Guilherme Pereira

April 23rd, 2022

CS – 470

Final Reflection

In this course, we took our MEAN stack that we developed in CS-465 and moved it onto the cloud by means of containerization and Amazon Web Services. We learned how to utilize Docker Compose to create separate containers and how to prepare an application for deployment. We also learned how to manage AWS' different services and how they work together to create a cloud-based web application. We learned how to create an S3 bucket which stores our files and data online. Lambda was used to create functions that would communicate between each service. API Gateway was a means for the client to connect to our web application and from there onto AWS. Finally, we used DynamoDB as our database to hold the questions and answers using a single-table model.

Experience and Strengths

This course has taught me more than just how to move a web application onto the cloud. For the earlier parts, it taught me how to read from documentation and to always double-check sources from the most recent versions of a program. I have also learned how to navigate a few documentation pages and received the experience of what the layout looks for each company handled (Docker Inc. and Amazon). While I still do not understand completely the technicalities behind how the programs interact with one another (source-code of Lambda), I have learned to utilize the service itself for its intended uses. I've developed the skills to create and manage a cloud project on AWS, and had my first experience into setting policies and roles for those who are developing with the project. I have also learned what makes DynamoDB more unique from its other competitors and how API Gateway works. I have also learned some HTTP protocol tricks I picked up earlier when testing my application through API Gateway as well.

My strengths as a software developer comes from debugging and re-tracing my steps. I have learned how to explain services well and how they work from a high-level perspective, but I would also like to learn more about the technicalities of the low-level side. Personally, I have always seen myself handling data as my greatest strength and representing the data on a chart or graph. However, I see myself looking more to the back-end sort of things for making an application. While I am an artist and do enjoy making the layout for a web-page, I would also like to learn how the back-end works as gaining an understanding of this area in the past two courses have given me an interest. I am always following best practices and learning to make my code as efficient as possible. I believe my greatest strength as a programmer would have to be my organization in code, but it was not something I had too much experience working on aside from creating the yaml files for Docker Compose.

My creative side would do well to learn CSS and HTML to make a captivating web page for the front-end. My curiosity for knowledge would definitely do well to pursue more about how the back-end works and understanding how the application comes together as a whole. Overall, the course has given me a new experience to pursue in the field of a web or full stack developer.

Planning Growth

One of the most important aspects I have learned about serverless applications is that it requires less attention on the server side and more attention to the application itself. I believe this is one of the strongest driving points to go serverless since developers and businesses do not need to manage different things related to the server such as outages, the load balancer, traffic, and API communication. Instead, different companies provide different services to make a program go serverless, whether it is Amazon’s AWS or Microsoft’s Azure. Additionally, serverless may already come with a lot of useful tools such as graphs to check on traffic and the number of requests being made. I believe going serverless would be beneficial for a company that requires more attention to the customer-side of things instead of the back-end.

The other benefit of serverless is how it scales. No matter how much traffic goes in or out, AWS can scale the incoming requests well enough to not allow the servers to go down and crash. Unfortunately, this is also where some of the drawbacks come in. To find the exact error on the server-side is an issue that AWS or other service providers would need to handle. In this course, I only managed to debug my application with FireFox’s Developer Tools to see why something wouldn’t load properly. To investigate the minutia of things would require me to access the server’s code, but I did not run into those issues for this project. Costs are based on a fixed rate AWS provides. It’s a very small amount but scales depending on the performance of each service, such as calls from API Gateway or executions by Lambda. In determining the cost of containers or serverless applications, I believe containers are more predictable, as it’s a fixed rate whenever it is running. However, serverless is cheaper and likely better to manage since you only pay for what you use. If a containerized or serverless application receive no traffic but are both “running”, the container will need to pay for the server space it has taken up while serverless will not pay anything since nothing was requested.

When deciding to expand space for a server, I believe serverless meets the criteria for meeting the demand of users. Expanding is great that it allows larger volumes of traffic to go through the site while not crashing it. The only issue I see with expanding a server is the added complexity it gives to those with permissions. Adding more space to new parts of the server might require developers to receive additional access, or there might be parts of the front-end or back-end that conflict with the new area in the server. I believe that the elasticity provided by serverless providers, and their pay-for-service model is a fair way of maintaining a website that fluctuates from time to time. For instance, many shops will have a higher demand from customers around the holidays or Christmas, so it would be good to use this model and not miss on potential sales due to a server crashing and preventing others from logging on.