Jacob McPherson

4/28/2024

CS 470 Final Reflection

https://www.youtube.com/watch?v=nG0G1OGt-Ck

Allow me to begin this by saying this class has been quite an experience for me. I've been given a firsthand look at how applications can be packaged into images for container use, on top of how to develop and link cloud services together to create a cloud application that can be accessed anywhere. With this I've improved my understanding of cloud security and the intricacies of setting up a secure API. Not only this but I've been granted the opportunity to learn several new development tools like DockerCompose and DynamoDB. Now I have a small list of development programs along with three major programming languages under my belt to impress a new employer. Even if I don't know the programs used by my new employer, my class history and tenacity should show that I can learn any language or program necessary to get the job done in a timely manner. I'm ready to begin working alongside other developers to improve legacy applications, create new software, or even do general IT work to get my foot in the door at the right company to continue building my skills.

Looking towards the future, I know that one day I'll be doing work on a web application that will be utilizing cloud services. Scaling such an application would be rather simple if services like AWS Lambda and Gateway were employed in which scaling is handled for you if your functions are made to be scalable. Handling the scaling on my own I would use MongoDB to handle my database due to the available complexity of query parameters that would allow me to further refine user searches. To keep user errors to a minimum I would first create policies to restrict user access to delicate resources, along with rules that would prevent users from searching with certain key characters that could lead them to "accidental" access of previously restricted resources. Where management becomes tricky is figuring out the price to maintain my cloud application. This would heavily depend on the type of application I plan to be running. A large interactive database that users will be adding on to continuously will require a fast-scaling database that will likely fall within the pay-for-service method of payment. This would result in a steady price increase over time that could be easily mapped out as more time passed. On the other hand, if the application is based more around a service that doesn't require saving new data all the time, then it could be more cost effective to use containers and save what information is needed locally on the user's system. Alternatively, local servers could be maintained to handle the core data while a smaller cloud service works as a distributor for a containerized image. This would lead to a more predictable and consistent cost as the need for an expanding database is much lower and maintenance could be kept in-house.

When we finally look at expanding, we must weigh in the benefits of one method of service compared to the others. In this case how much of our application we want working from the cloud. Fulling moving your application to the cloud can make it easily accessible to users anywhere in the world, however you then have new security concerns circling around the now numerous access points to your application and its' inner workings. This can be offset a bit by the ability to push out updates and changes whenever necessary. Not only does this make it easy to spread bugfixes to everyone quickly, but it also means your application will have next to no down time necessary for updates to be implemented. This can also serve as a drawback in that if an update were to break your application or cause a major function to stop working, then every user would have that same issue shortly after release which could cause a massive surge in call ins or reports.

Two of the biggest obstacles when it comes to the future growth of an application or service is the ease of scalability and the cost. For almost any application with a userbase that is ever expanding you'll want your application to be able to handle that expanding userbase without slowing down or losing integrity meaning you'll need more resources. On this same note you want your application to also be able to scale down when you run into "drought" times where user frequency is at its' lowest. This means clearing up resources that aren't being used to lower your active resource count. This plays into a payment model called pay-for-service, which means a cloud provider will only charge you for the resources actively being used at any given time. Keeping this payment model in mind, by making scalable applications companies can expand to handle more and more users while minimizing the number of resources that need to be maintained and paid for consistently. Keeping to this also means that any expansion or service extensions can have their costs calculated based on their potential user activity. This can really help with the budgeting when expansion is initially brought up for discussion.