Hi Sir,

As per my understanding to the scenario given, I am assuming that we are having a monolithic java application and we need to migrate that on AWS cloud.

Also, assuming that this application may be running in-house or on any other cloud service.

Migration is something like moving data from one place to another.

AWS SERVICES USED :

1) Amazon Elastic Beanstalk
2) Elastic Load Balancer
3) Auto Scaling
4) EC2 Instances
5) VPC , Security Groups, IAM Roles
6) Relational Database Service - Sql and No Sql
7) SQS
8) Route 53

The very first step that we need to follow is to gather the requirements and estimating the cost that we need to pay.

Every application works on 3-Tier principle, that includes

- GUI (Graphical User Interface) or the Interactive Windows
- Web Servers used
- Database

When we go for the Migration we start with migrating the GUI first, after that we migrate db at the last.

We can either use fully managed platform that AWS provides called Amazon Elastic Beanstalk or we can go for the below defined steps -

- we can have an **Elastic Load Balancer** attached to two server(2 **EC2 Instances**)having apache and tomcat installed in them. When the ELB will get the hit, it will evenly distribute the load to both the servers.
- Or we can have two servers of apache attached to ELB, and then two servers of tomcat that are connected to apache.

So, we will start with Migrating the Application part first.

We are using apache and tomcat both, with Apache serving static content and forwarding the requests for dynamic content to Tomcat. We know we are migrating a java application, it will be either in jar/war format so we will be using tomcat for deployment.

Apache will be working as wrapper. Tomcat cannot maintain much connection, so to maintain the request and connection we use apache.
Until the DB migration happens, Our application will be pointing to the current running database.

After the replatforming, we will be going for the database.

For Active MQ, We can use the service called **SQS(Simple Queue Service)** that AWS provides.

- It is a fully managed message queuing service. It makes it simple and cost effective to coordinate the components of a cloud application.Using SQS, you can send, store, and receive messages between software components at any volume, without losing messages or requiring other services to be always available.

Let say we are having an application running, and there are some consumers and vendors who wants to use the application, so the full load will be on the application and it may die, so instead of this we can use SQS, that provide us a DNS, and the vendors can upload their messages, SQS Keeps all the messages in the queue and keeps on updating in the database.
So when a consumer wants to use the application the load will be minimized on the application, it will take the data from database as per the requirement.

For Database, we can use **RDS (Relational Database service)**. we can use database product Oracle. We can have control who can access our RDS databases by using **AWS IAM** to define users and permissions. We can also protect the databases by putting it in a virtual private cloud.

To avoid excess load on database we will create a read replica of RDS instance which will handle the read operations only.

As Mongodb is a nosql database type, we can also use many other database service that RDS provide such as Oracle .

While migrating the database we can follow 2 ways -

- If app is not running then we can take the Database dump & import in RDS DB instance.
- If app is running then we can use DB replication from our running database to fresh created RDS database to prevent the data loss.

Now, we can change the URL(For URL, we can use ROUTE53 service provided by AWS), and allow our application to point at the migrated database.

For High Availability and Load Balancing we can use another service of AWS called as **Auto Scaling** which will create new instance from the **AMI(Amazaon Machine Image)** of previous instances and will equally balance the load using Load Balancer.