Trevor Mickelson

02/29/2024

CS 470 Final Reflection

https://www.youtube.com/watch?v=BA1lONRnmvc

**Experiences and Strength**

- In this course I learned what it means to take a full-stack application and migrate it over to the cloud. I've done this within a serverless environment using containerization, orchestration, lambdas, API gateway, and databases.
- I believe my strengths as a software developer revolve around my ability to learn new things, and or adapt what I already know to some other sort of system. An example of this would be transitioning from SQL to something like MongoDB. In principle, they're both still just databases. Another more relevant example would be migrating from a self-hosting machine to a cloud environment.
- In my new job, I believe I can handle most new incoming roles. I don't mean this at a professional level, but certainly at a junior level. I've written software in many languages such as Pythin, Javascript, Java, etc. And I've Interacted with many databases, and even the cloud.

**Planning for Growth:**

- I would handle scaling depending on the situation at hand. In some cases, the cloud isn't a viable option. Maybe it's too complex to switch to, or maybe the service running can't easily run in the cloud, an example of this would be a Minecraft server. In this unique case, I would scale manually using dedicated machines. However, if it's possible, I would solve scaling through the cloud, and migrate the application to the cloud.
- I would handle error handling just as any other developer would. The software should be riddled with error handling, otherwise, when something does go wrong, it will be extremely difficult to determine the actual problem.
- I can predict the cost of dedicated machines via the price of the machine. However, when it comes to the cloud, the cost could be predicated based on the resource consumption, as it wouldn't be a fixed price.
- Containers probably are the least predictable in this case, as they could still contain chaos, whereas a serverless environment is handled for you. However, why choose? Why not just run containers in a serverless environment?

A pro to expanding to the cloud is cost and management. A lot of the backend is handled for you through permissions, serverless functions, and scaling. The con in this case would probably just come down to time and complexity. For example, if the team of current developers doesn't know how to use the cloud, then transitioning from an already working application to saving some money probably isn't worth it.

The roles of elasticity/pay-for-service play probably the most relevant role in planning for growth or migration. This handles the scaling up and down for you, and the cost is only based on what is used. Without this existing, there wouldn't be a lot of reason to transition to the cloud.