Project: Design and Implement a CI/CD pipeline using GitHub Actions

April 27 2024

Dahi Nemutlu

UDACITY

Contents

Introduction

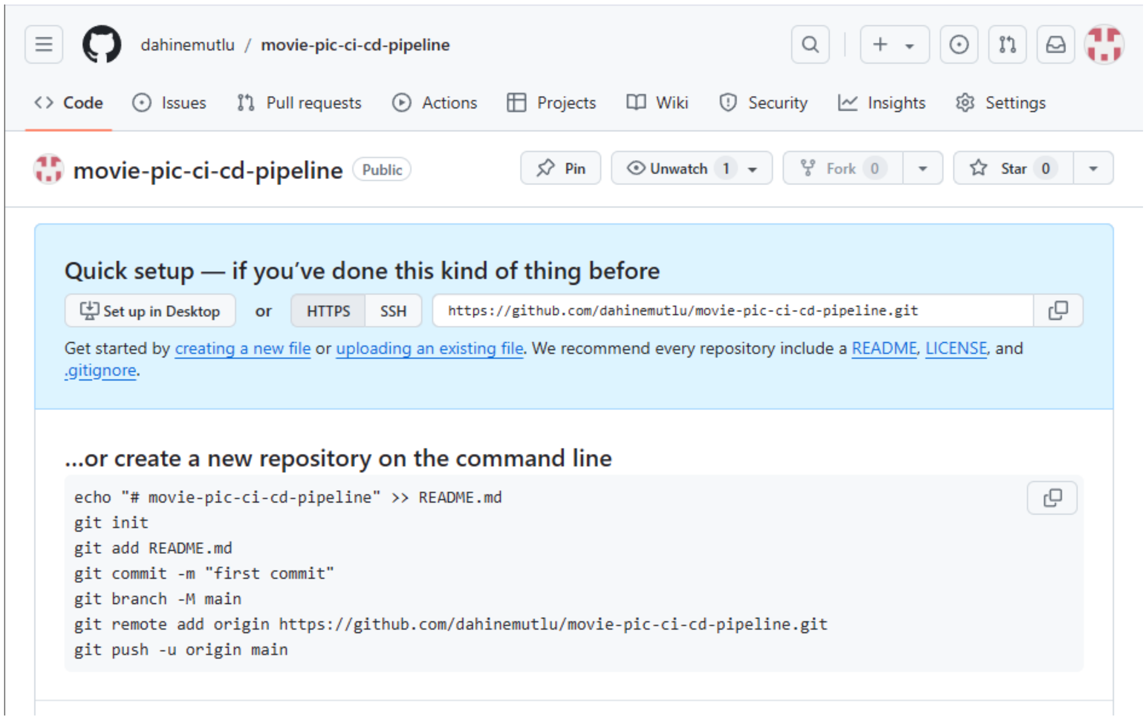
A web application that is a catalog of Movie Picture movies is comprised of two applications:

- A frontend UI built written in Typescript, using the React framework.
- A backend API written in Python using the Flask framework.

This project aims to design and implement CI/CD pipelines using GitHub Actions to automate the development team's workflows for this application. Therefore, we will take the following steps.

- Create GitHub Actions workflows that performs the necessary Continuous Integration steps on the frontend and backend applications:
    - lint
    - build
    - test
- Create the necessary AWS resources/infrastructure with Terraform.
- Create GitHub Actions workflows so that the pipeline deploys the frontend and backend apps to the Kubernetes cluster.

Now let's create a new GitHub repository, initialize our local Git repository, and push the project files to the new GitHub repository.



Quick setup — if you've done this kind of thing before

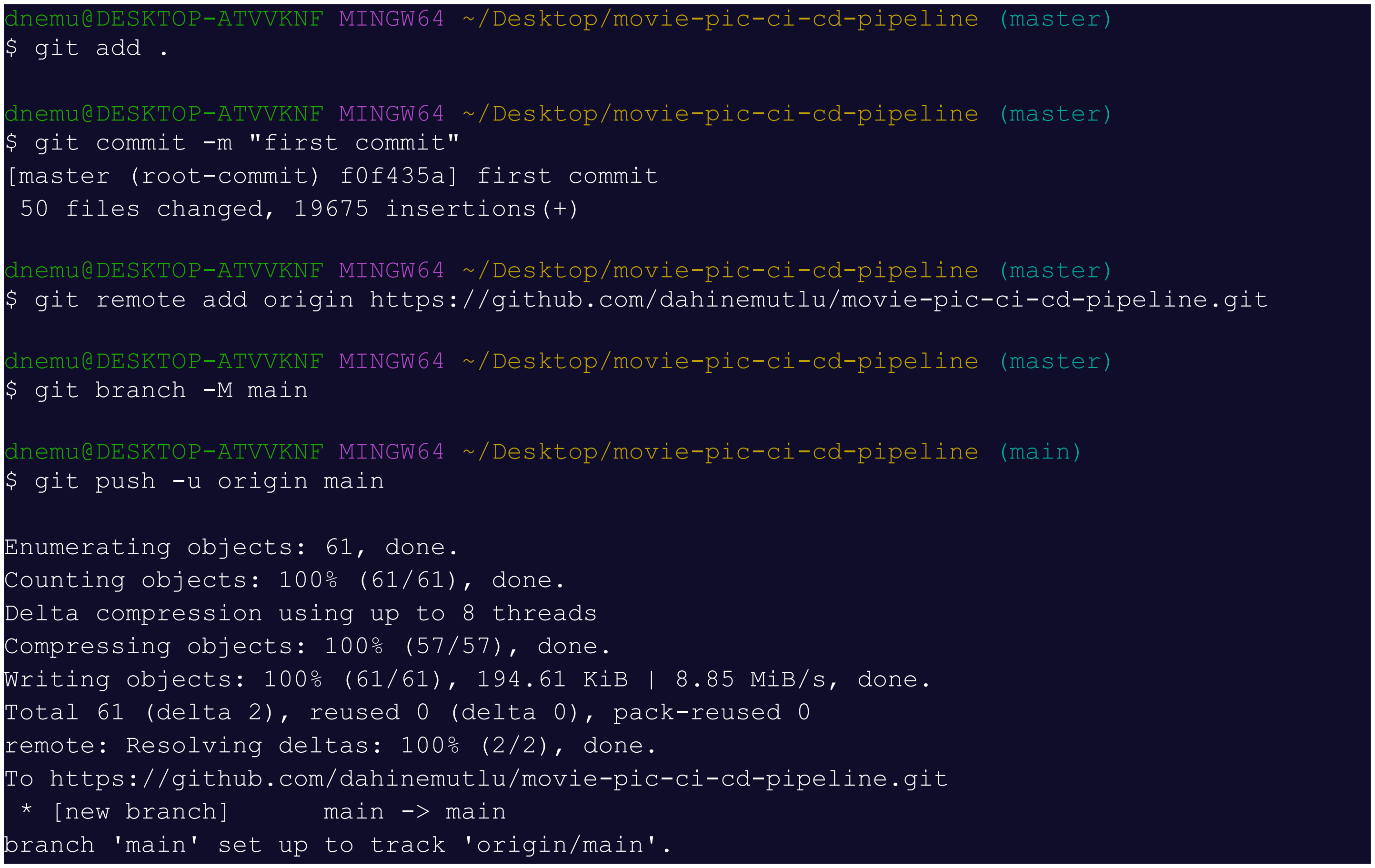
https://github.com/dahinemutlu/movie-pic-ci-cd-pipeline.git

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

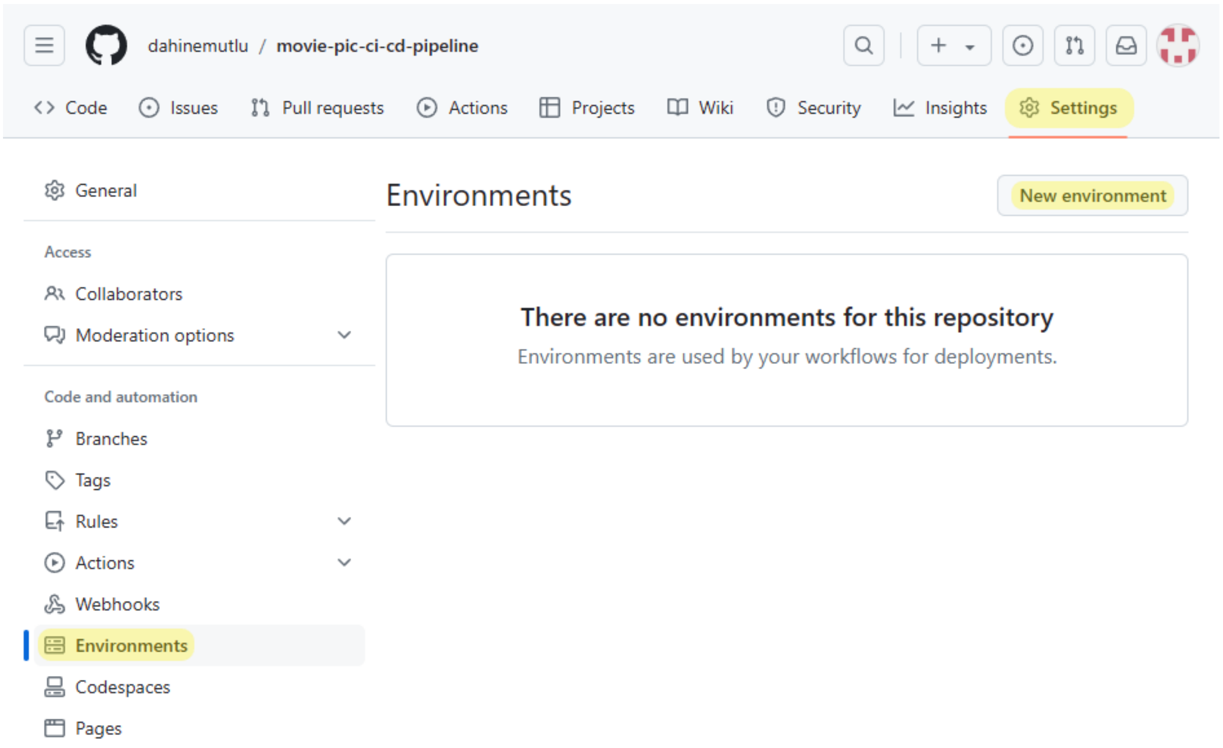
```
echo "# movie-pic-ci-cd-pipeline" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/dahinemutlu/movie-pic-ci-cd-pipeline.git
git push -u origin main
```

```
dnemu@DESKTOP-ATVVKNF MINGW64 ~/Desktop/movie-pic-ci-cd-pipeline
$ git init
Initialized empty Git repository in C:/Users/dnemu/Desktop/movie-pic-ci-cd-pipeline/.git/
```



```
dnemu@DESKTOP-ATVVKNF MINGW64 ~/Desktop/movie-pic-ci-cd-pipeline (master)
$ git add .

dnemu@DESKTOP-ATVVKNF MINGW64 ~/Desktop/movie-pic-ci-cd-pipeline (master)
$ git commit -m "first commit"
[master (root-commit) f0f435a] first commit
 50 files changed, 19675 insertions(+)

dnemu@DESKTOP-ATVVKNF MINGW64 ~/Desktop/movie-pic-ci-cd-pipeline (master)
$ git remote add origin https://github.com/dahinemutlu/movie-pic-ci-cd-pipeline.git

dnemu@DESKTOP-ATVVKNF MINGW64 ~/Desktop/movie-pic-ci-cd-pipeline (master)
$ git branch -M main

dnemu@DESKTOP-ATVVKNF MINGW64 ~/Desktop/movie-pic-ci-cd-pipeline (main)
$ git push -u origin main

Enumerating objects: 61, done.
Counting objects: 100% (61/61), done.
Delta compression using up to 8 threads
Compressing objects: 100% (57/57), done.
Writing objects: 100% (61/61), 194.61 KiB | 8.85 MiB/s, done.
Total 61 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/dahinemutlu/movie-pic-ci-cd-pipeline.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

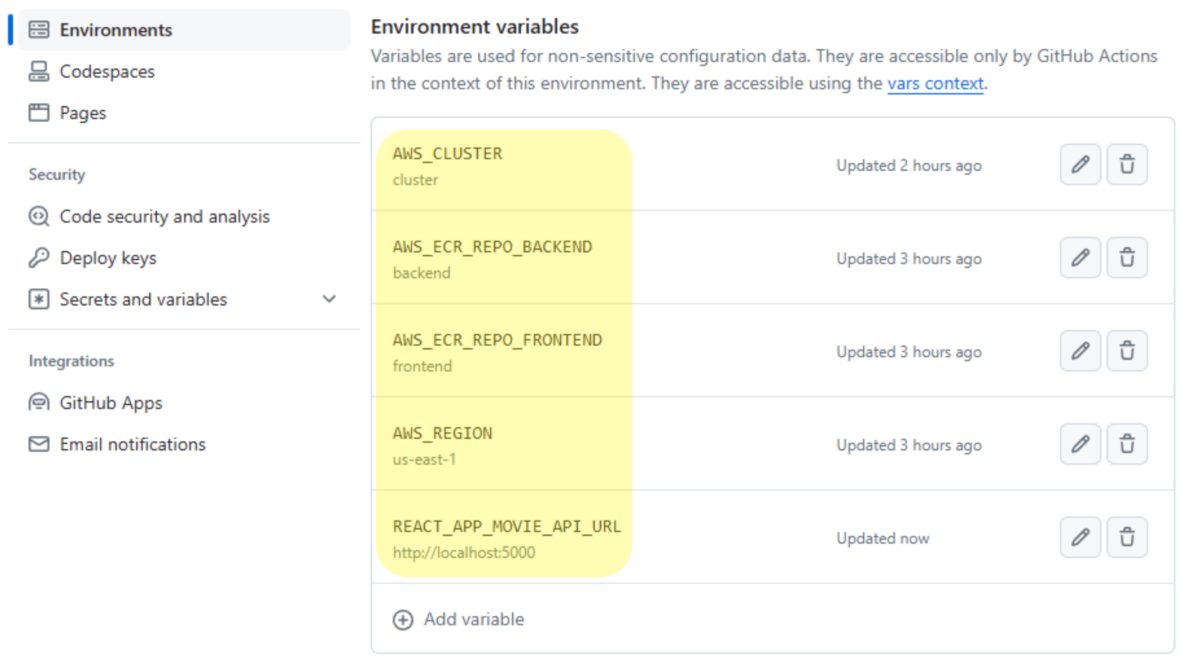
Create an environment in the repository and configure the following environment variables:



Environments

There are no environments for this repository

Environments are used by your workflows for deployments.



Environment variables

Variables are used for non-sensitive configuration data. They are accessible only by GitHub Actions in the context of this environment. They are accessible using the vars context.

AWS_CLUSTER	cluster	Updated 2 hours ago
AWS_ECR_REPO_BACKEND	backend	Updated 3 hours ago
AWS_ECR_REPO_FRONTEND	frontend	Updated 3 hours ago
AWS_REGION	us-east-1	Updated 3 hours ago
REACT_APP_MOVIE_API_URL	http://localhost:5000	Updated now

We will refer to these environment variables in the deployment workflows that we will create in the following sections.

Continuous Integration workflow for Frontend

Create the file **.github/workflows/frontend-ci.yml** file that:

- Is triggered on **pull_requests** events against the **main** branch,
    - Only when code in the frontend application changes.
- Can be run on-demand (i.e. manually without needing to push code)
- Runs the following tasks in parallel:
    - Runs a linting job that fails if the code doesn't adhere to eslint rules.
    - Runs a test job that fails if the test suite doesn't pass.
- Runs a build job only if the lint and test jobs succeed.

```
name: Frontend CI

on:
  # The workflow should run on manual dispatch (workflow_dispatch)
  # and whenever a pull request is opened or updated against the main branch.
  workflow_dispatch:
  pull_request:
    branches:
      - main

# Set default working directory as 'frontend'
# for all the subsequent run steps in the jobs.
defaults:
```

```yaml
  run:
    working-directory: frontend

jobs:
  Lint:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout
        uses: actions/checkout@v4
      - name: Setup Node.js environment
        uses: actions/setup-node@v4
        with:
          node-version: latest
          cache: "npm"
          cache-dependency-path: frontend/package-lock.json
      - name: Install dependencies
        run: npm ci
      - name: Run linter
        run: npm run lint

  Test:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout
        uses: actions/checkout@v4
      - name: Setup Node.js environment
        uses: actions/setup-node@v4
        with:
          node-version: latest
          cache: "npm"
          cache-dependency-path: frontend/package-lock.json
      - name: Install dependencies
        run: npm ci
      - name: Run the tests
        run: CI=true npm test

  Build:
    # Ensure that the linting and testing steps are completed successfully
    # before attempting the building step.
    needs: [Lint, Test]
    runs-on: ubuntu-latest
    steps:
      - name: Checkout
        uses: actions/checkout@v4
      - name: Setup Node.js environment
        uses: actions/setup-node@v4
        with:
          node-version: latest
          cache: "npm"
          cache-dependency-path: frontend/package-lock.json
      - name: Install dependencies
        run: npm ci
      - name: Build docker image
        run: docker build --build-arg=REACT_APP_MOVIE_API_URL=http://localhost:5000 --tag=mp-
frontend:latest .
      - name: Run container
        run: docker run --name mp-frontend -p 3000:3000 -d mp-frontend
```

Continuous Integration workflow for Backend

Create the file **.github/workflows/backend-ci.yml** file that:

- Runs on **pull_requests** against the **main** branch, only when code in the frontend application changes.
- Can be run on-demand (i.e., manually without needing to push code)
- Runs the following jobs in parallel:
    - Runs a linting job that fails if the code doesn't adhere to eslint rules.
    - Runs a test job that fails if the test suite doesn't pass.
- Runs a build job only if the lint and test jobs pass and successfully builds the application.

```
name: Backend CI

on:
  workflow_dispatch:
  pull_request:
    branches:
      - main

defaults:
  run:
    working-directory: backend

jobs:
  Lint:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout
        uses: actions/checkout@v4
      - name: Setup Python
        uses: actions/setup-python@v5
        with:
          python-version: "3.10"
      - name: Install dependencies
```

```yaml
        run: |
          python -m pip install --upgrade pip
          pip install flake8
          pip install pipenv
          pipenv install
      - name: Run linter
        run: pipenv run lint

  Test:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout
        uses: actions/checkout@v4
      - name: Setup Python
        uses: actions/setup-python@v5
        with:
          python-version: "3.10"
      - name: Install dependencies
        run: |
          python -m pip install --upgrade pip
          pip install pipenv
          pipenv install
      - name: Run the tests
        run: pipenv run test

  Build:
    needs: [Lint, Test]
    runs-on: ubuntu-latest
    steps:
      - name: Checkout
        uses: actions/checkout@v4
      - name: Setup Python
        uses: actions/setup-python@v5
        with:
          python-version: "3.10"
      - name: Install dependencies
        run: |
          python -m pip install --upgrade pip
          pip install pipenv
          pipenv install
      - name: Build docker image
        run: docker build --tag mp-backend:latest .
      - name: Run container
        run: docker run -p 5000:5000 --name mp-backend -d mp-backend
```

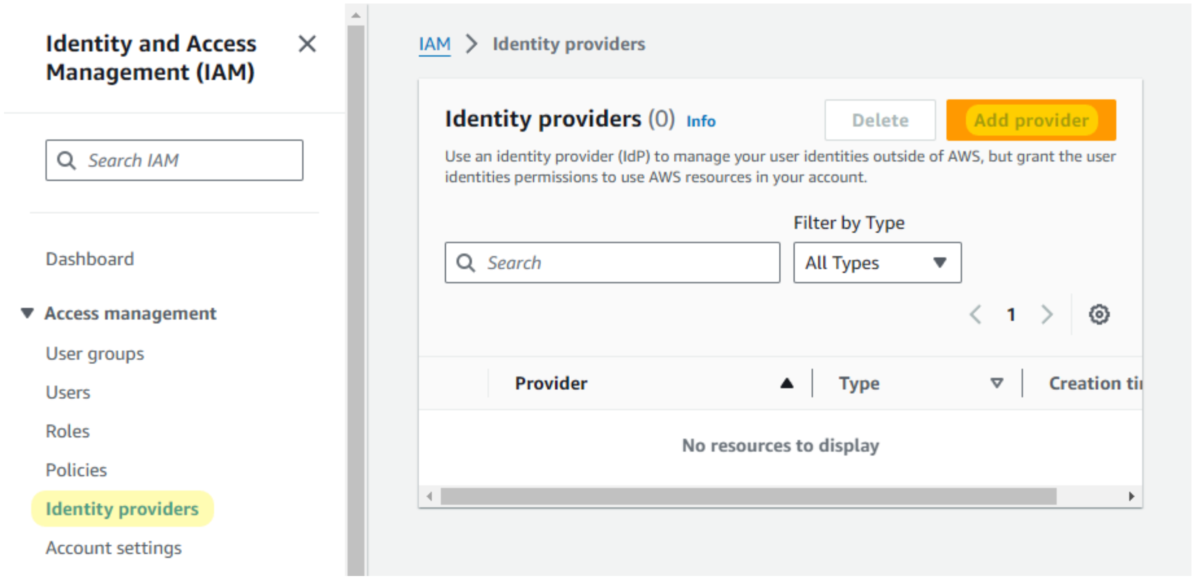
Configure OpenID Connect in Amazon Web Services

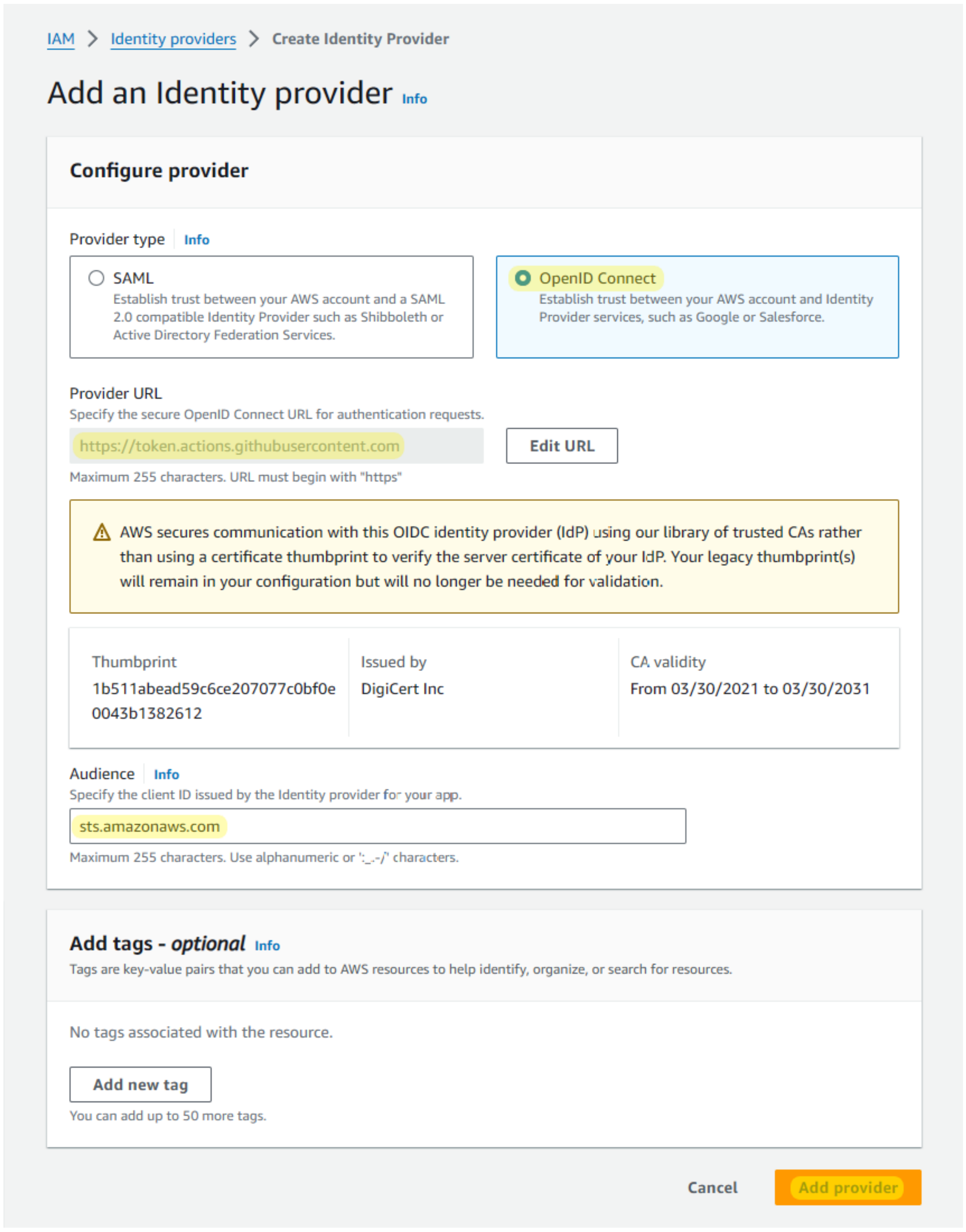
We will use OpenID Connect within our CD workflows to authenticate with and access resources in Amazon Web Services (AWS), without needing to store the AWS credentials as long-lived GitHub secrets.

Add identity provider to AWS

Add the GitHub OIDC provider to AWS IAM.

- For the Provider URL: https://token.actions.githubusercontent.com
- For the Audience: sts.amazonaws.com


Identity and Access Management (IAM)
Search IAM
Dashboard
Access management
User groups
Users
Roles
Policies
Identity providers
Account settings
IAM > Identity providers
Identity providers (0) Info
Delete
Add provider
Use an identity provider (IdP) to manage your user identities outside of AWS, but grant the user identities permissions to use AWS resources in your account.
Filter by Type
Search
All Types
1
Provider
Type
Creation ti
No resources to display



IAM > Identity providers > Create Identity Provider

Add an Identity provider Info

Configure provider

Provider type | Info

( ) SAML
Establish trust between your AWS account and a SAML 2.0 compatible Identity Provider such as Shibboleth or Active Directory Federation Services.

[✓ OpenID Connect]
Establish trust between your AWS account and Identity Provider services, such as Google or Salesforce.

Provider URL
Specify the secure OpenID Connect URL for authentication requests.
https://token.actions.githubusercontent.com
Edit URL
Maximum 255 characters. URL must begin with "https"

AWS secures communication with this OIDC identity provider (IdP) using our library of trusted CAs rather than using a certificate thumbprint to verify the server certificate of your IdP. Your legacy thumbprint(s) will remain in your configuration but will no longer be needed for validation.

Thumbprint	Issued by	CA validity
1b511abead59c6ce207077c0bf0e0043b1382612	DigiCert Inc	From 03/30/2021 to 03/30/2031

Audience | Info
Specify the client ID issued by the Identity provider for your app.
sts.amazonaws.com
Maximum 255 characters. Use alphanumeric or ':_.-/' characters.

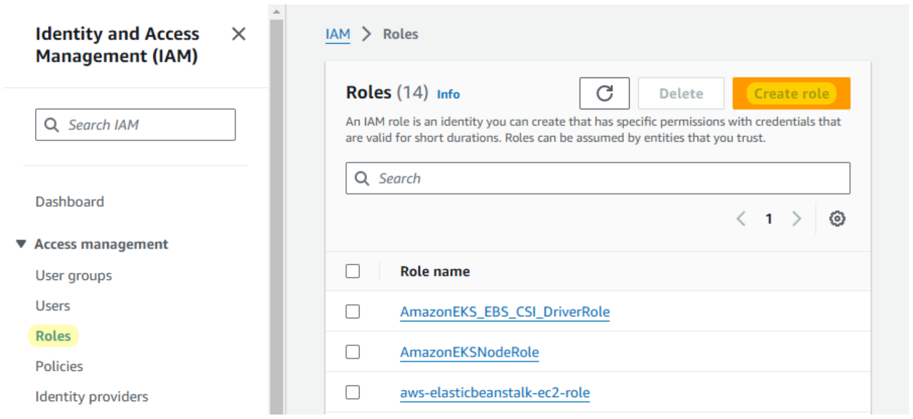
Add tags - *optional* Info
Tags are key-value pairs that you can add to AWS resources to help identify, organize, or search for resources.

No tags associated with the resource.

Add new tag
You can add up to 50 more tags.

Cancel   Add provider

Configure a role for GitHub OIDC identity provider



Identity and Access Management (IAM)

Search IAM

Dashboard

Access management

- User groups
- Users
- Roles
- Policies
- Identity providers

IAM > Roles

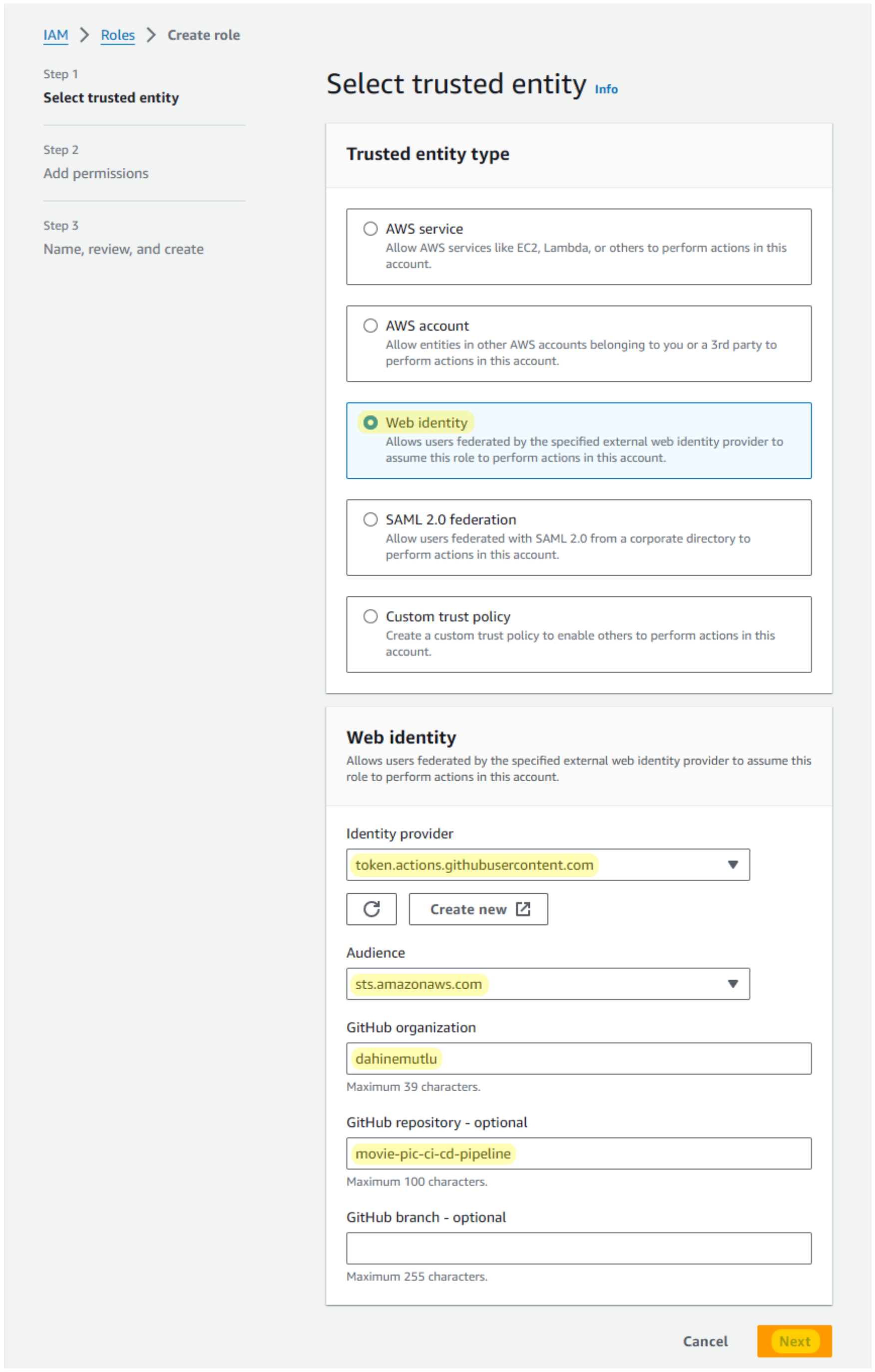
Roles (14) Info

Delete | Create role

An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

Search

Role name
AmazonEKS_EBS_CSI_DriverRole
AmazonEKSNodeRole
aws-elasticbeanstalk-ec2-role



IAM > Roles > Create role

Step 1
Select trusted entity

Step 2
Add permissions

Step 3
Name, review, and create

Select trusted entity Info

Trusted entity type

- ( ) AWS service
  Allow AWS services like EC2, Lambda, or others to perform actions in this account.
- ( ) AWS account
  Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
- [✓ Web identity]
  Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.
- ( ) SAML 2.0 federation
  Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.
- ( ) Custom trust policy
  Create a custom trust policy to enable others to perform actions in this account.

Web identity

Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.

Identity provider: token.actions.githubusercontent.com

Create new

Audience: sts.amazonaws.com

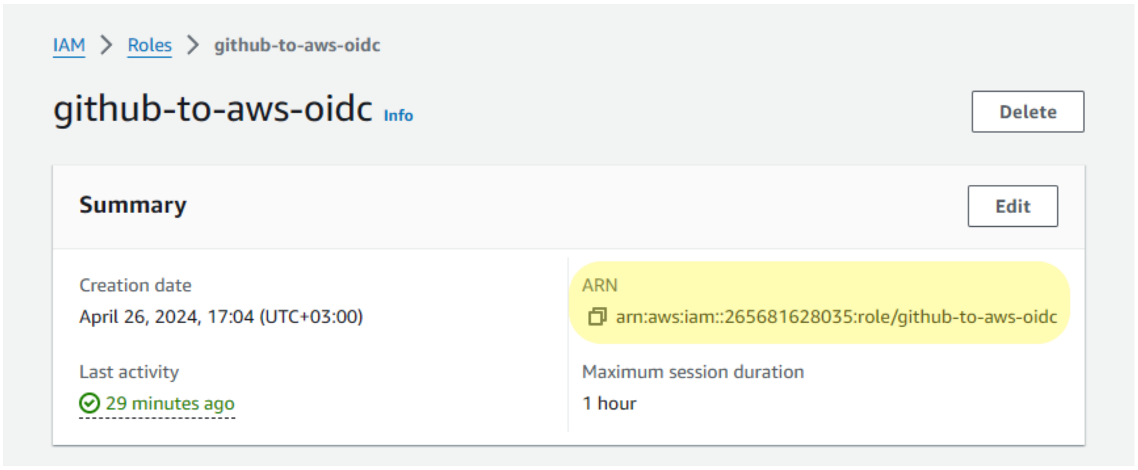
GitHub organization: dahinemutlu
Maximum 39 characters.

GitHub repository - optional: movie-pic-ci-cd-pipeline
Maximum 100 characters.

GitHub branch - optional:
Maximum 255 characters.

Cancel | Next

Make a note of this role's ARN to configure it later in the workflow.



IAM > Roles > github-to-aws-oidc

github-to-aws-oidc Info

Summary

Creation date
April 26, 2024, 17:04 (UTC+03:00)

ARN
arn:aws:iam::265681628035:role/github-to-aws-oidc

Last activity
29 minutes ago

Maximum session duration
1 hour

Also, create an identity mapping with this IAM role's ARN to allow access to the cluster.

```
dahi@DESKTOP-ATVVKNF:~$ eksctl create iamidentitymapping \
    --cluster cluster \
    --region us-east-1 \
    --arn arn:aws:iam::265681628035:role/github-to-aws-oidc \
    --group system:masters \
    --no-duplicate-arns \
    --username dahi
2024-04-27 01:27:45 [i]  checking arn arn:aws:iam::265681628035:role/github-to-aws-oidc
against entries in the auth ConfigMap
2024-04-27 01:27:45 [i]  adding identity "arn:aws:iam::265681628035:role/github-to-aws-
oidc" to auth ConfigMap
```

Create AWS infrastructure with Terraform

We will create a Kubernetes environment to deploy the applications to and verify the deployment step. Run the Terraform and type **yes** after reviewing the expected changes.

```
workspace root$ cd setup/terraform/
terraform root$ export AWS_ACCESS_KEY_ID='*****'
export AWS_SECRET_ACCESS_KEY='*****'
terraform root$ terraform init
Initializing the backend...
.
.
Terraform has been successfully initialized!
.
.
terraform root$ terraform apply
.
.
Apply complete! Resources: 24 added, 0 changed, 0 destroyed.

Outputs:
```

```
backend_ecr = "265681628035.dkr.ecr.us-east-1.amazonaws.com/backend"
cluster_name = "cluster"
cluster_version = "1.25"
frontend_ecr = "265681628035.dkr.ecr.us-east-1.amazonaws.com/frontend"
github_action_user_arn = "arn:aws:iam::265681628035:user/github-action-user"
```

Continuous Deployment workflow for Frontend

Create the file **.github/workflows/frontend-cd.yml** file that:

- Is triggered on push events against the main branch,
    - Only when code in the frontend application changes
- Can be run on-demand (i.e. manually without needing to push code)
- Accomplishes the following tasks:
    - Test
    - Build
        - Run after tests succeed
        - Tag the built docker image with the git sha (use GitHub Context)
    - If deploying to a Kubernetes cluster:
        - Push the image to ECR
        - Apply the Kubernetes manfiests using the image tag from build

```yaml
name: Frontend Continuous Deployment

on:
  workflow_dispatch:
  push:
    branches:
      - main

defaults:
  run:
    working-directory: frontend

jobs:
  Test:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout
        uses: actions/checkout@v4
      - name: Setup Node.js environment
        uses: actions/setup-node@v4
        with:
          node-version: latest
          cache: "npm"
          cache-dependency-path: frontend/package-lock.json
      - name: Install dependencies
        run: npm ci
      - name: Run the tests
        run: CI=true npm test

  Build:
    needs: Test
    runs-on: ubuntu-latest
    environment: deployment
    permissions:
      id-token: write
    outputs:
      image-uri: ${{ steps.build.outputs.image-uri }}
    steps:
      - name: Checkout
        uses: actions/checkout@v4
      - name: Configure AWS Credentials
        uses: aws-actions/configure-aws-credentials@v4
        with:
          aws-region: ${{ vars.AWS_REGION }}
          # Using GitHub's OIDC provider to retrieve credentials.
          # This method uses OIDC to get short-lived credentials
          # which is the secure and recommended method by GitHub officially.
          role-to-assume: arn:aws:iam::265681628035:role/github-to-aws-oidc
      - name: Login to Amazon ECR
        id: login-ecr
        uses: aws-actions/amazon-ecr-login@v2
      - name: Build, tag, and push docker image to Amazon ECR
        id: build
        env:
          # Set the variable (REGISTRY) with the output of the previous step (login-ecr).
          REGISTRY: ${{ steps.login-ecr.outputs.registry }}
          REPOSITORY: ${{ vars.AWS_ECR_REPO_FRONTEND }}
          # Set the image tag environment variable to the current GitHub commit SHA
          # to ensure a unique identifier for the Docker image.
          IMAGE_TAG: ${{ github.sha }}
          REACT_APP_MOVIE_API_URL: ${{ vars.REACT_APP_MOVIE_API_URL }}
        # Build the Docker image and push to the ECR registry.
        # Also appemd the image URI to be refered and used in the next dependent job.
        run: |
          docker build --build-arg=REACT_APP_MOVIE_API_URL=$REACT_APP_MOVIE_API_URL -t
$REGISTRY/$REPOSITORY:$IMAGE_TAG .
          docker push $REGISTRY/$REPOSITORY:$IMAGE_TAG
          echo "image-uri=$REGISTRY/$REPOSITORY:$IMAGE_TAG" >> "$GITHUB_OUTPUT"

  Deploy:
    needs: Build
    runs-on: ubuntu-latest
    environment: deployment
    permissions:
      id-token: write
    steps:
      - name: Checkout
        uses: actions/checkout@v4
      - name: Configure AWS Credentials
        uses: aws-actions/configure-aws-credentials@v4
        with:
          aws-region: ${{ vars.AWS_REGION }}
          role-to-assume: arn:aws:iam::265681628035:role/github-to-aws-oidc
      - name: Login to Amazon ECR
        id: login-ecr
        uses: aws-actions/amazon-ecr-login@v2
      - name: Deploy Kubernetes manifests
        env:
          IMAGE_URI: ${{ needs.Build.outputs.image-uri }}
          CLUSTER: ${{ vars.AWS_CLUSTER }}
          REGION: ${{ vars.AWS_REGION }}
        # Update the kubeconfig file to authenticate with the EKS cluster.
        # Set the Docker image URI in Kubernetes manifests
        # and builds the Kubernetes manifests using Kustomize
        # and apply them to the cluster.
        run: |
          cd k8s
          aws eks update-kubeconfig --name $CLUSTER --region $REGION
          kustomize edit set image frontend=$IMAGE_URI
          kustomize build | kubectl apply -f -
```

Continuous Deployment workflow for Backend

Create the file **.github/workflows/backend-cd.yml** file that:

- Runs on **push** against the **main** branch, only when code in the frontend application changes.
- Is able to be run on-demand (i.e. manually without needing to push code)
- Runs the same lint/test jobs as the Continuous Integration workflow.
- Runs a build job only when the lint and test jobs pass.
    - The built docker image should be tagged with the git sha.
- Runs a deploy job that applies the Kubernetes manifests to the provided cluster.
    - The manifest should deploy the newly created tagged image.

```
name: Backend Continuous Deployment

on:
  workflow_dispatch:
```

```yaml
  push:
    branches:
      - main

defaults:
  run:
    working-directory: backend

jobs:
  Test:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout
        uses: actions/checkout@v4
      - name: Setup Python
        uses: actions/setup-python@v5
        with:
          python-version: "3.10"
      - name: Install dependencies
        run: |
          python -m pip install --upgrade pip
          pip install pipenv
          pipenv install
      - name: Run the tests
        run: pipenv run test

  Build:
    needs: Test
    runs-on: ubuntu-latest
    environment: deployment
    permissions:
      id-token: write
    outputs:
      image-uri: ${{ steps.build.outputs.image-uri }}
    steps:
      - name: Checkout
        uses: actions/checkout@v4

      - name: Setup Python
        uses: actions/setup-python@v5
        with:
          python-version: "3.10"

      - name: Install dependencies
        run: |
          python -m pip install --upgrade pip
          pip install pipenv
          pipenv install

      - name: Configure AWS Credentials
        uses: aws-actions/configure-aws-credentials@v4
        with:
          aws-region: ${{ vars.AWS_REGION }}
          role-to-assume: arn:aws:iam::265681628035:role/github-to-aws-oidc

      - name: Login to Amazon ECR
        id: login-ecr
        uses: aws-actions/amazon-ecr-login@v2

      - name: Build, tag, and push docker image to Amazon ECR
        id: build
        env:
          REGISTRY: ${{ steps.login-ecr.outputs.registry }}
          REPOSITORY: ${{ vars.AWS_ECR_REPO_BACKEND }}
          IMAGE_TAG: ${{ github.sha }}
        run: |
          docker build -t $REGISTRY/$REPOSITORY:$IMAGE_TAG .
          docker push $REGISTRY/$REPOSITORY:$IMAGE_TAG
          echo "image-uri=$REGISTRY/$REPOSITORY:$IMAGE_TAG" >> "$GITHUB_OUTPUT"

  Deploy:
    needs: Build
    runs-on: ubuntu-latest
    environment: deployment
    permissions:
      id-token: write
    steps:
      - name: Checkout
        uses: actions/checkout@v4
      - name: Configure AWS Credentials
        uses: aws-actions/configure-aws-credentials@v4
        with:
          aws-region: ${{ vars.AWS_REGION }}
          role-to-assume: arn:aws:iam::265681628035:role/github-to-aws-oidc
      - name: Login to Amazon ECR
        id: login-ecr
        uses: aws-actions/amazon-ecr-login@v2
      - name: Deploy Kubernetes manifests
        env:
          IMAGE_URI: ${{ needs.Build.outputs.image-uri }}
          CLUSTER: ${{ vars.AWS_CLUSTER }}
          REGION: ${{ vars.AWS_REGION }}
        run: |
          cd k8s
          aws eks update-kubeconfig --name $CLUSTER --region $REGION
          kustomize edit set image backend=$IMAGE_URI
          kustomize build | kubectl apply -f -
```