



**AZURE CLOUD SOLUTION DOCUMENTATION FOR XAI-
SERVICE**

**Electrical and Computer Engineering
Summer 2023**

Dr. Yan Liu.

Table of Contents

<i>Electrical and Computer Engineering Summer 2023</i>	<i>1</i>
<i>1. Prerequisite Installations.....</i>	<i>3</i>
Note: LINUX Subsystem.....	3
1.1. WSL Installation:.....	3
1.2. Anaconda Installation	3
1.3. Docker Installation.....	3
1.4. Azure CLI Installation	4
1.5. Container Registry Creation.....	4
<i>4. CI-CD Deployment using AWS-Terraform</i>	<i>5</i>
4.1. Prerequisites:	5
4.1.1. azure-buildspec.yml:	5
4.1.2. S3 bucket:	5
4.1.3. CodeBuild Configuration:	5
4.1.4. Connection Strings:.....	5
4.2. S3 Configuration:.....	5
4.3. CodeBuild Configuration:.....	8
4.3. Connection String Configuration:	12
4.4. CodePipeline Configuration:	13
4.5. *Resource clean-up:	17

1. Prerequisite Installations

Note: LINUX Subsystem.

It is better to have WSL in windows than using Ubuntu. Please avoid ARM based processor for this application.

1.1. WSL Installation:

- STEP 1. After enabling the WSL feature, go to the Microsoft Store (you can search for it in the Start menu).
- STEP 2. Search for the Linux distribution you want to install (e.g., Ubuntu, Debian, Fedora, SUSE Linux Enterprise Server, etc.).
- STEP 3. Click on the distribution you prefer, and then click the "Install" button. This will download and install the Linux distribution on your system.

1.2. Anaconda Installation

- STEP 1. Open a new terminal.
- STEP 2. Download [Anaconda](#) for Linux from their official page.
- STEP 3. Execute: `chmod +x Anaconda3-2023.03-1-Linux-x86_64.sh`
- STEP 4. Execute: `./Anaconda3-2023.03-1-Linux-x86_64.sh`
- STEP 5. Accept Terms and Conditions
- STEP 6. Enter the installation directory as "home/<user>/anaconda"
- STEP 7. Execute: `nano ~/.bashrc`
- STEP 8. Add the path: `export PATH="/path/to/anaconda3/bin:$PATH"`
- STEP 9. `Ctrl+X-> y -> Enter.`
- STEP 10. `source ~/.bashrc`
- STEP 11. Open a new terminal.
- STEP 12. Execute: `su ->` give the user credentials.
- STEP 13. Execute `conda init bash`.
- STEP 14. Check the conda installation by executing "`conda --version`".
- STEP 15. Clone the repository and `cd` to the working directory.
- STEP 16. Create new environment: `conda create -n xai39 python=3.9`
- STEP 17. Execute: `conda activate xai39`
- STEP 18. Try running the application locally before dockerising the application.

1.3. Docker Installation

- STEP 1. Open a new terminal in sudo access ("`su`").
- STEP 2. Execute the following commands consequently,

- a. apt update
- b. apt install apt-transport-https ca-certificates curl software-properties-common
- c. curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
- d. echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu \$(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
- e. apt update
- f. apt install docker-ce docker-ce-cli containerd.io
- g. systemctl status docker
- h. usermod -aG docker \$USER

STEP 3. docker --version

STEP 4. docker login

STEP 5. Verify the login executed successfully.

1.4. Azure CLI Installation

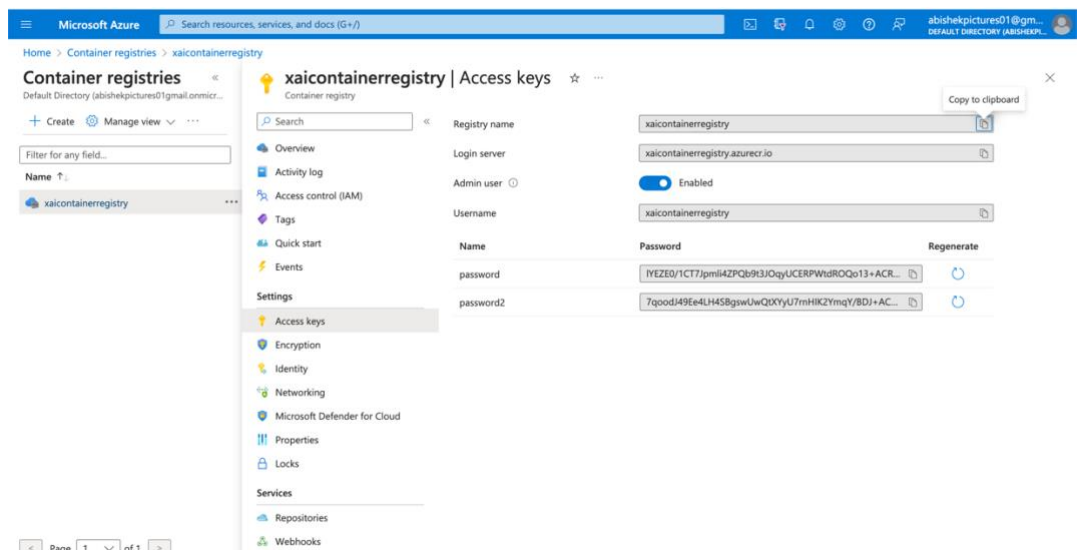
STEP 1. Visit <https://learn.microsoft.com/en-us/cli/azure/install-azure-cli-windows?tabs=azure-cli#install-or-update>.

STEP 2. Download and install Azure CLI.

1.5. Container Registry Creation

STEP 1. Create a container registry "xaicontainerregistry".

STEP 2. Enable admin user and use the password for the environment variables.



4. CI-CD Deployment using AWS-Terraform

4.1. Prerequisites:

4.1.1. azure-buildspec.yml:

The following file in the repository, stores the flow of series of commands to execute to deploy our application.

4.1.2. S3 bucket:

Create an S3 bucket to store the terraform state. We need the state to be saved, because on every ci-cd deployment conflict will arise as the state is not saved.

4.1.3. CodeBuild Configuration:

Here we set the location of our buildspec, the values for environment variables and the runtime for performing the build.

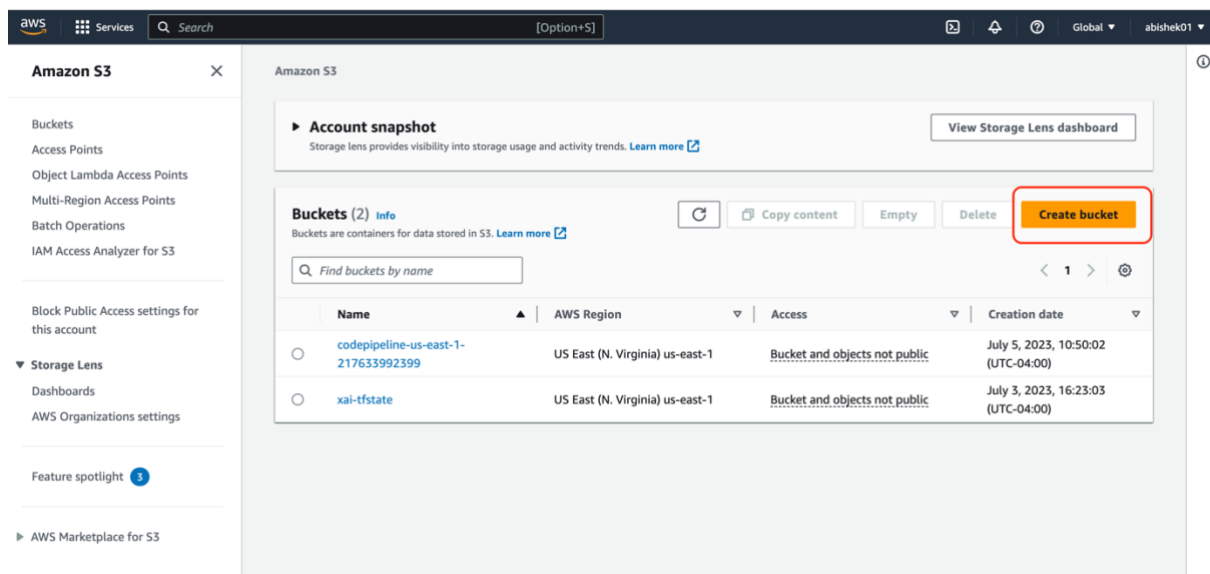
4.1.4. Connection Strings:

We need to save the azure and mongodb connection strings in AWS Systems Manager to access in our buildspec securely.

4.2. S3 Configuration:

STEP 1. In AWS, go to S3.

STEP 2. Create a bucket with name “xai-tfstate”, because our “provider.tf” has configured under this name.



STEP 3. Follow the below screenshots to and click create.

Create bucket [Info](#)

Buckets are containers for data stored in S3. [Learn more](#)

General configuration

Bucket name

Bucket name must be unique within the global namespace and follow the bucket naming rules. [See rules for bucket naming](#)

AWS Region

US East (N. Virginia) us-east-1

Copy settings from existing bucket - *optional*

Only the bucket settings in the following configuration are copied.

Choose bucket

Object Ownership [Info](#)

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

☒ ACLs disabled (recommended)

All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.

☐ ACLs enabled

Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

Bucket Versioning

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. [Learn more](#)

Bucket Versioning

☒ Disable

☐ Enable

Tags (0) - optional

You can use bucket tags to track storage costs and organize buckets. [Learn more](#)

No tags associated with this bucket.

Add tag

Default encryption [Info](#)

Server-side encryption is automatically applied to new objects stored in this bucket.

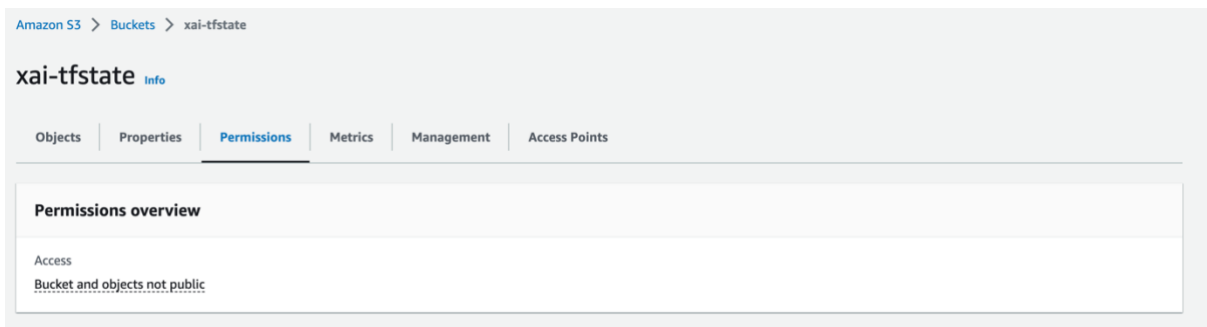
Encryption type [Info](#)

☒ Server-side encryption with Amazon S3 managed keys (SSE-S3)

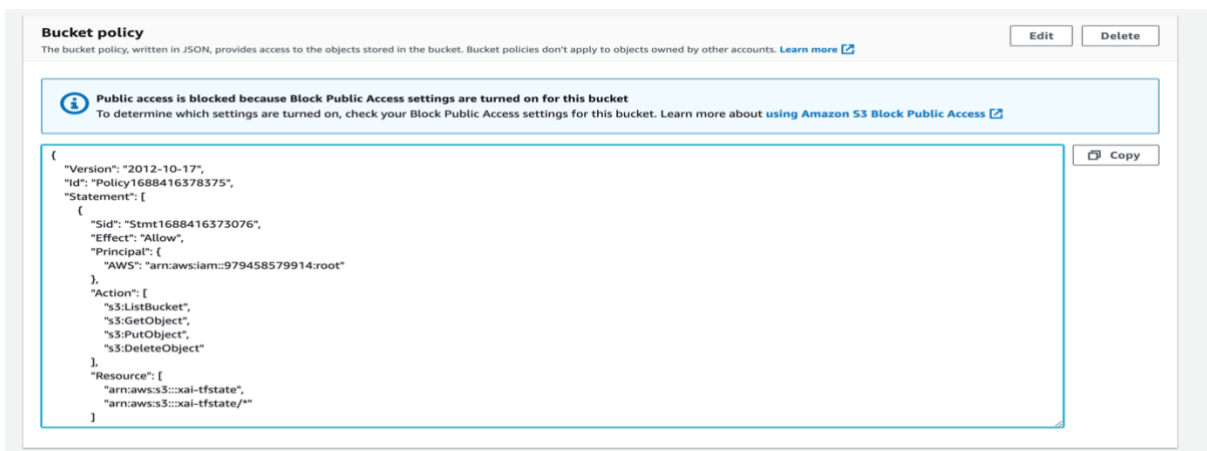
☐ Server-side encryption with AWS Key Management Service keys (SSE-KMS)

☐ Dual-layer server-side encryption with AWS Key Management Service keys (DSSE-KMS)

STEP 4. Click the bucket created and go to permissions tab.



STEP 5. Under bucket policy add the following json policy.



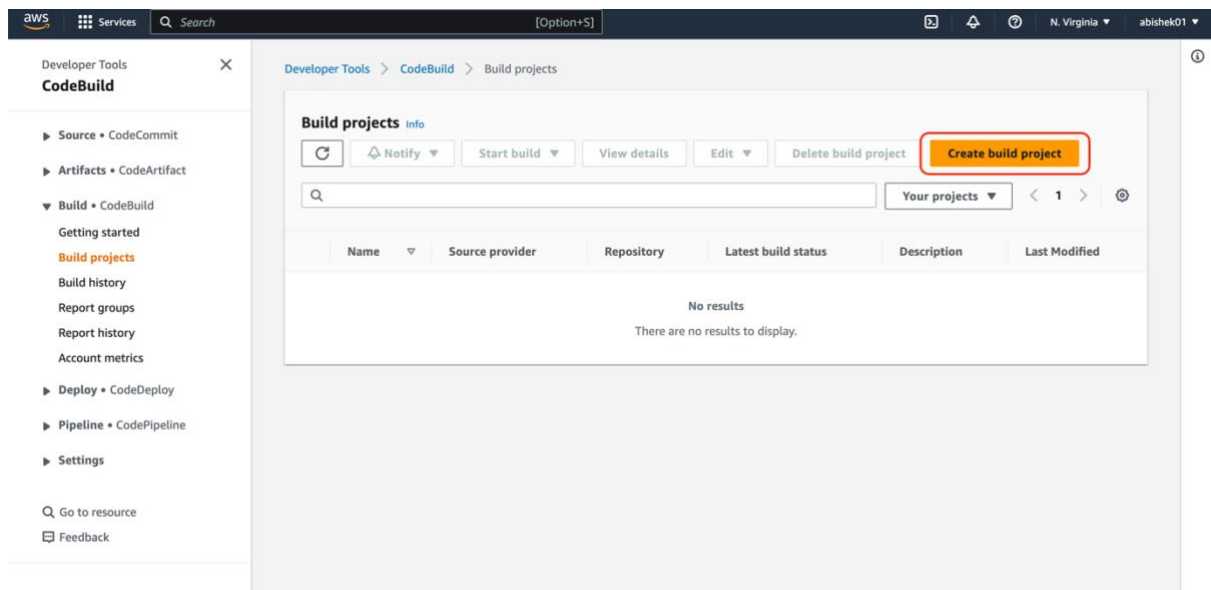
```
{
  "Version": "2012-10-17",
  "Id": "Policy1688416378375",
  "Statement": [
    {
      "Sid": "Stmnt1688416373076",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::<account-id>:root"
      },
      "Action": [
        "s3:ListBucket",
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObject"
      ],
      "Resource": [
        "arn:aws:s3:::xai-tfstate",
        "arn:aws:s3:::xai-tfstate/*"
      ]
    }
  ]
}
```

```
]
}
]
}
```

4.3. CodeBuild Configuration:

STEP 1. In AWS console, go to CodeBuild.

STEP 2. Click create build project.



STEP 3. Give the project name as “xai-build”.

STEP 4. Follow the steps shown in below images.

Create build project

Project configuration

Project name

xai-build

A project name must be 2 to 255 characters. It can include the letters A-Z and a-z, the numbers 0-9, and the special characters - and _.

Description - *optional*

Build badge - *optional*

☐ Enable build badge

Enable concurrent build limit - *optional*

Limit the number of allowed concurrent builds for this project.

☐ Restrict number of concurrent builds this project can start

► Additional configuration

tags

Source

Add source

Source 1 - Primary

Source provider

GitHub

Repository

☒ Public repository

☐ Repository in my GitHub account

Repository URL

https://github.com/ZeruiW/XAI-Service

https://github.com/<user-name>/<repository-name>

Connection status

You are connected to GitHub using OAuth.

Disconnect from GitHub

Source version - *optional* [Info](#)

Enter a pull request, branch, commit ID, tag, or reference and a commit ID.

► Additional configuration

Git clone depth, Git submodules

Environment



Environment image

☒ **Managed image**
Use an image managed by AWS CodeBuild

☐ **Custom image**
Specify a Docker image

Operating system

Amazon Linux 2 ▼

 The programming language runtimes are now included in the standard image of Ubuntu 18.04, which is recommended for new CodeBuild projects created in the console. See [Docker Images Provided by CodeBuild for details](#) .

Runtime(s)

Standard ▼

Image

aws/codebuild/amazonlinux2-x86_64-standard:5.0 ▼

Image version

Always use the latest image for this runtime version ▼

Image

aws/codebuild/amazonlinux2-x86_64-standard:5.0 ▼

Image version

Always use the latest image for this runtime version ▼

Environment type

Linux GPU ▼

Privileged

☒ Enable this flag if you want to build Docker images or want your builds to get elevated privileges

Service role

☒ **New service role**
Create a service role in your account

☐ **Existing service role**
Choose an existing service role from your account

Role name

codebuild-xai-build-service-role

Type your service role name

► Additional configuration

Timeout, certificate, VPC, compute type, environment variables, file systems

Build logs


Phase details

Reports

Environment variables

Build details

Resource utilization

Name	Value	Type
DOCKER_PASSWORD		PLAINTEXT
DOCKER_USERNAME		PLAINTEXT
AZURE_CLIENTID		PLAINTEXT
AZURE_PASSWORD		PLAINTEXT
AZURE_TENANTID		PLAINTEXT
OS_IMAGE_PASSWORD		PLAINTEXT
REGISTRY_PASSWORD		PLAINTEXT
AZURE_SUBSCRIPTIONID		PLAINTEXT

- OS_IMAGE_PASSWORD is user defined VM password.
- REGISTRY_PASSWORD is copied from the access keys from azure container registry.

STEP 5. Use buildspec file and give the path "backend/azure-buildspec.yml"

Batch configuration

You can run a group of builds as a single execution. Batch configuration is also available in advanced option when starting build.

☐ Define batch configuration - *optional*

You can also define or override batch configuration when starting a build batch.

Artifacts

Add artifact

Artifact 1 - Primary

Type

No artifacts ▼

You might choose no artifacts if you are running tests or pushing a Docker image to Amazon ECR.

▶ Additional configuration
Cache, encryption key

Logs

CloudWatch

☒ **CloudWatch logs - optional**
Checking this option will upload build output logs to CloudWatch.

Group name

Stream name

S3

☐ **S3 logs - optional**
Checking this option will upload build output logs to S3.

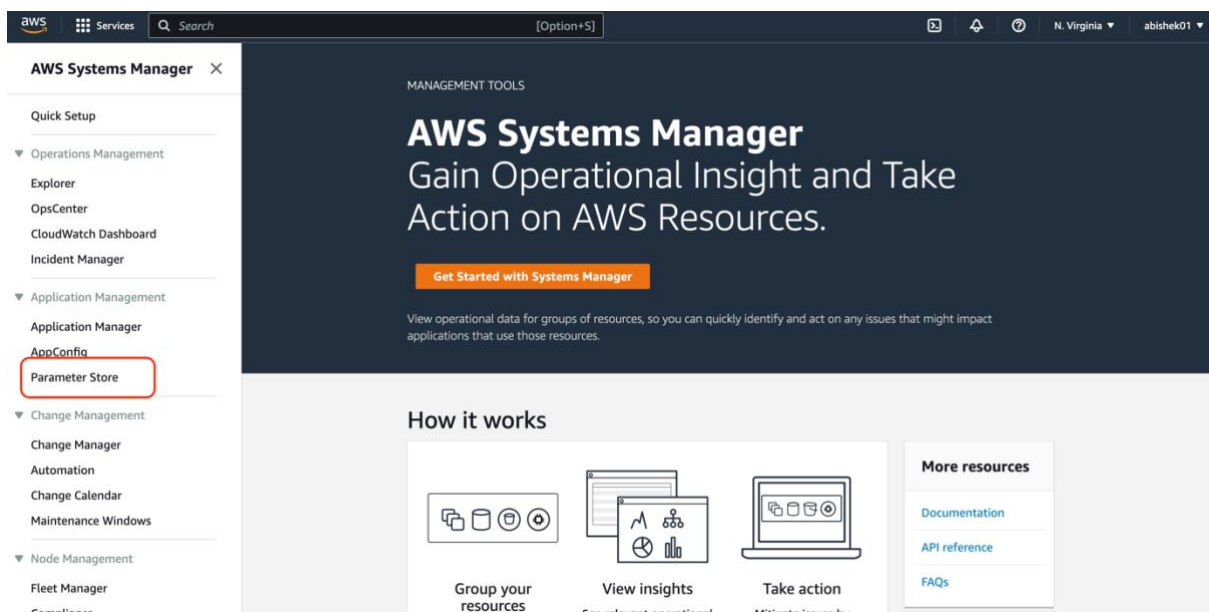
Cancel
Create build project

STEP 6. Click create.

4.3. Connection String Configuration:

STEP 1. In AWS Console, go to “Systems Manager”.

STEP 2. Go to parameter Store.



STEP 3. Use the name of the file as “azure-con-str” and “mongo-dev” as we have configured the name in buildspec.

Name

Description — Optional

Tier
Parameter Store offers standard and advanced parameters.

☒ **Standard**
Limit of 10,000 parameters. Parameter value size up to 4 KB. Parameter policies are not available. No additional charge.

☐ **Advanced**
Can create more than 10,000 parameters. Parameter value size up to 8 KB. Parameter policies are available. Charges apply

Type

☒ **String**
Any string value.

☐ **StringList**
Separate strings using commas.

☐ **SecureString**
Encrypt sensitive data using KMS keys from your account or another account.

Data type

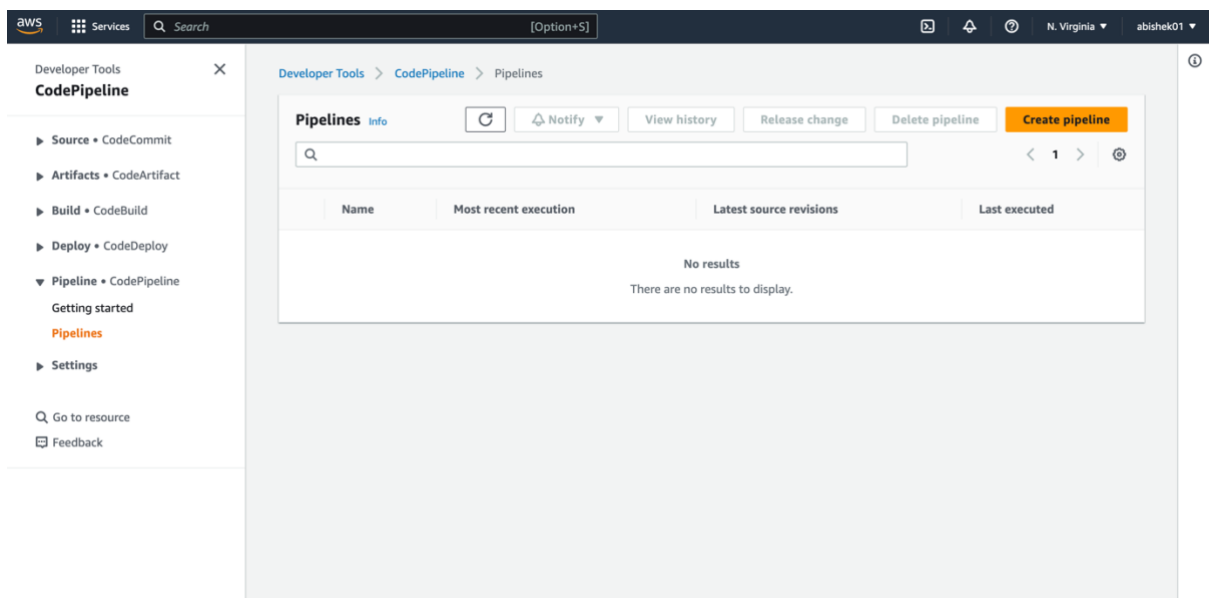
Value

Maximum length 4096 characters.

4.4. CodePipeline Configuration:

STEP 1. In AWS Console, go to CodePipeline.

STEP 2. Click Create Pipeline.



STEP 3. Give the pipeline name as “xai-pipeline”, rest as default and click next.

Choose pipeline settings [Info](#)

Pipeline settings

Pipeline name

Enter the pipeline name. You cannot edit the pipeline name after it is created.

No more than 100 characters

Service role

☒ **New service role**
Create a service role in your account

☐ **Existing service role**
Choose an existing service role from your account

Role name

Type your service role name

☒ **Allow AWS CodePipeline to create a service role so it can be used with this new pipeline**

▼ Advanced settings

Artifact store

☒ **Default location**
Use the default artifact store (Amazon S3 codepipeline-us-east-1-217633992399) designated in the same region and account as your pipeline

☐ **Custom location**
Choose an existing S3 location from your account in the same region and account as your pipeline

Encryption key

☒ **Default AWS Managed Key**
Use the AWS managed customer master key for CodePipeline in your account to encrypt the data in the artifact store.

☐ **Customer Managed Key**
To encrypt the data in the artifact store under an AWS KMS customer managed key, specify the key ID, key ARN, or alias ARN.

Cancel

Next

STEP 4. Source provider as GitHub (Version 2).

Add source stage [Info](#)

Source

Source provider

This is where you stored your input artifacts for your pipeline. Choose the provider and then provide the connection details.

GitHub (Version 2) ▼



New GitHub version 2 (app-based) action

To add a GitHub version 2 action in CodePipeline, you create a connection, which uses GitHub Apps to access your repository. Use the options below to choose an existing connection or create a new one. [Learn more](#)

Connection

Choose an existing connection that you have already configured, or create a new one and then return to this task.



or

Connect to GitHub

STEP 5. Connect to Github, give your github username when a window popup.

STEP 6. Follow the image for rest of the fields and click next.

Connection

Choose an existing connection that you have already configured, or create a new one and then return to this task.



arn:aws:codestar-connections:us-east-1:979458579914:connection/f41fef71 X

or

Connect to GitHub



Ready to connect

Your GitHub connection is ready for use.

Repository name

Choose a repository in your GitHub account.



abishekat/XAI-Service X

<account>/<repository-name>

Branch name

Choose a branch of the repository.



main X

Change detection options

☒ Start the pipeline on source code change

Automatically starts your pipeline when a change occurs in the source code. If turned off, your pipeline only runs if you start it manually or on a schedule.

Output artifact format

Choose the output artifact format.



CodePipeline default

AWS CodePipeline uses the default zip format for artifacts in the pipeline. Does not include Git metadata about the repository.



Full clone

AWS CodePipeline passes metadata about the repository that allows subsequent actions to do a full Git clone. Only supported for AWS CodeBuild actions.

Cancel

Previous

Next

STEP 7. Build provider as AWS CodeBuild and give the project name as "xai-build" created in section 4.2. Click on Next.

Build - optional

Build provider
This is the tool of your build project. Provide build artifact details like operating system, build spec file, and output file names.

AWS CodeBuild

Region

US East (N. Virginia)

Project name
Choose a build project that you have already created in the AWS CodeBuild console. Or create a build project in the AWS CodeBuild console and then return to this task.

xai-build

 or

Create project

Environment variables - optional
Choose the key, value, and type for your CodeBuild environment variables. In the value field, you can reference variables generated by CodePipeline. [Learn more](#)

Add environment variable

Build type

☒ **Single build**
Triggers a single build.

☐ **Batch build**
Triggers multiple builds as a single execution.

Cancel

Previous

Skip build stage

Next

STEP 8. Skip the deploy stage.

Add deploy stage [Info](#)

Deploy - optional

Deploy provider
Choose how you deploy to instances. Choose the provider, and then provide the configuration details for that provider.

Cancel

Previous

Skip deploy stage

Next

STEP 9. Click create pipeline.

Step 4: Add deploy stage

Deploy action provider
Deployment stage
No deploy

Cancel Previous Create pipeline

4.5. *Resource clean-up:

Once you have successfully deployed using terraform and want to remove the resources created. Follow the below steps.

- STEP 1. Go to the local setup of your xai-service.
- STEP 2. Open terminal and "cd backend/terraform".
- STEP 3. Use the command "terraform init". It copies the terraform state file from s3.
- STEP 4. Use the command "terraform destroy" and type yes on prompt.
- STEP 5. All the resources created by CodeBuild will be destroyed.