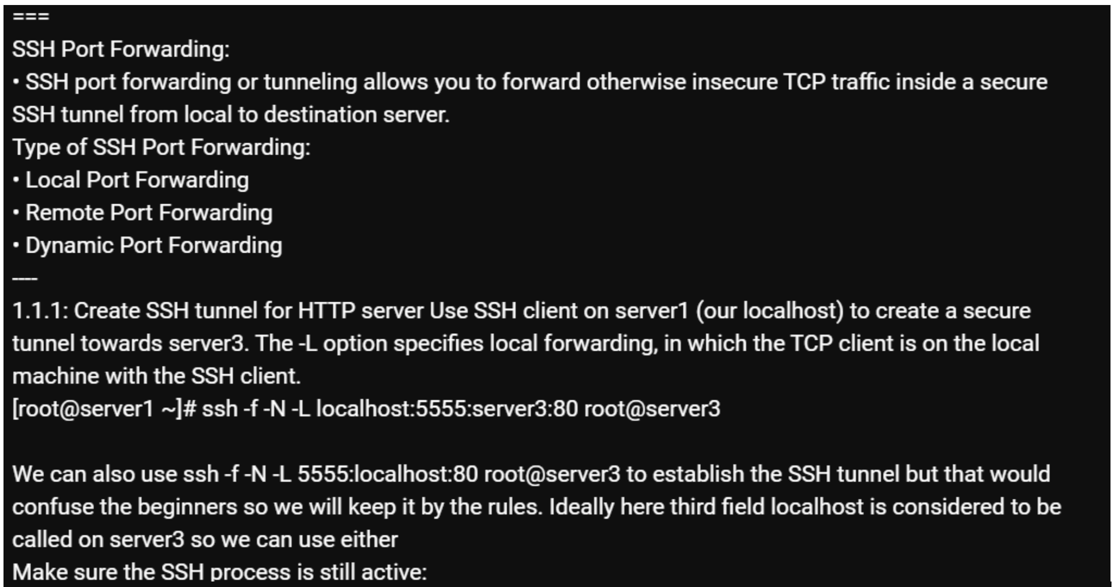
SSH Port Forwarding In Linux | SSH Tunneling in RHEL | Local, Remote & Dynamic Port Forwarding

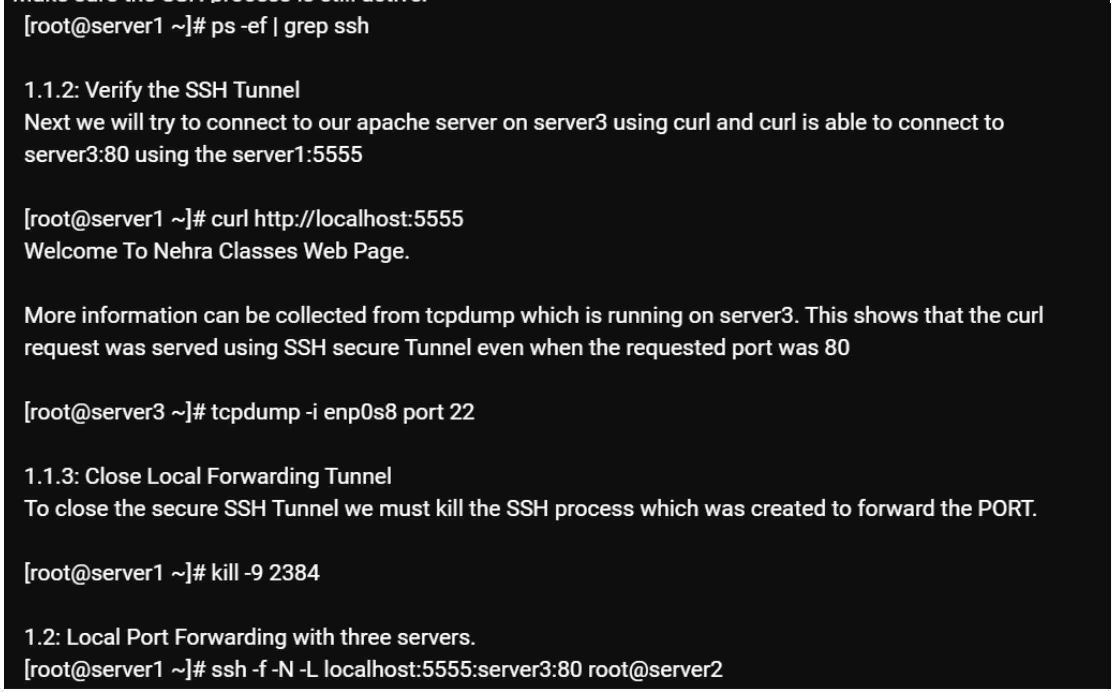


===
SSH Port Forwarding:
• SSH port forwarding or tunneling allows you to forward otherwise insecure TCP traffic inside a secure SSH tunnel from local to destination server.
Type of SSH Port Forwarding:
• Local Port Forwarding
• Remote Port Forwarding
• Dynamic Port Forwarding

1.1.1: Create SSH tunnel for HTTP server Use SSH client on server1 (our localhost) to create a secure tunnel towards server3. The -L option specifies local forwarding, in which the TCP client is on the local machine with the SSH client.

```
[root@server1 ~]# ssh -f -N -L localhost:5555:server3:80 root@server3
```

We can also use ssh -f -N -L 5555:localhost:80 root@server3 to establish the SSH tunnel but that would confuse the beginners so we will keep it by the rules. Ideally here third field localhost is considered to be called on server3 so we can use either
Make sure the SSH process is still active:



```
[root@server1 ~]# ps -ef | grep ssh
```

1.1.2: Verify the SSH Tunnel
Next we will try to connect to our apache server on server3 using curl and curl is able to connect to server3:80 using the server1:5555

```
[root@server1 ~]# curl http://localhost:5555
Welcome To Nehra Classes Web Page.
```

More information can be collected from tcpdump which is running on server3. This shows that the curl request was served using SSH secure Tunnel even when the requested port was 80

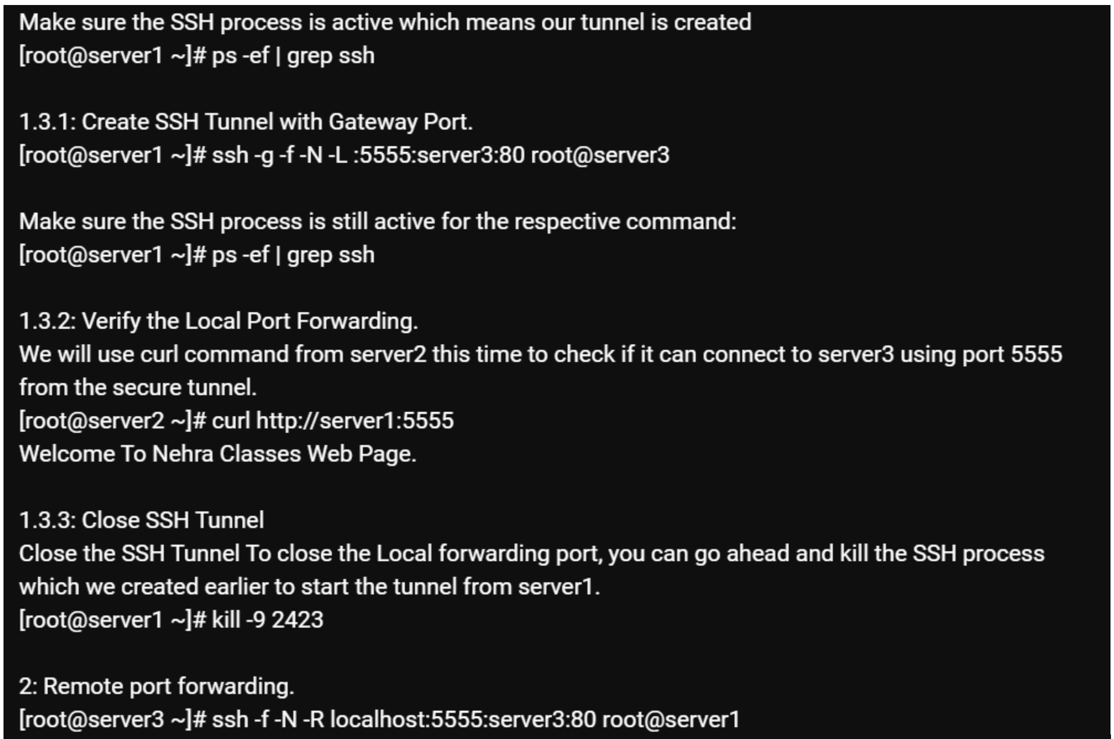
```
[root@server3 ~]# tcpdump -i enp0s8 port 22
```

1.1.3: Close Local Forwarding Tunnel
To close the secure SSH Tunnel we must kill the SSH process which was created to forward the PORT.

```
[root@server1 ~]# kill -9 2384
```

1.2: Local Port Forwarding with three servers.

```
[root@server1 ~]# ssh -f -N -L localhost:5555:server3:80 root@server2
```


Make sure the SSH process is active which means our tunnel is created

```
[root@server1 ~]# ps -ef | grep ssh
```

1.3.1: Create SSH Tunnel with Gateway Port.

```
[root@server1 ~]# ssh -g -f -N -L :5555:server3:80 root@server3
```

Make sure the SSH process is still active for the respective command:

```
[root@server1 ~]# ps -ef | grep ssh
```

1.3.2: Verify the Local Port Forwarding.

We will use curl command from server2 this time to check if it can connect to server3 using port 5555 from the secure tunnel.

```
[root@server2 ~]# curl http://server1:5555
Welcome To Nehra Classes Web Page.
```

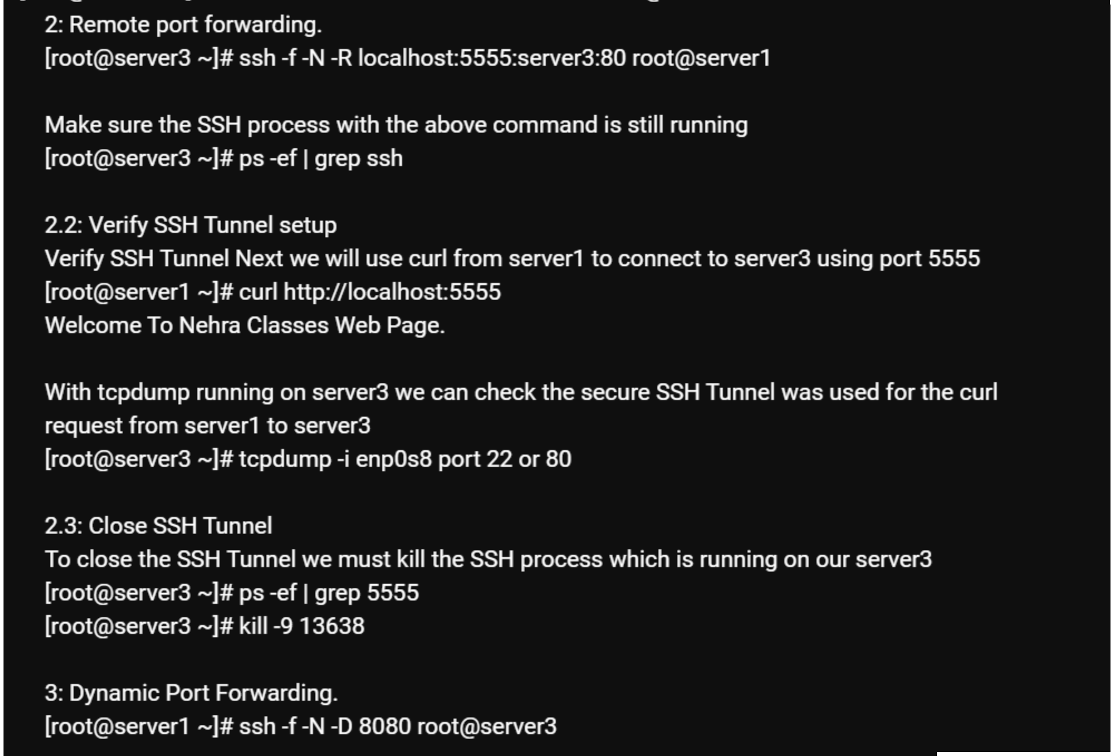
1.3.3: Close SSH Tunnel

Close the SSH Tunnel To close the Local forwarding port, you can go ahead and kill the SSH process which we created earlier to start the tunnel from server1.

```
[root@server1 ~]# kill -9 2423
```

2: Remote port forwarding.

```
[root@server3 ~]# ssh -f -N -R localhost:5555:server3:80 root@server1
```


2: Remote port forwarding.

```
[root@server3 ~]# ssh -f -N -R localhost:5555:server3:80 root@server1
```

Make sure the SSH process with the above command is still running

```
[root@server3 ~]# ps -ef | grep ssh
```

2.2: Verify SSH Tunnel setup

Verify SSH Tunnel Next we will use curl from server1 to connect to server3 using port 5555

```
[root@server1 ~]# curl http://localhost:5555
Welcome To Nehra Classes Web Page.
```

With tcpdump running on server3 we can check the secure SSH Tunnel was used for the curl request from server1 to server3

```
[root@server3 ~]# tcpdump -i enp0s8 port 22 or 80
```

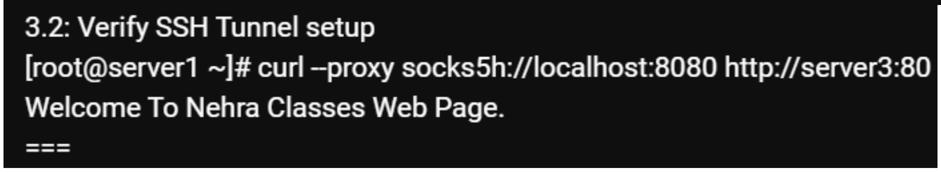
2.3: Close SSH Tunnel

To close the SSH Tunnel we must kill the SSH process which is running on our server3

```
[root@server3 ~]# ps -ef | grep 5555
[root@server3 ~]# kill -9 13638
```

3: Dynamic Port Forwarding.

```
[root@server1 ~]# ssh -f -N -D 8080 root@server3
```

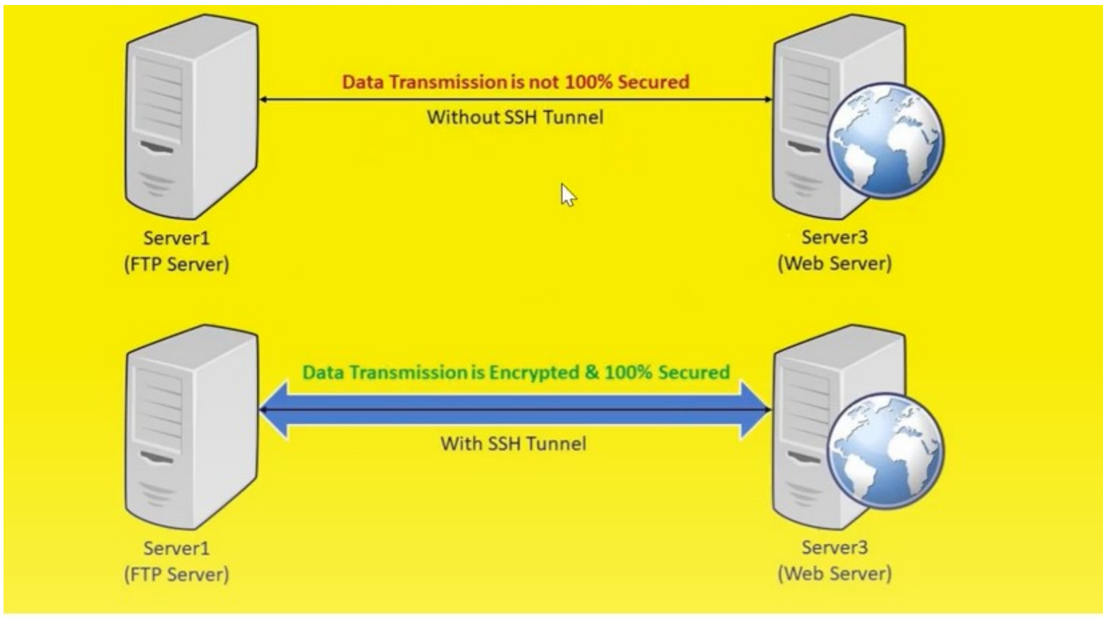

3.2: Verify SSH Tunnel setup

```
[root@server1 ~]# curl --proxy socks5h://localhost:8080 http://server3:80
Welcome To Nehra Classes Web Page.
```

===

SSH Port Forwarding:

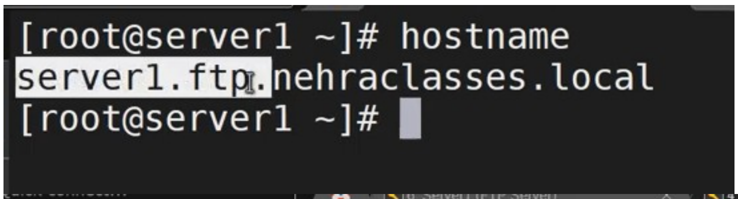
- SSH port forwarding or tunnelling allows you to forward otherwise insecure TCP traffic inside a **secure SSH tunnel** from local to destination server.
- Protocols such as FTP, POP3, SMTP, HTTP, TELNET, and others can all be forwarded inside this SSH tunnel.
- This will provide increased security features such as encryption and authentication that may not otherwise be supported.
- You must **create a new SSH connection to establish the tunnelling**.



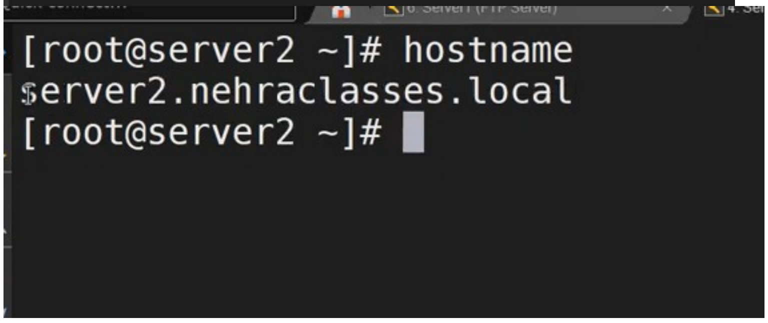
Type of SSH Port Forwarding?

There are three types of port forwarding mechanisms between local and remote host:

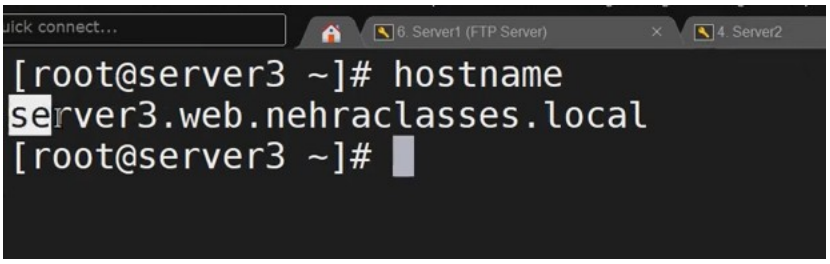
- **Local Port Forwarding:** Create a local port that is connected to a remote service.
- **Remote Port Forwarding:** Forward a port on a remote server on the Internet to a local port.
- **Dynamic Port Forwarding:** A SOCKS client connects via TCP, and indicates via the protocol the remote socket it wants to reach.



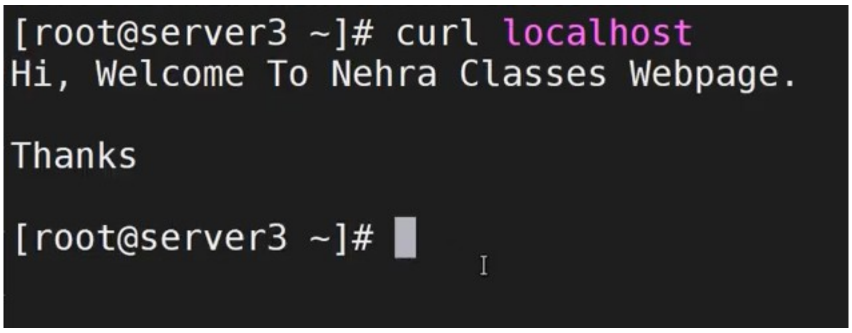
```
[root@server1 ~]# hostname
server1.ftp.nehraclasses.local
[root@server1 ~]#
```



```
[root@server2 ~]# hostname
server2.nehraclasses.local
[root@server2 ~]#
```



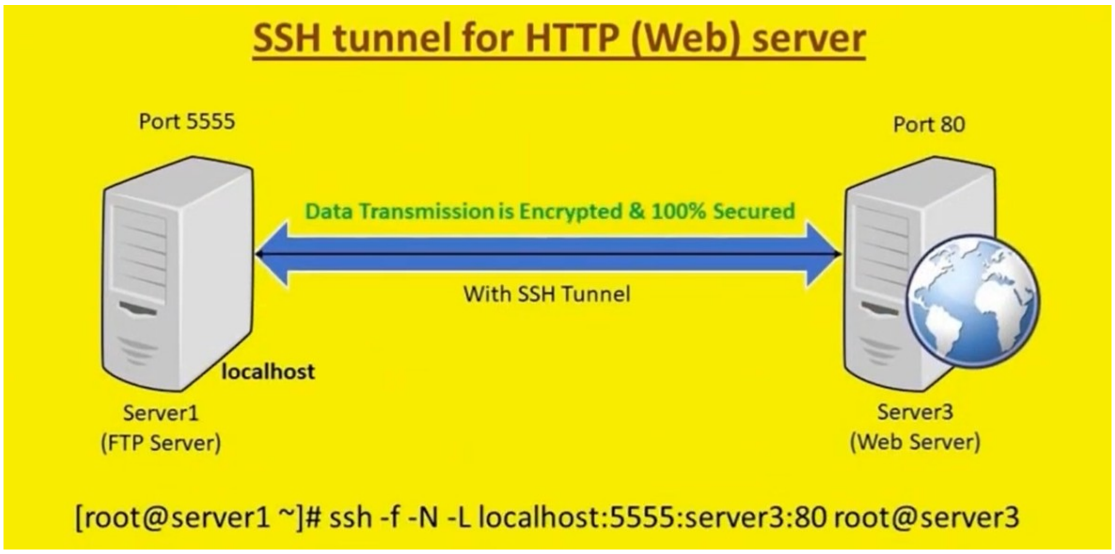
```
[root@server3 ~]# hostname
server3.web.nehraclasses.local
[root@server3 ~]#
```



```
[root@server3 ~]# curl localhost
Hi, Welcome To Nehra Classes Webpage.

Thanks

[root@server3 ~]#
```

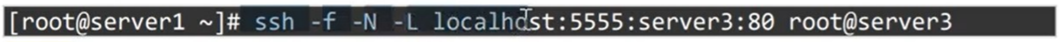
1: Local Port Forwarding

- This allows you to forward a port from your localhost server (`ssh_client`) to a port on target remote server (`ssh_server`).
- The basic **syntax** would be `ssh -L sourceHost:sourcePort:forwardToHost:onPort connectToHost`
- Here the **first field** `sourceHost` would be the localhost using on which you will enable the Port Forward
- The **second field** is `sourcePort` using which you will connect to the `remoteHost` and `remotePort`
- The **third field** is the `forwardToHost` i.e. the server to which you want to forward the request. You can also put localhost in this field (as it is the localhost of `remoteHost`)
- The **fourth field** is the `onPort` section i.e. the port to which the request has to be


SSH tunnel for HTTP (Web) server
Port 5555
Port 80
Data Transmission is Encrypted & 100% Secured
With SSH Tunnel
localhost
Server1
(FTP Server)
Server3
(Web Server)
[root@server1 ~]# ssh -f -N -L localhost:5555:server3:80 root@server3

1.1.1: Create SSH tunnel for HTTP server

Use SSH client on `server1` (our localhost) to create a secure tunnel towards `server3`. The –`L` option specifies local forwarding, in which the TCP client is on the local machine with the SSH client.

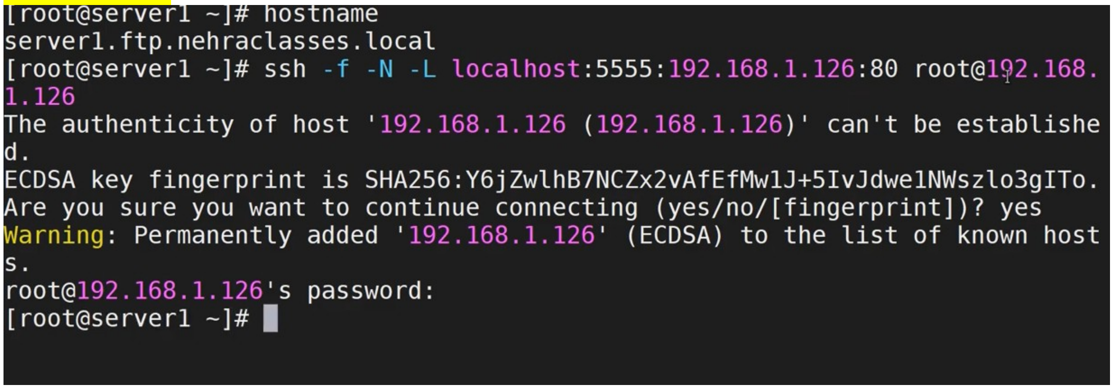


```
[root@server1 ~]# ssh -f -N -L localhost:5555:server3:80 root@server3
```

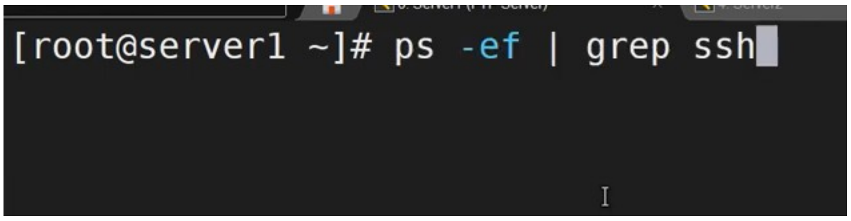
HINT:

We can also use `ssh -f -N -L 5555:localhost:80 root@server3` to establish the SSH tunnel but that would confuse the beginners so we will keep it by the rules. Ideally here third field `localhost` is considered to be called on `server3` so we can use either

#ssh to server3

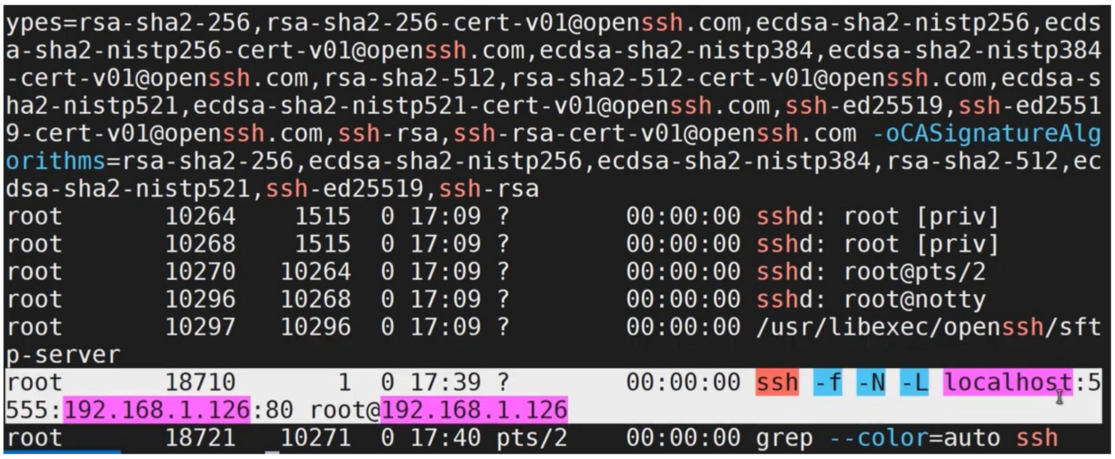


```
[root@server1 ~]# hostname
server1.ftp.nehraclasses.local
[root@server1 ~]# ssh -f -N -L localhost:5555:192.168.1.126:80 root@192.168.
1.126
The authenticity of host '192.168.1.126 (192.168.1.126)' can't be establishe
d.
ECDSA key fingerprint is SHA256:Y6jZwlhB7NCZx2vAfEfMw1J+5IvJdwe1NWszlo3gITo.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.1.126' (ECDSA) to the list of known host
s.
root@192.168.1.126's password:
[root@server1 ~]#
```

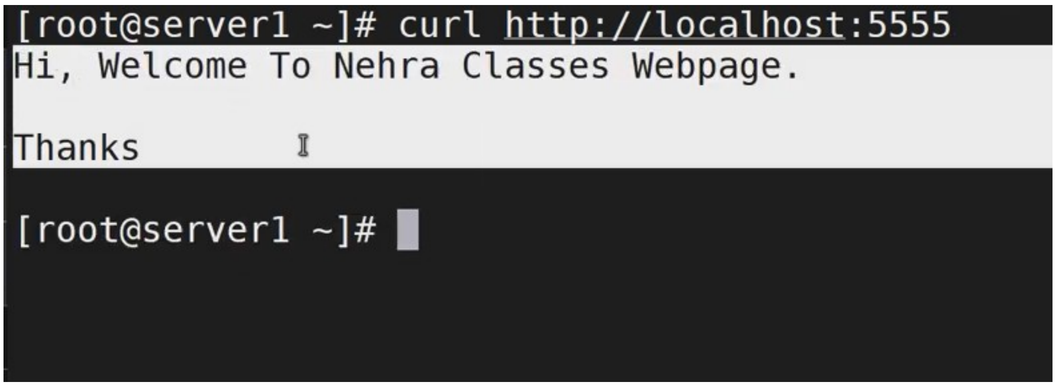


```
[root@server1 ~]# ps -ef | grep ssh
```

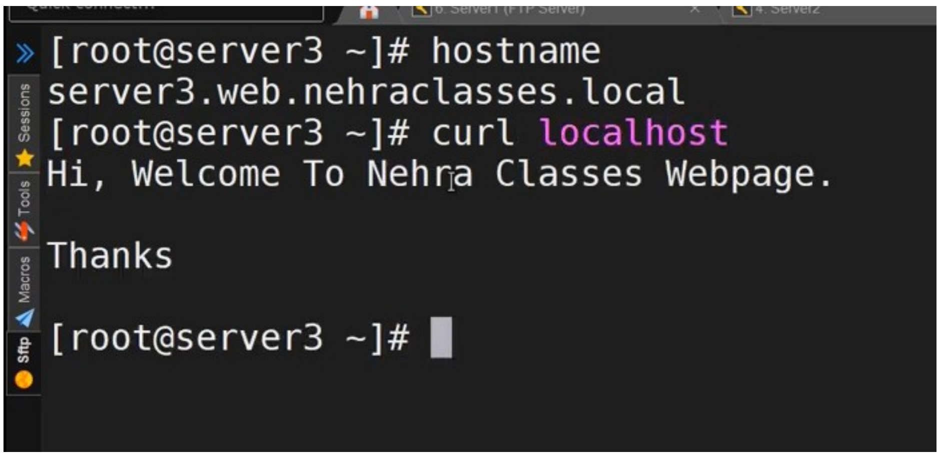
# port Forwarding



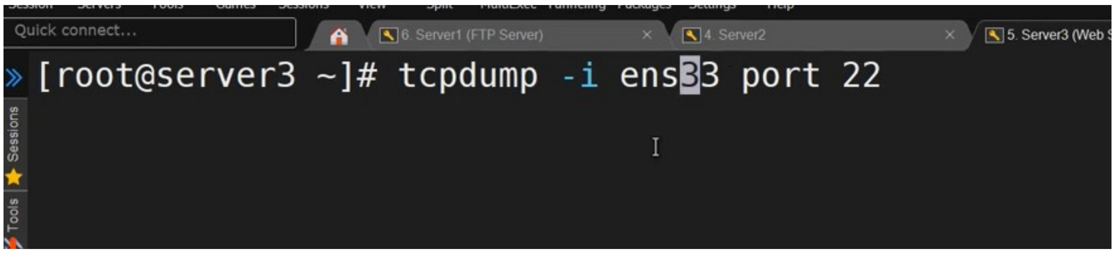
```
ypes=rsa-sha2-256,rsa-sha2-256-cert-v01@openssh.com,ecdsa-sha2-nistp256,ecds
a-sha2-nistp256-cert-v01@openssh.com,ecdsa-sha2-nistp384,ecdsa-sha2-nistp384
-cert-v01@openssh.com,rsa-sha2-512,rsa-sha2-512-cert-v01@openssh.com,ecdsa-s
ha2-nistp521,ecdsa-sha2-nistp521-cert-v01@openssh.com,ssh-ed25519,ssh-ed2551
9-cert-v01@openssh.com,ssh-rsa,ssh-rsa-cert-v01@openssh.com -oCASignatureAlg
orithms=rsa-sha2-256,ecdsa-sha2-nistp256,ecdsa-sha2-nistp384,rsa-sha2-512,ec
dsa-sha2-nistp521,ssh-ed25519,ssh-rsa
root       10264   1515  0 17:09 ?        00:00:00 sshd: root [priv]
root       10268   1515  0 17:09 ?        00:00:00 sshd: root [priv]
root       10270  10264  0 17:09 ?        00:00:00 sshd: root@pts/2
root       10296  10268  0 17:09 ?        00:00:00 sshd: root@notty
root       10297  10296  0 17:09 ?        00:00:00 /usr/libexec/openssh/sft
p-server
root       18710      1  0 17:39 ?        00:00:00 ssh -f -N -L localhost:5
555:192.168.1.126:80 root@192.168.1.126
root       18721  10271  0 17:40 pts/2    00:00:00 grep --color=auto ssh
```



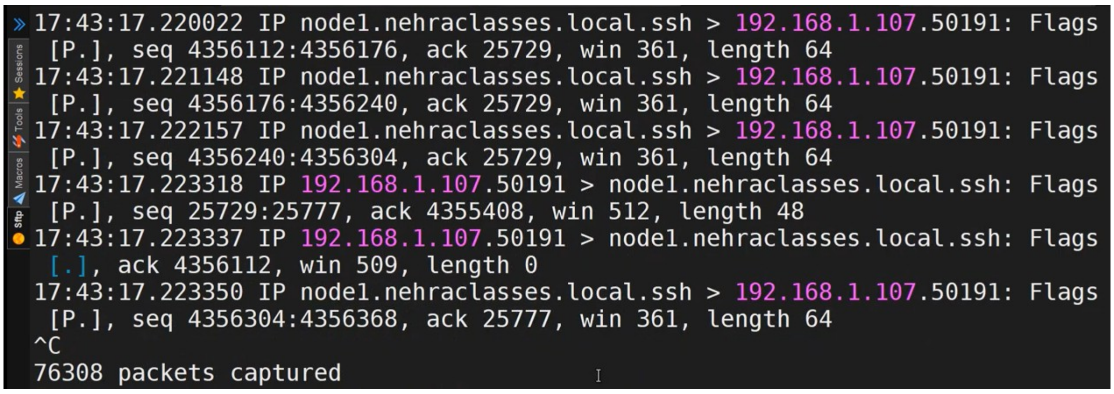
```
[root@server1 ~]# curl http://localhost:5555
Hi, Welcome To Nehra Classes Webpage.

Thanks

[root@server1 ~]#
```



```
[root@server3 ~]# hostname
server3.web.nehraclasses.local
[root@server3 ~]# curl localhost
Hi, Welcome To Nehra Classes Webpage.

Thanks

[root@server3 ~]#
```

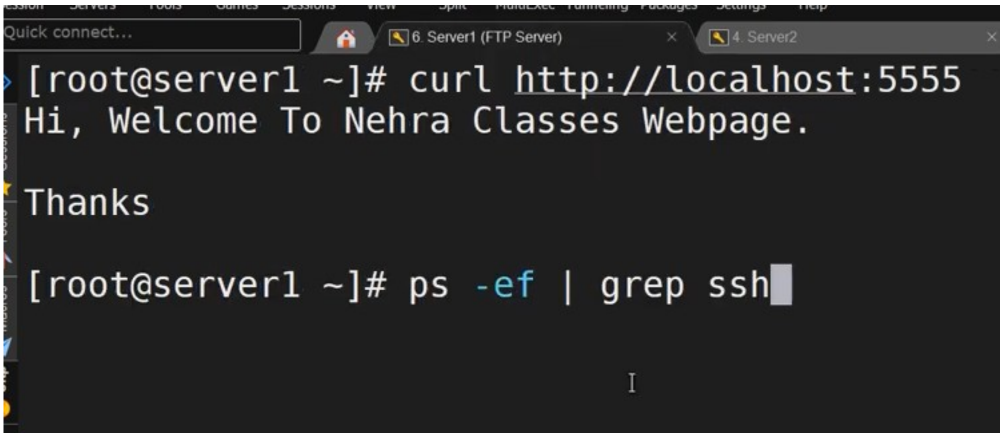


```
[root@server3 ~]# tcpdump -i ens33 port 22
```

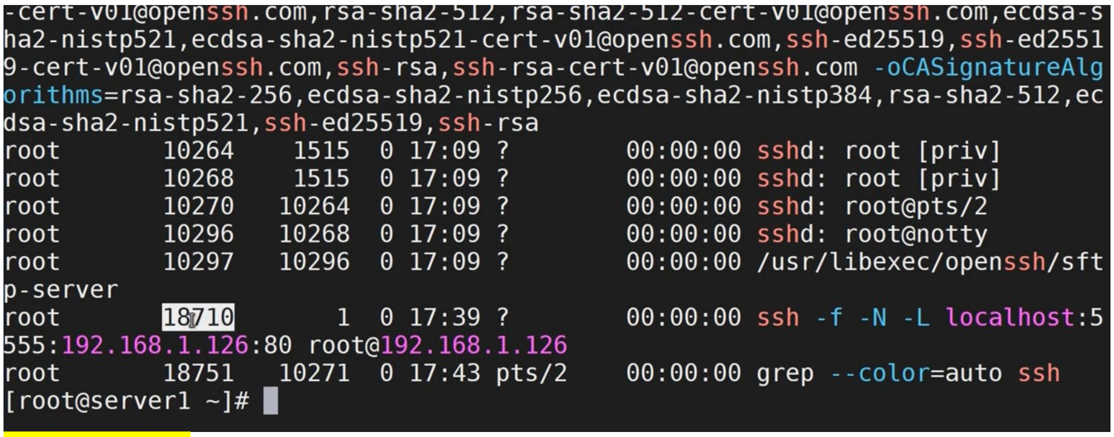


```
17:43:17.220022 IP node1.nehraclasses.local.ssh > 192.168.1.107.50191: Flags
  [P.], seq 4356112:4356176, ack 25729, win 361, length 64
17:43:17.221148 IP node1.nehraclasses.local.ssh > 192.168.1.107.50191: Flags
  [P.], seq 4356176:4356240, ack 25729, win 361, length 64
17:43:17.222157 IP node1.nehraclasses.local.ssh > 192.168.1.107.50191: Flags
  [P.], seq 4356240:4356304, ack 25729, win 361, length 64
17:43:17.223318 IP 192.168.1.107.50191 > node1.nehraclasses.local.ssh: Flags
  [P.], seq 25729:25777, ack 4355408, win 512, length 48
17:43:17.223337 IP 192.168.1.107.50191 > node1.nehraclasses.local.ssh: Flags
  [.], ack 4356112, win 509, length 0
17:43:17.223350 IP node1.nehraclasses.local.ssh > 192.168.1.107.50191: Flags
  [P.], seq 4356304:4356368, ack 25777, win 361, length 64
^C
76308 packets captured
```

# how to close server 1

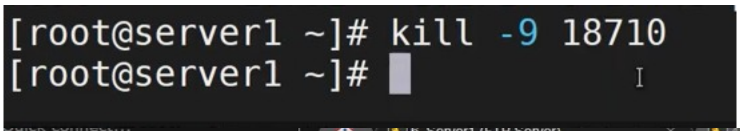


```
[root@server1 ~]# curl http://localhost:5555
Hi, Welcome To Nehra Classes Webpage.

Thanks

[root@server1 ~]# ps -ef | grep ssh
```

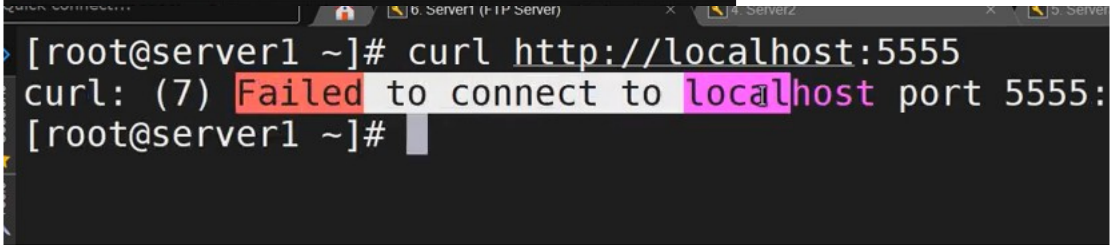


```
-cert-v01@openssh.com,rsa-sha2-512,rsa-sha2-512-cert-v01@openssh.com,ecdsa-s
ha2-nistp521,ecdsa-sha2-nistp521-cert-v01@openssh.com,ssh-ed25519,ssh-ed2551
9-cert-v01@openssh.com,ssh-rsa,ssh-rsa-cert-v01@openssh.com -oCASignatureAlg
orithms=rsa-sha2-256,ecdsa-sha2-nistp256,ecdsa-sha2-nistp384,rsa-sha2-512,ec
dsa-sha2-nistp521,ssh-ed25519,ssh-rsa
root      10264   1515  0 17:09 ?        00:00:00 sshd: root [priv]
root      10268   1515  0 17:09 ?        00:00:00 sshd: root [priv]
root      10270  10264  0 17:09 ?        00:00:00 sshd: root@pts/2
root      10296  10268  0 17:09 ?        00:00:00 sshd: root@notty
root      10297  10296  0 17:09 ?        00:00:00 /usr/libexec/openssh/sft
p-server
root      18710      1  0 17:39 ?        00:00:00 ssh -f -N -L localhost:5
555:192.168.1.126:80 root@192.168.1.126
root      18751  10271  0 17:43 pts/2    00:00:00 grep --color=auto ssh
[root@server1 ~]#
```

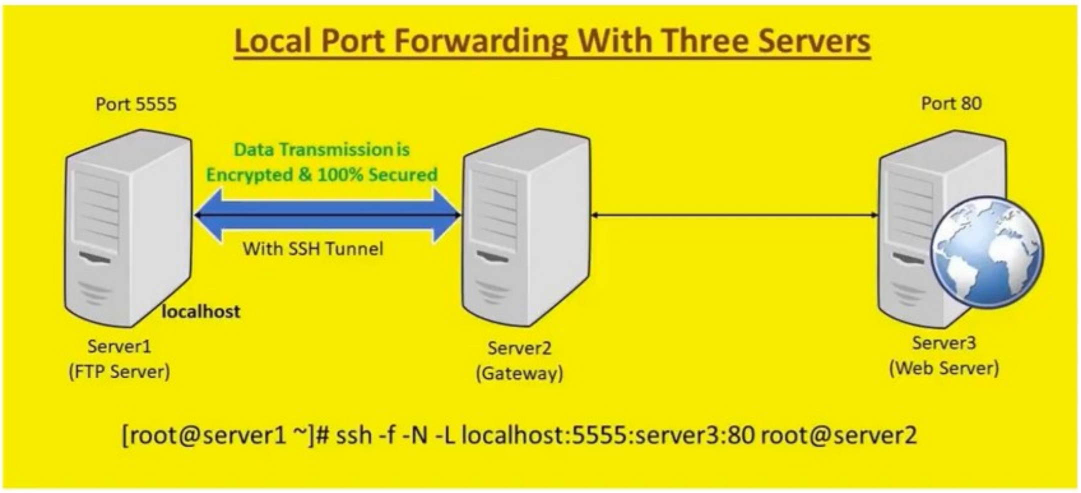
# Kill the process



```
[root@server1 ~]# kill -9 18710
[root@server1 ~]#
```



```
[root@server1 ~]# curl http://localhost:5555
curl: (7) Failed to connect to localhost port 5555:
[root@server1 ~]#
```


Local Port Forwarding With Three Servers
Port 5555
Data Transmission is Encrypted & 100% Secured
With SSH Tunnel
Port 80
localhost
Server1 (FTP Server)
Server2 (Gateway)
Server3 (Web Server)
[root@server1 ~]# ssh -f -N -L localhost:5555:server3:80 root@server2

1.2.1: Create SSH Tunnel

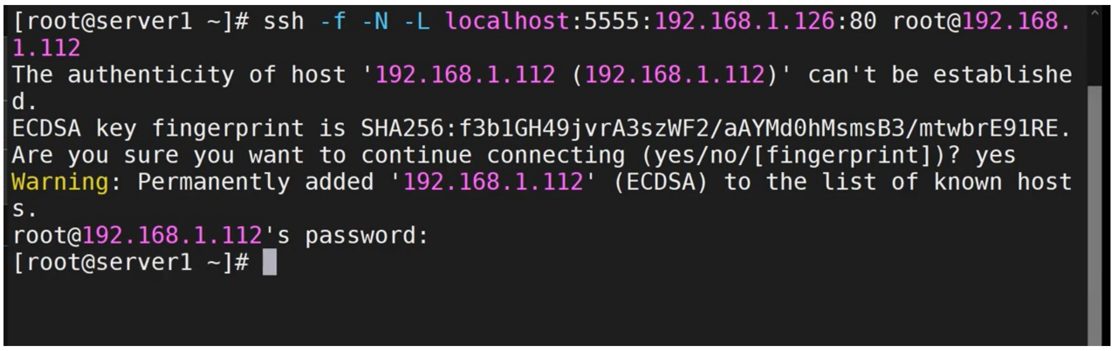
Create Local Forwarding port on your localhost (`server1`) using SSH client. You can also ignore mentioning localhost in this command as that is the default behaviour, I have written here just for the sake of explanation

```
[root@server1 ~]# ssh -f -N -L localhost:5555:server3:80 root@server2
```

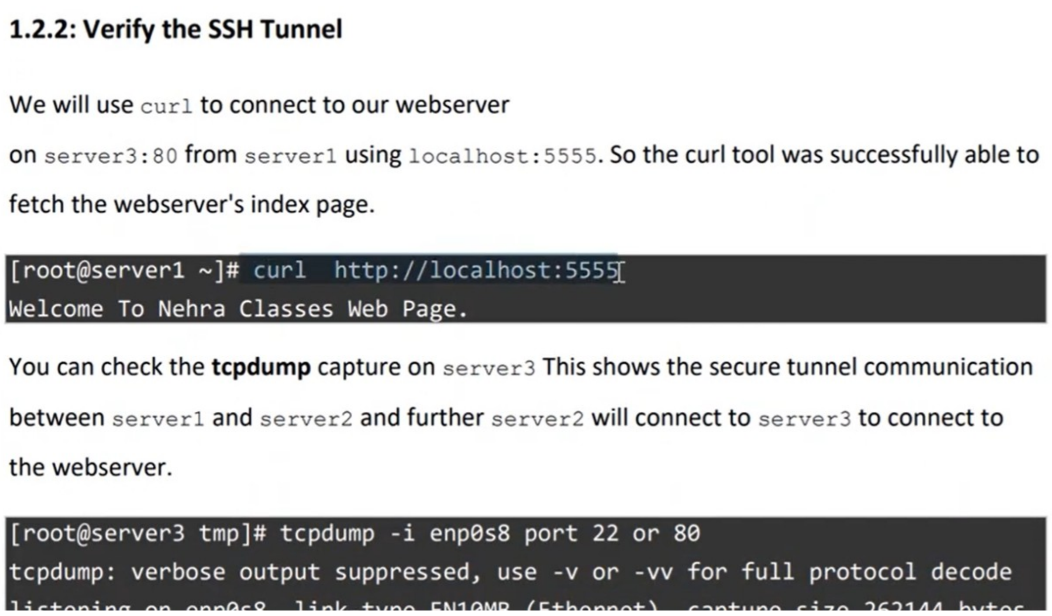
Make sure the SSH process is active which means our **tunnel is created**

```
[root@server1 ~]# ps -ef | grep ssh
root      1170     1  0 10:20 ?        00:00:00 /usr/sbin/sshd -D
root      1426  1170  0 10:21 ?        00:00:01 sshd: root@pts/1
```

# connect server 1 to Server 2



```
[root@server1 ~]# ssh -f -N -L localhost:5555:192.168.1.126:80 root@192.168.
1.112
The authenticity of host '192.168.1.112 (192.168.1.112)' can't be establishe
d.
ECDSA key fingerprint is SHA256:f3b1GH49jvrA3szWF2/aAYMd0hMsmsB3/mtwbrE91RE.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.1.112' (ECDSA) to the list of known host
s.
root@192.168.1.112's password:
[root@server1 ~]#
```



1.2.2: Verify the SSH Tunnel

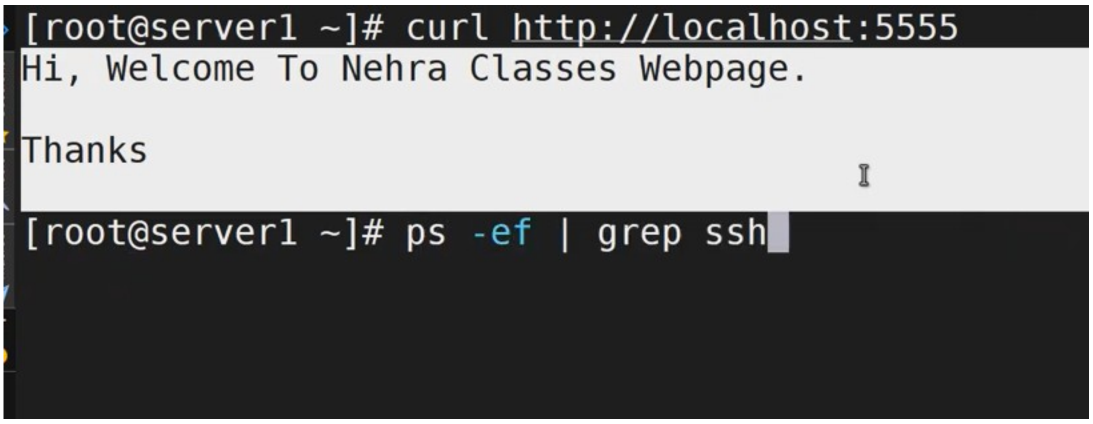
We will use `curl` to connect to our webserver

on `server3:80` from `server1` using `localhost:5555`. So the curl tool was successfully able to fetch the webserver's index page.

```
[root@server1 ~]# curl  http://localhost:5555
Welcome To Nehra Classes Web Page.
```

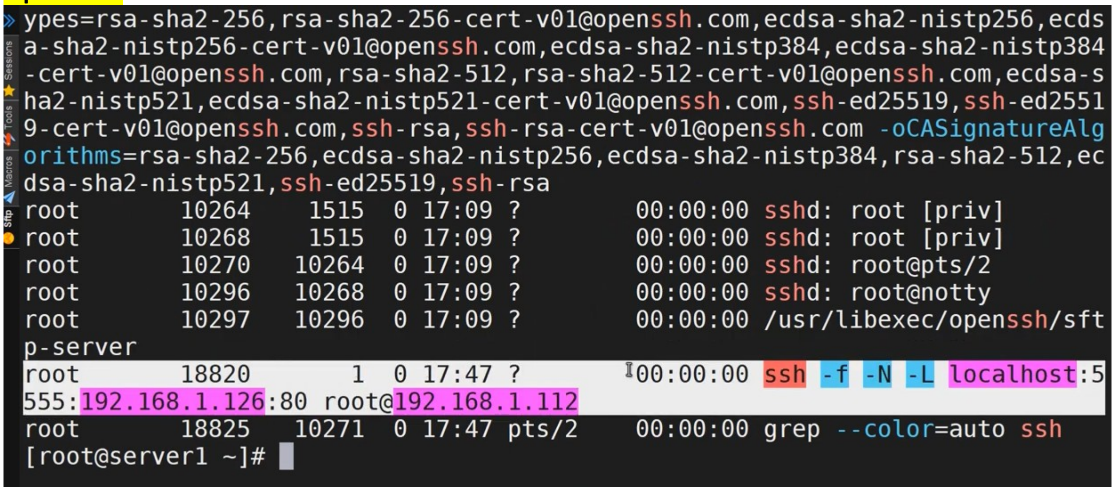
You can check the **tcpdump** capture on `server3` This shows the secure tunnel communication between `server1` and `server2` and further `server2` will connect to `server3` to connect to the webserver.

```
[root@server3 tmp]# tcpdump -i enp0s8 port 22 or 80
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on enp0s8, link-type EN10MB (Ethernet), capture size 262144 bytes
```

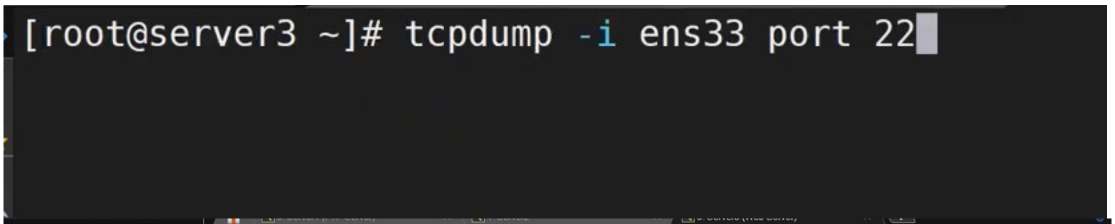


```
[root@server1 ~]# curl http://localhost:5555
Hi, Welcome To Nehra Classes Webpage.

Thanks

[root@server1 ~]# ps -ef | grep ssh
```

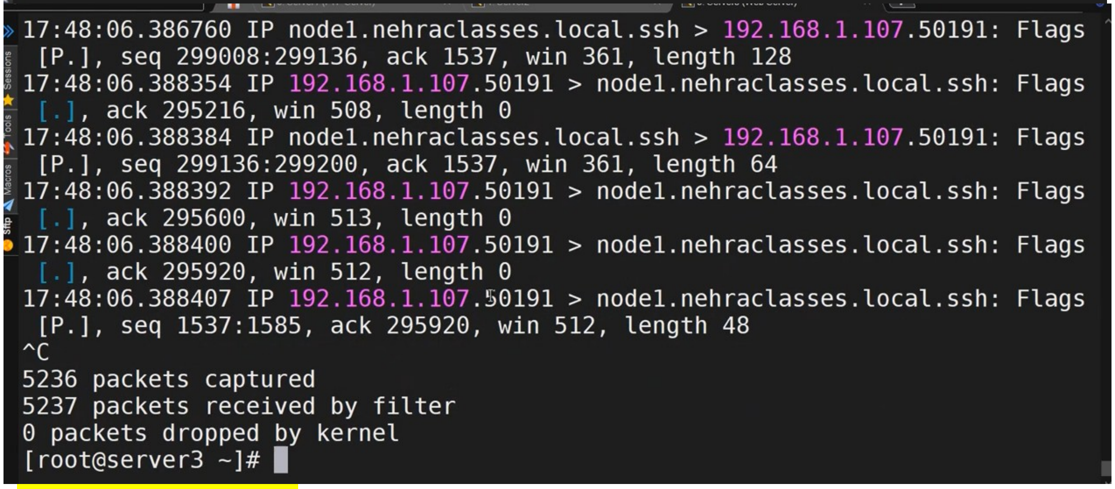
# port Fwd



```
ypes=rsa-sha2-256,rsa-sha2-256-cert-v01@openssh.com,ecdsa-sha2-nistp256,ecds
a-sha2-nistp256-cert-v01@openssh.com,ecdsa-sha2-nistp384,ecdsa-sha2-nistp384
-cert-v01@openssh.com,rsa-sha2-512,rsa-sha2-512-cert-v01@openssh.com,ecdsa-s
ha2-nistp521,ecdsa-sha2-nistp521-cert-v01@openssh.com,ssh-ed25519,ssh-ed2551
9-cert-v01@openssh.com,ssh-rsa,ssh-rsa-cert-v01@openssh.com -oCASignatureAlg
orithms=rsa-sha2-256,ecdsa-sha2-nistp256,ecdsa-sha2-nistp384,rsa-sha2-512,ec
dsa-sha2-nistp521,ssh-ed25519,ssh-rsa
root       10264    1515  0 17:09 ?        00:00:00 sshd: root [priv]
root       10268    1515  0 17:09 ?        00:00:00 sshd: root [priv]
root       10270   10264  0 17:09 ?        00:00:00 sshd: root@pts/2
root       10296   10268  0 17:09 ?        00:00:00 sshd: root@notty
root       10297   10296  0 17:09 ?        00:00:00 /usr/libexec/openssh/sft
p-server
root       18820       1  0 17:47 ?        00:00:00 ssh -f -N -L localhost:5
555:192.168.1.126:80 root@192.168.1.112
root       18825   10271  0 17:47 pts/2    00:00:00 grep --color=auto ssh
[root@server1 ~]#
```

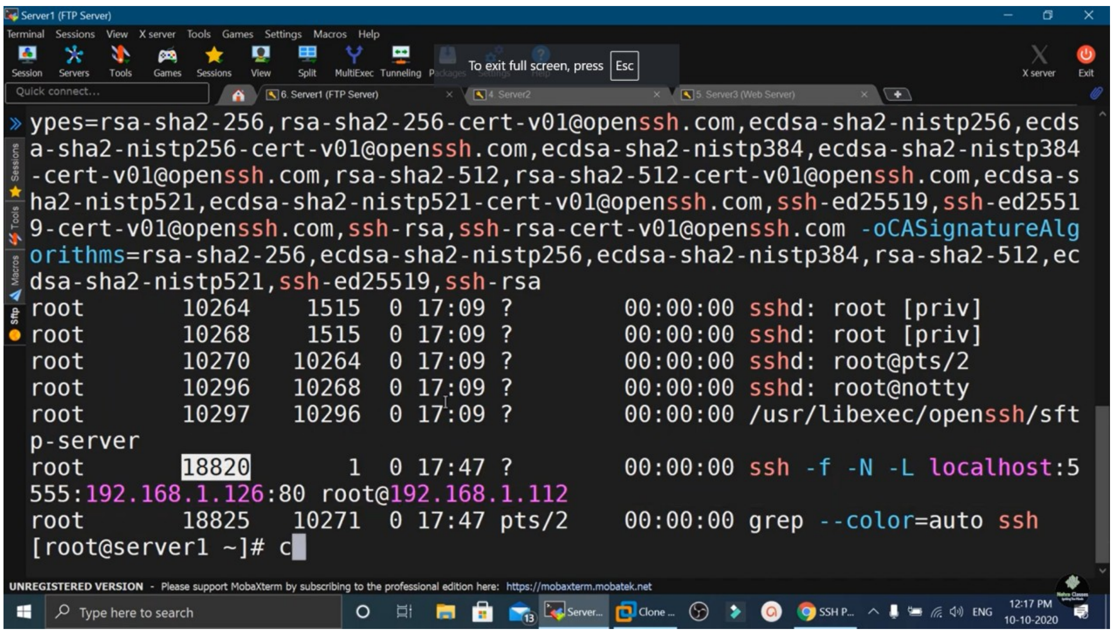


```
[root@server3 ~]# tcpdump -i ens33 port 22
```



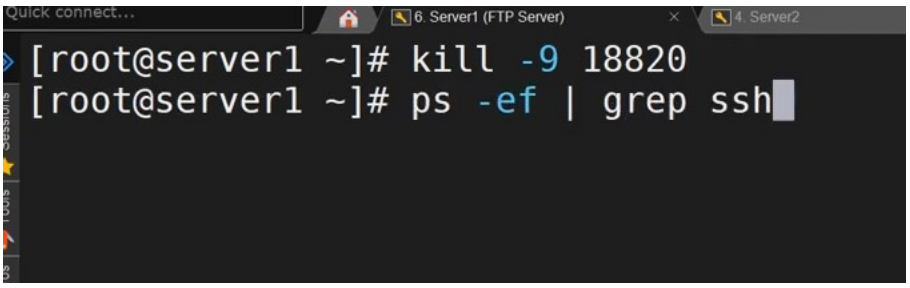
```
17:48:06.386760 IP node1.nehraclasses.local.ssh > 192.168.1.107.50191: Flags
 [P.], seq 299008:299136, ack 1537, win 361, length 128
17:48:06.388354 IP 192.168.1.107.50191 > node1.nehraclasses.local.ssh: Flags
 [.], ack 295216, win 508, length 0
17:48:06.388384 IP node1.nehraclasses.local.ssh > 192.168.1.107.50191: Flags
 [P.], seq 299136:299200, ack 1537, win 361, length 64
17:48:06.388392 IP 192.168.1.107.50191 > node1.nehraclasses.local.ssh: Flags
 [.], ack 295600, win 513, length 0
17:48:06.388400 IP 192.168.1.107.50191 > node1.nehraclasses.local.ssh: Flags
 [.], ack 295920, win 512, length 0
17:48:06.388407 IP 192.168.1.107.50191 > node1.nehraclasses.local.ssh: Flags
 [P.], seq 1537:1585, ack 295920, win 512, length 48
^C
5236 packets captured
5237 packets received by filter
0 packets dropped by kernel
[root@server3 ~]#
```

# **how to remove Port Fwd**

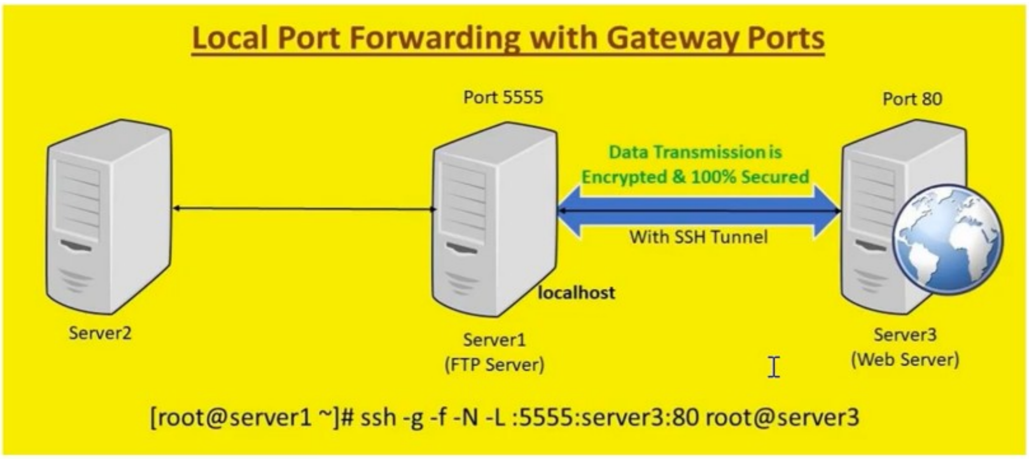

Server1 (FTP Server)
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling
To exit full screen, press Esc
X server Exit
6. Server1 (FTP Server)
4. Server2
5. Server3 (Web Server)

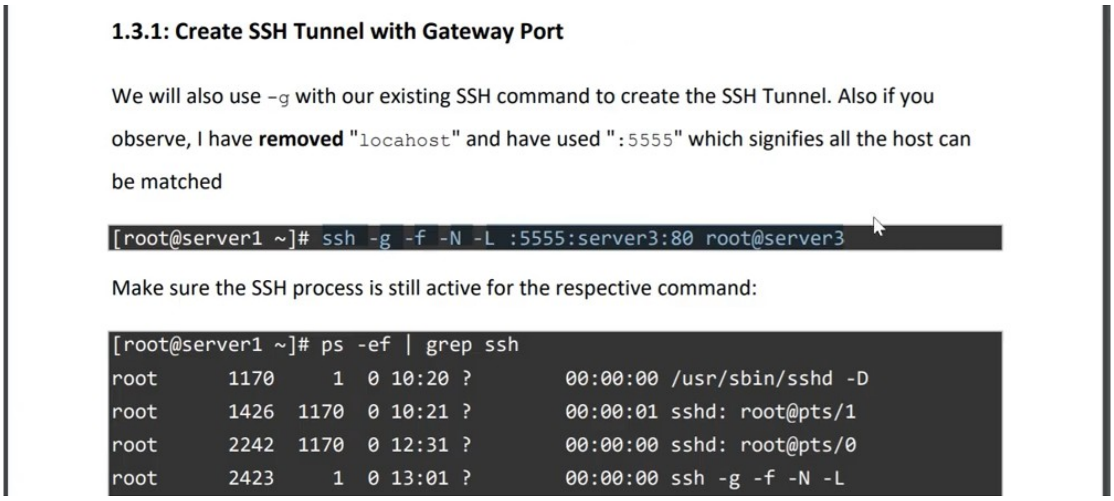
```
ypes=rsa-sha2-256,rsa-sha2-256-cert-v01@openssh.com,ecdsa-sha2-nistp256,ecds
a-sha2-nistp256-cert-v01@openssh.com,ecdsa-sha2-nistp384,ecdsa-sha2-nistp384
-cert-v01@openssh.com,rsa-sha2-512,rsa-sha2-512-cert-v01@openssh.com,ecdsa-s
ha2-nistp521,ecdsa-sha2-nistp521-cert-v01@openssh.com,ssh-ed25519,ssh-ed2551
9-cert-v01@openssh.com,ssh-rsa,ssh-rsa-cert-v01@openssh.com -oCASignatureAlg
orithms=rsa-sha2-256,ecdsa-sha2-nistp256,ecdsa-sha2-nistp384,rsa-sha2-512,ec
dsa-sha2-nistp521,ssh-ed25519,ssh-rsa
root      10264   1515  0 17:09 ?      00:00:00 sshd: root [priv]
root      10268   1515  0 17:09 ?      00:00:00 sshd: root [priv]
root      10270  10264  0 17:09 ?      00:00:00 sshd: root@pts/2
root      10296  10268  0 17:09 ?      00:00:00 sshd: root@notty
root      10297  10296  0 17:09 ?      00:00:00 /usr/libexec/openssh/sft
p-server
root      18820      1  0 17:47 ?      00:00:00 ssh -f -N -L localhost:5
555:192.168.1.126:80 root@192.168.1.112
root      18825  10271  0 17:47 pts/2  00:00:00 grep --color=auto ssh
[root@server1 ~]# c
```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net



```
[root@server1 ~]# kill -9 18820
[root@server1 ~]# ps -ef | grep ssh
```


Local Port Forwarding with Gateway Ports
Port 5555
Port 80
Data Transmission is Encrypted & 100% Secured
With SSH Tunnel
localhost
Server2
Server1 (FTP Server)
Server3 (Web Server)
[root@server1 ~]# ssh -g -f -N -L :5555:server3:80 root@server3



1.3.1: Create SSH Tunnel with Gateway Port

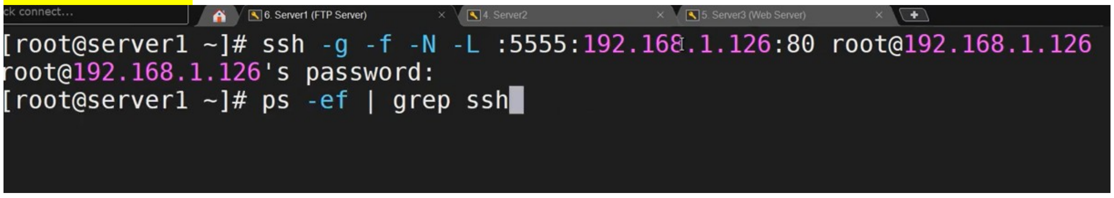
We will also use `-g` with our existing SSH command to create the SSH Tunnel. Also if you observe, I have **removed** "`localhost`" and have used "`:5555`" which signifies all the host can be matched

```
[root@server1 ~]# ssh -g -f -N -L :5555:server3:80 root@server3
```

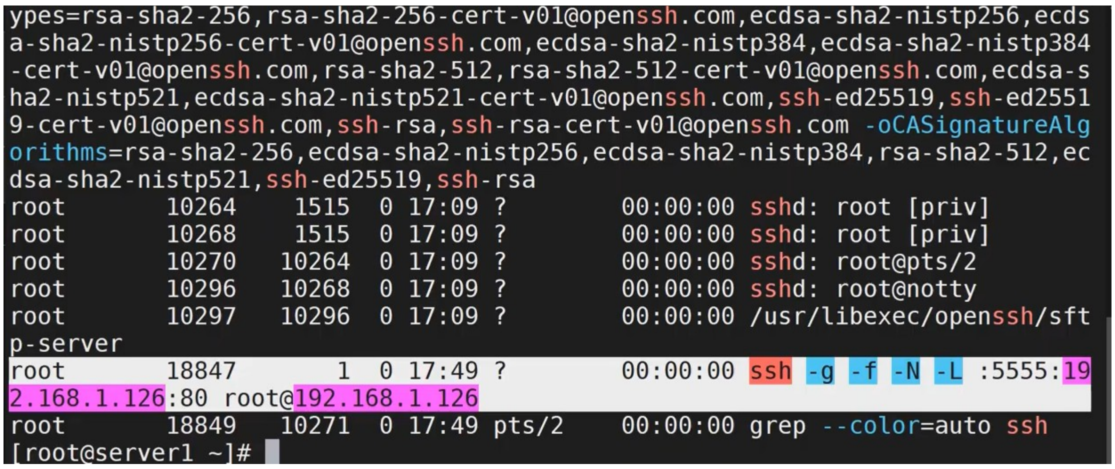
Make sure the SSH process is still active for the respective command:

```
[root@server1 ~]# ps -ef | grep ssh
root      1170     1  0 10:20 ?        00:00:00 /usr/sbin/sshd -D
root      1426  1170  0 10:21 ?        00:00:01 sshd: root@pts/1
root      2242  1170  0 12:31 ?        00:00:00 sshd: root@pts/0
root      2423     1  0 13:01 ?        00:00:00 ssh -g -f -N -L
```

# Tunnel Created



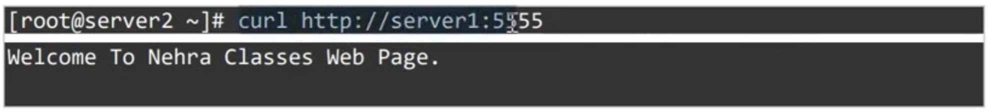
```
[root@server1 ~]# ssh -g -f -N -L :5555:192.168.1.126:80 root@192.168.1.126
root@192.168.1.126's password:
[root@server1 ~]# ps -ef | grep ssh
```



```
ypes=rsa-sha2-256,rsa-sha2-256-cert-v01@openssh.com,ecdsa-sha2-nistp256,ecds
a-sha2-nistp256-cert-v01@openssh.com,ecdsa-sha2-nistp384,ecdsa-sha2-nistp384
-cert-v01@openssh.com,rsa-sha2-512,rsa-sha2-512-cert-v01@openssh.com,ecdsa-s
ha2-nistp521,ecdsa-sha2-nistp521-cert-v01@openssh.com,ssh-ed25519,ssh-ed2551
9-cert-v01@openssh.com,ssh-rsa,ssh-rsa-cert-v01@openssh.com -oCASignatureAlg
orithms=rsa-sha2-256,ecdsa-sha2-nistp256,ecdsa-sha2-nistp384,rsa-sha2-512,ec
dsa-sha2-nistp521,ssh-ed25519,ssh-rsa
root       10264    1515  0 17:09 ?        00:00:00 sshd: root [priv]
root       10268    1515  0 17:09 ?        00:00:00 sshd: root [priv]
root       10270   10264  0 17:09 ?        00:00:00 sshd: root@pts/2
root       10296   10268  0 17:09 ?        00:00:00 sshd: root@notty
root       10297   10296  0 17:09 ?        00:00:00 /usr/libexec/openssh/sft
p-server
root       18847       1  0 17:49 ?        00:00:00 ssh -g -f -N -L :5555:19
2.168.1.126:80 root@192.168.1.126
root       18849   10271  0 17:49 pts/2    00:00:00 grep --color=auto ssh
[root@server1 ~]#
```

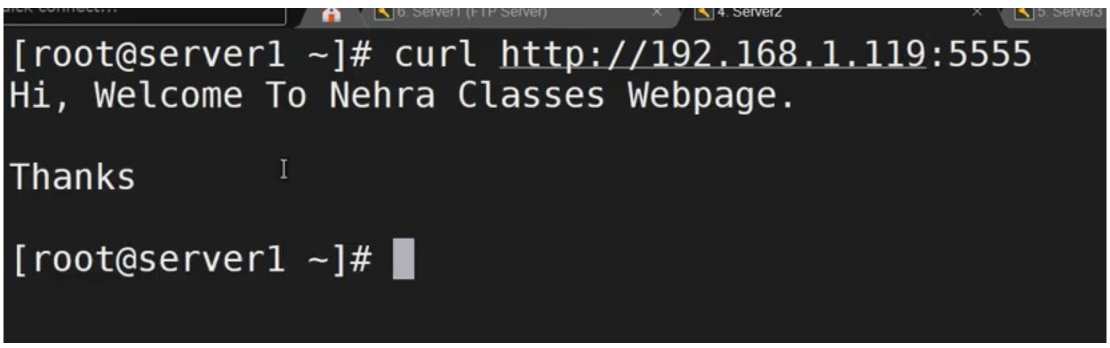
1.3.2: Verify the Local Port Forwarding

We will use `curl` command from `server2` this time to check if it can connect to `server3` using port `5555` from the secure tunnel

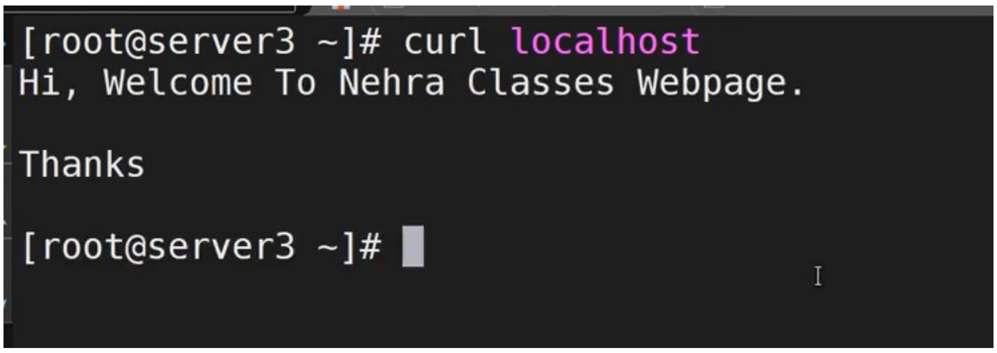


```
[root@server2 ~]# curl http://server1:5555
Welcome To Nehra Classes Web Page.
```

So the `server2` was successfully able to connect to `server3:80` and also to the webserver using forwarding port `server1:5555`



```
[root@server1 ~]# curl http://192.168.1.119:5555
Hi, Welcome To Nehra Classes Webpage.

Thanks

[root@server1 ~]#
```



```
[root@server3 ~]# curl localhost
Hi, Welcome To Nehra Classes Webpage.

Thanks

[root@server3 ~]#
```

```
[root@server1 ~]# kill -9 2423
```

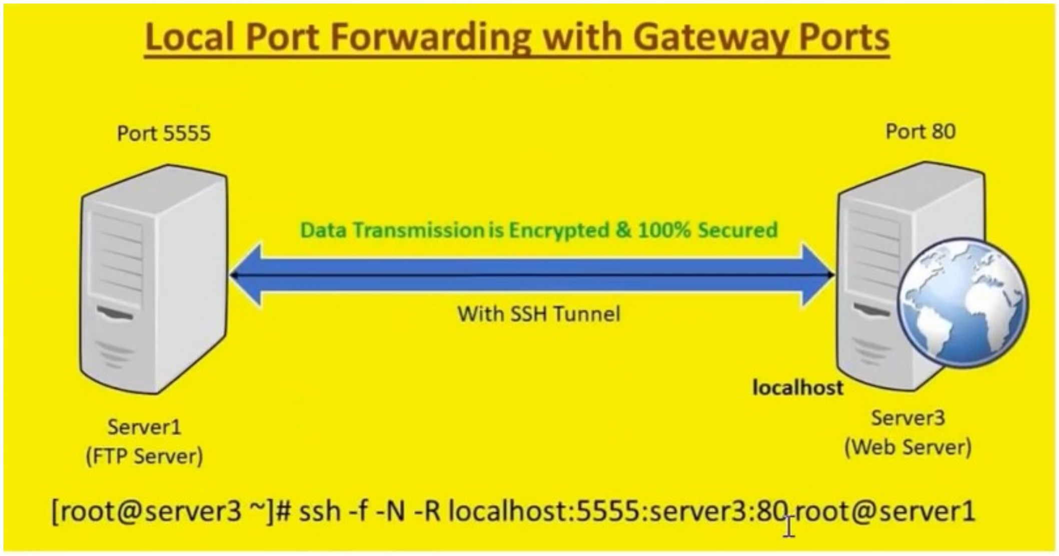
2: Remote port forwarding

- A remotely forwarded port is just like a local one, but the **directions are reversed**
- This time the TCP client is remote, its server is local, and a forwarded connection is initiated from the remote machine.
- Remote port forwarding is less common and can be used to connect to a local port that cannot be reached from the internet, to a port on the server that is available on the internet
- The `-R` option specifies remote forwarding. It is followed by three values, separated by colons as before but interpreted slightly differently.

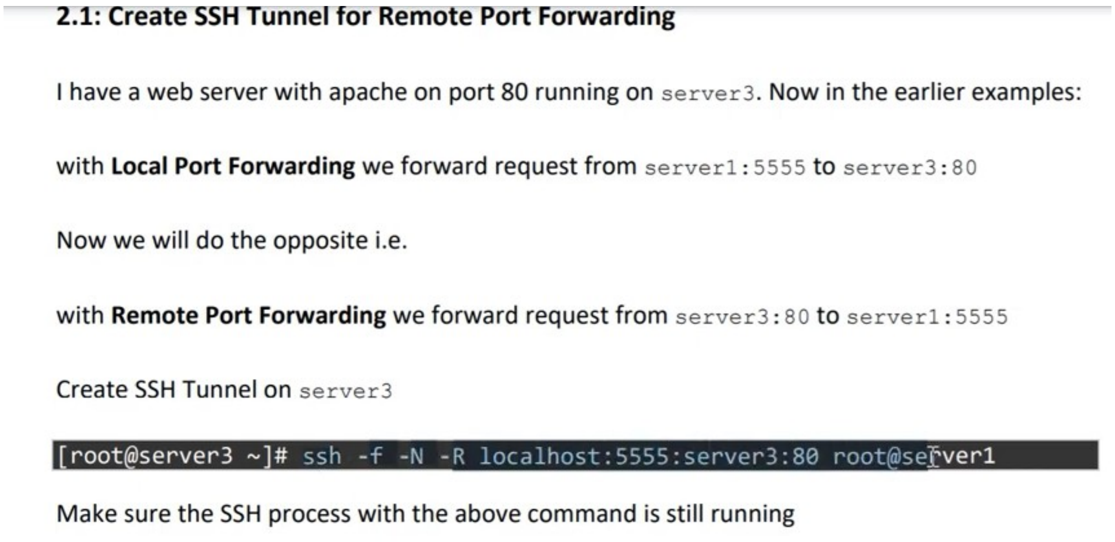
The **syntax** to perform Reverse Port Forwarding would be `ssh -R bindAddress:remotePort:forwardToHost:onPort connectToHost`

The **syntax** to perform Reverse Port Forwarding would be `ssh -R bindAddress:remotePort:forwardToHost:onPort connectToHost`

- The **first field** is the bind address on localhost. By default, TCP listening sockets on the server will be bound to the loopback interface only
- The **second field** is the `remotePort` which which will be used to connect to the destination port i.e. 80.
- The **third field** is for `forwardToHost` i.e. the hostname or IP of the server to which you wish to connect as part of forwarding
- The **fourth field** is the `onPort` to which the forwarding should happen on the "`forwardToHost`"
- Lastly **provide the server details** towards which the SSH Tunnel should be created


Local Port Forwarding with Gateway Ports
Port 5555
Port 80
Data Transmission is Encrypted & 100% Secured
With SSH Tunnel
localhost
Server1
(FTP Server)
Server3
(Web Server)

```
[root@server3 ~]# ssh -f -N -R localhost:5555:server3:80 root@server1
```



2.1: Create SSH Tunnel for Remote Port Forwarding

I have a web server with apache on port 80 running on `server3`. Now in the earlier examples:

with **Local Port Forwarding** we forward request from `server1:5555` to `server3:80`

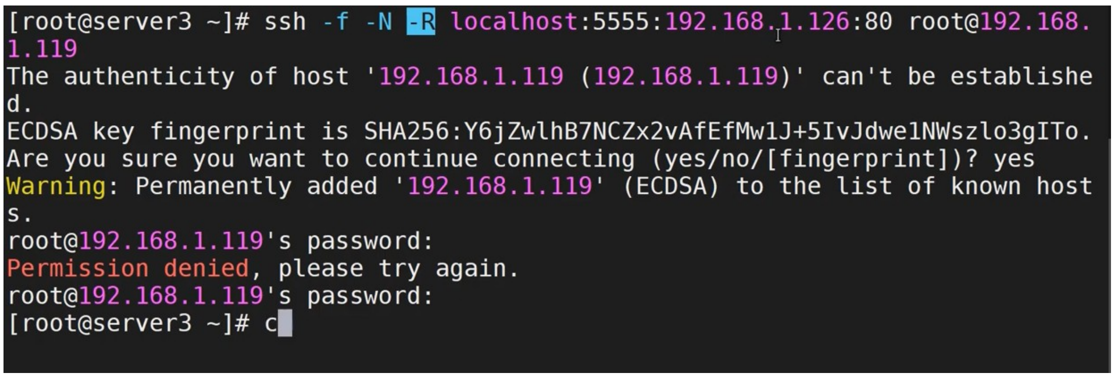
Now we will do the opposite i.e.

with **Remote Port Forwarding** we forward request from `server3:80` to `server1:5555`

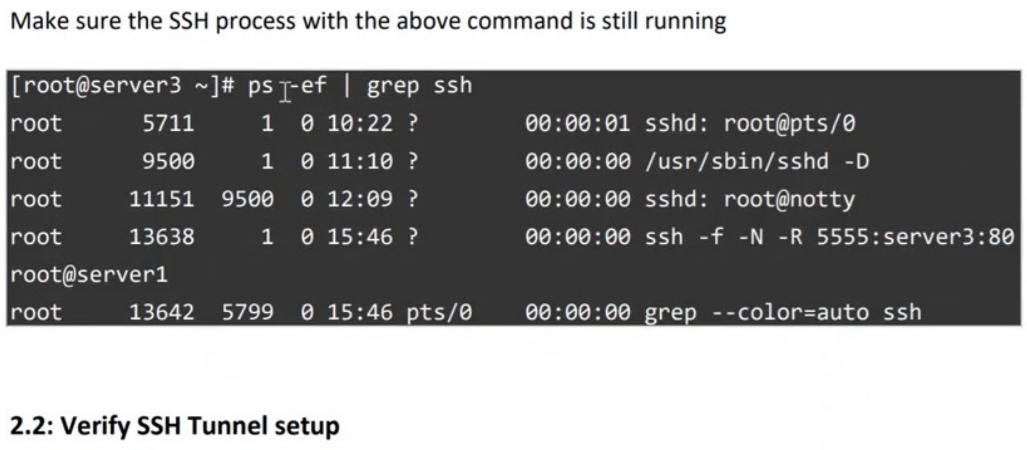
Create SSH Tunnel on `server3`

```
[root@server3 ~]# ssh -f -N -R localhost:5555:server3:80 root@server1
```

Make sure the SSH process with the above command is still running



```
[root@server3 ~]# ssh -f -N -R localhost:5555:192.168.1.126:80 root@192.168.
1.119
The authenticity of host '192.168.1.119 (192.168.1.119)' can't be establishe
d.
ECDSA key fingerprint is SHA256:Y6jZwlhB7NCZx2vAfEfMw1J+5IvJdwe1NWszlo3gITo.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.1.119' (ECDSA) to the list of known host
s.
root@192.168.1.119's password:
Permission denied, please try again.
root@192.168.1.119's password:
[root@server3 ~]# c
```

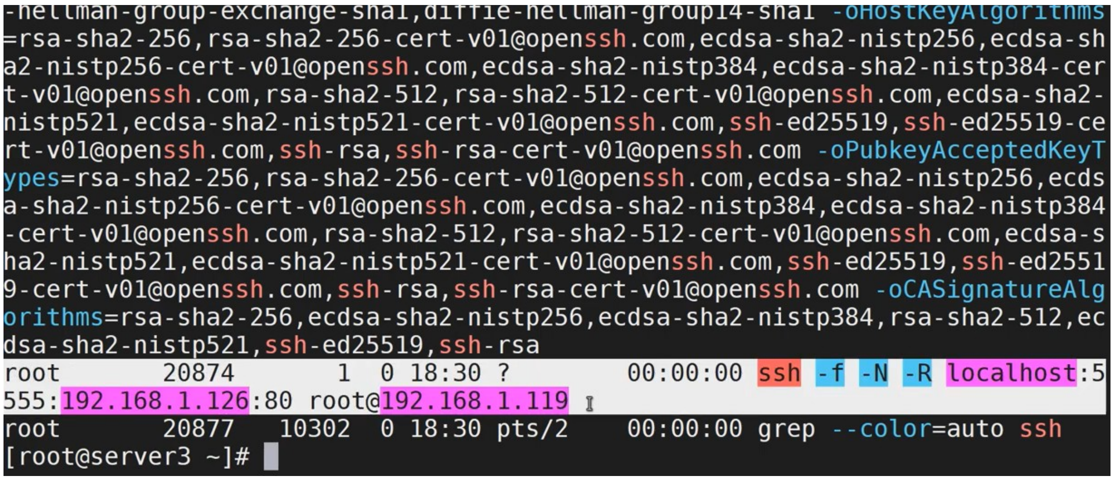


Make sure the SSH process with the above command is still running

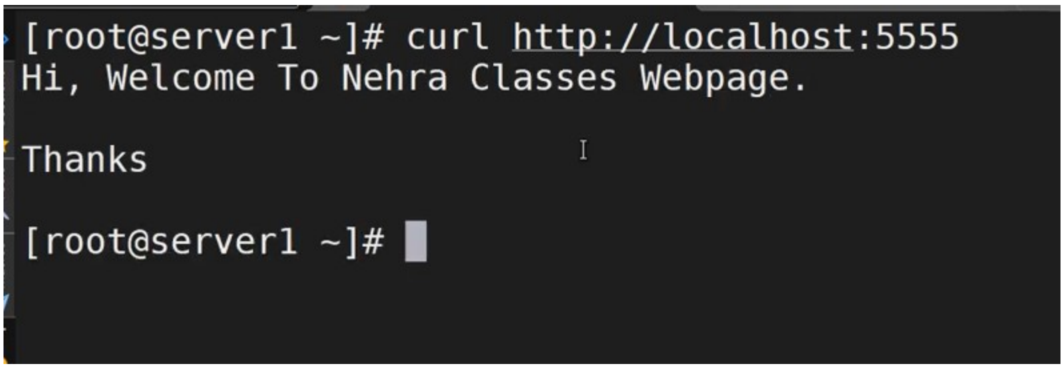
```
[root@server3 ~]# ps -ef | grep ssh
root      5711     1  0 10:22 ?        00:00:01 sshd: root@pts/0
root      9500     1  0 11:10 ?        00:00:00 /usr/sbin/sshd -D
root     11151  9500  0 12:09 ?        00:00:00 sshd: root@notty
root     13638     1  0 15:46 ?        00:00:00 ssh -f -N -R 5555:server3:80
root@server1
root     13642  5799  0 15:46 pts/0    00:00:00 grep --color=auto ssh
```

2.2: Verify SSH Tunnel setup

# Port Fwd



```
-hellman-group-exchange-sha1,diffie-hellman-group14-sha1 -oHostKeyAlgorithms
=rsa-sha2-256,rsa-sha2-256-cert-v01@openssh.com,ecdsa-sha2-nistp256,ecdsa-sh
a2-nistp256-cert-v01@openssh.com,ecdsa-sha2-nistp384,ecdsa-sha2-nistp384-cer
t-v01@openssh.com,rsa-sha2-512,rsa-sha2-512-cert-v01@openssh.com,ecdsa-sha2-
nistp521,ecdsa-sha2-nistp521-cert-v01@openssh.com,ssh-ed25519,ssh-ed25519-ce
rt-v01@openssh.com,ssh-rsa,ssh-rsa-cert-v01@openssh.com -oPubkeyAcceptedKeyT
ypes=rsa-sha2-256,rsa-sha2-256-cert-v01@openssh.com,ecdsa-sha2-nistp256,ecds
a-sha2-nistp256-cert-v01@openssh.com,ecdsa-sha2-nistp384,ecdsa-sha2-nistp384
-cert-v01@openssh.com,rsa-sha2-512,rsa-sha2-512-cert-v01@openssh.com,ecdsa-s
ha2-nistp521,ecdsa-sha2-nistp521-cert-v01@openssh.com,ssh-ed25519,ssh-ed2551
9-cert-v01@openssh.com,ssh-rsa,ssh-rsa-cert-v01@openssh.com -oCASignatureAlg
orithms=rsa-sha2-256,ecdsa-sha2-nistp256,ecdsa-sha2-nistp384,rsa-sha2-512,ec
dsa-sha2-nistp521,ssh-ed25519,ssh-rsa
root       20874       1  0 18:30 ?        00:00:00 ssh -f -N -R localhost:5
555:192.168.1.126:80 root@192.168.1.119
root       20877   10302  0 18:30 pts/2    00:00:00 grep --color=auto ssh
[root@server3 ~]#
```



```
[root@server1 ~]# curl http://localhost:5555
Hi, Welcome To Nehra Classes Webpage.

Thanks

[root@server1 ~]#
```