CS-470: Full Stack Development II

Professor Watson

Trice Bonomo

February 22, 2023

Final Reflection

Experiences and Strengths

During this course, I learned how to migrate an on-premises application to the AWS cloud platform. In addition to familiarizing myself with AWS services like S3, Lambda, Gateway, Dynamo, and CloudTrail, I learned how to use Docker, and Docker Compose, and revisited my PowerShell and Visual Studio Code skills.

My greatest strength as a software developer is my ability to parse through new information to obtain new skills and knowledge. I have a keen eye for detail, a passion for problem-solving, and exceptional written and verbal communication skills. I take pride in my work and strive for continuous self-improvement.

Without real-world experience, I hesitate to declare myself prepared for any software development role. Though I have done well in college, the fact that my college skills have little, if any, real-life value has become glaringly obvious over time. My degree showcases my ability to learn and persevere through adversity- but whether I can program in real-time, with other developers, on real-world systems is still up for debate. I'm stepping into the job market fully aware of my inadequacy. Presently, I am looking for roles that align with my interests and hoping someone will be willing to take a chance on me.

Planning for Growth

To handle errors in a serverless framework, a step function state machine can be created. Once a Lambda function and IAM roles have been assigned in AWS, the state machine will call the API and handle exceptions. The AWS Step Functions console is used to create the state machine, view execution details, and review the workflow.

In a serverless framework, scaling is automatically handled by the framework. AWS offers pricing information that can be used to approximate costs, but outside tools, like serverlesscal.com, can also be used to predict overall costs. Between containers and serverless, containers are going to provide the most predictable costs because calculations are handled in-house. Despite serverless benefits, auto-pricing means trusting AWS to correctly calculate usage, and having little recourse for contesting the bill.

When expanding, the questions shouldn't be which option is cheaper, but what feels reasonable. If handling scaling, load balancing, and other administrative tasks eats into development hours, choosing serverless might be the better option. Developers can focus on building out the application, and adding features, and so the company thrives despite the added cost of managed services. If customization, control, and latency are important aspects of a business, containerizing an application might be the best option. Developers might be spread a little thin having to manage both the application and servers, but the company will have an easier time predicting their expenses and tracking their resources. Each situation has its pros and cons, but neither option is inherently better or worse.