Kathryn McNeil

4/23/2023

# CS 470 Final Reflection

Southern New Hampshire University

https://youtu.be/ANy7LrZ6KFA

**Experiences and Strengths**

Throughout this course I have learned how to migrate an application to the cloud though the use of containerization and cloud providers such as Amazon Web Services. I have gained experience in Docker as well as Docker Compose by containerizing my application. I then worked with S3 buckets to upload and store data for my application. Next, I created Lambda functions to add and retrieve information from our database, DynamoDB. In DynamoDB I created tables to store my questions and answers that would be displayed in the application. Lastly, utilizing API Gateway I created methods for my Answers and Questions table and deployed my API. Working with these tools has given me hands on experience in migrating applications to the cloud. As cloud development continues to gain popularity, these skills will prove to be invaluable to me as a developer.

As a software developer, my strengths lie in my persistence and desire to continually learn. Throughout this course I encountered numerous problems which took time to overcome. I had to spend time researching, as well as trying various solutions to solve my issues. This would not have been possible without persistence. There was also quite a bit of collaboration that took place to problem solve as well. I worked with my classmates, sharing issues and possible solutions to overcome obstacles. I displayed my desire to learn through this course as I faced many new topics and tools I had not worked with before. Through this course I was able to grow as I researched and studied the things that were new to me.

In a new job I am prepared to take on the role of a full stack developer, or a backend developer. I have experience not only coding applications, but also migrating them to the cloud. I have worked with databases, development platforms such as angular, as well as the

containerization tool Docker.  I have also used various programming languages important to web application development such as JavaScript, Python, and HTML. I have practice in building and deploying APIs as well as securing the application through setting permissions and roles. I am also knowledgeable about the Agile development model, one of the leading development models of today. I have experienced various parts of the development lifecycle making me a valuable part of any team.

**Planning for Growth**

Both microservices and serverless can be used to produce efficiencies of management and scale in a web application. Microservices are standalone services that work with others to make up an application. Each microservice is a small, contained portion of an application that interacts with other microservices to make up a whole. This is advantageous to developers as they can work on small portions of the application at a time, making changes without having to alter the whole thing. This provides a way to make changes more safely as you are working on a smaller scale, and it also enables developers to create more complex systems that are also highly scalable.

Serverless applications improve efficiency as you do not need to manage any infrastructure, nor provide resources to secure servers. Cloud providers such as Amazon Web Services provide options for autoscaling and traffic management, further increasing efficiency while also managing scale and error handling. Both microservices and serverless support horizontal scaling, a type that has the potential to scale indefinitely. This is a desirable trait as it means that your application can support increases in demand without compromising performance.

When it comes to microservices, each one handles its own errors. This means it is easy to find and troubleshoot errors. Serverless services can automatically manage errors such as out-of-memory exceptions or timeouts. This significantly improves efficiency as developers do not need to waste time on trivial errors. Another benefit of going serverless is that you can predict what your cost will be. Each service provided by AWS clearly states its prices, and you can also set limits, so you do not use more resources than desired. This makes it easy to determine costs, especially when you are dealing with the unknown. It also improves performance as you may not be sure how much traffic you will receive, the cloud provider scales as needed up to the threshold you set so your users have a better experience without lagging of the application, while you are prepared to spend up to a certain amount.

When it comes to predicting costs, containers run at all times whereas with serverless you are only charged for the time your functions are running. It is important to know your audience to get an estimate of how much you will be spending. If you have a lot of traffic and your functions are called often, then it may be worth it to have containers running 24/7. On the flip side, if there will be downtime between function calls then constantly running containers will lead to monetary waste. In this situation serverless could be the better choice as you are only charged per function call.

When planning to expand an application it is important to consider cost, maintenance, resources, and potential growth. What would be the cost of expanding your application? What maintenance would have to be taken on to support this large application, and what resources are needed to make it happen? How much would your company grow due to this expansion? If the cost, maintenance, and use of resources outweigh the expected growth then it is not a good idea to expand. Expanding too soon can cripple young business especially when they have limited

resources such as developers and time. If they cannot properly keep up with the expanded application users will not have a good experience using the application leading to decreases in usage and traffic.

Elasticity is the process of gaining and discarding resources as needed. This is important for the growth of applications. When planning growth, resource needs may suddenly increase (or decrease), and elasticity can handle these dynamic changes. It is best for planning growth and is typically used short term. It is good at managing unexpected demand changes and is often adopted by small companies. For long term solutions, scalability is adopted to meet sudden changes in demand. Pay-for-service is a concept of paying for what is used. As you are planning for future growth it is important to consider how your expenses will change. For smaller companies, it may be cheaper to use the pay-for-service model as you only pay for what services you use. After growth, the application may use significantly more resources increasing costs quite a bit. Understanding how your costs may change is important to be prepared financially, as well as for comparing and contrasting different models such as a flat-rate pricing model.

Citations

AWS. (n.d.). *Microservices.* https://aws.amazon.com/microservices/

Cloudflare. (n.d.). *Serverless computing vs containers: How to choose*. https://www.cloudflare.com/learning/serverless/serverless-vs-containers/

Madhav Mohan [madhav_mohan]. (2023, January 16). Scalability and elasticity in cloud computing. Geeks for Geeks. https://www.geeksforgeeks.org/scalability-and-elasticity-in-cloud-computing/#

Red Hat. (2022, May 11). *What is serverless?* https://www.redhat.com/en/topics/cloud-native-apps/what-is-serverless

Schmitt, J. (2022, May 27). *Serverless vs. containers: Which is best for your application?* CircleCI Blog. https://circleci.com/blog/serverless-vs-containers/