Module Eight Assignment

Nathan Anglin

In this course, I applied my previous knowledge of full stack development and migrated an application to a cloud service, AWS. I found this process to be interesting and found the upsides to moving web applications to the cloud to be numerous. The following is a link to my final thoughts on the process.

YouTube Presentation URL: https://youtu.be/dqNZB03alsg

- **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?

    As with each class, I feel like the greatest experience I gained was learning something completely new to me. I had limited experience with cloud-based platforms, and by the end of the class, I feel that I have a good grasp on developing programs for AWS. I think that this quality is worth as much as any individual piece of knowledge, as adapting to a situation and learning something new is more important in the end.

  - Describe your strengths as a software developer.

    As a software developer, I feel that my best strength is being able to teach myself something new, with little or no previous knowledge going into the project. I feel that this is a good skill to have, as there are too many programming languages, frameworks, and middleware programs to be proficient in all of them at once but knowing how to learn and adapt are good skills to have.

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

    I would love to continue this work during my career. I think that programing in general is very interesting, and being able to see my work play out before me is very satisfying. With web development, this is quite prevalent, and I really do enjoy the work.

- **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?

    I feel that containerization is a great tool to handle scaling applications and being able to adapt to changing demand. With error handling, I feel that a good base will continue to pay dividends in the product, if you don't become lax in your ability to test and retest your code.

  - How would you predict the cost?

    I feel like this is a lot easier with cloud-based platforms, where your cost is based on usage. I think putting a bigger emphasis on how much money your business or application would be making vs the cost to run the service per person is the best bet. When working with a non-cloud solution, this becomes much harder, and finding the right amount of infrastructure to host your service requires more work.

  - What is more cost predictable, containers or serverless?

    I feel that serverless solutions are much more predictable, and you can really nail down the amount of usage per end user. Containers provide great expandability, but still take up resources when they are not in use.

- Explain several pros and cons that would be deciding factors in plans for expansion.

  I think based on the future growth of a company, and how set the platform is, is a big consideration. If you are not sure that you want to stick with cloud-based services, I don't think I would recommend starting with AWS. It seems to be very siloed and would be hard to move your application away from AWS if you decide to change providers. With containers, however, you are free to move your application to many different hosting services and have more future expansion capabilities.

- What roles do elasticity and pay-for-service play in decision making for planned future growth?

  Knowing how much traffic your site or application will create is a hard thing to do, especially with a company with big release dates. You may have a couple of days of intense traffic, followed by a trickle of customers afterwards. In this case, a pay-for-service helps you manage this, where the elasticity of a container-based system may be better for a more metered expansion of a commonly used and relatively stable demand service.