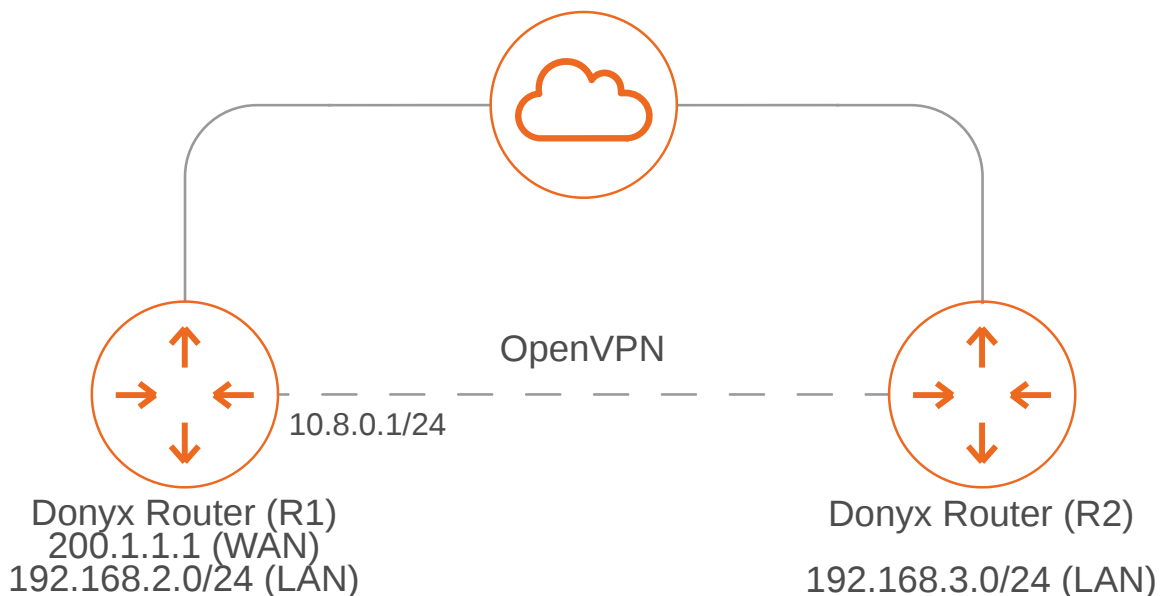


OpenVPN Server and Client Configuration on Donyx Routers

The objective is to establish a tunnel and ensure mutual connectivity between remote local networks. This configuration involves setting up one router as a **server** and the other as a **client**.

A public ("white") IP address is required only for the router acting as the **server**. The **client** router may utilize a private ("gray") IP address and can operate behind a **NAT** gateway.

OpenVPN Server-Client Network Topology:



In this example, router *R1* acts as the **server** with the public IP address *200.1.1.1* and the local network *192.168.2.0/24*. The client router *R2* has a local network with addresses in the *192.168.3.0/24* range.



The configuration of **OpenVPN** requires the presence of valid certificates for both the client and the server on the router. For instructions on how to create, export, and import certificates in *dnxOS*, please refer to the dedicated article.

Certificates may be generated using a standalone **Public Key Infrastructure (PKI)** (e.g., *easy-rsa*) or through the router's internal interface.

Configuration for Router R1 (Server)

Navigate to the `/service/openvpn-server` section. An example of the server configuration is illustrated in the figure below.

Disabled	<input type="checkbox"/>	Disable configuration
Dev-Type	tun	Virtual interface type
Protocol	udp	Tunnel transport protocol
Port	1194	Listening port number
Tunnel IP	10.8.0.1/24	Local tunnel IP address
Pool	10.8.0.10-10.8.0.20	Client IP address assignment range
Cipher	...select one or more...	Data channel encryption algorithm
	⊖ AES-256-CBC	
Auth	...select one or more...	Authentication algorithm
	⊖ SHA1	
TA Key		Hash-based Message Authentication Code (HMAC) key for packet validation
Ca	CA	Certificate Authority (CA) public key
Cert	server	Local Transport Layer Security (TLS) certificate bundle (cert + key)
Keepalive	10 60	
Push To Client		Configure keepalive ping and auto-restart on link failure
	⊖ route-gateway=10.8.0.1	⊖ route=192.168.2.0/24
	⊖ topology=subnet	
Flag	...select one or more...	Tunnel advanced configuration flags
	⊖ duplicate-cn	
Extra Parameters		Tunnel extra parameters (custom options)

Table 1. OpenVPN Server Parameters

Field	Value
Dev-Type	<i>Tun</i>
Protocol	<i>UDP</i> (default).
Port	<i>1194</i> (default).
Tunnel IP	<i>10.8.0.1/24</i>
Pool	<i>10.8.0.10-10.8.0.20</i>
Cipher	<i>AES-256-CBC</i> (default).
Auth	<i>SHA1</i> (default).
TA Key	(Leave blank).
CA	CA file name (in this example: <i>CA</i>).
Cert	Server certificate name (in this example: <i>server</i>).
Keepalive	<i>10 60</i>
Push To Client	<i>topology=subnet</i> (default); <i>route=192.168.2.0/24</i> (the route to the local network of router <i>R1</i> is pushed to the client); <i>route-gateway=10.8.0.1</i> (the tunnel address of router <i>R1</i> is assigned as the gateway for the pushed route).
Flag	<i>duplicate-cn</i> (default).

Click **Apply** to activate the configuration.

CLI Configuration (server)



Please note that due to the requirement for pre-installed certificates, configuration via the CLI can be more complex than the web-based setup, although both interfaces utilize the same underlying parameters.

```
/service openvpn-server
  auth -
  auth SHA1
  ca CA
  cert server
  cipher -
  cipher AES-256-CBC
  dev-type tun
  disabled -
  extra -
  flag -
  flag duplicate-cn
  keepalive 10 60
  pool 10.8.0.10-10.8.0.20
  port 1194
  protocol udp
  push -
  push route-gateway=10.8.0.1,route=192.168.2.0/24,topology=subnet
  ta-key -
  tunnel-ip 10.8.0.1/24

/service openvpn-server apply
```

Creating a Client Account

The client account is created in the `/service/client` section.

1. Click the **Add** button.
2. Specify the **Username**.



The **Username** must strictly match the **Common Name (CN)** of the certificate used by the client connecting to the **OpenVPN** server.

Configure the connecting client parameters:

1. In the **Service** field, select *openvpn*.
2. In the **Tunnel IP** field, the IP address and subnet mask (from the *Pool* range) can be specified for this client. If this field is left blank, the address is assigned automatically.
3. In the **Route** field, specify the local subnet of router *R2* that should be accessible from router *R1* (in this example, *192.168.3.0/24*).
4. Click **Apply**.

If multiple unique clients need to connect to **OpenVPN**, corresponding entries must be created in the `/service/client` section with unique **Username**s that match the **Common Name (CN)** of their respective certificates. If connectivity to the clients' local networks is required, their specific subnets must be specified in the **Route** field for each individual client account.

The **OpenVPN Server** can now be started in the `/service/openvpn-server` section. Uncheck the **Disabled** box and click **Apply**. The tunnel status will change to *running*.

client	
ip-address	10.8.0.1/24
listen	0.0.0.0:1194[udp]
mtu	1500
rx-tx	0.00KB/0.00KB
state	running
uptime	00:00:07

Configuration for Router R2 (Client)

Navigate to the `/tunnel/openvpn` section. An example of the client configuration is illustrated in the figure below.

OpenVPN

Disabled

Local IP auto

Remote IP

200.1.1.1

Tunnel IP

Device Type tun

Protocol udp

Encryption tls

Cipher ...select one or more...
AES-256-CBC

Auth ...select one or more...
SHA1

TA Key

Ca CA.pfx_0

Cert OpenVPN-Client.pfx_0

Username

Password

Flag ...select one or more...
pull

Extra Parameters

Table 2. OpenVPN Client Parameters

Field	Value
Local IP	<i>WAN</i>
Remote IP	<i>200.1.1.1</i>
Tunnel IP	<i>(Leave blank).</i>
Device Type	<i>tun</i>
Protocol	<i>udp</i>
Encryption	<i>tls</i>
Cipher	<i>AES-256-CBC</i>
Auth	<i>SHA1</i>
TA Key	<i>Not used in this example.</i>
CA	<i>CA certificate filename (e.g., CA.pfx_0).</i>
Cert	<i>Client certificate filename (e.g., OpenVPN-Client.pfx_0).</i>
Username	<i>Not used in this example.</i>
Password	<i>Not used in this example.</i>
Flag	<i>Pull (default).</i>
Extra Parameters	<i>Not used in this example.</i>

The *CA* and *Cert* filenames differ because they were generated on the *dnxOS* server and exported, which automatically modified the file identifiers.



To import certificates onto the router, first upload the files to the */storage/file* section. Then use the import function in the */storage/certificate/cert_import* section. For more information, please refer to the dedicated article.

To complete the configuration, uncheck the **Disabled** box and click **Apply**.

Once the tunnel is established, its status will be displayed as *running*:

ip-address	10.8.0.15/24
mtu	1500
name	OpenVPN
rx-tx	0.00KB/0.00KB
state	running
uptime	00:00:09

CLI Configuration (client)

```

/tunnel openvpn add name=OpenVPN
  auth -
  auth SHA1
  ca ca.crt
  cert client-bundle.pem
  cipher -
  cipher AES-256-CBC
  dev-type tun
  disabled -
  encryption tls
  extra -
  flag -
  flag pull
  local-ip auto
  password -
  protocol udp
  remote-ip -
  remote-ip 200.1.1.1
  ta-key -
  tunnel-ip -
  username -
  apply

```

Verify that the client has received the required routes from the server by checking the routing table in the `/ip/route/list` section.

NAME	STATE	DST-ADDR	INTERFACE	GATEWAY	METRIC	TABLE	SRC-ADDR	TYPE
1	D	10.8.0.0/24	OpenVPN		0	main	10.8.0.15	unicast
2	D	192.168.1.0/24	bridge0		0	main	192.168.1.2	unicast
3	D	192.168.2.0/24	OpenVPN	10.8.0.1	0	main	10.8.0.15	unicast
4	D	192.168.3.0/24	bridge1		200	main		unicast
5	D	200.1.1.0/24	WAN		200	main		unicast

Confirm that route #1 is correctly associated with the tunnel interface and route #3 is established to the R1 local network through the tunnel interface.



By default, all traffic for tunnel interfaces is permitted. If required, firewall rules can be configured in the `/firewall/filter` section.

CLI Configuration

```
admin@Router[/ip route list]>
```

NAME	STATE	DST-ADDR	INTERFACE	GATEWAY	METRIC	TABLE	SRC-ADDR
1	D	10.8.0.0/24	OpenVPN		0	main	10.8.0.15
2	D	192.168.1.0/24	bridge0		0	main	192.168.1.2
3	D	192.168.2.0/24	OpenVPN	10.8.0.1	0	main	10.8.0.15
4	D	192.168.3.0/24	bridge1		200	main	
5	D	200.1.1.0/24	WAN		200	main	

Ping (/tools/ping) — from client R2 to the server's local address

```

▶ Again ✕ Stop ✕ Close

PING 192.168.2.1 (192.168.2.1) 56(84) bytes of data.
64 bytes from 192.168.2.1: icmp_req=1 ttl=64 time=2.27 ms
64 bytes from 192.168.2.1: icmp_req=2 ttl=64 time=2.06 ms
64 bytes from 192.168.2.1: icmp_req=3 ttl=64 time=2.00 ms
64 bytes from 192.168.2.1: icmp_req=4 ttl=64 time=1.99 ms
64 bytes from 192.168.2.1: icmp_req=5 ttl=64 time=3.01 ms
64 bytes from 192.168.2.1: icmp_req=6 ttl=64 time=2.12 ms
64 bytes from 192.168.2.1: icmp_req=7 ttl=64 time=2.01 ms
64 bytes from 192.168.2.1: icmp_req=8 ttl=64 time=2.00 ms
64 bytes from 192.168.2.1: icmp_req=9 ttl=64 time=2.07 ms
64 bytes from 192.168.2.1: icmp_req=10 ttl=64 time=2.13 ms
--- 192.168.2.1 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 27ms
rtt min/avg/max/mdev = 1.994/2.169/3.010/0.296 ms, ipg/ewma 3.024/2.185 ms
Finished

```

Additional Hub-and-Spoke Configuration

Currently, *dnxOS* does not support pushing multiple routes to clients automatically, which may require manual configuration in certain scenarios. When connecting two or more clients to an **OpenVPN server** with the requirement for mutual access between their local networks, the following procedure is implemented.

Assume the local network for Client #2 (router *R3*) is *192.168.4.0/24*.

1. Create a certificate for Client #2 (router *R3*) on the server router *R1* or via a standalone **PKI** system.
2. In the */service/client* section of the server, create an additional client record and specify the corresponding LAN subnet in the **Route** field.
3. Configure the **OpenVPN** connection on the second client according to the standard procedure.

Upon completion, Client #2 receives the route to the server's subnet (*192.168.2.0/24*) from the server. Server *R1* identifies the route to the *R3* subnet (*192.168.4.0/24*) through the **Route** parameter in the client record.

Additionally, a manual static route to the local network of Client *R2* (*192.168.3.0/24*) must be configured on Client *R3* via the **OpenVPN** tunnel interface, using *10.8.0.1* as the gateway.

Target: 192.168.3.0/24

Source Interface: OpenVPN_New

Gateway: 10.8.0.1

Metric: 0

Disabled:

Target: 192.168.3.0/24

Gateway: 10.8.0.1

Metric: 0

Table: main

Source Address:

Type: unicast

Source Interface: OpenVPN_New

CLI Configuration

```
/ip route list add dst-addr=192.168.3.0/24 interface=OpenVPN_New
disabled -
gateway 10.8.0.1
metric -
src-addr -
table main
type unicast
apply
```

5. Additionally, on Client *R2*, a static route to the *R3* network (*192.168.4.0/24*) must be configured via the **OpenVPN** interface, using *10.8.0.1* as the gateway.

A configuration dialog box with a dark background. At the top left, there are two buttons: 'OK' with a play icon and 'Close' with a close icon. Below the buttons are four input fields:

- Target: 192.168.4.0/24
- Source Interface: OpenVPN (dropdown menu)
- Gateway: 10.8.0.1
- Metric: 0

A configuration dialog box with a dark background. At the top left, there is a 'Disabled' checkbox which is currently unchecked. Below it are several input fields:

- Target: 192.168.4.0/24
- Gateway: 10.8.0.1
- Metric: 0
- Table: main (dropdown menu)
- Source Address: (empty text field)
- Type: unicast (dropdown menu)
- Source Interface: OpenVPN (dropdown menu)

CLI Configuration

```
/ip route list add dst-addr=192.168.4.0/24 interface=OpenVPN
disabled -
gateway 10.8.0.1
metric -
src-addr -
table main
type unicast
apply
```

After configuration, all three nodes and their respective local networks have full **connectivity**.



All modifications are permanently saved to the router configuration only after executing the `/system config commit` command or clicking the **commit** button in the web interface.