Final Reflection

Jon M. Curtis

Southern New Hampshire University

CS-470: Full Stack Development II

Instructor John Watson

April 23, 2023

Author Note

YouTube: https://www.youtube.com/watch?v=VvGh0AcJ94k

Final Reflection

In this course we built a static website and API then containerized it. After we took it into the AWS environment and made it run there as well. Some things in this course were harder than others, but all the objectives built off each other. I would say being able to build this full-stack application from start to finish and then bringing it in the cloud was a remarkable lesson. I feel I have grown as a Developer and Programmer with this experience.

Experiences and Strengths

My professional goals are to become an independent developer and contractor. As I did research into the field, I saw that most developers end up working for multiple companies throughout their professional career. While I welcome the opportunity to work with or for companies big or small, my end goal is to eventually strike out on my own or with other like-minded developers. A lot of these smaller independent companies make game changing software that is adopted by larger corporations.

The skills I have learned will propel me into a future career in development. As someone who wants to be a "Indie Dev" full-stack is extremely important. Larger corporations will likely hire developers for just front-end and others for back-end. They may even break it down further than that, but an Indie Dev does not have that luxury of having to be able to put everything together, either solo or with a small team. Through this course I have established I can build an entire working application on my own and put it into the cloud.

I have become a "jack-of-all-trades" so to speak, but more than that I understand how it works and where to look for answer when I don't. I understand the value of "technical interviews" using "Leetcode", and how while being able to regurgitate memorized code is a starting point, the end goal is understanding. Coding without understanding is like quoting Shakespear out of context, technical right, but it doesn't work.

Planning for Growth

The use of micro-services or serverless has several advantages over traditional servers. Using serverless can improve future efficiency for customers interaction by being able to run AWS CloudFront that provides edge locations and high-speed transfers. I would also implement Elastic Load Balancing that improves the stability of high traffic applications by distributing traffic. Both of those implementations help for scaling, but for error handling I would like to implement Lambda functions and serialization of errors so they can be logged more accurately. Create a state machine with retry and catch logic. I would like to implement API throttle to protect from too many requests being made to the API.

To then predict the cost of the whole application I would build what is called a "Cost Model". I would need to gather all the services I am using no matter how miniscule and break down how they charge. Then how much data I'm storing and transferring including how long it takes to transfer. Know what regions the application is operating in, and what processes are being called. I would also do load testing to try and get some accurate numbers to feed the "Cost Model". All these steps should give a reasonably accurate predictor.

Containers are by far more predictable in cost than serverless. Containers are packed into a box and run the same every time. Serverless are hosted and how often they are used is a determining factor in cost. All these bits and pieces come into play for expansion as well. Serverless can be more cost effective if it is intermittently used but could cost more for a high demand application. Containers, while more predictable, could cost more if underutilized. So, it is important to know your market and consumer base.

If demand for an application is unknown or subject to fluctuation products like AWS give the elasticity needed to provide a stable platform for growth. It also features “pay-for-service” which can save a company money compared to traditional servers. Now if I had a known quantity application that was heavily in demand already. The elasticity that AWS provides may be overshadowed by its “pay-for-service” cost compared to traditional servers that provide better “Cost Models” and control at that point.

References

Helton, A. (2020, Mar 05). *5 Steps to Making a Predictable Cost Model for AWS Serverless Projects*. Retrieved from Ready, Set, Cloud!: https://www.readysetcloud.io/blog/allen.helton/5-steps-to-making-a-predictable-cost-model-for-aws-serverless-projects-78d78909bb82/