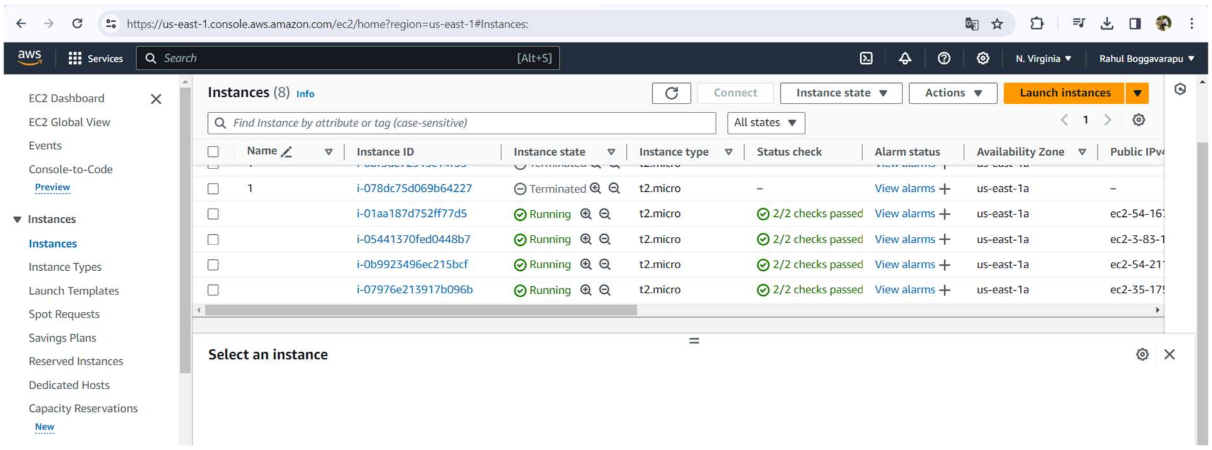
Name: B V N Rahul
Reg no: RA2111028010192

Lab Activity-2

Multiple Hosts with Same PEM Keys:

1.Create an EC2 instance with multiple instances with same key pair.



https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#Instances:

Instances (8)

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4
☐	1	i-078dc75d069b64227	Terminated	t2.micro	–	View alarms	us-east-1a	–
☐		i-01aa187d752ff77d5	Running	t2.micro	2/2 checks passed	View alarms	us-east-1a	ec2-54-16
☐		i-05441370fed0448b7	Running	t2.micro	2/2 checks passed	View alarms	us-east-1a	ec2-3-83-1
☐		i-0b9923496ec215bcf	Running	t2.micro	2/2 checks passed	View alarms	us-east-1a	ec2-54-21
☐		i-07976e213917b096b	Running	t2.micro	2/2 checks passed	View alarms	us-east-1a	ec2-35-17

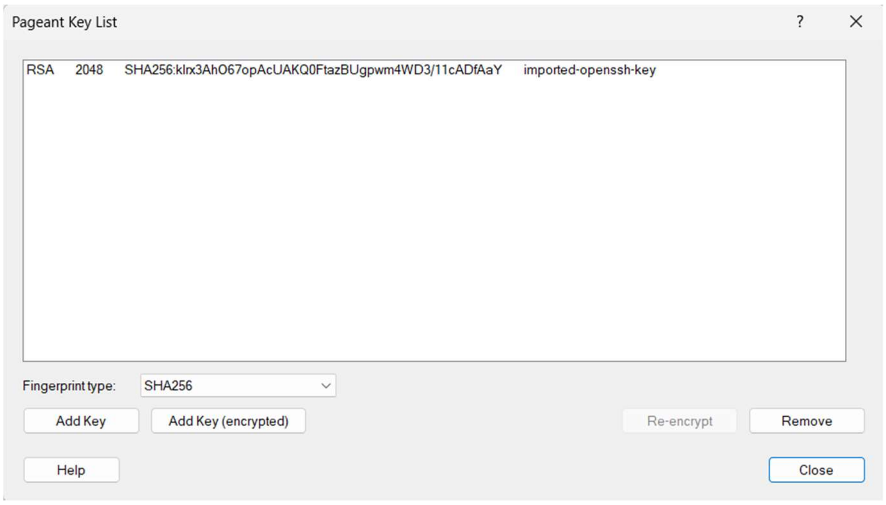
Select an instance

2.Download the putty gen in your windows and configure it.

3. Start Pageant from the PuTTY folder: Start Menu > All Programs > PuTTY > Pageant.

4. Pageant will start minimized in the system tray. Right-click on its icon and select Add Key.

5. In the file explorer, select your .ppk file(s) and click Open to load the keys into Pageant.



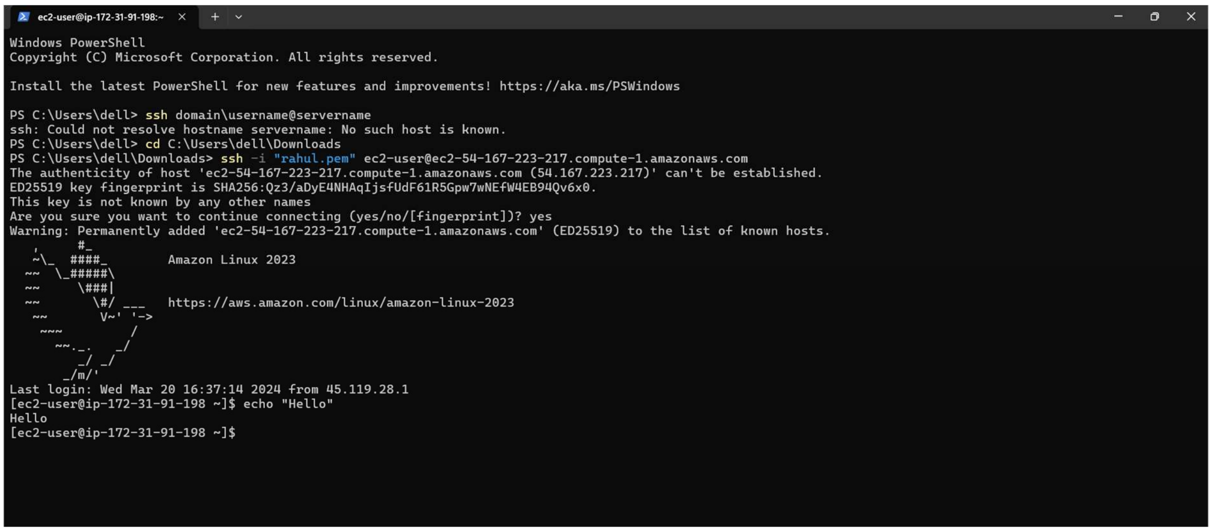
Pageant Key List

```
RSA    2048    SHA256:klrx3AhO67opAcUAKQ0FtazBUgpwm4WD3/11cADfAaY    imported-openssh-key
```

Fingerprint type: SHA256

5. When connecting via PuTTY, simply enter your hostname or IP address, and SSH user. PuTTY will automatically try to authenticate using the keys loaded in Pageant.

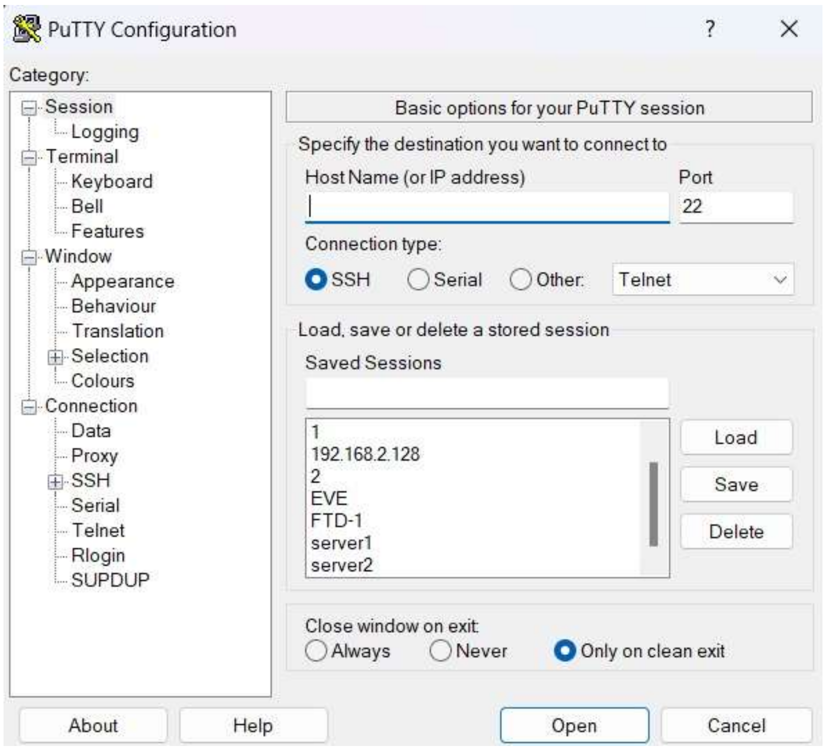
6. Now you type it in command prompt.



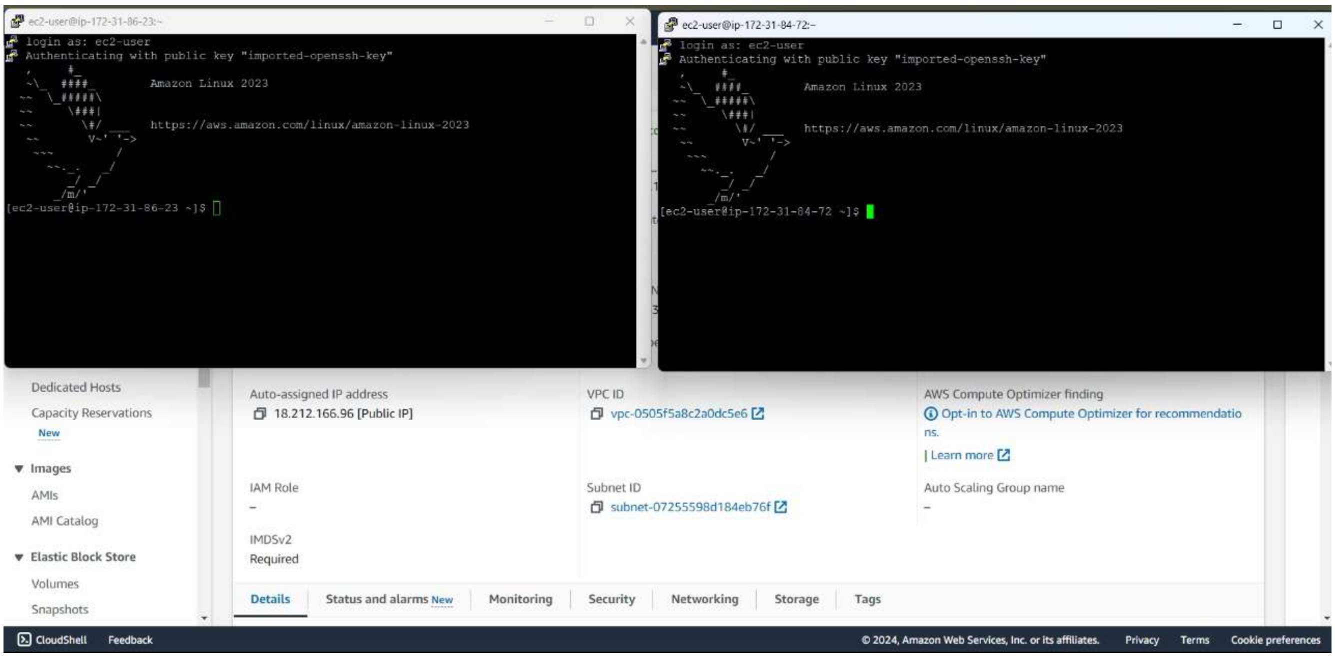
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\dell> ssh domain\username@servername
ssh: Could not resolve hostname servername: No such host is known.
PS C:\Users\dell> cd C:\Users\dell\Downloads
PS C:\Users\dell\Downloads> ssh -i "rahul.pem" ec2-user@ec2-54-167-223-217.compute-1.amazonaws.com
The authenticity of host 'ec2-54-167-223-217.compute-1.amazonaws.com (54.167.223.217)' can't be established.
ED25519 key fingerprint is SHA256:Qz3/aDyE4NHAqIjsfUdF61R5Gpw7wNEfW4EB94Qv6x0.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-54-167-223-217.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Wed Mar 20 16:37:14 2024 from 45.119.28.1
[ec2-user@ip-172-31-91-198 ~]$ echo "Hello"
Hello
[ec2-user@ip-172-31-91-198 ~]$
```

Multiple Hosts with Different PEM Keys:

1. **Create multiple sessions** in PuTTY for each host/key combination.
2. For each session, navigate to Connection > SSH > Auth and specify the correct private key file in the Private Key File for Authentication section.
3. Save the session with a unique name for easy access



Host Name (or IP address):
Port: 22
Connection type: [✓ SSH] Serial Other: Telnet
Saved Sessions:
1
192.168.2.128
2
EVE
FTD-1
server1
server2
Close window on exit: Always Never [✓ Only on clean exit]



ec2-user@ip-172-31-86-23:~

```
login as: ec2-user
Authenticating with public key "imported-openssh-key"
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-172-31-86-23 ~]$
```

ec2-user@ip-172-31-84-72:~

```
login as: ec2-user
Authenticating with public key "imported-openssh-key"
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-172-31-84-72 ~]$
```

Auto-assigned IP address: 18.212.166.96 [Public IP]

VPC ID: vpc-0505f5a8c2a0dc5e6

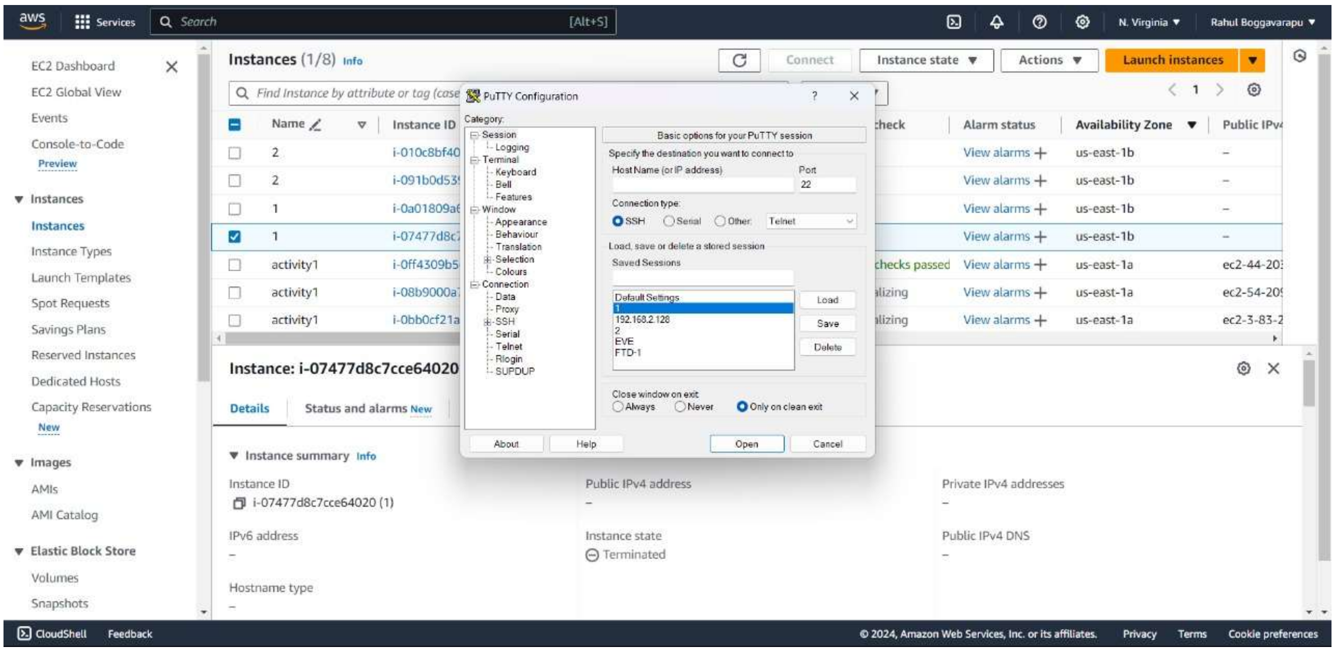
AWS Compute Optimizer finding: Opt-in to AWS Compute Optimizer for recommendations. | Learn more

IAM Role: –

Subnet ID: subnet-07255598d184eb76f

Auto Scaling Group name: –

IMDSv2: Required



Instances (1/8) Info

Name	Instance ID	Status check	Alarm status	Availability Zone	Public IPv4
2	i-010c8bf40		View alarms +	us-east-1b	–
2	i-091b0d53		View alarms +	us-east-1b	–
1	i-0a01809a		View alarms +	us-east-1b	–
1	i-07477d8c		View alarms +	us-east-1b	–
activity1	i-0ff4309b5	checks passed	View alarms +	us-east-1a	ec2-44-20
activity1	i-08b9000a	lizing	View alarms +	us-east-1a	ec2-54-20
activity1	i-0bb0cf21a	lizing	View alarms +	us-east-1a	ec2-3-83-2

Instance: i-07477d8c7cce64020

Instance summary Info

Instance ID: i-07477d8c7cce64020 (1)

Public IPv4 address: –

Private IPv4 addresses: –

IPv6 address: –

Instance state: Terminated

Public IPv4 DNS: –

Hostname type: –

PuTTY Configuration

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address):
Port: 22

Connection type: [✓ SSH] Serial Other: Telnet

Load, save or delete a stored session

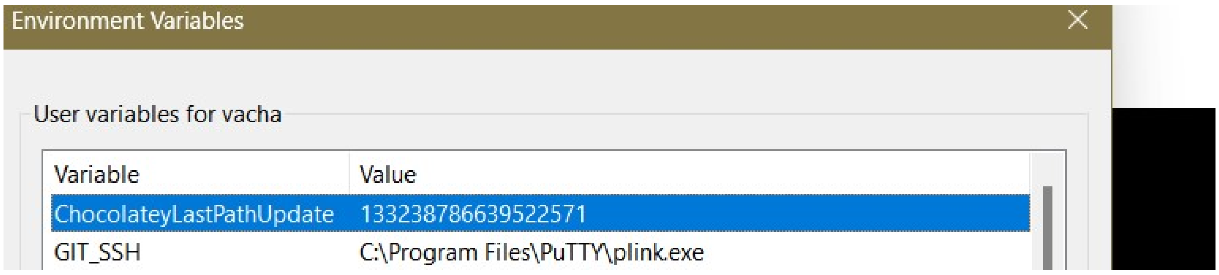
Saved Sessions

Default Settings
1
192.168.2.128
2
EVE
FTD-1

Close window on exit: Always Never [✓ Only on clean exit]

Environment Setup for Local and Remote Hosts

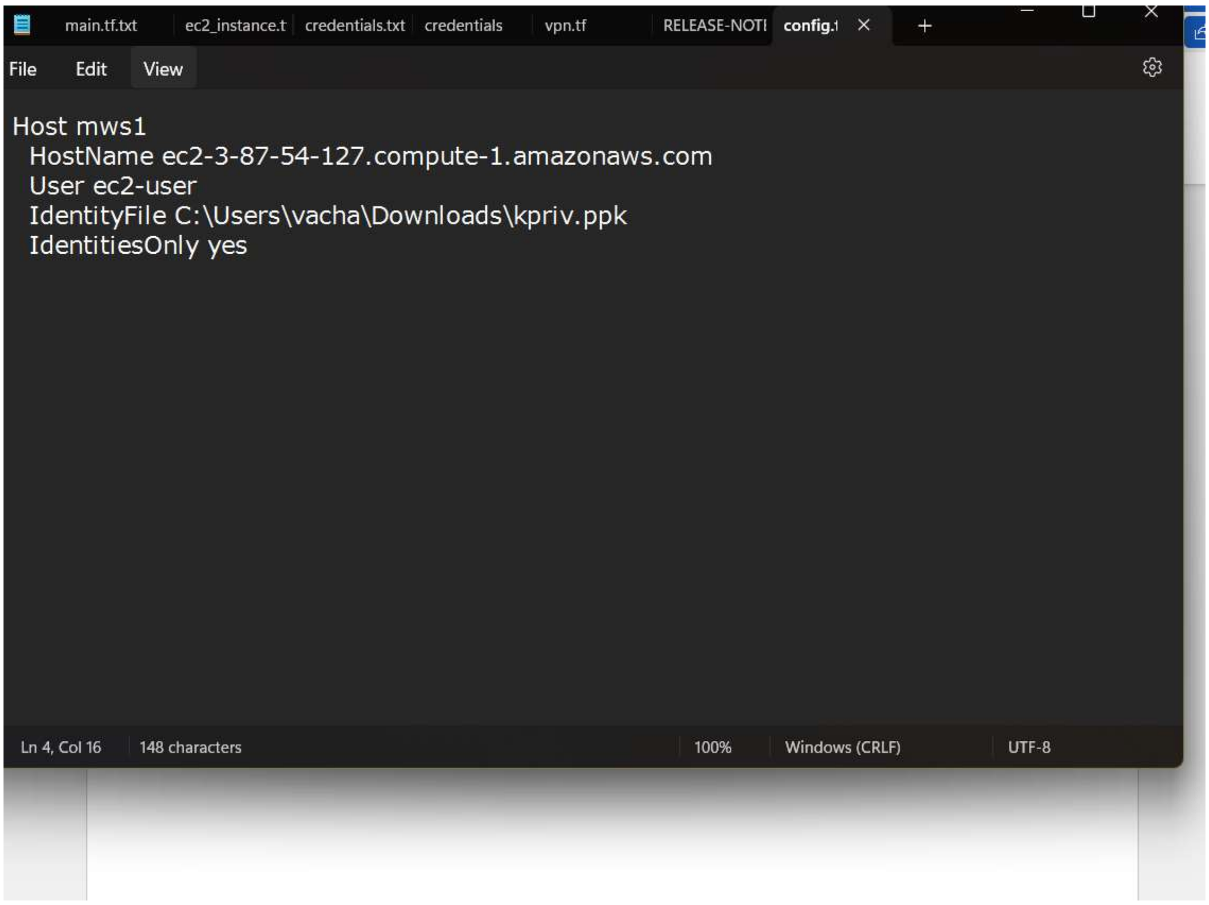
1.Set up an environment variable named GIT_SSHwith the path to your plink.exe.



User variables for vacha

Variable	Value
ChocolateyLastPathUpdate	133238786639522571
GIT_SSH	C:\Program Files\PuTTY\plink.exe

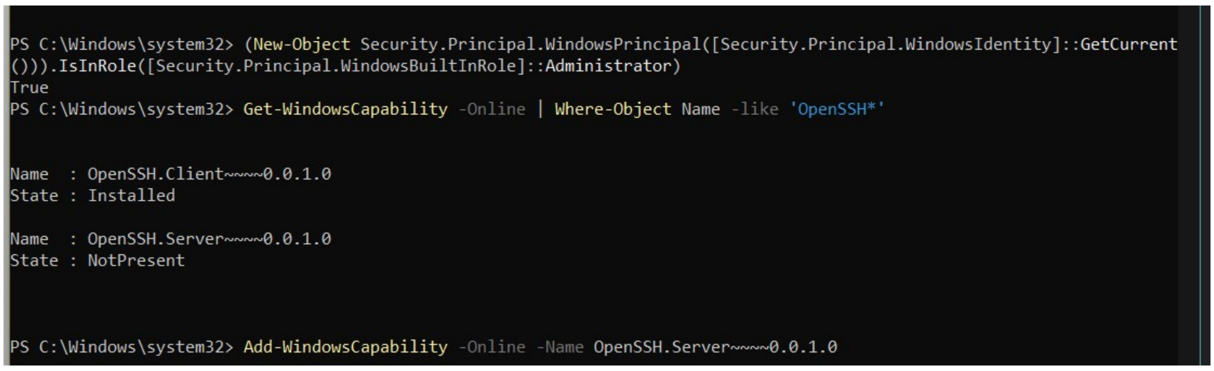
2.Use the ~/.ssh/configfile to configure host shortcuts and keys. Add Identity Agent SSH_AUTH_SOCKto your .ssh/config.



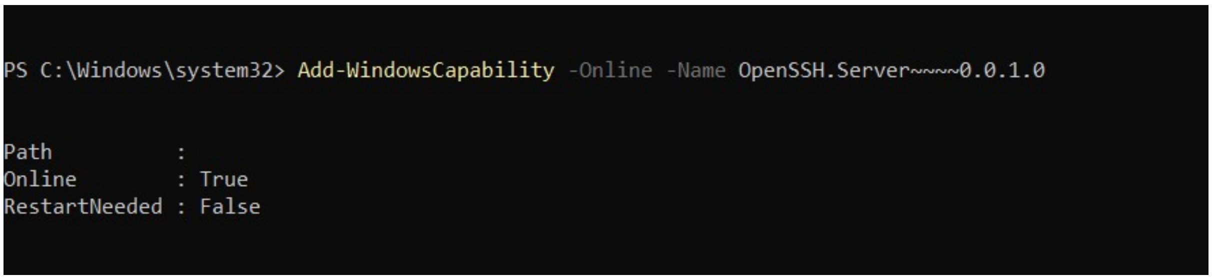
```
Host mws1
  HostName ec2-3-87-54-127.compute-1.amazonaws.com
  User ec2-user
  IdentityFile C:\Users\vacha\Downloads\kpriv.ppk
  IdentitiesOnly yes
```

1. Install OpenSSH:
   In windows powershell with admin access , type:

```
# Install the OpenSSH Client
Add-WindowsCapability -Online -Name OpenSSH.Client~~~~0.0.1.0

# Install the OpenSSH Server
Add-WindowsCapability -Online -Name OpenSSH.Server~~~~0.0.1.0
```



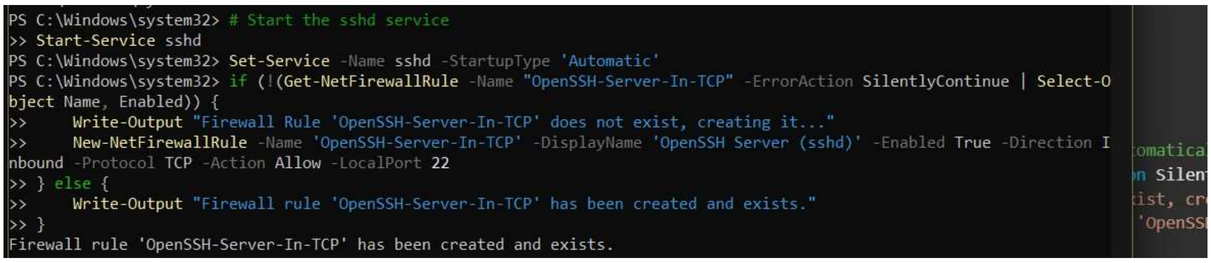
```
PS C:\Windows\system32> (New-Object Security.Principal.WindowsPrincipal([Security.Principal.WindowsIdentity]::GetCurrent
())).IsInRole([Security.Principal.WindowsBuiltInRole]::Administrator)
True
PS C:\Windows\system32> Get-WindowsCapability -Online | Where-Object Name -like 'OpenSSH*'


Name  : OpenSSH.Client~~~~0.0.1.0
State : Installed

Name  : OpenSSH.Server~~~~0.0.1.0
State : NotPresent



PS C:\Windows\system32> Add-WindowsCapability -Online -Name OpenSSH.Server~~~~0.0.1.0
```



```
PS C:\Windows\system32> Add-WindowsCapability -Online -Name OpenSSH.Server~~~~0.0.1.0


Path          :
Online        : True
RestartNeeded : False
```

To start and configure OpenSSH Server for initial use, open an elevated PowerShell prompt (right click, Run as an administrator), then run the following commands to start the sshd service:

```
# Start the sshd service Start-
Service sshd

# OPTIONAL but recommended:
Set-Service -Name sshd -StartupType 'Automatic'

# Confirm the Firewall rule is configured. It should be created automatically by setup. Run the
following to verify
if (!(Get-NetFirewallRule -Name "OpenSSH-Server-In-TCP" -ErrorAction SilentlyContinue |
Select-Object Name, Enabled)) {
Write-Output "Firewall Rule 'OpenSSH-Server-In-TCP' does not exist, creating it..."
New-NetFirewallRule -Name 'OpenSSH-Server-In-TCP' -DisplayName 'OpenSSH Server
(sshd)' -Enabled True -Direction Inbound -Protocol TCP -Action Allow -LocalPort 22
} else {
Write-Output "Firewall rule 'OpenSSH-Server-In-TCP' has been created and exists."
}
```



```
PS C:\Windows\system32> # Start the sshd service
>> Start-Service sshd
PS C:\Windows\system32> Set-Service -Name sshd -StartupType 'Automatic'
PS C:\Windows\system32> if (!(Get-NetFirewallRule -Name "OpenSSH-Server-In-TCP" -ErrorAction SilentlyContinue | Select-O
bject Name, Enabled)) {
>>     Write-Output "Firewall Rule 'OpenSSH-Server-In-TCP' does not exist, creating it..."
>>     New-NetFirewallRule -Name 'OpenSSH-Server-In-TCP' -DisplayName 'OpenSSH Server (sshd)' -Enabled True -Direction I
nbound -Protocol TCP -Action Allow -LocalPort 22
>> } else {
>>     Write-Output "Firewall rule 'OpenSSH-Server-In-TCP' has been created and exists."
>> }
Firewall rule 'OpenSSH-Server-In-TCP' has been created and exists.
```

Connect to your server:

ssh domain\username@servername