Title Slide

Hello, my name is Matthew Muller and this is my presentation on cloud development for the CS-470 class Full Stack Development II.

Overview

So just a bit about me to begin: I am a Computer Science major at Southern New Hampshire University and I am currently in my final term of classes with just an Internship left to complete to graduate. The purpose of this presentation is to communicate the intricacies of building, testing, deploying, and running software services in cloud environments to both technically informed and nontechnical audiences.

Containerization

One important concept in cloud development is containerization. Containerization is a software development process that bundles all of the components of an application into a single container image that can then be run in any environment. There are a number of different models that are commonly used for migrating full stack applications to the cloud using containers. The first model is Lift-and-Shift, also known as rehosting. In this model, an exact copy of an application, including its data stores and operating system, are migrated from one environment to another with no change to the application architecture and little or no change to the application code. This allows for faster and less costly migration compared to other models but may not allow for the use of all cloud capabilities. Another commonly used model is known as Platform as a Service. This model involves the use of a third-party–hosted server or operating system to develop and run applications. It simplifies application development and management because it eliminates the need for a business to maintain onsite infrastructure. However, it can have limitations in terms of customization and control over the applications underlying infrastructure.

One more commonly used model is Software as a Service. This model allows users to instantly access a collection of applications on the cloud by installing software through a cloud-hosting service and then setting up access permissions for multiple users in an organization. This makes it a good option for companies that require a lot of collaboration but they often run slower than applications that are run on a business's own servers. The two tools that you will need for containerization are the container itself and a container management software. In this class we used a Docker container and used Docker Compose for container management.

Orchestration

Docker Compose provides a number of benefits for utilizing containerization. It offers single host deployment, which allows you to run multiple components on a single piece of hardware. It provides fast and easy configuration due to its use of YAML scripts. It allows for high levels of productivity as it reduces the amount of time it takes to perform tasks. And finally, it has security benefits as all containers are isolated from each other, which reduces the threat landscape.

The Serverless Cloud

The serverless cloud; so what does serverless mean and why use it? Serverless is a cloud-native development model that allows developers to build and run applications without having to manage servers. Applications built with a serverless infrastructure are inherently scalable and will scale automatically as the user base grows or usage increases. This works hand-in-hand with the pricing model of serverless architecture, which is a pay-as-you-go format that only charges companies for what they use. It allows for flexible applications that can be expanded or updated quickly which decreases time to release as developers can very quickly upload code all at once or one function at a time. A common cloud-based storage solution is Amazon’s S3, which stands for Simple Storage Service. S3 is highly scalable, enabling virtually unlimited storage on a

pay-as-you-go pricing model. Like local storage, it is highly secure and protects file transfers and file storage with encryption technologies. However, one disadvantage as compared to local storage is increased latency leading to slower uploads and downloads.

The Serverless Cloud (Pt 2)

There are many benefits to using a serverless API. It makes it so that companies don't have to worry about server management, provides seamless scalability, reduced latency, a cost efficient pricing model, and fast development and deployment. Lambda API is an event-driven compute model provided by Amazon Web Services that runs code in response to events and maintains the computing resources required by that code automatically. Lamda supports numerous programming languages such as Java, Powershell, Node.js, Python and others. Now to discuss the integration of the front-end and the back-end. It begins with the creation of a Lambda function to respond to a specific request or perform a certain action. Next is the configuration of an API gateway to serve as the communication channel between the front and back ends. Then CORS is configured to allow the application to interact with resources from different domains. Finally, authentication and authorization are implemented using IAM.

The Serverless Cloud (Pt 3)

Two commonly used NoSQL database services are MongoDB and DynamoDB. MongoDB has a free, open-source community version but has also recently released Atlas, a pay-as-you-go cloud service. MongoDB needs to keep its working set in RAM to achieve acceptable performance, which can make it too expensive for many use cases. It is able to run on any platform unlike DynamoDB, which only runs on AWS. DynamoDB also uses a pay-as-you-go pricing model and delivers excellent performance, no matter how many records you store.

Cloud-Based Development Principles

Now to talk about the concept of elasticity and the pay-as-you-go model. Cloud Elasticity is the property of a cloud to grow or shrink capacity for CPU, memory, and storage resources to adapt to changing demands. This process can be automatic or manual, with the manual process consisting of the company receiving a notification regarding their resources and reacting accordingly. The pay-for-use model is a highly flexible pricing plan that charges companies or users based on only the resources that they actually use. This differs from traditional pricing models in which you pay for a set amount of resources, which may not all be utilized. These two concepts together help make cloud development extremely scalable.

Securing Your Cloud App

Now to talk about securing your cloud application. One tool for managing access on an application is IAM. IAM allows us to create and manage user identities and assign them specific permissions. We can also enable multi-factor authentication, which requires users to go through another layer above entering a password in order to log in, such as entering a code sent to a mobile device. Encryption provides another layer of security by ensuring that even if unauthorized access occurs, the data will be encrypted. The relationship between IAM roles and policies in AWS is that a role is a type of IAM identity that can be authenticated and authorized to utilize an AWS resource, whereas a policy defines the permissions of the IAM identity. In order to secure the connection between Lambda and Gateway, we set up API gateway authorizers to authenticate and authorize incoming requests before they reach the Lambda function. For Lambda and the database, we can assign IAM roles to our Lambda functions, granting specific permissions regarding their database access. Finally, we can apply policies to our S3 bucket to define who can access the bucket and what actions they can perform.

Conclusion

To recount a few main points regarding cloud development, containers bundle all of an application's necessary code and infrastructure into an image that can then run in any environment, allowing deployment to the cloud. Serverless computing allows companies to deploy full stack applications to the cloud without having to worry about managing their own server, allowing them to focus on their own product development. And finally, cloud computing provides a wide array of benefits, from seamless scalability and flexibility, to reliable security, and also a cost-effective pay-as-you-go pricing model that only charges for the resources an application actually uses. Thank you for your time.