Lab Project - 6

Objective: Linux SSH Connectivity Labs

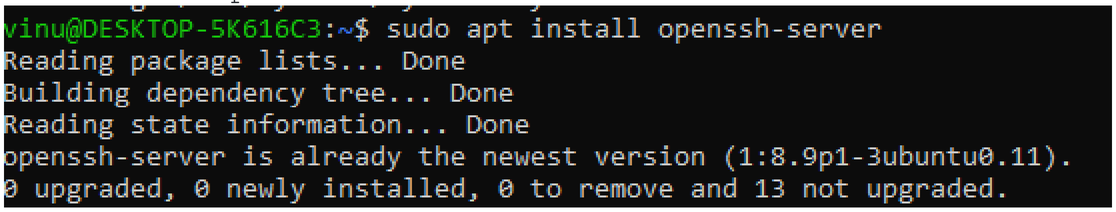
Lab 1: Basic SSH Connectivity

Objective:

• Learn how to set up and use SSH for basic remote access.

Tasks:

1. Install OpenSSH Server:



```
vinu@DESKTOP-5K616C3:~$ sudo apt install openssh-server
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
openssh-server is already the newest version (1:8.9p1-3ubuntu0.11).
0 upgraded, 0 newly installed, 0 to remove and 13 not upgraded.
```

o Install the OpenSSH server package on a Linux machine (if not already installed).

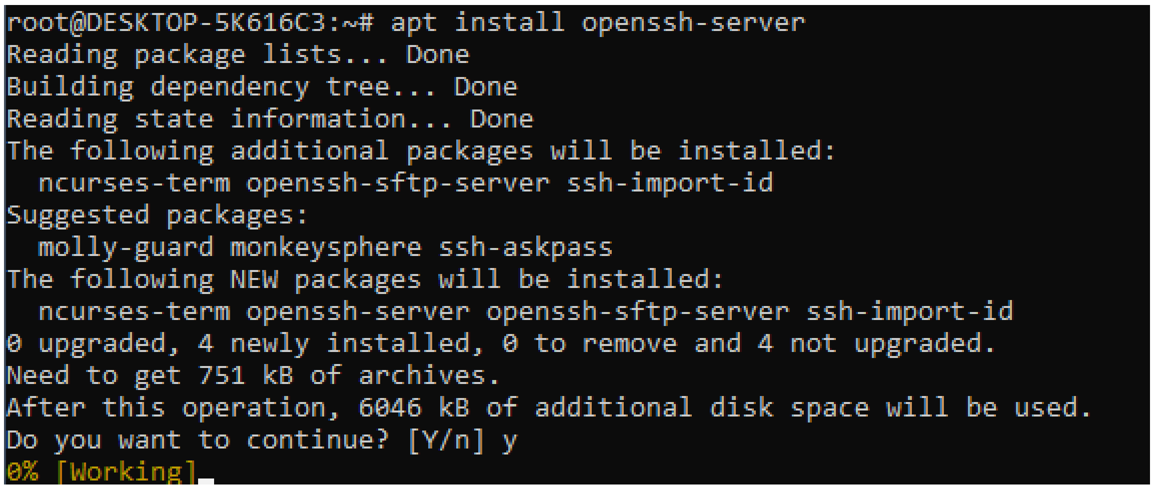
o For Ubuntu/Debian-based systems:

bash

Copy code

sudo apt update

sudo apt install openssh-server



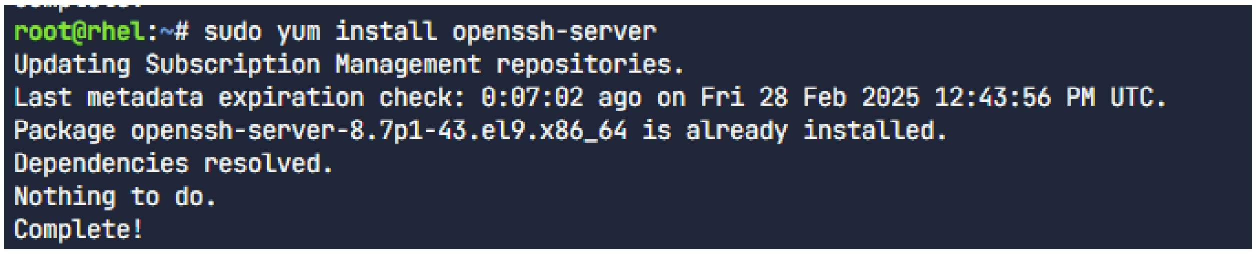
```
root@DESKTOP-5K616C3:~# apt install openssh-server
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  ncurses-term openssh-sftp-server ssh-import-id
Suggested packages:
  molly-guard monkeysphere ssh-askpass
The following NEW packages will be installed:
  ncurses-term openssh-server openssh-sftp-server ssh-import-id
0 upgraded, 4 newly installed, 0 to remove and 4 not upgraded.
Need to get 751 kB of archives.
After this operation, 6046 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
0% [Working]
```

o   For CentOS/RHEL-based systems:

bash

Copy code

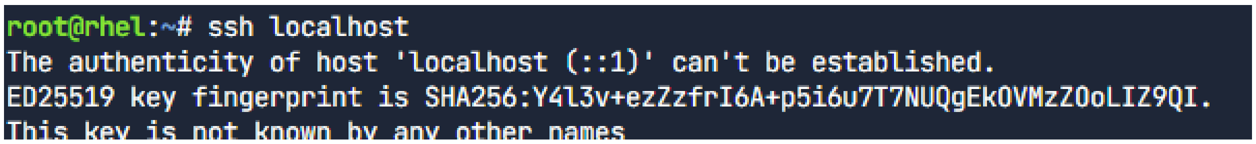
sudo yum install openssh-server



```
root@rhel:~# sudo yum install openssh-server
Updating Subscription Management repositories.
Last metadata expiration check: 0:07:02 ago on Fri 28 Feb 2025 12:43:56 PM UTC.
Package openssh-server-8.7p1-43.el9.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
```

2.Start and Enable SSH Service:

oStart the SSH service and enable it to start at boot.

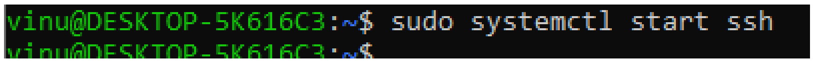


```
root@rhel:~# ssh localhost
The authenticity of host 'localhost (::1)' can't be established.
ED25519 key fingerprint is SHA256:Y4l3v+ezZzfrI6A+p5i6u7T7NUQgEkOVMzZOoLIZ9QI.
This key is not known by any other names
```

Bash

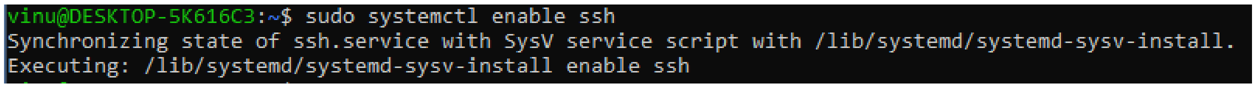
Copy code

sudo systemctl start ssh



```
vinu@DESKTOP-5K616C3:~$ sudo systemctl start ssh
vinu@DESKTOP-5K616C3:~$
```

sudo systemctl enable ssh



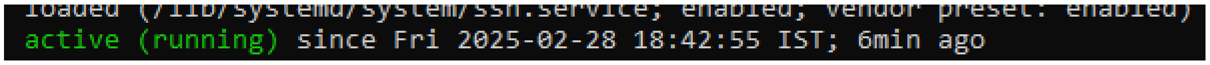
```
vinu@DESKTOP-5K616C3:~$ sudo systemctl enable ssh
Synchronizing state of ssh.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable ssh
```

```
3.Check SSH Service Status:

  oVerify that the SSH server is running.
```

bash

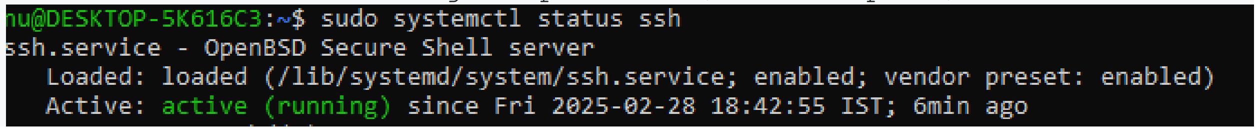
Copy code

sudo systemctl status ssh



```
 loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
 active (running) since Fri 2025-02-28 18:42:55 IST; 6min ago
```

```
4.Verify SSH Port:

  oEnsure SSH is running on port 22 (default port).
```



```
nu@DESKTOP-5K616C3:~$ sudo systemctl status ssh
ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2025-02-28 18:42:55 IST; 6min ago
```

```
5.Connect to the Remote Server via SSH:

  oFrom another machine, connect to the SSH server using:
```

sh

```sh
ssh user@192.168.1.100
```

If using a specific port (e.g., port 2222):

sh

```sh
ssh -p 2222 user@remote_host
```

If using an SSH key instead of a password:

sh

```sh
ssh -i /path/to/private key.pem user@remote host
```

1. **Install the OpenSSH server** (if it's not already installed): OpenSSH is the most common service for SSH access.

   bash Copy Edit

   ```bash
   sudo apt update sudo apt install openssh-server
   ```

2. **Start the SSH service**: You can manually start the SSH service using the following command:

   bash Copy Edit

   ```bash
   sudo systemctl start ssh
   ```

3. **Enable SSH to start at boot**: To ensure that the SSH service starts automatically on boot, use this command:

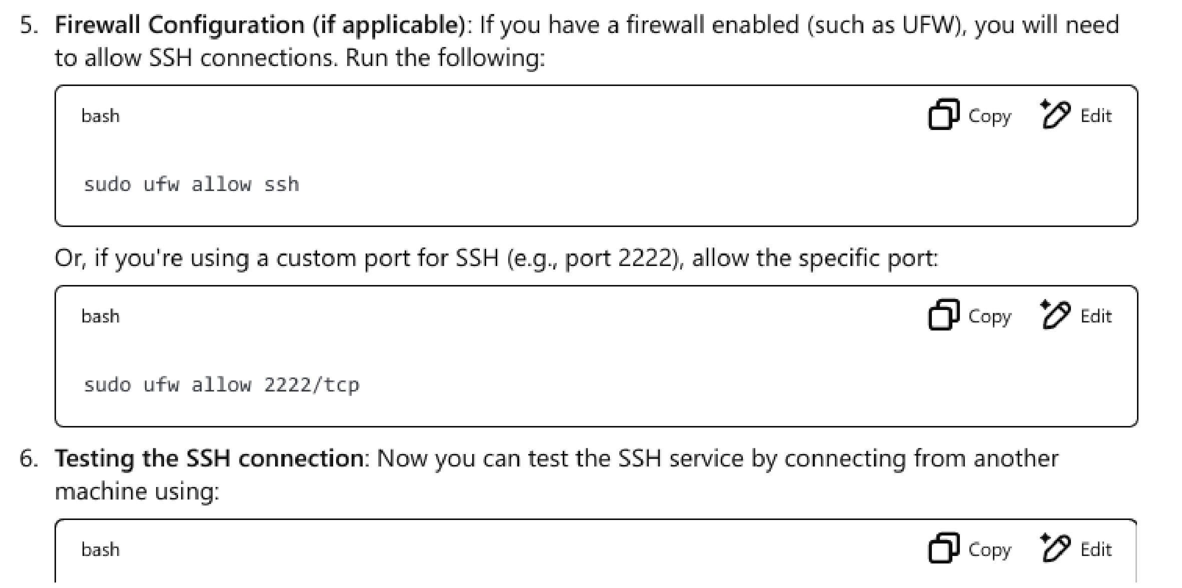
   bash Copy Edit

   ```bash
   sudo systemctl enable ssh
   ```

4. **Verify the SSH service status**: You can check the status of the SSH service to ensure it's running properly:

   bash Copy Edit

   ```bash
   sudo systemctl status ssh
   ```

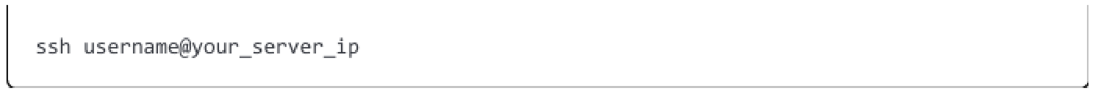

5. **Firewall Configuration (if applicable)**: If you have a firewall enabled (such as UFW), you will need to allow SSH connections. Run the following:

```bash
sudo ufw allow ssh
```

Or, if you're using a custom port for SSH (e.g., port 2222), allow the specific port:

```bash
sudo ufw allow 2222/tcp
```

6. **Testing the SSH connection**: Now you can test the SSH service by connecting from another machine using:

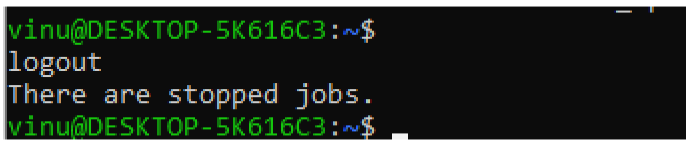


```bash
ssh username@your_server_ip
```

6. Log Out of SSH Session:

   o Use the exit command to end the SSH session.

Ctrl+D



```
vinu@DESKTOP-5K616C3:~$
logout
There are stopped jobs.
vinu@DESKTOP-5K616C3:~$
```

Lab 2: SSH Key-Based Authentication

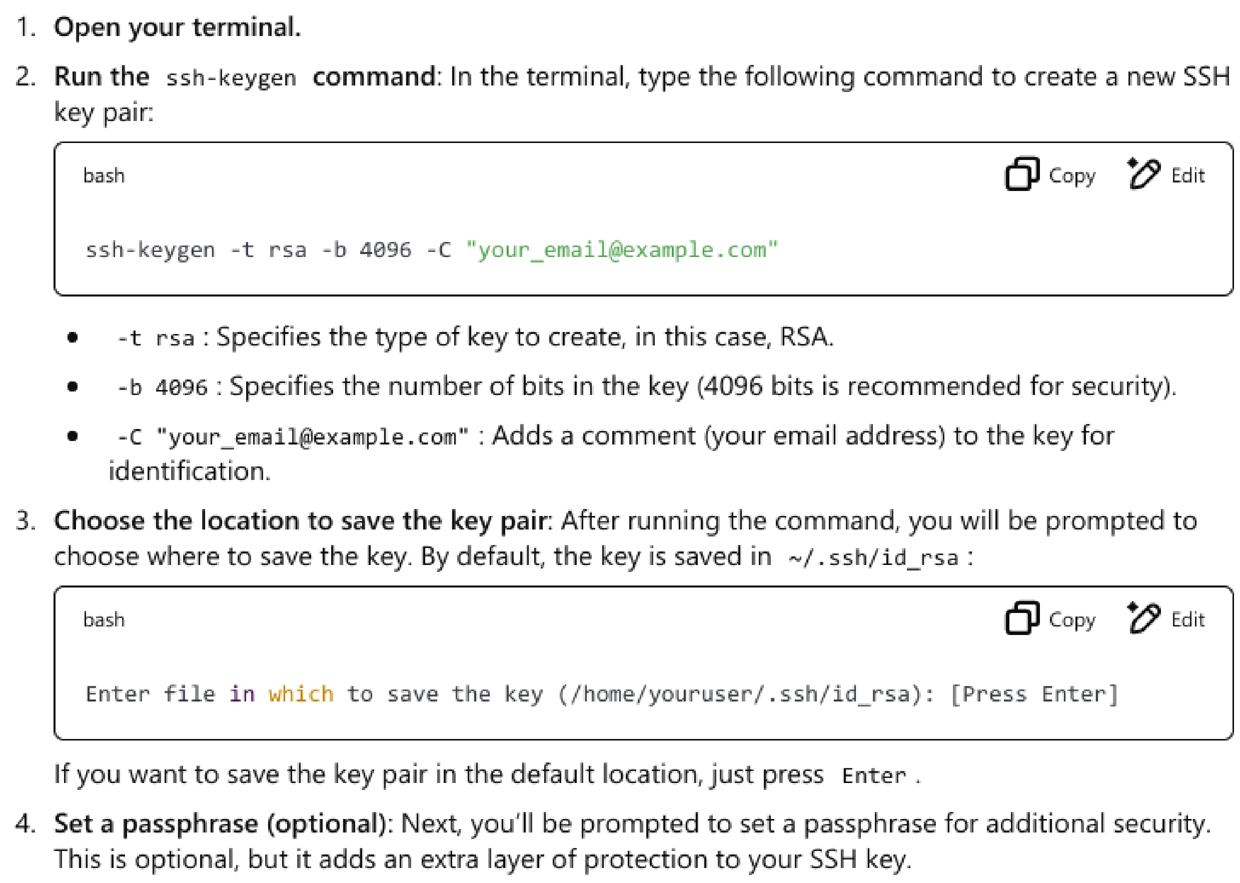
Objective:

• Learn how to configure SSH key-based authentication for more secure and password-less login.

Tasks:

1. Generate SSH Key Pair:

   o On your local machine, generate a new SSH key pair using ssh-keygen.



1. **Open your terminal.**
2. **Run the `ssh-keygen` command**: In the terminal, type the following command to create a new SSH key pair:

```bash
ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
```

   - `-t rsa` : Specifies the type of key to create, in this case, RSA.
   - `-b 4096` : Specifies the number of bits in the key (4096 bits is recommended for security).
   - `-C "your_email@example.com"` : Adds a comment (your email address) to the key for identification.

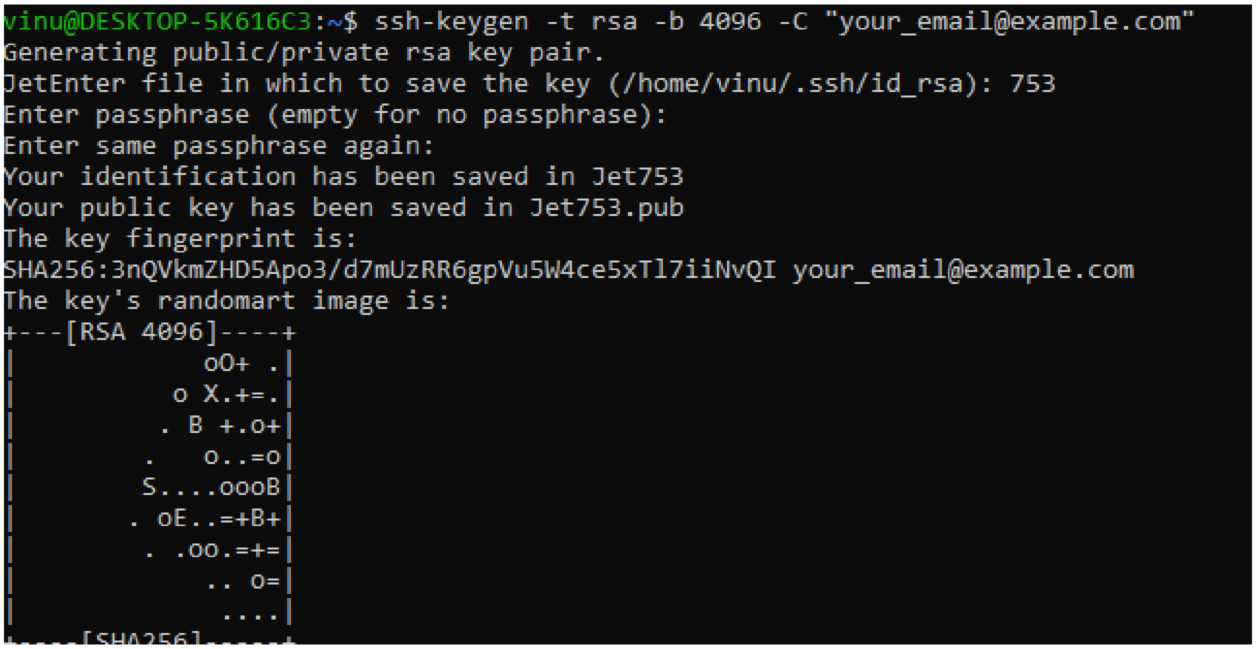
3. **Choose the location to save the key pair**: After running the command, you will be prompted to choose where to save the key. By default, the key is saved in `~/.ssh/id_rsa` :

```bash
Enter file in which to save the key (/home/youruser/.ssh/id_rsa): [Press Enter]
```

   If you want to save the key pair in the default location, just press `Enter` .

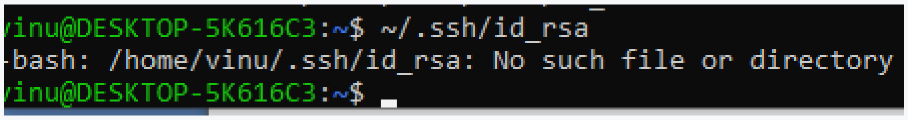
4. **Set a passphrase (optional)**: Next, you'll be prompted to set a passphrase for additional security. This is optional, but it adds an extra layer of protection to your SSH key.

ssh-keygen -t rsa -b 4096 -C your_email@example.com



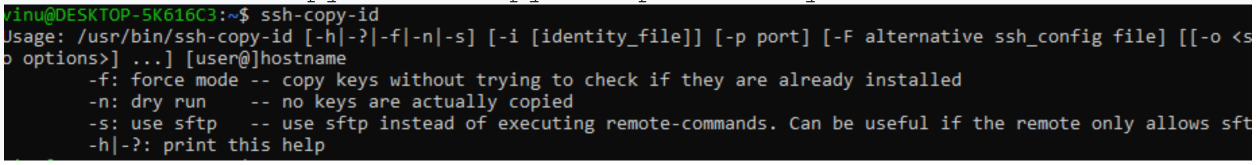
```
vinu@DESKTOP-5K616C3:~$ ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
Generating public/private rsa key pair.
JetEnter file in which to save the key (/home/vinu/.ssh/id_rsa): 753
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in Jet753
Your public key has been saved in Jet753.pub
The key fingerprint is:
SHA256:3nQVkmZHD5Apo3/d7mUzRR6gpVu5W4ce5xTl7iiNvQI your_email@example.com
The key's randomart image is:
+---[RSA 4096]----+
|            oO+ .|
|          o X.+=.|
|         . B +.o+|
|        .   o..=o|
|         S....oooB|
|        . oE..=+B+|
|         . .oo.=+=|
|            .. o=|
|              ....|
+----[SHA256]-----+
```

```
oFollow the prompts to save the key to a default location
(~/.ssh/id_rsa) and optionally set a passphrase.
```



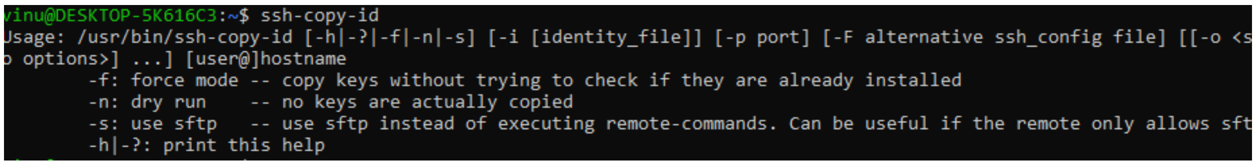
```
vinu@DESKTOP-5K616C3:~$ ~/.ssh/id_rsa
-bash: /home/vinu/.ssh/id_rsa: No such file or directory
vinu@DESKTOP-5K616C3:~$
```

```
2.Copy Public Key to the Remote Server:

  o    Use ssh-copy-id to copy the public key to the remote server.
```



```
vinu@DESKTOP-5K616C3:~$ ssh-copy-id
Usage: /usr/bin/ssh-copy-id [-h|-?|-f|-n|-s] [-i [identity_file]] [-p port] [-F alternative ssh_config file] [[-o <s
o options>] ...] [user@]hostname
        -f: force mode -- copy keys without trying to check if they are already installed
        -n: dry run    -- no keys are actually copied
        -s: use sftp   -- use sftp instead of executing remote-commands. Can be useful if the remote only allows sft
        -h|-?: print this help
```

ssh-copy-id username@server_ip



```
vinu@DESKTOP-5K616C3:~$ ssh-copy-id
Usage: /usr/bin/ssh-copy-id [-h|-?|-f|-n|-s] [-i [identity_file]] [-p port] [-F alternative ssh_config file] [[-o <s
o options>] ...] [user@]hostname
        -f: force mode -- copy keys without trying to check if they are already installed
        -n: dry run    -- no keys are actually copied
        -s: use sftp   -- use sftp instead of executing remote-commands. Can be useful if the remote only allows sft
        -h|-?: print this help
```

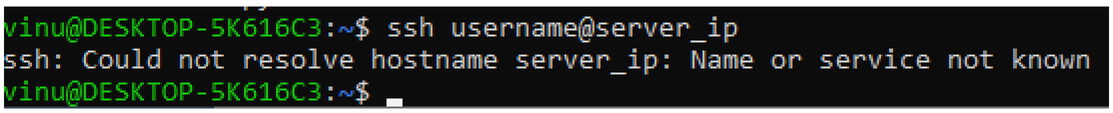
3.Test Key-Based Authentication:

oAttempt to SSH into the remote server. You should be logged in without needing to enter the password.

bash

Copy code

ssh username@server_ip



```
vinu@DESKTOP-5K616C3:~$ ssh username@server_ip
ssh: Could not resolve hostname server_ip: Name or service not known
vinu@DESKTOP-5K616C3:~$ _
```

4.Disable Password Authentication (optional):

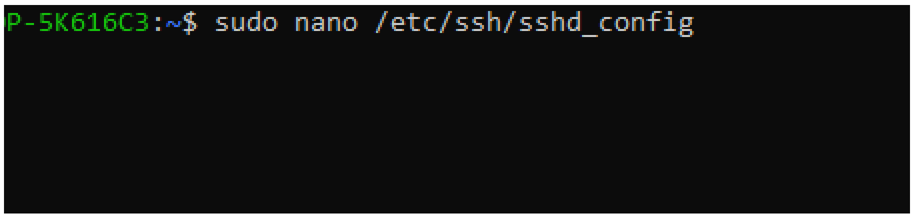
o For additional security, you can disable password-based login on the server by modifying the SSH configuration file (/etc/ssh/sshd_config).

#Set PasswordAuthentication to no.

bash

Copy code

sudo nano /etc/ssh/sshd_config



```
P-5K616C3:~$ sudo nano /etc/ssh/sshd_config
```

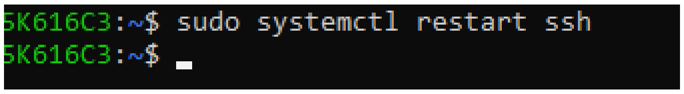
PasswordAuthentication no

o Restart the SSH service:

bash

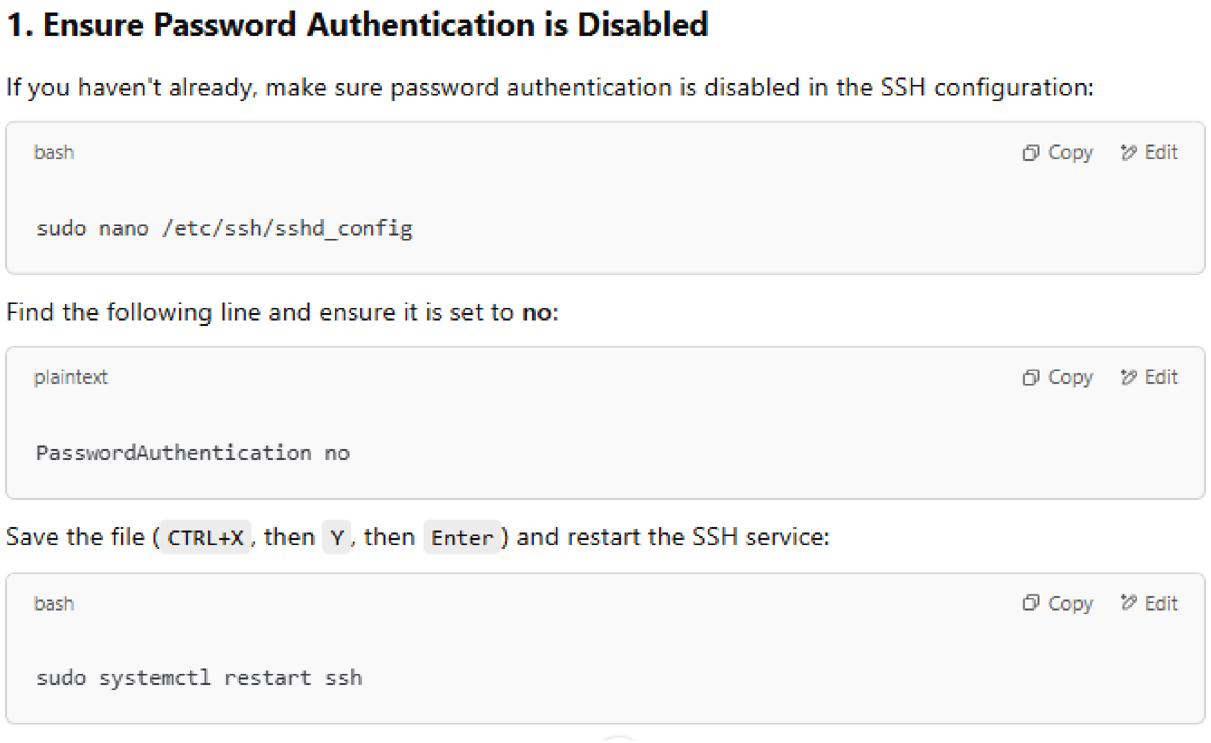
Copy code

sudo systemctl restart ssh



```
5K616C3:~$ sudo systemctl restart ssh
5K616C3:~$ _
```

5. Test SSH Connection After Disabling Password Authentication:



1. Ensure Password Authentication is Disabled

If you haven't already, make sure password authentication is disabled in the SSH configuration:

```bash
sudo nano /etc/ssh/sshd_config
```

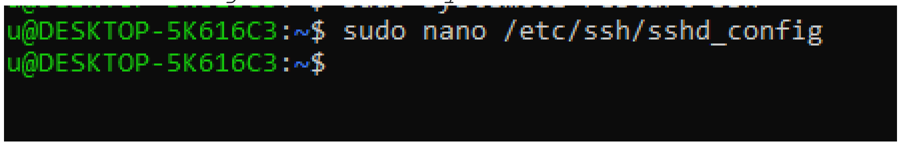
Find the following line and ensure it is set to **no**:

```
PasswordAuthentication no
```

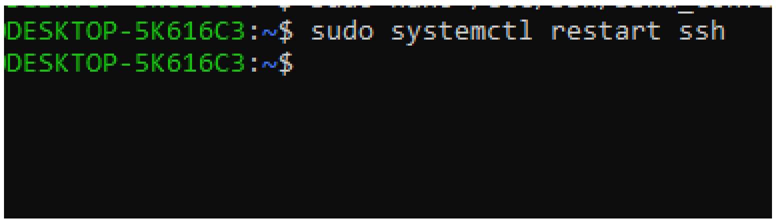
Save the file (`CTRL+X`, then `Y`, then `Enter`) and restart the SSH service:

```bash
sudo systemctl restart ssh
```

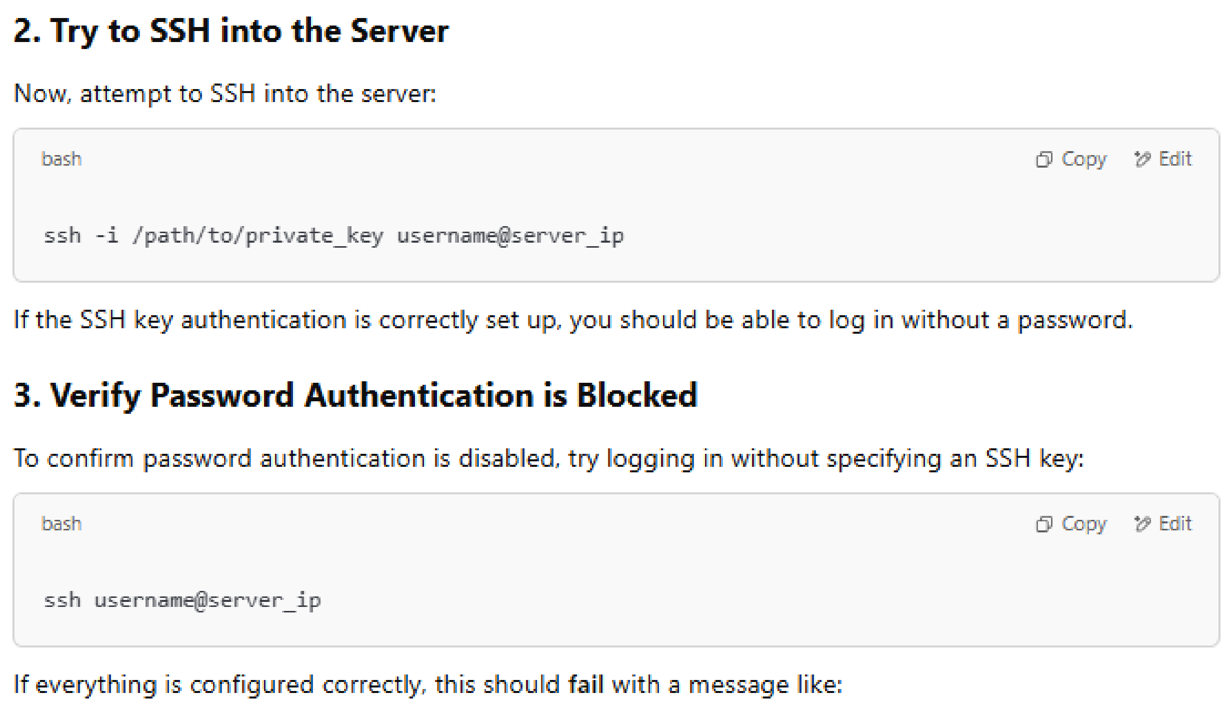
Try to SSH into the server again. You should only be able to connect using the SSH key.



```
u@DESKTOP-5K616C3:~$ sudo nano /etc/ssh/sshd_config
u@DESKTOP-5K616C3:~$
```



```
DESKTOP-5K616C3:~$ sudo systemctl restart ssh
DESKTOP-5K616C3:~$
```



2. Try to SSH into the Server

Now, attempt to SSH into the server:

bash

```
ssh -i /path/to/private_key username@server_ip
```

If the SSH key authentication is correctly set up, you should be able to log in without a password.

3. Verify Password Authentication is Blocked

To confirm password authentication is disabled, try logging in without specifying an SSH key:

bash

```
ssh username@server_ip
```

If everything is configured correctly, this should **fail** with a message like:

Lab 3: SSH Configuration and Security

Objective:

• Learn how to harden and secure your SSH configuration.

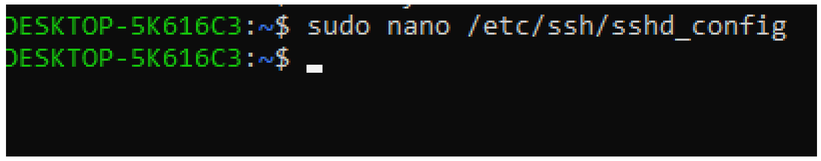
Tasks:

```
 1.Change Default SSH Port:

  oEdit the SSH configuration file (/etc/ssh/sshd_config) to change
the default port from 22 to another port (e.g., 2222).
```

bash

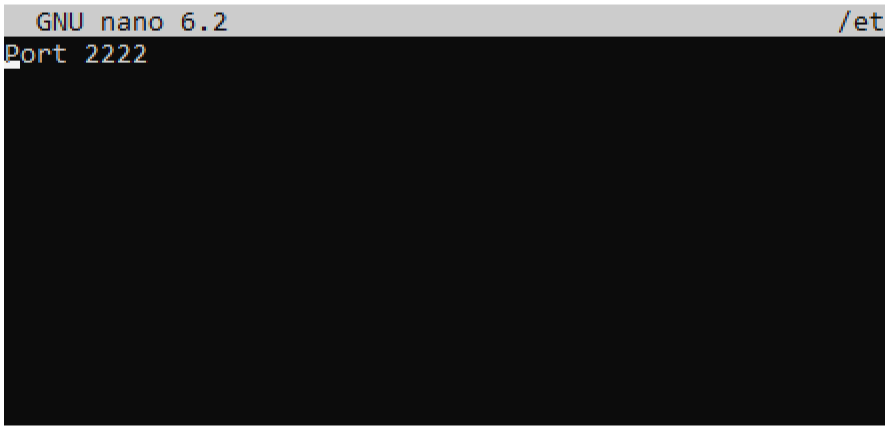
Copy code

sudo nano /etc/ssh/sshd_config



```
DESKTOP-5K616C3:~$ sudo nano /etc/ssh/sshd_config
DESKTOP-5K616C3:~$ 
```

Port 2222



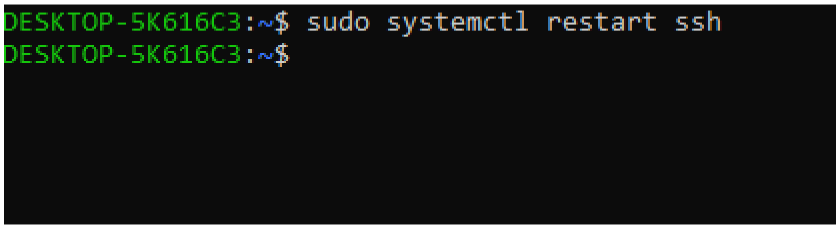
```
  GNU nano 6.2                                                   /et
Port 2222
```

oRestart the SSH service:

bash

Copy code

sudo systemctl restart ssh



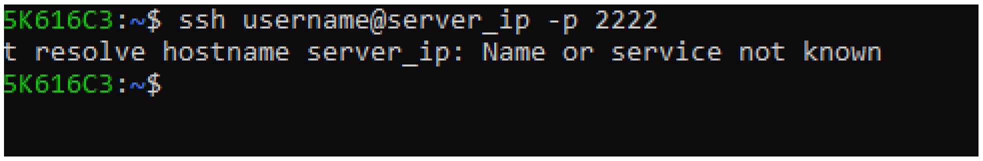
```
DESKTOP-5K616C3:~$ sudo systemctl restart ssh
DESKTOP-5K616C3:~$
```

o Test the connection by specifying the new port:

bash

Copy code

ssh username@server_ip -p 2222



```
5K616C3:~$ ssh username@server_ip -p 2222
t resolve hostname server_ip: Name or service not known
5K616C3:~$
```

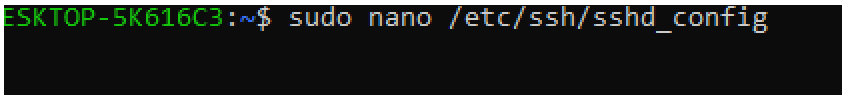
2.Disable Root Login via SSH:

oModify /etc/ssh/sshd_config to disable direct root login.

bash

Copy code

sudo nano /etc/ssh/sshd_config



```
ESKTOP-5K616C3:~$ sudo nano /etc/ssh/sshd_config
```

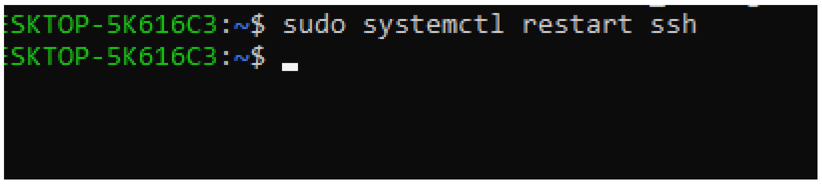
PermitRootLogin no

o Restart the SSH service:

bash

Copy code

sudo systemctl restart ssh



```
ESKTOP-5K616C3:~$ sudo systemctl restart ssh
ESKTOP-5K616C3:~$ _
```

3.Limit SSH Access to Specific Users or Groups:

oUse the AllowUsers or AllowGroups directive in /etc/ssh/sshd_config to allow only specific users or groups to log in via SSH.

bash

Copy code

sudo nano /etc/ssh/sshd_config

AllowUsers user1 user2

#or

AllowGroups sshusers

o Restart the SSH service:

bash

Copy code

sudo systemctl restart ssh

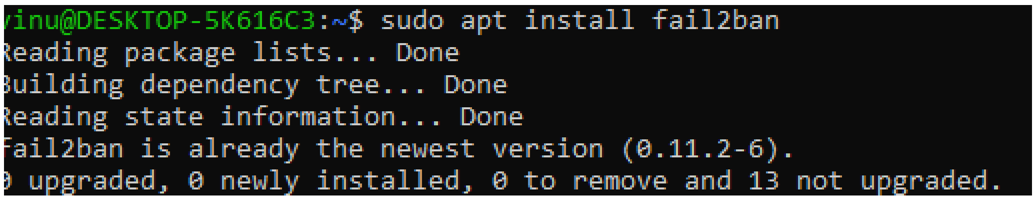
4.Enable SSH Rate Limiting with Fail2Ban:

oInstall fail2ban to block IP addresses that attempt too many failed SSH login attempts.

bash

Copy code

sudo apt install fail2ban



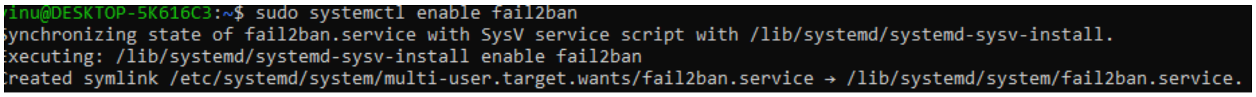
```
vinu@DESKTOP-5K616C3:~$ sudo apt install fail2ban
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
fail2ban is already the newest version (0.11.2-6).
0 upgraded, 0 newly installed, 0 to remove and 13 not upgraded.
```

o Enable and start the service:

bash

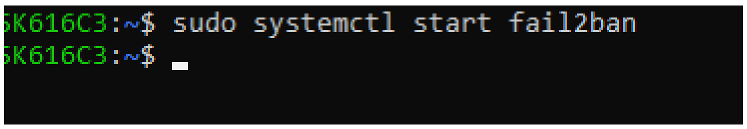
Copy code

sudo systemctl enable fail2ban



```
vinu@DESKTOP-5K616C3:~$ sudo systemctl enable fail2ban
Synchronizing state of fail2ban.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable fail2ban
Created symlink /etc/systemd/system/multi-user.target.wants/fail2ban.service → /lib/systemd/system/fail2ban.service.
```

sudo systemctl start fail2ban

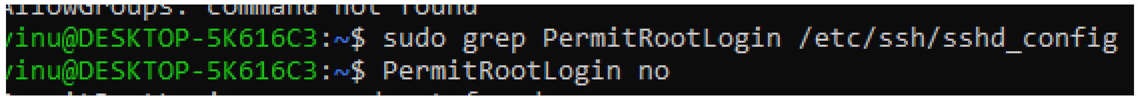


```
5K616C3:~$ sudo systemctl start fail2ban
5K616C3:~$
```

5.Test Security Configurations:

oTest that root login is disabled, specific users/groups can log in, and the new SSH port is working correctly.

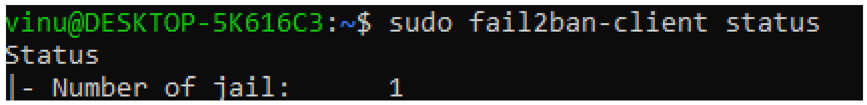
sudo grep PermitRootLogin /etc/ssh/sshd_config



```
AllowGroups: command not found
vinu@DESKTOP-5K616C3:~$ sudo grep PermitRootLogin /etc/ssh/sshd_config
vinu@DESKTOP-5K616C3:~$ PermitRootLogin no
```

oAttempt SSH connections with invalid passwords to check if fail2ban blocks the IP after multiple failed attempts.

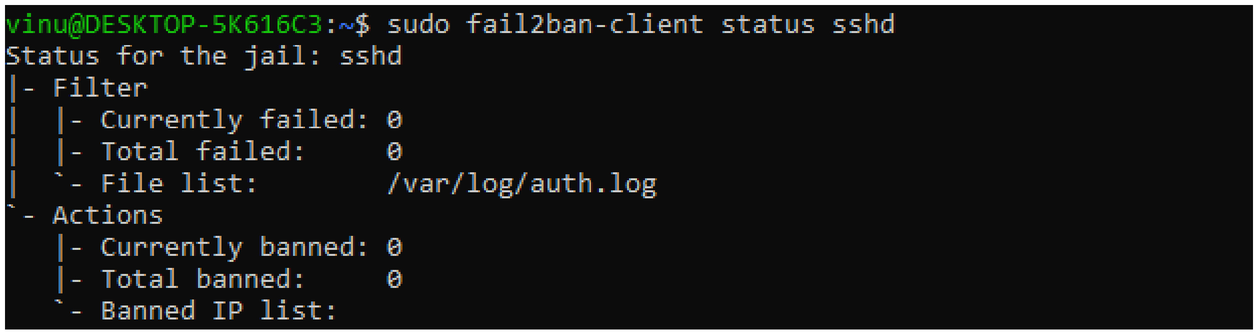
sudo fail2ban-client status



```
vinu@DESKTOP-5K616C3:~$ sudo fail2ban-client status
Status
|- Number of jail:      1
```

.

sudo fail2ban-client status sshd



```
vinu@DESKTOP-5K616C3:~$ sudo fail2ban-client status sshd
Status for the jail: sshd
|- Filter
|  |- Currently failed: 0
|  |- Total failed:     0
|  `- File list:        /var/log/auth.log
`- Actions
   |- Currently banned: 0
   |- Total banned:     0
   `- Banned IP list:
```

Lab 4: SSH Tunneling and Port Forwarding

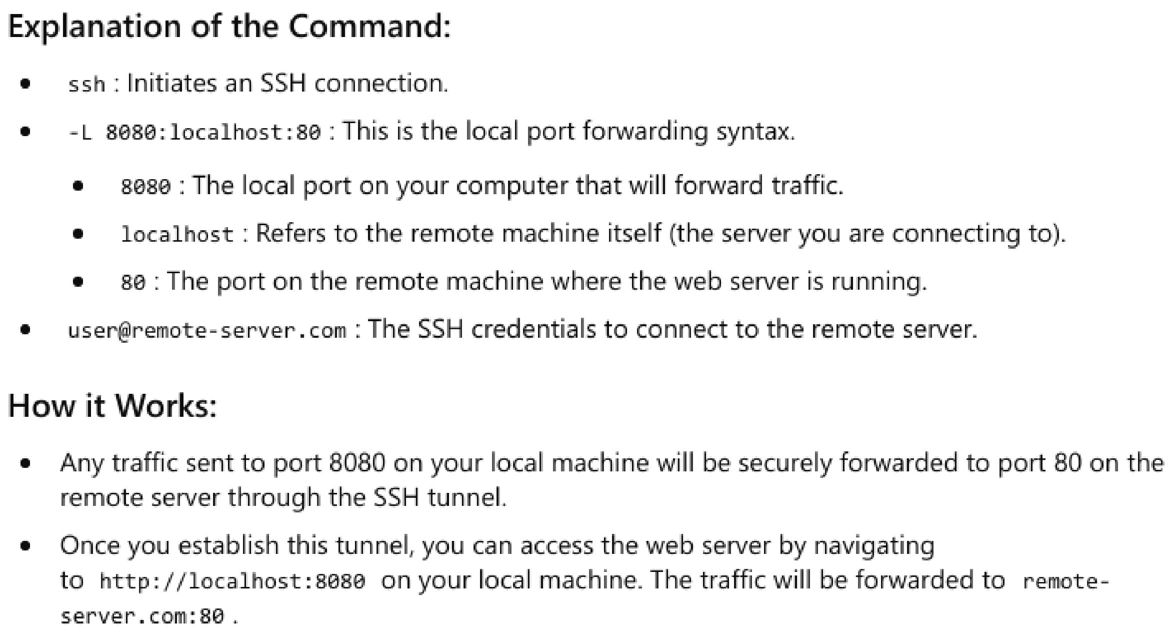
Objective:

• Learn how to set up SSH tunneling for secure communication between two systems.

Tasks:

1. Local Port Forwarding:

   oForward a local port to a remote server. For example, if you have a web server running on port 80 on a remote system, you can forward it to a local port:



Explanation of the Command:

- `ssh` : Initiates an SSH connection.
- `-L 8080:localhost:80` : This is the local port forwarding syntax.
  - `8080` : The local port on your computer that will forward traffic.
  - `localhost` : Refers to the remote machine itself (the server you are connecting to).
  - `80` : The port on the remote machine where the web server is running.
- `user@remote-server.com` : The SSH credentials to connect to the remote server.

How it Works:

- Any traffic sent to port 8080 on your local machine will be securely forwarded to port 80 on the remote server through the SSH tunnel.
- Once you establish this tunnel, you can access the web server by navigating to `http://localhost:8080` on your local machine. The traffic will be forwarded to `remote-server.com:80` .

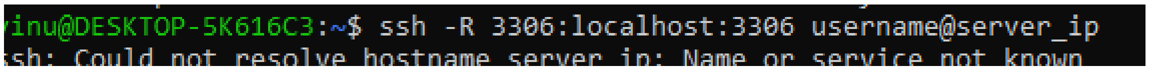
After establishing the connection, you can access the remote web server by navigating to http://localhost:8080 on your local browser.

```
MySQL  JS > http://localhost:8080
         -> 
```

2.Remote Port Forwarding:

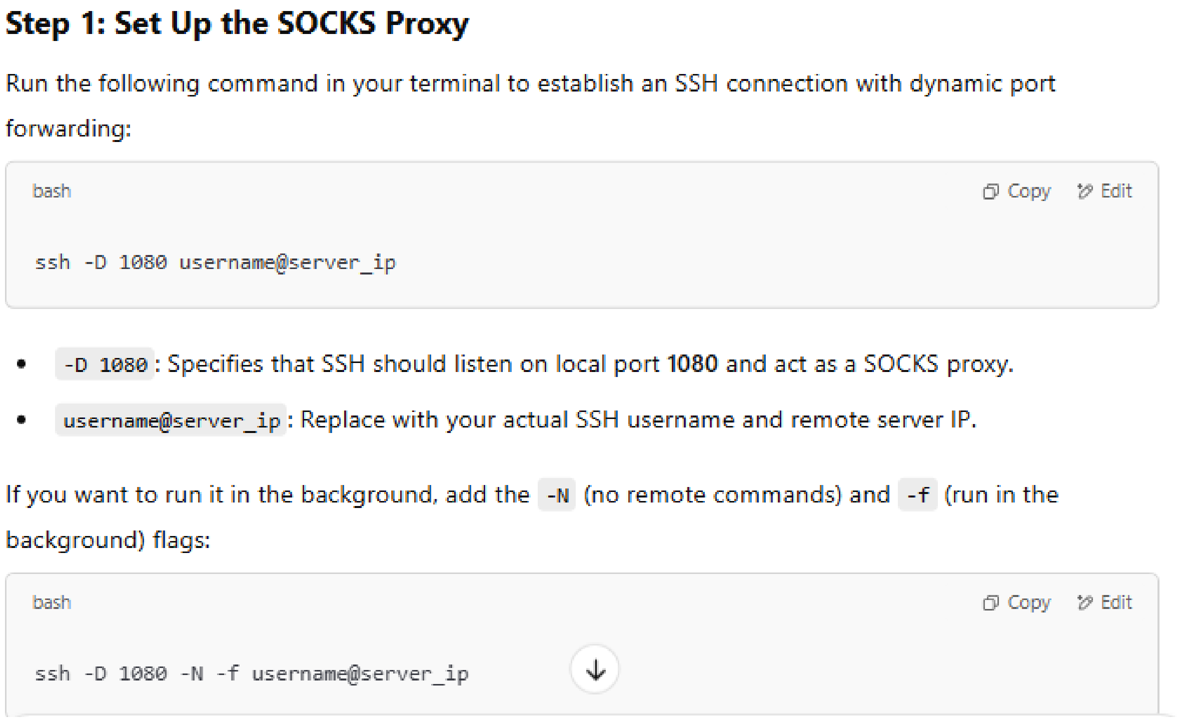
oForward a remote port to a local system. For example, if you want to access a service running locally on port 3306 from a remote server, you can use:

ssh -R 3306:localhost:3306 username@server_ip



```
vinu@DESKTOP-5K616C3:~$ ssh -R 3306:localhost:3306 username@server_ip
ssh: Could not resolve hostname server_ip: Name or service not known
```

3.Dynamic Port Forwarding (SOCKS Proxy):



Step 1: Set Up the SOCKS Proxy

Run the following command in your terminal to establish an SSH connection with dynamic port forwarding:

```bash
ssh -D 1080 username@server_ip
```

- `-D 1080`: Specifies that SSH should listen on local port **1080** and act as a SOCKS proxy.
- `username@server_ip`: Replace with your actual SSH username and remote server IP.

If you want to run it in the background, add the `-N` (no remote commands) and `-f` (run in the background) flags:

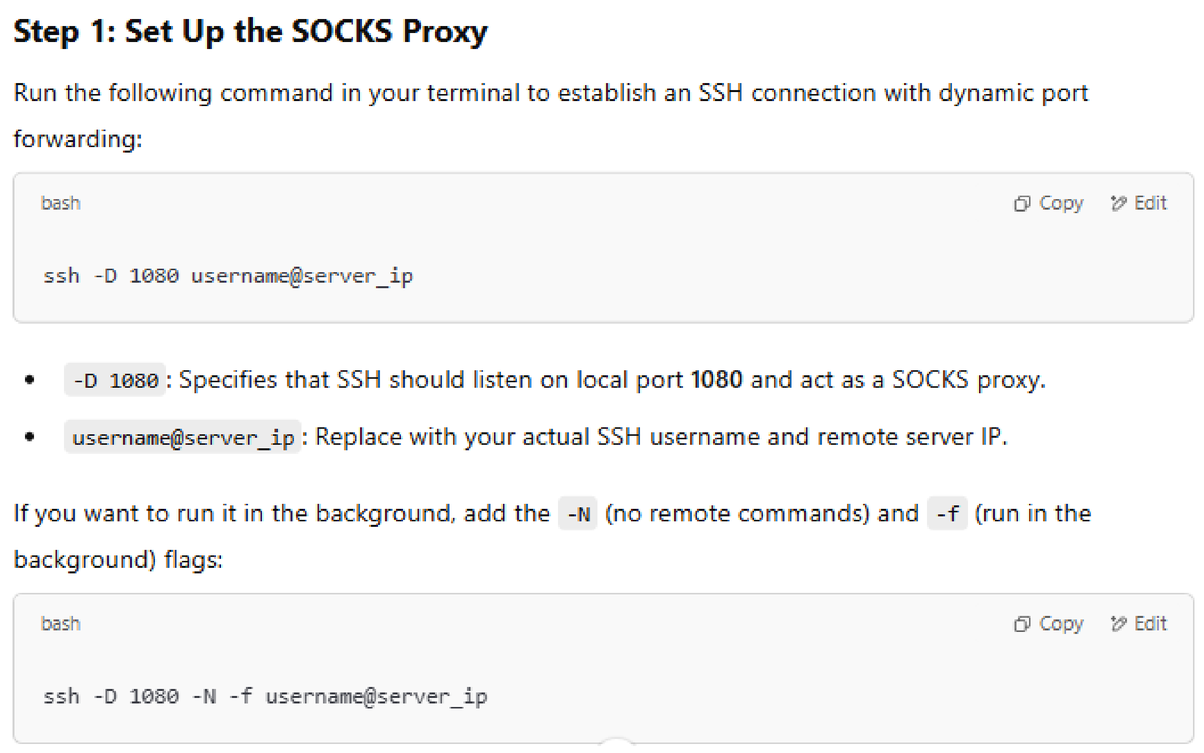
```bash
ssh -D 1080 -N -f username@server_ip
```

oSet up SSH to create a SOCKS proxy for secure browsing.

bash

Copy code

ssh -D 1080 username@server_ip



Step 1: Set Up the SOCKS Proxy

Run the following command in your terminal to establish an SSH connection with dynamic port forwarding:

```bash
ssh -D 1080 username@server_ip
```

- `-D 1080`: Specifies that SSH should listen on local port **1080** and act as a SOCKS proxy.
- `username@server_ip`: Replace with your actual SSH username and remote server IP.

If you want to run it in the background, add the `-N` (no remote commands) and `-f` (run in the background) flags:

```bash
ssh -D 1080 -N -f username@server_ip
```

o Configure your browser to use the SOCKS proxy on port 1080 to securely browse the web.

```bash
ssh -D 1080 -N -f username@server_ip
```

Step 2: Configure Your Browser to Use the Proxy

Once the SSH tunnel is established, configure your browser to use the SOCKS proxy:

For Firefox:

1. Open **Settings** → Scroll down to **Network Settings** → Click **Settings...**
2. Select **Manual proxy configuration**.
3. Under **SOCKS Host**, enter `127.0.0.1` and **Port** `1080`.
4. Choose **SOCKS v5**.
5. (Optional) Enable **Proxy DNS when using SOCKS v5** to prevent DNS leaks.

For Google Chrome (Using SwitchyOmega Extension)

1. Install the **SwitchyOmega** extension.
2. Create a new profile and configure:
   - **Protocol:** SOCKS5
   - **Server:** 127.0.0.1
   - **Port:** 1080
3. Apply the settings and switch to the new profile when using the proxy.

Step 3: Verify the Connection

To confirm that your traffic is being routed through the proxy, visit a site like https://www.whatismyip.com/ and check if your IP matches your SSH server's IP.

Step 4: Close the Proxy When Done

To terminate the SSH tunnel, find the process and kill it:

bash Copy Edit

```bash
ps aux | grep ssh
kill <PID>
```

or simply use:

bash Copy Edit

```bash
pkill -f "ssh -D 1080"
```

This setup helps encrypt your web traffic, bypass network restrictions, and anonymize your browsing. Let me know if you need more details! 🚀

```
2.     Copy Public Key to the Remote Server:

  o    Use ssh-copy-id to copy the public key to the remote server.
```

Lab 5: SSH Agent and Forwarding

Objective:

• Learn to use SSH agent forwarding for accessing remote servers that require authentication via SSH keys.

Tasks:

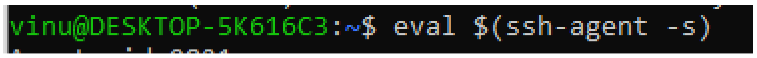
```
 1.Start the SSH Agent:

  oStart the SSH agent on your local machine.
```

.

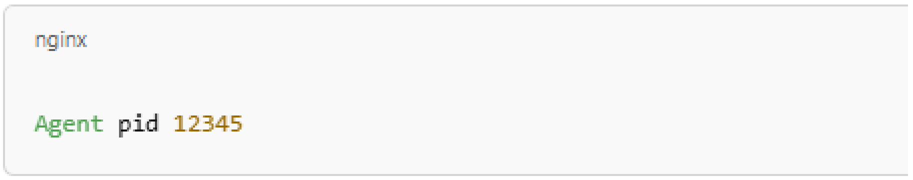
bash

Copy code

eval $(ssh-agent -s)



```
vinu@DESKTOP-5K616C3:~$ eval $(ssh-agent -s)
```



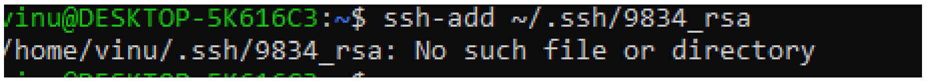
```
nginx

Agent pid 12345
```

```
2.Add SSH Key to the Agent:

  oAdd your private key to the SSH agent.
```

bash

Copy code

ssh-add ~/.ssh/id_rsa



```
vinu@DESKTOP-5K616C3:~$ ssh-add ~/.ssh/9834_rsa
/home/vinu/.ssh/9834_rsa: No such file or directory
```

2. Add your SSH key to the agent

Run:

```bash
ssh-add ~/.ssh/id_rsa
```

If your key has a different name, replace `id_rsa` with the correct key filename.

3. Verify the key has been added

Run:

```bash
ssh-add -l
```

Notes:

- If you receive a `Could not open a connection to your authentication agent` error, run `eval "$(ssh-agent -s)"` again and retry.
- If your SSH key is password-protected, you'll need to enter the passphrase.

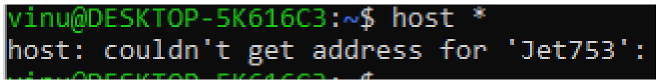
3.Enable SSH Agent Forwarding:

oOn your local machine, configure ~/.ssh/config to enable agent forwarding.

bash

Copy code

Host *



```
vinu@DESKTOP-5K616C3:~$ host *
host: couldn't get address for 'Jet753':
```

ForwardAgent yes

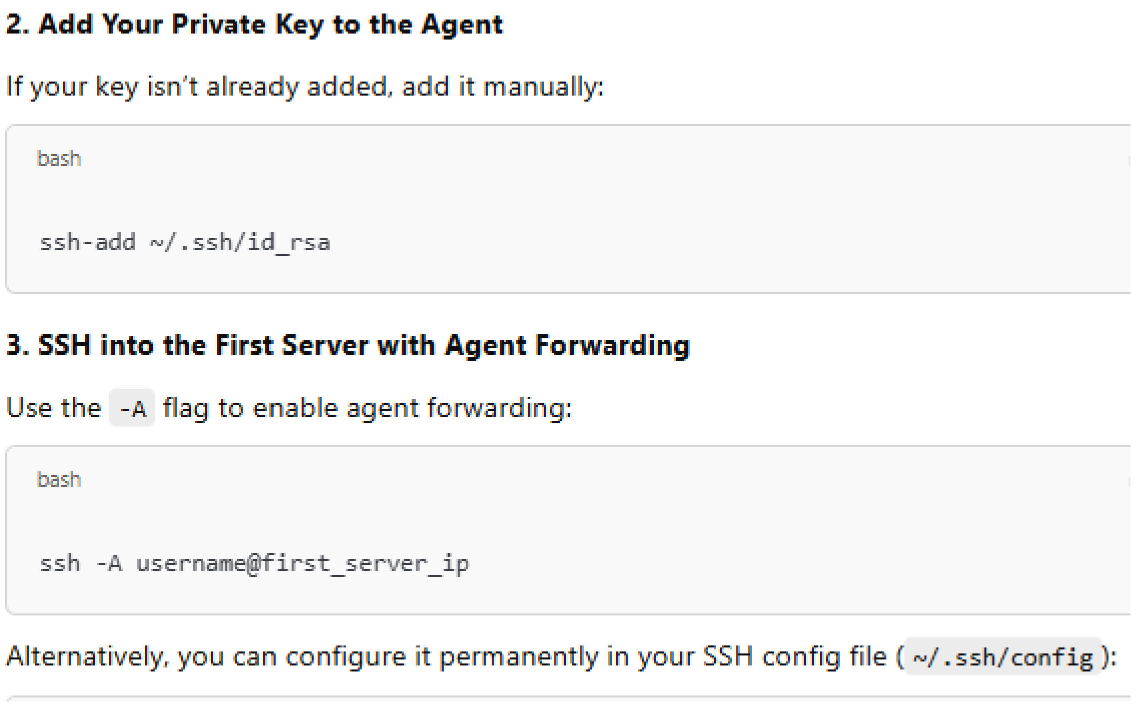
4.Access Remote Server with SSH Agent Forwarding:

oSSH into the first server and then SSH from that server to a second server. The SSH agent on your local machine will be forwarded, allowing you to use the SSH key for the second connection without needing to copy it over.

oExample:

ssh username@first_server_ip

ssh username@second_server_ip



2. Add Your Private Key to the Agent

If your key isn't already added, add it manually:

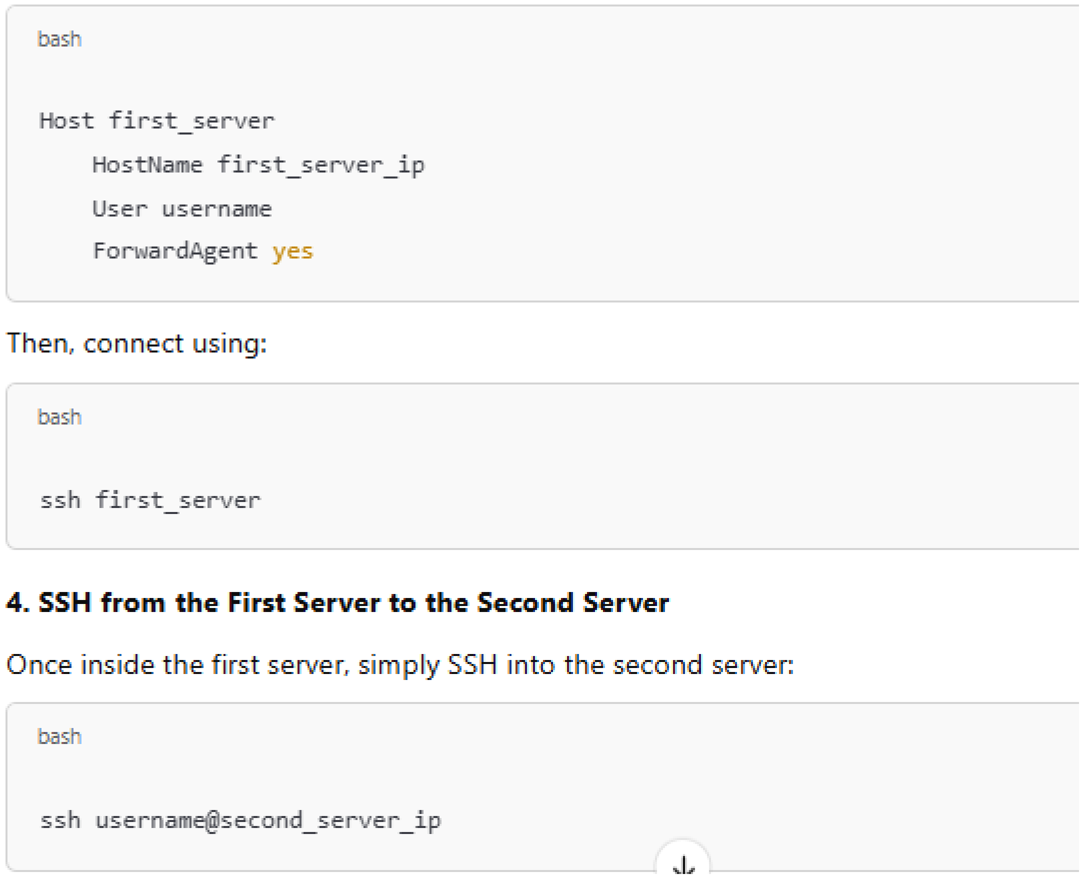
```bash
ssh-add ~/.ssh/id_rsa
```

3. SSH into the First Server with Agent Forwarding

Use the `-A` flag to enable agent forwarding:

```bash
ssh -A username@first_server_ip
```

Alternatively, you can configure it permanently in your SSH config file (`~/.ssh/config`):



```bash
Host first_server
    HostName first_server_ip
    User username
    ForwardAgent yes
```

Then, connect using:

```bash
ssh first_server
```

4. SSH from the First Server to the Second Server

Once inside the first server, simply SSH into the second server:

```bash
ssh username@second_server_ip
```

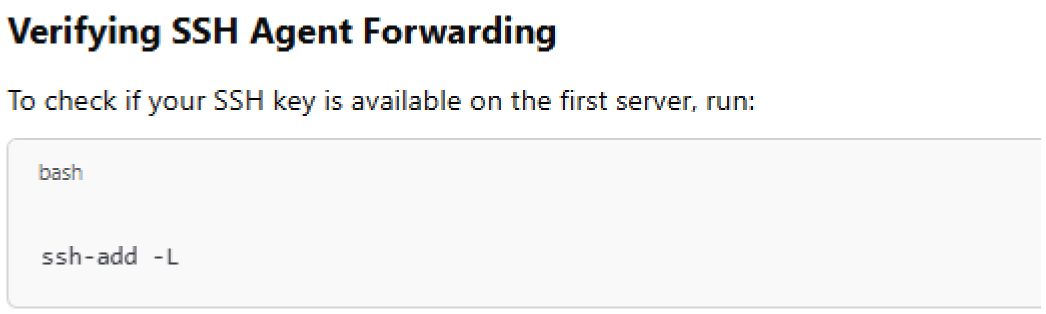
5.Verify SSH Agent Forwarding:

- Check if agent forwarding is enabled by running the following on the second server:

bash

Copy code

ssh-add –l



Verifying SSH Agent Forwarding

To check if your SSH key is available on the first server, run:

```bash
ssh-add -L
```