

# Service Workbench installation guide

## Contents

Overview .....	3
Service Workbench architecture .....	3
Main account .....	3
Hosting account .....	3
Authentication .....	4
Storage .....	4
AWS Service Catalog .....	4
Workspace management .....	4
Cost control .....	5
Accounts, indexes, and projects .....	5
Dashboard .....	5
Workspace sizes .....	5
Service Workbench installation components .....	7
Serverless framework and projects .....	7
Continuous integration/Continuous delivery .....	8
Installing Service Workbench .....	9
Pre-installation steps .....	9
Tool requirements .....	9
Software requirements .....	11
EC2 instance requirements .....	12
Configuration settings .....	14
Accessing the Service Workbench source code .....	16
Accessing reference documentation .....	16
Service Workbench installation .....	17
Installing AMIs for EC2 Workspace .....	17
Option1: Service Workbench installation using EC2 instance .....	17
Option2: Service Workbench installation using AWS Cloud9 .....	20
Upgrading Service Workbench .....	23
Upgrading AWS solution installation .....	23
Upgrade process for command line installations .....	24
Prerequisites .....	24
Accessing the account .....	24

Downloading source code .....	24
Setting the configuration.....	24
Upgrading Service Workbench .....	25
Post upgrade.....	25
Updating accounts.....	25
Testing the operation .....	25
Uninstalling Service Workbench.....	25
Regional code mapping .....	26
Troubleshooting .....	27

## 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.

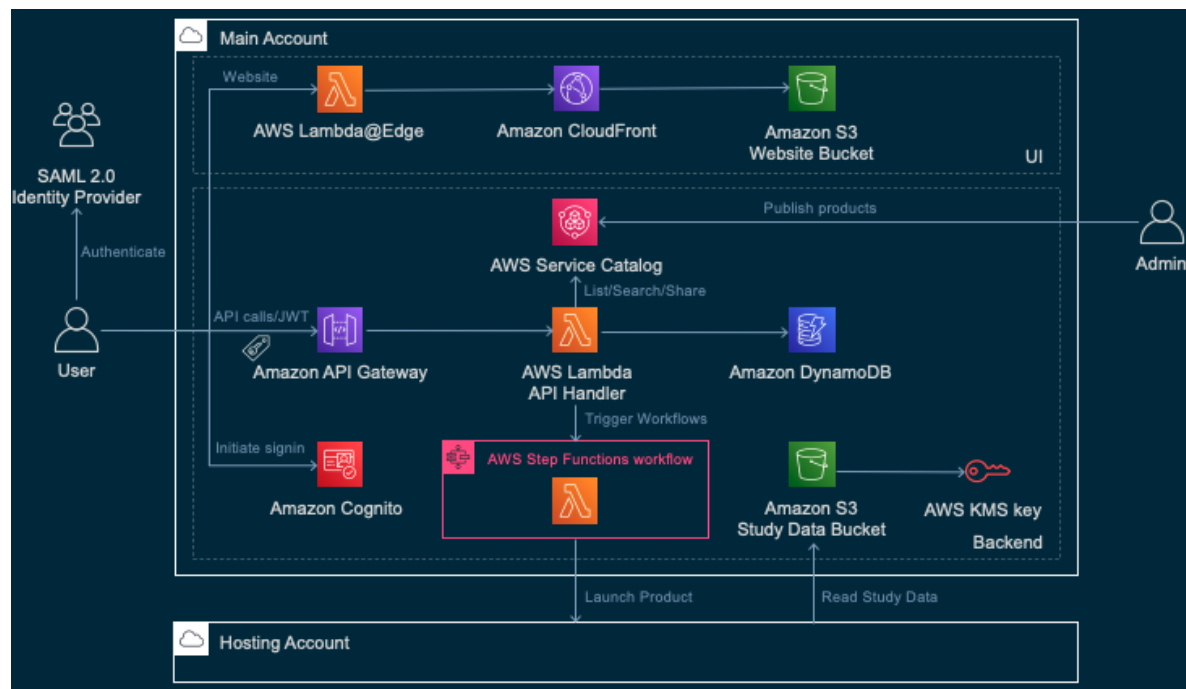
The Service Workbench installation guide covers installation of Service Workbench 3.0.0 and its related components. It provides information on Service Workbench architecture, installation, and troubleshooting.

## 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.

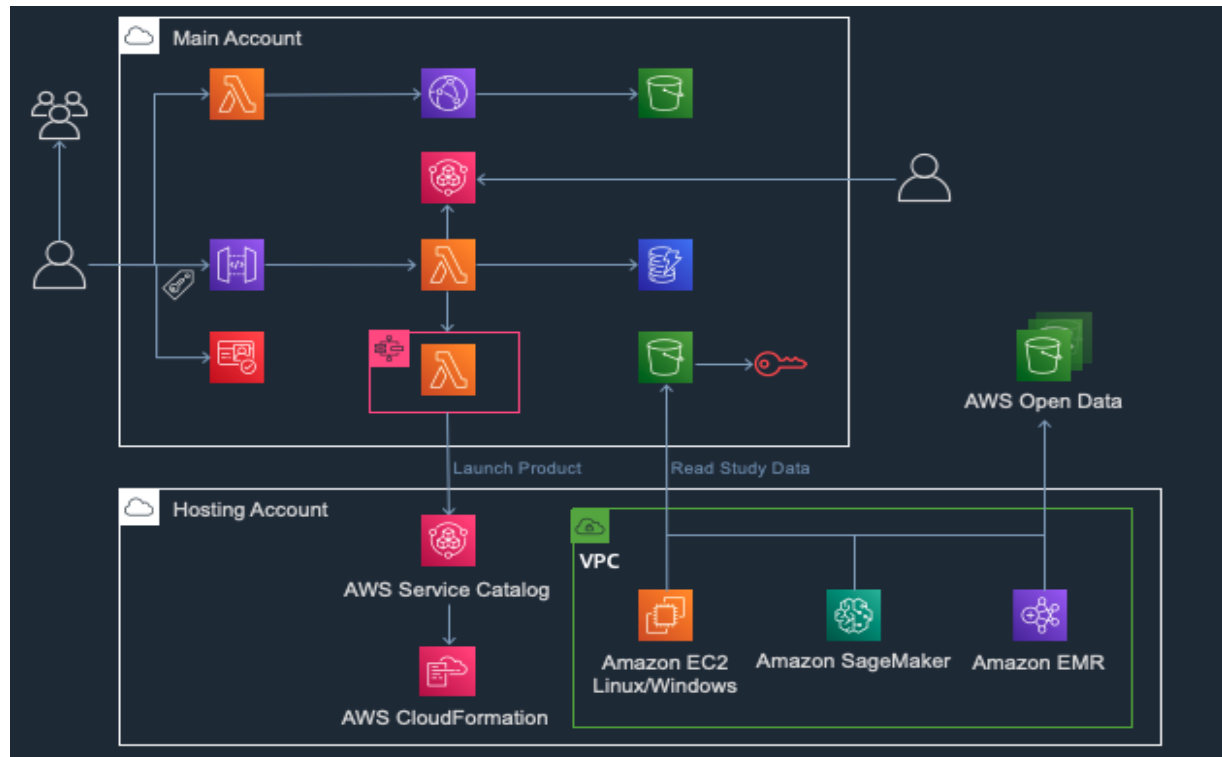
### Main account

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.

Research Workspaces
3

Create Research Workspace

All
Available
Pending
Errored
Terminated

AVAILABLE

### hcd1-test04

Created 2 weeks ago by Henner a6e3401c-fd93-4569-8bfc-96713ae81f8b

Connections
View Detail

Terminate

test

Owner	Henner
Studies	1
Project	HCD01
Restricted CIDR	205.251.233.178/32
Workspace Type	SageMaker Notebook-V1.1

\$2.82

YESTERDAY'S COST

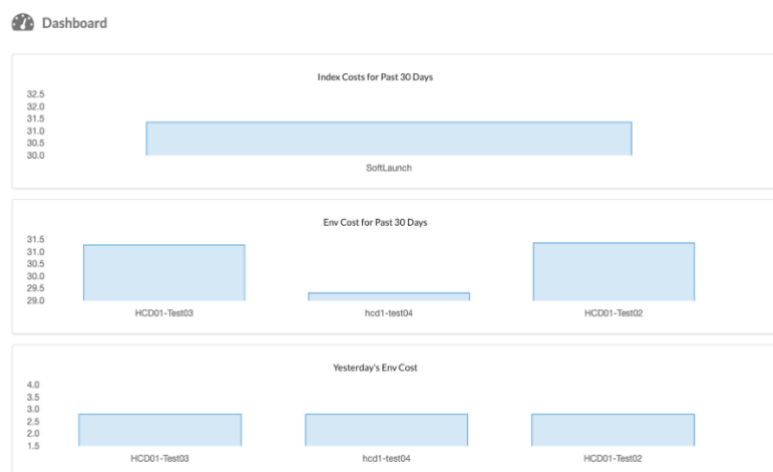
## Cost control

### 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	<code>solution/infrastructure/</code>	<p>The following AWS resources are created as part of this component deployment:</p> <ul style="list-style-type: none"><li>• S3 bucket is used for logging the following actions:<ul style="list-style-type: none"><li>○ Studying data uploads.</li><li>○ Accessing CloudFormation templates' bucket.</li><li>○ Accessing CloudFront distribution service.</li><li>○ Hosting the static Service Workbench website.</li></ul></li><li>• CloudFront distribution service to accelerate Service Workbench website access based on user location.</li></ul>
Backend	<code>solution/backend/</code>	<p>After the environment has been deployed, the backend component creates and configures the following AWS resources:</p> <p><b>S3 bucket</b></p> <ul style="list-style-type: none"><li>○ Stores uploaded study data. This bucket also receives an encryption key from <a href="#">AWS Key Management Service</a> for encrypting this data and making it available to the Service Workbench website.</li><li>○ Stores bootstrap scripts. These scripts are used to launch the Workspace instances like SageMaker, EC2, Amazon EMR.</li><li>○ Sets up IAM roles and policies for accessing Lambda functions and invoking step functions.</li></ul>
<a href="#">Amazon DynamoDB</a>	Backend SDC creates DynamoDB tables	<p>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:</p> <ul style="list-style-type: none"><li>• <b>Authentication layer handler</b> - Handles the authentication layer for API handlers.</li></ul>

		<ul style="list-style-type: none"> <li>• <b>Open data scrape handler</b> - Handles scraping the metadata from the AWS open data registry.</li> <li>• <b>API handler</b> - Provides a path for public and protected API operations.</li> <li>• <b>Workflow loop runner</b> - Invoked by AWS Step Functions.</li> </ul>
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
<a href="#">Pre-installation steps</a>	Provides information on: <ul style="list-style-type: none"><li>• creating AWS account.</li><li>• setting up AWS Cost Explorer.</li><li>• creating cost allocation tags.</li><li>• software requirements.</li><li>• accessing reference documentation.</li></ul>
<a href="#">EC2 instance requirements</a>	Provides information on creating and configuring an EC2 instance that will be used to install Service Workbench.
<a href="#">Accessing the Service Workbench source code</a>	Provides information about the Service Workbench source code location.
<a href="#">Accessing reference documentation</a>	Provides information on how to access Service Workbench documentation.
<a href="#">Service Workbench installation</a>	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.

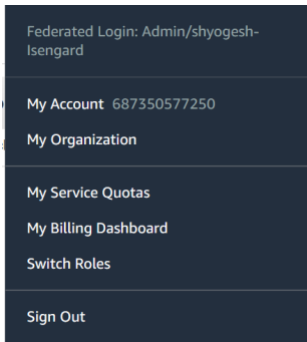
#### Enabling AWS Cost Explorer

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.

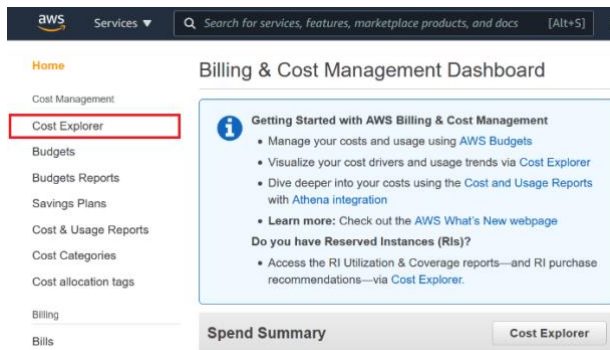
**Note:** You can enable AWS Cost Explorer even after installing Service Workbench.

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

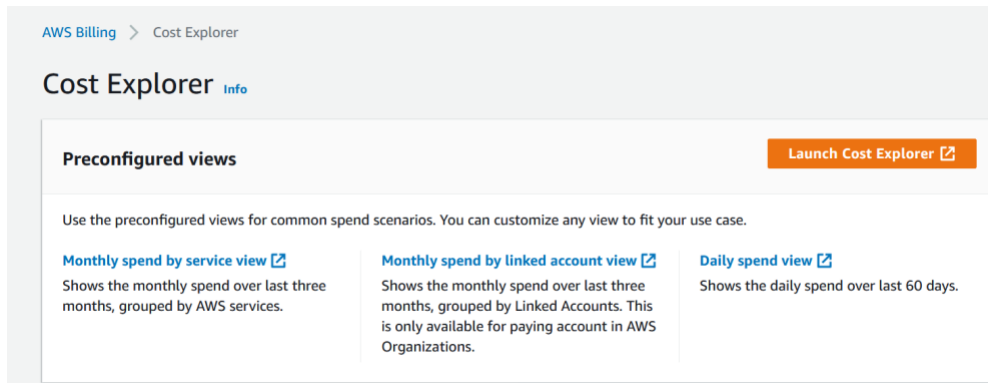
1. From the account drop-down, choose **My Billing Dashboard**.



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



3. Select **Launch Cost Explorer**.

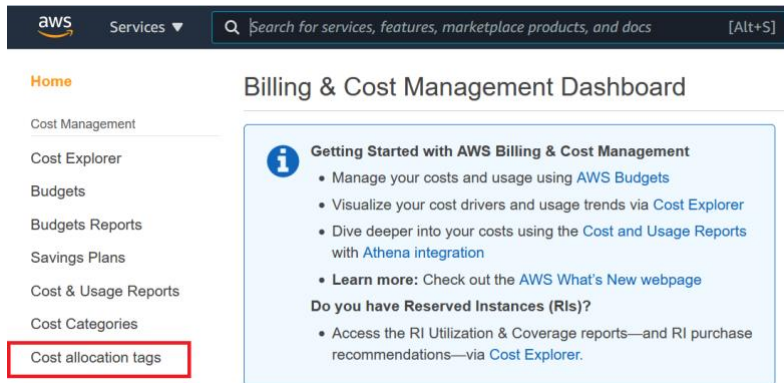


**Note:** The initialization can take up to 24 hours; however, it does not have to be completed before starting the installation process.

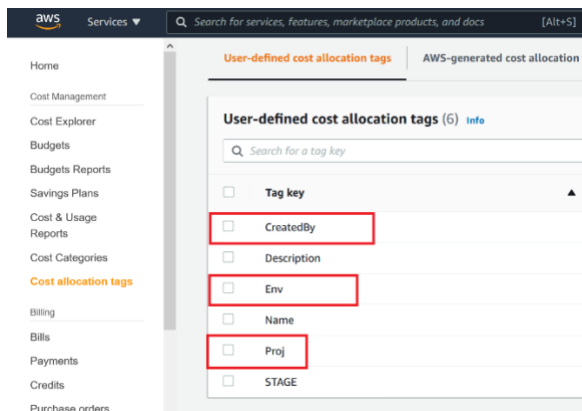
### Activating the cost allocation tags

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.
<a href="#">AWS Command Line Interface</a> (CLI)	Starts AWS services from your terminal. You must have appropriate <b>AWS programmatic credentials</b> ready. You must also have appropriate rights to deploy the platform on an AWS account.
<a href="#">Packer installation</a>	<ul style="list-style-type: none"> <li>Installs and manages JavaScript packages specified in the platform's dependencies. See <a href="#">Pnpm</a>.</li> <li>Builds JavaScript files. See <a href="#">Node.js</a>.</li> <li>Builds and packages the code for cloud deployment. See <a href="#">Serverless framework</a>. For any</li> </ul>

	issues with using node.js, refer to the <a href="#">Troubleshooting</a> section.
<a href="#">Go</a>	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
<a href="#">Selecting an EC2 instance</a>	Provides information on selecting the EC2 instance size for Service Workbench installation.
<a href="#">Configuring an EC2 instance</a>	Describes the procedure of configuring an EC2 instance, creating an IAM role, and assigning the administrator role to the EC2 instance.
<a href="#">Installing the required software on EC2 instance</a>	Describes the steps to install prerequisite software in the EC2 instance.
<a href="#">Configuration settings</a>	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

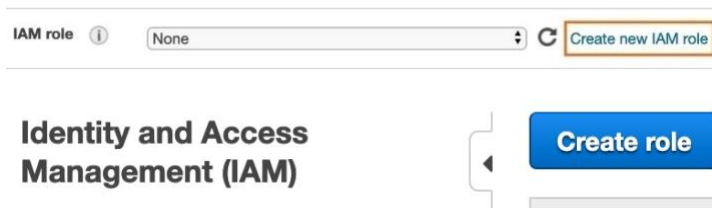
An Amazon EC2 instance can be assigned an instance profile that contains an AWS IAM role. The AWS IAM role will give the Amazon EC2 instance a set of permissions. The Amazon EC2 instance will only perform the actions defined by its AWS IAM role. Adding an AWS IAM role to the Amazon EC2 instance allows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

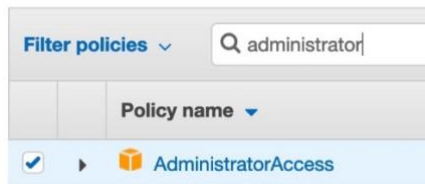
### Creating a new IAM role

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.



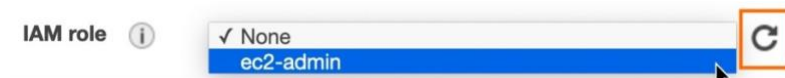
2. In **Permissions**, choose **AdministratorAccess** from the filter and proceed through **tags**.



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.

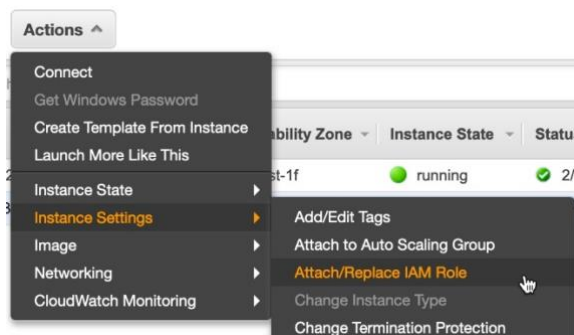


5. Create the Amazon EC2 instance.

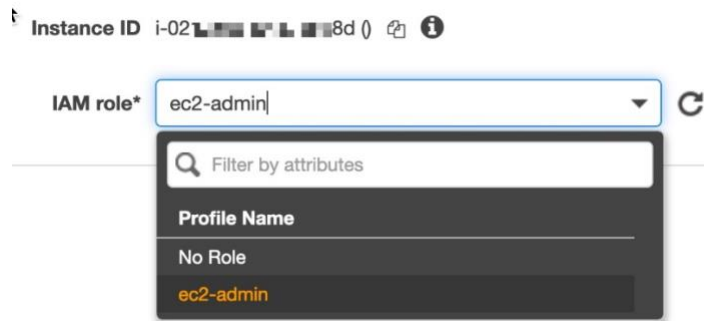
#### [Adding a role to an existing instance](#)

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

### 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
```

- b. Change to the directory containing the machine image source and build the AMIs.

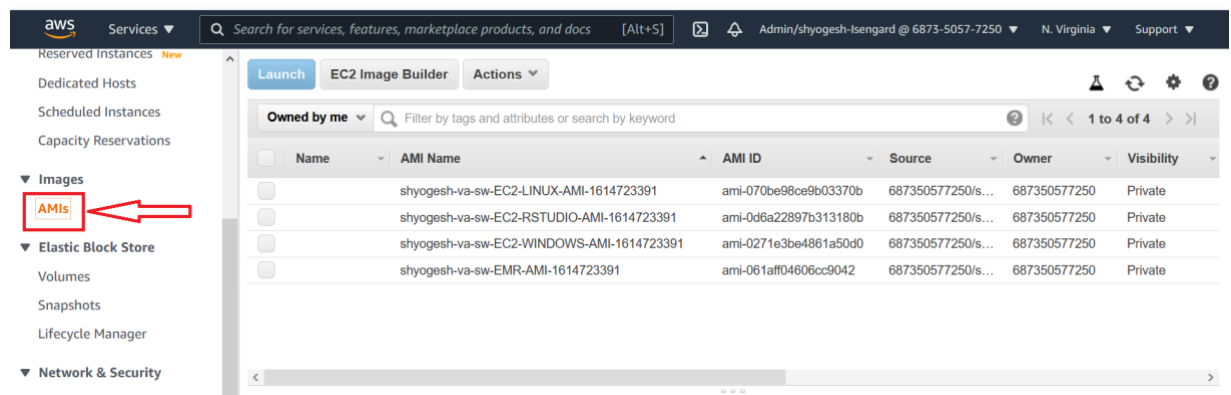
Ensure that the environment variable `STAGE_NAME` has been set before running this command.

```
# 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.	<a href="#">EC2 instance requirements</a>
Installing the node-based software.	<a href="#">Software requirements</a>
Downloading and unpacking the Service Workbench code.	<a href="#">Accessing the Service Workbench source code</a>
Choosing a stage name and edit the configuration file.	<a href="#">Configuration settings</a>
Running the deployment script, signing and adding an AWS account to Service Workbench.	<a href="#">Installing Service Workbench</a>

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

1. Go the AWS Cloud9 instance in EC2.



2. For **Actions**, choose **Modify Volume**.



3. For **Size**, enter **40**. 40 GB is the minimum suggested volume size needed for installation.
4. Choose **Modify**.
5. Choose **Yes** to accept the changes. Refresh your screen to view the modified volume size.

In the Linux prompt, type the following to view the disk space

```
df -hT
```

#### *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
```

#### **Verify the Go version**

```
go version
```

**Note:** Install everything in one directory.

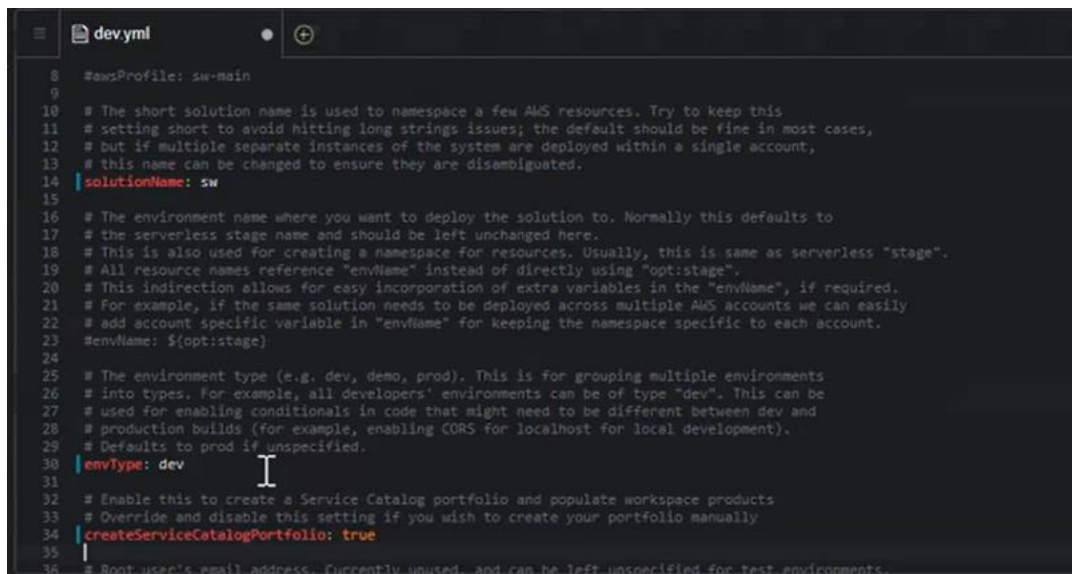
#### *Cloning the Git directory*

```
git clone https://github.com/aws-labs/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



```
8 #awsProfile: sw-main
9
10 # The short solution name is used to namespace a few AWS resources. Try to keep this
11 # setting short to avoid hitting long strings issues; the default should be fine in most cases,
12 # but if multiple separate instances of the system are deployed within a single account,
13 # this name can be changed to ensure they are disambiguated.
14 solutionName: sw
15
16 # The environment name where you want to deploy the solution to. Normally this defaults to
17 # the serverless stage name and should be left unchanged here.
18 # This is also used for creating a namespace for resources. Usually, this is same as serverless "stage".
19 # All resource names reference "envName" instead of directly using "opt:stage".
20 # This indirection allows for easy incorporation of extra variables in the "envName", if required.
21 # For example, if the same solution needs to be deployed across multiple AWS accounts we can easily
22 # add account specific variable in "envName" for keeping the namespace specific to each account.
23 #envName: ${opt:stage}
24
25 # The environment type (e.g. dev, demo, prod). This is for grouping multiple environments
26 # into types. For example, all developers' environments can be of type "dev". This can be
27 # used for enabling conditionals in code that might need to be different between dev and
28 # production builds (for example, enabling CORS for localhost for local development).
29 # Defaults to prod if unspecified.
30 envType: dev
31
32 # Enable this to create a Service Catalog portfolio and populate workspace products
33 # Override and disable this setting if you wish to create your portfolio manually
34 createServiceCatalogPortfolio: true
35
36 # Root user's email address. Currently unused, and can be left unspecified for test environments.
```

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.

## 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](#).

When upgrading Service Workbench, refer to the [CHANGELOG](#) for additional steps that might be required for the upgraded version.

### 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.

## 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.