



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

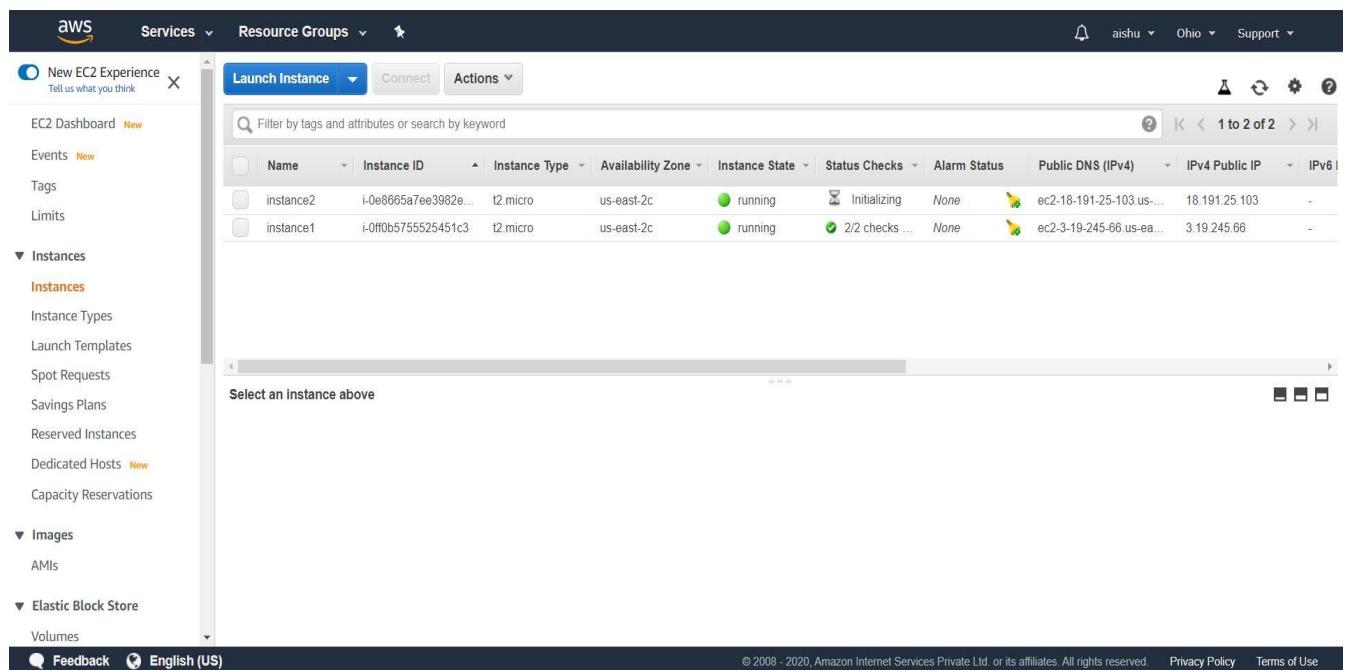
s.no	Project Name	Page.no
1.	Using the EBS Volumes to attach multiple EC2 Instance	4-16
2.	Accessing instance between different VPCs.	17-28
3.	RDS Lab	29-42
4.	Cloud Formation Lab	43-73

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



The screenshot shows the AWS EC2 Dashboard. On the left, there's a navigation sidebar with links for EC2 Dashboard, Events, Tags, Limits, Instances (with sub-links for Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, and Capacity Reservations), Images (AMIs), and Elastic Block Store (Volumes). The main content area displays a table of running instances. The table has columns for Name, Instance ID, Instance Type, Availability Zone, Instance State, Status Checks, Alarm Status, Public DNS (IPv4), IPv4 Public IP, and IPv6 Public IP. There are two rows in the table:

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public IP	IPv6 Public IP
instance2	i-0e8665a7ee3982e...	t2.micro	us-east-2c	running	Initializing	None	ec2-18-191-25-103.us...	18.191.25.103	-
instance1	i-0ff0b5755525451c3	t2.micro	us-east-2c	running	2/2 checks ...	None	ec2-3-19-245-66.us-ea...	3.19.245.66	-

At the bottom of the dashboard, there are links for Feedback, English (US), and footer links for Privacy Policy and Terms of Use.

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

AWS Services Resource Groups

Volumes > Create Volume

Create Volume

Volume Type: General Purpose SSD (gp2)

Size (GiB): 10 (Min: 1 GiB, Max: 16384 GiB)

IOPS: 100 / 3000 (Baseline of 3 IOPS per GiB with a minimum of 100 IOPS, burstable to 3000 IOPS)

Availability Zone*: us-east-2c

Throughput (MB/s): Not applicable

Snapshot ID: Select a snapshot

Encryption: Encrypt this volume

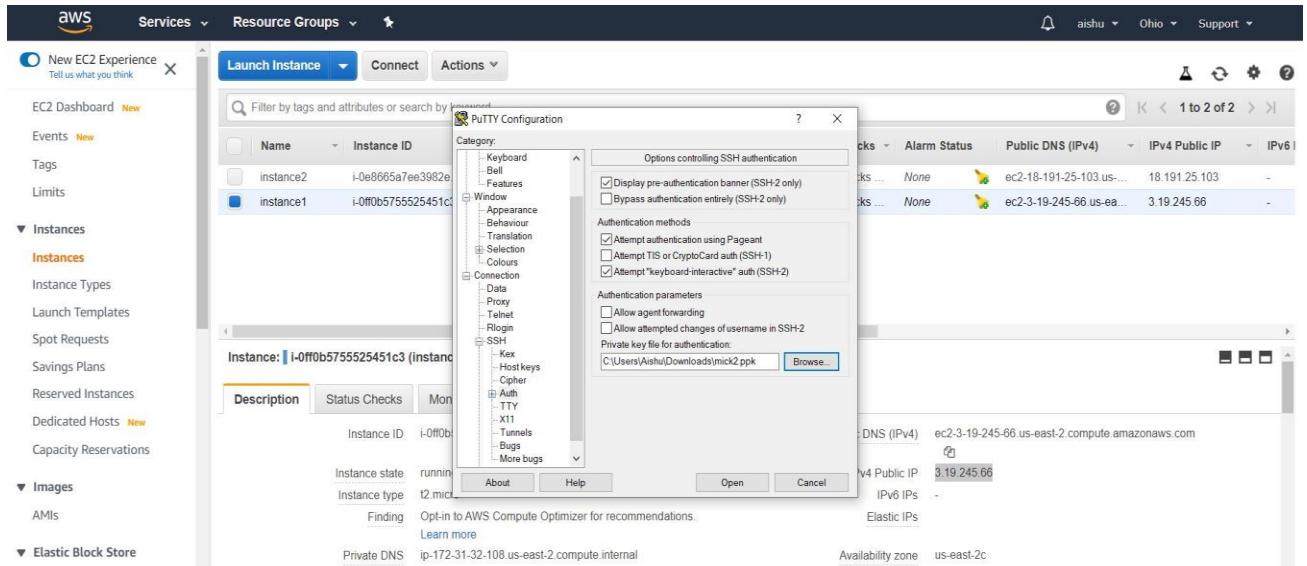
Key (128 characters maximum)

Value (256 characters maximum)

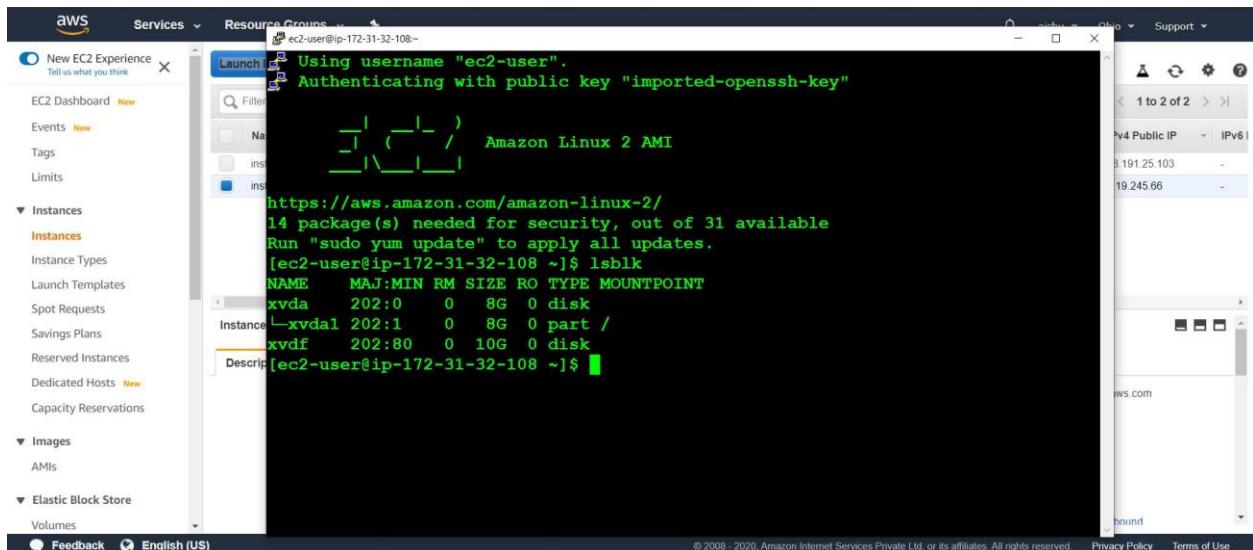
Feedback English (US) © 2008 - 2020, Amazon Internet Services Private Ltd or its affiliates. All rights reserved. Privacy Policy Terms of Use

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

Step 4: Through putty connecting the instances



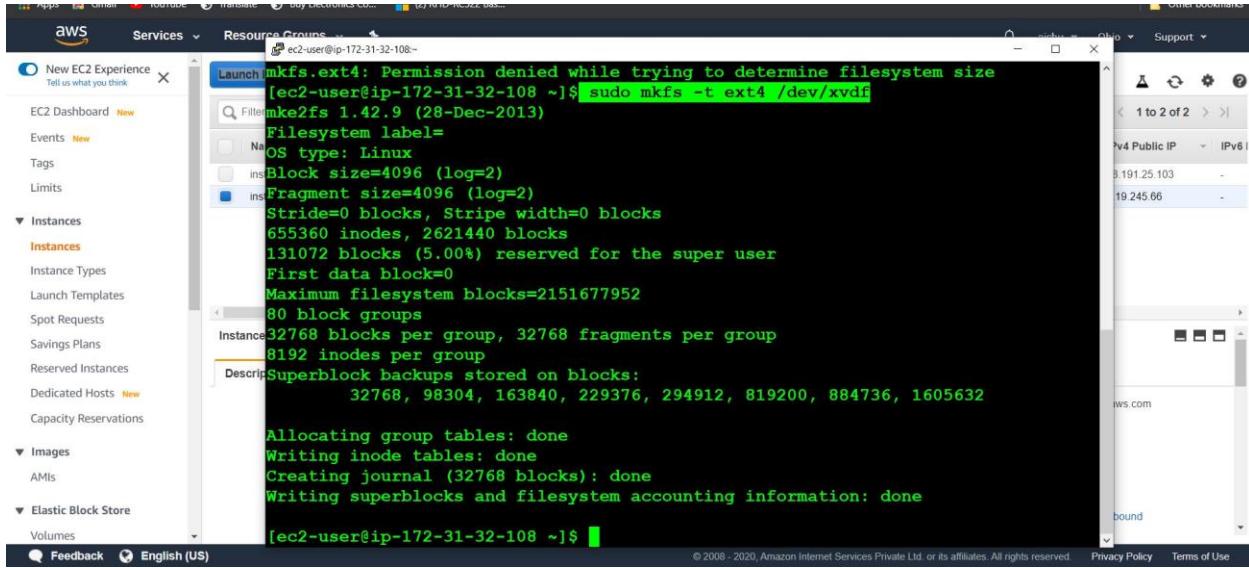
Step 5: Now SSH to your first Instance and type “`lsblk`” to list all the block device.



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***”



```
ec2-user@ip-172-31-32-108 ~]$ sudo mkfs -t ext4 /dev/xvdf
mke2fs 1.42.9 (28-Dec-2013)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
655360 inodes, 2621440 blocks
131072 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=2151677952
80 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
      32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done

[ec2-user@ip-172-31-32-108 ~]$
```

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***”

```
ec2-user@ip-172-31-32-108:~  
Stride=0 blocks, Stripe width=0 blocks  
655360 inodes, 2621440 blocks  
131072 blocks (5.00%) reserved for the super user  
First data block=0  
Maximum filesystem blocks=2151677952  
80 block groups  
32768 blocks per group, 32768 fragments per group  
8192 inodes per group  
Superblock backups stored on blocks:  
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632  
  
Allocating group tables: done  
Writing inode tables: done  
Creating journal (32768 blocks): done  
Writing superblocks and filesystem accounting information: done  
  
[ec2-user@ip-172-31-32-108 ~]$ mkdir /text  
mkdir: cannot create directory '/text': File exists  
[ec2-user@ip-172-31-32-108 ~]$ sudo mkdir /text  
mkdir: cannot create directory '/text': File exists  
[ec2-user@ip-172-31-32-108 ~]$ sudo mkdir /text2  
mkdir: cannot create directory '/text2': File exists  
[ec2-user@ip-172-31-32-108 ~]$ sudo mkdir /text4  
[ec2-user@ip-172-31-32-108 ~]$ █
```

Step 8: Now mount this directory to volume.

Command : “*mount /dev/xvdf/foldername/*”

```
[ec2-user@ip-172-31-32-108:/text4]
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
655360 inodes, 2621440 blocks
131072 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=2151677952
80 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done

[ec2-user@ip-172-31-32-108 ~]$ sudo mkdir /text4
mkdir: cannot create directory '/text4': File exists
[ec2-user@ip-172-31-32-108 ~]$ sudo mount dev/xvdf /text4/
mount: /text4: special device dev/xvdf does not exist.
[ec2-user@ip-172-31-32-108 ~]$ sudo mount /dev/xvdf /text4/
```

Now change the directory and move to newly created directory

Command: *cd /foldername*"

```
[ec2-user@ip-172-31-32-108:text4]
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
655360 inodes, 2621440 blocks
131072 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=2151677952
80 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done

[ec2-user@ip-172-31-32-108 ~]$ sudo mkdir /text4
mkdir: cannot create directory '/text4': File exists
[ec2-user@ip-172-31-32-108 ~]$ sudo mount dev/xvdf /text4/
mount: /text4: special device dev/xvdf does not exist.
[ec2-user@ip-172-31-32-108 ~]$ sudo mount /dev/xvdf /text4/
[ec2-user@ip-172-31-32-108 ~]$ cd /text4/
[ec2-user@ip-172-31-32-108 text4]$ sudo nano text4
```

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”

```
[ec2-user@ip-172-31-32-108:~] 131072 blocks (5.00%) reserved for the super user  
First data block=0  
Maximum filesystem blocks=2151677952  
80 block groups  
32768 blocks per group, 32768 fragments per group  
8192 inodes per group  
Superblock backups stored on blocks:  
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632  
  
Allocating group tables: done  
Writing inode tables: done  
Creating journal (32768 blocks): done  
Writing superblocks and filesystem accounting information: done  
  
[ec2-user@ip-172-31-32-108 ~]$ sudo mkdir /text4  
mkdir: cannot create directory '/text4': File exists  
[ec2-user@ip-172-31-32-108 ~]$ sudo mount dev/xvdf /text4/  
mount: /text4: special device dev/xvdf does not exist.  
[ec2-user@ip-172-31-32-108 ~]$ sudo mount /dev/xvdf /text4/  
[ec2-user@ip-172-31-32-108 ~]$ cd /text4/  
[ec2-user@ip-172-31-32-108 text4]$ sudo nano text4  
[ec2-user@ip-172-31-32-108 text4]$ cd  
[ec2-user@ip-172-31-32-108 ~]$ sudo umount -d /dev/xvdf  
[ec2-user@ip-172-31-32-108 ~]$
```

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

Name	Volume ID	Size	Volume Type	IOPS	Snapshot	Created	Availability Zone	State	Alarm Status	Attachment
vol-0c8cc9b3...	10 GiB	gp2	100		July 27, 2020 at 4:3...	us-east-2c	available	None		
instance2	vol-0a6d3ba...	8 GiB	gp2	100	snap-0dd5b79...	July 27, 2020 at 4:2...	us-east-2c	in-use	None	i-0e8665a7
instance1	vol-0167a76...	8 GiB	gp2	100	snap-0dd5b79...	July 27, 2020 at 4:2...	us-east-2c	in-use	None	i-0ff0b575c

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”

The screenshot shows a terminal window with the following session:

```
root@ip-172-31-34-7:~# Authenticating with public key "imported-openssh-key"
Last login: Mon Jul 27 13:21:38 2020 from 157.50.185.158
[Amazon Linux 2 AMI]
https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-34-7 ~]$ lsblk
NAME   MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
xvda    202:0    0  8G  0 disk
└─xvda1 202:1    0  8G  0 part /
xvdf    202:80   0 10G  0 disk
[ec2-user@ip-172-31-34-7 ~]$ sudo su
[root@ip-172-31-34-7 ec2-user]# lsblk
NAME   MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
xvda    202:0    0  8G  0 disk
└─xvda1 202:1    0  8G  0 part /
xvdf    202:80   0 10G  0 disk
[root@ip-172-31-34-7 ec2-user]# sudo mkdir /test1dir
[root@ip-172-31-34-7 ec2-user]# sudo mount /dev/xvdf/test1dir/
mount: /dev/xvdf/test1dir/: can't find in /etc/fstab.
[root@ip-172-31-34-7 ec2-user]# sudo mount /dev/xvdf /test1dir/
[root@ip-172-31-34-7 ec2-user]#
```

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

Shows successful mount to new volume.

```
root@ip-172-31-34-7:~# lsblk
NAME   MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
xvda    202:0    0  8G  0 disk
└─xvda1 202:1    0  8G  0 part /
xvdf    202:80   0 10G  0 disk
[ec2-user@ip-172-31-34-7 ~]$ sudo su
[root@ip-172-31-34-7 ec2-user]# lsblk
NAME   MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
xvda    202:0    0  8G  0 disk
└─xvda1 202:1    0  8G  0 part /
xvdf    202:80   0 10G  0 disk
[root@ip-172-31-34-7 ec2-user]# sudo mkdir /test1dir
[root@ip-172-31-34-7 ec2-user]# sudo mount /dev/xvdf/test1dir/
mount: /dev/xvdf/test1dir/: can't find in /etc/fstab.
[root@ip-172-31-34-7 ec2-user]# sudo mount /dev/xvdf /test1dir/
[root@ip-172-31-34-7 ec2-user]# lsblk
NAME   MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
xvda    202:0    0  8G  0 disk
└─xvda1 202:1    0  8G  0 part /
xvdf    202:80   0 10G  0 disk /test1dir
[root@ip-172-31-34-7 ec2-user]#
```

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

Command : “*cd /testdir*

ls”

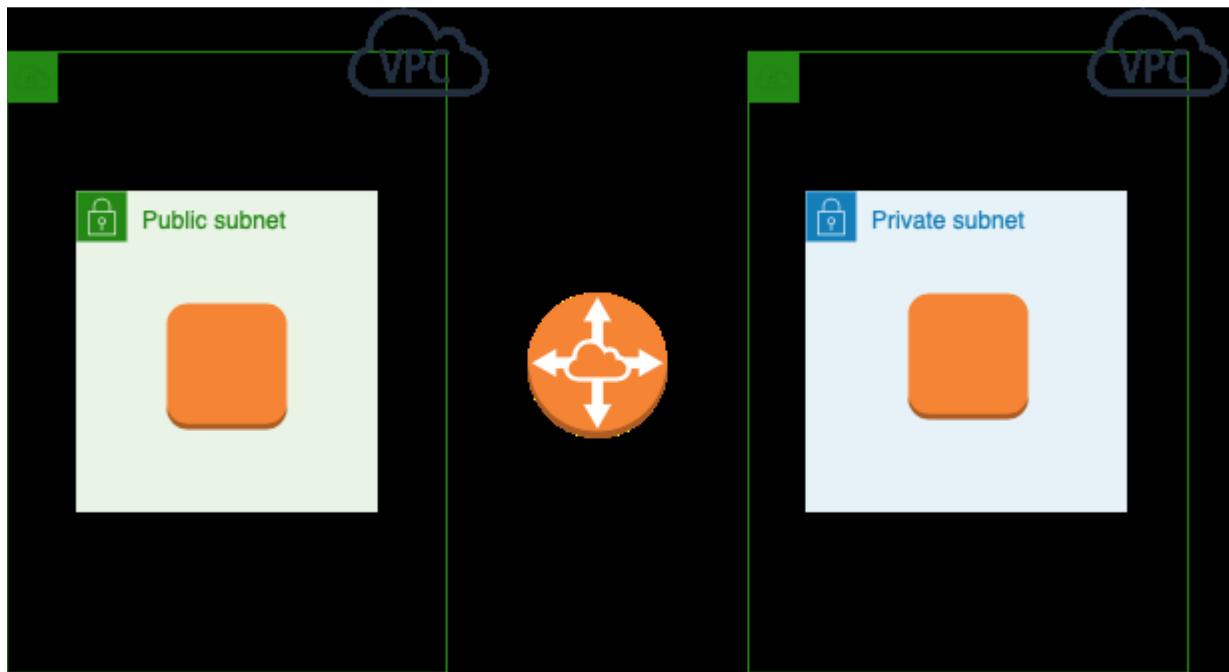
```
root@ip-172-31-34-7:/test1dir
[ec2-user@ip-172-31-34-7 ~]$ lsblk
NAME   MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
xvda    202:0    0   8G  0 disk
└─xvda1 202:1    0   8G  0 part /
xvdf    202:80   0  10G  0 disk
[ec2-user@ip-172-31-34-7 ~]$ sudo su
[root@ip-172-31-34-7 ec2-user]# lsblk
NAME   MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
xvda    202:0    0   8G  0 disk
└─xvda1 202:1    0   8G  0 part /
xvdf    202:80   0  10G  0 disk
[root@ip-172-31-34-7 ec2-user]# sudo mkdir /test1dir
[root@ip-172-31-34-7 ec2-user]# sudo mount /dev/xvdf/test1dir/
mount: /dev/xvdf/test1dir/: can't find in /etc/fstab.
[root@ip-172-31-34-7 ec2-user]# sudo mount /dev/xvdf /test1dir/
[root@ip-172-31-34-7 ec2-user]# lsblk
NAME   MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
xvda    202:0    0   8G  0 disk
└─xvda1 202:1    0   8G  0 part /
xvdf    202:80   0  10G  0 disk /test1dir
[root@ip-172-31-34-7 ec2-user]# cd /test1dir/
[root@ip-172-31-34-7 test1dir]# ls
lost+found  text4
[root@ip-172-31-34-7 test1dir]#
```

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

S.no 2: Accessing instance between different VPCs.

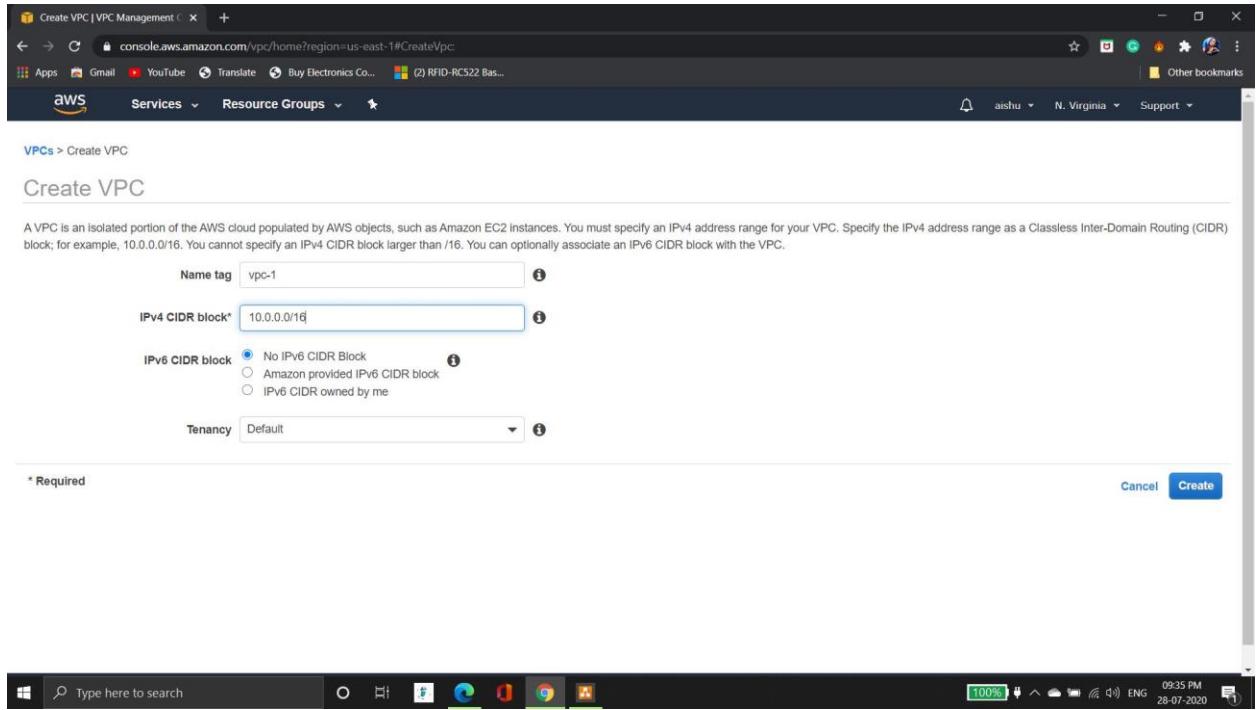
In this Case, we will create two VPC and we will then access instance from public VPC to the instance in private VPC

Architecture:

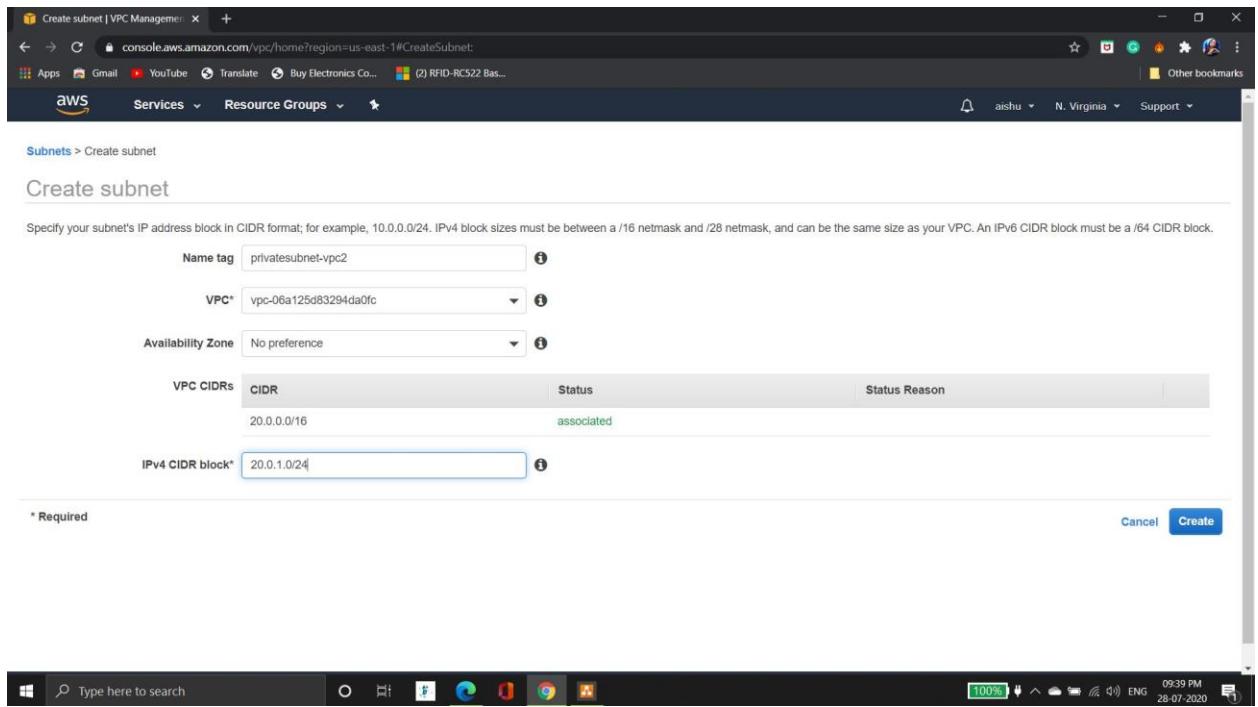


Step 1: Go to VPC Dashboard and select a region

Step 2: Create first VPC



Step 3: Create second VPC



Step 4: Create one public subnet for VPC_01

The screenshot shows the AWS VPC Management Console with the Subnets page open. The left sidebar shows the VPC Dashboard with options like Your VPCs, Subnets, Route Tables, Internet Gateways, Egress Only Internet Gateways, DHCP Options Sets, Elastic IPs, Managed Prefix Lists, Endpoints, Endpoint Services, NAT Gateways, and Peering Connections. The main area has a 'Create subnet' button and a table listing subnets. The table columns are Name, Subnet ID, State, VPC, IPv4 CIDR, Available IPv4, IPv6 CIDR, and Availability Zone. The listed subnets are:

Name	Subnet ID	State	VPC	IPv4 CIDR	Available IPv4	IPv6 CIDR	Availability Zone
publicsubnet-vpc01	subnet-02f2d23e60c33640d	available	vpc-02e3ee9e144734ee0 ...	10.0.1.0/24	251	-	us-east-1d
privatesubnet-vpc02	subnet-06935d8dff0ead91b	available	vpc-06a125d83294da0fc ...	20.0.2.0/24	251	-	us-east-1c
publicsubnet-vpc2	subnet-087020ab15a8eb1d5	available	vpc-06a125d83294da0fc ...	20.0.1.0/24	251	-	us-east-1c
subnet-1df44e13		available	vpc-c1eef3bb Default	172.31.64.0/20	4091	-	us-east-1f
subnet-4ebd7011		available	vpc-c1eef3bb Default	172.31.32.0/20	4091	-	us-east-1d
subnet-6a2f3b54		available	vpc-c1eef3bb Default	172.31.48.0/20	4091	-	us-east-1e
subnet-8cd11ead		available	vpc-c1eef3bb Default	172.31.80.0/20	4091	-	us-east-1b
subnet-933bfaf5		available	vpc-c1eef3bb Default	172.31.0.0/20	4091	-	us-east-1a
subnet-fc9303b1		available	vpc-c1eef3bb Default	172.31.16.0/20	4091	-	us-east-1c

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

The screenshot shows the AWS VPC Management console with the 'Route Tables' section selected. A new route table named 'defaultpublicroutetable vpc1' has been created and is highlighted. The table is associated with VPC ID 'vpc-02e3ee9e144734ee0'. The 'Routes' tab is active, displaying two routes: one for destination '10.0.0.0/16' with target 'local' and status 'active'; another for '0.0.0.0/0' with target 'igw-0ba00a720264984c4' and status 'active'. The propagated status for both is 'No'.

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.

The screenshot shows the AWS VPC Management interface with the 'Route Tables' section selected. On the left, there's a sidebar with various VPC-related options like Internet Gateways, Egress Only Internet Gateways, DHCP Options Sets, and Security. The main area displays a table of route tables with columns for Name, Route Table ID, Explicit subnet association, Edge associations, Main, VPC ID, and Owner. Three entries are listed: 'Default' (rtb-276df559), 'defaultpublicroutetable vpc1' (rtb-0d60a60f70a01b9e4), and 'defaultpublicroutetablevpc02' (rtb-05cd51f76199d...). Below the table, there are tabs for Summary, Routes, Subnet Associations, Edge Associations, Route Propagation, and Tags. Under the 'Routes' tab, there's a table showing routes with columns for Destination, Target, Status, and Propagated. Three routes are listed: '10.0.0.0/16' with target 'local' and status 'active'; '0.0.0.0' with target 'igw-0ba00a720264984c4' and status 'active'; and '20.0.0.0/16' with target 'pcx-05e79f11d70ec5436' and status 'active'. The status for all routes is 'active' and the propagated status is 'No'.

Similarly, to another Route Table of VPC_02

Destination	Target	Status	Propagated
20.0.0.16	local	active	No
0.0.0.0	igw-01a3918b17ea99b91	active	No
10.0.0.16	pcx-05e79f11d70ec5436	active	No

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

Destination	Target	Status	Propagated
10.0.0.16	local	active	No
20.0.0.16	pcx-05e79f11d70ec5436	active	No

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

Name	Route Table ID	Explicit subnet association	Edge associations	Main	VPC ID	Owner
Default	rtb-276df559	-	-	Yes	vpc-c1eeff3bb Default	607665745996
defaultpublicroutetable vpc1	rtb-0d60a60170a01...	-	-	Yes	vpc-02e3ee9e144734ee0 vpc-1	607665745996
vpc02 peering	rtb-088bd657d48b...	-	-	No	vpc-06a125d83294da0fc vpc-2	607665745996
vpc01 peering	rtb-05f08cc07b9ca...	-	-	No	vpc-02e3ee9e144734ee0 vpc-1	607665745996
defaultpublicroutetablevpc02	rtb-05cd51f76199d...	subnet-08702ab15a8eb1d5	-	Yes	vpc-06a125d83294da0fc vpc-2	607665745996

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

Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot Instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

Number of Instances: 1

Purchasing option: Request Spot Instances

Network: vpc-02e3ee9e144734ee0 | vpc-1

Subnet: subnet-02f2d23e60c33640d | publicsubnet-vpc01 | u | Create new subnet
251 IP Addresses available

Auto-assign Public IP: Enable

Placement group: Add instance to placement group

Capacity Reservation: Open

IAM role: None

Shutdown behavior: Stop

Stop - Hibernate behavior: Enable hibernation as an additional stop behavior

Enable termination protection: Protect against accidental termination

Buttons: Cancel, Previous, Review and Launch (highlighted), Next: Add Storage

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

Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot Instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

Number of Instances: 1

Purchasing option: Request Spot Instances

Network: vpc-06a125d83294da0fc | vpc-2

Subnet: subnet-06835d8dff6ead1b | privatesubnet-vpc02 | u | Create new subnet
251 IP Addresses available

Auto-assign Public IP: Use subnet setting (Disable)

Placement group: Add instance to placement group

Capacity Reservation: Open

IAM role: None

Shutdown behavior: Stop

Stop - Hibernate behavior: Enable hibernation as an additional stop behavior

Buttons: Cancel, Previous, Review and Launch (highlighted), Next: Add Storage

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

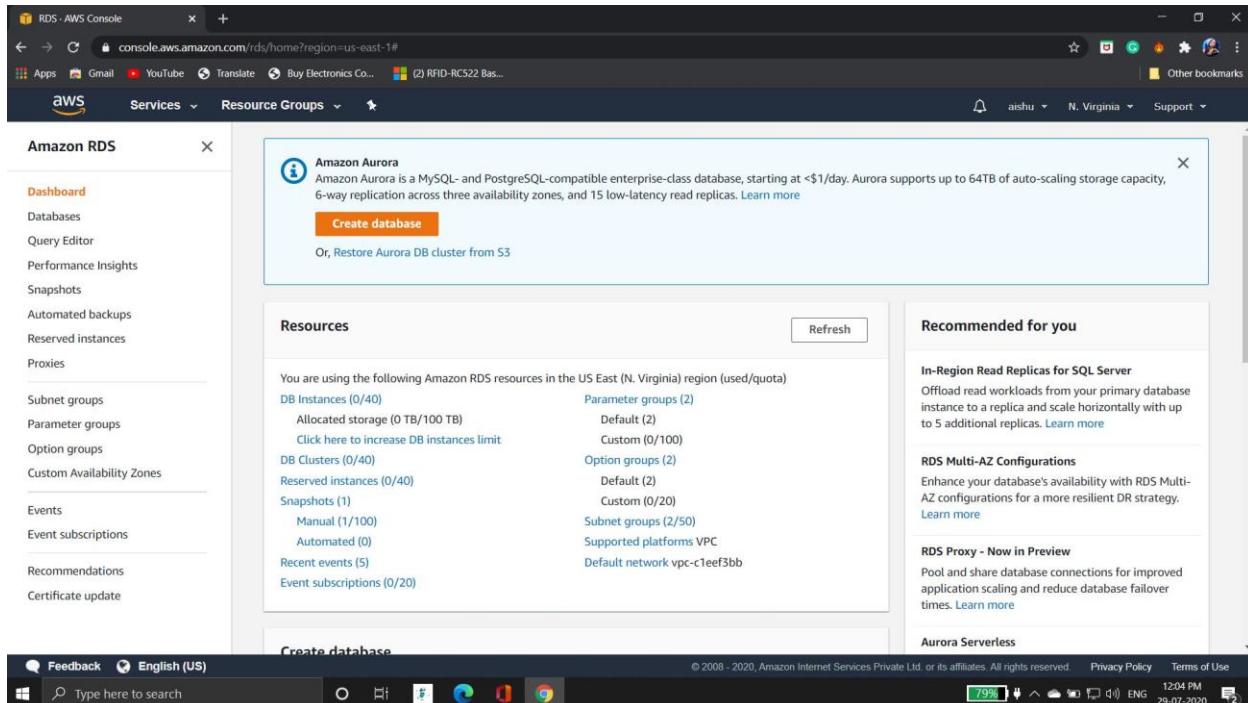
And here you've successfully logged into your private instance sitting in another VPC.

```
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

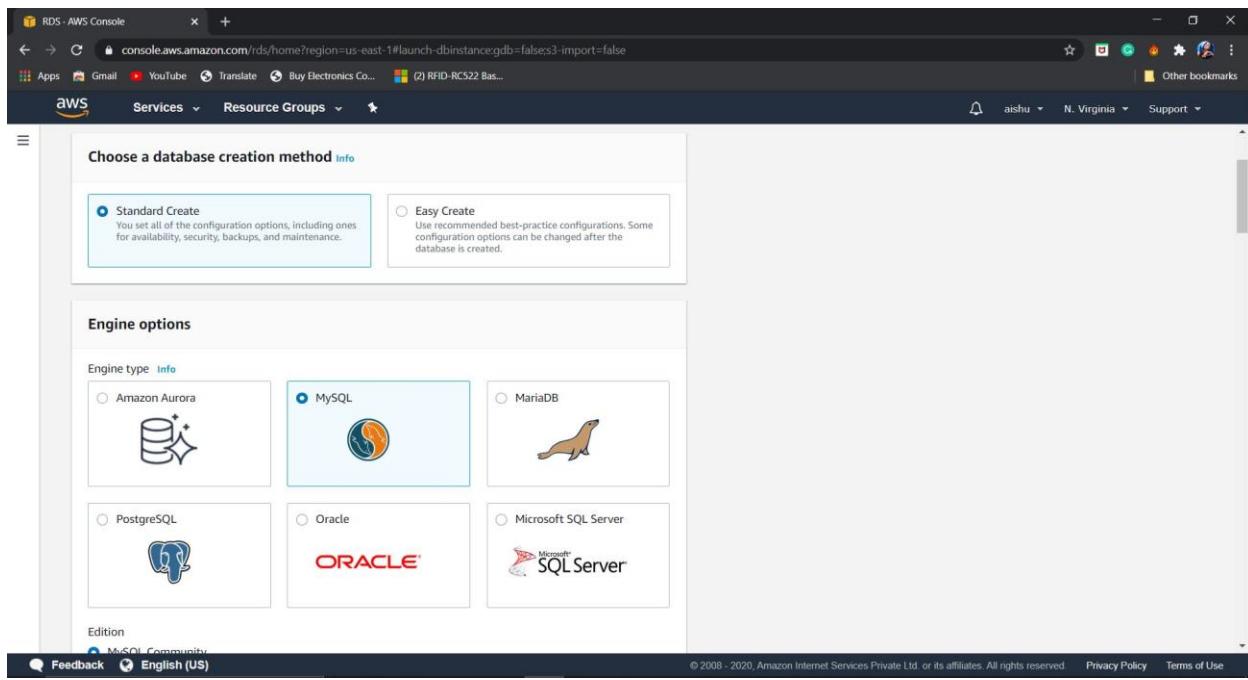
Launching an RDS Instance and connecting it with EC2 Instance

Step 1: Log in to RDS Console

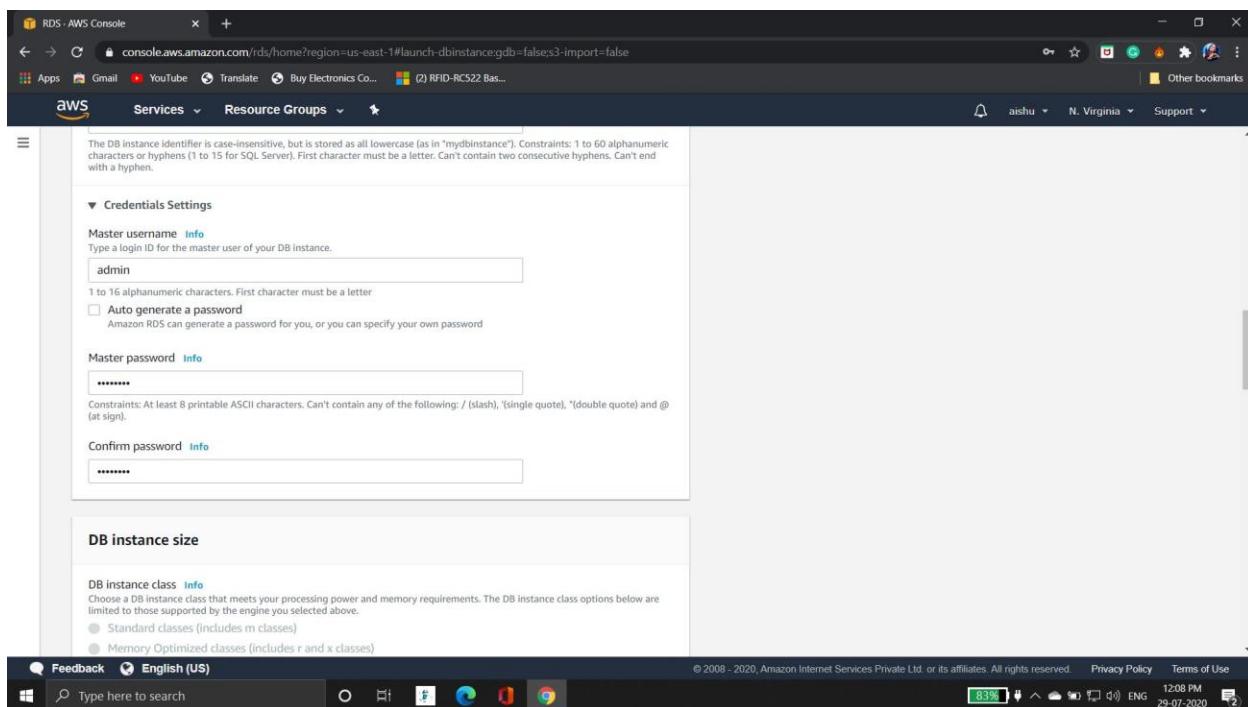


The screenshot shows the AWS RDS console for the Amazon Aurora service. The left sidebar lists various options like Dashboard, Databases, Query Editor, etc. The main content area displays information about Aurora, including a 'Create database' button and a note about restoring from S3. Below this is a 'Resources' section showing usage statistics for DB Instances, DB Clusters, Reserved instances, Snapshots, Recent events, and Event subscriptions. To the right, there's a 'Recommended for you' sidebar with links to In-Region Read Replicas for SQL Server, RDS Multi-AZ Configurations, RDS Proxy (Now in Preview), and Aurora Serverless.

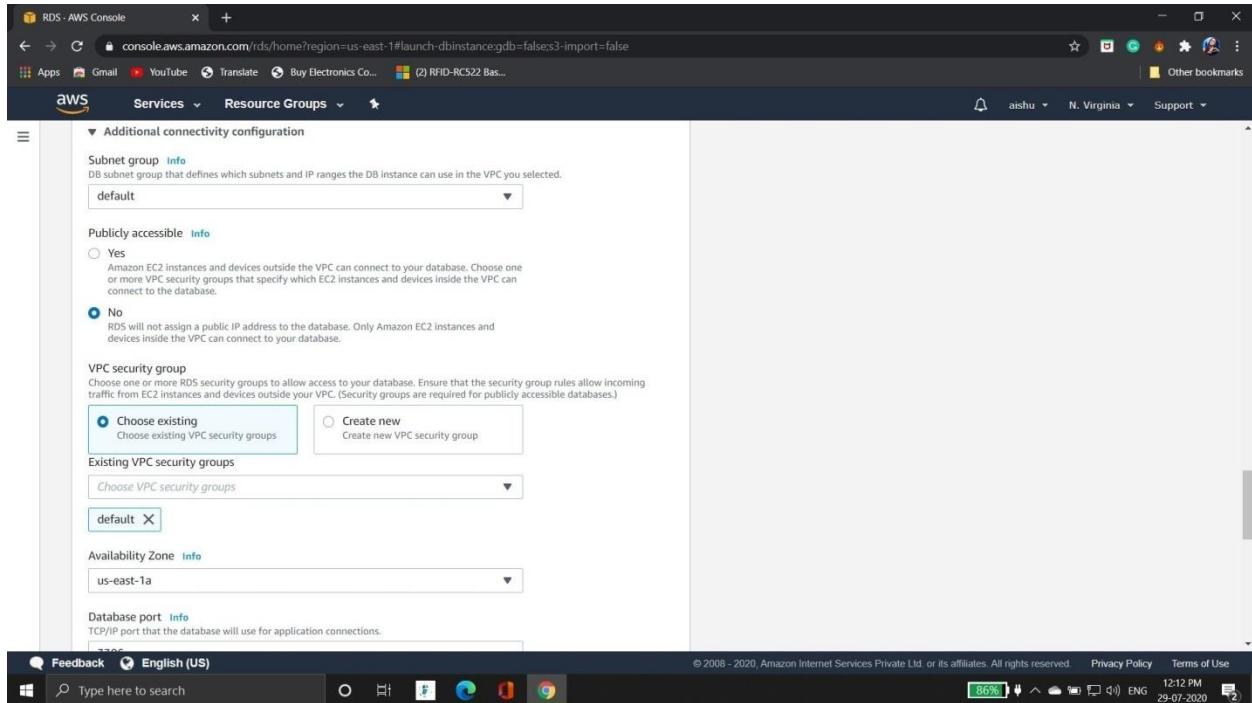
Step 2: Click on “Create Database” and select one of the database type. It is recommended to choose MySQL, Maria DB if you’re in Free Tier.
In this lab session we will choose, MySQL.



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

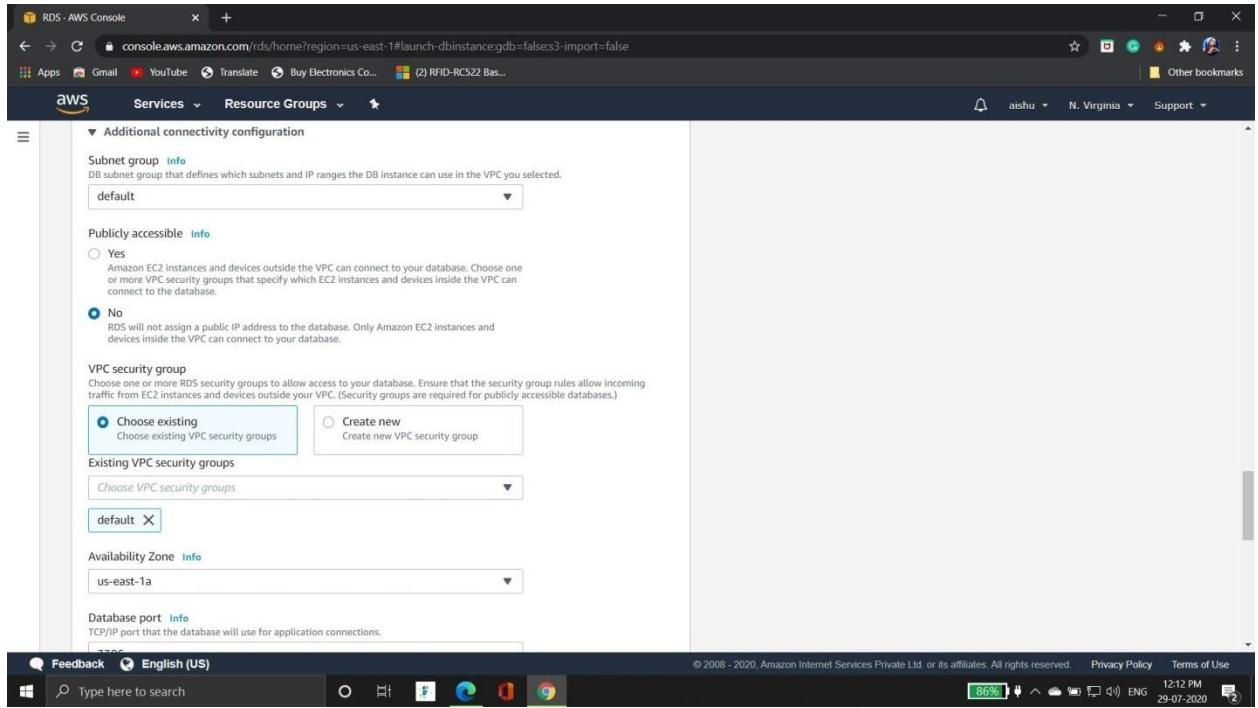


Step 4: Click on the Free Tier eligible for free tier configuration or you can choose any instance type

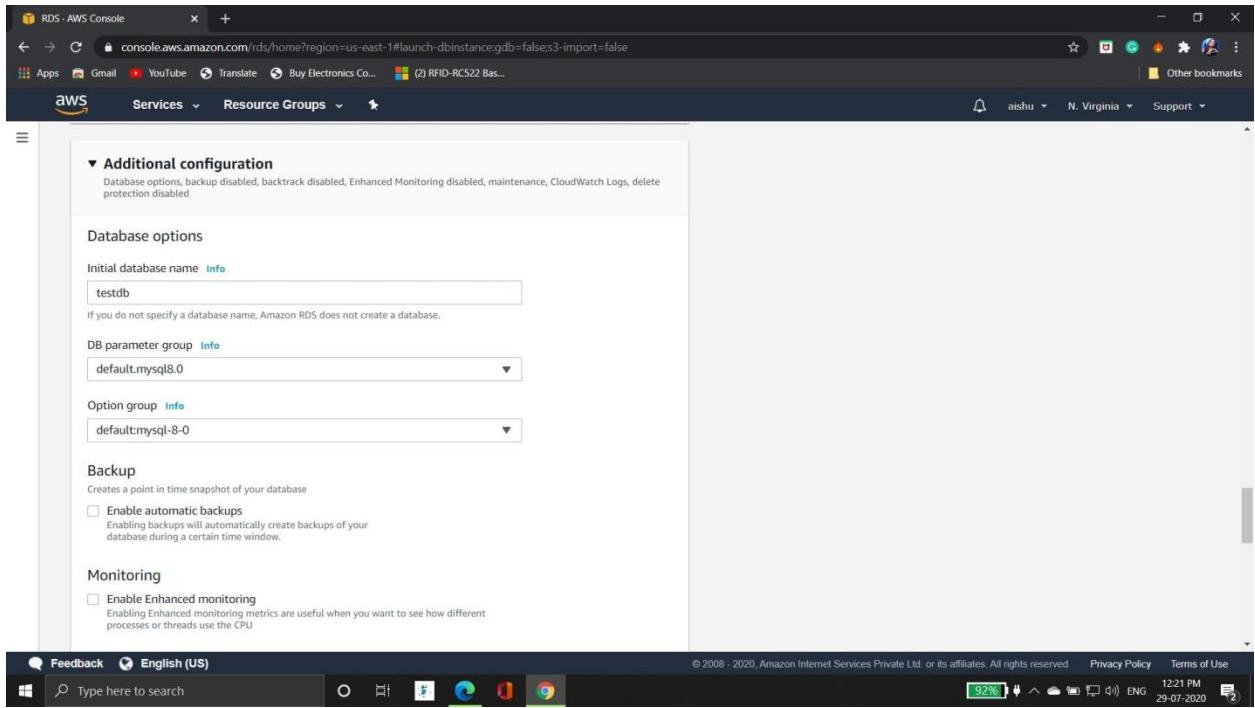


Step 5: Allocate storage for the database, min – 20GiB

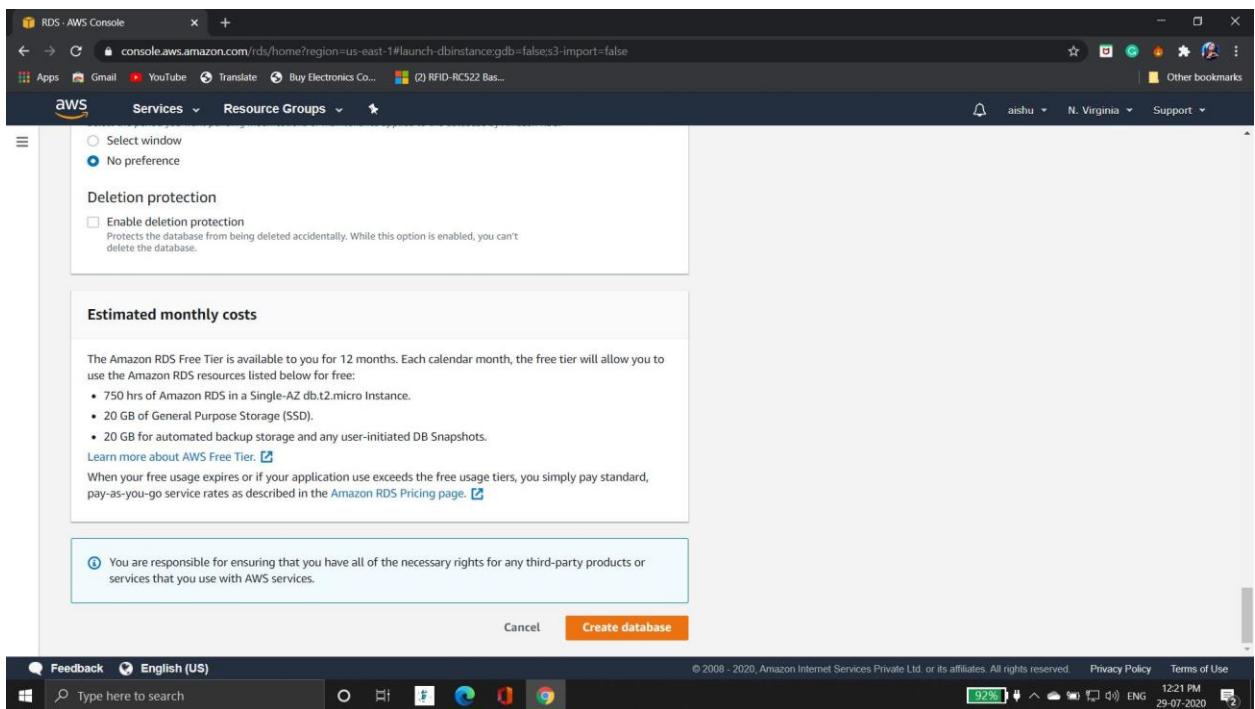
Step 6: Now, tag the DB Instance and give master username and password



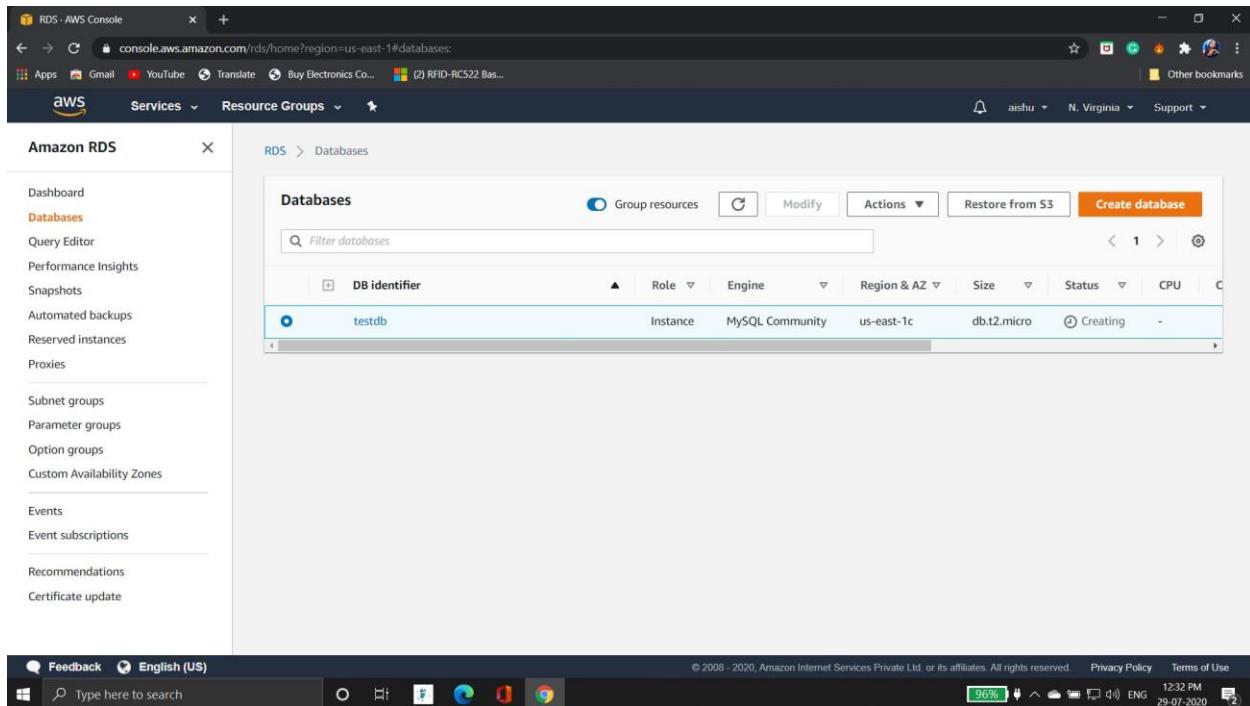
Step 7: Now choose the custom VPC in your Env if any otherwise go with default one and then choose a new private subnet for DB. Also select to create new Security Group



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

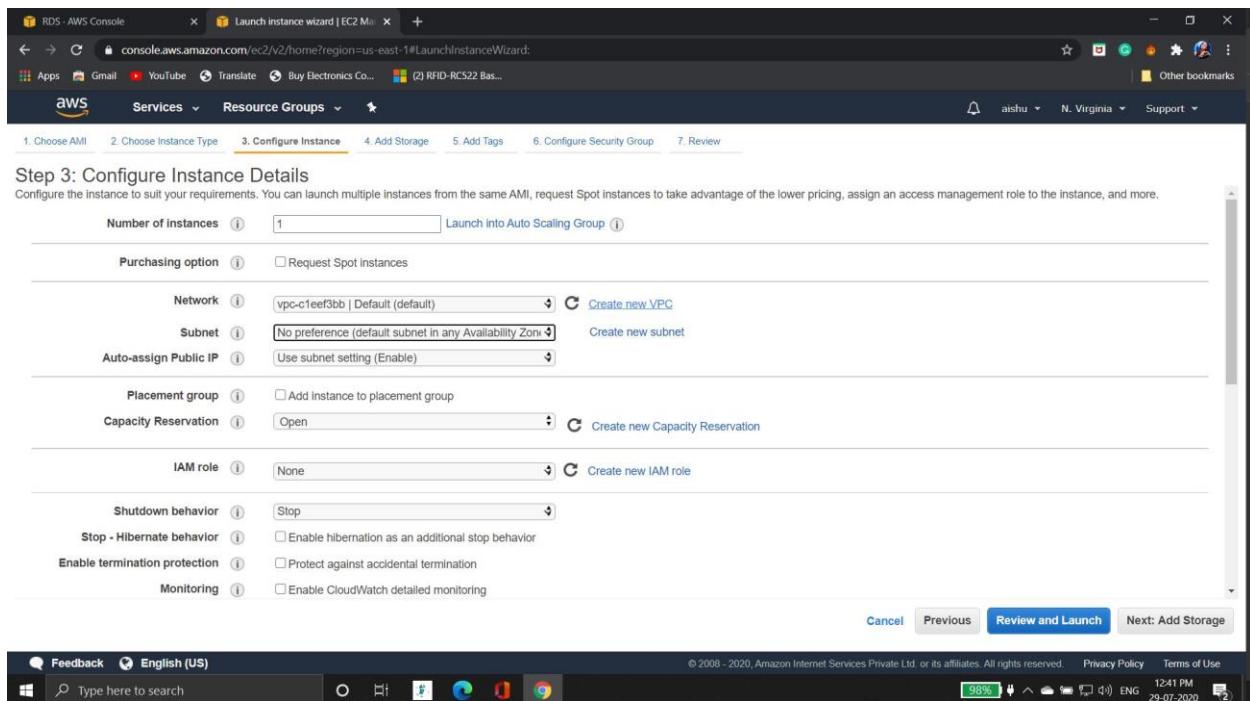


The screenshot shows the AWS RDS console with the URL console.aws.amazon.com/rds/home?region=us-east-1#databases;. The left sidebar is titled 'Amazon RDS' and includes options like Dashboard, Databases (which is selected), Query Editor, Performance Insights, Snapshots, Automated backups, Reserved instances, Proxies, Subnet groups, Parameter groups, Option groups, Custom Availability Zones, Events, Event subscriptions, Recommendations, and Certificate update. The main content area is titled 'Databases' and shows a table with one row:

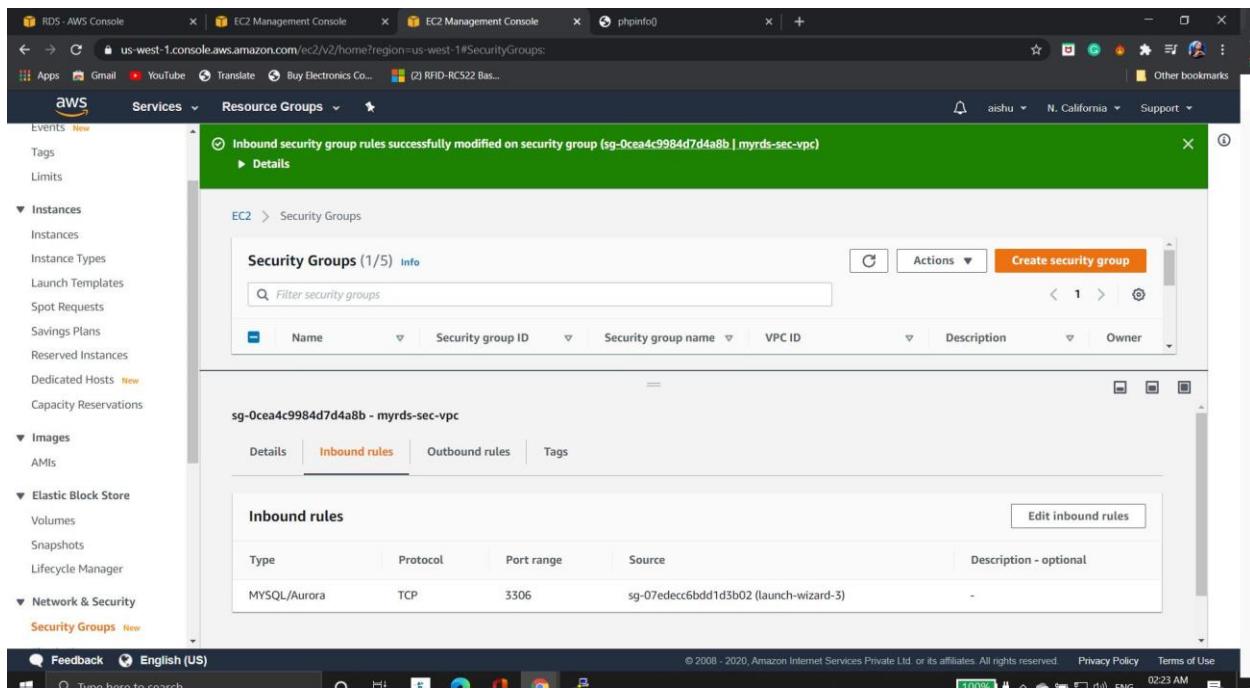
DB identifier	Role	Engine	Region & AZ	Size	Status	CPU
testdb	Instance	MySQL Community	us-east-1c	db.t2.micro	Creating	-

The status column for 'testdb' shows 'Creating'. The bottom of the screen shows the Windows taskbar with icons for Feedback, English (US), a search bar, and system status indicators.

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

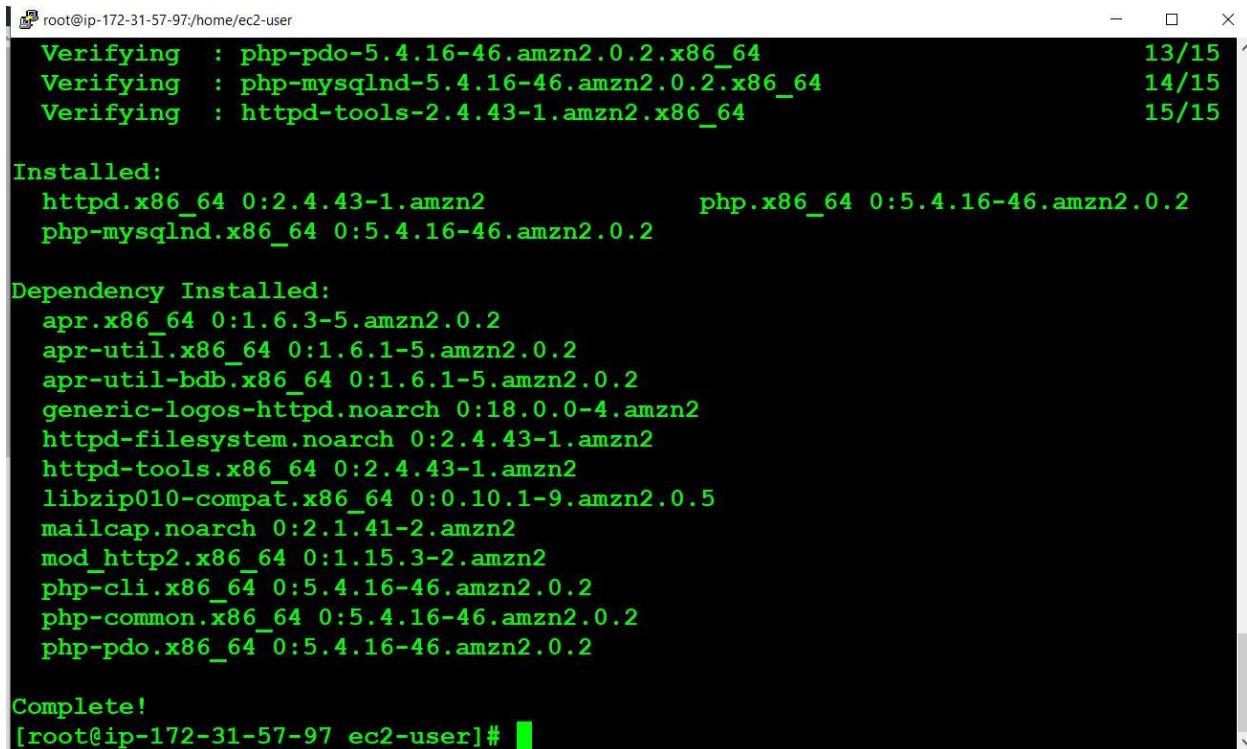


Step 11: Now SSH to your EC2 Instance.

A screenshot of a terminal window titled "root@ip-172-31-57-97:/home/ec2-user". The terminal shows the user has logged in as "ec2-user" and is using a public key for authentication. It then prompts for a password, which is masked. The user runs "sudo su" to become root. The terminal output includes the Amazon Linux 2 AMI logo and a link to the Amazon Linux 2 repository. The terminal window has a dark background with light-colored text.

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

Command: “*yum install httpd php php-mysql -y*”



```
root@ip-172-31-57-97:/home/ec2-user
Verifying : php-pdo-5.4.16-46.amzn2.0.2.x86_64 13/15
Verifying : php-mysqlnd-5.4.16-46.amzn2.0.2.x86_64 14/15
Verifying : httpd-tools-2.4.43-1.amzn2.x86_64 15/15

Installed:
httpd.x86_64 0:2.4.43-1.amzn2           php.x86_64 0:5.4.16-46.amzn2.0.2
php-mysqlnd.x86_64 0:5.4.16-46.amzn2.0.2

Dependency Installed:
apr.x86_64 0:1.6.3-5.amzn2.0.2
apr-util.x86_64 0:1.6.1-5.amzn2.0.2
apr-util-bdb.x86_64 0:1.6.1-5.amzn2.0.2
generic-logos-httpd.noarch 0:18.0.0-4.amzn2
httpd-filesystem.noarch 0:2.4.43-1.amzn2
httpd-tools.x86_64 0:2.4.43-1.amzn2
libzip010-compat.x86_64 0:0.10.1-9.amzn2.0.5
mailcap.noarch 0:2.1.41-2.amzn2
mod_http2.x86_64 0:1.15.3-2.amzn2
php-cli.x86_64 0:5.4.16-46.amzn2.0.2
php-common.x86_64 0:5.4.16-46.amzn2.0.2
php-pdo.x86_64 0:5.4.16-46.amzn2.0.2

Complete!
[root@ip-172-31-57-97 ec2-user]#
```

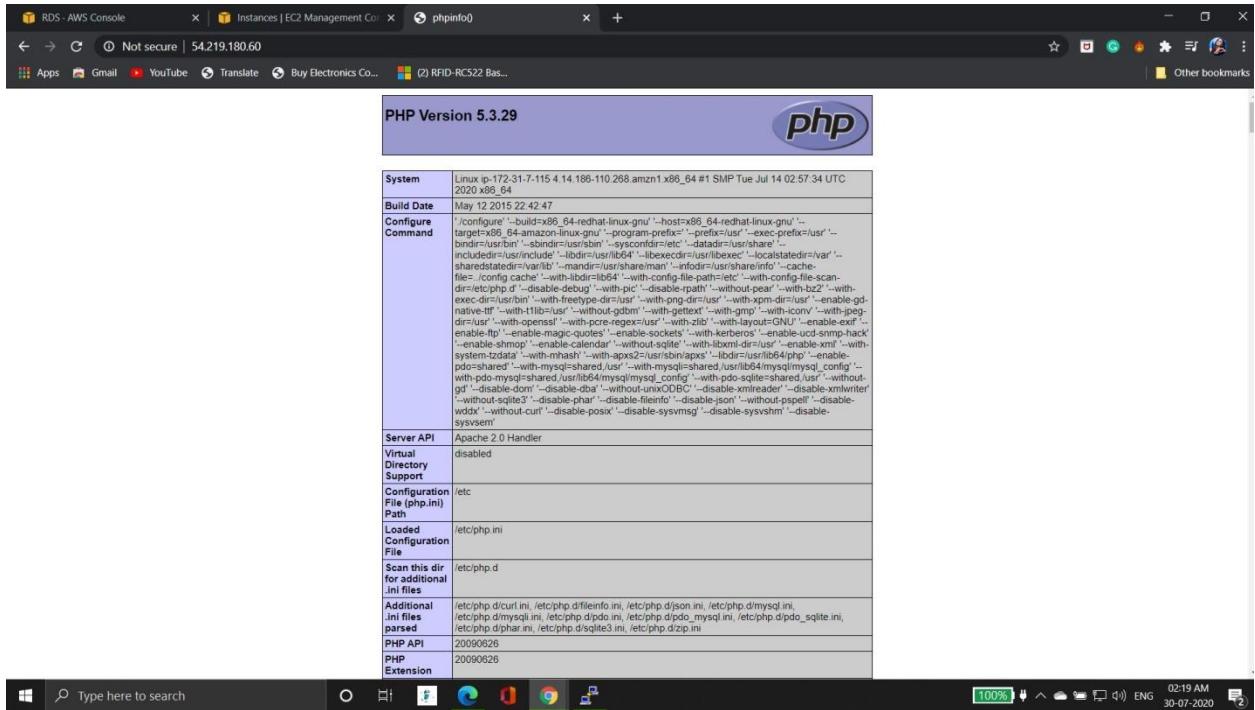
Step 13: Once installed, start the server and create test PHP page

Command: “*service httpd start*”

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

```
root@ip-172-31-7-115:/home/ec2-user
https://aws.amazon.com/amazon-linux-ami/2018.03-release-notes/
[ec2-user@ip-172-31-7-115 ~]$ sudo su
[root@ip-172-31-7-115 ec2-user]# yum install httpd php my-sql -y
Loaded plugins: priorities, update-motd, upgrade-helper
amzn-main                                         | 2.1 kB     00:00
amzn-updates                                     | 3.8 kB     00:00
Package httpd-2.2.34-1.16.amzn1.x86_64 already installed and latest version
Package php-5.3.29-1.8.amzn1.x86_64 already installed and latest version
No package my-sql available.
Nothing to do
[root@ip-172-31-7-115 ec2-user]# service httpd status
httpd (pid 2740) is running...
[root@ip-172-31-7-115 ec2-user]# echo "<?php phpinfo();?>" /var/www/html/index.php
>
<?php phpinfo();?> /var/www/html/index.php
[root@ip-172-31-7-115 ec2-user]# echo "<?php phpinfo();?>" > /var/www/html/index.php
[root@ip-172-31-7-115 ec2-user]#
```

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.

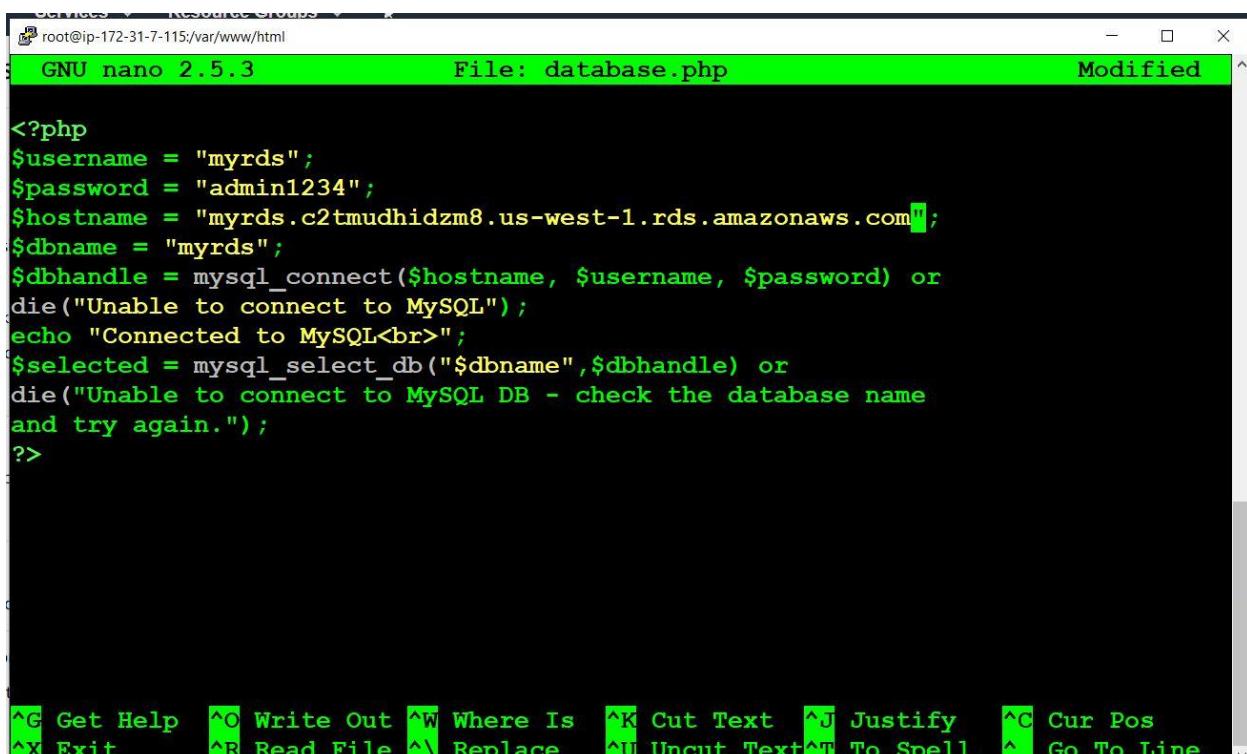
Code:

```
<?php
$username = "awsreconnect";
$password = "awsreconnect";
$hostname = "awsreconnect.cfjmmzvhedp.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 parameters

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



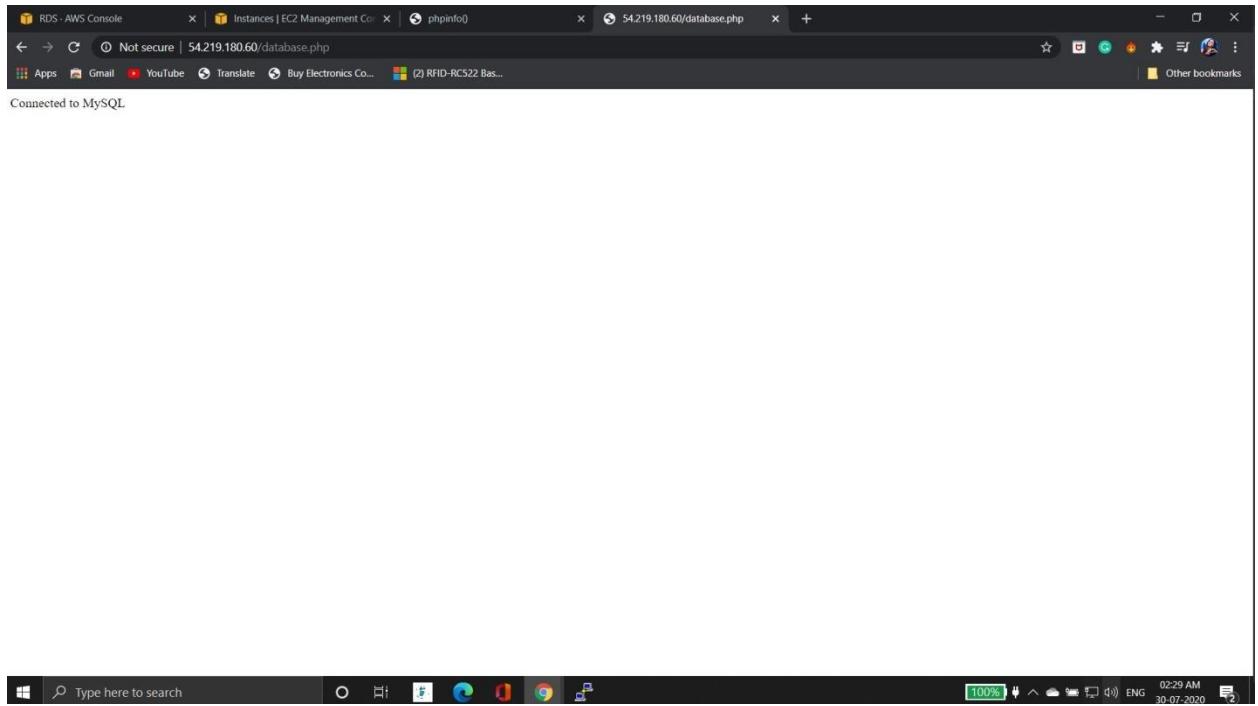
The screenshot shows a terminal window titled "root@ip-172-31-7-115:/var/www/html". Inside the terminal, the nano 2.5.3 editor is open with a file named "database.php". The code in the file is as follows:

```
<?php  
$username = "myrds";  
$password = "admin1234";  
$hostname = "myrds.c2tmudhidzm8.us-west-1.rds.amazonaws.com";  
$dbname = "myrds";  
$dbhandle = mysql_connect($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.");  
?>
```

At the bottom of the terminal window, there is a menu bar with various keyboard shortcuts for navigating the nano editor.

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”



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, Public 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": "Public-Subnet"  
            }  
]  
    }  
}
```

```
"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 Subnet"  
},  
"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 Subnet"
},
"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"
},
```



```
"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 existing Key",
    "Type": "AWS::EC2::KeyPair::KeyName"
},
"bastionSGDestinationAddress": {
    "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 "  
},  
"bastionSGDestinationAddress": {  
    "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/>