**Problem**

**========**

Consider a monolithic java application stack - Apache Web Server, Apache Tomcat application server with Active MQ and Oracle and MongoDB backend.

Propose a solution to migrate this application stack to AWS. Mention all the AWS services you would use and how you would maintain HA and Load Balancing (consider app to be stateless). Mention rationale for each design decision.

**Approach**

**=========**

we need to deploy an monolithic appication stack

for that we need to take our requirements,

Apache web server EC2

Apache Tomcat EC2

ActiveMQ SQS

RDS Oracle

Non-Relational DB MangoDb

Networking VPC

Storage S3

Infrastructure Management CloudFormation Stack Or Elastic beanStalk

High Availability Autoscaling and ELB

Authenticated users IAM

Monitoring service Logs CloudWatch

**Launch Apache Webserver and Tomcat**

**==================================**

The Apache webserver and Appllication server is deployed on the EC2 Instances and and set up with appropriate Inbound and Outbound rules in the Security Groups ,which certainly provides us with instance level security.

=> First of all we need to create an Image of Instance which is configured with Infrastructure needed to run on our application on Ec2 instance ,so that we can create a image and add it to the Auto scalling group and ELB which provides us with High Availability by maintaining the Instance in multiple Availability Zones, and Fault tolerant .

=> while configuring the Image of an Instance we should consider all the security groups Inboud and outbound of the instance where we need to have our web server security groups allows traffic to and from the application server only,and application servers allow traffic to RDS Databases.

=>It is also ensured that only HTTPS connections are made to the application by only allowing traffic on port 443 and not HTTP port 80.

=> Other ports has to be opened which are used to communicate with the Oracle db

=> webservers and application server and database servers has to be deployed in Demilitarized Zone which is in the Private Subnet ,However Public Facing ELB will be available inside the public Subnet.

**Authentication**

=============

Authentication has to be provided so that only privileged users have the access to aws resouces and permissions to change the data in the resources utilized by our application.

**Monitoring**

**==========**

aws resources used by our application can be monitored using the service availble with aws called Cloud Watch where u can set alarams and do trigger a topic in sns where we can have emails when ELB sense the critical cpu Utilization and any others.

**Storage**

**=======**

S3 can be used for storing data which is used by our application such as Images,Files which can be given acccess through IAM to Instance to access the S3 buckets.

**SQS**

**====**

It is a highly available and distributed message queuing system which makes data redundant on multiple availability zones.

**Oracle Database**

**==============**

It can be configured In available Relational Database Service,However RDS will provide all the Back up Retention Policy , automatic host replacement, and multi-AZ with synchronous replication for cross AZ redundancy useful for disaster recovery. Due to security concerns, it is a best practice to launch database instances in private subnets and configure its security Groups to allow traffic only from the application server Security Groups.

**Migration**

**========**

=> First we need to migrate the applications static files like images ,CSS,HTML Content to the S3 storage for low latency and high availability of Data.

=> Load Balancers policies has to configured with Aws ELB .

=>Elastic Beanstalk can be used to deploy code without having to manually provision and scale required resources.

=>If the application is deployed in multiple regions, Route 53 can be used to route traffic between regions based on routing policies.

=> we need to go through an testing of the Autoscaling and ELB initially until the Instances are recreated with all the same configuration we have set in the Image.

**Best Practice**

**==========**

=>IAM user credentials should never store in any of the Instance to provide security, and control over aws resources

=>Need to use the Cloudformation stack to programatically manage the infrastructure.

=> NAT gateways should to deployed in public subnets to provide internet to instances in private subnets.

=> Bastion hosts may be setup in the public subnets to connect to instances in private subnets from outside AWS.

=>Enable CloudWatch to monitor AWS resource usage and applications health and status.