Ron Bal

CS-470 Project 2 presentation script

2/25/2024

**Purpose of Presentation**

- •Good morning, everyone. I'm Ron Bal and today I will be discussing the migration of a full stack application, setting it up within a serverless environment, utilize cloud storage, and a cloud database.

- **•Containerization and Orchestration**

  - •The model used here in this project was containerization, which involves packaging the application into different containers, allowing deployment, scalability, and management easier. This method creates a consistent runtime environment across the different environments.
  - •A couple different tools were used when implementing containerization. Docker was used to build, run, and manage the containers. Amazon ECS to manage and scale the containerized applications across hosts.
  - •Docker compose is a great tool that defines and runs multi-container Docker applications. We can specify services, networks, and volumes that are required for the application with a single YAML file making the management of complex environments easy.

- **•The Serverless Cloud**

  - •Serverless
    - •Serverless is a cloud computation model where the application management is outsourced, reducing the need to upkeep and purchase hardware. It creates an environment that can be scaled on demand, which allows varying levels of traffic to be handled without manually intervening, leaving the application responsive under heavy load. They operate under a pay-as-you-go model,

where only the resources necessary for execution are used and charged. Having a cloud-based application ensures high availability and fault tolerances because they are deployed across multiple zones. The cloud provider automatically handles the load balancing, failover, and recovery, reducing the risk of downtime in case of hardware or other failures.

•S3 (Simple Storage Service) is a scalable object service offered by Amazon Web Services as a storage solution. Data is stored and retrieved from anywhere on the web, making it a very versatile and used for a multitude of storage concerns. S3 is scalable, being able to store essentially an unlimited amount of data, which automatically scales to accommodate any growth in storage needs without having to plan for capacity increases. Data that is stored within S3 is extremely durbale and resilient to hardware failures or errors. Data is replicated across several availability zones creating high availability. This storage solution is accessible from anywhere on the web, which is great for data that needs to be accessed globally. Since it provides RESTful API's and SDKs, integration with applications and services are easy. Local storage has a limited capacity and in order to scale, additional harddrives will need to be purchased. The reliability of the hardware and backup strategies control the availability and durability of the data. Local storage can only be accessed from the server or local network. Local storage can be quite costly when you include the cost of purchasing hardware, frequent maintenance, and potential data center space.

•Describe the advantages of using a serverless API.

•Lambda API logic is the business logic AWS implemented in AWS lambda funcitons that handle API requests and responses. Functions are defined that are triggered by API Gateway events when Lambda functions are used in the creation of a serverless API. The functions contain code that processes incoming requests, data validation, authentication, or data processing, and generates the appropriate response.

•A serverless API utilizing Lambda functions will need a few different scripts or configurations. Each Lambda function will need code that handles the API logic that can parse incoming requests,

execute logic, and generate responses. The API Gateway will need to be configured to define the API endpoints, integrating them with the Lambda functions. This involves setting up routes, methods, requests/response, and security settings. The deployment process was automated with the use of an an AWS SDK.

- When integrating the frontend with the backend in a serverless architecture environment, the API endpoints will need to be defined that the frontend will interact with. They will correspond to the Lambda functions that were implemented to handle various tasks. The Lambda functions will need to be implemented to handle the requests defined by the API endpoints. They convert the logic that process requests and interact with other services or databases to generate a response. The API Gateway will need to be configured to define the API's routes and methods, corresponding them to the appropriate Lambda functions. The frontend will need to be updated to make HTTP requests to the API endpoints defined by the API gateway.

- MongoDB is flexible, can be horizontally scaled, can handle large volumes of data, and handle different queries such as key lookups, joins, and aggregations. Performance is affected by hardware configuartions and the schema design. Through Atlas, mongoDB can be self0hosted and fully managed by a cloud service. Deployment, scaling, and maintenance can also be handled through the use of Atlas.
- DynamoDB provides predictable performance and can seamlessly scale, has low latency, doesn't need to be managed by users, and has scalable cost. It's fully managed by AWS and is designed to be highly available and scalable. The changes in workloads are automatically scaled to accommodate changes by distributing data across servers. Latency is low, which makes it a great solution for applications that require a high throughput at any scale. Since DynamoDB is managed by AWS, there's no need for it to be managed by users.
- The queries we performed were: Get an item, scan entire table, update item, delete item, and query a table.

- The scripts that need to be produced are Lambda functions, serverless configuration file, API Gateway configurations, request and response templates.

- **Cloud-Based Development Principles**

  - Elasticity in a cloud-based model is the ability dynamically scale resources of a system in either direction to meet changing demands. The resources being, computing power, storage, memory, and networking. Within computing power elasticity refers to the scaling of instances in either direction. In storage, storage capacity can be increased or decreased as necessary. Network elasticity allows the adjustment of bandwidth, load balancing, and other related resources.
  - Pay-for-Use Model is exactly how it sounds. You pay for what you use, which is great for applications that may not require a lot of resources. With this model, you can minimize costs and meet the demand of your application.

- **Securing Your Cloud Application**

  - To prevent unauthorized access we can implement multi-factor authentication for the user accounts. Use role based access control, ensuring users only have the minimum level of access necessary to effectively do their jobs. We should also regularly review and update access permissions.
  - Roles define the permissions that are granted to users, applications, or services within a cloud environment.
  - Policies are the documents that specify the permissions and rules for the access we wish to give.
  - In this project we created policies that allowed users to read and write.
  - How can you secure the connection between Lambda and Gateway?
  - For securing Lambda and the database, we followed the principles of least privilege, giving only the necessary permissions. Identity and Access Management roles were used to control the Lambda functions when accessing the database resources.

•S3 Buckets can be secured with the use of AWS' identity and access management by using bucket policies and regularly updates.

•Scalability, one of the advantages of cloud development is having the ability to quickly scale resources based on the demand. Developers are able to accommodate usage patterns and traffic changes without being limited by an in-house operation.

•Security, it's important to implement a multi-layered security approach, to protect against things such as data breaches or unauthorized access. This can be done through encryption, role based access control, and or a monitoring tool.

•Serverless architecture is a great approach that supplies developers with the ability to build services without having to manage a physical infrastructure.