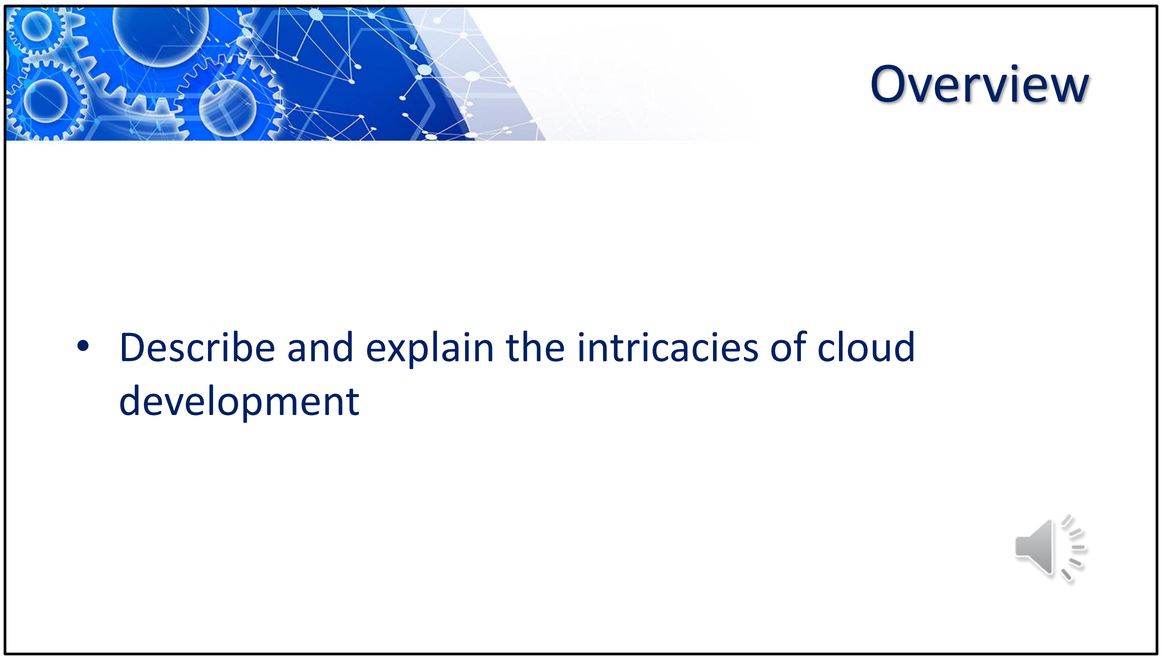
https://youtu.be/QF7VL1PN0IU
Jacob Bennett
8/24 Project Two Conference Presentation: Cloud Development

CS 470 Project Two Conference Presentation: Cloud Development

Jacob Bennett

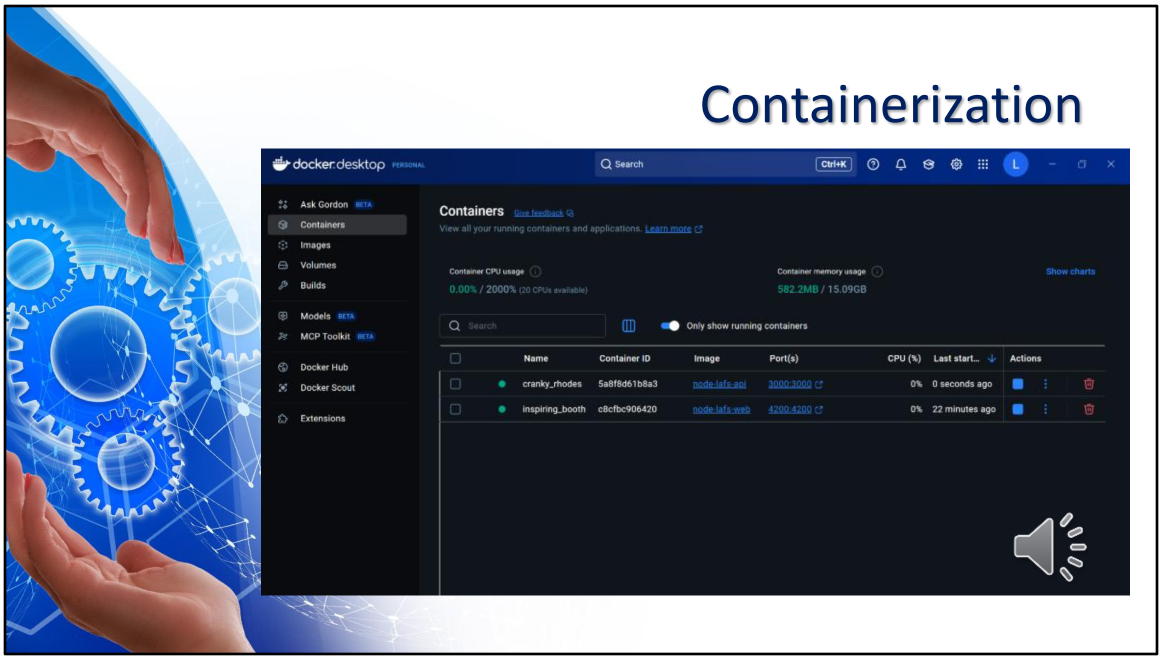
August 2025



Overview

- Describe and explain the intricacies of cloud development

Introduce yourself.
State the purpose of the presentation: to articulate the intricacies of cloud development to both technical and nontechnical audiences.



Containerization

Containers

View all your running containers and applications.

Container CPU usage 0.00% / 2000% (20 CPUs available)

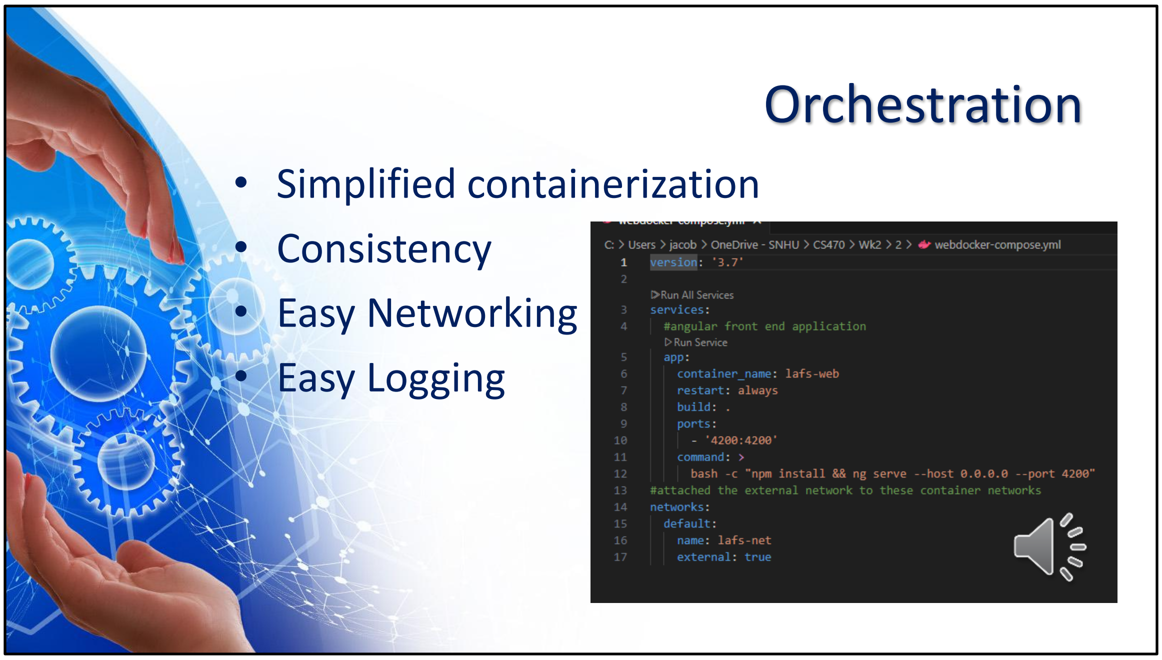
Container memory usage 582.2MB / 15.09GB

Name	Container ID	Image	Port(s)	CPU (%)	Last start...	Actions
cranky_rhodes	5a8f8d61b8a3	node-lafs-api	3000:3000	0%	0 seconds ago	
inspiring_booth	c8cfbc906420	node-lafs-web	4200:4200	0%	22 minutes ago	

Containerization is a concept used to package software so it can be run on multiple platforms without the hassle of different configurations.

Explain the models used to migrate a full stack application to the cloud. – We can use an AWS EC2 instance to host the applications as is! We can also switch to Docker Containers which improves scalability and performance without rewriting the application.

What tools are necessary for containerization? – We be using Docker to containerize our application. Some core tools that are necessary are, Docker itself, which includes Docker engine and the docker desktop GUI. The Docker file which defines how the application is packaged into a container. Docker compose is a tool that allows one to define and run several docker applications using a single configuration file. This is defined in the YAML file. Docker compose allows us to view small applications as services which makes managing a full stack application much easier.



Orchestration

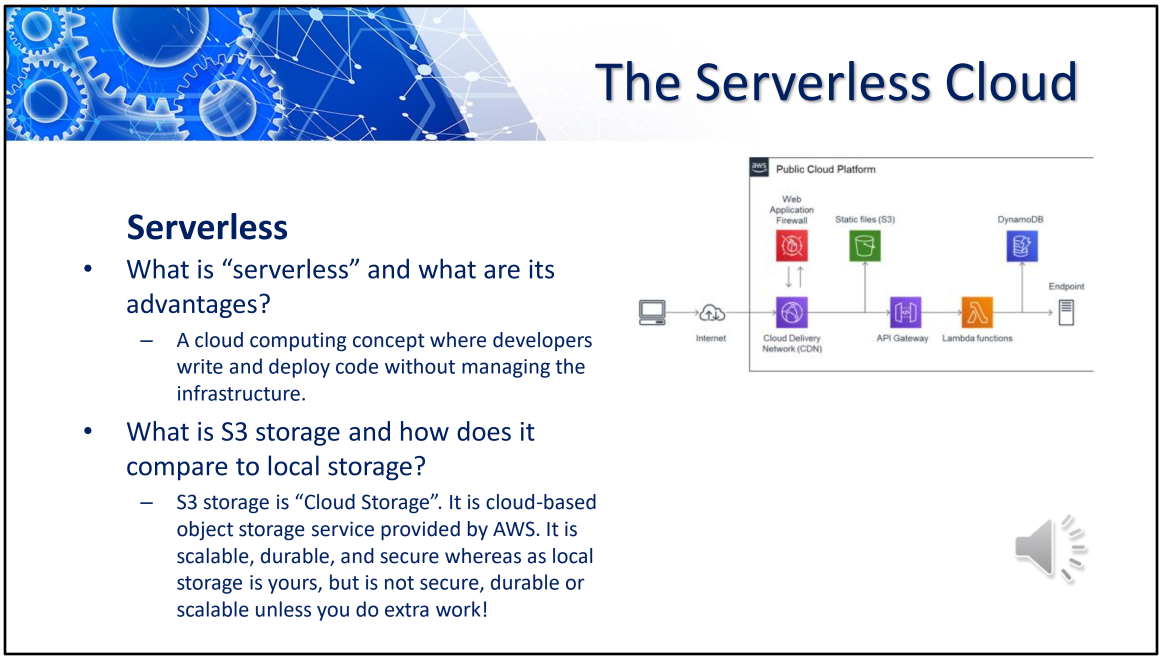
- Simplified containerization
- Consistency
- Easy Networking
- Easy Logging

C: > Users > jacob > OneDrive - SNHU > CS470 > Wk2 > 2 > webdocker-compose.yml

```yaml
version: '3.7'

services:
  #angular front end application
  app:
    container_name: lafs-web
    restart: always
    build: .
    ports:
      - '4200:4200'
    command: >
      bash -c "npm install && ng serve --host 0.0.0.0 --port 4200"
#attached the external network to these container networks
networks:
  default:
    name: lafs-net
    external: true
```

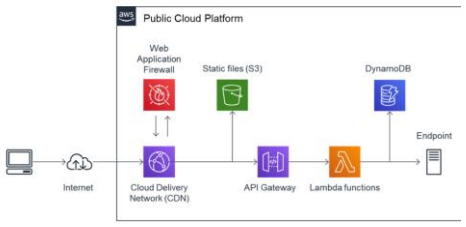
Simplified containerization – define services and networks, configurations scripts all within the yml file
The whole point of Docker is to have a consistent build that will work in many different computers. The idea is that it runs on the same on multiple environments
Automatic network creation per container, can reference other containers with Ips
Build in logging for all services within docker compose



The Serverless Cloud

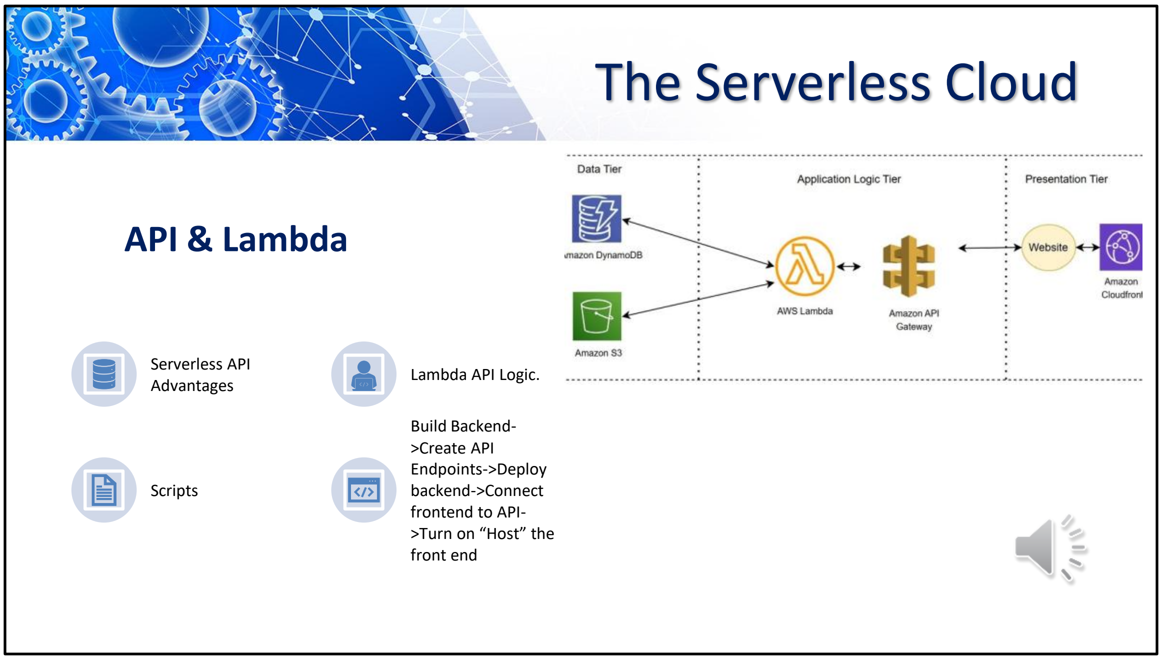
Serverless

- What is "serverless" and what are its advantages?
  - A cloud computing concept where developers write and deploy code without managing the infrastructure.
- What is S3 storage and how does it compare to local storage?
  - S3 storage is "Cloud Storage". It is cloud-based object storage service provided by AWS. It is scalable, durable, and secure whereas as local storage is yours, but is not secure, durable or scalable unless you do extra work!



1. You write code that responds to Events. We use AWS Lambda functions . We also only pay for compute time, not full time costs.

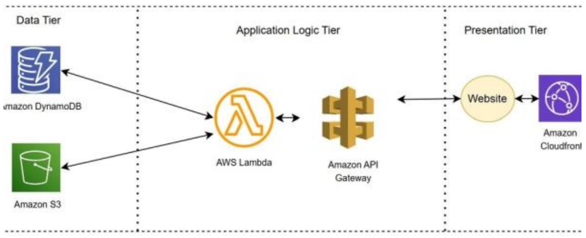
Reference: Kiran, S. (2024, March 5). *Optimizing your cloud ROI with serverless architecture - Capgemini*. Capgemini. https://www.capgemini.com/insights/expert-perspectives/optimizing-your-cloud-roi-with-serverless-architecture/



The Serverless Cloud

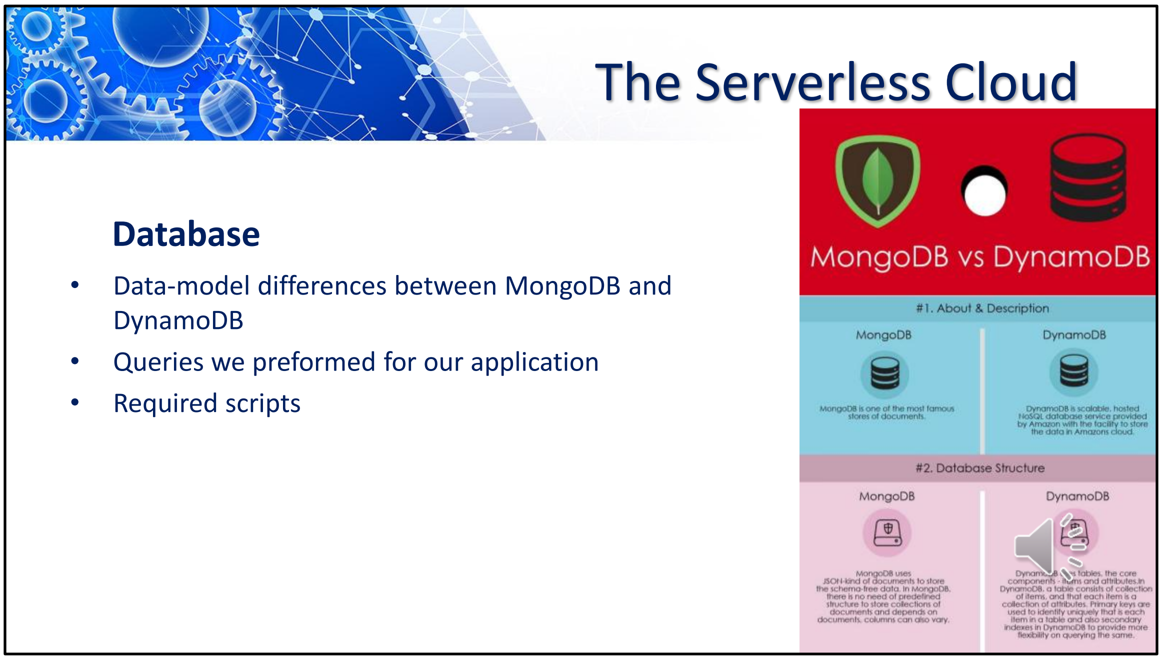
API & Lambda

- Serverless API Advantages
- Lambda API Logic.
- Scripts
- Build Backend->Create API Endpoints->Deploy backend->Connect frontend to API->Turn on “Host” the front end



Describe the advantages of using a serverless API. – Automatically scales yo your needs based on demand. Typically, faster to setup as your functions are simple and modular. Security for dummies! Handles patching, and updates to the infrastructure.
Lambda API Logic. – Is how a lambda function handles requests. A client sends a request, the API gateway receives request and routes to lambda function, lambda function does its thing, API gateway sends response to user
Several scripts required! – Lambda function script, The API gateway script, and the IAM Role Script.

References: Agarwal, R. (2024, December 31). *Deep dive into AWS Serverless Architecture*. DEV Community. https://dev.to/aws-builders/deep-dive-into-aws-serverless-architecture-3pgn



The Serverless Cloud

Database

- Data-model differences between MongoDB and DynamoDB
- Queries we preformed for our application
- Required scripts

MongoDB vs DynamoDB

#1. About & Description

MongoDB	DynamoDB
MongoDB is one of the most famous stores of documents.	DynamoDB is scalable, hosted NoSQL database service provided by Amazon with the facility to store the data in Amazons cloud.

#2. Database Structure

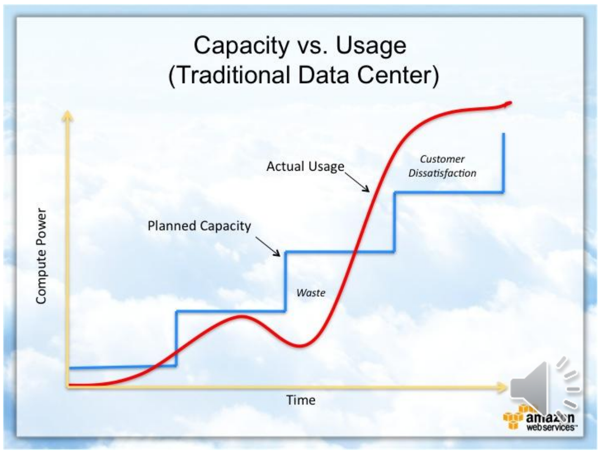
MongoDB	DynamoDB
MongoDB uses JSON-kind of documents to store the schema-free data. In MongoDB, there is no need of predefined structure to store collections of documents and depends on documents, columns can also vary.	DynamoDB uses tables, the core components - items and attributes.In DynamoDB, a table consists of collection of items, and that each item is a collection of attributes. Primary keys are used to identify uniquely that is each item in a table and also secondary indexes in DynamoDB to provide more flexibility on querying the same.

1. MongoDB is document based and uses JSON, whereas DynamoDB is key-value and document based (JSON-like), MongoDB is good for complex or different kinds of data structures, and DynamoDB is great for speed and scalability.
2. Store data, and tables within DynamoDB. We built queries such as answer without filter, question with filter, question without filter. We also have CRUD querie functions such as delete answer, get answer, get questions, insert answer, and insert question
3. We need to create five lambdas: TableScan, GetSingleRecord, UpsertQuestion, UpsertAnswer, and DeleteRecord. Each script was an MJS file, which is pretty much a javascript file.

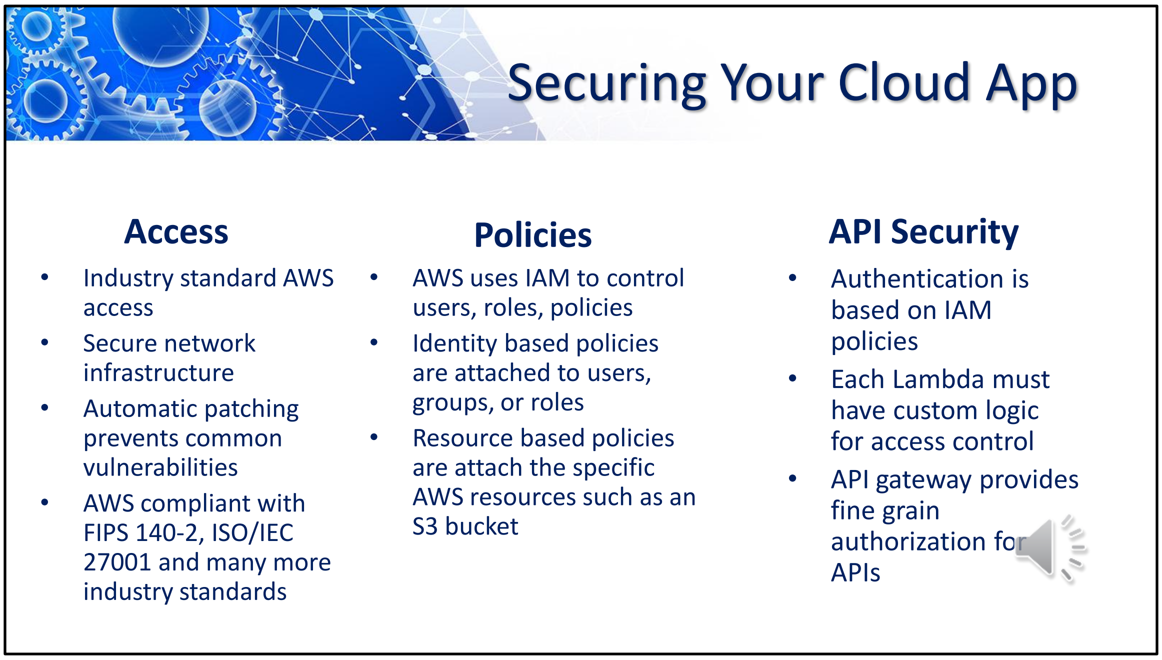
4. References: Pedamkar, P. (2024, May 10). *MongoDB vs DynamoDB*. EDUCBA. https://www.educba.com/mongodb-vs-dynamodb/

Cloud-Based Development Principles

- Elasticity
- Pay-for-use model



1. Elasticity is the ability to automatically scale resources up or down dependent on demand. AWS does this with elastic load balancing, and its auto scaling lambdas.
2. AWS has a pay for use model. This means we pay for what we use. There is on demand pricing for services such as compute, storage. There is no set minimum or maximum which is ideal for unpredictable workloads.
3. This allows a user to optimize costs without the headache of manually provisioning resources.



Securing Your Cloud App

Access

- Industry standard AWS access
- Secure network infrastructure
- Automatic patching prevents common vulnerabilities
- AWS compliant with FIPS 140-2, ISO/IEC 27001 and many more industry standards

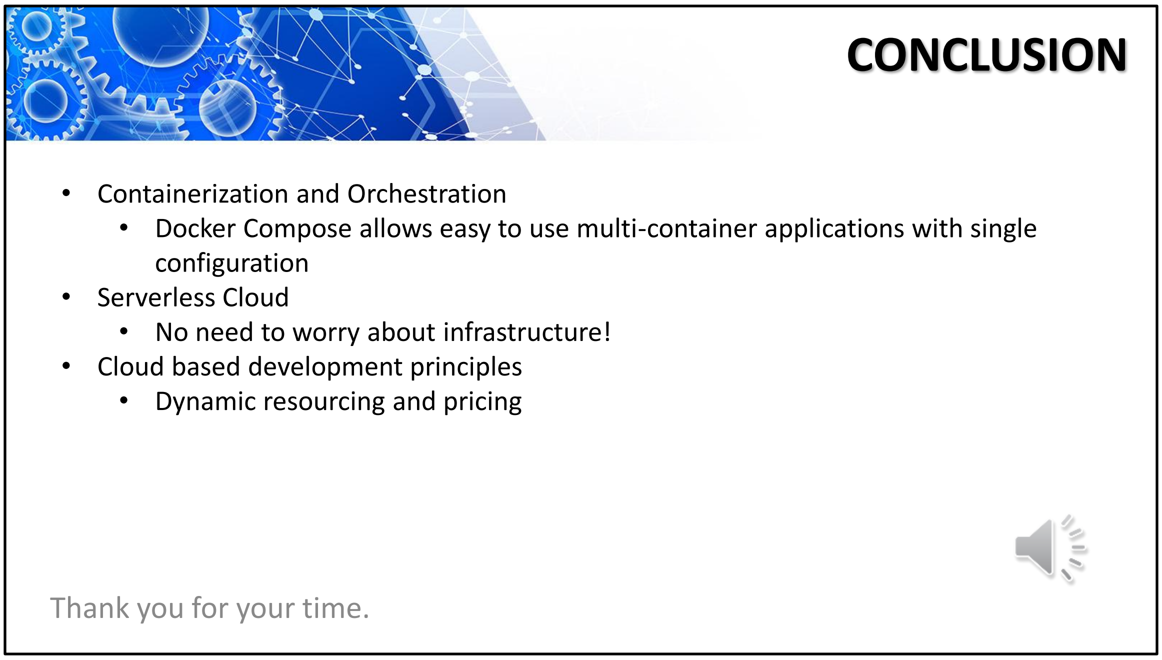
Policies

- AWS uses IAM to control users, roles, policies
- Identity based policies are attached to users, groups, or roles
- Resource based policies are attach the specific AWS resources such as an S3 bucket

API Security

- Authentication is based on IAM policies
- Each Lambda must have custom logic for access control
- API gateway provides fine grain authorization for APIs

References: *Security design principles - Security Overview of Amazon API Gateway*. (n.d.). https://docs.aws.amazon.com/whitepapers/latest/security-overview-amazon-api-gateway/security-design-principles.html



CONCLUSION

- Containerization and Orchestration
  - Docker Compose allows easy to use multi-container applications with single configuration
- Serverless Cloud
  - No need to worry about infrastructure!
- Cloud based development principles
  - Dynamic resourcing and pricing

Thank you for your time.