Annaliese Pintar

4/20/2025

CS 470 Final Reflection

https://youtu.be/I7OEtWcgeRw

Reflection

During this course I have gained many new skills. I have learned how to create a docker image along with how to use several AWS tools like Lambda functions, S3 buckets, and API Gateway. Also, I learned about IAM roles and policies which allow me to ensure that my applications are secure and have the correct permissions. My strengths as a software developer are that I am a great problem solver and I learn from the problems that I solve. When something isn't working as expected, I tackle the problem head on, find a solution and learn from the original problem to prevent similar issues later on. Now that I have completed the courses at Southern New Hampshire University, I am prepared for frontend, backend, full-stack, and embedded system development as a level 1 software developer or junior developer.

Planning for growth with cloud services requires strategic decisions between microservices and serverless architectures. Microservices allow for independent scaling of individual application components using container orchestration tools to manage scaling and recovery. Serverless functions automatically handle variable workloads without having to worry about provisioning. For error handling with microservices, circuit breakers and isolated failure domains are implemented while for a serverless platform, there are built-in redundancy and retry mechanisms.

Cost predictability varies significantly. The cost for containers is based on provisioned resources, providing more consistent costs for steady workloads. Serverless costs are based on the amount of resources consumed meaning there is zero costs during idle periods but a chance of higher costs at sustained high volumes. Containers are generally more cost-predictable for established services with consistent traffic patterns.

There are pros and cons to both container and serverless architectures when planning for expansion. Containers are better suited for long-running processes and allow for greater control over infrastructure. They allow for more predictable performance characteristics and lower per-execution costs at high volumes. They also have a mature ecosystem of management tools The cons of containers is that they require ongoing management and monitoring and have greater operational complexity. Also, underutilized resources still incur costs so containers still require capacity planning and provisioning. Serverless allows for minimal operational management and automatically scales with traffic with zero resource cost when resources are idle. Deployment is significantly simplified and there is built-in high availability. However, with serverless you might have some latendy issues, there is limited execution duration, you have less control over infrastructure, and higher costs at sustained high volume.

Elasticity is important for handling variable workloads without overprovisioning. It ensures resources scale up automatically to meet demand without maintaining the capacity constantly. Pay-for-service models align costs with actual business value. It avoids upfront infrastructure investment and allows for you to only pay for the resources that are used. When planning for growth, if you expect consistent usage patterns, then container architecture may be better. However, if there is a chance of variable usage, the elasticity and pay-for-service model of serverless architecture may be more beneficial.