Dipesh Patel
04/17/2023
Assignment name: CS 470 Final Reflection
https://youtu.be/LhGptu1a1Zc

I have been able to develop new skills in AWS cloud computing and Docker containerization as well has strenghten my skills as a software developer. I learned about S3 storage, Lambda functions, DynomoDB and API gateways just to name a few. These skills make me more marketable as more companies are switching to cloud computing.

This course has helped me strengthen my analytical and problem-solving skills, attention to detail, creativity, flexibility, and teamwork. Software development requires a solid foundation of technical skills, including programming languages, software design, and database management. Software developers are often tasked with finding solutions to complex problems. The ability to break down problems into smaller, more manageable components is critical to success in this field. Attention to detail is important as writing code requires an eye for detail and the ability to catch even the smallest mistakes. While software development is largely based on technical skills and processes, creativity is essential for developing new and innovative solutions. Technology is constantly evolving, and software developers need to be able to adapt to new tools, languages, and frameworks. Most software development projects are collaborative efforts. Being able to work well in a team environment and contribute to group goals is essential for success.

The knowledge I have gained will help further my career as a full stack developer. It also opens up a potential career path to the DevOps field. Being a full stack developer also opens up to sololy front-end or back-end specific development.

Microservices and serverless architectures can both provide efficiencies of management and scale for web applications. With microservices, each service can be individually scaled based on its specific needs. This allows for more efficient use of resources and can reduce costs. Serverless functions automatically scale based on demand, so you don't have to worry about provisioning and managing resources yourself. Both microservices and serverless architectures can help with error handling by providing built-in fault tolerance and isolation. This means that if one service or function fails, it won't necessarily bring down the entire application.

Serverless architectures can be more predictable in terms of cost, as you only pay for what you use. Cloud providers often offer pricing calculators and monitoring tools to help you estimate and track your costs. With microservices, cost prediction can be more complex as you need to consider the cost of running and managing each service. However, by using container orchestration tools like Kubernetes, you can more easily manage and scale your services.

There are some factors to consider when deciding between microservices and serverless architectures. Microservices can be more complex to manage and require more upfront investment in infrastructure and tools, but offer more control and flexibility. Serverless architectures can be easier to manage and offer more efficient use of resources, but can be less customizable and may have higher cold start times. Both microservices and serverless architectures require a certain level of expertise to implement and manage effectively.

Elasticity is a key feature of both microservices and serverless architectures. By automatically scaling resources up and down based on demand, these architectures can help you save money and ensure optimal performance. Pay-for-service is also a key aspect of serverless architectures. By only paying for what you use, you can avoid over-provisioning and save money. However, it's important to carefully monitor your usage and consider the costs of each function or service to ensure you're getting the best value.

Ultimately, the choice between microservices and serverless architectures will depend on your specific needs and use case. It's important to carefully evaluate your options and consider factors such as scalability, cost predictability, complexity, and expertise before making a decision.