

Service Workbench Post Deployment Guide

Table of Contents

Account Structure	3
Enable Local Users	3
Create or add compute hosting accounts	3
Create AWS Account	5
Add AWS Account.....	6
Prepare your Account for AppStream	9
Setting up an AppStream Image.....	10
Create Indexes and Projects	12
Create an Administrator User.....	13
Import Service Catalog Products	15
Import a Product	15
Configuration for EC2 Linux.....	18
Configuration for Amazon EC2 Windows	20
Configuration for Amazon SageMaker	23
Configuration for Amazon EMR.....	25
Viewing logs.....	28
Viewing Service Workbench logs in CloudWatch.....	28
Metrics.....	29
Deploying Updates	29
Reference	29
Add an AWS IAM Role to an Amazon EC2 Instance	30
Adding an Administrator Role to a New Amazon EC2 Instance.....	30
Adding a role to an existing instance	31
Usage of AWS Cloud Services	32
Amazon EC2.....	32
AWS IAM.....	32
AWS Organizations	34

Amazon S3	34
AWS Cost Explorer	35
Prepare the Master Account	36
Create a Configuration File	36
Deploy the Prepare Master Account SDC	36
Master Role	37

Account Structure

Service Workbench uses *three* types of accounts. You will see these account names throughout the documentation.

- **Main:** The account from which Service Workbench is deployed. Will be billed for all AWS usage charges in this deployment.
- **Master:** Holds the AWS Organization which creates Member accounts.
- **Hosting:** User accounts created within Service Workbench for individuals.

Read the following files in the source code documentation to learn more about the different types of AWS accounts within Service Workbench:

- README.md
- main/solution/prepare-master-acc/README.md

Enable Local Users

Local users are created only within the solution. Their credentials are stored in [Amazon DynamoDB](#). This is the easiest way to install. The alternative is to integrate with an Active Directory.

Create or add compute hosting accounts

After logging in as the **root** user for the first time, go to the **Accounts** page.

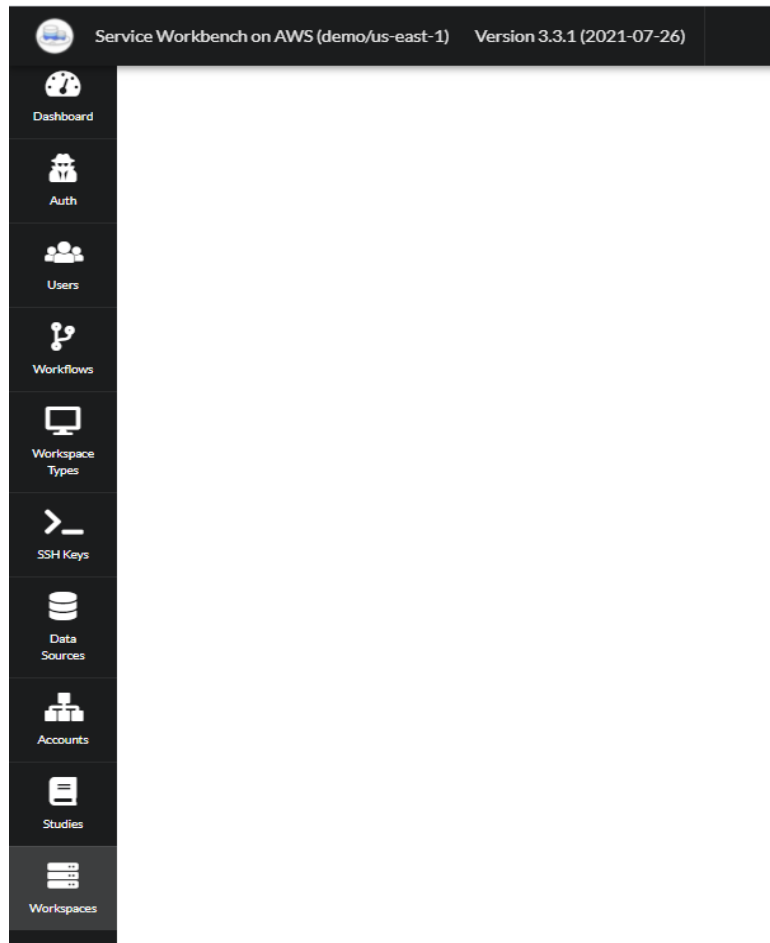


Figure: Service Workbench navigation bar

Service Workbench uses AWS accounts on this page for launching research workspaces. You can add existing AWS accounts or create new ones on the **Accounts** tab.

- **Create AWS Account:** Creates a new AWS account using AWS Organizations. Note that preparing organizational account steps outlined above are a pre-requisite should you require this capability.
- **Add AWS Account:** Associates an existing AWS account for purposes of hosting compute resources. This account can be responsible for its own billing.

Every user is linked to an account through a project and an index, so at least one account must be created or added before associating the first user to a project.

Note: If you do not need to create new AWS accounts from within Service Workbench, then skip to [Add AWS Account](#) section.

Create AWS Account

Prerequisites

Before creating an AWS account from Service Workbench, follow these guidelines:

- Configure an existing AWS account to be the organizational account for Service Workbench (steps outlined in *Preparing the organizational account* section). When Service Workbench creates new AWS accounts, billing for those accounts is applicable to the organizational account.

Creating a new Account

This creates a new **hosting** AWS account in the organization, whose billing goes to the **organizational** account.

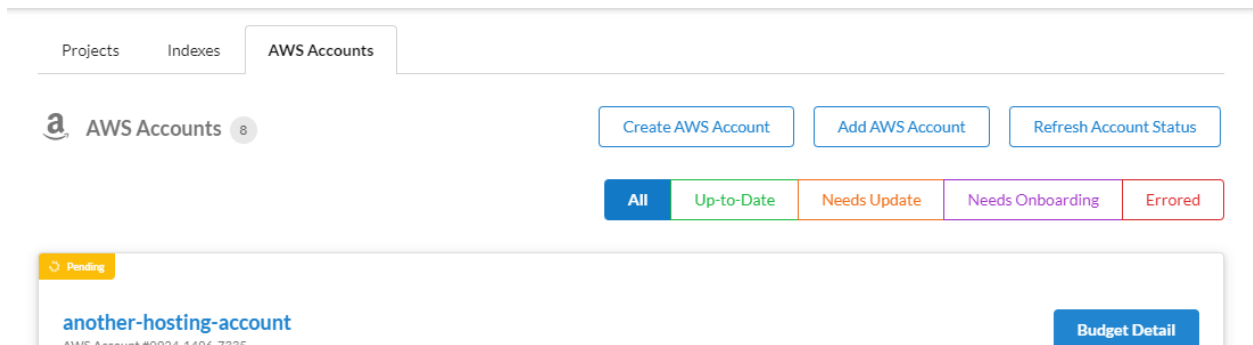


Figure: Create a new hosting account

To create an account:

1. In the Service Workbench console, choose **AWS Accounts** and then choose **Create AWS Account**.
 - In **Role ARN**, enter the **Master Role ARN** copied from the preparing the organizational account steps.
 - The email address that you specify here must be unique within the organization.
 - The **External ID** is **workbench** by default. See [IAM](#) for information on how to configure this to another value.
2. During processing, the following information displays in the **AWS Accounts** tab:
 - 'Trying to create accountID: xxx'
 - A workflow in progress in **Workflows** → **Provision Account** (see [Workflows](#))

Note: If instead you see an error message such as, 'Stop Internal State Account ID not found', check that there is an AWS Organization in the console of

your **organizational** account, if deploying Service Workbench in the **organizational** account. If you are deploying in a **Member** account, check and ensure that you followed the steps described in *Prepare Organizational Account*.

- Optionally, in the AWS console, you can inspect the following resources deployed by this script:
 - In AWS CloudFormation, a stack **prep-master** is running. It creates the **Master** role and its output is the **Master Role ARN**.
 - In the AWS Organization, in the **Organizational** account (see [IAM](#)), the new account displays.
 - In IAM, the new **Master** role is created.
- 3. Once the account is created it is listed in **AWS Accounts**.

Add AWS Account

Adding an existing AWS account enables Service Workbench to launch research workspaces. The existing account alignment (standalone or associated to an organization) determines the billing responsibility.

- On the **Accounts Page**, choose **AWS Accounts**, and then choose **Add AWS Account**.

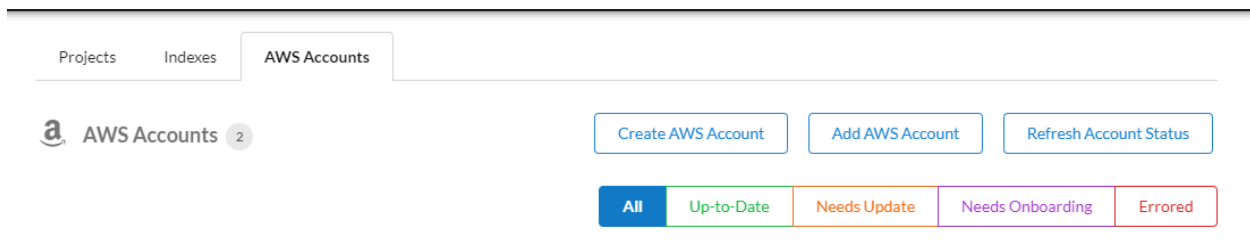


Figure: Add an existing account

- Create and run the AWS CloudFormation template:

If AppStream is not enabled

- Enter the **Account Name**, **Account ID** (12 digit AWS account ID for the account you wish to add), and **Description**.

Service Workbench on AWS (demo/us-east-1) Version 3.3.1 (2021-07-26)

Dashboard
Auth
Users
Workflows
Workspace Types
SSH Keys
Data Sources

Add AWS Account

Account Name
Type the name of this account

AWS Account ID
Type the 12-digit AWS account ID

Description
Type description for this AWS account

Figure: Specify account details

- The **Onboard AWS Account** screen is displayed, where you can select either of the following options:
 - **I have admin access** : You have administrator-level access to the hosting account that is being onboarded. Selecting this and then proceeding launches the CloudFormation template within the hosting account. Note that you need to be logged into the AWS account console for the hosting account when selecting this option and proceeding.
 - **I do not have admin access** : The CloudFormation template is generated and you can then share the template to be run by the party that does have administrator access in the AWS account that you are onboarding.

Onboard AWS Account

CloudFormation Stack Name
initial-stack-1632323433789

AWSTemplateFormatVersion: 2010-09-09
Description: This stack provisions resources necessary to use this AWS account with Service Workbench.
Parameters:
EnableAppStream:
Type: String
AllowedValues: [true, false]
Description: Onboard this account to support AppStream
Namespace:

AWS Account # 888888888888

I have admin access I do not have admin access

Figure: Onboard AWS account

If AppStream is enabled

If you have chosen to enable AppStream for your installation, there are additional values required when onboarding a hosting account.

Add AWS Account

Account Name

AWS Account ID

Description

AppStream Fleet Desired Instance

AppStreamDisconnectTimeoutSeconds

AppStreamIdleDisconnectTimeoutSeconds

Figure: Add account when AppStream is enabled

- **AppStream Fleet Desired Instance:** The maximum number of concurrently running AppStream sessions allowed. If you set this to 5, that would mean that five workspaces can be viewed concurrently.
- **AppStreamDisconnectTimeoutSeconds:** With a minimum of 60 seconds, this is the amount between a researcher disconnection from a session (Manual Stop, Auto Stop, or Terminate) and the release of the AppStream instance that is supporting that session.
- **AppStreamIdleDisconnectTimeoutSeconds:** The amount of time that an AppStream session idle time (meaning no activity within the session) before the AppStream instance disconnects.

AppStreamIdleDisconnectTimeoutSeconds

The amount of time that users can be idle (inactive) before they are disconnected from their streaming session

AppStreamMaxUserDurationSeconds

The maximum amount of time that a streaming session can remain active, in seconds

AppStreamImageName

The name of the image used to create the fleet

AppStreamInstanceType

The instance type to use when launching fleet instances. List of images available at <https://aws.amazon.com/appstream2/pricing/>

AppStreamFleetType

The fleet type. Should be either ALWAYS_ON or ON_DEMAND

Cancel

Onboard AWS Account

Figure: Add account when AppStream is enabled (contd..)

- **AppStreamMaxUserDurationSeconds:** The maximum amount of time for an AppStream session.
- **AppStreamImageName:** The exact image name produced when you follow the instructions to [build your AppStream image](#)
- **AppStreamInstanceType:** The instance type for your AppStream fleet. Note that these instance types are unique to AppStream. A complete list and specifications for valid instance types is available at <https://aws.amazon.com/appstream2/pricing/>
- **AppStreamFleetType:** ALWAYS_ON ensures that the desired number of instances remain available at all times. ON_DEMAND only runs the instances when required. It is a cost versus convenience choice.

Note: If you needed to change these values later, you can do so through the AWS Console of the hosting account without negative impact to Service Workbench.

Once these options are specified, choosing **Onboard Account** displays the **Onboard AWS Account** screen. The same choice of Admin vs. No Admin access applies, but there are several important pre-requisites to complete before proceeding.

Prepare your Account for AppStream

AWS AppStream service limits may block resource creation in a new AWS account, or an account that has not yet hosted AppStream resources. The following actions will best prepare the hosting account.

1. **Required:** Go to AppStream 2.0 services (AWS Console), and choose **Get Started**. This will take you to a screen asking if you want to try out some templates. At this screen choose **Next**. This will only need to be done once for the account (it activates a role required for AppStream to be initiated).
2. **Recommended:** Launch and terminate at least one EC2 instance (size of instance and duration of run prior to termination do not matter), as this establishes the base compute service limits.
3. **Optional:** Open a support ticket within the hosting account to AWS AppStream service, noting your intention to create a fleet. This should only be necessary if you are creating a large fleet, but if the account is very new or has never had resources created prior to being onboarded as a hosting account, this is recommended.

Once prerequisites are complete and the CloudFormation stack has been created, go to AppStream on the AWS console of the hosting account. Go to **Fleet** and then choose the newly created fleet. Choose **Action>Start** to start the fleet. This step must be completed for any researcher workspace to be successfully launched within the hosting account.

After proceeding from the **Onboard AWS Account** Screen, you will be returned to the **Account listing** page with a status.

- **Up to Date** : The account has successfully onboarded and the version of the main and hosting account code is in sync.
- **Needs Update** : This status means that the main account code is more up to date than the hosting account code. There is an update button available and it is advised that you use this to bring the hosting account to the latest version.
- **Needs Onboarding** : The account has been onboarded via the Service Workbench UI but the CloudFormation template has not been generated
- **Errored** : The onboarding of the hosting account failed. Check the CloudFormation stack in the hosting account.
- **Pending** : The account onboarding process is ongoing. If a hosting account remains in pending status beyond 30 minute, it is likely that there is an issue onboarding the account. Check the CloudFormation stack in the hosting account.

Setting up an AppStream Image

Overview

This section describes the procedure of setting up an AppStream image with the following applications installed: Firefox, Notepad, PuttyGen, and Putty. AppStream should be built in the **member account**. To do this:

- Launch an AppStream Image Builder instance.
- Log in to the AppStream Image builder instance and run a script to build an image with Firefox, PuttyGen, Putty, and Notepad.

Pre-requisites

1. Navigate to AWS AppStream in your main account using the AWS Management Console.
2. Choose **Get Started** and then choose **Next**. This activates AppStream in your main account, following which you could proceed with the commands in `SETUP.md` file.

Launching an AppStream Image Builder instance

1. Navigate to the `scripts/app-stream` directory.
2. Enter the following commands:

```
npm install
npm run start-image-builder -- <AWS Profile> <region> <Base image name>
<instance size>
# Example: npm run start-image-builder -- default us-east-1 AppStream-
WinServer2019-06-01-2021 stream.standard.medium
# If preferred you can choose the default base image name and instance
size by running this command: npm run start-image-builder -- default
us-east-1 default default
```

Note: Set up your AWS profile beforehand so that you have permission to launch an AppStream Image Builder instance in your AWS Account.

3. Once the Image Builder is launched and ready, choose the URL provided in the terminal console. This opens the AppStream page on the AWS Console.

Building AppStream image

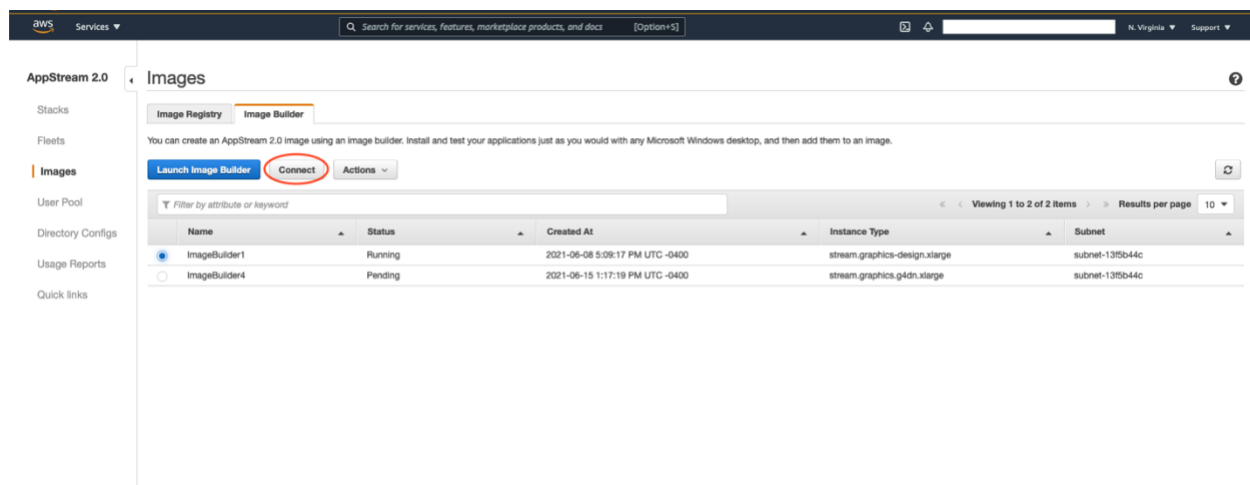


Figure: Create an AppStream image

1. On the AppStream page in the console, choose the AppStream image that was built in the previous step and choose **Connect**.

2. This opens a new tab in your browser. When prompted, log in as **Administrator**. This opens the Windows Desktop that you can interact with to create your AppStream image.
3. On the Windows Desktop, choose the **Start** button and enter `Windows Powershell`.
4. Right-click the application and choose `Run as administrator`.
5. Enter the following commands:

```
cd ~\Documents

# Pull the Image Builder script from Github
Invoke-WebRequest -Uri
https://raw.githubusercontent.com/aws-labs/service-workbench-on-aws/feat-secure-workspace-egress/scripts/app-stream/buildImage.ps1 -
OutFile buildImage.ps1

# Execute Image builder script
.\buildImage.ps1
```

6. At this point, the Image builder builds your image and the `Failed to reserve a session` message is displayed.

Log in to AppStream on the console again and wait till the AppStream image is built.

Create Indexes and Projects

Projects and **Indexes** form a hierarchy under '**Accounts**'. Each Account can have multiple **Indexes**, each **Index** can have multiple **Projects**. **Projects** are attached to **Users**, so you must create the **Projects** first.


After you create an [account](#) in the '**Accounts**' tab of the administrative interface, create an '**Index**' that links to the Account, by selecting the '**Account ID**' from the drop-down list.

1. On the '**Indexes**' tab, click '**Add Index**'. See the figure below.

Projects

Indexes

AWS Accounts

 Indexes 1

Add Index

Index Name	AWS Account	Description
<input type="text"/>	<input type="text"/>	<input type="text"/>
index1	crn:aws:iam::252337190622	Index 1


Figure : Create an Index

2. Create a **'Project'** that links to the new Index. In the **'Projects'** tab, click **'Add Project'**. See **Figure 6**.

Projects

Indexes

AWS Accounts

 Projects 1

Add Project

Project Name	Index Id	Description
<input type="text"/>	<input type="text"/>	<input type="text"/>
project1	index1	Project 1

Figure: Create a Project

Create an Administrator User

Once you create an [account](#) and an [index and project](#), you must create an administrator user in the **'Users'** tab. See **Figure 7**.

Dashboard


Auth

Users

API Keys

Users

Roles

 Users 6

Add Local User

Add Federated User








Name	Email	Identity Provid...	Type	Role	Project	Status	
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	
root	 @amazon.com	internal	Internal	admin	<<none>>	 Active	<div>Detail</div>
 @amazon.com	 @amazon.com	internal	Internal	admin	project1	 Active	<div>Detail</div>

Figure: Create an Administrator

Note: A root user account will already be created, however, you must not routinely use the root user account.

For testing purposes, you can create a local user by clicking **'Add Local User'**. Assign the user the administrator's role, and associate the user with the **Project** you created, and set the status to **'Active'**. See **Figure 8**.

 Add Local User

**Not for production usage**
Creating local users is not meant to be used in production environments.

Username

Username in email format

First Name

Last Name

Password

UserRole

Select user's role

Projects

Select projects that this user are associated with

✕

Status

Active

 or

Inactive

Active users can log into the Research Portal

Cancel

Add Local User

Figure: Add Local User

In prod environments we highly recommend using an IDP. For more details, refer to the ***Service Workbench Configuration Guide***.

Import Service Catalog Products

Service Workbench uses [AWS Service Catalog](#) to manage different types of computation resources available for researchers to use through the platform.

With AWS Service Catalog integration, Service Workbench allows Admin users to create and manage catalogs of IT services that are approved for use on AWS. These IT services can include everything from virtual machine images, servers, software, and databases to complete multi-tier application architectures.

With this integration, Service Workbench helps organization to centrally manage commonly deployed IT services, and helps achieve consistent governance and meet compliance requirements, while enabling users to quickly deploy only the approved IT services they need.

When Service Workbench is deployed, an AWS Service Catalog portfolio is created by default with four commonly used products: Amazon SageMaker, Amazon EC2 for Windows, Amazon EC2 for Linux and Amazon EMR. The **administrator** needs to import and configure these products using Service Workbench user interface before they can be deployed. If you want to include additional custom products in the AWS Service Catalog portfolio, complete these steps:

1. Add the AWS CloudFormation template in the following directory:

```
addons/addon-base-raas/packages/base-raas-cfn-  
templates/src/templates/service-catalog
```

2. Add the AWS CloudFormation template file name in the productsToCreate list in the following location:

```
addons/addon-base-raas/packages/base-raas-post-  
deployment/lib/steps/create-service-catalog-portfolio.js
```

Import a Product

In this step, you import a pre-defined product, configure parameters to be used for product launch, and approve the configured product to be used. The following sections use Amazon EC2 Linux as an example, followed by setting different configuration required for Amazon EC2 Windows, Amazon SageMaker and Amazon EMR.

Prerequisites

Ensure the following prerequisites are met in order to import a product.

AMI

Make sure you completed the step, deploy the Machine Images SDC as part of the deployment process.

To check if AMIs were created successfully, perform the following actions:

1. Navigate to Amazon EC2.
2. Select the '**AMI**' tab.
3. Note down the 4 AMIs created for (1) Amazon EC2 Linux, (2) Amazon EC2 Windows, (3) Amazon EMR, and (4) Amazon EC2 Rstudio.
4. Copy the AMI IDs and use for workspace import and configuration. Alternatively, you can also copy these AMI IDs from the terminal when the machine-images SDC is deployed.

Note: *If you run the machine images SDC multiple times, duplicated AMIs are created. This is okay and will not affect any Service Workbench functionalities. You can choose to remove the duplicates to avoid confusion or leave them as is.*

Service Catalog Portfolio

1. Log in to Service Workbench UI as an **administrator**.
2. Navigate to '**Workspace Types**' tab. Four AWS Service Catalog Products display as shown in **Figure 9**.



Available AWS Service Catalog Product Versions not imported yet

EC2 Windows-V1.0.0

V1.0.0

- An EC2 Windows instance with RDP access
- Secure compute in the cloud

Import ↓

SageMaker Notebook-V1.0.0

V1.0.0

An Amazon SageMaker Jupyter Notebook that comes with:

- TensorFlow
- Apache MXNet
- Scikit-learn

Import ↓

EC2 Linux-V1.0.0

V1.0.0

- An EC2 Linux instance with SSH access
- Secure compute in the cloud

Import ↓

EMR-V1.0.0

V1.0.0

An Amazon EMR research workspace that comes with:

- Hail 0.2
- Jupyter Lab
- Spark 2.4.4
- Hadoop 2.8.5

Import ↓

Figure: AWS Service Catalog Products

These four products come from the AWS Service Catalog portfolio created by the system during deployment. And they'll be ready for use once imported and configured.

If you wish to include other AWS computation resources in the future:

1. Add a new product to the existing Service Workbench portfolio in AWS Service Catalog
2. Update the role `ServiceCatalogLaunchConstraintRole` in [cloudformation.yml](#) to include permission needed to launch and terminate the product

Import

In this section, the Amazon EC2 Linux is used as an example.

1. Click the '**Import**' button under `ec2-linux-instance`.
2. Update **Name** and **Description** so you can easily identify the workspace.

Configure

Once you import a workspace type, perform the following actions:

1. Click '**Add Configuration**'
2. Add **ID**, **Name**, **Description**, and **Estimated Costs** for the configuration. A common naming convention here is to attach the instance size after the product name. For example, use ec2-linux-instance-V1-small for a small Linux Amazon EC2 instance.
3. Click '**Next**'.
4. Add access control for the workspace configuration.
5. Click '**Next**'

The input parameters are parameters used for the product, AWS CloudFormation template. The number and type of parameters are different for different products. Most of the parameters used for the four system created products can be evaluated automatically at launch time. These parameters are available for selection in the drop-down when filling the input parameters page.

Configuration for EC2 Linux

For Amazon EC2 Linux, the only two fields that are not available in the drop-down are '**InstanceType**' and '**Amild**'.

The following figures display screenshot images that exemplify Amazon EC2 Linux configurations.



Edit Configuration

first

Basic Information

Access Control

Input Parameters

Tags

EncryptionKeyArn

The ARN of the KMS encryption Key used to encrypt data in the instance

`${encryptionKeyArn}`



IamPolicyDocument

The IAM policy to be associated with the launched workstation

`${iamPolicyDocument}`



AccessFromCIDRBlock

The CIDR used to access the ec2 instances.

`${cidr}`



VPC

The VPC in which the EC2 instance will reside

`${vpclid}`



Figure: Configurations for Amazon EC2 Linux

The screenshot displays a series of configuration fields for an Amazon Linux EC2 instance within a CloudFormation template editor. Each field has a title, a description, and a text input box with a placeholder value and a clear button (X).


- EnvironmentInstanceFiles**: An S3 URI (starting with "s3://") that specifies the location of files to be copied to the environment instance, including any bootstrap scripts. Placeholder: `${environmentInstanceFiles}`.
- InstanceType**: EC2 instance type to launch. Placeholder: `t3.large`.
- Subnet**: The VPC subnet in which the EC2 instance will reside. Placeholder: `${subnetId}`.
- S3Mounts**: A JSON array of objects with name, bucket, and prefix properties used to mount data. Placeholder: `${s3Mounts}`.
- Namespace**: An environment name that will be prefixed to resource names. Placeholder: `${namespace}`.
- Amild**: Amazon Machine Image for the EC2 instance. Placeholder: `ami-0d1b1e57e0c0a90a1c`.

Figure: Configurations for Amazon Linux EC2

Configuration for Amazon EC2 Windows

For Amazon EC2 Windows, the only two fields that are not available in the drop-down are **'InstanceType'** and **'Amild'**. (Use the AMI ID you copied in Prerequisites - AMI)

The following figures display screenshot images that exemplify Amazon EC2 Windows configurations.

 Add Configuration

1 Basic Information
Enter basic information

2 Access Control
Define who can access

3 Input Parameters
Provide AWS CloudFormation Inputs

4 Tags
Specify Resource Tags

DownloadInterval

An interval in seconds to wait between two downloads in case of recurring downloads. This is only applicable when RecurringDownloads is set to "true". Note that this does not include the download time. This specifies the duration in seconds to wait before initiating the next download after the previous one completes.

20

RecurringDownloads

A flag indicating whether to keep syncing studies data to local EBS volumes on recurring basis. Setting this to false will download studies data only once at the instance bootstrap time. When this flag is set to true the instance will periodically sync changes from S3 to local EBS i.e., it will download any new files added to S3, re-download any files changed in S3 (will use object ETag value to determine if file changed in S3), delete files from local EBS if they are deleted from S3.

true

EncryptionKeyArn

The ARN of the KMS encryption Key used to encrypt data in the instance

\${encryptionKeyArn}

AccessFromCIDRBlock

The CIDR used to access the ec2 instances.

\${cidr}

VPC

The VPC in which the EC2 instance will reside

\${vpcid}

Figure: Configurations for EC2 Windows

S3Mounts	
A JSON array of objects with name, bucket, and prefix properties used to mount data	
<input type="text" value="{s3Mounts}"/>	✕
Namespace	
An environment name that will be prefixed to resource names	
<input type="text" value="{namespace}"/>	✕
KeyName	
Keypair name for admin password encryption/decryption	
<input type="text" value="{adminKeyPairName}"/>	✕
RaidDataVolumeSize	
The size of each volume in the RAID array used to hold studies data, in GiB. The template creates a striped volume (RAID 0) by joining 8 volumes. The total size of the data volume would be roughly 8 times the size specified here.	
<input type="text" value="10"/>	✕
StopRecurringDownloadsAfter	
Duration in seconds after which to stop the recurring downloads. Value of -1 means keep doing the recurring downloads (sync) indefinitely.	
<input type="text" value="-1"/>	✕
IamPolicyDocument	
The IAM policy to be associated with the launched workstation	
<input type="text" value="{iamPolicyDocument}"/>	✕
EnvironmentInstanceFiles	
An S3 URI (starting with "s3://") that specifies the location of files to be copied to the environment instance, including any bootstrap scripts	
<input type="text" value="{environmentInstanceFiles}"/>	✕

Figure: Configurations for EC2 Windows

The screenshot displays a configuration window for an Amazon SageMaker EC2 instance. It contains three input fields, each with a title, a description, and a value. The 'InstanceType' field is set to 'r5.2xlarge'. The 'Subnet' field contains a placeholder '\$(subnetId)'. The 'Amild' field contains a placeholder 'ami-O#####'. At the bottom, there are three buttons: 'Cancel', 'Previous', and 'Next'.

Field	Description	Value
InstanceType	EC2 instance type to launch	r5.2xlarge
Subnet	The VPC subnet in which the EC2 instance will reside	\$(subnetId)
Amild	Amazon Machine Image for the EC2 instance	ami-O#####

Figure: Configurations for EC2 Windows

Configuration for Amazon SageMaker

For Amazon SageMaker, the only field that's not available in the drop-down is '**InstanceType**'.

The following figures display screenshot images that exemplify Amazon SageMaker configurations.

Add Configuration

1 Basic Information
Enter basic information

2 Access Control
Define who can access

3 **Input Parameters**
Provide AWS CloudFormation Inputs

4 Tags
Specify Resource Tags

EncryptionKeyArn

The ARN of the KMS encryption Key used to encrypt data in the notebook

`${encryptionKeyArn}`

✕

IamPolicyDocument

The IAM policy to be associated with the launched workstation

`${iamPolicyDocument}`

✕

VPC

VPC for EMR nodes.

`${vpcId}`

✕

AccessFromCIDRBlock

The CIDR used to access sagemaker.

`${cidr}`

✕

EnvironmentInstanceFiles

An S3 URI (starting with "s3://") that specifies the location of files to be copied to the environment instance, including any bootstrap scripts

`${environmentInstanceFiles}`

✕

InstanceType

EC2 instance type to launch

`ml.t3.medium`

✕

Subnet

Subnet for EMR nodes, from the VPC selected above

`${subnetId}`

✕

Figure: Configurations for Amazon SageMaker

S3Mounts
A JSON array of objects with name, bucket and prefix properties used to mount data

Namespace
An environment name that will be prefixed to resource names

AutoStopIdleTimeInMinutes
Number of idle minutes for auto stop to shutdown the instance (0 to disable auto-stop)

Figure: Configurations for Amazon SageMaker

Configuration for Amazon EMR

Amazon EMR requires a few more fields that are not available in the drop-down menu, including the following:

- DiskSizeGB (≥ 10)
- CoreNodeCount (1-80)
- MasterInstanceType
- Market (ON_DEMAND / SPOT)
- WorkerBidPrice (only applicable when Market = SPOT. Specify 0 for Market = ON_DEMAND)
- WorkerInstanceType
- Amild (Use the AMI id we copied in prerequisites - AMI)

The following figures display screenshot images that exemplify Amazon EMR configurations.

DiskSizeGB	EBS Volume size (GB) for each node	10	✕
CoreNodeCount	Number of core nodes to provision (1-80)	1	✕
EncryptionKeyArn	The ARN of the KMS encryption Key used to encrypt data in the cluster	`\${encryptionKeyArn}`	✕
VPC	VPC for EMR nodes.	`\${vpcId}`	✕
AccessFromCIDRBlock	Restrict WebUI access to specified address or range	`\${cidr}`	✕
MasterInstanceType	EMR node ec2 instance type.	c5.xlarge	✕

Figure: Configurations for Amazon EMR

Market Which market to purchase workers on - ON_DEMAND or SPOT.	ON_DEMAND	✕
S3Mounts A JSON array of objects with name, bucket and prefix properties used to mount data	`\${s3Mounts}`	✕
Namespace An environment name that will be prefixed to resource names	`\${namespace}`	✕
KeyName SSH key pair to use for EMR node login	`\${adminKeyPairName}`	✕
IamPolicyDocument The IAM policy to be associated with the launched workstation	`\${iamPolicyDocument}`	✕
WorkerBidPrice Bid price for the worker spot nodes. This is only applicable when Market = SPOT. Specify 0 for Market = ON_DEMAND.	0	✕

Figure: Configurations for Amazon EMR

EnvironmentInstanceFiles
An S3 URI (starting with "s3://") that specifies the location of files to be copied to the environment instance, including any bootstrap scripts

Subnet
Subnet for EMR nodes, from the VPC selected above

WorkerInstanceType
EMR node ec2 instance type.

AmiId
Ami Id to use for the cluster

Cancel

Save

Figure: Configurations for Amazon EMR

Approve

Once the configuration completes, click the **'Approve'** button; the newly created workspace type will be available for launch in the **'Study and Workspace'** tab.

Viewing logs

Viewing Service Workbench logs in CloudWatch

Service Workbench has API Gateway access logging enabled. The logs are available in CloudWatch at the `/aws/api-gateway/<name of your API>` log group:

Following is the format of the access logs:

```
{ "authorizer.principalId": "u-000000000000", "error.message": "-", "extendedRequestId":  
"ZuT4rGDNoAMFxw=", "httpMethod": "GET", "identity.sourceIp": "22.22.222.22", "integration.error": "-",  
"integration.integrationStatus": "200", "integration.latency": "79", "integration.requestId": "67394741-90ae-
```

```
4c6c-94fb-df8bf7be33ec", "integration.status": "200", "path": "/dev/api/user-roles", "requestId": "468a1b4d-3015-4901-b749-37e4e0551029", "responseLatency": "83", "responseLength": "819", "stage": "dev", "status": "200"}
```

Lambda logs are also available in CloudWatch with the default log group names `/aws/lambda/<lambda function name>`.

Metrics

The default metrics for Lambda and API Gateway are available in CloudWatch. For the full list of available metrics, see:

- [Working with AWS Lambda function metrics - AWS Lambda](#)
- [Amazon API Gateway dimensions and metrics - Amazon API Gateway](#)

Service Workbench does not emit any custom metrics.

Deploying Updates

After a successful initial deployment, you can deploy individually to the five serverless projects that are a part of this solution.

Deploying Updates to the Infrastructure Serverless Project

```
$ cd solution/infrastructure$ pnpm sls deploy -s <stage>
```

Deploying Updates to the Backend Serverless Project

```
$ cd solution/backend$ pnpm sls deploy -s <stage>
```

Deploying Updates to the Machine-Images Serverless Project

```
$ cd solution/machine-images$ pnpm sls deploy -s <stage>
```

Deploying Updates to the Post-Deployment Serverless Project

```
$ cd solution/post-deployment$ pnpm sls invoke local -f postDeployment --env WEBPACK_ON=true -s <stage>
```

Deploying Updates to the UI Serverless Project

```
$ cd solution/ui$ pnpm sls package-ui --stage <stage> --local$ pnpm sls package-ui --stage <stage>$ pnpm sls deploy-ui --stage <stage> --invalidate-cache
```

Reference

Service Workbench on AWS interacts with multiple AWS resources, including Amazon EC2, AWS IAM, AWS Organizations, and more. You can easily add an AWS IAM role to an Amazon EC2 instance, leverage our AWS Organizations with your account structure, and create multiple Amazon S3 buckets.

Add an AWS IAM Role to an Amazon EC2 Instance

An Amazon EC2 instance can be assigned an **Instance Profile** that contains an AWS **IAM role**. The AWS **IAM role** will give the Amazon EC2 instance a set of permissions. The Amazon EC2 instance will only perform the actions defined by its AWS **IAM role**. Adding an AWS **IAM role** to the Amazon EC2 instance allows your application to make API calls securely—eliminating the need to manage security credentials.

The Service Workbench deployment application must be able to create AWS resources. The easiest way to meet this requirement is to give the Amazon EC2 instance an administrator role.

Adding an Administrator Role to a New Amazon EC2 Instance

When creating a new Amazon EC2 instance for a Service Workbench deployment, an **Instance Profile** may be assigned to the Amazon EC2 instance in **'Step 3: Configure Instance Details'**. Select **'Create a new IAM role'**—located next to the AWS IAM role drop-down. The following figure displays an image of the **'Create a New IAM'** role action in the AWS Management Console.



Figure: Create a New AWS IAM Role

To continue the process, highlight Amazon EC2 and proceed to permissions. In **'Permissions'**, filter for **'AdministratorAccess'** and select it. Proceed through **'Tags'**. On the **'Review'** page, give your role a memorable name. Return to the Amazon EC2 tab, refresh the AWS IAM role drop-down, and select your administrator role to attach to the new Amazon EC2 instance. Now, proceed through the process to create an Amazon EC2 instance. The following figures display images to help you complete this process.

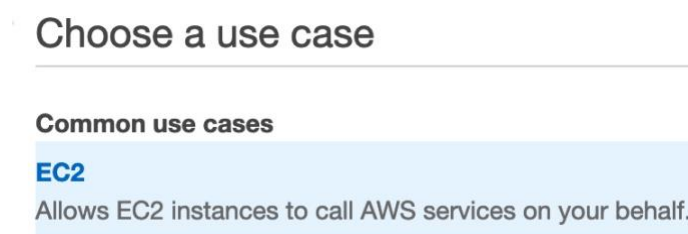


Figure: Permissions in Amazon EC2

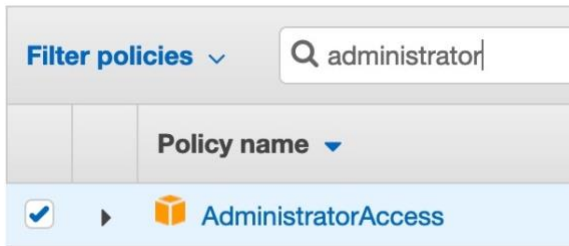


Figure: Filtering for AdministratorAccess



Figure: Choosing a Role Name for an Amazon EC2 Instance

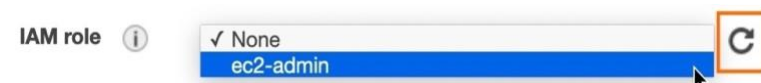


Figure: Selecting the Administrator Role of the Amazon EC2 Instance

Adding a role to an existing instance

To add a role to an Amazon EC2 instance that is already running, select the Amazon EC2 instance in the EC2 Console. Open the **'Action > Instance Settings'** menu, and select **'Attach/Replace IAM Role'**. The following figure shows the **Instance Settings** menu.

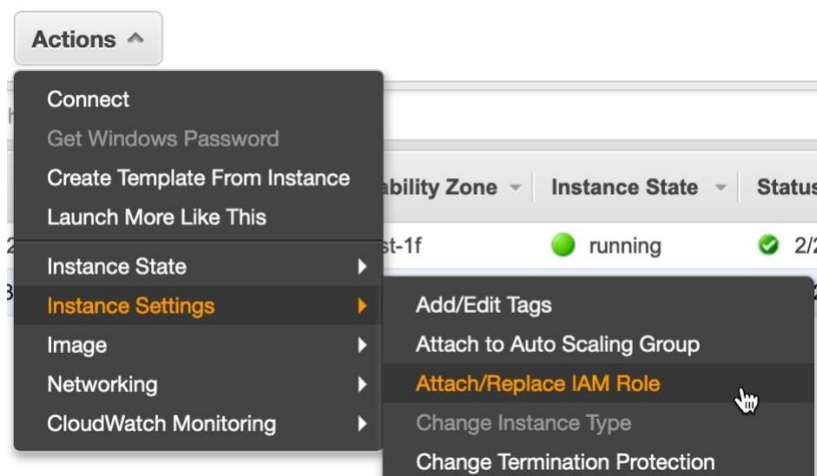


Figure: Attach/Replace an AWS IAM Role in the EC2 Console

In the **'Attach/Replace IAM Role'** screen, search for the role you created, select it, and click **Apply**. The following figure shows the screen where you can **'Attach/Replace IAM Role'**.

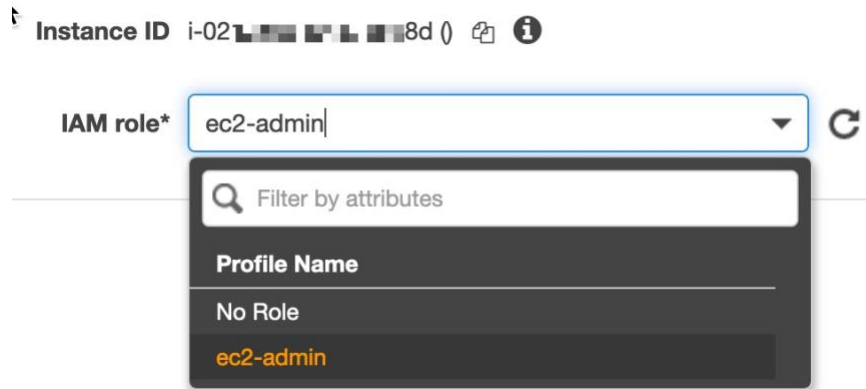


Figure: AWS IAM Role Search

Usage of AWS Cloud Services

This section describes some of the AWS Cloud services used by Service Workbench. The resource names usually include the **Namespace**, including the [Stage Name](#) used at deployment. You can deploy multiple instances of Service Workbench from the same account if you use a different Stage Name for each deployment.

Amazon EC2

Amazon EC2 is used only as a platform from which to deploy Service Workbench. For more details see the [Deployment Instance](#) section.

AWS IAM

Service Workbench creates several roles in your account. The role `<namespace>-prep-raas-master-MasterRole-XXX` is created when you run the [Post Deployment](#) SDC. This role possesses a trust relationship with the Main account from which you deployed Service Workbench. There are two policies that allow the Main account to assume a role in this Master account. The [Account Structure](#) defines each type of account. The following figure shows the AWS IAM 'Trust Relationships' tab.

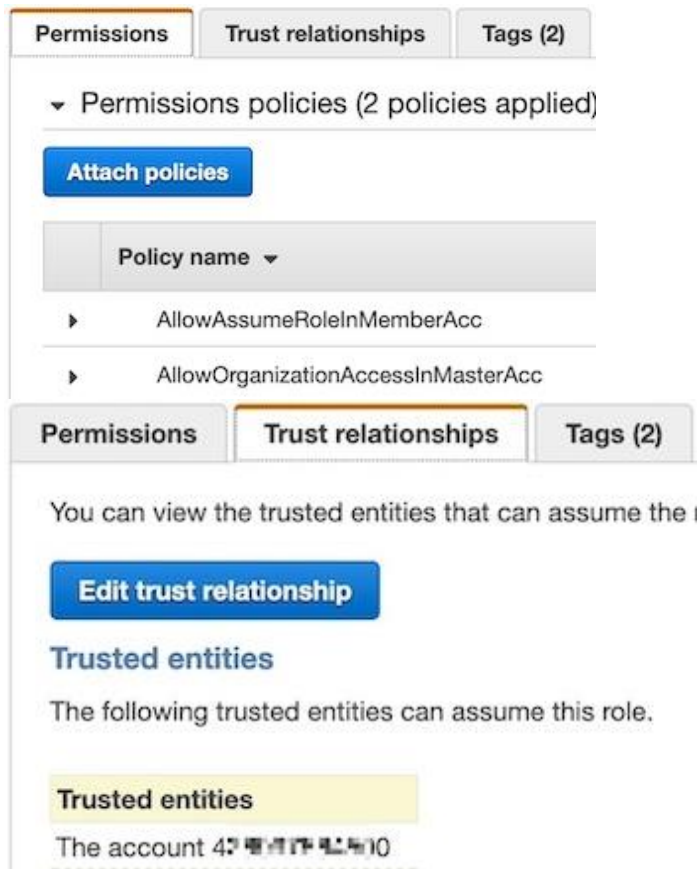


Figure: AWS IAM Trust Relationships Tab

An [External ID](#) is associated with the role. The External ID is an identifying string that is provided once a role is created. In order for the Trusted Entity (your Main account) to assume its role in the Master Account, it must supply this External ID. Providing the External ID of establishes a revocable relationship between the Trusted Entity and the Master account.

In the current Service Workbench deployment, the External ID is configured as a default value in the following string workbench:

```
main/solution/prepare-master-acc/config/settings/.defaults.yml
```

To change this value, create a stage-named configuration file (`mystagename.yml`) in the same directory. For more information, see the [Configuration](#) section. The following figure displays a screenshot image of the conditions that define how **Trusted Entities** assume a role.

Conditions


The following conditions define how and when trusted entities can assume the role.

Condition	Key	Value
StringEquals	sts:ExternalId	arn:aws:iam::123456789012:role/lambda-role

Figure: Defining Conditions for Trusted Entities

AWS Organizations

An AWS Organization is created in the **Master** account. The **Master** account is discussed in the Account Structure section of the Service Workbench User Guide . The AWS Organization use the **Master** account to create a separate account for each deployment. The account's name is the **Stage Name** used. The following figure shows a screenshot image of the AWS Organizations 'Accounts' tab.



AWS Organizations

[Accounts](#)
[Organize accounts](#)
[Policies](#)

Add account
Remove account

☐ Hide
Failed account creation requests

Filter

<input type="checkbox"/>	Account name	Email	Account ID	Status
<input type="checkbox"/>	★ AWS-1234567890	aws-1234567890@amazon.com	478901234567890	Joined on 5/25/20
<input type="checkbox"/>	★ AWS-9876543210	aws-9876543210@amazon.com	109876543210987	Created on 6/18/20

Figure: AWS Organizations Account Page

Amazon S3

Multiple Amazon S3 buckets are created by Service Workbench. Filtering by **Stage Name** shows the Amazon S3 buckets for a deployment. The following figure shows the Amazon S3 buckets for the Service Workbench deployment.

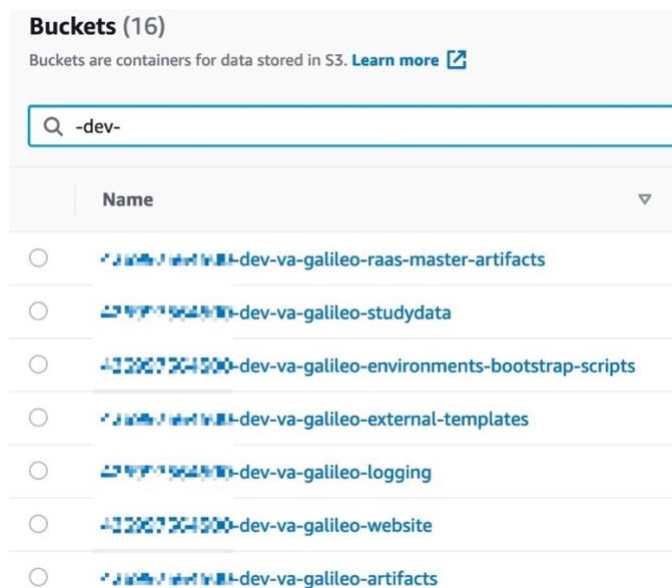


Figure: Amazon S3 Buckets for a Service Workbench Deployment

The 'studydata' bucket contains all the data for the various studies in this deployment at the individual and organization level. The following figure displays an image of the contents within the studydata bucket.

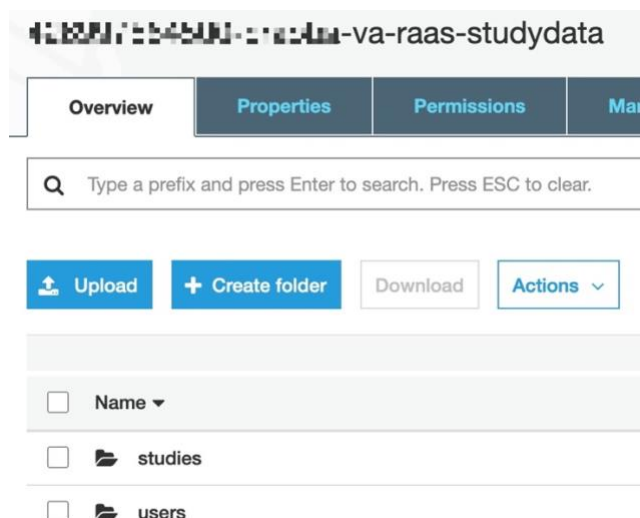


Figure: Amazon S3 StudyData Bucket

AWS Cost Explorer

Service Workbench has the ability to show actual cost incurred by workspaces running under the Master account. This is using the AWS Cost Explorer service in the AWS Management Console. AWS Cost Explorer must be manually set up for each Master account once in order to

allow requests for cost data to process. Setting this up requires background processes to complete in the Master account, which can take up to 24 hours.

Prepare the Master Account

This step is only required if Service Workbench is to be used to vend accounts in the AWS Organization, using the '**Create Account**' mechanism. If Service Workbench is to use only billing accounts imported through the '**Add Account**' mechanism, this step can be omitted.

In this step, deploy the **prepare_master_acc** SDC in the directory `main/solution/prepare-master-acc`. This will create in the Master account a role that allows **AssumeRole**, and has the **Main** account as its trusted entity. If you have deployed Service Workbench in a **Master** account, the **Main** account is also the **Master** account, and the trusted entity will be the same account ID as the **Master** account. If you have deployed Service Workbench in a **Member** account, the **Main** account is the **Member** account, and the trusted entity (the **Member** account) will be a different account ID than the **Master** account.

The default settings in this step are:

- **Main Account ID:** The current AWS Account ID
- **External ID:** The string **workbench**

These defaults are sufficient if you are deploying Service Workbench from the **Master** account, and the default profile has permissions for the **Master** account, since the **Main** account is also the **Master** account. If deploying Service Workbench in a **Member** account, you must create a configuration file to specify the **Main** account ID and the Profile to use. This profile must have permissions for the **Master** account.

Create a Configuration File

If deploying Service Workbench in an account other than that accessed by the current default profile, create a stage-named configuration file in the directory `main/solution/prepare-master-acc/config/settings` by copying `example.yml` to `<stage>.yml`. Edit the file as appropriate:

- **awsProfile:** The AWS Credentials profile with permissions for the Master account.
- **mainAccountID:** The 12 digit AWS Account ID for the Main AWS account, where the solution is deployed.
- **externalId:** As desired. The string **workbench** is often used. This string will be needed when creating an AWS account within Service Workbench.

Deploy the Prepare Master Account SDC

To deploy the `prepare_master_acc` SDC, perform the following:

1. Read the file: main/solution/prepare-master-acc/README.md.
2. Deploy the **Master** account SDC from the directory, main/solution/prepare-master-acc:

```
pnpx sls deploy --stage <stage>
```

3. To display the ARN of the **Master Role**, from the same directory:

```
pnpx sls info --verbose --stage <stage>
```

The **Master Role ARN** will be needed when adding accounts within Service Workbench.

Note: Running the convenience script `scripts/master-account-deploy.sh <stage>` will perform the same steps as `pnpx sls deploy`, above.

Master Role

The newly-created role will contain the String **MasterRole**, will have two policies, and will trust the **Main** account (see the figures below).

The screenshot shows the 'Permissions' tab of an IAM role. At the top, there are tabs for 'Permissions', 'Trust relationships', 'Tags (2)', 'Access Advisor', and 'Revoke sessions'. Below the tabs, it says 'Permissions policies (2 policies applied)'. There is a blue button 'Attach policies' and a link '+ Add inline policy'. A table lists the policies:

	Policy name ▾	Policy type ▾	
▶	AllowAssumeRoleInMemberAcc	Inline policy	✕
▶	AllowOrganizationAccessInMasterAcc	Inline policy	✕

Figure: Permissions Policies

The screenshot shows the 'Trust relationships' tab of the same IAM role. At the top, there are tabs for 'Permissions', 'Trust relationships', 'Tags (2)', 'Access Advisor', and 'Revoke sessions'. Below the tabs, it says 'You can view the trusted entities that can assume the role and the access conditions for the role. [Show policy document](#)'. There is a blue button 'Edit trust relationship'. The page is divided into two sections: 'Trusted entities' and 'Conditions'.

Trusted entities

The following trusted entities can assume this role.

Trusted entities
The account 268... 85

Conditions

The following conditions define how and when trusted entities can assume the role.

Condition	Key	Value
StringEquals	sts:ExternalId	galileo

Figure: Trusted Entities