CS 470 Final Reflection

Nicolas W. DeFrancisco

Southern New Hampshire University

https://studio.youtube.com/video/7QEhufgiebk/edit

Experiences and Strengths

In this course we learned some fundamental concepts of cloud-based development, how to containerize an application and migrate it to the cloud, and how to deploy the application in the cloud using various tools and cloud frameworks. I believe this class will help me with my professional goals as a junior developer because I got some exposure to working cloud-based tools and technologies. I was very impressed with the AWS management console as well as the tools it had to offer. The course reinforced my understanding of application architecture and how components are connected. It also reinforced the concept of APIs and how data flows through a cloud application.

My strengths as a software developer are that I'm good at solving problems. I try to do the best that I can to understand what is going on under the hood and make attempts to correct them. I also like to reference programming language and framework documentation to search for answers to challenges I encounter. I also can leverage my experience from my graduate level course work in business and apply systems-level thinking to application design and modelling.

The type of role I'm prepared to assume right now is a junior developer. I'm completely new to this industry and I believe I need to get some experience working in an IT or dev

environment. I have about 5 months left on active duty and plan to keep working on my own projects to continue building my portfolio. I want to make myself marketable enough to attract good employers so that I can continue on my software development journey. I got into software development by accident. This was never a plan for me until I was halfway through my MBA capstone and realized I had hole in my skillset: I didn't know as much as I believe should have known about the inner workings of tech firms. Sure I studied Amazon from the beginning of my MBA coursework until the very end; however, the tech side of their operations were not as easy to understand and I wanted to know how they leveraged technologies early on and were able to monetize it giving them additional streams of revenue while enhancing their market competitiveness. Ever since I started this program, it really grew on me. I enjoyed solving problems and building mini-projects that were able to run.

Planning for Growth.

Microservices split applications into distinct independent services. Each service is managed by individual groups that can work independently on individual services without impacting the work of other developers in other groups working on the same application. This allows scaling and growth simultaneously across services. Because of the modularity, errors or bugs are restricted to the service they reside in. Code discrepancies are isolated from each other, making it easier to manage.

The serverless model is a highly scalable cloud-based model that enables developers to focus solely on application development. The servers are managed by a cloud provider so that developers don't have to worry about all of the server management and configuration tasks. This separation of responsibilities is what enables rapid scaling.

Predicting cost becomes easier with serverless and microarchitectures when moved to the cloud. This is enabled by the pay-as-you-go pricing model in which you pay only for the resources that you consume. This eliminates painful upfront costs of using traditional data centers. Organizations can instead scale rapidly within a cloud environment. The pay-as-you-go model and the elasticity of the cloud environment provides a predictable approach to forecasting costs because capacity can be expanded with a rise in user sessions or contract during periods of low traffic. This is especially true for the serverless architecture because they are event-driven.

The strategies discussed have different drawbacks associated with each of hemThe serverless architectural have execution time limits, which may affect the performance of long-running tasks. Containerized microservices my require more upfront costs to setup and manage the infrastructure.

Containers have a predictable pricing model based on infrastructure usage and allocated resources. Containers may have more resource utilization overhead compared to serverless because of the need to allocate maintain resources for each instance. Containers are not as scalable as Serverless because they require provisioning additional resources for peak loads, thus incurring higher costs. Serverless cloud automats scaling based on incoming events to optimize resource allocation and cost efficiency.