Matthew Wilt

12/15/2022

CS 470

MOD 8: Final Reflection

https://www.youtube.com/watch?v=w2cvHdmpwuo

This course has helped introduce me to the core concepts of cloud development. I believe this will help in my career around cloud-based concepts. The future seems to be trending into more cloud-based applications and connectivity. These skills I have gained set me up to be skilled in cloud-based operations, at least to be familiar with where they fit together. With so many options, I've learned to slow down and think about the full process. Sometimes each process will need its own system analysis report to fix any issues and be better prepared for its integration. One strength that I have recognized lately is my ability to problem solve difficult ideas. When I had speedbumps in Project One, I would take my research and learning only so far before reaching out. I will say, sometimes I didn't know if the guide was intentional about that, but it confused me when I inputted some code. I can see the big picture better than I can all the minute details. Professionally I'm interested in System Design and Analysis. "Why this and why there?". I believe there can be many roles I could assume soon. These could be working on requirements, outlining the main components of the system, coding where needed, and testing the system. I do think it's also good to be more focused on a few components that work together seamlessly.

The use of Microservices or serverless is only growing. The ability for developers to package small separate pieces of code that will activate on a certain event means we can manage these smaller, independent systems more efficiently. Adding features and scalability suite itself well in this environment. No longer will we need to setup extra server hardware. With serverless, that is all handled automatically. If we have an app with many microservices involved, it can be a challenging job to handle errors. Here we can incorporate some cool functions in AWS Lambda. These are called AWS Step Functions. Instead of writing error

handling logic for every Lamba function, we can incorporate these Step Functions into our workflow. When certain errors or failure occur, the Step Functions can catch and retry with the inputted code. This also allows a space to easily debug the app, while most times preserving user experience. Predicting the cost is a tough ask. But we can create some baselines to know what we will pay. We have one user run through the app's functions once. This will be a baseline. Then we estimate 5k users from there etc. The cost for Containers is more easily predictable. We know that we will provision a machine that runs 24/7, if it can handle the app, this cost will be a specific number, no matter the fluctuation of users.

A few Pros of pursuing expansion is that we know we will only pay for what we use. This is a great model for an app with many users. We also don't need to worry about time spent upgrading servers and hardware. Our time will be better optimized designing and deploying the code. Microservices aren't a perfect solution for every scenario. It can add up to some headaches. You have more code and services in more places with multiple teams. Another is each new microservice will have its own set of costs. If not managed well, this can get costly. With that said there will be a need to add more personnel to handle any issues, errors, and updates within the microservice teams.

The two roles of elasticity and pay-for-service are a big deal when it comes to the cost of future growth. We need our service available no matter the spike in use. Serverless achieves this for us. Paying for what we use makes the most sense. If our services aren't being used, we know we aren't incurring any cost. When planning, we know that as we get bigger our servers and hardware will expand to fit the need.

Bibliography

Amazon AWS. (n.d.). *How do I handle errors in serverless applications?* Retrieved from aws.amazon.com: https://aws.amazon.com/getting-started/hands-on/handle-serverless-application-errors-step-functions-lambda/

HARRIS, C. (n.d.). *Microservices vs. monolithic architecture*. Retrieved from atlassian.com: https://www.atlassian.com/microservices/microservices-architecture/microservices-vs-monolith

www.graycelltech.com. (n.d.). *What is Serverless Architecture? Why use it and its Advantages and Drawbacks*. Retrieved from www.graycelltech.com: https://www.graycelltech.com/what-is-serverless-architecture-why-use-it-and-its-advantages-and-drawbacks/