Laboratory_20: SSH Tunneling - Four Tunneling Cases with Python HTTP Server.

Prerequisites

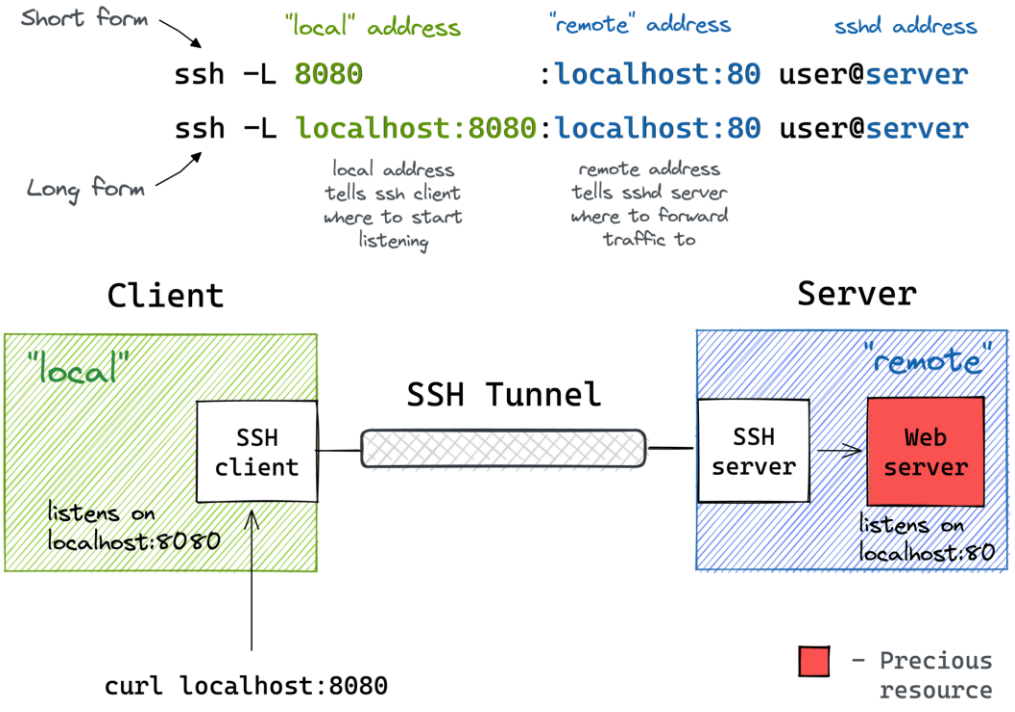
- VM1
- VM2
- SSH service running on both VMs
- Python 3 installed on both VMs

Background / Scenario

In this lab, you will explore four different SSH tunneling scenarios using two virtual machines. You'll learn how to establish secure tunnels to access services that are not directly reachable due to network topology or security restrictions.

Case 1: Local Port Forwarding (Most Common)

You want to access the HTTP server running on VM2's (Server) localhost from VM1 (Client).



1. On VM2 (Server), start the HTTP server:

   sudo python3 -m http.server --bind 127.0.0.1 80

2. Open the tunnel on VM1 (Client):

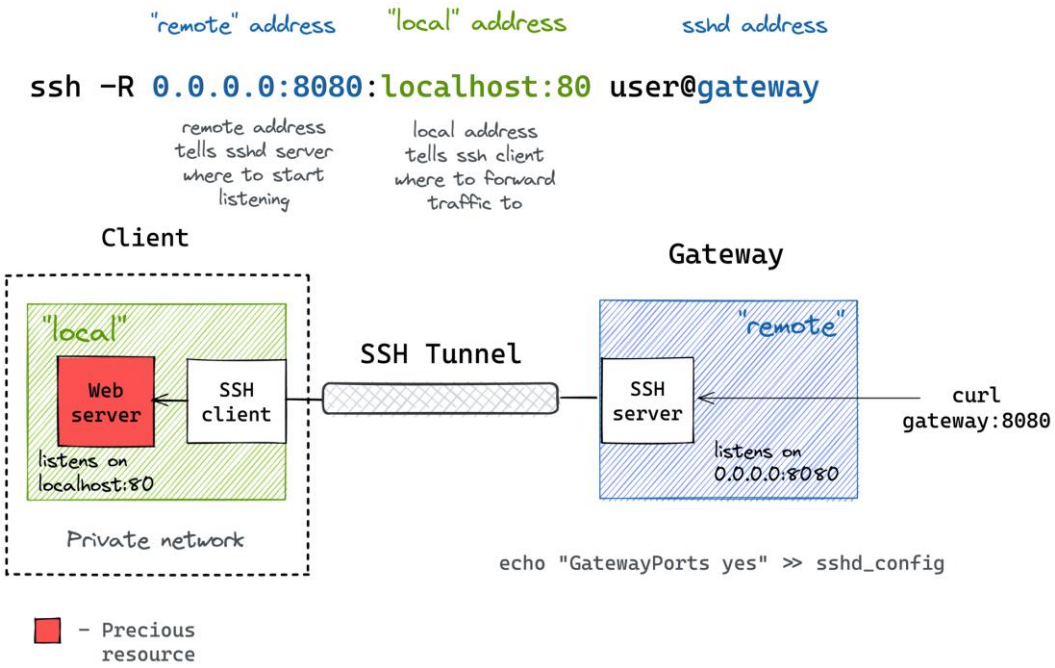
   ssh -N -L 8080:127.0.0.1:80 user@192.168.1.100 (probar con -f)

3. Test access from VM1 (Client):

   curl http://localhost:8080

Case 2: Remote Port Forwarding

You want to make a service on VM2 (Server) accessible from VM1 (Client) through SSH tunnel.



4. On VM1 (Client), start the HTTP server:

   sudo python3 -m http.server --bind 127.0.0.1 90

5. Open the tunnel on VM1 (Client):

ssh -N -R 9090:127.0.0.1:90 user@192.168.1.100 (probar con -f)

6. Test access from VM2 (Server)

   curl http://localhost:9090

Case 3: Dynamic Port Forwarding (SOCKS Proxy)

Scenario

Create a SOCKS proxy to route traffic through VM1, useful for accessing multiple services or networks through the SSH tunnel.

7. On VM2 (Server), start the HTTP server:

   sudo python3 -m http.server --bind 127.0.0.1 80

8. Open the tunnel on VM1 (Client):

   ssh -N -D 1080 user@192.168.1.100 (probar con -f)

9. Test access from VM1 (Client):

   curl --proxy socks5://localhost:1080 http://127.0.0.1:80