred, you see a descriptive error message and will need to try creating the app block again.

Update the App Block, VHD, and Setup Script

App block resources are immutable and do not allow you to change them once created. If you need to make backwards compatible updates to the VHD or setup script, it is recommended that you upload a new version of the file to the Amazon S3 bucket, overwriting the current version. New Elastic fleet streaming sessions will download the latest version of the objects, and use them.

If you need to make backwards incompatible updates to the VHD or setup script, it is recommended that you upload them as new objects to the Amazon S3 bucket, and create a new app block and application resource. You can then manage the deployment to your users as part of a change window or other outage.

AppStream 2.0 App Blocks

Elastic fleet streaming instances utilize applications that are installed on virtual hard disk (VHD) files stored within an Amazon S3 bucket in your account. When it comes to app blocks with custom packaging, you have the flexibility to create your own VHD file and upload it to an Amazon S3 bucket within your account. Alternatively, for app blocks with AppStream 2.0 packaging, you can take advantage of the app block builder, which handles the packaging of your applications, creates a VHD file, and uploads it to your Amazon S3 bucket.

By using the AppStream 2.0 packaged app block, you not only eliminate the need for manual steps in building a VHD file, but also remove the requirement for a setup script. It expands application compatibility with elastic fleets, as well as reduces manual administrative steps required to create

an app block. AppStream 2.0 handles the setup of app blocks with AppStream 2.0 packaging automatically without the need of any setup scripts. However, you can still provide optional post-setup scripts to customize the installation for your needs.

Contents

- Overview
- Unsupported Applications
- Create an AppStream 2.0 App Block
- Activate an App Block
- Create an App Block with an Existing App Package
- Test an App Block
- Associate or Disassociate an App Block

Overview

To create an app block with AppStream 2.0 packaging, you need to initiate a streaming session with an app block builder. After the session is launched, you can download your application installers and enable the recording options. From that point onwards, AppStream 2.0 records the file system and registry changes made on the app block builder using Application Redirection technology.

Application Redirection uses Windows filter driver framework to intercept and redirect file-system and registry changes. This redirection is seamless to the application being installed. The application will continue to interact with the original file locations on the C: drive. For example, if an installer for "TestApplication" is run on a machine with App Redirection set up, it will be installed by default to C:\Program Files\TestApplication. However, behind the scenes, all files and folders will be redirected to a mounted virtual hard disk (VHD), and a link will be created from the original file location to the actual file location. On the machine, TestApplication will still appear to be installed at C:\Program Files\TestApplication.

After all the installation changes are recorded, the VHD file is uploaded to an Amazon S3 bucket in your account.

When a user requests a session using an Elastic fleet, AppStream 2.0 downloads the VHD file, sets up the application, runs the post-installation setup scripts (optional), and starts the application streaming.

> **Note**
>
> Application Redirection technology does not record any file system changes under %USERPROFILE%, except new directories created under %APPDATA% and %LOCALAPPDATA% directories.
> Application Redirection technology does not record any registry changes under the current user, HKEY_CURRENT_USER (HKCU).

Unsupported Applications

Applications might encounter failures when installing or running in the following scenarios:

- **Applications with location-based checks during installation**: If an application's installation process verifies the actual location of the installed files, it might result in a failure. Because AppStream 2.0 redirects the files to the app block VHD, only links to the actual files are maintained at the original location.

If you are uncertain whether your application falls into any of these categories, you can use AppStream 2.0 packaging to create an app block. This process involves installing your application(s) on an app block builder instance. In the event that your application(s) fail to install on the app block builder instance, you can take the following actions:

- Check the logs. The error log file for your app block builder instance can be found at C:\AppStream\AppBlocks\errorLog. This log records all installation failures, including RegKeys/File operation processing. If you see any of the following logs in the errorLog, it indicates that the packaging of your application is currently unsupported by the AppStream 2.0 app block builder:
  - "Unable to create symbolic link"
  - "Service doesn't support file renaming"

  If there is no errorLog file, or if this file is empty, then check your application installation logs to identify the reason for failures.
- Report a problem. Select the **Report a problem** button, which is available on the application builder assistant in the app block builder. Selecting this option will gather all the AppStream 2.0 logs from your app block builder instance, and submit them to the AppStream 2.0 team for assistance.

- Create an app block with custom packaging: If you are unable to package your applications using the app block builder, you can try to create an app block using custom packaging methods. For more information, see the section called "Custom App Blocks".
- If you need more help, contact AWS Support. For more information, see AWS Support Center.

It is important to consider these potential limitations, and plan accordingly when using AppStream 2.0 packaging for your applications.

Create an AppStream 2.0 App Block

Follow these steps to create an app block with the AppStream 2.0 packaging type.

Step 1: Configure the app block

To configure the app block

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. From the left-hand navigation menu, choose **Applications Manager**, **App blocks**, and **Create app block**.
3. For app block packaging, select **AppStream 2.0**.
4. For **App block details**, type a unique name identifier for the app block. Optionally, you can also specify the following:
   - **Name** – A unique name for the app block.
   - **Display name** (optional) – A friendly name for the app block.
   - **Description** (optional) – A description for the app block.
5. (Optional) An app block with AppStream 2.0 packaging doesn't need a setup script. You can optionally provide post-installation steps the following **Advanced Options**:
   - For **Post setup script object in S3**, either enter the Amazon S3 URI that represents the post setup script object, or choose **Browse S3** to navigate to your Amazon S3 buckets and find the setup script object.
   - For **Post setup script executable**, enter the executable needed for your post setup script.

> **Note**
>
> If your setup script can execute directly, enter the filename of the setup script. If your setup script relies on another executable (for example, Microsoft PowerShell) to execute, enter the path to that executable.
> Path to Microsoft PowerShell on Microsoft Windows:
> `C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe`
> Optionally, for **Setup script executable arguments** under **Script settings**, enter in the arguments that need to be provided to the setup script executable to execute your setup script.

> **Note**
>
> If you are using a Microsoft PowerShell script, you must specify the "-File" parameter with the name of your setup script as an executable argument. Additionally, ensure that the Execution Policy allows your script to be run. To learn more, see about_Execution_Policies and What is PowerShell?.
> For **Execution duration in seconds** under **Script settings**, enter the timeout duration for your setup script.

> **Note**
>
> The execution duration in seconds is how long AppStream 2.0 waits for the setup script to run before continuing. If your setup script doesn't complete within this duration, an error is displayed to your user and the application will attempt to launch. The setup script is terminated after the execution duration has elapsed.

6. Under **Import Settings**, choose **Create new app block application file**. For **S3 Location** under **Import settings**, either enter the Amazon S3 URI that represents the bucket, or choose **Browse S3** to navigate to your Amazon S3 buckets and select an appropriate bucket. The list of Amazon S3 buckets is global and lists all the buckets across all regions. Make sure you select the bucket in the region where you want to create your app block. For more information about

setting bucket permissions, see the section called "Store Application Icon, Setup Script, Session Script, and VHD in an S3 Bucket".

7. Select an app block builder. Only app block builders that are not associated with other app blocks are available. If the list is empty, either create a new app block builder, or disassociate the existing ones to use. App block builder is a reusable resource that you can use to create your application package.

> **Note**
>
> If you do not select an app block builder here, you can still create your app block in the **Inactive** state, and activate your app block later. For more information, see the section called "Activate an App Block".

8. (Optional) For **Tags**, create tags for the app block resource.
9. Choose **Next**.
10. Review the information that you entered, and choose one of the following options:
    - Choose **Create app block** if you didn't select an app block builder in step 7.
    - Choose **Launch app block builder** if you chose an app block builder in step 7. Then continue to Step 2 to create your application package using the app block builder streaming session.

At this point, your app block resource is created, but it is **Inactive** and can't be used for Elastic fleets.

Step 2: Create the Application Package

Use the app block builder streaming instance to package your applications and activate your app block. The app block created using app block builder will have AppStream 2.0 packaging, and the application package will be uploaded onto the Amazon S3 bucket in your AWS account.

To create the application package

1. After your streaming session is on, the application builder assistant automatically starts. If it doesn't start, start it manually using the desktop icon.
2. The initial screen provides instructions for the application packaging process.
3. Bring your application installer onto your app block builder streaming session by using one of the following options:

- Download the application installers from the web.
- Use your streaming session file interface.
- Download the application installer from another AWS service using a machine role.

4. After you have all the required application installers, stop all the other apps running on the instance and choose **Start recording**. The app block builder starts recording system changes, and the screen says **Recording in progress**.
5. Start installing your applications one by one.
6. When you are done with application installation, choose **Stop recording**, and the system will stop recording changes. If you want to make any more changes to your application package, such as add more applications or remove an already installed application, choose **Start recording**, and make sure the system is in **Recording in progress** mode.

> **Note**
>
> If your application installation fails, choose **Report a problem** to collect AppStream 2.0 related logs from the instance, and report the problem to the AppStream 2.0 team. When you are done, end your app block builder streaming session. You can try to restart the process creating an app block by using a new app block builder instance. If the problem persists, then try to create your app block using custom packaging.

7. When you are done installing all the applications, choose **Stop recording**. You can test your application, by using the Start Menu or browsing the application using File Explorer.
8. Choose **Next** to review your app block details.

> **Note**
>
> The recommended size of an application package (VHD) file for an Elastic fleet is less than 1.5 GB. If your VHD file size is bigger than 1.5 GB, try reducing the number of applications packaged within one app block.
> Application package (VHD) file size will not shrink if you uninstall an application. Restart the application packaging process using a new app block streaming session, and install fewer applications.

9. Choose **Finish app block creation and disconnect** to create the application package and upload it to the Amazon S3 bucket. If you are successful, the streaming session will automatically disconnect, and the app block will be in an **Active** state.

> **Note**
>
> If your application installation fails, choose **Report a problem** to collect AppStream 2.0 related logs from the instance, and report the problem to the AppStream 2.0 team. When you are done, end your app block builder streaming session. You can try to restart the process creating an app block by using a new app block builder instance. If the problem persists, then try to create your app block using custom packaging.

Activate an App Block

If an app block with AppStream 2.0 packaging was created, but the application package (VHD) was not attached to it, then the app block will be in an inactive state, and it can't be used to associate applications with Elastic fleets. To activate an app block, an application package (VHD) must be associated with the app block.

To create the application package

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. From the left-hand navigation menu, choose **Applications Manager**, **App blocks**.
3. Select an **Inactive** app block that you want to activate, and choose **Activate** from the **Actions** menu.
4. Select an app block builder, and choose **Launch app block builder**.
   - If the list is empty, then you either don’t have an app block builder, or all of your app block builders are associated with other app blocks. Either create a new app block builder, or disassociate an existing app block builder and test it.
   - If the app block builder is already associated with an app block, then you can continue using it for activating the app block.
   - If the selected app block builder was not associated with an app block builder, then it will be associated with the one you select, and the streaming session will launch. The app block builder remains associated with this app block after the session ends.
5. After the app block builder streaming session starts, follow the steps in the section called “Step 2: Create the Application Package” to create your application package (VHD) and activate the app block.

Create an App Block with an Existing App Package

You can use your existing application package (VHD) with AppStream 2.0 packaging to create AppStream 2.0 app blocks. To do this, copy your application package (VHD) file from the source Amazon S3 bucket to another destination Amazon S3 bucket. The destination bucket can be in a different region.

To create an app block with existing app package

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. From the left-hand navigation menu, choose **Applications Manager**, **App blocks**, and **Create app block**.
3. For app block packaging, select **AppStream 2.0**.
4. For **App block details**, type a unique name identifier for the app block. Optionally, you can also specify the following:
   - **Name** – A unique name for the app block.
   - **Display name** (optional) – A friendly name for the app block.
   - **Description** (optional) – A description for the app block.
5. (Optional) An app block with AppStream 2.0 packaging doesn't need a setup script. You can optionally provide post-installation steps the following **Advanced Options**:
   - For **Post setup script object in S3**, either enter the Amazon S3 URI that represents the post setup script object, or choose **Browse S3** to navigate to your Amazon S3 buckets and find the setup script object.
   - For **Post setup script executable**, enter the executable needed for your post setup script.

> **Note**
>
> If your post setup script can execute directly, enter the filename of the post setup script. If your post setup script relies on another executable (for example, Microsoft PowerShell) to execute, enter the path to that executable.
> Path to Microsoft PowerShell on Microsoft Windows:
> `C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe`
> Optionally, for **Post setup script executable arguments**, enter in the arguments that need to be provided to the setup script executable to execute your setup script.

> **Note**
>
> If you are using a Microsoft PowerShell script, you must specify the "-File" parameter with the name of your post setup script as an executable argument. Additionally, ensure that the Execution Policy allows your script to be run. To learn more, see about_Execution_Policies and What is PowerShell?.
> For **Execution duration in seconds** under **Script settings**, enter the timeout duration for your setup script.

> **Note**
>
> The execution duration in seconds is how long AppStream 2.0 waits for the post setup script to run before continuing. If your post setup script doesn't complete within this duration, an error is displayed to your user and the application will attempt to launch. The setup script is terminated after the execution duration has elapsed.

6. Choose **Use existing app block application file** under **Import settings**. For **S3 Location**, you can enter the Amazon S3 URI for the object in an Amazon S3 bucket that represents the application package (VHD), Or, choose **Browse S3** to navigate to your Amazon S3 buckets and select the object in an Amazon S3 bucket. The list of Amazon S3 buckets is global and lists all the buckets across all regions. Make sure you select the bucket in the region where you want to create your app block.
7. Choose **Next**.
8. Review the information that you entered, and choose **Create app block**.

At this point your app block resource is created and in the **Active** state.

Test an App Block

You can use an app block builder to test your app block and verify your application functionalities. You don't need to launch an Elastic fleet for this option. You can also create multiple app block builders with different instance types or sizes, and test the performance of your application with different compute options.

> **Note**
>
> The test app block option is supported only for app blocks with AppStream 2.0 packaging.

To test an app block

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. From the left-hand navigation menu, choose **Applications Manager**, **App blocks**.
3. Select an app block that you want to test, and choose **Test** from the **Actions** menu.
4. Select an app block builder, and choose **Launch and test app block**.
   - If the list is empty, then you either don't have an app block builder, or all of your app block builders are associated with other app blocks. Either create a new app block builder, or disassociate an existing app block builder and test it.
   - If the app block builder is already associated with an app block, then you can continue using it for activating the app block.
   - If the selected app block builder was not associated with an app block builder, then it will be associated with the one you select, and the streaming session will launch. The app block builder remains associated with this app block after the session ends.
5. App block builder launches in a separate browser window in a Desktop streaming mode. The service downloads the app block from the Amazon S3 bucket and installs it on the app block builder instance.
6. Your applications can now be streamed and tested. You can open your application by either browsing it in File Explorer or using the Start menu.
7. When you are done testing, end the streaming session.

Associate or Disassociate an App Block

In order to create, test, or activate your app block with AppStream 2.0 packaging, you need to associate it with an app block builder. One app block builder can only be associated with only one app block, and vice versa.

If all your app block builders are associated with other app blocks, and you want to test, create, or activate another app block, then you can either create a new app block builder, or disassociate an existing app block builder from the app block and use it with the new app block.

> **Note**
>
> Associating and disassociating an app block is only supported for app blocks with AppStream 2.0 packaging.
> Disassociation is allowed only if an app block builder is in the **STOPPED** state.

Associate an app block with app block builder

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. From the left-hand navigation menu, choose **Applications Manager**, **App blocks**.
3. Select an app block, and choose **Associate** from the **Actions** menu.
4. Select an app block builder, and choose **Associate app block builder**.

   If the list is empty, then you either don't have an app block builder, or all of your app block builders are associated with other app blocks. Either create a new app block builder, or disassociate an existing app block builder and then associate.

Disassociate an app block from an app block builder

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. From the left-hand navigation menu, choose **Applications Manager**, **App blocks**.
3. Select an app block, and choose **Disassociate** from the **Actions** menu.
4. Select an already associated app block builder, and choose **Disassociate app block builder**.

Unsupported Applications

Applications might encounter failures when installing or running in the following scenarios:

- **Applications requiring reboots after installation**: If an application needs to perform additional changes or configurations after installation that require a reboot, it might fail. Currently, app block builder does not support restart, which can prevent the application from completing its required post-installation steps.
- **Applications relying on user-specific details**: Applications that are intended to be installed only for the currently logged-in user on app block builder, or that rely on the logged-in user details on app block builder, such as security identifiers (SIDs) during installation, might not

function correctly on Elastic fleets. This is due to the logged-in user changes within the elastic fleet environment. Additionally, application redirection does not record all directories under %USERPROFILE%. However, you have the option to configure post setup scripts to dynamically change your application configuration based on environment.

- **Applications relying on machine-specific details**: Applications that rely on machine-specific details on app block builder during installation, such as network adaptor GUID, might encounter issues on Eastic fleets. This is because the machine details, including network adaptor GUIDs, can change within the elastic fleet environment. To address this, you can configure the post setup scripts to handle the configuration of those machine-specific details.

If you are uncertain whether your application falls into any of these categories, you can use AppStream 2.0 packaging to create an app block. This process involves installing your application(s) on an app block builder instance. In the event that your application(s) fail to install on the app block builder instance, you can take the following actions:

- Check the logs. The error log file for your app block builder instance can be found at C:\AppStream\AppBlocks\errorLog. This log records all installation failures, including registry keys and file operation processing. If you see any of the following logs in the errorLog, it indicates that the packaging of your application is currently unsupported by the AppStream 2.0 app block builder:
  - "Unable to create symbolic link"
  - "Service doesn't support file renaming"

  If there is no errorLog file, or if this file is empty, then check your application installation logs to identify the reason for failures.
- Report a problem. Select the **Report a problem** button, which is available on the application builder assistant in the app block builder. Selecting this option will gather all the AppStream 2.0 logs from your app block builder instance, and submit them to the AppStream 2.0 team.
- Create an app block with custom packaging: If you are unable to package your applications using the app block builder, you can try to create an app block using custom packaging methods. For more information, see the section called “Custom App Blocks”.
- If you need more help, contact AWS Support. For more information, see AWS Support Center.

It is important to consider these potential limitations, and plan accordingly when using AppStream 2.0 packaging for your applications.

App Block Builder

An app block builder is a reusable resource that you can use to package your applications (or app block). You can also use it to test your application package before associating your application to an Elastic fleet. A single app block builder can be used to create and test multiple app blocks one by one. Each time a streaming session is created for app block builder for creating or testing an app block, a new instance is created and used. After the app block builder instance is terminated, the state of the instance is not persisted.

AppStream 2.0 Elastic fleets use Amazon EC2 instances to stream applications. You must provide your application package and associate it with your fleet. To create your own custom application packaging, connect to an app block builder instance, and then install and configure your applications for streaming. App block builder creates the packaging for your application and uploads it to an Amazon S3 bucket in your AWS account.

When you create an app block builder, you choose the following:

- An instance type — AppStream 2.0 provides different instance sizes with various CPU and memory configurations. The instance type must align with the instance family you need.
- The VPC, subnets, and security groups to use — Make sure that the subnets and security groups provide access to the network resources that your applications require. Typical network resources required by applications might include licensing servers, database servers, file servers, and application servers. App block builder uploads the application package on to an Amazon S3 bucket in your AWS account. The VPC you choose for your fleet must provide sufficient network access to the Amazon S3 bucket. For more information, see the section called “Store Application Icon, Setup Script, Session Script, and VHD in an S3 Bucket”.

Contents

- Create an App Block Builder
- Connect to an App Block Builder
- App Block Builder Actions

Create an App Block Builder

You can use app block builder instance to create your application package for AppStream 2.0 Elastic fleets.

To create an app block builder

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. Choose **Applications Manager** in the left navigation pane, then choose the **App block builders** tab and **Create app block builder**.
3. For **Step 1: Configure app block builder**, configure the app block builder by providing the following details:
   - **Name**: Type a unique name identifier for the app block builder.
   - **Display name (optional)**: Type a name to display for the app block builder (maximum of 100 characters).
   - **Operating system**: Select an operating system for you application. This must align with the operating system that you are going to select for you elastic fleet, which your end users will use to stream the application.
   - **IAM role (Optional)**: When you apply an IAM role from your account to an AppStream 2.0 app block builder, you can make AWS API requests from the app block builder instance without manually managing AWS credentials. To apply an IAM role to the app block builder, do either of the following:
     - To use an existing IAM role in your Amazon Web Services account, choose the role that you want to use from the **IAM role** list. The role must must be accessible from the image builder. For more information, see Configuring an Existing IAM Role to Use With AppStream 2.0 Streaming Instances.
     - To create a new IAM role, choose **Create new IAM role** and follow the steps in How to Create an IAM Role to Use With AppStream 2.0 Streaming Instances.
   - **Instance Type**: Select the instance type for the app block builder. Choose a type that matches the performance requirements of the applications that you plan to install.
   - **Tags (optional)**: Choose **Add Tag**, and type the key and value for the tag. To add more tags, repeat this step. For more information, see Tagging Your Amazon AppStream 2.0 Resources.
4. Choose **Next**.
5. For **Step 2: Configure Network**, do the following:
   - To add internet access for the app block builder in a VPC with a public subnet, choose **Default Internet Access**. If you are providing internet access by using a NAT gateway, leave **Default Internet Access** unselected. For more information, see Internet Access.

- For **VPC** and **Subnet 1**, choose a VPC and at least two subnets. For increased fault tolerance, we recommend that you choose three subnets in different Availability Zones. For more information, see Configure a VPC with Private Subnets and a NAT Gateway.

  If you don't have your own VPC and subnet, you can use the default VPC or create your own. To create your own, choose the **Create a new VPC** and **Create new subnet** links to create them. Choosing these links opens the Amazon VPC console. After you create your VPC and subnets, return to the AppStream 2.0 console and choose the refresh icon to the left of the **Create a new VPC** and **Create new subnet** links to display them in the list. For more information, see Configure a VPC for AppStream 2.0.
- For **Security group(s)**, choose up to five security groups to associate with this image builder. If you don't have your own security group and you don't want to use the default security group, choose the **Create new security group** link to create one. After you create your subnets in the Amazon VPC console, return to the AppStream 2.0 console and choose the refresh icon to the left of the **Create new security group** link to display them in the list. For more information, see Security Groups in Amazon AppStream 2.0.
- For **VPC Endpoints (Optional)**, you can create an interface VPC endpoint (interface endpoint) in your virtual private cloud (VPC). To create the interface endpoint, choose **Create VPC Endpoint**. Selecting this link opens the VPC console. To finish creating the endpoint, follow steps 3 through 6 in the section called "Creating and Streaming from Interface VPC Endpoints". After you create the interface endpoint, you can use it to keep streaming traffic within your VPC.

6. Choose **Next**.
7. Choose **Review** and confirm the details for the app block builder. To change the configuration for any section, choose **Edit** and make the needed changes.
8. After you finish reviewing the configuration details, choose **Create app block builder**.

> **Note**
>
> If an error message notifies you that you don't have sufficient limits (quotas) to create the image builder, submit a limit increase request through the Service Quotas console at https://console.aws.amazon.com/servicequotas/. For more information, see Requesting a quota increase in the *Service Quotas User Guide*.

Connect to an App Block Builder

You can connect to an app block builder by doing either of the following:

- Using the AppStream 2.0 console (for browser connections only)
- Creating a streaming URL (for browser or AppStream 2.0 client connections)

> **Note**
>
> App block builder doesn't support Active Directory domain join.

Contents

- AppStream 2.0 Console (Browser Connection)
- Streaming URL (AppStream 2.0 Client or Browser Connection)

AppStream 2.0 Console (Browser Connection)

To use the AppStream 2.0 console to connect to an app block builder through a browser, complete the following steps.

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. In the left navigation pane, choose **Applications Manager**, and then choose **App block builders**.
3. In the list of app block builders, choose the app block builder to which you want to connect. Verify that the status of the app block builder is **Running**, and choose **Connect**.

   For this step to work, you might need to configure your browser to allow pop-ups from https://stream.<aws-region>.amazonappstream.com/.
4. Start streaming the app block builder.

Streaming URL (AppStream 2.0 Client or Browser Connection)

You can create a streaming URL to connect to an app block builder through a browser or the AppStream 2.0 client. Unlike a streaming URL that you create to enable user access to a fleet instance, which is valid for a maximum of seven days, by default, a streaming URL that you create

to access an image builder expires after one hour. To set a different expiration time, you must generate the streaming URL by using the CreateAppBlockBuilderStreamingURL API action.

You can create a streaming URL in any of the following ways:

- AppStream 2.0 console
- The CreateAppBlockBuilderStreamingURL API action
- The create-app-block-builder-streaming-url AWS CLI command

To create a streaming URL and connect to the app block builder by using the AppStream 2.0 console, complete the steps in the following procedure.

To create a streaming URL and connect to the app block builder by using the AppStream 2.0 console

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. In the navigation pane, choose **Application Manager**, **App block builders**.
3. In the list of app block builders, choose the app block builder to which you want to connect. Verify that the status of the app block builder is **Running**.
4. Choose **Actions**, **Create streaming URL**.
5. Do one of the following:
   - To save the streaming URL to connect to the app block builder later, choose **Copy Link** to copy the URL, then save it to an accessible location.
   - To connect to the app block builder through the AppStream 2.0 client, choose **Launch in Client**. When you choose this option, the AppStream 2.0 client sign-in page is prepopulated with the streaming URL.
   - To connect to the app block builder through a browser, choose **Launch in Browser**. When you choose this option, a browser opens with the address bar prepopulated with the streaming URL.
6. After you create the streaming URL and connect to the app block builder, start streaming the app block builder.

App Block Builder Actions

You can perform the following actions on an app block builder, depending on the current state (status) of the app block builder instance.

Delete

Permanently delete an app block builder.

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

Connect

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

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

Start

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

The instance must be in a **Stopped** state, and associated with an app block.

Stop

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

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

Update

Update any of the app block builder properties, except the name.

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

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

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

Applications

Applications contain the details necessary to launch your application after the VHD has been mounted. Applications also include the name and icon that are displayed to your user on the application catalog. Applications are associated with the app block resource that contains the files and binaries for that application.

You can use the AppStream 2.0 console to create the application resource once you have uploaded your application icon to an Amazon S3 bucket and created the app block that contains the files and folders necessary to launch the application. To learn more about uploading the application icon to an Amazon S3 bucket, see the section called "Store Application Icon, Setup Script, Session Script, and VHD in an S3 Bucket".

> **Note**
>
> You must have IAM permissions to perform the `S3:GetObject` action on the application icon object in the S3 bucket to create the application resource.

To create the application resource

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. From the left-hand navigation menu, choose **Applications** and **Create application**.
3. For **Name** under **Application details**, enter a unique identifier for the application.
4. (Optional) For **Display name** under **Application details**, enter a friendly name that users will see in the application catalog.
5. (Optional) For **Description** under **Application details**, enter a description for the application.
6. For **Application icon object in S3** under **Application details**, either enter the S3 URI that represents the VHD object, or choose **Browse S3** to navigate to your S3 buckets and find the application icon object.
7. For **Application executable launch path** under **Application settings**, enter the path on the streaming instance to the application's executable.
8. (Optional) For **Application working directory** in the **Application settings** section, enter the directory on the streaming instance to use for the application's working directory.
9. (Optional) For **Application launch parameters** in the **Application settings** section, enter the parameters to provide to the application executable when launching the application.

10. For **Supported operating systems (OS)** in the **Application settings** section, choose which operating systems can launch this application.
11. For **Supported instance families** in the **Application settings** section, choose which instance families can launch this application.
12. For **App block** in the **Application settings** section, choose which app block contains the files and folders necessary for this application.
13. (Optional) In the **Tags** section, create tags for the app block resource.
14. Review the information that you entered, then choose **Create**.
15. If your application was created successfully, you will see a success message at the top of the console. If an error occurred, a descriptive error message will be provided and you will need to try creating the application again.

Store Application Icon, Setup Script, Session Script, and VHD in an S3 Bucket

You must store the application icons, setup scripts, session scripts, and VHDs that you use for your applications and app blocks in an Amazon Simple Storage Service (Amazon S3) bucket in your AWS account. AppStream 2.0 Elastic fleets download the application icon, setup script, and VHD from the S3 bucket when your user starts their streaming session. The S3 bucket must reside in the AWS Region that you intend to create AppStream 2.0 Elastic fleets within.

We recommend that you create a new S3 bucket that is used to store only the application icons, setup scripts, session scripts, and VHDs that you intend to use with Elastic fleets. We also recommend enabling versioning on the S3 bucket. This allows reverting to previous object versions if necessary. For more information about how to create a new S3 bucket, see Creating a bucket. For more information about how to manage object versioning, see Using versioning in S3 buckets.

> **Note**
>
> AppStream 2.0 uses your VPC to access the S3 bucket you select. The VPC you choose for your fleet must provide sufficient network access to the S3 bucket.

Amazon S3 Bucket Permissions

The Amazon S3 bucket that you choose must have a bucket policy that provides sufficient access to the AppStream 2.0 service principal to access and download objects from the Amazon S3 bucket. You will need to modify the following bucket policy, then apply it to the Amazon S3 bucket you intend to use for application icons, setup scripts, and VHDs. For more information about how to apply a policy to an Amazon S3 bucket, see Adding a bucket policy using the Amazon S3 console.

Make sure that the access control lists (ACLs) for your Amazon S3 buckets are disabled. For more information, see Disabling ACLs for all new buckets and enforcing Object Ownership.

This section presents examples of typical use cases for bucket policies. These sample policies use *bucket* as the resource value. To test these policies, replace the *user input placeholders* with your own information (such as your bucket name).

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
       "Sid": "AllowAppStream2.0ToRetrieveObjects",
       "Effect": "Allow",
       "Principal": {
          "Service": ["appstream.amazonaws.com"]
        },
        "Action": ["s3:GetObject"],
        "Resource": [
           "arn:aws:s3:::bucket/VHD object",
           "arn:aws:s3:::bucket/Setup script object",
           "arn:aws:s3:::bucket/Application icon object",
           "arn:aws:s3:::bucket/Session scripts zip file object"
         ]
      }
  ]
}
```

Note

The bucket policy example defines specific objects in the S3 bucket that AppStream 2.0 can access. You can also use prefixes and wildcards to simplify policy management as you increase your app blocks. For more information about bucket policies, see Using

bucket policies. For more information about common bucket examples, see Bucket policy examples.

If you are using an AppStream 2.0 app block, then AppStream 2.0 requires additional permissions to upload the application package to your appropriate Amazon S3 bucket. For more information about AppStream 2.0 app blocks, see the section called "AppStream 2.0 App Blocks".

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAppStream2.0ToPutAndRetrieveObjects",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "appstream.amazonaws.com"
        ]
      },
      "Action": [
        "s3:GetObject",
        "s3:ListBucket",
        "s3:PutObject",
        "s3:GetBucketOwnershipControls"
      ],
      "Resource": [
        "arn:aws:s3:::bucket",
        "arn:aws:s3:::bucket/VHD object",
        "arn:aws:s3:::bucket/Setup script object",
        "arn:aws:s3:::bucket/Application icon object",
        "arn:aws:s3:::bucket/Session scripts zip file object"
      ]
    }
  ]
}
```

Associate Applications to Elastic Fleets

Applications must be associated to Elastic fleets before they appear to users in the application catalog to be launched. You can manage application associations using the AppStream 2.0 console.

For information about how to associate applications when creating an Elastic fleet, and how to manage application associations on existing fleets, see Update an AppStream 2.0 Fleet .

Additional Resources

The following links provide information and other resources to help you package and deliver your applications with Elastic fleets.

Solution	Description
AWS	• Use Elastic fleets and Linux for inexpensive, secure bastion hosts in Amazon AppStream 2.0 — Describes how to package common bastion host applications and deliver them using Linux instances on Elastic fleets. • Automate AppStream 2.0 Elastic fleet application updates with AWS Systems Manager — Describes how you can automate the creation of VHDs and app blocks using an EC2 instance with AWS Systems Manager. • Stream applications at a lower cost with Amazon AppStream 2.0 Elastic fleets and Linux compatibility — Describes how you can package applications into a VHD and deliver them using Linux instances on Elastic fleets.
Liquidware FlexApp	Liquidware FlexApp integration with AppStream 2.0 Elastic fleet — Includes step-by-step instructions to package any windows-based application into FlexApp one format and deliver on AppStream 2.0 Elastic fleets.
Numecent Cloudpaging	Introducing AppStream 2.0 Elastic Fleets: How to Maximize Application Compatibility and Portability with Cloudpaging — Numecent Cloudpaging eliminate s application compatibility issues, allowing even the most complex legacy and highly customized applicati ons to be packaged with their dependencies and ready for deployment within a day. AppStream 2.0 users can then stream them on-demand without the need for IT

Solution	Description
	to repackage for various Windows operating systems or devices.
Turbo.Net	Introducing Turbo support for Amazon AppStream 2.0 Elastic fleets — Turbo delivers applications instantly on major platforms and devices. Turbo's unique container technology eliminates installs and manages dependencies, conflicts, and entitlements. AppStream 2.0 users can immediately launch any application in a Turbo Hub.

Fleets and Stacks

With Amazon AppStream 2.0, you create fleet instances and stacks as part of the process of streaming applications. A fleet consists of streaming instances that run the image that you specify. A stack consists of an associated fleet, user access policies, and storage configurations.

Contents

- Fleet Type
- Session Context
- AppStream 2.0 Fleet Types
- AppStream 2.0 Instance Families
- Create an AppStream 2.0 Fleet and Stack
- Customize an AppStream 2.0 Fleet to Optimize Your Users' Application Streaming Experience
- Update an AppStream 2.0 Fleet
- Fleet Auto Scaling for Amazon AppStream 2.0
- Multi-Session Recommendations

Fleet Type

The fleet type allows you to decide when your instances run and how you pay for them. When your instances runs determines how quickly your users application will launch when it is selected. You specify the fleet type when you create a fleet, and can't change the fleet type after it is created.

The available fleet types are:

Always-On

Streaming instances run continuously, even when no users are streaming applications and desktops.

On-Demand

Streaming instances run only when users are streaming applications and desktops. Streaming instances not yet assigned to users are in a stopped state.

Elastic

The pool of streaming instances is managed by AppStream 2.0. When your users select their application or desktop to launch, they will start streaming after the app block has been downloaded and mounted to a streaming instance. For more information about creating app blocks for your Elastic fleets, see App Blocks.

Use an Always-On fleet to provide your users with instant access to their applications. Use an On-Demand fleet to optimize your streaming charges and provide your users with access to their applications after a 1-2 minute wait. For more information, see Amazon AppStream 2.0 Pricing.

Session Context

You can pass parameters to your streaming application by using either of the following methods:

- Specify session content in the CreateStreamingURL AppStream 2.0 API operation. For more information, see CreateStreamingURL.
- Specify the session context as a SAML assertion in your SAML 2.0 identity provider's authentication response. For more information, see Step 5: Create Assertions for the SAML Authentication Response.

If your image uses a version of the AppStream 2.0 agent that was released on or after October 30, 2018, the session context is stored within the image as a Windows or Linux environment variable. For information about specific environment variables, see "User and Instance Metadata for AppStream 2.0 Fleets" in Customize an AppStream 2.0 Fleet to Optimize Your Users' Application Streaming Experience .

> **Note**
>
> The session context parameter is visible to the user in the AppStream 2.0 streaming URL. We strongly recommend that you never put confidential or sensitive information in the session context parameter. Because it is possible for users to modify the streaming URL, we recommend performing additional validation to determine that the session context is valid for the end user. For example, you can compare the session context with other session information, such as user and instance metadata for AppStream 2.0 fleets. AppStream 2.0 does not perform validation on the session context parameter.

Using Session Context to Pass Parameters to a Streaming Application

In the following steps, you'll use session context to start a web browser and automatically open a specific website. For instances running Windows, you'll use Firefox. For instances running Linux, you'll use Chromium.

To use session context to launch a website

1. In the left navigation pane, choose **Images**, **Image Builder**.
2. Choose the image builder to use, verify that it is in the **Running** state, and choose **Connect**.
3. Log in to the image builder by choosing **Administrator** on the **Local User** tab.
4. Create a child folder of `C:\`. For this example, use `C:\Scripts`.
5. Create a Windows batch file in the new folder. For this example, create `C:\Scripts\session-context-test.bat` and add a script that launches Firefox with the URL from the session context.

   Use the following script:

   ```
   CD "C:\Program Files (x86)\Mozilla Firefox"
   Start firefox.exe %APPSTREAM_SESSION_CONTEXT%
   ```

6. In Image Assistant, add `session-context-test.bat` and change the name to **`Firefox`**.

   You do not need to add Firefox. This step requires that you add only the batch file.

7. Create an image, fleet, and stack. For this example, use a fleet name of **`session-context-test-fleet`** and a stack name of **`session-context-test-stack`**.
8. After the fleet is running, you can call create-streaming-url with the `session-context` parameter, as shown in this example.

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

9. Open the streaming URL in a browser. The script file launches Firefox and loads `http://www.amazon.com`.

Similarly, you can perform the following steps to pass parameters to your Linux streaming application.

To pass parameters to your Linux streaming application

1. In the left navigation pane, choose **Images**, **Image Builder**.
2. Choose the image builder to use, verify that it is in the **Running** state, and choose **Connect**.
3. Log in to the image builder by default as **ImageBuilderAdmin**.
4. Create a script file (for example, launch-chromium.sh) by running the following command:

   sudo vim /usr/bin/launch-chromium.sh

5. Write the script and set executable permissions, such as the following:

   > **Note**
   >
   > #!/bin/bash and source /etc/profile are always required in the script.

   ```
   #!/bin/bash
   source /etc/profile
   /usr/bin/chromium-browser $APPSTREAM_SESSION_CONTEXT
   ```

6. Use the Image Assistant CLI to add launch-chromium.sh:

   ```
   sudo AppStreamImageAssistant add-application \
   --name chromium \
   --absolute-app-path /usr/bin/launch-chromium.sh
   ```

7. Create an image, fleet, and stack. For this example, use a fleet name of **`session-context-test-fleet`** and a stack name of **`session-context-test-stack`**.
8. After the fleet is running, you can call create-streaming-url with the `session-context` parameter, as shown in this example.

   ```
   aws appstream create-streaming-url --stack-name session-context-test-stack \
   --fleet-name session-context-test-fleet \
   --user-id username \
   --application-id chromium --session-context "www.amazon.com"
   ```

9. Open the streaming URL in a browser. The batch file launches Chromium and loads `http://www.amazon.com`.

AppStream 2.0 Fleet Types

The fleet type determines when your instances run and how you pay for them. You can specify a fleet type when you create a fleet. You cannot change the fleet type after you create the fleet.

The following are the possible fleet types:

Always-On

Streaming instances run all the time, even when no users are streaming applications and desktops. Streaming instances must be provisioned before a user is able to stream. The number of streaming instances provisioned is managed through auto scaling rules.

When your users choose their application or desktop, they will start streaming instantly. You are charged the running instance fee for all streaming instances, even when no users are streaming.

On-Demand

Streaming instances run only when users are streaming applications and desktops. Streaming instances not yet assigned to users are in a stopped state. Streaming instances must be provisioned before a user is able to stream. The number of streaming instances provisioned is managed through auto scaling rules.

When your users choose their application or desktop, they will start streaming after a 1-2 minute wait. You are charged a lower stopped instance fee for streaming instances that are not yet assigned to users, and the running instance fee for streaming instances that are assigned to users.

Elastic

The pool of streaming instances is managed by AppStream 2.0. When your users select their application or desktop to launch, they will start streaming after the app block has been downloaded and mounted to a streaming instance. For more information about creating app blocks for your Elastic fleets, see App Blocks.

You are charged the running instance fee for Elastic fleet streaming instances only for the duration of the streaming session, in seconds, with a minimum of 15 minutes.

For more information about how fleet types are charged, see Amazon AppStream 2.0 Pricing.

Always-On and On-Demand Fleets

Always-On and On-Demand fleets represent a pool of streaming instances that you manage the capacity of using auto scaling policies. Your users use the streaming instances to stream their applications and desktops. With an Always-On fleet, your user's application launches nearly instantly, and you pay the running instance rate per instance even when a user isn't streaming. With an On-Demand fleet, your user's application launches after a 1-2 minute wait as the streaming instance is started, and you pay a lower cost stopped instance fee for instances not in use, and the running instance fee for instances that are in use.

Applications for Always-On and On-Demand fleet instances are delivered through AppStream 2.0 images that are created by image builders. You can learn more about how to create an image builder, install your applications, and create an image by reading *Images*.

Always-On and On-Demand fleet streaming instances must be provisioned and unassigned to an existing user before a user can stream. You can use fixed or dynamic fleet autoscaling policies to manage the number of instances in your fleet, ensuring that you have sufficient available capacity to meet your user needs while controlling costs. You can learn more about scaling your fleets by reading the section called "Fleet Auto Scaling".

Elastic Fleets

Elastic fleets represent a pool of streaming instances that AppStream 2.0 manages. You do not need to predict concurrency, or create and manage any auto scaling policies for your users to stream their applications and desktops. When your user requests a streaming instance, a streaming instance is assigned from the pool, and made available to them after configuration completes.

Elastic fleets rely on applications that are stored on app blocks. When a user chooses an application from the catalog, the app block is downloaded to the instance, mounted, and then the application launches.

AWS manages the streaming instance provisioning and availability with an Elastic fleet. You need to configure the maximum concurrency you expect when creating and updating the fleet, and ensure that you have sufficient streaming instance limits to meet your user demand.

For more information about creating app blocks for your Elastic fleets, see App Blocks.

AppStream 2.0 Instance Families

Amazon AppStream 2.0 users stream applications from stacks that you create. Each stack is associated with a fleet. When you create a fleet, the instance type that you specify determines the hardware of the host computers used for your fleet. Each instance type offers different compute, memory, and GPU capabilities. Instance types are grouped into *instance families* based on these capabilities. For hardware specifications and pricing information, see AppStream 2.0 Pricing.

When you create a fleet or image builder, you must select an image that is compatible with the instance family on which you intend to run your fleet.

- When launching a new image builder, you are presented with a list of the images in your image registry. Select the appropriate base image.
- When launching a fleet, ensure that the private image you select was created from the appropriate base image.

The following table summarizes the available instance families and provides the base image naming format for each. Select an instance type from an instance family based on the requirements of the applications that you plan to stream on your fleet, and match the base image according to the following table.

> **Note**
>
> If your use case involves real-time audio-video (AV) or other scenarios that require high frame rates and your display performance isn't as expected, consider scaling up to a larger instance size.

Instance Family	Description	Base Image Name
General Purpose	Basic computing resources for running web browsers and most business applications.	AppStream-WinServer-*OperatingSystemVersion* -*MM-DD-YYYY* AppStream-AmazonLinux2-*MM-DD-YYYY*

Instance Family	Description	Base Image Name
Compute Optimized	Optimized for compute-bound applications that benefit from high performance processors.	AppStream-WinServer-*OperatingSystemVersion* -*MM-DD-YYYY* AppStream-AmazonLinux2-*MM-DD-YYYY*
Memory Optimized	Optimized for memory-intensive applications that process large amounts of data. **Note** If you plan to use AppStream 2.0 z1d-based instances, you must provision them from images that were created from AppStream 2.0 base images published on or after June 12, 2018.	AppStream-WinServer-*OperatingSystemVersion* -*MM-DD-YYYY* AppStream-AmazonLinux2-*MM-DD-YYYY*
Graphics Desktop	Uses NVIDIA GRID K520 GPU to support applications that benefit from or require graphics acceleration. This instance family supports DirectX, OpenGL, OpenCL, and CUDA. *This instance family is deprecated and therefore no longer available.*	Graphics-Desktop-Image-Builder-*MM-DD-YYYY*
Graphics Pro	Uses NVIDIA Tesla M60 GPUs and provide a high-performance, workstation-like experience for graphics applications that use DirectX, OpenGL, OpenCL, or CUDA.	AppStream-Graphics-Pro-*OperatingSystemVersion* -*MM-DD-YYYY*

Instance Family	Description	Base Image Name
Graphics Design	Uses AMD FirePro S7150x2 Server GPUs and AMD Multiuser GPU technology to support graphics applications that use DirectX, OpenGL, or OpenCL.	AppStream-Graphics-Design-*Operating SystemVersion* -*MM-DD-YYYY*
Graphics G4dn	Uses NVIDIA T4 GPUs to support graphics intensive applications.	AppStream-Graphics-G4dn-*Operating SystemVersion* -*MM-DD-YYYY*
Graphics G5	Uses NVIDIA A10G GPUs to support graphics-intensive applications such as remote workstations, video rendering , and gaming, to produce high fidelity graphics in real time.	AppStream-Graphics-G5-*OperatingSystemVersion* -07-17-2023

AppStream 2.0 instances have one 200 GB fixed-size volume, which is used for the C drive. Because AppStream 2.0 is non-persistent, each instance's volume is immediately deleted after each user session.

For more information, see the following:

- AppStream 2.0 Base Image and Managed Image Update Release Notes
- Amazon AppStream 2.0 Service Quotas
- AppStream 2.0 Pricing

Create an AppStream 2.0 Fleet and Stack

To stream your applications, Amazon AppStream 2.0 requires an environment that includes a fleet that is associated with a stack, as well as at least one application image. This tutorial describes the steps to set up a fleet and stack, and how to give users access to the stack. If you haven't already done so, we recommend that you try the procedures in Get Started with Amazon AppStream 2.0: Set Up With Sample Applications first.

If you want to create an image to use, see Tutorial: Create a Custom AppStream 2.0 Image by Using the AppStream 2.0 Console.

If you plan to join a fleet to an Active Directory domain, configure your Active Directory domain before completing the following steps. For more information, see Using Active Directory with AppStream 2.0.

Tasks

- Create a Fleet
- Create a Stack
- Provide Access to Users
- Clean Up Resources

Create a Fleet

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

> **Note**
>
> To create an Always-On or On-Demand fleet, you must have an image that has applications installed to create an Always-On or On-Demand fleet that your users can stream from. To create an image, see the section called "Tutorial: Create a Custom Image by Using the Console". To create an Elastic fleet, you must have applications associated to app blocks. To create applications and app blocks for an Elastic fleet, see Applications Manager.

To set up and create a fleet

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. Choose **Get Started** if you are new to the console, or **Fleets** from the left navigation pane. Choose **Create Fleet**.
3. For **Step 1: Select fleet type**, review the details of the fleet types, choose the type of fleet to create based on your use case, and select **Next**.

> **Note**
>
> The fleet type determines its immediate availability and how you pay for it. For more information, see Fleet Type.

4. For **Step 2: Configure fleet**, enter the following **details**:

   - For **Name**, enter a unique name identifier for the fleet. Special characters aren't allowed.
   - For **Display Name**, enter a name to display for the fleet (maximum of 100 characters). Special characters aren't allowed.
   - For **Description**, enter a description for the fleet (maximum of 256 characters).
   - For **Choose instance type**, choose the instance type that meets the performance requirements of your applications. All streaming instances in your fleet launch with the instance type that you select. For more information, see AppStream 2.0 Instance Families.
   - For Elastic fleets, for **Choose platform type**, choose the operating system that matches the requirements of your users' applications.
   - For **Maximum session duration in minutes**, choose the maximum amount of time that a streaming session can remain active. If users are still connected to a streaming instance five minutes before this limit is reached, they are prompted to save any open documents before being disconnected. After this time elapses, the instance is terminated and replaced by a new instance. The maximum session duration that you can set in the AppStream 2.0 console is 5760 minutes (96 hours). The maximum session duration that you can set using the AppStream 2.0 API and CLI is 432000 seconds (120 hours).
   - For **Disconnect timeout in minutes**, choose the amount of time that a streaming session remains 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 their previous session. Otherwise, they are connected to a new session with a new streaming instance. If you associate a stack with a fleet for which a redirect URL is specified, after users' streaming sessions end, the users are redirected to that URL.

     If a user ends the session by choosing **End Session** or **Logout** on the AppStream 2.0 toolbar, the disconnect timeout does not apply. Instead, the user is prompted to save any open documents, and then immediately disconnected from the streaming instance. The instance the user was using is then terminated.

- For **Idle disconnect timeout in minutes**, choose the amount of time that users can be idle (inactive) before they are disconnected from their streaming session and the **Disconnect timeout in minutes** time interval begins. Users are notified before they are disconnected due to inactivity. If they try to reconnect to the streaming session before the time interval specified in **Disconnect timeout in minutes** has elapsed, they are connected to their previous session. Otherwise, they are connected to a new session with a new streaming instance. Setting this value to 0 disables it. When this value is disabled, users are not disconnected due to inactivity.

  > **Note**
  >
  > Users are considered idle when they stop providing keyboard or mouse input during their streaming session. For domain-joined fleets, the countdown for the idle disconnect timeout doesn't begin until users log in with their Active Directory domain password or with a smart card. File uploads and downloads, audio in, audio out, and pixels changing do not qualify as user activity. If users continue to be idle after the time interval in **Idle disconnect timeout in minutes** elapses, they are disconnected.

- For Elastic fleets, for **Max concurrent sessions**, specify the maximum number of concurrent sessions this fleet should have.

  > **Note**
  >
  > If you get an error message that says “The maximum number of concurrent sessions for your account was exceeded," you can submit a limit increase, through the Service Quotas console at https://console.aws.amazon.com/servicequotas/. For more information, see Requesting a quota increase in the *Service Quotas User Guide*.

- **Multiple user sessions** — Choose this option if you want to provision multiple user sessions on a single instance. By default, every unique user session is served by an instance (single-session).

  > **Note**
  >
  > Multi-session is available only on Always-on and On-demand fleets powered by a Windows operations system. Multi-session is not available on Elastic fleets or the Linux operating system.

Only base images and managed image updates released on or after May 15, 2023 support multi-session fleets. For more details, see the section called "Base Image and Managed Image Update Release Notes".

- **Maximum sessions per instance** — Maximum number of user sessions on an instance. You must choose this value based on your end users' application performance needs. You can also adjust the maximum sessions per instance for a fleet after it is provisioned. In that case, the existing user sessions and instances will not be impacted, but the fleet will become consistent with the new value of maximum sessions per instance. The value must be between 2 and 50. Before setting this value for your fleet, see the section called "Multi-Session Recommendations".
- For Always-On and On-Demand fleets, for **Minimum capacity**, choose a minimum number of instances (for single-session fleets) or user sessions (for multi-session fleets) for your fleet based on the minimum number of expected concurrent users.
- For Always-On and On-Demand fleets, for **Maximum capacity**, choose a maximum number of instances (for single-session fleets) or user sessions (for multi-session fleets) for your fleet based on the maximum number of expected concurrent users.

> **Note**
>
> For multi-session, you must specify the capacity based on the number of user sessions. The service will calculate the required number of instances to be launched, based on your fleet configuration and the value of maximum sessions per instance.

- For **Stream view**, choose the AppStream 2.0 view that is displayed to your users during their streaming sessions. Choose **Application** to display only the windows of applications opened by users. Choose **Desktop** to display the standard desktop that is provided by the operating system.

> **Note**
>
> By default, AppStream 2.0 displays only the windows of applications opened by users during their streaming sessions. To enable **Desktop** view for your users, configure your fleet to use an AppStream 2.0 image that uses a version of the AppStream 2.0 agent released on or after February 19, 2020.

- For **Scaling details (Advanced)**, specify the scaling policies that AppStream 2.0 uses to increase and decrease the capacity of your fleet. Note that the size of your fleet is limited by the minimum and maximum capacity that you specified. For more information, see Fleet Auto Scaling for Amazon AppStream 2.0.
- For **IAM role (Advanced)**, when you apply an IAM role from your account to an AppStream 2.0 fleet instance, you can make AWS API requests from the fleet instance without manually managing AWS credentials. To apply an IAM role, do either of the following:
  - To use an existing IAM role in your AWS account, choose the role that you want to use from the **IAM role** list. The role must be accessible from the fleet instance. For more information, see Configuring an Existing IAM Role to Use With AppStream 2.0 Streaming Instances.
  - To create a new IAM role, choose **Create new IAM role** and follow the steps in How to Create an IAM Role to Use With AppStream 2.0 Streaming Instances.
- For Elastic fleets, for **USB Redirection (advanced)**, you can specify up to 10 strings that specify what types of USB devices that are attached to the local device can be redirected into the streaming session when using the Windows native client. For more information, see the section called "Qualify USB Devices for Use with Streaming Applications".

5. Choose **Next**.
6. If you chose to create an Always-On or On-Demand fleet, for **Step 3: Choose an Image**, choose an image that meets your needs and then choose **Next**.
7. If you chose to create an Elastic fleet, for **Step 3: Assign applications**, choose the applications that users can launch from this fleet.
8. For **Step 4: Configure Network**, do the following:
   - To add internet access for fleet instances in a VPC with a public subnet, choose **Default Internet Access**. If you are providing internet access by using a NAT gateway, leave **Default Internet Access** unselected. For more information, see Internet Access.

> **Note**
>
> Your VPC must provide access to Amazon Simple Storage Service (S3) if you enable features that rely on saving to an S3 bucket. For more information, see the section called "Amazon S3 VPC Endpoints".

- For **VPC** and **Subnet 1**, choose a VPC and at least one subnet that has access to the network resources that your application needs. For increased fault tolerance, we recommend that you choose two subnets in different Availability Zones. For more information, see Configure a VPC with Private Subnets and a NAT Gateway.

  > **Note**
  >
  > Elastic fleets require that you specify at least two subnets that are in different availability zones.

  If you don't have your own VPC and subnet, you can use the default VPC or create your own. To create your own, choose the **Create a new VPC** and **Create new subnet** links to create them. Choosing these links opens the Amazon VPC console. After you create your VPC and subnets, return to the AppStream 2.0 console and choose the refresh icon to the left of the **Create a new VPC** and **Create new subnet** links to display them in the list. For more information, see Configure a VPC for AppStream 2.0.

- For **Security group(s)**, choose up to five security groups to associate with this fleet. If you don't have your own security group and you don't want to use the default security group, choose the **Create new security group** link to create one. After you create your subnets in the Amazon VPC console, return to the AppStream 2.0 console and choose the refresh icon to the left of the **Create new security group** link to display them in the list. For more information, see Security Groups in Amazon AppStream 2.0.
- For Always-On and On-Demand fleets, for **Active Directory Domain (Optional)**, choose the Active Directory and organizational unit (OU) for your streaming instance computer objects. Ensure that the network access settings you selected enable DNS resolvability and communication with your directory. For more information, see Using Active Directory with AppStream 2.0.

9. Choose **Next**.
10. For **Step 5: Review**, confirm the details for the fleet. To change the configuration for any section, choose **Edit** and make the needed changes. After you finish reviewing the configuration details, choose **Create**.
11. In the pricing acknowledgement dialog box, select the acknowledgement check box, and choose **Create**.

> **Note**
>
> If an error message notifies you that you don't have sufficient limits (quotas) to create the fleet, submit a limit increase request through the Service Quotas console at https://console.aws.amazon.com/servicequotas/. For more information, see Requesting a quota increase in the *Service Quotas User Guide*.

12. While your fleet is being created, the status of your fleets displays as **Starting** in the **Fleets** list. Choose the **Refresh** icon periodically to update the fleet status until the status is **Running**. You cannot associate the fleet with a stack and use it for streaming sessions until the status of the fleet is **Running**.

Create a Stack

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

> **Note**
>
> You can enable Google Drive, OneDrive, and Application Settings Persistence only for stacks associated with a Windows fleet. Before you associate an existing stack with a Linux fleet, please make sure these settings are disabled.

To set up and create a stack

1. In the left navigation pane, choose **Stacks**, and then choose **Create Stack**.
2. For **Step 1: Stack Details**, Under **Stack details**, enter a unique name identifier for the stack. Optionally, you can do the following:
   - **Display name** — Enter a name to display for the stack (maximum of 100 characters).
   - **Description**— Enter a description for the stack (maximum of 256 characters).
   - **Redirect URL** — Specify a URL to which users are redirected after their streaming sessions end.
   - **Feedback URL** — Specify a URL to which users are redirected after they click the **Send Feedback** link to submit feedback about their application streaming experience. If you do not specify a URL, this link is not displayed.

- **Fleet** — Select an existing fleet or create a new one to associate with your stack.
- **Streaming Protocol Preference** — Specify the streaming protocol you'd like your stack to prefer, UDP or TCP. UDP is currently only supported in the Windows native client. For more information, see System Requirements and Feature Support (AppStream 2.0 Client).
- **Tags** — Choose **Add Tag**, and type the key and value for the tag. To add more tags, repeat this step. For more information, see Tagging Your Amazon AppStream 2.0 Resources.
- **VPC Endpoints (Advanced)** — You can create a private link, which is an interface VPC endpoint (interface endpoint), in your virtual private cloud (VPC). To start creating the interface endpoint, select **Create VPC Endpoint**. Selecting this link opens the VPC console. To finish creating the endpoint, follow steps 3 through 6 in *To create an interface endpoint*, in Creating and Streaming from Interface VPC Endpoints.

  After you create the interface endpoint, you can use it to keep streaming traffic within your VPC.
- **Embed AppStream 2.0 (Optional)** — To embed an AppStream 2.0 streaming session in a webpage, specify the domain to host the embedded streaming session. Embedded streaming sessions are only supported over HTTPS [TCP port 443].

  > **Note**
  >
  > You must meet prerequisites and perform additional steps to configure embedded AppStream 2.0 streaming sessions. For more information, see Embed AppStream 2.0 Streaming Sessions.

- **Tags** — Choose **Add Tag**, and enter the key and value for the tag. To add more tags, repeat this step. For more information, see Tagging Your Amazon AppStream 2.0 Resources.
- **VPC Endpoints (Advanced)** — You can create a private link, which is an interface VPC endpoint (interface endpoint), in your virtual private cloud (VPC). To start creating the interface endpoint, select **Create VPC Endpoint**. Selecting this link opens the VPC console. To finish creating the endpoint, follow steps 3 through 6 in *To create an interface endpoint*, in Creating and Streaming from Interface VPC Endpoints.

  After you create the interface endpoint, you can use it to keep streaming traffic within your VPC.

- **Embed AppStream 2.0 (Optional)** — To embed an AppStream 2.0 streaming session in a webpage, specify the domain to host the embedded streaming session. Embedded streaming sessions are only supported over HTTPS [TCP port 443].

> **Note**
>
> You must meet prerequisites and perform additional steps to configure embedded AppStream 2.0 streaming sessions. For more information, see Embed AppStream 2.0 Streaming Sessions.

3. Choose **Next.**
4. For **Step 2: Enable Storage**, you can provide persistent storage for your users by choosing one or more of the following:

- **Home Folders** — Users can save their files to their home folder and access existing files in their home folder during application streaming sessions. For information about requirements for enabling home folders, see Enable Home Folders for Your AppStream 2.0 Users.
- **Google Drive for Google Workspace** — Users can link their Google Drive for Google Workspace account to AppStream 2.0. During application streaming sessions, they can sign in to their Google Drive account, save files to Google Drive, and access their existing files in Google Drive. You can enable Google Drive for accounts in Google Workspace domains only, not for personal Gmail accounts.

> **Note**
>
> Enabling Google Drive is not supported for Linux-based stacks or stacks associated with multi-session fleets.

> **Note**
>
> After you select **Enable Google Drive**, type the name of at least one organizational domain that is associated with your Google Workspace account. Access to Google Drive during application streaming sessions is limited to users that are in the domains that you specify. You can specify up to 10 domains. For more information

about requirements for enabling Google Drive, see Enable Google Drive for Your AppStream 2.0 Users.

- **OneDrive for Business** — Users can link their OneDrive for Business account to AppStream 2.0. During application streaming sessions, they can sign in to their OneDrive account, save files to OneDrive, and access their existing files in OneDrive. You can enable OneDrive for accounts in OneDrive domains only, not for personal accounts.

> **Note**
>
> Enabling OneDrive is not supported for Linux-based stacks or stacks associated with multi-session fleets..

> **Note**
>
> After you select **Enable OneDrive**, enter the name of least one organizational domain that is associated with your OneDrive account. Access to OneDrive during application streaming sessions is limited to users that are in the domains that you specify. You can specify up to 10 domains. For more information about requirements for enabling OneDrive, see Enable OneDrive for Your AppStream 2.0 Users.

5. Choose **Next**.
6. For **Step 3: User Settings**, configure the following settings. When you're done, choose **Review**.

   Clipboard, file transfer, print to local device, and authentication permissions options:

> **Note**
>
> **Print to local device** and **Smart card sign in for Active Directory** options are currently not available for multi-session fleets.

- **Clipboard** — By default, users can copy and paste data between their local device and streaming applications. You can limit Clipboard options so that users can paste data to their remote streaming session only or copy data to their local device only. You can also disable Clipboard options entirely. Users can still copy and paste between applications in

their streaming session. You can choose **Copy to local device character limit** or **Paste to remote session character limit** or both to limit the amount of data that users can copy or paste when using the clipboard, either in or out of their AppStream 2.0 streaming session. The value can be between 1 and 20,971,520 (20 MB), and defaults to the maximum value when unspecified.

- **File transfer** — By default, users can upload and download files between their local device and streaming session. You can limit file transfer options so that users can upload files to their streaming session only or download files to their local device only. You can also disable file transfer entirely.

  > **Important**
  >
  > If your users require AppStream 2.0 file system redirection to access local drives and folders during their streaming sessions, you must enable both file upload and download. To use file system redirection, your users must have AppStream 2.0 client version 1.0.480 or later installed. For more information, see Enable File System Redirection for Your AppStream 2.0 Users.

- **Print to local device** — By default, users can print to their local device from within a streaming application. When they choose **Print** in the application, they can download a .pdf file that they can print to a local printer. You can disable this option to prevent users from printing to a local device.
- **Password sign in for Active Directory** — Users can enter their Active Directory domain password to sign in to an AppStream 2.0 streaming instance that is joined to an Active Directory domain.

  You can also enable **Smart card sign in for Active Directory**. At least one authentication must be enabled.
- **Smart card sign in for Active Directory** — Users can use a smart card reader and smart card connected to their local computer to sign in to an AppStream 2.0 streaming instance that is joined to an Active Directory domain.

  You can also enable **Password sign in for Active Directory**. At least one authentication method must be enabled.

> **Note**
>
> **Clipboard, file transfer, and print to local device settings** — These settings control only whether users can use AppStream 2.0 data transfer features. If your image provides access to a browser, network printer, or other remote resource, your users might be able to transfer data to or from their streaming session in other ways.
> **Authentication settings** — These settings control only the authentication method that can be used for Windows sign in to an AppStream 2.0 streaming instance (fleet or image builder). They do not control the authentication method that can be used for in-session authentication, after a user signs in to a streaming instance. For information about configuration requirements for using smart cards for Windows sign in and in-session authentication, see Smart Cards. These settings are not supported for Linux-based stacks.

Application settings persistence options:

- **Enable Application Settings Persistence** — Users' application customizations and Windows settings are automatically saved after each streaming session and applied during the next session. These settings are saved to an Amazon Simple Storage Service (Amazon S3) bucket in your account, within the AWS Region in which application settings persistence is enabled.
- **Settings Group** — The settings group determines which saved application settings are used for a streaming session from this stack. If the same settings group is applied to another stack, both stacks use the same application settings. By default, the settings group value is the name of the stack.

> **Note**
>
> For information about requirements for enabling and administering application settings persistence, see Enable Application Settings Persistence for Your AppStream 2.0 Users.

7. For **Step 4: Review**, confirm the details for the stack. To change the configuration for any section, choose **Edit** and make the needed changes. After you finish reviewing the configuration details, choose **Create**.

After the service sets up resources, the **Stacks** page appears. The status of your new stack appears as **Active** when it is ready to use.

Provide Access to Users

After you create a stack with an associated fleet, you can provide access to users through the AppStream 2.0 user pool, SAML 2.0 [single sign-on (SSO)], or the AppStream 2.0 API. For more information, see User Pool Administration and Amazon AppStream 2.0 Integration with SAML 2.0.

> **Note**
>
> Users in the AppStream 2.0 user pool can't be assigned to stacks with fleets that are joined to an Active Directory domain.

After you provide your users with access to AppStream 2.0, they can start AppStream 2.0 streaming sessions by using a web browser or by using the AppStream 2.0 client application for a supported device. If you provide access to users through the AppStream 2.0 user pool, they must use a web browser for streaming sessions. If you use SAML 2.0 or the AppStream 2.0 API, you can make the AppStream 2.0 client available to them. The AppStream 2.0 client is a native application that is designed for users who require additional functionality during their AppStream 2.0 streaming sessions. For more information, see Provide Access Through the AppStream 2.0 Client for Windows.

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.

Note that you cannot delete a stack with an associated fleet.

To clean up your resources

1. In the navigation pane, choose **Stacks**.
2. Select the stack and choose **Actions**, **Disassociate Fleet**. In the confirmation dialog box, choose **Disassociate.**
3. In the navigation pane, choose **Fleets**.
4. Select the fleet that you want to stop, choose **Actions**, and then choose **Stop**. It takes about 5 minutes to stop a fleet.

5. When the status of the fleet is **Stopped**, choose **Actions**, **Delete**.
6. In the navigation pane, choose **Stacks**.
7. Select the stack and choose **Actions**, **Delete**.

Customize an AppStream 2.0 Fleet to Optimize Your Users' Application Streaming Experience

By customizing AppStream 2.0 fleet instances, you can define specific aspects of your AppStream 2.0 environment to optimize your users' application streaming experience. For example, you can persist environment variables to dynamically pass settings across applications and set default file associations that are applied to all of your users. At a high level, customizing a fleet instance includes the following tasks:

- Connecting to an image builder and customizing it as needed.
- On the image builder, using Image Assistant to create a new image that includes your customizations.
- Creating a new fleet instance or modifying an existing one. When you configure the fleet instance, select the new customized image that you created.
- Creating a new stack or modifying an existing one and associating it with your fleet instance.

> **Note**
>
> For certain fleet customizations, in Active Directory environments, you might need to use the Group Policy Management Console (GPMC) to update Group Policy object (GPO) settings on a domain-joined computer.

Contents

- Persist Environment Variables
- Set Default File Associations for Your Users
- Disable Internet Explorer Enhanced Security Configuration
- Change the Default Internet Explorer Home Page for Users' Streaming Sessions
- User and Instance Metadata for AppStream 2.0 Fleets

Persist Environment Variables

Environment variables enable you to dynamically pass settings across applications. For example, many engineering applications rely on environment variables to specify the IP address or host name of a license server to locate and check out a license from that server.

Follow the steps in these procedures to make environment variables available across your fleet instances.

> **Note**
>
> The following instructions apply to Windows fleets only.

Contents

- Change System Environment Variables
- Change User Environment Variables
- Create an Environment Variable That is Limited in Scope

> **Note**
>
> If you are using Active Directory and Group Policy with AppStream 2.0, keep in mind that streaming instances must be joined to an Active Directory domain to use Group Policy for environment variables. For information about how to configure the Group Policy **Environment Variable** preference item, see Configure an Environment Variable Item in the Microsoft documentation.

Change System Environment Variables

Follow these steps to change system environment variables across your fleet instances.

To change system environment variables on an image builder

This procedure applies only to system environment variables, not user environment variables. To change user environment variables that persist across your fleet instances, perform the steps in the next procedure.

1. Connect to the image builder on which to change system environment variables and sign in with an account that has local administrator permissions. To do so, do either of the following:

   - Use the AppStream 2.0 console (for web connections only)
   - Create a streaming URL (for web or AppStream 2.0 client connections)

   > **Note**
   >
   > If the image builder that you want to connect to is joined to an Active Directory domain and your organization requires smart card sign in, you must create a streaming URL and use the AppStream 2.0 client for the connection. For information about smart card sign in, see Smart Cards.

2. Choose the Windows **Start** button, open the context (right-click) menu for **Computer**, and then choose **Properties**.
3. In the navigation pane, choose **Advanced system settings**.
4. In **System variables**, change the environment variables that you want to persist across your fleet instances, and then choose **OK**.
5. On the image builder desktop, open Image Assistant.
6. Follow the necessary steps in Image Assistant to finish creating your image. For more information, see Tutorial: Create a Custom AppStream 2.0 Image by Using the AppStream 2.0 Console.

   The changes to the system environment variables persist across your fleet instances and are available to streaming sessions launched from those instances.

   > **Note**
   >
   > Setting AWS CLI credentials as system environment variables might prevent AppStream 2.0 from creating the image.

Change User Environment Variables

Follow these steps to change user environment variables across your fleet instances.

To change user environment variables

1. Connect to the image builder on which to change system environment variables and sign in as a **Template User**. To do so, do either of the following:

   - Use the AppStream 2.0 console (for web connections only)
   - Create a streaming URL (for web or AppStream 2.0 client connections)

   > **Note**
   >
   > If the image builder that you want to connect to is joined to an Active Directory domain and your organization requires smart card sign in, you must create a streaming URL and use the AppStream 2.0 client for the connection. For information about smart card sign in, see Smart Cards.

   Template User lets you create default application and Windows settings for your users. For more information, see "Creating Default Application and Windows Settings for Your AppStream 2.0 Users" in Default Application and Windows Settings and Application Launch Performance.

2. On the image builder, choose the Windows **Start** button, **Control Panel**, **User Accounts**.
3. Choose **User Accounts** again. In the left navigation pane, choose **Change my environment variables.**
4. Under **User environment variables** for **DefaultProfileUser**, set or create the user environment variables as needed, then choose **OK**.
5. This disconnects your current session and opens the login menu. Log in to the image builder by doing either of the following:

   - If your image builder is not joined to an Active Directory domain, on the **Local User** tab, choose **Administrator**.
   - If your image builder is joined to an Active Directory domain, choose the **Directory User** tab, and log in as a domain user who has local administrator permissions on the image builder.

6. On the image builder desktop, open Image Assistant.
7. Follow the necessary steps in Image Assistant to finish creating your image. For more information, see Tutorial: Create a Custom AppStream 2.0 Image by Using the AppStream 2.0 Console.

Create an Environment Variable That is Limited in Scope

Follow these steps to create an environment variable that is limited in scope to the processes that are spawned off the script. This approach is useful when you need to use the same environment variable name with different values for different applications. For example, if you have two different applications that use the environment variable "LIC_SERVER", but each application has a different value for "LIC_SERVER".

To create an environment variable that is limited in scope

1. Connect to the image builder on which to create an environment variable that is limited in scope and sign in with an account that has local administrator permissions. To do so, do either of the following:

   - Use the AppStream 2.0 console (for web connections only)
   - Create a streaming URL (for web or AppStream 2.0 client connections)

   > **Note**
   >
   > If the image builder that you want to connect to is joined to an Active Directory domain and your organization requires smart card sign in, you must create a streaming URL and use the AppStream 2.0 client for the connection. For information about smart card sign in, see Smart Cards.

2. Create a child folder of C:\ drive for the script (for example, C:\Scripts).
3. Open Notepad to create the new script, and enter the following lines:

   ```
   set variable=value
   ```

   ```
   start " " "C:\path\to\application.exe"
   ```

   Where:

   variable is the variable name to be used

   value is the value for the given variable name

> **Note**
>
> If the application path includes spaces, the entire string must be encapsulated within quotation marks. For example:
> `start " " "C:\Program Files\application.exe"`

4. Choose **File**, **Save**. Name the file and save it with the .bat extension to C:\Scripts. For example, name the file LaunchApp.bat.
5. If needed, repeat steps 4 and 5 to create a script for each additional application that requires its own environment variable and values.
6. On the image builder desktop, start Image Assistant.
7. Choose **Add App**, navigate to C:\Scripts, and select one of the scripts that you created in step 5. Choose **Open**.
8. In the **App Launch Settings** dialog box, keep or change the settings as needed. When you're done, choose **Save**.
9. If you created multiple scripts, repeat steps 8 and 9 for each script.
10. Follow the necessary steps in Image Assistant to finish creating your image. For more information, see Tutorial: Create a Custom AppStream 2.0 Image by Using the AppStream 2.0 Console.

    The environment variable and specific value are now available for processes that are run from the script. Other processes cannot access this variable and value.

Set Default File Associations for Your Users

The associations for application file extensions are set on a per-user basis and so are not automatically applied to all users who launch AppStream 2.0 streaming sessions. For example, if you set Adobe Reader as the default application for .pdf files on your image builder, this change is not applied to your users.

> **Note**
>
> The following steps apply to Windows fleets only.

> **Note**
>
> The following steps must be performed on an image builder that is joined to an Active Directory domain. In addition, your fleet must be joined to an Active Directory domain. Otherwise, the default file associations that you set are not applied.

To set default file associations for your users

1. Connect to the image builder on which to set default file associations and sign in with a domain account that has local administrator permissions on the image builder. To do so, do either of the following:

   - Use the AppStream 2.0 console (for web connections only)
   - Create a streaming URL (for web or AppStream 2.0 client connections)

   > **Note**
   >
   > If your organization requires smart card sign in, you must create a streaming URL and use the AppStream 2.0 client for the connection. For information about smart card sign in, see Smart Cards.

2. Set default file associations as needed.
3. Open the Windows command prompt as an administrator.
4. At the command prompt, type the following command to export the image builder file associations as an XML file, and then press ENTER:

   ```
   dism.exe /online /export-DefaultAppAssociations:c:
   \default_associations.xml
   ```

   If you receive an error message stating that you cannot service a running 64-bit operating system with a 32-bit version of DISM, close the command prompt window. Open File Explorer, browse to C:\Windows\System32, right-click cmd.exe, choose **Run as Administrator**, and run the command again.

5. You can use either Local Group Policy Editor or the GPMC to set a default associations configuration file:

   - Local Group Policy Editor:

On your image builder, open the command prompt as an administrator, type `gpedit.msc`, and then press ENTER.

In the console tree, under **Computer Configuration**, expand **Administrative Templates**, **Windows Components**, and then choose **File Explorer**.

- GPMC:

  In your directory or on a domain controller, open the command prompt as an administrator, type `gpmc.msc`, and then press ENTER.

  In the left console tree, select the OU in which you want to create a new GPO, or use an existing GPO, and then do either of the following:

  - Create a new GPO by opening the context (right-click) menu and choosing **Create a GPO in this domain, Link it here**. For **Name**, provide a descriptive name for this GPO.
  - Select an existing GPO.

  Open the context menu for the GPO, and choose **Edit**.

  Under **User Configuration**, expand **Policies**, **Administrative Templates**, **Windows Components**, and then choose **File Explorer**.

6. Double-click **Set a default associations configuration file**.
7. In the **Set a default associations configuration file properties** dialog box, choose **Enabled**, and do one of the following:

   - If you are using Local Group Policy Editor, enter this path: `c:\default_associations.xml`.
   - If you are using the GPMC, enter a network path. For example, `\\networkshare\default_associations.xml`.
8. Choose **Apply**, **OK**.
9. Close Local Group Policy Editor or the GPMC.
10. On the image builder desktop, open Image Assistant.
11. Follow the necessary steps in Image Assistant to finish creating your image. For more information, see Tutorial: Create a Custom AppStream 2.0 Image by Using the AppStream 2.0 Console.

The file associations that you configured are applied to the fleet instances and user streaming sessions that are launched from those instances.

Disable Internet Explorer Enhanced Security Configuration

Internet Explorer Enhanced Security Configuration (ESC) places servers and Internet Explorer in a configuration that limits exposure to the internet. However, this configuration can impact the AppStream 2.0 end user experience. Users who are connected to AppStream 2.0 streaming sessions may find that websites do not display or perform as expected when:

- Internet Explorer ESC is enabled on fleet instances from which users' streaming sessions are launched
- Users run Internet Explorer during their streaming sessions
- Applications use Internet Explorer to load data

> **Note**
>
> The following steps apply to Windows fleets only.

To disable Internet Explorer Enhanced Security Configuration

1. Connect to the image builder on which to disable Internet Explorer ESC and sign in with an account that has local administrator permissions. To do so, do either of the following:

   - Use the AppStream 2.0 console (for web connections only)
   - Create a streaming URL (for web or AppStream 2.0 client connections)

   > **Note**
   >
   > If the image builder that you want to connect to is joined to an Active Directory domain and your organization requires smart card sign in, you must create a streaming URL and use the AppStream 2.0 client for the connection. For information about smart card sign in, see Smart Cards.

2. On the image builder, disable Internet Explorer ESC by doing the following:

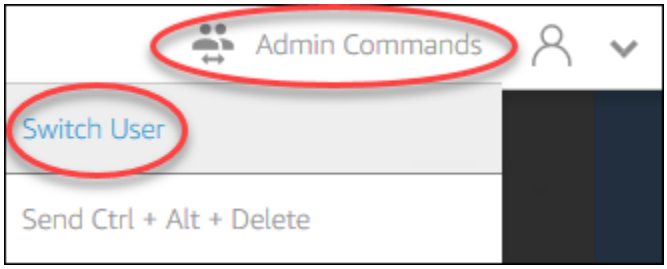
a. Open Server Manager. Choose the Windows **Start** button, and then choose **Server Manager**.

b. In the left navigation pane, choose **Local Server**.

c. In the right properties pane, choose the **On** link next to IE Enhanced Security Configuration.

d. In the **Internet Explorer Enhanced Configuration** dialog box, choose the **Off** option under **Administrators** and **Users**, then choose **OK**.

3. In the upper right area of the image builder desktop, choose **Admin Commands**, **Switch User**.



4. This disconnects your current session and opens the login menu. Log in to the image builder by doing either of the following:

   - If your image builder is not joined to an Active Directory domain, on the **Local User** tab, choose **Template User**.
   - If your image builder is joined to an Active Directory domain, choose the **Directory User** tab, and log in as a domain user who does not have local administrator permissions on the image builder.

5. Open Internet Explorer and reset your settings by doing the following:

   a. In the upper right area of the Internet Explorer browser window, choose the **Tools** icon, then choose **Internet options**.

   b. Choose the **Advanced** tab, then choose **Reset**.

   c. When prompted to confirm your choice, choose **Reset** again.

   d. When the **Reset Internet Explorer Settings** message displays, choose **Close**.

6. Reboot image builder.

7. Choose **Admin Commands**, **Switch User**, and then do either of the following:

   - If your image builder is not joined to an Active Directory domain, on the **Local User** tab, choose **Administrator**.

- If your image builder is joined to an Active Directory domain, choose the **Directory User** tab, and log in with the same domain account that you used in step 4.

8. On the image builder desktop, open Image Assistant.
9. In **Step 2. Configure Apps**, choose **Save settings**.
10. Follow the necessary steps in Image Assistant to finish creating your image. For more information, see Tutorial: Create a Custom AppStream 2.0 Image by Using the AppStream 2.0 Console.

Change the Default Internet Explorer Home Page for Users' Streaming Sessions

You can use Group Policy to change the default Internet Explorer home page for users' streaming sessions. Alternatively, if you do not have Group Policy in your environment or prefer not to use Group Policy, you can use the AppStream 2.0 Template User account instead.

> **Note**
>
> The following steps apply to Windows fleets only.

Contents

- Use Group Policy to Change the Default Internet Explorer Home Page
- Use the AppStream 2.0 Template User Account to Change the Default Internet Explorer Home Page

Use Group Policy to Change the Default Internet Explorer Home Page

In Active Directory environments, you use the Group Policy Management (GPMC) MMC-snap-in to set a default home page that users can't change. If Active Directory is not in your environment, you can use Local Group Policy Editor to perform this task. To set a home page that users can change, you must use the GPMC.

To use the GPMC, do the following first:

- Obtain access to a computer or an EC2 instance that is joined to your domain.

- Install the GPMC. For more information, see Installing or Removing Remote Server Administration Tools for Windows 7 in the Microsoft documentation.
- Log in as a domain user with permissions to create GPOs. Link GPOs to the appropriate organizational units (OUs).

To change the default Internet Explorer home page by using a Group Policy administrative template

You can use a Group Policy administrative template to set a default home page that users can't change. For more information about administrative templates, see Edit Administrative Template Policy Settings in the Microsoft documentation.

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. If you are not using Active Directory in your environment, open Local Group Policy Editor. If you are using Active Directory, open the GPMC. Locate the **Scripts (Logon\Logoff)** policy setting:

   - Local Group Policy Editor:

     On your image builder, open the command prompt as an administrator, type `gpedit.msc`, and then press ENTER.

     Under **User Configuration**, expand **Administrative Templates**, **Windows Components**, and then choose **Internet Explorer**.

   - GPMC:

     In your directory or on a domain controller, open the command prompt as an administrator, type `gpmc.msc`, and then press ENTER.

     In the left console tree, select the OU in which you want to create a new GPO, or use an existing GPO, and then do either of the following: :

     - Create a new GPO by opening the context (right-click) menu and choosing **Create a GPO in this domain, Link it here**. For **Name**, provide a descriptive name for this GPO.
     - Select an existing GPO.

     Open the context menu for the GPO, and choose **Edit**.

Under **User Configuration**, expand **Policies**, **Administrative Templates**, **Windows Components**, and then choose **Internet Explorer**.

3. Double-click **Disable changing home page settings**, choose **Enabled**, and in **Home Page**, enter a URL.
4. Choose **Apply**, **OK**.
5. Close Local Group Policy Editor or the GPMC.

To change the default Internet Explorer home page by using Group Policy preferences

You can use Group Policy preferences to set a default home page that users can change. For more information about working with Group Policy preferences, see Configure a Registry Item and Group Policy Preferences Getting Started Guide in the Microsoft documentation.

1. In your directory or on a domain controller, open the command prompt as an administrator, type `gpmc.msc`, and then press ENTER.
2. In the left console tree, select the OU in which you want to create a new GPO, or use an existing GPO, and then do either of the following:
   - Create a new GPO by opening the context (right-click) menu and choosing **Create a GPO in this domain, Link it here**. For **Name**, provide a descriptive name for this GPO.
   - Select an existing GPO.
3. Open the context menu for the GPO, and choose **Edit**.
4. Under **User Configuration**, expand **Preferences**, and then choose **Windows Settings**.
5. Open the context (right-click) menu for **Registry** and choose **New**, **Registry Item**.
6. In the **New Registry Properties** dialog box, specify the following registry settings for Group Policy to configure:
   - For **Action**, choose **Update**.
   - For **Hive**, choose **HKEY_CURRENT_USER**.
   - For **Key Path**, browse to and select HKEY_CURRENT_USER\SOFWARE\Microsoft\Internet Explorer\Main.
   - For **Value Name**, enter **Start Page**.
   - For **Value Data**, enter your home page URL.
7. On the **Common** tab, choose **Apply Once**, **Do not Re-Apply**.

> **Note**
>
> To enable your users to choose the **Use Default** button in their Internet Explorer browser settings and reset their default home page to your company home page, you can also set a value for Default_Page_URL without choosing **Apply Once** and **Do not Re-Apply**.

8. Choose **OK** and close the GPMC.

Use the AppStream 2.0 Template User Account to Change the Default Internet Explorer Home Page

Follow these steps to use the **Template User** account to change the default Internet Explorer home page.

To change the default Internet Explorer Home page by using the Template User account

1. Connect to the image builder on which to change the default Internet Explorer home page and sign in with the **Template User** account. To do so, do either of the following:

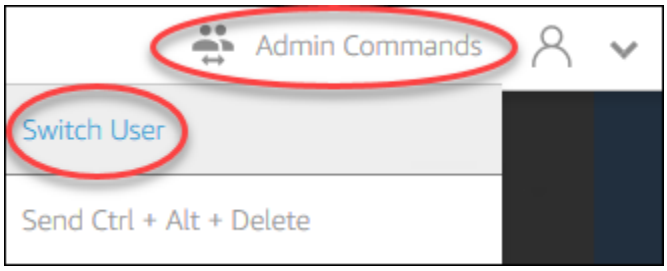
   - Use the AppStream 2.0 console (for web connections only)
   - Create a streaming URL (for web or AppStream 2.0 client connections)

   > **Note**
   >
   > If the image builder that you want to connect to is joined to an Active Directory domain and your organization requires smart card sign in, you must create a streaming URL and use the AppStream 2.0 client for the connection. For information about smart card sign in, see Smart Cards.

   Template User lets you create default application and Windows settings for your users. For more information, see "Creating Default Application and Windows Settings for Your AppStream 2.0 Users" in Default Application and Windows Settings and Application Launch Performance.

2. Open Internet Explorer and complete the necessary steps to change the default home page.

3. In the upper right area of the image builder desktop, choose **Admin Commands**, **Switch User**.



4. This disconnects your current session and opens the login menu. Log in to the image builder by doing either of the following:
   - If your image builder is not joined to an Active Directory domain, on the **Local User** tab, choose **Administrator**.
   - If your image builder is joined to an Active Directory domain, choose the **Directory User** tab, and log in as a domain user who has local administrator permissions on the image builder.
5. On the image builder desktop, open Image Assistant.
6. Follow the necessary steps in Image Assistant to finish creating your image. For more information, see Tutorial: Create a Custom AppStream 2.0 Image by Using the AppStream 2.0 Console.

User and Instance Metadata for AppStream 2.0 Fleets

AppStream 2.0 fleet instances have user and instance metadata available through Windows environment variables. You can use the following environment variables in your applications and scripts to modify your environment based on the fleet instance details.

Environment Variable	Context	Description
AppStream _Stack_Name	User	The name of the stack from which the streaming session started.
AppStream _User_Acc ess_Mode	User	The access mode used to manage user access to the stream. The available values are **`custom`**, **`userpool`**, or **`saml`**.
AppStream _Session_	User	The date and time when the user's streaming session started.

Environment Variable	Context	Description
Reservation_DateTime		
AppStream_UserName	User	The user name associated with the user.
AppStream_Session_ID	User	The session identifier for the user's streaming session.
APPSTREAM_SESSION_CONTEXT	Machine	Contains the parameters passed to your streaming application when a session is started. For more information, see Session Context. **Note** This environment variable is only available after the first application launch.
AppStream_Image_Arn	Machine	The ARN of the image that was used to create the streaming instance.
AppStream_Instance_Type	Machine	The streaming instance's type. For example, **`stream.standard.medium`** .
AppStream_Resource_Type	Machine	The type of AppStream 2.0 resource. The value is either **`fleet`** or **`image-builder`** .
AppStream_Resource_Name	Machine	The fleet's name.

On Linux fleet instances, these environment variables are exported through the following profile.d scripts:

- **User environment variables** in /etc/profile.d/appstream_user_vars.sh

- **System environment variables** in /etc/profile.d/appstream_system_vars.sh

To access the environment variables, you must explicitly source these files in your applications.

Update an AppStream 2.0 Fleet

You can make updates to an existing AppStream 2.0 fleet.

When you create a new AppStream 2.0 image, you must update your Always-On and On-Demand fleets to make the applications and data on the new image available to users. If your update is minor (for example, patching applications or the operating system), you can update your running fleet. When new streaming instances are created, they are created from the updated image. Changing the image on a running fleet does not disrupt users who have active streaming sessions. Unused streaming instances are replaced periodically, while streaming instances that users are connected to are terminated after the streaming sessions are finished.

You can update a fleet with a new image that runs the same operating system when the fleet is in a **Running** or **Stopped** state. However, you can update a fleet with a new image that runs a different operating system only when the fleet is in a **Stopped** state.

> **Note**
>
> The application catalog that AppStream 2.0 displays to users is based on the current image that is associated with the fleet. If the updated image contains applications that are not specified in the older image, the applications may not launch if the user streams from an instance that is based on the older image.

Update a fleet with a new image

To update an AppStream 2.0 fleet with a new image

To apply operating system updates or make new applications available to users, create a new image that has these changes. Then, update the fleet with the new image.

1. Connect to the image builder that you want to use and sign in with an account that has local administrator permissions on the image builder. To do so, do either of the following:

   - Use the AppStream 2.0 console (for web connections only)

- Create a streaming URL (for web or AppStream 2.0 client connections)

> **Note**
>
> If your organization requires smart card sign in, you must create a streaming URL and use the AppStream 2.0 client for the connection. For information about smart card sign in, see Smart Cards.

2. Do either or both of the following as required:

   - Install updates to the operating system.
   - Install applications.

   If an application requires the Windows operating system to restart, let it do so. Before the operating system restarts, you are disconnected from your image builder. After the restart is complete, connect to the image builder again, then finish installing the application.

3. On the image builder desktop, open Image Assistant.
4. Follow the necessary steps in Image Assistant to finish creating your image. For more information, see Tutorial: Create a Custom AppStream 2.0 Image by Using the AppStream 2.0 Console.

   After the image status changes to **Available**, you can update the fleet with your new image.

5. In the left navigation pane, choose **Fleets**.
6. Select the fleet that you want to update with the new image.
7. On the **Fleet Details** tab, choose **Edit**.
8. In the **Edit Fleet** dialog box, the list of available images displays in the **Name** list. Select the new image from the list.
9. Choose **Update**.

Manage applications associated to an Elastic fleet

To manage applications associated to an Elastic fleet

You can associate and disassociate applications from an Elastic fleet at any time. Changes to the applications associated to an Elastic fleet are visible to users currently streaming from the fleet, but may not take effect. For example, if you disassociate an application from a fleet, it will be removed

from the application catalog, but the virtual hard disk will remain mounted to existing streaming sessions.

1. Open the AppStream 2.0 console.
2. In the left navigation pane, choose **Fleets**, then either select the name of the fleet, or select the fleet radio button, then choose **View details**.
3. To associate a new application to the fleet, choose **Associate** in **Assigned applications**, select the application to be associated, and choose **Associate**.
4. To disassociate an existing application from the fleet, select the application to disassociate, choose **Disassociate**, and confirm that you want to disassociate the selected application by choosing **Disassociate**.

Fleet Auto Scaling for Amazon AppStream 2.0

Fleet Auto Scaling lets you change the size of your AppStream 2.0 Always-On or On-Demand fleet automatically to match the supply of available instances to user demand. The size of your fleet determines the number of users who can stream concurrently. For a multi-session fleet, more than one user can use a single instance. For a non multi-session fleet, one instance is required for each user session. You can specify your fleet capacity in terms of instances (for single-session fleets) and user sessions (for multi-session fleets). Based on your fleet configurations and auto scaling policies, the required number of instances will be made available. 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 available instances to match user demand. You can also choose to turn off automatic scaling and make the fleet run at a fixed size.

> **Note**
>
> Elastic fleet capacity is automatically managed by AppStream 2.0 for you. You do not need to create auto scaling rules to manage the number of fleet streaming instances that are available for Elastic fleets.

> **Note**
>
> As you develop your plan for AppStream 2.0 fleet scaling, make sure that your network configuration meets your requirements.

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 Using AWS Managed Policies and Linked Roles to Manage Administrator Access to AppStream 2.0 Resources and Using IAM Policies to Manage Administrator Access to Application Auto Scaling.

> **Note**
>
> When you use scaling, you work with the Application Auto Scaling API. For Fleet Auto Scaling to work correctly for AppStream 2.0, Application Auto Scaling requires permission to describe and update your AppStream 2.0 fleets and describe your Amazon CloudWatch alarms, and permissions to modify your fleet capacity on your behalf. For more information, see Roles Required for AppStream 2.0 , Application Auto Scaling, and AWS Certificate Manager Private CA and Using IAM Policies to Manage Administrator Access to Application Auto Scaling.

The following topics provide information to help you understand and use AppStream 2.0 Fleet Auto Scaling.

Contents

- Scaling Concepts
- Managing Fleet Scaling Using the Console
- Managing Fleet Scaling Using the AWS CLI
- Additional Resources

Scaling Concepts

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

For step-by-step guidance for working with AppStream 2.0 Fleet Auto Scaling, see Scaling Your Desktop Application Streams with Amazon AppStream 2.0 in the *AWS Compute Blog*.

To use Fleet Auto Scaling effectively, you must understand the following terms and concepts.

Multi-session vs. Single-session

In a single-session scenario, each user session has its own dedicated instance. In a multi-session mode, more than one user session can be provisioned on an instance. Fleet capacity and auto scaling policies must be configured in terms of user sessions, and the service will calculate and launch the required number of instances.

Minimum Capacity/Minimum User Sessions for fleet

The minimum number of instances (for single session fleets) or user sessions (for multi-session fleets). The number of instances (for single session fleets) or user sessions (for multi-session fleets) can't be below this value, and scaling policies will not scale your fleet below this value. For example, in a single-session scenario, if you set the minimum capacity for a fleet to 2, your fleet will never have less than 2 instances. Similarly, in a multi-session scenario, with the maximum number of sessions on an instance set to 5, if you set the minimum capacity for a fleet to 12, your fleet will never have less than roundup (12/5) = 3 instances.

Maximum Capacity/Maximum User Sessions for fleet

The maximum number of instances (for single session fleets) or user sessions (for multi-session fleets). The number of instances (for single session fleets) or user sessions (for multi-session fleets) can't be above this value, and scaling policies will not scale your fleet above this value. For example, in a single-session scenario, if you set the maximum capacity for a fleet to 10, your fleet will never have more than 10 instances. Similarly, in a multi-session scenario, with maximum number of sessions on an instance set to 5, if you set the maximum capacity for a fleet to 52, your fleet will never have more than roundup (52/5) = 11 instances.

Desired Capacity

The total number of instances (for single session fleets) or user sessions (for multi-session fleets) that are either running or pending. This value represents the total number of concurrent streaming sessions that your fleet can support in a steady state. To set the value for **Desired Capacity**, edit **Fleet Details**. We do not recommend changing the **Desired Capacity** value manually when you use **Scaling Policies**.

If the value of **Desired Capacity** 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 case, a scale-in activity does not adjust **Desired Capacity**, because it is already below the **Minimum Capacity** value.

If the value of **Desired Capacity** is set above the value of **Maximum Capacity** and a scale-in activity is triggered, Application Auto Scaling scales the **Desired Capacity** value down to the value of **Maximum Capacity** and then continues to scale in as required, based on the scaling policy. However, in this case, 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 single session fleets) or **user sessions** (for multi-session fleets). 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

Your scaling policy is based on this metric. The following metrics are available for scaling policies:

Capacity Utilization

The 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

The number of instances (for single session fleets) or user sessions (for multi-session fleets) in your fleet that are available for users. 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) (for single session fleets) or user session(s) (for multi-session fleets)".

Insufficient Capacity Error

The number of session requests rejected due to lack of capacity. You can use this metric to provision new instances for users who can't start streaming sessions due to lack of capacity.

For example, **Scaling Policy Condition**: "If Insufficient Capacity Error > 0" perform **Scaling Policy Action**: "Add 1 instance(s) (fr single session fleets) or user session(s) (for multi-session fleets)".

Managing Fleet Scaling Using the Console

You can set up and manage fleet scaling by using the AppStream 2.0 console in either of the following two ways: During fleet creation, or any time, by using the **Fleets** tab. Two default scaling policies are associated with newly created fleets after launch. You can edit these policies on the **Scaling Policies** tab in the AppStream 2.0 console. For more information, see Create a Fleet.

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 ro user sessions.

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 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 = 10
- Maximum Capacity = 50
- Scale Out = Add 5 instances (for single session fleets) or user sessions (for multi-session fleets) if Capacity Utilization > 75%
- Scale In = Remove 6 instances (for single session fleets) or user sessions (for multi-session fleets) if Capacity Utilization < 25%

> **Note**
>
> The above policy is applicable in both single-session and multi-session scenarios. In a single session scenario, 5 new instances will be launched during a scale out event, and 4 instances will be reclaimed during the scale down event. In a multi-session scenario, with the maximum sessions per instance = 4, the scale out event will trigger a launch of roundup (add 5 user sessions/ maximum sessions per instance 4) = 2 instances. During a scale in event, services will reclaim roundup (remove 6 user sessions/maximum sessions per instance 4) = 2 instances. Instances with running user sessions will not be reclaimed. Only instances with no user sessions running will be reclaimed.

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.
4. For **Scaling Policies**, remove all policies associated with the fleet.
5. For **Fleet Details**, edit the fleet to set **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 available 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 by using the AWS Command Line Interface (AWS CLI). For more advanced features such as setting up multiple scaling policies or setting scale-in and scale-out cooldown times, use the AWS CLI. Before running scaling policy commands, you must register your fleet as a scalable target. To do so, use the following register-scalable-target 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
```

Examples

- Example 1: Applying a Scaling Policy Based on Capacity Utilization
- Example 2: Applying a Scaling Policy Based on Insufficient Capacity Errors
- Example 3: Applying a Scaling Policy Based on Low Capacity Utilization
- Example 4: Change the Fleet Capacity Based on a Schedule
- Example 5: Applying a Target Tracking Scaling Policy

Example 1: Applying a Scaling Policy Based on Capacity Utilization

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

The following put-scaling-policy 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": 120
    }
}
```

If the command is successful, the output is similar to 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/scale-out-
utilization-policy"}
```

Now, set up a CloudWatch alarm for this policy. Use the names, Region, account number, and policy identifier that apply to you. 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 GreaterThanOrEqualToThreshold \
--dimensions "Name=Fleet,Value=fleetname" \
--evaluation-periods 1 --unit Percent \
--alarm-actions "arn:aws:autoscaling:your-region-code:account-
number-without-hyphens:scalingPolicy:policyid:resource/appstream/
fleet/fleetname:policyName/policyname"
```

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

This AWS CLI example sets up a scaling policy that scales out the fleet by 1 if the fleet returns 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": 120
    }
}
```

If the command is successful, the output is similar to 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/scale-out-insufficient-capacity-policy"}
```

Now, set up a CloudWatch alarm for this policy. Use the names, Region, account number, and policy identifier that apply to you. 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=Fleet,Value=fleetname" \
--evaluation-periods 1 --unit Count \
--alarm-actions "arn:aws:autoscaling:your-region-code:account-
number-without-hyphens:scalingPolicy:policyid:resource/appstream/
fleet/fleetname:policyName/policyname"
```

Example 3: Applying a Scaling Policy Based on Low Capacity Utilization

This AWS CLI example sets up a scaling policy that scales in the fleet to reduce actual capacity when `CapacityUtilization` is low.

The following command defines an excess capacity-based scaling policy:

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

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

```
{
    "PolicyName": "policyname",
    "ServiceNamespace": "appstream",
    "ResourceId": "fleet/fleetname",
    "ScalableDimension": "appstream:fleet:DesiredCapacity",
    "PolicyType": "StepScaling",
    "StepScalingPolicyConfiguration": {
        "AdjustmentType": "PercentChangeInCapacity",
        "StepAdjustments": [
            {
                "MetricIntervalUpperBound": 0,
                "ScalingAdjustment": -25
            }
        ],
        "Cooldown": 360
    }
}
```

If the command is successful, the output is similar to the following, although some details are unique to your account and Region. In this example, the policy identifier is `12ab3c4d-56789-0ef1-2345-6ghi7jk8lm90`.

```
{"PolicyARN": "arn:aws:autoscaling:us-west-2:123456789012:scalingPolicy:12ab3c4d-56789-0ef1-2345-6ghi7jk8lm90:resource/appstream/fleet/SampleFleetName:policyName/scale-in-utilization-policy"}
```

Now, set up a CloudWatch alarm for this policy. Use the names, Region, account number, and policy identifier that apply to you. 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 is less than or equal to 25
 percent" \
--metric-name CapacityUtilization \
--namespace AWS/AppStream \
--statistic Average \
--period 120 \
--threshold 25 \
--comparison-operator LessThanOrEqualToThreshold \
--dimensions "Name=Fleet,Value=fleetname" \
--evaluation-periods 10 --unit Percent \
--alarm-actions "arn:aws:autoscaling:your-region-code:account-
number-without-hyphens:scalingPolicy:policyid:resource/appstream/
fleet/fleetname:policyName/policyname"
```

Example 4: Change the Fleet Capacity Based on a Schedule

Changing your fleet capacity based on a schedule lets you scale your fleet capacity in response to predictable changes in demand. For example, at the start of a work day, you might expect a certain number of users to request streaming connections at one time. To change your fleet capacity based on a schedule, you can use the Application Auto Scaling PutScheduledAction API action or the put-scheduled-action AWS CLI command.

Before changing your fleet capacity, you can list your current fleet capacity by using the AppStream 2.0 describe-fleets AWS CLI command.

```
aws appstream describe-fleets --name fleetname
```

The current fleet capacity will appear similar to the following output (shown in JSON format):

```
{
    {
            "ComputeCapacityStatus": {
                "Available": 1,
                "Desired": 1,
                "Running": 1,
                "InUse": 0
            },
}
```

Then, use the `put-scheduled-action` command to create a scheduled action to change your fleet capacity. For example, the following command changes the minimum capacity to 3 and the maximum capacity to 5 every day at 9:00 AM UTC.

> **Note**
>
> For cron expressions, specify when to perform the action in UTC. For more information, see Cron Expressions.

```
aws application-autoscaling put-scheduled-action --service-namespace appstream \
--resource-id fleet/fleetname \
--schedule="cron(0 9 * * ? *)" \
--scalable-target-action MinCapacity=3,MaxCapacity=5 \
--scheduled-action-name ExampleScheduledAction \
--scalable-dimension appstream:fleet:DesiredCapacity
```

To confirm that the scheduled action to change your fleet capacity was successfully created, run the describe-scheduled-actions command.

```
aws application-autoscaling describe-scheduled-actions --service-namespace appstream --
resource-id fleet/fleetname
```

If the scheduled action was successfully created, the output appears similar to the following.

```
{
    "ScheduledActions": [
        {
            "ScalableDimension": "appstream:fleet:DesiredCapacity",
            "Schedule": "cron(0 9 * * ? *)",
            "ResourceId": "fleet/ExampleFleet",
            "CreationTime": 1518651232.886,
            "ScheduledActionARN": "<arn>",
            "ScalableTargetAction": {
                "MinCapacity": 3,
                "MaxCapacity": 5
            },
            "ScheduledActionName": "ExampleScheduledAction",
            "ServiceNamespace": "appstream"
        }
```

```
    ]
}
```

For more information, see Scheduled Scaling in the *Application Auto Scaling User Guide*.

Example 5: Applying a Target Tracking Scaling Policy

With target tracking scaling, you can specify a capacity utilization level for your fleet.

When you create a target tracking scaling policy, Application Auto Scaling automatically creates and manages CloudWatch alarms that trigger the scaling policy. The scaling policy adds or removes capacity as required to keep capacity utilization at, or close to, the specified target value. To ensure application availability, your fleet scales out proportionally to the metric as fast as it can but scales in more gradually.

The following put-scaling-policy command defines a target tracking scaling policy that attempts to maintain 75% capacity utilization for an AppStream 2.0 fleet.

```
aws application-autoscaling put-scaling-policy --cli-input-json file://config.json
```

The contents of the file `config.json` are as follows:

```
{
  "PolicyName":"target-tracking-scaling-policy",
  "ServiceNamespace":"appstream",
  "ResourceId":"fleet/fleetname",
  "ScalableDimension":"appstream:fleet:DesiredCapacity",
  "PolicyType":"TargetTrackingScaling",
  "TargetTrackingScalingPolicyConfiguration":{
    "TargetValue":75.0,
    "PredefinedMetricSpecification":{
      "PredefinedMetricType":"AppStreamAverageCapacityUtilization"
    },
    "ScaleOutCooldown":300,
    "ScaleInCooldown":300
  }
}
```

If the command is successful, the output is similar to the following, although some details are unique to your account and Region. In this example, the policy identifier is 6d8972f3-efc8-437c-92d1-6270f29a66e7.

```
{
    "PolicyARN": "arn:aws:autoscaling:us-west-2:123456789012:scalingPolicy:6d8972f3-
efc8-437c-92d1-6270f29a66e7:resource/appstream/fleet/fleetname:policyName/target-
tracking-scaling-policy",
    "Alarms": [
        {
            "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-fleet/fleetname-AlarmHigh-d4f0770c-b46e-434a-
a60f-3b36d653feca",
            "AlarmName": "TargetTracking-fleet/fleetname-AlarmHigh-d4f0770c-b46e-434a-
a60f-3b36d653feca"
        },
        {
            "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-fleet/fleetname-AlarmLow-1b437334-d19b-4a63-
a812-6c67aaf2910d",
            "AlarmName": "TargetTracking-fleet/fleetname-AlarmLow-1b437334-d19b-4a63-
a812-6c67aaf2910d"
        }
    ]
}
```

For more information, see Target Tracking Scaling Policies in the *Application Auto Scaling User Guide*.

Additional Resources

For step-by-step guidance for working with AppStream 2.0 Fleet Auto Scaling, see Scaling Your Desktop Application Streams with Amazon AppStream 2.0 in the *AWS Compute Blog*.

To learn more about using the Application Auto Scaling AWS CLI commands or API actions, see the following resources:

- application-autoscaling section of the *AWS CLI Command Reference*
- Application Auto Scaling API Reference
- Application Auto Scaling User Guide

Multi-Session Recommendations

When deciding the maximum number of user sessions on an instance in a multi-session environment, you should consider several factors to ensure optimal performance and streaming experience. The following are recommendations for you to determine the optimum number of user sessions on an instance:

- *Evaluate resource requirements*: Understand the resource requirements of the applications being used within the sessions. Consider factors such as CPU, memory, disk I/O, and network bandwidth. This evaluation will help determine the amount of resources each user session typically requires.
- *Consider instance specifications*: Take into account the specifications of the instance, including the number of CPUs, available memory, and GPU specifications. Instances with higher specifications can handle a larger number of user sessions. For more information on different instance types supported by AppStream 2.0 and pricing, see Amazon AppStream 2.0 pricing.
- *Performance testing*: Conduct performance testing on the applications and workload expected to run within the user sessions. Measure resource utilization, response times, and overall system performance. Use this data to assess the impact of concurrent user sessions on performance, and determine the optimal session-to-instance ratio. You can run these assessments across different instance types offered by AppStream 2.0 to find the optimal instance type or size for your end users. For more information on different instance types offered by AppStream 2.0, see the section called “Instance Families”.
- *Monitor resource utilization*: Continuously monitor the resource utilization of the instance during normal usage. Observe CPU, memory, and disk utilization. Ensure that the resource utilization remains within acceptable limits to avoid performance degradation. For a multi-session environment, you can view these metrics on AppStream 2.0 and the CloudWatch console. For more information, see the section called “Monitoring Resources”.
- *Consider user behavior patterns*: Analyze user behavior patterns to understand peak usage periods and potential concurrent usage. Some users might have intermittent or sporadic usage patterns, while others might have consistent usage throughout the day. Account for these patterns when determining the maximum number of user sessions to avoid resource contention during peak periods.

  AppStream 2.0 enables you to configure a maximum of 50 user sessions per instance, regardless of the instance type or size that you choose. However, this is just an upper limit, and not a recommended limit. The following is an example table to help you determine the maximum

number of user sessions on an instance in a multi-session fleet. The recommended maximum number of users listed in the table is based on general guidelines and assumptions. Testing with the real-life workload is crucial, since actual performance can vary, depending on the workload's individual characteristics, the application's resource requirements, and user behavior.

Recommendations based on workload types

End User Category	Workload Type	Example Users	Use Cases	Recommended Configuration(s)
End users who conduct a single task and use minimal applications	Light	Task workers, Front desk users	Data entry applications, Text editing, Bastion host	4 users per vCPU on Stream.standard.xlarge/2xlarge or Stream.compute.xlarge+ or Stream.memory.xlarge+ or
End users who conduct a single task and use minimal applications	Light to Medium	Task workers, Front desk users, Contact center employees	Data entry applications, Text editing, Bastion host, Chat, Email, Messaging apps	2 users per vCPU on Stream.standard.xlarge/2xlarge or Stream.compute.xlarge+ or Stream.memory.xlarge+ or
End users who create complex spreadsheets, presentations, and large documents	Medium	Task workers, Contact center employees, Business analysts	Data entry applications, Chat, Email, Messaging apps, Productivity apps	2 users per vCPU on Stream.memory.xlarge+ or Stream.compute.xlarge+
End users with high performance workloads	Medium to Heavy	Knowledge workers, Software	Software Scripting	1 user per vCPU on Stream.memory.xlarge+

End User Category	Workload Type	Example Users	Use Cases	Recommended Configuration(s)
		developers, Business intelligence analysts		or Stream.compute.xlarge+
End users with high performance workloads	Heavy	Knowledge workers, Software developers, Data scientists	Screen sharing, Software Scripting, Data analystics	1 user per vCPU on Stream.memory.xlarge+ or Stream.compute.xlarge+
End users with workloads that require graphics and heavy compute/memory resources	Heavy to Accelerated	Graphics/ Architecture designers, CAD/ CAM users	Graphics-intensive applications, such as remote graphics workstations	1 user per 2 vCPU Graphics.g4dn.*
End users with workloads that require graphics and heavy compute/memory resources	Accelerated	Video editors, Gamers and game developers, Data miners, GIS data engineers, AI scientists	Video transcoding and 3D rendering, Photo-realistic design, Graphics workstations, ML model training, ML inference	1 user per 2 vCPU Graphics.G5.*

User Authentication

The following topics provide information about Amazon AppStream 2.0 user authentication and authorization.

Contents

- AppStream 2.0 User Pools
- Amazon AppStream 2.0 Integration with SAML 2.0

AppStream 2.0 User Pools

The AppStream 2.0 user pool provides a simplified way to manage access to applications for your users through a persistent portal for each AWS Region. This feature is a built-in alternative to user management through Active Directory and SAML 2.0 federation. Stacks can't be assigned to users in the user pool if the stacks are associated with a fleet that is joined to an Active Directory domain.

The AppStream 2.0 user pool provides the following key features:

- Users can access application stacks through a persistent URL and login credentials by using their email address and a password that they choose.
- Users' email addresses are case-sensitive. During login, if they specify an email address that doesn't use the same capitalization as the email address specified when their user pool account was created, a "user does not exist" error message displays.
- You can assign multiple stacks to users. Doing so enables AppStream 2.0 to display multiple application catalogs to users when they log in.
- When you create new users, a welcome email is automatically sent to them. The email includes instructions, a login portal link, and a temporary password for connecting to the login portal.
- After you create users, they are enabled unless you specifically disable them.
- You can control which users have access to which application stacks, or disable access completely.

User Pool End User Experience

The following steps summarize the initial connection experience for users in the user pool.

1. You create new users in the Region you want by specifying their email addresses.

2. AppStream 2.0 sends them a welcome email.
3. You assign one or more stacks to the users.
4. AppStream 2.0 sends them an optional notification email. This email includes information about how to access the stacks that are newly assigned to them.
5. The users connect to the login portal by entering the information included in the welcome email, and they set a permanent password. The login portal link never expires and can be used any time.
6. They sign in to AppStream 2.0 by entering their email address and permanent password.
7. After they sign in, the users can view their application catalogs.

The login portal link provided in the welcome email should be saved for future use, as it does not change and is valid for all users in the user pool. The login portal URL and users in the user pool are managed on a per-Region basis.

Resetting a Forgotten Password

If users forget their password, follow these steps to connect to the login portal link (provided in the welcome email) and choose a new password.

To choose a new password

1. Open the AppStream 2.0 login portal by using the login link provided in the welcome email.
2. Choose **Forgot Password?**.
3. Type the email address that you used to create the user in the user pool, and choose **Next**.

   Your email address is case-sensitive. During login, if your email address doesn't use the same capitalization as the email address specified when your user pool account was created, a "user does not exist" error message displays.
4. Check your email for the password reset request message. If you are having difficulty finding the email, check your spam email folder. Type the verification code from the email in **Verification Code**.

   > **Note**
   >
   > The verification code is valid for 24 hours. If a new password is not chosen within this time, request a new verification code.

5. Following the password rules shown, type and confirm your new password. Choose **Reset Password**.

User Pool Administration

To create and manage users in the user pool, sign in to the AppStream 2.0 console for the AWS Region you want and choose **User Pool** in the left navigation pane. The User Pool dashboard supports bulk operations on a list of users for some actions. You can select multiple users on which to perform the same action from the **Actions** list. Users in the user pool are created and managed on a per-Region basis.

AppStream 2.0 does not support bulk user creation or disable. However, you can use Amazon Cognito with the CreateStreamingURL API action to manage access efficiently for multiple users. Amazon Cognito user pools let you quickly create your own directory to sign up and sign in users. In addition, you can use Amazon Cognito user pools to store user profiles. For information about how to integrate AppStream 2.0 with your Cognito User Pool, see the Create a SaaS Portal with Amazon AppStream 2.0 tutorial.

> **Note**
>
> AppStream 2.0 sends email to users on your behalf when you create a new user created or assign a user to a stack. To ensure the email is delivered, add `no-reply@accounts.`*`aws-region-code`*`.amazonappstream.com` to your allow list, where *`aws-region-code`* is a valid AWS Region code in which you are working. If users are having difficulty finding the emails, ask them to check their "spam" email folder.

Tasks

- Creating a User
- Deleting a User
- Assigning Stacks to Users
- Unassigning Stacks from Users
- Disabling Users
- Enabling Users
- Re-Sending Welcome Email

Creating a User

You must enter a valid and unique email address for each new user within a Region. However, you can reuse an email address for a new user in another Region.

When you create a new user, be aware of the following:

- You cannot change the email address, first name, or last name for a user that you have already created. To change this information for a user, disable the user. Then, recreate the user (as a new user) and specify the updated information as needed.
- Users' email addresses are case-sensitive. During login, if they specify an email address that doesn't use the same capitalization as the email address specified when their user pool account was created, a "user does not exist" error message displays.
- You can assign one or more stacks to the user after the user is created.

To create a new user

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. In the left navigation pane, choose **User Pool**, **Create User**.
3. For **Email**, type the unique email address for the user.
4. Type the user's first name and last name in the corresponding fields. These fields need not be unique.
5. Choose **Create User**.

After users are created, AppStream 2.0 sends them a welcome email. This email includes the login portal link, the login email address to be used, and a temporary password. By browsing to the login portal and typing their temporary password, users can set a permanent password to access their applications.

By default, the new user's status is **Enabled**, meaning you can assign one or more stacks to the user or perform other administrative actions.

Deleting a User

You can enable or disable a user, but you cannot delete a user by using the AppStream 2.0 console. To delete a user, you must use the DeleteUser API action.

Assigning Stacks to Users

You can assign one or more stacks to one or more users in the user pool. After they are assigned to at least one stack, users can log in to AppStream 2.0 and launch applications. If users are assigned to more than one stack, they are presented with a list of stacks as catalogs to choose from before launching applications.

> **Note**
>
> Stacks can't be assigned to users if the stacks are associated with a fleet that is joined to an Active Directory domain.

To assign a stack to users

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. In the left navigation pane, choose **User Pool** and select the users you want.
3. Choose **Actions**, **Assign stack**. For more information, see Using Active Directory with AppStream 2.0.
4. Review the list to confirm that the correct users are specified. For **Stack**, choose the stack you want to assign.
5. By default, **Send email notification to user** is enabled. Clear this option if you do not want to send the notification email to users now.
6. Choose **Assign stack**.

Unassigning Stacks from Users

You can unassign a stack from one or more users in the user pool. After a stack is unassigned from users, they can't launch applications from the stack. If users are connected when you unassign the stack, their sessions remain active until the session cookie expires (about one hour).

To unassign a stack from users

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. In the left navigation pane, choose **User Pool** and select the users you want.
3. Choose **Actions**, **Unassign stack**.

4. Review the list to confirm that the correct users are specified. For **Stack**, choose the stack you want to unassign. The list includes all stacks, assigned or unassigned.
5. Choose **Unassign stack**.

Disabling Users

You can disable one or more users in the user pool, one at a time. After they are disabled, users can no longer log in to AppStream 2.0 until they are re-enabled. This action does not delete users. If users are connected when you disable them, their sessions remain active until the session cookie expires (about one hour). Stack assignments for the users are retained. If the users are re-enabled, their stack assignments become active again.

To disable a user

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. In the left navigation pane, choose **User Pool** and select the user you want.
3. Choose **Actions**, **Disable user**.
4. Confirm that the correct user is specified, and choose **Disable User**.

Enabling Users

You can enable one or more users in the user pool, one at a time. After they are enabled, users can log in to AppStream 2.0 and launch applications from the stacks to which they are assigned. If the users were disabled, these assignments are retained.

To enable users

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. In the left navigation pane, choose **User Pool** and select the user you want.
3. Choose **Actions**, **Enable user**.
4. Confirm that the correct user is specified, and choose **Enable User**.

Re-Sending Welcome Email

You can re-send the welcome email with connection instructions to users in the user pool. Unused passwords expire after seven days. To provide a new temporary password, you must re-send the

welcome email. This option is only available until users set their permanent password. If they've already set their password and forgotten it, they can set a new one. For more information, see Resetting a Forgotten Password.

To resend the welcome email for a user

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. In the left navigation pane, choose **User Pool** and select the user you want.
3. For **User Details**, choose **Resend welcome email**.
4. Confirm that the success message displays at the top of the User Pool dashboard.

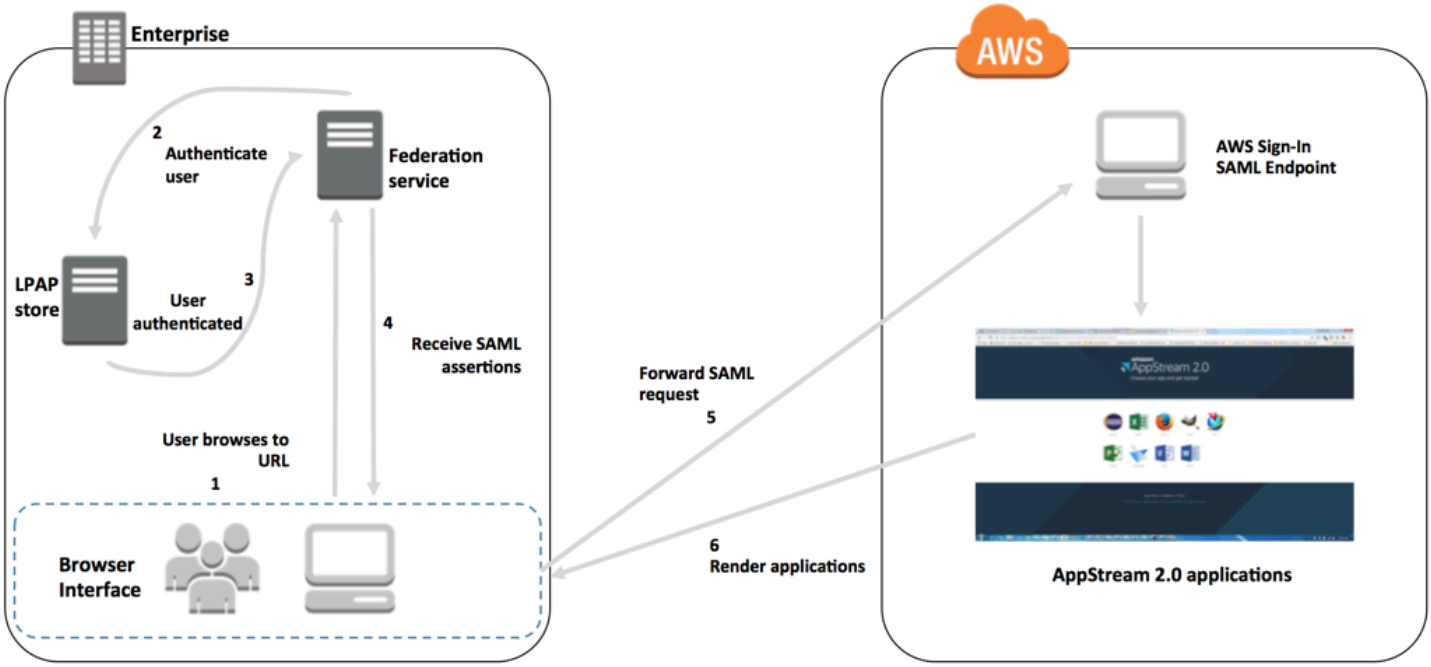
Amazon AppStream 2.0 Integration with SAML 2.0

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 (IdP) that supports SAML 2.0—such as Active Directory Federation Services (AD FS) in Windows Server, Ping One Federation Server, or Okta—to provide an 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 IdP. By using your IdP, you can control which users have access to a particular AppStream 2.0 stack.

Example Authentication Workflow

The following diagram illustrates the authentication flow between AppStream 2.0 and a third-party identity provider (IdP). In this example, the administrator has set up a sign-in page to access AppStream 2.0, called `applications.exampleco.com`. The webpage 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 AppStream 2.0.

   For information about working with SAML in the AWS GovCloud (US) Regions, see AWS Identity and Access Management in the *AWS GovCloud (US) User Guide*.
6. Using the authentication token from AWS, AppStream 2.0 authorizes the user and presents applications to the browser.

From the user's perspective, this process happens transparently. The user starts at your organization's internal portal and is automatically redirected to an AppStream 2.0 application portal without being required to enter AWS credentials.

Setting Up SAML

To enable users to sign in to AppStream 2.0 by using their existing credentials, and start streaming applications, you can set up identity federation using SAML 2.0. To do this, use an IAM role and a relay state URL to configure your SAML 2.0-compliant identity provider (IdP) and enable AWS to permit your federated users to access an AppStream 2.0 stack. The IAM role grants users the permissions to access the stack. The relay state is the stack portal to which users are forwarded after successful authentication by AWS.

Contents

- Prerequisites
- Step 1: Create a SAML Identity Provider in AWS IAM
- Step 2: Create a SAML 2.0 Federation IAM Role
- Step 3: Embed an Inline Policy for the IAM Role
- Step 4: Configure Your SAML-Based IdP
- Step 5: Create Assertions for the SAML Authentication Response
- Step 6: Configure the Relay State of Your Federation

Prerequisites

Complete the following prerequisites before configuring your SAML 2.0 connection.

1. Configure your SAML-based IdP to establish a trust relationship with AWS.
   - Inside your organization's network, configure your identity store to work with a SAML-based IdP. For configuration resources, see AppStream 2.0 Integration with SAML 2.0.
   - Use your SAML-based IdP to generate and download a federation metadata document that describes your organization as an IdP. This signed XML document is used to establish the relying party trust. Save this file to a location that you can access from the IAM console later.
2. Use the AppStream 2.0 management console to create an AppStream 2.0 stack. You need the stack name to create the IAM policy and to configure your IdP integration with AppStream 2.0, as described later in this topic.

   You can create an AppStream 2.0 stack by using the AppStream 2.0 management console, AWS CLI, or AppStream 2.0 API. For more information, see Create an AppStream 2.0 Fleet and Stack.

Step 1: Create a SAML Identity Provider in AWS IAM

First, create a SAML IdP in AWS IAM. This IdP defines your organization's IdP-to-AWS trust relationship using the metadata document generated by the IdP software in your organization. For more information, see Creating and Managing a SAML Identity Provider (Amazon Web Services Management Console) in the *IAM User Guide*. For information about working with SAML IdPs in the AWS GovCloud (US) Regions, see AWS Identity and Access Management in the *AWS GovCloud (US) User Guide*.

Step 2: Create a SAML 2.0 Federation IAM Role

Next, create a SAML 2.0 federation IAM role. This step establishes a trust relationship between IAM and your organization's IdP, which identifies your IdP as a trusted entity for federation.

To create an IAM role for the SAML IdP

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Roles**, **Create role**.
3. For **Role type**, choose **SAML 2.0 federation**.
4. For **SAML Provider**, select the SAML IdP that you created.

   > **Important**
   >
   > Do not choose either of the two SAML 2.0 access methods (**Allow programmatic access only** or **Allow programmatic and Amazon Web Services Management Console access**).

5. For **Attribute**, choose **SAML:aud**.
6. For **Value**, enter **`https://signin.aws.amazon.com/saml`**. This value restricts role access to SAML user streaming requests that include a SAML subject type assertion with a value of persistent. If the SAML:sub_type is persistent, your IdP sends the same unique value for the NameID element in all SAML requests from a particular user. For more information about the SAML:sub_type assertion, see the *Uniquely Identifying Users in SAML-Based Federation* section in Using SAML-Based Federation for API Access to AWS.
7. Review your SAML 2.0 trust information, confirming the correct trusted entity and condition, and then choose **Next: Permissions**.
8. On the **Attach permissions policies** page, choose **Next: Tags**.

9. (Optional) Enter a key and value for each tag that you want to add. For more information, see Tagging IAM Users and Roles.
10. When you're done, choose **Next: Review**. Later, you'll create and embed an inline policy for this role.
11. For **Role name**, enter a name that identifies the purpose of this role. Because multiple entities might reference the role, you can't edit the role's name once it is created.
12. (Optional) For **Role description**, enter a description for the new role.
13. Review the role details and choose **Create role**.
14. (Optional) If you plan to use attribute-based application entitlements using a third-party SAML 2.0 identity provider or certificate-based authentication, you must add the sts:TagSession permission to your new IAM role's trust policy. For more information, see Attribute-Based Application Entitlements Using a Third-Party SAML 2.0 Identity Provider and Passing session tags in AWS STS.

    In your new IAM role's details, choose the **Trust relationships** tab, and then choose **Edit trust relationship**. The Edit Trust Relationship policy editor starts. Add the **sts:TagSession** permission, as follows:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::ACCOUNT-ID-WITHOUT-HYPHENS:saml-
provider/IDENTITY-PROVIDER"
      },
      "Action": [
        "sts:AssumeRoleWithSAML",
        "sts:TagSession"
      ],
      "Condition": {
        "StringEquals": {
          "SAML:sub_type": "persistent"
        }
      }
    }
  ]
```

```
}
```

Replace *IDENTITY-PROVIDER* with the name of the SAML IdP you created in Step 1. Then choose **Update Trust Policy**.

Step 3: Embed an Inline Policy for the IAM Role

Next, embed an inline IAM policy for the role that you created. When you embed an inline policy, the permissions in that policy can't be accidentally attached to the wrong principal entity. The inline policy provides federated users with access to the AppStream 2.0 stack that you created.

1. In the details for the IAM role that you created, choose the **Permissions** tab, and then choose **Add inline policy**. The Create policy wizard starts.
2. In **Create policy**, choose the **JSON** tab.
3. Copy and paste the following JSON policy into the JSON window. Then, modify the resource by entering your AWS Region Code, account ID, and stack name. In the following policy, `"Action": "appstream:Stream"` is the action that provides your AppStream 2.0 users with permissions to connect to streaming sessions on the stack that you created.

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

Replace *REGION-CODE* with the AWS Region where your AppStream 2.0 stack exists. Replace *STACK-NAME* with the name of the stack. *STACK-NAME* is case-sensitive, and must match

the exact case and spelling as the stack name shown in the the **Stacks** dashboard of the AppStream 2.0 management console.

For resources in the AWS GovCloud (US) Regions, use the following format for the ARN:

`arn:aws-us-gov:appstream:`*`REGION-CODE`*`:`*`ACCOUNT-ID-WITHOUT-HYPHENS`*`:stack/`*`STACK-NAME`*

4. (Optional) If you are planning to use attribute-based application entitlements using a third-party SAML 2.0 identity provider with **SAML 2.0 Multi-stack Application Catalogs**, the Resource in your IAM role inline policy must be **`"Resource": "arn:aws:appstream:`*`REGION-CODE`*`:`*`ACCOUNT-ID-WITHOUT-HYPHENS`*`:stack/*"`** to allow application entitlements to control streaming access to stacks. To enforce additional protection on a stack resource, you can add an explicit deny in the policy. For more information, see Attribute-Based Application Entitlements Using a Third-Party SAML 2.0 Identity Provider and Policy evaluation logic.
5. When you're done, choose **Review policy**. The Policy Validator reports any syntax errors.

Step 4: Configure Your SAML-Based IdP

Next, depending on your SAML-based IdP, you may need to manually update your IdP to trust AWS as a service provider by uploading the `saml-metadata.xml` file at https://signin.aws.amazon.com/static/saml-metadata.xml to your IdP. This step updates your IdP's metadata. For some IdPs, the update may already be configured. If this is the case, proceed to the next step.

If this update is not already configured in your IdP, review the documentation provided by your IdP for information about how to update the metadata. Some providers give you the option to type the URL, and the IdP obtains and installs the file for you. Others require you to download the file from the URL and then provide it as a local file.

Step 5: Create Assertions for the SAML Authentication Response

Next, you may need to configure the information that your IdP sends to AWS as SAML attributes in its authentication response. Depending on your IdP, this information may already be pre-configured. If this is the case, skip this step and continue to Step 6,

If this information is not already configured in your IdP, provide the following:

- **SAML Subject NameID** – The unique identifier for the user who is signing in.

> **Note**
>
> For stacks with domain-joined fleets, the NameID value for the user must be provided in the format of "*domain*\username" using the sAMAccountName or "username@domain.com" using userPrincipalName. If you are using the sAMAccountName format, you can specify the *domain* by using either the NetBIOS name or the fully qualified domain name (FQDN). The sAMAccountName format is required for Active Directory one-way trust scenarios. For more information, see Using Active Directory with AppStream 2.0.

- **SAML Subject Type** (with a value set to `persistent`) – Setting the value to `persistent` ensures that your IdP sends the same unique value for the `NameID` element in all SAML requests from a particular user. Make sure that your IAM policy includes a condition to only allow SAML requests with a SAML sub_type set to `persistent`, as described in the section called "Step 2: Create a SAML 2.0 Federation IAM Role".
- **`Attribute` element with the `Name` attribute set to https://aws.amazon.com/SAML/Attributes/Role** – This element contains one or more `AttributeValue` elements that list the IAM role and SAML IdP to which the user is mapped by your IdP. The role and IdP are specified as a comma-delimited pair of ARNs.
- **`Attribute` element with the `Name` attribute set to https://aws.amazon.com/SAML/Attributes/RoleSessionName** – This element contains one `AttributeValue` element that provides an identifier for the AWS temporary credentials that are issued for SSO. The value in the `AttributeValue` element must be between 2 and 64 characters long, can contain only alphanumeric characters, underscores, and the following characters: + (plus sign), = (equals sign), , (comma), . (period), @ (at symbol), and - (hyphen). It cannot contain spaces. The value is typically a user ID (bobsmith) or an email address (bobsmith@example.com). It should not be a value that includes a space, such as a user's display name (Bob Smith).
- **`Attribute` element with the `Name` attribute set to https://aws.amazon.com/SAML/Attributes/PrincipalTag:SessionContext (optional)** – This element contains one `AttributeValue` element that provides parameters that can be used to pass session context parameters to your streaming applications. For more information, see Session Context.
- **`Attribute` element with the `Name` attribute set to https://aws.amazon.com/SAML/Attributes/PrincipalTag:ObjectSid (optional)** – This element contains one `AttributeValue` element that provides the Active Directory security identifier (SID) for the user who is signing in.

This parameter is used with certificate-based authentication to enable strong mapping to the Active Directory user.

- **`Attribute` element with the `Name` attribute set to https://aws.amazon.com/SAML/Attributes/PrincipalTag:Domain (optional)** – This element contains one `AttributeValue` element that provides the Active Directory DNS fully qualified domain name (FQDN) for the user who is signing in. This parameter is used with certificate-based authentication when the Active Directory `userPrincipalName` for the user contains an alternative suffix. The value must be provided in the format of **`domain.com`**, including any subdomains.
- **`Attribute` element with the `SessionDuration` attribute set to https://aws.amazon.com/SAML/Attributes/SessionDuration (optional)** – This element contains one `AttributeValue` element that specifies the maximum amount of time that a federated streaming session for a user can remain active before reauthentication is required. The default value is 60 minutes, or 3,600 seconds. For more information, see the *An optional Attribute element with the SessionDuration attribute set to https://aws.amazon.com/SAML/Attributes/SessionDuration* section in Configuring SAML Assertions for the Authentication Response.

> **Note**
>
> Although `SessionDuration` is an optional attribute, we recommend that you include it in the SAML response. If you do not specify this attribute, the session duration is set to a default value of 60 minutes.
> If your users access their streaming applications in AppStream 2.0 by using the AppStream 2.0 native client or using the web browser on the new experience, their sessions are disconnected after their session duration expires. If your users access their streaming applications in AppStream 2.0 by using a web browser on the old/classic experience, after the users' session duration expires and they refresh their browser page, their sessions are disconnected.

For more information about how to configure these elements, see Configuring SAML Assertions for the Authentication Response in the *IAM User Guide*. For information about specific configuration requirements for your IdP, see the documentation for your IdP.

Step 6: Configure the Relay State of Your Federation

Finally, use your IdP 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-endpoint?stack=stackname&accountId=aws-account-id-without-hyphens
```

Construct your relay state URL from your Amazon Web Services account ID, stack name, and the relay state endpoint associated with the Region in which your stack is located.

Optionally, you can specify the name of the application that you want to launch automatically. To find the application name, select the image in the AppStream 2.0 console, choose the **Applications** tab, and note the name that displays in the **Application Name** column. Alternatively, if you haven't yet created the image, connect to the image builder where you installed the application, and open Image Assistant. The names of applications display in the **Add Apps** tab.

If your fleet is enabled for the **Desktop** stream view, you can also choose to launch directly to the operating system desktop. To do so, specify **`Desktop`** at end of the relay state URL, after **`&app=`**.

With an identity provider (IdP) -initiated flow, after users sign into the IdP and select the AppStream 2.0 application from the IdP user portal, they are redirected to an AppStream 2.0 sign-in page in the browser with the following options:

- **Continue with Browser**
- **Open AppStream 2.0 client**

On the page, users can choose to start the session either in the browser, or with the AppStream 2.0 client application. Optionally, you can also specify which client should be used for a SAML 2.0 federation. To do this, specify either `native` or `web` at end of the relay state URL, after `&client=`. When the parameter is present in a relay state URL, the corresponding sessions will start in the specified client automatically, without users making the choice.

> **Note**
>
> This feature is only available if you use the new relay state region endpoints (in Table 1 below) to construct the relay state URL, and use the AppStream 2.0 client version 1.1.1300 and later.

With attribute-based application entitlements using a third-party SAML 2.0 identity provider, you can enable access to multiple stacks from a single relay state URL. Remove the stack and app (if present) parameters from the relay state URL, as follows:

```
https://relay-state-region-endpoint?accountId=aws-account-id-without-hyphens
```

When users federate to the AppStream 2.0 application catalog, they will be presented with all of the stacks where application entitlements have matched one or more applications to the user for the account ID and relay state endpoint associated with the Region in which your stacks are located. When a user selects a catalog, application entitlements will only display the applications the user is entitled to.

> **Note**
>
> Users cannot stream from multiple stacks at the same time.

For more information, see Attribute-Based Application Entitlements Using a Third-Party SAML 2.0 Identity Provider.

Table 1 below lists the relay state endpoints for the Regions where AppStream 2.0 is available. The relay state endpoints in Table 1 are compatible with the section called "AppStream 2.0 Web Browser Access (Version 2)" and the Windows client application version 1.1.1300 and later. If you are using older versions of the Windows client, you should use the old relay state endpoints listed in Table 2 to configure your SAML 2.0 federation. If you want your users to stream using a FIPS-compliant connection, you must use a FIPS-compliant endpoint. For more information about FIPS endpoints, see the section called "FIPS Endpoints".

Table 1: AppStream 2.0 relay state region endpoints

Region	Relay state endpoint
US East (N. Virginia)	`https://appstream2.euc-sso.us-east-1.aws.amazon.com/saml` (FIPS) `https://appstream2.euc-sso-fips.us-east-1.aws.amazon.com/saml`
US East (Ohio)	`https://appstream2.euc-sso.us-east-2.aws.amazon.com/saml`
US West (Oregon)	`https://appstream2.euc-sso.us-west-2.aws.amazon.com/saml` (FIPS) `https://appstream2.euc-sso-fips.us-west-2.aws.amazon.com/saml`
Asia Pacific (Mumbai)	`https://appstream2.euc-sso.ap-south-1.aws.amazon.com/saml`
Asia Pacific (Seoul)	`https://appstream2.euc-sso.ap-northeast-2.aws.amazon.com/saml`
Asia Pacific (Singapore)	`https://appstream2.euc-sso.ap-southeast-1.aws.amazon.com/saml`
Asia Pacific (Sydney)	`https://appstream2.euc-sso.ap-southeast-2.aws.amazon.com/saml`
Asia Pacific (Tokyo)	`https://appstream2.euc-sso.ap-northeast-1.aws.amazon.com/saml`
Canada (Central)	`https://appstream2.euc-sso.ca-central-1.aws.amazon.com/saml`
Europe (Frankfurt)	`https://appstream2.euc-sso.eu-central-1.aws.amazon.com/saml`

Region	Relay state endpoint
Europe (Ireland)	`https://appstream2.euc-sso.eu-west-1.aws.amazon.com/saml`
Europe (London)	`https://appstream2.euc-sso.eu-west-2.aws.amazon.com/saml`
AWS GovCloud (US-East)	`https://appstream2.euc-sso.us-gov-east-1.amazonaws-us-gov.com/saml` (FIPS) `https://appstream2.euc-sso-fips.us-gov-east-1.amazonaws-us-gov.com/saml` **Note** For more information about using AppStream 2.0 in AWS GovCloud (US) Regions, see Amazon AppStream 2.0 in the *AWS GovCloud (US) User Guide*.
AWS GovCloud (US-West)	`https://appstream2.euc-sso.us-gov-west-1.amazonaws-us-gov.com/saml` (FIPS) `https://appstream2.euc-sso-fips.us-gov-west-1.amazonaws-us-gov.com/saml` **Note** For more information about using AppStream 2.0 in AWS GovCloud (US) Regions, see Amazon AppStream 2.0 in the *AWS GovCloud (US) User Guide*.
South America (São Paulo)	`https://appstream2.euc-sso.sa-east-1.aws.amazon.com/saml`

Table 2 below lists the old relay state endpoints that are still available. However, it is recommended that you use the new relay state endpoints listed in Table 1 to configure your SAML 2.0 federations. In particular, with the new relay state endpoints, you can enable your users to launch the AppStream 2.0 client application (version 1.1.1300 and later) from IdP-initiated streaming sessions. The new relay state endpoints in Table 1 also allow users to sign into other AWS applications in different tabs of the same web browser, without impacting the ongoing AppStream 2.0 streaming session. The old relay state endpoints in Table 2 do not support this. For more information, see the section called "My AppStream 2.0 client users are getting disconnected from their AppStream 2.0 session after every 60 minutes."

Table 2: Old AppStream 2.0 relay state region endpoints

Region	Relay state endpoint
US East (N. Virginia)	`https://appstream2.us-east-1.aws.amazon.com/saml` (FIPS) `https://appstream2-fips.us-east-1.aws.amazon.com/saml`
US East (Ohio)	`https://appstream2.us-east-2.aws.amazon.com/saml`
US West (Oregon)	`https://appstream2.us-west-2.aws.amazon.com/saml` (FIPS) `https://appstream2-fips.us-west-2.aws.amazon.com/saml`
Asia Pacific (Mumbai)	`https://appstream2.ap-south-1.aws.amazon.com/saml`
Asia Pacific (Seoul)	`https://appstream2.ap-northeast-2.aws.amazon.com/saml`
Asia Pacific (Singapore)	`https://appstream2.ap-southeast-1.aws.amazon.com/saml`
Asia Pacific (Sydney)	`https://appstream2.ap-southeast-2.aws.amazon.com/saml`

Region	Relay state endpoint
Asia Pacific (Tokyo)	`https://appstream2.ap-northeast-1.aws.amazon.com/saml`
Canada (Central)	`https://appstream2.ca-central-1.aws.amazon.com/saml`
Europe (Frankfurt)	`https://appstream2.eu-central-1.aws.amazon.com/saml`
Europe (Ireland)	`https://appstream2.eu-west-1.aws.amazon.com/saml`
Europe (London)	`https://appstream2.eu-west-2.aws.amazon.com/saml`
AWS GovCloud (US-East)	`https://appstream2.us-gov-east-1.amazonaws-us-gov.com/saml` (FIPS) `https://appstream2-fips.us-gov-east-1.amazonaws-us-gov.com/saml` **Note** For more information about using AppStream 2.0 in AWS GovCloud (US) Regions, see Amazon AppStream 2.0 in the *AWS GovCloud (US) User Guide*.

Region	Relay state endpoint
AWS GovCloud (US-West)	`https://appstream2.us-gov-west-1.amazonaws-us-gov.com/saml` (FIPS) `https://appstream2-fips.us-gov-west-1.amazonaws-us-gov.com/saml` **Note** For more information about using AppStream 2.0 in AWS GovCloud (US) Regions, see Amazon AppStream 2.0 in the *AWS GovCloud (US) User Guide*.
South America (São Paulo)	`https://appstream2.sa-east-1.aws.amazon.com/saml`

Table 3 below lists all of the available parameters that you can use to construct a relay state URL.

Table 3: Relay state URL parameters

Parameter	Required	Format	Supported by
accountId	Required	12-character AWS account ID	New and old endpoints in Table 1 and 2
stack	Optional	Stack name	New and old endpoints in Table 1 and 2
app	Optional	App name or "Desktop"	New and old endpoints in Table 1 and 2
client	Optional	“native" or "web"	New endpoints in Table 1 only

AppStream 2.0 Integration with SAML 2.0

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

IdP solution	More information
AWS IAM Identity Center	Enable federation with IAM Identity Center and Amazon AppStream 2.0 — Describes how to use IAM Identity Center to federate user access to your AppStream 2.0 applications with their existing enterprise credentials.
Active Directory Federation Services (AD FS) for Windows Server	Enabling Identity Federation with AD FS 3.0 and Amazon AppStream 2.0 — Describes how to provide users with SSO access to AppStream 2.0 by using their existing enterprise credentials. You can configure federated identities for AppStream 2.0 by using AD FS 3.0.
Azure Active Directory (Azure AD)	Enabling Federation with Azure AD Single Sign-On and Amazon AppStream 2.0 — Describes how to configure federated user access for Amazon AppStream 2.0 by using Azure AD SSO for enterprise applications.
GG4L School Passport™	Enabling Identity Federation with GG4L's School Passport™ and Amazon AppStream 2.0 — Describes how to configure GG4L's School Passport™ to federate login to AppStream 2.0.
Google	Setting up G Suite SAML 2.0 federation with Amazon AppStream 2.0 — Describes how to use the G Suite Admin console to set up SAML federation to AppStream 2.0 for users in G Suite domains.
Okta	How to Configure SAML 2.0 for Amazon AppStream 2.0 — Describes how to use Okta to set up SAML federation to AppStream 2.0. For stacks that are joined to a domain, the "Application username format" must be set to "AD user principal name".

IdP solution	More information
Ping Identity	Configuring an SSO connection to Amazon AppStream 2.0 — Describes how to set up single sign-on (SSO) to AppStream 2.0.
Shibboleth	Single Sign-On: Integrating AWS, OpenLDAP, and Shibboleth — Describes how to set up the initial federatio n between the Shibboleth IdP and the AWS Managemen t Console. You must complete the following additional steps to enable federation to AppStream 2.0. Step 4 of the AWS Security whitepaper describes how to create IAM roles that define the permissions that federated users have to the AWS Management Console. After you create these roles and embed the inline policy as described in the whitepaper, modify this policy so that it provides federated users with permissions to access only an AppStream 2.0 stack. To do this, replace the existing policy with the policy noted in *Step 3: Embed an Inline Policy for the IAM Role*, in Setting Up SAML. When you add the stack relay state URL as described in *Step 6: Configure the Relay State of Your Federation*, in Setting Up SAML, add the relay state parameter to the federation URL as a target request attribute. The URL must be encoded. For information about configuring relay state parameters, see the SAML 2.0 section in the Shibboleth documentation. For more information, see Enabling Identity Federation with Shibboleth and Amazon AppStream 2.0.
VMware WorkSpace ONE	Federating Access to Amazon AppStream 2.0 from VMware Workspace ONE — Describes how to use the VMware Workspace ONE platform to federate user access to your AppStream 2.0 applications.

IdP solution	More information
SimpleSAMLphp	Enabling Federation with SimpleSAMLphp and Amazon AppStream 2.0 — Describes how to configure SAML 2.0 federation for AppStream 2.0 using SimpleSAMLphp.
OneLogin Single Sign-On (SSO)	OneLogin SSO with Amazon AppStream 2.0 — Describes how to configure federated user access for AppStream 2.0 using OneLogin SSO.
JumpCloud Single Sign-On (SSO)	Enable federation with JumpCloud SSO and Amazon AppStream 2.0 — Describes how to configure federated user access for AppStream 2.0 using JumpCloud SSO.
BIO-key PortalGuard	Enable federation with Bio-key PortalGuard and Amazon AppStream 2.0 — Describes how to configure BIO-key PortalGuard for federated logins to AppStream 2.0.

For solutions to common problems you may encounter, see Troubleshooting.

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

Using Active Directory with AppStream 2.0

You can join your Amazon AppStream 2.0 Always-On and On-Demand Windows fleets and image builders to domains in Microsoft Active Directory and use your existing Active Directory domains, either cloud-based or on-premises, to launch domain-joined streaming instances. You can also use AWS Directory Service for Microsoft Active Directory, also known as AWS Managed Microsoft AD, to create an Active Directory domain and use that to support your AppStream 2.0 resources. For more information about using AWS Managed Microsoft AD, see Microsoft Active Directory in the *AWS Directory Service Administration Guide*.

> **Note**
>
> Linux fleets, image builders, elastic fleets, and app block builders currently do not support domain join.

By joining AppStream 2.0 to your Active Directory domain, you can:

- Allow your users and applications to access Active Directory resources such as printers and file shares from streaming sessions.
- Use Group Policy settings that are available in the Group Policy Management Console (GPMC) to define the end user experience.
- Stream applications that require users to be authenticated using their Active Directory login credentials.
- Apply your enterprise compliance and security policies to your AppStream 2.0 streaming instances.

Contents

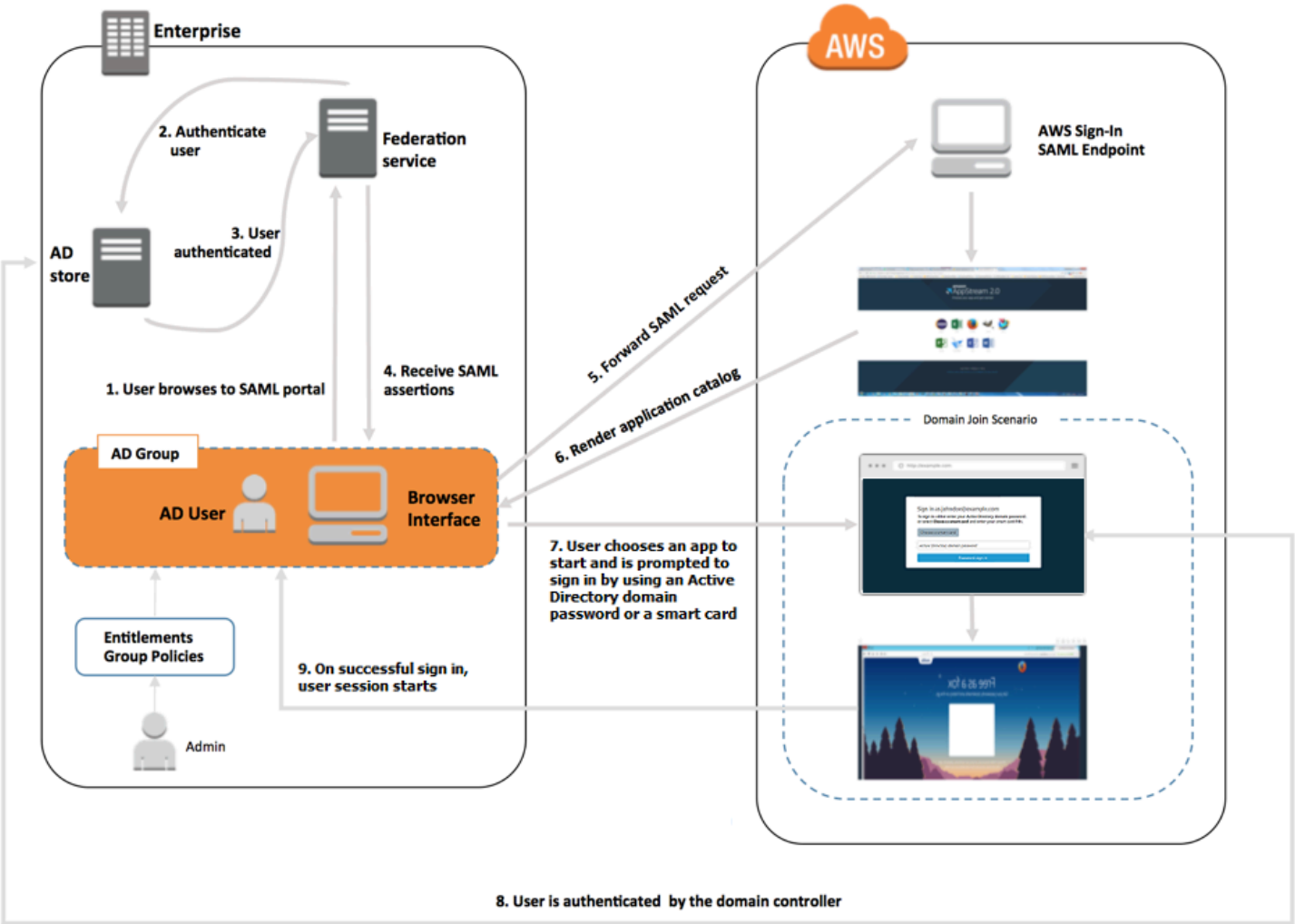
- Overview of Active Directory Domains
- Before You Begin Using Active Directory with AppStream 2.0
- Tutorial: Setting Up Active Directory
- Certificate-Based Authentication
- AppStream 2.0 Active Directory Administration
- More Info

Overview of Active Directory Domains

Using Active Directory domains with AppStream 2.0 requires an understanding of how they work together and the configuration tasks that you'll need to complete. You'll need to complete the following tasks:

1. Configure Group Policy settings as needed to define the end user experience and security requirements for applications.
2. Create the domain-joined application stack in AppStream 2.0.
3. Create the AppStream 2.0 application in the SAML 2.0 identity provider and assign it to end users either directly or through Active Directory groups.

For your users to be authenticated to a domain, several steps must occur when these users initiate an AppStream 2.0 streaming session. The following diagram illustrates the end-to-end user authentication flow from the initial browser request through SAML and Active Directory authentication.



User Authentication Flow

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.
7. The user chooses an application and, depending on the Windows login authentication method that is enabled on the AppStream 2.0 stack, they're prompted to enter their Active Directory domain password or choose a smart card. If both authentication methods are enabled, the user can choose whether to enter their domain password or use their smart card. Certificate-based authentication can also be used to authenticate users, removing the prompt.
8. The domain controller is contacted for user authentication.
9. After being authenticated with the domain, the user's session starts with domain connectivity.

From the user's perspective, this process is transparent. The user starts by navigating to your organization's internal portal and is redirected to an AppStream 2.0 application portal, without having to enter AWS credentials. Only an Active Directory domain password or smart card credentials are required.

Before a user can initiate this process, you must configure Active Directory with the required entitlements and Group Policy settings and create a domain-joined application stack.

Before You Begin Using Active Directory with AppStream 2.0

Before you use Microsoft Active Directory domains with AppStream 2.0, be aware of the following requirements and considerations.

Contents

- Active Directory Domain Environment
- Domain-Joined AppStream 2.0 Streaming Instances
- Group Policy Settings
- Smart Card Authentication

Active Directory Domain Environment

- You must have a Microsoft Active Directory domain to which to join your streaming instances. If you don't have an Active Directory domain or you want to use your on-premises Active Directory environment, see Active Directory Domain Services on the AWS Cloud: Quick Start Reference Deployment.

- You must have a domain service account with permissions to create and manage computer objects in the domain that you intend to use with AppStream 2.0. For information, see How to Create a Domain Account in Active Directory in the Microsoft documentation.

  When you associate this Active Directory domain with AppStream 2.0, provide the service account name and password. AppStream 2.0 uses this account to create and manage computer objects in the directory. For more information, see Granting Permissions to Create and Manage Active Directory Computer Objects.
- When you register your Active Directory domain with AppStream 2.0, you must provide an organizational unit (OU) distinguished name. Create an OU for this purpose. The default Computers container is not an OU and cannot be used by AppStream 2.0. For more information, see Finding the Organizational Unit Distinguished Name.
- The directories that you plan to use with AppStream 2.0 must be accessible through their fully qualified domain names (FQDNs) through the virtual private cloud (VPC) in which your streaming instances are launched. For more information, see Active Directory and Active Directory Domain Services Port Requirements in the Microsoft documentation.

Domain-Joined AppStream 2.0 Streaming Instances

SAML 2.0-based user federation is required for application streaming from domain-joined Always-On and On-Demand fleets. You cannot launch sessions to domain-joined instances by using CreateStreamingURL or the AppStream 2.0 user pool.

Also, you must use an image that supports joining image builders and fleets to an Active Directory domain. All public images published on or after July 24, 2017 support joining an Active Directory domain. For more information, see AppStream 2.0 Base Image and Managed Image Update Release Notes and Tutorial: Setting Up Active Directory.

> **Note**
>
> You can only join Windows Always-On and On-Demand fleet streaming instances to an Active Directory domain.

Group Policy Settings

Verify your configuration for the following Group Policy settings. If required, update the settings as described in this section so that they don't block AppStream 2.0 from authenticating and logging in your domain users. Otherwise, when your users try to log in to AppStream 2.0 the login may not succeed. Instead, a message displays, notifying users that "An unknown error occurred."

- **Computer Configuration > Administrative Templates > Windows Components > Windows Logon Options > Disable or Enable software Secure Attention Sequence** — Set this to **Enabled** for **Services**.
- **Computer Configuration > Administrative Templates > System > Logon > Exclude credential providers** — Ensure that the following CLSID is *not* listed: `e7c1bab5-4b49-4e64-a966-8d99686f8c7c`
- **Computer Configuration > Policies > Windows Settings > Security Settings > Local Policies > Security Options > Interactive Logon > Interactive Logon: Message text for users attempting to log on** — Set this to **Not defined**.
- **Computer Configuration > Policies > Windows Settings > Security Settings > Local Policies > Security Options > Interactive Logon > Interactive Logon: Message title for users attempting to log on** — Set this to **Not defined**.

Smart Card Authentication

AppStream 2.0 supports the use of Active Directory domain passwords or smart cards such as Common Access Card (CAC) and Personal Identity Verification (PIV) smart cards for Windows sign in to AppStream 2.0 streaming instances. For information about how to configure your Active Directory environment to enable smart card sign in by using third-party certification authorities (CAs), see Guidelines for enabling smart card logon with third-party certification authorities in the Microsoft documentation.

> **Note**
>
> AppStream 2.0 also supports the use of smart cards for in-session authentication after a user signs in to a streaming instance. This feature is supported only for users who have AppStream 2.0 client for Windows version 1.1.257 or later installed. For information about additional requirements, see Smart Cards.

Tutorial: Setting Up Active Directory

To use Active Directory with AppStream 2.0, you must first register your directory configuration by creating a Directory Config object in AppStream 2.0. This object includes the information required to join streaming instances to an Active Directory domain. You create a Directory Config object by using the AppStream 2.0 management console, AWS SDK, or AWS CLI. You can then use your directory configuration to launch domain-joined Always-On and On-Demand fleets and image builders.

> **Note**
>
> You can only join Always-On and On-Demand fleet streaming instances to an Active Directory domain.

Tasks

- Step 1: Create a Directory Config Object
- Step 2: Create an Image by Using a Domain-Joined Image Builder
- Step 3: Create a Domain-Joined Fleet
- Step 4: Configure SAML 2.0

Step 1: Create a Directory Config Object

The Directory Config object that you create in AppStream 2.0 will be used in later steps.

If you are using the AWS SDK, you can use the CreateDirectoryConfig operation. If you are using the AWS CLI, you can use the create-directory-config command.

To create a Directory Config object by using the AppStream 2.0 console

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. In the navigation pane, choose **Directory Configs**, **Create Directory Config**.
3. For **Directory Name**, provide the fully qualified domain name (FQDN) of the Active Directory domain (for example, `corp.example.com`). Each region can have only one **Directory Config** value with a specific directory name.
4. For **Service Account Name**, enter the name of an account that can create computer objects and that has permissions to join the domain. For more information, see Granting Permissions

to Create and Manage Active Directory Computer Objects. The account name must be in the format `DOMAIN\username`.

5. For **Password** and **Confirm Password**, type the directory password for the specified account.
6. For **Organizational Unit (OU)**, type the distinguished name of at least one OU for streaming instance computer objects.

> **Note**
>
> The OU name can't contain spaces. If you specify an OU name that contains spaces, when a fleet or image builder attempts to rejoin the Active Directory domain, AppStream 2.0 cannot cycle the computer objects correctly and the domain rejoin does not succeed. For information about how to troubleshoot this issue, see the *DOMAIN_JOIN_INTERNAL_SERVICE_ERROR* topic for "The account already exists" message in Active Directory Domain Join.
> In addition, the default Computers container is not an OU and cannot be used by AppStream 2.0. For more information, see Finding the Organizational Unit Distinguished Name.

7. To add more than one OU, select the plus sign (**+**) next to **Organizational Unit (OU)**. To remove OUs, choose the **x** icon.
8. Choose **Next**.
9. Review the configuration information and choose **Create**.

Step 2: Create an Image by Using a Domain-Joined Image Builder

Next, using the AppStream 2.0 image builder, create a new image with Active Directory domain-join capabilities. Note that the fleet and image can be members of different domains. You join the image builder to a domain to enable domain join and to install applications. Fleet domain join is discussed in the next section.

To create an image for launching domain-joined fleets

1. Follow the procedures in Tutorial: Create a Custom AppStream 2.0 Image by Using the AppStream 2.0 Console.

2. For the base image selection step, use an AWS base image released on or after July 24, 2017. For a current list of released AWS images, see AppStream 2.0 Base Image and Managed Image Update Release Notes.
3. For **Step 3: Configure Network**, select a VPC and subnets with network connectivity to your Active Directory environment. Select the security groups that are set up to allow access to your directory through your VPC subnets.
4. Also in **Step 3: Configure Network**, expand the **Active Directory Domain (Optional)** section, and select values for the **Directory Name** and **Directory OU** to which the image builder should be joined.
5. Review the image builder configuration and choose **Create**.
6. Wait for the new image builder to reach a **Running** state, and choose **Connect**.
7. Log in to the image builder in Administrator mode or as a directory user with local administrator permissions. For more information, see Granting Local Administrator Rights on Image Builders.
8. Complete the steps in Tutorial: Create a Custom AppStream 2.0 Image by Using the AppStream 2.0 Console to install applications and create a new image.

Step 3: Create a Domain-Joined Fleet

Using the private image created in the previous step, create an Active Directory domain-joined Always-On or On-Demand fleet for streaming applications. The domain can be different than the one that you used for the image builder to create the image.

To create a domain-joined Always-On or On-Demand fleet

1. Follow the procedures in Create a Fleet.
2. For the image selection step, use the image that was created in the previous step, Step 2: Create an Image by Using a Domain-Joined Image Builder.
3. For **Step 4: Configure Network**, select a VPC and subnets with network connectivity to your Active Directory environment. Select the security groups that are set up to allow communication to your domain.
4. Also in **Step 4: Configure Network**, expand the **Active Directory Domain (Optional)** section and select the values for the **Directory Name** and **Directory OU** to which the fleet should be joined.
5. Review the fleet configuration and choose **Create**.

6. Complete the remaining steps in Create an AppStream 2.0 Fleet and Stack so that your fleet is associated with a stack and running.

Step 4: Configure SAML 2.0

Your users must use your SAML 2.0-based identity federation environment to launch streaming sessions from your domain-joined fleet.

To configure SAML 2.0 for single sign-on access

1. Follow the procedures in Setting Up SAML.
2. AppStream 2.0 requires that the SAML_Subject `NameID` value for the user who is logging in be provided in either of the following formats:

   - *`domain`*`\username` using the sAMAccountName
   - `username@domain.com` using the userPrincipalName

   If you are using the sAMAccountName format, you can specify the *`domain`* by using either the NetBIOS name or the fully qualified domain name (FQDN).
3. Provide access to your Active Directory users or groups to enable access to the AppStream 2.0 stack from your identity provider application portal.
4. Complete the remaining steps in Setting Up SAML.

To log in a user with SAML 2.0

1. Log in to your SAML 2.0 provider's application catalog and open the AppStream 2.0 SAML application that you created in the previous procedure.
2. When the AppStream 2.0 application catalog is displayed, select an application to launch.
3. When a loading icon is displayed, you are prompted to provide a password. The domain user name provided by your SAML 2.0 identity provider appears above the password field. Enter your password, and choose **log in**.

The streaming instance performs the Windows login procedure, and the selected application opens.

Certificate-Based Authentication

You can use certificate-based authentication with AppStream 2.0 fleets joined to Microsoft Active Directory. This removes the user prompt for the Active Directory domain password when a user logs in. By using certificate-based authentication with your Active Directory domain, you can:

- Rely on your SAML 2.0 identity provider to authenticate the user and provide SAML assertions to match the user in Active Directory.
- Create a single sign-on logon experience with fewer user prompts.
- Enable passwordless authentication flows using your SAML 2.0 identity provider.

Certificate-based authentication uses AWS Private Certificate Authority (AWS Private CA) resources in your AWS account. With AWS Private CA, you can create private certificate authority (CA) hierarchies, including root and subordinate CAs. You can also create your own CA hierarchy and issue certificates from it that authenticate internal users. For more information, see What is AWS Private CA.

When you use AWS Private CA for certificate-based authentication, AppStream 2.0 requests certificates for your users automatically at session reservation for each AppStream 2.0 fleet instance. It authenticates users to Active Directory with a virtual smart card provisioned with the certificates.

Certificate-based authentication is supported on AppStream 2.0 domain-joined fleets that run Windows instances. It is currently not supported for multi-session fleets.

Contents

- Prerequisites
- Enable Certificate-based Authentication
- Manage Certificate-based Authentication

Prerequisites

Complete the following steps before you use certificate-based authentication.

1. Set up a domain-joined fleet and configure SAML 2.0. Ensure that you use the `username@domain.com` `userPrincipalName` format for the SAML_Subject `NameID`. For

more information, see the section called "Step 5: Create Assertions for the SAML Authentication Response".

> **Note**
>
> Don't enable **Smart card sign in for Active Directory** in your stack if you want to use certificate-based authentication. For more information, see the section called "Smart Cards".

2. Use AppStream 2.0 agent version 10-13-2022 or later with your image. For more information, see the section called "Keep Your AppStream 2.0 Image Up-to-Date".
3. (Optional) Configure the `ObjectSid` attribute in your SAML assertion. You can use this attribute to perform strong mapping with the Active Directory user. Certificate-based authentication fails if the `ObjectSid` attribute doesn't match the Active Directory security identifier (SID) for the user specified in the SAML_Subject `NameID`. For more information, see the section called "Step 5: Create Assertions for the SAML Authentication Response".
4. Add the `sts:TagSession` permission to the IAM role trust policy that you use with your SAML 2.0 configuration. For more information, see Passing session tags in AWS STS. This permission is required to use certificate-based authentication. For more information, see the section called "Step 2: Create a SAML 2.0 Federation IAM Role".
5. Create a private certificate authority (CA) using AWS Private CA, if you don't have one configured with your Active Directory. AWS Private CA is required to use certificate-based authentication. For more information, see Planning your AWS Private CA deployment. The following AWS Private CA settings are common for many certificate-based authentication use cases:
   - **CA type options**
     - **Short-lived certificate CA usage mode** – Recommended if the CA only issues end user certificates for certificate-based authentication.
     - **Single level hierarchy with a Root CA** – Choose a subordinate CA to integrate it with an existing CA hierarchy.
   - **Key algorithm options** – RSA 2048
   - **Subject distinguished name options** – Use the most appropriate options to identify this CA in your Active Directory Trusted Root Certification Authorities store.
   - **Certificate revocation options** – CRL distribution

> **Note**
>
> Certificate-based authentication requires an online CRL distribution point accessible from both the AppStream 2.0 fleet instance and the domain controller. This requires unauthenticated access to the Amazon S3 bucket configured for AWS Private CA CRL entries, or a CloudFront distribution with access to the Amazon S3 bucket if it blocks public access. For more information about these options, see Planning a certificate revocation list (CRL).

6. Tag your private CA with a key entitled `euc-private-ca` to designate the CA for use with AppStream 2.0 certificate-based authentication. This key doesn't require a value. For more information, see Managing tags for your private CA. For more information about the AWS managed policies used with AppStream 2.0 to grant permissions to resources in your AWS account, see the section called "AWS Managed Policies Required to Access AppStream 2.0 Resources".
7. Certificate-based authentication uses virtual smart cards to log on. For more information, see Guidelines for enabling smart card logon with third-party certification authorities. Follow these steps:
   a. Configure domain controllers with a domain controller certificate to authenticate smart card users. If you have an Active Directory Certificate Services enterprise CA configured in your Active Directory, it automatically enrolls domain controllers with certificates that enable smart card logon. If you don't have Active Directory Certificate Services, see Requirements for domain controller certificates from a third-party CA. You can create a domain controller certificate with AWS Private CA. If you do this, don't use a private CA configured for short-lived certificates.

> **Note**
>
> If you use AWS Managed Microsoft AD, you can configure Certificate Services on an Amazon EC2 instance that satisfies the requirement for domain controller certificates. See Deploy Active Directory to a new Amazon Virtual Private Cloud for example deployments of AWS Managed Microsoft AD configured with Active Directory Certificate Services.
> With AWS Managed Microsoft AD and Active Directory Certificate Services, you must also create outbound rules from the controller's VPC security group to the Amazon EC2 instance running Certificate Services. You must provide the security group

access to TCP port 135, and ports 49152 through 65535 to enable certificate auto-enrollment. The Amazon EC2 instance must also allow inbound access on these same ports from domain instances, including domain controllers. For more information on locating the security group for AWS Managed Microsoft AD, see Configure your VPC subnets and security groups.

b. On the AWS Private CA console, or with the SDK or CLI, export the private CA certificate. For more information, see Exporting a private certificate.

c. Publish the private CA to Active Directory. Log on to a domain controller or a domain-joined machine. Copy the private CA certificate to any *<path>**<file>* and run the following commands as a domain administrator. You can also use Group Policy and the Microsoft PKI Health Tool (PKIView) to publish the CA. For more information, see Configuration instructions.

```
certutil -dspublish -f <path>\<file> RootCA
```

```
certutil -dspublish -f <path>\<file> NTAuthCA
```

Make sure that the commands complete successfully, then remove the private CA certificate file. Depending on your Active Directory replication settings, it can take several minutes for the CA to publish to your domain controllers and AppStream 2.0 fleet instances.

> **Note**
>
> Active Directory must distribute the CA to the Trusted Root Certification Authorities and Enterprise NTAuth stores automatically for AppStream 2.0 fleet instances when they join the domain.

> **Note**
>
> Active Directory domain controllers must be in Compatibility mode for certificate strong enforcement to support certificate-based authentication. For more information, see KB5014754—Certificate-based authentication changes on Windows domain controllers in the Microsoft Support documentation. If you are using AWS Managed Microsoft AD, see Configure directory security settings for more information.

Enable Certificate-based Authentication

Complete the following steps to enable certificate-based authentication.

To enable certificate-based authentication

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. In the navigation pane, choose **Directory Configs**. Select the directory config you want to configure, and choose **Edit**.
3. Choose **Enable Certificate-Based Authentication.**
4. Confirm that your private CA ARN is associated in the list. To appear in the list, you should store the private CA in the same AWS account and AWS Region. You must also tag the private CA with a key named `euc-private-ca`.
5. Configure directory log in fallback. With Fallback, users can log in with their AD domain password if certificate-based authentication is unsuccessful. This is recommended only in cases where users know their domain passwords. When fallback is turned off, a session can disconnect the user if a lock screen or Windows log off occurs. If fallback is turned on, the session prompts the user for their AD domain password.
6. Choose **Save Changes**.
7. Certificate-based authentication is now enabled. When users authenticate with SAML 2.0 to an AppStream 2.0 stack using the domain-joined fleet from the AppStream 2.0 web client or the client for Windows (version 1.1.1099 and later), they will no longer receive a prompt for the domain password. Users will see a “Connecting with certificate-based authentication...” message when connecting to a session enabled for certificate-based authentication.

Manage Certificate-based Authentication

After you enable certificate-based authentication, review the following tasks.

Private CA Certificate

In a typical configuration, the private CA certificate has a validity period of 10 years. For more information about replacing a private CA with an expired certificate, or reissuing the private CA with a new validity period, see Managing the private CA lifecycle

End User Certificates

End user certificates issued by AWS Private CA for AppStream 2.0 certificate-based authentication don't require renewal or revocation. These certificates are short-lived. AppStream 2.0 automatically issues a new certificate for each new session, or every 24 hours for sessions with a long duration. The AppStream 2.0 session governs the use of these end user certificates. If you end a session, AppStream 2.0 stops using that certificate. These end user certificates have a shorter validity period than a typical AWS Private CA CRL distribution. As a result, end user certificates don't need to be revoked and won't appear in a CRL.

Audit Reports

You can create an audit report to list all of the certificates that your private CA has issued or revoked. For more information, see Using audit reports with your private CA.

Logging and Monitoring

You can use CloudTrail to record API calls to a private CA by AppStream 2.0. For more information see What Is AWS CloudTrail? and Using CloudTrail. In CloudTrail Event history you can view **GetCertificate** and **IssueCertificate** event names from **acm-pca.amazonaws.com** event source made by the AppStream 2.0 **EcmAssumeRoleSession** user name. These events will be recorded for every AppStream 2.0 certificate-based authentication request. For more information, see Viewing events with CloudTrail Event history.

AppStream 2.0 Active Directory Administration

Setting up and using Active Directory with AppStream 2.0 involves the following administrative tasks.

Tasks

- Granting Permissions to Create and Manage Active Directory Computer Objects
- Finding the Organizational Unit Distinguished Name
- Granting Local Administrator Rights on Image Builders
- Updating the Service Account Used for Joining the Domain
- Locking the Streaming Session When the User is Idle
- Editing the Directory Configuration

- Deleting a Directory Configuration
- Configuring AppStream 2.0 to Use Domain Trusts
- Managing AppStream 2.0 Computer Objects in Active Directory

Granting Permissions to Create and Manage Active Directory Computer Objects

To allow AppStream 2.0 to perform Active Directory computer object operations, you need an account with sufficient permissions. As a best practice, use an account that has only the minimum privileges necessary. The minimum Active Directory organizational unit (OU) permissions are as follows:

- Create Computer Object
- Change Password
- Reset Password
- Write Description

Before setting up permissions, you'll need to do the following first:

- Obtain access to a computer or an EC2 instance that is joined to your domain.
- Install the Active Directory User and Computers MMC snap-in. For more information, see Installing or Removing Remote Server Administration Tools for Windows 7 in the Microsoft documentation.
- Log in as a domain user with appropriate permissions to modify the OU security settings.
- Create or identify the user, service account, or group for which to delegate permissions.

To set up minimum permissions

1. Open **Active Directory Users and Computers** in your domain or on your domain controller.
2. In the left navigation pane, select the first OU on which to provide domain join privileges, open the context (right-click) menu , and then choose **Delegate Control**.
3. On the **Delegation of Control Wizard** page, choose **Next**, **Add**.
4. For **Select Users, Computers, or Groups**, select the pre-created user, service account, or group, and then choose **OK**.

5. On the **Tasks to Delegate** page, choose **Create a custom task to delegate**, and then choose **Next**.
6. Choose **Only the following objects in the folder**, **Computer objects**.
7. Choose **Create selected objects in this folder**, **Next**.
8. For **Permissions**, choose **Read**, **Write**, **Change Password**, **Reset Password**, **Next**.
9. On the **Completing the Delegation of Control Wizard** page, verify the information and choose **Finish**.
10. Repeat steps 2-9 for any additional OUs that require these permissions.

If you delegated permissions to a group, create a user or service account with a strong password and add that account to the group. This account will then have sufficient privileges to connect your streaming instances to the directory. Use this account when creating your AppStream 2.0 directory configuration.

Finding the Organizational Unit Distinguished Name

When you register your Active Directory domain with AppStream 2.0, you must provide an organizational unit (OU) distinguished name. Create an OU for this purpose. The default Computers container is not an OU and cannot be used by AppStream 2.0. The following procedure shows how to obtain this name.

> **Note**
>
> The distinguished name must start with **OU=** or it cannot be used for computer objects.

Before you complete this procedure, you'll need to do the following first:

- Obtain access to a computer or an EC2 instance that is joined to your domain.
- Install the Active Directory User and Computers MMC snap-in. For more information, see Installing or Removing Remote Server Administration Tools for Windows 7 in the Microsoft documentation.
- Log in as a domain user with appropriate permissions to read the OU security properties.

To find the distinguished name of an OU

1. Open **Active Directory Users and Computers** in your domain or on your domain controller.
2. Under **View**, ensure that **Advanced Features** is enabled.
3. In the left navigation pane, select the first OU to use for AppStream 2.0 streaming instance computer objects, open the context (right-click) menu, and then choose **Properties**.
4. Choose **Attribute Editor**.
5. Under **Attributes**, for **distinguishedName**, choose **View**.
6. For **Value**, select the distinguished name, open the context menu, and then choose **Copy**.

Granting Local Administrator Rights on Image Builders

By default, Active Directory domain users do not have local administrator rights on image builder instances. You can grant these rights by using Group Policy preferences in your directory, or manually, by using the local administrator account on an image builder. Granting local administrator rights to a domain user allows that user to install applications on and create images in an AppStream 2.0 image builder.

Contents

- Using Group Policy preferences
- Using the local Administrators group on the image builder

Using Group Policy preferences

You can use Group Policy preferences to grant local administrator rights to Active Directory users or groups and to all computer objects in the specified OU. The Active Directory users or groups to which you want to grant local administrator permissions must already exist. To use Group Policy preferences, you'll need to do the following first:

- Obtain access to a computer or an EC2 instance that is joined to your domain.
- Install the Group Policy Management Console (GPMC) MMC snap-in. For more information, see Installing or Removing Remote Server Administration Tools for Windows 7 in the Microsoft documentation.
- Log in as a domain user with permissions to create Group Policy objects (GPOs). Link GPOs to the appropriate OUs.

To use Group Policy preferences to grant local administrator permissions

1. In your directory or on a domain controller, open the command prompt as an administrator, type `gpmc.msc`, and then press ENTER.
2. In the left console tree, select the OU where you will create a new GPO or use an existing GPO, and then do either of the following:
   - Create a new GPO by opening the context (right-click) menu and choosing **Create a GPO in this domain, Link it here**. For **Name**, provide a descriptive name for this GPO.
   - Select an existing GPO.
3. Open the context menu for the GPO, and choose **Edit**.
4. In the console tree, choose **Computer Configuration**, **Preferences**, **Windows Settings**, **Control Panel Settings**, and **Local Users and Groups**.
5. Select **Local Users and Groups** selected, open the context menu , and choose **New**, **Local Group**.
6. For **Action**, choose **Update**.
7. For **Group name**, choose **Administrators (built-in)**.
8. Under **Members**, choose **Add...** and specify the Active Directory users or groups to which to assign local administrator rights on the streaming instance. For **Action**, choose **Add to this group**, and choose **OK**.
9. To apply this GPO to other OUs, select the additional OU, open the context menu and choose **Link an Existing GPO**.
10. Using the new or existing GPO name that you specified in step 2, scroll to find the GPO, and then choose **OK**.
11. Repeat steps 9 and 10 for additional OUs that should have this preference.
12. Choose **OK** to close the **New Local Group Properties** dialog box.
13. Choose **OK** again to close the GPMC.

To apply the new preference to the GPO, you must stop and restart any running image builders or fleets. The Active Directory users and groups that you specified in step 8 are automatically granted local administrator rights on the image builders and fleets in the OU to which the GPO is linked.

Using the local Administrators group on the image builder

To grant Active Directory users or groups local administrator rights on your image builder, you can manually add these users or groups to the local Administrators group on the image builder. Image builders that are created from images with these rights maintain the same rights.

The Active Directory users or groups to which to grant local administrator rights must already exist.

To add Active Directory users or groups to the local Administrators group on the image builder

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. Connect to the image builder in Administrator mode. The image builder must be running and domain-joined. For more information, see Tutorial: Setting Up Active Directory.
3. Choose **Start**, **Administrative Tools**, and then double-click **Computer Management**.
4. In the left navigation pane, choose **Local Users and Groups** and open the **Groups** folder.
5. Open the **Administrators** group and choose **Add...**.
6. Select all Active Directory users or groups to which to assign local administrator rights and choose **OK**. Choose **OK** again to close the **Administrator Properties** dialog box.
7. Close Computer Management.
8. To log in as an Active Directory user and test whether that user has local administrator rights on the image builder, choose **Admin Commands**, **Switch user**, and then enter the credentials of the relevant user.

Updating the Service Account Used for Joining the Domain

To update the service account that AppStream 2.0 uses for joining the domain, we recommend using two separate service accounts for joining image builders and fleets to your Active Directory domain. Using two separate service accounts ensures that there is no disruption in service when a service account needs to be updated (for example, when a password expires).

To update a service account

1. Create an Active Directory group and delegate the correct permissions to the group.
2. Add your service accounts to the new Active Directory group.
3. When needed, edit your AppStream 2.0 Directory Config object by entering the sign-in credentials for the new service account.

After you've set up the Active Directory group with the new service account, any new streaming instance operations will use the new service account, while in-process streaming instance operations continue to use the old account without interruption.

The service account overlap time while the in-process streaming instance operations complete is very short, no more than a day. The overlap time is needed because you shouldn't delete or change the password for the old service account during the overlap period, or existing operations can fail.

Locking the Streaming Session When the User is Idle

AppStream 2.0 relies on a setting that you configure in the GPMC to lock the streaming session after your user is idle for specified amount of time. To use the GPMC, you'll need to do the following first:

- Obtain access to a computer or an EC2 instance that is joined to your domain.
- Install the GPMC. For more information, see Installing or Removing Remote Server Administration Tools for Windows 7 in the Microsoft documentation.
- Log in as a domain user with permissions to create GPOs. Link GPOs to the appropriate OUs.

To automatically lock the streaming instance when your user is idle

1. In your directory or on a domain controller, open the command prompt as an administrator, type `gpmc.msc`, and then press ENTER.
2. In the left console tree, select the OU where you will create a new GPO or use an existing GPO, and then do either of the following:
   - Create a new GPO by opening the context (right-click) menu and choosing **Create a GPO in this domain, Link it here**. For **Name**, provide a descriptive name for this GPO.
   - Select an existing GPO.
3. Open the context menu for the GPO, and choose **Edit**.
4. Under **User Configuration**, expand **Policies**, **Administrative Templates**, **Control Panel**, and then choose **Personalization**.
5. Double-click **Enable screen saver**.
6. In the **Enable screen saver** policy setting, choose **Enabled**.
7. Choose **Apply**, and then choose **OK**.
8. Double-click **Force specific screen saver**.

9. In the **Force specific screen saver** policy setting, choose **Enabled**.
10. Under **Screen saver executable name**, enter `scrnsave.scr`. When this setting is enabled, the system displays a black screen saver on the user's desktop.
11. Choose **Apply**, and then choose **OK**.
12. Double-click **Password protect the screen saver**.
13. In the **Password protect the screen saver** policy setting, choose **Enabled**.
14. Choose **Apply**, and then choose **OK**.
15. Double-click **Screen saver timeout**.
16. In the **Screen saver timeout** policy setting, choose **Enabled**.
17. For **Seconds**, specify the length of time that users must be idle before the screen saver is applied. To set the idle time to 10 minutes, specify 600 seconds.
18. Choose **Apply**, and then choose **OK**.
19. In the console tree, under **User Configuration**, expand **Policies**, **Administrative Templates**, **System**, and then choose **Ctrl+Alt+Del Options**.
20. Double-click **Remove Lock Computer**.
21. In the **Remove Lock Computer** policy setting, choose **Disabled**.
22. Choose **Apply**, and then choose **OK**.

Editing the Directory Configuration

After a AppStream 2.0 directory configuration has been created, you can edit it to add, remove, or modify organizational units, update the service account username, or update the service account password.

To update a directory configuration

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. In the left navigation pane, choose **Directory Configs** and select the directory configuration to edit.
3. Choose **Actions**, **Edit**.
4. Update the fields to be changed. To add additional OUs, select the plus sign (**+**) next to the topmost OU field. To remove an OU field, select the **x** next to the field.

> **Note**
>
> At least one OU is required. OUs that are currently in use cannot be removed.

5. To save changes, choose **Update Directory Config**.
6. The information in the **Details** tab should now update to reflect the changes.

Changes to the service account sign-in credentials do not impact in-process streaming instance operations. New streaming instance operations use the updated credentials. For more information, see Updating the Service Account Used for Joining the Domain.

Deleting a Directory Configuration

You can delete an AppStream 2.0 directory configuration that is no longer needed. Directory configurations that are associated with any image builders or fleets cannot be deleted.

To delete a directory configuration

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. In the left navigation pane, choose **Directory Configs** and select the directory configuration to delete.
3. Choose **Actions**, **Delete**.
4. Verify the name in the pop-up message, and choose **Delete**.
5. Choose **Update Directory Config**.

Configuring AppStream 2.0 to Use Domain Trusts

AppStream 2.0 supports Active Directory domain environments where network resources such as file servers, applications, and computer objects reside in one domain, and the user objects reside in another. The domain service account used for computer object operations does not need to be in the same domain as the AppStream 2.0 computer objects.

When creating the directory configuration, specify a service account that has the appropriate permissions to manage computer objects in the Active Directory domain where the file servers, applications, computer objects and other network resources reside.

Your end user Active Directory accounts must have the "Allowed to Authenticate" permissions for the following:

- AppStream 2.0 computer objects
- Domain controllers for the domain

For more information, see Granting Permissions to Create and Manage Active Directory Computer Objects.

Managing AppStream 2.0 Computer Objects in Active Directory

AppStream 2.0 does not delete computer objects from Active Directory. These computer objects can be easily identified in your directory. Each computer object in the directory is created with the `Description` attribute, which specifies a fleet or an image builder instance and the name.

Computer Object Description Examples

Type	Name	Description Attribute
Fleet	ExampleFleet	`AppStream 2.0 - fleet:ExampleFleet`
Image builder	ExampleImageBuilder	`AppStream 2.0 - image-builder:ExampleImageBuilder`

You can identify and delete inactive computer objects created by AppStream 2.0 by using the following `dsquery computer` and `dsrm` commands. For more information, see Dsquery computer and Dsrm in the Microsoft documentation.

The `dsquery` command identifies inactive computer objects over a certain period of time and uses the following format. The `dsquery` command should also be run with the parameter `-desc "AppStream 2.0*"` to display only AppStream 2.0 objects.

```
dsquery computer "OU-distinguished-name" -desc "AppStream 2.0*" -inactive number-of-weeks-since-last-login
```

- *`OU-distinguished-name`* is the distinguished name of the organizational unit. For more information, see Finding the Organizational Unit Distinguished Name. If you don't provide the *`OU-distinguished-name`* parameter, the command searches the entire directory.

- *number-of-weeks-since-last-log-in* is the desired value based on how you want to define inactivity.

For example, the following command displays all computer objects in the `OU=ExampleOU,DC=EXAMPLECO,DC=COM` organizational unit that have not been logged into within the past two weeks.

```
dsquery computer OU=ExampleOU,DC=EXAMPLECO,DC=COM -desc "AppStream 2.0*" -inactive 2
```

If any matches are found, the result is one or more object names. The `dsrm` command deletes the specified object and uses the following format:

```
dsrm objectname
```

Where *objectname* is the full object name from the output of the `dsquery` command. For example, if the `dsquery` command above results in a computer object named "ExampleComputer", the `dsrm` command to delete it would be as follows:

```
dsrm "CN=ExampleComputer,OU=ExampleOU,DC=EXAMPLECO,DC=COM"
```

You can chain these commands together by using the pipe (`|`) operator. For example, to delete all AppStream 2.0 computer objects, prompting for confirmation for each, use the following format. Add the `-noprompt` parameter to `dsrm` to disable confirmation.

```
dsquery computer OU-distinguished-name -desc "AppStream 2.0*" -inactive number-of-
weeks-since-last-log-in | dsrm
```

More Info

For more information related to this topic, see the following resources:

- Troubleshooting Notification Codes—Resolutions to notification code errors.
- Troubleshooting Active Directory —Help with common difficulties.
- Microsoft Active Directory—Information about using AWS Directory Service.

Add Your Custom Branding to Amazon AppStream 2.0

To create a familiar experience for your users when they stream applications, you can customize the appearance of AppStream 2.0 with your own branding images, text, and website links, and you can choose from one of several color palettes. When you customize AppStream 2.0, your branding is displayed to users during application streaming sessions rather than the default AppStream 2.0 branding.

Custom Branding Options

You can customize the appearance of the streaming application catalog page by using the following branding options.

> **Note**
>
> Custom branding is not available for the user pool sign-in portal or for the email notifications that AppStream 2.0 sends to user pool users.

Branding element	Description	Requirements and recommendations
Organization logo	Enables you to display an image that is familiar to your users. The image appears in the header of the streaming application catalog page, which is displayed to users after they sign in to AppStream 2.0.	File type: .png, jpg, .jpeg, or .gif Maximum dimensions: 1000 px x 500 px Maximum file size: 300 KB
Organization website links	Enables you to display links to helpful resources for your users,	Maximum number of links: 3 Format (URL): https://example.com or http://example.com

Branding element	Description	Requirements and recommendations
	such as your organizat ion's IT support and product marketing sites. The links are displayed in the footer of the streaming application catalog page.	Maximum length (display name): 100 letters, spaces, and numbers Special characters allowed (display name): @ . / # & + $
Color theme	Applied to website links, text, and buttons. These colors are also applied as accents in the background for the streaming application catalog page.	Predefined themes from which to choose: 4 For information about each color theme, see Color Theme Palettes later in this topic.
Page title	Displayed at the top of the browser tab during users' applicati on streaming sessions.	Maximum length: 200 letters, spaces, and numbers. Special characters allowed: @ . / # & + $
Favicon	Enables your users to recognize their applicati on streaming site in a browser full of tabs or bookmarks. The favicon icon is displayed at the top of the browser tab for the applicati on streaming site during users' streaming sessions.	File type: .png, .jpg, .jpeg, .gif, or .ico Maximum dimensions: 128 px x 128 px Maximum file size: 50 KB

Branding element	Description	Requirements and recommendations
Redirect URL	Enables you to specify a URL to which users are redirected when they end a streaming session.	Format: https://example.com or http://example.com This URL is configured in the **Details** page for a stack when you create or edit a stack, not in the **Branding** page.
Feedback URL	Enables you to specify a URL for a **Send Feedback** link, so that your users can submit feedback. If you do not specify a URL, the **Send Feedback** link is not displayed.	Format: https://example.com or http://example.com This URL is configured in the **Details** page for a stack when you create or edit a stack, not in the **Branding** page.

Adding Your Custom Branding to AppStream 2.0

To customize AppStream 2.0 with your organizational branding, use the AppStream 2.0 console to select the stack to customize, and then add your branding.

To add your custom branding to AppStream 2.0

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. In the left pane, choose **Stacks**.
3. In the stack list, select the stack to customize with your branding.
4. Choose **Branding**, **Custom**.
5. For **Application catalog page**, customize how the streaming application catalog page appears to users after they sign in to AppStream 2.0.

   a. For **Organization logo**, do either of the following:

      - If you are uploading a logo for the first time, choose **Upload**, and then select the image to display in the header of the streaming application catalog page.

- If you have already uploaded a logo and need to change it, choose **Change Logo**, and then select the image to display.

b. For **Organization website links**, specify up to three website links to display in the page footer. For each link, choose the **Add Link** button, and then enter a display name and URL. To add more links, repeat these steps for each link to add. To remove a link, choose the **Remove** button under the link URL.

c. For **Color theme**, choose the colors to use for your website links, body text, and buttons, and as an accent for the page background. For information about each color theme, see Color Theme Palettes later in this topic.

6. For **Browser tab**, customize the page title and icon to display to users at the top of their browser tab during streaming sessions.

   a. For **Page title**, enter the title to display at the top of the browser tab.

   b. For **Favicon**, do either of the following:

      - If you are uploading a favicon for the first time, choose **Upload**, and then select the image to display at the top of the browser tab.
      - If you have already uploaded a favicon and want to change it, choose **Change Logo**, and then select the image to display.

7. Do either of the following:

   - To apply your branding changes, choose **Save**. When users connect to new streaming sessions that are launched for the stack, your branding changes are displayed.

     > **Note**
     >
     > AppStream 2.0 retains the custom branding changes that you save. If you save your custom branding changes, but then choose to restore the AppStream 2.0 default branding, your custom branding changes are saved for later use. If you restore the AppStream 2.0 default branding and decide later to reapply your custom branding, choose **Custom**, **Save**. In this case, the most recently saved custom branding is displayed to your users.

   - To discard your branding changes, choose **Cancel**. When prompted to confirm your choice, choose **Confirm**. If you cancel your changes, the most recently saved branding is displayed to your users.

Specifying a Custom Redirect URL and Feedback URL

You can specify a URL to which your users are redirected when they end their streaming session, as well as a URL where your users can submit feedback. By default, AppStream 2.0 displays a **Send Feedback** link that enables users to submit feedback to AWS about the quality of their application streaming session. To enable your users to submit feedback to a site that you specify, you can provide a custom feedback URL. You can specify the redirect URL and feedback URL when you create a new stack or edit the details for an existing stack. For more information, see Create a Stack.

Previewing Your Custom Branding Changes

You can preview how your branding changes will appear to your users by applying your branding changes to a test stack before you apply them to a production stack, and then creating a streaming URL for the test stack. After you validate your branding changes, you can them deploy them to your production stack. For information, see Step 2: Provide Access to Users in *Getting Started with Amazon AppStream 2.0*.

Color Theme Palettes

When you choose a color theme, the colors for that theme are applied to the website links, text, and buttons in your streaming application catalog page. A color is also applied as an accent in the background for your streaming application catalog page. For each color in a color theme palette, the hex value is also noted.

Color Themes

- Red
- Light Blue
- Blue
- Pink

Red

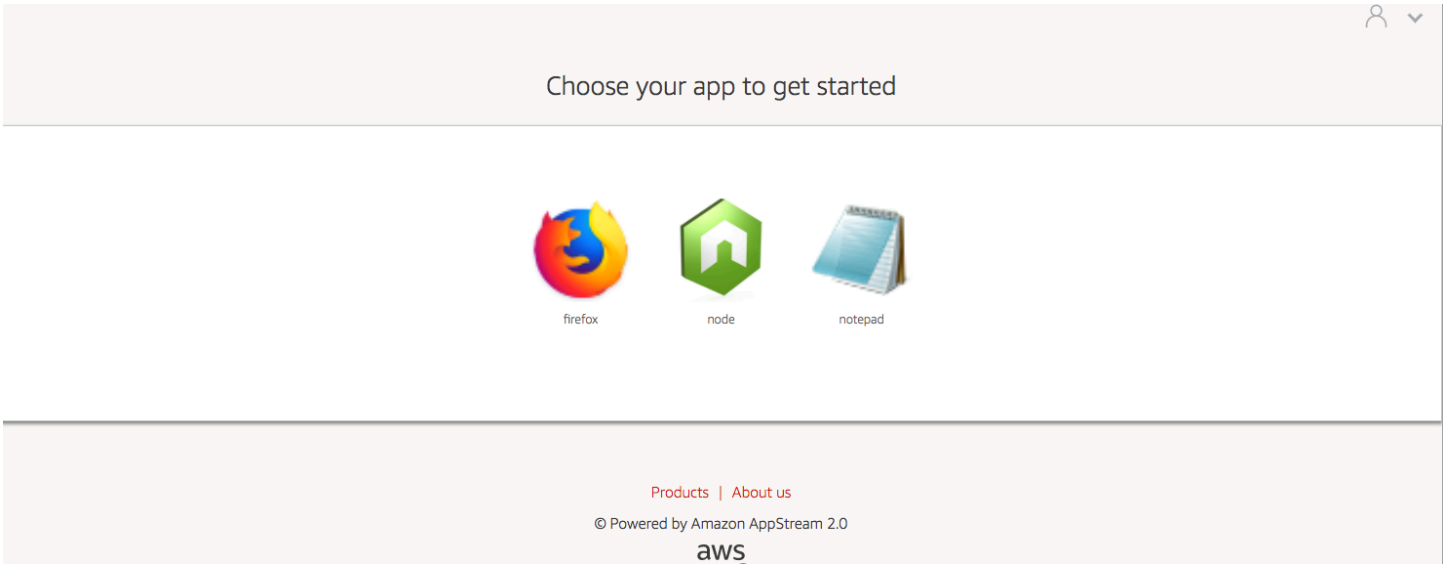
The following colors are applied when you select the red color theme.

Red (#d51900) – Used for buttons and website links.

White (#faf9f7) – Used as a background accent.

Dark grey (#404040) – Used for the body text and in the progress spinner.

When you choose the red color theme, the website links, body text, and background accent appear in your streaming application catalog page as follows.



Light Blue

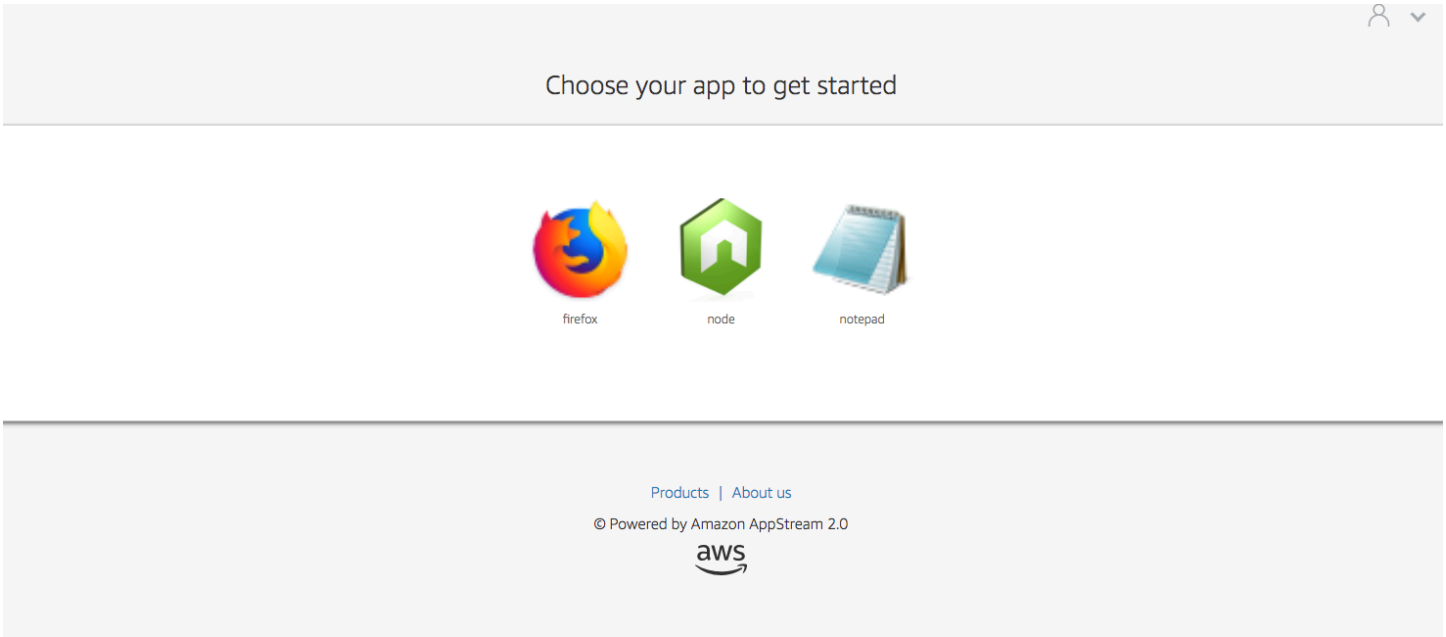
The following colors are applied when you select the light blue color theme:

Light blue (#1d83c2) – Used for buttons and website links.

White (#f6f6f6) – Used as a background accent.

Dark grey (#333333) – Used for the body text and in the progress spinner.

When you choose the light blue color theme, the website links, body text, and background accent appear in your streaming application catalog page as follows.



Blue

The following colors are applied when you select the blue color theme:

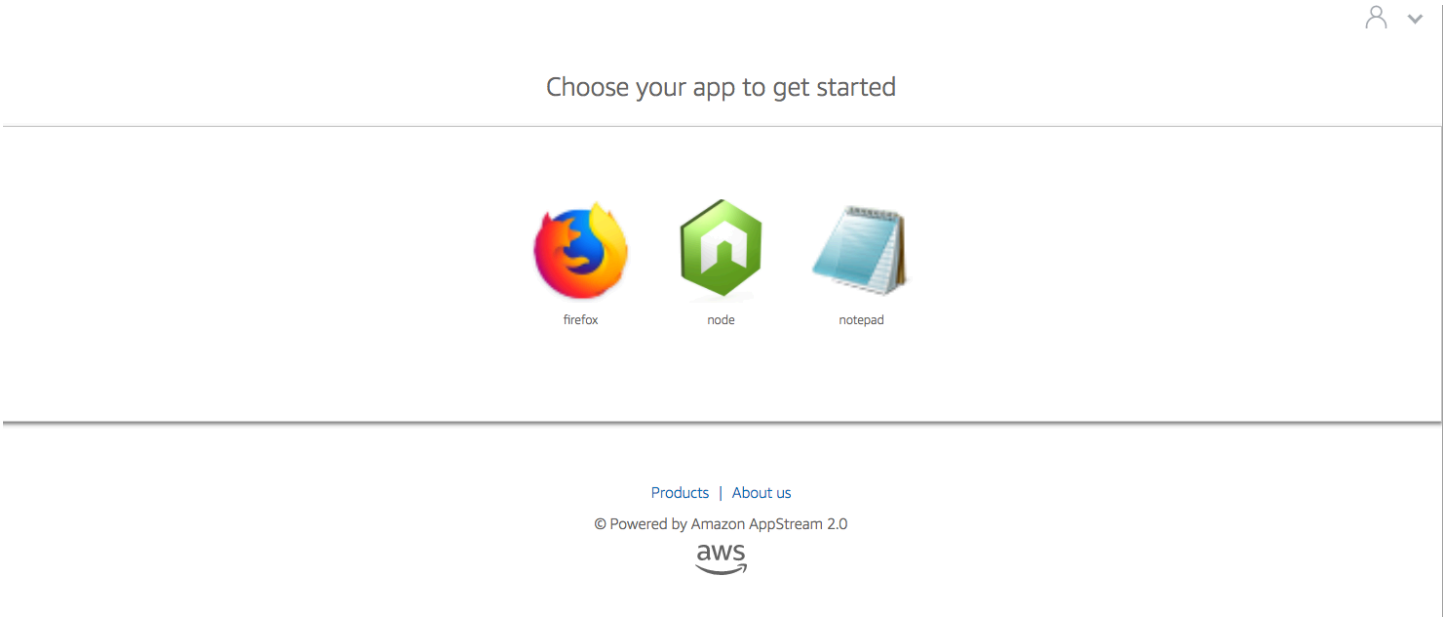
Blue (#0070ba) – Used for website links.

White (#ffffff) – Used as a background accent.

Light green (#8ac53e) – Used for buttons.

Grey (#666666) – Used for the body text and in the progress spinner.

When you choose the blue color theme, the website links, body text, and background accent appear in your streaming application catalog page as follows.



Pink

The following colors are applied when you select the pink color theme:

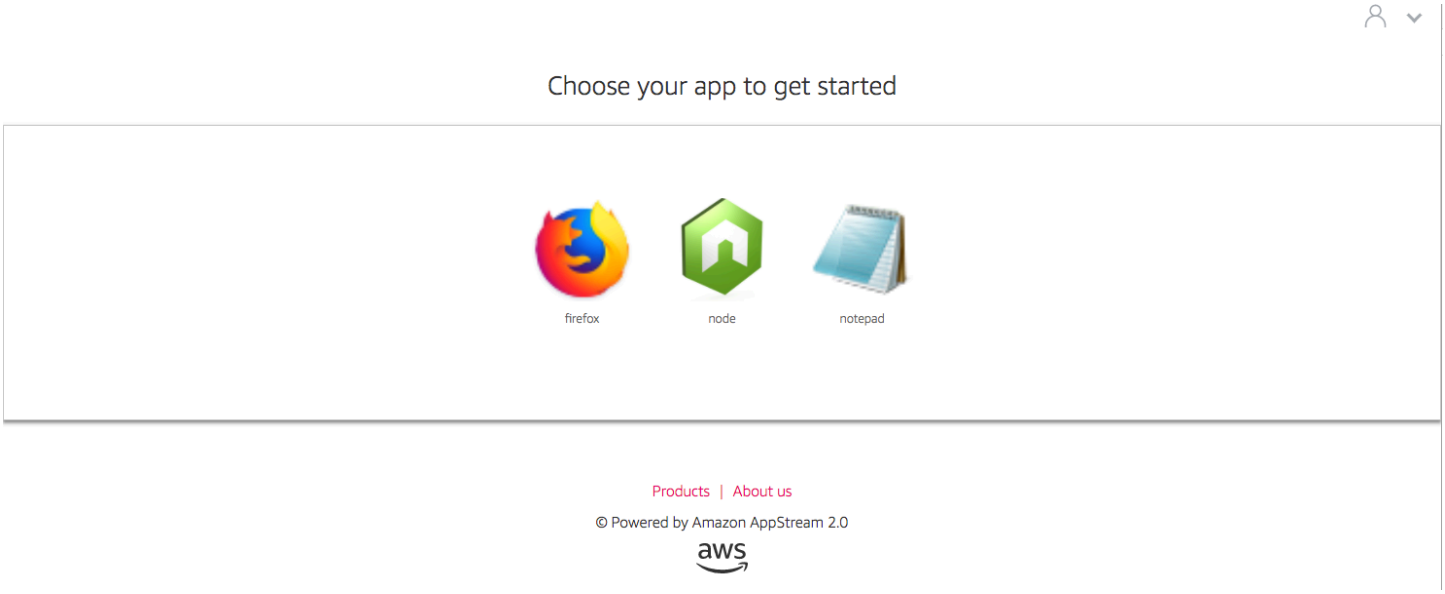
Pink (#ec0069) – Used for website links.

White (#ffffff) – Used as a background accent.

Blue (#3159a2) – Used for buttons.

Dark grey (#333333) – Used for the body text and in the progress spinner.

When you choose the pink color theme, the website links, body text, and background accent appear in your streaming application catalog page as follows.


Choose your app to get started
firefox
node
notepad
Products | About us
© Powered by Amazon AppStream 2.0
aws

Embed AppStream 2.0 Streaming Sessions

You can create a dynamic, interactive, and customized experience for your users by embedding an AppStream 2.0 streaming session within your website. Embedded AppStream 2.0 streaming sessions let your users interact with 3D models, maps, and datasets directly from your website. For example, users can view training instructions or educational materials alongside their AppStream 2.0 streaming session.

Contents

- Prerequisites
- Recommendations and Usage Considerations
- Step 1: Specify a Host Domain for Embedded AppStream 2.0 Streaming Sessions
- Step 2: Create a Streaming URL for User Authentication
- Step 3: Download the Embedded AppStream 2.0 Files
- Step 4. Configure Your Website for Integration with AppStream 2.0
- Constants, Functions, and Events for Embedded AppStream 2.0 Streaming Sessions

Prerequisites

To embed an AppStream 2.0 streaming session in a website, you must have the following:

- A configured AppStream 2.0 environment that includes an AppStream 2.0 image, fleet, and stack. For information about how to create these resources, see the following topics in the *AppStream 2.0 Administration Guide*:
  - Tutorial: Create a Custom AppStream 2.0 Image by Using the AppStream 2.0 Console or Create Your AppStream 2.0 Image Programmatically by Using the Image Assistant CLI Operations
  - Create a Fleet
  - Create a Stack
- A streaming URL for user authentication. SAML 2.0 and AppStream 2.0 user pools are currently not supported as authentication methods for embedded AppStream 2.0 streaming sessions.
- Optionally, you can use custom domains for embedded AppStream 2.0 streaming sessions. You can use custom domains so that your own company URL displays for users rather than an AppStream 2.0 URL. Custom domains are required if your users have web browsers that block third-party cookies.

> **Note**
>
> You can configure custom domains by using Amazon CloudFront. For information, see Using Custom Domains with AppStream 2.0.

When you use a custom domain, you must:

- Create a streaming URL that uses the same domain.
- Add **`appstream-custom-url-domain`** to the header of the webpage that will host the embedded AppStream 2.0 streaming sessions. For the header value, use the domain that your reverse proxy displays to users. For more information, see Configuration Requirements for Using Custom Domains.

Recommendations and Usage Considerations

Consider the following recommendations and usage notes for embedded AppStream 2.0 streaming sessions.

- To maintain maximum control over the embedded AppStream 2.0 streaming experience for your users, we recommend that you configure short-lived streaming URLs that last approximately 5 seconds. Any user can inspect the contents of a webpage and view its source. This includes the document object model (DOM) and the src (source) URL of the iframe. If the URL is still valid when a user copies it, that user can paste the URL in a separate browser tab and stream the session with the standard AppStream 2.0 portal user interface, without the embed options.
- Concurrent sessions are not supported when custom domains are used for embedded AppStream 2.0 streaming sessions. Concurrent sessions occur when users start two embedded AppStream 2.0 streaming sessions either on the same webpage or across two different browser tabs. You can't have a single user with concurrent sessions, but you can have multiple users. For example, a user logs into your app, your app generates a streaming URL to give to the customer (which counts as a unique user for billing), a customer loads the streaming URL, and the customer is assigned to an appstream instance within your specified pool.

Step 1: Specify a Host Domain for Embedded AppStream 2.0 Streaming Sessions

To embed an AppStream 2.0 streaming session in a webpage, first update your stack to specify the domain to host the embedded streaming session. This a security measure to ensure that only authorized website domains can embed AppStream 2.0 streaming sessions. AppStream 2.0 adds the domain or domains that you specify to the **`Content-Security-Policy`** (CSP) header. For more information, see Content Security Policy (CSP) in the Mozilla MDN Web Docs documentation.

To update your stack to specify the domain to host the embedded streaming session, use any of the following methods:

- The AppStream 2.0 console
- The `EmbedHostDomains` API action
- The `embed-host-domains` AWS command line interface (AWS CLI) command

To specify a host domain by using the AppStream 2.0 console, perform the following steps.

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. In the left navigation pane, choose **Stacks**, and select the stack that you want.
3. Choose **Edit**.
4. Expand **Embed AppStream 2.0 (Optional)**.
5. In **Host Domains**, specify a valid domain. For example: **`training.example.com`**.

   > **Note**
   >
   > Embedded streaming sessions are only supported over HTTPS [TCP port 443].

6. Choose **Update**.

Step 2: Create a Streaming URL for User Authentication

You must create a streaming URL to authenticate users for embedded AppStream 2.0 streaming sessions. SAML 2.0 and user pools are currently not supported for embedded streaming sessions. To create a streaming URL, use one of the following methods:

- AppStream 2.0 console
- The CreateStreamingURL API action
- The create-streaming-url AWS CLI command

Configuration Requirements for Using Custom Domains

Whether you use custom domains to apply your company branding or to ensure that embedded AppStream 2.0 streaming sessions work with browsers that block third-party cookies, the configuration requirements are the same.

For web browsers that block third-party cookies, custom domains are required. AppStream 2.0 uses browser cookies to authenticate streaming sessions and lets users reconnect to an active session without being prompted to provide their sign-in credentials every time. By default, AppStream 2.0 streaming URLs include **appstream.com** as the domain. When you embed a streaming session within your website, **appstream.com** is treated as a third-party domain. As a result, streaming sessions may be blocked when modern browsers are used that block third-party cookies by default.

To avoid embedded AppStream 2.0 streaming sessions from being blocked in this scenario, follow these steps:

1. Specify a custom domain to host your embedded AppStream 2.0 streaming sessions.

   When you configure your custom domain, make sure that the domain is a subdomain of the webpage in which you plan to embed AppStream 2.0. For example, if you update your stack to specify **training.example.com** as the host domain, you can create a subdomain called **content.training.example.com** for your embedded streaming sessions.
2. Create a streaming URL for embedded AppStream 2.0 streaming sessions that uses the same custom subdomain. To create the streaming URL, use the CreateStreamingURL API action or the create-streaming-url AWS CLI command. You cannot use the AppStream 2.0 console to create a streaming URL in this scenario.

   To create a streaming URL for embedded AppStream 2.0 streaming sessions, in the URL, replace **appstream2.*region*.aws.amazon.com** with your own domain.

   By default, AppStream 2.0 streaming URLs are formatted as follows:

   ```
   https://appstream2.region.aws.amazon.com/authenticate?parameters=authenticationcode
   ```

If your subdomain is **`content.training.example.com`**, your new streaming URL follows this format:

```
https://content.training.example.com/authenticate?parameters=authenticationcode
```

> **Note**
>
> When you create a custom domain, you can use the domain for embedded AppStream 2.0 streaming sessions only in the AWS Region for which it was configured. If you plan to support custom domains in multiple Regions, create a custom domain for each applicable Region. Also, embedded streaming sessions are only supported over HTTPS [TCP port 443].

3. Add **`appstream-custom-url-domain`** to the header of the webpage that will host the embedded streaming sessions. For the header value, use the domain that your reverse proxy displays to users. For example:

```
Header name: appstream-custom-url-domain
Header value: training.example.com
```

   Setting a custom domain and creating a streaming URL that specifies the same domain lets the cookies be saved as first-party cookies. For information about how to configure custom domains by using Amazon CloudFront, see Using Custom Domains with AppStream 2.0.

After you set up a custom domain for your embedded AppStream 2.0 streaming sessions, if your streaming URLs don't redirect to your custom domain, or if your custom domain doesn't display correctly for your users, see the following troubleshooting topics:

- I set up a custom domain for my embedded AppStream 2.0 streaming sessions, but my AppStream 2.0 streaming URLs aren't redirecting to my custom domain.

Step 3: Download the Embedded AppStream 2.0 Files

To host embedded AppStream 2.0 streaming sessions, you must download and configure the provided AppStream 2.0 API JavaScript file.

1. On the Embedding AppStream 2.0 in Your Website webpage, choose the link in step 1 to download the AppStream 2.0 Embed Kit .zip file, **appstream_embed_<version>.zip**.
2. Navigate to the location where you downloaded the .zip file, and extract the contents of the file.
3. The extracted contents of the file comprise one folder, **appstream-embed**. In addition to the **COPYRIGHT.txt** and **THIRD_PARTY_NOTICES.txt** file, this folder contains the following two files:
   - **appstream-embed.js** — Provides the embedded AppStream 2.0 API. This JavaScript file includes the functions and API actions for configuring and controlling your embedded AppStream 2.0 streaming session.
   - **embed-sample.html** — Describes how to use the embedded AppStream 2.0 API to initialize a streaming session, call functions, and listen for events. This sample file expands on the information in this topic, to provide an example use case for developers.

Step 4. Configure Your Website for Integration with AppStream 2.0

The following sections provide information about how to configure your webpage to host embedded AppStream 2.0 streaming sessions.

Contents

- Import the appstream-embed JavaScript File
- Initialize and Configure the AppStream.Embed Interface Object
- Examples for Hiding Items in the AppStream 2.0 User Interface

Import the appstream-embed JavaScript File

1. On the webpage where you plan to embed the AppStream 2.0 streaming session, import the **appstream-embed.js** file into the webpage by adding the following code:

```
<script type="text/javascript" src="./appstream_embed.js"> </script>
```

2. Next, create an empty container div. The ID of the div that you set is passed into the AppStream 2.0 embed constructor. It's then used to inject an iframe for the streaming session. To create the div, add the following code:

```
<div id="appstream-container"> </div>
```

Initialize and Configure the `AppStream.Embed` Interface Object

To initialize the `AppStream.Embed` interface object in JavaScript, you must add code that creates an `AppStream.Embed` object with options for the streaming URL and user interface configuration. These options, and the div ID that you created, are stored in an object called `appstreamOptions`.

The following example code shows how to initialize the `AppStream.Embed` interface object.

```
var appstreamOptions = {
     sessionURL: 'https://appstream2.region.aws.amazon.com/authenticate?
parameters=authenticationcode...',
     userInterfaceConfig:{[AppStream.Embed.Options.HIDDEN_ELEMENTS]:
[AppStream.Embed.Elements.TOOLBAR]}
 };
 appstreamEmbed = new AppStream.Embed("appstream-container", appstreamOptions);
```

In the code, replace *`sessionURL`* and *`userInterfaceConfig`* with your own values.

> **Note**
>
> The value specified for *`userInterfaceConfig`* hides the entire AppStream 2.0 toolbar. This value, which is included as an example, is optional.

`sessionUrl`

The streaming URL that you created by using the AppStream 2.0 console, the CreateStreamingURL API action, or the create-streaming-url AWS CLI command. This parameter is case-sensitive.

Type: String

Required: Yes

userInterfaceConfig

The configuration that generates the initial state of the user interface elements. The configuration is a key-value pair.

The key, `AppStream.Embed.Options.HIDDEN_ELEMENTS`, specifies the user interface objects that are initially hidden when the embedded AppStream 2.0 streaming session is initialized. Later, you can return both hidden and visible objects by using the `getInterfaceState` parameter.

The value is an array of constants (toolbar buttons). For a list of constants that you can use, see Working with `HIDDEN_ELEMENTS`.

Type: Map (*key*:*value*)

Required: No

Examples for Hiding Items in the AppStream 2.0 User Interface

The examples in this section show how to hide items in the AppStream 2.0 user interface from users during their embedded AppStream 2.0 streaming sessions.

Examples

- Example 1: Hide the entire AppStream 2.0 toolbar
- Example 2: Hide a specific button on the AppStream 2.0 toolbar
- Example 3: Hide multiple buttons on the AppStream 2.0 toolbar

Example 1: Hide the entire AppStream 2.0 toolbar

To prevent users from accessing any button on the AppStream 2.0 toolbar during embedded streaming sessions, use the `AppStream.Embed.Elements.TOOLBAR` constant. This constant lets you hide all AppStream 2.0 toolbar buttons.

```
var appstreamOptions = {
     sessionURL: 'https://appstream2.region.aws.amazon.com/authenticate?
parameters=authenticationcode...',
     userInterfaceConfig:{[AppStream.Embed.Options.HIDDEN_ELEMENTS]:
[AppStream.Embed.Elements.TOOLBAR]}
```

```
 };
```

Example 2: Hide a specific button on the AppStream 2.0 toolbar

You can display the AppStream 2.0 toolbar, while preventing users from accessing a specific toolbar button during embedded streaming sessions. To do so, specify the constant for the button that you want to hide. The following code uses the `AppStream.Embed.Elements.FILES_BUTTON` constant to hide the **My Files** button. This prevents users from accessing persistent storage options during embedded streaming sessions.

```
var appstreamOptions = {
     sessionURL: 'https://appstream2.region.aws.amazon.com/authenticate?
parameters=authenticationcode...',
     userInterfaceConfig:{[AppStream.Embed.Options.HIDDEN_ELEMENTS]:
[AppStream.Embed.Elements.FILES_BUTTON]}
 };
```

Example 3: Hide multiple buttons on the AppStream 2.0 toolbar

You can display the AppStream 2.0 toolbar, while preventing users from accessing more than one toolbar button during embedded streaming sessions. To do so, specify the constants for the buttons that you want to hide. The following code uses the `AppStream.Embed.Elements.END_SESSION_BUTTON` and `AppStream.Embed.Elements.FULLSCREEN_BUTTON` constants to hide the **End Session** and **Fullscreen** buttons.

> **Note**
>
> Separate each constant with a comma, with no preceding or following space.

```
var appstreamOptions = {
     sessionURL: 'https://appstream2.region.aws.amazon.com/authenticate?
parameters=authenticationcode... (https://appstream2.region.aws.amazon.com/#/)',
     userInterfaceConfig:{[AppStream.Embed.Options.HIDDEN_ELEMENTS]:
[AppStream.Embed.Elements.END_SESSION_BUTTON,AppStream.Embed.Elements.FULLSCREEN_BUTTON]}
 };
```

Constants, Functions, and Events for Embedded AppStream 2.0 Streaming Sessions

The following topics provide reference information for constants, functions, and events that you can use to configure embedded AppStream 2.0 streaming sessions.

Contents

- Working with HIDDEN_ELEMENTS
- Functions for the AppStream.Embed Object
- Events for Embedded AppStream 2.0 Streaming Sessions
- Examples for Adding Event Listeners and Ending an Embedded AppStream 2.0 Streaming Session

The following AppStream 2.0 user interface elements can be passed into the `HIDDEN_ELEMENTS` configuration option when an embedded AppStream 2.0 streaming session is initialized.

Working with `HIDDEN_ELEMENTS`

The following AppStream 2.0 user interface elements can be passed as constants into the `HIDDEN_ELEMENTS` configuration option when an embedded AppStream 2.0 streaming session is initialized.

```
AppStream.Embed.Elements.TOOLBAR
AppStream.Embed.Elements.FULLSCREEN_BUTTON
AppStream.Embed.Elements.END_SESSION_BUTTON
AppStream.Embed.Elements.TOOLBAR
AppStream.Embed.Elements.CATALOG_BUTTON
AppStream.Embed.Elements.WINDOW_SWITCHER_BUTTON
AppStream.Embed.Elements.FILES_BUTTON
AppStream.Embed.Elements.CLIPBOARD_BUTTON
AppStream.Embed.Elements.COPY_LOCAL_BUTTON
AppStream.Embed.Elements.PASTE_REMOTE_BUTTON
AppStream.Embed.Elements.SETTINGS_BUTTON
AppStream.Embed.Elements.STREAMING_MODE_BUTTON
AppStream.Embed.Elements.SCREEN_RESOLUTION_BUTTON
AppStream.Embed.Elements.REGIONAL_SETTINGS_BUTTON
AppStream.Embed.Elements.FULLSCREEN_BUTTON
AppStream.Embed.Elements.END_SESSION_BUTTON
```

The following three elements can be passed as strings into HIDDEN_ELEMENTS, rather than as constants.

String	Description
`'adminCommandsButton'`	When you are connected to an AppStream 2.0 image builder, the **Admin Commands** button displays on the top right corner of the AppStream 2.0 toolbar. Passing this string into `HIDDEN_ELEMENTS` hides the **Admin Commands** button.
`'softKeyboardButton'`	During AppStream 2.0 streaming sessions on touch-enabled devices, users can tap the keyboard icon on the AppStream 2.0 toolbar to display the on-screen keyboard. Passing this string into `HIDDEN_ELEMENTS` hides the keyboard icon.
`'keyboardShortcutsButton'`	During AppStream 2.0 streaming sessions on touch-enabled devices, users can tap the Fn icon on the AppStream 2.0 toolbar to display keyboard shortcuts. Passing this string into `HIDDEN_ELEMENTS` hides the Fn icon.

Functions for the `AppStream.Embed` Object

The following table lists the functions that can be performed on the `AppStream.Embed` object.

Function	Description
`AppStream.Embed(containerId:string, options:object)`	The `AppStream.Embed` object constructor. This constructor initializes and communicates with the `AppStream.Embed` object, and it uses a div container ID. The ID is used to inject the iframe. It also injects an object that includes the configuration options for `appstreamOptions` (*`sessionURL`* and `HIDDEN_ELEMENTS`).

Function	Description
`endSession()`	This function ends the streaming session, but does not destroy the iframe. If you specify a redirect URL, the iframe attempts to load the URL. Depending on the CORS headers of the page, the URL may not load.
`launchApp(appId:string)`	This function programmatically launches an application with the application ID that was specified during image creation.
`launchAppSwitcher()`	This function sends the **AppSwitcher** command to the AppStream 2.0 portal. This triggers the **AppSwitcher** command on the instance.
`getSessionState()`	This function returns an object for `sessionStatus` . For more information, see Events for Embedded AppStream 2.0 Streaming Sessions.
`getUserInterfaceState()`	This function returns an object for `UserInterfaceState` . The object contains the key-value pairs for the following: `sessionStatus` : State enumeration `sessionTerminationReason` : String `sessionDisconnectionReason` : String For more information, see Events for Embedded AppStream 2.0 Streaming Sessions.
`addEventListener(name, callback)`	This function adds a callback function to call when the specified event is triggered. For a list of the events that can be triggered, see Events for Embedded AppStream 2.0 Streaming Sessions.
`removeEventListener(name, callback)`	This function removes the callback for the specified events.

Function	Description
`destroy()`	This function deletes the iframe and cleans up resources. This function does not affect streaming sessions that are in progress.

Events for Embedded AppStream 2.0 Streaming Sessions

The following table lists the events that can be triggered during embedded AppStream 2.0 streaming sessions.

Event	Data	Description
`AppStream.Embed.Events.SESSION_STATE_CHANGE`	`sessionStatus` : `State enumeration` `sessionTerminationReason` : String `sessionDisconnectionReason` : String	This event is triggered when any session state change occurs. The event includes a map of the states that changed. To retrieve the full session state, use the `getSessionState()` function. Following are the session states: `AppStream.Embed.Se`

Event	Data	Description
		`ssionStatus.Unknown` — The session has not started and is not reserved `AppStream.Embed.SessionStatus.Reserved` — The session is reserved but has not started. `AppStream.Embed.SessionStatus.Started` — The user connected to the session and started streaming. `AppStream.Embed.SessionStatus Disconnected` — The user

Event	Data	Description
		disconnected from the session. `AppStream.Embed.SessionStatus.Ended` — The session was marked as ended or expired.
`AppStream.Embed.Events.SESSION_INTERFACE_STATE_CHANGE`	`hiddenElements` : Array of strings `isFullscreen` : Boolean `isSoftKeyboardVisible` : Boolean	This event is triggered when any session state change occurs. The event includes a map of the states that changed. To retrieve the full session state, use the `getSessionState()` function.

Event	Data	Description
`AppStream.Embed.Events.SESSION_ERROR`	`errorCode` : Number `errorMessage` : String	This event is triggered when any errors occur during a session.

Examples for Adding Event Listeners and Ending an Embedded AppStream 2.0 Streaming Session

The examples in this section show how to do the following:

- Add event listeners for embedded AppStream 2.0 streaming sessions.
- Programmatically end an embedded AppStream 2.0 streaming session.

Example 1: Add event listeners for embedded AppStream 2.0 streaming sessions

To add event listeners for session state changes, session interface state changes, and session errors during embedded streaming sessions, use the following code:

```
appstreamEmbed.addEventListener(AppStream.Embed.Events.SESSION_STATE_CHANGE,
 updateSessionStateCallback);

appstreamEmbed.addEventListener(AppStream.Embed.Events.SESSION_INTERFACE_STATE_CHANGE,
 updateUserInterfaceStateCallback);

appstreamEmbed.addEventListener(AppStream.Embed.Events.SESSION_ERROR, errorCallback);
```

In this example, `AppStream.Embed.Events.SESSION_STATE_CHANGE`, `AppStream.Embed.Events.SESSION_INTERFACE_STATE_CHANGE`, and `AppStream.Embed.Events.SESSION_ERROR` are event names.

The `updateSessionStateCallback`, `updateUserInterfaceStateCallback`, and `errorCallback` functions are ones that you implement. These functions are passed into the `addEventListener` function and called when an event is triggered.

Example 2: Programmatically end an embedded AppStream 2.0 streaming session

To end an embedded AppStream 2.0 streaming sessions, use the following function:

```
appstreamEmbed.endSession();
```

Enable and Administer Persistent Storage for Your AppStream 2.0 Users

Amazon AppStream 2.0 supports the following persistent storage options for users in your organization:

- Home folders
- Google Drive for Google Workspace
- OneDrive for Business

You can enable one or more options for your organization. As an AppStream 2.0 administrator, you must understand how to perform the following tasks to enable and administer persistent storage for your users.

Contents

- Enable and Administer Home Folders for Your AppStream 2.0 Users
- Enable and Administer Google Drive for Your AppStream 2.0 Users
- Enable and Administer OneDrive for Business for Your AppStream 2.0 Users

For troubleshooting information, see Troubleshooting Persistent Storage Issues.

Enable and Administer Home Folders for Your AppStream 2.0 Users

AppStream 2.0 supports the following persistent storage options for users in your organization:

- Home folders
- Google Drive for Google Workspace
- OneDrive for Business

You can enable one or more options for your organization. When you enable home folders for an AppStream 2.0 stack, users of the stack can access a persistent storage folder during their application streaming sessions. No further configuration is required for your users to access their

home folder. Data stored by users in their home folder is automatically backed up to an Amazon Simple Storage Service bucket in your Amazon Web Services account and is made available to those users in subsequent sessions.

Files and folders are encrypted in transit using Amazon S3's SSL endpoints. Files and folders are encrypted at rest using Amazon S3-managed encryption keys.

Home folders are stored on fleet instances in the following default locations:

- For single-session, non-domain-joined Windows instances: C:\Users\PhotonUser\My Files\Home Folder
- For multi-session, non-domain-joined Windows instances: C:\Users\as2-xxxxxxxx\My Files\Home Folder , where as2-xxxxxxxxx is a random user name assigned to each user session. You can determine your local user name through env variable $USERNAME.
- Domain-joined Windows instances: C:\Users\%username%\My Files\Home Folder
- Linux instances: ~/MyFiles/HomeFolder

As an administrator, use the applicable path if you configure your applications to save to the home folder. In some cases, your users may not be able to find their home folder because some applications do not recognize the redirect that displays the home folder as a top-level folder in File Explorer. If this is the case, your users can access their home folder by browsing to the same directory in File Explorer.

Contents

- Files and Directories Associated with Compute-Intensive Applications
- Enable Home Folders for Your AppStream 2.0 Users
- Administer Your Home Folders

Files and Directories Associated with Compute-Intensive Applications

During AppStream 2.0 streaming sessions, saving large files and directories associated with compute-intensive applications to persistent storage can take longer than saving files and directories required for basic productivity applications. For example, it might take longer for applications to save a large amount of data or frequently modify the same files than it would to save files created by applications that perform a single write action. It might also take longer to save many small files.

If your users save files and directories associated with compute-intensive applications and AppStream 2.0 persistent storage options aren't performing as expected, we recommend that you use a Server Message Block (SMB) solution such as Amazon FSx for Windows File Server or an AWS Storage Gateway file gateway. Following are examples of files and directories associated with compute-intensive applications that are more suitable for use with these SMB solutions:

- Workspace folders for integrated development environments (IDEs)
- Local database files
- Scratch space folders created by graphics simulation applications

For more information, see:

- *Amazon FSx for Windows File Server Windows User Guide*
- Using Amazon FSx with Amazon AppStream 2.0
- File gateways in the *AWS Storage Gateway User Guide*

Enable Home Folders for Your AppStream 2.0 Users

Before enabling home folders, you must do the following:

- Check that you have the correct AWS Identity and Access Management (IAM) permissions for Amazon S3 actions. For more information, see Using IAM Policies to Manage Administrator Access to the Amazon S3 Bucket for Home Folders and Application Settings Persistence.
- 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 AppStream 2.0 Base Image and Managed Image Update Release Notes.
- Enable network connectivity to Amazon S3 from your virtual private cloud (VPC) by configuring internet access or a VPC endpoint for Amazon S3. For more information, see Networking and Access for Amazon AppStream 2.0 and Using Amazon S3 VPC Endpoints for AppStream 2.0 Features.

You can enable or disable home folders while creating a stack (see Create a Stack), or after the stack is created by using the AWS Management Console for AppStream 2.0, AWS SDK, or AWS CLI. For each AWS Region, home folders are backed up by an Amazon S3 bucket.

The first time you enable home folders for an AppStream 2.0 stack in an AWS Region, the service creates an Amazon S3 bucket in your account in that same Region. The same bucket is used to store the content of home folders for all users and all stacks in that Region. For more information, see Amazon S3 Bucket Storage.

> **Note**
>
> For guidance that you can provide your users to help them get started with using home folders during AppStream 2.0 streaming sessions, see Use Home Folders.

To enable home folders while creating a stack

- Follow the steps in Create a Stack, and make sure that **Enable Home Folders** is selected.

To enable home folders for an existing stack

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. In the left navigation pane, choose **Stacks**, and select the stack for which to enable home folders.
3. Below the stacks list, choose **Storage** and select **Enable Home Folders**.
4. In the **Enable Home Folders** dialog box, choose **Enable**.

Administer Your Home Folders

Contents

- Disable Home Folders
- Amazon S3 Bucket Storage
- Home Folder Content Synchronization
- Home Folder Formats
- Using the AWS Command Line Interface or AWS SDKs
- Additional Resources

Disable 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:

- Users who are connected to active streaming sessions for the stack receive an error message. They are informed that they can no longer store content in their home folder.
- Home folders do not appear for any new sessions that use the stack with home folders disabled.
- Disabling home folders for one stack does not disable it for other stacks.
- 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 by following the steps described earlier in this topic.

To disable home folders while creating a stack

- Follow the steps in Create a Stack and make sure that the **Enable Home Folders** option 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. In the left navigation pane, choose **Stacks**, and select the stack.
3. Below the stacks list, choose **Storage** and clear **Enable Home Folders**.
4. In the **Disable Home Folders** dialog box, type `CONFIRM` (case-sensitive) to confirm your choice, then choose **Disable**.

Amazon S3 Bucket Storage

AppStream 2.0 manages user content stored in home folders by using Amazon S3 buckets created in your account. For every AWS Region, AppStream 2.0 creates a bucket in your account. All user content generated from streaming sessions of stacks in that Region is stored in that bucket. The buckets are fully managed by the service without any input or configuration from an administrator. 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 Amazon Web Services 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 the US West (Oregon) Region (us-west-2) on account number 123456789012, the service creates an Amazon S3 bucket in that Region with the name shown. Only an administrator with sufficient permissions can delete this bucket.

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

As mentioned earlier, disabling home folders for stacks does not delete any user content stored in the Amazon 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 Using IAM Policies to Manage Administrator Access to the Amazon S3 Bucket for Home Folders and Application Settings Persistence.

Home Folder Content Synchronization

When home folders are enabled, AppStream 2.0 creates a unique folder for each user in which to store their content. The folder is created as a unique Amazon S3 prefix that uses a hash of the user name within an S3 bucket for your Amazon Web Services account and Region. After AppStream 2.0 creates the home folder in Amazon S3, it copies the accessed content in that folder from the S3 bucket to the fleet instance. This enables the user to access their home folder content quickly, from the fleet instance, during their streaming session. Changes that you make to a user's home folder content in an S3 bucket and that the user makes to their home folder content on a fleet instance are synchronized between Amazon S3 and AppStream 2.0 as follows.

1. At the beginning of a user's AppStream 2.0 streaming session, AppStream 2.0 catalogs the home folder files that are stored for that user in the Amazon S3 bucket for your Amazon Web Services account and Region.
2. A user's home folder content is also stored on the AppStream 2.0 fleet instance from which they stream. When a user accesses their home folder on the AppStream 2.0 fleet instance, the list of cataloged files is displayed.
3. AppStream 2.0 downloads a file from the S3 bucket to the fleet instance only after the user uses a streaming application to open the file during their streaming session.
4. After AppStream 2.0 downloads the file to the fleet instance, synchronization occurs after the file is accessed

5. If the user changes the file during their streaming session, AppStream 2.0 uploads the new version of the file from the fleet instance to the S3 bucket periodically or at the end of the streaming session. However, the file is not downloaded from the S3 bucket again during the streaming session.

The following sections describe synchronization behavior when you add, replace, or remove a user's home folder file in Amazon S3.

Contents

- Synchronization of files that you add to a user's home folder in Amazon S3
- Synchronization of files that you replace in a user's home folder in Amazon S3
- Synchronization of files that you remove from a user's home folder in Amazon S3

Synchronization of files that you add to a user's home folder in Amazon S3

If you add a new file to a user's home folder in an S3 bucket, AppStream 2.0 catalogs the file and displays it in the list of files in the user's home folder within a few minutes. However, the file isn't downloaded from the S3 bucket to the fleet instance until the user opens the file with an application during their streaming session.

Synchronization of files that you replace in a user's home folder in Amazon S3

If a user opens a file in their home folder on the fleet instance during their streaming session, and you replace the same file in their home folder in an S3 bucket with a new version during that user's active streaming session, the new version of the file is not immediately downloaded to the fleet instance. The new version is downloaded from the S3 bucket to the fleet instance only after the user starts a new streaming session and opens the file again.

Synchronization of files that you remove from a user's home folder in Amazon S3

If a user opens a file in their home folder on the fleet instance during their streaming session, and you remove the file from their home folder in an S3 bucket during that user's active streaming session, the file is removed from the fleet instance after the user does either of the following:

- Opens the home folder again
- Refreshes the home folder

Home Folder Formats

The hierarchy of a user folder depends on how a user launches a streaming session, as described in the following sections.

AWS SDKs and AWS CLI

For sessions launched 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 and *`user-id-SHA-256-hash`* is the user-specific folder name created using a lowercase 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 CLI Command Reference*.

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

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

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

SAML 2.0

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 lowercase SHA-256 hash hexadecimal string generated from the `NameID` SAML attribute value passed in the SAML federation request. To differentiate users who have the same name but belong to two different domains, send the SAML request with `NameID` in the format `domainname\username`. For more information, see Amazon AppStream 2.0 Integration with SAML 2.0.

The following example folder structure applies to session access using SAML federation with `NameID` SAMPLEDOMAIN\testuser, account ID 123456789012 in the US West (Oregon) Region:

```
appstream2-36fb080bb8-us-west-2-123456789012/user/
federated/8dd9a642f511609454d344d53cb861a71190e44fed2B8aF9fde0C507012a9901
```

When part or all of the NameID string is capitalized (as the domain name *SAMPLEDOMAIN* is in the example), AppStream 2.0 generates the hash value based on the capitalization used in the string. Using this example, the hash value for SAMPLEDOMAIN\testuser is 8DD9A642F511609454D344D53CB861A71190E44FED2B8AF9FDE0C507012A9901. In the folder for that user, this value is displayed in lowercase, as follows: 8dd9a642f511609454d344d53cb861a71190e44fed2B8aF9fde0C507012a9901.

You can identify the folder for a user by generating the 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 by using the AWS CLI or AWS SDKs.

Use the following create-stack command to enable home folders while creating a new stack:

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

Use the following update-stack command to enable home folders for an existing stack:

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

Use the following command to disable home folders for an existing stack. This command does not delete any user data.

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

Additional Resources

For more information about managing Amazon S3 buckets and best practices, see the following topics in the *Amazon Simple Storage Service User Guide*:

- You can provide offline access to user data for your users with Amazon S3 policies. For more information, see Amazon S3: Allows IAM Users Access to Their S3 Home Directory, Programmatically and In the Console in the *IAM User Guide*.
- You can enable file versioning for content stored in Amazon S3 buckets used by AppStream 2.0. For more information, see Using Versioning.

Enable and Administer Google Drive for Your AppStream 2.0 Users

> **Note**
>
> Amazon AppStream 2.0's use and transfer to any other app of information received from Google APIs will adhere to Google API Services User Data Policy, including the Limited Use requirements.

Amazon AppStream 2.0 supports the following persistent storage options for users in your organization:

- Google Drive for Google Workspace
- OneDrive for Business
- Home folders

You can enable one or more options for your organization. When you enable Google Drive for Google Workspace for an AppStream 2.0 stack, users of the stack can link their Google Drive for Google Workspace account to AppStream 2.0. Then they can sign into their Google Drive for Google Workspace account and access their Google Drive folder during application streaming sessions. Any changes that they make to files or folders in Google Drive during those sessions are automatically backed up and synchronized, so that they are available to users outside of their streaming sessions.

> **Important**
>
> You can enable Google Drive for Google Workspace for accounts in your Google Workspace domains only, but not for personal Gmail accounts.

> **Note**
>
> You can enable Google Drive for Windows stacks, but not for Linux stacks or stacks associated with multi-session fleets.

Contents

- Enable Google Drive for Your AppStream 2.0 Users
- Disable Google Drive for Your AppStream 2.0 Users

Enable Google Drive for Your AppStream 2.0 Users

Before enabling Google Drive, you must do the following:

- Have an active Google Workspace account with a valid organizational domain and users in the domain to use with AppStream 2.0.
- Configure an AppStream 2.0 stack with an associated fleet.

  The fleet must use an image that uses a version of the AppStream 2.0 agent released on or after May 31, 2018. For more information, see AppStream 2.0 Agent Release Notes. The fleet must also have access to the internet.
- Add Amazon AppStream 2.0 as a trusted app in one or more domains associated with your Google Workspace account. You can enable Google Drive for up to 10 domains.
- Have a Windows-based stack. (Linux-based stacks are not supported).

Follow these steps to add Amazon AppStream 2.0 as a trusted app in your Google Workspace domains.

To add Amazon AppStream 2.0 as a trusted app in your Google Workspace domains

1. Sign in to the Google Workspace Admin console at https://admin.google.com/.
2. In the left navigation sidebar, choose **Security**, **Access and data control**, **API controls**.
3. At the top of the page, in the **App access control** section, choose **MANAGE THIRD-PARTY APP ACCESS**.
4. Choose **Add app**, and then choose **OAuth App Name Or Client ID**.

5. Enter the Amazon AppStream 2.0 OAuth client ID for your AWS Region, and then choose **SEARCH**. For a list of client IDs, see the table that follows this procedure.
6. In the search results, choose Amazon AppStream 2.0, and then choose **Select**.
7. In the **Client ID** page, under **OAuth Client ID**, verify that the correct ID appears in the list, and then select the check box to the left of the ID.
8. On the lower right of the page, choose **SELECT**.
9. Configure which organizational units in your Google Workspace organization should gain access.
10. Under **Access to Google Data**, choose **Trusted: Can access all Google services**, and then choose **CONTINUE**.
11. Review that the selections made are correct, then when you are satisfied, choose **FINISH**.
12. Verify that the Amazon AppStream 2.0 app, with the correct OAuth ID, appears in the list of connected apps.

Amazon AppStream 2.0 OAuth2 client IDs

Region	Amazon AppStream 2.0 OAuth client ID
US East (N. Virginia)	`266080779488-15n5q5nkiclp6m524qibnmhmbsg0hk92.apps.googleusercontent.com`
US East (Ohio)	`723951369598-6tvdlf52g2qh0qa141o4k1avasvnj51i.apps.googleusercontent.com`
US West (Oregon)	`1026466167591-i4jmemrggsjomp9tnkkcs5tniggfiujb.apps.googleusercontent.com`
Asia Pacific (Mumbai)	`325827353178-coqs1c374mf388ctllrlls374dc1bmb2.apps.googleusercontent.com`
Asia Pacific (Seoul)	`562383781419-am1i2dnvt050tmdltsvr36i8l2js40dj.apps.googleusercontent.com`
Asia Pacific (Singapore)	`856871139998-4eia2n1db5j6gtv4c1rdte1fh1gec8vs.apps.googleusercontent.com`

Region	Amazon AppStream 2.0 OAuth client ID
Asia Pacific (Sydney)	`151535156524-b889372osskprm4dt1clpm53mo3m9omp.apps.googleusercontent.com`
Asia Pacific (Tokyo)	`922579247628-qpl9kpihg3hu5dul2lphbjs4qbg6mjm2.apps.googleusercontent.com`
Canada (Central)	`872792838542-t39aqh72jv895c89thtk6v83sl6jugm2.apps.googleusercontent.com`
Europe (Frankfurt)	`643727794574-1se5360a77i84je9j3ap12obov1ib76q.apps.googleusercontent.com`
Europe (Ireland)	`599492309098-098muc7ofjfo9vua5rm5u9q2k3mlok3j.apps.googleusercontent.com`
Europe (London)	`682555519925-usbn2sk1ffgo8odgf23nj66ri71na0k5.apps.googleusercontent.com`
AWS GovCloud (US-East)	`20306576244-gqqkappmhhv9fj06sdk7as60he89e7ce.apps.googleusercontent.com` **Note** For more information about using AppStream 2.0 in the AWS GovCloud (US) Regions, see Amazon AppStream 2.0 in the *AWS GovCloud (US) User Guide*.

Region	Amazon AppStream 2.0 OAuth client ID
AWS GovCloud (US-West)	`996065833880-litfkb2vfd7c65nt7s24r7t8le5bc9bl.apps.googleusercontent.com` **Note** For more information about using AppStream 2.0 in the AWS GovCloud (US) Regions, see Amazon AppStream 2.0 in the *AWS GovCloud (US) User Guide*.
South America (São Paulo)	`891888628791-1ltbtedva29esqvqadiatlj4htcgcjfo.apps.googleusercontent.com`

Follow these steps to enable Google Drive for your AppStream 2.0 users.

To enable Google Drive while creating a stack

- Follow the steps in Create a Stack, make sure that **Enable Google Drive** is selected, and that you have specified at least one organizational domain associated with your Google Workspace account.

To enable Google Drive for an existing stack

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. In the left navigation pane, choose **Stacks**, and select the stack for which to enable Google Drive.
3. Below the stacks list, choose **Storage** and select **Enable Google Drive for Google Workspace**.
4. In the **Enable Google Drive for Google Workspace** dialog box, in **Google Workspace domain name**, type the name of at least one organizational domain that is associated with your Google Workspace account. To specify another domain, choose **Add another domain**, and type the name of the domain.
5. After you add domain names, choose **Enable**.

> **Note**
>
> For guidance that you can provide your users to help them get started with using Google Drive during AppStream 2.0 streaming sessions, see Use Google Drive.

Disable Google Drive for Your AppStream 2.0 Users

You can disable Google Drive for a stack without losing user content that is already stored on Google Drive. Disabling Google Drive for a stack has the following effects:

- Users who are connected to active streaming sessions for the stack receive an error message. They are informed that they do not have permissions to access their Google Drive.
- Any new sessions that use the stack with Google Drive disabled do not display Google Drive.
- Only the specific stack for which Google Drive is disabled is affected.
- Even if Google Drive is disabled for all stacks, AppStream 2.0 does not delete the user content stored in their Google Drive.

Follow these steps to disable Google Drive for an existing stack.

To disable Google Drive for an existing stack

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. In the left navigation pane, choose **Stacks**, and select the stack for which to disable Google Drive.
3. Below the stacks list, choose **Storage**, and clear the **Enable Google Drive for Google Workspace** option.
4. In the **Disable Google Drive for Google Workspace** dialog box, type `CONFIRM` (case-sensitive) to confirm your choice, then choose **Disable**.

   When users of the stack start their next AppStream 2.0 streaming session, they can no longer access their Google Drive folder from within that session and future sessions.

Enable and Administer OneDrive for Business for Your AppStream 2.0 Users

AppStream 2.0 supports the following persistent storage options for users in your organization.

- OneDrive for Business
- Google Drive for Google Workspace
- Home folders

You can enable one or more options for your organization. When you enable OneDrive for Business for an AppStream 2.0 stack, users of the stack can link their OneDrive for Business account to AppStream 2.0. Then they can sign into their OneDrive for Business account and access their OneDrive folder during application streaming sessions. Any changes that they make to files or folders in OneDrive during those sessions are automatically backed up and synchronized, so that they are available to users outside of their streaming sessions.

Important

You can enable OneDrive for Business for accounts in your OneDrive domains only, but not for personal accounts. AppStream 2.0 requires that you configure your Microsoft Azure Active Directory environment to allow end-user consent to applications. For more information, see Configure how end-users consent to applications in the Azure Active Directory Application management documentation.

The admin consent workflow lets administrators grant access to applications that require administrator approval. If the admin consent workflow is configured in your Azure Active Directory environment, contact AWS Support. For information about how to contact AWS Support, see AWS Support Center.

Note

You can enable OneDrive for Business for Windows stacks, but not for Linux stacks or stacks associated with multi-session fleets.

Contents

- Enable OneDrive for Your AppStream 2.0 Users
- Disable OneDrive for Your AppStream 2.0 Users

Enable OneDrive for Your AppStream 2.0 Users

Before enabling OneDrive, you must do the following:

- Have an active Microsoft Office 365 or OneDrive for Business account with a valid organizational domain and users in the domain to use with AppStream 2.0.
- Configure an AppStream 2.0 stack with an associated fleet.

  The fleet must use an image that uses a version of the AppStream 2.0 agent released on or after July 26, 2018. For more information, see AppStream 2.0 Agent Release Notes. The fleet must also have access to the internet.
- Have a Windows-based stack. (Linux-based stacks are not supported).

Follow these steps to enable OneDrive for your AppStream 2.0 users.

To enable OneDrive while creating a stack

- Follow the steps in Create a Stack, make sure that **Enable OneDrive** is selected, and that you have specified at least one organizational domain that is associated with your OneDrive for Business account.

To enable OneDrive for an existing stack

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. In the left navigation pane, choose **Stacks**, and select the stack for which to enable OneDrive.
3. Below the stacks list, choose **Storage**, and select **Enable OneDrive for Business**.
4. In the **Enable OneDrive for Business** dialog box, in **OneDrive domain name**, type the name of at least one organizational domain that is associated with your OneDrive account. To specify another domain, choose **Add another domain**, and type the name of the domain.
5. After you add OneDrive domain names, choose **Enable**.

Before your users can use OneDrive with AppStream 2.0, you must provide them with permissions to link their OneDrive account with third-party web applications. To do so, follow the steps in the next section.

> **Important**
>
> You must configure your Microsoft Azure Active Directory environment to allow end-user consent to applications. For more information, see Configure how end-users consent to applications in the Azure Active Directory Application management documentation.

Provide Your Users with Permissions to Link OneDrive with AppStream 2.0

You must enable Integrated Apps in your Office 365 or OneDrive for Business admin console before users can link their OneDrive for Business account to AppStream 2.0.

1. Sign in to Office 365 or the OneDrive for Business admin console.
2. In the left navigation pane of the console, choose **Settings**, **Services & add-ins**.
3. From the list of services and add-ins, choose **Integrated Apps**.
4. On the **Integrated apps** page, turn on the option to allow users in your organization to let third party web apps access their Office 365 information.

> **Note**
>
> For guidance that you can provide your users to help them get started with using OneDrive during AppStream 2.0 streaming sessions, see Use OneDrive for Business.

Disable OneDrive for Your AppStream 2.0 Users

You can disable OneDrive for a stack without losing user content that is already stored on OneDrive. Disabling OneDrive for a stack has the following effects:

- Users who are connected to active streaming sessions for the stack receive an error message. They are informed that they do not have permissions to access their OneDrive.
- Any new sessions that use the stack with OneDrive disabled do not display OneDrive.
- Only the specific stack for which OneDrive is disabled is affected.

- Even if OneDrive is disabled for all stacks, AppStream 2.0 does not delete the user content stored in their OneDrive.

Follow these steps to disable OneDrive for an existing stack.

To disable OneDrive for an existing stack

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. In the left navigation pane, choose **Stacks**, and select the stack for which to disable OneDrive.
3. Below the stacks list, choose **Storage**, and clear **Enable OneDrive for Business** option.
4. In the **Disable OneDrive for Business** dialog box, type `CONFIRM` (case-sensitive) to confirm your choice, then choose **Disable**.

   When users of the stack start their next AppStream 2.0 streaming session, they can no longer access their OneDrive folder from within that session and future sessions.

Enable Application Settings Persistence for Your AppStream 2.0 Users

AppStream 2.0 supports persistent application settings for Windows-based stacks. This means that your users' application customizations and Windows settings are automatically saved after each streaming session and applied during the next session. Examples of persistent application settings that your users can configure include, but are not limited to, browser favorites, settings, webpage sessions, application connection profiles, plugins, and UI customizations. These settings are saved to an Amazon Simple Storage Service (Amazon S3) bucket in your account, within the AWS Region in which application settings persistence is enabled. They are available in each AppStream 2.0 streaming session.

> **Note**
>
> Enabling application settings persistence is currently not supported for Linux-based stacks.

> **Note**
>
> Standard Amazon S3 charges may apply to data that is stored in your S3 bucket. For more information, see Amazon S3 Pricing.

Contents

- How Application Settings Persistence Works
- Enabling Application Settings Persistence
- Administer the VHDs for Your Users' Application Settings

How Application Settings Persistence Works

Persistent application settings are saved to a Virtual Hard Disk (VHD) file. This file is created the first time a user streams an application from a stack on which application settings persistence is enabled. If the fleet associated with the stack is based on an image that contains default application and Windows settings, the default settings are used for the user's first streaming

session. For more information about default settings, see *Step 3: Create Default Application and Windows Settings* in Tutorial: Create a Custom AppStream 2.0 Image by Using the AppStream 2.0 Console.

When the streaming session ends, the VHD is unmounted and uploaded to an Amazon S3 bucket within your account. The bucket is created when you enable persistent application settings for the first time for a stack in an AWS Region. The bucket is unique to your AWS account and the Region. The VHD is encrypted in transit using Amazon S3 SSL endpoints, and at rest using AWS Managed CMKs.

The VHD is mounted to the streaming instance in both C:\Users\%username% and D:\%username%. If your instance is not joined to an Active Directory domain, the Windows user name is PhotonUser. If your instance is joined to an Active Directory domain, the Windows user name is that of the logged in user.

Application settings persistence does not work across different operating system versions. For example, if you enable application settings persistence on a stack and the stack is associated with a fleet that uses a Windows Server 2012 R2 image, if you update the fleet to use an image that runs a different operating system (such as Windows Server 2016), settings from previous streaming sessions are not saved for users of the stack. Instead, after you update the fleet to use the new image, when users launch a streaming session from a fleet instance, a new Windows user profile is created. However, if you apply an update to the same operating system on the image, users' customizations and settings from previous streaming sessions are saved. When updates to the same operating system are applied to an image, the same Windows user profile is used when users launch a streaming session from the fleet instance.

Important

AppStream 2.0 supports applications that rely on the Microsoft Data Protection API only when the streaming instance is joined to a Microsoft Active Directory domain. In cases where a streaming instance is not joined to an Active Directory domain, the Windows user, PhotonUser, is different on each fleet instance. Due to the way in which the DPAPI security model works, users' passwords don’t persist for applications that use DPAPI in this scenario. In cases where streaming instances are joined to an Active Directory domain and the user is a domain user, the Windows user name is that of the logged in user, and users’ passwords persist for applications that use DPAPI.

AppStream 2.0 automatically saves all files and folders in this path, except for the following folders:

- Contacts
- Desktop
- Documents
- Downloads
- Links
- Pictures
- Saved Games
- Searches
- Videos

Files and folders created outside of these folders are saved within the VHD and synced to Amazon S3. The default VHD maximum size is 1GB. The size of the saved VHD is the total size of the files and folders that it contains. AppStream 2.0 automatically saves the HKEY_CURRENT_USER registry hive for the user. For new users (users whose profiles don't exist in Amazon S3), AppStream 2.0 creates the initial profile by using the default profile. This profile is created in the following location on the image builder: C:\users\default.

> **Note**
>
> The entire VHD must be downloaded to the streaming instance before a streaming session can begin. For this reason, a VHD that contains a large amount of data can delay the start of the streaming session. For more information, see Best Practices for Enabling Application Settings Persistence.

When you enable application settings persistence, you must specify a settings group. The settings group determines which saved application settings are used for a streaming session from this stack. AppStream 2.0 creates a new VHD file for the settings group that is stored separately within the S3 bucket in your AWS account. If the settings group is shared between stacks, the same application settings are used in each stack. If a stack requires its own application settings, specify a unique settings group for the stack.

Enabling Application Settings Persistence

Contents

- Prerequisites for Enabling Application Settings Persistence
- Best Practices for Enabling Application Settings Persistence
- How to Enable Application Settings Persistence

Prerequisites for Enabling Application Settings Persistence

To enable application settings persistence, you must first do the following:

- Check that you have the correct AWS Identity and Access Management (IAM) permissions for Amazon S3 actions. For more information, see the *IAM Policies and the Amazon S3 Bucket for Home Folders* section in Identity and Access Management for Amazon AppStream 2.0.
- Use an image that was created from a base image published by AWS on or after December 7, 2017. For a current list of released AWS base images, see AppStream 2.0 Base Image and Managed Image Update Release Notes.
- Associate the stack on which you plan to enable this feature with a fleet based on an image that uses a version of the AppStream 2.0 agent released on or after August 29, 2018. For more information, see AppStream 2.0 Agent Release Notes.
- Enable network connectivity to Amazon S3 from your virtual private cloud (VPC) by configuring internet access or a VPC endpoint for Amazon S3. For more information, see the *Home Folders and VPC Endpoints* section in Networking and Access for Amazon AppStream 2.0.

Best Practices for Enabling Application Settings Persistence

To enable application settings persistence without providing internet access to your instances, use a VPC endpoint. This endpoint must be in the VPC to which your AppStream 2.0 instances are connected. You must attach a custom policy to enable AppStream 2.0 access to the endpoint. For information about how to create the custom policy, see the *Home Folders and VPC Endpoints* section in Networking and Access for Amazon AppStream 2.0. For more information about private Amazon S3 endpoints, see VPC Endpoints and Endpoints for Amazon S3 in the *Amazon VPC User Guide*.

How to Enable Application Settings Persistence

You can enable or disable application settings persistence while creating a stack or after the stack is created by using the AppStream 2.0 console, AppStream 2.0 API, an AWS SDK, or the AWS Command Line Interface (CLI). For each AWS Region, persistent application settings are stored in an S3 bucket in your account.

The first time you enable application settings persistence for a stack in an AWS Region, AppStream 2.0 creates an S3 bucket in your AWS account in the same Region. The same bucket stores the application settings VHD file for all users and all stacks in that AWS Region. For more information, see *Amazon S3 Bucket Storage* in Administer the VHDs for Your Users' Application Settings.

To enable application settings persistence while creating a stack

- Follow the steps in Create a Stack, and make sure that **Enable Application Settings Persistence** is selected.

To enable application settings persistence for an existing stack

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. In the left navigation pane, choose **Stacks**, and select the stack for which to enable application settings persistence.
3. Below the stacks list, choose **User Settings**, **Application Settings Persistence**, **Edit**.
4. In the **Application Settings Persistence** dialog box, choose **Enable Application Settings Persistence**.
5. Confirm the current settings group or type the name of a new settings group. When you're done, choose **Update**.

New streaming sessions now have application settings persistence enabled.

Administer the VHDs for Your Users' Application Settings

Contents

- Amazon S3 Bucket Storage
- Reset a User's Application Settings

- Enable Amazon S3 Object Versioning and Revert a User's Application Settings
- Increase the Size of the Application Settings VHD

Amazon S3 Bucket Storage

When you enable application settings persistence, your users' application customizations and Windows settings are automatically saved to a Virtual Hard Disk (VHD) file that is stored in an Amazon S3 bucket created in your AWS account. For every AWS Region, AppStream 2.0 creates a bucket in your account that is unique to your account and the Region. All application settings configured by your users are stored in the bucket for that Region.

You do not need to perform any configuration tasks to manage these S3 buckets; they are fully managed by the AppStream 2.0 service. The VHD file that is stored in each bucket is encrypted in transit using Amazon S3's SSL endpoints and at rest using AWS Managed CMKs. The buckets are named in a specific format as follows:

```
appstream-app-settings-region-code-account-id-without-hyphens-random-identifier
```

region-code

This is the AWS Region code in which the stack is created with application settings persistence.

account-id-without-hyphens

Your AWS account ID. The random identifier ensures there is no conflict with other buckets in that Region. The first part of the bucket name, `appstream-app-settings`, does not change across accounts or Regions.

For example, if you enable application settings persistence for stacks in the US West (Oregon) Region (us-west-2) on account number 123456789012, AppStream 2.0 creates an Amazon S3 bucket within your account in that Region with the name shown. Only an administrator with sufficient permissions can delete this bucket.

```
appstream-app-settings-us-west-2-1234567890123-abcdefg
```

Disabling application settings persistence does not delete any VHDs stored in the S3 bucket. To permanently delete settings VHDs, you or another administrator with adequate permissions must do so by using the Amazon S3 console or API. 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 Application Settings Persistence* in Identity and Access Management for Amazon AppStream 2.0.

When application settings persistence is enabled, a unique folder is created for each settings group to store the settings VHD. The hierarchy of the folder in the S3 bucket depends on how the user launches a streaming session, as described in the following section.

The path for the folder where the settings VHD is stored in the S3 bucket in your account uses the following structure:

```
bucket-name/Windows/prefix/settings-group/access-mode/user-id-SHA-256-hash
```

bucket-name

The name of the S3 bucket in which users' application settings are stored. The name format is described earlier in this section.

prefix

The Windows version-specific prefix. For example, v4 for Windows Server 2012 R2.

settings-group

The settings group value. This value is applied to one or more stacks that share the same the same application settings.

access-mode

The identity method of the user: `custom` for the AppStream 2.0 API or CLI, `federated` for SAML, and `userpool` for user pool users.

user-id-SHA-256-hash

The user-specific folder name. This name is created using a lowercase SHA-256 hash hexadecimal string generated from the user ID.

The following example folder structure applies to a streaming session that is accessed using the API or CLI with a user ID of `testuser@mydomain.com`, an AWS account ID of `123456789012`, and the settings group `test-stack` in the US West (Oregon) Region (us-west-2):

```
appstream-app-settings-us-west-2-1234567890123-abcdefg/Windows/v4/test-stack/custom/
a0bcb1da11f480d9b5b3e90f91243143eac04cfccfbdc777e740fab628a1cd13
```

You can identify the folder for a user by generating the lowercase SHA-256 hash value of the user ID using websites or open source coding libraries available online.

Reset a User's Application Settings

To reset a user's application settings, you must find and delete the VHD and associated metadata file from the S3 bucket in your AWS account. Make sure that you do not do this during a user's active streaming session. After you delete the user's VHD and the metadata file, the next time the user launches a session from a streaming instance that has application settings persistence enabled, AppStream 2.0 creates a new settings VHD for that user.

To reset a user's application settings

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Bucket name** list, choose the S3 bucket that contains the application settings VHD that you want to reset.
3. Locate the folder that contains the VHD. For more information about how to navigate the S3 bucket folder structure, see *Amazon S3 Bucket Storage* earlier in this topic.
4. In the **Name** list, select the check box next to the VHD and the REG, choose **More**, and then choose **Delete**.
5. In the **Delete objects** dialog box, verify that the VHD and the REG are listed, and then choose **Delete**.

The next time the user streams from a fleet on which application settings persistence is enabled with the applicable settings group, a new application settings VHD is created. This VHD is saved to the S3 bucket at the end of the session.

Enable Amazon S3 Object Versioning and Revert a User's Application Settings

You can use Amazon S3 object versioning and lifecycle policies to manage your users' application settings when your users change them. With Amazon S3 object versioning, you can preserve, retrieve, and restore every version of the settings VHD. This enables you to recover from both unintended user actions and application failures. When versioning is enabled, after each streaming session, a new version of the application settings VHD is synced to Amazon S3. The new version does not overwrite the previous version, so if an issue with your users' settings occurs, you can revert to a previous version of the VHD.

> **Note**
>
> Each version of the application settings VHD is saved to Amazon S3 as a separate object and is charged accordingly.

Object versioning is not enabled by default in your S3 bucket, so you must explicitly enable it.

To enable object versioning for your application settings VHD

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Bucket name** list, choose the S3 bucket that contains the application settings VHD on which to enable object versioning.
3. Choose **Properties**.
4. Choose **Versioning**, **Enable versioning**, and then choose **Save**.

To expire older versions of your application settings VHDs, you can use Amazon S3 lifecycle policies. For information, see How Do I Create a Lifecycle Policy for an S3 Bucket? in the *Amazon Simple Storage Service User Guide*.

To revert a user's application settings VHD

You can revert to a previous version of a user's application settings VHD by deleting newer versions of the VHD from the applicable S3 bucket. Do not do this when the user has an active streaming session.

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Bucket name** list, choose the S3 bucket that contains the user's application settings VHD version to revert to.
3. Locate and select the folder that contains the VHD. For information about how to navigate the S3 bucket folder structure, see *Amazon S3 Bucket Storage* earlier in this topic.

   When you select the folder, the settings VHD and associated metadata file display.
4. To display a list of the VHD and metadata file versions, choose **Show**.
5. Locate the version of the VHD to revert to.
6. In the **Name** list, select the check boxes next to the newer versions of the VHD and associated metadata files, choose **More**, and then choose **Delete**.

7. Verify that the application settings VHD that you want to revert to and the associated metadata file are the newest versions of these files.

The next time the user streams from a fleet on which application settings persistence is enabled with the applicable settings group, the reverted version of the user's settings displays.

Increase the Size of the Application Settings VHD

The default VHD maximum size is 1 GB. If a user requires additional space for application settings, you can download the applicable application settings VHD to a Windows computer to expand it. Then, replace the current VHD in the S3 bucket with the larger one. Do not do this when the user has an active streaming session.

To increase the size of the application settings VHD

> **Note**
>
> The full VHD must be downloaded before a user can stream applications. Increasing the size of an application settings VHD can increase the time it takes for users to start application streaming sessions.

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Bucket name** list, choose the S3 bucket that contains the application settings VHD to expand.
3. Locate and select the folder that contains the VHD. For information about how to navigate the S3 bucket folder structure, see *Amazon S3 Bucket Storage* earlier in this topic.

   When you select the folder, the settings VHD and associated metadata file display.
4. Download the Profile.vhdx file to a directory on your Windows computer. Do not close your browser after the download completes, because you'll use the browser again later to upload the expanded VHD.
5. To use Diskpart to increase the size of the VHD to 2 GB, open the command prompt as an administrator, and type the following commands.

   ```
   diskpart
   ```

   ```
   select vdisk file="C:\path\to\application\settings\profile.vhdx"
   ```

```
expand vdisk maximum=2000
```

6. Then, type the following Diskpart commands to find and attach the VHD, and display the list of volumes:

   ```
   select vdisk file="C:\path\to\application\settings\profile.vhdx"
   ```

   ```
   attach vdisk
   ```

   ```
   list volume
   ```

   In the output, make note of the volume number with the label "AppStreamUS". In the next step, you select this volume so that you can enlarge it.
7. Type the following command:

   ```
   select volume ###
   ```

   where ### is the number in the list volume output.
8. Type the following command:

   ```
   extend
   ```
9. Type the following commands to confirm that the size of the partition on the VHD increased as expected (2 GB in this example):

   ```
   diskpart
   ```

   ```
   select vdisk file="C:\path\to\application\settings\profile.vhdx"
   ```

   ```
   list volume
   ```
10. Type the following command to detach the VHD so that it can be uploaded:

    ```
    detach vdisk
    ```
11. Return to your browser with the Amazon S3 console, choose **Upload**, **Add files**, and then select the enlarged VHD.
12. Choose **Upload**.

After the VHD is uploaded, the next time the user streams from a fleet on which application settings persistence is enabled with the applicable settings group, the larger application settings VHD is available.

Enable Regional Settings for Your AppStream 2.0 Users

AppStream 2.0 lets you or your users configure certain Windows settings that are specific to your users' location or language. AppStream 2.0 also lets you configure regional settings while creating Linux images. For more information, see Tutorial: Enable Japanese Support for Your Linux Images.

> **Note**
>
> Configuring regional settings by users is currently not supported for multi-session fleets.

Contents

- Configure Default Regional Settings for Your AppStream 2.0 Users
- Enable Your AppStream 2.0 Users to Configure Their Regional Settings

Configure Default Regional Settings for Your AppStream 2.0 Users

> **Note**
>
> The instructions on this page only apply to Windows fleets. Default regional settings are not supported for Elastic fleets.

In AppStream 2.0, users in a Windows stack can configure their streaming sessions to use settings that are specific to their location or language. For more information, see Enable Your AppStream 2.0 Users to Configure Their Regional Settings. You can also configure your fleets to use default settings that are specific to your users' location or language. In particular, you can apply the following Windows settings to your fleets:

- **Time Zone** — Determines the system time used by Windows and any applications that rely on the operating system time. AppStream 2.0 makes available the same options for this setting as Windows Server 2012 R2, Windows Server 2016, and Windows Server 2019.
- **Display Language** — Determines the display language used by the Windows operating system and certain Windows applications.

- **System Locale** — Determines the code pages (ANSI, MS-DOS, and Macintosh) and bitmap font files that Windows uses for non-Unicode applications in different languages.
- **User Locale** (also known as culture) — Determines the conventions used by Windows and any applications that query the Windows culture when formatting dates, numbers, or currencies or when sorting strings.
- **Input Method** — Determines the keystroke combinations that can be used to enter characters in another language.

Currently, AppStream 2.0 supports English and Japanese only for these language settings.

Contents

- Specify a Default Time Zone
- Specify a Default Display Language
- Specify a Default System Locale
- Specify a Default User Locale
- Specify a Default Input Method
- Special Considerations for Application Settings Persistence
- Special Considerations for Japanese Language Settings

Specify a Default Time Zone

To specify a default time zone to be used in your users' streaming sessions, perform the steps in either of the following two procedures.

Procedures

- Specify a Default Time Zone (Windows Server 2012 R2)
- Specify a Default Time Zone (Windows Server 2016, Windows Server 2019, and Windows Server 2022)

> **Note**
>
> Currently, AppStream 2.0 supports only **UTC** and **(UTC+9:00) Osaka, Sapporo, Tokyo**.

Specify a Default Time Zone (Windows Server 2012 R2)

1. Connect to the image builder that you want to use and sign in with a user that has local administrator permissions. To do so, do either of the following:

   - Use the AppStream 2.0 console (for web connections only)
   - Create a streaming URL (for web or AppStream 2.0 client connections)

   > **Note**
   >
   > If the image builder that you want to connect to is joined to an Active Directory domain and your organization requires smart card sign in, you must create a streaming URL and use the AppStream 2.0 client for the connection. For information about smart card sign in, see Smart Cards.

2. On the image builder desktop, choose the Windows **Start** button, and choose **Control Panel**.
3. Choose **Clock, Language, and Region**, then **Date and Time**, then **Change time zone**.
4. In the **Time zone** list, choose a time zone, and choose **OK**.
5. To apply any change to the time zone setting, restart your image builder. To do so, choose the Windows **Start** button, and choose **Windows PowerShell**. In PowerShell, use the **restart-computer** cmdlet.
6. While Windows restarts, the AppStream 2.0 login prompt displays. Wait for 10 minutes before you log in to the image builder again. Otherwise, you may receive an error. After 10 minutes, you can log in as **Administrator**.
7. If required, configure additional default regional or language settings. Otherwise, on the image builder desktop, open Image Assistant and install and configure applications for streaming.
8. After you finish configuring your image builder, follow the necessary steps in Image Assistant to finish creating your image. For information about how to create an image, see Tutorial: Create a Custom AppStream 2.0 Image by Using the AppStream 2.0 Console.
9. Do one of the following:

   - Create a new fleet and choose your new image for the fleet. For more information, see Create an AppStream 2.0 Fleet and Stack.
   - Update an existing fleet to use the new image.

10. Associate your fleet with the stack that is assigned to the users for whom you are configuring the default settings.

    The default time zone setting that you configured is applied to the fleet instances and user streaming sessions that are launched from those instances.

Specify a Default Time Zone (Windows Server 2016, Windows Server 2019, and Windows Server 2022)

1. Connect to the image builder that you want to use and sign in with an account that has local administrator permissions. To do so, do either of the following:

   - Use the AppStream 2.0 console (for web connections only)
   - Create a streaming URL (for web or AppStream 2.0 client connections)

   > **Note**
   >
   > If the image builder that you want to connect to is joined to an Active Directory domain and your organization requires smart card sign in, you must create a streaming URL and use the AppStream 2.0 client for the connection. For information about smart card sign in, see Smart Cards.

2. On the image builder desktop, choose the Windows **Start** button, and choose **Control Panel**.
3. Specify the default time zone by using PowerShell or the Windows user interface:

   - **PowerShell**
     - Open PowerShell and run the following command:

       ```
       Run Set-TimeZone -Id "Tokyo Standard Time"
       ```

       > **Note**
       >
       > To run this command, you must be logged in to the applicable computer as **Administrator**.

   - **Windows user interface**

   1. On the image builder desktop, choose the Windows **Start** button, and type **`timedate.cpl`** to open the **Date and Time** control panel item.
   2. Right-click the **Date and Time** icon, and choose **Run as administrator**.
   3. When prompted by **User Account Control** to choose whether you want to allow the app to make changes to your device, choose **Yes**.
   4. Choose **Change time zone**.
   5. In the **Time zone** list, choose a time zone, and choose **OK**.

4. If required, configure additional default regional or language settings. Otherwise, on the image builder desktop, open Image Assistant and install and configure applications for streaming.
5. After you finish configuring your image builder, follow the necessary steps in Image Assistant to finish creating your image. For information about how to create an image, see Tutorial: Create a Custom AppStream 2.0 Image by Using the AppStream 2.0 Console.
6. Do one of the following:
   - Create a new fleet and choose your new image for the fleet. For more information, see Create an AppStream 2.0 Fleet and Stack.
   - Update an existing fleet to use the new image.
7. Associate your fleet with the stack that is assigned to the users for whom you are configuring the default settings.

   The default time zone setting that you configured is applied to the fleet instances and user streaming sessions that are launched from those instances.

> **Note**
>
> Your users can change their time zone from the default setting that you configured. They can configure their regional settings during an application streaming session, as described in Enable Your AppStream 2.0 Users to Configure Their Regional Settings. Also, if a user previously selected a time zone when streaming from any fleet instance in the same AWS Region, the user-specified time zone setting automatically overrides any default time zone setting you specify through your image builder.

Specify a Default Display Language

There are two ways to specify the default display language for your users' streaming sessions. Use the AppStream 2.0 default application and Windows settings feature, or configure your image builder while signed in with an account that has local administrator permissions. The procedure in this section describes how to specify a default display language by using the AppStream 2.0 default application and Windows settings feature.

> **Note**
>
> Changing the display language in Windows also automatically changes the user locale and input method to match the language and region of the display language. If you want all three settings to match, you do not need to separately change the user locale or input method.

1. Connect to the image builder that you want to use and sign in with the **Template User** account. To do so, do either of the following:

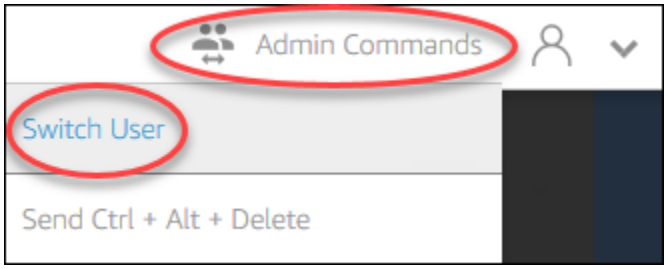
   - Use the AppStream 2.0 console (for web connections only)
   - Create a streaming URL (for web or AppStream 2.0 client connections)

   > **Note**
   >
   > If the image builder that you want to connect to is joined to an Active Directory domain and your organization requires smart card sign in, you must create a streaming URL and use the AppStream 2.0 client for the connection. For information about smart card sign in, see Smart Cards.

   Template User lets you create default application and Windows settings for your users. For more information, see "Creating Default Application and Windows Settings for Your AppStream 2.0 Users" in Default Application and Windows Settings and Application Launch Performance.
2. On the image builder desktop, choose the Windows **Start** button, and choose **Control Panel**.
3. Choose **Clock, Language, and Region**, then **Language**, **Add a language**.
4. Choose a language, and choose **Add**.

> **Note**
>
> Currently, AppStream 2.0 supports only **English (United States)** and **Japanese**.

5. The language that you selected appears in the list of languages you added to Windows. Choose the language that you just added. Then choose **Move up** until the language appears at the top of the language list.
6. Choose **Advanced Settings**. Under **Override for Windows display language**, choose your language from the list.
7. If you want to use the input method associated with the language that you added, under **Override for default input method**, choose the input method for the language.
8. Choose **Save**. When prompted to log off, choose **Log off now**.
9. When prompted, log in again to the image builder as **Template User**. Confirm that Windows is using the display language that you selected.
10. In the upper right area of the image builder desktop, choose **Admin Commands**, **Switch User**.



11. When prompted, log in as **Administrator**.
12. If required, configure additional default regional or language settings. Otherwise, on the image builder desktop, open Image Assistant and install and configure applications for streaming.
13. In Step 2 of the Image Assistant process, choose **Save settings**.
14. Follow the necessary steps in Image Assistant to finish creating your image. For information about how to create an image, see Tutorial: Create a Custom AppStream 2.0 Image by Using the AppStream 2.0 Console.
15. Do one of the following:
    - Create a new fleet and choose your new image for the fleet. For information, see Create an AppStream 2.0 Fleet and Stack.
    - Update an existing fleet to use the new image.

16. Associate your fleet with the stack that is assigned to the users for whom you are configuring the default settings.

    The default display language and associated user locale and input method settings that you configured are applied to the fleet instances and user streaming sessions that are launched from those instances.

    Alternatively, you can configure a default display language while logged in to the image builder as **Administrator**. If you chose different display languages while you were logged in under the **Template User** and **Administrator** accounts and you chose **Save settings** in Step 2 of the Image Assistant process, the **Template User** settings take precedence.

> **Note**
>
> Your users can change their user locale and input method from the default settings that you configured. They can change to any one of 11 different supported locales and nine different supported input methods. To do so, they can configure their regional settings during application streaming sessions, as described in Enable Your AppStream 2.0 Users to Configure Their Regional Settings. Also, if a user previously selected a user locale or input method when streaming from any fleet instance in the same Region, those user-specified settings automatically override any default user locale and input method that you specify through your image builder.

Specify a Default System Locale

To specify a default system locale for your users' streaming sessions, perform the following steps.

1. Connect to the image builder that you want to use and sign in with an account that has local administrator permissions. To do so, do either of the following:

    - Use the AppStream 2.0 console (for web connections only)
    - Create a streaming URL (for web or AppStream 2.0 client connections)

    > **Note**
    >
    > If the image builder that you want to connect to is joined to an Active Directory domain and your organization requires smart card sign in, you must create a

streaming URL and use the AppStream 2.0 client for the connection. For information about smart card sign in, see Smart Cards.

2. On the image builder desktop, choose the Windows **Start** button, and choose **Control Panel**.
3. Choose **Clock, Language, and Region**, then **Region**.
4. In the **Region** dialog box, choose the **Formats** tab.
5. Choose **Change system locale**.
6. In the **Region Settings** dialog box, in the **Current system locale** list, choose a language and region.

> **Note**
>
> Currently, AppStream 2.0 supports only **English (United States)** and **Japanese (Japan)**.

7. Choose **OK** to close the **Region Settings** dialog box, and choose **OK** again to close the **Region** dialog box.
8. When prompted to restart your computer, allow Windows to restart.
9. While Windows restarts, the AppStream 2.0 login prompt displays. Wait for 10 minutes before you log in to the image builder again. Otherwise, you may receive an error. After 10 minutes, you can log in as **Administrator**.
10. If required, configure additional default regional or language settings. Otherwise, on the image builder desktop, open Image Assistant and install and configure applications for streaming. After you finish configuring your image builder, follow the necessary steps in Image Assistant to finish creating your image. For information about how to create an image, see Tutorial: Create a Custom AppStream 2.0 Image by Using the AppStream 2.0 Console.
11. Do one of the following:
    - Create a new fleet and choose your new image for the fleet. For more information, see Create an AppStream 2.0 Fleet and Stack.
    - Update an existing fleet to use the new image.
12. Associate your fleet with the stack that is assigned to the users for whom you are configuring the default settings.

    The default system locale setting that you configured is applied to the fleet instances and user streaming sessions that are launched from those instances.

Specify a Default User Locale

To specify a default user locale for your users' streaming sessions, perform the following steps.

> **Note**
>
> If you plan to configure the display language and you want the user locale and display language to match, you do not need to change the user locale. Changing the display language automatically changes the user locale to match.

1. Connect to the image builder that you want to use and sign in with an account that has local administrator permissions. To do so, do either of the following:

   - Use the AppStream 2.0 console (for web connections only)
   - Create a streaming URL (for web or AppStream 2.0 client connections)

   > **Note**
   >
   > If the image builder that you want to connect to is joined to an Active Directory domain and your organization requires smart card sign in, you must create a streaming URL and use the AppStream 2.0 client for the connection. For information about smart card sign in, see Smart Cards.

2. On the image builder desktop, choose the Windows **Start** button, and choose **Control Panel**.
3. Choose **Clock, Language, and Region**, then **Region**.
4. In the **Region** dialog box, choose the **Formats** tab.
5. In the **Format** list, choose a language and region.

   > **Note**
   >
   > Currently, AppStream 2.0 supports only **English (United States)** and **Japanese (Japan)**.

6. Choose **OK** to close the **Region** dialog box.
7. If required, configure additional default regional or language settings. Otherwise, on the image builder desktop, open Image Assistant and install and configure applications for streaming.

8. In Step 2 of the Image Assistant process, choose **Save settings**.
9. Follow the necessary steps in Image Assistant to finish creating your image. For information about how to create an image, see Tutorial: Create a Custom AppStream 2.0 Image by Using the AppStream 2.0 Console.
10. Do one of the following:
    - Create a new fleet and choose your new image for the fleet. For more information, see Create an AppStream 2.0 Fleet and Stack.
    - Update an existing fleet to use the new image.
11. Associate your fleet with the stack that is assigned to the users for whom you are configuring the default settings.

    The default user locale setting that you configured is applied to the fleet instances and user streaming sessions that are launched from those instances.

> **Note**
>
> Your users can change their user locale from the default setting that you configured to any one of 11 different supported locales. To do so, they can configure their regional settings during application streaming sessions, as described in Enable Your AppStream 2.0 Users to Configure Their Regional Settings. Also, if a user previously selected a user locale when streaming from any fleet instance in the same Region, that user-specified setting automatically overrides any default user locale setting that you specify through your image builder.

Specify a Default Input Method

To specify a default input method to be used in your users' streaming sessions, perform the following steps.

> **Note**
>
> If you plan to configure the display language, and you want the input method and display language to match, you do not need to change the input method. Changing the display language in Windows also automatically changes the user locale and input method to

match the language and region of the display language. If you want all three settings to match, you do not need to separately change the user locale or input method.

1. Connect to the image builder that you want to use and sign in with an account that has local administrator permissions. To do so, do either of the following:
   - Use the AppStream 2.0 console (for web connections only)
   - Create a streaming URL (for web or AppStream 2.0 client connections)

   > **Note**
   >
   > If the image builder that you want to connect to is joined to an Active Directory domain and your organization requires smart card sign in, you must create a streaming URL and use the AppStream 2.0 client for the connection. For information about smart card sign in, see Smart Cards.

2. On the image builder desktop, choose the Windows **Start** button, and choose **Control Panel**.
3. Choose **Clock, Language, and Region**, then **Language**, **Add a language**.
4. Choose a language, and choose **Add**.

   > **Note**
   >
   > Currently, AppStream 2.0 supports only **English (United States)** and **Japanese**.

5. The language that you chose appears in the list of languages you added to Windows.
6. Choose **Advanced Settings**. Under **Override for default input method**, choose the input method for the language you added.
7. Choose **Save**.
8. Log off and log in again. To do so, choose the Windows **Start** button on the image builder desktop. Choose **ImageBuilderAdmin**, **Sign out**. When prompted, log in as Administrator.
9. If required, configure additional default regional or language settings. Otherwise, on the image builder desktop, open Image Assistant and install and configure applications for streaming.
10. In Step 2 of the Image Assistant process, choose **Save settings**.

11. Follow the necessary steps in Image Assistant to finish creating your image. For information about how to create an image, see Tutorial: Create a Custom AppStream 2.0 Image by Using the AppStream 2.0 Console.
12. Do one of the following:
    - Create a new fleet and choose your new image for the fleet. For information, see Create an AppStream 2.0 Fleet and Stack.
    - Update an existing fleet to use the new image.
13. Associate your fleet with the stack that is assigned to the users for whom you are configuring the default settings.

    The default input method that you configured is applied to the fleet instances and user streaming sessions that are launched from those instances.

> **Note**
>
> Your users can change their input method from the default setting that you configured to any one of nine different supported input methods. They can configure this setting by configuring their regional settings during application streaming sessions, as described in Enable Your AppStream 2.0 Users to Configure Their Regional Settings. Also, if a user previously selected an input method when streaming from any fleet instance in the same Region, that user-specified setting automatically overrides any default input method that you specify through your image builder.

Special Considerations for Application Settings Persistence

When you create a stack in the AppStream 2.0 console, in **Step 3: User Settings**, if you use the same settings group under **Application settings persistence** as another stack that uses different regional settings, only one set of regional settings is used for both stacks. For each user, the default regional settings for the stack that the user logs into first automatically override the default regional settings of any other stacks in the same application settings group. To avoid this problem, do not use the same application settings group for two different stacks that have different regional settings.

Special Considerations for Japanese Language Settings

This section describes key points to keep in mind when configuring Japanese language settings for your AppStream 2.0 users.

AWS CLI

Changing the Windows system locale to Japanese requires that your image builder have AWS Command Line Interface (AWS CLI) version 1.16.30 or later installed. To update the version of AWS CLI on your image builder, follow the steps in Installing the AWS Command Line Interface.

Japanese Keyboards

If your image builder input method is set to Japanese when you create an image, AppStream 2.0 automatically configures your image to use a Japanese keyboard. Any fleets that use the image are also automatically configured to use Japanese keyboards. However, if you want to use a Japanese keyboard within your image builder session, update the following registry settings for the HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\i8042prt\Parameters registry key:

Name	Type	Data
LayerDriver JPN	REG_SZ	kbd106.dll
OverrideKeyboardIdentifier	REG_SZ	PCAT_106KEY
OverrideKeyboardSubtype	DWORD	2
OverrideKeyboardType	DWORD	7

After changing these settings, restart your image builder. To do so, choose the Windows **Start** button, and choose **Windows PowerShell**. In PowerShell, use the **restart-computer** cmdlet.

Enable Your AppStream 2.0 Users to Configure Their Regional Settings

> **Note**
>
> Enabling users to configure regional settings is currently not supported in Linux-based streaming sessions or streaming sessions powered by multi-session fleets.

Users can configure their Amazon AppStream 2.0 Windows streaming sessions to use settings that are specific to their location or language. In particular, users can configure the following settings:

- **Time zone** — Determines the system time used by Windows and any applications that rely on the operating system time. AppStream 2.0 makes available the same options for this setting as the Windows Server version used in your fleet.
- **Locale** (also known as culture) — Determines the conventions used by Windows and any applications that query the Windows culture when formatting dates, numbers, or currencies or when sorting strings. For a list of locales that AppStream 2.0 supports, see Supported Locales.
- **Input method** — Determines the keystroke combinations that can be used to input characters in another language.

If users change regional settings during their streaming sessions, the changes are applied to any future streaming sessions in the same AWS Region.

> **Note**
>
> For guidance that you can provide your users to help them get started with configuring their regional settings, see Configure Regional Settings.

Contents

- Supported Locales
- Enable Regional Settings for Your AppStream 2.0 Users

Supported Locales

AppStream 2.0 supports the following locales:

Locale	Language culture name
Chinese (Simplified, China)	zh-CN
Chinese (Simplified, Singapore)	zh-SG
Chinese (Traditional)	zh-TW
Dutch (The Netherlands)	nl-NL
English (Australia)	en-AU
English (Canada)	en-CA
English (United Kingdom)	en-GB
English (United States)	en-US
French (France)	fr-FR
German (Germany)	de-DE
Italian (Italy)	it-IT
Japanese (Japan)	ja-JP
Korean (Korea)	ko-KR
Portuguese (Brazil)	pt-BR
Spanish (Spain, International Sort)	es-ES
Thai (Thailand)	th-TH

Enable Regional Settings for Your AppStream 2.0 Users

To enable users to configure regional settings for a given stack during their AppStream 2.0 streaming sessions, your stack must be associated with a fleet based on an image that uses a version of the AppStream 2.0 agent released on or after June 6, 2018. For more information, see AppStream 2.0 Agent Release Notes. Additionally, your image must have Windows PowerShell 5.1 or later installed. Images created from AppStream 2.0 base images published on or after June 12, 2018 meet both criteria. Images created from AppStream 2.0 base images published before June 12, 2018 do not have Windows PowerShell 5.1 by default.

To update an existing image to include Windows PowerShell 5.1

1. Launch a new image builder using your existing image as the base image by doing the following:

   a. In the left navigation pane in the AppStream 2.0 console, choose **Images**.

   b. Choose the **Image Builder** tab, **Launch Image Builder**, and then select your existing image.

   c. If you are prompted to update the AppStream 2.0 agent when you launch the image builder, select the check box, and then choose **Start**.

2. After your image builder is running, connect to it and sign in with an account that has local administrator permissions. To do so, do either of the following:

   - Use the AppStream 2.0 console (for web connections only)
   - Create a streaming URL (for web or AppStream 2.0 client connections)

   > **Note**
   >
   > If the image builder that you want to connect to is joined to an Active Directory domain and your organization requires smart card sign in, you must create a streaming URL and use the AppStream 2.0 client for the connection. For information about smart card sign in, see Smart Cards.

3. From the image builder desktop, open Windows PowerShell. Choose the Windows **Start** button, and then choose **Windows PowerShell**.

4. At the PowerShell command prompt, type the command `$PSVersionTable` to determine the version of Windows PowerShell that is installed on your image builder. If your image builder does not include Windows PowerShell 5.1 or later, use the following steps to install it.
5. Open a web browser and follow the steps in Install and Configure WMF 5.1 in the Microsoft documentation, making sure that you download the Windows Management Framework (WMF) 5.1 package for Windows Server 2012 R2. WMF 5.1 includes Windows PowerShell 5.1.
6. At the end of the WMF 5.1 installation process, the installer prompts you to restart your computer. Choose **Restart Now** to restart the image builder.
7. Wait about 10 minutes before logging in to your image builder, even though AppStream 2.0 prompts you to do so immediately. Otherwise, you might encounter an error.
8. After logging in to your image builder again, open Windows PowerShell and type the command `$PSVersionTable` to confirm that Windows PowerShell 5.1 is installed on your image builder.
9. Use the image builder to create a new image. This new image now includes the latest versions of the AppStream 2.0 agent and Windows PowerShell.
10. Update your fleet to use the new image by doing the following:

    a. In the left navigation pane in the AppStream 2.0 console, choose **Fleets**, and then choose the fleet associated with the stack for which you want to enable regional settings.

    b. On the **Fleet Details** tab, choose **Edit**.

    c. In **Image name,** choose the new image to use for the fleet.

For more information about using image builders to create images, see Tutorial: Create a Custom AppStream 2.0 Image by Using the AppStream 2.0 Console.

Manage Application Entitlements

Amazon AppStream 2.0 can dynamically build the application catalog to display the AppStream 2.0 applications that users are entitled to access. Application entitlements can be assigned based on attributes using a third-party SAML 2.0 identity provider, or by using the AppStream 2.0 Dynamic Application Framework. The following sections describe how to manage application entitlements.

> **Note**
>
> Attribute-based application entitlements using a third-party SAML 2.0 identity provider is recommended for most scenarios. If you would like to use an existing Dynamic App Provider that manages application package delivery in addition to entitlement, so that applications do not need to be installed in an AppStream 2.0 image, Dynamic Application Framework is recommended. For more information, see Additional Resources for Learning About Dynamic App Providers and the Dynamic Application Framework.

Contents

- Attribute-Based Application Entitlements Using a Third-Party SAML 2.0 Identity Provider
- Application Entitlements from a Dynamic App Provider Using the Dynamic Application Framework

Attribute-Based Application Entitlements Using a Third-Party SAML 2.0 Identity Provider

Application entitlements control access to specific applications within your AppStream 2.0 stacks. This works by using SAML 2.0 attribute assertions from a third-party SAML 2.0 identity provider. The assertion is matched to a value when a user identity federates to an AppStream 2.0 2.0 SAML application. If the entitlement is true, and the attribute name and value match, access is entitled for the user identity to one or more applications within the stack.

Attribute-based application entitlements using a third-party SAML 2.0 identity provider do not apply in the following scenarios. In other words, the entitlement is ignored in cases such as the following:

- AppStream 2.0 user pool authentication. For more information, see AppStream 2.0 User Pools.

- AppStream 2.0 streaming URL authentication. For more information, see Streaming URL.
- The desktop application when AppStream 2.0 fleets are configured for **Desktop Stream view**. For more information, see Create an AppStream 2.0 Fleet and Stack.
- Stacks using the Dynamic Application Framework. Dynamic Application Framework provides separate application entitlement features. For more information, see Application Entitlements from a Dynamic App Provider Using the Dynamic Application Framework.
- When users federate to the AppStream 2.0 application catalog, application entitlements will only display the applications the user is entitled to. Applications are not restricted from running within the AppStream 2.0 session. For example, in a fleet configured for Desktop Stream view, a user can launch an application directly from the desktop.

Create Application Entitlements

Before you create application entitlements, you must do the following:

- Create an AppStream 2.0 fleet and stack with an image containing one or more applications (Always-On or On-Demand fleet) or assigned applications (Elastic fleet) that will meet your needs. For more information, see Create an AppStream 2.0 Fleet and Stack.
- Provide user access to the stack using a third-party SAML 2.0 identity provider. For more information, see Amazon AppStream 2.0 Integration with SAML 2.0. If you are using an existing SAML 2.0 identity provider that you setup previously, see Step 2: Create a SAML 2.0 Federation IAM Role for the steps to add the sts:TagSession permission to your IAM role trust policy. For more information, see Passing session tags in AWS STS. This permission is required to use application entitlements.

To create an application entitlement

1. Open the AppStream 2.0 console.
2. In the left navigation pane, choose **Stacks**, and select the stack for which to manage application entitlements.
3. In the **Application Entitlements** dialog box, choose **Create**.
4. Enter a **Name** and **Description** for your entitlement.
5. Define the attribute name and value for your entitlement.

When mapping attributes, specify the attribute in the format https://aws.amazon.com/SAML/Attributes/PrincipalTag:{TagKey}, where {TagKey} is one of the following attributes:

- roles
- department
- organization
- groups
- title
- costCenter
- userType

The attributes that you defined are used to entitle applications in your stack to a user when they federate into an AppStream 2.0 session. Entitlement works by matching the attribute name to a key value name in the SAML assertion created during federation. For more information, see SAML PrincipalTag Attribute.

> **Note**
>
> One or more values can be included in any supported attribute, separated by a colon (:).
> For example, groups information can be passed in a SAML attribute name https://aws.amazon.com/SAML/Attributes/PrincipalTag:groups with value "group1:group2:group3" and your entitlement can allow applications based on a single group value, i.e. "group1". For more information, see SAML PrincipalTag Attribute.

6. Configure application settings in your stack to entitle all applications, or select applications. Choosing **All applications (*)** applies all applications available on the stack, including applications that are added in the future. Choosing **Select applications** will filter on specific application names.
7. Review your settings and create your entitlement. You can repeat the process and create additional entitlements. Entitlement to applications in a stack will be a union of all entitlements that match the user based on attribute names and values.
8. In your SAML 2.0 identity provider, configure your AppStream 2.0 SAML application attribute mappings to send the attribute and value defined in your entitlement. When users federate

to the AppStream 2.0 application catalog, application entitlements will only display the applications the user is entitled to.

SAML 2.0 Multi-Stack Application Catalog

With attribute-based application entitlements using a third-party SAML 2.0 identity provider, you can enable access to multiple stacks from a single relay state URL. Remove the stack and app (if present) parameters from the relay state URL, as follows:

```
https://relay-state-region-endpoint?accountId=aws-account-id-without-hyphens
```

When users federate to the AppStream 2.0 application catalog, they will be presented with all of the stacks where application entitlements have matched one or more applications to the user for the account ID and relay state endpoint associated with the Region in which your stacks are located. When a user selects a catalog, application entitlements will only display the applications the user is entitled to. For more information, see Step 6: Configure the Relay State of Your Federation.

> **Note**
>
> To use SAML 2.0 Multi-Stack Application Catalogs, you need to configure the inline policy for your SAML 2.0 Federation IAM Role. For more information, see Step 3: Embed an Inline Policy for the IAM Role.

Application Entitlements from a Dynamic App Provider Using the Dynamic Application Framework

> **Note**
>
> Managing application entitlement with the Dynamic Application Framework is currently not supported for Linux-based stacks.

Amazon AppStream 2.0 supports dynamically building the application catalog that displays for your users when they stream from an AppStream 2.0 stack. You can use the API operations

provided by AppStream 2.0 to develop a dynamic app provider that modifies, in real time, the applications that users can access on the streaming instance. Alternatively, you can implement a third-party dynamic app provider that uses these API operations.

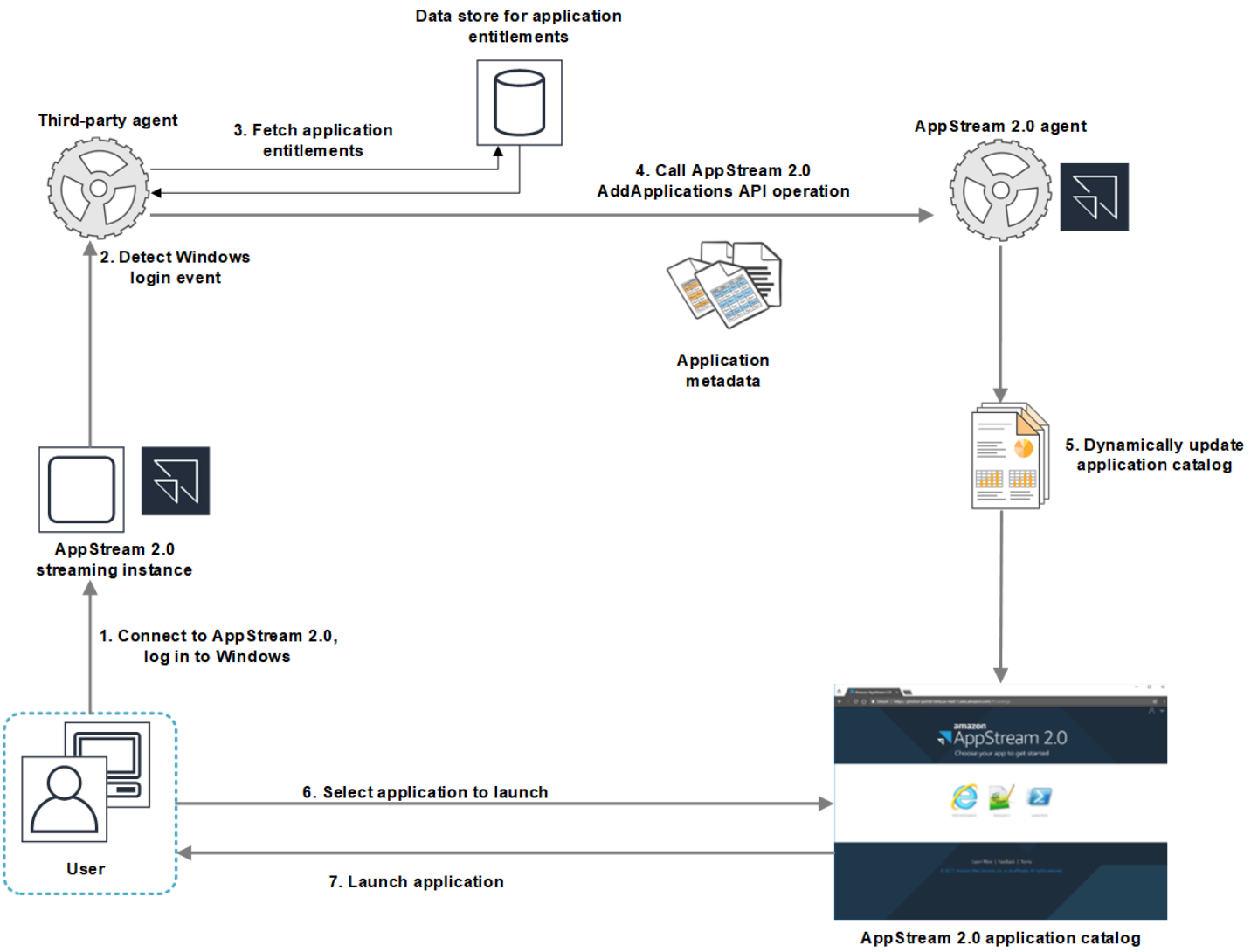
> **Note**
>
> This feature requires an AppStream 2.0 Always-On or On-Demand fleet that is joined to a Microsoft Active Directory domain. For more information, see Using Active Directory with AppStream 2.0. This feature is not available on multi-session fleets.

Contents

- Example API Operations Work Flow for the Dynamic Application Framework
- Use the AppStream 2.0 Dynamic Application Framework to Build a Dynamic App Provider
- Enable and Test Dynamic App Providers
- Additional Resources for Learning About Dynamic App Providers and the Dynamic Application Framework

Example API Operations Work Flow for the Dynamic Application Framework

The following diagram is an example of the API operations flow between AppStream 2.0 and a third-party application provider.



1. The user connects to AppStream 2.0. A fleet streaming instance is assigned to the user and Windows login occurs.
2. Your service or agent detects the Windows logon event and determines the user who is logging in to Windows.
3. The service or agent fetches the application entitlements for the user. In the example diagram, the application entitlements are stored in a database. This information can be stored and retrieved in different ways. For example, application entitlements may be fetched from server software, or group names in Active Directory may be parsed to locate the application identifiers (IDs).
4. Your dynamic app provider calls the AppStream 2.0 agent `AddApplications` API operation with the application metadata for the applications that the user should have.

5. The AppStream 2.0 agent dynamically updates the application catalog with the modified application list.
6. The user selects an application to launch.
7. The application is launched by using the application metadata specified by your service or agent.

From the user's perspective, the process happens transparently. The user connects to AppStream 2.0 and logs in to the fleet instance. After login, the list of applications specified in the image and provided by your dynamic app provider displays for the user.

Use the AppStream 2.0 Dynamic Application Framework to Build a Dynamic App Provider

The AppStream 2.0 dynamic application framework provides API operations within an AppStream 2.0 streaming instance that you can use to build a dynamic app provider. Dynamic app providers can use the API operations provided to modify the catalog of applications that your users can access in real time. The applications managed by the dynamic app providers can be within the image, or they can be off-instance, such as from a Windows file share or an application virtualization technology.

> **Note**
>
> This feature requires an AppStream 2.0 Always-On or On-Demand fleet that is joined to a Microsoft Active Directory domain. For more information, see Using Active Directory with AppStream 2.0.

Contents

- About the Dynamic Application Framework
- Dynamic Application Framework Thrift Definitions and Named Pipe Name
- API Actions for Managing App Entitlement for AppStream 2.0

About the Dynamic Application Framework

The dynamic application framework uses the Apache Thrift software framework for inter-process messaging. It is exposed through Named Pipes in Windows. Using the Thrift framework allows you to build your dynamic app provider in your software development language of choice.

The dynamic application framework consists of three API operations: `AddApplications`, `RemoveApplications`, and `ClearApplications`.

Dynamic Application Framework Thrift Definitions and Named Pipe Name

Thrift enables you to use simple definition files provided by AppStream 2.0 to compile RPC clients. The RPC clients let you communicate with the AppStream 2.0 agent software running on a streaming instance. For information about how to compile the RPC client for your language, see the Apache Thrift documentation. After you compile the Thrift libraries for the language of your choice, build a Thrift client by using the Named Pipe transport. Use D56C0258-2173-48D5-B0E6-1EC85AC67893 as the pipe name.

AppStreamServer.thrift

```
namespace netstd AppStream.ApplicationCatalogService.Model

const string ServiceEndpoint = "D56C0258-2173-48D5-B0E6-1EC85AC67893";


struct AddApplicationsRequest
{
    1: required string userSid;
    2: required list<Application> applications;
}

struct AddApplicationsResponse
{
}

struct RemoveApplicationsRequest
{
    1: required string userSid;
    2: required list<string> applicationIds;
}

struct RemoveApplicationsResponse
{
}

struct ClearApplicationsRequest
{
    1: required string userSid;
}
```

```
struct ClearApplicationsResponse
{
}

struct Application
{
    1: required string id;
    2: required string displayName;
    3: required string launchPath;
    4: required string iconData;
    5: string launchParams;
    6: string workingDirectory;
}

exception AppStreamClientException
{
    1: string errorMessage,
    2: ErrorCode errorCode
}

exception AppStreamServerException
{
    1: string errorMessage,
    2: ErrorCode errorCode
}

enum ErrorCode
{
}

service ApplicationCatalogService
{
    AddApplicationsResponse AddApplications(1:AddApplicationsRequest request)
    throws (1: AppStreamClientException ce, 2: AppStreamServerException se),

    RemoveApplicationsResponse RemoveApplications(1:RemoveApplicationsRequest request)
    throws (1: AppStreamClientException ce, 2: AppStreamServerException se),

    ClearApplicationsResponse ClearApplications(1:ClearApplicationsRequest request)
    throws (1: AppStreamClientException ce, 2: AppStreamServerException se),
}
```

API Actions for Managing App Entitlement for AppStream 2.0

You can use the following API operations to manage application entitlement for AppStream 2.0.

`AddApplicationsRequest` operation

Adds applications to the application catalog for AppStream 2.0 users. The application catalog displayed by AppStream 2.0 includes the applications that you add by using this API operation and the applications that you add in the image. After you add applications by using one or both of these methods, your users can launch the applications.

Request syntax

```
string userSid;

list<Application> applications;
```

Request parameters

`userSid`

The SID of the user who the request applies to.

Type: String

Required: Yes

Length constraints: Minimum length of 1, maximum length of 208 characters.

`applications`

The list of applications that the request applies to.

Type: String

Required: Yes

`Application` object

Describes the application metadata required to display and launch the application. The application identifier must be unique and not in conflict with other applications specified through the API operation or the image.

id

The identifier of the application being specified. This value, which corresponds to the `application_name` value in an AppStream 2.0 applications report, is provided when a user launches the application. When you enable usage reports, for each day that users launch at least one application during their streaming sessions, AppStream 2.0 exports an applications report to your Amazon S3 bucket. For more information about applications reports, see Applications Report Fields.

Type: String

Required: Yes

Length constraints: Minimum length of 1, maximum length of 512 characters.

displayName

The display name of the application being specified. This name is displayed to the user in the application catalog.

Type: String

Required: Yes

Length constraints: Minimum length of 1, maximum length of 512 characters.

launchPath

The Windows file system path to the executable of the application to be launched.

Type: String

Required: Yes

Length constraints: Minimum length of 1, maximum length of 32,767 characters.

iconData

The base-64 encoded image to display in the application catalog. The image must be in one of the following formats: .png, .jpeg, or .jpg.

Type: String

Required: Yes

Length constraints: Minimum length of 1, maximum length of 1,000,000 characters.

launchParams

The parameters used to launch the application.

Type: String

Required: No

Length constraints: Maximum length of 32,000 characters.

workingDirectory

The Windows file system path to the working directory the application should be launched in.

Type: String

Required: No

Length constraints: Maximum length of 32,767 characters.

`RemoveApplicationsRequest` operation

Removes applications that were added by using the `AddApplicationsRequest` operation. The applications are removed from the application catalog for the user. After applications are removed, they can't be launched. If an application is still running, AppStream 2.0 does not close it. Applications that are specified directly in the AppStream 2.0 image can't be removed.

Request syntax

```
string userSid;

list<Application> applications;
```

Request parameters

userSid

The SID of the user the request applies to.

Type: String

Required: Yes

Length constraints: Minimum length of 1, maximum length of 208 characters.

applications

The list of applications that the request applies to.

Type: String

Required: Yes

`ClearApplicationsRequest` operation

Removes all applications that were added to the application catalog by using the `AddApplicationsRequest` operation. After applications are removed, they can't be launched. If the applications are running when the `ClearApplicationsRequest` operation is used, AppStream 2.0 does not close them. Applications that are specified directly in the AppStream 2.0 image can't be removed.

Request syntax

```
string userSid;
```

Request parameters

userSid

The SID of the user the request applies to.

Type: String

Required: Yes

Length constraints: Minimum length of 1, maximum length of 208 characters.

Enable and Test Dynamic App Providers

Dynamic app providers must first be enabled within an AppStream 2.0 image. After you enable these providers, they can manage applications for users on the streaming instance.

Enable Dynamic App Providers

To enable this capability, you must add your dynamic app provider details to a configuration file on the image builder. The image builder must be joined to a Microsoft Active Directory domain.

Perform the following steps on an image builder, then you can test your dynamic apps to verify that they function as expected. Finally, finish creating your image.

> **Note**
>
> Third-party dynamic app providers may modify the configuration file during install. For installation instructions, see the documentation for the applicable provider.

To enable dynamic app providers

1. Connect to the image builder that you want to use and sign in with a domain account that has local administrator permissions on the image builder. To do so, do either of the following:

   - Use the AppStream 2.0 console (for web connections only)
   - Create a streaming URL (for web or AppStream 2.0 client connections)

   > **Note**
   >
   > If your organization requires smart card sign in, you must create a streaming URL and use the AppStream 2.0 client for the connection. For information about smart card sign in, see Smart Cards.

2. Navigate to C:\ProgramData\Amazon\AppStream\AppCatalogHelper\DynamicAppCatalog\, and open the **Agents.json** configuration file.
3. In the **Agents.json** file, add the following entries:

   "DisplayName": "<Uninstall hive display name value>",

   "Path": "<C:\path\to\client\application>"

   DisplayName must match the **DisplayName** registry value for the **HKEY_LOCAL_MACHINE \Software\Microsoft\Windows\CurrentVersion\Uninstall** key created for your application.
4. Install your dynamic app provider.
5. On the image builder desktop, open Image Assistant.
6. Optionally, install any other applications that you want to include in the image.
7. In Image Assistant, on the **1. Add Apps** page, select the **Enable dynamic app providers** check box.

8. On the same page, if you installed other applications as described in step 8, choose **+Add App**, and specify the applications to add.

> **Note**
>
> When you use a dynamic app provider, you don't need to specify any applications in the image. If you specify applications in the image, they can't be removed by dynamic app providers.

9. Proceed to the steps in the next section to test your dynamic app provider.

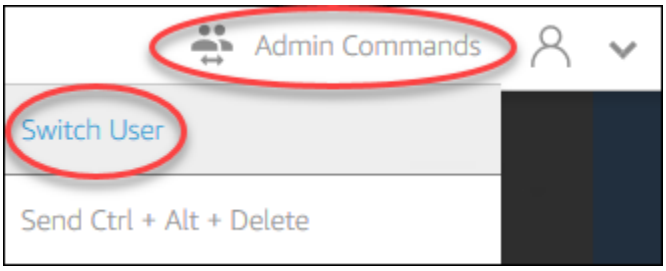
Test Dynamic App Providers (Optional)

After you enable your dynamic app provider on an image builder, you can test the provider to verify that it functions as expected. To do so, perform the following steps before you finish creating the image.

To test dynamic app providers

1. Do one of the following:

   - If you are already connected to the image builder on which you enabled dynamic app providers and you are logged on as **Administrator,** you must switch to an account that does not have local administrator permissions on the image builder. To do so, in the upper right corner of the image builder session toolbar, choose **Admin Commands**, **Switch User**.

   - If you are not already connected to the image builder, connect by either using the AppStream 2.0 console (for web connections only) or creating a streaming URL (for web or AppStream 2.0 client connections).

> **Note**
>
> When you are prompted to sign in, choose **Directory User**, and sign in with a domain account that does not have local administrator permissions on the image builder.

2. On the image builder desktop, open Image Assistant, if it is not already open.
3. On the **Test Apps** page, if you specified any applications in the image that are not from the dynamic app provider, they display first in the list. It may take a few moments for applications from dynamic app providers to appear in the list.
4. Choose an application from the list and open it to verify that it functions as expected.
5. After you finish testing, in the lower right corner of the **Test Apps** page, choose **Switch user**.
6. Choose **Administrator**, and log back into the image builder.
7. Follow the necessary steps in Image Assistant to finish creating your image. For information about how to create an image, see Tutorial: Create a Custom AppStream 2.0 Image by Using the AppStream 2.0 Console.

   AppStream 2.0 automatically optimizes the agents that are specified in the **Agents.json** configuration file.

Additional Resources for Learning About Dynamic App Providers and the Dynamic Application Framework

The following links provide information to help you learn more about dynamic app providers and the dynamic application framework.

Solution	Description
Liquidware FlexApp	FlexApp — Provides an overview of Liquidware FlexApp. FlexApp is a third-party provider that uses the AppStream 2.0 dynamic application framework to manage applicati on entitlements and delivery in real time. FlexApp layering delivers applications to any Windows desktop environment, independent of the Windows operating system version or delivery platform.

Solution	Description
App-V	Bring your App-V packages to AppStream 2.0 with the dynamic application framework — Describes how to integrate App-V with AppStream 2.0 by using the dynamic application framework.
AppStream 2.0	Use the AppStream 2.0 Dynamic Application Framework to Build a Dynamic App Provider — Describes how to use the AppStream 2.0 dynamic application framework to develop your own dynamic app provider.

Provide Your Users with Access to AppStream 2.0

Users can access AppStream 2.0 streaming sessions by using either a web browser or the AppStream 2.0 client on a supported device.

Depending on your organizational requirements, you can enable user access to AppStream 2.0 streaming sessions by: Setting up identity federation using SAML 2.0, using an AppStream 2.0 user pool, or creating a streaming URL. Following are recommendations for choosing a connection method.

- SAML 2.0: Use this connection method when you have an identity provider that manages your users and supports SAML 2.0 federation.

  > **Note**
  >
  > This connection method is required when your AppStream 2.0 fleet is joined to a Microsoft Active Directory domain.

- AppStream 2.0 user pools: Use this connection method when:
  - You want to set up a Proof-of-Concept (POC) quickly before you configure your SAML 2.0-compliant identity provider.
  - You don't have a SAML 2.0-compliant identity provider.
  - You want to manage users directly within the AppStream 2.0 console.
- Streaming URL: Use this connection method when you want to programmatically provide access to AppStream 2.0 by using temporary URLs. We recommend this connection method when you want to use your existing identity provider to provide programmatic access to AppStream 2.0

Supported Features

The following table compares the features that are supported by the different access types.

Feature	Browser-Based Access	Client-Based Access	Notes
Trusted devices support	N/A	N/A	Handled at IdP level

Feature	Browser-Based Access	Client-Based Access	Notes
Client access restrictions (OS-based)	N/A	N/A	Handled at IdP level
USB redirection	✗	✓	Supported on AppStream 2.0 client accessing Windows-based fleets. For more information, see the section called "USB Redirection".
Audio input (for web conferences and calls)	✓	✓	Not supported on Linux. AppStream 2.0 supports USB microphones.
Video input (conferencing applications)	✓	✓	
Storage redirection	✗	✓	For more information, see the section called "Enable File System Redirection".

Feature	Browser-Based Access	Client-Based Access	Notes
USB/local printer redirection	Available with exceptions	✓	AppStream 2.0 indirect printing on browser. Full redirection not supported on Linux-based stacks.
Clipboard redirection	✓	✓	
HIPAA/PCI compliance	✓	✓	For more information, see Compliance.
Active Directory authentication	✓	✓	For more information, see *Using Active Directory* .
MFA (multi-factor authentication)	✓	✓	For AppStream 2.0, MFA is supported via SAML 2.0.
Smart card (CAC and PIV readers)	✗	✓	For more information, see the section called "Smart Card Redirection".

Feature	Browser-Based Access	Client-Based Access	Notes
Certificates for access control (OS-based)	✓	✓	For AppStream 2.0, certificate authentication is supported via SAML 2.0.
Encryption at rest	N/A	N/A	For more information, see Encryption.
Client customization	Available with exceptions	Available with exceptions	AppStream 2.0 supports web-based branding and custom URLs. For more information, see *Add Your Custom Branding to AppStream 2.0*.

Feature	Browser-Based Access	Client-Based Access	Notes
YubiKey support	✗	Available with exceptions	Supported on AppStream 2.0 client. For more information, see the section called "Qualify USB Devices for Use with Streaming Applications".
Monitor support	Web access dual-monitor support. For more information, see the section called "Dual-Monitor Support".	Windows multi-monitor support. For more information, see the section called "Multiple Monitors".	

The next topics provide information about how to configure user access to AppStream 2.0 for application streaming.

For guidance that you can provide your users to help them get started with application streaming, see Guidance for AppStream 2.0 Users.

Provide Access Through a Web Browser

Your users can start an AppStream 2.0 streaming session by using a web browser or the AppStream 2.0 client application for a supported device. The following topics provide information to help you provide user access through a web browser.

Contents

- System Requirements and Feature Support (Web Browser)
- Configure a Connection Method for Your AppStream 2.0 Users (Web Browser)

For information about how to provide user access to AppStream 2.0 through the AppStream 2.0 client, see Provide Access Through the AppStream 2.0 Client for Windows.

System Requirements and Feature Support (Web Browser)

This topic provides information to help you understand the requirements for providing user access to AppStream 2.0 through a web browser. It also provides information about supported features.

System Requirements and Considerations

Users can access AppStream 2.0 through an HTML5-capable web browser on a desktop computer such as a Windows, Mac, Chromebook, or Linux computer. HTML5-capable web browsers that can be used include the following:

- Google Chrome
- Mozilla Firefox
- Safari
- Microsoft Edge

No browser extensions or plugins are required to use AppStream 2.0 in a web browser.

Users can also access AppStream 2.0 fleet streaming sessions on the following browsers and devices:

- Chrome or Safari on an iPad (iOS 11 or later)
- Android (Android 8 or later)
- Microsoft Surface Pro (Windows 10) tablet

AppStream 2.0 is not supported on devices that have screen resolutions smaller than 1024x768 pixels.

Feature and Device Support

AppStream 2.0 provides the following feature and peripheral device support for users who access AppStream 2.0 through a web browser.

Topics

- Dual-Monitor Support
- Touchscreen Device Support
- Drawing Tablet Support
- Relative Mouse Offset

Dual-Monitor Support

AppStream 2.0 supports the use of multiple monitors during streaming sessions, including monitors that have different resolutions. To help ensure an optimal streaming experience, we recommend that users who have monitors with different resolutions set the display scale for their monitors to 100 percent.

Dual monitors are supported for streaming sessions that are started on the following web browsers:

- Google Chrome
- Mozilla Firefox
- Safari

For browser-based streaming sessions on dual monitors, a maximum display resolution of 2560x1600 pixels is supported per monitor. If your users require more than two monitors, or a display resolution that is greater than 2560x1600 pixels per monitor, the AppStream 2.0 client is available.

> **Note**
>
> Dual monitors are not supported on mobile devices or for embedded AppStream 2.0 streaming sessions.

In addition to user connections for streaming sessions, AppStream 2.0 also supports the use of dual monitors for administrative connections to image builders.

Touchscreen Device Support

AppStream 2.0 supports gestures on touch-enabled iPads, Android tablets, and Windows devices. All touch events are passed through to the streaming session and handled according to Windows conventions. Examples of supported touch gestures include long-tap to right-click, swipe to scroll, pinch to zoom, and two-finger rotation for supporting applications.

> **Note**
>
> To enable support for gestures on touch-enabled devices, your AppStream 2.0 image must use a version of the AppStream 2.0 agent released on or after March 7, 2019. For more information, see AppStream 2.0 Agent Release Notes.

For guidance that you can provide your users to help them get started with touch-enabled devices during their AppStream 2.0 streaming sessions, see Touchscreen Devices.

Drawing Tablet Support

Drawing tablets, also known as pen tablets, are computer input devices that let users draw with a stylus (pen). With AppStream 2.0, your users can connect a drawing tablet, such as a Wacom drawing tablet, to their local computer and use the tablet with their streaming applications.

Following are requirements and considerations for enabling your users to use drawing tablets with their streaming applications.

- To enable your users to use this feature, you must configure your AppStream 2.0 fleet to use an image that runs Windows Server 2019.
- To use this feature, users must access AppStream 2.0 through the Google Chrome or Mozilla Firefox browsers only, or the AppStream 2.0 client.
- Streaming applications must support Windows Ink technology. For more information, see Pen interactions and Windows Ink in Windows apps.
- Some applications, such GIMP, must detect drawing tablets on the streaming instance to support pressure sensitivity. If this is the case, your users must use the AppStream 2.0 client to access AppStream 2.0 and stream these applications. In addition, you must qualify your users' drawing

tablets, and users must share their drawing tablets with AppStream 2.0 every time they start a new streaming session. For step-by-step guidance, see Qualify USB Devices for Use with Streaming Applications.

- This feature is not supported on Chromebooks.

To get started with using drawing tablets during application streaming sessions, users connect their drawing tablet to their local computer with USB and use a supported web browser or the AppStream 2.0 client, if it is installed, to start a streaming session. No USB redirection is required to use this feature.

Relative Mouse Offset

By default, during users' streaming sessions, AppStream 2.0 transmits information about mouse movements to the streaming instance by using absolute coordinates and rendering the mouse movements locally. For graphics-intensive applications, such as computer-aided design (CAD)/computer-aided manufacturing (CAM) software or video games, mouse performance improves when relative mouse mode is enabled. Relative mouse mode uses relative coordinates, which represent how far the mouse moved since the last frame, rather than the absolute x-y coordinate values within a window or screen. When relative mouse mode is enabled, AppStream 2.0 renders the mouse movements remotely.

Users can enable this feature during their AppStream 2.0 streaming sessions by doing either of the following:

- Windows: Pressing Ctrl+Shift+F8
- Mac: Pressing Control+Fn+Shift+F8

Configure a Connection Method for Your AppStream 2.0 Users (Web Browser)

Depending on your organizational requirements, you can provide users with access to AppStream 2.0 through a web browser by doing one of the following: Setting up identity federation using SAML 2.0, using an AppStream 2.0 user pool, or creating a streaming URL.

Contents

- SAML 2.0
- AppStream 2.0 User Pool

- Streaming URL
- Next Steps

SAML 2.0

Users enter the URL that you provide for them to access your internal organizational portal. After they enter their organizational credentials, they're redirected to AppStream 2.0.

For more information, see Setting Up SAML.

> **Note**
>
> If your organization requires a smart card for Windows sign in to Active Directory-joined streaming instances and in-session authentication for streaming applications, your users must install and use the AppStream 2.0 client. For more information, see Smart Cards.

AppStream 2.0 User Pool

When you create a new user in the AppStream 2.0 user pool, or assign a user pool user to an AppStream 2.0 stack, AppStream 2.0 sends email to users on your behalf. Users enter the URL that was provided to them in the welcome email, enter their credentials, and then choose **Connect**.

For more information, see AppStream 2.0 User Pools.

Streaming URL

To create a streaming URL, use one of the following methods:

- AppStream 2.0 console
- The CreateStreamingURL API action
- The create-streaming-url AWS CLI command

To create a streaming URL by using the AppStream 2.0 console, complete the steps in the following procedure.

To create a streaming URL by using the AppStream 2.0 console

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.

2. In the navigation pane, choose **Fleets**.
3. In the list of fleets, choose the fleet that is associated with the stack for which you want to create a streaming URL. Verify that the status of the fleet is **Running**.
4. In the navigation pane, choose **Stacks**. Choose the stack, and then choose **Actions**, **Create streaming URL**.
5. In **User id**, enter the user ID.
6. For **URL Expiration**, choose an expiration time, which determines how long the generated URL is valid. This URL is valid for a maximum of seven days.
7. Choose **Get URL**.
8. Copy the URL, save it to an accessible location, and then provide it to your users.

Next Steps

After you configure a web browser connection method, you can provide your users with the following step-by-step guidance to help them connect to AppStream 2.0 and start a streaming session: Connect to AppStream 2.0.

Provide Access Through the AppStream 2.0 Client for Windows

Your users can start AppStream 2.0 streaming sessions by using the AppStream 2.0 client application for a supported device or by using a web browser.

The AppStream 2.0 client is a native application that is designed for users who require the following functionality during their AppStream 2.0 streaming sessions:

- Require support for more than two monitors or 4K resolution.
- Use their USB devices with applications streamed through AppStream 2.0.
- Use their local webcam for video conferencing within their streaming sessions.
- Use keyboard shortcuts during their streaming sessions.
- Require seamless access to local drives and folders during their streaming sessions.
- Require the ability to redirect print jobs from their streaming application to a printer that is connected to their local computer.
- Prefer to interact with remote streaming applications in much the same way as they interact with locally installed applications.

The following topics provide information to help you provide user access through the AppStream 2.0 client. For information about how to provide user access to AppStream 2.0 through a web browser, see Provide Access Through a Web Browser.

Contents

- System Requirements and Feature Support (AppStream 2.0 Client)
- Install and Configure the AppStream 2.0 Client
- AppStream 2.0 Client Release Notes

System Requirements and Feature Support (AppStream 2.0 Client)

This topic provides information to help you understand the requirements for the AppStream 2.0 client and supported features.

System Requirements and Considerations

The AppStream 2.0 client requires the following:

- Follow the principle of least privilege when launching the AppStream 2.0 client. The client should only run with the level of privilege required to complete a task.
- Operating system — Windows 10 (32-bit or 64-bit), Windows 11 (64-bit)
- Microsoft Visual C++ 2019 Redistributable or later. For information about the latest Visual C++ redistributable packages for Visual Studio 2015, 2017, and 2019, see The latest supported Visual C++ downloads in the Microsoft Support documentation.
- RAM — 2 GB minimum
- Hard drive space — 200 MB minimum
- Local administrator rights — Used if you want to install the AppStream 2.0 USB driver for USB driver support.
- An AppStream 2.0 image that uses the latest AppStream 2.0 agent or agent versions published on or after November 14, 2018. For information about AppStream 2.0 agent versions, see AppStream 2.0 Agent Release Notes.
- The Windows native client supports UDP as well as the default TCP-based streaming over NICE DCV. For more information about NICE DCV and UDP, see Enabling the QUIC UDP transport protocol. If you want to enable UDP streaming for the Windows native client, make sure you meet the following requirements. If you don't meet the following requirements, the Windows native client will default back to TCP-based streaming.

- Your Stack has been configured to prefer UDP in the **Streaming Setting Experience** section. For more information, see Create an AppStream 2.0 Fleet and Stack.
- Your network allows UDP traffic on port 8433 for the AWS IP Ranges. For more information, see AWS IP address ranges.
- You are using the latest base image when creating your fleet. For more information, see AppStream 2.0 Base Image and Managed Image Update Release Notes.
- Your end users are using the latest Windows native client. For more information, see Supported clients.

> **Note**
>
> We recommend an internet connection for AppStream 2.0 client installation. In some cases, the client can't be installed on a computer that is not connected to the internet, or USB devices might not work with applications streamed from AppStream 2.0. For more information, see Troubleshooting AppStream 2.0 User Issues.

Feature and Device Support

The AppStream 2.0 client supports the following features and devices.

Topics

- Native Application Mode
- Automatic and On-Demand Diagnostic Log Uploads
- Peripheral Devices

Native Application Mode

> **Note**
>
> Native application mode is not available when streaming from Linux instances.

Native application mode provides a familiar experience for your users during their AppStream 2.0 streaming sessions. When your users connect to AppStream 2.0 in this mode, they can work with their remote streaming applications in much the same way that they work with applications that

are installed on their local computer. Each streaming application in native application mode opens in its own window, and application icons appear on the taskbar on your users' local PC.

If you want your users to connect to AppStream 2.0 in classic mode only, you can configure the `NativeAppModeDisabled` registry value to disable native application mode. For more information, see Choose Whether to Disable Native Application Mode.

For more information about native application mode and classic mode, and for guidance that you can provide to your users, see AppStream 2.0 Client Connection Modes.

> **Note**
>
> Native application mode is not available if your fleet is enabled for the **Desktop** stream view. For information about how to configure the **Desktop** stream view, see Create a Fleet.

Requirements

To enable this feature for your users, you must use an image that uses a version of the AppStream 2.0 agent released on or after February 19, 2020. In addition, version 1.1.129 or later of the AppStream 2.0 client must be installed on your users' PCs. For more information about client versions, see AppStream 2.0 Client Release Notes.

If AppStream 2.0 client version 1.1.129 or later is installed on your users' computer, but you are not using an image that uses an agent version released on or after February 19, 2020, the client falls back to classic mode even if native application mode is selected.

Known Issues

When users try to dock or undock tabs in one browser window into separate windows during a streaming session in native application mode, their remote streaming browser doesn't work the same way as a local browser. To perform this task during a streaming session in native application mode, users must press the Alt key until their browser tabs are docked into separate browser windows.

Automatic and On-Demand Diagnostic Log Uploads

To help with troubleshooting issues that might occur when your users are using the AppStream 2.0 client, you can enable automatic or on-demand diagnostic log uploads, or let your users do so themselves.

> **Note**
>
> Diagnostic logs do not contain sensitive information. You can disable automatic and on-demand diagnostic log uploads on user PCs that you manage, or allow your users to disable these features themselves.

Automatic diagnostic log uploads

When you install the client on PCs that you manage, you can configure the AppStream 2.0 client to upload diagnostic logs automatically. That way, when a client issue occurs, the logs are sent to AppStream 2.0 (AWS) without user interaction. For more information, see Configure Additional AppStream 2.0 Client Settings for Your Users.

Or, you can let your users choose whether to enable automatic diagnostic log uploads when they install the AppStream 2.0 client, or after client installation. For guidance that you can provide your users to help them perform this task, see Setup.

On-demand diagnostic log uploads

If you require more control over logging, you can disable automatic logging and enable on-demand diagnostic log uploads. If you let your users upload diagnostic logs on demand, they can also choose whether to send minidumps (error reports) to AppStream 2.0 (AWS) if an exception occurs or the client stops responding.

For guidance that you can provide your users to help them perform these tasks, see Logging.

Peripheral Devices

The AppStream 2.0 client provides the following support for peripheral devices such as monitors, webcams, mice, keyboards, and drawing tablets.

> **Note**
>
> With certain exceptions, USB redirection is required for the AppStream 2.0 client to support USB devices. And in most cases, when USB redirection is required for a device, you must qualify the device before it can be used with AppStream 2.0 streaming sessions. For more information, see USB Redirection.

Topics

- Multiple Monitors
- Real-Time Audio-Video (Client for Windows)
- USB Devices
- Drawing Tablets
- Keyboard Shortcuts
- Relative Mouse Offset

Multiple Monitors

AppStream 2.0 supports the use of multiple monitors during streaming sessions, including monitors that have different resolutions. To help ensure an optimal streaming experience, we recommend that users who have monitors with different resolutions set the display scale for their monitors to 100 percent.

> **Note**
>
> For AppStream 2.0 streaming sessions that use native application mode, monitors with up to 2K resolution are supported. If higher-resolution monitors are used for streaming sessions, the AppStream 2.0 client falls back to classic mode. In this scenario, the AppStream 2.0 classic mode streaming view occupies 2K of the screen, and the remaining portion of the screen is black.

Multiple Monitors (up to 2K Resolution)

The following AppStream 2.0 instance types support up to 4 monitors and a maximum display resolution of 2560x1600 pixels per monitor: General Purpose, Memory Optimized, Compute Optimized, Graphics Design, and Graphics Pro.

Multiple Monitors (up to 4K Resolution)

The following AppStream 2.0 instance types support up to 2 monitors with a maximum display resolution of 4096x2160 pixels per monitor: Graphics Design and Graphics Pro.

> **Note**
>
> Non-graphics instance types (General Purpose, Memory Optimized, and Compute Optimized) support a maximum display resolution of 2560x1600 pixels per monitor.

Real-Time Audio-Video (Client for Windows)

AppStream 2.0 supports real-time audio-video (AV) by redirecting local webcam video input to AppStream 2.0 streaming sessions. This capability enables your users to use their local webcam for video and audio conferencing within an AppStream 2.0 streaming session. With real-time AV and support for real-time audio, your users can collaborate by using familiar video and audio conferencing applications without having to leave their AppStream 2.0 streaming session.

When a user starts a video conference from within an AppStream 2.0 streaming session, AppStream 2.0 compresses the webcam video and microphone audio input locally before transmitting this data over a secure channel to a streaming instance. During their streaming sessions, users can enable audio and video input by using the AppStream 2.0 toolbar. If users have more than one webcam (for example, if they have a USB webcam that is connected to their local computer and a built-in webcam), they can also choose which webcam to use during their streaming session.

> **Note**
>
> For multi-session fleets, only audio-video out functionalities are accessible. Audio-video in is not yet available for multi-session fleets.

To configure and test support for real-time AV, complete the following steps.

Configure and test support for real-time AV

1. Create a new image builder or connect to an existing image builder that meets the following requirements:
   - The image builder must run Windows Server 2016 or Windows Server 2019.
   - The image builder must use a version of the AppStream 2.0 agent released on or after June 1, 2021.

- For AppStream 2.0 agents released on or after May 17, 2021, real-time AV is enabled by default. To create a streaming URL for testing, you can skip steps 3 through 6 and disconnect from the image builder. If you need to disable real-time AV, complete all of the steps, and disable webcam permissions in step 4.
- The image builder must use a version of the AppStream 2.0 agent released on or after June 24, 2021 to support video when connecting using web browser access. For more information about supported web browsers, see the section called "Web Browser Access".

For information about how to create an image builder, see Launch an Image Builder to Install and Configure Streaming Applications.

2. Connect to the image builder that you want to use and sign in as Administrator. To connect to the image builder, do either of the following:

   - Use the AppStream 2.0 console (for web connections only)
   - Create a streaming URL (for web or AppStream 2.0 client connections)

   > **Note**
   >
   > If the image builder that you want to connect to is joined to an Active Directory domain and your organization requires smart card sign in, you must create a streaming URL and use the AppStream 2.0 client for the connection. For information about smart card sign in, see Smart Cards.

3. On the image builder, open Registry Editor. To do so, on the image builder desktop, in the search box on the taskbar, type **`regedit`**. Then, select the top result for **Registry Editor**.
4. Under **HKEY_LOCAL_MACHINE\SOFTWARE\Amazon\AppStream**, create a new registry value that has the following type, name, and value data:

   - Registry value type: DWORD
   - Registry value name: WebcamPermission
   - Registry value data (Hexademical): 1 to enable or 0 to disable webcam permissions

5. After you create the registry value, switch to **Template User** or to a domain account that does not have administrator permissions on the image builder. To switch to **Template User**, in the toolbar on the top right of the session window, choose **Admin Commands**, **Switch User**, **Template User**.

6. Switch back to **Administrator**.
7. Disconnect from the image builder and create a streaming URL for the image builder. To do so:
   a. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
   b. In the navigation pane, choose **Images**, then choose **Image Builder**.
   c. Select the image builder from which you just disconnected, and choose **Actions**, **Create streaming URL**.
   d. Choose **Copy Link** and save the link to a secure and accessible location. You will use the link in the next step to connect to the image builder.
8. Using the streaming URL that you just created, connect to the image builder by using the AppStream 2.0 client or web browser access.
9. Test the real-time AV experience on the image builder by following the steps in Video and Audio Conferencing (Client for Windows).
10. After you verify that real-time AV is working as expected, disconnect from your streaming session, reconnect to the image builder and follow the necessary steps in Image Assistant to finish creating your image. For information about how to create an image, see Tutorial: Create a Custom AppStream 2.0 Image by Using the AppStream 2.0 Console.

After you finish configuring your image builder and creating an image that supports real-time AV, you can make this feature available to your users on AppStream 2.0 fleets. Ensure that version 1.1.257 or later of the AppStream 2.0 client is installed on your users' computers.

> **Note**
>
> To use real-time AV with the AppStream 2.0 client, your AppStream 2.0 base image and agent version should be June 1, 2021 or later. We recommend using the latest AppStream 2.0 client. For guidance that you can provide to your users to help them use real-time AV, see Video and Audio Conferencing (Client for Windows).
> To use real-time AV with web browser access, your AppStream 2.0 image must use a version of the AppStream 2.0 agent released on or after June 24, 2021. For more information on supported web browsers, see the section called “Web Browser Access”.

USB Devices

The following sections provide information about AppStream 2.0 support for USB devices.

Contents

- USB Redirection
- Smart Cards

USB Redirection

USB redirection is required for most local USB devices to be used during AppStream 2.0 streaming sessions. When USB redirection is required, you must qualify the device before your users can use it during their AppStream 2.0 streaming sessions. After you qualify the device, users must share the device with AppStream 2.0. With USB redirection, during AppStream 2.0 streaming sessions, users' devices are not accessible for use with local applications.

In other cases, USB devices are already enabled for use with AppStream 2.0 and no further configuration is required. For example, smart card redirection is already enabled by default when the AppStream 2.0 client is installed. Because USB redirection isn't used when this feature is enabled, you don't need to qualify smart card readers, and users don't need to share these devices with AppStream 2.0 to use them during streaming sessions.

> **Note**
>
> USB redirection is currently not supported for Linux-based fleet instances.

Smart Cards

AppStream 2.0 supports using a smart card for Windows sign in to Active Directory-joined streaming instances and in-session authentication for streaming applications. Because smart card redirection is enabled by default, users can use smart card readers that are connected to their local computer and their smart cards without USB redirection.

Contents

- Windows Sign In and In-Session Authentication
- Smart Card Redirection

Windows Sign In and In-Session Authentication

AppStream 2.0 supports the use of Active Directory domain passwords or smart cards such as Common Access Card (CAC) and Personal Identity Verification (PIV) smart cards for Windows sign in to AppStream 2.0 streaming instances (fleets and image builders). Your users can use smart card readers connected to their local computer and their smart cards to sign in to an AppStream 2.0 streaming instance that is joined to a Microsoft Active Directory domain. They can also use their local smart card readers and smart cards to sign in to applications within their streaming session.

To ensure that your users can use their smart cards for Windows sign in to Active Directory-joined streaming instances and for in-session authentication for streaming applications, you must:

- Use an image that meets the following requirements:
  - The image must be created from a base image published by AWS on or after December 28, 2020. For more information, see AppStream 2.0 Base Image and Managed Image Update Release Notes.
  - The image must use a version of the AppStream 2.0 agent released on or after January 4, 2021. For more information, see AppStream 2.0 Agent Release Notes.
- Enable **Smart card sign in for Active Directory** on the AppStream 2.0 stack that your users access for streaming sessions, as described in this section.

> **Note**
>
> This setting controls only the authentication method that can be used for Windows sign in to an AppStream 2.0 streaming instance (fleet or image builder). It doesn't control the authentication method that can be used for in-session authentication, after a user signs in to a streaming instance.

- Ensure that your users have AppStream 2.0 client version 1.1.257 or later installed. For more information, see AppStream 2.0 Client Release Notes.

By default, password sign in for Active Directory is enabled on AppStream 2.0 stacks. You can enable smart card sign in for Active Directory by performing the following steps in the AppStream 2.0 console.

To enable smart card sign in for Active Directory by using the AppStream 2.0 console

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.

2. In the left navigation pane, choose **Stacks**.
3. Choose the stack for which you want to enable smart card authentication for Active Directory.
4. Choose the **User Settings** tab, and then expand the **Clipboard, file transfer, print to local device, and authentication permissions** section.
5. For **Smart card sign in for Active Directory**, choose **Enabled**.

   You can also enable **Password sign in for Active Directory**, if it's not already enabled. At least one authentication method must be enabled.
6. Choose **Update**.

Alternatively, you can enable smart card sign in for Active Directory by using the AppStream 2.0 API, an AWS SDK, or the AWS Command Line Interface (AWS CLI).

Smart Card Redirection

When the AppStream 2.0 client is installed, smart card redirection is enabled by default. When this feature is enabled, users can use smart card readers that are connected to their local computer and their smart cards during AppStream 2.0 streaming sessions without USB redirection. During AppStream 2.0 streaming sessions, users' smart card readers and smart cards remain accessible for use with local applications. The AppStream 2.0 client redirects the smart card API calls from users' streaming applications to their local smart card.

> **Note**
>
> Smart card redirection is currently not supported for Linux-based fleet instances or multi-session fleet instances.

> **Note**
>
> If your smart card requires middleware software to operate, the middleware software must be installed on both the user's device, and the AppStream 2.0 streaming instance.

You can disable smart card redirection during client installation on managed devices. For more information, see Choose Whether to Disable Smart Card Redirection. If you disable smart card redirection, your users can't use their smart card reader and smart card during an AppStream 2.0

streaming session without USB redirection. In this case, you must qualify the device. After you qualify the device, users must share the device with AppStream 2.0. When smart card redirection is disabled, during users' AppStream 2.0 streaming sessions, their smart card reader and smart card are not accessible for use with local applications.

Drawing Tablets

Drawing tablets, also known as pen tablets, are computer input devices that let users draw with a stylus (pen). With AppStream 2.0, your users can connect a drawing tablet, such as a Wacom drawing tablet, to their local computer and use the tablet with their streaming applications.

Following are requirements and considerations for enabling your users to use drawing tablets with their streaming applications.

- To enable your users to use this feature, you must configure your AppStream 2.0 fleet to use an image that runs Windows Server 2019.
- To use this feature, users must access AppStream 2.0 by using the AppStream 2.0 client, or through the Google Chrome or Mozilla Firefox browsers only.
- Streaming applications must support Windows Ink technology. For more information, see Pen interactions and Windows Ink in Windows apps.
- Some applications, such GIMP, must detect drawing tablets on the streaming instance to support pressure sensitivity. If this is the case, your users must use the AppStream 2.0 client to access AppStream 2.0 and stream these applications. In addition, you must qualify your users' drawing tablets, and users must share their drawing tablets with AppStream 2.0 every time they start a new streaming session. For more information, see Qualify USB Devices for Use with Streaming Applications.
- This feature is not supported on Chromebooks.

To get started with using drawing tablets during application streaming sessions, users connect their drawing tablet to their local computer with USB, share the device with AppStream 2.0 if required for pressure sensitivity detection, and then use the AppStream 2.0 client or a supported web browser to start an AppStream 2.0 streaming session.

Keyboard Shortcuts

Most operating system keyboard shortcuts are supported. Supported keyboard shortcuts include Alt + Tab, Clipboard shortcuts (Ctrl + X, Ctrl + C, Ctrl+ V), Esc, and Alt + F4

Relative Mouse Offset

By default, during users' streaming sessions, AppStream 2.0 transmits information about mouse movements to the streaming instance by using absolute coordinates and rendering the mouse movements locally. For graphics-intensive applications, such as computer-aided design (CAD)/computer-aided manufacturing (CAM) software or video games, mouse performance improves when relative mouse mode is enabled. Relative mouse mode uses relative coordinates, which represent how far the mouse moved since the last frame, rather than the absolute x-y coordinate values within a window or screen. When relative mouse mode is enabled, AppStream 2.0 renders the mouse movements remotely.

Users can enable this feature during their AppStream 2.0 streaming sessions by doing either of the following:

- Pressing Ctrl+Shift+F8
- Choosing **Relative Mouse Position [Ctrl+Shift+F8]** from the **Settings** menu on the AppStream 2.0 toolbar in the top left area of their streaming session window. This method works when they use classic mode or **Desktop View**.

Install and Configure the AppStream 2.0 Client

You can have your users install the AppStream 2.0 client themselves, or you can install the AppStream 2.0 client for them by running PowerShell scripts remotely.

You must qualify the USB devices that you want to enable your users to use with their streaming session. If their USB device is not qualified, it won't be detected by AppStream 2.0 and can't be shared with the session.

The following topics describe how to install and configure the AppStream 2.0 client.

Contents

- Have Your Users Install the AppStream 2.0 Client Themselves
- Install the AppStream 2.0 Client And Customize the Client Experience for Your Users
- Update the AppStream 2.0 Enterprise Deployment Tool, Client, and USB Driver Manually
- Qualify USB Devices for Use with Streaming Applications
- Configure a Connection Method for Your AppStream 2.0 Client Users
- Enable Users to Share a USB Device with an AppStream 2.0 Streaming Session

- Redirect a Streaming Session from the Web Browser to the AppStream 2.0 Client
- Enable File System Redirection for Your AppStream 2.0 Users
- Enable Local Printer Redirection for Your AppStream 2.0 Users

Have Your Users Install the AppStream 2.0 Client Themselves

For step-by-step guidance that you can provide your users to help them install the AppStream 2.0 client, see Setup.

> **Important**
>
> If your organization has deployed antivirus software that prevents users from running .exe files, you must add an exception to allow your users to run the AppStream 2.0 client installation .exe program. Otherwise, when users try to install the client, either nothing happens, or they receive an error after they start the installation program.

After users install the client, if you plan to let your users use USB devices during their AppStream 2.0 streaming sessions, the following requirements must be met:

- You must qualify the USB devices that can be used with AppStream 2.0. For more information, see Qualify USB Devices for Use with Streaming Applications.
- After their devices are qualified, your users must share the devices with AppStream 2.0 every time they start a new streaming session. For guidance that you can provide your users to them complete this task, see USB Devices.

Install the AppStream 2.0 Client And Customize the Client Experience for Your Users

The following sections describe how to install the AppStream 2.0 client and customize the client experience for your users. If you plan to download and install the client for your users, first download the Enterprise Deployment Tool. You can then run PowerShell scripts to install the AppStream 2.0 client and configure client settings remotely.

Contents

- Download the Enterprise Deployment Tool

- Install the AppStream 2.0 Client and USB Driver
- Accessing AppStream 2.0 with the AppStream 2.0 Client
- Set the StartURL Registry Value for AppStream 2.0 Client Users
- Set the TrustedDomains Registry Value to Enable Other Domains for the AppStream 2.0 Client
- Create the AS2TrustedDomains DNS TXT Record to Enable Your Domain for the AppStream 2.0 Client Without Registry Changes
- Disable DNS TXT Record Lookup for Trusted Domains
- Choose Whether to Disable Automatic Client Updates
- Choose Whether to Disable On-Demand Diagnostic Log Uploads
- Choose Whether to Disable Native Application Mode
- Choose Whether to Disable Local Printer Redirection
- Choose Whether to Disable Smart Card Redirection
- Configure Additional AppStream 2.0 Client Settings for Your Users
- Using Group Policy to Customize AppStream 2.0 Client Experience

Download the Enterprise Deployment Tool

The Enterprise Deployment Tool includes the AppStream 2.0 client installation files and a Group Policy administrative template.

1. To download the Enterprise Deployment Tool, on the bottom right of the AppStream 2.0 supported clients page, select the **Enterprise Deployment Tool** link. This link opens a .zip file that contains the required files for the latest version of the tool.
2. To extract the required files, navigate to the location where you downloaded the tool, right-click the **AmazonAppStreamClient_EnterpriseSetup_<version>** folder, and choose **Extract All**. The folder contains two installation programs and a Group Policy administrative template:

   - AppStream 2.0 client installer (AmazonAppStreamClientSetup_<version>.msi) — Installs the AppStream 2.0 client.
   - AppStream 2.0 USB driver installer (AmazonAppStreamUsbDriverSetup_<version>.exe) — Installs the AppStream 2.0 USB driver that is required to use USB devices with applications streamed through AppStream 2.0.
   - AppStream 2.0 client Group Policy administrative template (as2_client_config.adm) — Lets you configure the AppStream 2.0 client through Group Policy.

Install the AppStream 2.0 Client and USB Driver

After you download the AppStream 2.0 client installation files, run the following PowerShell script on users' computers to install the AppStream 2.0 client installation file, AppStreamClient.exe, and the USB driver silently.

> **Note**
>
> To run this script, you must be logged in to the applicable computer with Administrator permissions. You can also run the script remotely under the System account on startup.

```
Start-Process msiexec.exe -Wait -ArgumentList  '/i
 AmazonAppStreamClientSetup_<version>.msi /quiet'

Start-Process AmazonAppStreamUsbDriverSetup_<version>.exe -Wait -ArgumentList  '/quiet'
```

After you install the Enterprise Deployment Tool on a user's computer, the AppStream 2.0 client is installed as follows:

1. The AppStream 2.0 client installation file is copied to the following path on the user's computer: C:\Program Files (x86)\Amazon AppStream 2.0 Client Installer\AppStreamClient.exe.
2. The first time the user logs on to their computer after the Enterprise Deployment Tool is installed, the AppStream 2.0 client is installed.

   > **Note**
   >
   > If the Enterprise Deployment Tool detects that the AppStream 2.0 Client folder, **AppStreamClient**, already exists in **%localappdata%**, the tool does not install the client.

If a user uninstalls the AppStream 2.0 client, the client isn’t installed again until you update the AppStream 2.0 Enterprise Deployment Tool.

Accessing AppStream 2.0 with the AppStream 2.0 Client

By default, when users launch the AppStream 2.0 client, they can connect only to URLs that include the AppStream 2.0 domain or domains that include a DNS TXT record that enables the connection.

You can let client users access domains other than the AppStream 2.0 domain by doing any of the following:

- Set the `StartURL` registry value to specify a custom URL that users can access, such as the URL for your organization's login portal.
- Set the `TrustedDomains` registry value to specify trusted domains that users can access.
- Create the `AS2TrustedDomains` DNS TXT record to specify trusted domains that users can access. This method lets you avoid registry changes.

> **Note**
>
> The AppStream 2.0 client and DNS TXT record configuration do not prevent users from using other connection methods to access the domains or URLs that you specify. For example, users can access specified domains or URLs by using a web browser, if they have network access to the domains or URLs.

Set the StartURL Registry Value for AppStream 2.0 Client Users

You can use the `StartUrl` registry value to set a custom URL that is populated in the AppStream 2.0 client when a user launches the client. You can create this HKLM registry key while installing the client so that your users don’t need to specify a URL when they launch the client.

After the AppStream 2.0 client is installed, you can run the following PowerShell script to create this registry key, or you can use the administrative template that is included in the AppStream 2.0 client Enterprise Deployment Tool to configure the client through Group Policy.

Replace the `StartUrl` value with a URL for your identity provider (IdP). The URL must use a certificate that is trusted by the device. This means that the certificate that is used by the `StartUrl` webpage must contain a Subject Alternative Name (SAN) that includes the URL's domain name. For example, if your `StartUrl` is set to https://appstream.example.com, the SSL certificate must have a SAN that includes appstream.example.com.

> **Note**
>
> To run this script, you must be logged in to the applicable computer with Administrator permissions. You can also run the script remotely under the System account on startup.

```
$registryPath="HKLM:\Software\Amazon\AppStream Client"
New-Item -Path "HKLM:\Software\Amazon" -Name "AppStream Client" -Force

New-ItemProperty -Path $registryPath -Name "StartUrl" -Value "https://www.example.com"
 -PropertyType String -Force | Out-Null
```

Set the TrustedDomains Registry Value to Enable Other Domains for the AppStream 2.0 Client

You can configure the AppStream 2.0 client to connect to URLs in trusted domains that you specify. For example, you might want to let users connect to any URL in your organizational domain or to any URL in one or more of your IdP domains. When you specify the URL, use the following format: *.*example-idp.com*.

You can specify a list of trusted domains in a comma-separated format. Add this list as a registry value to the AppStream 2.0 `TrustedDomains` HKLM registry key. We recommend that you create this registry key and specify the list of trusted domains when you install the AppStream 2.0 client or, if you are using Microsoft Active Directory, through Group Policy. That way, your users can connect to a URL in any trusted domain immediately after the client is installed.

After the AppStream 2.0 client is installed, you can run the following PowerShell script to create this registry key. Or, you can use the administrative template that is included in the AppStream 2.0 client Enterprise Deployment Tool to configure the client through Group Policy.

Replace the `TrustedDomains` value with a comma-separated list for one or more of your organizational or IdP domains. The certificate used by the trusted domain webpage must contain a SAN that includes the URL's domain. For example, if your trusted domain includes *.example.com, and users specify https://appstream.example.com, the SSL certificate must have a SAN that includes appstream.example.com.

> **Note**
>
> To run this script, you must be logged in to the applicable computer with Administrator permissions. You can also run the script remotely under the System account on startup.

```
$registryPath="HKLM:\Software\Amazon\AppStream Client"
New-Item -Path "HKLM:\Software\Amazon" -Name "AppStream Client" -Force
```

```
New-ItemProperty -Path $registryPath -Name "TrustedDomains" -Value "*.example1.com,
 *.example2.com, aws.amazon.com" -PropertyType String -Force | Out-Null
```

The following are requirements and considerations for formatting trusted domain names.

- The following characters are supported: a-z, 0-9, -, *
- DNS treats the * character either as a wildcard or as an asterisk character (ASCII 42), depending on where it appears in the domain name. Following are restrictions when using * as a wildcard in the name of a DNS record:
  - The * must replace the leftmost label in a domain name. For example, *.example.com or *.prod.example.com. If you include * in any other position, such as prod.*.example.com, DNS treats it as an asterisk character (ASCII 42), not as a wildcard.
  - The * must replace the entire label. For example, you can't specify *prod.example.com or prod*.example.com.
  - The * applies to the subdomain level that includes the *, and to all the subdomains of that subdomain. For example, if an entry is named *.example.com, the AppStream 2.0 client allows zenith.example.com, acme.zenith.example.com, and pinnacle.acme.zenith.example.com.

Create the AS2TrustedDomains DNS TXT Record to Enable Your Domain for the AppStream 2.0 Client Without Registry Changes

You can enable users to connect to any URL in your organizational domain (for example, *.example.com) or to any URL in your IdP domains (for example, *.example-idp.com) by creating a DNS TXT record in that domain. When you create the DNS TXT record, the `StartURL` or `TrustedDomains` registry values are not required to allow a user to connect to a URL.

You can specify a list of trusted subdomains in a comma-separated format, prefixed with `AS2TrustedDomains=`. Then, create a DNS TXT record for the appropriate domain. The `AS2TrustedDomains` DNS TXT record can only enable the same domain, or subdomains, of the domain in which the DNS TXT record is created. You cannot use the DNS TXT record to enable other domains.

For more information about setting up the DNS record, see Enable your organizational domain for the AppStream 2.0 client with a Route 53 DNS TXT record and Creating an AS2TrustedDomains DNS TXT record to redirect the AppStream 2.0 native client to a third-party identity provider.

> **Note**
>
> When you create DNS TXT records, any users can stream from enabled domains that are not included in the `StartURL` or `TrustedDomains` registry values. The AppStream 2.0 client and DNS TXT record configuration do not prevent users from using other connection methods to access the domains or URLs that you specify. For example, users can access specified domains or URLs by using a web browser, if they have network access to the domains or URLs.

DNS TXT Record Configuration Example

The following is an example of a DNS TXT record configuration. With the configuration for this example, users can launch the AppStream 2.0 client and connect to appstream.example.com or appstream-dev.example.com. However, they cannot connect to example.com.

- `Domains to enable` — appstream.example.com, appstream-dev.example.com
- `DNS TXT record location` — example.com
- `DNS TXT record value` — AS2TrustedDomains=appstream.example.com,appstream-dev.example.com

Requirements and Considerations

Keep in mind the following requirements and considerations for creating a DNS TXT record:

- You must create the TXT record at the second-level domain. For example, if your domain is prod.appstream.example.com, you must create the DNS TXT record at example.com.
- The TXT record value must start with `AS2TrustedDomains=`
- The following characters are supported: a-z, 0-9, -, *
- DNS treats the * character either as a wildcard or as an asterisk character (ASCII 42), depending on where it appears in the domain name. Following are restrictions when using * as a wildcard in the name of a DNS record:
  - The * must replace the leftmost label in a domain name. For example, *.example.com or *.prod.example.com. If you include * in any other position, such as prod.*.example.com, DNS treats it as an asterisk character (ASCII 42), not as a wildcard.

- The * must replace the entire label. For example, you can't specify *prod.example.com or prod*.example.com.
- The * applies to the subdomain level that includes the *, and to all the subdomains of that subdomain. For example, if an entry is named *.example.com, the AppStream 2.0 client allows connections to the following domains: zenith.example.com, acme.zenith.example.com, and pinnacle.acme.zenith.example.com.

Disable DNS TXT Record Lookup for Trusted Domains

By default, when users launch the AppStream 2.0 and specify a URL that is not an AppStream 2.0 domain, the client performs a DNS TXT record lookup. The lookup is performed on the second-level domain of the URL so that the client can determine whether the domain is included in the `AS2TrustedDomains` list. This behavior lets users connect to domains that are not specified in the `StartURL` or `TrustedDomains` registry keys, or AppStream 2.0 domains.

You can disable this behavior by setting the value for the `DnsTxtRecordQueryDisabled` registry key to `true`. You can create this registry key when you install the AppStream 2.0 client. That way, the client connects only to URLs that are specified by the `StartURL` or `TrustedDomains` registry keys.

After the AppStream 2.0 client is installed, you can run the following PowerShell script to create this registry key. Or, you can use the administrative template that is included in the AppStream 2.0 client Enterprise Deployment Tool to configure the client through Group Policy.

> **Note**
>
> To run this script, you must be logged in to the applicable computer with Administrator permissions. You can also run the script remotely under the System account on startup.

```
$registryPath="HKLM:\Software\Amazon\AppStream Client"
New-Item -Path "HKLM:\Software\Amazon" -Name "AppStream Client" -Force

New-ItemProperty -Path $registryPath -Name "DnsTxtRecordQueryDisabled" -Value "true" -
PropertyType String -Force | Out-Null
```

Choose Whether to Disable Automatic Client Updates

By default, when a new version of the AppStream 2.0 client is available, the client updates automatically to the latest version. You can disable automatic updates by setting the value for the `AutoUpdateDisabled` registry key to `true`. You can create this registry key when you install the AppStream 2.0 client. That way, the client is not updated automatically whenever a new version is available.

After the AppStream 2.0 client is installed, you can run the following PowerShell script to create this registry key. Or, you can use the administrative template that is included in the AppStream 2.0 client Enterprise Deployment Tool to configure the client through Group Policy.

> **Note**
>
> To run this script, you must be logged in to the applicable computer with Administrator permissions. You can also run the script remotely under the System account on startup.

```
$registryPath="HKLM:\Software\Amazon\AppStream Client"
New-Item -Path "HKLM:\Software\Amazon" -Name "AppStream Client" -Force

New-ItemProperty -Path $registryPath -Name "AutoUpdateDisabled" -Value "True" -
PropertyType String -Force | Out-Null
```

Choose Whether to Disable On-Demand Diagnostic Log Uploads

By default, the AppStream 2.0 client allows users to upload diagnostic logs and minidumps on demand to AppStream 2.0 (AWS). In addition, if an exception occurs or the AppStream 2.0 client stops responding, users are prompted to choose whether they want to upload the minidump and associated logs. For more information about on-demand diagnostic logging, see Automatic and On-Demand Diagnostic Log Uploads.

You can disable these behaviors by setting the value for the `UserUploadOfClientLogsAllowed` registry key to `false`. You can create this HKLM registry key when you install the AppStream 2.0 client.

After the AppStream 2.0 client is installed, you can run the following PowerShell script to create this registry key. Or, you can use the administrative template that is included in the AppStream 2.0 client Enterprise Deployment Tool to configure the client through Group Policy.

> **Note**
>
> To run this script, you must be logged in to the applicable computer with Administrator permissions. You can also run the script remotely under the System account on startup.

```
$registryPath="HKLM:\Software\Amazon\AppStream Client"
New-Item -Path "HKLM:\Software\Amazon" -Name "AppStream Client" -Force

New-ItemProperty -Path $registryPath -Name "UserUploadOfClientLogsAllowed" -Value
 "false" -PropertyType String -Force | Out-Null
```

Choose Whether to Disable Native Application Mode

By default, the AppStream 2.0 client can run in either classic mode or native application mode. You can disable native application mode by setting the value for the `NativeAppModeDisabled` registry key to `true`. You can create this HKLM registry key when you install the AppStream 2.0 client. If the value is set to `true`, the client runs in classic mode only. For information about native application mode, see Native Application Mode.

After the AppStream 2.0 client is installed, you can run the following PowerShell script to create this registry key. Or, you can use the administrative template that is included in the AppStream 2.0 client Enterprise Deployment Tool to configure the client through Group Policy.

> **Note**
>
> To run this script, you must be logged in to the applicable computer with Administrator permissions. You can also run the script remotely under the System account on startup.

```
$registryPath="HKLM:\Software\Amazon\AppStream Client"
New-Item -Path "HKLM:\Software\Amazon" -Name "AppStream Client" -Force

New-ItemProperty -Path $registryPath -Name "NativeAppModeDisabled" -Value "True" -
PropertyType String -Force | Out-Null
```

Choose Whether to Disable Local Printer Redirection

By default, the AppStream 2.0 client enables users to redirect print jobs from their streaming applications to a printer that is connected to their local computer. You can disable local printer redirection by setting the value for the `PrinterRedirectionDisabled` registry key to `true`. You can create this HKLM registry key when you install the AppStream 2.0 client. If the value is set to `true`, the client does not redirect print jobs from users' streaming applications to a printer that is connected to their local computer.

After you install the AppStream 2.0 client, you can run the following PowerShell script to create this registry key. Or, you can use the administrative template that is included in the AppStream 2.0 client Enterprise Deployment Tool to configure the client through Group Policy.

> **Note**
>
> To run this script, you must be logged in to the applicable computer with Administrator permissions. You can also run the script remotely under the System account on startup.

```
$registryPath="HKLM:\Software\Amazon\AppStream Client"
New-Item -Path "HKLM:\Software\Amazon" -Name "AppStream Client" -Force

New-ItemProperty -Path $registryPath -Name "PrinterRedirectionDisabled" -Value "True" -
PropertyType String -Force | Out-Null
```

Choose Whether to Disable Smart Card Redirection

By default, smart card redirection is enabled for the AppStream 2.0 client. When this feature is enabled, users can use smart card readers that are connected to their local computers and their smart cards during AppStream 2.0 streaming sessions without USB redirection. During AppStream 2.0 streaming sessions, users' smart card readers and smart cards remain accessible for use with local applications. The client redirects the smart card API calls from users' streaming applications to their local smart card. You can disable smart card redirection by setting the value for the `SmartCardRedirectionDisabled` registry key to `true`. You can create this HKLM registry key when you install the AppStream 2.0 client.

If the value is set to `true`, your users can't use their smart card readers and smart cards during an AppStream 2.0 streaming session without USB redirection. In this case, users can't sign in to their streaming applications by using a smart card that is connected to their local computer unless you

qualify the device. After you qualify the device, users must share the device with AppStream 2.0. When smart card redirection is disabled, during users' AppStream 2.0 streaming sessions, their smart card readers and smart cards are not accessible for use with local applications.

After you install the AppStream 2.0 client, you can run the following PowerShell script to create this registry key. Or, you can use the administrative template that is included in the AppStream 2.0 client Enterprise Deployment Tool to configure the client through Group Policy.

> **Note**
>
> To run this script, you must be logged in to the applicable computer with Administrator permissions. You can also run the script remotely under the System account on startup.

```
$registryPath="HKLM:\Software\Amazon\AppStream Client"
New-Item -Path "HKLM:\Software\Amazon" -Name "AppStream Client" -Force

New-ItemProperty -Path $registryPath -Name "SmartCardRedirectionDisabled" -Value "True"
 -PropertyType String -Force | Out-Null
```

Configure Additional AppStream 2.0 Client Settings for Your Users

The AppStream 2.0 client uses registry keys to configure the following additional client settings:

- AppStream 2.0 client End-User License Agreement (EULA) acceptance
- AppStream 2.0 client EULA version accepted
- Automatic diagnostic log uploads for the AppStream 2.0 client
- Automatic updates for the USB driver that is used to pass USB drivers to AppStream 2.0
- Enabling hardware rendering in the AppStream 2.0 client
- Setting custom folder paths for file system redirection in the AppStream 2.0 client

The following table summarizes the registry values for additional client settings that you can use to customize the AppStream 2.0 client experience for your users.

> **Note**
>
> These values are case sensitive.

Value	Registry path	Type	Description	Data
`EULAAccepted`	`HKCU\Software\Amazon\Appstream Client`	String	Set this value to `true` to accept the AppStream 2.0 client EULA on behalf of your users.	**`true`/`false`**
`AcceptedEULAVersion`	`HKCU\Software\Amazon\Appstream Client`	String	The version of EULA that is accepted. If the current version of the AppStream 2.0 client EULA is different from the version of the EULA that is accepted, users are prompted to accept the current version of the EULA.	**`1.0`**
`DiagnosticInfoCollectionAllowed`	`HKCU\Software\Amazon\Appstream Client`	String	Set this value to `true` to enable AppStream 2.0 to automatically send diagnostic logs from the AppStream 2.0 client to	**`true`/`false`**

Value	Registry path	Type	Description	Data
			AppStream 2.0 (AWS).	
`USBDriverOptIn`	`HKCU\Software\Amazon\Appstream Client`	String	Set this value to `true` to enable AppStream 2.0 to automatically update the USB driver that is used to pass USB drivers to AppStream 2.0.	**`true`/`false`**
`HardwareRenderingEnabled`	`HKCU\Software\Amazon\Appstream Client`	String	Set this value to `true` to enable hardware rendering in the AppStream 2.0 client.	**`true`/`false`**

Value	Registry path	Type	Description	Data
`FileRedirectionCustomDefaultFolders`	`HKCU\Software\Amazon\Appstream Client`	String	Set this value to include at least one folder path for file system redirection. Separate multiple folder paths by using '|'. By default, the following folder paths are specified: %USERPROFILE%\Desktop|%USERPROFILE%\Documents|%USERPROFILE%\Downloads	*Valid folder path*

After the AppStream 2.0 client is installed, you can run the following PowerShell script to create these registry keys. If you don't want to create all of the registry keys, modify the script as needed to create only the registry keys that you want. Or, you can use the administrative template that is provided in the AppStream 2.0 client Enterprise Deployment Tool to configure the client through Group Policy.

> **Note**
>
> You must set the following entries for each user.

```
$registryPath="HKCU:\Software\Amazon\AppStream Client"
```

```
New-Item -Path "HKCU:\Software\Amazon" -Name "AppStream Client" -Force
New-ItemProperty -Path $registryPath -Name "EULAAccepted" -Value "true" -PropertyType
 String -Force | Out-Null
New-ItemProperty -Path $registryPath -Name "AcceptedEULAVersion" -Value "1.0" -
PropertyType String -Force | Out-Null
New-ItemProperty -Path $registryPath -Name "DiagnosticInfoCollectionAllowed" -Value
 "true" -PropertyType String -Force | Out-Null
New-ItemProperty -Path $registryPath -Name "USBDriverOptIn" -Value "true" -PropertyType
 String -Force | Out-Null
New-ItemProperty -Path $registryPath -Name "HardwareRenderingEnabled" -Value "true" -
PropertyType String -Force | Out-Null
New-ItemProperty -Path $registryPath -Name "FileRedirectionCustomDefaultFolders" -Value
 "%USERPROFILE%\Desktop|%USERPROFILE%\Documents|%USERPROFILE%\Downloads" -PropertyType
 String -Force | Out-Null
```

Using Group Policy to Customize AppStream 2.0 Client Experience

You can use the administrative template that is provided in the AppStream 2.0 client Enterprise Deployment Tool to configure the client through Group Policy. To learn how to load administrative templates into the Group Policy Management Console, see Recommendations for managing Group Policy administrative template (.adm) files in the Microsoft Support documentation.

Update the AppStream 2.0 Enterprise Deployment Tool, Client, and USB Driver Manually

By default, the AppStream 2.0 client and USB driver are updated automatically when a new client version is released. However, if you used the Enterprise Deployment Tool to install the AppStream 2.0 client for your users and you disabled automatic updates, you must update the AppStream 2.0 Enterprise Deployment Tool, client, and USB driver manually. To do so, perform the following steps to run the required PowerShell commands on users' computers.

> **Note**
>
> To run these commands, you must either be logged in to the applicable computer as Administrator, or you can run the script remotely under the SYSTEM account on startup.

1. Uninstall the AppStream 2.0 Enterprise Deployment Tool silently:

```
Start-Process msiexec.exe -Wait -ArgumentList '/x
 AmazonAppStreamClientSetup_<existing_version>.msi /quiet'
```

2. Uninstall the AppStream 2.0 USB driver silently:

```
Start-Process -Wait AmazonAppStreamUsbDriverSetup_<existing_version>.exe -
ArgumentList '/uninstall /quiet /norestart'
```

3. Uninstall the AppStream 2.0 client silently:

```
Start-Process "$env:LocalAppData\AppStreamClient\Update.exe" -ArgumentList '--
uninstall'
```

> **Note**
>
> This process also removes the registry keys that are used to configure the AppStream 2.0 client. After you reinstall the AppStream 2.0 client, you must recreate these keys.

4. Clean the application installation directory:

```
Remove-Item -Path $env:LocalAppData\AppStreamClient -Recurse -Confirm:$false –
Force
```

5. Restart the computer:

```
Restart-computer
```

6. Install the latest version of the AppStream 2.0 Enterprise Deployment Tool silently:

```
Start-Process msiexec.exe -Wait -ArgumentList '/i
 AmazonAppStreamClientSetup_<new_version>.msi /quiet'
```

7. Install the latest version of the AppStream 2.0 USB driver silently:

```
Start-Process AmazonAppStreamUsbDriverSetup_<new_version>.exe -Wait -ArgumentList
 '/quiet'
```

Qualify USB Devices for Use with Streaming Applications

There are two methods for specifying which USB devices your users can redirect into their AppStream 2.0 streaming instances:

> **Note**
>
> USB redirection is currently only supported on Windows AppStream 2.0 streaming instances.

- You can create the USB device filter strings within the configuration file saved on an image. This method can only be used with Always-On and On-Demand fleets.
- You can specify USB device filter strings when you create the fleet, either with the AWS Management Console or with the `CreateFleet` API. For detailed information about these strings, see the section below. This method can only be used with Elastic fleets.

You can create a file on your AppStream 2.0 image that specifies which USB devices a user can make available for their streaming applications. To qualify your users' USB devices so that the devices can be used with streaming applications, perform these steps.

> **Note**
>
> For security reasons, only qualify USB devices from approved trusted sources. Qualifying all generic devices or classes of devices might allow unapproved devices to be used with your streaming applications.

1. If you haven't already done so, install the AppStream 2.0 client. For information, see Install and Configure the AppStream 2.0 Client.
2. Connect the USB device that you want to qualify to your computer.
3. Navigate to **C:\Users\<logged-in-user>\AppData\Local\AppStreamClient**, and double-click **dcvusblist.exe**.
4. In the **DCV - USB devices** dialog box, the list of USB devices that are connected to your local computer displays. The **Filter** column displays the filter string for every USB device. Right-click the list entry for a USB device that you want to enable, and choose **Copy filter string**.

5. On your desktop, choose the Windows **Start** button, and search for Notepad. Double-click **Notepad** to open a new file, copy the filter string to the file, and save it. Later, you'll use the filter string to qualify the USB device.
6. Launch a new image builder. For more information, see Launch an Image Builder to Install and Configure Streaming Applications.
7. After your image builder is in the **Running** state, perform the following steps to create a streaming URL and connect to the image builder by using the AppStream 2.0 client.
   a. With your image builder selected in the list, choose **Actions**, **Create streaming URL**.
   b. In the **Create streaming URL** dialog box, choose **Copy link**, and copy and paste the web address into a separate file for later use. You'll use this URL to reconnect to the image builder in step 12.
   c. Choose **Launch in Client**.
   d. If the **Launch Application** dialog box opens and prompts you to choose the application to use when opening the link, choose **Amazon AppStream**, **Open link**. To prevent this dialog box from displaying the next time you perform this step to connect to an image builder, select the **Remember my choice for amazonappstream links** check box.
   e. If the AppStream 2.0 client displays links to the AWS Customer Agreement, AWS Service Terms, and the AWS Privacy Notice, and third-party notices, review this information, and then choose **Finish**.
   f. If the client sign-in page is displayed, the web address field is prepopulated with the streaming URL. Choose **Connect**.
   g. If prompted, log in to the image builder as Administrator.
8. After you are connected to the image builder, if your USB device requires you to install drivers before you use it, download and install the drivers on the image builder. For example, if you use the Connexion 3D mouse, you must download and install the required Connexion drivers on the image builder.
9. On your image builder desktop, choose the Windows **Start** button, and search for Notepad. Right-click **Notepad**, and choose **Run as Administrator**.
10. Choose **File**, **Open**, and open the following file: `C:\ProgramData\Amazon\Photon\DCV\usb_device_allowlist.txt`. You can also allow an entire category of devices or all devices from a specific manufacturer by using wildcard expressions in the `usb_device_allowlist.txt` file.

11. Copy the filter string from your local computer to the image builder. The filter string for a specific USB device is a comma-separated string of the following fields: **Name**, **Base Class**, **SubClass**, **Protocol**, **ID Vendor**, **ID Product**, **Support Autoshare**, and **Skip Reset**. For detailed information about these strings, see Working with USB Device Filter Strings.
12. Disconnect from your image builder, restart it, and reconnect to it by using the AppStream 2.0 client. To do so, open the AppStream 2.0 client and paste the streaming URL that you created in step 7 into the client sign-in web address field, and choose **Connect**.
13. On the image builder, test your USB device to confirm that it works as expected.
14. Before your users can use the USB device in an AppStream 2.0 session, they must first share the device with their session. For guidance that you can provide your users to help them perform this task, see USB Devices.
15. If the USB device works with the image builder as expected, create an image. For more information, see Tutorial: Create a Custom AppStream 2.0 Image by Using the AppStream 2.0 Console.
16. After you finish creating the image, update your AppStream 2.0 fleet to use the new image.

Working with USB Device Filter Strings

This section describes the filter strings that are available for qualifying USB devices for AppStream 2.0 streaming sessions. It also provides guidance for working with these strings. The following filter strings are available:

- `Name` — By default, the value for this filter string is the name of the device, but you can specify your own value.
- `Base Class,SubClass,Protocol` — The USB class code for the device. For more information, see Defined Class Codes.
- `ID Vendor (VID)` — A unique identifier that is assigned by the USB organization to the manufacturer of the USB device.
- `ID Product (PID)` — A unique identifier that assigned by the manufacturer to the USB device.
- `Support Autoshare` — Lets the AppStream 2.0 client automatically share the device when a streaming session starts. Set this value to `1` to allow automatic device sharing. Set this value to `0` to not allow automatic device sharing.
- `Skip Reset` — By default, when a USB device is shared by AppStream 2.0 with a streaming session, the device is reset to ensure that it functions correctly. However, some USB devices don't function correctly during the streaming session if they are reset. To prevent this problem from

occurring, set the value for this filter string to 1 to instruct the AppStream 2.0 client not to reset the device while it is shared with a streaming session. To ensure that the device is reset while it is shared with a streaming session, set this value to `0`. When you set a value for `Skip Reset`, make sure that you set the value for `Support Autoshare` to `0` or 1.

The filter string that is copied from the local computer is specific to a USB device. In some cases, you might want to allow an entire class of devices instead of allowing every possible USB device. For example, you might want to allow your users to use any kind of Wacom design tablets or use any USB mass storage device. In such scenarios, you can provide wildcard characters for specific filter string fields. If you don't know the VID and PID for your USB devices, you can search for this information in the USB ID database.

The following examples show how to configure filter strings for USB device sharing during streaming sessions:

- Allow all mass storage devices automatically on starting a streaming session — "Mass storage, 8, *, *, *, *,1,0"
- Allow all Wacom devices automatically on starting a streaming session — "Wacom tablets, 3, *, *, 1386, *,1,0"
- Allow all devices that provide an audio interface — "Audio, 1, *, *, *, *,1,0"
- Allow device X, but don't reset it while the device is shared. Don't share the device automatically on starting a streaming session — "X, Y, *, *, 1386, *,0,1"

Configure a Connection Method for Your AppStream 2.0 Client Users

After you install the AppStream 2.0 client on your users' local computers, they can use the AppStream 2.0 client to connect to a streaming session. Depending on your organizational requirements, you can provide client users with access to AppStream 2.0 by doing one of the following: Setting up identity federation using SAML 2.0, using an AppStream 2.0 user pool, or creating a streaming URL.

Contents

- SAML 2.0
- AppStream 2.0 User Pool
- Streaming URL
- Next Steps

SAML 2.0

If you use external identity providers to federate your users to an AppStream 2.0 stack, you must create a registry value to configure the AppStream 2.0 client with a prepopulated URL whenever the client is launched. The URL must use a certificate that is trusted by the device. The certificate must contain a Subject Alternative Name (SAN) that includes the URL's domain name.

For more information, see:

- Setting Up SAML
- Set the StartURL Registry Value for AppStream 2.0 Client Users

AppStream 2.0 User Pool

When you create a new user in the AppStream 2.0 user pool, or assign a user pool user to an AppStream 2.0 stack, AppStream 2.0 sends email to users on your behalf. Users enter the URL that was provided to them in the welcome email, enter their credentials, and then choose **Connect**.

For more information, see AppStream 2.0 User Pools.

Streaming URL

To create a streaming URL, use one of the following methods:

- AppStream 2.0 console
- The CreateStreamingURL API action
- The create-streaming-url AWS CLI command

To create a streaming URL by using the AppStream 2.0 console, complete the steps in the following procedure.

To create a streaming URL by using the AppStream 2.0 console

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. In the navigation pane, choose **Fleets**.
3. In the list of fleets, choose the fleet that is associated with the stack for which you want to create a streaming URL. Verify that the status of the fleet is **Running**.
4. In the navigation pane, choose **Stacks**. Choose the stack, and then choose **Actions**, **Create streaming URL**.

5. In **User id**, enter the user ID.
6. For **URL Expiration**, choose an expiration time, which determines how long the generated URL is valid. This URL is valid for a maximum of seven days.
7. Choose **Get URL**.
8. Copy the URL, save it to an accessible location, and then provide it to your users.

   In the AppStream 2.0 client sign-in page, users enter the streaming URL that you created as the web address, and then choose **Connect**.

Next Steps

After you configure a client connection method, you can provide your users with the following step-by-step guidance to help them connect to AppStream 2.0 and start a streaming session: Connect to AppStream 2.0.

Enable Users to Share a USB Device with an AppStream 2.0 Streaming Session

Before users share their USB devices with an AppStream 2.0 session, the USB devices must be qualified. Otherwise, when users start a streaming session, their USB device is not detected by AppStream 2.0 and cannot be shared with the session. For more information, see Qualify USB Devices for Use with Streaming Applications.

Redirect a Streaming Session from the Web Browser to the AppStream 2.0 Client

You can configure AppStream 2.0 to redirect a streaming session from a web browser to the AppStream 2.0 client. That way, when your users sign in to AppStream 2.0 and start a streaming session in their web browser, their session is redirected to the AppStream 2.0 client. To do so, perform these steps.

1. Use the AppStream 2.0 `CreateStreamingURL` API action to generate a streaming URL.
2. Add the following prefix for the custom AppStream 2.0 client handler to the streaming URL: **`amazonappstream:`**

   Together, the prefix and streaming URL are formatted as follows:

   `amazonappstream:`*`base64encoded(streamingURL)`*
3. When users are redirected to the streaming URL, their browser detects that the link must be opened by the AppStream 2.0 client.

4. Users are prompted to choose whether they want to start the streaming session by using the AppStream 2.0 client.
5. After the prompt, either of the following occurs:
   - If the AppStream 2.0 client is installed, the user can choose to continue the streaming session by using the AppStream 2.0 client.
   - If the AppStream 2.0 client is not installed, the browser behavior varies as follows:
     - Chrome — No message is displayed.
     - Firefox — A message states that the user needs a new app to open Amazon AppStream.
     - Microsoft Edge — No message is displayed.
     - Internet Explorer — A message notifies the user that the AppStream 2.0 client is not installed.

       In this case, users can select the **Download AppStream Client** link to download the client. After they download the client, they can install it, and refresh their browser to start the streaming session by using the client.

Enable File System Redirection for Your AppStream 2.0 Users

AppStream 2.0 file system redirection lets users who have the AppStream 2.0 client installed access files on their local computer from within their streaming session. When you enable file system redirection, you can specify the list of local drives and folders that your users can choose to access. When users sign in to AppStream 2.0 and start a streaming session, they can choose the drive or folder that they want to access from the list. Then they can share the drive or folder with AppStream 2.0. The drive or folder remains available for them to access during their streaming sessions. Users can stop sharing their local drives or folders at any time.

> **Note**
>
> File system redirection is currently not supported for Linux-based fleet instances or multi-session fleet instances.

Prerequisites for File System Redirection

To enable AppStream 2.0 file redirection:

- You must use an image that uses a version of the AppStream 2.0 agent released on or after August 8, 2019. For more information, see AppStream 2.0 Agent Release Notes.
- Your users must have AppStream 2.0 client version 1.0.480 or later installed. For more information, see AppStream 2.0 Client Release Notes.
- File upload and download must be enabled on the stack that your users access for streaming sessions. See the following procedure.

How to Enable File System Redirection

Perform the following steps to enable both file upload and download on the stack that your users access for streaming sessions.

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. In the left navigation pane, choose **Stacks**.
3. Choose the stack for which you want to enable file system redirection.
4. Choose the **User Settings** tab, and then expand the **Clipboard, file transfer, and local print permissions** section.
5. For **File transfer**, verify that **Upload and download** is selected. If not, choose **Edit**, and then choose **Upload and download**.
6. Choose **Update**.

Make Default Drives and Folders Available for Your Users to Share

By default, when you enable file direction for users of a stack, the following drives and folders are made available for those users to share in their streaming session:

- Drives:
  - All local hard disks (physical drives, such as the C Drive and D Drive)
  - All virtual drives (network and virtual drives such as mapped drive letters, Google Drive, and OneDrive)
  - All local USB drives
- Folders:
  - %USERPROFILE%\Desktop
  - %USERPROFILE%\Documents

- %USERPROFILE%\Downloads

These drive and folder paths prepopulate the **Share your local drives and folders** dialog box. This dialog box is displayed when users sign in to AppStream 2.0, start a streaming session, and choose **Settings**, **Local Resources**, and **Local Drives and Folders**.

You can change or define your own default drive and folder paths by editing the registry. You can also use the administrative template file that is provided in the AppStream 2.0 client Enterprise Deployment Tool. This template lets you configure the client by using Group Policy. For more information, see Install and Configure the AppStream 2.0 Client.

When users access their shared local drives and folders during a streaming session, the corresponding paths appear with backslashes replaced by underscores. They are also suffixed with the name of the local computer and a drive letter. For example, for a user with the user name janedoe and a computer name of ExampleCorp-123456, the default Desktop, Documents, and Downloads folder paths appear as follows:

C_Users_janedoe_Desktop (\\ExampleCorp-123456) (F:)

C_Users_janedoe_Documents (\\ExampleCorp-123456) (G:)

C_Users_janedoe_Downloads (\\ExampleCorp-123456) (H:)

Provide Your AppStream 2.0 Users with Guidance for Working with File System Redirection

To help your users understand how to work with file redirection during their streaming sessions, you can provide them with the information in Local File Access.

Enable Local Printer Redirection for Your AppStream 2.0 Users

With local printer redirection, your AppStream 2.0 users can redirect print jobs from their streaming application to a printer that is connected to their local computer, including any network printers that the users have mapped. You don't need a printer driver installed on the AppStream 2.0 streaming instance to enable users to print documents during their streaming sessions.

> **Note**
>
> Enabling local printer redirection is currently not supported for Linux-based stacks.

Prerequisites for Local Printer Redirection

To ensure that your users can use local printer redirection, you must:

- Use an image that uses a version of the AppStream 2.0 agent released on or after July 30, 2020. For more information, see AppStream 2.0 Agent Release Notes.
- Ensure that your users have AppStream 2.0 client version 1.1.179 or later installed. For more information, see AppStream 2.0 Client Release Notes.
- Ensure printer redirection is enabled on the stack that your users access for streaming sessions.

How to Enable or Disable Local Printer Redirection

By default, local printer redirection is enabled when the AppStream 2.0 client is installed. However, if local printer redirection is not enabled on the stack that your users access for streaming sessions, you can enable it in the AppStream 2.0 console by performing the following steps.

To enable local printer redirection by using the AppStream 2.0 console

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. In the left navigation pane, choose **Stacks**.
3. Choose the stack for which you want to enable local printer redirection.
4. Choose the **User Settings** tab, and then expand the **Clipboard, file transfer, print to local device, and authentication permissions** section.
5. For **Print to local device**, verify that **Enabled** is selected. If not, choose **Edit**, and then choose **Enabled**.
6. Choose **Update**.

Alternatively, you can enable local printer redirection by using the AppStream 2.0 API, an AWS SDK, or the AWS Command Line Interface (AWS CLI).

To disable local printer redirection

You can disable local printer redirection in any of the following ways:

- During client installation on managed devices. For more information, see Choose Whether to Disable Local Printer Redirection.
- By using the AppStream 2.0 console to disable this option on an AppStream 2.0 stack.

- By using the AppStream 2.0 API, an AWS SDK, or the AWS Command Line Interface (AWS CLI) to disable this option on an AppStream 2.0 stack.

AppStream 2.0 Client Release Notes

The AppStream 2.0 client is a native application that is designed for users who require additional functionality during their AppStream 2.0 streaming sessions. The following table describes the latest updates that are available in released versions of the AppStream 2.0 client.

For more information about the client, see Provide Access Through the AppStream 2.0 Client for Windows.

Client version	Release date	Changes
1.1.1303	04-03-2024	• Includes bug fixes and improvements
1.1.1300	03-28-2024	• Added support for launching the AppStream 2.0 client from IdP-initiated streaming sessions • Added support for new relay state regional endpoints • Upgrades the embedded Chromium browser to version 121.3.70 • Includes bug fixes and improvements
1.1.1259	02-08-2024	• Includes bug fixes and improvements
1.1.1246	01-18-2024	• Includes improved accessibility features

Client version	Release date	Changes
		• Includes bug fixes and improvements • Upgrades the embedded Chromium browser to version 119.4.30
1.1.1228	11-01-2023	• Includes bug fixes and improvements • Upgrades the embedded Chromium browser to version 114.1.120
1.1.1183	06-22-2023	• Includes bug fixes and improvements • Upgrades the embedded Chromium browser to version 111.2.20
1.1.1159	05-09-2023	• Includes bug fixes and improvements
1.1.1130	02-09-2023	• Upgrades the embedded Chromium browser to version 108.4.130
1.1.1118	11-07-2022	• Upgrades the embedded Chromium browser to version 106.0.26
1.1.1099	10-13-2022	• Includes bug fixes and improvements

Client version	Release date	Changes
1.1.1066	08-17-2022	• Upgrades the embedded Chromium browser to version 102.0.9. Microsoft Visual C++ 2019 Redistributable must be installed as a prerequisite.
1.1.1025	06-29-2022	• Adds support for UDP streaming. For more information, see Amazon AppStream 2.0 enables UDP streaming for Windows native client.
1.1.421	05-19-2022	• Includes bug fixes
1.1.414	04-26-2022	• Includes bug fixes and UI improvements
1.1.398	02-23-2022	• Includes bug fixes
1.1.394	02-08-2022	• Upgrades the embedded Chromium browser to version 97
1.1.386	12-20-2021	• Upgrades the embedded Chromium browser to version 94.4 • Includes bug fixes

Client version	Release date	Changes
1.1.360	11-15-2021	• Adds support for Linux application streaming • Adds support for Elastic fleets. For more information, see Amazon AppStream 2.0 launches Elastic fleets. • Fixes a bug with the Japanese keyboard
1.1.333	09-08-2021	• Bug fixes for the embedded Chromium browser
1.1.319	08-16-2021	• Resolves an issue with the caps lock, number lock, and scroll lock keys • Resolves an issue for the domain join sign-in experience
1.1.304	08-02-2021	• Upgrades the embedded Chromium browser to version 91 • Updated USB driver to include important fixes

Client version	Release date	Changes
1.1.294	04-26-2021	• Resolves an issue with SAML 2.0 authentic ation • Resolves a client stability issue with Windows 7 • Resolves an issue with folder sharing on client reconnection
1.1.285	03-08-2021	• Includes fixes that improve compatibility with antivirus software
1.1.257	12-28-2020	• Adds support for real-time audio-video (AV) • Adds support for using a smart card for Windows sign in to Active Directory -joined streaming instances and in-sessio n authentication for streaming applications • Resolves an issue that causes Microsoft Excel sheets to lose focus during streaming sessions

Client version	Release date	Changes
1.1.195	08-18-2020	• Improves the experience of sharing local drives and folders that belong to cloud-based persistent storage solutions such as OneDrive when file redirection is used during streaming sessions • Upgrades the embedded Chromium browser to version 81 • Resolves `AS2TrustedDomain` DNS TXT record lookup failures for domains specified in the `AS2TrustedDomains` list. These failures may occur with some URI schemes. For more information, see Create the AS2TrustedDomains DNS TXT Record to Enable Your Domain for the AppStream 2.0 Client Without Registry Changes • Resolves an intermittent issue that causes the client to stop

Client version	Release date	Changes
		functioning when audio is enabled
1.1.179	07-08-2020	• Adds support for local printer redirection • Resolves an issue with concurrent HTTP connections that prevents streaming with some proxy settings • Resolves an issue that causes file downloads for files greater than a few gigabytes to stop, and then fail • Resolves an issue that causes subsequent connection attempts to AppStream 2.0 to fail if users sign in and connect to AppStream 2.0 over SAML, disconnec t from the session without closing the AppStream 2.0 client, and then try to start a new AppStream 2.0 streaming session

Client version	Release date	Changes
1.1.160	04-28-2020	• Resolves an issue that prevents the application catalog page from opening on a Windows PC that has .NET Framework version 4.7.1 or earlier installed • Resolves an intermittent issue that causes the client to stop responding when users close the client application

Client version	Release date	Changes
1.1.156	04-22-2020	• Adds support for defining trusted subdomains for user connections in a DNS TXT record • Adds support for on-demand diagnostic log and minidump uploads • Adds support for displaying custom branding for users who stream in native application mode **Note** Users who have this version of the AppStream 2.0 client installed must have .NET Framework version 4.7.2 or later installed on the same PC. For a list of the .NET Framework versions available for download, see Download .NET Framework.

Client version	Release date	Changes
1.1.137	03-08-2020	• Reverts the updates in version 1.1.136
1.1.136	03-05-2020	• Adds support for defining trusted subdomains for user connections in a DNS TXT record
1.1.129	02-28-2020	• Adds support for native application mode • Improves the user interface for the DCV Printer experience • Resolves an issue with using Surface Pro Pen with streaming applications • Resolves an issue with downloading files with file names that have international characters
1.0.525	12-12-2019	• Resolves a DPI issue that causes the mouse cursor to point to the wrong location when a user clicks on an application during a streaming session

Client version	Release date	Changes
1.0.511	10-16-2019	• Adds support for up to 4 monitors with a maximum display resolution of 2560x1600 pixels per monitor • Adds support for up to 2 monitors with a maximum display resolution of 4096x2160 pixels per monitor on the Graphics Design and Graphics Pro instance types • Adds support for seamless user connections to streaming sessions that were started using custom uniform resource identifier (URI) redirects • Adds support for adding trusted domains for start URLs

Client version	Release date	Changes
1.0.499	09-26-2019	• Resolves an issue with client-side hardware rendering • Resolves an issue with the client not working correctly when Bluetooth headsets are connected to the local computer
1.0.480	08-20-2019	• Adds support for AppStream 2.0 file system redirection
1.0.467	07-29-2019	• Includes fixes and enhancements to ensure compatibility with updates made to AppStream 2.0 portal endpoints

Client version	Release date	Changes
1.0.407	05-16-2019	• Adds support for configuring the amount of time that users can be idle (inactive) before they are disconnected from their streaming session. For more information, see "Create a Fleet" in Create an AppStream 2.0 Fleet and Stack. • Resolves an issue with the “session alert” window appearing when a SAML 2.0 session has expired • Includes bug fixes for printing a document to a print server

Client version	Release date	Changes
1.0.375	03-07-2019	• Adds touch screen support on Windows PCs • Adds support for automatically connecting USB devices when a new streaming session starts • Adds support for running session scripts • Adds support for delivering virtualiz ed applications using the AppStream 2.0 dynamic application framework APIs

Client version	Release date	Changes
1.0.320	01-19-2019	• Adds multi monitor support for Graphics Design instances • Adds support for client display scaling factors greater than 100 percent • Adds support for AppStream 2.0 regional settings • Adds support for the AppStream 2.0 user pool • Adds support for honoring client-side proxy settings
1.0.247	11-20-2018	Initial release

Tagging Your Amazon AppStream 2.0 Resources

AWS enables you to assign metadata to your AWS resources in the form of tags. You can use these tags to help manage your AppStream 2.0 image builders, images, fleets, and stacks, and also organize data, including billing data.

You can:

- Logically group resources in different ways (for example, by purpose, owner, or environment).

  This is useful when you have many resources of the same type.
- Quickly identify a specific resource based on the tags that you've assigned to it
- Identify and control AWS costs

For example, you can identify and group AppStream 2.0 fleets that are in different environments (such as Development or Production) or that are assigned to different business units (such as HR or Marketing). You can then track the associated AWS costs for these fleets on a detailed level. To do this, sign up to get your Amazon Web Services account bill with tag key values included. For more information about setting up a cost allocation report with tags, see Monthly Cost Allocation Report in the *AWS Billing User Guide*.

Contents

- Tagging Basics
- Tag Restrictions
- Adding Tags during Resource Creation in the AppStream 2.0 Console
- Adding, Editing, and Deleting Tags for Existing Resources in the AppStream 2.0 Console
- Working with Tags by Using the AppStream 2.0 API, an AWS SDK, or AWS CLI

Tagging Basics

Tags consist of a key-value pair, similar to other AWS services tags. To tag a resource, you specify a *key* and a *value* for each tag. A key can be a general category, such as "project", "owner", or "environment", with specific associated values, and you can share the same key and value across multiple resources. You can tag an AppStream 2.0 resource immediately after you create it or later

on. If you delete a resource, the tags are removed from that resource on deletion. However, other AppStream 2.0 and AWS resources that have the same tag key are not impacted.

You can edit tag keys and values, and you can remove tags from a resource at any time. You can set the value of a tag to an empty string, but you can't set the name of a tag to null. If you add a tag that has the same key as an existing tag on that resource, the new value overwrites the old value. If you delete a resource, any tags for the resource are also deleted.

> **Note**
>
> If you plan to set up a monthly cost allocation report to track AWS costs for AppStream 2.0 resources, keep in mind that tags added to existing AppStream 2.0 resources appear in your cost allocation report on the first of the following month for resources that are renewed in that month.

Tag Restrictions

- The maximum number of tags per AppStream 2.0 resource is 50.
- The maximum key length is 128 Unicode characters in UTF-8.
- The maximum value length is 256 Unicode characters in UTF-8.
- Tag keys and values are case-sensitive.
- Do not use the "aws:" prefix in your tag names or values because it is a system tag that is reserved for AWS use. You cannot edit or delete tag names or values with this prefix. Tags with this prefix do not count against your tags per resource limit.
- Generally allowed characters are: letters, numbers, and spaces representable in UTF-8, and the following special characters: + - = . _ : / @.
- Although you can share the same key and value across multiple resources, you cannot have duplicate keys on the same resource.
- You can add tags for resources during resource creation. You can also add, edit, and delete tags for resources that are already created.

Adding Tags during Resource Creation in the AppStream 2.0 Console

When you create a resource in the AppStream 2.0 console, you can add one or more tags to manage the resource. For more information, see the following topics:

- Image builders — Launch an Image Builder to Install and Configure Streaming Applications, step 4
- Images — Step 6: Finish Creating Your Image, step 1
- Fleets — Create a Fleet, step 3
- Stacks — Create a Stack, step 2

Adding, Editing, and Deleting Tags for Existing Resources in the AppStream 2.0 Console

You can add, edit, and delete tags for existing resources by using the AppStream 2.0 console.

To add, edit, or delete tags for an existing AppStream 2.0 resource

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. From the navigation bar, select the Region that contains the resource for which you want to add, edit, or delete tags.
3. In the navigation pane, select the resource type. The resource type can be an image builder, image, fleet, or stack.
4. Select the resource from the resource list.
5. Choose **Tags**, **Add/Edit Tags**, and then do one or more of the following:
   - To add a tag, choose **Add Tag**, and type the key and value for each tag.
   - To edit a tag, modify the key and value for the tag as needed.
   - To delete a tag, choose the **Delete** icon (X) for the tag.
6. Choose **Save**.

Working with Tags by Using the AppStream 2.0 API, an AWS SDK, or AWS CLI

If you're using the AppStream 2.0 API, an AWS SDK, or the AWS Command Line Interface (AWS CLI), you can use the following AppStream 2.0 operations with the `tags` parameter to add tags when you create new resources.

> **Note**
>
> You can use spaces in tag keys and values. To indicate a space when you use the AWS CLI, use "\s" (without the quotation marks).

Task	AWS CLI	API Operation
Add one or more tags for a new fleet	create-fleet	CreateFleet
Add one or more tags for a new image builder	create-imagebuilder	CreateImageBuilder
Add one or more tags for a new stack	create-stack	CreateStack

You can use the following AppStream 2.0 operations to add, edit, remove, or list tags for existing resources:

Task	AWS CLI	API Operation
Add or overwrite one or more tags for a resource	tag-resource	TagResource
Remove one or more tags for a resource	untag-resource	UntagResource
List one or more tags for a resource	list-tags-for-resource	ListTagsForResource

When you use the AppStream 2.0 API, an AWS SDK, or AWS CLI actions to add, edit, remove, or list tags for an existing AppStream 2.0 resource, specify the resource by using its Amazon Resource Name (ARN). An ARN uniquely identifies an AWS resource and uses the following general syntax.

```
arn:aws:appstream:region:account:resourceType/resourceName
```

region

The AWS Region in which the resource was created (for example, `us-east-1`).

account

The AWS account ID, with no hyphens (for example, `123456789012`).

resourceType

The type of resource. You can tag the following AppStream 2.0 resource types: `image-builder`, `image`, `fleet`, and `stack`.

resourceName

The name of the resource.

For example, you can obtain the ARN for an AppStream 2.0 fleet by using the AWS CLI describe-fleets command. Copy the following command.

```
aws appstream describe-fleets
```

For an environment that contains a single fleet named `TestFleet`, the ARN for this resource would appear in the JSON output similar to the following.

```
"Arn": "arn:aws:appstream:us-east-1:123456789012:fleet/TestFleet"
```

After you obtain the ARN for this resource, you can add two tags by using the tag-resource command:

```
aws appstream tag-resource --resource arn:awsappstream:us-east-1:123456789012:fleet/
TestFleet --tags Environment=Test,Department=IT
```

The first tag, `Environment=Test`, indicates that the fleet is in a test environment. The second tag, `Department=IT`, indicates that the fleet is in the IT department.

You can use the following command to list the two tags that you added to the fleet.

```
aws appstream list-tags-for-resource --resource arn:aws:appstream:us-
east-1:123456789012:fleet/TestFleet
```

For this example, the JSON output appears as follows:

```
{
    "Tags": {
       "Environment" : "Test",
       "Department" : "IT"
    }
}
```

Monitoring and Reporting

Monitoring and reporting are an important part of maintaining the reliability, availability, and performance of your Amazon AppStream 2.0 streaming instances and providing your users with a responsive streaming experience.

Contents

- Monitoring Amazon AppStream 2.0 Resources
- AppStream 2.0 Usage Reports
- Logging AppStream 2.0 API Calls with AWS CloudTrail

For more information, see the following:

- Creating custom logging and CloudWatch alerting in AppStream 2.0
- Getting started with your AWS Health Dashboard – Your account health
- Monitoring AWS Health events with EventBridge

Monitoring Amazon AppStream 2.0 Resources

AppStream 2.0 publishes metrics to Amazon CloudWatch to enabled 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. For more information, see the Amazon CloudWatch User Guide.

Contents

- Viewing Fleet Usage Using the Console
- Viewing Instance and Session Performance Metrics Using the Console
- AppStream 2.0 Metrics and Dimensions

Viewing Fleet Usage Using the Console

You can monitor your Amazon AppStream 2.0 fleet usage using the AppStream 2.0 or CloudWatch console.

To view fleet usage in the AppStream 2.0 console

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. In the left pane, choose **Fleets**.
3. Select a fleet and choose its **Fleet Usage** tab.
4. By default, the graph displays the following metrics:
   - `ActualCapacity`, `InUseCapacity`, `DesiredCapacity`, `AvailableCapacity`, `PendingCapacity`, and `CapacityUtilization` for single-session fleets.
   - `ActualUserSessionCapacity`, `ActiveUserSessionCapacity`, `AvailableUserSessionCapacity`, `DesiredUserSessionCapacity`, `PendingUserSessionCapacity`, and `CapacityUtilization` for multi-session fleets.

To view fleet usage in the CloudWatch console

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the left pane, choose **Metrics**.
3. Choose the **AppStream** namespace and then choose **Fleet Metrics**.
4. Select the metrics to graph.

Viewing Instance and Session Performance Metrics Using the Console

You can monitor Amazon AppStream 2.0 fleet instances and session performance using the AppStream 2.0 console or the CloudWatch console.

Performance metrics are collected at a 5-minute interval. After a new session is provisioned, the first metric data point will show up in 5 minutes. Subsequent metric data points will be available at every 5-minute interval.

> **Note**
>
> Performance metrics are currently available only for multi-session fleets

To view instance and session in the AppStream 2.0 console

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.

2. In the left pane, choose **Fleets**.
3. Select a fleet and choose **View Details** and **View Sessions**.
4. Select a session to view the metrics.
5. By default, the graph displays the following metrics:
   - Instance metrics
     - CpuUtilizationInstance
     - MemoryUtilizationInstance
     - PagingFileUtilizationInstance
     - DiskUtilizationInstance
   - Session metrics
     - CpuUtilizationSession
     - MemoryUtilizationSession

To view instance and session performance in the CloudWatch console

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the left pane, choose **Metrics**.
3. Choose the **AppStream** namespace and then choose **Fleet Instance Metrics** or **Fleet Session Metrics**.
4. Select the metrics to graph.

AppStream 2.0 Metrics and Dimensions

Amazon AppStream 2.0 sends the following metrics and dimension information to Amazon CloudWatch.

All of the following metrics except `InsufficientConcurrencyLimitError` apply to Always-On and On-Demand fleets. The only metrics that apply to Elastic fleets are `InUseCapacity` and `InsufficientCapacityError`.

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

Fleet Usage Metrics for Single-session Fleets

Metric	Description
`ActualCapacity`	The total number of instances that are available for streaming or are currently streaming. `ActualCapacity = AvailableCapacity + InUseCapacity` Units: Count Valid statistics: Average, Minimum, Maximum
`AvailableCapacity`	The number of idle instances currently available for user sessions. `AvailableCapacity = ActualCapacity - InUseCapacity` Units: Count Valid statistics: Average, Minimum, Maximum
`CapacityUtilization`	The percentage of instances in a fleet that are being used, using the following formula. `CapacityUtilization = (InUseCapacity/ActualCapacity) * 100` 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
`DesiredCapacity`	The 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 = ActualCapacity + PendingCapacity`

Metric	Description
	Units: Count Valid statistics: Average, Minimum, Maximum
`InUseCapacity`	The number of instances currently being used for streaming sessions. One `InUseCapacity` count represents one streaming session. Units: Count Valid statistics: Average, Minimum, Maximum
`PendingCapacity`	The 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
`RunningCapacity`	The total number of instances currently running. Represents the number of concurrent streaming sessions that can be supported by the fleet in its current state. This metric is provided for Always-On fleets only, and has the same value as the `ActualCapacity` metric. Units: Count Valid statistics: Average, Minimum, Maximum

Metric	Description
`InsufficientCapacityError`	The number of session requests rejected due to lack of 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
`InsufficientConcurrencyLimitError`	The number of Elastic fleet session requests rejected due to reaching max concurrent streaming 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

Fleet Usage Metrics for Multi-session Fleets

Metric	Description
`CapacityUtilization`	The percentage of sessions in a fleet that are being used, using the following formula. `UserSessionCapacityUtilization = (ActiveUserSessions / ActualUserSessionCapacity) * 100` 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

Metric	Description
`ActualUserSessionCapacity`	The total number of session slots that are available for streaming or are currently streaming. `ActualUserSessionCapacity = AvailableUserSessionCapacity + ActiveUserSessionCapacity` Units: Count Valid statistics: Average, Minimum, Maximum
`AvailableUserSessionCapacity`	The number of idle session slots currently available for user sessions. `AvailableUserSessionCapacity = ActualUserSessionCapacity - ActiveUserSessions` Units: Count Valid statistics: Average, Minimum, Maximum
`DesiredUserSessionCapacity`	The total number of session slots that are either running or pending. This represents the total number of concurrent streaming sessions your fleet can support in a steady state. `DesiredUserSessionCapacity = ActualUserSessionCapacity + PendingUserSessionCapacity` Units: Count Valid statistics: Average, Minimum, Maximum
`ActiveUserSessionCapacity`	The number of user sessions currently being used for streaming sessions. Units: Count Valid statistics: Average, Minimum, Maximum

Metric	Description
`PendingUserSessionCapacity`	The number of session slots 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
`RunningUserSessionCapacity`	The total number of session slots currently that are available for streaming or are currently streaming. Represents the number of concurrent streaming sessions that can be supported by the fleet in its current state. This metric is provided for Always-On fleets only, and has the same value as the `ActualUserSessionCapacity` metric. Units: Count Valid statistics: Average, Minimum, Maximum

Instance and Session Performance Metrics for Multi-session Fleets

Metric	Description
`CpuUtilizationInstance`	The percentage of allocated compute units that are currently in use on the instance. Units: Percent
`MemoryUtilizationInstance`	The percentage of allocated physical memoty units that are currently in use on the instance. Units: Percent

Metric	Description
`PagingFileUtilizationInstance`	The percentage of the paging file that is currently in use to extend the Memory (RAM) capacity. Units: Percent
`DiskUtilizationInstance`	The percentage of disk units that are currently in use to run programs and carry out tasks on the instance. Units: Percent
`CpuUtilizationSession`	The percentage of allocated compute units that are currently in use by the session. Units: Percent
`MemoryUtilizationSession`	The percentage of allocated physical memoty units that are currently in use by the session. Units: Percent

Dimensions for Amazon AppStream 2.0 Metrics

To filter the metrics provided by Amazon AppStream 2.0, use the following dimensions.

Metric Type	Dimension	Description	Metrics
Fleet metrics	`Fleet`	The name of the fleet.	Fleet capacity metrics
Fleet Instance Metrics	`Fleet Name`	The name of the fleet.	Fleet instance performance metrics
Fleet Instance Metrics	`Instance Id`	The instance identifier.	Fleet instance

Metric Type	Dimension	Description	Metrics
			performance metrics
Fleet Session Metrics	`Fleet Name`	The name of the fleet.	Fleet session performance metrics
Fleet Session Metrics	`Instance Id`	The instance identifier.	Fleet session performance metrics
Fleet Session Metrics	`Session Id`	The session identifier.	Fleet session performance metrics

AppStream 2.0 Usage Reports

You can subscribe to Amazon AppStream 2.0 usage reports to receive detailed reports about how your users are using the service. Each day that users launch one or more streaming sessions in your AWS account, two .csv files are exported to your Amazon Simple Storage Service (Amazon S3) account.

> **Note**
>
> To enable AppStream 2.0 usage reports, you must use an image that uses a version of the AppStream 2.0 agent released on or after May 7, 2019.

Contents

- Enable AppStream 2.0 Usage Reports
- AppStream 2.0 Usage Reports Fields
- Create Custom Reports and Analyze AppStream 2.0 Usage Data

Enable AppStream 2.0 Usage Reports

To receive usage reports, you subscribe to them by using the AppStream 2.0 console, the AWS Command Line Interface (AWS CLI), or the `CreateUsageReportSubscription` API operation. You must enable usage reports separately for each AWS Region for which you want to receive usage data.

> **Note**
>
> You can start or stop your subscription to usage reports at any time. There is no charge for subscribing to usage reports, but standard Amazon S3 charges may apply to reports that are stored in your S3 bucket. For more information, see Amazon S3 Pricing.

To subscribe to usage reports for AppStream 2.0 by using the AppStream 2.0 console, perform the following steps.

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. Choose the AWS Region for which you want to enable usage reports.
3. In the navigation pane, choose **Usage Reports**.
4. Choose **Enabled**, and then choose **Apply**.

If you enabled on-instance session scripts and Amazon S3 logging for your session script configuration, AppStream 2.0 created an S3 bucket to store the script output. The bucket is unique to your account and Region. When you enable usage reporting in this case, AppStream 2.0 uses the same bucket to store your usage reports. If you haven't already enabled on-instance session scripts, when you enable usage reports, AppStream 2.0 creates a new S3 bucket in the following location:

```
appstream-logs-region-code-account-id-without-hyphens-random-identifier
```

region-code

The AWS Region code for the Region in which usage reporting is enabled.

account-id-without-hyphens

Your Amazon Web Services account identifier. The random ID ensures that there is no conflict with other buckets in the same Region. The first part of the bucket name, `appstream-logs`, does not change across accounts or Regions.

For example, if you enable usage reporting in the US West (Oregon) Region (us-west-2) on account number 123456789012, AppStream 2.0 creates an Amazon S3 bucket within your account in that Region similar to the name shown in the following example:

```
appstream-logs-us-west-2-1234567890123-abcdefg
```

Only an administrator with sufficient permissions can delete this bucket.

AppStream 2.0 Sessions Reports

For each day that users launch at least one streaming session in your Amazon Web Services account, AppStream 2.0 exports a sessions report to your Amazon S3 bucket. The report, named **daily-session-report-[YYYY]-[MM]-[DD].csv**, is stored in a nested folder structure in your Amazon S3 account, using the following folder path:

[bucket_name]/sessions/schedule=DAILY/year=[YYYY]/month=[MM]/day=[DD]/

This nesting structure facilitates partitioning if you choose to query your reports by using Amazon Athena. Athena is a serverless, interactive query service that you can use to analyze data stored in your S3 buckets using standard SQL. For more information, see Create Custom Reports and Analyze AppStream 2.0 Usage Data.

Each user session is described in a single record in a sessions report. Sessions reports are generated daily according to UTC time within 24 hours of the close of the day that is the subject of the report. If a session spans more than one day, the session record appears in the sessions report corresponding to the day in which the session ends. For information about the data included in sessions reports, see Sessions Report Fields.

AppStream 2.0 Applications Reports

For each day that users launch at least one application during their streaming sessions, AppStream 2.0 exports an applications report to your Amazon S3 bucket. The report, named **daily-app-report-**

[YYYY]-[MM]-[DD].csv, is stored in a nested folder structure in your Amazon S3 account, using the following folder path:

[bucket_name]/applications/schedule=DAILY/year=[YYYY]/month=[MM]/day=[DD]/

This nesting structure facilitates partitioning if you choose to query your reports by using Amazon Athena. Athena is a serverless, interactive query service that you can use to analyze data stored in your S3 buckets using standard SQL. For more information, see Create Custom Reports and Analyze AppStream 2.0 Usage Data.

Each application launch is described in a single record in an applications report. For example, if a user launches five separate applications during a session, five separate records appear in the relevant applications report. An application is recorded as launched if any of the following events occurs:

- The application is launched directly when the session begins, because the application ID is embedded in either the streaming URL or the relay state.
- A user chooses the application from the application catalog when launching a new streaming session.
- A user chooses the application from the application catalog list during a streaming session.

The applications report doesn't include applications that are launched in other ways. For example, if you provide users with access to Windows Explorer, PowerShell, or the Windows desktop **Start** menu, and users use those tools to launch applications directly, or if another program or script launches an application, those application launches are not included in the applications report.

Applications reports are generated daily according to UTC time within 24 hours of the close of the day that is the subject of the report. If a session spans more than one day, applications launched during the session are reflected in the applications report corresponding to the day in which the session ends. For information about the data included in applications reports, see Applications Report Fields.

AppStream 2.0 Usage Reports Fields

This topic provides information about the fields included in AppStream 2.0 usage reports.

Contents

- Sessions Report Fields

- Applications Report Fields

Sessions Report Fields

The following table describes the fields included in AppStream 2.0 sessions reports.

Field name	Description
`user_session_id`	The unique identifier (ID) for the session.
`aws_account_id`	The Amazon Web Services account ID.
`region`	The AWS Region.
`session_start_time`	The date and time that the session started. Must be specified in ISO 8601 format and as UTC.
`session_end_time`	The date and time that the session ended. Must be specified in ISO 8601 format and as UTC.
`session_duration_in_seconds`	The duration of the session in seconds.
`user_id`	The unique ID for the user within the authentication type.
`user_arn`	The Amazon Resource Name (ARN) for the user.
`authentication_type`	The method used to authenticate the user.

Field name	Description
	Possible values: `CUSTOM` | `SAML` | `USERPOOL`
`authentication_type_user_id`	The concatenation of the user ID and authentication type, which uniquely identifies the user for the purpose of assessing user fees. For more information, see AppStream 2.0 Pricing.
`fleet_name`	The name of the fleet associated with the session.
`stack_name`	The name of the stack associated with the session.
`instance_type`	The AppStream 2.0 instance type used for the session. For a list of instance types, see AppStream 2.0 Pricing.
`eni_private_ip_address`	The IP address of the elastic network interface used by the AppStream 2.0 instance for network communications.
`connected_at_least_once`	Indicates whether the user connected to the session at least once. Possible values: `true` | `false`

Field name	Description
`client_ip_addresses`	The IP addresses associated with the user device or devices used to connect to the session. If the user connected and then disconnected from the session more than once, up to the last 10 distinct IP addresses are stored, separated by semicolons.
`google_drive_enabled`	Indicates whether Google Drive was enabled as a persistent storage option for the session. For more information, see Enable and Administer Google Drive for Your AppStream 2.0 Users. Possible values: `true` | `false`
`one_drive_enabled`	Indicates whether OneDrive was enabled as a persistent storage option for the session. For more information, see Enable and Administer Google Drive for Your AppStream 2.0 Users. Possible values: `true` | `false`
`home_folders_storage_location`	The Amazon S3 bucket used for files that are stored using home folders.

Field name	Description
`user_settings_clipboard_copy_from_local_device`	Indicates whether the user was able to copy data from the local device to the streaming session using the clipboard during the session. Possible values: `ENABLED` | `DISABLED`
`user_settings_clipboard_copy_to_local_device`	Indicates whether the user was able to copy data from the streaming session to the local device using the clipboard during the session. Possible values: `ENABLED` | `DISABLED`
`user_settings_file_upload`	Indicates whether the user was able to upload files from the local device to the streaming session during the session. Possible values: `ENABLED` | `DISABLED`
`user_settings_file_download`	Indicates whether the user was able to download files from the streaming session to the local device during the session. Possible values: `ENABLED` | `DISABLED`

Field name	Description
`user_settings_printing_to_local_device`	Indicates whether the user was able to print files from the streaming session to the local device during the session. Possible values: `ENABLED` | `DISABLED`
`application_settings_enabled`	Indicates whether applicati on settings persistence was enabled for the session. Possible values: `true` | `false`
`domain_joined`	Indicates whether the AppStream 2.0 streaming instance was joined to an Active Directory domain at session launch. For more information, see Using Active Directory with AppStream 2.0. Possible values: `Y` | `N`
`max_session_duration`	The maximum allowed duration of the session, in seconds.
`session_type`	The session type. Possible values: `ALWAYS_ON` | `ON_DEMAND`

Field name	Description
`stream_view`	The stream view. Possible values: `APPLICATION` | `DESKTOP`
`streaming_experience_settings_protocol`	The protocol that the session ended streaming with. Possible values: `UDP` | `TCP`
`instance_id`	The instance ID associated with the user session.
`is_multisession`	Indicates whether the session belongs to a multi-session fleet. Possible values: `true` | `false`

Applications Report Fields

The following table describes the fields included in AppStream 2.0 applications reports.

Field name	Description
user_session_id	The unique identifier (ID) for the session.
application_name	The name of the application, as specified in Image Assistant. This value is provided when a user launches an application through the AppStream 2.0 interface.

Field name	Description
schedule	The frequency with which reports are generated. Possible value: DAILY
year	The year of the report.
month	The month of the report.
day	The day of the report.

Create Custom Reports and Analyze AppStream 2.0 Usage Data

Amazon Athena is a serverless, interactive query service that you can use to analyze data stored in your S3 buckets using standard SQL queries. You can use Athena to aggregate your usage reports or generate other types of custom reports.

Contents

- Create an AWS Glue Crawler
- Create a Data Catalog by Using the AWS Glue Crawler
- Create and Run Athena Queries
- Working with Athena Queries

Create an AWS Glue Crawler

AWS Glue is a fully managed extract, transform, and load (ETL) service that lets you create a database from your Amazon S3 data and query that database by using Athena. This database is also referred to as an AWS Glue Data Catalog. An AWS Glue crawler can automatically detect the schema of your Amazon S3 data and create the corresponding database and tables. AppStream 2.0 provides an AWS CloudFormation template that you can use to create the necessary AWS Glue resources.

> **Important**
>
> Completing the steps in the following procedure creates an AWS Glue crawler. However, these steps don't start the crawler. To start the crawler, you must perform the steps in the next procedure. For more information about AWS Glue crawlers, see Defining Crawlers.

To create an AWS Glue crawler

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. Choose the AWS Region for which you have subscribed to usage reports.
3. In the navigation pane, choose **Usage Reports**, and verify that usage reports logging is enabled.
4. On the **Report Details** tab, in the paragraph next to **Analytics**, choose the **CloudFormation template** link.

   Choosing the link opens the AWS CloudFormation console, where you can review the parameters of the AWS CloudFormation stack specified by the template before you run it. The template, when run, creates an AWS Glue crawler and several sample Athena queries.
5. On the **Specify Details** page, next to **ScheduleExpression**, either keep the default value or specify a different cron expression value for the frequency that you want to run the crawler. Do not change any other default value. When you're done, choose **Next**.

   By default, the crawler is scheduled to run on a daily basis, but you can configure the crawler to run weekly, monthly, or on another frequency. For information about cron syntax, see Cron Expressions.
6. On the **Options** page, keep the default values, and choose **Next**.
7. On the **Review** page, select the check box next to "I acknowledge that AWS CloudFormation might create IAM resources with custom names," and then choose **Create**.

   You must have sufficient AWS Glue and AWS Identity and Access Management (IAM) permissions to create and run the AWS CloudFormation stack. If you don't have the required permissions, ask your Amazon Web Services account administrator either to perform these steps in your account or to grant you the following permissions.

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

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:CreateNamedQuery",
                "athena:BatchGetNamedQuery",
                "athena:GetNamedQuery",
                "athena:StartQueryExecution",
                "athena:GetQueryResults",
                "athena:GetQueryExecution",
                "athena:ListNamedQueries",
                "cloudformation:DescribeStacks",
                "cloudformation:GetStackPolicy",
                "cloudformation:DescribeStackEvents",
                "cloudformation:CreateStack",
                "cloudformation:GetTemplate",
                "cloudformation:ListChangeSets",
                "cloudformation:ListStackResources",
                "iam:GetRole",
                "iam:CreateRole",
                "iam:GetRolePolicy",
                "s3:GetBucketLocation",
                "s3:ListBucketMultipartUploads",
                "s3:ListBucket",
                "s3:ListMultipartUploadParts",
                "s3:PutObject",
                "s3:GetObject",
                "s3:AbortMultipartUpload"
            ],
            "Resource": [
                "arn:aws:iam::*:role/AppStreamUsageReports-
AppStreamUsageReportGlueRole*",
                "arn:aws:cloudformation:*:*:stack/AppStreamUsageReports/*",
                "arn:aws:athena:*:*:workgroup/primary",
                "arn:aws:s3:::aws-athena-query-results-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:AttachRolePolicy",
                "iam:PutRolePolicy",
                "s3:GetObject",
                "s3:ListBucket"
```

```
            ],
            "Resource": [
                "arn:aws:s3:::appstream-logs-*",
                "arn:aws:iam::*:role/AppStreamUsageReports-
AppStreamUsageReportGlueRole*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/AppStreamUsageReports-
AppStreamUsageReportGlueRole*"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "glue.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudformation:GetTemplateSummary",
                "glue:GetResourcePolicy",
                "glue:GetCrawlers",
                "glue:BatchGetCrawlers",
                "glue:GetClassifiers",
                "glue:CreateClassifier",
                "glue:ListCrawlers",
                "glue:GetTags",
                "glue:GetCrawlerMetrics",
                "glue:GetClassifier",
                "tag:GetResources"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "athena:RunQuery",
            "Resource": "arn:aws:athena:*:*:workgroup/primary"
        },
```

```
        {
            "Effect": "Allow",
            "Action": [
                "glue:GetTables",
                "glue:GetPartitions",
                "glue:GetTable"
            ],
            "Resource": [
                "arn:aws:glue:*:*:table/appstream-usage/*",
                "arn:aws:glue:*:*:database/appstream-usage",
                "arn:aws:glue:*:*:catalog"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "glue:GetDatabase",
                "glue:CreateDatabase",
                "glue:GetDatabases"
            ],
            "Resource": [
                "arn:aws:glue:*:*:database/appstream-usage",
                "arn:aws:glue:*:*:catalog"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "glue:GetCrawler",
                "glue:StartCrawler",
                "glue:CreateCrawler"
            ],
            "Resource": "arn:aws:glue:*:*:crawler/appstream-usage*"
        },
        {
            "Effect": "Allow",
            "Action": "glue:GetCatalogImportStatus",
            "Resource": "arn:aws:glue:*:*:catalog"
        }
    ]
}
```

Create a Data Catalog by Using the AWS Glue Crawler

The AWS Glue crawler, when run, creates a Data Catalog and schema that are mapped to the structure of your sessions and applications reports. Each time a new report is stored in your Amazon S3 bucket, you must run the crawler to update your AWS Glue Data Catalog with the data from the new report.

> **Note**
>
> Charges may apply to the running of your AWS Glue crawler. For more information, see AWS Glue Pricing.

1. Open the AWS Glue console at https://console.aws.amazon.com/glue/.
2. Choose the AWS Region for which you have subscribed to usage reports.
3. Select the check box next to the crawler named **appstream-usage-sessions-crawler**, and then choose **Run crawler**. Repeat this step for the crawler named **appstream-usage-apps-crawler**.

   Performing these steps runs the crawlers and schedules them to run automatically according to the schedule specified in the AWS CloudFormation stack.
4. After both crawlers finish running, in the navigation pane, choose **Databases**. A database named **appstream-usage**, which represents your usage reports, displays. This database is an AWS Glue Data Catalog that was created when **appstream-usage-sessions-crawler** and **appstream-usage-apps-crawler** were run.
5. To view the tables in the database, choose **appstream-usage**, **Tables**. Two tables, **applications** and **sessions**, which represent your applications and sessions usage reports respectively, display. Choose either table to view its schema.

   You can now query these tables in Athena by using SQL.

Create and Run Athena Queries

To query your usage reports by using Athena, perform the following steps.

> **Note**
>
> Charges may apply to Athena queries that you run. For more information, see Amazon Athena Pricing.

1. Open the Athena console at https://console.aws.amazon.com/athena/.
2. Under **Database**, choose **appstream-usage**.
3. In the query pane, enter a SQL query, and choose **Run query**.

Working with Athena Queries

This section provides SQL queries that you can run in Athena to analyze the usage reports data in your Amazon S3 bucket.

To create a consolidated report of all sessions in a given month, run the following query:

```
SELECT *
FROM "appstream-usage"."sessions"
WHERE year='four-digit-year'
AND month='two-digit-month'
```

You can also perform join operations between the **applications** and **sessions** tables in your query. For example, to view the distinct users who launched each application in a given month, run the following query:

```
SELECT DISTINCT apps.application_name, sessions.user_id
FROM "appstream-usage"."applications" apps
   INNER JOIN "appstream-usage"."sessions" sessions ON (apps.user_session_id =
 sessions.user_session_id AND sessions.year='four-digit-year' AND sessions.month='two-
digit-month')
WHERE apps.year='four-digit-year'
  AND apps.month='two-digit-month'
ORDER BY 1, 2
```

Athena query results are stored as .csv files in an Amazon S3 bucket in your account that is named `aws-athena-query-results-`*`account-id-without-hyphens`*`-`*`region-code`*. For ease in locating query results, choose **Save as** and provide a name for your query before you run it. You

can also choose the download icon in the **Athena Results** pane to download the results of the query as a .csv file.

To enhance performance and reduce costs, Athena uses partitioning to reduce the amount of data scanned in queries. For more information, see Partitioning Data. Usage reports are partitioned in your Amazon S3 buckets by year, month, and day. You can restrict your queries to certain date range partitions using the **year**, **month**, and **day** fields as conditions in your queries. For example, the following query ingests only the sessions reports for the week of May 19, 2019.

```
SELECT SUBSTRING(session_start_time, 1, 10) AS report_date,
    COUNT(DISTINCT user_session_id) AS num_sessions
FROM "appstream-usage"."sessions"
WHERE year='2019'
   AND month='05'
   AND day BETWEEN '19' and '25'
GROUP BY 1
ORDER BY 1
```

In contrast, the following query produces identical results, but because it isn't restricted to any partitions, it ingests all sessions reports stored in your Amazon S3 bucket.

```
SELECT SUBSTRING(session_start_time, 1, 10) AS report_date,
    COUNT(DISTINCT user_session_id) AS num_sessions
FROM "appstream-usage"."sessions"
WHERE session_end_time BETWEEN '2019-05-19' AND '2019-05-26'
GROUP BY 1
ORDER BY 1
```

If a session spans more than one day, the session and application records appear in the sessions and applications reports, respectively, corresponding to the day in which the session ended. For this reason, if you need to find records that relate to all sessions that were active during a given date range, consider expanding the partition set of your query by the maximum session length you have configured for your fleets.

For example, to view all sessions that were active for a given fleet during a calendar month, where the fleet had a maximum session duration of 100 hours, run the following query to expand your partition set by five days.

```
SELECT *
FROM "appstream-usage"."sessions"
```

```
WHERE fleet_name = 'fleet_name'
   AND session_start_time BETWEEN '2019-05-01' AND '2019-06-01'
   AND year='2019'
   AND (month='05' OR (month='06' AND day<='05'))
ORDER BY session_start_time
```

The AWS CloudFormation template that created the AWS Glue crawlers also created and saved several sample queries in your Athena account that you can use to analyze your usage data. These sample queries include the following:

- Aggregated monthly session report
- Average session length per stack
- Number of sessions per day
- Total streaming hours per user

> **Note**
>
> On-demand usage charges are rounded up to the next hour for each session.

- Distinct users per app

To use any of these queries, perform the following steps.

1. Open the Athena console at https://console.aws.amazon.com/athena/.
2. Choose **Saved Queries**. The five queries noted before this procedure should display. The name of each query begins with "AS2." For example, "AS2_users_per_app_curr_mo."
3. To run a query, choose the query name rather than the option next to the name.
4. The text of the query appears in the query pane. Choose **Run query**.

To view these queries in a separate AWS CloudFormation template, see athena-sample-queries-appstream-usage-data_template.yml in the *AWS Code Sample Catalog.*

Logging AppStream 2.0 API Calls with AWS CloudTrail

Amazon AppStream 2.0 is integrated with AWS CloudTrail. CloudTrail is a service that provides a record of actions taken by a user, role, or an AWS service in AppStream 2.0. CloudTrail captures API calls for AppStream 2.0 as events. The calls captured include calls from the AppStream 2.0 console

and code calls to the AppStream 2.0 API operations. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for AppStream 2.0. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**. You can use the information collected by CloudTrail to determine details such as request information. For example, CloudTrail collects the following information: What request was made to AppStream 2.0, the IP address from which the request was made, who made the request, and when it was made.

To learn more about CloudTrail, including how to configure and enable it, see the AWS CloudTrail User Guide.

AppStream 2.0 Information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When supported event activity occurs in AppStream 2.0, that activity is recorded in a CloudTrail event along with other AWS service events in **Event history**. You can view, search, and download recent events in your AWS account. For more information, see Viewing Events with CloudTrail Event History.

For an ongoing record of events in your AWS account, including events for AppStream 2.0, create a trail. A *trail* enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all AWS Regions. The trail logs events from all Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see the following:

- Overview for Creating a Trail
- CloudTrail Supported Services and Integrations
- Configuring Amazon SNS Notifications for CloudTrail
- Receiving CloudTrail Log Files from Multiple Regions and Receiving CloudTrail Log Files from Multiple Accounts

AppStream 2.0 supports logging the following actions as events in CloudTrail log files:

- AssociateFleet
- BatchAssociateUserStack
- BatchDisassociateUserStack
- CopyImage

- CreateDirectoryConfig
- CreateFleet
- CreateImageBuilder
- CreateImageBuilderStreamingURL
- CreateStack
- CreateStreamingURL
- DeleteDirectoryConfig
- DeleteFleet
- DeleteImage
- DeleteImageBuilder
- DeleteImagePermissions
- DeleteStack
- DescribeDirectoryConfigs
- DescribeFleets
- DescribeImageBuilders
- DescribeImagePermissions
- DescribeImages
- DescribeSessions
- DescribeStacks
- DescribeUserStackAssociations
- ExpireSession
- ListAssociatedFleets
- ListAssociatedStacks
- ListTagsForResource
- StartFleet
- StartImageBuilder
- StopFleet
- StopImageBuilder
- TagResource
- UntagResource

- UpdateDirectoryConfig
- UpdateFleet
- UpdateImagePermissions
- UpdateStack

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

- Whether the request was made with root or IAM user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see the CloudTrail userIdentity Element.

Example: AppStream 2.0 Log File Entries

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

The following example shows a CloudTrail log entry that demonstrates the `AssociateFleet` event.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE:janeroe",
    "arn": "arn:aws:sts:: 123456789012:assumed-role/Admin/janeroe",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2019-03-12T06:41:50Z"
      },
```

```
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam:: 123456789012:role/Admin",
        "accountId": "123456789012",
        "userName": "Admin"
      }
    }
  },
  "eventTime": "2019-03-12T06:58:09Z",
  "eventSource": "appstream.amazonaws.com",
  "eventName": "AssociateFleet",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "198.51.100.15",
  "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6) AppleWebKit/537.36
 (KHTML, like Gecko) Chrome/72.0.3626.121 Safari/537.36",
  "requestParameters": {
    "fleetName": "ExampleFleet1",
    "stackName": "ExampleStack1"
  },
  "responseElements": null,
  "requestID": "3210a159-4494-11e9-8017-873084baf125",
  "eventID": "a6fbde60-a55a-46fe-87d4-89ead558dffd",
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
}
```

The following example shows a CloudTrail log entry that demonstrates the `CreateImage` event when an image is created using the AppStream 2.0 image builder.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "arn": "arn:aws:appstream:us-east-1: 123456789012:image-builder/
ExampleImageBuilder",
    "accountId": "123456789012"
  },
  "eventTime": "2019-03-21T22:32:05Z",
  "eventSource": "appstream.amazonaws.com",
  "eventName": "CreateImage",
  "awsRegion": "us-east-1",
  "requestParameters": null,
  "responseElements": null,
```

```
  "eventID": "12b2d6e2-c9a9-402e-8886-2c388d3df610",
  "readOnly": false,
  "eventType": "AwsServiceEvent",
  "recipientAccountId": "123456789012",
  "serviceEventDetails": {
    "ImageName": "ExampleImage1",
    "ImagePlatform": "WINDOWS",
    "PublicBaseImageReleasedDate": "Tue Jan 15 22:19:56 UTC 2019",
    "ImageDisPlayName": "Example Image 1",
    "ImageBuilderSupported": "True",
    "ImageCreatedTime": "Thu Mar 21 22:32:05 UTC 2019",
    "ImageDescription": "Example image for testing",
    "ImageState": "PENDING"
  }
}
```

Security in Amazon AppStream 2.0

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

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

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS Compliance Programs. For information about the compliance programs that apply to AppStream 2.0, see AWS Services in Scope by Compliance Program.
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations

This documentation helps you understand how to apply the shared responsibility model when using AppStream 2.0. It shows you how to configure AppStream 2.0 to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your AppStream 2.0 resources.

Contents

- Data Protection in Amazon AppStream 2.0
- Identity and Access Management for Amazon AppStream 2.0
- Logging and Monitoring in Amazon AppStream 2.0
- Compliance Validation for Amazon AppStream 2.0
- Resilience in Amazon AppStream 2.0
- Infrastructure Security in Amazon AppStream 2.0
- Security Groups in Amazon AppStream 2.0
- Update Management in Amazon AppStream 2.0
- Amazon AppStream 2.0 Cross-Service Confused Deputy Prevention

Data Protection in Amazon AppStream 2.0

The AWS shared responsibility model applies to data protection in Amazon AppStream 2.0. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. This content includes the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the Data Privacy FAQ. For information about data protection in Europe, see the AWS Shared Responsibility Model and GDPR blog post on the *AWS Security Blog*.

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

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We recommend TLS 1.2.
- Set up API and user activity logging with AWS CloudTrail.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing personal data that is stored in Amazon S3.
- If you require FIPS 140-2 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-2.

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form fields such as a **Name** field. This includes when you work with AppStream 2.0 or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

Encryption at Rest

AppStream 2.0 fleet instances are ephemeral in nature. After a user's streaming session is finished, the underlying instance and its associated Amazon Elastic Block Store (Amazon EBS) volume are terminated. In addition, AppStream 2.0 periodically recycles unused instances for freshness.

When you enable application settings persistence or home folders for your users, the data that is generated by your users and stored in Amazon Simple Storage Service buckets is encrypted at rest. AWS Key Management Service is a service that combines secure, highly available hardware and software to provide a key management system scaled for the cloud. Amazon S3 uses AWS Managed CMKs to encrypt your Amazon S3 object data.

Encryption in Transit

The following table provides information about how data is encrypted in transit. Where applicable, other data protection methods for AppStream 2.0 are also listed.

Data	Network path	How protected
Web assets This traffic includes assets such as images and JavaScript files.	Between AppStream 2.0 users and AppStream 2.0	Encrypted using TLS 1.2
Pixel and related streaming traffic	Between AppStream 2.0 users and AppStream 2.0	Encrypted using 256-bit Advanced Encryption Standard (AES-256) Transported using TLS 1.2
API traffic	Between AppStream 2.0 users and AppStream 2.0	Encrypted using TLS 1.2 Requests to create a connection are signed using SigV4
Application settings and home folder data generated by users Applicable when application settings	Between AppStream 2.0 users and Amazon S3	Encrypted using Amazon S3 SSL endpoints

Data	Network path	How protected
persistence and home folders are enabled.		
AppStream 2.0-managed traffic	Between AppStream 2.0 streaming instances and: • AppStream 2.0 management services • AWS services and resources in your Amazon Web Services account • Non-AWS services and resources (such as Google Drive and Microsoft OneDrive)	Encrypted using TLS 1.2 Requests to create a connection are signed using SigV4 where applicable

Administrator Controls

AppStream 2.0 provides administrative controls that you can use to limit the ways in which users can transfer data between their local computer and an AppStream 2.0 fleet instance. You can limit or disable the following when you create or update an AppStream 2.0 stack:

- Clipboard/copy and paste actions
- File upload and download, including folder and drive redirection
- Printing

When you create an AppStream 2.0 image, you can specify which USB devices are available to redirect to AppStream 2.0 fleet instances from the AppStream 2.0 client for Windows. The USB devices that you specify will be available for use during users' AppStream 2.0 streaming sessions. For more information, see Qualify USB Devices for Use with Streaming Applications.

Application Access

By default, AppStream 2.0 enables the applications that you specify in your image to launch other applications and executable files on the image builder and fleet instance. This ensures that applications with dependencies on other applications (for example, an application that launches the browser to navigate to a product website) function as expected. Make sure that you configure your administrative controls, security groups, and other security software to grant users the

minimum permissions required to access resources and transfer data between their local computers and fleet instances.

You can use application control software, such as Microsoft AppLocker, and policies to control which applications and files your users can run. Application control software and policies help you control the executable files, scripts, Windows installer files, dynamic-link libraries, and application packages that your users can run on AppStream 2.0 image builders and fleet instances.

> **Note**
>
> The AppStream 2.0 agent software relies on the Windows command prompt and Windows Powershell to provision streaming instances. If you choose to prevent users from launching the Windows command prompt or Windows Powershell, the policies must not apply to the Windows NT AUTHORITY\SYSTEM or users in the Administrators group.

Rule type	Action	Windows user or group	Name/Path	Condition	Description
Executable	Allow	NT AUTHORITY \System	*	Path	Required for the AppStream 2.0 agent software
Executable	Allow	BUILTIN\A dministrators	*	Path	Required for the AppStream 2.0 agent software
Executable	Allow	Everyone	%PROGRAMF ILES%\nod ejs*	Path	Required for the AppStream 2.0 agent software

Rule type	Action	Windows user or group	Name/Path	Condition	Description
Executable	Allow	Everyone	%PROGRAMFILES%\NICE*	Path	Required for the AppStream 2.0 agent software
Executable	Allow	Everyone	%PROGRAMFILES%\Amazon*	Path	Required for the AppStream 2.0 agent software

Rule type	Action	Windows user or group	Name/Path	Condition	Description
Executable	Allow	Everyone	%PROGRAMFILES% \<*default-browser* > *	Path	Required for the AppStream 2.0 agent software when persistent storage solutions, such as Google Drive or Microsoft OneDrive for Business, are used. This exception is not required when AppStream 2.0 home folders are used.

Identity and Access Management for Amazon AppStream 2.0

Your security credentials identify you to services in AWS and grant you unlimited use of your AWS resources, such as your AppStream 2.0 resources. You can use features of AppStream 2.0 and AWS Identity and Access Management (IAM) to allow other users, services, and applications to use your AppStream 2.0 resources without sharing your security credentials.

You can use IAM to control how other users use resources in your Amazon Web Services account, and you can use security groups to control access to your AppStream 2.0 streaming instances. You can allow full use or limited use of your AppStream 2.0 resources.

Contents

- Network Access to Your Streaming Instance
- Using AWS Managed Policies and Linked Roles to Manage Administrator Access to AppStream 2.0 Resources
- Using IAM Policies to Manage Administrator Access to Application Auto Scaling
- Using IAM Policies to Manage Administrator Access to the Amazon S3 Bucket for Home Folders and Application Settings Persistence
- Using an IAM Role to Grant Permissions to Applications and Scripts Running on AppStream 2.0 Streaming Instances

Network Access to Your Streaming Instance

A security group acts as a stateful firewall that controls what traffic is allowed to reach your streaming instances. When you launch an AppStream 2.0 streaming instance, assign it to one or more security groups. Then, add rules to each security group that control traffic for the instance. You can modify the rules for a security group at any time. The new rules are automatically applied to all instances to which the security group is assigned.

For more information, see Security Groups in Amazon AppStream 2.0.

Using AWS Managed Policies and Linked Roles to Manage Administrator Access to AppStream 2.0 Resources

By default, IAM users don't have the permissions required to create or modify AppStream 2.0 resources, or perform tasks by using the AppStream 2.0 API. This means that these users can't perform these actions in the AppStream 2.0 console or by using AppStream 2.0 AWS CLI commands. To allow IAM users to create or modify resources and perform tasks, attach an IAM policy to the IAM users or groups that require those permissions.

When you attach a policy to a user, group of users, or IAM role, it allows or denies the users permission to perform the specified tasks on the specified resources.

Contents

- AWS Managed Policies Required to Access AppStream 2.0 Resources
- Roles Required for AppStream 2.0 , Application Auto Scaling, and AWS Certificate Manager Private CA
- Checking for the AmazonAppStreamServiceAccess Service Role and Policies
- Checking for the ApplicationAutoScalingForAmazonAppStreamAccess Service Role and Policies
- Checking for the AWSServiceRoleForApplicationAutoScaling_AppStreamFleet Service-Linked Role and Policies
- Checking for the AmazonAppStreamPCAAccess Service Role and Policies

AWS Managed Policies Required to Access AppStream 2.0 Resources

To provide full administrative or read-only access to AppStream 2.0, you must attach one of the following AWS managed policies to the IAM users or groups that require those permissions. An *AWS managed policy* is a standalone policy that is created and administered by AWS. For more information, see AWS Managed Policies in the *IAM User Guide*.

AmazonAppStreamFullAccess

This managed policy provides full administrative access to AppStream 2.0 resources. To manage AppStream 2.0 resources and perform API actions through the AWS Command Line Interface (AWS CLI), AWS SDK, or AWS Management Console, you must have the permissions defined in this policy.

If you sign into the AppStream 2.0 console as an IAM user, you must attach this policy to your AWS account. If you sign in through console federation, you must attach this policy to the IAM role that was used for federation.

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

```
                "application-autoscaling:DeleteScalingPolicy",
                "application-autoscaling:DescribeScalableTargets",
                "application-autoscaling:DescribeScalingPolicies",
                "application-autoscaling:PutScalingPolicy",
                "application-autoscaling:RegisterScalableTarget",
                "application-autoscaling:DescribeScheduledActions",
                "application-autoscaling:PutScheduledAction",
                "application-autoscaling:DeleteScheduledAction"
            ],
            "Effect": "Allow",
            "Resource": ""
        },
        {
           "Action": [
                "cloudwatch:DeleteAlarms",
                "cloudwatch:DescribeAlarms",
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:PutMetricAlarm"
            ],
            "Effect": "Allow",
            "Resource": ""
        },
        {
            "Action": [
                "ec2:DescribeRouteTables",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "ec2:DescribeVpcEndpoints"
            ],
            "Effect": "Allow",
            "Resource": ""
        },
        {
            "Action": "iam:ListRoles",
            "Effect": "Allow",
            "Resource": ""
       },
        {
            "Action": "iam:PassRole",
            "Effect": "Allow",
            "Resource": "arn:aws:iam:::role/service-role/
ApplicationAutoScalingForAmazonAppStreamAccess",
            "Condition": {
```

```
                "StringLike": {
                    "iam:PassedToService": "application-autoscaling.amazonaws.com"
                }
            }
        },
        {
            "Action": "iam:CreateServiceLinkedRole",
            "Effect": "Allow",
            "Resource": "arn:aws:iam:::role/aws-service-role/appstream.application-
autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_AppStreamFleet
 (http://appstream.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_AppStreamFleet)",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "appstream.application-
autoscaling.amazonaws.com"
                }
            }
        }
    ]
}
```

AmazonAppStreamReadOnlyAccess

This managed policy provides read-only access to AppStream 2.0 resources.

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

The AppStream 2.0 console uses two additional actions that provide functionality that is not available through the AWS CLI or AWS SDK. The **AmazonAppStreamFullAccess** and **AmazonAppStreamReadOnlyAccess** policies both provide permissions for these actions.

Action	Description	Access Level
`GetImageBuilders`	Grants permission to retrieve a list that describes one or more specified image builders, if the image builder names are provided. Otherwise, all image builders in the account are described.	Read
`GetParametersForThemeAssetUpload`	Grants permission to upload theme assets for custom branding. For more information, see Add Your Custom Branding to Amazon AppStream 2.0.	Write

AmazonAppStreamPCAAccess

This managed policy provides full administrative access to AWS Certificate Manager Private CA resources in your AWS account for certificate-based authentication.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "acm-pca:IssueCertificate",
                "acm-pca:GetCertificate",
```

```
                "acm-pca:DescribeCertificateAuthority"
            ],
            "Resource": "arn:*:acm-pca:*:*:*",
            "Condition": {
                "StringLike": {
                    "aws:ResourceTag/euc-private-ca": "*"
                }
            }
        }
    ]
}
```

AmazonAppStreamServiceAccess

This managed policy is the default policy for the AppStream 2.0 service role.

```
{
    "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",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeVpcEndpoints",
                "s3:ListAllMyBuckets",
                "ds:DescribeDirectories"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:CreateBucket",
                "s3:ListBucket",
```

```
                "s3:GetObject",
                "s3:PutObject",
                "s3:DeleteObject",
                "s3:GetObjectVersion",
                "s3:DeleteObjectVersion",
                "s3:GetBucketPolicy",
                "s3:PutBucketPolicy",
                "s3:PutEncryptionConfiguration"
            ],
            "Resource": [
                "arn:aws:s3:::appstream2-36fb080bb8-*",
                "arn:aws:s3:::appstream-app-settings-*",
                "arn:aws:s3:::appstream-logs-*"
            ]
        }
    ]
}
```

ApplicationAutoScalingForAmazonAppStreamAccess

This managed policy enables application autoscaling for AppStream 2.0.

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

```
    ]
}
```

AWSApplicationAutoscalingAppStreamFleetPolicy

This managed policy grants permissions for Application Auto Scaling to access AppStream 2.0 and CloudWatch .

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

AppStream 2.0 updates to AWS managed policies

View details about updates to AWS managed policies for AppStream 2.0 since this service began tracking these changes. For automatic alerts about changes to this page, subscribe to the RSS feed on the *Document History* page.

Change	Description	Date
AppStream 2.0 started tracking changes	AppStream 2.0 started tracking changes for its AWS managed policies	October 31, 2022

Roles Required for AppStream 2.0 , Application Auto Scaling, and AWS Certificate Manager Private CA

In AWS, IAM roles are used to grant permissions to an AWS service so it can access AWS resources. The policies that are attached to the role determine which AWS resources the service can access and what it can do with those resources. For AppStream 2.0, in addition to having the permissions defined in the **AmazonAppStreamFullAccess** policy, you must also have the following roles in your AWS account.

Roles

- AmazonAppStreamServiceAccess
- ApplicationAutoScalingForAmazonAppStreamAccess
- AWSServiceRoleForApplicationAutoScaling_AppStreamFleet
- AmazonAppStreamPCAAccess

AmazonAppStreamServiceAccess

This role is a service role that is created for you automatically when you get started with AppStream 2.0 in an AWS Region. For more information about services roles, see Creating a role to delegate permissions to an AWS service in the *IAM User Guide*.

While AppStream 2.0 resources are being created, the AppStream 2.0 service makes API calls to other AWS services on your behalf by assuming this role. To create fleets, you must have this role in your account. If this role is not in your AWS account and the required IAM permissions and trust relationship policies are not attached, you cannot create AppStream 2.0 fleets.

For more information, see Checking for the AmazonAppStreamServiceAccess Service Role and Policies.

ApplicationAutoScalingForAmazonAppStreamAccess

This role is a service role that is created for you automatically when you get started with AppStream 2.0 in an AWS Region. For more information about services roles, see Creating a role to delegate permissions to an AWS service in the *IAM User Guide*.

Automatic scaling is a feature of AppStream 2.0 fleets. To configure scaling policies, you must have this service role in your AWS account. If this service role is not in your AWS account and the required IAM permissions and trust relationship policies are not attached, you cannot scale AppStream 2.0 fleets.

For more information, see Checking for the ApplicationAutoScalingForAmazonAppStreamAccess Service Role and Policies.

AWSServiceRoleForApplicationAutoScaling_AppStreamFleet

This role is a service-linked role that is created for you automatically. For more information, see Service-linked roles in the *Application Auto Scaling User Guide*.

Application Auto Scaling uses a service-linked role to perform automatic scaling on your behalf. A *service-linked role* is an IAM role that is linked directly to an AWS service. This role includes all the permissions that the service requires to call other AWS services on your behalf.

For more information, see Checking for the `AWSServiceRoleForApplicationAutoScaling_AppStreamFleet` Service-Linked Role and Policies.

AmazonAppStreamPCAAccess

This role is a service role that is created for you automatically when you get started with AppStream 2.0 in an AWS Region. For more information about services roles, see Creating a role to delegate permissions to an AWS service in the *IAM User Guide*.

Certificate-based authentication is a feature of AppStream 2.0 fleets joined to Microsoft Active Directory domains. To enable and use certificate-based authentication, you must have this service role in your AWS account. If this service role is not in your AWS account and the required IAM permissions and trust relationship policies are not attached, you cannot enable or use certificate-based authentication.

For more information, see the section called “Checking for the AmazonAppStreamPCAAccess Service Role and Policies”.

Checking for the AmazonAppStreamServiceAccess Service Role and Policies

Complete the steps in this section to check whether the **AmazonAppStreamServiceAccess** service role is present and has the correct policies attached. If this role is not in your account and must be created, you or an administrator with the required permissions must perform the steps to get started with AppStream 2.0 in your Amazon Web Services account.

To check whether the AmazonAppStreamServiceAccess IAM service role is present

1. Open the IAM console at https://console.aws.amazon.com/iam/.

2. In the navigation pane, choose **Roles**.
3. In the search box, type **amazonappstreamservice** to narrow the list of roles to select, and then choose **AmazonAppStreamServiceAccess**. If this role is listed, select it to view the role **Summary** page.
4. On the **Permissions** tab, confirm whether the **AmazonAppStreamServiceAccess** permissions policy is attached.
5. Return to the role **Summary** page.
6. On the **Trust relationships** tab, choose **Show policy document**, and then confirm whether the **AmazonAppStreamServiceAccess** trust relationship policy is attached and follows the correct format. If so, the trust relationship is correctly configured. Choose **Cancel** and close the IAM console.

AmazonAppStreamServiceAccess trust relationship policy

The **AmazonAppStreamServiceAccess** trust relationship policy must include the AppStream 2.0 service as the principal. A *principal* is an entity in AWS that can perform actions and access resources. This policy must also include the `sts:AssumeRole` action. The following policy configuration defines AppStream 2.0 as a trusted entity.

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

Checking for the ApplicationAutoScalingForAmazonAppStreamAccess Service Role and Policies

Complete the steps in this section to check whether the **ApplicationAutoScalingForAmazonAppStreamAccess** service role is present and has the correct policies attached. If this role is not in your account and must be created, you or an administrator

with the required permissions must perform the steps to get started with AppStream 2.0 in your Amazon Web Services account.

To check whether the ApplicationAutoScalingForAmazonAppStreamAccess IAM service role is present

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Roles**.
3. In the search box, type **applicationautoscaling** to narrow the list of roles to select, and then choose **ApplicationAutoScalingForAmazonAppStreamAccess**. If this role is listed, select it to view the role **Summary** page.
4. On the **Permissions** tab, confirm whether the **ApplicationAutoScalingForAmazonAppStreamAccess** permissions policy is attached.
5. Return to the role **Summary** page.
6. On the **Trust relationships** tab, choose **Show policy document**, and then confirm whether the **ApplicationAutoScalingForAmazonAppStreamAccess** trust relationship policy is attached and follows the correct format. If so, the trust relationship is correctly configured. Choose **Cancel** and close the IAM console.

ApplicationAutoScalingForAmazonAppStreamAccess trust relationship policy

The **ApplicationAutoScalingForAmazonAppStreamAccess** trust relationship policy must include the Application Auto Scaling service as the principal. This policy must also include the `sts:AssumeRole` action. The following policy configuration defines Application Auto Scaling as a trusted entity.

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

Checking for the `AWSServiceRoleForApplicationAutoScaling_AppStreamFleet` Service-Linked Role and Policies

Complete the steps in this section to check whether the `AWSServiceRoleForApplicationAutoScaling_AppStreamFleet` service-linked role is present and has the correct policies attached. If this role is not in your account and must be created, you or an administrator with the required permissions must perform the steps to get started with AppStream 2.0 in your Amazon Web Services account.

To check whether the `AWSServiceRoleForApplicationAutoScaling_AppStreamFleet` IAM service-linked role is present

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Roles**.
3. In the search box, type **applicationautoscaling** to narrow the list of roles to select, and then choose `AWSServiceRoleForApplicationAutoScaling_AppStreamFleet`. If this role is listed, select it to view the role **Summary**page.
4. On the **Permissions** tab, confirm whether the `AWSApplicationAutoscalingAppStreamFleetPolicy` permissions policy is attached.
5. Return to the **Role** summary page.
6. On the **Trust relationships** tab, choose **Show policy document**, and then confirm whether the `AWSServiceRoleForApplicationAutoScaling_AppStreamFleet` trust relationship policy is attached and follows the correct format. If so, the trust relationship is correctly configured. Choose **Cancel** and close the IAM console.

AWSServiceRoleForApplicationAutoScaling_AppStreamFleet trust relationship policy

The `AWSServiceRoleForApplicationAutoScaling_AppStreamFleet` trust relationship policy must include **appstream.application-autoscaling.amazonaws.com** as the principal. This policy must also include the `sts:AssumeRole` action. The following policy configuration defines **appstream.application-autoscaling.amazonaws.com** as a trusted entity.

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

```
      "Effect": "Allow",
      "Principal": {
        "Service": "appstream.application-autoscaling.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Checking for the AmazonAppStreamPCAAccess Service Role and Policies

Complete the steps in this section to check whether the **AmazonAppStreamPCAAccess** service role is present and has the correct policies attached. If this role is not in your account and must be created, you or an administrator with the required permissions must perform the steps to get started with AppStream 2.0 in your Amazon Web Services account.

To check whether the AmazonAppStreamPCAAccess IAM service role is present

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Roles**.
3. In the search box, type **appstreampca** to narrow the list of roles to select, and then choose **AmazonAppStreamPCAAccess**. If this role is listed, select it to view the role **Summary**page.
4. On the **Permissions** tab, confirm whether the **AmazonAppStreamPCAAccess** permissions policy is attached.
5. Return to the **Role** summary page.
6. On the **Trust relationships** tab, choose **Show policy document**, and then confirm whether the **AmazonAppStreamPCAAccess** trust relationship policy is attached and follows the correct format. If so, the trust relationship is correctly configured. Choose **Cancel** and close the IAM console.

AmazonAppStreamPCAAccess trust relationship policy

The **AmazonAppStreamPCAAccess** trust relationship policy must include prod.euc.ecm.amazonaws.com as the principal. This policy must also include the `sts:AssumeRole` action. The following policy configuration defines ECM as a trusted entity.

To create the AmazonAppStreamPCAAccess trust relationship policy using the AWS CLI

1. Create a JSON file named `AmazonAppStreamPCAAccess.json` with the following text.

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

2. Adjust the `AmazonAppStreamPCAAccess.json` path as needed and run the following AWS CLI commands to create the trust relationship policy and attach the AmazonAppStreamPCAAccess managed policy. For more information about the managed policy, see the section called "AWS Managed Policies Required to Access AppStream 2.0 Resources".

```
aws iam create-role --path /service-role/ --role-name AmazonAppStreamPCAAccess --
assume-role-policy-document file://AmazonAppStreamPCAAccess.json
```

```
aws iam attach-role-policy –role-name AmazonAppStreamPCAAccess –policy-arn
 arn:aws:iam::aws:policy/AmazonAppStreamPCAAccess
```

Using IAM Policies to Manage Administrator Access to Application Auto Scaling

Automatic scaling for fleets is made possible by a combination of the AppStream 2.0, Amazon CloudWatch, and Application Auto Scaling APIs. AppStream 2.0 fleets are created with AppStream 2.0, alarms are created with CloudWatch, and scaling policies are created with Application Auto Scaling.

In addition to having the permissions defined in the AmazonAppStreamFullAccess policy, the IAM user that accesses fleet scaling settings must have the required permissions for the services

that support dynamic scaling. IAM users must have permissions to use the actions shown in the following example policy.

```
{
  "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:ListRoles"
      ],
      "Resource": "*"
    },
    {
      "Sid": "iamPassRole",
      "Effect": "Allow",
      "Action": [
          "iam:PassRole"
      ],
      "Resource": "*",
      "Condition": {
         "StringEquals": {
             "iam:PassedToService": "application-autoscaling.amazonaws.com"
          }
      }
    }
  ]
}
```

You can also create your own IAM policies to set more specific permissions for calls to the Application Auto Scaling API. For more information, see Authentication and Access Control in the *Application Auto Scaling User Guide*.

Using IAM Policies to Manage Administrator Access to the Amazon S3 Bucket for Home Folders and Application Settings Persistence

The following examples show how you can use IAM policies to manage access to the Amazon S3 bucket for home folders and application settings persistence.

Examples

- Deleting the Amazon S3 Bucket for Home Folders and Application Settings Persistence
- Restricting Administrator Access to the Amazon S3 Bucket for Home Folders and Application Settings Persistence

Deleting the Amazon S3 Bucket for Home Folders and Application Settings Persistence

AppStream 2.0 adds an Amazon S3 bucket policy to the buckets that it creates to prevent them from being accidentally deleted. To delete an S3 bucket, you must first delete the S3 bucket policy. Following are the bucket policies that you must delete for home folders and application settings persistence.

Home folders policy

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

Application settings persistence policy

```
{
```

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

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

Restricting Administrator Access to the Amazon S3 Bucket for Home Folders and Application Settings Persistence

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

```
{
  "Sid": "RestrictedAccess",
  "Effect": "Deny",
  "NotPrincipal":
  {
    "AWS": [
      "arn:aws:iam::account:role/service-role/AmazonAppStreamServiceAccess",
      "arn:aws:sts::account:assumed-role/AmazonAppStreamServiceAccess/PhotonSession",
      "arn:aws:iam::account:user/IAM-user-name"
    ]
  },
    "Action": "s3:*",
    "Resource": "arn:aws:s3:::home-folder-or-application-settings-persistence-s3-
bucket-region-account"
  }
 ]
}
```

This policy allows S3 bucket access only 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:user/IAM-user-name"
```

In the following example, the policy restricts access to the home folder S3 bucket for anyone other than IAM users marymajor and johnstiles. It also allows access to the AppStream 2.0 service, in AWS Region US West (Oregon) 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/marymajor",
      "arn:aws:iam::123456789012:user/johnstiles"
    ]
  },
    "Action": "s3:*",
    "Resource": "arn:aws:s3:::appstream2-36fb080bb8-us-west-2-123456789012"
  }
 ]
}
```

Using an IAM Role to Grant Permissions to Applications and Scripts Running on AppStream 2.0 Streaming Instances

Applications and scripts that run on AppStream 2.0 streaming instances must include AWS credentials in their AWS API requests. You can create an IAM role to manage these credentials. An IAM role specifies a set of permissions that you can use to access AWS resources. This role is not uniquely associated with one person, however. Instead, it can be assumed by anyone that needs it.

You can apply an IAM role to an AppStream 2.0 streaming instance. When the streaming instance switches to (assumes) the role, the role provides temporary security credentials. Your application or scripts use these credentials to perform API actions and management tasks on the streaming instance. AppStream 2.0 manages the temporary credential switch for you.

Contents

- Best Practices for Using IAM Roles With AppStream 2.0 Streaming Instances
- Configuring an Existing IAM Role to Use With AppStream 2.0 Streaming Instances
- How to Create an IAM Role to Use With AppStream 2.0 Streaming Instances
- How to Use the IAM Role With AppStream 2.0 Streaming Instances

Best Practices for Using IAM Roles With AppStream 2.0 Streaming Instances

When you use IAM roles with AppStream 2.0 streaming instances, we recommend that you follow these practices:

- Limit the permissions that you grant to AWS API actions and resources.

  Follow least privilege principles when you create and attach IAM policies to the IAM roles associated with AppStream 2.0 streaming instances. When you use an application or script that requires access to AWS API actions or resources, determine the specific actions and resources that are required. Then, create policies that allow the application or script to perform only those actions. For more information, see Grant Least Privilege in the *IAM User Guide*.
- Create an IAM role for each AppStream 2.0 resource.

  Creating a unique IAM role for each AppStream 2.0 resource is a practice that follows least privilege principles. Doing so also lets you modify permissions for a resource without affecting other resources.
- Limit where the credentials can be used.

  IAM policies let you define the conditions under which your IAM role can be used to access a resource. For example, you can include conditions to specify a range of IP addresses that requests can come from. Doing so prevents the credentials from being used outside of your environment. For more information, see Use Policy Conditions for Extra Security in the *IAM User Guide*.

Configuring an Existing IAM Role to Use With AppStream 2.0 Streaming Instances

This topic describes how to configure an existing IAM role so that you can use it with image builders and fleet streaming instances.

Prerequisites

The IAM role that you want to use with an AppStream 2.0 image builder or fleet streaming instance must meet the following prerequisites:

- The IAM role must be in the same Amazon Web Services account as the AppStream 2.0 streaming instance.
- The IAM role cannot be a service role.
- The trust relationship policy that is attached to the IAM role must include the AppStream 2.0 service as the principal. A *principal* is an entity in AWS that can perform actions and access resources. The policy must also include the `sts:AssumeRole` action. This policy configuration defines AppStream 2.0 as a trusted entity.

- If you are applying the IAM role to an image builder, the image builder must run a version of the AppStream 2.0 agent released on or after September 3, 2019. If you are applying the IAM role to a fleet, the fleet must use an image use an image that uses a version of the agent released on or after the same date. For more information, see AppStream 2.0 Agent Release Notes.

To enable the AppStream 2.0 service principal to assume an existing IAM role

To perform the following steps, you must sign into the account as an IAM user who has the permissions required to list and update IAM roles. If you don't have the required permissions, ask your Amazon Web Services account administrator either to perform these steps in your account or to grant you the required permissions.

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Roles**.
3. In the list of roles in your account, choose the name of the role that you want to modify.
4. Choose the **Trust relationships** tab, and then choose **Edit trust relationship**.
5. Under **Policy Document**, verify that the trust relationship policy includes the `sts:AssumeRole` action for the `appstream.amazonaws.com` service principal:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
```

```
            "appstream.amazonaws.com"
          ]
        },
        "Action": "sts:AssumeRole"
      }
    ]
}
```

6. When you are finished editing your trust policy, choose **Update Trust Policy** to save your changes.
7. The IAM role that you selected will display in the AppStream 2.0 console. This role grants permissions to applications and scripts to perform API actions and management tasks on streaming instances.

How to Create an IAM Role to Use With AppStream 2.0 Streaming Instances

This topic describes how to create a new IAM role so that you can use it with image builders and fleet streaming instances.

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Roles**, and then choose **Create role**.
3. For **Select type of trusted entity**, choose **AWS service**.
4. From the list of AWS services, choose **AppStream 2.0**.
5. Under **Select your use case**, **AppStream 2.0 — Allows AppStream 2.0 instances to call AWS services on your behalf** is already selected. Choose **Next: Permissions**.
6. If possible, select the policy to use for the permissions policy or choose **Create policy** to open a new browser tab and create a new policy from scratch. For more information, see step 4 in the procedure Creating IAM Policies (Console) in the *IAM User Guide*.

   After you create the policy, close that tab and return to your original tab. Select the check box next to the permissions policies that you want AppStream 2.0 to have.
7. (Optional) Set a permissions boundary. This is an advanced feature that is available for service roles, but not service-linked roles. For more information, see Permissions Boundaries for IAM Entities in the *IAM User Guide*.
8. Choose **Next: Tags**. You can optionally attach tags as key-value pairs. For more information, see Tagging IAM Users and Roles in the *IAM User Guide*.
9. Choose **Next: Review**.

10. For **Role name**, type a role name that is unique within your Amazon Web Services account. Because other AWS resources might reference the role, you can't edit the name of the role after it has been created.
11. For **Role description**, keep the default role description or type a new one.
12. Review the role, and then choose **Create role**.

How to Use the IAM Role With AppStream 2.0 Streaming Instances

After you create an IAM role, you can apply it to an image builder or fleet streaming instance when you launch the image builder or create a fleet. You can also apply an IAM role to existing fleets. For information about how to apply IAM role when you launch an image builder, see Launch an Image Builder to Install and Configure Streaming Applications. For information about how to apply IAM role when you create a fleet, see Create a Fleet.

When you apply an IAM role to your image builder or fleet streaming instance, AppStream 2.0 retrieves temporary credentials and creates the **appstream_machine_role** credential profile on the instance. The temporary credentials are valid for 1 hour, and new credentials retrieved every hour. The previous credentials do not expire, so you can use them for as long as they are valid. You can use the credential profile to call AWS services programmatically by using the AWS Command Line Interface (AWS CLI), AWS Tools for PowerShell, or the AWS SDK with the language of your choice.

When you make the API calls, specify **appstream_machine_role** as the credential profile. Otherwise, the operation fails due to insufficient permissions.

AppStream 2.0 assumes the specified role while the streaming instance is provisioned. Because AppStream 2.0 uses the elastic network interface that is attached to your VPC for AWS API calls, your application or script must wait for the elastic network interface to become available before making AWS API calls. If API calls are made before the elastic network interface is available, the calls fail.

The following examples show how you can use the **appstream_machine_role** credential profile to describe streaming instances (EC2 instances) and to create the Boto client. Boto is the Amazon Web Services (AWS) SDK for Python.

Describe Streaming Instances (EC2 instances) by Using the AWS CLI

```
aws ec2 describe-instances --region us-east-1 --profile appstream_machine_role
```

Describe Streaming Instances (EC2 instances) by Using AWS Tools for PowerShell

You must use AWS Tools for PowerShell version 3.3.563.1 or later, with the Amazon Web Services SDK for .NET version 3.3.103.22 or later. You can download the AWS Tools for Windows installer, which includes AWS Tools for PowerShell and the Amazon Web Services SDK for .NET, from the AWS Tools for PowerShell website.

```
Get-EC2Instance -Region us-east-1 -ProfileName appstream_machine_role
```

Creating the Boto Client by Using the AWS SDK for Python

```
session = boto3.Session(profile_name='appstream_machine_role')
```

Logging and Monitoring in Amazon AppStream 2.0

Monitoring is an important part of maintaining the reliability, availability, and performance of Amazon AppStream 2.0. This topic describes the services and tools that AWS provides for monitoring your AppStream 2.0 resources and responding to potential incidents.

Amazon CloudWatch Alarms

Amazon CloudWatch alarms let you watch a single metric over a time period that you specify. If the metric exceeds a given threshold, a notification is sent to an Amazon Simple Notification Service topic or AWS Auto Scaling policy. CloudWatch alarms do not invoke actions that are in a particular state. Instead, the state must have changed and been maintained for a specified number of periods. For more information, see Monitoring Amazon AppStream 2.0 Resources.

> **Note**
>
> AppStream 2.0 currently can't be configured as a target for CloudWatch Events. For a list of services that you can configure as targets for CloudWatch events, see What Is Amazon CloudWatch Events.

AWS CloudTrail

AWS CloudTrail provides a record of actions taken by a user, role, or an AWS service in AppStream 2.0. This record lets you determine the request that was made to AppStream 2.0, the IP address from which the request was made, who made the request, when it was made, and additional details. For more information, see Logging AppStream 2.0 API Calls with AWS CloudTrail.

AWS Trusted Advisor

AWS Trusted Advisor inspects your AWS environment and then recommends ways to save money, improve system availability and performance, or help close security gaps. Trusted Advisor uses best practices collected from a wide variety of AWS customers. All AWS customers have access to five Trusted Advisor checks. If you have a Business or Enterprise support plan, you can view all Trusted Advisor checks.

When you enable application settings persistence or home folders for your users, the data that is generated by your users is stored in Amazon S3 buckets. Trusted Advisor contains the following checks related to Amazon S3:

- Logging configuration of Amazon S3 buckets.
- Security checks for Amazon S3 buckets that have open access permissions.
- Fault tolerance checks for Amazon S3 buckets that don't have versioning enabled, or have versioning suspended.

For more information, see AWS Trusted Advisor in the *AWS Support User Guide*.

Amazon S3 Access Logs

If your users have application settings data or home folders data stored in Amazon S3 buckets, consider viewing Amazon S3 server access logs to monitor access. These logs provide detailed records about requests that are made to a bucket. Server access logs are useful for many applications. For example, access log information can be useful in security and access audits. For more information, see Amazon S3 Server Access Logging in the *Amazon Simple Storage Service User Guide*.

AppStream 2.0 Usage Reports

You can subscribe to AppStream 2.0 usage reports to receive detailed reports about how your users are using the service. The reports include how long users stream and which applications they launch. For more information, see AppStream 2.0 Usage Reports.

Compliance Validation for Amazon AppStream 2.0

Third-party auditors assess the security and compliance of Amazon AppStream 2.0 as part of multiple AWS compliance programs. These include the following: SOC, PCI, ISO, FedRAMP, HIPAA, MTCS, ENS High, HITRUST CSF, VPAT, and others.

> **Note**
>
> AppStream 2.0 supports FIPS 140-2. For information about how to use AppStream 2.0 FIPS endpoints for administrative use or streaming, see the section called "FIPS Endpoints". AppStream 2.0 is also undergoing assessment for the Department of Defense (DoD) Cloud Computing Security Requirements Guide (SRG).

For a list of AWS services in scope of specific compliance programs, see AWS Services in Scope by Compliance Program. For general information, see AWS Compliance Programs.

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

Your compliance responsibility when using AppStream 2.0 is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. AWS provides the following resources to help with compliance:

- Security and Compliance Quick Start Guides – These deployment guides discuss architectural considerations and provide steps for deploying security- and compliance-focused baseline environments on AWS.
- Architecting for HIPAA Security and Compliance Whitepaper – This whitepaper describes how companies can use AWS to create HIPAA-compliant applications.
- AWS Compliance Resources – This collection of workbooks and guides might apply to your industry and location.
- Evaluating Resources with Rules in the *AWS Config Developer Guide* – The AWS Config service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- AWS Security Hub – This AWS service provides a comprehensive view of your security state within AWS that helps you check your compliance with security industry standards and best practices.

Resilience in Amazon AppStream 2.0

The AWS global infrastructure is built around AWS Regions and Availability Zones. Regions provide multiple physically separated and isolated Availability Zones, which are connected through low-latency, high-throughput, and highly redundant networking. With Availability Zones, you

can design and operate applications and databases that automatically fail over between zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

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

Infrastructure Security in Amazon AppStream 2.0

As a managed service, Amazon AppStream 2.0 is protected by AWS global network security. For information about AWS security services and how AWS protects infrastructure, see AWS Cloud Security. To design your AWS environment using the best practices for infrastructure security, see Infrastructure Protection in *Security Pillar AWS Well-Architected Framework*.

You use AWS published API calls to access AppStream 2.0 through the network. Clients must support the following:

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

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

The following topics provide additional information about AppStream 2.0 infrastructure security.

Contents

- Network Isolation
- Isolation on Physical Hosts
- Controlling Network Traffic
- AppStream 2.0 Interface VPC Endpoints
- Protecting Data in Transit with FIPS Endpoints

Network Isolation

A virtual private cloud (VPC) is a virtual network in your own logically isolated area in the Amazon Web Services Cloud. Use separate VPCs to isolate infrastructure by workload or organizational entity.

A subnet is a range of IP addresses in a VPC. When you launch an instance, you launch it into a subnet in your VPC. Use subnets to isolate the tiers of your application (for example, web, application, and database) within a single VPC. Use private subnets for your instances if they should not be accessed directly from the internet.

You can stream from AppStream 2.0 streaming instances in your VPC without going through the public internet. To do so, use an interface VPC endpoint (interface endpoint). For more information, see Creating and Streaming from Interface VPC Endpoints.

You can also call AppStream 2.0 API operations from your VPC without sending traffic over the public internet by using an interface endpoint. For information, see Access AppStream 2.0 API Operations and CLI Commands Through an Interface VPC Endpoint.

Isolation on Physical Hosts

Different streaming instances on the same physical host are isolated from each other as though they are on separate physical hosts. The hypervisor isolates CPU and memory, and the instances are provided virtualized disks instead of access to the raw disk devices.

When you stop or terminate a streaming instance, the memory allocated to it is scrubbed (meaning, it's set to zero) by the hypervisor before it is allocated to a new instance, and every block of storage is reset. This ensures that your data is not exposed to another instance.

Controlling Network Traffic

To help control network traffic to your AppStream 2.0 streaming instances, consider these options:

- When you launch an Amazon AppStream streaming instance, you launch it into a subnet in your VPC. You can deploy streaming instances in a private subnet if they should not be accessible from the internet.
- To provide internet access to your streaming instances in a private subnet, use a NAT gateway. For more information, see Configure a VPC with Private Subnets and a NAT Gateway.
- Security groups that belong to your VPC let you control the network traffic between AppStream 2.0 streaming instances and VPC resources such as license servers, file servers, and database

servers. Security groups also isolate traffic between your streaming instances and AppStream 2.0 management services.

Use security groups to restrict access to your streaming instances. For example, you can allow traffic only from the address ranges for your corporate network. For more information, see Security Groups in Amazon AppStream 2.0.

- You can stream from AppStream 2.0 streaming instances in your VPC without going through the public internet. To do so, use an interface VPC endpoint (interface endpoint). For more information, see Creating and Streaming from Interface VPC Endpoints.

  You can also call AppStream 2.0 API operations from your VPC without sending traffic over the public internet by using an interface endpoint. For more information, see Access AppStream 2.0 API Operations and CLI Commands Through an Interface VPC Endpoint.
- Use IAM roles and policies to manage administrator access to AppStream 2.0, Application Auto Scaling, and Amazon S3 buckets. For more information, see the following topics:
  - Using AWS Managed Policies and Linked Roles to Manage Administrator Access to AppStream 2.0 Resources
  - Using IAM Policies to Manage Administrator Access to Application Auto Scaling
  - Restricting Administrator Access to the Amazon S3 Bucket for Home Folders and Application Settings Persistence
- You can use SAML 2.0 to federate authentication to AppStream 2.0. For more information, see Amazon AppStream 2.0 Service Quotas.

> **Note**
>
> For smaller AppStream 2.0 deployments, you can use AppStream 2.0 user pools. By default, user pools support a maximum of 50 users. For more information about AppStream 2.0 quotas (also referred to as limits), see Amazon AppStream 2.0 Service Quotas. For deployments that must support 100 or more AppStream 2.0 users, we recommend using SAML 2.0.

AppStream 2.0 Interface VPC Endpoints

A virtual private cloud (VPC) is a virtual network in your own logically isolated area in the Amazon Web Services Cloud. If you use Amazon Virtual Private Cloud to host your AWS resources, you can

establish a private connection between your VPC and AppStream 2.0. You can use this connection to enable AppStream 2.0 to communicate with your resources on your VPC without going through the public internet.

Interface endpoints are powered by AWS PrivateLink, a technology that lets you keep streaming traffic within a VPC that you specify by using private IP addresses. When you use the VPC with an AWS Direct Connect or AWS Virtual Private Network tunnel, you can keep the streaming traffic within your network.

The following topics provide information about AppStream 2.0 interface endpoints.

Contents

- Creating and Streaming from Interface VPC Endpoints
- Access AppStream 2.0 API Operations and CLI Commands Through an Interface VPC Endpoint

Creating and Streaming from Interface VPC Endpoints

You can use an interface VPC endpoint in your Amazon Web Services account to restrict all network traffic between your Amazon VPC and AppStream 2.0 to the Amazon network. After you create this endpoint, you configure your AppStream 2.0 stack or image builder to use it.

Prerequisites

Before you set up interface VPC endpoints for AppStream 2.0, be aware of the following prerequisites:

- Internet connectivity is required to authenticate users and deliver the web assets that AppStream 2.0 requires to function. The streaming interface endpoint maintains the streaming traffic within your VPC. Streaming traffic includes pixel, USB, user input, audio, clipboard, file upload and download, and printer traffic. To allow this traffic, you must allow the domains listed in Allowed Domains.
- The network to which your users' devices are connected must be able to route traffic to the interface endpoint.
- The security groups that are associated with the interface endpoint must allow inbound access to port 443 (TCP) and ports 1400-1499 (TCP) from the IP address range from which your users connect.
- The network access control list for the subnets must allow outbound traffic from ephemeral network ports 1024-65535 (TCP) to the IP address range from which your users connect.

- You must have an IAM permissions policy in your AWS account that provides permissions to perform the `ec2:DescribeVpcEndpoints` API action. By default, this permission is defined in the IAM policy that is attached to the AmazonAppStreamServiceAccess role. If you have the required permissions, this service role is automatically created by AppStream 2.0, with the required IAM policies attached, when you get started with the AppStream 2.0 service in an AWS Region. For more information, see Identity and Access Management for Amazon AppStream 2.0.

To create an interface endpoint

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Endpoints**, **Create Endpoint**.
3. Choose **Create Endpoint**.
4. For **Service category**, ensure that **AWS services** is selected.
5. For **Service Name**, choose **`com.amazonaws.`***`<AWS Region>`***`.appstream.streaming`**.
6. Specify the following information. When you're done, choose **Create endpoint**.

   - For **VPC**, choose a VPC in which to create the interface endpoint. You can choose a different VPC than the VPC with AppStream 2.0 resources.
   - For **Subnets**, choose the subnets (Availability Zones) in which to create the endpoint network interfaces. We recommend that you choose subnets in at least two Availability Zones.
   - Ensure that the **Enable Private DNS Name** check box is selected.

     > **Note**
     >
     > If your users use a network proxy to access streaming instances, disable any proxy caching on the domain and DNS names that are associated with the private endpoint.

   - For **Security group**, choose the security groups to associate with the endpoint network interfaces.

     > **Note**
     >
     > The security groups must provide inbound access to the ports from the IP address range from which your users connect.

While your interface endpoint is being created, the status of the endpoint in the console appears as **Pending**. After your endpoint is created, the status changes to **Available.**

To update a stack to use the interface endpoint that you created for streaming sessions, perform the following steps.

To update a stack to use a new interface endpoint

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.

   Ensure that you open the console in the same AWS Region as the interface endpoint that you want to use.
2. In the navigation pane, choose **Stacks**, and then choose the stack that you want.
3. Choose the **VPC Endpoints** tab, and then choose **Edit**.
4. In the **Edit VPC Endpoint** dialog box, for **Streaming Endpoint**, choose the endpoint through which to stream traffic.
5. Choose **Update**.

Traffic for new streaming sessions will be routed through this endpoint. However, traffic for current streaming sessions continues to be routed through the previously specified endpoint.

> **Note**
>
> Users cannot stream using the internet endpoint when an interface endpoint is specified.

Access AppStream 2.0 API Operations and CLI Commands Through an Interface VPC Endpoint

If you use Amazon Virtual Private Cloud to host your AWS resources, you can connect directly to AppStream 2.0 API operations or command line interface (CLI) commands through an interface VPC endpoint (interface endpoint) in your virtual private cloud (VPC) instead of connecting over the internet. Interface endpoints are powered by AWS PrivateLink, a technology that lets you keep streaming traffic within a VPC that you specify by using private IP addresses. When you use an interface endpoint, communication between your VPC and AppStream 2.0 is conducted entirely and securely within the AWS network.

> **Note**
>
> This topic describes how to access the AppStream 2.0 API operations and CLI commands through an interface endpoint. For information about how to create and stream from AppStream 2.0 interface endpoints, see Creating and Streaming from Interface VPC Endpoints.

Prerequisites

To use interface endpoints, you must meet the following prerequisites:

- The security groups that are associated with the interface endpoint must allow inbound access to port 443 (TCP) from the IP address range from which your users connect.
- The network access control list for the subnets must allow outbound traffic from ephemeral network ports 1024-65535 (TCP) to the IP address range from which your users connect.

Create an Interface Endpoint to Access AppStream 2.0 API Operations and CLI Commands

Perform the following steps to create an interface endpoint.

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Endpoints**, **Create Endpoint**.
3. Choose **Create Endpoint**.
4. For **Service category**, ensure that **AWS services** is selected.
5. For **Service Name**, choose **`com.amazonaws.`***`<AWS Region>`***`.appstream.api`**.
6. Specify the following information. When you're done, choose **Create endpoint**.
   - For **VPC**, select a VPC in which to create the interface endpoint.
   - For **Subnets**, select the subnets (Availability Zones) in which to create the endpoint network interfaces. We recommend that you choose subnets in at least two Availability Zones.
   - Optionally, you can select the **Enable Private DNS Name** check box.

> **Note**
>
> If you select this option, ensure that you configure VPC and DNS as needed to support private DNS. For more information, see Private DNS in the *Amazon VPC User Guide*.

- For **Security group**, select the security groups to associate with the endpoint network interfaces.

> **Note**
>
> The security groups must provide inbound access to the ports from the IP address range from which your users connect.

While your interface endpoint is being created, the status of the endpoint in the console appears as **Pending**. After your endpoint is created, the status changes to **Available.**

Use an Interface Endpoint to Access AppStream 2.0 API Operations and CLI Commands

After the status of the interface VPC endpoint that you create changes to **Available**, you can use the endpoint to access AppStream 2.0 API operations and CLI commands. To do so, specify the `endpoint-url` parameter with the DNS name of the interface endpoint when you use these operations and commands. The DNS name is publicly resolvable, but it only successfully routes traffic in your VPC.

The following example shows how to specify the DNS name of the interface endpoint when you use the **describe-fleets** CLI command:

```
aws appstream describe-fleets --endpoint-url <vpc-endpoint-id>.api.appstream.<aws-
region>.vpce.amazonaws.com
```

The following example shows how to specify the DNS name of the interface endpoint when you instantiate the AppStream 2.0 Boto3 Python client:

```
appstream2client = boto3.client('appstream',region_name='<aws-
region>',endpoint_url='<vpc-endpoint-id>.api.appstream.<aws-region>.vpce.amazonaws.com'
```

Subsequent commands using the `appstream2client` object automatically use the interface endpoint that you specified.

If you enabled the private DNS host names on the interface endpoint, you don't need to specify the endpoint URL. The AppStream 2.0 API DNS host name that the API and CLI use by default resolves within your VPC. For more information about private DNS host names, see Private DNS in the *Amazon VPC User Guide*.

Protecting Data in Transit with FIPS Endpoints

By default, when you communicate with the AppStream 2.0 service, whether as an administrator using the AppStream 2.0 console, the AWS Command Line Interface (AWS CLI), or an AWS SDK, or as a user streaming from an image builder or a fleet instance, all data in transit is encrypted using TLS 1.2.

If you require FIPS 140-2 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. AppStream 2.0 offers FIPS endpoints in all United States AWS Regions where AppStream 2.0 is available. When you use a FIPS endpoint, all data in transit is encrypted using cryptographic standards that comply with Federal Information Processing Standard (FIPS) 140-2. For information about FIPS endpoints, including a list of AppStream 2.0 endpoints, see Federal Information Processing Standard (FIPS) 140-2.

FIPS Endpoints for Administrative Use

To specify a FIPS endpoint when you run an AWS CLI command for AppStream 2.0, use the `endpoint-url` parameter. The following example uses the AppStream 2.0 FIPS endpoint in the US West (Oregon) Region to retrieve a list of all stacks in the Region:

```
aws appstream describe-stacks --endpoint-url https://appstream2-fips.us-
west-2.amazonaws.com
```

To specify a FIPS endpoint for AppStream 2.0 API operations, use the procedure in your AWS SDK for specifying a custom endpoint.

FIPS Endpoints for User Streaming Sessions

If you use SAML 2.0 or a streaming URL to authenticate users, you can configure FIPS-compliant connections for your users' streaming sessions.

To use a FIPS-compliant connection for users who authenticate using SAML 2.0, specify an AppStream 2.0 FIPS endpoint when you configure the relay state of your federation. For more information about constructing a relay state URL for identity federation using SAML 2.0, see Setting Up SAML.

To configure a FIPS-compliant connection for users who authenticate through a streaming URL, specify an AppStream 2.0 FIPS endpoint when you call the CreateStreamingURL or CreateImageBuilderStreamingURL operation from the AWS CLI or an AWS SDK. A user who connects to a streaming instance using the resulting URL is connected through a FIPS-compliant connection. The following example uses the AppStream 2.0 FIPS endpoint in the US East (Virginia) Region to generate a FIPS-compliant streaming URL:

```
aws appstream create-streaming-url --stack-name stack-name --fleet-name fleet-name --
user-id user-id --endpoint-url https://appstream2-fips.us-east-1.amazonaws.com
```

Exceptions

FIPS-compliant connections are not supported in the following scenarios:

- Administration of AppStream 2.0 through the AppStream 2.0 console
- Streaming sessions for users who authenticate using the AppStream 2.0 user pool feature
- Streaming using an interface VPC endpoint
- Generating FIPS-compliant streaming URLs through the AppStream 2.0 console
- Connections to your Google Drive or OneDrive storage accounts where your storage provider does not provide a FIPS endpoint

Security Groups in Amazon AppStream 2.0

You can provide additional access control to your VPC from streaming instances in a fleet or an image builder in Amazon AppStream 2.0 by associating them with VPC security groups. Security groups that belong to your VPC allow you to control the network traffic between AppStream 2.0 streaming instances and VPC resources such as license servers, file servers, and database servers. For more information, see Security Groups for your VPC in the *Amazon VPC User Guide*.

The rules that you define for your VPC security group are applied when the security group is associated with a fleet or image builder. The security group rules determine what network traffic

is allowed from your streaming instances. For more information, see Security Group Rules in the *Amazon VPC User Guide*.

You can associate up to five security groups while launching a new image builder or while creating a new fleet. You can also associate security groups with an existing fleet or change the security groups for a fleet (to change security groups for a fleet, you must first stop the fleet). For more information, see Working with Security Groups in the *Amazon VPC User Guide*.

If you don't select a security group, your image builder or fleet is associated with the default security group for your VPC. For more information, see Default Security Group for Your VPC in the *Amazon VPC User Guide*.

Use these additional considerations when using security groups with AppStream 2.0.

- All end user data, such as internet traffic, home folder data, or application communication with VPC resources, are affected by the security groups associated with the streaming instance.
- Streaming pixel data is not affected by security groups.
- If you have enabled default internet access for your fleet or image builder, the rules of the associated security groups must allow internet access.

You can create or edit rules for your security groups or create new security groups using the Amazon VPC console.

- **To associate security groups with an image builder** — Follow the instructions at Launch an Image Builder to Install and Configure Streaming Applications.
- **To associate security groups with a fleet**
  - *While creating the fleet* — Follow the instructions at Create a Fleet.
  - *For an existing fleet* — Edit the fleet settings using the AWS Management Console.

You can also associate security groups to your fleets using the AWS CLI and SDKs.

- **AWS CLI** — Use the create-fleet and update-fleet commands.
- **AWS SDKs** — Use the CreateFleet and UpdateFleet API operations.

For more information, see the AWS Command Line Interface User Guide and Tools for Amazon Web Services.

Update Management in Amazon AppStream 2.0

AppStream 2.0 provides an automated way to update your image builder with newer AppStream 2.0 software. When your images are configured to always use the latest AppStream 2.0 agent version, your streaming instances are automatically updated with the latest features, performance improvements, and security updates that are available from AWS. For information about how to manage AppStream 2.0 agent versions, see Manage AppStream 2.0 Agent Versions.

You are responsible for installing and maintaining the updates for the Windows operating system, your applications, and their dependencies. For more information, see Keep Your AppStream 2.0 Image Up-to-Date.

You can keep your AppStream 2.0 image up-to-date by using managed AppStream 2.0 image updates. This update method provides the latest Windows operating system updates and driver updates, and the latest AppStream 2.0 agent software. For more information, see Update an Image by Using Managed AppStream 2.0 Image Updates.

To manage updates for applications on your streaming instances, you can use any automatic update services provided. You can also follow the recommendations for installing updates provided by the application vendor.

Amazon AppStream 2.0 Cross-Service Confused Deputy Prevention

The confused deputy problem is a security issue where an entity that doesn't have permission to perform an action coerces a more-privileged entity to perform the action. In AWS, cross-service impersonation can leave account resources vulnerable to the confused deputy problem. Cross-service impersonation occurs when one service (the *calling service*) calls another service (the *called service*). The calling service can manipulate the called service to use its permissions to act on a customer's resources in ways that the calling service doesn't have permission to perform for itself. To prevent this, AWS provides tools that helps you protect your data for all services with service principals that have access to resources in your account.

We recommend using the `aws:SourceArn` and `aws:SourceAccount` global condition context keys in resource policies to limit permissions when accessing these resources. The following guidelines detail recommendations and requirements when you use these keys to protect your resources:

- Use `aws:SourceArn` if you want only one resource associated with cross-service access.
- Use `aws:SourceAccount` if you want to allow any resource in the specified account associated with cross-service use.
- If the `aws:SourceArn` key doesn't contain an account ID, you must use both global condition context keys (`aws:SourceArn` and `aws:SourceAccount`) to limit permissions.
- If you use both global condition context keys and the `aws:SourceArn` value contains an account ID, the `aws:SourceAccount` key must use the same account ID when used in the same policy statement.

The most effective way to protect against the confused deputy problem is to use the exact Amazon Resource Name (ARN) of the resource you want to allow. If you don't know the full ARN of the resource, use the `aws:SourceArn` global context condition key with wildcards (such as *) for the unknown portions of the ARN. You can also use a wildcard in the ARN if you want to specify multiple resources. For example, you can format the ARN as `arn:aws:`*`servicename`*`::`*`region-name`*`::`*`your AWS account ID`*`:*`.

Example: AppStream 2.0 service role cross-service confused deputy prevention

AppStream 2.0 assumes a service role using a variety of resource ARNs, which leads to a complicated conditional statement. We recommend using a wildcard resource type to prevent any unexpected AppStream 2.0 resources failures.

Example `aws:SourceAccount` Conditional:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "appstream.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "your AWS account ID"
```

```
                }
            }
        }
    ]
}
```

Example `aws:SourceArn` Conditional:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "appstream.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:{aws partition}:appstream:{your region name}:
{your AWS account ID}:*"
                }
            }
        }
    ]
}
```

Example: AppStream 2.0 fleet machine role cross-service confused deputy prevention

Example `aws:SourceAccount` Conditional:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "appstream.amazonaws.com"
```

```
                ]
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "your AWS account ID"
                }
            }
        }
    ]
}
```

Example `aws:SourceArn` Conditional:

> **Note**
>
> If you want to use one IAM role for multiple fleets, we recommend using the `aws:SourceArn` global context condition key with wildcards (*) to match multiple AppStream 2.0 fleet resources.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "appstream.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:{aws partition}:appstream:{your region name}:
{your AWS account ID}:fleet/{your fleet name}"
                }
            }
        }
    ]
}
```

Example: AppStream 2.0 Elastic fleets session script Amazon S3 bucket policy cross-service confused deputy prevention

Example `aws:SourceAccount` Conditional:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "appstream.amazonaws.com"
                ]
            },
            "Action": "s3:GetObject",
            "Resource": "your session script S3 path",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "your AWS account ID"
                }
            }
        }
    ]
}
```

Example `aws:SourceArn` Conditional:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "appstream.amazonaws.com"
                ]
            },
            "Action": "s3:GetObject",
            "Resource": "your session script S3 path",
            "Condition": {
                "ArnLike": {
```

```
                    "aws:SourceArn": "arn:{aws partition}:appstream:{your region name}:
{your AWS account ID}:fleet/{your fleet name}"
                }
            }
        }
    ]
}
```

Example: AppStream 2.0 Application Amazon S3 bucket policy cross-service confused deputy prevention

When you store data in an Amazon S3 bucket, the bucket might be exposed to confused deputy issues. This can leave data such as Elastic fleets, app blocks, setup scripts, application icons, and session scripts vulnerable to malicious actors.

To prevent confused deputy issues, you can specify the `aws:SourceAccount` condition or the `aws:SourceArn` condition in the Amazon S3 bucket policy for `ELASTIC-FLEET-EXAMPLE-BUCKET`.

The resource policies below show how to prevent the confused deputy problem with either of the following:

- The `aws:SourceAccount` with your AWS account ID
- The global condition context key `aws:SourceArn`

AppStream 2.0 currently doesn't support confused deputy prevention for application icons. The service only supports VHD files and setup scripts. If you try to add additional conditions for application icons, the icons won't be displayed to end users.

In the following example, the bucket policy only allows AppStream 2.0 Elastic fleet resources in the owner's account to access `ELASTIC_FLEET_EXAMPLE_BUCKET`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ConfusedDeputyPreventionExamplePolicy",
            "Effect": "Allow",
            "Principal": {
                "Service": "appstream.amazonaws.com"
```

```
            },
            "Action": "s3:GetObject",
            "Resource": [
                "arn:aws:s3:::ELASTIC-FLEET-EXAMPLE-BUCKET/vhd-folder/*",
                "arn:aws:s3:::ELASTIC-FLEET-EXAMPLE-BUCKET/scripts/*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "your AWS account ID"
                }
            }
        },
        {
            "Sid": "AllowRetrievalPermissionsToS3AppIconsForAppStream",
            "Effect": "Allow",
            "Principal": {
                "Service": "appstream.amazonaws.com"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::ELASTIC-FLEET-EXAMPLE-BUCKET/app-icons/*"
        }
    ]
}
```

You can also use the `aws:SourceArn` condition to limit resource access for specific resources.

> **Note**
>
> If you don't know the full ARN of a resource, or you want to specify multiple resources, use the `aws:SourceArn` global context condition key with wildcards (*) for the unknown portions of the ARN.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ConfusedDeputyPreventionExamplePolicy",
            "Effect": "Allow",
            "Principal": {
                "Service": "appstream.amazonaws.com"
            },
```

```
            "Action": "s3:GetObject",
            "Resource": [
                "arn:aws:s3:::ELASTIC-FLEET-EXAMPLE-BUCKET/vhd-folder/*",
                "arn:aws:s3:::ELASTIC-FLEET-EXAMPLE-BUCKET/scripts/*"
            ],
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:{aws-partition}:appstream:{your region name}:
{your AWS account ID}:app-block/*"
                }
            }
        },
        {
            "Sid": "AllowRetrievalPermissionsToS3AppIconsForAppStream",
            "Effect": "Allow",
            "Principal": {
                "Service": "appstream.amazonaws.com"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::ELASTIC-FLEET-EXAMPLE-BUCKET/app-icons/*"
        }
    ]
}
```

You can use the `aws:SourceArn` and `aws:SourceAccount` conditions to limit the resource access for specific resources and accounts.

> **Note**
>
> If you don't know the full ARN of a resources, or if you want to specify multiple resources, use the `aws:SourceArn` global context condition key with wildcards (*) for the unknown portions of the ARN.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ConfusedDeputyPreventionExamplePolicy",
            "Effect": "Allow",
            "Principal": {
                "Service": "appstream.amazonaws.com"
```

```
            },
            "Action": "s3:GetObject",
            "Resource": [
                "arn:aws:s3:::ELASTIC-FLEET-EXAMPLE-BUCKET/vhd-folder/*",
                "arn:aws:s3:::ELASTIC-FLEET-EXAMPLE-BUCKET/scripts/*"
            ],
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:{aws partition}:appstream:{your region name}:
{your AWS account ID}:app-block/*"
                },
                "StringEquals": {
                    "aws:SourceAccount": "your AWS account ID"
                }
            }
        },
        {
            "Sid": "AllowRetrievalPermissionsToS3AppIconsForAppStream",
            "Effect": "Allow",
            "Principal": {
                "Service": "appstream.amazonaws.com"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::ELASTIC-FLEET-EXAMPLE-BUCKET/app-icons/*"
        }
    ]
}
```

Troubleshooting

If you encounter difficulties when working with Amazon AppStream 2.0, consult the following troubleshooting resources.

Contents

- General Troubleshooting
- Troubleshooting Image Builders
- Troubleshooting Fleets
- Troubleshooting Active Directory
- Troubleshooting AppStream 2.0 User Issues
- Troubleshooting Persistent Storage Issues
- Troubleshooting Notification Codes

General Troubleshooting

The following are general issues that might occur when you use Amazon AppStream 2.0.

Issues

- SAML federation is not working. The user is not authorized to view AppStream 2.0 applications.
- After federating from an ADFS portal, my streaming session doesn't start. I am getting the error "Sorry connection went down".
- I get an invalid redirect URI error.
- My image builders and fleets never reach the running state. My DNS servers are in a Simple AD directory.
- I've enabled application settings persistence for my users, but their persistent application settings aren't being saved or loaded.
- I've enabled application settings persistence for my users, but for certain streaming applications, my users’ passwords aren’t persisting across sessions.
- Google Chrome data is filling the VHD file that contains my users' persistent application settings. This is preventing their settings from persisting. How can I manage the Chrome profile?
- I set up a custom domain for my embedded AppStream 2.0 streaming sessions, but my AppStream 2.0 streaming URLs aren't redirecting to my custom domain.

- I launched an app on a smartcard-enabled AppStream 2.0 fleet, and there are a limited number of certificates (or none) available to the app for authentication.
- The Certification Propagation service isn't starting on my smartcard-enabled AppStream 2.0 fleet.

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

This might happen because the inline policy that is embedded for the SAML 2.0 federation IAM role does not include permissions to the stack ARN. The IAM role is assumed by the federated user who is accessing an AppStream 2.0 stack. Edit the role permissions to include the stack ARN. For more information, see Amazon AppStream 2.0 Integration with SAML 2.0 and Troubleshooting SAML 2.0 Federation with AWS in the *IAM User Guide*.

After federating from an ADFS portal, my streaming session doesn't start. I am getting the error "Sorry connection went down".

Set the claim rule's **Incoming Claim Type** for the **NameID** SAML attribute to **UPN** and try the connection again.

I get an invalid redirect URI error.

This error occurs 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 that is displayed in the 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 Amazon AppStream 2.0 Integration with SAML 2.0.

My image builders and fleets never reach the running state. My DNS servers are in a Simple AD directory.

AppStream 2.0 relies on the DNS servers within your VPC to return a non-existent domain (NXDOMAIN) response for local domain names that don’t exist. This enables the AppStream 2.0-managed network interface to communicate with the management servers.

When you create a directory with Simple AD, AWS Directory Service creates two domain controllers that also function as DNS servers on your behalf. Because the domain controllers don't provide the NXDOMAIN response, they can't be used with AppStream 2.0.

I've enabled application settings persistence for my users, but their persistent application settings aren't being saved or loaded.

AppStream 2.0 automatically saves application settings that are created in certain locations on the Windows instance. The settings are saved only if your application saves them to one of these locations. For a list of supported locations, see How Application Settings Persistence Works. If your application is configured to save to C:\Users\%username% and your users' settings for the application aren't persisting between sessions, the mount point might not be created. This prevents the settings from being saved to the VHD file that contains your users' persistent application settings.

To resolve this issue, follow these steps:

1. On the fleet instance, open File Explorer and browse to the user profile directory at C:\Users\%username%.
2. Confirm whether this directory contains a symlink, and then do either of the following:
   - If there is a symlink, confirm that it points to D:\%username%.
   - If there isn't a symlink, try to delete the C:\Users\%username% directory.

     If you can't delete this directory, identify the file in the directory that is preventing it from being deleted and the application that created the file. Then contact the application vendor for information about how to change the file permissions or move the file.

     If you can delete this directory, contact AWS Support for further guidance to resolve this issue. For more information, see AWS Support Center.

I've enabled application settings persistence for my users, but for certain streaming applications, my users' passwords aren't persisting across sessions.

This issue occurs when:

- Users are streaming applications such as Microsoft Outlook, which use the Microsoft Data Protection API.
- App settings persistence is enabled for streaming instances that are not joined to Active Directory domains.

In cases where a streaming instance is not joined to an Active Directory domain, the Windows user, PhotonUser, is different on each fleet instance. Due to the way in which the DPAPI security model works, users' passwords don't persist for applications that use DPAPI in this scenario. In cases where streaming instances are joined to an Active Directory domain and the user is a domain user, the Windows user name is that of the logged in user, and users' passwords persist for applications that use DPAPI.

Google Chrome data is filling the VHD file that contains my users' persistent application settings. This is preventing their settings from persisting. How can I manage the Chrome profile?

By default, Google Chrome stores both user data and the local disk cache in the Windows user profile. To prevent the local disk cache data from filling the VHD file that contains users' persistent application settings, configure Chrome to save only the user data. To do so, on the fleet instance, open the command line as an administrator and start Chrome with the following parameters to change the location of the disk cache:

```
chrome.exe --disk-cache-dir C:\path-to-unsaved-location\
```

Running Chrome with these parameters prevents the disk cache from being persisted between AppStream 2.0 sessions.

I set up a custom domain for my embedded AppStream 2.0 streaming sessions, but my AppStream 2.0 streaming URLs aren't redirecting to my custom domain.

To resolve this issue, verify that when you created your AppStream 2.0 streaming URL, you replaced the AppStream 2.0 endpoint with your custom domain. By default, AppStream 2.0 streaming URLs are formatted as follows:

```
https://appstream2.region.aws.amazon.com/authenticate?parameters=authenticationcode
```

To replace the default AppStream 2.0 endpoint in your streaming URL, replace **`https://appstream2.`***`region`* in the URL with your custom domain. For example, if your custom domain is **`training.example.com`**, your new streaming URL must follow this format:

```
https://training.example.com/authenticate?parameters=authenticationcode
```

For more information about configuring custom domains for embedded AppStream 2.0 streaming sessions, see Configuration Requirements for Using Custom Domains.

I launched an app on a smartcard-enabled AppStream 2.0 fleet, and there are a limited number of certificates (or none) available to the app for authentication.

This happens when the application is launched before the Certificate Propagation service is in a running state.

To resolve this issue, use the PowerShell module Get-Service to query the Certificate Propagation service's status, and make sure that it's in a running state before launching your application.

For example, the following script won't launch the application until the Certificate Propagation service is running:

```
$logFile = "$Env:TEMP\AS2\Logging\$(Get-Date -Format "yyyy-MM-dd-HH-mm-
ss")_applaunch.log"
New-Item -path $logfile -ItemType File -Force | Out-Null

Function Write-Log {
    Param ([string]$message)
    $stamp = Get-Date -Format "yyyy/MM/dd HH:mm:ss"
    $logoutput = "$stamp $message"
    Add-content $logfile -value $logoutput
}

if (Get-Service -Name "CertPropSvc" | Where-Object -Property Status -eq Running) {

    Write-Log "The Certificate Propagation Service is running. Launching
 Application..."
    try {
        Start-Process -FilePath "Path to Application" -WindowStyle Maximized -
ErrorAction Stop
```

```
    }
    catch {
        Write-Log "There was an error launching the application: $_"

    }

}
else {

    do {

        $status = Get-Service "CertPropSvc" | select-object -ExpandProperty Status
        Write-Log "The Certificate Propagation service status is currently $status"
        Start-Sleep -Seconds 2


    } until (Get-Service -Name "CertPropSvc" | Where-Object -Property Status -eq
 Running)

    write-log "The Certificate Propagation Service is running. Launching
 Application..."
    try {
        Start-Process -FilePath "Path to Application" -WindowStyle Maximized -
ErrorAction Stop
    }
    catch {
        Write-Log "There was an error launching the application: $_"

    }
}
```

The Certification Propagation service isn't starting on my smartcard-enabled AppStream 2.0 fleet.

If the Certificate Propagation service isn't starting, the service's startup type might be set to **Disabled**. To resolve this, on the AppStream 2.0 image builder used to create your fleet's image, launch the Windows Services Microsoft Management Console, and make sure that the startup type of the Certificate Propagation service isn't set to **Disabled**.

If the startup type isn't set to **Disabled**, and the service is still not starting on your AppStream 2.0 fleet, use the PowerShell module Start-Service to start the Certificate Propagation service when your fleet instance starts.

For example, the following PowerShell script will start the service if it detects that it's in a stopped state:

```
$logFile = "C:\AppStream\Logging\$(Get-Date -Format "yyyy-MM-dd-HH-mm-
ss")_certpropcheck.log"
New-Item -path $logfile -ItemType File -Force | Out-Null

Function Write-Log {
    Param ([string]$message)
    $stamp = Get-Date -Format "yyyy/MM/dd HH:mm:ss"
    $logoutput = "$stamp $message"
    Add-content $logfile -value $logoutput
}

if (Get-Service -Name "CertPropSvc" | Where-Object -Property Status -eq Running) {

    Write-Log "The Certificate Propagation Service is running. Exiting..."
    Exit
}
else {
    do {

        if (Get-Service -Name "CertPropSvc" | Where-Object -Property Status -eq
 Stopped) {

            Write-Log "The Certificate Propagation Service is stopped, attepmting to
 start..."
            try {
                Start-Service -Name "CertPropSvc" -ErrorAction Stop

            }
            catch {
                Write-Log "There was a problem starting the service: $_"
                break
            }

            $status = Get-Service "CertPropSvc" | select-object -ExpandProperty Status
            Write-Log "The Certificate Propagation service status is currently $status"


        }
        else {
```

```
            $status = Get-Service "CertPropSvc" | select-object -ExpandProperty Status
            Write-Log "The Certificate Propagation service status is currently $status"
            break
        }

    } until (Get-Service -Name "CertPropSvc" | Where-Object -Property Status -eq
 Running)
}
```

Troubleshooting Image Builders

The following are issues that might occur when you use Amazon AppStream 2.0 image builders.

Issues

- I cannot connect to the internet from my image builder.
- When I tried installing my application, I see an error that the operating system version is not supported.
- I want to use a Windows PowerShell script to open my applications.
- I want to make ClickOnce applications available to users.
- 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.
- When I switched between admin and test modes, I saw a request for a password. I don't know how to get a password.
- I get an error when I add my installed application.
- I accidentally quit a background service on the image builder and got disconnected. I am now unable to connect to that image builder.
- The application fails to launch in test mode.
- The application could not connect to a network resource in my VPC.
- 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.
- My application is missing a command line parameter when launching.
- I am unable to use my image with a fleet after installing an antivirus application.
- My image creation failed.

- The Image Assistant create-image operation failed with an error message that access to the PrewarmManifest.txt is denied

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. Or, you can configure an internet gateway in your VPC, and attach an Elastic IP address to your image builder. For more information, see Networking and Access for Amazon AppStream 2.0.

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, Windows Server 2016, and Windows Server 2019 can be added to an AppStream 2.0 image. Check if your application is supported on one of these three operating systems, as applicable for your image builder.

I want to use a Windows PowerShell script to open my applications.

You can use Windows PowerShell scripts to open your applications in the fleet instance. You might want to do this to configure the application or environment before the application opens. To launch a Windows PowerShell script for your application, specify the PowerShell .exe file in Image Assistant. Navigate to `C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe`, and specify the following launch parameters:

-file "C:\Path\To\PowerShell\Script.ps1"

> **Note**
>
> To allow the specified script to open the application, you must override the PowerShell script execution policy. To do so, add **`-ExecutionPolicy Bypass`** to the launch parameter.

I want to make ClickOnce applications available to users.

To make a ClickOnce application available to your AppStream 2.0 users, you must install the application on your image builder first as an Administrator, and then as a Template User. Because ClickOnce applications require a user-specific installation, you must install your application as a Template User to enable users to launch the application from fleet instances. To install the ClickOnce application as an Administrator and then as a Template User, perform these steps.

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. In the left navigation pane, choose **Images**, **Image Builder**.
3. In the list, select the image builder that you want to use, and log into it as an Administrator.
4. Create a batch file that calls the `appref-ms` file within the user profile. Use the %APPDATA % environment variable to replace C:\Users\username\AppData\Roaming. Here is an example batch file call:

   ```
   explorer "%APPDATA%\Microsoft\Windows\Start Menu\Programs\Company\ClickOnce.appref-ms"
   ```

5. On the image builder desktop, open Image Assistant.
6. On the **Configure Apps** page, choose **Switch user**.
7. On the **Local User** tab, choose **Template User**.
8. After you log in as a Template User, install the application again.
9. On the image builder desktop, open Image Assistant.
10. On the **Configure Apps** page, open the ClickOnce application to verify that it functions correctly. After you finish testing, choose **Switch user**.
11. Log back in as an Administrator and perform the required steps in Image Assistant to finish creating your image.

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 might 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 might 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 `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 must launch (create) a new image builder. Do not stop any background services running on the image builder instance. Doing so might 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 might 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 time settings, 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 must 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. For more information, see AWS Support Center.

The Image Assistant `create-image` operation failed with an error message that access to the PrewarmManifest.txt is denied

The application optimization manifest was created with elevated privileges. To create the image, do either of the following, and then try again:

- Run the Image Assistant command line interface (CLI) executable file (Image-Assistant.exe) with administrator privileges.
- Delete the application optimization manifest file.

Troubleshooting Fleets

The following are issues that might occur when users connect to Amazon AppStream 2.0 streaming sessions launched from fleet instances.

Issues

- I tried to increase my fleet capacity, but the update isn't taking effect.
- My applications won't work correctly unless I use the Internet Explorer defaults. How do I restore the Internet Explorer default settings?
- I need to persist environment variables across my fleet instances.
- I want to change the default Internet Explorer home page for my users.
- When my users end a streaming session and then start a new one, they see a message that says no streaming resources are available.

I tried to increase my fleet capacity, but the update isn't taking effect.

You can increase your fleet capacity in either of the two following ways:

- Manually, by increasing the **Minimum capacity** value on the **Scaling Policies** tab for the fleet in the AppStream 2.0 console.
- Automatically, by configuring a fleet scaling policy that manages your capacity for the fleet.

If your manual modification or scaling policy exceeds your current AppStream 2.0 quota for your fleet instance type and size, the new values will not take effect. If you experience this issue, you can use the AWS Command Line Interface (CLI) describe-scaling-activities command to verify

whether your capacity request exceeds your quota for the applicable fleet instance type and size. This command uses the following format:

```
aws application-autoscaling describe-scaling-activities
  --service-namespace appstream \
  --resource-id fleet/fleetname \
```

For example, the following command provides information for the **`TestFleet`** fleet in the **`us-west-2`** AWS Region.

```
aws application-autoscaling describe-scaling-activities --service-namespace appstream
 --resource-id fleet/TestFleet --region us-west-2
```

The following JSON output shows that a scaling policy for **`TestFleet`** with a **Minimum capacity** value of 150 was set. This value exceeds the limit (quota) for **`TestFleet`**, which is 100, so the new scaling policy doesn't take effect. In the output, the **StatusMessage** parameter provides detailed information about the cause of the error, including the fleet instance type (in this case, stream.standard.medium), and the current quota, which is 100.

> **Note**
>
> AppStream 2.0 instance type and size quotas are per Amazon Web Services account, per AWS Region. If you have multiple fleets in the same Region that use the same instance type and size, the total number of instances in all fleets in that Region must be less than or equal to the applicable quota.

```
{
    "ScalingActivities": [
        {
            "ActivityId": "id",
            "ServiceNamespace": "appstream",
            "ResourceId": "fleet/TestFleet",
            "ScalableDimension": "appstream:fleet:DesiredCapacity",
            "Description": "Setting desired capacity to 150.",
            "Cause": "minimum capacity was set to 150",
            "StartTime": 1596828816.136,
            "EndTime": 1596828816.646,
```

```
            "StatusCode": "Failed",
            "StatusMessage": "Failed to set desired capacity to 150. Reason: The
Instance type 'stream.standard.medium' capacity limit for fleet TestFleet' was
exceeded. Requested: 150, Limit: 100 (Service: AmazonAppStream; Status Code: 400;
Error Code: LimitExceededException; Request ID: id; Proxy: null)."
```

If you run the `describe-scaling-activities` command and the output indicates that your capacity request exceeds your current quota, you can resolve the issue by:

- Changing your capacity request to a value that doesn't exceed your quota.
- Requesting a quota increase. To request a quota increase, use the AppStream 2.0 Limits form.

My applications won't work correctly unless I use the Internet Explorer defaults. How do I restore the Internet Explorer default settings?

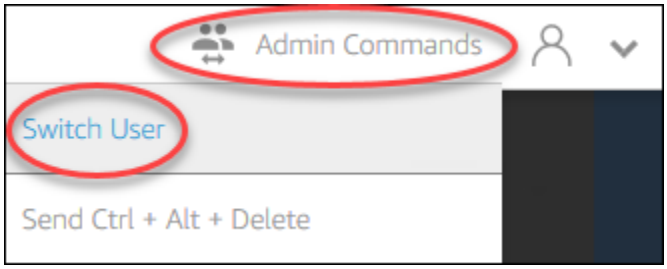
If your AppStream 2.0 environment includes applications that render elements, you might need to restore the Internet Explorer default settings to enable full enable access to the internet.

To automatically restore the Internet Explorer default settings

1. Open the AppStream 2.0 console at https://console.aws.amazon.com/appstream2.
2. In the left navigation pane, choose **Images**, **Image Builder**.
3. Choose the image builder on which to restore the Internet Explorer default settings, verify that it is in the **Running** state, and choose **Connect**.
4. Log in to the image builder by doing either of the following:
   - If your image builder is not joined to an Active Directory domain, on the **Local User** tab, choose **Template User**.
   - If your image builder is joined to an Active Directory domain, choose the **Directory User** tab, enter the credentials for a domain user who does not have local administrator permissions on the image builder, and then choose **Log in**.
5. Open Internet Explorer and reset your settings by doing the following:
   a. In the upper right area of the Internet Explorer browser window, choose the **Tools** icon, then choose **Internet options**.
   b. Choose the **Advanced** tab, and then choose **Reset**.
   c. When prompted to confirm your choice, choose **Reset** again.

   d. When the **Reset Internet Explorer Settings** message is displayed, choose **Close**.

6. In the upper right area of the image builder desktop, choose **Admin Commands**, **Switch User**.



7. This disconnects your current session and opens the login menu. Do either of the following:

   - If your image builder is not joined to an Active Directory domain, on the **Local User** tab, choose **Administrator**.
   - If your image builder is joined to an Active Directory domain, choose the **Directory User** tab, and log in as a domain user who has local administrator permissions on the image builder.

8. On the image builder desktop, open Image Assistant.

9. Follow the required steps in Image Assistant to finish creating your image. For more information, see Tutorial: Create a Custom AppStream 2.0 Image by Using the AppStream 2.0 Console.

I need to persist environment variables across my fleet instances.

Environment variables enable you to dynamically pass settings across applications. You can make user environment variables and system environment variables available across your fleet instances. You can also create environment variables with limited scope, which is useful when you need to use the same environment variable with different values across different applications. For more information, see Persist Environment Variables.

I want to change the default Internet Explorer home page for my users.

You can use Group Policy to set the default home page in Internet Explorer for your users. You can also enable users to change the default page that you set. For more information, see Change the Default Internet Explorer Home Page for Users' Streaming Sessions.

When my users end a streaming session and then start a new one, they see a message that says no streaming resources are available.

When a user ends a session, AppStream 2.0 terminates the underlying instance and creates a new instance if needed to meet the desired capacity of the fleet. If a user tries to start a new session before AppStream 2.0 creates the new instance and all other instances are in use, the user will receive an error stating that no streaming resources are available. If your users start and stop sessions frequently, consider increasing your fleet capacity. For more information, see Fleet Auto Scaling for Amazon AppStream 2.0. Or, consider increasing the maximum session duration for your fleet and instructing your users to close their browser during periods of inactivity rather than ending their session.

Troubleshooting Active Directory

The following are issues that might occur when you set up and use Active Directory with Amazon AppStream 2.0. For help troubleshooting notification codes, see Troubleshooting Notification Codes.

Issues

- My image builders and fleet instances are stuck in the PENDING state.
- My users aren't able to log in with the SAML application.
- My fleet instances work for one user but don't cycle correctly.
- My user Group Policy objects aren't being successfully applied.
- My AppStream 2.0 streaming instances aren't joining the Active Directory domain.
- User login is taking a long time to complete on a domain-joined streaming session.
- My users can't access a domain resource in a domain-joined streaming session, but they can access the resource from a domain-joined image builder.
- My users receive the error “Certificate-Based Authentication not available” and are prompted to enter their domain password. Or users receive the error “Disconnected from session” when they are starting a session enabled with certificate-based authentication.

My image builders and fleet instances are stuck in the PENDING state.

Image builders and fleet instances can take up to 25 minutes to move into a ready state and become available. If your instances are taking longer than 25 minutes to become available, in

Active Directory, verify whether new computer objects were created in the correct organizational units (OUs). If there are new objects, the streaming instances will be available soon. If the objects aren't there, check the directory configuration details in your AppStream 2.0 Directory Config: Directory name (the fully qualified domain name of the directory, service account sign-in credentials, and the OU distinguished name.

Image builder and fleet errors are displayed in the AppStream 2.0 console on the **Notifications** tab for the fleet or image builder. Fleet errors are also available using the AppStream 2.0 API through the DescribeFleets operation or the CLI command describe-fleets.

My users aren't able to log in with the SAML application.

AppStream 2.0 relies on the SAML_Subject "NameID" attribute from your identity provider to populate the username field to log in your user. The username can either be formatted as "`domain\username`", or "`user@domain.com`". If you are using "`domain\username`" format, `domain` can either be the NetBIOS name or the fully qualified domain name. If using "`user@domain.com`" format, the UserPrincipalName attribute can be used. If you've verified your SAML_Subject attribute is configured correctly and the problem persists, contact AWS Support. For more information, see AWS Support Center.

My fleet instances work for one user but don't cycle correctly.

Fleet instances are cycled after a user completes a session, ensuring that each user has a new instance. When the cycled fleet instance is brought online, it joins the domain using the computer name of the previous instance. To ensure that this operation happens successfully, the service account requires **Change Password** and **Reset Password** permissions on the organizational unit (OU) to which the computer object is joining. Check the service account permissions and try again. If the problem persists, contact AWS Support. For more information, see AWS Support Center.

My user Group Policy objects aren't being successfully applied.

By default, computer objects apply computer-level policies based on the OU in which the computer object resides, while applying user-level policies based on the OU in which the user resides. If your user-level policies aren't being applied, you can do one of the following:

- Move the user-level policies to the OU in which the user Active Directory object resides
- Enable computer-level loopback processing, which applies the user-level policies in the computer object OU.

For more information, see Loopback processing of Group Policy at Microsoft Support.

My AppStream 2.0 streaming instances aren't joining the Active Directory domain.

The Active Directory domain to use with AppStream 2.0 must be accessible through its fully qualified domain name (FQDN) through the VPC in which your streaming instances are launched.

To test that your domain is accessible

1. Launch an Amazon EC2 instance in the same VPC, subnet, and security groups that you use with AppStream 2.0.
2. Manually join the EC2 instance to your Active Directory domain by using the FQDN (for example, `yourdomain.example.com`) with the service account that you intend to use with AppStream 2.0. Use the following command in a Windows PowerShell console:

   ```
   netdom join computer /domain:FQDN /OU:path /ud:user /pd:password
   ```

   If this manual join fails, go to the next step.
3. If you cannot manually join to your domain, open a command prompt and verify that you can resolve the FQDN using the `nslookup` command. For example:

   ```
   nslookup yourdomain.exampleco.com
   ```

   Successful name resolution returns a valid IP address. If you are unable to resolve your FQDN, you might need to update your VPC DNS servers by using a DHCP option set for your domain. Then, come back to this step. For more information, see DHCP Options Sets in the *Amazon VPC User Guide*.
4. If the FQDN resolves, use the `telnet` command to validate connectivity.

   ```
   telnet yourdomain.exampleco.com 389
   ```

   A successful connection shows a blank command prompt window without any connection errors. You might need to install the Telnet Client feature on your EC2 instance. For more information, see Install Telnet Client in the Microsoft documentation.

If you were not able to manually join the EC2 instance to your domain, but were successful in resolving the FQDN and testing connectivity with the Telnet Client, your VPC security groups might be preventing access. Active Directory requires certain network port settings. For more information, see Active Directory and Active Directory Domain Services Port Requirements in the Microsoft documentation.

User login is taking a long time to complete on a domain-joined streaming session.

AppStream 2.0 performs a Windows login action after users provide their domain password. After successful authentication, AppStream 2.0 launches the application. The login and launch times are impacted by many variables, such as network contention for the domain controllers or the time it takes to apply Group Policy settings to the streaming instance. If domain authentication takes too long to complete, try performing the following actions.

- Minimize the network latency from your AppStream 2.0 Region to your domain controllers by choosing the correct domain controllers. For example, if your fleet is in `us-east-1`, use domain controllers with high bandwidth and low latency to `us-east-1` through Active Directory Sites and Services zone mappings. For more information, see Active Directory Sites and Services in the Microsoft documentation.
- Ensure that your Group Policy settings and user login scripts don't take prohibitively long to apply or run.

If your domain users' login to AppStream 2.0 fails with the message "An unknown error occurred," you might need to update the Group Policy settings described in Before You Begin Using Active Directory with AppStream 2.0. Otherwise, these settings might prevent AppStream 2.0 from authenticating and logging in your domain users.

My users can't access a domain resource in a domain-joined streaming session, but they can access the resource from a domain-joined image builder.

Confirm that your fleet is created in the same VPC, subnets, and security groups as your image builder, and that your user has the permissions required to access and use the domain resource.

My users receive the error “Certificate-Based Authentication not available” and are prompted to enter their domain password. Or users receive the error “Disconnected from session” when they are starting a session enabled with certificate-based authentication.

These errors occur if certificate-based authentication was unsuccessful for the session. The “Certificate-Based Authentication not available” error is displayed when certificate-based authentication is enabled to allow fallback to password logon. The “Disconnected from session” error is displayed when certificate-based authentication is enabled without fallback.

The user can refresh the page on the web client or reconnect from the client for Windows, as this may be an intermittent issue with certificate-based authentication. If the problem continues, certificate-based authentication failure can result from one of the following issues:

- AppStream 2.0 could not communicate with AWS Private CA, or AWS Private CA did not issue the certificate. Check CloudTrail to determine if a certificate was issued. For more information, see What Is AWS CloudTrail? and the section called “Manage Certificate-based Authentication”.
- The domain controller has no domain controller certificate for smart card logon, or it is expired. For more information, see step 7.a in the section called “Prerequisites”.
- The certificate is not trusted. For more information, see step 7.c in the section called “Prerequisites”.
- The userPrincipalName format for the SAML_Subject NameID is not formatted properly, or does not resolve to the actual domain for the user. For more information, see step 1 in the section called “Prerequisites”.
- The (optional) ObjectSid attribute in your SAML assertion does not match the Active Directory security identifier (SID) for the user specified in the SAML_Subject NameID. Confirm that the attribute mapping is correct in your SAML federation, and that your SAML identity provider is synchronizing the SID attribute for the Active Directory user.
- The AppStream 2.0 agent does not support certificate-based authentication. Use AppStream 2.0 agent version 10-13-2022 or later.
- There are Group Policy settings that are modifying the default Active Directory settings for smart card logon, or taking action if a smart card is removed from a smart card reader. These settings may cause additional unexpected behavior other than the errors listed above. Certificate-based authentication presents a virtual smart card to the instance operating system, and removes it after logon is complete. For more information, see Primary Group Policy settings for smart cards

and Additional smart card Group Policy settings and registry keys. Do not enable **Smart card sign in for Active Directory** in your stack if you want to use certificate-based authentication. For more information, see the section called “Smart Cards”.

- The CRL distribution point for the private CA is not online or accessible from either the AppStream 2.0 fleet instance or the domain controller. For more information, see step 5 in the section called “Prerequisites”.

Additional troubleshooting steps involve reviewing the AppStream 2.0 instance Windows event logs. A common event to review for logon failure is 4625(F): An account failed to log on. For more information about capturing log information, see Persisting application and Windows event logs. Alternatively, to troubleshoot an active AppStream 2.0 session as an administrator, you can connect to the logs using an Event Viewer on another computer. For more information, see How to Select Computers in Event Viewer. Or ,you can connect by using Remote Desktop to connect to the instance private IP address from another computer that can connect to Remote Desktop Services in your AppStream 2.0virtual private cloud (VPC). Use the AWS CLI to determine the IP address for the session based on the AWS Region, AppStream 2.0 stack name, fleet name, user ID, and authentication type. For more information, see the AWS Command Line Interface.

If the problem persists, contact AWS Support. For more information, see AWS Support Center.

Troubleshooting AppStream 2.0 User Issues

The following are possible issues that might occur for your users when they use AppStream 2.0.

Issues

- My users' AppStream 2.0 client installations fail, and they're getting a message stating that .NET Framework 4.6 is required.
- My users' USB driver installations fail when they install the AppStream 2.0 client, and now they can't use their USB devices with AppStream 2.0.
- My AppStream 2.0 client users are getting disconnected from their AppStream 2.0 session after every 60 minutes.
- My users can’t copy and paste between their local device and their streaming session.
- Some keyboard shortcuts aren’t working for users during their streaming sessions.
- My users' drawing tablets are not working with the streaming applications I deployed.
- The Japanese language input method doesn't work for my users during their streaming sessions

- My user sees an error about reaching the max number of streaming sessions when they try to launch an application from the application catalog.
- My user sees a black screen or the desktop, and their application doesn't launch on an Elastic fleet. No error appears.

My users' AppStream 2.0 client installations fail, and they're getting a message stating that .NET Framework 4.6 is required.

When users install the AppStream 2.0 client, AppStream 2.0 also installs .NET Framework version 4.6.2, if that version or a later version is not already installed. If the PC on which the client is being installed is not connected to the internet, .NET Framework can't be installed. In this case, a message prompts users to install .NET Framework version 4.6 manually. However, when users choose **Install**, an error message is displayed stating that the installation failed. Users are then prompted to try installing the latest version of the .NET Framework manually. When they choose **Close**, they exit the installation.

To resolve this issue, users must establish an internet connection from the PC on which they plan to install the client, and then download and install .NET Framework version 4.6.2 or later on the same PC. For a list of the .NET Framework versions available for download, see Download .NET Framework.

> **Note**
>
> Users who have version 1.1.156 of the AppStream 2.0 client installed must have .NET Framework version 4.7.2 or later installed on the same PC.

My users' USB driver installations fail when they install the AppStream 2.0 client, and now they can't use their USB devices with AppStream 2.0.

When users install the AppStream 2.0 client, they choose whether to install the AppStream 2.0 USB driver. The driver is required to use USB devices with applications streamed through AppStream 2.0. However, the USB driver installation fails if both of the following occur:

- The root certificate used to sign the `AppStreamUsbDriver.exe` file is not present in the Windows certificate store.

- The PC on which the client is being installed is not connected to the internet.

In this case, the certificate for the Amazon AppStream USB driver can't be validated, and an error message notifies users that the USB driver installation failed. When users choose **OK**, the AppStream 2.0 client installation is completed without the USB driver. Although users can still use the AppStream 2.0 client for application streaming, their USB devices won't work with applications streamed through AppStream 2.0.

To resolve this issue, users must establish an internet connection from the PC on which they plan to install the AppStream 2.0 client, and reinstall the client.

My AppStream 2.0 client users are getting disconnected from their AppStream 2.0 session after every 60 minutes.

If you have configured identity federation using SAML 2.0 for access to AppStream 2.0, depending on your identity provider (IdP), you may need to configure the information that the IdP passes as SAML attributes to AWS as part of the authentication response. This includes configuring the **Attribute** element with the `SessionDuration` attribute set to `https://aws.amazon.com/SAML/Attributes/SessionDuration`.

`SessionDuration` specifies the maximum amount of time that a federated streaming session for a user can remain active before reauthentication is required. Although `SessionDuration` is an optional attribute, we recommend that you include it in the SAML authentication response. If you do not specify this attribute, the session duration is set to a default value of 60 minutes.

To resolve this issue, configure your SAML-compatible IdP to include the `SessionDuration` value in the SAML authentication response, and set the value as required. For more information, see Step 5: Create Assertions for the SAML Authentication Response.

> **Note**
>
> If your users access their streaming applications in AppStream 2.0 by using the AppStream 2.0 native client or by using the web browser on the new experience, their sessions are disconnected after their session duration expires. If your users access their streaming applications in AppStream 2.0 by using a web browser on the old/classic experience, after the users' session duration expires and they refresh their browser page, their sessions are disconnected.

If your users sign in to the new portal experience with a SAML-compatible IdP, and they continue to have random disconnections, it might be due to the session cookies used by the AppStream 2.0 session being invalidated by other web applications using `aws.amazon.com` as a subdomain. The following are common user scenarios:

- If a user initiates a new AppStream 2.0 session in the same browser, the existing AppStream 2.0 session will be disconnected.
- If a user initiates any other web applications in the same browser, resulting in a new user authentication under the `aws.amazon.com` domain, the existing AppStream 2.0 session will be disconnected.
- If a user signs into an AWS Management Console with new IAM credentials in the same browser, the existing AppStream 2.0 session will be disconnected.

You can resolve this issue by using the new relay state endpoints to configure your SAML 2.0 federation, and by using the AppStream 2.0 client version 1.1.1300 and later. For more information, see Table 1 on the section called "Step 6: Configure the Relay State of Your Federation".

My users can't copy and paste between their local device and their streaming session.

AppStream 2.0 takes advantage of the W3C specification for enabling asynchronous clipboard operations in web applications. This enables users to copy and paste content between their local device and their streaming session in the same ways that they copy and paste between applications on their local device, including using keyboard shortcuts.

The only browser that currently supports the W3C asynchronous clipboard specification is Google Chrome version 66 or later, which supports copying and pasting only for text. For all other browsers, users can use the clipboard feature in the AppStream 2.0 web portal, which provides a dialog box for copying or pasting text.

If your users run into issues using the clipboard during their streaming sessions, you can provide them with the following information:

- **I'm using Chrome version 66 or later, and keyboard shortcuts aren't working.**

  Chrome displays a prompt for you to choose whether to allow AppStream 2.0 to access content copied to the clipboard. Choose **Allow** to enable pasting to your remote session. If you're copying

text from your remote session to your local device, both the Chrome application and the tab containing your streaming session must stay in focus on your local device long enough for the text to be copied from your streaming session. Small amounts of text should be copied almost immediately, but for large amounts of text, you might need to wait 1 to 2 seconds before switching away from Chrome or from the tab containing your streaming session. The time required to copy the text varies based on network conditions.

- **Copying and pasting doesn’t work when I try to copy and paste a large amount of text.**

  AppStream 2.0 has a default limit of 20 MB for the amount of text that you can copy and paste between your local device and your streaming session. If you try to copy more than 20 MB, no text is copied. However, the text will be truncated if your admin set a limit and you go beyond that limit. This limit doesn’t apply if you try to copy and paste text between applications on your local device or between applications in your streaming session. Administrators can also limit the number of characters that you copy/paste in/out of your streaming sessions. If you need to copy or paste text more than 20 MB or the specified limit between your local device and your streaming session, you can divide it into smaller chunks or upload it as a file instead.

- **I’m using the AppStream 2.0 web portal clipboard feature to paste text to my streaming session and it’s not working.**

  In some cases, after you paste text into the clipboard dialog box and the dialog box closes, nothing happens when you try to use keyboard shortcuts to paste the text in your streaming session. This issue occurs because when the clipboard dialog box appears, it takes the focus away from your streaming application. After the dialog box closes, the focus might not automatically return to your streaming application. Clicking your streaming application should return the focus to it and enable you to use keyboard shortcuts to paste your text into your streaming session.

Some keyboard shortcuts aren’t working for users during their streaming sessions.

The following keyboard shortcuts work on users' local computers, but are not passed to AppStream 2.0 streaming sessions:

Windows:

- Win+L
- Ctrl+Alt+Del

Mac:

- Ctrl+F3
- All shortcuts that use Alt or Option key combinations

This issue is due to the following limitations on users' local computers:

- The keyboard shortcuts are filtered by the operating system that is running on users' local computers and not propagated to the browsers on which users are accessing AppStream 2.0. This behavior applies to the Windows Win+L and Ctrl+Alt+Del keyboard shortcuts and Mac Ctrl+F3 keyboard shortcut.
- When used with web applications, some keyboard shortcuts are filtered by the browser and don't generate an event for the web applications. As a result, the web applications can't respond to the keyboard shortcuts typed by users.
- The keyboard shortcuts are translated by the browser before a keyboard event is generated and so are not translated correctly. For example, Alt key combinations and Option key combinations on Mac computers are translated as if they are Alt Graph key combinations on Windows. When this occurs, the results are not as the users intend when they use these key combinations.

My users' drawing tablets are not working with the streaming applications I deployed.

If your users' drawing tablets are not working with streaming applications, make sure that you meet the requirements and understand additional considerations for enabling this feature. Following are the requirements and considerations for enabling your users to use drawing tablets during AppStream 2.0 streaming sessions.

> **Note**
>
> Drawing tablets are supported for users who access AppStream 2.0 by using the AppStream 2.0 client, or through a supported web browser.

- To enable your users to use this feature, you must configure your AppStream 2.0 fleet to use an image that runs Windows Server 2019.

- To use this feature, users must access AppStream 2.0 by using the AppStream 2.0 client, or through the Google Chrome or Mozilla Firefox browsers only.
- Streaming applications must support Windows Ink technology. For more information, see Pen interactions and Windows Ink in Windows apps.
- Some applications, such GIMP, must detect drawing tablets on the streaming instance to support pressure sensitivity. If this is the case, your users must use the AppStream 2.0 client to access AppStream 2.0 and stream these applications. In addition, you must qualify your users' drawing tablets, and users must share their drawing tablets with AppStream 2.0 every time they start a new streaming session.
- This feature is not supported on Chromebooks.

The Japanese language input method doesn't work for my users during their streaming sessions

To enable your users to use the Japanese language input method during their AppStream 2.0 streaming sessions, do the following:

- Configure your fleet to use the Japanese input method. To do so, enable the Japanese input method on your image builder when you create an image, and then configure your fleet to use the image. For more information, see Specify a Default Input Method. Doing so enables AppStream 2.0 to automatically configure your image to use a Japanese keyboard. For more information, see Japanese Keyboards.
- Ensure that the Japanese input method is also enabled on the user's local computer.

If the fleet instance and the user’s local computer don't use the same language input method, the mismatch might result in unexpected keyboard inputs on the fleet instance during the user’s streaming sessions. For example, if the fleet instance uses the Japanese input method and the user’s local computer uses the English input method, during a streaming session, the local computer will send keys to the fleet instance that have different key mappings than the fleet instance.

To verify whether the Japanese input method is enabled for a fleet instance, enable the **Desktop** stream view for the fleet. For more information, see Step 6 in Create a Fleet.

Windows Keyboard Shortcuts

Following are Windows keyboard shortcuts for switching Japanese input modes and for Japanese conversions. For these keyboard shortcuts to work, the AppStream 2.0 streaming session must be active.

Windows keyboard shortcuts for switching Japanese input modes

Keyboard shortcut	Description
半角/全角/漢字 (Hankaku/Zenkaku/Kanji) Or Alt+`	Switches the input mode between alphanumeric and Japanese mode
無変換 (Muhenkan)	Converts characters to Hiragana, full-width Katakana, and half-width Katakana in sequence
カタカナ/ひらがな/ローマ字 (Katakana/Hiragana/Romaji)	Changes the input mode to Hiragana
Shift+カタカナ/ひらがな/ローマ字 (Katakana/Hiragana/Romaji)	Changes the input mode to Katakana
Alt+カタカナ/ひらがな/ローマ字 (Katakana/Hiragana/Romaji)	Switches the input mode between Japanese Romaji and Japanese Kana

Windows keyboard shortcuts for Japanese conversions

Keyboard shortcut	Description
変換 (Henkan) + Space	Lists conversion options
F6	Converts to Hiragana

Keyboard shortcut	Description
F7	Converts to full-width Katakana
F8	Converts to half-width Katakana
F9	Converts to full-width Romaji
F10	Converts to half-width Romaji

Mac Keyboard Shortcuts

For information about Mac keyboard shortcuts for switching Japanese input methods and for Japanese conversions, see the following articles in the Mac Support documentation.

> **Note**
>
> Because AppStream 2.0 streaming sessions run on Windows instances, Mac users might experience different key mappings.

- Keyboard shortcuts for switching Japanese input methods — Set up and switch to a Japanese input source on Mac
- Keyboard short link cuts for Japanese conversions — Keyboard shortcuts for Japanese conversions on Mac

My user sees an error about reaching the max number of streaming sessions when they try to launch an application from the application catalog.

With AppStream 2.0 Elastic fleets, you specify a maximum number of users that can stream concurrently using the max concurrency parameter. Any user that tries to stream beyond that value receives this error. To resolve this issue, you can increase the maximum number of concurrent streams, or advise your user to wait for another user to complete their streaming session.

> **Note**
>
> You might need to request a limit increase to increase the instance type and size limit.

My user sees a black screen or the desktop, and their application doesn’t launch on an Elastic fleet. No error appears.

This can happen if the application launch path is incorrect, and AppStream 2.0 can't launch the application. You can validate the application launch path by using Desktop View on the fleet to navigate the root volume. Validate that the application executable exists at the path specified.

If you're not able to find the app block's VHD or setup script on the streaming instance, AppStream 2.0 might not have been able to download them from the S3 bucket. Validate that the VPC you specified has access to S3. For more information, see Using Amazon S3 VPC Endpoints for AppStream 2.0 Features.

Troubleshooting Persistent Storage Issues

Amazon AppStream 2.0 supports the following options for persistent storage: Home folders, Google Drive for G Suite, and OneDrive for Business. Because content synchronization behaviors are consistent across these persistent storage solutions, we recommend that you review Home Folder Content Synchronization for information about expected behavior.

The following are issues that might occur when you or your users use AppStream 2.0 persistent storage.

Issues

- My stack's home folders aren't working correctly.
- My users can't access their home folder directory from one of our applications.
- I removed or replaced a file in a user’s home folder in Amazon S3, but my users don’t see the changes in their home folder on the fleet instance during their streaming sessions.
- Persistent storage isn't performing as expected. My users' files are taking longer than expected to save to persistent storage.
- My users are getting errors that files are already in use when their files are not in use.
- When a folder contains thousands of files, AppStream 2.0 might take a long time to display the list of files.

My stack's home folders aren't working correctly.

Problems with home folder backup to an S3 bucket can occur in the following scenarios:

- There is no internet connectivity from the streaming instance, or there is no access to the private Amazon S3 VPC endpoint, if applicable.
- Network bandwidth consumption is too high. For example, multiple large files are being downloaded or streamed by the user while the service is trying to back up a home folder that contains large files to Amazon S3.
- An administrator deleted the bucket created by the service.
- An administrator incorrectly edited the Amazon S3 permissions for the `AmazonAppStreamServiceAccess` service role.

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

My users can't access their home folder directory from one of our applications.

Some applications do not recognize the redirect that displays the home folder as a top-level folder in File Explorer. If this is the case, your users can access their home folder from within an application during a streaming session by choosing **File Open** from the application interface and browsing to either of the following directories:

- Non-domain-joined Windows instances: C:\Users\PhotonUser\My Files\Home Folder
- Domain-joined Windows instances: C:\Users\%username%\My Files\Home Folder
- Linux instances: ~/MyFiles/HomeFolder

I removed or replaced a file in a user’s home folder in Amazon S3, but my users don’t see the changes in their home folder on the fleet instance during their streaming sessions.

Differences between content that is stored in a user’s home folder in an S3 bucket and content that is available to a user on a fleet instance during their streaming sessions may be due to the way in

which home folder content stored in Amazon S3 buckets is synchronized with home folder content stored on AppStream 2.0 fleet instances.

At the beginning of a user's AppStream 2.0 streaming session, AppStream 2.0 catalogs the user's home folder files stored in the Amazon S3 bucket for your Amazon Web Services account and Region. When a user uses a streaming application to open a file in their home folder on their fleet instance, AppStream 2.0 downloads the file to the fleet instance.

Changes that a user makes to files on a fleet instance during their active streaming session are uploaded to their home folder in the S3 bucket every few seconds, or at the end of the user's streaming session.

If a user opens a file in their home folder on a fleet instance during a streaming session and then closes the file without making any changes or saving the file, and you remove the file from that user's home folder in an S3 bucket during the streaming session, the file is removed from the fleet instance if the user refreshes the folder. If the user modifies the file and saves the file locally, the file remains available to the user on the fleet instance during their current streaming session. The file is also uploaded to the S3 bucket again. However, the file may or may not be available to the user on the fleet instance during their next streaming session.

The availability of the file on the fleet instance during a user's next streaming session depends on whether the user changed the file on the fleet instance before or after you changed the file in the S3 bucket.

For more information, see Home Folder Content Synchronization.

Persistent storage isn't performing as expected. My users' files are taking longer than expected to save to persistent storage.

During AppStream 2.0 streaming sessions, saving large files and directories associated with compute-intensive applications to persistent storage can take longer than saving files and directories required for basic productivity applications. For example, it might take longer for applications to save a large amount of data or frequently modify the same files than it would to save files created by applications that perform a single write action. It might also take longer to save many small files.

If your users save files and directories associated with compute-intensive applications and AppStream 2.0 persistent storage options aren't performing as expected, we recommend that you use a Server Message Block (SMB) solution such as Amazon FSx for Windows File Server or an

AWS Storage Gateway file gateway. Following are examples of files and directories associated with compute-intensive applications that are more suitable for use with these SMB solutions:

- Workspace folders for integrated development environments (IDEs)
- Local database files
- Scratch space folders created by graphics simulation applications

For more information, see:

- *Amazon FSx for Windows File Server Windows User Guide*
- Using Amazon FSx with Amazon AppStream 2.0
- File gateways in the *AWS Storage Gateway User Guide*

> **Note**
>
> Before proceeding with further troubleshooting, first ensure that the issue your users are experiencing with saving files and directories is associated with AppStream 2.0 persistent storage only, and not another cause. To rule out other causes, have your users try saving the files or directories to the Temporary Files directory that is available on their streaming instance.

My users are getting errors that files are already in use when their files are not in use.

This behavior is typically occurs in the following cases:

- When users' files are still being uploaded after the files were last saved
- Files that are modified frequently (for example, database files)

Large file uploads might take significant time. Also, each attempt to upload the file might result in another file update, which can lead to repeated file upload attempts.

To resolve this issue, we recommend that you use a Server Message Block (SMB) solution such as Amazon FSx for Windows File Server or an AWS Storage Gateway file gateway. For more information, see:

- *Amazon FSx for Windows File Server Windows User Guide*
- Using Amazon FSx with Amazon AppStream 2.0
- File gateways in the *AWS Storage Gateway User Guide*

When a folder contains thousands of files, AppStream 2.0 might take a long time to display the list of files.

AppStream 2.0 uses API calls to retrieve the content of folders that are stored in AppStream 2.0 persistent storage. There is a limit to the number of items that an API call can retrieve each time the call runs. For this reason, if AppStream 2.0 must retrieve thousands of files in a single folder, it might take more time to display the list of all the files than it would to display the list of files in a folder that contains fewer files.

To resolve this issue, if you have thousands of files in one folder, we recommend that you divide this content into groups of fewer files and store each group in a different folder. Doing so reduces the number of API calls that are required to display the list of files in each folder.

Troubleshooting Notification Codes

The following are notification codes and resolution steps for notifications that you might see when you set up and use Amazon AppStream 2.0. These notifications can be found in the **Notifications** tab in the AppStream 2.0 console, after selecting an image builder or fleet. You can also get fleet notifications by using the AppStream 2.0 API operation DescribeFleets or the describe-fleets CLI command.

Active Directory Internal Service

Follow these steps if you receive an internal service error when you set up and use Active Directory with Amazon AppStream 2.0.

INTERNAL_SERVICE_ERROR

Message: The user name or password is incorrect.

Resolution: This error might occur when the computer object that was created in the Microsoft Active Directory domain for the resource was deleted or disabled. You can resolve this error by enabling the computer object in the Active Directory domain, and then starting the resource again. You might also need to reset the computer object account in the Active Directory

domain. If you continue to encounter this error, contact AWS Support. For more information, see AWS Support Center.

Active Directory Domain Join

The following are notification codes and resolution steps for issues with domain join that you might encounter when you set up and use Active Directory with Amazon AppStream 2.0.

DOMAIN_JOIN_ERROR_ACCESS_DENIED

Message: Access is denied.

Resolution: The service account specified in the directory configuration does not have permissions to create the computer object or reuse an existing one. Validate the permissions and start the image builder or fleet. For more information, see Granting Permissions to Create and Manage Active Directory Computer Objects.

DOMAIN_JOIN_ERROR_LOGON_FAILURE

Message: The username or password is incorrect.

Resolution: The service account specified in the directory configuration has an invalid username or password. Update the configuration and re-create the image builder or fleet that had the error.

DOMAIN_JOIN_NERR_PASSWORD_EXPIRED

Message: The password of this user has expired.

Resolution: The password for the service account specified in the AppStream 2.0 directory configuration has expired. Change the password for the service account in your Active Directory domain, update the configuration, and then re-create the image builder or fleet that had the error.

DOMAIN_JOIN_ERROR_DS_MACHINE_ACCOUNT_QUOTA_EXCEEDED

Message: Your computer could not be joined to the domain. You have exceeded the maximum number of computer accounts you are allowed to create in this domain. Contact your system administrator to have this limit reset or increased.

Resolution: The service account specified on the directory configuration does not have permissions to create the computer object or reuse an existing one. Validate the permissions

and start the image builder or fleet. For more information, see Granting Permissions to Create and Manage Active Directory Computer Objects.

DOMAIN_JOIN_ERROR_INVALID_PARAMETER

Message: A parameter is incorrect. This error is returned if the `LpName` parameter is NULL or the `NameType` parameter is specified as `NetSetupUnknown` or an unknown nametype.

Resolution: This error can occur when the distinguished name for the OU is incorrect. Validate the OU and try again. If you continue to encounter this error, contact AWS Support. For more information, see AWS Support Center.

DOMAIN_JOIN_ERROR_MORE_DATA

Message: More data is available.

Resolution: This error can occur when the distinguished name for the OU is incorrect. Validate the OU and try again. If you continue to encounter this error, contact AWS Support. For more information, see AWS Support Center.

DOMAIN_JOIN_ERROR_NO_SUCH_DOMAIN

Message: The specified domain either does not exist or could not be contacted.

Resolution: The streaming instance was unable to contact your Active Directory domain. To ensure network connectivity, confirm your VPC, subnet, and security group settings. For more information, see My AppStream 2.0 streaming instances aren't joining the Active Directory domain.

DOMAIN_JOIN_NERR_WORKSTATION_NOT_STARTED

Message: The Workstation service has not been started.

Resolution: An error occurred starting the Workstation service. Ensure that the service is enabled in your image. If you continue to encounter this error, contact AWS Support. For more information, see AWS Support Center.

DOMAIN_JOIN_ERROR_NOT_SUPPORTED

Message: The request is not supported. This error is returned if a remote computer was specified in the `lpServer` parameter and this call is not supported on the remote computer.

Resolution: Contact AWS Support for assistance. For more information, see AWS Support Center.

DOMAIN_JOIN_ERROR_FILE_NOT_FOUND

Message: The system cannot find the file specified.

Resolution: This error occurs when an invalid organizational unit (OU) distinguished name is provided. The distinguished name must start with **OU=**. Validate the OU distinguished name and try again. For more information, see Finding the Organizational Unit Distinguished Name.

DOMAIN_JOIN_INTERNAL_SERVICE_ERROR

Message: The account already exists.

Resolution: This error can occur in the following scenarios:

- If the issue isn't permissions-related, check the Netdom logs for errors and make sure that you provided the correct OU.
- The service account specified in the directory configuration does not have permissions to create the computer object or reuse an existing one. If this is the case, validate the permissions and start the image builder or fleet. For more information, see Granting Permissions to Create and Manage Active Directory Computer Objects.
- After AppStream 2.0 creates the computer object, it is moved from the OU in which it was created. In this case, the first image builder or fleet is created successfully, but any new image builder or fleet that uses the computer object fails. When Active Directory searches for the computer object in the specified OU and detects that an object with the same name exists elsewhere in the domain, the domain join is not successful.
- The name of the OU specified in the AppStream 2.0 Directory Config includes spaces before or after the commas in the directory configuration. In this case, when a fleet or image builder attempts to rejoin the Active Directory domain, AppStream 2.0 cannot cycle the computer objects correctly and the domain rejoin does not succeed. To resolve this issue for a fleet, do the following:

  1. Stop the fleet.
  2. Edit the Active Directory domain settings for the fleet to remove the Directory Config and Directory OU to which the fleet is joined. For more information, see Step 3: Create a Domain-Joined Fleet.
  3. Update the AppStream 2.0 Directory Config to specify an OU that doesn't contain spaces. For more information, see Step 1: Create a Directory Config Object.
  4. Edit the Active Directory domain settings for the fleet to specify the Directory Config with the updated Directory OU.

To resolve this issue for an image builder, do the following:

1. Delete the image builder.
2. Update the AppStream 2.0 Directory Config to specify an OU that doesn't contain spaces. For more information, see Step 1: Create a Directory Config Object.
3. Create a new image builder and specify the Directory Config with the updated Directory OU. For more information, see Launch an Image Builder to Install and Configure Streaming Applications.

Image Internal Service

If you receive an internal service error after you use managed AppStream 2.0 image updates to initiate an image update, follow these steps.

INTERNAL_SERVICE_ERROR

Message: AppStream 2.0 could not update image *image-name*. Failed to update/install/configure/disable <software name>. Check your source image and try again. If this problem persists, contact AWS Support.

Resolution: This error can occur when there is an issue with the source image. Try to update the image again.

If updating again doesn't work, make sure that you're using the latest version of SSM Agent. For version information, see the section called “Base Image and Managed Image Update Release Notes”. For installation information, see Manually install SSM Agent on EC2 instances for Windows Server.

If the error continues to occur, launch an image builder from the image. For more information, see Launch an Image Builder to Install and Configure Streaming Applications. If you can't launch image builder from the image, there is another issue with the image that needs to be resolved before you can use managed AppStream 2.0 image updates to update the image. If you continue to encounter this error, contact AWS Support. For more information, see AWS Support Center.

Session Provisioning

The following are notification codes and resolution steps for issues with session provisioning that you might encounter when your end users try to provision the streaming session.

> **Note**
>
> "X" below equals the number of sessions that encountered a given error code.

USER_PROFILE_MOUNTING_FAILURE

Message: X session(s) encountered user profile mounting failures.

Resolution: To troubleshoot this issue, check if any user profiles have been corrupted or if any third party processes on instance are interfering with user profile mounting. If you continue to encounter this error, contact AWS Support. For more information, see AWS Support Center.

USER_PROFILE_DOWNLOADING_FAILURE

Message: X session(s) encountered user profile downloading failures.

Resolution: To troubleshoot this issue, check your network configuration. If you continue to encounter this error, contact AWS Support. For more information, see AWS Support Center.

HOME_FOLDER_MOUNTING_FAILURE

Message: X session(s) encountered home folder mounting failures.

Resolution: To troubleshoot this issue, check your network configuration. If you continue to encounter this error, contact AWS Support. For more information, see AWS Support Center.

Amazon AppStream 2.0 Service Quotas

AppStream 2.0 provides different resources that you can use. AppStream 2.0 resources include stacks, fleets, images, and image builders. When you create your Amazon Web Services account, we set default quotas (also referred to as limits) on the number of resources that you can create, and on the number of users who can use the AppStream 2.0 service.

To request a quota increase, you can use the Service Quotas console at https://console.aws.amazon.com/servicequotas/. For more information, see Requesting a quota increase in the *Service Quotas User Guide*.

The following table lists the default quotas for each AppStream 2.0 resource and for users in the AppStream 2.0 user pool. The actual quotas for your account could be higher or lower, depending on when you created your account.

Name	Default	Adjustable
Stacks	10	Yes
Fleets	10	Yes
Compute-optimized fleet instances *	• stream.compute.large: 10 • stream.compute.xlarge: 10 • stream.compute.2xlarge: 0 • stream.compute.4xlarge: 0 • stream.compute.8xlarge: 0	Yes
Graphics fleet instances *	• stream.graphics-design.large: 3 • stream.graphics-design.xlarge: 3 • stream.graphics-design.2xlarge: 3 • stream.graphics-design.4xlarge: 0 • stream.graphics-desktop.2xlarge: 0 • stream.graphics-pro.4xlarge: 0 • stream.graphics-pro.8xlarge: 0 • stream.graphics-pro.16xlarge: 0	Yes

Name	Default	Adjustable
	• stream.graphics.g4dn.xlarge: 0 • stream.graphics.g4dn.2xlarge: 0 • stream.graphics.g4dn.4xlarge: 0 • stream.graphics.g4dn.8xlarge: 0 • stream.graphics.g4dn.12xlarge: 0 • stream.graphics.g4dn.16xlarge: 0 • stream.graphics.g5.xlarge: 0 • stream.graphics.g5.2xlarge: 0 • stream.graphics.g5.4xlarge: 0 • stream.graphics.g5.8xlarge: 0 • stream.graphics.g5.12xlarge: 0 • stream.graphics.g5.16xlarge: 0 • stream.graphics.g5.24xlarge: 0	
Memory-optimized fleet instances *	• stream.memory.large: 10 • stream.memory.xlarge: 10 • stream.memory.2xlarge: 0 • stream.memory.4xlarge: 0 • stream.memory.8xlarge: 0 • stream.memory.z1d.large: 10 • stream.memory.z1d.xlarge: 10 • stream.memory.z1d.2xlarge: 0 • stream.memory.z1d.3xlarge: 0 • stream.memory.z1d.6xlarge: 0 • stream.memory.z1d.12xlarge: 0	Yes

Name	Default	Adjustable
Standard fleet instances *	• stream.standard.small: 50 • stream.standard.medium: 50 • stream.standard.large: 50 • stream.standard.xlarge: 10 • stream.standard.2xlarge: 10	Yes
Image builders (total)	10	Yes
Images	10	Yes
Compute-optimized image builder instances	• stream.compute.large: 3 • stream.compute.xlarge: 3 • stream.compute.2xlarge: 0 • stream.compute.4xlarge: 0 • stream.compute.8xlarge: 0	Yes
Graphics image builder instances	• stream.graphics-design.large: 1 • stream.graphics-design.xlarge: 1 • stream.graphics-design.2xlarge: 1 • stream.graphics-design.4xlarge: 0 • stream.graphics-desktop.2xlarge: 0 • stream.graphics-pro.4xlarge: 0 • stream.graphics-pro.8xlarge: 0 • stream.graphics-pro.16xlarge: 0 • stream.graphics.g4dn.xlarge: 0 • stream.graphics.g4dn.2xlarge: 0 • stream.graphics.g4dn.4xlarge: 0 • stream.graphics.g4dn.8xlarge: 0 • stream.graphics.g4dn.12xlarge: 0 • stream.graphics.g4dn.16xlarge: 0	Yes

Name	Default	Adjustable
Memory-optimized image builder instances	• stream.memory.large: 3 • stream.memory.xlarge: 3 • stream.memory.2xlarge: 0 • stream.memory.4xlarge: 0 • stream.memory.8xlarge: 0 • stream.memory.z1d.large: 3 • stream.memory.z1d.xlarge: 3 • stream.memory.z1d.2xlarge: 0 • stream.memory.z1d.3xlarge: 0 • stream.memory.z1d.6xlarge: 0 • stream.memory.z1d.12xlarge: 0	Yes
Standard image builder instances	• stream.standard.small: 5 • stream.standard.medium: 5 • stream.standard.large: 5 • stream.standard.xlarge: 3 • stream.standard.2xlarge: 3	Yes
Number of AWS accounts an image can be shared with	100	Yes
Concurrent image copies per destination Region	2	Yes
Concurrent image updates	5	Yes
Users in the user pool	50	Yes

Name	Default	Adjustable
Max concurrent sessions for Elastic fleets	Amazon Linux 2 • stream.standard.small: 10 • stream.standard.medium: 10 • stream.standard.large: 5 • stream.standard.xlarge: 2 • stream.standard.2xlarge: 2 Windows Server 2019 • stream.standard.small: 10 • stream.standard.medium: 10 • stream.standard.large: 5 • stream.standard.xlarge: 2 • stream.standard.2xlarge: 2	Yes
App block builders (total)	10	Yes
Max number of app block builders	• stream.standard.small: 1 • stream.standard.medium: 1 • stream.standard.large: 1 • stream.standard.xlarge: 1 • stream.standard.2xlarge: 1	Yes

* AppStream 2.0 instance type and size quotas are per AWS account per AWS Region. If you have multiple fleets in the same Region that use the same instance type and size, the total number of instances in all fleets in that Region must be less than or equal to the applicable quota. To determine which instance types are available in which Regions or Availability Zones, see *Pricing by AWS Region – Always-On, On-Demand, app block builders, and image builder instances* in AppStream 2.0 Pricing.

For fleets that have **Default Internet Access** enabled, the quota is 100 fleet instances. If your deployment must support more than 100 concurrent users, use the NAT gateway configuration instead. For more information about enabling internet access for a fleet, see Internet Access.

Guidance for AppStream 2.0 Users

If you are an AppStream 2.0 administrator, you can provide your users with the guidance in this section to help them get started with using AppStream 2.0.

If you are a user who now has access to AppStream 2.0, the topics in this section will help you use AppStream 2.0 for application streaming. With AppStream 2.0, your administrator makes your applications available for you to access remotely, so that you don't have to install the applications on your own device. To access your applications, connect to AppStream 2.0 and start an application streaming session.

Contents

- AppStream 2.0 Access Methods and Clients
- File Storage Options
- Configure Regional Settings

AppStream 2.0 Access Methods and Clients

You can connect to AppStream 2.0 by using a web browser or the AppStream 2.0 client for Windows.

Contents

- Web Browser Access
- AppStream 2.0 Client Application for Windows

Web Browser Access

The following information helps you use a web browser to connect to AppStream 2.0 and stream applications.

Contents

- Requirements
- Setup
- Connect to AppStream 2.0
- AppStream 2.0 Web Browser Access (Version 2)

- Monitors and Display Resolution
- USB Devices
- Touchscreen Devices
- Function Keys
- Remap the Mac Option and Command Keys
- Video and Audio Conferencing
- Drawing Tablets
- Relative Mouse Offset
- Troubleshooting

Requirements

You can connect to AppStream 2.0 from any location by using an HTML5-capable web browser. Supported browsers include the following:

- Google Chrome
- Mozilla Firefox
- Safari
- Microsoft Edge

> **Note**
>
> Only the Google Chrome or Mozilla Firefox browsers are supported for use with drawing tablets during AppStream 2.0 streaming sessions. Webcam redirection for video and audio conferencing is supported on Chromium-based web browsers, including Google Chrome and Microsoft Edge.

Setup

No browser extensions or plugins are required to use AppStream 2.0 in a web browser.

Connect to AppStream 2.0

Follow these steps to connect to AppStream 2.0 and start an application streaming session.

1. If your administrator requires you to sign in first through your organization's sign-in page, complete the tasks in this step.

   If your administrator doesn't require you to sign in through your organization's sign-in page, skip the tasks in this step and proceed to step 2.

   a. Navigate to your organizational sign-in page and enter your domain credentials when prompted.

   b. After you sign in, you are redirected to a page that displays one or more applications that are available for your AppStream 2.0 streaming session. **Desktop View** is also available, if enabled by your administrator.

   c. Choose an application or, if available, **Desktop View**.

2. If your administrator doesn't require you to sign in first through your organization's sign-in page, do either of the following:

   - If this is the first time that you've used AppStream 2.0 and you receive a welcome email that notifies you to start accessing your applications using AppStream 2.0:

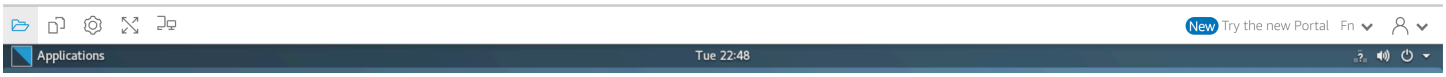
     1. Open the email, and then select the **Login page** link.
     2. Enter your email address and the temporary password that was provided in the email, and then choose **Log in**.
     3. When prompted, enter a new password, confirm it, and then choose **Set Password**.
     4. After a few moments, the AppStream 2.0 portal opens, displaying one or more applications that are available for your AppStream 2.0 streaming session. **Desktop View** is also available, if enabled by your administrator.
     5. Choose an application or, if available, **Desktop View**.

   - If this isn't the first time that you've used AppStream 2.0 and your administrator has provided you with the web address (URL) for the AppStream 2.0 portal:

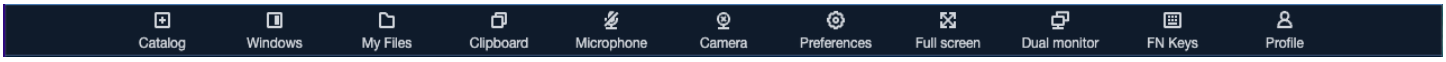
     1. Enter the URL provided by your administrator to navigate to the AppStream 2.0 portal.
     2. Enter your password when prompted, and choose **Connect**.
     3. After a few moments, the AppStream 2.0 portal opens, displaying one or more applications that are available for your AppStream 2.0 streaming session. **Desktop View** is also available, if enabled by your administrator.

AppStream 2.0 Web Browser Access (Version 2)

Your end users can take advantage of an improved streaming experience by choosing the **Profile** menu and **Try the new portal**.



AppStream 2.0 web browser access version 2 offers an enhanced end user experience, including menu options that are easily discoverable and textual guidance for end users. No new menu items have been added, and all configuration choices that were available in the previous version are still available in the new one. These setting options have been reorganized as a result of usability testing conducted by the AppStream 2.0 team.



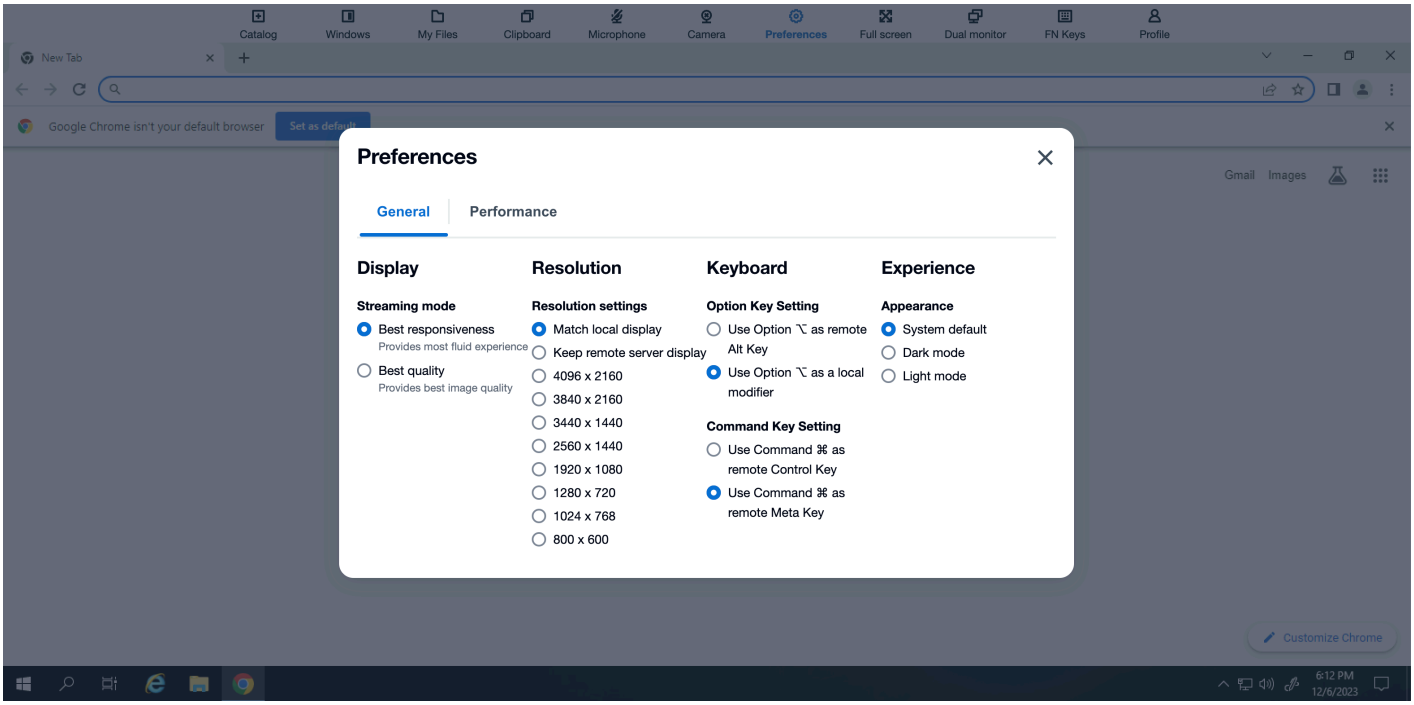
End users can access an enhanced AppStream 2.0 toolbar, plus the following features available under **My files**:

- Download or delete multiple files. Select the file(s), choose **Actions**, and then choose **Delete** or **Download**.
- Upload a folder using drag and drop.
- Sort the files based on **Name**, **Last modified date**, and **Size**.
- Resize the column width.
- Wrap the file name column to accommodate long file names. Choose the small gear icon in the top-right corner, **Wrap lines**, and **Confirm**.

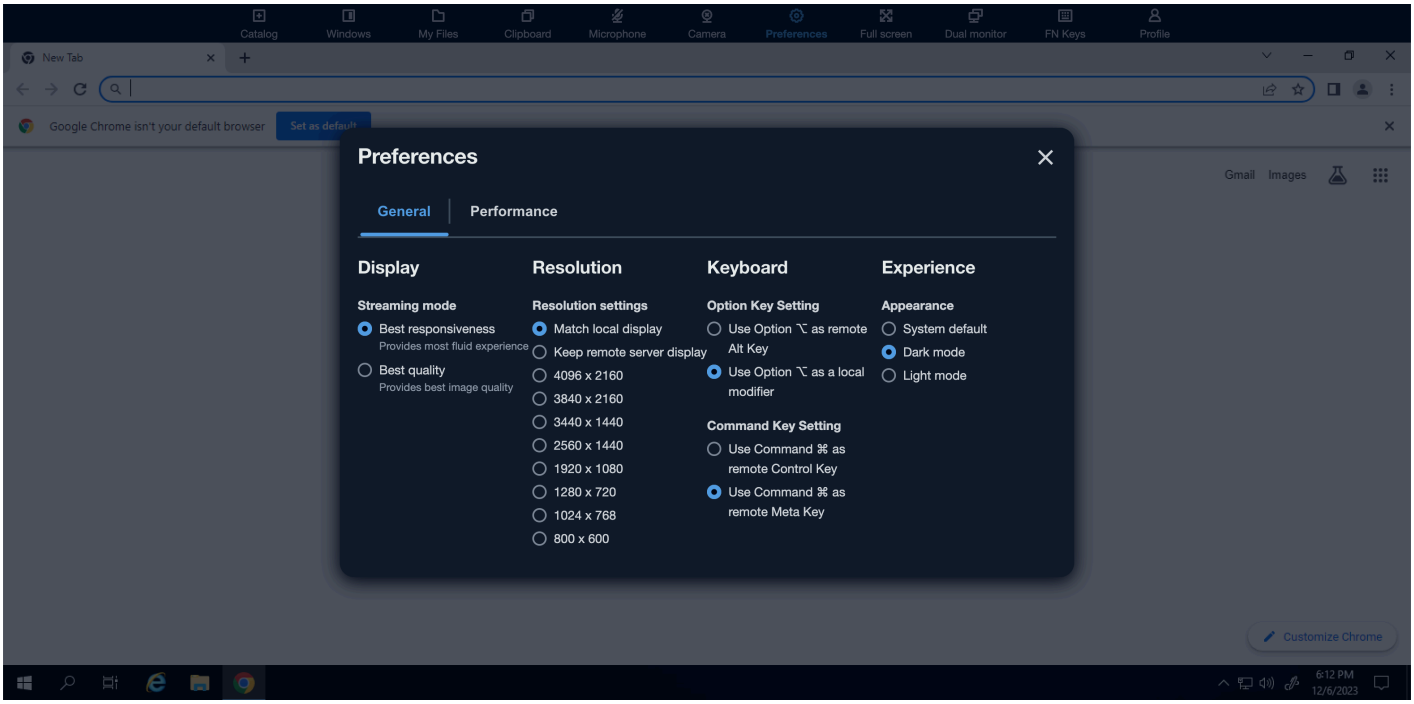
End users can also access the following features:

- To switch between two visual modes (light and dark), choose **Preferences**, **General**, **Experience**, **Appearance**, and **Light mode** or **Dark mode**.

  The following image is an example of light mode.

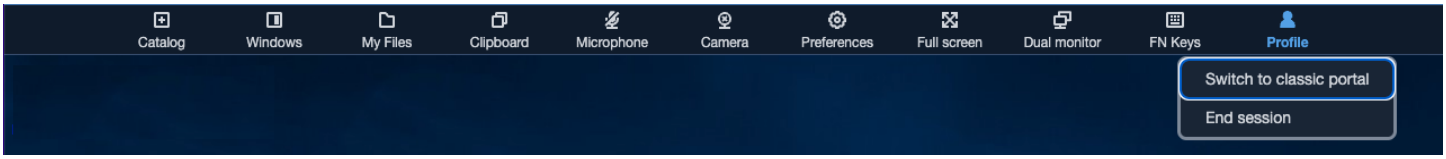


The following image is an example of dark mode.



- To take focus away from the streaming session, and put it on the first element of the toolbar to enable keyboard-based usage during sessions, use the keyboard shortcut **`ctrl + alt + shift + F11`**.

To switch back to AppStream 2.0 web browser access v1, choose the **Profile** menu and **Switch to classic portal**. You will see a prompt to provide feedback about the AppStream 2.0 web browser access v2 experience.



When AppStream 2.0 web browser access v2 becomes the default experience, end users can still switch to the v1 experience. However, we recommend using the v2 experience, which will include all new AppStream 2.0 functionalities and enhancements when it becomes the default.

Monitors and Display Resolution

AppStream 2.0 supports the use of multiple monitors during streaming sessions, including monitors that have different resolutions. To help ensure an optimal streaming experience, we recommend that you set the display scale for your monitors to 100 percent if you use multiple monitors.

You can use dual monitors for application streaming sessions that are started on the following web browsers:

- Google Chrome
- Mozilla Firefox
- Safari
- Microsoft Edge

For browser-based streaming sessions on dual monitors, a maximum display resolution of 2560x1600 pixels is supported per monitor. If you require more than two monitors, or a display resolution that is greater than 2560x1600 pixels per monitor, you must use the AppStream 2.0 client.

USB Devices

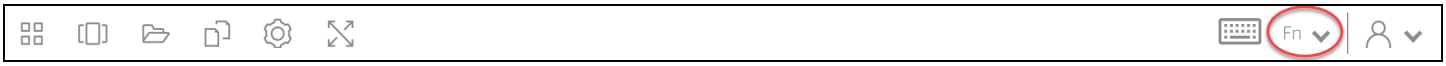
USB devices are not supported for browser-based AppStream 2.0 streaming sessions. To use your USB devices with applications streamed through AppStream 2.0, you must use the AppStream 2.0 client. For more information, see AppStream 2.0 Client Application for Windows.

Touchscreen Devices

AppStream 2.0 supports gestures on touch-enabled iPads, Android tablets, and Windows devices. Examples of supported touch gestures include long-tap to right-click, swipe to scroll, pinch to zoom, and two-finger rotation for supporting applications.

To display the on-screen keyboard on an iPad or Android tablet, tap the keyboard icon on the AppStream 2.0 toolbar. The keyboard icon turns blue, and you can use the on-screen keyboard to input text in the streaming application. Tap the keyboard icon again to hide the on-screen keyboard.

Tap the Fn icon to display a row of Windows-specific keys and keyboard shortcuts.



For touch-enabled devices, the *remote keyboard*, which is displayed when you tap the keyboard icon on the AppStream 2.0 toolbar, is different than the *local keyboard*, the on-screen keyboard that a touch-enabled device automatically displays when you tap inside an input control in a locally running application. During AppStream 2.0 streaming sessions, you can use the remote keyboard to input text into streaming applications only. You can display or hide the remote keyboard only by tapping the keyboard icon on the AppStream 2.0 toolbar. A blue keyboard icon on the AppStream 2.0 toolbar indicates that the remote keyboard is active.

You can use the local keyboard to input text into elements of the AppStream 2.0 web portal, including the **My Files** dialog box. However, you can't use this keyboard to input text into streaming applications. Also, you can't display or hide it by using the keyboard icon on the AppStream 2.0 toolbar.

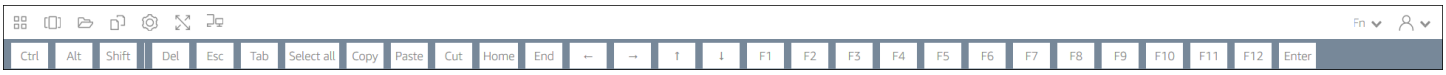
> **Note**
>
> To display the on-screen keyboard on a Windows computer, tap the keyboard icon in the Windows system tray. If the keyboard icon doesn't appear in the Windows system tray, switch to Windows tablet mode. Tap the keyboard icon in the Windows system tray again to hide the on-screen keyboard.

For more information about function keys, see the next section.

Function Keys

You can use keyboard shortcuts during AppStream 2.0 streaming sessions to enter special keystrokes or key combinations. To display a row of Windows-specific keys and keyboard shortcuts during your streaming session, choose the Fn icon (or the FN Keys on AppStream 2.0 web browser access v2). The Fn icon is displayed in the AppStream 2.0 toolbar in the top right of your session window.

Following is an example of how Windows-specific keys and keyboard shortcuts are displayed when you choose the Fn icon (or the FN Keys on AppStream 2.0 web browser access v2). If not all keys are displayed, you can scroll to the right or left on the shortcut toolbar to display more keys.



To use a key combination that includes the Windows Control key, choose the Ctrl key on the shortcut toolbar, and then type any key on the shortcut toolbar (or, if you are using a touch-enabled device, the on-screen keyboard). Choosing the Ctrl key changes the color to blue. In this case, any other key that you select is interpreted as a key combination that includes the Control key.

Choose the Ctrl key again to release it. For example, to use the keyboard shortcut Ctrl + F, choose the Ctrl key on the shortcut toolbar, and then type the f key. Choose the Ctrl key on the shortcut toolbar again to release the Control key. To use shortcuts that include the Alt or Shift keys, choose the Alt key or the Shift key on the shortcut toolbar in the same way. You can use the Shift key on the shortcut toolbar only for keyboard shortcuts. If you are using a touch-enabled device, this key doesn't affect the capitalization of keys that you type on the on-screen keyboard.

Remap the Mac Option and Command Keys

When you use a device that runs macOS or Mac OS X to connect to AppStream 2.0, you can remap the Mac Option and Command keys on your keyboard.

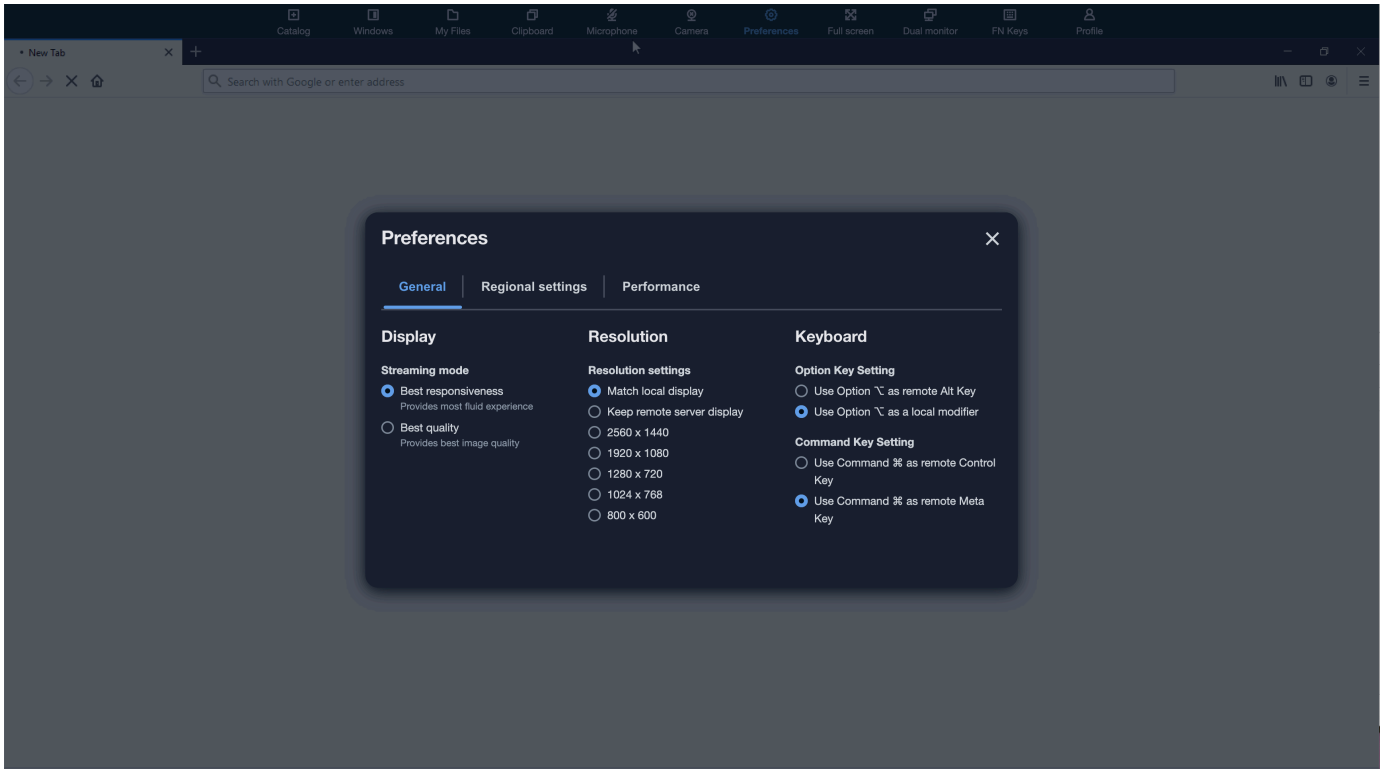
A *modifier key* modifies the action of another key when you use both keys together. You can use a modifier key with another key to perform a task such as printing. A *Meta key* is a special type of modifier key. You can use a Meta key to temporarily change the function of another key when you use both keys together.

You can remap this Mac key	To this key during a streaming session
Option key 	• Remote Alt key • Local modifier key
Command key 	• Remote Control key • Remote Meta key

Follow these steps to remap the Mac Option and Command keys during an AppStream 2.0 streaming session.

To remap the Mac Option and Command keys

1. Use a web browser to connect to AppStream 2.0.
2. In the top left of the AppStream 2.0 toolbar, choose the **Settings** icon, and choose **Keyboard Settings**.
3. Choose the options that correspond to the keys that you want to remap.



Follow these steps to remap the Mac Option and Command keys on AppStream 2.0 web browser access v2.

To remap the Mac Option and Command keys on AppStream 2.0 web browser access v2

1. Use a web browser to connect to AppStream 2.0.
2. From the top menu of the AppStream 2.0 toolbar, choose the **Preferences** menu.
3. Choose **General**, **Keyboard**, and the options that correspond to the keys that you want to remap.

Video and Audio Conferencing

AppStream 2.0 real-time audio-video (AV) redirects your local webcam video and microphone audio input to AppStream 2.0 streaming sessions. That way, you can use your local devices for video and audio conferencing within your AppStream 2.0 streaming session.

To use a local webcam and microphone within an AppStream 2.0 streaming session

1. Connect to AppStream 2.0 from a Chromium-based web browser, including Google Chrome and Microsoft Edge.

> **Note**
>
> Most popular HTML5 compatible browsers support audio input in an AppStream 2.0 session, including Chrome, Edge, and Firefox.

> **Note**
>
> If your web browser doesn't support video or audio input, the options won't appear on the AppStream 2.0 toolbar.

2. Configure your web browser's camera and microphone permissions to set default devices and allow access to AppStream 2.0.

> **Note**
>
> For information about how to configure Google Chrome, see Use your camera & microphone.

3. In the top-left of the AppStream 2.0 toolbar, choose the Settings icon, and then choose **Enable webcam**. For AppStream 2.0 web browser access v2, choose the **Camera** option from the AppStream 2.0 toolbar (which turns the option blue).

> **Note**
>
> If the microphone or webcam icons don't appear in the **Settings** menu, contact your AppStream 2.0 administrator. Your web browser might not support video or audio input, or your administrator might need to perform additional configuration tasks. For more information, see the section called "Real-Time Audio-Video (Client for Windows)".

4. Depending on your web browser settings, you might be prompted to allow the camera to be used by your web browser. Choose **Allow** to enable your camera.
5. In the top-left of the AppStream 2.0 toolbar, choose the Settings icon, and then choose Enable microphone. For AppStream 2.0 web browser access v2, choose the **Microphone** option from the AppStream 2.0 toolbar (which turns the option blue).

6. Depending on your web browser settings, you might be prompted to the microphone to be used by your web browser. Choose **Allow** to enable your microphone.

> **Note**
>
> If you have more than one webcam or microphone and want to change the devices that you use for streaming within an AppStream 2.0 session, you must clear your web browser settings for the AppStream 2.0 website URL and configure default devices. Then, refresh your browser or start a new session for the changes to take effect, and repeat the above steps to enable your webcam and microphone.

Drawing Tablets

Drawing tablets, also known as pen tablets, are computer input devices that let you draw with a stylus (pen). With AppStream 2.0, you can connect a drawing tablet, such as a Wacom drawing tablet, to your local computer and use the tablet with your streaming applications.

Following are requirements and considerations for using drawing tablets with your streaming applications.

- To use this feature, you must connect to AppStream 2.0 through the Google Chrome or Mozilla Firefox browsers only, or by using the AppStream 2.0 client.
- The applications that you stream must support Windows Ink technology. For more information, see Pen interactions and Windows Ink in Windows apps.
- Depending on the streaming applications that you use, your drawing tablet might require USB redirection to function as expected. This is because some applications, such as GIMP, require USB redirection to support pressure sensitivity. If this is the case for your streaming applications, you must connect to AppStream 2.0 by using the AppStream 2.0 client, and share the drawing tablet with your streaming session. For information about how to share USB devices with your streaming session, see USB Devices.
- This feature is not supported on Chromebooks.

To get started with using a drawing tablet during your application streaming sessions, connect your drawing tablet to your local computer with USB, share the device with AppStream 2.0 if required for pressure sensitivity detection, and then start an AppStream 2.0 streaming session. You can use a supported web browser or the AppStream 2.0 client, if it is installed, to start a streaming session.

Relative Mouse Offset

By default, during a streaming session, AppStream 2.0 transmits information about mouse movements by using absolute coordinates and rendering the mouse movements locally. For graphics-intensive applications, such as computer-aided design (CAD)/computer-aided manufacturing (CAM) software or video games, mouse performance improves when relative mouse mode is enabled. Relative mouse mode uses relative coordinates, which represent how far the mouse moved since the last frame, rather than the absolute x-y coordinate values within a window or screen. When you enable relative mouse mode, AppStream 2.0 renders the mouse movements remotely.

You can enable this feature during an AppStream 2.0 streaming session by doing either of the following:

- Windows: Pressing Ctrl+Shift+F8
- Mac: Pressing Control+Fn+Shift+F8

Troubleshooting

If issues occur when you use AppStream 2.0, your AppStream 2.0 session ID can help your administrator with troubleshooting. This section describes how to find the session ID.

The session ID is created when you request a streaming session. The session ID, and other information used by AppStream 2.0, is stored in the session storage location for your browser. You can use the developer tools that are available for your browser interface to find this location.

For information about the developer tools that are available for common web browsers, see the following resources:

- Apple Safari Developer Help: Storage tab
- View And Edit Session Storage With Chrome DevTools
- Firefox Developer Tools: Local Storage / Session Storage
- Microsoft Edge (Chromium) Developer Tools
- Microsoft Edge (EdgeHTML) Developer Tools

After you locate the developer tools for your browser, search for the session storage for the AppStream 2.0 website. The domain for the website is **https://appstream2.<*aws-***

***region*>.aws.amazon.com**. Expand the domain, and choose **sessionStorage.as2SessionData**. The session ID is stored in the key **sessionId**.

AppStream 2.0 Client Application for Windows

The following information helps you use the AppStream 2.0 client for Windows to connect to AppStream 2.0 and stream applications.

Contents

- Features
- Requirements
- Setup
- Connect to AppStream 2.0
- Monitors and Display Resolution
- USB Devices
- Local File Access
- Printer Redirection
- Video and Audio Conferencing (Client for Windows)
- Drawing Tablets
- Relative Mouse Offset
- Logging
- Troubleshooting

Features

The AppStream 2.0 client for Windows is an application that you install on your Windows PC. This application provides additional capabilities that are not available when you access AppStream 2.0 by using a web browser. For example, the AppStream 2.0 client lets you do the following:

- Use more than two monitors or 4K resolution.
- Use your USB devices with applications streamed through AppStream 2.0.
- Access your local drives and folders during your streaming sessions.
- Redirect print jobs from your streaming application to a printer that is connected to your local computer.

- Use your local webcam for video and audio conferencing within your streaming sessions.
- Use keyboard shortcuts during your streaming sessions.
- Interact with your remote streaming applications in much the same way as you interact with locally installed applications.

Requirements

The AppStream 2.0 client for Windows must be installed on a computer that meets the following requirements:

- Operating system — Windows 10 (32-bit or 64-bit), Windows 11 (64-bit)
- Microsoft Visual C++ 2019 Redistributable or later for AppStream 2.0 client version 1.1.1066 and above. For information about the latest Visual C++ redistributable packages for Visual Studio 2015, 2017, and 2019, see The latest supported Visual C++ downloads in the Microsoft Support documentation.
- RAM — 2 GB minimum
- Hard drive space — 200 MB minimum

In addition, to install the AppStream 2.0 USB driver for USB driver support, you must have local administrator rights on your PC.

Setup

Follow these steps to install the client.

1. On the PC where you want to install the AppStream 2.0 client, download the AppStream 2.0 client for Windows application from AppStream 2.0 supported clients.
2. Navigate to the location where you downloaded the application .exe file, and then double-click the file to begin the installation.

> **Important**
>
> Contact your network administrator if nothing happens when you double-click the file or if an error message is displayed. Your organization might be using antivirus software that prevents the AppStream 2.0 client installation program from running.

3. If a message notifies you that Windows Defender SmartScreen prevented an unrecognized app from starting, choose the **More info** link. For **App**, verify that **AmazonAppStreamClientSetup_**<*version-number*>**.exe** is displayed, and then choose **Run anyway**.
4. The installation wizard displays links to the AWS Customer Agreement, AWS Service Terms, and the AWS Privacy Notice, and third-party notices. Review this information, and then choose **Next**.
5. On the **Client Diagnostics** page, to enable the AppStream 2.0 client to automatically upload device logs to help with troubleshooting issues, keep **Client logging** selected, and then choose **Next**.
6. On the **Optional Components** page, to enable your USB devices to be used with streaming applications, select the **AppStream 2.0 Client USB Driver** check box, and then choose **Finish**.
7. If the **AppStream 2.0 USB driver** wizard setup wizard opens, choose **Install**.
8. If prompted by **User Account Control** to choose whether to allow the app to make changes to your device, choose **Yes**.
9. When a message notifies you that the USB driver installation is complete, choose **Close**.

   The AppStream 2.0 sign-in page opens. For information about how to connect to AppStream 2.0 and start an application streaming session, see Connect to AppStream 2.0.

Connect to AppStream 2.0

After the AppStream 2.0 client for Windows is installed on your PC, you can use it to connect to AppStream 2.0.

AppStream 2.0 Client Connection Modes

The AppStream 2.0 client provides two connection modes: *Native application mode* and *classic mode*. The connection mode that you choose determines the options that are available to you during application streaming, and how your streaming applications function and display. In addition, **Desktop view** is also available, if your administrator has enabled it.

Native application mode

Native application mode lets you work with remote streaming applications in much the same way that you work with applications that are installed on your local PC.

When you connect to AppStream 2.0 in native application mode, the AppStream 2.0 Application Launcher window opens and displays the list of applications that are available for you to stream. When you open a streaming application in this mode, the AppStream 2.0 Application Launcher window remains open, and the application opens in its own window. During your streaming session, the remote streaming application functions in much the same way as a locally installed application. The application icon is displayed in the taskbar of your local PC, just as the icons do for your local applications. Unlike the icons for your local applications, the icons for your streaming applications in native application mode include the AppStream 2.0 logo.

During your AppStream 2.0 streaming session, you can switch quickly between your locally installed applications and your remote streaming applications by clicking the taskbar icon of the remote or local application you want to work with. You can also switch AppStream 2.0 connection modes. If you want to work in classic mode instead, you can switch from native application mode to classic mode.

Classic mode

When you use classic application mode, you work with remote streaming applications in the AppStream 2.0 session window. If your administrator has made more than one application available to you, you can open multiple applications during your session. All applications that you open are displayed in the same AppStream 2.0 session window.

When you connect to AppStream 2.0 in classic mode, the AppStream 2.0 Application Launcher window opens and displays the list of applications that are available for you to stream. When you open a streaming application in this mode, the Application Launcher window closes, and the application opens in the AppStream 2.0 session window.

If your administrator has not disabled native application mode, you can switch from classic mode to native application mode. For more information, see How to Switch AppStream 2.0 Connection Modes.

Desktop view

When you connect to AppStream 2.0 and choose **Desktop view**, AppStream 2.0 provides a standard Windows desktop view for your streaming session. The icons of applications that are available for you to stream appear on the Windows desktop. In addition, the AppStream 2.0 toolbar, which enables you to configure settings for your streaming session, appears in the top left area of your streaming session window.

Connect to AppStream 2.0

Follow these steps to connect to AppStream 2.0 and start an application streaming session.

1. If your administrator requires you to sign in first through your organization's sign-in page, complete the tasks in this step, then proceed to step 3.

   If your administrator doesn't require you to sign in through your organization's sign-in page, skip the tasks in this step and proceed to step 2.

   a. Navigate to your organizational sign-in page and enter your domain credentials when prompted.

   b. After you sign in, you are redirected to the AppStream 2.0 portal, which displays one or more applications that are available for your AppStream 2.0 streaming session. **Desktop View** is also available, if enabled by your administrator.

   c. Choose an application or, if available, **Desktop View**.

2. If your administrator doesn't require you to sign in first through your organization's sign-in page, complete the following steps to start the AppStream 2.0 client:

   a. On your local computer where the AppStream 2.0 client is installed, in the lower left of your screen, choose the Windows search icon on the taskbar and enter **`AppStream`** in the Search box.

   b. In the search results, select **Amazon AppStream** to start the AppStream 2.0 client.

   c. On the AppStream 2.0 client sign-in page, you can choose whether to use the client in native application mode or classic mode.

      - To use native application mode, keep the **Start in native application mode** check box selected.
      - To use classic mode, clear the **Start in native application mode** check box.

   d. Do either of the following:

      - If the client sign-in page is prepopulated with a web address (URL), choose **Connect**.
      - If the client sign-in page is not prepopulated with a URL, enter the URL that your AppStream 2.0 administrator provided for AppStream 2.0, and then choose **Connect**. If you don't know the URL, contact your administrator.

   e. After a few moments, the AppStream 2.0 portal opens, displaying one or more applications that are available for your AppStream 2.0 streaming session. **Desktop View** is also available, if enabled by your administrator.
   f. Choose an application or, if available, **Desktop View**.
3. Depending on the authentication settings that your AppStream 2.0 administrator enabled, after you choose an application or **Desktop View**, you might be prompted to enter your Active Directory domain credentials to sign in to your AppStream 2.0 session. If this is the case, do one of the following:
   - If your organization has enabled password authentication, enter your Active Directory domain password, and then choose **Password sign in**.
   - If your organization has enabled smart card authentication, select **Choose a smart card**, follow the instructions to choose your smart card certificate and enter your smart card PIN, and then choose **Smart card sign in**.
   - If your organization has enabled both authentication methods, either enter your Active Directory domain password and choose **Password sign in**, or select **Choose a smart card**, and follow the instructions to complete the smart card sign-in.

How to Switch AppStream 2.0 Connection Modes

If your administrator has not disabled native application mode for your streaming sessions, you can switch between native application mode and classic mode.

To switch from native application mode to classic mode

1. In the upper left of the AppStream 2.0 Application Launcher window, choose the **Settings** icon, and then choose **Switch to classic mode**.
2. When you switch to classic mode, the Application Launcher window closes and the AppStream 2.0 session window opens. Any application that you are streaming in native application mode opens in the AppStream 2.0 session window.

Follow these steps to switch from classic mode to native application mode.

To switch from classic mode to native application mode

1. In the upper left of the AppStream 2.0 session window, choose the **Settings** icon, and then choose **Switch to native application mode**.

2. When you switch from classic mode back to native application mode, the AppStream 2.0 session window closes and the AppStream 2.0 Application Launcher window opens. Any application that you are streaming in classic mode opens in a separate window.

Monitors and Display Resolution

AppStream 2.0 supports the use of multiple monitors during streaming sessions, including monitors that have different resolutions. To help ensure an optimal streaming experience, we recommend that you set the display scale for your monitors to 100 percent if you use multiple monitors.

The AppStream 2.0 client supports multiple monitors with the following display resolutions:

- Multiple monitors (up to 2K resolution) — Up to 4 monitors with a maximum display resolution of 2560x1600 pixels per monitor
- Multiple monitors (4K resolution) — Up to 2 monitors with a maximum display resolution of 4096x2160 pixels per monitor

> **Note**
>
> If you are connected to an AppStream 2.0 streaming session using native application mode, you can use monitors with up to 2K resolution. If you use higher-resolution monitors, the AppStream 2.0 client falls back to classic mode. In this case, the AppStream 2.0 classic mode streaming view occupies 2K of the screen, and the remaining portion of the screen is black.

USB Devices

With certain exceptions, USB redirection is required for the AppStream 2.0 client to support USB devices. When USB redirection is required for a device, you must share the device with AppStream 2.0 every time you start a new streaming session.

How to Use a Smart Card During a Streaming Session

Depending on the authentication settings that your administrator has enabled, you might need to use a smart card for authentication during an AppStream 2.0 streaming session. For example, if you

open a browser during your streaming session and navigate to an internal organizational site that requires smart card authentication, you must enter your smart card credentials.

By default, smart card redirection is enabled for AppStream 2.0 streaming sessions, which means that you can use the smart card reader that is attached to your local computer without sharing it with AppStream 2.0. During your streaming session, your smart card reader and smart card are available for you to use with local applications, as well as with streaming applications.

If your administrator has disabled smart card redirection, you must share your smart card reader with AppStream 2.0. For more information, see the next section.

How to Share a USB Device with AppStream 2.0

If you are using a drawing tablet, USB redirection might not be required to use it with AppStream 2.0. However, if you are streaming an application such as the Gnu Image Manipulation Program (GIMP), which requires USB redirection to support pressure sensitivity, you must share your drawing tablet with AppStream 2.0. For information about drawing tablets, see Drawing Tablets.

To share a USB device with AppStream 2.0

1. Use the AppStream 2.0 client to start a streaming session.
2. In the top left area, choose the **Settings** icon, and then choose **USB Devices**.
3. If your USB device is connected to your computer, the USB device name appears in the dialog box. If your USB device is not detected, contact your AppStream 2.0 administrator for assistance.
4. Switch the **Share** toggle key next to the name of the USB device that you want to share with the streaming session.

   Your USB device is now available for use with your streaming applications.

   > **Important**
   >
   > USB devices can't be simultaneously used between local and remote applications. So after you share a USB device with a streaming session, you can't use it with applications on your local computer. To use your USB device on your local computer, switch the **Share** toggle key next to the name of the USB device that you want to use locally. This disables sharing with the streaming session.

5. You can also enable your USB device to automatically connect when a new streaming session starts. To do so, select the option next to the toggle key for the USB device that you want to connect. After you enable this option, when your next streaming session starts, the USB device is connected automatically.

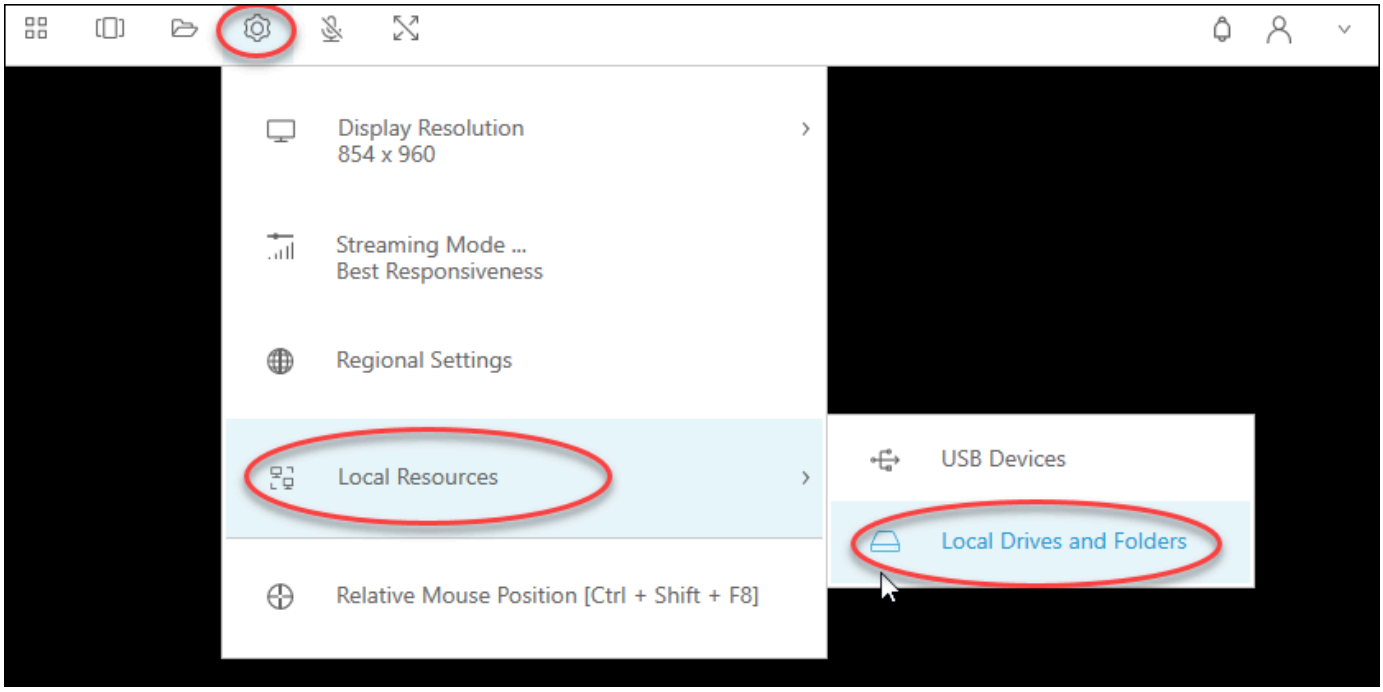
Local File Access

AppStream 2.0 file redirection lets you access files on your local computer from your AppStream 2.0 streaming session. To use file redirection, open the AppStream 2.0 client, connect to a streaming session, and choose the drives and folders that you want to share. After you share a local drive or folder, you can access all files in the shared drive or folder from your streaming session. You can stop sharing local drives and folders at any time.

> **Important**
>
> To use AppStream 2.0 file redirection, you must have the AppStream 2.0 client installed on your local computer. File redirection is not available when you connect to AppStream 2.0 by using a web browser.

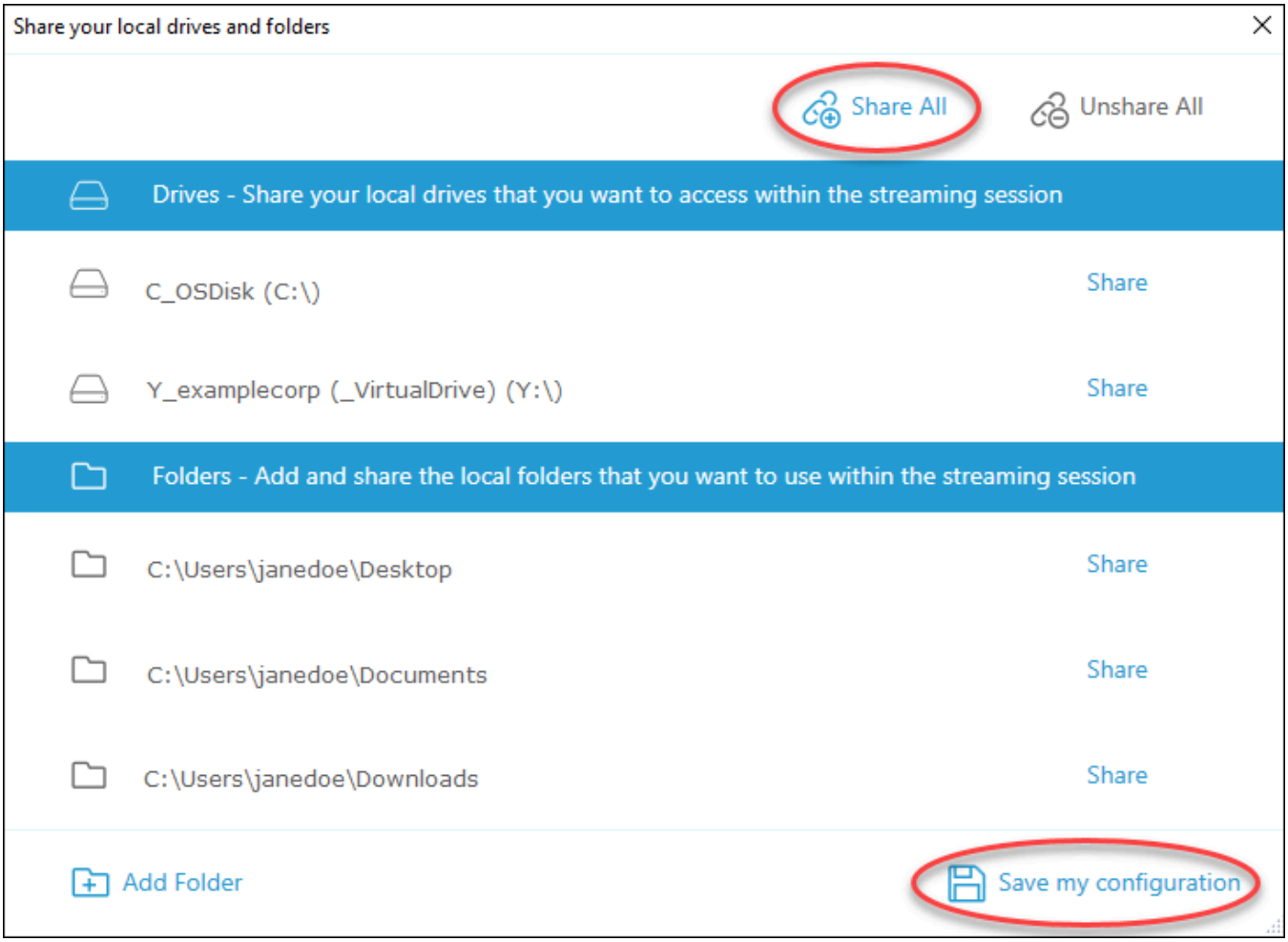
To share local drives and folders

1. Open the AppStream 2.0 client and connect to a streaming session.
2. In your AppStream 2.0 session, in the top left area, choose the **Settings** icon, and then choose **Local Resources**, **Local Drives and Folders**.

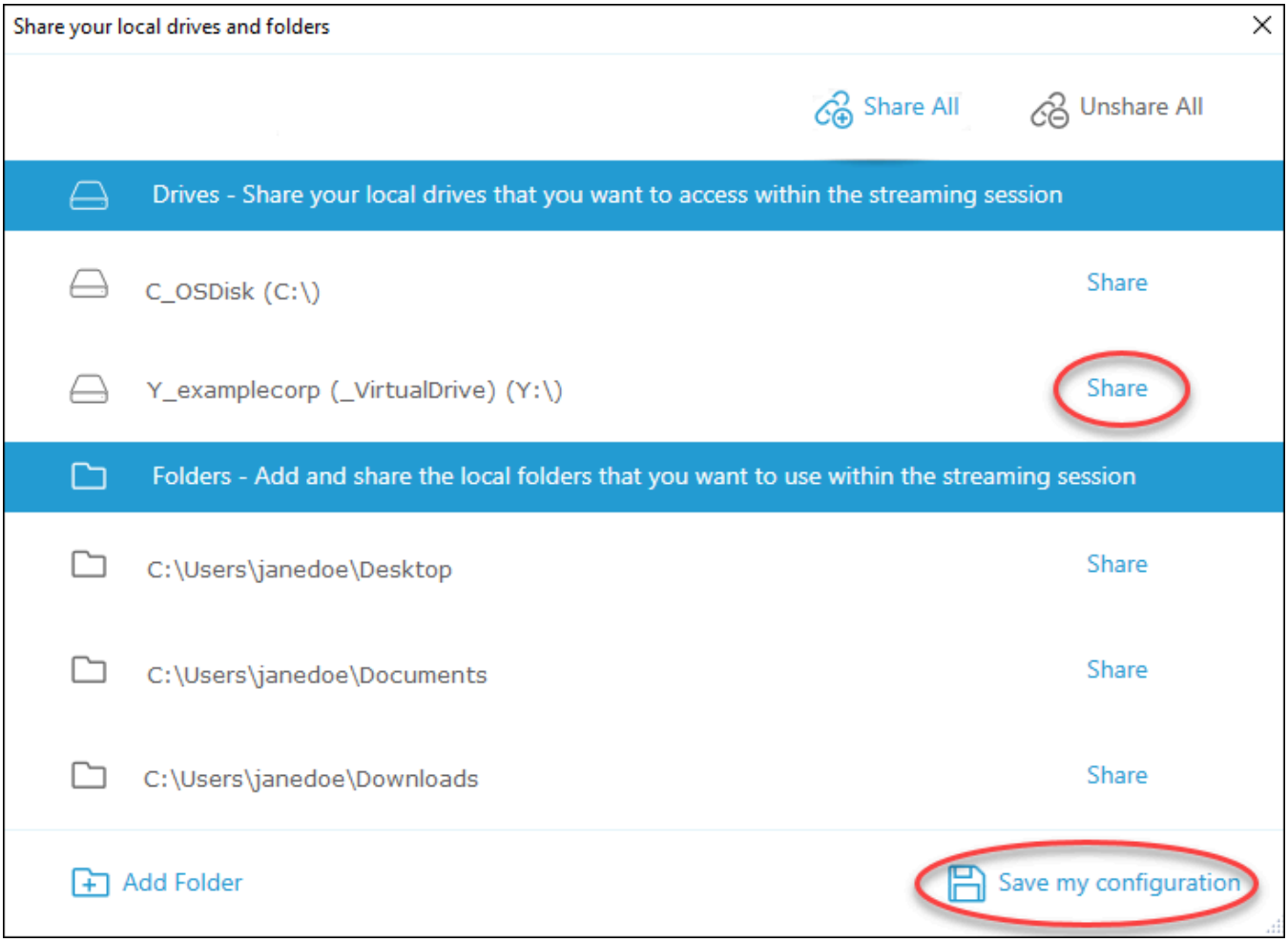


The **Share your local drives and folders** dialog box displays the drives and folders that your administrator has made available for you to share. You can share all or specific drives and folders, or just one. You can also add your own drives and folders. To share drives and folders, do one of the following:

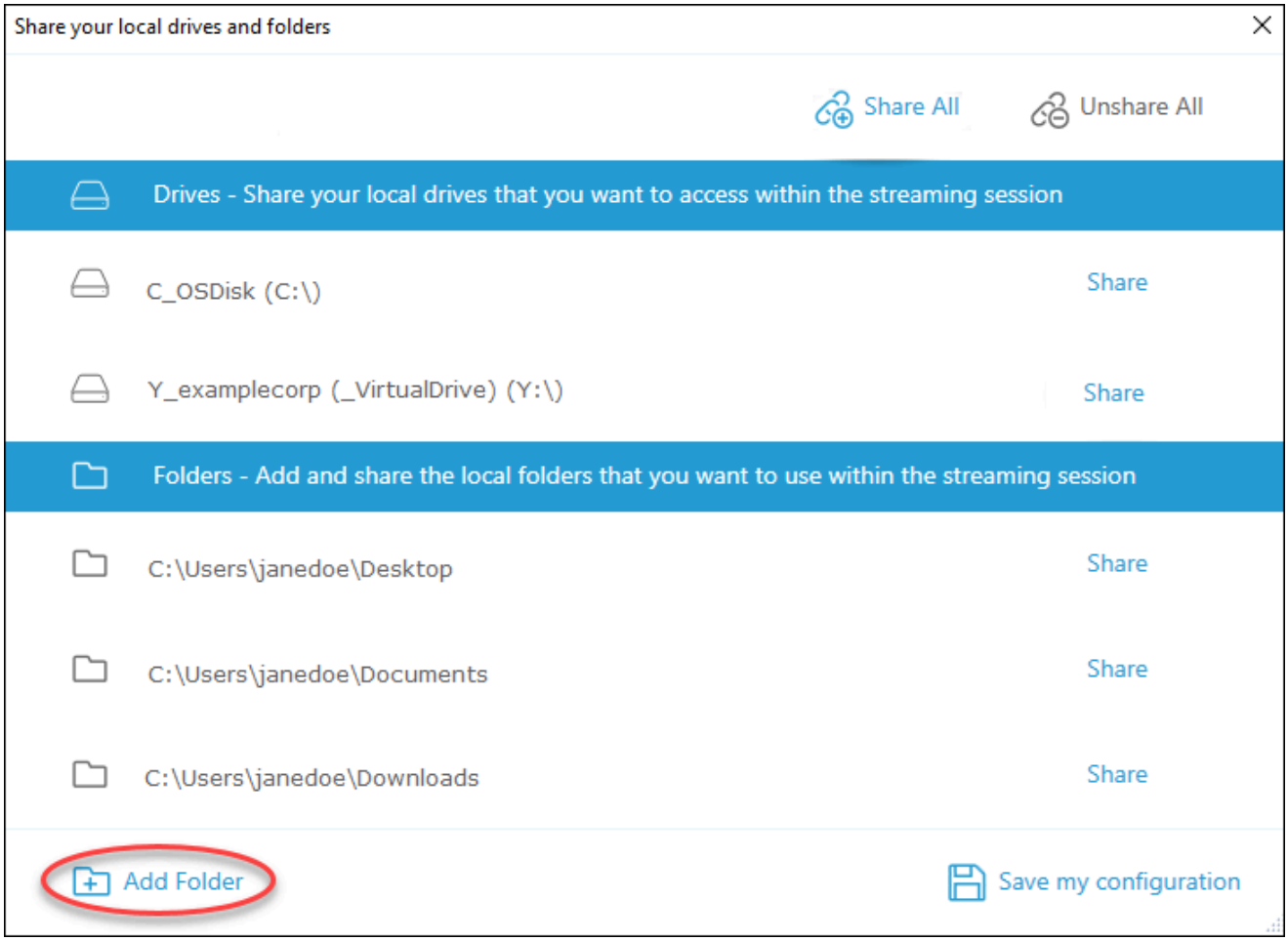
- To share all local drives and folders displayed in the **Share your local drives and folders** dialog box, choose **Share All**. To apply your changes to future streaming sessions, choose **Save my configuration**.



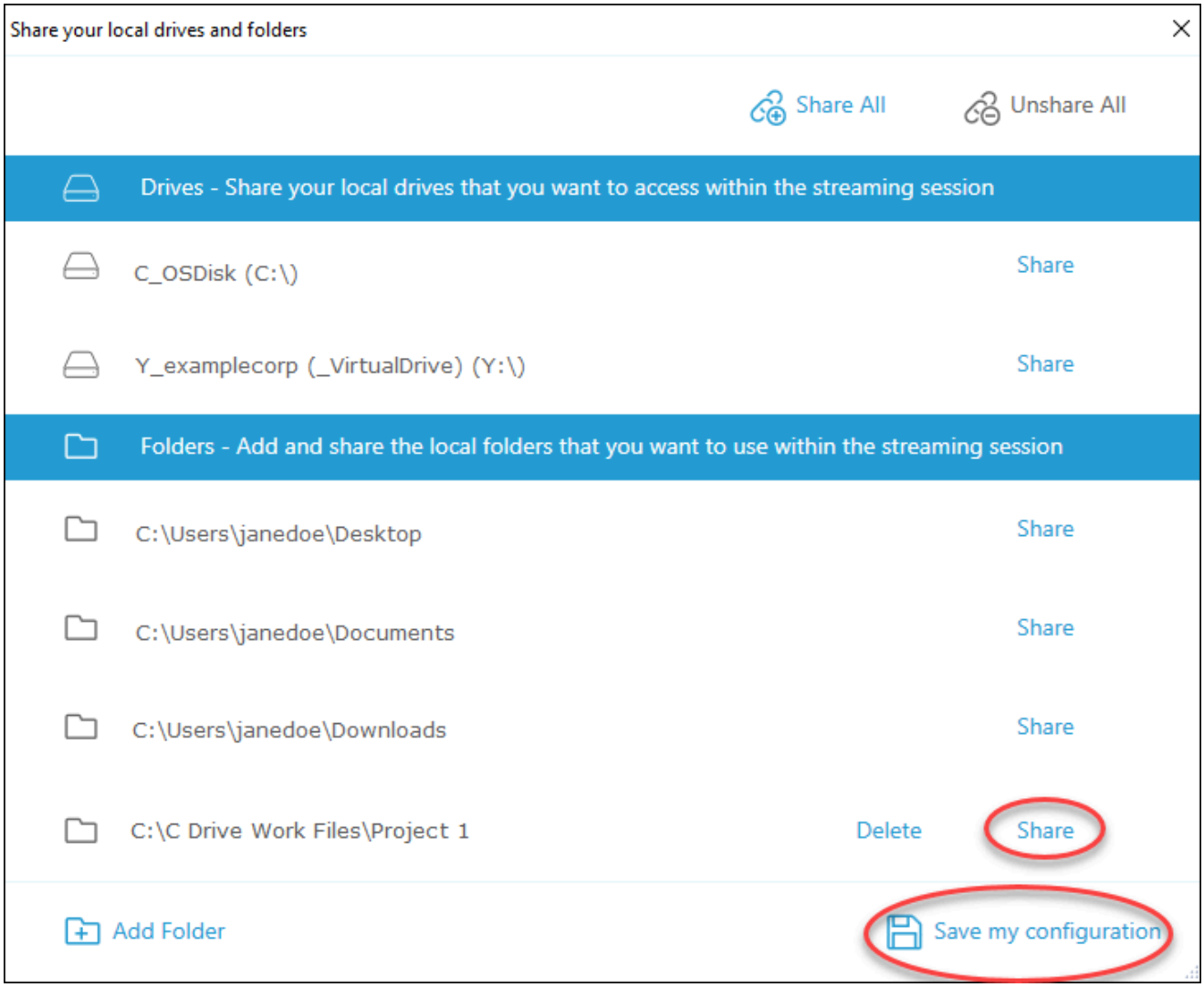
- To share a specific local drive or folder, select the drive or folder that you want to access, and choose **Share**, **Save my configuration**. To share another local drive or folder, repeat these steps as needed.



- If the local drive or folder that you want to share is not displayed, you can add it. For example, your administrator might make your entire local C Drive available for you to share. However, you might only need to access a specific folder on that drive. In this case, you can add the folder that you need and share only that folder. To choose a folder, do the following:
  - In the **Share your local drives and folders** dialog box, choose **Add Folder**.



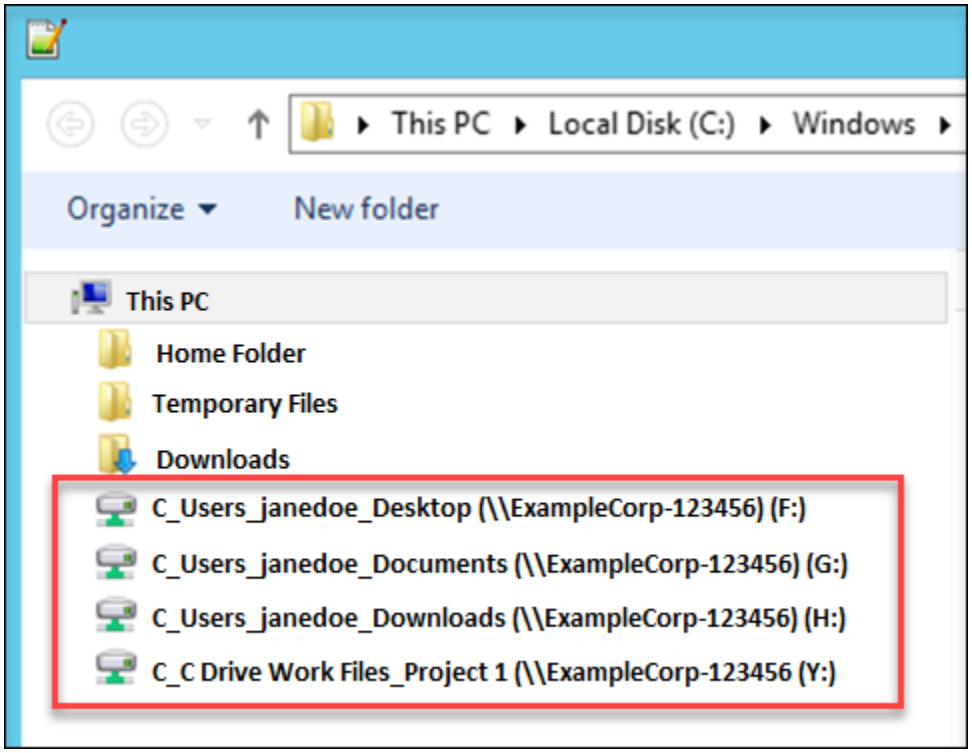
- Browse to the folder that you want to share, and choose **OK**.
- The folder that you selected is now available to share. Select the folder, and choose **Share**, **Save my configuration**. To add another local drive or folder, repeat these steps as needed.



After you share a local drive or folder, perform the following steps to access files in the shared drive or folder from your streaming session.

To access files in a shared local drive or folder

1. Open the AppStream 2.0 client and connect to a streaming session.
2. In your AppStream 2.0 session, open the application that you want to use.
3. From your application interface, choose **File Open**, and browse to the file that you want to access. The following screenshot shows how shared local drives and folders appear in the Notepad++ browse dialog box for Jane Doe when she browses for a file.



In the browse dialog box, the corresponding paths for her shared drives and folders are shown in the red box. The paths appear with backslashes replaced by underscores. At the end of each path is the name of Jane's computer, ExampleCorp-123456, and a drive letter.

4. When you're done working with the file, use the **File Save** or **File Save As** command to save it to the location that you want.

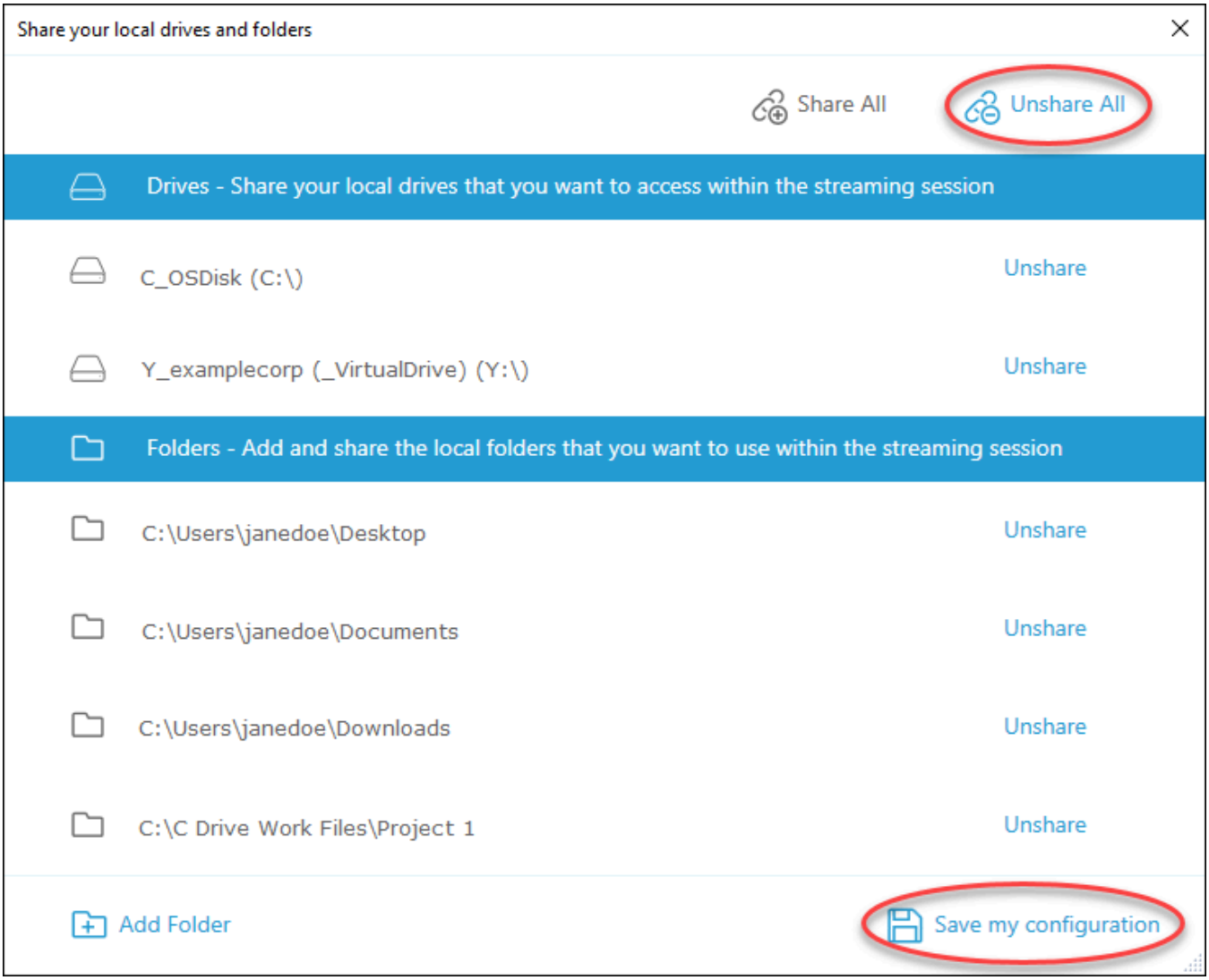
If you want to stop sharing a local drive or folder, perform the following steps.

To stop sharing local drives and folders

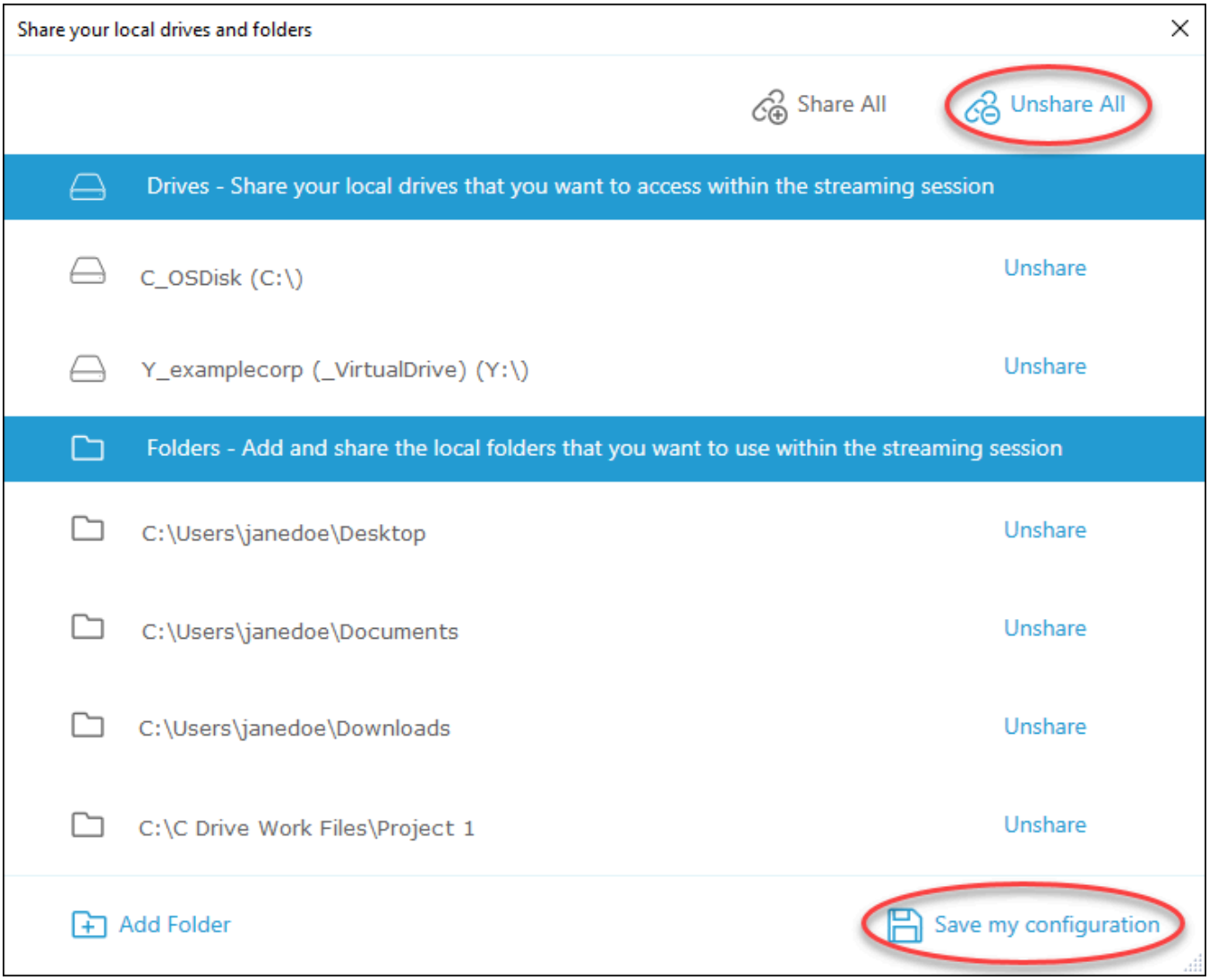
1. Open the AppStream 2.0 client and connect to a streaming session.
2. In your AppStream 2.0 session, in the top left area, choose the **Settings** icon, and then choose **Local Resources**, **Local Drives and Folders**.

   The **Share your local drives and folders** dialog box displays the drives and folders that your administrator has made available for you to share, and any that you added, if applicable. To stop sharing one or more local drives and folders, do either of the following:

   - To stop sharing all shared local drives and folders, choose **Unshare All**, **Save my configuration**.



- To stop sharing a specific shared local drive or folder, select the drive or folder, and choose **Unshare**, **Save my configuration**. To stop sharing another local drive or folder, repeat these steps as needed.



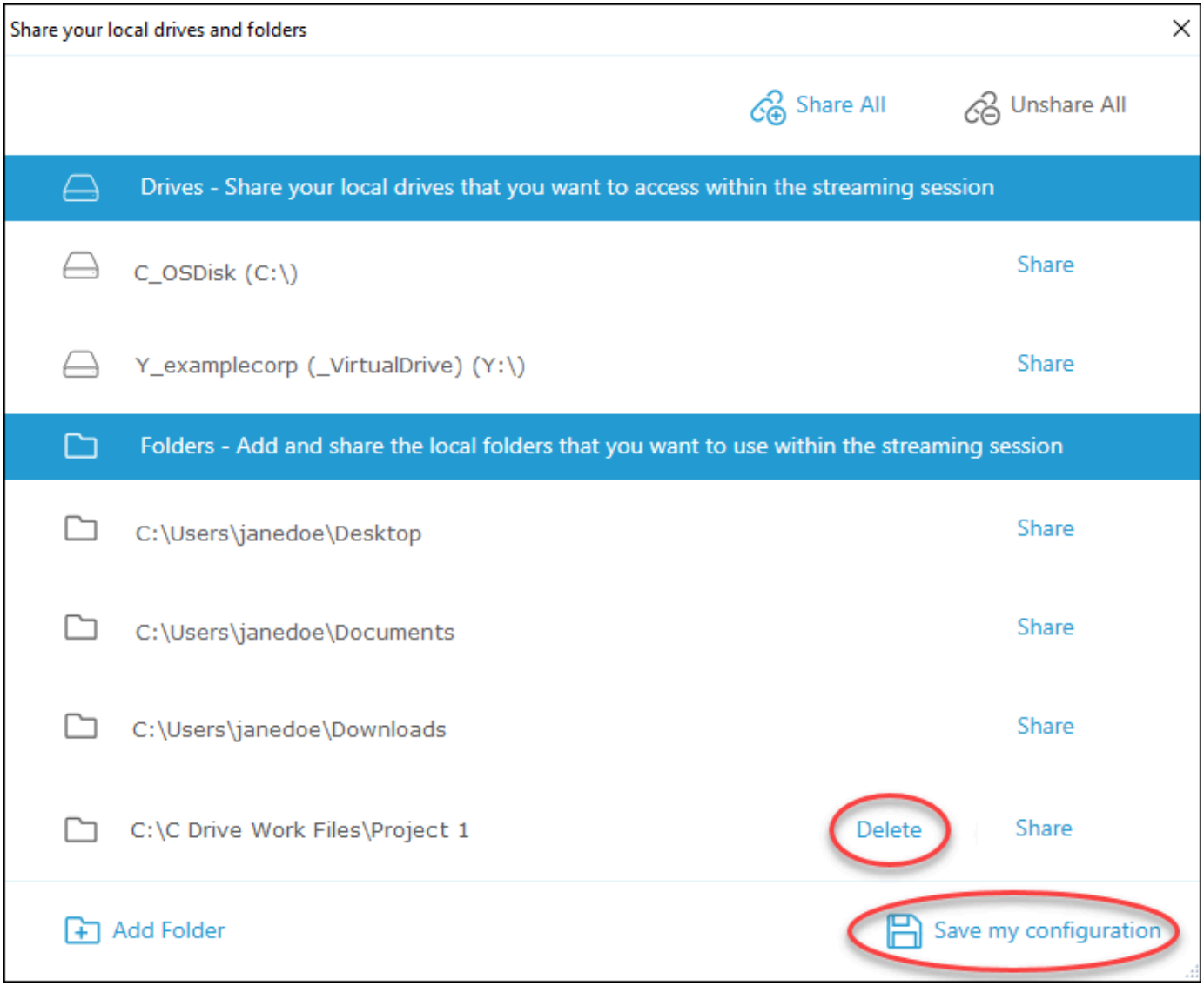
You can delete local drives and folders that you add to the **Share your local drives and folders** dialog box. However, you can't delete local drives or folders that your administrator has made available for you to share. Also, if you have already shared a local drive or folder, you must stop sharing it before you can delete it.

To delete local drives and folders

1. Open the AppStream 2.0 client and connect to a streaming session.
2. In your AppStream 2.0 session, in the top left area, choose the **Settings** icon, and then choose **Local Resources**, **Local Drives and Folders**.

The **Share your local drives and folders** dialog box displays the drives and folders that your administrator has made available for you to share. If you added any drives or folders, they are also displayed.

3. Select the local drive or folder that you want to delete, and then choose **Delete**, **Save my configuration**.



Printer Redirection

AppStream 2.0 local printer redirection lets you access printers that are connected to your local computer from your AppStream 2.0 streaming session. That way, you can redirect print jobs from your streaming application to a local printer, or to a network printer that you have mapped.

> **Important**
>
> To use AppStream 2.0 printer redirection, you must have the AppStream 2.0 client installed on your local computer, and you must use the client to connect to a streaming session. Printer redirection is not available when you connect to AppStream 2.0 by using a web browser.

To redirect a print job to a local printer

1. Open the AppStream 2.0 client and connect to a streaming session.
2. In your streaming application, choose **File**, **Print Now**.
3. In the top-right area of the AppStream 2.0 session window, select the new notification that appears next to the notification icon.
4. In the **Notifications** dialog box, choose the **Print Jobs** tab.
5. On the **Print Jobs** tab, choose **Print**.
6. The **Print** dialog box for your streaming application opens.
7. In the **Print** dialog box, a list of available local printers is displayed. Choose the local printer that you want to use, and then proceed with printing.

Video and Audio Conferencing (Client for Windows)

AppStream 2.0 real-time audio-video (AV) redirects your local webcam video input to AppStream 2.0 streaming sessions. That way, you can use your local devices for video and audio conferencing within your AppStream 2.0 streaming session.

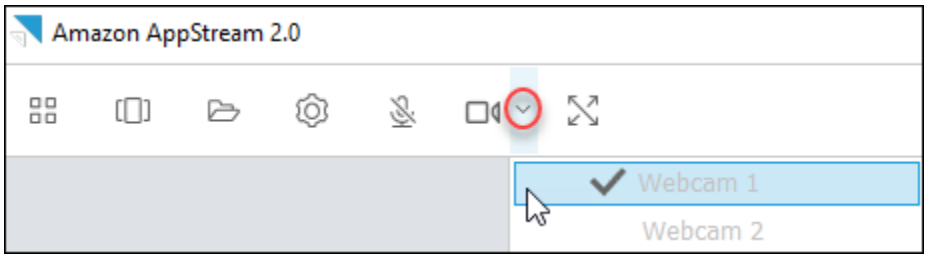
To use a local webcam and microphone within an AppStream 2.0 streaming session

1. Open the AppStream 2.0 client and connect to a streaming session.
2. In the AppStream 2.0 toolbar in the top left of your session window, do either of the following:
   - If the video icon has a diagonal line through it (as shown in the following screenshot), this indicates that the AppStream 2.0 real-time AV feature is available for use but no webcams are attached to your streaming session. Choose the video icon to attach one or more webcams.

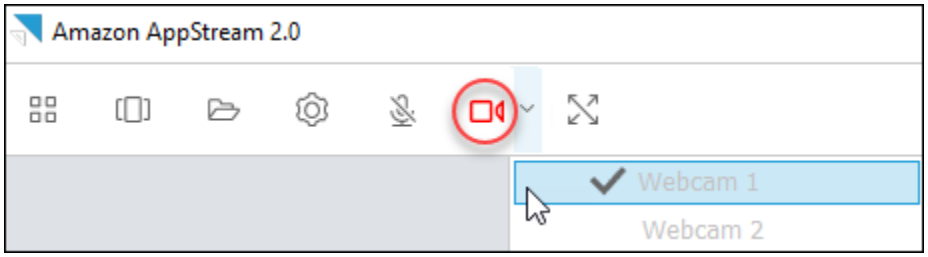
- If the video icon does not have a diagonal line through it (as shown in the following screenshot), one or more webcams are already attached to your streaming session. Skip this step and proceed to the next step.

> **Note**
>
> If the video icon doesn't display in the AppStream 2.0 toolbar, contact your AppStream 2.0 administrator. Your administrator might need to perform additional configuration tasks, as described in Real-Time Audio-Video (Client for Windows).

3. To display the names of the webcams that are attached to your streaming session, choose the downward arrow next to the video icon. If you have more than one webcam (for example, if you have a USB webcam that is connected to your laptop and a built-in webcam), a check mark appears next to the name of the webcam that is selected for use for video conferencing within your streaming session.



4. To use the selected webcam for video conferencing within your AppStream 2.0 streaming session, start the video conferencing application that you want to use. When the webcam is active (being used for video conferencing within your streaming session), the video icon is red.



5. To enable the microphone, choose the microphone icon.

> **Note**
>
> If you have more than one webcam and want to change the one that you use for streaming within an AppStream 2.0 session, you must first detach your webcams from the session. For more information, see the next procedure.

To change the local webcam to use within an AppStream 2.0 streaming session

1. Within your AppStream 2.0 streaming session, in the AppStream 2.0 toolbar in the top left of your session window, do either of the following:

   - If the video icon does not have a diagonal line through it (as shown in the following screenshot), this indicates that the AppStream 2.0 real-time AV feature is available for use and that webcams are still attached to your streaming session. Choose the video icon to detach the webcams.

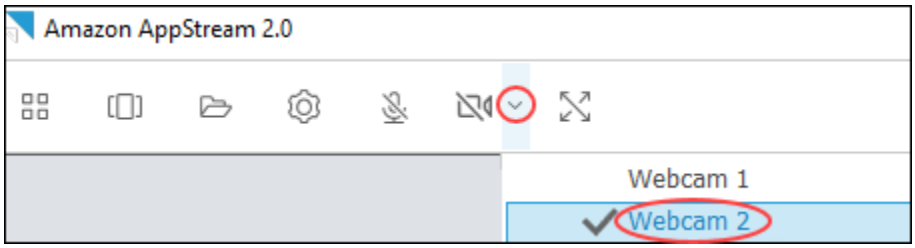
     

   - If the video icon has a diagonal line through it (as shown in the following screenshot), your webcams are already detached from your streaming session. Skip this step and proceed to the next step.

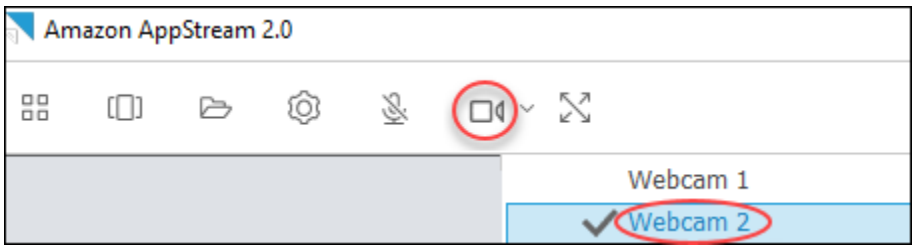
     

2. Display the names of your webcams by choosing the downward arrow next to the video icon, then select the name of the webcam that you want to use.

> **Note**
>
> You must select the name of the webcam you want to use. If you select the check mark next to the name of the webcam you want to use, the webcam won't change.



3. Choose the video icon to reattach the webcams to your AppStream 2.0 streaming session.



Drawing Tablets

Drawing tablets, also known as pen tablets, are computer input devices that let you draw with a stylus (pen). With AppStream 2.0, you can connect a drawing tablet, such as a Wacom drawing tablet, to your local computer and use the tablet with your streaming applications.

Following are requirements and considerations for using drawing tablets with your streaming applications.

- To use this feature, you must connect to AppStream 2.0 by using the AppStream 2.0 client, or through the Google Chrome or Mozilla Firefox browsers only.
- The applications that you stream must support Windows Ink technology. For more information, see Pen interactions and Windows Ink in Windows apps.
- Depending on the streaming applications that you use, your drawing tablet might require USB redirection to function as expected. This is because some applications, such as GIMP, require USB redirection to support pressure sensitivity. If this is the case for your streaming applications, you

must connect to AppStream 2.0 by using the AppStream 2.0 client and share the drawing tablet with your streaming session.

- This feature is not supported on Chromebooks.

To get started with using a drawing tablet during your application streaming sessions, connect your drawing tablet to your local computer with USB, share the device with AppStream 2.0 if required for pressure sensitivity detection, and then start an AppStream 2.0 streaming session. You can use the AppStream 2.0 client or a supported web browser to start a streaming session.

Relative Mouse Offset

By default, during a streaming session, AppStream 2.0 transmits information about mouse movements by using absolute coordinates and rendering the mouse movements locally. For graphics-intensive applications, such as computer-aided design (CAD)/computer-aided manufacturing (CAM) software or video games, mouse performance improves when relative mouse mode is enabled. Relative mouse mode uses relative coordinates, which represent how far the mouse moved since the last frame, rather than the absolute x-y coordinate values within a window or screen. When you enable relative mouse mode, AppStream 2.0 renders the mouse movements remotely.

You can enable this feature during an AppStream 2.0 streaming session in either of the following ways:

- Pressing Ctrl+Shift+F8
- Choosing **Relative Mouse Position [Ctrl+Shift+F8]** from the **Settings** menu on the AppStream 2.0 toolbar in the top left area of your streaming session window. This method works when you use classic mode or **Desktop View**.

Logging

To help with troubleshooting if an issue with the AppStream 2.0 client occurs, you can enable diagnostic logging. The log files that are sent to AppStream 2.0 (AWS) include detailed information about your device and connection to the AWS network. You can enable automatic log uploads so that these files are sent to AppStream 2.0 (AWS) automatically. You can also upload log files on an as-needed basis, before or during an AppStream 2.0 streaming session.

Automatic logging

You can enable automatic logging when you install the AppStream 2.0 client. For information about how to enable automatic logging when you install the AppStream 2.0 client, see step 5 in Setup.

On-demand logging

If an issue occurs during an AppStream 2.0 streaming session, you can also send log files on an as-needed basis. If an issue occurs that causes the AppStream 2.0 client to stop responding, a notification prompts you to choose whether to send an error report and the associated log files to AppStream 2.0 (AWS).

The following procedures describe how to send log files before you sign in to an AppStream 2.0 streaming session and during an AppStream 2.0 streaming session.

To send log files before an AppStream 2.0 streaming session

1. On your local PC where the AppStream 2.0 client is installed, in the lower left of your screen, choose the Windows search icon on the taskbar, and enter **`AppStream`** in the Search box.
2. In the search results, select **Amazon AppStream** to start the AppStream 2.0 client.
3. At the bottom of the AppStream 2.0 sign-in page, choose the **Send Diagnostic Logs** link.
4. To continue connecting to AppStream 2.0, if your AppStream 2.0 administrator has provided you with a web address (URL) to use to connect to AppStream 2.0 for application streaming, enter the URL, and then choose **Connect**.

To send log files during an AppStream 2.0 streaming session

1. If you are not already connected to AppStream 2.0 and streaming an application, use the AppStream 2.0 client to start a streaming session.
2. In the upper right of the AppStream 2.0 session window, choose the **Profiles** icon, and then choose **Send Diagnostic Logs**.

Troubleshooting

If issues occur when you use the AppStream 2.0 client for Windows, your AppStream 2.0 client ID and version number can help your administrator with troubleshooting. The following sections describe how to find the client ID and client version number.

How to find the AppStream 2.0 client ID

The AppStream 2.0 client ID uniquely identifies your device. This ID is created when you install the AppStream 2.0 client for Windows. To find your client ID, open the AppStream 2.0 client. On the bottom left of the client sign-in page, choose the **Client Options** link. The client ID is displayed at the top of the **AppStream 2.0 Client Options** dialog box. You can use your mouse to select the client ID, and then copy it to your clipboard by using your keyboard or mouse.

How to find the AppStream 2.0 client version number

AppStream 2.0 periodically releases new client versions to add features and functionality or resolve issues. To find the version of the AppStream 2.0 client that you have installed, open the AppStream 2.0 client. On the bottom of the client sign-in page, choose the **About Amazon AppStream 2.0** link. The client version is displayed below the Amazon AppStream 2.0 logo.

File Storage Options

If your AppStream 2.0 administrator has enabled it, you can use one or more of the following storage options for your files and folders during application streaming sessions.

- Home folders
- Google Drive
- OneDrive for Business

> **Note**
>
> Google Drive and OneDrive for Business are currently not supported for Linux-based streaming instances.

Use Home Folders

If your AppStream 2.0 administrator has enabled this file storage option, when you are signed in to an AppStream 2.0 streaming session, you can use your home folder. You can do the following with your home folder:

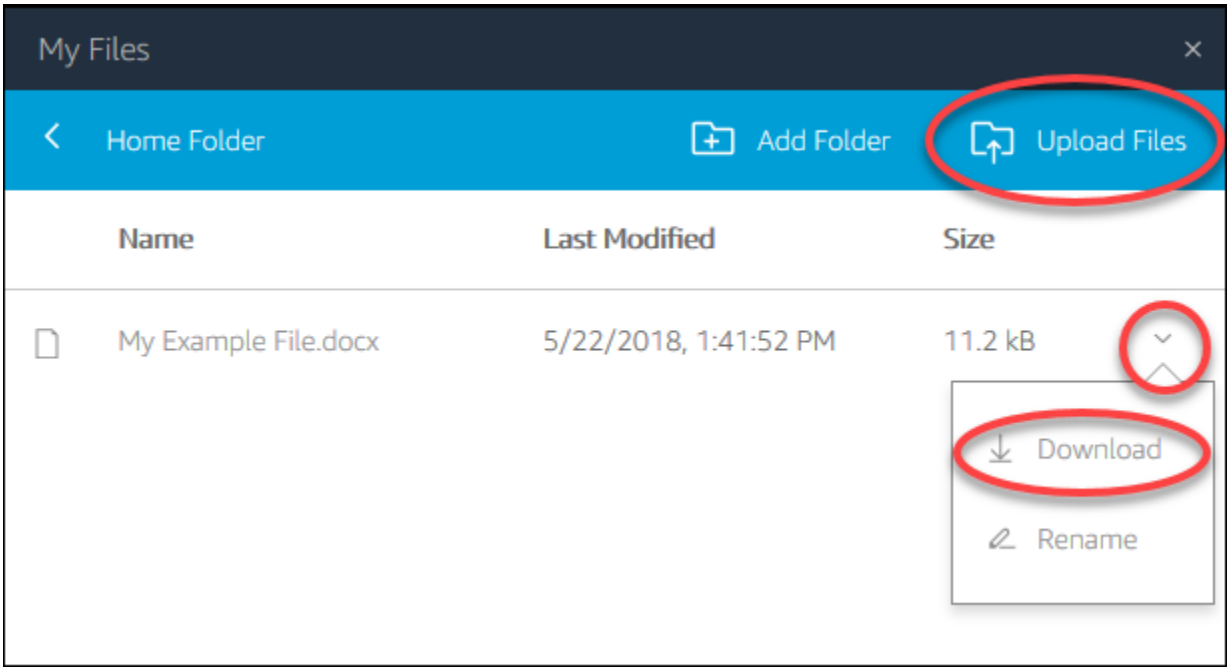
- Open and edit files and folders that you store in your home folder. Content that is stored in your home folder cannot be accessed by other users.

- Upload and download files between your local computer and your home folder. AppStream 2.0 continuously checks for the most recently modified files and folders and backs them up to your home folder.
- When you are working in an application, you can access files and folders that are stored in your home folder. Choose **File Open** from the application and browse to the file or folder that you want to open. To save your changes in a file to your home folder, choose **File Save** from the application interface, and browse to the location in your home folder where you want to save the file.
- You can also access your home folder by choosing **My Files** from the web view session toolbar.

> **Note**
>
> If your home folder doesn't appear, view your home folder files by browsing to the following directory in File Explorer: C:\Users\PhotonUser\My Files\Home Folder.

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

1. In the top left of the AppStream 2.0 toolbar, choose the **My Files** icon.
2. Navigate to an existing folder, or choose **Add Folder** to create a folder.
3. When the folder that you want is displayed, do one of the following:
   - To upload a file to the folder, select the file that you want to upload, and choose **Upload**.
   - To download a file from the folder, select the file that you want to download, choose the down arrow to the right of the file name, and choose **Download**.



Use Google Drive

> **Note**
>
> Amazon AppStream 2.0's use and transfer to any other app of information received from Google APIs will adhere to Google API Services User Data Policy, including the Limited Use requirements.

If your AppStream 2.0 administrator has enabled this file storage option, you can add your Google Drive account to AppStream 2.0. After you add your account and you sign in to an AppStream 2.0 streaming session, you can do the following in Google Drive:

> **Note**
>
> Google Drive is currently not supported for Linux-based streaming instances.

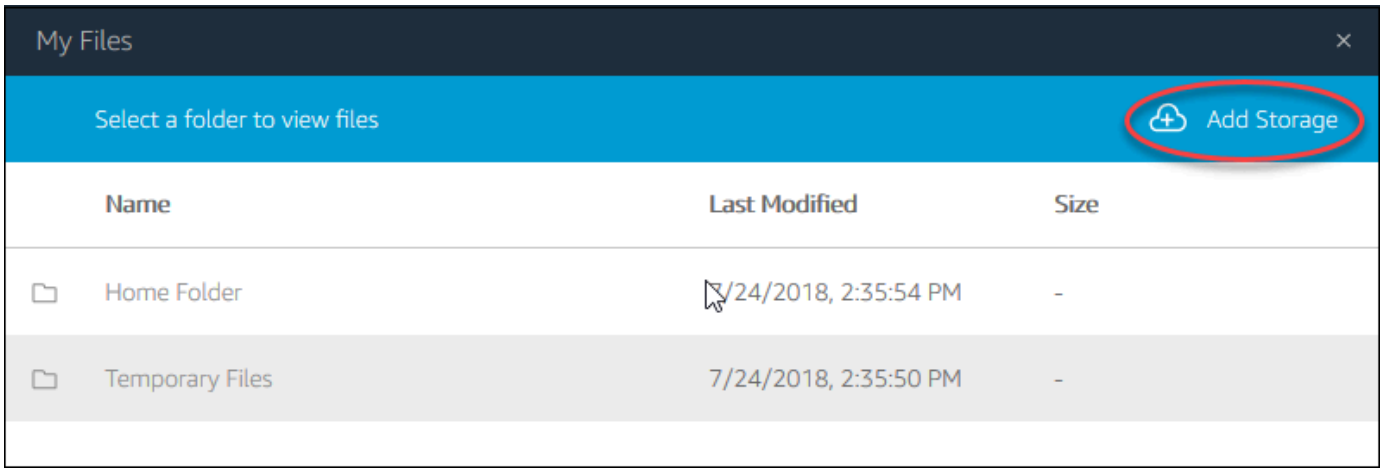
- Open and edit files and folders that you store in Google Drive. Other users cannot access your content unless you choose to share it.

- Upload and download files between your local computer and Google Drive. Any changes that you make to your files and folders in Google Drive during a streaming session are automatically backed up and synchronized. They are available to you when you sign in to your Google Drive account and access Google Drive outside of your streaming session.
- When you are working in an application, you can access your files and folders that are stored in Google Drive. Choose **File**, **Open** from the application interface and browse to the file or folder that you want to open. To save your changes in a file to your Google Drive, choose **File**, **Save** from the application and browse to the location in Google Drive where you want to save the file.
- You can also access Google Drive by choosing **My Files** from top left of the AppStream 2.0 toolbar.

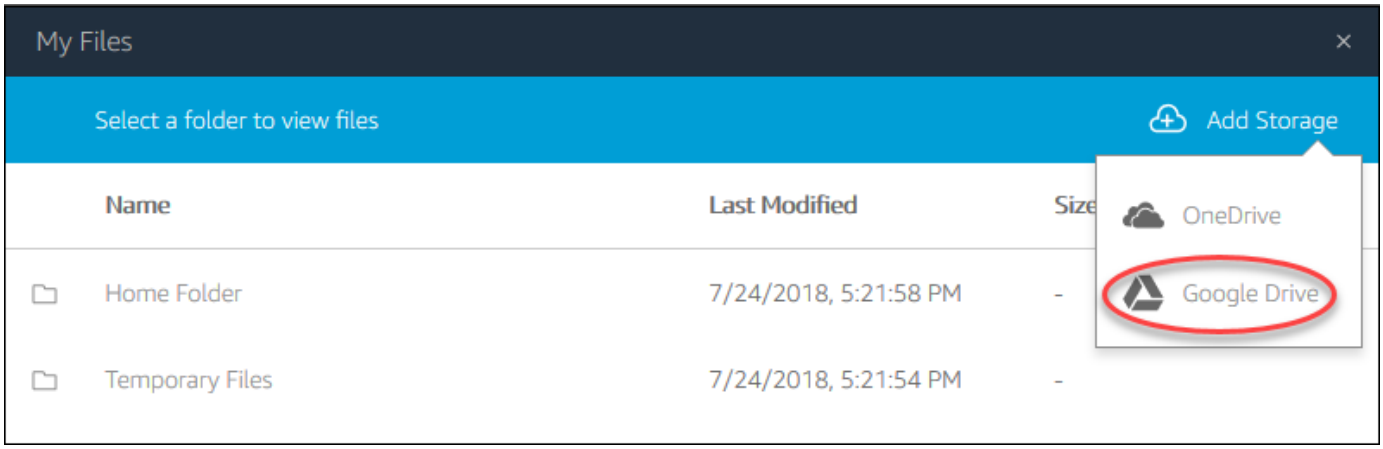
To add your Google Drive account to AppStream 2.0

To access your Google Drive during AppStream 2.0 streaming sessions, you must first add your Google Drive account to AppStream 2.0.

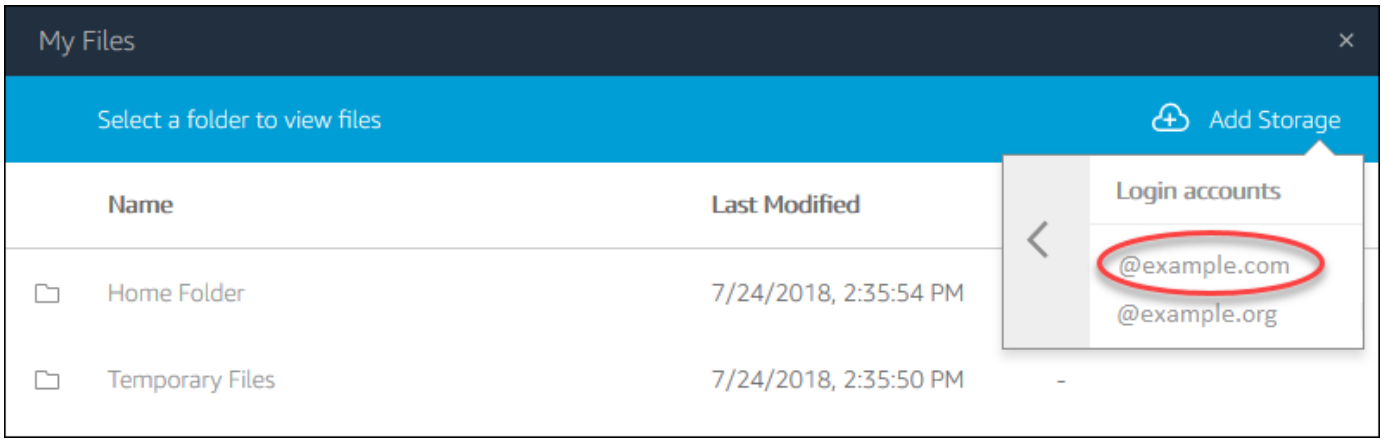
1. In the top left of the AppStream 2.0 toolbar, choose the **My Files** icon.
2. In the **My Files** dialog box, choose **Add Storage**.



3. Choose **Google Drive**.

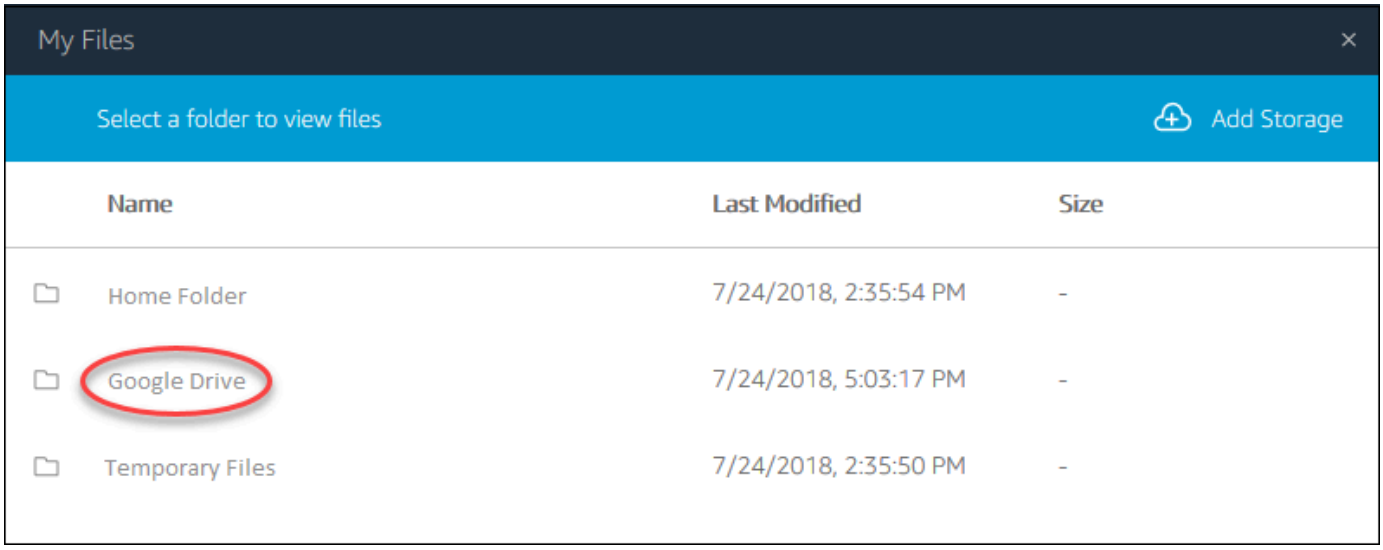


4. Choose the domain for your Google Drive account.



5. The **Sign in with Google** dialog box is displayed. Enter the sign-in credentials for your Google Drive account when prompted.

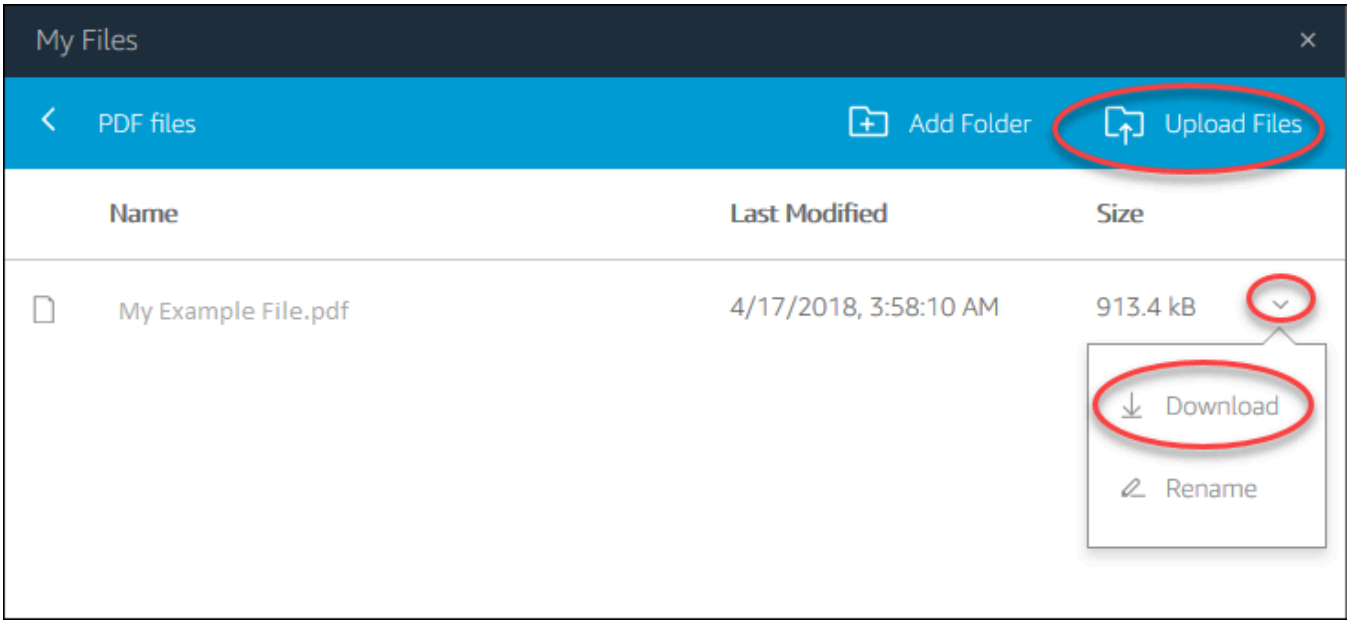
   After your Google Drive account is added to AppStream 2.0, your Google Drive folder is displayed in **My Files**.



6. To work with your files and folders in Google Drive, choose the **Google Drive** folder and browse to the file or folder you want. If you do not want to work with files in Google Drive during this streaming session, close the **My Files** dialog box.

To upload and download files between your local computer and your Google Drive

1. In the top left of the AppStream 2.0 toolbar, choose the **My Files** icon.
2. In the **My Files** dialog box, choose **Google Drive**.
3. Navigate to an existing folder, or choose **Add Folder** to create a folder.
4. When the folder that you want is displayed, do one of the following:
   - To upload a file to the folder, select the file that you want to upload, and choose **Upload**.
   - To download a file from the folder, select the file that you want to download, choose the down arrow to the right of the file name, and choose **Download**.



Use OneDrive for Business

> **Note**
>
> OneDrive for Business is currently not supported for Linux-based streaming instances.

If your AppStream 2.0 administrator has enabled this file storage option, you can add your OneDrive account to AppStream 2.0. After you add your account and sign in to an AppStream 2.0 streaming session, you can do the following in OneDrive:

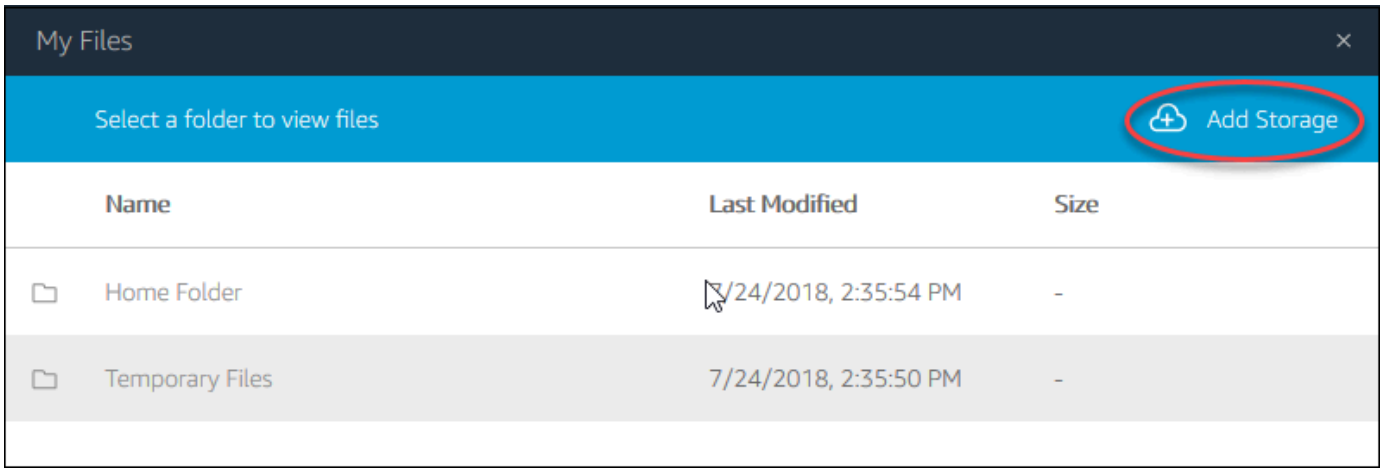
- Open and edit files and folders that you store in OneDrive. Other users cannot access your content unless you choose to share it.
- Upload and download files between your local computer and OneDrive. Any changes that you make to your files and folders in OneDrive during a streaming session are backed up and synchronized automatically. They are available to you when you sign in to your OneDrive account and access OneDrive outside of your streaming session.
- When you are working in an application, you can access your files and folders that are stored in OneDrive. Choose **File**, **Open** from the application interface and browse to the file or folder that you want to open. To save your changes in a file to OneDrive, choose **File**, **Save** from the application and browse to the location in OneDrive where you want to save the file.

- You can also access OneDrive by choosing **My Files** from the top left of the AppStream 2.0 toolbar.

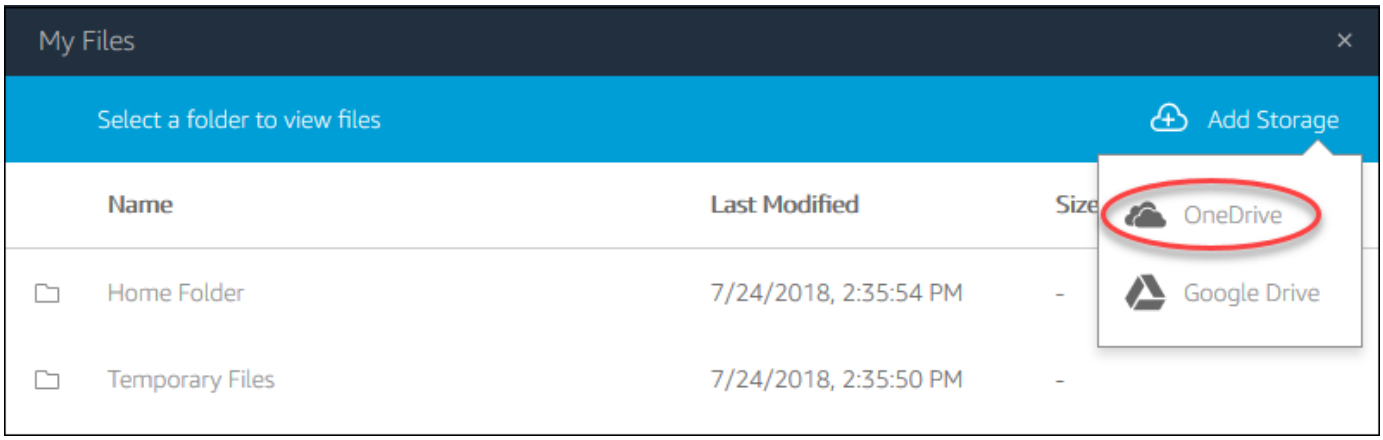
To add your OneDrive account to AppStream 2.0

To access your OneDrive during AppStream 2.0 streaming sessions, you must first add your OneDrive account to AppStream 2.0.

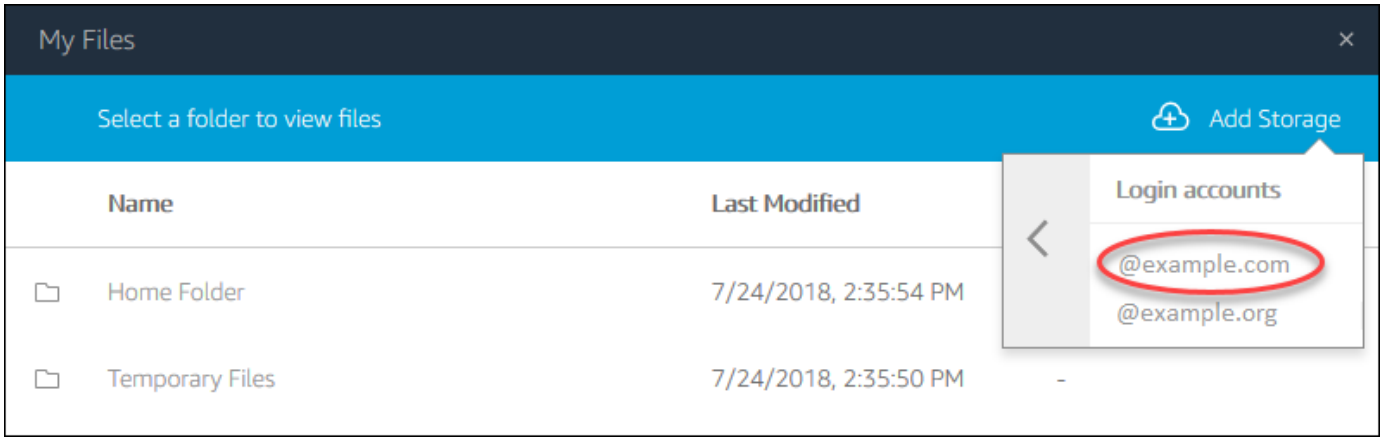
1. In the top left of the AppStream 2.0 toolbar, choose the **My Files** icon.
2. In the **My Files** dialog box, choose **Add Storage**.



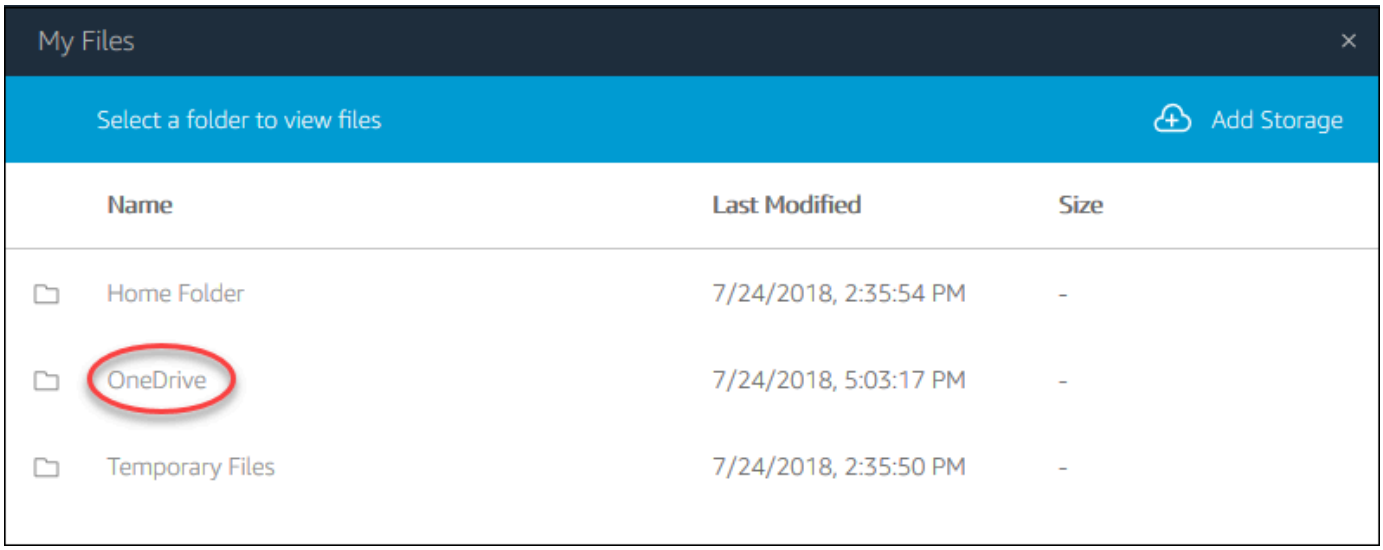
3. Choose **OneDrive**.



4. Under **Login accounts**, choose the domain for your OneDrive account.



5. In the **Sign in** dialog box, enter the sign-in credentials for your account.
6. After your OneDrive account is added to AppStream 2.0, your OneDrive folder is displayed in **My Files**.

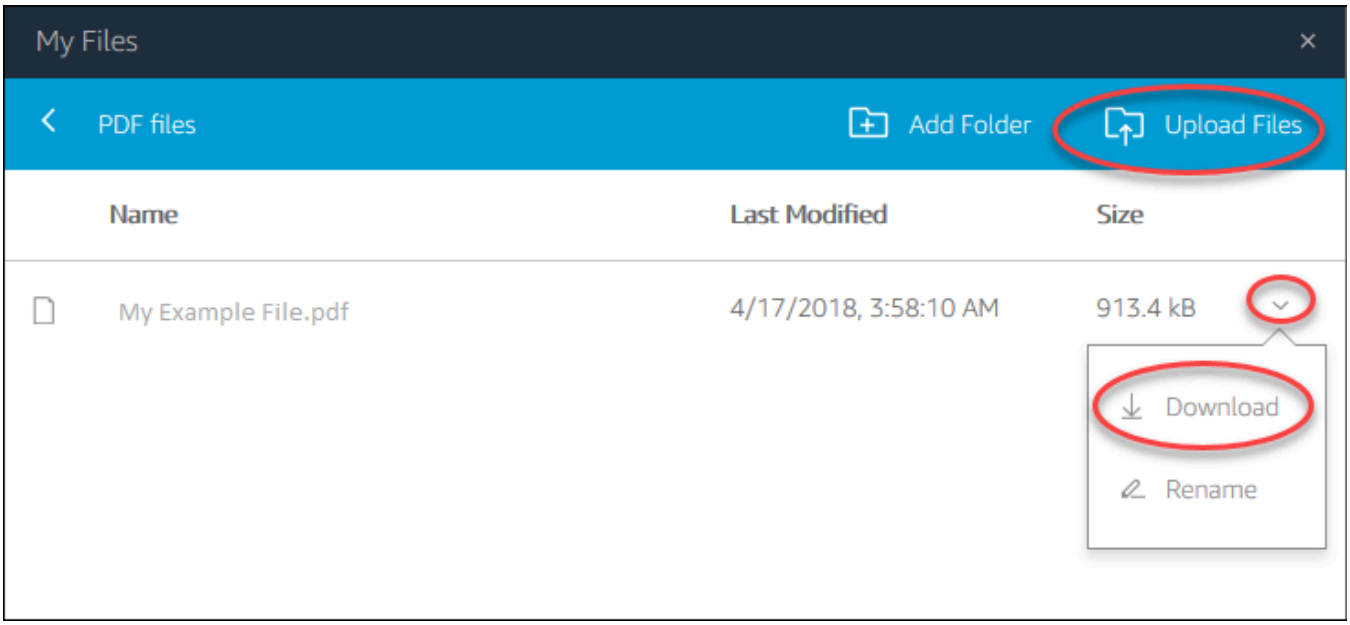


7. To work with your files and folders in OneDrive, choose the **OneDrive** folder and browse to the file or folder you want. If you do not want to work with files in OneDrive during this streaming session, close the **My Files** dialog box.

To upload and download files between your local computer and your OneDrive

1. In the top left of the AppStream 2.0 toolbar, choose the **My Files** icon.
2. In the **My Files** dialog box, choose **OneDrive**.
3. Navigate to an existing folder, or choose **Add Folder** to create a folder.
4. When the folder is displayed, do one of the following:

- To upload a file to the folder, select the file that you want to upload, and choose **Upload**.
- To download a file from the folder, select the file that you want to download, choose the down arrow to the right of the file name, and choose **Download**.



To remove OneDrive permissions from AppStream 2.0

If you no longer want to use OneDrive during your AppStream 2.0 streaming sessions, follow these steps to remove OneDrive permissions from AppStream 2.0.

> **Note**
>
> You can restore these permissions at any time during an AppStream 2.0 streaming session.

1. Sign in to Office 365 or OneDrive for Business.
2. In the right pane, under **My accounts**, choose **My account**.
3. On the account dashboard page, in **App permissions**, choose **Change app permissions**.
4. On the **App permissions** page, under **Amazon AppStream 2.0**, choose **Revoke**.

Configure Regional Settings

You can configure regional settings so that your AppStream 2.0 Windows streaming sessions use settings that are specific to your location or language. Changes that you make during your streaming session are applied to future streaming sessions.

To configure regional settings for your Windows AppStream 2.0 streaming sessions

1. In the top left of the AppStream 2.0 toolbar, choose the **Settings** icon, and then choose **Regional settings**.
2. In the **Regional settings** dialog box, set the following options as needed. When you're done, choose **Save**.

   - **Time zone** — Determines the system time used by Windows and any applications that rely on the operating system time.
   - **Locale** (also known as culture) — Determines how Windows displays numbers, currency, time, and dates. AppStream 2.0 supports the following locales: Chinese (Simplified and Traditional), Dutch, English, French, German, Italian, Japanese, Korean, Portuguese, Spanish, and Thai.
   - **Input method** — Determines the keystroke combinations that can be used to input characters in another language.

Configuring regional settings is not yet supported for Linux streaming sessions. However, you can switch between different input methods available in your streaming sessions with shortcut key combinations specified by your administrator. The default shortcut key combinations are “Super + Space” and “Shift + Super + Space”. “Super” is the “Windows” key on a Windows keyboard or the “Command” key on an Apple keyboard. Always check with your administrators for the shortcut keys they specified when creating the image. For example, in Tutorial: Enable Japanese Support for Your Linux Images, the shortcut key combinations have been changed to “Control + Space” and “Shift + Control + Space”.

Extension SDK Developer Guide

Amazon AppStream 2.0 uses NICE DCV technology to provide secure, high-performance access to your applications. With the NICE DCV Extension SDK, developers can customize AppStream 2.0 experiences for end users, including the following actions:

- Facilitate custom hardware support.
- Enhance the usability of third-party applications in remote sessions. For example, you can add local audio termination for VoIP applications or local video playback for conferencing applications.
- Provide accessibility software such as screen readers with information about the remote session and applications running remotely.
- Allow security software to analyze the security posture of the local endpoint to allow conditional access policies.
- Perform arbitrary data transfers over an established remote session.

To get started with the NICE DCV Extension SDK, see What is the NICE DCV Extension SDK?. The SDK itself can be found in the NICE DCV Extension SDK Github repository. In addition, integration examples of the SDK can be found in the NICE DCV Extension SDK Samples Github repository.

Prerequisites

Before you start working with the NICE DCV Extension SDK, make sure that your AppStream 2.0 client applications and your AppStream 2.0 servers meet the following requirements.

Supported AppStream 2.0 clients:

- AppStream 2.0 Windows client version 1.1.1154 or above

> **Note**
>
> AppStream 2.0 web access doesn't support the NICE DCV Extension SDK.

Supported AppStream 2.0 streaming instances:

- Windows Server 2016 and 2019

- AppStream 2.0 agent version for Windows releaed on May 8, 2023 or later
- Managed AppStream 2.0 image updates released on May 8, 2023 or later

Third-party vendor extensions

AWS supports the NICE DCV Extension SDK API within the AppStream 2.0 host and client processes. However, please note that extensions developed by third-party Independent Software Vendors (ISVs) are not developed or maintained by AWS. Therefore, support for the extensions themselves, including their installation, configuration, troubleshooting, and updates, is the responsibility of the third-party vendor who developed the extension. If you have any issues or questions related to third-party extensions, please contact the relevant third-party vendor for support.

Document History for Amazon AppStream 2.0

- **API version:** 2016-12-01

The following table describes important additions to the AppStream 2.0 service (including AppStream 2.0 base image, AppStream 2.0 agent, and AppStream 2.0 client releases) and to the *Amazon AppStream 2.0 Administration Guide* documentation from June 4, 2018 onward. We also update the documentation frequently to address the feedback that you send us.

For notification about these updates, you can subscribe to the Amazon AppStream 2.0 RSS feed.

Change	Description	Date
New client version	Client version 1.1.1303	April 4, 2024
New relay state region endpoints	Relay state endpoints for Windows client application version 1.1.1300 and later	April 1, 2024
New client version	Client version 1.1.1300	March 28, 2024
Latest base image	Base image released on 01-26-2024	February 16, 2024
Use session scripts on multi-session fleets	When using session scripts on multi-session fleets, there are additional requirements and considerations to ensure optimal performance and security.	February 15, 2024
New agent version	01-17-2024 version for AppStream 2.0 Agent	February 15, 2024
Clipboard update	You can choose **Copy to local device character limit** or **Paste to remote session character limit** or both to	February 15, 2024

	limit the amount of data that users can copy or paste when using the clipboard	
New client version	Client version 1.1.1259	February 8, 2024
New client version	Client version 1.1.1246	January 18, 2024
Windows Server 2022 support	Added support for Windows Server 2022 Base	December 14, 2023
Web browser access version 2	AppStream 2.0 web browser access version 2 offers an enhanced end user experienc e, including menu options that are easily discoverable and textual guidance for end users	December 11, 2023
New client version	Client version 1.1.1228	November 1, 2023
Multi-session fleets	A multi-session fleet enables you to provision more than one user session on a single fleet instance	October 26, 2023
New agent version	08-22-2023 version for AppStream 2.0 Agent	August 25, 2023
New Instance Family	New Graphics G5 Instance Family	July 26, 2023
New Agent version	06-11-2023 version for AppStream 2.0 Agent	July 25, 2023
App block builder	An app block builder is a reusable resource that you can use to package your applications (or app block)	June 29, 2023

New client version	Client version 1.1.1183	June 26, 2023
New Agent version	05-30-2023 version for AppStream 2.0 Agent	June 15, 2023
Extension SDK Developer Guide	Amazon AppStream 2.0 uses NICE DCV technolog y to provide secure, high-performance access to your applications	May 26, 2023
New Agent version	05-08-2023 version for AppStream 2.0 Agent	May 12, 2023
New client version	Client version 1.1.1159	May 9, 2023
New Agent version	04-13-2023 version for AppStream 2.0 Agent	April 25, 2023
Latest base images	Latest Linux base images released on 03-15-2023	April 5, 2023
Documentation update: New region — AWS GovCloud (US-East)	Updated Amazon AppStream 2.0 in the *AWS GovCloud (US) User Guide* and updated the relay state endpoint table in "Setting Up SAML" and other content as needed.	April 5, 2023
New Agent version	03-21-2023 version for AppStream 2.0 Agent	April 3, 2023
New region support	South America (São Paulo) is now supported	December 15, 2022
New client version	Client version 1.1.1118	November 7, 2022

Client features table	Added a table that compares the features that are supported by the different access types.	November 7, 2022
Certificate-based authentication	You can use certificate-based authentication with AppStream 2.0 fleets joined to Microsoft Active Directory.	October 31, 2022
Latest base images	Latest Linux base images released on 10-05-2022	October 27, 2022
New agent version	Agent version 10-13-2022	October 24, 2022
New client version	Client version 1.1.1099	October 13, 2022
Webcam support for Linux	Enable and disable webcam for Linux-based images	October 5, 2022
Latest base images	Latest Linux base images released on 09-21-2022	October 3, 2022
New client version	Client version 1.1.1066	August 17, 2022
Latest base images	Latest Windows base images released on 07-12-2022	July 21, 2022
New Agent version	06-20-2022 version for AppStream 2.0 Agent	June 30, 2022
New client version	Client version 1.1.421	June 29, 2022
New region support	US East (Ohio) is now supported	June 28, 2022
New default quotas	Max concurrent sessions for Elastic fleets	May 31, 2022
New client version	Client version 1.1.414	April 27, 2022

Japanese support	Enable Japanese support for a Linux image	April 19, 2022
Session scripts for Elastic fleets	Configure and specify session scripts for Elastic fleets	April 14, 2022
Canada (Central) support	Canada (Central) region now supported	March 31, 2022
New Agent version	03-14-2022 version for AppStream 2.0 Agent	March 25, 2022
New agent version	Agent version 03-14-2022	March 19, 2022
Latest base images	Latest Windows base images released on 03-03-2022	March 14, 2022
Managed image update	Released base image 02-18-2022	March 3, 2022
New client version	Client version 1.1.398	February 23, 2022
New client version	Client version 1.1.394	February 8, 2022
Managed image update	AppStream 2.0 Agent version 12-20-2021	January 6, 2022
Attribute-based application entitlements	Application entitlements control access to specific applications within your AppStream 2.0 stacks.	January 5, 2022
App blocks and applications	When using an Elastic fleet, you can create app blocks and applications.	November 19, 2021
Create Linux images	You can now create Linux-based Amazon AppStream 2.0 images.	November 15, 2021

New agent version	Agent version 10-19-2021.	October 26, 2021
Managed image and base image updates	Various updates.	October 21, 2021
New client version	Client version 1.1.333	September 14, 2021
Managed image updates	Managed imaged updates for August 12, 2021.	August 23, 2021
New agent version	Agent version 08-02-2021.	August 18, 2021
New client version	Client version 1.1.304.	August 2, 2021
AppStream 2.0 client update: version 1.1.304	Upgrades the embedded Chromium browser to version 91	August 2, 2021
AppStream 2.0 agent update	Resolves multiple issues	August 2, 2021
Base image update	Base image update (07-19-2021).	July 23, 2021
AppStream 2.0 agent update	Resolves multiple issues	July 1, 2021
Base image update	Base image update (06-01-2021).	June 10, 2021
Install AMD Driver on Graphics Design Instances	If you need to update the AMD driver on your Image Builder that is using a Graphics Design instance, you can either use the latest AppStream 2.0 Graphics Design base images, or download the AMD driver and install it on your Image Builder.	June 4, 2021

Documentation update: New Graphics Design base images	Added two new released images and updated other content as needed.	June 3, 2021
Documentation update: New AppStream 2.0 agent version	Added the entry for the 05-17-2021 agent version and updated other content as needed.	May 26, 2021
AppStream 2.0 agent update	Resolves multiple issues	May 17, 2021
AppStream 2.0 client update: version 1.1.294	Resolves issues with SAML 2.0 authentication, client stability on Windows 7, and folder sharing on client reconnection	April 26, 2021
New client version	Client version 1.1.1154	April 25, 2021
Documentation update: Support for managed AppStream 2.0 image updates	Created the *Update an Image by Using Managed AppStream 2.0 Image Updates* section in "Administer Your Images" and updated other content as needed.	April 8, 2021
AppStream 2.0 client update: version 1.1.285	Includes fixes that improve compatibility with antivirus software	March 8, 2021
AppStream 2.0 agent update	Resolves multiple issues	March 4, 2021
New agent version	Agent version 02-21-2022	February 24, 2021
New client version	Client version 1.1.1130	February 9, 2021

Documentation update: Support for smart cards	Created the *Smart Cards* section in "System Requirements and Feature Support" and updated other content as needed.	January 12, 2021
AppStream 2.0 agent update	Adds support for using smart cards for Windows sign in to streaming instances and in-session authentication	January 4, 2021
Documentation update: Support for real-time audio-video (AV)	Created the *Real-Time Audio-Video* section in "System Requirements and Feature Support" and updated other content as needed.	December 28, 2020
AppStream 2.0 client update: version 1.1.257	Adds support for real-time audio video (AV) and smart card authentication and resolves an issue with Excel	December 28, 2020
AppStream 2.0 base image update	Updates for Base, Graphics Design, Graphics G4dn, Graphics Pro: Includes Microsoft Windows updates up to December 9, 2020; AWS CLI version 1.18.138; and Amazon SSM Agent version 3.0.431.0	December 28, 2020
AppStream 2.0 agent update	Resolves multiple issues	December 17, 2020
AppStream 2.0 agent update	Resolves multiple issues	October 8, 2020
AppStream 2.0 agent update	Resolves multiple issues	September 1, 2020

AppStream 2.0 client update: version 1.1.195	Improves local drive and folder sharing when file redirection is used, and provides other enhancements and fixes.	August 18, 2020
Documentation update: Support for local printer redirection	Created the *Enable Local Printer Redirection* section in "System Requirements and Feature Support" and updated other content as needed.	August 7, 2020
AppStream 2.0 agent update	Adds support for local printer redirection to the AppStream 2.0 client and resolves multiple issues	July 30, 2020
AppStream 2.0 base image update	Updates for Base, Graphics Design, Graphics G4dn, Graphics Pro: Includes Microsoft Windows updates up to June 9, 2020; AWS CLI version 1.18.86; and Amazon SSM Agent version 2.3.1319.0	July 16, 2020
New Region: Mumbai	Updated the relay state endpoint table in "Setting Up SAML" and other content as needed.	July 8, 2020
AppStream 2.0 client update: version 1.1.179	Adds support for local printer redirection, and provides other enhancements and fixes.	July 8, 2020

Documentation update: Support for drawing tablets	Created the *Drawing Tablets* section in "System Requirements and Feature Support" and updated other content as needed.	June 26, 2020
AppStream 2.0 agent update	Resolves multiple issues	May 27, 2020
AppStream 2.0 client update: version 1.1.160	Resolves an issue that prevents the application catalog page from opening on a Windows PC that has .NET Framework version 4.7.1 or earlier installed; also resolves another intermittent issue	April 28, 2020
Documentation update: Support for on-demand logging	Created the *Automatic and On-Demand Diagnostic Log Uploads* section in "System Requirements and Feature Support" and updated other content as needed.	April 22, 2020
Documentation update: Support for defining trusted subdomains for user connections in a DNS TXT record	Created the *Create the AS2TrustedDomains DNS TXT Record to Enable Your Domain for the AppStream 2.0 Client Without Registry Changes* section in "System Requirements and Feature Support" and updated other content as needed.	April 22, 2020

AppStream 2.0 client update: version 1.1.156	Adds support for on-demand diagnostic log and minidump uploads, defining trusted subdomains for user connections in a DNS TXT record, and other enhanceme nts	April 22, 2020
AppStream 2.0 agent update	Resolves an issue that causes streaming sessions to fail; improves performance with IAM roles	April 20, 2020
AppStream 2.0 base image update	Updates for Base, Graphics Design, Graphics G4dn, Graphics Pro (Windows Server 2019): Includes Microsoft Windows updates up to March 10, 2020; AWS CLI version 1.18.21; and Amazon SSM Agent version 2.3.930.0	April 18, 2020
AppStream 2.0 base image update	Updates for Base, Graphics Design, Graphics Pro: Includes Microsoft Windows updates up to February 11, 2020; AWS CLI version 1.17.5; and Amazon SSM Agent version 2.3.842.0	March 18, 2020

AppStream 2.0 base image update	Adds support for Graphics G4dn instances (Windows Server 2012 R2); also includes Microsoft Windows updates up to February 11, 2020; AWS CLI version 1.17.5; and Amazon SSM Agent version 2.3.842.0	March 16, 2020
AppStream 2.0 client update: version 1.1.137	Reverts the updates in version 1.1.136	March 8, 2020
AppStream 2.0 client update: version 1.1.136	Adds support for defining trusted subdomains for user connections in a DNS TXT record, and provides other enhancements and fixes	March 5, 2020
AppStream 2.0 base image update	Adds support for Graphics g4dn instances (Windows Server 2016, Windows Server 2019); also includes Microsoft Windows updates up to February 11, 2020; AWS CLI version 1.17.5; and Amazon SSM Agent version 2.3.842.0	March 5, 2020
Documentation update: Support for native application mode	Created the *Native applicati on mode* section in "System Requirements and Feature Support" and updated other content as needed.	February 28, 2020
AppStream 2.0 client update: version 1.1.129	Adds support for native application mode and provides other enhancements and fixes	February 28, 2020

AppStream 2.0 agent update	Adds support for native application mode and the **Desktop** stream view	February 19, 2020
AppStream 2.0 base image update	Updates for Graphics Design: Adds support for Windows Server 2019, with Microsoft Windows updates up to November 12, 2019	January 13, 2020
AppStream 2.0 agent update	Resolves multiple issues	January 13, 2020
Documentation update: Enhanced the documentation for security in AppStream 2.0	Created "Security in Amazon AppStream 2.0" and updated other content as needed.	December 23, 2019
AppStream 2.0 client update: version 1.0.525	Resolves a DPI issue that causes the mouse cursor to point to the wrong location when a user clicks on an application during a streaming session	December 12, 2019
AppStream 2.0 base image update	Includes Microsoft Windows updates up to December 12, 2019; AWS CLI version 1.16.284; and Amazon SSM Agent version 2.3.760.0	December 12, 2019
Documentation update: Support for AppStream 2.0 z1d-based instances	Updated "AppStream 2.0 Instance Families"	November 21, 2019
AppStream 2.0 agent update	AppStream 2.0 assemblie s are now signed, including executables and installer packages	November 13, 2019

Documentation update: Embedded AppStream 2.0 streaming sessions	Created "Embed AppStream 2.0 Streaming Sessions" and updated other content as needed.	November 1, 2019
AppStream 2.0 client update: version 1.0.511	Adds support for up to 4 monitors and provides other enhancements	October 16, 2019
Documentation update: New region — AWS GovCloud (US-West)	Created Amazon AppStream 2.0 in the *AWS GovCloud (US) User Guide* and updated the relay state endpoint table in "Setting Up SAML" and other content as needed.	October 9, 2019
AppStream 2.0 agent update	Modifies the AppStream 2.0 storage connector to no longer bypass the system proxy server	October 8, 2019
Documentation update: FIPS-compliant endpoints	Created "Protecting Data in Transit with FIPS Endpoints" and updated other content as needed.	October 7, 2019
AppStream 2.0 client update: version 1.0.499	Resolves issues with client-side hardware rendering and with the client not working correctly when Bluetooth headsets are connected to the local computer	September 26, 2019
AppStream 2.0 agent update	Resolves multiple issues	September 23, 2019

AppStream 2.0 base image update	Updates for all Base and Graphics Pro instances, and for Graphics Design Windows Server 2012 R2 instances: Includes Microsoft Windows updates up to August 13th, 2019 and AWS CLI version 1.16.222. Additional updates for Base, Graphics Design, and Graphics Pro instances	September 18, 2019
Documentation update: Support for applying IAM roles to AppStream 2.0 streaming instances	Created "Using an IAM Role to Grant Permissions to Applications and Scripts Running on AppStream 2.0 Streaming Instances" and updated other content as needed.	September 9, 2019
AppStream 2.0 base image update	Updates for Graphics Design instances: Includes Microsoft Windows updates up to August 13th, 2019; AWS CLI version 1.16.222; and AMD driver 24.20.13028.3002	September 5, 2019
AppStream 2.0 agent update	Adds support for applying IAM roles to AppStream 2.0 streaming instances	September 3, 2019
Documentation update: AppStream 2.0 file system redirection	Created "Enable File System Redirection for Your AppStream 2.0 Users" and updated other content as needed.	August 20, 2019

Documentation update: Interface VPC endpoints	Created "Creating and Streaming From Interface VPC Endpoints." Also created "Access AppStream 2.0 API Operations and CLI Commands Through an Interface VPC Endpoint" and updated other content as needed.	August 19, 2019
AppStream 2.0 client update: version 1.0.480	Adds support for AppStream 2.0 file system redirection	August 14, 2019
AppStream 2.0 agent update	Adds support for AppStream 2.0 file system redirection	August 8, 2019
Documentation update: Programmatic AppStream 2.0 image creation	Created "Create Your AppStream 2.0 Image Programmatically" and updated other content as needed.	August 1, 2019
AppStream 2.0 agent update	Adds support for creating AppStream 2.0 images programmatically	July 26, 2019
Documentation update: Support for Windows Server 2016 and Windows Server 2019 base images	Updated "AppStream 2.0 Base Image Version History" and other content as needed.	June 28, 2019
AppStream 2.0 agent update	Adds support for Windows Server 2016 and Windows Server 2019 base images	June 19, 2019
AppStream 2.0 base image update	Adds support for Windows Server 2016 and Windows Server 2019	June 10, 2019

AppStream 2.0 base image update	Includes Microsoft Windows updates up to May 14, 2019	May 28, 2019
Documentation update: AppStream 2.0 Usage reports	Created "AppStream 2.0 Usage Reports" and updated other content as needed.	May 21, 2019
Documentation update: Support for disconnecting idle users	Updated "Create a Fleet" in "Create an AppStream 2.0 Fleet and Stack."	May 17, 2019
AppStream 2.0 client update: version 1.0.407	Adds support for configuring the amount of time that users can be idle (inactive) before they are disconnected from their streaming session	May 16, 2019
AppStream 2.0 agent update	Adds support for configuring the amount of time that users can be idle (inactive) before they are disconnected from their streaming session. Also adds support for subscribi ng to AppStream 2.0 usage reports.	May 7, 2019
AppStream 2.0 base image update	Includes Microsoft Windows updates up to April 20, 2019; AWS CLI version 1.16.126; and NVIDIA Graphics Driver 412.16 for Graphics Pro instances	April 29, 2019
Documentation update: Logging AppStream 2.0 API calls with AWS CloudTrail	Created "Logging AppStream 2.0 API Calls with AWS CloudTrail."	April 25, 2019

Documentation update: HIPAA compliance	Created "AppStream 2.0 Compliance."	March 28, 2019
Documentation update: Gesture support	Created "Touchscreen Devices" and updated other content as needed.	March 13, 2019
AppStream 2.0 client update: version 1.0.375	Adds touch screen support on Windows PCs and support for: Automatically connectin g USB devices when a new streaming session starts, running session scripts, and delivering virtualized applicati ons using the AppStream 2.0 dynamic application framework APIs	March 7, 2019
AppStream 2.0 agent update	Adds support for gestures on touch-enabled iPads, Android tablets, and Windows devices	March 7, 2019
New Region: Seoul	Updated the relay state endpoint table in "Setting Up SAML" and other content as needed.	February 13, 2019
Documentation update: Session scripts	Created "Use Session Scripts to Manage Your AppStream 2.0 Users' Streaming Experience" in "Images" and updated other content as needed.	January 27, 2019

AppStream 2.0 base image update	Includes Microsoft Windows updates up to December 10, 2018; AWS CLI version 1.16.84; and NVIDIA Graphics Driver 391.58 for Graphics Pro instances	January 22, 2019
AppStream 2.0 agent update	Adds support for using on-instance session scripts. Also adds support for adding tags to the following AppStream 2.0 resource types during resource creation: image builders, images, fleets, and stacks.	January 22, 2019
AppStream 2.0 client update: version 1.0.320	Adds support for AppStream 2.0 dynamic application framework APIs, AppStream 2.0 regional settings, the AppStream 2.0 user pool, and provides other enhancements	January 19, 2019
Documentation update: Default regional settings	Created "Configure Default Regional Settings for Your AppStream 2.0 Users" and updated other content as needed.	December 13, 2018
Documentation update: Dynamic application framework	Created "Manage App Entitlement" and updated other content as needed.	December 7, 2018

AppStream 2.0 agent update	Adds support for using the AppStream 2.0 dynamic application framework to build a dynamic app provider. Also adds support for using a Japanese keyboard with web clients that run Windows.	December 4, 2018
Documentation update: AppStream 2.0 client	Created "The AppStream 2.0 Client" and updated other content as needed.	November 20, 2018
AppStream 2.0 client update: version 1.0.247	Initial release	November 20, 2018
AppStream 2.0 agent update	Adds support for launching streaming sessions using the AppStream 2.0 Windows client	November 14, 2018
Documentation update: Image sharing	Created "Administer Your AppStream 2.0 Images" and updated other content as needed.	September 14, 2018
Documentation update: Application settings persistence	Created "Enable Application Settings Persistence for Your AppStream 2.0 Users" and updated other content as needed.	September 5, 2018
AppStream 2.0 agent update	Adds support for application settings persistence	August 29, 2018

Documentation update: OneDrive support	Created "Enable and Administer OneDrive for Your AppStream 2.0 Users" and updated other content as needed.	July 31, 2018
AppStream 2.0 agent update	Adds support for OneDrive persistent storage	July 26, 2018
AppStream 2.0 agent update	Resolves an issue with optimizing images for application launch.	June 19, 2018
Documentation update: Regional settings	Created "Enable Regional Settings for Your AppStream 2.0 Users" and updated other content as needed.	June 14, 2018
Documentation update: Default application and Windows settings	Added "Step 4: Create Default Application and Windows Settings" to "Tutorial: Create a Custom Image" and updated other content as needed.	June 14, 2018
AppStream 2.0 base image update	Includes Microsoft Windows updates up to May 9, 2018 and Windows PowerShell 5.1.	June 12, 2018
AppStream 2.0 agent update	Adds support for regional settings and default applicati on and Windows settings.	June 6, 2018
Documentation update: Google Drive support	Created "Enable and Administer Google Drive for Your AppStream 2.0 Users" and updated other content as needed.	June 4, 2018

Earlier Updates

The following table describes important additions to the AppStream 2.0 service and the *Amazon AppStream 2.0 Administration Guide* documentation before June 4, 2018.

> **Note**
>
> Individual AppStream 2.0 base image and AppStream 2.0 agent releases are not included in this table. For information about these releases, see AppStream 2.0 Base Image and Managed Image Update Release Notes and AppStream 2.0 Agent Release Notes.

Change	Description	Date
Administrative controls for data transfer	Updated the "Create a Stack" section in "Create AppStream 2.0 Fleets and Stacks" and updated other content as needed	May 24, 2018
New region: Frankfurt	Updated the relay state endpoint table in "Setting Up SAML" and other content as needed	March 28, 2018
Custom branding	Created "Add Your Custom Branding to Amazon AppStream 2.0" and updated other content as needed.	March 26, 2018
Image copy	Updated "Tutorial: Create a Custom Image" and updated other content as needed.	February 23, 2018
New regions: Singapore and Sidney	Updated the relay state endpoint table in "Setting Up SAML" and other content as needed.	January 24, 2018
Resource tagging	Created "Tagging Your Amazon Amazon AppStream 2.0 Resources" and updated other content as needed.	December 15, 2017

Change	Description	Date
Managed AppStream 2.0 agent updates	Created "Amazon AppStream 2.0 Agent Version History" and updated other content as needed.	December 7, 2017
On-Demand fleets	Added "Fleet Type" section to “Amazon AppStream 2.0 Fleets and Stacks” and updated other content as needed.	September 19, 2017
Instance types	Created "Amazon AppStream 2.0 Instance Families" and updated other content as needed.	July 25, 2017
Active Directory	Created "Using Active Directory with Amazon AppStream 2.0" and updated other content as needed.	July 24, 2017
User pool	Created "Manage Access Using the AppStream 2.0 User Pool" and updated other content as needed.	June 15, 2017
Security groups	Added "Security Groups" section to "Network Settings for Amazon AppStream 2.0" and updated other content as needed.	May 26, 2017
Home folders	Created "Enable and Administer Home Folders for Your AppStream 2.0 Users" and updated other content as needed.	May 18, 2017
Default internet access	Created "Network Settings for Amazon AppStream 2.0" and updated other content as needed.	April 21, 2017
Fleet automatic scaling	Created "Fleet Auto Scaling for Amazon AppStream 2.0" and updated other content as needed.	March 23, 2017

Change	Description	Date
Fleet management	Created "Amazon AppStream 2.0 Fleets and Stacks" and updated other content as needed.	February 22, 2017
SAML 2.0 support	Created "Single Sign-on Access to AppStream 2.0 Using SAML 2.0" and updated other content as needed.	February 15, 2017
Image builders	Created "AppStream 2.0 Image Builders" and updated other content as needed.	January 19, 2017
Initial documentation release	Created the initial release of the Amazon AppStream 2.0 Administration Guide.	December 01, 2016