Jacob Bayer

SNHU

4/24/2022

CS 470 Final Reflection

https://youtu.be/6s4Gso0452Y

Throughout this course we have learned about many relevant technologies, the knowledge of which will undoubtably help me throughout my career. We learned about several deployment mechanisms for full stack applications and implemented projects using containerization with Docker and Docker compose, as well as serverless using the AWS platform. This goes along well with my personal strengths as a developer; I excel in creating functional code like backends or tools that perform an action that does not require a very sophisticated visual design. I find it easy to take a problem, break it into chunks, determine what technologies would work well for it, and figure out how to stitch the pieces together. I would love to get a role as either a backend developer for either web applications or mobile, or possibly something along the lines of PCB development, interfacing technology with the real world. To me it seems like my aptitude for problem solving would still be applicable, just adding a physical element to the work.

When it comes to scaling a web application, technologies like serverless and containerization make it very easy to achieve these goals. If working with containerization, you can use orchestration to quickly set up and deploy necessary pieces of the application to run on more servers and using load balancing to split traffic between all the servers. Cost is an easy thing to determine in this situation, as you know what server space you purchased (either in-house or cloud) and you would know how much that space costs. With serverless, Scaling is much easier, as AWS will handle load balancing as the requests to your Lambdas increases, but costs can be variable depending on how much traffic you are getting. With containerization, you would need to dedicate an increasing number of man hours to ensure smooth operation, as you would still maintain some control over your backend infrastructure, but serverless is very elastic and will scale well without too much help.

These are both very good technologies to choose, and which one to use is situational. Containerization lets you retain control of your backend infrastructure, at the cost of manpower, can be deployed quickly, and has very predictable costs. Serverless is quicker and easier to set up, scales easier than containerization, and is less hands off allowing you to focus on code rather than how to run it, but you lose some flexibility in the operating environment and can have variable costs depending on how much traffic your site receives. How the factors of elasticity and pay-per-service models' factor into the decision is, again, situational. High elasticity is great, but only for projects you want, need, or expect to scale quickly. If you don't mind losing elasticity for increased control, containerization is a good choice. If it's the other way around, serverless. If you know how much traffic you're likely to receive and thus know how much server power you will need and want or need tight control of how much you spend, containerization is the best bet. If you want to prioritize quick scaling and that factors in more than cost, the pay-per-service model of serverless architecture is the best choice