Contents

[Prerequisites 2](#_Toc44855980)

[Lab 6A: Multi-tier Web App with ALB and NLB 2](#_Toc44855981)

[Lab 6A using Web Management Console 2](#_Toc44855982)

[Create VPC, Subnet, IGW, NAT, Routing Table, Keys, Sec Groups (Labs4c1) VPC Peering (Labs4c2), Instances on App Layer (Labs5c1) 2](#_Toc44855983)

[Create Security Group for Instances A, B and C, D 4](#_Toc44855984)

[Create Target Groups for NLB 7](#_Toc44855985)

[Create NLB and its listener 9](#_Toc44855986)

[Configure Web Instances using Internal NLB 11](#_Toc44855987)

[Create ALB 14](#_Toc44855988)

[Review with the browser 17](#_Toc44855989)

[Lab 6A using Command Line (Windows) 18](#_Toc44855990)

[Create VPC, Subnet, IGW, NAT, Routing Table, Keys, Sec Groups (Labs4c1) VPC Peering (Labs4c2), Instances on App Layer (Labs5c1) 18](#_Toc44855991)

[Create Sec Groups for NLB, Target Groups, Register Instances and finally, create NLB and its listener 25](#_Toc44855992)

[Create instances for Web Tier using Internal balancer, Sec Groups for ALB, Target Groups for ALB, ALB and ALB’ Listener 26](#_Toc44855993)

[Review using browser 29](#_Toc44855994)

[Clean Resources 30](#_Toc44855995)

# Prerequisites

Labs1c1 have to be done and the context for Administrative user have to activated on Command Line Session.

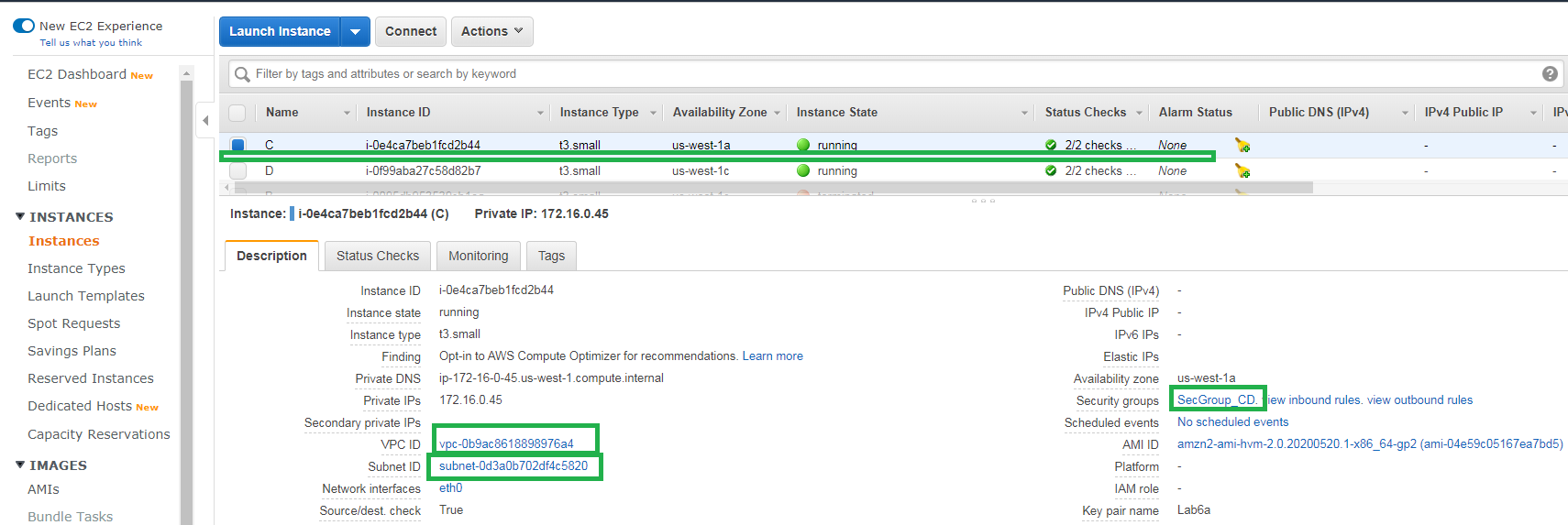
Labs5c1 have to be done, because you learn how to: Deploy Network Infrastructure, Securize instances, deploy applications using Docker and create a functional ALB.

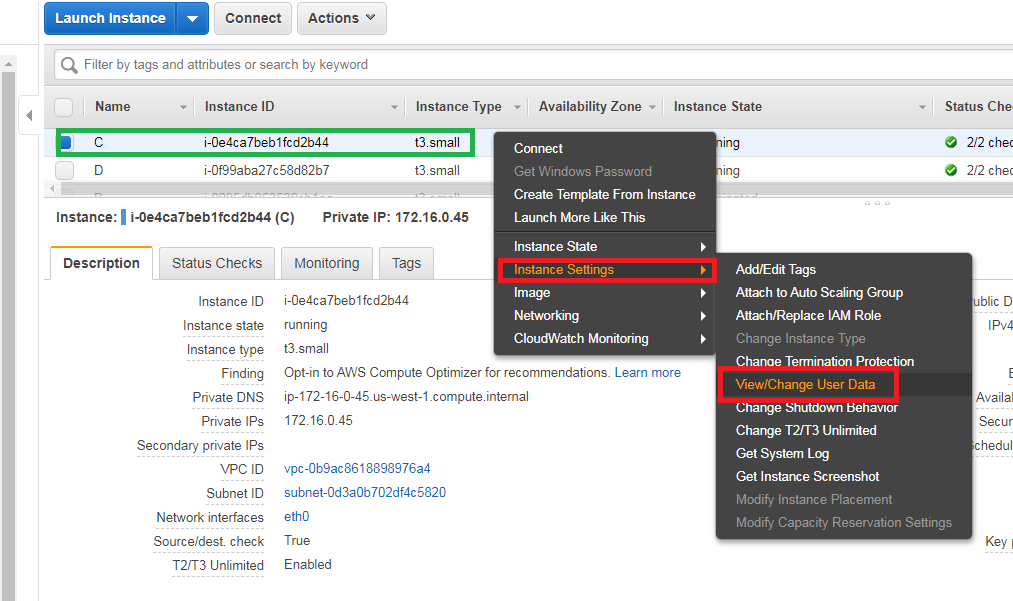
# Lab 6A: Multi-tier Web App with ALB and NLB

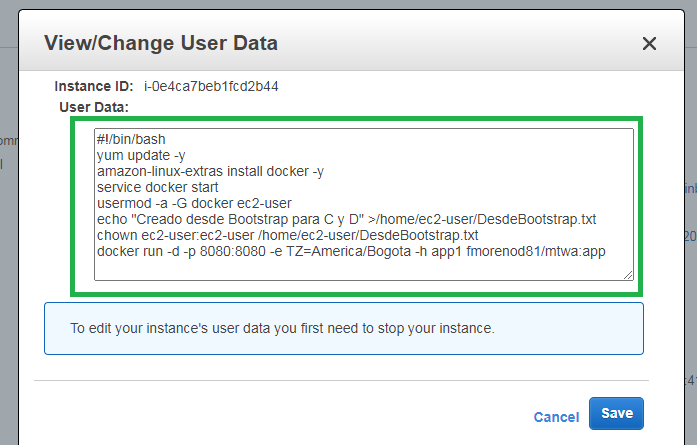
## Lab 6A using Web Management Console

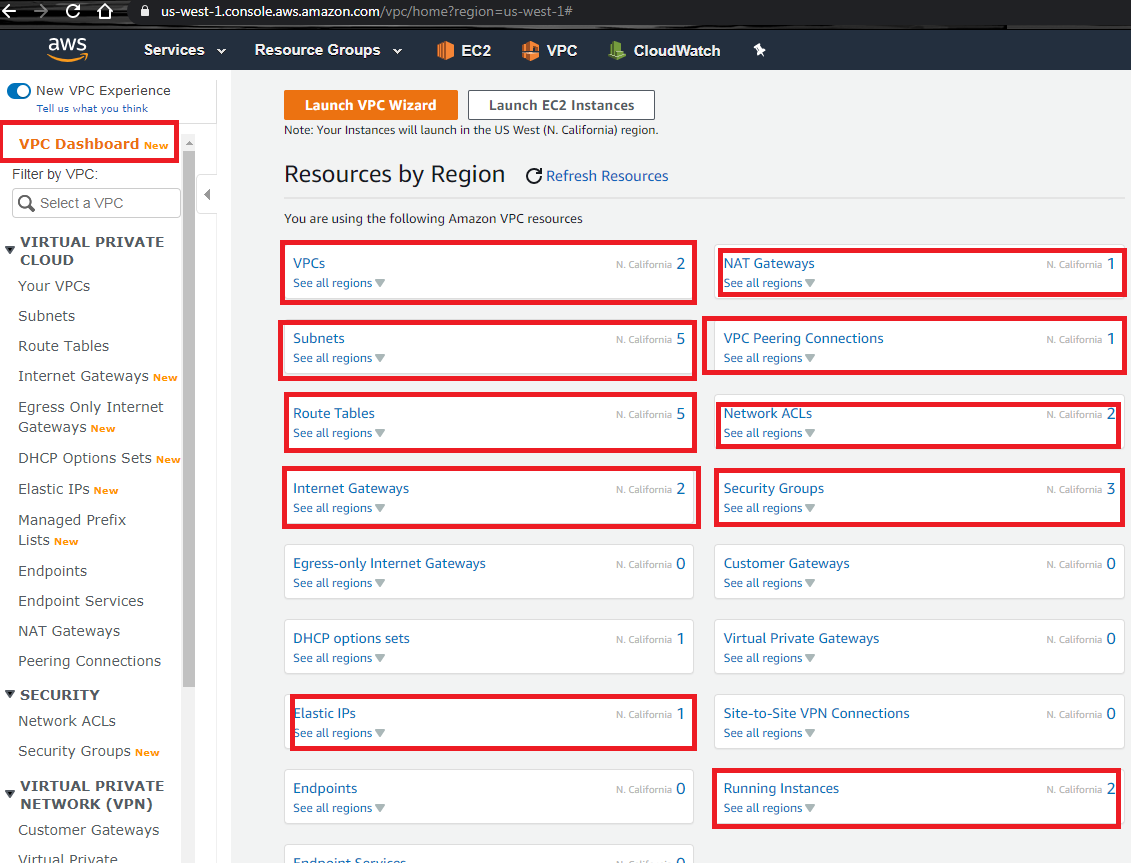
### Create VPC, Subnet, IGW, NAT, Routing Table, Keys, Sec Groups (Labs4c1) VPC Peering (Labs4c2), Instances on App Layer (Labs5c1)

To remember the creation of the instance using a bootstrapping (user-data), and the correct network information and its security groups

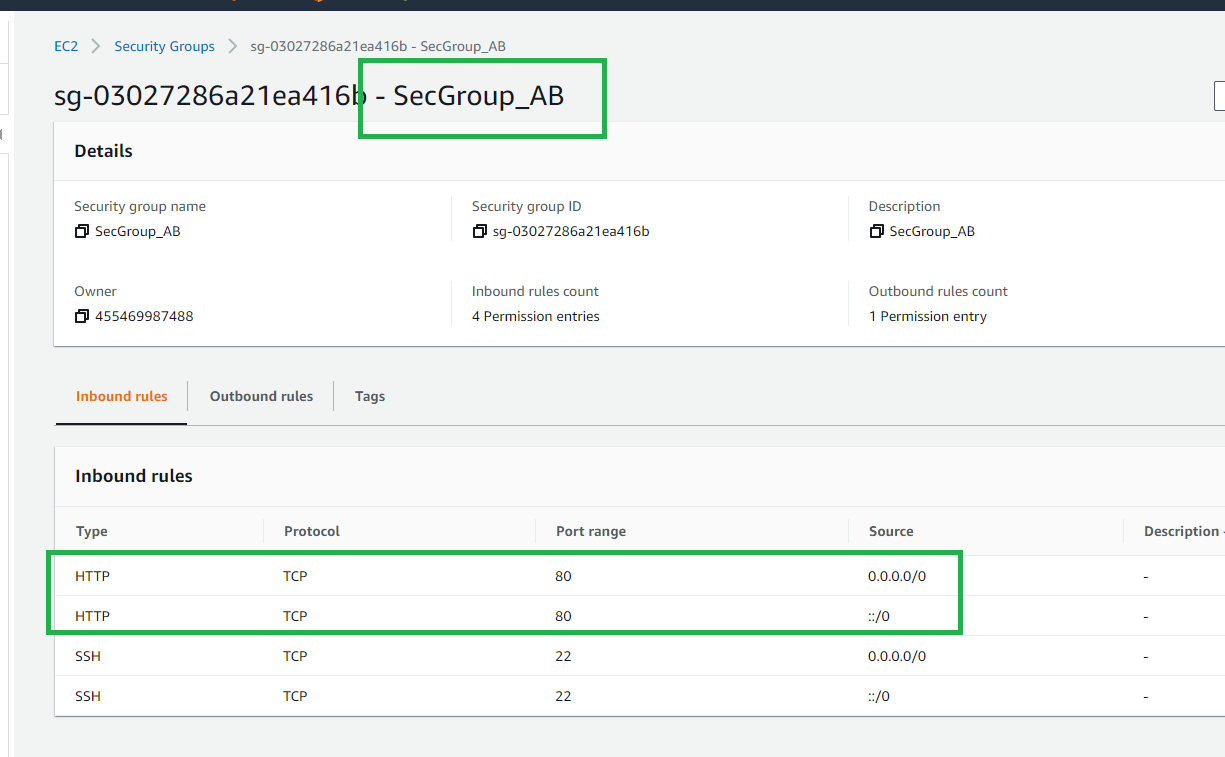


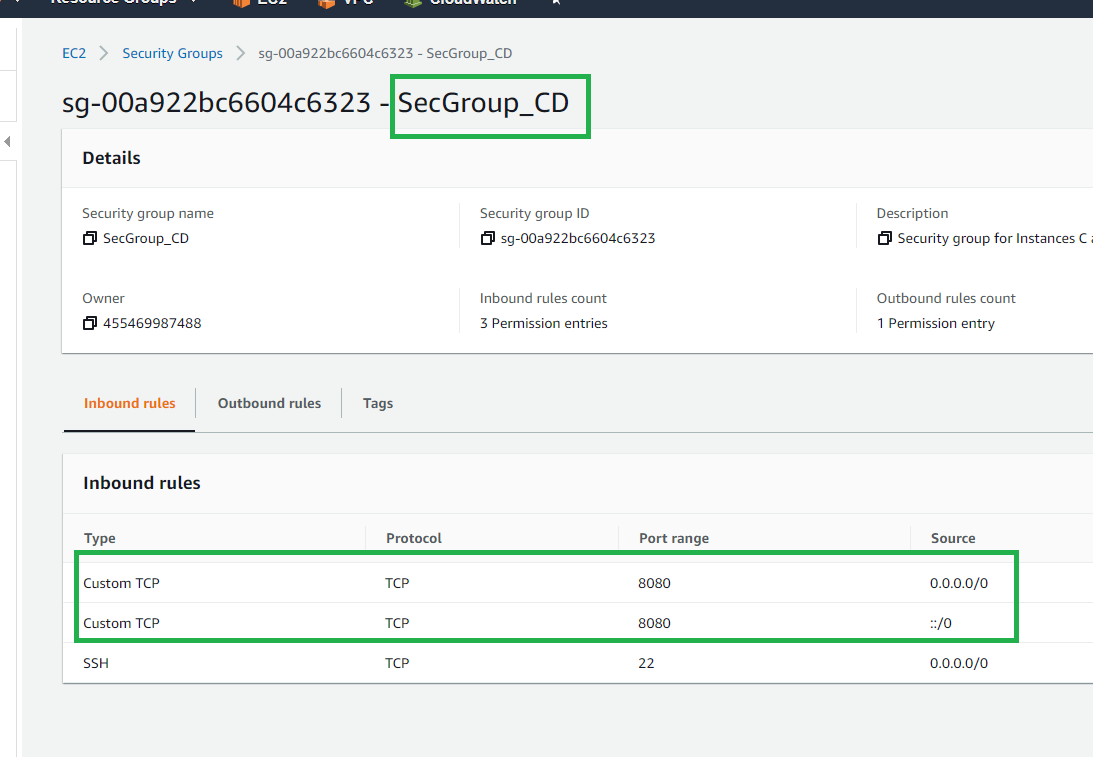






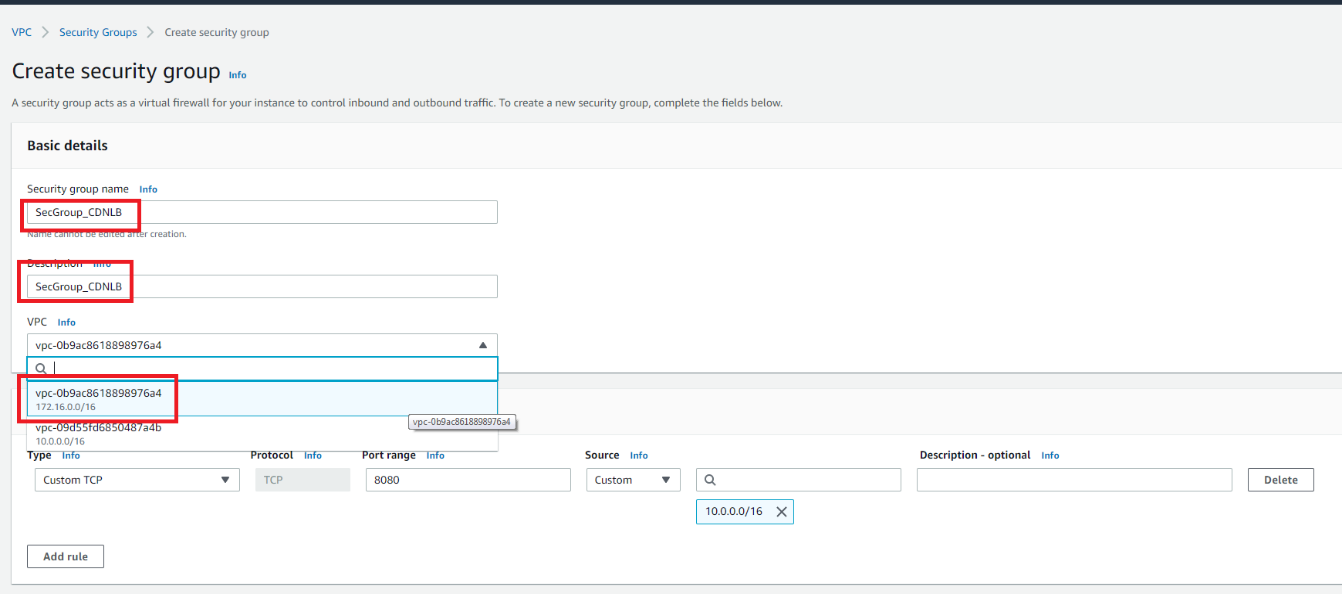
### Create Security Group for Instances A, B and C, D

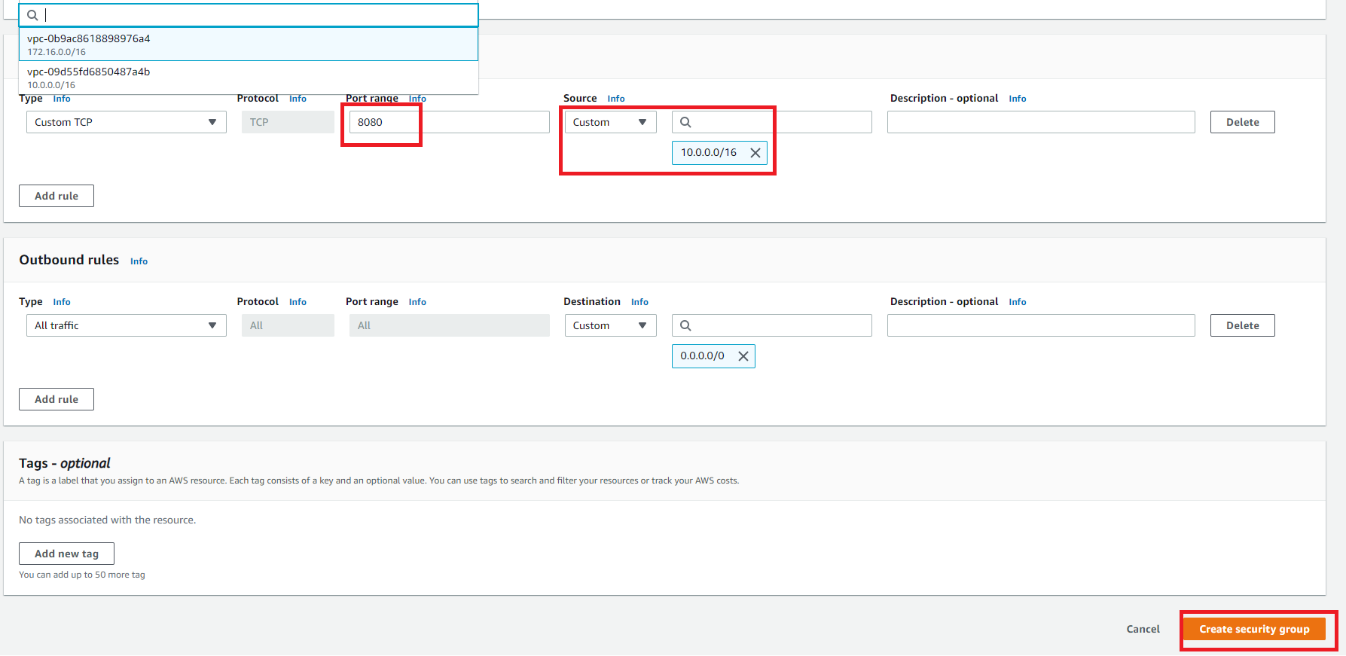




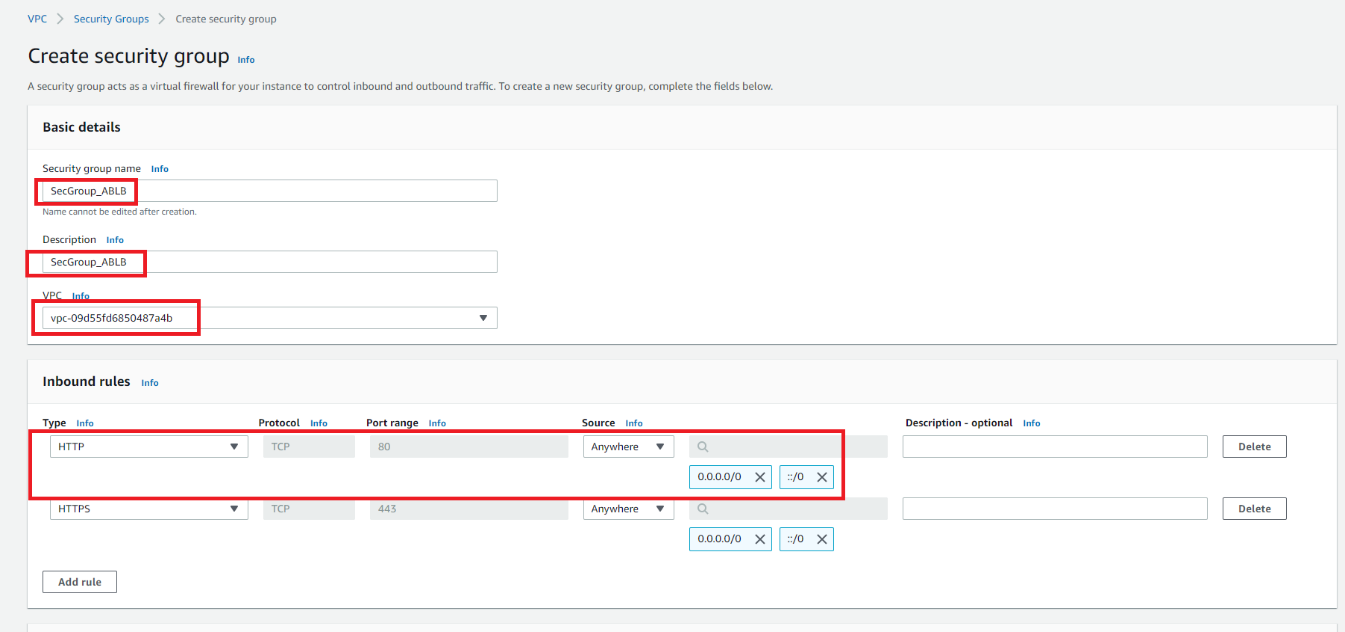
Create Security Groups for Balancers

For Internal Balancer,

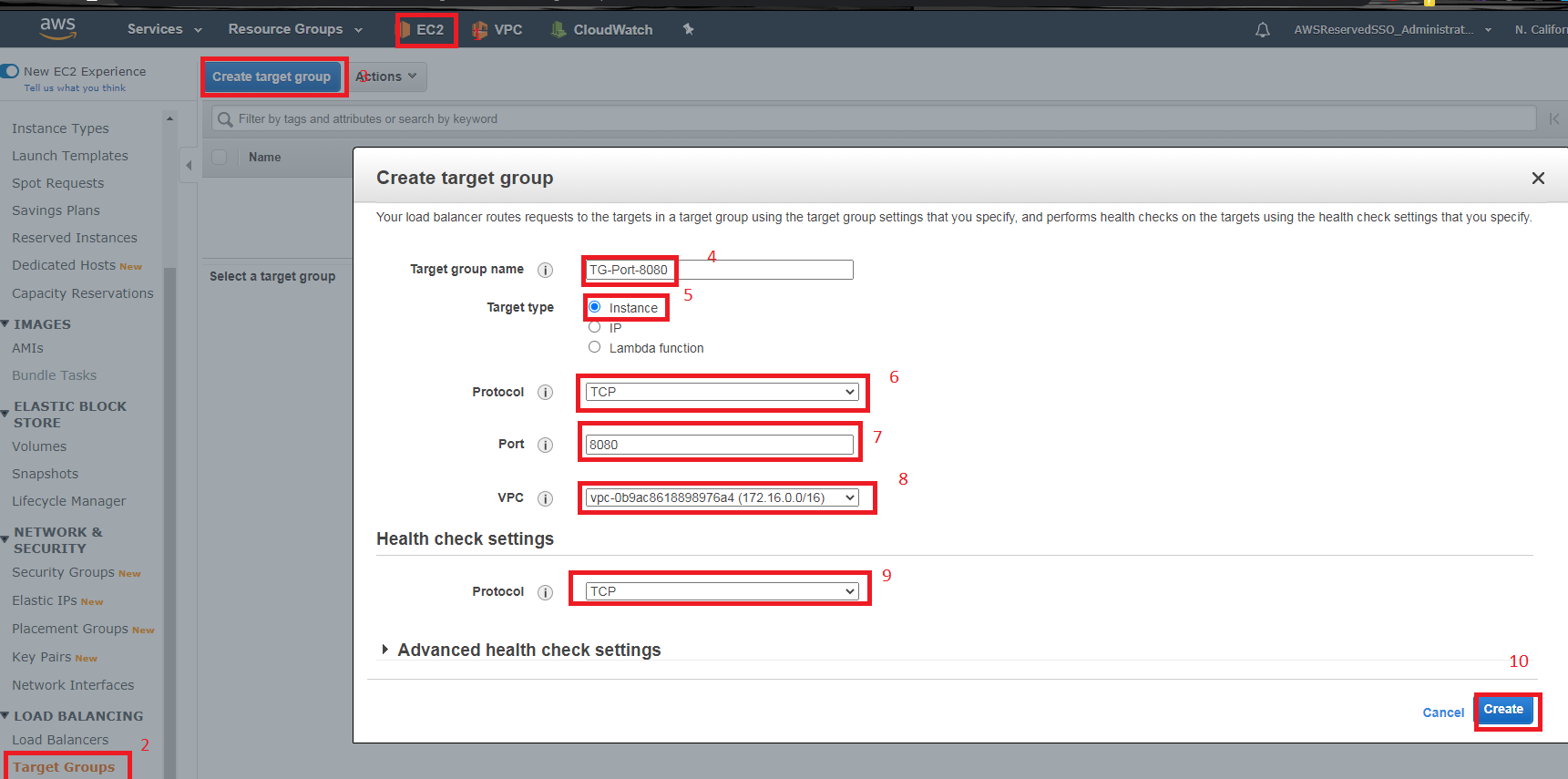


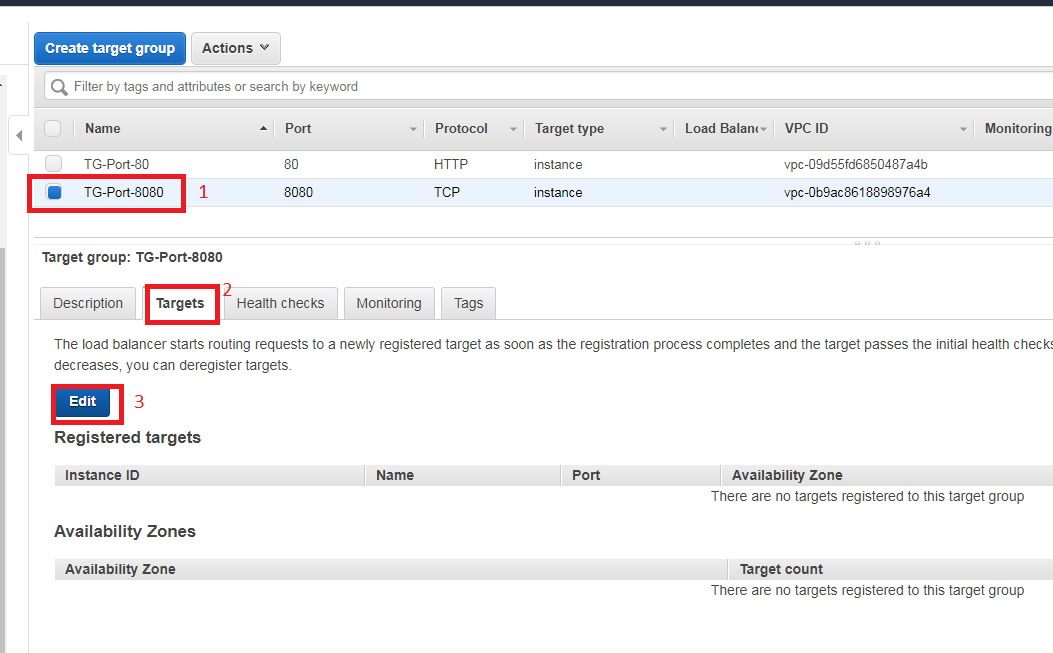


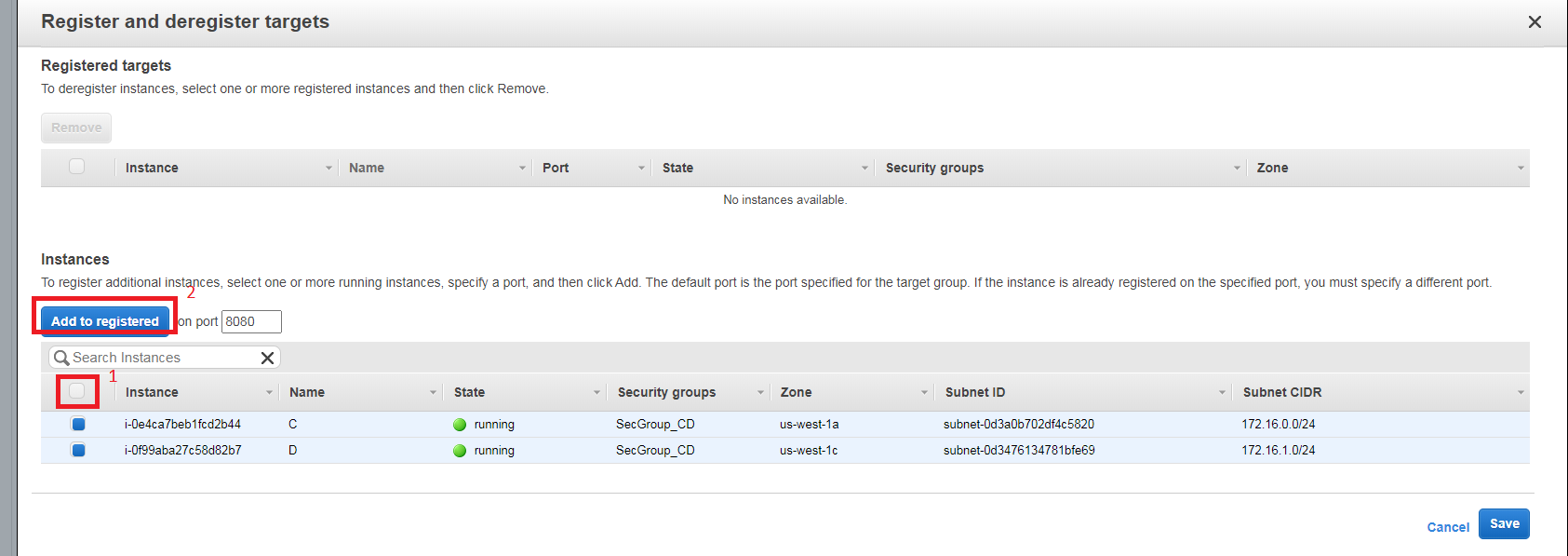
For External Balancer,

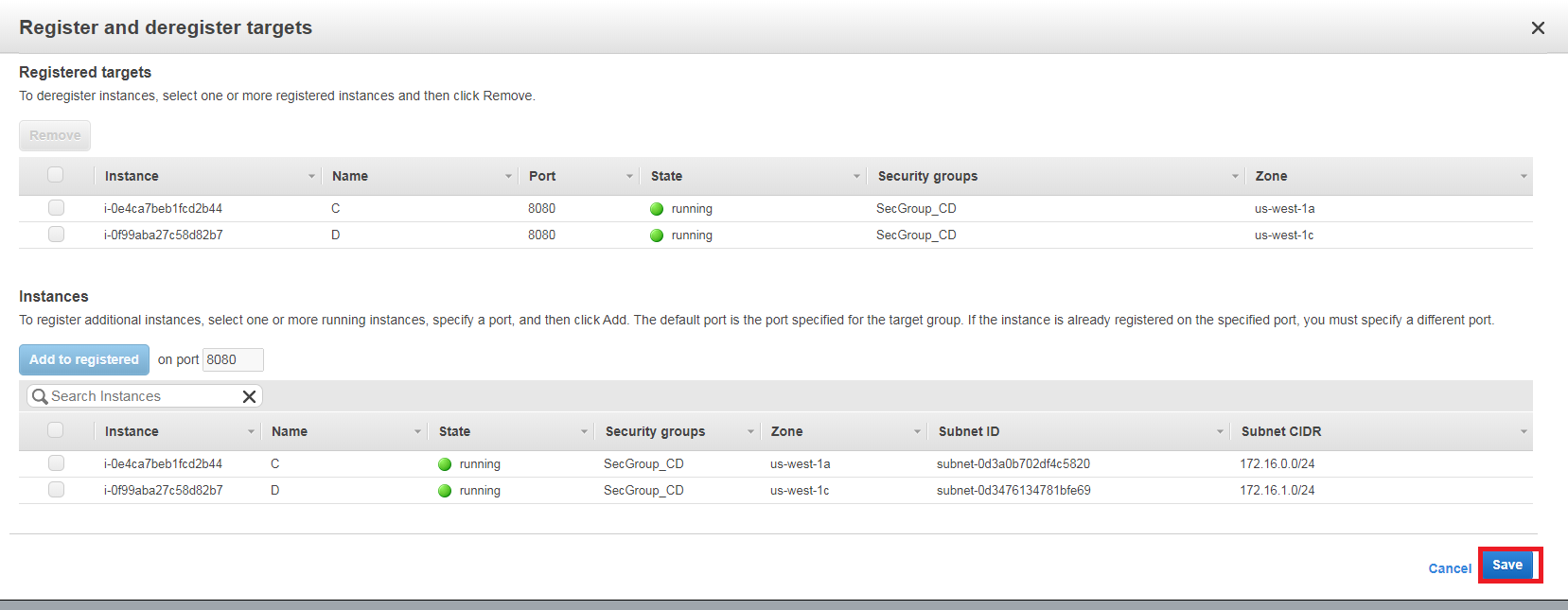


### Create Target Groups for NLB

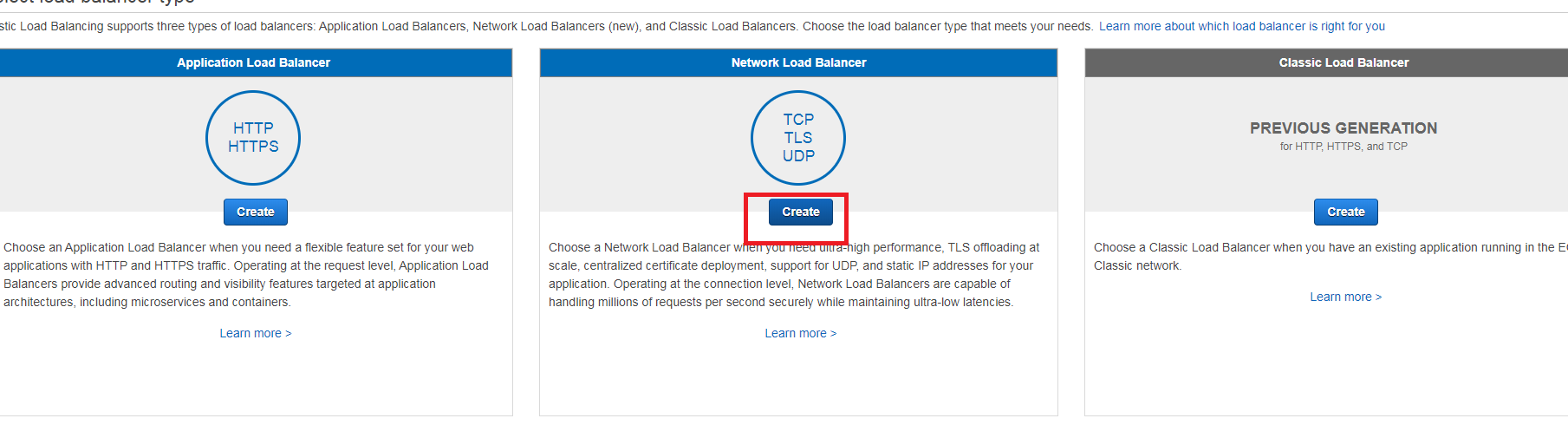


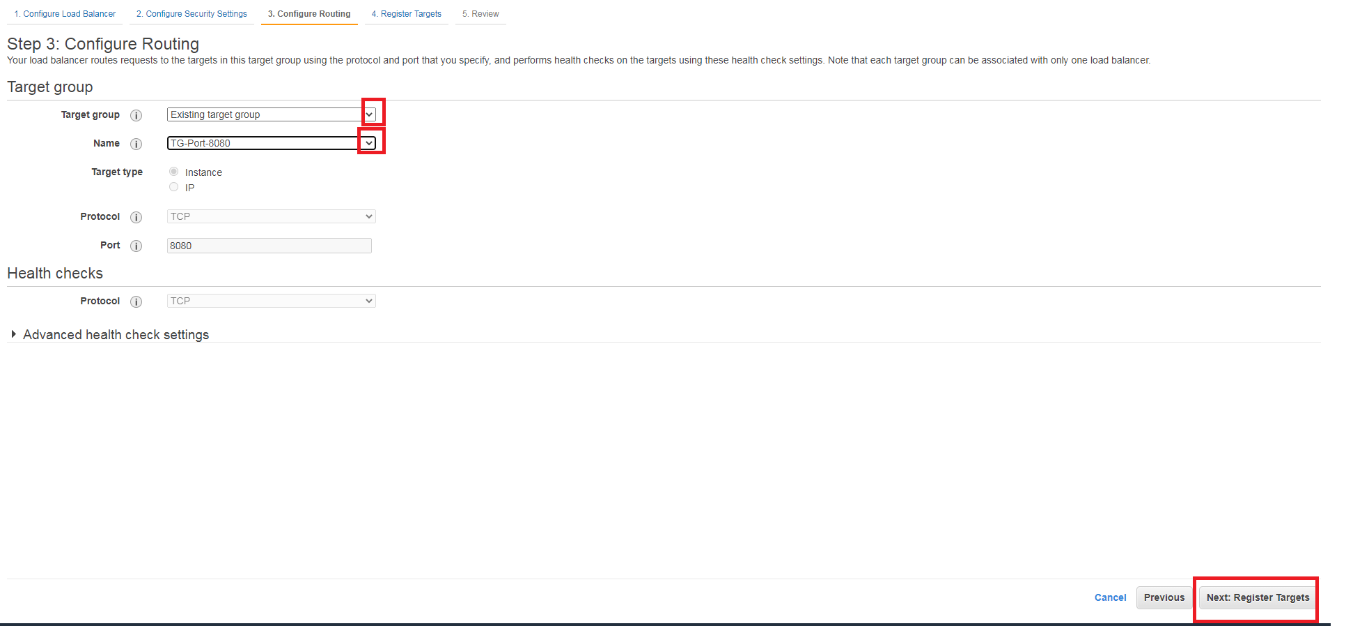
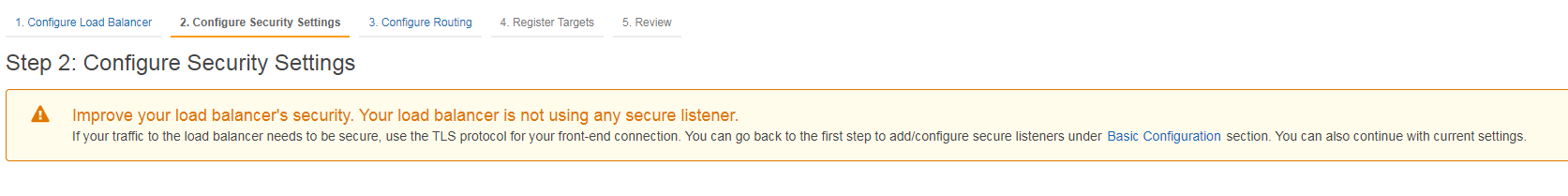
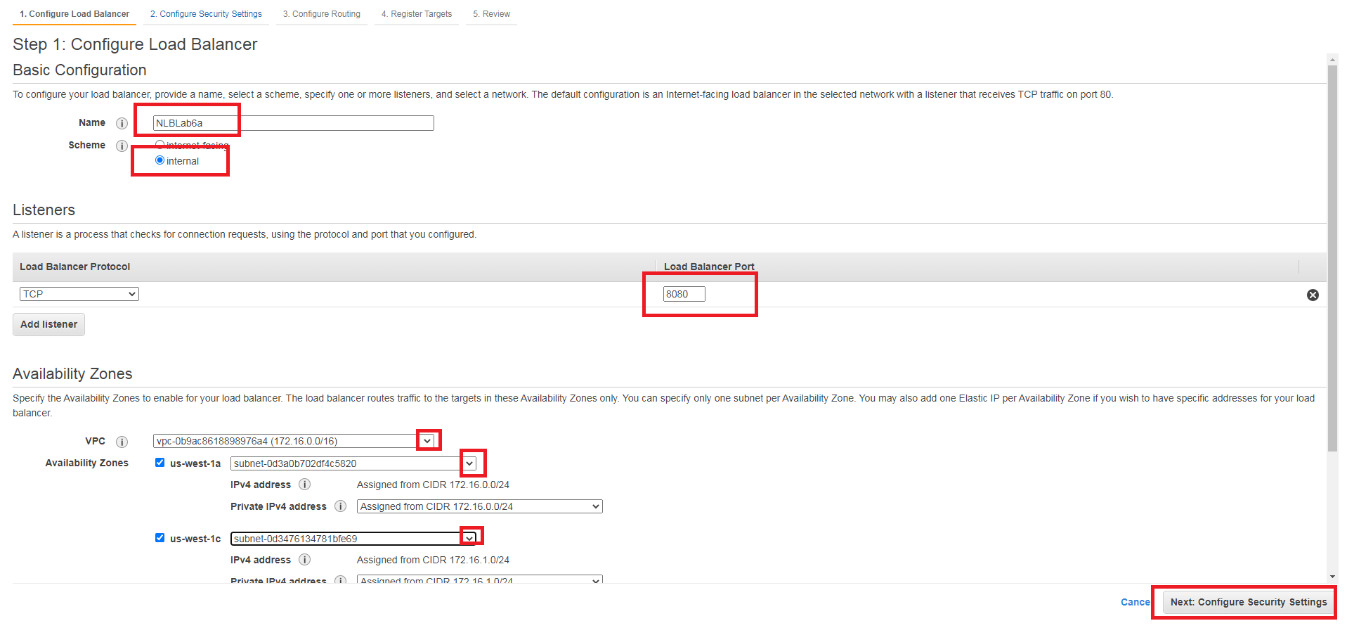


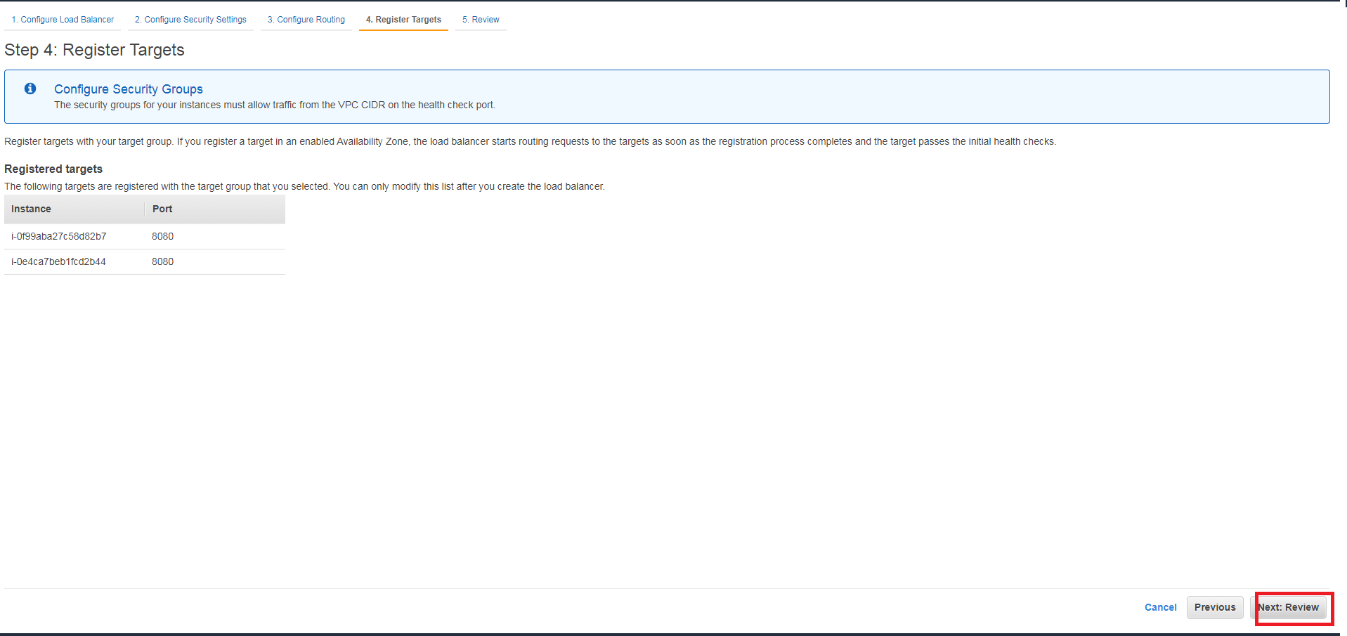


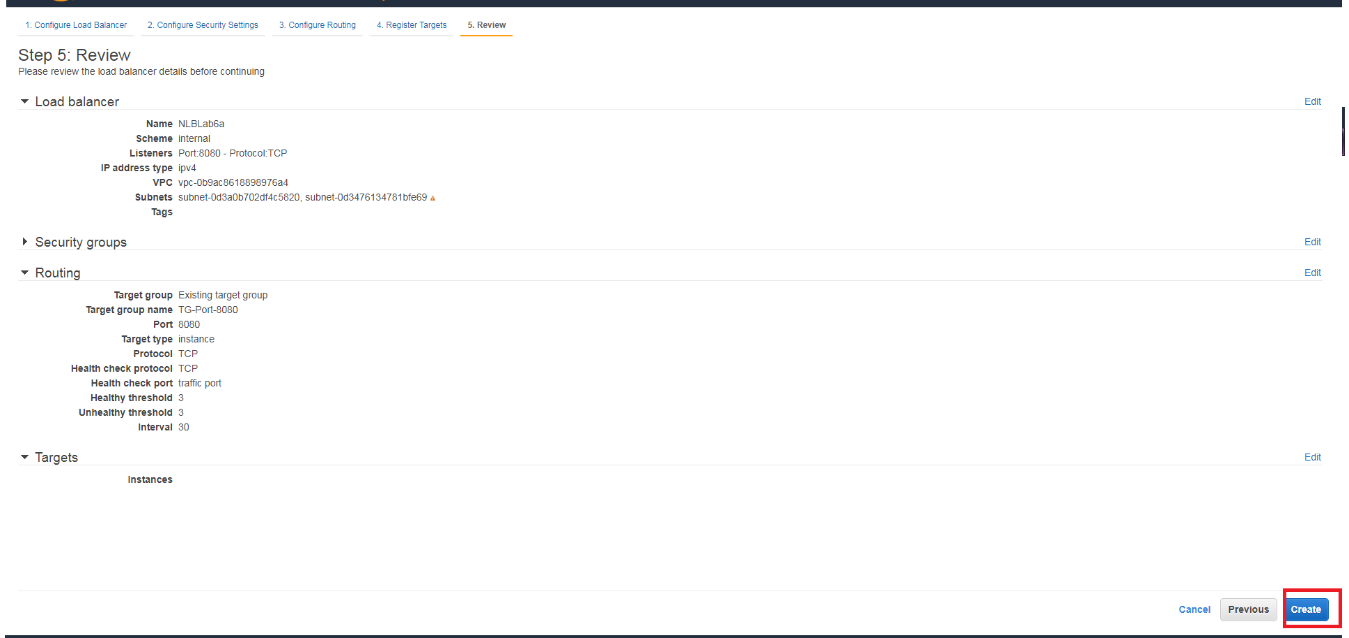


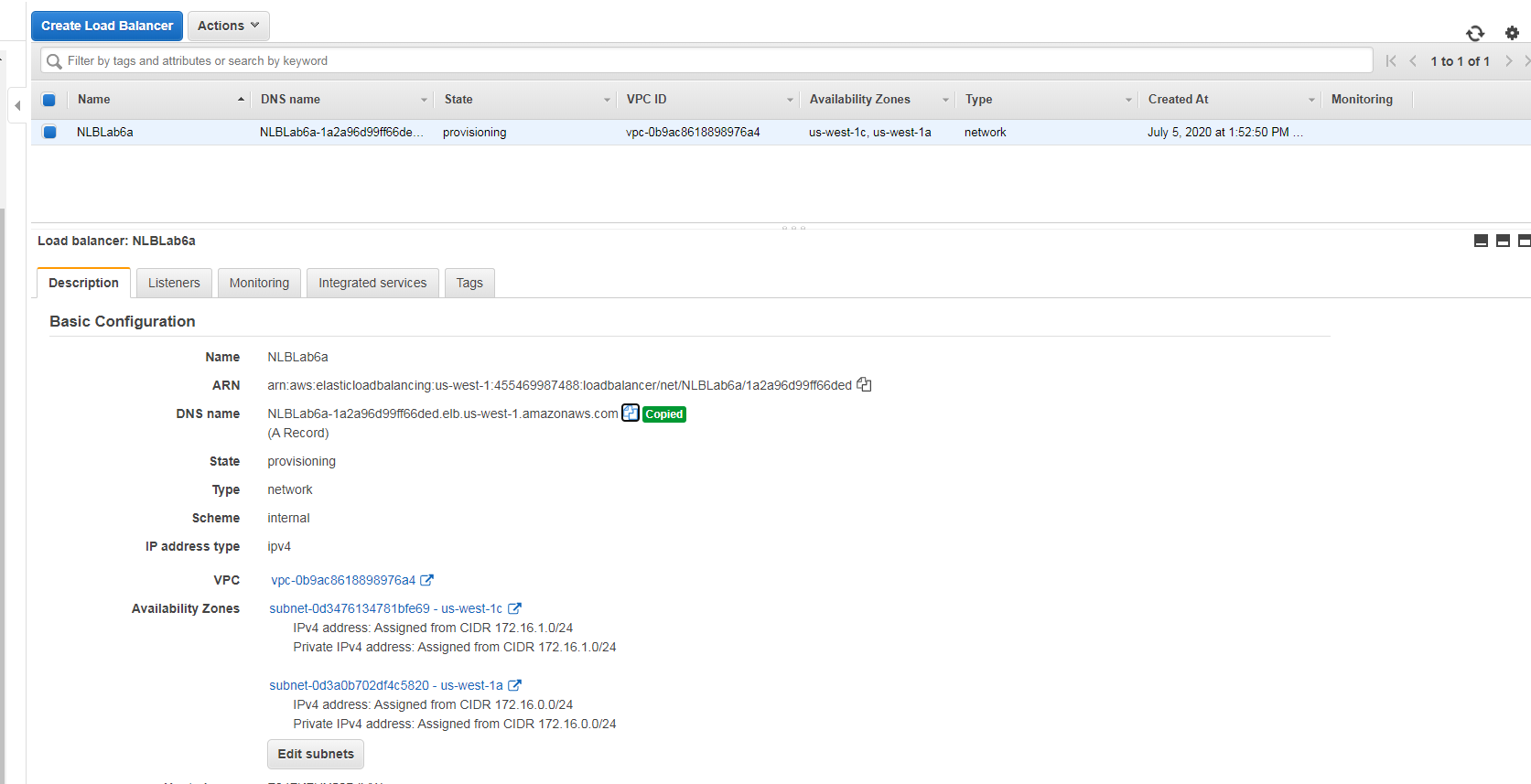
### Create NLB and its listener







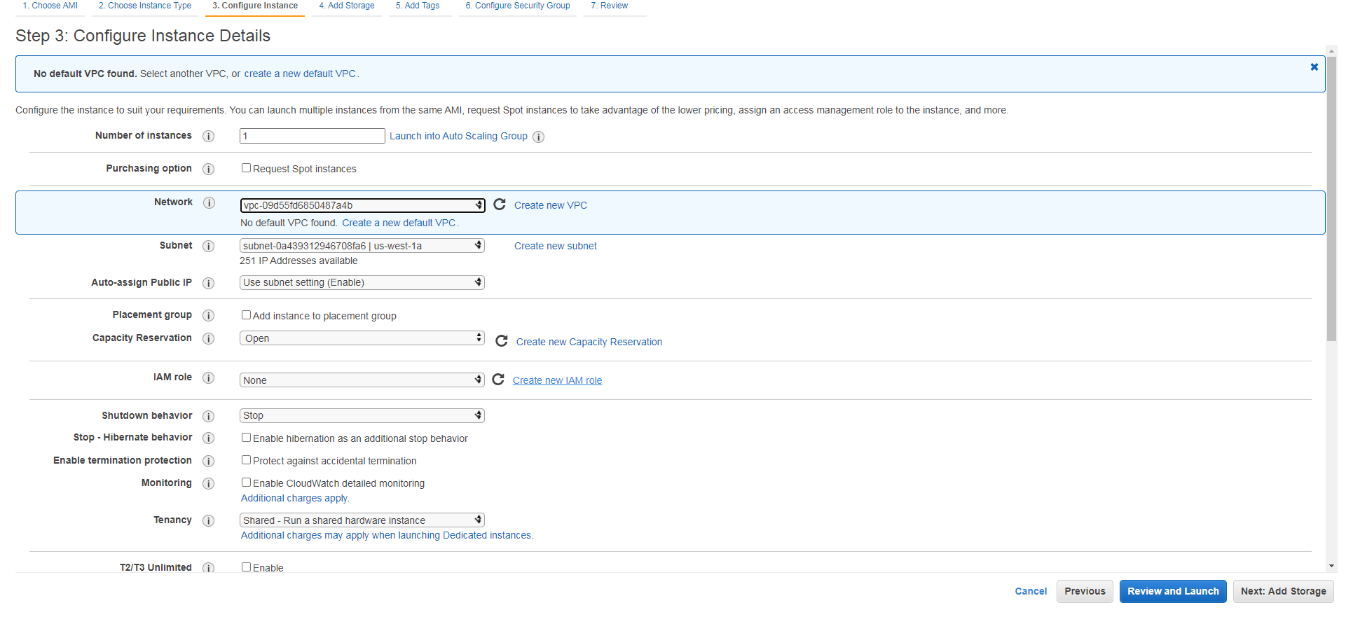


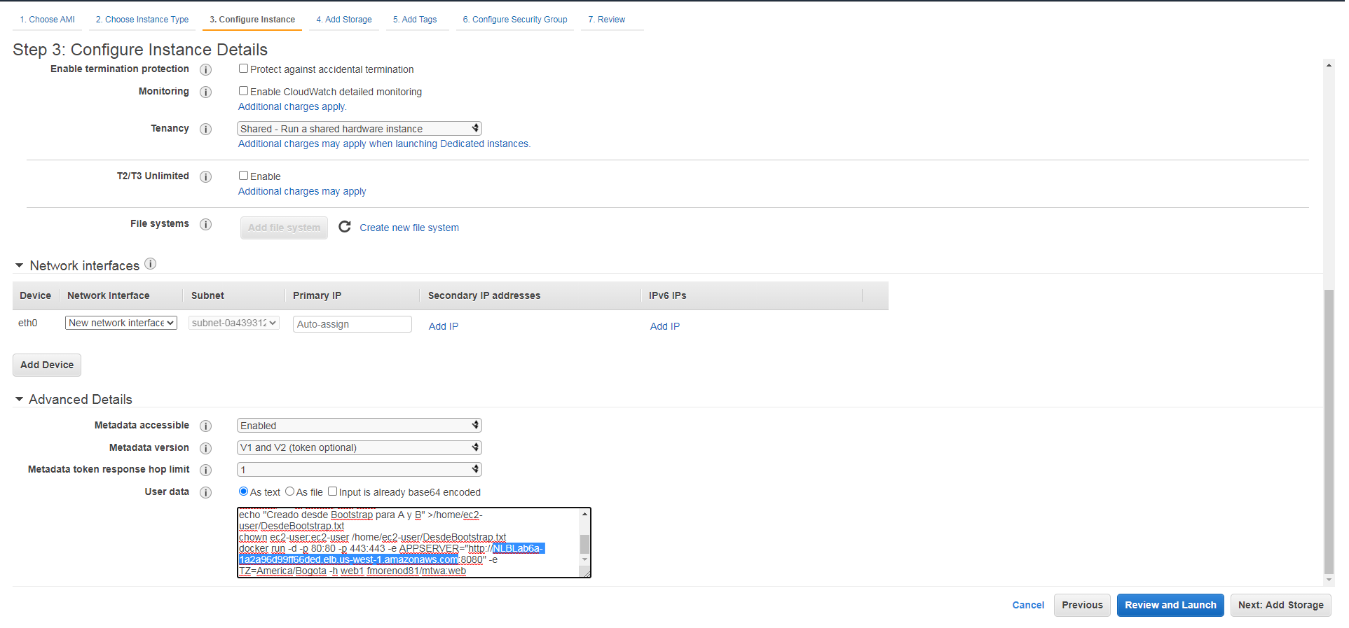


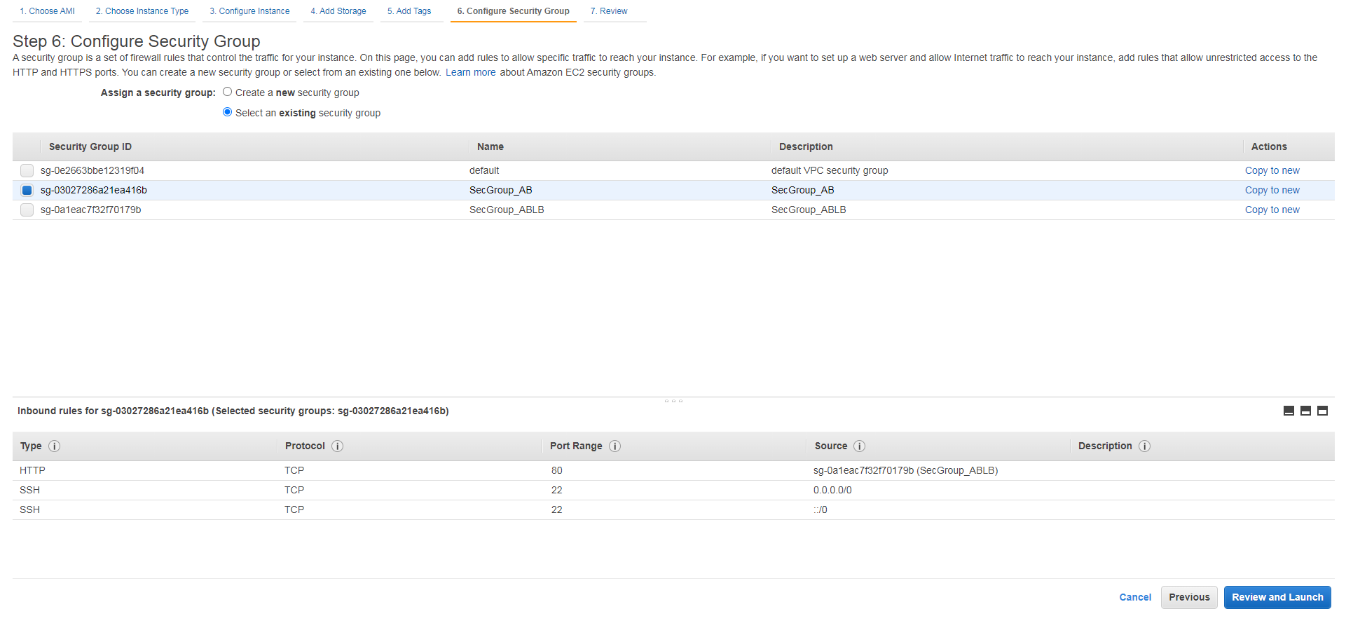
Copy the Internal NLB to configure internal variable on Web App

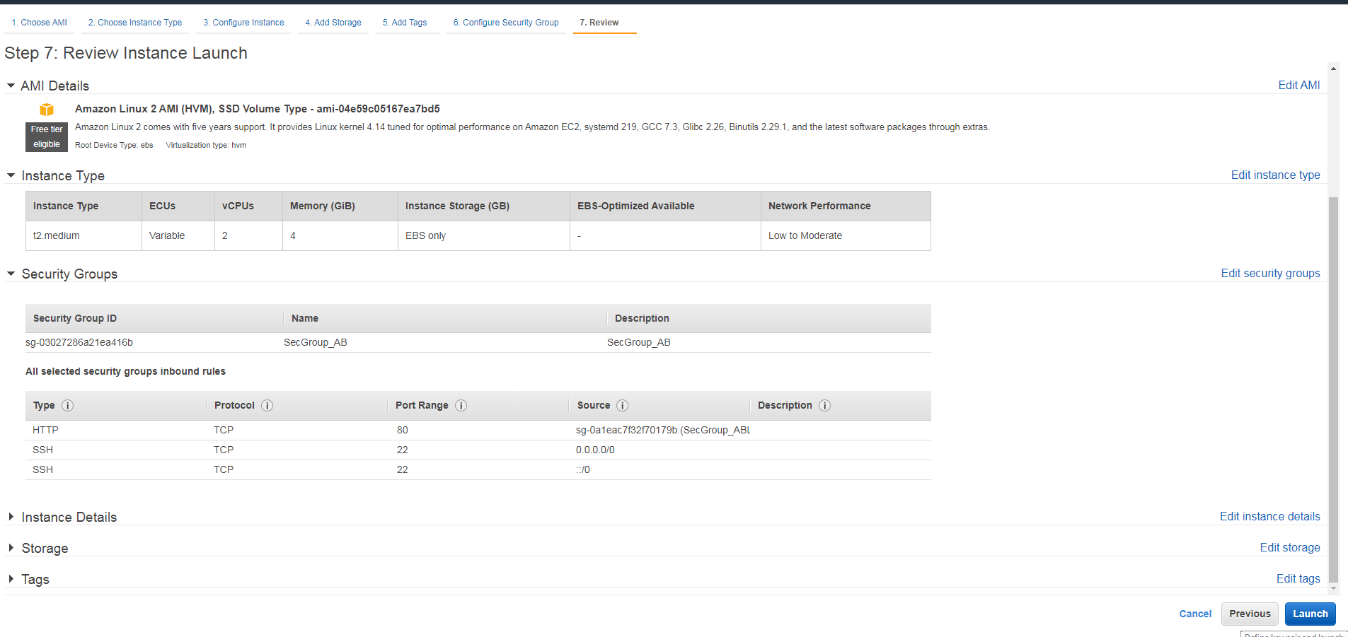
### Configure Web Instances using Internal NLB

We use the normal procedure to create an instance: correct subnet, security group and the configuration of the initial script.

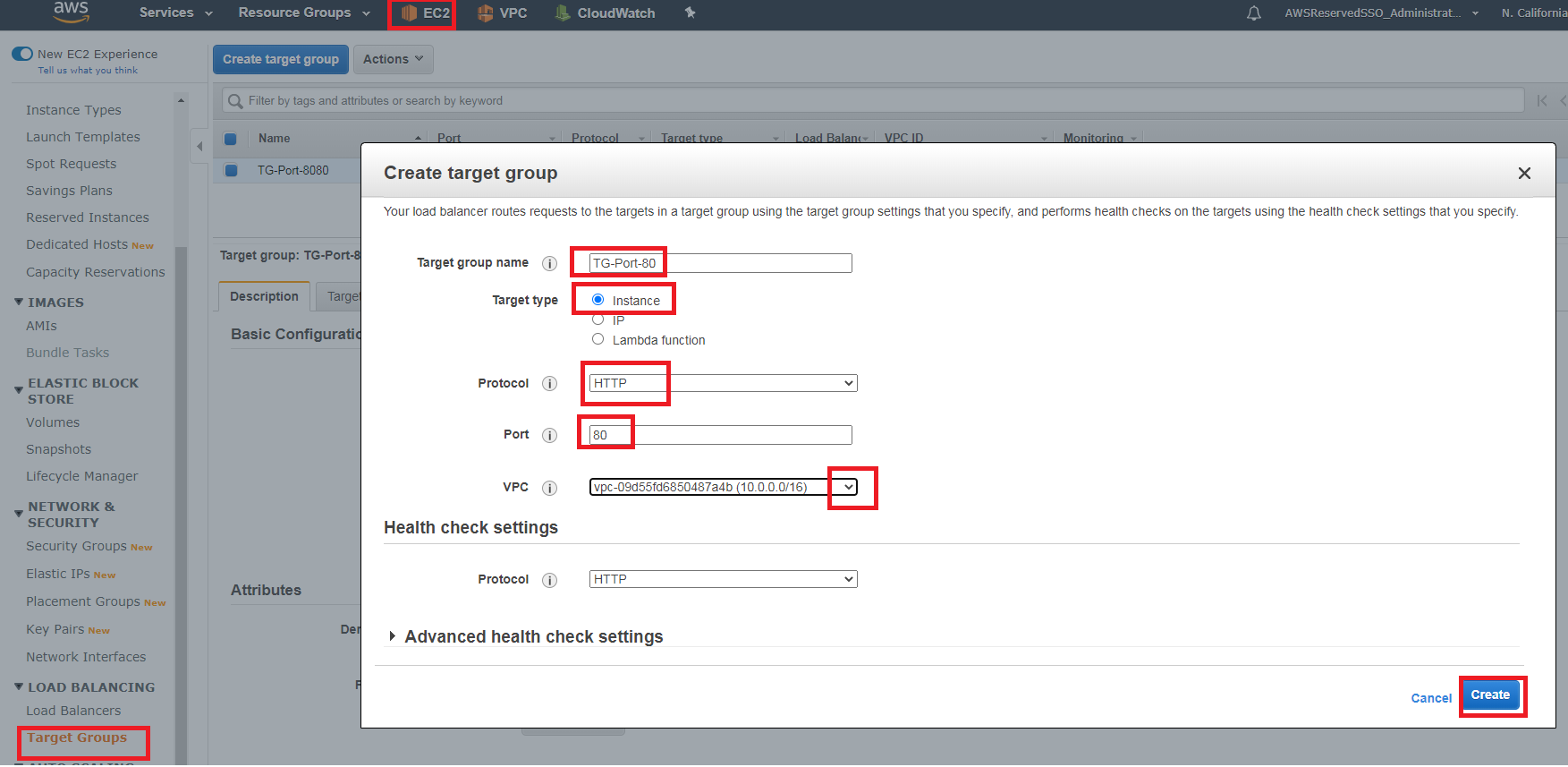


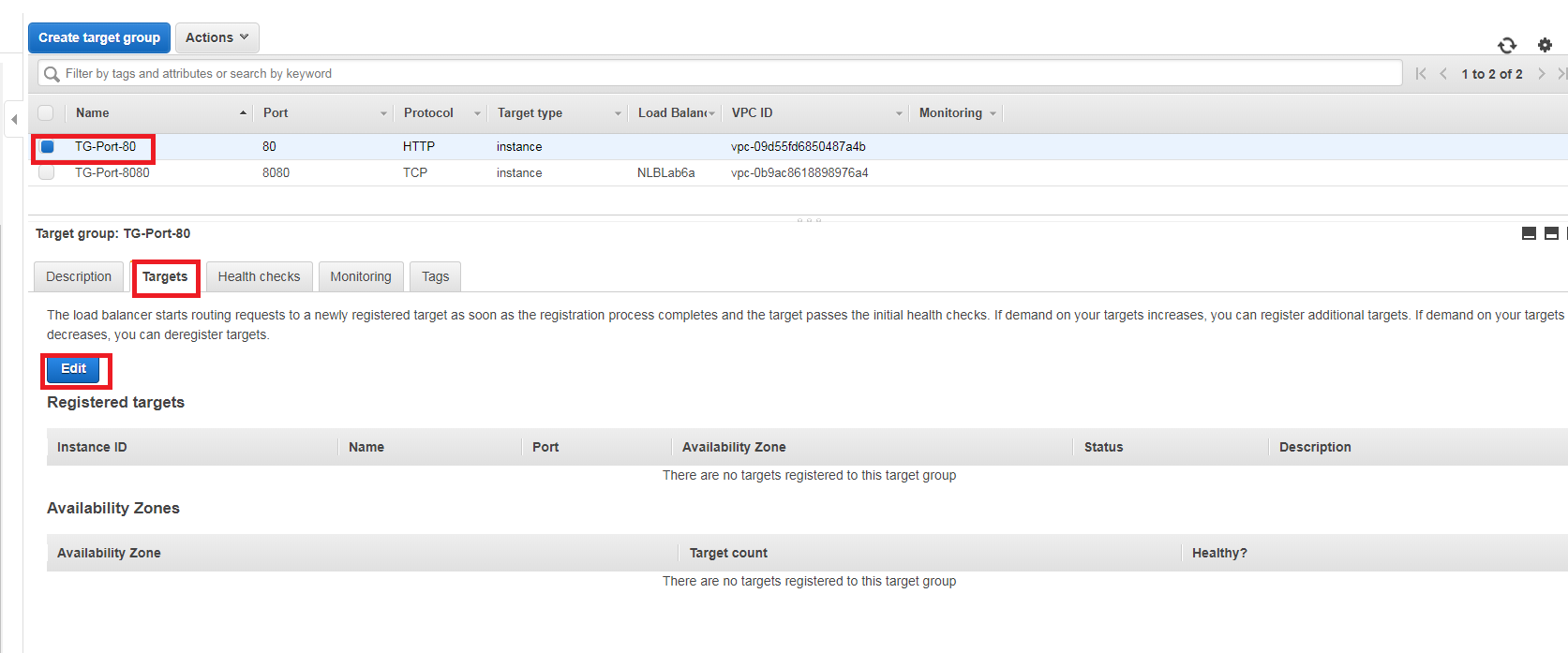


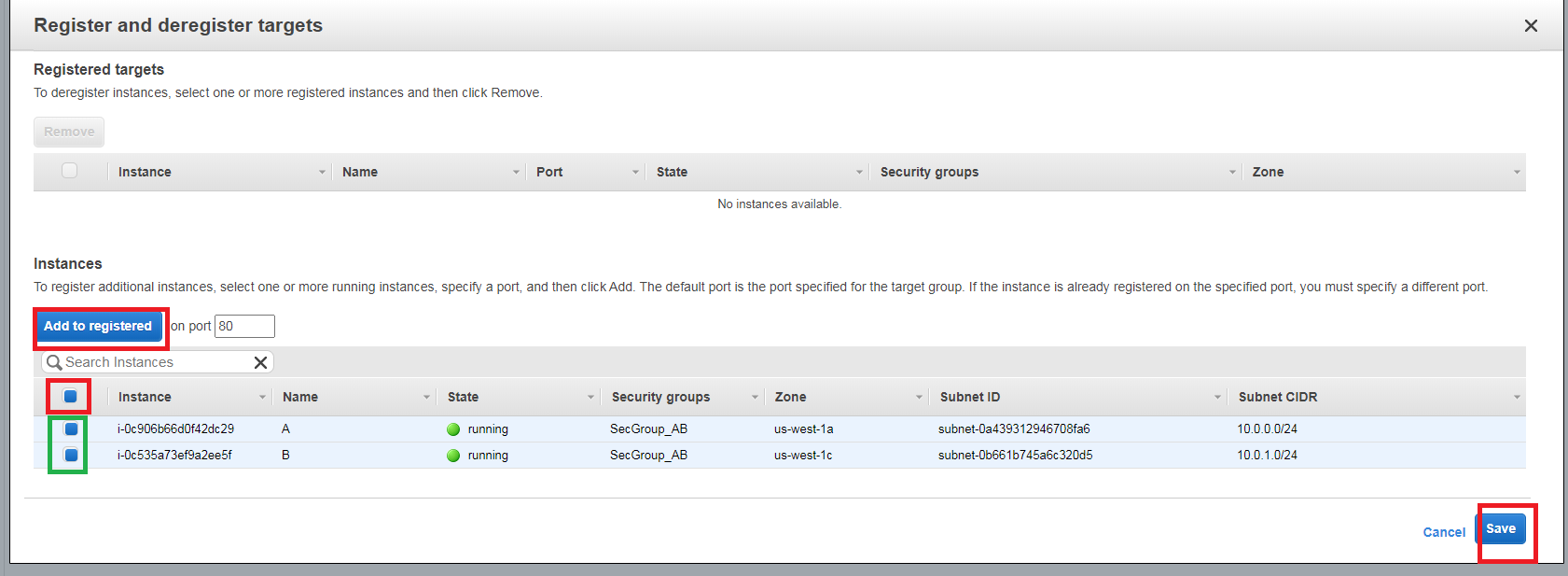


Create Target Group for Web Tier and register instances

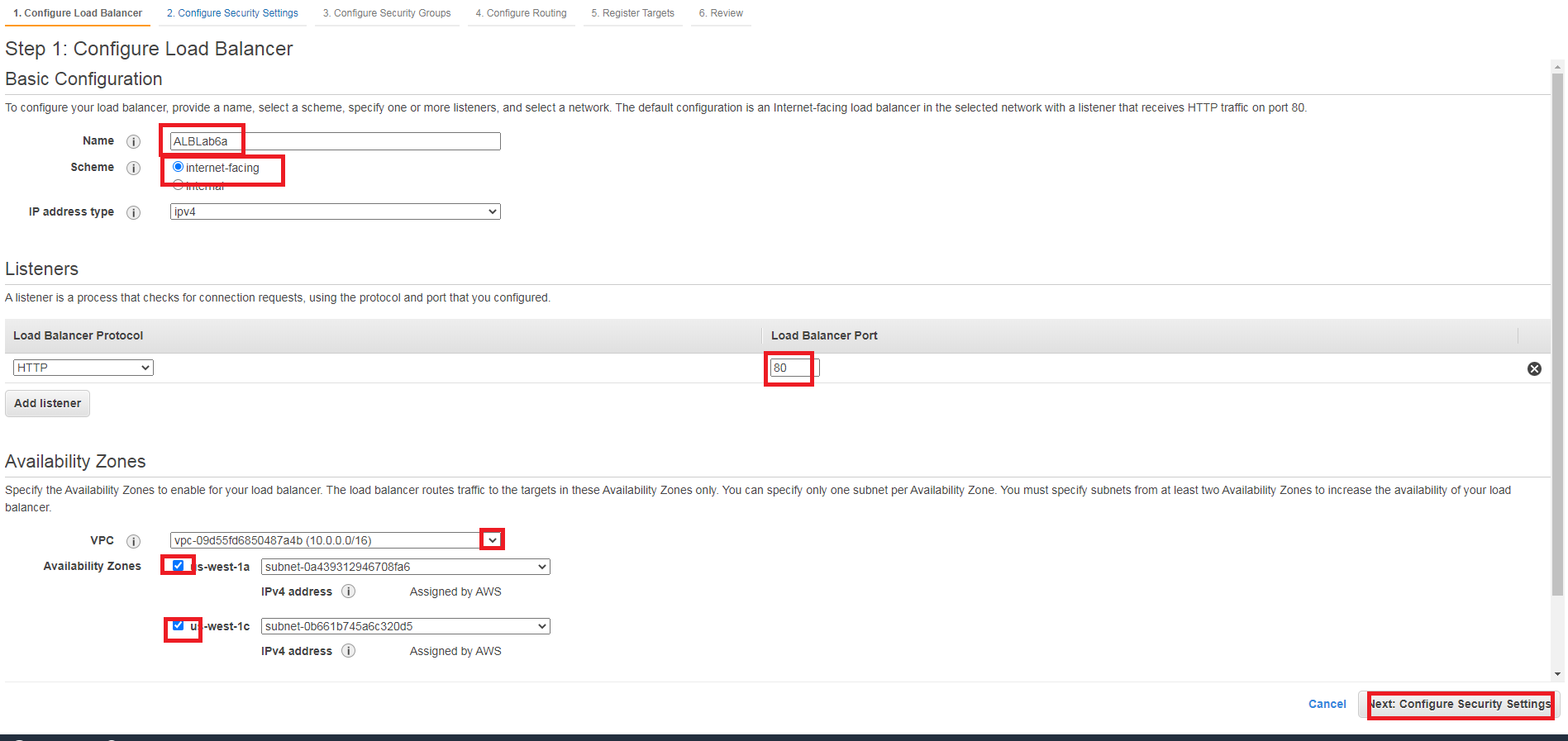
As the same procedure for App Tier.

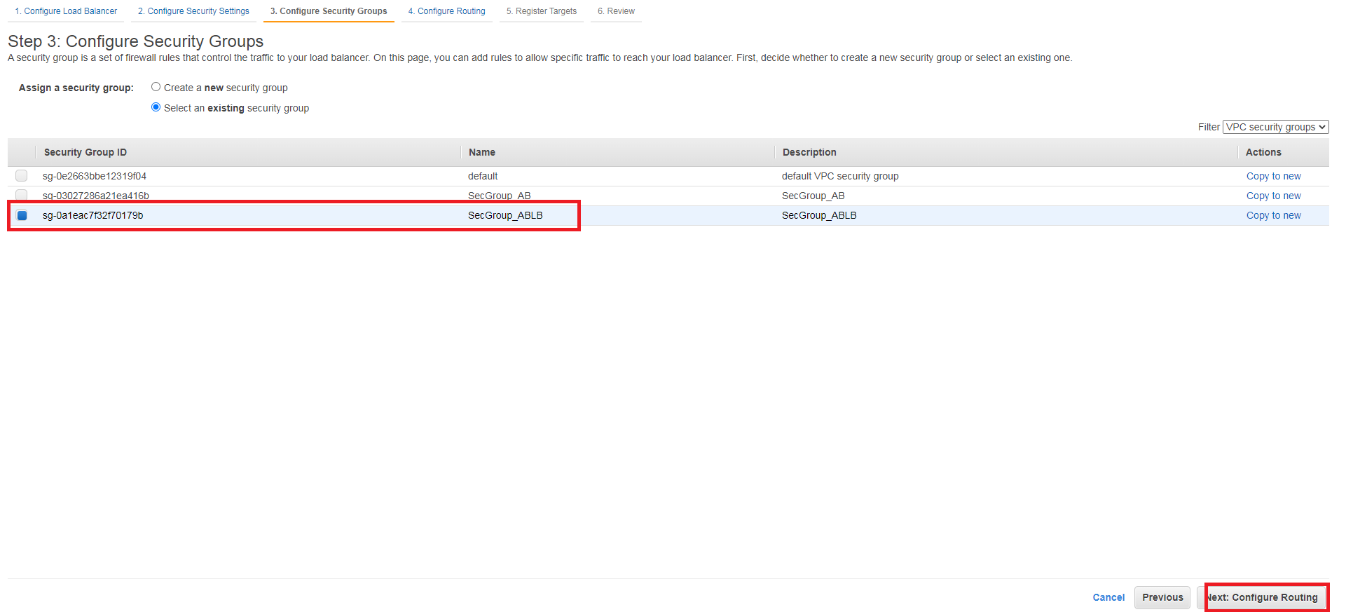


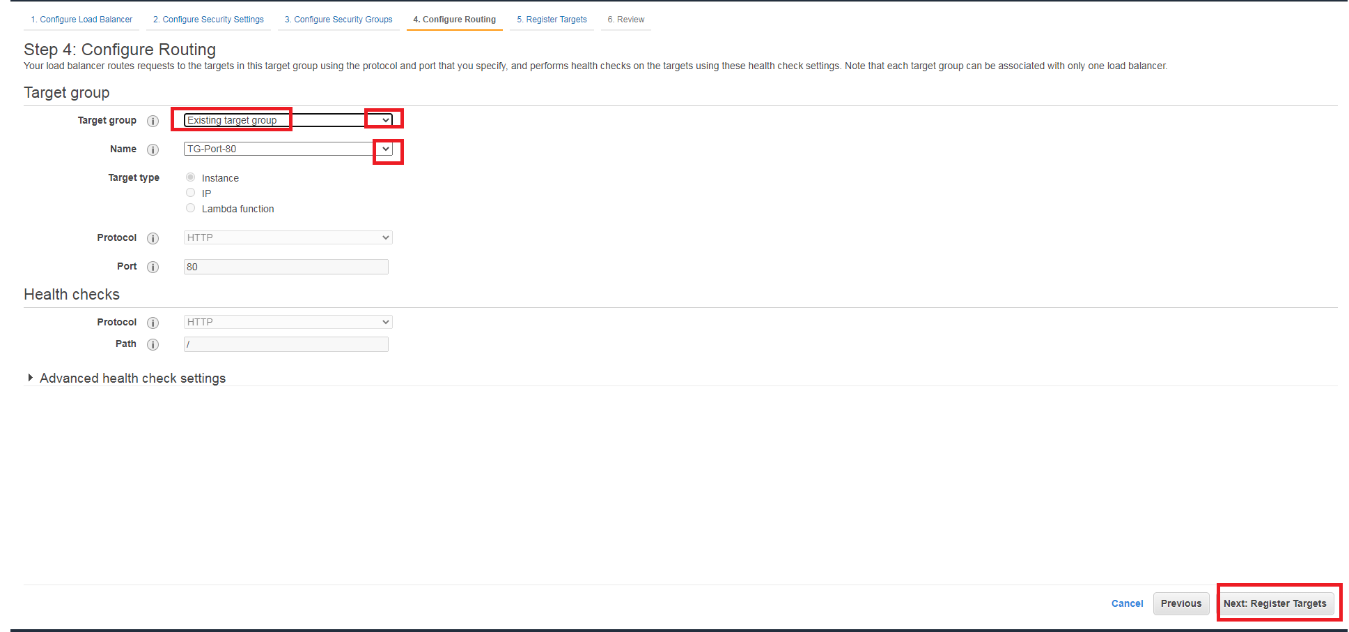


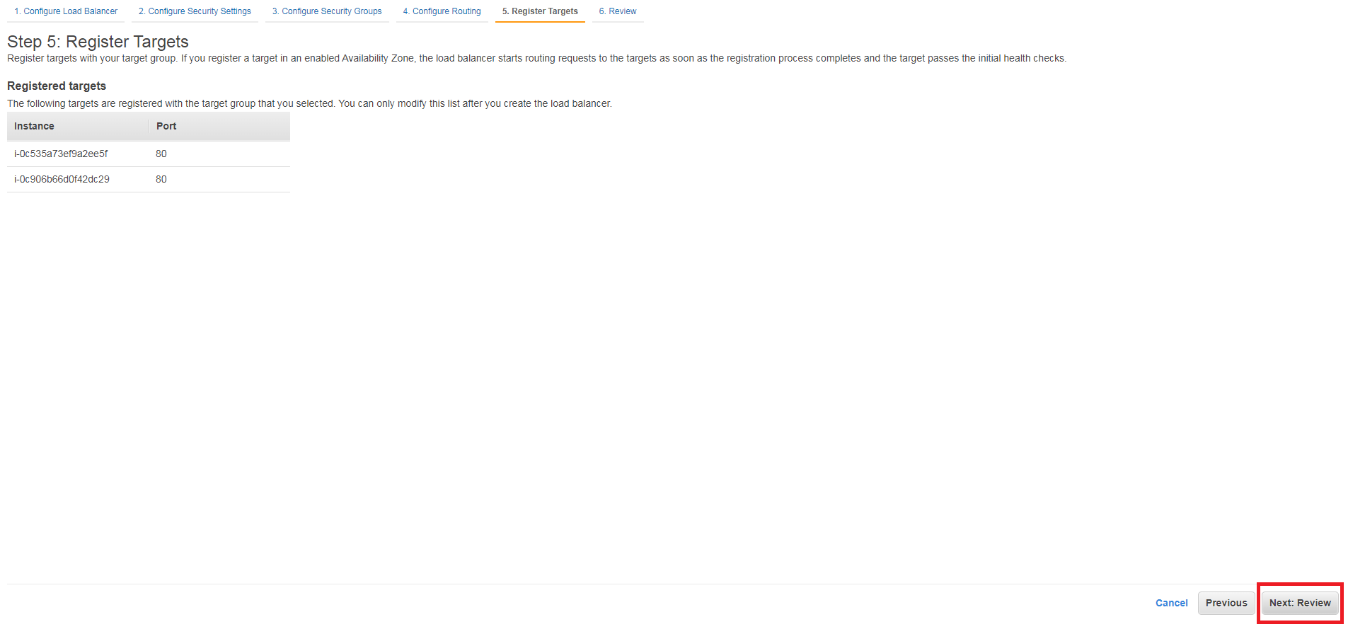


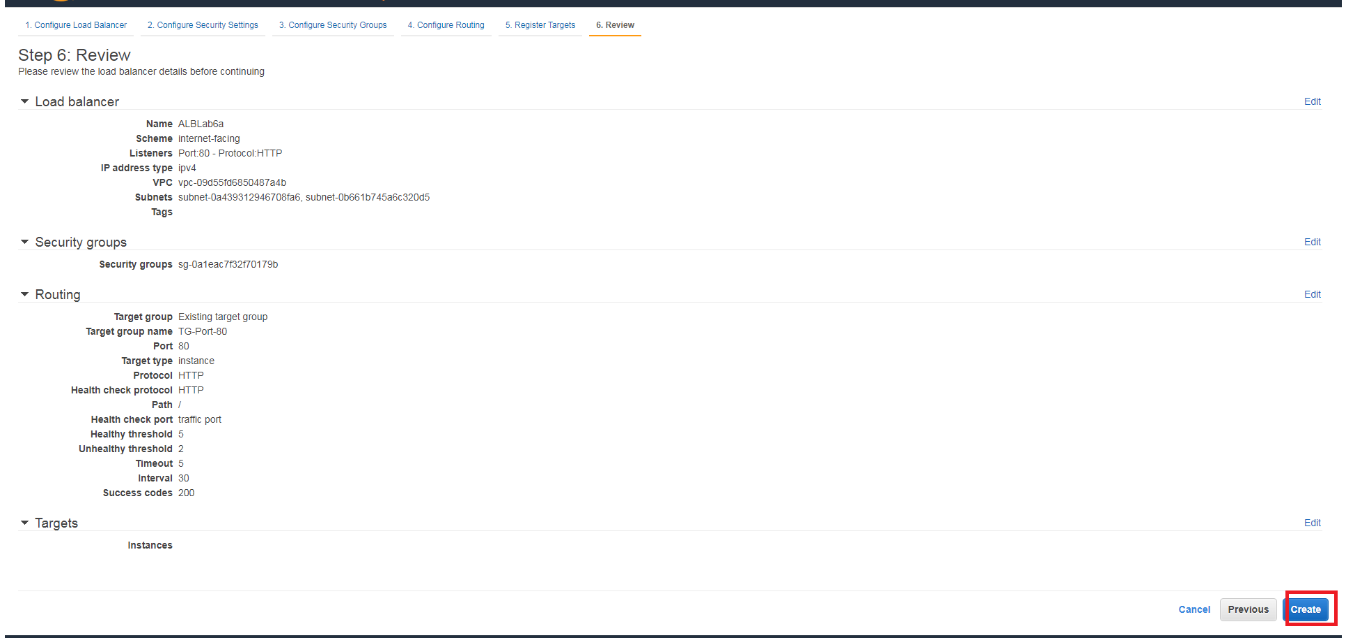
### Create ALB

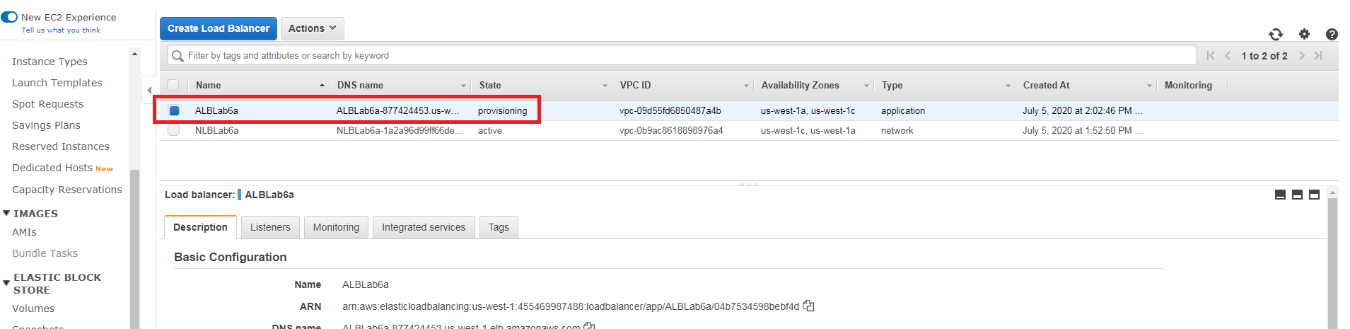


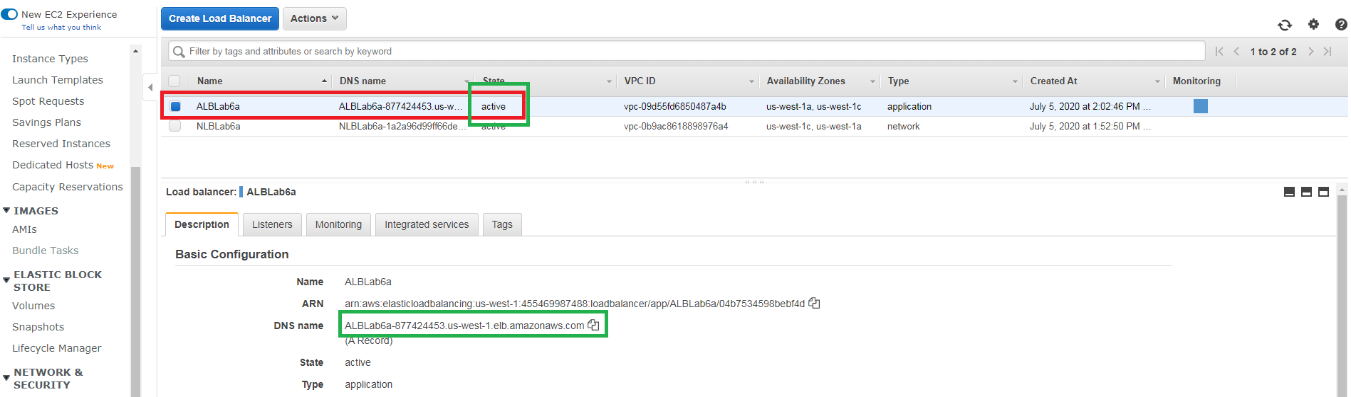




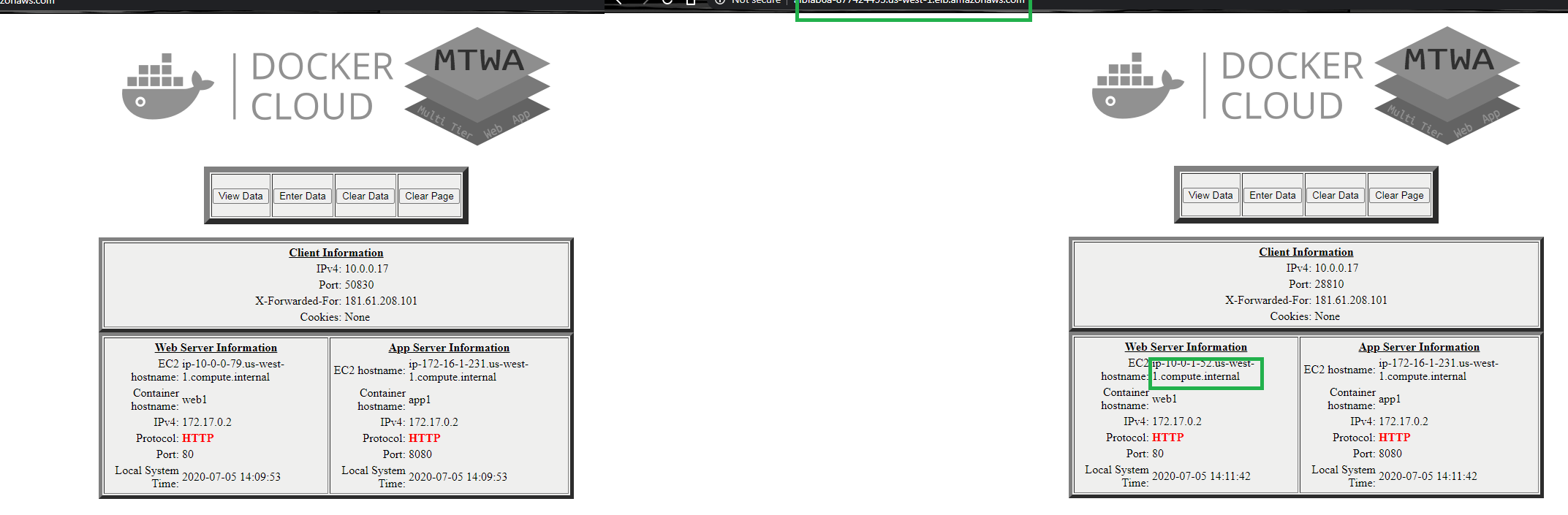








### Review with the browser



You can play with the healthcheck to determine the possible responses to load. Please attend the batch file (Labs6c1.bat), section PLAY WITH HEALTHCHECK.

## Lab 6A using Command Line (Windows)

### Create VPC, Subnet, IGW, NAT, Routing Table, Keys, Sec Groups (Labs4c1) VPC Peering (Labs4c2), Instances on App Layer (Labs5c1)

rem Setear las variables de su grupo. Clase A: 10.x.x.x/8 Clase B: 172.16.x.x a 172.31.x.x

set vpcn\_Mask="10.0.0.0/16"

set pbsn1\_Mask="10.0.0.0/24"

set pbsn2\_Mask="10.0.1.0/24"

set vpcp\_Mask="172.16.0.0/16"

set pbsp1\_Mask="172.16.0.0/24"

set pbsp2\_Mask="172.16.1.0/24"

set pbsn3\_Mask="172.16.2.0/24"

set first\_az="us-west-1a"

set second\_az="us-west-1c"

set instance\_type="t3.small"

rem Crear las VPC y habilitar resolucion DNS

aws ec2 create-vpc --cidr-block %vpcn\_Mask%|jq ".Vpc.VpcId" >tmpFile

set /p vpcn\_Id= < tmpFile

aws ec2 modify-vpc-attribute --vpc-id %vpcn\_Id% --enable-dns-hostnames "{\"Value\":true}"

aws ec2 create-vpc --cidr-block %vpcp\_Mask%|jq ".Vpc.VpcId" >tmpFile

set /p vpcp\_Id= < tmpFile

aws ec2 modify-vpc-attribute --vpc-id %vpcp\_Id% --enable-dns-hostnames "{\"Value\":true}"

rem Crear subredes Publicas

aws ec2 create-subnet --vpc-id %vpcn\_Id% --cidr-block %pbsn1\_Mask% --availability-zone %first\_az%|jq ".Subnet.SubnetId" >tmpFile

set /p pbsn1\_Id= < tmpFile

aws ec2 create-subnet --vpc-id %vpcn\_Id% --cidr-block %pbsn2\_Mask% --availability-zone %second\_az%|jq ".Subnet.SubnetId" >tmpFile

set /p pbsn2\_Id= < tmpFile

rem Permitir que las instancias que se ejecutan en la subredes se hagan publicas. Ver https://docs.aws.amazon.com/vpc/latest/userguide/VPC\_Internet\_Gateway.html

aws ec2 modify-subnet-attribute --subnet-id %pbsn1\_Id% --map-public-ip-on-launch

aws ec2 modify-subnet-attribute --subnet-id %pbsn2\_Id% --map-public-ip-on-launch

rem Crear el Internet Gateway IGW y asignarlo a la VPC

aws ec2 create-internet-gateway|jq ".InternetGateway.InternetGatewayId"  >tmpFile

set /p IGW\_Id= < tmpFile

aws ec2 attach-internet-gateway --vpc-id %vpcn\_Id% --internet-gateway-id %IGW\_Id%

rem Crear tabla de ruteo publica y asignarle IGW como ruta por defecto

aws ec2 create-route-table --vpc-id %vpcn\_Id%|jq ".RouteTable.RouteTableId" >tmpFile

set /p Public\_RT\_Id= < tmpFile

aws ec2 create-route --route-table-id %Public\_RT\_Id% --destination-cidr-block 0.0.0.0/0 --gateway-id %IGW\_Id%

rem Asociar la tabla de ruta a las subredes

aws ec2 associate-route-table  --subnet-id %pbsn1\_Id% --route-table-id %Public\_RT\_Id%

aws ec2 associate-route-table  --subnet-id %pbsn2\_Id% --route-table-id %Public\_RT\_Id%

rem Redes privadas

rem Crear subredes Privadas y la unica publica para el NAT

aws ec2 create-subnet --vpc-id %vpcp\_Id% --cidr-block %pbsp1\_Mask% --availability-zone %first\_az%|jq ".Subnet.SubnetId" >tmpFile

set /p pbsp1\_Id= < tmpFile

aws ec2 create-subnet --vpc-id %vpcp\_Id% --cidr-block %pbsp2\_Mask% --availability-zone %second\_az%|jq ".Subnet.SubnetId" >tmpFile

set /p pbsp2\_Id= < tmpFile

aws ec2 create-subnet --vpc-id %vpcp\_Id% --cidr-block %pbsn3\_Mask% --availability-zone %second\_az%|jq ".Subnet.SubnetId" >tmpFile

set /p pbsn3\_Id= < tmpFile

rem Solicitar una IP Elastica para hacer el Nat Gateway

aws ec2 allocate-address --domain vpc |jq ".AllocationId" >tmpFile

set /p NAT\_EIP= < tmpFile

rem Crear el NAT Gateway, asignarlo a una EIP Anterior.

aws ec2 create-nat-gateway --subnet-id  %pbsn3\_Id% --allocation-id %NAT\_EIP%|jq ".NatGateway.NatGatewayId" >tmpFile

set /p NATGW\_Id= < tmpFile

rem Crear el Internet Gateway IGW y asignarlo a la VPC

aws ec2 create-internet-gateway|jq ".InternetGateway.InternetGatewayId"  >tmpFile

set /p IGW2\_Id= < tmpFile

aws ec2 attach-internet-gateway --vpc-id %vpcp\_Id% --internet-gateway-id %IGW2\_Id%

rem Crear tabla de ruteo publica de la NAT para las redes privadas y asignar el IGW como ruta por defecto. Asociarla

aws ec2 create-route-table --vpc-id %vpcp\_Id%|jq ".RouteTable.RouteTableId" >tmpFile

set /p Public\_Private\_RT\_Id= < tmpFile

aws ec2 create-route --route-table-id %Public\_Private\_RT\_Id% --destination-cidr-block 0.0.0.0/0 --gateway-id %IGW2\_Id%

aws ec2 associate-route-table  --subnet-id %pbsn3\_Id% --route-table-id %Public\_Private\_RT\_Id%

rem Crear tabla de ruteo publica para las redes privadas y asignar el NAT GW como ruta por defecto. Asociarla

aws ec2 create-route-table --vpc-id %vpcp\_Id%|jq ".RouteTable.RouteTableId" >tmpFile

set /p Private\_RT\_Id= < tmpFile

aws ec2 create-route --route-table-id %Private\_RT\_Id% --destination-cidr-block 0.0.0.0/0 --nat-gateway-id %NATGW\_Id%

aws ec2 associate-route-table  --subnet-id %pbsp1\_Id% --route-table-id %Private\_RT\_Id%

aws ec2 associate-route-table  --subnet-id %pbsp2\_Id% --route-table-id %Private\_RT\_Id%

rem Crear y aceptar el VPC Peering

aws ec2 create-vpc-peering-connection --vpc-id %vpcn\_Id% --peer-vpc-id %vpcp\_Id%|jq ".VpcPeeringConnection.VpcPeeringConnectionId" >tmpFile

set /p VPCPeering\_Id= < tmpFile

aws ec2 accept-vpc-peering-connection --vpc-peering-connection-id %VPCPeering\_Id%

rem Agregar las rutas en las 2 tablas de ruteo

aws ec2 create-route --route-table-id %Private\_RT\_Id% --destination-cidr-block %vpcn\_Mask% --vpc-peering-connection-id %VPCPeering\_Id%

aws ec2 create-route --route-table-id %Public\_RT\_Id% --destination-cidr-block %vpcp\_Mask% --vpc-peering-connection-id %VPCPeering\_Id%

rem Crear las llaves para el SSH a las nuevas instancias y convertirlas a PPK para usar Putty ya sea con puttygen o winscp

aws ec2 create-key-pair --key-name Lab6a --query "KeyMaterial" --output text > Lab6a.pem

winscp.com /keygen "Lab6a.pem" /output="Lab6a.ppk"

rem Crear los Security Groups para instancias A y B

aws ec2 create-security-group --group-name "SecGroup\_AB" --description "Security group for Instances A and B" --vpc-id %vpcn\_Id% |jq ".GroupId">tmpFile

set /p SecGroup\_AB\_Id= < tmpFile

aws ec2 authorize-security-group-ingress --group-id %SecGroup\_AB\_Id% --protocol tcp --port 22 --cidr 0.0.0.0/0

aws ec2 authorize-security-group-ingress --group-id %SecGroup\_AB\_Id% --protocol tcp --port 80 --cidr 0.0.0.0/0

aws ec2 authorize-security-group-ingress --group-id %SecGroup\_AB\_Id% --protocol tcp --port 443 --cidr 0.0.0.0/0

rem Crear los Security Groups para instancias C y D

aws ec2 create-security-group --group-name "SecGroup\_CD" --description "Security group for Instances C and D" --vpc-id %vpcp\_Id% |jq ".GroupId">tmpFile

set /p SecGroup\_CD\_Id= < tmpFile

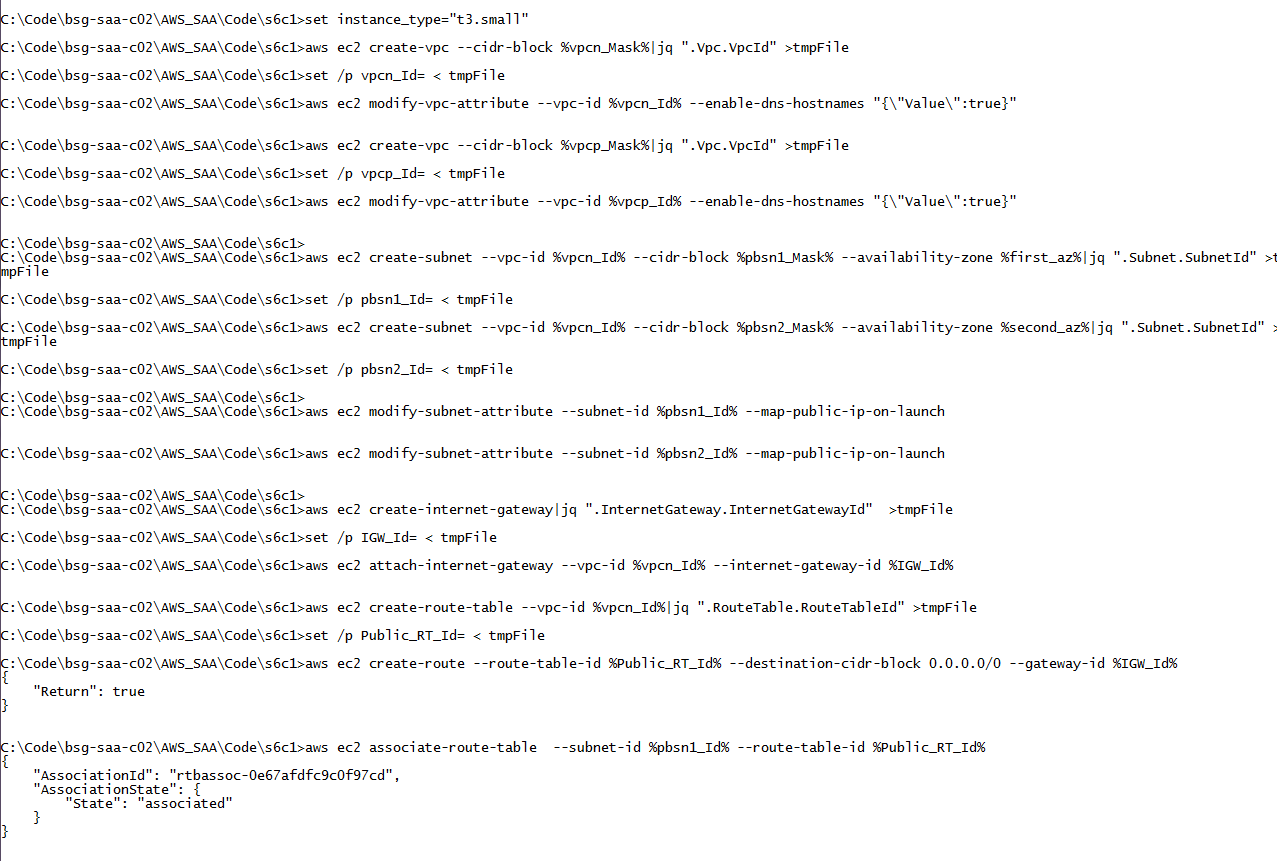
aws ec2 authorize-security-group-ingress --group-id %SecGroup\_CD\_Id% --protocol tcp --port 22 --cidr 0.0.0.0/0

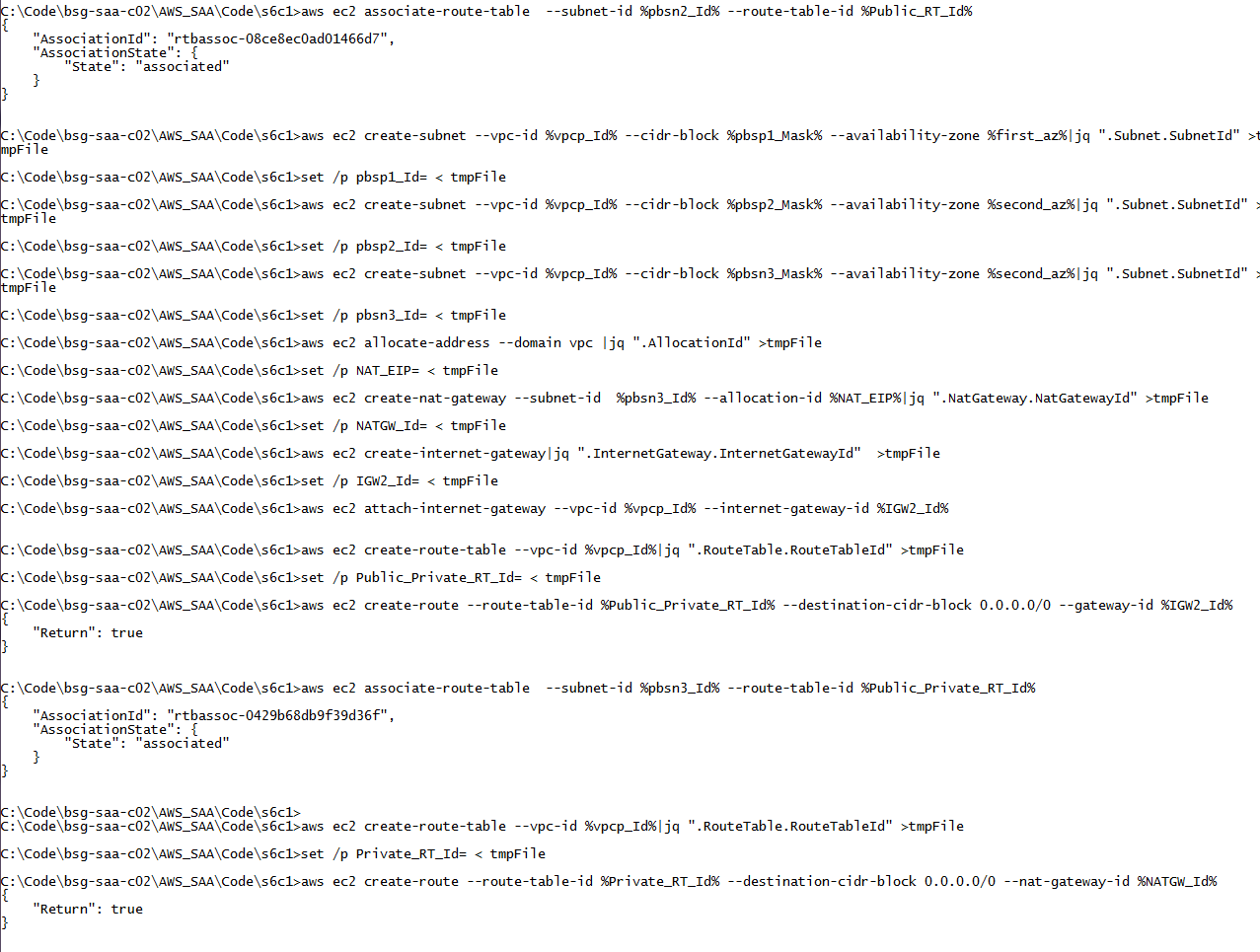
aws ec2 authorize-security-group-ingress --group-id %SecGroup\_CD\_Id% --protocol tcp --port 8080 --cidr 0.0.0.0/0

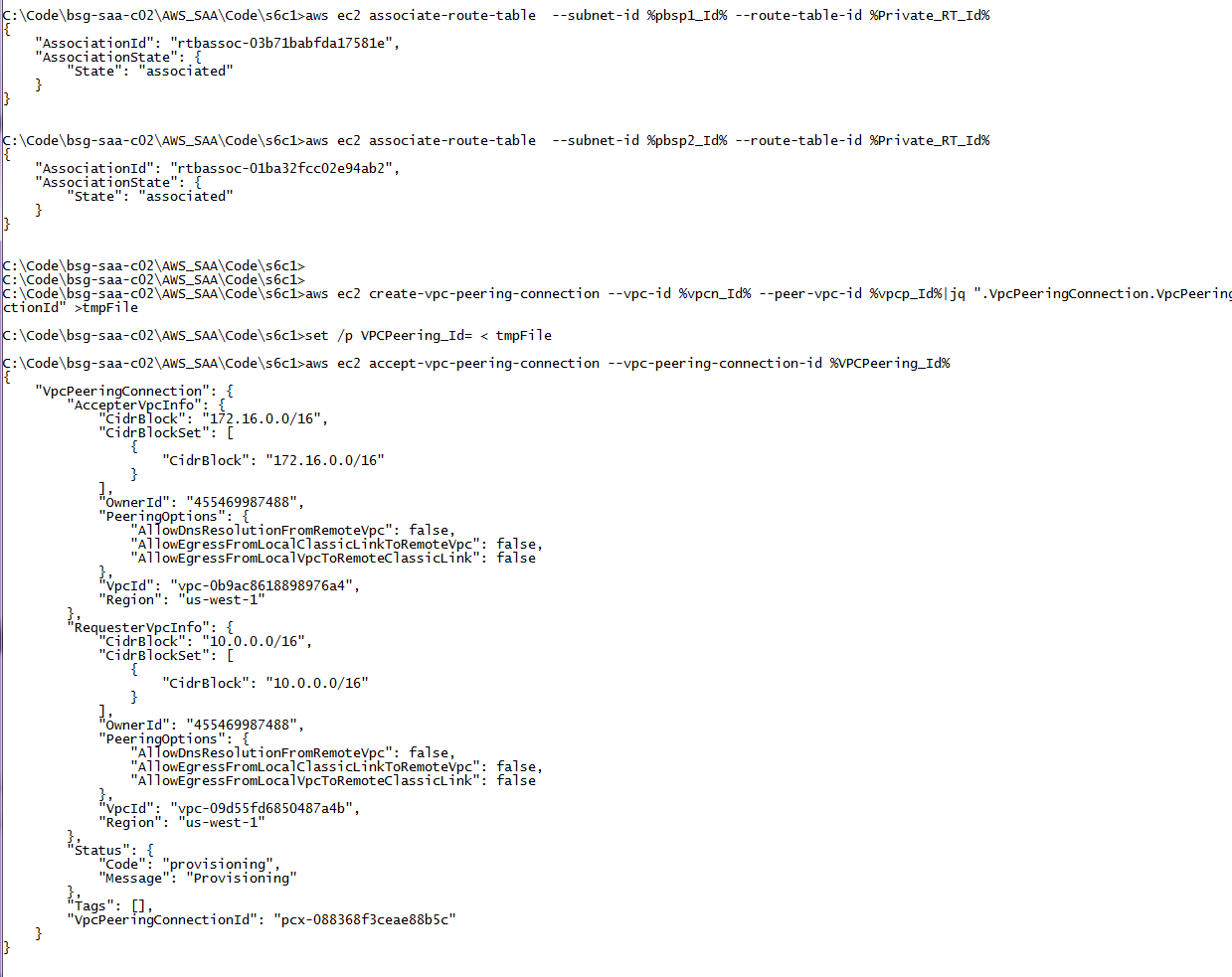
rem AWS sugiere que se tome el AMI Amazon Linux 2 y se instale docker desde linea de comandos: https://docs.aws.amazon.com/AmazonECS/latest/developerguide/docker-basics.html#install\_docker

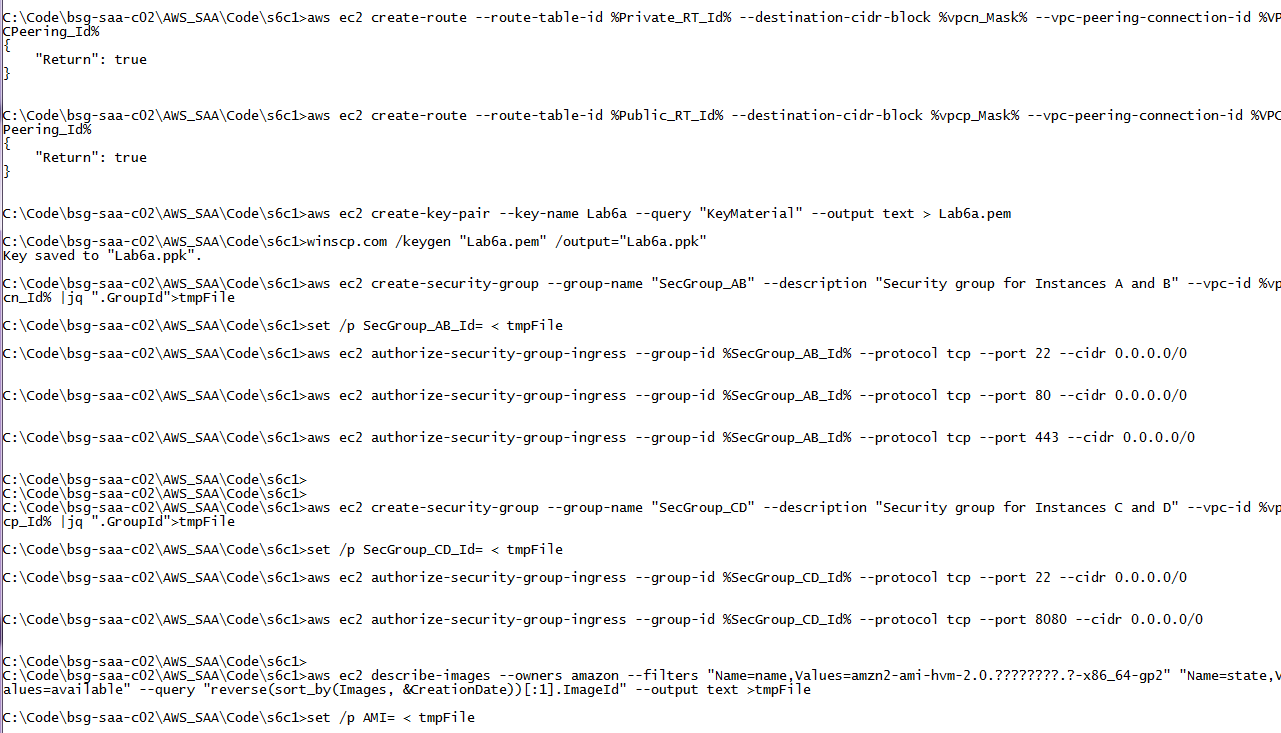
aws ec2 describe-images --owners amazon --filters "Name=name,Values=amzn2-ami-hvm-2.0.????????.?-x86\_64-gp2" "Name=state,Values=available" --query "reverse(sort\_by(Images, &CreationDate))[:1].ImageId" --output text >tmpFile

set /p AMI= < tmpFile









Create instances for App Tier

rem Se solicitan instancias y se adiciona un bootstrap para comprobar que el docker fue instalado

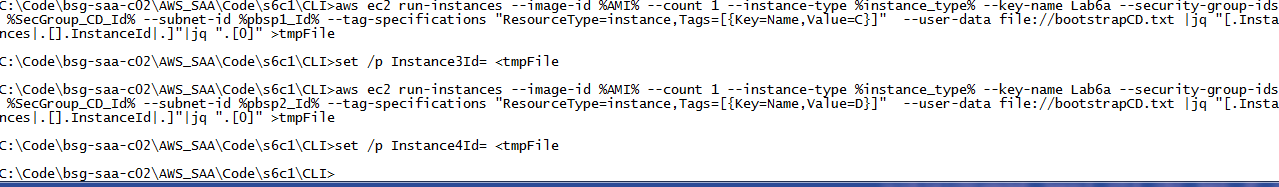
rem Se arrancan con las instancias de backend ya que es necesario modificar posteriormente la capa de presentacion con el nombre del balanceador

aws ec2 run-instances --image-id %AMI% --count 1 --instance-type %instance\_type% --key-name Lab6a --security-group-ids %SecGroup\_CD\_Id% --subnet-id %pbsp1\_Id% --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=C}]"  --user-data file://bootstrapCD.txt |jq "[.Instances|.[].InstanceId|.]"|jq ".[0]" >tmpFile

set /p Instance3Id= <tmpFile

aws ec2 run-instances --image-id %AMI% --count 1 --instance-type %instance\_type% --key-name Lab6a --security-group-ids %SecGroup\_CD\_Id% --subnet-id %pbsp2\_Id% --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=D}]"  --user-data file://bootstrapCD.txt |jq "[.Instances|.[].InstanceId|.]"|jq ".[0]" >tmpFile

set /p Instance4Id= <tmpFile



### Create Sec Groups for NLB, Target Groups, Register Instances and finally, create NLB and its listener

rem Se hace el Balanceador Interno con NLB

rem Crear los Security Group del Balanceador

aws ec2 create-security-group --group-name "SecGroup\_CDNLB" --description "Security group for NLB" --vpc-id %vpcp\_Id% |jq ".GroupId">tmpFile

set /p SecGroup\_CDNLB\_Id= < tmpFile

aws ec2 authorize-security-group-ingress --group-id %SecGroup\_CDNLB\_Id% --protocol tcp --port 8080 --cidr 0.0.0.0/0

rem Permitir que el balanceador pueda ver las instancias, sirve para el balanceo y el healthcheck

aws ec2 authorize-security-group-ingress --group-id %SecGroup\_CD\_Id% --protocol tcp --port 8080 --source-group %SecGroup\_CDNLB\_Id%

rem Crear los target groups y registrar las instancias a los mismos en cada puerto

aws elbv2 create-target-group --name TG-Port-8080 --protocol TCP --port 8080 --target-type instance --vpc-id %vpcp\_Id% |jq ".TargetGroups[].TargetGroupArn" >tmpFile

set /p TG8080\_ARN= < tmpFile

aws elbv2 register-targets --target-group-arn %TG8080\_ARN% --targets Id=%Instance3Id% Id=%Instance4Id%

rem Crear el NLB

aws elbv2 create-load-balancer --name NLBLab6a --scheme internal --type network --subnets %pbsp1\_Id% %pbsp2\_Id%  >tmpFile2

cat tmpFile2|jq ".LoadBalancers[].LoadBalancerArn" >tmpFile

set /p NLB\_ARN= < tmpFile

cat tmpFile2|jq ".LoadBalancers[].DNSName" >tmpFile

set /p NLB\_DNSName= < tmpFile

del tmpFile2

rem Se crea el Listener para Puerto 8080

aws elbv2 create-listener --load-balancer-arn %NLB\_ARN% --protocol TCP --port 8080 --default-actions Type=forward,TargetGroupArn=%TG8080\_ARN%|jq ".Listeners[].ListenerArn" >tmpFile

set /p LST8080\_ARN= < tmpFile



### Create instances for Web Tier using Internal balancer, Sec Groups for ALB, Target Groups for ALB, ALB and ALB’ Listener

rem Se crean las instancias de la capa superiro modificando el bootstrap script para que tomen el balanceador interno

echo El nombre del balanceador interno es %NLB\_DNSName%



**You have to modify the bootstrapAB.txt file with the internal NLB**

rem Antes de lanzarla se tiene que modificar el archivo bootrstrapAB.txt con el nombre del balanceador interno que esta en la variable %NLB\_DNSName% agregarle el puerto 8080

aws ec2 run-instances --image-id %AMI% --count 1 --instance-type %instance\_type% --key-name Lab6a --security-group-ids %SecGroup\_AB\_Id% --subnet-id %pbsn1\_Id% --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=A}]"  --user-data file://bootstrapAB.txt |jq "[.Instances|.[].InstanceId|.]"|jq ".[0]" >tmpFile

set /p Instance1Id= <tmpFile

aws ec2 run-instances --image-id %AMI% --count 1 --instance-type %instance\_type% --key-name Lab6a --security-group-ids %SecGroup\_AB\_Id% --subnet-id %pbsn2\_Id% --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=B}]"  --user-data file://bootstrapAB.txt |jq "[.Instances|.[].InstanceId|.]"|jq ".[0]" >tmpFile

set /p Instance2Id= <tmpFile

rem Crear los Security Group del Balanceador

aws ec2 create-security-group --group-name "SecGroup\_ABLB" --description "Security group for ALB" --vpc-id %vpcn\_Id% |jq ".GroupId">tmpFile

set /p SecGroup\_ABLB\_Id= < tmpFile

aws ec2 authorize-security-group-ingress --group-id %SecGroup\_ABLB\_Id% --protocol tcp --port 80 --cidr 0.0.0.0/0

aws ec2 authorize-security-group-ingress --group-id %SecGroup\_ABLB\_Id% --protocol tcp --port 443 --cidr 0.0.0.0/0

rem Permitir que el balanceador pueda ver las instancias, sirve para el balanceo y el healthcheck

aws ec2 authorize-security-group-ingress --group-id %SecGroup\_AB\_Id% --protocol tcp --port 80 --source-group %SecGroup\_ABLB\_Id%

aws ec2 authorize-security-group-ingress --group-id %SecGroup\_AB\_Id% --protocol tcp --port 443 --source-group %SecGroup\_ABLB\_Id%

rem Crear los target groups y registrar las instancias a los mismos en cada puerto

aws elbv2 create-target-group --name TG-Port-80 --protocol HTTP --port 80 --target-type instance --vpc-id %vpcn\_Id% |jq ".TargetGroups[].TargetGroupArn" >tmpFile

set /p TG80\_ARN= < tmpFile

aws elbv2 register-targets --target-group-arn %TG80\_ARN% --targets Id=%Instance1Id% Id=%Instance2Id%

rem Crear el ALB

aws elbv2 create-load-balancer --name ALBLab6a --subnets %pbsn1\_Id% %pbsn2\_Id% --security-groups %SecGroup\_ABLB\_Id% >tmpFile2

cat tmpFile2|jq ".LoadBalancers[].LoadBalancerArn" >tmpFile

set /p LB\_ARN= < tmpFile

cat tmpFile2|jq ".LoadBalancers[].DNSName" >tmpFile

set /p LB\_DNSName= < tmpFile

del tmpFile2

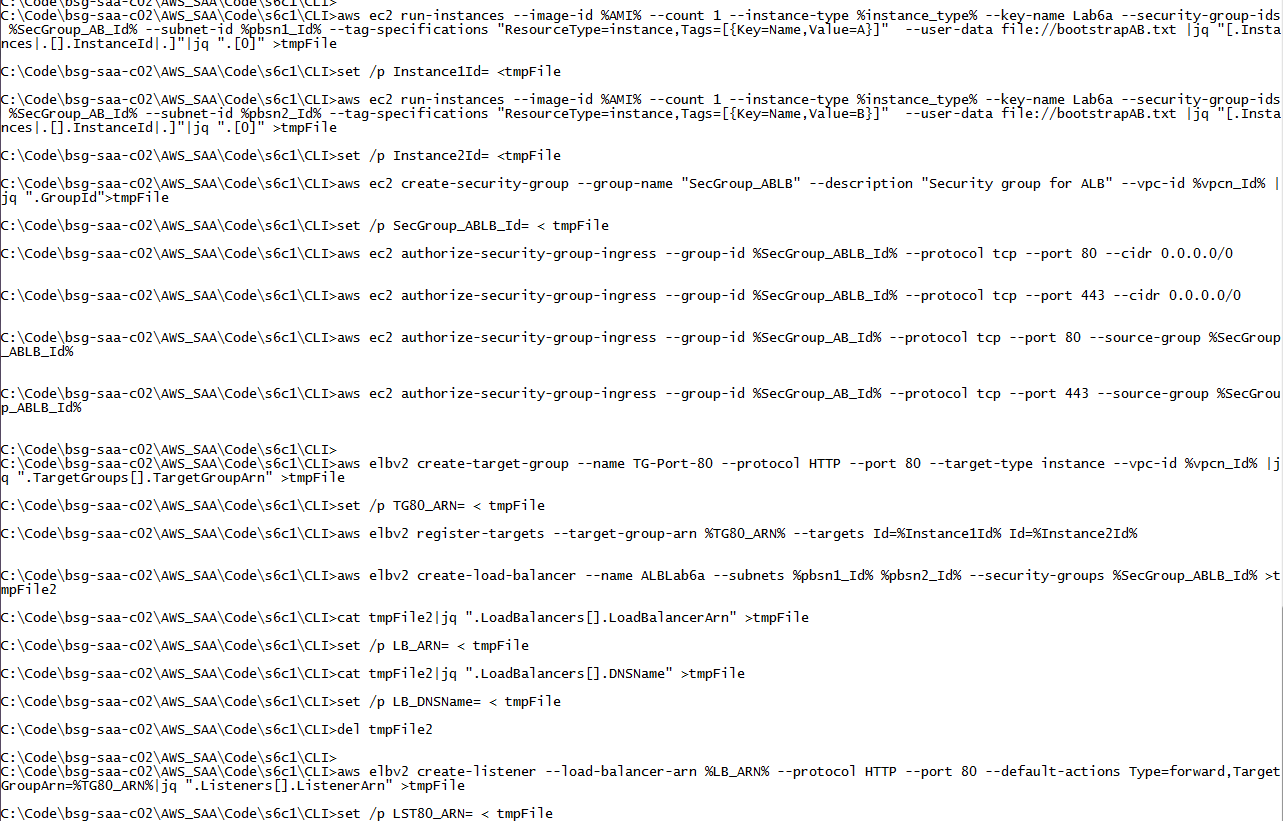
rem Se crea el Listener para Puerto 80

aws elbv2 create-listener --load-balancer-arn %LB\_ARN% --protocol HTTP --port 80 --default-actions Type=forward,TargetGroupArn=%TG80\_ARN%|jq ".Listeners[].ListenerArn" >tmpFile

set /p LST80\_ARN= < tmpFile

rem Se prueba que el ALB llegue al target group desde un navegador despues de que el estado del ALB este en active

echo Para navegar a %LB\_DNSName%

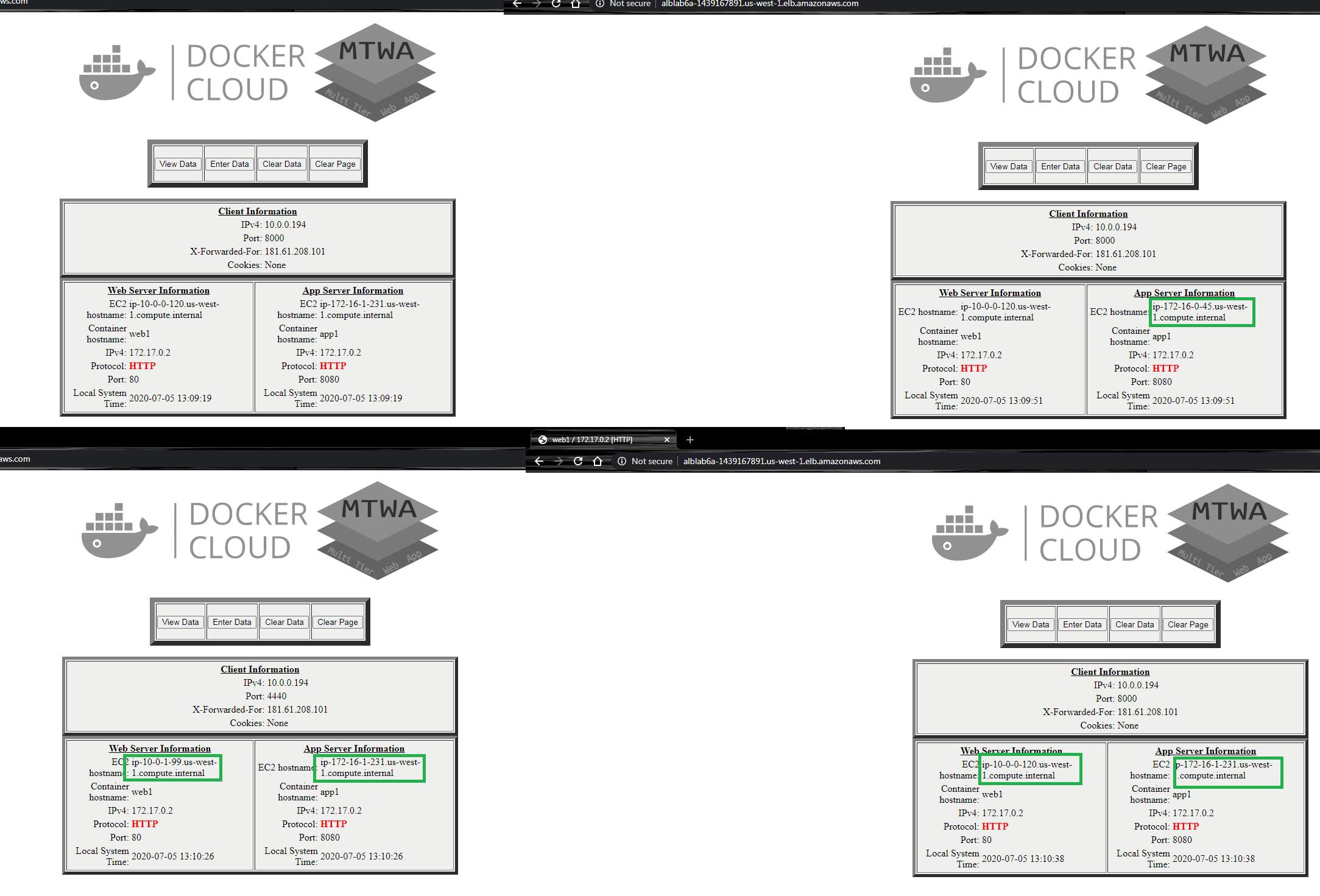


You have to check if its ALB is on Running State on Web Console and then you can go using browser.



### Review using browser

Using the browser make several refreshs and detect the changes of the Internal IP on Web and App Layers.



rem \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* PLAY WITH HEALTHCHECK \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

rem Modifica Health Check para NLB para deshabilitarlo o ponerlo en modo desconectado aleatoriamente

aws elbv2 modify-target-group --target-group-arn %TG80\_ARN% --health-check-protocol HTTP --health-check-port 80 --health-check-path "/longdelay.py" >tmpFile

aws elbv2 describe-target-groups --target-group-arns %TG80\_ARN%

rem Si ambos maquinas fallan, la documentacion dice que envia el paquete sin importar el estado asi que es mejor volver ir a una maquina A o B y detener el docker.

rem La documentacion esta en https://docs.aws.amazon.com/elasticloadbalancing/latest/application/target-group-health-checks.html y dice "If a target group contains only unhealthy registered targets, the load balancer routes requests to all those targets, regardless of their health status."

aws ec2 describe-instances --filters "Name=tag:Name,Values=A"  "Name=instance-state-name,Values=running"  |  jq -r  ".Reservations[] | .Instances[]|.PublicIpAddress" >tmpFile

set /p A\_IP= < tmpFile

putty -i "Lab6a.ppk" ec2-user@%A\_IP%

rem Ir al putty

docker ps -a

docker stop $(docker ps -aq)

docker rm $(docker ps -aq)

rem Para hacer estas revisiones es mejor ejecutarlo con la siguiente configuracion del healthcheck:

rem Healthy threshold y Unhealthy threshold   2

rem Timeout 17s

rem Interval 20s

rem Revisar el balanceador que solo lo envie a la instancia correcta...luego lo volvemos a ejecutar la ultima linea del bootstrapAB.txt en el docker y vamos al docker y reemplazamos el index.py por longdelay.py

docker run -d -p 80:80 -p 443:443 -e APPSERVER="http://NLBLab6a-a9219a23041903f7.elb.us-east-1.amazonaws.com:8080" -e TZ=America/Bogota -h web1 fmorenod81/mtwa:web

docker exec -ti 680 sh

cd /var/www/html/appdemo

mv longdelay.py index.py

rem Alli comprobamos que la maquina funciona sin embargo esta deshabilitada en el balanceador

mv shortdelay.py index.py

rem Volvemos a ejecutar el balanceador y esperamos que se ejecute el healthy a la instancia

rem Salir del putty

## Clean Resources

VPC: Peering Connections

EC2: Instances

EC2: Load LoadBalancers

EC2: TargetGroups

VPC: NAT gateway

VPC: EIP

EC2: Keypair

EC2: Security Group

VPC: VPC