Project Topic:

Troubleshooting, Design & implementing Clients Windows 10 and 11 in Centralized CentOS Server administration in AWS Cloud infrastructure.

Team Members

Pratik Bagade

Rutik Borate

Balaji Dande

Pankaj Kaushik

Mayuri Londhe

Rachana Mangalaram

Under the Guidance of

Zakir Hussain

I. Introduction

- Overview of AWS cloud infrastructure

AWS (Amazon Web Services) is a comprehensive cloud platform that offers on-demand computing resources. It eliminates the need for physical infrastructure by allowing users to provision virtual machines (EC2 instances), databases, and networks in a scalable, flexible environment. AWS supports a wide range of operating systems, including Windows and Linux (like CentOS), making it ideal for hosting multi-OS environments.

In this project , AWS is used to host both CentOS and Windows 10/11 virtual machines (VMs) in a real-time scenario, enabling cross-platform operations and administration tasks. It makes it easier to manage, configure, and troubleshoot OS-level issues

In this context, AWS plays a crucial role in enabling hybrid cloud environments, where Windows clients and CentOS servers coexist and function together seamlessly.

- Importance of CentOS Administration

CentOS (Community Enterprise Operating System) is a powerful, enterprise-grade Linux distribution derived from Red Hat Enterprise Linux (RHEL).

In cloud infrastructure, CentOS is often used for server-side applications, management of networks, and resource allocation. Proficient CentOS administration is crucial for managing services like DNS, file sharing (using Samba), and remote administration. It also serves as a bridge between different operating systems, such as integrating Windows clients into Linux-based networks, which is a common requirement in enterprise environments.

Effective CentOS administration is vital for ensuring smooth operation, especially in environments where Windows and Linux systems need to collaborate.

- Windows 10/11 client OS integration

Windows 10/11 is often the standard for end-user workstations, especially in corporate environments. The integration of Windows clients with CentOS servers involves tasks like domain management, file sharing, and remote access. In AWS cloud scenarios, Windows clients may need to authenticate with Linux-based servers, access shared directories, or participate in secure, controlled environments managed by CentOS. Seamlessly connecting these two environments requires expertise in networking, security protocols, and OS-level configuration.

This integration allows the seamless exchange of data between Windows users and Linux servers, fostering productivity while maintaining a secure environment.

II. Design and Implementation Considerations

Chapter 1

Planning and Designing Windows OS on CentOS

When integrating Windows 10/11 client OS with CentOS in a cloud-based infrastructure (such as AWS), proper planning and design are essential to ensure seamless operation, efficient resource utilization, and robust security. In this setup, we have utilized Windows Server 2019 and Windows Server 2022 to connect to the centralized CentOS 7 server, ensuring compatibility and communication between instances. This architecture allows for streamlined management of resources while maintaining robust security measures across all connected systems.

- Hardware requirements

Local Machine:

Host OS: Windows Family.

RAM: At least 8 GB (preferably 16 GB or more to run multiple VMs and connections simultaneously).

Processor: Intel i5 or higher (or equivalent AMD processor).

Storage: SSD with 50 GB or more of free space.

Network: Stable broadband internet connection to access AWS services and remote instances.

- Software requirements

1) AWS Cloud:

EC2 Instances: Windows Server 2019, Windows Server 2022, CentOS 7.

VPC Configuration: Custom VPC with subnets and security groups.

Key Pairs: Generated in AWS for instance access.

2) Local Machine:

Oracle VirtualBox: Installed to access the instances from Windows 10 VM.

RDP Client: For connecting to Windows instances via Remote Desktop.

SSH Client (e.g., PuTTY): To securely connect to CentOS instances.

3) CentOS 7 Configuration:

Samba: Installed and configured for file sharing and communication between Windows instances and CentOS.

Firewall Management: Configured to allow necessary traffic, such as ICMP (for ping), SSH, and Samba.

4) Windows Configuration:

Remote Desktop Services: Enabled and configured for Windows instances.

Network Settings: Configured to match the subnet and gateway settings for communication with CentOS.

- Network architecture

1. Virtual Private Cloud (VPC):

All instances are deployed in a single VPC.The VPC CIDR block is configured to accommodate your private IP range, ensuring both Windows and CentOS instances reside within this address space.

2. Subnets:

Subnet is assigned to the instances (Windows Server 2019 and Windows Server 2022 & CentOS), with a CIDR block that covers the IP addresses assigned to them.

3. Security Groups:

A security group acts as a virtual firewall for your instance. It controls inbound and outbound traffic based on rules we define.

A single security group is used for all instances, allowing the necessary traffic.

- RDP (port 3389) is open for remote desktop connections to both Windows instances.

- SSH (port 22) is open for access to the CentOS instance.

- ICMP (All ICMP - IPv4) is enabled for pinging across all instances.

- All traffic is allowed between the Windows and CentOS instances to ensure smooth communication.

4. Instances:

Instances refer to virtual servers that run applications and workloads on the cloud. These instances are part of the Amazon EC2 (Elastic Compute Cloud) service, which provides scalable computing capacity in the cloud.

Windows 1 (Microsoft Windows Server 2019 Base):

- Private IP: 192.168.1.45

- Security Group: Shared with Windows 2

- Key pair: win.pem

Windows 2 (Microsoft Windows Server 2022 Base):

- Private IP: 192.168.1.26

- Security Group: Same as Windows 1

- Key pair: win.pem

CentOS 7 instance (Centralized server):

- Private IP: Falls within the VPC range (for example: 192.168.1.x)

- Acts as a centralized server for both Windows instances.

- Key pair: Unique to CentOS

5. AMI (Amazon Machine Image):

An Amazon Machine Image (AMI) is a template that contains the software configuration (OS, application server, applications) required to launch an instance. When you launch an EC2 instance, you choose an AMI to specify which operating system and software environment the instance will have.

6. Key Pair:

When launching an instance, AWS requires you to select or create a key pair. This key pair consists of a private key (downloaded by the user) and a public key (stored in AWS). It's used to securely connect to the instance via SSH (for Linux instances) or RDP (for Windows instances).

7. Firewall Configuration:

Firewalls on both Windows instances are disabled to allow for smooth communication (ping, RDP, etc). Ensure that CentOS also has necessary firewall rules for inbound/outbound traffic for ICMP, SSH.

8. Routing Table:

A single route table is used within the VPC to manage internal routing between all subnets (Windows & CentOS) so that they can communicate with each other.

9. Internet Gateway

A component of VPC that allows communication between your VPC and the internet. It supports IPv4 and IPv6 traffic. It does not cause availability risks or bandwidth constraints on your network traffic.

10. Connection Flow:

Windows Instances: Can communicate with each other via ping (ICMP), and you can connect to both via RDP using their private IP addresses.

CentOS Instance: Acts as the centralized server, connecting to both Windows instances. You can SSH into CentOS, and CentOS should be able to ping and manage both Windows instances.

11. SELinux

The command chcon -t samba_share_t /samba/apps used to change the SELinux (Security-Enhanced Linux) security context of the /samba/apps directory to the type samba_share_t. This is necessary for making the directory accessible to the Samba service, which enables file sharing between Linux and Windows systems.

12. IAM

AWS IAM (Identity and Access Management) helps you manage and control access to AWS services and resources securely. It allows you to create users, roles, and groups, and define permissions using policies, ensuring that only authorized people or services have access to specific resources.

13. Monitoring:

AWS offers CloudWatch to monitor instance performance and resource usage, like CPU, memory, and network traffic.

Chapter 2

Implementing Windows OS on CentOS

- Installation and configuration

Go to AWS Account, Login as root user

Select Region and click on Create VPC

Select and perform the following

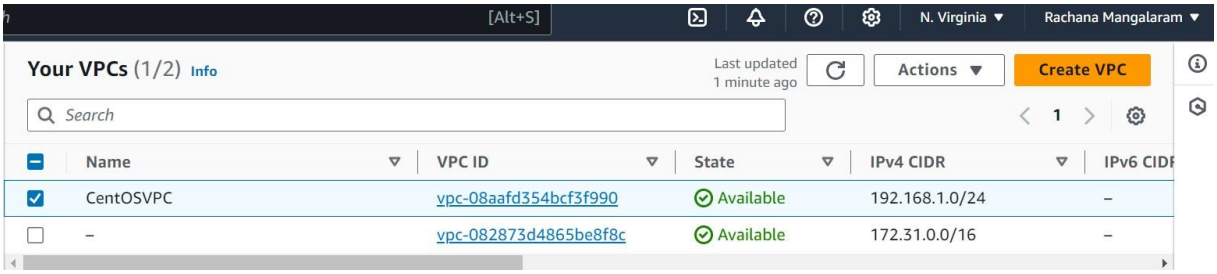
- VPC only

- Give name to the VPC

- Add IPv4 CIDR (192.168.1.0/24)

- Set tenancy as default itself

- Click on create VPC



Go to Subnet

- Click on create subnet

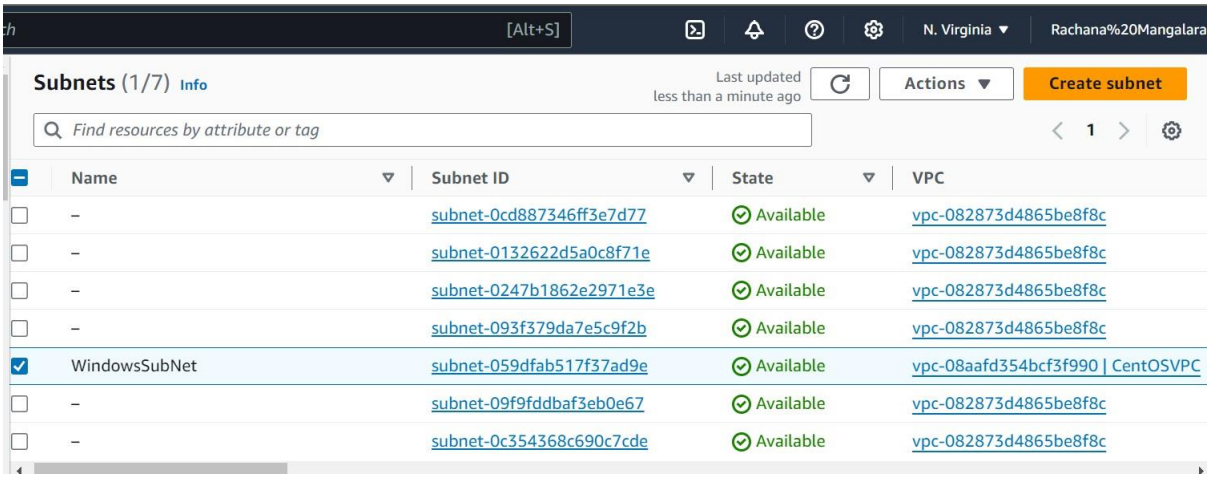
- Select the VPC which has been created, i.e CentOSVPC

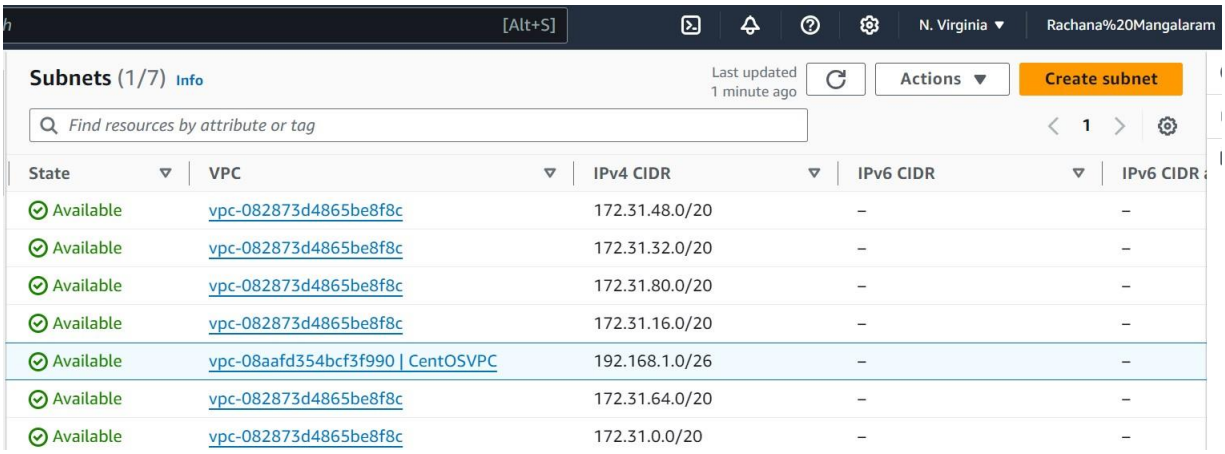
- Name the subnet

- Choose the AZ

- Add the IPv4 subnet cidr block as 192.168.1.0/26

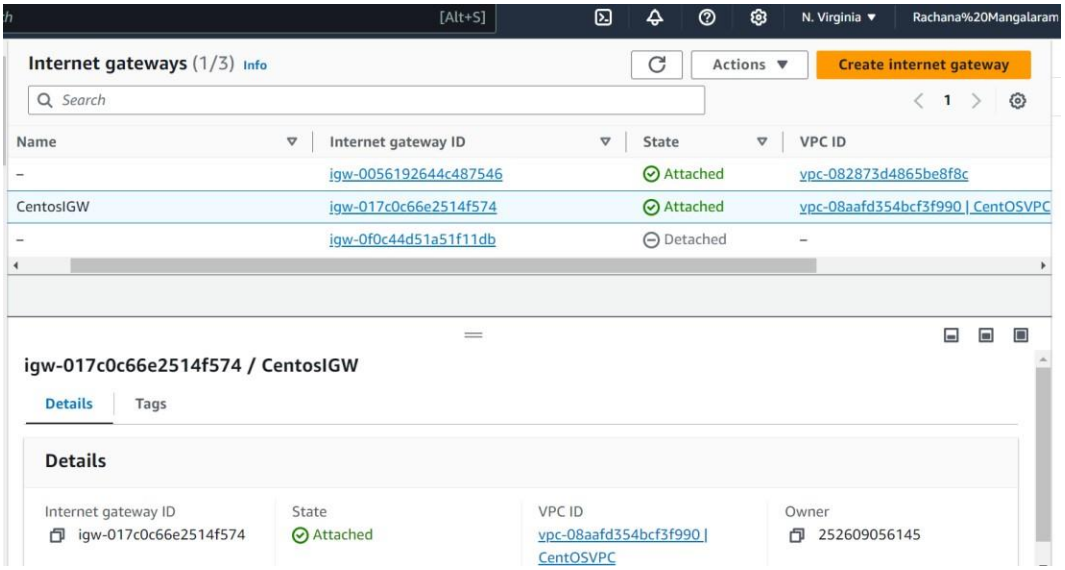
- Click on Create Subnet





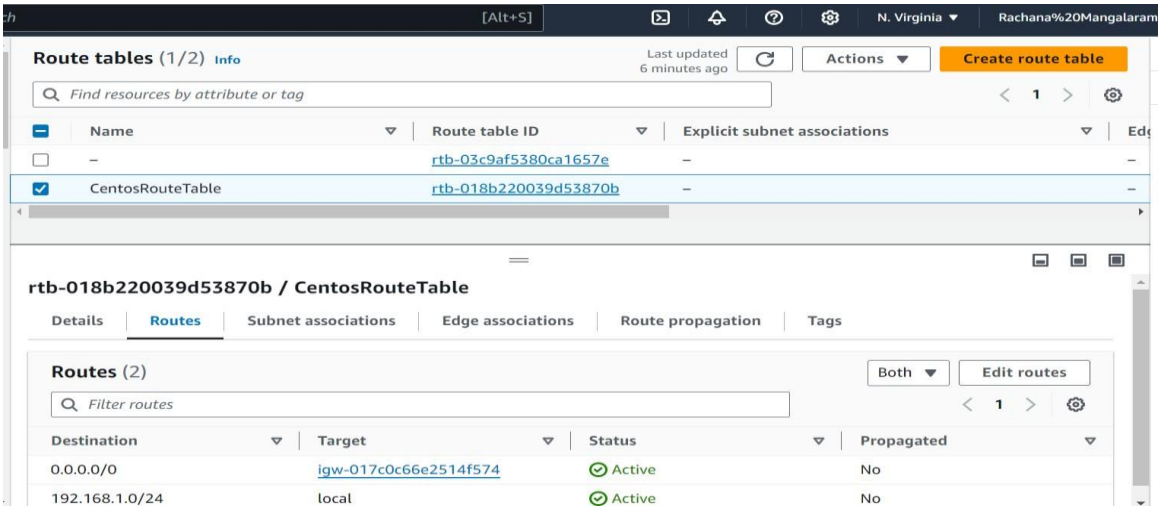
Create an Internet gateway by giving its name, and select it, click on Actions -> Attach to VPC

Select the VPC (CentOSVPC) and click on attach internet gateway

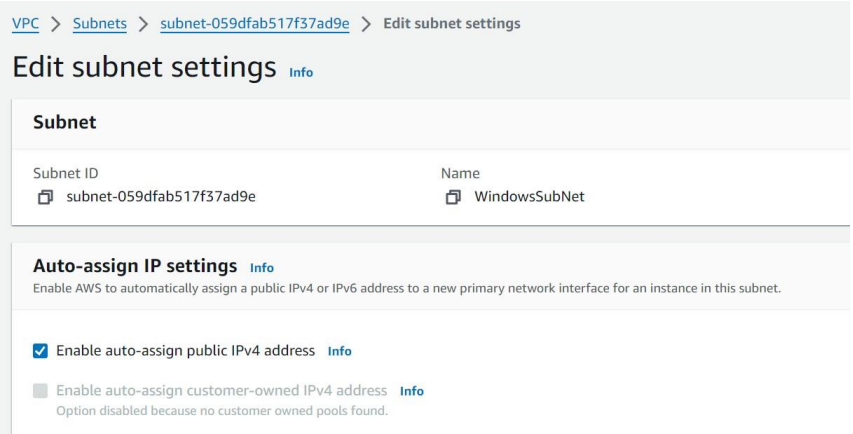


Go to Route Tables

Select the Main Route table of the VPC

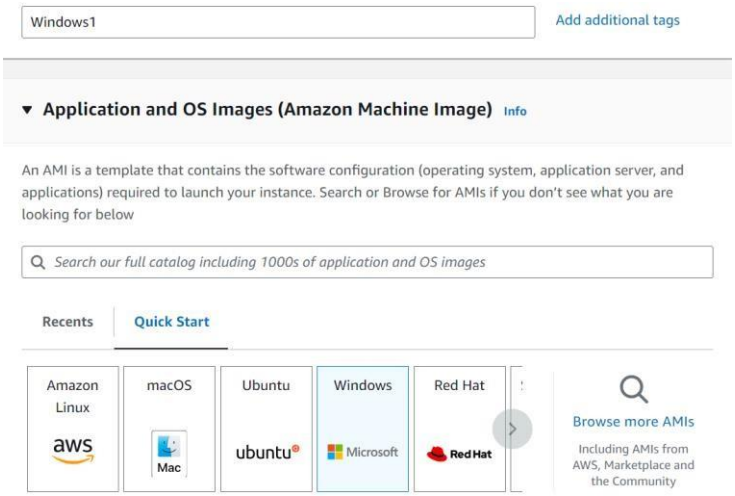


Go to the subnet, click on the subnet which you have created, edit subnet settings, enable auto-assign public ipv4 address-> click on save

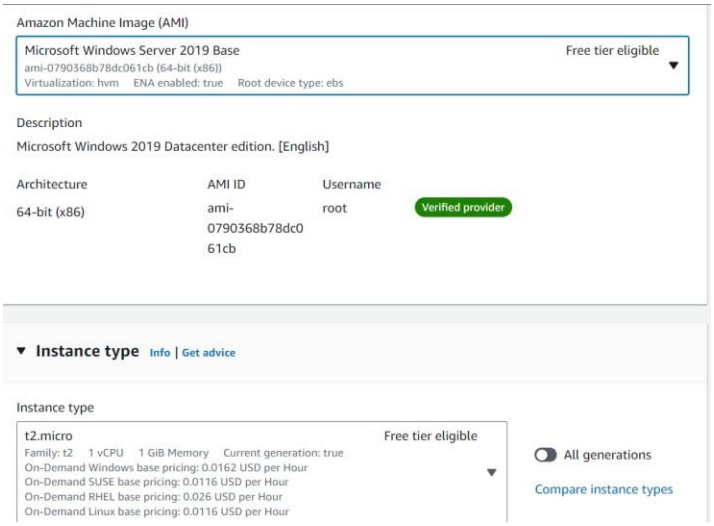


Now, lets proceed to Instances creation step:

Go to Instances-> Launch instances ->Give name to the Instance 1 and select windows OS, select Windows 2019 Base



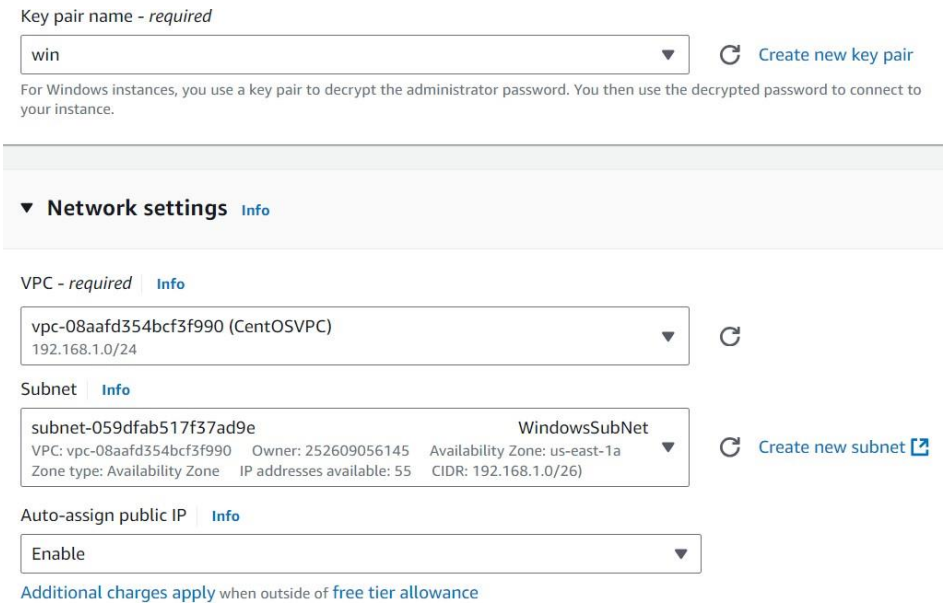
Select the t2.micro



Select Create new Key pair, Name it is win.pem-> Click on Create key pair(it gets downloaded)

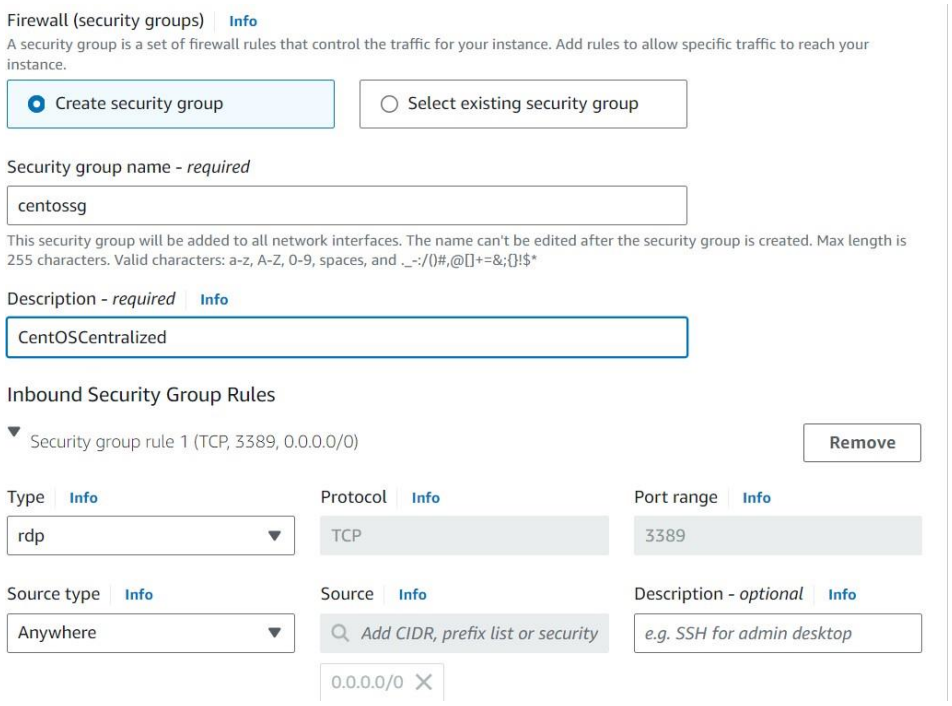
Network Settings:

- Select **CentOSVPC** as VPC
- Select **WindowsSubnet** as its Subnet
- **Auto-assign public IP**: Enable.



Select create Secuirty group, name it, add its description

- Add inbound rules like, RDP, SSH, All ICMP- Ipv4 and All traffic



Then, Click on Launch Instance

Follow the Same steps for other two instances, named:

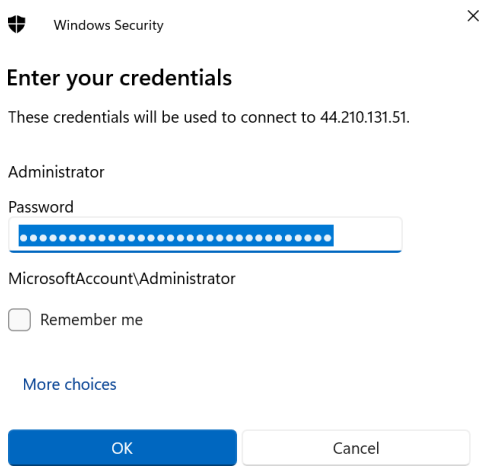
Windows2 -> AMI (Windows 2022 Base)-> t2.micro ->Select existing Key pair i.e win-> Same Network settings just like used in Windows1 and select existing Secuirty group and then click on Launch instance

CentOS -> AMI (CentOS 7)->t2.micro-> Create new key pair-> Name it as centoskp-> Same Network settings just like used in Windows1 and select existing Secuirty group and then click on Launch instance.

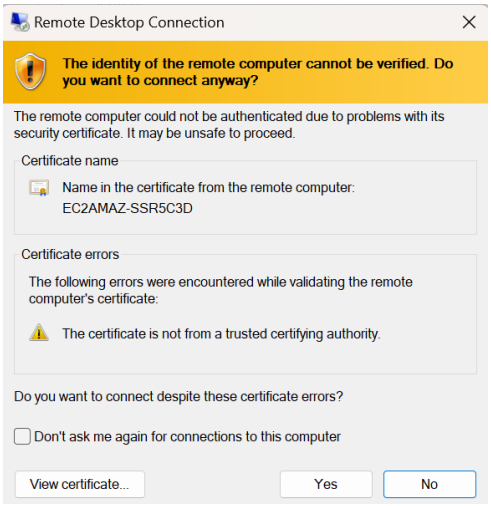
All three instances are in same network and subnet, i.e the can communicate with each other.

Copy the Public IP address of Windows1 instance -> Paste it in Remote Desktop Connection-> Connect

Enter username (Administrator will be by default), and get the password, by selecting the instance, go to Connect-> RDP Client-> Get Password -> Upload the Private Key File i.e .pem-> Decrypt Password-> Copy Paste that password in the Password field of Remote Desktop Conn-> Click on OK

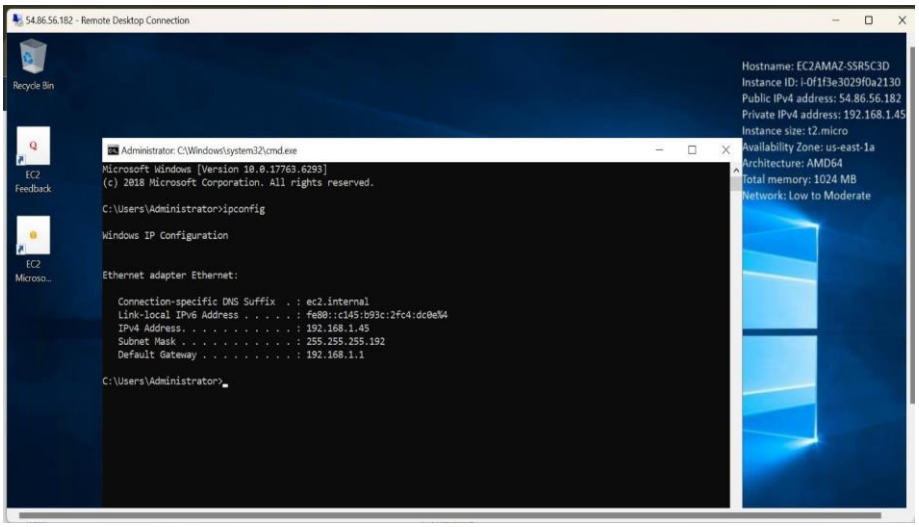


Click on Yes

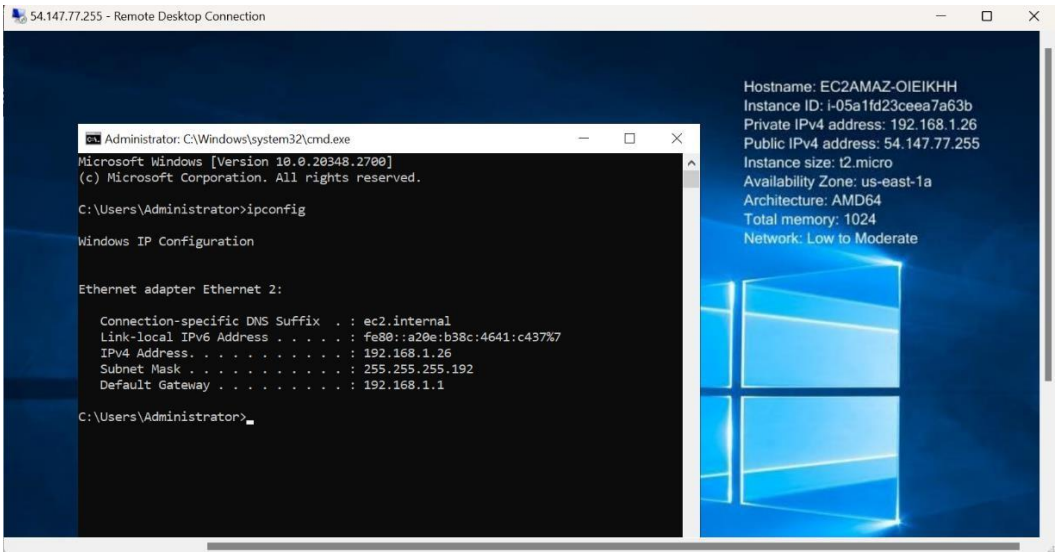


(Similar steps for Windows2 instance to connect to Remote Desktop)

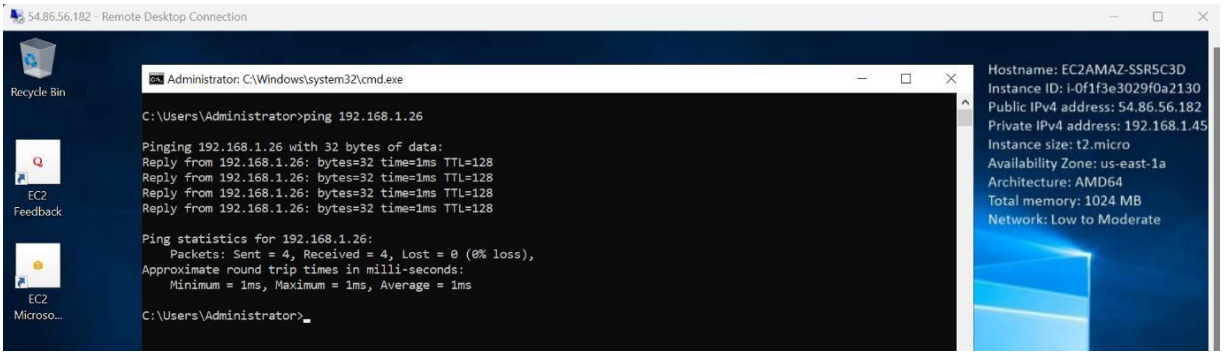
Go to Command Prompt of Windows1 instance



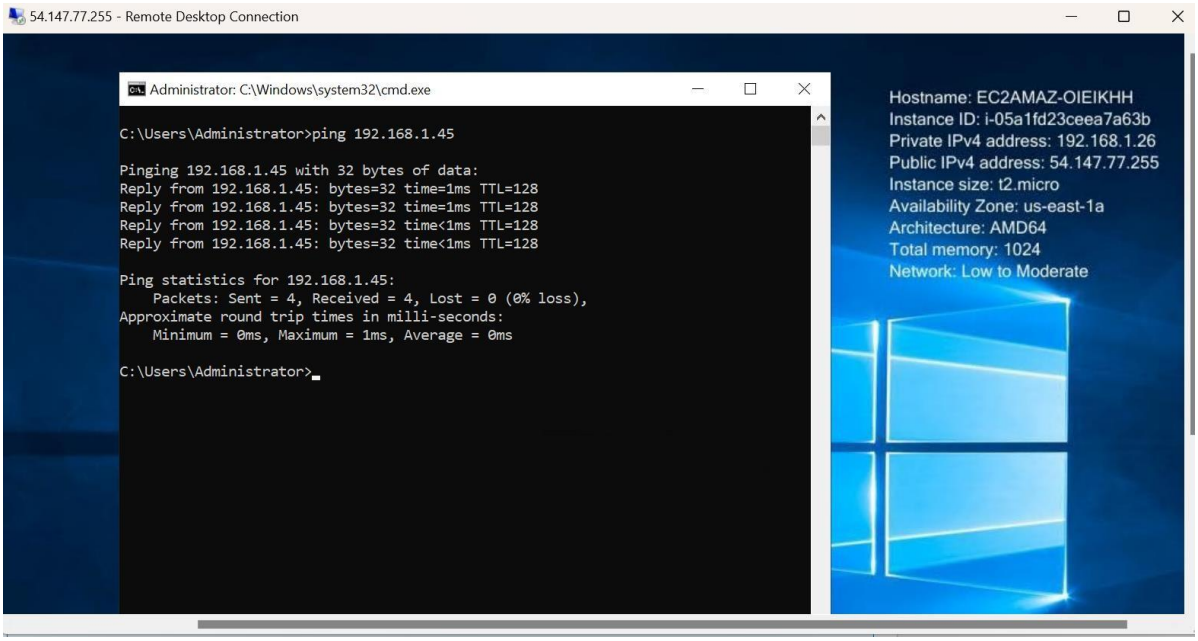
Go to Command Prompt of Windows2 instance



Ping Private IP address of Windows2 from Windows1



Ping Private IP address of Windows1 from Windows2



- Integration with CentOS

Go to your host/physical machine's cmd/powershell(run as administrator)

Here we used Powershell:

Go to CentOS instance, i.e putty App

To set strict read-only permissions on the centoskp.pem private key file.

```
[centos@ip-192-168-1-46 ~]$ chmod 400 /home/centos/centoskp.pem
```

```
[centos@ip-192-168-1-46 ~]$ cd /home/centos/
[centos@ip-192-168-1-46 ~]$ ls
centoskp.pem
[centos@ip-192-168-1-46 ~]$ cat centoskp.pem
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAlhIHkttHFBtYXVJpwS3NpCzn6ftgQXCsyM+uQwSuMw0SNcaD
ytStsf4pIvMrDN0Vd/fkeYiSEZOh7rU/XFK8MEaWE4e84T1jU0AXdXVAl1Nisp0V
5DrSBF4f64HmE24N1mnZRhiy9cGqhFtaSFXxIe53lEsq5ap1cUDu/Dg1aLMPEfoV
62SaCn3jYjYfID+TP0ydxs6IYZalwDJoUaputt5JTZGZE+9qfOBTqNFoMELrb4ZM
T3xBrfvHTJfDAT5j+EfV9dNugCPWc3HtuhOu8UsFeOUCB8xOvw+iEiXVBXpkxBdD
iwtqLzHUbZXzz7eGOxu6jYAr0V/mtom1+Aa1UQIDAQABAoIBAF9Ny+mDwDwQsaly
cR7xi6s7qXx628IzYEeXPL/q9QYOwwXLsF7begOTuWQssaZVsRCjKUHVbY4kB66l
BDwJfuHZseltQpJ4fd+c0EkT0ZVIq2FrST/L6xxC+lqCAC9OsRL93bIadWyFNVkg
+tD9Y0l9PAu9QNzhx/vIlP0wZRq7ww3FP++5/k0kItzBPqhZglWcvNaEh8OlpZFR
f3+bMqUgvR47YBhhiiLh6dfuHhH2/3NIciLxB1ezPKy1eiAhc96WrMzqLS+02rmH
rHvOylqyUqHlB90hhCP8vfnziSIRxHkPDUJXQRcFppnNzQHj7DBQCWohTGsj+0bW
BMlbOMECgYEAzhdQjdRJAVB+DSW3RVIRtC1FlMqCPp5809snYrJC6xHhcwecYGee
eWgtJVuVEDvxawo/cCZHA2MqnhBnjMRlWnq5TUqY4Fd4Lg8V3WBrdR4yixy6h/M+
tYNNz3+gGyNCSRv/+Q0cSnHrUn0zGGuBUWs5uV0YrTOzuGS74ZQCXqUCgYEAummy
OxM7sRbjLHdfBHVJF0uB0P4PoFdHhp07JDkDKsZBZK9bQzrCvgagWsxWgmtq6IdH
kJUmH2kJ9+fCooIPaTKydXS890DxEvXAe7LfxHTcHgWDp2lNyzGqCx6Od0vsPViZ
jXRJTK0Rxa0LM6vAzlh0rjAd8Fr+loXbcA+tCD0CgYBmLd8Ngrvzf+IPSKzWHLuU
VoHSJJlwzIldTKH5aCKPg90AeDyK+7TxwrnndMGyO9YVWQ5c302EG22s0rtUaqdJ
teEbtimXySZXx7jyenSsq50ppcPB1dM4kXJJt6A89D1QHCbMZzionFStIE/uApcq
M/x5r/OlJCYt/Ru92vFV1QKBgQC1bfPbDVAWfokiGRhwdVprAsFZUDyTOP9TZCpQ
ifCsLAMvZIB1gXZSjo55cNFt4/BMWoXrkU5mWcwq7PWBRwaKw53vNWxJVXqMkoUB
n8msInSt/33wbUwazL6MTfQo5csKtfXVN+2zY+qdM1tosexZoeUGN6yb52RLygIW
IhutnQKBgQCW747I5mHvJ0ICoiXnWlGnM9pSmuHpP24szQL7PQrmX36sBnVCZNet
v3VreKHXDkBv9yK8oNmEGFD9Dc0nwBlynS7LdOoJ+eeO/729BnUER4Za5DTUE6SQ
4EqoIaJoCF9yTYSM2b/3Z0livg7MllZtxI3+Er5sLCWHBYEVKZIgDw==
-----END RSA PRIVATE KEY-----[centos@ip-192-168-1-46 ~]$
```

To check appropriate permissions:

```
[centos@ip-192-168-1-46 ~]$ ls -l
total 4
-r--------. 1 centos centos 1678 Oct  8 18:09 centoskp.pem
[centos@ip-192-168-1-46 ~]$
```

To securely connect to a remote CentOS instance (located at the public IP 54.163.68.56) using SSH.

```
[centos@ip-192-168-1-46 ~]$ ssh -i /home/centos/centoskp.pem centos@54.163.68.56
[centos@ip-192-168-1-46 ~]$
```

To edit the DNS resolver configuration file on a CentOS system with superuser (sudo) privileges.

```
[centos@ip-192-168-1-46 ~]$ sudo vi /etc/resolv.conf
[centos@ip-192-168-1-46 ~]$ sudo cat /etc/resolv.conf
nameserver 8.8.8.8
nameserver 8.8.4.4
```

To test, ping to Google DNS server:

```
[centos@ip-192-168-1-46 ~]$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=117 time=0.649 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=117 time=0.657 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=117 time=0.965 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=117 time=0.709 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=117 time=0.597 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=117 time=0.583 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=117 time=1.23 ms
64 bytes from 8.8.8.8: icmp_seq=8 ttl=117 time=0.647 ms
64 bytes from 8.8.8.8: icmp_seq=9 ttl=117 time=0.971 ms
64 bytes from 8.8.8.8: icmp_seq=10 ttl=117 time=0.733 ms
^C
--- 8.8.8.8 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9004ms
rtt min/avg/max/mdev = 0.583/0.774/1.234/0.203 ms
```

To edit the CentOS-Base.repo:

Inside the file:

- Comment out the mirrorlist=... lines by adding # at the beginning.
- Replace them with the following base URL:

baseurl=http://vault.centos.org/7.9.2009/os/x86_64/

```
[centos@ip-192-168-1-46 ~]$ sudo vi /etc/yum.repos.d/CentOS-Base.repo
```

Clean and update the yum package

```
[centos@ip-192-168-1-46 ~]$ sudo yum clean all
Loaded plugins: fastestmirror
Cleaning repos: base extras updates
Cleaning up list of fastest mirrors
[centos@ip-192-168-1-46 ~]$ sudo yum update -y
Loaded plugins: fastestmirror
Determining fastest mirrors
base                                                                    | 3.6 kB  00:00:00
extras                                                                  | 3.6 kB  00:00:00
updates                                                                 | 3.6 kB  00:00:00
(1/6): base/primary_db                                                  | 6.1 MB  00:00:00
(2/6): extras/group_gz                                                  | 153 kB  00:00:00
(3/6): extras/primary_db                                                | 6.1 MB  00:00:00
(4/6): updates/group_gz                                                 | 153 kB  00:00:00
(5/6): base/group_gz                                                    | 153 kB  00:00:00
(6/6): updates/primary_db                                               | 6.1 MB  00:00:00
No packages marked for update
```

Install samba:

```
[centos@ip-192-168-1-46 ~]$ sudo yum install samba -y
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
Resolving Dependencies
--> Running transaction check
---> Package samba.x86_64 0:4.10.16-5.el7 will be installed
--> Processing Dependency: samba-libs = 4.10.16-5.el7 for package: samba-4.10.16-5.el7.x86_64
--> Processing Dependency: samba-common-tools = 4.10.16-5.el7 for package: samba-4.10.16-5.el7.x86_64
--> Processing Dependency: samba-common-libs = 4.10.16-5.el7 for package: samba-4.10.16-5.el7.x86_64
--> Processing Dependency: samba-common = 4.10.16-5.el7 for package: samba-4.10.16-5.el7.x86_64
--> Processing Dependency: samba-common = 4.10.16-5.el7 for package: samba-4.10.16-5.el7.x86_64
--> Processing Dependency: samba-client-libs = 4.10.16-5.el7 for package: samba-4.10.16-5.el7.x86_64
--> Processing Dependency: libwbclient = 4.10.16-5.el7 for package: samba-4.10.16-5.el7.x86_64
--> Processing Dependency: libxattr-tdb-samba4.so(SAMBA_4.10.16)(64bit) for package: samba-4.10.16-5.el7.x86_64
--> Processing Dependency: libutil-tdb-samba4.so(SAMBA_4.10.16)(64bit) for package: samba-4.10.16-5.el7.x86_64
--> Processing Dependency: libutil-reg-samba4.so(SAMBA_4.10.16)(64bit) for package: samba-4.10.16-5.el7.x86_64
--> Processing Dependency: libtevent.so.0(TEVENT_0.9.9)(64bit) for package: samba-4.10.16-5.el7.x86_64
--> Processing Dependency: libtevent.so.0(TEVENT_0.9.21)(64bit) for package: samba-4.10.16-5.el7.x86_64
--> Processing Dependency: libtevent.so.0(TEVENT_0.9.16)(64bit) for package: samba-4.10.16-5.el7.x86_64
--> Processing Dependency: libtevent-util.so.0(TEVENT_UTIL_0.0.1)(64bit) for package: samba-4.10.16-5.el7.x86_64
--> Processing Dependency: libtdb.so.1(TDB_1.2.5)(64bit) for package: samba-4.10.16-5.el7.x86_64
--> Processing Dependency: libtdb.so.1(TDB_1.2.1)(64bit) for package: samba-4.10.16-5.el7.x86_64
--> Processing Dependency: libtalloc.so.2(TALLOC_2.0.2)(64bit) for package: samba-4.10.16-5.el7.x86_64
--> Processing Dependency: libsys-rw-samba4.so(SAMBA_4.10.16)(64bit) for package: samba-4.10.16-5.el7.x86_64
```

Ensure that CentOS instance can ping Windows instances. From the CentOS terminal, run:

Windows1 pinged:

```
[centos@ip-192-168-1-46 ~]$ ping 192.168.1.45
PING 192.168.1.45 (192.168.1.45) 56(84) bytes of data.
64 bytes from 192.168.1.45: icmp_seq=1 ttl=128 time=1.15 ms
64 bytes from 192.168.1.45: icmp_seq=2 ttl=128 time=1.47 ms
64 bytes from 192.168.1.45: icmp_seq=3 ttl=128 time=1.14 ms
64 bytes from 192.168.1.45: icmp_seq=4 ttl=128 time=0.637 ms
64 bytes from 192.168.1.45: icmp_seq=5 ttl=128 time=1.06 ms
64 bytes from 192.168.1.45: icmp_seq=6 ttl=128 time=0.844 ms
64 bytes from 192.168.1.45: icmp_seq=7 ttl=128 time=0.706 ms
64 bytes from 192.168.1.45: icmp_seq=8 ttl=128 time=1.26 ms
64 bytes from 192.168.1.45: icmp_seq=9 ttl=128 time=0.472 ms
64 bytes from 192.168.1.45: icmp_seq=10 ttl=128 time=0.981 ms
64 bytes from 192.168.1.45: icmp_seq=11 ttl=128 time=1.13 ms
64 bytes from 192.168.1.45: icmp_seq=12 ttl=128 time=1.63 ms
64 bytes from 192.168.1.45: icmp_seq=13 ttl=128 time=0.889 ms
64 bytes from 192.168.1.45: icmp_seq=14 ttl=128 time=1.80 ms
64 bytes from 192.168.1.45: icmp_seq=15 ttl=128 time=1.12 ms
^C
--- 192.168.1.45 ping statistics ---
15 packets transmitted, 15 received, 0% packet loss, time 14016ms
rtt min/avg/max/mdev = 0.472/1.089/1.807/0.351 ms
```

Windows2 pinged:

```
[centos@ip-192-168-1-46 ~]$ ping 192.168.1.26
PING 192.168.1.26 (192.168.1.26) 56(84) bytes of data.
64 bytes from 192.168.1.26: icmp_seq=1 ttl=128 time=1.26 ms
64 bytes from 192.168.1.26: icmp_seq=2 ttl=128 time=0.640 ms
64 bytes from 192.168.1.26: icmp_seq=3 ttl=128 time=1.60 ms
64 bytes from 192.168.1.26: icmp_seq=4 ttl=128 time=0.878 ms
64 bytes from 192.168.1.26: icmp_seq=5 ttl=128 time=1.32 ms
64 bytes from 192.168.1.26: icmp_seq=6 ttl=128 time=1.31 ms
64 bytes from 192.168.1.26: icmp_seq=7 ttl=128 time=1.46 ms
64 bytes from 192.168.1.26: icmp_seq=8 ttl=128 time=0.861 ms
64 bytes from 192.168.1.26: icmp_seq=9 ttl=128 time=1.34 ms
64 bytes from 192.168.1.26: icmp_seq=10 ttl=128 time=0.657 ms
64 bytes from 192.168.1.26: icmp_seq=11 ttl=128 time=1.94 ms
64 bytes from 192.168.1.26: icmp_seq=12 ttl=128 time=1.28 ms
64 bytes from 192.168.1.26: icmp_seq=13 ttl=128 time=1.15 ms
64 bytes from 192.168.1.26: icmp_seq=14 ttl=128 time=1.89 ms
^C
--- 192.168.1.26 ping statistics ---
14 packets transmitted, 14 received, 0% packet loss, time 13017ms
rtt min/avg/max/mdev = 0.640/1.259/1.948/0.390 ms
[centos@ip-192-168-1-46 ~]$
```

SAMBA installation and sharing:

Login using root

Add samba service and reload it

Check its status

```
[root@ip-192-168-1-46 centos]# sudo firewall-cmd --permanent --zone=public --add-service=samba
Warning: ALREADY_ENABLED: samba
success
[root@ip-192-168-1-46 centos]# sudo firewall-cmd --reload
success
[root@ip-192-168-1-46 centos]# systemctl status firewalld.service
● firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2024-10-09 04:31:13 UTC; 59min ago
     Docs: man:firewalld(1)
 Main PID: 570 (firewalld)
   CGroup: /system.slice/firewalld.service
           └─570 /usr/bin/python2 -Es /usr/sbin/firewalld --nofork --nopid

Oct 09 04:31:12 ip-192-168-1-46.ec2.internal systemd[1]: Starting firewalld - dynamic firewall daemon..
Oct 09 04:31:13 ip-192-168-1-46.ec2.internal systemd[1]: Started firewalld - dynamic firewall daemon.
```

Stop the firewall service, should display ‘inactive(dead)’

```
[root@ip-192-168-1-46 centos]# systemctl stop firewalld
[root@ip-192-168-1-46 centos]# systemctl status firewalld.service
● firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled; vendor preset: enabled)
   Active: inactive (dead) since Wed 2024-10-09 05:31:38 UTC; 4s ago
     Docs: man:firewalld(1)
  Process: 570 ExecStart=/usr/sbin/firewalld --nofork --nopid $FIREWALLD_ARGS (code=exited, status=0/SUCCESS)
 Main PID: 570 (code=exited, status=0/SUCCESS)

Oct 09 04:31:12 ip-192-168-1-46.ec2.internal systemd[1]: Starting firewalld - dynamic firewall daemon...
```

Create a directory apps inside samba directroy

```
[root@ip-192-168-1-46 centos]# mkdir -p /samba/apps
[root@ip-192-168-1-46 centos]# cd  /samba/apps/
[root@ip-192-168-1-46 apps]# ls -ltr
total 0
```

The chcon -t samba_share_t /samba/apps command changes the SELinux context of the /samba/apps directory to allow Samba access, enabling it to share files securely between Linux and Windows systems when SELinux is enforced.

Then, edit the smb.conf file by adding the below lines at the bottom:

[global]

workgroup = SAMBA

netbios name = centos

security = user

map to guest = bad user

dns proxy = no

[Apps]

path = /samba/apps

browsable = yes

writable = yes

guest ok = yes

guest only = yes

read only = no

```
[root@ip-192-168-1-46 apps]# chcon -t samba_share_t  /samba/apps
[root@ip-192-168-1-46 apps]# vi /etc/samba/smb.conf
```

Move to the root directroy, and apply the permissions

```
[root@ip-192-168-1-46 apps]# cd ..
[root@ip-192-168-1-46 samba]# cd ..
[root@ip-192-168-1-46 /]# chmod a+rwx /samba
[root@ip-192-168-1-46 /]# chmod a+rwx /samba/apps/
[root@ip-192-168-1-46 /]# chmod a+rwx /samba/apps/*
```

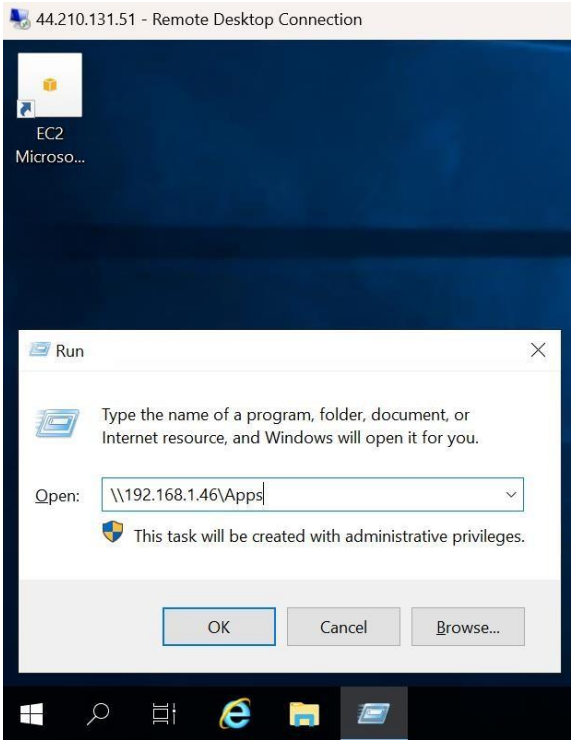
Start and enable the services

```
[root@ip-192-168-1-46 /]# systemctl start smb nmb
[root@ip-192-168-1-46 /]# systemctl enable smb nmb
Created symlink from /etc/systemd/system/multi-user.target.wants/smb.service to /usr/lib/systemd/system/smb.service.
Created symlink from /etc/systemd/system/multi-user.target.wants/nmb.service to /usr/lib/systemd/system/nmb.service.
[root@ip-192-168-1-46 /]# systemctl status smb nmb
● smb.service - Samba SMB Daemon
   Loaded: loaded (/usr/lib/systemd/system/smb.service; enabled; vendor preset: disabled)
   Active: active (running) since Wed 2024-10-09 05:26:29 UTC; 13min ago
     Docs: man:smbd(8)
           man:samba(7)
           man:smb.conf(5)
 Main PID: 4422 (smbd)
   Status: "smbd: ready to serve connections..."
   CGroup: /system.slice/smb.service
           ├─4422 /usr/sbin/smbd --foreground --no-process-group
           ├─4425 /usr/sbin/smbd --foreground --no-process-group
           ├─4426 /usr/sbin/smbd --foreground --no-process-group
           ├─4427 /usr/sbin/smbd --foreground --no-process-group
           ├─4430 /usr/sbin/smbd --foreground --no-process-group
           └─4440 /usr/sbin/smbd --foreground --no-process-group

Oct 09 05:26:29 ip-192-168-1-46.ec2.internal systemd[1]: Stopped Samba SMB Daemon.
Oct 09 05:26:29 ip-192-168-1-46.ec2.internal systemd[1]: Starting Samba SMB Daemon...
Oct 09 05:26:29 ip-192-168-1-46.ec2.internal smbd[4422]: [2024/10/09 05:26:29.136786,  0] ../../lib/util/become_daemon.c:136(daemon_ready)
Oct 09 05:26:29 ip-192-168-1-46.ec2.internal systemd[1]: Started Samba SMB Daemon.
Oct 09 05:26:29 ip-192-168-1-46.ec2.internal smbd[4422]:   daemon_ready: daemon 'smbd' finished starting up and ready to serve connections

● nmb.service - Samba NMB Daemon
   Loaded: loaded (/usr/lib/systemd/system/nmb.service; enabled; vendor preset: disabled)
   Active: active (running) since Wed 2024-10-09 05:26:29 UTC; 13min ago
     Docs: man:nmbd(8)
```

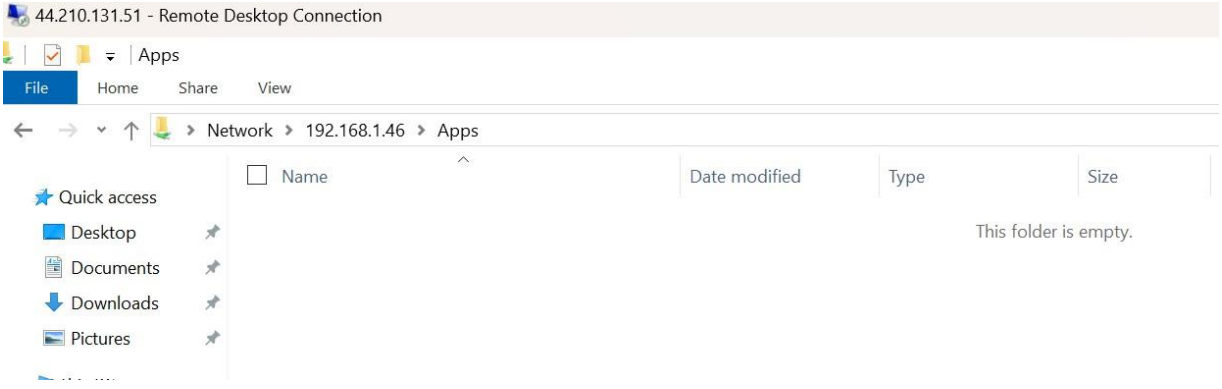
Move to apps directory, initially it is empty

```
[root@ip-192-168-1-46 /]# cd /samba/apps/
[root@ip-192-168-1-46 apps]# ls
[root@ip-192-168-1-46 apps]#
```

To check it in windows instances, go to the windows instance, press win+R



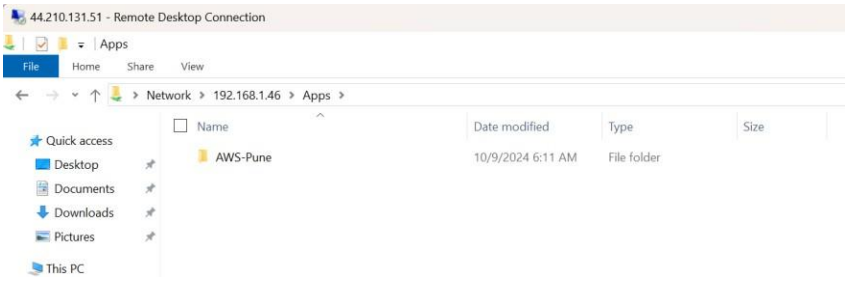
Even, in windows instances, it shows empty



Create a directory in CentOS instance

```
[root@ip-192-168-1-46 apps]# mkdir AWS-Pune
[root@ip-192-168-1-46 apps]# ls
AWS-Pune
[root@ip-192-168-1-46 apps]#
```

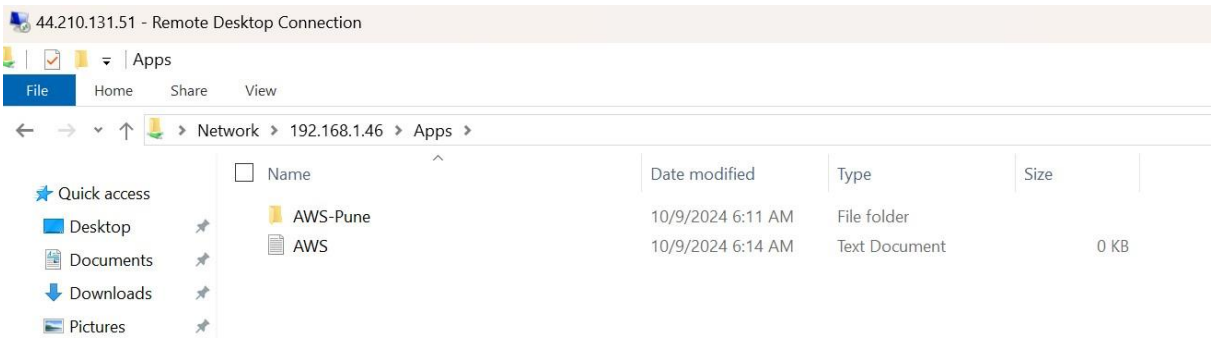
Display the directory in windows instance, which we have created in CentOS instance



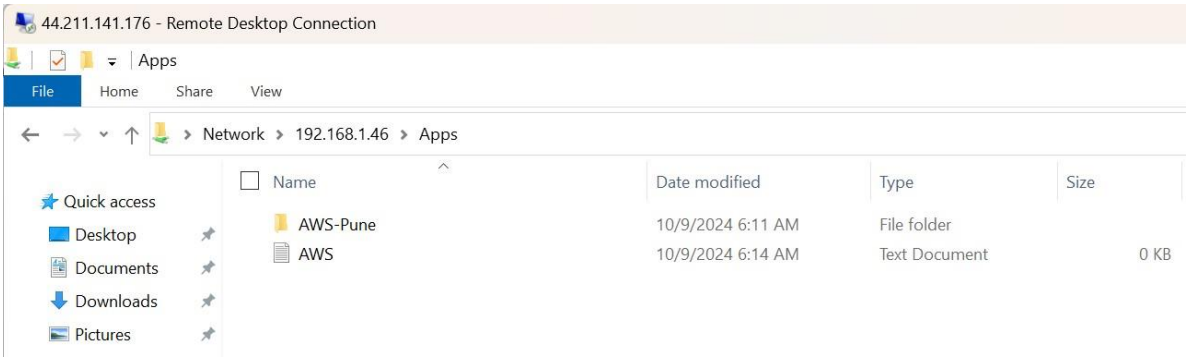
Create a file in CentOS instance

```
[root@ip-192-168-1-46 apps]# touch AWS.txt
[root@ip-192-168-1-46 apps]# ls
AWS-Pune  AWS.txt
[root@ip-192-168-1-46 apps]#
```

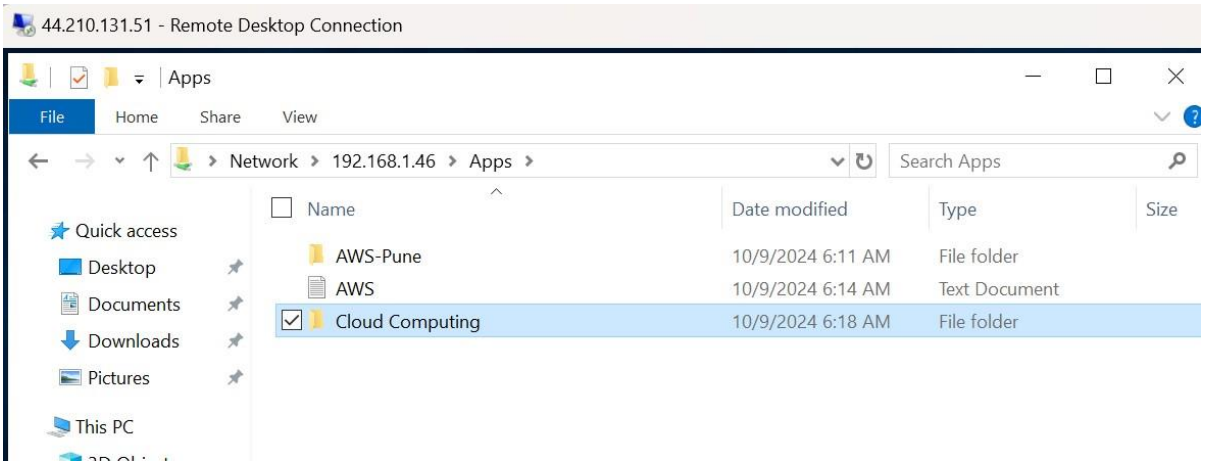
Displays the file in 1st windows instance too, which was created in CentOS instance



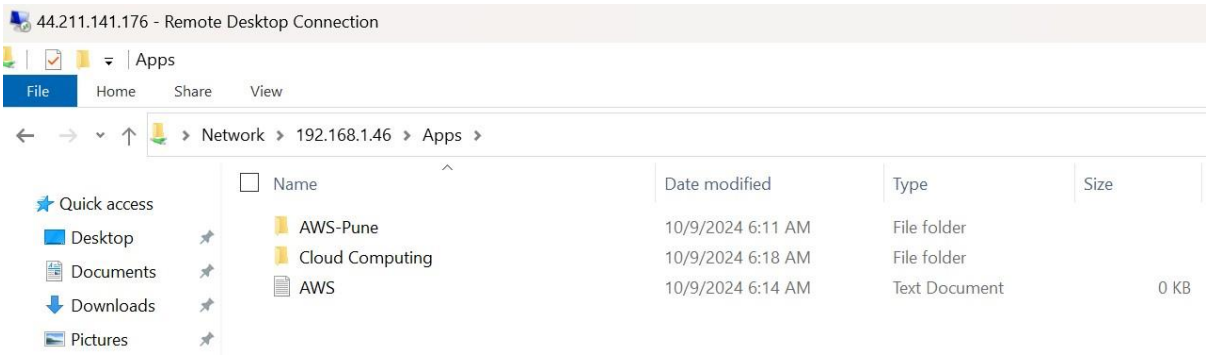
Also displays in 2nd windows instance



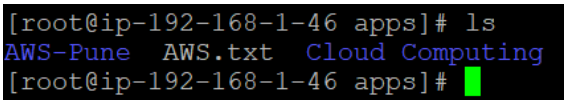
Create a directory/folder in 1st windows instance



It is reflected in 2nd windows instance



As well as in CentOS instance



Use of S3 Browser in Windows 10 VM to access AWS

S3 Browser is an Amazon S3 Windows client to access files and manage S3 storage settings.

Advantages of Installing an S3 Browser On-Premises:

1) Enhanced Security: Local control over data access and storage enhances data security and privacy.

2) Faster Data Transfer: Improved speed for large file uploads and downloads without relying on external networks.

3) Cost Savings: Reduces the cost of cloud storage by allowing better data management and the option to move data off-cloud.

4) Offline Access: Ability to access and manage S3 buckets and data even when the internet is unavailable.

5) Customization: Tailor the browser environment to meet specific needs without depending on third-party hosting services.

1. Prerequisites

- **VirtualBox**: Installed on your host machine with a running Windows 10 virtual machine.
- **AWS IAM Account**: With programmatic access enabled (IAM user with access keys).
- **Access to S3 Buckets**: Necessary permissions for S3 operations (list, read, write).

2. VirtualBox Configuration

1. **Install VirtualBox**:

   Download and install VirtualBox from the official website

   Create a new virtual machine with Windows 10 installed.

2. **Create Windows 10 VM**:

   Open VirtualBox and click *New*.

   Name your VM, select type as *Microsoft Windows*, and choose *Windows 10* (64-bit).

   Allocate memory (at least 4GB recommended).

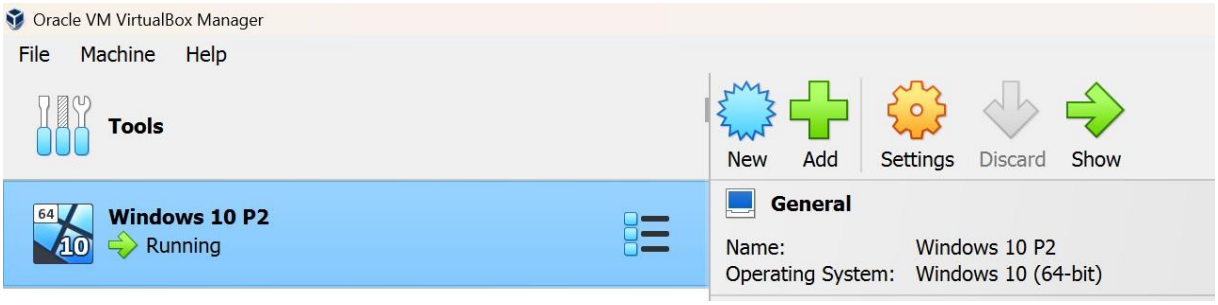
   Create a new virtual hard disk (25GB or more).

3. **Install Windows 10**:

   Insert the Windows 10 ISO file or physical disk and start the VM.

   Complete the installation steps for Windows 10.

Select the Machine, and click on Start



3. S3 Browser Installation in Windows 10 VM

1. Download S3 Browser:

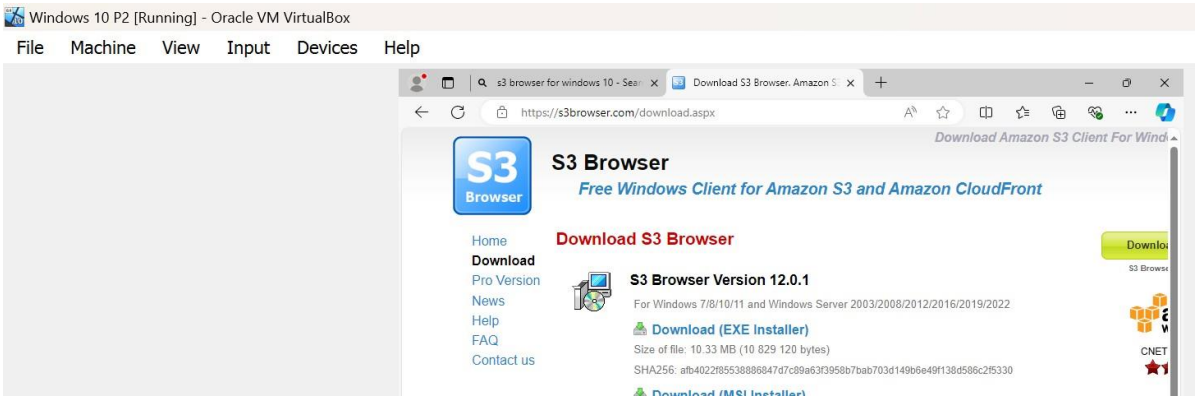
Inside the Windows 10 VM, open a web browser and go to S3 Browser.

Download the latest version of S3 Browser.

2. Install S3 Browser:

Run the downloaded file and follow the installation wizard.

Accept the license agreement, choose the installation directory, and complete the installation.



4. Setting Up AWS IAM Access Keys for S3 in AWS

1. Create an IAM User:

Go to the AWS Management Console and sign in.

Navigate to IAM (Identity and Access Management).

Click on Users -> Create User

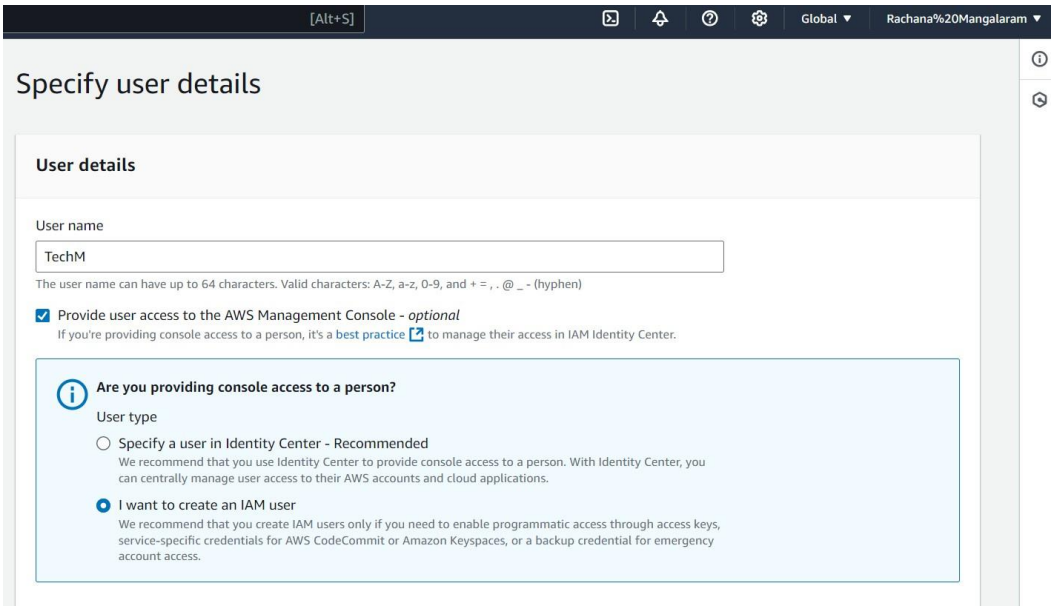
Enter a username-> Check the Provide user access to AWS Management Console – *optional*

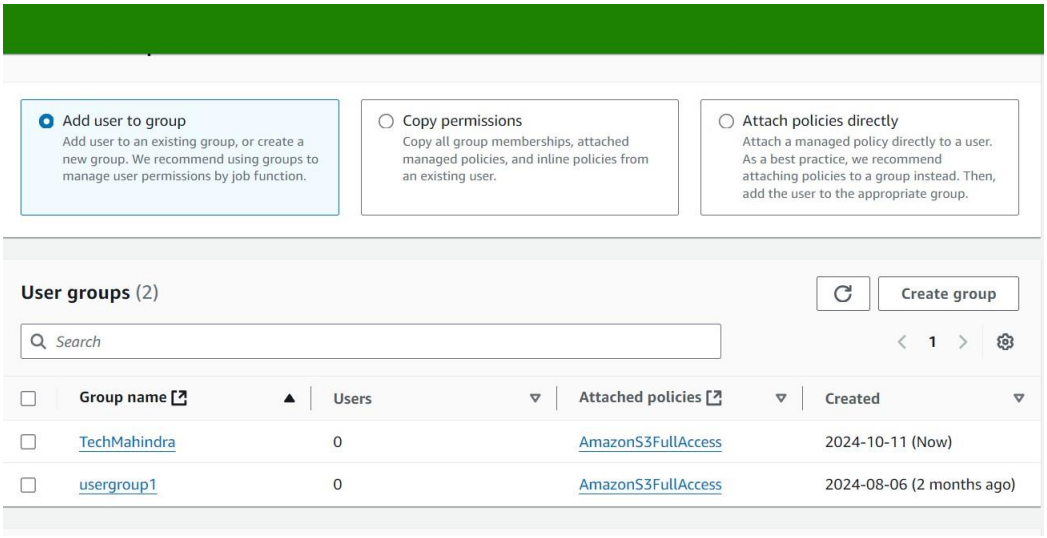
Select I want to create IAM user->Next

In Permission Options, Select Attach Policies directly

Search for the desired permission, i.e AmazonS3FullAccess -> Next-> Create User-> Return to user list

Select the user which has been created->Under security credentials->Access Keys->Create Access Key-> Select CLI->Check the checkbox->Next->Create Access Key-> Copy Paste the Access key and Secret Access Key or can download the .csv file-> Done.





5. Configuring S3 Browser with IAM Access Keys from Windows 10

1. Open S3 Browser:

Launch the S3 Browser from the Windows 10 start menu.

2. Add New Account:

In the S3 Browser, click on Accounts in the top menu, then select Add New Account.

3. Enter AWS Credentials:

Input your AWS IAM Access Key ID and Secret Access Key.

Optionally, specify the default region (e.g., us-east-1 or another region where your S3 buckets are hosted).

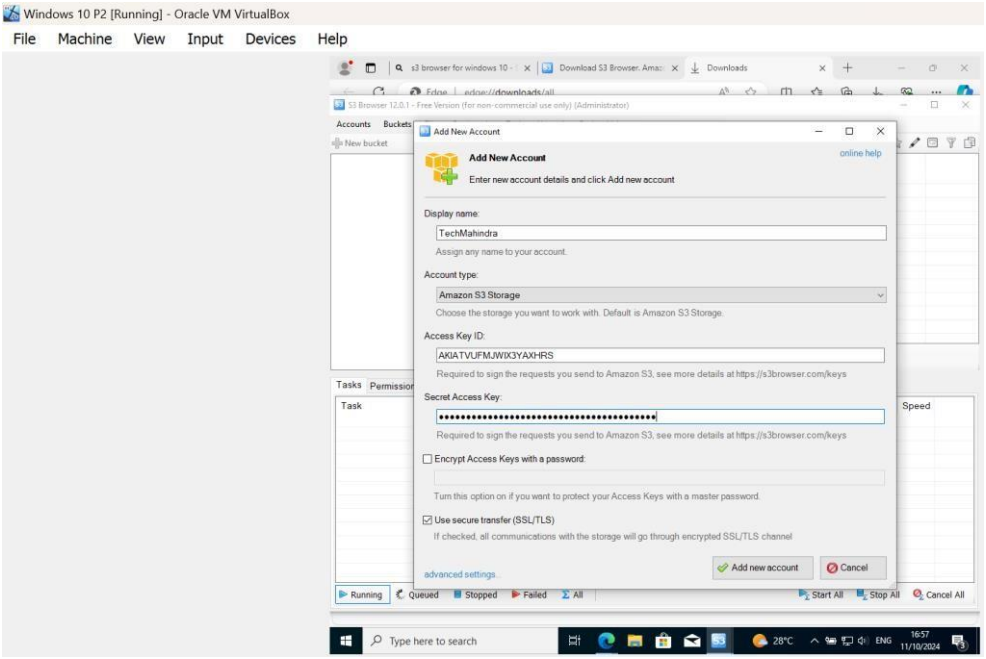
Click Add New Account to save.

4. Access S3 Buckets:

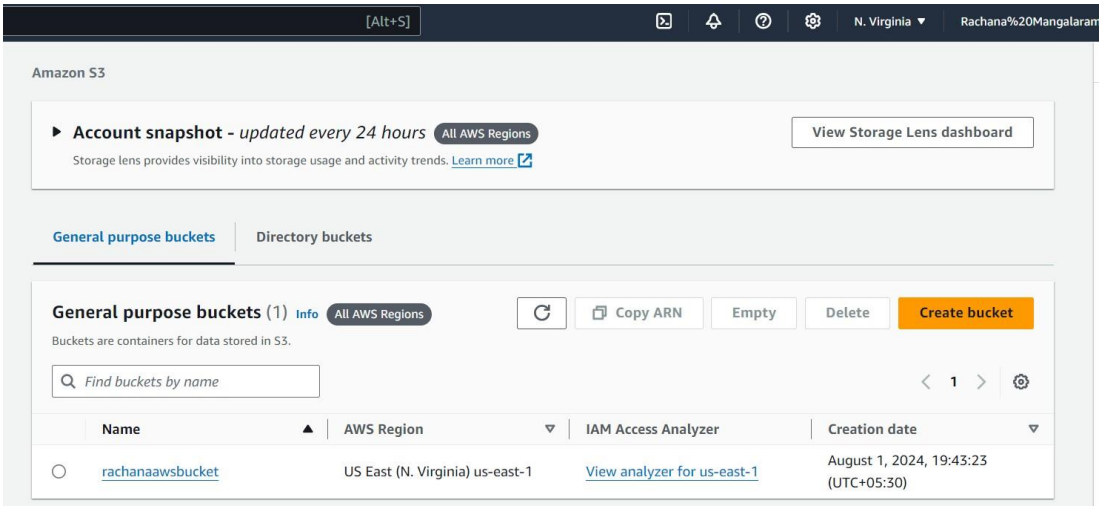
Once the account is added, you will see a list of your S3 buckets.

You can now upload, download, or manage files in your S3 buckets

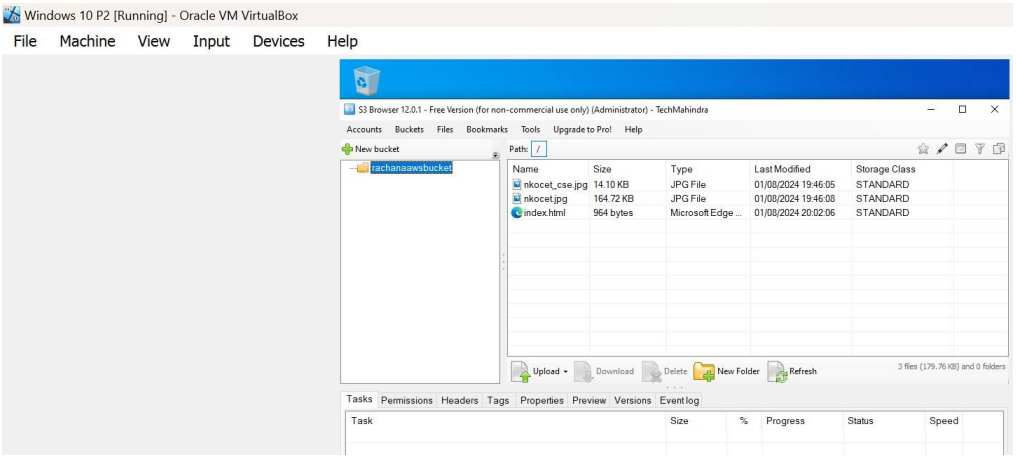
Accessing Our Aws Account By Using Access Keys



Initially there one bucket already exist, in AWS



Also, visible in Windows 10 VM



6. Testing and Validation

1. Create a Test Bucket:

In S3 Browser, right-click in the bucket list and select Create New Bucket.

Provide a unique bucket name and select the desired region.

2. Upload a File:

Open your new bucket, click Upload, and choose a file from your VM to upload to the bucket.

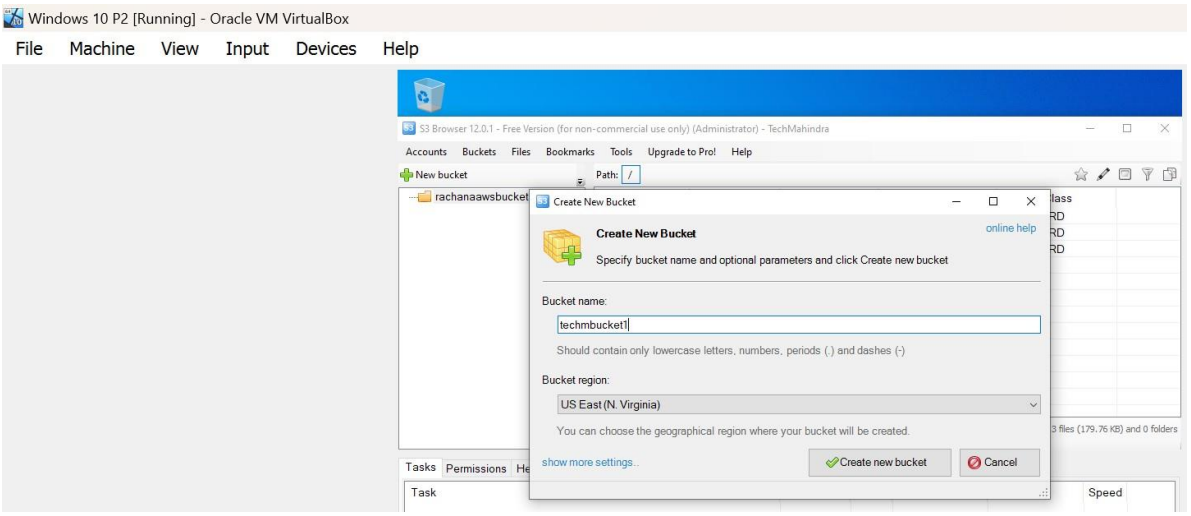
3. Download a File:

Select a file from any of your S3 buckets, right-click, and choose Download to retrieve it to your VM.

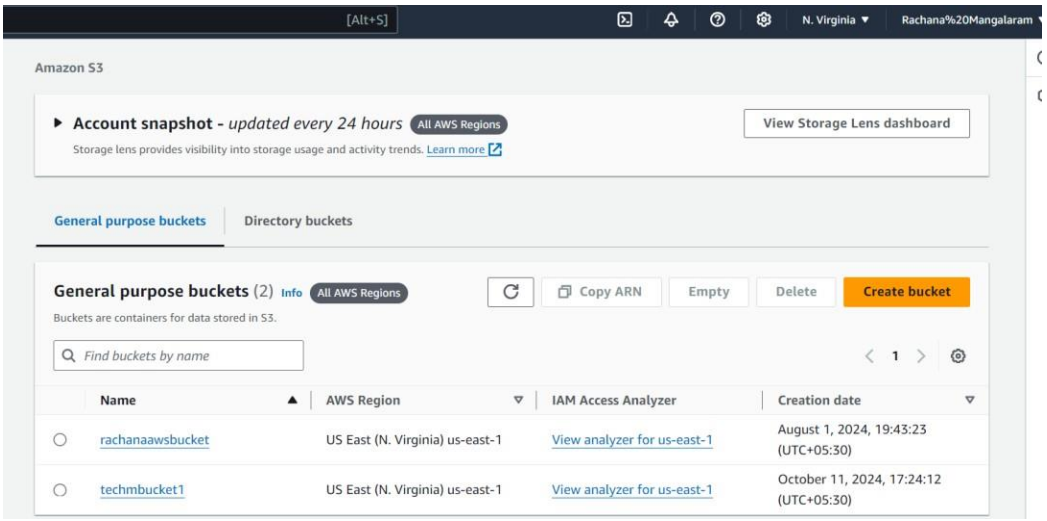
4. Validate Permissions:

Ensure you can perform all required S3 operations such as listing, uploading, and downloading without any access issues.

Create a new Bucket in Windows 10 ie from S3Browser

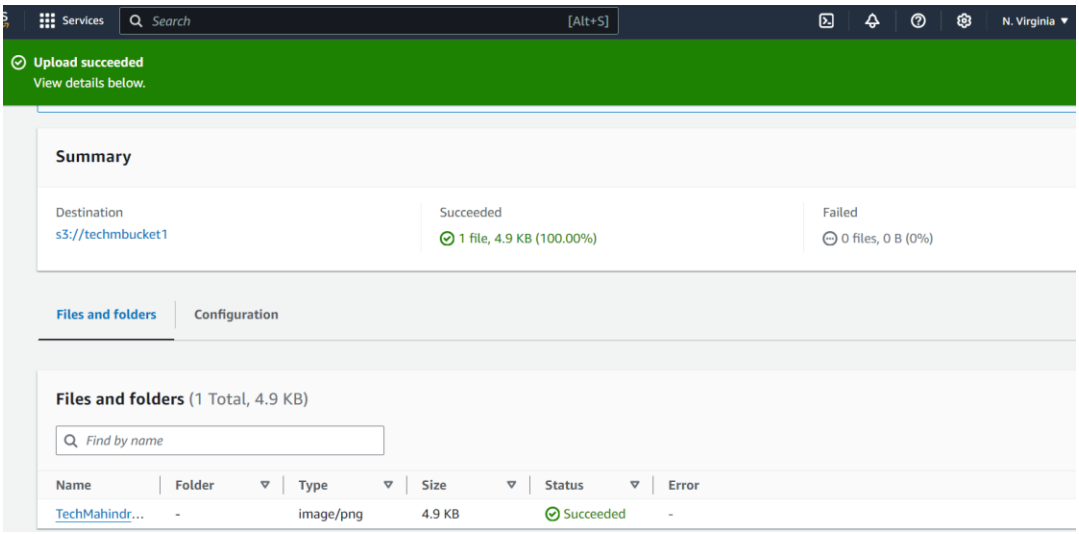


It reflected in AWS

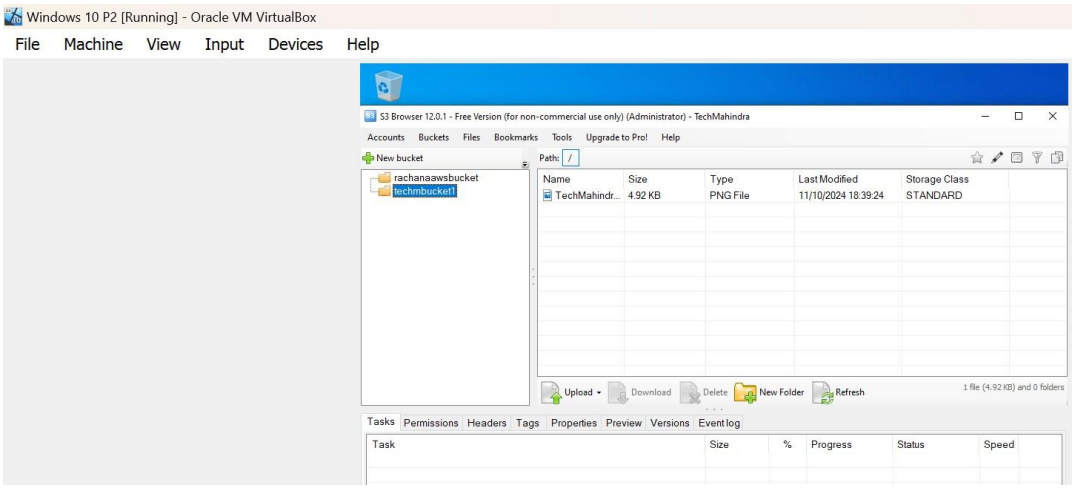


Uploading Images From Aws Account

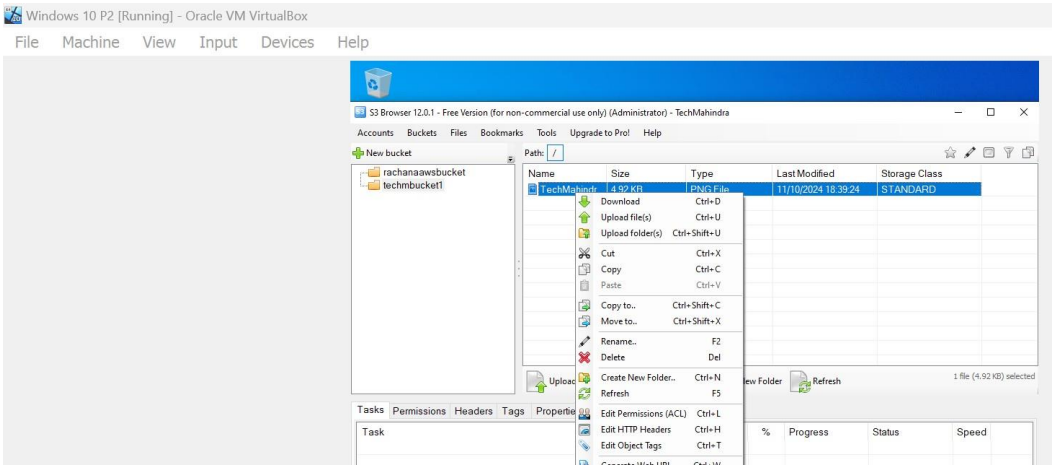
Select the bucket-> Click on Upload-> Add files-> Select the Image->Upload

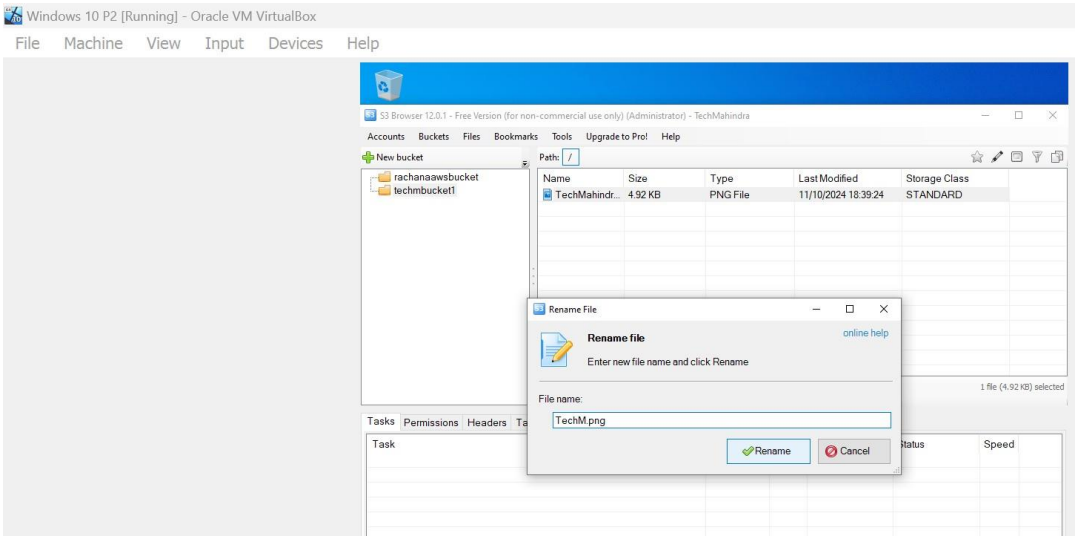


Accessing the image from S3 Browser in Windows 10

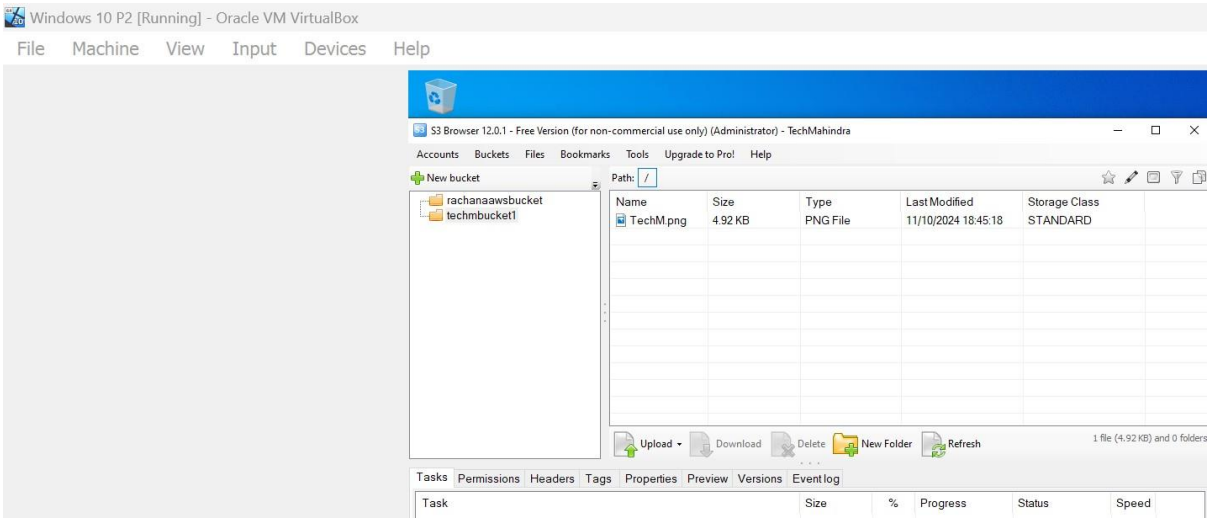


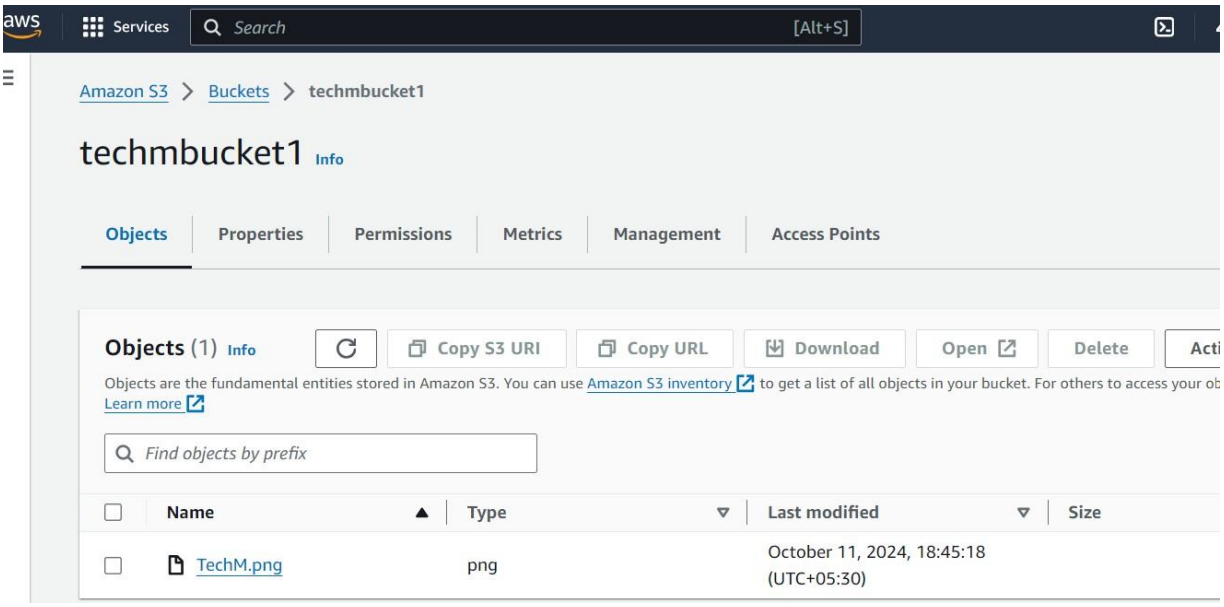
Renaming image from S3 Browser





Accessing Renamed file in S3 Browser and AWS S3 too





In summary, With S3 Browser, you can perform a wide range of operations to manage your Amazon S3 buckets and objects efficiently. You can upload and download files directly to and from your S3 buckets, making data transfers seamless. Additionally, you have the ability to create, delete, and configure S3 buckets as needed, allowing you to organize and manage your cloud storage with ease. Furthermore, S3 Browser provides tools to set and adjust permissions on your buckets and objects, enabling precise control over access and security for your stored data.

Additional Services Used:

CloudWatch

Amazon CloudWatch is a monitoring and observability service provided by Amazon Web Services (AWS) that enables users to collect and track metrics, monitor log files, set alarms, and automatically react to changes in AWS resources. It helps users gain insights into the operational health, performance, and resource utilization of their AWS infrastructure and applications. Amazon Cloud Watch is a monitoring service offered by Amazon Web Services to monitor applications like the following.

- Performance.

- Health of the application.

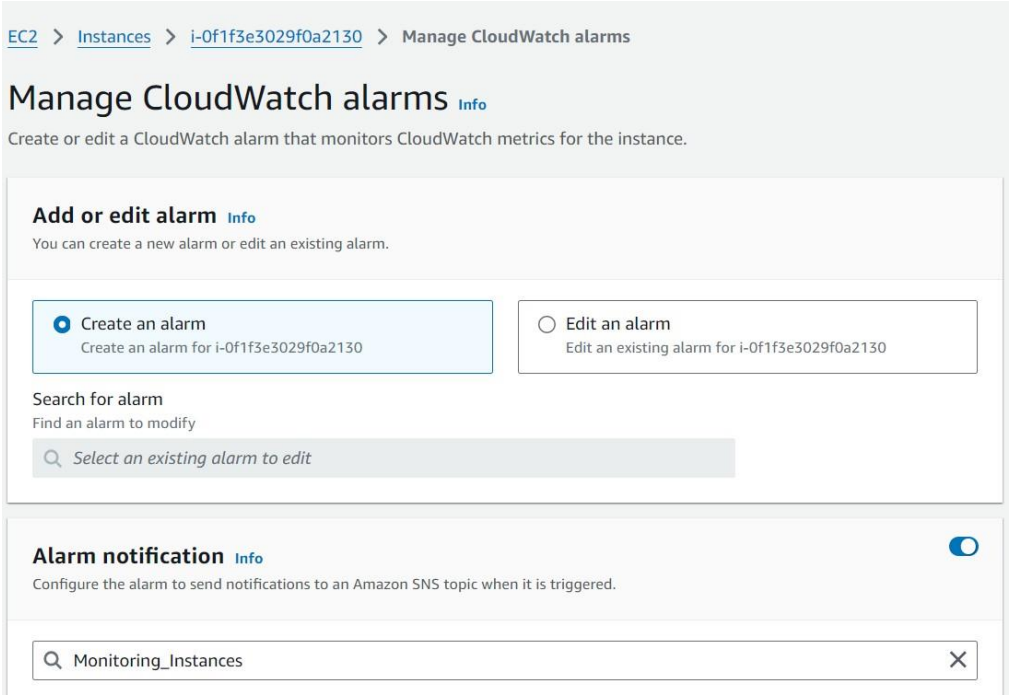
- Monitors the resource use, etc.

We can set the alarm to the to the resource use of the applications when the limits are exceeded

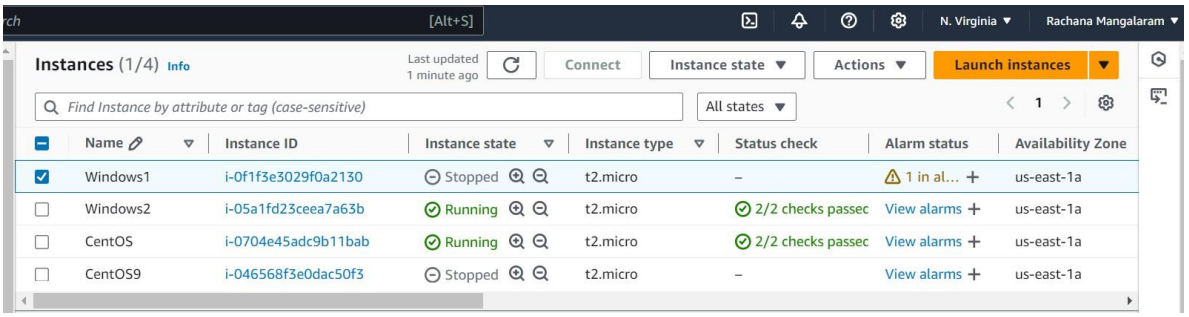
Go to SNS-> Create topic->Name the topic->Create topic

Create subscription-> Select the topic and Protocol ->Create Subscription

Now, Select the instance-> View Alarm + -> Select Create Alarm-> Select the Alarm notification(Topic which we have created)-> Select ‘Stop’ in Alarm Action-> Create.



Now. Check that the Instance has stopped, as it has reached the specified value



III. Troubleshooting Common Issues

EC2 Instances

1. Connectivity Issue

- Unable to Connect via SSH/RDP

For SSH (Linux): Port 22 is open.

- If it is Linux-based servers we need to check SSH port number 22 to be added in Security Group Inbound rules

- SSH allows you to remotely access Linux-based servers (like EC2 instances on AWS)

For RDP (Windows): Port 3389 is open.

If you want to access Remote Desktop

When your EC2 Instance not able to connect Remote Desktop,

- We need to check whether the RDP port is provided in Inbound Rules of Security of Groups or Not

- If not present, add RDP port number 3389 then save rules.

2. Instance Current Status

- Running or not, If not, right click on the instance, go to Instance State -> Start instance

3. EC2 Instance Status checks – 2/2

i. 1/2 – N/w, Antivirus, EBS, NW Interface

- If you face this issue, simply stop and start the instance

ii. 0/2 – We can't troubleshoot anything, need to check with AWS | Backend H/w issue

4. Need to check instance screenshot from EC2 console

- Go to Actions

- Monitor and troubleshoot

- Get instance screenshot

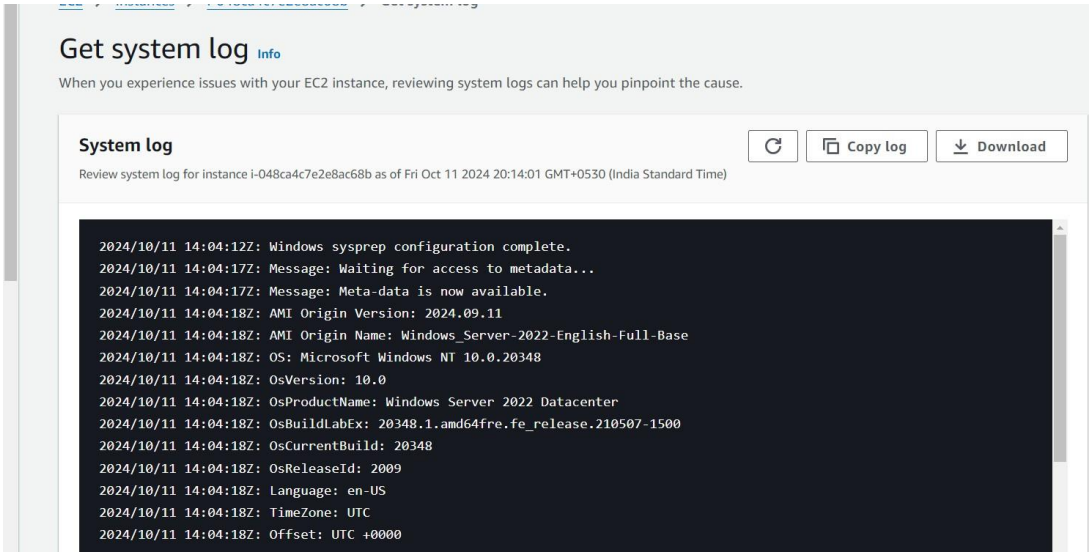
Check if there is any BSOD issues, Driver issues, Corrupted issues etc..

- If it shows any BSOD, We can Restore from latest backup (Snapshot)

5. System logs from EC2 console

- Instance Boot Issues

- Check for any warnings and error messages



6. Cloud Trail

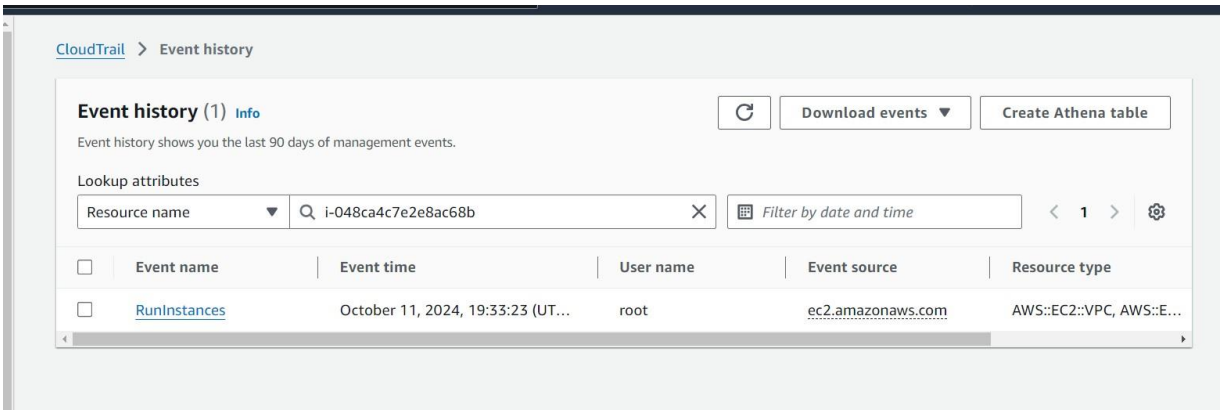
- Check if any recent changes to that instance

Copy Instance ID

Go to Cloud Trail

Event History

Search with Resource name followed by Instance ID



7. Subnet Mask Issues:

Verify that all instances have the correct subnet mask (255.255.255.192). Mismatched subnet masks prevent communication between instances.

8. Security Group Misconfiguration:

Check that the security groups allow necessary inbound and outbound traffic, especially for SSH, RDP, and internal communication between instances.

9. Incorrect IP Addressing:

Ensure that the correct private and public IP addresses are assigned to instances. Verify that each instance can communicate within the VPC by pinging internal IP addresses.

10. Performance Issues

a. High CPU/Memory Usage:

Monitor with CloudWatch: Use Amazon CloudWatch to monitor CPU, memory, and network metrics. Identify any spikes in usage.

Check Running Processes: For Linux, SSH into the instance and use commands like top, htop, or free -m to see which processes are consuming resources.

For Windows

11. Check Disk Usage:

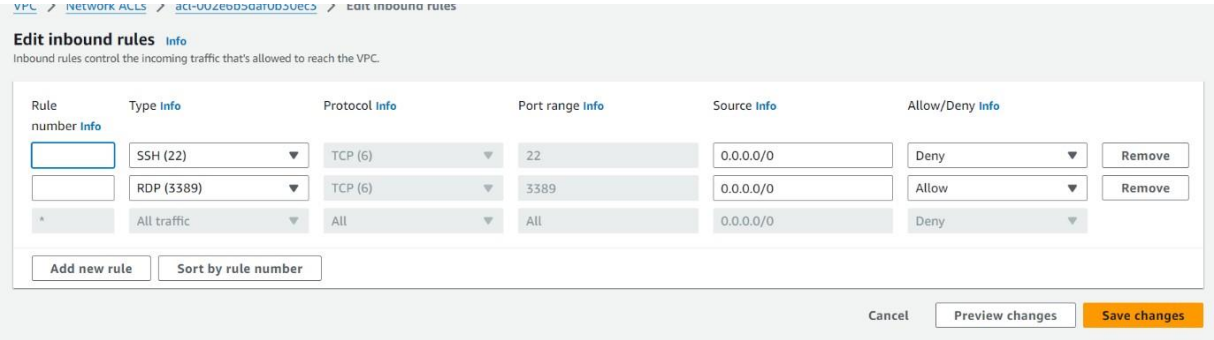
Use df -h (Linux) or Disk Management (Windows) to check if your volumes are running out of space.

- Network Perspective Troubleshooting:

1. Any rule in place on ACL

- Check Network ACL If there any port denying the incoming traffic

- Make it allow



2. Check our EC2 is located in Public or Private Subnet

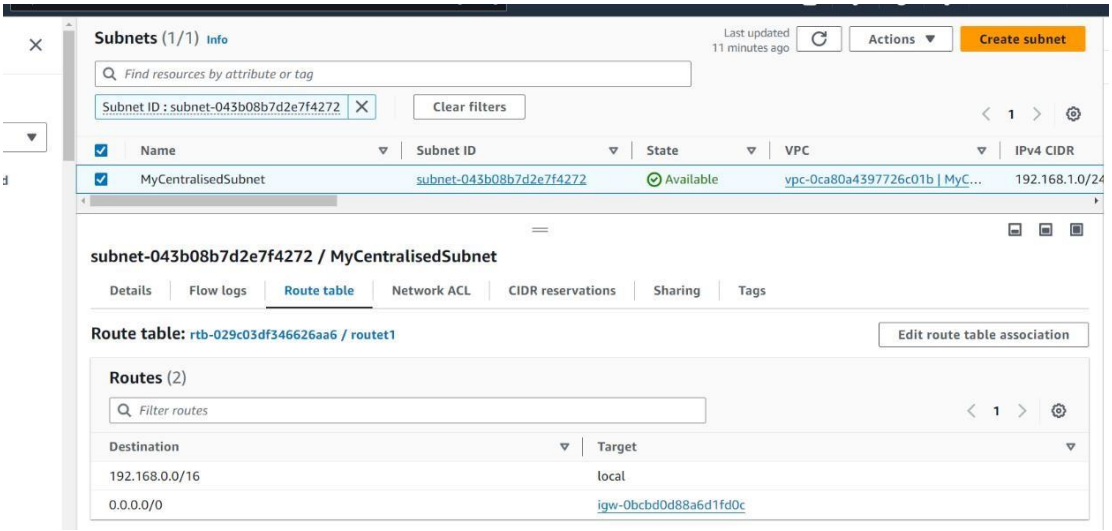
- Go to networking

- Click on Subnet ID

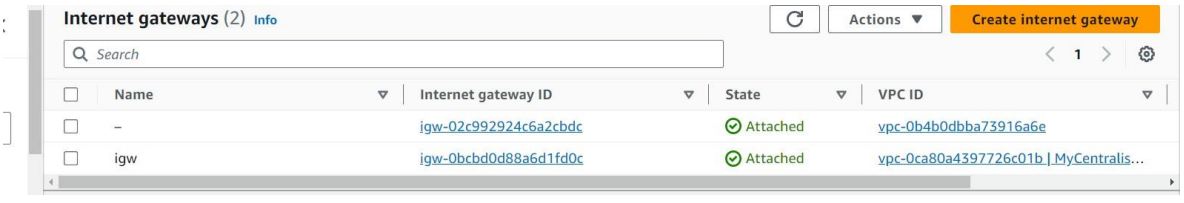
- Go to Route table

- Check for Internet Gateway

- If Route has Internet connectivity it is definitely going to be a public subnet.



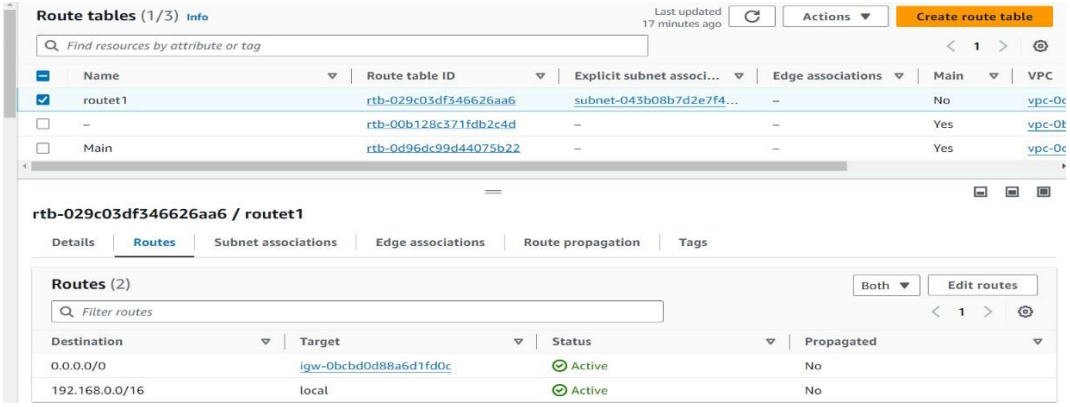
3. VPC – Has IGW attached or not



4. Check VPC has proper Route entries

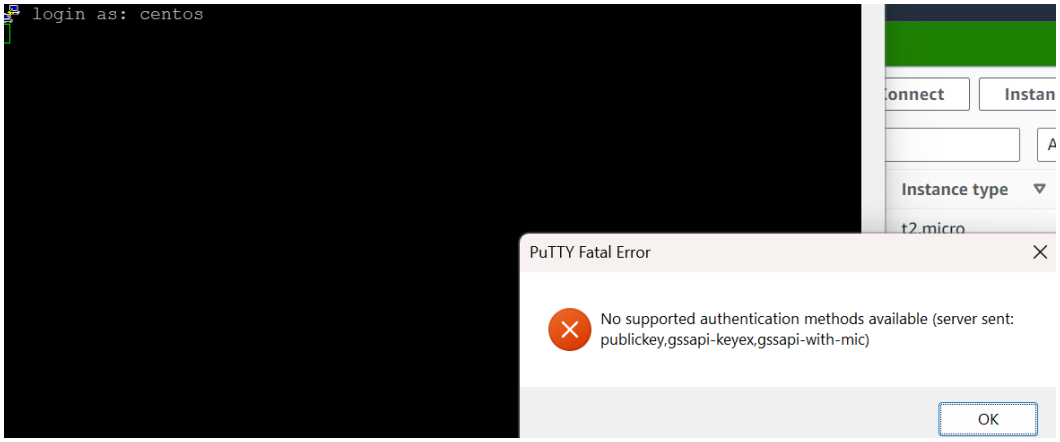
- Check whether it has Internet Gateway entry or not

- If Route table don't have IGW entry, even though our VPC has Internet Gateway you won't be able to connect with EC2 instances but there may be a chances will come.



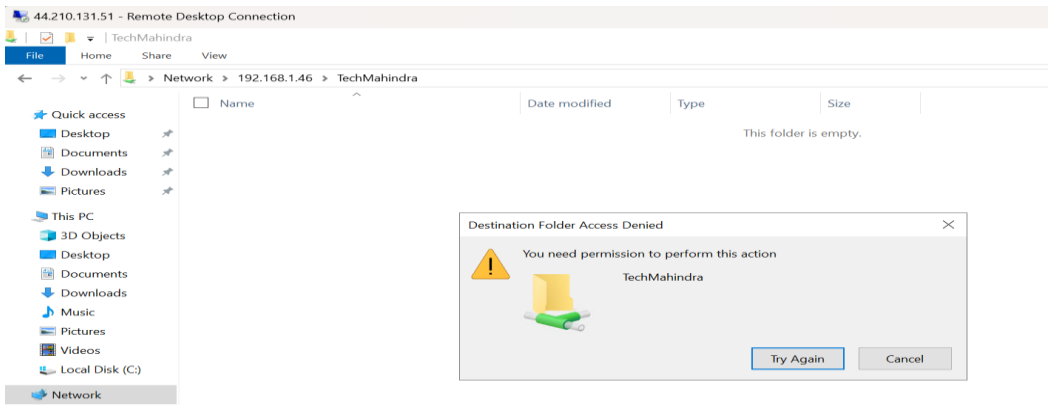
5. Putty App SSH Issue:

Convert the .pem file into .ppk file using PuTTYgen and upload the .ppk file for CentOS instance.



6. Files and Directroy Issue:

Give proper permissions to the files and directories using chmod command



Conclusion:

The project successfully implements a centralized server using CentOS on AWS, managing two client machines running Windows Server 2019 and Windows Server 2022. By utilizing EC2 instances within a custom VPC, the setup ensures secure communication through proper subnetting and security groups. The CentOS server functions as the administrative hub, handling essential network configurations, managing shared services like file sharing via Samba, and enabling remote management through RDP access.

This infrastructure design offers a robust solution for centralized administration, combining AWS services with traditional system administration tools to streamline network connectivity, security, and resource sharing.