Implementation Guide

Service Workbench on AWS

Service Workbench on AWS: Implementation Guide

Table of Contents

Solution overview

Publication date: *December 2020 (lastupdate: November 2023)*

Service Workbench on AWS enables IT teams to provide secure, repeatable, and federated control of data access, tooling, and compute power needed by researchers. IT professionals can automate the creation of baseline research environments, simplify data access, and provide price transparency. Researchers no longer need to navigate cloud infrastructure and can focus on quickly achieving research missions and completing essential work in pre-configured research environments. Both researchers and IT departments save time which they can reinvest in following cloud best practices and achieving reproducible research environments.

This implementation guide provides an overview of the Service Workbench on AWS solution, its reference architecture and components, considerations for planning the deployment, and configuration steps for deploying the Service Workbench on AWS solution to the Amazon Web Services (AWS) Cloud.

Use this navigation table to quickly find answers to these questions:

If you want to . . .	Read . . .
Know the cost for running this solution.	Cost
Understand the security considerations for this solution.	Security
Know how to plan for quotas for this solution.	Quotas
Know which AWS Regions are supported for this solution.	Supported AWS Regions
Deploy the solution.	Deploy the solution

This guide is intended for solution architects, DevOps engineers, data scientists, and cloud professionals who want to implement Service Workbench on AWS in their environment.

Features and benefits

Service Workbench on AWS provides the following features:

Simple user interface

Researchers can provision pre-approved workloads with only a few clicks.

Expense dashboard

A dashboard available on login displays current resource consumption and cost.

Federated identity management

Service Workbench leverages SAML 2.0-compliant identity providers.

Preconfigured sample research environments

Service Workbench deploys with five sample research environments: EC2 instance for Linux, EC2 instance for Windows, RStudio on EC2, SageMaker with Jupyter Notebook, and Jupyter on EMR. Administrators can use these sample environments to create other workspace types.

Service Workbench on AWS provides the following benefits:

Reduce time to science

Access research environments in minutes.

Federate access to data

Controlled access to datasets at scale.

Conduct research securely

Maintain consistent security, compliance, and governance.

Globally accessible

Collaborate with researchers around the world.

Scale and agility to grow

Virtually limitless tooling via Service Catalog

Spend controls

Cost visibility, centralized budgeting, and chargeback management.

Glossary

The following terms are specific to this document:

Data Source

Simple Storage Service (S3) resources that hold Studies. They can be within the same AWS account that deployed Service Workbench, or within other AWS Accounts.

Hosting Account

Secondary AWS accounts used to host compute environments and/or data storage.

Index

Cost-allocation structure within Service Workbench. Each AWS account used within Service Workbench must have at least one Index created.

Primary Account

The main AWS account that deployed Service Workbench.

Project

Cost-allocation structure within Service Workbench. Each Index must have at least one Project created. Provisioned Workspaces are assigned to a Project.

Study

A storage location or collection of data. Studies can be personal and only accessible to its creator, or organizational and available to multiple users. Organizational studies can have read-only or read-writer permissions with a per user granularity.

TRE

Trusted research environment. This configuration option for Service Workbench deployments provides the highest levels of data security.

Workspace

A term to describe compute resources provisioned within Service Workbench. Not to be confused with Amazon WorkSpaces.

Workspace Types

The compute options approved by administrators and made available for users to provision.

For a general reference of AWS terms, refer to the AWS glossary in the *AWS General Reference*.

Architecture overview

This section provides a reference implementation architecture diagram for the components deployed with this solution.

Service Workbench on AWS is a serverless environment deployed using an event-driven API framework. Its components are deployed across Lambda instances and static webpages using CloudFront and Amazon S3. Service Workbench on AWS relies on Service Catalog to host and manage the CloudFormation templates that define the workspaces.

Architecture diagram

Deploying this solution with the default parameters deploys the following components in your AWS account.

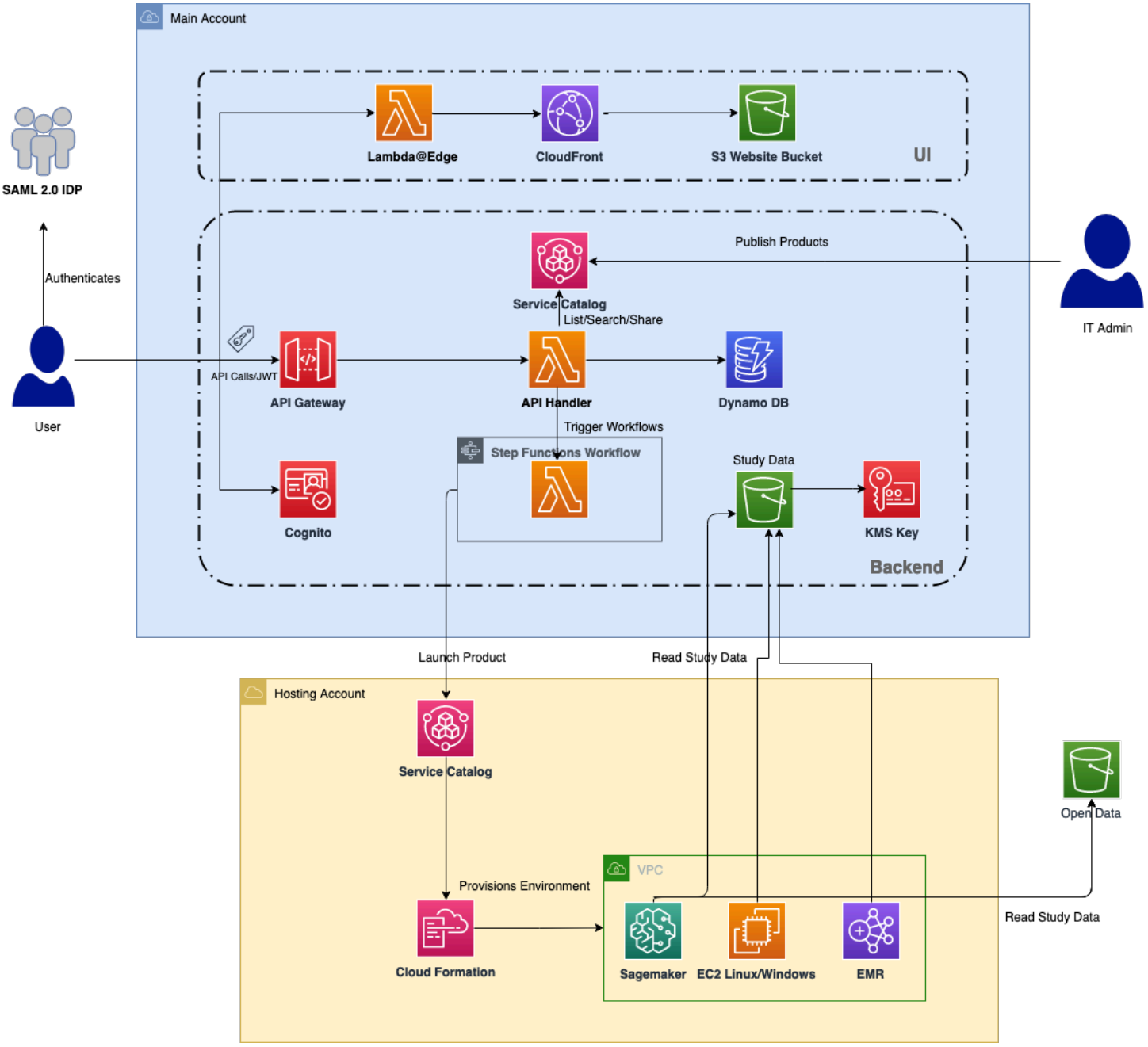

Figure 1: Service Workbench architecture on AWS

The high-level process flow for the solution components is as follows:

1. **Main Account**

   This account contains all the infrastructure needed for Service

2. **Amazon API Gateway**

Hosts the APIs called by the web front end to access the AWS Lambda functions that perform the application functions.

3. **Amazon Cognito**

   User pool stores the identities of the users creating the dashboards

4. **API Handler Lambda**

   Contains all API routes, retrieves information from AWS Service Catalog and manages interaction with Database

5. **Service Catalog**

   Contains templates and resources used for workspace creation

6. **Amazon DynamoDB**

   Persist metadata needed for service workbench such as information concerning user authentication, AWS accounts, workflows, access tokens, study data etc

7. **Step Functions Workflow AWS Lambda**

   This lambda runs continuously looking for changes happening in workspaces updating status or notifying when ready to use

8. **Study Data Amazon S3 Bucket**

   Stores Encrypted uploaded study data and cloud formation templates used on workspace creation

9. **AWS Key Management Service (KMS)**

   Manages encryption of study data

10. **Amazon Lambda@Edge**

    Adds security headers to requests made to the website

11. **AmazonCloudFront**

    Delivers low latency website content to different regions

12. **Website Amazon S3 Bucket**

    Stores the content of the Service Workbench Website such as images, styles and web pages

13. **Hosting Account**

This account is responsible for hosting all environments created in SWB

14 **Service Catalog**

Provide templates to cloud formation for SWB workspace types

15 **AWS CloudFormation**

Creates AWS infrastructure using code for installing SWB. Workspaces based on templates are provisioned by Service Catalog via CloudFormation.

16 **Machine Images**

Deploys spot instances using machine image for EC2 and EMR templates

17 **Open Data AWS S3 Bucket**

Custom Buckets not owed by SWB accounts containing read-only Open Data studies

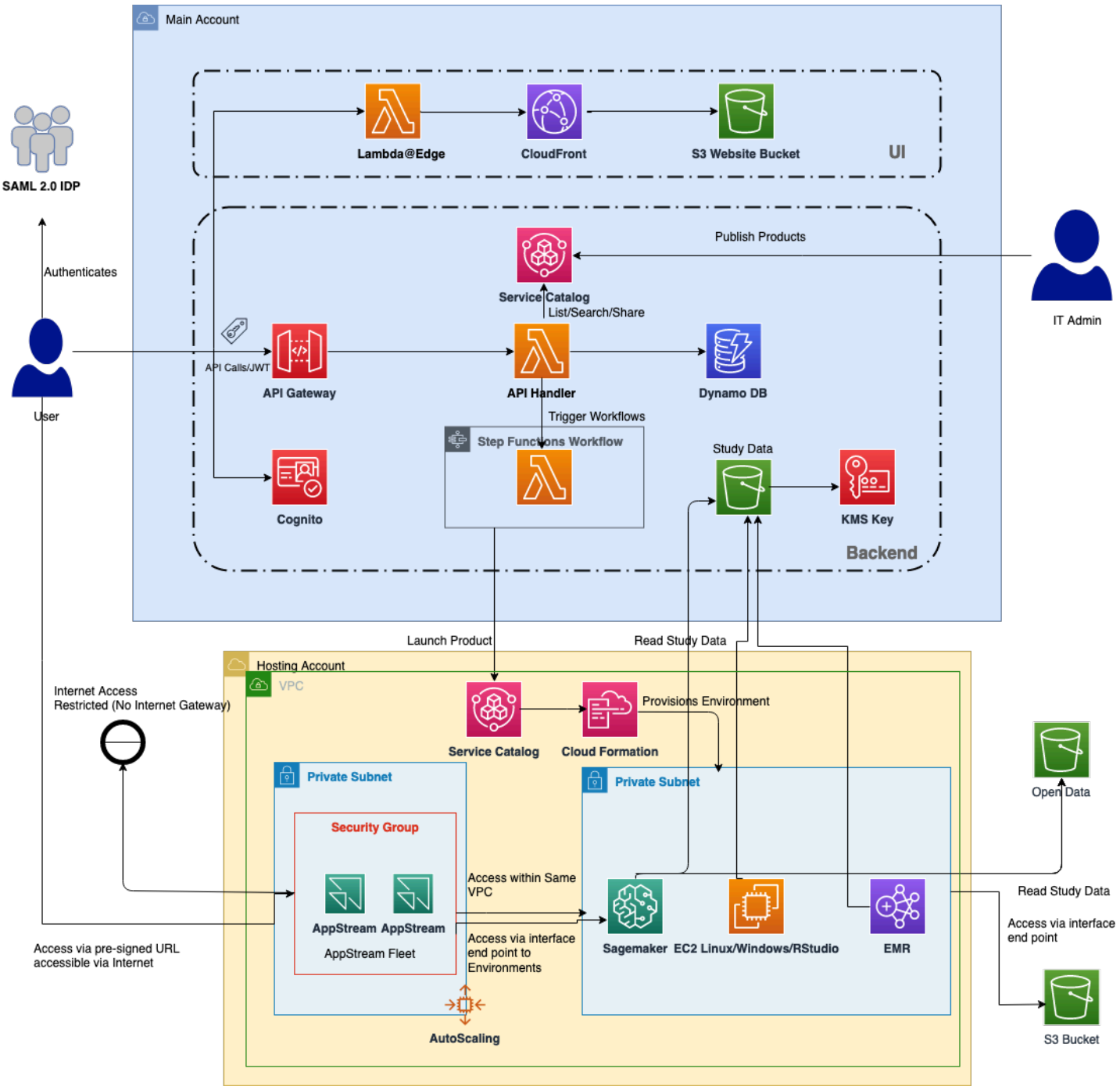

Figure 2: Service Workbench on AWS with TRE architecture

The high-level process flow for the solution components is as follows:

1. **Private Subnet**

   Restricts access to Trusted Research Environment Workspaces to be only by AppStream, not reachable via the internet

2. **Amazon AppStream**

   Allows a secure single point of access to Trusted Research Environment Workspaces

3. **Egress Bucket S3 Bucket**

   Stores Study Data from Trusted Research Environment Workspaces

AWS Well-Architected

This solution was built using the AWS Well-Architected Framework. The framework includes six pillars — operational excellence, security, reliability, performance efficiency, cost optimization, and sustainability—domain-specific lenses, hands-on labs, and the AWS Well-Architected Tool.

This section describes how the design principles and best practices of the AWS Well-Architected Framework were applied when building this solution.

Operational excellence

Service Workbench provides metrics to Amazon CloudWatch for infrastructure visibility, and all processes handled by our compute layer are logged in CloudWatch. This includes both our API servers and our backend worker processes. Code through the serverless framework manages infrastructure deployment. Code using AWS CodePipeline manages our CI/CD pipeline. As part of our software development lifecycle, GitHub actions guard our mainline branch. We write frequent, small, and reversible pull requests which are merged into our develop branch to be deployed into a test environment where we run integration tests. The code must pass all tests before we merge to mainline.

For more information, read the Operational Excellence whitepaper.

Security

- All inter-service communications use AWS IAM roles.
- All multi-account communications use AWS IAM roles.
- All roles used by the solution follow least-privilege access. It only requires minimum permissions to function properly.
- Amazon Cognito handles communication between the end user and the Amazon API Gateway by generating a Bearer token.

- To provide an extra layer of security beyond Implicit grant, an Authorization code grant activates the Cognito App Client.
- All data storage, including Amazon S3 buckets, have encryption at rest.

For more information, read the Security whitepaper.

Reliability

- To ensure high availability and recovery from service failure, the solution uses AWS Serverless Services wherever possible (examples include AWS Lambda, API Gateway, and Amazon S3).
- Data processing uses AWS Lambda functions. DynamoDB and Amazon Simple Storage Service (Amazon S3) stores data so it persists in multiple Availability Zones (AZs) by default.
- DynamoDB automatically scales the database capacity based on traffic.

For more information, read the Reliability whitepaper.

Performance efficiency

- The solution uses AWS serverless architecture throughout.
- The solution can launch in any region that supports AWS services used in this solution such as: AWS Lambda, Amazon API Gateway, Amazon S3, Amazon Cognito, and Service Catalog.

For more information, read the Performance Efficiency whitepaper.

Cost optimization

- Because the solution uses serverless architecture, you only pay for what you use.
- DynamoDB scales capacity on demand, so you only pay for the capacity you need.
- To further optimize cost, you can activate AutoStop for SageMaker workspace types. This stops SageMaker instances after a defined period of inactivity.

For more information, read the Cost Optimization whitepaper.

Sustainability

To minimize the environmental impact of the backend services, Service Workbench uses managed and serverless services. Serverless technology (such as AWS Lambda and Amazon DynamoDB) reduces the carbon footprint compared to continually operating on-premise servers. To further optimize your footprint, Researchers can schedule workspaces instances to start and stop according to their work schedule.

For more information, read the Sustainability whitepaper.

Architecture details

This section describes the components and AWS services that make up this solution and the architecture details on how these components work together.

User web portal

Service Workbench on AWS includes a web portal used by both administrators and research users. The web portal is deployed into the main account using Amazon CloudFront and a static website hosted in an Amazon S3 website bucket. The research user web portal uses Amazon DynamoDB to store metadata about users, available datasets, and compute workspaces.

Research studies S3 bucket

Service Workbench on AWS allows users to create and share research studies. Studies are hosted in an internal Amazon S3 bucket created during installation. Research users can use the web portal to create and manage studies as well as upload data.

AWS open data integration

Service Workbench on AWS uses API calls to access AWS Open Data. From the web portal, researchers can select data sets to use in their compute environments.

CI/CD pipeline

Service Workbench on AWS includes a CI/CD pipeline to install the application in your account. The pipeline uses an AWS CodeBuild project created during deployment using the CloudFormation template.

AWS services in this solution

AWS service	Description
Lambda@Edge **Core**	Contains an inline JavaScript interceptor function that adds security headers to the Cloudfront output response

AWS service	Description
Amazon CloudFront **Core**	Allows low latency when loading content from different regions in SWB website
Amazon API Gateway **Core**	Serves as a single point of access for SWB API enabling security measures like rate limiting
Amazon Cognito **Core**	Manages Identities of SWB users in user pools
AWS Service Catalog **Core**	Stores Cloud Formation Templates used in workspace creation
AWS Lambda **Core**	Contains all SWB APIs and functions used to communicate between DynamoDB and Service Catalog
Amazon DynamoDB **Core**	Persist metadata needed for service workbench such as information concerning user authentication, AWS accounts, workflows, access tokens, study data etc
AWS KMS **Core**	Manages encryption of Study Data in S3 Buckets
Amazon AppStream 2.0 **Core**	Manages access to TRE Workspaces

Plan your deployment

Service Workbench on AWS requires at least one AWS account, but can be used in a multi-account environment. In a multi-account environment, the accounts can either be part of an organization or two or more independent accounts. The account which deploys Service Workbench will be designated as the primary account. Other accounts, referred to as hosting accounts, host compute workloads or share data.

Service Workbench installs to a single region. If using multiple accounts, they must use the same AWS region. Multi-region deployments are not supported at this time. However, though the accounts must use the same Region, the Availability Zone in each hosting account may be different. Each hosting account supports one subnet in a single Availability Zone.

With Amazon Route53 for Domain Name Services (DNS) and AWS Certificate Manager to serve SSL/TLS certificates, Service Workbench can be configured to use custom domain names and certificates.

Service Workbench creates custom encryption keys using AWS Key Management Service (KMS) during deployment. Other custom KMS keys can be used for data storage shared with Service Workbench, but SWB must be given permission to use these keys.

Cost

> **Note**
>
> You are responsible for the cost of the AWS services used while running this solution. As of November 2023, the cost for running this solution with the default settings in the US East (N. Virginia) is approximately **$9.13 a month** (not including Workspaces and Study data). Refer to the pricing webpage for each AWS service used in this solution.

We recommend creating a budget through AWS Cost Explorer to help manage costs. Prices are subject to change. For full details, refer to the pricing webpage for each AWS service used in this solution.

Sample cost table

The following table provides a sample cost breakdown for deploying this solution with the default parameters in the US East (N. Virginia) Region for one month.

AWS service	Dimensions	Cost [$]
Amazon API Gateway	1,000,000 REST API calls per month	$ 3.50
Step Functions	100,000 transitions	$ 2.50
AWS Lambda	2,000,000 requests with 200 ms duration	Free
Amazon DynamoDB	2,000,000 reads, 500,000 writes	$ 1.13
KMS keys	2 customer keys	$ 2.00

Sample cost table with study data

The following table provides a sample cost breakdown for deploying this solution with the default parameters in the US East (N. Virginia) Region for one month.

AWS service	Dimensions	Cost [$]
Amazon API Gateway	1,000,000 REST API calls per month	$ 3.50
Step Functions	100,000 transitions	$ 2.50
AWS Lambda	2,000,000 requests with 200 ms duration	Free
Amazon DynamoDB	2,000,000 reads, 500,000 writes	$ 1.13

AWS service	Dimensions	Cost [$]
KMS keys	2 customer keys	$ 2.00
Amazon S3 – storage	2 TB	$47.00
Amazon S3 – requests	1,000,000 requests	$ 5.00

Example: 2,000 compute hours per month

AWS service	Dimensions	Cost [$]
Amazon EC2	2,000 compute hours per month	$ 384
Amazon SageMaker	2,000 compute hours per month	$ 1,614
Amazon EMR	2,000 compute hours per month	$ 480

Security

When you build systems on AWS infrastructure, security responsibilities are shared between you and AWS. This shared model reduces your operational burden because AWS operates, manages, and controls the components including the host operating system, the virtualization layer, and the physical security of the facilities in which the services operate. For more information about AWS security, visit AWS Cloud Security.

IAM roles

AWS Identity and Access Management (IAM) roles allow customers to assign granular access policies and permissions to services and users on the AWS Cloud. This solution creates IAM roles that grant the solution's AWS Lambda functions access to create Regional resources. IAM roles created by Service Workbench on AWS are named with the `SolutionName` and `StageName` specified during the deployment.

Amazon CloudFront

This solution deploys a web console hosted in an Amazon Simple Storage Service (Amazon S3) bucket. To help reduce latency and improve security, this solution includes an Amazon CloudFront distribution with an origin access identity, which is a CloudFront user that provides public access to the solution's static website content in the Amazon S3 bucket. For more information, refer to Restricting Access to Amazon S3 Content by Using an Origin Access Identity in the *Amazon CloudFront Developer Guide*.

Amazon Virtual Private Cloud

The Amazon Virtual Private Cloud (VPCs) in this solution are designed to control and isolate network traffic into research compute environments. We recommend that you review the configuration and further restrict access as needed once the solution is deployed.

Amazon CloudWatch

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

Access logs example format:

```
{
"authorizer.principalId":
"u-000000000000",
"error.message": "-",
"extendedRequestId": "ZuT4rGDNoAMFxXw=",
"httpMethod": "GET",
"identity.sourceIp": "22.22.222.22",
"integration.error": "-",
"integration.integrationStatus": "200",
"integration.latency": "79",
"integration.requestId":
"67394741-90ae-4c6c-94fb-df8bf7be33ec",
"integration.status": "200",
"path": "/dev/api/user-roles",
"requestId":
"468a1b4d-3015-4901-b749-37e4e0551029",
"responseLatency": "83",
"responseLength": "819",
"stage": "dev",
```

```
"status": "200"
}
```

Metrics

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

- Working with AWS Lambda function metrics – AWS Lambda
- Amazon API Gateway dimensions and metrics – Amazon API Gateway

Service Workbench on AWS does not produce any custom metrics.

Security groups

The security groups created in this solution are designed to control and isolate network traffic between the Lambda functions, certificate signing request (CSR) instances, and remote VPN endpoints. We recommend that you review the security groups and further restrict access as needed once the deployment is up and running.

Supported AWS Regions

This solution uses the AWS CodeBuild, Amazon CloudFront, and AWS Service Catalog services, which are not currently available in all AWS Regions. You must launch this solution in an AWS Region where these services are available. For the most current availability of AWS services by Region, refer to the AWS Regional Services List.

Service Workbench on AWS is supported in the following AWS Regions:

Region name	
US East (Ohio)	Canada (Central)
US East (N. Virginia)	Europe (Frankfurt)
US West (N. California)	Europe (Ireland)
US West (Oregon)	Europe (London)

Region name	
Asia Pacific (Mumbai)	Europe (Milan)
Asia Pacific (Seoul)	Europe (Paris)
Asia Pacific (Singapore)	Europe (Stockholm)
Asia Pacific (Sydney)	Middle East (Bahrain)
Asia Pacific (Tokyo)	South America (São Paulo)

Quotas

Service quotas, also referred to as limits, are the maximum number of service resources or operations for your AWS account.

Quotas for AWS services in this solution

Make sure you have sufficient quota for each of the services implemented in this solution. For more information, see AWS service quotas.

Click one of the following links to go to the page for that service. To view the service quotas for all AWS services in the documentation without switching pages, view the information in the Service endpoints and quotas page in the PDF instead.

AWS CloudFormation quotas

Your AWS account has AWS CloudFormation quotas that you should be aware of when launching the stack in this solution. By understanding these quotas, you can avoid limitation errors that would prevent you from deploying this solution successfully. For more information, refer to AWS CloudFormation quotas in the in the *AWS CloudFormation User's Guide*.

Deploy the solution

This solution uses AWS CloudFormation templates and stacks to automate its deployment. The CloudFormation template(s) to describes the AWS resources included in this solution and their properties. The CloudFormation stack provisions the resources that are described in the template(s).

Deployment process overview

Before you launch the solution, review the cost, architecture, security, and other considerations discussed in this guide. Follow the step-by-step instructions in this section to configure and deploy the solution into your account.

Time to deploy: Approximately 30-45 minutes

> **Note**
>
> If you have previously deployed this solution, refer to Update the solution for update instructions.

AWS Cloud9 Deployment	Amazon EC2 Deployment
Step 1. Create the AWS Cloud9 Instance	Step 1. Create an EC2 instance
Step 2. Modify the volume	Step 2. Connect to the EC2 instance
Step 3. Configure the AWS Cloud9 environment	Step 3. Install Node and Go
Step 4. Clone the GitHub directory	Step 4. Clone the GitHub directory
Step 5. Prepare the environment file	Step 5. Prepare the environment file
Step 6. Run the install script	Step 6. Run the install script

Choose your deployment option

AWS Cloud9 offers a straight-forward deployment process to get users started more quickly:

Deploy using AWS Cloud9

If you are already using EC2 instances, this method may be best for your use case:

Deploy using an EC2 instance

Deploy using AWS Cloud9

Follow the step-by-step instructions in this section to configure and deploy the solution into your account.

Time to deploy: Approximately 30 minutes

Step 1: Create the AWS Cloud9 Instance

1. Go to the AWS Cloud9 product page and sign in.
2. Choose **Create environment**.
3. Enter a name and description for the AWS Cloud9 environment.
4. Choose **Next step**.
5. Configure your Environment settings:

    a. Environment type—Create a new EC2 instance for environment (direct access)
    b. Instance type—m5.large
    c. Platform—Amazon Linux 2

    > **Note**
    >
    > Keep all other settings in the default selections.

6. Choose **Next step**.
7. Review your environment name and settings and click **Create environment**.

Your environment should load a welcome screen and command line interface.

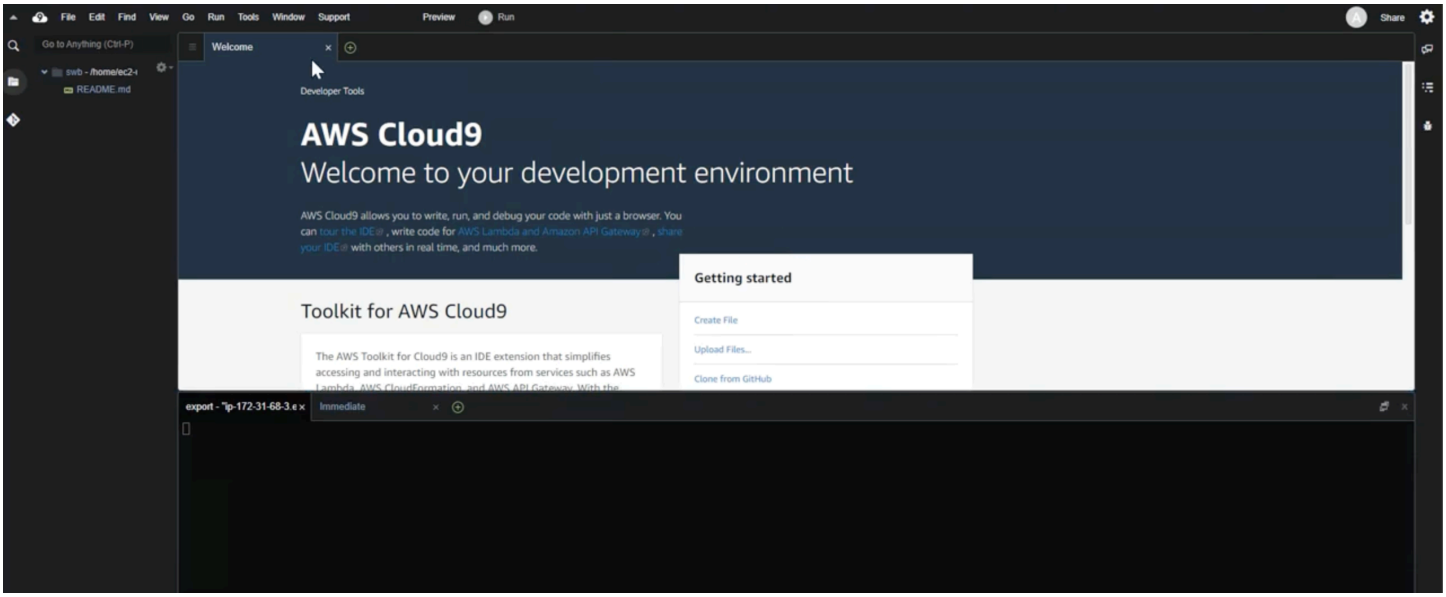

Figure 1: Welcome screen

Step 2: Modify the volume

1. In the **Cloud9** menu, choose **Go To Your Dashboard**.

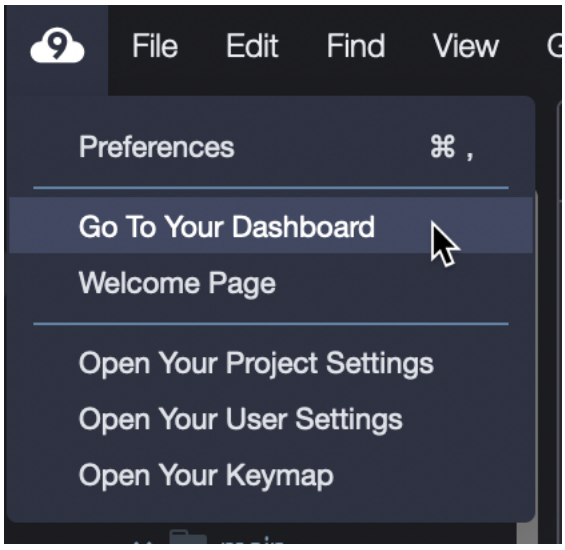

Figure 2: Cloud9 menu

2. Use the **Services** menu to go to **EC2**.
3. From the left navigation, choose **Volumes**.
4. Select the volume associated with your AWS Cloud9 instance.
5. From the **Actions** menu, choose **Modify Volume**.

6. Increase the size to at least 40 GB.
7. Choose **Modify**, and then choose **Modify** again to confirm.

For more information about modifying the volume, see Resize an Amazon EBS volume used by an environment.

Step 3: Configure the AWS Cloud9 environment

Return to the AWS Cloud9 command line environment to complete the following steps.

> **Note**
>
> Keep all other settings in the default selection.

1. Verify the file size by entering the command:

```
df -hT
```

2. To increase the partition size, enter:

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

3. Increase the file system inside the partition:

```
sudo xfs_growfs -d /
```

4. Verify that the file size increased by entering:

```
df -hT
```

5. Check the node version by entering:

```
node --version
```

6. Install the node package manager:

```
npm install -g serverless pnpm@latest-8
```

7. A serverless packer is necessary to build AMIs. Follow the README instructions to install. Open a new terminal window after you complete the installation.

Step 4: Clone the GitHub directory

1. From the terminal, clone the GitHub directory:

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

   The Git directory is now available on your instance.

2. Ensure you are working from the `service-workbench-on-aws` folder for the remainder of the process:

   ```
   cd service-workbench-on-aws/
   ```

Step 5: Prepare the environment file

1. From the toggletree, navigate to select `example.yml`:

   `main` > `config` > `settings` > `example.yml`

2. Copy the `example.yml` file and rename it.

   Example: *`dev.yml`*

3. In the copied file, uncomment the following:

   a. `awsRegion: us-east-1`

   b. `solutionName: sw`

   c. `envType: dev`

   d. `createServiceCatalogPortfolio: true`

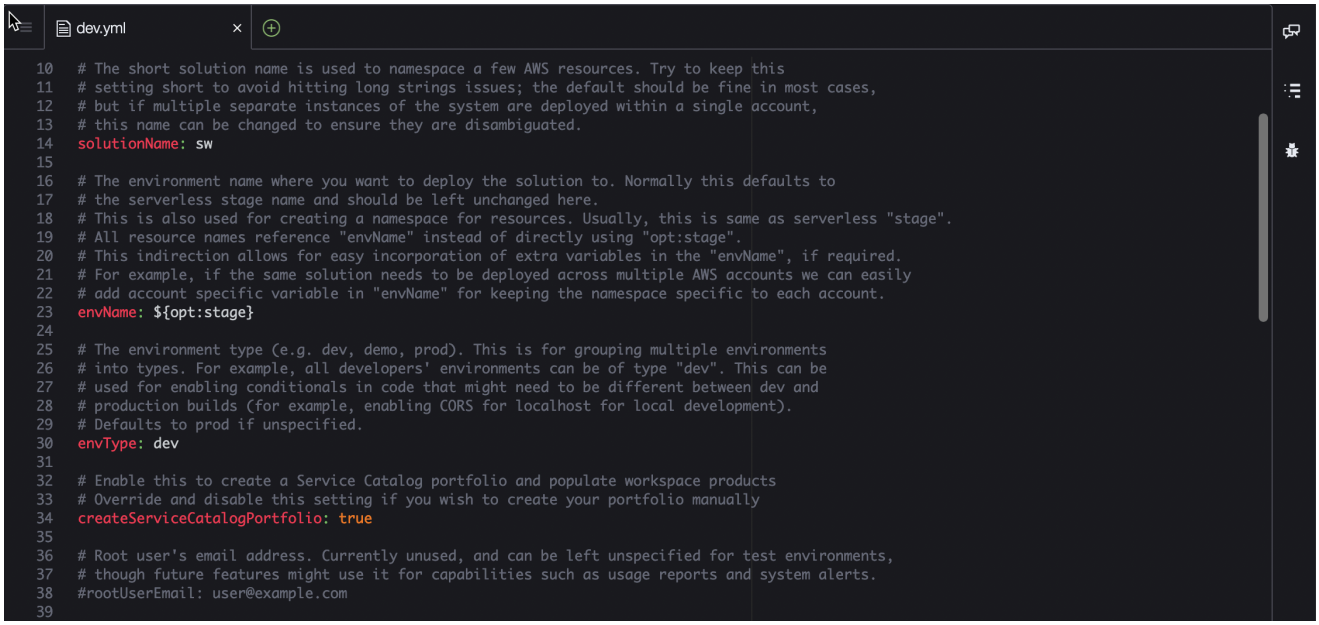

Figure 3: dev.yml file

4. Save the copied file.

Step 6: Run the install script

1. Ensure you are working from the service-workbench-on-aws folder:

```
cd service-workbench-on-aws/
```

2. Deploy the install script:

```
scripts/environment-deploy.sh [copied file name]
```

Example:

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

The install may take up to an hour. Once successfully installed you will receive a success message and login details.

Figure 4 : Environment deployed successfully

For more information about logging in to Service Workbench, see Login and setup user account.

Deploy using an EC2 instance

Follow the step-by-step instructions in this section to configure and deploy the solution into your account.

Time to deploy: Approximately 45 minutes

Step 1: Create an EC2 instance

If you do not already have an available EC2 instance, follow these steps to create one:

1. From the EC2 console, choose **Launch instances**.
2. Choose **Select** for Amazon Linux 2 AMI (HVM) – Kernel 5.10, SSD Volume Type.
3. Select an instance of **t2.medium** or larger.
4. Choose **Next: Configure Instance Details**.
5. Select an IAM role with administrative permissions.

   If you do not have an available IAM, choose **Create new IAM role**. Once you have created a role, return to **Configure Instance Details** and choose refresh next to IAM role. The role should now be available.

> **Note**
>
> For more information on creating IAM admin roles, see Creating your first IAM admin user and user group.

6. Choose **Next: Add Storage**.
7. Change the size to at least 40 GB.
8. Choose **Review and Launch**.
9. Review the details of your instance and choose **Launch**.
10. Select an existing key pair or create a new key pair for your instance.
11. Choose **Launch Instances**.

Step 2: Connect to the EC2 instance

1. From the EC2 console, select the instance that will host Service Workbench. Choose **Connect**.
2. Select your connection method, and follow the onscreen directions.

Step 3: Install Node and Go

Install Node.js

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.35.3/install.sh | bash
source ~/.bashrc
nvm install 18
npm install -g serverless pnpm@latest-8 hygen
```

Install Go

```
cd ~ wget -c https://golang.org/dl/go1.15.2.linux-amd64.tar.gz
sudo tar -C /usr/local -xvzf go1.15.2.linux-amd64.tar.gz
echo "PATH=$PATH:/usr/local/go/bin" >> ~/.bashrc
source ~/.bashrc
go version
```

Step 4: Clone the GitHub directory

1. From terminal, install git:

```
sudo yum install -y git
```

2. Clone the git repository:

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

Step 5: Prepare the environment file

1. Open the Service Workbench folder:

```
cd service-workbench-on-aws/
```

2. Navigate to the main configuration directory:

```
cd main/config/settings
```

3. Make a copy of the example.yml file and rename it. For this example, we will rename the file to dev.yml.

```
 cp example.yml dev.yml
```

4. Open the newly created configuration file to edit:

```
vim dev
```

5. Uncomment and set the following values:

   a. **awsRegion**

      Ensure you use the same AWS Region where you use the AWS Management Console.

   b. **solutionName**

6. To quit editing, press ESC and then enter:

```
:wq
```

Step 6: Run the install script

Return to the root directory before launching. If you changed the copied configuration file name to a name other than dev, use that in the place of dev.

1. Run the installation script:

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

2. Note the URL and password provided. You can display the URL and password again with:

```
scripts/get-info.sh dev
```

Login and setup user account

1. Copy and paste the provided URL in a browser.
2. Choose **Login**.

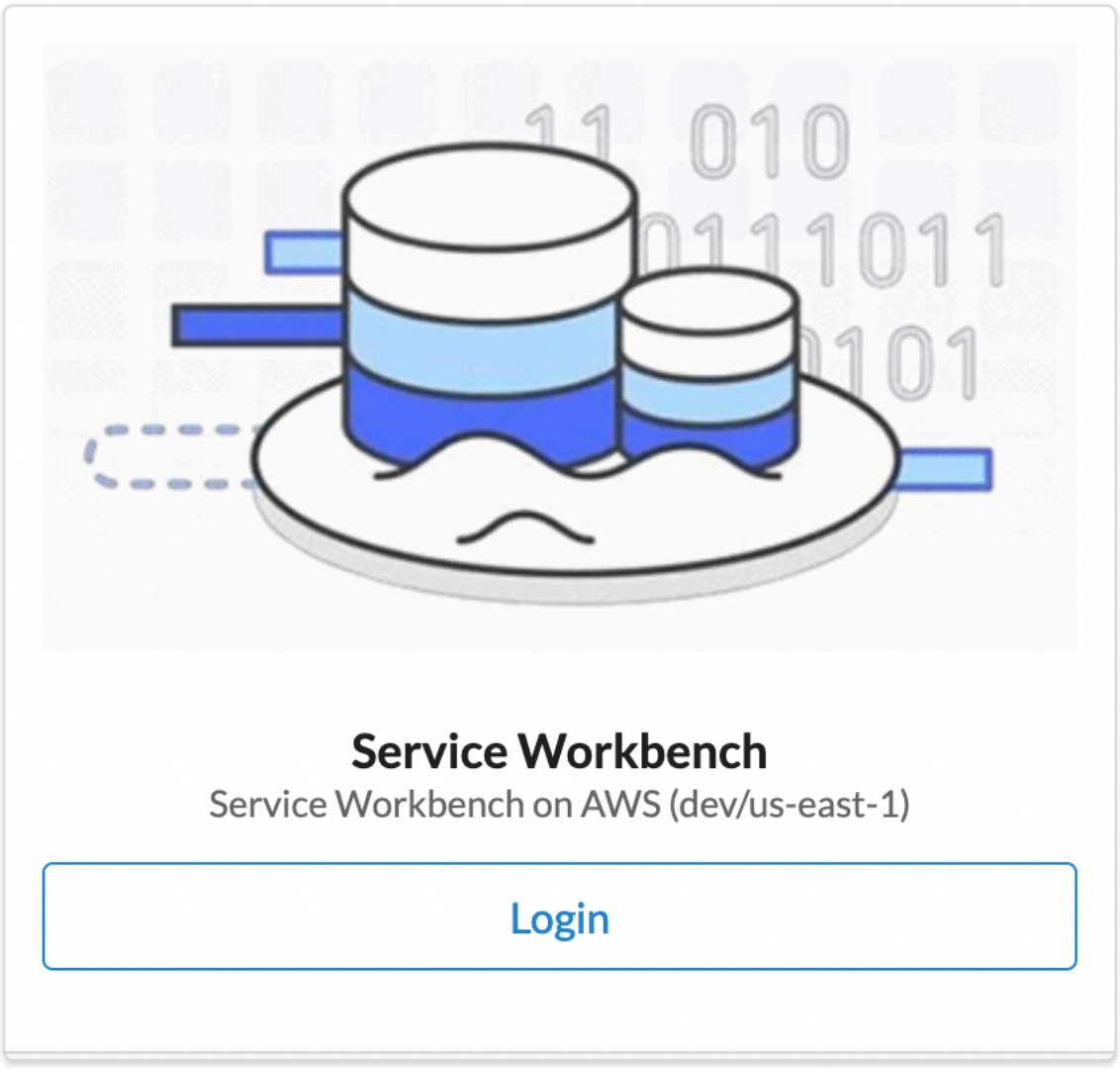

Figure 5 : Service Workbench on AWS login screen

3. Your username will be `<root@example.com>` and the password will be the one provided at the end of the install.

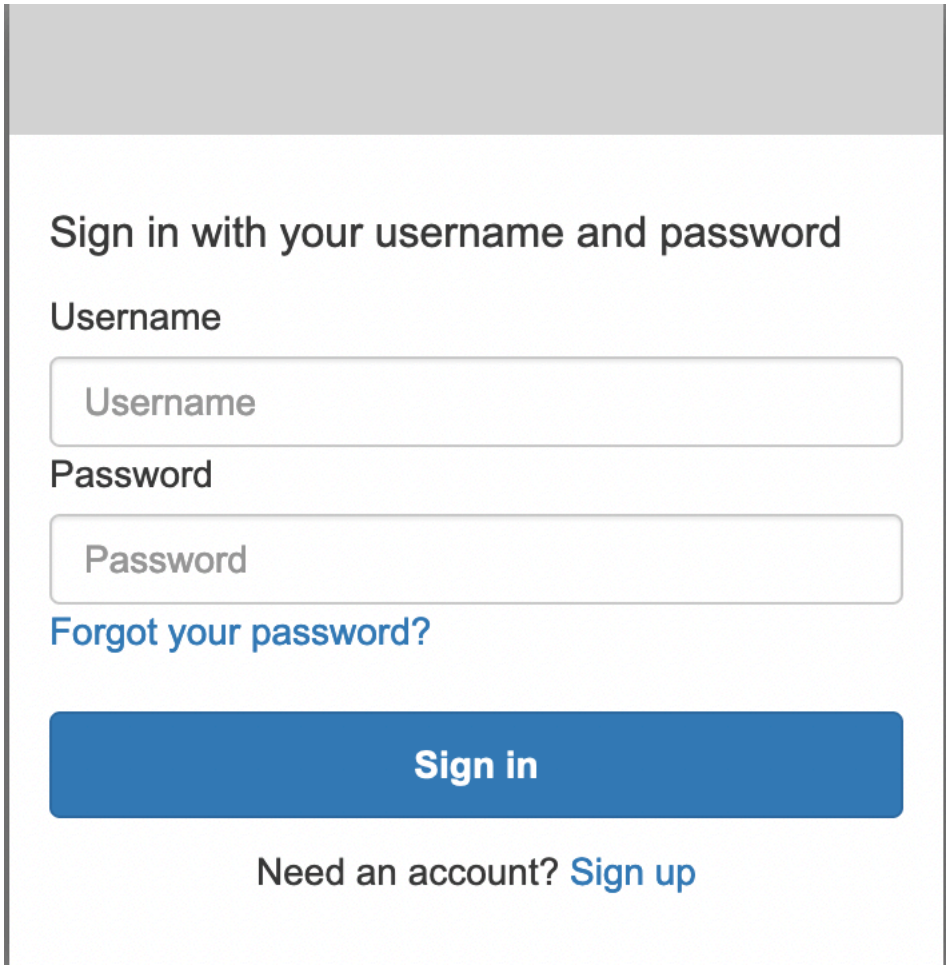

Figure 6: Service Workbench on AWS account sign in screen

4. Change your password when prompted.

Update the solution

If you have previously deployed the solution, you can update Service Workbench on AWS to get the latest version of the solution's framework by:

1. Pull the latest version from GitHub.

   > **Note**
   >
   > If you pull the latest version to a forked branch, you will need to resolve any conflicts.

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

   The application automatically calculates and updates the necessary CloudFormation stacks.

   > **Note**
   >
   > This command does not update the main/solutions/machine-images stack. Please follow instructions in that package's README.md file to create AMIs if recent updates are pulled in this package.

README files:

- Backend
- Infra
- Post-deployment

Uninstall the solution

You can uninstall the Service Workbench on AWS solution by using the AWS Command Line Interface. You must manually delete the AMIs and the Cloud9 IDE created by this solution.

Using AWS Command Line Interface

Determine whether the AWS Command Line Interface (AWS CLI) is available in your environment. For installation instructions, refer to What Is the AWS Command Line Interface in the *AWS CLI User Guide*. After confirming that the AWS CLI is available, run the following command:

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

This command deletes all Service Catalog resources and CloudFormation stacks along with resources created during Service Workbench on AWS installation. Any resources that were not deleted by the `environment-delete` script will be listed at the end of the script run.

Deleting AMIs and snapshots

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

Deleting Cloud9 IDE

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

Source code

Visit our GitHub repository to download the source files for this solution and to share your customizations with others.

Reference

This section includes pointers to related resources, and a list of builders who contributed to this solution.

Related resources

- Amazon API Gateway
- Amazon CloudFront
- Amazon Cognito
- Amazon DynamoDB
- Amazon EC2
- Amazon EMR
- Amazon S3
- Amazon SageMaker
- AWS CodeBuild
- AWS CloudFormation
- AWS Identity and Access Management
- AWS Lambda and Lambda@Edge
- Service Catalog
- AWS Step Functions
- AWS Systems Manager
- Open Data on AWS
- AWS Key Management Service

Contributors

- Karl Camp
- Maggie Krummel
- Neel Sethia
- Robert Smayda
- Simon Woldemichael
- Yanyu Zheng
- Balaji Raman
- Tim Nguyen
- Sanket Dharwadkar
- Marianna Ghirardelli
- Kevin Park

- Travis Berkley
- Fernando Aranda Carrillo
- Greg Grieff
- Brandi Hopkins

Revisions

For more information, refer to the CHANGELOG.md file in the GitHub repository.

Date	Change
December 2020	Initial release
January 2021	Release version 1.4.4 - For more information, refer to the CHANGELOG.md file in the GitHub repository.
April 2021	Documentation updates
March 2023	Updated implementation information and included manual installation information. Updated Updating and Uninstalling sections.
November 2023	Support for Node 18 added.

Notices

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

Service Workbench on AWS is licensed under the terms of the of the Apache License Version 2.0 available at The Apache Software Foundation.