Assignment – 3

Consider a monolithic java application stack.

Apache Web Server, Apache Tomcat application server with Active MQ and Oracle and MongoDB backend.

In order to migrate an application to AWS Cloud, following approach need to be followed :

1. **Motivation to Migration :**

We need to be very clear as to what needs to achieved via migration.

* Improve scalability without investing into new hardware.
* Cut down administration costs by automating deployments.
* Flexibility , fault-tolerance, Zer o downtime.

1. M**igration Plan, Strategy, & Execution Steps :**

* Cloud Assessment

On financial assessment, We should be able to map the hardware configuration of physical servers to equivalent EC2 instance types and estimate the combined storage and bandwidth requirements.

* Proof of Concept

The outcome of PoC should be like a prototype of the application running as per requirement on the Cloud using different services. Get performance metrics using Amazon CloudWatch, so as to reach to a conclusion on factors like durability, fault-tolerance, high-availability etc..

* Data Migration

Once the proof of concept was complete, **move all of the application’s static files** (Images, JS, CSS, video, audio, and static HTML content) into an Amazon S3 bucket, created a CloudFront distribution of that Amazon S3 bucket, and modified the references in web pages so that end-users get the content directly from Amazon S3 and Amazon CloudFront. With the help of a few scripts and the AWS SDK for Java library, this can be done.

* Application Migration

During the application migration phase, the development team **launched both small and large instances3** for their web and application servers. They modified their build and deployment scripts to use the cloud as an endpoint. **Security Groups** were defined to isolate web servers from the applications and database servers. Testing (functional, load, performance etc.) should be performed to ensure that the systems were performing at expected levels, and that exit criteria for each component were met.

**Co-existence Phase**

During the migration phase, the collocation infrastructure was not deprecated immediately. employe a **hybrid migration strategy** during the migration of all web and application servers. The configuration of the on-premise hardware load balancer was modified to send requests to the new instances in the cloud. For a short duration, the load balancer was routing traffic to the servers in the cloud in as well as to the physical servers. After verifying that the servers in the cloud were performing at required levels, the physical servers were dismissed one by one, the load balancers were updated, and all of the web traffic was being served up by the EC2 instances running in the cloud.

After testing was completed, the DNS was switched to point to the cloud-based web servers and the application was fully migrated to the AWS cloud.

1. **Leveraging the Cloud**

Once the production site was launched, Company A was looking forward to the time when they could use some of the advanced features of AWS. The team automated some processes so that once the server is started it could be easily “attached” to the topology. They **created an Auto Scaling group** of web servers and were able to provision more capacity automatically when specific resources reach a certain threshold (Apache web servers CPU utilization above 80% for 10 min). The team invested some time and resources in **streamlining their development and testing processes** to Amazon Web Services **Migration Scenarios: Web Application Architecture** make it is easy to clone testing environments. They gained lot of experience using AWS resources and also invested time in **leveraging multiple Availability Zones** for even higher availability.

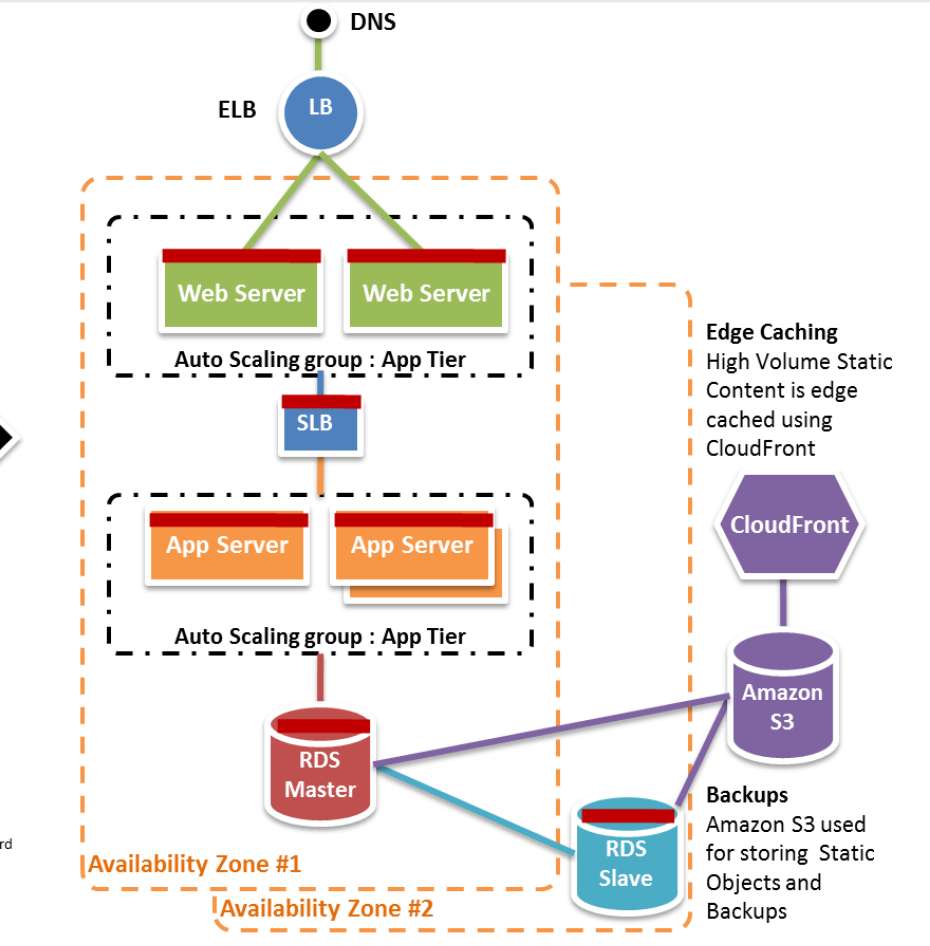
1. **Optimization**

During the optimization phase, we need to analyze the utilization The team also integrated Amazon CloudWatch into their existing dashboards so that they can monitor the system metrics of every instance in their cloud fleet.

1. **Conclusion**

Using the phased-driven approach, the development team was able to resolve all the financial, technical and social-political concerns. Deciding to invest in a proof of concept proved extremely valuable. The resulting architecture was not only elastic and scalable but also flexible and easier to maintain.

Sample Screenshot :



**AWS Services that will be used for migration :**

1. **Apache WebServer** – EC2 (EC2 instance having web server installed on OS say Linux)
2. **Apache Tomcat Server with ActiveMQ** - EC2 (EC2 instance having Apache Tomcat/ Application server installed on OS say Linux).
3. **ActiveMQ** can be substituted by SQS but only after cross-examining all requirements
4. **Oracle** – RDS (Endpoint is mapped to EC2 instance)
5. **MongoDB** – DynamoDB (Endpoint is mapped to EC2 instance).
6. **Route53** – DNS records to be managed by DNS.
7. **Amazon S3** – For object storage.
8. **Amazon EFS** – For file storage.
9. **CloudFront** – For caching most visited objects,files.
10. **CloudWatch** – For monitoring.
11. **Load Balancing** – ELB (Application Load Balancer for application servers and Classic load balancer for web server )
12. **CloudHSM** – If you have specific requirements for HA else ELB can be configured for HA.

**Note** : Amazon SMS can be used for Server Migration.

Furthermore optimation can be done base on :

1. What t-shirt size/ instance type to be used to create EC2 instance.
2. For Database, setting of Provisioned throughput.
3. As new features are keep on added to AWS services, regularly go through pricing models and use-cases so that optimization can be acieved as much as possible.

Reference document :

Amazon Web Services **Migration Scenarios: Web Application Architecture**