* IPv4 is a 32 bit number organized into 4 octets (8 bits) addressed as x.x.x.x
* IP Address id combination of network id and host id

ip: 192.168.0.118

sm: 255.255.255.0

--------------------

network id (255): 1,2,3 => 192.168.0

host id (0): 4 => 118

size: bits for host id => 1 octet => 8 => 2^8 - 2 => 256-2 = 254

192.168.0.0 (network id)

192.168.0.255 (broadcast)

**Network Principle**

* A Device can communicate directly with other device in same network
* Router is a device which can transfer packets from one network to other

### IP Addressing (Class Based)

* We have classes to define ip ranges
  + Class A (Class A addresses are for networks with large number of total hosts.)
  + Class B (Class B addresses are for medium to large sized networks.)
  + Class C (Class C addresses are used in small local area networks (LANs).)
  + Class D (Class D IP addresses are not allocated to hosts and are used for multicasting.)
  + Class E (Class E IP addresses are not allocated to hosts and are not available for general use. These are reserved for research purposes.)

### CIDR (Classless inter domain routing)

* Quick revision

ip: 192.168.0.118

sm: 255.255.255.0

n (host id) = 1 octet = 8 bits

N (Network id) = 32 - n = 24 bits

* Find subnet mask for a network of 500 devices

500

2^n - 2 ~= 500

2^n ~= 500

n = 9

N = 32 - 9 = 23

SM: 11111111.11111111.11111110.00000000

255.255.254.0

* In this notation ip is expressed as x.x.x.x/N

192.168.0.0/24

N (network id) = 24 bits

n (host id) = 32 - 24 = 8

ip: 192.168.0.x => 192.168.0.0 to 192.168.0.255

SM: 11111111.11111111.11111111.00000000

10.0.0.0/16

N = 16

n = 32 -16 = 16

ip: 10.0.x.x => 10.0.0.0 to 10.0.255.255

SM: 11111111.11111111.00000000.00000000

**Private vs Public Network**

* Private Network: Network which cannot be accesed directly from internet
* Public Network: Network which can be accessed from internet
* Private network cidr ranges
  + 10.0.0.0/8: 10.0.0.0 to 10.255.255.255
  + 172.16.0.0/12: 172.16.0.0 to 172.31.255.255
  + 192.168.0.0/16: 192.168.0.0 to 192.168.255.255
* Each device requires 100 devices (total 4 floors)

each floor ~= 100

2^n ~= 100

n = 7

N = 25

building ~= 4x100 = 400

2^n ~= 400

n = 9

N = 23

10.0.0.0/8

172.16.0.0/12

192.168.0.0/16

172.16.0.0/23

BD SM: 11111111.11111111.11111110.00000000

FL SM: 11111111.11111111.11111111.10000000

----------------------------------------------

x.x0000000

0.00000000 => 172.16.0.0/25

0.10000000 => 172.16.0.128/25

1.00000000 => 172.16.1.0/25

1.10000000 => 172.16.1.128/25

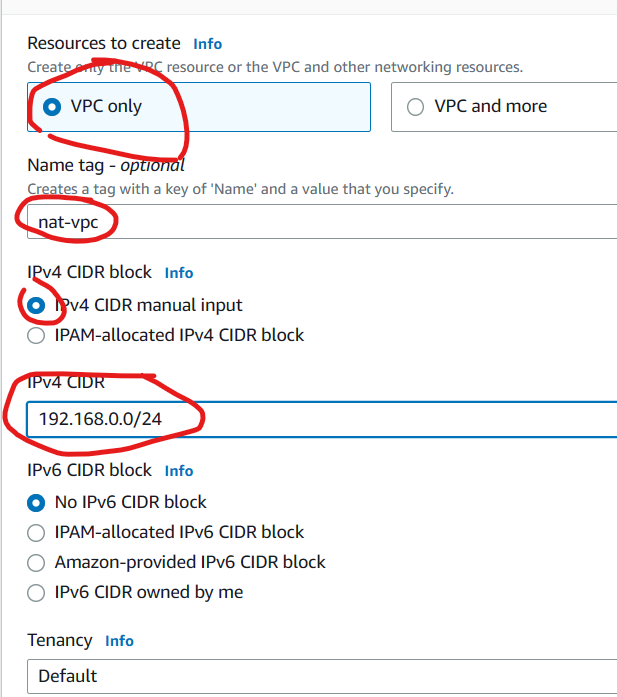
### Network Interface

* A device is connected to the network using Network interface
* The ip that the network interface recieves can be used to access the device/system/server
* A device can have multiple network interfaces
* Network interfaces gets an ip address assigned to it by DHCP (Dynamic Host Configuration Protocol) server
* In your home networks, wifi routers have built in DHCP which assigns the ip
* DHCP can assign dynamic ip or static ip depending on configuration.

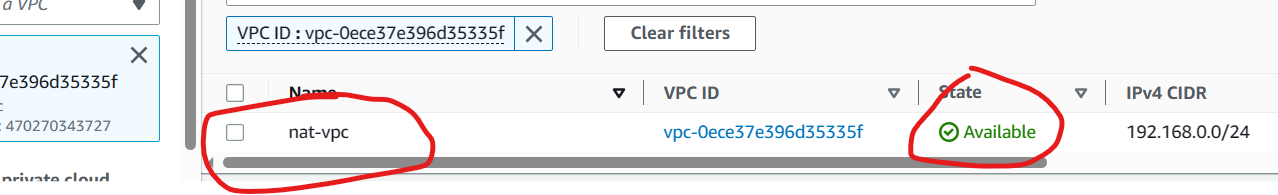
NAT (Network Address Translation) gateway:

* NAT is used to provide internet access to private subnets
* AWS provides NAT by
  + NAT instance
  + NAT Gateway
* Create a vpc with one public and private subnet

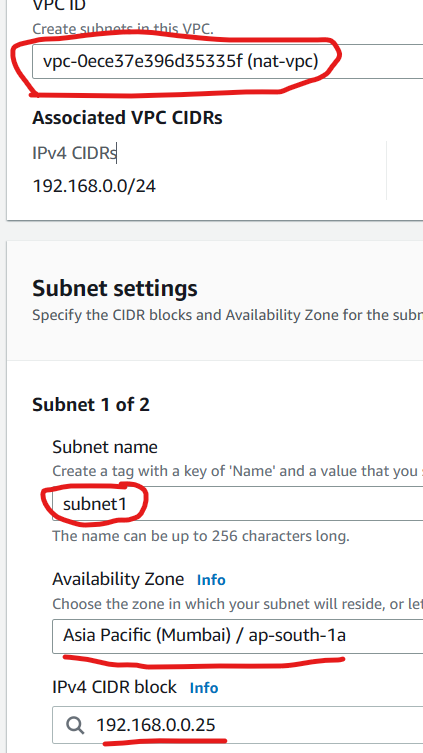
Create a vpc



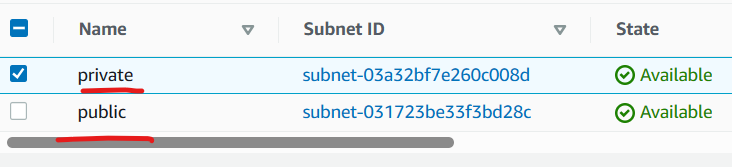
VPC got created



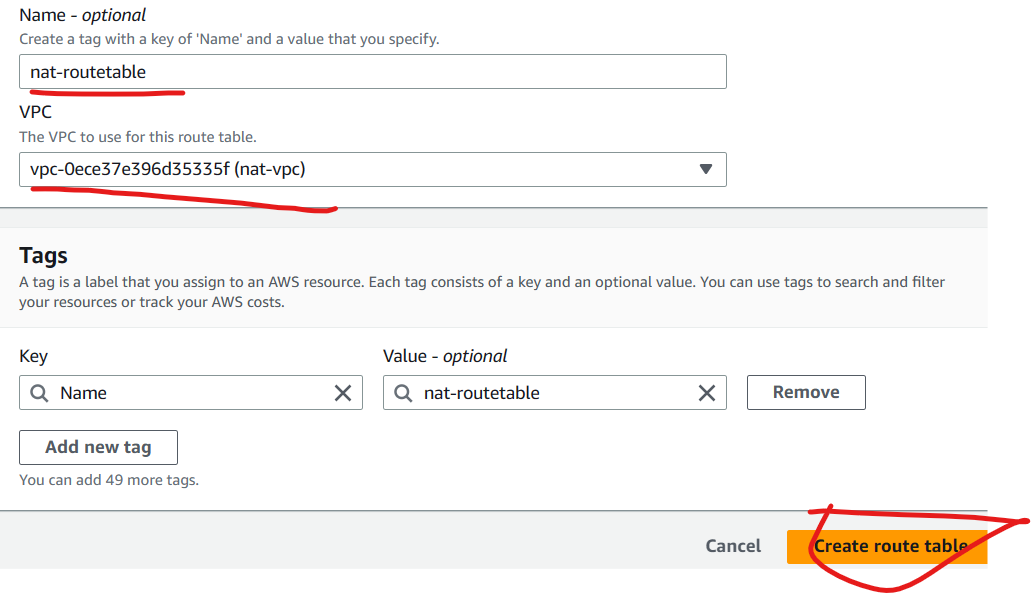
Create subnets with your vpc

changed subnet names

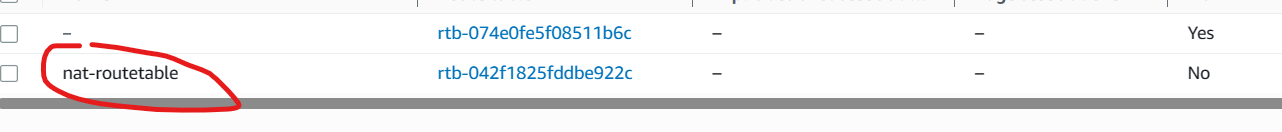
Subnets got created



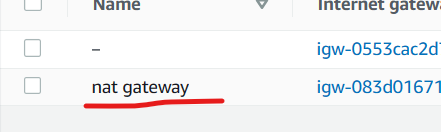
Create a route table



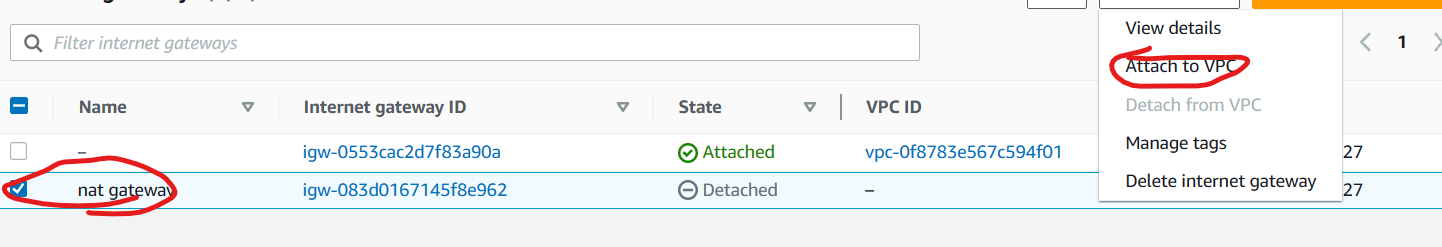
Route table got created



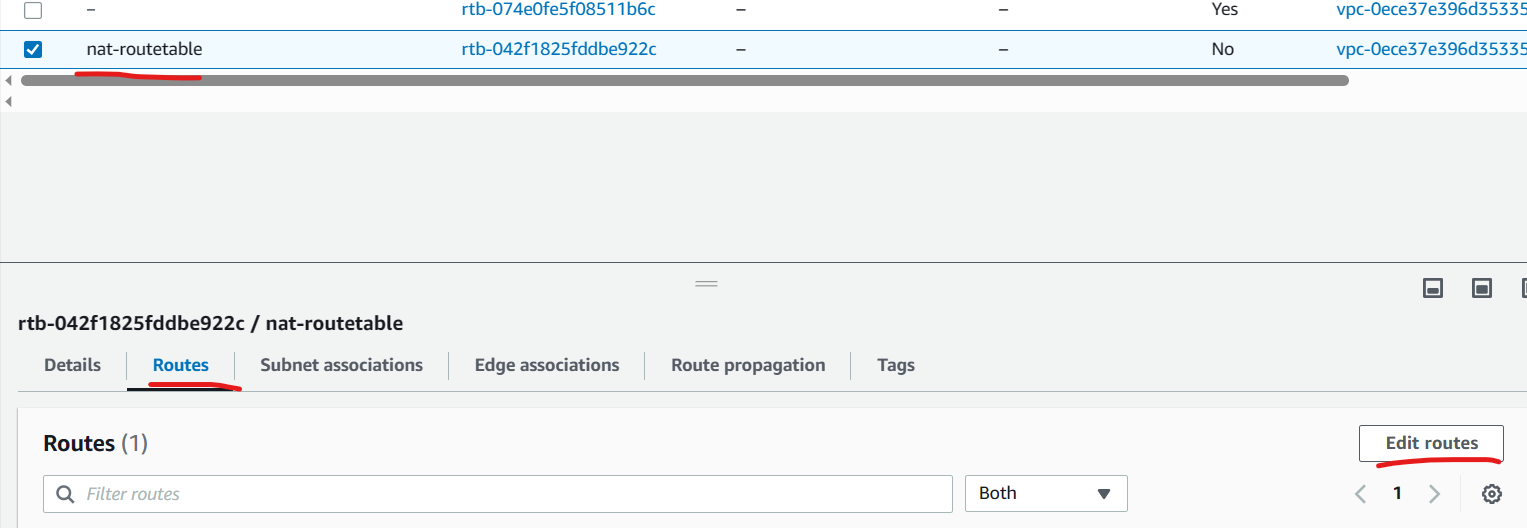
Create a gateway

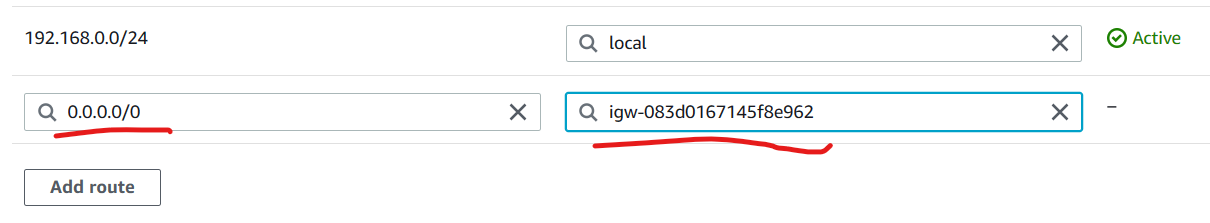


Attach your gateway to your vpc

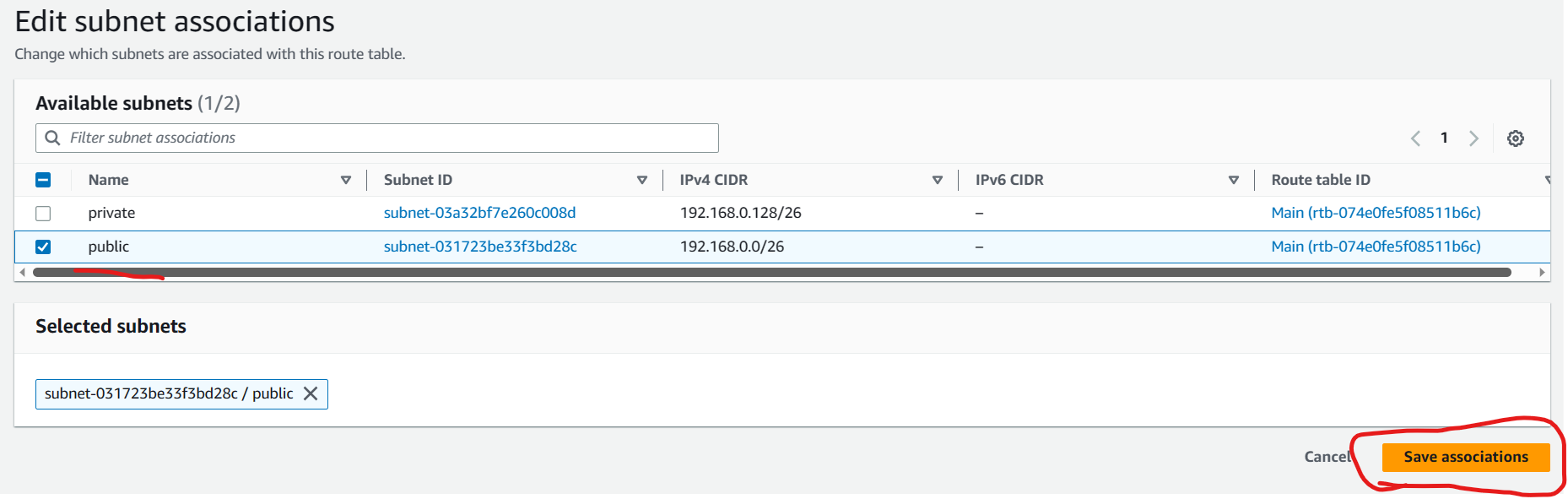


Now edit route table

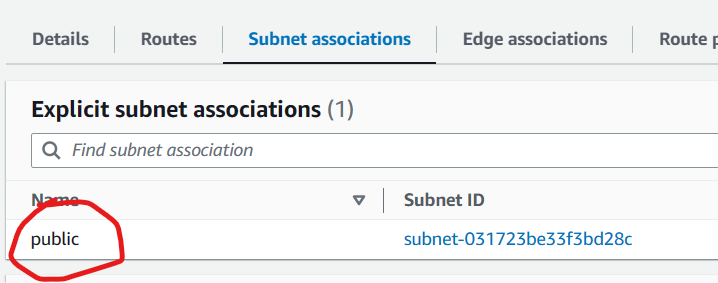




Now associate your subnets to gateway

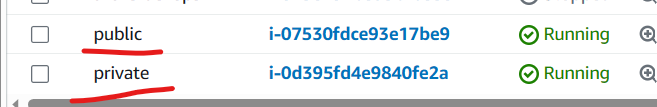


Associated public subnet to route table

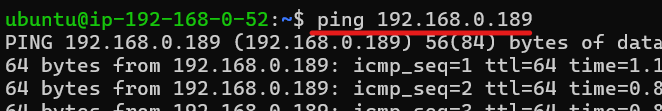


* ensure you have ec2 instances in public and private subnet

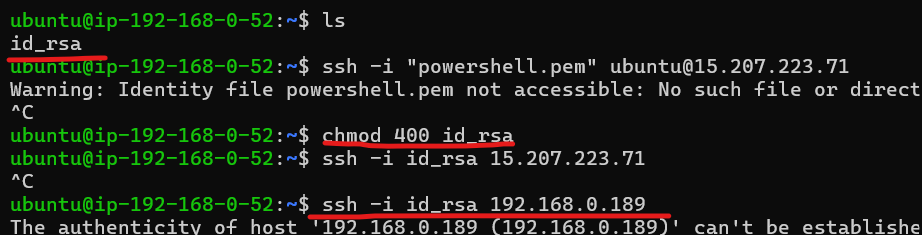
create one ec2 in public subnet and one in private subnet



Login into public machine and ping to private machine with private ip

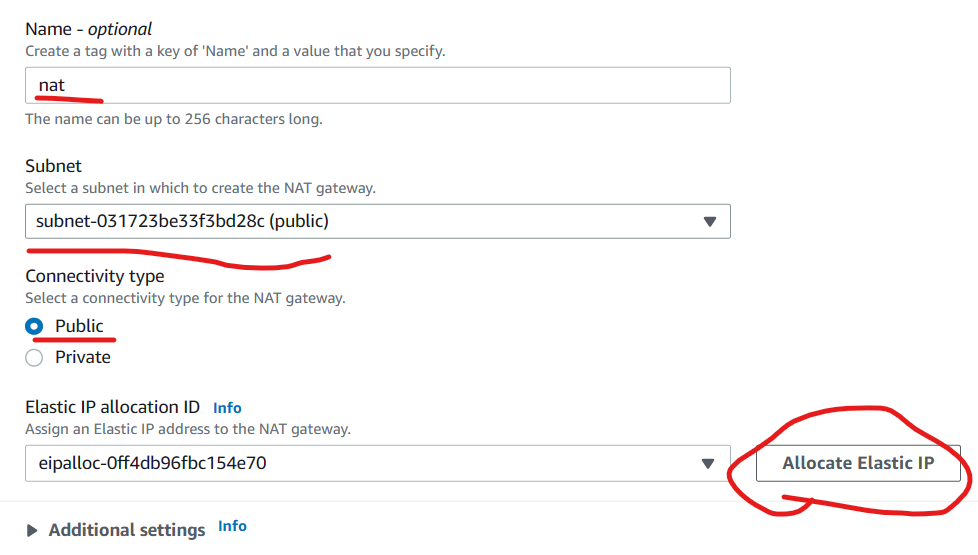


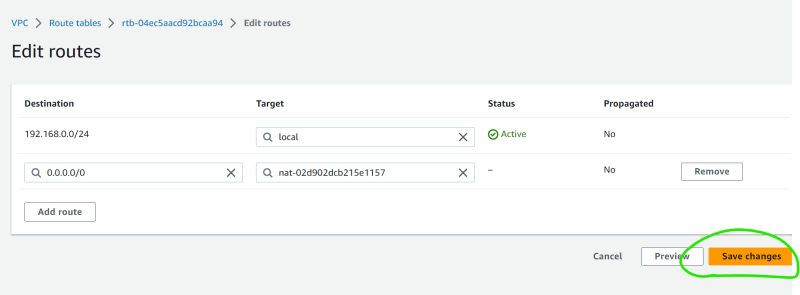
Now copy the id\_rsa key to public machine using sftp and login to private machine using private IP.



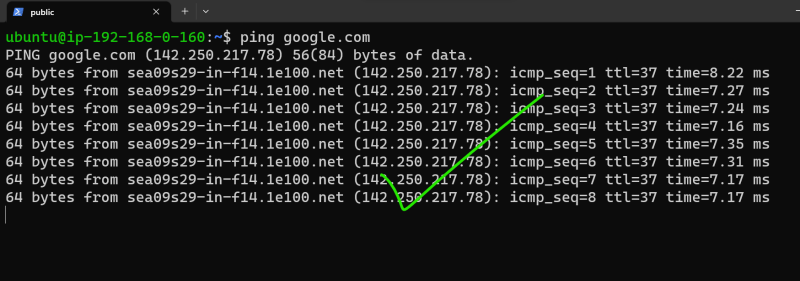
* Now create a NAT gateway in public subnet

Allocate a public ip to the gateway



Now modify private route table to forward packets to NAT Gateway to access internet  


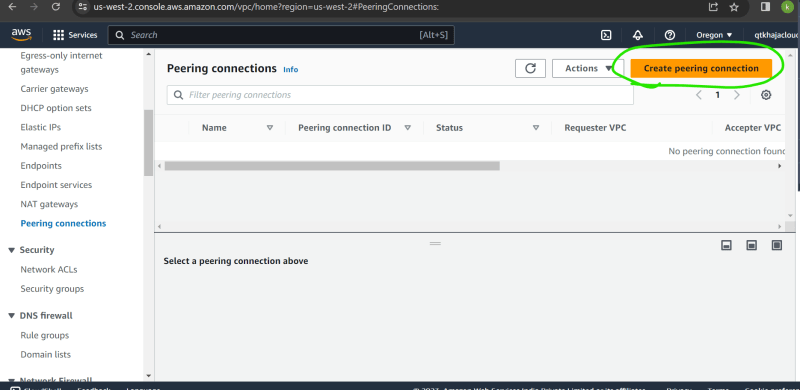
Now from private machine ping google.com

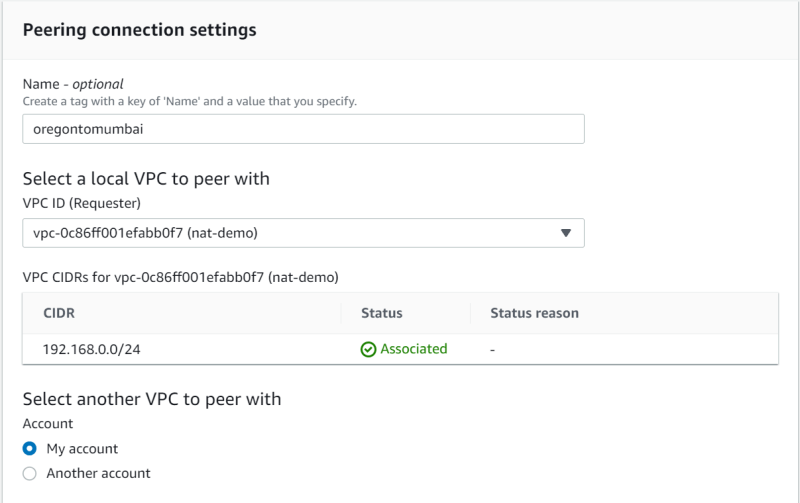
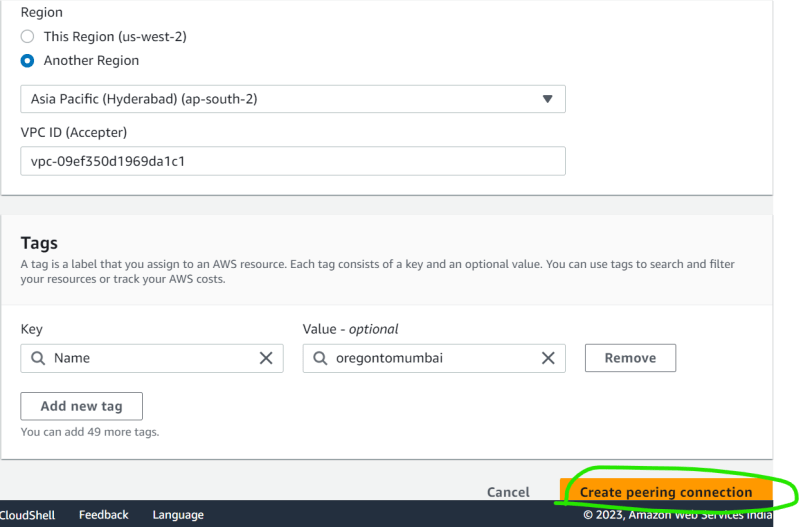


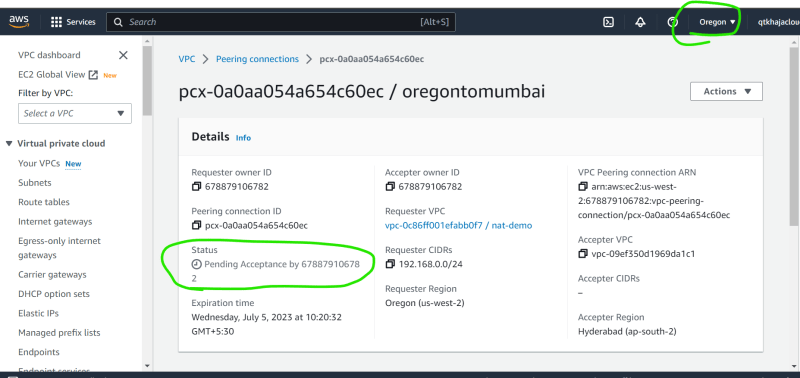
Now delete NAT gateway and release elastic ip.

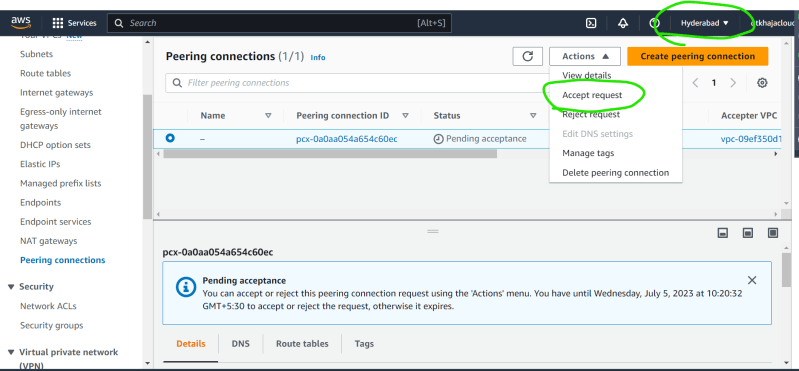
Elastic ip will be deleted only after machines are stopped.

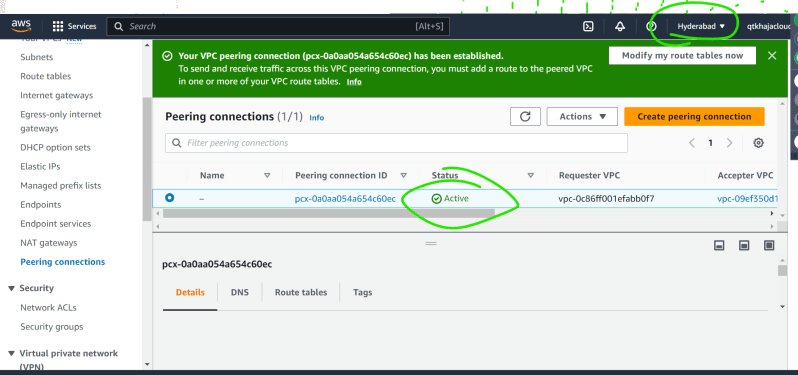
**VPC Peering**

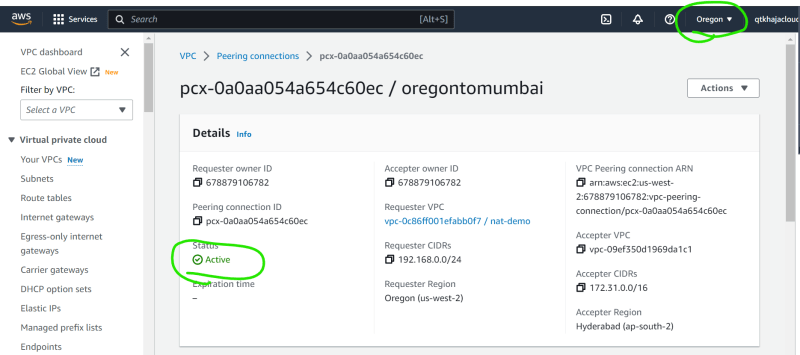
* AWS VPC Peering allows private communication between two vpc’s belonging to any regions or any accounts
* The destination vpc should approve the peering request then in two vpc’s peering connection objects will be created.
* Create a peering connection from one vpc  
  

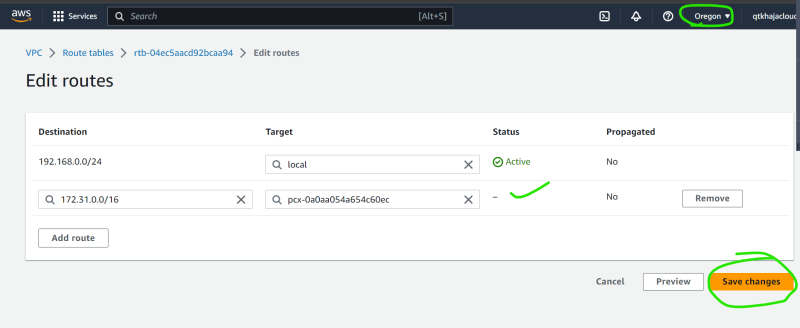
  


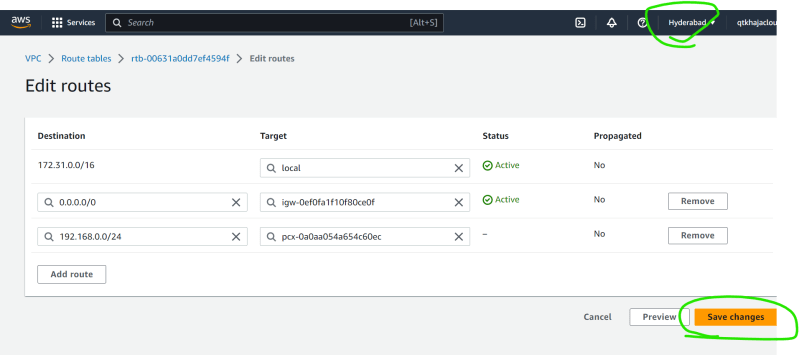




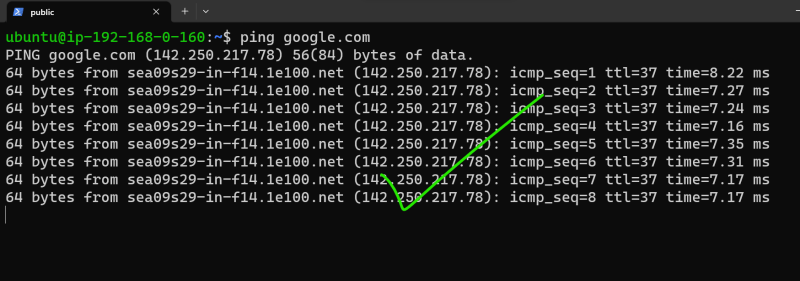


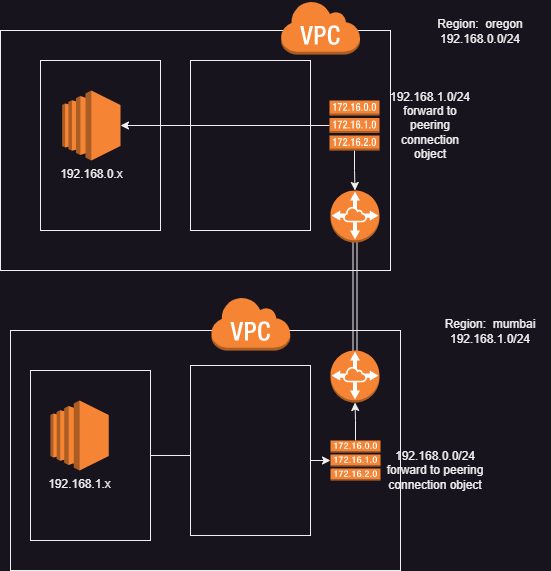


* Now since there is infra to communicate, now modify route tables to forward packets to each other  
  



* Now ping from one ec2 to other using private ip

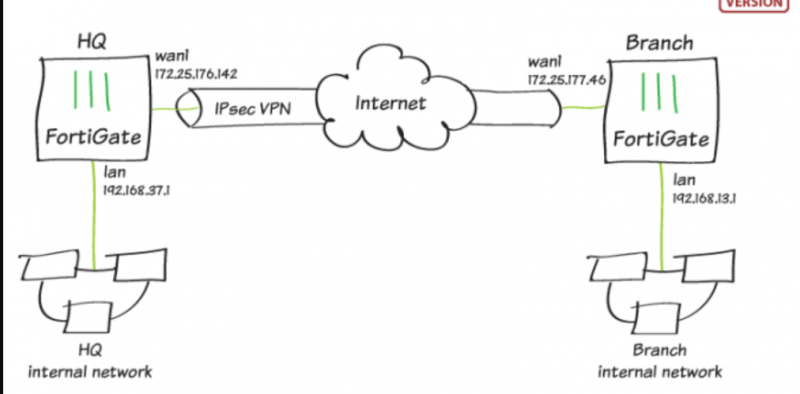


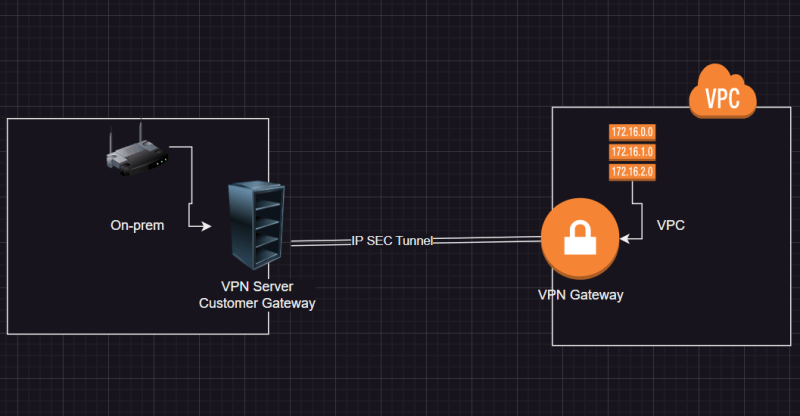
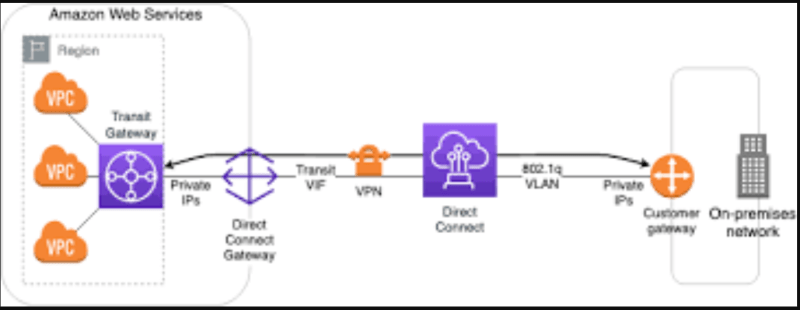
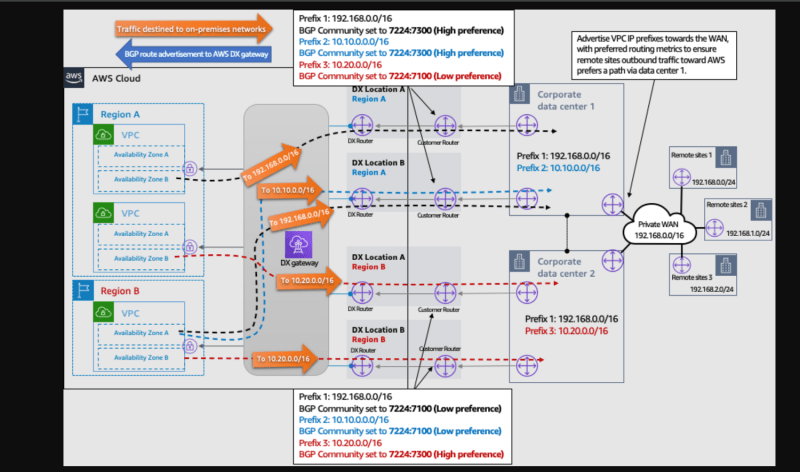
* AWS VPC Peering workflow  
  

VPN (Virtual Private Network):

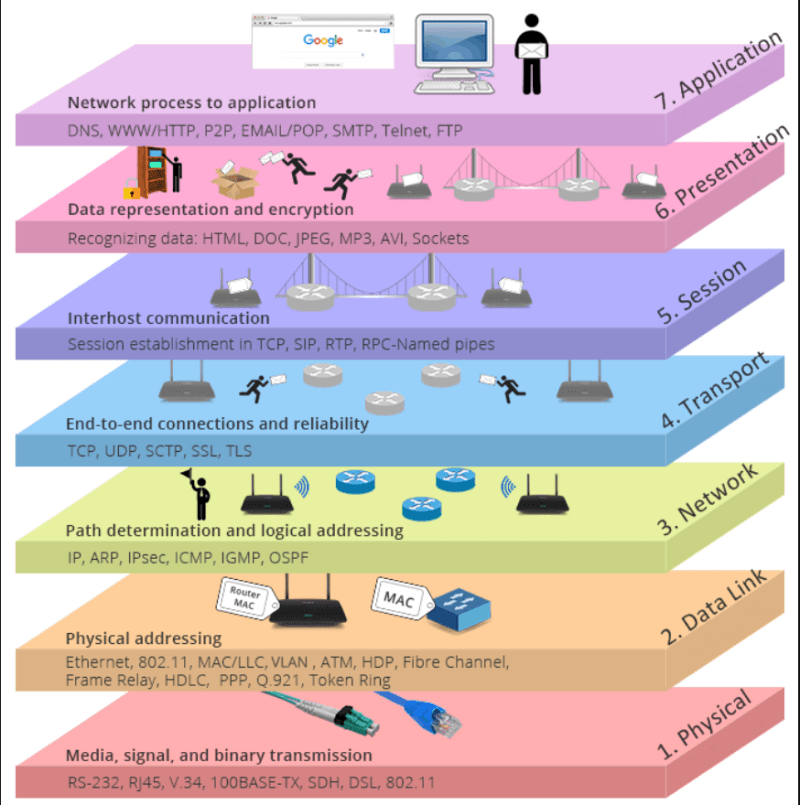
**Site to Site VPN (Virtual Private Networks)**

* Site to Site VPN Overview



* VPN servers at both sides will connected with licence.
* Data from HQ is encoded at VPN server and sent to branch server .
* Now the branch server decodes the data .
* Site to Site VPN in AWS  
  
* Here one side VPN gets replaced with VPN gateway.
* Multi vpc in a region to on-prem  
  
* Transit gateway can communicate with multiple gateways only in one region.
* These can communicate over internet or directly.
* Multi vpc’s to Multi Datacenters on-prem  
  

### OSI Model ( **Open Systems Interconnection** )

* This model has 7 layers  
  

Proxy Server: This server is used to filter out all the outbound network traffic

Reverse Proxy Server: This server is used to forward the incoming traffic to the application

Load Balancers: These servers distribute the traffic equally to servers

### Load Balancer

* to ensure the request is forwarded only to servers which are responding, load balancers perform health checks
* Load Balancer is of two types
  + Layer 4 load balancer:
    - Layer 4 in osi knows about ip, port, protocol
  + Layer 7 Load Balancer
    - Layer 7 in osi knows about http, ip, port, protocol, sessions

### Load Balancers in AWS

* AWS has 3 load balancers
  + Classic Load Balancer: can perform layer 4 and layer 7 lb. This is no longer recommended and is present in aws for backword compatability .
  + Network Load Balancer: This performs layer 4 load Balancing
  + Application Load Balancer: This performs layer 7 load Balancing .
* Create one ec2 instance for test
* Create two instances with names web1 and web2 public ip disable , provide the below commands in advance options path during creation.

#!/bin/bash

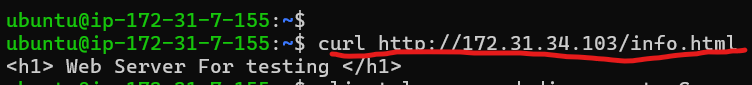
apt update

apt install apache2 -y

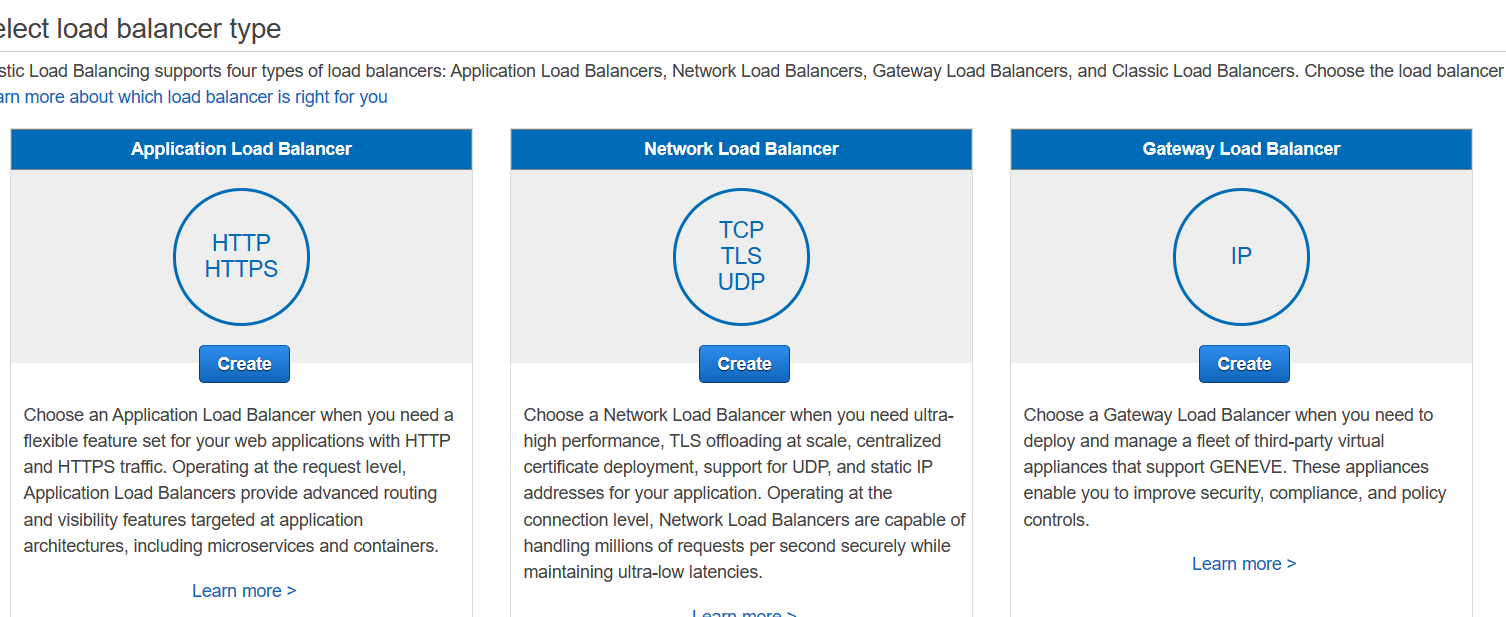
echo "<h1> Web Server For testing </h1>" > /var/www/html/info.html

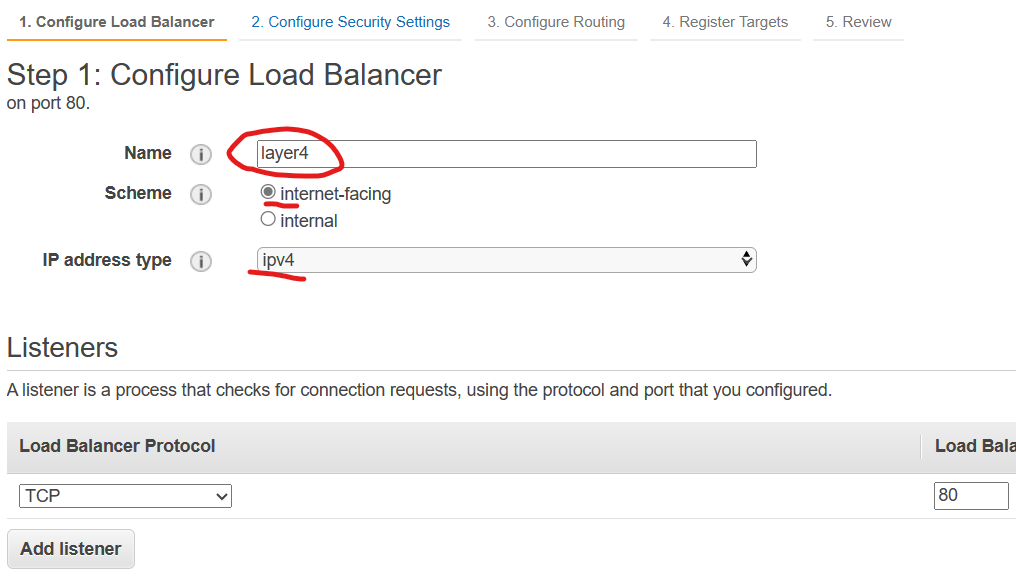
Now from test machine ping to the web1 machine using private ip

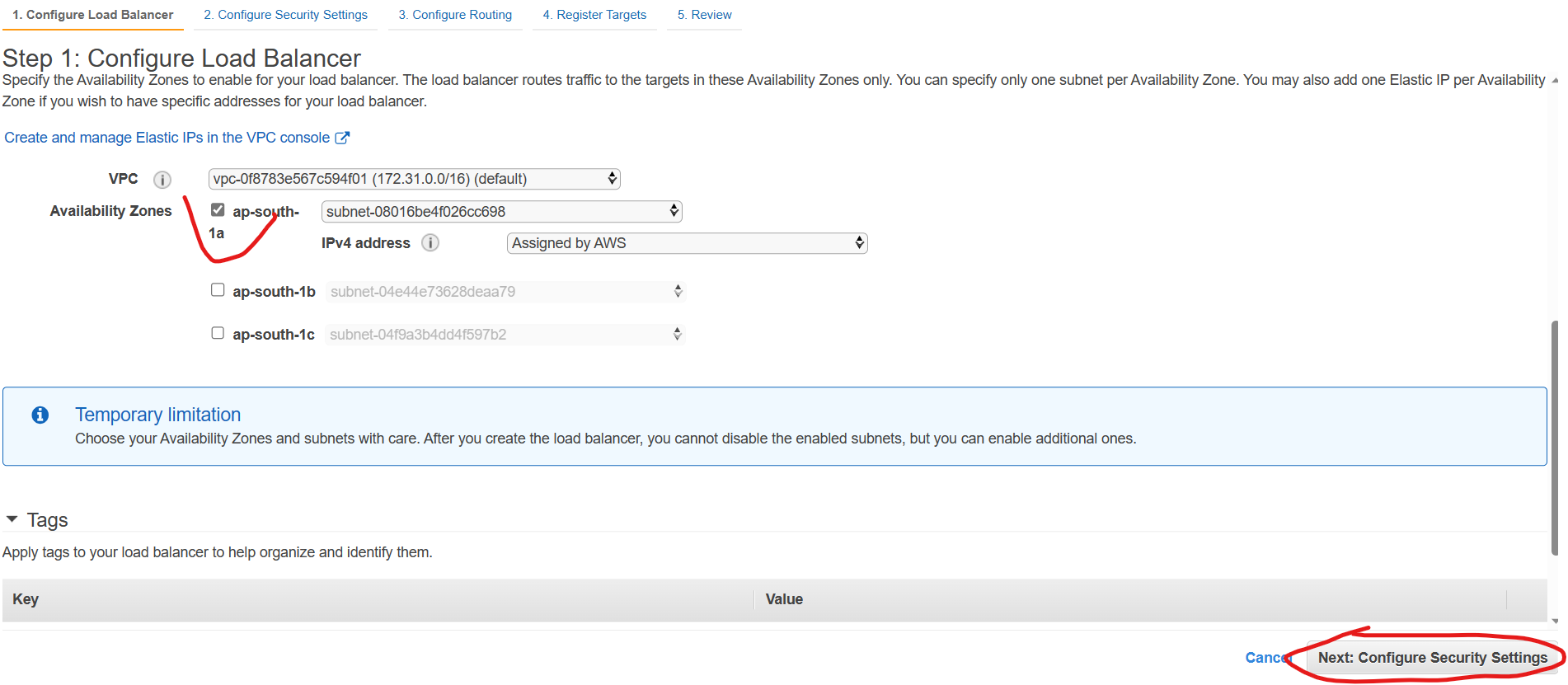
Curl http://<private-ip>/info.html

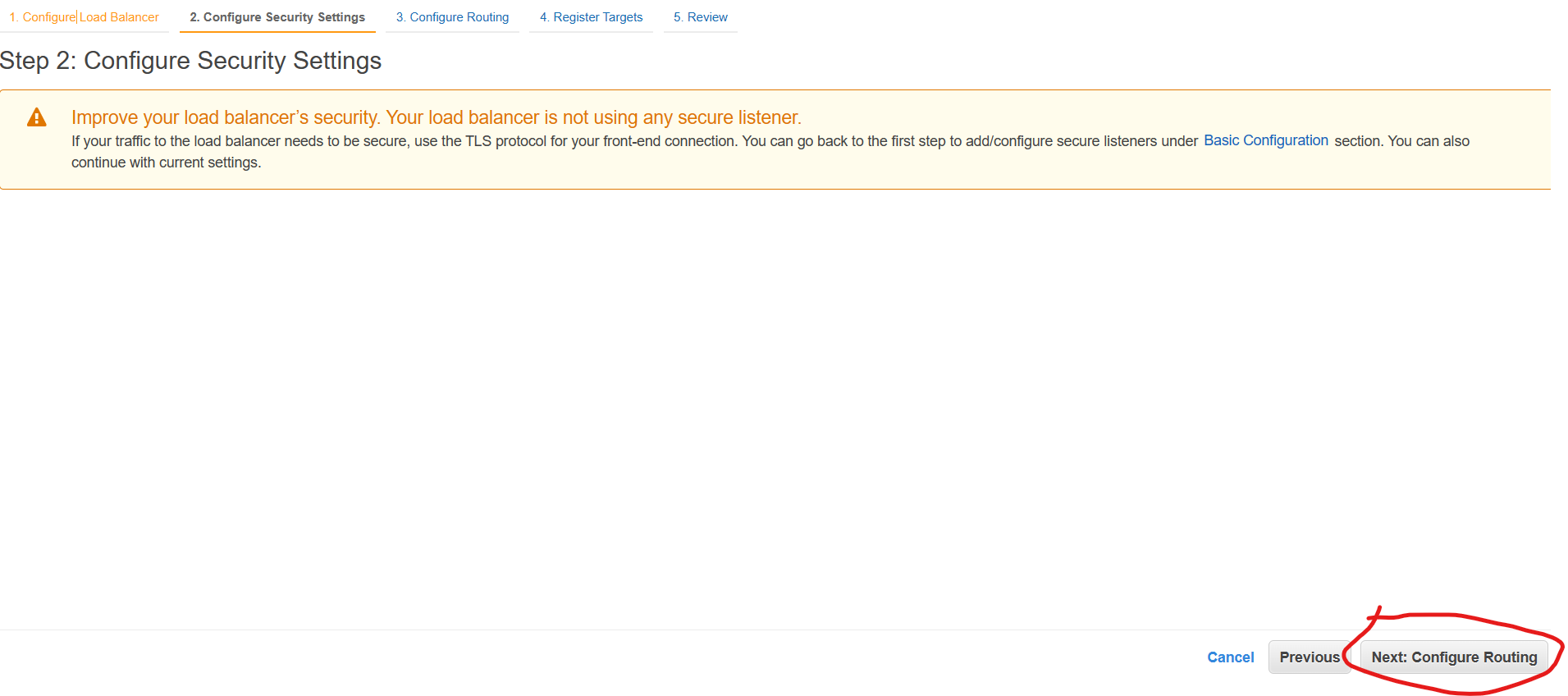


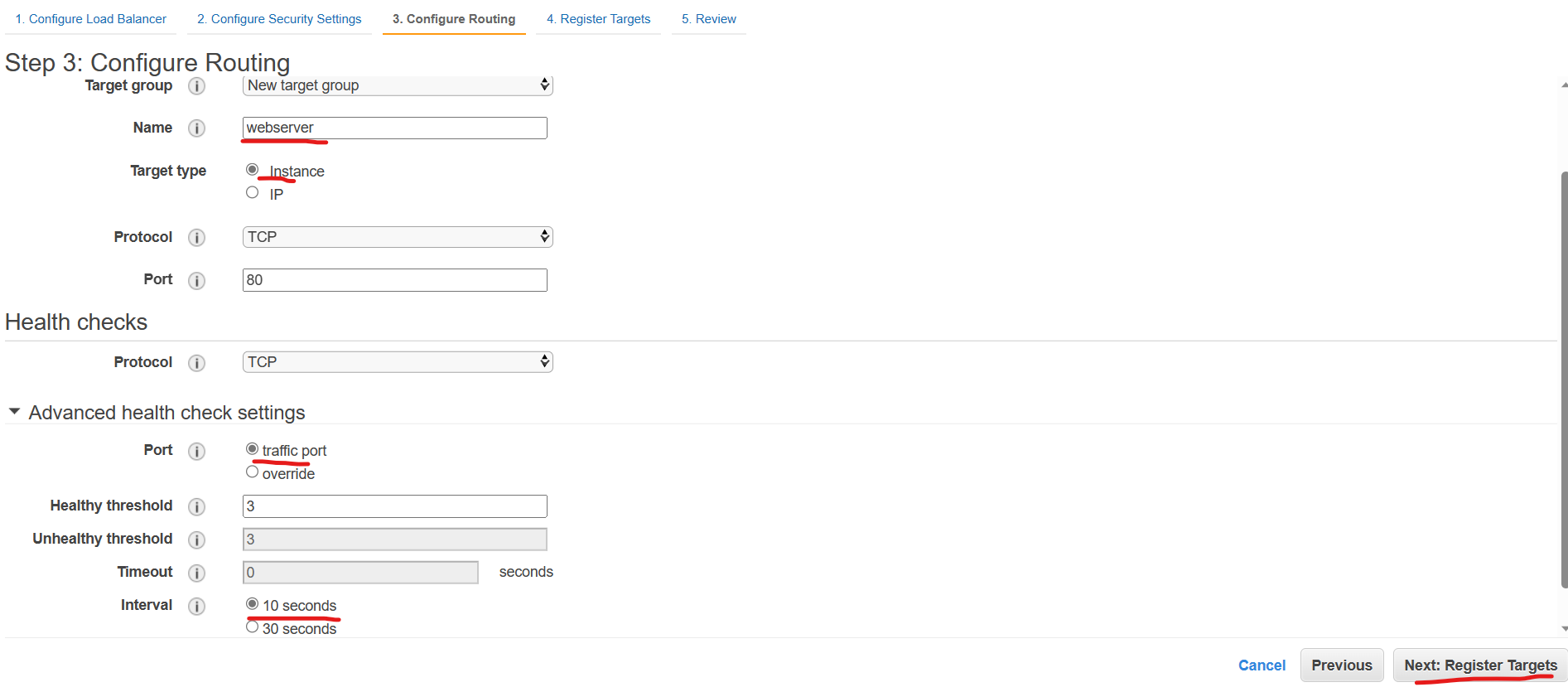
Create a load balancer

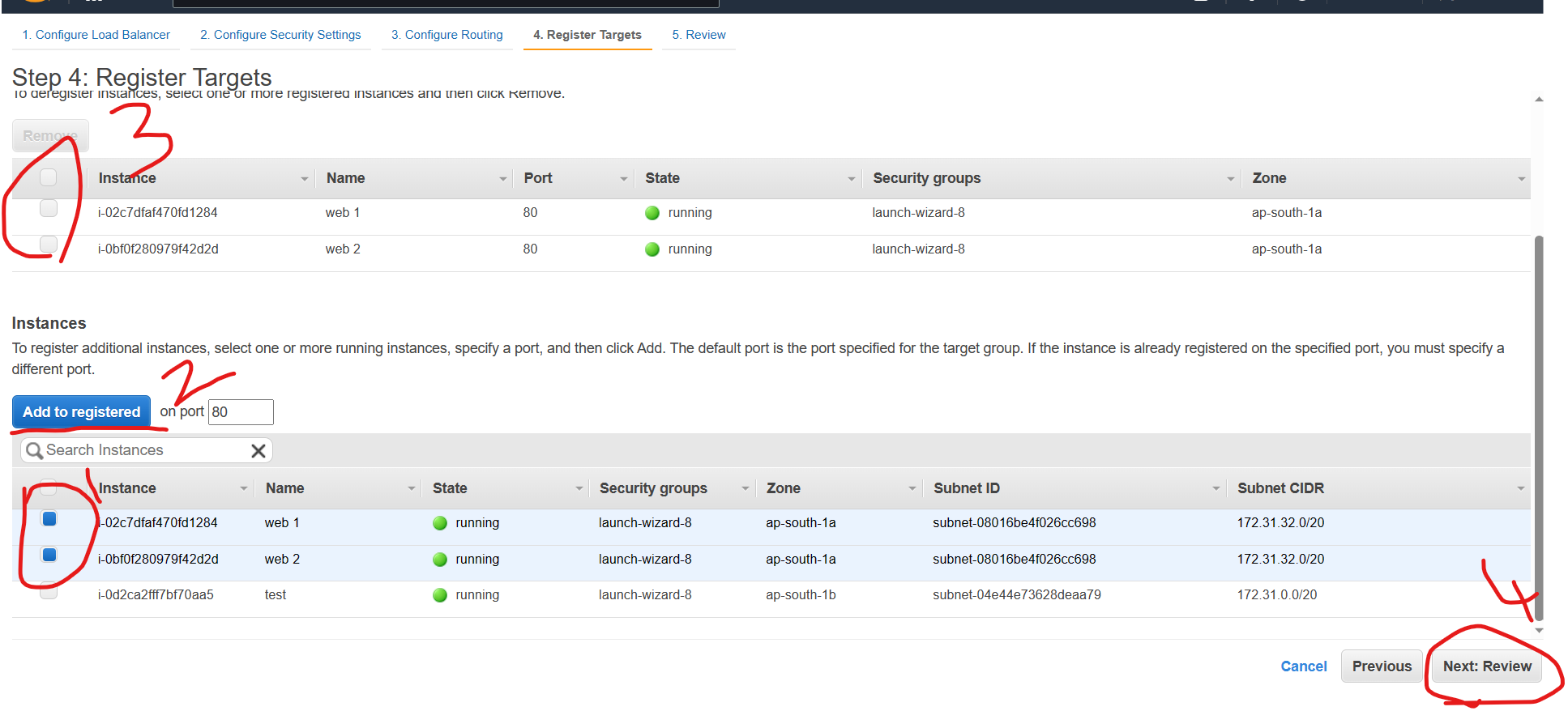
Select network load balancer (layer 4)



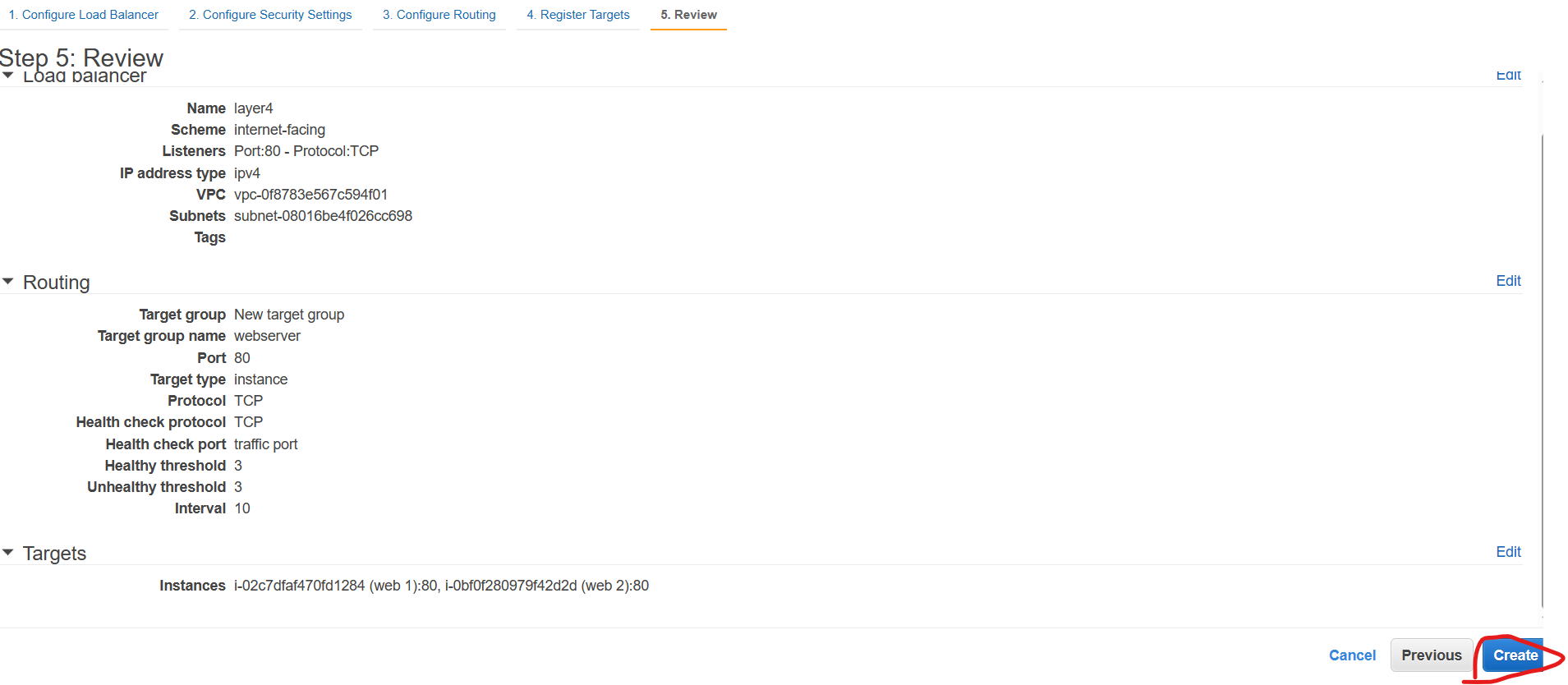


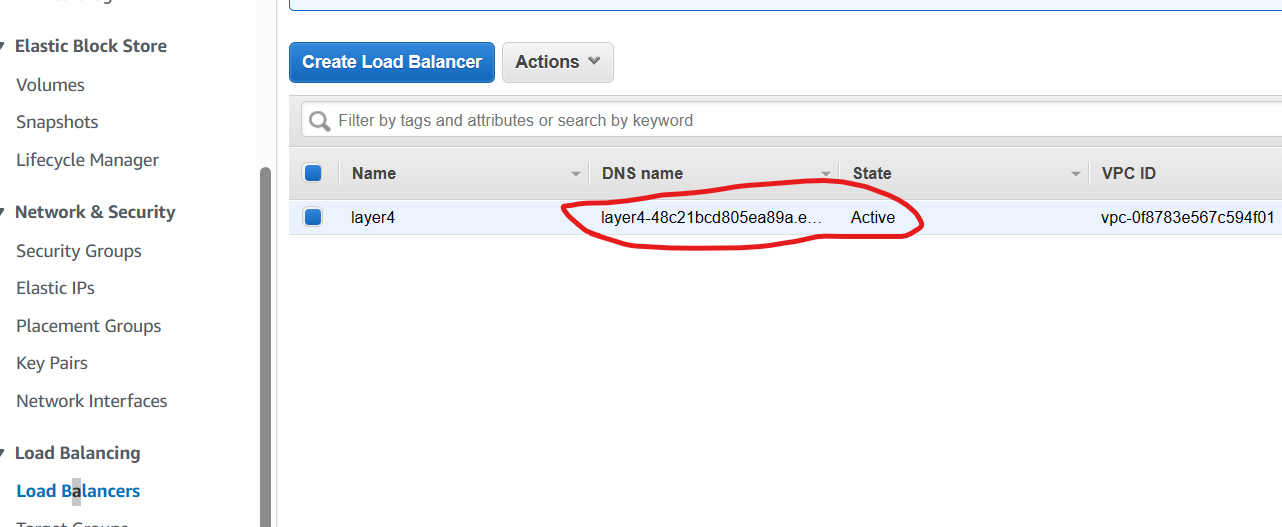




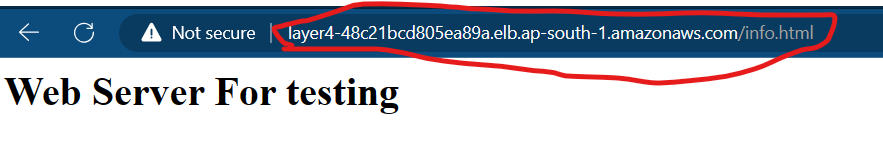


Select the instances and add to registered





Now access the server with DNS name



Application Load Balancer ( layer 7)

