Bryant Franks
6/25/2024

CS 470 FINAL REFLECTION

https://youtu.be/nFKMPe3g_wM

As a software engineer who specializes in UI/UX development, this class has helped me to understand aspects of software deployment that I have rarely had to interact with. My experience is in Angular client-side development. Creating reusable components, and services to create custom forms for lead generation purposes. My focus is on creating a cohesive experience to gather and pass the required data to several other internal services to service these leads.

As such, my daily tasks rarely involve handing hosting. Our applications use AWS extensively, and this course has given me an excellent introduction to concepts and terms I generally only hear in passing at my work. How an application implements lambdas to pass information to DynamoDB has generally been the realm of other engineers in my organization, but with what I have learned in this class, I can begin to understand what would be required if I transitioned to a more dev/ops role.

In my experience, various aspects of applications I work on have heavier traffic and utilization than others within the greater breadth of the application. So, scalability across the application may not be linear and across the board. Utilizing microservices allow us to scale what is needed, while also localizing potential failure points, mitigating catastrophic risk across the entire application.

In terms of cost predictability, containerized services prove to be more predictable, as you're usually paying up front for storage and usage, whether you maximize that use or not. Whereas most serverless environments calculate cost through usage. While serverless options can potentially be more cost-effective, the predictability of this is lesser, depending on outside factors.

When considering expansion and scalability, an organization must consider several factors. What is the volatility of expansion? Is the organization expecting predictable incremental growth? Would the application be capable of rapidly scaling up and down as traffic changes? In AWS, elasticity, and payment through a pay for service model offers flexibility for these unknowns. It removes forecasting potential storage, and computing needs which may prove to be inaccurate. Growth can be an unpredictable thing, and keeping your systems elastic, and only paying for services you use allows a company to remain prepared, while avoiding unnecessary spending on services if you overpredict usage.