

6. Scheduled backup (HQ-SAM-2) Configure a scheduled backup of the /share/users directory from HQ-DC to /backup/users on HQ-SAM-2 every 5 minutes. The backup must be a tar.gz archive including a timestamp in the format: YYYY-MM-DD-HH-MM-SS.tar.gz.

ChatGPT Plus

Great — here's exactly how to configure a **scheduled backup every 5 minutes** from:

- **Source:** /share/users (on **HQ-DC**)
- **Destination:** /backup/users (on **HQ-SAM-2**)
- **Format:** YYYY-MM-DD-HH-MM-SS.tar.gz

We'll assume:

- You have **SSH access** from HQ-DC to HQ-SAM-2 using **key-based authentication** (passwordless).
- You want to **run the cron job on HQ-DC**, and **send the archive to HQ-SAM-2**.

✓ 1. Set up SSH key-based authentication (HQ-DC → HQ-SAM-2)

On **HQ-DC**, generate SSH keys if not already created:

```
ssh-keygen -t rsa -b 4096
```

Then copy the public key to **HQ-SAM-2**:

```
ssh-copy-id user@HQ-SAM-2
```

Test:

```
ssh user@HQ-SAM-2
```

✓ 2. Create the backup script on HQ-DC

🔧 **Example: /usr/local/bin/user_backup.sh**

```
#!/bin/bash
```

```
# === CONFIGURATION ===  
SOURCE_DIR="/share/users"  
DEST_HOST="user@HQ-SAM-2"  
DEST_DIR="/backup/users"  
TIMESTAMP=$(date +%F-%H-%M-%S)  
ARCHIVE_NAME="users-backup-$TIMESTAMP.tar.gz"  
TMP_ARCHIVE="/tmp/$ARCHIVE_NAME"
```

```
# === CREATE LOCAL BACKUP ===  
tar -czf "$TMP_ARCHIVE" "$SOURCE_DIR"
```

```
# === SEND TO REMOTE SERVER ===
```

```
scp "$TMP_ARCHIVE" "$DEST_HOST:$DEST_DIR"
```

```
# === REMOVE LOCAL TEMP FILE ===  
rm -f "$TMP_ARCHIVE"
```



Make it executable:

```
chmod +x /usr/local/bin/user_backup.sh
```



3. Set up the cron job (on HQ-DC)



Edit crontab:

```
crontab -e
```



Add this line:

```
*/5 * * * * /usr/local/bin/user_backup.sh
```

This runs the backup every 5 minutes.



4. Confirm Permissions

Ensure:

- /share/users is readable by the script user.
- SSH user on HQ-SAM-2 has write permissions to /backup/users.
- Enough disk space on both ends.



5. Optional: Log Output

You can modify the script to include logging:

```
LOGFILE="/var/log/user_backup.log"  
echo "[$(date)] Starting backup..." >> "$LOGFILE"
```



Example Result on HQ-SAM-2

Files in /backup/users will look like:

```
users-backup-2025-09-09-15-00-00.tar.gz  
users-backup-2025-09-09-15-05-00.tar.gz
```

17. Routing (R-HQ, R-INT and R-BR)

Use a dynamic routing protocol on the public network.

Got it 👍 You want to run **RIP (Routing Information Protocol)** over a **WireGuard VPN tunnel**. I'll break down how to configure this step by step.

1. Install Required Packages

On both peers (let's call them **Router A** and **Router B**):

```
sudo apt update
sudo apt install wireguard frr -y
```

- wireguard → VPN interface
- frr (FRRouting) → RIP daemon

2. Configure WireGuard

On **Router A** (/etc/wireguard/wg0.conf):

```
[Interface]
Address = 10.0.0.1/30
PrivateKey = <RouterA_PrivateKey>
ListenPort = 51820

[Peer]
PublicKey = <RouterB_PublicKey>
AllowedIPs = 10.0.0.2/32
Endpoint = <RouterB_PublicIP>:51820
```

On **Router B** (/etc/wireguard/wg0.conf):

```
[Interface]
Address = 10.0.0.2/30
PrivateKey = <RouterB_PrivateKey>
ListenPort = 51820

[Peer]
PublicKey = <RouterA_PublicKey>
AllowedIPs = 10.0.0.1/32
Endpoint = <RouterA_PublicIP>:51820
```

Then enable:

```
sudo systemctl enable wg-quick@wg0
sudo systemctl start wg-quick@wg0
```

Check:

```
ping 10.0.0.2 # from Router A
```

3. Configure RIP in FRRouting

Enable FRR daemons:

```
sudo nano /etc/frr/daemons
```

Set:

```
ripd=yes
```

Restart FRR:

```
sudo systemctl restart frr
```

4. Set RIP on WireGuard Interface

Enter FRR vtysh on each router:

```
sudo vtysh
```

On **Router A**:

```
configure terminal
router rip
network wg0
redistribute connected
exit
write memory
```

On **Router B**:

```
configure terminal
router rip
network wg0
redistribute connected
exit
write memory
```

5. Verify

Check RIP neighbors:

```
show ip rip neighbor
```

Check learned routes:

```
show ip route rip
```

✅ Now RIP is running **inside the WireGuard tunnel**. The WG interface (wg0) behaves like a regular point-to-point interface, and RIP exchanges routes through it.

Do you want me to make you a **lab topology example** (with 2 routers + LANs) so you can actually test dynamic routing via RIP over WireGuard?

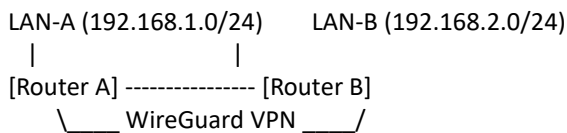
Tú dijiste:

Yes, go on!

ChatGPT Plus

Perfect 😊 Let's build a **small lab topology** where RIP runs inside a WireGuard tunnel.

🖧 Topology



- **Router A**

- LAN: 192.168.1.1/24
- WG: 10.0.0.1/30

- **Router B**

- LAN: 192.168.2.1/24
- WG: 10.0.0.2/30

Goal: PCs on LAN-A and LAN-B should reach each other using RIP to advertise LAN networks via the WG tunnel.

1. Install Packages

On both routers (Debian/Ubuntu style):

```
sudo apt update
sudo apt install wireguard frr -y
```

2. Configure WireGuard

Router A → /etc/wireguard/wg0.conf:

```
[Interface]
Address = 10.0.0.1/30
PrivateKey = <RouterA_PrivateKey>
ListenPort = 51820

[Peer]
PublicKey = <RouterB_PublicKey>
AllowedIPs = 10.0.0.2/32
Endpoint = <RouterB_PublicIP>:51820
```

Router B → /etc/wireguard/wg0.conf:

```
[Interface]
Address = 10.0.0.2/30
PrivateKey = <RouterB_PrivateKey>
ListenPort = 51820

[Peer]
PublicKey = <RouterA_PublicKey>
AllowedIPs = 10.0.0.1/32
Endpoint = <RouterA_PublicIP>:51820
```

Enable tunnel:

```
sudo systemctl enable wg-quick@wg0
sudo systemctl start wg-quick@wg0
```

Check connectivity:

```
ping 10.0.0.2 # from Router A
```

3. Configure FRR and RIP

Enable RIP daemon:

```
sudo nano /etc/frr/daemons
```

Set:

```
ripd=yes
```

Restart FRR:

```
sudo systemctl restart frr
```

4. FRR Configuration

Router A

```
sudo vtysh
```

```
configure terminal
router rip
network wg0
network eth0 ! LAN interface (192.168.1.0/24)
redistribute connected
exit
write memory
```

Router B

```
sudo vtysh
```

```
configure terminal
router rip
```

```
network wg0
network eth0    ! LAN interface (192.168.2.0/24)
redistribute connected
exit
write memory
```

5. Verification

- On Router A:

```
show ip rip neighbor
show ip route rip
```

You should see 192.168.2.0/24 learned via RIP.

- On Router B:

```
show ip rip neighbor
show ip route rip
```

You should see 192.168.1.0/24 learned via RIP.

Now PCs behind each router can **ping each other** through the WireGuard tunnel with dynamic routing provided by RIP 🎉.

18. IPv6 (R-HQ, R-INT, HOME and R-BR)

Add: Configure IPv6 addressing for Internet access. VPN services should be reachable over IPv6.
(Topology is provided.)

Tú dijiste:

configure RIP with wireguard and IPv6 addressing

ChatGPT Plus

Got it  — you want to configure **RIP routing (RIPng for IPv6)** on top of a **WireGuard VPN** tunnel with IPv6 addressing.

Let's go step by step:

1. Prerequisites

- Two (or more) Linux machines (Debian/Ubuntu/RHEL/...).
- Installed packages:

```
sudo apt install wireguard quagga -y
```

or if using FRR instead of Quagga:

```
sudo apt install frr frr-rpki-rtrlib -y
```

2. Configure WireGuard Tunnel (IPv6 only)

On **Router A** (wg0.conf):

```
[Interface]
PrivateKey = <A-private-key>
Address = fd00:1::1/64
ListenPort = 51820
```

```
[Peer]
PublicKey = <B-public-key>
AllowedIPs = fd00:1::2/128
Endpoint = <B-public-IP>:51820
```

On **Router B** (wg0.conf):

```
[Interface]
PrivateKey = <B-private-key>
Address = fd00:1::2/64
ListenPort = 51820
```

```
[Peer]
PublicKey = <A-public-key>
AllowedIPs = fd00:1::1/128
Endpoint = <A-public-IP>:51820
```

Enable and bring up:

```
sudo wg-quick up wg0
```

You should now have IPv6 connectivity between fd00:1::1 and fd00:1::2.

3. Enable RIPng (RIP for IPv6)

Using **FRR** (recommended over Quagga):

1. Enable ripngd:

```
sudo nano /etc/frr/daemons
```

Set:

```
ripngd=yes
```

2. Configure RIPng on **Router A**:

```
vysh
configure terminal
router ripng
  network wg0
exit
write
```

Do the same on **Router B**.

4. Verify Routing

On Router A:

```
show ipv6 route ripng
```

You should see prefixes learned via RIPng over the WireGuard tunnel.

Summary:

- WireGuard provides the secure IPv6 tunnel (wg0).
- RIPng runs inside that tunnel to dynamically exchange IPv6 routes.
- Each router will learn the other's LAN prefixes via ripng.

Do you want me to also include an **example with LAN networks on both routers** (so that RIPng advertises them through the WireGuard tunnel), or just the **point-to-point RIPng over wg0**?

Tú dijiste:

Yes, go on!

ChatGPT Plus

Perfect 👍 let's extend the setup so that each router has a **LAN behind it**, and we'll use **RIPng** over the WireGuard tunnel (wg0) to advertise those LAN prefixes.

Scenario

- **Router A**

- LAN: 2001:db8:a::/64 on eth0
- Tunnel IP: fd00:1::1/64 on wg0

- **Router B**

- LAN: 2001:db8:b::/64 on eth0
- Tunnel IP: fd00:1::2/64 on wg0

Goal: hosts in LAN A should reach LAN B, and vice versa, using RIPng over the WireGuard VPN.

1. WireGuard config (as before)

Router A (/etc/wireguard/wg0.conf)

```
[Interface]
PrivateKey = <A-private-key>
Address = fd00:1::1/64
ListenPort = 51820

[Peer]
PublicKey = <B-public-key>
AllowedIPs = fd00:1::2/128
Endpoint = <B-public-IP>:51820
```

Router B (/etc/wireguard/wg0.conf)

```
[Interface]
PrivateKey = <B-private-key>
Address = fd00:1::2/64
ListenPort = 51820

[Peer]
PublicKey = <A-public-key>
AllowedIPs = fd00:1::1/128
Endpoint = <A-public-IP>:51820
```

Enable:

```
sudo wg-quick up wg0
```

2. Assign IPv6 to LAN interfaces

On **Router A**:

```
sudo ip -6 addr add 2001:db8:a::1/64 dev eth0
```

On **Router B**:

```
sudo ip -6 addr add 2001:db8:b::1/64 dev eth0
```

(Clients in each LAN get addresses via SLAAC/DHCPv6 from these subnets.)

3. Configure RIPng in FRR

Enable ripngd

Edit /etc/frr/daemons:

```
ripngd=yes
```

Restart FRR:

```
sudo systemctl restart frr
```

Router A RIPng config

```
vttysh
configure terminal
router ripng
  network wg0
  network eth0
exit
write
```

Router B RIPng config

```
vttysh
configure terminal
router ripng
  network wg0
  network eth0
exit
write
```

4. Verification

On Router A:

```
show ipv6 route ripng
```

You should see 2001:db8:b::/64 learned via RIPng.

On Router B:

```
show ipv6 route ripng
```

You should see 2001:db8:a::/64 learned via RIPng.

5. Test connectivity

From a host in LAN A (say 2001:db8:a::100):

```
ping6 2001:db8:b::100
```

This should reach a host in LAN B.

✅ Now you have:

- A **WireGuard secure IPv6 tunnel** between routers.
- **RIPng exchanging routes dynamically.**
- Full connectivity between LAN A and LAN B without static routes.