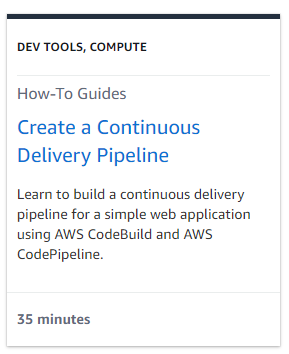
Loc Nguyen

CS.4650.02

Professor Johanssen

Tutorial 1 – Setting up Continuous Delivery Pipeline



Background – Currently, I’m working on a web application that deploys continuously from GitHub. However, that deployment is with GitHub workflows and is only 1-way, from GitHub repo to AWS EC2 instance. Users data will be written onto the live server database but that information doesn’t get written back to the GitHub database repo file. Hence, every time new patches are pushed, the server database which contains data will get overwritten by the GitHub empty database file. This is a good opportunity for me to learn proper CI/CD pipeline.

1. Launching a web-app with NodeJS using AWS Elastic Beanstalk

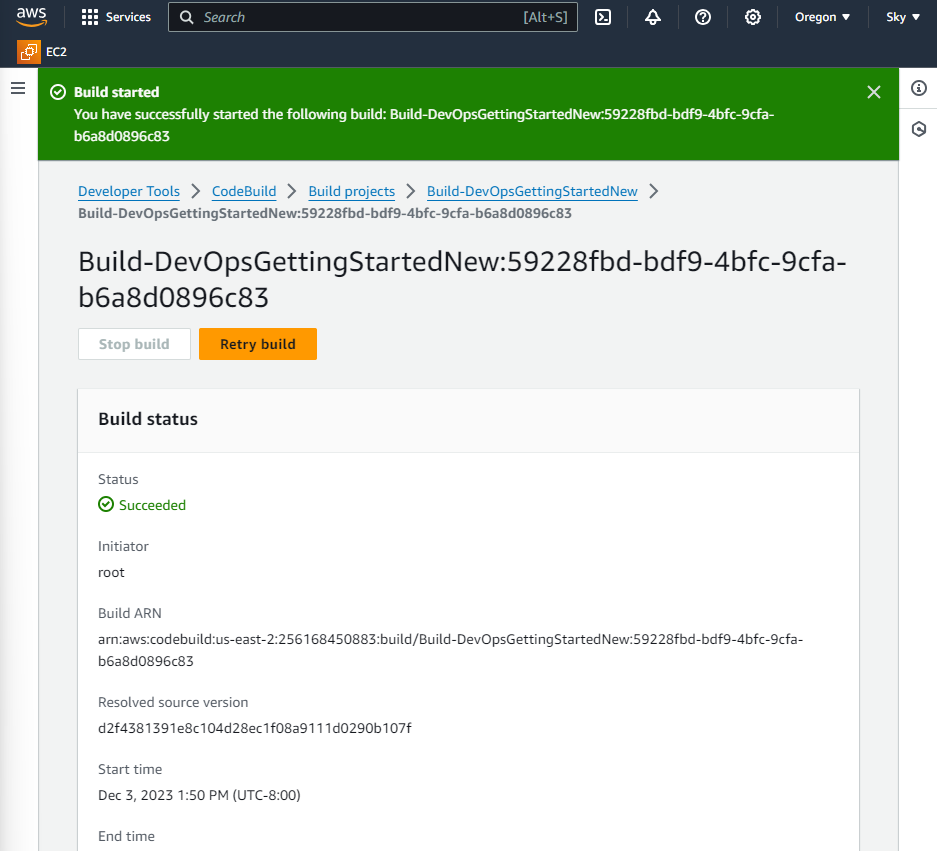
A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated

1. Using CodeBuild for continuous integration using GitHub as source repo



1. Use CodePipeline for continuous deployment with GitHub as source repo and CodeBuild for auto build and testing

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Adding manual review as a step to code deployment pipeline. Print out a line “I love blue” to test the pipeline

A screenshot of a program

Description automatically generated

A white background with black dots

Description automatically generated

1. Winding down and deleting resources
2. Final architecture

A diagram of a construction site

Description automatically generated

I’m super proud that I know a little bit more about how to deploy code secured and fast with AWS microservices. I do want to learn more about domain registration and firewall policies when deploying a production websites versus a development website.

Tutorial 2 – Build a Basic Web Application Static and serverless

A screenshot of a web application

Description automatically generated

Background – Web application are great for new businesses. I’m excited to learn more about different ways to create web app

1. Overall architecture

A diagram of a cloud

Description automatically generated

1. Deploy the web app with zip file index.html with AWS Amplify

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

1. Create serverless functions with AWS Lambda and test it

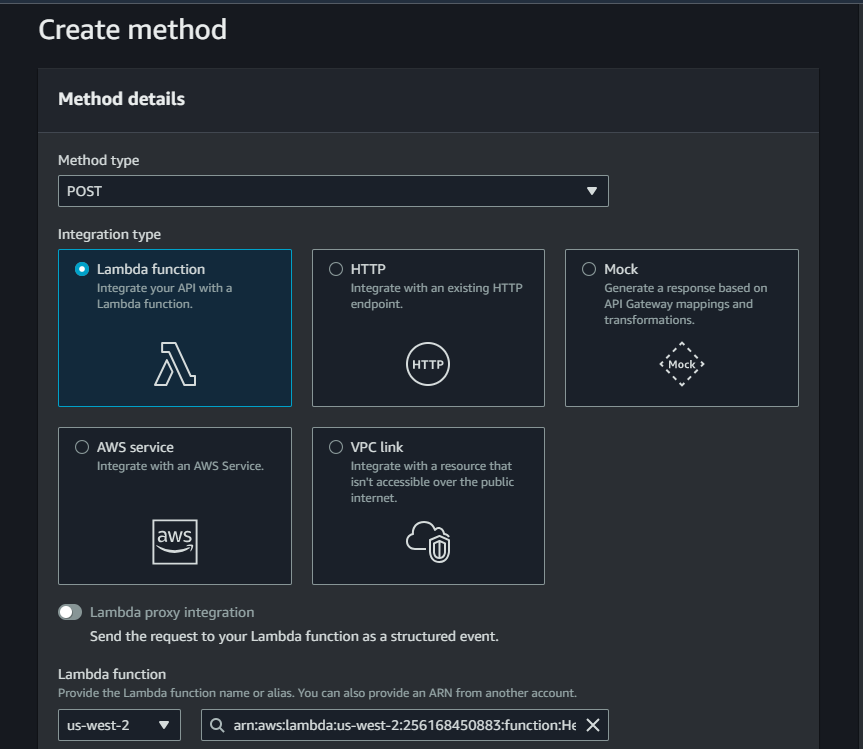
A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

1. Use Amazon API Gateway to create RESTful API to make calls to Lambda function from web client. Successfully deployed an API and tested it. Response with code 200 OK.

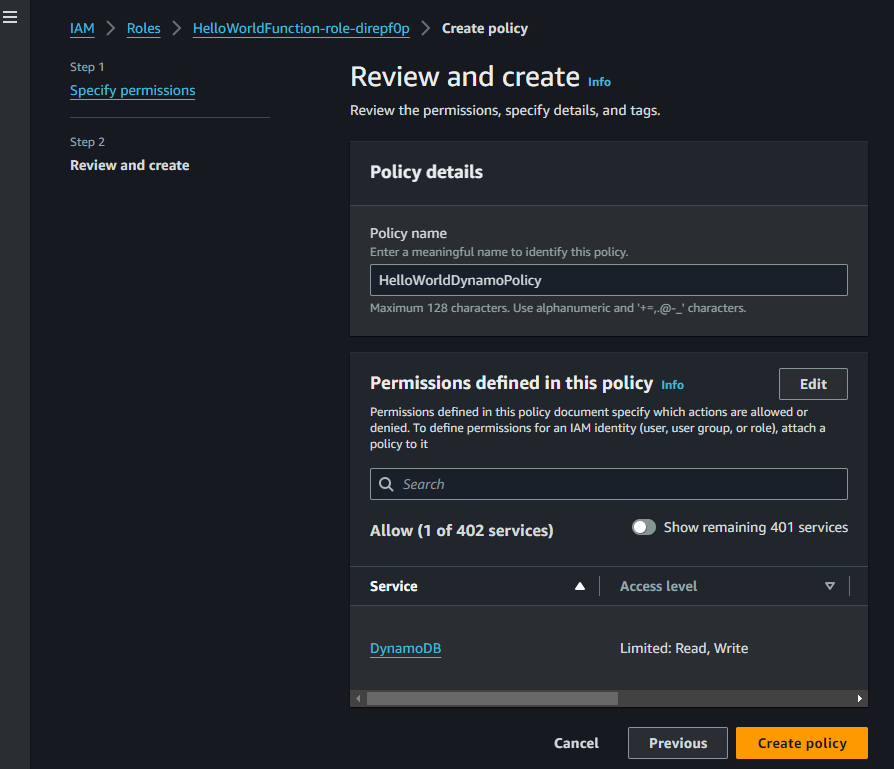


A screenshot of a computer

Description automatically generated

1. Use DynamoDB to store data. Use AWS IAM to give services permissions to interact with each other.

Created a DynomoDB table and IAM policy



Tested and made sure data is written into the DB

A screenshot of a computer

Description automatically generated

Current Architecture

A diagram of a cloud computing system

Description automatically generated

1. Call an API Gateway from an HTML image

Modify the original HTML to call the API Gateway using the invoke URL - <https://2jl7k63sh2.execute-api.us-east-2.amazonaws.com/dev> and deploy it on AWS Amplify

A screen shot of a computer program

Description automatically generated

Tested the interactive website with an API call

A screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Final application architecture

A diagram of a cloud

Description automatically generated

End of lesson takeaway – super stoked to learn how web applications deployment work and will incorporate my learning into future projects. I’m curious about limitations behind this process and thinking about how I can utilize this to make a weather app calling the OpenWeather API that’s has 1000 free API call everyday.