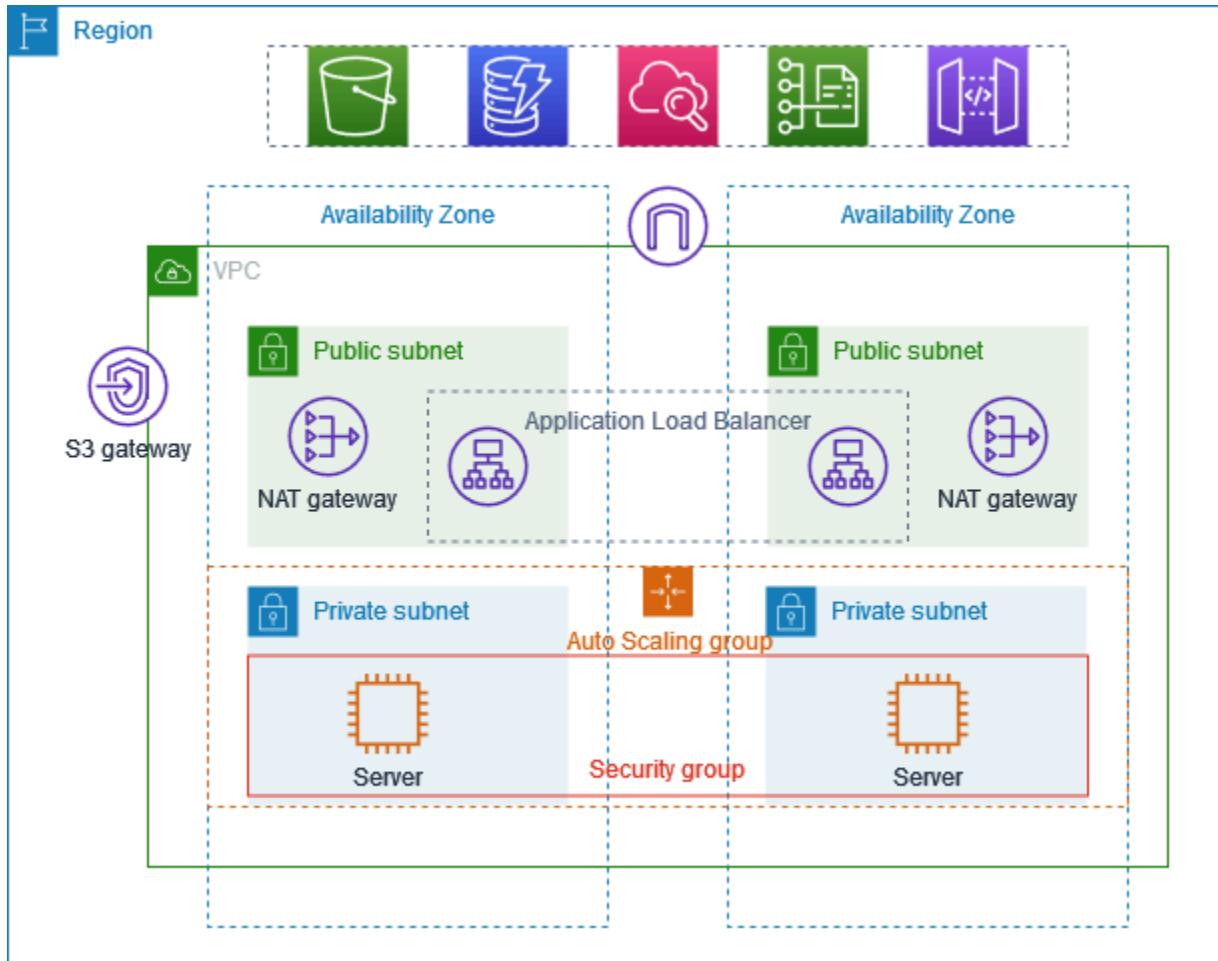


# VPC with servers in private subnets and NAT



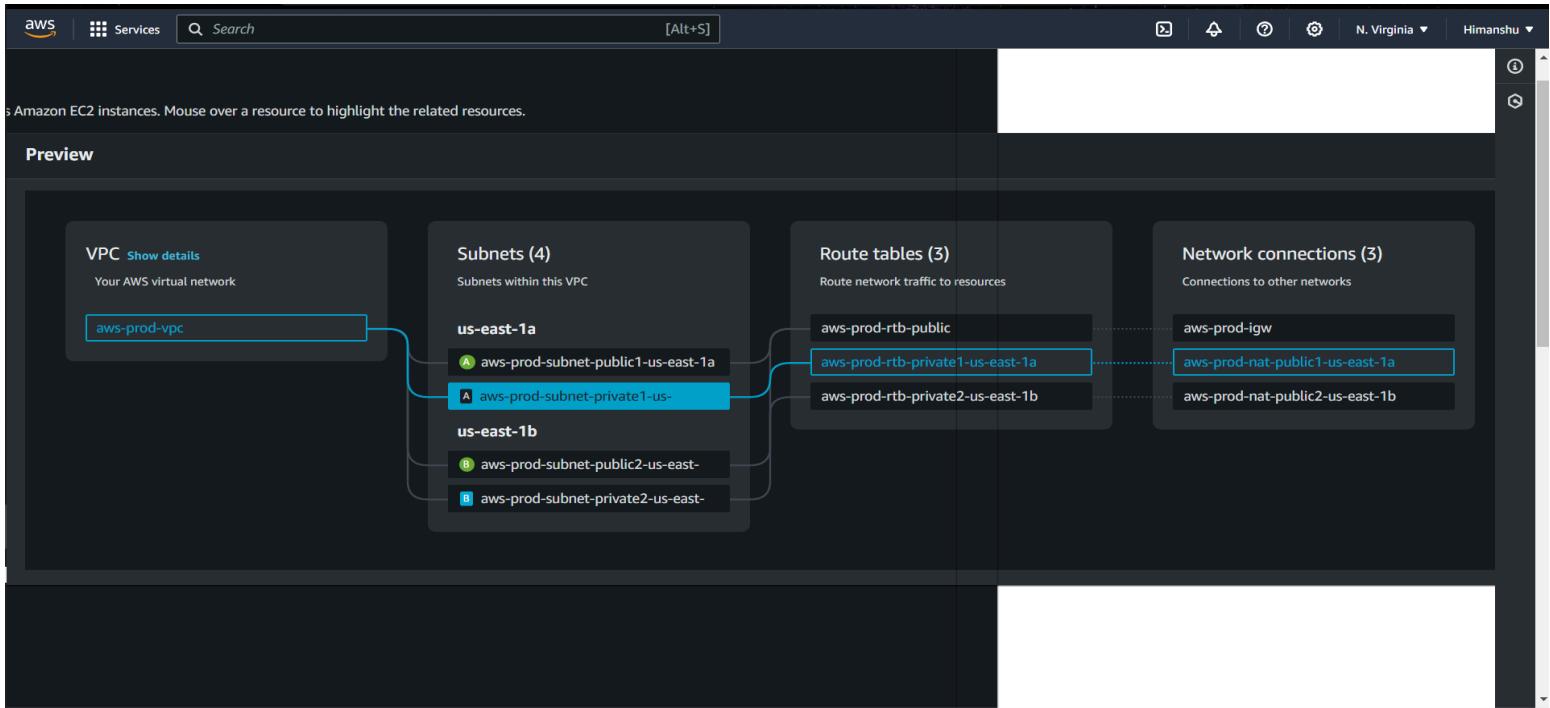
★ A Heartfelt Thank You to Abhishek Veeramalla! ★

I completed this VPC project. The insights and detailed explanations provided in [this video](#) were incredibly helpful.

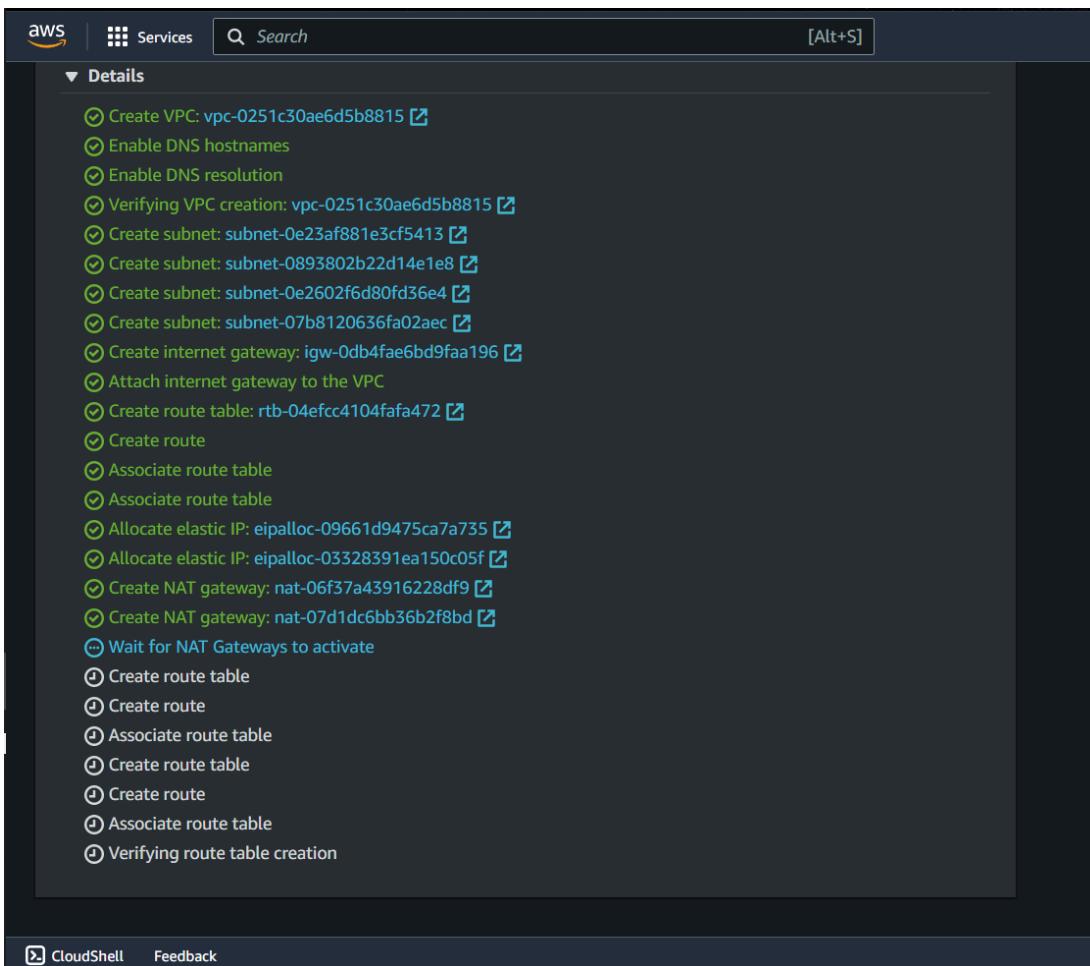
Through this video, I learned so many new things about VPCs and cloud networking. Abhishek, your ability to break down complex concepts and make them accessible is truly remarkable.

Thank you for sharing your knowledge and for making such valuable content available to the community. Your work is making a significant impact, and I'm grateful for the learning opportunity you provided.

So in this project we are going to create a VPC(Virtual Private Cloud) first.



We have to create in 2 AZ with each having one public and private subnet .



So , we can see in the above screenshot what all things VPC creates for us.

vpc-0251c30ae6d5b8815 / aws-prod-vpc Actions ▾

**Details** Info

VPC ID vpc-0251c30ae6d5b8815	State <span style="color: green;">Available</span>	DNS hostnames Enabled	DNS resolution Enabled
Tenancy Default	DHCP option set dopt-0e75d1eb5d8be3772	Main route table rtb-07aebd2cf977743e6	Main network ACL acl-0d16abad00e7011a4
Default VPC No	IPv4 CIDR 10.0.0.0/16	IPv6 pool -	IPv6 CIDR (Network border group) -
Network Address Usage metrics Disabled	Route 53 Resolver DNS Firewall rule groups -	Owner ID 730335632289	

**Resource map** Info

**Subnets (4)**  
Subnets within this VPC

**us-east-1a**

- aws-prod-subnet-public1-us-east-1a
- aws-prod-subnet-private1-us-east-1a

**Route tables (4)**  
Route network traffic to resources

- rtb-07aebd2cf977743e6
- aws-prod-rtb-private2-us-east-1b
- aws-prod-rtb-private1-us-east-1a

**Network connections (3)**  
Connections to other networks

- aws-prod-igw
- aws-prod-nat-public1-us-east-1a
- aws-prod-nat-public2-us-east-1b

- Now we need to create the Auto Scaling Group and for creating ASG we need to create the launch template .
- A launch template in an Auto Scaling Group (ASG) in AWS specifies information for configuring Amazon EC2 instances. It includes details like the Amazon Machine Image (AMI) ID, instance type, key pair, and security groups. Launch templates are similar to launch configurations.

The screenshot shows the AWS Auto Scaling 'Create Auto Scaling group' wizard. The left sidebar lists steps from 1 to 7. Step 1 is 'Choose launch template', which is currently active. Step 2 is 'Choose instance launch options'. Step 3 is 'Configure advanced options' (optional). Step 4 is 'Configure group size and scaling' (optional). Step 5 is 'Add notifications' (optional). Step 6 is 'Add tags' (optional). Step 7 is 'Review'. The main content area is titled 'Choose launch template' with a 'Info' link. It instructs the user to specify a launch template for common settings. A 'Name' input field is present, with 'Auto Scaling group name' and a placeholder 'Enter a name to identify the group.' Below it is a note: 'Must be unique to this account in the current Region and no more than 255 characters.' The 'Launch template' section follows, with a note about the restriction for accounts created after May 31, 2023. It includes a dropdown for selecting a launch template and a link to create one.

AWS Services Search [Alt+S] N. Virginia Himanshu

EC2 > Auto Scaling groups > Create Auto Scaling group

Step 1 Choose launch template

Step 2 Choose instance launch options

Step 3 - optional Configure advanced options

Step 4 - optional Configure group size and scaling

Step 5 - optional Add notifications

Step 6 - optional Add tags

Step 7 Review

## Choose launch template Info

Specify a launch template that contains settings common to all EC2 instances that are launched by this Auto Scaling group.

**Name**

Auto Scaling group name  
Enter a name to identify the group.

Must be unique to this account in the current Region and no more than 255 characters.

**Launch template Info**

ⓘ For accounts created after May 31, 2023, the EC2 console only supports creating Auto Scaling groups with launch templates. Creating Auto Scaling groups with launch configurations is not recommended but still available via the CLI and API until December 31, 2023.

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.

Select a launch template ▾ C

Create a launch template ↗

## Creating a launch template :

The screenshot shows the AWS EC2 'Create launch template' interface. At the top, there's a navigation bar with the AWS logo, 'Services' (selected), a search bar, and a keyboard shortcut '[Alt+S]'. The main title is 'Create launch template' under the 'EC2 > Launch templates' path. On the left, a sidebar lists 'Launch template name and description', 'Template version description', 'Auto Scaling guidance' (with an info link), and sections for 'Template tags' and 'Source template'. The main content area starts with a 'Software Image (AMI)' section, followed by 'Virtual server type (instance type)', 'Firewall (security group)', and 'Storage (volumes)'. A callout box highlights 'Auto Scaling guidance' with the note: 'Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 750 hours of public IPv4 address usage per month, 30 GiB of EBS storage, 2 million I/Os, 1 GB of snapshots, and 100 GB of bandwidth to the internet.' At the bottom right are 'Cancel' and 'Create launch template' buttons.

After creating a launch template you can select it in ASG.

S Services Search [Alt+S] N. Virginia ▾ Himanshu ▾

EC2 > Auto Scaling groups > Create Auto Scaling group

Step 1 Choose launch template

Step 2 Choose instance launch options

Step 3 - optional Configure advanced options

Step 4 - optional Configure group size and scaling

Step 5 - optional Add notifications

Step 6 - optional Add tags

Step 7 Review

### Choose launch template Info

Specify a launch template that contains settings common to all EC2 instances that are launched by this Auto Scaling group.

**Name**

Auto Scaling group name  
Enter a name to identify the group.  
**aws-prod-project**

Must be unique to this account in the current Region and no more than 255 characters.

### Launch template Info

For accounts created after May 31, 2023, the EC2 console only supports creating Auto Scaling groups with launch templates. Creating Auto Scaling groups with launch configurations is not recommended but still available via the CLI and API until December 31, 2023.

**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.

**aws-prod-project** C

[Create a launch template](#)

**Version**

**Default (1)** C

[Create a launch template version](#)

Description	Launch template	Instance type
my first vpc project	<a href="#">aws-prod-project</a> lt-0a749876eacf1ea1	t2.micro
AMI ID	Security groups	Request Spot Instances
ami-04a81a99f5ec58529	-	No
Key pair name	Security group IDs	
aws-login	<a href="#">sg-07fe7c85dc2007730</a>	

**Additional details**

Storage (volumes)	Date created
-	Mon Aug 05 2024 09:17:55 GMT+0530 (India Standard Time)

Cancel Next

Here you can see , we selected the launch Template in ASG. After that you need to select in which VPC and AZ you want to launch the instance.

So, we can see I selected the VPC and the AZ with a subnet .

The screenshot shows the AWS EC2 Auto Scaling group creation wizard at Step 1: Choose instance launch options. The left sidebar lists steps from 1 to 7. The main area shows the 'Instance type requirements' section, which includes a launch template named 'aws-prod-project' (version Default, ID lt-0a749876eacf1ea1), an instance type 't2.micro', and an 'Override launch template' button. Below this is the 'Network' section, which allows selecting a VPC and defining availability zones and subnets. A dropdown menu shows 'vpc-0251c30ae6d5b8815 (aws-prod-vpc) 10.0.0.0/16'. Under 'Availability Zones and subnets', two subnets are listed: 'us-east-1a | subnet-0e2602f6d80fd36e4 (aws-prod-subnet-private1-us-east-1a) 10.0.128.0/20' and 'us-east-1b | subnet-07b8120636fa02aec (aws-prod-subnet-private2-us-east-1b) 10.0.144.0/20'. A 'Create a subnet' button is also present. At the bottom are 'Cancel', 'Skip to review', 'Previous', and 'Next' buttons.

Step 1  
Choose launch template

Step 2  
Choose instance launch options

Step 3 - optional  
Configure advanced options

Step 4 - optional  
Configure group size and scaling

Step 5 - optional  
Add notifications

Step 6 - optional  
Add tags

Step 7  
Review

**Choose instance launch options** Info

Choose the VPC network environment that your instances are launched into, and customize the instance types and purchase options.

**Instance type requirements** Info

You can keep the same instance attributes or instance type from your launch template, or you can choose to override the launch template by specifying different instance attributes or manually adding instance types.

Launch template	Version	Description
aws-prod-project <small>aws-prod-project</small>	Default	my first vpc project

Instance type  
t2.micro

**Network** Info

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.

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

vpc-0251c30ae6d5b8815 (aws-prod-vpc)  
10.0.0.0/16

Create a VPC aws-create-vpc

**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-1a | subnet-0e2602f6d80fd36e4 (aws-prod-subnet-private1-us-east-1a)  
10.0.128.0/20

us-east-1b | subnet-07b8120636fa02aec (aws-prod-subnet-private2-us-east-1b)  
10.0.144.0/20

Create a subnet aws-create-subnet

Cancel Skip to review Previous Next

# We are not configuring the Load balancer here .

The screenshot shows the 'Create Auto Scaling group' wizard in the AWS Management Console. The current step is 'Configure advanced options - optional'. The sidebar on the left lists optional steps: Step 1 (Choose launch template), Step 2 (Choose instance launch options), Step 3 (Configure advanced options), Step 4 (Configure group size and scaling), Step 5 (Add notifications), Step 6 (Add tags), and Step 7 (Review). The main content area is divided into sections: 'Load balancing', 'VPC Lattice integration options', 'Health checks', and 'Additional settings'. In the 'Load balancing' section, the 'No load balancer' option is selected. In the 'VPC Lattice integration options' section, the 'No VPC Lattice service' option is selected. In the 'Health checks' section, 'EC2 health checks' are enabled ('Always enabled'). In the 'Additional settings' section, 'Monitoring' and 'Default instance warmup' are configured. At the bottom, there are 'Cancel', 'Skip to review', 'Previous', and 'Next' buttons.

Step 1  
Choose launch template

Step 2  
Choose instance launch options

Step 3 - optional  
Configure advanced options

Step 4 - optional  
Configure group size and scaling

Step 5 - optional  
Add notifications

Step 6 - optional  
Add tags

Step 7  
Review

### Configure advanced options - optional Info

Integrate your Auto Scaling group with other services to distribute network traffic across multiple servers using a load balancer or to establish service-to-service communications using VPC Lattice. You can also set options that give you more control over health check replacements and monitoring.

#### Load balancing Info

Use the options below to attach your Auto Scaling group to an existing load balancer, or to a new load balancer that you define.

No load balancer  
Traffic to your Auto Scaling group will not be fronted by a load balancer.

Attach to an existing load balancer  
Choose from your existing load balancers.

Attach to a new load balancer  
Quickly create a basic load balancer to attach to your Auto Scaling group.

#### VPC Lattice integration options Info

To improve networking capabilities and scalability, integrate your Auto Scaling group with VPC Lattice. VPC Lattice facilitates communications between AWS services and helps you connect and manage your applications across compute services in AWS.

No VPC Lattice service  
VPC Lattice will not manage your Auto Scaling group's network access and connectivity with other services.

Attach to VPC Lattice service  
Incoming requests associated with specified VPC Lattice target groups will be routed to your Auto Scaling group.

#### Create new VPC Lattice service [+]

#### Health checks

Health checks increase availability by replacing unhealthy instances. When you use multiple health checks, all are evaluated, and if at least one fails, instance replacement occurs.

##### EC2 health checks

ⓘ Always enabled

##### Additional health check types - optional Info

Turn on Elastic Load Balancing health checks  
Elastic Load Balancing monitors whether instances are available to handle requests. When it reports an unhealthy instance, EC2 Auto Scaling can replace it on its next periodic check.

Turn on VPC Lattice health checks  
VPC Lattice can monitor whether instances are available to handle requests. If it considers a target as failed a health check, EC2 Auto Scaling replaces it after its next periodic check.

##### Health check grace period Info

This time period delays the first health check until your instances finish initializing. It doesn't prevent an instance from terminating when placed into a non-running state.

300 seconds

#### Additional settings

##### Monitoring Info

Enable group metrics collection within CloudWatch

##### Default instance warmup Info

The amount of time that CloudWatch metrics for new instances do not contribute to the group's aggregated instance metrics, as their usage data is not reliable yet.

Enable default instance warmup

Cancel Skip to review Previous Next [+]

# Here we select the size and scaling of ASG.

The screenshot shows the AWS Auto Scaling Groups 'Create Auto Scaling group' wizard at Step 4: **Configure group size and scaling - optional**. The page is titled 'Configure group size and scaling - optional' with an 'Info' link. It explains that you can define your group's desired capacity and scaling limits. You can optionally add automatic scaling to adjust the size of your group.

**Group size** Info  
Set the initial size of the Auto Scaling group. After creating the group, you can change its size to meet demand, either manually or by using automatic scaling.

**Desired capacity type**  
Choose the unit of measurement for the desired capacity value. vCPUs and Memory(GiB) are only supported for mixed instances groups configured with a set of instance attributes.

**Units (number of instances)** ▾

**Desired capacity**  
Specify your group size.  
2

**Scaling** Info  
You can resize your Auto Scaling group manually or automatically to meet changes in demand.

**Scaling limits**  
Set limits on how much your desired capacity can be increased or decreased.

**Min desired capacity** 1  
Equal or less than desired capacity

**Max desired capacity** 1  
Equal or greater than desired capacity

**Automatic scaling - optional**  
Choose whether to use a target tracking policy Info  
You can set up other metric-based scaling policies and scheduled scaling after creating your Auto Scaling group.

**No scaling policies**  
Your Auto Scaling group will remain at its initial size and will not dynamically resize to meet demand.

**Target tracking scaling policy**  
Choose a CloudWatch metric and target value and let the scaling policy adjust the desired capacity in proportion to the metric's value.

**Instance maintenance policy** Info  
Control your Auto Scaling group's availability during instance replacement events. This includes health checks, instance refreshes, maximum instance lifetime features and events that happen automatically to keep your group balanced, called rebalancing events.

Choose a replacement behavior depending on your availability requirements

**Mixed behavior**  
 **No policy**  
For rebalancing events, new instances will launch before terminating others. For all other events, instances terminate and launch at the same time.

**Prioritize availability**  
 **Launch before terminating**  
Launch new instances and wait for them to be ready before terminating others. This allows you to go above your desired capacity by a given percentage and may temporarily increase costs.

**Control costs**  
 **Terminate and launch**  
Terminate and launch instances at the same time. This allows you to go below your desired capacity by a given percentage and may temporarily reduce availability.

**Flexible**  
 **Custom behavior**  
Set custom values for the minimum and maximum amount of available capacity. This gives you greater flexibility in setting how far below and over your desired capacity EC2 Auto Scaling goes when replacing instances.

**Instance scale-in protection**  
Scale-in protection prevents newly launched instances from being terminated by scaling activities. Make sure to remove scale-in protection for the group or individual instances when instances are ready to be terminated.

Enable instance scale-in protection

**Cancel** **Skip to review** **Previous** **Next**

The screenshot shows the AWS EC2 Auto Scaling group creation wizard at Step 5: Add notifications. The main title is "Add notifications - optional". A sub-instruction says "Send notifications to SNS topics whenever Amazon EC2 Auto Scaling launches or terminates the EC2 instances in your Auto Scaling group." Below this is a large "Add notification" button. To the right are "Cancel", "Skip to review", "Previous", and "Next" buttons. On the left, there's a vertical sidebar with steps: Step 1 (Choose launch template), Step 2 (Choose instance launch options), Step 3 (optional: Configure advanced options), Step 4 (optional: Configure group size and scaling), Step 5 (optional: Add notifications), Step 6 (optional: Add tags), and Step 7 (Review).

Auto Scaling Group created two instances in the two different AZ us-east-1b and us-east-1a .

The screenshot shows the AWS EC2 Instances page. The sidebar on the left includes "EC2 Dashboard", "EC2 Global View", "Events", "Console-to-Code Preview", "Instances" (selected), "Instances Types", "Launch Templates", "Spot Requests", "Savings Plans", "Reserved Instances", "Dedicated Hosts", "Capacity Reservations", "Images" (AMIs, AMI Catalog), and "Elastic Block Store" (Volumes, Snapshots). The main area displays "Instances (2)" with a table. The table columns are: Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public IP. Two instances are listed:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
	i-0ac2fa2d09d6cb09d	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1b	-
	i-0f22575cf44095746	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	-

A modal window titled "Select an instance" is open at the bottom.

We can review all the settings which we did to create ASG.

AWS Services Search [Alt+S] N. Virginia Himanshu

EC2 > Auto Scaling groups > Create Auto Scaling group

Step 1 Choose launch template

Step 2 Choose instance launch options

Step 3 - optional Configure advanced options

Step 4 - optional Configure group size and scaling

Step 5 - optional Add notifications

Step 6 - optional Add tags

Step 7 Review

### Review Info

**Step 1: Choose launch template** Edit

**Group details**

Auto Scaling group name: aws-prod-project

**Launch template**

Launch template	Version	Description
aws-prod-project [2] lt-0a749876eacf1ea1	Default	my first vpc project

**Step 2: Choose instance launch options** Edit

**Network**

VPC: vpc-0251c30ae6d5b8815 [2]

Availability Zone	Subnet	
us-east-1a	subnet-0e2602f6d80fd36e4 [2]	10.0.128.0/20
us-east-1b	subnet-07b8120636fa02aec [2]	10.0.144.0/20

**Instance type requirements**

This Auto Scaling group will adhere to the launch template.

**Step 3: Configure advanced options** Edit

**Load balancing**

Load balancer: -

**VPC Lattice integration options**

VPC Lattice target groups: -

**Health checks**

Health check type	Health check grace period
EC2	300 seconds

**Additional settings**

Monitoring	Default instance warmup
Disabled	Disabled

**Step 4: Configure group size and scaling policies** Edit

**Group size**

Desired capacity	Desired capacity type
2	Units (number of instances)

**Scaling**

Minimum desired capacity	Maximum desired capacity
1	1

Target tracking policy: -

**Instance maintenance policy**

Replacement behavior	Min healthy percentage	Max healthy percentage
No policy	-	-

**Instance scale-in protection**

Instance scale-in protection:  Enable instance protection from scale in

**Step 5: Add notifications** Edit

**Notifications**

No notifications

**Step 6: Add tags** Edit

**Tags (0)**

Key	Value	Tag new instances
No tags		

**Create Auto Scaling group**

# Here you can see the AGS is created.

The screenshot shows the AWS EC2 Auto Scaling groups page. At the top, a green banner displays the message "aws-prod-project created successfully". Below the banner, the page title is "Auto Scaling groups (1) Info". A search bar and a navigation breadcrumb "EC2 > Auto Scaling groups" are present. The main table lists one Auto Scaling group:

Name	Launch template/configuration	Instances	Status	Desired capacity	Min	Max	Available
aws-prod-project	aws-prod-project   Version Default	0	Updating capacity...	1	1	2	us-east-...

The screenshot shows the detailed configuration page for the "aws-prod-project" Auto Scaling group. The top navigation includes "EC2 > Auto Scaling groups > aws-prod-project" and tabs for "Details", "Activity", "Automatic scaling", "Instance management", "Monitoring", and "Instance refresh".

**Group details**

Auto Scaling group name aws-prod-project	Desired capacity 2	Desired capacity type Units (number of instances)	Amazon Resource Name (ARN) arn:aws:autoscaling:us-east-1:730335632289:autoScalingGroup:7d1ac7c9-246e-4b69-9789-46beb2a23582:autoScalingGroupName/aws-prod-project
Date created Mon Aug 05 2024 09:27:06 GMT+0530 (India Standard Time)	Minimum capacity 1	Status Updating capacity	
	Maximum capacity 4		

**Launch template**

Launch template <a href="#">lt-0a749876eacf1ea1</a> aws-prod-project	AMI ID <a href="#">ami-04a81a99f5ec58529</a>	Instance type t2.micro	Owner arn:aws:iam::730335632289:root
Version Default	Security groups -	Security group IDs <a href="#">sg-07fe7c85dc2007730</a>	Create time Mon Aug 05 2024 09:17:55 GMT+0530 (India Standard Time)
Description my first vpc project	Storage (volumes) -	Key pair name aws-login	Request Spot Instances No

**Network**

Availability Zones us-east-1a, us-east-1b	Subnet ID subnet-0e2602ff680fd36e4, subnet-07b8120636fa02aec
--	---

**Instance type requirements**

Your Auto Scaling group adheres to the launch template for purchase option and instance type.

**Load balancing**

Load balancer target groups -	Classic Load Balancers -
----------------------------------	-----------------------------

**VPC Lattice integration options**

VPC Lattice target groups -
--------------------------------

**Health checks**

Health check type EC2	Health check grace period 300
--------------------------	----------------------------------

**Instance maintenance policy**

Replacement behavior No policy	Min healthy percentage -	Max healthy percentage -
-----------------------------------	-----------------------------	-----------------------------

**Advanced configurations**

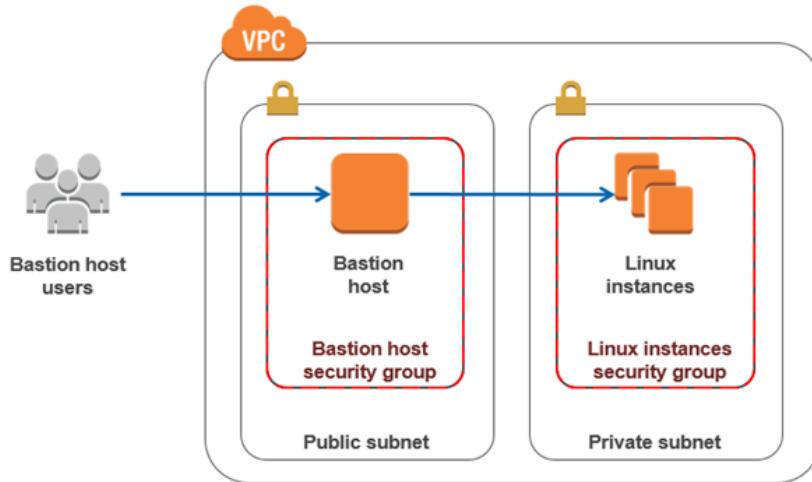
Instance scale-in protection Not protected from scale in	Termination policies Default	Maximum instance lifetime -	Service-linked role arn:aws:iam::730335632289:role/arn:aws-service-role/autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling
Placement group -	Suspended processes -	Default cooldown 300	Default instance warmup Disabled

**Tags (0)**

Key	Value	Tag new instances
-----	-------	-------------------

- Why do we need to create a Bastion Host ?

→ An AWS bastion host is a secure computer that acts as a gateway to access other resources within a virtual private cloud (VPC). It's designed to be the SSH entrypoint for a VPC, and it can provide secure access to Linux instances in both private and public subnets.



- Here we are creating a bastion host so that we can connect to our private Ec2 instance, we have to do network settings so that this bastion host should be inside our VPC publicly available .

**Network settings** Info

VPC - required Info  
vpc-0251c30ae6d5b8815 (aws-prod-vpc)  
10.0.0.0/16

Subnet Info  
subnet-0e23af881e3cf5413 aws-prod-subnet-public1-us-east-1a  
VPC: vpc-0251c30ae6d5b8815 Owner: 730335632289  
Availability Zone: us-east-1a IP addresses available: 4090 CIDR: 10.0.0.0/20

Auto-assign public IP Info  
Enable

Additional charges apply when outside of free tier allowance

Firewall (security groups) Info  
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

Create security group  Select existing security group

Security group name - required  
launch-wizard-1

This security group will be added to all network interfaces. The name can't be edited after the security group is created. Max length is 255 characters. Valid characters: a-z, A-Z, 0-9, spaces, and \_-!@#\$%^&+=`~,;!

Description - required Info  
launch-wizard-1 created 2024-08-05T04:06:59.562Z

**Summary**

Number of instances Info  
1

Software Image (AMI)  
Canonical, Ubuntu, 24.04 LTS, ...read more  
ami-04a81a99f5ec58529

Virtual server type (instance type)  
t2.micro

Firewall (security group)  
New security group

Storage (volumes)  
1 volume(s) - 8 GiB

ⓘ Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 750 hours of public IPv4 address usage per month, 20 GiB

Cancel
**Launch instance**
Review commands

- For accessing bastion host i copied my .pem to my WSL

```
himanshu@DESKTOP-LVJ61KE: ~  
—(himanshu@ DESKTOP-LVJ61KE)-[~]  
$ cp /mnt/d/AWS_RSA_KEY/aws-login.pem ~/  
—(himanshu@ DESKTOP-LVJ61KE)-[~]  
$ ls  
aws aws-login.pem awsStatus.sh awsStatus.txt first-shell-script.sh linux.txt  
—(himanshu@ DESKTOP-LVJ61KE)-[~]  
$
```

I got an error here when i am copy .pem file using scp to bastion host,

- Actually I didn't give the correct permission to the .pem file so I gave it .

```
(himanshu@DESKTOP-LVJ61KE) -[~]
$ scp -i aws-login.pem aws-login.pem ubuntu@44.195.77.144:/home/ubuntu
The authenticity of host '44.195.77.144 (44.195.77.144)' can't be established.
ED25519 key fingerprint is SHA256:hlZp8KLKDCi06JNCG/VRn/CVo0cY00ZxNLcw9maz9xo.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added '44.195.77.144' (ED25519) to the list of known hosts.
@@@@@@@@@@@WARNING: UNPROTECTED PRIVATE KEY FILE! @
@@@@@@@@@@@Permissions 0755 for 'aws-login.pem' are too open.
It is required that your private key files are NOT accessible by others.
This private key will be ignored.
Load key "aws-login.pem": bad permissions
ubuntu@44.195.77.144: Permission denied (publickey).
scp: Connection closed
```

- If you get this error first run chmod 400 "aws-login.pem" this for the permission.
  - After that I transferred .pem file to my bastion host so that i can connect to the private instance with that .pem file.

- Login into the the bastion host machine

```
[himanshu@DESKTOP-LVJ61KE] ~]$ ssh -i aws-login.pem ubuntu@44.195.77.144
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1009-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

System information as of Mon Aug  5 04:36:33 UTC 2024

System load: 0.0          Processes: 105
Usage of /: 22.9% of 6.71GB Users logged in: 0
Memory usage: 19%          IPv4 address for enX0: 10.0.1.60
Swap usage: 0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Mon Aug  5 04:34:44 2024 from 14.143.179.90
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-10-0-1-60:~$
```

- After login to bastion host did ssh to ec2 instance in private subnet

```
ubuntu@ip-10-0-1-60:~$ ssh -i aws-login.pem ubuntu@10.0.153.11
The authenticity of host '10.0.153.11 (10.0.153.11)' can't be established.
ED25519 key fingerprint is SHA256:33dnzrxLo9qCwQD+Q6cKnpyuHwoKnEIhqIgsZs/5GRY.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.153.11' (ED25519) to the list of known hosts.
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1009-aws x86_64)
```

```
* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/pro
```

System information as of Mon Aug 5 04:38:27 UTC 2024

```
System load: 0.0          Processes:      105
Usage of /: 22.8% of 6.71GB  Users logged in: 0
Memory usage: 30%          IPv4 address for enx0: 10.0.153.11
Swap usage: 0%
```

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.  
See <https://ubuntu.com/esm> or run: sudo pro status

The list of available updates is more than a week old.  
To check for new updates run: sudo apt update

The programs included with the Ubuntu system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/\*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by  
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".  
See "man sudo\_root" for details.

```
ubuntu@ip-10-0-153-11:~$
```

- After this i made a index.html file and run the python server

```
ubuntu@ip-10-0-153-11:~$ python3 -m http.server 8000
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

We didn't created the load balancer After this we need to create load balancer

The screenshot shows the 'Compare and select load balancer type' section of the AWS EC2 Load Balancers page. It compares three types:

- Application Load Balancer**: Handles HTTP and HTTPS traffic at the request level, supporting Lambda functions.
- Network Load Balancer**: Handles TCP, UDP, and TLS traffic, offloading TLS at scale and supporting VPC.
- Gateway Load Balancer**: Manages a fleet of third-party virtual appliances, supporting GEN1/EV.

A note at the bottom indicates that ALB is chosen for its flexible feature set for applications with HTTP and HTTPS traffic.

And will go with ALB( Application load Balancer ), basically it is OSI Layer 7 load balancer which mostly works on HTTP and HTTPS.

The screenshot shows the 'Listeners and routing' configuration for an ALB. A listener is defined for port 80, configured to forward requests to a target group. The 'Default action' is set to 'Forward to' a target group.

Protocol	Port	Action
HTTP	80	Forward to Select a target group

Below the table, there are sections for 'Listener tags - optional' and 'Add listener'.

**Listener tags - optional**  
Consider adding tags to your listener. Tags enable you to categorize your AWS resources so you can more easily manage them.

**Add listener tag**  
You can add up to 50 more tags.

**Add listener**

- What is Target Group ?

→ Target groups route requests to individual registered targets, such as EC2 instances, using the protocol and port number that you specify. You can register a target with multiple target groups. You can configure health checks on a per target group basis. Health checks are performed on all targets registered to a target group that is specified in a listener rule for your load balancer.

The screenshot shows the 'Create target group' wizard in the AWS Management Console. The current step is 'Specify group details'. The page title is 'Specify group details' and the sub-section title is 'Basic configuration'. The 'Instances' option is selected under 'Choose a target type'. The 'Target group name' is set to 'aws-prod-project'. The 'Protocol : Port' section shows 'HTTP' selected with port '80'. The 'IP address type' section shows 'IPv4' selected. The 'VPC' section lists 'aws-prod-vpc' with its ID and CIDR range. The 'Protocol version' section shows 'HTTP1' selected. The 'Health checks' section includes fields for 'Health check protocol' (set to 'HTTP') and 'Health check path' (set to '/'). The 'Attributes' section contains a note about default attributes being applied. The 'Tags - optional' section encourages adding tags for categorization. At the bottom, there are 'Cancel' and 'Next' buttons.

## Review targets

### Targets (2)

[Remove all pending](#)

Filter targets

Show only pending

< 1 > |

Instance ID	Name	Port	State	Security groups	Zone	Private IPv4 address	Subnet ID
i-0f22575cf44095746		8000	<span>Running</span>	aws-prod-project	us-east-1a	10.0.142.192	subnet-0e260
i-0ac2fa2d09d6cb09d		8000	<span>Running</span>	aws-prod-project	us-east-1b	10.0.153.11	subnet-07b81

2 pending

[Cancel](#)

[Previous](#)

[Create target group](#)

⌚ Successfully created load balancer: **aws-prod-project-load-Balancer**

It might take a few minutes for your load balancer to fully set up and route traffic. Targets will also take a few minutes to complete the registration process and pass initial health checks.

[EC2](#) > [Load balancers](#) > [aws-prod-project-load-Balancer](#)

## aws-prod-project-load-Balancer

### ▼ Details

Load balancer type	Status	VPC	Load balancer IP address type
Application	Provisioning	<a href="#">vpc-0251c30ae6d5b8815</a>	IPv4
Scheme	Hosted zone	Availability Zones	Date created
Internet-facing	Z35SXDOTRQ7X7K	<a href="#">subnet-0893802b22d14e1e8</a> us-east-1b (use1-az2) <a href="#">subnet-0e23af881e3cf5413</a> us-east-1a (use1-az1)	August 5, 2024, 10:34 (UTC+05:30)

### Load balancer ARN

arn:aws:elasticloadbalancing:us-east-1:730335632289:loadbalancer/app/aws-project-load-Balancer/4cbe13b6d2b7ec0

### DNS name [Info](#)

aws-prod-project-load-Balancer-1034337732.us-east-1.elb.amazonaws.com (A Record)

[Listeners and rules](#)

[Network mapping](#)

[Resource map - new](#)

[Security](#)

[Monitoring](#)

[Integrations](#)

[Attributes](#)

[Tags](#)

# Load balancer is created with target group

Screenshot of the AWS EC2 Load Balancers console showing the configuration of the 'aws-prod-project-load-Balancer'. The load balancer is of type Application, Internet-facing, with status Active. It is associated with a VPC (vpc-0251c30ae6d5b8815) and two Availability Zones (subnet-0893802b22d14e1e8 and subnet-0e23af881e3cf5413). The DNS name copied is aws-prod-project-load-Balancer-1034337732.us-east-1.elb.amazonaws.com (A Record).

- So load balance is ready now and I also opened port 80 in the security group and now we can see the application is running .

Screenshot of a web browser displaying the response from the load balancer. The page shows a cartoon chipmunk character with the text 'Hello' above it and 'HOL' below it. The browser status bar indicates the URL is 'aws-prod-project-load-balancer-1034337732.us-east-1.elb.amazonaws.com' and the content type is 'CTYPE html>'. A warning message 'Not secure' is visible.

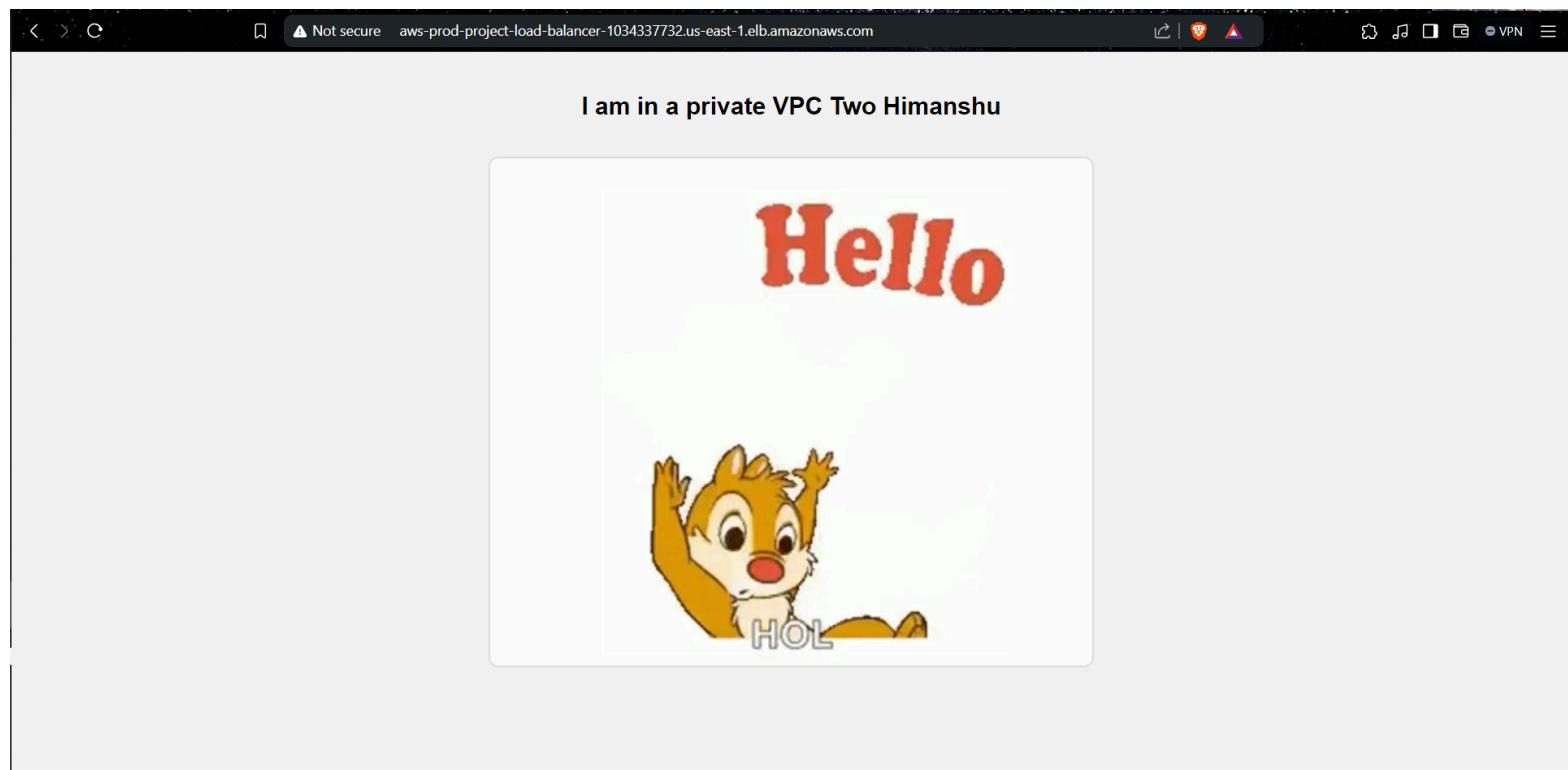
```
ubuntu@ip-10-0-153-11:~$ python3 -m http.server 8000
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
10.0.16.65 - - [05/Aug/2024 05:06:33] "GET / HTTP/1.1" 200 -
10.0.14.70 - - [05/Aug/2024 05:06:48] "GET / HTTP/1.1" 200 -
10.0.16.65 - - [05/Aug/2024 05:07:03] "GET / HTTP/1.1" 200 -
10.0.14.70 - - [05/Aug/2024 05:07:18] "GET / HTTP/1.1" 200 -
10.0.16.65 - - [05/Aug/2024 05:07:33] "GET / HTTP/1.1" 200 -
10.0.14.70 - - [05/Aug/2024 05:07:48] "GET / HTTP/1.1" 200 -
10.0.16.65 - - [05/Aug/2024 05:08:03] "GET / HTTP/1.1" 200 -
10.0.14.70 - - [05/Aug/2024 05:08:18] "GET / HTTP/1.1" 200 -
10.0.16.65 - - [05/Aug/2024 05:08:33] "GET / HTTP/1.1" 200 -
10.0.14.70 - - [05/Aug/2024 05:08:48] "GET / HTTP/1.1" 200 -
10.0.16.65 - - [05/Aug/2024 05:09:03] "GET / HTTP/1.1" 200 -
10.0.16.65 - - [05/Aug/2024 05:09:11] "GET / HTTP/1.1" 200 -
10.0.16.65 - - [05/Aug/2024 05:09:12] code 404, message File not found
10.0.16.65 - - [05/Aug/2024 05:09:12] "GET /favicon.ico HTTP/1.1" 404 -
10.0.14.70 - - [05/Aug/2024 05:09:18] "GET / HTTP/1.1" 200 -
10.0.16.65 - - [05/Aug/2024 05:09:33] "GET / HTTP/1.1" 200 -
10.0.14.70 - - [05/Aug/2024 05:09:48] "GET / HTTP/1.1" 200 -
10.0.16.65 - - [05/Aug/2024 05:10:03] "GET / HTTP/1.1" 200 -
10.0.14.70 - - [05/Aug/2024 05:10:18] "GET / HTTP/1.1" 200 -
10.0.16.65 - - [05/Aug/2024 05:10:33] "GET / HTTP/1.1" 200 -
10.0.14.70 - - [05/Aug/2024 05:10:48] "GET / HTTP/1.1" 200 -
10.0.16.65 - - [05/Aug/2024 05:11:03] "GET / HTTP/1.1" 200 -
10.0.14.70 - - [05/Aug/2024 05:11:18] "GET / HTTP/1.1" 200 -
10.0.16.65 - - [05/Aug/2024 05:11:33] "GET / HTTP/1.1" 200 -
10.0.14.70 - - [05/Aug/2024 05:11:48] "GET / HTTP/1.1" 200 -
10.0.16.65 - - [05/Aug/2024 05:12:03] "GET / HTTP/1.1" 200 -
10.0.14.70 - - [05/Aug/2024 05:12:18] "GET / HTTP/1.1" 200 -
10.0.16.65 - - [05/Aug/2024 05:12:33] "GET / HTTP/1.1" 200 -
```

- So now what i did i ran the server in both EC2 in private VPC

```
ubuntu@ip-10-0-142-192:~$ vim index.html
ubuntu@ip-10-0-142-192:~$ python3 -m http.server 8000
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
10.0.16.65 - - [05/Aug/2024 05:22:34] "GET / HTTP/1.1" 200 -
10.0.14.70 - - [05/Aug/2024 05:22:49] "GET / HTTP/1.1" 200 -
10.0.16.65 - - [05/Aug/2024 05:23:04] "GET / HTTP/1.1" 200 -
```



- We can see now load balance is transferring the traffic to both VPC one and VPC Two



This is step by step process to delete the every things which we created :

#### Clean-Up Steps:

##### 1. Delete the Auto Scaling Group:

- Remove the scaling configurations.

##### 2. Delete the Load Balancer:

- Remove the ALB you created.

##### 3. Delete the Target Group:

- Remove the associated target group.

##### 4. Delete the NAT Gateway:

- Remove the NAT gateway from your VPC.

##### 5. Delete All EC2 Instances:

- Terminate the three instances you created.

##### 6. Release the Elastic IP:

- Release any Elastic IP addresses associated with the instances.

##### 7. Delete the VPC:

- Remove the VPC and all associated resources