AWS Foundation

INTRODUCTION TO ELASTIC LOAD BALANCER & AUTO SCALING

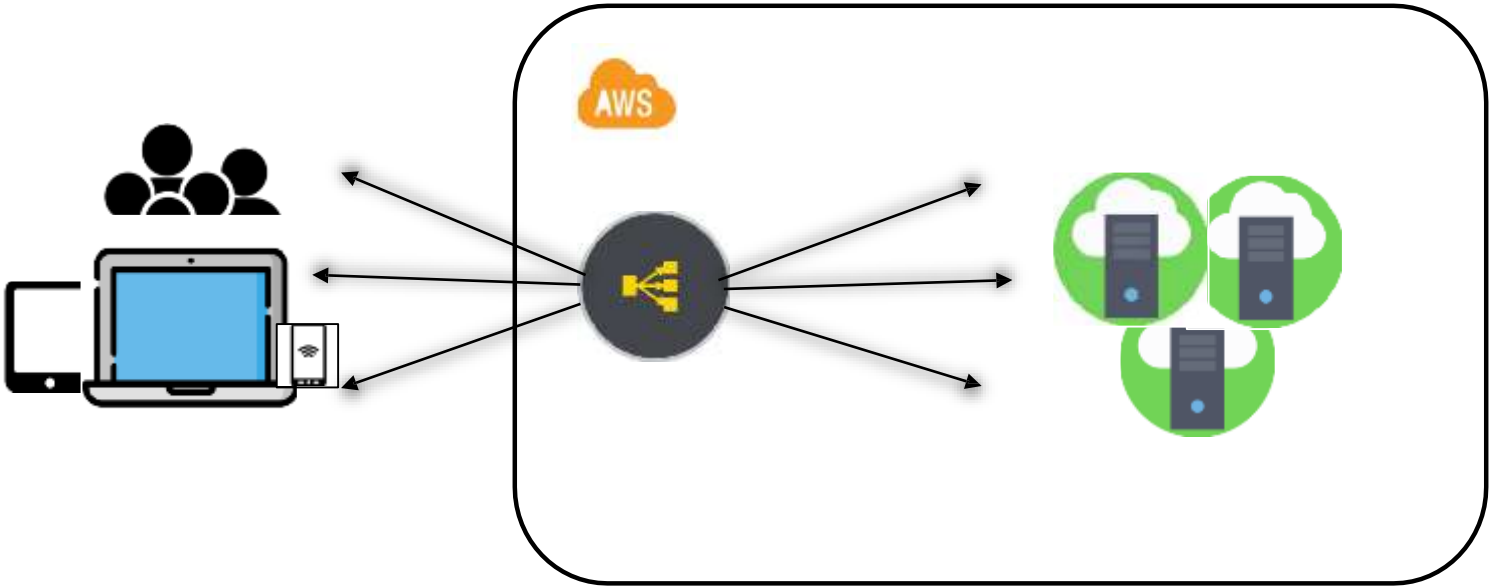
Agenda

1. Introduction to Elastic Load Balancer
2. Types of ELB: Classic, Network, Application & Gateway
3. Cross-zone Load Balancing
4. Introduction to Autoscaling
5. Vertical & Horizontal Scaling
6. Components of Autoscaling
7. Scaling Policy
8. Introduction to Route53
9. Routing Policy
10. Vertical & Horizontal Scaling

Introduction to ELB

Elastic Load Balancer

Load balancer is a service that uniformly distributes the network traffic and workloads across multiple servers or a cluster of servers. Load balancer increases the availability and fault tolerance of an application



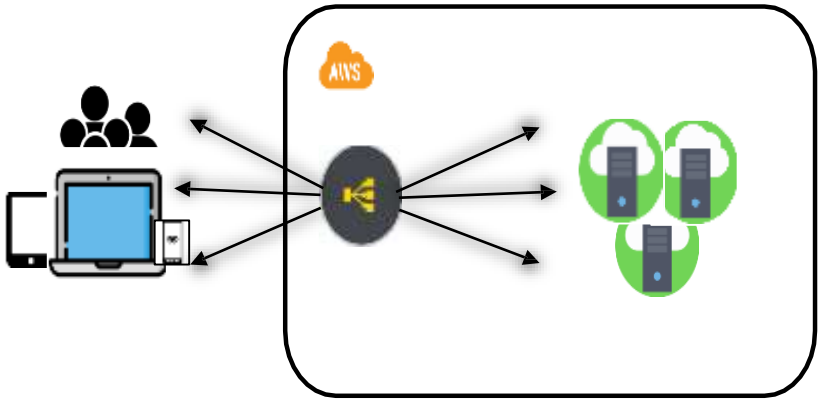
Elastic Load Balancer

Elastic Load Balancer (ELB) is a load balancing service for the AWS deployments

ELB scales itself as necessary to handle the load

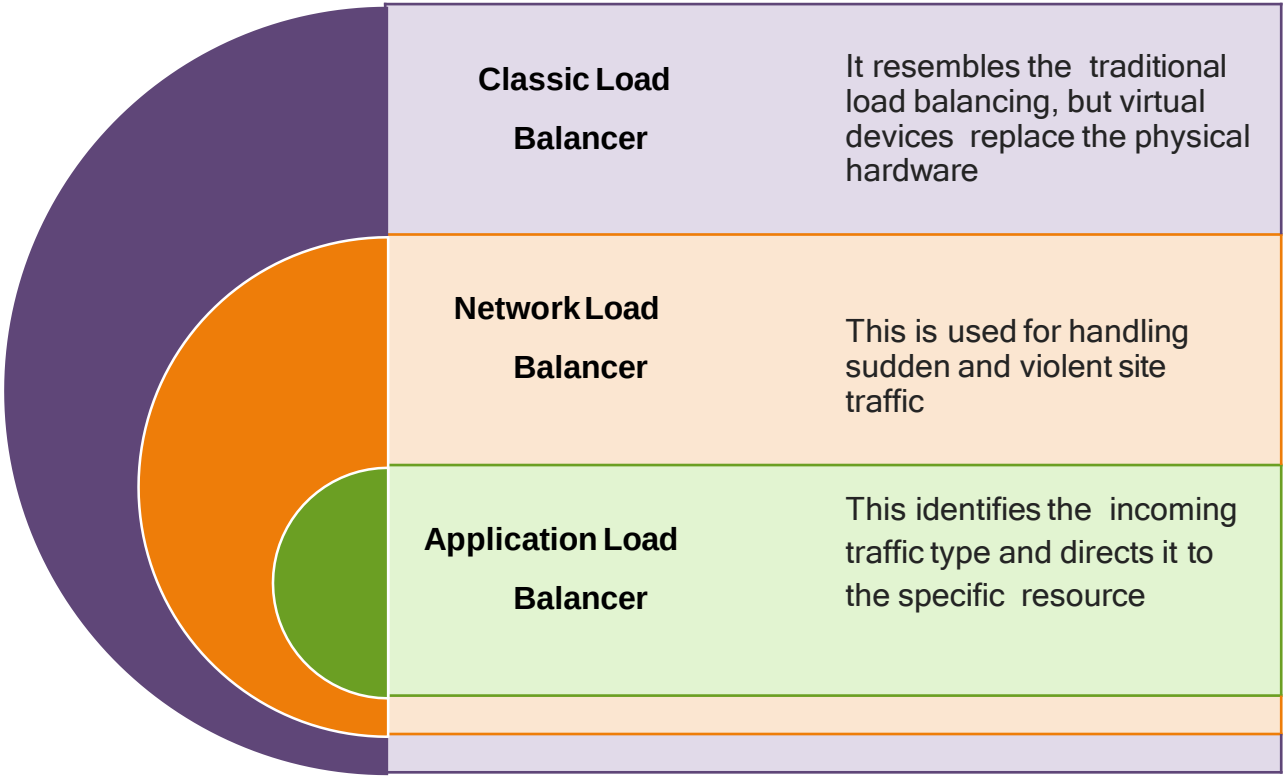
Incoming traffic is distributed across EC2 instances in multiple availability zones

Load balancer is the single point of contact for clients



Types of Elastic Load Balancer (ELB)

Types of Load Balancer

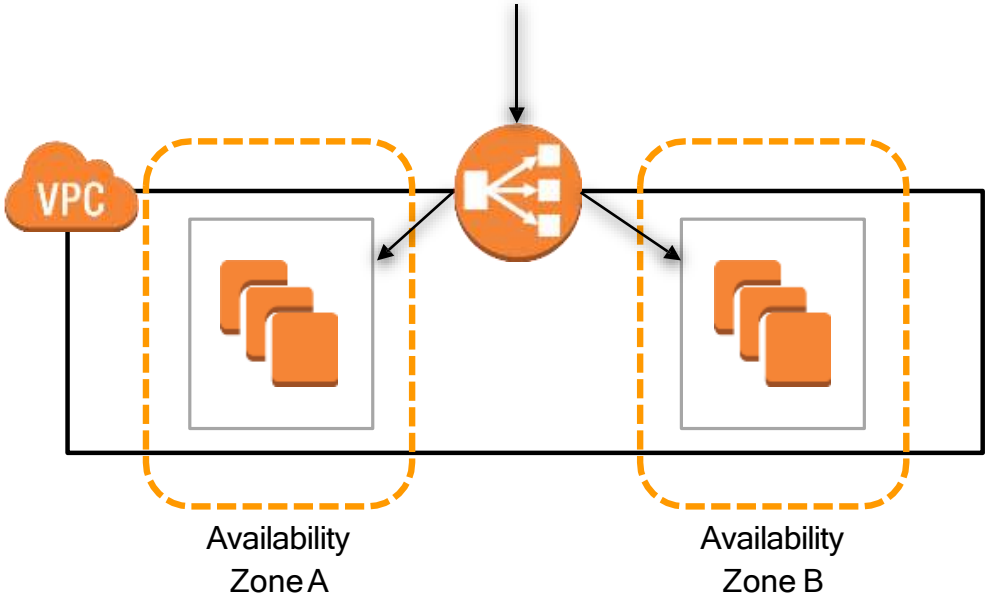


Classic Load Balancer

Distributes the incoming application traffic across EC2 instances in multiple AZs and functions at Layer 7 of the OSI model

Routes the traffic to healthy instances only, and it is evenly distributed

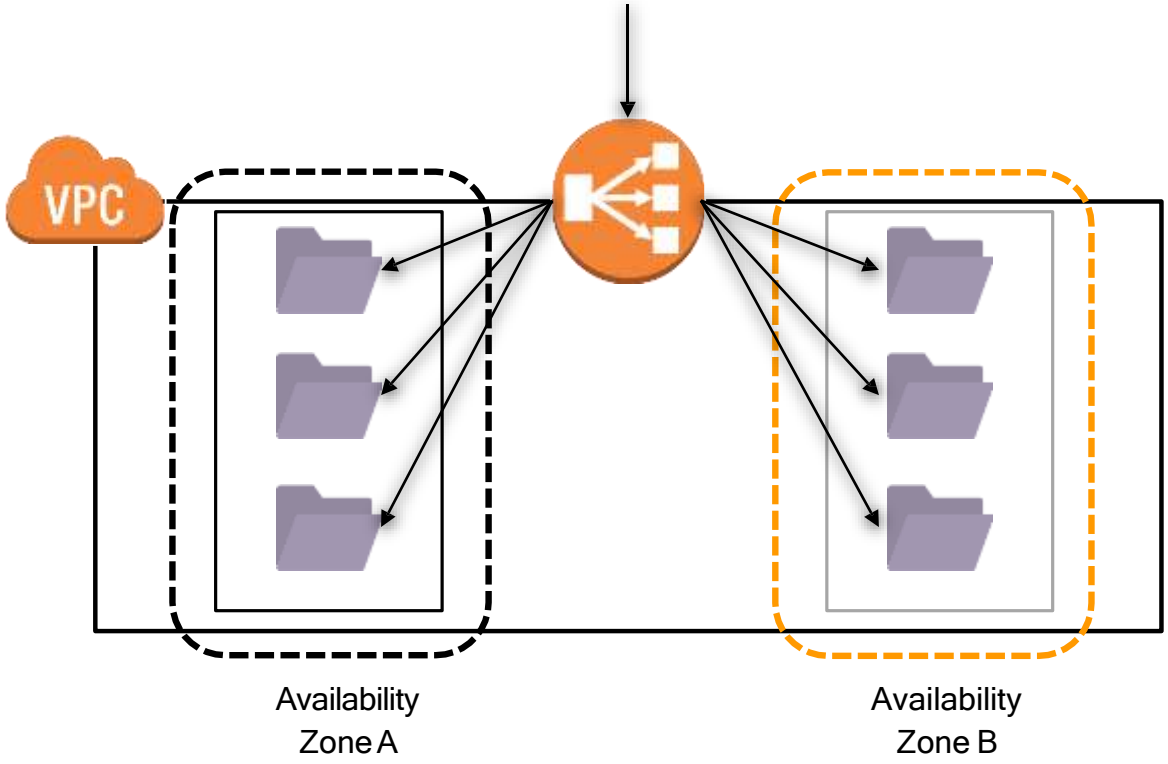
Internet and Internal-facing load balancer



Application Load Balancer

Functions at the 7th layer of the OSI model

Identifies the incoming resource type and directs it to the appropriate server

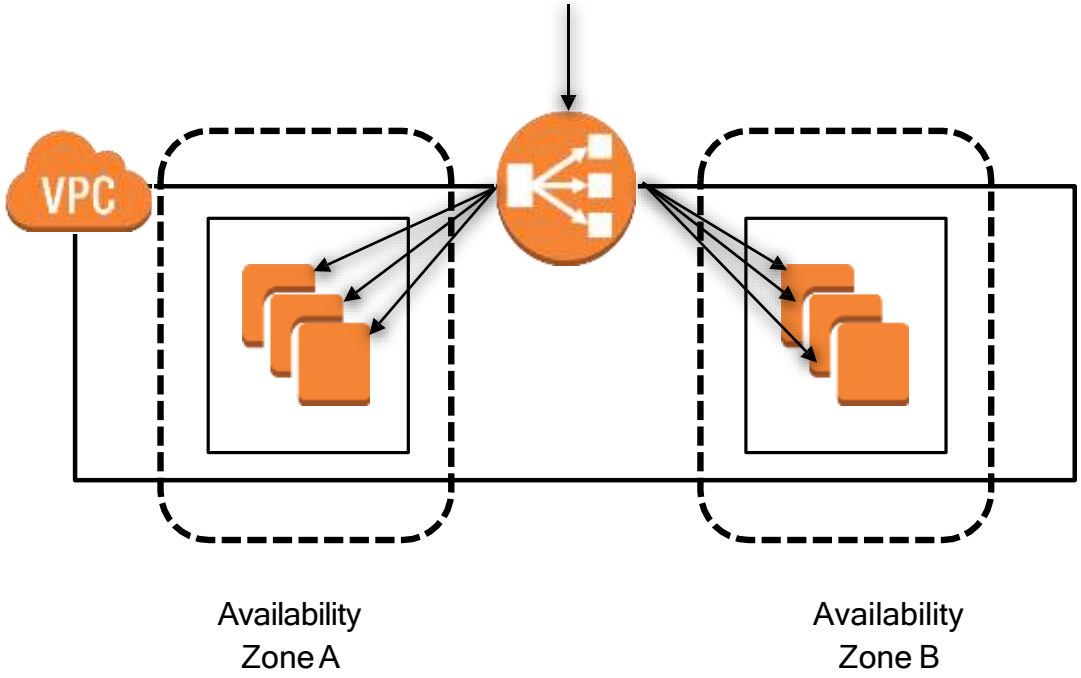


Network Load Balancer

Functions at the 4th layer of the OSI model

Handles millions of requests per second and maintains low latency

Ideal for load balancing the TCP traffic and supports elastic or static IP



IntelliPaat

Gateway Load Balancing

Gateway Load Balancer

Gateway Load Balancing automatically distributes your incoming traffic across multiple targets in one or more Availability Zones, such as EC2 instances, containers, and IP addresses. It monitors the health of its registered targets and routes traffic only to those that are in good condition.

Gateway Load Balancer

Benefits of Gateway Load Balancer

- Deploy third-party virtual appliances more quickly
- Scale virtual appliances while keeping costs in check
- Increase the availability of your third-party virtual appliances & continuously monitor health and performance metrics.
- Using Gateway Load Balancer Endpoints, ensure private connectivity across the AWS network.

IntelliPaat

Steps for Creating Gateway Load Balancing

Steps to Create Gateway Load Balancer

Creating Gateway Load Balancer

- Load Balancers can be found in the navigation pane under Load Balancing.
- Select Create Load Balancer.
- Select Create under Gateway Load Balancer.
- Enter a name for your load balancer in the Load balancer name field. For instance, my-glb.

Steps to Create Gateway Load Balancer

Creating a Gateway Load Balancer

- Because your clients can only communicate with the load balancer using IPv4 addresses, you must select IPv4 as the IP address type.
- Select the service provider VPC for VPC.
- Select all of the Availability Zones in which you launched security appliance instances, as well as the corresponding public subnets, for Mappings.
- Choose a target group to which traffic should be routed as the default action. Create a target group first if you don't already have one. The GENEVE protocol must be used by the target group.
- Review your configuration, and then choose Create load balancer.

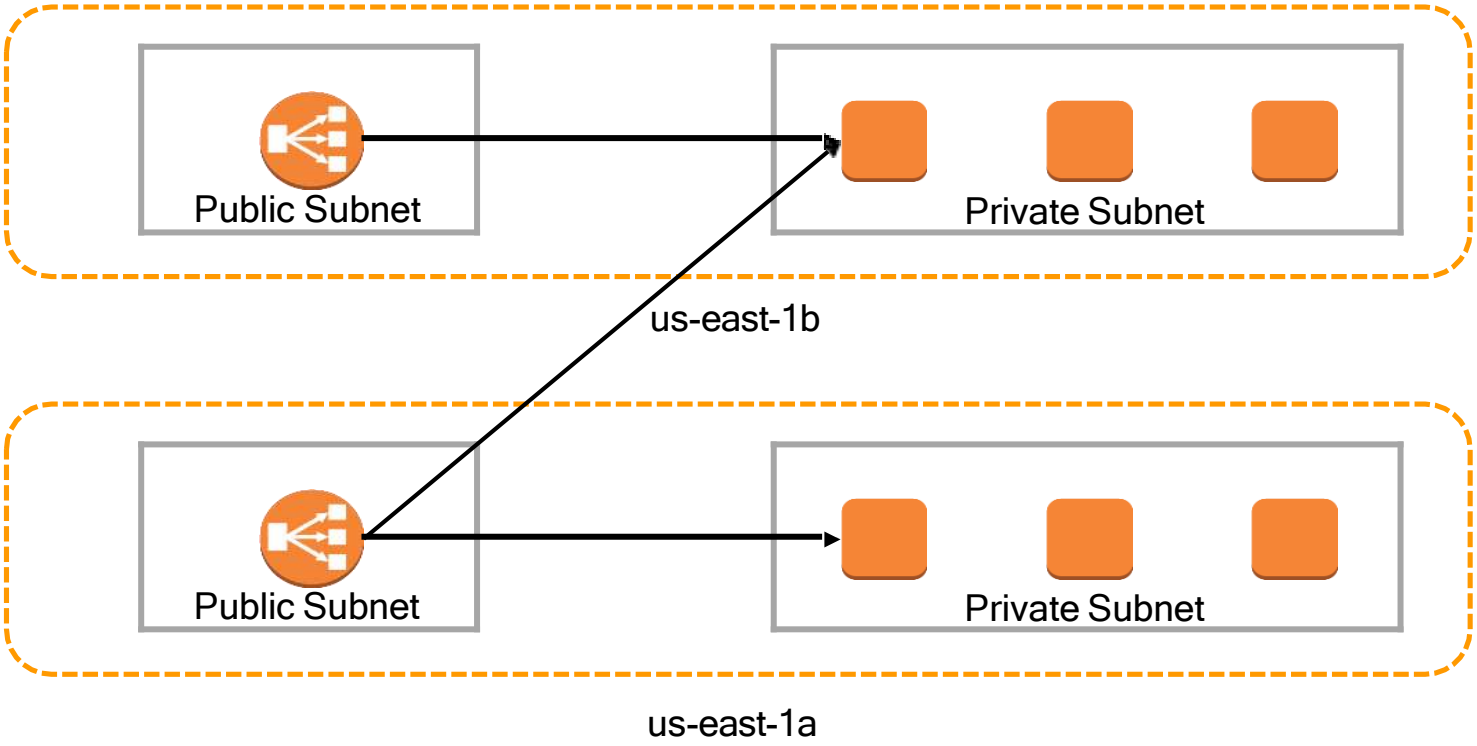
Steps to Create Gateway Load Balancer

Steps to Create :

- Because your clients can only communicate with the load balancer using IPv4 addresses, you must select IPv4 as the IP address type.
- Select the service provider VPC for VPC.
- Select all of the Availability Zones in which you launched security appliance instances, as well as the corresponding public subnets, for Mappings.
- Choose a target group to which traffic should be routed as the default action. Create a target group first if you don't already have one. The GENEVE protocol must be used by the target group.
- Review your configuration, and then choose Create load balancer.

Load Balancer Architecture

Load Balancer Architecture



Blue/Green Deployment using Weighted Routing

Weighted Routing

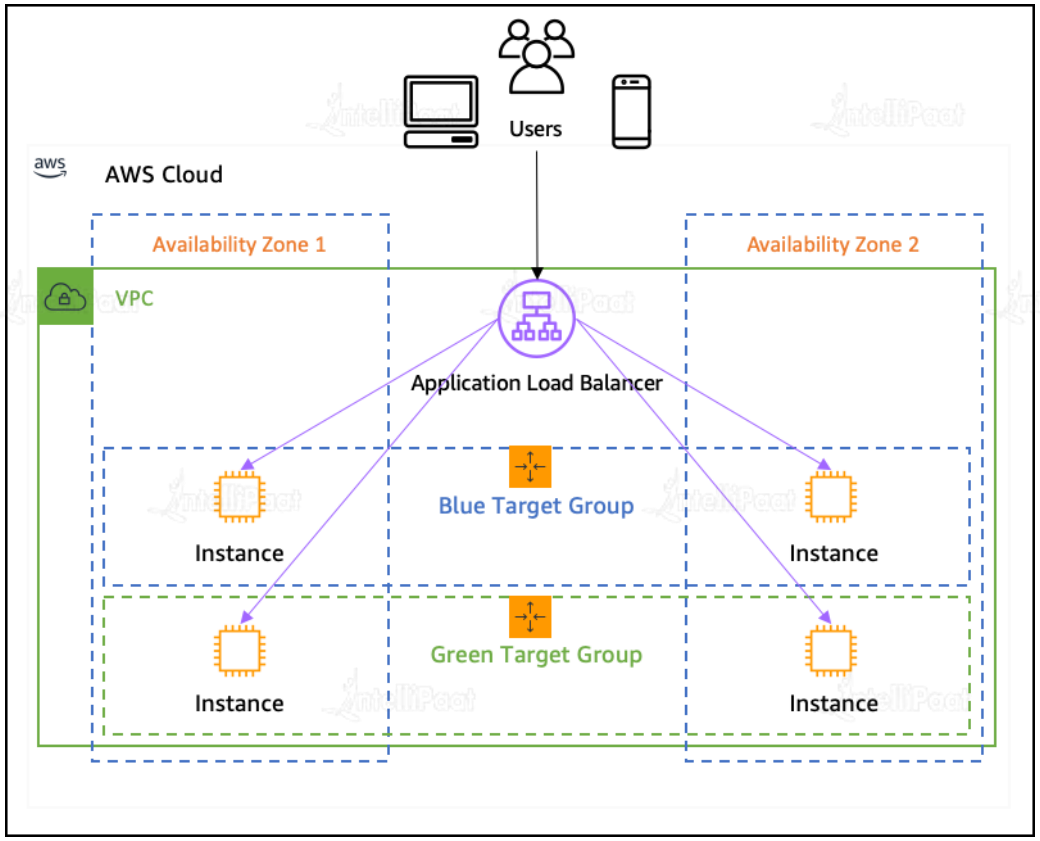
With weighted routing, we can switch the traffic between the versions of our application. This configuration allows us to control the distribution of the traffic to our application

Old App

New App

For example, if we define a rule having two target groups with weights 8 and 2, the load balancer will route 80% of the traffic to the first target group and 20% to the other

Weighted Routing



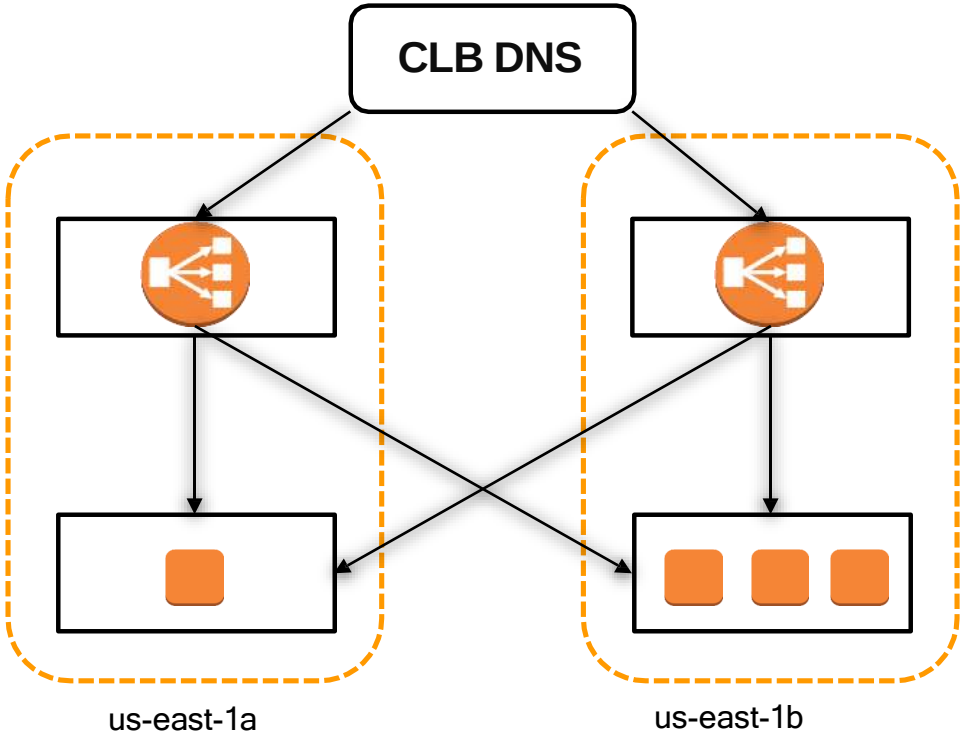
Cross Zone Load Balancing

Cross Zone Load Balancing

By default, CLB nodes distribute the traffic to instances in their availability zone only

We enable cross-zone load balancing to route evenly across EC2 instances

CLB routes each request to the instance with the smallest load



IntelliPaat

Connection Draining

Connection Draining

By enabling this feature, you will be able to better manage the resources behind the Elastic Load Balancer, such as replacing backend instances without disrupting the user experience. Taking an instance out of service and replacing it with a new EC2 instance with updated software, for example, while avoiding breaking open network connections.

Connection Draining

Steps for Enabling Connection Draining using AWS console

- Access the AWS Management Console.
- Go to the EC2 dashboard.
- Click Load Balancers in the navigation panel under Load Balancing.
- Choose an Elastic Load Balancer.
- From the bottom panel, select the Instances tab.
- Verify whether the Connection Draining status is enabled:

Connection Draining

Advantages of Connection Draining using AWS console

- Access the AWS Management Console.
- Go to the EC2 dashboard.
- Click Load Balancers in the navigation panel under Load Balancing.
- Choose an Elastic Load Balancer.
- From the bottom panel, select the Instances tab.
- Verify whether the Connection Draining status is enabled:

IntelliPaat

Sticky Sessions

Sticky Sessions

Sticky session also known as session persistence, refers to the process by which a load balancer establishes an affinity between a client and a specific network server for the duration of a session (i.e., the time a specific IP spends on a website). Sticky sessions can help to improve user experience while also optimizing network resource usage.

Sticky Sessions

A load balancer assigns an identifying attribute to a user with sticky sessions, typically by issuing a cookie or tracking their IP details. Then, based on the tracking ID, a load balancer can begin routing all of this user's requests to a specific server for the duration of the session.

Sticky Sessions

Advantages for using sticky sessions

- Reduced data exchange - When using sticky sessions, servers in your network do not need to exchange session data, which is an expensive process when done on a large scale.

- RAM cache utilization - Sticky sessions enable more efficient use of your application's RAM cache, resulting in improved responsiveness.

Sticky Sessions

Steps for using sticky sessions

To enable duration-based sticky sessions for a load balancer using the console

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under Load Balancing, choose Load Balancers.
3. Select your load balancer.
4. On the Description tab, choose Edit stickiness.
5. On the Edit stickiness page, select Enable load balancer generated cookie stickiness.

IntelliPaat

Types of Cookie based Sticky Session Persistence

Cookie Based Session Persistence

This is known as cookie-based persistence. Instead of relying on the SSL/TLS session ID, the load balancer would insert a cookie to uniquely identify the session the first time a client viewed the site, and then refer to that cookie in subsequent requests to maintain the connection to the proper server.

Duration Based Session Persistence

Your load balancer sends out a cookie that specifies the period for session stickiness. When the load balancer gets a client request, it checks for the presence of this cookie. The session is no longer sticky once the stated time period has passed and the cookie has expired.

Application Based Session Persistence

Application-based stickiness allows you to define your own client-target stickiness criteria. When application-based stickiness is enabled, the load balancer directs the initial request to a target inside the target group based on the algorithm selected. To enable stickiness, the target is required to set a custom application cookie that matches the cookie setup on the load balancer.

IntelliPaat

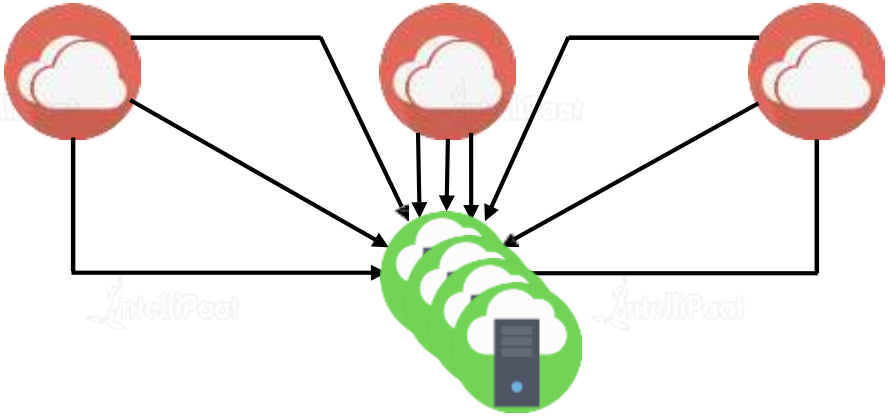
What is Autoscaling ?

What is Auto-Scaling ?

AWS Auto Scaling monitors your applications and adjusts capacity automatically to ensure consistent, predictable performance at the lowest possible cost. It is simple to set up application scaling for multiple resources across multiple services using AWS Auto Scaling in minutes.

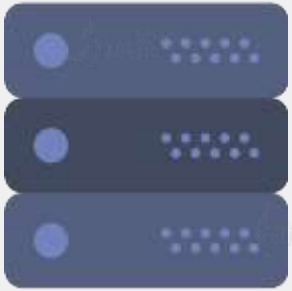
What is Auto-Scaling ?

- Scaling is the process of adding/removing capacity/resources as needed
- Scale out is adding the capacity/resources
- Scale in is removing the capacity/resources
- Types: Vertical and Horizontal



What is Auto-Scaling ?

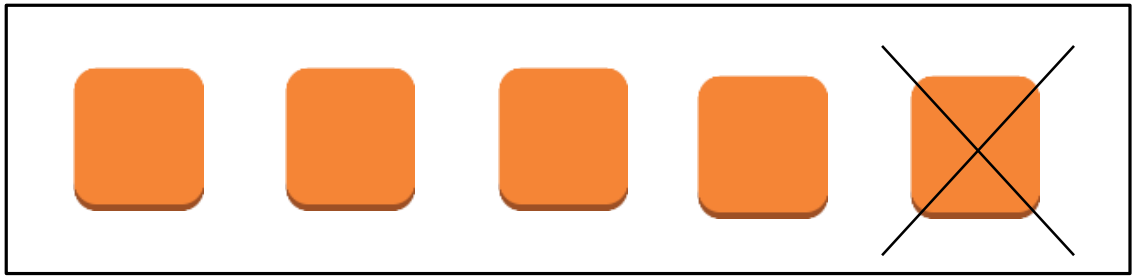
- Scaling Types: Vertical and Horizontal
- Vertical:
  - CPU: 2.0 GHz to 3.2 GHz
  - RAM: 1024 GB to 2048 GB
  - N/W Bandwidth: 4 Gbps to 10 Gbps



- Horizontal:
  - CPU: 1 server with 1.0 GHz to 3 servers with 1.0 GHz
  - RAM: 1 server with 500 GB to 3 servers with 500 GB

What is Auto-Scaling ?

- Autoscaling is scaling out/in automatically without any manual intervention
- It helps ensure that the correct number of EC2 instances are available to handle the load
- Multi-AZ EC2 instances provide high availability solutions

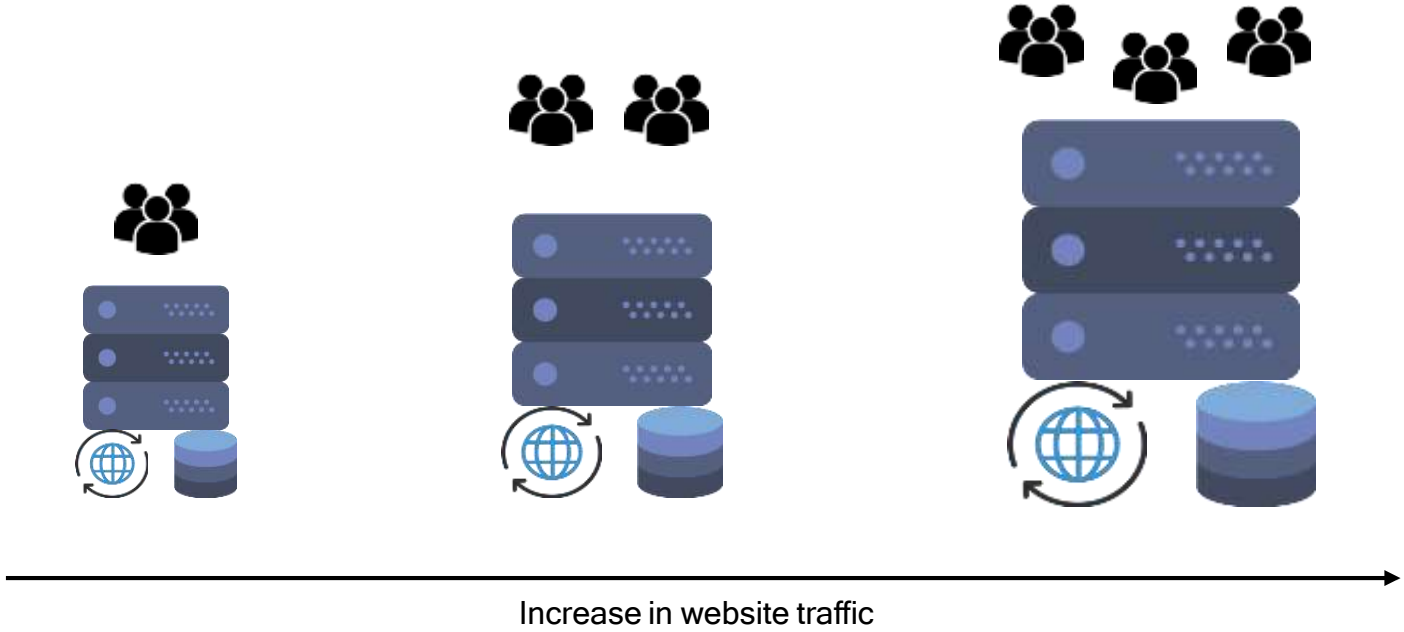


- Autoscaling can dynamically increase or decrease capacity as needed

Vertical & Horizontal Scaling

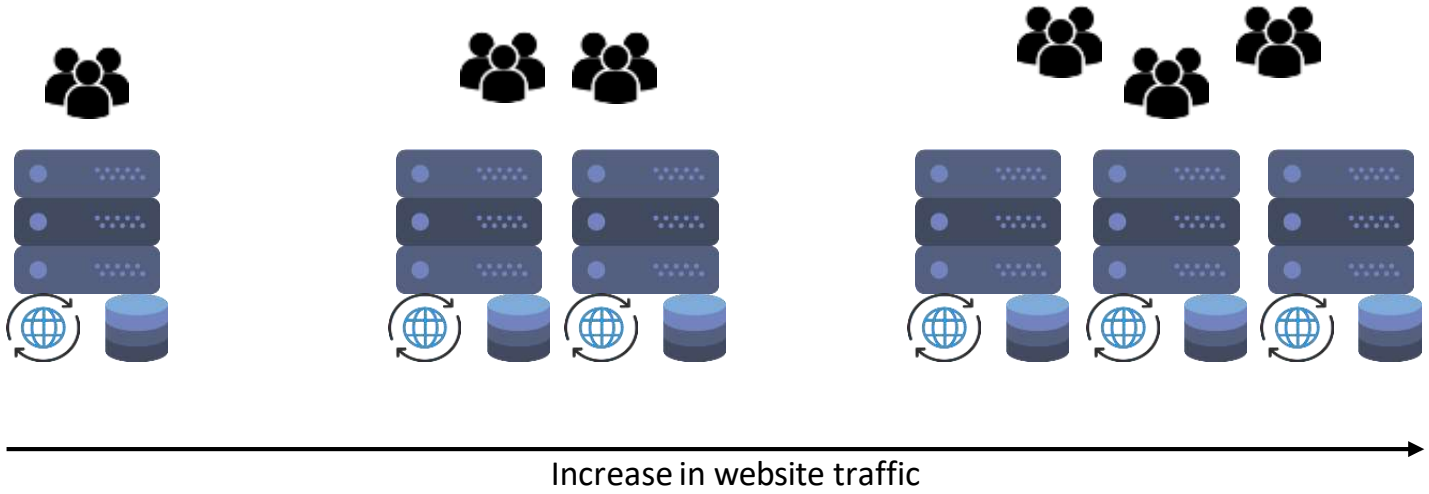
Vertical & Horizontal Scaling

Vertical Scaling



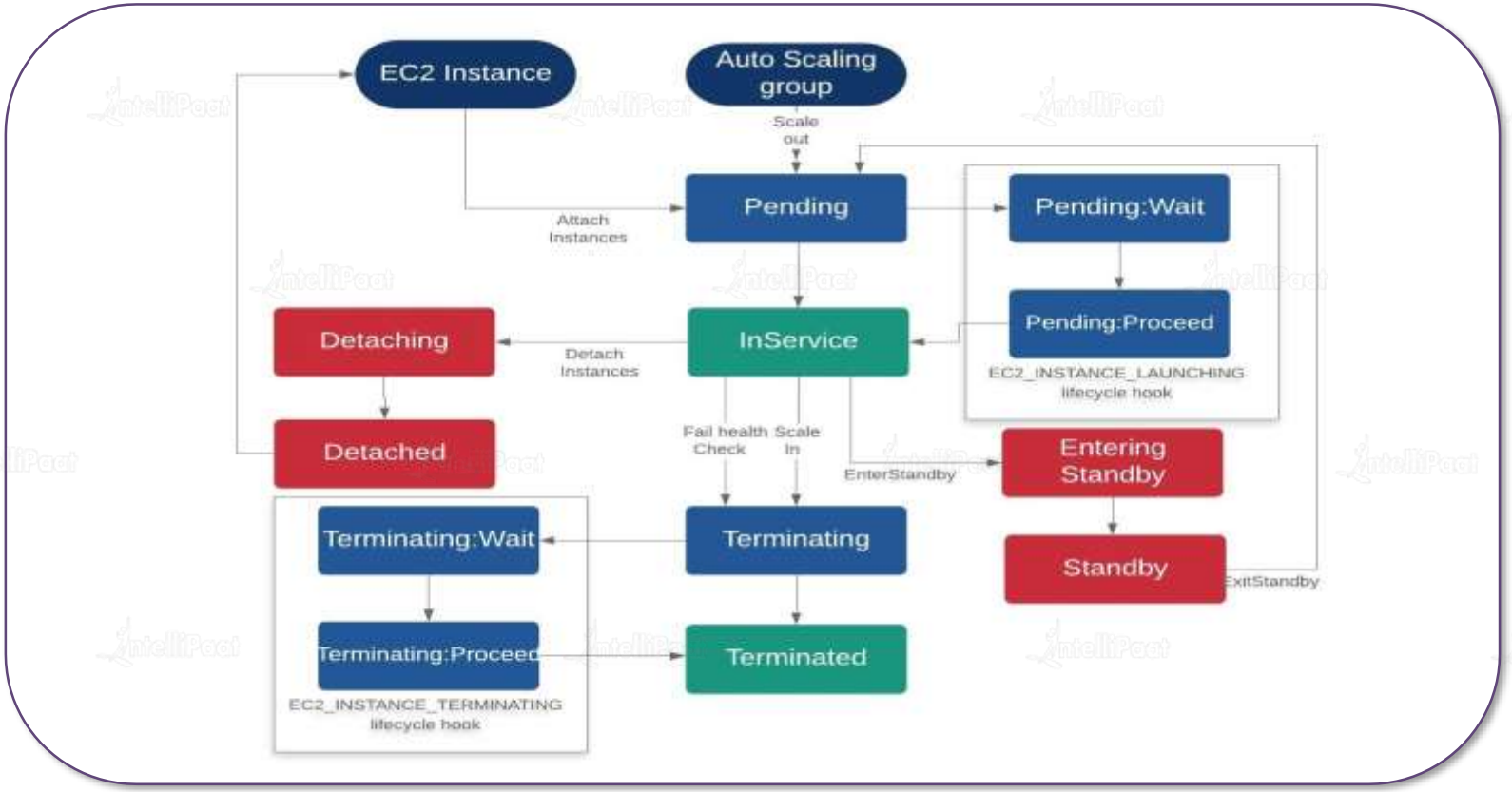
Vertical & Horizontal Scaling

Horizontal Scaling



Lifecyle of Autoscaling

Lifecycle of Autoscaling



Components of Autoscaling

Auto-Scaling Components

Groups

EC2 instances are in groups so that they can be considered as a logical unit (for scaling and management)

When we create a group, we can mention the following attributes: The max, min, and desired number of instances

Configuration Templates

These are used as configuration templates for the EC2 Instances

Launch template or Launch configuration is also used

Scaling Options

Autoscaling provides several ways to scale the group

- Manual scaling
- Dynamic scaling
- Scaling based on demand or schedule

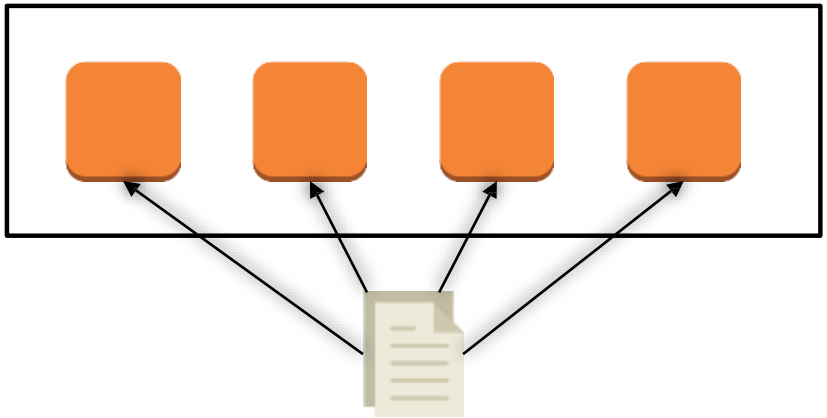
Auto-Scaling Groups

An autoscaling group contains a collection of EC2 instances that are exactly the same

While creating an autoscaling group, the launch configuration must be specified After specifying, the launch configuration cannot be changed

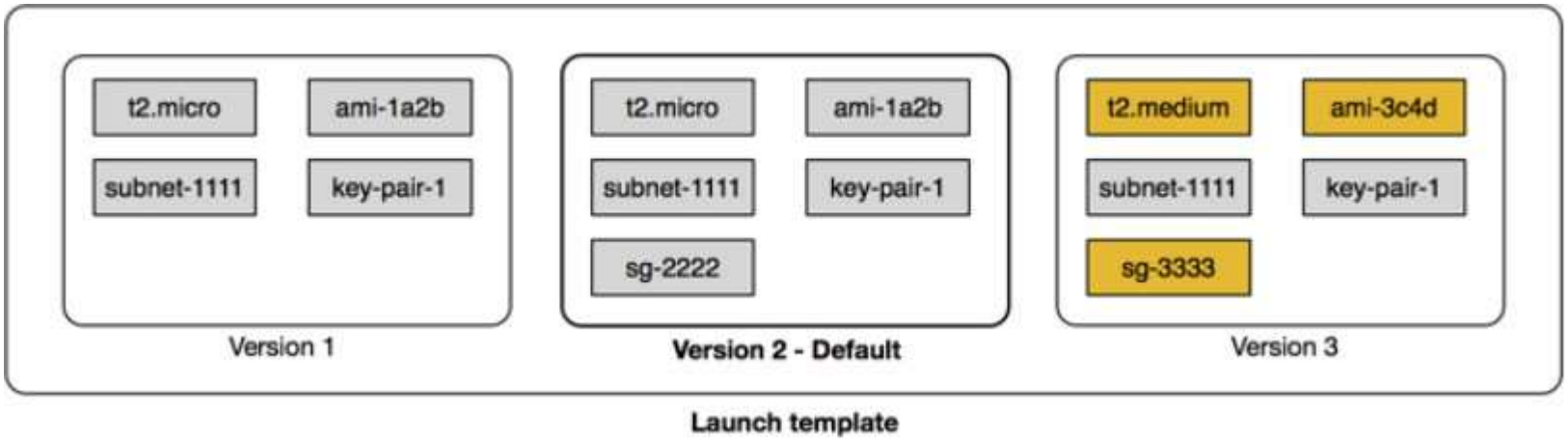
New instances are launched using a new configuration

EC2 instances are launched and terminated using scaling policies



Configuration Templates

Launch template can also be used with autoscaling groups



Configuration Templates

Launch configuration is a template that is used to launch EC2 instances for the autoscaling purpose

Autoscaling groups (the next topic) use launch configuration to launch instances

Launch configuration cannot be modified after creation

It can be created in two ways:

- From scratch: Image ID, instance type, storage devices, etc.
- From an EC2 instance: Attributes from the instance are copied. Block device mapping of the AMI is included. Any additional device that was attached after launching the instances is not considered in the launch configuration

Scaling Options

Scaling policies and alarms:

Scaling policies mention how to scale, and alarms decide when to scale

CloudWatch alarms are set to monitor individual metrics, e.g., CPU utilization, etc.

When the threshold is breached, scaling policies are executed

Minimum, maximum, and the desired capacity

Scaling Policy:

- Increase 2 instances at a time
- Decrease 1 instance at a time

Alarm:

If CPU utilization > 80% for more than 10 mins, ring the bell

- Minimum capacity: 2
- Desired capacity: 4
- Maximum capacity: 10

Scaling Options

Scaling based on a schedule: This type of a scaling method is used to scale at a given time and date

Scaling based on demand: Here, scaling occurs when the CPU utilization of the current running instances grows beyond a fixed usage limit

Scaling based on a schedule:

- Increase the instances by 2 at 2:30 pm today
- Decrease the instances by 1 at 12:00 am tomorrow

Scaling based on demand:

- If CPU utilization > 80% for more than 10 mins, increase the instance by 1
- If CPU utilization < 50% for more than 5 mins, decrease the instances by 2

Scaling Policy

Scaling Policy

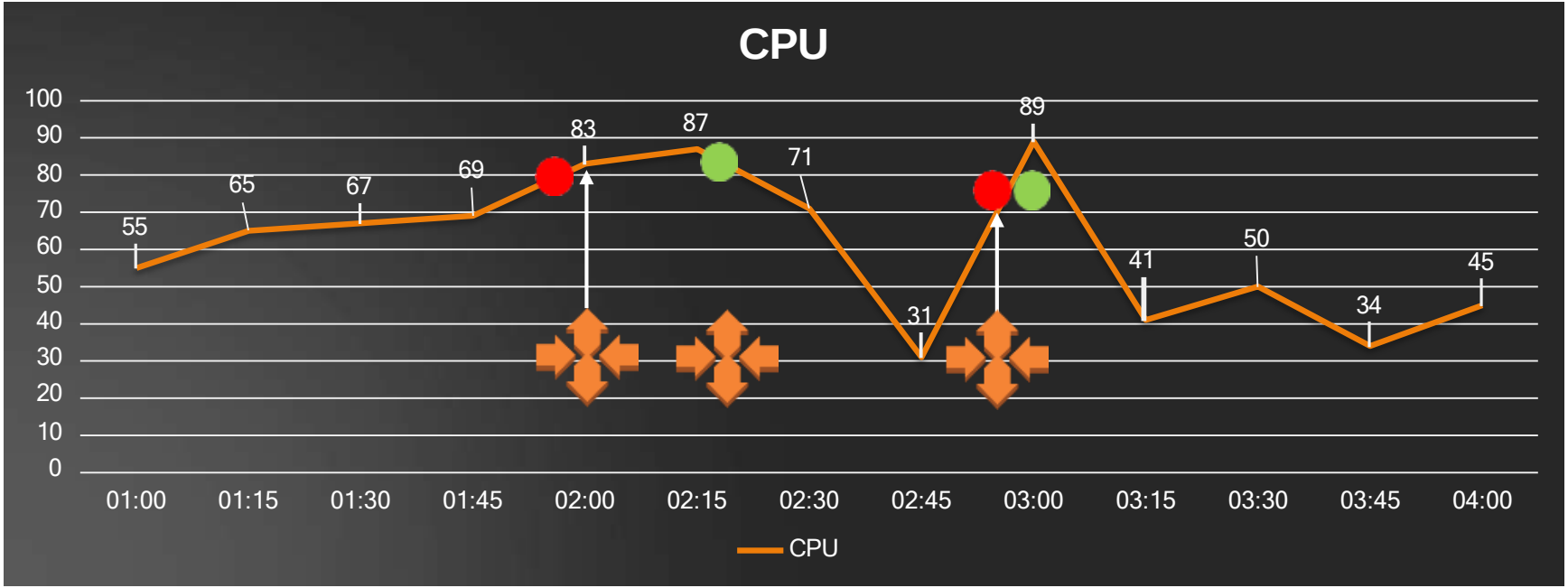
Scaling Policy:

- Increase 2 instances at a time
- Decrease 1 instance at a time

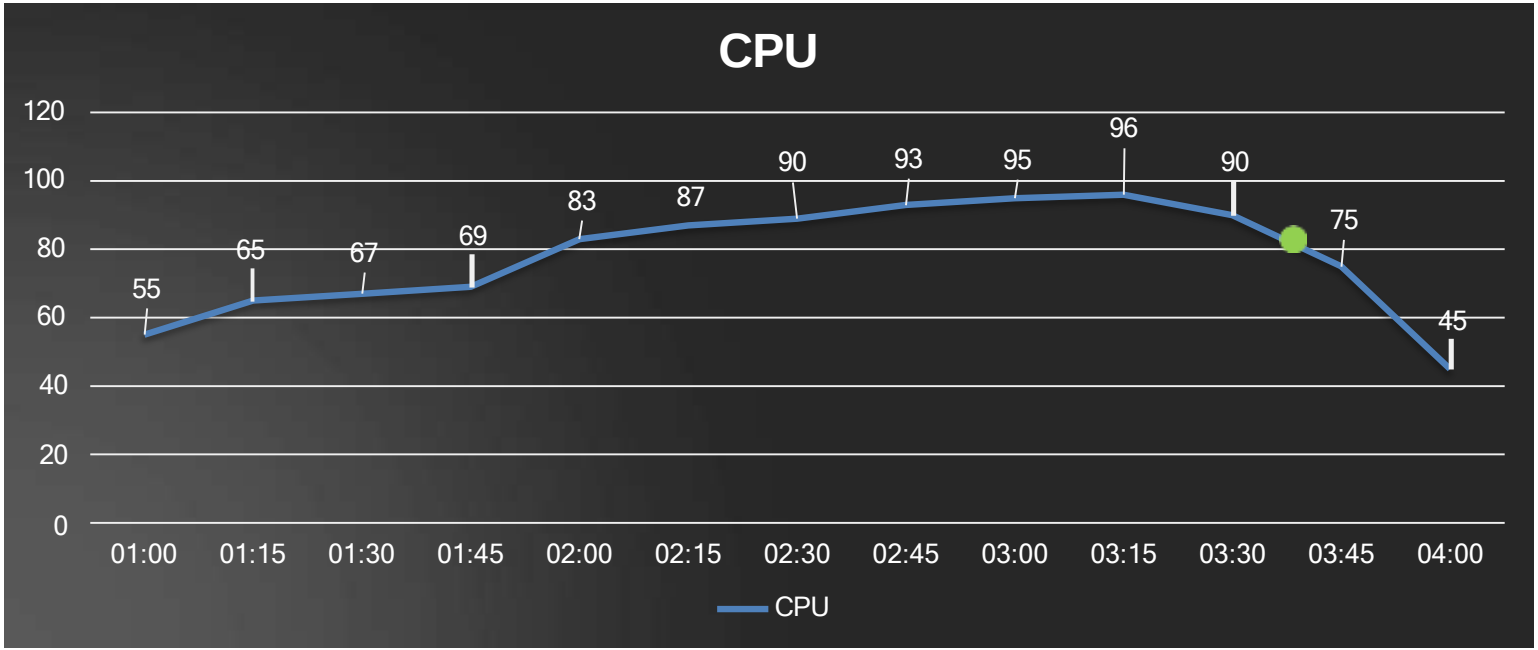
Alarm:

If CPU utilization > 80% for more than 10 mins, ring the bell

- Minimum capacity: 2
- Desired capacity: 4
- Maximum capacity: 10



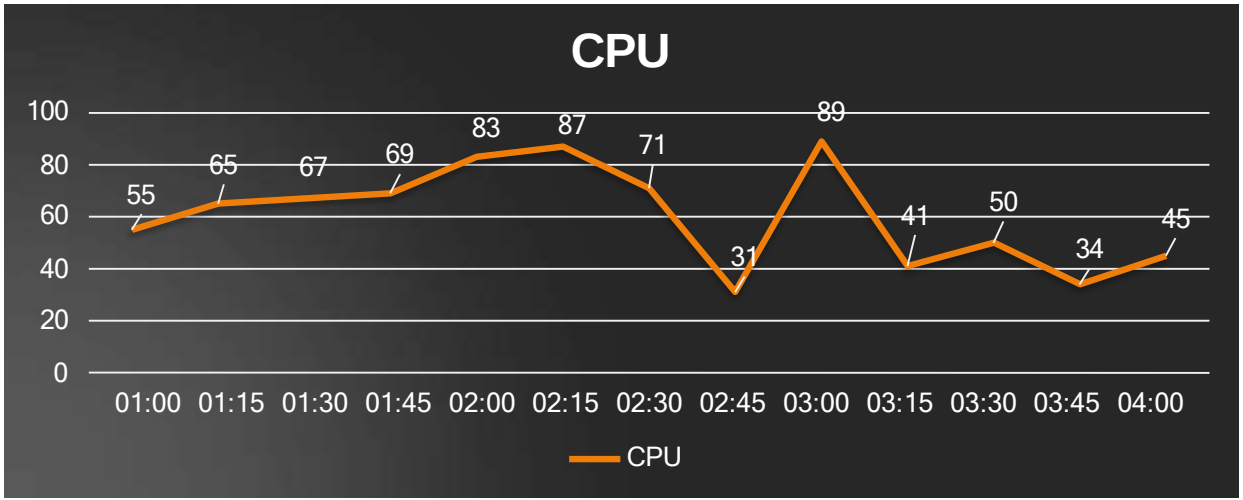
Scaling Policy



Scaling Policy

Cool-down Period: Simple Scaling Policy

The cool-down period ensures that autoscaling does not launch or terminate any more instances until a specified period is completed. Scaling activity is suspended until the cool-down period is in effect

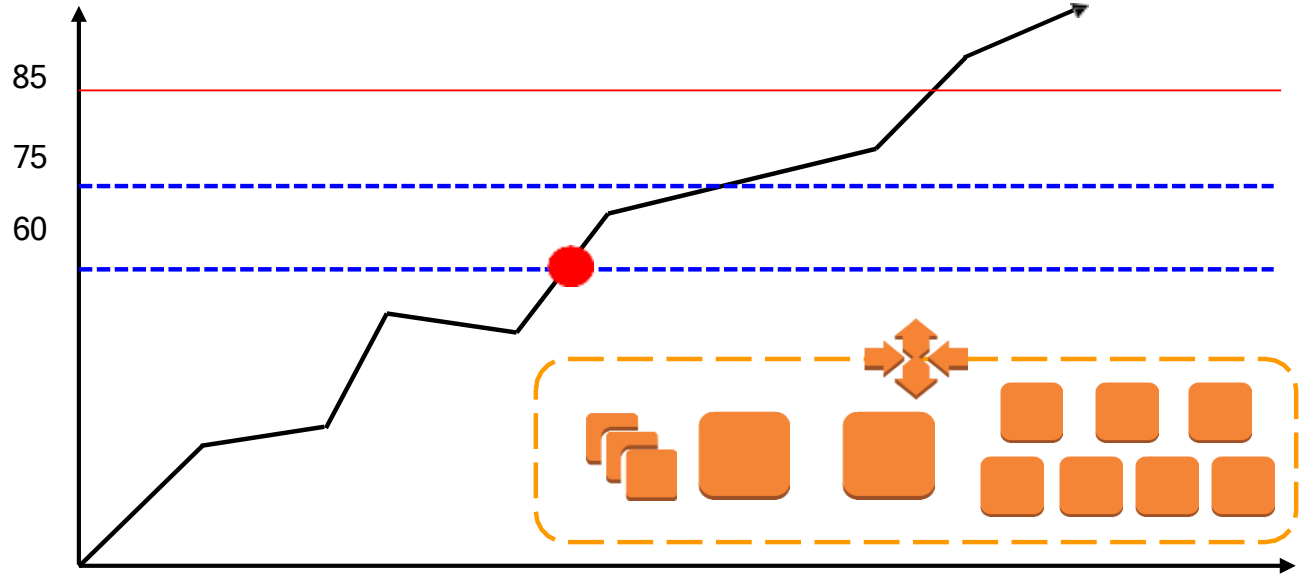


Scaling Policy

Step Scaling

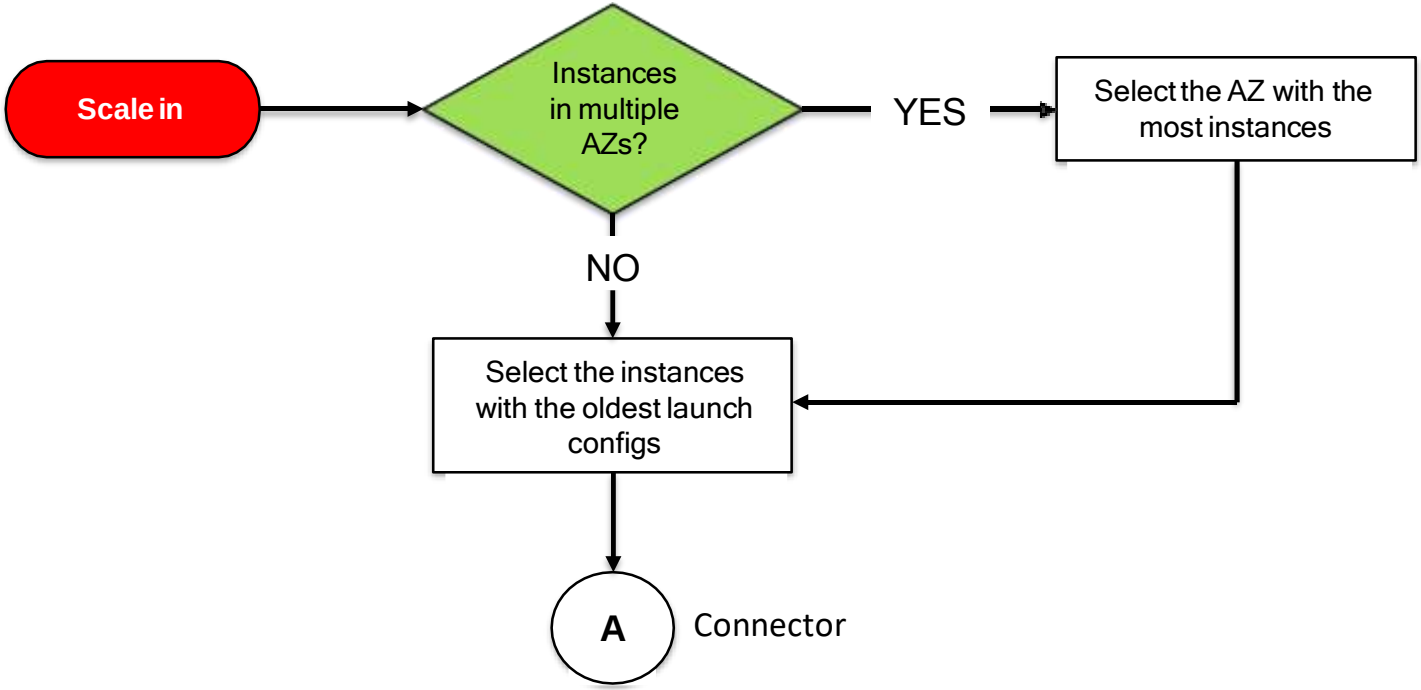
- **Alarm: CPU > 60%**
- **Action: Add 2 Instances**

CPU	Add
> 60%	2
> 75%	3
> 85%	4

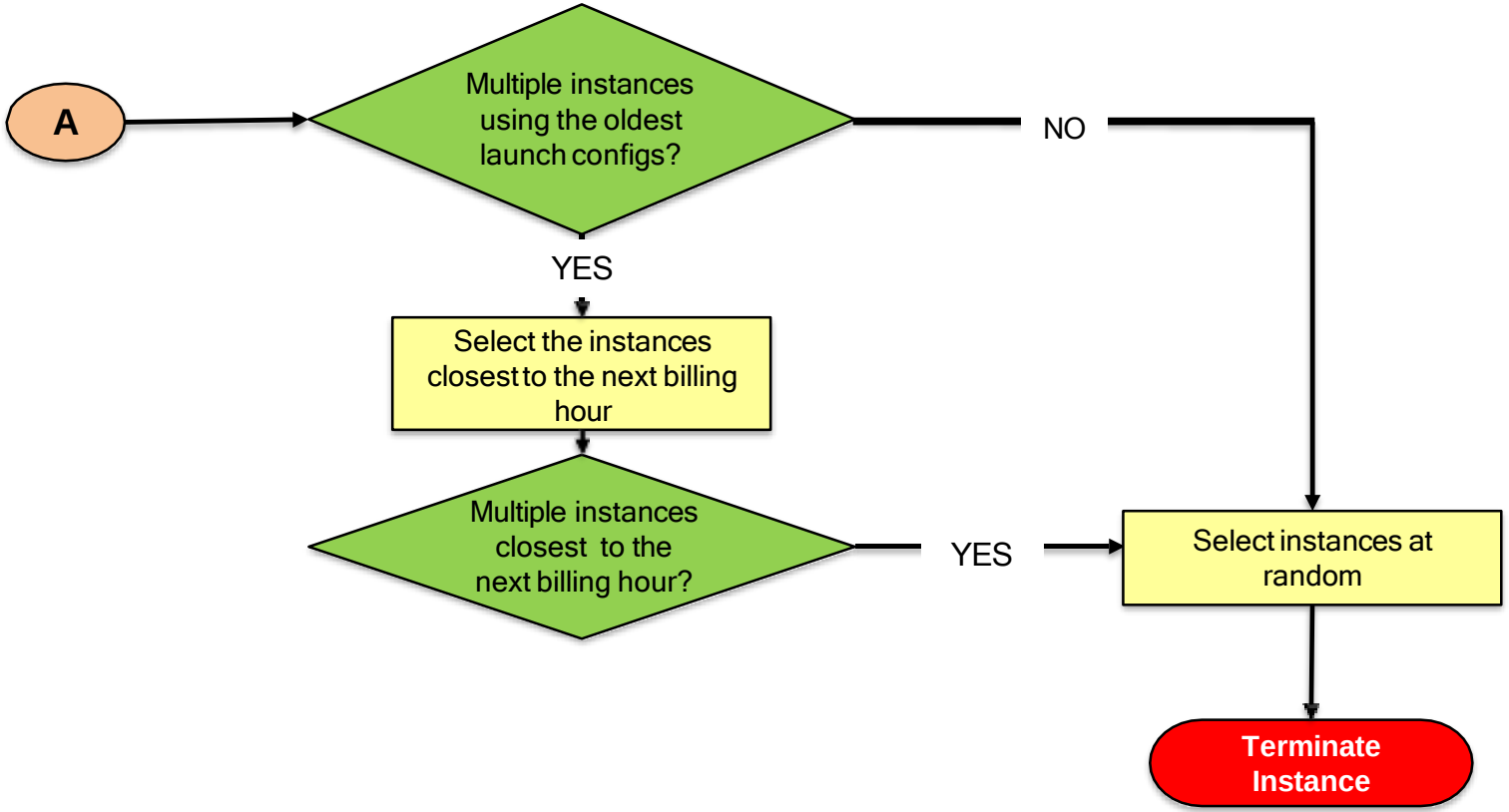


Instance Termination

Instance Termination



Instance Termination



Instance Termination

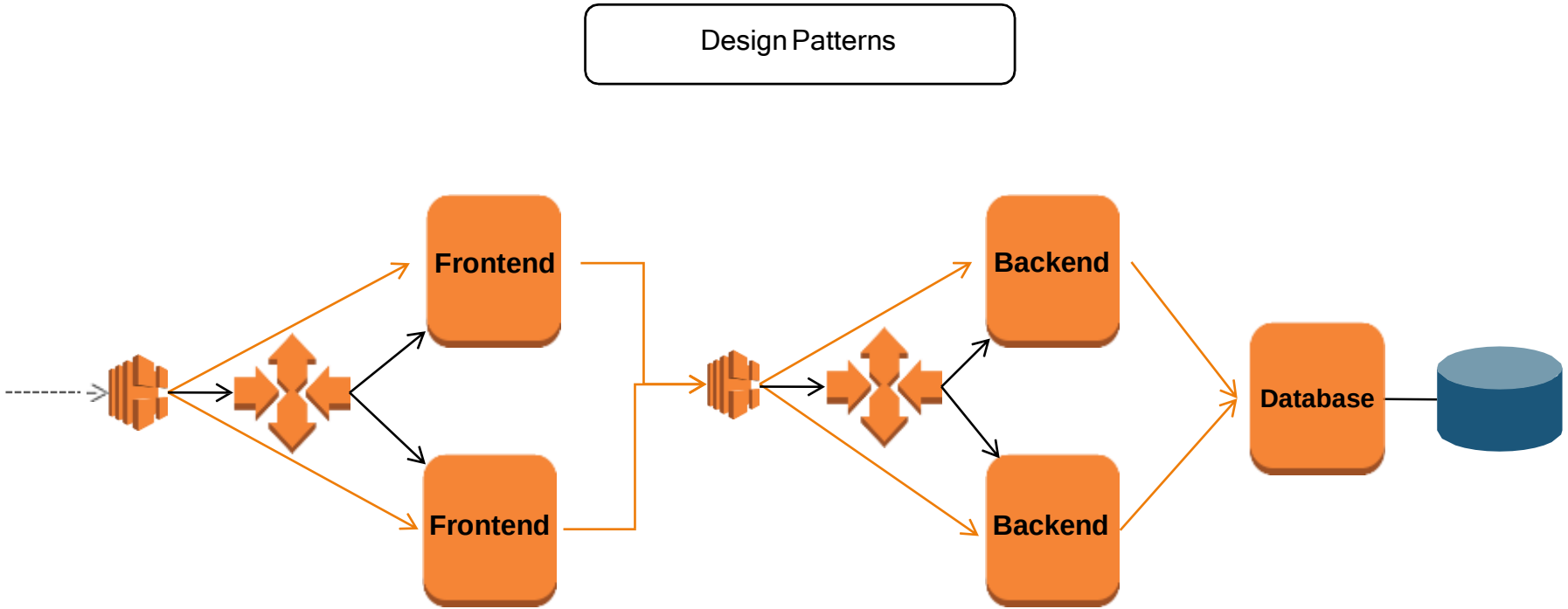
- Termination Policies (Other than the default)
    - Oldest instance

      Newest instance
    - Oldest launch configuration
    - Closest to the next instance hour
- Instance protection does not terminate an instance during a scale in event. It can be enabled at the autoscaling group or individual instance level

Auto-Scaling Pricing

- No additional fees
- Underlying instances are charged hourly
- For details, visit: https://aws.amazon.com/autoscaling/pricing/

Auto-Scaling Design Patterns



ELB & AS Integration

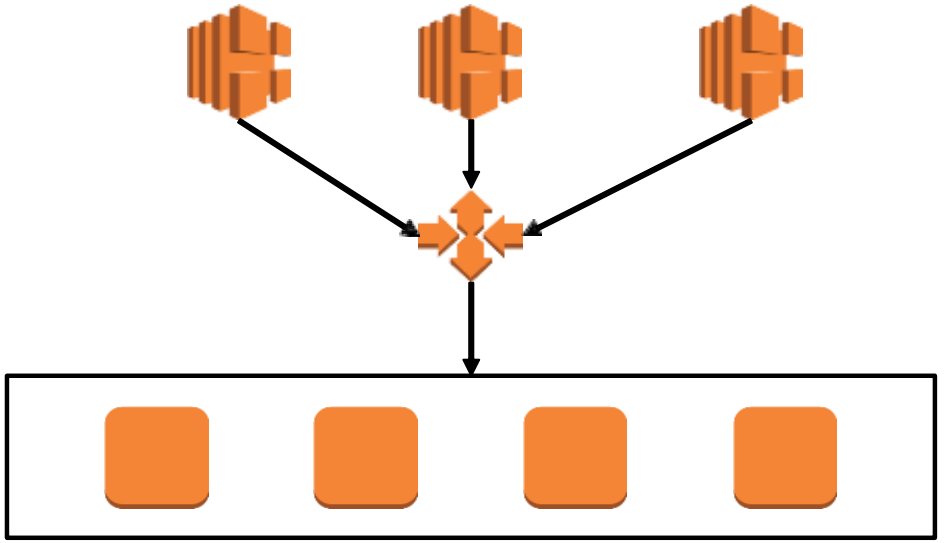
ELB & Auto-Scaling Integration

Autoscaling: Adds and removes capacity as per requirement

Load Balancer: Distributes the incoming traffic evenly across all EC2 instances

Placing ELB in front of AS makes sure that all the incoming traffic are distributed, dynamically changing the number of EC2 instances

ELB is the point of contact between the clients and the backend EC2 instances

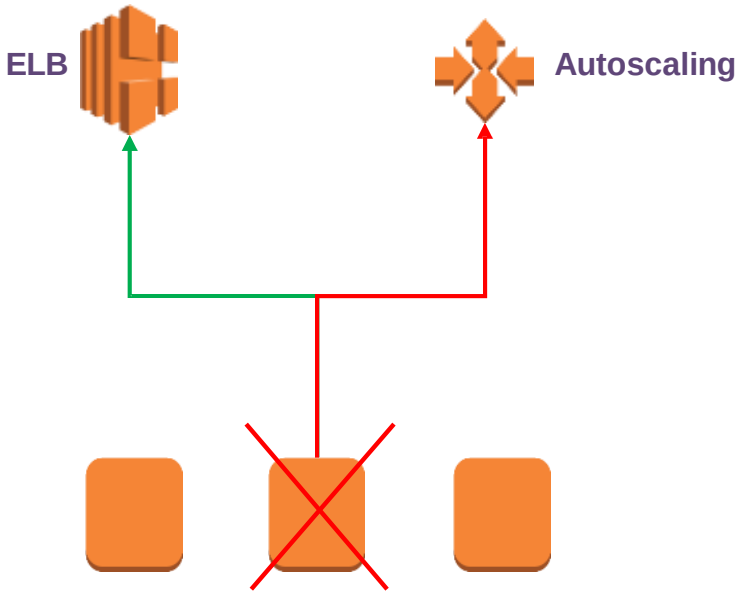


ELB & Auto-Scaling Integration

Load balancer automatically registers instances in the group

Health checks:

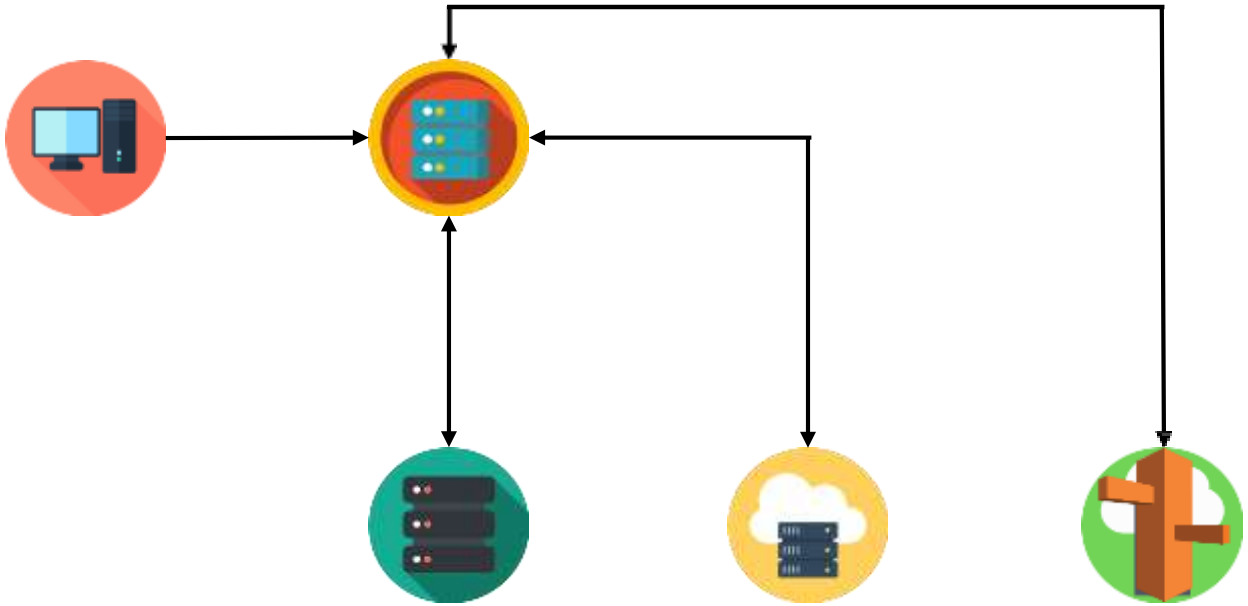
- EC2 instance only: EC2 status checks are considered
- EC2 and ELB health checks: An instance is considered unhealthy if either of the health checks fail



Route 53

What is Route53 ?

Route 53 is the highly available and scalable Domain Name System provided by AWS



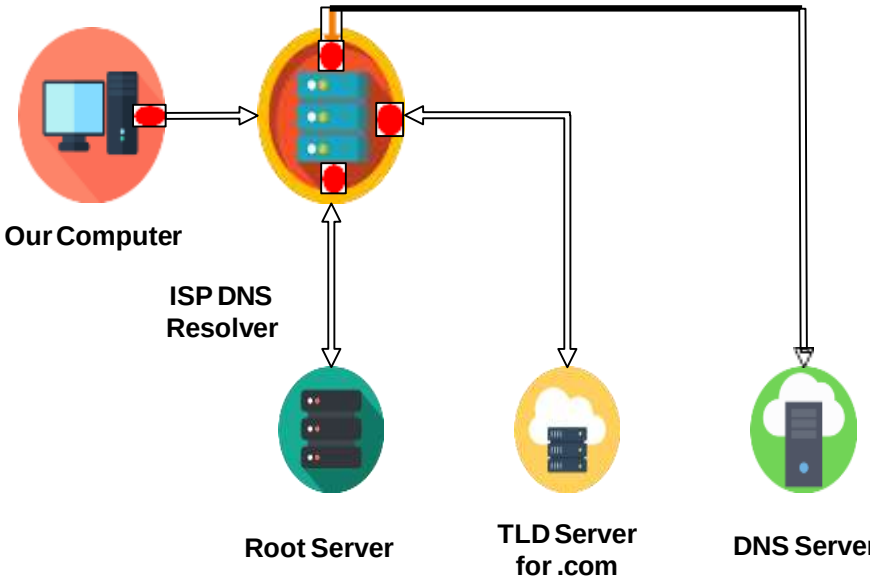
Domain Name System

In www.amazon.com

com: Top-level domain name

Amazon: Domain name

Domain Name System is an Internet service that translates the domain names into IP addresses

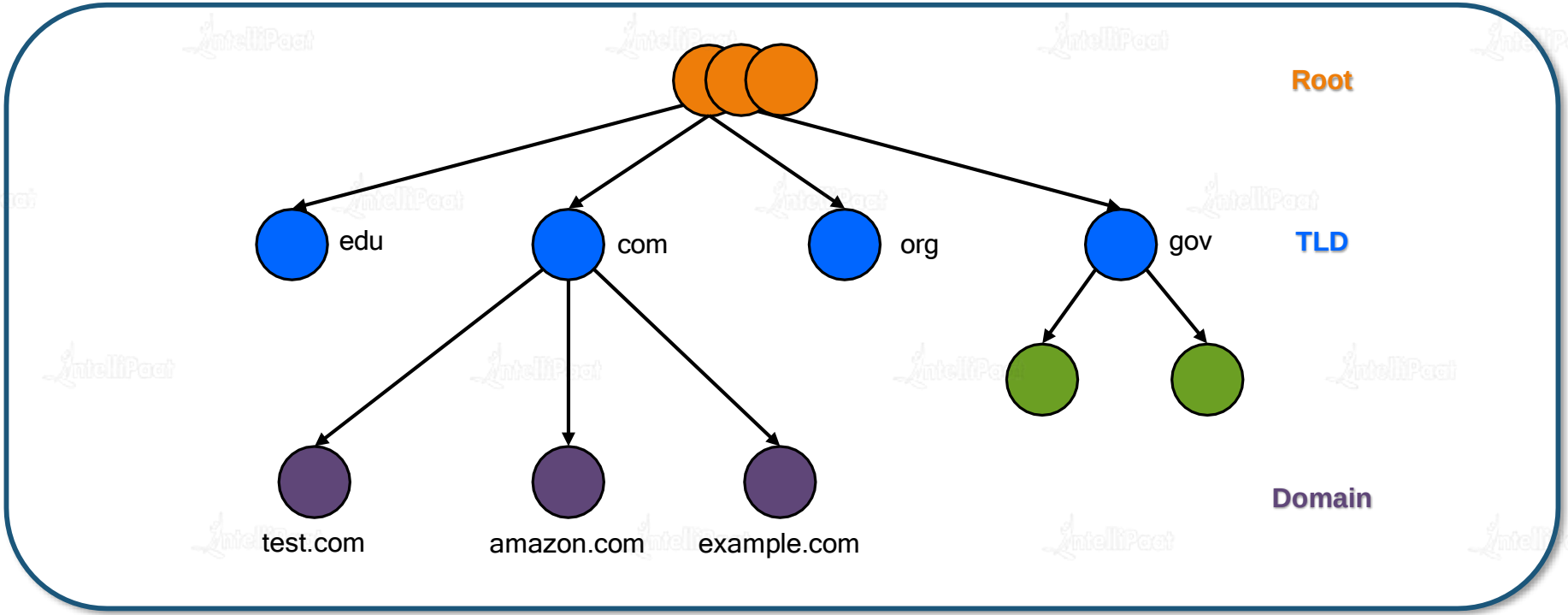


Root servers keep information about TLD servers

TLD servers keep information about authoritative name servers

Name servers contain information about IP addresses for individual domains

DNS Hierarchy



Hosting a Website

Step 1: Start up a server/host where the web service will run (Say, the IP address of the server is 10.20.30.40)

Step 2: Get a domain name from the domain name providers such as GoDaddy, Freenom, etc.

Step 3: Link the domain name with the IP address from Step 1 by using the Domain Name Service/System

DNS Literature

Authoritative Name Server: The server component in Domain Name System (DNS) that holds actual DNS records such as A Name, CNAME, Alias, etc.

'A' NAME Record: It maps the domain name to the IP address of the backend host. 'A' is for address. The A NAME record format is mentioned below:

Type	Domain/Host Name	Address	TTL
A	www.abc.com	101.202.30.40	60
A	www.apple-orange.com	54.28.14.6	300
AAAA	www.example.com	fe80::1cb2:373a:3dd1:8f46	600

DNS Literature

- CNAME (Canonical Name) Record: It maps one name to another name instead of an IP address

Type	Domain/Host Name	Address	TTL
CNAME	www.fruits.com	www.apple-orange.com	300
CNAME	www.vegetables.com	www.fruits.com	600
A	www.apple-orange.com	54.28.14.6	900

- Alias Name is similar to the CNAME record with a 'little' difference

www.vegetables.com www.fruits.com

DNS Literature

Type	Domain/Host Name	Address	TTL
CNAME	www.fruits.com	www.apple-orange.com	300
CNAME	www.vegetables.com	www.fruits.com	600
A	www.apple-orange.com	54.28.14.6	900

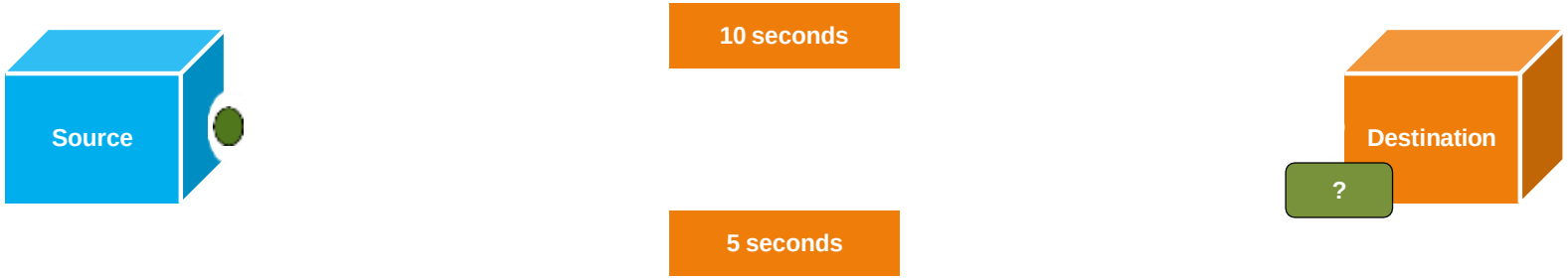
www.fruits.com

www.apple-orange.com

54.28.14.6

Network Latency and Bandwidth

Network latency is the amount of time taken to deliver some amount of data over n/w



Latency = 10 + 5 = 15 seconds

Routing Policy

Routing Policy

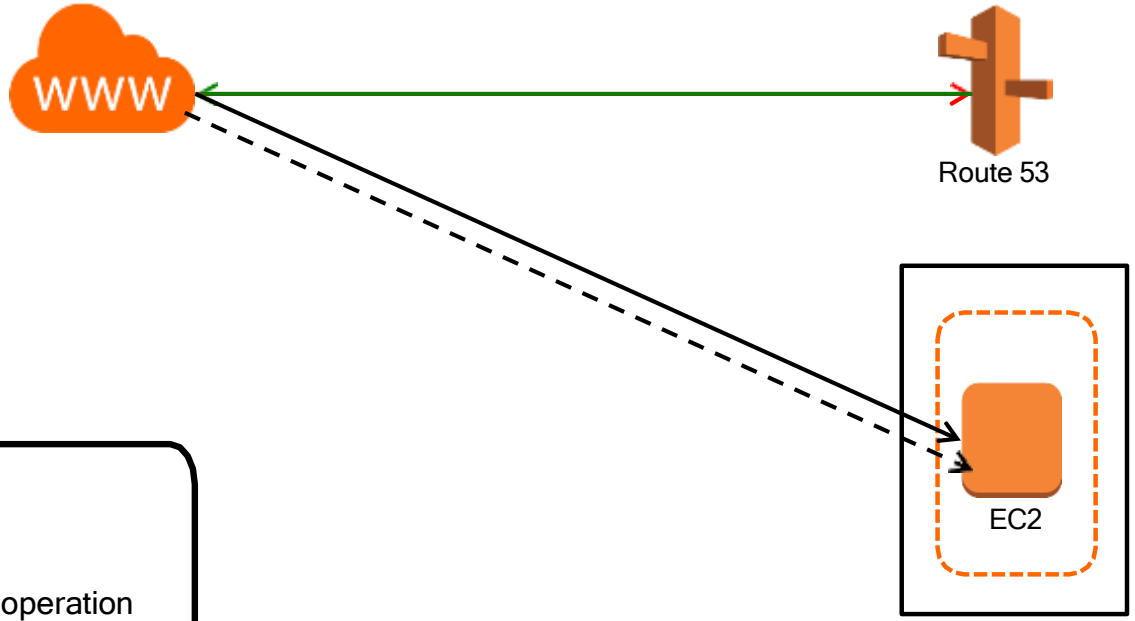
- **Public Hosted Zone** contains information about how the traffic on the Internet should be routed for a domain
- NS record set: The authoritative name servers for a domain name
- SOA (Start of Authority) record set: Contains the base DNS information about the domain

```
ns-2048.awsdns-64.net. hostmaster.example.com.  1  7200  900  1209600  86400
```

ns-2048.awsdns-64.net: Host that created the SOA record
hostmaster.example.com: The email address of the admin with '@' being replaced by '.'
86400: Minimum TTL

- **Private Hosted Zone** contains information about how to route the traffic for a domain within one or more VPCs Note: To use private hosted zones, following VPC settings have to be set to TRUE:
  - enableDnsHostnames enableDnsSupport

Routing Policy

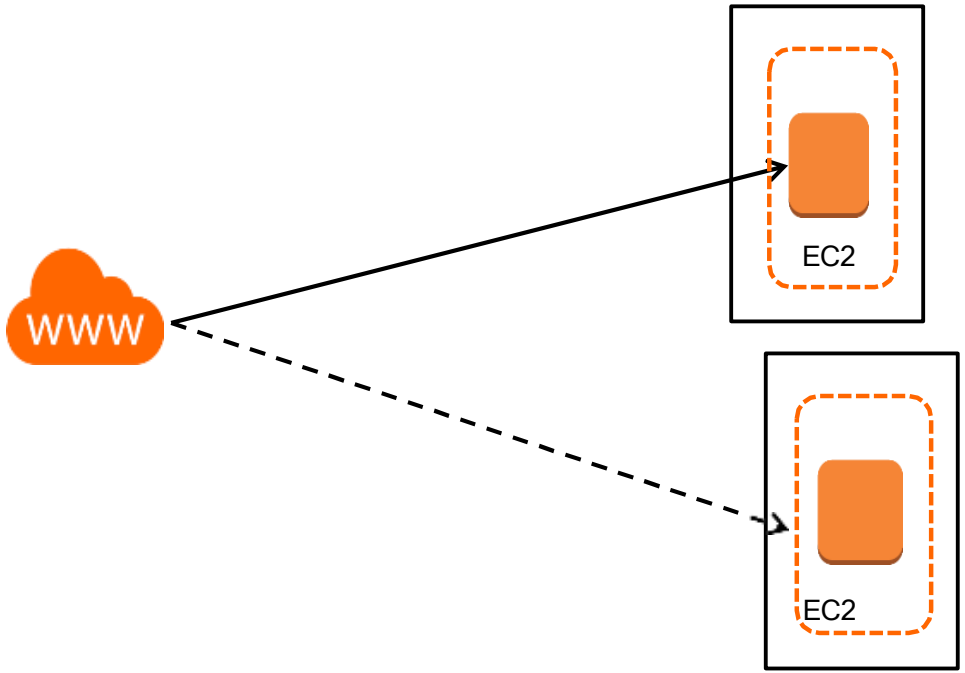


Simple Routing Policy:

A single server performing the desired operation

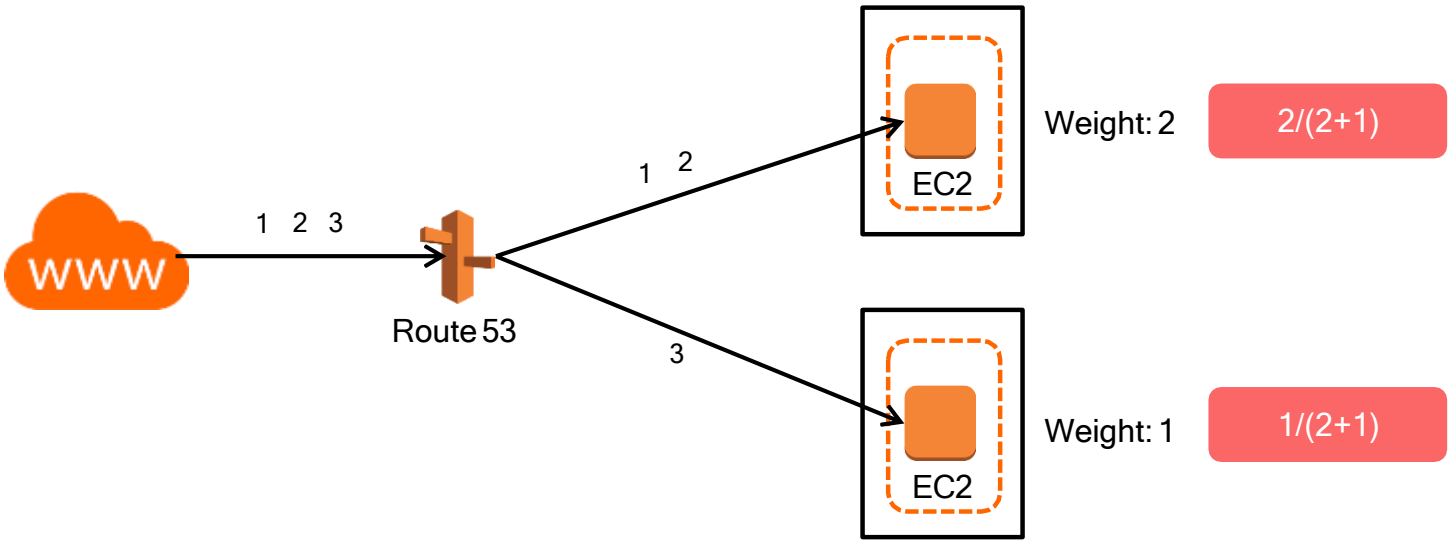
Routing Policy

Failover Routing Policy: Two servers performing the Active-Passive routing



Routing Policy : Weighted

- It associates multiple resources with the same DNS name and type
- Each record set is given a weight and a set ID

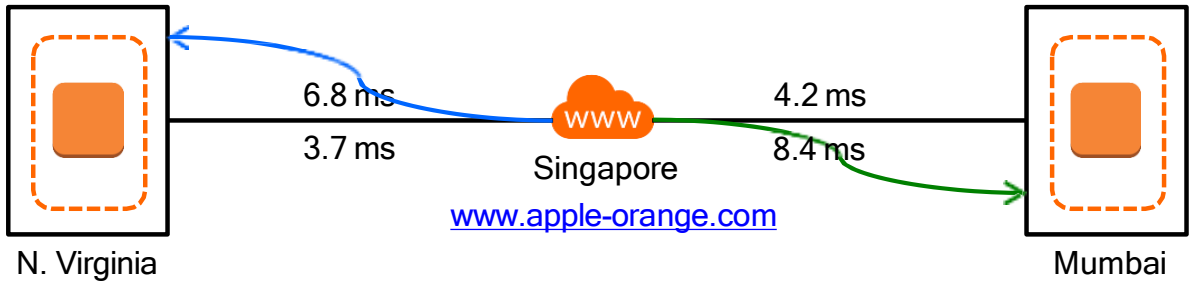


Routing Policy : Latency Based

- If an application is hosted on EC2 instances in multiple regions, user latency can be reduced by serving requests from the region where
- network latency is the lowest
- We have to create a latency resource record set for the Amazon EC2 resource in each region that hosts the

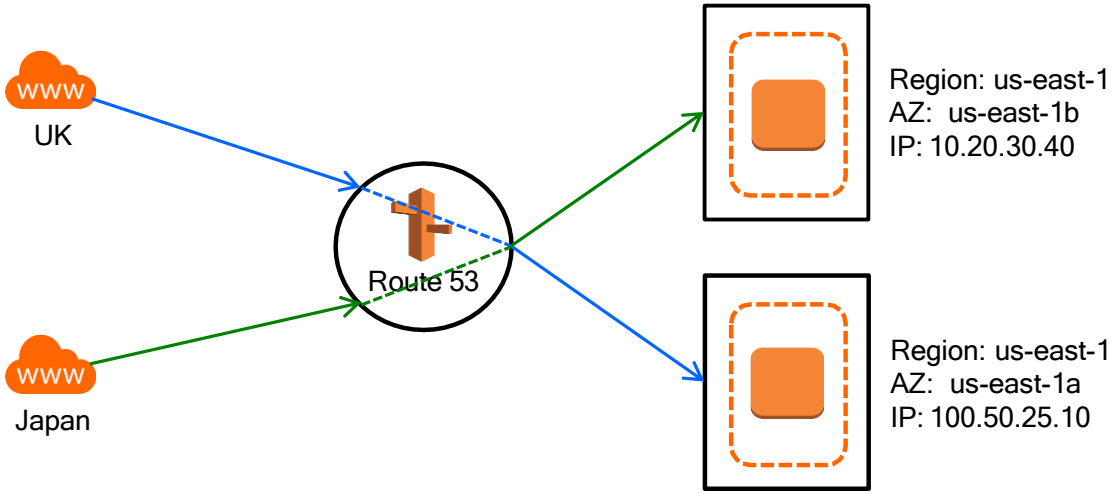
  application Latency record sets can be created for both ELB and EC2 instances

- Latency on the Internet can change over time due to changes in routing or something else



Routing Policy : Geo-Location

- Geolocation routing can be used to send the traffic to resources based on the geographical location of users. For example, all queries from Europe can be routed to the IP address 10.20.30.40
- Geolocation works by mapping IP addresses, irrespective of regions, to locations



India : +91-7847955955

US : 1-800-216-8930 (TOLL FREE)

support@intellipaat.com

24/7 Chat with Our Course Advisor