


Lab 29 How to connect to an internal network using OpenVPN

Step 1: Connect to Ubuntu via SSH

`ssh osboxes@192.168.164.193`

A screenshot of a terminal window titled 'osboxes@osboxes: ~'. The user is in the directory '/home/fem1'. They run the command 'ssh osboxes@192.168.164.193'. The terminal shows the SSH handshake process, including a warning about a new host key fingerprint. The user is prompted for a password, but it is denied. The terminal then displays system information for Ubuntu 24.10, including system load, memory usage, and available updates. The user is prompted to run 'apt list --upgradable' to see the list of updates.

```
(root@kali) ~ (/home/fem1)
→ ssh osboxes@192.168.164.193
The authenticity of host '192.168.164.193 (192.168.164.193)' can't be established.
ED25519 key fingerprint is SHA256:a0Uj+NmzLmLile6g5745eLAZ/zN3lWnIoqUd2H+gAQ.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.164.193' (ED25519) to the list of known hosts.
osboxes@192.168.164.193's password:
Permission denied, please try again.
osboxes@192.168.164.193's password:
Welcome to Ubuntu 24.10 (GNU/Linux 6.11.0-21-generic x86_64)

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

System information as of Sun Jul  6 05:51:56 AM UTC 2025

System load:  0.3          Processes:           115
Usage of /:   7.0% of 97.87GB Users logged in:      0
Memory usage: 15%         IPv4 address for enp0s3: 192.168.164.193
Swap usage:   0%

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.
   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

74 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

The list of available updates is more than a week old.
```

✓ What it does:

- `ssh` → *secure shell*, lets you remotely log in to another Linux machine
- `osboxes` → the **username** on the Ubuntu server
- `192.168.164.193` → the **IP address** of the Ubuntu machine

SSH opens an encrypted tunnel to the Ubuntu machine so you can run commands on it as if you're sitting there directly.

```
sudo apt update
sudo apt upgrade openvpn
```

 **What it does:**

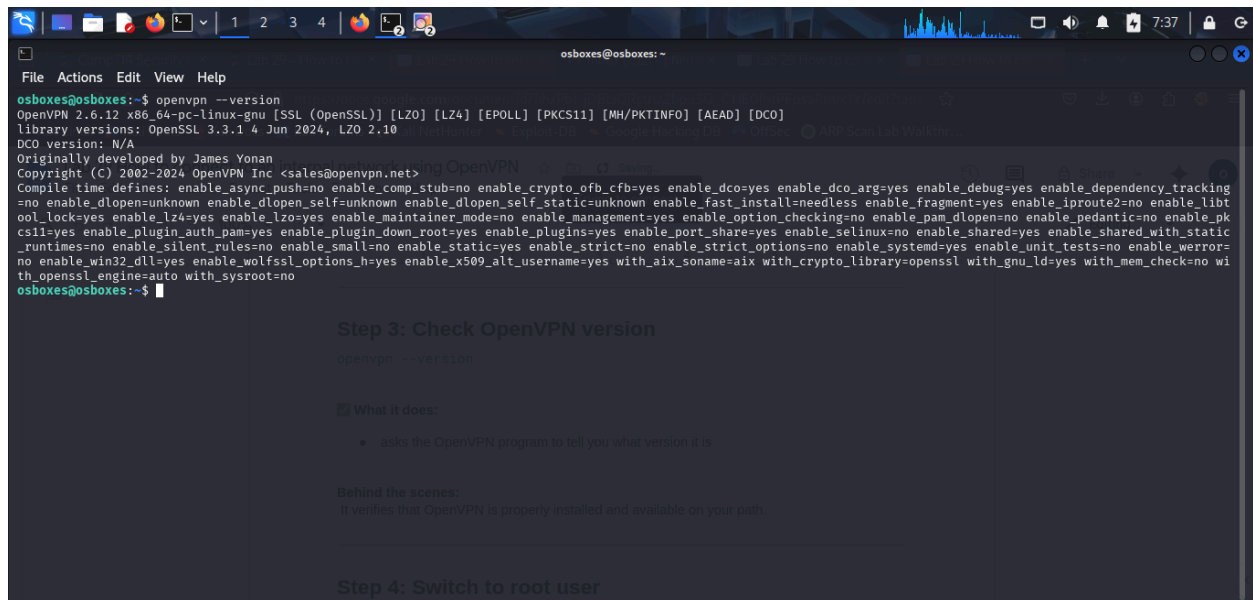
- ### Behind the scenes:

- `apt update` downloads the *latest list of available software*

- **apt upgrade** checks what you have installed, and upgrades it to match the newest available version
→ *This ensures you get security patches and no outdated bugs.*

Step 3: Check OpenVPN version

openvpn --version



```
osboxes@osboxes:~$ openvpn --version
OpenVPN 2.6.12 x86_64-linux-gnu [SSL (OpenSSL)] [LZO] [LZ4] [EPOLL] [PKCS11] [MH/PKTINFO] [AEAD] [DCO]
library versions: OpenSSL 3.3.1 4 Jun 2024, LZO 2.10
DCO version: N/A
Originally developed by James Yonan
Copyright (C) 2002-2024 OpenVPN Inc <sales@openvpn.net>
Compile time defines: enable_async_push=no enable_comp_stub=no enable_crypto_ofb_cfb=yes enable_dco=yes enable_dco_arg=yes enable_debug=yes enable_dependency_tracking
=no enable_dlopen_unknown enable_dlopen_self_unknown enable_dlopen_self_static_unknown enable_fast_install=needless enable_fragment=yes enable_iproute2=no enable_libt
ool_lock=yes enable_lz4=yes enable_lzo=yes enable_maintainer_mode=no enable_management=yes enable_option_checking=no enable_pam_dlopen=no enable_pedantic=no enable_pk
cs11=yes enable_plugin_auth_pam=yes enable_plugin_down_root=yes enable_plugins=yes enable_port_share=yes enable_selinux=no enable_shared=yes enable_shared_with_static
_runtimes=no enable_silent_rules=no enable_small=no enable_static=yes enable_strict=no enable_strict_options=no enable_systemd=yes enable_unit_tests=no enable_werror=
no enable_win32_dll=yes enable_wolfssl_options_h=yes enable_x509_alt_username=yes with_aix_soname=aix with_crypto_library=openssl with_gnu_ld=yes with_mem_check=no wi
th_openssl_engine=auto with_sysroot=no
osboxes@osboxes:~$
```

Step 3: Check OpenVPN version

`openvpn --version`

What it does:

- asks the OpenVPN program to tell you what version it is

Behind the scenes:

It verifies that OpenVPN is properly installed and available on your path.

Step 4: Switch to root user

✓ What it does:

- asks the OpenVPN program to tell you what version it is

Behind the scenes:

It verifies that OpenVPN is properly installed and available on your path.

Step 4: Switch to root user

sudo su -

```
File Actions Edit View Help
osboxes@osboxes:~$ openvpn --version
OpenVPN 2.6.12 x86_64-pc-linux-gnu [SSL (OpenSSL)] [LZO] [LZ4] [EPOLL] [PKCS11] [MH/PKTINFO] [AEAD] [DCO]
Library versions: OpenSSL 3.3.1 4 Jun 2024, LZO 2.10
DCO version: N/A
Originally developed by James Yonan
Copyright (C) 2002-2024 OpenVPN Inc <sales@openvpn.net>
Compile time defines: enable_async_push=no enable_comp_stub=no enable_crypto_ofb_cfb=yes enable_dco=yes enable_dco_arg=yes enable_debug=yes enable_dependency_tracking
no enable_dlopen_unknown enable_dlopen_self=unknown enable_dlopen_self_static=unknown enable_fast_install=needless enable_fragment=yes enable_iproute2=no enable_libt
ool_lock=yes enable_lzo=yes enable_maintainer_mode=no enable_management=yes enable_option_checking=no enable_pam_dlopen=no enable_pedantic=no enable_pk
cs11=yes enable_plugin_auth_pam=yes enable_plugin_down_root=yes enable_plugins=yes enable_port_share=yes enable_selinux=no enable_shared=yes enable_shared_with_static
_runtimes=no enable_silent_rules=no enable_small=no enable_static=yes enable_strict=no enable_strict_options=no enable_systemd=yes enable_unit_tests=no enable_werror=
no enable_win32_dll=yes enable_wolfssl_options_h=yes enable_x509_alt_username=yes with_aix_soname=aix with_crypto_library=openssl with_gnu_ld=yes with_mem_check=no wi
th_openssl_engine=auto with_sysroot=no
osboxes@osboxes:~$ sudo su
[sudo] password for osboxes:
root@osboxes:~#
```

Step 4: Switch to root user

```
sudo su
```

What it does:

- `sudo` → again, "run as administrator"
- `su` → switch user to root

Behind the scenes:

- root has total control on Linux

✓ What it does:

- `sudo` → again, "run as administrator"
- `su` → switch user to root

Behind the scenes:

- root has total control on Linux
- you need root to change files under `/etc/openvpn`
→ normal users cannot write there.

Go to the OpenVPN folder

👉 Type this to move into the server config folder:

`cd /etc/openvpn/server`

```
File Actions Edit View Help
root@osboxes:/etc/openvpn# ls
client server update-resolv-conf
root@osboxes:/etc/openvpn# cd server
root@osboxes:/etc/openvpn/server# ls
root@osboxes:/etc/openvpn/server#
```



Why?

Because OpenVPN is designed to look for its server settings in this folder.
You want to put your config there.

Create the server.conf file with the config inside



Now, you will *paste* a block of text to create `server.conf`.

This means you will *tell* Ubuntu:

“Hey, copy everything I’m about to type into a new file.”



Type this carefully:

```
cat << EOF > server.conf
```

Then, *press ENTER*.



Your terminal will just sit there waiting for you to type. That is normal.

● STEP 4: Now paste these lines (copy them, then right-click to paste):

```
local 192.168.1.206
port 1194
proto udp4
dev tun
keepalive 10 120
topology subnet
server 10.8.0.0 255.255.255.0
verb 3
auth-nocache
user nobody
```

```
group nogroup
explicit-exit-notify
auth SHA512
persist-key
persist-tun
cipher AES-256-GCM
tls-server
remote-cert-eku "TLS Web Client Authentication"
ca ca.crt
cert server.crt
key server.key
tls-crypt tc.key
dh none
```

✅ *Don't worry* if you don't understand every line — these are standard VPN server rules, and I'll help you test them later.

● STEP 5: Tell the terminal you are finished

👉 Type this on a new line:

```
nginx
CopyEdit
EOF
```

✅ Then press ENTER.

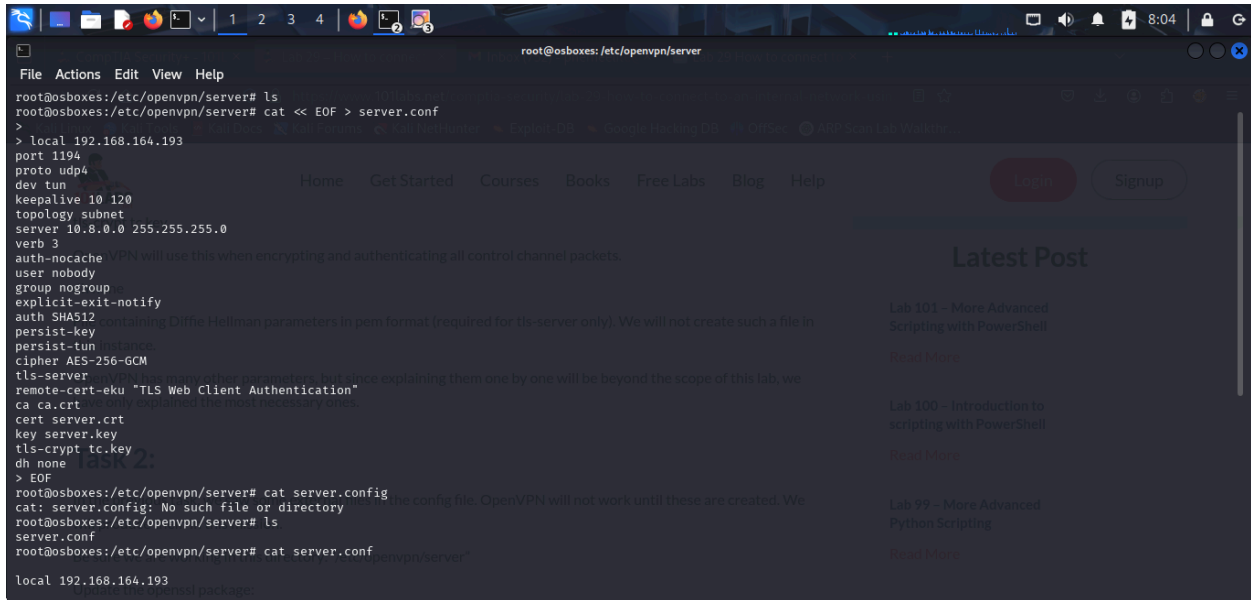
That tells the computer:

“OK, I finished typing my block, please save it now.”

STEP 6: Check your work

👉 Type:

cat server.conf



```
root@osboxes: /etc/openvpn/server
File Actions Edit View Help
root@osboxes:/etc/openvpn/server# ls
root@osboxes:/etc/openvpn/server# cat << EOF > server.conf
>
> local 192.168.164.193
port 1194
proto udp4
dev tun
keepalive 10 120
topology subnet
server 10.8.0.0 255.255.255.0
verb 3
auth-nocache VPN will use this when encrypting and authenticating all control channel packets.
user nobody
group nogroup
explicit-exit-notify
auth SHA512 containing Diffie-Hellman parameters in pem format (required for tls-server only). We will not create such a file in
persist-key
persist-tun
cipher AES-256-GCM
tls-server
remote-cert-eku "TLS Web Client Authentication"
ca ca.crt
cert server.crt
key server.key
tls-crypt tc.key
dh none
> EOF
root@osboxes:/etc/openvpn/server# cat server.conf
cat: server.conf: No such file or directory
root@osboxes:/etc/openvpn/server# ls
server.conf
root@osboxes:/etc/openvpn/server# cat server.conf
local 192.168.164.193
```

✓ This shows you what you just wrote, to confirm it went into the file correctly.

Step 5: Create OpenVPN server config

cat << EOF > /etc/openvpn/server/server.conf
(followed by the big block of config)
EOF

✓ What it does:

- `cat << EOF` → creates a *heredoc*, which takes text you type between `EOF` and puts it in a file

- `> server.conf` → means “write that text to server.conf”

Behind the scenes:

- `cat` is a Linux command to *print files*
- using `<< EOF` you can “type a file interactively” rather than using a text editor
→ Linux captures all your lines until it sees `EOF` and saves them to `server.conf`.

This is a quick way to paste a config without opening `nano` or `vi`.

Config explanation (server.conf)

Some critical lines:

- `local 192.168.1.206` → tells the server to listen on its own IP
- `port 1194` → default port for OpenVPN
- `proto udp4` → use UDP version 4 protocol
- `server 10.8.0.0 255.255.255.0` → gives out VPN client addresses in this subnet
- `cipher AES-256-GCM` → secure encryption algorithm
- `ca ca.crt` → file to verify client certificates
- `cert server.crt` → server’s own certificate
- `key server.key` → server’s private key
- `tls-crypt tc.key` → additional encryption for handshake
- `dh none` → no Diffie-Hellman because you use ECDHE

Behind the scenes:

When you launch the server, OpenVPN reads this file line-by-line to know:

- which IP to use
 - how to encrypt
 - what certificates to validate clients
→ This is what sets up the secure VPN server behavior.
-

Step 6: Create Root Certificate Authority (CA)

Cd into the root directory:

cd ~

Then run the below command;

```
openssl req -x509 -newkey rsa:4096 -keyout ca.key -sha256 -days 3650 \  
-set_serial 00 -out ca.crt -subj \  
"/C=UK/ST=UK/L=LONDON/O=101LABS/CN=101 Labs Root CA" \  
-addext nsComment="101 LABS Class 1 ROOT CA"
```

After running the above command, it would request for a password.

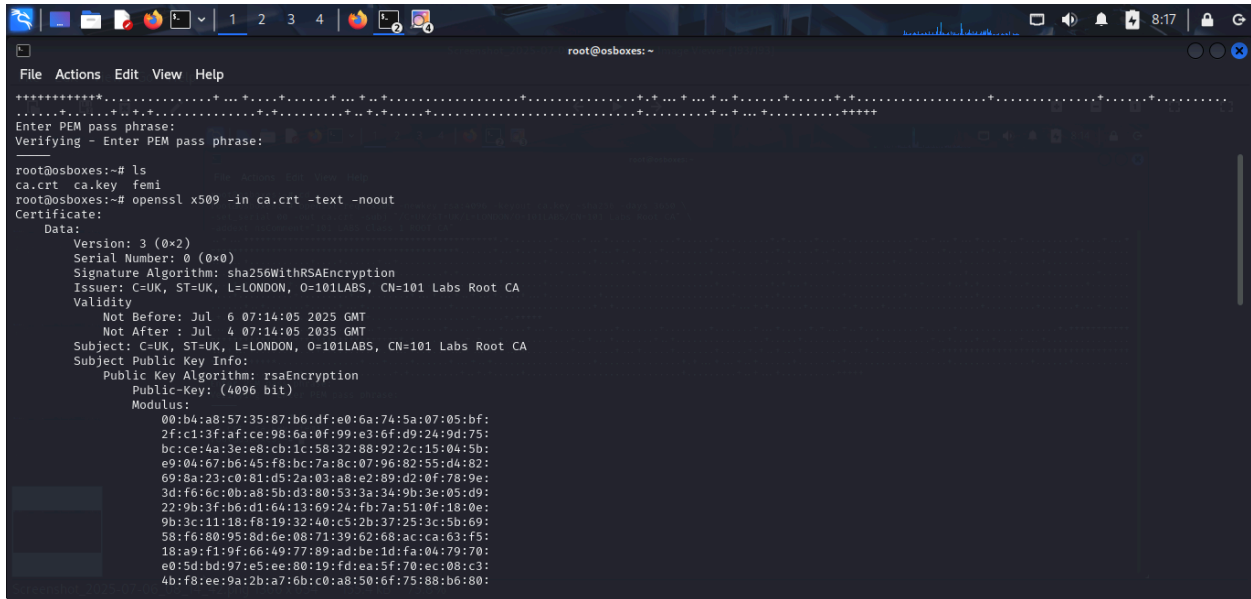
Input a password you know you would remember

✓ What it does:

- **openssl req** → start creating a certificate
- **-x509** → means *self-signed*
- **-newkey rsa:4096** → make a new key with RSA 4096-bit strength
- **-keyout ca.key** → save the private key to **ca.key**

✓ What it does:

- shows the human-readable info in the certificate
→ you verify it was made correctly



```
root@osboxes: ~  
File Actions Edit View Help  
+++++  
Enter PEM pass phrase:  
Verifying - Enter PEM pass phrase:  
root@osboxes:~# ls  
ca.crt ca.key femi  
root@osboxes:~# openssl x509 -in ca.crt -text -noout  
Certificate:  
Data:  
Version: 3 (0x2)  
Serial Number: 0 (0x0)  
Signature Algorithm: sha256WithRSAEncryption  
Issuer: C=UK, ST=UK, L=LONDON, O=101LABS, CN=101 Labs Root CA  
Validity  
Not Before: Jul 6 07:14:05 2025 GMT  
Not After : Jul 4 07:14:05 2035 GMT  
Subject: C=UK, ST=UK, L=LONDON, O=101LABS, CN=101 Labs Root CA  
Subject Public Key Info:  
Public Key Algorithm: rsaEncryption  
Public-Key: (4096 bit)  
Modulus:  
00:b4:a8:57:35:87:b6:df:e0:6a:74:5a:07:05:bf:  
2f:c1:3f:af:ce:98:6a:0f:99:e3:6f:d9:24:9d:75:  
bc:ce:4a:3e:e8:cb:1c:58:32:88:92:2c:15:04:5b:  
e9:04:67:b6:45:f8:bc:7a:8c:07:96:82:55:d4:82:  
69:8a:23:c0:81:d5:2a:03:a8:e2:89:d2:0f:78:9e:  
3d:f6:6c:0b:a8:5b:d3:80:53:3a:34:9b:3e:05:d9:  
22:9b:3f:b6:d1:64:13:69:24:fb:7a:51:0f:18:0e:  
9b:3c:11:18:f8:19:32:40:c5:2b:37:25:3c:5b:69:  
58:f6:80:95:8d:6e:08:71:39:62:68:ac:ca:63:f5:  
18:a0:f1:9f:66:49:77:80:ad:be:1d:fa:04:79:70:  
e0:5d:bd:97:e5:ee:80:19:fd:ea:5f:70:ec:08:c3:  
4b:f8:ee:9a:2b:a7:6b:c0:a8:50:6f:75:88:b6:80:
```

Step 8: Make server certificate signing request

```
openssl req -new -newkey rsa:2048 -nodes -keyout server.key -out  
server.csr \  
-subj "/C=UK/ST=UK/L=LONDON/O=LAB101/CN=SERVER"
```

✓ What it does:

- asks to make a new private key (2048 bits)
- outputs the key to `server.key`
- makes a *certificate signing request* to `server.csr`

- [illegible]

→ a CSR says “Hey CA, please sign me so I’m trusted!”
it does not sign itself, it waits for the CA to approve.

Run this to open a simple editor (like nano):

✓ Then copy and paste this text inside nano:

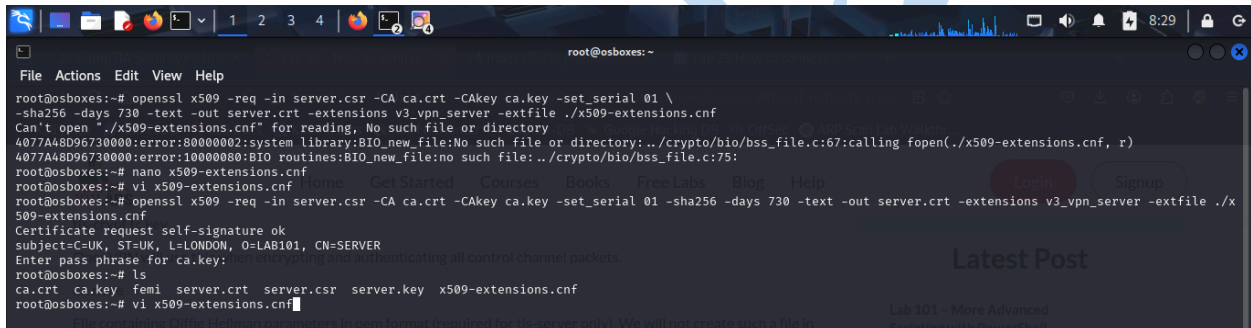
```
extendedKeyUsage = serverAuth
```

Step 9: Sign server certificate

```
openssl x509 -req -in server.csr -CA ca.crt -CAkey ca.key -set_serial 01 \
-sha256 -days 730 -text -out server.crt -extensions v3_vpn_server
-extfile ./x509-extensions.cnf
```

✓ What it does:

- uses CA to sign the server's CSR
- produces server.crt
- sets a serial number (01)
- uses rules from `v3_vpn_server` in the extensions file
→ this certificate is now trusted by the CA



```
root@osboxes:~# openssl x509 -req -in server.csr -CA ca.crt -CAkey ca.key -set_serial 01 \
-sha256 -days 730 -text -out server.crt -extensions v3_vpn_server -extfile ./x509-extensions.cnf
Can't open './x509-extensions.cnf' for reading: No such file or directory
4077A48D96730000:error:80000002:system library:BIO_new_file:No such file or directory:../crypto/bio/bss_file.c:67:calling fopen('./x509-extensions.cnf', r)
4077A48D96730000:error:10000000:BIO routines:BIO_new_file:no such file:../crypto/bio/bss_file.c:75:
root@osboxes:~# nano x509-extensions.cnf
root@osboxes:~# vi x509-extensions.cnf
root@osboxes:~# openssl x509 -req -in server.csr -CA ca.crt -CAkey ca.key -set_serial 01 -sha256 -days 730 -text -out server.crt -extensions v3_vpn_server -extfile ./x
509-extensions.cnf
Certificate request self-signature ok
subject=C=UK, ST=UK, L=LONDON, O=LAB101, CN=SERVER
Enter pass phrase for ca.key:
root@osboxes:~# ls
ca.crt ca.key femi server.crt server.csr server.key x509-extensions.cnf
root@osboxes:~# vi x509-extensions.cnf
```

Behind the scenes:

OpenVPN will later match the server's `server.crt` to clients by checking if it was signed by `ca.crt`.

Step 10: Make client certificate signing request:

1 : Create the client key + CSR

✓ Type this as root (or stay in the same folder as before):

```
openssl req -new -newkey rsa:2048 -nodes -keyout client.key -out  
client.csr \  
-subj "/C=UK/ST=UK/L=LONDON/O=LAB101/CN=CLIENT"
```

✅ This does exactly the same as for the server:

- makes a **client private key** (`client.key`)
- makes a **certificate signing request** (`client.csr`)

But this time with CN=CLIENT.

👉 The CN (common name) is how you know this certificate is for the client.

2: Create a client extensions file

👉 We'll need to tell OpenSSL "this is a client certificate, not a server certificate"

✅ Make a new file:

```
vix509-client-extensions.cnf
```

✅ Then paste this:

```
[ v3_vpn_client ]  
basicConstraints = CA:FALSE  
nsCertType = client  
keyUsage = digitalSignature  
extendedKeyUsage = clientAuth
```

3: Sign the client certificate

✅ Use your CA to sign the CSR:

```
openssl x509 -req -in client.csr -CA ca.crt -CAkey ca.key -set_serial  
02 \  
-sha256 -days 730 -text -out client.crt -extensions v3_vpn_client  
-extfile ./x509-client-extensions.cnf
```

👉 **Explanation:**

- `-in client.csr` → sign the client's request
- `-CA ca.crt -CAkey ca.key` → use the same CA to approve it
- `-set_serial 02` → give this cert serial 2 (the server had 1)
- `-out client.crt` → your finished client certificate
- `-extfile x509-client-extensions.cnf` → use client-specific rules

✅ It will ask you for the **CA passphrase** again → type it in.

4: Confirm what you have

👉 Check your directory:

```
ls
```

(similar steps, just changes `client` instead of `server`)

✅ You repeat:

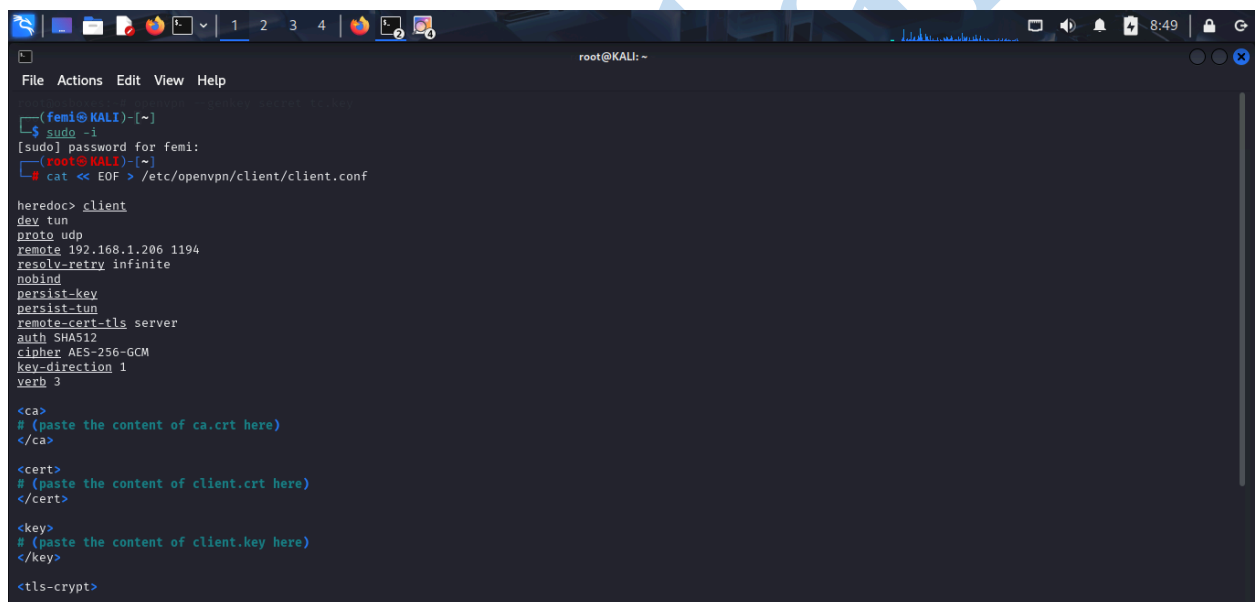
- generate client key + client CSR
- sign it with the CA
→ then you have client.crt

Step 12: Build client config on Kali

```
cat << EOF > /etc/openvpn/client/client.conf
# (pasted config lines)
EOF
```

✓ What it does:

- writes a config file for the Kali machine, just like you did for the server
- points to correct certificates and server IP



The screenshot shows a terminal window on a Kali Linux system. The user is in a shell as 'femi@KALI'. They run 'sudo -i' to become root. Then, they use 'cat << EOF > /etc/openvpn/client/client.conf' to create a configuration file. The configuration is pasted from a heredoc, starting with 'client' and ending with 'tls-crypt'. The config includes settings for tun mode, UDP protocol, remote server IP (192.168.1.206), and various security and performance options. It also includes placeholders for certificates and keys, such as '<ca> # (paste the content of ca.crt here) </ca>'. The terminal window has a dark theme and standard Linux window controls.

```
root@KALI: ~
File Actions Edit View Help

(femi@KALI)-[~]
$ sudo -i
[sudo] password for femi:
(root@KALI)-[~]
$ cat << EOF > /etc/openvpn/client/client.conf

heredoc> client
dev tun
proto udp
remote 192.168.1.206 1194
resolv-retry infinite
nobind
persist-key
persist-tun
remote-cert-tls server
auth SHA512
cipher AES-256-GCM
key-direction 1
verb 3

<ca>
# (paste the content of ca.crt here)
</ca>

<cert>
# (paste the content of client.crt here)
</cert>

<key>
# (paste the content of client.key here)
</key>

<tls-crypt>
```

Behind the scenes:

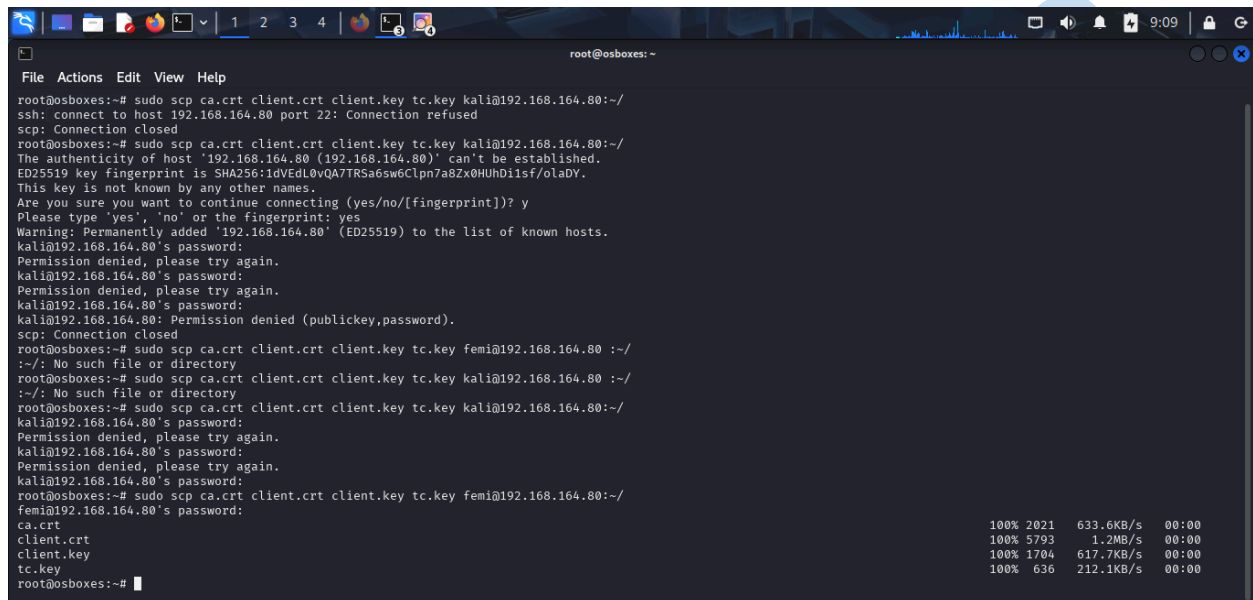
this tells OpenVPN on Kali how to talk to your Ubuntu server securely.

Step 13: Move files to Kali with SCP

```
sudo scp ca.crt client.crt client.key tc.key kali@192.168.164.80:~/
```

✓ What it does:

- uses secure copy (**scp**) to transfer certificates
- Kali needs these files to validate the server



```
root@osboxes: ~  
File Actions Edit View Help  
root@osboxes:~# sudo scp ca.crt client.crt client.key tc.key kali@192.168.164.80:~/  
ssh: connect to host 192.168.164.80 port 22: Connection refused  
scp: Connection closed  
root@osboxes:~# sudo scp ca.crt client.crt client.key tc.key kali@192.168.164.80:~/  
The authenticity of host '192.168.164.80 (192.168.164.80)' can't be established.  
ED25519 key fingerprint is SHA256:1dVEDL0vQA7TRSa6sw6Clpn7a8Zx0HUHd1isf/olaDY.  
This key is not known by any other names.  
Are you sure you want to continue connecting (yes/no/[fingerprint])? y  
Please type 'yes', 'no' or the fingerprint: yes  
Warning: Permanently added '192.168.164.80' (ED25519) to the list of known hosts.  
kali@192.168.164.80's password:  
Permission denied, please try again.  
kali@192.168.164.80's password:  
Permission denied, please try again.  
kali@192.168.164.80's password:  
kali@192.168.164.80: Permission denied (publickey,password).  
scp: Connection closed  
root@osboxes:~# sudo scp ca.crt client.crt client.key tc.key femi@192.168.164.80 :~/  
:~/: No such file or directory  
root@osboxes:~# sudo scp ca.crt client.crt client.key tc.key kali@192.168.164.80 :~/  
:~/: No such file or directory  
root@osboxes:~# sudo scp ca.crt client.crt client.key tc.key kali@192.168.164.80:~/  
kali@192.168.164.80's password:  
Permission denied, please try again.  
kali@192.168.164.80's password:  
Permission denied, please try again.  
kali@192.168.164.80's password:  
root@osboxes:~# sudo scp ca.crt client.crt client.key tc.key femi@192.168.164.80:~/  
femi@192.168.164.80's password:  
ca.crt                                100% 2021    633.6KB/s   00:00  
client.crt                           100% 5793      1.2MB/s   00:00  
client.key                           100% 1704    617.7KB/s   00:00  
tc.key                               100% 636     212.1KB/s   00:00  
root@osboxes:~#
```

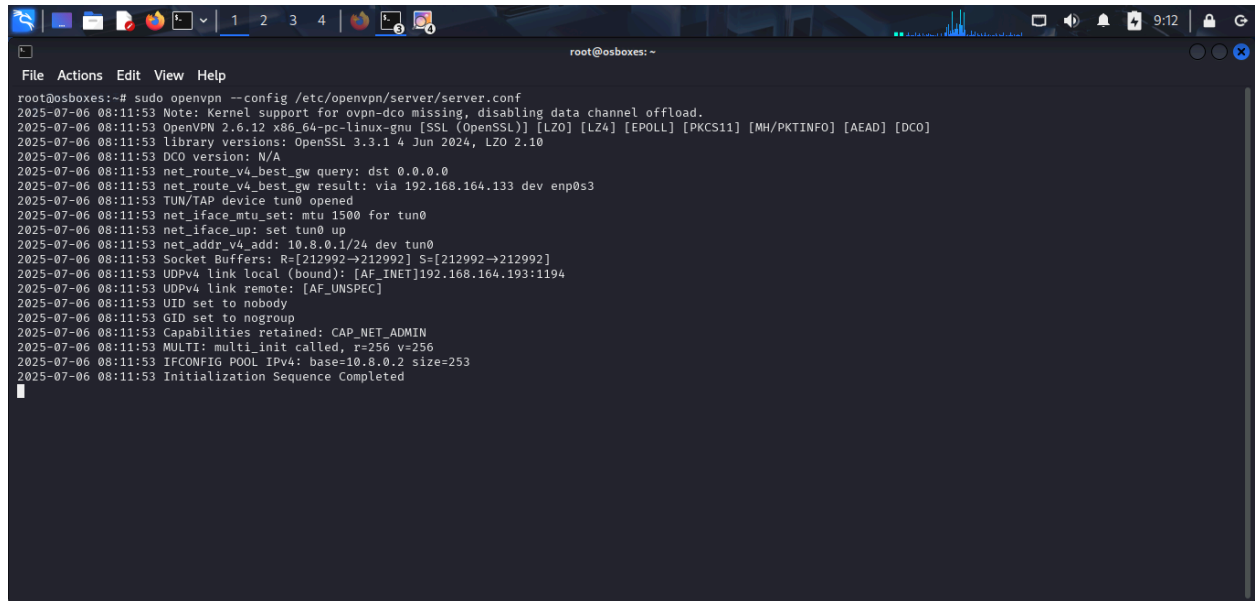
Behind the scenes:

no certificates = no VPN.
SCP uses SSH, so the copy is encrypted.

Step 14: Start the VPN tunnel

✓ On server:

`sudo openvpn --config /etc/openvpn/server/server.conf`



A terminal window titled 'root@osboxes: ~' showing the output of the command 'sudo openvpn --config /etc/openvpn/server/server.conf'. The window has a menu bar with 'File', 'Actions', 'Edit', 'View', and 'Help'. The terminal output consists of a series of log messages with timestamps from 2025-07-06 08:11:53. The messages include a note about kernel support for ovpn-dco, OpenVPN version information, library versions (OpenSSL 3.3.1, LZ4, EPOLL, PKCS11), DCO version (N/A), net_route_v4_best_gw query results, TUN/TAP device opening, net_iface_mtu setting, net_iface_up setting, net_addr_v4_add, Socket Buffers, UDPv4 link local (bound), UDPv4 link remote, UID set to nobody, GID set to nogroup, Capabilities retained (CAP_NET_ADMIN), MULTI: multi_init called, IFCONFIG POOL IPv4 base and size, and finally 'Initialization Sequence Completed'.

```
root@osboxes:~# sudo openvpn --config /etc/openvpn/server/server.conf
2025-07-06 08:11:53 Note: Kernel support for ovpn-dco missing, disabling data channel offload.
2025-07-06 08:11:53 OpenVPN 2.6.12 x86_64-pc-linux-gnu [SSL (OpenSSL)] [LZO] [LZ4] [EPOLL] [PKCS11] [MH/PKTINFO] [AEAD] [DCO]
2025-07-06 08:11:53 library versions: OpenSSL 3.3.1 4 Jun 2024, LZ4 2.10
2025-07-06 08:11:53 DCO version: N/A
2025-07-06 08:11:53 net_route_v4_best_gw query: dst 0.0.0.0
2025-07-06 08:11:53 net_route_v4_best_gw result: via 192.168.164.133 dev enp0s3
2025-07-06 08:11:53 TUN/TAP device tun0 opened
2025-07-06 08:11:53 net_iface_mtu_set: mtu 1500 for tun0
2025-07-06 08:11:53 net_iface_up: set tun0 up
2025-07-06 08:11:53 net_addr_v4_add: 10.8.0.1/24 dev tun0
2025-07-06 08:11:53 Socket Buffers: R=[212992→212992] S=[212992→212992]
2025-07-06 08:11:53 UDPv4 link local (bound): [AF_INET]192.168.164.193:1194
2025-07-06 08:11:53 UDPv4 link remote: [AF_UNSPEC]
2025-07-06 08:11:53 UID set to nobody
2025-07-06 08:11:53 GID set to nogroup
2025-07-06 08:11:53 Capabilities retained: CAP_NET_ADMIN
2025-07-06 08:11:53 MULTI: multi_init called, r=256 v=256
2025-07-06 08:11:53 IFCONFIG POOL IPv4: base=10.8.0.2 size=253
2025-07-06 08:11:53 Initialization Sequence Completed
```

✓ On client:

`sudo openvpn --config /etc/openvpn/client/client.conf`

```
root@KALI: /home/femi
File Actions Edit View Help
root@KALI: /home/femi
sudo openvpn --config /etc/openvpn/client/client.conf

2025-07-06 09:27:42 Note: Kernel support for vpn-dco missing, disabling data channel offload.
2025-07-06 09:27:42 OpenVPN 2.6.13 x86_64-pc-linux-gnu [SSL (OpenSSL)] [LZO] [LZ4] [EPOLL] [PKCS11] [MH/PKTINFO] [AEAD] [DCO]
2025-07-06 09:27:42 library versions: OpenSSL 3.4.1 11 Feb 2025, LZO 2.10
2025-07-06 09:27:42 DCO version: N/A
2025-07-06 09:27:42 TCP/UDP: Preserving recently used remote address: [AF_INET]192.168.164.193:1194
2025-07-06 09:27:42 Socket Buffers: R=[212992->212992] S=[212992->212992]
2025-07-06 09:27:42 UDPv4 link local: (not bound)
2025-07-06 09:27:42 UDPv4 link remote: [AF_INET]192.168.164.193:1194
2025-07-06 09:27:42 TLS: Initial packet from [AF_INET]192.168.164.193:1194, sid=f194eb5f 469f0a3c
2025-07-06 09:27:42 VERIFY OK: depth=1, C=UK, ST=UK, L=LONDON, O=101LABS, CN=101 Labs Root CA
2025-07-06 09:27:42 VERIFY KU OK
2025-07-06 09:27:42 Validating certificate extended key usage
2025-07-06 09:27:42 ++ Certificate has EKU (str) TLS Web Server Authentication, expects TLS Web Server Authentication
2025-07-06 09:27:42 VERIFY ECU OK
2025-07-06 09:27:42 VERIFY OK: depth=0, C=UK, ST=UK, L=LONDON, O=LAB101, CN=SERVER
2025-07-06 09:27:42 Control Channel: TLSv1.3, cipher TLSv1.3 TLS_AES_256_GCM_SHA384, peer certificate: 2048 bits RSA, signature: RSA-SHA256, peer temporary key: 253 b
its X25519
2025-07-06 09:27:42 [SERVER] Peer Connection Initiated with [AF_INET]192.168.164.193:1194
2025-07-06 09:27:42 TLS: move_session: dest=TM_ACTIVE src=TM_INITIAL reinit_src=1
2025-07-06 09:27:42 TLS: tls_multi_process: initial untrusted session promoted to trusted
2025-07-06 09:27:42 PUSH: Received control message: 'PUSH_REPLY,route-gateway 10.8.0.1,topology subnet,ping 10,ping-restart 120,ifconfig 10.8.0.2 255.255.255.0,peer-i
d 0,cipher AES-256-GCM,protocol-flags cc-exit tls-ekm dyn-tls-crypt,tun-mtu 1500'
2025-07-06 09:27:42 OPTIONS IMPORT: --ifconfig/up options modified
2025-07-06 09:27:42 OPTIONS IMPORT: route-related options modified
2025-07-06 09:27:42 OPTIONS IMPORT: tun-mtu set to 1500
2025-07-06 09:27:42 TUN/TAP device tun0 opened
2025-07-06 09:27:42 net_iface_mtu_set: mtu 1500 for tun0
2025-07-06 09:27:42 net_iface_up: set tun0 up
2025-07-06 09:27:42 net_addr_v4_add: 10.8.0.2/24 dev tun0
2025-07-06 09:27:42 Initialization Sequence Completed
2025-07-06 09:27:42 Data Channel: cipher 'AES-256-GCM', peer-id: 0
```

Behind the scenes:

OpenVPN starts up, reads its config, loads the certificates, and opens a secure tunnel.

- it sets up a new network interface (**tun0**)
- packets going to the other side go through the tunnel
- the server assigns a VPN IP like 10.8.0.2

Step 15: Check it works

✓ On the client, you might do:

ping 10.8.0.1

```
root@KALI: ~  
File Actions Edit View Help  
root@KALI: ~  
ping 10.8.0.1  
PING 10.8.0.1 (10.8.0.1) 56(84) bytes of data:  
64 bytes from 10.8.0.1: icmp_seq=1 ttl=64 time=2.60 ms  
64 bytes from 10.8.0.1: icmp_seq=2 ttl=64 time=1.61 ms  
64 bytes from 10.8.0.1: icmp_seq=3 ttl=64 time=1.74 ms  
64 bytes from 10.8.0.1: icmp_seq=4 ttl=64 time=2.08 ms  
64 bytes from 10.8.0.1: icmp_seq=5 ttl=64 time=1.85 ms  
64 bytes from 10.8.0.1: icmp_seq=6 ttl=64 time=1.93 ms  
64 bytes from 10.8.0.1: icmp_seq=7 ttl=64 time=2.29 ms  
64 bytes from 10.8.0.1: icmp_seq=8 ttl=64 time=1.66 ms  
64 bytes from 10.8.0.1: icmp_seq=9 ttl=64 time=1.81 ms  
64 bytes from 10.8.0.1: icmp_seq=10 ttl=64 time=1.33 ms  
64 bytes from 10.8.0.1: icmp_seq=11 ttl=64 time=1.84 ms  
64 bytes from 10.8.0.1: icmp_seq=12 ttl=64 time=2.06 ms  
64 bytes from 10.8.0.1: icmp_seq=13 ttl=64 time=2.47 ms  
64 bytes from 10.8.0.1: icmp_seq=14 ttl=64 time=1.87 ms  
64 bytes from 10.8.0.1: icmp_seq=15 ttl=64 time=1.98 ms  
64 bytes from 10.8.0.1: icmp_seq=16 ttl=64 time=1.75 ms  
64 bytes from 10.8.0.1: icmp_seq=17 ttl=64 time=1.30 ms  
64 bytes from 10.8.0.1: icmp_seq=18 ttl=64 time=3.63 ms  
64 bytes from 10.8.0.1: icmp_seq=19 ttl=64 time=1.81 ms  
64 bytes from 10.8.0.1: icmp_seq=20 ttl=64 time=1.38 ms  
64 bytes from 10.8.0.1: icmp_seq=21 ttl=64 time=1.35 ms  
64 bytes from 10.8.0.1: icmp_seq=22 ttl=64 time=3.83 ms  
64 bytes from 10.8.0.1: icmp_seq=23 ttl=64 time=1.74 ms  
64 bytes from 10.8.0.1: icmp_seq=24 ttl=64 time=1.68 ms  
64 bytes from 10.8.0.1: icmp_seq=25 ttl=64 time=1.79 ms  
^C  
--- 10.8.0.1 ping statistics ---  
25 packets transmitted, 25 received, 0% packet loss, time 24047ms  
rtt min/avg/max/mdev = 1.295/1.973/3.827/0.605 ms
```

→ tests if you can talk to the VPN server across the tunnel.

What's really happening behind the scenes?

✓ You:

- build a **mini certificate authority**
- issue your own trusted certificates
- configure a secure VPN server
- configure a secure VPN client
- transfer the trust files
- connect over a *TLS*-protected tunnel

Every command basically revolves around:

- Making keys
 - Making certificates
 - Signing them
 - Putting them in a config
 - Starting the tunnel
-



In total:

- 👉 You control the server (Ubuntu)
- 👉 You control the client (Kali)
- 👉 You trust yourself with your own certificates
- 👉 You connect them together **securely** with OpenVPN