Brian Carnrike

CS470

10/21/21

CS 470 Final Reflection

YouTube presentation: https://youtu.be/FHw6xarvuFk

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 example website used was the "Learn Angular Step by Step" website found here:

https://github.com/AngularTemplates/learn-angular-from-scratch-step-by-step

The final result of the migrated website to a serverless architecture running in AWS, hosted on an S3 bucket, can be found here:

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

As a developer, my strengths are learning new libraries and languages quickly (thanks to wide variety of them taught at SNHU), and I also wear many hats. I'm not only a software developer, I'm familiar with graphic design, 3d modeling and animation.

One example of some of my strengths is my CS330 project, where I created an OpenGL 3D renderer. I modeled and textured all objects in the scene in Blender (3d modeling software), wrote the renderer and my own model loader and file format.

https://youtu.be/g65SJwUaEys

In a new job, I would be capable of assuming a SysOps or Developer role on an AWS project, but I have experience in game development (Notably, Unreal Engine 4 experience) as well as Android development. All of these fields are of interest to me.

Planning for Growth

Scale and error handling

To handle scale and error handling, I would ensure that there is some kind of fall-through page that faulty requests are directed to through the API gateway. For scale, not much needs to be done as AWS scales the application nicely on it's own.

How would you predict the cost?

The prices for the Lambda service are listed here:

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

The cost of 1 millisecond of Lambda use, with 128 megabyte memory allocation, is $0.0000000021. The Angular example website deals strictly with CRUD operations on user submitted text, in the form of questions and answers, so sticking with the 128-megabyte allocation is ideal. This makes the cost of the serverless website pretty predictable, in terms of how much a single request cost. Knowing how much a request costs, the key to an accurate cost prediction is understanding the amount of traffic that the website has handled in the past. By following its growth trend, predicting the number of expected requests over the next few months is possible.

What is more cost predictable, containers or serverless?

Arguably, serverless containers are more predictable- simply because there always must be machines and running to instantiate them, whether it be a server on location or a virtual machine running in the cloud. However, the more cost-effective solution for an application that can be containerized is likely to be serverless, which is still relatively predictable, as discussed above.

Expansion and Elasticity

When it comes to deciding whether to use containers or serverless, it depends on the project and budget. In some cases, if It's known that the application will need to be running practically forever, then it's wiser to use containers on a local server. Something like an inventory or transaction system used in a brick-and-mortar retail store. But for a customer facing website, a serverless approach is a better decision.

At the end of the day, the type of service, expected growth and demands placed on capacity are factors that should be given thought regarding elasticity. Being elastic is more difficult when using local hardware. But in pay-for-service models, elasticity is already handled- but might be less cost effective depending on the scale of the business.