Project Two Conference Presentation: Cloud Development

https://youtu.be/ow2UItshxew

James Porter

Department of Computer Science, Southern New Hampshire University

CS 470 Full Stack Development 2

Professor Mohammad Alam

December 11, 2022

Slide 1: Title

Good afternoon everybody, I am James Porter, a software developer here to talk about my experience migrating our full stack application to a serverless environment by using cloud storage and a cloud database.

Slide 2: Overview

With that being said, we are going to discuss some topics today and give you some tools that might be useful when migrating your own full stack applications to a serverless environment such as AWS. We are going to start with containerization and discuss what tools are necessary for this process. Next, we are going to discuss the parts of the serverless process and how they parallel the MEAN Stack. This includes S3 storage, API, Lambda, and DynamoDB when using AWS. Then we are going to discuss elasticity and how we can minimize waste when we migrate to serverless applications. Lastly, we are going to wrap things up with security and then if you have any questions, you can ask them at the end.

Slide 3: Containerization

Before we can discuss some of the tools we need to use containerization, we need to first discuss what a container is. According to IBM cloud education, a containerized project is a lightweight packaging of everything needed to run the software in one place. This allows us to have code that is more portable and very resource friendly.

Another technology that is very similar to containerization is called a virtual machine. The biggest difference between the two is the lack of a hypervisor layer in a container. Otherwise, both technologies are very similar so it raises the question of whether we should use a virtual machine or a container.

According to Matthews, if you are making simple changes to an already completed system, it would be advantageous to just use a Virtual Machine, but if you are starting fresh, containerization would be beneficial by providing increased development speed as well as an increase in security. However, no matter which you go with, you can change your mind, and (with a little work) it is possible to switch between a Virtual Machine and containerization.

Now that we have some definitions and terms, we can discuss the main tool we used for containerization and that is the Docker Desktop application and Docker Compose using the Windows PowerShell or a similar command window.

Slide 4: Orchestration

Containers in and of themselves are wonderful tools, but there are some drawbacks. The biggest of which being that containers do not communicate with each other without explicit commands to do such. This is not an issue when you are working with one container at a time, but anytime we need two or more containers to work together there are some extra steps that need to be taken to overcome the drawbacks of no communication.

This is where docker compose comes in. Using this tool in a command window of your choosing can allow you to let two or more containers work together using tools such as bridges. You can also use the Docker Desktop application to see and run these bridges and containers .

Slide 5: Serverless

To run a full stack application, you would need a certain number of servers to make sure that the users are able to visit your application without crashing or causing a slow and causing a negative user experience. This means that if you want a lot of users to be able to access your application you either need to have a large number of servers that allow many users to access it or you need a serverless application.

What is serverless? Well, serverless is the method of moving your resources and routing everything through a third party so you don't have to keep the physical servers like you would with an application that you wanted to run and develop on your own. Hence the name serverless.

We get some advantages from this development and using serverless development. First and foremost, it frees up local resources, so you don't need to keep as many servers in a physical location. Additionally, we have the added benefit of being able to scale as needed when we need to use more, or fewer resources and it reduces waste.

When using AWS, the S3 storage bucket is one way that we can store lots of data, such as program files or dependencies, without needing physical resources to store that data. In our process, we used the S3 bucket to store and run the front end of our full stack application.

Slide 6: API & Lambda

The next part of the AWS stack is the interaction between API and Lambda. These two tools have many advantages including, and like the S3 bucket, there is no need for a server or physical storage. Additionally, this can lead to faster development times and less concerns over resources because of the serverless environment.

These two tools work together nicely and can be seen as two sides of the same coin. On one side of the coin, we have the lambda functions that we write which tell us how we should execute orders and on the other side we have API which tell us which links or addresses to use to initiate these changes. In our development process, we combined this process with the S3 bucket that we created in the first section and added a couple of lines of code to make the front and the back end communicate and thus creating an almost full stack running in AWS.

Slide 7: Database

The last tool we need for our AWS stack is the database. In our AWS stack we used DynamoDB, which is very similar to MongoDB, but there are some key differences.

First, and the biggest difference is that MongoDB can be used across multiple platforms whereas DynamoDB is limited to AWS. Also like the other parts of the AWS stack, the Dynamo database will scale as the database grows or shrinks.

The biggest difference between the two databases is that DynomoDB uses a single table model which is different than what MongoDB offers. One last difference is that DynamoDB offers very limited querying values when compared to MongoDB.

Slide 8: Development Principles

There are many benefits when using serverless storage, but one of the biggest is the limitation of waste as well as being able to scale as needed to match usage. The elasticity provided by serverless development allows us to take full advantage of the benefits.

Additionally, this model follows a pay-for-use pattern so that you only pay for the resources that you use.

Use graph

For example, let us take a look at the graph on this slide that shows the planned capacity in blue and the actual usage in red. This graph shows that if you were using traditional storage, at first everything scales fine. There is a little scare here (*point to slight cross of blue and red line*), but we quickly get that in check, and everything seems fine, but what if we have a nation or worldwide pandemic and all of a sudden, the demand for your application decreases. We have a lot of wasted resources here that we are not using.

We can also have the opposite happen where we don't have enough resources to handle all the users. In our example, let us imagine that all of a sudden you are featured in a major media outlet and demand for your product skyrockets. When too many users start trying to access your application and there is not proper computing power, then we have customer dissatisfaction, and we run into the potential for losing more users, because of this mishandle of resources.

All of this hassle can be mitigated by moving to a serverless full stack application which scales as your usage increases.

Slide 9: Securing Cloud

When allowing users access, it is important to control what they are allowed to access and what they should be forbidden from seeing. This can be done in a number of ways both simple and complex, but the first way is a practice that we should keep in all of our developments even beyond the AWS stack,

That concept is the principle of least privilege. This principle essentially says that each user should only be given access to just what they need and not anything more. This helps make sure that other security measures that are put in place can work properly and effectively.

This leads us to our next access method and that is IAM Roles. These roles can be assigned to an individual to give them access to what they need to get the job done. I will talk more about this in a little bit, but first we need to identify the connection between roles and policies.

This connection is quite simple and the roles you create for individuals give them permissions, but the policies that you follow enforce need to reflect the permissions that you give to each user.

In our development process this meant implementing custom policies to allow for CRUD operations with our database and allows the user to enact these changes according to permissions. The last step we are going to discuss about is API security.

Security of the cloud is managed by AWS, but we are responsible for making sure that we have security within the cloud to protect our information. The first area we need to protect is the connection between Lambda and Gateway. This involves making sure that the lambda functions are properly linked to the correct API link so no improper query is made. We can also limit the users that can have access to certain API's and only the API's that they need access to.

The next step is to make sure that the Lambda functions and database is properly secured. This means that we make sure that the calls to the database are prewritten out and that someone can't try and get access to data using an SQL injection or similar technique. This involves limiting the data that they user can enter into requests and making sure all database calls can not be infiltrated.

S3 Bucket security measures are very similar to other security measures that we have discussed, but we can also disable or limit the access that certain users have to the bucket. This is an extension of the policy of least privilege.

Slide 10: Conclusion

There are a wide variety of resources that can be saved by migrating to a serverless application. This can be everything from physical resources such as hardware and money saved from needing and maintaining those resources to the non-physical resources such as time.

We have also shown how easy it can be to move your MEAN stack application to a serverless application using a cloud service such as AWS.

BUT We also need to keep in mind that serverless applications are not always the answer and there are other ways to develop applications that work and provide the best user experience.

Thank you so much for your time and if you have any questions, you can ask them now.

Slide 11: Sources

This slide has no voice over and is just there for the sources I used for my project.