Project: Movie Picture Pipeline

learn.udacity.com/nanodegrees/nd9991/parts/cd12354/lessons/616ed4d7-a5ad-43cc-a147-80ee24bad75f/concepts/616ed4d7-a5ad-43cc-a147-80ee24bad75f-project-rubric

Build CI Pipeline for Frontend

Success Criteria

Build a Continuous Integration pipeline for the frontend end application using Github Actions. The pipeline should be configured to meet the team's needs, fulfilling the requirements of linting, testing, and building of the application on every pull request against the `main` branch.

Ensure the workflow is named Frontend Continuous Integration and the file should be called frontend-ci.yaml.

Specifications

There should be a file called .github/workflows/frontend-ci.yaml in the root of the project.

The following jobs should be present

LINT JOB: There should be a job in the workflow that runs linting. The job should have these steps:

- Checkout code
- Setup NodeJS
- Perform a cache action to restore cache before dependency install
- Install dependencies
- Run the `npm run lint` command

TEST JOB: There should be a job in the workflow that runs the tests The job should have these steps:

- Checkout code
- Setup NodeJS
- Perform a cache action to restore cache before dependency install
- Install dependencies
- Run the `npm run test` command

The two jobs above should run in parallel

BUILD JOB: This job should only run after the first 2 succeed (student has to use the "needs" syntax) There should be a step that builds the application using docker. The job should have these steps:

- Checkout code
- Setup NodeJS

Success Criteria

Specifications

- Perform a cache action to restore cache before dependency install
- Install dependencies
- Run the `npm run test` command

The pipeline should be executed automatically on pull_request The pipeline should be able to be run manually The pipeline should be running without errors with all tests passing and no output failures from any of the steps

Build CI Pipeline for Backend

Success Criteria

Build a Continuous Integration pipeline for the backend application using Github Actions. The pipeline should be configured to meet the team's needs, fulfilling the requirements of linting, testing, and building of the application on every pull request against the `main` branch.

Ensure the workflow is named "Backend Continuous Integration" and the file should be called "backend-ci.yaml"

Specifications

There should be a file called .github/workflows/backend-ci.yaml in the root of the project There should be a job in the workflow that runs linting. There should be a job in the workflow that runs the tests Linting and testing should be done in parallel.

The job and lint should complete before proceeding to the build step There should be a job that builds the application using docker.

The pipeline should be executed automatically on pull_request The pipeline should also be able to be run manually The pipeline should be running without errors with all tests passing and no output failures from any of the steps

Build CD Pipeline for Frontend

Success Criteria

Specifications

Success Criteria	Specifications
Build a Continuous Deployment pipeline for the frontend application using Github Actions. The pipeline should be configured to meet the team's needs, fulfilling the requirements of linting, testing, building and deploying of the application on every merge to the main branch. Ensure the workflow is named "Frontend Continuous Deployment" and the file should be called "frontend-cd.yaml"	There should be a file called .github/workflows/frontend-cd.yaml in the root of the project There should be a step in the workflow that runs linting and passes There should be a step in the workflow that runs the tests and passes There should be a step that builds the application using docker only after linting and testing complete (use the needs directive) This step should also utilize build-args to ensure the application is built with an environment variable REACT_APP_MOVIE_API_URL There should be a step that utilizes aws-actions/amazon-ecr-login action for logging into ECR. (using 3rd party actions) The ECR login step should also be accessing Github Secrets for credentials. (secure approach) There should be a step that pushes the docker image to ECR in the AWS account. There should be a step that deploys the application using kubectl to the eks cluster The pipeline should be executed automatically on merges to the `main` branch The pipeline should be able to be run manually for verification purposes The pipeline should be running without errors with all tests passing and no output failures from any of the steps If there are AWS credentials anywhere in any of the pipelines = FAIL If any of the pipelines are failing to run or have failed steps = FAIL If any of the pipelines pass when there's a test failure = FAIL (will provide steps on how to simulate test failure) If the docker image doesn't get uploaded to ECR = FAIL If the application isn't successfully running on the cluster = FAIL (the frontend should be able to pull the list of movies and verifies the environment variable was passed correctly)

Build CD Pipeline for Backend

Success Criteria	Specifications

Success Criteria	Specifications
Build a Continuous Deployment pipeline for the backend application using Github Actions. The pipeline should be configured to meet the team's needs, fulfilling the requirements of linting, testing, building and deploying of the application on every merge to the main branch. Ensure the workflow is named "Backend Continuous Deployment" and the file should be called "backend-cd.yaml"	There should be a file called .github/workflows/frontend-cd.yaml in the root of the project There should be a step in the workflow that runs linting. There should be a step in the workflow that runs the tests There should be a step that builds the application using docker. There should be a step that utilizes aws-actions/amazon-ecr-login action for logging into ECR. (using 3rd party actions) The ECR login step should also be accessing Github Secrets for credentials. (secure approach) There should be a step that pushes the docker image to ECR in the AWS account. There should be a step that deploys the application using kubectl to the kubernetes cluster The pipeline should be executed automatically on merges to the `main` branch The pipeline should be able to be run manually for verification purposes The pipeline should be running without errors with all tests passing and no output failures from any of the steps If there are AWS credentials anywhere in any of the pipelines = FAIL If any of the pipelines are failing to run or have failed steps = FAIL If any of the pipelines pass when there's a test failure = FAIL (will provide steps on how to simulate test failure) If the docker image doesn't get uploaded to ECR = FAIL If the application isn't successfully running on the cluster = FAIL

Suggestions to Make Your Project Stand Out

Implementing a custom, reusable action to reduce repeated steps would be a huge improvement to the project.

Reducing build times via caching dependencies or building docker images from cache

Any post-workflow action such as writing a comment to the PR would be an excellent way to showcase further experience with Github Actions

---