Tyler Strickland
10/26/2024
CS-470 Final Reflection
Video within repository (cant upload to youtube)

With attending this class I've learned many skills that will help me in the future pertaining to my career in computer science and full stack software development. A lot of this course has been focused around advancing my knowledge in serverless solutions and containerization particularly within AWS. I now know how to migrate a program to be serverless or just create and run a program with a containerization platform such as Docker. I believe this class has helped me improve one of my core strengths as a software developer: my flexibility to work across multiple coding languages and platforms. While I didn't write any code for this class, the learning of two new widely used platforms and the understanding of how each works (AWS and Docker) I feel greatly improves my reliability as a solid hire for any business-scale software development that may require a persistent website.

Serverless solutions like AWS help scale your project automatically as they gain more power and resources to use as they are needed. AWS has built in assistance with error handling, but alongside that depending on the scale of the problem the errors could likely be replicated and tracked down using the logs integrated in AWS. The cost for serverless would be estimated upon how many people are predicted to be visiting/using your website at any given time. For example, if you're a gift company you'd expect costs to be lower in the non-holiday seasons as there'd be a substantial uptick in visitors. Or if you're a website that provides a constant service like a streaming platform you'd expect your costs to be fairly consistent (not accounting for growth) for all times of the day or year. As for cost predictability for containers and serverless; containers have a very definable cost as you'd have to run the servers off your own machines and power them. Thus, the cost may be higher than the alternative but consistent and possibly wasteful. Serverless would have to adjust depending on usage as I just mentioned and thus would be less predictable.

For expansion with serverless you'd just be paying for however much more resources you'd be using, thus it'd be lower upfront cost but easier to implement as the resources are readily available at any time. For containers you'd have to purchase more machines and power them all, upkeep them, and make sure you future proof decently far ahead for higher traffic. Thus, there'd be machines that are sometimes not in use but costing you money. However, that information would be stored locally meaning it can be easily manipulated with or without the internet. Though, you'd also have to purchase a lot of storage.

This is why elasticity is important for expansion, serverless offers elasticity for a lower up-front cost and naturally integrated into the system. That elasticity is what directly determines what you'd be paying, or the pay-for-service model, to upkeep your website.