Philip Ippolito

SNHU

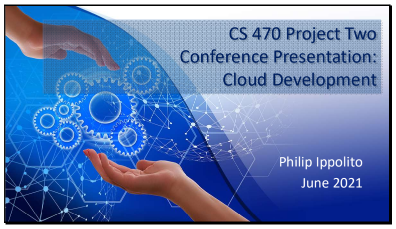
CS-470

Project 2: Presentation

Video Link:

https://www.youtube.com/watch?v=lW5zqgBzp_w&ab_channel=Ippo

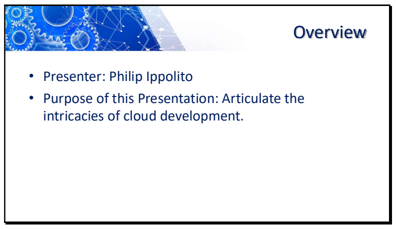
Slide 1



CS 470 Project Two
Conference Presentation:
Cloud Development

Philip Ippolito
June 2021

Slide 2

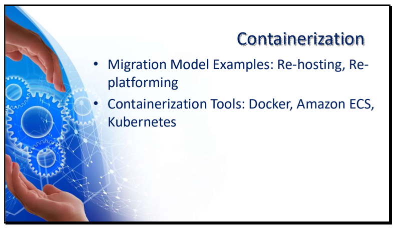


Overview

- Presenter: Philip Ippolito
- Purpose of this Presentation: Articulate the intricacies of cloud development.

Hello, my name is Philip Ippolito. This presentation is to articulate the intricacies of cloud development. It will be presented in a way that will be understandable to both technical and non-technical audiences.

Slide 3

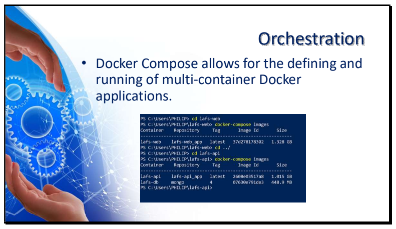


Containerization

- Migration Model Examples: Re-hosting, Re-platforming
- Containerization Tools: Docker, Amazon ECS, Kubernetes

There are a number of different models that can be used when migrating a full-stack application to the cloud. A couple examples of these are re-hosting and re-platforming. Re-hosting simply transfers the application onto the cloud server with little to no changes. Re-platforming is similar, but makes changes to aspects of the application such as the API during and after the transfer process to take advantage of the features of the cloud service.
There are also various different tools available to perform containerization on an application. One common tool that can be used to create and run containers for an application is Docker.

Slide 4



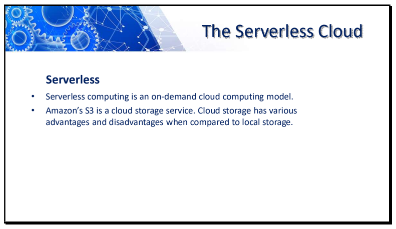
Orchestration

- Docker Compose allows for the defining and running of multi-container Docker applications.

```
PS C:\Users\PHILIP> cd lafs-web
PS C:\Users\PHILIP\lafs-web> docker-compose images
Container    Repository     Tag       Image Id       Size
-----------------------------------------------------------
lafs-web    lafs-web_app   latest   37d278178302   1.328 GB
PS C:\Users\PHILIP\lafs-web> cd ../
PS C:\Users\PHILIP> cd lafs-api
PS C:\Users\PHILIP\lafs-api> docker-compose images
Container    Repository     Tag       Image Id       Size
-----------------------------------------------------------
lafs-api    lafs-api_app   latest   2608e03517a8   1.015 GB
lafs-db     mongo          4        07630e791de3   448.9 MB
PS C:\Users\PHILIP\lafs-api>
```

Docker Compose is a tool that can define and run multi-container Docker applications. The value in using Docker compose is that it provides a way of managing multiple containers that are intended to work together. This is necessary for applications which require multiple containers that serve different purposes.

Slide 5



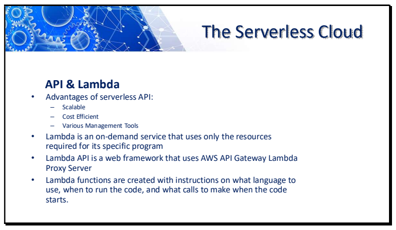
The Serverless Cloud

Serverless

- Serverless computing is an on-demand cloud computing model.
- Amazon's S3 is a cloud storage service. Cloud storage has various advantages and disadvantages when compared to local storage.

Serverless computing is an on-demand cloud computing model. This means that the cloud provider allocates resources to the client as needed based on their usage. This allows clients to only have to pay for the exact amount of resources that they use. Serverless computing also has the benefit of being easily scalable because the cloud provider can simply provide additional machine resources as they are needed.
Amazon's Simple Storage Service, or S3, is a cloud storage service. Compared to local storage which stores data directly on a piece of hardware, cloud storage has the benefit of being much more easily scalable and accessible from multiple locations. The disadvantage of cloud storage over local storage is that accessing the data requires an internet connection and is not available while offline.

Slide 6



The Serverless Cloud

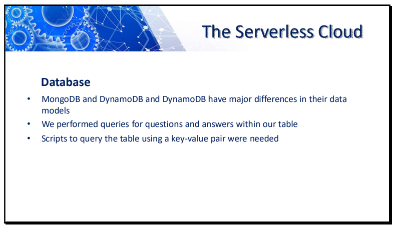
API & Lambda

- Advantages of serverless API:
  - Scalable
  - Cost Efficient
  - Various Management Tools
- Lambda is an on-demand service that uses only the resources required for its specific program
- Lambda API is a web framework that uses AWS API Gateway Lambda Proxy Server
- Lambda functions are created with instructions on what language to use, when to run the code, and what calls to make when the code starts.

There are multiple advantages to using a serverless API. As previously mentioned in regards to serverless computing, serverless APIs are easily scalable and allow the client to be charged only for the API calls they make. Additionally, using serverless APIs through a service such as Amazon's AWS API Gateway grants access a number of management features such as security options and access controls.
Lambda is an on-demand service that uses only the resources required for its specific program. Lambda API is a web framework that uses the AWS API Gateway Lambda Proxy Server. Several

steps need to be taken to integrate the front-end with the backend. Lambda functions must be created and have instructions for what language the code uses, when to run the code, and what calls to make. The AWS API Gateway takes API calls and maps them to a service. AWS API Gateway supports multiple types of APIs such as RESTful and WebSocket for doing this. The API and Lambda functions must be connected through the API methods and Lambda script's response headers. The end result would be, in the example of a RESTful API, that the API Gateway would receive the API request, forward it to the appropriate Lambda function, and then return the result.

Slide 7



The Serverless Cloud

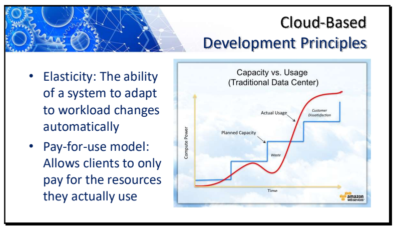
Database

- MongoDB and DynamoDB and DynamoDB have major differences in their data models
- We performed queries for questions and answers within our table
- Scripts to query the table using a key-value pair were needed

MongoDB and DynamoDB have fundamental differences in how they store and access data. MongoDB stores data through JSON while DynamoDB stores data using tables, items, and attributes. Because of this, MongoDB is a more robust database and can perform multiple types of queries such as single keys, ranges, and faceted searches. DynamoDB, on the other hand, is much more limited in the data types and sizes it can support. DynamoDB does, however, have the benefit of being integrated with and managed by AWS, making it easy to use and manage when AWS is being used for an application.
The application we created performed several queries. These included querying for specific questions and answers by ID number. We also included a query to find any one

question in the database. To perform these queries, we created scripts to query the database table using the question and answer ID numbers as attributes.

Slide 8

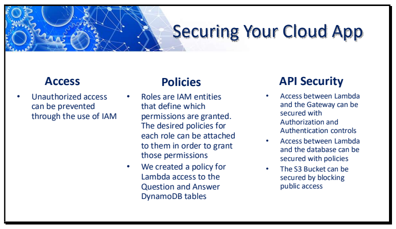

Cloud-Based
Development Principles

- Elasticity: The ability of a system to adapt to workload changes automatically
- Pay-for-use model: Allows clients to only pay for the resources they actually use

Capacity vs. Usage
(Traditional Data Center)
Compute Power
Time
Planned Capacity
Actual Usage
Waste
Customer Dissatisfaction
amazon webservices

Two major principles of Cloud-based development are elasticity and the Pay-for-use model. Elasticity is defined as “the degree to which a system is able to adapt to workload changes by provisioning and de-provisioning resources in an autonomic manner, such that at each point in time the available resources match the current demand as closely as possible.” In short, this means the system’s ability to automatically adapt to workload changes and match the resources being used with the current demand. The Pay-for-use model is a model that allows client’s to only be charged for the amount of resources they actually need and use. This is possible because of the ability of on-demand service to only provide resources as they are needed.

Slide 9



Securing Your Cloud App

Access

- Unauthorized access can be prevented through the use of IAM

Policies

- Roles are IAM entities that define which permissions are granted. The desired policies for each role can be attached to them in order to grant those permissions
- We created a policy for Lambda access to the Question and Answer DynamoDB tables

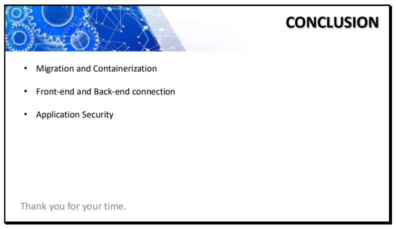
API Security

- Access between Lambda and the Gateway can be secured with Authorization and Authentication controls
- Access between Lambda and the database can be secured with policies
- The S3 Bucket can be secured by blocking public access

Unauthorized access to your Cloud App can be prevented through the use of Identity Access Management, or IAM. IAM encompasses various types of security features, such as roles and policies. Policies control the access permissions of functions. For example, a policy can be created to grand Read and Write access to the database, then assigned to the specific functions that would require that access. Functions that do not need Read and Write access to the database will not have that same access permission. Roles are IAM entities which define which permissions are granted. They are not associated with individual users and are assumed temporarily as needed. Policies are attached to roles to define which permissions those roles can grant access to. For our application, we created a policy to grant Lambda access to the Question and Answer tables in our DynamoDB database.

Access to various parts of the application can be secured in different ways. Access between the Lambda and the Gateway can be secured with Authorization and Authentication controls. Access between the Lambda and the database can be secured through the use of policies. Access to the S3 bucket can be secured by blocking public access.

Slide 10



CONCLUSION

- Migration and Containerization
- Front-end and Back-end connection
- Application Security

Thank you for your time.

In conclusion, I will reiterate three major steps in launching a Cloud based application. The first major step is to containerize and migrate the application onto a cloud service such as AWS. The next major step is to connect the front-end and back-end of the application through the use of Lambda and the API Gateway. Finally, all aspects of the application should be secured through the use of IAM. That is all I have for this presentation, thank you for your time.