

Brain-Task-App Documentation

Step 1: Cloning the repository

- git clone <https://github.com/Vennilavan12/Brain-Tasks-App.git>

Step 2: Dockerize the application using dockfile

```
# Use a lightweight Nginx image to serve the static files.  
FROM nginx:alpine
```

```
# Copy the static files from the 'dist' directory into the Nginx web root.  
COPY ./dist /usr/share/nginx/html
```

```
# Expose port 80 to the host machine.  
EXPOSE 80
```

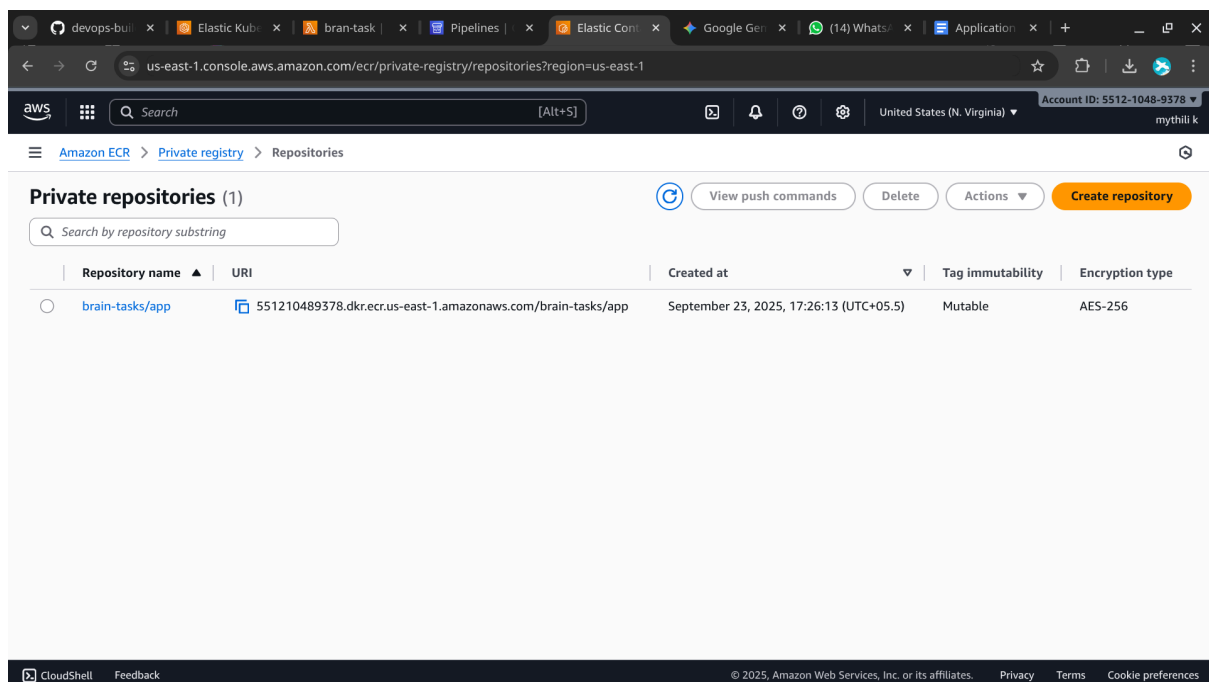
```
# The default command for the Nginx image starts the server.  
CMD ["nginx", "-g", "daemon off;"]
```

Step 3: Build and run locally

- docker build -t brain-tasks-app .
- docker run -p 80:3000 brain-tasks-app

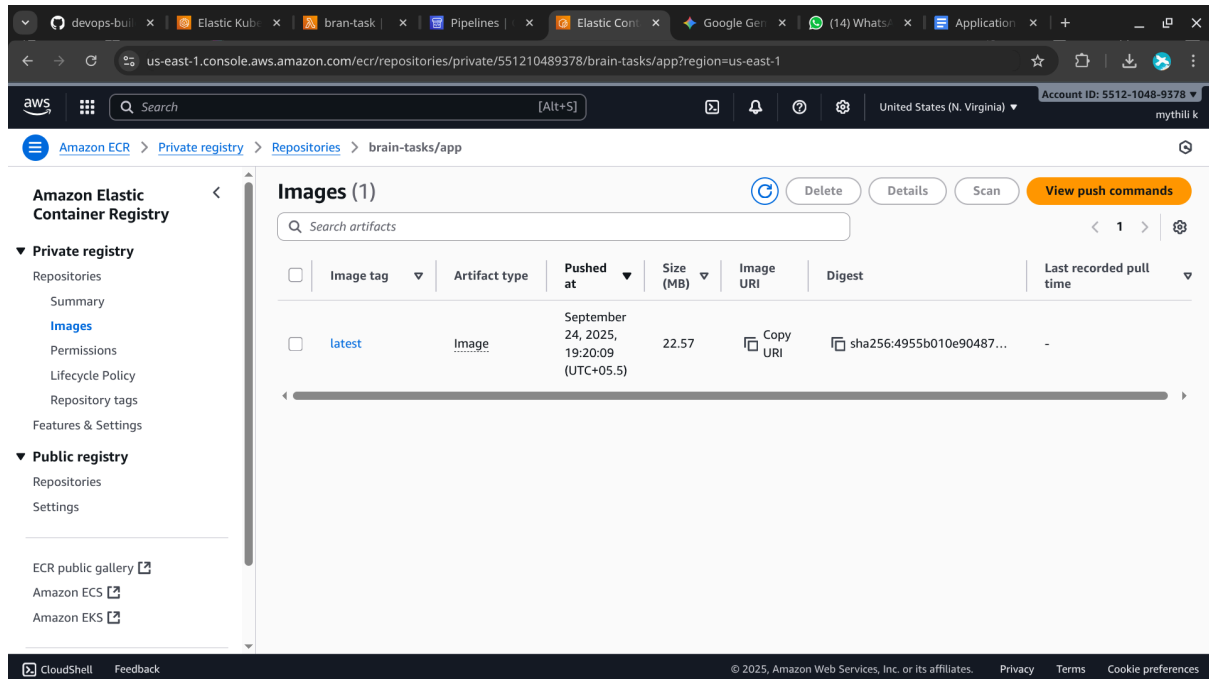
Step 4: Create a ECR repository

- aws ecr create-repository --repository-name brain-tasks-app



Step 5: Tag and push the image

- `aws ecr get-login-password --region <your-region> | docker login --username AWS --password-stdin <aws_account_id>.dkr.ecr.<region>.amazonaws.com`
- `docker tag brain-tasks-app <your-ecr-uri>:latest`
- `docker push <your-ecr-uri>:latest`



Step 6: Creation of AWS EKS cluster and verifying Cluster

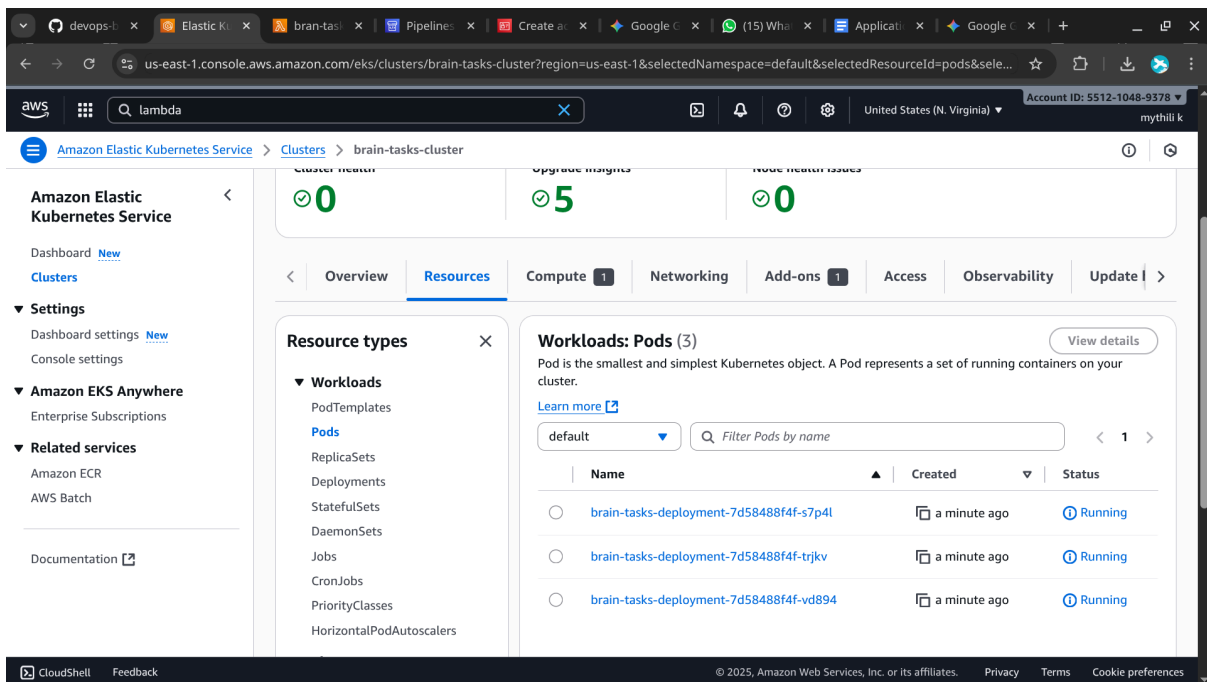
Set up an EKS cluster. Ensure the IAM role for your worker nodes has permissions to create AWS Load Balancers. The EKS cluster must also be configured to allow access from your Lambda function.

- `eksctl create cluster --name brain-tasks-cluster --region <your-region> --nodegroup-name brain-tasks-nodes --node-type t3.medium --nodes 2`

```

t configure the requested permissions; the recommended way to provide IAM permissions for "vpc-cni" add-on is via pod identity associa
tions; after add-on creation is completed, add all recommended policies to the config file, under 'addon.PodIdentityAssociations', and
run 'eksctl update addon'
2025-09-26 21:24:47 [i] creating addon: vpc-cni
2025-09-26 21:24:48 [i] successfully created addon: vpc-cni
2025-09-26 21:24:48 [i] creating addon: kube-proxy
2025-09-26 21:24:48 [i] successfully created addon: kube-proxy
2025-09-26 21:24:49 [i] creating addon: coredns
2025-09-26 21:24:49 [i] successfully created addon: coredns
2025-09-26 21:26:51 [i] building managed nodegroup stack "eksctl-my-task-cluster-nodegroup-standard-workers"
2025-09-26 21:26:51 [i] deploying stack "eksctl-my-task-cluster-nodegroup-standard-workers"
2025-09-26 21:26:51 [i] waiting for CloudFormation stack "eksctl-my-task-cluster-nodegroup-standard-workers"
2025-09-26 21:27:21 [i] waiting for CloudFormation stack "eksctl-my-task-cluster-nodegroup-standard-workers"
2025-09-26 21:28:21 [i] waiting for CloudFormation stack "eksctl-my-task-cluster-nodegroup-standard-workers"
2025-09-26 21:29:44 [i] waiting for CloudFormation stack "eksctl-my-task-cluster-nodegroup-standard-workers"
2025-09-26 21:29:44 [i] waiting for the control plane to become ready
2025-09-26 21:29:46 [✓] saved kubeconfig as "/root/.kube/config"
2025-09-26 21:29:46 [i] no tasks
2025-09-26 21:29:46 [✓] all EKS cluster resources for "my-task-cluster" have been created
2025-09-26 21:29:46 [i] nodegroup "standard-workers" has 2 node(s)
2025-09-26 21:29:46 [i] node "ip-192-168-19-224.ap-south-1.compute.internal" is ready
2025-09-26 21:29:46 [i] node "ip-192-168-58-97.ap-south-1.compute.internal" is ready
2025-09-26 21:29:46 [i] waiting for at least 2 node(s) to become ready in "standard-workers"
2025-09-26 21:29:46 [i] nodegroup "standard-workers" has 2 node(s)
2025-09-26 21:29:46 [i] node "ip-192-168-19-224.ap-south-1.compute.internal" is ready
2025-09-26 21:29:46 [i] node "ip-192-168-58-97.ap-south-1.compute.internal" is ready
2025-09-26 21:29:46 [✓] created 1 managed nodegroup(s) in cluster "my-task-cluster"
2025-09-26 21:29:48 [i] kubectl command should work with "/root/.kube/config", try 'kubectl get nodes'
2025-09-26 21:29:48 [✓] EKS cluster "my-task-cluster" in "ap-south-1" region is ready
root@fedora-bubu: /home/bubu/Pictures/Brain-Tasks-App#

```



Step 7: Kubernetes Files

deployment.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

name: brain-tasks-deployment

Spec:

replicas: 3

selector:

```

      matchLabels:
        app: brain-tasks-app
    template:
      metadata:
        labels:
          app: brain-tasks-app
      spec:
        containers:
          - name: brain-tasks-app
            image:
551210489378.dkr.ecr.us-east-1.amazonaws.com/brain-tasks/app:latest
            ports:
              - containerPort: 80

```

Service.yaml

```

apiVersion: v1
kind: Service
metadata:
  name: brain-tasks-service
  annotations:
    service.beta.kubernetes.io/aws-load-balancer-type: "external"
    service.beta.kubernetes.io/aws-load-balancer-target-group-arn:
"arn:aws:elasticloadbalancing:us-east-1:551210489378:loadbalancer/app/lb/0e93f76c2d388a
24"
  spec:
    type: LoadBalancer
    selector:
      app: brain-tasks-app
    ports:
      - protocol: TCP
        port: 80
        targetPort: 80

```

Step 8: AWS Codebuild files

- Source: Link it to this GitHub repository.
- Environment: Choose a managed image like Amazon Linux 2.
- Buildspec: Use the buildspec.yml file from the repository to define the Docker build and artifact creation process.
- Environment Variables: Define the following variables: AWS_ACCOUNT_ID, AWS_DEFAULT_REGION, and IMAGE_REPO_NAME.

buildspec.yml

version: 0.2

phases:

pre_build:

commands:

- echo "Logging in to Amazon ECR..."
- aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 551210489378.dkr.ecr.us-east-1.amazonaws.com

build:

commands:

- echo "Building the Docker image..."
- docker build -t brain-tasks-app .
- docker tag brain-tasks-app:latest 551210489378.dkr.ecr.us-east-1.amazonaws.com/brain-tasks/app:latest

post_build:

commands:

- echo "Pushing the Docker image to ECR..."
- docker push 551210489378.dkr.ecr.us-east-1.amazonaws.com/brain-tasks/app:latest
- echo "Creating imagedefinitions.json file..."
- printf '[{"name":"brain-tasks-app","imageUri":"%s"}]' 551210489378.dkr.ecr.us-east-1.amazonaws.com/brain-tasks/app:latest > imagedefinitions.json

artifacts:

files:

- 'imagedefinitions.json'
- 'kubernetes/deployment.yaml'
- 'kubernetes/service.yaml'
- 'appspec.yml'
- 'scripts/deploy.sh'

appspec.yml

version: 0.0

os: linux

hooks:

BeforeInstall:

- location: scripts/deploy.sh
- timeout: 300
- runas: root

us-east-1.console.aws.amazon.com/codesuite/codebuild/551210489378/projects/brain-task-app/build/brain-task-app%3Ad2c4229c-b9c0-41c9-88f6-c...

Developer Tools
CodeBuild

Source • CodeCommit
Artifacts • CodeArtifact
Build • CodeBuild
Getting started
Build projects
Build project
Settings
Build history
Report groups
Report history
Compute fleets [New](#)
Account metrics
Related integrations
Jenkins [New](#)
GitHub Actions [New](#)

brain-task-app:d2c4229c-b9c0-41c9-88f6-c48da880429e

Stop build Debug build Retry build

Build status

Status	Initiator	Build ARN
✓ Succeeded	codepipeline/pipe-2	arn:aws:codebuild:us-east-1:551210489378:build/brain-task-app:d2c4229c-b9c0-41c9-88f6-c48da880429e
Resolved source version	Start time	End time
e2cafe882b675b8fc3bb35c2adc96e6418aa23a	Sep 24, 2025 7:19 PM (UTC+5:30)	Sep 24, 2025 7:20 PM (UTC+5:30)
Build number		
32		

Build logs Phase details Reports Environment variables Build details Resource utilization

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

us-east-1.console.aws.amazon.com/codesuite/codebuild/551210489378/projects/brain-task-app/build/brain-task-app%3Ad2c4229c-b9c0-41c9-88f6-c...

Developer Tools
CodeBuild

Source • CodeCommit
Artifacts • CodeArtifact
Build • CodeBuild
Getting started
Build projects
Build project
Settings
Build history
Report groups
Report history
Compute fleets [New](#)
Account metrics
Related integrations
Jenkins [New](#)
GitHub Actions [New](#)

```
130 [Container] 2025/09/24 13:50:09.385933 Running command echo "Creating imagedefinitions.json file..."
131 Creating imagedefinitions.json file...
132
133 [Container] 2025/09/24 13:50:09.394885 Running command printf '[[{"name":"brain-tasks-app","imageUri":"%s"}]]'
551210489378.dkr.ecr.us-east-1.amazonaws.com/brain-tasks/app:latest > imagedefinitions.json
134
135 [Container] 2025/09/24 13:50:09.402189 Phase complete: POST_BUILD State: SUCCEEDED
136 [Container] 2025/09/24 13:50:09.402204 Phase context status code: Message:
137 [Container] 2025/09/24 13:50:09.409331 Expanding base directory path: .
138 [Container] 2025/09/24 13:50:09.409202 Assembling file list
139 [Container] 2025/09/24 13:50:09.403221 Expanding .
140 [Container] 2025/09/24 13:50:09.406764 Expanding file paths for base directory .
141 [Container] 2025/09/24 13:50:09.406781 Assembling file list
142 [Container] 2025/09/24 13:50:09.406785 Expanding imagedefinitions.json
143 [Container] 2025/09/24 13:50:09.500221 Expanding kubernetes/deployment.yaml
144 [Container] 2025/09/24 13:50:09.503710 Expanding kubernetes/service.yaml
145 [Container] 2025/09/24 13:50:09.507168 Expanding appspec.yml
146 [Container] 2025/09/24 13:50:09.510557 Expanding scripts/deploy.sh
147 [Container] 2025/09/24 13:50:09.513943 Found 5 file(s)
148 [Container] 2025/09/24 13:50:09.517049 Set report auto-discover timeout to 5 seconds
149 [Container] 2025/09/24 13:50:09.517100 Expanding base directory path: .
150 [Container] 2025/09/24 13:50:09.520520 Assembling file list
151 [Container] 2025/09/24 13:50:09.520535 Expanding .
152 [Container] 2025/09/24 13:50:09.523071 Expanding file paths for base directory .
153 [Container] 2025/09/24 13:50:09.523086 Assembling file list
154 [Container] 2025/09/24 13:50:09.523090 Expanding **/*
155 [Container] 2025/09/24 13:50:09.527607 No matching auto-discover report paths found
156 [Container] 2025/09/24 13:50:09.527631 Report auto-discover file discovery took 0.010582 seconds
157 [Container] 2025/09/24 13:50:09.527650 Phase complete: UPLOAD_ARTIFACTS State: SUCCEEDED
158 [Container] 2025/09/24 13:50:09.527661 Phase context status code: Message:
159
```

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

The screenshot shows the AWS CodeBuild console for a project named 'brain-task-app'. The build status is 'Succeeded'. The left sidebar shows the 'Build' section with 'Build project' selected. The main table lists the build steps and their durations.

Step Name	Status	Duration	Start Time	End Time
PROVISIONING	Succeeded	5 secs	Sep 24, 2025 7:19 PM (UTC+5:30)	Sep 24, 2025 7:19 PM (UTC+5:30)
DOWNLOAD_SOURCE	Succeeded	1 sec	Sep 24, 2025 7:19 PM (UTC+5:30)	Sep 24, 2025 7:19 PM (UTC+5:30)
INSTALL	Succeeded	<1 sec	Sep 24, 2025 7:19 PM (UTC+5:30)	Sep 24, 2025 7:19 PM (UTC+5:30)
PRE_BUILD	Succeeded	11 secs	Sep 24, 2025 7:19 PM (UTC+5:30)	Sep 24, 2025 7:20 PM (UTC+5:30)
BUILD	Succeeded	2 secs	Sep 24, 2025 7:20 PM (UTC+5:30)	Sep 24, 2025 7:20 PM (UTC+5:30)
POST_BUILD	Succeeded	1 sec	Sep 24, 2025 7:20 PM (UTC+5:30)	Sep 24, 2025 7:20 PM (UTC+5:30)
UPLOAD_ARTIFACTS	Succeeded	<1 sec	Sep 24, 2025 7:20 PM (UTC+5:30)	Sep 24, 2025 7:20 PM (UTC+5:30)
FINALIZING	Succeeded	<1 sec	Sep 24, 2025 7:20 PM (UTC+5:30)	Sep 24, 2025 7:20 PM (UTC+5:30)
COMPLETED	Succeeded	-	Sep 24, 2025 7:20 PM (UTC+5:30)	-

Step 9: Deployment using Lambda function

- Runtime: Python 3.9
- IAM Role: The function's execution role must have permissions for s3:GetObject on the CodePipeline artifact bucket, and full administrative access to your EKS cluster (eks:*)).

Lambda_function.py

```
import os
import subprocess
import logging

logger = logging.getLogger()
logger.setLevel(logging.INFO)

def handler(event, context):
    try:
        logger.info("Starting Lambda deployment function...")

        # Get environment variables from Lambda configuration
        cluster_name = os.environ.get("EKS_CLUSTER_NAME")
        aws_region = os.environ.get("AWS_REGION")

        # 1. Download and configure kubectl
        logger.info("Downloading and configuring kubectl...")
        subprocess.run(
```

```

["curl", "-o", "/tmp/kubectl",
"https://s3.us-west-2.amazonaws.com/amazon-eks/1.24/2022-09-21/bin/linux/amd64/kubectl
"]
)
os.chmod("/tmp/kubectl", 0o755)
os.environ["PATH"] = os.environ["PATH"] + ":/tmp"

# 2. Configure kubeconfig for EKS cluster access
logger.info("Configuring kubeconfig...")
subprocess.run([
    "aws", "eks", "update-kubeconfig",
    "--name", cluster_name,
    "--region", aws_region
], check=True)

# 3. Execute the custom deployment script from the artifacts
logger.info("Executing custom deployment script...")
subprocess.run(["chmod", "+x", "/scripts/deploy.sh"], check=True)
subprocess.run(["./scripts/deploy.sh"], check=True)

logger.info("Deployment successful.")
return {"statusCode": 200, "body": "Deployment successful"}

except subprocess.CalledProcessError as e:
    logger.error(f"Deployment failed: {e}")
    return {"statusCode": 500, "body": f"Deployment failed: {e.output}"}
except Exception as e:
    logger.error(f"An unexpected error occurred: {e}")
    return {"statusCode": 500, "body": "An unexpected error occurred"}

```

Step 10: Deploying application via Lambda

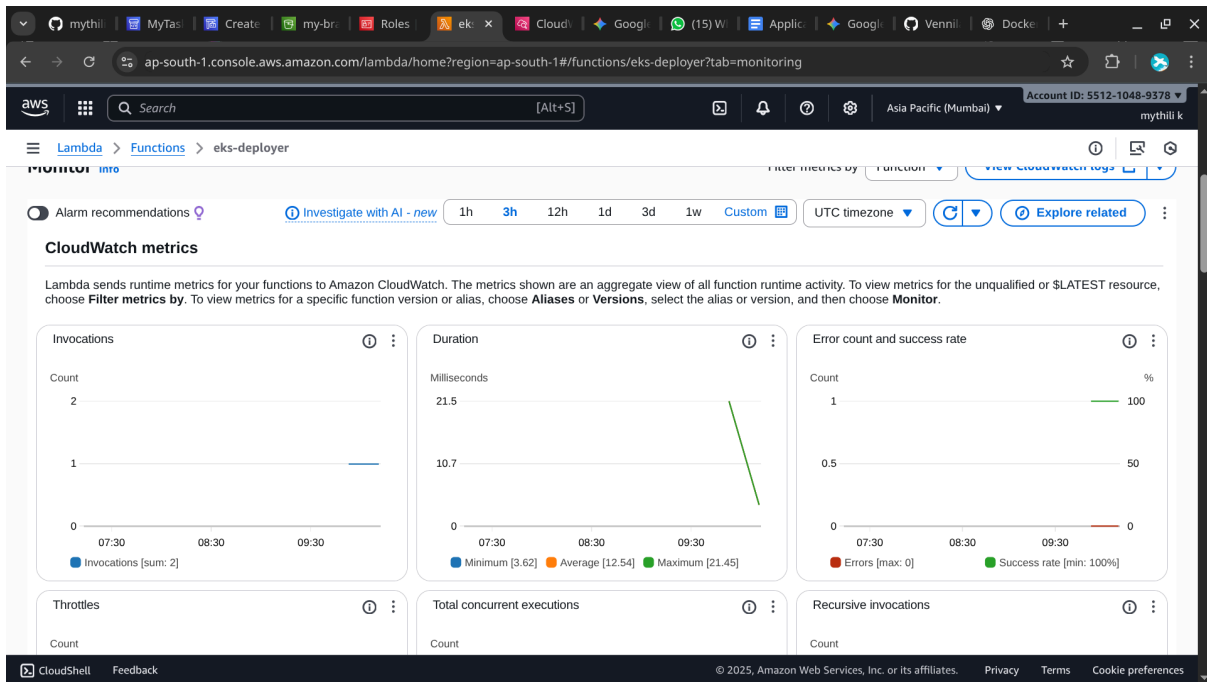
```

#!/bin/bash

# Apply Kubernetes deployment and service
kubectl apply -f kubernetes/deployment.yaml
kubectl apply -f kubernetes/service.yaml

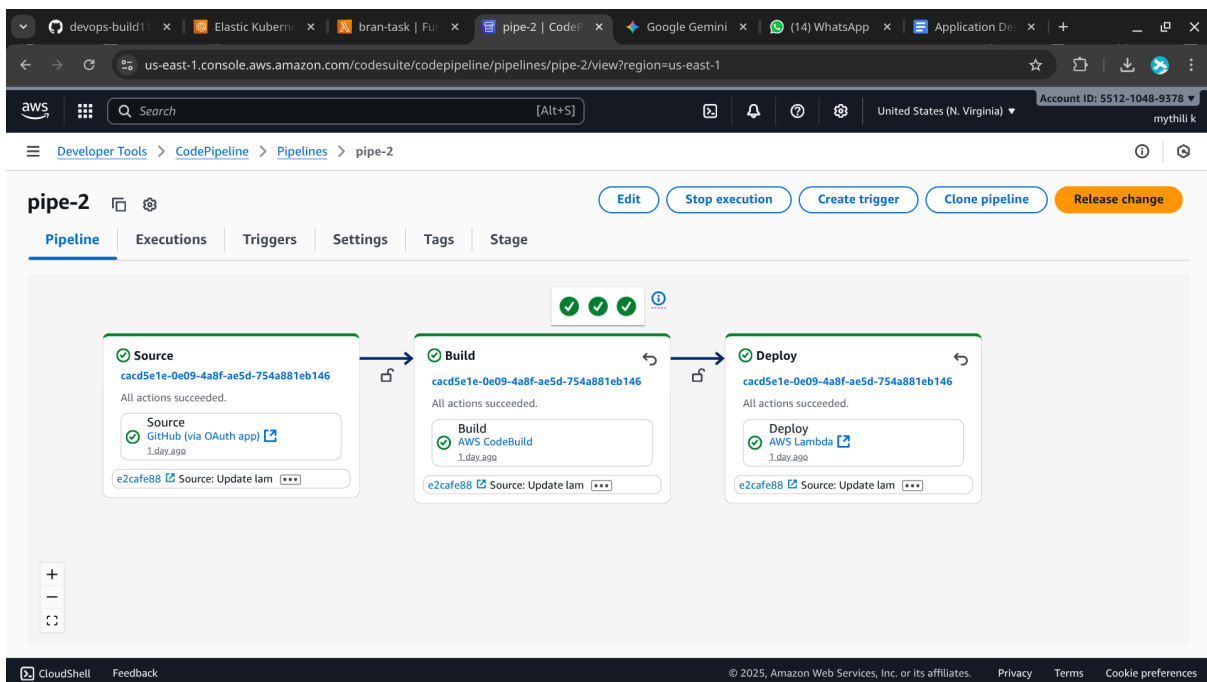
# Wait for the deployment to be ready
kubectl rollout status deployment/brain-tasks-deployment --namespace default

```

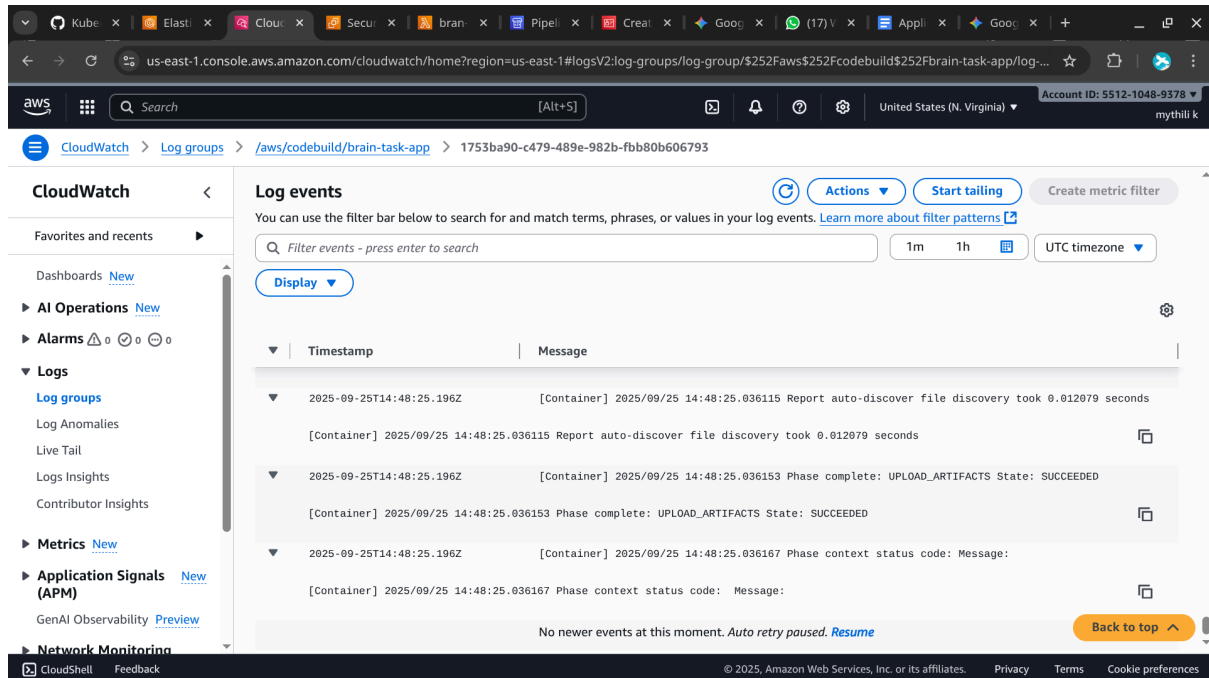
Step 11: Setting up Code-Pipeline

- Source Stage: Select GitHub and link to this repository.
- Build Stage: Choose the CodeBuild project you created.
- Deploy Stage: Select AWS Lambda as the action provider and choose the Lambda function you created. Configure the input artifact to be the output from the "Build" stage.



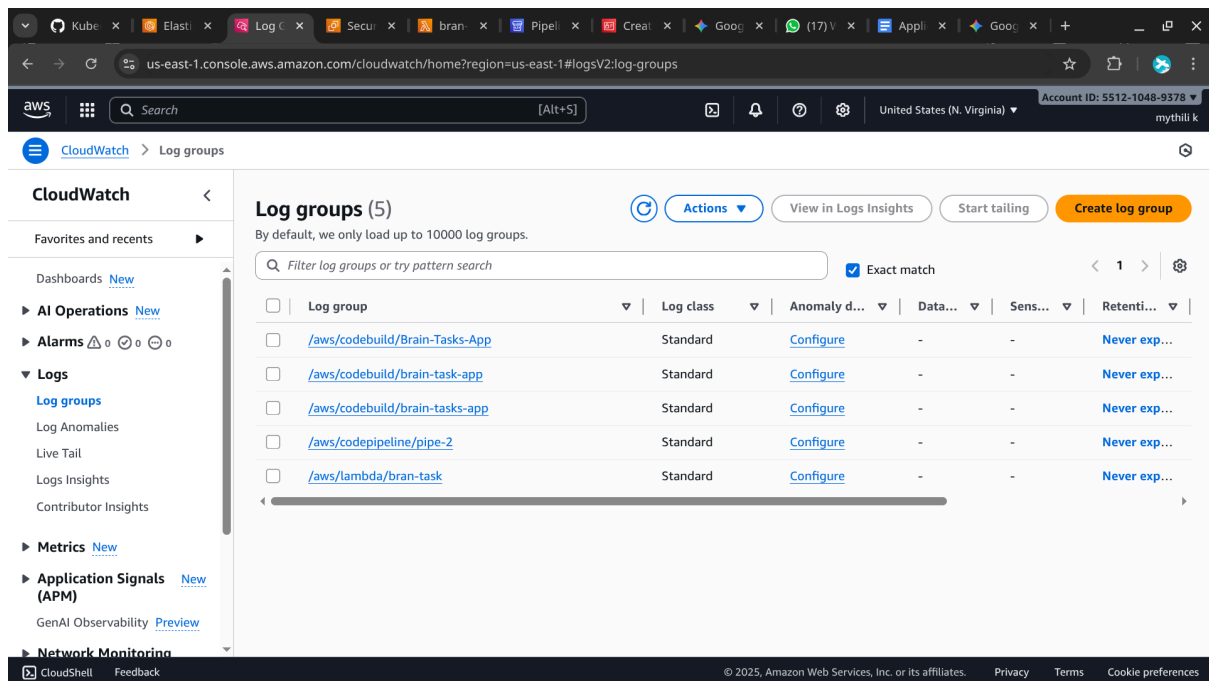
Step 12: Monitoring

- Enable logs in codebuild and lambda
- Use kubectl logs for pods logs in EKS
- Use kubectl get svc to get loadbalancer url



The screenshot shows the AWS CloudWatch console with the 'Log events' view selected. The left sidebar contains navigation options like 'Dashboards', 'AI Operations', 'Alarms', 'Logs', 'Metrics', 'Application Signals (APM)', and 'Network Monitoring'. The main content area displays a list of log events with columns for 'Timestamp' and 'Message'. The events show the completion of the 'UPLOAD_ARTIFACTS' phase for a CodeBuild task. The bottom of the page includes a footer with copyright information and links to 'Privacy', 'Terms', and 'Cookie preferences'.

Timestamp	Message
2025-09-25T14:48:25.196Z	[Container] 2025/09/25 14:48:25.036115 Report auto-discover file discovery took 0.012079 seconds
2025-09-25T14:48:25.196Z	[Container] 2025/09/25 14:48:25.036115 Report auto-discover file discovery took 0.012079 seconds
2025-09-25T14:48:25.196Z	[Container] 2025/09/25 14:48:25.036153 Phase complete: UPLOAD_ARTIFACTS State: SUCCEEDED
2025-09-25T14:48:25.196Z	[Container] 2025/09/25 14:48:25.036153 Phase complete: UPLOAD_ARTIFACTS State: SUCCEEDED
2025-09-25T14:48:25.196Z	[Container] 2025/09/25 14:48:25.036167 Phase context status code: Message:
2025-09-25T14:48:25.196Z	[Container] 2025/09/25 14:48:25.036167 Phase context status code: Message:



The screenshot shows the AWS CloudWatch console with the 'Log groups (5)' view selected. The left sidebar is the same as the previous screenshot. The main content area displays a list of log groups with columns for 'Log group', 'Log class', 'Anomaly d...', 'Data...', 'Sens...', and 'Retenti...'. The log groups are listed with their respective classes and configurations. The bottom of the page includes a footer with copyright information and links to 'Privacy', 'Terms', and 'Cookie preferences'.

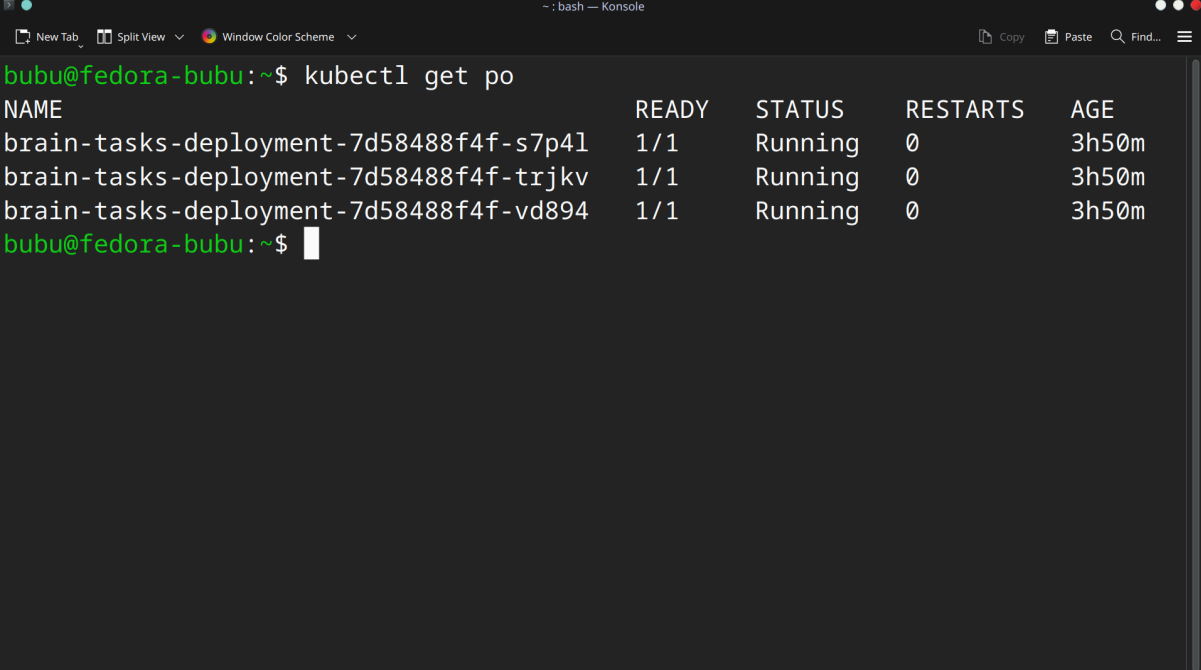
Log group	Log class	Anomaly d...	Data...	Sens...	Retenti...
/aws/codebuild/Brain-Tasks-App	Standard	Configure	-	-	Never exp...
/aws/codebuild/brain-task-app	Standard	Configure	-	-	Never exp...
/aws/codebuild/brain-task-app	Standard	Configure	-	-	Never exp...
/aws/codepipeline/pipe-2	Standard	Configure	-	-	Never exp...
/aws/lambda/bran-task	Standard	Configure	-	-	Never exp...

Step 13: Git Commands

- `git init`
- `git remote add origin https://github.com/yourusername/brain-tasks-app-deploy.git`
- `git add .`
- `git commit -m "Initial commit with deployment"`
- `git push -u origin main`

Step 14: Output

After a successful deployment, the AWS Load Balancer ARN for the application can be found in the EC2 console.

A terminal window titled "Konsole" with a dark background. The prompt is "bubu@fedora-bubu:~\$". The command "kubectl get po" has been executed, resulting in a table of pods. The table has five columns: NAME, READY, STATUS, RESTARTS, and AGE. There are three rows of data, all showing pods in a 'Running' state with 0 restarts and an age of 3h50m. The terminal window includes standard UI elements like a title bar, tabs, and a search bar.

```
bubu@fedora-bubu:~$ kubectl get po
NAME                                READY   STATUS    RESTARTS   AGE
brain-tasks-deployment-7d58488f4f-s7p4l  1/1     Running   0           3h50m
brain-tasks-deployment-7d58488f4f-trjvk  1/1     Running   0           3h50m
brain-tasks-deployment-7d58488f4f-vd894  1/1     Running   0           3h50m
bubu@fedora-bubu:~$
```

```
~ : bash — Konsole
New Tab Split View Window Color Scheme Copy Paste Find...
bubu@fedora-bubu:~$ kubectl get svc
NAME                                TYPE        CLUSTER-IP      EXTERNAL-IP
brain-tasks-service                LoadBalancer  10.100.11.106   ae773b9e401834694b69f66e8d12f3ab-83a60a31650ca33d.elb.us-east-1.amazonaws.com
kubernetes                         ClusterIP     10.100.0.1      <none>
443/TCP                           2d5h
```

