CS470 Project 2 Conference Presentation: Cloud Development

https://youtu.be/cHjMgalTR_0

Jessica Ayer

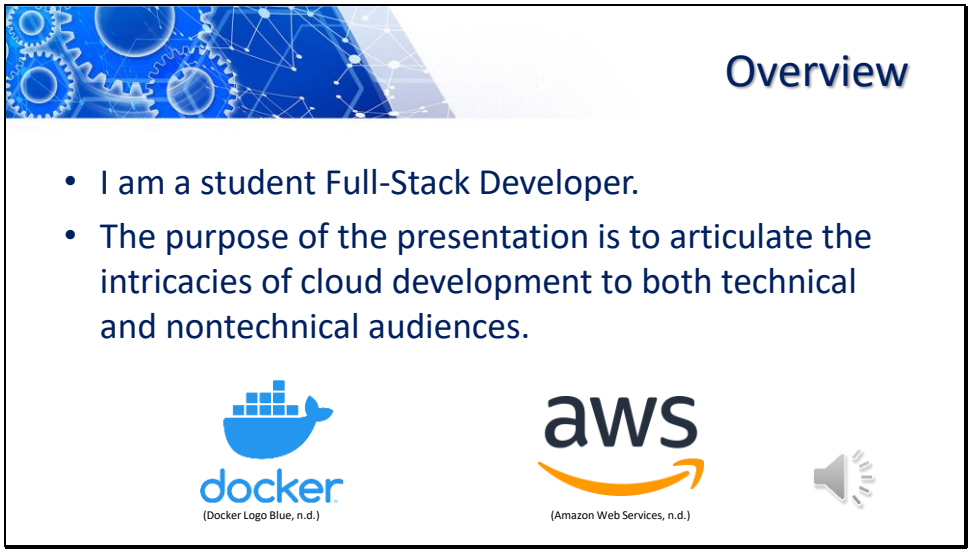
June 17, 2023

Instructor: Kaan Esendemir

Slide 1

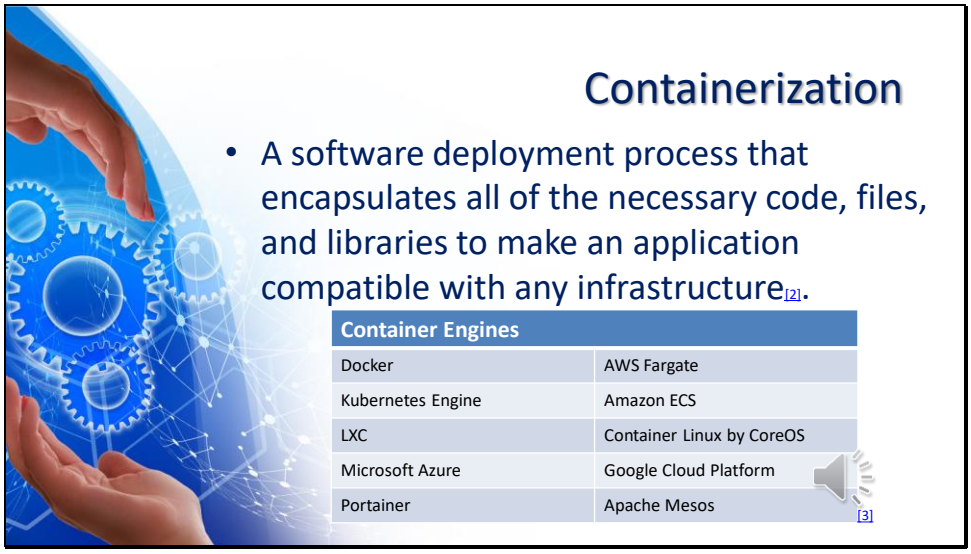
- Hello, my name is Jessica Ayer and I will be your presenter today. Thank you for taking the time to view my presentation on cloud development.

Slide 2



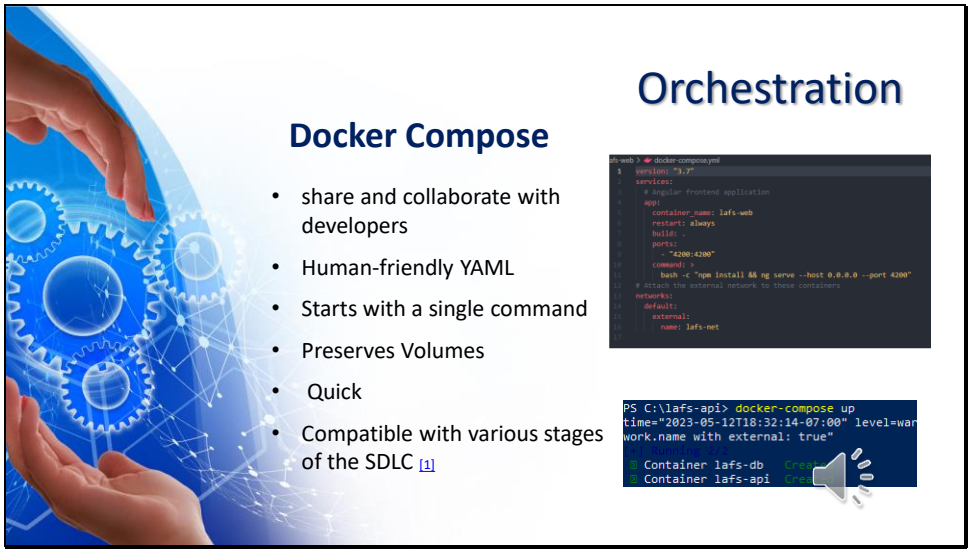
- I am a student Full-Stack developer and I am about to graduate with a bachelors in Computer Science and a minor in Software Engineering from SNHU.
- I recently had the opportunity to experience using Docker to containerize a full-stack web application and then migrate it to an AWS serverless solution.
- This presentation is to explain those tools and articulate the intricacies of cloud development to both technical and nontechnical audiences.
- I will cover some of the benefits of containerization with Docker and tools AWS provides to make the migration process seamless including Amazon S3, Lambda, DynamoDB, and API-Gateway.

Slide 3



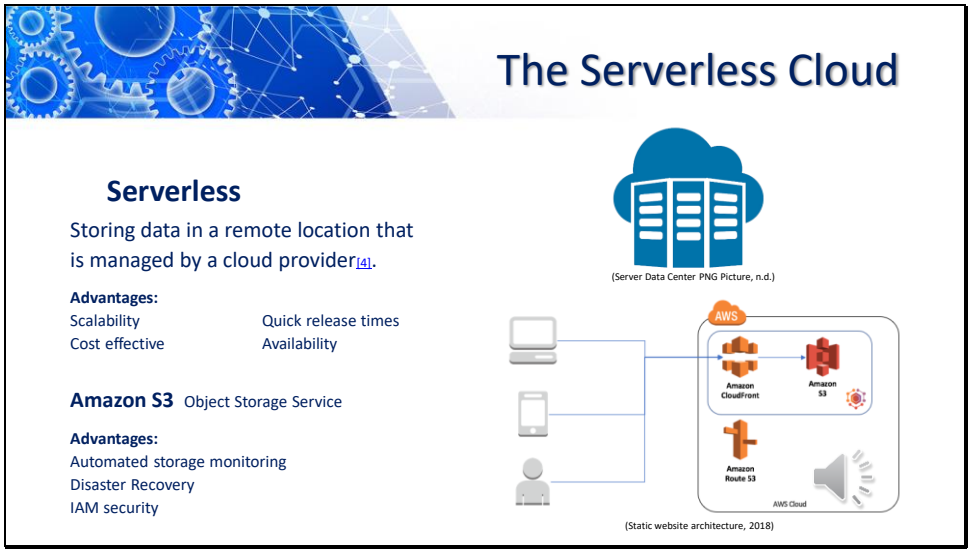
- Let us start with Containerization.
- Containerization is a software deployment process that encapsulates all of the necessary code, files and libraries needed to make an application compatible with any infrastructure.
- To containerize an application, you need a container engine. I used Docker, however there are several other tools available. I've listed ten of the more popular containerization engines. If you would like to explore further the reference slide which will be made available to all in attendance, has a link that provides a brief explanation of each.
- Each of these tools allow a developer to move their application to a operating system level virtualization. They can break down complex applications into modules or microservices that are isolated but able to communicate making the application easy to manage with container orchestration frameworks (*Docker Compose Overview,* 2023)

Slide 4



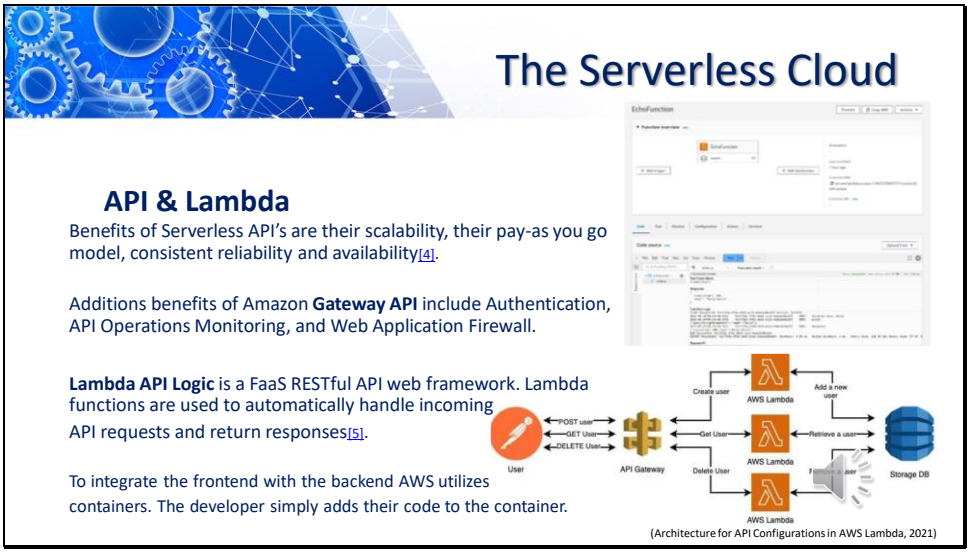
- Docker Compose is an orchestration framework.
- It is a valuable tool to use as it is a user-friendly way to share and collaborate with other developers by making an applications environment reproducible on other machines (*Docker Compose Overview,* 2023).
- It uses the human-friendly, readable language YAML to configure an applications service in an isolated environment which can then be started with a single command (*Docker Compose Overview,* 2023).
- It preserve volumes by copying containers from previous runs so that data does not get lost (*Docker Compose Overview,* 2023).
- Only the changed containers are recreated making Docker Compose quick to use (*Docker Compose Overview,* 2023).
- Docker Compose is compatible with various stages of the software development lifecycle from staging to testing for deployment making it versatile and convenient for many use cases (*Docker Compose Overview,* 2023).

Slide 5



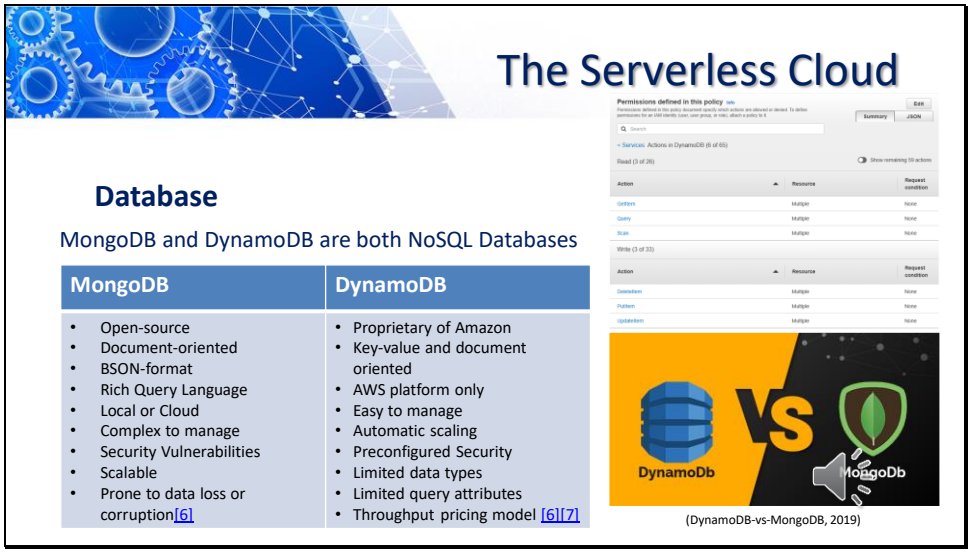
- Serverless means storing data in a remote location that is managed by a cloud provider. Unlike traditional cloud hosting which is permanent, serverless computing is an event driven setup without permanent infrastructure (Doerrfeld, 2017). This means that cloud providers manage the infrastructure allowing the developer to focus on their code which can result in quicker release times.
- Serverless storage, has the advantage of scalability. Local disc storage has limited space making scaling time consuming and costly. Cloud storage has unlimited storage space in which the user only pays for what the use regardless of upscaling or downscaling their storage needs making it cost effective to the user. Another advantage is availability, the serverless cloud can be accessed from any location or device with access to the internet (*Amazon S3*, 2023).
- Simple Storage Service, or S3 is a serverless cloud platform offered by Amazon Web Services. Unlike local storage, it is an object storage service meaning it can store large volumes of unstructured data from multiple devices and make it all accessible from a virtual storage repository on any single device. It separates data into units bundled with metadata and gives them unique identifiers. S3 stores these objects in what Amazon refers to as “buckets” *(Amazon S3*, 2023).
- It provides additional advantages such as disaster recovery. In the unlikely event that an S3 server fails, backup servers are available. S3 also has built in security features such as Identity and Access Management which we’ll go into more depth on shortly.

Slide 6



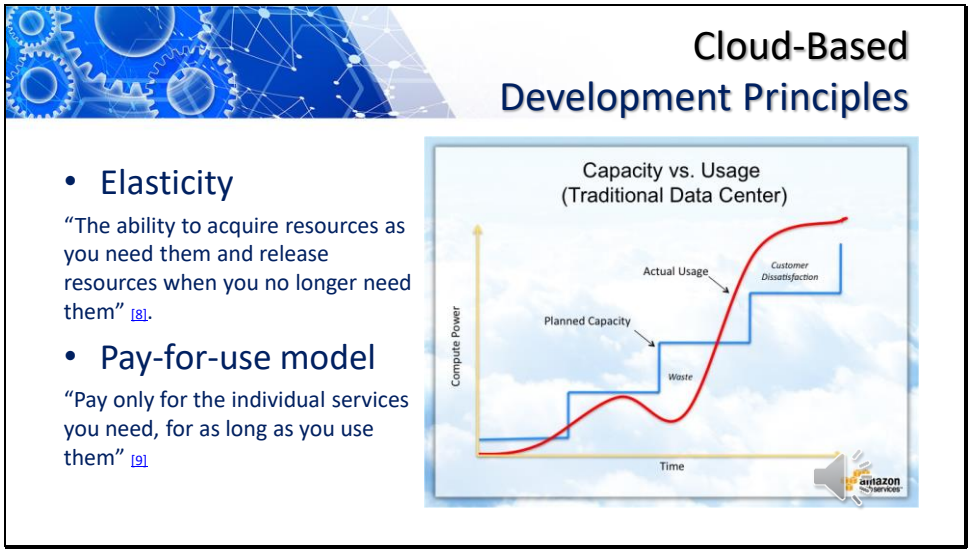
- You're probably starting to see a theme tied to the word "serverless", Scalability, Cost efficiency and Availability. Using a serverless API means no capacity issues. Serverless servers are auto-created on a per-need basis and scale on demand as needed (Doerrfeld, 2017)
- You won't be wasting money paying for idle time. You only pay by the millisecond or possibly even nothing if you use a service like Amazon Web Services Lambda and your application fits within their free tier (Doerrfeld, 2017).
- Consistent reliability and availability mean that you can trust that you can access your applications data whenever you are connected to the internet.
- Using Amazon Gateway API adds even more benefits which you can find in the online documentation but some of my favorites include Authentication, API Operations Monitoring and a Web Application Firewall.
- Amazon's Lambda API Logic is a Function as a Service RESTful API web framework. All a developer needs to do is insert their code into a Lambda function and the Lambda function will automatically trigger the appropriate actions for each API request and return the responses. When not in use functions shut down automatically. The user only pays for the number of times an application is executed and the time needed to complete the execution. Lambda has default permissions and then allows the user to configure roles from there. Amazon Web services handles server management including operating system patches. All the user has to do is manage the functions themselves (NAKIVO, 2022).
- To integrate the frontend with the backend, Amazon Web Services utilizes containers to place all of the dependences for an application into a single location. A developer simply adds their code to the container and lets Amazon Web services handle the rest.

Slide 7



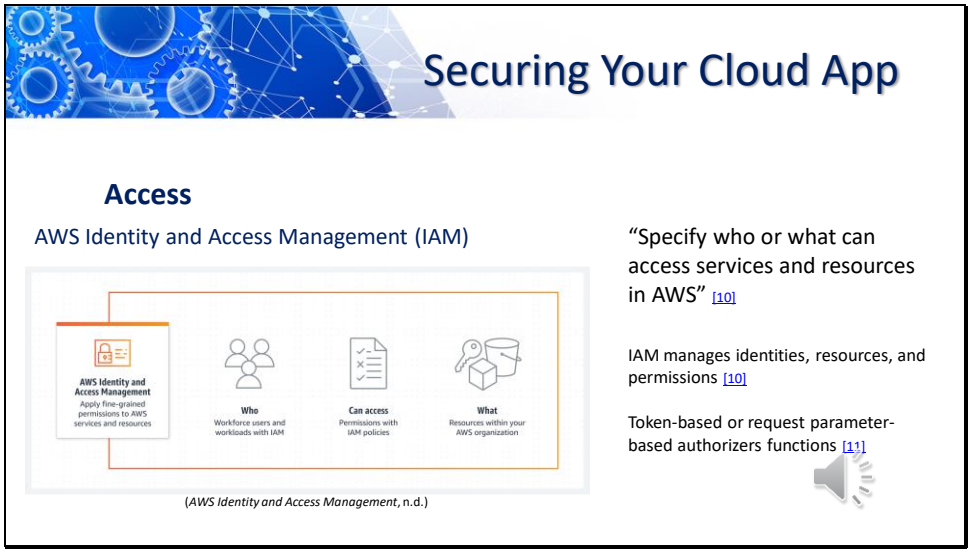
- The application that I migrated to AWS originally used MongoDB for the database. The database was migrated to AWS's DynamoDB. Both databases are NoSQL databases meaning they are schemeless.
- MongoDB is an open-source document-oriented database that utilizes BSON-format and uses rich query language. It can be deployed both locally and on any cloud provider but is complex to set up, configure, and maintain. Authentication is disabled by default lending MongoDB to security vulnerabilities. MongoDB is easily scalable but, due to MongoDB's complex data transactions it is prone to data loss or corruption (Livingston, 2022).
- DynamoDB is a proprietary of Amazon and supports both key-value and document-oriented data but is limited to deployment on AWS platforms. Due to being incorporated into AWS, configuration management is done by AWS including updates and scaling making setup fast and easy. It also comes with preconfigured, high-quality security (Livingston 2022). Primary keys are limited to three data types and key-value queries are limited to 2 attributes (*Comparing DynamoDB and MongoDB, n.d.)*. DynamoDB's pricing is based on a throughput model and therefore is variable (Livingston, 2022).
- For my application, I created a policy that allowed read and write queries. In the Read queries I used GetItem, Query, and Scan, in the Write permissions, I used DeleteItem, PutItem, and UpdateItem. This permission policy was then attached to Lambda functions to allow the application to interact with the DynamoDB database.
- I created two DynamoDB tables, Questions and Answers. I then created Lambda function to scan the entire tables and display all data, get a single record from either table, find a single question, add a question, add an answer, and delete a record from either table.

Slide 8



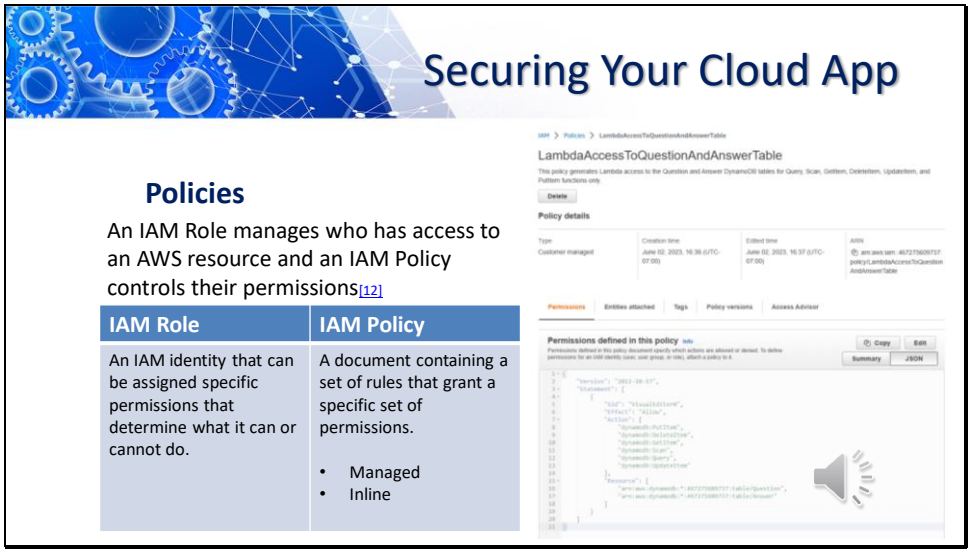
- Let's circle back around to the cloud-based development principles of elasticity and the pay-for-use model.
- Elasticity is "The ability to acquire resources as you need them and release resources when you no longer need them". Both upscaling and downscaling should be done automatically in the cloud. (*AWS Well –Architected Framework,* n.d).
- We've touched on the Pay-for-use model a few times already but to reiterate, this means you "Pay only for the individual services you need, for as long as you use them" There are no long term contracts. You simply pay for the services you consume. When you stop using them, there are no additional costs (*AWS Pricing*, n.d.)

Slide 9



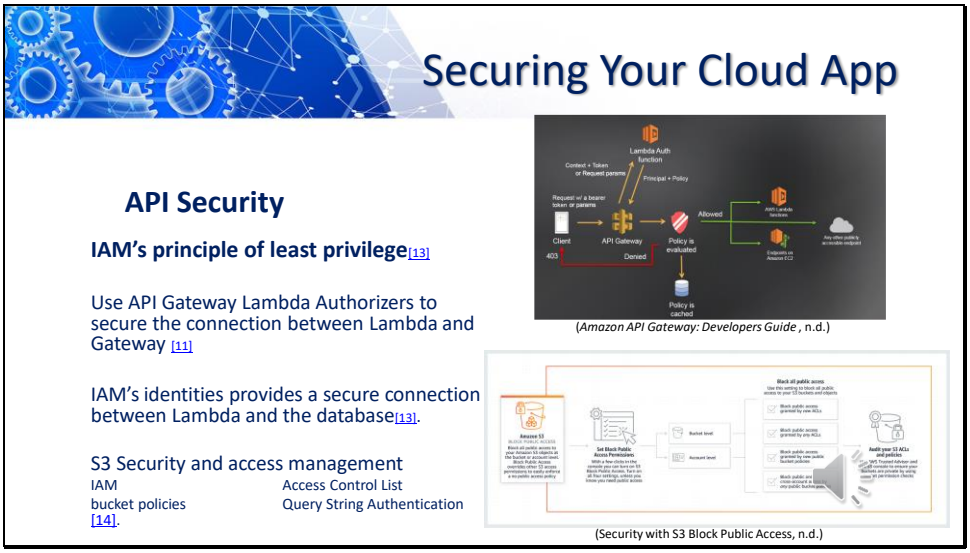
- "Specify who or what can access services and resources in AWS" (*AWS Identity and Access Management,* n.d.)
- AWS Identity and Access Management Services can be used to manage identities, resources, and permissions. It includes sign-up and sign-in functionality saving the developer time and allowing them to focus on their application (*AWS Identity*, n.d.).
- Lambda authorizer can be used to control access through a token-based or a request parameter-based authorizer function (*Amazon API Gateway: Developers Guide*, n.d.).

Slide 10



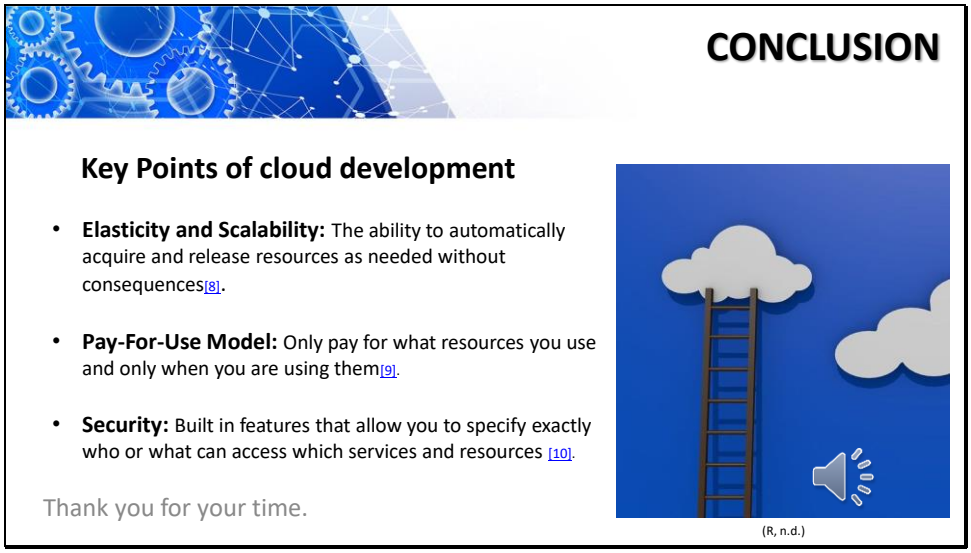
- An IAM role manages who has access to an AWS resources and an IAM policy controls their permissions.
- More specifically, An IAM role is an identity that can be assigned specific permissions that determine what it can or cannot do.
- An IAM Policy is the document that contains the set of rules that grant the specific set of permissions.
    - There are two types of IAM policies. Managed policies and Inline policies.
        - Managed policies can be reused and attached to multiple entitles.
        - Inline policies apply directly to IAM entities and cannot be reused or attached to other entities.
- For my application I created the customer managed policy LambdaAccessToQuestionsAndAnswersTable which allowed DynamoDB access to the roles I assigned to the Lambda Functions. As you can see in the action portion of the JSON policy, the defined permissions were the ability to PutItem, DeleteItem, GetItem, Scan, Query, and UpdatItem within the DynamoDB Questions table and Answer Table show in the Resource portion of the code.

Slide 11



- IAM's principle of least privilege. This principle means granting only the permissions required to complete a task which can be done through IAM policies (Potter, 2020).
- To secure the connection between Lambda and Gateway you can use API Gateway Lambda Authorizers. This is a feature that uses an additional Lambda function to control API access.
    - It allows for custom authorizations schemes that utilize token authentication or request parameters to determine a user's identity. The Lambda authorizer inputs the user's identity and returns the IAM policy as output (*Amazon API Gateway: Developers Guide,* n.d.) .
- IAM's principle of least privilege provides a secure connection between Lambda and the database through use of identities (Potter, 2020).
- S3 Security and access management defaults by only giving the creator access to the S3 resources. Access must be granted to other users through IAM, Access Control Lists, bucket policies, or Query String Authentication.
    - IAM can create users and manage their access
    - Access Control Lists make individual objects accessible to authorized users
    - Bucket policies configure permissions for all objects withing a single bucket
    - Query String Authentication grants time-limited access to others with temporary URLs (*Amazon S3 Security and Access Management,* n.d.)

Slide 12



- I'd like to take this time to highlight what I feel are three of the main takeaways about cloud development.
- Elasticity and scalability which is the ability to automatically acquire and release resources as needed without consequences. (*AWS Well –Architected Framework,* n.d).
- Pay-For-Use Model meaning only pay for what resources you use and only when you are using them(*AWS Pricing*, n.d.)
- Take advantage of security features like IAM that allow you to specify exactly who or what can access which services and resources (*AWS Identity and Access Management,* n.d.)
- Thank you for your time.

Slide 13

REFERENCES

Amazon API Gateway: Developers Guide (n.d.) Amazon Web Services. https://docs.aws.amazon.com/apigateway/latest/developerguide/welcome.html [11]

Amazon S3 (2023) AWS. https://aws.amazon.com/pm/serv-s3/?trk=fc204d25-9fcb-4256-8efb-9ff782ed1674&sc_channel=ps&s_kwcid=AL!4422!10!71605922716739!71606461502681&ef_id=88d4cd1cba851399173ba97a0405e525:G:s

Amazon S3 Security and Access Management (n.d.) AWS. https://aws.amazon.com/s3/security/

Amazon Web Services (n.d.) [Digital Art]. Retrieved June 16, 2023. Amazon Press Center. https://press.aboutamazon.com/logos

Architecture for API Configuration in AWS Lambda (December 14, 2021). [Digital Art]. Retrieved June 17, 2023. Graphable. https://www.graphable.ai/blog/aws-serverless-api/

AWS Identity and Access Management (n.d.) AWS. https://aws.amazon.com/iam/ [10]

AWS Pricing (n.d.) AWS. https://aws.amazon.com/pricing/?nc2=h_ql_pr_ln&aws-products-pricing.sort-by=item.additionalFields.productNameLowercase&aws-products-pricing.sort-order=asc&awsf.Free%20Tier%20Type=*all&awsf.tech-category=*all [9]

AWS Well –Architected Framework (n.d.). AWS. https://wa.aws.amazon.com/wat.concept.elasticity.en.html [8]

Comparing DynamoDB and MongoDB (n.d.). MongoDB. https://www.mongodb.com/compare/mongodb-dynamodb[7]

Docker Compose Overview (May 10, 2023). Docker Documentation. https://docs.docker.com/compose/ [1]

Docker Logo Blue (n.d.) [Digital Art]. Retrieved June 16, 2023. Docker. https://www.docker.com/company/newsroom/media-resources/

Slide 14

REFERENCES

Doerrfeld, Bill (February 16, 2017). *The Benefits of a Serverless API Backend*. Nordic APIS. https://nordicapis.com/the-benefits-of-a-serverless-api-backend/ [4]

DynamoDB-vs-MongoDB (November 27, 2019). [Digital Art]. Retrieved June 17, 2023). WHIZLABS. https://www.whizlabs.com/blog/dynamodb-vs-mongodb/
NAKIVO (June 2, 2022). *AWS Lambda vs. Amazon EC2: Which One Should You Choose?*. https://www.nakivo.com/blog/aws-lambda-vs-amazon-ec2-which-one-to-choose/ [5]

Livingston, Zephin (April 21, 2022). *MongoDB vs DynamoDB: Which NoSQL database is right for you?* Educative. https://www.educative.io/blog/mongo-db-vs-dynamo-db-nosql-databases [6]

Potter, Ben (December 2, 2020). *Techniques for writing least privilege IAM policies*. AWS Security Blog. https://aws.amazon.com/blogs/security/techniques-for-writing-least-privilege-iam-policies/ [13]

R (n.d.) [Digital Art]. Retrieved June 17, 2023. ADHD Interrupted. http://www.adhdinterrupted.com/our-products.html

Security with S3 Block Public Access (n.d.) [Digital Art]. Retrieved June 17, 2023). AWS. https://aws.amazon.com/s3/security/

Server Data Center PNG Picture (n.d.). [Digital Art]. Retrieved June 16, 2023). PNG ALL. https://www.pngall.com/data-center-png/

Static Website Architecture (November 7, 2018). [Digital Art]. Retrieved June 16, 2023. Medium. https://itnext.io/static-website-over-https-with-s3-cloudfront-gatsby-continuously-delivered-b2b33bb7fa29

Top 10 Best Container Software In 2023 (May 8, 2023). Software Testing Help. https://www.softwaretestinghelp.com/container-software/ [2]

What are the Differences between IAM roles and IAM policies?(March 3, 2022). Learn AWS. https://www.learnaws.org/2022/03/03/iam-roles-policies/ [12]

What is Containerization (n.d.). AWS. https://aws.amazon.com/what-is/containerization/ [3]