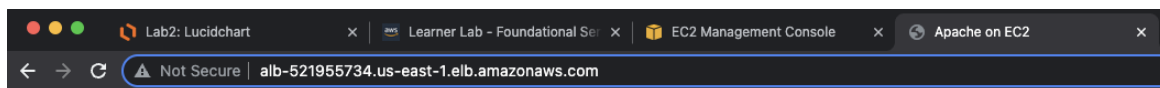


# Lab2: Building Highly Available Website in Custom VPC

The objective of this assignment is to build out a highly available website according to AWS best practices and security principals.

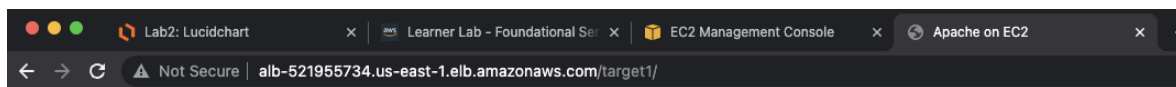
The website serves 3 different webpages based on the path specified in the URL. These webpages are served by 3 different target groups.

The goal is to have the message below served by default target group at <http://<ALB DNS>/>



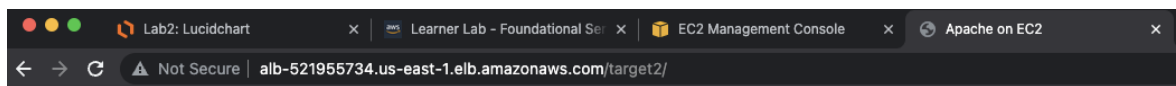
**Welcome to the default page!**

Target group 1 will serve the below message at <http://<ALB DNS>/target1/>



**Welcome to the TARGET1 page!**

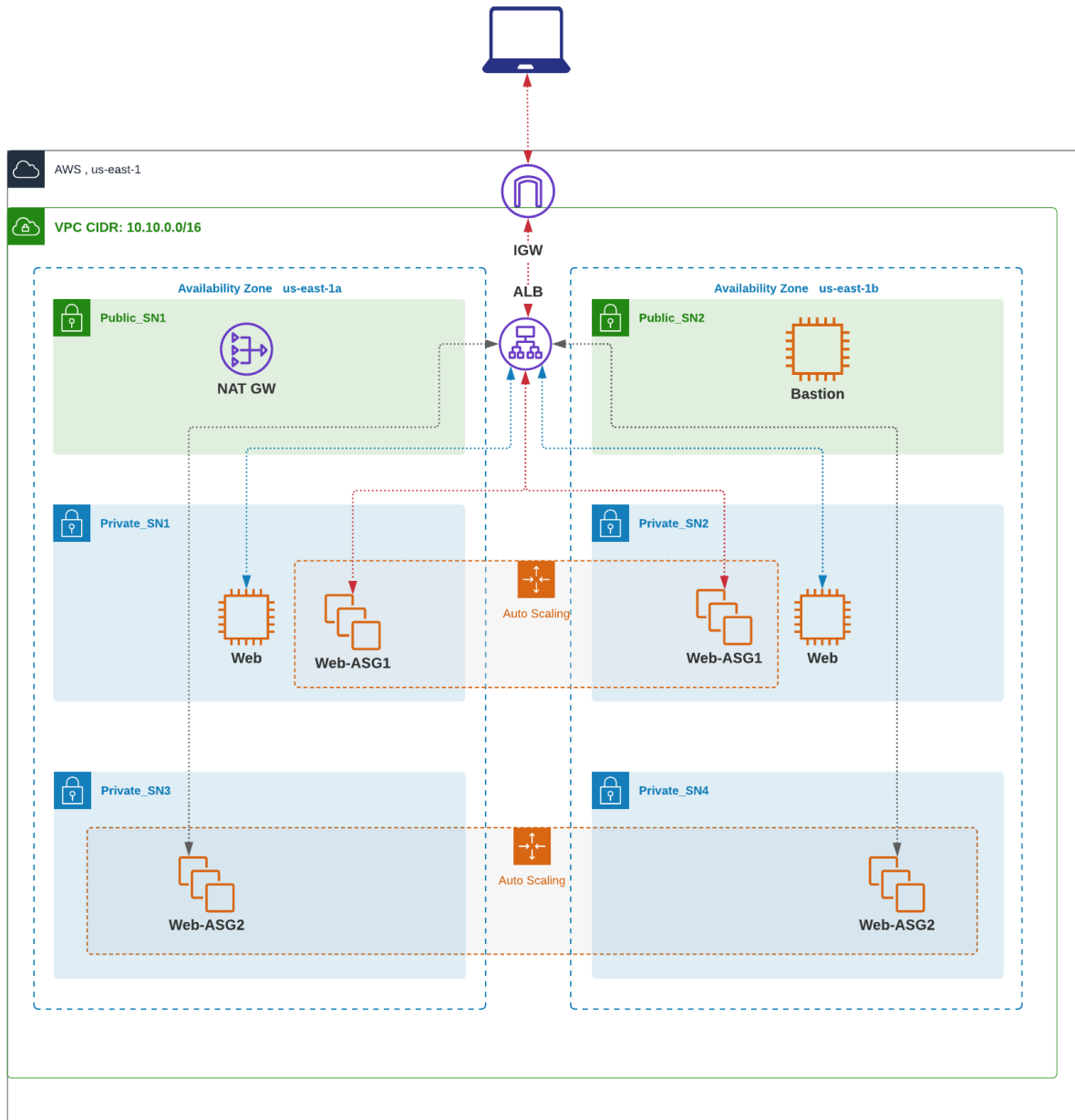
Target group 2 will serve the below message at <http://<ALB DNS>/target2/>



**Welcome to the TARGET2 page!**

The EC2 instances are deployed in private subnets making them safe from malicious attacks from the Internet. The website is externally accessible through Application Load Balancer (ALB) deployed into public subnets. In addition to public/private subnets separation, Security Groups are designed in a way that ensures network security.

The diagram below depicts the main infrastructure components and web traffic flows that are in scope of this assignment. Please note that some infrastructure pieces (such as route tables, subnet CIDR ranges and SG) are missing from the diagram below. Those are part of your assignment and are left for you to design.



**NOTE I Switched from Linux VM to my Windows Edge PC the last session before handing it in.**

### PROCESS OF CREATING THE ABOVE IMAGE!

Creating the Custom VPC

```
ddd_v1_w_MVbI_882554@runweb41469:~$ aws ec2 create-vpc --cidr-block 10.1.0.0/16
--query Vpc.VpcId --output text
vpc-0b841e81532030523
```

The range was give to us so that is what I used

Subnets:

Subnet Name	Availability Zone	Number of IP addresses	Private	Public
Public_SN1	us-east-1c	256 /24		X
Public_SN2	us-east-1b	256 /24		X
Private_SN1	us-east-1c	512 /23	X	
Private_SN2	us-east-1b	512 /23	X	
Private_SN3	us-east-1c	128 /25	X	
Private_SN4	us-east-1b	128 /25	X	

Public SN1 ID: (10.1.10.0/24)

Public SN2 ID: (10.1.11.0/24)

Private SN1 ID: (10.1.20.0/23)

Private Sn2 ID: (10.1.22.0/23)

Private SN3 ID: (10.1.30.0/25)

Private Sn4 ID: (10.1.30.128/25)

I chose to use the closest IP I could each time that I switched IP ranges. For the public IP blocks I used 10&11 as they require only one octet. I then used 20&21 for The first private SN and 22&23 for the second private subnet. This is required because you need more then 8 bits, and then I I was creating a private block

Creating Private SN1

```
ddd_v1_w_MvBI_882554@runweb41660:~$ aws ec2 create-subnet --vpc-id ${VPC_ID} --
cidr-block 10.1.10.0/24 --availability-zone us-east-1a
{
  "Subnet": {
    "AvailabilityZone": "us-east-1a",
    "AvailabilityZoneId": "use1-az1",
    "AvailableIpAddressCount": 251,
    "CidrBlock": "10.1.10.0/24",
    "DefaultForAz": false,
    "MapPublicIpOnLaunch": false,
    "State": "available",
    "SubnetId": "subnet-075f43428e2e523ee",
    "VpcId": "vpc-0749c147b22850616",
    "OwnerId": "305139984171",
    "AssignIpv6AddressOnCreation": false,
    "Ipv6CidrBlockAssociationSet": [],
    "SubnetArn": "arn:aws:ec2:us-east-1:305139984171:subnet/subnet-075f4342
8e2e523ee"
  }
}
```

```
ddd_v1_w_MvBI_882554@runweb41481:~$ export private_SN1=subnet-0784d38fd85cce035
```

```
ddd_v1_w_MvBI_882554@runweb41481:~$ aws ec2 create-tags --resources ${private_SN1} --tags Key=Name,Value=VPC-Luka-Private-SN-1
```

I created the IP and then exported it and tagged it with a name so its easy to seePrivate SN2

```
ddd_v1_w_MvBI_882554@runweb41668:~$ aws ec2 create-subnet --vpc-id ${VPC_ID} --cidr-block 10.1.11.0/24 --availability-zone us-east-1b
{
  "Subnet": {
    "AvailabilityZone": "us-east-1b",
    "AvailabilityZoneId": "use1-az2",
    "AvailableIpAddressCount": 251,
    "CidrBlock": "10.1.11.0/24",
    "DefaultForAz": false,
    "MapPublicIpOnLaunch": false,
    "State": "available",
    "SubnetId": "subnet-08e1e6bbde6a8a9d5",
    "VpcId": "vpc-0749c147b22850616",
    "OwnerId": "305139984171",
    "AssignIpv6AddressOnCreation": false,
    "Ipv6CidrBlockAssociationSet": [],
    "SubnetArn": "arn:aws:ec2:us-east-1:305139984171:subnet/subnet-08e1e6bbde6a8a9d5"
  }
}
```

```
ddd_v1_w_MvBI_882554@runweb41481:~$ aws ec2 create-tags --resources ${private_SN2} --tags Key=Name,Value=VPC-Luka-Private-SN-2
```

```
ddd_v1_w_MvBI_882554@runweb41481:~$ export private_SN2=subnet-0c5e06b6407e4e2dd
```

I created the IP and then exported it and tagged it with a name so its easy to see

### Private SN3

```
ddd_v1_w_MvBI_882554@runweb41542:~$ aws ec2 create-subnet --vpc-id ${VPC_ID} --cidr-block 10.1.30.0/25 --availability-zone us-east-1a
{
  "Subnet": {
    "AvailabilityZone": "us-east-1a",
    "AvailabilityZoneId": "use1-az1",
    "AvailableIpAddressCount": 123,
    "CidrBlock": "10.1.30.0/25",
    "DefaultForAz": false,
    "MapPublicIpOnLaunch": false,
    "State": "available",
    "SubnetId": "subnet-0d3d2a6927f916d61",
    "VpcId": "vpc-05db83923efbda8a9",
    "OwnerId": "305139984171",
    "AssignIpv6AddressOnCreation": false,
    "Ipv6CidrBlockAssociationSet": [],
    "SubnetArn": "arn:aws:ec2:us-east-1:305139984171:subnet/subnet-0d3d2a6927f916d61"
  }
}
```

```
ddd_v1_w_MvBI_882554@runweb41481:~$ export private_SN3=subnet-009ecd978e375a438
```

## Private SN4

```
ddd_v1_w_MvBI_882554@runweb41481:~$ aws ec2 create-subnet --vpc-id ${VPC_ID} --cidr-block 10.1.132.128/25 --availability-zone us-east-1b
{
  "Subnet": {
    "AvailabilityZone": "us-east-1b",
    "AvailabilityZoneId": "use1-az2",
    "AvailableIpAddressCount": 123,
    "CidrBlock": "10.1.132.128/25",
    "DefaultForAz": false,
    "MapPublicIpOnLaunch": false,
    "State": "available",
    "SubnetId": "subnet-8a9548b3fb4429b1c",
    "VpcId": "vpc-8b841e81532039523",
    "OwnerId": "395139984171",
    "AssignIpv6AddressOnCreation": false,
    "Ipv6CidrBlockAssociationSet": [],
    "SubnetArn": "arn:aws:ec2:us-east-1:395139984171:subnet/subnet-8a9548b3fb4429b1c"
  }
}
```

```
ddd_v1_w_MvBI_882554@runweb41481:~$ export private_SN4=subnet-8a9548b3fb4429b1c
```

## Associating a tag with SN3 & SN4

```
ddd_v1_w_MvBI_882554@runweb41481:~$ aws ec2 create-tags --resources ${private_SN3} --tags Key=Name,Value=VPC-Luka-Private-SN-3
ddd_v1_w_MvBI_882554@runweb41481:~$ aws ec2 create-tags --resources ${private_SN4} --tags Key=Name,Value=VPC-Luka-Private-SN-4
ddd_v1_w_MvBI_882554@runweb41481:~$
```

## Creating Public SN1

```
ddd_v1_w_MvBI_882554@runweb41542:~$ aws ec2 create-subnet --vpc-id ${VPC_ID} --cidr-block 10.1.10.0/24 --availability-zone us-east-1b
{
  "Subnet": {
    "AvailabilityZone": "us-east-1b",
    "AvailabilityZoneId": "use1-az2",
    "AvailableIpAddressCount": 251,
    "CidrBlock": "10.1.10.0/24",
    "DefaultForAz": false,
    "MapPublicIpOnLaunch": false,
    "State": "available",
    "SubnetId": "subnet-8b98a21a843b5bd24",
    "VpcId": "vpc-05db83923efbda8a9",
    "OwnerId": "395139984171",
    "AssignIpv6AddressOnCreation": false,
    "Ipv6CidrBlockAssociationSet": [],
    "SubnetArn": "arn:aws:ec2:us-east-1:395139984171:subnet/subnet-8b98a21a843b5bd24"
  }
}
```



## Creating Public SN2

```
ddd_v1_w_MVbI_882554@runweb41542:~$ aws ec2 create-subnet --vpc-id ${VPC_ID} --cidr-block 10.1.11.0/24 --availability-zone us-east-1a
{
  "Subnet": {
    "AvailabilityZone": "us-east-1a",
    "AvailabilityZoneId": "use1-az1",
    "AvailableIpAddressCount": 251,
    "CidrBlock": "10.1.11.0/24",
    "DefaultForAz": false,
    "MapPublicIpOnLaunch": false,
    "State": "available",
    "SubnetId": "subnet-0e1287c0a32c1956e",
    "VpcId": "vpc-05db83923efbda8a9",
    "OwnerId": "305139984171",
    "AssignIpv6AddressOnCreation": false,
    "Ipv6CidrBlockAssociationSet": [],
    "SubnetArn": "arn:aws:ec2:us-east-1:305139984171:subnet/subnet-0e1287c0a32c1956e"
  }
}
```

## Setting all subnet tags

```
ddd_v1_w_MVbI_882554@runweb41542:~$ export Public_SN1=subnet-06685707d39fe1461
ddd_v1_w_MVbI_882554@runweb41542:~$ export Public_SN2=subnet-06fd20f33c75df3b0
ddd_v1_w_MVbI_882554@runweb41542:~$ export Private_SN1=subnet-0d3d2a6927f916d61
ddd_v1_w_MVbI_882554@runweb41542:~$ export Private_SN2=subnet-01688644017a24a37
ddd_v1_w_MVbI_882554@runweb41542:~$ export Private_SN3=subnet-0b00a21a843b5bd24
ddd_v1_w_MVbI_882554@runweb41542:~$ export Private_SN4=subnet-0e1287c0a32c1956e
ddd_v1_w_MVbI_882554@runweb41542:~$
ddd_v1_w_MVbI_882554@runweb41542:~$ aws ec2 create-tags --resources ${Public_SN1} --tags Key=Name,Value=VPC-Luka-Public-SN-1
ddd_v1_w_MVbI_882554@runweb41542:~$ aws ec2 create-tags --resources ${Public_SN2} --tags Key=Name,Value=VPC-Luka-Public-SN-1
ddd_v1_w_MVbI_882554@runweb41542:~$ aws ec2 create-tags --resources ${Private_SN1} --tags Key=Name,Value=VPC-Luka-Private-SN-1
ddd_v1_w_MVbI_882554@runweb41542:~$ aws ec2 create-tags --resources ${Private_SN2} --tags Key=Name,Value=VPC-Luka-Private-SN-2
ddd_v1_w_MVbI_882554@runweb41542:~$ aws ec2 create-tags --resources ${Private_SN3} --tags Key=Name,Value=VPC-Luka-Private-SN-3
ddd_v1_w_MVbI_882554@runweb41542:~$ aws ec2 create-tags --resources ${Private_SN4} --tags Key=Name,Value=VPC-Luka-Private-SN-4
```

Creating the internal gateway and adding the name tag. This will allow our public IP's out of the VPC and allow us access to the network

```
ddd_v1_w_MVbI_882554@runweb41660:~$ aws ec2 create-internet-gateway --query InternetGateway.InternetGatewayId --output text
igw-0ec6648f455a14a5d
ddd_v1_w_MVbI_882554@runweb41660:~$ export IGW=igw-0ec6648f455a14a5d\
> ^C
ddd_v1_w_MVbI_882554@runweb41660:~$ export IGW=igw-0ec6648f455a14a5d
ddd_v1_w_MVbI_882554@runweb41660:~$ aws ec2 create-tags --resources ${IGW} --tags Key=Name,Value=VPC-Luka-Internet-Gateway
```

Once the tag is created we must associate the IGW with a VPC

```
ddd_v1_w_MVbI_882554@runweb41660:~$ aws ec2 attach-internet-gateway --vpc-id ${VPC_ID} --internet-gateway-id ${IGW}
```

Now that the gateway is created we have to create the routing table for each of the subnets.

Since we have an internet gateway this does not mean that we have a route table for it. We need a routing table with a default route pointing toward the internet gateway we just created

Public Routing table:

We want to create a route that are not targeting VPC IP addresses, and we want this packet to go to the internet gateway.

```
ddd_v1_w_MVbI_882554@runweb41660:~$ aws ec2 create-route-table --vpc-id ${VPC_ID} --query RouteTable.RouteTableId --output text
rtb-04fe33c35a44ef281
ddd_v1_w_MVbI_882554@runweb41660:~$ export RTBL_Public=rtb-04fe33c35a44ef281
ddd_v1_w_MVbI_882554@runweb41660:~$ aws ec2 create-tags --resources ${RTBL_Public} --tags Key=Name,Value=VPC-Luka-PUBLIC_RT
ddd_v1_w_MVbI_882554@runweb41660:~$ █
```

Now with this command I will create the route to allow all traffic that is not in our private network to be redirected to our internal gateway

```
ddd_v1_w_MVbI_882554@runweb41660:~$ aws ec2 create-route --route-table-id ${RTBL_Public} --destination-cidr-block 0.0.0.0/0 --gateway-id ${IGW}
{
  "Return": true
}
```

Now we have a routing table that holds the local route (private address space) and we have a second rule checking if the packet is not for the private(local) network. If its not it will be sent to the Internet gateway we just created (sent back out).

Once we confirm the routing table is active with our subnets listed, we will have to associate the routing table with our subnets, so we need to add our two public subnets and 4 private subnets in.

```
ddd_v1_w_MVbI_882554@runweb41660:~$ aws ec2 describe-subnets --filters "Name=vpc-id,Values=${VPC_ID}" --query "Subnets[*].{ID:SubnetId,CIDR:CidrBlock}"
[
  {
    "ID": "subnet-04a193685c8f4670c",
    "CIDR": "10.1.22.0/23"
  },
  {
    "ID": "subnet-08e1e6bbde6a8a9d5",
    "CIDR": "10.1.11.0/24"
  },
  {
    "ID": "subnet-0259e3967b346b6ca",
    "CIDR": "10.1.30.0/25"
  },
  {
    "ID": "subnet-075f43428e2e523ee",
    "CIDR": "10.1.10.0/24"
  },
  {
    "ID": "subnet-0567184c1976ead2b",
    "CIDR": "10.1.30.128/25"
  }
]
```

Now I will check if my routes have been added to the public routing table I just created



Activities Firefox Web Browser Nov 9 17:32

Learner Lab - Foundation Route tables | VPC Manager

https://console.aws.amazon.com/vpc/home 60%

Services Search for services, features, marketplace products [Alt+S]

New VPC Experience

VPC Dashboard EC2 Global View

Filter by VPC: Select a VPC

VIRTUAL PRIVATE CLOUD

Your VPCs

Subnets

Route Tables

Internet Gateways

Egress Only Internet Gateways

Carrier Gateways

DHCP Options Sets

Elastic IPs

Managed Prefix Lists

Endpoints

Endpoint Services

NAT Gateways

Peering Connections

SECURITY

Network ACLs

Security Groups

REACHABILITY

Reachability Analyzer

Feedback English (US)

Route tables (1/3) info

Filter route tables

Name	Route table ID	Explicit subnet associations	Edge associations	Main
-	rtb-03e116b96bf1a1958	-	-	Yes
-	rtb-0039fce38955d5e14	-	-	Yes
<input checked="" type="checkbox"/> VPC-Luka-PUBLIC_RT	rtb-04fe33c35a44ef281	-	-	No

Routes (2)

Filter routes Both

Destination	Target	Status
10.1.0.0/16	local	Active
0.0.0.0/0	igw-0ec6648f455a14a5d	Active

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use Cookie preferences

Now I have a routing table with no subnets associated with it. So I will now associate the subnets

aws Services Search for services, features, marketplace products [Alt+S]

New VPC Experience

VPC Dashboard EC2 Global View

Filter by VPC: Select a VPC

Terminal

Your VPCs

Subnets

Route Tables

Internet Gateways

Egress Only Internet Gateways

Carrier Gateways

DHCP Options Sets

Elastic IPs

Managed Prefix Lists

Endpoints

Endpoint Services

NAT Gateways

Peering Connections

SECURITY

Network ACLs

Security Groups

REACHABILITY

Reachability Analyzer

Route tables (1/3) info

Filter route tables

Name	Route table ID	Explicit subnet associations	Edge associations	Main
-	rtb-03e116b96bf1a1958	-	-	Yes
-	rtb-0039fce38955d5e14	-	-	Yes
<input checked="" type="checkbox"/> VPC-Luka-PUBLIC_RT	rtb-04fe33c35a44ef281	-	-	No

Explicit subnet associations (0)

Find subnet association

Subnet ID	IPv4 CIDR	IPv6 CIDR
No subnet associations		
You do not have any subnet associations.		

Subnets without explicit associations (6)

The following subnets have not been explicitly associated with any route tables and are therefore associated with the main route table:

Edit subnet associations

Find subnet association

And I have 2 that I want to set as public routes. So I will add them

The screenshot shows the AWS Management Console interface. On the left, the 'Route Tables' section is selected under 'VIRTUAL PRIVATE CLOUD'. The main area displays 'Route tables (1/3)' with a table listing three route tables. The third route table, 'VPC-Luka-PUBLIC\_RT', is selected. Below this, a section titled 'Subnets without explicit associations (6)' lists six subnets. The first two subnets, 'subnet-04a193085c8f4070c / VPC-Luka-Private-SN-2' and 'subnet-08e1e6bbd6a8a9d3 / VPC-Luka-Public-SN-2', are highlighted in blue.

Name	Route table ID	Explicit subnet associa...	Edge associations	Main	VPC
-	rtb-03e110b90bf1e1958	-	-	Yes	vpc-08092c6ad5ad00de
-	rtb-0039f0c38955d5e14	-	-	Yes	vpc-0749c147b22850616   VP...
VPC-Luka-PUBLIC_RT	rtb-04fe33c33e44ef281	-	-	No	vpc-0749c147b22850616   VP...

Subnet ID	IPv4 CIDR	IPv6 CIDR
subnet-04a193085c8f4070c / VPC-Luka-Private-SN-2	10.1.22.0/23	-
subnet-08e1e6bbd6a8a9d3 / VPC-Luka-Public-SN-2	10.1.11.0/24	-
subnet-0259e3967834fb0ca / VPC-Luka-Private-SN-3	10.1.30.0/23	-
subnet-075f43428e2e523ee / VPC-Luka-Public-SN-1	10.1.10.0/24	-
subnet-0567184c1970ead2b / VPC-Luka-Private-SN-4	10.1.30.128/25	-
subnet-038be2740dc4c4c4c / VPC-Luka-Private-SN-1	10.1.20.0/23	-

NOTE: you can see all 6 of the subnets I have created. But I will only add Public SN1 one and Public SN2. Associating the two public subnets to my Public Routing table.

```
ddd_v1_w_MvBI_882554@runweb41008:~$  
ddd_v1_w_MvBI_882554@runweb41008:~$ aws ec2 associate-route-table --subnet-id ${Public_SN1} --route-table-id ${RTBL_Public}  
{  
  "AssociationId": "rtbassoc-0e4842da7c52c9d14",  
  "AssociationState": {  
    "State": "associated"  
  }  
}  
ddd_v1_w_MvBI_882554@runweb41008:~$ aws ec2 associate-route-table --subnet-id ${Public_SN2} --route-table-id ${RTBL_Public}  
{  
  "AssociationId": "rtbassoc-0652d8a8adc972d34",  
  "AssociationState": {  
    "State": "associated"  
  }  
}  
ddd_v1_w_MvBI_882554@runweb41008:~$
```

Now once that is completed we will create a NAT Gateway so our instances can install the necessary packages from the YUM servers. The NAT will translate our private instance's IP's and will convert them to a public one to communicate outside of our network. The NAT will also translate the IP's back into private IP's when the data comes back from the YUM servers.

## Creating the NAT Gateway

This IP address needs to be static so we must allocate a elastic ip address (allocate address) You will use the command allocate-address to create a static IP. It needs to be static because if it changes, the packets that are send before its changed will not be able to find its way back as the destination IP is now different and will disregard your packets.

```
ddd_v1_w_MvBI_882554@runweb41668:~$ aws ec2 allocate-address --domain vpc
{
  "PublicIp": "52.73.184.118",
  "AllocationId": "eipalloc-01fec56cca98bffe8",
  "PublicIpv4Pool": "amazon",
  "NetworkBorderGroup": "us-east-1",
  "Domain": "vpc"
}
ddd_v1_w_MvBI_882554@runweb41668:~$ export EIP=eipalloc-01fec56cca98bffe8
```

Now I will create the gateway, export the id and create my tag

```
ddd_v1_w_MvBI_882554@runweb41668:~$ aws ec2 create-nat-gateway --subnet-id ${Public_SN1} --allocation-id ${EIP}
{
  "ClientToken": "e5594681-f896-4428-8f64-9a08df8a9f82",
  "NatGateway": {
    "CreateTime": "2021-11-09T22:45:04+00:00",
    "NatGatewayAddresses": [
      {
        "AllocationId": "eipalloc-01fec56cca98bffe8"
      }
    ],
    "NatGatewayId": "nat-01a25ce1c4d3338f7",
    "State": "pending",
    "SubnetId": "subnet-075f43428e2e523ee",
    "VpcId": "vpc-0749c147b22850616"
  }
}
ddd_v1_w_MvBI_882554@runweb41668:~$
```

Next we will create the private routing tables for the private subnets

If you check the route tables for these subnets, there is no route associated with the private subnets.

We need to create a private routing table for these subnets

We need a routing table to hold the private IP's until we can convert them back into public IP's

```
ddd_v1_w_MvBI_882554@runweb41668:~$ aws ec2 create-route-table --vpc-id ${VPC_ID} --query RouteTable.RouteTableId --output text
rtb-0b440664214c7d82d
```

```
ddd_v1_w_MvBI_882554@runweb41668:~$ aws ec2 create-tags --resources ${RTBL_PR} --tags Key=Name,Value=VPC-Luka-PRIVATE_RT
ddd_v1_w_MvBI_882554@runweb41668:~$
```

This time the default route will be going to the nat gateway instead of the Internal Gateway, due to our subnet needing to be Private! (it needs to be private because I am communicating with our instances which have a private IP)

```
ddd_v1_w_MvBI_882554@runweb41668:~$ aws ec2 create-route --route-table-id ${RTBL_PR} --destination-cidr-block 0.0.0.0/0 --gateway-id ${NATGW}
{
  "Return": true
}
ddd_v1_w_MvBI_882554@runweb41668:~$
```

There is nothing in the associated subnets when looking in VPC->Route tables->Click on specificVPC->click on subnet association. So the routing table is not set up for these

The screenshot displays the AWS Management Console interface for the 'Route tables' section of a VPC. The top navigation bar shows the AWS logo, a search bar, and the user's profile. The left sidebar contains various navigation options, including 'VPC Dashboard', 'Route Tables', and 'Subnets'. The main content area shows a table of route tables for the selected VPC. The table has columns for Name, Route table ID, Explicit subnet associations, Edge associations, Main, and VPC. The selected route table is 'VPC-Luka-PRIVATE\_RT' with ID 'rtb-0b440664214c7d82d'. Below the table, the 'Subnet associations' tab is active, showing 'Explicit subnet associations (0)' and 'Subnets without explicit associations (4)'. The console interface includes a sidebar with navigation options like 'VPC Dashboard', 'Route Tables', and 'Subnets', and a top navigation bar with the AWS logo and search bar.

I have no associations and my 4 subnets I created that still need to be associated. I will now associate them.

```
ddd_v1_w_MVbI_882554@runweb41660:~$ aws ec2 associate-route-table --subnet-id $
{Private_SN1} --route-table-id ${RTBL_PR}
{
  "AssociationId": "rtbassoc-06e767f4d6107f3c1",
  "AssociationState": {
    "State": "associated"
  }
}
ddd_v1_w_MVbI_882554@runweb41660:~$ aws ec2 associate-route-table --subnet-id $
{Private_SN2} --route-table-id ${RTBL_PR}
{
  "AssociationId": "rtbassoc-0e1a81764f819a66a",
  "AssociationState": {
    "State": "associated"
  }
}
ddd_v1_w_MVbI_882554@runweb41660:~$ aws ec2 associate-route-table --subnet-id $
{Private_SN3} --route-table-id ${RTBL_PR}
{
  "AssociationId": "rtbassoc-02c949f98411a85f0",
  "AssociationState": {
    "State": "associated"
  }
}
ddd_v1_w_MVbI_882554@runweb41660:~$ aws ec2 associate-route-table --subnet-id $
{Private_SN4} --route-table-id ${RTBL_PR}
{
  "AssociationId": "rtbassoc-095776796b1a9f5fc",
  "AssociationState": {
    "State": "associated"
  }
}
ddd_v1_w_MVbI_882554@runweb41660:~$
```

Once this is completed I check that my route tables are complete In the Private Route table



## Private routing table

The screenshot shows the AWS Management Console interface for the 'Route tables | VPC Manager' section. The left sidebar contains navigation links for VPC Dashboard, EC2 Global View, and various VPC services. The main content area displays a list of route tables. The 'VPC-Luka-PRIVATE...' route table is selected, showing its details. Below the table list, the 'Explicit subnet associations (4)' section is visible, listing four subnets with their respective IPv4 and IPv6 CIDR blocks.

Name	Route table ID	Explicit subnet assoc...	Edge associations	Main	VPC
-	rtb-03e110b9b1a1950	-	-	Yes	vpc-08092ceade3ad00de
-	rtb-0039fce38955d5e14	-	-	Yes	vpc-0749c147b22850016   VP...
VPC-Luka-PUBLIC_RT	rtb-04fe33c35a44ef281	2 subnets	-	No	vpc-0749c147b22850016   VP...
VPC-Luka-PRIVATE...	rtb-0b440664214c7d82d	4 subnets	-	No	vpc-0749c147b22850016   VP...

Subnet ID	IPv4 CIDR	IPv6 CIDR
subnet-04a193685b8f670c / VPC-Luka-Private-SN-2	10.1.22.0/25	-
subnet-0239e39678346b6ca / VPC-Luka-Private-SN-3	10.1.30.0/25	-
subnet-0507184c1970ead2b / VPC-Luka-Private-SN-4	10.1.30.128/25	-
subnet-038be2740db4c0c / VPC-Luka-Private-SN-1	10.1.20.0/25	-

## Public Routing table

The screenshot shows the AWS Management Console interface for the 'Route tables | VPC Manager' section. The left sidebar contains navigation links for VPC Dashboard, EC2 Global View, and various VPC services. The main content area displays a list of route tables. The 'VPC-Luka-PUBLIC\_RT' route table is selected, showing its details. Below the table list, the 'Explicit subnet associations (2)' section is visible, listing two subnets with their respective IPv4 and IPv6 CIDR blocks.

Name	Route table ID	Explicit subnet assoc...	Edge associations	Main	VPC
-	rtb-03e110b9b1a1950	-	-	Yes	vpc-08092ceade3ad00de
-	rtb-0039fce38955d5e14	-	-	Yes	vpc-0749c147b22850016   VP...
VPC-Luka-PUBLIC_RT	rtb-04fe33c35a44ef281	2 subnets	-	No	vpc-0749c147b22850016   VP...
VPC-Luka-PRIVATE...	rtb-0b440664214c7d82d	4 subnets	-	No	vpc-0749c147b22850016   VP...

Subnet ID	IPv4 CIDR	IPv6 CIDR
subnet-08e1e6b0d6a8a9d5 / VPC-Luka-Public-SN-2	10.1.11.0/24	-
subnet-075f43428e2e323ee / VPC-Luka-Public-SN-1	10.1.10.0/24	-

**PLEASE NOTE THAT I HAD TO EDIT SOME OF THE SECURITY GROUPS AND OTHER FIELDS USING GUI.  
Due to not being able to use the command line properly.**

Now we have to start a bastion host.

bastion runs in a public subnet, we are able to go from the bastion host to outside the vpc. from the bastion host, we can ssh anywhere in the instance. So I can communicate with bastion and then with the web server. SSH to bastion host -> then SSH into instance from bastion host

```
ddd_v1_w_MVbI_882554@runweb41668:~$ export BASTION_SG=sg-0bdbd7b8be22b8c63
ddd_v1_w_MVbI_882554@runweb41668:~$ aws ec2 create-tags --resources ${BASTION_SG} --tags Key=Name,Value=Luka-BASTION_SG
ddd_v1_w_MVbI_882554@runweb41668:~$ aws ec2 authorize-security-group-ingress --group-id ${BASTION_SG} --protocol tcp --port 22 --cidr 0.0.0.0/0
ddd_v1_w_MVbI_882554@runweb41668:~$
```

Make sure that you are only opening port 22 for the bastion host. She is using 0.0.0.0/0 because she does not know her own ip address.

**Now we will create the EC2 Instance (bastion)**

```
ddd_v1_w_MVbI_882554@runweb41668:~$ aws ec2 authorize-security-group-ingress --group-id ${BASTION_SG} --protocol tcp --port 22 --cidr 0.0.0.0/0
ddd_v1_w_MVbI_882554@runweb41668:~$ aws ec2 run-instances --image-id ami-8c1be5f6 --count 1 --instance-type t2.micro --key-name vockey --security-group-
```

```
{
  "Groups": [],
  "Instances": [
    {
      "AmiLaunchIndex": 0,
      "ImageId": "ami-8c1be5f6",
      "InstanceId": "i-001bf4ba7331b3cbf",
      "InstanceType": "t2.micro",
      "KeyName": "vockey",
      "State": "disabled",
    },
    {
      "Placement": {
        "AvailabilityZone": "us-east-1c",
        "GroupName": "",
        "Tenancy": "default"
      },
      "PrivateDnsName": "ip-172-31-22-118.ec2.internal",
      "PrivateIpAddress": "172.31.22.118",
      "ProductCodes": [],
      "PublicDnsName": "",
      "State": {
        "Code": 0,
        "Name": "pending"
      },
      "StateTransitionReason": "",
      "SubnetId": "subnet-84b2d68b8dc937247",
      "VpcId": "vpc-88092ceada5ad00de",
      "Architecture": "x86_64",
      "BlockDeviceMappings": [],
    }
  ]
}
```

```
ddd_v1_w_MVbI_882554@runweb41668:~$ export BASTION=i-001bf4ba7331b3cbf
```

Once I create the bastion host I will export it and Tag it. Then I will check if the instance is starting to be instantiated.

```
    "Monitoring": {
      "State": "disabled"
    },
    "Placement": {
      "AvailabilityZone": "us-east-1c",
      "GroupName": "",
      "Tenancy": "default"
    },
    "PrivateDnsName": "ip-172-31-22-118.ec2.internal",
  },
}

ddd_v1_w_MVbI_882554@runweb41668:~$ aws ec2 create-tags --resources ${BASTION}
--tags Key=Name,Value=VPC-Luka-Bastion
ddd_v1_w_MVbI_882554@runweb41668:~$ aws ec2 describe-instances --instance-ids $
{BASTION}

{
  "Reservations": [
    {
      "Groups": [],
      "Instances": [
        {
          "AmiLaunchIndex": 0,
          "ImageId": "ami-8c1be5f6",
          "InstanceId": "i-001bf4ba7331b3cbf",
          "InstanceType": "t2.micro",
          "KeyName": "vockey",
          "LaunchTime": "2021-11-09T23:15:27+00:00",
          "Monitoring": {
            "State": "disabled"
          }
        }
      ]
    }
  ]
}
```

```
    "Monitoring": {
      "State": "disabled"
    },
    "Placement": {
      "AvailabilityZone": "us-east-1c",
      "GroupName": "",
      "Tenancy": "default"
    },
    "PrivateDnsName": "ip-172-31-22-118.ec2.internal",
  },
}

:█
```

The bastion host needs a public IP. Since there is no public IP, we can automatically assign one using AWS.

```
ddd_v1_w_MVbI_882554@runweb41668:~$ aws ec2 modify-subnet-attribute --subnet-id
${Public_SN1} --map-public-ip-on-launch
ddd_v1_w_MVbI_882554@runweb41668:~$ aws ec2 modify-subnet-attribute --subnet-id
${Public_SN2} --map-public-ip-on-launch
```

Since assigning an IP is done whenever an instance its being provisioned. We will have to terminate and instantiate the instance.

Next we have to Provision the Security group for our application load balancer

```
ddd_v1_w_MVbI_882554@runweb41668:~$ aws ec2 create-security-group --group-name
VPC-Luka-ALB-SG --description "ALB-SG" --vpc-id ${VPC_ID}
{
  "GroupId": "sg-085088d83d3801d48"
}
ddd_v1_w_MVbI_882554@runweb41668:~$ ^C
ddd_v1_w_MVbI_882554@runweb41668:~$ aws ec2 create-tags --resources ${ALB_SG} -
-tags Key=Name,Value=ALB_SG
```

We do not need to do anything else as we are not hosting any certificates, otherwise we would need to also host https

```
ddd_v1_w_MVbI_882554@runweb41668:~$ aws ec2 authorize-security-group-ingress --
group-id ${ALB_SG} --protocol tcp --port 80 --cidr 0.0.0.0/0
ddd_v1_w_MVbI_882554@runweb41668:~$ █
```

We are allowing port 80 to all ip's because we do not know the ip of our webserver ec2 instances.

Once the Security group for the Load balancer is completed, I will create the security group for the EC2 Webservers

```
ddd_v1_w_MVbI_882554@runweb41668:~$ aws ec2 create-security-group --group-name
VPC-Luka-Web-SG --description "webserver SG" --vpc-id ${VPC_ID}

{
  "GroupId": "sg-0413af186ce3287da"
}
```

Creating access to all HTTP Ports on our inbound web server security group

```
ddd_v1_w_MVbI_882554@runweb41668:~$ aws ec2 authorize-security-group-ingress --
group-id ${WEB_SG} --protocol tcp --port 80 --source-group ${ALB_SG}
```



Creating access to all HTTP Ports on our inbound web server security group

### Done in GUI

Note when I am creating the security group for Web applications (instances). I am not referencing ip addresses, I am referencing The ALB and Bastion Security groups.

### Creating SSH access from our instance to the bastion instance.

```
ddd_v1_w_MVbI_882554@runweb41668:~$ aws ec2 authorize-security-group-ingress --group-id ${WEB_SG} --protocol tcp --port 22 --source-group ${BASTION_SG}
```

Why do we need these two rules for the webserver's SG? Allowing port 80 on the Application Load balancer Security Group allows load balancer to send and receive http packets.

And allowing port 22 on the bastion clients Security Group, Allows you to SSH from the bastion client to the instances

### Now I will create the two webserver's.

To create the servers I will create a script to paste the data inside.

I created two servers and have associated them with the right security groups.

I also added a short html file into the webserver to be deployed as the root ALB url.

### SSH into Bastion host

```
ddd_v1_w_MVbI_882554@runweb41732:~$ ssh -i ~/.ssh/labsuser.pem ec2-user@54.205.107.160
The authenticity of host '54.205.107.160 (54.205.107.160)' can't be established.
ECDSA key fingerprint is SHA256:0W6yebw18esYzCt41jzxQjr66h87gGEHLLPBz6x9azo.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '54.205.107.160' (ECDSA) to the list of known hosts.
```

```
 _ | _ | _ )
 _ | ( /   Amazon Linux AMI
 _ | \ | _ |
```

```
https://aws.amazon.com/amazon-linux-ami/2017.09-release-notes/
```

```
34 package(s) needed for security, out of 68 available
```

```
Run "sudo yum update" to apply all updates.
```

```
Amazon Linux version 2018.03 is available.
```

```
[ec2-user@ip-10-1-10-199 ~]$
```

```
[ec2-user@ip-10-1-10-199 ~]$
```

```
[ec2-user@ip-10-1-10-199 ~]$
```

```
[ec2-user@ip-10-1-10-199 ~]$
```

Checking if HTTP is allowed from bastion to instances by using curl to pull the html file.



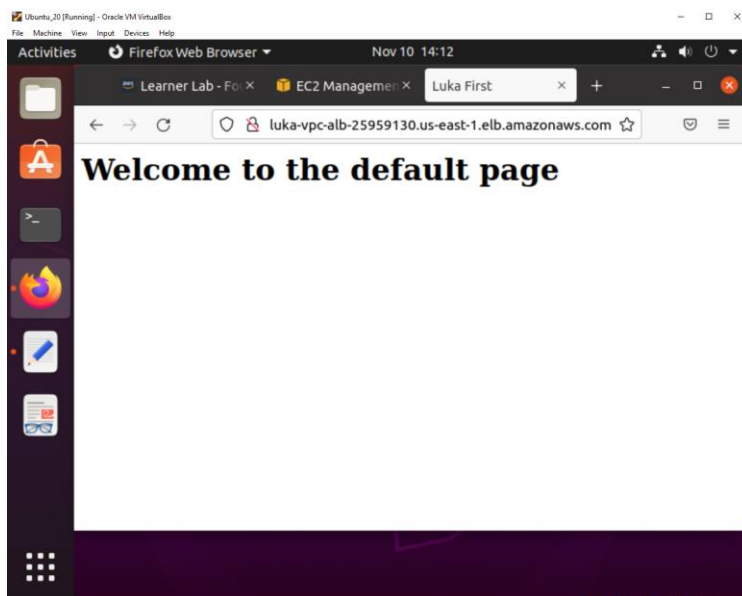
## SSH into Web Instances

```
[ec2-user@ip-10-1-10-199 ~]$ ssh -i ~aws_ssh_key.pem ec2-user@10.1.21.72

 _ _ _ _ _
| | | | |
| | ( / Amazon Linux 2 AMI
|_|_|_|_|

https://aws.amazon.com/amazon-linux-2/
13 package(s) needed for security, out of 46 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-10-1-21-72 ~]$
```

After checking my ALB security group, I noticed it was not set to the default security group. Once I fixed this I was able to load the default page.



The next step is to create an application load balancer. To do this I will need to create a target group so I am able to target my instances and have all of the ip's be hosted on the same url + They are then able to balance the load of requests.

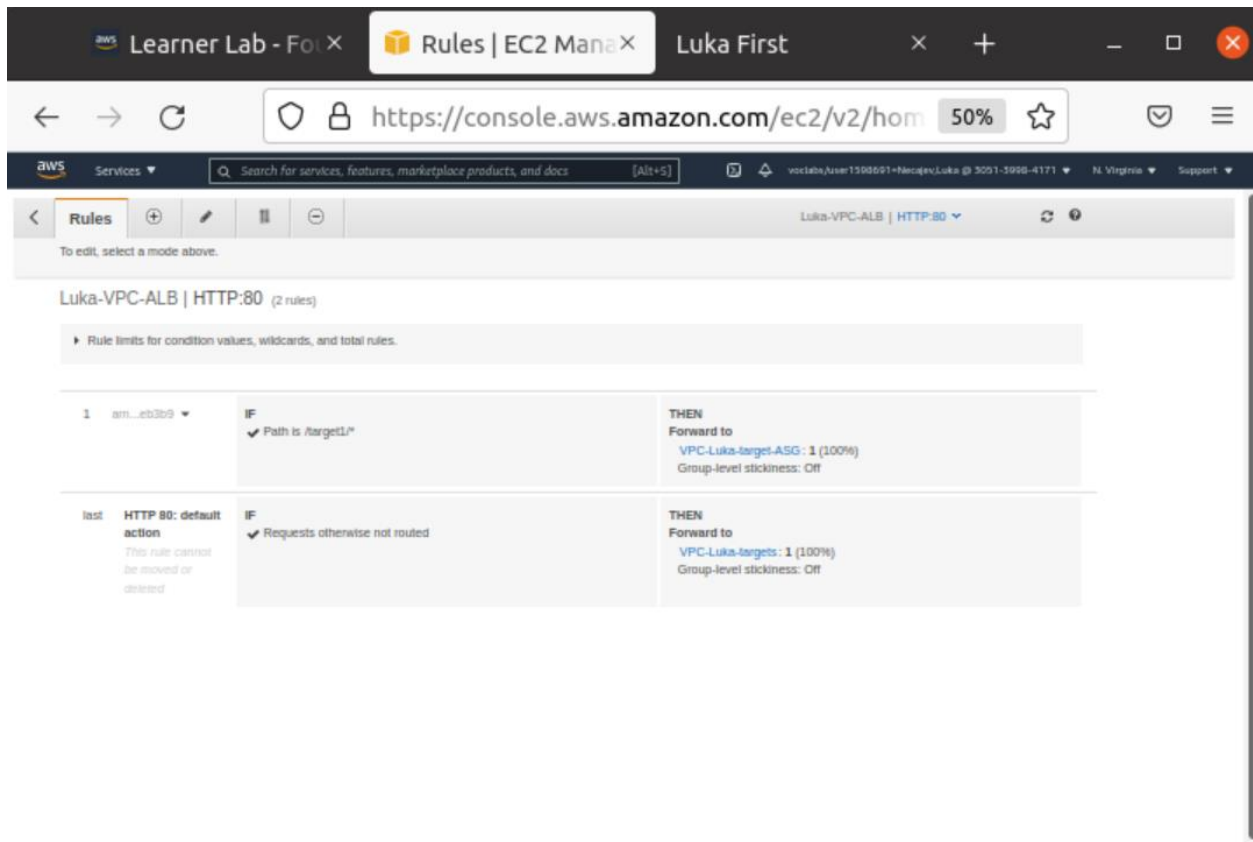
### Creating target groups

```
ddd_v1_w_MVbI_882554@runweb41752:~$ aws elbv2 create-target-group --name VPC-Luka-target-ASG --protocol HTTP --port 88 --vpc-id ${VPC_ID}
{
  "TargetGroups": [
    {
      "TargetGroupArn": "arn:aws:elasticloadbalancing:us-east-1:385139984171:targetgroup/VPC-Luka-target-ASG/2268f1a8456e1833",
      "TargetGroupName": "VPC-Luka-target-ASG",
      "Protocol": "HTTP",
      "Port": 88,
      "VpcId": "vpc-053b1014a1c69bacf",
      "HealthCheckProtocol": "HTTP",
      "HealthCheckPort": "traffic-port",
      "HealthCheckEnabled": true,
      "HealthCheckIntervalSeconds": 38,
      "HealthCheckTimeoutSeconds": 5,
      "HealthyThresholdCount": 5,
      "UnhealthyThresholdCount": 2,
      "HealthCheckPath": "/",
      "Matcher": {
```

Creating Target group listener that is pointing/listening for a url with the /target1 on the end

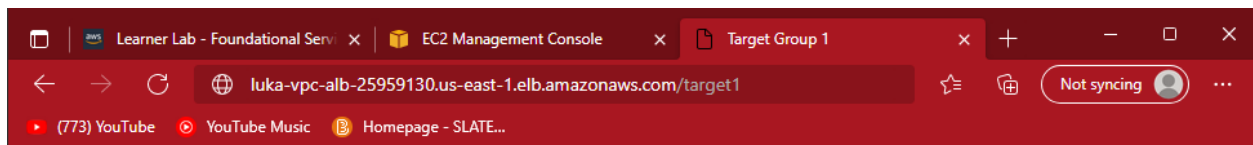
```
ddd_v1_w_MVbI_882554@runweb41752:~$ export LISTENER_ARN=arn:aws:elasticloadbalancing:us-east-1:385139984171:listener/app/Luka-VPC-ALB/3827122ac6675bd6/aaf3b7f5719572b1
ddd_v1_w_MVbI_882554@runweb41752:~$ aws elbv2 create-rule --listener-arn ${LISTENER_ARN} --priority 10 --conditions Field=path-pattern,Values='/target1/*' --actions Type=forward,TargetGroupArn=${TARGET_ASG_ARN}
{
  "Rules": [
    {
      "RuleArn": "arn:aws:elasticloadbalancing:us-east-1:385139984171:listener-rule/app/Luka-VPC-ALB/3827122ac6675bd6/aaf3b7f5719572b1/ee54bd33e75eb3b9",
      "Priority": "10",
      "Conditions": [
        {
          "Field": "path-pattern",
          "Values": [
            "/target1/*"
          ]
        }
      ],
      "Actions": [
        {
          "Type": "forward",
          "TargetGroupArn": "arn:aws:elasticloadbalancing:us-east-1:385139984171:targetgroup/VPC-Luka-target-ASG/2268f1a8456e1833"
        }
      ]
    }
  ]
}
```

Showing that the rule was added to my VPC



After this, I was able to view the albURL/target1 website

**Target one working!**



## Luka Target Group 1

To get the second Path to the second file I have to create a third listener that is attached to my ALB, and is attached to my Autoscaling Target Group.

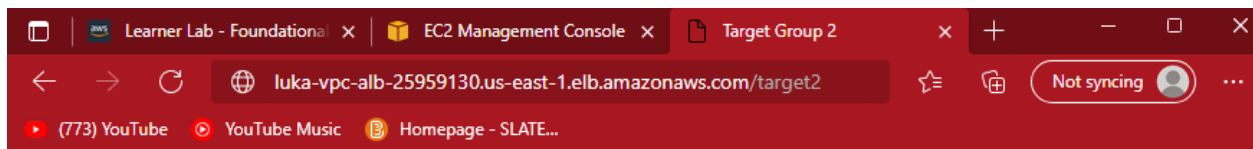
```

ddd_v1_w_MvBI_882554@runweb41748:~$ aws elbv2 create-rule --listener-arn ${LISTENER_ARN} --priority 9 --conditions Field=path-pattern,Values='/target2/*' --actions Type=forward,TargetGroupArn=${TARGET_ASG_ARN}
{
  "Rules": [
    {
      "RuleArn": "arn:aws:elasticloadbalancing:us-east-1:305139984171:listener-rule/app/Luka-VPC-ALB/3827122ac6675bd6/aaf3b7f5719572b1/dfba71d461e4cecc",
      "Priority": "9",
      "Conditions": [
        {
          "Field": "path-pattern",
          "Values": [
            "/target2/*"
          ],
          "PathPatternConfig": {
            "Values": [
              "/target2/*"
            ]
          }
        }
      ],
      "Actions": [
        {
          "Type": "forward",
          "TargetGroupArn": "arn:aws:elasticloadbalancing:us-east-1:305139984171:targetgroup/VPC-Luka-target-ASG/2260f1a8456e1833",
          "ForwardConfig": {
            "TargetGroups": [
              {
                "TargetGroupArn": "arn:aws:elasticloadbalancing:us-east-1:305139984171:targetgroup/VPC-Luka-target-ASG/2260f1a8456e1833",
                "Weight": 1
              }
            ],
            "TargetGroupStickinessConfig": {
              "Enabled": false
            }
          }
        }
      ],
      "IsDefault": false
    }
  ]
}
ddd_v1_w_MvBI_882554@runweb41748:~$

```

Once this is completed. I updated the script and re-pushed the startup.sh file I created to host both the target paths. Listening to these paths I was able to view the desired website by changing the url.

### Target two working



## Luka Target Group 2

The screenshot shows the AWS Management Console interface for configuring a VPC ALB. The top navigation bar includes the AWS logo, 'Services' dropdown, a search bar, and user information. The main content area is titled 'Rules' and shows 'Luka-VPC-ALB | HTTP:80 (3 rules)'. Below this, there's a section for 'Rule limits for condition values, wildcards, and total rules.' The rules are listed as follows:

Rule ID	Condition (IF)	Action (THEN)
1 am...dfba71d461e4cecc	Path is /target2/*	Forward to VPC-Luka-target-ASG: 1 (100%) Group-level stickiness: Off
2 am...ee54bd33e75eb3b9	Path is /target1/*	Forward to VPC-Luka-target-ASG: 1 (100%) Group-level stickiness: Off
last HTTP 80: default action	Requests otherwise not routed	Forward to VPC-Luka-targets: 1 (100%) Group-level stickiness: Off

This is stating if the packet is going to target1 or target2 send it to the auto scaling group target, if it is going to the default home page, send it to my original web servers.

NOTE: New Bastion IP since I turned my session off.

The screenshot shows the 'Details' tab of an EC2 instance named 'i-06e6468facce849aa (VPC-Luka-Bastion)'. The instance is in a 'Running' state. The public IPv4 address is 54.90.221.224, and the private IPv4 address is 10.1.10.199. The instance is running on an Amazon Linux 2 AMI.

Instance ID	Public IPv4 address	Private IPv4 addresses
i-06e6468facce849aa (VPC-Luka-Bastion)	54.90.221.224   <a href="#">open address</a>	10.1.10.199

IP of one of my AutoScaling instances



## Instance: i-05c7b031499fa3056 (AutoScalingSomething)



Details	Security	Networking	Storage	Status checks	Monitoring	Tags
▼ Instance summary <a href="#">Info</a>						
Instance ID i-05c7b031499fa3056 (AutoScalingSomething)	Public IPv4 address -		Private IPv4 addresses 10.1.30.204			
IPv6 address -	Instance state Running		Public IPv4 DNS -			
Private IPv4 DNS ip-10-1-30-204.ec2.internal	Instance type t2.micro		Elastic IP addresses -			

## SSH into auto scaling group from my bastion host

```
ddd_v1_w_MVbI_882554@runweb41748:~$ ssh -i ~/.ssh/labsuser.pem ec2-user@54.90.221.224
The authenticity of host '54.90.221.224 (54.90.221.224)' can't be established.
ECDSA key fingerprint is SHA256:0w6yebW18esYzCt41jzxQjr66hB7gGEHLLPBz6x9azo.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '54.90.221.224' (ECDSA) to the list of known hosts.
Last login: Wed Nov 10 21:37:30 2021 from 34.218.75.69
```

```

_ | _ | _ )
_ | ( / Amazon Linux AMI
_ | \ | _ |

```

<https://aws.amazon.com/amazon-linux-ami/2017.09-release-notes/>

34 package(s) needed for security, out of 68 available

Run "sudo yum update" to apply all updates.

Amazon Linux version 2018.03 is available.

```
[ec2-user@ip-10-1-10-199 ~]$ ssh -i ~aws_ssh_key.pem ec2-user@10.1.30.204
```

The authenticity of host '10.1.30.204 (10.1.30.204)' can't be established.

ECDSA key fingerprint is SHA256:3UtMoeXed2Z1JXMUKfCgCFNaTfoq4rg7VnXZD7S47HM.

ECDSA key fingerprint is MD5:21:c5:6c:56:33:8e:ab:67:12:ce:fd:3a:b0:73:df:6d.

Are you sure you want to continue connecting (yes/no)? yes

Warning: Permanently added '10.1.30.204' (ECDSA) to the list of known hosts.

```

_ | _ | _ )
_ | ( / Amazon Linux 2 AMI
_ | \ | _ |

```

<https://aws.amazon.com/amazon-linux-2/>

13 package(s) needed for security, out of 46 available

Run "sudo yum update" to apply all updates.

```
[ec2-user@ip-10-1-30-204 ~]$
```

## Showing Target Groups

EC2 > Target groups

Target groups (1/2) Info

Search or filter target groups

	Name	ARN	Port	Protocol	Target type	Load balancer
<input type="checkbox"/>	VPC-Luka-target-ASG	arn:aws:elasticloadbalancing:us-east-1:3051-3988-4171:targetgroup/VPC-Luka-target-ASG/2260f1a8456e1833	80	HTTP	Instance	Luka-VPC
<input checked="" type="checkbox"/>	VPC-Luka-targets	arn:aws:elasticloadbalancing:us-east-1:3051-3988-4171:targetgroup/VPC-Luka-targets/2260f1a8456e1833	80	HTTP	Instance	Luka-VPC

Details | Targets | Monitoring | Health checks | Attributes | Tags

Details

Target type	Instance	Protocol : Port	HTTP: 80	Protocol version	HTTP1	VPC
IP address type	IPv4	Load balancer	Luka-VPC-ALB			

EC2 > Target groups

Target groups (1/2) Info

Search or filter target groups

	Name	ARN	Port	Protocol	Target type	Load balancer
<input checked="" type="checkbox"/>	VPC-Luka-target-ASG	arn:aws:elasticloadbalancing:us-east-1:3051-3988-4171:targetgroup/VPC-Luka-target-ASG/2260f1a8456e1833	80	HTTP	Instance	Luka-VPC
<input type="checkbox"/>	VPC-Luka-targets	arn:aws:elasticloadbalancing:us-east-1:3051-3988-4171:targetgroup/VPC-Luka-targets/2260f1a8456e1833	80	HTTP	Instance	Luka-VPC

VPC-Luka-target-ASG

arn:aws:elasticloadbalancing:us-east-1:3051-3988-4171:targetgroup/VPC-Luka-target-ASG/2260f1a8456e1833

Details | Targets | Monitoring | Health checks | Attributes | Tags

Registered targets (2)

Filter resources by property or value

	Instance ID	Name	Port	Zone	Health status	Health status details
<input type="checkbox"/>	i-05c7b031499fa3056	AutoScalingSomething	80	us-east-1b	healthy	
<input type="checkbox"/>	i-04937d924536dcb39	AutoScalingSomething	80	us-east-1c	healthy	

This target group was created to combine my two original web servers I hosted and my two autoscaling servers that I accidentally named “autoScalingSomething”. This allowed me to have access my targets at the same time.

The screenshot shows the AWS Management Console interface. The left sidebar contains navigation links for various AWS services, including EC2, IAM, and CloudFormation. The main content area displays the 'Target groups' page for the 'us-east-1' region. The 'VPC-Luka-targets' target group is selected, and its details are shown in the 'Targets' tab. The 'Registered targets' table lists two instances: 'VPC-Luka-Web-2' (unhealthy) and 'VPC-Luka-Web-1' (healthy).

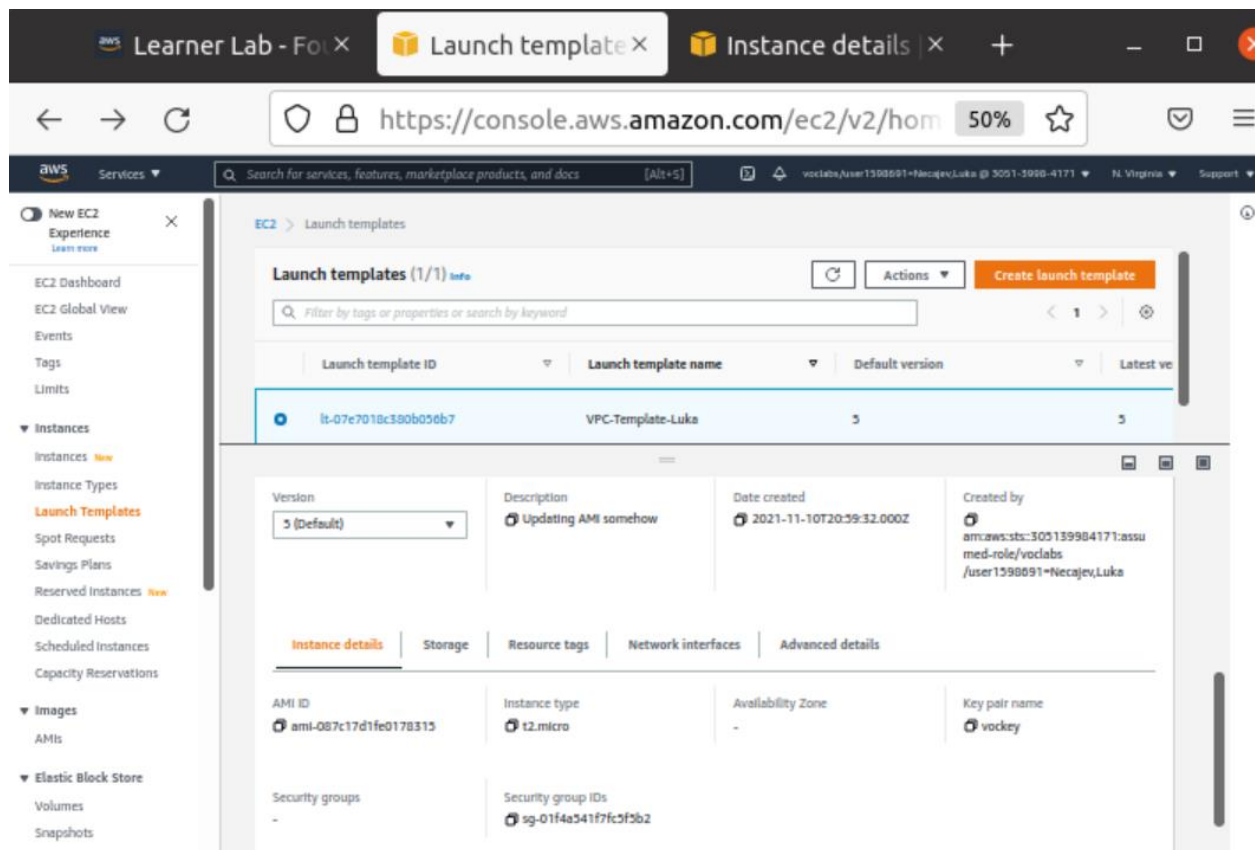
Name	ARN	Port	Protocol	Target type	Load balancer
VPC-Luka-target-ASG	arn:aws:elasticloadbalancing:us-east-1:3051-3998-4171:targetgroup/VPC-Luka-target-ASG/80	80	HTTP	Instance	Luka-VPC-LoadBalancer
VPC-Luka-targets	arn:aws:elasticloadbalancing:us-east-1:3051-3998-4171:targetgroup/VPC-Luka-targets/80	80	HTTP	Instance	Luka-VPC-LoadBalancer

Instance ID	Name	Port	Zone	Health status	Health status details
i-062e583a88e88bbbc	VPC-Luka-Web-2	80	us-east-1b	unhealthy	Request timed out
i-07f8c0e1306ddce69	VPC-Luka-Web-1	80	us-east-1c	healthy	

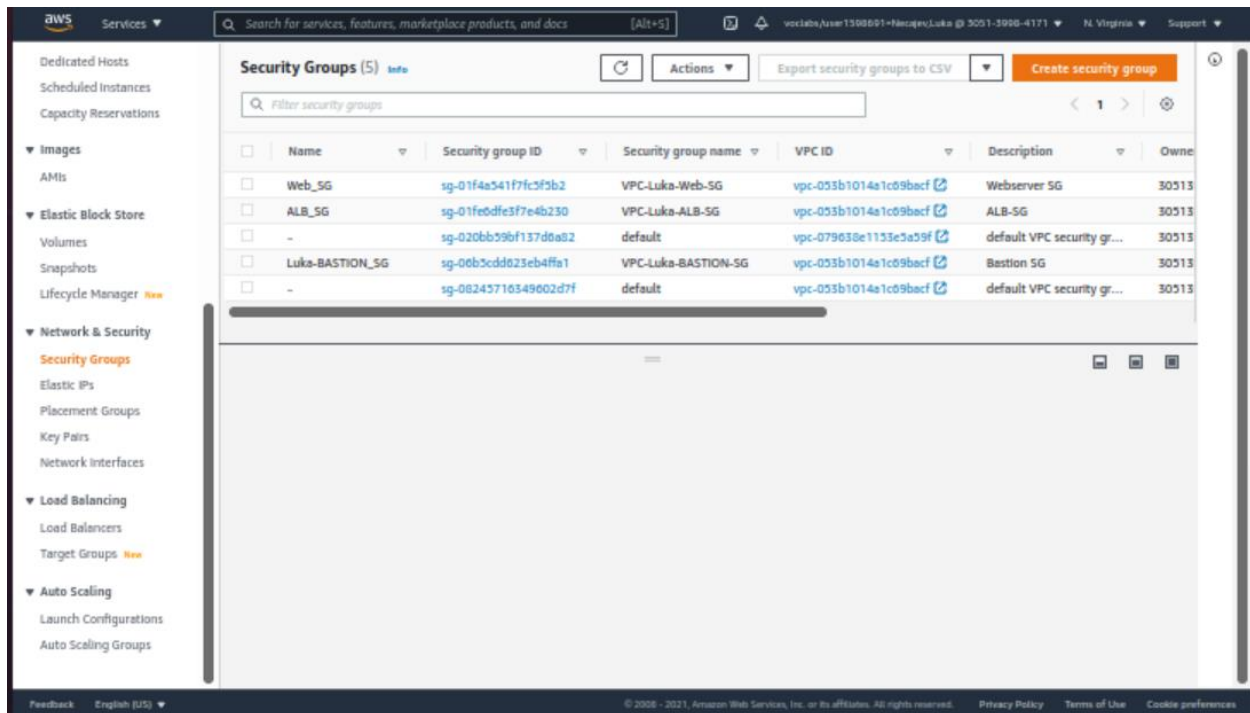
Then I have my original target group that is with my regular instances without Autoscaling enabled.

## Showing my Launch Template



This is the launch template that I created. I had to change it 5 times to finally get it working, but once I got it working I associated it with my autoscaling group and created my two other web servers that have autoscaling enabled. So I will always have a minimum of two servers going at a time.

## Security Group



The screenshot displays the AWS Management Console interface for the 'Security Groups' section. The left-hand navigation pane lists various AWS services, with 'Network & Security' expanded to show 'Security Groups' as the active selection. The main content area, titled 'Security Groups (5)', contains a search bar and a table listing five security groups. Each row in the table includes a checkbox, the group's name, its ID, the associated VPC name, the VPC ID, a description, and the owner's account ID. The security groups listed are 'Web\_SG', 'ALB\_SG', a default group, 'Luka-BASTION\_SG', and another default group. The 'Luka-BASTION\_SG' is highlighted with a blue background. At the bottom of the console, there is a footer with copyright information and links to privacy policy, terms of use, and cookie preferences.

	Name	Security group ID	Security group name	VPC ID	Description	Owner
<input type="checkbox"/>	Web_SG	sg-01f4a541f7fc5f5b2	VPC-Luka-Web-SG	vpc-053b1014a1c09bacf	Webserver SG	30513
<input type="checkbox"/>	ALB_SG	sg-01fe0dfe3f7e4b230	VPC-Luka-ALB-SG	vpc-053b1014a1c09bacf	ALB-SG	30513
<input type="checkbox"/>	-	sg-020bb59bf137d6a82	default	vpc-079638e1133e3a59f	default VPC security gr...	30513
<input type="checkbox"/>	Luka-BASTION_SG	sg-06b5cdd823eb4ffa1	VPC-Luka-BASTION-SG	vpc-053b1014a1c09bacf	Bastion SG	30513
<input type="checkbox"/>	-	sg-08245710349602d7f	default	vpc-053b1014a1c09bacf	default VPC security gr...	30513

Here is a list of my security groups at the end, I will talk you through each one later in the document



## Showing load balancers

The screenshot shows the AWS Management Console interface for a Load Balancer. At the top, there is a search bar and navigation links. Below the search bar, there is a table with columns: Name, DNS name, State, VPC ID, Availability Zones, and Type. The table contains one entry: 'Luka-VPC-ALB' with a DNS name of 'Luka-VPC-ALB-25959130.u...', State of 'Active', VPC ID of 'vpc-053b1014a1c09bacf', Availability Zones of 'us-east-1c, us-east-1b', and Type of 'application'.

Below the table, there is a detailed view of the Load Balancer configuration. The configuration includes:

- Type: application
- Scheme: Internet-facing
- IP address type: ipv4 (with an 'Edit IP address type' button)
- VPC: vpc-053b1014a1c09bacf (with a link icon)
- Availability Zones: subnet-020160179afa3771c - us-east-1c (with a link icon) and subnet-077908271100c83f0 - us-east-1b (with a link icon). Both subnets have an 'IPv4 address: Assigned by AWS'.
- Hosted zone: Z35SXDOTRQ7X7K
- Creation time: November 10, 2021 at 10:56:47 AM UTC-5

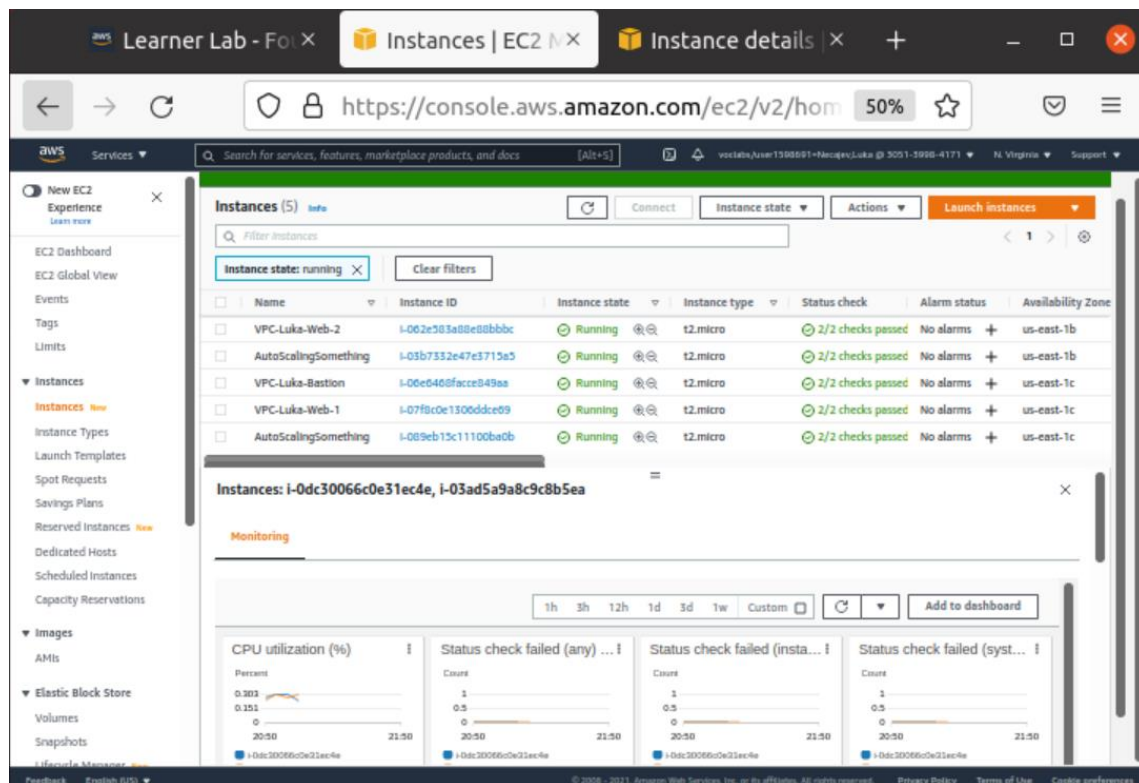
Below the configuration, there is a 'Security' section showing the Security groups: sg-01fe6dfe37e4b230, VPC-Luka-ALB-SG, and ALB-SG. There is an 'Edit security groups' button.

At the bottom of the console, there is a footer with copyright information: © 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved. There are also links for Privacy Policy, Terms of Use, and Cookie preferences.

Here is my load balancer and the availability zones I am using. I had a bunch of problems creating any sort of instance in us-east 1a so I used us-east 1c instead. I kept getting a too many instances in us-east 1a when I had no instances set up yet. So I decided to re-do the entire thing again.

I only have one listener created, and it is only listening to HTTP traffic.

## Instances Created by autoscaling



My two autoscaling instances working as well as my other servers working as well (2 Web servers and my bastion host)

**NOTE: I switched to my pc and used Microsoft Edge on the last day to do the bonus**

## Infrastructure Description:

Region: **us-east-1b & 1c**

VPC CIDR: **10.1.0.0/16** (custom VPC, **do not use default VPC for this assignment**)

**Attached I have a set of all of my commands used in the project. Do note that I had to assign some of my instance/ALB security groups using GUI and I did not mention it because I was troubleshooting and forgot.**

ALB is deployed into public subnets with 1 listener that have 3 rules:

- <http://<ALB DNS>/> is served by a target group of 2 EC2 instances depicted as EC2 instances named "Web" on the diagram. These instances are not part of an auto-scaling group. These are

a part of target group “VPC-Luka-Targets”

- <http://<ALB DNS>/target1/> is served by a target group attached to an auto-scaling group of 2 EC2 instances depicted as EC2 instances named “VPC-Luka-target-ASG” on the diagram.
- <http://<ALB DNS>/target2/> is served by a target group attached to an auto-scaling group of 2 EC2 instances depicted as EC2 instances named “VPC-Luka-target-ASG” on the diagram. These instances are serving the message below. (Bonus question)

Bastion host is deployed into public subnet to allow ssh access to webserver running in the private subnets.

NAT GW is deployed into public subnet to allow webserver outbound access to the yum repositories. Webserver need outbound access to install httpd server.

The cloud infrastructure was deployed with the network security in mind and has the following requirements to the security groups **inbound** traffic:

**ALB SG:**

- HTTP traffic from the internet (0.0.0.0/0)

The Application Load Balancer Security Group allows only HTTP traffic from the internet into our system. Since we want all users to be able to access our website we will allow all IP's into our system.

**Webserver SG:**

- HTTP traffic from ALB

Since we are hosting a website, we want traffic requests for HTTP to be accepted from the Application load balancer. Therefore we have to create an opening on port 80 and specify the Application Load Balancer Security Group.

- ICMP, HTTP, SSH traffic from Bastion

**Bastion SG:**

- SSH traffic from the Internet (0.0.0.0/0)

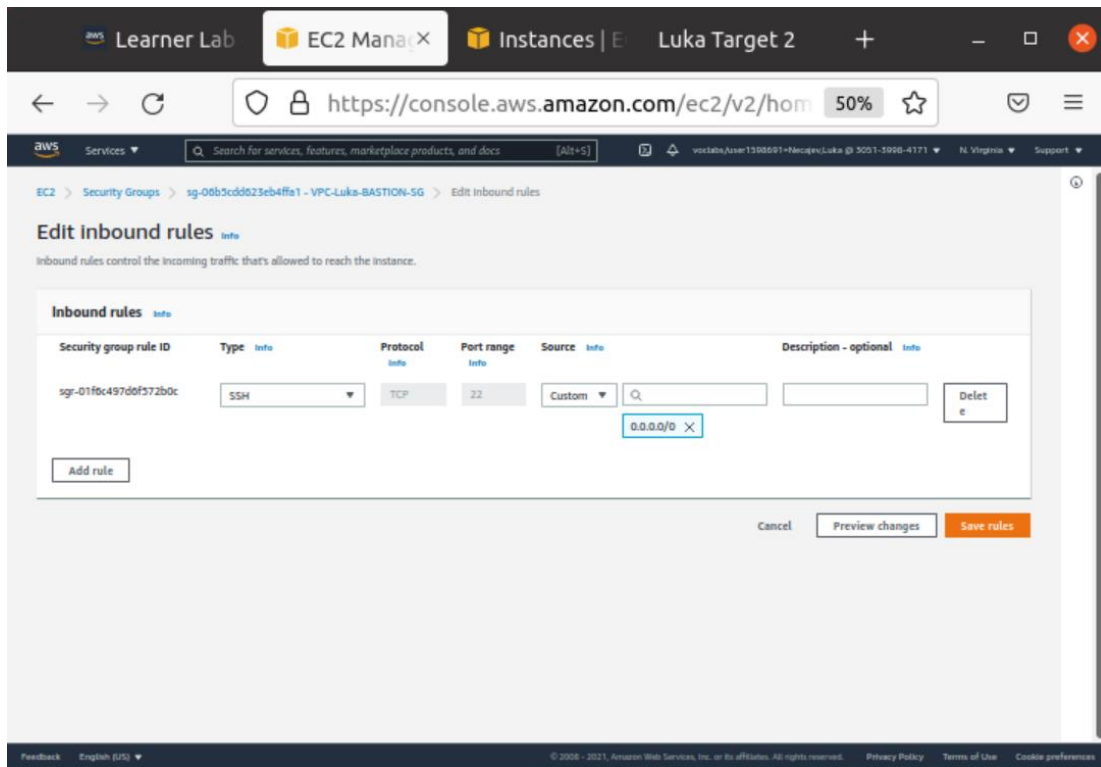
Since we do not know the IP of the aws admin/aws user is. And I do not know the ip of the aws session I will want to allow all IP's to ssh into our bastion system as long as they have the authentication key.

## Security Groups

Bastion Security Groups



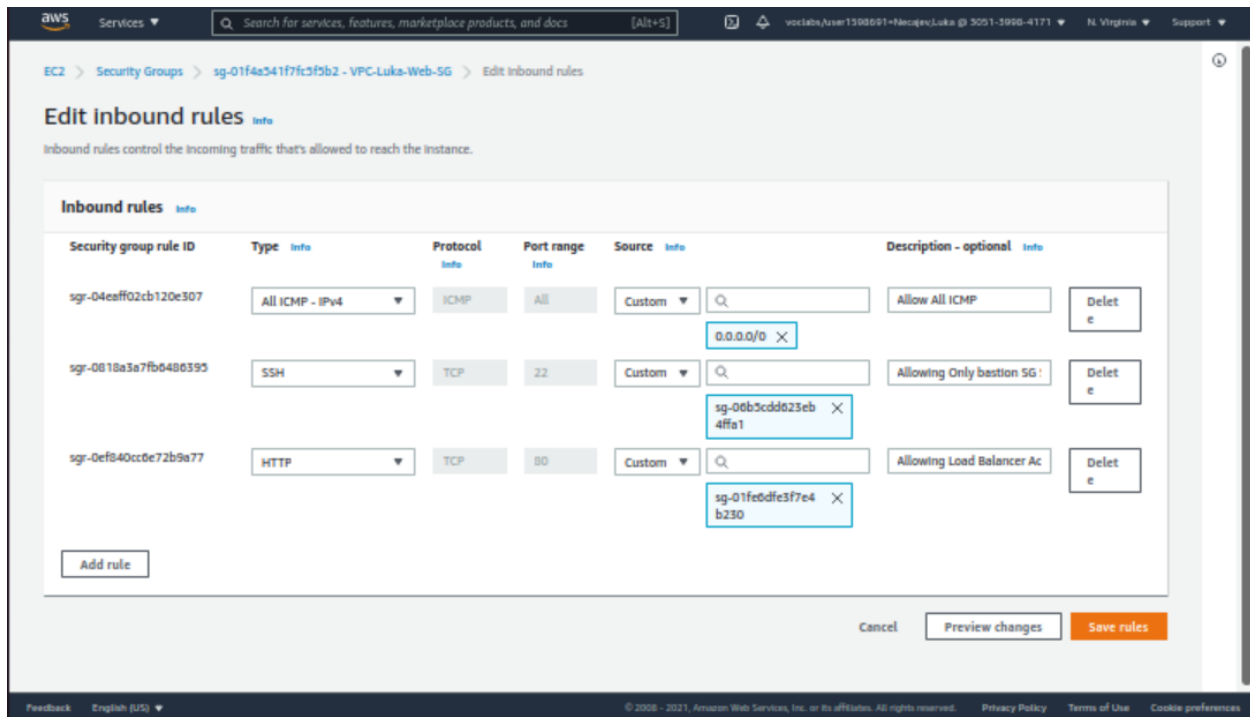
## Application Load Balancer Security Group



The Application Load Balancer Security Group allows only HTTP traffic from the internet into our system. Since we want all users to be able to access our website we will allow all IP's into our system. Therefore, I am allowing only port 80 allowed into my Application Load Balancer Security Group



## Web Server Security Groups



Since we are hosting a website, we want traffic requests for HTTP to be accepted from the Application load balancer. Therefore, we have to create an opening on port 80 and specify the Application Load Balancer Security Group

## Explaining the traffic flows

The diagram below depicts different traffic flows (red, blue, and grey) related to the deployment, configuration, and operation of our web site.

Explain all the flows in as much details as possible. Refer to the following questions:

### RED Line Explanation

Step one of the red line is sending a message to the NAT Gateway saying I need to access to download the httpd files . The NAT Gateway is used to convert our web instances private ip's into a public IP. So for step one, Our private ip will be taken and converted into a public IP by the NAT gateway which is in our public subnet and in our public routing table. This brings us to step two of our image. Now that the IP is public we can send it to the Internal Gateway which will allow public IP's access to the internet. Once at step 3 the packets will be sent from our Interior gateway to the yum data server which hosts the httpd files. Step 4, Once the files are retrieved it will send them back to the Interior gateway. Once at the gateway it will establish a connection with the nat gateway and start sending the files back to the NAT gateway. Step 6, the files are translated back into a private ip and sent to the web server. The webserver takes the files and puts them in a folder. Once that is done the webserver will closet the connection.

### Red line quesitons

- At what stage of application lifecycle will this flow happen?

This flow will happen after you finish setting up all of the equipment in the picture, meaning the Bastion host NAT GW, IGW, Application Load balancer(ALB) and the three security groups for ALB, Webserver and the Bastion host. Note we will also need to create Private and public routes so we can reach all of the devices we need to get to. So once all of that is created, then the lifecycle can start when we need to create our web instances.

- How/what are the users in this flow

The users in the flow are the Webserver, the Nat Gateway, Internal Gateway and the outside source YUM downloads

- What kind of traffic is it (web content, systems administration etc.)

The Red line is Downloading packages for

### **Blue line Explanation**

Step one, the user clicks the url of our load balancer. The user will be directed to the Internal Gateway. The internal gateway will provide a Public ip and establish communication with the Application load balancer. Step 2 the application load balancer will take the ip and send the packet to whatever webserver has less requests for data will be the one it is directed to. The ALB will direct the public packet directly to the webserver. Step 4 once the data has reached the webserver, the webserver will send the content it is hosting on its webserver back through the ALB using the ip it had received as the destination and the private one as its source. Step 5 the IP of the ALB replaces the source ip and the destination address is still the users laptop ip. Once ready the ALB will send the data to the Interior gateway and will present the user with the requested website/data.

### **BLUE**

- At what stage of application lifecycle will this flow happen

This will be a user who is trying to get access to your website. This will happen at the end of the lifecycle when everything is deployed and all of the security groups, load-balancers, NAT and IGW's are set up.

- How is this flow triggered?

The flow is triggered by a user going on their laptop and trying go to your url. The user will be directed to the Internal Gateway. This Interior gateway has a security group allowing only HTTP traffic. This is due to the fact that we do not want any other information to be allowed into our private network. The IGW only has access the ALB's Security group, meaning that only 'equipment' associated with the ALB security group. The only device in the ALB security group is our load balancer. Once the Load balancer receives this request it will direct it to the appropriate web sever based on the amount of requests it has gotten, making sure both servers are equally balanced in workload. Once the servers receive this request to get the HTTP data request, we send a response back through our Application load balancer. At the load balancer it will get the session id and pass it back to the IGW to send it out of our VPC.

- How/what are the users in this flow

The First user is the actual user who is requesting the website from their laptop, phone pc..., He has a url that he got from google, and clicks it. The url is going to be directed to our IGW as it is set up to accept any incoming or outgoing packets into our system. The IGW will notice that it is requesting the ALB's IP address and will direct it to the ALB. Once at the ALB, the ALB will direct the traffic to one of our two webserver, depending on availability of the servers. Once the packet is sent to the private addressed instances (webserver) they will return the HTTP webpage that the user is requesting to the ALB. The ABL will take that session ID and pass it back to the IGW which will route the packet back to the user.

- What kind of traffic is it (web content, systems administration etc.)

The Blue line is WEB Content! HTTP packets that are being requested from the user's laptop so you can display the web-page on their pc's.

### **Gray Line Explanation**

Step 1 for the gray line is for the aws admin to decide that he wants to access a web instance's console and change some of the settings of the webserver. This will require us to go through the bastion as it is the only server that has access to the public subnet as well as has access to the private subnets that the users are in. So Step 1 is to ssh into the bastion host. Step two is to ssh from the bastion host into the web instance. Step 3 is when the admin finishes his session in the web instance, he finished changing the settings and wants to go back to his pc. Step 4 he logs out of the web session and bastion session back to his own laptop.

### **GREY**

- At what stage of application lifecycle will this flow happen

This flow can be the first flow that will happen in this image as this when our AWS administrators are setting up or changing settings within a specific webserver/ec2 instance. This flow is used whenever we want to ssh into our instances to change settings in them.

- How is this flow triggered

This flow is triggered by an aws admin or someone who is authorized wants to change something within the webserver. This flow is triggered by someone logging into the bastion host. Once they are in the bastion instance, they will then be able to ssh into the specific instances that are in the private networks due to our webserver security groups allowing ssh only to the bastion host security group! Once the admin is in the webserver they can change content on the server and then exit the system and view the changes. This might require a reboot of the instance depending on the task.

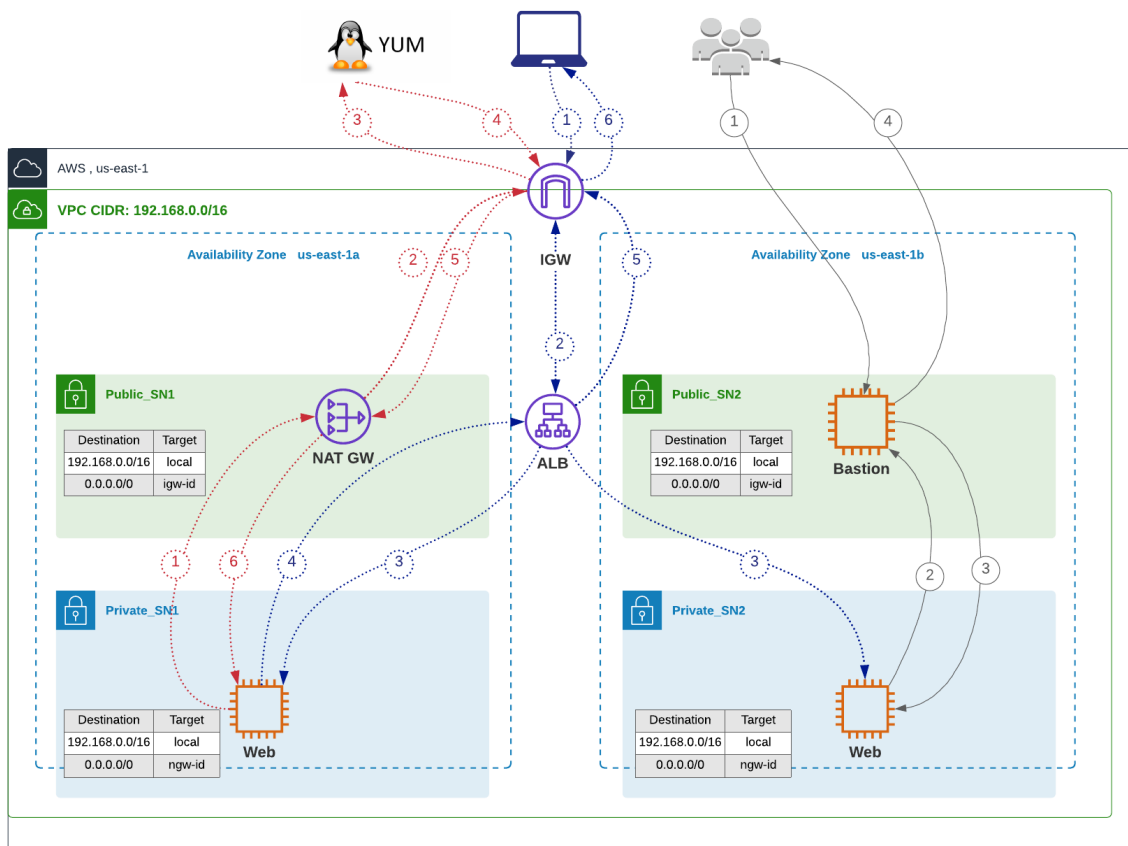
- How/what are the users in this flow

The users in this flow is the Aws admin, the initiation of the ssh session. The bastion host, the median that is allowed to communicate to the private instances created as webserver using port 22 SSH. Then the third user is the Webserver itself.

- What kind of traffic is it (web content, systems administration etc.)

The type of traffic is administrative content as we are most likely administering commands and changes to the webserver. The type of traffic allowed is SSH traffic, using port 22.

**Note:** CIDR ranges and routes in this diagram are for demo purposes only. Those are **not** the CIDR ranges you must deploy.



## Grading

### General grading notes:

- Keep your explanations relevant and concise.

Task	Artifacts to submit	Points
Deployment of VPC, public and private subnets, route tables, NAT GW, and Internet GW	<ul style="list-style-type: none"> <li>AWS CLI used to create the infrastructure (if deployed with AWS CLI)</li> <li>Screenshots of provisioned infrastructure: <ul style="list-style-type: none"> <li>VPC</li> <li>Subnets (make sure CIDRs, AZs and VPC are visible)</li> <li>Route tables</li> <li>NAT GW</li> </ul> </li> </ul>	20

	<ul style="list-style-type: none"> <li>○ Internet GW</li> <li>• Explanation of the rationale behind route tables and subnets definitions</li> </ul>	
Security groups definition	<ul style="list-style-type: none"> <li>• AWS CLI used to create the infrastructure (if deployed with AWS CLI)</li> <li>• Screenshots of inbound rules for all the SGs (make sure SG name is visible)</li> <li>• Explanation of the rationale behind SG definitions</li> </ul>	10
ALB and listener deployment and configuration	<ul style="list-style-type: none"> <li>• AWS CLI used to create the infrastructure (if deployed with AWS CLI)</li> <li>• Screenshots of ALB listener rules configuration</li> <li>• Explanation of the rationale behind the listener and rules definition</li> </ul> <p>Note: make sure screenshot shows that ALB is deployed into correct VPC/subnets/AZs</p>	10
Web target group (not auto-scaling) deployment, traffic served at <a href="http://&lt;ALB DNS&gt;/">http://&lt;ALB DNS&gt;/</a>	<ul style="list-style-type: none"> <li>• AWS CLI used to create the infrastructure (if deployed with AWS CLI)</li> <li>• Screenshot of the target group health checks</li> <li>• Screenshot of the web browser <a href="http://&lt;ALB DNS&gt;/">http://&lt;ALB DNS&gt;/</a></li> <li>• Screenshot of ssh access to web servers</li> </ul> <p>Note: make sure screenshot shows that EC2 are deployed into correct VPC/subnets/AZs</p>	10
Web-ASG1 target group (auto-scaling) deployment, traffic served at <a href="http://&lt;ALB DNS&gt;/target1/">http://&lt;ALB DNS&gt;/target1/</a>	<ul style="list-style-type: none"> <li>• AWS CLI used to create the infrastructure (if deployed with AWS CLI)</li> <li>• Screenshot of the target group health checks</li> <li>• Screenshot of the web browser <a href="http://&lt;ALB DNS&gt;/target1/">http://&lt;ALB DNS&gt;/target1/</a></li> <li>• Screenshot of ssh access to web servers</li> </ul> <p>Note: make sure screenshot shows that EC2 are deployed into correct VPC/subnets/AZs</p>	30
Web target group (not auto-scaling) deployment, traffic served at <a href="http://&lt;ALB DNS&gt;/target2/">http://&lt;ALB DNS&gt;/target2/</a>  <b>BONUS QUESTION!</b>	<ul style="list-style-type: none"> <li>• AWS CLI used to create the infrastructure (if deployed with AWS CLI)</li> <li>• Screenshot of the target group health checks</li> <li>• Screenshot of the web browser <a href="http://&lt;ALB DNS&gt;/target2/">http://&lt;ALB DNS&gt;/target2/</a></li> <li>• Screenshot of ssh access to web servers</li> </ul>	10



	Note: make sure screenshot shows that EC2 are deployed into correct VPC/subnets/AZs	
Explain the traffic flows	<ul style="list-style-type: none"><li>• Explanation of the traffic flows</li><li>• Clearly state what happens at every step</li></ul>	20