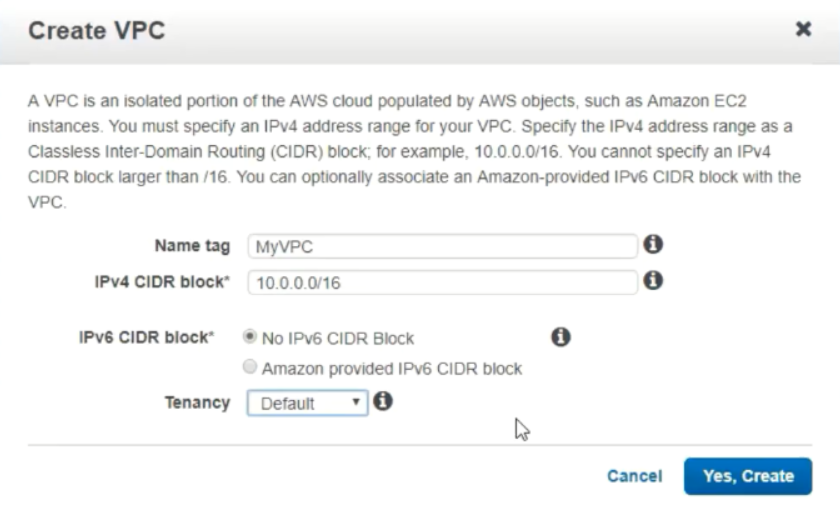
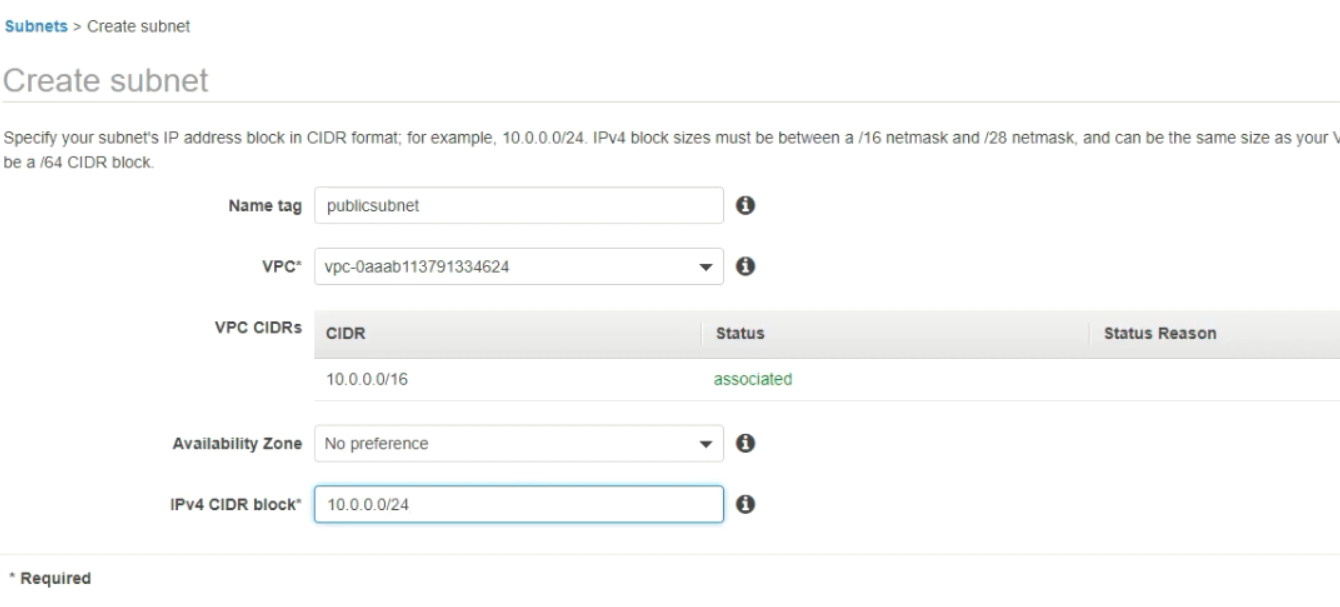
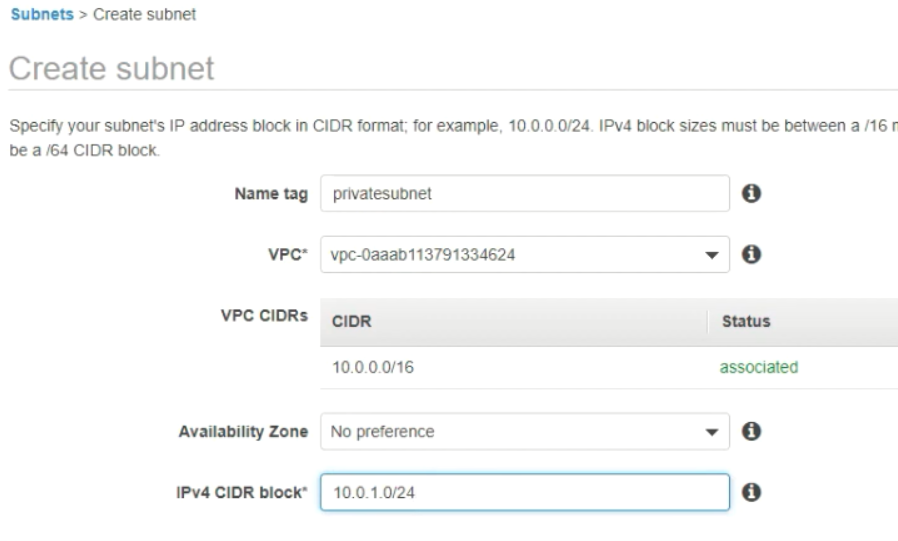
create vpc 1 -> 10.10.0.0/16



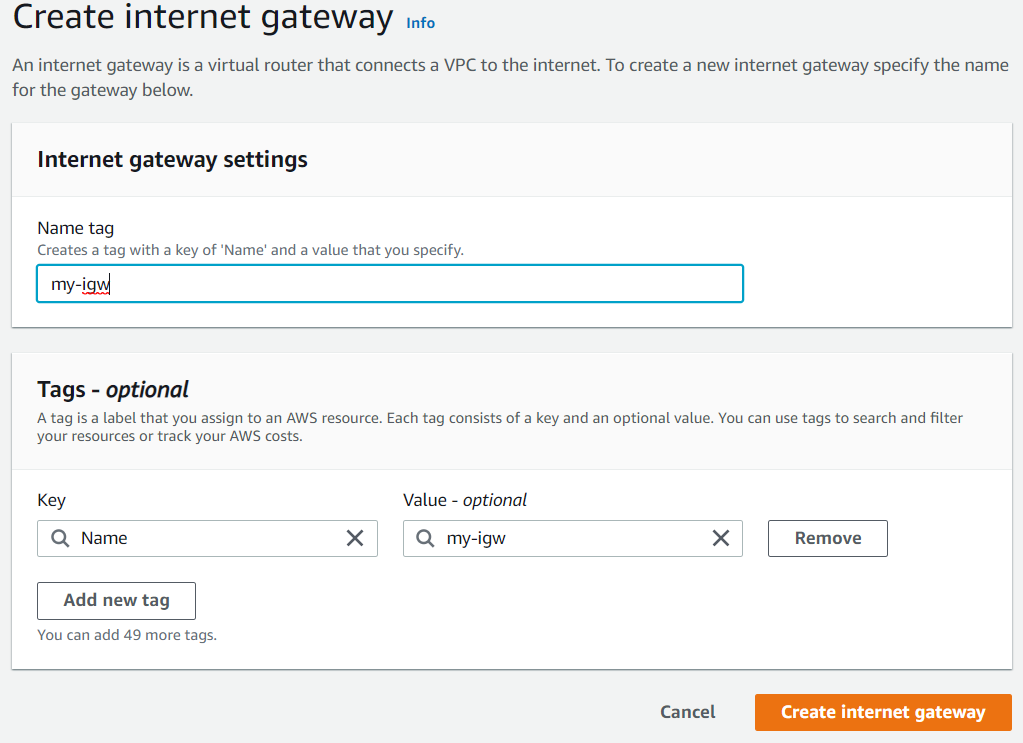
create public subnet -> 10.10.0.0/24

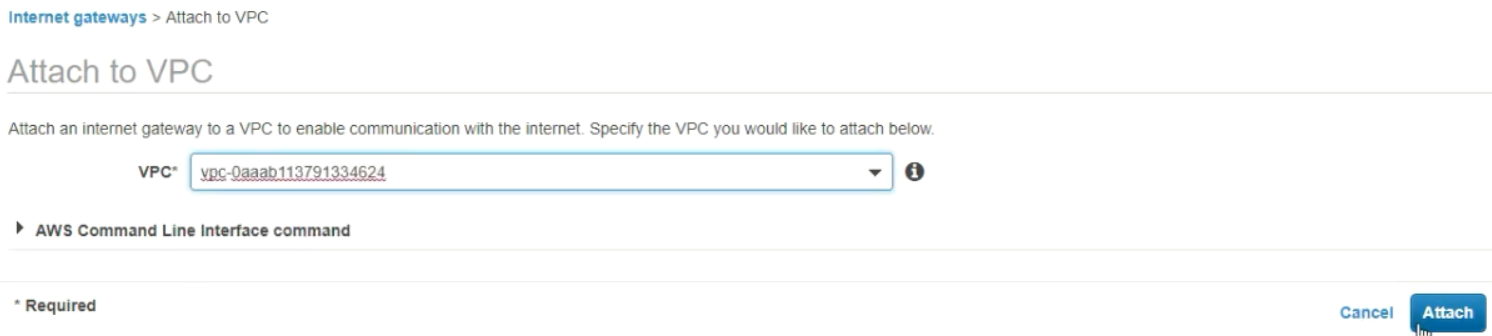


create private subnet -> 10.10.1.0/24

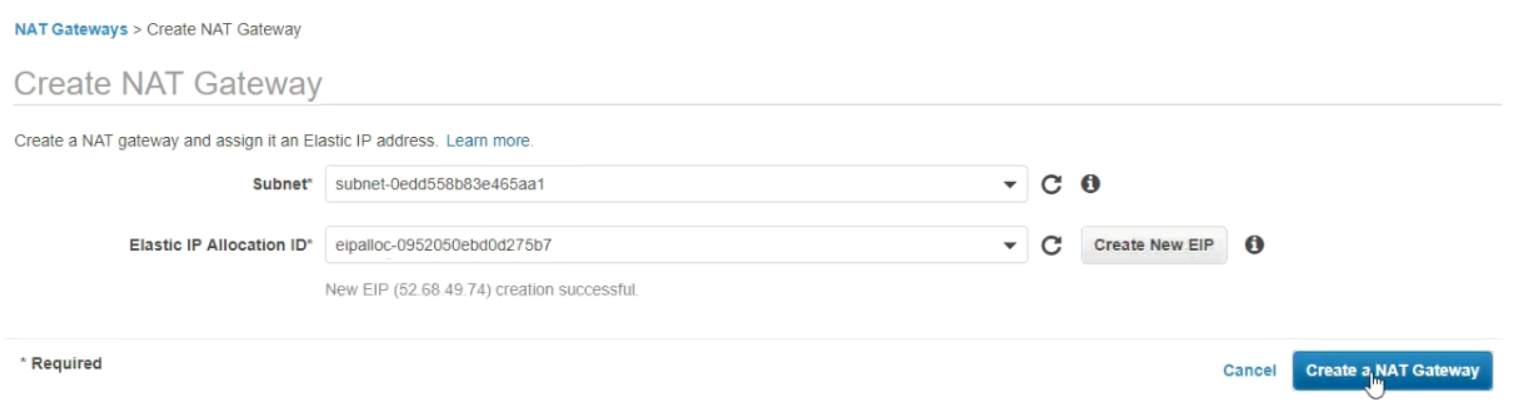


create an igw and assign to vpc

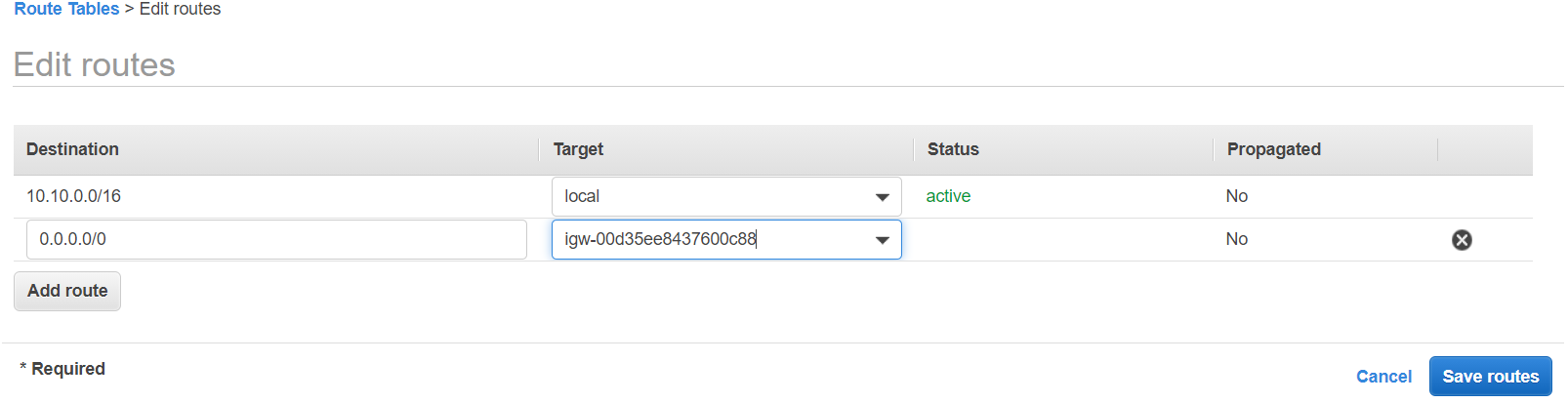




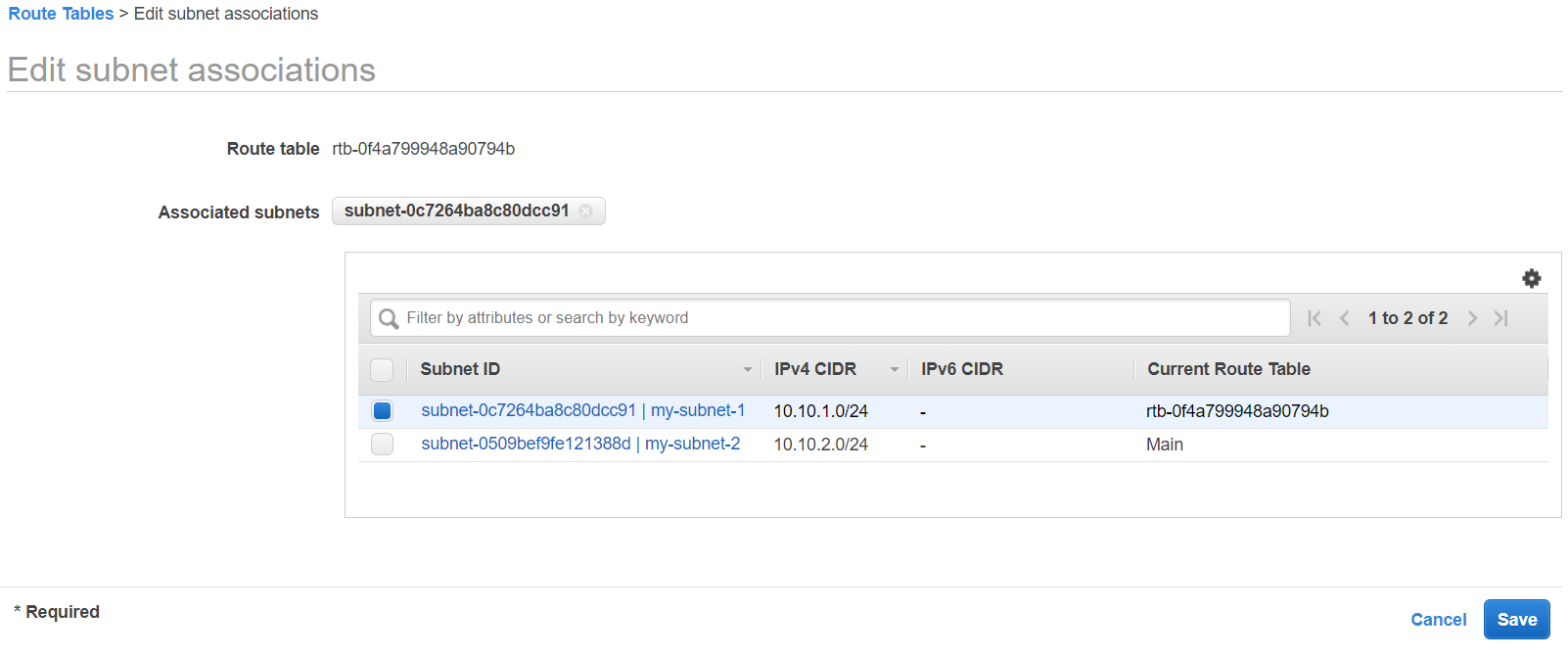
create a NAT gateway and assign elastic ip address and attach to public subnet



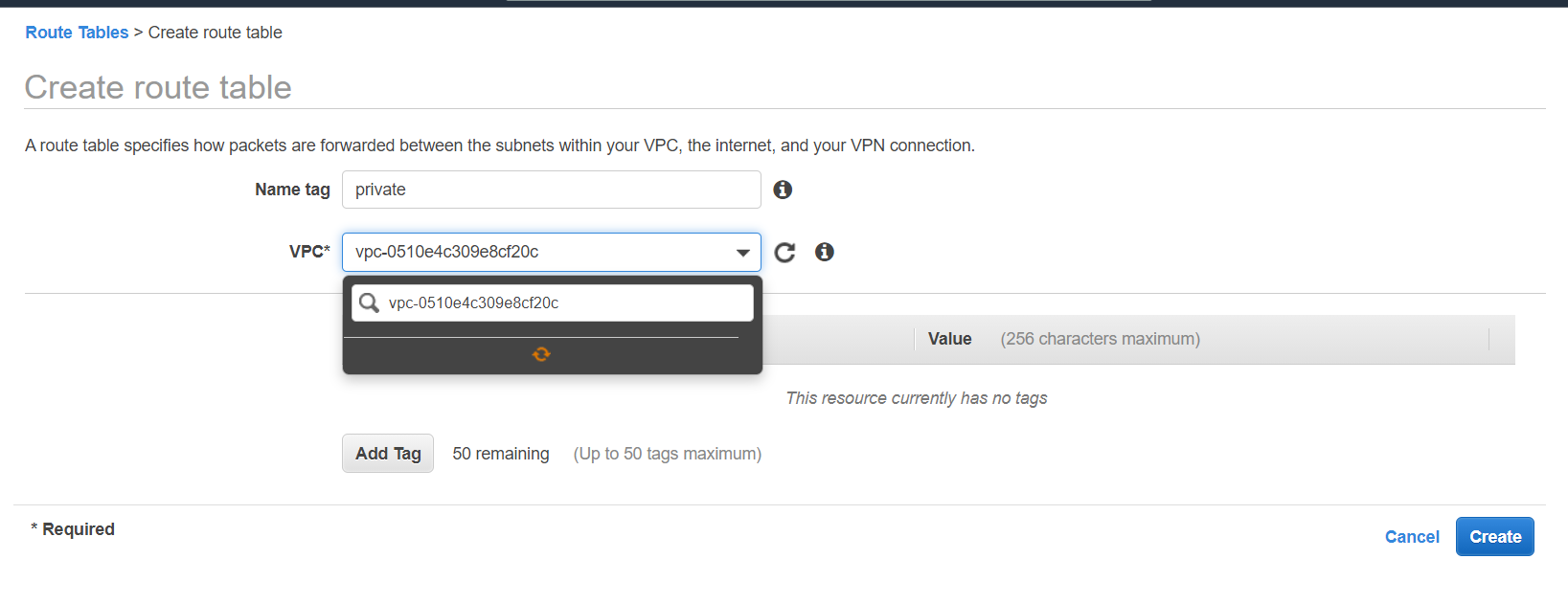
Create a two route tables



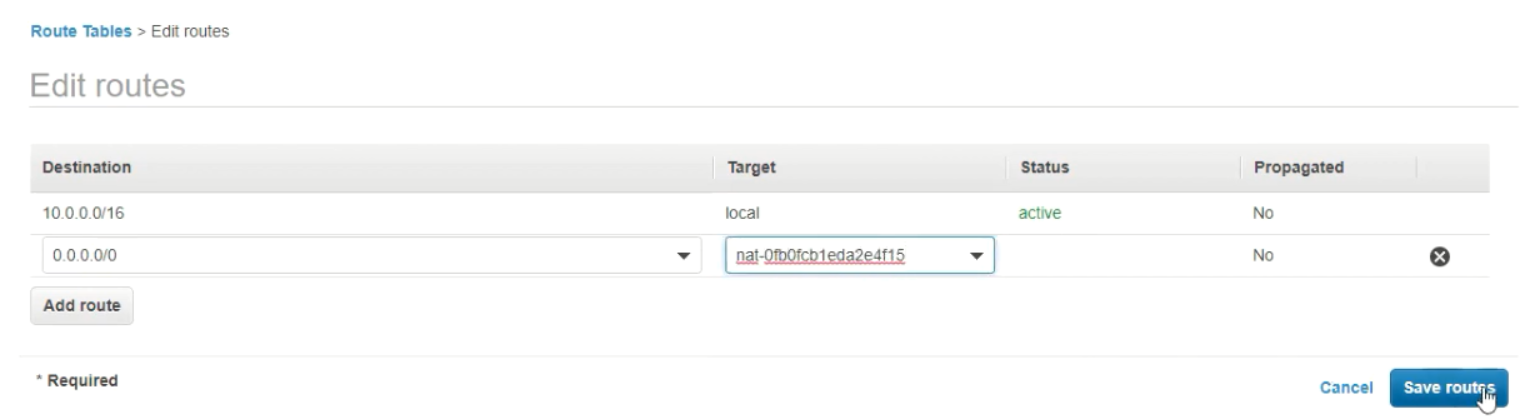
Attach to subnet-1 (public)



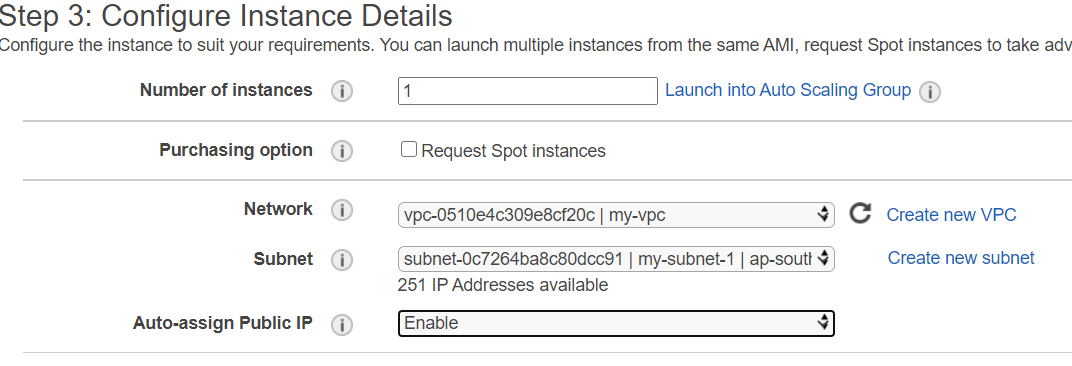
Creating private rt



Editing the route



Creating the ec2 machine in the public subnet and Launch another ec2 in the private subnet



Bastion server: connecting from public subnet to private subnet

For this we need pem file present on the bastion server

scp -i “vpc.pem” vpc.pem ubuntu @10.10.2.4:/home/ubuntu/vpc.pem

now login in the pubic ec2

ls you can see the vpc.pem file

chmod 400 vpc.pem

ssh -i vpc.pem [ubuntu@10.10.1.4](mailto:ubuntu@10.10.1.4)

now you can access the private ec2 machine login into pubic ec2 and not exposing to public

still if you not able to access it check the security group ports (openall)

nacl

create nacl of denial attacks means if we are getting traffic from one ip address and we want block we use it.

Nacl use for security for subnet level and it has allow and deny rules

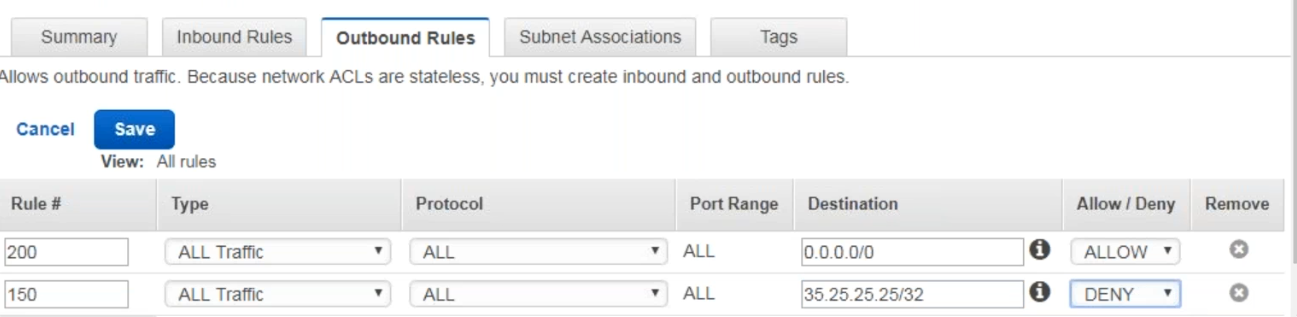
Create a nacl



If you connect nacl to subnet no traffic will come inside and go outside

By default, it will deny every thing

If you want to block particular ip we use 35.25.25.25/32



########### using cli ################

1. Now continue to create vpc with 4 subnets out of which two subnets are private and 2 are public.

## Create a vpc

aws ec2 create-vpc --cidr-block '10.10.0.0/16'

# For shell scripts export vpcid = 'vpc-0bec25679d1be0400'

# For powershell $vpcid = 'vpc-0bec25679d1be0400'

# vpc id

aws ec2 create-subnet --cidr-block '10.10.0.0/24' --vpc-id $vpcid

# subnet id

# $subnet1 = 'subnet-01cc74cc2dde74755'

aws ec2 create-subnet --cidr-block '10.10.1.0/24' --vpc-id $vpcid

# subnet id

# $subnet2 = 'subnet-02eff93367aab7879'

aws ec2 create-subnet --cidr-block '10.10.2.0/24' --vpc-id $vpcid

# subnet id

# $subnet3 = 'subnet-0ae049fff83fcc624'

aws ec2 create-subnet --cidr-block '10.10.3.0/24' --vpc-id $vpcid

# subnet id

# $subnet4 = 'subnet-0c05bfe3d7b003aa2'

# create internet gateway

aws ec2 create-internet-gateway

# internet gateway id

# $igw = 'igw-0a2618cdb8bf0fad3'

# attach internet gateway to vpc

aws ec2 attach-internet-gateway --internet-gateway-id $igw --vpc-id $vpcid

# create private route table

aws ec2 create-route-table --vpc-id $vpcid

# {

# "RouteTable": {

# "Associations": [],

# "PropagatingVgws": [],

# "RouteTableId": "rtb-053c4d74337724a5a",

# "Routes": [

# {

# "DestinationCidrBlock": "10.10.0.0/16",

# "GatewayId": "local",

# "Origin": "CreateRouteTable",

# "State": "active"

# }

# ],

# "Tags": [],

# "VpcId": "vpc-0bec25679d1be0400",

# "OwnerId": "798279872530"

# }

# }

# $privatert = 'rtb-053c4d74337724a5a'

# create public route table

aws ec2 create-route-table --vpc-id $vpcid

# $publicrt = 'rtb-0fbabde61ee2edf64'

# create a route for public rt to internet gateway

aws ec2 create-route --gateway-id $igw --route-table-id $publicrt --destination-cidr-block '0.0.0.0/0'

# associate public rt to subnet1 and subnet2

aws ec2 associate-route-table --route-table-id $publicrt --subnet-id $subnet1

# {

# "AssociationId": "rtbassoc-0ed5fb13e526212ab",

# "AssociationState": {

# "State": "associated"

# }

# }

aws ec2 associate-route-table --route-table-id $publicrt --subnet-id $subnet2

# {

# "AssociationId": "rtbassoc-031c89f0ff89746dd",

# "AssociationState": {

# "State": "associated"

# }

# }

# associate private rt to subnet3 and subnet4

aws ec2 associate-route-table --route-table-id $privatert --subnet-id $subnet3

# {

# "AssociationId": "rtbassoc-09df82b5c4c045b91",

# "AssociationState": {

# "State": "associated"

# }

# }

aws ec2 associate-route-table --route-table-id $privatert --subnet-id $subnet4

# {

# "AssociationId": "rtbassoc-01327320d20923690",

# "AssociationState": {

# "State": "associated"

# }

# }

1. Create a Security Group and NACL

# Create a security group to allow ssh http and https inbound and everything outbound

aws ec2 create-security-group --description 'Allowsshandhttp' --group-name 'allowimp' --vpc-id $vpcid

# $sgid = 'sg-0a32918b1a48c9986'

aws ec2 authorize-security-group-ingress --group-id $sgid --protocol 'tcp' --port '22' --cidr '0.0.0.0/0'

aws ec2 authorize-security-group-ingress --group-id $sgid --protocol 'tcp' --port '80' --cidr '0.0.0.0/0'

aws ec2 authorize-security-group-ingress --group-id $sgid --protocol 'tcp' --port '443' --cidr '0.0.0.0/0'

# create nacl

aws ec2 create-network-acl --vpc-id $vpcid

# "NetworkAcl": {

# "Associations": [],

# "Entries": [

# {

# "CidrBlock": "0.0.0.0/0",

# "Egress": true,

# "IcmpTypeCode": {},

# "PortRange": {},

# "Protocol": "-1",

# "RuleAction": "deny",

# "RuleNumber": 32767

# },

# {

# "CidrBlock": "0.0.0.0/0",

# "Egress": false,

# "IcmpTypeCode": {},

# "PortRange": {},

# "Protocol": "-1",

# "RuleAction": "deny",

# "RuleNumber": 32767

# }

# ],

# "IsDefault": false,

# "NetworkAclId": "acl-0f171f7081fa484c5",

# "Tags": [],

# "VpcId": "vpc-0bec25679d1be0400",

# "OwnerId": "798279872530"

# }

# }

# $nacl = 'acl-0f171f7081fa484c5'

# add nacl entry to allow communication within vpc both inbound and outbound

aws ec2 create-network-acl-entry --network-acl-id $nacl --ingress --rule-number 300 --protocol tcp --port-range From=0,To=65535 --rule-action allow --cidr-block 10.10.0.0/16

aws ec2 create-network-acl-entry --network-acl-id $nacl --egress --rule-number 300 --protocol tcp --port-range From=0,To=65535 --rule-action allow --cidr-block 0.0.0.0/0

aws ec2 create-network-acl-entry --network-acl-id $nacl --ingress --rule-number 310 --protocol tcp --port-range From=22,To=22 --rule-action allow --cidr-block 0.0.0.0/0

aws ec2 create-network-acl-entry --network-acl-id $nacl --ingress --rule-number 320 --protocol tcp --port-range From=80,To=80 --rule-action allow --cidr-block 0.0.0.0/0

# deny all traffic from 8.8.8.8

aws ec2 create-network-acl-entry --network-acl-id $nacl --ingress --rule-number 290 --protocol tcp --port-range From=0,To=65535 --rule-action deny --cidr-block 8.8.8.8/32