Final Reflection

Joshua Wozny

Southern New Hampshire University

CS-470 Full Stack Development II

April 25, 2024

Final Reflection

Experiences and Strengths

Skills Learned and Developed:
Throughout the course CS 470, I practiced my abilities in several key areas of cloud development. I learned the use of AWS Cloud Services, which included hands-on experience with AWS Lambda for serverless computing, Amazon S3 for data storage, and AWS API Gateway for managing and securing APIs (Amazon Web Services, Inc. (b), 2022). These skills are highly relevant to my career aspirations in data analytics and data engineering, as they enable me to design, deploy, and scale data-intensive applications efficiently.

Strengths as a Software Developer:
My strengths lie in my ability to integrate complex datasets into scalable cloud applications, optimize data processing workflows, and ensure robust data security. My proficiency in Python and SQL, combined with my newfound skills in cloud architecture, positions me strongly for roles that require both developmental agility and data governance.

Prepared Roles:
This course has prepared me for several roles, including:

- Cloud Solutions Architect, where I would design multi-faceted cloud environments for data processing.
- Data Engineer, focusing on building and maintaining scalable data pipelines in a cloud setup.
- DevOps Engineer, specializing in continuous integration and continuous deployment (CI/CD) practices for cloud applications.

Planning for Growth

Synthesizing Knowledge about Cloud Services:
Planning for future growth involves understanding and applying cloud service concepts effectively, especially the utilization of microservices and serverless architectures. These approaches enable modular application development, where components are loosely coupled and independently scalable.

Efficiencies through Microservices and Serverless:

- Microservices allow each part of the application to scale independently, making it easier to manage and update.
- Serverless computing reduces the overhead of server management and increases cost efficiency by scaling automatically and charging only for the resources used during execution.

Handling Scale and Error Handling:
To handle scaling, I would implement auto-scaling policies within AWS to ensure that the application can handle increased loads without manual intervention. For error handling, AWS Lambda provides built-in error handling mechanisms that can be configured to manage unexpected issues effectively (Amazon Web Services, Inc (a)., 2022).

Predicting Costs:
Cost prediction in cloud services can be complex, but AWS provides tools like the AWS Pricing Calculator to estimate costs based on expected usage. Serverless models are generally more cost-predictable compared to containers, as they eliminate the overhead costs associated with idle compute resources (Amazon Web Services, Inc., 2021).

Pros and Cons for Expansion:

- Pros: Scalability, reduced operational overhead, and the ability to pay for only what you use (Amazon Web Services, Inc., 2021).
- Cons: Potential for increased complexity in debugging and monitoring, and sometimes higher long-term costs if not managed carefully (Amazon Web Services, Inc., 2021).

Role of Elasticity and Pay-for-Service:
Elasticity ensures that the application can adapt to workload changes smoothly without human intervention, which is crucial for maintaining performance during peak loads. The pay-for-service model is fundamental in decision-making for future growth, as it allows for flexible financial planning and ensures that costs directly correlate to application usage, avoiding over-provisioning and underutilization.

REFERENCES

Amazon Web Services, Inc. (a) (2022). AWS Lambda Developer Guide. Retrieved from https://docs.aws.amazon.com/lambda/latest/dg/welcome.html

Amazon Web Services, Inc. (2021). Architecting for the cloud: AWS best practices. Retrieved from https://aws.amazon.com/architecture/

Amazon Web Services, Inc. (b). (2022) Amazon EC2 User Guide for Linux Instances, (Section "Instance Types"). Retrieved from https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instance-types.html