Migration Plan, Strategy, & Execution Steps

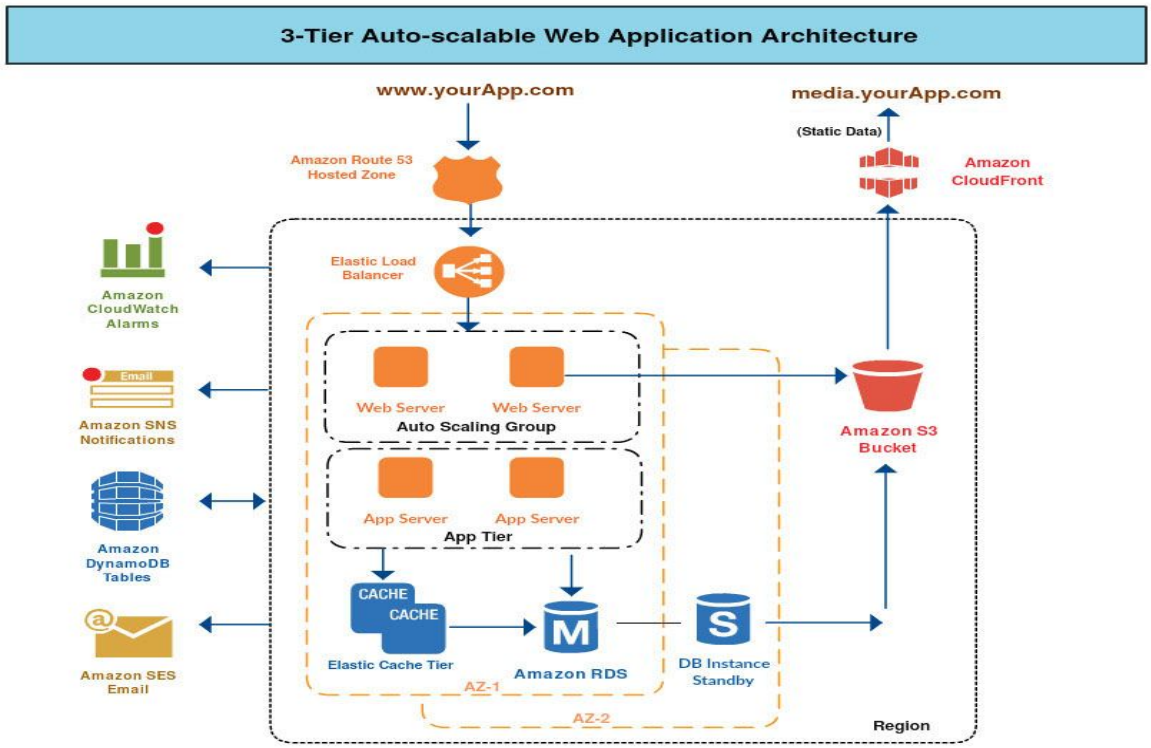
Cloud Assessment

Firstly, we would map the hardware configuration of physical servers to equivalent EC2 instance types and estimate the combined storage and bandwidth requirements. During this assessment we would gather all the facts about the resources procurement.

Designing an architecture

Application architecture plays a vital role. It starts with designing a network (VPC with subnets) followed by implementing all the required services like cloudwatch (monitoring), SNS (notifications), Cloudfront (CDN), RDS (Managed database service), Security groups, NACL, VPN tunneling, EC2 machines, Autoscaling, EIB, Route53, etc.

Designing all at once, can impact the migration. We need to do this in phases according to the migration cycle. We can consider the below architecture to be an ideal one, except services varying according to our use case.



Data Migration

- Before deploying our stack, we need to migrate the data to cloud which will include the below steps:
- Creating a secure VPN tunnel from source network to AWS VPC (virtual private cloud) so as to make the data migration over a private channel
- We will spin up the required EC2 machines (stand alone) along with the EBS (as per the data size)
- Will upload the static content like images, documents on S3
- For MongoDB migration will create a replica set (Secondary) of the existing database (data center) on AWS mongodb EC2 instance. Later, will switch it to primary while going live.
- For Oracle migration we will use the DMS (database migration service), in which we will define the source and destination database (RDS with oracle database). During the final migration we will take a downtime (to avoid any new transactions) and switch the database endpoints. For licencing either will opt for BYOL (if we have) or will get it from AWS.

Application Migration

- During the application migration phase, we will launch both small and large instances for the web and tomcat servers. We would copy the apache and the tomcat configuration files and would follow the same on these servers with few tweaks like server name, document root & environment variables
- ActiveMQ can be migrated with fresh installation on a dedicated server or we can also use SQS for the queues (if it meets our requirements)
- This phase may include modifying or extending our deployment jobs to point to our AWS infrastructure. For ex : In jenkins either we can clone the existing jobs or will change the primary ones. Migrating the CI/CD tool initially is not preferable as will keep our both infra running parallely for few weeks. Once will get all the testing done with this new infra thereafter we will migrate the jenkins
- As suggested above, we will use the route53 weighted policy to distribute the load across the 2 networks (50:50). To leverage this feature, we would require to do the DNS migration from X place to Route53. This includes exporting the zone files to Route53 & updating the NS records.

Post Migration

- Once will have the basic infrastructure ready, we will go with some advance techniques implementation like Autoscaling, configuration management tool (Ansible) setup, optimising the deployment lifecycle, Backup strategy, etc.
- As said above, while implementing Autoscaling we would require few stats about the application like which resource intensive is this? This will help us in setting up the

scaling policies. Also we will see if there is any specific trend in the traffic, according to which we can set up time based scaling

- Load balancing will be done using classic ELB. It will be configured against few parameters like healthcheck, Listeners, SG and health threshold.

NOTE: This strategy might change depending upon the complexity of the application.