

# Oracle® Linux

## Configuring Virtual Private Networks

**ORACLE®**

F45242-01  
October 2021

---

## Oracle Legal Notices

Copyright © 2021 Oracle and/or its affiliates.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software" or "commercial computer software documentation" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

### Abstract

[Oracle® Linux: Configuring Virtual Private Networks](#) describes how to configure and use OpenSSH to securely connect to systems on the network.

Document generated on: 2021-10-07 (revision: 12497)

---

---

# Table of Contents

Preface .....	v
1 About Virtual Private Networks .....	1
2 Configuring a VPN by Using WireGuard .....	3
2.1 Installing WireGuard .....	3
2.2 Configuring WireGuard .....	3
2.3 Enabling the WireGuard Tunnel .....	5
2.4 Testing the WireGuard Tunnel .....	6
2.5 Shutting Down the WireGuard Tunnel .....	7
3 Configuring a VPN by Using Libreswan .....	9
3.1 Installing Libreswan .....	9
3.2 Configuring IPsec VPN .....	9
3.2.1 Creating a Host to Host Connection .....	9
3.2.2 Creating a Site to Site Connection .....	11
3.3 Verifying the Status of VPN Services .....	11



---

## Preface

*Oracle® Linux: Configuring Virtual Private Networks* describes how to use virtual private networks (VPNs) in Oracle Linux to deploy tunneled connections to remote systems.

## Audience

This document is intended for administrators who need to configure and administer Oracle Linux features. It is assumed that readers are familiar with web technologies and have a general understanding of using the Linux operating system, including knowledge of how to use a text editor such as *emacs* or *vim*, essential commands such as *cd*, *chmod*, *chown*, *ls*, *mkdir*, *mv*, *ps*, *pwd*, and *rm*, and using the *man* command to view manual pages.

## Related Documents

The documentation for this product is available at:

<https://docs.oracle.com/en/operating-systems/linux.html>.

## Conventions

The following text conventions are used in this document:

Convention	Meaning
<b>boldface</b>	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
<i>italic</i>	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
<code>monospace</code>	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

## Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at

<https://www.oracle.com/corporate/accessibility/>.

## Access to Oracle Support for Accessibility

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit

<https://www.oracle.com/corporate/accessibility/learning-support.html#support-tab>.

## Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings

and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

---

# Chapter 1 About Virtual Private Networks

*Oracle® Linux: Configuring Virtual Private Networks* provides a brief description about VPNs and their benefits. It also introduces the two technologies that businesses currently use to set up VPNs.



## Note

This document includes content that was tested against Oracle Linux 8, but generally applies to most Oracle Linux releases, and may also apply to other distributions.

VPNs are used to enable remote access between endpoints and provide site-to-site connections that simulate a larger network beyond the limitations of a deployed physical network. Encryption is applied to secure the traffic traversing the tunneled connections.

Implementations of VPNs have varied over time through the release of different VPN protocols, technologies and application. Oracle Linux supports two VPN technologies: IPsec OpenVPN implementation in Libreswan and WireGuard®.

WireGuard differs from OpenVPN in that OpenVPN uses certificates for identification and encryption. WireGuard uses public key encryption for those tasks. Secure key generation and management is handled **in the** background, and an option is available to pre-share a key for an additional layer of security.

For more information, refer to the following resources:

- WireGuard: <https://www.wireguard.com>
- Libreswan: <https://libreswan.org>





---

# Chapter 2 Configuring a VPN by Using WireGuard

## Table of Contents

2.1 Installing WireGuard .....	3
2.2 Configuring WireGuard .....	3
2.3 Enabling the WireGuard Tunnel .....	5
2.4 Testing the WireGuard Tunnel .....	6
2.5 Shutting Down the WireGuard Tunnel .....	7

WireGuard is a cross-platform technology that enables you to create a VPN setup that is simple, fast, but secure through its implementation of the latest cryptography.



### Note

WireGuard is supported beginning with Unbreakable Enterprise Kernel Release 6 Update 3. Ensure that your system is upgraded to this release before proceeding to the steps in this chapter. For more information, see the [Unbreakable Enterprise Kernel: Release Notes for Unbreakable Enterprise Kernel Release 6 Update 3](#).

## 2.1 Installing WireGuard

To configure a VPN with WireGuard, download the package with the following command:

```
sudo dnf install -y wireguard-tools
```

You must install WireGuard on the server and all of its clients.

## 2.2 Configuring WireGuard

For simplicity, the following sections describe how to deploy WireGuard by using two hosts as examples. One host functions as the VPN server while the other is a client.

To use WireGuard, you need the following requirements:

- IP addresses of both hosts. Use the `ip addr sh` command to obtain this information. For the procedures that follow, the IP addresses of the server and client are 10.0.0.1 and 10.0.0.2, respectively
- Private IP addresses to be assigned to the WireGuard interfaces of both hosts. For the procedures that follow, the private IP addresses of the server and client are 192.168.2.1 and 192.168.2.2, respectively.
- Name of the WireGuard interface. For the procedures that follow, the Wireguard interface name of both server and client is `wg0`.
- Console connections to the client.

Instructions for setting up console connections is outside the scope of this documentation. To use a VNC server for remote connections, see <https://docs.oracle.com/en/learn/install-vnc-oracle-linux/#introduction>. If you are using an instance in Oracle Cloud Infrastructure, see [https://docs.oracle.com/iaas/Content/Compute/References/serialconsole.htm#Instance\\_Console\\_Connections](https://docs.oracle.com/iaas/Content/Compute/References/serialconsole.htm#Instance_Console_Connections).

### On the server

1. Edit the `/etc/sysctl.conf` file with the following changes:

```
net.ipv4.ip_forward = 1
```

2. Run the `sysctl` command to reread the `/etc/sysctl.conf` file.

```
sudo sysctl -p
```

You can disregard the error messages that might appear about the command not being able to perform `stat`.

3. Create a directory where you can store the WireGuard key pair, for example:

```
mkdir ~/.wireguard  
cd ~/.wireguard  
umask 077
```

4. In the directory you just created, generate a WireGuard cryptographic key pair.

You can specify any preferred names for the files to contain the server's private and public keys.

```
wg genkey | tee privatekey | wg pubkey > publickey
```

5. Obtain the server's private and public keys and store them in a temporary storage.

```
cat ~/.wireguard/privatekey  
cat ~/.wireguard/publickey
```

### On the client

1. Create a directory where you can store the WireGuard key pair, for example:

```
mkdir ~/.wireguard  
cd ~/.wireguard  
umask 077
```

2. In the directory you just created, generate a WireGuard cryptographic key pair.

You can specify any preferred names for the files to contain the client's private and public keys.

```
wg genkey | tee privatekey | wg pubkey > publickey
```

3. Obtain the client's private and public keys and store them in a temporary storage.

```
cat ~/.wireguard/privatekey  
cat ~/.wireguard/publickey
```

4. Edit the `/etc/wireguard/wg0.conf` file to contain the following:

```
[Interface]  
Address = 192.168.2.2/24  
SaveConfig = true  
ListenPort = 60477  
PrivateKey = client's private key  
  
[Peer]  
PublicKey = server's public key  
AllowedIPs = 0.0.0.0/0, ::/0  
Endpoint = 10.0.0.1:51820
```

### On the server

1. Edit the `/etc/wireguard/wg0.conf` file to contain the following:

```
[Interface]
Address = 192.168.2.1/24
SaveConfig = true
PostUp = iptables -A FORWARD -i wg0 -j ACCEPT; iptables -t nat -A POSTROUTING -o eno1 -j MASQUERADE
PostDown = iptables -D FORWARD -i wg0 -j ACCEPT; iptables -t nat -D POSTROUTING -o eno1 -j MASQUERADE
ListenPort = 51820
PrivateKey = server's private key
[Peer]
PublicKey = client's public key
AllowedIPs = 192.168.2.2/32
Endpoint = 10.0.0.2:60477
```



### Note

In the `PostUp` and `PostDown` lines of the example, the server's interface name is `eno1`. Ensure that you specify the correct interface name of your server for those same lines. To display the server's IP interface, run the `ip addr sh` command.

## 2.3 Enabling the WireGuard Tunnel

1. Start the tunnel on the server.

```
wg-quick up wg0
[#] ip link add wg0 type wireguard
[#] wg setconf wg0 /dev/fd/63
[#] ip link set mtu 1420 up dev wg0
[#] ip -4 route add 192.168.2.2/32 dev wg0
[#] iptables -A FORWARD -i wg0 -j ACCEPT; iptables -t nat -A POSTROUTING
-o eno1 -j MASQUERADE; ip6tables -A FORWARD -i wg0 -j ACCEPT; ip6tables
-t nat -A POSTROUTING -o eno1 -j MASQUERADE
```

2. Check the status of the tunnel in the server.

```
sudo wg
interface: wg0
  public key: server's public key
  private key: (hidden)
  listening port: 51820

peer: client public's key
  endpoint: 10.0.0.2:60477
  allowed ips: 192.168.2.2/32
```

3. On a separate terminal window, log in to the client by using a console connection.

From this point on, all network traffic will be routed through the tunnel. If you are currently logged in to the client through SSH and then issue the command to start the tunnel, that connection becomes lost. You can only reconnect to the client through a console connection.

4. Start the tunnel on the client.

```
wg-quick up wg0
[#] ip link add wg0 type wireguard
[#] wg setconf wg0 /dev/fd/63
[#] ip -4 address add 192.168.2.2/24 dev wg0
[#] ip link set mtu 1420 up dev wg0
[#] ip -6 route add ::/0 dev wg0 table 51820
[#] ip -6 rule add not fwmark 51820 table 51820
[#] ip -6 rule add table main suppress_prefixlength 0
[#] ip6tables-restore -n
[#] ip -4 route add 0.0.0.0/0 dev wg0 table 51820
```

```
[#] ip -4 rule add not fwmark 51820 table 51820
[#] ip -4 rule add table main suppress_prefixlength 0
[#] sysctl -q net.ipv4.conf.all.src_valid_mark=1
[#] iptables-restore -n
```

### 5. Check the status of the client.

```
sudo wg
interface: wg0
  public key: client's public key
  private key: (hidden)
  listening port: 60477
  fwmark: 0xca6c

peer: server public's key
  endpoint: 10.0.0.1:51820
  allowed ips: 0.0.0.0/0, ::/0
```

You can use other commands to check the tunnel information, such as `ip addr sh` and `ip link`, for example:

```
ip addr sh
...
3: wg0: <POINTOPOINT,NOARP,UP,LOWER_UP> mtu 8920 qdisc noqueue state UNKNOWN group default qlen 1000
    link/none
    inet 192.168.2.1/24 scope global wg0
        valid_lft forever preferred_lft forever

ip link
3: wg0: <POINTOPOINT,NOARP,UP,LOWER_UP> mtu 8920 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1
    link/none
```

## 2.4 Testing the WireGuard Tunnel

On the client, use the `ping` command to check communications with the WireGuard server, for example:

```
ping -c 3 192.168.2.1
PING 192.168.2.1 (192.168.2.1) 56(84) bytes of data.
64 bytes from 192.168.2.1: icmp_seq=1 ttl=64 time=0.640 ms
64 bytes from 192.168.2.1: icmp_seq=2 ttl=64 time=0.707 ms
64 bytes from 192.168.2.1: icmp_seq=4 ttl=64 time=0.643 ms

--- 192.168.2.1 ping statistics ---
3 packets transmitted, 4 received, 0% packet loss, time 2088ms
rtt min/avg/max/mdev = 0.643/0.690/0.713/0.038 ms
```

Likewise, the `wg` command indicates whether a connection to the WireGuard tunnel is established, as shown in the following example:

```
sudo wg
interface: wg0
  public key: client's public key
  private key: (hidden)
  listening port: 60477
  fwmark: 0xca6c

peer: server's public key
  endpoint: 10.0.0.1:51820
  allowed ips: 0.0.0.0/0, ::/0
  latest handshake: 44 seconds ago
  transfer: 28.22 KiB received, 48.64 KiB sent
```

The `handshake` and `transfer` output show the traffic in the tunneled connection.

If communications between the server and the client cannot be established, check if the firewall might be blocking access to the ports that have been set up for WireGuard. You might need to open the ports to link both server and client. For example, you would type the following commands on the server:

```
sudo firewall-cmd --add-port=51820/udp --permanent
sudo firewall-cmd --reload
```

## 2.5 Shutting Down the WireGuard Tunnel

To shut down the WireGuard tunnel, type the following command on both server and client.

```
wg-quick down wireguard-interface
```

After shutting down, you can optionally check the status of the tunnels by using one of the commands that were provided previously: `wg`, `ip addr`, or `ip link`.



---

# Chapter 3 Configuring a VPN by Using Libreswan

## Table of Contents

3.1 Installing Libreswan .....	9
3.2 Configuring IPsec VPN .....	9
3.2.1 Creating a Host to Host Connection .....	9
3.2.2 Creating a Site to Site Connection .....	11
3.3 Verifying the Status of VPN Services .....	11

Libreswan is the software that implements VPN by using the IPsec protocol as well as the Internet Key Exchange (IKE) standards.

## 3.1 Installing Libreswan

To configure an IPsec VPN with Libreswan, download the package as follows:

1. Ensure that the AppStream repository is enabled.
2. Install Libreswan.

```
sudo dnf install -y libreswan
```

3. Start `ipsec` as a persistent service.

```
sudo systemctl enable ipsec --now
```

4. Add the `ipsec` service to the firewall service.

```
sudo firewall-cmd --add-service="ipsec"  
sudo firewall-cmd --runtime-to-permanent
```

## 3.2 Configuring IPsec VPN

VPN configurations range from simple setups such as one between hosts to complex ones that involve entire sites.

### 3.2.1 Creating a Host to Host Connection

Regardless of the types of VPN connections that you want to configure, a common but important step involves obtaining RSA keys that would enable connections between endpoints.

On a host-to-host connection, for example, do the following:

1. Generate an RSA key pair by running the following command.

```
sudo ipsec newhostkey --output /etc/ipsec.d/hostkey.secrets  
Generated RSA key pair with CKOID 6e6e724aa180b071128632dc09c7d2b25a852d7e was stored in the NSS database
```

The command generates an RSA key pair with a specific `ckoid` value.

You must run the command on **both** hosts.

2. On the first host, display the `leftrsasigkey` key.



**Note**

In `libreswan`, the *left* and *right* designations typically refer to the local host and the remote host, respectively. However, because both hosts are peers, the designations can be used interchangeably.

```
sudo ipsec showhostkey --list
```

From the output, identify the `ckaid`, for example, `6e6e724aa180b071128632dc09c7d2b25a852d7e` and use it to display the `leftrsasigkey` key, as follows:

```
sudo ipsec showhostkey --left --ckaid 6e6e724aa180b071128632dc09c7d2b25a852d7e
# rsakey AwEAAaxdf
leftrsasigkey=0sAwEAAaxdfaCPrZ72pAm1kjvhAQHHLn3Wg3gAulZ0U+3FWeh7FN+bHtfy
...
9f8=
```

3. On the second host, display the `rightrsasigkey` key.

```
sudo ipsec showhostkey --list
```

From the output, identify the `ckaid`, for example, `5dddc2334515702c3a605bc00daed1e44e18767d` and use it to display the `rightrsasigkey` key, as follows:

```
ipsec showhostkey --right --ckaid 5dddc2334515702c3a605bc00daed1e44e18767d
# rsakey AwEAAblnC
rightrsasigkey=0sAwEAAAdSSYrNO2QOY8RXgLlJZilBokPb9cFzCbU+VYY7eFcoZMmVWPVI
...
zu+/7BE5kjXHAAlfvYha+CFbuh6KYAlpoHvX81ALusfQs+6wwTsde5j1fcrXNlqX
```

4. On each host, create and edit a configuration file in `/etc/ipsec.d`, for example, `host2host.conf`, with the following entries:

```
conn tunnel-name
leftid=@host1-tunnel-id
left=host1-IPaddress
leftrsasigkey=host1-leftrsasigkey
rightid=@host2-tunnel-id
right=host2-IPaddress
rightrsasigkey=host2-rightrsasigkey
authby=rsasig
```

For more information about the configuration file and other parameters you can set, see the `ipsec.conf(5)` man page.

5. Restart the IPsec service.

```
sudo systemctl restart ipsec
```

6. Start `libreswan`.

```
sudo ipsec setup start
```

7. Load the VPN tunnel connection.

```
sudo ipsec auto --add tunnel-name
```



- Establish the tunnel connection.

```
sudo ipsec auto --up tunnel-name
```

- Start the tunnel automatically when the `ipsec` service is started by adding the following line to the configuration file:

```
auto=start
```

## 3.2.2 Creating a Site to Site Connection

A VPN connection between sites means that a connection is established between two networks. When you configure a pair of hosts for this type of connection, the hosts effectively become gateways through which traffic can enter or exit to access other hosts in the network.

To configure a site to site VPN, a configured host to host VPN must already be existing and operational as described in [Section 3.2.1, “Creating a Host to Host Connection”](#). To proceed with configuring a connection between sites, follow these steps:

- Create a copy of the host to host configuration file to serve as the configuration file for the site to site connection, for example:

```
sudo cp /etc/ipsec.d/host2host.conf /etc/ipsec.d/site2site.conf
```

Copies must exist on both hosts.

- Edit the new configuration file by adding subnet information, for example:

```
conn subnet-name
  also=tunnel-name
  leftsubnet=subnet1-IP
  rightsubnet=subnet2-IP
  auto=start

host connection information...
```



### Note

The subnets can be in CIDR notation.

## 3.3 Verifying the Status of VPN Services

To check if the `ipsec` service is running, type this command:

```
sudo systemctl status ipsec
```

For example:

```
sudo systemctl status ipsec
ipsec.service - Internet Key Exchange (IKE) Protocol Daemon for IPsec
Loaded: loaded (/usr/lib/systemd/system/ipsec.service; enabled; vendor prese>
Active: active (running) since Mon 2021-04-26 02:27:39 PDT; 7h ago
   Docs: man:ipsec(8)
         man:pluto(8)
         man:ipsec.conf(5)
...

```

To check the correctness of the `ipsec` configuration, type this command:

```
sudo ipsec verify
```

For example:

```

sudo ipsec verify
Verifying installed system and configuration files

Version check and ipsec on-path [OK]
Libreswan 3.32 (netkey) on 5.4.17-2036.104.5.el8uek.x86_64
Checking for IPsec support in kernel [OK]
NETKEY: Testing XFRM related proc values
    ICMP default/send_redirects [NOT DISABLED]

    Disable /proc/sys/net/ipv4/conf/*/send_redirects or XFRM/NETKEY will act on or
    cause sending of bogus ICMP redirects!

    ICMP default/accept_redirects [NOT DISABLED]

    Disable /proc/sys/net/ipv4/conf/*/accept_redirects or XFRM/NETKEY will act on
    or cause sending of bogus ICMP redirects!

    XFRM larval drop [OK]
Pluto ipsec.conf syntax [OK]
Checking rp_filter [ENABLED]
    /proc/sys/net/ipv4/conf/all/rp_filter [ENABLED]
    rp_filter is not fully aware of IPsec and should be disabled
Checking that pluto is running [OK]
    Pluto listening for IKE on udp 500 [OK]
    Pluto listening for IKE/NAT-T on udp 4500 [OK]
    Pluto ipsec.secret syntax [OK]
Checking 'ip' command [OK]
Checking 'iptables' command [OK]
Checking 'prelink' command does not interfere with FIPS [OK]
Checking for obsolete ipsec.conf options [OK]

```

To test the tunnel connections, install the `tcpdump` utility to monitor network traffic.

Run the following command on one of the peers to monitor traffic explicitly on the *interface*. The utility tracks Encapsulated Security Payload (ESP) packets as well as traffic traversing the UDP ports 500 and 4500 that are used by the ipsec service:

```

tcpdump -n -i interface esp or udp port 500 or udp port 4500
dropped privs to tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on interface, link-type EN10MB (Ethernet), capture size 262144 bytes
10:05:53.578884 IP 10.147.25.195 > 10.147.25.196: ESP(spi=0xcba1dd78,seq=0x2325), length 96
10:05:53.579353 IP 10.147.25.196 > 10.147.25.195: ESP(spi=0x979dcdbe,seq=0x2325), length 124
10:05:56.585128 IP 10.147.25.195 > 10.147.25.196: ESP(spi=0xcba1dd78,seq=0x2326), length 96
10:05:56.585527 IP 10.147.25.196 > 10.147.25.195: ESP(spi=0x979dcdbe,seq=0x2326), length 124
...

```

The utility first reports traffic that is due to the peers exchanging keys regularly.

While the `tcpdump` is running, go to the other peer and perform a network operation, such as a network `ping`, to the first host. The host that is monitoring the traffic should report network activity over the VPN from the second peer.

Press `Ctrl+c` to end the operations on both peers.