Andrew McPherson
10/23/2022
CS-470 Final Reflection
https://youtu.be/86u2smJwuc0

Though much of my future career field is left unknown to me, the skills I learned through this course can be applied in many potentially different fields and jobs. Not only did I learn a way to create a full stack web application and run it locally on my machine, I also learned how to work with Docker and creating containerization, and then I also learned to work with AWS a bit more which could be useful in certain job positions as AWS offers great services that may be employed by many companies. With the variety of skills learned in just this course alone, I could end up using anything that I learned in some way in the future.

I think my strengths lie in the ability to adapt to using many different software, and over time learn the more intricate details of them. As throughout much of my SNHU journey I have learned software that, at the time, was completely new to me and I had to learn on the fly how to work with it to complete my assignments. This means I am fairly flexible and that may be desirable for potential future employers that I am capable of not only using different programming languages, but also the different software that they operate with. I also think I may be decent at discerning where bugs in code are and how to resolve some of them, though I don't think I am overly confident in my skills at this time, it may be the type of role I search for when I start looking for career opportunities, perhaps as someone to manage and upkeep code and try to find problems or errors.

Due to the fact AWS offers scaling on many of its microservices, I would be able to leave much of the handling of scale up to AWS and it would save me the concern of having to either increase or decrease local physical storage or other resources, as for error handling, one way this could be resolved is through the use of AWS Lambda, as there is a way to preform tests on the Lambda functions you create to ensure they run correctly and you receive the execution result with a status code to confirm it is running correctly.

When it comes to predicting costs, assuming I am still working through AWS, they provide a service called "AWS Pricing Calculator" that I could use to get an estimate of costs for the web application I may be running. Depending on the scale of the web application and what it will be used for, using their price calculator can handle most if not all the predictions to get a reliable estimate on what I expect to be paying for the use of AWS microservices. Once again, depending on the scale planned for my web application the decision between containers or going serverless would need to be considered, if the application is to be something small and static, as it doesn't have many plans to grow in any meaningful way, it may be best to simply use containers as there is not much that would need to be dynamically changed, and because of this it may be more cost effective to simply use containers because I wouldn't need much of the services provided by AWS, thus allowing me to save more money. However, on the opposite end of this, if my planned web application is to be larger and perhaps be changing in scale as time goes (even through using AWS elasticity), it may be more cost effective to rely on AWS for such

a larger operation. While it may be true that a larger web application will ultimately cost more for the services and scale I would be using, this would potentially be better for my web application so I wouldn't have to worry about handling scale and leave that for AWS to manage.

Using AWS microservices offers a lot of pros in my eyes that would lead me to wanting to use them for any plans for expansion. The only way it would be more of a con to me is if the scale of my company or my plans for web applications reached a point where I would desire more or full control over every aspect of my business or company, otherwise I could see a benefit from many of the services provided, such as the scalability and elasticity they provide making it easier to manage my applications without having to worry if I need to allocate more resources to the project, and then risk having too much that never gets used. Also, one of the pros I see of using AWS is how many of their microservices have a way of working together and have some sort of integration with each other. Having all the various components needed to run an application all be found in the same place makes managing what little I would have to on my end that much easier. Also, since it is on the cloud it makes it easier for various people to work on the application remotely, without much need to have a local office where everyone works from.

Simply put, elasticity and pay-for-use models are quite useful for planning the growth of the whatever company I may be in. If my company or web application was growing, even exponentially, I would be more at ease knowing that elasticity from AWS would allow my application to run smoothly as more high traffic demand comes in, I would be able to rely on elasticity and not have to manage the resource allocation myself. In a way it future proofs my application because if high traffic does occur and my application grows suddenly, I would already have a plan in place for when this inevitably occurs. And if such large growth doesn't occur nothing would change for my current plans, it is almost like a fail-safe or a safety-net in the off chance a large spike in traffic occurs or that my application is steadily growing.