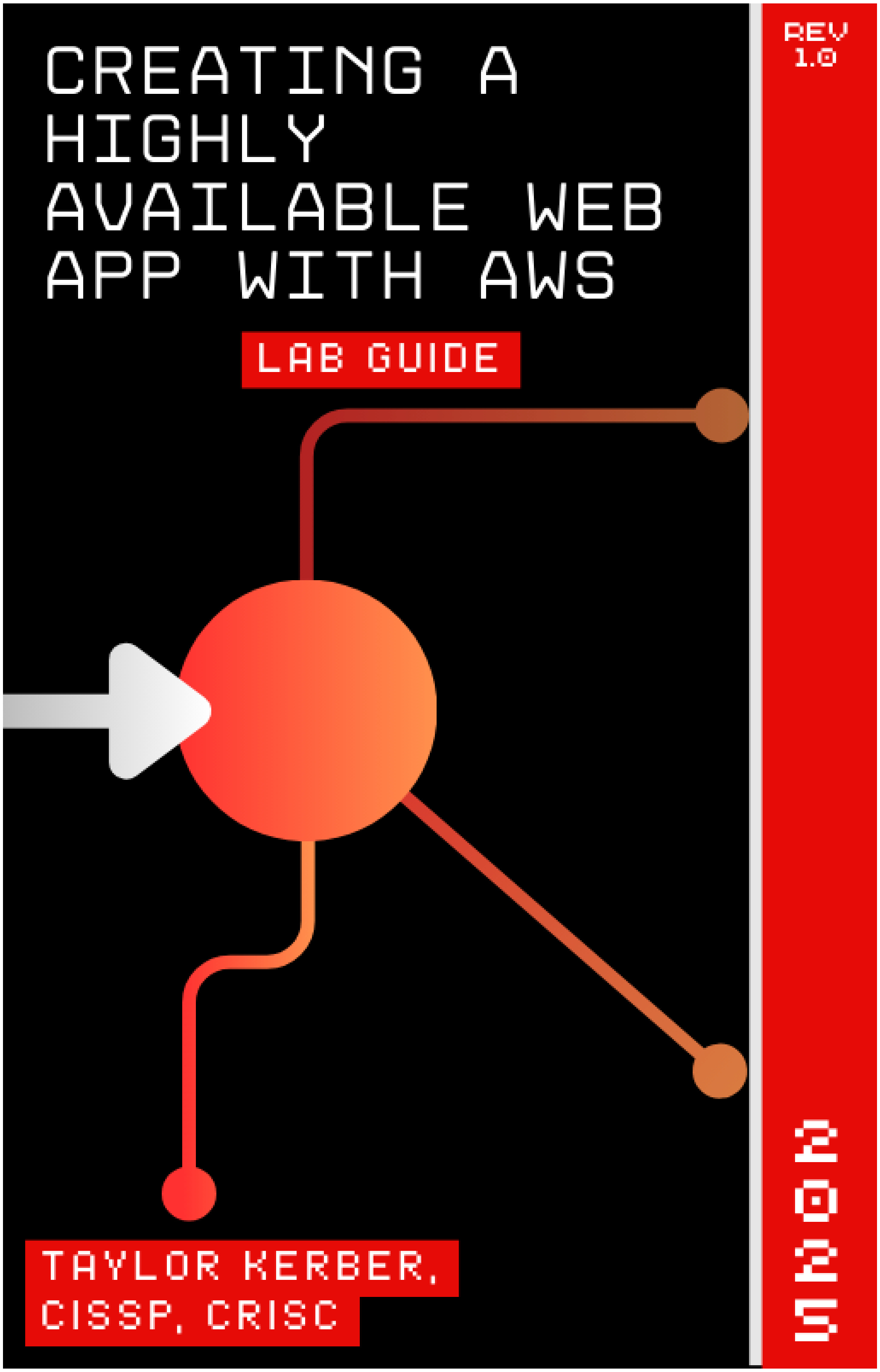

CREATING A HIGHLY AVAILABLE WEB APP WITH AWS
LAB GUIDE
REV 1.0
TAYLOR KERBER, CISSP, CRISC
2025

AWS Web App with Load Balancing Lab

Objective

This lab will demonstrate how to create a basic web application with two EC2 instances in different availability zones, behind a load balancer. Upon completion of this lab, you should have a better understanding of how to navigate the AWS console,

Pre-requisites

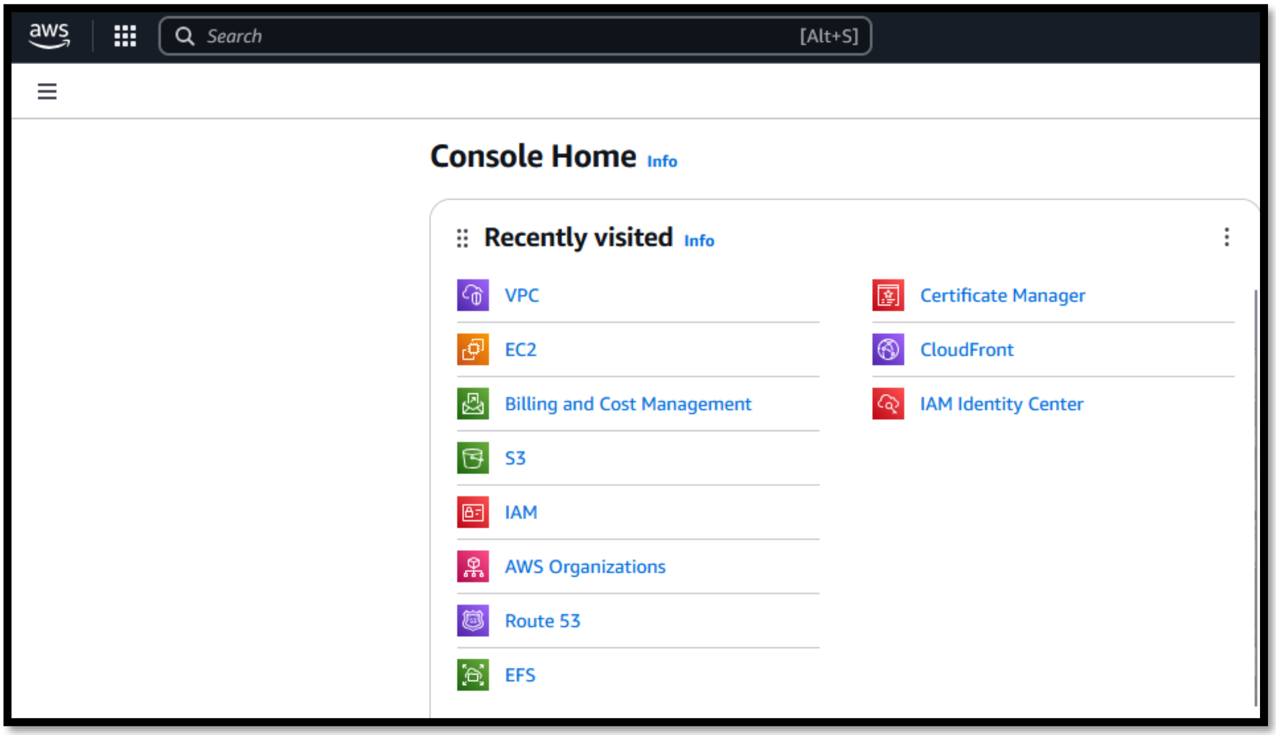
- A personal or lab AWS account (This lab does not walk-through how-to set up an AWS account.)
- Access to an internet / WAN connection to use the AWS console.
- A modern web browser like: Chrome, Firefox, Edge, etc.
- Windows: Install the application PuTTY and PuTTYgen.
- Linux: Able to use SSH over the web.

Table of Content

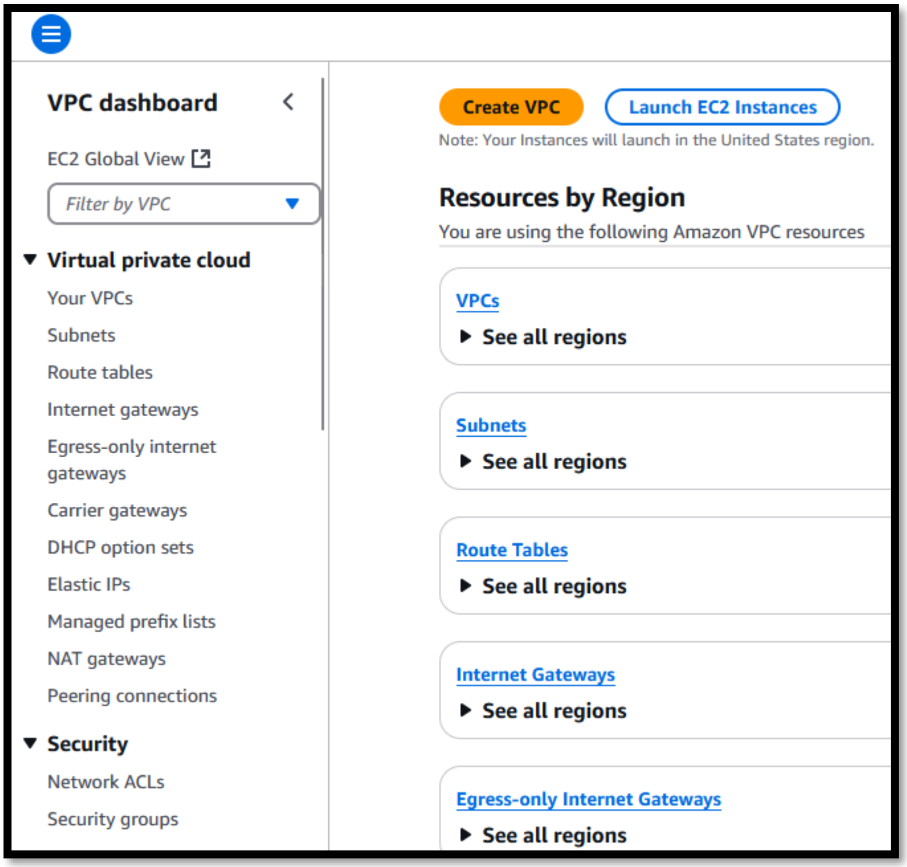
VPC Creation

In this section we will create our VPC, subnets, internet gateway, and routing.

1) Log into the AWS console and on the main page search for VPC or click on it (if you recently used it).

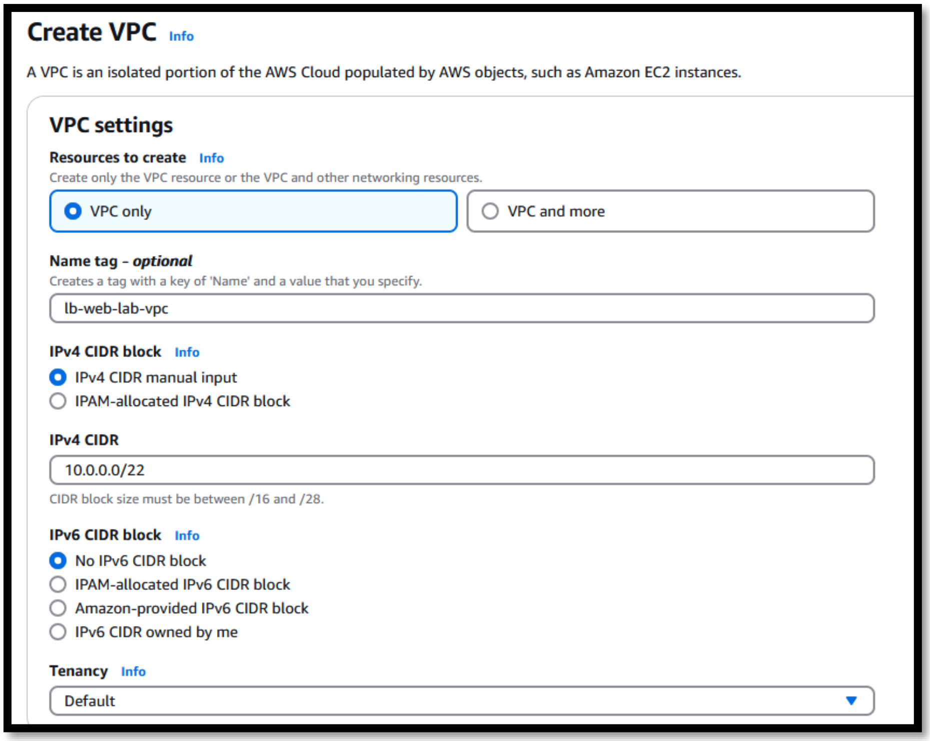


2) Once the VPC dashboard is open you will see a menu on the left-hand side. Click on Your VPCs and in the top right-hand side click the create VPC button.



3) Fill out a name and choose a CIDR block. In my example I’m choosing a /22.

4) When you’re done click the Create VPC button. **Create VPC**



Create VPC Info

A VPC is an isolated portion of the AWS Cloud populated by AWS objects, such as Amazon EC2 instances.

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.

lb-web-lab-vpc

IPv4 CIDR block Info

● IPv4 CIDR manual input

○ IPAM-allocated IPv4 CIDR block

IPv4 CIDR

10.0.0.0/22

CIDR block size must be between /16 and /28.

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

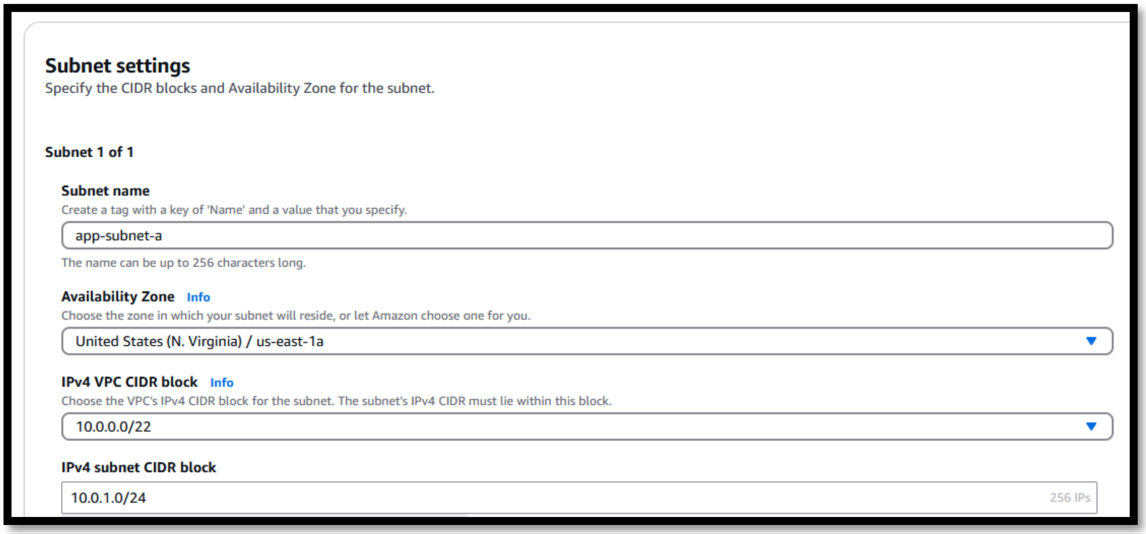
Default ▼

Subnets

1) On the VPC dashboard you will look on the left-hand side again and click on Subnets. The Subnets dashboard will display, and you will click the Create Subnet button.

2) I've chosen to carve out (2) /24 networks of my /22 VPC. This will give me subnets with 256 Ips each. Remember AWS reserves (5) IP addresses from each subnet CIDR for it's own use.

3) I also chose the 'A' availability zone and have named my subnet accordingly.



Subnet settings
Specify the CIDR blocks and Availability Zone for the subnet.

Subnet 1 of 1

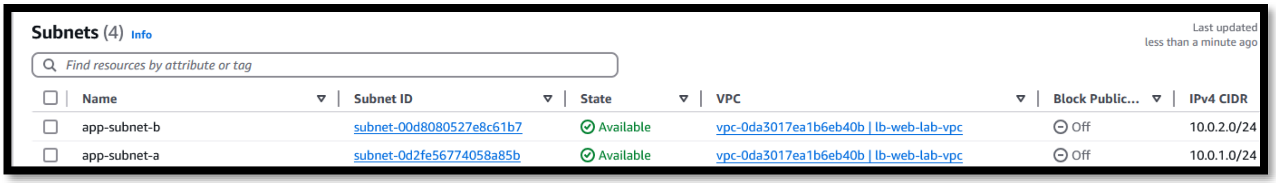
Subnet name
Create a tag with a key of 'Name' and a value that you specify.
app-subnet-a
The name can be up to 256 characters long.

Availability Zone Info
Choose the zone in which your subnet will reside, or let Amazon choose one for you.
United States (N. Virginia) / us-east-1a

IPv4 VPC CIDR block Info
Choose the VPC's IPv4 CIDR block for the subnet. The subnet's IPv4 CIDR must lie within this block.
10.0.0.0/22

IPv4 subnet CIDR block
10.0.1.0/24 256 IPs

Repeat Steps 1-3 for the other subnet. When you're finished it should appear similar to the screenshot below:



Subnets (4) Info

Last updated less than a minute ago

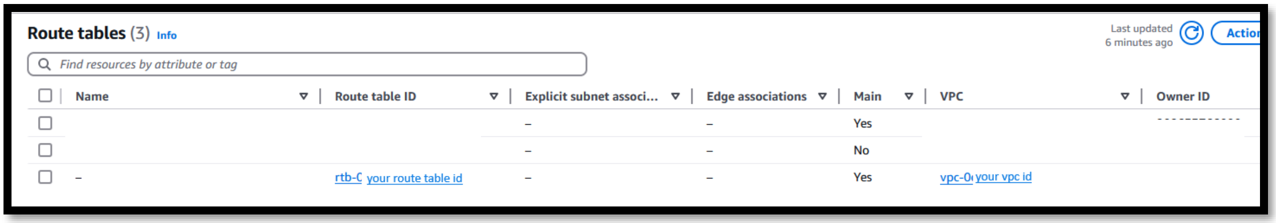
Find resources by attribute or tag

□	Name	Subnet ID	State	VPC	Block Public...	IPv4 CIDR
□	app-subnet-b	subnet-00d8080527e8c61b7	✓ Available	vpc-0da3017ea1b6eb40b | lb-web-lab-vpc	Off	10.0.2.0/24
□	app-subnet-a	subnet-0d2fe56774058a85b	✓ Available	vpc-0da3017ea1b6eb40b | lb-web-lab-vpc	Off	10.0.1.0/24

Routes

In AWS when you create a VPC, a route table is created by default. When we created our VPC earlier, unbeknownst to us, it created a route table automatically. We are going to name the route table, create an internet gateway, and create a default route so our EC2 instances can get to the internet.

1) On the VPC dashboard click on Route tables.



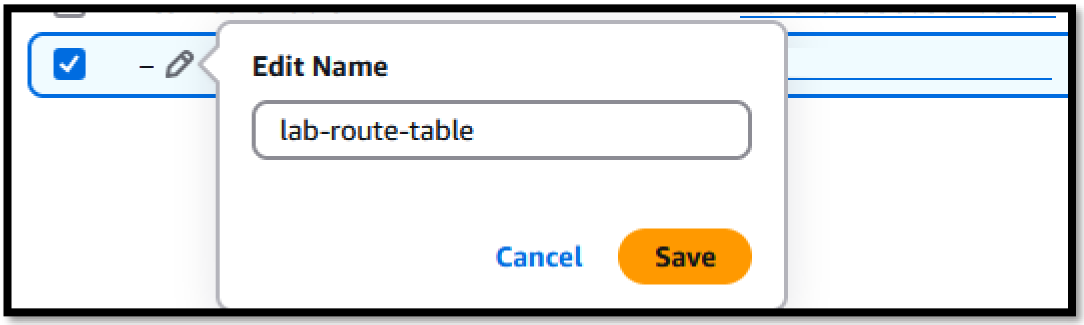
Route tables (3) Info

Last updated 6 minutes ago
Actio
Find resources by attribute or tag

□	Name	Route table ID	Explicit subnet associ...	Edge associations	Main	VPC	Owner ID
□			–	–	Yes		------------
□			–	–	No		
□	–	rtb-C your route table id	–	–	Yes	vpc-0 your vpc id	

*For brevity I've removed information on some of my other routes. You should see just (1) route without a name. This route will have the VPC ID that matches your VPC and it should be a **Main** route table.*

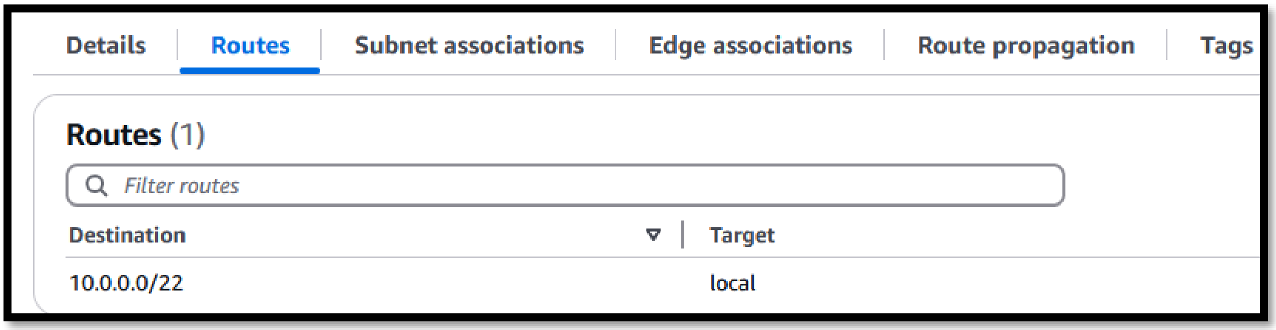
2) Click on the route and there should be an edit icon under the blank name. Rename it and click Save.



Edit Name

lab-route-table

3) Next, click on routes at the bottom and you'll notice it automatically has a route for your entire /22 VPC. Your route table does not however, have a default route to the internet.



Routes (1)

Destination	Target
10.0.0.0/22	local

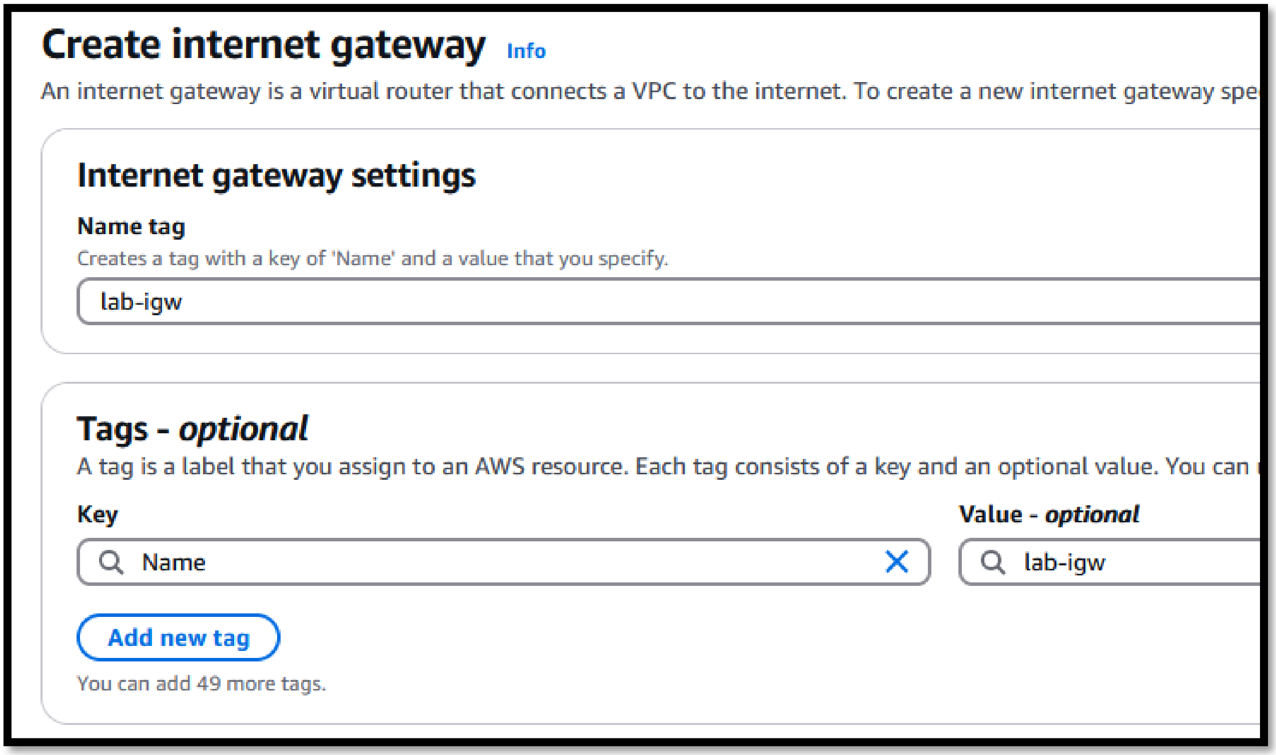
Internet Gateway

Now we’ve named our routing table and verified intra-VPC routing, we need to create an internet gateway. An internet gateway allows inbound and outbound connectivity from our VPC to the internet.

1) On the VPC dashboard we will click on Internet gateways.

2) Next you will click Create internet gateway.

3) All you must do is provide a name. For me, I am going to abbreviate internet gateway with igw. It will keep my name shorter, and it is an industry recognized abbreviation.



Create internet gateway Info

An internet gateway is a virtual router that connects a VPC to the internet. To create a new internet gateway spe

Internet gateway settings

Name tag

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

lab-igw

Tags - *optional*

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can

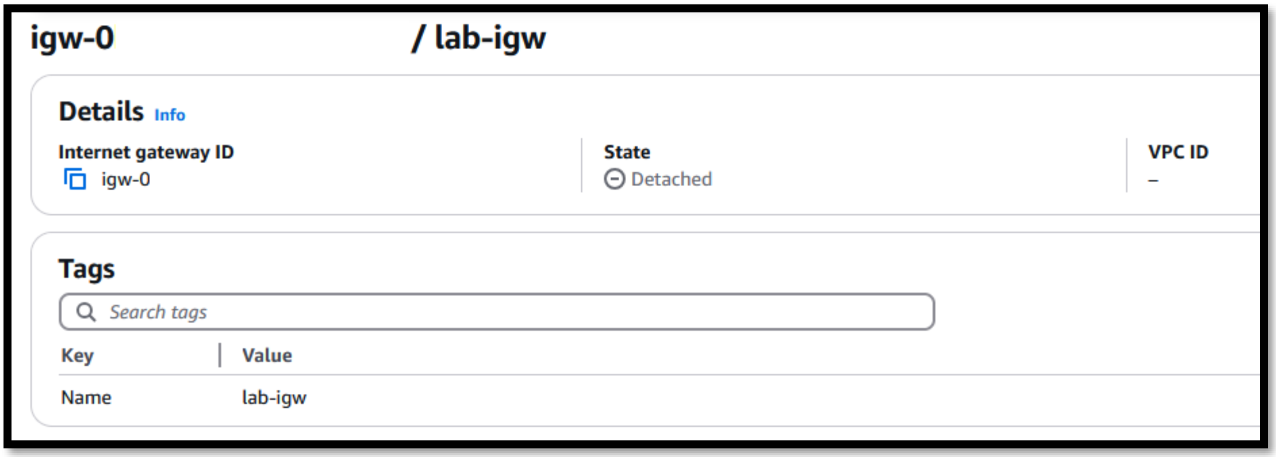
Key: Name

Value - *optional*: lab-igw

You can add 49 more tags.

4) When you are finished click Create internet gateway.

5) Lastly, the igw may take a few minutes to complete and you will want to check the state. It will most likely be detached and that is expected behavior. All you have left to do is attach it to your VPC.



igw-0 / lab-igw

Details Info

Internet gateway ID: igw-0

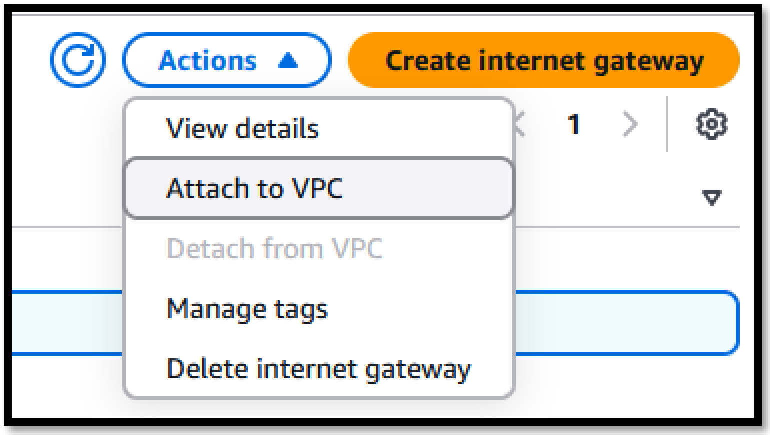
State: Detached

VPC ID: –

Tags

Search tags

Key	Value
Name	lab-igw



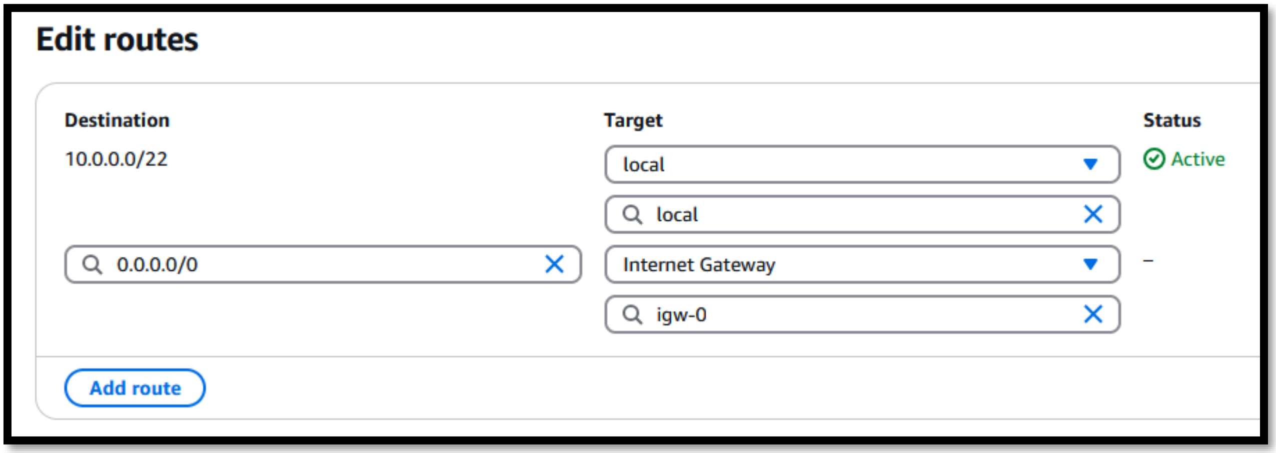
Default Route

Now you've created a VPC, two subnets, and an internet gateway attached to your VPC. All that is left is to create default route pointing to your internet gateway.

1) Go back to Route tables on the VPC dashboard and open your route table you named earlier.

2) In the route table settings click on the Routes button **Routes** and next click the Edit routes button. **Edit routes**

3) Click on Add route and create your route so it looks like the screenshot below:



Edit routes

Destination	Target	Status
10.0.0.0/22	local ▼ 🔍 local ✕	✓ Active
🔍 0.0.0.0/0 ✕	Internet Gateway ▼ 🔍 igw-0 ✕	–

Recap

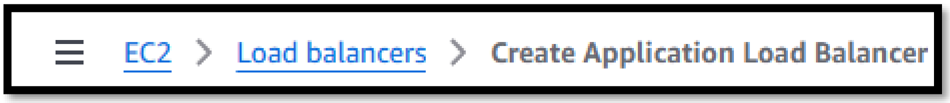
Before moving on to the next section ensure you've created and configured the following:

- ✓ A VPC.
- ✓ (2) Subnets, one in each availability zone.
- ✓ A default route table.
- ✓ An internet gateway attached to your VPC.
- ✓ A default 0.0.0.0/0 route pointing to the internet gateway.

Server and Load Balancer Creation

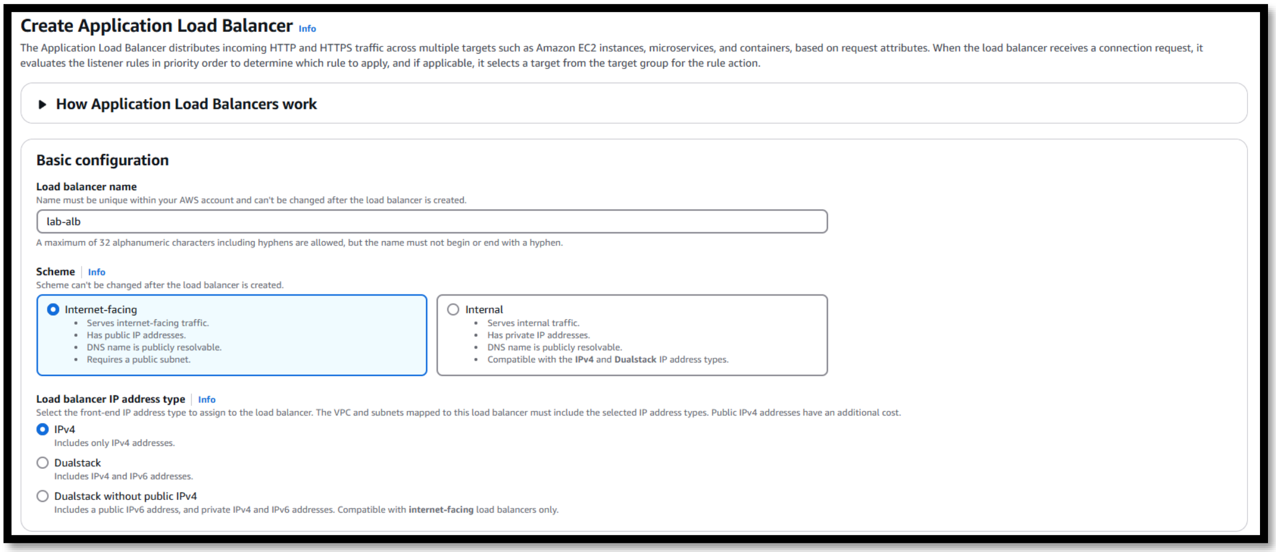
In this section of the lab, we will create our Application Load Balancer (ALB), then create two instances serving a simple HTTP website.

1) Under EC2 you will go to Load balancers and click create an Application Load Balancer.



☰ EC2 > Load balancers > Create Application Load Balancer

2) You will want to configure this as IPv4 only and it will be an internet-facing scheme.



Create Application Load Balancer Info

The Application Load Balancer distributes incoming HTTP and HTTPS traffic across multiple targets such as Amazon EC2 instances, microservices, and containers, based on request attributes. When the load balancer receives a connection request, it evaluates the listener rules in priority order to determine which rule to apply, and if applicable, it selects a target from the target group for the rule action.

▸ How Application Load Balancers work

Basic configuration

Load balancer name

Name must be unique within your AWS account and can't be changed after the load balancer is created.

lab-alb

A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

Scheme | Info

Scheme can't be changed after the load balancer is created.

● Internet-facing
- Serves internet-facing traffic.
- Has public IP addresses.
- DNS name is publicly resolvable.
- Requires a public subnet.

○ Internal
- Serves internal traffic.
- Has private IP addresses.
- DNS name is publicly resolvable.
- Compatible with the **IPv4** and **Dualstack** IP address types.

Load balancer IP address type | Info

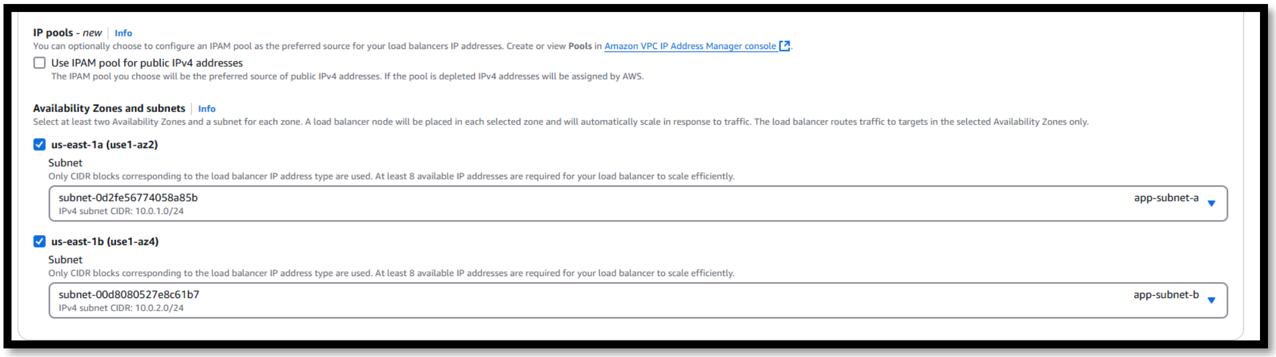
Select the front-end IP address type to assign to the load balancer. The VPC and subnets mapped to this load balancer must include the selected IP address types. Public IPv4 addresses have an additional cost.

● IPv4
Includes only IPv4 addresses.

○ Dualstack
Includes IPv4 and IPv6 addresses.

○ Dualstack without public IPv4
Includes a public IPv6 address, and private IPv4 and IPv6 addresses. Compatible with **internet-facing** load balancers only.

3) Next, ensure you select both availability zones you created earlier.



IP pools - *new* | Info

You can optionally choose to configure an IPAM pool as the preferred source for your load balancers IP addresses. Create or view **Pools** in Amazon VPC IP Address Manager console.

□ Use IPAM pool for public IPv4 addresses

The IPAM pool you choose will be the preferred source of public IPv4 addresses. If the pool is depleted IPv4 addresses will be assigned by AWS.

Availability Zones and subnets | Info

Select at least two Availability Zones and a subnet for each zone. A load balancer node will be placed in each selected zone and will automatically scale in response to traffic. The load balancer routes traffic to targets in the selected Availability Zones only.

☑ **us-east-1a (use1-az2)**

Subnet

Only CIDR blocks corresponding to the load balancer IP address type are used. At least 8 available IP addresses are required for your load balancer to scale efficiently.

subnet-0d2fe56774058a85b app-subnet-a
IPv4 subnet CIDR: 10.0.1.0/24

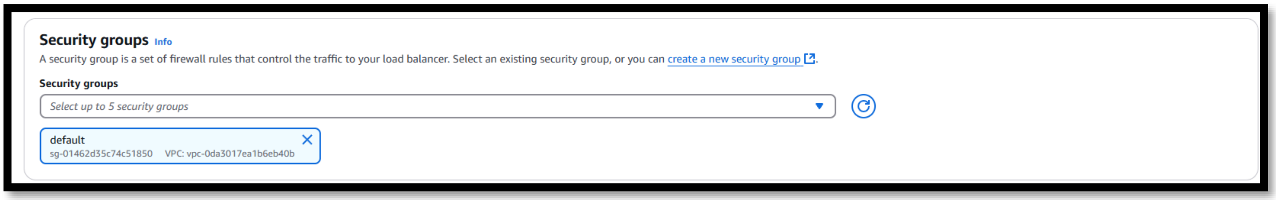
☑ **us-east-1b (use1-az4)**

Subnet

Only CIDR blocks corresponding to the load balancer IP address type are used. At least 8 available IP addresses are required for your load balancer to scale efficiently.

subnet-00d8080527e8c61b7 app-subnet-b
IPv4 subnet CIDR: 10.0.2.0/24

4) Leave the Security group settings default for now and continue.



Security groups Info

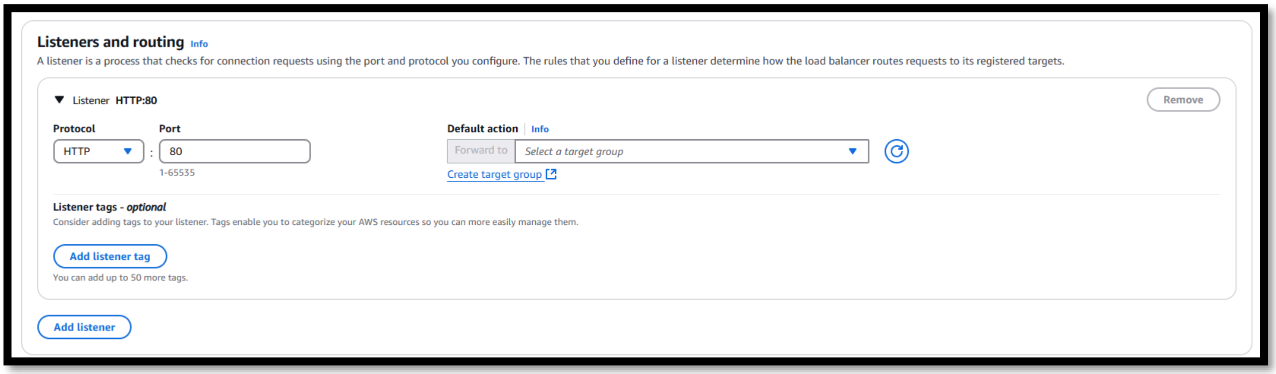
A security group is a set of firewall rules that control the traffic to your load balancer. Select an existing security group, or you can create a new security group.

Security groups

Select up to 5 security groups

default
sg-01462d35c74c51850 VPC: vpc-0da3017ea1b6eb40b

5) Next go ahead and click the Create target group button. Even though we haven't created our EC2 instances yet, we are going to create this group.



Listeners and routing Info

A listener is a process that checks for connection requests using the port and protocol you configure. The rules that you define for a listener determine how the load balancer routes requests to its registered targets.

▼ Listener **HTTP:80**

Protocol: HTTP

Port: 80
1-65535

Default action | Info

Forward to *Select a target group*

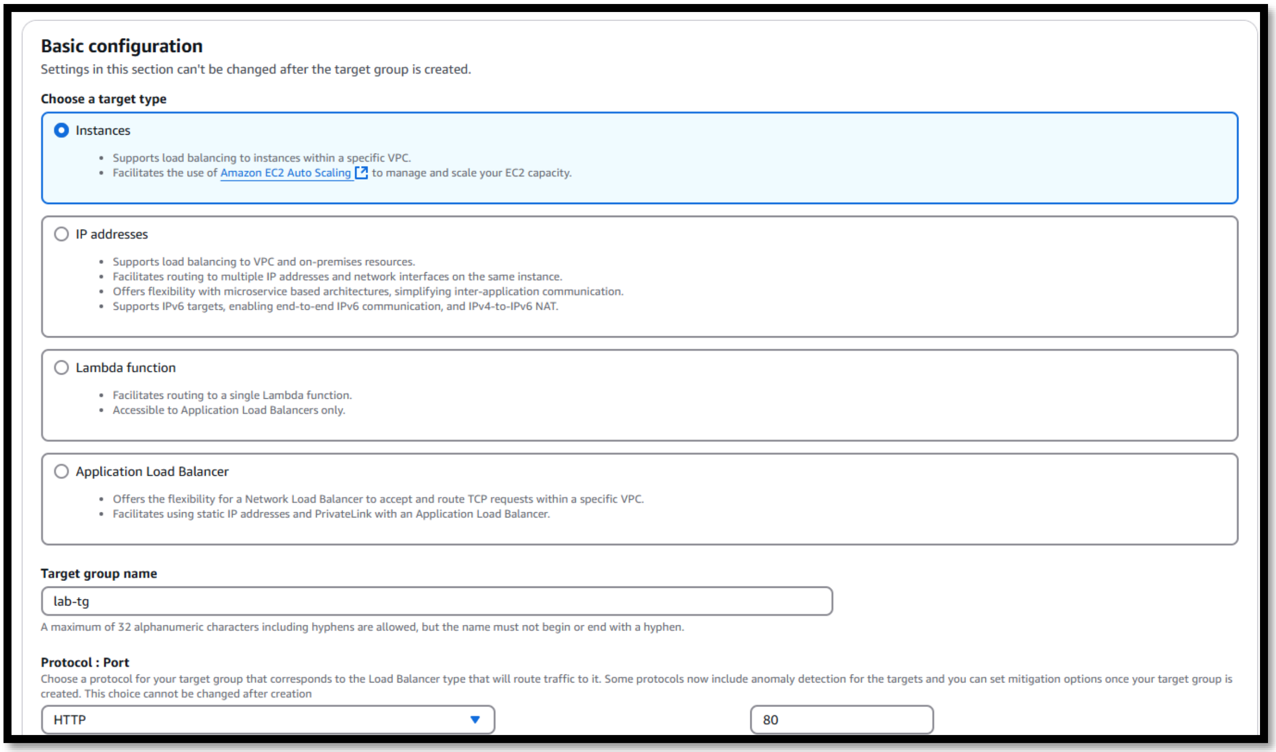
Create target group

Listener tags - ***optional***

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

You can add up to 50 more tags.

6) Give your target group a name and leave almost everything else default.



Basic configuration

Settings in this section can't be changed after the target group is created.

Choose a target type

● Instances
- Supports load balancing to instances within a specific VPC.
- Facilitates the use of Amazon EC2 Auto Scaling to manage and scale your EC2 capacity.

○ IP addresses
- Supports load balancing to VPC and on-premises resources.
- Facilitates routing to multiple IP addresses and network interfaces on the same instance.
- Offers flexibility with microservice based architectures, simplifying inter-application communication.
- Supports IPv6 targets, enabling end-to-end IPv6 communication, and IPv4-to-IPv6 NAT.

○ Lambda function
- Facilitates routing to a single Lambda function.
- Accessible to Application Load Balancers only.

○ Application Load Balancer
- Offers the flexibility for a Network Load Balancer to accept and route TCP requests within a specific VPC.
- Facilitates using static IP addresses and PrivateLink with an Application Load Balancer.

Target group name

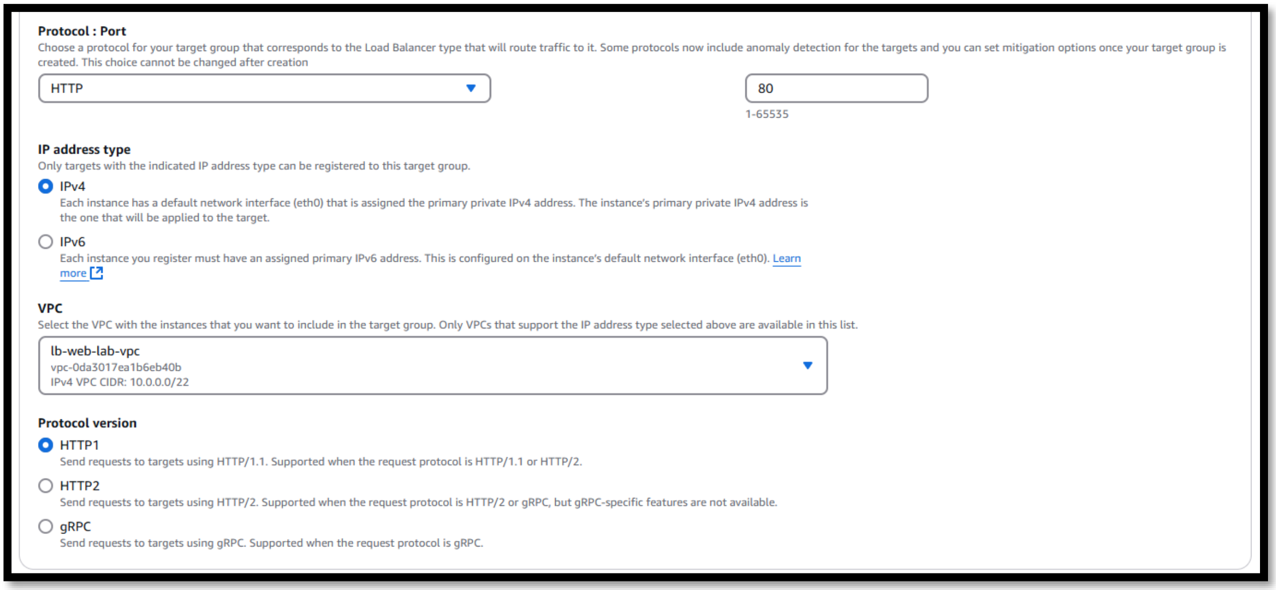
lab-tg

A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

Protocol : Port

Choose a protocol for your target group that corresponds to the Load Balancer type that will route traffic to it. Some protocols now include anomaly detection for the targets and you can set mitigation options once your target group is created. This choice cannot be changed after creation

HTTP ▼ 80



Protocol : Port

Choose a protocol for your target group that corresponds to the Load Balancer type that will route traffic to it. Some protocols now include anomaly detection for the targets and you can set mitigation options once your target group is created. This choice cannot be changed after creation

HTTP ▼ 80

1-65535

IP address type

Only targets with the indicated IP address type can be registered to this target group.

● IPv4
Each instance has a default network interface (eth0) that is assigned the primary private IPv4 address. The instance's primary private IPv4 address is the one that will be applied to the target.

○ IPv6
Each instance you register must have an assigned primary IPv6 address. This is configured on the instance's default network interface (eth0). Learn more

VPC

Select the VPC with the instances that you want to include in the target group. Only VPCs that support the IP address type selected above are available in this list.

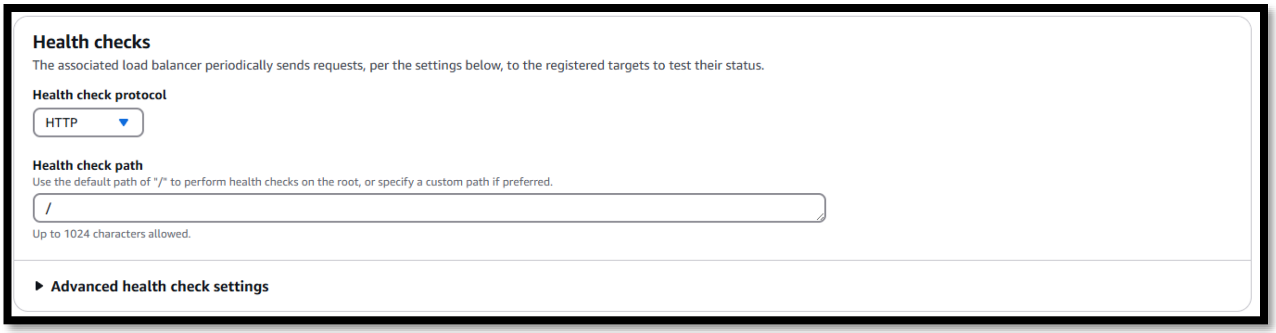
lb-web-lab-vpc
vpc-0da3017ea1b6eb40b
IPv4 VPC CIDR: 10.0.0.0/22 ▼

Protocol version

● HTTP1
Send requests to targets using HTTP/1.1. Supported when the request protocol is HTTP/1.1 or HTTP/2.

○ HTTP2
Send requests to targets using HTTP/2. Supported when the request protocol is HTTP/2 or gRPC, but gRPC-specific features are not available.

○ gRPC
Send requests to targets using gRPC. Supported when the request protocol is gRPC.



Health checks

The associated load balancer periodically sends requests, per the settings below, to the registered targets to test their status.

Health check protocol

HTTP ▼

Health check path

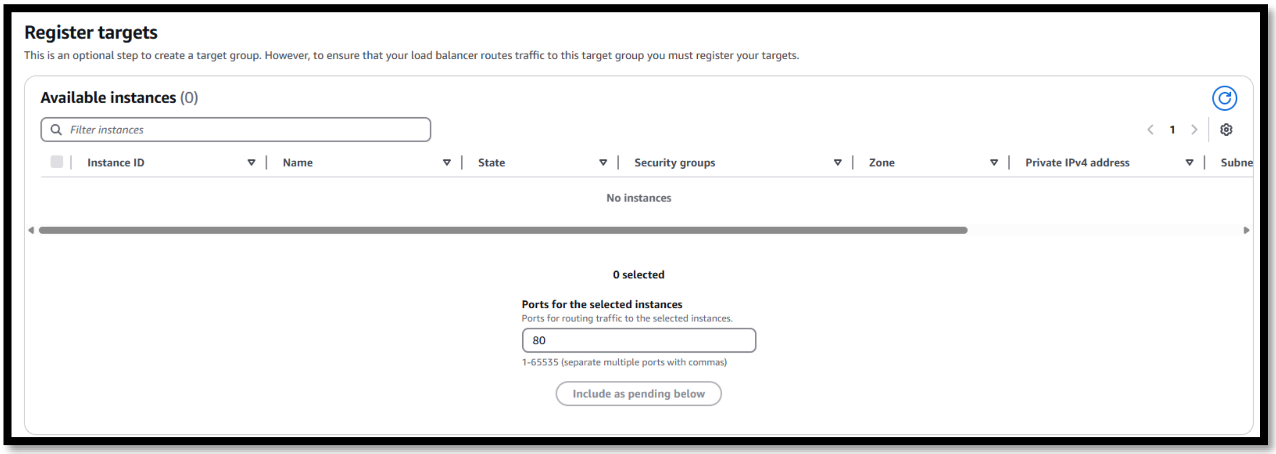
Use the default path of "/" to perform health checks on the root, or specify a custom path if preferred.

/

Up to 1024 characters allowed.

▶ **Advanced health check settings**

4) On the next page you will be asked to select targets. You have none right now and we will come back to add our instances once we create them. Continue to create the target group.



Register targets

This is an optional step to create a target group. However, to ensure that your load balancer routes traffic to this target group you must register your targets.

Available instances (0)

Filter instances

□	Instance ID	Name	State	Security groups	Zone	Private IPv4 address	Subne
	No instances						

0 selected

Ports for the selected instances

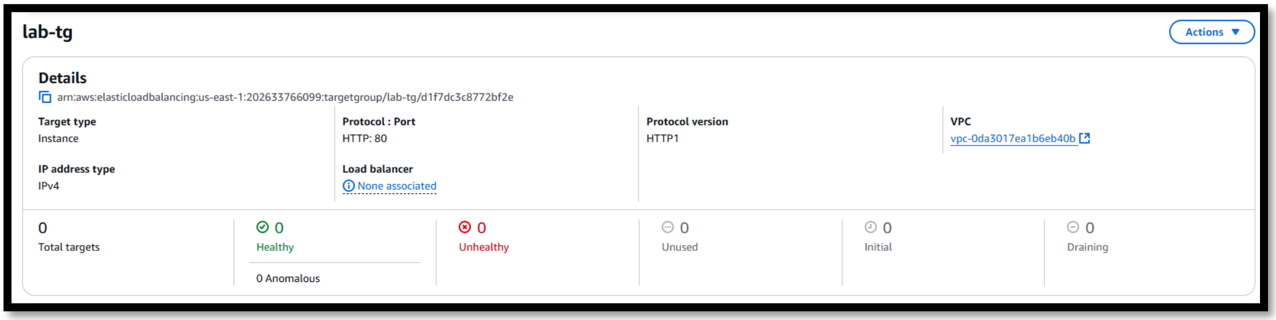
Ports for routing traffic to the selected instances.

80

1-65535 (separate multiple ports with commas)

Include as pending below

5) Your target group is now created and empty.



lab-tg

Details

arn:aws:elasticloadbalancing:us-east-1:202633766099:targetgroup/lab-tg/d1f7dc3c8772bf2e

Target type
Instance

Protocol : Port
HTTP: 80

Protocol version
HTTP1

VPC
vpc-0da3017ea1b6eb40b

IP address type
IPv4

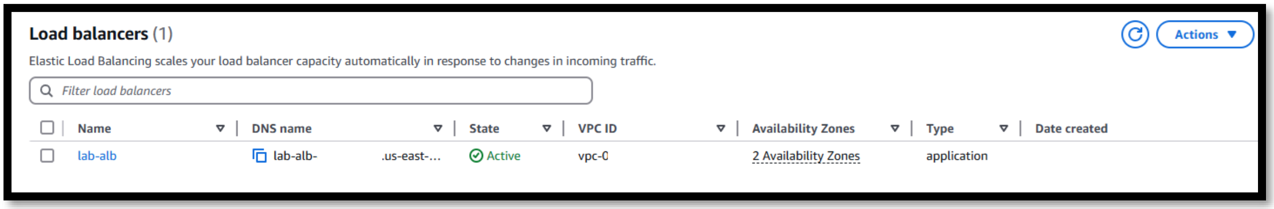
Load balancer
None associated

0	✓ 0	⊗ 0	0	0	0
Total targets	Healthy	Unhealthy	Unused	Initial	Draining
	0 Anomalous				

6) Go back to your "Create application load balancer" tab and click the Create load balancer button.

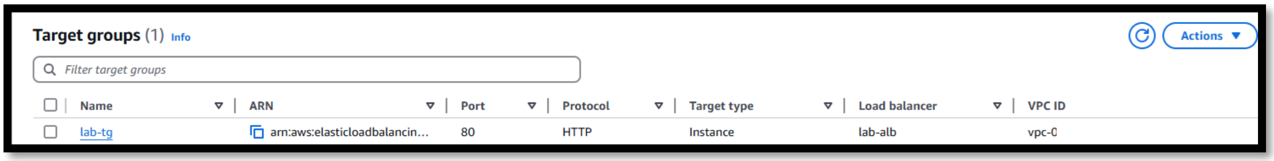
7) With your new target group selected you should be able to create this.

8) When finished you should have a load balancer and target group configured similarly to what is seen below:



Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.

□	Name	DNS name	State	VPC ID	Availability Zones	Type	Date created
□	lab-alb	lab-alb- .us-east-…	✓ Active	vpc-0	2 Availability Zones	application	

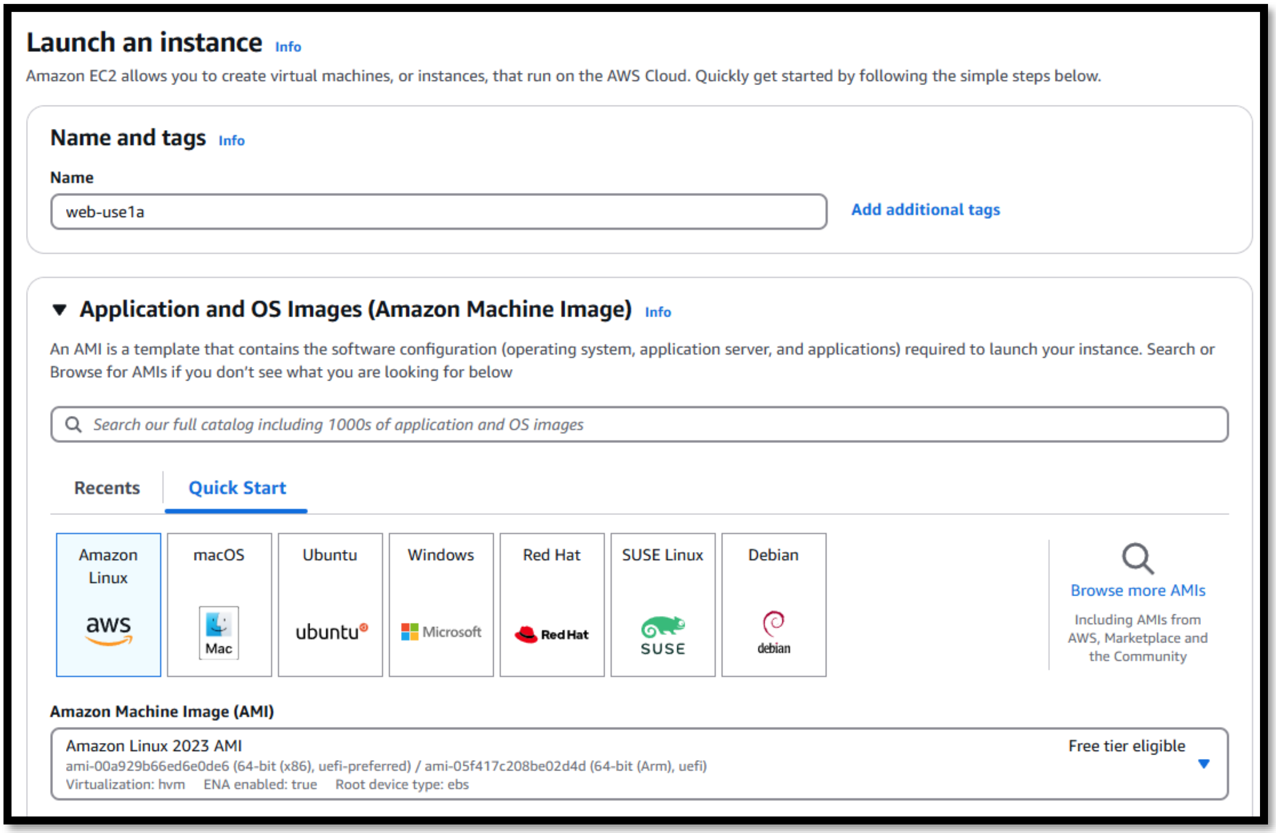


□	Name	ARN	Port	Protocol	Target type	Load balancer	VPC ID
□	lab-tg	arn:aws:elasticloadbalancin…	80	HTTP	Instance	lab-alb	vpc-0

EC2 Instances

In this section we will create our two web servers and configure them as basic HTTP servers. I will keep everything free-tier so you can follow along.

1) Go to EC2 and Launch an instance.
   a. I am going to use Amazon Linux and keep most settings default.
   b. Give it an appropriate name.



Launch an instance

Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags

Info

Name: web-use1a

Add additional tags

▼ Application and OS Images (Amazon Machine Image)

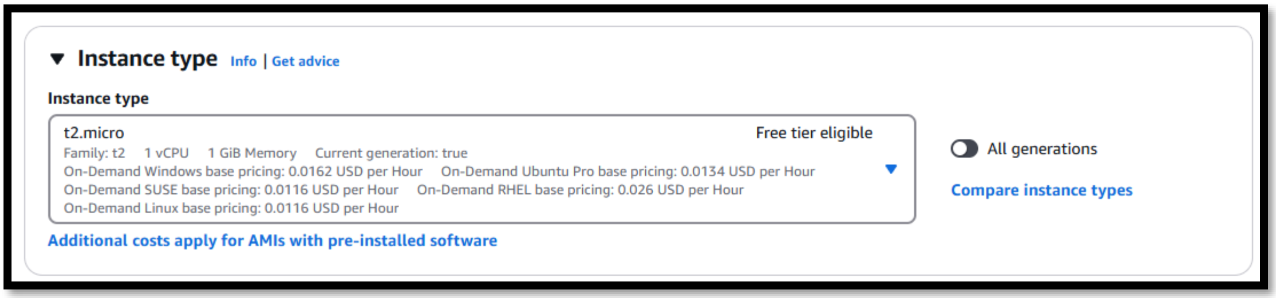
Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search our full catalog including 1000s of application and OS images

Amazon Machine Image (AMI)

Amazon Linux 2023 AMI — Free tier eligible
ami-00a929b66ed6e0de6 (64-bit (x86), uefi-preferred) / ami-05f417c208be02d4d (64-bit (Arm), uefi)
Virtualization: hvm ENA enabled: true Root device type: ebs



▼ Instance type

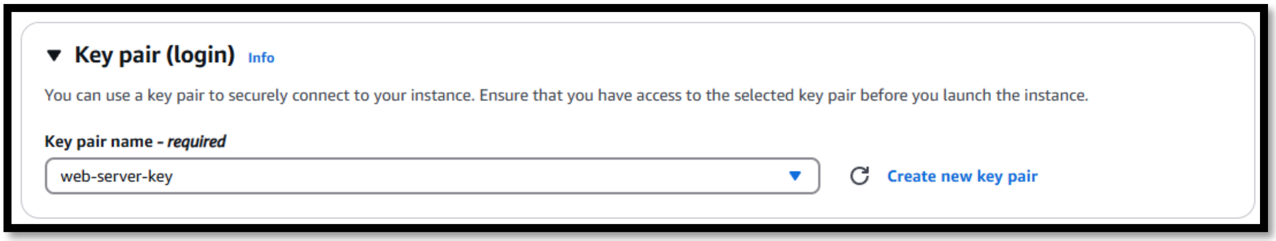
Info | Get advice

Instance type

t2.micro — Free tier eligible
Family: t2 1 vCPU 1 GiB Memory Current generation: true
On-Demand Windows base pricing: 0.0162 USD per Hour On-Demand Ubuntu Pro base pricing: 0.0134 USD per Hour
On-Demand SUSE base pricing: 0.0116 USD per Hour On-Demand RHEL base pricing: 0.026 USD per Hour
On-Demand Linux base pricing: 0.0116 USD per Hour

Additional costs apply for AMIs with pre-installed software

2) Create a new SSH key pair or use an existing one you've already created.



Key pair (login)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

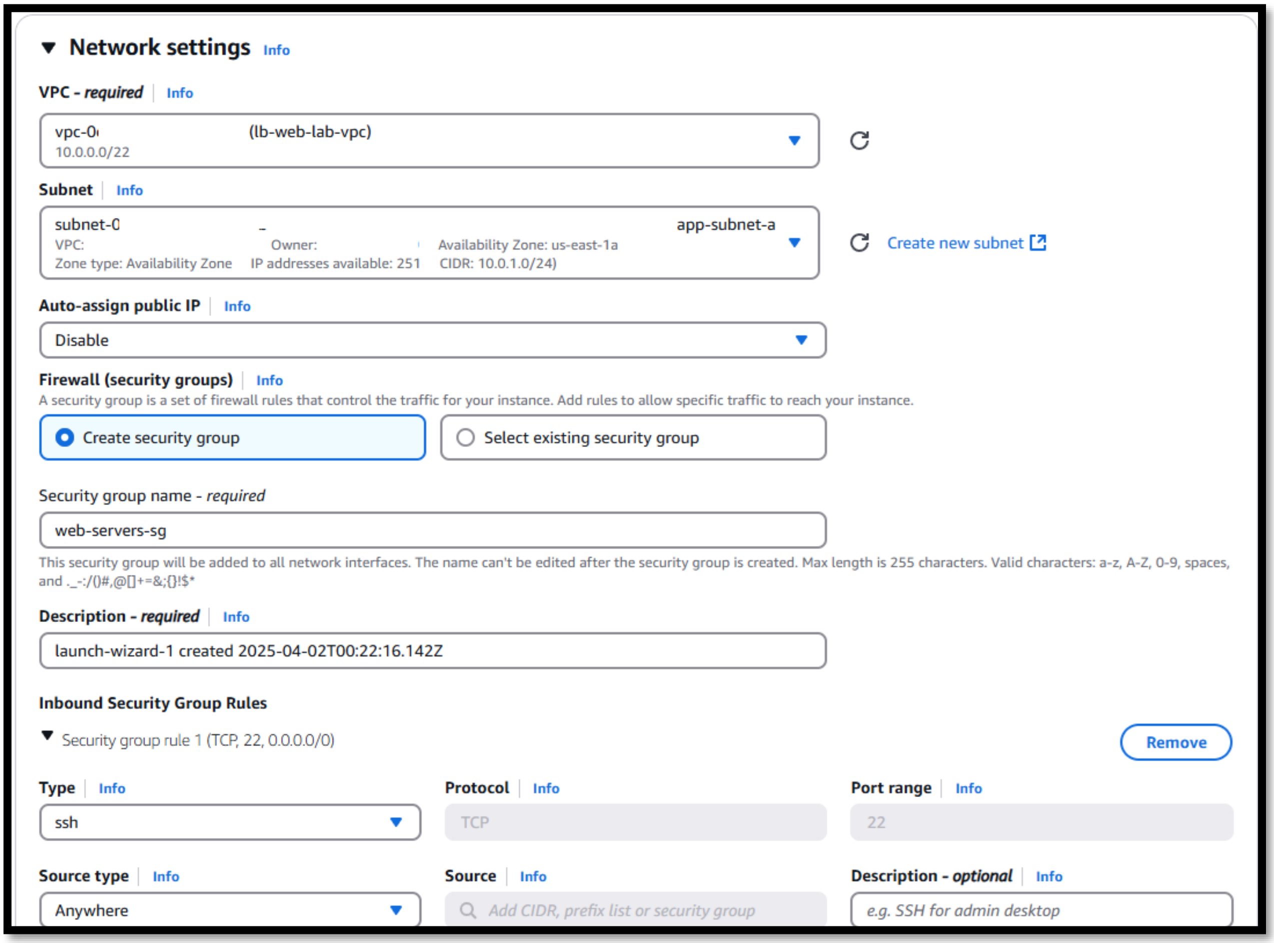
Key pair name - *required*

web-server-key

3) Ensure you have the correct VPC and subnet selected.

4) Use the default security group and allow SSH for now from anywhere. Give it a different name to identify it by.

5) Give your server a public IP by changing Auto-assign public IP to enable.



Network settings

VPC - *required*

vpc-0 (lb-web-lab-vpc)
10.0.0.0/22

Subnet

subnet-0 _ app-subnet-a
VPC: Owner: Availability Zone: us-east-1a
Zone type: Availability Zone IP addresses available: 251 CIDR: 10.0.1.0/24)

Auto-assign public IP

Disable

Firewall (security groups)

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*

web-servers-sg

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*

launch-wizard-1 created 2025-04-02T00:22:16.142Z

Inbound Security Group Rules

Security group rule 1 (TCP, 22, 0.0.0.0/0)

Type: ssh

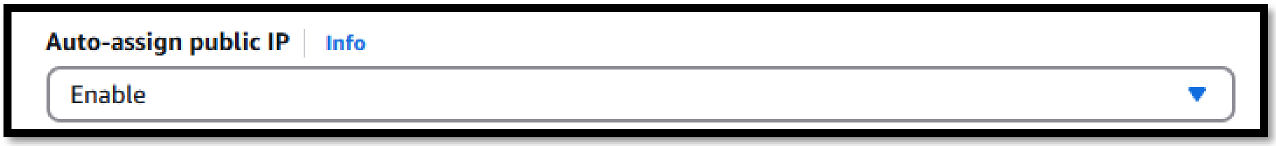
Protocol: TCP

Port range: 22

Source type: Anywhere

Source: Add CIDR, prefix list or security group

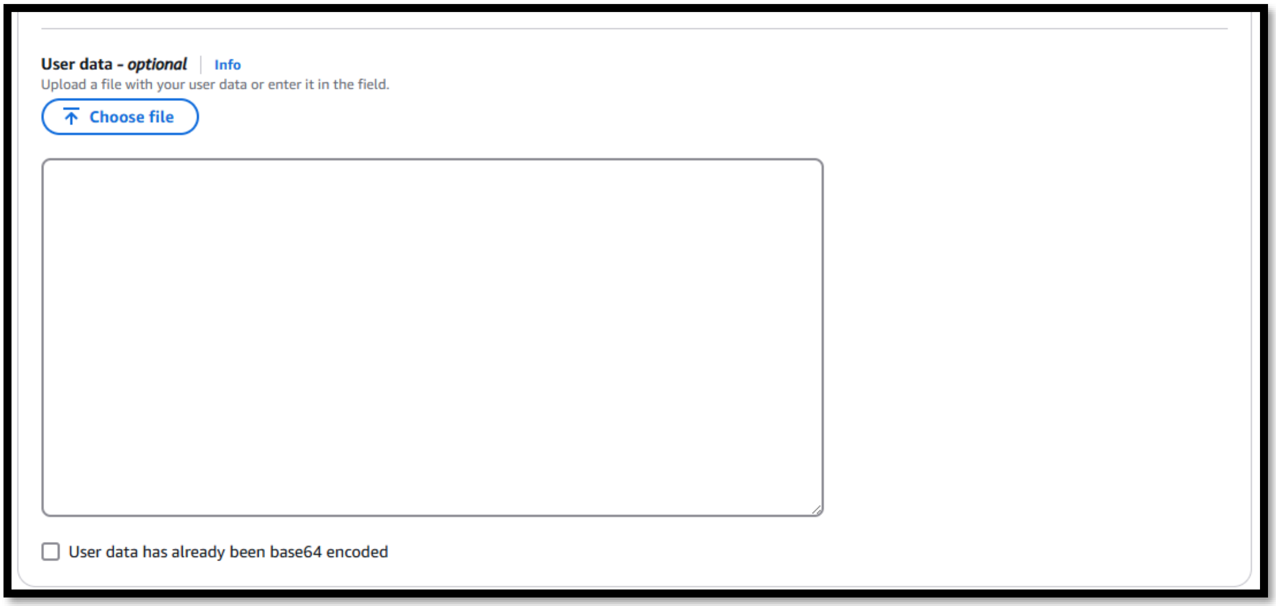
Description - *optional*: e.g. SSH for admin desktop



Auto-assign public IP | Info

Enable ▼

6) I want you to go to Advanced options and look for user data. This is data you can pass directly to the instance when it is launched. This allows for quick bootstrapping and automation. We are not going to use this in this lab because we're going to configure our servers manually. I want to point this out because it's something to note of importance. All major cloud providers offer this.

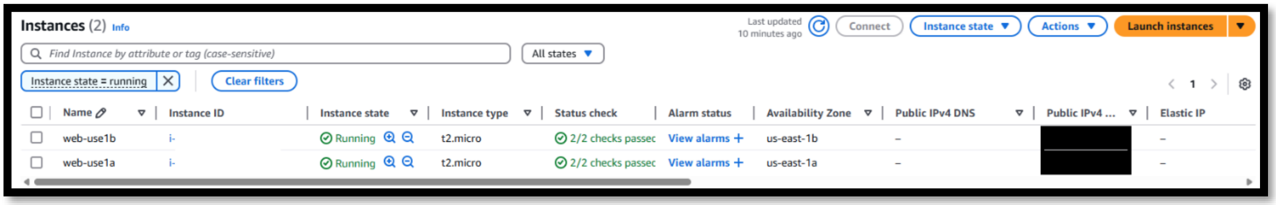


User data - *optional* | Info

Upload a file with your user data or enter it in the field.

□ User data has already been base64 encoded

7) Refresh your EC2 dashboard and you should have two public instances with public Ips.



Instances (2) Info

Last updated 10 minutes ago
Connect
Instance state ▼
Actions ▼
Launch instances ▼
Find Instance by attribute or tag (case-sensitive)
All states ▼
Clear filters
1

Instance state = running ✕

□	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
□	web-use1b	i-	✓ Running	t2.micro	✓ 2/2 checks passec	View alarms +	us-east-1b	–	[redacted]	–
□	web-use1a	i-	✓ Running	t2.micro	✓ 2/2 checks passec	View alarms +	us-east-1a	–	[redacted]	–

Recap

Before moving on to the next section ensure you've created and configured the following:

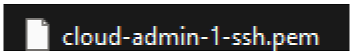
- ✓ A load balancer.
- ✓ An empty target group.
- ✓ (2) EC2 free-tier instances running Amazon Linux, one in each availability zone.
- ✓ (2) public IPs, one attached to each instance created.

Connecting to EC2 Instances

This section will walk-through how to connect to our EC2 instances initially and how to setup a simple HTTP server on our EC2 instances. Then we will finalize our load balancer config and test it all out.

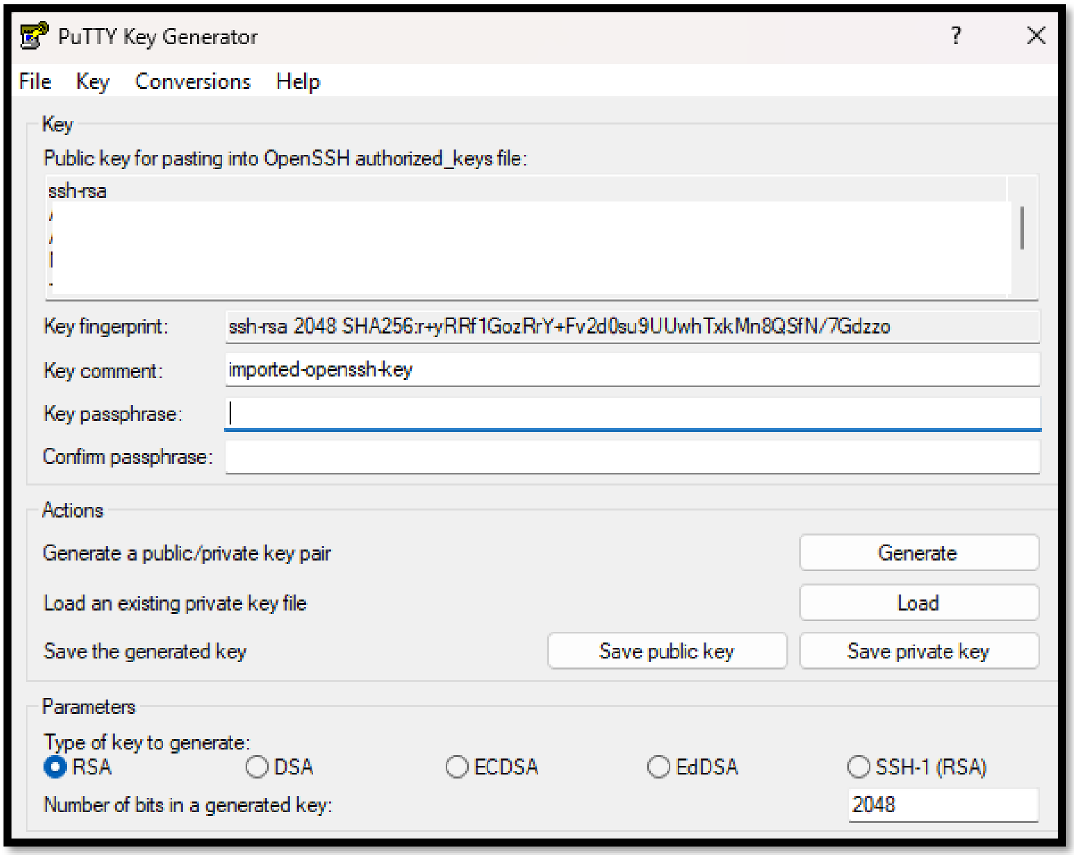
PuTTY and PuTTYgen

Before we get started you will want PuTTY for Windows and PuTTYgen (comes with PuTTY) to create a .ppk file from our .pem key we downloaded earlier when we created our EC2 instances. My private key is still in my downloads folder as seen below.



cloud-admin-1-ssh.pem

1) First, open PuTTYgen and click File, then Load private key.



Public key for pasting into OpenSSH authorized_keys file:

ssh-rsa [redacted]

Key fingerprint: ssh-rsa 2048 SHA256:r+yRRf1GozRrY+Fv2d0su9UUwhTxkMn8QSfN/7Gdzzo

Key comment: imported-openssh-key

Key passphrase:

Confirm passphrase:

Generate a public/private key pair

Load an existing private key file

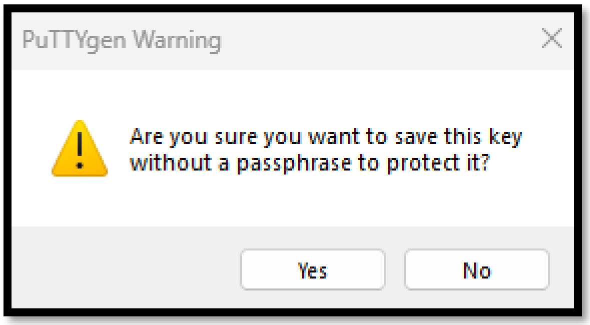
Save the generated key

Type of key to generate:

● RSA ○ DSA ○ ECDSA ○ EdDSA ○ SSH-1 (RSA)

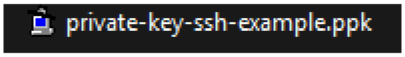
Number of bits in a generated key: 2048

2) Next, I am going to just hit save private key and I will get a warning, and I hit yes. I don’t care about passwords because this is a lab. Obviously, in the real world you would want to utilize a passphrase.

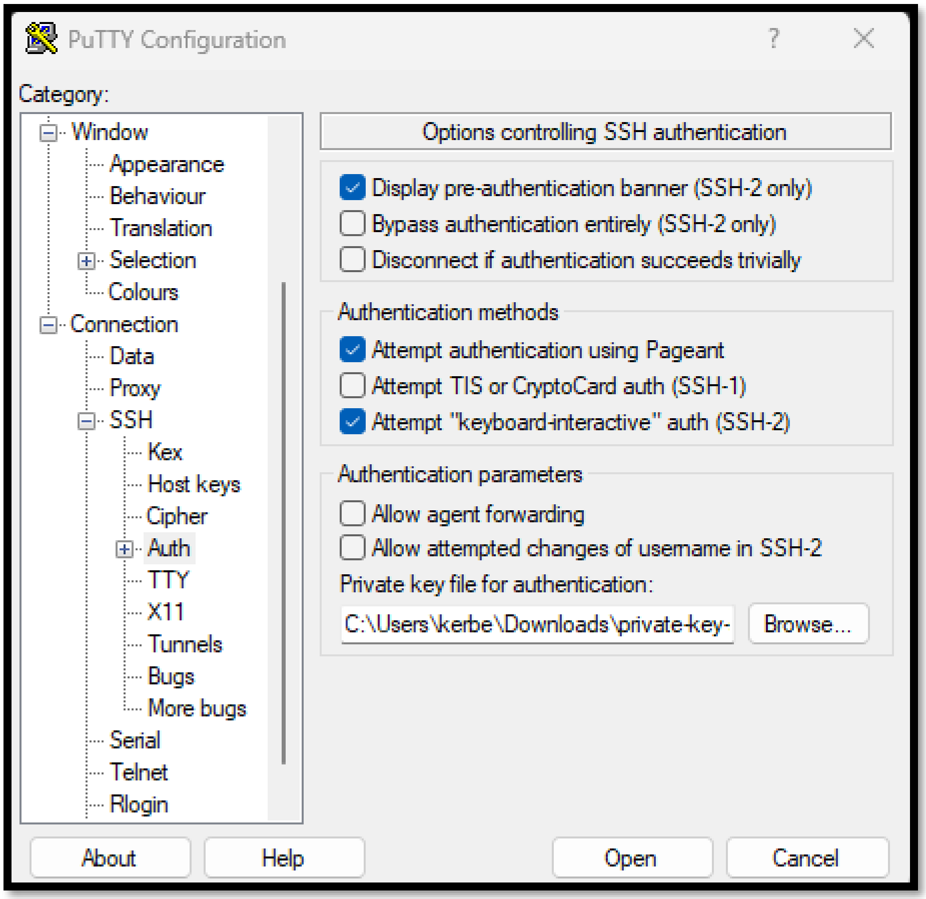

PuTTYgen Warning

Are you sure you want to save this key without a passphrase to protect it?

3) I save this as a .ppk and now I have my key to connect to my instance.


private-key-ssh-example.ppk

4) Now we will open PuTTY and drill to Connection, SSH, Auth, Private key file for authentication:



Options controlling SSH authentication

☑ Display pre-authentication banner (SSH-2 only)
□ Bypass authentication entirely (SSH-2 only)
□ Disconnect if authentication succeeds trivially

Authentication methods

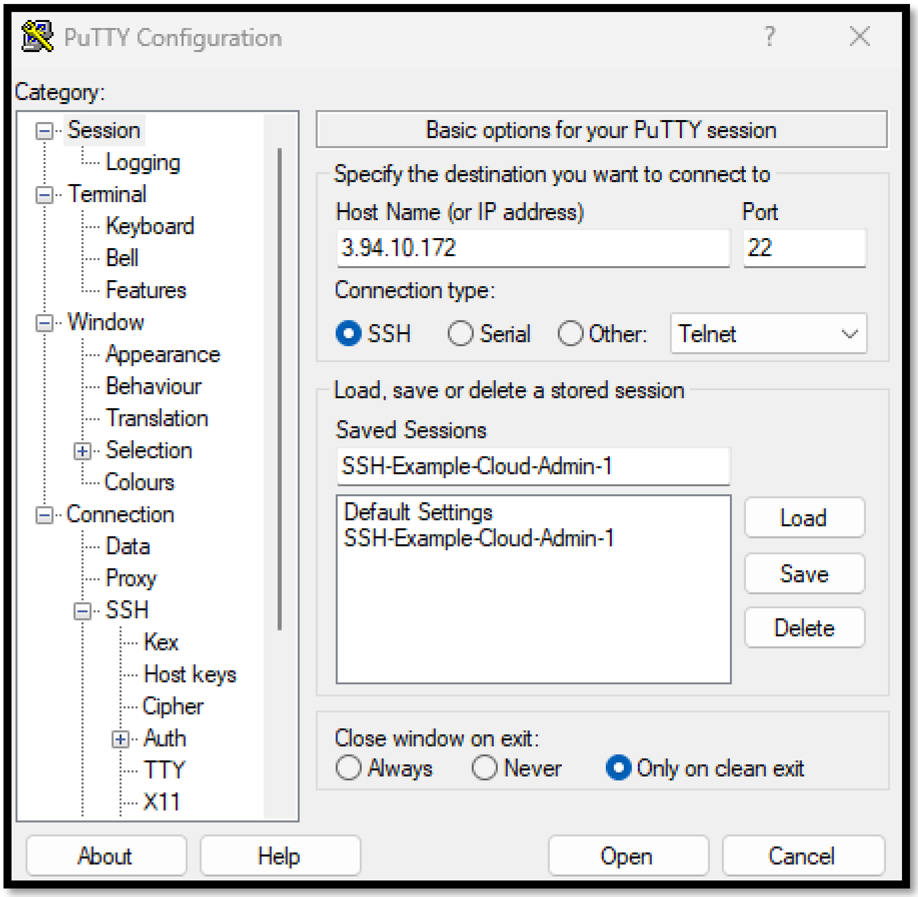
☑ Attempt authentication using Pageant
□ Attempt TIS or CryptoCard auth (SSH-1)
☑ Attempt "keyboard-interactive" auth (SSH-2)

Authentication parameters

□ Allow agent forwarding
□ Allow attempted changes of username in SSH-2

Private key file for authentication:
C:\Users\kerbe\Downloads\private-key- Browse...

5) Provide your newly created .ppk then go back to Session and add your IP address and click SSH. I also recommend saving these settings so you don’t have to configure this every time. I saved mine as “SSH-Example-Cloud-Admin-1”



PuTTY Configuration

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): 3.94.10.172

Port: 22

Connection type: ● SSH ○ Serial ○ Other: Telnet

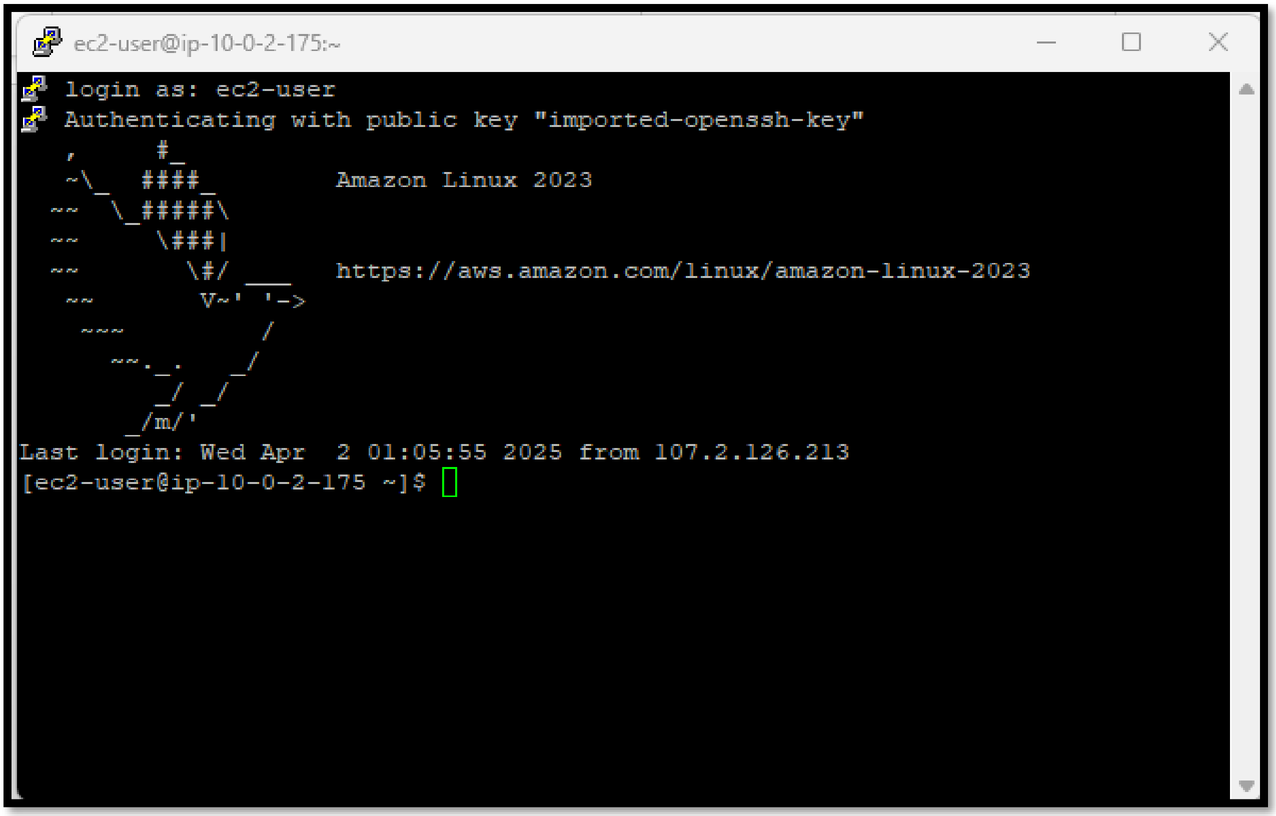
Load, save or delete a stored session

Saved Sessions: SSH-Example-Cloud-Admin-1

Default Settings
SSH-Example-Cloud-Admin-1

Close window on exit: ○ Always ○ Never ● Only on clean exit

6) You will now be asked for a user, and you will type **ec2-user** exactly as you see it here.



```
login as: ec2-user
Authenticating with public key "imported-openssh-key"
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Wed Apr  2 01:05:55 2025 from 107.2.126.213
[ec2-user@ip-10-0-2-175 ~]$ 
```

You are now successfully connected to your EC2 instances. Ensure you can connect to both!

Recap

Before moving on to the next section ensure you've created and configured the following:

- ✓ You can successfully connect and log into both EC2 instances
- ✓ Both EC2 instances can successfully ping 8.8.8.8 or any external IP

Configure Web Servers

Now that you can log into both servers, we need to configure them to be actual servers. For this lab we are going to use python and a useful library for python called simple http server. You will create a simple configuration on both machines with only one difference, the names on the html pages.

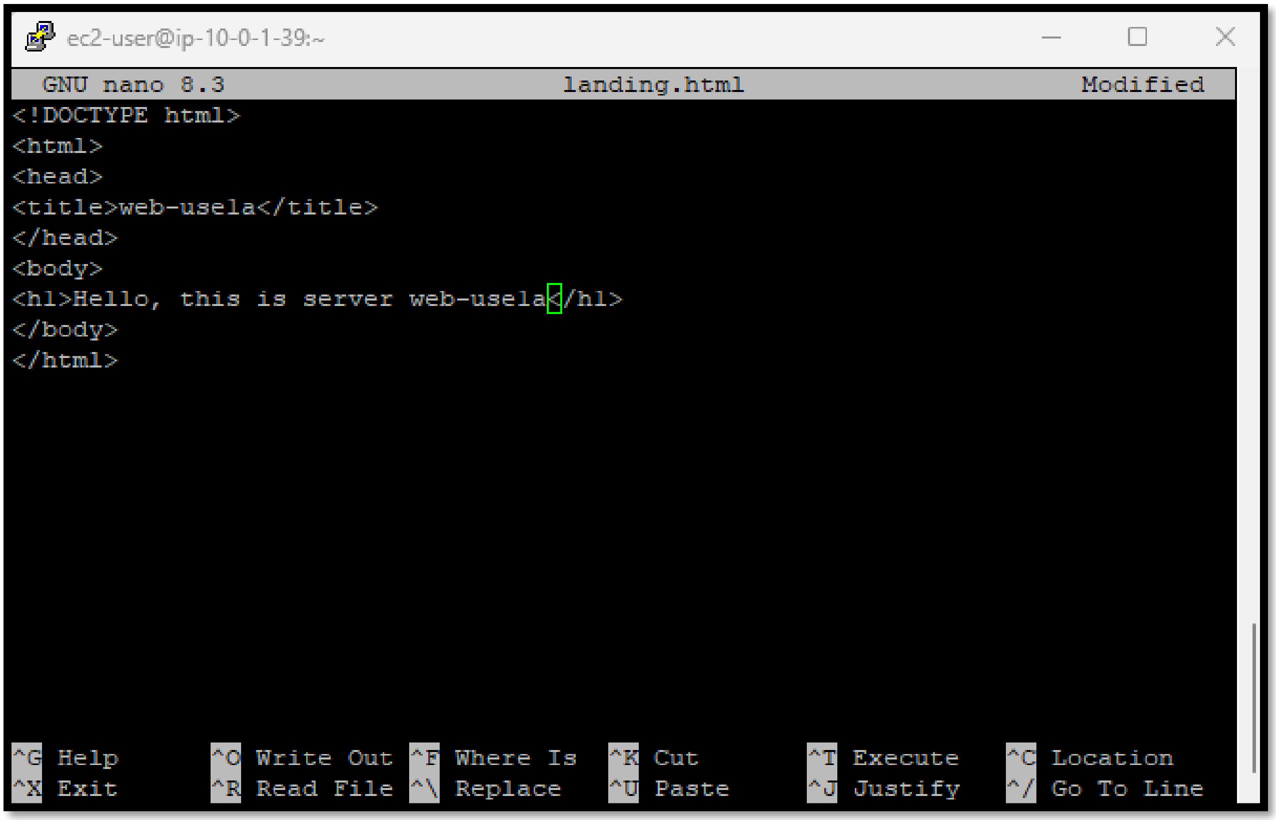
1) Run the following commands on each EC2 instance.

```
sudo yum install python3 -y
sudo yum install python3-pip -y
python3 -m pip install simple_http_server
```

2) In your user directly create a basic html file and call it landing.html. You can use nano or vim (I prefer nano). You can do this by running the following command.

```
nano landing.html
```

3) Copy and paste the following HTML code and make sure you modify the name for which server page you’re creating. When finished hit CTRL + O key to save then CTRL + X



```
  GNU nano 8.3                    landing.html                       Modified
<!DOCTYPE html>
<html>
<head>
<title>web-usela</title>
</head>
<body>
<h1>Hello, this is server web-usela</h1>
</body>
</html>
```

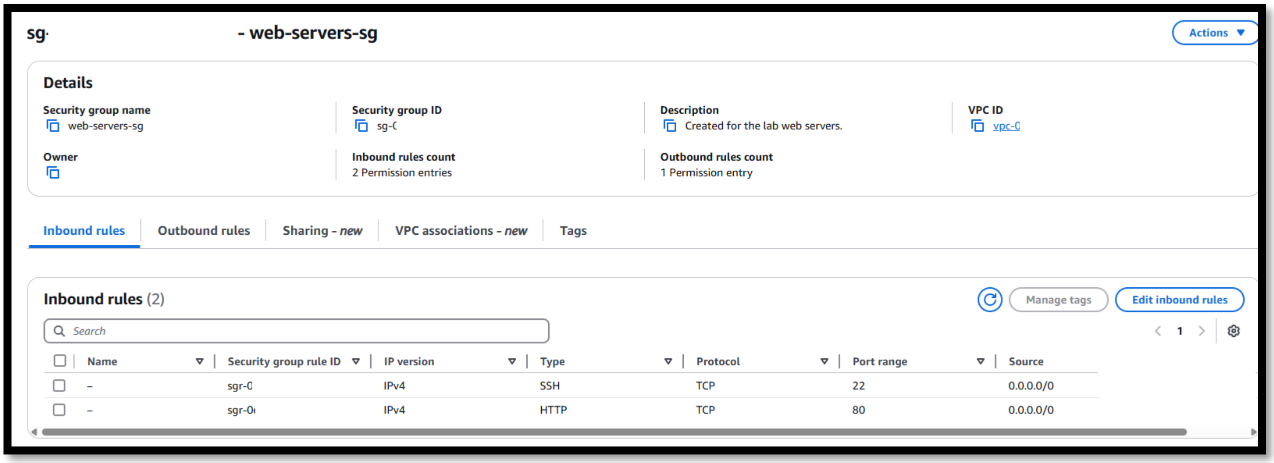
4) Lastly, run the following command from the directory your file is in (it should be your user directory).

```
sudo python3 -m http.server 80
```

5) You should now see a running web server service in the CLI.

6) To test this we need to allow HTTP 80- from anywhere in the world. Go to VPC then Security groups in the AWS consol.

7) Find your security group from earlier and click the Edit inbound rules button.



sg· - web-servers-sg

Details

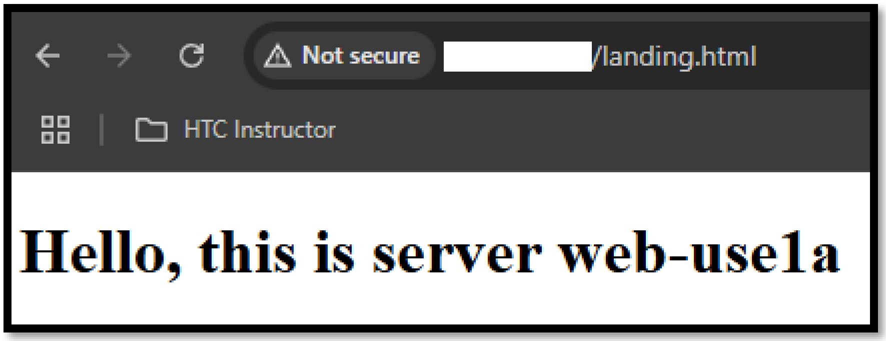
Security group name web-servers-sg	Security group ID sg-(	Description Created for the lab web servers.	VPC ID vpc-0
Owner	Inbound rules count 2 Permission entries	Outbound rules count 1 Permission entry	

Inbound rules (2)

□	Name	Security group rule ID	IP version	Type	Protocol	Port range	Source
□	–	sgr-0	IPv4	SSH	TCP	22	0.0.0.0/0
□	–	sgr-0	IPv4	HTTP	TCP	80	0.0.0.0/0

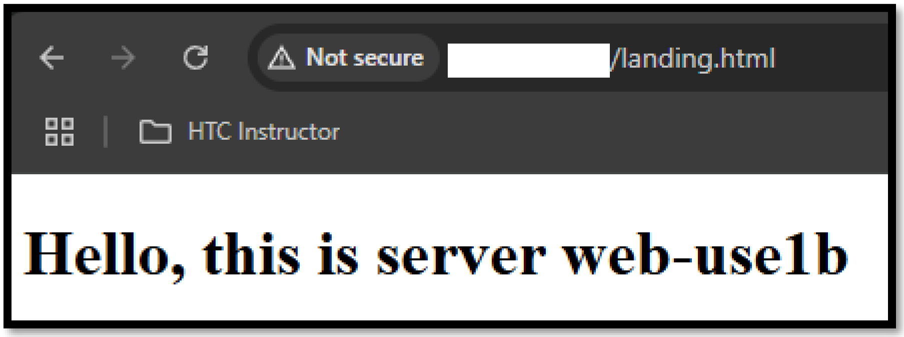
8) Create a new policy to allow TCP 80 from anywhere inbound.

9) Open your web browser and put the public IP address of your EC2 instance in, starting with http://x.x.x.x/landing.html. You should now see both pages displayed! Congratulations, you've got two successful HTTP servers.



[redacted]/landing.html

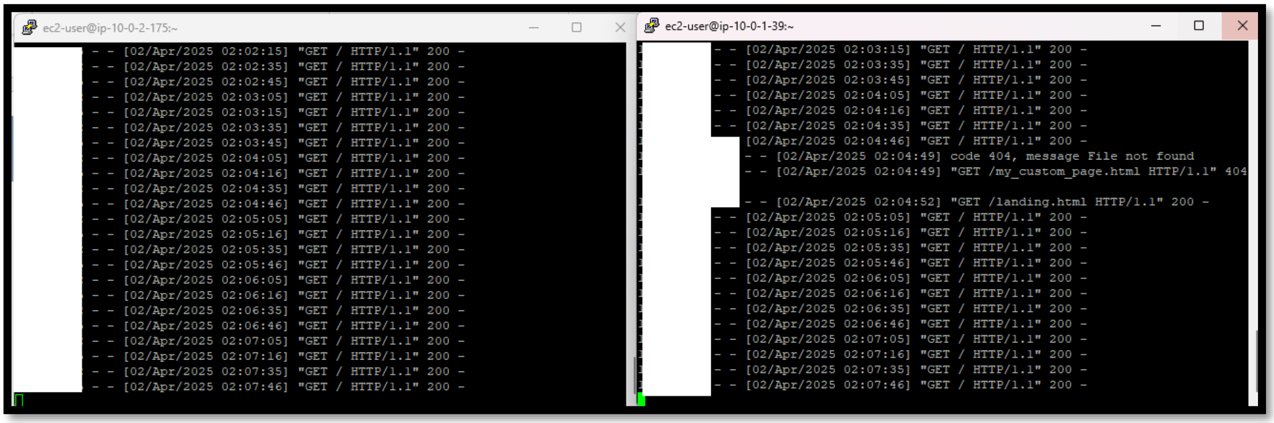
Hello, this is server web-use1a



[redacted]/landing.html

Hello, this is server web-use1b

10) What is also great about using python simple http server is that it also allows you to confirm your traffic. If you look at the CLI of your PuTTY sessions, you'll see the GET Requests as they come in.



```
[redacted] - - [02/Apr/2025 02:02:15] "GET / HTTP/1.1" 200 -
 - - [02/Apr/2025 02:02:35] "GET / HTTP/1.1" 200 -
 - - [02/Apr/2025 02:02:45] "GET / HTTP/1.1" 200 -
 - - [02/Apr/2025 02:03:05] "GET / HTTP/1.1" 200 -
 - - [02/Apr/2025 02:03:15] "GET / HTTP/1.1" 200 -
 - - [02/Apr/2025 02:03:35] "GET / HTTP/1.1" 200 -
 - - [02/Apr/2025 02:03:45] "GET / HTTP/1.1" 200 -
 - - [02/Apr/2025 02:04:05] "GET / HTTP/1.1" 200 -
 - - [02/Apr/2025 02:04:16] "GET / HTTP/1.1" 200 -
 - - [02/Apr/2025 02:04:35] "GET / HTTP/1.1" 200 -
 - - [02/Apr/2025 02:04:46] "GET / HTTP/1.1" 200 -
 - - [02/Apr/2025 02:05:05] "GET / HTTP/1.1" 200 -
 - - [02/Apr/2025 02:05:16] "GET / HTTP/1.1" 200 -
 - - [02/Apr/2025 02:05:35] "GET / HTTP/1.1" 200 -
 - - [02/Apr/2025 02:05:46] "GET / HTTP/1.1" 200 -
 - - [02/Apr/2025 02:06:05] "GET / HTTP/1.1" 200 -
 - - [02/Apr/2025 02:06:16] "GET / HTTP/1.1" 200 -
 - - [02/Apr/2025 02:06:35] "GET / HTTP/1.1" 200 -
 - - [02/Apr/2025 02:06:46] "GET / HTTP/1.1" 200 -
 - - [02/Apr/2025 02:07:05] "GET / HTTP/1.1" 200 -
 - - [02/Apr/2025 02:07:16] "GET / HTTP/1.1" 200 -
 - - [02/Apr/2025 02:07:35] "GET / HTTP/1.1" 200 -
 - - [02/Apr/2025 02:07:46] "GET / HTTP/1.1" 200 -
```

```
[redacted] - - [02/Apr/2025 02:03:15] "GET / HTTP/1.1" 200 -
 - - [02/Apr/2025 02:03:35] "GET / HTTP/1.1" 200 -
 - - [02/Apr/2025 02:03:45] "GET / HTTP/1.1" 200 -
 - - [02/Apr/2025 02:04:05] "GET / HTTP/1.1" 200 -
 - - [02/Apr/2025 02:04:16] "GET / HTTP/1.1" 200 -
 - - [02/Apr/2025 02:04:35] "GET / HTTP/1.1" 200 -
     [02/Apr/2025 02:04:46] "GET / HTTP/1.1" 200 -
       - - [02/Apr/2025 02:04:49] code 404, message File not found
       - - [02/Apr/2025 02:04:49] "GET /my_custom_page.html HTTP/1.1" 404

       - - [02/Apr/2025 02:04:52] "GET /landing.html HTTP/1.1" 200 -
 - - [02/Apr/2025 02:05:05] "GET / HTTP/1.1" 200 -
 - - [02/Apr/2025 02:05:16] "GET / HTTP/1.1" 200 -
 - - [02/Apr/2025 02:05:35] "GET / HTTP/1.1" 200 -
 - - [02/Apr/2025 02:05:46] "GET / HTTP/1.1" 200 -
 - - [02/Apr/2025 02:06:05] "GET / HTTP/1.1" 200 -
 - - [02/Apr/2025 02:06:16] "GET / HTTP/1.1" 200 -
 - - [02/Apr/2025 02:06:35] "GET / HTTP/1.1" 200 -
 - - [02/Apr/2025 02:06:46] "GET / HTTP/1.1" 200 -
 - - [02/Apr/2025 02:07:05] "GET / HTTP/1.1" 200 -
 - - [02/Apr/2025 02:07:16] "GET / HTTP/1.1" 200 -
 - - [02/Apr/2025 02:07:35] "GET / HTTP/1.1" 200 -
 - - [02/Apr/2025 02:07:46] "GET / HTTP/1.1" 200 -
```

Recap

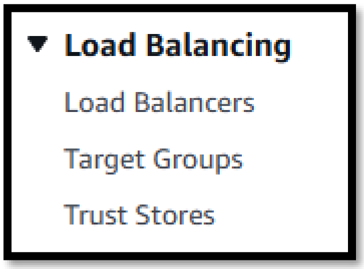
This has been a monster of a lab, but we are not done yet. To move to the final part of this lab you need to ensure the following:

- ✓ You have two EC2 instances running python3 and simple http server
- ✓ Both EC2 instances are successfully serving http server pages

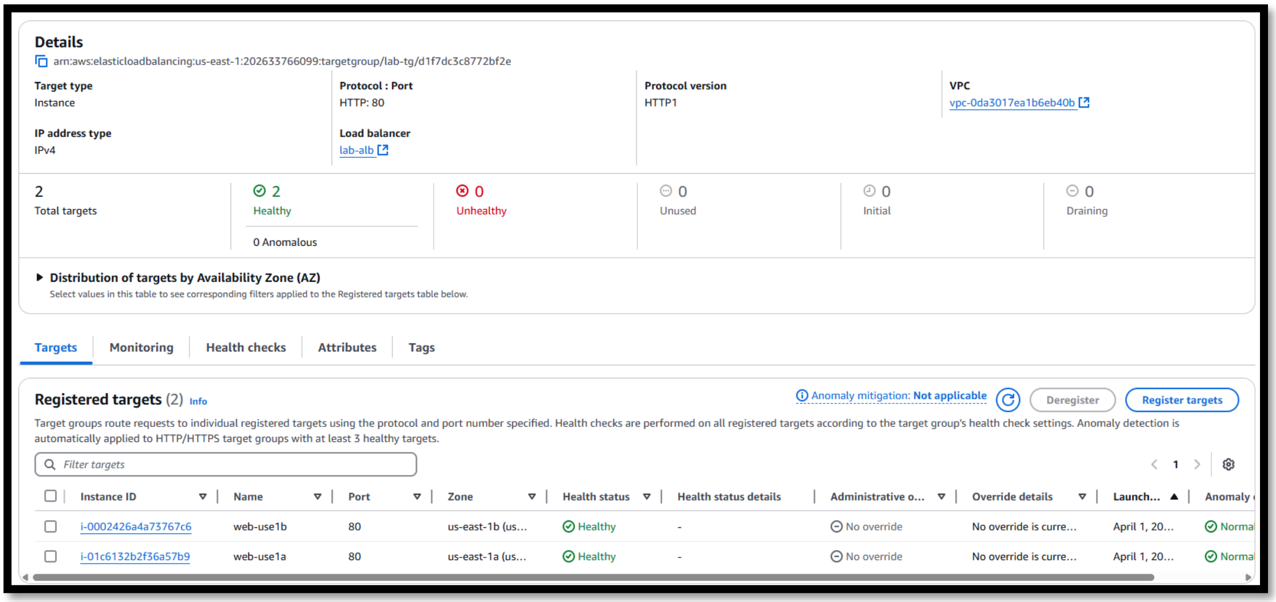
Finalize and Test Deployment

We've completed almost every single piece of this lab, and it has been done in small phases. We initially created our load balancer and target pool but left it empty. Now that we have confirmed our web servers are behaving accordingly, we can go back and finish the load balancer config.

1) First, we need to update our target group with our new healthy web servers. We can do this by going to EC2 on the AWS console, then Load Balancing, and finally Target Groups



2) Add our new EC2 instances as targets then register them at the bottom of the page. Once completed and after a few minutes, you should see something like the output below:



Details

arn:aws:elasticloadbalancing:us-east-1:202633766099:targetgroup/lab-tg/d1f7dc3c8772bf2e

Target type
Instance

Protocol : Port
HTTP: 80

Protocol version
HTTP1

VPC
vpc-0da3017ea1b6eb40b

IP address type
IPv4

Load balancer
lab-alb

2 Total targets

✓ 2 Healthy

0 Anomalous

⊗ 0 Unhealthy

0 Unused

0 Initial

0 Draining

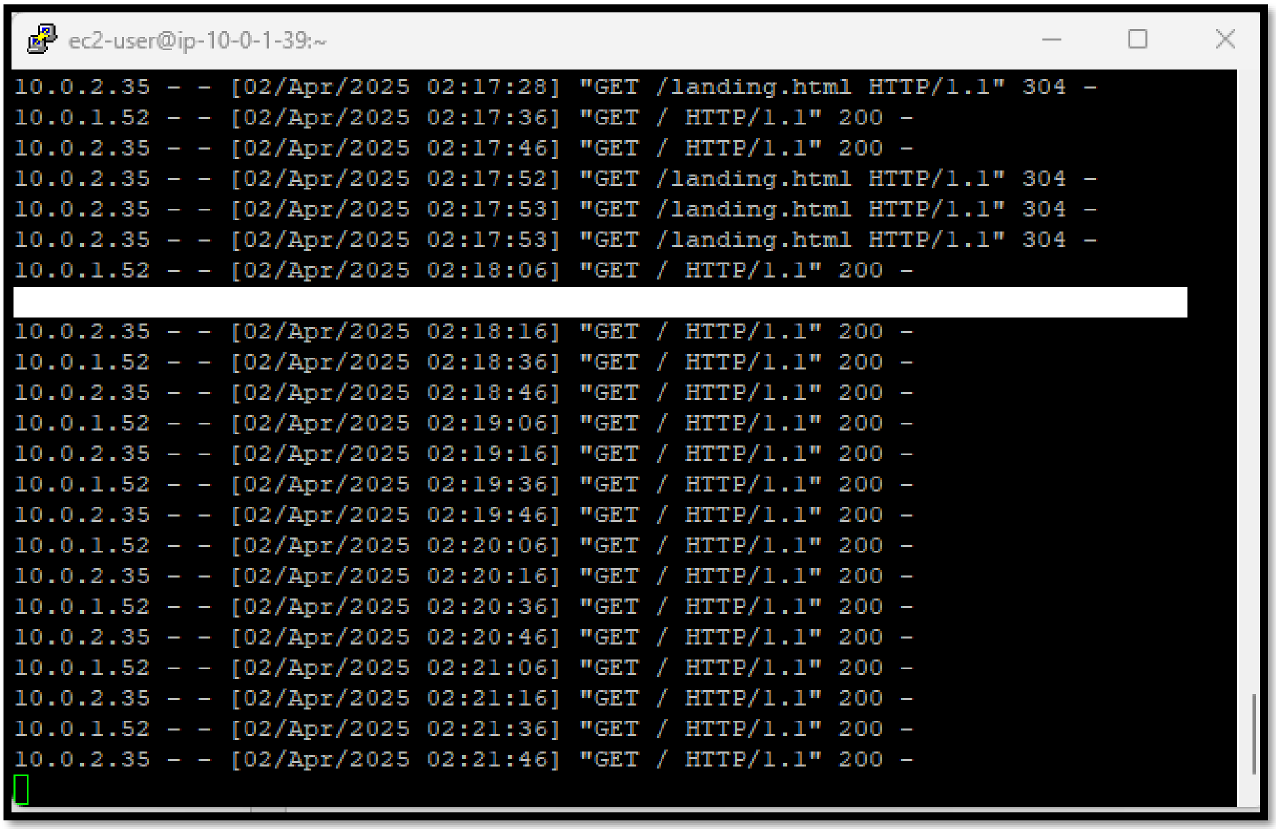
▶ **Distribution of targets by Availability Zone (AZ)**
Select values in this table to see corresponding filters applied to the Registered targets table below.

Registered targets (2) Info

Target groups route requests to individual registered targets using the protocol and port number specified. Health checks are performed on all registered targets according to the target group's health check settings. Anomaly detection is automatically applied to HTTP/HTTPS target groups with at least 3 healthy targets.

□	Instance ID	Name	Port	Zone	Health status	Health status details	Administrative o...	Override details	Launch... ▲	Anomaly
□	i-0002426a4a73767c6	web-use1b	80	us-east-1b (us...	✓ Healthy	-	No override	No override is curre...	April 1, 20...	✓ Normal
□	i-01c6132b2f36a57b9	web-use1a	80	us-east-1a (us...	✓ Healthy	-	No override	No override is curre...	April 1, 20...	✓ Normal

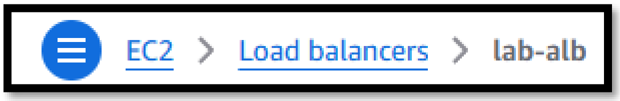
3) Open your PuTTY connection and notice you will start seeing lots of HTTP GET requests from an internal IP. This is the AWS load balancer checking the health of your servers now.



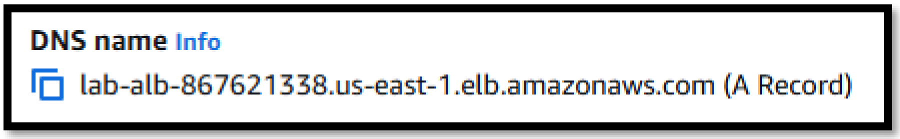
```
10.0.2.35 - - [02/Apr/2025 02:17:28] "GET /landing.html HTTP/1.1" 304 -
10.0.1.52 - - [02/Apr/2025 02:17:36] "GET / HTTP/1.1" 200 -
10.0.2.35 - - [02/Apr/2025 02:17:46] "GET / HTTP/1.1" 200 -
10.0.2.35 - - [02/Apr/2025 02:17:52] "GET /landing.html HTTP/1.1" 304 -
10.0.2.35 - - [02/Apr/2025 02:17:53] "GET /landing.html HTTP/1.1" 304 -
10.0.2.35 - - [02/Apr/2025 02:17:53] "GET /landing.html HTTP/1.1" 304 -
10.0.1.52 - - [02/Apr/2025 02:18:06] "GET / HTTP/1.1" 200 -

10.0.2.35 - - [02/Apr/2025 02:18:16] "GET / HTTP/1.1" 200 -
10.0.1.52 - - [02/Apr/2025 02:18:36] "GET / HTTP/1.1" 200 -
10.0.2.35 - - [02/Apr/2025 02:18:46] "GET / HTTP/1.1" 200 -
10.0.1.52 - - [02/Apr/2025 02:19:06] "GET / HTTP/1.1" 200 -
10.0.2.35 - - [02/Apr/2025 02:19:16] "GET / HTTP/1.1" 200 -
10.0.1.52 - - [02/Apr/2025 02:19:36] "GET / HTTP/1.1" 200 -
10.0.2.35 - - [02/Apr/2025 02:19:46] "GET / HTTP/1.1" 200 -
10.0.1.52 - - [02/Apr/2025 02:20:06] "GET / HTTP/1.1" 200 -
10.0.2.35 - - [02/Apr/2025 02:20:16] "GET / HTTP/1.1" 200 -
10.0.1.52 - - [02/Apr/2025 02:20:36] "GET / HTTP/1.1" 200 -
10.0.2.35 - - [02/Apr/2025 02:20:46] "GET / HTTP/1.1" 200 -
10.0.1.52 - - [02/Apr/2025 02:21:06] "GET / HTTP/1.1" 200 -
10.0.2.35 - - [02/Apr/2025 02:21:16] "GET / HTTP/1.1" 200 -
10.0.1.52 - - [02/Apr/2025 02:21:36] "GET / HTTP/1.1" 200 -
10.0.2.35 - - [02/Apr/2025 02:21:46] "GET / HTTP/1.1" 200 -
```

4) Next go to the Load Balancers tab in the AWS Console under EC2 and select your ALB.



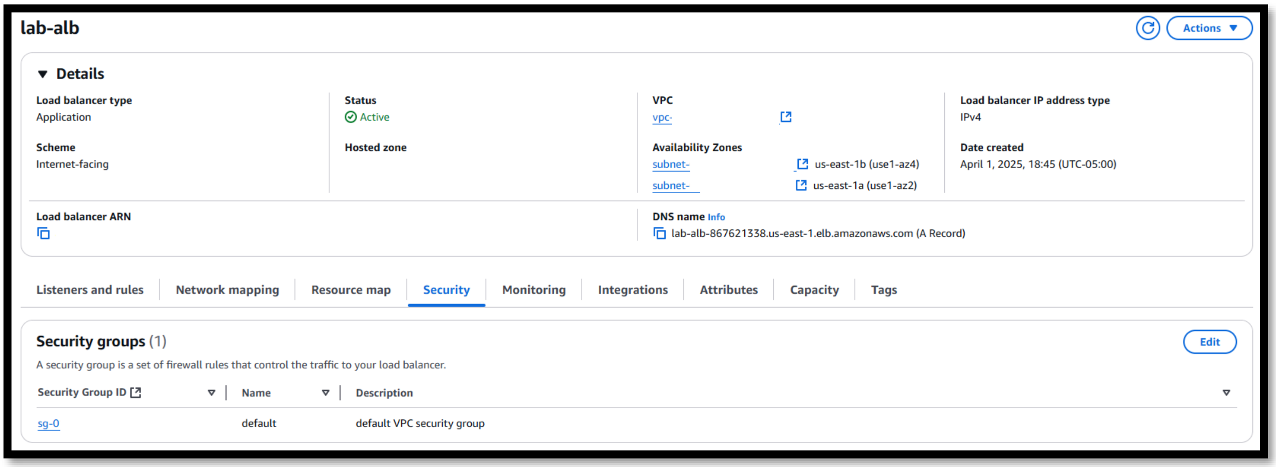
5) Find its DNS name, it should look like mine below. You will want to copy this and paste it into your browser momentarily. This is your ALB frontend DNS name. We can use it to connect to our backend web servers.



DNS name Info

lab-alb-867621338.us-east-1.elb.amazonaws.com (A Record)

6) Before you do this, you need to allow TCP 80 traffic to your ALB from anywhere in the world. In your ALB configuration click on security and then click on the security group.



lab-alb

▼ Details

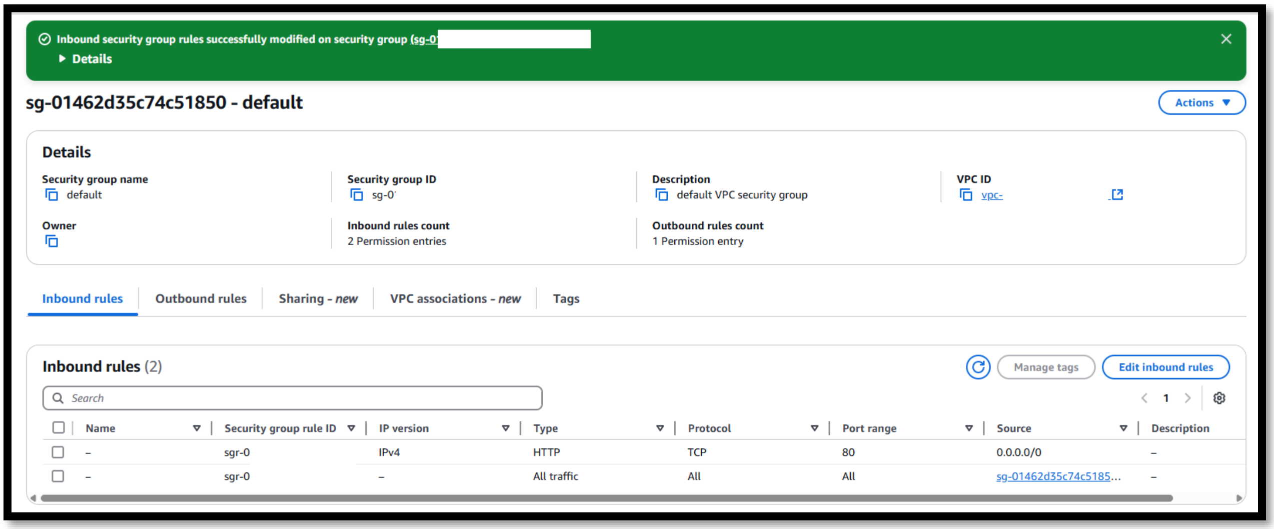
Load balancer type Application	**Status** ✓ Active	**VPC** vpc-	**Load balancer IP address type** IPv4
Scheme Internet-facing	**Hosted zone**	**Availability Zones** subnet- us-east-1b (use1-az4) subnet- us-east-1a (use1-az2)	**Date created** April 1, 2025, 18:45 (UTC-05:00)
Load balancer ARN		**DNS name** Info lab-alb-867621338.us-east-1.elb.amazonaws.com (A Record)	

Security groups (1)

A security group is a set of firewall rules that control the traffic to your load balancer.

Security Group ID	Name	Description
sg-0	default	default VPC security group

7) Create one last security group policy like the one seen below.



Inbound security group rules successfully modified on security group (sg-0[redacted]

▶ Details

sg-01462d35c74c51850 - default

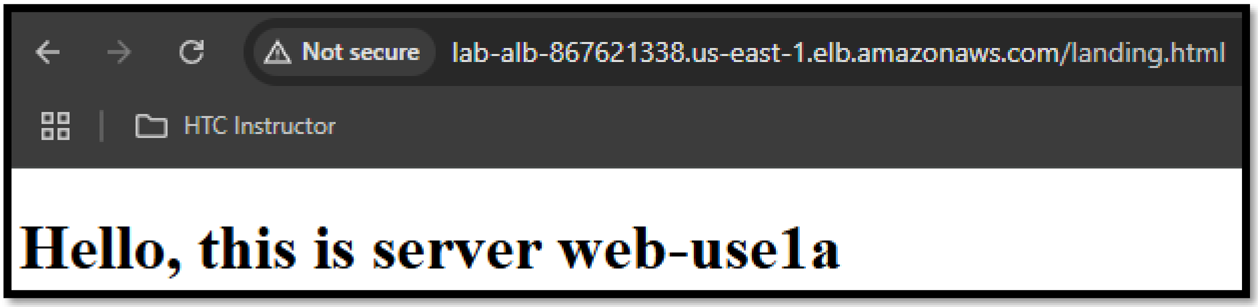
Details

Security group name default	**Security group ID** sg-0'	**Description** default VPC security group	**VPC ID** vpc-
Owner	**Inbound rules count** 2 Permission entries	**Outbound rules count** 1 Permission entry	

Inbound rules (2)

□	Name	Security group rule ID	IP version	Type	Protocol	Port range	Source	Description
□	–	sgr-0	IPv4	HTTP	TCP	80	0.0.0.0/0	–
□	–	sgr-0	–	All traffic	All	All	sg-01462d35c74c5185…	–

8) Finally, we can test our full deployment with the DNS name of the ALB we grabbed from step 5. Ensure you put /landing.html behind the name and it will look like mine below.



Hello, this is server web-use1a

Summary

Congratulations on finishing this lab! This lab is by no means short or easy. Keep in mind this lab is meant to be a large introduction to utilizing AWS and building real-world based solutions. If you find this lab to be confusing or not making any sense, you may want to head over to the references on the next page and check out some of those links. If you have additional questions or have found errors, typos, and would like to contact me, please do so with the email address below:

taylor.kerber@hennepintech.edu

References

URL:	Description:
What is an Application Load Balancer?	An in-depth AWS technical document to explain Application Load Balancers.
Python SimpleHTTPServer	A quick tutorial of Python Simple HTTP Server commands and how to use it.
Connect to your Linux instance using PuTTY	A quick guide from AWS on how to connect to a Linux EC2 intance.
AWS Free Tier	An AWS official document for the types of services and limitations of free tier.
Enable internet access for a VPC using an internet gateway	Official AWS documentation on how to configure an AWS internet gateway.

Revisioning

Revision:	Editor:	Description:
1.0	Taylor Kerber	Initial release of lab.