Module 8 Final Reflection

Samuel A. Meade
Southern New Hampshire University
IDS-403-H7578 Technology and Society 23EW1
M. Shah Alam
12/17/2023

Experiences and Strengths: Explain how this course will help you in reaching your professional goals.

In this course, I have learned basic principles of cloud computing, and how to refactor local applications into containers to make them more portable and efficient, and ultimately port them into a cloud offering like AWS. My strengths as a developer are my ability to code and solution outside of the box. This creativity has helped me excel not only in optimizing code, but also thinking about software design with foresight to scalability and sustainability. In my current role as a software engineer, this is exercised frequently. Working at an enterprise scale company, moving applications off on-prem is a consistent ask with modernizing our tech-debt and making tech spend "count". Leveraging Docker, Kubernetes, and AWS, this becomes a lot more realistic, now having a means of provisioning some of this technology into more manageable components and deploying them to the cloud.

Planning for Growth: Synthesize the knowledge you have gathered about cloud services.

Leveraging microservices/serverless architecture can significantly enhance the efficiency of managing and scaling web applications. Microservices enable modular development and deployment, allowing teams to independently scale components based on demand. Serverless computing, on the other hand, automatically scales resources in response to varying workloads, reducing the need for manual intervention. Both approaches offer improved error handling by isolating components, minimizing the impact of failures on the overall system. Cost prediction in a microservices environment involves assessing the usage of individual services, whereas serverless computing offers a more granular pay-as-you-go model, potentially enhancing cost predictability. Serverless architectures often provide more predictable costs than containers, as they abstract infrastructure management, allowing organizations to focus on code execution rather than infrastructure provisioning and maintenance.

When considering expansion, containerized code's portability makes it a flexible option to freely move the stack to the most performant and cost-effective offering available. If this means retaining an app on on-prem, or shuffling between Google Cloud and AWS, the team has the freedom to follow the most competitive infrastructure. Additionally, when leveraging serverless through AWS, a lot of the infrastructure is abstracted away from the team, making the app easier to manage. Drawbacks may include the quality of legacy tech, and the time investment to refactor the code for containerization. Some legacy applications defer feature/update deployments, progressively making the repository a giant bug fix, meaning moving off deprecated libraries and dependencies can be a nightmare. The size of the application, and the age of the app are two major factors. Calculating the resources required for the migration and the ROI is imperative.

Elasticity, with its ability to dynamically scale resources based on demand, ensures operational efficiency and cost optimization, allowing businesses to seamlessly accommodate growth without overcommitting resources. Pay-for-service models, such as pay-as-you-go or subscription-based pricing, offer a flexible cost structure aligning expenses with actual usage, reducing upfront financial burdens and facilitating resource optimization. Together, these concepts empower teams to make informed decisions expanding operations, fostering adaptability, and innovation while mitigating risks, encouraging a more agile and responsive approach to planned future growth.

YouTube Link: https://youtu.be/nKVI5_kvzLQ?si=jEmolnQ2Br1mftuK

Resources:

Docker. (2020, February 10). *Overview of Docker Compose*. Docker Documentation. https://docs.docker.com/compose/

Amazon DynamoDB. (n.d.). Amazon Web Services, Inc. https://aws.amazon.com/pm/dynamodb/

AWS Lambda. (n.d.). Amazon Web Services, Inc. https://aws.amazon.com/pm/lambda/

IAM roles. (2020). Amazon.com. https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html