Jayden Chavarria

02/25/2025

CS 470 Final Reflection

https://www.youtube.com/watch?v=vRCbtmNu16U

Experiences and Strengths

Throughout CS470, I've gained valuable skills that have significantly enhanced my marketability as a software developer (if this is what I get into). Working on a full-stack web application in the cloud has provided me with hands-on experience in modern development practices that could be highly sought after in today's job market.

Skills Developed

- **Cloud Architecture Design**: I've learned to architect applications specifically for cloud environments, understanding the unique considerations compared to traditional on-premises deployments.
- **API Development and Implementation**: Creating RESTful APIs that effectively connect frontend and backend components has become one of my core competencies.
- **Serverless Application Development**: I've mastered implementing serverless functions using AWS Lambda, which has given me insight into event-driven architecture.
- **Infrastructure as Code**: Using CloudFormation and similar tools has taught me how to automate infrastructure deployment, ensuring consistency and reducing manual configuration errors.
- **Continuous Integration/Deployment**: Setting up CI/CD pipelines for cloud deployments has given me practical experience with DevOps practices.
- **Cost Optimization Strategies**: I've developed the ability to analyze and optimize cloud resource usage for maximum cost efficiency.

Developer Strengths

My experience in CS470 has reinforced several of my strengths as a developer:

- **Full-Stack Proficiency**: I can work comfortably across the entire application stack, from frontend UI/UX to backend services and database management.
- **Problem-Solving Approach**: I excel at breaking down complex problems into manageable components and implementing efficient solutions.
- **Adaptability to New Technologies**: The rapid pace of cloud service evolution has honed my ability to quickly learn and apply new technologies.
- **Security-Focused Development**: I prioritize implementing secure practices throughout the development lifecycle, particularly important in cloud environments.
- **Detail-Oriented Documentation**: I create comprehensive documentation that facilitates collaboration and maintenance, as demonstrated in my course presentation.

Prepared Roles

Based on the skills acquired in CS470, I am prepared to assume roles such as:

- **Cloud Solutions Architect**: Designing scalable, resilient cloud infrastructures
- **Full-Stack Developer**: Building complete web applications with modern cloud integration
- **DevOps Engineer**: Managing CI/CD pipelines and cloud infrastructure automation
- **API Developer**: Creating and maintaining robust API services
- **Cloud Application Developer**: Specializing in cloud-native application development

Planning for Growth

Microservices and Serverless Efficiencies

As my web application grows, implementing microservices and serverless architectures would provide significant advantages:

Scale and Error Handling

- **Independent Scaling**: Converting monolithic components into microservices would allow scaling individual functions based on specific demand patterns rather than scaling the entire application.
- **Isolated Failures**: With microservices, errors would be contained within specific services, preventing system-wide failures.
- **Auto-scaling Lambda Functions**: AWS Lambda could automatically handle traffic spikes for specific functions without manual intervention.
- **Concurrency Management**: Setting appropriate concurrency limits for Lambda functions would prevent resource exhaustion during unexpected traffic surges.

Cost Prediction and Management

- **Fine-grained Monitoring**: Breaking the application into microservices or serverless functions would provide detailed usage metrics for each component.
- **Pay-per-Execution Model**: With serverless, costs would directly correlate with actual usage, making expenditure more predictable based on traffic patterns.
- **Resource Utilization Analysis**: AWS Cost Explorer and CloudWatch would help identify cost hotspots in the application.

Cost Predictability: Containers vs. Serverless

- **Serverless Predictability**: Serverless offers superior cost predictability for variable or unpredictable workloads since you pay exactly for what you use.
- **Container Predictability**: Containers provide better cost predictability for consistent, high-volume workloads where reserved instances can be efficiently utilized.

Pros and Cons for Expansion Plans

Advantages of Microservices/Serverless Expansion

- **Independent Development Cycles**: Teams can develop, test, and deploy individual services without affecting the entire application.
- **Technology Flexibility**: Different services can use different technologies best suited to their specific requirements.
- **Granular Resource Allocation**: Computing resources can be allocated precisely where needed, reducing waste.

- **Improved Fault Isolation**: Issues in one service don't necessarily affect others, improving overall system resilience.
- **Simplified Maintenance**: Smaller, focused codebases are easier to understand and maintain.

Challenges of Microservices/Serverless Expansion

- **Increased Complexity**: Managing numerous services introduces operational complexity in monitoring, deployment, and troubleshooting.
- **Data Consistency Issues**: Distributed data across services creates challenges in maintaining consistency.
- **Cold Start Latency**: Serverless functions may experience cold start delays, affecting user experience for infrequently used functions.
- **Communication Overhead**: Service-to-service communication adds network latency and potential points of failure.
- **Debugging Complexity**: Tracing issues across distributed services requires sophisticated monitoring and logging solutions.

Elasticity and Pay-for-Service in Growth Planning

Elasticity and pay-for-service are fundamental to cost-effective growth planning:

- **Demand-Based Resource Allocation**: Elastic services automatically adjust capacity to meet demand, eliminating the need to predict exact resource requirements months in advance.
- **Market Testing Enablement**: New features can be launched with minimal upfront investment since infrastructure costs scale with adoption.
- **Geographic Expansion**: Serverless and microservices architectures facilitate deploying to multiple regions to improve global performance without duplicating entire infrastructure.

- **Experimentation Freedom**: The low-risk financial model encourages innovation since failed experiments have limited cost impact.
- **Seasonal Business Alignment**: Pay-for-service perfectly accommodates seasonal business patterns, automatically scaling down during low periods.
- **Capital Expenditure Reduction**: Shifting from capital expenditure to operational expenditure improves cash flow and reduces financial risk.