

COMPREHENSIVE GUIDE ON SSH TUNNELING



Introduction

Basically, tunnelling is a process that allows data sharing or communication between two different networks privately. Tunneling is normally performed by encapsulating the private network data and protocol information inside the public network broadcast units so that the private network protocol information is visible to the public network as data.

SSH Tunnel: Tunneling is the concept of encapsulating the network protocol into another protocol. Here we put it into SSH, so all network communication is encrypted. Because tunnelling involves repackaging the traffic data into a different form, perhaps with encryption as standard, a third use is to hide the nature of the traffic that is run through the tunnels.

Types of SSH Tunneling:

1. Dynamic SSH tunneling
2. Local SSH tunneling
3. Remote SSH tunneling

Let's Begin!!

Objective: To establish an SSH connection between remote PC and the local system of the different network.

Here I have set my own lab which consists of three systems in the following network:

SSH server (two Ethernet interface)

IP 192.168.1.104 connected with the remote system

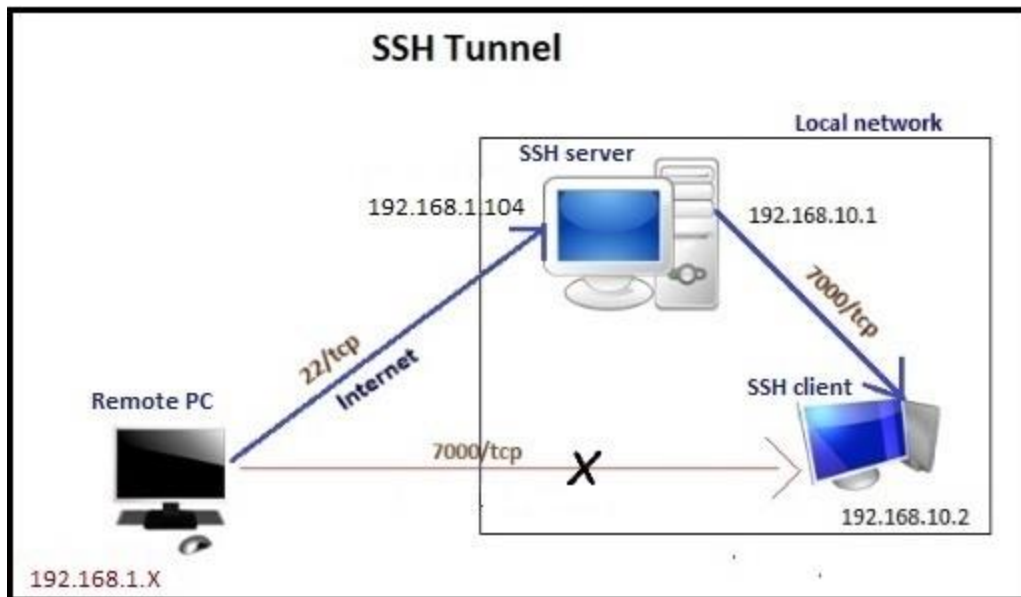
IP 192.168.10.1 connected to local network system 192.168.10.2

SSH client (local network) holds IP 192.168.10.2

Remote system (outside the network)

In the following image, we are trying to explain the SSH tunnelling process where a remote PC is trying to connect to 192.168.10.2, which is on the INTRANET of another network. To establish a connection with **an SSH client (raj)**, the remote PC will create an SSH tunnel which will connect with the local system via an **SSH server (Ignite)**.

NOTE: Service SSH must be activated



The given below image describes the network configuration for the **SSH server** where it shows two IPs, 192.168.1.104 and another 192.168.10.1.

```
raj@ubuntu:~$ ifconfig
ens33  Link encap:Ethernet  HWaddr 00:0c:29:d7:e7:43
       inet addr:192.168.1.104  Bcast:192.168.1.255  Mask:255.255.255.0
       inet6 addr: fe80::836f:2737:911b:8a26/64 Scope:Link
       UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
       RX packets:12658 errors:0 dropped:0 overruns:0 frame:0
       TX packets:118 errors:0 dropped:0 overruns:0 carrier:0
       collisions:0 txqueuelen:1000
       RX bytes:10548981 (10.5 MB)  TX bytes:14008 (14.0 KB)

ens38  Link encap:Ethernet  HWaddr 00:0c:29:d7:e7:4d
       inet addr:192.168.10.1  Bcast:192.168.10.255  Mask:255.255.255.0
       inet6 addr: fe80::1266:d6b6:8e79:fb52/64 Scope:Link
       UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
       RX packets:95 errors:0 dropped:0 overruns:0 frame:0
       TX packets:98 errors:0 dropped:0 overruns:0 carrier:0
       collisions:0 txqueuelen:1000
       RX bytes:11912 (11.9 KB)  TX bytes:11976 (11.9 KB)

lo     Link encap:Local Loopback
       inet addr:127.0.0.1  Mask:255.0.0.0
       inet6 addr: ::1/128 Scope:Host
       UP LOOPBACK RUNNING  MTU:65536  Metric:1
       RX packets:204 errors:0 dropped:0 overruns:0 frame:0
       TX packets:204 errors:0 dropped:0 overruns:0 carrier:0
       collisions:0 txqueuelen:1000
       RX bytes:15028 (15.0 KB)  TX bytes:15028 (15.0 KB)
```

Another image given below describes the network configuration for the **SSH client**, which shows IP 192.168.10.2.


```

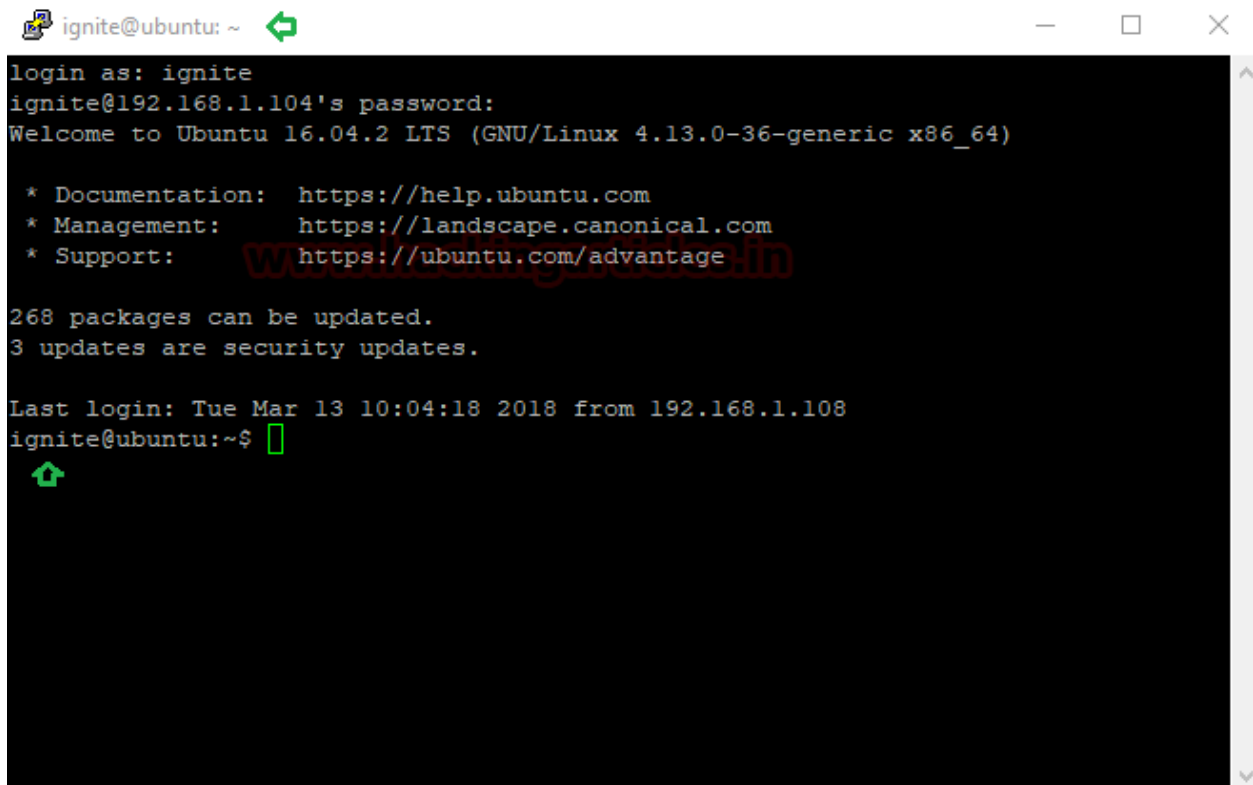
root@ignite:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:56:4f:2e
          inet addr:192.168.10.2  Bcast:192.168.10.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe56:4f2e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1037 errors:0 dropped:1 overruns:0 frame:0
          TX packets:298 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:142045 (142.0 KB)  TX bytes:48788 (48.7 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:345 errors:0 dropped:0 overruns:0 frame:0
          TX packets:345 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:27024 (27.0 KB)  TX bytes:27024 (27.0 KB)

```

Dynamic SSH Tunneling through Windows

The **remote PC** is attempting to connect to an **SSH server (192.168.1.104)** via **port 22** and has successfully logged in to the server. Here, we used putty to establish a connection between the SSH server (Ubuntu) and the remote user (Windows).



```

ignite@ubuntu: ~
login as: ignite
ignite@192.168.1.104's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.13.0-36-generic x86_64)

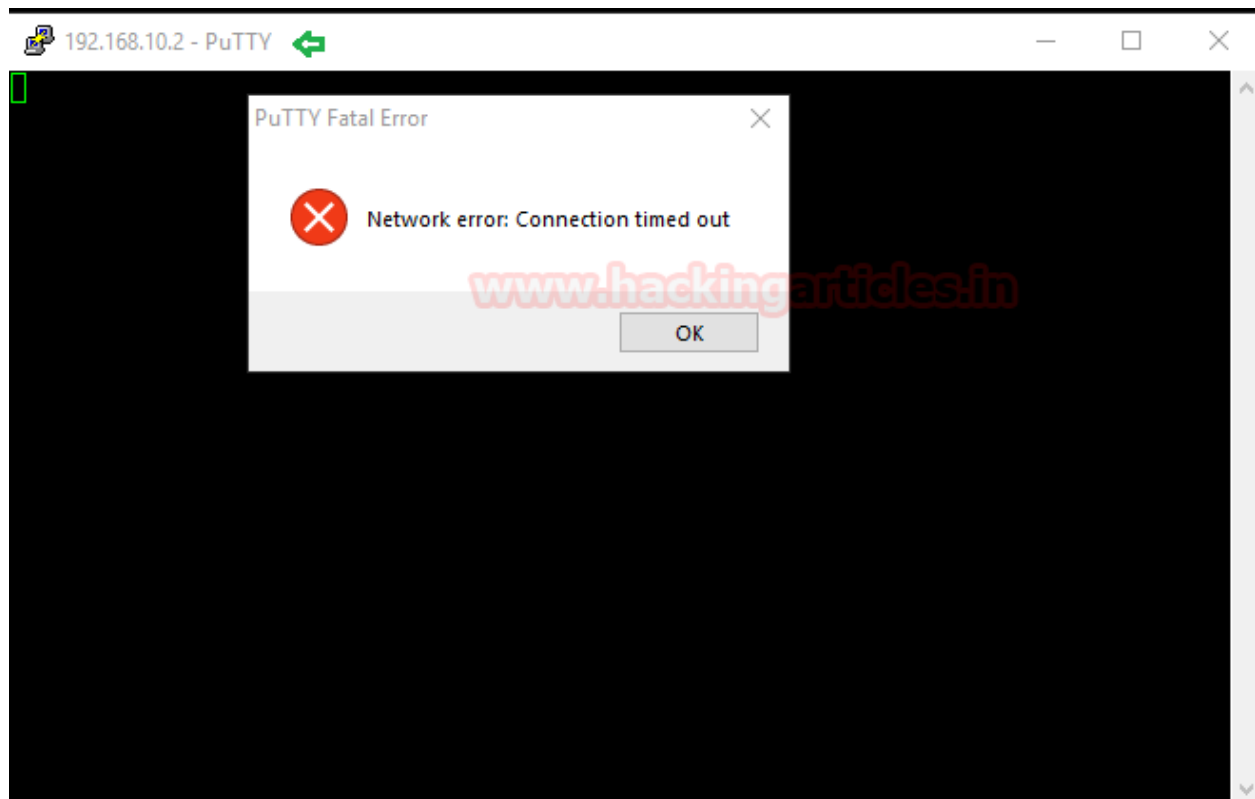
 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

268 packages can be updated.
3 updates are security updates.

Last login: Tue Mar 13 10:04:18 2018 from 192.168.1.108
ignite@ubuntu:~$

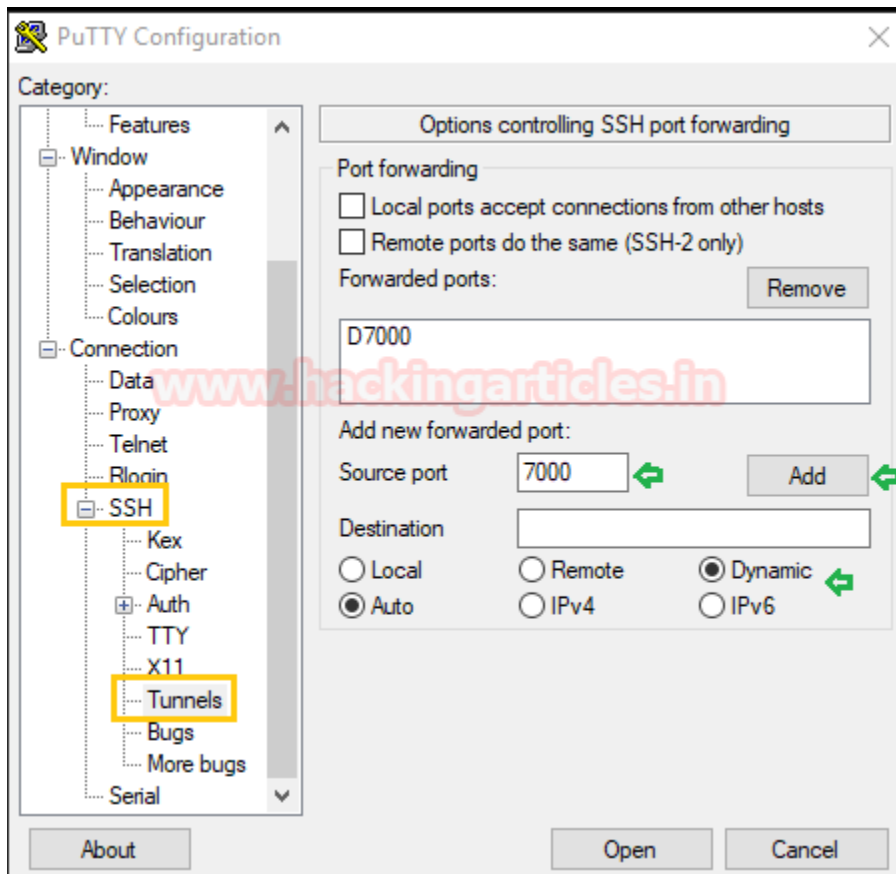
```

Similarly, now the remote PC is trying to connect with the **client PC (192.168.10.2)** via **port 22**, since they belong to different networks, he receives a network error.

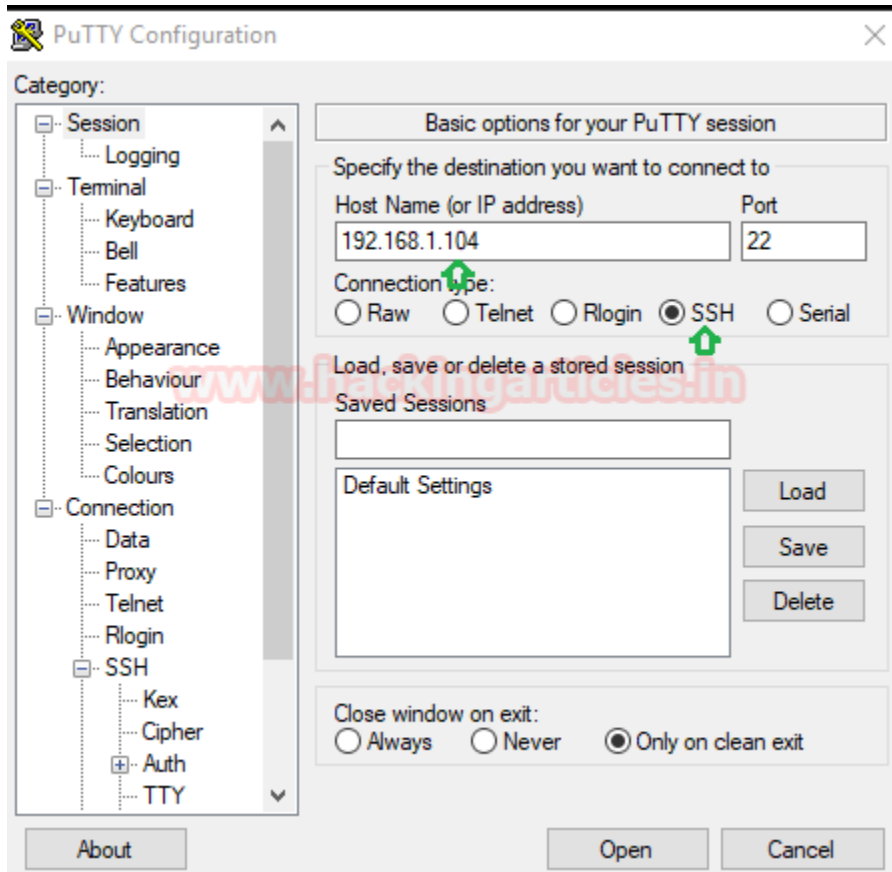


Step for Dynamic SSH tunneling

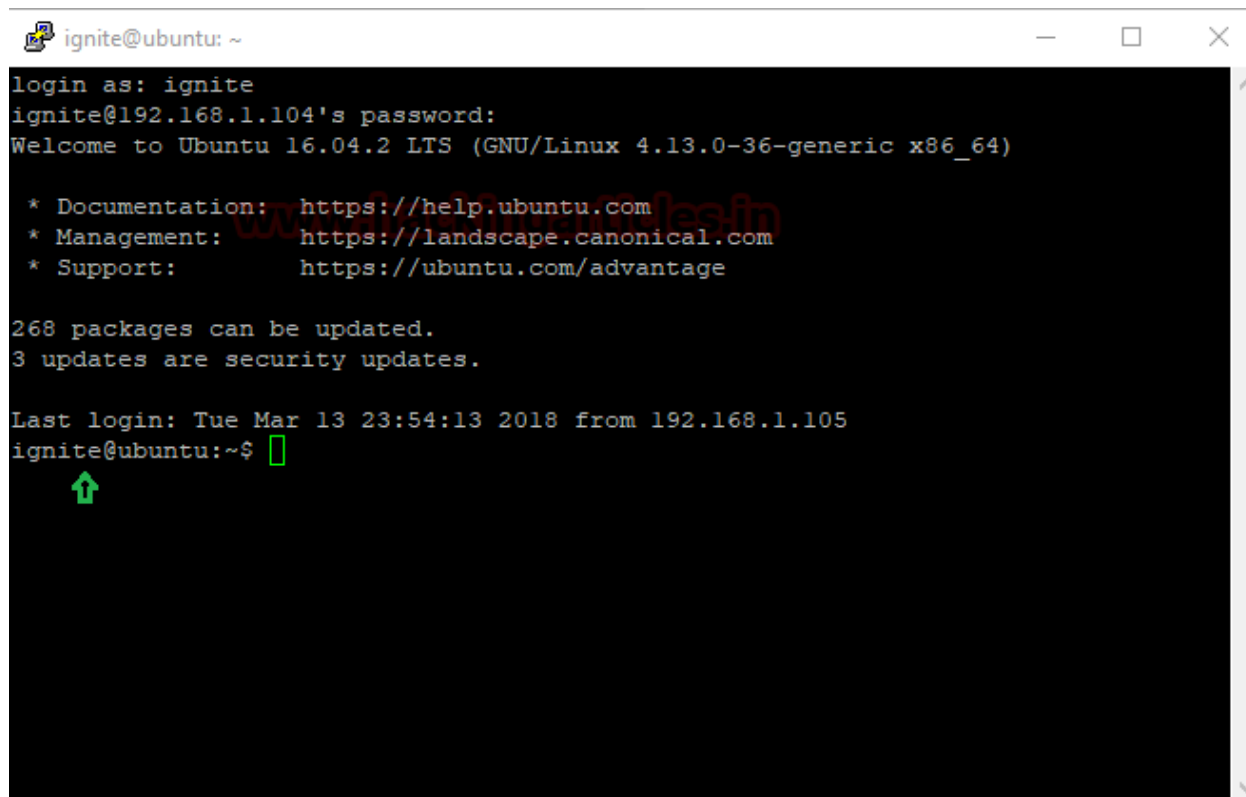
- Choose option **SSH > Tunnel** given in the left column of the category.
- Give new port forwarded as **7000** and connection type as **dynamic** and click on ADD at last.



Now connect to SSH server 192.168.1.104 via port 22 and then click on **"open"** when all things are set.



First, it will connect to the SSH server. As you can see, we are connected to the SSH server (Ignite).

A terminal window titled 'ignite@ubuntu: ~' with standard window controls. The text inside shows a successful login for the 'ignite' user on an Ubuntu 16.04.2 LTS system. It displays the system's kernel version (4.13.0-36-generic x86_64) and provides links for documentation, management, and support. It also informs the user that 268 packages can be updated, including 3 security updates. The last login time is shown as Tuesday, March 13, 2018, at 23:54:13 from IP 192.168.1.105. The prompt 'ignite@ubuntu:~\$' is followed by a green cursor and a green upward arrow icon.

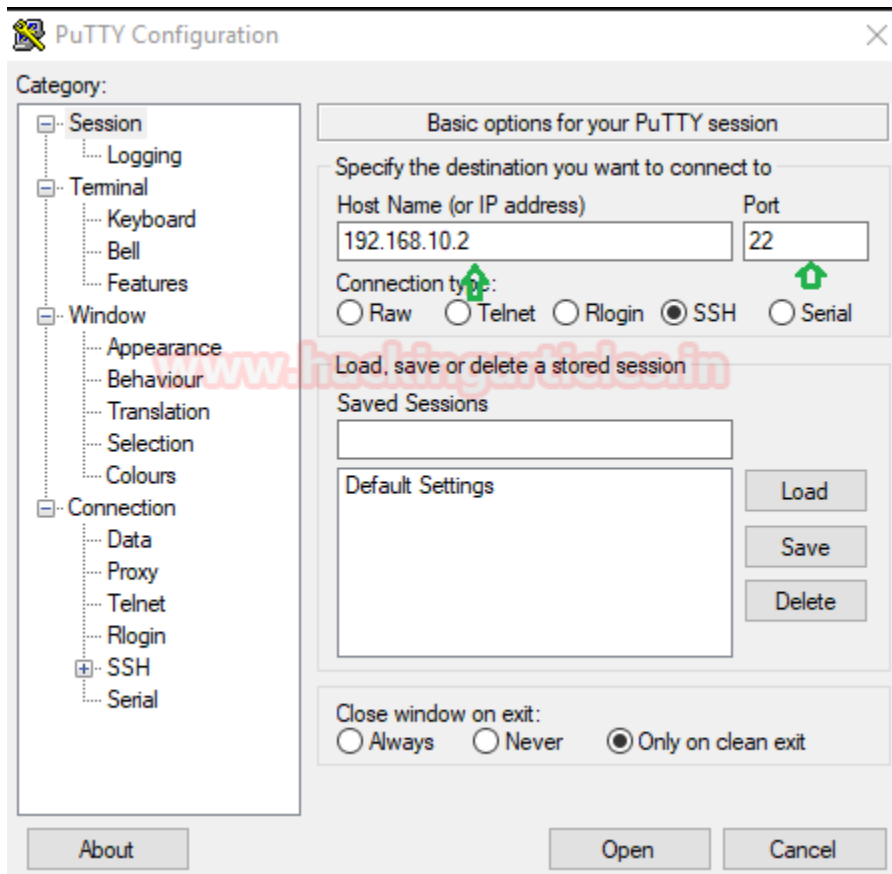
```
ignite@ubuntu: ~
login as: ignite
ignite@192.168.1.104's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.13.0-36-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

268 packages can be updated.
3 updates are security updates.

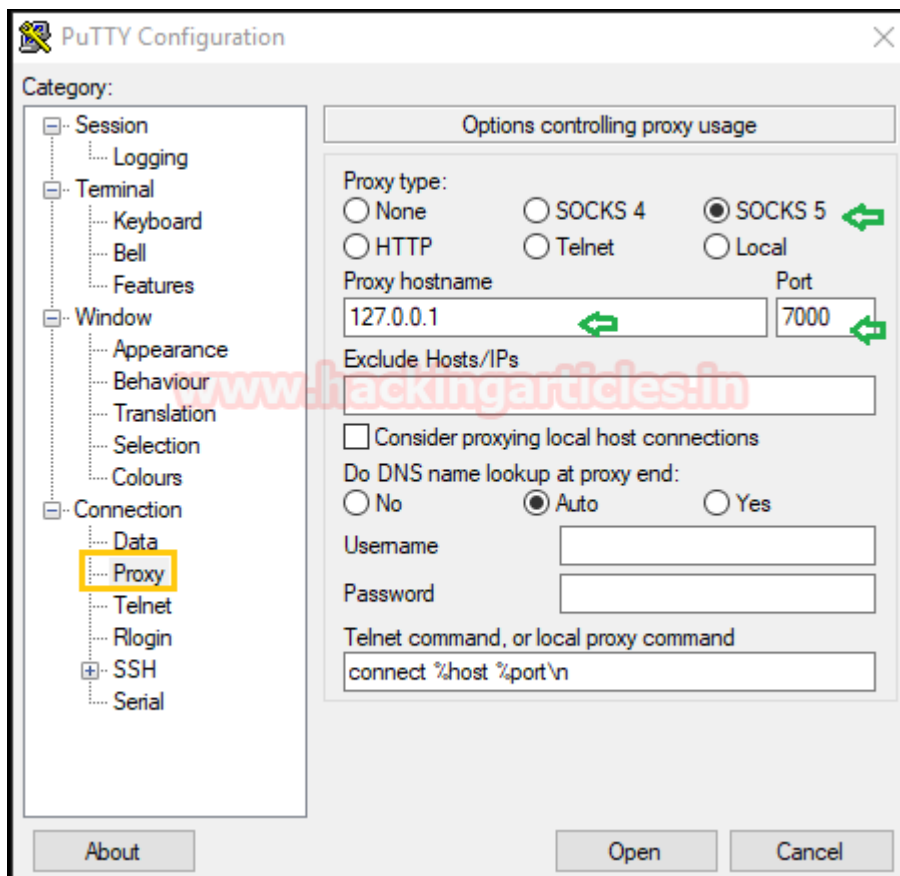
Last login: Tue Mar 13 23:54:13 2018 from 192.168.1.105
ignite@ubuntu:~$
```

Now login into Putty again and give the IP of the client system as Host Name **192.168.10.2** and Port **22** for SSH, then click on **open**.



The open previous running window of putty choose **Proxy** option from the category and follow given below step:

- Select proxy type as **SOCKS 5**
- Give proxy hostname as 127.0.0.1 and port 7000
- Click on open to establish a connection.



Awesome!! We have successfully accessed the SSH client (raj) via port 7000.

```
raj@ignite: ~  
login as: raj  
raj@192.168.10.2's password:  
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)  
  
* Documentation: https://help.ubuntu.com/  
  
Last login: Tue Mar 13 10:01:13 2018 from 192.168.10.1  
raj@ignite:~$ ifconfig  
eth0      Link encap:Ethernet  HWaddr 00:0c:29:56:4f:2e  
          inet addr:192.168.10.2  Bcast:192.168.10.255  Mask:255.255.255.0  
          inet6 addr: fe80::20c:29ff:fe56:4f2e/64 Scope:Link  
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1  
          RX packets:1139 errors:0 dropped:1 overruns:0 frame:0  
          TX packets:326 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:1000  
          RX bytes:152385 (152.3 KB)  TX bytes:55223 (55.2 KB)  
  
lo        Link encap:Local Loopback  
          inet addr:127.0.0.1  Mask:255.0.0.0  
          inet6 addr: ::1/128 Scope:Host  
          UP LOOPBACK RUNNING  MTU:65536  Metric:1  
          RX packets:349 errors:0 dropped:0 overruns:0 frame:0  
          TX packets:349 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:0  
          RX bytes:27364 (27.3 KB)  TX bytes:27364 (27.3 KB)  
  
raj@ignite:~$
```

Dynamic SSH Tunneling through Kali Linux on Port 80

Now we are employing Kali Linux for SSH tunnelling and demonstrating how an attacker or Linux user can take the privilege of tunnelling and can establish an SSH connection with client systems.

```
ssh -D 7000 ignite@192.168.1.104
```

Enter the user's password for login and get access to the **SSH server** as shown below.

```
root@kali:~# ssh -D 7000 ignite@192.168.1.104
ignite@192.168.1.104's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.13.0-36-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

268 packages can be updated.
3 updates are security updates.

Last login: Tue Mar 13 23:57:45 2018 from 192.168.1.105
ignite@ubuntu:~$
```

Next, we need to set a network proxy for enabling socks5 and for that follow below steps.

- In your web browser "Firefox" go to option for general setting tab and open **Network Proxy**.
- Choose **No Proxy**
- Enable **socksv5**

Add **localhost, 127.0.0.1** as Manual proxy

Connection Settings

Configure Proxies to Access the Internet

☐ No proxy

☐ Auto-detect proxy settings for this network

☐ Use system proxy settings

☒ Manual proxy configuration:

HTTP Proxy:

Port:

0

☐ Use this proxy server for all protocols

SSL Proxy:

Port:

0

FTP Proxy:

Port:

0

SOCKS Host:

127.0.0.1

Port:

7000

☐ SOCKS v4

☒ SOCKS v5

No Proxy for:

localhost, 127.0.0.1

Example: .mozilla.org, .net.nz, 192.168.1.0/24

☐ Automatic proxy configuration URL:

Reload

☐ Do not prompt for authentication if password is saved

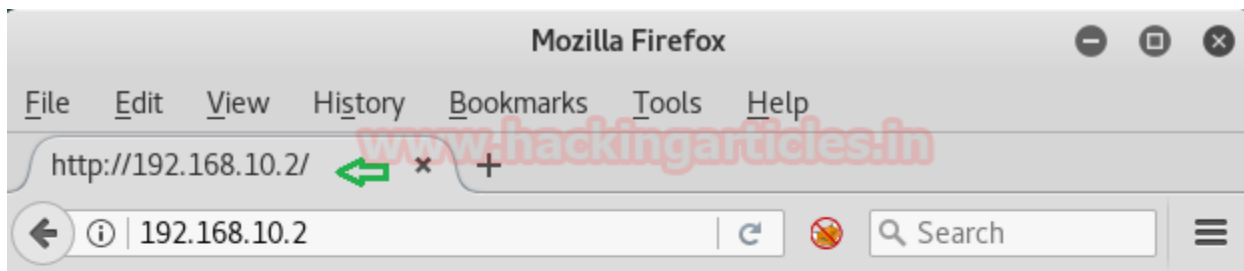
☐ Proxy DNS when using SOCKS v5

Help

Cancel

OK

So from the given below image, you can perceive that now we are able to connect with the client at 192.168.10.2 via port 80.



Welcome to Hacking Articles

Dynamic SSH Tunneling through Kali Linux on Port 22

Now connect to the client machine through the given below command:

```
ssh -D 7000 ignite@192.168.1.104
```

```
root@kali:~# ssh -D 7000 ignite@192.168.1.104
ignite@192.168.1.104's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.13.0-36-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

268 packages can be updated.
3 updates are security updates.

Last login: Tue Mar 13 23:57:45 2018 from 192.168.1.105
ignite@ubuntu:~$
```

Install tsocks from the apt repository with the following command:

```
apt install tsocks
```

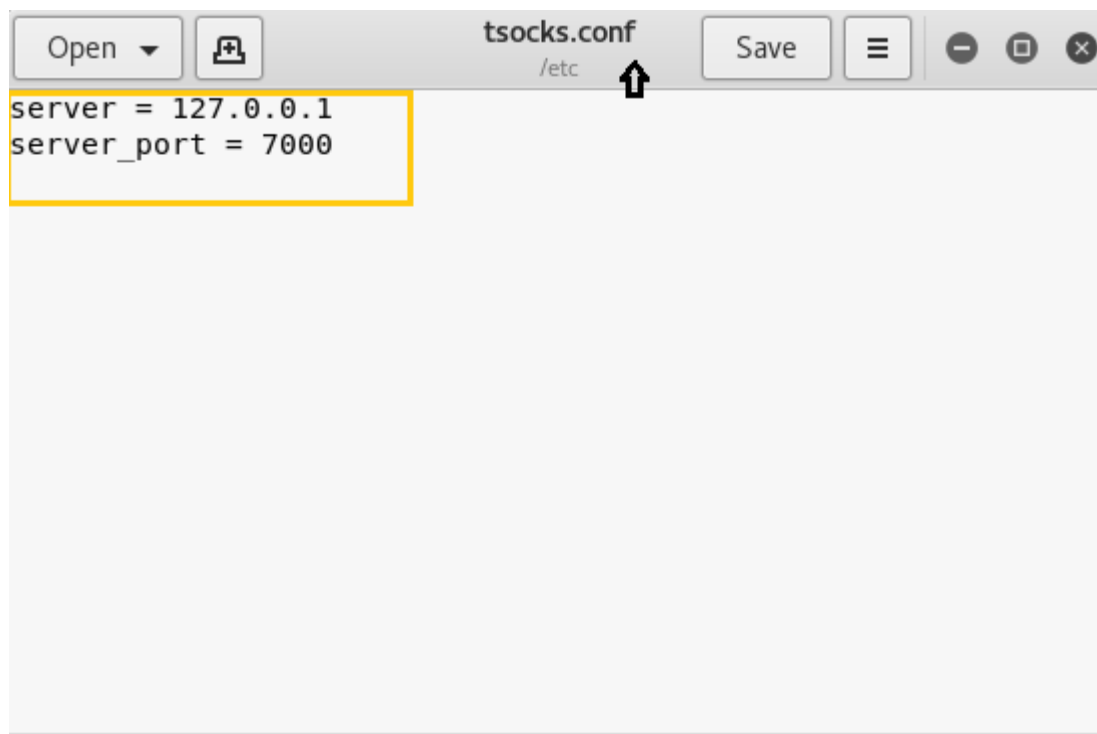

tsocks -Library for intercepting outgoing network connections and redirecting them through a SOCKS server.

```
root@kali:~# apt install tsocks ↵
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
gir1.2-networkmanager-1.0 gir1.2-nmgtk-1.0 gnome-themes-standard keepnote lib
libfreerdp-gdi1.1 libfreerdp-locale1.1 libfreerdp-primitives1.1 libfreerdp-ut
libpoppler68 libpoppler72 libproj12 libqgis-analysis2.14.21 libqgis-core2.14.
libradare2-2.3 libtxc-dxtn-s2tc libvpx4 libwinpr-crt0.1 libwinpr-crypto0.1 li
libwinpr-interlocked0.1 libwinpr-library0.1 libwinpr-path0.1 libwinpr-pool0.1
libx264-148 multiarch-support php7.0-mysql python-functools32 python-httprett
Use 'apt autoremove' to remove them.
The following NEW packages will be installed:
tsocks
```

Open the **tsocks.conf** file for editing socks server IP and port, in our case we need to mention below two lines and then save it.

Server = 127.0.0.1

Server_port = 7000



Now connect to the SSH client with the help tsocks using the given below command.

```
tsocks ssh raj@192.168.10.2
```

Enter the password and enjoy access to the SSH client.

```

root@kali:~# tsocks ssh raj@192.168.10.2
The authenticity of host '192.168.10.2 (192.168.10.2)' can't be established.
ECDSA key fingerprint is SHA256:JSfyM0DY2DlxXdaVStLUUx17WaTUIzqTKe0uKnCi1So.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.10.2' (ECDSA) to the list of known hosts.
raj@192.168.10.2's password:
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

Last login: Wed Mar 14 00:49:44 2018 from 192.168.10.1
raj@ignite:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:56:4f:2e
          inet addr:192.168.10.2  Bcast:192.168.10.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe56:4f2e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1675 errors:0 dropped:1 overruns:0 frame:0
          TX packets:582 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:215941 (215.9 KB)  TX bytes:97157 (97.1 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:373 errors:0 dropped:0 overruns:0 frame:0
          TX packets:373 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:29404 (29.4 KB)  TX bytes:29404 (29.4 KB)

```

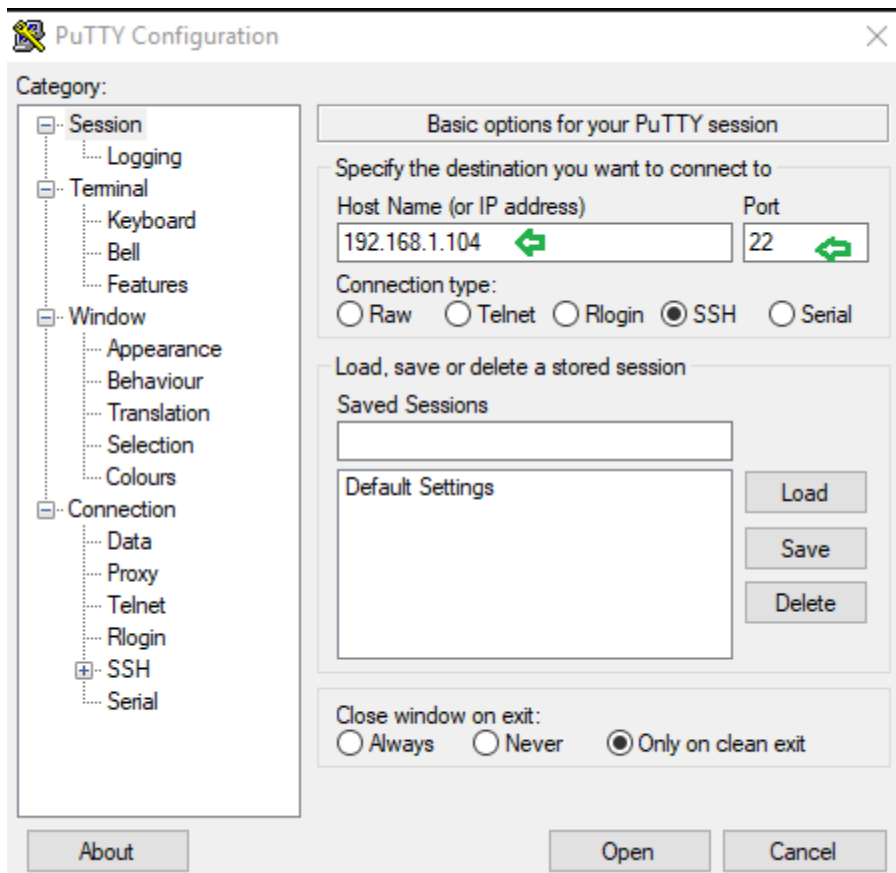
Local SSH Tunneling through Windows

Local tunneling is a process to access a specific SSH client machine for communication. It lets you establish a connection on a specific machine that is not connected to the internet.

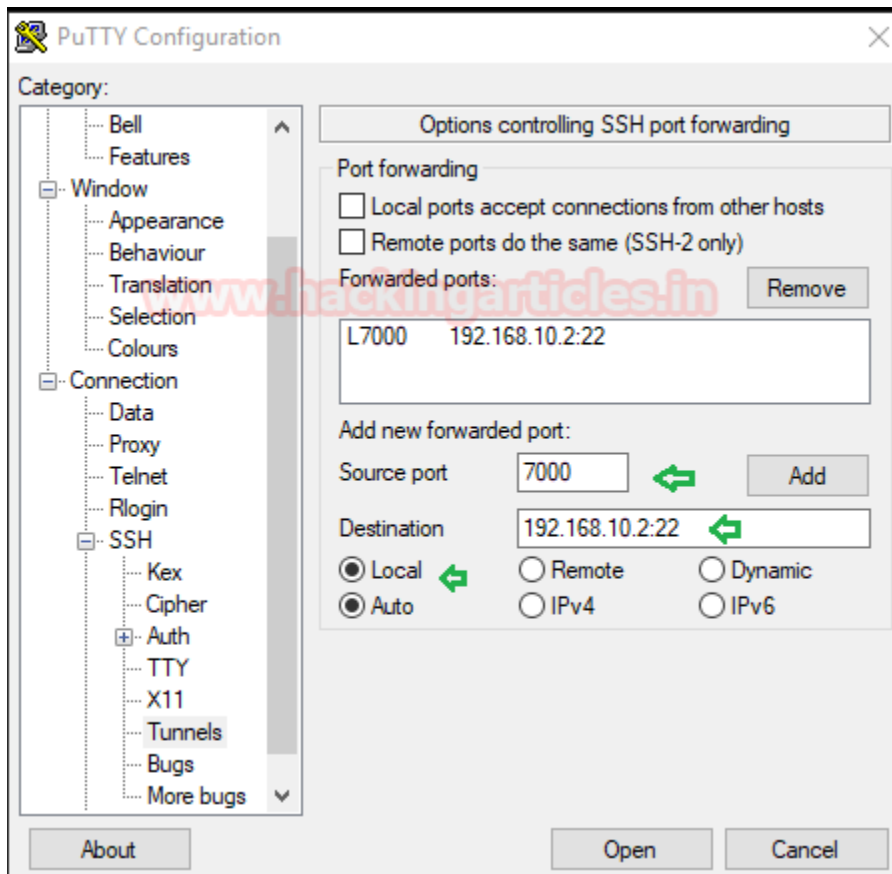
The only difference between dynamic tunneling and local tunneling is that dynamic tunneling requires a socks proxy for tunneling all TCP traffic, while local tunneling only requires a destination IP address.

Step for SSH Local tunneling

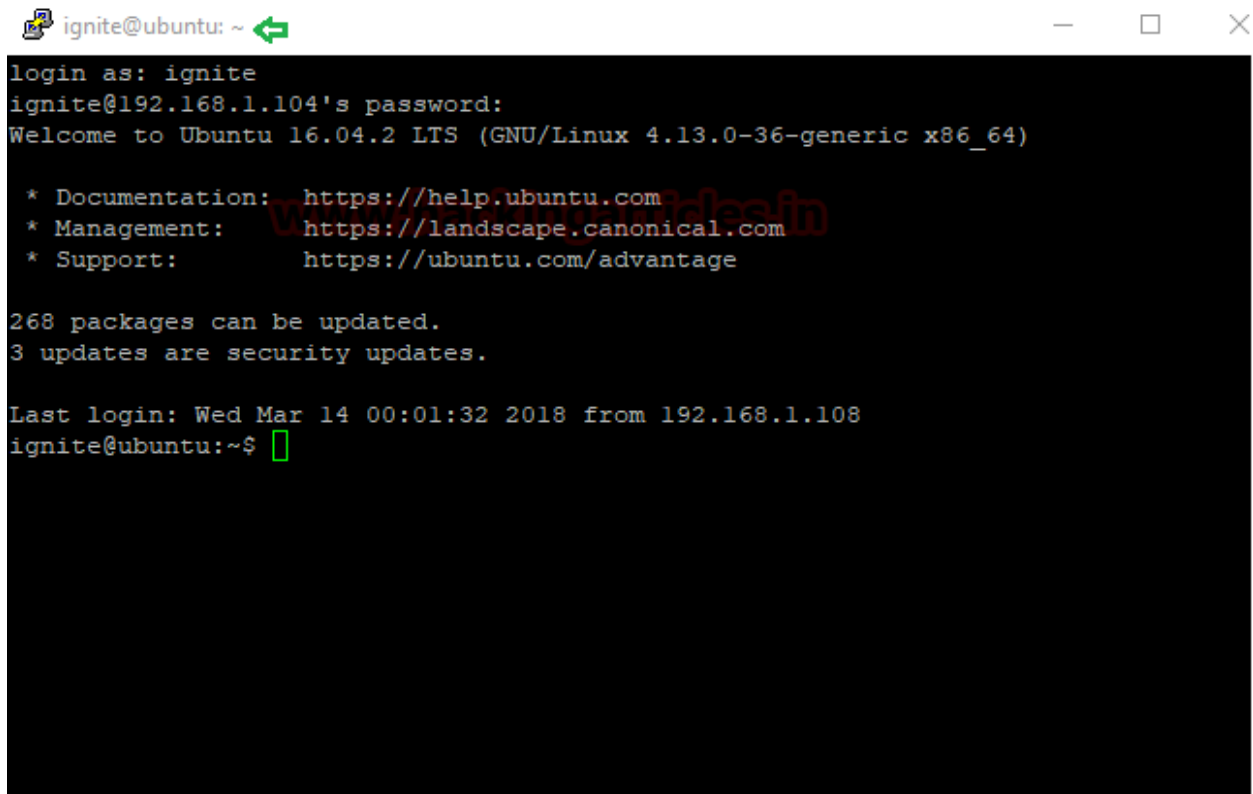
- Use putty to connect **SSH server (192.168.1.104)** via **port 22** and choose option **SSH > Tunnel** given in the left column of the category.



- Give new **port forwarded** as **7000** and connection type as **local**
- Destination address as **198.168.10.2:22** for establishing a connection with the specific client and click on ADD at last.
- Click on **open** when all things get set.



First, this will establish a connection between the remote PC and the SSH server.



A terminal window titled 'ignite@ubuntu: ~' with standard window controls. The terminal output shows a successful login for the user 'ignite' at IP 192.168.1.104. It displays the Ubuntu 16.04.2 LTS welcome message, system information (GNU/Linux 4.13.0-36-generic x86_64), and links for documentation, management, and support. It also shows that 268 packages can be updated, including 3 security updates. The last login was on Wednesday, March 14, 2018, at 00:01:32 from IP 192.168.1.108. The prompt 'ignite@ubuntu:~\$' is shown with a green cursor.

```
login as: ignite
ignite@192.168.1.104's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.13.0-36-generic x86_64)

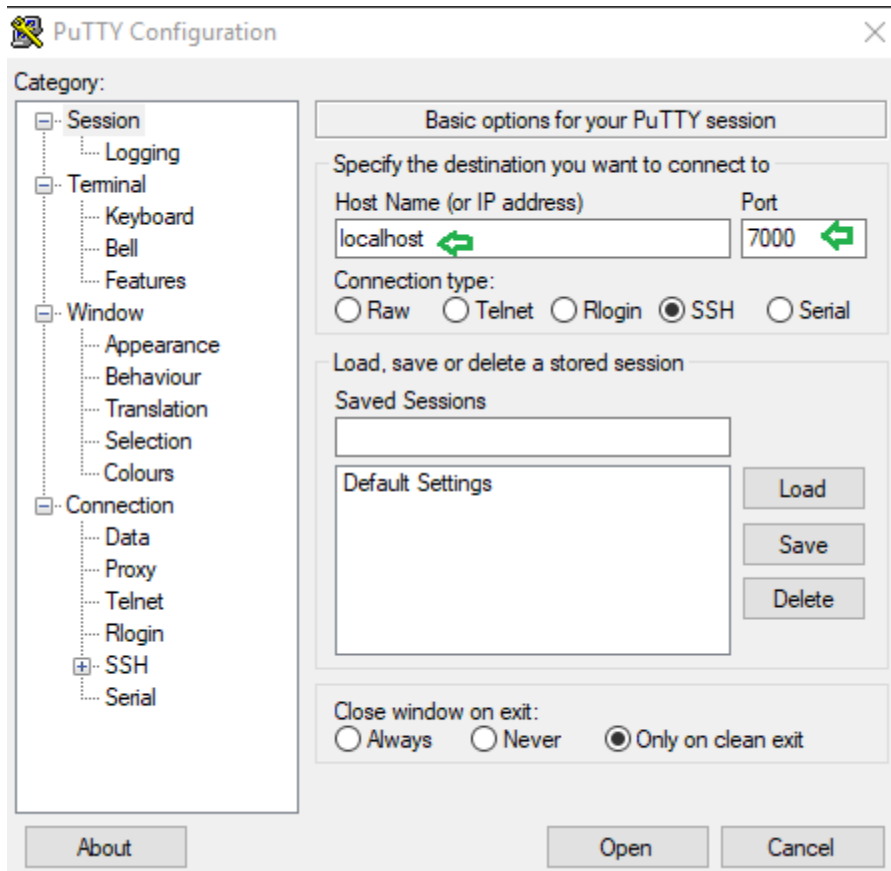
 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

268 packages can be updated.
3 updates are security updates.

Last login: Wed Mar 14 00:01:32 2018 from 192.168.1.108
ignite@ubuntu:~$
```

Open a new window of putty and follow given below step:

- Give hostname as **localhost** and port **7000** and connection type **SSH**.
- Click on **open** to establish a connection.



Awesome!! We have successfully access SSH client via port 7000


```
raj@ignite: ~  
login as: raj  
raj@localhost's password:  
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)  
  
* Documentation: https://help.ubuntu.com/  
  
Last login: Tue Mar 13 23:59:29 2018 from 192.168.10.1  
raj@ignite:~$ ifconfig  
eth0      Link encap:Ethernet  HWaddr 00:0c:29:56:4f:2e  
          inet addr:192.168.10.2  Bcast:192.168.10.255  Mask:255.255.255.0  
          inet6 addr: fe80::20c:29ff:fe56:4f2e/64 Scope:Link  
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1  
          RX packets:1274 errors:0 dropped:1 overruns:0 frame:0  
          TX packets:362 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:1000  
          RX bytes:170610 (170.6 KB)  TX bytes:63564 (63.5 KB)  
  
lo        Link encap:Local Loopback  
          inet addr:127.0.0.1  Mask:255.0.0.0  
          inet6 addr: ::1/128 Scope:Host  
          UP LOOPBACK RUNNING  MTU:65536  Metric:1  
          RX packets:353 errors:0 dropped:0 overruns:0 frame:0  
          TX packets:353 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:0
```

Local SSH Tunneling through Kali Linux

Now again, we switch into Kali Linux for local tunneling, which is quite easy as compared to dynamic. To forward ports to the local machine, run the command listed below.

```
ssh -L 7000:192.168.10.2:22 ignite@192.168.1.104
```

```
root@kali:~# ssh -L 7000:192.168.10.2:22 ignite@192.168.1.104  
ignite@192.168.1.104's password:  
bind: Address already in use  
channel_setup_fwd_listener_tcpip: cannot listen to port: 7000  
Could not request local forwarding.  
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.13.0-36-generic x86_64)  
  
* Documentation: https://help.ubuntu.com  
* Management:   https://landscape.canonical.com  
* Support:      https://ubuntu.com/advantage  
  
268 packages can be updated.  
3 updates are security updates.  
  
Last login: Wed Mar 14 00:10:31 2018 from 192.168.1.105  
ignite@ubuntu:~$
```

Now open a new terminal and type the below command to connect to the SSH client.

```
ssh raj@127.0.0.1 -p 7000
```

Awesome!! We have successfully accessed the SSH client via port 7000

```
root@kali:~# ssh raj@127.0.0.1 -p 7000 ↵
raj@127.0.0.1's password:
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

Last login: Wed Mar 14 00:11:24 2018 from 192.168.10.1
raj@ignite:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:56:4f:2e
          inet addr:192.168.10.2  Bcast:192.168.10.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe56:4f2e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1324 errors:0 dropped:1 overruns:0 frame:0
          TX packets:392 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:177194 (177.1 KB)  TX bytes:69433 (69.4 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:357 errors:0 dropped:0 overruns:0 frame:0
          TX packets:357 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:28044 (28.0 KB)  TX bytes:28044 (28.0 KB)
```

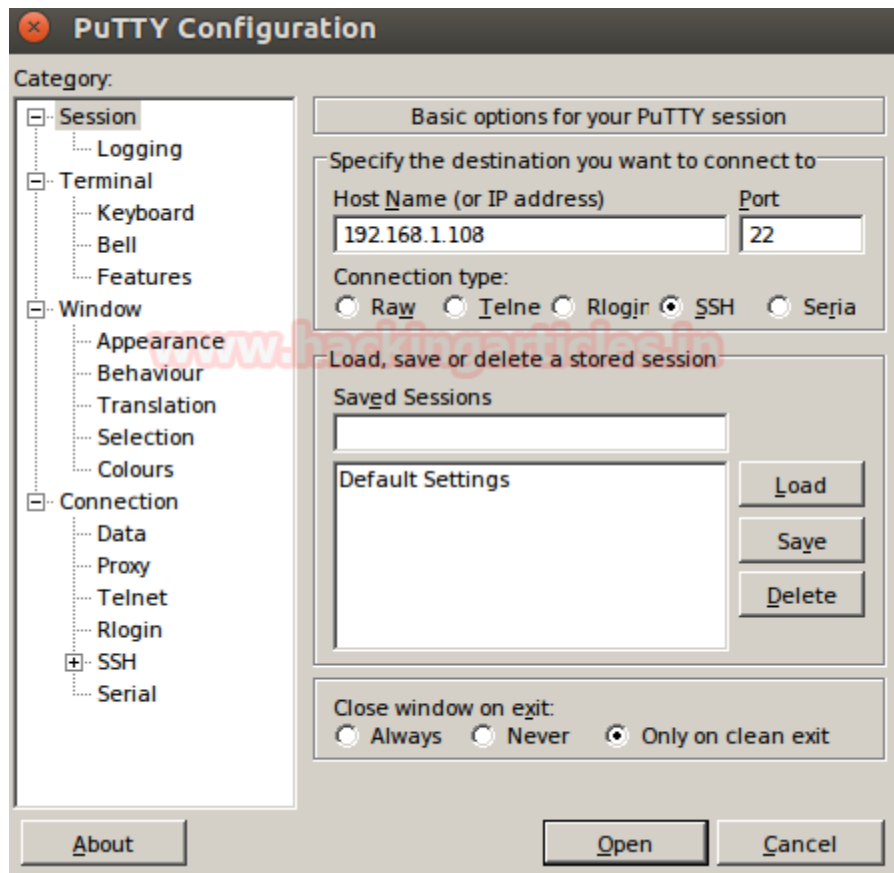
Remote SSH Tunneling through Putty

Remote tunneling is functional when a client machine wants to access a remote system that is outside of its network.

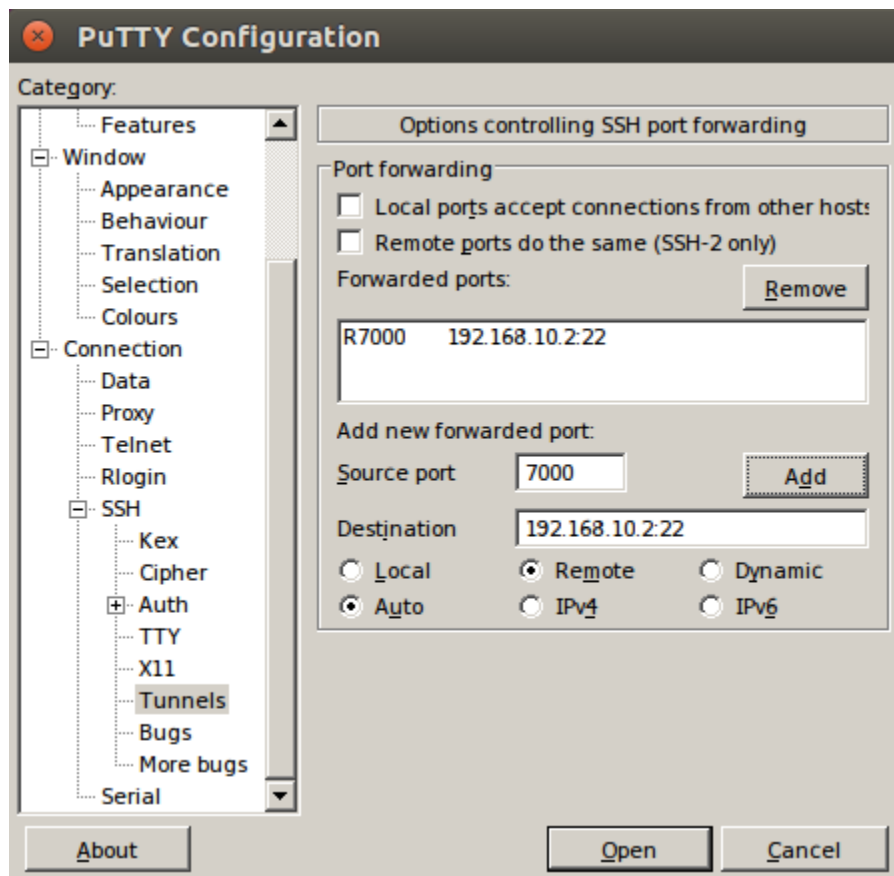
First, you need to install Putty on our SSH server (ignite) and then follow the given steps.

Step for remote tunneling

- Enter remote system IP **192.168.1.108**
- Mention port 22
- Go to SSH>tunnel options



- Give new **port forwarded** as **7000** and connection type as **Remote**
- Destination address as **198.168.10.2:22** for establishing a connection with the specific client and click on ADD at last.
- Click on **open** when all things get set.



Now the server will get connected to Remote system as shown in below image.

```
root@kali: ~  
login as: root  
root@192.168.1.108's password:  
The programs included with the Kali GNU/Linux system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.  
  
Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent  
permitted by applicable law.  
Last login: Wed Mar 14 03:37:51 2018 from 192.168.1.104  
  
RAJ@KALI:~$  
root@kali:~# ifconfig  
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 192.168.1.108 netmask 255.255.255.0 broadcast 192.168.1.255  
    inet6 fe80::20c:29ff:fe55:365a prefixlen 64 scopeid 0x20<link>  
    ether 00:0c:29:55:36:5a txqueuelen 1000 (Ethernet)  
    RX packets 49285 bytes 60407396 (57.6 MiB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 17970 bytes 1577306 (1.5 MiB)
```

Come back to the remote system and enter the following command to start with the SSH client machine.

```
ssh raj@127.0.0.1 -p 7000
```

From the given below image, you can observe that we have successfully connected with an SSH client machine via port 7000.

```

root@kali:~# ssh raj@127.0.0.1 -p 7000 ↵
raj@127.0.0.1's password:
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

Last login: Wed Mar 14 00:15:46 2018 from 192.168.10.1
raj@ignite:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:56:4f:2e
          inet addr:192.168.10.2  Bcast:192.168.10.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe56:4f2e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1403 errors:0 dropped:1 overruns:0 frame:0
          TX packets:423 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:186936 (186.9 KB)  TX bytes:75270 (75.2 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:361 errors:0 dropped:0 overruns:0 frame:0
          TX packets:361 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:28384 (28.3 KB)  TX bytes:28384 (28.3 KB)

raj@ignite:~$ █

```

Remote SSH Tunneling through Ubuntu

If you are not willing to use putty for remote tunneling, then you can execute the following command.

```
ssh -R 7000:192.168.10.2:22 root@192.168.1.108
```

Here, 192.168.1.10.2 is our local client (raj) IP and 192.168.1.108 is our remote system IP.

```

ignite@ubuntu:~$ ssh -R 7000:192.168.10.2:22 root@192.168.1.108 ↵
root@192.168.1.108's password:

The programs included with the Kali GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Mar 15 13:29:11 2018 from 192.168.1.111

```

Come back to the remote system and enter the following command to start with the SSH client machine.

```
ssh raj@127.0.0.1 -p 7000
```


From the given below image, you can observe that we have successfully connected with an SSH client machine via port 7000.

```
root@kali:~# ssh raj@127.0.0.1 -p 7000 ↵
raj@127.0.0.1's password:
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
www.hackingarticles.in
Last login: Wed Mar 14 00:15:46 2018 from 192.168.10.1
raj@ignite:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:56:4f:2e
          inet addr:192.168.10.2  Bcast:192.168.10.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe56:4f2e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1403 errors:0 dropped:1 overruns:0 frame:0
          TX packets:423 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:186936 (186.9 KB)  TX bytes:75270 (75.2 KB)
www.hackingarticles.in
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:361 errors:0 dropped:0 overruns:0 frame:0
          TX packets:361 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:28384 (28.3 KB)  TX bytes:28384 (28.3 KB)

raj@ignite:~$ █
```

JOIN OUR TRAINING PROGRAMS

