SNHU: CS-470 Full Stack Development II

Module 8 – Final Reflection

Matthew Bandyk – 4/27/2024

https://youtu.be/MCrVpSyqQEs

Experiences and Strengths

Skills Learned, Developed, and Mastered:

Throughout this course, I learned key concepts regarding cloud computing, migrating an application to the cloud, and developing and utilizing best practices in cloud computing security. I developed my understanding of containerization, orchestration as well as AWS solutions like Simple Storage Solution (S3), Lambda, API Gateway, DynamoDB, and Identity and Access Management (IAM). Utilizing these developed skills, I was able to master the execution of roles and policies, ensuring security is at the forefront of my work. All these skills learned, developed, and mastered make me a more marketable candidate in cloud computing.

My Strengths as a Developer:

As a developer, my strengths lie in many components. I am a strategic thinker, always looking at the project holistically. This allows me to ensure upfront planning is successful and a strong foundation to any project I work on. My openness to try and learn new things ensures that I am always staying ahead of the curve with new technologies and solutions to software development. Additionally, problem solving is a strength I utilize on a day-to-day basis, allowing me to dissect a problem into its individual parts to help find the best solutions possible.

Roles I can Assume:

After completing this course, I am prepared to take on many roles in cloud computing. My understanding of Docker solutions has set me up to handle development of containers and managing them through orchestration. My work with AWS solutions has prepared me to work in the cloud development field, managing cloud-based applications, from event driven functionality through lambda and API's to creating and maintaining databases through AWS DynamoDB. I am ready to handle application security, setting up applications through AWS and ensuring they stay secure and are only accessible to those that need access.

Planning and Growth

Microservices/Serverless Architecture Considerations:

Scale and Error Handling

In a microservices or serverless architecture, handling scale involves automatically adjusting the number of instances or resources allocated to each component based on demand. For example, in a serverless environment, AWS Lambda functions can scale automatically in response to incoming requests, ensuring that the application can handle varying levels of traffic without manual intervention. Error handling can be centralized by implementing robust logging and monitoring solutions. For instance, services like AWS CloudWatch can capture and track errors across all components, providing visibility into issues and enabling timely resolution. Additionally, implementing retry mechanisms and circuit breakers within microservices can help mitigate and recover from transient errors, ensuring the overall reliability and resilience of the application.

Cost Prediction (Containers/Serverless)

Predicting costs in a cloud environment involves analyzing usage patterns, resource consumption, and pricing models. Serverless architectures often offer more predictable costs due to their pay-as-you-go pricing model, where users only pay for actual usage. Since serverless services automatically scale based on demand, costs can be estimated based on expected traffic and usage patterns. On the other hand, containers may require more upfront planning and monitoring to predict costs accurately. While containers provide more control over resource allocation, managing container clusters and orchestrating resources can introduce additional complexities and potential cost variability.

Pros and Cons to Consider in Expansion

Expansion plans in a microservices or serverless architecture need to consider various factors. Pros include the ability to scale components independently, enabling faster development cycles and deployment of new features. Serverless architectures offer reduced operational overhead and cost efficiency, particularly for applications with variable workloads. However, cons may include potential vendor lock-in and limitations in terms of customization and performance optimization. Containers provide more flexibility and control over the environment but may require more management effort and upfront configuration. We need to weigh these pros and cons carefully when planning expansion to ensure alignment with business goals and requirements of the system.

Roles of Elasticity and Pay-For-Service in Expansion

Elasticity plays an important role in decision-making for planned future growth by enabling applications to scale resources dynamically based on demand. With elastic scaling, we can accommodate fluctuating workloads efficiently, ensuring optimal resource utilization and cost

management. Pay-for-service models also inform decision-making by aligning costs with actual usage, forcing cost consideration and efficient resource utilization. By leveraging elastic scaling and pay-for-service models, we can achieve scalability, flexibility, and cost efficiency in our cloud environments, laying the foundation for sustainable growth and innovation.