Best Practices for Deploying Amazon AppStream 2.0

Best Practices for Deploying Amazon AppStream 2.0:

Table of Contents

Best Practices for Deploying Amazon AppStream 2.0

Publication date: **January 19, 2022** (*Document revisions*)

Abstract

This whitepaper outlines a set of best practices for the deployment of Amazon AppStream 2.0 . The paper covers Amazon Virtual Private Cloud (VPC) design, image creation and management, fleet customization, and fleet auto scaling strategies. It includes user connection methods, authentication, and integration with Microsoft Active Directory. This paper also includes recommendations for designing AppStream 2.0 security, monitoring, and cost optimization.

This whitepaper was written to enable quick access to relevant information. It is intended for network engineers, application delivery specialists, directory engineers, or security engineers.

Introduction

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. It scales automatically, and provides access to your users on demand. AppStream 2.0 provides end users access to the applications they need on the device of their choice, with a responsive user experience, indistinguishable from natively installed applications.

The following sections provide details about Amazon AppStream 2.0, explain how the service works, describe what you need to launch the service, and tell you what options and features are available for you to use. When deploying AppStream 2.0 for end users, it is important to implement best practices to provide an outstanding user experience. Additionally, companies of all sizes benefit from cost optimization that reduces monthly operational costs.

Key concepts

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

- **Image** — An *image* is a pre-configured instance template. An image contains applications that you can stream to your users, and default Windows and application settings to enable your users to get started with their applications quickly. AWS provides base images that you can use to 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.
- **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 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.
- **Fleet** — A *fleet* consists of fleet instances (also known as streaming instances) that run the image that you specify. You can set the desired number of streaming instances for your fleet, and configure policies to scale your fleet automatically based on demand. Note that each user requires one instance.
- **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 Amazon Elastic Compute Cloud (Amazon EC2) instance that is made available to a single user for application streaming. After the user's session completes, the instance is terminated by Amazon EC2.

VPC design

Design guidelines

Deploy AppStream 2.0 into a dedicated VPC. When designing the AppStream 2.0 VPC, size for forecasted growth. Reserve IP address capacity for new use cases, and additional Availability Zones (AZs) that may be added at a later time. A fundamental design point of AppStream 2.0 is that only one user can consume an AppStream 2.0 instance. When allocating IP space, think one user as one IP address per AppStream 2.0 instance. With AppStream 2.0, it is possible for a user to consume multiple AppStream 2.0 instances. Therefore, planning IP space must also account for use cases that require additional AppStream 2.0 instances.

Although the maximum size of a VPC Classless Inter-Domain Routing (CIDR) is /16, AWS recommends not over-allocating private IP addresses. It is possible to extend the size of the VPC through additional CIDRs, but there is a limit to this; therefore, allocate what is needed from the onset.

If the AppStream 2.0 deployment is joined to an Active Directory domain, the DHCP options set for the VPC must have the domain DNS configured. The domain name server should specify the DNS IP addresses that are either authoritative for the Active Directory domain, or the DNS should forward DNS requests to the authoritative DNS instances for the Active Directory domain. Also, the VPC must have `enableDnsHostnames` and `EnableDnsSupport` configured.

Availability Zones

An Availability Zone (AZ) is one or more discrete data centers with redundant power, networking, and connectivity in an AWS Region. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

Amazon AppStream 2.0 requires only one subnet for a fleet to launch in. The best practice is to configure a minimum of two Availability Zones, one subnet per unique Availability Zone. To optimize fleet auto scaling, use more than two Availability Zones. Scaling horizontally has the added benefit of adding IP space in subnets for growth, which is covered in the following Subnet sizing section of this document. The AWS Management Console provides for only two subnets to be specified during the creation of a fleet. Use the AWS Command Line Interface (AWS CLI) or AWS CloudFormation to allow for more than two subnet IDs.

Subnet sizing

Dedicate subnets to AppStream 2.0 fleets to allow for flexibility in routing policies, and Network Access Control List. Stacks will likely have separate resource requirements. For example, AppStream 2.0 Stacks can have isolation requirements giving way to separate rule sets. When several Amazon AppStream 2.0 fleets use the same subnets, ensure the sum of all fleets' **Maximum Capacity** doesn't exceed the total number of IP addresses available.

If the maximum capacity for all fleets in the same subnet could, or has, exceeded the total number of IP addresses available, migrate fleets to dedicated subnets. This prevents automatic scaling events from exhausting allocated IP space. If the total capacity for a fleet exceeds the allocated IP space of the subnets assigned, use the API, or AWS CLI "update fleet" to assign more subnets. For more information, refer to Amazon VPC quotas, and how to increase them.

It is a best practice to scale out the number of subnets, sizing subnets accordingly while reserving capacity to grow in your VPC. Additionally, ensure that AppStream 2.0 fleet maximums do not exceed the total IP space allocated by subnets. For every subnet in AWS, five IP addresses are reserved when calculating the total amount of IP space. Using more than two subnets and scaling horizontally offers several benefits, such as:

- Greater resilience from an Availability Zone failure
- Greater throughput when automatic scaling fleet instances
- More efficient usage of private IP addresses, avoiding IP burn

When sizing subnets for Amazon AppStream 2.0, consider the total number of subnets, and the expected peak concurrency during peak utilization. This can be monitored using (`InUseCapacity`) plus reserved capacity (`AvailableCapacity`) for a fleet. In Amazon AppStream 2.0, the sum of consumed and available-to-be-consumed AppStream 2.0 fleet instances is labeled `ActualCapacity`. To properly size total IP space, forecast the required `ActualCapacity`, and divide by the number of subnets, minus one subnet for resilience, assigned to the fleet.

For example, if the anticipated maximum number of fleet instances at peak is 1000, and the business requirement is to be resilient in one Availability Zone failure, 3 x/23 subnets satisfy the technical and business requirements.

- /23 = 512 Hosts — 5 Reserved = 507 fleet instances per subnet
- 3 subnets — 1 subnet = 2 subnets
- 2 subnets x 507 fleet instance per subnet = 1,014 fleet instances at peak

Subnet sizing example

While 2 x /22 subnets would also satisfy resiliency, consider the following:

- Instead of 1,536 IP addresses being reserved, using two AZs results in 2,048 IP addresses being reserved, wasting IP addresses that could go to other functions.
- If one AZ becomes inaccessible, the ability to scale out fleet instances is limited by the throughput of an AZ. This can extend the duration of `PendingCapacity`.

Subnet routing

It is a best practice to create private subnets for AppStream 2.0 instances, routing to the public internet through a centralized VPC for outbound traffic. Inbound traffic for the AppStream 2.0 session streaming is handled through Amazon AppStream 2.0 service via Streaming Gateways: you do not need to configure public subnets for this.

Intra-Region connectivity

For AppStream 2.0 fleet instances joined to an Active Directory Domain, configure Active Directory Domain Controllers in a Shared Services VPC in each AWS Region. Sources for Active Directory can be either Amazon EC2-based Domain Controllers or AWS Microsoft Managed AD. Routing between the shared services and AppStream 2.0 VPCs can be either through a VPC peering connection or a transit gateway. Although transit gateways solve the complexity of routing at scale, there are a number of reasons why VPC peering is preferable in most settings:

- VPC peering is a direct connection between the two VPCs (no extra hop).
- There is no hourly charge, just the standard data transfer rate between Availability Zones.
- There is no limit on bandwidth.
- Support for accessing Security Groups between VPCs.

This is especially true if AppStream 2.0 instances connect to application infrastructure and/or file servers with large datasets in a shared service VPC. By optimizing the path to these commonly accessed resources, VPC peering connection is preferred, even in designs where all other VPC and internet routing are performed via transit gateway.

Outbound internet traffic

While routing directly to shared services is mostly optimized through a peering connection, outbound traffic for AppStream 2.0 can be designed by creating a single internet exit point from multiple VPCs using AWS Transit Gateway. In a multi-VPC design, it is a standard practice to have a dedicated VPC that controls all outgoing internet traffic. With this configuration, Transit Gateways have greater flexibility, and control of routing over standard routing tables attached to subnets. This design also supports transitive routing without additional complexity, and removes the need for redundant network address translation (NAT) gateways, or NAT instances in each VPC.

Once all outbound internet traffic is centralized into a singular VPC, NAT gateways or NAT instances are a common design choice. To determine which is best for your organization, view the administration guide for comparing NAT gateways and NAT instances. AWS Network Firewall can extend protection beyond security group and network access control levels by protecting at the route level and offering stateless and stateful rules from layers 3 through 7 in the OSI model. For more information, refer to Deployment models for AWS Network Firewall. If your organization has chosen a third-party product that performs advanced features such as URL filtering, deploy the

service into your outbound internet VPC. This can replace NAT gateways or NAT instances. Follow the guidelines provided by the third-party vendor.

On-premises

When connectivity to on-premises resources is required, especially for AppStream 2.0 instances joined to Active Directory, establish a highly resilient connection through AWS Direct Connect.

VPC endpoints

Amazon S3 VPC endpoint

Many Amazon AppStream 2.0 deployments require user state persistence through home folders and application settings. Enable private communication to these Amazon Simple Storage Service (Amazon S3) locations, as this avoids using the public internet. You can achieve this through a VPC endpoint gateway. A VPC endpoint gateway is preferred over the AWS PrivateLink for Amazon S3 because:

- It is cost optimized for AppStream 2.0 network access requirements
- Amazon S3 bucket access is not required from on-premises resources
- A custom policy document can be used to restrict access only from the AppStream 2.0 instances

Once you create the VPC endpoint gateway, it is a best practice to secure the privatized connection by creating a custom policy. Custom policy starts with the Amazon Resource Name (ARN) of the AppStream 2.0 service Identity and Access Management role. Explicitly specify the S3 actions required for user state persistence.

> **Note**
>
> The following example in the `Resources` section specifies the state home folder path first and the applications settings path second.

Example

```
{
```

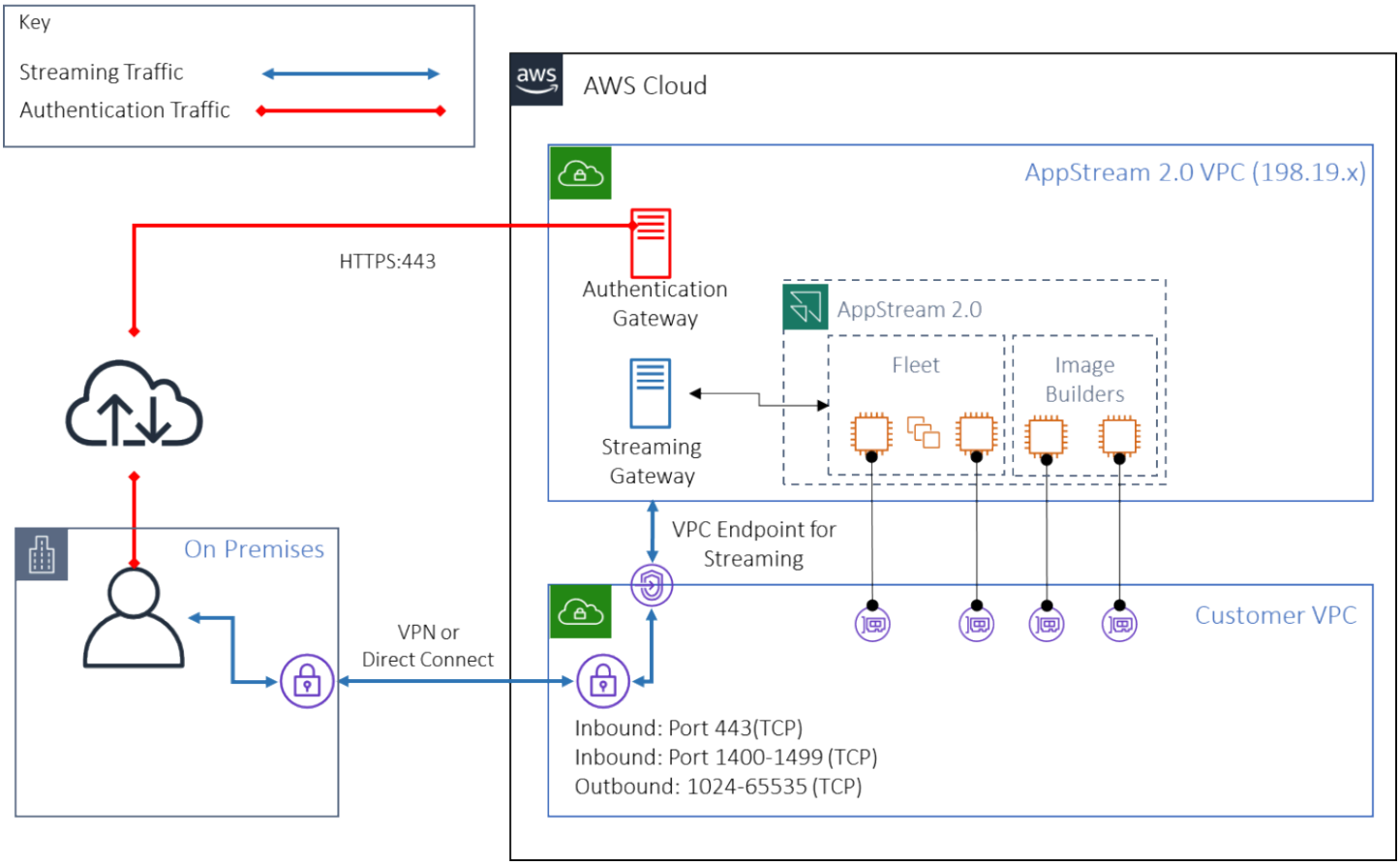
```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Allow-AppStream-to-access-home-folder-and-
application-settings",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:sts::account-id-without-hyphens:assumed-
role/AmazonAppStreamServiceAccess/AppStream2.0"
      },
      "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-*"
      ]
    }
  ]
}
```

Amazon AppStream 2.0 API interface VPC endpoint

In design scenarios where API and CLI commands to Amazon AppStream 2.0 originate in your VPC, privatize these programmatic calls through an interface VPC endpoint.

Amazon AppStream 2.0 streaming interface VPC endpoint

While it is possible to route Amazon AppStream 2.0 streaming traffic through an interface VPC endpoint, use this configuration with caution. The default streaming behavior through the public internet is the most efficient and performant delivery method for Amazon AppStream 2.0 streaming traffic.



Amazon AppStream 2.0 streaming interface VPC endpoint

As shown in the previous figure, the public internet is the most efficient path to Amazon AppStream 2.0 Streaming Gateways. Routing through the customer-managed VPC and networking adds complexity and latency. It also adds data transfer fees over AWS Direct Connect.

> **Note**
>
> Only streaming is supported by the VPC endpoint, and authentication must still take place over the public internet. Prerequisite access such as SAML Single Sign-On (SSO) Identity Provider (IdP) remain a requirement that are accessible only through the public internet.

Image creation and management

When launching a fleet or image builder in AppStream 2.0, you must select one of the AppStream 2.0 base images. Administrators can then build on the base image to add their own applications and configuration settings.

There are key considerations when building an image to ensure applications work correctly and securely. In addition, there are design considerations for how that image will be maintained.

Building an AppStream 2.0 image

When building a new image, it is important to consider the following:

- Operating system
- Applications
- User profile
- Security
- Performance
- Agent version
- Image Assistant CLI

Building an AppStream 2.0 image

In November 2021, AppStream 2.0 launched support for Amazon Linux 2. With this announcement, AppStream 2.0 now supports four platform types:

- Windows Server 2012 R2
- Windows Server 2016
- Windows Server 2019
- Amazon Linux 2

It's possible that you may have to choose a particular platform based on what is required by your application (for example, if your application requires Windows, Amazon Linux 2 will not be an

option). Beyond application requirements, reference the following comparison matrix to help you choose which platform type best fits your use case and environment:

Table 1 — Platform types, when to use them, and pricing

Platform type	When to use	Fleet pricing*
Windows Server (2012 R2, 2016 or 2019)	Your application can be run only in Windows (and it does not support Amazon Linux 2). You wish to domain join your streaming instances. You wish to use existing Group Policy on your AppStream 2.0 streaming instances (Linux does not adhere to Group Policy, but you can use Session Scripts to automate configuration when a session starts). You will use Desktop View and your users prefer the Windows desktop experience. You prefer to use the Image Assistant application, which provides a step-by-step wizard, to create your application catalog and image. Currently , you must create your Amazon Linux 2 image using terminal commands (see this tutorial for more info). You want to use Application Settings Persistence. Enabling application settings persisten ce is currently not supported for Linux-based stacks.	RDS SAL (Microsoft Remote Desktop Services Subscriber Access License) fee of $4.19 per month for each unique user** plus the following: 1. $0.10 per hour for Always-On, On-Demand fleets 2. $0.15 per hour for Elastic fleets

Platform type	When to use	Fleet pricing*
Amazon Linux 2	You want to take advantage of lower cost streaming instances and avoid RDS SAL license fees.Your applications are compatible with Amazon Linux 2	Linux instances are lower cost compared to Window instances. With Linux, you pay no RDS SAL fees and the following hourly fees: 1. $0.084 per hour for Always-On, On-Demand fleets 2. $0.112 per hour for Elastic fleets

* Based on stream.standard.medium in N Virginia Region

** Eligible customers can bring their own license to eliminate AWS RDS SAL fees. See the AppStream 2.0 pricing page for more details. Education customers may also be eligible for a special. Schools, universities, and certain public institutions may qualify for a reduced Microsoft RDS SAL user fee.

Applications

Prior to installing applications, it is important to review application requirements such as application dependencies and hardware requirements. After successfully installing applications on image builder instances, make sure to switch users and test applications under the test user context.

When planning your application deployment, be aware of the service endpoints and quotas. Additionally, clean up installer and helper files to optimize total C Drive space prior to creating an image. As a reminder, the AppStream 2.0 instances have one 200 GB fixed-size volume. Optimizing on disk space after installations is a best practice to ensure that fixed-size volume is never exceeded.

If you would like to modify the catalog of applications your users can access in real-time, dynamic application framework provides API operations. 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. This feature requires an AppStream 2.0 fleet that is

joined to a Microsoft Active Directory domain. For more information, refer to Using Active Directory with AppStream 2.0.

App blocks

App blocks represent the setup script and application files necessary to launch the applications your users will use. The virtual hard disk (VHD) can be any object from Amazon S3. It is recommended that this object be less than 1.5GB, since it has to fully downloaded before the user can access the application.

Optimizing app blocks

For Windows-based fleets, it is recommended you create a VHDX file to contain your application. For Linux-based fleets, it is recommended you create an image (IMG). These virtual disk should be created as small as possible, to host the application files. Virtual disks can be zipped to further decrease their size. In the setup script, you will need to unzip the disk before mounting. The example Windows PowerShell setup script has the unzip functionality included. There is a trade off between expanding an archive (zip) and download speed. Some testing may be necessary to find a balance that offers the fastest application launch time.

Updating applications

Applications can have both minor and major changes. For minor updates, use enable versioning on the Amazon S3 bucket that hosts your app block files. This setting allows administrators to roll back to previous versions of a specific application by changing the version of the application VHD object in question without changing the app block configuration. With major updates, create a new App block for the updated VHD. This will allow administrators to separate major application changes at the app block level opposed to the versioning level, which provides a more organized approach for administrative application management.

User profile customization

Amazon AppStream 2.0 is by design a non-persistent application and desktop solution. When a user session is terminated, both system and user changes are terminated as well. Enable application settings persistence only when required. It can add overhead to the logon process, and cost considerations for the required S3 storage.

In situations where application settings persistence is required, AWS recommends securing that connection through custom policy and S3 VPC gateway endpoint. Evaluate the overall application

settings size, and minimize the settings saved in application settings persistence to optimize cost and performance.

User profile customization can be configured on an AppStream 2.0 Image Builder instance. This includes adding and modifying registry keys, adding files, and other user specific configurations. From the AppStream 2.0 Image Assistant, there is an option to create a user profile. This copies the template user profile to the default user profile. After the image is deployed to a fleet, end users who stream sessions from the fleet will have their user profile created from the default user profile. It is important to consider minimizing the user profile size, especially when Application Settings Persistence is enabled. By default, the maximum VHDx size for user profile is 1 GB. Each time a streaming session starts, a user profile VHDx file is downloaded from an S3 bucket. This increases the streaming session preparation time and introduces a risk of exceeding the limit, which will cause a failure of the user profile mount using the VHDx file.

For use cases which require a user profile larger than 1 GB, AWS recommends using alternative methods to store profiles. For example, using Roaming profiles, or FSLogix Profile Containers on shared storage such as Amazon FSx for Windows File Server. For more information, refer to Use Amazon FSx for Windows File Server and FSLogix to Optimize Application Settings Persistence on Amazon AppStream 2.0.

Security

There are different security measurements developers need to consider. AppStream administrators are responsible for installing and maintaining the updates for the Windows operating system, your applications, and their dependencies. For additional guidance on keeping base images up to date, refer to Keep Your AppStream 2.0 Image Up-to-Date for additional guidance on keeping base images up to date.

By default, AppStream 2.0 allows users or applications to start any program on the instance, beyond what is specified in the image application catalog. This is useful when your application relies on another application as part of a workflow, but you don't want the user to be able to start that dependent application directly. For example, your application starts the browser to provide help instructions from the application vendor's website, but you don't want the user to start the browser directly. In some situations, you may want to control which applications can be launched on the streaming instances. Microsoft AppLocker is application control software that uses explicit control policies to enable, or disable, which applications a user can run.

Antivirus software can adversely affect streaming sessions and image builder instances. AWS recommends that you do **not** enable automatic updates for the antivirus software. For more information on Windows Defender, refer to Antivirus Software.

Performance

Before creating a new image, it is important to test applications as a test user. Testing as a test user enables you to ensure that applications can run under a non-administrator user context. Additionally, check application performance and user experience using built-in tools such as Task Manager and Performance Monitor. It is a best practice to monitor resource utilization such as CPU, memory, and GPU memory. If there is CPU, memory, or GPU memory resource constraint, consider upgrading the instance type. To enhance performance:

- Disable browser pop-up windows
- Disable Enhanced IE Security

AppStream 2.0 agent version selection

When creating a new image, you can opt to use the latest AppStream 2.0 agent software, or not update. Each version of the AppStream 2.0 agent software includes bug fixes and feature enhancements. Keep your image with the most up-to-date software. Review mechanisms for this in the *Image updates* section of this document.

You can choose the **Use latest agent** option. This option ensures that on start, the latest AppStream 2.0 agent is always installed. However, unexpected changes may affect user experiences, and an agent update can increase the time to start an instance. Updating a base image requires recreation of the image. It is also important that you perform testing before rolling out the updated image to production to minimize startup time.

Image Assistant Command Line Interface (CLI)

For developers who want to automate or programmatically create AppStream 2.0 images, use the Image Assistant CLI. This is available on image builders with the AppStream 2.0 agent software released on or after July 26, 2019. 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 the default application and Windows settings for your users.
6. Use the Image Assistant `create-image` CLI operation to create the image.

For more information, refer to Create Your AppStream 2.0 Image Programmatically by Using the Image Assistant CLI Operations.

Managing users' streaming experience

Customization using session scripts

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.

Specify session scripts within an AppStream 2.0 image. For more information on configuring session scripts, review the administration guide's section on using session scripts to manage your user's experience. Used with a network share or AWS Identity and Access Management (IAM) profile, you can use session scripts to retrieve additional scripting from a storage location. With this additional scripting, you can run further user experience optimization. This can minimize the number of images and fleets required to deliver application environments to your users.

Using Active Directory Group Policy

If you are planning to use AppStream 2.0 fleets in an Active Directory domain, you can use Group Policies Objects (GPOs) to manage user experience. GPOs can be assigned to the Organizational

Unit (OU) in which the AppStream 2.0 instances are created. To simplify image creation, launch the base AppStream 2.0 Image in an OU that blocks inheritance. This prevents other domain policies impacting the AppStream 2.0 user experiences. Deploy each fleet into its dedicated OU, with unique GPOs establishing the environment allows the one-to-many consolidated benefit of AppStream 2.0 image management.

An example of using Group Policy is to specify image set different Internet Explorer homepages for each AppStream 2.0 fleet.

Image updates

Software patching is critical for the security and performance of compute resources. Frequent patching is listed as a best practice in the Security Pillar of the Well-Architected Framework.

When your image is built and deployed, there are four categories of software that require patching in your AppStream 2.0 image:

- **Applications and dependencies** — You are responsible for patching the applications and dependencies in your images.
- **Microsoft Windows operating system** — You are responsible for installing and maintaining updates for Windows.
- **Software components** — These are drivers, agents, and other software that is required for AppStream 2.0 operation (for example, the Amazon CloudWatch agent). AppStream 2.0 periodically releases new base images that contain new agents and drivers. You can rebuild your image using the latest base to bring the software components on their images to the latest baseline. The process to rebuild an image on the latest base can be time-consuming and cumbersome when there are many applications, or with complex application installs.
- **AppStream 2.0 agent** — You can choose **Always use the latest agent version** in Image Assistant. With this option, streaming instances that are launched from the image automatically use the latest version of the agent.

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. This managed method **does** update service and Microsoft operating system components, but it **does not** allow you to update your application

components. It is a best practice to use this method when application installs are complex, or require manual configuration.

- Update the AppStream 2.0 Agent Software by Using Managed AppStream 2.0 Image Versions – This update method provides the latest AppStream 2.0 agent software. This method **does** allow you to update your application components.

Fleet customization

Fleet type

When creating a fleet, customers must choose a fleet type. Each fleet type provides different benefits for user experience, cost and maintenance overhead. Regardless of the fleet type chosen, each option supports both the Windows and Linux platform types, and Desktop View or Application View.

Customers can now choose from the following fleet types:

- **Always-On** — This fleet type provides users with instant-on access to their apps. You will be charged for all running instances in your fleet even if no users are streaming apps.
- **On-Demand** — Select this fleet type to optimize your streaming costs. With an on-demand fleet, users will experience a start time of about one to two minutes for their session. However, you will only be charged the streaming instance fees when users are connected, and a small hourly fee for each instance in the fleet that is not streaming apps.
- **Elastic** — Elastic fleets can be used for applications that don't require installation and can be run from a virtual hard disk (VHD). Elastic fleets don't support AppStream 2.0 images, nor do they require scaling policies. You are charged only for the duration of a streaming session.

Table 2 — Amazon AppStream 2.0 fleet types

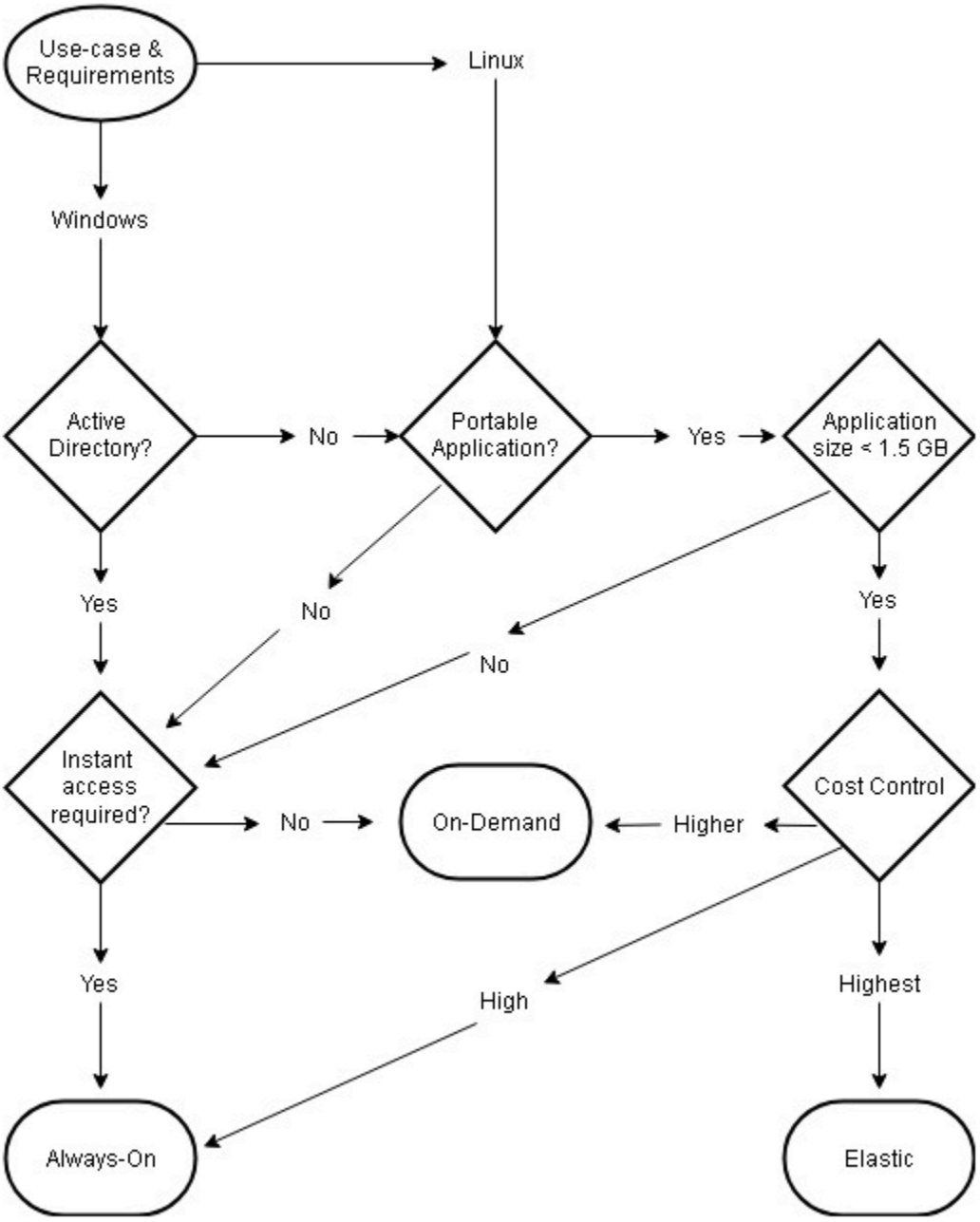
Fleet type	When to use	User experience	Pricing model	Notes
Always-On	Your users require instant access to applications when they start a session. You will not have significant excess capacity in your fleet,	Instant access to applications	You pay full price for every instance that is available in your fleet (regardless of whether it's being used for a session).	Supports custom image and scaling policies.

Fleet type	When to use	User experience	Pricing model	Notes
	perhaps because your usage patterns are predictable and you can reliably control costs with scaling policies.			

Fleet type	When to use	User experience	Pricing model	Notes
On-Demand	You must maintain significant excess capacity in your fleetsYou want the most cost optimized environment and don't want to pay full price for unused capacityYour users can wait one to two minutes to access their applications after starting a session.You are using larger instance types. The hourly cost of a running instance is much more expensive than the stopped instance fee.	Users wait one to two minutes to access their applications after starting a session.	You pay full price only for streaming instances with an active session, and then a small hourly cost for idle instances.	Supports custom image and scaling policies.

Fleet type	When to use	User experience	Pricing model	Notes
Elastic	Your application and its dependencies are smaller than ~1.5 GB. Every time a user starts a session in an Elastic fleet, your virtual hard disk (VHD) file must be downloaded from Amazon S3 into the session. As a result, larger VHD files (i.e. greater than 1.5 GB in size) will result in a poor end user experience.Your application is portable. That is, your application and all its dependencies can be placed onto a VHD and launched from the VHD.You do not require domain joined streaming instances	User waits 45 seconds to 3 minutes to access applications after starting session (wait time is dependent on size of Virtual Hard Disk).	You are charged only for the duration of a streaming session. Because there is no concept of idle instances with Elastic fleets, you incur no charges for unused instances.	Does not support custom image (customer provides VHD with applications) or scaling policies. Currently supports `stream.standard.small` and `stream.standard.medium` instances. If your use case requires a different instance type, please contact your AWS account team.

Fleet type	When to use	User experience	Pricing model	Notes
	(domain joining is not currently available with Elastic fleets)You want to pay only for active sessions (i.e. you don't pay for unused capacity in your fleet).Your users can wait 45 seconds or more to access their applications after starting a session.You want AWS to manage scaling for you (no scaling policies to manage).			



Fleet type use cases and requirements

Fleet sizing

Minimum capacity and scheduled scaling

When sizing your AppStream 2.0 fleet, there are several considerations that directly translate to user experience and cost. The value entered for **Minimum capacity** ensures that the number of

AppStream 2.0 instances will seldom be less than this value. After an AppStream 2.0 session is ended, if the total AppStream 2.0 instances are less than the **Minimum capacity** value, a new fleet instance starts. As always, it is important to remember one AppStream 2.0 instance maps directly to one user session, directly influencing the value for **Minimum capacity**.

Entering a value for **Minimum capacity** that is beyond the anticipated concurrency results in increased cost, although user experience is not impacted. A value that is too low results in low costs, but impacts user experience when total requests exceed the available capacity. Administrators will observe “Insufficient Capacity” errors in this type of situation. For example, waiting for `PendingCapacity` become `AvailableCapacity` is an inefficient use of the user’s time when the number of anticipated connections at the start of the day is a predictably consistent value.

Start with a minimum capacity that accommodates typical off-peak hours, and then use scheduled scaling policy to effectively reset the minimum capacity prior to the start of the work day. Do not forgot to create another scheduled scaling policy to revert the Minimum capacity to the off-peak hours. For more information about scaling policies and how to implement them, refer to the *Fleet auto-scaling strategies* section in this document.

Maximum capacity and service quotas

Setting maximum capacity might appear to be an arbitrary value, but when properly forecasted and set, it optimizes total resource consumption and cost. A value entered that is higher than the service quota for the AppStream 2.0 fleet in your AWS account can appear to be valid, but, when auto scaling events attempt to scale resources to the maximum capacity, they fail to launch, as the maximum capacity value exceeds the available service quota. Ensure that a service quota request is placed for the desired maximum capacity to ensure automatic scaling functions as your organization anticipates.

Another important consideration when setting a maximum capacity value is cost. For more information, refer to the *Optimizing costs with fleet type choice* section of this document.

Choosing Desktop View or Application View

The determination for choosing an application view or desktop view has no impact on performance or cost. Only one view is accessible at any given time per AppStream 2.0 fleet. You can change the **Stream view** option. Plan this change during off-peak business hours, as changing the stream view requires a restart of the fleet.

There is no single best practice for stream view. The impact of stream view options is summarized through the following:

- Detailed reporting for application usage through the **Usage Reports** feature for administrators
- Overall experience and workflow for end users (for example, does a full desktop address the needs of the use case or will only viewing the applications be sufficient?).

Desktop View

For use cases where all the user's workflow is performed in session, Desktop View simplifies the user experience by keeping all applications focused in one environment. Desktop View can give a more consistent user experience for deployments of more than 3-5 applications that require integration with the operating system (OS). Desktop View is effective when maintaining two separate and distinct environments. For example, a user can have concurrent access to both a production and pre-production desktop environment to validate changes to layout, configuration, and application access.

AppStream 2.0 Usage Reports creates a daily application report for Desktop View. The resulting output for application is simply 'desktop', mapping directly to the AppStream 2.0 session. For more information, refer to the **Monitoring user usage** section of this document.

Applications Only view

The Applications Only view is also effective when the AppStream 2.0 stack is intended to deliver a few applications that are intermittently required. In kiosk environments, a securely locked down delivery of applications is delivered through Application View. With Application View, AppStream 2.0 replaces the default Windows shell with a custom shell. This custom shell presents only running applications, minimizing the attack surface of the OS.

For use cases where AppStream 2.0 is used to augment an existing organization's desktop environment, the Applications Only view is preferred. Deploy the AppStream 2.0 Windows Client in native application mode to minimize user confusion by allowing full use of keyboard shortcuts.

Amazon 2.0 Usage Reports creates a daily application report for application view. For more granular reporting of application and run use, consider a third-party solution to report at the operating system level. You can use Microsoft AppLocker in reporting mode, or consider solutions that are available in the AWS Marketplace, such as Liquidware's Stratusphere UX.

AWS Identity and Access Management role configuration

If a workload requires the AppStream 2.0 end users to access other AWS services from within their session, it is a best practice to delegate access through the use of AWS Identity and Access Management (IAM) roles. IAM roles can be directly attached to your end user's session through the assignment at the fleet level. For additional best practices when using IAM roles with AppStream 2.0, see this section of the administrator guide.

Using static credentials

Some workloads may require static inputs for the IAM access keys opposed to inheriting them from the attached role. There are two methods for receiving these credentials. The first method involves storing the access keys within an AWS service and then giving your end users explicit IAM access to pull that specific value from the service. Two examples of access keys storage mechanisms is using AWS Secrets Manager or AWS SSM Parameter Store. The second method is to use the AppStream 2.0 credential provider to access the attached role's access keys. This can be done by invoking the credential provider and parsing the output for your access key and secret key. An example of how to perform this action within PowerShell follows.

```
$CMD = 'C:\Program Files\Amazon\Photon\PhotonRoleCredentialProvider
\PhotonRoleCredentialProvider.exe'
$role = 'Machine'

$output = & $CMD --role=$role
$parsed = $output | ConvertFrom-Json

$access_key = $parsed.AccessKeyId
$secret_key = $parsed.SecretAccessKey
$session_token = $parsed.SessionToken
```

Protecting your AppStream 2.0 S3 bucket

If your AppStream 2.0 workload is configured with Home Folder and/or Application Persistence, then it is a best practice to protect the Amazon S3 bucket that the persistent data is being stored in from unauthorized access or accidental deletion. The first layer of protection is to add an Amazon S3 bucket policy to prevent accidental deletion of the bucket. The second layer of protection is to add a bucket policy that aligns to the principle of least privilege. Aligning to the principle can be done by only allowing bucket access to the necessary parties.

Fleet auto scaling strategies

Understanding AppStream 2.0 instances

AppStream 2.0 fleet instances have a 1:1 user to fleet instance ratio. This means each user has their own streaming instance. The number of users you connect concurrently will determine the size of the fleet.

Scaling policies

AppStream 2.0 fleets are launched in an Application Auto Scaling Group. This allows the fleet to scale based on usage to meet demand. As usage increases, the fleet scales out, and as users disconnect, the fleet scales back in. This is controlled by setting scaling policies. You can set scheduled-based scaling, step scaling, and target tracking scaling policies. For more information about these scaling policies, refer to Fleet Auto Scaling for Amazon AppStream 2.0.

Step scaling

These policies increase or decrease the fleet capacity by a percentage of the current fleet size or a specific number of instances. Step scaling policies are triggered by AppStream 2.0 CloudWatch metrics of `Capacity Utilization`, `Available Capacity`, or `Insufficient Capacity Errors`.

When using step scaling policies, AWS recommends that you add a percentage of capacity and not a fixed number of instances. This ensures that your scaling actions are proportional to the size of your fleet. It will help to avoid situations where you scale out too slowly (because you added a small number of instances relative to your fleet size) or too many instances when your fleet is small.

Target tracking

With this policy specifies a capacity utilization level for the fleet. Application Autoscaling creates and manages CloudWatch alarms that trigger the scaling policy. This adds or removes capacity to keep the fleet 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. When configuring target tracking, consider the scaling cooldown to ensure scale-out and scale-in happen in desired intervals.

Target tracking is effective for high churn situations. Churn is when a large number of users start, or end, sessions in a short period of time. You can identify churn by examining CloudWatch metrics for your fleet. Periods of time when your fleet has non- zero pending capacity without change (or with very little change) in desired capacity indicate that high churn is likely occurring. In high churn situations, configure target tracking policies where (100 – target utilization percent) is more than the churn rate in a 15-minute period. For example, if 10% of your fleet will be terminated in 15 minutes due to user turnover, set a capacity utilization target of 90% or less to offset high churn.

Scheduled-based scaling

These policies enable you to set the desired fleet capacity based on a time-based schedule. This policy is effective when you understand login behavior, and can predict changes in demand.

For example, at the start of the work day, you might expect 100 users to request streaming connections at 9:00 AM. You can configure a scheduled-based scaling policy to set the minimum fleet size to 100 at 8:40 AM. This allows the fleet instances to be created and become available at the start of the work day, and allows 100 users to connect at the same time. You can then set another scheduled policy to scale in the fleet to a minimum of ten at 5:00 PM. This enables you to save cost, as the demand for sessions after hours is less than during the work day.

Scaling policies in production

You can choose to combine different types of scaling policies in a single fleet to help define precise scaling policies for your user behavior. In the previous example, you can combine the scheduled scaling policy with target tracking or step scaling policies to maintain a specific level of utilization. The combination of scheduled scaling and target tracking scaling can help reduce the impact of a sharp increase in utilization levels when capacity is needed immediately.

Users connected to streaming sessions when a scaling policy changes the desired number of instances are not affected by a scale-in or scale-out. Scaling policies will not end existing streaming sessions. Existing sessions will continue uninterrupted until the session is ended by the user or a fleet time-out policy.

Monitoring AppStream 2.0 usage with CloudWatch metrics can help you optimize your scaling policies over time. For example, it is common to over-provision resources during initial setup and you might see long periods of low utilization. Alternatively, if the fleet is under-provisioned, you might see high-capacity utilization and “Insufficient Capacity” errors. Reviewing CloudWatch metrics can help drive adjustments to your scaling policies to help mitigate these errors. For more

information, and examples of AppStream 2.0 scaling policies that you can use, refer to Scale your Amazon AppStream 2.0 fleets.

Best practices for scaling policy design

Combine scaling policies

Many customers choose to combine different types of scaling policies in a single fleet to increase the power and flexibility of Auto Scaling in AppStream 2.0. For example, you might configure a scheduled scaling policy to increase your fleet minimum at 6:00 AM in anticipation of users starting their work day, and to decrease the fleet minimum at 4:00 PM before users stop working. You can combine this scheduled scaling policy with target tracking or step scaling policies to maintain a specific level of utilization and scale-in or -out during the day to handle spiky usage. The combination of scheduled scaling and target tracking scaling can help reduce the impact of a sharp increase in utilization levels when capacity is needed immediately.

Avoid scaling churn

Consider whether your fleet might experience a high degree of churn due to your use case. Churn occurs when a large number of users start and then end sessions in a short period of time. This might occur when many users simultaneously access an application in your fleet for just a few minutes before signing off.

In such situations, your fleet size may drop far below the desired capacity, as instances are ended when users end their sessions. Step scaling policies may not add instances quickly enough to offset churn and, as a result, your fleet gets stuck at a certain size.

You can identify churn by examining CloudWatch metrics for your fleet. Periods of time when your fleet has non-zero pending capacity without change (or with very little change) in desired capacity indicate that high churn is likely occurring. To account for high churn situations, use target tracking scaling policies and pick a target utilization so that (100 – target utilization percent) is more than churn rate in a 15-minute period. For example, if 10% of your fleet will be ended in a 15-minute period due to user turnover, set a capacity utilization target of 90% or less to offset high churn.

Understand maximum provisioning rate

Customers who are managing AppStream 2.0 fleets for a large number of users should consider provisioning rate limits. This limit will impact how quickly instances can be added to a fleet or across all fleets within an AWS account.

There are two limits to consider:

- For a single fleet, AppStream 2.0 provisions at a maximum rate of 20 instances per minute.
- For a single AWS account, AppStream 2.0 provisions at a rate of rate of 60 instances per minute (with a burst of 100 instances per minute).

If more than three fleets are scaled up in parallel, the account provisioning rate limit is shared across these fleets (for example, six fleets scaling in parallel could each provision up to 10 instances per minute). In addition, consider the amount of time for a given streaming instance to finish provisioning in response to a scaling event. For fleets not joined to an Active Directory domain, this is typically 15 minutes. For fleets joined to an Active Directory domain, this can take as long as 25 minutes.

Given those constraints, consider the following examples:

- If you want to scale a single fleet from 0 to 1000 instances, it will take 50 minutes (1000 instances/20 instances per minute) for provisioning to complete, and then an additional 15-25 minutes for all instances to become available for end users, for a total of 65-75 minutes.
- If you want to simultaneously scale three fleets from 0 to 333 instances (for a total of 999 instances in the AWS account), it will take approximately 17 minutes (999/60 instances per minute) for all fleets to complete provisioning and then an additional 15 minutes for those instances to become available for end users, for a total of 32-42 minutes.

Utilize multiple Availability Zones

Choose multiple AZs in the Region for your fleet deployment. When you select multiple AZs for your fleet, you increase the likelihood that your fleet will be able to add instances in response to a scaling event. The CloudWatch metric PendingCapacity is a starting point to assess how optimized the fleet AZ design is in large fleet deployments. A high, sustained value for PendingCapacity can indicate a need to extend horizontal (across AZs) scaling. For more information, refer to Monitoring Amazon AppStream 2.0 Resources.

For example, if auto scaling attempts to provision instances to increase the size of your fleet and the selected AZ has insufficient capacity, auto scaling will instead add instances in the other AZs which you've specified for your fleet. For more information about Availability Zones and AppStream 2.0 design, refer to **Availability Zones** in this document.

Monitor Insufficient Capacity Error metrics

“Insufficient Capacity Error” is a CloudWatch metric for AppStream 2.0 fleets. This metric specifies the number of session requests rejected due to lack of capacity.

When you make changes to your scaling policies, it is helpful to create a CloudWatch alarm to notify you when any Insufficient Capacity Errors occur. This enables you to quickly adjust your scaling policies to optimize availability for users. The administration guide gives detailed steps to monitor your AppStream 2.0 resources.

Connection methods

When streaming sessions in AppStream 2.0, users have two connection methods available:

- **Web Browser Access** — Any HTML5-capable browser is supported. No plug- ins or downloads are required.
- **AppStream 2.0 Windows Client**

As a best practice, consider the feature and device requirements for your user's use case to align which browser, or device, best supports their requirements.

> **Note**
>
> AppStream 2.0 is not supported on devices that have screen resolutions smaller than 1024 x 768 pixels.

Summary feature and device support

Table 3 — Summary feature and device support

	Web browser access	AppStream 2.0 Windows Client
Multiple monitor (up to 2k resolution)	Supported	Supported
Multiple monitor (up to 4k resolution)	N/A	Supported
Drawing tablet support	Supported *	Supported
Touchscreen device support	Supported	N/A
USB passthrough device support	N/A	Supported
Keyboard shortcuts	Supported	Supported

	Web browser access	AppStream 2.0 Windows Client
Relative mouse offset	Supported	Supported
File transfer	Supported	Supported
Local printer redirection	N/A	Supported
Local drive redirection	N/A	Supported
Web-cam support	Supported	Supported

*Google Chrome and Mozilla Firefox only

Web browser access

AppStream 2.0 web browser access allows access to applications without the need to install a dedicated client. Users can connect using a supported HTML5-capable browser. There is no requirement for any browser plugin or extension.

Web browser access provides for a wide choice of end device operating systems and types.

AppStream 2.0 client for Windows

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 using a web browser. For example, the AppStream client enables 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 in the applications being accessed during your streaming sessions

- Interact with your remote streaming applications in much the same way as you interact with locally installed applications

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. Administrators can control users' ability to switch between native application mode and classic mode.

- **Classic mode** streams applications in the AppStream 2.0 session window. This is similar to how end users stream applications in a web browser. Use classic mode if end users prefer to stream applications in the same way as browsers, while making use of additional features such as connection for local file and printer redirection. Classic mode is the recommended default connection mode. Classic mode is the only mode supported for Desktop View.
- **Native application mode** enables end users to work with remote streaming applications in a similar way as other locally installed applications. If end users are used to working with applications installed locally, native application mode provides a seamless experience. 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. Native application mode is the recommended connection mode when users want to use application keyboard shortcuts, and readily switch between individual local and individual remote applications using keyboard shortcuts.

Client deployment and management

Users can install the AppStream 2.0 client themselves, or administrators can install the AppStream 2.0 client for them by running PowerShell scripts remotely, or repackaging the AppStream 2.0 client with customized settings.

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. After their devices are qualified, your users must share the devices with AppStream 2.0 every time they start a new streaming session.

When deploying the AppStream 2.0 client at scale, AWS recommends using the Enterprise Deployment Tool. The Enterprise Deployment Tool includes the AppStream client installation files and a Group Policy administrative template.

Custom domains

When deploying AppStream 2.0 programmatically, it is possible to create a custom domain which can provide users with a familiar experience for streaming sessions. In SAML 2.0 IdP deployments of AppStream 2.0, it is important to highlight that user access begins at the IdP, not AppStream 2.0. Users do not require AppStream 2.0 URLs, as these are provided by the IdP after authentication. Therefore, custom domain names are not required for SAML 2.0 IdP deployments.

Authentication

With AppStream 2.0, authentication can either take place outside of Amazon AppStream 2.0, or as part of the AppStream 2.0 service. Selecting how authentication will take place for your AppStream 2.0 deployment is a fundamental consideration of your design. It's not uncommon for an organization to have multiple deployments of AppStream 2.0 for different use-cases. Each use-case can have a different authentication method.

There are three types of authentication methods for AppStream 2.0:

- SAML 2.0
- User Pool
- Programmatic

Determining optimized method

Amazon AppStream 2.0 is architected to be flexible to apply to most organizational design requirements. When determining the optimized method for authentication, it is a best practice to consider the objectives and purposes of those who consume the service, and the organizational policies and procedures.

Here are some examples of combining use-cases with organizational objectives.

Table 4 — Use cases with organizational objectives

Example	Description	Authentication
Domain joined fleet instances are required	Applications installed on the AppStream image are accessible only to domain joined resources.	SAML 2.0
Heavy integration with Microsoft services	Organizational dependenc e on developing Microsoft Group Policies and backend infrastructure	SAML 2.0

Example	Description	Authentication
Existing enterprise Single Sign-on (SSO)	All new services must leverage an enterprise SSO solution that has several reporting and security processes established.	SAML 2.0
Smart card support for applications	Smart cards (such as Private Identity Verification and common access cards) for in-session authentication to streamed applications through a smart card reader.	SAML 2.0
Seasonal workforce with temporary staffing	A few months out of the year, temporary workers are assigned a small set of applications that do not include internal resources to complete activities.	User Pool
Limited IT Support	Smaller organizations with less than 50 users and limited IT staff, looking to remove the overhead of maintaining an Identity Provider (IdP)	User Pool
Independent Software Vendor (ISV)	Proprietary solution built by your organization that includes user entitlement and authentication, extending AppStream 2.0 as part of your solution.*	Programmatic

Example	Description	Authentication
Technology showcase	Completely ephemeral environment that showcases a proprietary technology as part of a guided tour of your solution with no requirement to store user information.	Programmatic
Interactive website experience	Make your website interacti ve with streaming Windows applications.**	Programmatic

*Refer to Software vendors: Deliver your applications to any user device for more information.

**Refer to Embed AppStream 2.0 Streaming Sessions for more information.

If your organization has a use-case or policy that is not listed in the examples previously given, it is a best practice to forecast the desired end state of AppStream 2.0 workflow consumption to ensure the authentication solution does not conflict with it.

Configuring your identity provider

SAML 2.0

Security Assertion Markup Language (SAML) 2.0 is a common deployment option for enabling users to use AWS resources. Various third-party SAML 2.0 identity providers support AppStream 2.0. Whether your AppStream 2.0 resources are domain joined or not, SAML 2.0 IdP requires you to use IAM.

As most IdPs generate a unique metadata.xml with specific SAML attributes for each SAML application, every AppStream 2.0 stack requires a Role that has a trusted relationship with the SAML IdP and a Policy that has a single permission to appstream:Stream with conditions that match the requirements of the SAML IdP and the ARN of the AppStream 2.0 Stack.

The AppStream 2.0 administration guide provides an example configuration for single AppStream 2.0 stack design. For multiple stack deployments, refer to the optional steps for using SAML 2.0 multi-stack application catalog.

User pool

The **User Pool** tab in AppStream 2.0 is a valid option for small proof of concepts. As a best practice, it is best to avoid user pools for any use case and organization that uses AppStream 2.0 to deliver production applications.

One important thing to note about user pools is that users' email addresses are case-sensitive; therefore it is a best practice to ensure users are educated on how to properly enter user credentials.

Streaming url

For deployments that call AppStream 2.0 resources from a centralized service (typically ISVs), programmatic authentication relies on an application to make programmatic calls to AWS to dynamically pass information and create an AppStream 2.0 session for its users. Use the API authentication method (commonly referred to as 'programmatic') when creating streaming URLs using the CreateStreamingURL operation. The user who makes the `CreateStreamingURL` call must be using a valid user or role with permission for `appstream:CreateStreamingURL`.

When creating the policy for programmatic access, it is a best practice to secure access by specifying the exact AppStream 2.0 Stack ARN in the **Resources** section in place of the default '*'. For example:

Example

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "appstream:createStreamingURL"
            ],
            "Resource": "arn:aws:appstream:us-east-
1:031421429609:stack/BestPracticesStack"
        }
    ]
}
```

> **Note**
>
> You can quickly retrieve the ARNs of your AppStream 2.0 Stacks by using the describe stacks API or AWS CLI.

AppStream 2.0 instances should start as generic instances. Through information passed to it from the application, the AppStream 2.0 instance establishes the environment using session context to make things dynamic for the user.

While local GPOs can be used to specify settings at user logon, session context is a best practice when using `CreateStreamingURL`, and passing key attributes such as Customer ID or database connection settings, to be used in the AppStream session.

Application entitlement

AppStream 2.0 can dynamically build the application catalog that is presented to users. Application entitlements are based on SAML 2.0 attributes, or by using AppStream 2.0 Dynamic Application Framework.

Attribute-based application entitlements using SAML 2.0 is recommended in most scenarios. To manage application package delivery, Dynamic Application Framework is recommended.

Integration with Microsoft Active Directory

Amazon AppStream 2.0 Image Builders and fleets can be integrated with Microsoft Active Directory. This enables you to provide a centralized method for user authentication, authorization, and to apply Active Directory Group policies to domain- joined AppStream 2.0 instances. Using AppStream fleets joined to a domain provides the same administrative benefits an on-premises environment. This includes centralized management of network file shares, user-app entitlements, roaming profiles, printer access, and other policy-based settings.

When integrating an AppStream 2.0 environment with Active Directory, it is important to note that the initial authentication to the AppStream 2.0 stack is still managed by a SAML2.0 IdP. After the user is successfully authenticated to the IdP, when the user launches a session, they must enter their domain password or a smart card authentication for the Active Directory domain.

When designing the Active Directory Domain Services (ADDS) environment that will be used with AppStream 2.0, there are two service options and many deployment scenarios available. Also, ensure that the AppStream 2.0 networking is reviewed with your Active Directory site topology owner.

Service options

Active Directory can also be deployed using AWS Managed Microsoft Active Directory (AD). AWS Managed Microsoft AD is a fully managed service that allows you to run Microsoft Active Directory. Microsoft Active Directory can also be used in a self-hosted environment, running on EC2 or on-premises.

Deployment scenarios

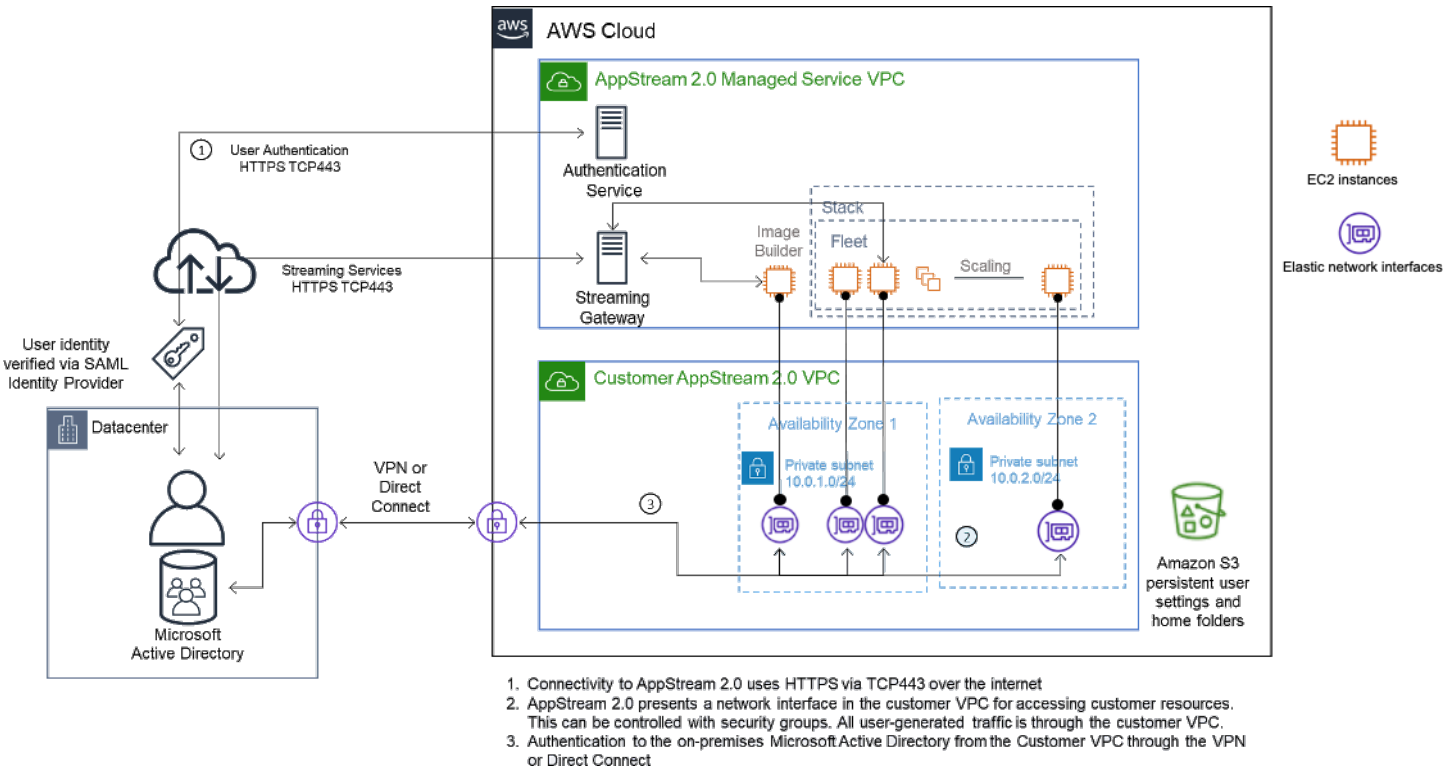
The following deployment scenarios listed are commonly used and recommended integration options for AppStream 2.0 with Microsoft Managed AD or a customer's self-managed Active Directory. All of the architecture diagrams listed below use core Amazon constructs.

- **Amazon Virtual Private Cloud (VPC)** — Creation of an Amazon VPC dedicated for AppStream 2.0 services with at least four private subnets spread across four AZs. Two of the private subnets are used for AppStream fleets and Image Builders. The remaining two subnets are used for the domain controllers on EC2 or Microsoft Managed AD).

- **Dynamic Host Configuration Protocol (DHCP) Options Set** — Provides a standard for passing configuration info to the AppStream 2.0 fleet and Image Builders that will be provisioned in the VPC. The DHCP Option Set is defined at the VPC level. It enables customers to define a specified domain name and DNS settings that will be used with the AppStream 2.0 instanced upon being provisioned.
- **AWS Directory Services** — Amazon Microsoft Managed AD can be deployed into two private subnets that will be used in conjunction with AppStream 2.0 workloads.
- **AppStream 2.0 fleets** — The AppStream 2.0 fleets or Image Builders are hosted in the AWS Managed VPC. Each AppStream 2.0 instance has two Elastic Network Interfaces (ENI). The primary interface (`eth0`) is used for management purposes and brokering the end-user connection to the instance through the streaming gateway. The secondary interface (`eth1`) is injected into the customer-VPC and can be used to access other resources in the bespoke VPC or on-premises.

Scenario 1: Active Directory Domain Services (ADDS) deployed on-premises

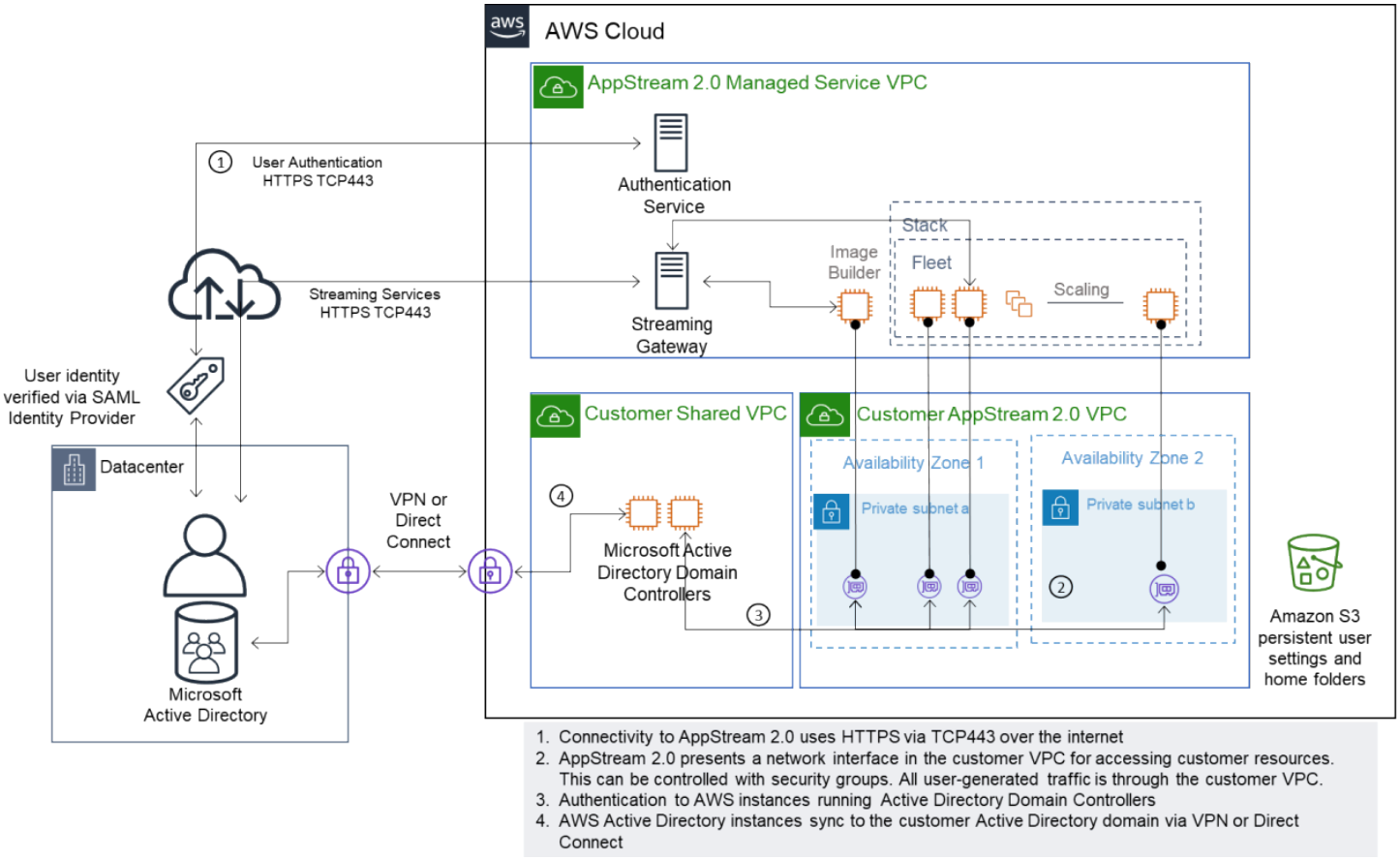
All authentication traffic traverses the VPN or Direct Connect connection from the customer VPC to the customer gateway. The advantage of this scenario is the benefit of using a possibly already deployed AD environment without having to provision additional domain controllers in the customer VPC. The disadvantage is the sole dependency on the VPN or Direct Connect to authenticate and authorize users for the AppStream 2.0 fleet. If there is any network connectivity issue, the AppStream 2.0 fleet or Image Builders would be directly impacted. Providing dual VPN tunnels or Direct Connect connections with different paths mitigates this potential risk.



Scenario 1 — Active Directory Domain Services (ADDS) deployed on-premises

Scenario 2: Extend Active Domain Services (ADDS) into AWS customer VPC

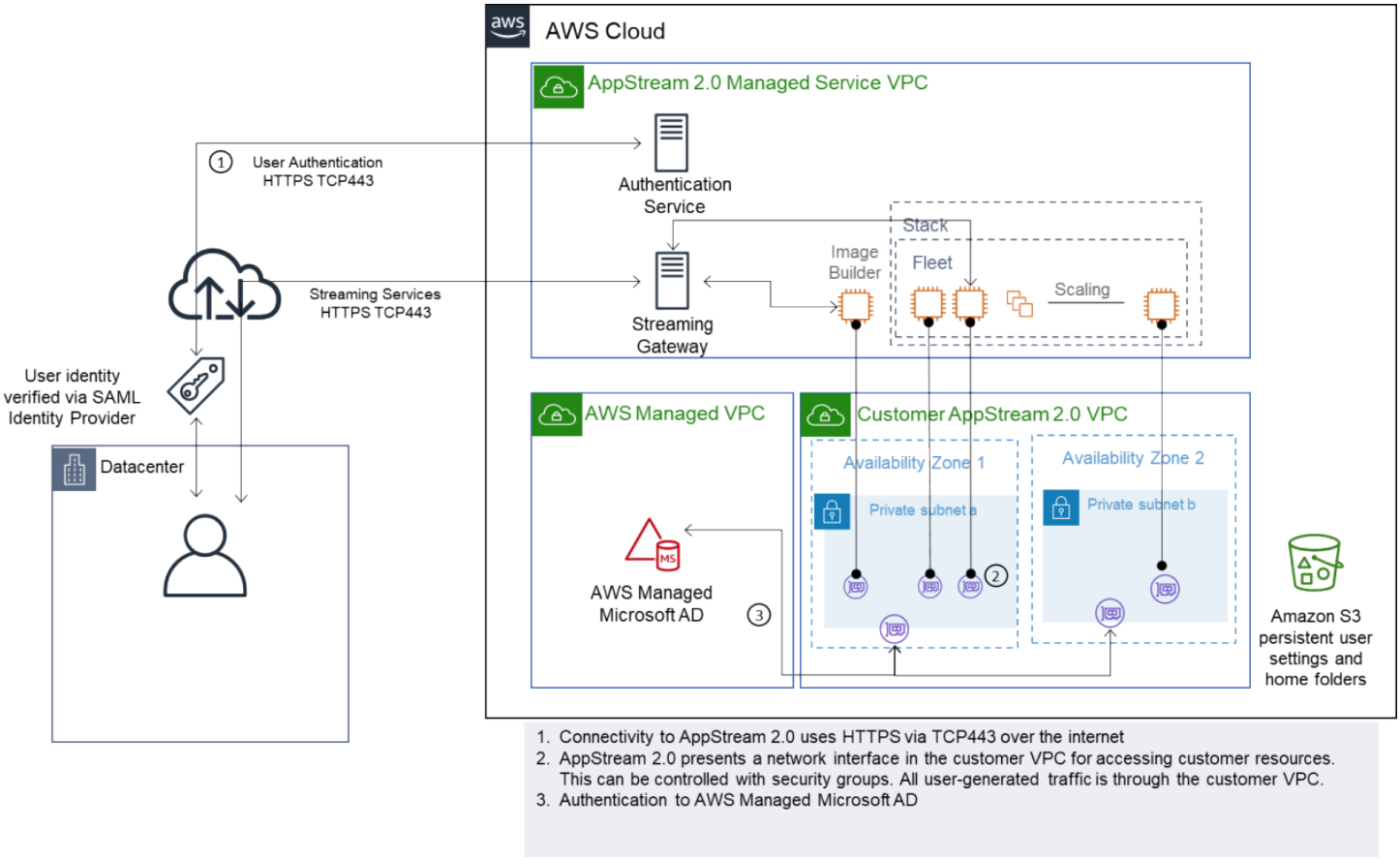
The Active Directory is extended to your customer VPC. An Active Directory site should be created for the new domain controllers in the customer VPC. The authentication traffic is routed to the domain controllers in the AWS customer VPC instead of traversing the VPN or Direct Connect connection.



Scenario 2 — Extend Active Domain Services into AWS customer Virtual Private Cloud

Scenario 3: AWS Managed Microsoft Active Directory

AWS Managed Microsoft AD is deployed in the AWS Cloud and is used as the identity and resource domain for the AppStream 2.0 fleets and Image Builders.



Scenario 3 — AWS Managed Active Directory

Active Directory Service Site Topology

An Active Directory service site topology is a logical representation of your physical network.

A site topology helps you efficiently route client queries and Active Directory replication traffic. A well-designed and maintained site topology helps your organization achieve the following benefits:

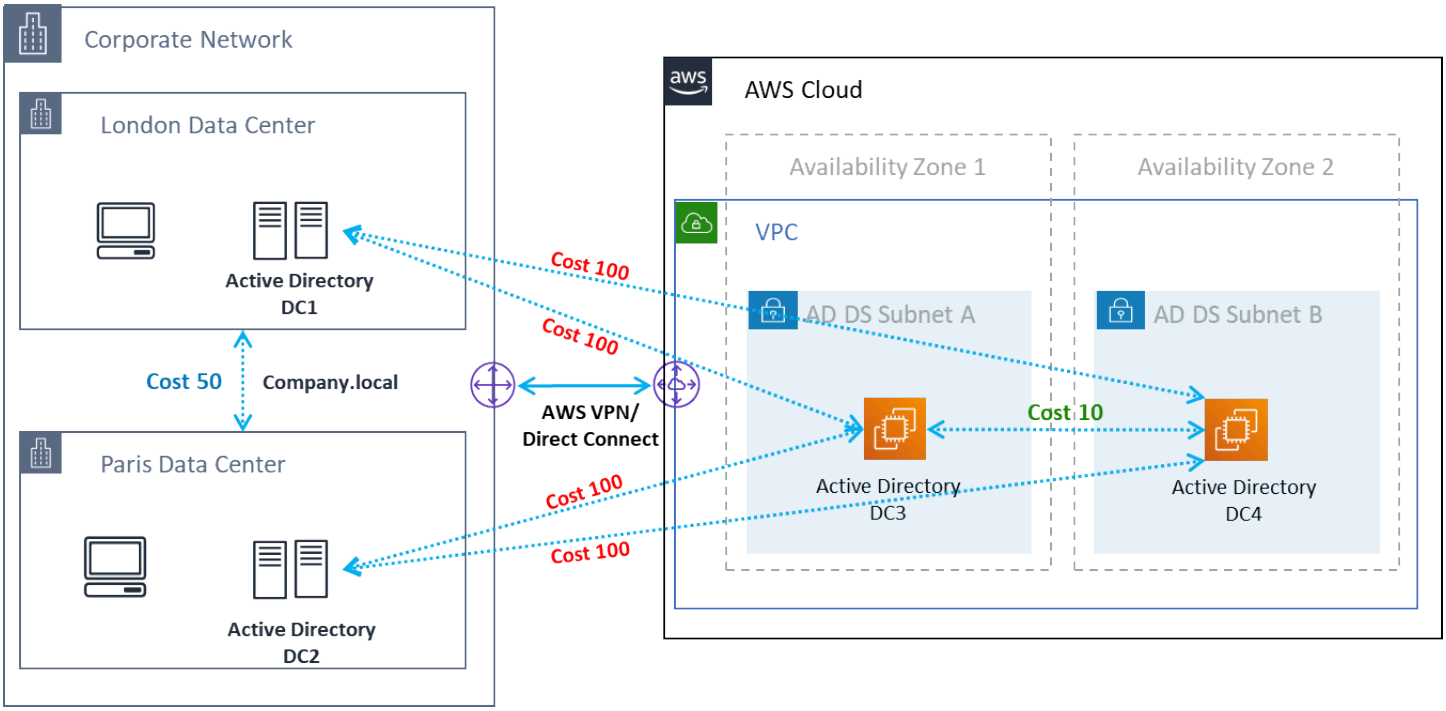
- Minimize the cost of replicating Active Directory data when synchronizing between on-premises and AWS Cloud.
- Optimize the ability of client computers to locate the nearest resources, such as domain controllers. This helps to reduce network traffic over slow wide area network (WAN) links, improve logon and logoff processes, and speed up resource access operations.

When introducing AppStream 2.0 services, ensure that the address ranges used for the AppStream 2.0 instances' subnets are assigned to the correct site for your environment.

For Scenario 1 and Scenario 2, sites and services are critical components for the best user experience in terms of logon times, and time for Active Directory resource access.

Site topology controls Active Directory replication between domain controllers within the same site and across site boundaries.

Defining the correct site topology ensures client affinity, meaning that clients (in this case, AppStream 2.0 streaming instances) use their preferred local domain controller.



Active Directory sites and services — client affinity

> **Tip**
>
> As a best practice, define high cost for site links between on-premises AD DS and the AWS Cloud. The preceding figure is an example of what costs you should assign to the site links (cost 100) to ensure site-independent client affinity.

For more information on site topology, refer to Designing the Site Topology.

Active Directory Organizational Units

AWS recommends storing the Organizational Units (OUs) configured in a single AppStream 2.0 Directory Config object. It is a best practice for each AppStream 2.0 stack to have its own OU. This

allows you the flexibility to have specific GPOs per stack. Ensure that the OUs are dedicated to AppStream 2.0 computer objects to avoid mixing AppStream 2.0-specific policies with on-premises desktops. Consider using sub-OUs for each AWS Region you deploy AppStream 2.0 into.

Active Directory computer object cleanup

AppStream 2.0 instances are ephemeral. A fleet creates and reuses Active Directory computer objects as fleets scale out and scale in.

AWS recommends creating an AD cleanup process to delete stale Active Directory computer objects that can exist after an AppStream fleet is removed.

Security

Cloud security at Amazon Web Services (AWS) is the highest priority. Security and compliance is a shared responsibility between AWS and the customer. For more information, refer to the Shared Responsibility Model. As an AWS and AppStream 2.0 customer, it is important to implement security measures on different layers such as stack, fleet, image, and networking.

Due to its ephemeral nature, AppStream 2.0 is often preferred as a secure solution to application and desktop delivery. Consider whether antivirus solutions that are commonplace in Windows deployments are relevant in your use cases for an environment that is predefined and purged at the end of a user session. Antivirus adds overhead to virtualized instances, making it is a best practice to mitigate unnecessary activities. For example, scanning the system volume (which is ephemeral) at boot, for instance, does not add to the overall security of AppStream 2.0.

The two key questions for security AppStream 2.0 are centered on:

- Is persisting user state beyond the session a requirement?
- How much access should a user have within a session?

Securing persistent data

Deployments of AppStream 2.0 can require the user state to persist in some form. It might be to persist data for individual users, or to persist data for collaboration using a shared folder. AppStream 2.0 instance storage is ephemeral and has no encryption option.

AppStream 2.0 provides user state persistence through home folders and application settings in Amazon S3. Some use cases require greater control over user state persistence. For these use cases, AWS recommends using a Server Message Block (SMB) file share.

User state and data

Because most Windows applications perform best and most securely when co-located with application data created by the user, it is a best practice to keep this data in the same AWS Region as AppStream 2.0 fleets. Encrypting this data is a best practice. The default behavior of the user home folder is to encrypt files and folders at rest using Amazon S3-managed encryption keys from the AWS key management services (AWS KMS). It is important to note that AWS Administrative Users with access to the AWS Console or Amazon S3 bucket will be able to access those files directly.

In designs that require a Server Message Block (SMB) target from a Windows File Share to store user files and folders, the process is either automatic or requires configuration.

Table 5 — Options for securing user data

SMB target	Encryption-at-rest	Encryption-in-transit	Antivirus (AV)
FSx for Windows File Server	Automatic through AWS KMS	Automatic through SMB encryption	AV installed on a remote instance performs scan on mapped drive
File Gateway, AWS Storage Gateway	By default, all data stored by AWS Storage Gateway in S3 is encrypted server-side with Amazon S3-Managed Encryption Keys (SSE-S3). You can optionally configure different gateway types to encrypt stored data with AWS Key Management Service (KMS)	All data transferred between any type of gateway appliance and AWS storage is encrypted using SSL.	AV installed on a remote instance performs scan on mapped drive
EC2-based Windows File Servers	Enable EBS encryption	PowerShell; `Set-SmbServer Configuration – EncryptData $True`	AV installed on server performs scan on local drives

Endpoint security and antivirus

The brief ephemeral nature of Amazon AppStream 2.0 instances and the lack of persistency of data means a different approach is required to ensure user experience and performance is not

compromised by activities that would be required on a persistent desktop. Endpoint Security agents are installed in AppStream 2.0 images when there is an organizational policy or when used with external data ingress e.g. e-mail, files ingress, external web browsing.

Removing unique identifiers

Endpoint Security agents may have a globally unique identifier (GUID) which must be reset during the fleet instance creation process. Vendors have instructions on installing their products in images which will ensure a new GUID is generated for each instance generated from an image.

To ensure the GUID is not generated, install the Endpoint Security agent as the last action before running the AppStream 2.0 Assistant to generate the image.

Performance optimization

Endpoint Security Vendors provide switches and setting that optimize the performance of AppStream 2.0. The settings vary between vendors and can be found in their documentation, typically in a section on VDI. Some common settings include but are not limited to are:

- Turn off boot up scans to ensure instance creation, startup and login times are minimized
- Turn off scheduled scans to prevent unnecessary scans
- Turn off signature caches to prevent file enumeration
- Enable VDI optimized IO settings
- Exclusions required by applications to ensure performance

Endpoint security vendors provide instructions for use with virtual desktop environments which optimize performance.

- Trend Micro Office Scan Support for Virtual Desktop Infrastructure - Apex One/OfficeScan (trendmicro.com)
- CrowdStrike and How to Install the CrowdStrike Falcon in the Data Center
- Sophos and Sophos Central Endpoint: How to install on a gold image to avoid duplicate identities and Sophos Central: Best practices when installing Windows Endpoints in Virtual Desktop Environments
- McAfee and McAfee Agent provisioning and deployment on Virtual Desktop Infrastructure systems

- Microsoft Endpoint Security and Configuring Microsoft Defender Antivirus for non-persistent VDI machines - Microsoft Tech Community

Scanning exclusions

If security software is installed in AppStream 2.0 instances, the security software must not interfere with the following processes.

Table 6 — AppStream 2.0 processes security software must not interfere with the following processes.

Service	Processes
AmazonCloudWatchAgent	"C:\Program Files\Amazon\AmazonCloudWatchAgent\start-amazon- cloudwatch-agent.exe"
AmazonSSMAgent	"C:\Program Files\Amazon\SSM\amazon-ssm-agent.exe"
NICE DCV	"C:\Program Files\NICE\DCV\Server\bin\dcvserver.exe" "C:\Program Files\NICE\DCV\Server\bin\dcvagent.exe"
AppStream 2.0	"C:\ProgramFiles\Amazon\AppStream2\StorageConnector\StorageConnector.exe" In the folder "C:\Program Files\Amazon\Photon\" ".\Agent\PhotonAgent.exe" ".\WebServer\PhotonAgentWebServer.exe" ".\CustomShell\PhotonWindowsAppSwitcher.exe" ".\CustomShell\PhotonWindowsCustomShell.exe"

Service	Processes
	".\CustomShell\PhotonWindowsCustomShellBackground.exe"

Folders

If security software is installed in AppStream 2.0 instances, the software must not interfere with the following folders:

Example

```
C:\Program Files\Amazon\*
C:\ProgramData\Amazon\*
C:\Program Files (x86)\AWS Tools\*
C:\Program Files (x86)\AWS SDK for .NET\*
C:\Program Files\NICE\*
C:\ProgramData\NICE\*
C:\AppStream\*
C:\Program Files\Internet Explorer\*
C:\Program Files\nodejs\
```

Endpoint security console hygiene

Amazon AppStream 2.0 will create new unique instances each time a user connects beyond the idle and disconnect timeouts. The instances will have a unique name and will build up in endpoint security management condoles. Setting unused aged machines over 4 or more days old (or lower depending on AppStream 2.0 session timeouts) to be deleted will minimize the number of expired instances in the console.

Network exclusions

The AppStream 2.0 management network range (`198.19.0.0/16`) and following ports and addresses should not be blocked by any security / firewall or antivirus solutions within AppStream 2.0 instances.

Table 7 — Ports in AppStream 2.0 streaming instances security software must not interfere with

Port	Usage
8300, 3128	This is used for establishing the streaming connection
8000	This is used for managing the streaming instance by AppStream 2.0
8443	This is used for managing the streaming instance by AppStream 2.0
53	DNS

Table 8 — AppStream 2.0 managed service addresses security software must not interfere with

Port	Usage
169.254.169.123	NTP
169.254.169.249	NVIDIA GRID License Service
169.254.169.250	KMS
169.254.169.251	KMS
169.254.169.253	DNS
169.254.169.254	Metadata

Securing an AppStream session

Limiting application and operating system controls

AppStream 2.0 gives the administrator the ability to specify exactly which applications can be launched from the web page in application streaming mode. This does not, however, guarantee that only those applications specified can be run.

Windows utilities and applications can be launched through the operating system through additional means. AWS recommends using Microsoft AppLocker to ensure that only the applications that your organization requires can be run. The default rules must be modified, as they grant everyone path access to critical system directories.

> **Note**
>
> Windows Server 2016 and 2019 require the Windows Application Identity service to be running to enforce AppLocker rules. Application access from AppStream 2.0 using Microsoft AppLocker is detailed in the AppStream Admin Guide.

For fleet instances joined to an Active Directory domain, use Group Policy Objects (GPOs) to deliver user and system settings to secure the users application and resource access.

Firewalls and routing

When creating an AppStream 2.0 fleet, subnets and a Security Group must be assigned. Subnets have existing assignments of Network Access Control Lists (NACLs) and route table(s). You can associate up to five security groups while launching a new image builder or while creating a new fleet Security Groups can have up to five assignments from the existing Security Groups. For each security group, you add rules that control the outbound and inbound network traffic from and to your instances

A NACL is an optional layer of security for your VPC that acts as a stateless firewall for controlling traffic in and out of one or more subnets. You might set up network ACLs with rules similar to your security groups in order to add an additional layer of security to your VPC. For more information about the differences between security groups and network ACLs, see the compare security groups and NACLs page.

When designing and applying Security Group and NACL rules, consider the AWS Well-Architected best practices for least privilege. *Least privilege* is a principle of granting only the permissions required to complete a task.

For customers who have a high-speed private network connecting their on premise environment to AWS (via an AWS Direct Connect), you may consider using the VPC Endpoints for AppStream, which will mean the streaming traffic will be routed via your private network connectivity rather than going across the public internet. For more information on this topic, see the AppStream 2.0 streaming interface VPC endpoint section of this document.

Data loss prevention

We'll look at two kinds of data loss prevention.

Client to AppStream 2.0 Instance Data Transfer Controls

Table 9 — Guidance for controlling data ingress and egress

Setting	Options	Guidance
Clipboard	• Copy and paste to remote session only • Copy to local device only • Disabled	Disabling this setting does not disable copy and paste within the session. If copying data into the session is required, choose Paste to remote session only to minimize the potential for data leakage.
File transfer	• Upload and download • Upload only • Download only • Disabled	Avoid enabling this setting to prevent data leakage.
Print to local device	• Enabled • Disabled	If printing is required, use network mapped printers that are controlled and monitored by your organization.

Consider the advantages of the existing organizational data transfer solution over the stack settings. These configurations are not designed to replace a comprehensive secure data transfer solution.

Controlling egress traffic from the AppStream 2.0 instance

Where data loss is a concern, it’s important to cover off what a User can access once they are inside of their AppStream 2.0 instance. What does the network exit (or egress) path look like?

It is a common requirement to have public internet access available to the end user inside their AppStream 2.0 instance, so placing a WebProxy or Content Filtering Solution in the network path needs to be considered. Other considerations include a local Antivirus application and other endpoint security measures inside the AppStream instance (see the section “Endpoint Security and Antivirus” for more information).

Using AWS services

AWS Identity and Access Management

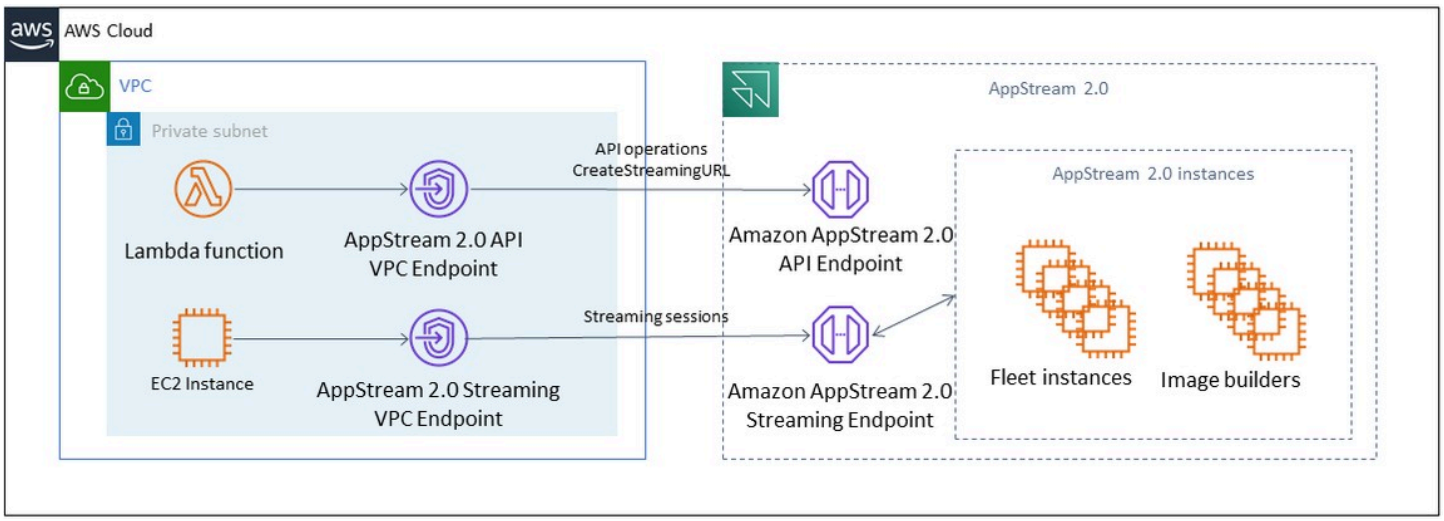
Using an IAM role to access AWS services, and being specific in the IAM policy attached to it, is a best practice that provides only the users in AppStream 2.0 sessions have access without managing additional credentials. Follow the best practices for using IAM Roles with AppStream 2.0.

Create IAM policies to protect Amazon S3 buckets that are created to persist user data in both home folders and application settings persistence. This prevents non-AppStream 2.0 administrators from access.

VPC endpoints

A VPC endpoint enables private connections between your VPC and supported AWS services and VPC endpoint services powered by AWS PrivateLink. AWS PrivateLink is a technology that enables you to privately access services by using private IP addresses. Traffic between your VPC and the other service does not leave the Amazon network. If public internet access is required only for AWS services, VPC endpoints remove the requirement for NAT gateways and internet gateways altogether.

In environments where automation routines or developers require making API calls for AppStream 2.0, create an interface VPC endpoint for AppStream 2.0 API operations. For example, if there are EC2 instances in private subnets without public internet access, a VPC endpoint for AppStream 2.0 API can be used to call AppStream 2.0 API operations such as CreateStreamingURL. The following diagram shows an example setup where AppStream 2.0 API and streaming VPC endpoints are consumed by Lambda functions and EC2 instances.



VPC endpoint

The streaming VPC endpoint allows you to stream sessions through a VPC endpoint. 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 use the VPC endpoint, the VPC endpoint setting must be enabled at the AppStream 2.0 stack. This serves as an alternative to streaming user sessions over the public internet from locations that have limited internet access and would benefit from accessing through a Direct Connect instance. Streaming user sessions through a VPC endpoint require the following:

- 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.
- Internet connectivity is required to authenticate users and deliver the web assets that AppStream 2.0 requires to function.

To learn more about restricting traffic to AWS services with AppStream 2.0, see the administration guide for creating and streaming from VPC endpoints.

When full public internet access is required, it's a best practice to disable Internet Explorer Enhanced Security Configuration (ESC) on the Image Builder. For more information, see the AppStream 2.0 administration guide to disable Internet Explorer enhanced security configuration.

Disaster recovery

Amazon AppStream 2.0 has built in redundancy across up to three availability zones. This means that if a user has an active session in an availability zone that becomes degraded, they can simply disconnect and reconnect which will reserve them a session in a healthy availability zone assuming you have capacity. While this provides high availability within the Region, it does not provide a disaster recovery solution if the service experiences issues at a regional level.

To provide a disaster recovery plan for your AppStream 2.0 users, you will first need to build out an AppStream 2.0 environment in your secondary Region. From a design perspective, this environment should have redundant connections to your on-premises environment, if applicable, and should have no dependency on the primary Region. For example, if your AppStream 2.0 fleet is domain joined, you should have additional domain controllers in the secondary Region with Sites and Services configured. From an AppStream 2.0 perspective, this environment should consist of the same fleet and stack settings that you have in your primary Region. The fleet itself should run your same base image, which can be copied to your secondary Region via the console or programmatically. If the applications that run within your AppStream 2.0 sessions have a backend dependency tied to your primary Region, that too should have regional redundancy to ensure the users can still access the application's backend if the primary Region goes down. Your service level limits in your destination Region should match your primary Region.

Identity routing

There are two distinct methods to providing access to applications in a DR scenario. At a high level, the two methods differ by how the users are directed to the failover Region. The first method is performed with a single AppStream 2.0 application configuration in your IdP and the second method is having two separate application configurations.

Method 1: Changing the relay state of your application

When users login to AppStream 2.0 from an Identity Provider (IdP), following their authentication they are relayed to a specific URL that aligns to the Region and stack they are intended to have access to. For more information around the Relay State URL, refer to the Amazon AppStream 2.0 Administration Guide. The administrator can configure a cross-Region stack built on the same AppStream 2.0 image as the primary Region for users to failover to. The administrator can control this failover by simply updating the Relay State URL to point to the failover stack. For this method to operate properly, the associated IAM policies will need to reflect access to both stacks; primary

and failover. For more details on how these IAM policies should be configured, see the following example policy.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "appstream:Stream",
            "Resource": [
            "arn:aws:appstream:PrimaryRegion:190836837966:stack/StackName",
            "arn:aws:appstream:FailoverRegion:190836837966:stack/StackName"
            ],
            "Condition": {
                "StringEquals": {
                    "appstream:userId": "${saml:sub}"
                }
            }
        }
    ]
}
```

Method 2: Configuring two AppStream 2.0 applications within your IdP

This method requires the administrator to build out two separate applications for AppStream 2.0 within the IdP. They then can either present both applications and let the user choose where to go, or they lock/hide an application until it's time to failover. This method is better aligned to the use case of having global users that move around often. Those users should be streaming from the closest endpoint, therefore having both applications assigned gives them the option to choose the application that is configured for their nearest Region. This can also be automated, for more information see this blog post.

Storage persistance

When leveraging the included data persistence features of AppStream 2.0, such as Application Persistence and Home Folder Synchronization, you will need to replicate that data to your failover region. These features store the persistent data in an Amazon S3 bucket in the given AppStream 2.0 region. To have the data persist cross region, you will need to replicate all changes on the source bucket to the failover regions AppStream 2.0 bucket. This can be done with native Amazon

S3 features, such as Amazon S3 cross region replication. Each users persistent data will reside under a folder of their hashed username. Since the username will be hashed the same cross region, simply replicating the data will provide data persistence in your secondary region. For more information about the Amazon S3 buckets used by AppStream 2.0, see this guide.

Monitoring

Using dashboards

Monitoring fleet utilization is a regular activity that can be performed through CloudWatch metrics and creating a dashboard. Alternatively, from the AppStream 2.0 console, use the **Fleet Usage** tab. Regularly monitor your fleet usage, as user behavior is not always predictable, and demand can exceed even first-rate upfront planning. A full listing of AppStream 2.0 metrics and dimensions for CloudWatch can be found in the AppStream 2.0 administration guide under Monitoring Resources.

Anticipating growth

Whenever there is a large jump in `PendingCapacity`, an auto scaling event has occurred. It is important to confirm that `AvailableCapacity` and `PendingCapacity` have an inverse relationship while new AppStream 2.0 fleet instances become available to host user sessions. Create a CloudWatch Alarm for `InsufficientCapacityError` for each AppStream 2.0 fleet to notify administrators to ensure automatic scaling is not falling behind demand.

If demand exceeds capacity and `InsufficientCapacityError` metric values are common, consider raising the minimum capacity through a Scheduled Scaling policy for the start of the work day. In addition, have a second Scheduled Scaling policy to lower the minimum capacity after the demand has been satisfied. Keep in mind that lowering the value for minimum capacity does not impact existing sessions. Lowering the minimum capacity prior to the end of the work day effectively enables scale to function as intended by lowering the value for `ActualCapacity`. This optimizes cost.

If demand is consistently unpredictable, use Target Tracking scaling policy to ensure that there is adequate `AvailableCapacity` in the AppStream 2.0 fleet to meet demand while determining usage patterns. Continue to monitor as Target Tracking uses a percentage of fleet consumption. As the total number of fleet instances grows, the total number of unused fleet instances multiplies. This can become wasteful unless the maximum capacity is set to a conservative value. Use multiple types of scaling policies (for example, Scheduled and Target Tracking) to balance reliability with cost optimization.

Monitoring user usage

Monitoring unique users, as there is a cost associated for that in the form of user fees. This user fee cost is due to Image Assistant (RDS) subscriber access licenses (SAL). Evaluating unique users can either be performed through reporting from the IdP where authentication is performed, or through usage reports.

Usage reports are stored as separate `.csv` files in your S3 bucket, which you can download and analyze using third-party business intelligence (BI) tools. You can analyze your usage data in AWS without downloading your reports or create reports over custom date ranges without concatenating multiple `.csv` files. For example, you can use Amazon Athena and Amazon QuickSight to create custom reports and visualizations of your AppStream 2.0 usage data.

Persisting application and Windows event logs

When an AppStream 2.0 instance session is complete, the instance is ended. This means all application and Windows event logs used in the session are lost. If there is a requirement to persist these application and Windows event logs, one method is to use Amazon Data Firehose to deliver them in real-time to S3 and search with Amazon OpenSearch Service (OpenSearch Service). If queries are not anticipated to be frequent, to optimize on cost, use Amazon Athena to search as opposed to running Amazon OpenSearch Service.

Auditing network and administrative activity

If not already set up, it is a best practice to configure AWS CloudTrail for the AWS account with Amazon AppStream 2.0. To audit AppStream 2.0 API calls specifically, use the filter event source with a value of `appstream.amazonaws.com`.

Enable VPC flow logs to audit access into customer-managed resources. VPC flow logs can be published to CloudWatch Logs to perform queries when auditing is required.

Monitoring subnet IP allocation is important as AppStream 2.0 fleets grow. Report on IP assignment by running the describe-subnets CLI to report the available IP addresses in each subnet assigned to fleets. Ensure that your organization has sufficient IP address capacity to meet the demand of all fleets running at maximum capacity.

Cost optimization

Cost optimization focuses on avoiding unneeded costs. Key topics include understanding and controlling where money is being spent, and choosing the most appropriate and correct number of resource types. Analyze spend over time and scaling to meet business needs. The following AppStream 2.0 resources incur pay-as-you-go fees:

- Always-On fleet instances
- On-Demand fleet instances
- On-Demand stopped instance fee
- Image builder instances
- User fees

For current pricing information, refer to the AWS website for Amazon AppStream 2.0 pricing.

Designing cost efficient AppStream 2.0 deployments

First step in planning and design of AppStream 2.0 deployment is using simple pricing tool to estimate the baseline of your AWS fees related to your usage. Provide your total number of users, actual concurrent usage per hour, instance type, and fleet utilization, and the pricing tool estimates your per-user price. It also shows the estimated price savings when you use an On-Demand fleet instead of an Always-On fleet.

Customers like the AppStream 2.0 pricing model of paying only for the instances they provision to meet their users' streaming needs. This model is different from their existing application streaming environments. These are typically based on provisioning for peak capacity, even during nights, weekends, and holidays, when the load is lower. The Amazon AppStream 2.0 Pricing Tool provides only an estimate of your AWS fees related to your usage of AppStream 2.0, and doesn't include any taxes that might apply. Your actual fees depend on a variety of factors, including your actual usage of AWS services.

The AppStream 2.0 Pricing Tool is provided as a Microsoft Excel or OpenOffice Calc spreadsheet that enables you to enter basic information about your fleet, then provides a cost estimate for the AppStream 2.0 environment for on-demand and always-on fleets based on your usage pattern. You could simulate costs based on historical or anticipated usage trends. Elastic fleets free the administrator of the need to predict usage, create, maintain scaling policies and images by having

these features built-in. Elastic fleets and instances running of Amazon Linux 2 (all fleet types) are billed for duration of the streaming session, in seconds, with a minimum of 15 minutes.

Optimizing costs with choice of instance type

For fleet and image builder instances, there are a range of different instance families and types available that you choose for your application.

End user testing — The next step is to rollout the AppStream 2.0 fleet to a group of pilot users for testing to validate our choice of instance type. It is important to request pilot users to test all their regular and heavy workflows to capture metrics around memory, CPU and graphics so that you can capture baseline performance metrics. The pilot group should contain the various user roles that use the application to ensure you are testing it from multiple user experiences. The user acceptance testing enables you to gather feedback on streaming session experience. When creating or updating a stack, there is an option to use a custom feedback URL. Users are redirected to this URL after they choose the Send Feedback link to submit feedback about their application streaming experience. If there is a performance bottleneck, use Windows performance metrics to analyze resource constraints. For example, if the current fleet instance type stream.standard.medium is showing resource constraint, then upgrade the instance type to stream.standard.large. Conversely, if performance metrics show high levels of under-use of resources, consider downgrading the instance type.

Optimizing costs with fleet type choice

When creating a new AppStream 2.0 fleet, developers must choose either an Always- On or On-Demand fleet type. While choosing the instance type from the pricing perspective, it is important to understand how AppStream 2.0 manages fleet instances. For Always-On fleets, fleet instances stay in the running state. Therefore, when users try to stream sessions, fleet instances are always ready to start streaming sessions.

For On-Demand fleets, after fleet instances are launched, they are kept in the stopped state. The stopped instance fee is lower than the running instance fee, which can help with reducing costs. The On-Demand fleet instances must be started from a stopped state. A user must wait approximately two minutes for their streaming session to be available.

Elastic fleets are good candidates for applications that are self-contained and can be installed to virtual hard drives saved in an Amazon Simple Storage Service (Amazon S3) bucket. Elastic fleets may further reduce costs for some use cases due to the per-second billing charged only for the

duration of streaming. The rate is a function of the instance type and size and operating system that you choose when creating the fleet.

If end users need fleet instances during business hours, it is better to keep the same streaming sessions. This is because fleet instances are charged per hour, and every time a new streaming session starts, that incurs another fleet instance fee.

Table 10 — AppStream 2.0 fleet type comparison

Fleet type	Advantages	Considerations
Always-On	Less wait time for streaming sessions	Users pay for the hourly instance fee as there is no option to keep instances in stopped state.
On-Demand	Cost saving as instances stay in stopped state	Longer wait time for streaming sessions
Elastic	Per-second billing maybe useful for use cases that have sporadic usage patterns for applications that can be installed on virtual hard disk	As the size of application virtual hard disk becomes bigger, the time taken to mount it to a streaming instance can be long

AppStream 2.0 monitors your fleet utilization and performs automatic adjustments to fleet capacity to meet your user demand at the lowest possible cost. The capacity adjustments are made based on scaling policies that you define, based either on the current utilization or based on a schedule. Regularly review fleet usage metrics to validate that the fleet scaling policies do not have high levels of spare capacity.

Scaling policies

Fleet Auto Scaling allows you to optimize fleet resources by not having to over-commit resources waiting for users to login. Administrators can adjust the size of the fleet based on a variety of utilization to match the user demand. Use CloudWatch AppStream 2.0 Fleet Metrics or third party monitoring tools to learn about user activity and configure scaling policies to expand or shrink AppStream 2.0 fleets based on expected usage. User logs are an essential mechanism to gain

understanding of real usage. This insight can be used to dynamically change fleet size based with Auto Scaling.

In many cases, AppStream 2.0 fleets are created based on maximum number of users and not adjusted for different times of the day and week such as nights and weekends. Often times, the concurrent user count of streamed applications is less than the total number of users especially when users have the flexibility to work remotely. It is important to take these factors into consideration while projecting usage patterns. Overestimating leads to over-provisioning of AppStream 2.0 instances resulting in additional costs. To arrive an optimal configuration, you may need to combine one or more scheduled scaling policies with scale out policies.

To learn more about implementing Scaling Policies, review Scaling your Amazon AppStream 2.0 fleets.

User fees

User fees are charged per user, per month in each AWS Region where users stream applications from AppStream 2.0 fleet instances. Instead of generating different user IDs, have consistent user IDs for AppStream 2.0 users. User fees are not charged when connecting to image builders.

Schools, universities, and certain public institutions may qualify for a reduced Microsoft RDS SAL user fee of $0.44 per user per month. For qualification requirements, refer to Microsoft Licensing Terms and Documents.

If you have Microsoft License Mobility, you may be eligible to bring your own Microsoft RDS Client Access Licenses (CALs) and use them with Amazon AppStream 2.0. If you are covered by your own license, you won't incur monthly user fees. For more information about whether you can use your existing Microsoft RDS CAL licenses with Amazon AppStream 2.0, refer to the AWS License Mobility guidance, or consult with your Microsoft licensing representative.

Image Builder usage

AppStream 2.0 Image Builder instances are charged hourly. The Image Builder instance charge includes compute, storage, and any network traffic used by the streaming protocol. All Image Builder instances that are running are charged the applicable running instance fee. This fee is based on the instance type and size, even when no administrators are connected.

As a best practice to optimize the cost, shut down an Image Builder instance when it is not being used. CloudWatch Events rules can be used to schedule a daily job, such as invoking a Lambda function to stop image builder instances.

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. When using this method to update images, an Image Builder is automatically started, and stopped, as part of the managed service process.

Conclusion

With AppStream 2.0, you can easily add your existing desktop applications to AWS and enable your users to stream them instantly. 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 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 an outstanding experience.

Amazon AppStream 2.0 is designed to be integrated into existing IT systems and processes, and this whitepaper described the best practices for doing this. The result of following the guidelines in this whitepaper is a cost-effective cloud desktop deployment that can securely scale with your business on the AWS global infrastructure.

Contributors

Contributors to this document include:

- Andrew Wood, Sr. Solutions Architect, Amazon Web Services
- Andrew Morgan, EUC Specialist SA, Amazon Web Services
- Arun PC, Sr EUC Specialist SA, Amazon Web Services
- Asriel Agronin, Sr. Solutions Architect, Amazon Web Services
- Dustin Shelton, Sr EUC Specialist SA, Amazon Web Services
- Jeremy Schiefer, Sr Solutions Architect, Amazon Web Services
- Navi Magee, Principal Solutions Architect, Amazon Web Services
- Pete Fergus, Sr Cloud Support Engineer, Amazon Web Services
- Phil Persson, Principal EUC Specialist SA, Amazon Web Services
- Richard Spaven, Sr EUC Specialist SA, Amazon Web Services
- Spencer DeBrosse, Sr. Solutions Architect, Amazon Web Services
- Stephen Stetler, Sr. Solutions Architect, Amazon Web Services
- Taka Matsumoto, Sr Cloud Support Engineer, Amazon Web Services
- Vasant Sirsat, Sr EUC Specialist SA, Amazon Web Services

Further reading

For additional information, see:

- Amazon AppStream 2.0 Administration Guide
- Amazon AppStream API Reference
- Use Amazon FSx for Windows File Server and FSLogix to Optimize Application Settings Persistence on Amazon AppStream 2.0
- Monitoring Amazon AppStream 2.0 with Amazon ElasticSearch and Amazon Firehose
- Analyze your Amazon AppStream 2.0 Usage Reports Using Amazon Athena and Amazon QuickSight
- Scale your Amazon AppStream 2.0 fleets
- Using Microsoft AppLocker to manage application experience on Amazon AppStream 2.0
- Using custom domain with Amazon AppStream 2.0
- How do I use my own Microsoft RDS CALs with AppStream 2.0?
- Amazon AppStream 2.0 Pricing Tool
- Create an Online Software Trial with AppStream 2.0
- Create a SaaS Portal with Amazon AppStream 2.0

Document revisions

To be notified about updates to this whitepaper, subscribe to the RSS feed.

Change	Description	Date
Document updated	Updates to include Elastic fleets, at tribute-based application entitlements, multi-stack application catalog, Linux-based fleets, data ingress and egress, disaster recovery, and other updates.	June 14, 2022
Document updated	HTML version published.	January 19, 2022
Initial publication	Whitepaper published.	June 8, 2021

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.