Southern New Hampshire University

Project Two – Presentation Narrative

Joshua Norris

https://www.youtube.com/watch?v=7rk0H\_uSBmE

Containerization and Orchestration

The Serverless Cloud

Cloud-Based Development Principles

Securing Your Cloud Application

Two common models for migrating a full stack application to the cloud are refactoring and lift-and-shift. Refactoring means to alter your existing programs to build them for the cloud. It’s the “back to the drawing board” option. Lift-and-shift instead is all about simplicity, taking an existing program and shifting it to the cloud as-is. Lift-and-sift gets your program up and running in the cloud quickly, but refactoring may be better in the long term to get the most benefit from using the cloud.

There are a few different tools for containerization, such as Kubernetes and Docker Compose, which we used in this project. Docker Compose is used to build, ship and run multi-container applications. Docker Compose uses a docker file that has the commands a user would call on the command line to assemble an image, which is then used to create containers.

Docker Compose uses the docker-compose.yml file to configure multiple containers at once. This is how we were able to connect our web, API, and database containers.

Serverless cloud is a model where the server requirements are handled by a cloud provider. Some benefits are being able to quickly get a project up and running, without having to worry about procuring and setting up servers. Also, going serverless allows for automatic scaling and lower costs due to only paying for what you need/use.

Amazon Simple Storage Service, or S3, is an object storage service. It uses buckets as containers for *objects*, or files. The common theme of scalability continues with S3, with the ability to increase storage as needed, with the only caveat being a possible increase in pricing. Compare this to local storage, where you put a specific amount of storage aside and once it’s full, you’re forced to increase storage.

The benefit of serverless API is not having to provision servers to run on. AWS Lambda is Amazon’s serverless answer to being able to run code without a server. API Gateway is used to forward REST API requests to the applicable Lambda function. In this way, a request from the frontend is sent to the backend where Lambda runs the applicable code, and then returns results to the frontend.

MongoDB and DynamoDB are both options for NoSQL databases. NoSQL databases can accept any type of data, whether it’s structured or unstructured. MongoDB is open-sourced and has flexibility of where it can be used. DynamoDB must be used within AWS. MongoDB has fixed pricing, where DynamoDB has the standard AWS variable pricing model.

Some queries we performed were table scan, finding one question, and getting single record.

Elasticity refers to scaling as needed. This includes expanding when needed, and contracting back down when resources aren’t needed. Elasticity has the benefit of only providing what you need. This leads to the pay-for-use model. Elasticity allows for resources to meet a user’s needs, and the user only pays for what they used. This is compared to traditional models, where a user would pay for what they estimate they will need. If needs grow, the user would have to procure additional resources to meet the needs.

Roles and policies were used to prevent unauthorized access. In AWS, a role is an identity that is authorized to use resources, and the policy defines the permissions of the identity. The LambdaAccessToQuestionAndAnswerTable is a policy created to allow Lambda access to the DynamoDB tables. It allows for querying, scanning, getting items, deleting, updating, and putting.

**References**

Author, A. A. (Year). Title of book. Home URL of the eBook publisher or https://doi.org/10.xxxxxxxxxx