Brandon Lingenfelter

10/21/2022

CS 470 Final Reflection

Link for presentation: https://youtu.be/7vhg1h94ksA

The completion of this course has provided me with the experience necessary to become a more rounded developer. I believe that the more aspects of the industry a developer is exposed to and gains experience in will only help provide them with more opportunities in the future. This course added tools and experience to my resume that I think when added to my resume will make me a more marketable candidate. I would describe my perseverance in solving problems as my strength when it comes to software development. I feel that this course has provided me with the experience to assume a role in cloud migration or cloud development.

Microservices allow for concurrency and partitioning which lets us break our application up into smaller pieces that can run in parallel to each other. Utilizing microservices with scaling by using a serverless model that allows us to pay per the data storage we use and not extra until we need to expand. Error handling becomes easier in the microservice model since we have broken our application down into smaller, independent services that can be altered or updated without making major changes to the application as a whole. While using containers may provide more predictability when it comes to cost of the infrastructure we will need, it does not mean that you can not come up with an estimated figure for going fully serverless. With containers we have to pay for the servers for the entire time our application is running. This is a predictable reoccurring cost assuming you are not scaling. Serverless will charge per use, that is we get charged when our code executes. If we have an idea of how often our code will be triggered to execute, we could come up with a figure that would provide us with the estimated cost.

The plans for expanding the application and scaling the infrastructure will rely completely on if the application provides enough utility to justify the cost or if you are adding utility to the application that can be justified against the expense of scaling. If the application can not adequately prove the need to scale or add utility than the expansion would not be a necessary expense and would not be wise.

Cloud elasticity can result in great savings for an application that sees surges in demand or steep declines in usage. The elasticity model provides access to resources when demand surges and lets you minimize resources when there are declines in demand. Cloud elasticity may not be benefit an application that does not see drastic changes in demand for infrastructure. Pay per service is a model that charges for as much infrastructure is consumed by the application, like a utility bill. Elasticity may be a great model for rolling out an application that has some notoriety and may see a surge in usage. Pay per service might better serve an application still in development or with a fairly fixed demand.