Service Workbench installation guide

Contents

Overview

Service Workbench on AWS is a cloud solution that provides secure access to data, tooling, and compute power. Using Service Workbench, researchers can perform research in a secure and configured environment. Service Workbench enables the creation of baseline research setup. It simplifies data access and provides cost transparency.

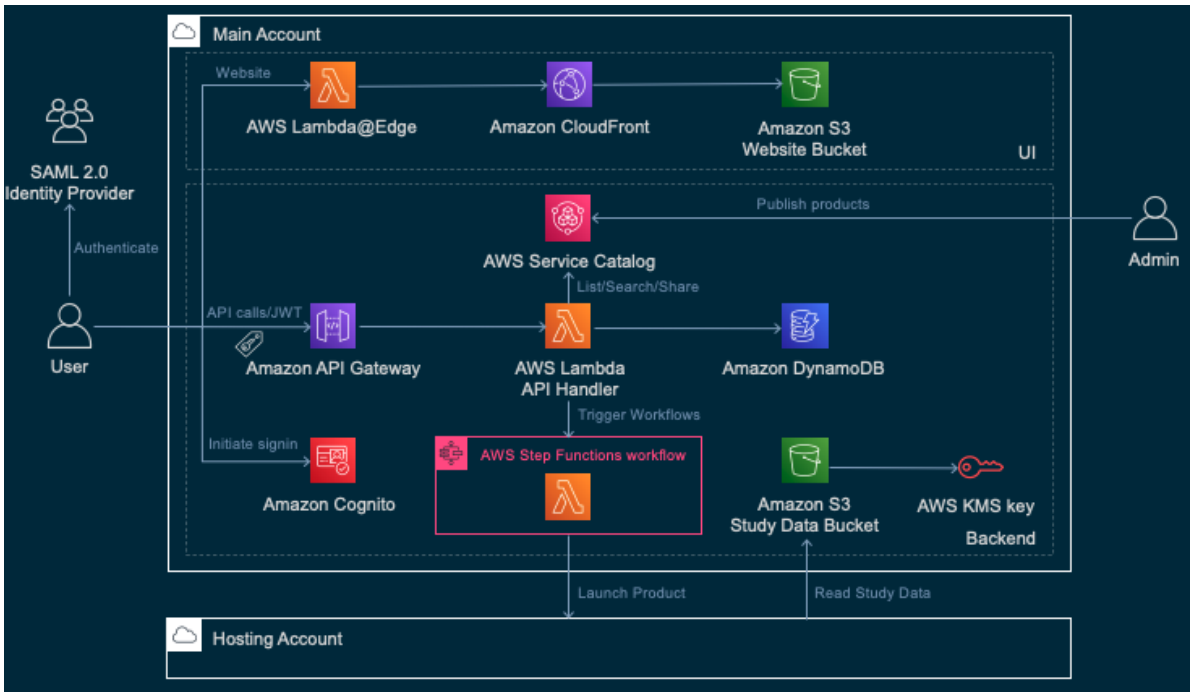
The Service Workbench installation guide covers installation of Service Workbench 3.0.0 and its related components. It provides information on Service Workbench architecture, installation, and troubleshooting.

Service Workbench architecture

Service Workbench integrates existing AWS services, such as Amazon CloudFront, AWS Lambda, and AWS Step Functions. Service Workbench enables you to create your own custom templates and share those templates with other organizations. To provide cost transparency, Service Workbench has been integrated with AWS Cost Explorer, AWS Budgets, and AWS Organizations.

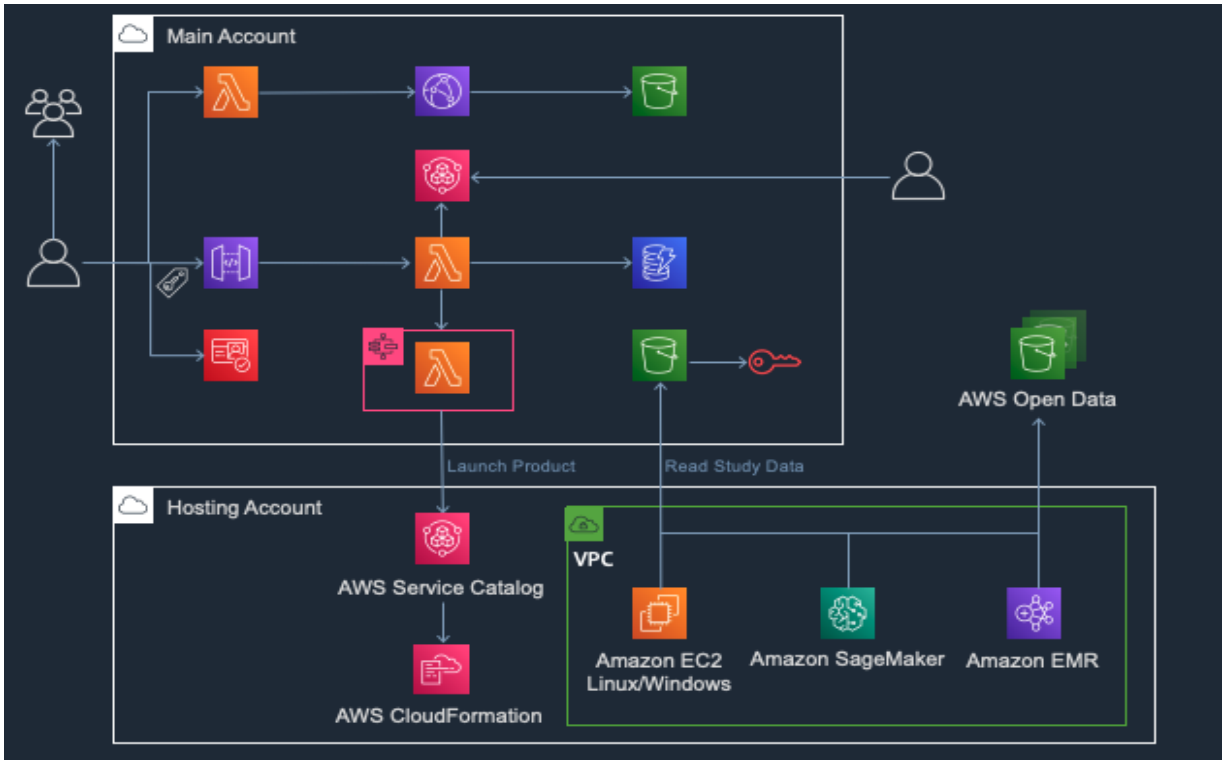
Main account

This is the account where Service Workbench infrastructure is deployed.



Hosting account

This is the account where compute resources are deployed.



Authentication

Service Workbench on AWS can use Amazon Cognito as a source of authentication. Amazon Cognito can federate with different authentication providers, which make it easier to federate with Active Directory, Auth0, or other identity providers.

Storage

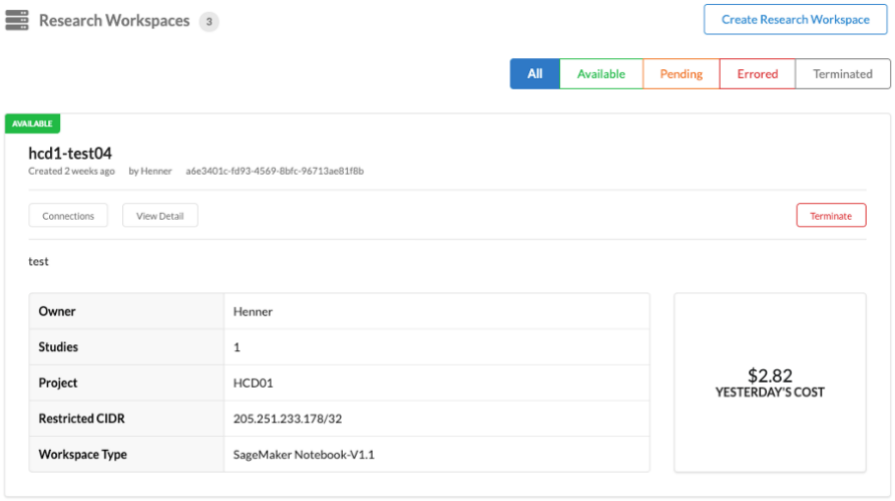
Service Workbench distinguishes between three types of research study data: *My Studies*, *Organizational Studies*, and *Open Data*. The former two are datasets stored and maintained either by you or the overall organization or groups. *Open Data* refers to data available through open data on AWS. Frequent scans against the open dataset ensure that latest open datasets are available to users.

AWS Service Catalog

The core of the Workspace management in Service Workbench is AWS Service Catalog. Here, the system finds and manages the templates that are used to define Workspaces. When you want to use a new Workspace type, it can be created as an AWS CloudFormation template inside AWS Service Catalog.

Workspace management

Besides provisioning an environment using templates, you can access your Workspaces, view billing details, or decommission them.



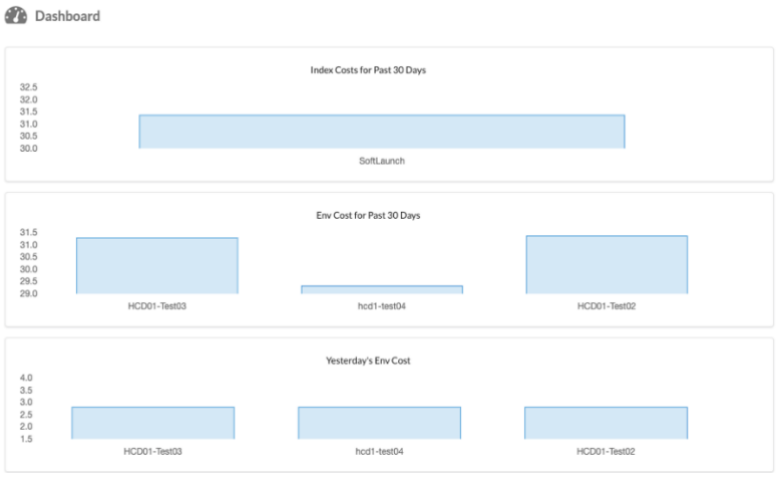
Cost control

Accounts, indexes, and projects

Service Workbench uses AWS accounts to manage compute Workspaces. This way, you can use different accounts for different projects, cost centers, or another purpose and manage cost. With the vending capability, an administrator can generate new AWS accounts under the same AWS Organizations by using the Service Workbench interface.

Dashboard

A dashboard displays a quick overview of the cost your Workspaces or projects have accumulated. This helps you to stay on budget and track Workspaces that possibly consume more resources.



Workspace sizes

When you create a Workspace from a template, you can choose the Workspace type in addition to multiple environment sizes. An administrator can pre-define these sizes and associate them with users based on individual permissions.

Configuration

SageMaker Notebook - Small

A Small Amazon SageMaker Jupyter Notebook workspace that comes with:

- TensorFlow
- Apache MXNet
- Scikit-learn

Estimated Cost

$1.63 per day

SageMaker Notebook - Medium

A Small Amazon SageMaker Jupyter Notebook workspace that comes with:

- TensorFlow
- Apache MXNet
- Scikit-learn

Estimated Cost

$96.54 per day

SageMaker Notebook - Large

A Small Amazon SageMaker Jupyter Notebook workspace that comes with:

- TensorFlow
- Apache MXNet
- Scikit-learn

Estimated Cost

$158.21 per day

Service Workbench installation components

Serverless framework and projects

Service Workbench on AWS is a serverless environment that is deployed using an event-driven API framework. Its components are spread across AWS Lambda instances, static webpages using Amazon CloudFront, and Amazon S3. It can use Amazon Cognito for authentication. Service Workbench relies on AWS Service Catalog to host and manage AWS CloudFormation templates that define the Workspaces. Service Workbench contains five serverless projects. You can find these components under the `<service_workbench>/main/solution` directory.

Component	Installation directory	What does it contain?

Infrastructure	`solution/infrastructure/`	The following AWS resources are created as part of this component deployment:
		• S3 bucket is used for logging the following actions:
		  ○ Studying data uploads.
		  ○ Accessing CloudFormation templates' bucket.
		  ○ Accessing CloudFront distribution service.
		  ○ Hosting the static Service Workbench website.
		• CloudFront distribution service to accelerate Service Workbench website access based on user location.

Backend	`solution/backend/`	After the environment has been deployed, the backend component creates and configures the following AWS resources: **S3 bucket** ○ Stores uploaded study data. This bucket also receives an encryption key from AWS Key Management Service for encrypting this data and making it available to the Service Workbench website. ○ Stores bootstrap scripts. These scripts are used to launch the Workspace instances like SageMaker, EC2, Amazon EMR. ○ Sets up IAM roles and policies for accessing Lambda functions and invoking step functions.
Amazon DynamoDB	Backend SDC creates DynamoDB tables	Stores information concerning user authentication, AWS accounts, workflows, access tokens, study data etc. This component is also responsible for deploying the following Lambda functions/services: • **Authentication layer handler** - Handles the authentication layer for API handlers.

		• **Open data scrape handler** - Handles scraping the metadata from the AWS open data registry. • **API handler** - Provides a path for public and protected API operations. • **Workflow loop runner** - Invoked by AWS Step Functions.
Edge Lambda	`main/solution/edge-lambda`	An inline JavaScript interceptor function that adds security headers to the CloudFront output response. This function is declared inline because the code requires API Gateway URL for the backend API operations.
Machine images	`solution/machine-images/`	Deploys Spot Instances using machine images for EC2 and Amazon EMR templates.
Prepare master accounts	`main/solution/prepare-master-acc`	Creates a master IAM role for organization access.
Post deployment	`solution/post-deployment/`	Creates an IAM role for the post deployment function with policies granting permission to S3 buckets, DynamoDB tables, KMS encryption key, CloudFront, and Lambda functions.
User interface	`solution/ui/`	Contains code used to create and support the UI functionality of the application.

Continuous integration/Continuous delivery

The Service Workbench solution includes the Continuous Integration/Continuous delivery feature:

- `cicd/cicd-pipeline/serverless.yml`
- `cicd/cicd-source/serverless.yml`

Installing Service Workbench

This section covers the following information required to install Service Workbench:

Section	Description
Pre-installation steps	Provides information on: • creating AWS account. • setting up AWS Cost Explorer. • creating cost allocation tags. • software requirements. • accessing reference documentation.
EC2 instance requirements	Provides information on creating and configuring an EC2 instance that will be used to install Service Workbench.
Accessing the Service Workbench source code	Provides information about the Service Workbench source code location.
Accessing reference documentation	Provides information on how to access Service Workbench documentation.
Service Workbench installation	Describes the Service Workbench installation procedure using EC2 instance/Cloud9. Administrators can install Service Workbench using either of the two operations.

Pre-installation steps

Tool requirements

The initial prerequisites include creating a main account in AWS, enabling the AWS Cost Explorer and activating the cost allocation tags.

Important: *Before installing Service Workbench, ensure that you have practical knowledge of core AWS services.*

Setting up the main account

Main account is the AWS account where Service Workbench is deployed.

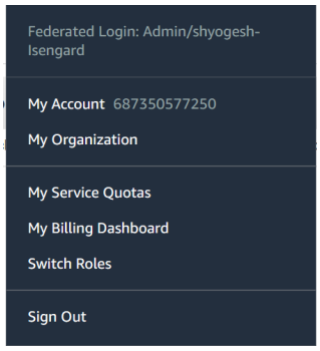
Enabling AWS Cost Explorer

In order to see the actual cost in dashboards and Workspaces, you must set up a master account in AWS Cost Explorer. The master account holds the AWS Organization that creates member accounts.

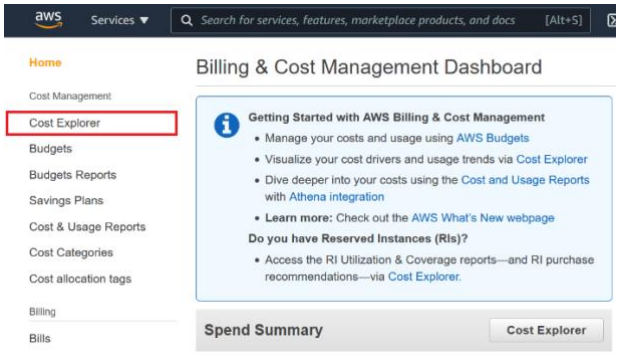
Note: You can enable AWS Cost Explorer even after installing Service Workbench.

To enable AWS Cost Explorer in the account into which you will be deploying Service Workbench on AWS, follow these steps:

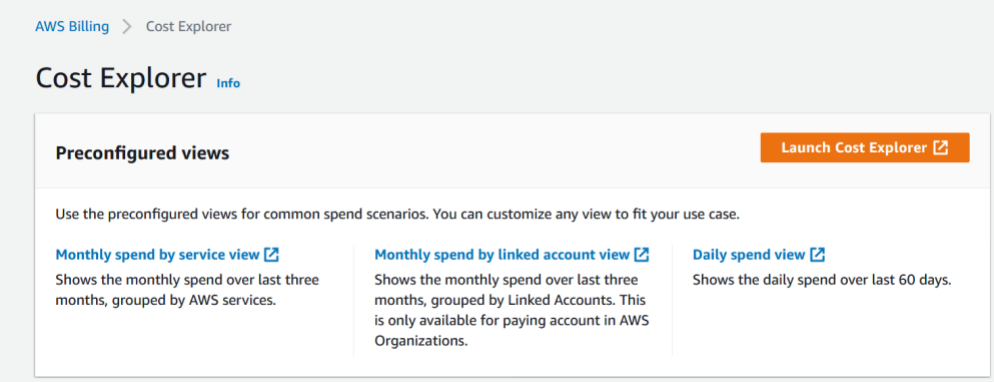
1. From the account drop-down, choose **My Billing Dashboard**.



2. Choose **Cost Explorer** from the sidebar.



3. Select **Launch Cost Explorer**.



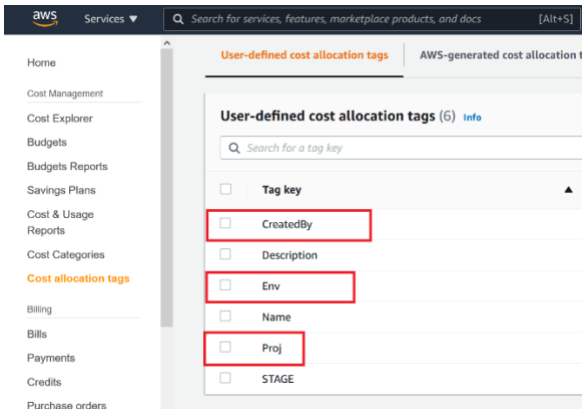
Note: The initialization can take up 24 hours; however, it does not have to be completed before starting the installation process.

Activating the cost allocation tags

Activate the necessary cost allocation tags in the **AWS Billing & Cost Management Dashboard**:

1. Sign in to the AWS Management Console and open the **Billing & Cost Management Dashboard** here.
2. In the navigation pane, choose **Cost allocation tags**.

3. Under **User-defined cost allocation tags**, choose the **createdBy**, **Env**, and **Proj** tags.



Note: There may be a delay after enabling the AWS Cost Explorer before these tags are available. If you have enabled Cost Explorer, but you do not see these tags through the AWS Console, you can still proceed with the installation. Check later (it could be up to 24 hours after enabling Cost Explorer) and enable the tags for cost reporting to function correctly in Service Workbench.

For more information, see Billing and Cost Management.

Software requirements

Software	**Functions**
Main AWS account	Deploys Service Workbench. We recommend you to dedicate an AWS account to this deployment. Additionally, you will also need admin access to the AWS accounts where you want to deploy Workspaces.
AWS Command Line Interface (CLI)	Starts AWS services from your terminal. You must have appropriate **AWS programmatic credentials** ready. You must also have appropriate rights to deploy the platform on an AWS account.
Packer installation	Used to build AMIs. For information on installing Packer, refer to this README.
Pnpm and Node.JS	• Installs and manages JavaScript packages specified in the platform's dependencies. See Pnpm. • Builds JavaScript files. See Node.js.

	• Builds and packages the code for cloud deployment. See Serverless framework. For any issues with using node.js, refer to the Troubleshooting section.
Go	Used for creating a multipart S3 downloader tool that is used in AWS Service Catalog EC2 Windows-based research environments.

EC2 instance requirements

This section covers the following:

Section	Description
Selecting an EC2 instance	Provides information on selecting the EC2 instance size for Service Workbench installation.
Configuring an EC2 instance	Describes the procedure of configuring an EC2 instance, creating an IAM role, and assigning the administrator role to the EC2 instance.
Installing the required software on EC2 instance	Describes the steps to install prerequisite software in the EC2 instance.
Configuration settings	Provides information about the configuration files required to install Service Workbench.

Selecting an EC2 instance

You can create an EC2 instance with the following specifications:

- **Amazon EC2 instance type**: Use a T2.medium Amazon EC2 instance or larger. Larger machines have faster networking and larger disks have higher performance.
  Important: 40 GB is the suggested disk drive size needed for installation.
- **VPC and subnets**: Use the default VPC and subnet.
- **AWS IAM role**: Attach it to your instance an AWS IAM role with sufficient permission, such as the administrator access. For more information, see Configuring an EC2 instance.

Configuring an EC2 instance

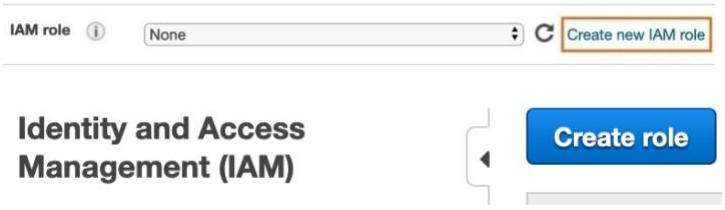
An Amazon EC2 instance can be assigned an instance profile that contains an AWS IAM role. The AWS IAM role will give the Amazon EC2 instance a set of permissions. The Amazon EC2 instance will only perform the actions defined by its AWS IAM role. Adding an AWS IAM role to the Amazon EC2 instance allows your application to make API calls securely—reducing 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.

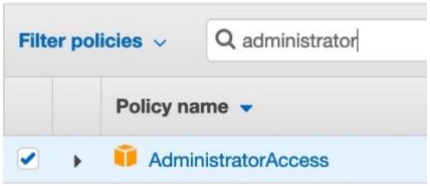
Creating a new IAM role

When creating a new Amazon EC2 instance, an instance profile may be assigned to the Amazon EC2 instance.

1. Choose **Create a new IAM role** located next to the AWS IAM role drop-down. To continue the process, highlight Amazon EC2 and proceed to permissions.

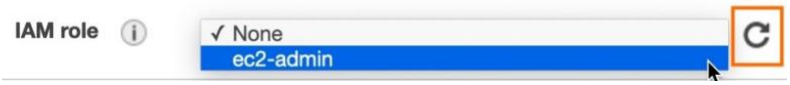


2. In **Permissions**, choose **AdministratorAccess** from the filter and proceed through **tags**.



3. In the **Review** page, enter the role name.

4. Return to the **Amazon EC2** tab, refresh the **IAM role** drop-down, and select your administrator role to attach to the new Amazon EC2 instance.

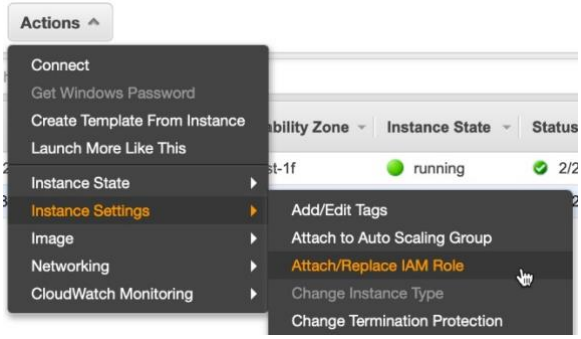


5. Create the Amazon EC2 instance.

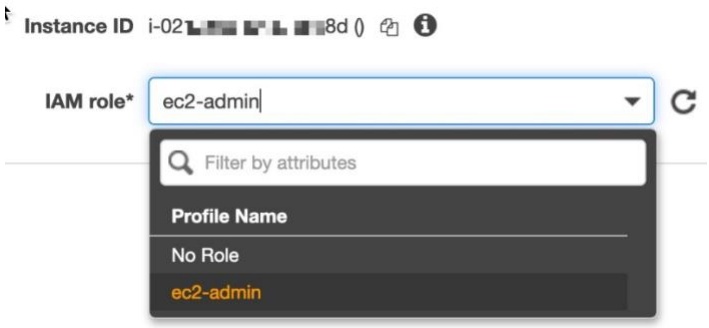
Adding a role to an existing instance

To add a role to an Amazon EC2 instance that is already running:

1. Select the Amazon EC2 instance in the EC2 console.
2. On the **Actions** menu**,** choose **Instance Settings**, **Attach/Replace IAM Role**.



3. In the **Attach/Replace IAM Role** screen, select the role you created and click **Apply**.



Installing the required software on EC2 instance

1. Install prerequisite software (serverless and pnpm) for installing Service Workbench on AWS on the EC2 instance:

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.35.3/install.sh | bash

source ~/.bashrc

nvm install 12

npm install -g serverless pnpm hygen
```

2. Run the following command to display the version of the serverless package:

```
serverless –v
```

Configuration settings

> **Note**: Setting the configuration is required. If you are deploying an installation, you can use the default configuration.

Stage name

A stage name is used to allow multiple Service Workbench deployments from the same account. It represents the name of the configuration files. For limitations in Amazon Simple Storage Service (Amazon S3) deployment buckets, the stage name must not be longer than five characters. Buckets are the fundamental containers in Amazon S3 for data storage.

You can select your own stage name. If you are planning to deploy the solution only once, a common convention, is to use your own login. In the following sections of the guide, customized stage name is represented as <*stage*>.

Separately deployable components

The Service Workbench code is divided into multiple (currently seven) separately deployable components (SDCs): **backend, UI**, **post-deployment**, **edge-lambda**, **infrastructure**, **machine-images**, **and prepare-aster-acc**. Each SDC has a directory in the location, `main/solution`. You can run the script either from the root directory or also deploy each SDC separately using individual scripts. For more information, see serverless framework and projects.

Prepare main configuration file

You can make a copy of the sample global AWS Config file, name it for your stage, and modify it. The current default values for the main configuration are stored in the default file in the directory, `main/config/settings/.defaults.yml`. If the stage-named settings file is not available, the values are read from this default file.

To create a custom (stage-named) settings file, in the directory, `main/config/settings`, copy `example.yml` to `<stage>.yml` and edit this new file. Default values are read from `.defaults.yml` unless the values are overridden in this file. Following table describes the default values:

Configuration	Value
awsRegion	`us-east-1`
awsProfile	No default; set this to your current AWS profile unless using a default or instance profile.
solutionName	`sw`
envName	Same as stage name
envType	`prod`
enableExternalResearchers	`false`

Table: Configuration values

Custom domain names

To use a custom domain name, enter the domain name and the ARN for the manually created TLS certificate.

```
domainName: host.domain.toplevel
certificateArn: <ARN>
```

Note: The current implementation assumes that DNS is handled elsewhere. A future improvement will automatically handle creation of the cert and Route 53 entries.

Namespace

The names of many deployed resources include a namespace string such as `mystage-va-sw`. This string is made by concatenating the following:

- Environment name
- Region short name (for example: `va` for US-East-1, `or` for US-West-2, defined in `.defaults.yml`)
- Solution name

Prepare SDC configuration files

Each SDC has a `config/settings` directory, where you can place customized settings. Settings files are named after the stage name <*`mystagename.yml`*>. Some of the SDC settings directories contain an

`example.yml` file that may be copied and renamed as a settings file for that SDC. Otherwise, a default file `.defaults.yml` in that directory is read and used regardless of the stage name.

Accessing the Service Workbench source code

Download the latest source code by using this link and run the following command:

```
curl -o serviceworkbench.zip <URL>
```

Supported regions

The following regions support all the AWS services and features needed to run Service Workbench:

- US East (Ohio)
- US East (N. Virginia)
- US West (N. California)
- US West (Oregon)
- Asia Pacific (Mumbai)
- Asia Pacific (Seoul)
- Asia Pacific (Singapore)
- Asia Pacific (Sydney)
- Asia Pacific (Tokyo)
- Canada (Central)
- Europe (Frankfurt)
- Europe (Ireland)
- Europe (London)
- Europe (Paris)
- Europe (Stockholm)
- South America (São Paulo)

Accessing reference documentation

- The Service Workbench documentation can be accessed here.
- Alternatively, you can access the documentation on your local machine, via a local web server, once you've completed the following steps:
  1. Download the source code for Service Workbench on AWS to your local computer. Note that this local copy is for documentation only.
  2. Install docusaurus (requires node and yarn)
  3. From within source code directory on your local computer, run the following commands to host the platform reference documentation:

     ```
     cd docs
     yarn start
     ```

  4. Access the documentation here.

Service Workbench installation

You can install Service Workbench either by creating and configuring an EC2 instance or by creating a Cloud9 instance. This section describes the steps to install Service Workbench by using either of the two options.

Installing AMIs for EC2 Workspace

In order to use EC2-based Workspaces, you must first install EC2 AMIs for these Workspaces. This process may be run in parallel while `environment-deploy.sh` is running. To run both simultaneously, open another SSH or SSM session to your EC2 instance.

1. Build AMIs for EC2-based Workspaces. This takes up to 15 minutes and may run in parallel with the main install script.
   a. Install packer from the root of your home directory:

```
# In your home directory

wget https://releases.hashicorp.com/packer/1.6.2/packer_1.6.2_linux_amd64.zip
unzip packer_1.6.2_linux_amd64.zip
```

   For more information about packer installation, refer to the README.

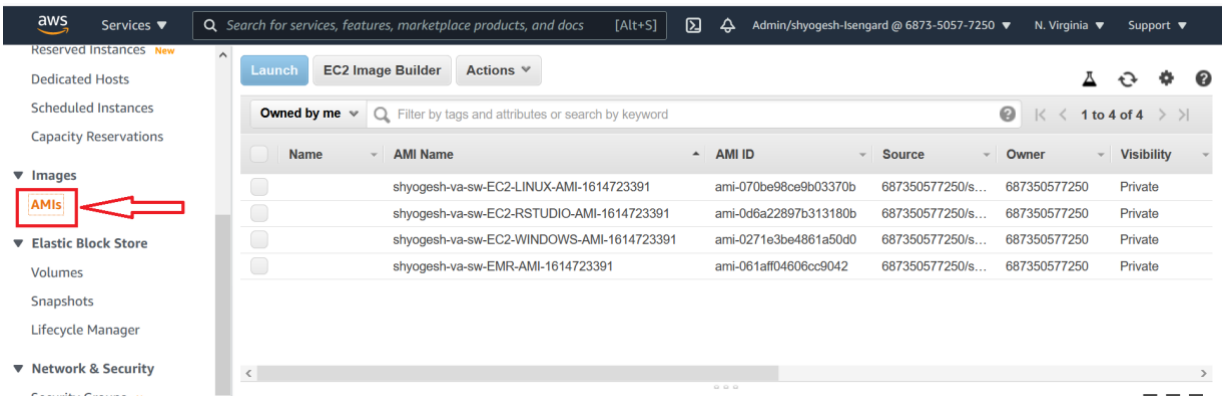
   b. Change to the directory containing the machine image source and build the AMIs. Ensure that the environment variable `STAGE_NAME` has been set before running this command.

```
# In the Service Workbench directory main/solution/machine-images

pnpx sls build-image -s ${STAGE_NAME}
```

2. Verify that the AMIs have been created:

   In the Amazon EC2 service console, select **AMI** in the left-hand navigation. You should see AMIs for EC2-LINUX, EC2-RSTUDIO, EC2-WINDOWS, and Amazon EMR.



Warning: Each AMI build results in a new set of AMIs.

Option1: Service Workbench installation using EC2 instance

Service Workbench can be installed in the following stages from your local machine using an Amazon Elastic Compute Cloud (Amazon EC2) instance:

Installation stages	Reference
Creating an EC2 instance.	EC2 instance requirements
Installing the node-based software.	Software requirements
Downloading and unpacking the Service Workbench code.	Accessing the Service Workbench source code
Choosing a stage name and edit the configuration file.	Configuration settings
Running the deployment script, signing and adding an AWS account to Service Workbench.	Installing Service Workbench

Installing Service Workbench

1. Download the Service Workbench on AWS source code using this link and then run the following commands:

   ```
   sudo yum install -y git
   git clone https://github.com/awslabs/service-workbench-on-aws.git
   ```

2. Create a main Service Workbench on AWS configuration file for your installation. To do this:
   a. Create an environment variable holding the stage name of your installation. The stage name is included in the name of the Amazon S3 storage bucket. Hence, it must be S3-compatible.

      Example:
      `export STAGE_NAME=dev`

      Note: Set the environment variable when you open a new terminal window.

   b. In the main configuration directory (`main/config/settings`), make a copy of the example configuration file using the suggested stage name `demo`. This creates the `dev.yml` file.
      `cp example.yml ${STAGE_NAME}.yml`

   c. In the newly created configuration file, uncomment and set values for the following values.
      i. `awsRegion` (for example: `us-east-1` or `eu-west-2`)
      Ensure that you use the same Region when you are using the AWS Management Console.
      ii. `solutionName` (for example: `sw`)
      The `solutionName` is used in S3 bucket names so must be S3-compatible.

      Note: Ensure that there is no leading space before the value name.

3. Run the main installation script. This takes up to 15 minutes and can be run in parallel with the next step (installing AMIs).
   `./scripts/environment-deploy.sh ${STAGE_NAME}`

4. Once the preceding step has completed, capture the root password and website URL. You can display the URL and root password again by running the following command:
   `scripts/get-info.sh ${STAGE_NAME}`

5. Verify that Service Workbench is running by using the URL and root password, using the user `root`.

Option2: Service Workbench installation using AWS Cloud9

You can install Service Workbench by using AWS Cloud9. This section provides information about the installation procedure for Service Workbench using AWS Cloud9 IDE.

Section	Description
Creating AWS Cloud9 instance	Describes the steps to create an AWS Cloud9 instance that will be used for Service Workbench installation.
Modifying the volume	Describes the steps to modify the volume size.
Increasing the partition	Describes the commands to increase the partition size for Service Workbench installation.
Installing Node Package Manager	Describes the commands to install Node Package Manager.
Cloning the Git directory	Describes the commands to clone Git directory that contains Service Workbench installation.
Making a copy of the environment file	Describes the steps to make a copy of the environment file and make required settings inside the file for Service Workbench installation.
Running the script to install Service Workbench	Describes the steps to install Service Workbench.

Creating AWS Cloud9 instance

1. Go to the AWS Cloud9 product page.
2. Click the **Create environment** button.
3. Enter the name and description for the AWS Cloud9 environment.
4. Choose **Next step**.
5. For **Instance type**, choose **m5.large (8 GiB + 2 vCPU)**
6. For **Platform**, choose **Amazon Linux 2**.
7. Choose **Next step**.
8. Review all the changes and choose **Create environment**.

Modifying the volume

For information on modifying the volume, refer to *Resize an Amazon EBS volume used by an environment*.

Increasing the partition

To increase the partition size, type the following command

```
sudo growpart /dev/nvme0n1 1
```

Increase the file system inside the partition

```
sudo xfs_growfs -d /
```

Verifying the file size

To verify the file size, type:

```
df -hT
```

Check the node version installed

Type `node --version`

To install long-term support version, type:

```
nvm install lts/erbium
```

Installing Node Package Manager

```
npm install -g pnpm
```

Install Packer

A packer is used to build AMIs. For steps on packer installation, refer to the README.

Verify the Go version

```
go version
```

Note: Install everything in one directory.

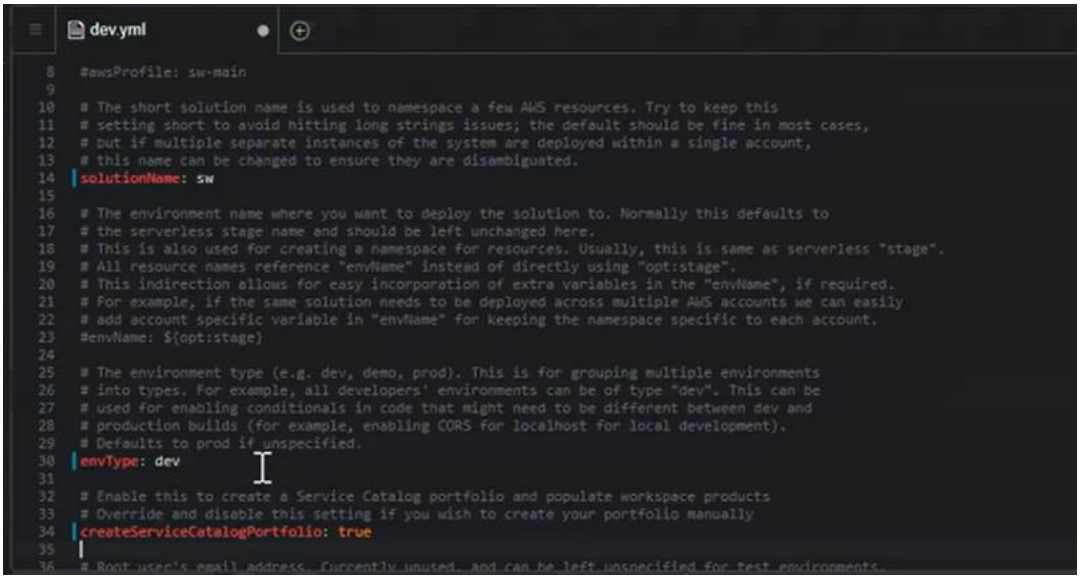
Cloning the Git directory

```
git clone https://github.com/awslabs/service-workbench-on-aws.git
```

See Installing Service Workbench for more information.

Making a copy of the environment file

1. In the file explorer, choose **example.yml**.
2. Copy this file and create a new version. Example, **dev.yml**.
3. Uncomment the following in **dev.yml**:
    a. **awsRegion**: us-east-1
    b. **solutionName**: `sw`
    c. **envType**: `dev`
    d. **createServiceCatalogPortfolio**: `true`



4. Save **dev.yml**.

Running the script to install Service Workbench

`scripts/environment-deploy.sh <stage>`

Example: `scripts/environment-deploy.sh dev`

Copying CloudFront URL details

Once the installation completes, the following details are displayed on your screen. Note the website URL and the root password. You can use this URL and password to sign in to Service Workbench.

Setting up RStudio ALB workspace

Overview

You can access the RStudio workspace type by using the template and AMI provided in AWS partner's repository. The following table summarizes the sequence of steps that you need to follow to complete the installation for RStudio ALB workspace in the Service Workbench environment.

Step.	Task	Description
1.	Creating main account and hosting accoun	Provides information on how to set up the environment used for RStudio workspace types. In these accounts, you will set up hosted zones, configure the certificates required for deploying RStudio ALB workspace.
2.	Creating a public hosted zone in the main account	Provides routing information about how you want to route traffic for a domain/subdomain.
3.	Creating a new name server record in the shared domain	Provides a name to the hosted zone. Here, you will copy the values (containing routing information) that you created in the main account.
4.	Requesting a public certificate in main account for the hosted zone	Request a public certificate in the main account for the hosted zone. The region is chosen as `us-east-1`. Using this option, you request a public certificate from Amazon. By default, these public certificates are trusted by browsers and operating systems.
5.	Creating a staging file using the latest Service Workbench code	Download the latest Service Workbench code and set up the staging file for deploying RStudio ALB workspace. For more information about staging file, see Configuration settings.
6.	Creating AMIs	Provides the basis of RStudio environment that researchers use for research. Refer to the AWS partner's README for more information.
7.	Installing EC2 RStudio Server	Deploys RStudio into Service Workbench's portfolio. Refer to the AWS partner's README for more information.
8.	Requesting a new public certificate within member account for hosted zone domain	Provides steps on how to request a public certificate for member (hosting) account using ACM. The certificate contains the ARN value that is used to import the products.
9.	Creating a new record in the main account hosted zone	Validates the member account public certificate. Without the validation, no workspaces can be created.
10.	Accessing RStudio Workspace in Service Workbench	Configure RStudio ALB using Service Workbench.

Creating main account and hosting account

Task: You will learn to create a main account and a hosting account and assign **Admin** role to these accounts.

1. Choose **Create/Register Account**.
2. On the **Account Creation** page, choose **Create**.
3. On the **Create AWS Accounts** page, specify the account email address, account name, secondary owner, financial owner, ownership group, description, and account type.
4. Choose **Submit**.
5. Choose **Manage Account**.
6. Create a console role:
    a. Choose **Add**.
    b. For **One Click Roles**, choose **Admin**.

Note: Follow the same steps above to create a hosting account and choose the admin role. In the description, specify that it is a hosting account.

Creating a public hosted zone in the main account

Task: You will learn to create a public hosted zone for the main account and add routing information to it.

1. Sign in to the AWS Management Console.
2. Under **AWS Services**, choose **Route 53**.
3. On the Route 53 dashboard, choose **Hosted zones**.
4. Choose **Create hosted zone**.
5. On the **Create hosted zone** page:
    a. For **Domain name**, enter the domain name.
    b. For **Description – *optional***, enter the description.
    c. For **Type**, choose **Public hosted zone**.
6. Choose **Create hosted zone**.
7. Copy the values from the **Value/Route traffic to** field for later use.

Creating a new name server record in the shared domain

Task: You will learn to create a new record name for the public hosted zone.

1. Sign in to the AWS Management Console.
2. Under **AWS Services**, choose **Route 53**.
3. On the Route 53 dashboard, choose **Hosted zones**.
4. Choose the hosted zone that you just created.
5. Choose **Create record**.

6. On the **Quick create record** page, choose **Add another record**.
7. For **Record name**, enter the name.
8. For **Record type**, choose **NS – Name servers for a hosted zone**.
9. For **Value**, enter the values that you copied from the **Value/Route traffic to** field earlier.
10. For **TTL (seconds)**, enter the value.
11. For **Routing policy**, choose Simple routing.
12. Choose **Create records**.

Requesting a public certificate in main account for the hosted zone

Task: You will learn how to request a certificate for the main account in AWS Certificate Manager (ACM) and access the CNAME record details from that certificate.

Requesting a certificate in AWS Certificate Manager

1. Sign in to the AWS Management Console.
2. Go to AWS Certificate Manager.

   Note: To use an **ACM certificate** with Amazon CloudFront, you must request or import the **certificate** in the **US East** (N. Virginia) region.

3. Choose **Request a certificate**.
4. On the **Request certificate** page, choose **Request a public certificate**.
5. Choose **Next**.
6. On the Request a public certificate page:
   a. For **Domain names**, enter the domain name.
   b. For **Select validation method**, choose **DNS validation – recommended**.
7. Choose **Request**.
8. Refresh your screen to view the newly created certificate.
9. Choose the certificate to view the details.
10. Under **Domains**, copy values for **CNAME name** and **CNAME value**.

Creating a new record in the main account

1. Go the main account.
2. Choose the hosted zone created earlier. See Creating a public hosted zone in the main account.
3. Create a new record.
4. For **Record name**, paste the first part of the CNAME record created in the previous section.
5. For **Value**, paste the CNAME value copied in the previous section.
6. Choose **Create records**.

Creating a staging file using the latest Service Workbench code

Task: You will learn how to create and configure the staging file for setting up the RStudio ALB workspace.

1. Go to the `/main/config/settings` directory.
2. Create a new staging file. Example:
   `cp example.yml albr.yml`
3. In the newly created stage file, remove comments from the following sections and enter the values:
    a. Keep `awsRegion` as `us-east-1`. You can change the value of `awsProfile` , for example, `rstudio`.
        i. Update the configuration (`config` file) and credentials (`credentials` file) within the `.aws` folder so that the new account details are included. The updated `config` file should have the following values: `region` (default is `us-east-1`), `output` (yaml), `account_id`, and `role_name`.
        ii. To update credentials, you need to create a CLI user. For more information on creating a user, see IAM users.
    b. For `solutionName`, enter `sw`.
    c. For `envName`, enter the stage name, for example, `albr`.
    d. For `envType`, enter a name, for example, `dev`.
    e. For `createServiceCatalogPortfolio`, enter the value as `true`.
    f. For `rootUserEmail`, enter the email address.
    g. For `domainName`, enter the domain name of the main account hosted zone.
    h. For `certificateArn`, copy the certificate ARN of the main account and paste it.
    i. From `defaults.yml` file, copy the `hostedZoneId` into the newly created staging file (for example, `albr.yml`). Copy the **Hosted zone ID** details **Hosted zone page** in Route 53 and paste it in `hostedZoneId` in your staging file.

Your new stage file is now ready for deployment.

Creating AMIs

Follow the instructions in the README to install EC2 RStudio server AMIs. You can go to your main account and access these AMIs under Images.

Note: Edit the configuration file (`RStudio/machine-images/config/infra/configuration.json`). You need to update the `region` (to match your deployment config file), `amiName` (suggested: rstudio-alb), and `awsProfile` (to match your deployment config file). Update `stageName` to match your deployment configuration file.

Installing EC2 RStudio Server

Follow the instructions in the README to install EC2 RStudio server.

Requesting a new public certificate within member account for hosted zone domain

Task: You will learn to request a public certificate for member (hosting) account using ACM and copy the ARN value.

1. Sign in to the AWS Management Console.
2. Choose **Certificate Manager**.
3. On the **AWS Certificate Manager** page, choose **Request a certificate**.
4. Choose **Next**.
5. For **Certificate type**, choose **Request a public certificate**.
6. Choose **Next**.
7. On the **Request a public certificate** page:
    a. For **Domain names**, enter the domain name in the following two ways: `example.corp.com` and `*.example.corp.com`. Here, `example.corp.com` is the full domain name.
    b. For **Select validation method**, choose **DNS validation – recommended**.
8. Choose **Request**.
9. On the Certificate page, within **Certificate status**, copy the ARN. The ARN will be used to import the products.

Creating a new record in the main account hosted zone

Task: This task validates the member account public certificate.

1. Go to the hosting account certificate manager.
2. On the certificate page, copy values for **CNAME name** and **CNAME value.**
3. Switch over to the main account.
4. In the management console, choose **Route 53**.
5. Choose **Hosted zones**.
6. Choose the domain name.
7. Choose **Create new record**.
8. For **Record name**, paste the first part of the CNAME record created in the previous section.
9. For Record type, choose **CNAME – Routes traffic to another domain**
10. For **Value**, paste the CNAME value copied in the previous section.
11. Choose **Create records**.

Accessing RStudio Workspace in Service Workbench

Task: You will log in to Service Workbench, create a user and configure the account for using RStudio ALN workspace.

1. Sign in to Service Workbench first time as the root user.
2. Add a local user.
3. For **Accounts**, **AWS Accounts**, **Add AWS Account**, add the member account.

4. For **Accounts**, **Indexes**, **Add Index** associated with member account, create an index.
5. For **Accounts**, **Projects**, **Add Project** associated with that index, create a new project.
6. For **Users**, **Users**, **Detail**, **Edit**, associate the user with the new project.
7. Import EC2 RStudio Server and add new configuration for the workspace type. `ACMSSLCertArn` is asking for the member account certificate ARN.
8. Enter the instance type for the EC2 instance. `AmiId` is the AMI ID of the newly created AMI in the main account.
9. Approve the configuration.
10. Create a new workspace in by selecting the RStudio Server workspace type and configuration.

Connect to the RStudio Workspace. Make sure popups are enabled.

Upgrading Service Workbench

We are requiring one of two options for customers who have Service Workbench installations:

- **Upgrade**:
  - **When installed from the CLI**: Download the latest source code, configure, run the installation script.
  - **When installed using AWS solution** (CloudFormation template): Edit and re-run the AWS CodeBuild project.
  - After either of these two cases, perform the post-upgrade process.
- **Delete**: Remove Service Workbench

Upgrading AWS solution installation

This procedure is for upgrading Service Workbench installations automatically installed from the AWS solution. In this installation model, a CloudFormation template initiates the installation, which is performed by AWS CodeBuild project. To upgrade a Service Workbench deployment, you need access to the Service Workbench installation.

1. Log in to the AWS Management Console for the account where Service Workbench is installed.
2. Open the AWS CodeBuild console and locate the Service Workbench project with the name **swb-Setup.**
3. Enter the project, click on the most recent successful build, and open the **Environment Variables** tab. Note the following values:
   a. `OBJECT_KEY_NAME:` Record the version number (for example: '1.4.3') from this string, which is used as part of the URL from where to download the Service Workbench source code.
   b. `SOLUTION_NAME:` Default value is `swb.`
   c. `STAGE_NAME:` Default value is `test.`
4. In the **Build projects** page:
   a. For **Edit**, choose **Environment**.
   b. Expand the **Additional configuration** section.
5. Edit the value of `OBJECT_KEY_NAME` and set it to `service-workbench-on-aws/v1.4.5`
6. If necessary, set the values of `SOLUTION_NAME` and `STAGE_NAME` to match those previously used.

7. Choose **Update environment**, which returns you to the **Build projects** page.
8. Choose **Start build**. The project runs for 20-30 minutes.
9. Test the deployment by visiting the site URL.
10. Update each account in Service Workbench by following the instructions in Post upgrade (after either upgrade process).

Upgrade process for command line installations

You can upgrade Service Workbench deployments that were installed from the command line using downloaded source code.

Prerequisites

- Access to the account where Service Workbench is installed.
- An EC2 deployment instance to be used in this account.
- The latest source code.
- Configuration files matching those used in the original deployment.

Accessing the account

Similarly, to an initial installation, it is easy to perform an upgrade from an EC2 instance. To do this, use an instance role giving access to the Service Workbench account. To set up an instance and install the prerequisite software, see pre-installation steps.

Log in to the instance and test access to the account by listing the S3 buckets:

```
aws s3 ls
```

Seven buckets with similar name stems are displayed. The name stem includes several values needed later in the configuration files. They have the following format:

`<account>-<stage>-<region>-<solution>-<purpose>`

For example, the bucket `012345678901-demo-va-sw-studydata` is in account **012345678901**, stage name **demo**, Region code **va** (us-east-1), solution name **sw**, and it hosts the study data.

Downloading source code

On the deployment instance, verify if there is a directory named `service-workbench-on-aws`, from the initial deployment. If yes, either rename it or move it into a subdirectory before downloading the source code. This prevents name duplication.

Download the latest source code from GitHub using `git clone`, as described in Installing Service Workbench.

Setting the configuration

Follow the steps in Configuration settings, where the name of the file comes from the stage name in the bucket name stem. In the configuration file, configure the settings:

- `awsRegion`: Refer to the Regional code mapping section to verify the full region name for the region code. For example, set `awsRegion: us-east-1` for the region code **va**.
- `solutionName`: Use the solution name from the bucket name stem (for example: `solutionName: sw`)

Upgrading Service Workbench

After creating the configuration file, run the main deployment script as described in Installing Service Workbench.

```
# In the main Service Workbench directory
./scripts/environment-deploy.sh ${STAGE_NAME}
```

After the upgrade, update each account in Service Workbench as described in the Post upgrade (after either upgrade process) section.

Upgrading to deprecated internal authentication

In the Service Workbench version 5.0.0, internal authentication has been deprecated. To upgrade to this version from your current version of Service Workbench, the administrators need to prepare the environment for an upgrade by deactivating, terminating, or disassociating internal users and the resources to which they are associated. The following tasks must be completed to prepare the Service Workbench environment for an upgrade:

- Workspaces created by internal authentication users must be terminated.
- SSH keys generated by internal authentication users must be deactivated or deleted.
- Internal authentication users cannot be associated to any projects.
- Internal authentication users cannot be associated to any organizational studies.
- Internal authentication users (including the root user) must be deactivated or deleted.

To allow administrators to revert to the Service Workbench version when internal authentication was supported, we recommend deactivating instead of deleting the resources.

If you haven't created IdP/native Cognito users before deploying version 5.0.0, you can re-deploy the previous release to create them, and then retry deploying version 5.0.0.

Important: Follow these instructions in this order to avoid resources being removed before they should have been. Some resources depend on others being active to allow for the deactivation.

You may be required to use the Service Workbench API library to complete this process. You can either do this by using Postman (an external software that can either be used as a web version or download to your desktop), or the command line package cURL. For instructions on how to set up your Postman environment, see Setting up Postman for Service Workbench.

Terminating workspaces

1. Log in to Service Workbench as an administrator using either the internal or external authentication administrator's account.
2. Terminate workspaces: Go to the Workspaces page and terminate all active workspaces owned by internal users.

Deactivating/deleting SSH keys

Using Service Workbench UI to delete SSH Keys

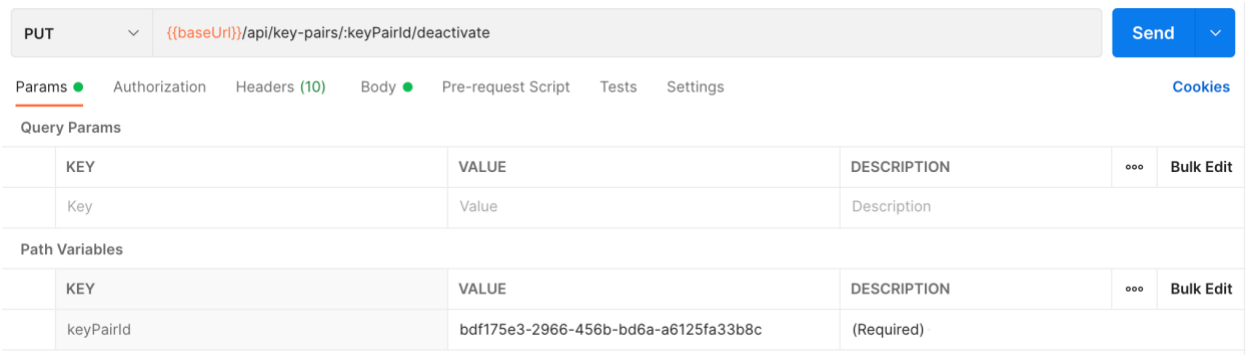
Using the Service Workbench UI, you can only see your own SSH keys and can only delete the SSH keys. One option is to have all of your users log in before deployment and delete all of their SSH Keys. Alternatively, if you attempt to deploy the latest version of Service Workbench, the output of the failed upgrade should list the SSH Key IDs you need to deactivate. An administrator can deactivate another

user's SSH Key using the `{{baseURL}}/api/key-pairs/:keyPairId/deactivate` API. Repeat whichever process you choose for each SSH key that needs to be deactivated.

Using Postman to deactivate SSH keys

When using Postman, make sure you have a valid token in your environment variable (for instructions to get this token, see Getting authentication token.

1. In the Collections or APIs section, navigate to the `service-workbench-on-aws/api/key-pairs/{key Pair Id}` API.
2. Choose **PUT Deactivate key pair**.
3. In the **Params** section, paste an SSH Key Id from the output of the failed deployment in the path variable for **keyPairId**.
4. Choose **Send**.



Using command line to deactivate SSH keys

When using command line, make sure you have a valid token. For more information on how to get this token, see Getting authentication token. In your terminal, paste and substitute the following:

```
curl --location --request PUT 'http://{{baseURL}}/api/key-
pairs/{{keyPairId}}/deactivate' \
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer {{token}}' \
--data-raw '{
 "rev": 0
}'
```

Disassociating projects

1. Within the Service Workbench UI, go to the **Users** page and choose the **Users** tab.
2. For any user that the identity prover is internal, if the **Project** column has a value other than **none**, choose **Detail**.
3. Choose **Edit**.
4. Choose the **X** next to any values in the **Project** field.
5. Choose **Save** and repeat for all internal authentication users.

Disassociating organization studies

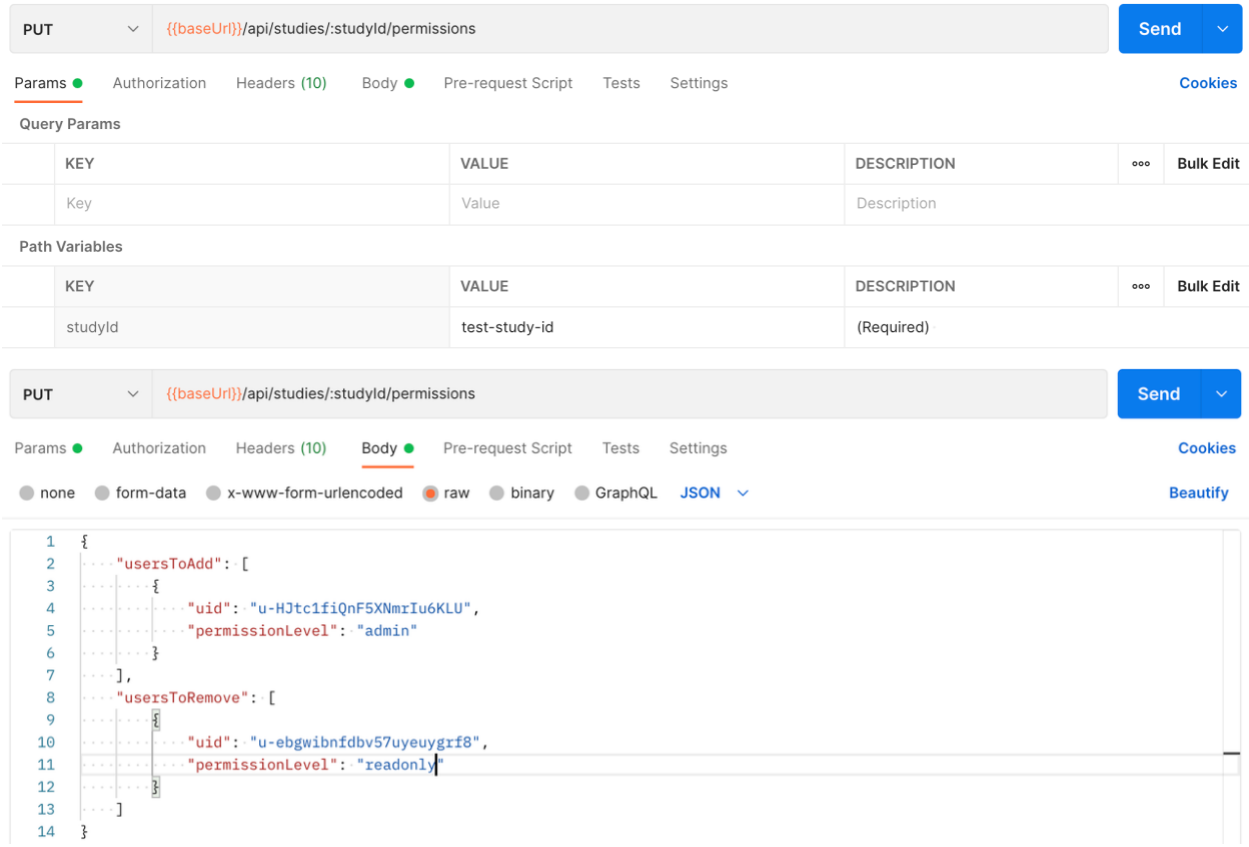
Using Service Workbench UI to disassociate organization studies

If you attempt to deploy the newest version of Service Workbench, the output of the failed upgrade should list the organization studies *and their admins* you need to disassociate with internal authentication users. A Service Workbench administrator user can change permissions on any organization study using the `{{baseURL}}/api/studies/:studyId/permissions` API. Repeat whichever process you choose for each organization study that needs to have permissions edited. You can combine removing multiple users from the same organization study but can only edit one study per request.

Using Postman to disassociate organization study

When using Postman, make sure you have a valid token in your environment variable. For more information on how to get this token, see Getting authentication token.

1. In the Collections or APIs section, navigate to the `service-workbench-on-aws/api/studies/{study Id}/permissions` API.
2. Choose **PUT update study permissions**.
3. In the **Params** section, paste an organization studyId from the output of the failed deployment in the path variable for studyId.
4. In the **Body** section, add the user Ids of the users you want to add and to remove and their respective permission levels.
5. Choose **Send**.



Using command line to disassociate organization study

When using command line, make sure you have a valid token. For more information on how to get this token, see Getting authentication token. In your terminal, paste and substitute the following:

```
curl --location --request PUT 'http://{{baseURL}}/api/studies/{{study-
id}}/permissions' \
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer {{token}}' \
--data-raw '{
 "usersToAdd": [
 {
 "uid": "u-HJtc1fiQnF5XNmrIu6KLU",
 "permissionLevel": "admin"
 }
 ],
 "usersToRemove": [
 {
 "uid": "u-ebgwibnfdbv57uyeuygrf8",
 "permissionLevel": "readonly"
 }
 ]
}'
```

Before you proceed to the last step, try to deploy to upgrade to the newest version of Service Workbench. If any checks besides the check for internal authentication users do not pass, remedy that by repeating the respective step above before proceeding in this document.

Deactivating internal authentication users

This is the last step. To deactivate internal authentication users:

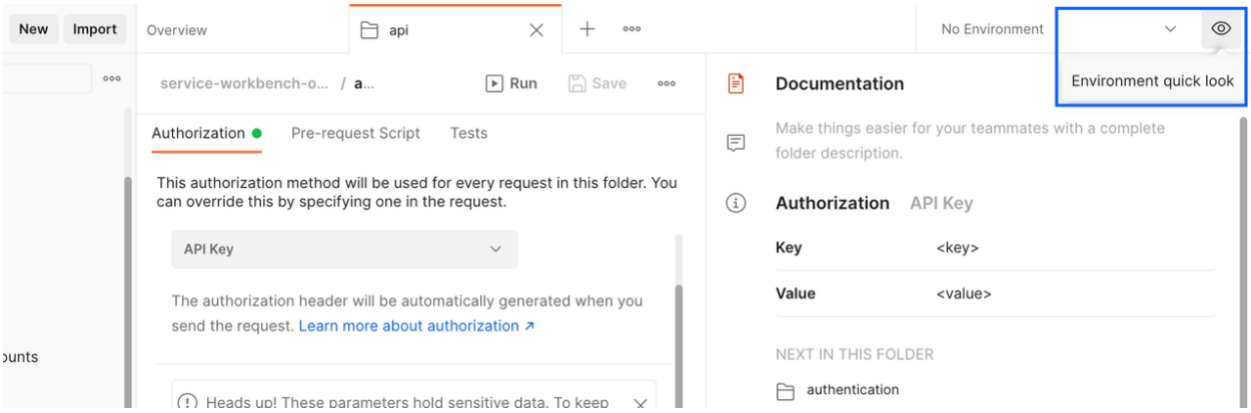
1. On the Service Workbench UI, go to the **Users** page.
2. For each internal user, choose **Detail**.
3. Choose **Deactivate User**. Only the root user can deactivate itself. You must have Service Workbench version 4.2.0 or later deployed to deactivate the root user as itself. Deactivate the root user after at least one external authentication user has been created.
4. To deactivate the root user, log in to the Service Workbench UI as the internal authentication root user.
5. Choose **Detail**.
6. Choose **Deactivate user**. You immediately lose access to the UI if you are logged into the user account you just deactivated.

After completing all of those steps, you are now ready to deploy and see if you pass all of the checks. If a check does not pass, check the output on the command line and repeat the step to deactivate/terminate/disassociate the listed resource. Check the CloudWatch log for predeployment in the main account for more information. There should be additional logging there to help debug.

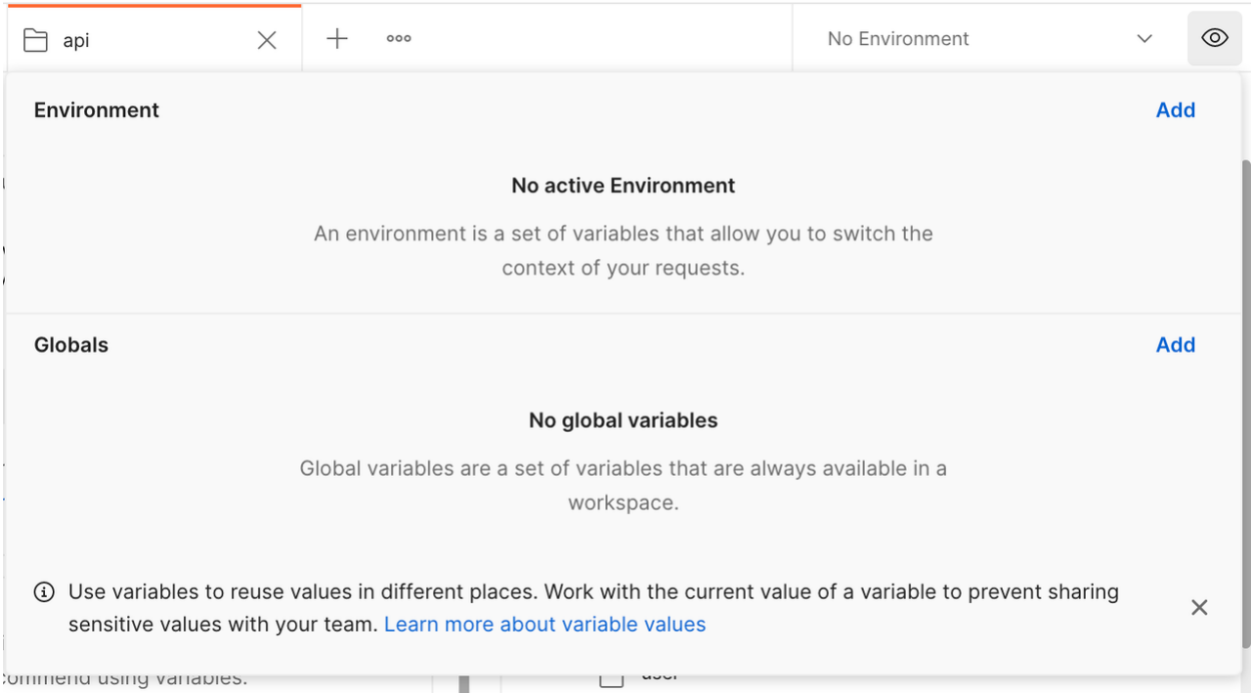
Important: After the upgrade to Service Workbench 5.0.0, My Studies owned by internal users would need to be transferred over to external IdP/Cognito native users. Service Workbench administrator must migrate these My Studies to the respective external authentication users after deployment. Administrators are responsible for completing this migration. This can be done by utilizing an API tool of choice. For instructions on how to complete this migration, see Migrating My Study resources.

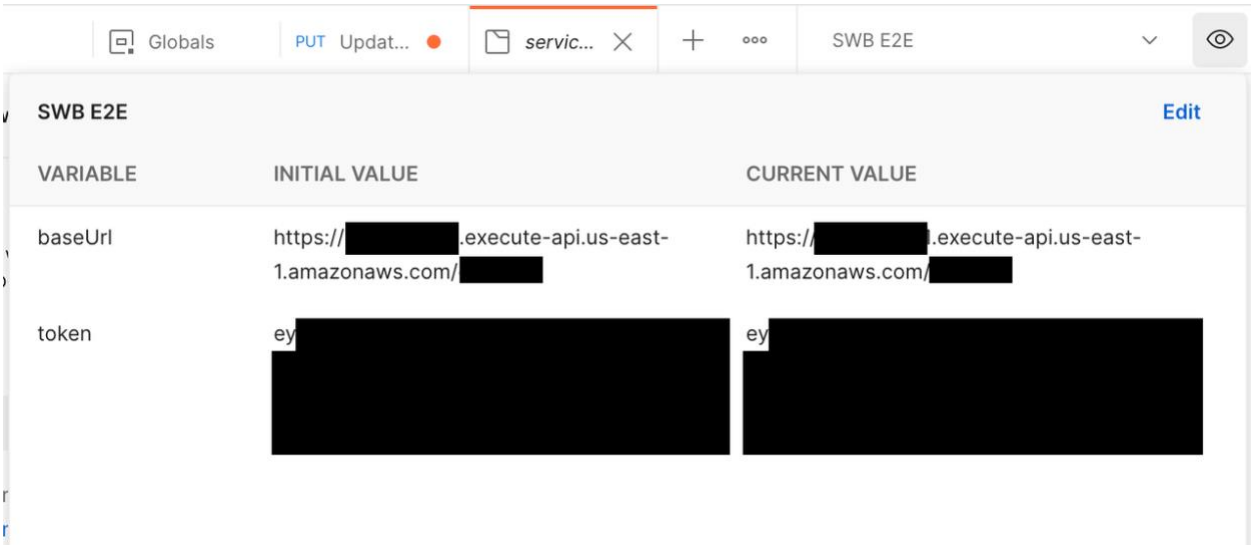
Setting up Postman for Service Workbench

1. Download Postman.
2. After opening Postman, either sign in, create a free account, or continue to app without signing in (there should be an option for this at the bottom).
3. On the navigation pane, choose **Import**.
4. Import `openapi.yml` from the Service Workbench repository. This imports the APIs as a collection if you are not logged in.
5. Choose the **Environment quick look** icon.



6. In the **Environment** section, either choose **Add** (if you do not have an environment yet set for the Service Workbench environment you need to edit) or select the environment that corresponds to the Service Workbench environment you are preparing for upgrade. It will appear as:





a. If you are adding a new environment, you will need to add two new variables.

   i. Create a `baseUrl` variable. The value should be the API endpoint for the Service Workbench environment. This can be found in the information displayed after deployed the environment or after running `scripts/get-info.sh <stage>` in your `service-workbench-on-aws` repository folder:

```
:::service-workbench-on-aws % scripts/get-info.sh stage
STAGE supplied as command line argument: stage
------------------------------------------------------------------------
--------
Summary:
------------------------------------------------------------------------
--------
Env Name  :stage
Solution  : sw
Website URL : https://------------.cloudfront.net
API Endpoint  : https://----------.execute-api.us-east-
1.amazonaws.com/stage
...
------------------------------------------------------------------------
--------
```

   ii. Create a token variable. For more information on how to set this value, see Getting authentication token.

   iii. Save these values and set this environment as active by choosing the three dots at the top of the panel and then choosing **Set as active environment**. Now, when you choose the **Environment quick look** icon, you should see the information you just added.

b. If you are using an existing environment, navigate to that environment from the dropdown that displays your current active environment.

   i. Choose the desired environment.

   ii. Set this environment as active by choosing the three dots at the top of the panel and then choosing **Set as active environment**.

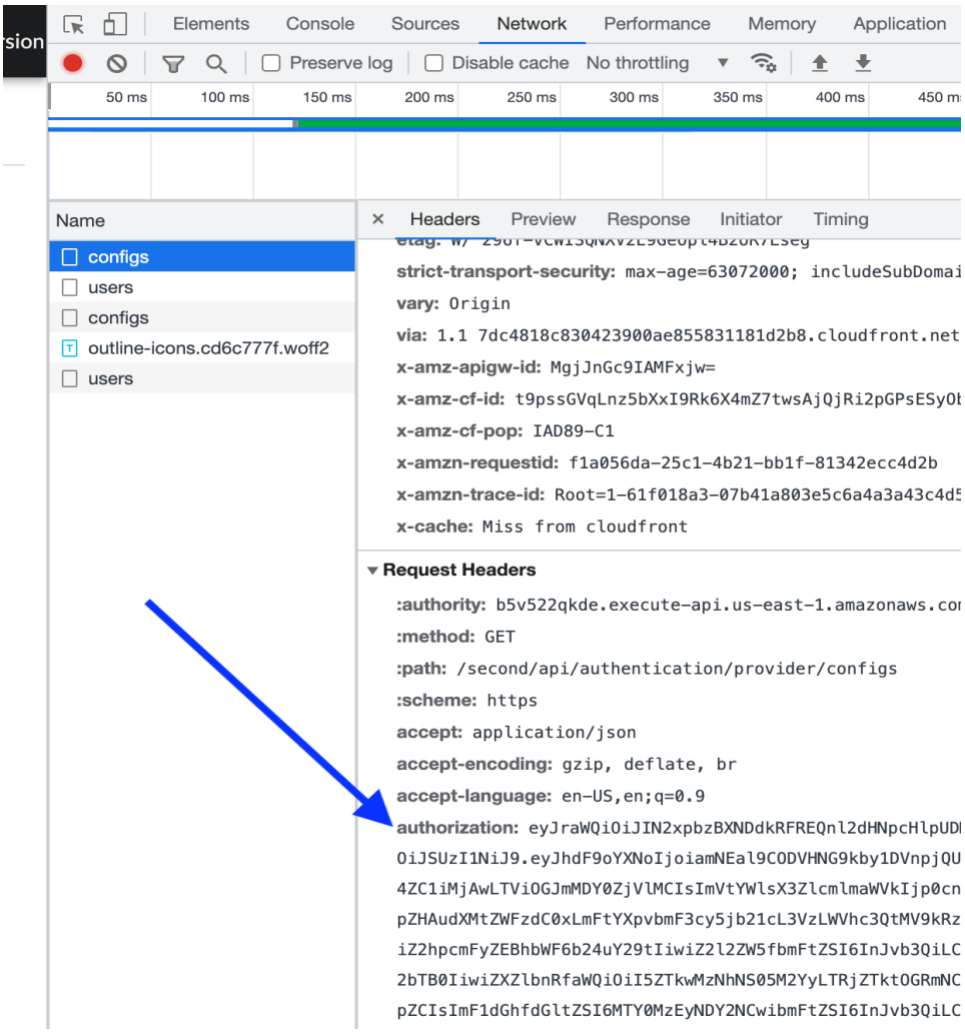
iii. Now, when you choose the **Environment quick look** icon, you should be able to view the environment's information.

7. If you have not set it up yet (only in the case of existing environments) navigate to the **Authorization** tab for the service-workbench-on-aws collection or API.
8. For **Type** drop-down, choose **Bearer Token**.
9. For **Token**, enter `{{token}}`.
10. Save this configuration.

Getting authentication token

Using Service Workbench UI

1. Log in to Service Workbench on your web browser.
2. Right click and inspect the page to display the developer console.
3. Navigate to the **Network** tab.
4. Choose another page within the Service Workbench UI.
5. You should see items populate in the **Name** column.
6. Choose one of them and then choose the **Headers** tab.
7. In the **Request Headers** section, there should be an **authorization** property. Copy that value completely. This is your authentication token.



Migrating My Study resources

We will be using API calls to migrate My Study resources.

1. If you do not know which My Study resources you need to migrate, first use the `{{baseUrl}}/api/migrate/my-studies` GET API.

    a) When using Postman, make sure you have a valid token in your environment variable. For more information on how to get this token, see Getting authentication token.

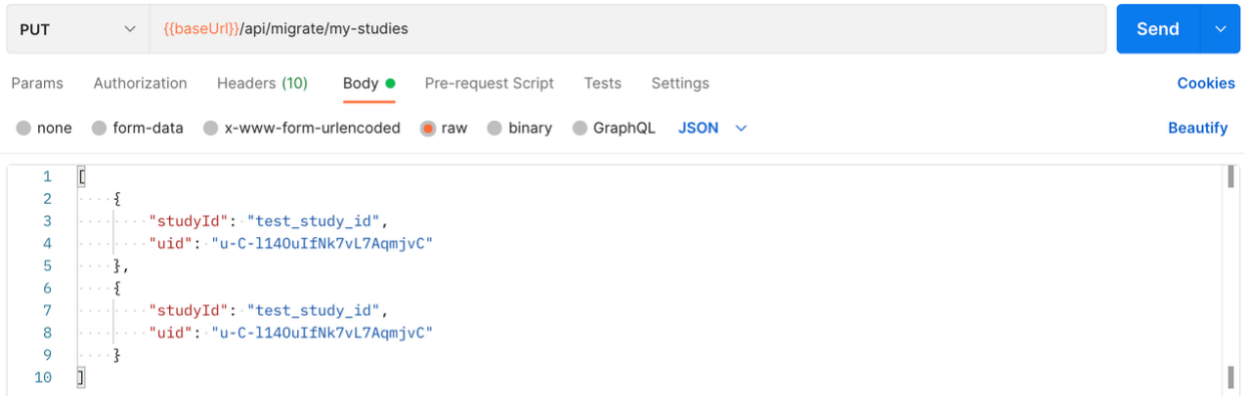
        i. In the **Collections or APIs** section, navigate to the `service-workbench-on-aws/api/migrate/my-studies` API.
        ii. Choose **GET list My Study(s) in environment**.

iii. Choose **Send**. The response will either be an empty list or a list of My Study Ids and current owner user IDs. In the former case, nothing more is needed to be done in this section as no migration of My Study resources is needed.

b) For command line, make sure you have a valid token. For more information on how to get this token, see Getting authentication token. In your terminal paste and substitute the following:

```
curl --location --request GET
'https://{{baseUrl}}/api/migrate/my-studies' \
--header 'Authorization: Bearer {{token}}'
```

2. If you have My Study resources to migrate, you can make one API call to the `{{baseUrl}}/api/migrate/my-studies` PUT API. You need to have the list of study IDs you wish to migrate and the non-internal authentication user IDs to whom you wish to migrate the ownership.
   a) For Postman, make sure you have a valid token in your environment variable For more information on how to get this token, see Getting authentication token.
   b) In the Collections or APIs section, navigate to the `service-workbench-on-aws/api/migrate/my-studies` API.
   c) Choose **PUT migrate My Study(s) to new user(s)**.
   d) In the body section, create objects with studyId and uid values.
   e) Choose **Send**.



For command line, make sure you have a valid token. For more information on how to get this token, see Getting authentication token. In your terminal paste and substitute the following:

```
curl --location --request PUT 'https://{{baseUrl}}/api/migrate/my-studies' \
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer {{token}}' \
--data-raw '[
 {
 "studyId": "test_study_id",
 "uid": "u-C-l14OuIfNk7vL7AqmjvC"
 },
 {
 "studyId": "test_study_id",
```

```
 "uid": "u-C-l14OuIfNk7vL7AqmjvC"
 }
]'
```

You can verify that all My Study resources are migrated by repeating step 1.

Post upgrade

Updating accounts

Any new hosting accounts added to Service Workbench must be updated with new permissions. For this process, you need `onboard-account.cfn.yml,` which is part of the source-code distribution.

For each hosting account:

1. Log in to the AWS Management Console of the hosting account.
2. In the CloudFormation console of that account, select the stack used for onboarding the member account (usually the stack name starts with "`initial-stack-`")
3. Choose **Update stack** and select the `addons/addon-base-raas/packages/base-raas-cfn-templates/src/templates/onboard-account.cfn.yml` file, which may also be downloaded here: onboard-account.cfn.yml. All existing parameters on that stack should still work.

Testing the operation

After the deployment script has completed without errors, log in to the Service Workbench UI and test its functionality. To display the URL and root password, run `scripts/get-info.sh ${STAGE_NAME}.`

Uninstalling Service Workbench

Follow these guidelines to delete the following for uninstalling Service Workbench:

CloudFormation stack

- For Workspaces that are running, manually delete the PVRE role additions before the stack is successfully deleted.
- Empty every bucket before deleting the stack.
- The artifacts bucket has to be deleted manually.

Products from AWS Service Catalog

- Select the portfolio, remove each product, delete the entries in groups, roles and users, and then delete the portfolio.
- Go to **Products** and delete the products from the list.

AMIs and snapshots

- Go to EC2 console, select AMIs in the left-hand menu, choose all AMIs (from SWB) and then choose **Deregister**.
- Select all snapshots and choose **Delete**.

Lambda functions

Delete Service Workbench-related Lambda functions.

CloudWatch log groups

Go to CloudWatch console, then select and delete the log groups. Alternatively, set the retention and the log groups will be deleted automatically.

AWS Cloud9 IDE

If you used AWS Cloud9 to deploy Service Workbench, you can delete this environment.

Regional code mapping

Region code mapping is defined in the file `main/config/settings/.defaults.yml.`

```
'us-east-2': 'oh'
'us-east-1': 'va'
'us-west-1': 'ca'
'us-west-2': 'or'
'ap-east-1': 'hk'
'ap-south-1': 'mum'
'ap-northeast-3': 'osa'
'ap-northeast-2': 'sel'
'ap-southeast-1': 'sg'
'ap-southeast-2': 'syd'
'ap-northeast-1': 'ty'
'ca-central-1': 'ca'
```

```
'cn-north-1': 'cn'
'cn-northwest-1': 'nx'
'eu-central-1': 'fr'
'eu-west-1': 'irl'
'eu-west-2': 'ldn'
'eu-west-3': 'par'
'eu-north-1': 'sth'
'me-south-1': 'bhr'
'sa-east-1': 'sao'
'us-gov-east-1': 'gce'
'us-gov-west-1': 'gcw'
```

Troubleshooting

Problem: Running `scripts/environment-deploy.sh $STAGE` fails with message `Uploaded file must be a non-empty zip`.

Workaround: This problem occurs because of a known issue with AWS CDK described in https://github.com/aws/aws-cdk/issues/12536. Update/downpatch `Node.js` to version 12.X.

Problem: Building machine images hangs and fails to generate any output in the terminal.

Workaround: Check that you have a default VPC and this has an internet gateway attached. If a default VPC is not available or cannot be created, then you can manually create an internet connected VPC. You will have to make a copy of `/main/solution/machine-images/config/settings/example.yml` to create a similar config file for your stage. In this file, specify the VPCId and a subnetId for your custom VPC.