**Slide 1**

Welcome to this presentation for course CS-470 Project Two: Cloud Development. My name is Eric Wallace and in this presentation, it is my goal to give you a better understanding of cloud development.

**Slide 2**

During this presentation we will discuss a several topics such as What is Containerization and Orchestration, the tools required for implementing containerization and orchestration, the benefits of using orchestration, concepts, and advantages of serverless, popular AWS serverless services and how to secure cloud applications. Without further ado, lets jump right in.

**Slide 3**

Models used for migrating a full stack application to the cloud. Full stack application deployment can be a complex subject and involves an almost infinite number of scenarios. For instance, if an on-premise application is containerized would there be a need to leave it as a containerized application and use cloud-based services such as Kubernetes or Docker Swarm. Migrating a containerized application to a service-based cloud provider such as AWS or Azure it might be more involved. Usually, it involves moving data and functionality of a container to a cloud-based service which provides the same functionality. Additional setup for connecting services together as well as securing services would be required, we will get into this a little later.

**Slide 4**

Simply put, containerization is the process of consolidating an application and all of its dependencies into a single unit known as a container. Containers can be deployed and run on any platform. Docker is probably the tool most widely used today.

**Slide 5**

Orchestration is a little different, it is automating deployment, scaling, and management of containers. The value of orchestration is it gives developers the ability to share scripts used for orchestration which very easily generates an environment to which an application can be developed. A commonly used tool for this is Docker Compose, Compose allows multiple containers to be created with the use of a single file and usually only requires one command to create, configure and run an application.

**Slide 6**

Serverless computing is a cloud execution model for creating, deploying, and running applications in which business has no need to worry about the server infrastructure. There are many advantages to using serverless computing but here are a few:

Automatic scaling – The provider will automatically scale the resource based on demand. High Availability – There are no down times, replication ensures access to the resource is available 24/7 from anywhere.

Eliminates provisioning and server patching – All server maintenance is handled by the provider, so the customer does not have to.

Reduced cost – It reduces the cost to the customer because there is no need to purchase expense hardware and most resources are billed based on usage.

Faster development of applications – It allows developers to develop applications faster because services in some cases can be duplicated with only minor changes needing to be made.

Creating connections between services in some cases are not required in code because they are facilitated in service configuration. Now, lets dive into a few services we used throughout the course.

**Slide 7**

The first service we are going to cover is S3, S3 is a serverless storage service which is to store files which can be accessed from anywhere. When comparing S3 to local file storage, S3 has many advantages such as:

High Scalability – S3 buckets are high scalable which is also done automatically,

High Availability – Data files are available from anywhere with a cloud connection 24/7,

Security – Leveraging AWS security makes data stored in buckets very secure.

Cost – Without the need to purchase hardware and being billed based on resource usage makes them much cheaper than on-premises storage.

Storing data locally means scaling is done manually with the need to purchase additional hardware. Availability is dependent upon internet service at the local site, security is maintained by onsite IT and hardware for data storage can become quite expensive. With S3 there is none of this needed, it is all handled by the service provider.

**Slide 8**

Lambda is a serverless, event-driven service that runs code for virtually any type of applications when triggered by another service. Lambda logic is the code that runs on the service and can handle requests made by other AWS services such as API Gateway or S3 buckets configured as a website and returns a response in JSON format. NodeJS scripts are produced which are what handles the logic.

Integrating a frontend to a backend with Lambda is straightforward. First you start out by creating a Lambda function, then create an API Gateway with all the proper methods, next configure the API Gateway to utilize a Lambda function and finally, setup headers including CORS on the API Gateway request.

There are many advantages of serverless API, but a few are Scalability, Pay-for-use-billing, high availability, connection to other AWS services, and security

**Slide 9**

Two of the most well-known cloud databases are MongoDB Atlas and DynamoDB. When comparing the two databases we can see there are many differences.

Here we have a table comparing the two databases. One the first row we have the data model used; we can see MongoDB Atlas uses a document-based while DynamoDB uses key-value store, the next row is querying; MongoDB Atlas has single keys, ranges, faceted search, JOINs, graph traversals, and geospatial queries and DynamoDB has primary-key and can have a max of 2 attributes which limits querying flexibility. On the third row we have indexing, MongoDB Atlas has a plethora of ways to index including hash, compound, unique, array, partial, TTL, geospatial, sparse, text, and wildcard and DynamoDB only has hash or hash-range. Finally, on the last row we have availability; MongoDB Atlas can be deployed on AWS, Azure, and GCP while DynamoDB can only be used on AWS.

Using Lambda functions with test scripts is the perfect way to retrieve, insert, update and delete data from a DynamoDB database table.

**Slide 10**

Now we are going to talk about cloud-based development principles and the two we are going to discuss is Elasticity and Pay-for-use-model.

First off, we have Elasticity; Elasticity is the ability to automatically scale a service up or down depending on demand.

Lastly, there is pay-for-use-model which is a billing model cloud service providers use in which customers are only charges for the usage of resources and results in a lower cost to the customer. This model is the most widely used model in the cloud computing industry.

**Slide 11**

The next three slides we will discuss three techniques used to secure a cloud app.

The first technique we are going to discuss is Access.

Probably one of the most important rules to follow when developing any application and not just a cloud app is to follow industry standards and guidelines.

Cloud service providers increase security to services and data because they use improved and current security standards to protect data and services.

Using authentication methods such as two-factor authentication when accessing resources can increase the security of an application.

**Slide 12**

The next technique we are going to cover is policies.

AWS offers roles and policies as two methods for securing data and services.

Roles are a set of permissions which dictates what actions a user or service is allowed to perform.

Policies are used to control how services or services be accessed.

In this course, we created a policy that defined what actions our Lambda functions were allowed to perform on our DynamoDB tables.

**Slide 13**

The final technique we will cover is API security.

In AWS, securing a connection between a Lambda function and an API gateway could be done using roles and policies. API Gateways themselves can be secured using API keys.

Connections between Lambda functions and a DynamoDB most commonly would be secured using roles and policies. Restricting how a Lambda function can interact with a DynamoDB table is a very good way of securing this type of connection.

Now, on to S3 buckets, the type of security used will be dictated by the access required. Roles and policies to limit the data an API has access to and what it can do with data is one form of security.

No matter how data will be accessed, using a least privilege approach to securing data is a must.

Another way of securing data in an S3 bucket is to use a control list, which is used to give services access to data.

Finally, is encrypting data at rest and data in transit, this will ensure if data were to fall into the wrong hands nothing could be done with it.

**Slide 14**

In conclusion, Developing or migrating an application to cloud services has many advantages, such as reducing business cost, increase to availability, faster development and simplifies the management of resources.

Utilizing services offered by AWS can make data more secure and less susceptible to outside attacks.

Accessing AWS services from other services cannot be understated, while there is a learning curve, accessing resources of one service from another service can be quickly setup which could eliminate the need of having to it within code.

Thank you for your time and listening to my presentation on Cloud Development

**Slide 15**