Randall Rash

8/22/2021

CS 470 Final Reflection

https://www.youtube.com/watch?v=3jLom9Wzzc4

Throughout my learning in CS 470, I have gained a much better understanding of the processes required to host on a cloud-based platform. This information will be vital to performing well in an industry that is moving toward cloud-based architecture across the board. I believe that the knowledge I have gained for cloud hosting is something that most employers are looking for in the field. I believe that being able to show that I have exposure to Amazon Web Services will help a lot with gaining a job in the market. I have specifically gained a lot more skill in running containers. Not just how to create a container, but how they work, and how they can be implemented in a structure like AWS. Containerization was a foreign concept to me prior to this class, but the exposure that I have received has taught me a lot about them. Another skill that has been learned in this class is the idea of API management. Prior to this class I would have performed API management completely wrong. Knowing now that APIs are really the backbone to making a site work as well as a security essential, I will implement them properly in the future.

As a software engineer, I feel that I have one of the same strengths that most have, which is persistence. Seeing something fail to work for the 100^{th} time is something that will happen all the time in this profession. Another strength that goes together with persistence is patience. I believe that both are vital in a software engineer. These allow you to keep testing things, and keep watching them fail, to learn how to do it better. With these strengths, as well as a knowledge of object oriented programing, I feel that I'm ready to take on roles within the field.

The roles that I believe I could maintain would be, front-end, back-end, and full stack. While I typically prefer back-end development, I feel confident that I could assume the role of front-end developer as well.

Microservices and serverless can be used to produce efficiencies that make maintenance a lot easier. Scaling the resources needed to meet the demand of the website is all handled by the provider of the cloud service. This ensures that there is no need to keep expanding your own server as load increases. Error handling is something that is handled in a limited scope by the provider as well. Any errors with the system are handled by the provider. If there is downtime, most providers have separate storage sites that can reboot your site within minutes. Predicting the cost of traffic to a website would be trickier with a serverless model. With a serverless model, you are looking at a variable cost based on the amount of traffic that is hitting the site. You can estimate the cost of your serverless hosting by using the average number of hits on your site and maybe err on the side of caution and go slightly above that. This will give you a ballpark range, but you will be at the mercy of number of visits. Using containers is something that could give you a straightforward estimate on costs, because there isn't downtime. That server will always be running. This costs more than serverless but gives you a more exact number than you would get from hosting serverless. Overall, I think that containers are better for cost predictability, but are also more expensive.

When expanding, there are several pros and cons that can be observed. Some of the pros are that cost decreases, avoiding unforeseen downtime, and you also only pay for the storage that you need. Some of the cons are that you would be less secure in the cloud, potential cost fluctuations as traffic increases or decreases, and having a smaller amount of control over customer service. These things together should all be considered when planning an expansion.

Elasticity and pay-for-service are two things that must be considered when looking at future growth. If the site is hosted locally and growth is happening, you have to constantly upgrade your servers to stay ahead of the traffic load increase. This can have monetary impact though, as having too much server space, results in overpaying, while having too little results in customers having a bad experience. This is solved with pay-for-service, which scales the amount of server space according to traffic flow. This ensures that you spend as little as possible while still having the resources to meet the requirements of your end users.