Monisha Gutierrez
December 2024, CS 470 Final Reflection
https://youtu.be/wIh_t-iZmAw

Experiences and Strengths

How This Course Helped Me Reach My Professional Goals

Over the duration of CS 470, I learned how to design and develop a full-stack web application hosted in the cloud. This involved setting up front-end interfaces, back-end APIs, and leveraging cloud services for deployment, storage, and scalability. By working with technologies like container orchestration (e.g., Docker, Kubernetes) and serverless platforms (e.g., AWS Lambda, Azure Functions), I gained practical, hands-on experience that will make me more competitive in the job market.

Skills Gained, Developed, and Mastered

1. **Cloud Infrastructure Setup**
    - Configuring virtual machines, storage, and networking in a cloud environment (AWS, Azure, or GCP).
    - Understanding how load balancers, auto-scaling groups, and high-availability setups ensure reliability.
2. **API Development and Testing**
    - Designing and implementing RESTful APIs with frameworks like Express.js, Flask, or Django.
    - Using tools like Postman and automated testing scripts to ensure code quality and reliability.
3. **Deployment Pipelines (CI/CD)**
    - Setting up automated deployment pipelines using GitHub Actions, Jenkins, or CircleCI.
    - Learning how to implement automated testing, build, and deployment strategies to ensure rapid iteration.
4. **Security and Best Practices**
    - Applying authentication and authorization strategies (e.g., OAuth, JWT).
    - Encrypting data in transit (SSL/TLS) and rest, ensuring a secure cloud infrastructure.
5. **Collaboration and Project Management**
    - Utilizing Git for version control and collaboration with team members.
    - Employing agile methodologies to plan, track, and iterate on project features.

Strengths as a Software Developer

- **Adaptability**: Ability to pick up new technologies quickly, from containerization to serverless, ensuring I can meet shifting project requirements.

- **Problem-Solving**: Comfortable troubleshooting complex problems in a distributed environment, whether it is debugging APIs or addressing performance bottlenecks.
- **Communication**: Skilled at explaining technical concepts clearly in both verbal and written form, enabling effective collaboration with cross-functional teams and stakeholders.
- **Attention to Detail**: Consistently testing code at each iteration, minimizing defects, and maintaining documentation to ensure maintainability and scalability.

Potential Roles and Responsibilities I Am Prepared For

- **Full-Stack Engineer**: Capable of building and maintaining both client-facing interfaces and back-end services.
- **DevOps Engineer**: Experienced in setting up CI/CD pipelines, container orchestration, and monitoring.
- **Cloud Solutions Architect (Junior Level)**: Proficient in designing cloud infrastructure, picking services that optimize cost, scale, and performance.
- **Software Developer in Test (SDET)**: Capable of designing robust testing frameworks and automations to ensure software quality.

Planning for Growth

Microservices or Serverless for Future Efficiency

As my web application grows, I may transition it from a monolithic design to a microservices or serverless architecture to improve manageability, resilience, and scalability.

1. **Handling Scale and Error Handling**
    - **Microservices**: Each service handles a specific function, allowing independent scaling based on demand. If one service fails, it doesn't necessarily crash the entire system, improving fault tolerance.
    - **Serverless**: Functions scale automatically in response to incoming events. Built-in error handling and retry logic in many serverless platforms can simplify error recovery.
2. **Predicting Cost**
    - **Microservices in Containers**: Costs can be more predictable if the workload is steady. You pay for the infrastructure you keep running (EC2 instances, container hosts). You can roughly estimate monthly costs based on usage patterns and capacity.
    - **Serverless**: You pay per execution and compute time. This can be extremely cost-effective if traffic is intermittent or spiky, but cost predictions can be tricky if you suddenly see large spikes in usage or if your code is not optimized.
3. **Which Is More Cost Predictable—Containers or Serverless?**

    - Containers (e.g., running on ECS, Kubernetes, or Azure Container Instances) provide a clearer monthly baseline cost if you have steady usage because you typically pay for reserved capacity (unless you use an on-demand approach).
    - Serverless can be unpredictable if usage patterns vary significantly. However, for low to medium traffic with occasional spikes, serverless can often be cheaper overall because you only pay for what you use.
    - In general, if you have sustained, high-traffic services, container-based solutions might provide better cost predictability. If your traffic is variable, serverless might be more efficient.

Pros and Cons of Scaling Approaches

Approach	Pros	Cons
Microservices	- Independent development and deployment - Easier horizontal scaling - Resilience to partial failures	- Increased operational complexity - Potentially more expensive if each service is underutilized
Serverless	- Automatic scaling - Pay-per-use billing - Reduced operational overhead	- Limited execution time, memory, and supported runtimes - Vendor lock-in concerns - Can become costly with unpredictable spikes

Elasticity and Pay-for-Service in Future Growth

- **Elasticity**: By leveraging elasticity in the cloud, the application can scale up or down automatically based on demand. This is crucial for maintaining performance during peak loads while minimizing costs during off-peak hours.
- **Pay-for-Service**: The cloud's pay-for-service model rewards efficient architecture. Designing stateless, granular services with on-demand resources ensures you only pay for what you use. This can influence architectural decisions like splitting monolithic services into serverless functions or containers to reduce idle time and optimize usage.

Conclusion

CS 470 provided me with a solid foundation in full-stack development and cloud-based deployment. I have learned essential skills like containerization, serverless architecture, API development, and testing, which align with my professional goals as a developer. My experience in designing, implementing, and documenting a real-world cloud-based application has enhanced my adaptability, problem-solving, and communication skills—key strengths that will prepare me for roles such as Full-Stack Engineer or DevOps Engineer.

As I plan for future growth, I will keep a focus on:

- **Choosing the right architecture** (microservices vs. serverless) to balance cost, scalability, and manageability.
- **Predicting cost and usage** by monitoring traffic, optimizing services, and employing best practices.
- **Leveraging elasticity and pay-for-service** to build a solution that meets variable demands while preserving budget efficiency.

By continuously refining these skills and planning for growth, I am well-positioned to build high-performing, scalable, and cost-effective applications in the evolving cloud landscape.