David Smith

February 27, 2022

CS 470 Final Reflection

https://youtu.be/3cZxr5u_m3c

This course has given me a chance to explore containerization, orchestration, and moving a full stack web application to AWS services. These services include S3, DynamoDB, Lambda, and API Gateway.

As a software developer, I am able to analyze situations and determine appropriate solutions to customer problems. I have experienced Agile and Waterfall methods of development. I have also helped design systems for military and commercial airline applications.

As someone who has already worked in multiple fields including software development, I have experience finding solutions to problems that are both technical and nontechnical. I believe I am ready for front-end, back-end, systems analyst, software engineer, and other roles.

The application that I worked on in this course can further be increased in efficiency by ensuring that Lambda functions have a reasonable timeout based on their complexity and defining points at which the application should scale out. Predicting the cost could be achieved using the analytics that AWS provides to examine the amount of usage for each part of the system. It may become necessary re-architect or reprovision some parts of the application if they turn out to require more resources than expected. Overall, I think serverless is more cost predictable, though that cost can be more than using containers. Whichever method is used to host the application, it is important to ensure that distributed denial of service attacks, and other attacks that cause high traffic, are stopped as quickly as possible, as they can create latency or unavailability for customers and can exponentially increase costs.

When thinking about expanding the serverless application migrated to AWS in this course, it is important to consider how much more S3 storage would be needed to store the front-end website, how much more compute power would be needed in Lambda functions, and how much more DynamoDB space and interactions would be needed. All these factors will increase the cost of running the application, and those costs would increase proportionally to both the extra effort needed for the expansion and the number of users using them. However, each of these expanded areas would also be able to scale out and in to accommodate user numbers.

When considering future growth, elasticity can help with the number of clients that can connect without permanently expanding the number of services being purchased. Pay-for-service models could become prohibitively expensive at some future point, which could cause extensive reworking and migration to another platform if AWS becomes too costly. This is because AWS services are all proprietary and are not available on other serverless platforms. This is a case where containers would be more appropriate, especially Docker, as they are available across almost all cloud platforms. This is something that should be considered early in the application lifecycle.