

## Table of Contents

1.	AWS ELB .....	4
	What is AWS ELB? .....	4
	<b>Types of Load Balancer in AWS</b> .....	4
	a. Classic Load Balancers.....	4
	b. Application Load Balancers.....	5
	Benefits of Application Load balancers- .....	6
	c. Network Load Balancers.....	6
	Benefits of Network Load Balancers-.....	7
	Conclusion.....	7
2.	AWS ELB setup .....	8
	Application Load Balancer setup: .....	8
	Steps:.....	9
	Step1: Create Two instances: Orders & Payments .....	9
	Step2: Target Group creation .....	9
	Step3: Load Balancer Creation.....	11
3.	AWS Scaling-group setup .....	16
	Working:.....	16
	Launch Templates:.....	16
	Steps:.....	17
	Step1: Create Image of an instance .....	17
	Step2: Create Autoscaling group using Launch template .....	18
4.	AWS EKS Setup .....	23
	•   Setup kubectl .....	23
	•   Setup eksctl.....	23
	•   Create an IAM Role and attache it to EC2 instance .....	24
	•   Create your cluster and nodes.....	24
	•   To delete the EKS clsuter .....	24
	•   Validate your cluster using by creating by checking nodes and by creating a pod .....	25
5.	AWS ROUTE53 DNS Records .....	25
	1- SOA (Start of Authority Records) .....	25
	2- NS Record (Name Server Records) .....	26
	1- A Record ( URL to IPv4) .....	27
	2- CNAME (Canonical Records- URL to URL) .....	27
	3- Alias Record: .....	28

4- AAAA: (URL to IPv6) .....	29
5- MX Record (Main Exchange Record) .....	29
6. Routing Policy.....	30
1- Simple Routing Policy: .....	30
2- Weighted Routing Policy.....	31
3- Latency Routing Policy .....	33
4- Failover Routing Policy.....	34
5- Geo Location Routing Policy .....	36
6- Multi Value Routing Policy.....	37
7. Amazon Elastic Disaster Recovery .....	38
Definition: .....	38
Recovery-why?.....	38
Process: .....	39
Architecture: .....	39
How it works? .....	40
EDR Dashboard: .....	40
Steps:.....	41
Step1: Settings .....	41
Step2: IAM Role setup .....	41
Step3: Install Agent .....	41
Step4: Recovery Job.....	42
Step5: Failback .....	43
8. VPC .....	44
Architecture: .....	44
Process: .....	44
Steps:.....	44
Step1: Create VPC .....	44
Step2: Create Internet Gateway and attach it to VPC .....	45
Step3: Create Subnets in Different Availability Zones .....	45
Step4: Create and Assign public route table to public subnet 1 & 2 .....	47
Step5: Create Two Private Route tables and attach it to 4 subnets .....	51
Step6: Create two Elastic IP's.....	52
Step7: Create NAT Gateway.....	53



## 1. AWS ELB

### What is AWS ELB?

**AWS ELB (Amazon Elastic Load Balance)** helps to distribute the application traffic to various different targets such as EC2 instances. The vacant targets which are ready to collect the traffic are monitored by Amazon ELB whether they are healthy or not and the traffic is sent to the healthy one.

AWS ELB comes in three versions which perform different tasks.

- The Version 1 provides detailed instructions for using *Classic Load Balancers*.
- 2nd version provides detailed instructions for using *Application Load Balancers*.
- 3rd provides detailed instructions for using *Network Load Balancers*.

### Types of Load Balancer in AWS

These are the unique Versions of AWS ELB, let's discuss them one by one:

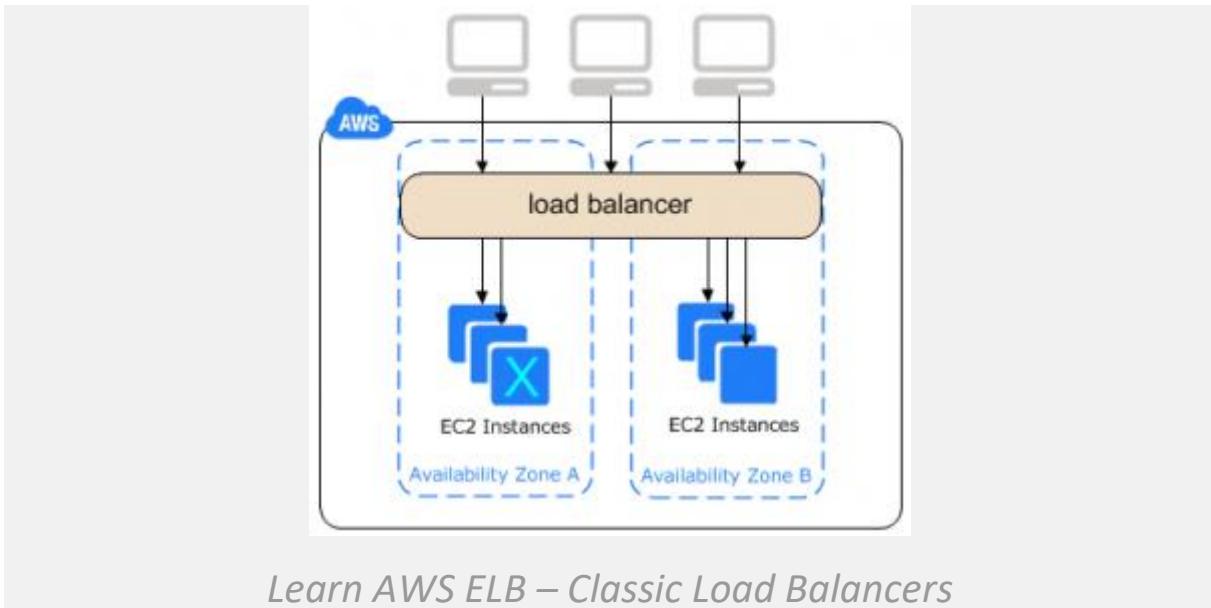
#### a. Classic Load Balancers

Classic Load Balancers distribute upcoming traffic to different EC2 instances in multiple Availability Zones. During this process, there is a chance of the fault tolerance of your application. These Load Balancers detect healthy and unhealthy instance and direct the traffic towards only healthy ones.

It also helps in a way such that without disrupting the flow of requests to your application you can add or remove instances from your load balancers as your need changes.

AWS ELB can calculate the majority of workloads automatically. Protocol and port which a person configures are used to detect the connection requests from clients it also forwards requests to one or more registered instances.

The number of instances can be modified. Health checks can be monitored so that the load balancer only sends requests to the healthy instances.



### *Learn AWS ELB – Classic Load Balancers*

#### **Benefits of Load Balancers-**

- Provides Support to TCP and SSL listeners.
- Support to Sticky Session.
- Support to EC2- Classic.

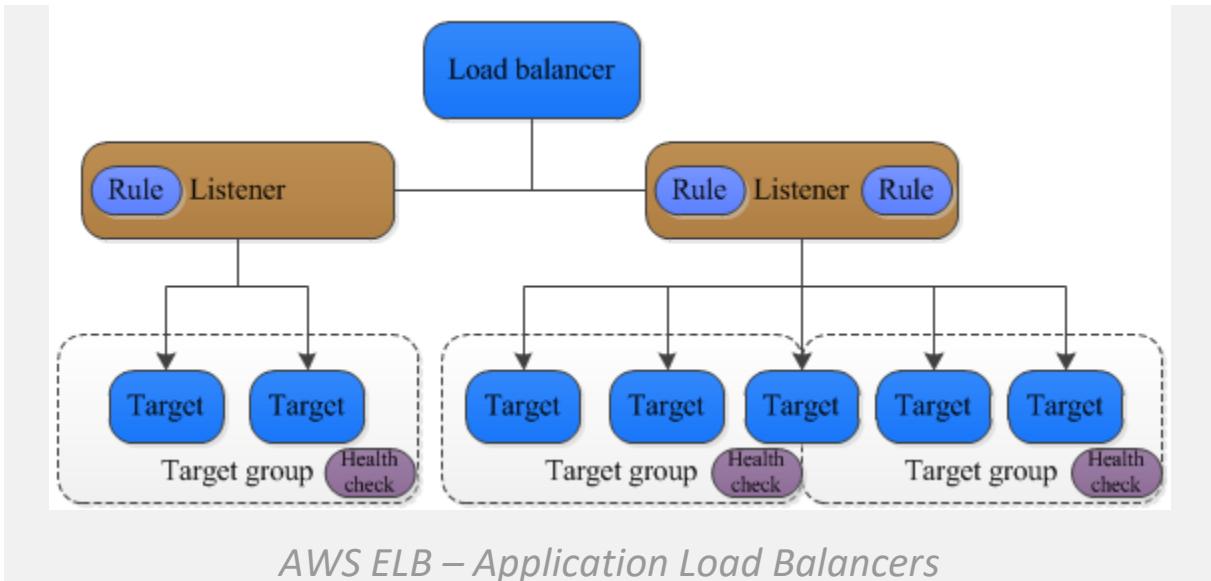
#### **b. Application Load Balancers**

After receiving the request Application Load Balancer analyzes the rules provided by the listener in priority order and determines the rule which has to apply. After that, it selects a target from the target group for the rule action.

An Application Load Balancer functions at the Open Systems Interconnection (OSI) model which is the seventh layer of the OSI model.

The User can analyze the rules of the listener and can modify it by sending it to different target groups based on the content of the application traffic even when the target is associated with multiple target groups.

Addition and removal of tags can do from the load balancers as per your needs. This can done without breaking the flow of your requests of the application.



Benefits of Application Load balancers-

- Load Balancer's performance improve in Application Load Balancer.
- Access logs containing information compress such that they may not require the additional space.
- Provides benefit for registering targets by IP address, including targets outside the VPC for the load balancer.

### c. Network Load Balancers

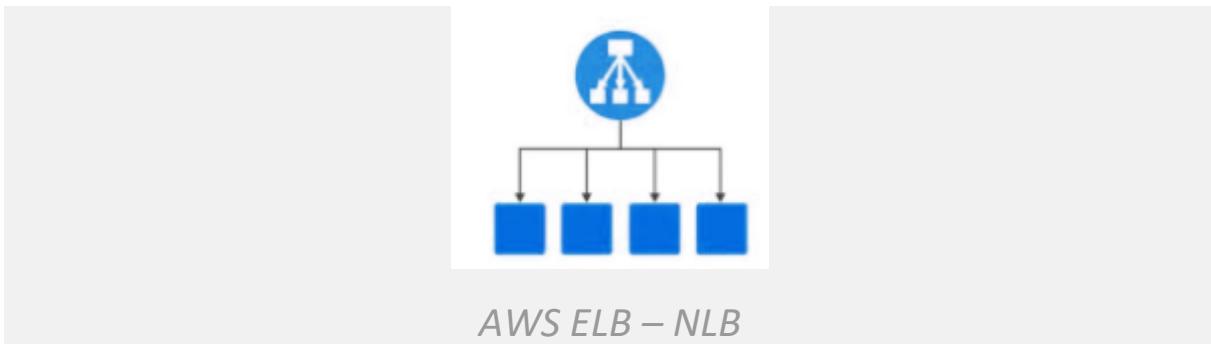
It is the fourth layer of the Open System Interconnection Model. After the load balancer receives a connection request, it selects a target from the group which targets for the default rule.

After enabling the availability zone Elastic Load Balancer creates the load balancer node in the availability zone. Each load balancer node automatically distributes traffic across the registered targets in its Availability Zone only.

Cross-zone Load Balancing enables to distribute traffic across the registered targets in all enabled Availability Zones.

Enabling Multiple Availability Zone can cause harm by increasing the fault tolerance of the applications and it will happen if each target group has at least one target in each enabled Availability Zone.

The problem can overcome in such a way that if one or more target groups do not have a healthy target in an Availability Zone, the IP address for the corresponding subnet from DNS is removed. If a person attempts again the request fails.



Benefits of Network Load Balancers-

- NLB Provides the Support for static IP addresses for the load balancer.
- Provides support for registering targets by IP address which includes target outside the VPC for the Load Balancer.
- Provides support for monitoring the health of each service independently.

So, this was all about AWS ELB Tutorial. Hope you like our explanation.

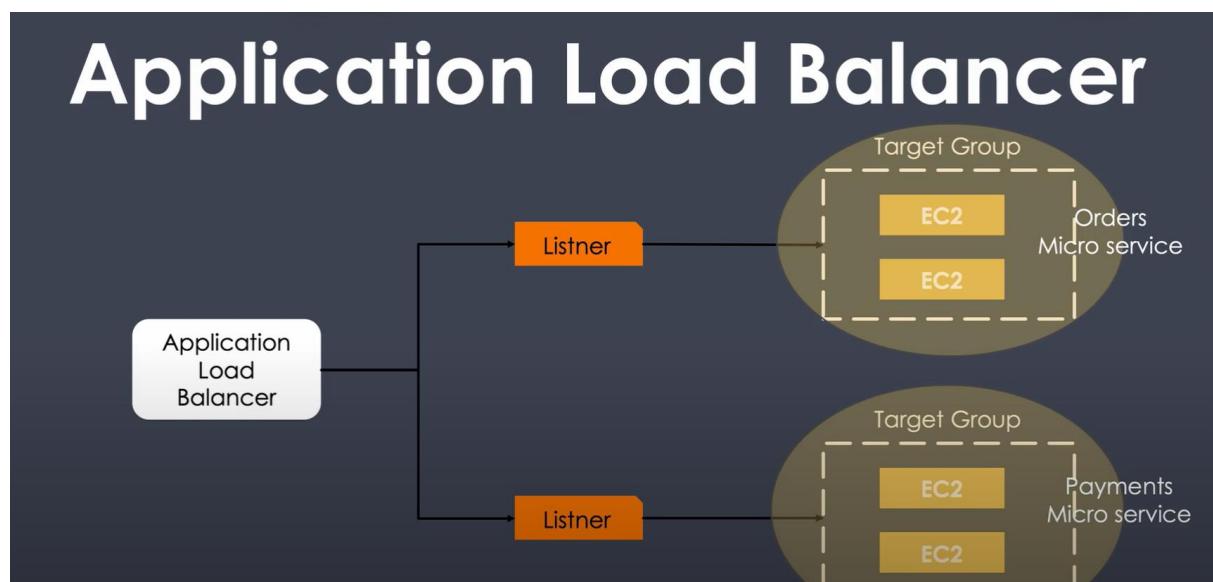
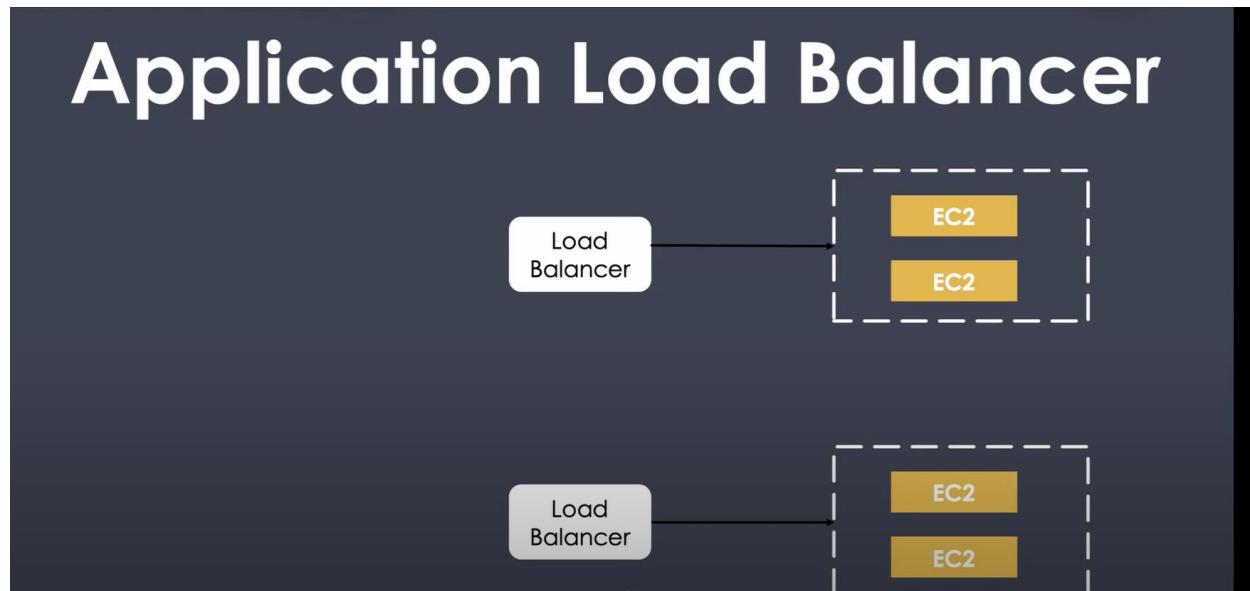
## Conclusion

Hence, in AWS ELB tutorial, we studied the load balancer distributes traffic evenly across the Availability Zones that you enable for your load balancer. AWS Elastic Load Balance is one of the effective ways to manage and deploy the application as it is fast, reliable, and efficient.

Load Balancer simplifies and improves the security of your application and hence provides security which makes it a reliable function. Furthermore, if you have any doubt regarding AWS ELB, feel free to ask in the comment box.

## 2. AWS ELB setup

Application Load Balancer setup:



Listener make routing decisions like when to route traffic to orders and when to route traffic to Payments.

Routing decisions are made based on the path or port (rules)

## Steps:

### Step1: Create Two instances: Orders & Payments

- Create an Ec2 instance with Linux image.
- During the instance creation, use the below code in the User instructions to create an html file in Advanced Settings → User data for Orders.

```
apache.sh
1  #!/bin/bash
2
3  yum install httpd -y
4
5  systemctl enable httpd
6
7  mkdir /var/www/html/orders/
8
9  echo "<h1>This is Orders App</h1>" > /var/www/html/orders/index.html
10
11 systemctl start httpd
```

- Launch instance
- Similarly, launch instance for Payments application by giving the user data as shown below.

```
apache.sh
1  #!/bin/bash
2
3  yum install httpd -y
4
5  systemctl enable httpd
6
7  mkdir /var/www/html/orders/
8
9  echo "<h1>This is Orders App</h1>" > /var/www/html/payments/index.html
10
11 systemctl start httpd
12
```

### Step2: Target Group creation

- Choose Load Balancing → Target groups from AWS console Left Pane and **Create**
- Enter Details like Target group name as required, Target Type as **Instance**, Protocol as **HTTP**, port as **80**, VPC as required.
- Under Health check settings, update path as **"/orders/index.html"** path of the application and click **Create**

**Create target group**

Your load balancer routes requests to the targets in a target group using the target group settings that you specify, and performs health checks on the targets using the health check settings that you specify.

Target group name	<input type="text" value="orders"/>
Target type	<input checked="" type="radio"/> Instance <input type="radio"/> IP <input type="radio"/> Lambda function
Protocol	<input type="text" value="HTTP"/>
Port	<input type="text" value="80"/>
VPC	<input type="text" value="vpc-2397a44b (172.31.0.0/16) (My Default VPC)"/>
<b>Health check settings</b>	
Protocol	<input type="text" value="HTTP"/>
Path	<input type="text" value="/orders/index.html"/>
<b>Advanced health check settings</b>	

- Create another Target group for Payments Micro-service and update the Path as “**/payments/index.html**”
- Once the Target groups are created, associate the Targets for the Target group.

**Create target group** Actions ▾

Filter by tags and attributes or search by keyword

Name	Port	Protocol	Target type	Load Balancer	VPC ID	Monitoring
orders	80	HTTP	instance		vpc-2397a44b	

**Target group: orders**

Description Targets Health checks Monitoring Tags

**Basic Configuration**

Name	orders
ARN	arn:aws:elasticloadbalancing:ap-south-1:288621183532:targetgroup/orders/fb7b26f48796d679
Protocol	HTTP
Port	80
Target type	instance
VPC	vpc-2397a44b
Load balancer	

- Click **Edit** & register the instances as Target in the Target group & Save

**Register and deregister targets**

**Registered targets**  
To deregister instances, select one or more registered instances and then click Remove.

Remove	Instance	Name	Port	State	Security groups	Zone
<input checked="" type="checkbox"/>	i-08858ebdac7b4dbe9	Orders	80	<span style="color: green;">running</span>	apache-sg	ap-south-1a

**Instances**  
To register additional instances, select one or more running instances, specify a port, and then click Add. The default port is the port specified for the target group. If the instance is already registered on the specified port, you must specify a different port.

Add to registered	on port 80						
<input checked="" type="checkbox"/>	i-08858ebdac7b4dbe9	Orders	<span style="color: green;">running</span>	apache-sg	ap-south-1a	subnet-6bb4ec03	172.31.32.0/20
<input type="checkbox"/>	i-037498db731...	Payments	<span style="color: green;">running</span>	apache-sg	ap-south-1a	subnet-6bb4ec03	172.31.32.0/20

**Cancel** **Save**

- Similarly, register the “Payments” instance as Target for “Payment” Target group
- You can check the status of Target groups

**Target group: orders**

**Description** **Targets** **Health checks** **Monitoring** **Tags**

The load balancer starts routing requests to a newly registered target as soon as the registration process completes and the target passes the initial health checks. If demand on your targets increases, you can register additional targets. If demand on your targets decreases, you can deregister targets.

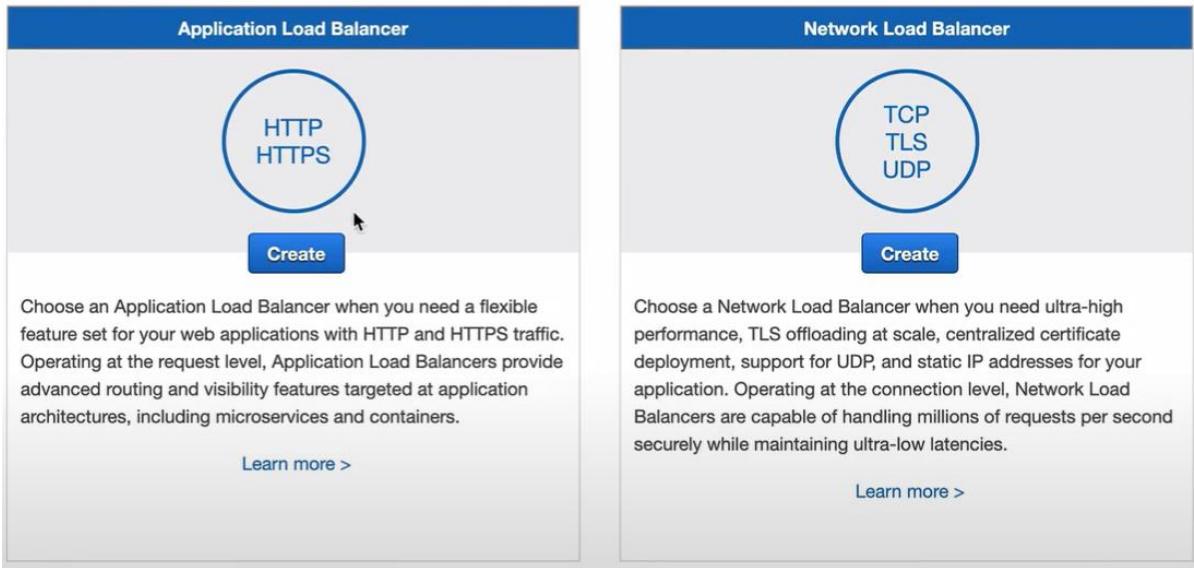
**Edit**

**Registered targets**

Instance ID	Name	Port	Availability Zone	Status
i-08858ebdac7b4dbe9	Orders	80	ap-south-1a	unused <span>(i)</span>

## Step3: Load Balancer Creation

- Select Load Balancing → Load Balancer from AWS Console LP & Create Load Balancer



- In **Configure Load Balancer**, Enter as Required, Scheme as **Internet-facing**, IP address type as IPv4, Select default VPC from Availability zones and select the availability zones as required.
- Note:** The instances must reside on one of the availability zones listed.

Step 1: Configure Load Balancer

1. Configure Load Balancer    2. Configure Security Settings    3. Configure Security Groups    4. Configure Routing    5. Register Targets    6. Review

HTTP    80

Add listener

#### Availability Zones

Specify the Availability Zones to enable for your load balancer. The load balancer routes traffic to the targets in these Availability Zones only. You can choose from at least two Availability Zones to increase the availability of your load balancer.

VPC	vpc-2397a44b (172.31.0.0/16) (default)
Availability Zones	<input checked="" type="checkbox"/> ap-south-1a    subnet-6bb4ec03 <input checked="" type="checkbox"/> ap-south-1b    subnet-4f961f03 <input checked="" type="checkbox"/> ap-south-1c    subnet-c36fbfb8

- **Configure Security group** as required

### Step 3: Configure Security Groups

A security group is a set of firewall rules that control the traffic to your load balancer. On this page, you can add rules to allow specific traffic to reach your load balancer. First, decide whether to create a new security group or select an existing one.

Assign a security group:  Create a new security group  
 Select an existing security group

Security Group ID	Name	Description
<input type="checkbox"/> sg-06c1a47532c1d4644	all-traffic	all-traffic
<input checked="" type="checkbox"/> sg-0372e0cb4cc962a05	apache-sg	apache-sg
<input type="checkbox"/> sg-5048e13f	default	default VPC security group
<input type="checkbox"/> sg-036b439cb05fb18ea	launch-wizard-1	launch-wizard-1 created 2019-07-12T22:04:30.308+05:30
<input type="checkbox"/> sg-0343987060f61181f	launch-wizard-2	launch-wizard-2 created 2019-07-12T22:05:25.859+05:30
<input type="checkbox"/> sg-083ea5d3c1ec0f54c	launch-wizard-3	launch-wizard-3 created 2019-07-13T18:07:39.020+05:30
<input type="checkbox"/> sg-0bbe5064ee70038ff	webapp-sg	webapp-sg

- Configure Routing, Select the Target group created for Orders.

### Step 4: Configure Routing

Your load balancer routes requests to the targets in this target group using the protocol and port that you specify, and performs health checks on the targets using these health check settings. Note that each target group can be associated with only one load balancer.

#### Target group

Target group	<input type="radio"/> Existing target group
Name	orders
Target type	<input checked="" type="radio"/> Instance <input type="radio"/> IP <input type="radio"/> Lambda function
Protocol	HTTP
Port	80

#### Health checks

Protocol	HTTP
Path	/orders/index.html

► Advanced health check settings

- Register Targets

### Step 5: Register Targets

Register targets with your target group. If you register a target in an enabled Availability Zone, the load balancer starts routing requests to the targets as soon as the registration process completes and the target passes the initial health checks.

#### Registered targets

The following targets are registered with the target group that you selected. You can only modify this list after you create the load balancer.

Instance	Port
i-08858ebdac7b4dbe9	80

- Review & Create

### Step 6: Review

Please review the load balancer details before continuing

Load balancer

Name	javahome-alb
Scheme	internet-facing
Listeners	Port:80 - Protocol:HTTP
IP address type	ipv4
VPC	vpc-2397a44b
Subnets	subnet-6bb4ec03, subnet-4f961f03, subnet-c36fbfb8
Tags	

Security groups

Security groups	sg-0372e0cb4cc962a05
-----------------	----------------------

Routing

Target group	Existing target group
Target group name	orders
Port	80
Target type	instance
Protocol	HTTP
Health check protocol	HTTP
Path	/orders/index.html
Health check port	traffic port
Healthy threshold	5
Unhealthy threshold	2
Timeout	5

- Go to **Listeners**, it will have a default Listener and Rule

Create Load Balancer Actions ▾

Filter by tags and attributes or search by keyword

Name	DNS name	State	VPC ID	Availability Zones	Type
javahome-alb	javahome-alb-1209396547.a...	provisioning	vpc-2397a44b	ap-south-1b, ap-south-...	application

Load balancer: javahome-alb

Description    **Listeners**    Monitoring    Integrated services    Tags

A listener checks for connection requests using its configured protocol and port, and the load balancer uses the listener rules to route requests to targets. You can add, remove, or update listeners and listener rules.

Add listener    Edit    Delete

Listener ID	Security policy	SSL Certificate	Rules
HTTP : 80	N/A	N/A	Default: forwarding to <a href="#">orders</a> View/edit rules

- Edit the rule to forward to Target group- Orders if path is **/orders/\***

AWS Services Resource Groups ▾ JavaHome Mumbai Support ▾

Rules + ⌂ ⌂ ⌂ ⌂ To edit, select a mode above.

javahome-alb | HTTP:80 (2 rules)

Rule limits for condition values, wildcards, and total rules.

1	arn...a70b2	IF ✓ Path is /orders*	THEN Forward to orders
last	HTTP 80: default action <i>This rule cannot be moved or deleted</i>	IF ✓ Requests otherwise not routed	THEN Forward to orders

- Add another rule to forward to Target group- Payments if path is **/payments/\***

The screenshot shows the AWS Lambda@Edge Rule Editor for a resource named "javahome-alb | HTTP:80". There are two rules defined:

- Rule 1:** IF Path... is /payments THEN Forward to orders
- Rule 2:** IF Path is /orders\* THEN Forward to orders

- Now, check the Target group status, it would be updated as Healthy

The screenshot shows the AWS Lambda@Edge Target Groups page for the "orders" target group. It displays the following information:

- Targets:**

Name	Port	Protocol	Target type	Load Balancer	VPC ID
orders	80	HTTP	instance	javahome-alb	vpc-2397a44b
payments	80	HTTP	instance	javahome-alb	vpc-2397a44b
- Registered targets:**

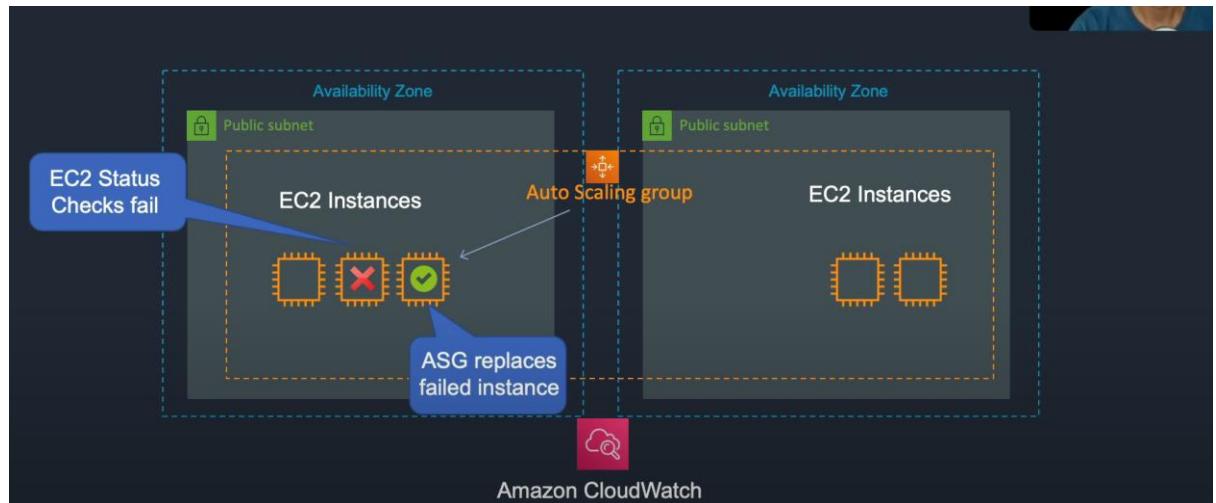
Instance ID	Name	Port	Availability Zone	Status
i-08858ebdac7b4dbe9	Orders	80	ap-south-1a	healthy (i)
- Availability Zones:**

Availability Zone	Target count	Healthy?
ap-south-1a	1	Yes

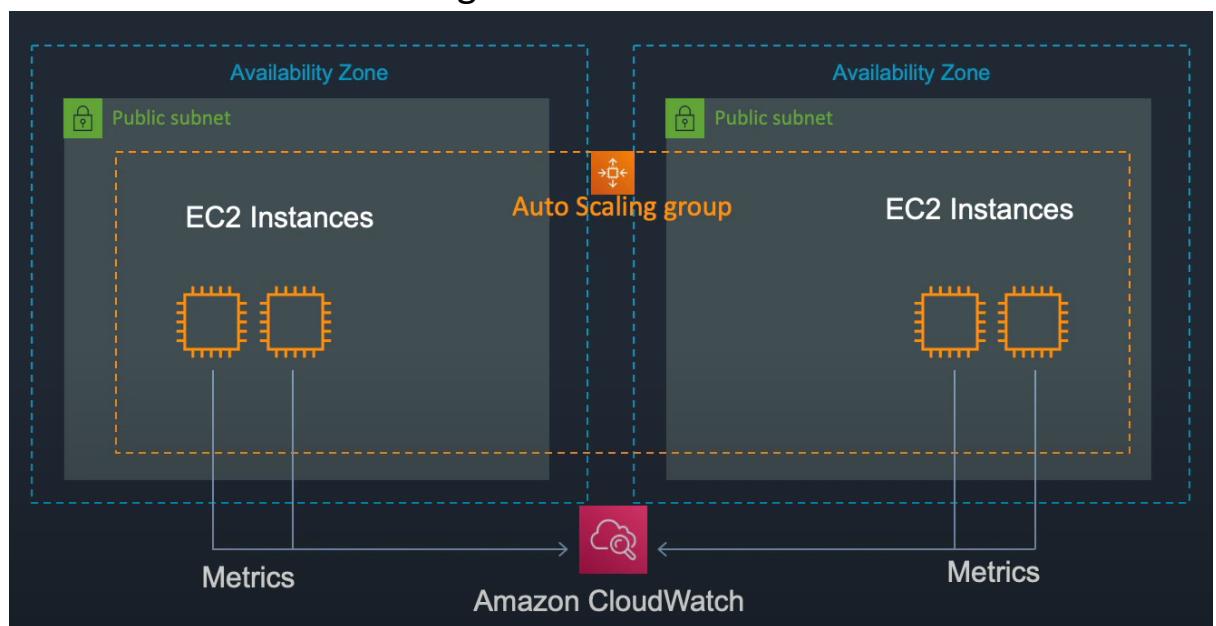
### 3. AWS Scaling-group setup

Working:

- Based on the Health checks



- Based on the Metrics using CloudWatch



Launch Templates:

- Features:

## EC2 Auto Scaling – Launch Templates

Press Esc to exit full screen

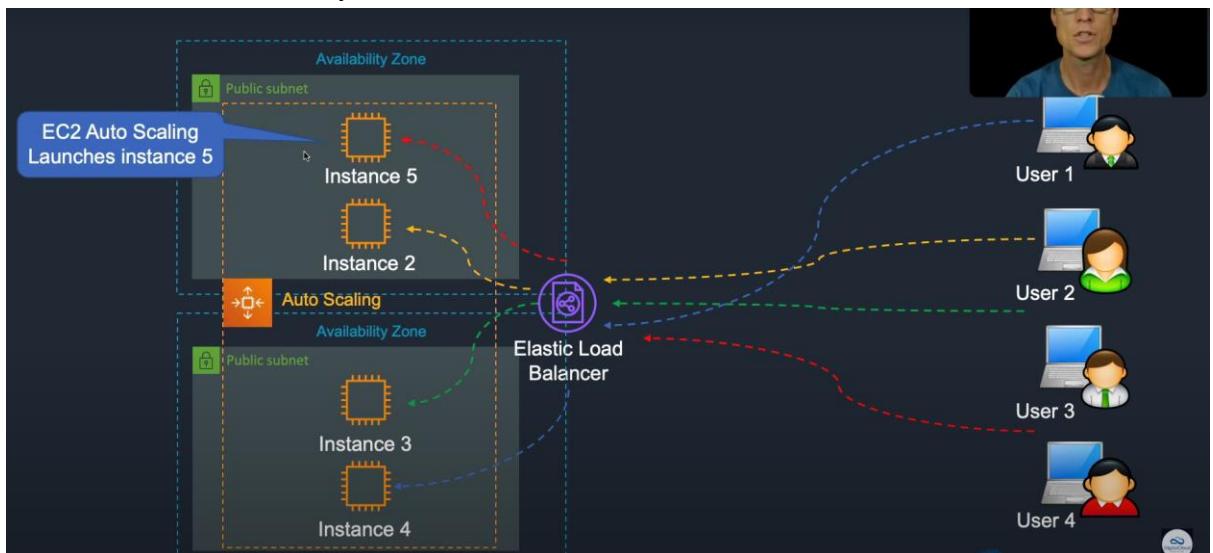
Similar to a Launch Configuration but offers some additional features:

- Can have multiple versions of a template (launch configurations cannot be edited)
- Use dedicated hosts
- Use both Spot and On-demand instances
- Use multiple instance types
- Configure advanced settings such as termination protection, shutdown behavior, placement groups etc.

Steps:

Step1: Create Image of an instance

- Create an Image by selecting an EC2 instance,  
Actions→Image→ Create Image
- Then Select AMI from AWS LP to see the image being created.
- Create ELB with Target group but need not add any Targets in the Target group as it will be taken care as a part of Auto-scaling group
- The ELB should be pointing to Public subnets available in the different availability Zones



## Step2: Create Autoscaling group using Launch template

- Select the Auto Scaling → Auto Scaling Group from AWS LP
- Click Create Auto-scaling group
- Enter Name of the ASG and click **Create a launch template**

The screenshot shows the 'Create Auto Scaling group' wizard. On the left, a vertical sidebar lists steps: Step 1 (current), Step 2 (Configure settings), Step 3 (optional) (Configure advanced options), Step 4 (optional) (Configure group size and scaling policies), Step 5 (optional) (Add notifications), Step 6 (optional) (Add tags), and Step 7 (Review). The main area is titled 'Choose launch template or configuration'. It contains a 'Name' input field with 'Auto Scaling group name' and a note: 'Enter a name to identify the group.' Below it is a note: 'Must be unique to this account in the current Region and no more than 255 characters.' A 'Launch template' section follows, with a dropdown menu showing 'Select a launch template' and a link 'Create a launch template'. At the bottom right are 'Cancel' and 'Next' buttons.

- Enter Launch template name as required, in select AMI → Select the Image created from EC2 Instance, Instance Type, Key Pair, Select VPC, Security group, IAM Instance profile, and then click Launch Template.

EC2 > Launch templates > Create launch template

## Create launch template

Creating a launch template allows you to create a saved instance configuration that can be reused, shared and launched at a later time. Templates can have multiple versions.

### Launch template name and description

Launch template name - required

Must be unique to this account. Max 128 chars. No spaces or special characters like '&', '\*', '@'.

Template version description

Max 255 chars

Auto Scaling guidance [Info](#)  
Select this if you intend to use this template with EC2 Auto Scaling  
 Provide guidance to help me set up a template that I can use with EC2 Auto Scaling

► Template tags  
► Source template

Launch template contents

Specify the details of your launch template below. Leaving a field blank will result in the field not being included in the launch template.

**Amazon machine image (AMI) - required** [Info](#)

AMI - required

**Instance type** [Info](#)

Instance type

t2.micro	Free tier eligible	<a href="#">Instance types</a>
Family: General purpose    1 vCPU    1 GiB Memory		
On-Demand Linux pricing: 0.0146 USD per Hour		
On-Demand Windows pricing: 0.0192 USD per Hour		

**Key pair (login)** [Info](#)

Key pair name

[Create new key pair](#)

**Network settings**

Networking platform [Info](#)

Virtual Private Cloud (VPC)  
Launch into a virtual network in your own logically isolated area within the AWS cloud

EC2-Classic  
Launch into a single flat network that you share with other customers

Security groups [Info](#)

[Create new security group](#)

**Storage (volumes)** [Info](#)

- Go back one step, choose the newly create Launch template and click Next

EC2 > Auto Scaling groups > Create Auto Scaling group

**Step 1 Choose launch template or configuration**

**Step 2 Configure settings**

**Step 3 (optional) Configure advanced options**

**Step 4 (optional) Configure group size and scaling policies**

**Step 5 (optional) Add notifications**

**Step 6 (optional) Add tags**

**Step 7 Review**

### Choose launch template or configuration Info

Specify a launch template that contains settings common to all EC2 instances that are launched by this Auto Scaling group. If you currently use launch configurations, you might consider migrating to launch templates.

Name	
Auto Scaling group name	Enter a name to identify the group. <b>ASG1</b>
Must be unique to this account in the current Region and no more than 255 characters.	

Launch template <small>Info</small>		Switch to launch configuration
Launch template		
Choose a launch template that contains the instance-level settings, such as the Amazon Machine Image (AMI), instance type, key pair, and security groups.		
<b>LaunchTemplate1</b>		<input type="button" value="C"/>
<a href="#">Create a launch template</a>		
Version		
<b>Default (1)</b>		<input type="button" value="C"/>
<a href="#">Create a launch template version</a>		
Description	Launch template <b>LaunchTemplate1</b> lt-0e367a8cd1016e2d4	Instance type <b>t2.micro</b>
AMI ID	Security groups -	Security group IDs <b>sg-0cc5dec3fb96b0ab1</b>

- You can choose Instances distribution in **Configure settings** & choose the Public subnets from different availability zones from the VPC

EC2 > Auto Scaling groups > Create Auto Scaling group

**Step 1 Choose launch template or configuration**

**Step 2 Configure settings**

**Step 3 (optional) Configure advanced options**

**Step 4 (optional) Configure group size and scaling policies**

**Step 5 (optional) Add notifications**

**Step 6 (optional) Add tags**

**Step 7 Review**

### Configure settings Info

Configure the settings below. Depending on whether you chose a launch template, these settings may include options to help you make optimal use of EC2 resources.

Purchase options and instance types <small>Info</small>	
<input checked="" type="radio"/> Adhere to launch template The launch template determines the purchase option (On-Demand or Spot) and instance type.	<input type="radio"/> Combine purchase options and instance types Specify how much On-Demand and Spot capacity to launch and multiple instance types (optional). This choice is most helpful for optimizing the scale and cost for a fleet of instances.

Network <small>Info</small>	
<p>Select subnets</p> <p><b>ap-southeast-2a   subnet-7f2ddf19</b> 172.31.0.0/20 Default</p> <p>ap-southeast-2a   subnet-0ae1b82af937c072 (Private-2A) 172.31.48.0/20</p> <p>ap-southeast-2b   subnet-e3bd52ab 172.31.32.0/20 Default</p> <p>ap-southeast-2c   subnet-76f79a2e 172.31.16.0/20 Default</p> <p><input type="button" value="C"/></p> <p><b>Select subnets</b></p> <p><b>ap-southeast-2a   subnet-7f2ddf19 X</b> 172.31.0.0/20 Default</p> <p><input type="button" value="C"/></p> <p><a href="#">Create a subnet</a></p>	

- Attach the Target group created in the **Configure Advanced options**.

**Configure advanced options** Info

Choose a load balancer to distribute incoming traffic for your application across instances. You can also set options that give you more control over checking the health of instances.

**Load balancing - optional** Info

Enable load balancing

Application Load Balancer or Network Load Balancer

Classic Load Balancer

Choose a target group for your load balancer

Select target group

TG1

Create a target group [+]

**Health checks - optional**

Health check type Info

EC2 Auto Scaling automatically replaces instances that fail health checks. If you enabled load balancing, you can enable ELB health checks in addition to the EC2 health checks that are always enabled.

EC2  ELB

Health check grace period

The amount of time until EC2 Auto Scaling performs the first health check on new instances after they are put into service.

300  seconds

**Additional settings - optional**

- You can enter the Desired, Minimum and Maximum scaling capacity for the Auto Scaling group.

**Configure group size and scaling policies** Info

Set the desired, minimum, and maximum capacity of your Auto Scaling group. You can optionally add a scaling policy to dynamically scale the number of instances in the group.

**Group size - optional** Info

Specify the size of the Auto Scaling group by changing the desired capacity. You can also specify minimum and maximum capacity limits. Your desired capacity must be within the limit range.

Desired capacity

Minimum capacity

Maximum capacity

**Scaling policies - optional**

Choose whether to use a scaling policy to dynamically resize your Auto Scaling group to meet changes in demand. Info

Target tracking scaling policy  
Choose a desired outcome and leave it to the scaling policy to add and remove capacity as needed to achieve that outcome.

None

- You can add notifications, Tags if required else click Next until you get option to Create Autoscaling group.

- Once this is done, open the Auto-scaling group to view the details

The screenshot shows the AWS Auto Scaling Groups console for an Auto Scaling group named ASG1. The top navigation bar includes EC2 > Auto Scaling groups > ASG1. Below the navigation, there are tabs: Details, Activity, Automatic scaling, Instance management (which is selected), Monitoring, and Instance refresh.

### Instances (2)

Instance ID	Lifecycle	Instance type	Weighted capacity	Launch template/configuration	Availability Zone
i-06c34339...	InService	t2.micro	-	LaunchTemplate1   Version 1	ap-southeast-2a
i-0ad584ff...	InService	t2.micro	-	LaunchTemplate1   Version 1	ap-southeast-2b

### Lifecycle hooks (0)

No lifecycle hooks are currently specified.

[Create lifecycle hook](#)

### CloudWatch monitoring details

Auto Scaling | EC2

Auto Scaling group metrics collection:

Enable

All times shown are in UTC.

[View all CloudWatch metrics](#)

Add to dashboard 1h

Minimum Group Size (Count)	Maximum Group Size (Count)	Desired Capacity (Count)	In Service
1	1	1	1
No data available. Try adjusting the dashboard time range.	No data available. Try adjusting the dashboard time range.	No data available. Try adjusting the dashboard time range.	0.5
0	0	0	0

- If you go to Target groups, you can see the Targets created in the target group automatically.

The screenshot shows the AWS Lambda console with the following details:

- Function name:** HelloWorld
- Description:** A simple Lambda function that prints "Hello World" to the CloudWatch logs.
- Runtime:** Python 3.8
- Memory size:** 128 MB
- Timeout:** 3 seconds
- Code:** A ZIP file containing the Lambda function code.
- Environment:** No environment variables are defined.
- Logs:** CloudWatch Logs are selected as the log destination.
- Deployment:** The function is deployed to the us-east-1 region.
- Logs:** CloudWatch Logs are selected as the log destination.

## 4. AWS EKS Setup

- **Setup kubectl**
  - a. Download kubectl version 1.20
  - b. Grant execution permissions to kubectl executable
  - c. Move kubectl onto /usr/local/bin
  - d. Test that your kubectl installation was successful

```
curl -o kubectl https://amazon-eks.s3.us-west-2.amazonaws.com/1.19.6/2021-01-05/bin/linux/amd64/kubectl
```

```
chmod +x ./kubectl
```

```
mv ./kubectl /usr/local/bin
```

```
kubectl version --short --client
```

- **Setup eksctl**
  - a. Download and extract the latest release

- b. Move the extracted binary to /usr/local/bin
- c. Test that your eksctl installation was successful

```
curl --silent --location  
"https://github.com/weaveworks/eksctl/releases/latest/downl  
oad/eksctl_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp  
  
sudo mv /tmp/eksctl /usr/local/bin
```

eksctl version

- **Create an IAM Role and attach it to EC2 instance**  
Note: create IAM user with programmatic access if your bootstrap system is outside of AWS  
IAM user should have access to  
IAM  
EC2  
VPC  
CloudFormation
- **Create your cluster and nodes**  
eksctl create cluster --name cluster-name \  
--region region-name \  
--node-type instance-type \  
--nodes-min 2 \  
--nodes-max 2 \  
--zones <AZ-1>,<AZ-2>

example:

```
eksctl create cluster --name valaxy-cluster \  
--region ap-south-1 \  
--node-type t2.small \  
--nodes-min 2 \  
--nodes-max 2
```

- **To delete the EKS cluster**  
eksctl delete cluster valaxy --region ap-south-1

- Validate your cluster using by creating by checking nodes and by creating a pod  
kubectl get nodes  
kubectl run pod tomcat --image=tomcat

## 5. AWS ROUTE53 DNS Records

It's a global service. You need to buy a domain in order to work with Route53, Go to Route53 Service & Click on register domain. Enter the domain name & check availability, Add to cart & click on continue.

Route53 can use basically:

Public domain names you own (or buy) or Private domain names that can be resolved by your instances in your VPCs.

Route53 has many features such as **Load balancing, Health checks, Routing policy like Simple, Failover, Geolocation, Latency, Weighted, Multi value.**

You pay \$0.50 per month per hosted zone.

In AWS Route53, we have many types of records. Let's talk about various records.

### 1- SOA (Start of Authority Records)

Basic SOA stores information about below things.

Name of Server that supplied the data for zone.

The administrator of that zone & current version of data file.

Eg:

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

***Route53 Name server that create SOA record: ns-2048.awsdns-64.net.***

***Email Address of Administrator: hostmaster.example.com***

## 2- NS Record (Name Server Records)

NS records is basically your name server records which are used by top level domain servers to direct traffic to content DNS server which contains the authoritative records.

*So whenever we create a hosted zone in Route53, Two types of records automatically created, one is SOA & second is NS.*

The screenshot shows the AWS Route 53 console interface. On the left, there's a sidebar with options like 'Hosted zones', 'Health checks', 'Traffic flow', 'Traffic policies', and 'Policy records'. Under 'Domains', 'gaurav.com.' is listed. The main area is titled 'Record Set Name' with a search bar and filters for 'Any Type', 'Aliases Only', and 'Weighted Only'. It displays two record sets:

Name	Type	Value	TTL
gaurav.com.	NS	ns-656.awsdns-18.net. ns-1077.awsdns-06.org. ns-1683.awsdns-18.co.uk. ns-505.awsdns-63.com.	172800
gaurav.com.	SOA	ns-656.awsdns-18.net. awsdns-hostmaster.amazon.	900

## Hosted Zone(SOA & NS Records)

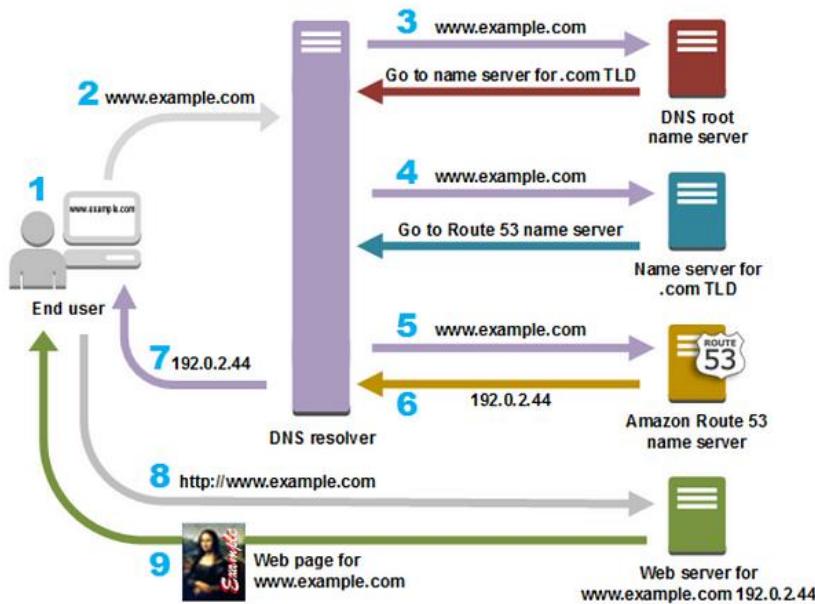
Before we head over to important type of records used in AWS, let's talk about one important concept TTL(Time to Live)

### TTL (Time to Live):

TTL is mandatory for each DNS record. So TTL is length that a DNS records is cached on either the resolving server or user own Laptop. The Lower the TTL, the faster changes to DNS records. Whenever you created record set, you need to define TTL for it.

The screenshot shows the 'Create Record Set' dialog box. It has fields for 'Name' (set to 'gaurav.com.'), 'Type' (set to 'A – IPv4 address'), 'Alias' (radio button set to 'No'), and 'TTL (Seconds)' (radio buttons for 300, 1m, 5m, 1h, 1d). The '1m' option is highlighted.

TTL(Time to Live)



How Amazon Route 53 Routes Traffic for Your Domain (Source AWS Official)

Let's talk about the most common records in AWS.

When you create basic records, you specify the following values.

Name

Type

Alias

TTL (Time to Live)

Value

Routing Policy

### 1- A Record ( URL to IPv4)

The “A” record stands for Address record. The A record is used by computer to translate the name of the domain to an IP address.

Eg: (<http://medium.com> might point to <http://126.78.98.90>)

### 2- CNAME (Canonical Records- URL to URL)

CNAME Points a URL to any other URL. (`gaurav.gupta.com` => `gkg.example.com`), We use it only for Non-Root Domain(aka. `something.mydomain.com`)

**Create Record Set**

**Name:** example.gaurav.com.

**Type:** CNAME – Canonical name

**Alias:**  Yes  No

**TTL (Seconds):** 300 1m 5m 1h 1d

**Value:** newexample

The domain name that you want to resolve to instead of the value in the Name field.  
Example:  
www.example.com

**Routing Policy:** Simple

Resolving example.gaurav.com to newexample.gaurav.com

### 3- Alias Record:

Alias record points a URL to an AWS Resource, Alias record are used to map resource record sets in your hosted zone to Elastic Load Balancer, CloudFront or S3 Buckets websites.

**Create Record Set**

**Name:** example.gaurav.com.

**Type:** A – IPv4 address

**Alias:**  Yes  No

**Alias Target:**

You can also type

- CloudFront distributions
- Elastic Beanstalk environments
- ELB load balancers
- S3 website endpoints
- Resource records
- VPC endpoint: example.vpc.amazonaws.com
- API Gateway custom resources
- ELB Application load balancers
- ELB Classic load balancers
- ELB Network load balancers
- CloudFront distributions

No Targets Available

**Routing Policy:** Simple

Route 53 responds to queries based only on the values in this record. [Learn More](#)

**Evaluate Target Health:**  Yes  No

## Alias Record

### 4- AAAA: (URL to IPv6)

An **AAAA record** maps a domain name to the IP address (Version 6) of the computer hosting the domain. An **AAAA record** is used to find the IP address of a computer connected to the internet from a name.

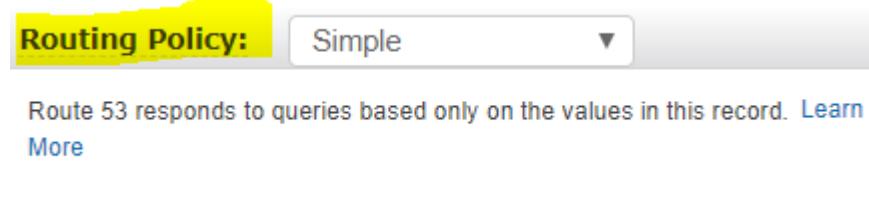
### 5- MX Record (Main Exchange Record)

A mail Exchanger **record (MX record)** specifies the mail server responsible for accepting email messages on behalf of a domain name. It is a **resource record** in the Domain Name System (**DNS**). It is

possible to configure several **MX records**, typically pointing to an array of mail servers for load balancing and redundancy.

This is all about various type records, let's talk about Routing policy. So what is routing policy.

***When you create a record, you choose a routing policy, which determines how Amazon Route 53 responds to queries.***



## 6. Routing Policy

There is total 6 types of routing policy in Route53, let's talk about one by one.

### 1- Simple Routing Policy:

In case of simple routing policy, you can have only one record with multiple IP addresses. If you specify multiple values in record, Route53 returns all values in random order to the user.

Maps a domain to one URL, Use when you need to redirect to a single resource. You can't attach health checks to simple routing policy. If multiple values are returned, a random one is chosen by the client.

**Create Record Set**

**Name:** example.gaurav.com.

**Type:** A – IPv4 address

**Alias:**  Yes  No

**TTL (Seconds):** 300  1m  5m  1h  1d

**Value:**

```
52.34.56.78
76.45.67.89
```

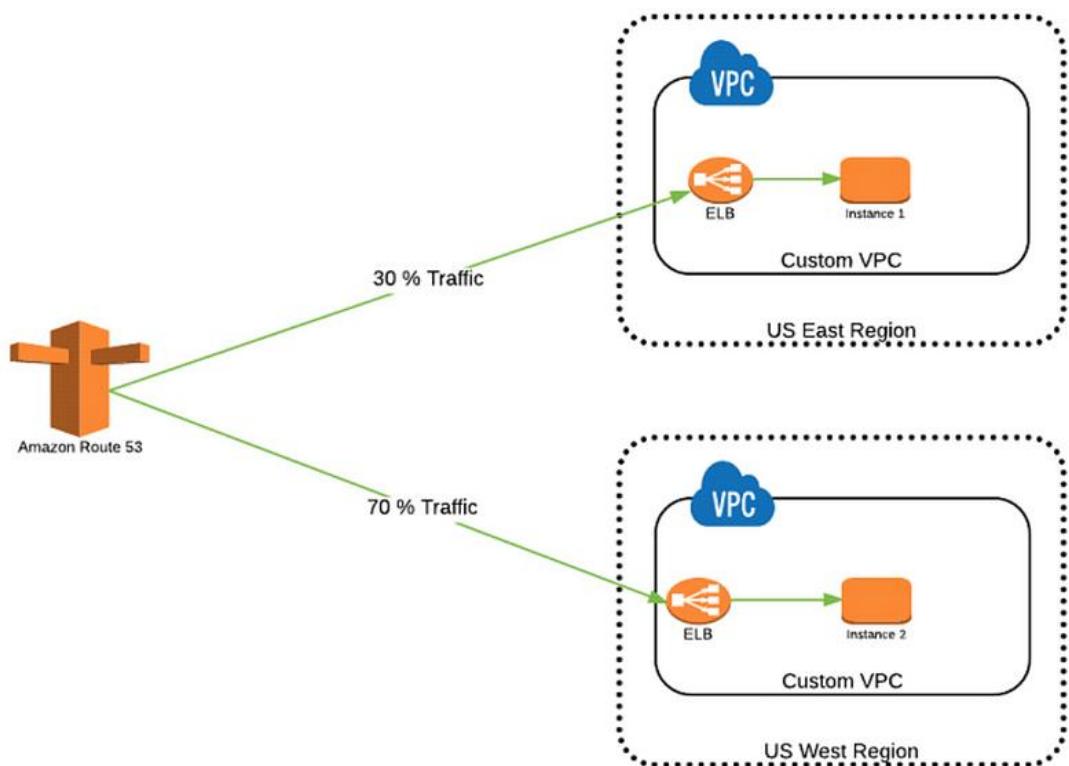
IPv4 address. Enter multiple addresses  
on separate lines.  
Example:  
192.0.2.235  
198.51.100.234

**Routing Policy:** Simple

## Simple Routing Policy

## 2- Weighted Routing Policy

Weighted Routing Policy controls the what percentage % of the requests that go to specific endpoint. It's helpful to test 1% of traffic on new app version. It is also helpful to split traffic between two regions. We can associate Health checks with it.



## Weighted Policy

**Name:** example.gaurav.com.

**Type:** A – IPv4 address

**Alias:**  Yes  No

**TTL (Seconds):** 300

**Value:**

```
52.34.56.78
76.45.67.89
```

IPv4 address. Enter multiple addresses  
on separate lines.  
Example:  
192.0.2.235  
198.51.100.234

**Routing Policy:** Weighted

Route 53 responds to queries based on weighting that you specify in this and other record sets that have the same name and type. [Learn More](#)

**Weight:** 30

**Set ID:** 1st Set

Description of this record set that is unique  
within the group of weighted sets.  
Example:  
My Seattle Data Center

**Associate with Health Check:**  Yes  No

**Create**

### 3- Latency Routing Policy

It allows you to route your traffic based on lowest network latency for your end user. It redirects to the server that has the least latency close to us also helpful when latency of users is a priority. Latency is evaluated in terms of user to designated AWS Region. For example: Germany may be directed to the US (if that's the lowest latency)

Create Record Set

**Name:** example.gaurav.com.

**Type:** A – IPv4 address ▾

**Alias:**  Yes  No

**TTL (Seconds):** 300 1m 5m 1h 1d

**Value:** 52.34.56.78

IPv4 address. Enter multiple addresses on separate lines.  
Example:  
192.0.2.235  
198.51.100.234

**Routing Policy:** Latency ▾

Route 53 responds to queries based on regions that you specify in this and other record sets that have the same name and type. [Learn More](#)

**Region:** us-west-2 ▾

**Set ID:**

Description of this record set that is unique within the group of latency sets.  
Example:  
My Seattle Data Center

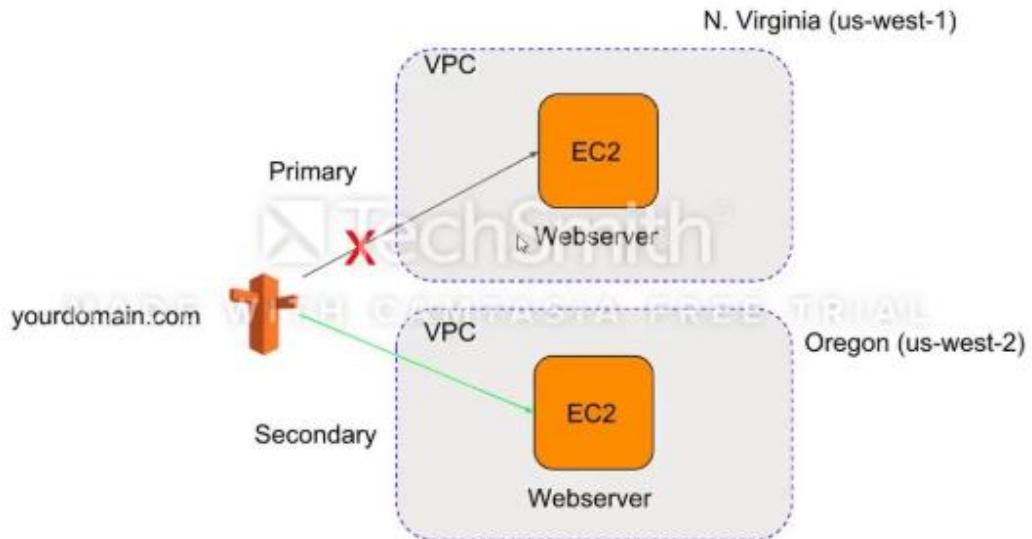
**Associate with Health Check:**  Yes  No

**Create**

#### 4- Failover Routing Policy

**Failover routing** lets you **route** traffic to a resource when the resource is healthy or to a different resource when the first resource is unhealthy. The primary and secondary records can **route** traffic to anything from an Amazon S3 bucket that is configured as a website to a complex tree of records.

You can associate health check with this type of policy.



**Create Record Set**

**Name:** example.gaurav.com.

**Type:** A – IPv4 address

**Alias:**  Yes  No

**TTL (Seconds):** 300  1m  5m  1h  1d

**Value:** 52.34.56.78

IPv4 address. Enter multiple addresses on separate lines.  
Example:  
192.0.2.235  
198.51.100.234

**Routing Policy:** Failover

Route 53 responds to queries using primary record sets if any are healthy, or using secondary record sets otherwise. [Learn More](#)

**Failover Record Type:**  Primary  Secondary

**Set ID:** example-Primary

**Associate with Health Check:**  Yes  No

**Create**

## Failover Routing Policy

### 5- Geo Location Routing Policy

**Geolocation routing** lets you choose the resources that serve your traffic based on the geographic location of your users, meaning the location that DNS queries originate from. For example, you might want all queries from Europe to be **routed** to an ELB load balancer in the Frankfurt region.

This is routing based on user location. Health check associated.

**Routing Policy:** Geolocation

Route 53 responds to queries based on the locations from which DNS queries originate. We recommend that you create a Default location resource record set [Learn More](#)

**Location:** Choose a location

**Set ID:**

Description of this record set that is unique within the group of geolocation sets.  
Example:  
Route to Seattle data center

**Associate with Health Check:**  Yes  No

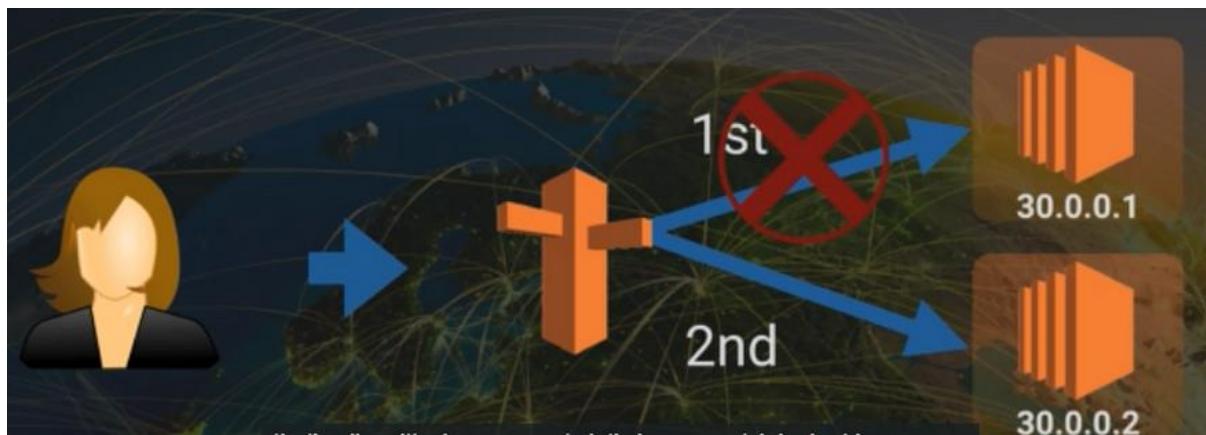
**Create**

## Geolocation Policy

### 6- Multi Value Routing Policy

It helps distribute DNS responses across **multiple** resources. For example, use **multivalue answer routing** when you want to associate your **routing** records with a **Route 53** health check.

Use multivalue answer routing when you need to return multiple values for a DNS query and route traffic to multiple IP addresses. Up to 8 healthy records are returned for each Multi Value query. Multi Value is not a substitute for having an ELB.



**Routing Policy:** Multivalue Answer

Route 53 responds to DNS queries with up to eight healthy records selected at random. [Learn More](#)

**Set ID:** Set-1

Description of this record set that is unique within the group of multivalue answer sets.

Example:  
Route to Seattle data center

**Associate with Health Check:**  Yes  No

## MultiValue Routing

### 7. Amazon Elastic Disaster Recovery

#### Definition:

Minimizes downtime, data loss with fast reliable recovery of on-premises & cloud-based applications using affordable storages, minimal compute & point-in-time recovery

#### Recovery-why?

- Disaster happened at one Availability zone, at that point of time, we have to point/route all the requests to the recovery environment where data is already replicated at the backend.
- Users will not get impacted & the new data will get routed to the new server whichever we consider it as recover server called as **Failover**(to launch recovery)

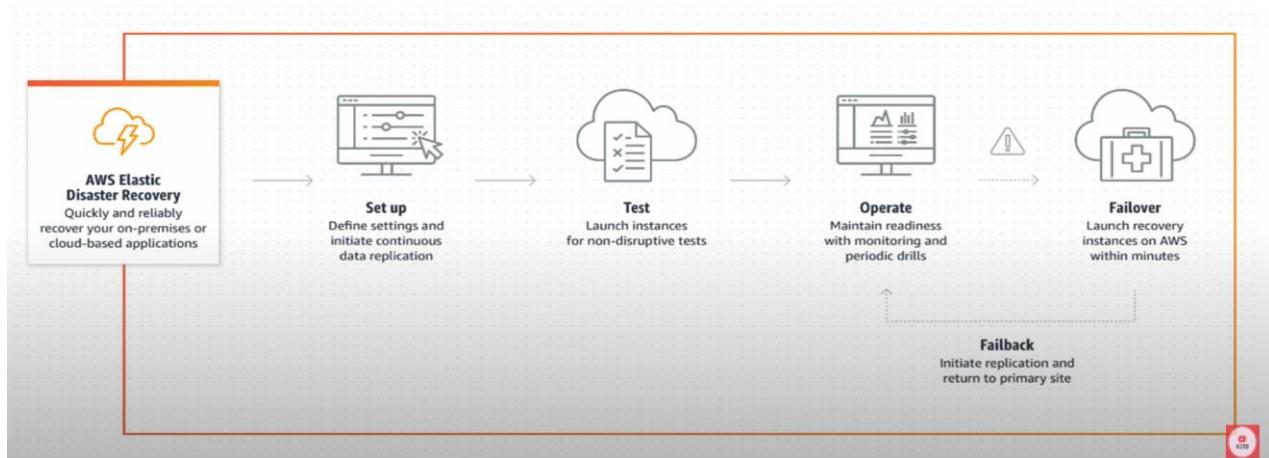
instances having previous data & new requests will be routed to new server)

- Disaster is over, **failback** from recovery server to the original source server.

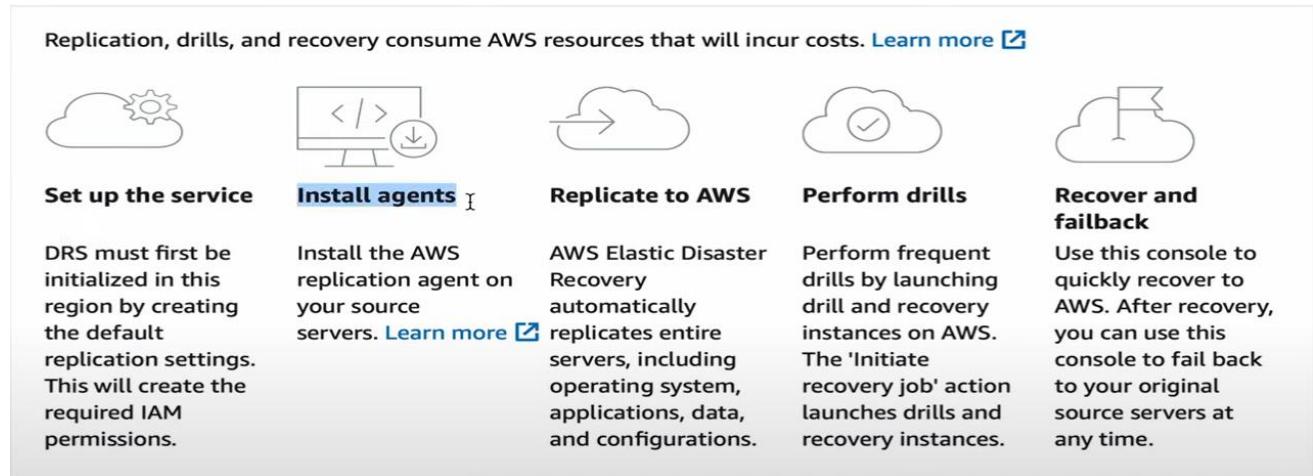
Process:

- Create an replication agent which acts as a media b/w source server & DR environment & tries to replicate(in time interval) . In the backend, it takes the snapshot of the source environment if any disaster happens.
- During Failover, the latest sanpshot can be taken for recovering our instances & source environment if any disaster happens.
- Available for Region-to-region; within a region; on-premises system

Architecture:



## How it works?



## EDR Dashboard:

The screenshot shows the AWS EDR Server info dashboard for server **s-3cb42a058b346820d**. The left sidebar includes links for Source servers, Recovery instances, Recovery job history, Settings, and Documentation. The main panel displays the following information:

Ready for recovery	Pending actions	Last recovery result	Recovery instance (fallback possible from recovery instances)
⌚	⌚	⌚	⌚

Below the table, there are tabs for Recovery dashboard, Server info, Tags, Disks settings, and Replication settings. A progress bar indicates "Loading server details".

- Server Info → RAM, Recommended Instance type, Hardware
- Tag → Name
- Disk settings → Disk name & storage
- Replication settings →
- Launch settings → Ec2 Launch Template
  - Create template
    - Size/Tier
    - Key pair
    - SG
    - Subnet
    - Storage → gp3
  - Template version → Set as “default version”

Steps:

### Step1: Settings

- Go to AWS → Services → Storage → Amazon Elastic Disaster Recovery → Settings → Edit default Application settings
- Subnet → Public/Private
- Replication server instance type → t3.micro
- Volume → SSD gp3 cheaper than gp2
- EBS Encryption → Default
- Security Group → Custom SG
- Data Routing:
  - VPN, Direct Connect, VPC peering
  - Public IP creation
- Point in Time → Snapshot retention(1-365)

### Step2: IAM Role setup

- Role setup → IAM → Role → Add or Create Role
  - Entity → EC2
  - Tag → <name>
- Policy/Permission:  
AWSElasticDisasterRecoveryRecoveryInstancePolicy
- Attach the role to your user
- Assign the role to the running instances → Actions → Security → Modify IAM Role → Enter Role & Save it

### Step3: Install Agent

- Install Agent → Pull the code/file to CLI using wget (refer Amazon documentation)
- Execute the file as root user
  - Enter AWS region
  - Enter Access Key & Secret Key [IAM → Users → Security credentials]

- Choose the disks to replicate → Enter to replicate all
- Once this is done, replication agent will be downloaded & installed, then in AWS EDR dashboard, we can see the hostname with source servers Private IP
- Data replication will be initiated
- We will have one new instance as “**AWS Disaster Recovery Replication Server**” along with source instance.

#### Step4: Recovery Job

- Once replication is done & snapshot is created(Source server is ready for recover) and the same can be viewed in EDR Dashboard

The screenshot shows the AWS EBS Snapshots page. On the left, there's a sidebar with options like Instances, AMIs, and Elastic Block Store. Under EBS, the 'Solutions' section is expanded, showing 'Snapshots'. The main area displays a table of snapshots:

Name	Snapshot ID	Size	Description
AWS Elastic Disaster Recovery Snapshot	snap-0a4a5ab64646b2d8c	8 GiB	AWS Elastic Disaster Recov...
AWS Elastic Disaster Recovery Snapshot	snap-098e43d30b7a1f372	8 GiB	AWS Elastic Disaster Recov...
AWS Elastic Disaster Recovery Snapshot	snap-01d5d9825c54f3e5c	8 GiB	AWS Elastic Disaster Recov...
AWS Elastic Disaster Recovery Base Snapshot	snap-01a77fedaff175e18	1 GiB	AWS Elastic Disaster Recov...

Below the table, a message says "Select a snapshot above."

- Select Hostname → Initiate Recovery job → Initiate recovery

The screenshot shows the AWS EDR Source servers page. The left sidebar has 'Source servers' selected. The main area shows a single server entry:

**s-3cb42a058b346820d**

The server status is "Ready for recovery". Below the status, there are tabs for "Recovery dashboard", "Server info", "Tags", "Disks settings", and "Replication settings". A progress bar at the bottom indicates "Loading server details".

- Select the required PIT
- Initiate recovery

- This will create a conversion server in EC2 instances as “**AWS Disaster Recovery Conversion Server**”
- This is not the replicates server
- This will stop & terminated once the replicated server instance is created/initiated

The screenshots show the AWS EC2 Instances page. In the first screenshot, three instances are listed: 'awselasticDR' (Instance ID i-08552749787e88226, t2.micro), 'AWS Elastic Disaster Recovery Replication Server' (Instance ID i-0c5064e08a99f6a34, t3.small), and 'AWS Elastic Disaster Recovery Conversion Server' (Instance ID i-0a17da0e8711bf32b, m4.large). All three instances are shown as 'Running'. In the second screenshot, a filter for 'Instance state = running' is applied, and the 'Conversion Server' instance is highlighted with a blue selection bar.

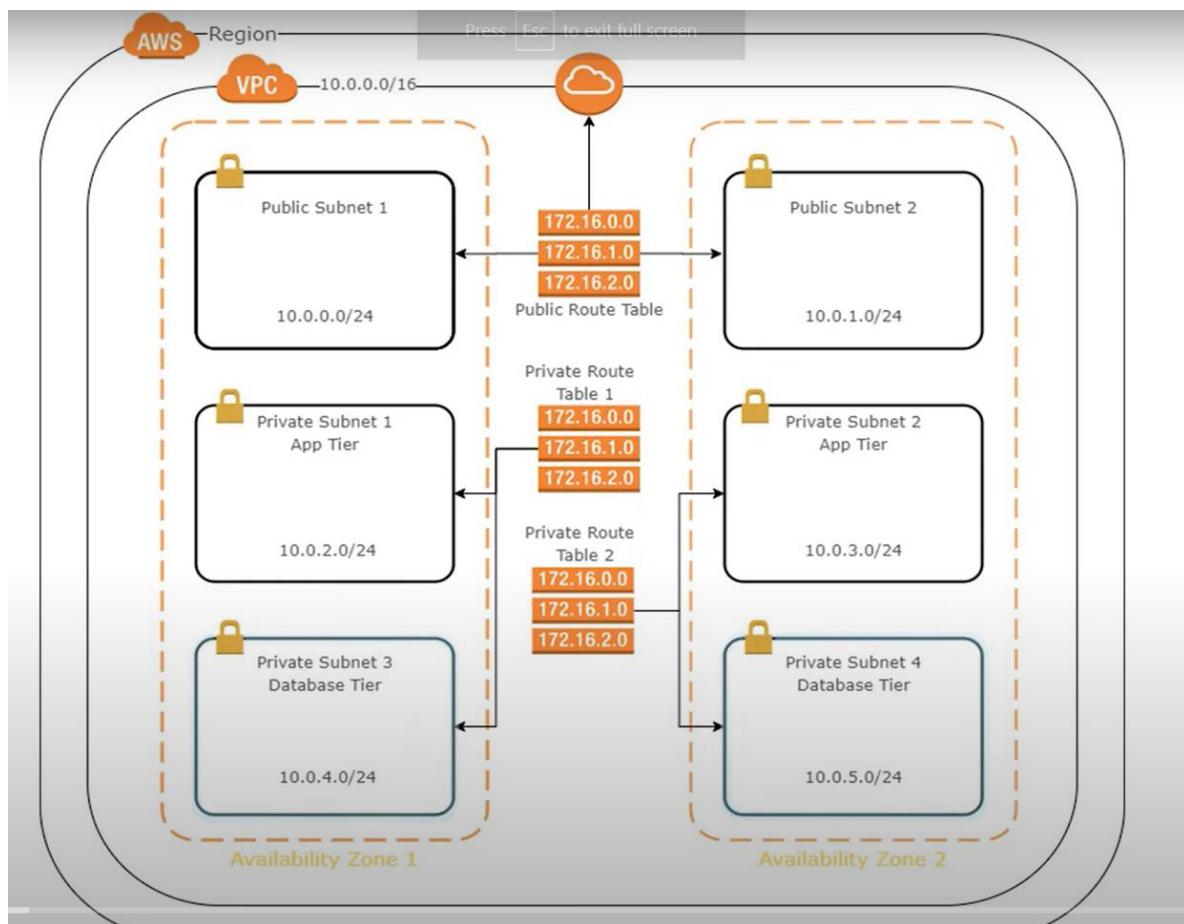
## Step5: Failback

- Install agents in the replicated server
- Go to Recovery instances → Select → Failback

The screenshot shows the AWS Elastic Disaster Recovery console under the 'Recovery instances' tab. It displays a single recovery instance. Below the instance details, there is a prominent orange 'Failback' button.

## 8. VPC

Architecture:



Process:

Steps:

Step1: Create VPC

- Select region
- Services → Networking & Content delivery → VPC
- Create VPC
- Enter CIDR block

VPCs > Create VPC

### Create VPC

A VPC is an isolated portion of the AWS cloud populated by AWS objects, such as Amazon EC2 instances. You must specify an IPv4 address range for your VPC. Specify the IPv4 address range as a Classless Inter-Domain Routing (CIDR) block; for example, 10.0.0.0/16. You cannot specify an IPv4 CIDR block larger than /16. You can optionally associate an IPv6 CIDR block with the VPC.

Name tag	Demo VPC	<small>i</small>
IPv4 CIDR block*	10.0.0.0/16	<small>i</small>
IPv6 CIDR block	<input checked="" type="radio"/> No IPv6 CIDR Block <input type="radio"/> Amazon provided IPv6 CIDR block <input type="radio"/> IPv6 CIDR owned by me	<small>i</small>
Tenancy	Default	<small>i</small>

\* Required

Cancel Create

- Create
- The system will automatically create Route table which is private

### Step2: Create Internet Gateway and attach it to VPC

- Select Internet Gateway → Create
- Give name “Demo IGW”
- Close
- Then Select created gateway → Actions → Attach to VPC
- Enter the VPC
- Click “Attach”

Internet gateways > Attach to VPC

### Attach to VPC

Attach an internet gateway to a VPC to enable communication with the internet. Specify the VPC you would like to attach below.

VPC*	Select a VPC	<small>i</small>
------	--------------	------------------

AWS Command Line Interface command

\* Required

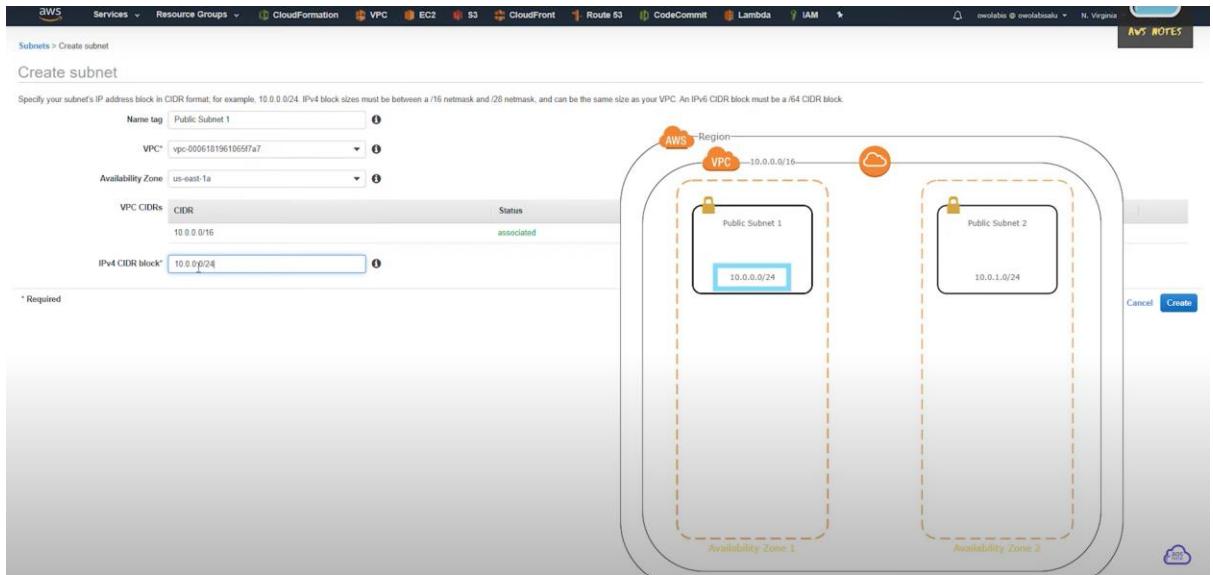
Cancel Attach

Create internet gateway Actions

	Name	ID	State	VPC	Owner
<input checked="" type="checkbox"/>	Demo IGW	igw-0a032580f2d...	attached	vpc-00061819610...	960957692776
<input type="checkbox"/>	igw-e468059f		attached	vpc-12122c68	960957692776

### Step3: Create Subnets in Different Availability Zones

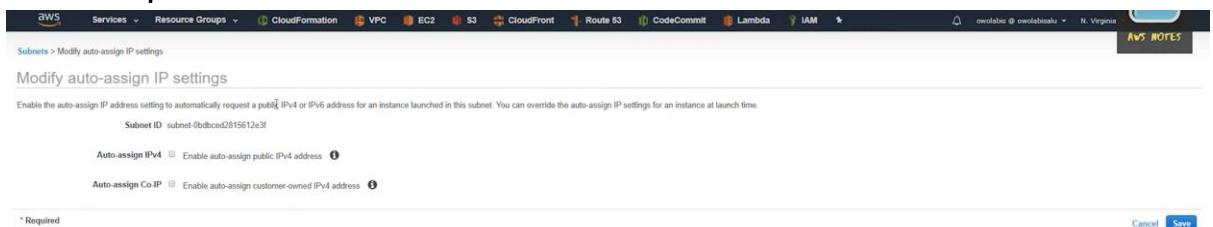
- Click on “Subnets” in Left Pane
- Fill the details like required VPC, Availability zone, Name of subnet(Public subnet1), CIDR block



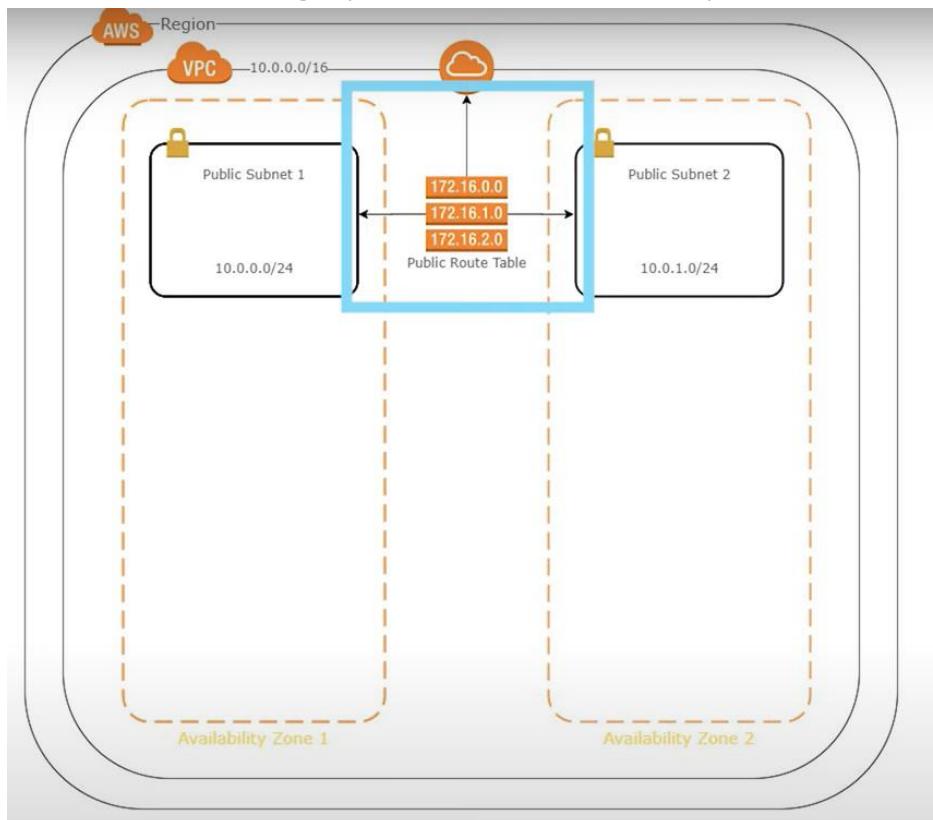
- Repeat the previous step to create another subnet (“Public subnet2”) and change the Availability Zone alone.



- To assign IP automatically to this subnet, select the Subnet → Action → Modify auto-assign IP settings & check-in **Enable IPv4**



#### Step4: Create and Assign public route table to public subnet 1 & 2



- Select “Route table” in the Left pane of AWS console and select Create
- Enter details like Name & Enter VPC “Demo VPC” & **Create**

Route Tables > Create route table

### Create route table

A route table specifies how packets are forwarded between the subnets within your VPC, the internet, and your VPN connection.

Name tag	Public Route Table	?
VPC*	vpc-0006181961065f7a7	I C ?

\* Required

Cancel **Create**

- Select created Route Table → Routes
- Enter Route as “0.0.0.0/16” and Target as “Internet Gateway” and **Save Routes**

Route Tables > Edit routes

### Edit routes

Destination	Target	Status	Propagated
10.0.0/16	local	active	No
0.0.0.0/0	igw-0a0032580f2d84bbe		No

**Add route**

\* Required Cancel **Save routes** 

- To associate the subnets, Select Route table → Subnet Associations → Edit

**Create route table** Actions ▾

Filter by tags and attributes or search by keyword

Name	Route Table ID	Explicit subnet association	Edge associations	Main	VPC ID	Owner
rtb-0174ad7c38754403c	rtb-0174ad7c38754403c	-	-	Yes	vpc-0006181961065ff7a7   ...	960957692776
<b>Public Route Table</b>	<b>rtb-03f178b1a31935128</b>	-	-	No	vpc-0006181961065ff7a7   ...	960957692776

Summary Routes Subnet Associations Edge Associations Route Propagation Tags

**Edit subnet associations**

Subnet ID IPv4 CIDR IPv6 CIDR

None found

- Check-in the created Subnets and click Save

Route Tables > Edit subnet associations

### Edit subnet associations

Route table rtb-03f178b1a31935128 (Public Route Table)

Associated subnets **subnet-0b5393e6fd9dc4bb7** **subnet-0bdbced2815612e3f**

Subnet ID	IPv4 CIDR	IPv6 CIDR	Current Route Table
subnet-0bdbced2815612e3f   Public Sub...	10.0.0.0/24	-	Main
subnet-0b5393e6fd9dc4bb7   Public Sub...	10.0.1.0/24	-	Main

\* Required Cancel **Save** 

- Create the two private subnets by following the same steps, but it doesn't have any Internet gateway associated with it

Screenshot of the AWS VPC Subnets > Create subnet interface. The form fields are:

- Name tag: Private Subnet 1 | App Tier
- VPC\*: vpc-0006181961965f7a7
- Availability Zone: us-east-1a
- VPC CIDs: CIDR 10.0.0.0/16 Status: associated
- IPv4 CIDR block\*: 10.0.0.0/24

The diagram shows a VPC with two Availability Zones (AZ1 and AZ2). AZ1 contains Public Subnet 1 (10.0.0.0/24) and Private Subnet 1 (App Tier) (10.0.2.0/24). AZ2 contains Public Subnet 2 (10.0.1.0/24), Private Subnet 2 (App Tier) (10.0.3.0/24), and Private Subnet 3 (Database Tier) (10.0.4.0/24). A Public Route Table routes traffic from the subnets to the internet.

Screenshot of the AWS VPC Subnets > Create subnet interface. The form fields are:

- Name tag: Private Subnet 2 | App Tier
- VPC\*: vpc-0006181961965f7a7
- Availability Zone: No preference
- VPC CIDs: CIDR 10.0.0.0/16 Status: associated
- IPv4 CIDR block\*:

The diagram shows a VPC with two Availability Zones (AZ1 and AZ2). AZ1 contains Public Subnet 1 (10.0.0.0/24) and Private Subnet 1 (App Tier) (10.0.2.0/24). AZ2 contains Public Subnet 2 (10.0.1.0/24), Private Subnet 2 (App Tier) (10.0.3.0/24), and Private Subnet 3 (Database Tier) (10.0.4.0/24). A Public Route Table routes traffic from the subnets to the internet.

**Create subnet**

Specify your subnet's IP address block in CIDR format; for example, 10.0.0.0/24. IPv4 block sizes must be between a /16 netmask and /26 netmask, and can be the same size as your VPC. An IPv6 CIDR block must be a /64 CIDR block.

Name tag	Private Subnet 3   Database Tier
VPC*	vpc-0006181961065f7a7
Availability Zone	us-east-1a
VPC CIDRs	CIDR: 10.0.0.0/16 Status: associated
IPv4 CIDR block*	10.0.4.0/28

\* Required

**Create subnet**

Specify your subnet's IP address block in CIDR format; for example, 10.0.0.0/24. IPv4 block sizes must be between a /16 netmask and /26 netmask, and can be the same size as your VPC. An IPv6 CIDR block must be a /64 CIDR block.

Name tag	Private Subnet 4   Database Tier
VPC*	vpc-0006181961065f7a7
Availability Zone	us-east-1b
VPC CIDRs	CIDR: 10.0.0.0/16 Status: associated
IPv4 CIDR block*	10.0.5.0/24

\* Required

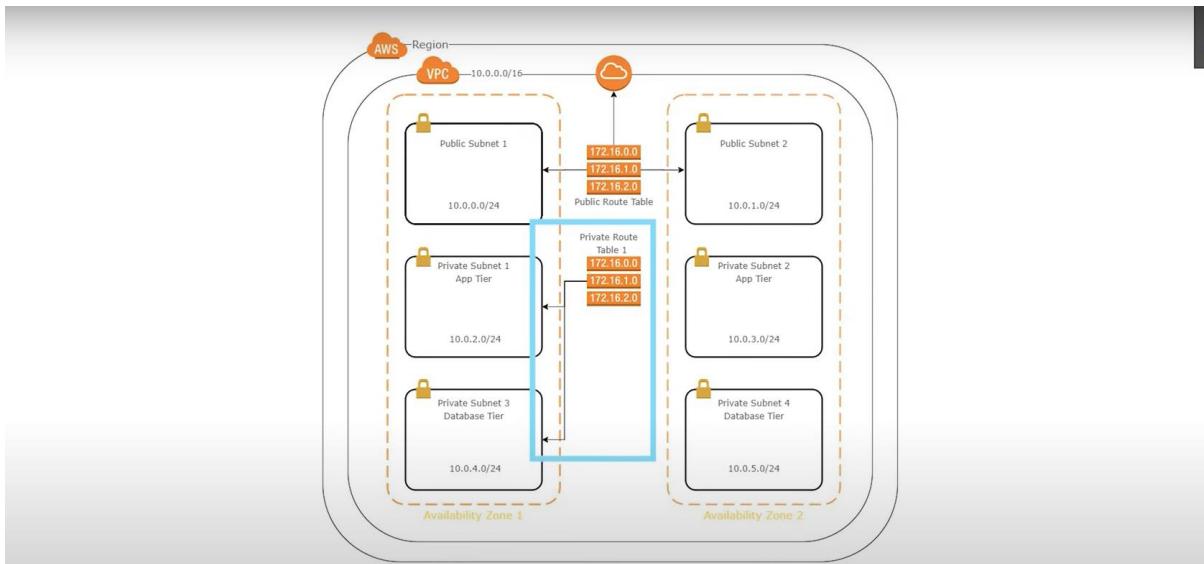
## After Subnet creation:

**Create subnet Actions ▾**

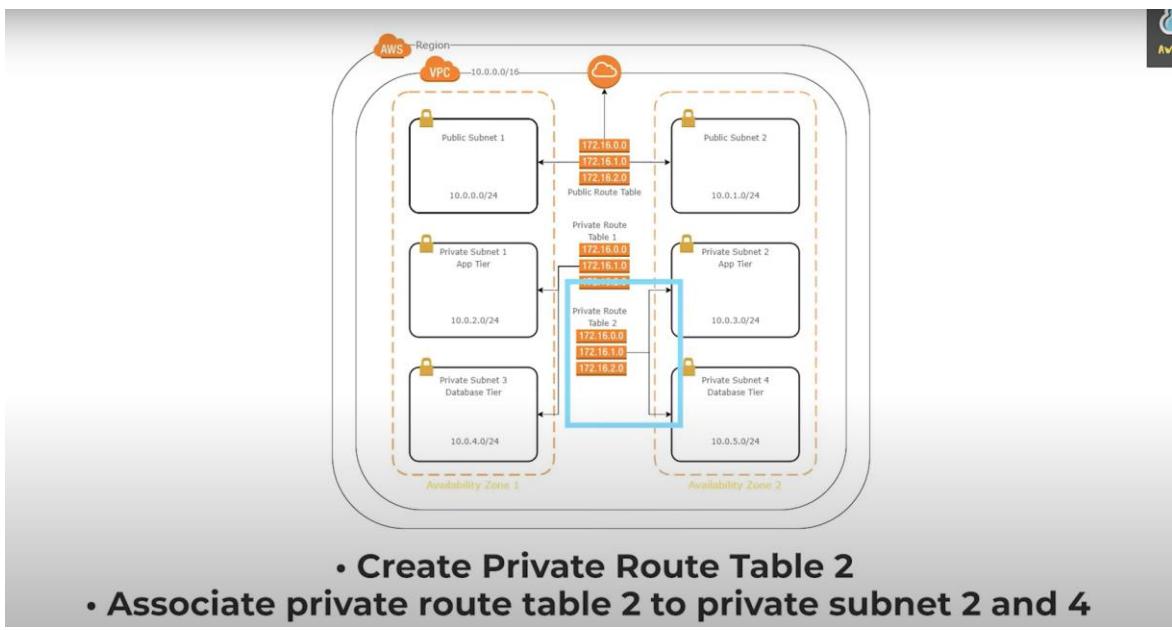
Filter by tags and attributes or search by keyword

Name	Subnet ID	State	VPC	IPv4 CIDR	Available IPv4	IPv6 CIDR	Availability Zone	Availability Zone ID	Route table
Public Subnet 1	subnet-0bdbced2815612e3f	available	vpc-0006181961065f7a7   Demo ...	10.0.0.0/24	251	-	us-east-1a	use1-az4	rtb-03f178b1a31935128   Publi...
Public Subnet 2	subnet-0b5393e6f49dc4bb7	available	vpc-0006181961065f7a7   Demo ...	10.0.1.0/24	251	-	us-east-1b	use1-az6	rtb-03f178b1a31935128   Publi...
Private Subnet 1   App Tier	subnet-0350e43373270712	available	vpc-0006181961065f7a7   Demo ...	10.0.2.0/24	251	-	us-east-1a	use1-az4	rtb-0174adfc38754403c
Private Subnet 2   App Tier	subnet-0d60a0d2696a2c45	available	vpc-0006181961065f7a7   Demo ...	10.0.3.0/24	251	-	us-east-1b	use1-az6	rtb-0174adfc38754403c
Private Subnet 3   Database Tier	subnet-0044dac448709531	available	vpc-0006181961065f7a7   Demo ...	10.0.4.0/24	251	-	us-east-1a	use1-az4	rtb-0174adfc38754403c
Private Subnet 4   Database Tier	subnet-073ac77ba7df90910	available	vpc-0006181961065f7a7   Demo ...	10.0.5.0/24	251	-	us-east-1b	use1-az6	rtb-0174adfc38754403c

## Step5: Create Two Private Route tables and attach it to 4 subnets



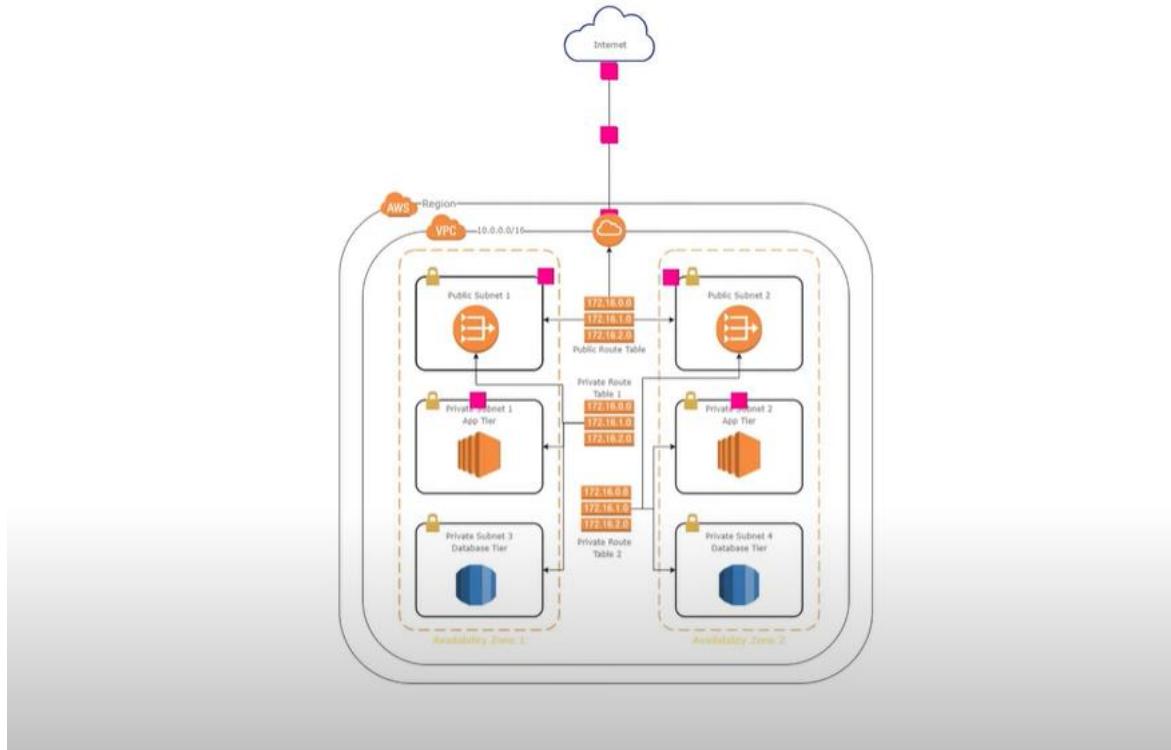
- Create Private Route Table 1
- Associate private route table 1 to private subnet 1 and 3



- Create Private Route Table 2
- Associate private route table 2 to private subnet 2 and 4

- Follow the step 4 till the Subnet Associations

## Step6: Create two Elastic IP's



- Select Services → Networking & Content Delivery → VPC
- Click on “Elastic IP’s” in the Left pane of AWS dashboard
- Click “Allocate Elastic IP address”

VPC > Elastic IP addresses > Allocate Elastic IP address

### Allocate Elastic IP address

Allocate an Elastic IP address by selecting the public IPv4 address pool from which the public IP address is to be allocated. You can have one Elastic IP (EIP) address associated with a running instance at no charge. If you associate additional EIPs with that instance, you will be charged for each additional EIP associated with that instance on a pro rata basis. Additional EIPs are only available in Amazon VPC. To ensure efficient use of Elastic IP addresses, we impose a small hourly charge when these IP addresses are not associated with a running instance or when they are associated with a stopped instance or unattached network interface. [Learn more](#)

**Elastic IP address settings**

Public IPv4 address pool  
Public IP addresses are allocated from Amazon's pool of public IP addresses, from a pool that you own and bring to your account, or from a pool that you own and continue to advertise.

Amazon's pool of IPv4 addresses

Public IPv4 address that you bring to your AWS account (option disabled because no pools found) [Learn more](#)

Customer owned pool of IPv4 addresses (option disabled because no customer owned pools found) [Learn more](#)

[Cancel](#) [Allocate](#)

- Click “Allocate”
- Repeat the same step to create another Elastic IP

Elastic IP addresses (2)										
Name	Allocated IPv4 address	Type	Allocation ID	Associated instance ID	Private IP address	Association ID	Network interface			
-	3.85.181.61	Public IP	eipalloc-0e10042c1ba65c997	-	-	-	-			
-	54.165.99.81	Public IP	eipalloc-093b224aa7b44c61f	-	-	-	-			

## Step7: Create NAT Gateway

- Select “NAT Gateway” from LP and click **Create**
- Enter the details like Public subnet tag and Elastic IP for NAT gateway & click **Create**

NAT Gateways > Create NAT Gateway

Create NAT Gateway

Create a NAT gateway and assign it an Elastic IP address. Learn more.

Subnet: subnet-0bdbcd2015612e3

Elastic IP Allocation ID: eipalloc-0e10042c1ba65c997

Key: Value:

Add Tag: 50 remaining (Up to 50 tags maximum)

\* Required

Create a NAT Gateway

Your NAT gateway has been created.

Note: In order to use your NAT gateway, ensure that you edit your route tables to include a route with the following NAT gateway. Find out more.

NAT Gateway ID: nat-0d9ebdd2136c5a3f

Edit route tables Close

**After you've created a NAT gateway, you must update the route table associated with the private subnets to point internet-bound traffic to the NAT gateway.**

- Click “Edit Routes”
- Select the Private Route table 1 and click “Routes” & Edit

- Enter the Open IPv4 and Target as “NAT Gateway” and click Save

The screenshot shows the AWS VPC Management console with the 'Edit routes' page open. A new route is being added to a route table. The 'Destination' field contains '0.0.0.0/0'. The 'Target' dropdown menu is open, showing 'nat-0d9ebddcb2136c5a3f' as the selected option. The 'Status' is set to 'active' and 'Propagated' is set to 'No'. At the bottom right, there are 'Cancel' and 'Save routes' buttons, with 'Save routes' being the active button.

Destination	Target	Status	Propagated
10.0.0.0/16	local	active	No
0.0.0.0/0	nat-0d9ebddcb2136c5a3f		No

- Repeat the same steps to create another NAT gateway and assign it to Private Route Table 2.