



<https://youtu.be/rsgUB0oIX08>

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

The goal of this course was to take a full stack application development and migrate so that it runs in the cloud. I learned about the mean stack through the coding of a full-stack JavaScript example application project and connecting a backend API to an Angular front-end application employing the MEAN stack to perform CRUD operations on a MongoDB database. Through the application of cloud-based development principles and best practices, I took my software stack experience from Full Stack Development I and utilized the frameworks to build the cloud architecture upon which the software stack application will run. The Full Stack Development II has given the exposure of cloud development frameworks for leveraging cloud infrastructure by migrating your web application to a cloud-based environment. In addition, I was able to demonstrate my career-readiness by articulating highly technical content in various formats. This experience has brought out strengths as a system engineering developer, it helped identify the different type of roles within software that can be utilized when needed in my assumed new job.

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

To conclude, cloud computing can help businesses reduce the upfront investments in hardware. Instead of having to invest heavily in data centers and servers before you know how you're going to use them, you pay for how much you consume.

Applications can be built on low-level infrastructure or can use higher level services that provide abstraction from the management, architecting, and scaling requirements of core infrastructure. Cloud service platform that provides an on-demand delivery of compute power, database storage, and other IT resources via the Internet with pay-as-you-go pricing. It is important to remember and consider all options as many projects will need a system analysis to make the best business decision. A cloud-based application could have either been created in the cloud or have been migrated from an existing infrastructure to take advantage of the benefits of cloud computing.

Overview


- **Migrate full stack applications to the cloud**
 - Amazon Web Services (AWS)
 - Pay-as-you-go
- **Reduce cost**
- **Ensure scalability**
- **Serverless**



Script:

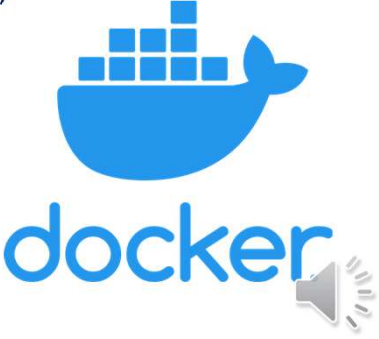
Hello everyone, welcome. My name is Sonny Garcia, and I will be presenting on the process of migrating a full stack application to Amazon's Web Services (AWS). With this migration to the cloud, businesses can reduce cost and ensure scalability by essentially renting computing resources and going serverless.

AWS is a cloud service platform that provides an on-demand delivery of compute power, database storage, and other IT resources via the Internet with pay-as-you-go pricing. With this platform, businesses do not need to make large upfront investments in hardware which in turn do not need to manage this hardware.



Containerization

- **Migration models**
 - 1) Lift and Shift (Rehost)
 - 2) Rearchitect (Refactor)
- **Containers**
 - Docker Compose




Script:

There are typical two migration models to migrate a full stack application to the cloud: Lift and Shift (Rehost) and Rearchitect (Refactor). The lift and shift is basically taking your application as is and moving it into a cloud environment whereas rearchitect means refactoring the application's code which induces more risk.


So, the model we used is the lift and shift as it requires little upfront effort in the migration. With this approach we aren't changing the design or architecture of the application, we are simply moving it into a container. A container consists of an entire runtime environment which include the application, plus all its dependencies, libraries and other binaries needed to run the application. Fundamentally, a container is nothing but a running process, with some added encapsulation features in order to keep it isolated from the host and from other containers. This is a common approach in moving to the cloud because the cloud providers are able to run containers.

If your application needs multiple components for different purposes, you will need a tool to define the containers involved in your application stack and how these containers work together. Docker Compose is the toolkit provided by Docker to define, build, and run multi-container applications.



Orchestration

- **What's the value of using Docker Compose?**
 - Shares kernel
 - Lightweight
 - Deliver software quickly



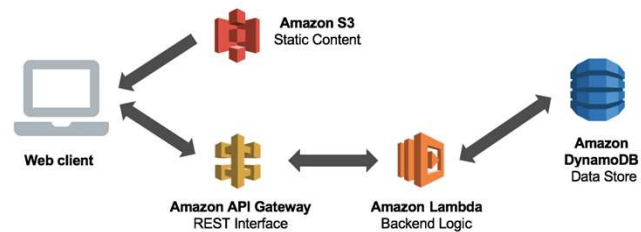
Script:

Docker Compose lets you orchestrate, or coordinate, multiple containers at the same time. It is the most common gateway to managing micro-services. The importance of this is so that the containers can easily be moved between environments or to the cloud. A container runs natively on Linux and shares the kernel of the host machine with other containers. It runs a discrete process, taking no more memory than any other executable, making it lightweight. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly.

The Serverless Cloud

Serverless

- Advantages
 - Lower cost
 - No need to run and maintain servers
- Simple Storage Service (S3)
 - How does it compare to local storage?
 - Stop guessing capacity
 - Stop spending money running and maintaining data centers



Script:

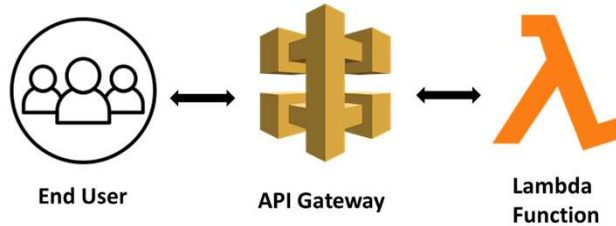
Serverless computing is the most foundational shifts in application development in decades. Prior to this, application logic needed servers to run in. These servers need to be installed, configured, and secured within an operating system. The point of all that overhead and infrastructure was to provide an environment for your application logic to execute within. Serverless Cloud computing provides a simple way to access servers, storage, databases and a broad set of application services over the Internet. Amazon recognized this and have built an infrastructure with the ability to execute your code without having to deploy and manage operating systems, virtual machines, containers, or application servers. This Serverless offloading allows developers to focus the application itself.

An example of this infrastructure is Amazon's Simple Storage Service (S3) which an easy-to-use simple web service interface that can be used to store and retrieve any amount of data at any time. You can choose a region where you want your data to be stored, create a bucket, and you can begin storing data in S3. S3 automatically makes copies of the objects stored on multiple devices across multiple facilities so you don't have to worry about losing valuable data. You can preserve, retrieve, and restore every version of every object in a S3 bucket.

The Serverless Cloud

API & Lambda

- Advantages of serverless API
 - Balance of memory
 - Lower cost
 - Scalability
- Lambda API Logic
 - Request > Authorize > Respond
- Steps needed to integrate the frontend with the backend
 - 1) Create Lambda function
 - 2) Configure API methods
 - 3) Allow permissions
 - 4) Test



Script:

Continuing with the serverless offloading, AWS includes load balancing, security isolation, and managing utilization. Cost should be lower in the case of serverless APIs since cost will be based on runtimes and how often the services are being utilized with requests. So, the more activity, the more the service will cost but this allows the application to scale easily when needed.

Another services offered by AWS is Lambda. Lambda is a serverless, ideal compute platform service. It is an event-driven compute service that runs code for virtually any type of application or backend service without provisioning or managing servers. Lambda encapsulates the code you want to run while dynamically managing the compute fleet that offers a balance of memory, CPU, network, and other resources.

When configuring Lambda functions, Lambda requires a trigger and needs to know when to execute your code. Within the AWS infrastructure, we can combine Lambda with another service to create an API. The API would react to incoming events. In our case, that is the API Gateway.

The API Gateway has one primary purpose: to map incoming API calls to a service or

endpoint. Gateway maps the API to the Lambda function, using Cross-Origin Resource Sharing (CORS) to allow resource sharing across domains. In our case, we set up API methods that have the general CRUD operations for a database. API Gateway will receive a REST API request, forward that request to the correct Lambda function, and manage returning the result. Gateway will also act a security barrier between the outside world and Lambda. The outside world will only be able to access the APIs we deploy, and the API Gateway will be the service to allow permissions to call the Lambda functions.



The Serverless Cloud

Database

MongoDB

- Flexible queries
- Multiple table structure

DynamoDB

- Faster queries
- Single-table structure

- CRUD operations
 - GET
 - POST
 - DELETE



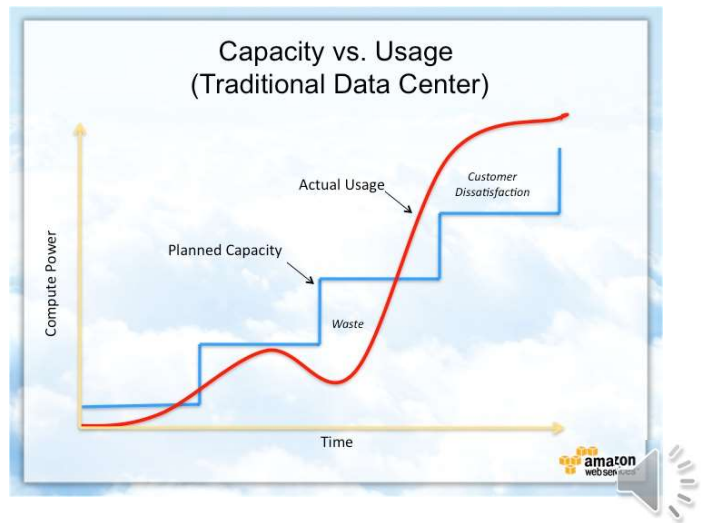
Script:

As part of the migration, we created a database to replace the Mongo database used in full stack application. AWS offers DynamoDB which is a NoSQL database like Mongo with some subtle differences. MongoDB has more flexibility with queries since there are more ways to query data whereas in DynamoDB relies on querying with a primary and secondary index. MongoDB builds have multiple tables that can store data and DynamoDB uses the single-table structure. The concept is that instead of decomposing your data into different entities that are normalized, you store the complete entity in a single table. The table is then partitioned using a partition key. Single-table model allows for faster performance from inside DynamoDB as it alleviates the need for making multiple requests to a database.

We created two single tables for Questions and Answers. We were able to relate the two via ids so we can match the question to the answer. We then used our Lambda functions to build basic CRUD methods in order to query the tables. DynamoDB is accessible via an HTTP API making it a perfect fit for building Serverless applications. DynamoDB also performs authentication and authorization via IAM roles, which we will get into more details in a later slide.

Cloud-Based Development Principles

- Elasticity
- Pay-for-use model



Script:

As mentioned before, AWS is a cloud service platform that provides an on-demand delivery of compute power, database storage, and other IT resources via the Internet with pay-as-you-go pricing. AWS offers businesses a dynamic scaling infrastructure. Customers pay for only what they use. With cloud services, resources can scale with usage and can reduce waste.

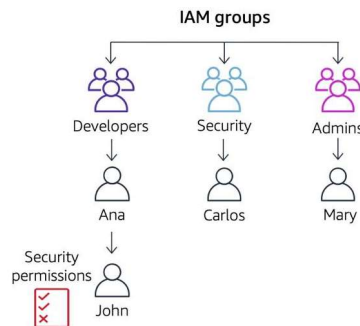
Securing Your Cloud App

Access

- Build security by default
 - Deny-All

Policies

- Specific permissions for different roles or group



API Security

- Create policies to grant certain actions
- Policies grant access to resources

Script:

For security, Amazon's Identity and Access Management or IAM allows you to configure policies which sets permission and control access to AWS resources. The policies control the permissions that the users have to the resources. The permissions include which users can access specific services, methods or actions they can perform, and which resources they can access. Role based policies dictate what specific role is permitted, while resource-based policies define which identities have access to the resource. A role can be assigned to as many individuals as needed.

The policy we created was the Scan, Update, and Delete permissions that granted actions on the Question and Answer DynamoDB tables. We then attached those policies to the IAM role attached to the lambda functions we created. This allowed us to view and make changes to the tables we created in DynamoDB. We set policies to control public access for our S3 bucket which were set to private by default. AWS has many of these built-in security and they just need to be configured by the developer.



CONCLUSION

- **Cloud computing can reduce upfront investments**
- **Scalability**
- **Focus on the application**

Thank you for your time.



Scripts:

To conclude, cloud computing can help businesses reduce the upfront investments in hardware. Instead of having to invest heavily in data centers and servers before you know how you're going to use them, you pay for how much you consume.

Applications can be built on low-level infrastructure or can use higher level services that provide abstraction from the management, architecting, and scaling requirements of core infrastructure.

It is important to remember and consider all options as many projects will need a system analysis to make the best business decision. A cloud-based application could have either been created in the cloud or have been migrated from an existing infrastructure to take advantage of the benefits of cloud computing.

Thank you for your time.



REFERENCES

George, S. (2022, January 22). *Trigger AWS Lambda using API Gateway to access EC2 instance*. Medium. Retrieved October 19, 2022, from <https://awstip.com/trigger-aws-lambda-using-api-gateway-to-access-ec2-instance-44bcc882c577>

Fonolleda, M. (n.d.). *ALBIRA Solutions - Firsts impressions on the use of AWS Serverless*. Retrieved October 19, 2022, from <https://www.albirasolutions.com/en/blog/40-firsts-impressions-on-the-use-of-aws-serverless>