Shashona Robinson

CS 470 Full Stack Development ll

02/27/2024

CS 470 Final Reflection

YouTube presentation: **https://www.youtube.com/watch?v=9ikULCKLopI**

Experiences and Strengths

In Full Stack 2, I learned how to containerize a full stack application, orchestrate its micro services, and how to migrate that same application to the AWS cloud. The result of the migrated website to a serverless architecture running in AWS, hosted on an S3 bucket, can be found here:

http://ndsnhu2.s3-website-us-east-1.amazonaws.com

As a developer, I excel in swiftly acquiring proficiency in new libraries and languages, a skill honed through exposure to a diverse array of them during my studies at SNHU. Moreover, I possess a multifaceted skill set, extending beyond software development to encompass graphic design, 3D modeling, and animation.

An illustrative showcase of my capabilities is demonstrated through my CS330 project. In this endeavor, I crafted an OpenGL 3D renderer from scratch. Utilizing Blender for modeling and texturing, I meticulously designed all objects within the scene. Additionally, I engineered both the renderer and a bespoke model loader, complete with its own file format.

https://github.com/robinsonshona/Comp-Graphic-and-Visualization

In a new position, I'm fully equipped to take on the responsibilities of a Developer role within an AWS project, drawing upon my Android development experience. I'm deeply passionate about both fields and eager to contribute my skills and expertise to any project.

Planning for Growth

Scale and error handling.

To manage scalability and error handling effectively, I'd implement a fallback page to which erroneous requests are routed via the API gateway. In terms of scalability, minimal intervention is required since AWS inherently scales the application seamlessly.

How would you predict the cost?

The prices for the Lambda service are listed here:

https://aws.amazon.com/lambda/pricing/

The cost of utilizing 1 millisecond of Lambda function with a 128-megabyte memory allocation is $0.0000000021. Given that the Angular example website focuses solely on CRUD operations for user-submitted text, maintaining the 128-megabyte allocation is optimal. This ensures predictability in the cost of the serverless website, particularly in terms of the expense per individual request.

To accurately forecast costs, it's crucial to understand the historical traffic volume handled by the website. By analyzing its growth trajectory, it becomes feasible to predict the anticipated number of requests over the forthcoming months. This understanding forms the cornerstone of precise cost projections.

What is more cost predictable, containers or serverless?

One could argue that serverless containers offer greater predictability since there is always a requirement for underlying infrastructure, whether it's an on-premises server or a virtual machine in the cloud, to instantiate them. Nonetheless, for applications suitable for containerization, the more cost-effective option is likely to be serverless, which remains reasonably predictable, as previously discussed.

Expansion and Elasticity

When deciding between containers and serverless architecture, the choice depends on the specific project requirements and budget constraints. For applications requiring continuous, long-term operation, such as inventory or transaction systems for retail stores, utilizing containers on local servers is often more prudent. However, for customer-facing websites, a serverless approach is typically preferable.

Considerations such as the nature of the service, anticipated growth, and capacity demands are crucial when evaluating elasticity. Local hardware deployments can struggle to adapt elastically to fluctuating demands, whereas serverless architectures inherently offer greater elasticity. However, while pay-for-service models handle elasticity more seamlessly, they may be less cost-effective depending on the scale of the business.