Workload Simulation
using Amazon Web Services

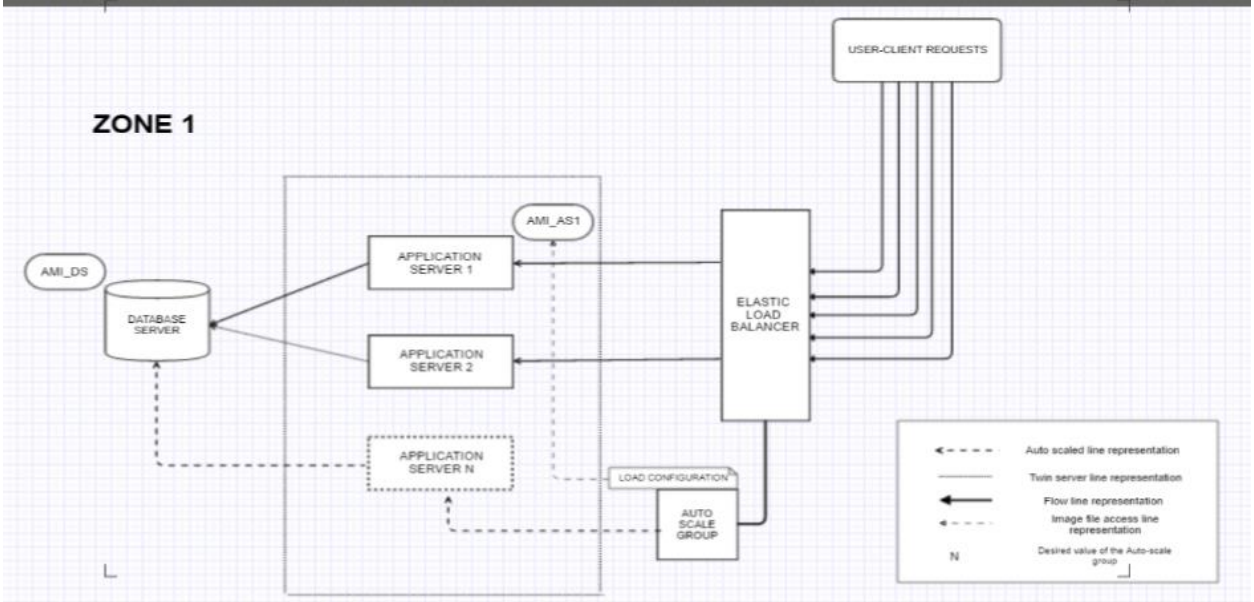
Swapnil Hete
Ujjwal Garad
Srimannarayana Potluri
Himaja Vadaga

- Objective
- Architecture
- Concept
- Design
- Challenges and Insights
- Disaster Recovery
- Cost and Performance

OBJECTIVE

- Develop, establish, deploy and test a scalable web application with high availability on the amazon EC2 cloud
- Design Business model which could provide justification and focus on ideal pricing by adequate cost and high performance
- Employ various cloud tools and solutions like ELB, DB MySQL, Auto Scaling, Cloud Watch to build a steady solution
- Configure key elements of the cloud architecture, study and retrospect the roadblocks, bottlenecks and limitations encountered during this process
- Develop a disaster recovery plan for ensuring business continuity

ARCHITECTURE



Description

° Elastic Load Balancer for instances to manage access to the application

° Autoscaling configuration to handle spikes/lows in usage

°Configuring and managing security groups for launching new instances

°AWS SSD Instance Store Volumes Proposal for Disaster Recovery

Concept

Load Balancing

What?

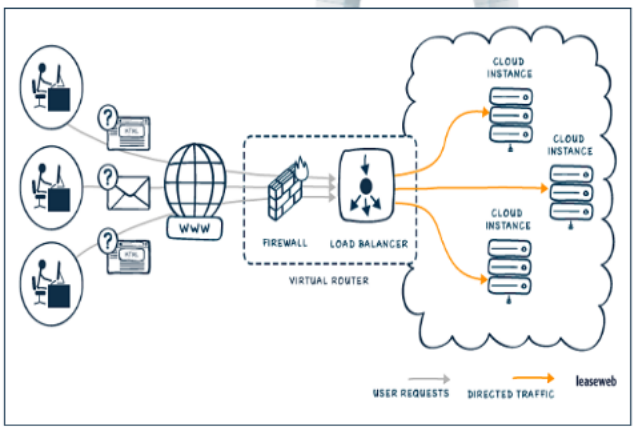
Elastic Load Balancing automatically distributes incoming traffic across multiple EC2 instances.

Why?

We can add/remove ec2 instances on the backend without hindering the user experience

Ease and speed of Scaling

Reliability



AUTO SCALING

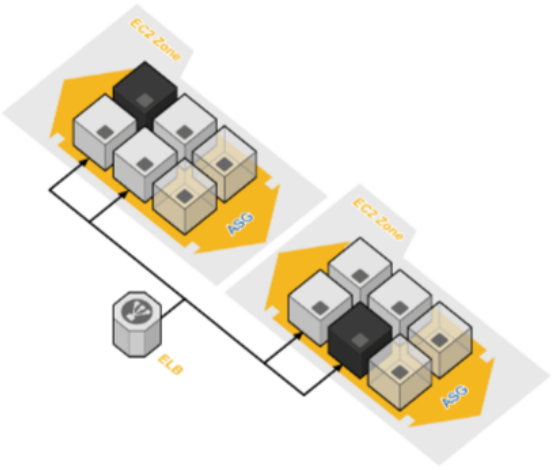
What?

Number 'N' of EC2 instances available to handle the load for your application

'N' can be Minimum, maximum and desired number of instances

Why?

- ensures that your application is getting the compute capacity that you expect.
- follow the demand curve for your applications closely, reducing the need to manually provision Amazon EC2 capacity in advance



SSL Security

Cloud Watch

What?

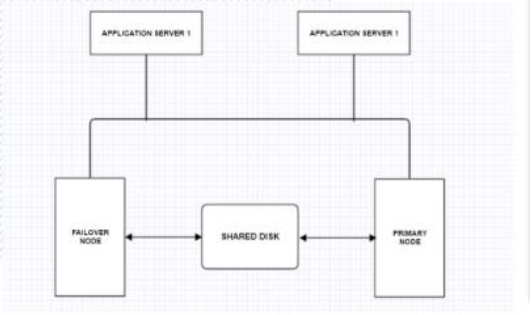
Monitoring service for AWS cloud resources and your applications

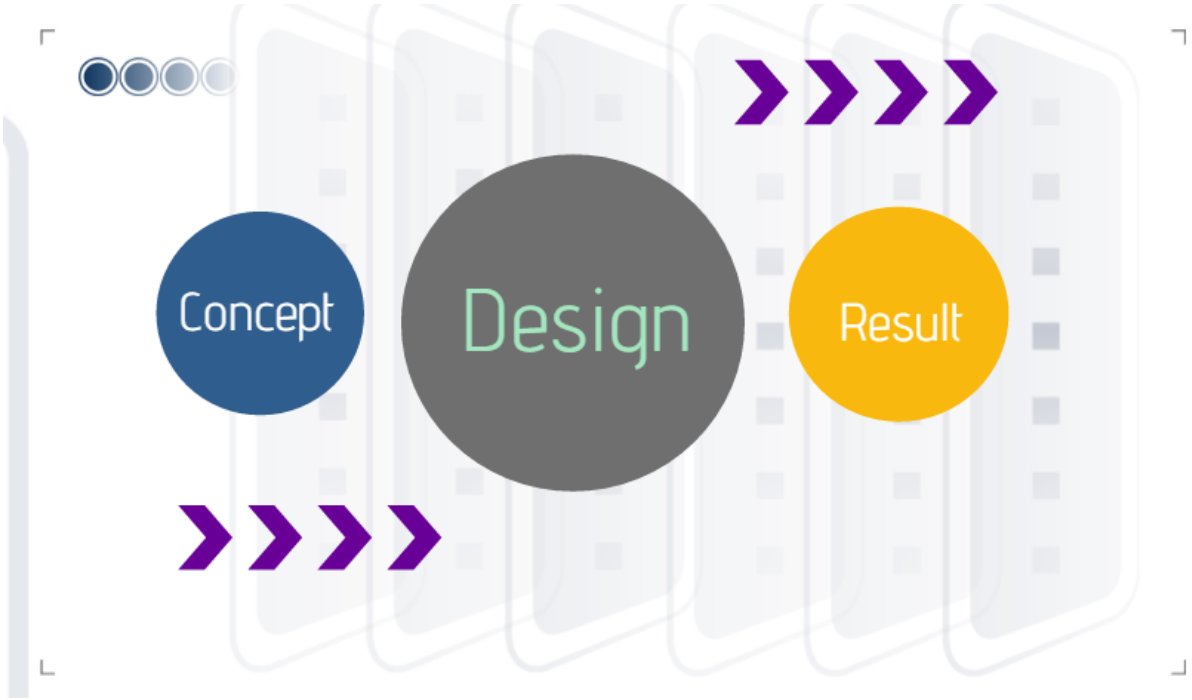
Why?

- Monitor Metrics: CPUUtilization, RequestCount, DiskReadBytes
- Analyze and retrospect using graphs and statistics
- Set alarms and react to resource changes
- Track AWS Resources and store logs

AWS SSD Instance Store Volumes: Disaster Recovery

- Provides very high I/O performance
- Database Mirroring
- Replication
- Log Shipping
- Backup and Restore




Concept
Design
Result

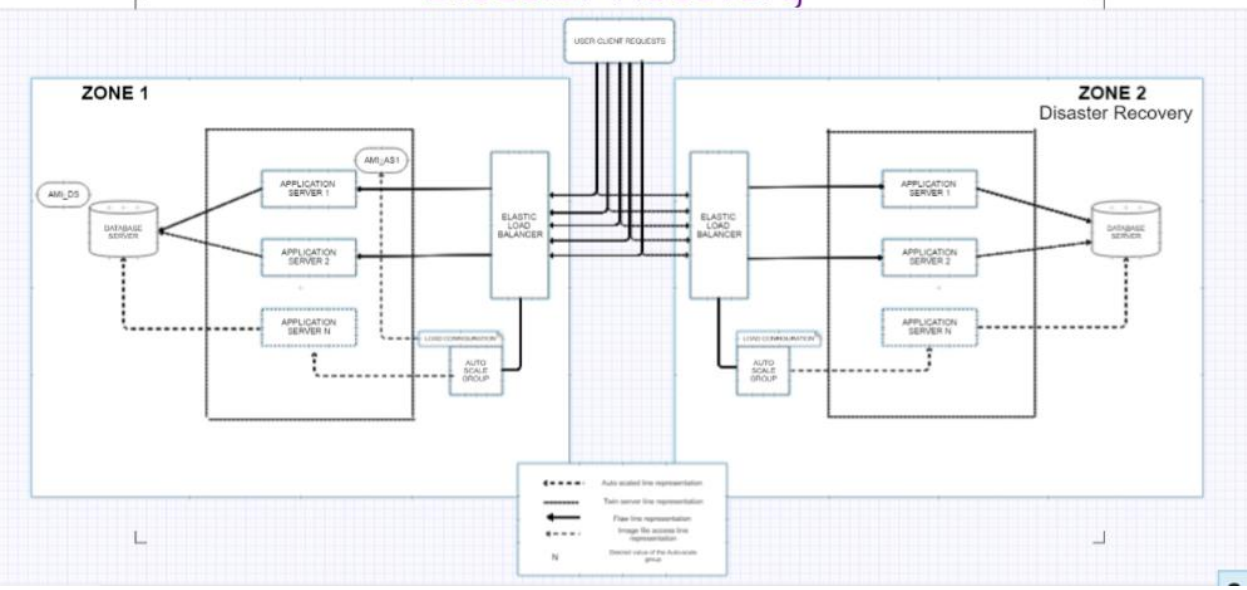
Design Configuration

Project Demo

Challenges and Insights

- AMI's: We wanted to have AMI's to bundle our web application stack so that launching new instances would be made simple. We figured out you can share AMI's across user accounts,this resulted in decreasing development time
- Elastic IPs': Our first approach was to use elastic ips' to make different instances talk to each other but we can't have Elastic IPs' assigned for load balancers
- http/https issues: Once we configured our application to work with https, all references to external libraries like bootstrap, jquery had to be changed to https to get the application working
- Security Group: Had to configure the instances to listen on multiple ports for http and https requests, so we created a custom security group for our AMI to launch our instances

Disaster Recovery



Cost vs Performance

Application estimated Cost per month: $55 ($54.86 exact)

Application Cost for 6 Months: $330($329.19)


Thanks.