

AWS CONCEPTS AND SETUPS



1

Table of Contents

1.	AWS ELB	6
	What is AWS ELB?	6
	Types of Load Balancer in AWS	6
	a. Classic Load Balancers.....	6
	b. Application Load Balancers.....	7
	Benefits of Application Load balancers-	8
	c. Network Load Balancers.....	8
	Benefits of Network Load Balancers-.....	9
	Conclusion.....	9
2.	AWS ELB setup	10
	Application Load Balancer setup:	10
	Steps:.....	11
	Step1: Create Two instances: Orders & Payments	11
	Step2: Target Group creation	11
	Step3: Load Balancer Creation.....	13
3.	AWS Scaling-group setup.....	18
	Working:.....	18
	Launch Templates:.....	18
	Steps:.....	19
	Step1: Create Image of an instance	19
	Step2: Create Autoscaling group using Launch template.....	20
4.	AWS EKS Setup.....	25
	• Setup kubectl	25
	• Setup eksctl.....	25
	• Create an IAM Role and attache it to EC2 instance	26
	• Create your cluster and nodes.....	26
	• To delete the EKS cluser	26
	• Validate your cluster using by creating by checking nodes and by creating a pod	27
5.	AWS ROUTE53 DNS Records	27
	1- SOA (Start of Authority Records)	27
	2- NS Record (Name Server Records)	28
	1- A Record (URL to IPv4).....	29
	2- CNAME (Canonical Records- URL to URL).....	29

3- Alias Record:	30
4- AAAA: (URL to IPv6)	31
5- MX Record (Main Exchange Record)	31
6. Routing Policy.....	32
1- Simple Routing Policy:	32
2- Weighted Routing Policy.....	33
3- Latency Routing Policy	35
4- Failover Routing Policy.....	36
5- Geo Location Routing Policy	38
6- Multi Value Routing Policy.....	39
7. Amazon Elastic Disaster Recovery	40
Definition:	40
Recovery-why?.....	40
Process:	41
Architecture:	41
How it works?	42
EDR Dashboard:	42
Steps:.....	43
Step1: Settings	43
Step2: IAM Role setup	43
Step3: Install Agent	43
Step4: Recovery Job	44
Step5: Failback	45
8. VPC	46
Architecture:	46
Process:	46
Steps:.....	46
Step1: Create VPC	46
Step2: Create Internet Gateway and attach it to VPC	47
Step3: Create Subnets in Different Availability Zones	47
Step4: Create and Assign public route table to public subnet 1 & 2	49
Step5: Create Two Private Route tables and attach it to 4 subnets	53
Step6: Create two Elastic IP's.....	54
Step7: Create NAT Gateway.....	55
9. AWS 3-Tier Architecture with VPC, RDS, ELB:.....	56
Steps required for the 3-tier Setup.....	57

Steps:.....	57
Step1: Create VPC	57
Step2: Create Subnets.....	58
Step3: Setting up the Internet Gateway	59
Step4: Create Route Table & Subnet Associations	60
Step4.1: Create Two Private Route tables and attach it to 4 subnets.....	64
Step5: Create the Elastic IP's.....	65
Step5.1: Create NAT Gateway.....	66
Step6: Launch Template	67
Step7: Create Database	70
Step8: Enable connection between the 3 Tiers	74
10. AWS CloudWatch Setup.....	75
Four Pillars:	75
How CloudWatch Works?	76
Implementation of Four Pillars:	77
CloudWatch Alarms:	78
Design- Autoscaling with CloudWatch Alarms:.....	80
Create Alarm with SNS notification for CPU utilization:.....	80
Steps:.....	80
Create Alarm with Autoscaling Group for CPU Utilization	83
Steps:.....	83
Step1: Update Dynamic scaling policy in Auto-scaling group.....	83
Step2: Create CloudWatch Alarm	84
11. Amazon Elastic Container Service(ECS creation)	88
Steps:.....	88
12. AWS DEVOPS.....	92
Code Commit:	92
Steps:.....	92
Step1: Create a New Repository	92
Step2: Create a new user	92
Step3: Clone repository	94
Step4: Push to CodeCommit Repository.....	95
Code Build :	97
Steps:.....	97
Step1: Create Code Build Project.....	97
Code Deploy	101

Steps:.....	102
Step1: Create IAM Role & Ec2 instances.....	102
Step2: Code Deploy Application	103
Step3: Code Deploy: Deployment group creation	103
Code Pipeline	105
Steps:.....	105
Step1: Create Code Pipeline	105
Step2: Source Stage	106
Step3: Build Stage	106
Step4: Deploy Stage	107

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:

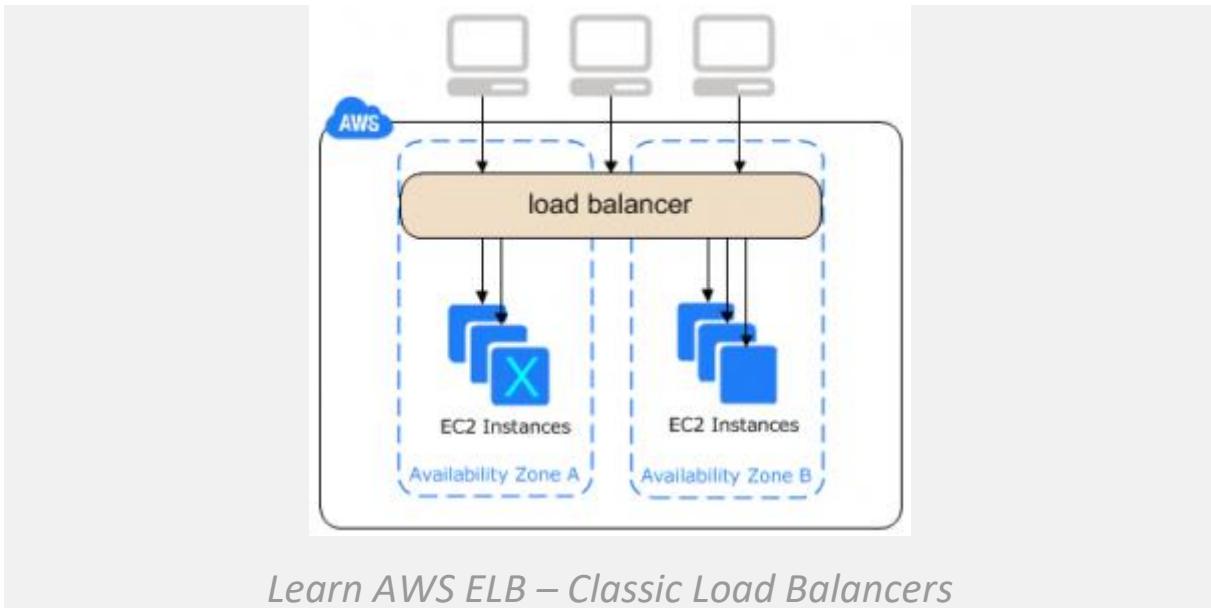
a. Classic Load Balancers

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.

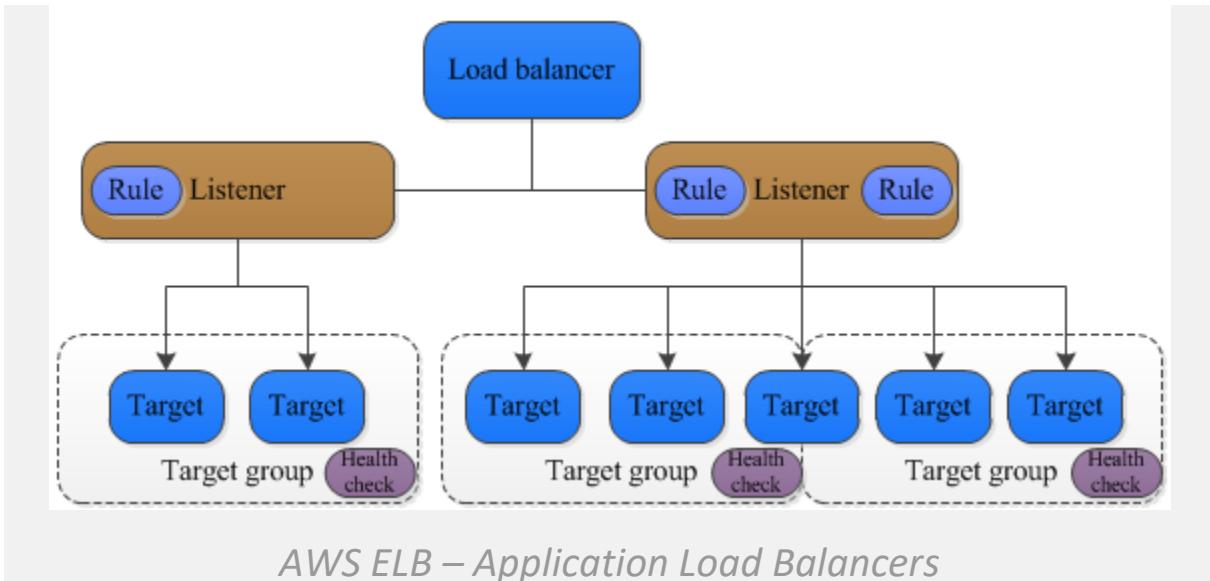
b. Application Load Balancers

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

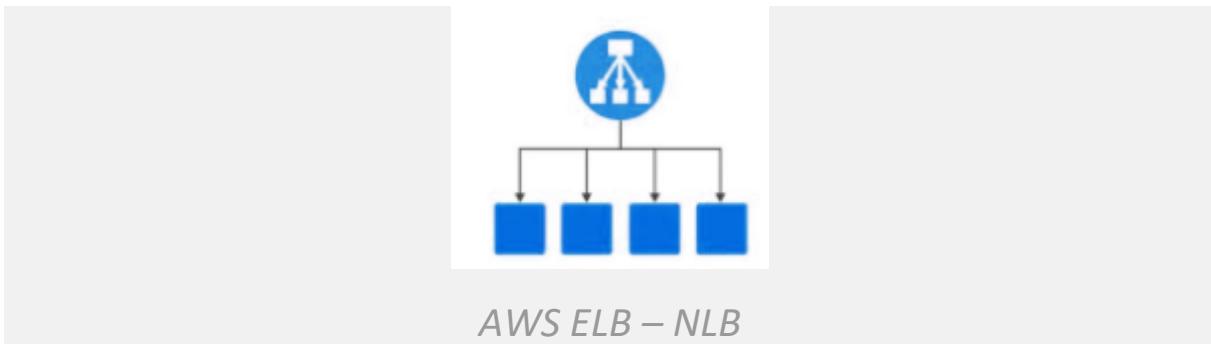
It is the fourth layer of the Open System Interconnection Model. After the load balancer receives a connection request, it selects a target from the group which targets for the default rule.

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.

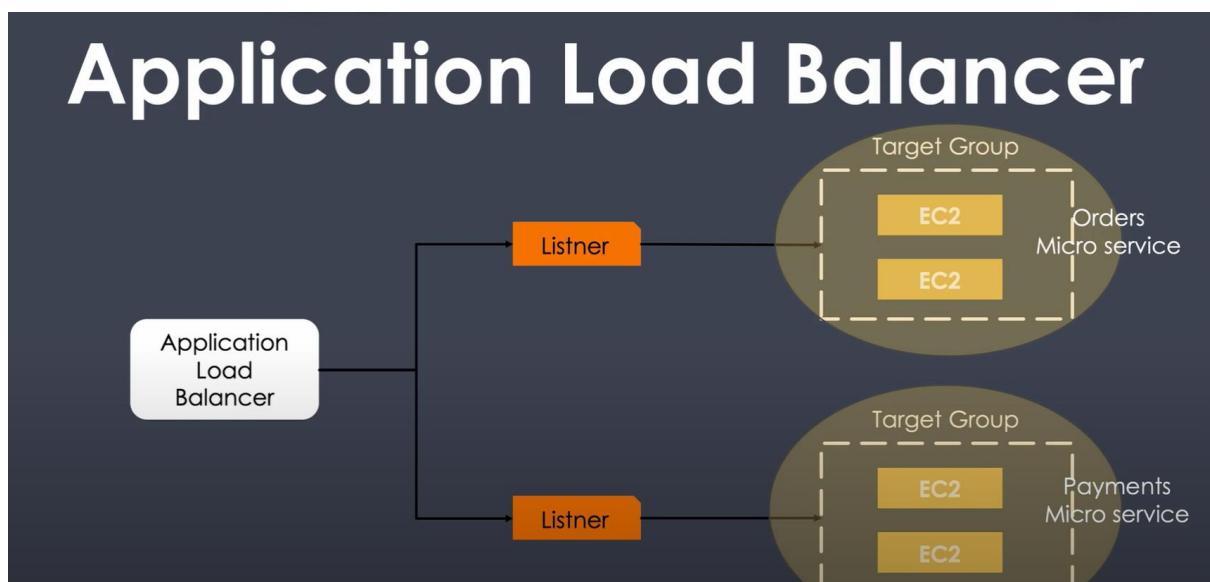
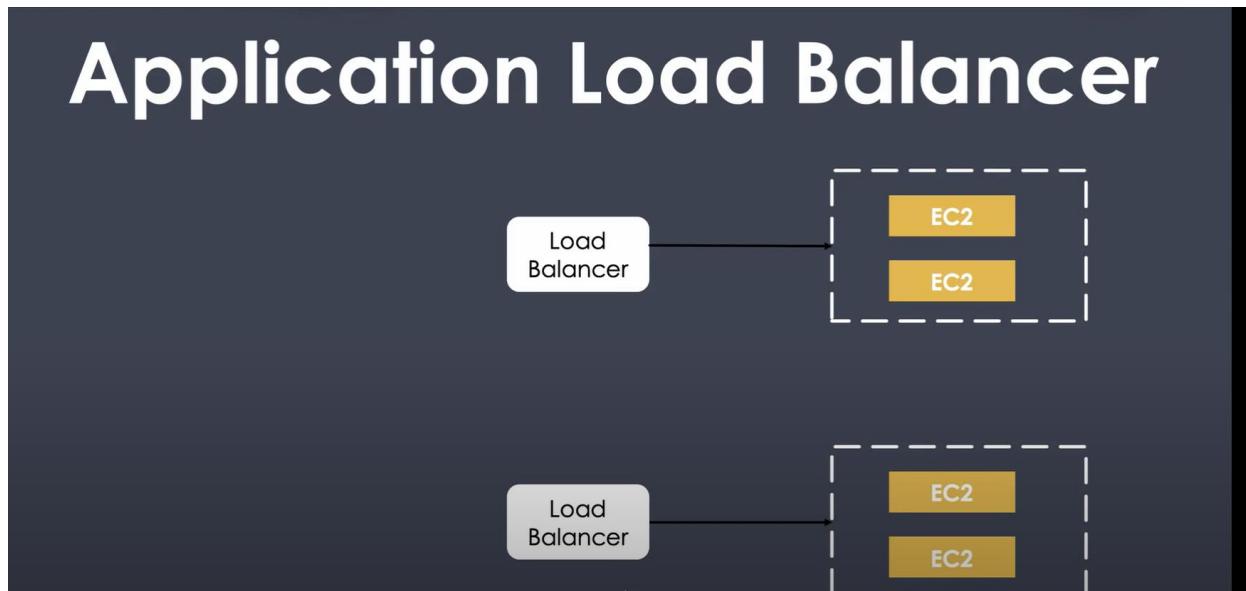
Conclusion

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)"/>
Health check settings	
Protocol	<input type="text" value="HTTP"/>
Path	<input type="text" value="/orders/index.html"/>
Advanced health check settings	

- 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	running	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	running	apache-sg	ap-south-1a	subnet-6bb4ec03	172.31.32.0/20
<input type="checkbox"/>	i-037498db731...	Payments	running	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.

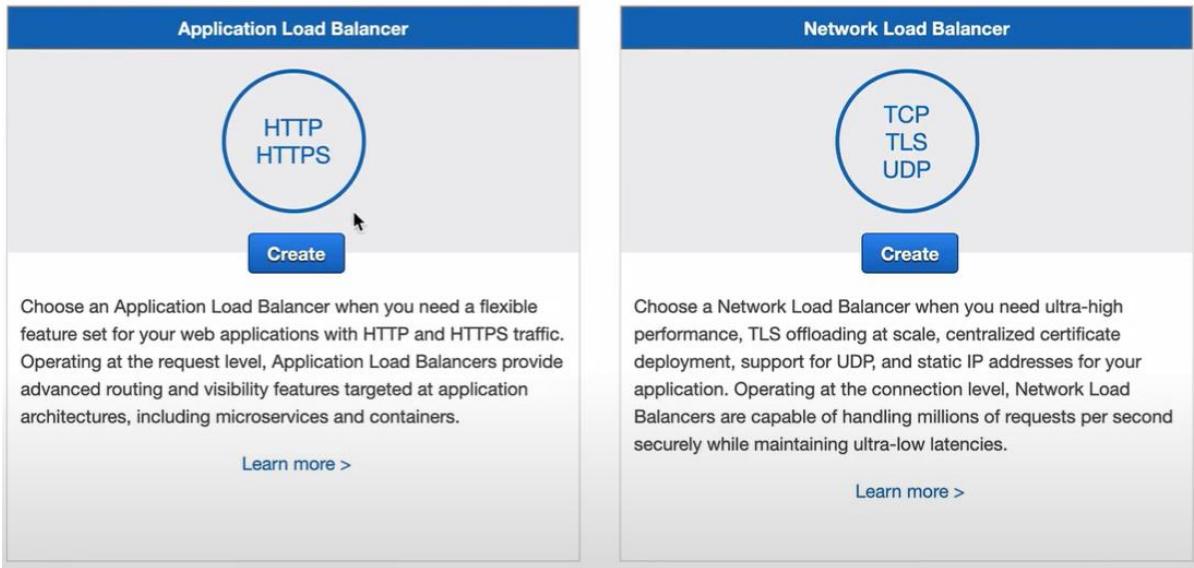
Edit

Registered targets

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

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-1c	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 orders arn...606a5dc843f41d83 ▾ View/edit rules

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

AWS Services Resource Groups ▾

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". It displays two rules:

- Rule 1:** IF (all match) Path... is /payments/* or Value THEN Forward to orders
- Rule 2:** IF Path is /orders* THEN Forward to payments

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

The screenshot shows the AWS Lambda@Edge Target Groups page. It lists two target groups:

Name	Port	Protocol	Target type	Load Balancer	VPC ID	Monitoring
orders	80	HTTP	instance	javahome-alb	vpc-2397a44b	Enabled
payments	80	HTTP	instance	javahome-alb	vpc-2397a44b	Disabled

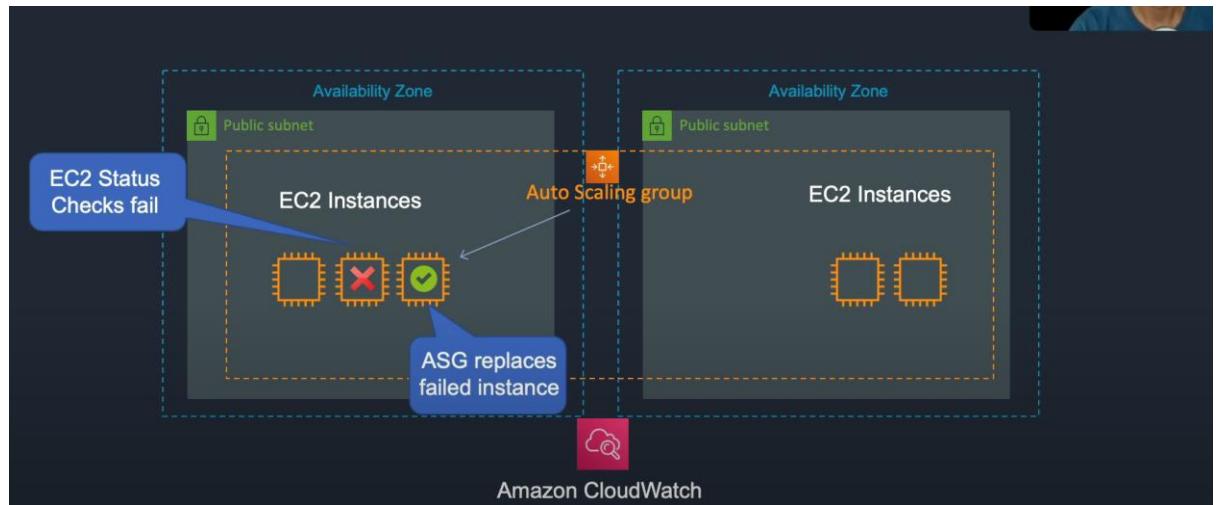
Under the 'targets' tab for the 'orders' target group, it shows:

- Registered targets: One target named 'Orders' (Instance ID: i-08858ebdac7b4dbe9, Port: 80, Availability Zone: ap-south-1a, Status: healthy)
- Availability Zones: One availability zone (ap-south-1a) with a target count of 1 and a healthy status.

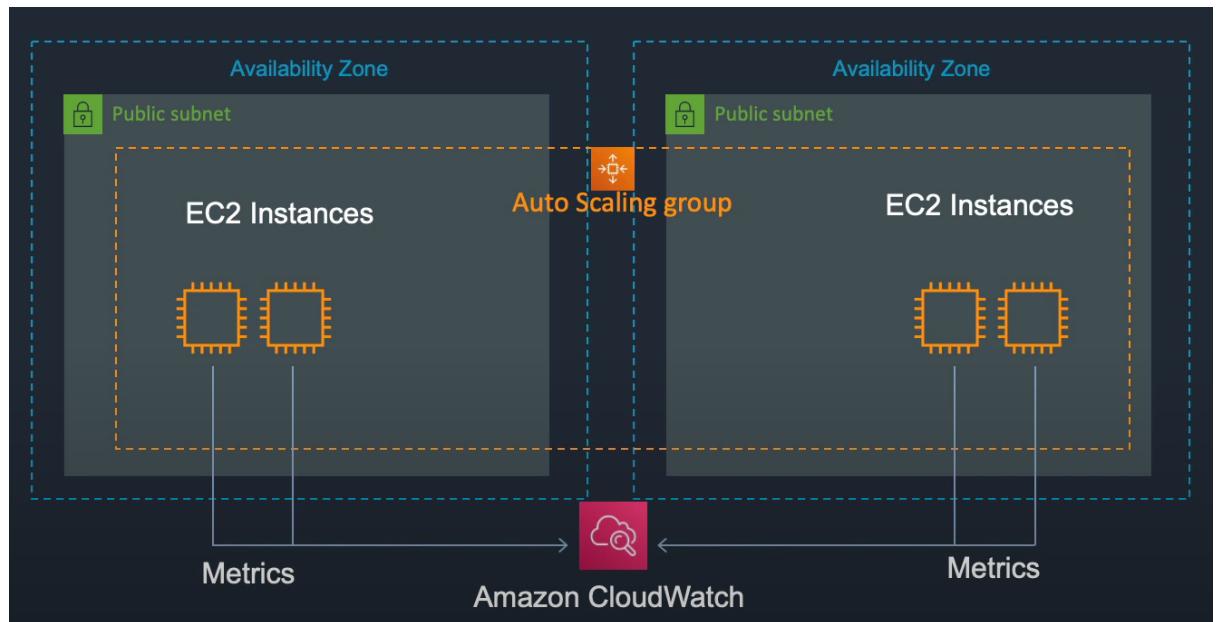
3. AWS Scaling-group setup

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

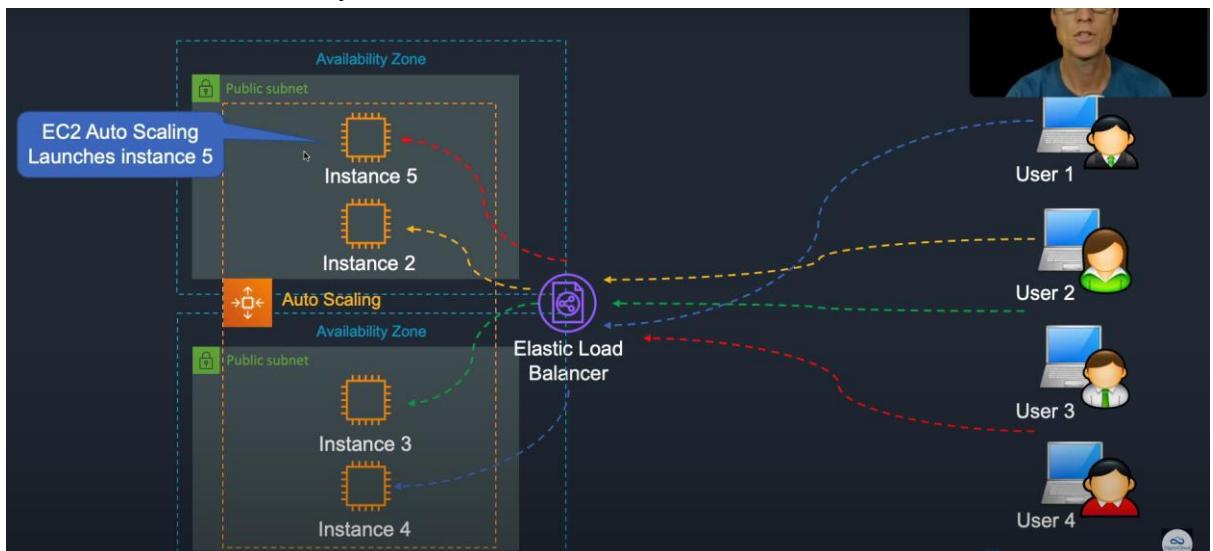
Similar to a Launch Configuration but offers some additional features:

- 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 AWS EC2 'Create Auto Scaling group' wizard. The current step is 'Step 1: Choose launch template or configuration'. On the left, a sidebar lists steps 1 through 7. Step 1 is 'Choose launch template or configuration' (selected). Step 2 is 'Configure settings'. Step 3 (optional) is 'Configure advanced options'. Step 4 (optional) is 'Configure group size and scaling policies'. Step 5 (optional) is 'Add notifications'. Step 6 (optional) is 'Add tags'. Step 7 is 'Review'. The main area displays the 'Choose launch template or configuration' section. It includes a note: '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.' Below this is a 'Name' input field labeled 'Auto Scaling group name' with a placeholder 'Enter a name to identify the group.' A note below the field states: 'Must be unique to this account in the current Region and no more than 255 characters.' To the right of the name field is a 'Launch template' section with a dropdown menu labeled 'Select a launch template' and a 'Create a launch template' button. 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

Please select an AMI

Instance type [Info](#)

Instance type

t2.micro	Free tier eligible	Instance types
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

Sydney-KP

[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](#)

Select security groups

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. ASG1
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.		
LaunchTemplate1		<input type="button" value="C"/>
Create a launch template		
Version		
Default (1)		<input type="button" value="C"/>
Create a launch template version		
Description	Launch template LaunchTemplate1 lt-0e367a8cd1016e2d4	Instance type t2.micro
AMI ID	Security groups -	Security group IDs sg-0cc5dec3fb96b0ab1

- 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>Adhere to launch template</p> <p>The launch template determines the purchase option (On-Demand or Spot) and instance type.</p> <p>Combine purchase options and instance types</p> <p>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.</p> <p>Select subnets</p> <p>ap-southeast-2a subnet-7f2ddf19 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>Select subnets</p> <p>ap-southeast-2a subnet-7f2ddf19 X 172.31.0.0/20 Default</p> <p>Create a subnet</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 the 'ASG1' group. The 'Instance management' tab is selected. The 'Instances (2)' section lists two instances: 'i-06c34339...' and 'i-0ad584ff...', both in 'InService' state and running 't2.micro' instances. The 'Lifecycle hooks (0)' section indicates 'No lifecycle hooks are currently specified' and has a 'Create lifecycle hook' button.

The 'CloudWatch monitoring details' section shows metrics collection settings. The 'Auto Scaling' tab is selected. It displays 'Auto Scaling group metrics collection:' with an 'Enable' checkbox, a note that all times are in UTC, and a link to 'View all CloudWatch metrics'. It also includes four horizontal bar charts for 'Minimum Group Size (Count)', 'Maximum Group Size (Count)', 'Desired Capacity (Count)', and 'In Service' status, each showing a value of 1 and a note 'No data available. Try adjusting the dashboard time range.' Below the charts is a timeline from 00:00 to 02:00.

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

The screenshot shows the AWS Elastic Load Balancing Target Groups console. The top navigation bar includes 'EC2 > Target groups > TG1'. Below the navigation is the target group name 'TG1' and its ARN: arn:aws:elasticloadbalancing:ap-southeast-2:778642078716:targetgroup/TG1/7e0a2d0fe06f348d. A 'Basic configuration' section displays 'Target type: instance', 'Protocol: Port', 'HTTP : 80', 'VPC: vpc-49202a2e', and 'Load bal: myALB'. Below this are tabs for 'Group details', 'Targets' (which is selected), 'Monitoring', and 'Tags'. The 'Registered targets' section shows two instances: 'i-0ad584ff814512c4a' and 'i-06c34339afff8f28d', both listed as healthy with port 80 and zone ap-southeast-2b/2a.

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' and shows two entries:

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)

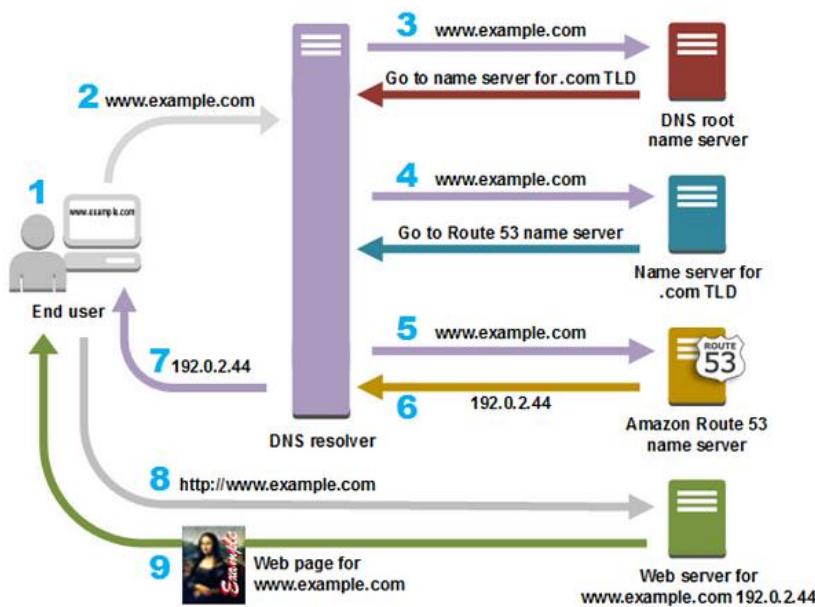
Before we head over to important type of records used in AWS, let's talk about one important concept TTL(Time to Live)

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' (gaurav.com.), 'Type' (A – IPv4 address), 'Alias' (No selected), and 'TTL (Seconds)' (set to 300). There are also buttons for 1m, 5m, 1h, and 1d.

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

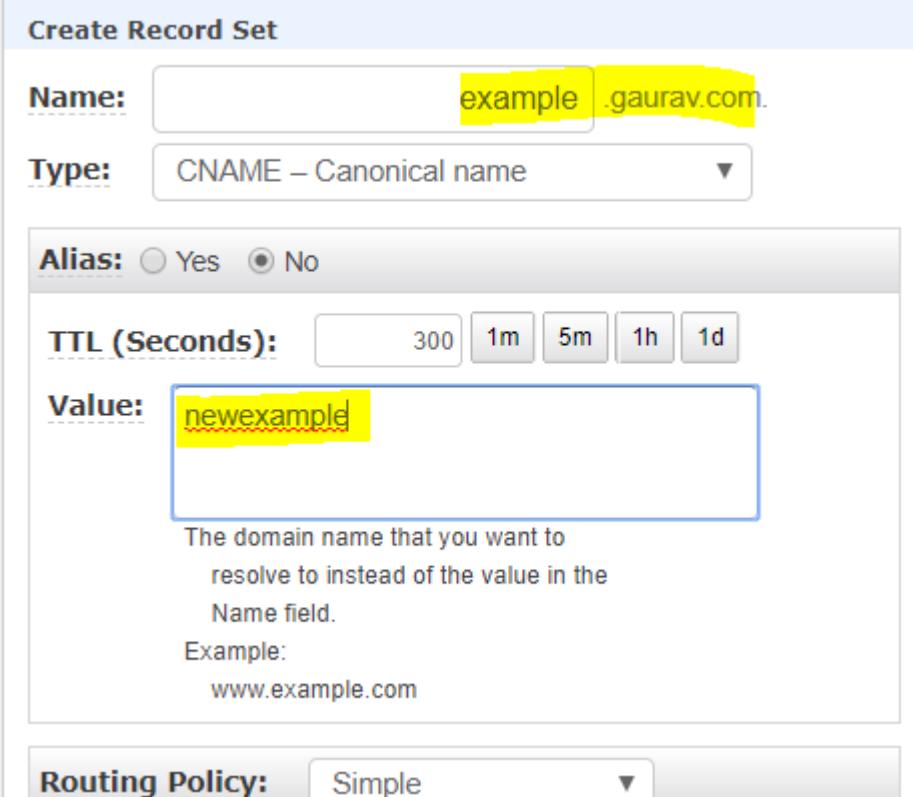
Alias: Yes No

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
```

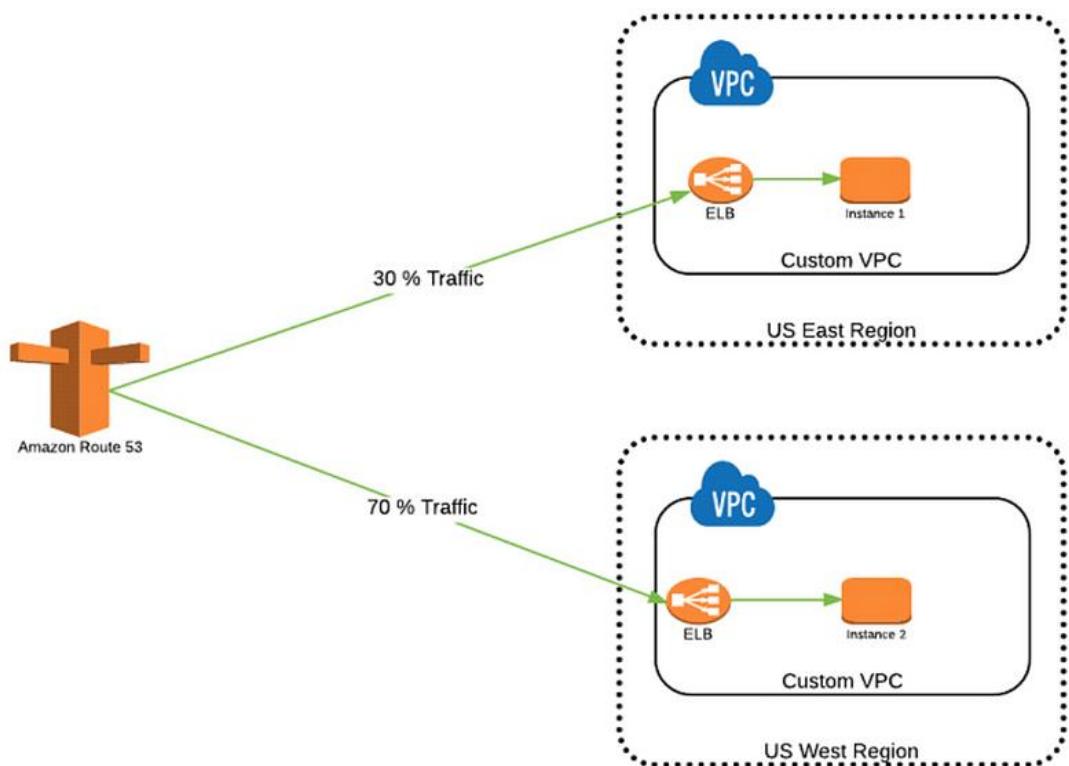
IPv4 address. Enter multiple addresses
on separate lines.
Example:
192.0.2.235
198.51.100.234

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

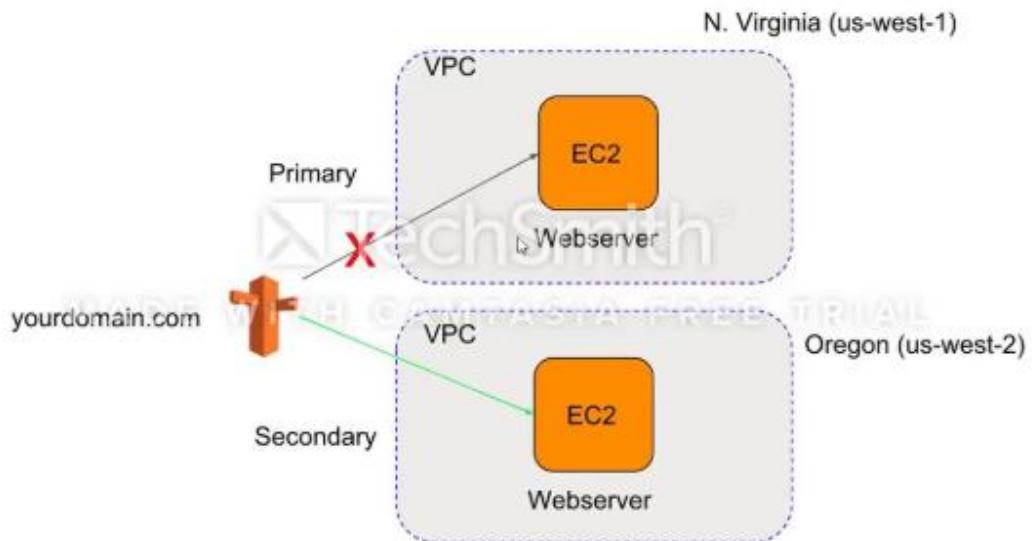
Associate with Health Check: Yes No

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)

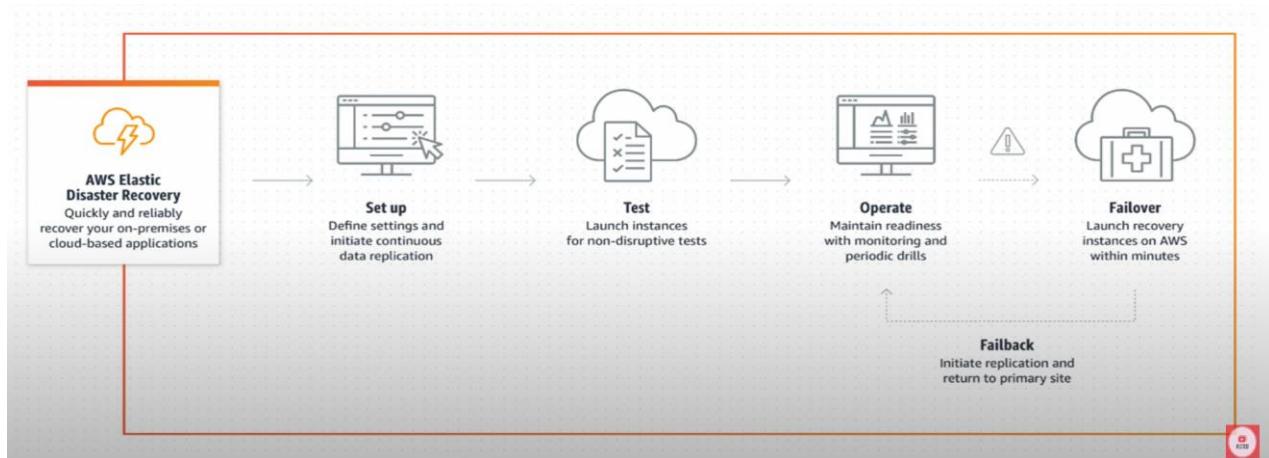
instances having previous data & new requests will be routed to new server)

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

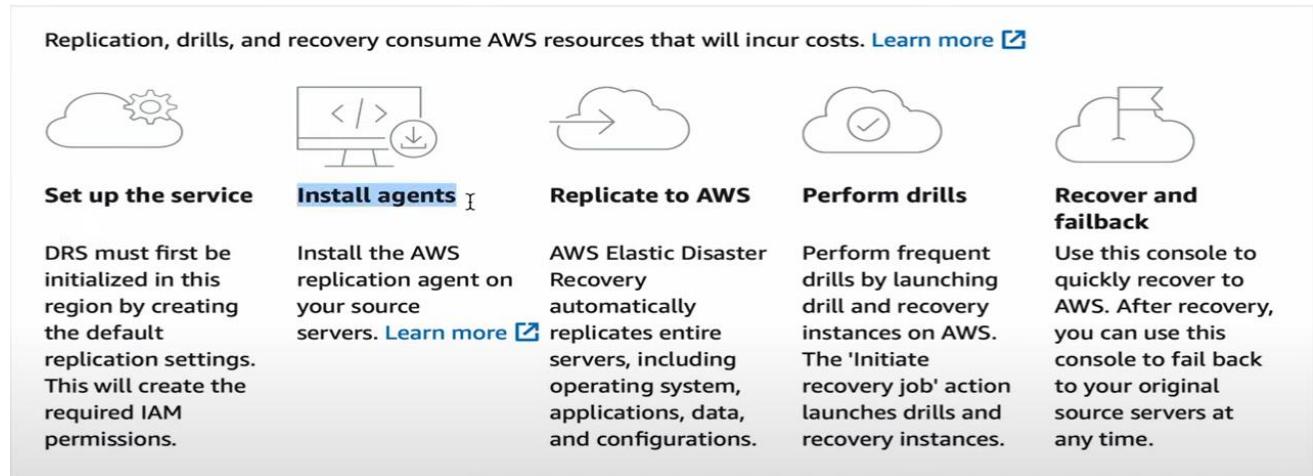
Process:

- 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 main view displays the following details:

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

Below the main view, there are tabs for **Recovery dashboard**, **Server info**, **Tags**, **Disk 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, with options like Recovery instances, Recovery job history, and Settings. The main area shows a server named "s-3cb42a058b346820d". The top right has "Actions" and "Initiate recovery job". Below that is an "Overview" section with tabs for "Info" and "Actions". The "Info" tab shows the following status indicators:

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

Below the overview 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, there are three running instances listed:

Name	Instance ID	Instance state	Instance type
awselasticDR	i-08552749787e88226	Running	t2.micro
AWS Elastic Disaster Recovery Replication Server	i-0c5064e08a99f6a34	Running	t3.small
AWS Elastic Disaster Recovery Conversion Server	i-0a17da0e8711bf32b	Running	m4.large

In the second screenshot, the 'Conversion Server' instance has been selected, highlighted with a blue border.

Step5: Failback

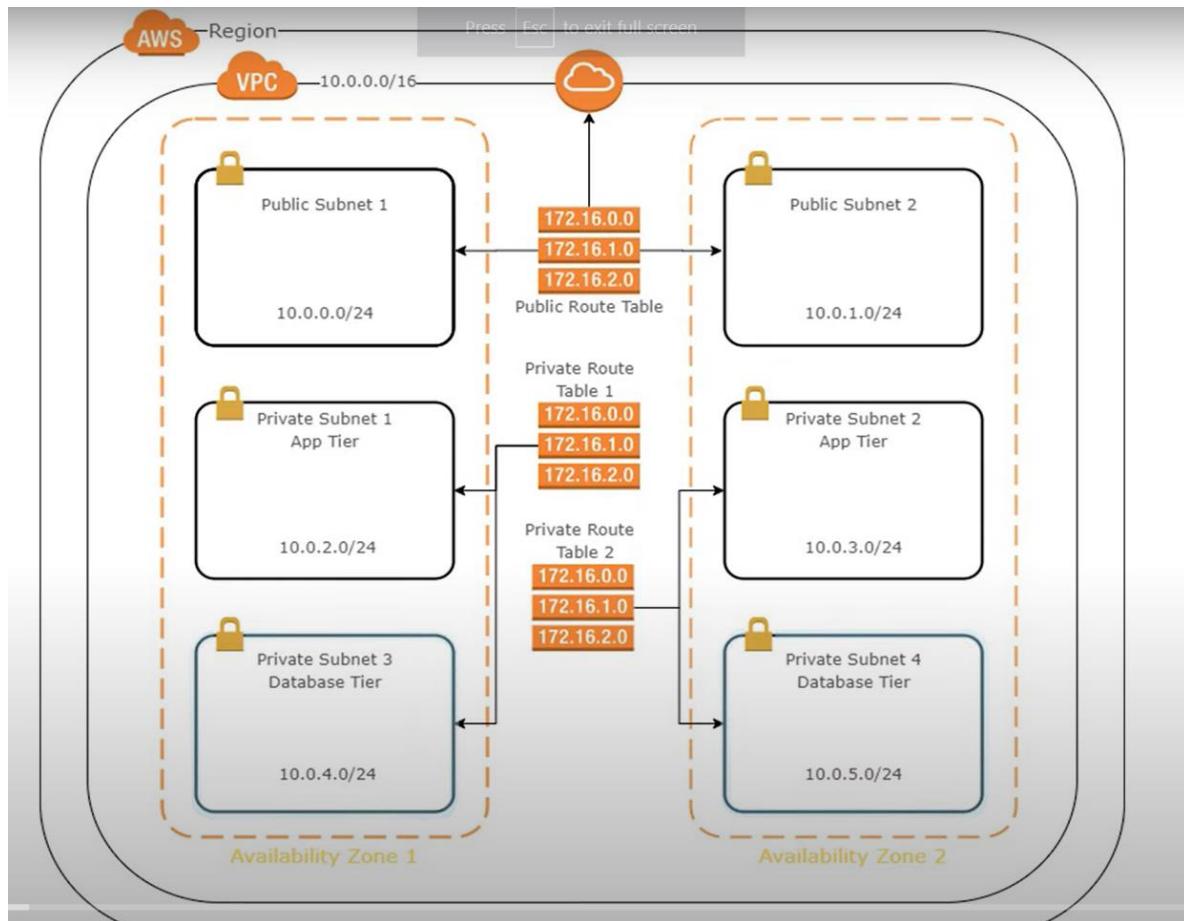
- Install agents in the replicated server
- Go to Recovery instances → Select → Failback

The screenshot shows the AWS Elastic Disaster Recovery console under the 'Recovery instances' section. It displays one recovery instance:

Actions	Failback
<input checked="" type="checkbox"/> Instance ID	
Fallback state	
Data replication status	
Additional details	
Source serv	

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

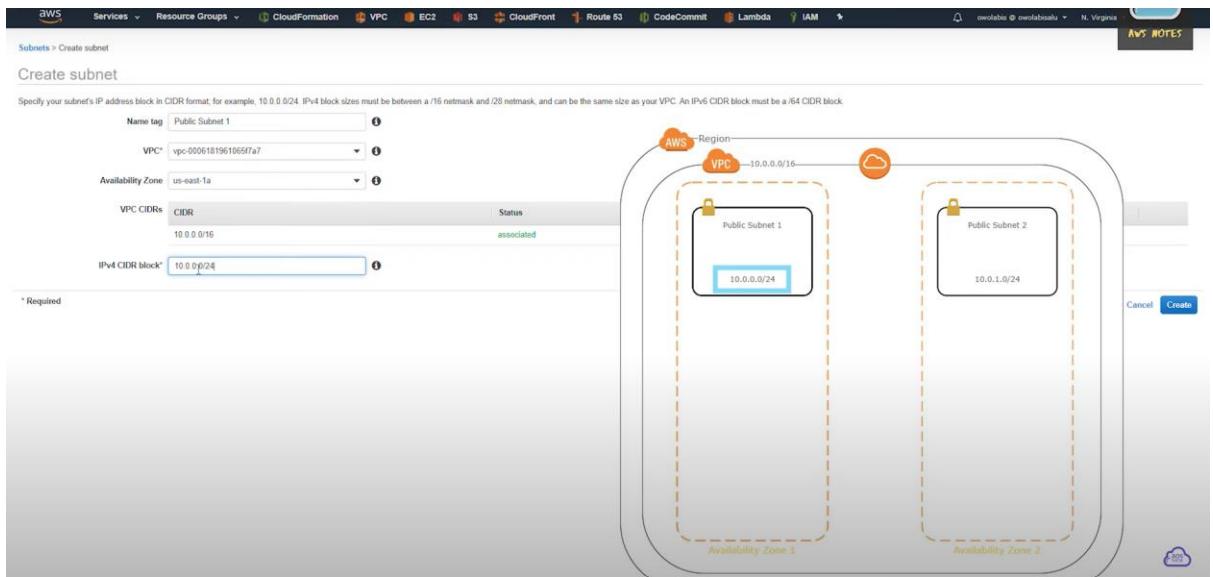
Cancel Attach

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

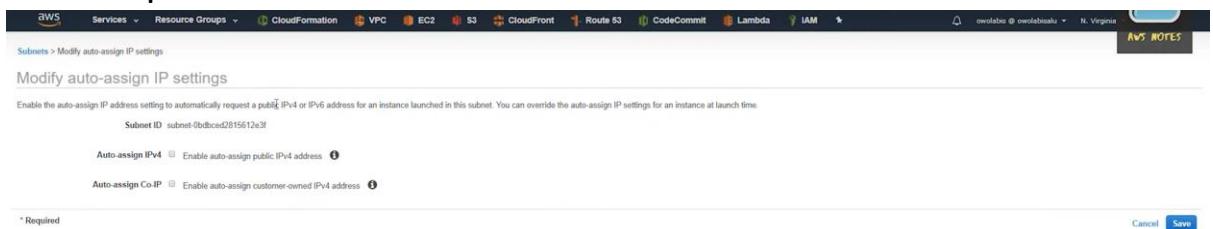
- Click on “Subnets” in Left Pane
- Fill the details like required VPC, Availability zone, Name of subnet(Public subnet1), CIDR block



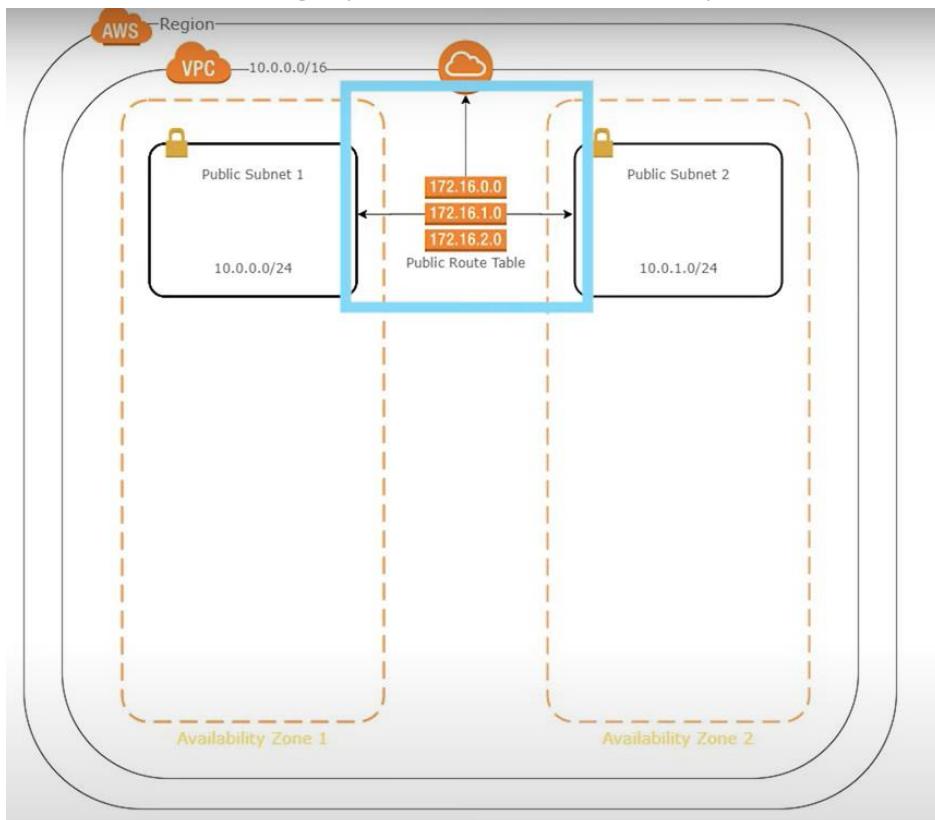
- Repeat the previous step to create another subnet (“Public subnet2”) and change the Availability Zone alone.



- 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 i

VPC* I C i

* 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
Public Route Table	rtb-03f178b1a31935128	-	-	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 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) and Private Subnet 2 (App Tier) (10.0.3.0/24), and Private Subnet 4 (Database Tier) (10.0.5.0/24). A Public Route Table routes traffic from the subnets to the internet.

Screenshot of the AWS 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) and Private Subnet 2 (App Tier) (10.0.3.0/24), and Private Subnet 4 (Database Tier) (10.0.5.0/24). 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

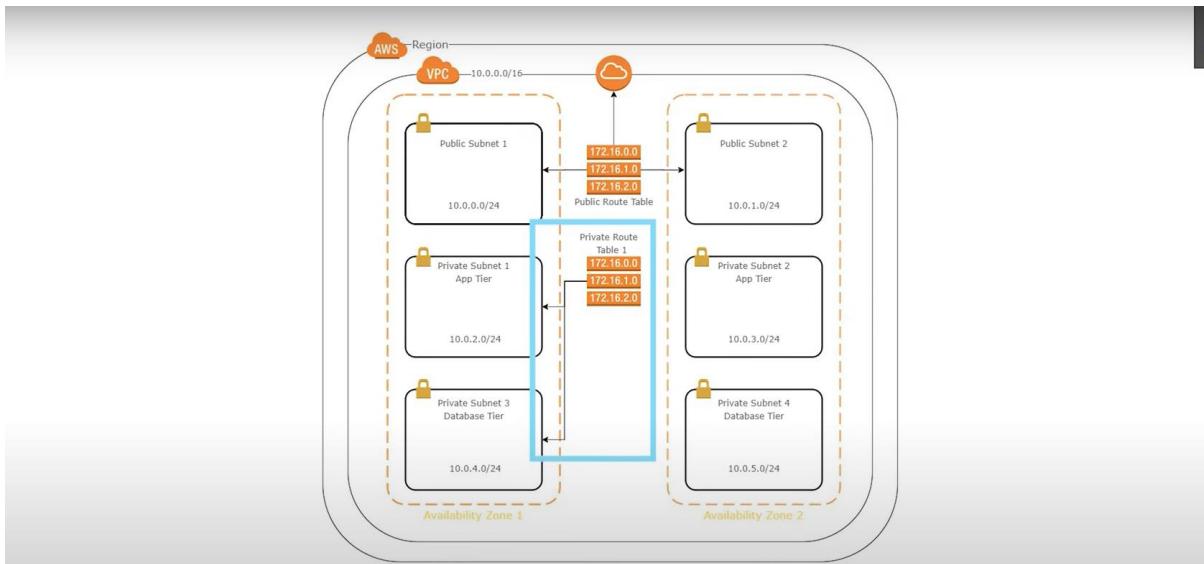
After Subnet creation:

Create subnet Actions ▾

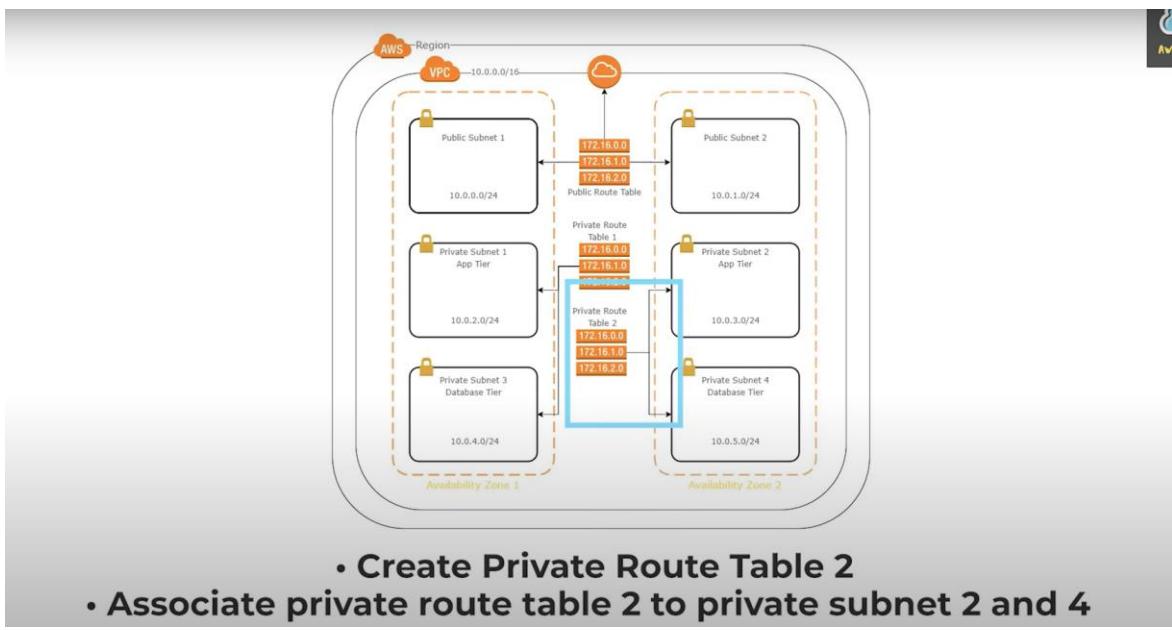
Filter by tags and attributes or search by keyword

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 Publ...
Public Subnet 2	subnet-0b5393e6f49dc4bb7	available	vpc-0006181961065f7a7 Demo ...	10.0.1.0/24	251	-	us-east-1b	use1-az6	rtb-03f178b1a31935128 Publ...
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



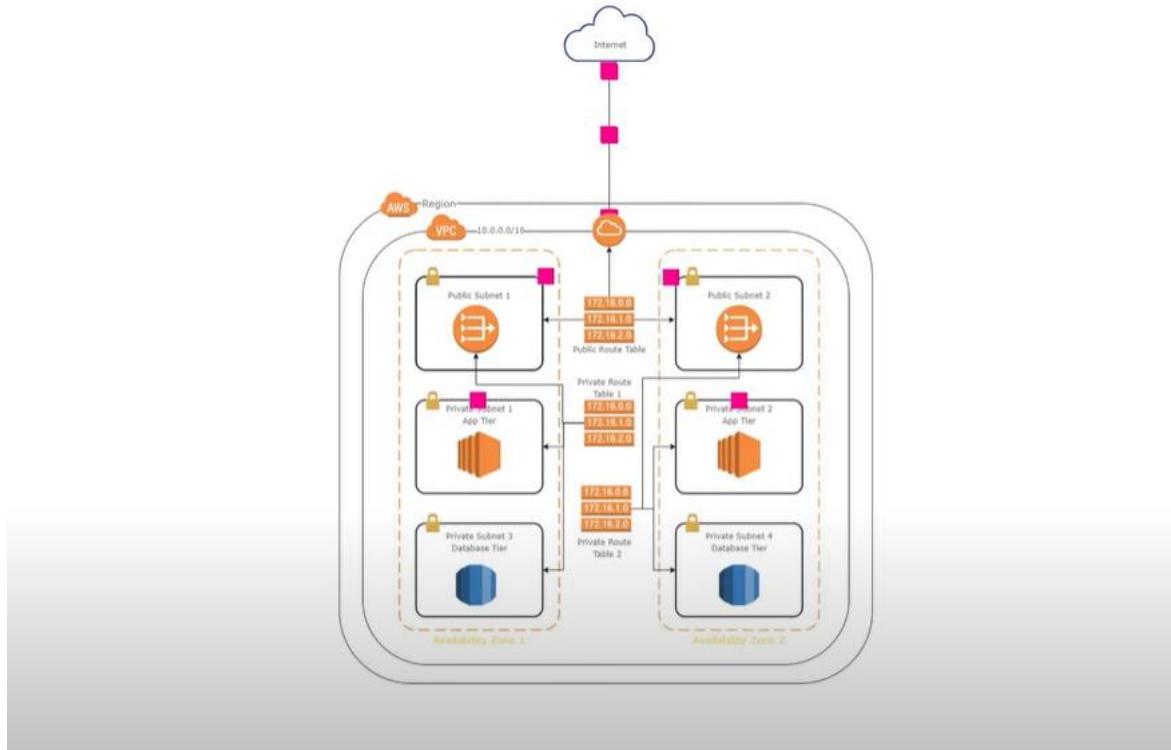
- Create Private Route Table 1
- Associate private route table 1 to private subnet 1 and 3



- 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-0bdbcd2015612e3f

Elastic IP Allocation ID: eipalloc-0e10042c1ba65c997

Key (128 characters maximum) Value (256 characters maximum)

Add Tag 50 remaining (Up to 50 tags maximum)

* Required

Create a NAT Gateway

NAT Gateways > Create NAT Gateway

Create 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

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

Add route

* Required

Cancel **Save routes**

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

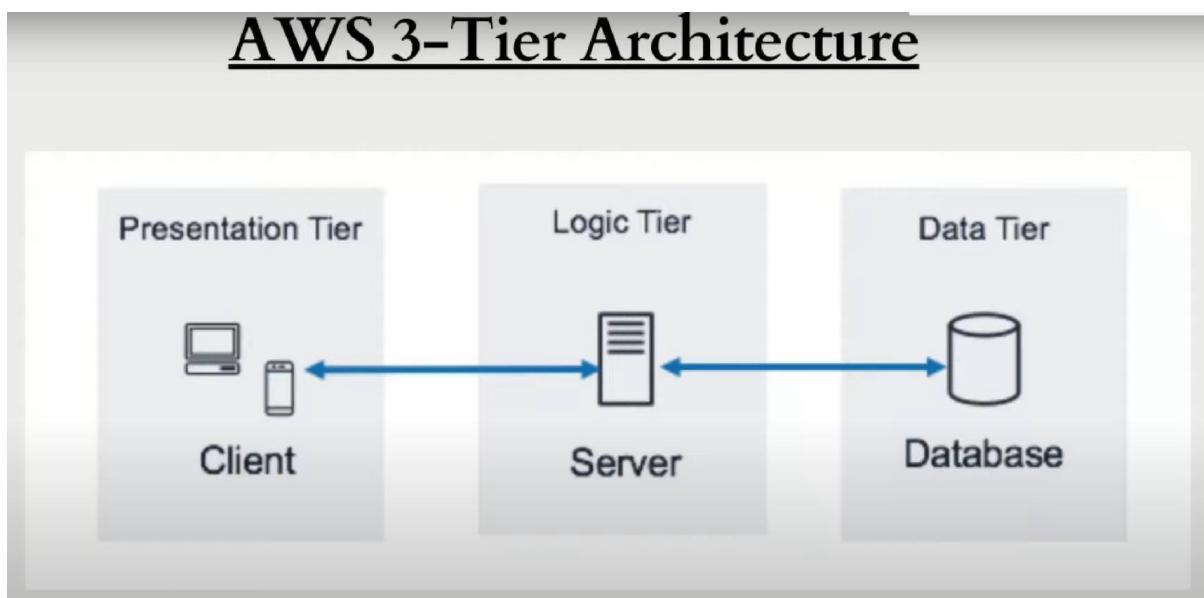
9. AWS 3-Tier Architecture with VPC, RDS, ELB:

AWS 3-Tier architecture consists of 3 Tiers namely Presentation Layer, Application Layer & Database Layer.

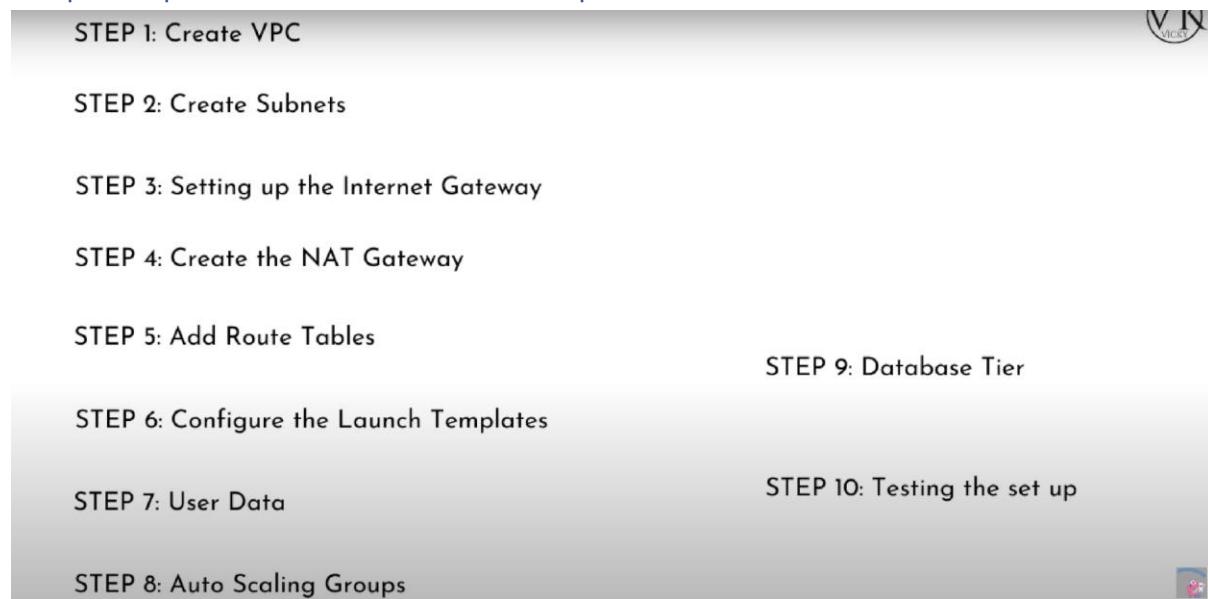
Presentation layer is the layer where the application is running. Eg: Amazon Web Portal

Application Layer is the layer where the business logic for computation is present. Eg: Java code, Springboot, Python, NodeJS

Database layer is the layer where the data is stored/maintained.



Steps required for the 3-tier Setup



Steps:

Step1: Create VPC

- Select region
- Services → Networking & Content delivery → VPC
- Create VPC
- Enter CIDR block
- Create
- The system will automatically create Route table which is private

VPC settings

Resources to create [Info](#)
Create only the VPC resource or the VPC and other networking resources.

VPC only VPC and more

Name tag - *optional*
Creates a tag with a key of 'Name' and a value that you specify.

3tierapplication

IPv4 CIDR block [Info](#)

IPv4 CIDR manual input
 IPAM-allocated IPv4 CIDR block

IPv4 CIDR
 10.0.0.0

IPv6 CIDR block [Info](#)

No IPv6 CIDR block
 IPAM-allocated IPv6 CIDR block
 Amazon-provided IPv6 CIDR block
 IPv6 CIDR owned by me

Tenancy [Info](#)

Default ▾

IPAM-allocated IPv4 CIDR block

IPv4 CIDR
 10.0.0.0/16

IPv6 CIDR block [Info](#)

No IPv6 CIDR block
 IPAM-allocated IPv6 CIDR block
 Amazon-provided IPv6 CIDR block
 IPv6 CIDR owned by me

Tenancy [Info](#)

Default ▾

Tags
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

Key	Value - <i>optional</i>
<input type="text"/> Name	<input type="text"/> 3tierapplication

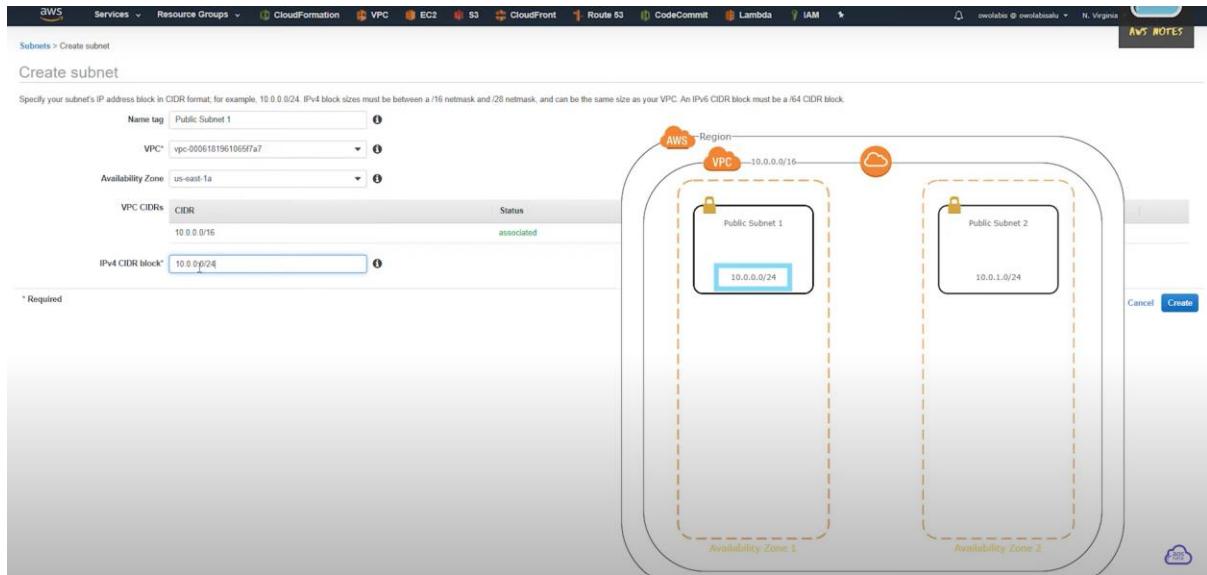
Add new tag

You can add 49 more tags.

Cancel

Step2: Create Subnets

- 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. Likewise, create two Private subnets for Application Tier in two different availability zones & two private subnets for Database tier in two different availability zones as like Presentation & Application Tier

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-0174ad7c38754403c
Public Subnet 2	subnet-0b5393e6fd9dc4bb7	available	vpc-0006181961065f7a7 Demo ...	10.0.1.0/24	251	-	us-east-1b	use1-az5	rtb-0174ad7c38754403c

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

Enable the auto-assign IP address setting to automatically request a public IPv4 or IPv6 address for an instance launched in this subnet. You can override the auto-assign IP settings for an instance at launch time.

Subnet ID: subnet-0bdbced2815612e3f

Auto-assign IPv4 Enable auto-assign public IPv4 address [?](#)

Auto-assign Co-IP Enable auto-assign customer-owned IPv4 address [?](#)

* Required

Cancel [Save](#)

Step3: Setting up the Internet Gateway

- 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

AWS Command Line Interface command

* Required

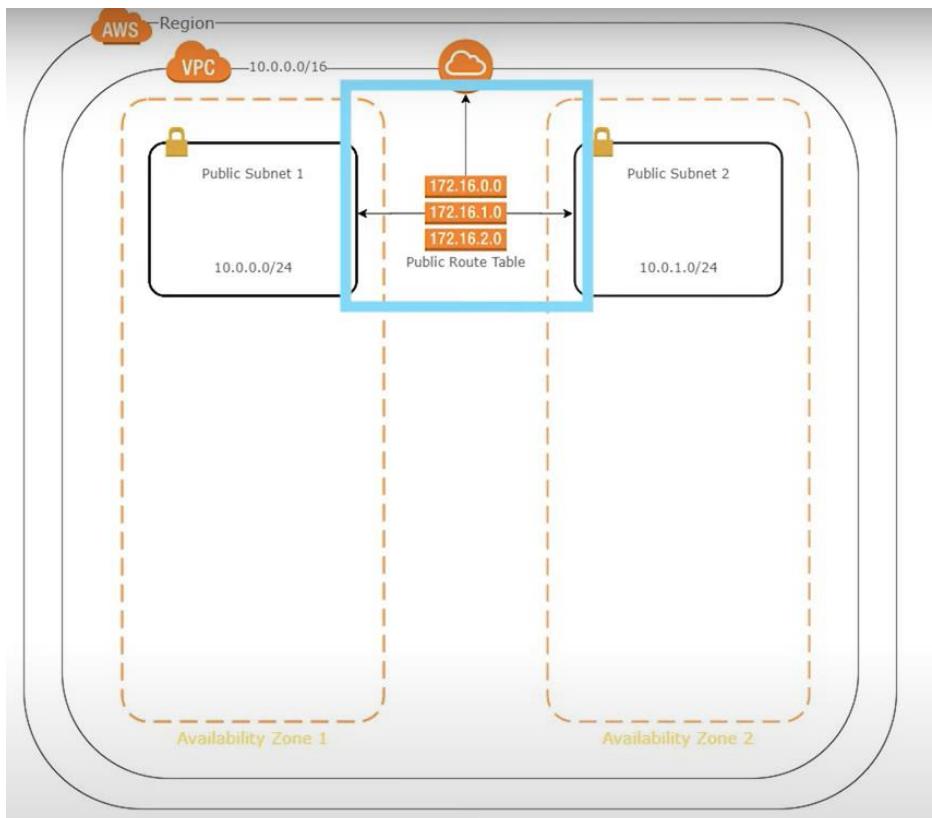
Cancel Attach

Create internet gateway Actions

Q Filter by tags and attributes or search by keyword

Name	ID	State	VPC	Owner
Demo IGW	igw-0a0032580f2d...	attached	vpc-00061819610...	960957692776
	igw-e468059f	attached	vpc-12122c68	960957692776

Step4: Create Route Table & Subnet Associations



- 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	
* 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
10.0.0.0/16	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		AWS NOTES			
<input type="text"/> Filter by tags and attributes or search by keyword							
Name	Route Table ID	Explicit subnet association	Edge associations	Main	VPC ID	Owner	
rtb-0174ad7c38754403c	-	-	-	Yes	vpc-0006181961065f7a7 ...	960957692776	
Public Route Table	rb-03f178b1a31935128	-	-	No	vpc-0006181961065f7a7 ...	960957692776	

[Edit subnet associations](#)

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**

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

Name tag: Private Subnet 1 | App Tier

VPC: vpc-0006181561065f7a7

Availability Zone: us-east-1a

VPC CIDRs: CIDR
10.0.0.0/16

IPv4 CIDR block: 10.0.0.0/24

Region: 10.0.0.0/16

Public Subnet 1: 10.0.0.0/24

Private Subnet 1: 10.0.2.0/24

Private Subnet 3: 10.0.4.0/24

Public Subnet 2: 10.0.1.0/24

Private Subnet 2: 10.0.3.0/24

Private Subnet 4: 10.0.5.0/24

Public Route Table: 172.16.0.0, 172.16.1.0, 172.16.2.0

Cancel Create

Subnets > Create subnet AWS NOTES

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 /28 netmask, and can be the same size as your VPC. An IPv6 CIDR block must be a /64 CIDR block.

Name tag	Private Subnet 2 App Tier	i
VPC*	vpc-000618196106597a7	i
Availability Zone	No preference	i
VPC CIDRs	CIDR 10.0.0.0/16	Status associated
IPv4 CIDR block*	<input type="text"/>	i

* Required

Subnets > Create subnet AWS NOTES

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 /28 netmask, and can be the same size as your VPC. An IPv6 CIDR block must be a /64 CIDR block.

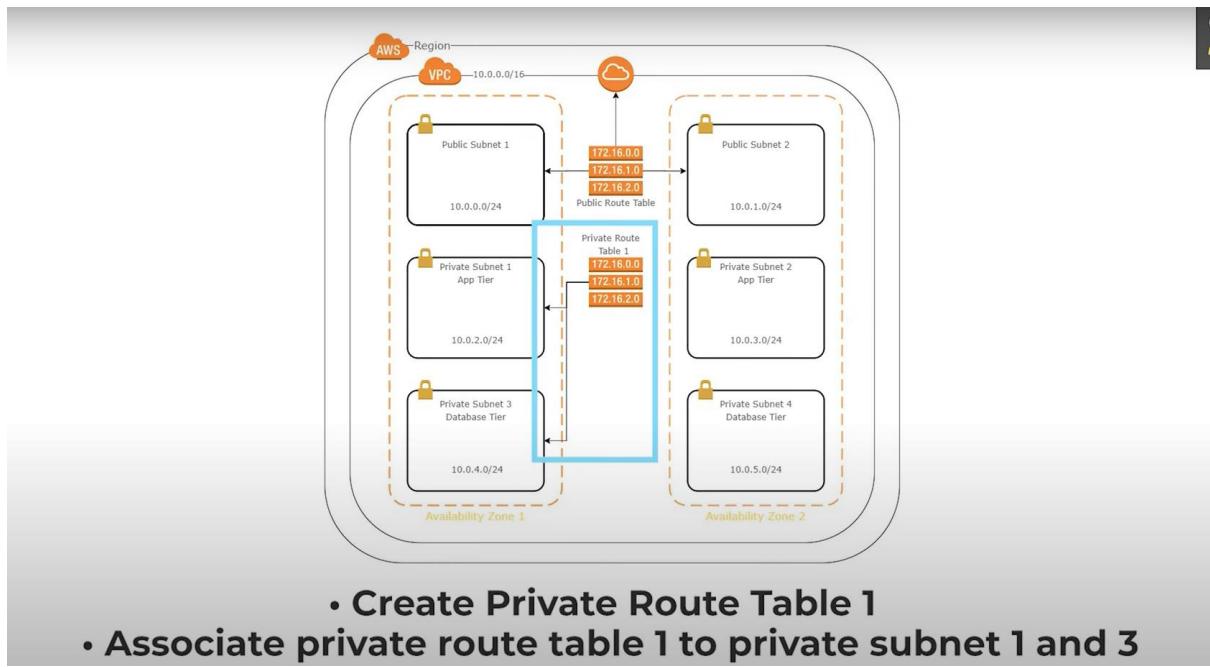
Name tag	Private Subnet 3 Database Tier	i
VPC*	vpc-000618196106597a7	i
Availability Zone	us-east-1a	i
VPC CIDRs	CIDR 10.0.0.0/16	Status associated
IPv4 CIDR block*	<input type="text"/> 10.0.4.0/24	i

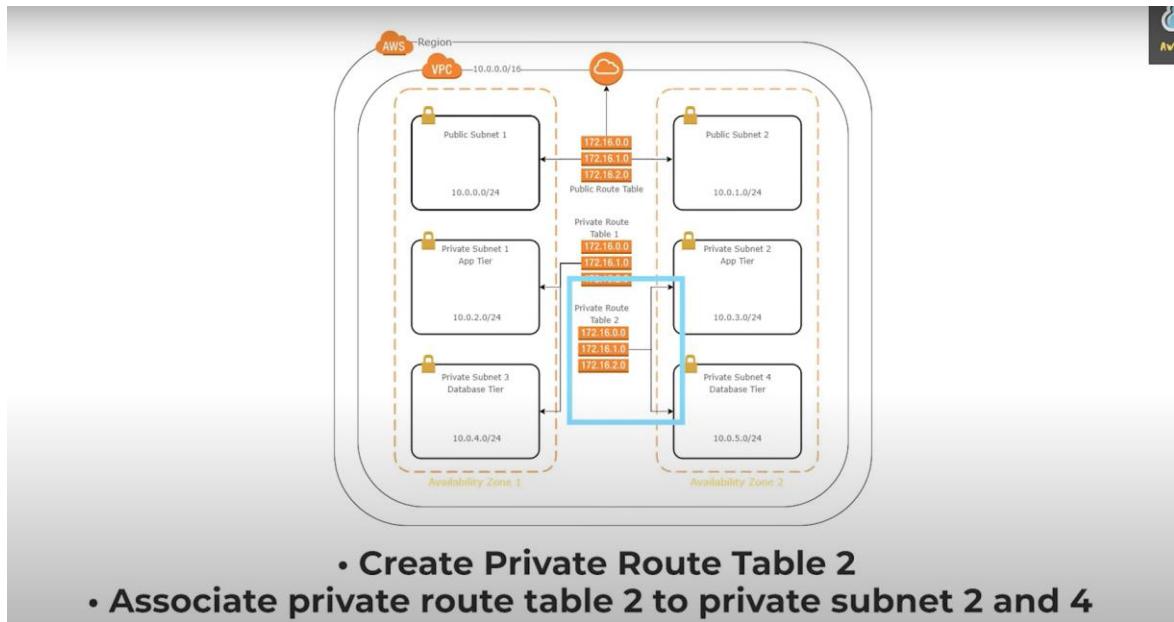
* Required

After Subnet creation:

Name	Subnet ID	State	VPC	IPv4 CIDR	Available IPv4	IPv6 CIDR	Availability Zone	Availability Zone ID	Route table
Public Subnet 1	subnet-0bdbced281561203f	available	vpc-0006181961065f7a7 Demo ...	10.0.0.0/24	251	-	us-east-1a	use1-az4	rtb-03f178b1a31935128 Publi...
Public Subnet 2	subnet-0653936f649e4b7	available	vpc-0006181961065f7a7 Demo ...	10.0.1.0/24	251	-	us-east-1b	use1-az5	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-0174ad7c38754403c
Private Subnet 2 App Tier	subnet-04660ad026fd62fa5	available	vpc-0006181961065f7a7 Demo ...	10.0.3.0/24	251	-	us-east-1b	use1-az5	rtb-0174ad7c38754403c
Private Subnet 3 Database Tier	subnet-00444dac448709631	available	vpc-0006181961065f7a7 Demo ...	10.0.4.0/24	251	-	us-east-1a	use1-az4	rtb-0174ad7c38754403c
Private Subnet 4 Database Tier	subnet-073ac77ba7d995910	available	vpc-0006181961065f7a7 Demo ...	10.0.5.0/24	251	-	us-east-1b	use1-az5	rtb-0174ad7c38754403c

Step4.1: Create Two Private Route tables and attach it to 4 subnets

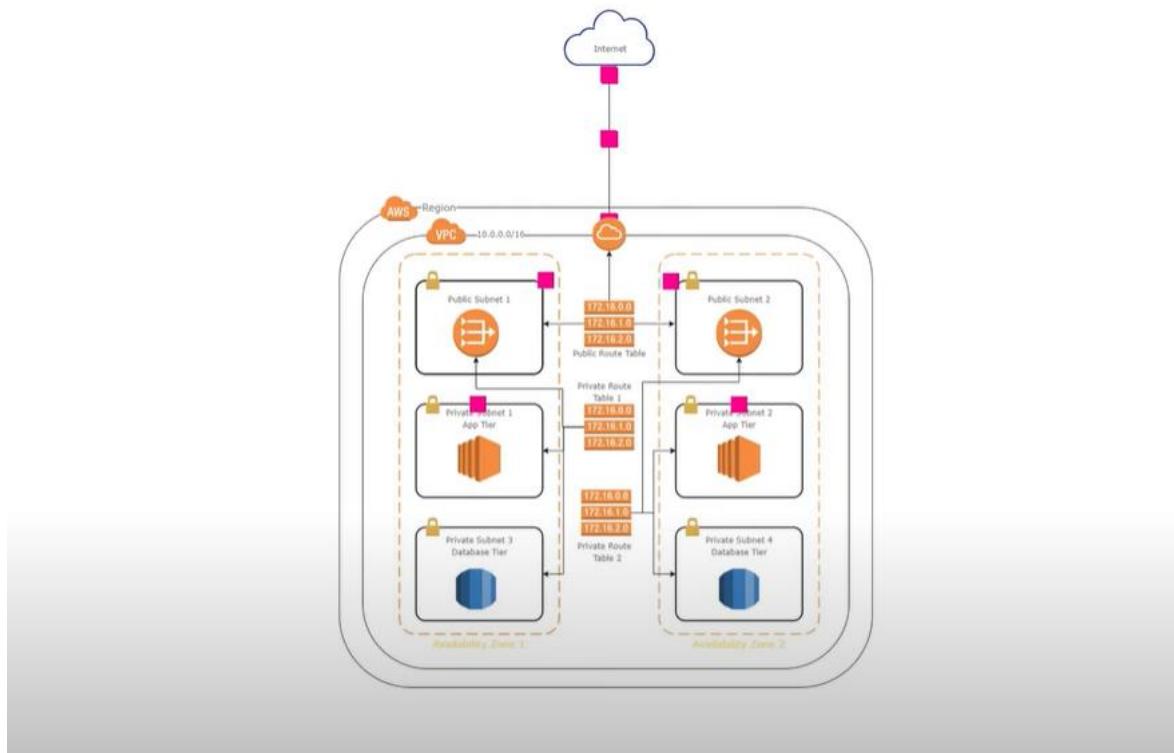




- Create Private Route Table 2
- Associate private route table 2 to private subnet 2 and 4

Follow the step 4 till the Subnet Associations

Step5: Create the 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 address allocated successfully.
Elastic IP address 54.165.99.81

Associate this Elastic IP address

Name	Allocated IPv4 add...	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	-	-	-	-

Step5.1: 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**

Create NAT Gateway | VPC Manager | Introducing Amazon VPC NAT

Create NAT Gateway

Create a NAT gateway and assign it an Elastic IP address. [Learn more](#).

Subnet:

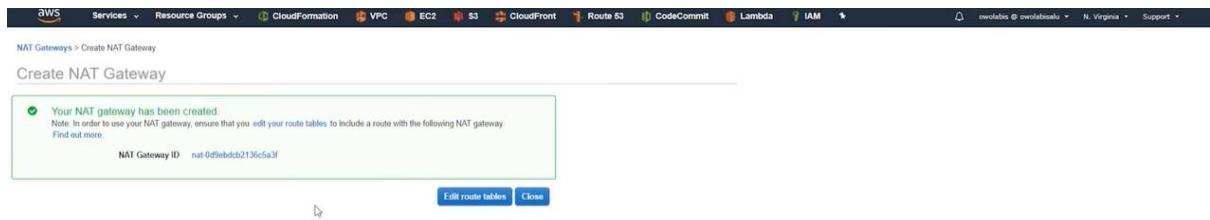
Elastic IP Allocation ID: **Allocate Elastic IP address**

Key: Value:

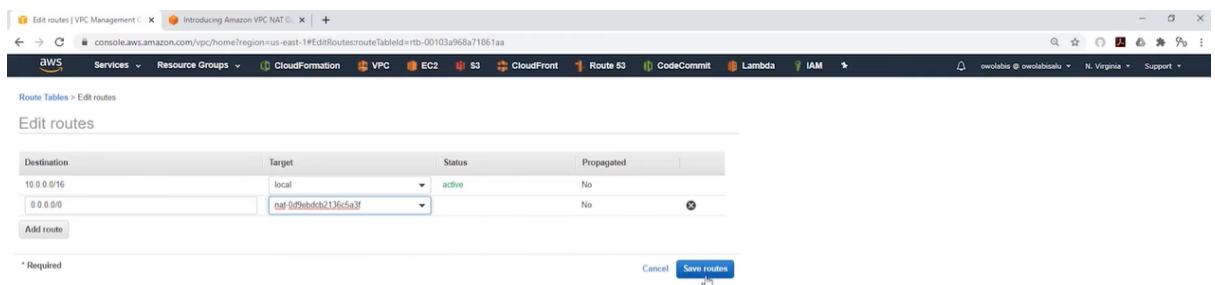
This resource currently has no tags

Add Tag (Up to 50 tags maximum)

* Required Cancel **Create a 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



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

Step6: Launch Template

- Go to AWS → EC2 → Launch Template → Create Launch Template
- Give Template Name, Version(optional), Check-in Provide Guidance, **Amazon linux** as AMI, **t2.micro** as Instance Type, Keypair.

- Create new Security group
 - Give name for SG as Web-Tier SG
 - Give our VPC value
 - Enter new Inbound Security group rule for SSH
 - Type-SSH; Protocol- TCP; Port range-22
 - Source- Custom; Source- 0.0.0.0/0
 - Enter new Inbound Security group rule for SSH
 - Type-HTTP; Protocol- TCP; Port range-80
 - Source- Custom; Source- 0.0.0.0/0
- Enter the script in the user-data

```
#!/bin/bash
yum update -y
yum install -y httpd
systemctl start httpd
systemctl enable https
echo "<html><body><h1>Vicky Website - Demo for 3Tier Application</h1>
</body></html>" > /var/www/html/index.html
```

- Go to Auto Scaling → Auto- Scaling Group
 - Enter the name of the Auto-Scaling Group as LT3tierApplication
 - Template as the one which we created.

Launch template Info		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.		
<input type="text" value="LT3tierapplication"/> ▼ C		Create a launch template
Version <input type="button" value="Default (1)"/> ▼ C Create a launch template version		
Description	Launch template	Instance type
-	LT3tierapplication lt-0165aff60f3f629d0	t2.micro
AMI ID	Security groups	Request Spot Instances
ami-0cff7528ff583bf9a	-	No
Key pair name	Security group IDs	
ssmdemo	sg-0e0a0677a007ef734	

- Select our VPC

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

VPC
Choose the VPC that defines the virtual network for your Auto Scaling group.

vpc-9b601ae6
172.31.0.0/16 Default

vpc-0f289a79d1df029b1 (3tierapplication)
10.0.0.0/16

vpc-9b601ae6
172.31.0.0/16 Default

Select Availability Zones and subnets

Create a subnet

For most applications, you can use multiple Availability Zones and let EC2 Auto Scaling balance your instances across the zones. The default VPC and default subnets are suitable for getting started quickly.

can use in the chosen VPC.

Instance type requirements

Override launch template

Launch template **LT3tierapplication** Version Default Description -

lt-0165aff60f3f629d0

Instance type t2.micro

- Availability zones as our two public subnets

Add notifications

Step 6 (optional)
Add tags

Step 7
Review

Availability Zones and subnets
Define which Availability Zones and subnets your Auto Scaling group can use in the chosen VPC.

Select Availability Zones and subnets

us-east-1c | subnet-Offa480a6f7c43b92 (webtier-public)
10.0.32.0/24

us-east-1c | subnet-0cfaddfdb13084722 (webtier2-public)
10.0.0.0/24

Create a subnet

Instance type requirements

Override launch template

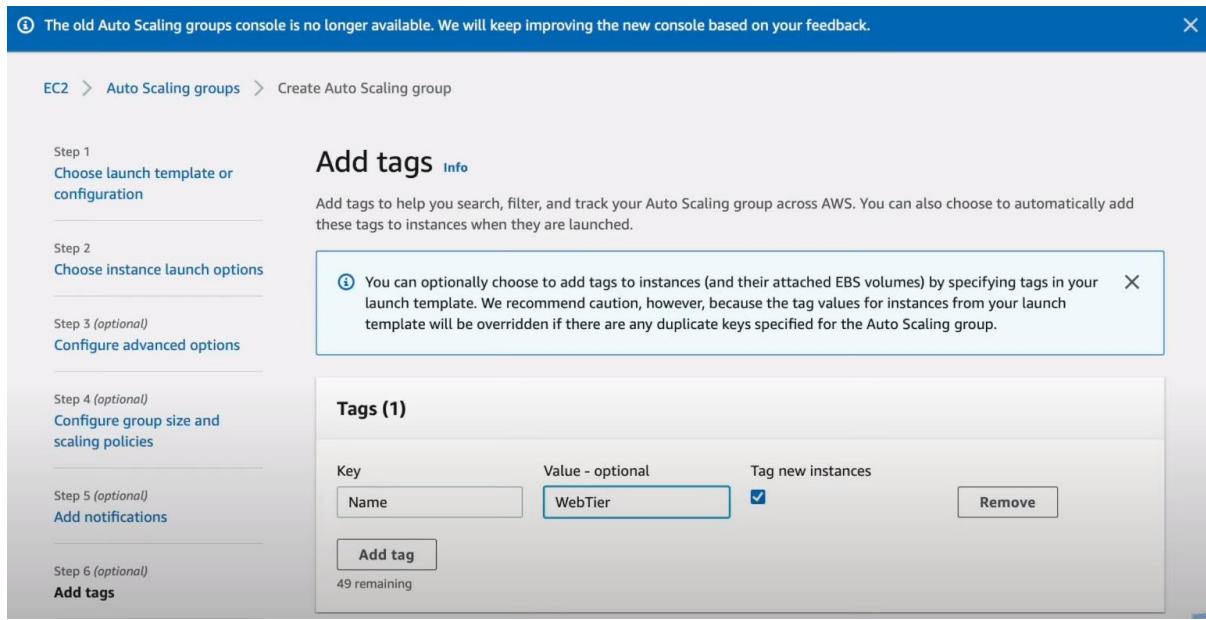
Launch template **LT3tierapplication** Version Default Description -

lt-0165aff60f3f629d0

Instance type t2.micro

Cancel Previous Skip to review **Next**

- No ELB as of now, give the size as Minimum capacity, Desired Capacity, Maximum capacity as 1.
- Give the Tag as **Web-Tier Machine**



- Then click create
- This will create the Auto-scaling group for the Web-Tier
- Similarly, create Auto-scaling group for Application tier with new Launch template for Application-Tier with VPC as our VPC, SG with Following Inbound rules
 - Enter new Inbound Security group rule for MySQL/Aurora
 - Type-MySQL/Aurora; Protocol- TCP; Port range-3306
 - Source- Custom; Source- 0.0.0.0/0 or **SG of Database tier**
 - Enter new Inbound Security group rule for SSH
 - Type-SSH; Protocol- TCP; Port range-22
 - Source- Custom; Source- 0.0.0.0/0 or **SG of Web-tier**
 - Enter new Inbound Security group rule for HTTP
 - Type-HTTP; Protocol- TCP; Port range-80
 - Source- Custom; Source- 0.0.0.0/0

Step7: Create Database

- Create RDS by choosing RDS → Create Database
- Enter details like **Standard** as Creation method, **Free tier** as Template, **MySQL** as Engine & version, Give credentials details in **Credentials settings**, **Burstable classes** & change tier as **t2.micro** in Instance configurations.

- Select our VPC in the connectivity.
 - Create Subnet group → New subnet group
 - Set Public access as ‘No’
 - VPC Security group → Create New
 - We can give ***availability zone preference***
 - Db port as **3306**
- Database Authentication
 - Enable Password authentication
- Then click Create Database

RDS > Create database

Create database

Choose a database creation method Info

Standard create
You set all of the configuration options, including ones for availability, security, backups, and maintenance.

Easy create
Use recommended best-practice configurations. Some configuration options can be changed after the database is created.

Engine options

Engine type Info

- Amazon Aurora
- MySQL
- MariaDB

Engine options

Engine type Info

- Amazon Aurora
- MySQL
- MariaDB

- PostgreSQL
- Oracle
- Microsoft SQL Server

Edition

MySQL Community

Known issues/limitations
Review the [Known issues/limitations](#) to learn about potential compatibility issues with specific database versions.

Version

MySQL 8.0.28

Templates
Choose a sample template to meet your use case.

- Production**
Use defaults for high availability and fast, consistent performance.
- Dev/Test
This instance is intended for development use outside of a production environment.
- Free tier**
Use RDS Free Tier to develop new applications, test existing applications, or gain hands-on experience with Amazon RDS.

Settings

DB instance identifier [Info](#)
Type a name for your DB instance. The name must be unique across all DB instances owned by your AWS account in the current AWS Region.

The DB instance identifier is case-insensitive, but is stored as all lowercase (as in "mydbinstance"). Constraints: 1 to 60 alphanumeric characters or hyphens. First character must be a letter. Can't contain two consecutive hyphens. Can't end with a hyphen.

Credentials Settings

Master username [Info](#)
Type a login ID for the master user of your DB instance.

1 to 16 alphanumeric characters. First character must be a letter.
 Auto generate a password
Amazon RDS can generate a password for you, or you can specify your own password.

Master password [Info](#)

Constraints: At least 8 printable ASCII characters. Can't contain any of the following: / (slash), '(single quote), "(double quote) and @ (at sign).

Confirm password [Info](#)

Instance configuration

The DB instance configuration options below are limited to those supported by the engine that you selected above.

DB instance class [Info](#)
 Standard classes (includes m classes)
 Memory optimized classes (includes r and x classes)
 Burstable classes (includes t classes)

db.t3.micro
2 vCPUs 1 GiB RAM Network: 2,085 Mbps

 Include previous generation classes

Storage

Storage type [Info](#)

Baseline performance determined by volume size

Allocated storage
 GiB
(Minimum: 20 GiB. Maximum: 16,384 GiB) Higher allocated storage can improve IOPS performance.

Storage

Storage type [Info](#)

Baseline performance determined by volume size

Allocated storage
 GiB
(Minimum: 20 GiB. Maximum: 16,384 GiB) Higher allocated storage can improve IOPS performance.

Storage autoscaling [Info](#)
Provides dynamic scaling support for your database's storage based on your application's needs.

Enable storage autoscaling
Enabling this feature will allow the storage to increase after the specified threshold is exceeded.

Maximum storage threshold [Info](#)
Charges will apply when your database autoscales to the specified threshold
 GiB
Minimum: 22 GiB. Maximum: 16,384 GiB

Connectivity

Virtual private cloud (VPC) [Info](#)
VPC that defines the virtual networking environment for this DB instance.

3tierapplication (vpc-0f289a79d1df029b1) ▾
Only VPCs with a corresponding DB subnet group are listed.

ⓘ After a database is created, you can't change its VPC.

Subnet group [Info](#)
DB subnet group that defines which subnets and IP ranges the DB instance can use in the VPC you selected.

Create new DB Subnet Group ▾

Public access [Info](#)
 Yes
Amazon EC2 instances and devices outside the VPC can connect to your database. Choose one or more VPC security groups that specify which EC2 instances and devices inside the VPC can connect to the database.
 No
RDS will not assign a public IP address to the database. Only Amazon EC2 instances and devices inside the VPC can connect to your database.

VPC security group
Choose a VPC security group to allow access to your database. Ensure that the security group rules allow the appropriate incoming traffic.

Choose existing
Choose existing VPC security groups

Create new
Create new VPC security group

New VPC security group name
database

Availability Zone [Info](#)
No preference

Additional configuration

Database port [Info](#)
TCP/IP port that the database will use for application connections.
3306

Database authentication

Database authentication options [Info](#)
 Password authentication
Authenticates using database passwords.
 Password and IAM database authentication
Authenticates using the database password and user credentials through AWS IAM users and roles.
 Password and Kerberos authentication
Choose a directory in which you want to allow authorized users to authenticate with this DB instance using Kerberos Authentication.

Estimated monthly costs

The Amazon RDS Free Tier is available to you for 12 months. Each calendar month, the free tier will allow you to use the Amazon RDS resources listed below for free:

- 750 hrs of Amazon RDS in a Single-AZ db.t2.micro, db.t3.micro or db.t4g.micro Instance.
- 20 GB of General Purpose Storage (SSD).
- 20 GB for automated backup storage and any user-initiated DB Snapshots.

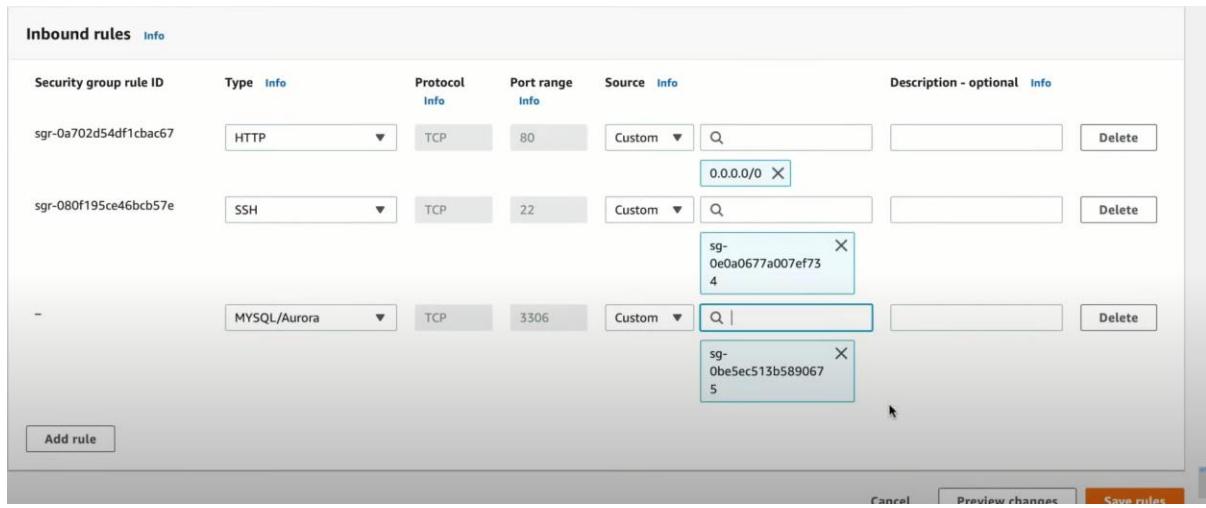
Learn more about AWS Free Tier. [\[?\]](#)

When your free usage expires or if your application use exceeds the free usage tiers, you simply pay standard, pay-as-you-go service rates as described in the [Amazon RDS Pricing page](#). [\[?\]](#)

ⓘ You are responsible for ensuring that you have all of the necessary rights for any third-party products or services that you use with AWS services.

Cancel [Create database](#)

- This will create the database.
- Once the database is created, take the **SG of this database and paste it in the inbound traffic of the Application Tier**

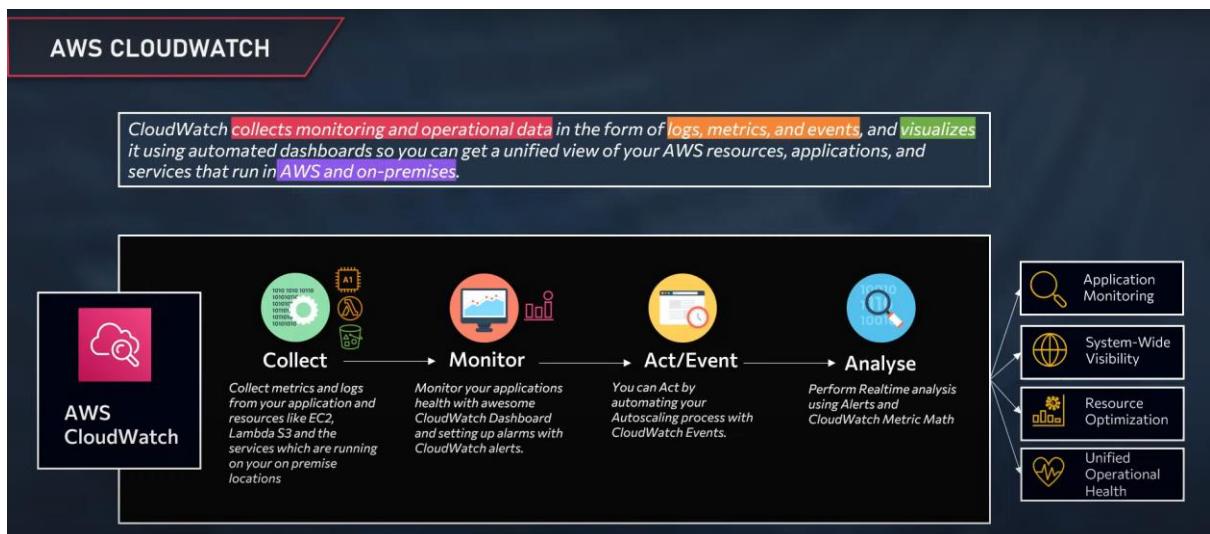


Step8: Enable connection between the 3 Tiers

- Login to Web-tier instance, copy the ssh key file and paste it into Application-tier instance or use cmd: ssh-copy-id ec2-user@<pvt Ip of application tier>
- To check connection, cmd: ping ec2-user@<pvt Ip of application tier>
- To connect Database & Application tier
 - Copy the endpoint of Database from Console
 - Install MySql in Application-tier console:
 - Cmd: sudo yum install mysql -y
 - Command to establish connection for database:
 - Cmd: mysql -h <end-point> -P 3306 -u <username of db> -p <password of db>

10. AWS CloudWatch Setup

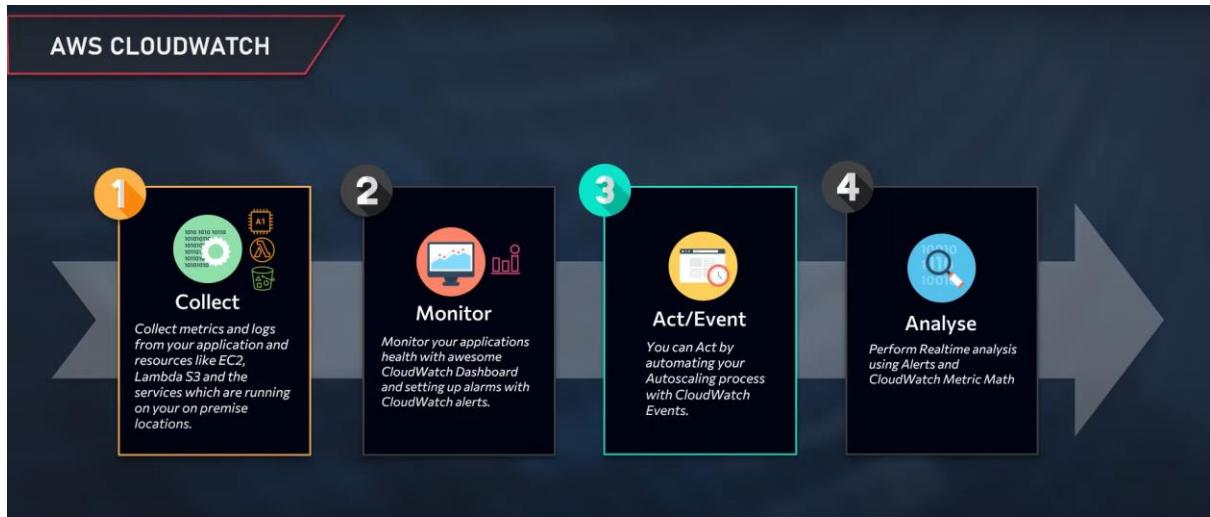
AWS CloudWatch is the feature provided by the AWS to Collect, Monitor, Act, Analyze the resources which are part of our AWS project. We can visualize the logs collected by CloudWatch by using the dashboards provided by the AWS.



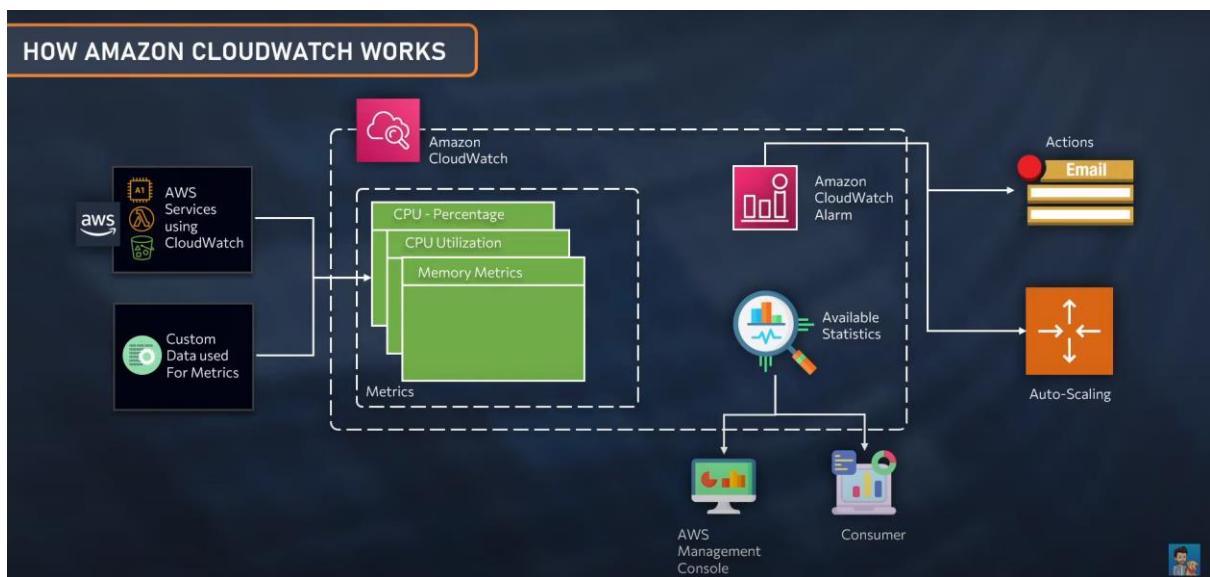
Four Pillars:

- Collect
- Monitor

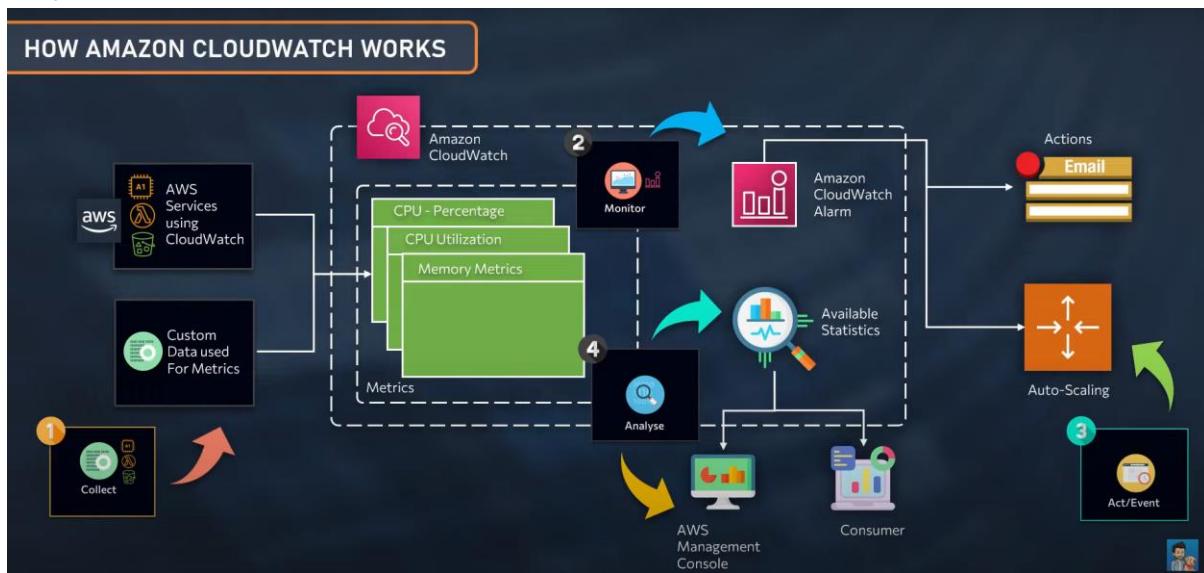
- Act
- Analyze



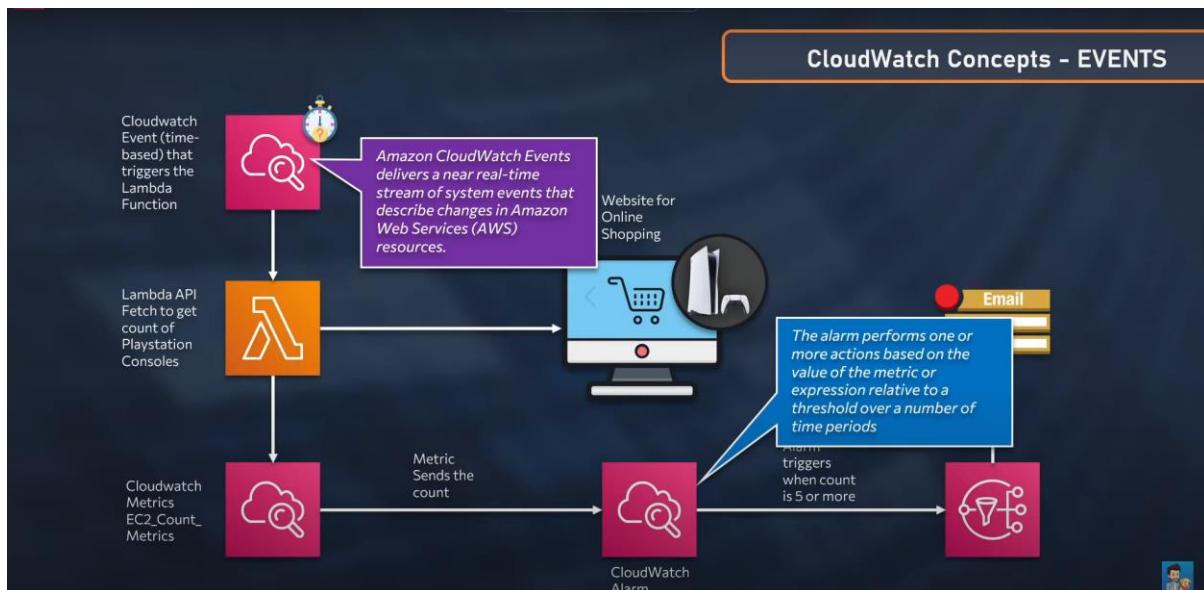
How CloudWatch Works?



Implementation of Four Pillars:



CloudWatch events:

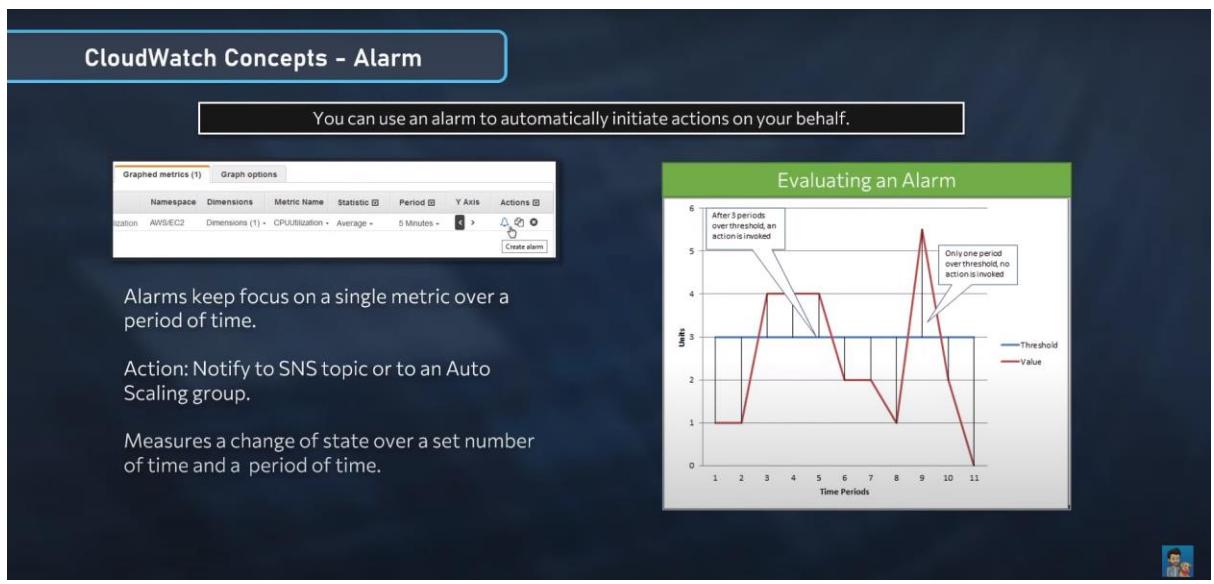


- Suppose we wish to be notified whenever there is a stock for PS5 consoles, we can write a Lambda API fetch to get the count of PS5 consoles(not to try this as this is unethical, mentioned here just for an example). We can setup an event to trigger the Lambda API fetch to trigger once in every minute.
- Data fetched through Lambda API fetch is sent across Cloudwatch Metrics.
- We can setup an cloudWatch Alarm which is based on the count delivered from the metrics, when the count is more than

5 or so, then CloudWatch will notify us that PS5 is available via Mail through SNS.

CloudWatch Alarms:

- You can use an alarm to automatically initiate actions on your behalf.
- Alarms keeps on focussing over a single metric over a period of time.
- Action: Notify to SNS topic or to an Auto Scaling group
- Measures a change of state over a set number of time & a period of time.

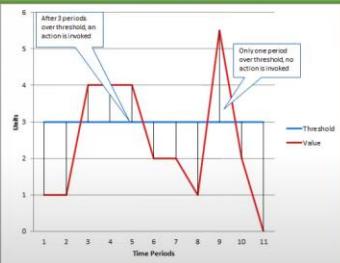


- When you create an alarm, there are 3 settings to be set.
 - Period: The length of time to evaluate the metric or expression to create each individual data point for an alarm.
 - Evaluation periods: The number of the most recent periods, or data points, to evaluate when determining the alarm state.
 - Datapoints to Alarm: The number of data points within the evaluation periods that must be breaching to cause the alarm to go to the ALARM state.

CloudWatch Concepts - Alarm

You can use an alarm to automatically initiate actions on your behalf.

Evaluating an Alarm



When you create the CloudWatch alarm, you set the 3 settings:

Period: The length of time to evaluate the metric or expression to create each individual data point for an alarm.

Evaluation Periods: The number of the most recent periods, or data points, to evaluate when determining alarm state.

Datapoints to Alarm: The number of data points within the Evaluation Periods that must be breaching to cause the alarm to go to the ALARM state.

IN ALARM

The metric or expression is **outside** of the defined threshold.

OK

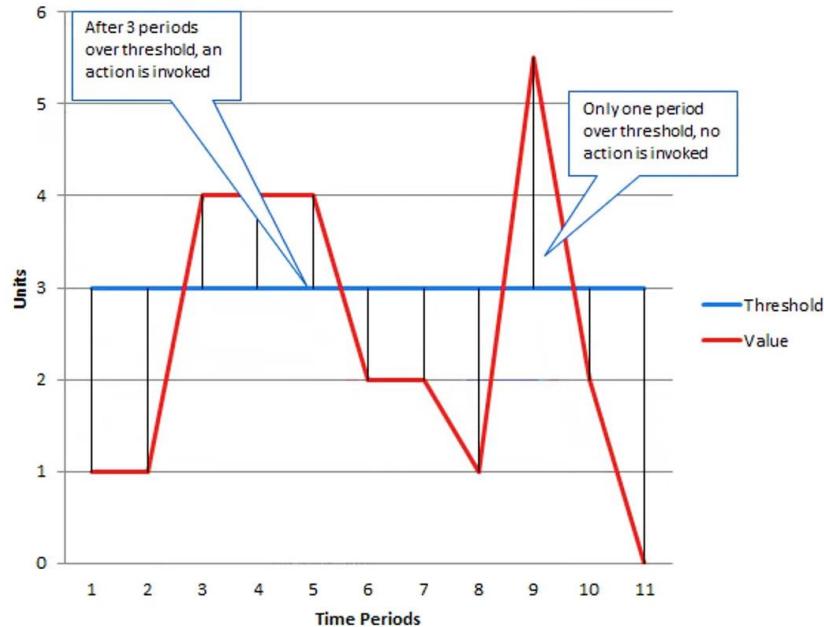
The metric or expression is **within** the defined threshold.

INSUFFICIENT DATA

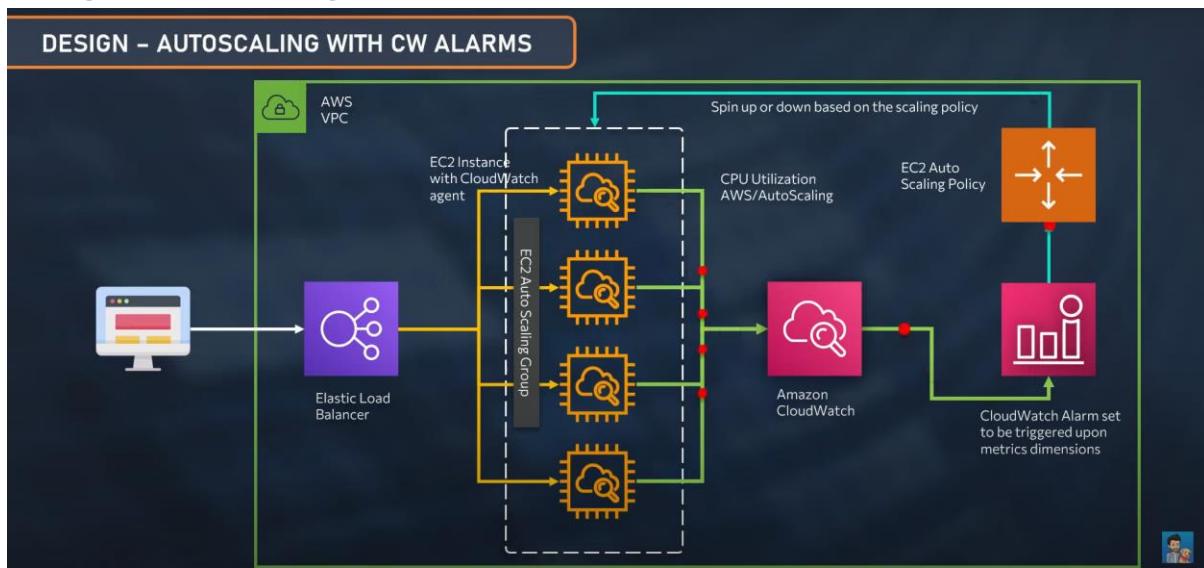
The alarm has just **started** or not enough data is available.



Evaluating an Alarm



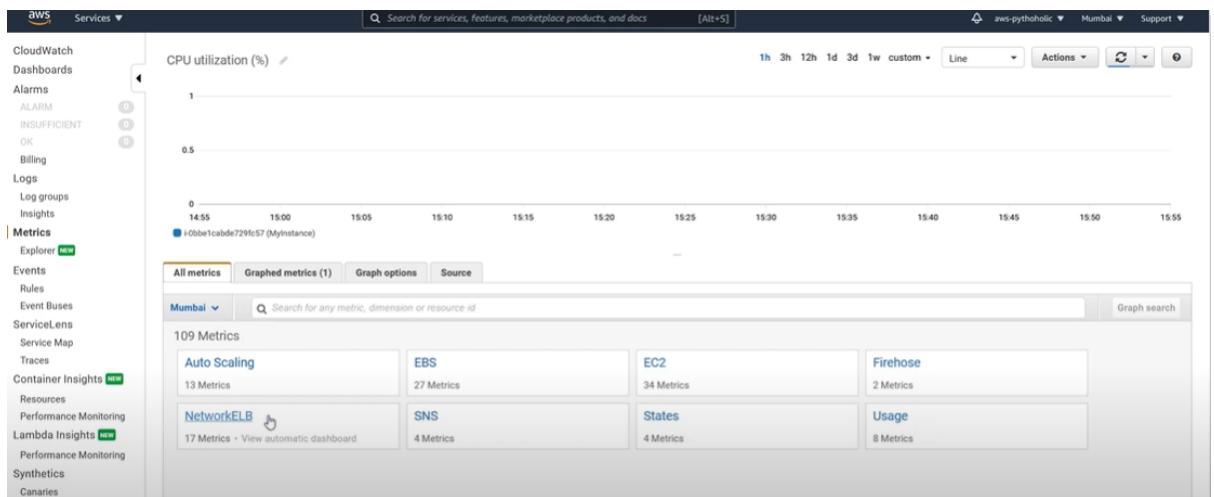
Design- Autoscaling with CloudWatch Alarms:



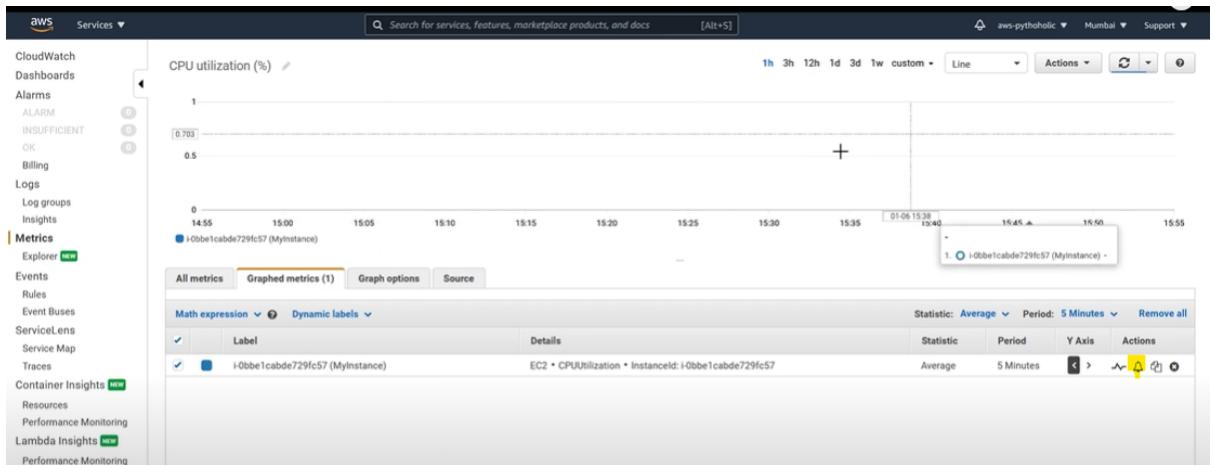
Create Alarm with SNS notification for CPU utilization:

Steps:

- Go to AWS → Cloudwatch → Metrics



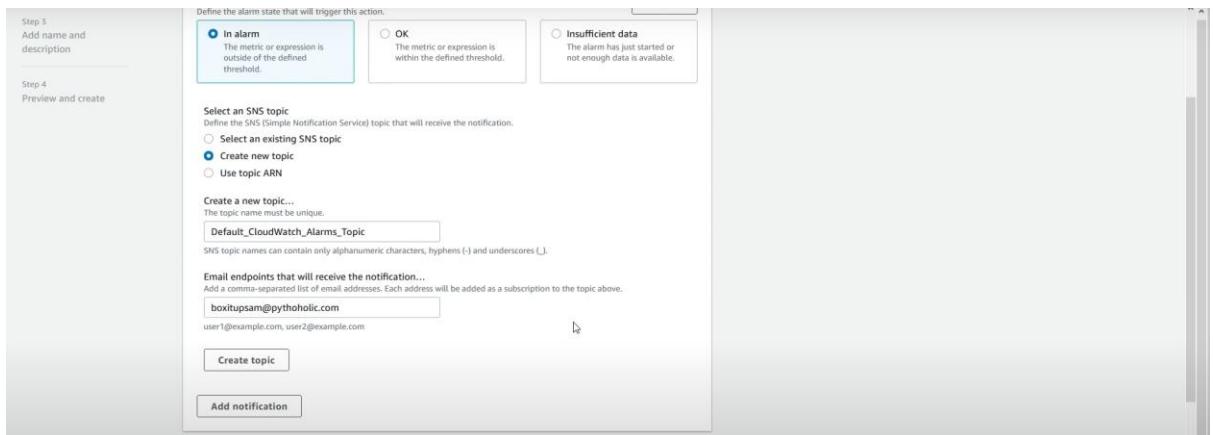
- Select the required instance from All Metrics and click 'Alarm' icon in the instance line.



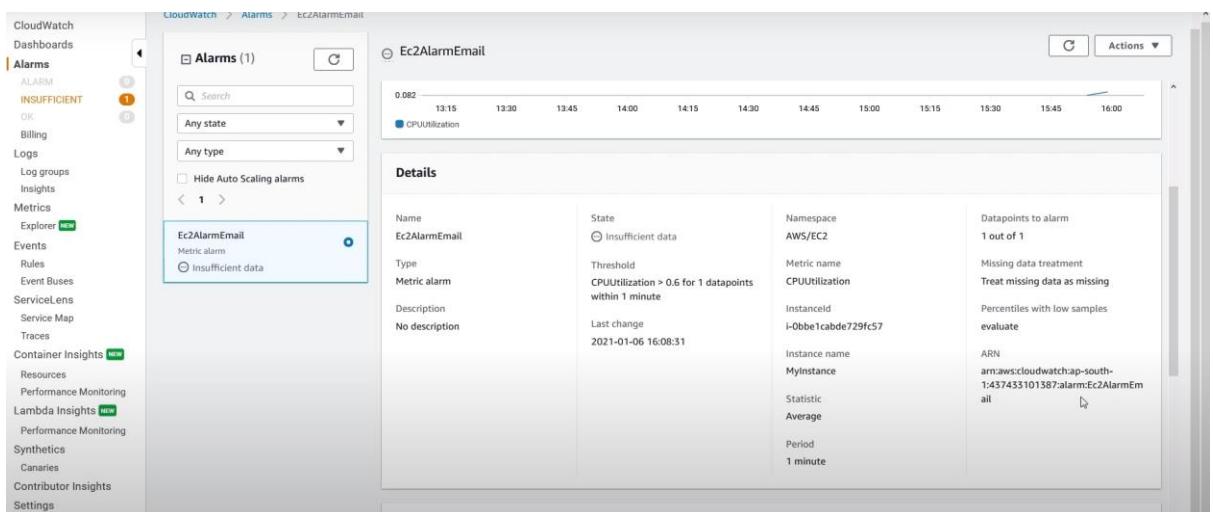
- Enter Metric name, Instance Id, Statistics, Period as **1 minute**, Threshold type as **Static**, Whenever CPUutilization is greater than **0.6** & Click **Next**

The first screenshot shows the 'Specify metric and conditions' step of the 'Create alarm' wizard. It shows a graph of CPUUtilization for instance i-0bbe1cabde729fc57. The threshold is set to 0.6. The second screenshot shows the 'Conditions' step, where the 'Threshold type' is set to 'Static' and the condition is 'Greater' than 0.6.

- In Configure actions, Alarm state as **In Alarm**, Select an SNS topic as **Create new topic**, Email endpoint as **Mail Id** & click Create Topic.



- Enter name for Alarm and click **Create Alarm**
- The Alarm will be created with state as “Insufficient Data”.



- You can select & create this view as Dashboard.



- To test this, install stress application in the Ec2 instance.
Cmd: Sudo amazon-linux-extras install epel -y
Sudo yum install stress -y
Stress –help
Stress –cpu 4 –timeout 180

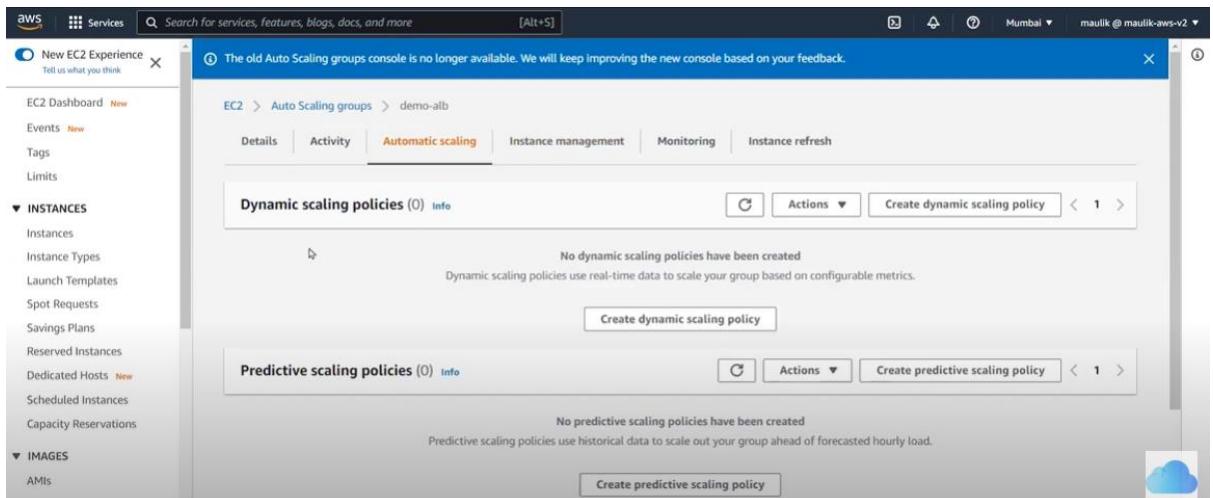
Create Alarm with AutoScaling Group for CPU Utilization

To create a CloudWatch Alarm to trigger auto-scaling group to create new instance if the CPU utilization is above mentioned threshold limit.

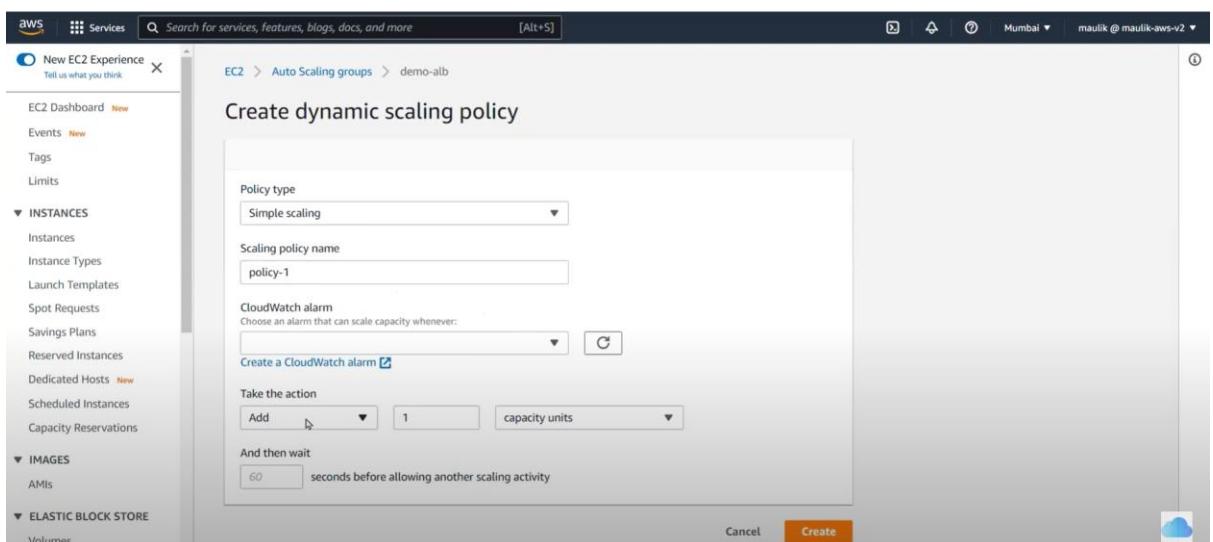
Steps:

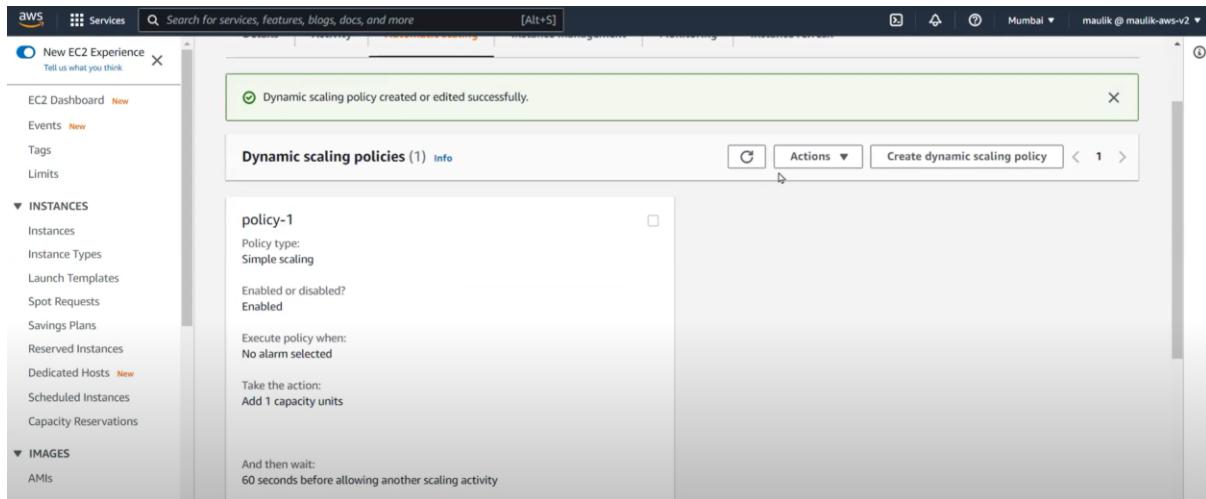
Step1: Update Dynamic scaling policy in Auto-scaling group

- Go to EC2 → Auto-scaling group → Select created Auto-scaling group
- In Automatic scaling section, click **Create Dynamic Scaling Policy**

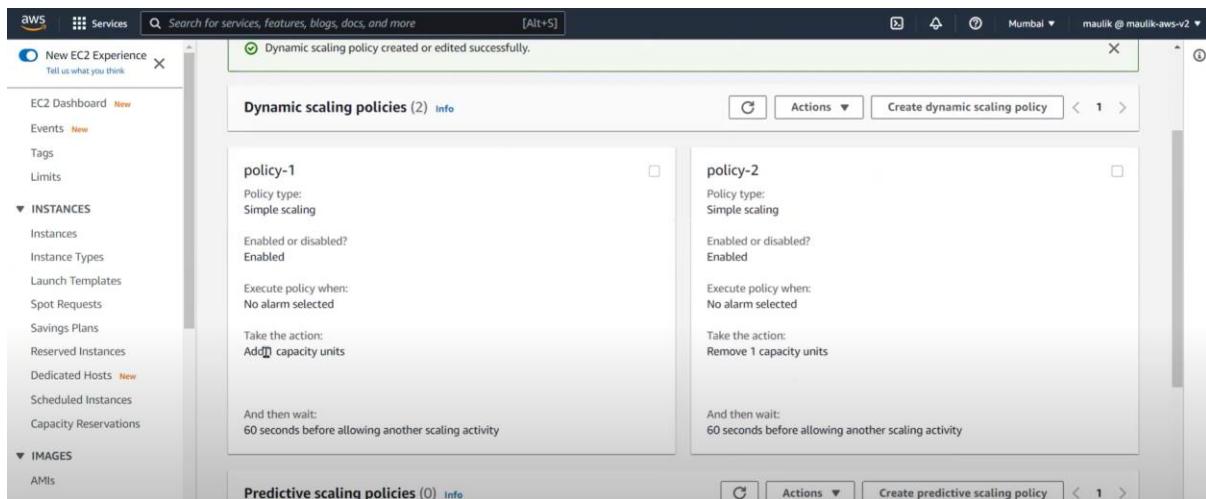


- Enter Policy type as **Simple Scaling**, Policy name as **Policy-1**, Take the action as **Add**, Quantity as **1**, And Then Wait as **60** seconds.





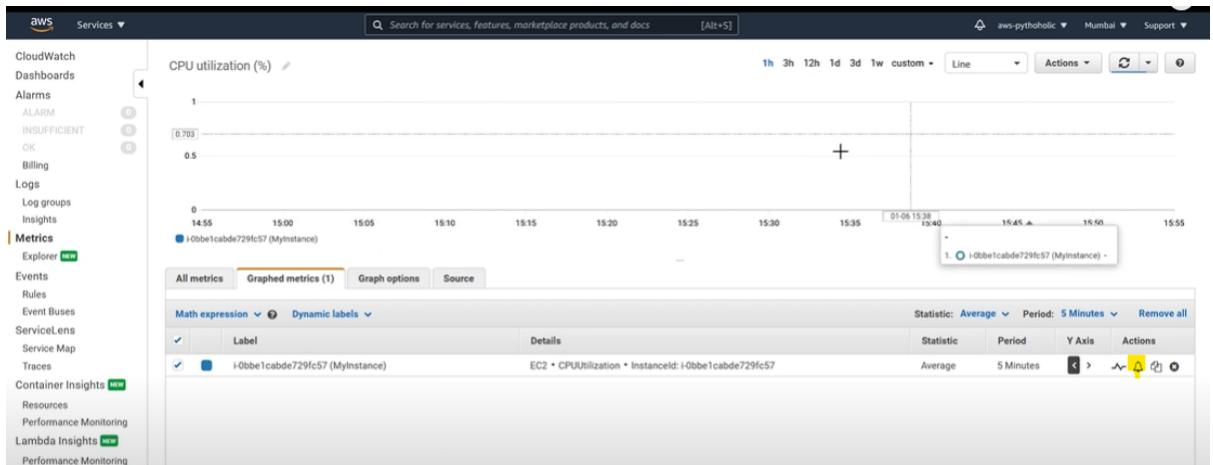
- Similarly, Create another policy. Enter Policy type as **Simple Scaling**, Policy name as **Policy-2**, Take the action as **Remove**, Quantity as **1**, And Then Wait as **60** seconds.



- You can update Desired capacity in Auto-scaling group as '1' in Edit mode to create a new instance for the group.

Step2: Create CloudWatch Alarm

- Go to AWS → CloudWatch → Alarm → Create → our Auto-scaling group.

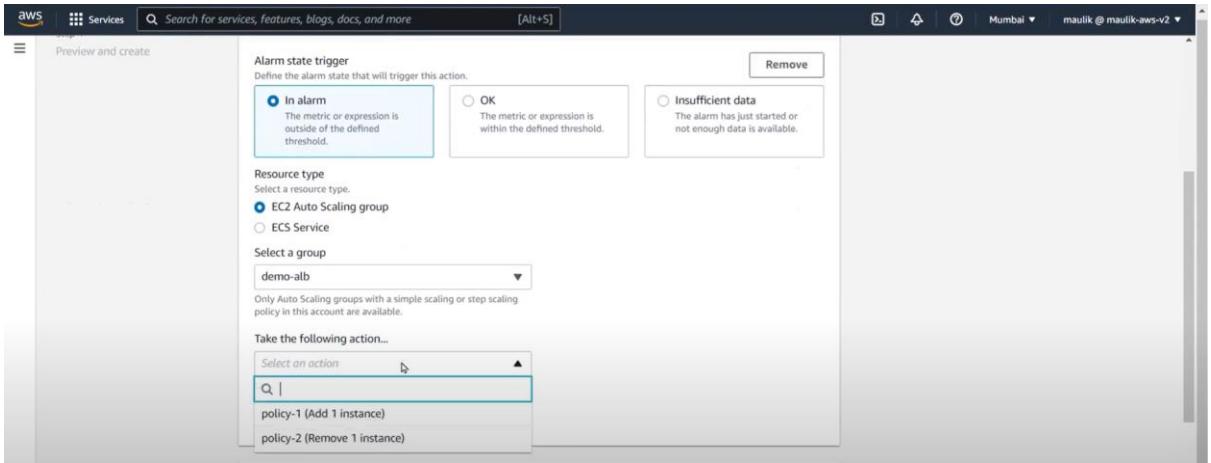


- Enter Metric name, Autoscaling group name, Statistics, Period as **1 minute**, Threshold type as **Static**, Whenever CPUutilization is **greater than 50** & Click **Next**

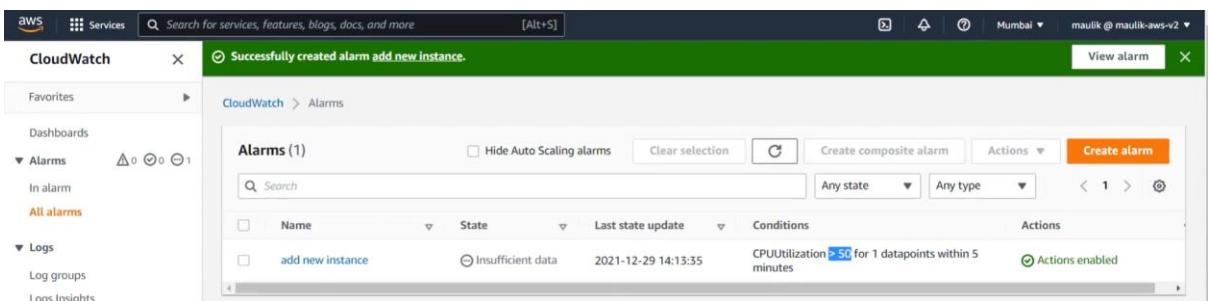
The screenshot shows the "Create alarm" wizard in the AWS CloudWatch Alarms section. It consists of four steps:

- Step 1: Specify metric and conditions**: Shows a graph of CPUUtilization over time. The Y-axis is labeled "Percent" and ranges from 0.28 to 0.31. The X-axis shows hours from 06:30 to 08:30. A blue line represents the metric. To the right, settings are defined: Namespace "AWS/EC2", Metric name "CPUUtilization", AutoScalingGroupName "demo-alb", Statistic "Average", and Period "5 minutes".
- Step 2: Configure actions**: Not visible in the screenshot.
- Step 3: Add name and description**: Not visible in the screenshot.
- Step 4: Preview and create**: Not visible in the screenshot.

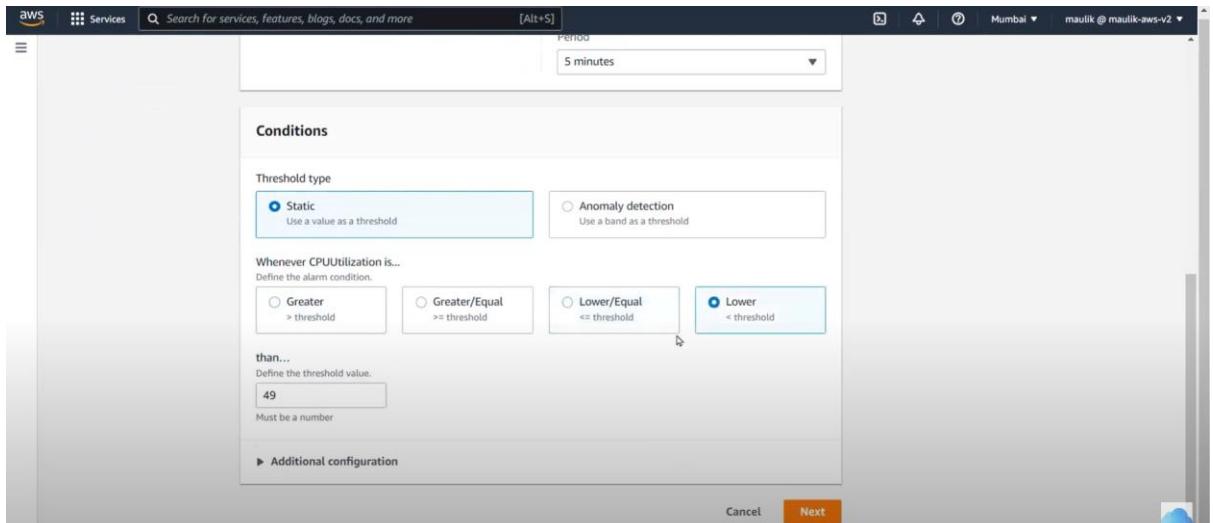
- In Configure actions, Remove Notifications. Click **Add Autoscaling action**.
 - Alarm state trigger as **In-Alarm**, Resource Type as **EC2-Autoscaling group** & Select our **Auto-scaling group**, Take the following action as “**Policy-1**” & click **Next**



- Enter name for Alarm and click **Create Alarm**
- The Alarm will be created with state as “Insufficient Data”.



- You can select & click ‘**Copy**’ to copy this & create another Alarm for **Policy-2**



Name	State	Last state update	Conditions	Actions
remove new instance	In alarm	2021-12-29 14:15:19	CPUUtilization < 50 for 1 datapoints within 5 minutes	Actions enabled
add new instance	OK	2021-12-29 14:14:41	CPUUtilization > 50 for 1 datapoints within 5 minutes	Actions enabled

- If you go to our Auto-scaling group, we can find the alerts attached with respective scaling policies.

Policy Type	Enabled or disabled?	Execute policy when:	Take the action:	And then wait:
Simple scaling	Enabled	breaches the alarm threshold: CPUUtilization > 50 for 1 consecutive periods of 300 seconds for the metric dimensions: AutoScalingGroupName = demo-alb	Add 1 capacity units	60 seconds before allowing another scaling activity
Simple scaling	Enabled	remove new instance breaches the alarm threshold: CPUUtilization < 50 for 1 consecutive periods of 300 seconds for the metric dimensions: AutoScalingGroupName = demo-alb	Remove 1 capacity units	60 seconds before allowing another scaling activity

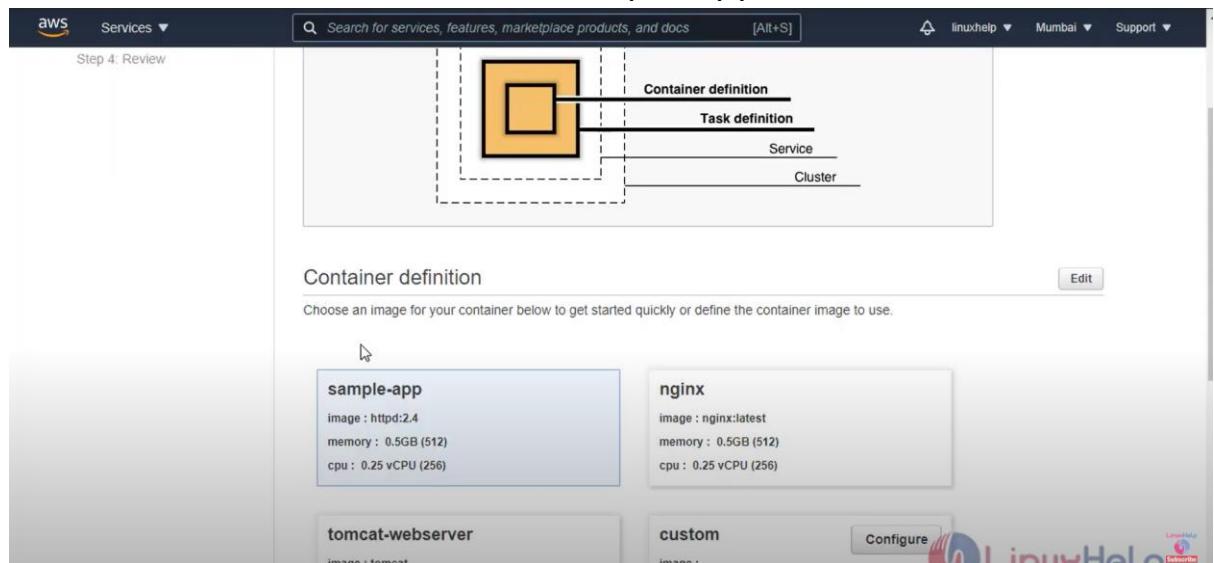
- To test this, install stress application in the Ec2 instance.
Cmd: Sudo amazon-linux-extras install epel -y
Sudo yum install stress -y
Stress –help
Stress –cpu 4 –timeout 180

11. Amazon Elastic Container Service(ECS creation)

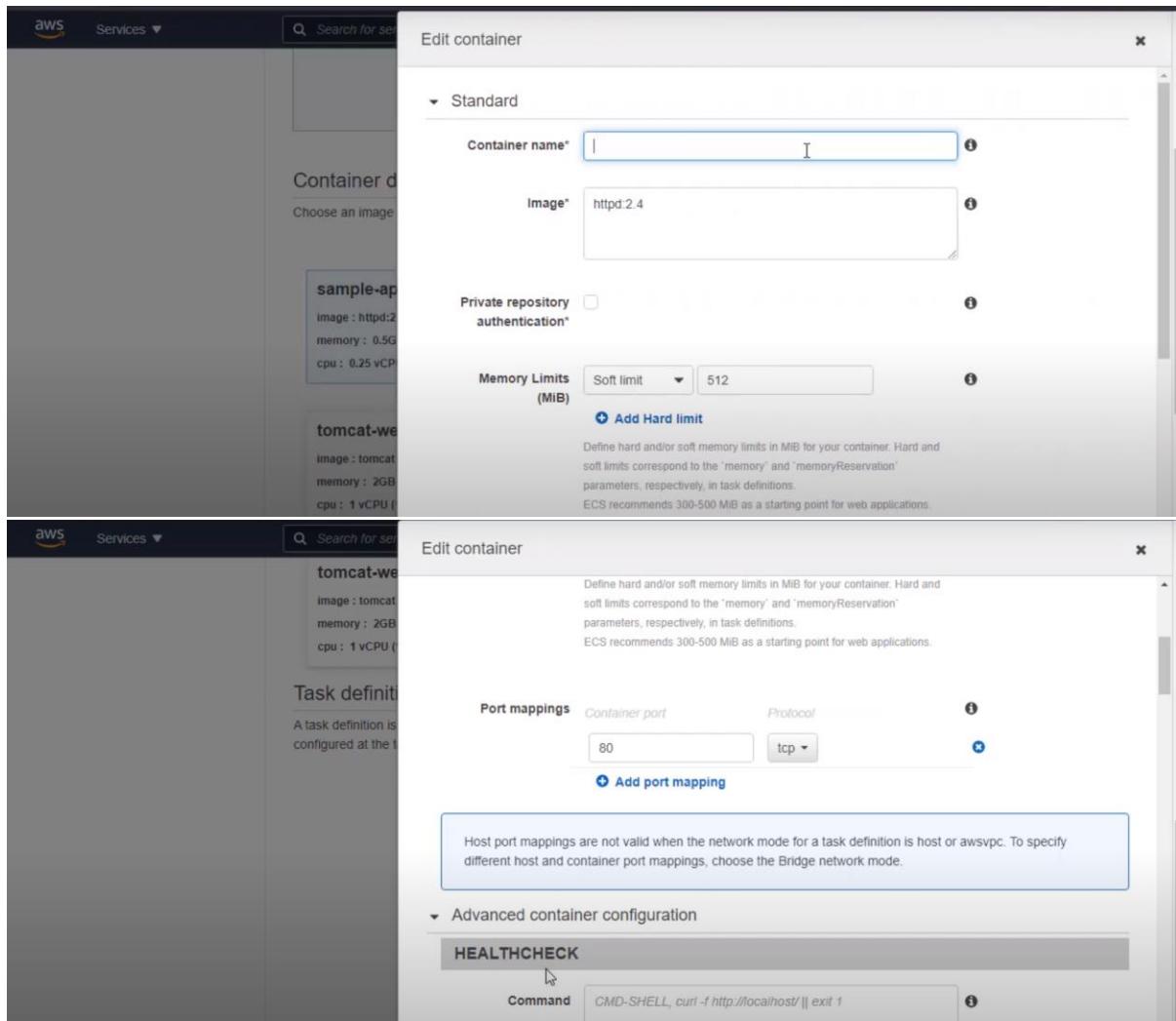
Amazon Elastic Container Service(ECS) is a highly scalable, faster container management service that makes it easy to run, stop & manage containers on a cluster. Your containers are defined in a task definition that you use to run individual tasks or tasks within a service. Amazon ECS enables you to launch and stop your container-based application by using simple API calls. You can also retrieve the state of your cluster from a centralized service & have access to many familiar Amazon EC2 features.

Steps:

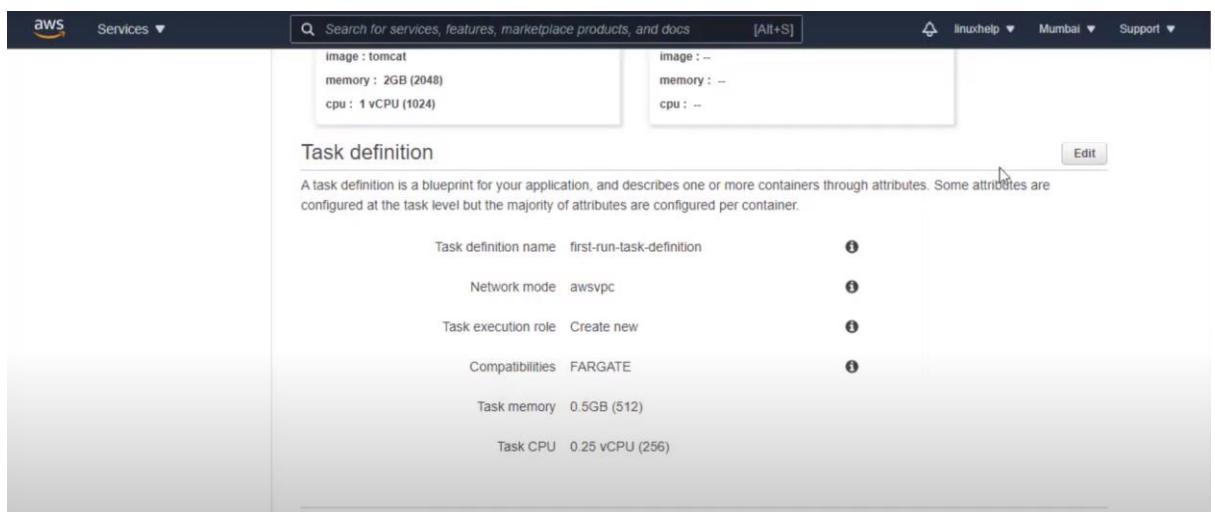
- Go to AWS → Elastic Container Service → Get started
- In Container definition, choose Sample App



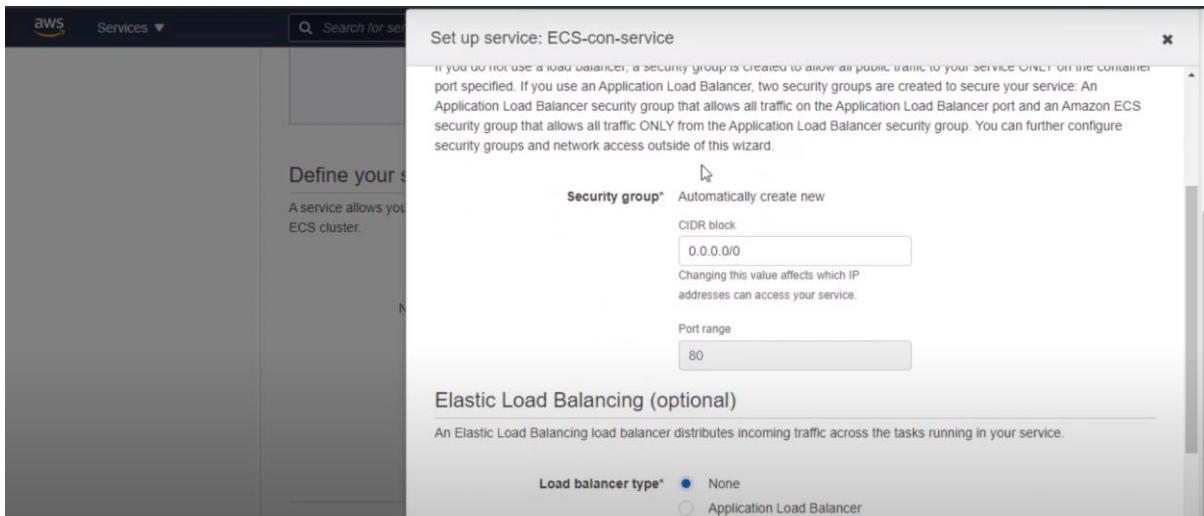
- Click Edit and Enter container name, you can also update memory limits, health check if required.



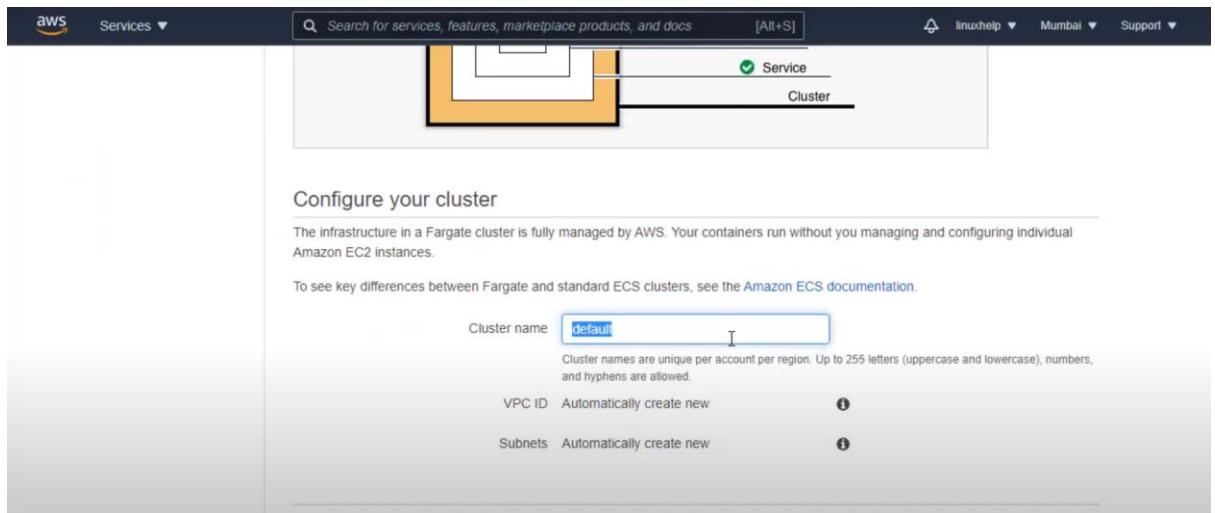
- In **Task definition**, you can update the Task memory & Task CPU & click Next



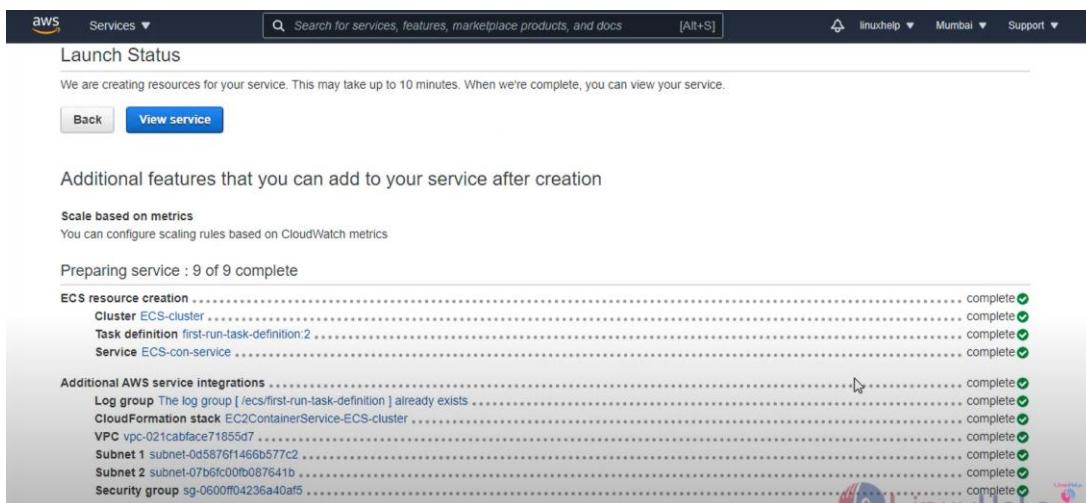
- In **Service definition**, you can enter Service name, Security group, Port range, ELB if required & then click Save.



- In **Configure your cluster**, You can give your cluster name, VPC & Subnets if required and click Next



- You can review the configurations in the next screen & then click Create



- Once the ECS Service is created, click **View service**

Service : ECS-con-service

Cluster	ECS-cluster	Desired count	1
Status	ACTIVE	Pending count	1
Task definition	first-run-task-definition:2	Running count	0
Service type	REPLICA		
Launch type	FARGATE		
Service role	AWSServiceRoleForECS		
Created By	arn:aws:iam::633544438557:root		

Load Balancing

Load Balancer Name	Container Name	Container Port
No load balancers		

- Go to Tasks → Select the Task definition created
- In selected task, go To Network → Select ENI ID to open the network interface.

Name	Network interface ID	Subnet ID	VPC ID	Availability Zone
-	eni-0efcd65ee28060875	subnet-07b6fc00fb087641b	vpc-021cabface71855d7	ap-south-1b

Public IPv4 address: 13.233.104.24
Public IPv4 DNS: ec2-13-233-104-24.ap-south-1.compute.amazonaws.com
IPv6 addresses: -
Secondary private IPv4 addresses: -
Association ID: -
MAC address: -
Elastic IP address owner: amazon

- Use the public IP to view the container running on web.
- To delete this configurations, delete the cluster and task definition created.

12. AWS DEVOPS

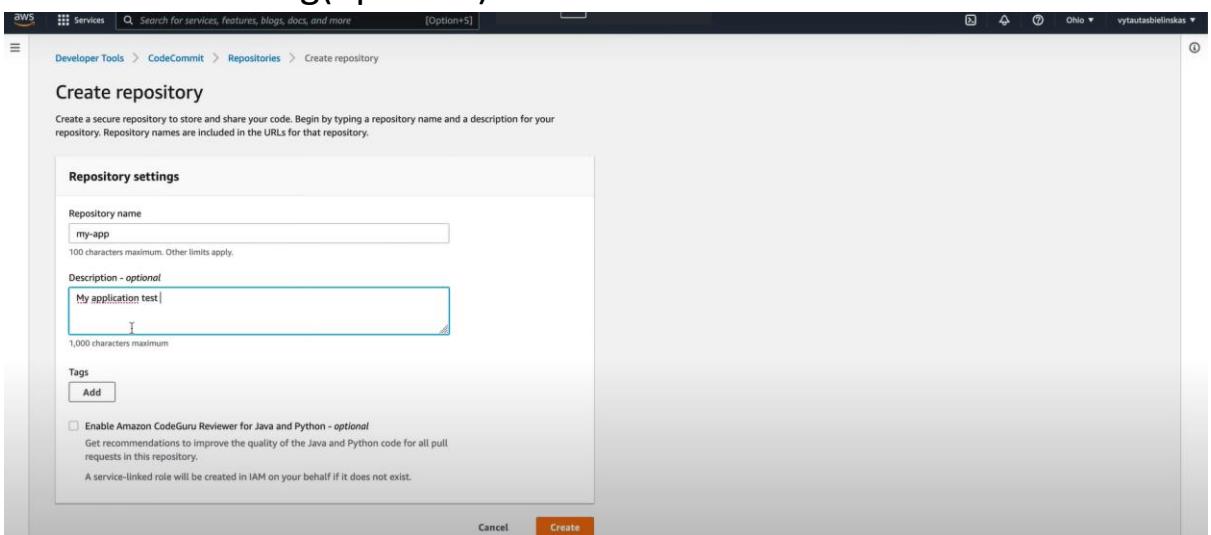
Code Commit:

Code Commit is the process used to store different formats of code/files in AWS repository. It is fully managed by AWS, Secured, has High availability.

Steps:

Step1: Create a New Repository

- Go to AWS Management Console and Search “Code Commit”
- Developer Tools → Code Commit → Repositories & click Create Repository.
- Enter Name & Tag(optional)



- You can select either HTTPS/SSH connection.
As root user, you cannot create SSH connection. To use SSH connection create a new user apart from Root user.

Step2: Create a new user

- Go to IAM → User → New User

- Enter User Name, Enable Access Key & Password

Add user

Set user details

User name* vytautas

Select AWS access type

Select how these users will primarily access AWS. If you choose only programmatic access, it does NOT prevent users from accessing the console using an assumed role. Access keys and autogenerated passwords are provided in the last step. [Learn more](#)

Select AWS credential type*

- Access key - Programmatic access Enables an access key ID and secret access key for the AWS API, CLI, SDK, and other development tools.
- Password - AWS Management Console access Enables a password that allows users to sign-in to the AWS Management Console.

Console password*

- Autogenerated password
- Custom password

Require password reset User must create a new password at next sign-in. Users automatically get the [IAMUserChangePassword](#) policy to allow them to change their own password.

- In the Next screen, create a new group(IAM-codecommit) with Administration Access Policy attached & add the user to the group.

Add user

Create group

Create a group and select the policies to be attached to the group. Using groups is a best-practice way to manage users' permissions by job functions, AWS service access, or your custom permissions. [Learn more](#)

Group name iam-group

Create policy Refresh

Filter policies Search Showing 728 results

Policy name	Type	Used as	Description
<input checked="" type="checkbox"/> AdministratorAccess	Job function	None	Provides full access to AWS services and resources.
<input type="checkbox"/> AdministratorAccess-Amplify	AWS managed	None	Grants account administrative permissions while explicitly allowing direct access to resources needed by A...
<input type="checkbox"/> AdministratorAccess-AWSElasticBeanstalk	AWS managed	None	Grants account administrative permissions. Explicitly allows developers and administrators to gain direct ac...
<input type="checkbox"/> alb-policy	Customer managed	Permissions policy (1)	Policy copied from dataservice-ibm repo for kubernetes pipeline.
<input type="checkbox"/> AlexaForBusinessDeviceSetup	AWS managed	None	Provide device setup access to AlexaForBusiness services
<input type="checkbox"/> AlexaForBusinessFullAccess	AWS managed	None	Grants full access to AlexaForBusiness resources and access to related AWS Services
<input type="checkbox"/> AlexaForBusinessGatewayExecution	AWS managed	None	Provide gateway execution access to AlexaForBusiness services
<input type="checkbox"/> AlexaForBusinessLifesizeDelegatedAccessPolicy	AWS managed	None	Provide access to Lifesize AVS devices
<input type="checkbox"/> AlexaForBusinessPolyDelegatedAccessPolicy	AWS managed	None	Provide access to Poly AVS devices
<input type="checkbox"/> AlexaForBusinessReadDelegatedAccess	AWS managed	None	Provides read-only access to AlexaForBusiness resources

Cancel Create group

- Click Create User & download the Access Key & Password in .CSV format.

- Open the User created and go to Security credentials to generate the HTTPS Git credentials for CodeCommit.

The screenshot shows the AWS Identity and Access Management (IAM) console. On the left, there's a navigation sidebar with options like Dashboard, Access management, Users (which is selected), Roles, Policies, Identity providers, Account settings, Access reports, and Credential report. The main content area is titled 'Access keys' and shows a table of existing access keys. One key is listed: 'Access key ID: AKIAQWIN6X2GXWRKJGV', 'Created: 2022-01-18 15:31 UTC+0200', and 'Last used: N/A'. The status is 'Active'. Below the table, there's a section for 'SSH keys for AWS CodeCommit' with a button to 'Upload SSH public key'. Under 'HTTPS Git credentials for AWS CodeCommit', there's a 'Generate credentials' button. A note says 'No credentials have been generated.' At the bottom, there's another 'Generate credentials' button. The URL in the browser is https://console.aws.amazon.com/iamv2/home?region=us-east-1#/users/akiaqwin6x2gxwrkjgv/security_credentials.

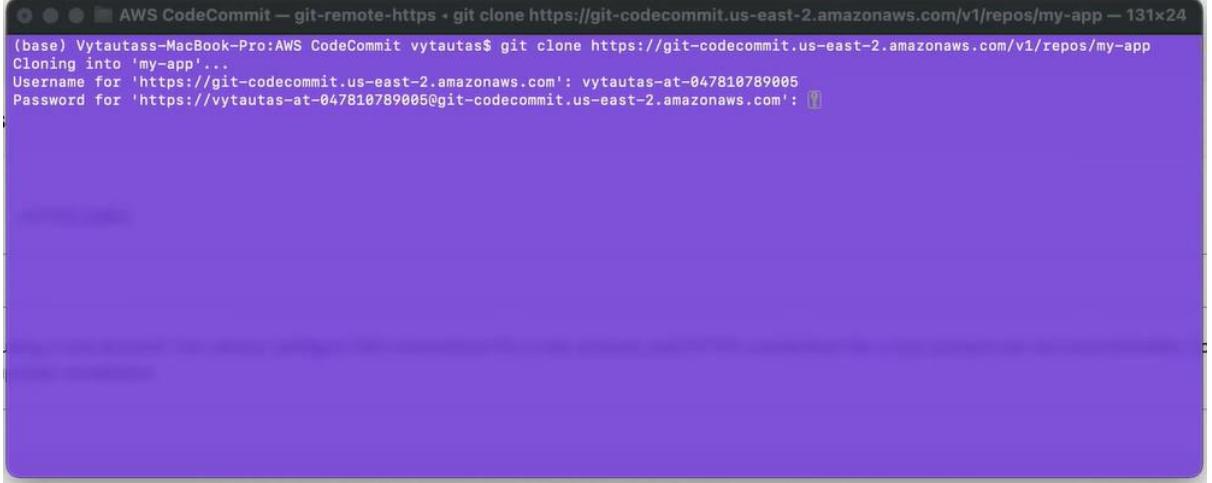
- Click Generate Credentials & download the same.

This screenshot shows a modal window titled 'Generate credentials' over the IAM interface. The modal contains a green success message: 'Your new credentials are available'. It also includes instructions to 'Save your user name and password now (or download a credentials file)'. Below this, it says 'This is the only time the password can be viewed or downloaded. You cannot recover it later. However, you can reset your password at any time.' It shows the generated credentials: 'User name: vytutais-at-047810789005' and 'Password: *****'. There's a 'Download credentials' button with a download icon. At the bottom right of the modal is a 'Close' button. The background of the modal is semi-transparent, showing parts of the IAM dashboard.

Step3: Clone repository

- Go to the created Repository and click Clone URL → Clone HTTPS & copy the HTTPS URL
- Open Linux CLI in the local system
- Cmd: git clone <https url>

- Give the Username & Password for Repository in Linux CLI when prompted



AWS CodeCommit — git-remote-https · git clone https://git-codecommit.us-east-2.amazonaws.com/v1/repos/my-app — 131x24

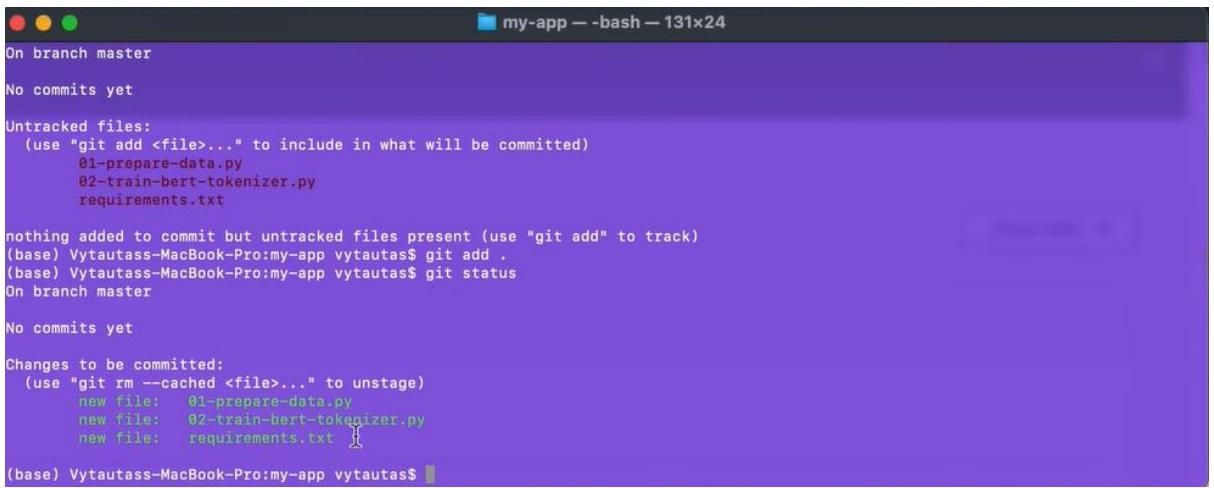
```
(base) Vytautass-MacBook-Pro:AWS CodeCommit vytautas$ git clone https://git-codecommit.us-east-2.amazonaws.com/v1/repos/my-app
Cloning into 'my-app'...
Username for 'https://git-codecommit.us-east-2.amazonaws.com': vytautas-at-047810789005
Password for 'https://vytautas-at-047810789005@git-codecommit.us-east-2.amazonaws.com': [REDACTED]
```

- Cmd: ls to view the cloned repository in Linux cli.

Step4: Push to CodeCommit Repository

- Open the cloned repository, store some files/create some files
- Once the files are created/stored,
use cmd: git add .

git commit -m “files added to repository”



```
my-app — -bash — 131x24

On branch master
No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    01-prepare-data.py
    02-train-bert-tokenizer.py
    requirements.txt

nothing added to commit but untracked files present (use "git add" to track)
(base) Vytautass-MacBook-Pro:my-app vytautas$ git add .
(base) Vytautass-MacBook-Pro:my-app vytautas$ git status
On branch master
No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   01-prepare-data.py
    new file:   02-train-bert-tokenizer.py
    new file:   requirements.txt [REDACTED]

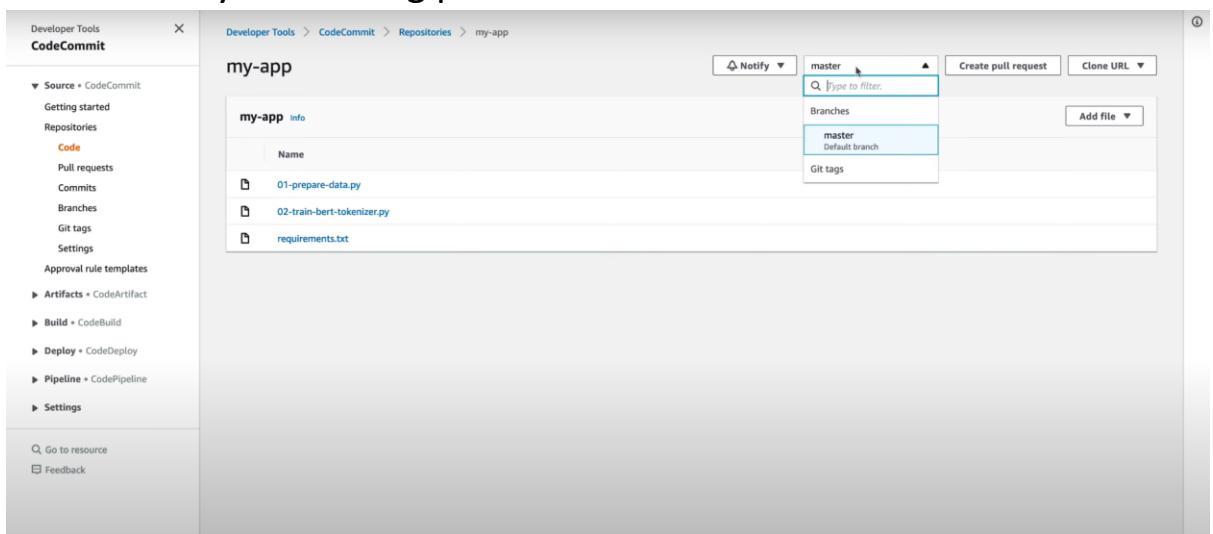
(base) Vytautass-MacBook-Pro:my-app vytautas$
```

```
(base) Vytautass-MacBook-Pro:my-app vytautas$ git commit -m "my first commit on AWS CodeCommit"
[master (root-commit) 5b92afdf] my first commit on AWS CodeCommit
 3 files changed, 62 insertions(+)
 create mode 100644 01-prepare-data.py
 create mode 100644 02-train-bert-tokenizer.py
 create mode 100644 requirements.txt
(base) Vytautass-MacBook-Pro:my-app vytautas$
```

- Once the files are committed to Local repo, push it to AWS Repo using cmd: git push origin master/main

```
(base) Vytautass-MacBook-Pro:my-app vytautas$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 1.18 KiB | 1.18 MiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
To https://git-codecommit.us-east-2.amazonaws.com/v1/repos/my-app
 * [new branch]      master -> master
(base) Vytautass-MacBook-Pro:my-app vytautas$
```

- If you view the AWS codecommit repo, you can see the files from Local system being pushed there.



- Repeat step4 for every change you make in the local system/folder.

Code Build :

Code Build is the AWS service which is used to build any code getting the code from Source repository and providing artifacts as output.

Steps:

Step1: Create Code Build Project

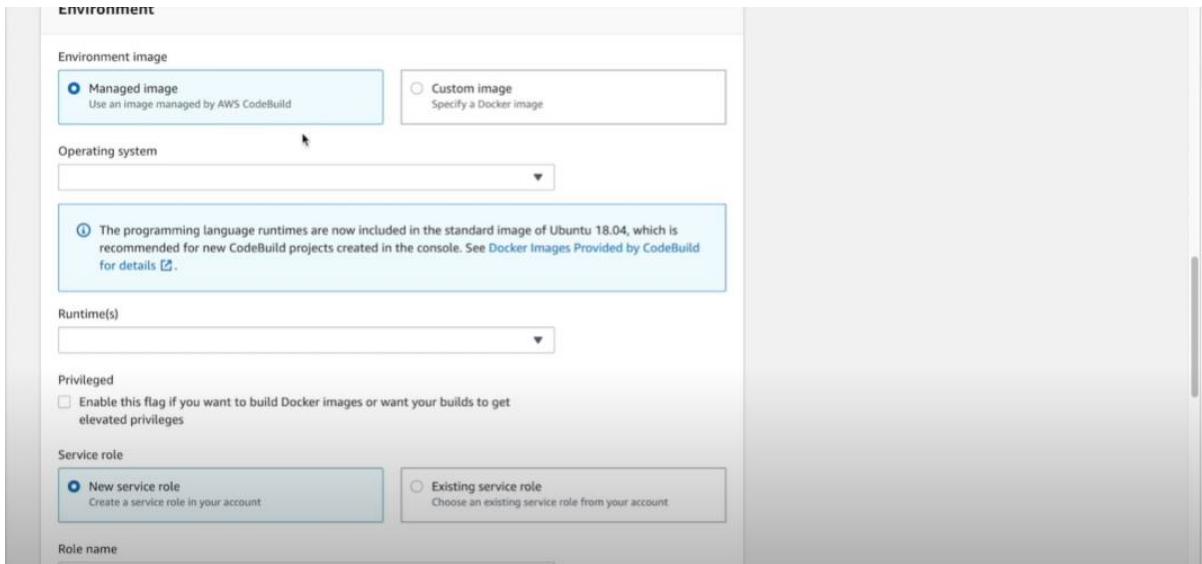
- Go to AWS Management Console and Search “Code Build”
- Developer Tools → Code Build → click Create Project.
- Enter Project Name & Description(optional)



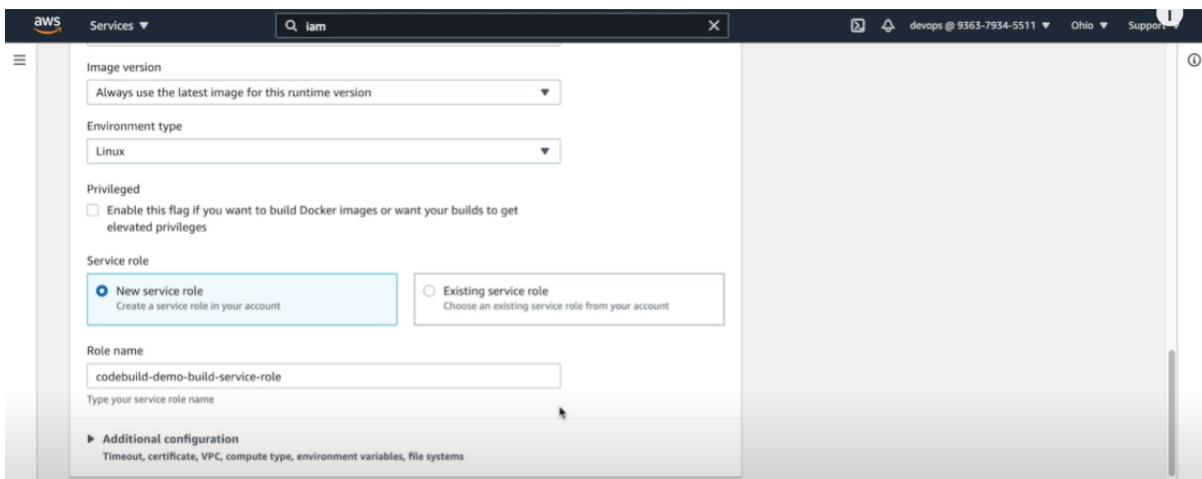
- Give Source as CodeCommit repository or any source code repository and provide the URL for the same.

The screenshot shows the 'Source' configuration page for a CodeBuild project. The 'Source' tab is active. Under 'Source provider', 'GitHub' is selected. The 'Repository' section shows 'Public repository' is chosen, and the 'Repository URL' field contains a placeholder URL: 'https://github.com/<user-name>/<repository-name>'. Below this, a message indicates a successful connection to GitHub using OAuth. At the bottom, there's an optional 'Source version - optional info' field with a note about entering a pull request, branch, commit ID, tag, or reference and a commit ID.

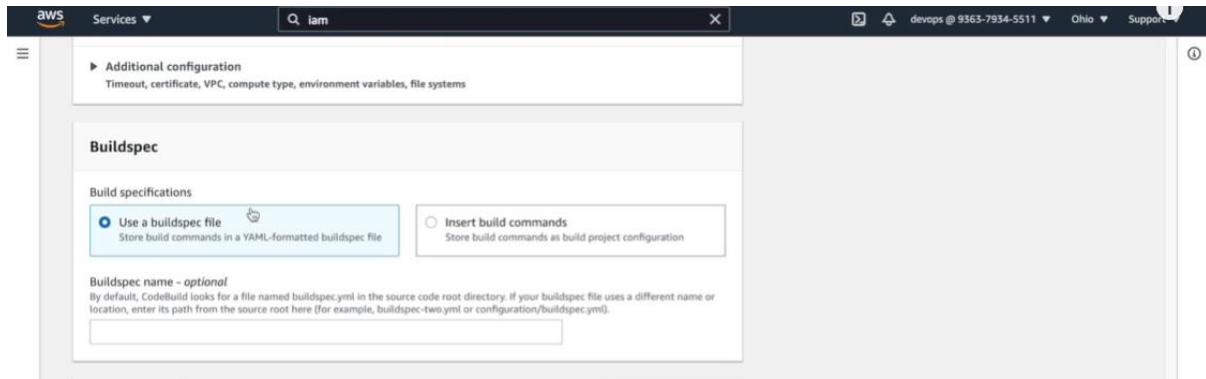
- Choose the Environment as Managed Image/Custom Image, Operating system as Amazon Linux, Runtime as Standard, Image as suggested.



- Also, you can add new Service role/existing service role for this build action.



- You can also add BuildSpec file, this is similar to Dockerfile, you can specify details like how to build this project.



- Build file has Pre-build & Build phase & also we can mention the files to be stored in the Artifacts section

```
buildspec.yaml
1  version: 0.2
2
3  phases:
4    pre_build:
5      commands:
6        - echo Nothing to do in the pre_build phase...
7    build:
8      commands:
9        - echo Build started on `date`
10       - mvn install
11
12  artifacts:
13    files:
14      - '**/LoginWebApp-1.war'
15      - 'appspec.yaml'
16    discard-paths: yes
```

- We can add the artifact type and where we want to store the artifacts from the Code Build of this project.

Artifacts

Artifact 1 - Primary

Type: Amazon S3

Bucket name: java-artifacts-devops4solutions

Name: The name of the folder or compressed file in the bucket that will contain your output artifacts. Use Artifacts packaging under Additional configuration to choose whether to use a folder or compressed file. If the name is not provided, defaults to project name.

Path - optional: The path to the build output ZIP file or folder. Example: MyPath/MyArtifact.zip.

Namespace type - optional: None

Artifacts packaging:

- None: The artifact files will be uploaded to the bucket.
- Zip: AWS CodeBuild will upload artifacts into a compressed file that is put into the specified bucket.

Disable artifact encryption: Disable encryption if using the artifact to publish a static website or sharing content with others

Additional configuration

Cache, encryption key

- Click Build Project

Logs

CloudWatch

CloudWatch logs - optional: Checking this option will upload build output logs to CloudWatch.

Group name: [Input field]

Stream name: [Input field]

S3

S3 logs - optional: Checking this option will upload build output logs to S3.

Create build project

- Once the project is created, it will be added to the dashboard and has details like Source code repository, where to store artifacts.

- Click on Start Build to initiate the build process.
- We can also use Notify for notifications, Edit for Editing configuration & Project, Build Logs to view the log for the project build.
- We can also view multiple builds of the project using Build History.

Build run	Status	Project	Build number	Source version	Submitter	Duration	Completed
sample-javaprojects:a2b52074-74d7-4f7e-9dc6-ae63467bd509	Succeeded	sample-javaprojects	33	arn:aws:s3::codipeline-us-east-2-792249311510/java-sample/SourceArti/oWT5BX7	codepipeline/e/java-sample	39 seconds	1 day ago
sample-javaprojects:e693d1b3-636a-4156-a725-7b1806f6bf6d	Succeeded	sample-javaprojects	32	arn:aws:s3::codipeline-us-east-2-792249311510/java-sample/SourceArti/HoArFZ2	codepipeline/e/java-sample	44 seconds	1 day ago

Code Deploy

AWS CodeDeploy is a fully managed deployment service that automates software deployments to a variety of compute services such as Amazon EC2, AWS Fargate, AWS Lambda, and your on-premises servers.

CodeDeploy makes it easier for you to rapidly release new features, helps you avoid downtime during application deployment, and handles the complexity of updating your applications.

Centralized control - AWS CodeDeploy allows you to easily launch and track the status of your application deployments through AWS management console.

Minimize downtime - AWS CodeDeploy helps maximize your application availability during the software deployment process.

Steps:

Step1: Create IAM Role & Ec2 instances

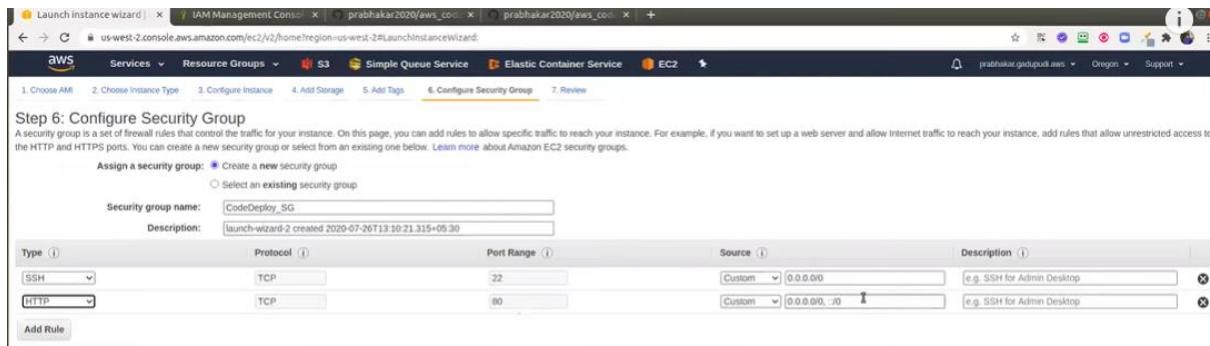
- Create IAM roles for CodeDeployRole & EC2CodeDeploy and attach it to the EC2 Instances which we are creating for Code Deploy. Policies required:

- Amazon EC2 Auto Scaling
- AWS CodeDeploy
- Amazon Elastic Compute Cloud
- Elastic Load Balancing
- AWS Identity and Access Management
- Amazon Simple Storage Service
- Amazon Simple Notification Service
- Amazon CloudWatch

- User data:

```
#!/bin/bash
sudo yum -y update
sudo yum -y install ruby
sudo yum -y install wget
cd /home/ec2-user
wget https://aws-codedeploy-ap-south-1.s3.ap-south-1.amazonaws.com/latest/install
sudo chmod +x ./install
sudo ./install auto
sudo yum install -y python-pip
sudo pip install awscli
```

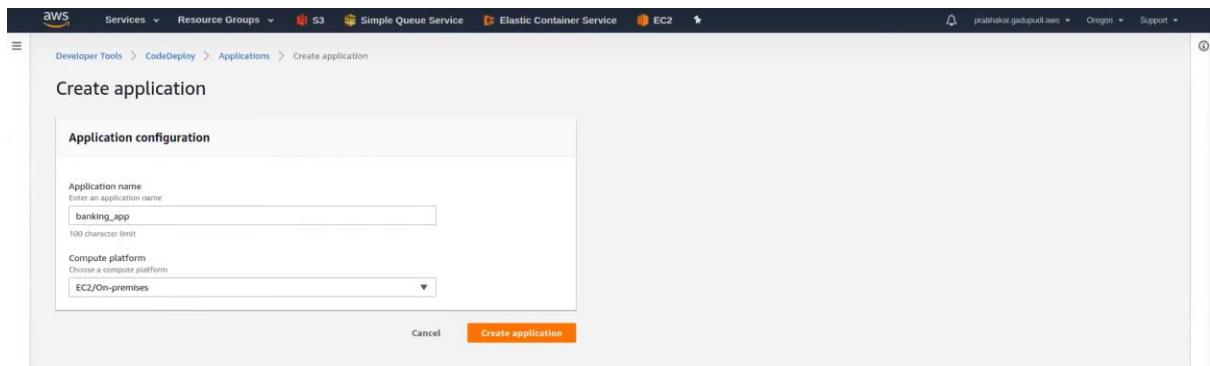
- Create New security group to expose port 80 (HTTP) for the Code Deployment instances



- Click Review & Launch Instance.

Step2: Code Deploy Application

- Go to AWS Management Console and Search “Code Deploy”
- Developer Tools → Code Deploy → Applications → Click Create Application
- Enter the Application name & Choose Platform as one of them(AWS Lambda, AWS ECS, AWS EC2) as required



Step3: Code Deploy: Deployment group creation

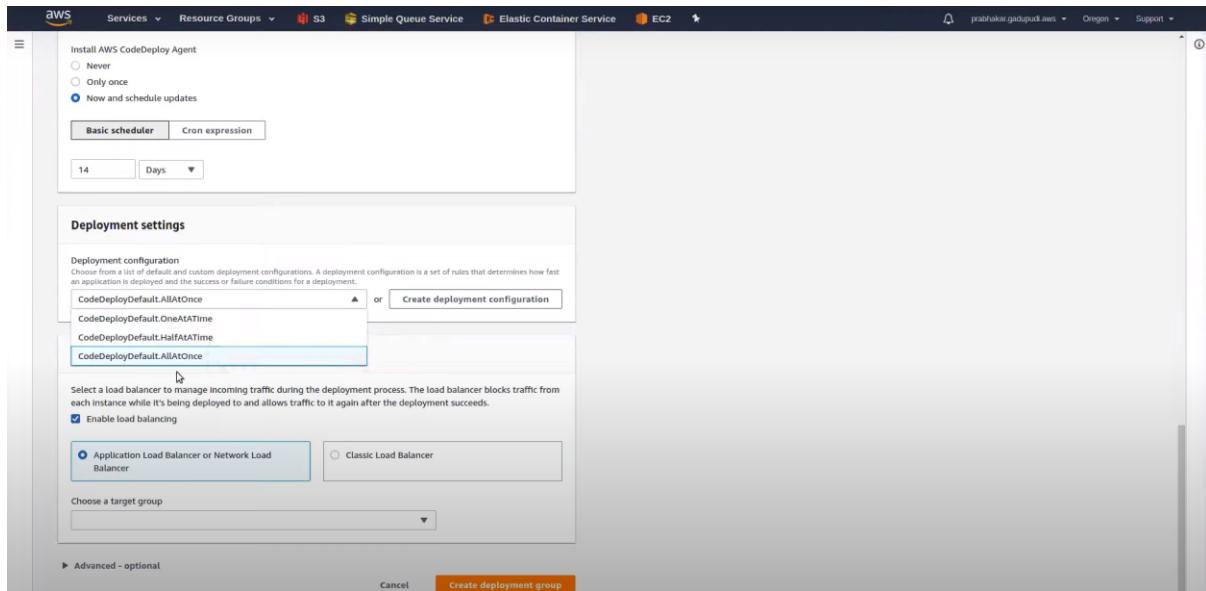
- Open the application created & click *Deployment group*
- Enter the Application deployment group name, Service Role as required (which is created to grant AWS codedeploy access to your target instances), Deployment Type as *In-Place*

The screenshot shows the AWS CodeDeploy console interface. At the top, there's a navigation bar with 'Services' dropdown, 'Resource Groups' dropdown, 'S3', 'Simple Queue Service', 'Elastic Container Service', 'EC2', and user information 'prabhar.gadupudi.us Oregon Support'. Below the navigation, there's a sidebar with 'Application banking_app Compute type EC2/On-premises'. The main area has sections for 'Deployment group name' (containing 'banking_app_deploymentgroup'), 'Service role' (containing 'arn:aws:iam::327585364132:role/CodeDeployRole'), and 'Deployment type' (with 'In-place' selected). A note says 'Enter a service role with CodeDeploy permissions that grants AWS CodeDeploy access to your target instances.'

- In Environment Configuration, Choose *Amazon Ec2 Instances* and give the Tag as *Tag_name*(which is created for newly launched machines)

The screenshot shows the 'Environment configuration' step of the deployment process. It asks to 'Select any combination of Amazon EC2 Auto Scaling groups, Amazon EC2 instances, and on-premises instances to add to this deployment'. Under 'Amazon EC2 Instances', the checkbox is checked. Below it, there's a note about adding tags: 'You can add up to three groups of tags for EC2 instances to this deployment group.' A 'Tag group 1' is being configured with a key 'Name' and a value 'Tag_group_1'. There are buttons for 'Add tag' and '+ Add tag group'. At the bottom, there's an option for 'On-premises instances'.

- In Deployment settings, choose *Deploy All At Once, Enable or Disable ALB* based on requirement & click *Create Deployment group*



Code Deploy Setup is done

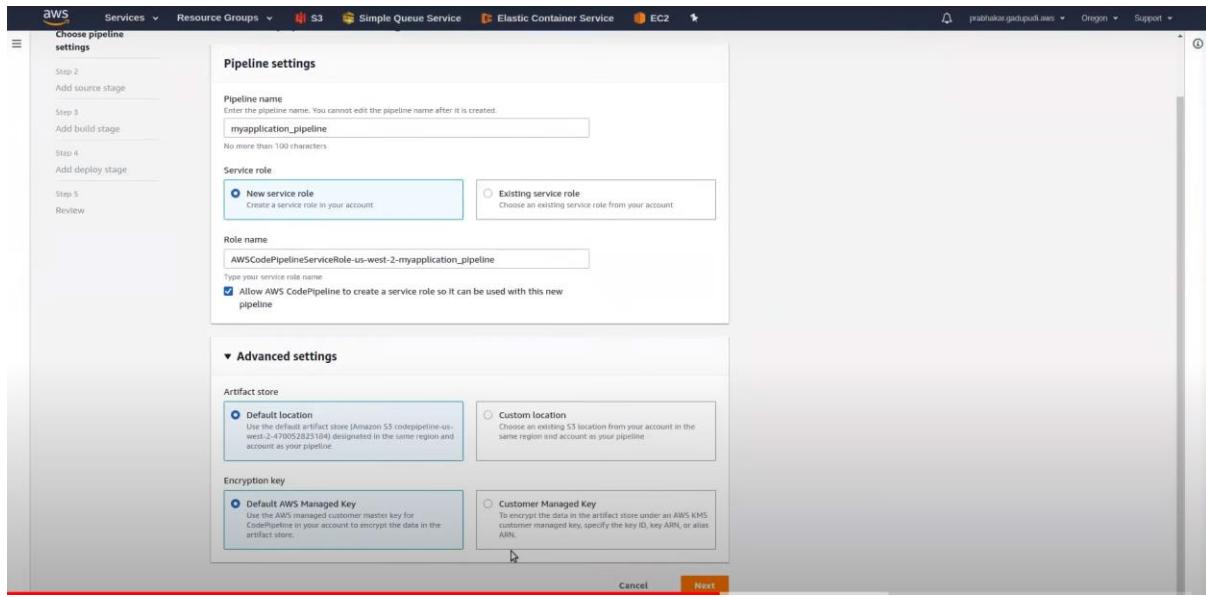
Code Pipeline

AWS CodeDeploy is a fully managed pipeline service that automates application build, deployment using other AWS services like CodeCommit, CodeBuild, CodeDeploy

Steps:

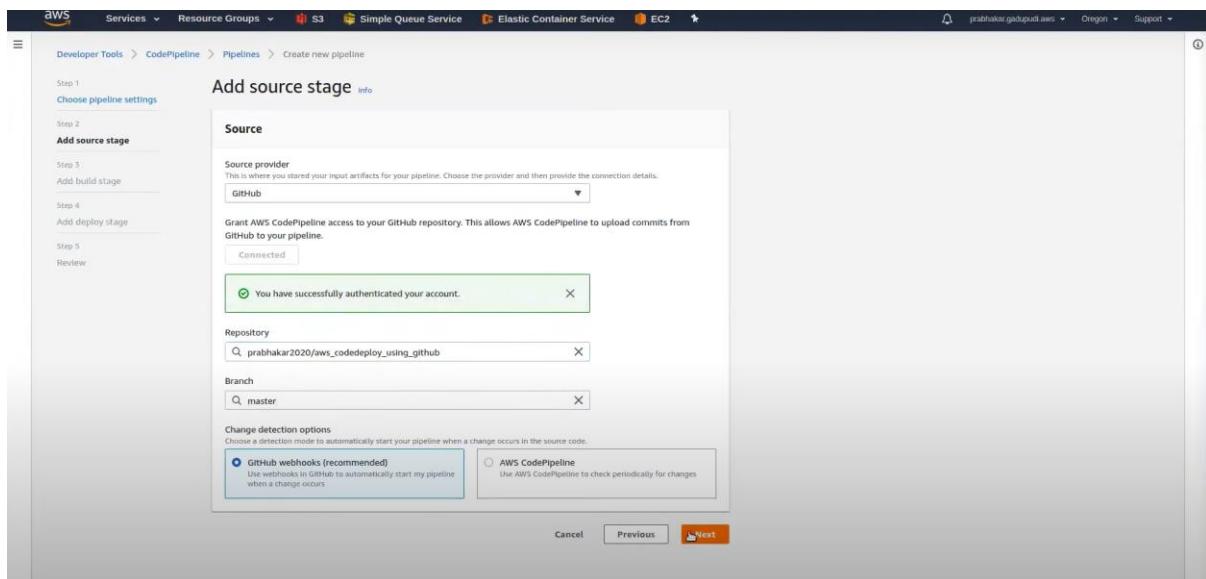
Step1: Create Code Pipeline

- Go to AWS Management Console and Search “Code Pipeline”
- Developer Tools → Code Pipeline → Pipelines → Click Create New pipeline
- Enter the Pipeline name as *Name*, Service role as *New Service Role*, Artifact store as *Default* & Encryption key as *Default* & click *Next*



Step2: Source Stage

- Enter Source Provider as *Github*, Enable *Github connection*, Repository as *public_github_repository*, Branch as *Master*, then click *Next*

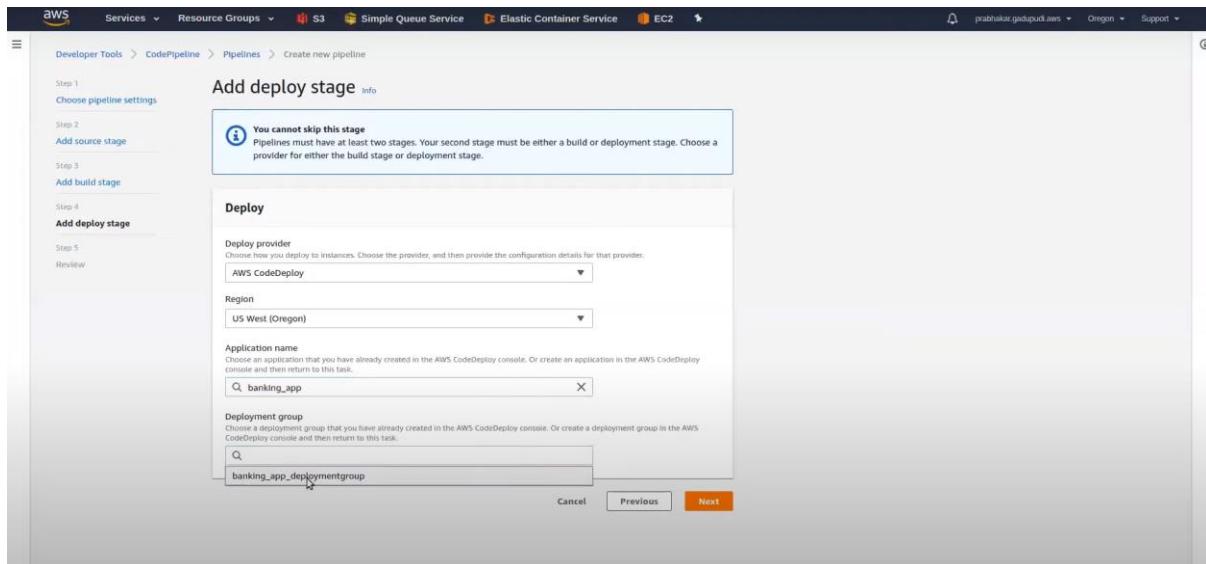


Step3: Build Stage

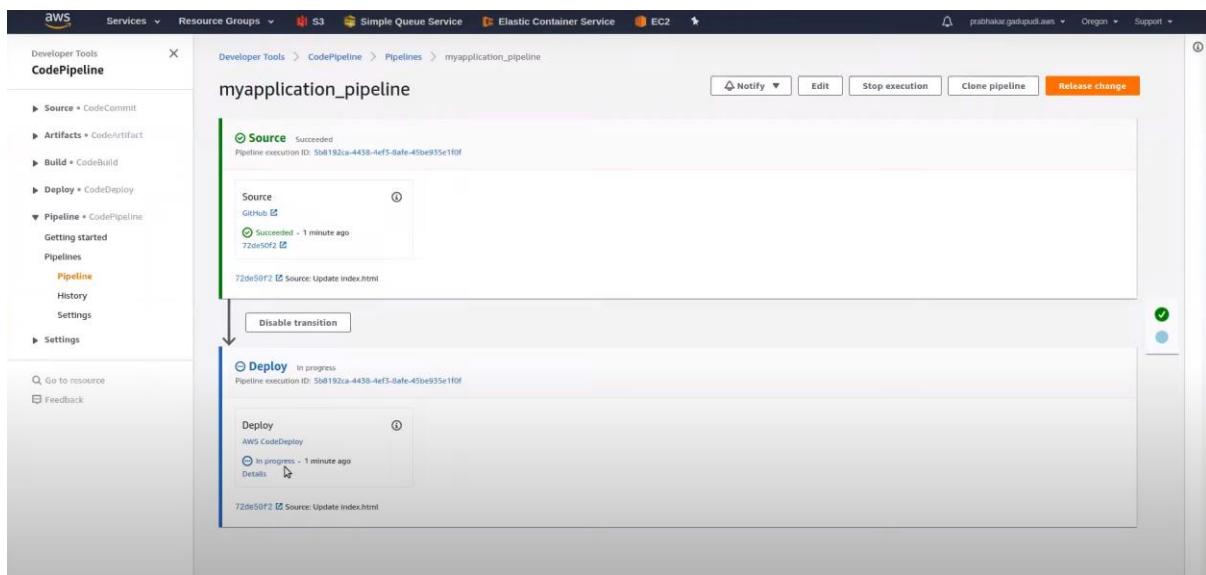
- If Project doesn't need build, you can ignore Build or choose between *Code Build or Jenkins*

Step4: Deploy Stage

- In Deploy stage, choose Deploy Provider as *AWS CodeDeploy*, Region as *required*, Application name as *application_name*, Deployment group as *Deployment group*.



- Then click create Pipeline & Execute



Note: Since we didn't have a build stage, make sure that all the dependencies are already in the Instances created for easy deployment.

