Ozobia Nwadibia

25 April 2025

CS 470 Final Reflection

Link: CS 470 Project 2 Presentation

Experiences and Strengths

This course has been highly informative. I have, in the past, built a full-stack application and deployed it to Heroku. This course presented the first opportunity to make a similar deployment to Amazon Web Services (AWS) instead. So, I can say I now have first-hand experience with AWS. It is a useful skill in today's software development market.

My strengths as a software developer will only improve as I continue to practice what I learn. I love learning new things. This is critical in the software development world. I have a preference for Java. I love its structure and robustness. I also like front-end technologies like HTML, CSS, and JavaScript, along with its derivatives like React. I can comfortably build a website if I need to.

For a new role in software development, I believe I could either work as a Java Developer or a front-end developer. AWS is making me look at Cloud application development. The former two, though, are more likely.

Planning for Growth

Microservices are like miniature applications, in contrast to monolithic ones. Serverless computing implies using a platform to manage the backend portion of applications (no need for servers, since a platform, like AWS, handles all that). This feature makes it possible for apps to scale up or down in AWS. Normally, there would be an alliance between Lambda, API Gateway, and possibly a database server like DynamoDB. As for error handling, it would seem that the logic for it ought to be included in the Lambda functions. This is neither tenable nor pragmatic due to possible failure or service memory shortages. This is where AWS step functions come in. ("Handle errors in Serverless applications," n.d.). They are designed to handle unexpected error events in Lambda code and will prove a lifesaver in dealing with all sorts of faults that may arise during code execution.

The services used or made available by AWS are mostly based on a pay-as-you-go model. The AWS platform makes provisions for users to assess or estimate the monthly cost of using a

supported service. The utility is a web app called the “AWS Pricing Calculator.” It is simple. It is straightforward.

The determination of cost predictability between a serverless setup and a container one really depends on the application’s needs regarding what it is meant to do. Serverless permits developers to focus on code without worrying about managing or provisioning servers. Containers, on the other hand, help abstract a software program and its required dependencies.

This, in essence, means that there is no precise way of telling which model is more suitable. A serverless setup is perfect for applications with unpredictable workloads, while a container setup will be most suitable, pricewise, for applications with steady resource needs.

The idea of expansion could arise when there is a need to make an application ‘more’ available. This could be due to increased demand requiring the supporting service to scale accordingly. Here are a few advantages to help in deciding on a planned expansion:

(a) Lower operation costs – no servers to provision or maintain
(b) Abundance of IT resources – this includes services like API Gateway, Lambda, etc.
(c) Automatic scaling – applications on AWS are designed to scale up or down as needed.

Some disadvantages may include:

(a) Vendor lock-in – a problem may arise if the chosen platform (e.g., AWS) does not possess all the required services, forcing a user to opt for a hybrid approach.
(b) Reliance on the internet – access to cloud services generally requires an internet connection, and disruption in this critical infrastructure could impact access to needed services.
(c) Complicated pricing – the separate service pricing for individual services like DynamoDB (on the AWS platform) could prove complicated and perhaps expensive in the long run.

Elasticity refers to the way an application can scale up or down as needed. This is directly related to the pay-for-service ideal. With the latter, a user simply pays only for app uptime. The roles that both features play in the decision making for planned growth are generally mostly related to use cost. Having a good understanding of the concepts and utilizing a pricing tool can greatly assist decision-making for planned future growth.

References

Bigelow, S. (2025, January 29). *The pros and cons of cloud computing explained*. Cloud Computing. https://www.techtarget.com/searchcloudcomputing/tip/Explore-the-pros-and-cons-of-cloud-computing

Getting started. (n.d.). https://docs.aws.amazon.com/pricing-calculator/latest/userguide/getting-started.html

Handle errors in Serverless applications. (n.d.). Amazon Web Services, Inc. https://aws.amazon.com/tutorials/handle-serverless-application-errors-step-functions-lambda/

Serverless vs containers: Which is best for your needs? (2024, March 12). DigitalOcean | Cloud Infrastructure for Developers. https://www.digitalocean.com/resources/articles/serverless-vs-containers