Prince Oppong Yeboah

12/22/2024

CS 470 Final Reflection

https://youtu.be/t_5D6jWAMmU

Experiences and Strengths

How this course will help in reaching professional goals:
This AWS cloud computing course has provided foundational knowledge in designing and deploying scalable, reliable, and cost-effective cloud solutions to hosting applications. This aligns with my goal of becoming a software engineer who would be working in the cloud environment often.

Skills learned, developed, or mastered:

- Building and deploying applications in AWS using services like S3, Lambda, API Gateway and DynamoDB.
- Implementing security best practices with IAM.

Strengths as a software developer:

- Strong problem-solving skills and adaptability.
- Proficiency in cloud-native architecture and microservices design.
- A focus on writing clean, scalable, and secure code.

Roles prepared to assume:

- Cloud Solutions Architect
- DevOps Engineer
- Backend Developer specializing in serverless applications
- Site Reliability Engineer (SRE)

Planning for Growth

Synthesizing knowledge about cloud services:
AWS provides scalable, secure, and flexible services, making it easier to build applications that can adapt to changing workloads (elasticity) while optimizing costs (pay-for-use).

Using microservices or serverless for efficiencies:

- **Scale and error handling:** AWS Lambda automatically scales based on demand, while services like CloudWatch and X-Ray help monitor and handle errors efficiently.
- **Cost prediction:** AWS Cost Explorer and Budgets enable tracking and predicting costs based on usage.
- **Cost predictability (containers vs. serverless):** Serverless (e.g. Lambda) is typically more cost-predictable for sporadic workloads due to pay-per-invocation pricing, while containers (e.g., ECS/EKS) may suit steady workloads.

Pros and cons for expansion:

- **Pros:**
    - **Serverless:** Reduced operational overhead, automatic scaling.
    - **Containers:** Greater control over environment and configurations.
- **Cons:**
    - **Serverless:** Limited execution time and runtime dependencies.
    - **Containers:** More complex management and scaling needs.

Elasticity and pay-for-service in growth decisions:
Elasticity ensures that applications can scale dynamically with demand, avoiding under- or over-provisioning. The pay-for-service model minimizes upfront costs, allowing for efficient use of resources as the business grows. This flexibility is key to future-proofing applications in a dynamic market.