

# How to Create a Private Mesh Network with Vultr and OVH Using Netbird

Learn how to set up a secure private mesh network between Vultr and OVH servers using Netbird. Step-by-step guide for seamless connectivity across providers.

# Contents

01	Introduction	3
02	Prerequisites	3
03	Deploy the NetBird Control Plane on Vultr	3
04	Configure OVH Firewall for NetBird Peers	6
05	Add Peers to the NetBird Network	7
06	Final Verification	8
07	Route Traffic Through an Exit Node	8
08	NetBird Use Cases and Components	12
09	Conclusion	12

# Introduction

---

[NetBird](#) is an open-source networking platform that uses WireGuard to create encrypted peer-to-peer overlays. Instead of relying on manual VPN tunnels or complex firewall rules, it automatically forms a mesh that lets servers in different locations talk to each other over private, low-latency links.

In this guide, you [self-host the NetBird control plane](#) on a Vultr instance and connect it with an OVHcloud server to form a cross-provider network. The steps cover preparing Vultr and OVH environments, adjusting firewall settings, enrolling peers with setup keys, and validating secure connectivity. You also configure a Vultr peer as an exit node so that OVH traffic flows through Vultr for centralized routing and policy control.

## Prerequisites

---

Before building your Vultr–OVH mesh network, make sure you have:

- An [Ubuntu-based Vultr instance](#) to host the NetBird control plane.
  - Example: deployed in the Delhi (DEL) region.
  - Attach a domain name with its [DNS A record](#) pointing to the public IP, such as `netbird.example.com`.
- A second Ubuntu-based Vultr instance to act as a peer.
  - Example: deployed in the Amsterdam (AMS) region.
- An Ubuntu-based server at OVHcloud (VPS, Public Cloud, or Bare Metal) deployed in a region of your choice.

## Deploy the NetBird Control Plane on Vultr

---

The control plane is the backbone of your NetBird deployment. It manages peer registration, distributes configuration, and provides the signaling and TURN/

STUN services needed to keep tunnels online. In this setup, you deploy the control plane on a Vultr instance and install everything using Docker.

1. Allow required ports on the Vultr control plane for HTTPS, signaling, management, and TURN/STUN traffic.

**CONSOLE**

```
$ sudo ufw allow 80/tcp
$ sudo ufw allow 443/tcp
$ sudo ufw allow 33073/tcp
$ sudo ufw allow 10000/tcp
$ sudo ufw allow 33080/tcp
$ sudo ufw allow 3478/udp
$ sudo ufw allow 49152:65535/udp
$ sudo ufw reload
```

2. Add Docker's repository and install Docker Engine, the Compose plugin, and supporting tools.

**CONSOLE**

```
$ sudo apt update
$ sudo apt install ca-certificates curl gnupg lsb-release -y
$ sudo install -m 0755 -d /etc/apt/keyrings
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg |
sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
$ echo "deb [arch=$(dpkg --print-architecture) signed-by=/
etc/apt/keyrings/docker.gpg] https://download.docker.com/
linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/
sources.list.d/docker.list > /dev/null
$ sudo apt update
$ sudo apt install docker-ce docker-ce-cli containerd.io
docker-buildx-plugin docker-compose-plugin jq -y
```

3. Enable and start the Docker service.

**CONSOLE**

```
$ sudo systemctl enable --now docker
```

4. Add your user to the Docker group to avoid prefixing commands with `sudo`.

## CONSOLE

```
$ sudo usermod -aG docker $USER
$ newgrp docker
```

5. Verify that the Docker Compose plugin is available.

## CONSOLE

```
$ docker compose version
```

6. Install NetBird by setting your domain name in the `NETBIRD_DOMAIN` variable.

## CONSOLE

```
$ export NETBIRD_DOMAIN=netbird.example.com
$ curl -fsSL https://github.com/netbirdio/netbird/releases/
latest/download/getting-started-with-zitadel.sh | bash
```

 Note

Replace `netbird.example.com` with the domain that points to your Vultr control plane.

7. When the script completes, open the NetBird dashboard at:

```
https://netbird.example.com
```

Copy the credentials and setup key displayed in your terminal before closing it. These are shown only once.

 Warning

Do not close your terminal before copying the setup key and credentials. These values are only shown once during installation.

# Configure OVH Firewall for NetBird Peers

OVHcloud servers may use two layers of filtering: the **Network Firewall** in the OVHcloud Manager, and the host-level firewall inside the VM. By default, outbound traffic is allowed, but if you have enabled the Network Firewall or host-level restrictions, you must permit outbound access so the peer can reach the Vultr control plane.

1. Log in to the **OVHcloud Manager**.
2. In the left sidebar, go to **Cloud > Servers** (or **Public Cloud > Instances**, depending on your service).
3. Select the server you want to configure.
4. Under **Network Firewall**, check if it is enabled.
5. If enabled, edit the firewall rules to allow the following outbound ports:
  - **TCP:** 80, 443, 33073, 10000, 33080
  - **UDP:** 3478, 49152-65535
6. On the OVH VM itself, allow the same outbound ports with `ufw` (if it is enabled):

## CONSOLE

```
$ sudo ufw allow out 80/tcp
$ sudo ufw allow out 443/tcp
$ sudo ufw allow out 33073/tcp
$ sudo ufw allow out 10000/tcp
$ sudo ufw allow out 33080/tcp
$ sudo ufw allow out 3478/udp
$ sudo ufw allow out 49152:65535/udp
$ sudo ufw reload
```

7. Verify connectivity from the OVH VM to your Vultr control plane domain.

## CONSOLE

```
$ curl -I https://netbird.example.com
```

A response such as `200 OK` confirms the peer can reach the control plane.

## Add Peers to the NetBird Network

The recommended way to connect Vultr and OVH servers to your NetBird mesh is by using **setup keys**. Setup keys are pre-authorized tokens that let peers join automatically without requiring an interactive login. See the [NetBird Setup Keys documentation](#) for more details.

1. In the NetBird Admin Panel, go to **Setup Keys** and click **Create Setup Key**.
  - Give the key a name (for example, `vultr-ovh-peers`).
  - Configure usage limits if needed.
  - Copy the generated key.
2. Install the NetBird client on each peer.

CONSOLE

```
$ curl -fsSL https://pkgs.netbird.io/install.sh | sh
```

3. Register the peer with your control plane.

CONSOLE

```
$ sudo netbird up --management-url https://netbird.example.com --admin-url https://netbird.example.com --setup-key <SETUP_KEY>
```

Replace `<SETUP_KEY>` with the copied key.

4. In the Admin Panel, verify that the peer appears online. Rename it to something descriptive, such as `vultr-ams` or `ovh-vps`, and add it to a group if you plan to configure an exit node.

## Final Verification

After registering both Vultr and OVH peers, confirm that they can securely communicate over the NetBird private mesh.

1. In the NetBird Admin Panel, go to the **Peers** tab.
  - Both peers should appear as **Online**.
  - Each will have a `100.x.x.x` NetBird-assigned mesh IP address.
2. From your Vultr peer (for example, your Vultr AMS VM), test connectivity to the OVH peer using its mesh IP:

### CONSOLE

```
$ ping 100.x.x.x # Replace with the actual mesh IP of the  
OVH peer
```

Your output should be similar to the one below:

```
PING 100.100.2.7 (100.100.2.7) 56(84) bytes of data.  
64 bytes from 100.100.2.7: icmp_seq=1 ttl=64 time=22.5 ms  
64 bytes from 100.100.2.7: icmp_seq=2 ttl=64 time=22.7 ms
```

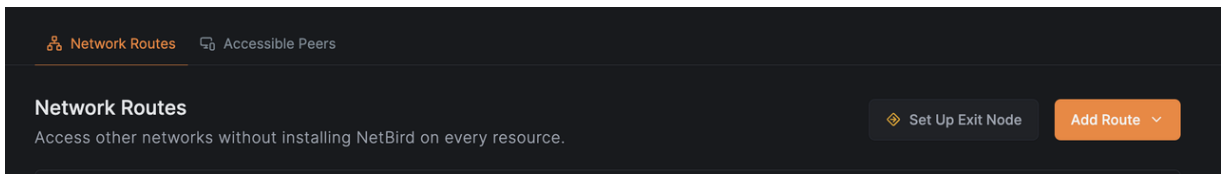
3. Repeat the ping test in the opposite direction (from OVH VPS > Vultr peer) to confirm two-way connectivity.

## Route Traffic Through an Exit Node

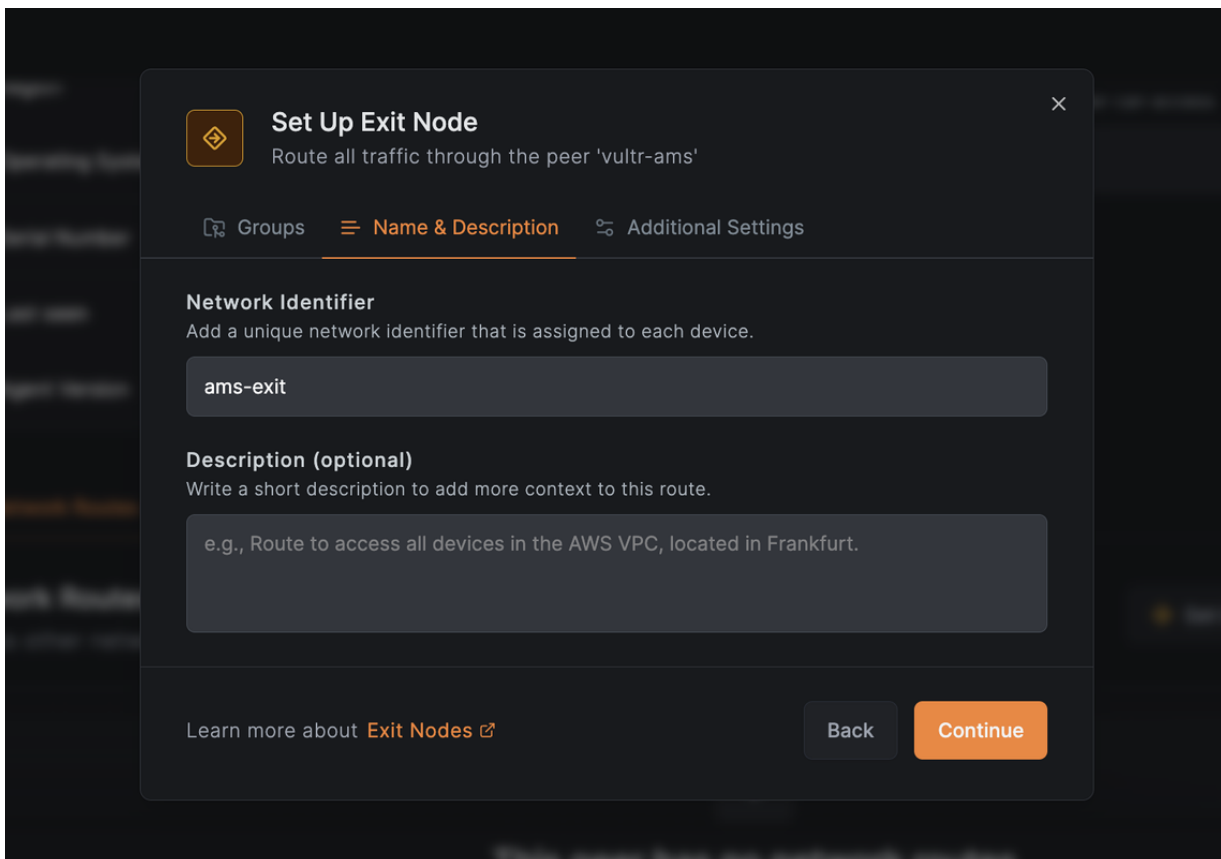
You can configure one peer as an **exit node** so other peers route their internet traffic through it. In this example, the Vultr AMS instance acts as the exit node, and the OVH peer forwards its traffic through it.

## Designate the Vultr AMS Instance as Exit Node

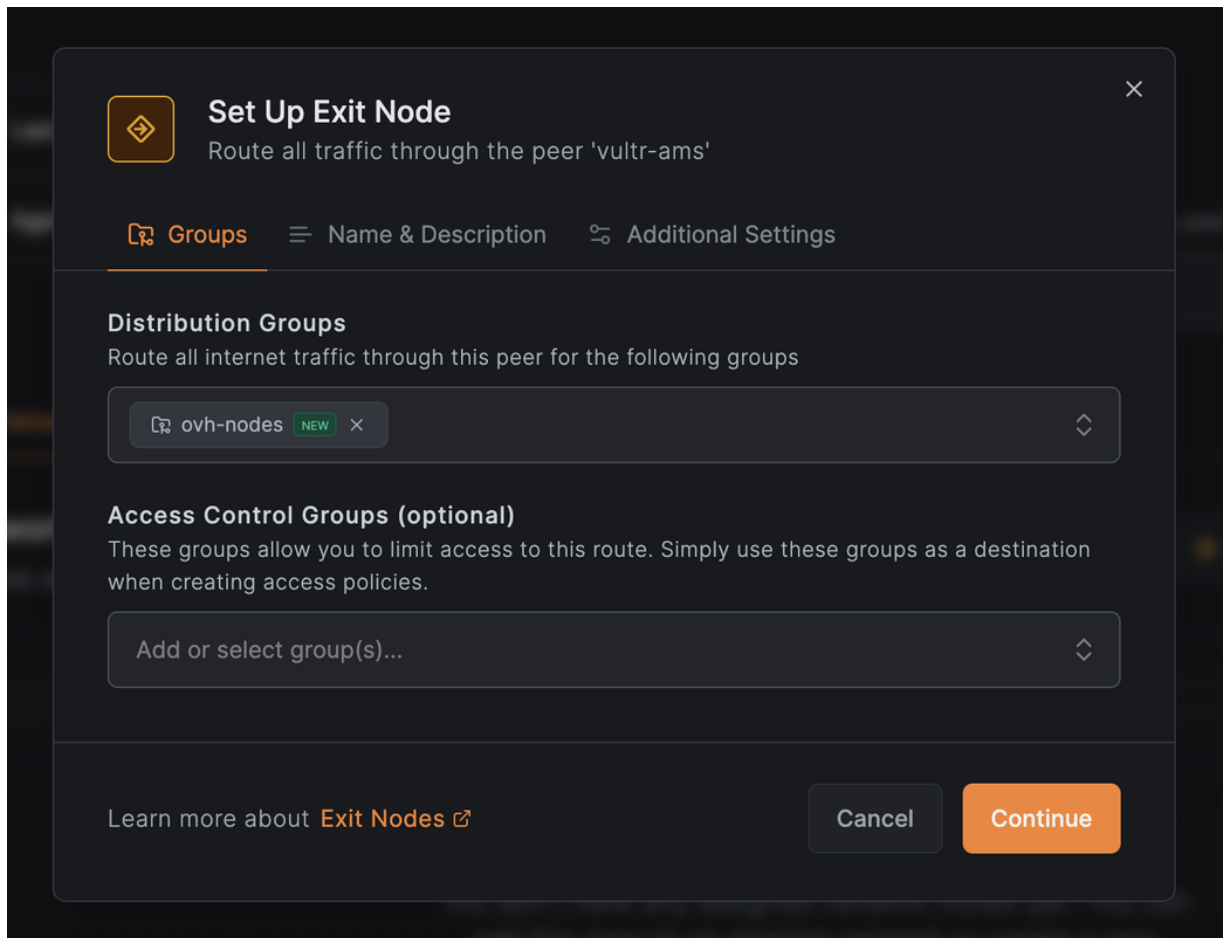
1. In the **Peers** tab of the NetBird Admin Panel, select the peer named `vultr-ams`.
2. Scroll down and click **Set Up Exit Node**.



3. Enter an identifier such as `ams-exit`.



4. In the **Distribution Groups** dropdown, select or create a group that will include the OVH peer (for example, `ovh-nodes`).



##### 5. Click **Save Changes**.

The `vultr-ams` instance is now configured as an exit node.

## Assign the OVH Peer to the Distribution Group

1. In the **Peers** view, select the OVH peer (for example, `ovh-vps`).
2. Assign it to the `ovh-nodes` group.
3. Confirm that it now shows up as part of the peers using `vultr-ams` as an exit node.

## Verify Routing

To confirm that OVH traffic flows through the Vultr exit node, run the following commands on the AMS-region Vultr peer.

1. Enable IP forwarding.

## CONSOLE

```
$ sudo sysctl -w net.ipv4.ip_forward=1
```

2. Add a MASQUERADE rule to NAT outbound traffic via the correct network interface.

## CONSOLE

```
$ sudo iptables -t nat -A POSTROUTING -o $(ip route get 1.1.1.1 | awk '{print $5}') -j MASQUERADE
```

3. Confirm that IP forwarding is enabled.

## CONSOLE

```
$ sudo sysctl net.ipv4.ip_forward
```

4. Monitor the NAT rule counters.

## CONSOLE

```
$ sudo watch -n1 "iptables -t nat -v -L POSTROUTING"
```

The output shows a live counter of packets hitting the MASQUERADE rule. The values should increase as traffic from the OVH peer flows through the Vultr AMS exit node.

```
Chain POSTROUTING (policy ACCEPT 150 packets, 12034 bytes)
pkts bytes target      prot opt in      out     source      destination
58   4801 MASQUERADE  all  --  any     enp1s0  anywhere    anywhere
```

 Note

If packet counts do not increase, verify that the OVH peer is assigned to the `ovh-nodes` group and ensure both firewalls allow outbound connections.

# NetBird Use Cases and Components

---

## Common Multi-Cloud Use Cases

- **Cross-Provider Networking:** Seamlessly connect workloads between Vultr and OVH without exposing them to the public internet.
- **Centralized Egress Control:** Route OVH traffic through a Vultr exit node to standardize outbound IPs for compliance or monitoring.
- **Hybrid Infrastructure:** Combine bare metal servers in OVH with Vultr VMs into one private mesh network for testing or production.

## Key Components

- **Management Service:** Coordinates peer registration, configuration, and network policies.
- **Signal & TURN Server:** Helps peers behind NATs establish connections, and relays encrypted traffic when direct tunneling fails.
- **Peer Agent:** The lightweight client installed on Vultr and OVH instances to maintain encrypted WireGuard tunnels.
- **Setup Keys:** Pre-authorized tokens for adding peers non-interactively.
- **SSO Integration:** Optional authentication integration with identity providers like GitHub, Google, or a custom OIDC provider.

## Conclusion

---

In this guide, you deployed the NetBird control plane on a Vultr instance and extended the mesh network to include an OVH server. You registered peers using setup keys, confirmed secure private connectivity, and configured a Vultr exit node to handle OVH traffic. With this setup, you can interconnect Vultr and OVH workloads into a single encrypted overlay, simplify multi-cloud networking, and centralize traffic control for monitoring, compliance, or egress management.



VULTR

