How to Set up SSH Tunneling (Port Forwarding)

Updated Nov 5, 2020

•

8 min read

SSH tunneling or SSH port forwarding is a method of creating an encrypted SSH connection between a client and a server machine through which services ports can be relayed.

SSH forwarding is useful for transporting network data of services that use an unencrypted protocol, such as VNC or FTP , accessing geo-restricted content, or bypassing intermediate firewalls. Basically, you can forward any TCP port and tunnel the traffic over a secure SSH connection.

There are three types of SSH port forwarding:

- Local Port Forwarding. - Forwards a connection from the client host to the SSH server host and then to the destination host port.
- Remote Port Forwarding. - Forwards a port from the server host to the client host and then to the destination host port.
- Dynamic Port Forwarding. - Creates a SOCKS proxy server that allows communication across a range of ports.

This article explains how to set up local, remote, and dynamic encrypted SSH tunnels.

Local Port Forwarding

Local port forwarding allows you to forward a port on the local (ssh client) machine to a port on the remote (ssh server) machine, which is then forwarded to a port on the destination machine.

In this forwarding type, the SSH client listens on a given port and tunnels any connection to that port to the specified port on the remote SSH server, which then connects to a port on the destination machine. The destination machine can be the remote SSH server or any other machine.

Local port forwarding is mostly used to connect to a remote service on an internal network such as a database or VNC server.

In Linux, macOS, and other Unix systems, to create a local port forwarding, pass the `-L` option to the `ssh` client:

```
ssh -L
[LOCAL_IP:]LOCAL_PORT:DESTINATION:DESTINATION_POR
T [USER@]SSH_SERVER
```

The options used are as follows:

- `[LOCAL_IP:]LOCAL_PORT` - The local machine IP address and port number. When `LOCAL_IP` is omitted, the ssh client binds on the localhost.
- `DESTINATION:DESTINATION_PORT` - The IP or hostname and the port of the destination machine.
- `[USER@]SERVER_IP` - The remote SSH user and server IP address.

You can use any port number greater than `1024` as a `LOCAL_PORT`. Ports numbers less than `1024` are privileged ports and can be used only by root. If your SSH server is listening on a port other than 22 (the default), use the `-p [PORT_NUMBER]` option.

The destination hostname must be resolvable from the SSH server.

Let’s say you have a MySQL database server running on machine `db001.host` on an internal (private) network, on port 3306, which is accessible from the machine `pub001.host`, and you want to connect using your local machine MySQL client to the database server. To do so, you can forward the connection using the following command:

```
ssh -L 3336:db001.host:3306 user@pub001.host
```

Once you run the command, you'll be prompted to enter the remote SSH user password. Once entered, you will be logged into the remote server, and the SSH tunnel will be established. It is also a good idea to set up an SSH key-based authentication and connect to the server without entering a password.

Now, if you point your local machine database client to `127.0.0.1:3336`, the connection will be forwarded to the `db001.host:3306` MySQL server through the `pub001.host` machine that acts as an intermediate server.

You can forward multiple ports to multiple destinations in a single ssh command. For example, you have another MySQL database server running on machine `db002.host`, and you want to connect to both servers from your local client, you would run:

```
ssh -L 3336:db001.host:3306 3337:db002.host:3306
user@pub001.host
```

To connect to the second server, you would use `127.0.0.1:3337`.

When the destination host is the same as the SSH server, instead of specifying the destination host IP or hostname, you can use `localhost`.

Say you need to connect to a remote machine through VNC, which runs on the same server, and it is not accessible from the outside. The command you would use is:

```
ssh -L 5901:127.0.0.1:5901 -N -f user@remote.host
```

The `-f` option tells the `ssh` command to run in the background and `-N` not to execute a remote command. We are using `localhost` because the VNC and the SSH server are running on the same host.

If you are having trouble setting up tunneling, check your remote SSH server configuration and make sure `AllowTcpForwarding` is not set to `no`. By default, forwarding is allowed.

Remote Port Forwarding

Remote port forwarding is the opposite of local port forwarding. It allows you to forward a port on the remote (ssh server) machine to a port on the local (ssh client) machine, which is then forwarded to a port on the destination machine.

In this forwarding type, the SSH server listens on a given port and tunnels any connection to that port to the specified port on the local SSH client, which then connects to a port on the destination machine. The destination machine can be the local or any other machine.

In Linux, macOS, and other Unix systems to create a remote port forwarding, pass the `-R` option to the `ssh` client:

```
ssh -R
[REMOTE:]REMOTE_PORT:DESTINATION:DESTINATION_PORT
[USER@]SSH_SERVER
```

The options used are as follows:

- `[REMOTE:]REMOTE_PORT` - The IP and the port number on the remote SSH server. An empty `REMOTE` means that the remote SSH server will bind on all interfaces.
- `DESTINATION:DESTINATION_PORT` - The IP or hostname and the port of the destination machine.
- `[USER@]SERVER_IP` - The remote SSH user and server IP address.

Remote port forwarding is mostly used to give access to an internal service to someone from the outside.

Let's say you are developing a web application on your local machine, and you want to show a preview to your fellow developer. You do not have a public IP, so the other developer can't access the application via the Internet.

If you have access to a remote SSH server, you can set up a remote port forwarding as follows:

```
ssh -R 8080:127.0.0.1:3000 -N -f user@remote.host
```

The command above will make the ssh server listen on port `8080`, and tunnel all traffic from this port to your local machine on port `3000`.

Now your fellow developer can type `the_ssh_server_ip:8080` in his/her browser and preview your awesome application.

If you are having trouble setting up remote port forwarding, make sure `GatewayPorts` is set to `yes` in the remote SSH server configuration.

Dynamic Port Forwarding

Dynamic port forwarding allows you to create a socket on the local (ssh client) machine, which acts as a SOCKS proxy server. When a client connects to this port, the connection is forwarded to the remote (ssh server) machine, which is then forwarded to a dynamic port on the destination machine.

This way, all the applications using the SOCKS proxy will connect to the SSH server, and the server will forward all the traffic to its actual destination.

In Linux, macOS, and other Unix systems to create a dynamic port forwarding (SOCKS) pass the `-D` option to the `ssh` client:

```
ssh -D [LOCAL_IP:]LOCAL_PORT [USER@]SSH_SERVER
```

The options used are as follows:

- `[LOCAL_IP:]LOCAL_PORT` - The local machine IP address and port number. When `LOCAL_IP` is omitted, the ssh client binds on localhost.
- `[USER@]SERVER_IP` - The remote SSH user and server IP address.

A typical example of a dynamic port forwarding is to tunnel the web browser traffic through an SSH server.

The following command will create a SOCKS tunnel on port `9090`:

```
ssh -D 9090 -N -f user@remote.host
```

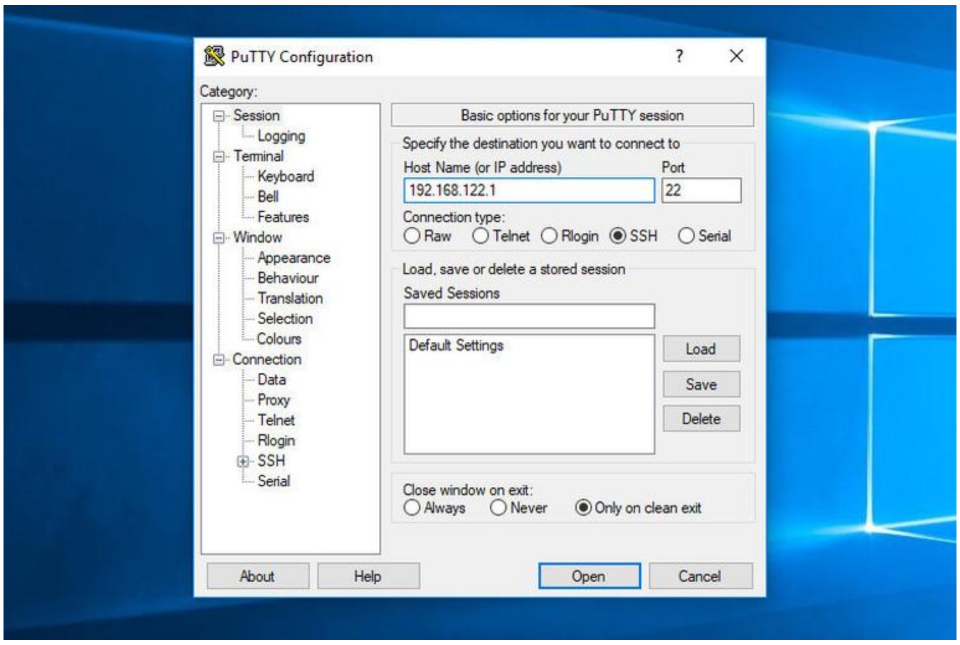
Once the tunneling is established, you can configure your application to use it. This article explains how to configure Firefox and Google Chrome browser to use the SOCKS proxy.

The port forwarding has to be separately configured for each application that you want to tunnel the traffic thought it.

Set up SSH Tunneling in Windows

Windows users can create SSH tunnels using the PuTTY SSH client. You can download PuTTY here .

1. Launch Putty and enter the SSH server IP Address in the `Host name (or IP address)` field.



Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): 192.168.122.1

Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

Load, save or delete a stored session

Saved Sessions

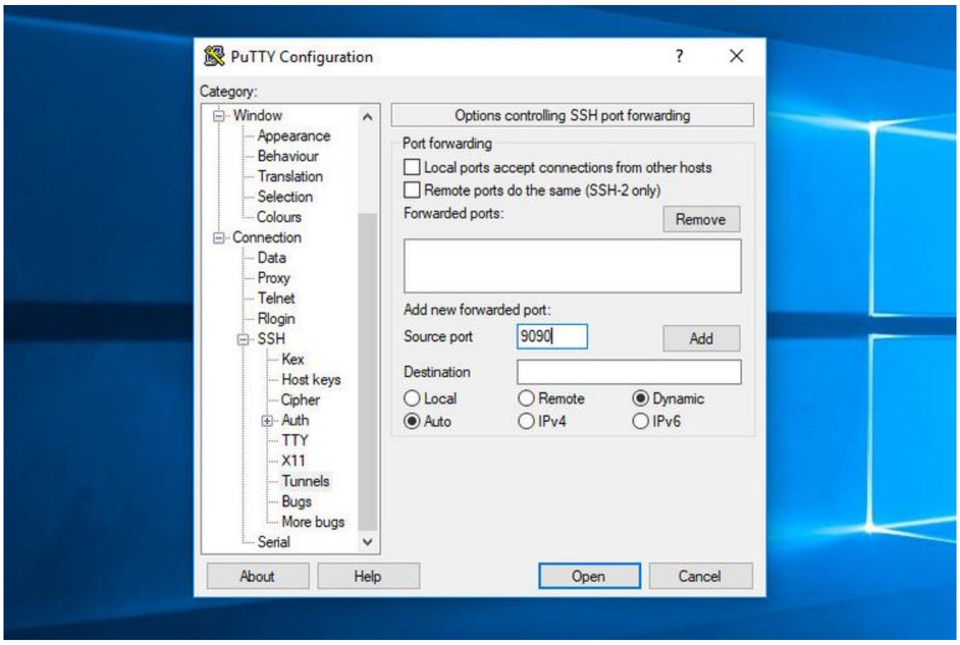
Default Settings

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

2. Under the `Connection` menu, expand `SSH` and select `Tunnels`. Check the `Local` radio button to setup local, `Remote` for remote, and `Dynamic` for dynamic port forwarding.

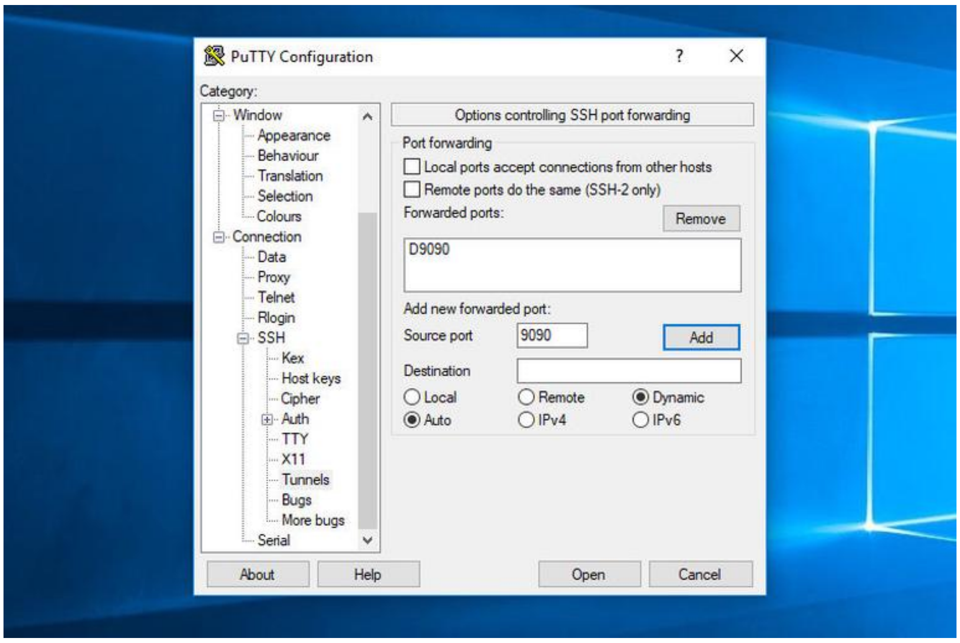
   - When setting up local forwarding, enter the local forwarding port in the `Source Port` field and in `Destination` enter the destination host and IP, for example, `localhost:5901`.

  - For remote port forwarding, enter the remote SSH server forwarding port in the `Source Port` field and in `Destination` enter the destination host and IP, for example, `localhost:3000`.
  - If setting up dynamic forwarding, enter only the local SOCKS port in the `Source Port` field.



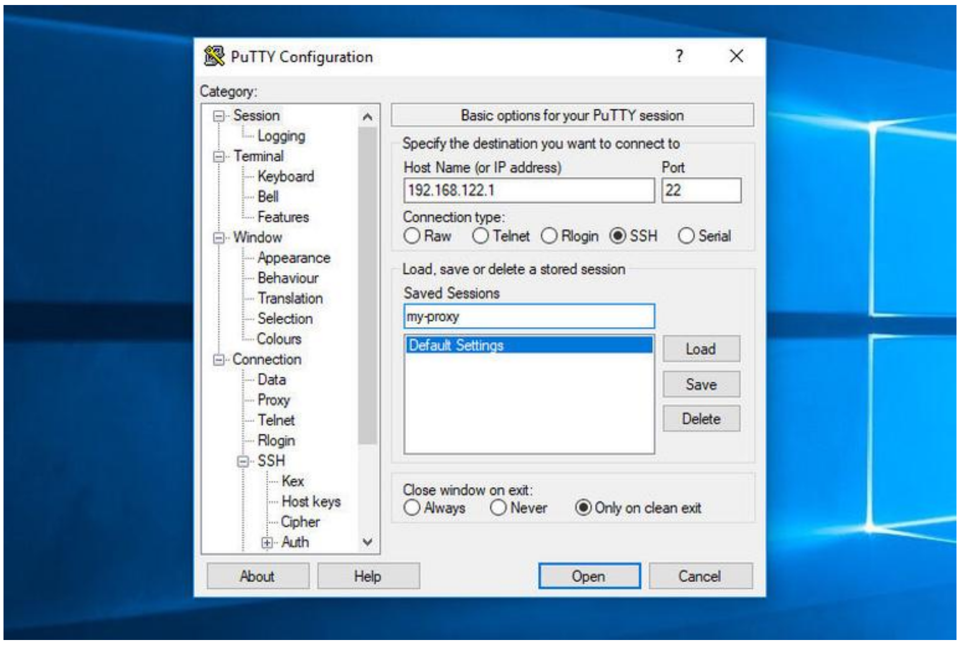
☐ Local ports accept connections from other hosts
☐ Remote ports do the same (SSH-2 only)
Forwarded ports:
Add new forwarded port:
Source port: 9090
Destination:
○ Local ○ Remote ● Dynamic
● Auto ○ IPv4 ○ IPv6

3. Click on the `Add` button, as shown in the image below.



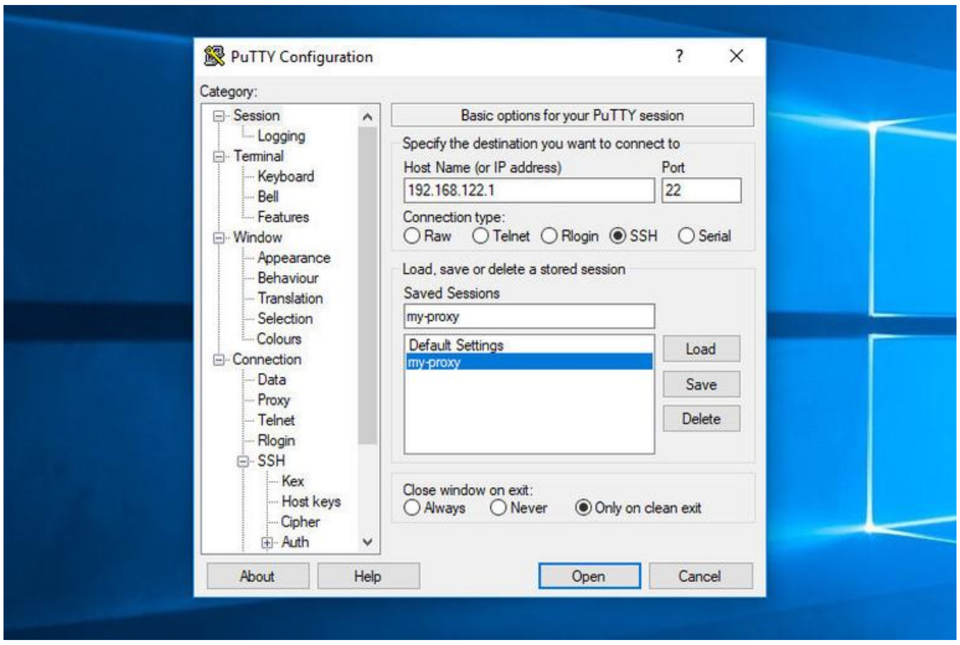
☐ Local ports accept connections from other hosts
☐ Remote ports do the same (SSH-2 only)
Forwarded ports: D9090
Add new forwarded port:
Source port: 9090
Destination:
○ Local ○ Remote ● Dynamic
● Auto ○ IPv4 ○ IPv6

4. Go back to the `Session` page to save the settings so that you do not need to enter them each time. Enter the session name in the `Saved Session` field and click on the `Save` button.



Specify the destination you want to connect to
Host Name (or IP address): 192.168.122.1
Port: 22
Connection type: Raw Telnet Rlogin [✓ SSH] Serial
Load, save or delete a stored session
Saved Sessions: my-proxy
Default Settings
Close window on exit: Always Never [✓ Only on clean exit]

5. Select the saved session and log in to the remote server by clicking on the `Open` button.



Specify the destination you want to connect to
Host Name (or IP address): 192.168.122.1
Port: 22
Connection type: Raw Telnet Rlogin [✓ SSH] Serial
Load, save or delete a stored session
Saved Sessions: my-proxy
Default Settings
[✓ my-proxy]
Close window on exit: Always Never [✓ Only on clean exit]

A new window asking for your username and password will show up. Once you enter your username and password, you will be logged in to your server, and the SSH tunnel will be started.

Setting up public key authentication allows you to connect to your server without entering a password.

Conclusion

We have shown you how to set up SSH tunnels and forward the traffic through a secure SSH connection. For ease of use, you can define the

SSH tunnel in your SSH config file or create a Bash alias that will set up the SSH tunnel.

If you hit a problem or have feedback, leave a comment below.