Service Workbench Quick Start Guide

June, 2021

1. Prerequisites

Before getting started with the installation of Service Workbench on AWS, you should ensure that you have the requisite expertise, and have completed the pre-installation steps outlined below.

1. Practical knowledge of core AWS services (EC2, IAM) and Linux command line.
2. Have administrator access to an AWS account into which you can deploy Service Workbench on AWS.

1.1 Enable AWS Cost Explorer

1. Enable AWS Cost Explorer by visiting the AWS Cost Management service for the account that you will be deploying Service Workbench on AWS into.

2. (After 24h) Activate the following cost allocation tags in the AWS Billing and Cost Management Dashboard to enable billing information within Service Workbench on AWS. These tags appear after enabling the AWS Cost Explorer.

   - `Proj`, `Env`, `CreatedBy`

1.2 Create a Development Environment using AWS Cloud9

To deploy and configure Service Workbench on AWS, you will need a development environment. To avoid any confusion with your local environment, we will use AWS Cloud9, which is a cloud-based integrated development environment (IDE) that lets you write, run, and debug your code with just a browser.

1. Within your AWS Console, check that you are in the region in which you will be deploying Service Workbench.
2. Open the AWS Cloud9 console
3. Choose **Create Environment**
4. Fill the Name field with: `c9-swb-dev`
5. Add a meaningful description
6. Click on **Next Step**
7. Leave the Environment Type as *Create a new EC2 instance for environment (direct access)*
8. Under **Instance type**, Choose **Other instance type** - choose **m5.xlarge** (you can use a t3.medium with customers)
9. In Platform, choose Amazon Linux 2
10. Scroll down to tags and include:
    1. Key: **Owner**, Value: Your loginname, alias, or another unique identifier
    2. Key: **Env**, Value: `dev`
    3. Key: **Proj**, Value: `swb-workshop`

11. Click on **Next Step**
12. Review the configuration, and click on **Create Environment**

1.3 Install required tools and utilities

This step clones a repository of utility scripts, and runs the script `tools-init.sh` to install all libraries and tools required to deploy Service Workbench on AWS. The script also enlargens the default 10 GB instance store provided with Cloud9.

1. Open a new Terminal in Cloud9
2. Clone the Service Workbench on AWS using Cloud9 boostrap repository

```
cd ~/environment
git clone https://github.com/aws-samples/aws-swb-cloud9-init
cd aws-swb-cloud9-init
```

2. Review, and then run, the init script `tools-init.sh`. This script:

    - Sets your current region as the `AWS_REGION` environment variable, uses this value as the default for `aws configure`, and appends this setting to your `.bashrc` file.
    - Clones the Service Workbench source code from its GitHub repository to the directory `service-workbench-on-aws/`.
    - Runs the script `cloud9-resize.sh` to change the Cloud9 volume from 10 GB to 50 GB.
    - Installs **nvm** (Node Version Manager), and uses this to install the latest version of **Node**.
    - Installs the Node software **serverless**, **pnpm**, **hygen**, **yarn**, **docusaurus**.
    - Installs **Packer**, used to build custom AMI images.

```
# From directory ~/environment/aws-swb-cloud9-init
source tools-init.sh
```

1.4 (Optional) Access platform reference documentation

A static version of the Service Workbench documentation is available here

The documentation is available via Docusaurus, which runs as a web server. To build and view the documentation, either clone the Service Workbench source code to your local machine, or modify the security group of your Cloud9 instance as described in this documentation

```
cd ~/environment/service-workbench-on-aws/docs
yarn
yarn start --port 3000 --host 0.0.0.0
```

2. Installing and configuring Service Workbench on AWS

This section describes how to deploy Service Workbench on AWS using the AWS Cloud9 instance that we created in the pre-requisites section. Alternatively, it is possible to deploy from a local computer using local AWS CLI credentials. Keep in mind that tools and libraries installed in the previous section might be incompatible with your local system.

2.1 Creating a new Service Workbench Configuration

In this section, you will install Service Workbench components into your AWS account. Once the last step is underway, you can proceed to the next section (Install AMIs for EC2-based workspaces) to run both processes simultaneously.

1. In the terminal, export the **Stage Name** that is going to be used in the deployment process, for this guide we will be using `demo` as the Stage Name.
    - *Note: The stage name is included in the name of the Amazon S3 storage bucket, so must be Amazon S3-compatible (lower-case characters, numbers, periods, and dashes), and fewer than 10 characters.*

```
echo 'export STAGE_NAME=demo' >> ~/.bashrc
source ~/.bashrc
```

2. Create a copy of the example configuration that comes bundled with the repository.

```
# The configuration file must be named after the stage name
cd ~/environment/service-workbench-on-aws/main/config/settings
cp example.yml ${STAGE_NAME}.yml
```

3. Open the configuration file `demo.yml` in the Cloud9 editor, and uncomment and set values for:
    - `solutionName`: The solutionName is used in Amazon S3 bucket names so must be Amazon S3-compatible (lower-case characters, numbers, periods, and dashes)
    - `awsRegion`: The region code (eg `us-east-1`) you will be using for the deployment. Make sure to use the same region when you are using the AWS Console. Region codes may be looked up here: Regional Endpoints
4. In the terminal, run the `environment-deploy.sh` script to complete the installation, specifying the stage name as a parameter.
    - Note: This takes up to 15 minutes and can be ran in parallel with the AMI installation step, below.

```
cd ~/environment/service-workbench-on-aws/
./scripts/environment-deploy.sh ${STAGE_NAME}
```

5. Once the above step has completed, copy the root account password and website URL for later use.
    - Note: To retrieve this information again, run this command:
        - `scripts/get-info.sh ${STAGE_NAME}`
6. Using the URL and root password from above, visit the Service Workbench on AWS website and log in using user **root**.

2.2 Install AMIs for EC2-based workspaces

To use EC2-based workspaces, you must first install Amazon EC2 AMIs for these workspaces. This process may be run concurrently with the previous section (while `environment-deploy.sh` is running). To run both simultaneously, open a new terminal in Cloud9 (check that the environment variable `STAGE_NAME` is set correctly in the new terminal)

1. In the terminal, run the following command to start the Amazon EC2 AMI generation

```
cd ~/environment/service-workbench-on-aws/main/solution/machine-images/
pnpx sls build-image -s ${STAGE_NAME}
```

2. Once the process has been completed, you can verify that the Amazon EC2 AMI were created by running:

```
swb-ami-list
```

Note: This alias is defined in your `~/.bashrc file` as a shortcut to query the Amazon EC2 API. You should see AMIs for **EC2-LINUX**, **EC2-RSTUDIO**, **EC2-WINDOWS**, and **EMR**.

---

3. Configuring Accounts

In this section, you will set up your Service Workbench instance with accounts, workspaces, and other features, that can be used for evaluating the main Service Workbench features. You can perform the steps in this section after logging into your Service Workbench web interface as root for the first time (the verification step of the Install the Service Workbench on AWS platform section above).

3.1 Compute Hosting account setup

Compute Hosting accounts are the accounts in which research compute resources are deployed, and which are responsible for the billing of those resources. In this deployment, the hosting account will be the same as the deployment account.

1. Enter the host account script directory, and run the following command to create an AWS CloudFormation stack named `swb-hosting-compute-$HOSTINGACCOUNT_ARG-$STAGE_NAME-stack`, using the 12-digit account number that was passed to the script, and the `STAGE_NAME` variable.

```
# Go to our hosting account script dir
cd ~/environment/aws-swb-cloud9-init/hosting-account/
# Deploy the cloudformation stack for the hosting acount
./create-host-account.sh 12-DIGIT-ACCOUNT-ID
```

- Note: This step is dependent on the `STAGE_NAME` environment variable being set, from Step 2.1.1
- Note: If you want to use the same account as the AWS Cloud9 is running, retrieve the account id by running:

```
aws sts get-caller-identity | jq -r '.Account'
```

2. (For information only.) The parameters passed to the stack are described below, but can also be seen on the `hosting-account-cfn-args-${STAGE_NAME}.json` inside the `aws-swb-cloud9-init` directory.

Parameter Name	Value
Namespace	Stage name
CentralAccountId	The current AWS Cloud9 AccountId
ExternalId	`workbench`
VpcCidr	Default (10.0.0.0/16)
VpcPublicSubnet1Cidr	Default (10.0.0.0/19)
ApiHandlerArn	ApiHandlerRoleArn created in the first step
LaunchConstraintPolicyPrefix	Default (*)
LaunchConstraintRolePrefix	Default (*)
WorkflowRoleArn	WorkflowLoopRunnerRoleArn created in the first step

4. Deploy the stack from the CloudFormation console. The Outputs of the stack will contain values similar to:

Key	Value
CrossAccountEnvMgmtRoleArn	`arn:aws:iam::0000:role/<stage>-xacc-env-mgmt`
CrossAccountExecutionRoleArn	`arn:aws:iam::0000:role/<stage>-cross-account-role`
EncryptionKeyArn	`arn:aws:kms:us-east-2:0000:key/f00-f00-f00`
VPC	`vpc-f00f00`
VpcPublicSubnet1	`subnet-f00f00`

5. In the website of your Service Workbench deployment, select “Accounts” (left navigation), “AWS Accounts” (tab), “Add Account” (button). Fill in values as follows:

Field	Value
Account Name	As desired
AWS Account ID	12-digit ID of imported account
Role ARN	**CrossAccountExecutionRoleArn** value
AWS Service Catalog Role Arn	**CrossAccountEnvMgmtRoleArn** value
External ID	As specified (default: `workbench`)
Description	As desired
VPC ID	**VPC** value
Subnet ID	**VpcPublicSubnet1** value
KMS Encryption Key ARN	**EncryptionKeyArn** value

Verify that the account now appears under ‘AWS Accounts’.

3.2 Create default index, project, and admin account

Service Workbench supports a three-tier hierarchy for managing how research resources are deployed and billed.

- At the top level, Service Workbench supports one or more host accounts. Each of these accounts hosts, and is billed for, the compute resources deployed in the account.
- Each host account is linked to by one or more Indexes.
- Each Index is linked to by one or more Projects.
- Service Workbench users are associated with Projects.

In this section, you will create an index, project, and a local user to be the administrator of the project. Users are associated with hosting accounts through projects and indexes, so the AWS account that hosts a research resource will be determined by the project and index that the user belongs to.

1. Navigate to Accounts (left navigation), Indexes (tab), Add Index (button)
    1. Enter a unique name for the index (e.g. “index01”)
    2. Select the AWS account that you added to Service Workbench on AWS in the last section
    3. Enter a description
    4. **Add Index**
2. Select the Projects tab in the Accounts interface, select Add Project
    1. Enter a unique name for the project (e.g. “project01”)
    2. Select the index that you created in the last step
    3. Enter a description
    4. Leave the project admin blank (for now)
    5. **Add Project**
3. Navigate to Users (left navigation), select Add Local User (button)
    1. Enter a username in email format (this does not have to be a real email address)
    2. Enter first and last names of the user
    3. Enter a secure password for the user
    4. Select **admin** for the user role
    5. Select the project that you created in the last section for Projects
    6. Select Active for Status
    7. **Add Local User**
4. Repeat this step to create a user with Researcher role, for demonstration purposes
5. Go back to Projects under Accounts
    1. Select **Detail** for the project that you created, then **Edit**
    2. For Project Admins, select the admin user you created in the last step
    3. **Save**
6. Log out of the root account and log back in using your new admin user

3.3 Configuring Workspaces and Studies

In this section, we will make one of the five default workspace types available for researchers. Workspace types appear in Service Workbench on AWS after they have been configured in AWS Service Catalog (done during the installation process). A workspace type is made available for deployment by a user by creating a Configuration, which defines the size of the resource deployed as well as which roles that can deploy the configuration.

3.4 Create a Workspace Configuration

1. Navigate to Workspace Types
    1. Select **SageMaker Notebook**
    2. Select **Import**
2. Add a workspace configuration:
    1. Select **Import Workspace Type** in the SageMaker Notebook workspace type, to proceed to Configurations
    2. Select **Configurations**, then **Add Configuration**
    3. Fill in the **Basic Information**:

        - All fields are required
        - Id may not contain spaces
        - Description and Estimate Costs are free text supporting MarkDown
    4. Fill in the **Access Control**:
        - In **Roles Allowed**, select **Admin** and **Researcher**
    5. Fill in **Input Parameters**:
        - For most fields, begin typing the name of the field, and select the autocomplete option of that name.
            - For **AccessFromCIDRBlock**, begin typing `cidr` to autocomplete
            - For **Instance type**, use **ml.t3.medium**
            - For **AutoStopIdleTimeInMinutes**, enter a numerical value
    6. Add tags, if required, and select **Done** to create the configuration
3. Under **Workspace Types**, select **Approve** to make this product available for deployment by users

3.5 Create a Study

Studies are storage locations, appearing to users as file systems mounted within their workspace. Each study is implemented as a policy-protected path in the single S3 bucket storing all data in this Service Workbench on AWS deployment.

There are three classes of Studies: **My Study**, which are private to the user; **Organization**, which may be shared with other users, and **Open Data**, which is the AWS Open Data dataset.

1. Open **Studies**, and select **Create Study**
    - ID must not contain spaces
    - Select **Organization** Study type
    - **Name** and **Description** are required
    - Select your project from **Project ID**
2. In the **Organization** tab, use **Upload Files** to store some files in your Organization Study
3. Modify the permissions to share the study with the second user, created earlier

Note: Studies created using My Study cannot be shared with other users. Therefore, it's recommended that users create Organization studies, which can be single-user access, or shared, if there is a possibility that the study data would be shared in the future.

3.6 Create and deploy a Workspace

Workspaces are created from within the Studies interface, after selecting the studies to be mounted to the workspace. If a workspace is created from the Workspaces interface, it will not have access to any studies.

1. Open **Studies**
    - Select your study from the **Organization** tab
        - Note 'Selected Studies: 1' is displayed. You may attach multiple Studies to a Workspace. c. Select **Next**
2. In **Select Compute**, select **SageMaker Notebook**
3. From **Create Workspace**, launch the workspace

    - The Name may not contain spaces
    - The default CIDR may be widened
    - Select your Project ID
    - Select your configuration
    - Select **Create Research Workspace**
4. Open **Workspaces** in the left sidebar
    - Select **Connect** to open a new tab with a Jupyter Notebook running on the SageMaker workspace

4. Post-Deployment Tasks

Once your basic installation is complete, you can stop or terminate the AWS Cloud9 instance that you used to deploy Service Workbench on AWS, as the instance is only needed to for the deployment process. Stopping the instance, rather than terminating it, is recommended if you intend to update your Service Workbench on AWS deployment as the platform is updated.

For creating a Service Workbench on AWS deployment that can be demonstrated to researchers, it's recommended that you create workspace configurations for several of the default workspace types.