Amber Caldwell
02/27/2022
CS 470 Final Reflection
https://youtu.be/dnVHvFkMi8w

- **Experiences and Strengths:** Explain how this course will help you in reaching your professional goals.

  Before this course I had very little experience with Amazon Web Services. The most I had done was manage users in Amazon Connect. I had never used a container tool, and I only had a vague idea of what a container was or could be used for. I have had the privilege of getting to know more about Docker and the benefits of containers. I know what container orchestration is, and how useful Docker compose is. I also have the skills to migrate a full-stack application to AWS in the cloud. I have had experience with a variety of AWS services including IAM, S3, Lambda, API Gateway, and DynamoDB.

  These new skills will be essential, because so many companies are looking to move their resources to the cloud and AWS offers a lot of services. I think one of my biggest strengths as a software developer is the ability to learn quickly. I may not know something at the start of a project, but I can learn fast. Additionally, I can research issues well when I get stuck. Also, I feel that I can convey technical issues well to a non-technical audience. Considering these strengths, I think I would be prepared to take on a role that recommends cloud solutions or a developer role that allows for growth, such as entry level or junior.

- **Planning for Growth:** Synthesize the knowledge you have gathered about cloud services.

  One way that error handling can be implemented is using API Gateway and Lambda. Currently, we only have the status code of 200 implemented. However, there are so many other status codes to account for. Liston recommends a chart to follow, which we can implement using our previous methods for status code 200. This will involve some tweaking for each status code but should be easily implemented. In terms of scaling, the S3 bucket will automatically scale to the performance. In terms of cost prediction, if we take our containers that we made on Docker for instance and compare them to that of the serverless models on AWS, the cost prediction of Docker will always be more predictable. We know that using the free version of Docker, we're not paying anything, and with AWS depending on when our application code is being used (after our free tier expires) we can be charged. Let's take this a step farther. Let's assume we're an actual business and we're using an Azure Container Instance, there is a possibility that this could be more expensive than serverless because we're being charged for storage 24/7. With AWS we are on a pay-for-use model. At this point, it would really take a lot of scrutinizing as to what is needed and how often it's being used as which option is most beneficial. However, to summarize, in terms of predictability, containers are more predictive in terms of

cost. It's hard to predict when your application will be used in a serverless environment, which makes cost unpredictable.

Growth is inevitable. Growth doesn't have to mean scaling in size. It can simply mean that hardware is being upgraded or security is being upgraded. Therefore, there is growth. Pros for growth should be used storage space outweighs available storage space, age of hardware requires an upgrade of hardware, company requires a new solution which is incompatible with current solution, scaling is required, a more secure solution is required, old hardware broke / died / is about to go, or this is a more cost-effective decision. Cons for growth could include recently scaled, plenty of storage currently, upcoming change is compatible with current systems, old hardware meets security standards, room to scale in the future, or not a cost-effective decision. Considering that hardware solutions can be quite inelastic, elasticity should be kept in mind during this solution. This means that storage can be allocated and de-allocated as needed without an issue. With hardware this is not the case. Furthermore, in a pay-for-use or pay-for-service model, you'll be charged as the service is used. In acquiring hardware, you must plan for the storage, and whether the storage or resources are used, you're still charged for what you paid upfront. This can be a giant waste of money.

References

Liston, B. (2016, June 13). *Error Handling Patterns in Amazon API Gateway and AWS Lambda*. AWS. Retrieved February 27, 2022, from https://aws.amazon.com/blogs/compute/error-handling-patterns-in-amazon-api-gateway-and-aws-lambda/