Joseph Iadarola

10/23/2022

Project Two Conference Presentation: Cloud Development

https://youtu.be/hQNOweKeGQk

Hello, my name is Joseph Iadarola and I am here to present on how to migrate a full stack application to an AWS solution.

First off, a little about me. I am a senior CS major at SNHU and I will be graduating at the end of the year. I have previously worked on teams developing desktop and database applications before the Covid pandemic, and now I am completing my degree so that I can eventually become a team leader or some other management position.

Some starting point that needs to be made, I am assuming that everyone has at least a basic understanding of what a full stack application is and that you will be using AWS to migrate your application.

Containerization is the visualization of the overall architecture in a format that allows for a developer to contain an application in its own environment. There are two main models for cloud migration. These are refactor, and rehost or in other words list and shift. Rehost is what I used because it is basically just drag and drop into the AWS environment. The local tools that I used are Docker and its related tools.

One of the related tools is Docker Compose. This uses YAML files and the command line to coordinate multiple isolated Docker containers. Docker Compose also has the benefits of being efficient with hardware resources and the YAML file is easy to configure without needing to modify multiple files in the application.

Serverless Cloud development is the ability to manage development without the need to host your own server or pay for server access when your application is not running. The way that this is done is to only charge for when an application is running and uses server access. This leads to several advantages such as automated scalability and secure server access. AWS provides S3 as a cloud-based storage service. S3 stands for Simple Storage Service and like all cloud-based storage it provides benefits over local storage in the form of data retrieval and multi-machine access to the data on the cloud.

The API I utilized is a REST API with CORS integration to utilize Lambda scripts to authenticate CRUD or create/write, read, update and delete functions to update previously created Question and Answer tables. Serverless API allows for as needed functionality, automatically scales based on usage, and other advantages.

The use of databases is essential in the current world. There are many diverse types of databases and queries to manage those databases. For this presentation I used MongoDB for the initial local machine application and DynamoDB for the AWS migration. MongoDB is an open source, NoSQL database that runs on most platforms and is highly versatile. DynamoDB has a narrower

scope and has limited functionality, but it integrates with other AWS services. Through AWS, I created Lambda functions that allowed for CRUD operations on the Question and Answer tables. The query scripts and the TableScan function allow for easy management of the two tables and AWS allowed for the setup of authentication and access policies that protect the data saved on the tables.

A few notable principles to touch on before going into security are elasticity and pay-for-use. Elasticity is something that many people do not think about, but it is the management of resources by the cloud provider so that no resources are idle. This is done by giving resources only to what is running, and more resources are given temporarily when needed. When those resources are no longer needed, they go somewhere else to be used. This leads to the next principle of pay-for-use. This model is based on the customer only paying for the quote resources that they are using from the cloud provided server. I say quote resources because this can include paying to use the development environment provided by cloud providers even if no application is running.

Security of cloud applications and databases hosted on the cloud is a hot topic and like other software, security is most vulnerable at the human level. In this case that means setting up the cloud application with appropriate access permissions and keeping login credentials secure. Like other cloud providers, AWS has built in protection from outside threats but if you set up your S3 bucket to allow public access or your data tables with the wrong permissions then there will be problems. In AWS a role is a type of IAM identity, IAM means identity and access management, and this role will allow you to set access authorizations and permissions. This is different from policies which define the permission and access authorization that are applied to that role. The custom policies that I created were for the CRUD functionality so that the data could be managed with proper access permissions. When connecting Lambda and API Gateway services, the custom policies that I created were used. The connection between Lambda and the database is also secured through a policy created by a custom authentication scheme to limit access. The S3 bucket that I created for the migration also required settings to be modified but could also have been secured using a user role with the same permissions.

In conclusion, AWS provides several benefits that make the migration from local host to cloud-based applications a worthwhile consideration. The basic cloud advantages include the ones stated earlier like data retrieval and multiple machines having access to the application and database if they have permission. But I will focus on three that I think are the most important benefits. The first is the pay-for-use model that allows you to only pay for what you need. It might not seem like it, but your bosses will never outright say no if you bring them a proposal to save money and this type of model might just be it. Next is scalability; or as I like to think of it, one less thing to worry about. With automatic scalability, the worry about having enough resources to complete a large job is no longer a concern since AWS does the job for you. That is not to say that there are no new concerns, but I will take a reliable work efficiency over loading bars any day. And finally, this is a new term to describe an overarching concept, but AWS uses microservices so that each part of the application is run by a different service. This means that the functions are run by the Lambda service and the database tables are hosted through the

DynamoDB service. I think that separating the components of the application allows for easier development and customization through the options provided by AWS while they are all connected.