

Janece Gates

Southern New Hampshire University

CS 470 Final Reflection

Professor A. Alam

October 13, 2023

This AWS course has provided me with the skills and knowledge necessary to excel as a software developer in various roles. I learned a number of things about AWS services, infrastructure as code, security best practices, and cost management. As a software developer, my strengths include strong problem-solving skills, coding proficiency, and adaptability. With this skill set, I'm prepared to take on roles such as software developer making me a competitive candidate in the software development field. I will continue to learn more about AWS in the future.

YouTube Link: <https://youtu.be/Tk6wNhuSOIw>

Microservices and Serverless for Efficiency and Scalability:

- **Microservices:** Implementing microservices in the web application architecture can help achieve efficiency and scale. Each microservice can focus on specific functions or features, allowing for easy maintenance, updates, and independent scaling. This modular approach improves code maintainability and the ability to

scale individual components as needed. Error handling can be isolated to specific microservices, making it easier to manage and troubleshoot.

- **Serverless:** Utilizing serverless computing services, such as AWS Lambda or Azure Functions, can improve efficiency by enabling automatic scaling based on demand. Serverless functions only run when triggered, reducing operational overhead. For tasks with variable workloads, like data processing or background tasks, serverless can help manage scaling seamlessly.
- **Scale and Error Handling:**
- For scale, both microservices and serverless offer options:
 - **Microservices:** You can employ container orchestration tools like Kubernetes to manage the scaling of microservices. This provides control and flexibility over container instances.
 - **Serverless:** Automatic scaling is inherent in serverless platforms, responding to increased load without manual intervention.
- Error handling:
 - **Microservices:** Errors can be monitored and managed at the microservice level, making it easier to identify and resolve issues specific to a component.
 - **Serverless:** Error handling is often simplified, as serverless platforms handle many operational tasks, such as monitoring, logging, and retries.
- **Cost Predictability:**

- **Predicting Costs:** Predicting costs can be challenging, but it depends on the application's workload and architecture.
- **Containers vs. Serverless:** Containers may offer more cost predictability because you have more control over the infrastructure, but it requires more management effort. Serverless can provide cost savings because you only pay for actual usage, but it might be less predictable for workloads with highly variable demand.
- **Pros and Cons for Expansion:**
- **Microservices Pros:**
 - Modular and maintainable code.
 - Independent scaling.
 - Fosters innovation and experimentation.
- **Microservices Cons:**
 - Requires additional operational complexity.
 - Can lead to higher infrastructure management overhead.
- **Serverless Pros:**
 - Automatic scaling.
 - Reduced operational burden.
 - Cost-efficient for variable workloads.
- **Serverless Cons:**

- Limited control over infrastructure.
- Cold start latency for some functions.
- **Elasticity and Pay-for-Service:**
- **Elasticity:** Elasticity is crucial for handling variable workloads efficiently. Both microservices and serverless can be designed for elasticity, but serverless platforms inherently offer auto-scaling capabilities.
- **Pay-for-Service:** In serverless, you pay per execution or resource usage, which can be cost-effective for sporadic workloads. In microservices, you need to manage resources like containers or virtual machines, potentially leading to higher fixed costs.