Continuous Integration and Continuous Deployment of a dockerized application using AWS CodeBuild, CodePipeline and GitHub

1. CONTINUOUS INTEGRATION

All the source code is saved in a GitHub repository. We utilized AWS CodeBuild for the purpose of continuous integration, so whenever a change is committed in the GitHub repository, a build automatically triggers in AWS CodeBuild, which builds the docker image and pushes the image to AWS ECR. The build steps are mentioned in buildspec.yml file provided below.

Store the source files in GitHub repository.

Example files:

app.js

```
console.log("Hello, World");
```

package.json

```
{
  "name": "my-node-app",
  "version": "1.0.0",
  "description": "A simple Node.js 'Hello, World!' application",
  "main": "app.js",
  "scripts": {
    "start": "node app.js"
  },
  "author": "Your Name",
  "license": "MIT",
  "dependencies": {
    "express": "^4.17.1"
  }
}
```

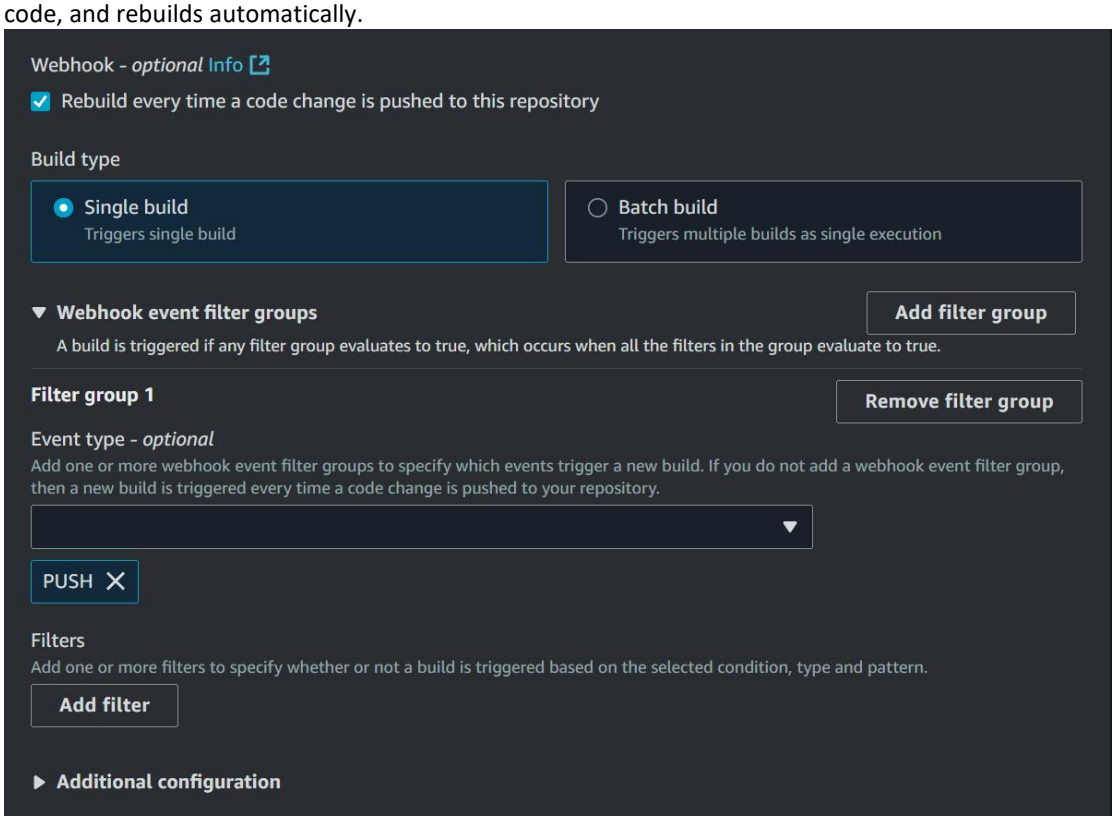
Dockerfile

```
# Use an official Node.js runtime as a parent image
FROM node:latest

# Set the working directory inside the container
WORKDIR /usr/src/app

# Copy package.json and package-lock.json to the container
COPY package*.json ./

# Install app dependencies
RUN npm install

# Copy the rest of your application source code to the container
COPY . .

# Expose a port for your Node.js application
EXPOSE 3000

# Define the command to run your Node.js application
CMD ["node", "app.js" ]
```

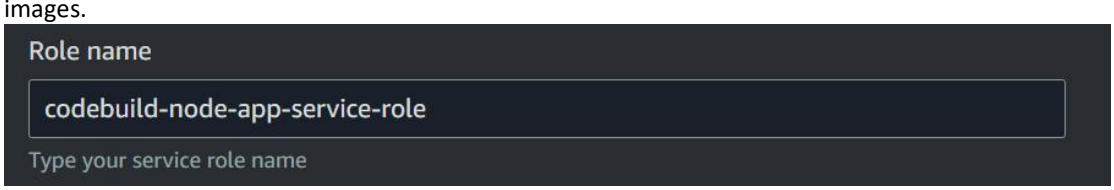
buildspec.yml
Be sure to enter the ECR login command and repository's URI, so the image could be pushed to the intended repository

```yaml
version: 0.2

phases:
  pre_build:
    commands:
      - echo Logging in to Amazon ECR...
      - aws --version
      - aws ecr <login-command>
      - REPOSITORY_URI=
      - COMMIT_HASH=$(echo $CODEBUILD_RESOLVED_SOURCE_VERSION | cut -c 1-7)
      - IMAGE_TAG=build-$(echo $CODEBUILD_BUILD_ID | awk -F":" '{print $2}')

  build:
    commands:
      - echo Build started on `date`
      - echo Building the Docker image...
      - docker build -t $REPOSITORY_URI:latest .
      - docker tag $REPOSITORY_URI:latest $REPOSITORY_URI:$IMAGE_TAG

  post_build:
    commands:
      - echo Build completed on `date`
      - echo Pushing the Docker images...
      - docker push $REPOSITORY_URI:latest
      - docker push $REPOSITORY_URI:$IMAGE_TAG
      - echo Writing image definitions file...
      - printf '[{"name":"nodeapp","imageUri":"%s"}]' $REPOSITORY_URI:$IMAGE_TAG > imagedefinitions.json
      - cat imagedefinitions.json

artifacts:
    files: imagedefinitions.json
```

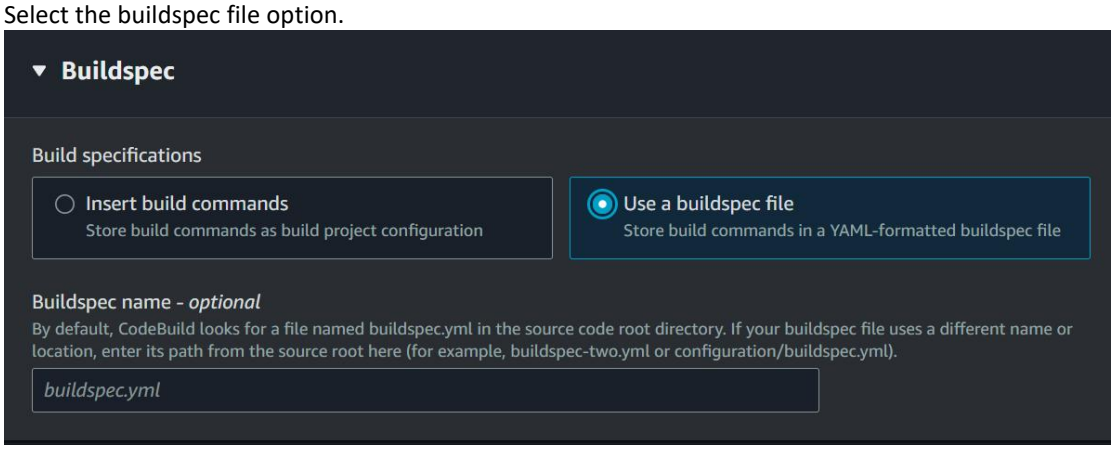
Now create a project on AWS CodeBuild
Set your GitHub repository as your primary source
And then select the webhook so the project instantly gets access to any change committed in GitHub code, and rebuilds automatically.



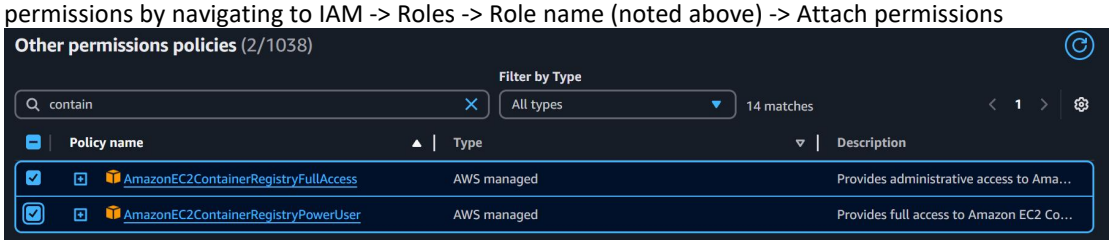
note the service role name, as you'll have to give this role access to ECR to store the built docker images.



Select the buildspec file option.



After creating the project, give the service role the following permissions (you can give these permissions by navigating to IAM -> Roles -> Role name (noted above) -> Attach permissions



These permissions allow the built docker images to be automatically pushed to ECR during continuous integration.

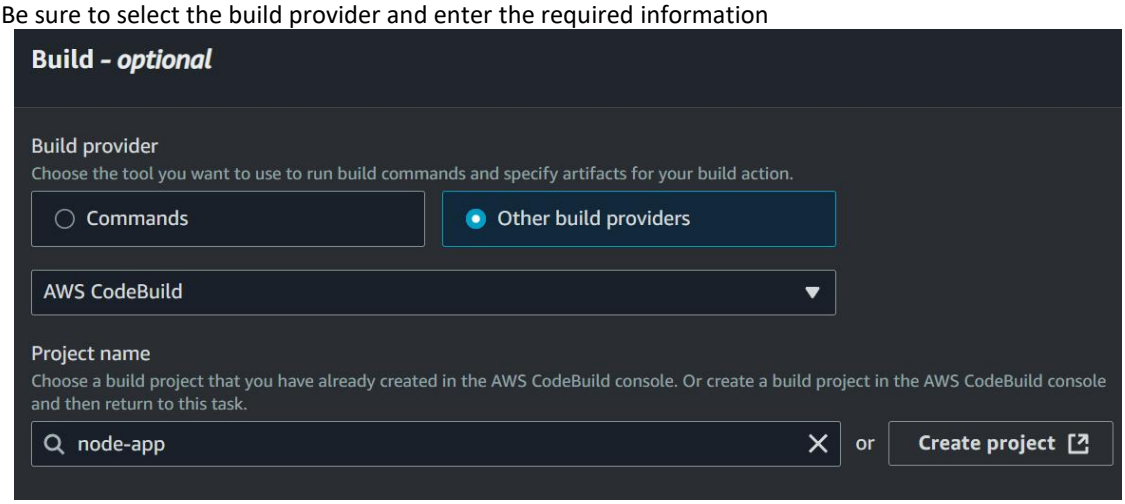
Now whenever you commit a change in GitHub repository, a build will automatically be triggered and a new image will be pushed to ECR.

2. CONTINUOUS DEPLOYMENT

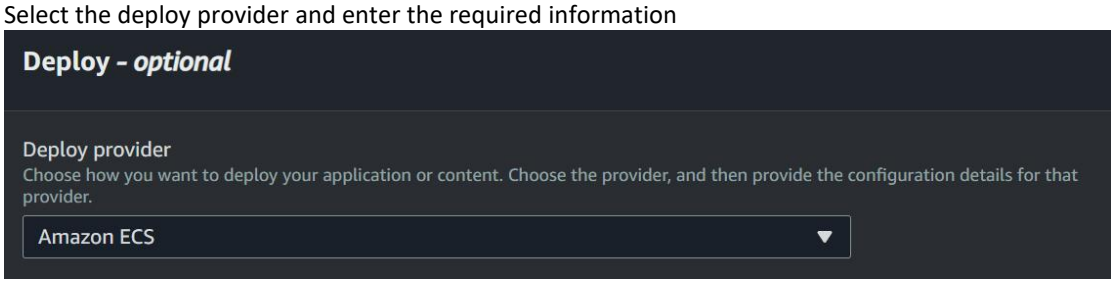
Create an ECS cluster, create a task definition using the image pushed by CodeBuild in the previous step, and then create a service within the cluster using the task definition.

Create a CodePipeline with custom configurations.

Be sure to select the build provider and enter the required information



Select the deploy provider and enter the required information



Once the pipeline is created, the app will automatically be deployed after any change commits in GitHub.