



AWS CLOUD SOLUTION DOCUMENTATION FOR XAI-SERVICE

**Electrical and Computer Engineering
Summer 2023**

Dr. Yan Liu.

Abstract:

This documentation explains the procedure to deploy the xai-service in two methods. One without using terraform and the other with terraform.

Table of Contents

1.	<i>Prerequisite Installations</i>	3
	Note: LINUX Subsystem.....	3
1.1	VMbox and Ubuntu:.....	3
1.2	Anaconda Installation	3
1.3	Docker Installation.....	4
1.4	AWS CLI Installation	4
2.	<i>IAM role creation – W/O Terraform</i>	7
2.1.	ECS – Instance Role.....	7
2.2.	ECS – Task Execution Role.....	9
3.	<i>Deployment Using AWS console without terraform</i>	11
3.1.	Creating a cluster.....	11
	Note: Instance- g3.4xlarge/ g3s.xlarge	12
3.2.	Creating a task definition - TD.....	14
3.2.1.	Configure task and container definitions.	15
3.2.2.	Task execution IAM role	15
3.2.3.	Container Definitions.....	16
3.3.	Creating a Service.....	17
4.	<i>CI-CD Deployment using AWS-Terraform</i>	20
4.1.	Prerequisites:.....	20
4.1.1.	buildspec.yml:.....	20
4.1.2.	S3 bucket:	20
4.1.3.	CodeBuild Configuration:	20
4.1.4.	Connection Strings:.....	20
4.2.	S3 Configuration:.....	20
4.3.	CodeBuild Configuration:.....	23
4.3.	Connection String Configuration:	27
4.4.	CodePipeline Configuration:	27
4.5.	*Resource clean-up:	32

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 VMbox and Ubuntu:

- STEP 1. Download and install [VMBox](#) depending on the OS.
- STEP 2. Download the [ISO image](#) to install Linux.
- STEP 3. Launch VMBox to create a new VM with the image downloaded with the following constraints,
 - a. Username and password.
 - b. The number of CPUs is 5.
 - c. The size of the HDD is about 100GB.

1.2 Anaconda Installation

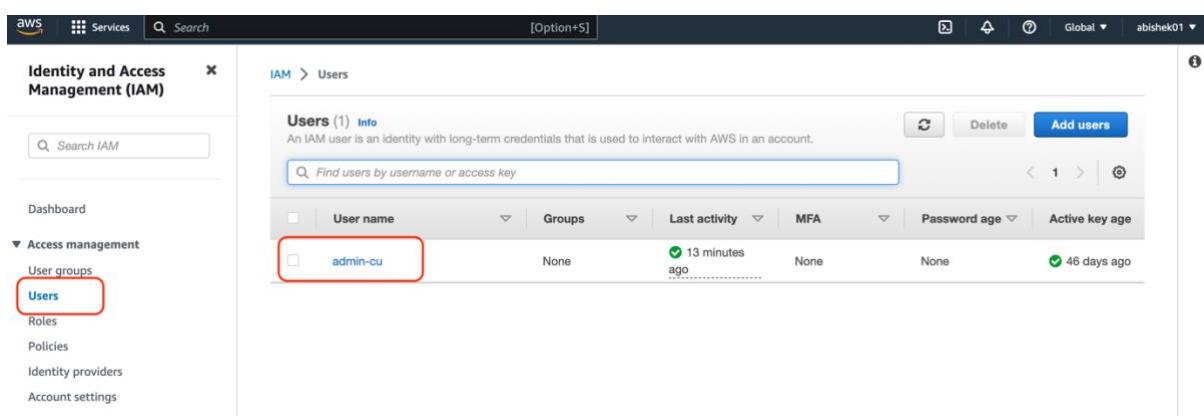
- STEP 1. Open a new terminal.
- STEP 2. Download [Anaconda](#) for Linux from their official page.
- STEP 3. Execute: chmod +x Anaconda3-2023.03-1-Linux-x86_64.sh
- STEP 4. Execute: ./Anaconda3-2023.03-1-Linux-x86_64.sh
- STEP 5. Accept Terms and Conditions
- STEP 6. Enter the installation directory 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 AWS CLI Installation

- STEP 1. Execute the command as sudo user in the new terminal “apt-get install awscli”
- STEP 2. Use “aws configure” to configure the aws credentials for cli access.
- STEP 3. How to get the AWS keys for login.
- STEP 4. Go to IAM and select Users on the left tab.



- STEP 5. Select the user and go to “Security Credentials”.

admin-cu [Info](#)

[Delete](#)

Summary		
ARN arn:aws:iam::979458579914:user/admin-cu	Console access Disabled	Access key 1 AKIA6IDCH3XFNQ5ZF6PO - Active Used today. 46 days old.
Created May 20, 2023, 09:21 (UTC-04:00)	Last console sign-in -	Access key 2 Not enabled

Permissions | Groups | Tags | **Security credentials** | Access Advisor

Console sign-in	
Console sign-in link https://979458579914.signin.aws.amazon.com/console	Console password Not enabled

[Enable console access](#)

STEP 6. Now scroll to find “Access Keys”.

STEP 7. Create access key.

Access keys (1)

Use access keys to send programmatic calls to AWS from the AWS CLI, AWS Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) at a time. [Learn more](#)

[Create access key](#)

AKIA6IDCH3XFNQ5ZF6PO	
Description	Status
-	Active
Last used	Created
16 minutes ago	46 days ago
Last used region	Last used service
us-east-1	iam

[Actions ▾](#)

STEP 8. Choose CLI.

Avoid using long-term credentials like access keys to improve your security. Consider the following use cases and alternatives.

Use case

- Command Line Interface (CLI)**
You plan to use this access key to enable the AWS CLI to access your AWS account.
- Local code**
You plan to use this access key to enable application code in a local development environment to access your AWS account.
- Application running on an AWS compute service**
You plan to use this access key to enable application code running on an AWS compute service like Amazon EC2, Amazon ECS, or AWS Lambda to access your AWS account.
- Third-party service**
You plan to use this access key to enable access for a third-party application or service that monitors or manages your AWS resources.
- Application running outside AWS**
You plan to use this access key to enable an application running on an on-premises host, or to use a local AWS client or third-party AWS plugin.
- Other**
Your use case is not listed here.

Alternatives recommended

- Use AWS CloudShell, a browser-based CLI, to run commands. [Learn more](#)
- Use the AWS CLI V2 and enable authentication through a user in IAM Identity Center. [Learn more](#)

Confirmation

I understand the above recommendation and want to proceed to create an access key.

[Cancel](#) [Next](#)

STEP 9. Provide a name.

Set description tag - optional [Info](#)

The description for this access key will be attached to this user as a tag and shown alongside the access key.

Description tag value

Describe the purpose of this access key and where it will be used. A good description will help you rotate this access key confidently later.

cli-keys

Maximum 256 characters. Allowed characters are letters, numbers, spaces representable in UTF-8, and: _ . : / = + - @

[Cancel](#)

[Previous](#)

[Create access key](#)

STEP 10. Download the keys to local.

Access key best practices

- Never store your access key in plain text, in a code repository, or in code.
- Disable or delete access key when no longer needed.
- Enable least-privilege permissions.
- Rotate access keys regularly.

For more details about managing access keys, see the [Best practices for managing AWS access keys](#).

[Download .csv file](#)

[Done](#)

STEP 11. Now use the creds for “aws configure”, with region as us-east-1.

2. IAM role creation – W/O Terraform.

Open [IAM](#) in AWS Console.

2.1. ECS – Instance Role

Click on Roles in the left side tab and click “create role”.

The screenshot shows the AWS IAM Roles page. On the left, there's a sidebar with 'Identity and Access Management (IAM)' selected. The main area shows a table of existing roles, each with a checkbox, a role name, a 'Trusted entities' column (showing AWS services like autoscaling, ecs, servicequotas, support, and trustedadvisor), and a 'Last activity' column. A red box highlights the 'Create role' button at the top right of the table area. The table has columns for Role name, Trusted entities, and Last activity.

Select AWS Service, choose “Elastic Container Service” and click EC2 role for Elastic Container Service.

The screenshot shows the 'Add permissions' step in the IAM role creation wizard. It's Step 3: Name, review, and create. There are four options: 'AWS service' (selected and highlighted with a red box), 'AWS account', 'Web identity', and 'SAML 2.0 federation'. Below these is a 'Use case' section with 'Common use cases' for 'EC2' and 'Lambda', and a dropdown for 'Use cases for other AWS services' containing 'Elastic Container Service' (selected and highlighted with a red box) and 'EC2 Role for Elastic Container Service' (also highlighted with a red box).

Verify the following policy is added to the role.

The screenshot shows the 'Add permissions' step of the IAM role creation wizard. In the 'Permissions policies' section, a single policy named 'AmazonEC2ContainerServiceforEC2Role' is listed. This row is highlighted with a red box. The table columns include 'Policy name', 'Type', and 'Attached entities'. The 'Attached entities' column shows 'AWS m...' and '2'.

Give the role name “ecsInstanceRole”.

The screenshot shows the 'Name, review, and create' step. In the 'Role details' section, the 'Role name' field contains 'ecsInstanceRole'. The 'Description' field contains 'Allows EC2 instances in an ECS cluster to access ECS.' Both fields have character limits indicated below them: 64 characters for the role name and 1000 characters for the description.

Verify the following in step 2 and create the cluster.

The screenshot shows the 'Add permissions' step. It displays a summary of the attached policy 'AmazonEC2ContainerServiceforEC2Role', which is a 'AWS managed' policy. An 'Edit' button is visible at the top right.

Tags

Add tags - optional Info
Tags are key-value pairs that you can add to AWS resources to help identify, organize, or search for resources.

No tags associated with the resource.

Add tag

You can add up to 50 more tags.

Create role **Previous** **Cancel**

2.2. ECS – Task Execution Role

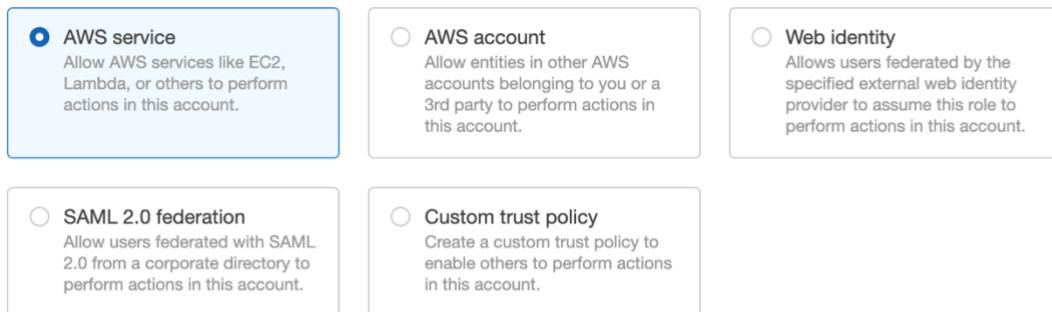
Click on Roles in the left side tab and click “create role”.

The screenshot shows the AWS IAM Roles page. On the left, there's a sidebar with 'Identity and Access Management (IAM)' selected under 'Access management'. The main area shows a table of existing roles:

Role name	Trusted entities	Last activity
AWSServiceRoleForAutoScaling	AWS Service: autoscaling (Service-Linked Role)	1 hour ago
AWSServiceRoleForECS	AWS Service: ecs (Service-Linked Role)	1 hour ago
AWSServiceRoleForServiceQuotas	AWS Service: servicequotas (Service-Linked Role)	6 days ago
AWSServiceRoleForSupport	AWS Service: support (Service-Linked Role)	6 days ago
AWSServiceRoleForTrustedAdvisor	AWS Service: trustedadvisor (Service-Linked Role)	-
ecs-autoscaling-role	AWS Service: application-autoscaling	-

A blue 'Create role' button is located at the top right of the table area.

Select AWS Service, choose “Elastic Container Service” and click Elastic Container Service Task.



Use case

Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Common use cases

- EC2**
Allows EC2 instances to call AWS services on your behalf.
- Lambda**
Allows Lambda functions to call AWS services on your behalf.

Use cases for other AWS services:

- Elastic Container Service**
 - Elastic Container Service**
Allows ECS to create and manage AWS resources on your behalf.
 - Elastic Container Service Autoscale**
Allows Auto Scaling to access and update ECS services.
 - Elastic Container Service Task**
Allows ECS tasks to call AWS services on your behalf.
 - EC2 Role for Elastic Container Service**
Allows EC2 instances in an ECS cluster to access ECS.

In “Add permissions” select “AmazonECSTaskExecutionRolePolicy”, check the checkbox, and click next. Give the role name “ecsTaskExecutionRole” and click create.

Permissions policies (856) [Info](#)

Choose one or more policies to attach to your new role.

[Create policy](#)

Filter policies by property or policy name and press enter.

1 match

<

1

>



"AmazonECSTaskExecutionRolePolicy" [X](#)

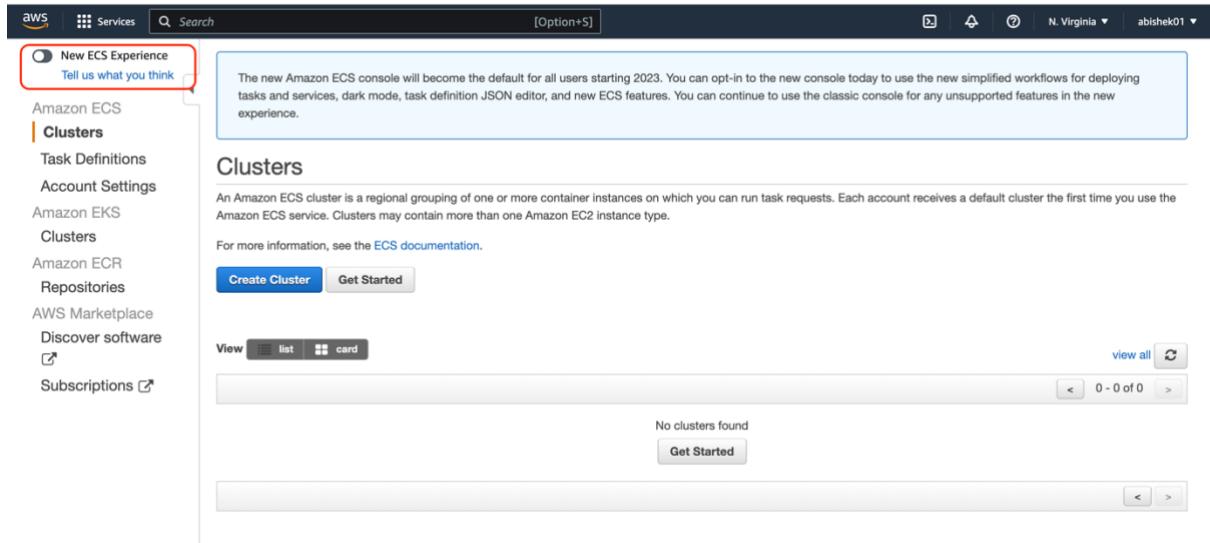
[Clear filters](#)

<input type="checkbox"/>	Policy name	Type	Description
<input type="checkbox"/>	AmazonECSTaskExecutionRolePolicy	AWS m...	Provides access to other AWS service resources that are required to ru...

3. Deployment Using AWS console without terraform.

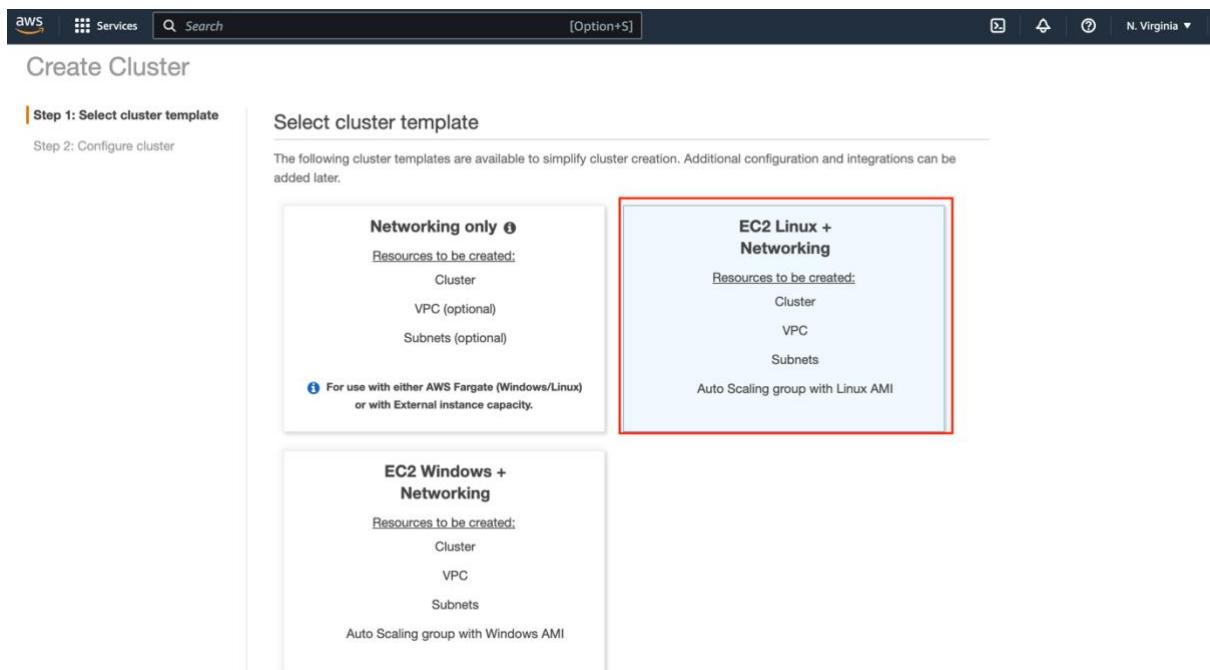
3.1. Creating a cluster.

Login to [AWS Console](#). Open Amazon Elastic Container Service and disable the “New ECS Experience” to create a cluster.



The screenshot shows the AWS ECS Clusters page. On the left, there's a sidebar with links like Task Definitions, Account Settings, Amazon EKS, Clusters, Amazon ECR, Repositories, AWS Marketplace, Discover software, and Subscriptions. A red box highlights the "New ECS Experience" link at the top of the sidebar. The main content area has a message about the new console becoming default in 2023. Below it, the "Clusters" section is titled "Clusters". It says an Amazon ECS cluster is a regional grouping of one or more container instances. There are "Create Cluster" and "Get Started" buttons. The "View" dropdown is set to "list". A message at the bottom says "No clusters found" and has a "Get Started" button.

Start creating a cluster by clicking the create cluster. Choose “EC2 Linux + Networking” as the cluster template.



The screenshot shows the "Create Cluster" wizard, Step 1: Select cluster template. It has two tabs: "Step 1: Select cluster template" (selected) and "Step 2: Configure cluster". The main area shows three cluster templates: "Networking only", "EC2 Linux + Networking" (which is highlighted with a red box), and "EC2 Windows + Networking". Each template has a description of the resources it creates and a note about compatibility with Fargate or external instance capacity.

Provide a cluster name, for instance, configuration, and use the “On-Demand Instance” as the provisioning model. EC2 instance type as “**g3.4xlarge/ g3s.xlarge**”. Number

of instances as 1. Root EBS Volume Size (GiB) as 100. For key pair, if you already have .pem key to ssh, we can use it else we can choose “None – Unable to SSH”.

*: **g3.4xlarge/ g3s.xlarge** is used because the instance has GPU which needs XAI-Service.

Instance configuration

Provisioning Model On-Demand Instance
With On-Demand Instances, you pay for compute capacity by the hour, with no long-term commitments or upfront payments.

Spot
Amazon EC2 Spot Instances let you take advantage of unused EC2 capacity in the AWS cloud. Spot Instances are available at up to a 90% discount compared to On-Demand prices.
[Learn more](#)

EC2 instance type* ⓘ Manually enter desired instance type

Number of instances* ⓘ

EC2 AMI ID* amzn2-ami-ecs-gpu-hvm-2.0.20230606-x86_64-ebs [ami-0817f4be8d3c41be4]
The GPU-optimized AMI for Amazon ECS must be used to take advantage of GPUs.
By using the GPU-optimized AMI, you agree to NVIDIA's End User License Agreement [\[?\]](#)

Root EBS Volume Size (GiB) ⓘ

Key pair ⓘ
You will not be able to SSH into your EC2 instances without a key pair. You can create a new key pair in the [EC2 console](#) [\[?\]](#).

Note: Instance- g3.4xlarge/ g3s.xlarge

Amazon EC2 G3 instances are the latest generation of Amazon EC2 GPU graphics instances that deliver a powerful combination of CPU, host memory, and GPU capacity. G3 instances are ideal for graphics-intensive applications such as 3D visualizations, mid to high-end virtual workstations, virtual application software, 3D rendering, application streaming, video encoding, gaming, and other server-side graphics workloads.

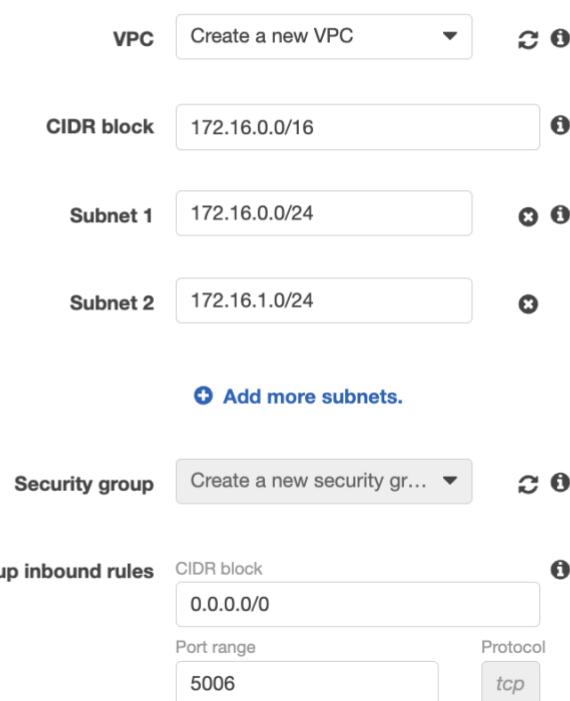
G3 instances provide access to NVIDIA Tesla M60 GPUs, each with up to 2,048 parallel processing cores, 8 GiB of GPU memory, and a hardware encoder supporting up to 10 H.265 (HEVC) 1080p30 streams and up to 18 H.264 1080p30 streams. With the latest driver releases, these GPUs provide support for OpenGL, DirectX, CUDA, OpenCL, and Capture SDK (formerly known as GRID SDK).

Name	GPUs	vCPU	Memory (GiB)	GPU Memory (GiB)	Price/hr* (Linux)
g3s.xlarge	1	4	30.5	8	\$0.75
g3.4xlarge	1	16	122	8	\$1.14

Continue to Create a cluster, in Network Configuration select vpc as “create a new vpc”. CIDR block as “172.16.0.0/16”. Subnet 1 and Subnet 2 as “172.16.0.0/24” and “172.16.1.0/24” respectively. Security group as “Create new security group”. Security group inbound rules, CIDR block “0.0.0.0/0” and **port range as 5006** depending on microservice.
 *Once the security group is created, we can give more inbound rules with all the port ranges for the xai-service microservices.

Networking

Configure the VPC for your container instances to use. A VPC is an isolated portion of the AWS cloud populated by AWS objects, such as Amazon EC2 instances. You can choose an existing VPC, or create a new one with this wizard.



The screenshot shows the AWS VPC creation wizard. It includes fields for:

- VPC:** Create a new VPC (dropdown menu)
- CIDR block:** 172.16.0.0/16
- Subnet 1:** 172.16.0.0/24
- Subnet 2:** 172.16.1.0/24
- Add more subnets:** A blue button with a plus sign.
- Security group:** Create a new security gr... (dropdown menu)
- Security group inbound rules:**
 - CIDR block: 0.0.0.0/0
 - Port range: 5006
 - Protocol: tcp

From the Container instance IAM role configuration as “ecsInstanceRole” and click “create”.

Container instance IAM role

The Amazon ECS container agent makes calls to the Amazon ECS API actions on your behalf, so container instances that run the agent require the ecsInstanceRole IAM policy and role for the service to know that the agent belongs to you. If you do not have the ecsInstanceRole already, we can create one for you.

Container instance IAM role ecsInstanceRole

For container instances to receive the new ARN and resource ID format, the root user needs to opt in for the container instance IAM role. Opt in and try again.

Tags

Key Value

Add key Add value

CloudWatch Container Insights

CloudWatch Container Insights Enable Container Insights

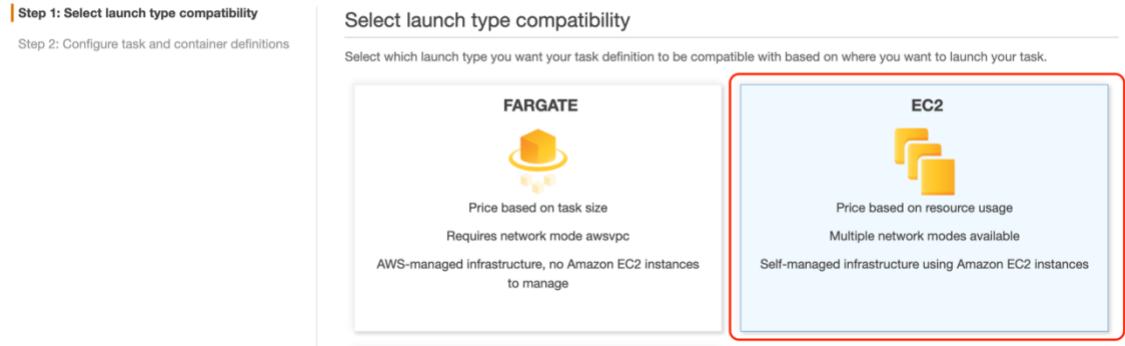
*Required Cancel Previous Create

3.2. Creating a task definition - TD.

Login to [AWS Console](#). Open Amazon [Elastic Container Service](#) and disable the “New ECS Experience”. Click the [Task Definitions](#) on the left tab of ECS to create a TD.

Click “Create a new Task Definitions” and select launch type compatibility as “EC2”.

Create new Task Definition



3.2.1. Configure task and container definitions.

1. The task definition name is “td-xai-central”.
2. Task Role is none or left untouched. [If an issue arises create an IAM role as None, better left untouched].
3. Network mode as default.

Configure task and container definitions

A task definition specifies which containers are included in your task and how they interact with each other. You can also specify data volumes for your containers to use. [Learn more](#)

Task definition name* td-xai-central i

Requires compatibilities* EC2

Task role Select a role... ↻
Optional IAM role that tasks can use to make API requests to authorized AWS services. Create an Amazon Elastic Container Service Task Role in the [IAM Console](#) i

Network mode <default> i
If you choose <default>, ECS will start your container using Docker's default networking mode, which is Bridge on Linux and NAT on Windows. Windows tasks support the <default> and awsvpc network modes.

3.2.2. Task execution IAM role

1. Task execution role as “ecsTaskExecutionRole”.

Task execution IAM role

This role is required by tasks to pull container images and publish container logs to Amazon CloudWatch on your behalf. If you do not have the ecsTaskExecutionRole already, we can create one for you.

Task execution role ecsTaskExecutionRole i

3.2.3. Container Definitions

1. Click on Add Container.

Task size

The task size allows you to specify a fixed size for your task. Task size is required for tasks using the Fargate launch type and is optional for the EC2 or External launch type. Container level memory settings are optional when task size is set. Task size is not supported for Windows containers.

Task memory (MiB)

The amount of memory (in MiB) used by the task. It can be expressed as an integer using MiB, for example 1024, or as a string using GB, for example '1GB' or '1 gb'.

Task CPU (unit)

The number of CPU units used by the task. It can be expressed as an integer using CPU units, for example 1024, or as a string using vCPUs, for example '1 vCPU' or '1 vcpu'.

Container definitions

Add container

Container ...	Image	Hard/Soft ...	CPU Unit...	GPU	Inference A...	Essential ...	
No results							

2. Use the following container configurations. Note*, we use ports 5006 in our container image exposed in 5006.

Add container

▼ Standard

Container name*

Image*

Private repository authentication*

Memory Limits (MiB)*
Hard limit
Soft limit

Define hard and/or soft memory limits in MiB for your container. Hard and soft limits correspond to the 'memory' and 'memoryReservation' parameters, respectively, in task definitions.
ECS recommends 300-500 MiB as a starting point for web applications.

Port mappings

Host port	Container port	Protocol
5006	5006	tcp

* Required

3. Please note that in the earlier project stage, we created the image using docker and the image is pushed to ECR(Only for reference for step 2).

The screenshot shows the AWS ECR console with the 'Private' tab selected. There are seven private repositories listed:

Repository name	URI	Created at	Tag immutability	Scan frequency	Encryption type	Pull through cache
backend-azure-blob	979458579914.dkr.ecr.us-east-1.amazonaws.com/backend-azure-blob	June 15, 2023, 00:07:58 (UTC-04)	Disabled	Manual	AES-256	Inactive
backend-azure-cog	979458579914.dkr.ecr.us-east-1.amazonaws.com/backend-azure-cog	June 14, 2023, 19:58:30 (UTC-04)	Disabled	Manual	AES-256	Inactive
backend-central	979458579914.dkr.ecr.us-east-1.amazonaws.com/backend-central	June 14, 2023, 23:27:18 (UTC-04)	Disabled	Manual	AES-256	Inactive
backend-evaluation_service	979458579914.dkr.ecr.us-east-1.amazonaws.com/backend-evaluation_service	June 14, 2023, 23:48:05 (UTC-04)	Disabled	Manual	AES-256	Inactive

A tooltip 'Repository URI copied' appears over the third row.

4. Click Create to create a TD.

3.3. Creating a Service.

Creating one microservice and follow the same procedure for rest of the microservices.

- STEP 1. In AWS Console, go to ECS and select the cluster created in section 3.1.
 STEP 2. Uncheck the new user experience.

The screenshot shows the AWS ECS console with the 'Clusters' tab selected. One cluster, 'ecs-xai-ec2-cluster', is listed with the following details:

Cluster	Services	Tasks	Registered container instances	CloudWatch monitoring	Capacity
ecs-xai-ec2-cluster	6	5 Pending 0 Running	1	Container Insights	No de

A red box highlights the 'New ECS Experience' button in the top left corner of the sidebar.

- STEP 3. In services tab click create.

Draining service count 0 Fargate, 0 EC2, 0 External										
Services		Tasks	ECS Instances	Metrics	Scheduled Tasks	Tags	Capacity Providers			
Create		Update	Delete	Actions ▾		Last updated on July 5, 2023 12:50:38 PM (0m ago)				
Filter in this page		Launch type	ALL	Service type	ALL					
<input type="checkbox"/>	Service Name	Status	Service ty...	Task Defini...	Desired ta...	Running ta...	Launch ty...	Platform v...		
<input type="checkbox"/>	xai-azure-cog	ACTIVE	REPLIC...	td-xai-azure...	1	0	EC2	--		
<input type="checkbox"/>	xai-central	ACTIVE	REPLIC...	td-xai-centr...	1	0	EC2	--		
<input type="checkbox"/>	xai-evaluation_service	ACTIVE	REPLIC...	td-xai-evalu...	1	0	EC2	--		
<input type="checkbox"/>	xai-azure-blob	ACTIVE	REPLIC...	td-xai-azure...	1	0	EC2	--		
<input type="checkbox"/>	xai-restnet50	ACTIVE	REPLIC...	td-xai-mod...	1	0	EC2	--		

STEP 4. Launch type as EC2.

Configure service

A service lets you specify how many copies of your task definition to run and maintain in a cluster. You can optionally use an Elastic Load Balancing load balancer to distribute incoming traffic to containers in your service. Amazon ECS maintains that number of tasks and coordinates task scheduling with the load balancer. You can also optionally use Service Auto Scaling to adjust the number of tasks in your service.

Launch type FARGATE ⓘ
 EC2 ⓘ
 EXTERNAL ⓘ

[Switch to capacity provider strategy](#) ⓘ

- STEP 5. Use the TD created in section 3.2. and use the latest revision.
- STEP 6. Use the cluster created in section 3.1.
- STEP 7. Service name is user defined.
- STEP 8. Number of tasks is 1 and rest are defaults.

Task Definition Family Enter a value

Revision ⓘ

Cluster ⓘ

Service name ⓘ

Service type* REPLICA DAEMON ⓘ

Number of tasks ⓘ

Minimum healthy percent ⓘ

Maximum percent ⓘ

Deployment circuit breaker ⓘ

Deployments

Choose a deployment option for the service.

- Deployment type* Rolling update i
- Blue/green deployment (powered by AWS CodeDeploy) i
- This sets AWS CodeDeploy as the deployment controller for the service. A CodeDeploy application and deployment group are created automatically with **default settings** for the service. To change to the rolling update deployment type after the service has been created, you must re-create the service and select the "rolling update" deployment type.

Task Placement

Lets you customize how tasks are placed on instances within your cluster. Different placement strategies are available to optimize for availability and efficiency.

Placement Templates AZ Balanced Spread Edit

This template will spread tasks across availability zones and within the availability zone spread tasks across instances. [Learn more](#)

Strategy: spread(attribute:ecs.availability-zone), spread(instanceId)

Task tagging configuration

Note: Task tagging configuration is only supported for Tasks that support tagging, which require you to be opted-in for the new Tasks ARN/ID format.

Tags

Key	Value
Add key	Add value

*Required

[Cancel](#)

[Next step](#)

STEP 9. Click next.

STEP 10. In “Configure Network” all are defaults and proceed to next.

Configure network

VPC and security groups

VPC and security groups are configurable when your task definition uses the awsvpc network mode.

Health check grace period

If your service's tasks take a while to start and respond to ELB health checks, you can specify a health check grace period of up to 2,147,483,647 seconds during which the ECS service scheduler will ignore ELB health check status. This grace period can prevent the ECS service scheduler from marking tasks as unhealthy and stopping them before they have time to come up. This is only valid if your service is configured to use a load balancer.

Health check grace period requires a load balancer.

i

STEP 11. In “Set Auto Scaling” all are defaults and proceed to next.

STEP 12. Create Service.

4. CI-CD Deployment using AWS-Terraform

4.1. Prerequisites:

4.1.1. 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.

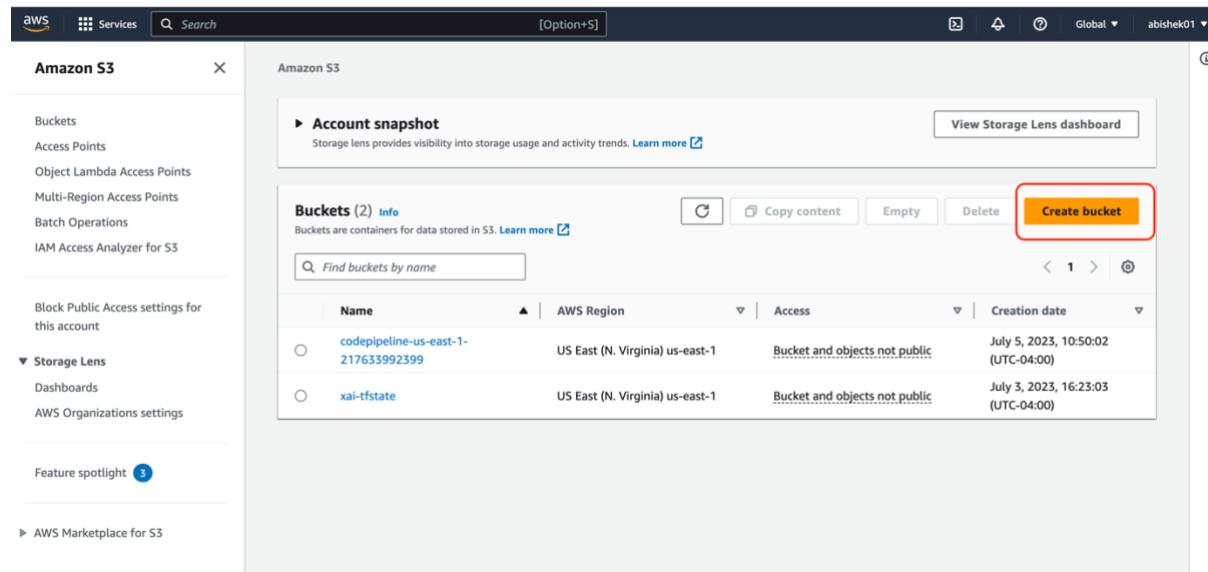
4.1.4. Connection Strings:

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

4.2. S3 Configuration:

STEP 1. In AWS, go to S3.

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



The screenshot shows the AWS S3 console. On the left, there's a navigation sidebar with options like Buckets, Access Points, Object Lambda Access Points, Multi-Region Access Points, Batch Operations, IAM Access Analyzer for S3, Block Public Access settings for this account, Storage Lens, Dashboards, and AWS Organizations settings. The main area is titled "Amazon S3" and "Account snapshot". It shows two existing buckets: "codepipeline-us-east-1-217633992399" and "xai-tfstate". Below the buckets, there's a "Create bucket" button highlighted with a red box. At the bottom of the page, there's a "Feature spotlight" section and a link to "AWS Marketplace for S3".

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

xai-tfstate

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.

The screenshot shows the 'Permissions' tab selected in the navigation bar of the AWS S3 console. Below it, the 'Permissions overview' section displays the access settings for the bucket. It indicates that 'Bucket and objects not public'.

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

The screenshot shows the 'Bucket policy' editor. A message box states: 'Public access is blocked because Block Public Access settings are turned on for this bucket'. The JSON policy code is displayed in a large text area, with a 'Copy' button available.

```
{
  "Version": "2012-10-17",
  "Id": "Policy1688416378375",
  "Statement": [
    {
      "Sid": "Stmt1688416373076",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::979458579914:root"
      },
      "Action": [
        "s3>ListBucket",
        "s3GetObject",
        "s3PutObject",
        "s3DeleteObject"
      ],
      "Resource": [
        "arn:aws:s3:::xai-tfstate",
        "arn:aws:s3:::xai-tfstate/*"
      ]
    }
  ]
}
```

```
{
  "Version": "2012-10-17",
  "Id": "Policy1688416378375",
  "Statement": [
    {
      "Sid": "Stmt1688416373076",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::<account-id>:root"
      },
      "Action": [
        "s3>ListBucket",
        "s3GetObject",
        "s3PutObject",
        "s3DeleteObject"
      ],
    }
  ]
}
```

```

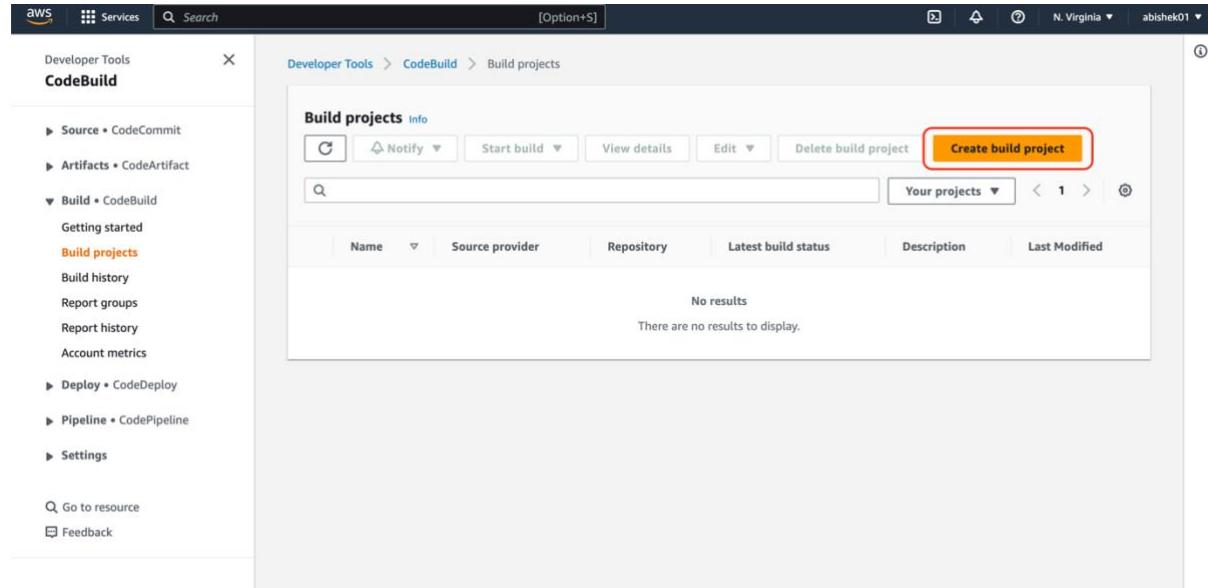
"Resource": [
    "arn:aws:s3:::xai-tfstate",
    "arn:aws:s3:::xai-tfstate/*"
]
}
]
}

```

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

Buildspec

Build specifications

Use a buildspec file
Store build commands in a YAML-formatted buildspec file

Insert build commands
Store build commands as build project configuration

Buildspec name - optional
By default, CodeBuild looks for a file named buildspec.yml in the source code root directory. If your buildspec file uses a different name or location, enter its path from the source root here (for example, buildspec-two.yml or configuration/buildspec.yml).

backend/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

Artifact 1 - Primary

Type

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.

STEP 5. Click create.

4.3. Connection String Configuration:

- STEP 1. In AWS Console, go to “Systems Manager”.
- STEP 2. Go to parameter Store.

The screenshot shows the AWS Systems Manager Parameter Store interface. The left sidebar has a red box around the 'Parameter Store' link under the 'Application Management' section. The main content area features a title 'AWS Systems Manager' with the subtitle 'Gain Operational Insight and Take Action on AWS Resources.' Below this is a 'Get Started with Systems Manager' button and a brief description of operational data. The 'How it works' section contains three icons: 'Group your resources' (with a file icon), 'View insights' (with a bar chart icon), and 'Take action' (with a laptop icon). To the right is a 'More resources' sidebar with links to Documentation, API reference, and FAQs.

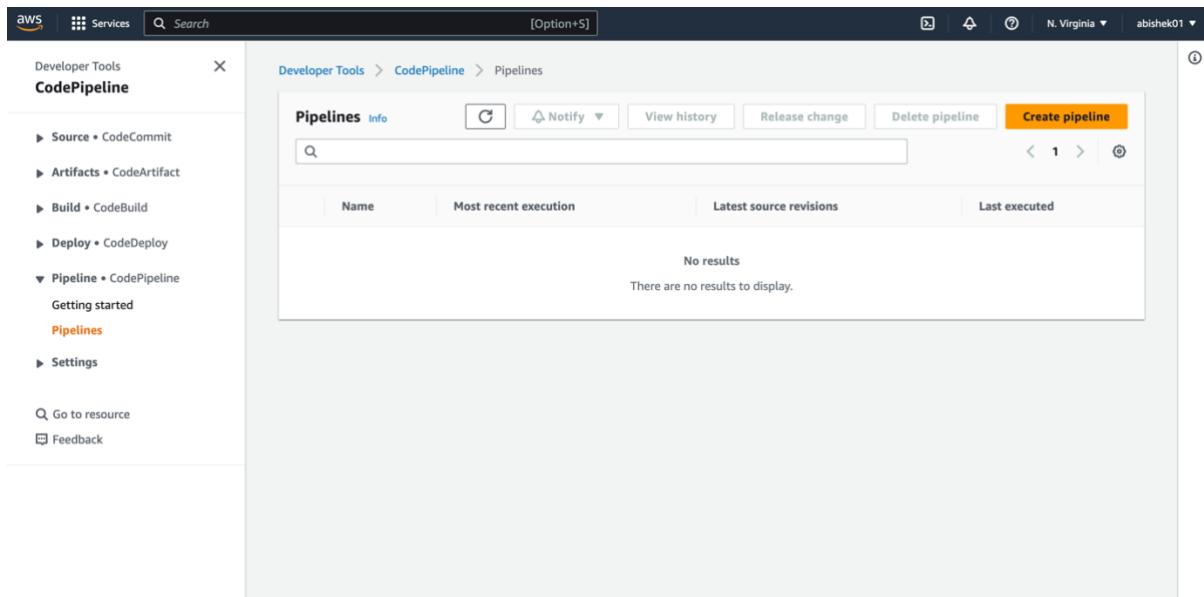
- STEP 3. Use the name of the file as “azure-con-str” and “mongo-dev” as we have configured the name in buildspec.

The screenshot shows the 'Create Parameter' form. The 'Name' field is filled with 'mongo-dev'. The 'Type' section shows 'String' selected. The 'Value' field contains 'give the value'. The 'Description — Optional' field is empty. The 'Tier' section shows 'Standard' selected, with a note about 10,000 parameters and no additional charge. The 'Advanced' tier is also shown with a note about more than 10,000 parameters and charges apply.

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.

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

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 or Connect to GitHub



Ready to connect

Your GitHub connection is ready for use.

Repository name

Choose a repository in your GitHub account.

abishekpat/XAI-Service

<account>/<repository-name>

Branch name

Choose a branch of the repository.

main

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.

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.

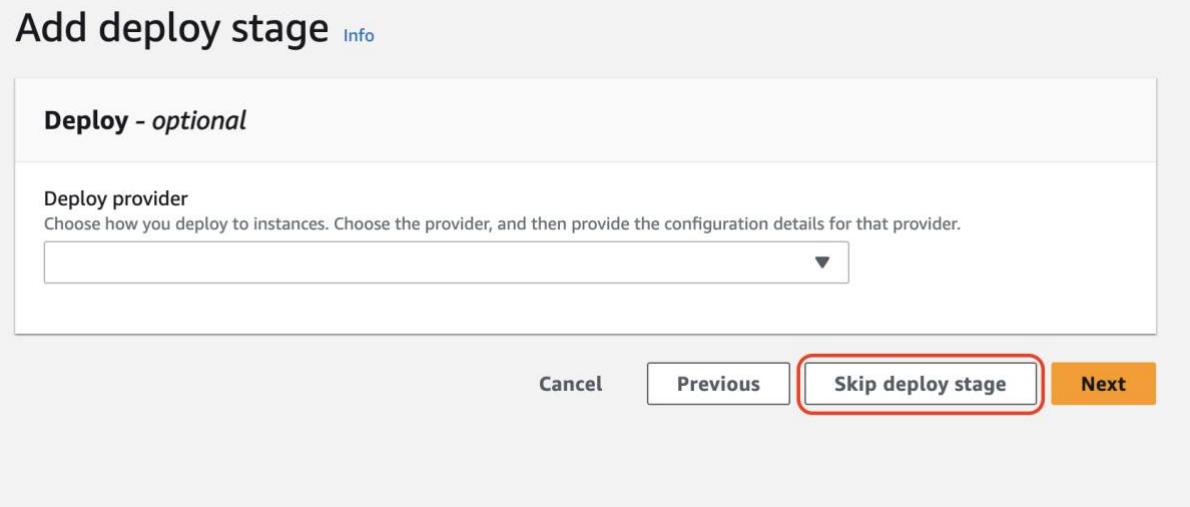
Batch build
Triggers multiple builds as a single execution.

[Cancel](#) [Previous](#) [Skip build stage](#) [Next](#)

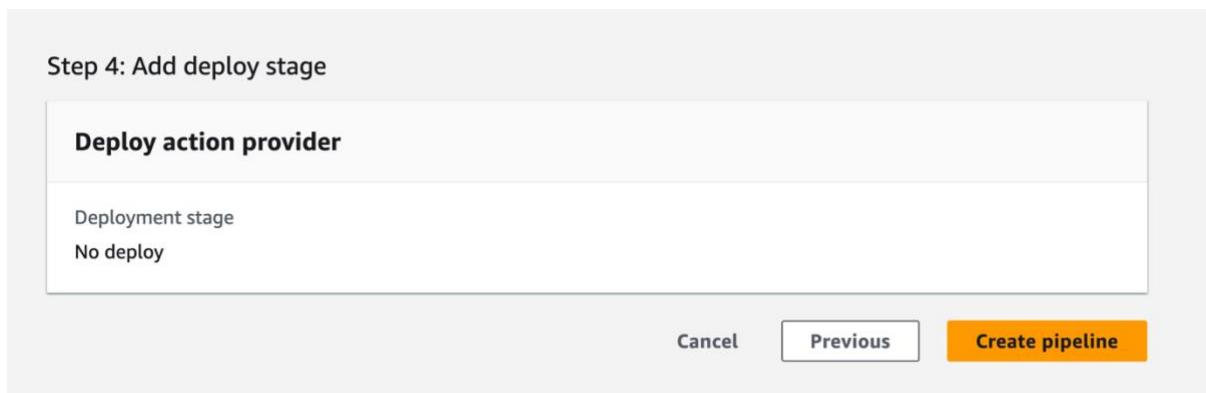
STEP 8. Add environment variables.

Build logs	Phase details	Reports	Environment variables	Build details	Resource utilization
Name	Value	Type			
DOCKER_PASSWORD	[REDACTED]	PLAINTEXT			
DOCKER_USERNAME	[REDACTED]	PLAINTEXT			
AZURE_CLIENTID	[REDACTED]	PLAINTEXT			
AZURE_PASSWORD	[REDACTED]	PLAINTEXT			
AZURE_TENANTID	[REDACTED]	PLAINTEXT			
OS_IMAGE_PASSWORD	[REDACTED]	PLAINTEXT			
REGISTRY_PASSWORD	[REDACTED]	PLAINTEXT			
AZURE_SUBSCRIPTIONID	[REDACTED]	PLAINTEXT			

STEP 9. Skip the deploy stage.



STEP 10. Click 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.