AWS Mainframe Modernization is currently in preview. It is subject to change. We do not recommend using it for production workloads.

Setting up Amazon AppStream for use with AWS Mainframe Modernization Enterprise Analyzer and Enterprise Developer

This document is intended for members of the customer operations team. It describes how to set up Amazon AppStream fleets and stacks to host the Micro Focus Enterprise Analyzer and Micro Focus Enterprise Developer tools used with AWS Mainframe Modernization.

These stacks allow the customer application teams to use Enterprise Analyzer and Enterprise Developer directly from their web browsers. The application files they work on reside either in Amazon S3 buckets or CodeCommit repositories: for more information, see the Setting up Enterprise Analyzer (https://d1vi4vxke6c2hu.cloudfront.net / tutorial/set-up-ea.pdf) and Setting up Enterprise Developer https://d1vi4vxke6c2hu.cloudfront.net/tutorial/set-up-ed.pdf) tutorials.

Important

Before trying to start an Amazon AppStream fleet, make sure that the Amazon AppStream images for Enterprise Analyzer and Enterprise Developer, named m2-enterprise-analyzer-v7.0.1.R1 and m2-enterprise-developer-v7.0.3.R1 in their initial version, are shared by AWS Mainframe Modernization with your account.

This sharing is triggered by clicking on the various links pointing to Amazon AppStream from the AWS Mainframe Modernization console. You can validate this sharing by scanning the list of images available in your account in the Amazon AppStream console. Choose **Images**, then **Image Registry**. Those images are usually toward the end of the long list of available images.

Topics

Step 1: Manage users and data

Currently in Amazon AppStream, you use the User Pool feature to manage users (create them, assign them to stacks, etc.). You manage users from the customer account. For more information, see AppStream 2.0 User Pools (https://docs.aws.amazon.com/appstream2/latest/developerguide/user-pool.html) in the *Amazon AppStream 2.0 Administration Guide*. In particular, for details on executing any of the possible management actions, see User Pool Administration (https://docs.aws.amazon.com/appstream2/latest/developerguide/user-pool-admin.html) in the *Amazon AppStream 2.0 Administration Guide*.

You define the user accounts with their email addresses so that Amazon AppStream can notify users when events, such as assignment, happen on the stacks. Amazon AppStream sends a welcome email when users are

added so they can set up their final password and first access. It is better to define users when at least one stack is running, so that at least one entry is displayed in the Amazon AppStream menu.

Amazon AppStream supports persistent application settings for Windows-based stacks. This means that users' application customizations and Windows settings are automatically saved after each streaming session and applied during the next session. Examples of persistent application settings that users can configure include, but are not limited to, browser favorites, settings, IDE and application connection profiles, plugins, and UI customizations. These settings are saved to an Amazon S3 bucket in the customer account, within the AWS Region where application settings persistence is enabled. The settings are available in each Amazon AppStream streaming session.

Application Settings persistence saves all files in the `C:\Users\PhotonUser` (aka `D:\PhotonUser`) folder, except Contacts, Desktop, Documents, Downloads, Links, Pictures, Saved Games, Searches, and Videos. Those folders are Windows 10 symbolic links (https://blogs.windows.com/windowsdeveloper/2016/12/02/symlinks-windows-10/) to a subfolder in `C:\ProgramData\UserDataFolders\`, which is only visible from the specific instance. We recommend that users avoid storing any data in those unsaved folders.

All details about Application Settings persistence are available in How Application Settings Persistence Works (https://docs.aws.amazon.com/appstream2/latest/developerguide/how-it-works-app-settings-persistence.html) in the *Amazon AppStream 2.0 Administration Guide*.

Persistence of home folders is activated for the Enterprise Analyzer and Enterprise Developer stacks. Users of these stacks can access a persistent storage folder during their application streaming sessions. No further configuration is required. Data stored in their home folders is automatically backed up to an Amazon S3 bucket in the AWS account hosting the fleets and stacks and is made available to those users in subsequent sessions. On the Windows instances, as described here, the home folder of each user is accessible at `C:\Users\PhotonUser\My Files\Home Folder` or at `D:\PhotonUser\My Files\Home Folder`.

Consequently, for each Amazon AppStream user, permanent data is stored in two distinct Amazon S3 buckets whose names are fixed as defined by Amazon AppStream:

- *appstream2-36fb080bb8-region-account*, which stores the Windows home folders of the fleet users under path `user/ > userpool/`.

  Note

  By design, the home folder is unique across all Amazon AppStream fleets of a given account. For Enterprise Analyzer and Enterprise Developer, data stored in the home folder by a given user in the Enterprise Analyzer fleet instance is also visible in the home folder when the same user is connected to an Enterprise Developer instance. The Amazon S3 bucket sub-folder is named using an SHA-256 hash of the user's name. Account administrators must compute this hash for a given user name if they need to access the home folder of a particular user (see below).

- `appstream-app-settings-region-account-unique value`, where path `> Windows/ > v6/ > Server-2019/ > stack-name/ > userpool > user-SHA256-hash` stores a Windows virtual disk (Profile.vhdx – Virtual Windows disk – up to 1 GB) and corresponding Windows Registry key (ProfileList.reg). This disk is attached to the instance when the user starts a session and saved back to Amazon S3 when the session ends. The Amazon S3 folder name of this disk depends on the stack name because there is one such pair of files per stack assigned to a given user. The backup and restore process for application settings is described in detail in How Application Settings Persistence Works (https://docs.aws.amazon.com/appstream2/latest/developerguide/how-it-works-app-settings-persistence.html) in the *Amazon AppStream 2.0 Administration Guide*.

To locate the home folder for a specific user, use a service computing online SHA256 hash. For example, by going to Online Tools - SHA256 (https://emn178.github.io/online-tools/sha256.html) and entering `foo@bar.com` as the user name, the service will return `0c7e6a405862e402eb76a70f8a26fc732d07c32931e9fae9ab1582911d2e8a3b`, which is the name of the Amazon S3 subfolder for this user.

Step 2: Create fleets and stacks

You must create one Amazon AppStream fleet for each of the Enterprise Analyzer and Enterprise Developer images. Then create one Amazon AppStream stack per fleet, and assign the stack to the corresponding fleet. The required Windows images are shared in the customer account (for running only – NOT for building on top) from the AWS Mainframe Modernization service account during the customer onboarding process. The initial versions of these images are `m2-enterprise-analyzer-v7.0.1.R1` and `m2-enterprise-developer-v7.0.3.R1`. The version suffix will change as new versions of those images are produced. When shared, they appear in the AppStream 2.0 console page under **Images > Image Registry**, usually at the end of the list, where service-provided images come first.

The pricing of Amazon AppStream is defined on the AppStream 2.0 pricing (http://aws.amazon.com/appstream2/pricing/) page. The additional fee for use of Enterprise Analyzer and Enterprise Developer is defined on the pricing page for AWS Mainframe Modernization. For more information, see AWS Mainframe Modernization Pricing (http://aws.amazon.com/mainframe-modernization/pricing/edrf) .

You can use an AWS CloudFormation template to automate the creation of the fleets and stacks. During onboarding to AWS Mainframe Modernization, we provide a template named `cfn-m2-appstream-fleet-ea-ed.yaml` to the customer. The current version of the template uses the **Default Internet Access** option in Amazon AppStream. Upcoming versions of this AWS CloudFormation template will allow for more advanced versions (private subnets, with or without NAT and Internet Gateway, etc.) For various possible options, see Internet Access (https://docs.aws.amazon.com/appstream2/latest/developerguide/internet-access.html) .

To use this template, you must adapt at a minimum the following four parameters to the context of the customer AWS account:

- AppStreamApplication: `ea` or `ed`.
- AppStreamImageName: `m2-enterprise-analyzer-v7.0.1.R1` or `m2-enterprise-developer-v7.0.3.R1`, depending on the application.
- AppStreamFleetVpcSubnet: make sure you specify a subnet of the default VPC.
- AppStreamFleetSecurityGroup: make sure you specify the default security group of the default VPC.

To use AWS CloudFormation Stacks, see Working with stacks (https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/stacks.html) in the *AWS CloudFormation User Guide*.

⚠ Important

It is essential to get the fleets running properly so that they obtain access to the product license in real-time. This access is possible only if you assign them an execution role using the definitions made in the AWS CloudFormation template for the `s3:PutObject` and `sts:AssumeRole` credentials. If you don't use the provided template, or if you modify it, be sure to keep the role security definitions unchanged. Similarly, if you define the fleets and stacks in another way, such as the console or CLI, make sure the role policy definition is identical to the following JSON.

If you define your own fleet and stack in a AWS CloudFormation template, implement the same `AppStreamServiceRole` with the same `AssumeRolePolicyDocument` and `PolicyDocument` as in the sample AWS CloudFormation template, as follows:

```
  AppStreamServiceRole:
    Type: AWS::IAM::Role
    DeletionPolicy: Delete
    Properties:
      RoleName: !Join
                  - ''
                  -  - 'm2-appstream-fleet-service-role-'
                     - !Ref AppStreamApplication
                     - !Select [0, !Split ['-', !Select [2, !Split [/, !Ref
AWS::StackId ]]]]
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - 'appstream.amazonaws.com'
            Action: 'sts:AssumeRole'
      Path: /
      Policies:
        - PolicyName: !Join
                        - ''
                        -  - 'm2-appstream-fleet-service-policy-'
                           - !Ref AppStreamApplication
                           - !Select [0, !Split ['-', !Select [2, !Split [/, !Ref
AWS::StackId ]]]]
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              - Effect: Allow
                Action:
                  - 'sts:AssumeRole'
```

```
      Resource:
        - '*'
      - Effect: Allow
        Action:
          - 's3:PutObject'
        Resource:
          - !Sub 'arn:aws:s3:::aws-m2-repo-*-${AWS::Region}-prod/*'
```

We recommend that you use AWS CloudFormation. If you make the IAM definitions interactively using the AWS Console, make sure to define your IAM elements as follows, and replace `us-west-2` with the AWS region in which your fleet is running:

Trust Policy:

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

Authorization Policy:

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

```
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::aws-m2-repo-*-us-west-2-prod/*"
            ],
            "Effect": "Allow"
        }
    ]
}
```

The additional important parameters are as follows, in case account administrators need to change them when creating the fleets and stacks manually or automatically.

Fleet creation

Create fleets based on Create an AppStream 2.0 Fleet and Stack (https://docs.aws.amazon.com/appstream2/latest/developerguide/set-up-stacks-fleets.html) . The key parameters are as follows:

- Fleet type: **Always-on** provides users 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** optimizes 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. For pricing information, see AppStream 2.0 pricing (http://aws.amazon.com/appstream2/pricing/) .
- Instance type: choose **General Purpose stream.standard.large**. For pricing information, see AppStream 2.0 pricing (http://aws.amazon.com/appstream2/pricing/) .
- Capacity: adjust to accommodate the expected number of concurrent users.
- Stream View: choose **Desktop** to allow users to launch other Windows applications (such as Command Prompt, File Explorer, browser, etc.) if needed.
- Image: choose **image-m2-ea** or **image-m2-ed**.
- **Default internet access**: selected.
- **VPC, subnets & security group**: choose the default VPC, and its corresponding subnets and default security group.

Stack Creation

Create stacks based on Create an AppStream 2.0 Fleet and Stack (https://docs.aws.amazon.com/appstream2/latest/developerguide/set-up-stacks-fleets.html) . The important parameters are as follows:

- **Fleet**: select the fleet that you just created.
- **Enable Home folders**: selected.

- **Google Drive for G Suite** and **OneDrive**: to avoid data leaks to an uncontrolled environment, do not choose.

All user experience parameters following these can retain their default values.

In particular, make sure that the **Settings Group** remains distinct for each of the Enterprise Analyzer and Enterprise Developer stacks. The settings group determines which saved application settings are used for a streaming session from this stack. If the same settings group is applied to another stack, both stacks use the same application settings. By default, the settings group value is the name of the stack. We recommend that you retain this set up to avoid confusion between the Enterprise Analyzer and Enterprise Developer stacks.

When the fleet is in the `Running` state and the attached stack is in the `Active` state, you can grant users access to the Windows instances by choosing **AppStream 2.0 > User Pool**, selecting a user for the **Assign Stack** action, and choosing the desired stack in the list. The associated checkmark will trigger sending an email to the user to announce the newly granted access and provide the corresponding streaming URL.

Step 3: Clean up resources

The procedure to clean up the created stack and fleets is described in Create an AppStream 2.0 Fleet and Stack (https://docs.aws.amazon.com/appstream2/latest/developerguide/set-up-stacks-fleets.html) .

When the Amazon AppStream objects have been deleted, the account administrator can also, if appropriate, clean up the Amazon S3 buckets for Application Settings and Home Folders.

Note

The home folder for a given user is unique across all fleets, so you might need to retain it if other Amazon AppStream stacks are active in the same account.

Finally, Amazon AppStream does not currently allow you to delete users using the console. Instead, you must use the service API with the CLI. For more information, see User Pool Administration (https://docs.aws.amazon.com/appstream2/latest/developerguide/user-pool-admin.html) in the *Amazon AppStream 2.0 Administration Guide*.