Dominic Drury

10/14/2024

CS-470

Video Link: <https://youtu.be/1ZoAPPd-afc>

Project Two Conference Presentation: Cloud Development

**Overview**:

Hello and welcome to my presentation. My name is Dominic and today I will be discussing the intricacies of cloud development for both the technical and nontechnical individual.

**Containerization**:

The models used to migrate the full stack application to the cloud in this example are Containerization and Lift and Shift.

In containerization the application is packaged, deployed, and managed within isolated environments called containers. This model is highly flexible, efficient, and scalable both vertically and horizontally. The example in this course used a multi-container application, meaning that one container managed the front-end, another managed the back-end, and a third managed the database.

In a Lift and Shift model the application is migrated "as is" to the cloud without making any significant changes. The positives to this model are that is fast and cheap to perform since code changes will be minimal to non-existent. The negatives are that doing this will likely not take full advantage of the native features of the cloud service and that any inefficiencies that existed will persist after the migration. The example in this course also used the lift and shift method utilizing AWS.

Moving back to containerization, there are several tools that are needed in order to create a containerized application. You will need an engine like Docker to create, manage, and run the containers. You will also need an orchestration tool like Docker Swarm to manage the lifecycle of the container. Finally, you will need an image registry like Docker Hub to have a central repositor for the container images being used.

**Orchestration**:

There is a lot of value to using docker compose. The first being that setting up a multi-container application is easy and simple allowing users to have a different container for the front-end, back-end, and database. The next being the that since Docker uses a single YAML file for its configuration you can configure the application in one easy to manage place. Docker Compose also offers automatic dependency management so containers like databases are started before other containers that rely on it to function. Docker compose is cross-platform, so it behaves the same in development, testing, and production regardless of the environment. Having the different containers also means that the application is easily scalable, and different sections can be scaled up or down to meet the current needs of the application. The next reason to use docker compose is the simplified workflow, it offers a few simple commands that can allow you to build a container, start or stop a container, and monitor logs from containers. Finally, there is the portability offered by having such sharable containers and configuration files that function regardless of the environment.

**The Serverless Cloud**

**– Serverless**:

While serverless is a bit of a misnomer since a server is still used, this model shifts the management of the server to a cloud provider like AWS, Azure, or Google Cloud. There are several advantages to using a serverless model. Since the cloud provider deals with the server there is no infrastructure management, the pay-per-use model means that it is cost efficient, the platforms provide auto scaling both up and down to meet current needs, and finally since the service is Function as a Service resources are never underutilizing and only allocated when needed.

Amazon simple storage service, or S3, is the cloud-based storage service provided by Amazon Web Service. It allows its users to store and retrieve data from anywhere that has an internet connection. Compared to local storage it has better accessibility, scalability, durability, redundancy, and better cost for any medium to large scale use case. The only times that local storage is better is if the storage needs are low enough that hardware is cost effective, transferring is only needed on a local network, or when internet access is limited.

**- API & Lambda**:

The three main advantages to using a serverless API are all that I have mentioned on previous slides. Serverless means that all the management of the servers are done by the cloud provider, the scaling is automatic both up and down so resources are only allocated when being used, and the serverless API allows the application to be developed, tested, and released on all environments that have an internet connection.

Lambda API Logic is the logic behind AWS. An API Gateway receives an HTTP request from a user, that gateway then triggers a lambda function that executes and returns a response, and finally that response is forwarded to the user by the gateway.

The scripts produced in a basic Lambda API are the Lambda function code that is triggered by the API Gateway and is where the API request is made, the infrastructure code in the YAML file that connects the AWS resources, and the IAM roles that grant appropriate permissions to users to make the HTTP requests.

- **Database**:

MongoDB and DynamoDB are very similar in functionality and usability but do differ in a few ways. MongoDB uses documents while DynamoDB uses a key-value pair for data storing, Mongo uses nested documents while Dynamo uses a single table, and Mongo allows for horizontal scaling with sharding while Dynamo auto scales based on needs.

While both allow for CRUD functions, the actual methods are a bit different. MongoDB uses native ts files to manage the HTTP requests and responses and DynamoDB uses API Gateway with Lambda functions and IAM roles to manage its queries and results.

To perform these query operations, MongoDB uses mongoose to send and receive requests made and DynamoDB uses AWS to link the gateway and lambda to perform its requests

**Cloud-Based Development Principles**:

One of the biggest draws to using cloud-based services is the elasticity of resources and the pay-for-use model. Elasticity refers to how serverless models automatically scale up and down resources based on the current needs. Pay-for-use means that the user gets charged only for the resources they use. This auto scaling and pay model means that in the times when actual usage and planned usage do not align you both do not experience interruptions in service and do not waste money on resources that you do not use.

**Securing Your Cloud App**:

Securing your cloud app is done by using IAM roles with appropriate policies and secured connections in your API. The first step is to create the Identity and Access Management roles that will be needed by the different types of users in the application like management, employees, users, IT, etc. For this class I used the LabRole IAM role to grant access to the read and write policies needed for my Lambda to function.

To protect your API security there are several steps that can be taken. The first is to secure the connection between the Lambda and gateway. AWS uses HTTPS by default to encrypt the data that is in transit and as a developer you can limit access using IAM roles and abiding by the principle of least privilege. The next step is to protect the data between the lambda and database, this can again be done using IAM roles and the HTTPS but also use a secrets manager to encrypt the credentials that can be used to gain access. The third step in securing the API is to again utilize IAM roles, keep the S3 bucket private, enable encryption for data in transit and at rest, and utilize the bucket key to lower the costs of encryption.

**Conclusion**:

Out of the many points that I have made regarding serverless applications, I think that the three main points are that with serverless applications, like that offered by AWS, there is no server infrastructure that needs to be created, maintained, or repaired, resources are automatically scaled up and down to meet the needs of your application, and finally that keeping the API secure can be done simply and efficiently by utilizing the default HTTPS for encryption and IAM roles to manage user access to the least privilege that they need in order to complete their goals and tasks.