Brooklen Ashleigh

CS470

Professor Tepe

October 27th, 2024

Assignment 8-1 – Final Reflection

Youtube Link: https://www.youtube.com/watch?v=ZMnKnHhiU6c

Experiences and Strengths: Explain how this course will help you in reaching your professional goals.

- What skills have you learned, developed, or mastered in this course to help you become a more marketable candidate in your career field?

After this course, I have a much better understanding of how full stack applications can be deployed using serverless AWS services. Though I've heard a lot about serverless applications, I'd never seen on set up within AWS, and now that I have a good idea of the features and possibilities, I want to know more about the constraints and limitations of the platform. Naturally, the personal projects that I am eager to take on and build to gain experience with web development will have a cost prescribed to them, but I would like to see what is possible from the platform.

- Describe your strengths as a software developer.

My strengths as a software developer are my ability to adapt and take on new concepts and build methods. One of the hardest things about getting into web development for me is the reliance on the work of others through frameworks and platforms made to abstract a lot of the work away from developers. While this saves a lot of time, it means the initial learning process is more difficult. Sometimes it's hard to be successful without good literature or a tutorial that shows you the overall intent of a framework before you start using it.

- Identify the types of roles you are prepared to assume in a new job.

I would love to begin work as a software developer, either as a frontend or backend engineer. Fullstack has always appealed to me because I love the marriage of the frontend and backend. I think web development can be pushed to be more creative and interesting to suit its users. I believe I have the school set to begin learning how custom applications are built on demand for customers that need dynamic websites.

- **Planning for Growth:** Synthesize the knowledge you have gathered about cloud services.
    - Identify various ways that microservices or serverless may be used to produce efficiencies of management and scale in your web application in the future. Consider the following:
        - How would you handle scale and error handling?
        - How would you predict the cost?
        - What is more cost predictable, containers or serverless?
    - Explain several pros and cons that would be deciding factors in plans for expansion.
    - What roles do elasticity and pay-for-service play in decision making for planned future growth?

One of the biggest complaints I've seen about pay for usage in the case of AWS and websites like Vercel are that they can have unexpected cost spikes. Amazon has a price calculator, which can be used to predict the cost of various services, but the developers starting the project must have an idea of the traffic that the service will be seeing to know the costs of usage. In a commercial setting, a company will likely have some basis and past experience to bridge from when launching a new service. As with any other type of production, the cost of the platform must be balanced for its use case.

Taking advantage of AWS's elasticity is one of the most straightforward ways to scale and grow a platform with usage. In a non-serverless environment, an ecosystem that can add servers and deploy new pieces of an application to suit traffic would be harder and riskier to manage. And even then, those costly increases are in steps, where AWS can match traffic as needed, only ever charging for what is used, which avoids the problem of overshooting or overallocating resources.

I would argue that containers have a more predictable cost because of that step-based allocation. Traffic and usage are still variable, but the server resources generally have a specific cost for the resources allocated to the instance, with traffic on top. VPS can offer static costs, as well, though that has limitations.

Scaling and error handling can both be handled by AWS services, at least on the server infrastructure side. That's one of the benefits of serverless—the server resources don't have to be managed. Additionally, AWS has a large catalog of services that can help with things like logging and log analysis to check for problems and help define them. I believe the risk with serverless errors is that a lot of the settings are within AWS's infrastructure, and fixing problems requires managing those settings within AWS, rather than pushing them with a code base. I'd like to learn more about the downsides of serverless. For example, I've heard that updating microservices that rely on others can be a nightmare when pushing changes in production.