CS-470 Project Two

Project Two Conference Presentation: Cloud Development

Christopher Clark

December 11, 2022

Video Presentation Link (YouTube Unlisted)

https://youtu.be/kvnoSfsHyIQ

Slide One

This is a conference presentation based on my CS-470 Project Two for Cloud Development.

Slide Two

Hello, and thank you for joining us here today. My name is Christopher Clark, and I would like to speak with you about our success in developing a full stack application and migrating it to the cloud. We have employed the services of Amazon's AWS using S3, Lambda, API Gateway, and DynamoDB. We have also had great success with our initial development in using Angular front end, MEAN stack API and MongoDB with Docker for our application containers.

Slide Three

When migrating an application over to the cloud there are several models to consider. Some of those include *lift and shift, SaaS (Software as a Service), Refactoring, and Re-platforming.*

Lift and shift is considered to be the fastest and easiest way to migrate to the cloud. In this model we use the same software with little to no learning curve for personnel. The application will look and feel much like it had prior to migration. (Team & Written by Shadow-Soft Team View more posts written by this author », 2022)

With SaaS, or software as a service, we only migrate services that we need. If you have a need to free up resources, there are cloud-based services that specialize in certain applications and can manage them for you. This can reduce the need for licensing these services. (Team & Written by Shadow-Soft Team View more posts written by this author », 2022)

Refactoring is a good option if you have legacy services that need modernization. Refactoring allows you to run your outdated software alongside newer applications while development is underway. (Team & Written by Shadow-Soft Team View more posts written by this author », 2022)

With the re-platforming model we basically replace the entire application down to the base code and build it as could native. Security and updates happen quickly and efficiently. Re-platforming also gives you the option to move from one cloud service to another as needed. (Team & Written by Shadow-Soft Team View more posts written by this author », 2022)

For our application we are using containers to house the services. We have one for our Angular front end, one for our MEAN stack API, and one for our MongoDB. Some of the tools available to enable containerization include *Kubernetes, Docker, and Apache Mesos.*

Kubernetes is a Linux-based tool that allows you to manage your application containers. It is simple and robust. It will manage your containers automatically and is very reliable. (David, 2022)

Docker is a container tool that works with Kubernetes. It runs on top of a Kubernetes cluster that you set up in your program. Docker is best utilized on a locally hosted system. It can prove to be very helpful and does the job well. (David, 2022)

Apache Mesos an open-source tool that is used to manage deploy larger container-based applications. This tool is probably better used in larger applications with a more distributed management system. (David, 2022)

Slide Four

For this application we found Docker to be the best choice and used it to build containers for our applications. We could have chosen to build everything in Lambda, but our use case required the ability to use programming languages not necessarily supported in Lambda. Using Docker gave us the flexibility needed without code refactoring. (Allen, n.d.)

Some of the benefits of Docker Compose include *single host deployment, fast configuration, high productivity and security.* With a single host deployment, we can run everything on one system. Docker allows for configuration YAML files and in this slide, I have included a sample of our YAML file. Here we configure our application and identify the API container and the port it is listening to then link it to the database. Finally, we attach the containers to the network we created. In addition, we can benefit from better security through service isolation reducing the level of external threats to manage. (Afreen, 2022)

Slide Five

Serverless architecture is a cloud-based system that allows for deployment and hosting of applications without the need for local infrastructure. The cloud service provides the tools we need and manages the infrastructure and hosting needs for our system. One key benefit is the scalability of serverless architecture. The service will have the ability to scale our applications up or down as our needs change.

Amazon's Simple Storage Service (S3) is a cloud-based storage service that allows for a great deal of flexibility. With S3 you only pay for what you use, there is no need to figure out how much storage capacity is needed ahead of time because the storage capacity will grow as your needs do. With local storage there is a hefty upfront cost for hardware installation and maintenance. The system capacity is fixed and cannot expand without more cost. Many of the tools available to manage storage on S3 will also not be available to locally hosted storage. (AWS.Amazon.com, n.d.)

Slide Six

Having tools like serverless APIs available to us is very useful. In most cases it is a much better option to leave certain services to third party developers rather than try to create them on our own. Here we are using Amazon's API Gateway. Development and maintenance of the API is handled by AWS. We rely on the speed, availability, and scalability of the serverless cloud to handle our API requirements.

When creating our APIs, we mapped them to Lambda and generated the functions that were required for each HTTP request. Lambda is a computing function provided by AWS. It is an event-based service the executes the code built into each function as a response to events. Through the API different events can be distributed to whatever functions we need, and Lambda will scale up or down based on that need.

The functions we created incorporate CRUD features we needed by building GET, PUT, POST and DELETE methods to run the scripts necessary for these operations. Each API we built, and the attached resources, have these methods. Once configured we were able to test our functions and see the results.

To complete our migration, we needed to integrate the front and back ends. To perform this task, we manually enabled Cross-Origin Resource Sharing or CORS. This required the addition of an OPTIONS method within our API structure. Once completed, we published the site and deployed the APIs to generate our AWS URL. To get the Angular front end going we had to modify some of the code previously generated. In the slide we can see the main.ts file that was modified for this purpose. We now have a published, cloud-based, full stack system using Amazon's AWS service.

Slide Seven

When we initially developed our database, we utilized MongoDB. As we migrated to the cloud, we chose to use Amazon's DynamoDB. One big difference between the two is that mongo is open sourced and available to use on most platforms. Dynamo is part of AWS and is only used in that system. Dynamo is also fully managed by AWS whereas mongo can be self-managed, or you can use a service like Atlas to manage it.

Structurally, mongo is set up to run JSON style documents. Mongo uses collections with key values and documents that allow for nesting. With Dynamo you can use documents as values or blobs. Both support multiple languages. Mongo has excellent support for various datatypes and dynamo is a bit more limited in that area. Queries on mongo are enabled by an in-depth language with use of single keys, ranges, and joins. Dynamo, on the other hand, is limited to local and global secondary indexes. (Rockset, n.d.)

To retrieve the data requested within our system we generate a couple of queries. Through a loopback explorer we were able to add a question to the database as shown in the slide. Once migrated over to AWS we used queries embedded in the Lambda functions we generated. When an event is triggered, the necessary function is executed to complete the request.

Slide Eight

Elasticity is the ability for a cloud-based service to scale resources and adapt to the ever-changing demands of their customers. There are advantages to this as shown in our graph. If we choose intermittent upgrades to infrastructure, we will have a scenario much like what we see in the graph. Client demands for service fluctuate a great deal. Without elasticity we will inevitably see times of excess infrastructure as well as times when client demands cannot be met. Both of those scenarios can cost money and generate customer dissatisfaction. (Vmware, 2022)

Most serverless cloud providers have a pay-for-use model when charging for their services. With AWS we only pay for the services we use and only for the time we use them. There are no contracts to worry about and AWS handles all the licensing requirements. (Bigelow & Sullivan, 2022)

Slide Nine

One very important issue to consider with every application is security. How do we control access to our site and prevent bad actors from creating issues for us and our clients? For this project we are relying on the security features embedded in AWS. AWS uses an Identity and Access Management (IAM) service to help provide secure access. Using this and establishing roles for the application can help secure your environment and prevent unwanted access.

Roles allow us to set up specific entities and grant certain permissions for what they can and cannot do within AWS. When granting access, we must assign specific permissions to the role by creating a policy and attaching it to the role. Policies are what we use to define the permissions given to a user. (AWS.Amazon.com, n.d.)

For our system we created a policy to help us secure our system. We started with DynamoDB. Some of the options we set up include get, query, and scan for the Read section as well as delete, put, and update in the Write section. Once this policy was attached to the role, we were able to run and test our functions.

Securing the Lambda to API Gateway services can be accomplished by using multi-factor authentication, user logging, and properly managing user data. Fortunately, Lambda API always utilizes transport layer security between endpoints as well as data encryption. With S3, the connection to Lambda is natively secure, but does require some work on our part. We need to be sure we properly set up application roles and develop security policies that work for our purposes. (AWS.Amazon.com, n.d.)

Slide Ten

In conclusion, I feel that learning about containers has shown me just how useful they are. Containers provide the portability and ease of use that make development much more efficient. Since everything the application needs is contained within that space, we can run applications over our primary OS and retain the control over what resources it can use.

I also now have a much different opinion of cloud-based, serverless, development. I have always held to the idea that local infrastructure and personnel is a better solution for most situations. In using could development for this project, I have changed my mind. The benefits of having everything you need in one package are seemingly endless. The scalability and cost effectiveness of cloud development is certainly something to consider for each business case we deal with.

Finally, as our industry moves forward, security becomes more important. Having our systems hosted by cloud-based providers with good security options can help us ensure our data is safe and our systems remain available 24/7. The configuration of roles and policies used in AWS is efficient and does not require a massive learning curve for system operators.

I want to thank you for your time today. This has been an eye-opening journey and I am grateful for the opportunity to be a part of this team. Thank you.

References (Slide Eleven)

Afreen, S. (2022, November 23). *What is Docker compose: Example, benefits and basic commands: Simplilearn*. Simplilearn.com. Retrieved December 10, 2022, from https://www.simplilearn.com/tutorials/docker-tutorial/docker-compose

Allen, J. (n.d.). *When to use AWS lambda vs docker*. TinyStacks. Retrieved December 10, 2022, from https://www.tinystacks.com/blog-post/choosing-between-aws-lambda-and-docker/

Bigelow, S. J., & Sullivan, E. (2022, October 25). *What is pay-as-you-go cloud computing (PAYG cloud computing)? A definition from techtarget.com*. Storage. Retrieved December 10, 2022, from https://www.techtarget.com/searchstorage/definition/pay-as-you-go-cloud-computing-PAYG-cloud-computing

David, M. (2022, January 26). *Top 5 open-source tools for containerization*. TechBeacon. Retrieved December 10, 2022, from https://techbeacon.com/devops/top-5-open-source-tools-containerization

Team, S.-S., & Written by Shadow-Soft Team View more posts written by this author ». (2022, July 12). *Cloud migration models: How to pick the right strategy*. Shadow. Retrieved December 10, 2022, from https://shadow-soft.com/cloud-migration-models/

What is Amazon S3? - amazon simple storage service. (n.d.). Retrieved December 10, 2022, from https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html

Mongodb vs dynamodb head-to-head: Which should you choose? Rockset RSS. (n.d.). Retrieved December 10, 2022, from https://rockset.com/blog/mongodb-vs-dynamodb-head-to-head-which-should-you-choose/

What is cloud elasticity?: Vmware glossary. VMware. (2022, December 1). Retrieved December 10, 2022, from https://www.vmware.com/topics/glossary/content/cloud-elasticity.html