Luke Lauck

August 20, 2023

# CS 470 Final Reflection

The skills I learned, developed, and mastered during full stack development two will take me far. This course was about applications on the cloud. The first two weeks explained docker and containerization, helping me understand how dependencies aren't a problem. Then, we explored AWS through S3, EC2, Lambda, and DynamoDB. I may not be perfect, but this course has settled a lot of cloud information I wouldn't have learned otherwise. My strengths as a software developer come from analyzing and solving problems as they come. There have been issues along the way, but nothing I haven't been able to conquer with work and dedication. I am prepared to learn new technologies and revisit old ones as I am given different challenges down the road. Since I am new to this field, I am prepared to ask questions and finish any work given to me, no matter the challenges.

Like any business, scaling a web application comes down to cost as either the host pays more or the application becomes more efficient. The most cost-efficient methods are serverless. They are run only when called for, and much of the work is done on the user's end. Containers do have a place when it comes to constantly running servers. They are also more efficient than virtual machines, so switching over to docker beforehand is best. The predicted cost of the application can be seen through analytics but also through the predicted user base. It is important to remember that users will rake in more cash as well, so if the user payment scales with server costs, it's okay. Handling errors will come to preemptive attempts such as unit testing. During runtime, however, errors are handled as they appear. They will either be brought up as an error, or there will be a pattern of traffic as a problem can appear without it specifically being an error. However, these problems can be solved through maintenance and routine checks.

Expansion can bring in risks but also a larger user base. It doesn't have to happen all at once, especially if using Amazon Web Services. The services are charged per usage and not in installments. The web application should be remodeled to a size befitting whenever the service costs become too high or the speed too slow. But given that the web application was designed to be scalable, such as using lambdas and containerization to its full potential, there should not be many issues with expanding through numbers. I am referring to horizontal scaling in this instance, as vertical scaling will require more risks and changes in the code base.

Video about the course: https://www.youtube.com/watch?v=32Xtsc_4798