M-4, 1st Floor, Old DLF Colony, Sector 14, Gurugram, Haryana 122001

Project report on AWS

Name

Jairam J S

Email ID

jsjairam01@gmail.com

Acknowledgement

Not all professionals do their work by themselves. Although they can be as prolific or as adept in their respective fields, they will still need assistance one way or another. For instance, writing a body of work takes a lot of research. They often depend on their assistants or subordinates to gather information about the subject matter. Aside from the research people, other individuals can also receive credit for their contributions to the writer's work.

CONTENTS

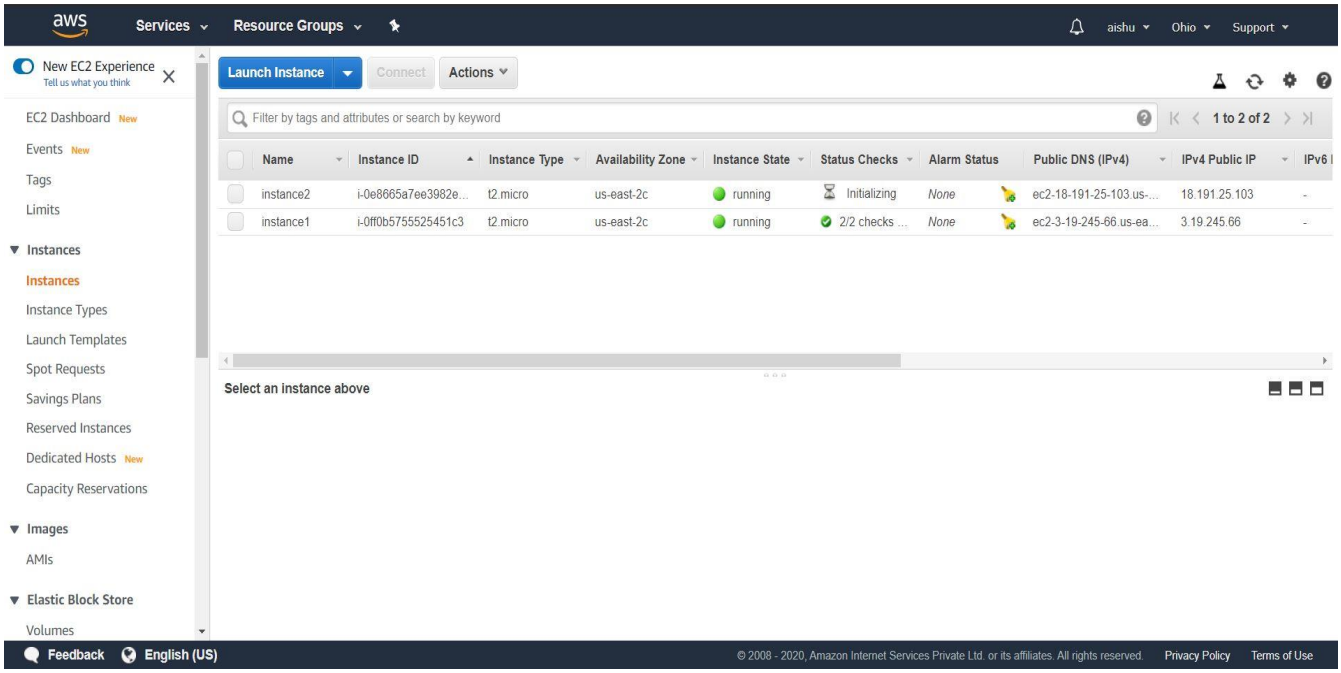
s.no	Project Name	Page.no
1.	Using the EBS Volumes to attach multiple EC2 Instance	4-16
2.	Accessing instance between different VPCs.	17-28
3.	RDS Lab	29-42
4.	Cloud Formation Lab	43-73

S.no 1: Using the EBS Volumes to attach multiple EC2 Instance

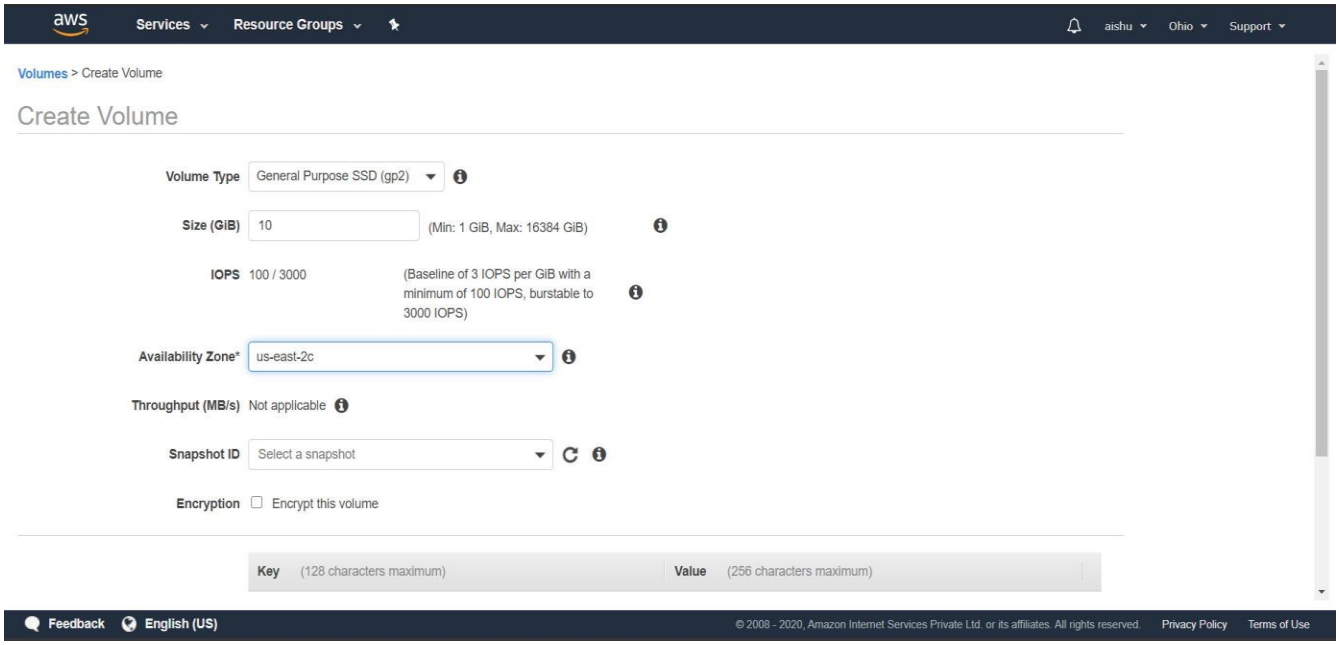
An EC2 instance can have multiple EBS volumes attached at the same time, but a single EBS Volumes cannot be associated with multiple EC2 instance at the same time. So we will see how to attach Volume, write some data into that EBS and detach and attach to Another EC2 Instance.

Important: Make sure your EC2 and EBS are in the same Availability Zone.

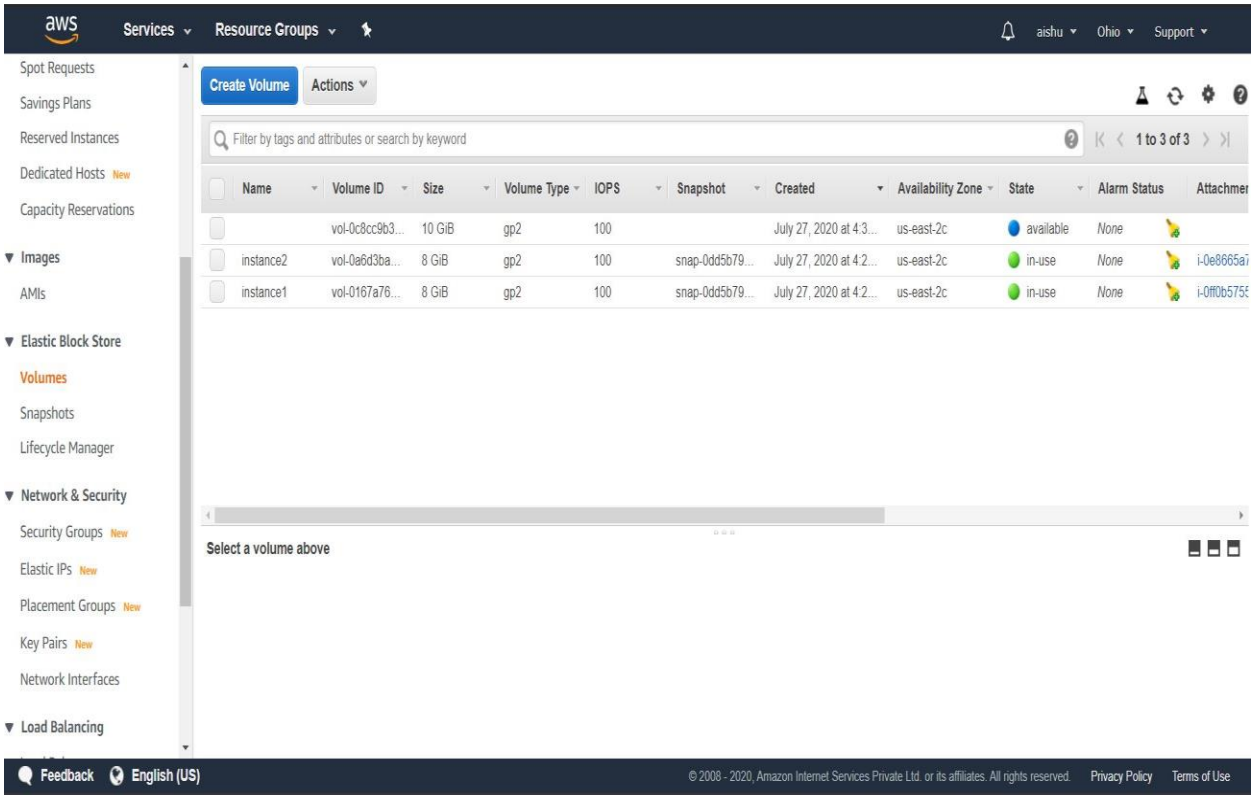
Step 1: Launch Two EC2 Instance in same Availability Zone

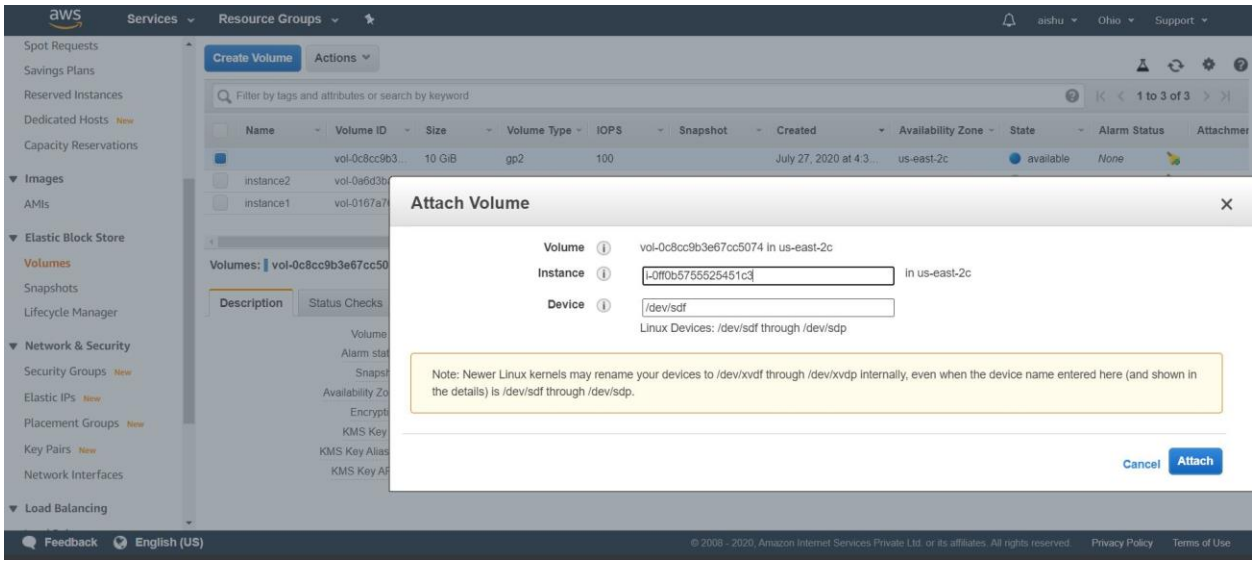


Step 2: Go to Volumes from the left pane and create one volume of any size. We will create a Volume of size 10GB in the same Availability Zone

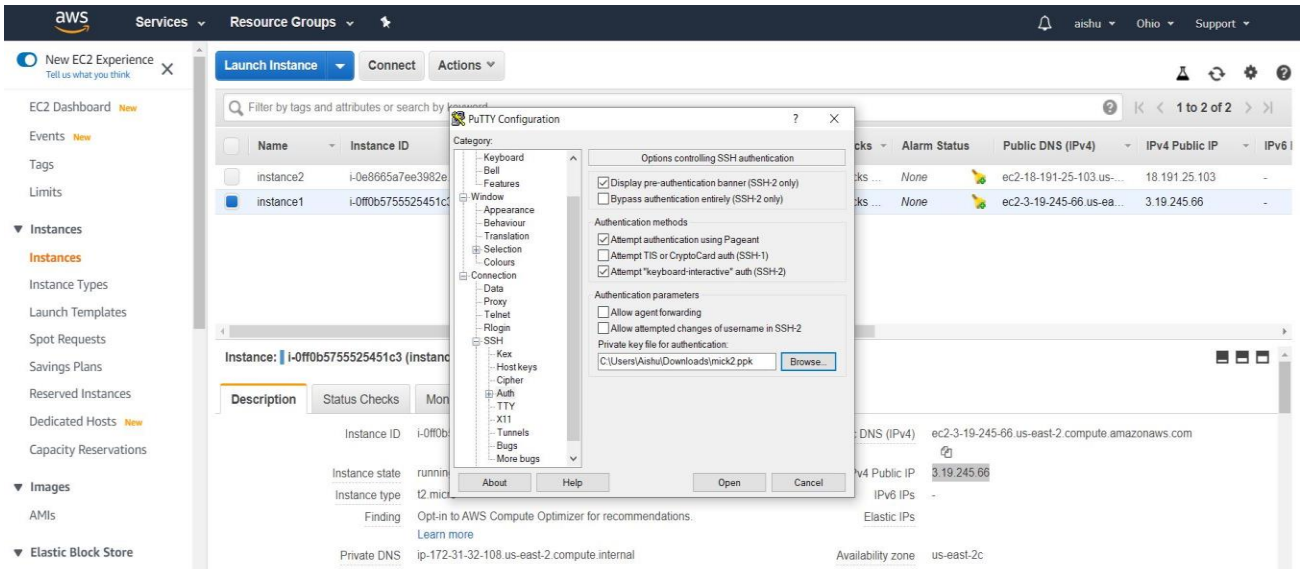


Step 3: Now to the newly create Volume, click on Action and attach to First Instance.

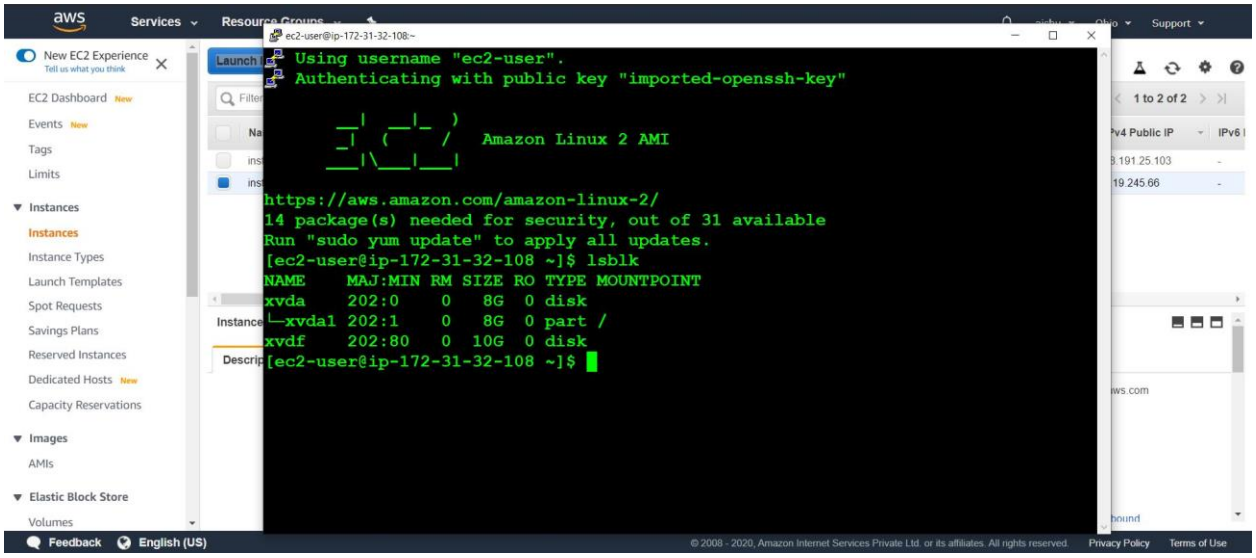




Step 4: Through putty connecting the instances



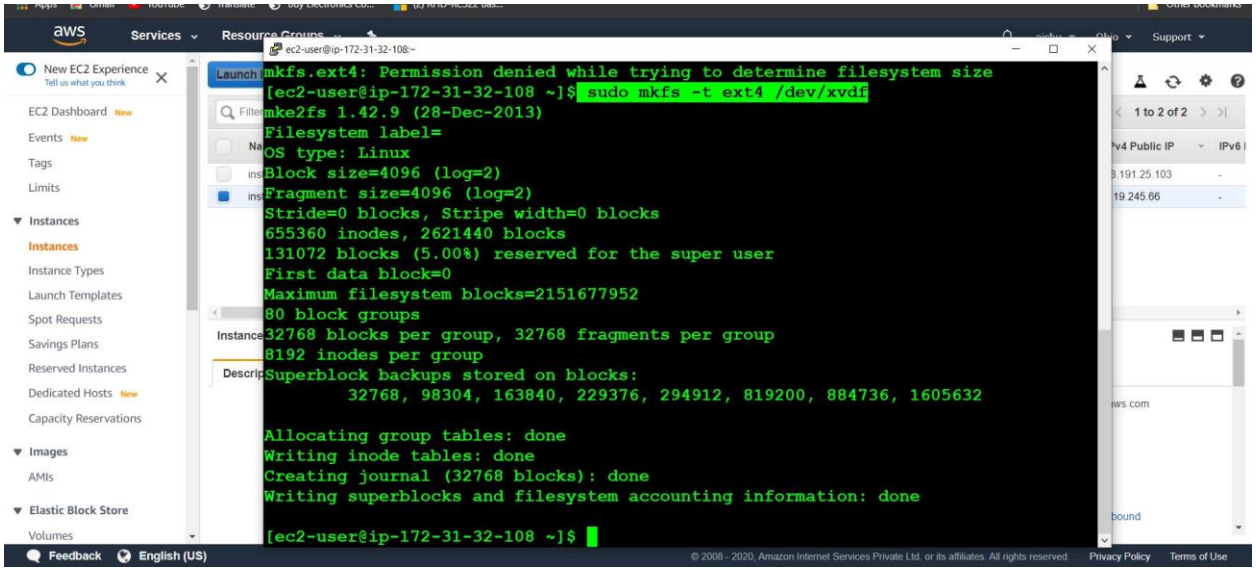
Step 5: Now SSH to your first Instance and type "***lsblk***" to list all the block device.



And here you can see, the newly created and attached 10GB volume is successfully attached. Remember, we have just created and attached volume, yet we need to mount that file system.

Step 6: Create a File system

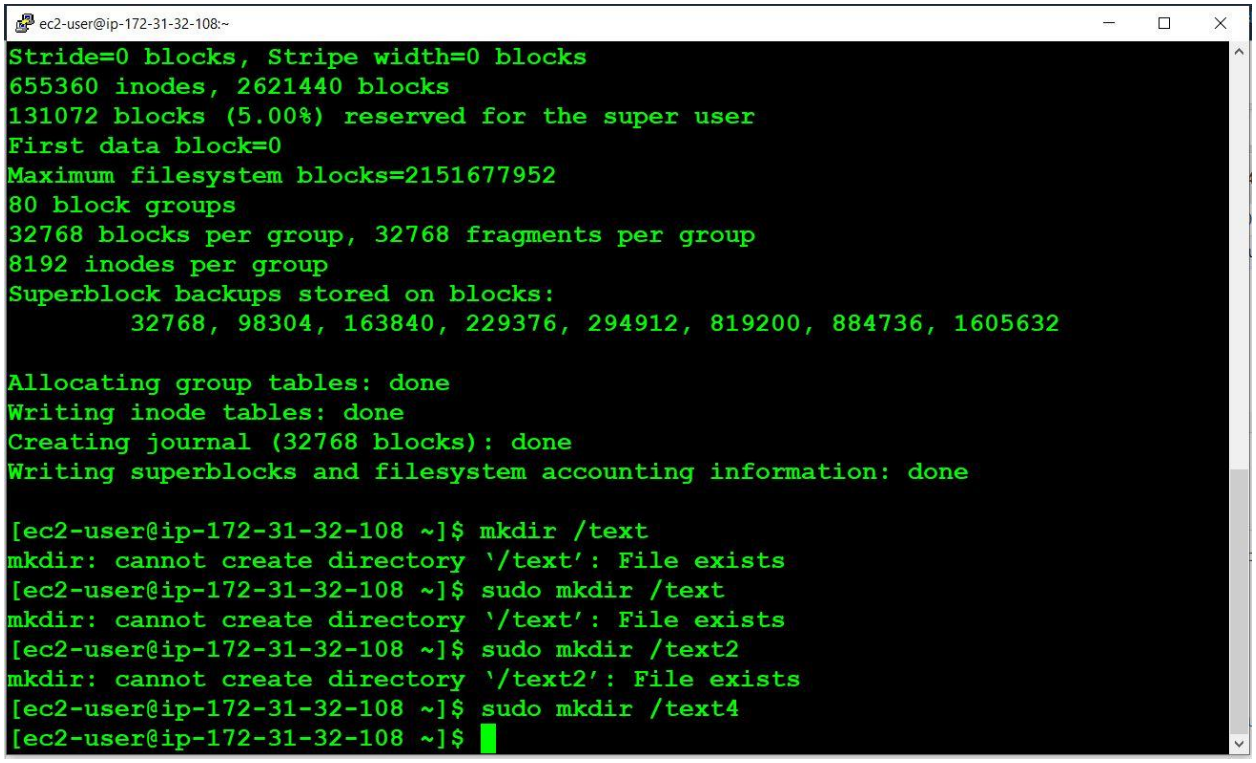
Type command ***" sudo mkfs -t ext4 /dev/xvdf***"



And it will create a EXT4 based file system in that volume device.

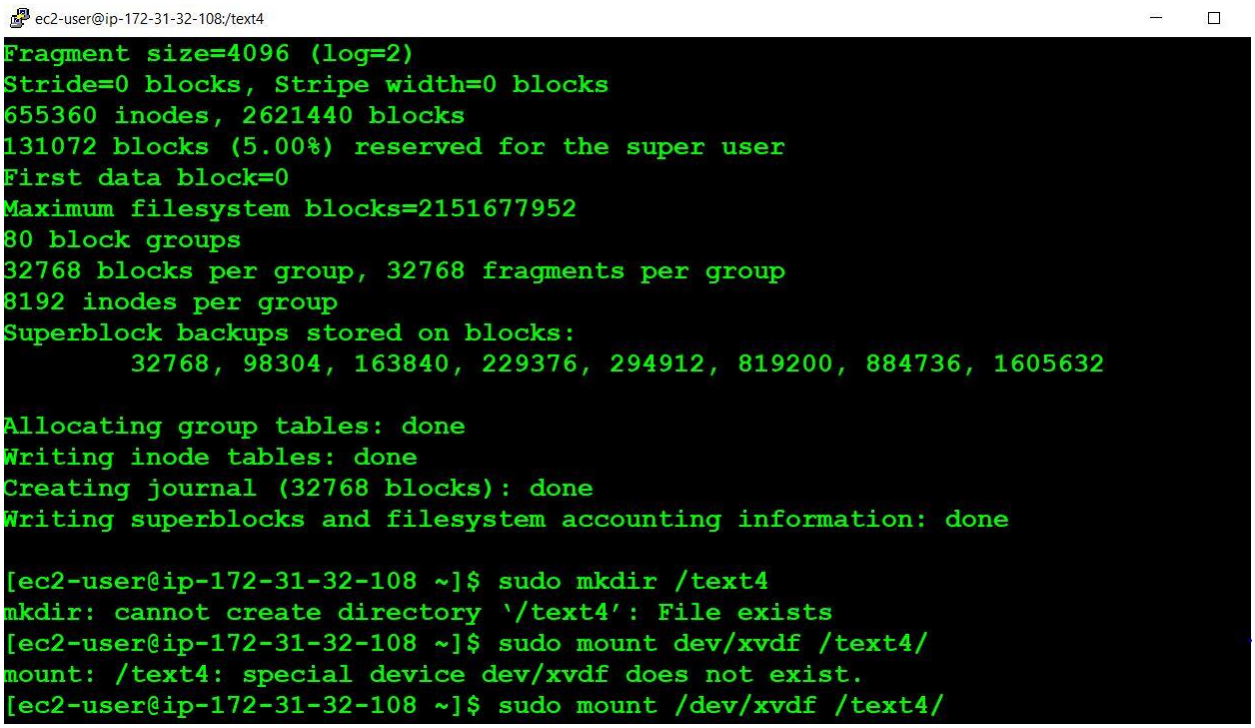
Step 7: Make folder/directory to mount then to Volume

Command: "***mkdir /text4***"



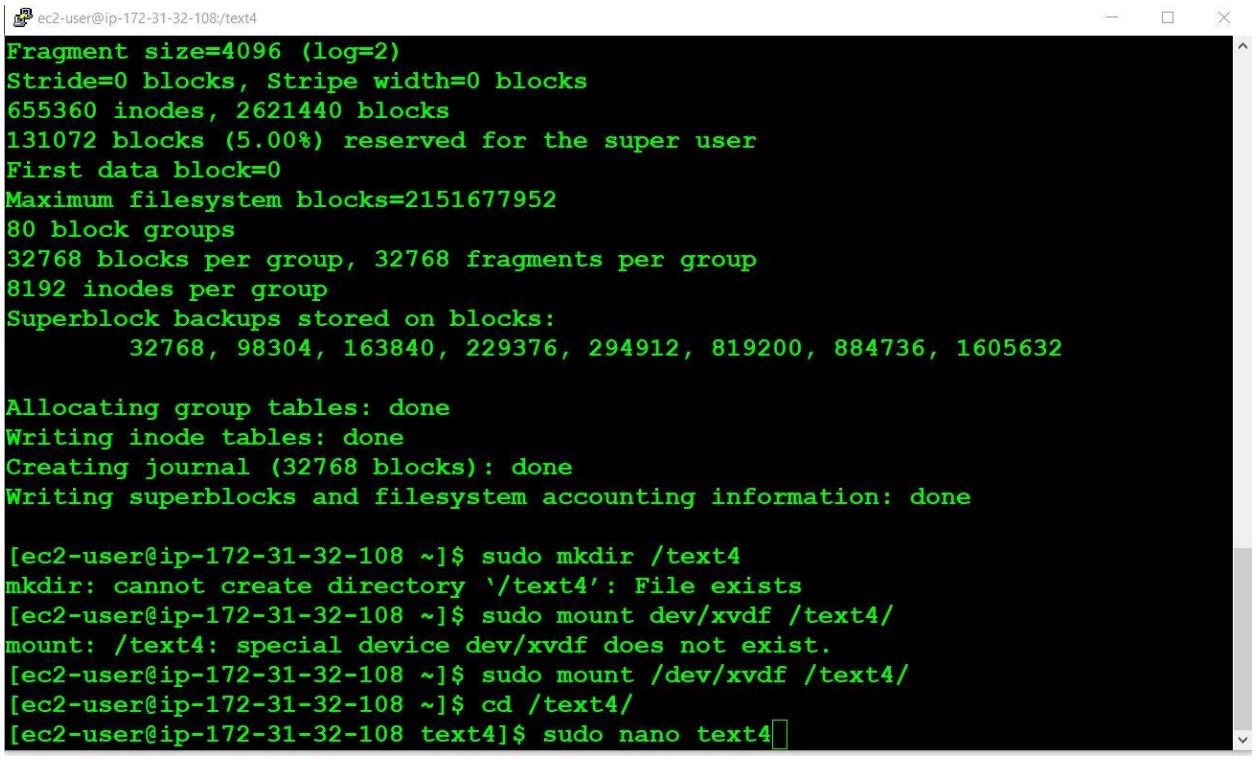
Step 8: Now mount this directory to volume.

Command : "***mount /dev/xvdf /foldername/***"



Now change the directory and move to newly created directrory

Command: ***cd /foldername***"

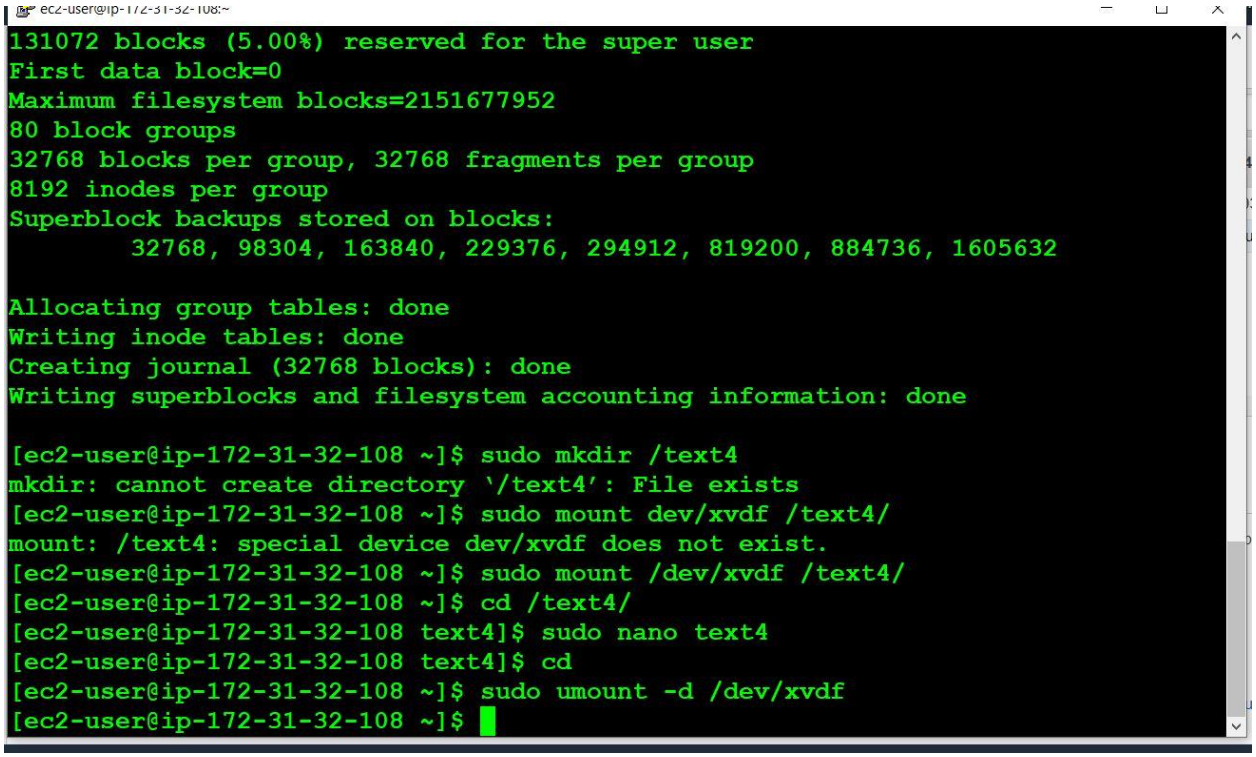


Step 9: Now here will make a text.txt file to check whether file exits when we detach and attach to another EC2 Instance.

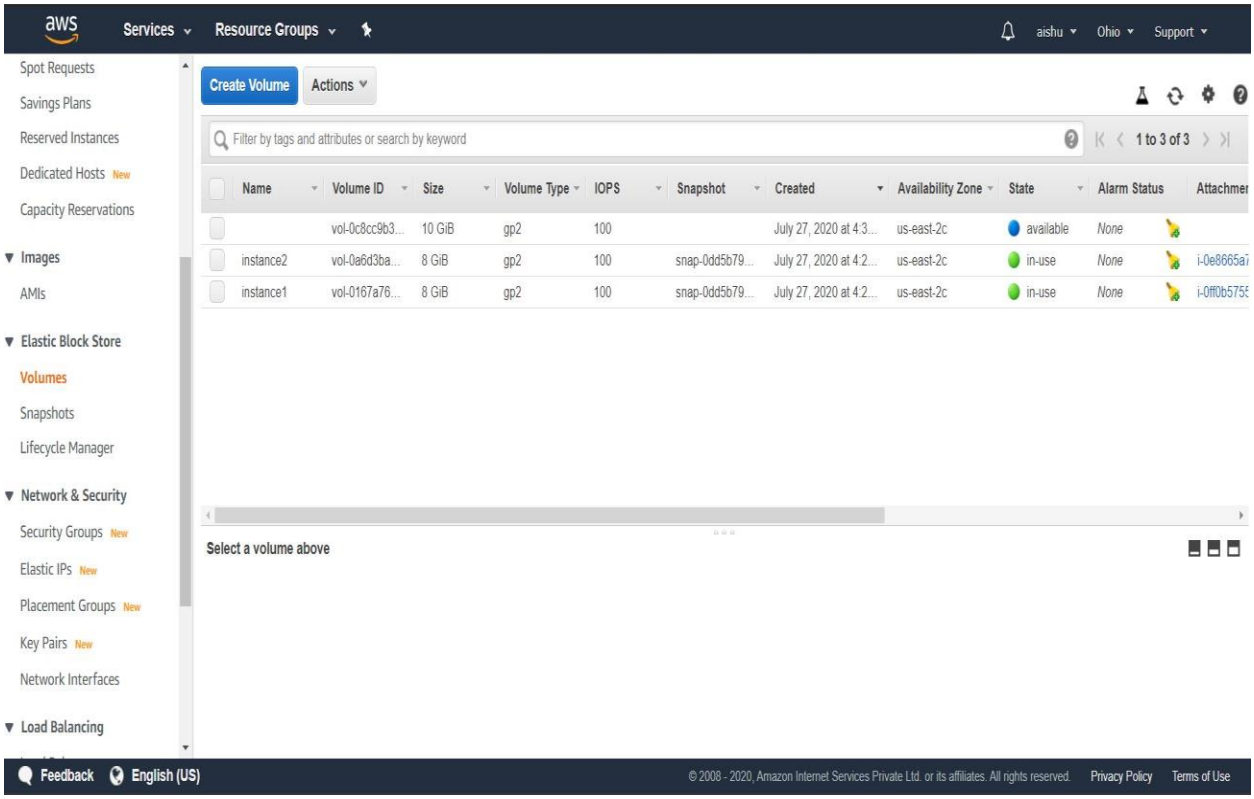
Step 10: Now let's unmount this volume from this EC2 Instance

Command: "***cd ..***

umount -d /dev/xvdf"



Step 11: Now go to AWS Console and from the volume section, detach this volume from Instance1 and attach it to Instance2.



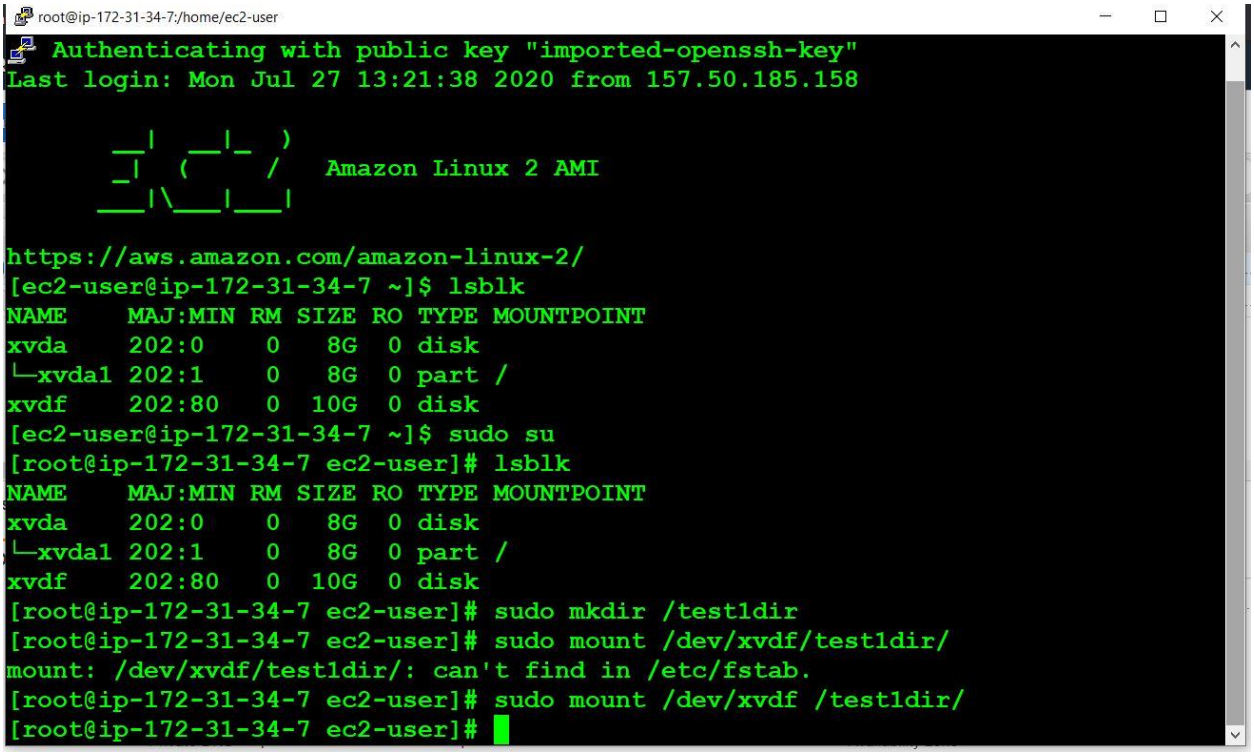
Step 12: Now SSH to Instance2 and check the volume by command "***lsblk***"

Here you can see, it shows successful attachment of volume.

Step 13: Now we need to again create a directory and mount to our volume.

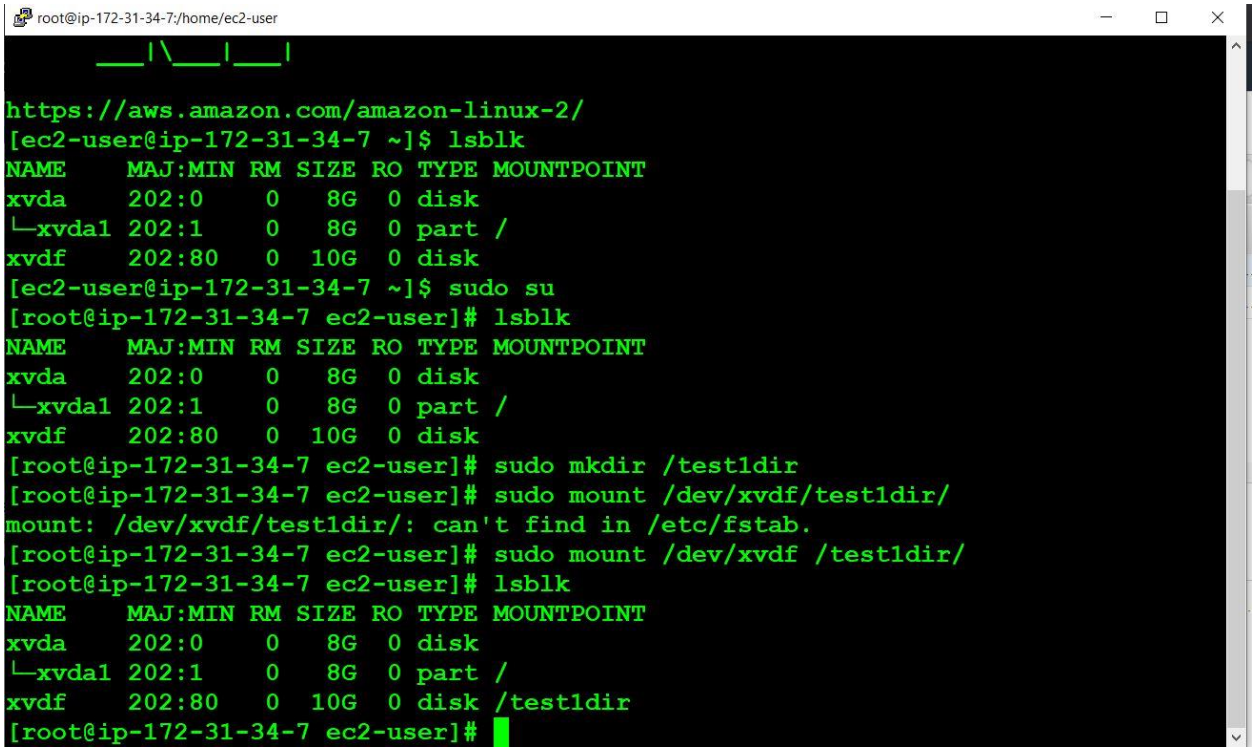
Command: ***" sudo mkdir /testdir***

Sudo Mount / dev/xvdf /testdir"



Step 14: Now to check the mount, type "***lsblk***" and test the mount point

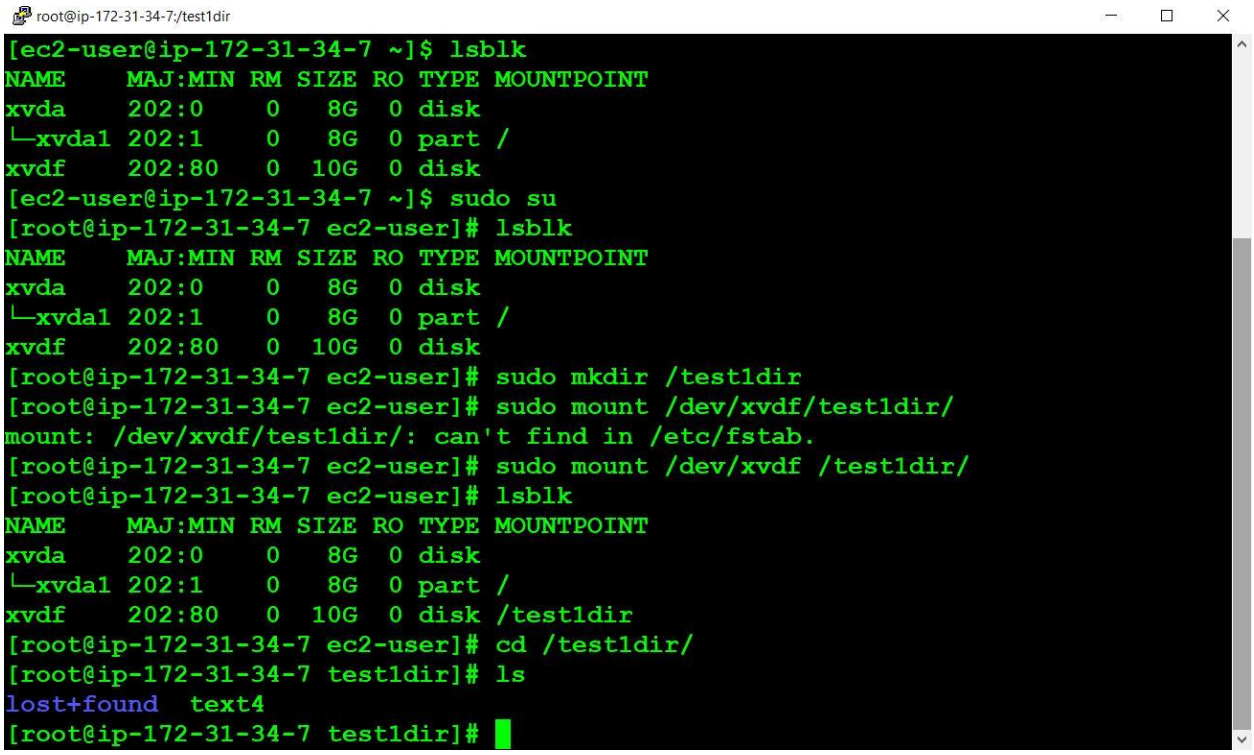
Shows successful mount to new volume.



Step 15 : Now move to directory and list the files.

Command : "***cd /testdir***

ls"

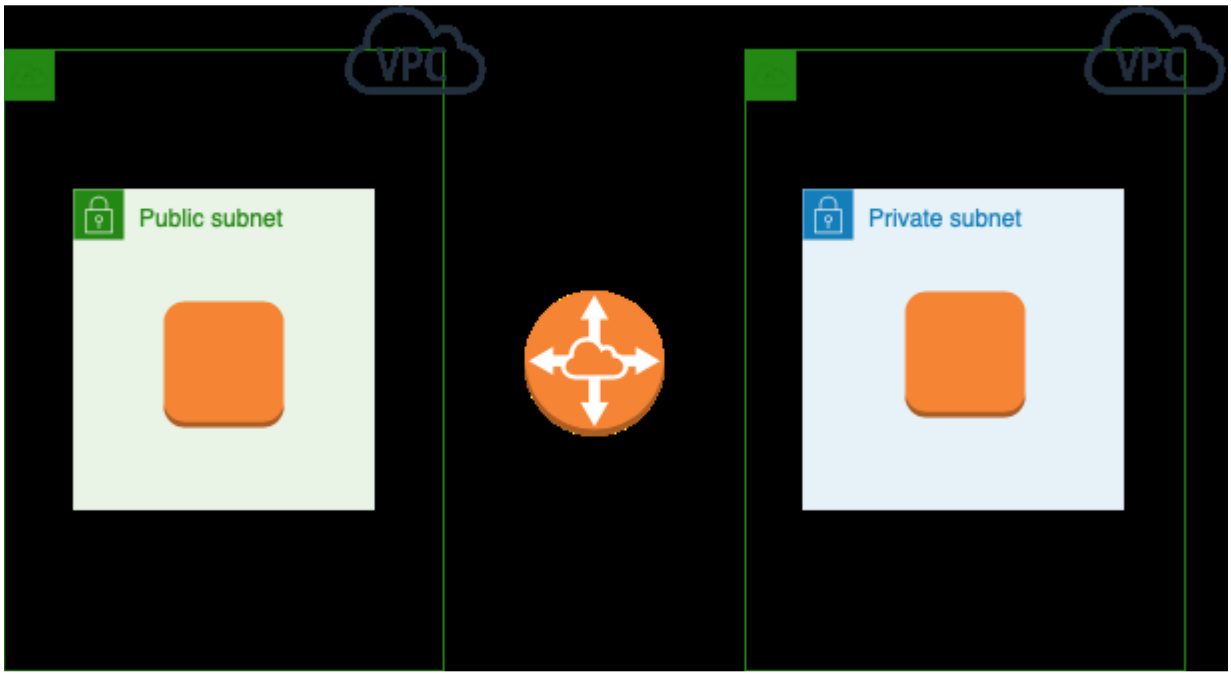


And here you can see file “text.txt” which we had create in Instance1

S.no 2: Accessing instance between different VPCs.

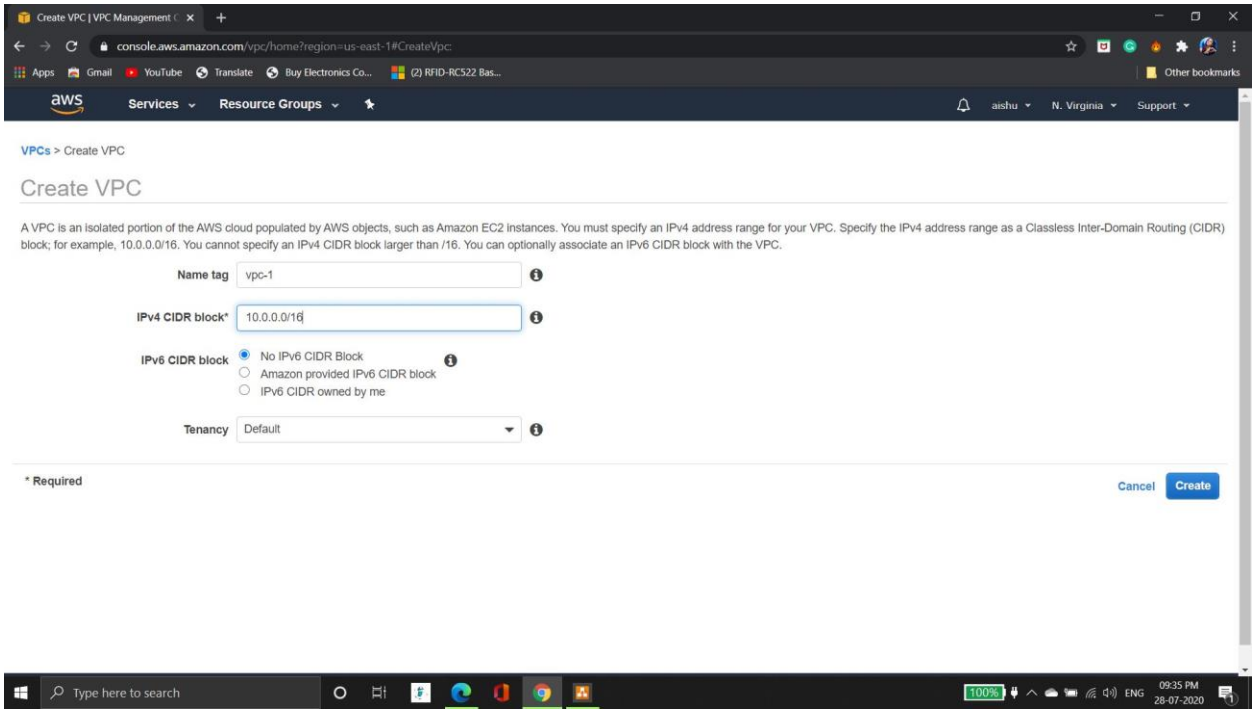
In this Case, we will create two VPC and we will then access instance from public VPC to the instance in private VPC

Architecture:

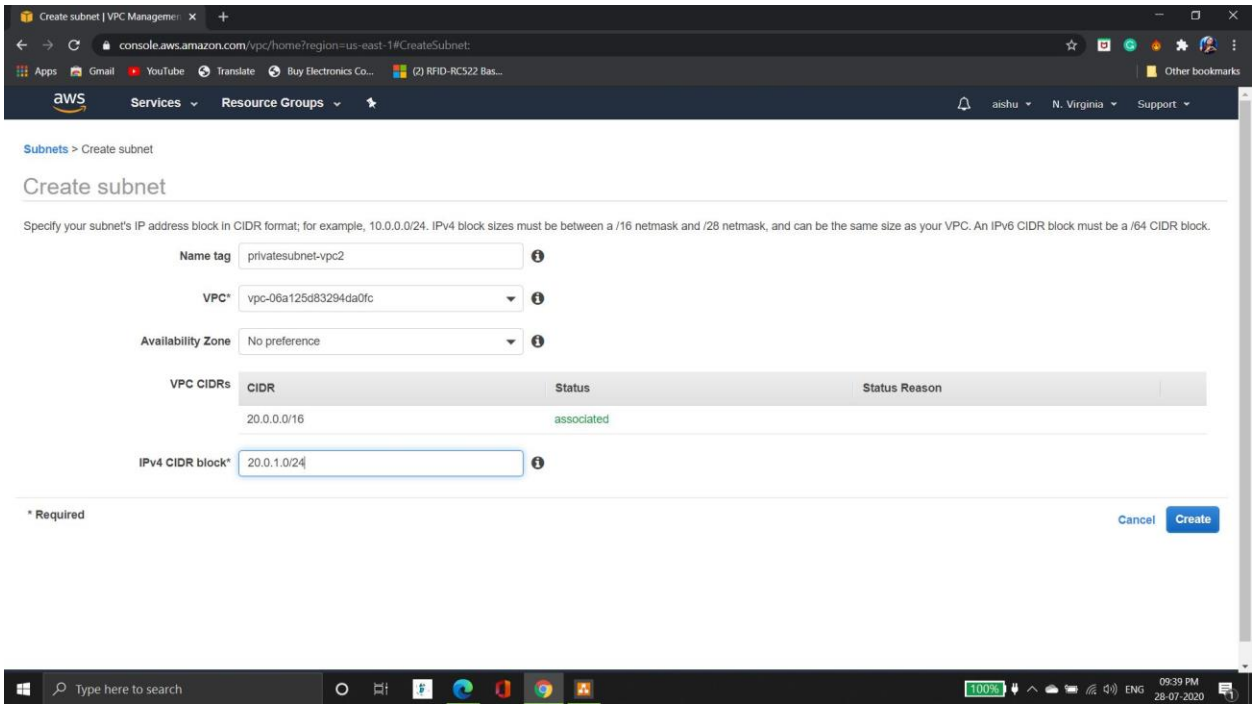


Step 1: Go to VPC Dashboard and select a region

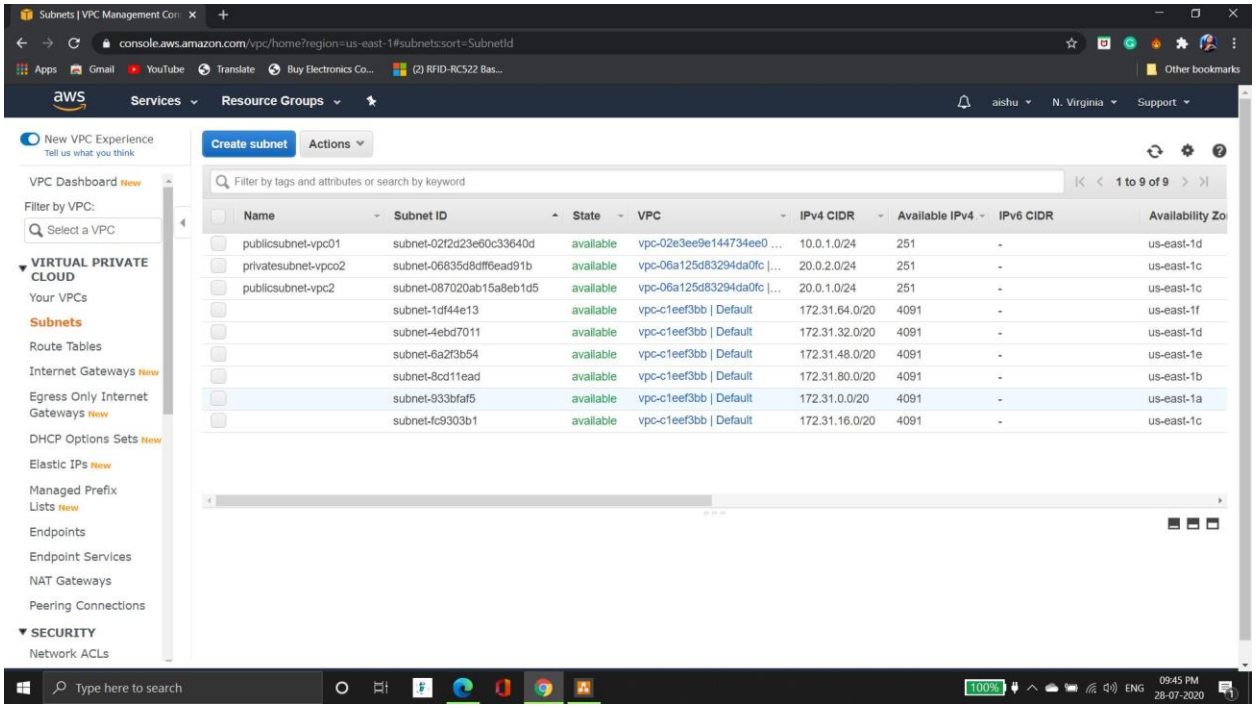
Step 2: Create first VPC



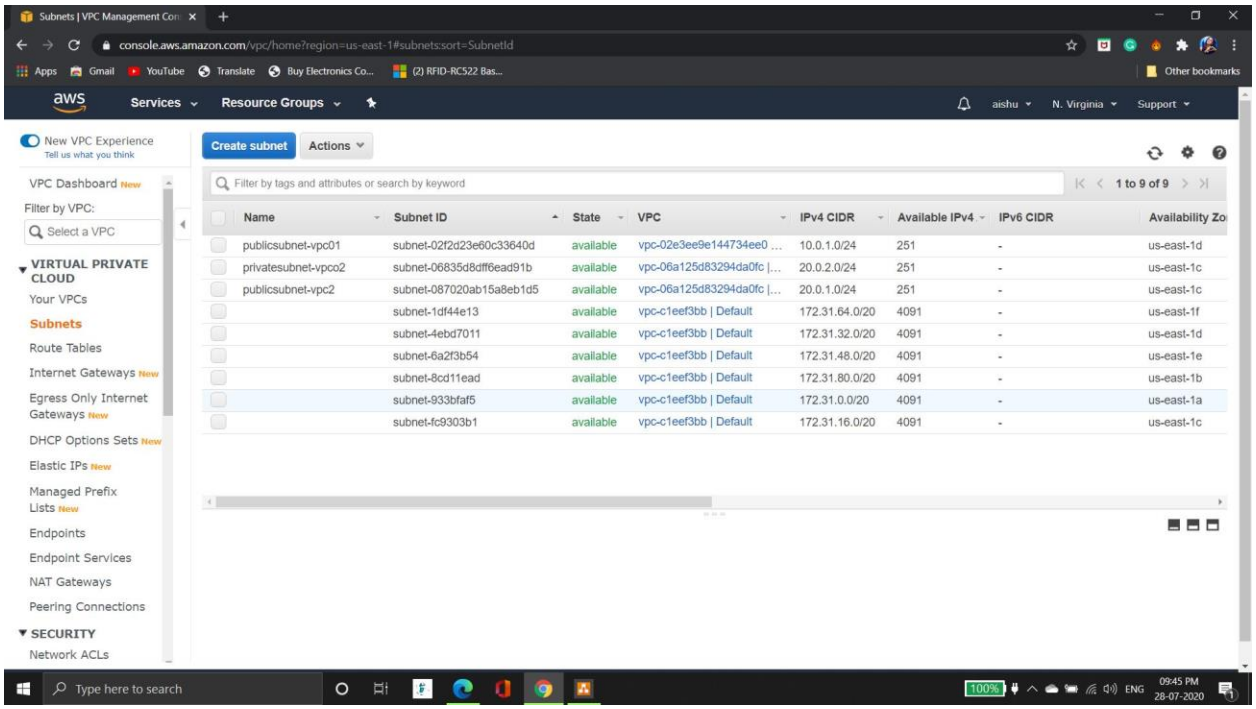
Step 3: Create second VPC



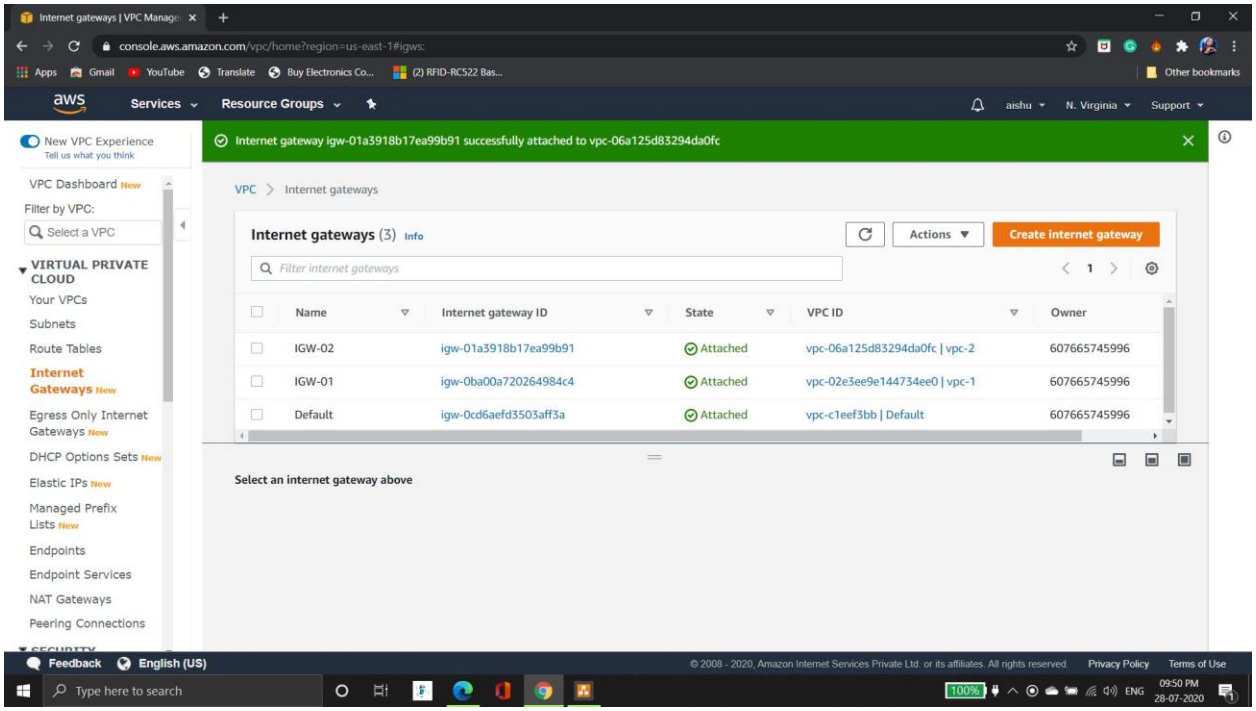
Step 4: Create one public subnet for VPC_01



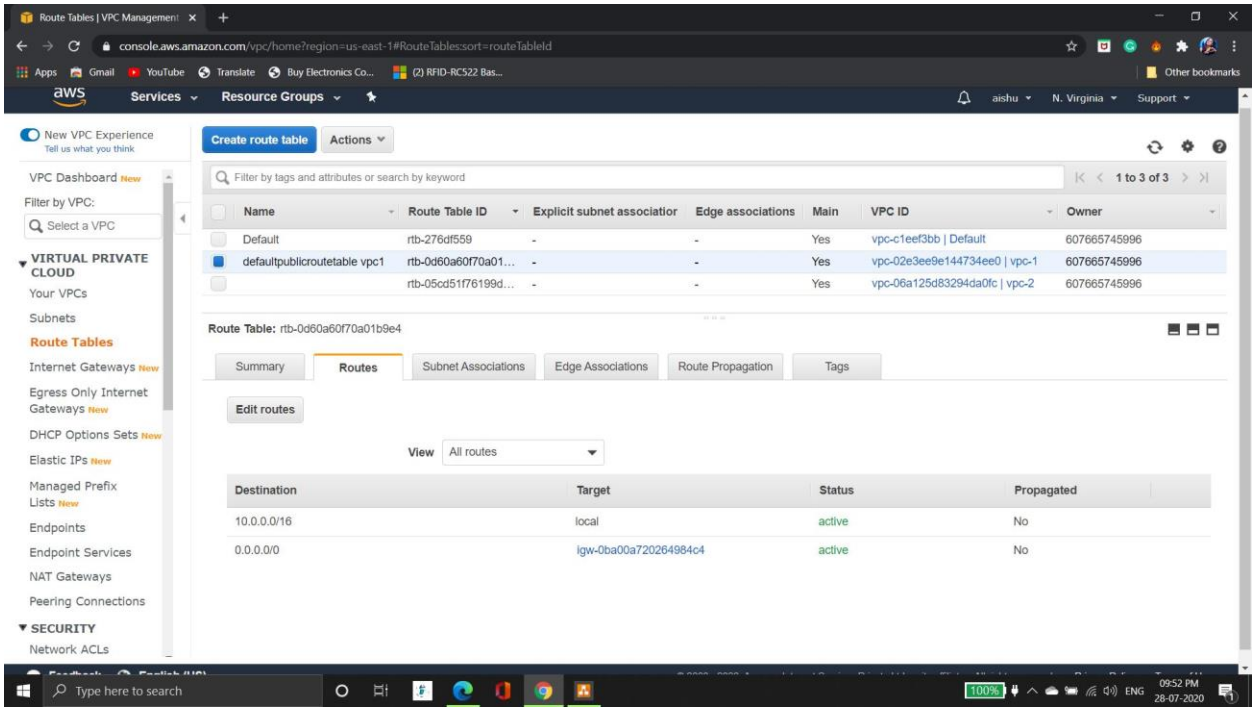
Step 5: Create One Public and One Private Subnet for VPC_02



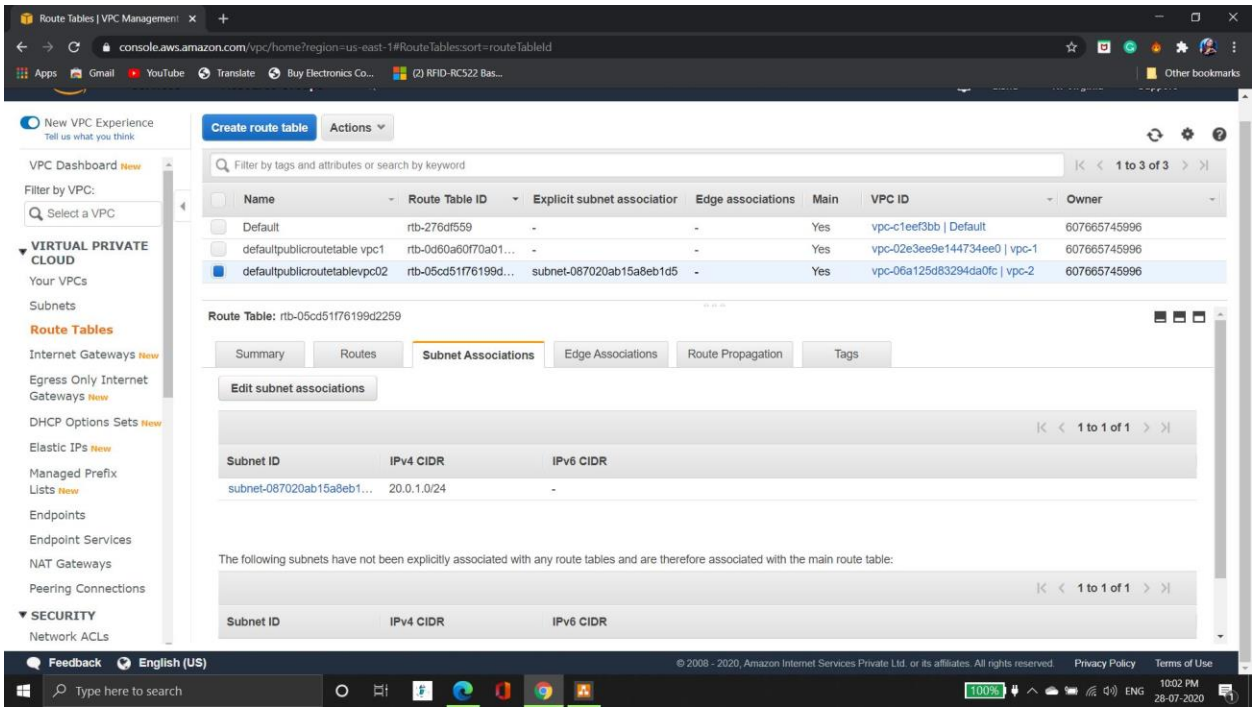
Step 6: Now create and attach Internet Gateway to both the VPCs



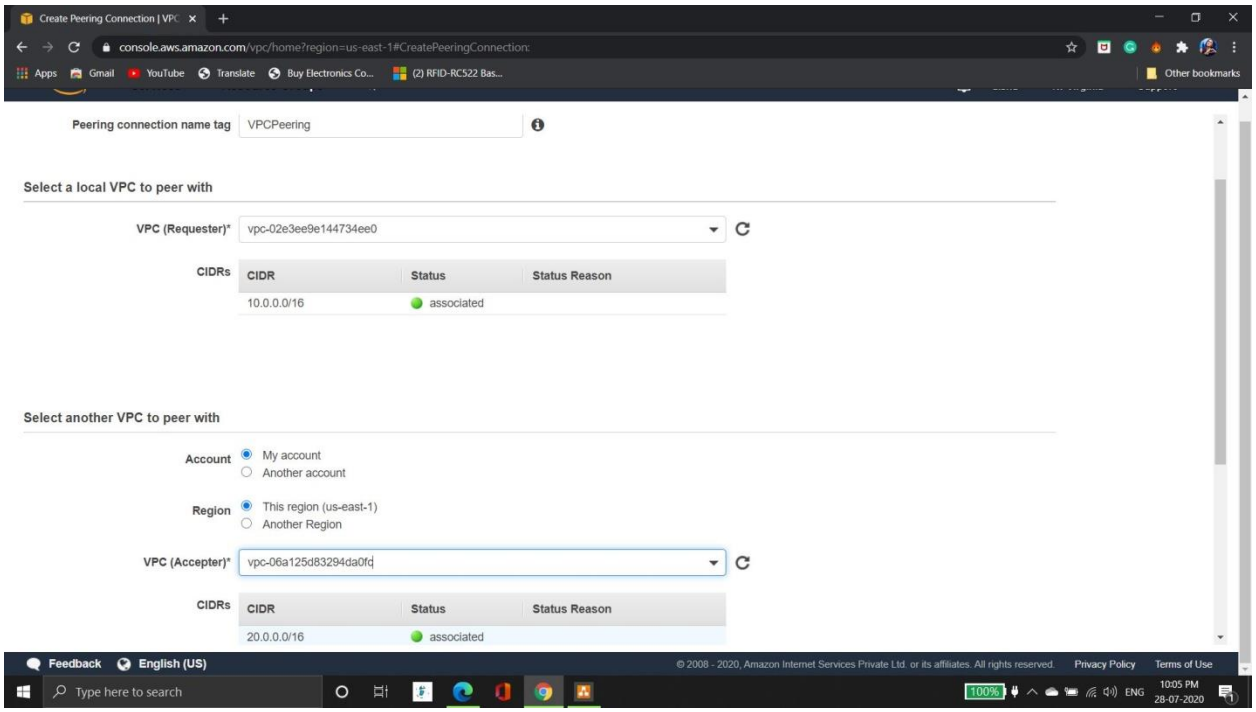
Step 7: Now we will attach this Internet Gateway to Default Route Table of VPC_01 and associate Public subnet



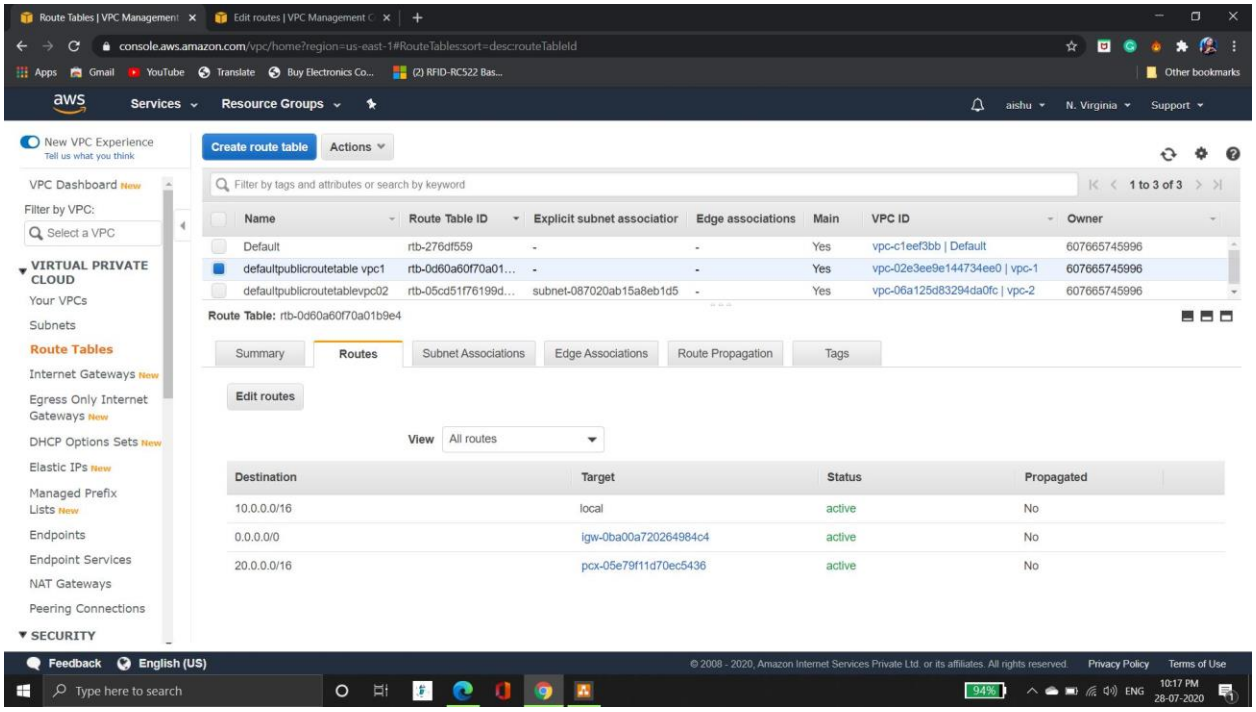
Step 8: Similarly, we will now associate Internet Gateway to Another default Route Table of VPC_02 and associate public subnet



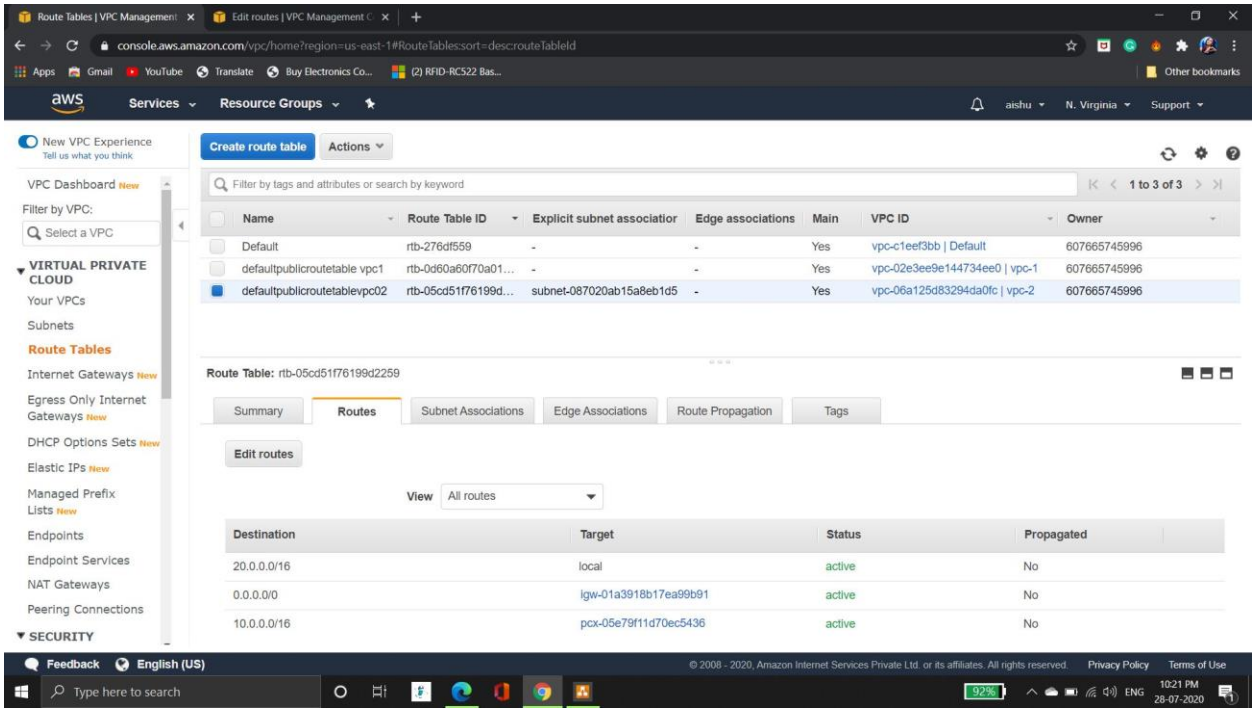
Step 9: Now go to VPC Peering connection setting in your left pane, and create a peering connection between both the VPCs



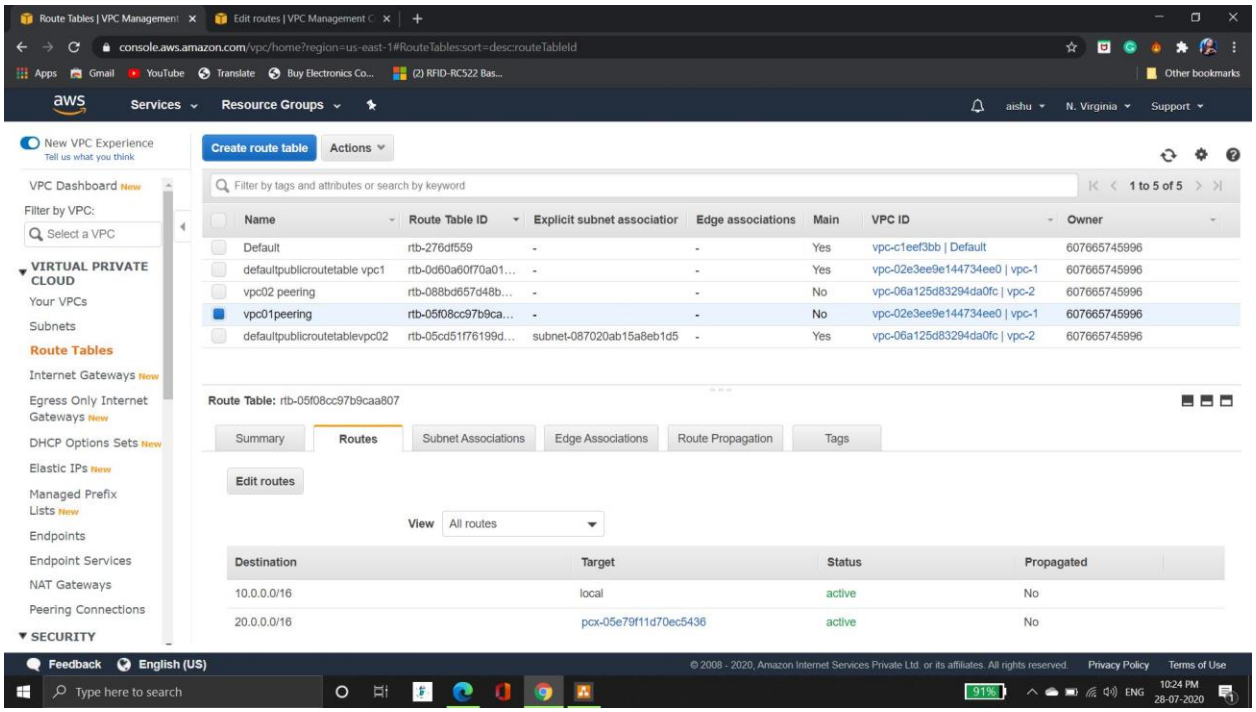
Step 10: Now, in the same route table of both the VPCs, add one more rule and add VPC Peering connection.



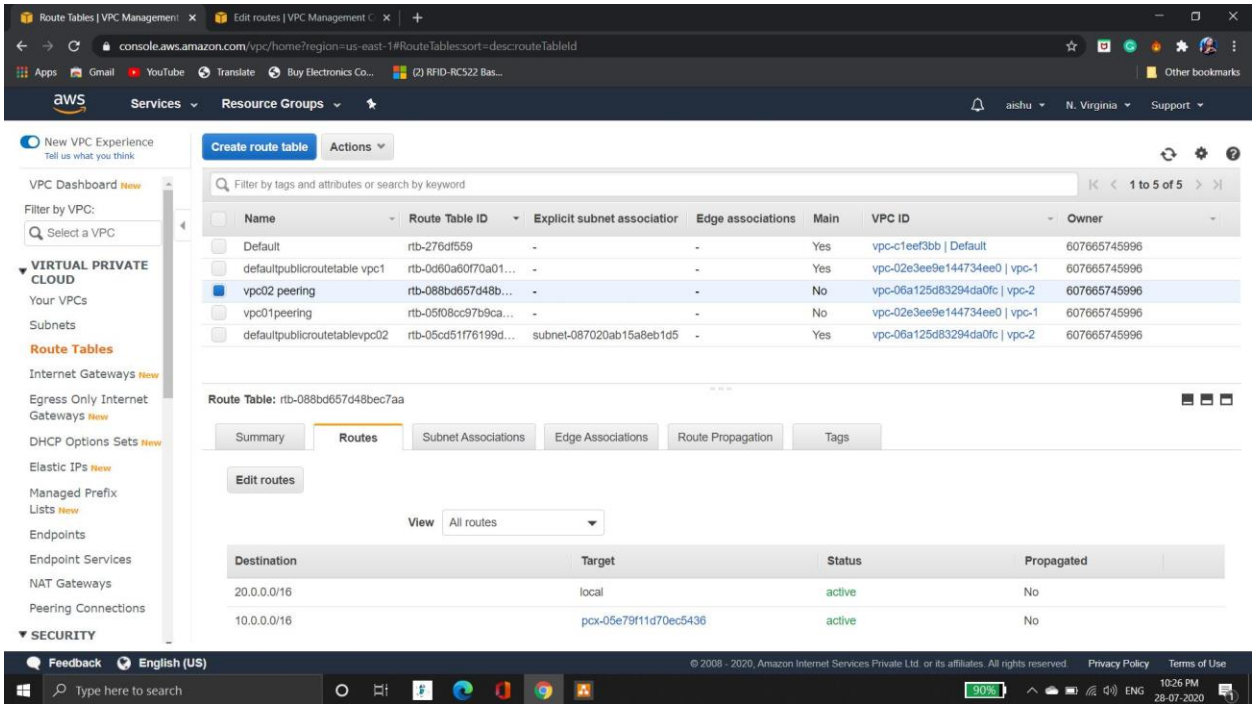
Similarly, to another Route Table of VPC_02



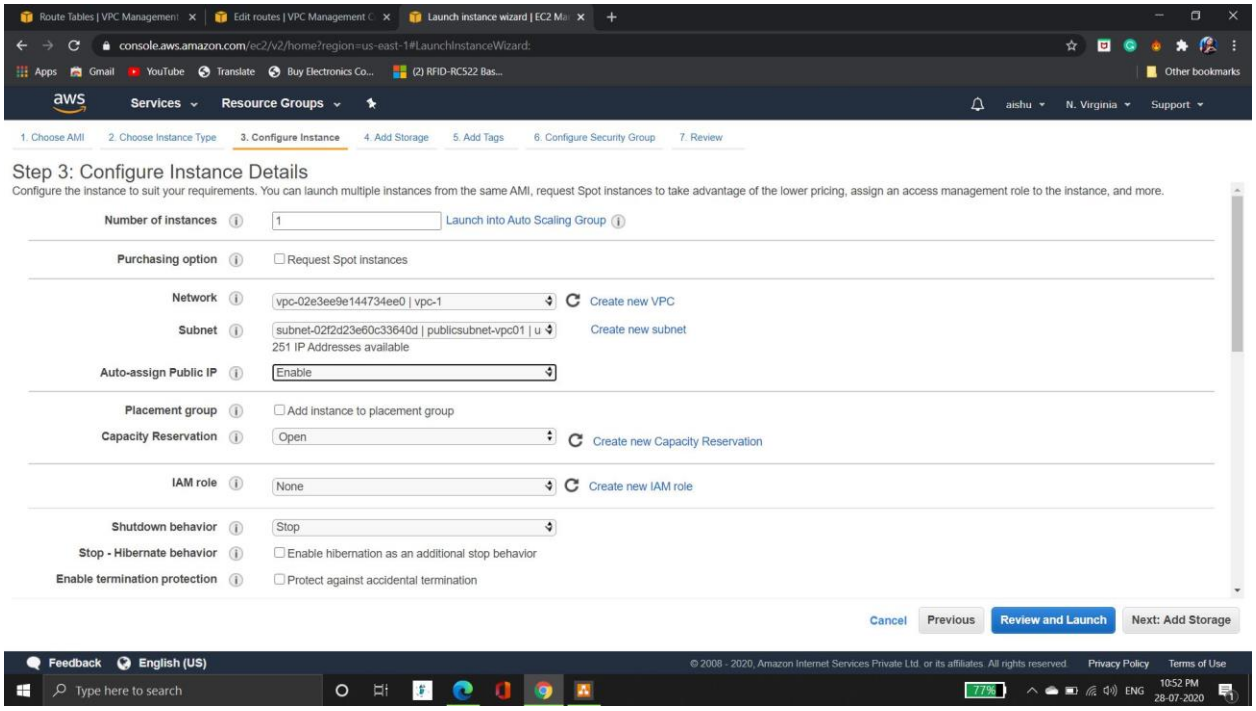
Step 11: Now create one more Route Table and add route to allow traffic from VPC_02



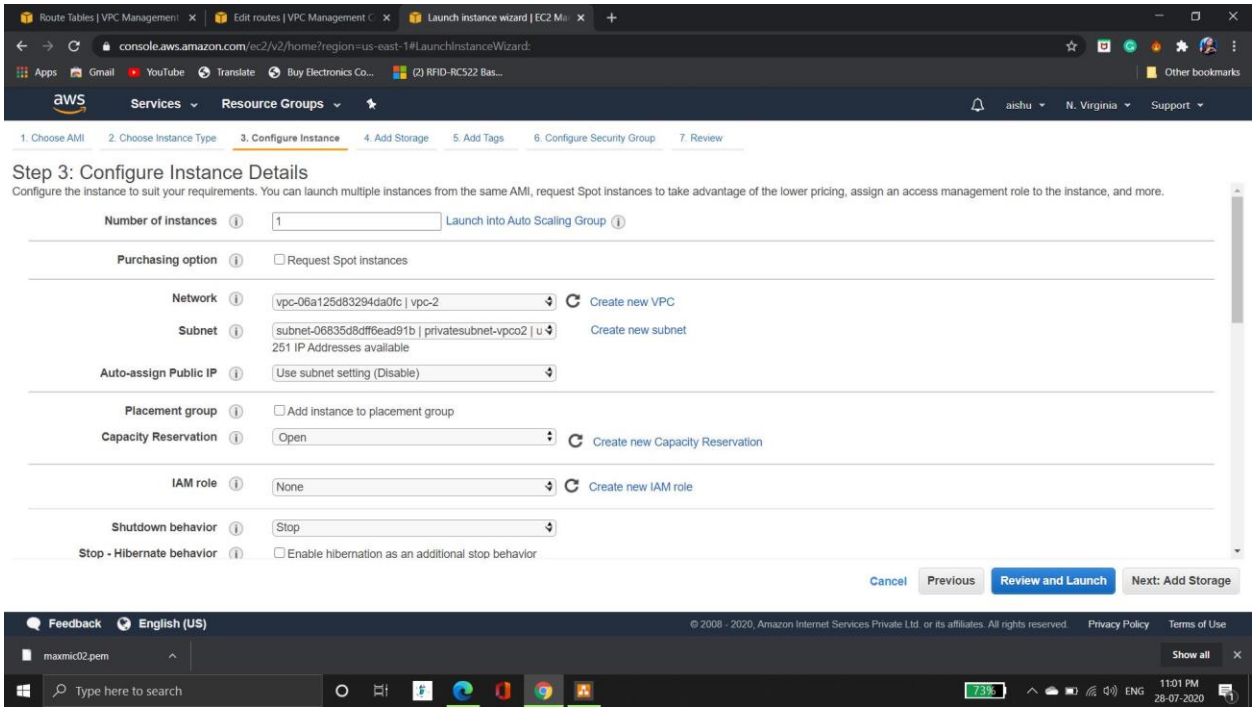
Step 12: Similarly, Create one more Route Table and add route of VPC_01 to flow



Step 13: Now launch EC2 Instance in VPC_01 public subnet and ssh

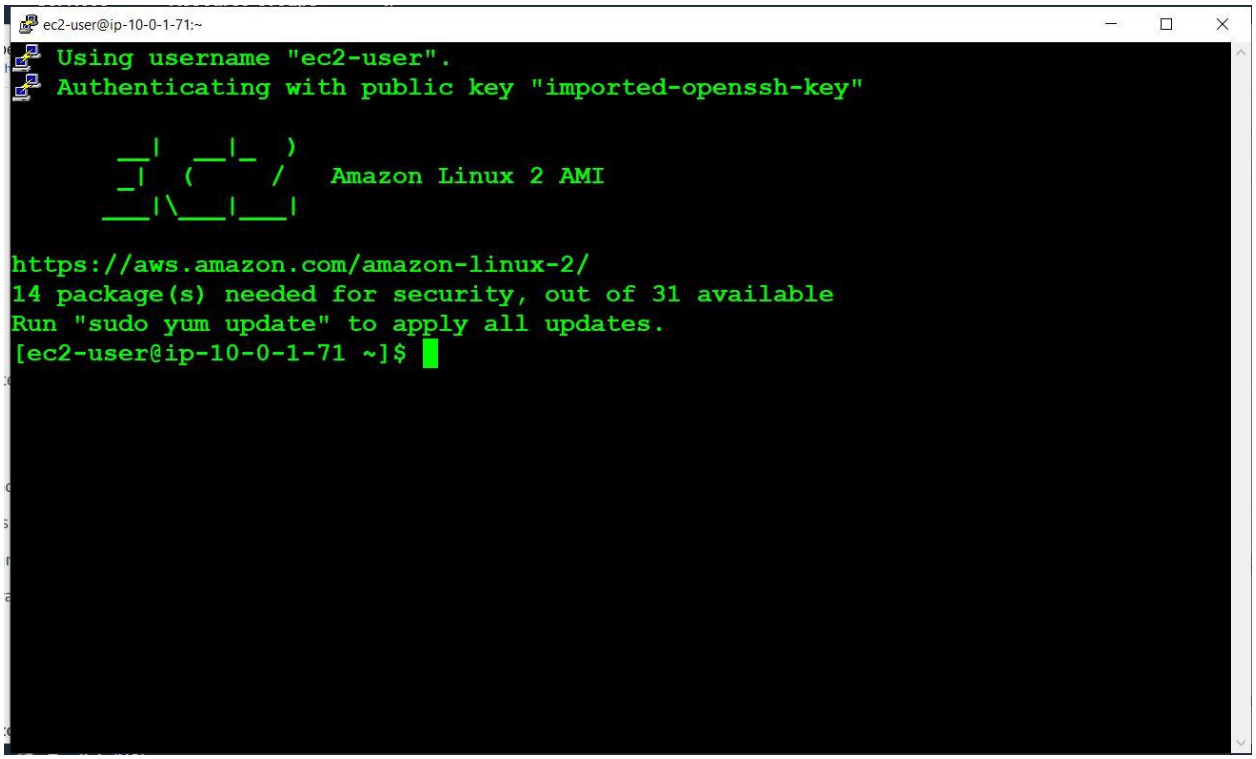


Step 14: Similary, Launch one more instance in private subnet of VPC_02

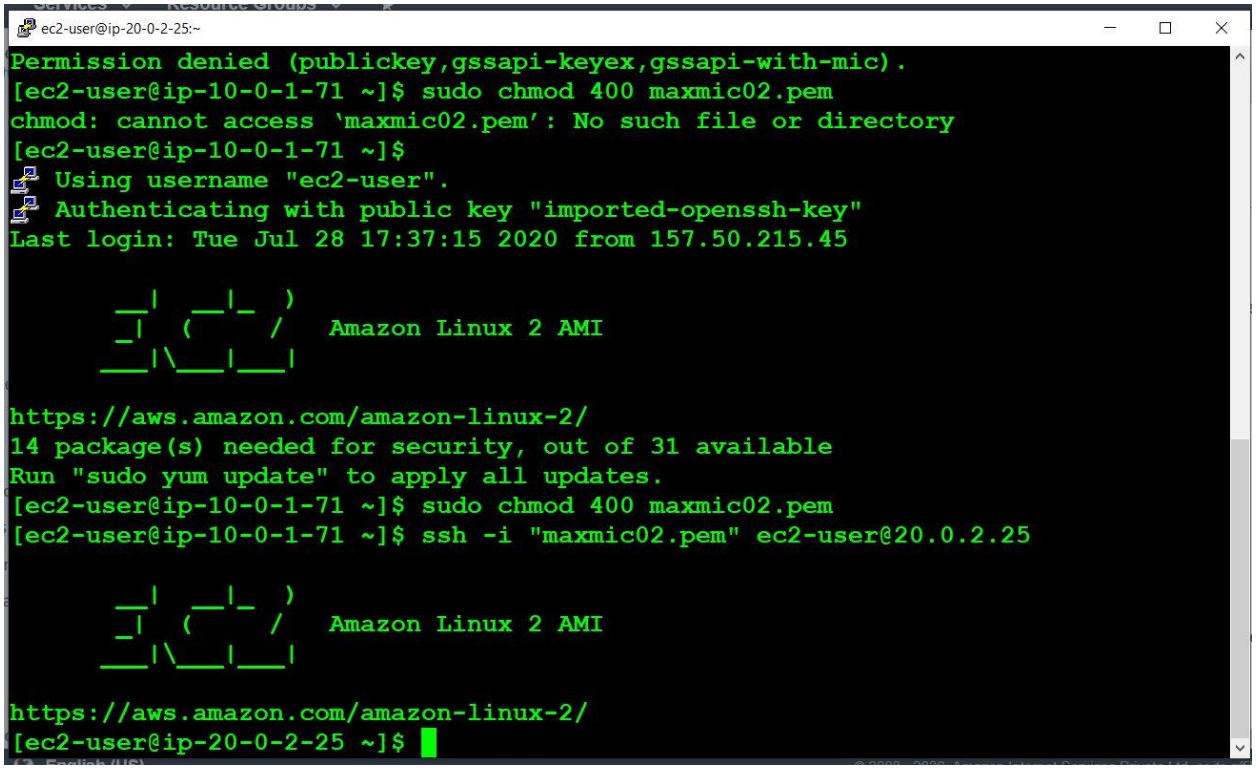


Step 15 : Now, connect to your public Instance i.e. SSH to public Instance

Step 16 : Once Logged in, create key and SSH to private Instance



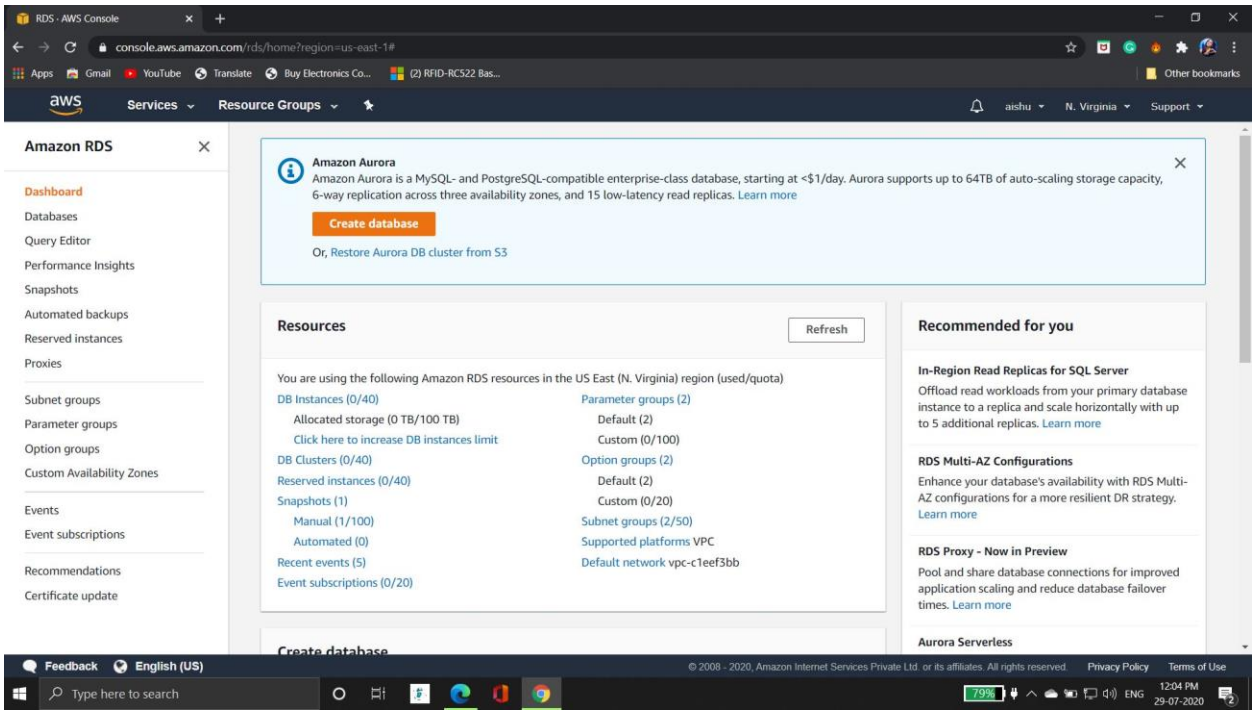
And here you've successfully logged into your private instance sitting in another VPC.


ec2-user@ip-20-0-2-25:~
Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
[ec2-user@ip-10-0-1-71 ~]$ sudo chmod 400 maxmic02.pem
chmod: cannot access 'maxmic02.pem': No such file or directory
[ec2-user@ip-10-0-1-71 ~]$
Using username "ec2-user".
Authenticating with public key "imported-openssh-key"
Last login: Tue Jul 28 17:37:15 2020 from 157.50.215.45
Amazon Linux 2 AMI
https://aws.amazon.com/amazon-linux-2/
14 package(s) needed for security, out of 31 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-10-0-1-71 ~]$ sudo chmod 400 maxmic02.pem
[ec2-user@ip-10-0-1-71 ~]$ ssh -i "maxmic02.pem" ec2-user@20.0.2.25
Amazon Linux 2 AMI
https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-20-0-2-25 ~]$

Task: RDS Lab

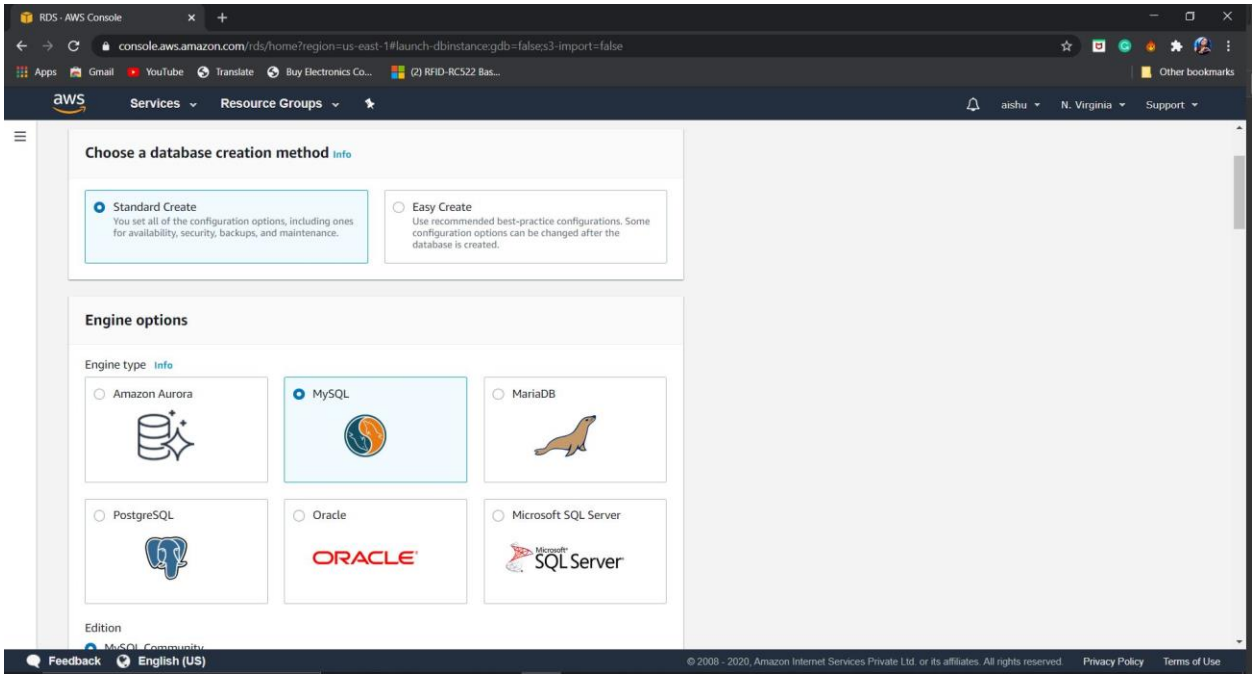
Launching an RDS Instance and connecting it with EC2 Instance

Step 1: Log in to RDS Console

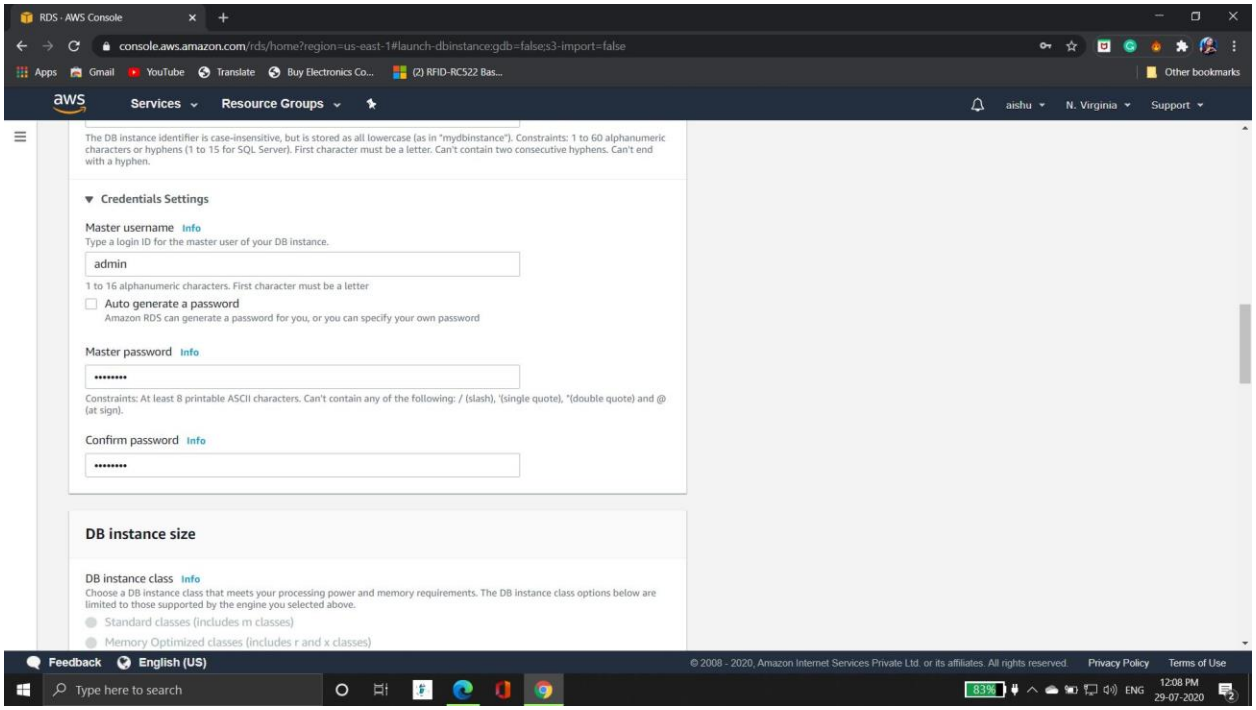


Step 2: Click on "Create Database" and select one of the database type. It is recommended to choose My SQL, Maria DB if you're in Free Tier.

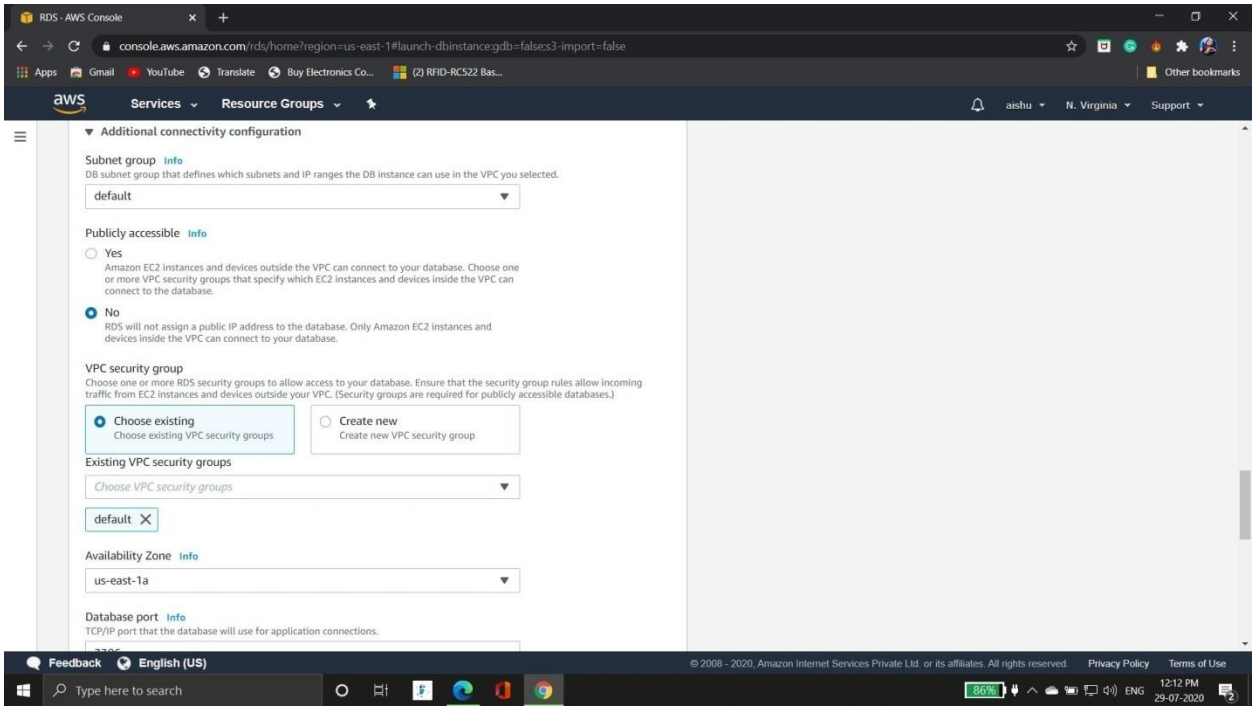
In this lab session we will choose, My SQL.



Step 3: Click on next and select “Dev/Test Environment” and click on next

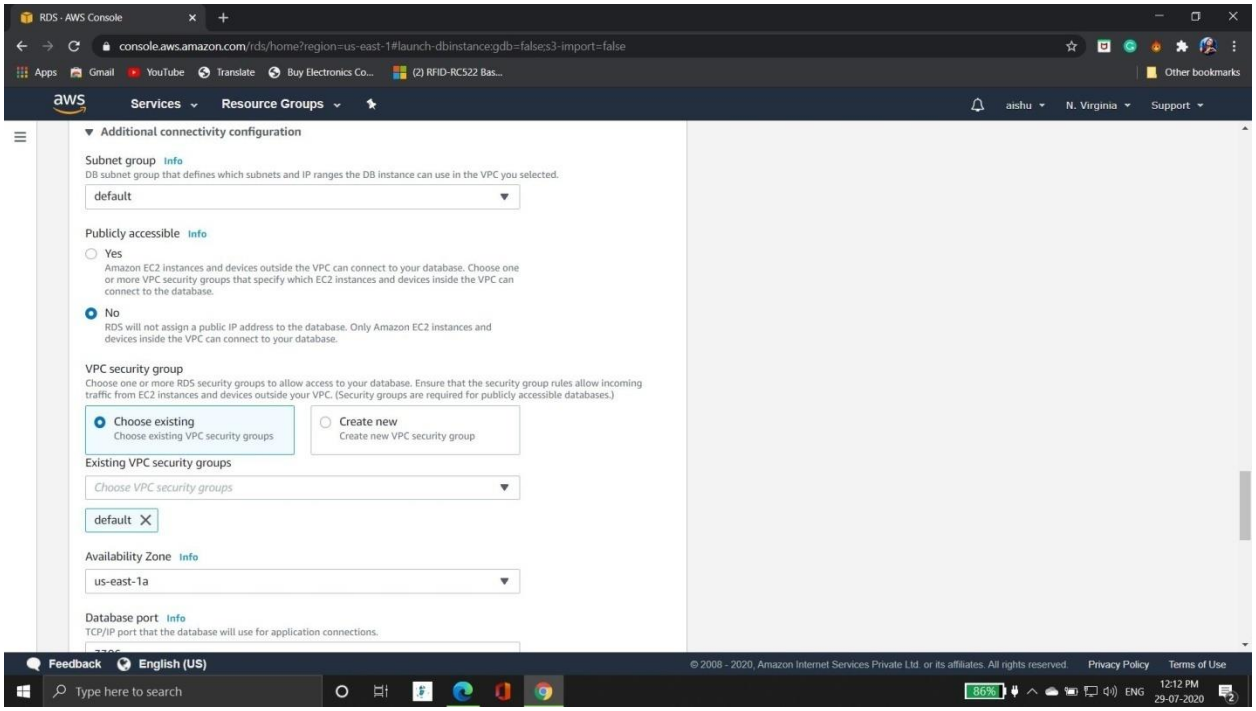


Step 4: Click on the Free Tier eligible for free tier configuration or you can choose any instance type

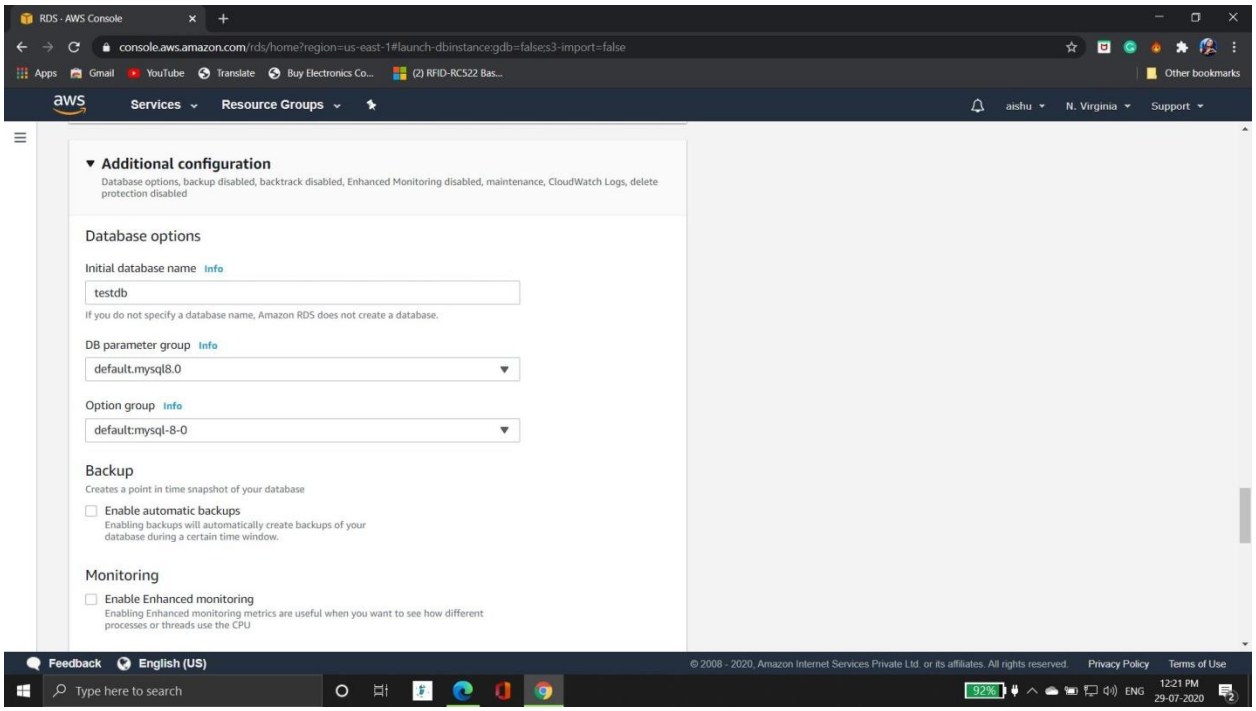


Step 5: Allocate storage for the database, min – 20GiB

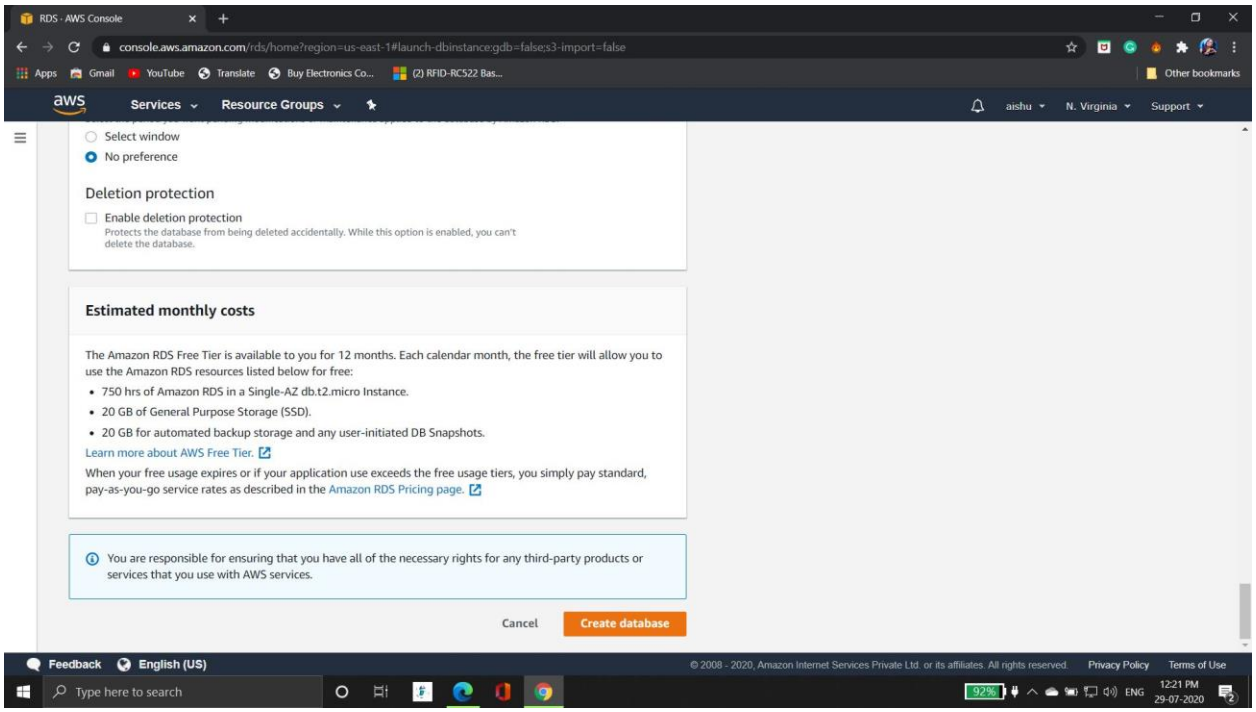
Step 6: Now, tag the DB Instance and give master username and password



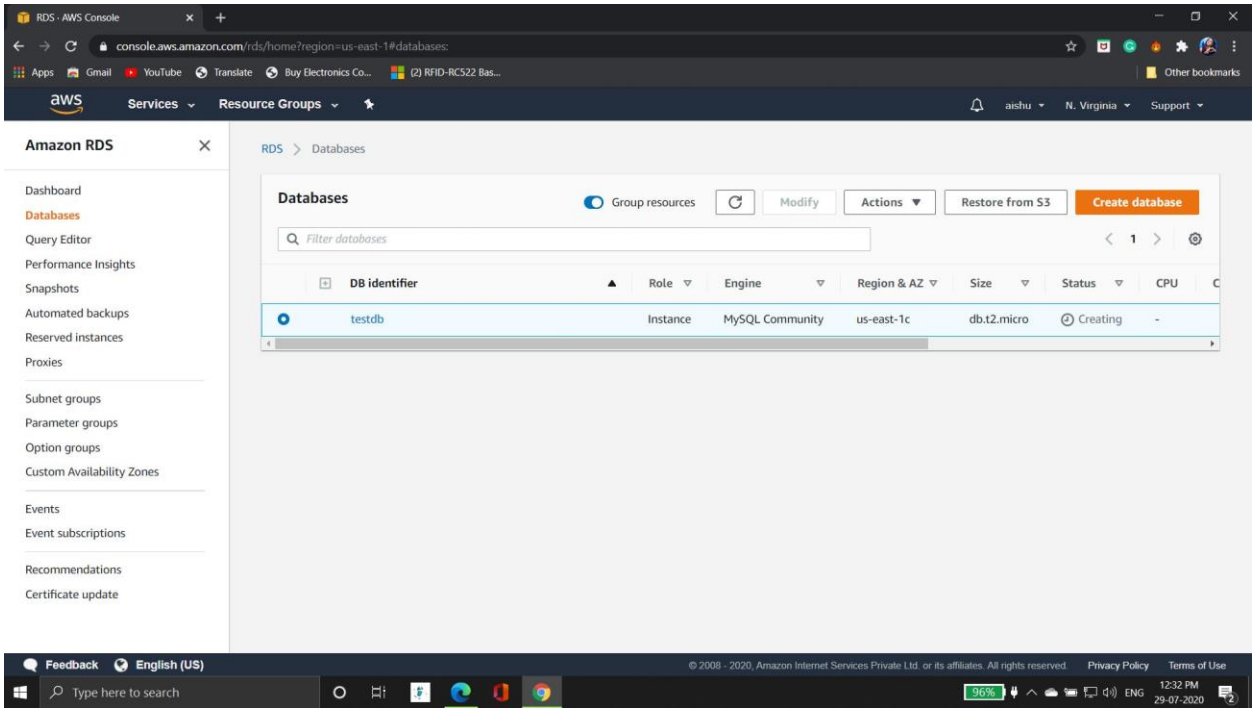
Step 7: Now choose the custom VPC in your Env if any otherwise go with default one and then choose a new private subnet for DB. Also select to create new Security Group



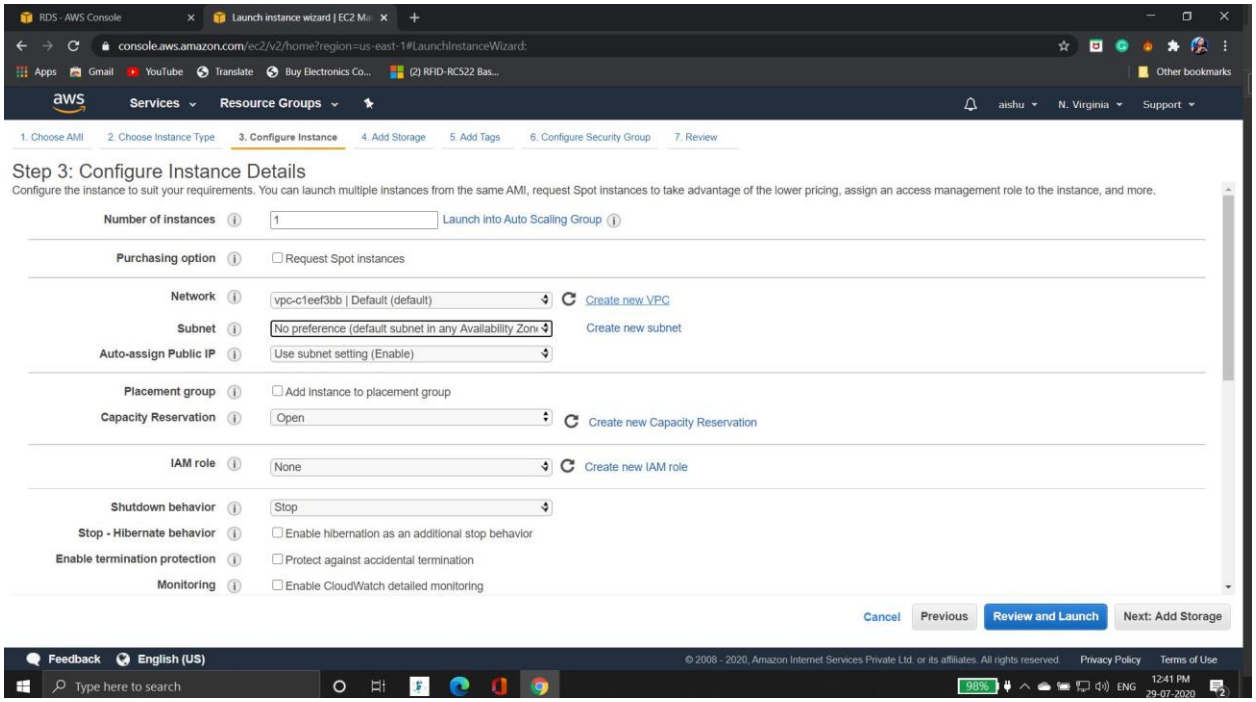
Step 8: Configure the Database And keep everything default and hit create database



It will take some time to launch the Database Instance. Wait till the status becomes Available

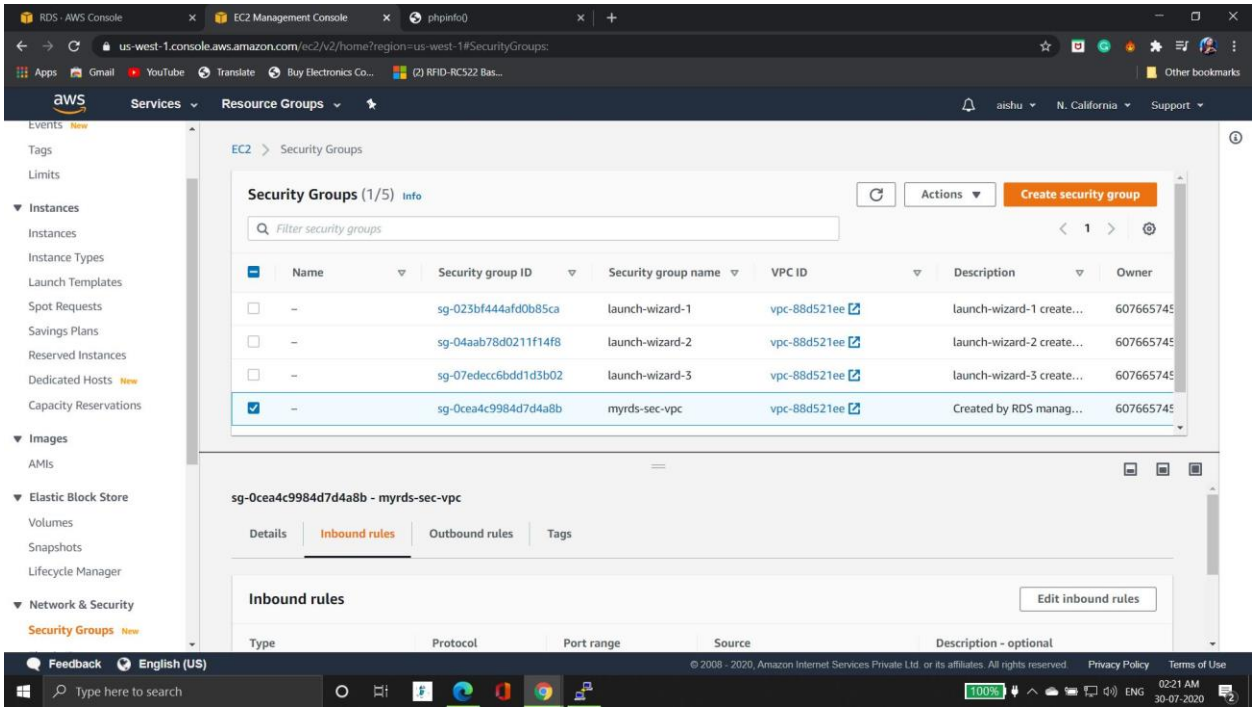


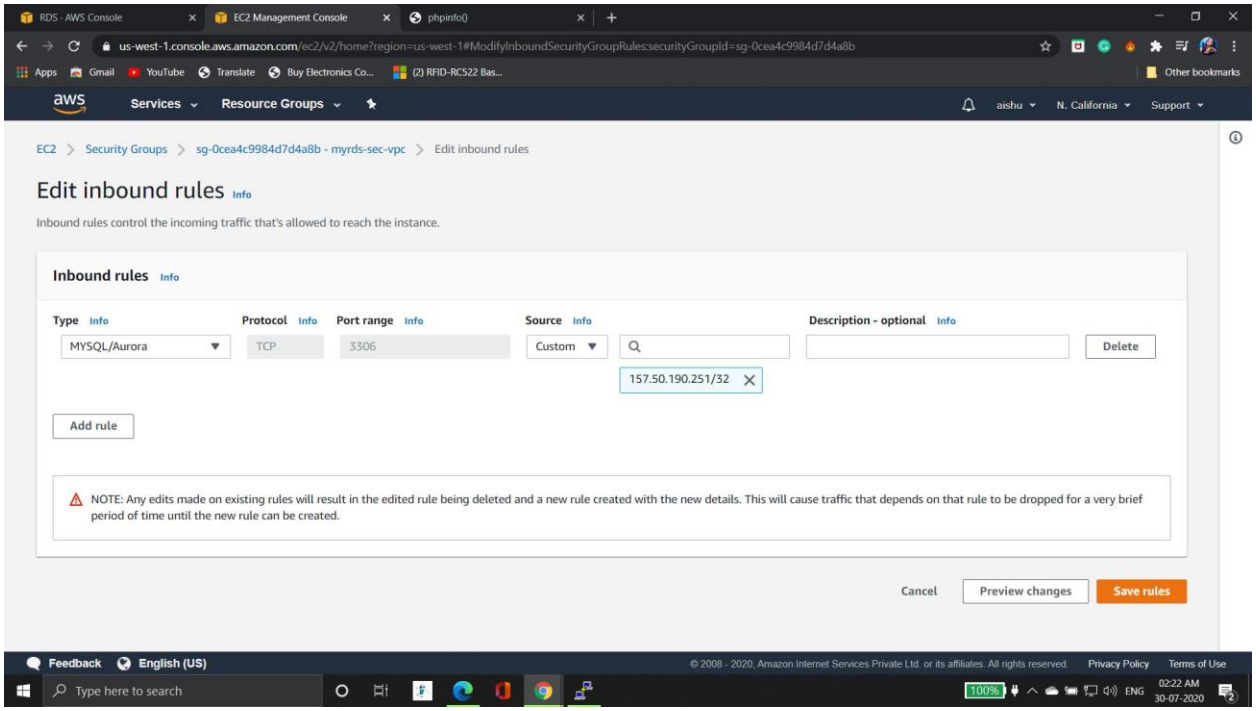
Step 9: Now launch a EC2 Instance in the same VPC

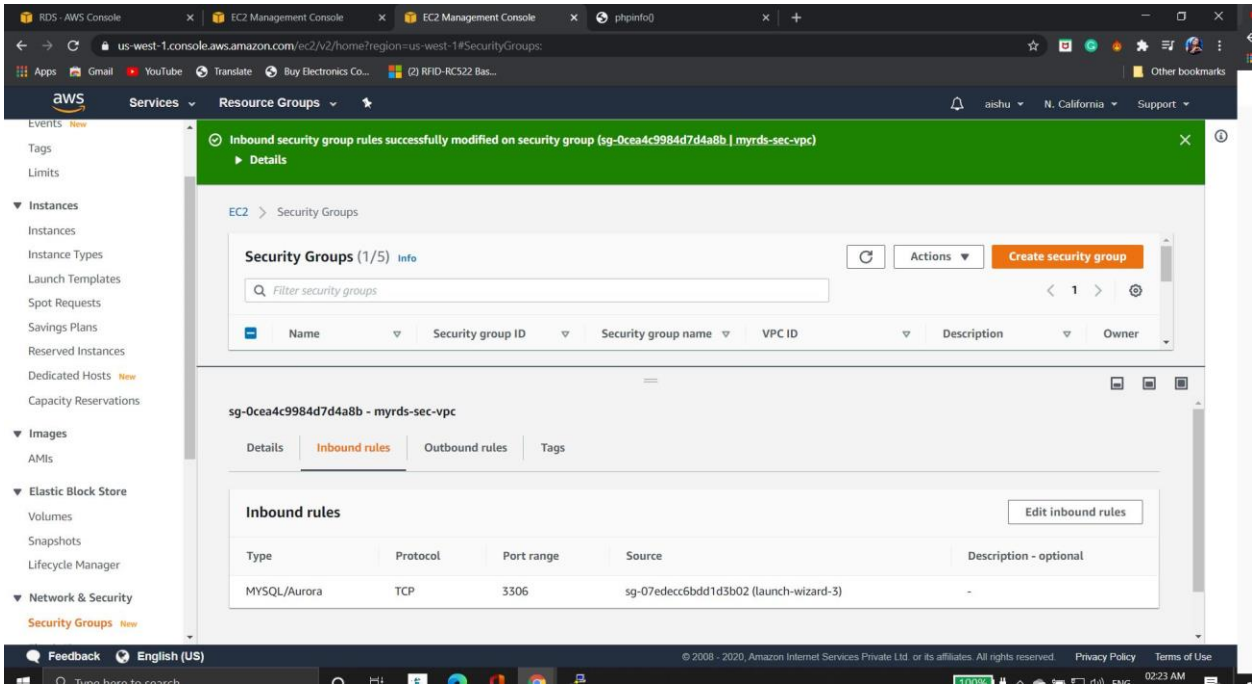


Step 10: Now you must be having two security group : 1 for EC2 Instance and Another of RDS Instance that both are created automatically.

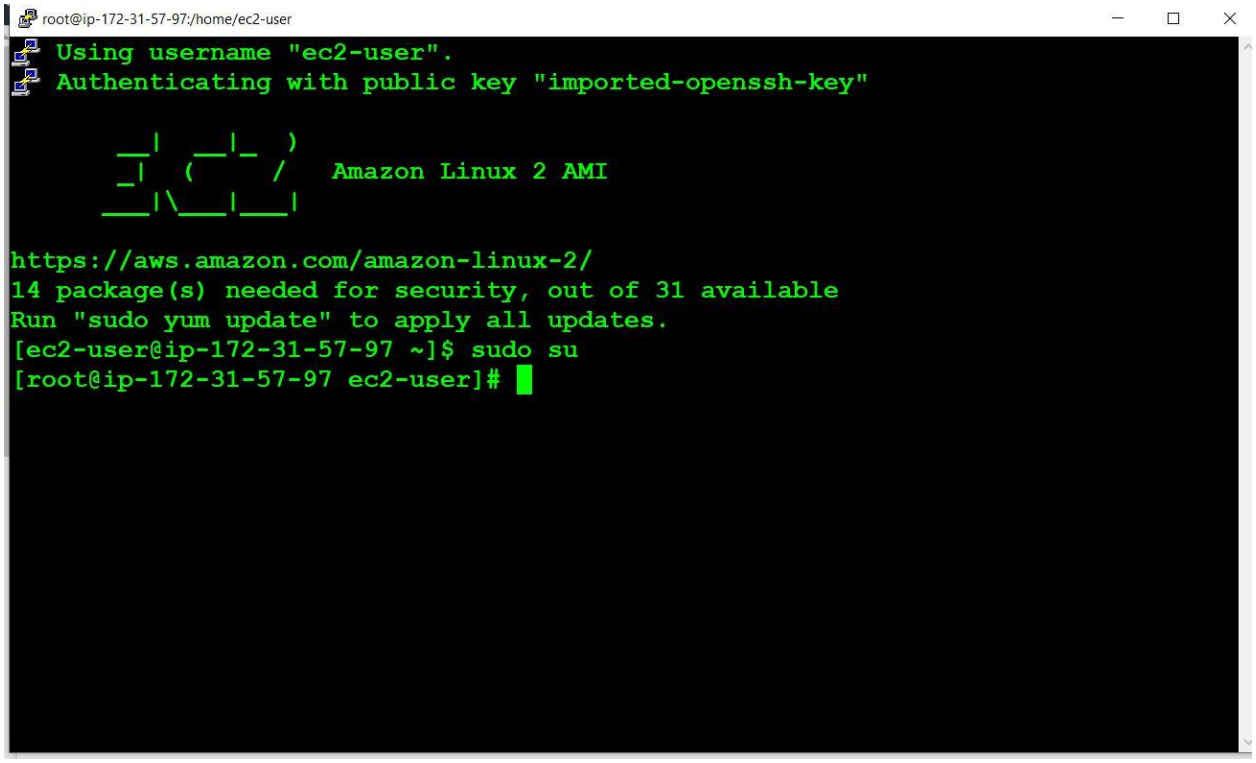
Here in RDS instance add the security group of EC2 intance


RDS - AWS Console
EC2 Management Console
phpinfo()
us-west-1.console.aws.amazon.com/ec2/v2/home?region=us-west-1#SecurityGroups:
Services
Resource Groups
aishu
N. California
Support
EC2 > Security Groups
Security Groups (1/5) Info
Actions
Create security group
Filter security groups
Name
Security group ID
Security group name
VPC ID
Description
Owner
sg-023bf444afd0b85ca launch-wizard-1 vpc-88d521ee launch-wizard-1 create...
sg-04aab78d0211f14f8 launch-wizard-2 vpc-88d521ee launch-wizard-2 create...
sg-07edecc6bdd1d3b02 launch-wizard-3 vpc-88d521ee launch-wizard-3 create...
sg-0cea4c9984d7d4a8b myrds-sec-vpc vpc-88d521ee Created by RDS manag...
sg-0cea4c9984d7d4a8b - myrds-sec-vpc
Details
Inbound rules
Outbound rules
Tags
Edit inbound rules
Type
Protocol
Port range
Source
Description - optional
Feedback
English (US)
Privacy Policy
Terms of Use


RDS - AWS Console
EC2 Management Console
phpinfo()
us-west-1.console.aws.amazon.com/ec2/v2/home?region=us-west-1#ModifyInboundSecurityGroupRules:securityGroupId=sg-0cea4c9984d7d4a8b
Services
Resource Groups
aishu
N. California
Support
EC2 > Security Groups > sg-0cea4c9984d7d4a8b - myrds-sec-vpc > Edit inbound rules
Edit inbound rules Info
Inbound rules control the incoming traffic that's allowed to reach the instance.
Inbound rules Info
Type Info
Protocol Info
Port range Info
Source Info
Description - optional Info
MYSQL/Aurora
TCP
3306
Custom
157.50.190.251/32
Delete
Add rule
NOTE: Any edits made on existing rules will result in the edited rule being deleted and a new rule created with the new details. This will cause traffic that depends on that rule to be dropped for a very brief period of time until the new rule can be created.
Cancel
Preview changes
Save rules
Feedback
English (US)
Privacy Policy
Terms of Use

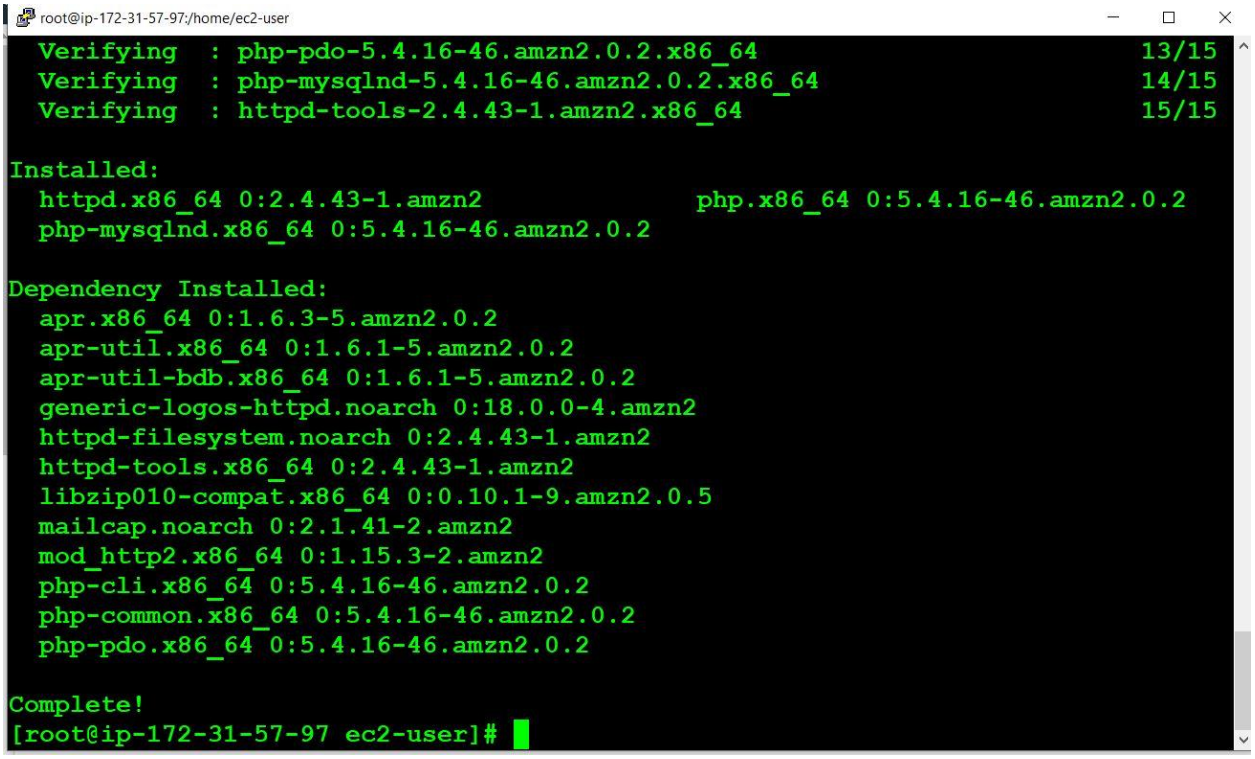


Step 11: Now SSH to your EC2 Instance.



Step 12: Now Install PHP and Apache server to test our connection.

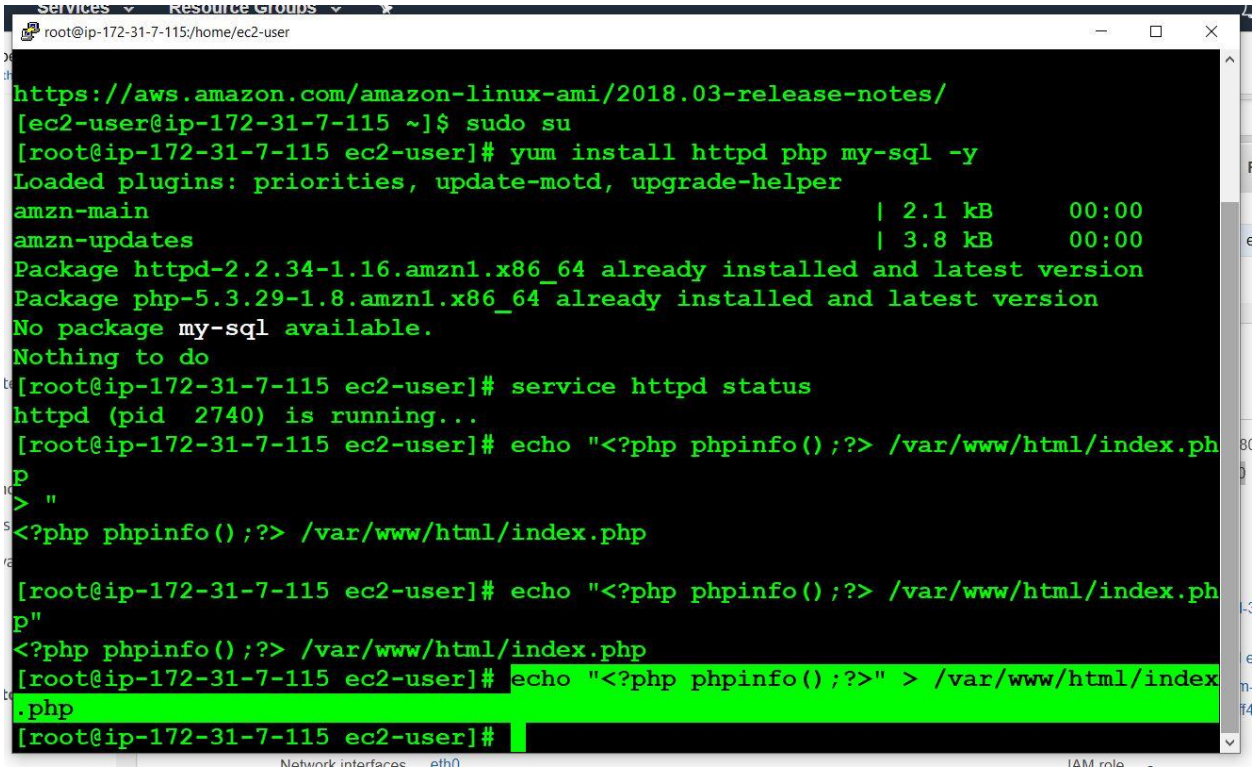
Command: "***yum install httpd php php-mysql -y***"



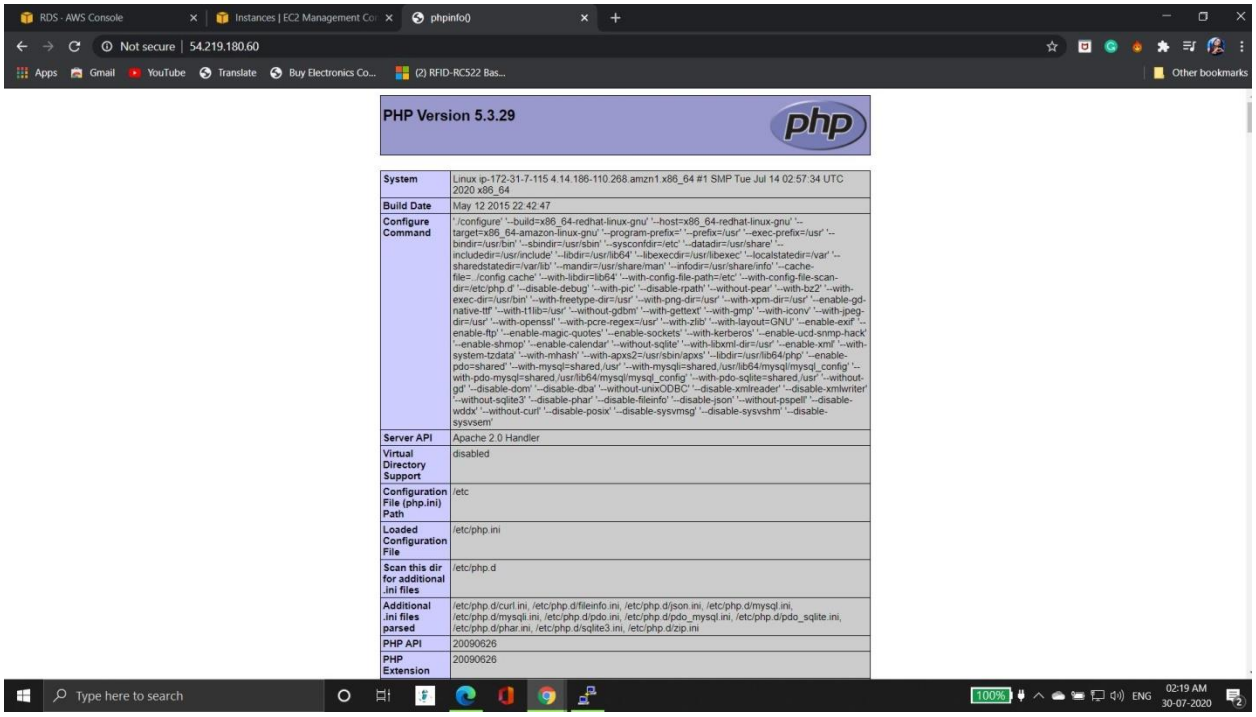
Step 13: Once installed, start the server and create test PHP page

Command: "***service httpd start"***

"echo "<?php phpinfo();?>" > /var/www/html/index.php"



Step 14: Now copy the Instance Public IP and check whether Apache and PHP has successfully installed or not?



If you see this output, then it means – Apache and PHP has been successfully installed

Step 15: Now we will create one PHP File that connects to our database.

Code:

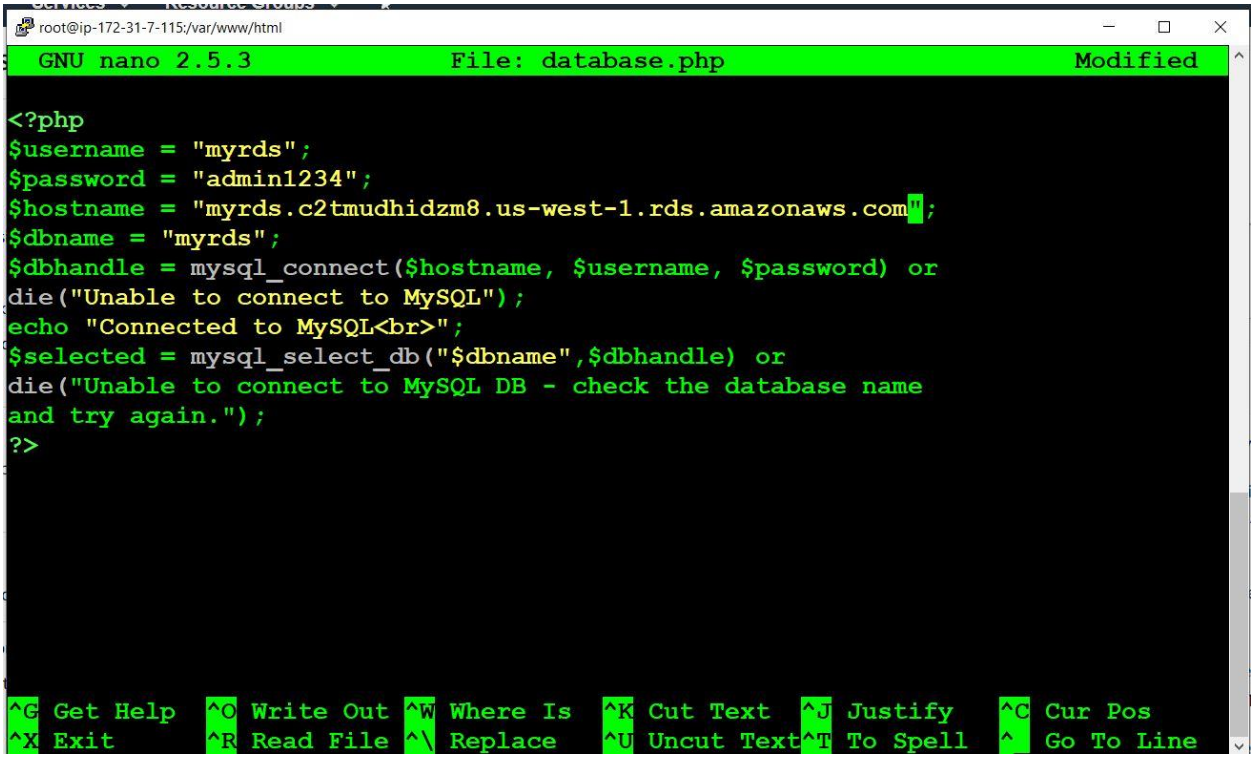
```
<?php
$username = "awsreconnect";
$password = "awsreconnect";
$hostname = "awsreconnect.cfjmmzvhexdp.ap-south-1.rds.amazonaws.com";
$dbname = "awsreconnect";
$dbhandle = mysql_connect($hostname, $username, $password) or die("Unable to connect to MySQL");
echo "Connected to MySQL<br>";
```

$selected = mysql_select_db("$dbname",$dbhandle) or die("Unable to connect to MySQL DB - check the database name and try again.");

?>

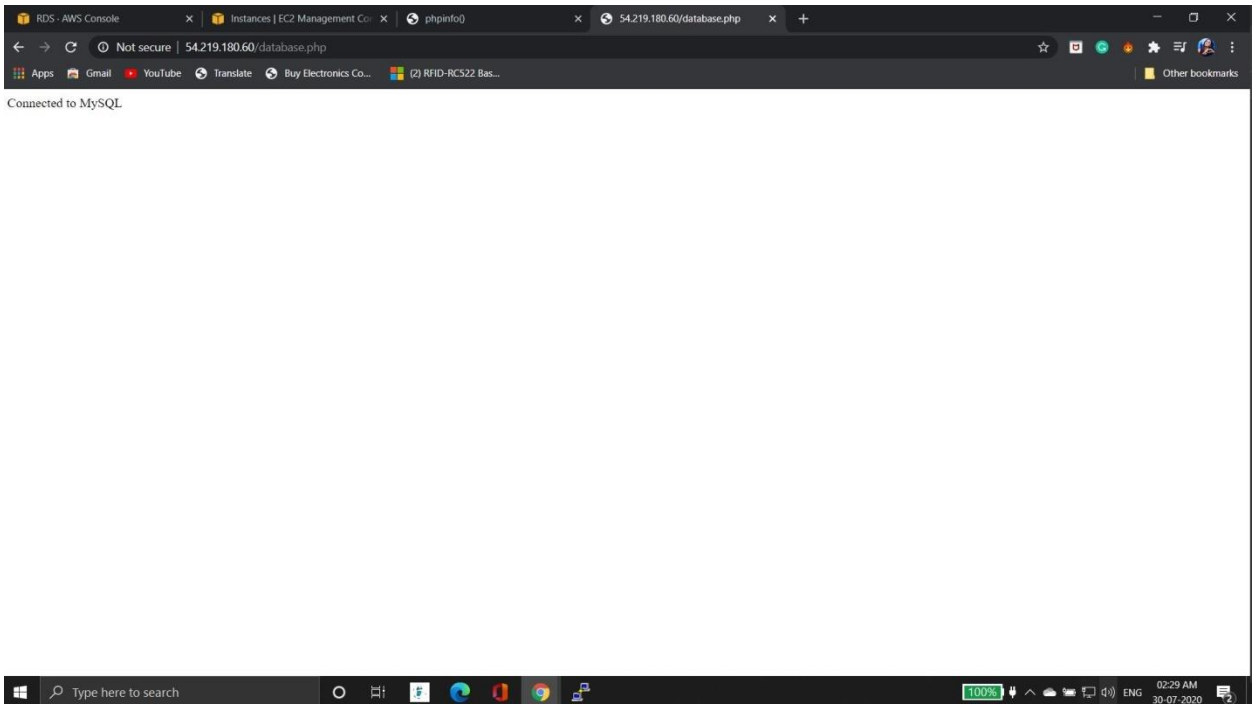
Replace username, password, hostname and db name with actual paramters

Here you'll find the RDS Endpoint copy and paste in the code.



Now save the file with control+o and hit enter

Step 16: Now go to browser and after your Instance public Ip write "ip/database.php"


RDS - AWS Console
Instances | EC2 Management Co
phpinfo()
54.219.180.60/database.php
Not secure | 54.219.180.60/database.php
Apps
Gmail
YouTube
Translate
Buy Electronics Co...
(2) RFID-RC522 Bas...
Other bookmarks
Connected to MySQL
Type here to search
100%
ENG
02:29 AM
30-07-2020

Cloud Formation Lab

Create VPC with Flow logs, Subnets, Route Table and Bastion Host

In this lab session you will learn how to design infrastructure with IAAC (Infrastructure as a Code). This template can be used to create infrastructure in any AWS Region. So this is multi-regionals acceptable Cloud Formation Template with complete customization

"Description": "CloudFormation Template to create VPC with Flow Log, Internet Gateway,Pubic Subnets, Management Subnet and Bastion Host",
"Resources": {
"VPC": {
"Type": "AWS::EC2::VPC",
"Properties": {
"CidrBlock": {
"Ref": "VPCCIDR"
},
"EnableDnsSupport": "true",
"EnableDnsHostnames": "true",
"InstanceTenancy": "default",
"Tags": [
{
"Key": "Name",
"Value": {
"Ref": "VPCName"
}
}

```
]
}
},
"Role": {
"Type": "AWS::IAM::Role",
"Properties": {
"AssumeRolePolicyDocument": {
"Version": "2012-10-17",
"Statement": [
{
"Effect": "Allow",
"Principal": {
"Service": [
"vpc-flow-logs.amazonaws.com"
```

```
]
},
"Action": "sts:AssumeRole"
}
]
},
"Policies": [
{
"PolicyName": "flowlogs-policy",
"PolicyDocument": {
"Version": "2012-10-17",
"Statement": [
{
"Effect": "Allow",
"Action": [
"logs:CreateLogGroup",
"logs:CreateLogStream",
"logs:PutLogEvents",
"logs:DescribeLogGroups",
"logs:DescribeLogStreams"
],
"Resource": "*"
}
]
}
}
```

```
],
"RoleName": "VPCFlowLogsRole"
}
},
"VPCLogGroup": {
"Type": "AWS::Logs::LogGroup",
"Properties": {
"RetentionInDays": {
"Ref": "RetentionInDays"
}
}
},
"FlowLog": {
"Type": "AWS::EC2::FlowLog",
"Properties": {
"DeliverLogsPermissionArn": {
"Fn::GetAtt": [
"Role",
"Arn"
]
},
```

```
"LogGroupName": {
"Ref": "VPCLogGroup"
},
"ResourceId": {
"Ref": "VPC"
},
"ResourceType": "VPC",
"TrafficType": {
"Ref": "TrafficType"
}
}
},
"InternetGateway": {
"Type": "AWS::EC2::InternetGateway",
"Properties": {
"Tags": [
{
"Key": "Name",
"Value": {
"Fn::Join": [
"-",
[
{
"Ref": "VPCName"
},
"InternetGateway"
```

```
]
]
}
}
]
}
},
"AttachGateway": {
"Type": "AWS::EC2::VPCGatewayAttachment",
"Properties": {
"VpcId": {
"Ref": "VPC"
},
"InternetGatewayId": {
"Ref": "InternetGateway"
}
}
},
"pubsubnet": {
"Type": "AWS::EC2::Subnet",
"Properties": {
```

```
"VpcId": {
"Ref": "VPC"
},
"CidrBlock": {
"Ref": "publicsubnetcidr"
},
"AvailabilityZone": {
"Ref": "publicAZ"
},
"Tags": [
{
"Key": "Name",
"Value": "Public-Subnet"
}
]
}
},
"CustRouteTable": {
"Type": "AWS::EC2::RouteTable",
"Properties": {
"VpcId": {
"Ref": "VPC"
},
"Tags": [
{
"Key": "Name",
```

```
"Value": "PrivateRouteTable"
}
]
}
},
"DefaultRT": {
"Type": "AWS::EC2::RouteTable",
"Properties": {
"VpcId": {
"Ref": "VPC"
},
"Tags": [
{
"Key": "Name",
"Value": "PublicRouteTable"
}
]
}
},
"Routeupdate": {
"Type": "AWS::EC2::Route",
```

```
"Properties": {
"RouteTableId": {
"Ref": "DefaultRT"
},
"DestinationCidrBlock": "0.0.0.0/0",
"GatewayId": {
"Ref": "InternetGateway"
}
}
},
"publicsubnetassociation": {
"Type": "AWS::EC2::SubnetRouteTableAssociation",
"Properties": {
"SubnetId": {
"Ref": "pubsubnet"
},
"RouteTableId": {
"Ref": "DefaultRT"
}
},
"DependsOn": "pubsubnet"
},
"bastionSubnet": {
"Type": "AWS::EC2::Subnet",
"Properties": {
"VpcId": {
```

```
"Ref": "VPC"
},
"CidrBlock": {
"Ref": "bastionPublicSubnetCIDR"
},
"AvailabilityZone": {
"Ref": "bastionAZ"
},
"Tags": [
{
"Key": "Name",
"Value": {
"Fn::Join": [
"-",
[
"Management",
{
"Ref": "bastionSubnetName"
}
]
]
```

```
}
}
]
}
},
"BastionSubnetAssociation": {
"Type": "AWS::EC2::SubnetRouteTableAssociation",
"Properties": {
"SubnetId": {
"Ref": "bastionSubnet"
},
"RouteTableId": {
"Ref": "DefaultRT"
}
}
},
"bastionSG": {
"Type": "AWS::EC2::SecurityGroup",
"Properties": {
"GroupName": "BastionSecGroup",
"GroupDescription": "Open SSH",
"VpcId": {
"Ref": "VPC"
},
"SecurityGroupIngress": [
{
```

"IpProtocol": "tcp",
"FromPort": "22",
"ToPort": "22",
"CidrIp": {
"Ref": "bastionSGDestionationAddress"
}
}
],
"Tags": [
{
"Key": "Name",
"Value": "BastionSG"
}
]
}
},
"bastionServer": {
"Type": "AWS::EC2::Instance",
"Properties": {
"InstanceType": {
"Ref": "InstanceType"

```
},
"NetworkInterfaces": [
{
"AssociatePublicIpAddress": "true",
"DeviceIndex": "0",
"GroupSet": [
{
"Ref": "bastionSG"
}
],
"SubnetId": {
"Ref": "bastionSubnet"
}
}
],
"DisableApiTermination": {
"Ref": "BastionServerTermination"
},
"ImageId": {
"Fn::FindInMap": [
"RegionMap",
{
"Ref": "AWS::Region"
},
"AMI"
]
```

```
},
"BlockDeviceMappings": [
{
"DeviceName": "/dev/xvda",
"Ebs": {
"VolumeSize": {
"Ref": "BastionVolumeSize"
},
"DeleteOnTermination": {
"Ref": "bastionVolumeTermination"
},
"VolumeType": "gp2"
}
}
],
"KeyName": {
"Ref": "BastionKeyPair"
},
"Tags": [
{
"Key": "Name",
```

```
"Value": {
"Fn::Join": [
"-",
[
"Bastion",
{
"Ref": "bastionTAG"
}
]
]
}
}
]
}
},
"bastionEIP": {
"Type": "AWS::EC2::EIP",
"Properties": {
"Domain": "VPC",
"InstanceId": {
"Ref": "bastionServer"
}
}
}
},
"Parameters": {
```

"VPCCIDR": {
"Type": "String",
"Default": "10.10.0.0/16",
"AllowedPattern": "(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})/(\\d{1,2})",
"Description": "Enter VPC Range"
},
"VPCName": {
"Description": "Enter Name for VPC",
"Type": "String"
},
"publicAZ": {
"Type": "AWS::EC2::AvailabilityZone::Name"
},
"publicsubnetcidr": {
"Type": "String",
"Default": "10.10.1.0/24",
"AllowedPattern": "(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})/(\\d{1,2})",
"Description": "Enter range for Public Subent"
},
"bastionPublicSubnetCIDR": {
"Type": "String",

"Default": "10.10.2.0/24",
"AllowedPattern": "(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})/(\\d{1,2})",
"Description": "Enter range for Public Subent"
},
"bastionAZ": {
"Type": "AWS::EC2::AvailabilityZone::Name"
},
"bastionSubnetName": {
"Type": "String",
"Description": "Enter Subnet Name (Note: Name is preceded by \"Management\")"
},
"bastionTAG": {
"Type": "String",
"Description": "Enter the Environment/Tag Name for EC2 servers (Note: Name is preceded by \"Bastion\")"
},
"bastionVolumeTermination": {
"Type": "String",
"Default": "false",
"AllowedValues": [
"false",
"true"
],
"Description": "Select \"true\" if volume is to be retained post universe server deletion"
},

"BastionServerTermination": {
"Type": "String",
"Default": "false",
"AllowedValues": [
"false",
"true"
],
"Description": "Selecting \"true\" will require to disable the termination protection manually from the console"
},
"InstanceType": {
"Type": "String",
"Default": "t2.micro",
"AllowedValues": [
"t1.micro",
"t2.nano",
"t2.micro",
"t2.small",
"t2.medium",
"t2.large",
"m1.small",
"m1.medium",

"m1.large",
"m1.xlarge",
"m2.xlarge",
"m2.2xlarge",
"m2.4xlarge",
"m3.medium",
"m3.large",
"m3.xlarge",
"m3.2xlarge",
"m4.large",
"m4.xlarge",
"m4.2xlarge",
"m4.4xlarge",
"m4.10xlarge",
"c1.medium",
"c1.xlarge",
"c3.large",
"c3.xlarge",
"c3.2xlarge",
"c3.4xlarge",
"c3.8xlarge",
"c4.large",
"c4.xlarge",
"c4.2xlarge",
"c4.4xlarge",
"c4.8xlarge",

"g2.2xlarge",
"g2.8xlarge",
"r3.large",
"r3.xlarge",
"r3.2xlarge",
"r3.4xlarge",
"r3.8xlarge",
"i2.xlarge",
"i2.2xlarge",
"i2.4xlarge",
"i2.8xlarge",
"d2.xlarge",
"d2.2xlarge",
"d2.4xlarge",
"d2.8xlarge",
"hi1.4xlarge",
"hs1.8xlarge",
"cr1.8xlarge",
"cc2.8xlarge",
"cg1.4xlarge"
],

"Description": "Select the instance type "
},
"BastionVolumeSize": {
"Type": "Number",
"Description": "EBS VolumeSize (GB)",
"Default": "10"
},
"BastionKeyPair": {
"Description": "Select the exsisting Key",
"Type": "AWS::EC2::KeyPair::KeyName"
},
"bastionSGDestionationAddress": {
"Description": "Enter the IP to be whitelisted to SSH Bastion Host",
"Type": "String",
"Default": "0.0.0.0/0"
},
"RetentionInDays": {
"Description": "Specify number of days you want to retain VPC_Flow_log.",
"Type": "Number",
"Default": 14,
"AllowedValues": [
1,
3,
5,
7,
14,

30,
60,
90,
120,
150,
180,
365,
400,
545,
731,
1827,
3653
]
},
"TrafficType": {
"Description": "The type of traffic to stream in cloudtrail bucket",
"Type": "String",
"Default": "REJECT",
"AllowedValues": [
"ACCEPT",
"REJECT",

```
"ALL"
]
}
},
"Mappings": {
"RegionMap": {
"us-east-1": {
"AMI": "ami-a4827dc9"
},
"us-west-1": {
"AMI": "ami-11790371"
},
"us-west-2": {
"AMI": "ami-f303fb93"
},
"ap-south-1": {
"AMI": "ami-e2b6dc8d"
},
"ap-northeast-2": {
"AMI": "ami-69e92207"
},
"ap-southeast-1": {
"AMI": "ami-a2c111c1"
},
"ap-southeast-2": {
"AMI": "ami-d9d7f9ba"
```

```
},
"ap-northeast-1": {
"AMI": "ami-6154bb00"
},
"eu-central-1": {
"AMI": "ami-7df01e12"
},
"eu-west-1": {
"AMI": "ami-c39604b0"
},
"sa-east-1": {
"AMI": "ami-106ee57c"
}
}
},
"Metadata": {
"AWS::CloudFormation::Interface": {
"ParameterGroups": [
{
"Label": {
"default": "VPC"
```

```
},
"Parameters": [
"VPCCIDR",
"VPCName",
"RetentionInDays",
"TrafficType"
]
},
{
"Label": {
"default": "Public Subnets",
"Description": "Public subnet for Internet Gateway"
},
"Parameters": [
"publicAZ",
"publicsubnetcidr"
]
},
{
"Label": {
"default": "Management Subnets",
"Description": "Management subnet for Bastion Host"
},
"Parameters": [
"bastionAZ",
"bastionPublicSubnetCIDR",
```

"bastionSubnetName"
]
},
{
"Label": {
"default": "Management Server",
"Description": "Management Server for Bastion Host"
},
"Parameters": [
"bastionTAG",
"bastionVolumeTermination",
"BastionServerTermination",
"InstanceType",
"BastionVolumeSize",
"BastionKeyPair",
"bastionSGDestionationAddress"
]
}
],
"ParameterLabels": {

"VPCCIDR": {
"default": "VPC Range "
},
"VPCName": {
"default": "VPC Name "
},
"RetentionInDays": {
"default": "FlowLogs Retaintion Period "
},
"TrafficType": {
"default": "Traffic type Role "
},
"publicsubnetcidr": {
"default": "Public Subnet "
},
"publicAZ": {
"default": "Select Availability Zone "
},
"bastionSubnetName": {
"default": "Management Subnet Name "
},
"bastionAZ": {
"default": "Select Availability Zone "
},
"bastionPublicSubnetCIDR": {
"default": "Subnet Range "

```
},
"bastionTAG": {
"default": "Environment/Tag "
},
"bastionVolumeTermination": {
"default": "volume Termination Protection "
},
"BastionServerTermination": {
"default": "EC2 Termination Protection "
},
"InstanceType": {
"default": "Instance Type "
},
"BastionVolumeSize": {
"default": "Volume Size "
},
"BastionKeyPair": {
"default": "Key Pair "
},
"bastionSGDestionationAddress": {
"default": "ASource IP for Bastion SG"
```

```
}
}
}
},
"Outputs": {
"VPC": {
"Description": "VPC ID",
"Value": {
"Ref": "VPC"
},
"Export": {
"Name": "Network-VPCID"
}
},
"VPCCIDR": {
"Description": "VPC CIDR",
"Value": {
"Fn::GetAtt": [
"VPC",
"CidrBlock"
]
},
"Export": {
"Name": "Network-VPCRange"
}
},
```

```
"PrivateRouteTableID": {
"Description": "Private Route Table ID",
"Value": {
"Ref": "CustRouteTable"
},
"Export": {
"Name": "Network-PrivateRouteTableID"
}
},
"PublicRouteTableID": {
"Description": "Public Route Table ID",
"Value": {
"Ref": "DefaultRT"
},
"Export": {
"Name": "Network-PublicRouteTableID"
}
},
"BastionEIP": {
"Description": "Bastion IP",
"Value": {
"Fn::GetAtt": [
"bastionServer",
"PrivateIp"
]
},
```

```
"32"
]
]
},
"Export": {
"Name": "Network-BastionEIP"
}
},
"MyStacksRegion": {
"Description": "VPC Region",
"Value": {
"Ref": "AWS::Region"
},
"Export": {
"Name": "Network-VPCRegion"
}
}
}
}
```

Bibliography And Reference Taken.

- ✓ **AWS official website**
- ✓ **Project workbook of AWS by cosmic skill**

https://www.azureskynet.com/

https://www.cosmicskills.com/course-status/