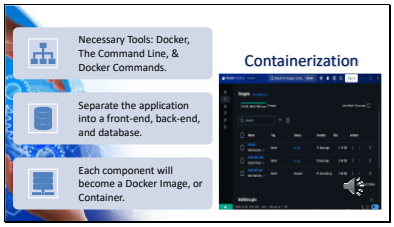
Slide 1

Slide 2

Hello, everyone. My name is Matt Knutson. I am a computer science major at SNHU, and I currently live in Galveston, TX. The purpose of this presentation is to explain how to move a locally hosted MEAN application into a cloud environment. This project will use the AWS Cloud Architecture, and it will take advantage of several of Amazon's Microservices. The Services we will use to build the cloud's infrastructure will replace each element used in the MEAN Stack. The MEAN Stack originally consists of MongoDB as the database, Express.js & Node.js as the back-end, & Angular as the front-end, or client side. Each microservice we use will provide the same functionality in the cloud as it's local counterpart. The AWS S3 service will replace Angular on the front-end, AWS API Gateway & Lambda will replace Express & Node.js on the back-end, and DynamoDB will now be our cloud native database.

Slide 3

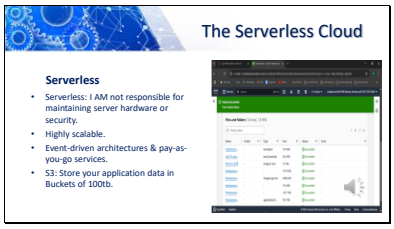


To get started, the front-end, back-end, and database will need to be separated and Containerized in an application called Docker. By using Docker commands in the command prompt, images will be created for each of the necessary components for the application.

Slide 4

When multiple Containers will be used to build an application, Docker allows the developer to consolidate these images into a Docker Compose file. This file is called a YAML file, and with one line of code, all of the containers for the application can be run consecutively. A Network Bridge is responsible for communication between containers within the Docker Composition.

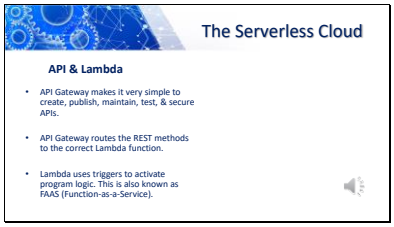
Slide 5



The term serverless refers to a scenario where the developer of an application is not responsible for maintaining and securing the physical hardware that the system runs on. Instead, the cloud provider manages all the server's hardware and keeps it secure. A serverless infrastructure is also highly scalable due to its access to a virtually unlimited amount of storage. The serverless cloud also offers event-driven architecture & pay-as-you-go services, where the customer is only charged for the time and resources they use. This greatly reduces the cost of physically adding

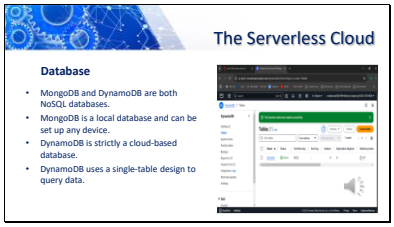
storage to a system and managing local servers. AWS S3 is cloud-based, virtual storage service that is very similar to the local storage on a personal device. While each Bucket will hold 100tb of data, accessing more storage on S3 is as simple as purchasing another bucket. This is a much simpler process than the classic example of adding a new hard drive to a personal laptop or PC.

Slide 6



API Gateway and Lambda make up the serverless back-end of our application. API Gateway not only allows the developer to create, publish, maintain, test, & secure an API, it also acts as a centralized storage for all the APIs in use. API Gateway is also responsible for routing the REST methods (Create, Read, Update, Delete, and Post) to the correct Lambda functions. The Lambda functions are then triggered by changes in data, shifts in the system's state, and actions by users.

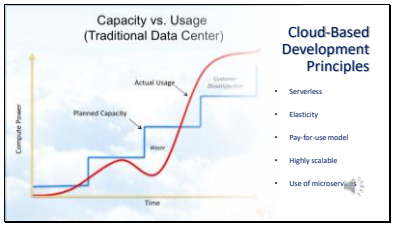
Slide 7



The final piece to the puzzle is the database. While MongoDB & DynamoDB are both NoSQL databases, they have a couple of distinctive characteristics. MongoDB is a local database that is kept on a physical device or server. DynamoDB on the other hand, is strictly a cloud-based database which can be accessed anywhere on the internet with proper authorization. DynamoDB also utilizes a single-table design to query data. Unlike MongoDB, where data is stored as key-value pairs in collections, DynamoDB's single-table design stores

data in entire blocks. These blocks are then partitioned using a partition-key. In DynamoDB, a sort-key can be combined with a partition-key to create a Composite Primary Key. This primary key is then used to quickly recover data collections from the database.

Slide 8

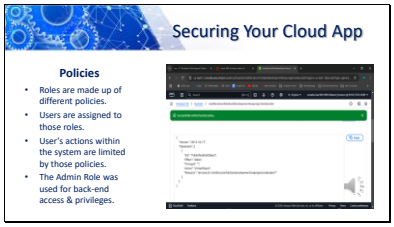


As we have seen from the previous slides, moving to the cloud is relatively simple and yet it comes with many advantages for the client. The serverless model provides elasticity for an application's user traffic, allowing it to easily scale up, or down, based on usage. The pay-as-you-go model keeps the client from investing heavily in resources they won't need and gives them access to unlimited resources should they need them. And using microservices keeps the development process much simpler than building from the ground up.

Slide 9

AWS Identity & Access Management, or IAM, is Amazon's centralized security service. This service is used to create and store roles and is responsible for enforcing security throughout the entire system. IAM checks for access and authorization each time a role-based user moves from one service to another.

Slide 10

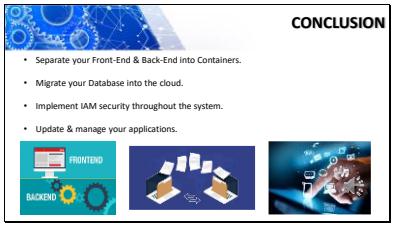


Policies are permissions that are granted to a user. A role is a collection of policies that dictate what that role can do within the systems framework. Rather than assigning policies to every user, users can simply be added to roles where the policies are already set in stone. By using roles, access and authorization are strictly enforced throughout the application. For example, the Admin role in this project can Create, Read, Update, and Delete data from the database, while the average user can only Read information from the website.

Slide 11

To apply security measures to your API Gateway on AWS, all the developer needs to do is initialize an API Key. This can easily be automated when each API is created. Once a key has been put in place, access will be denied to users who don't posses the key. This type of security measure can be added to each layer of services, adding multiple layers of secure connections.

Slide 12



So, there you have it! A brief overview of the tools and services needed to move a MEAN Stacked application from a local server to the AWS Cloud. First, you merely separate your front-end, back-end, and database into Containers. Use Docker to Compose your Containers into a working, cloud-based application, and then migrate your local data into the cloud's database. Make sure you implement IAM security measures throughout your system. Finally, create, update, and manage your applications without the hassle of server and hardware overhead. Thank you very much for your time and it has been a pleasure speaking here today.