Oracle Linux 10

Setting Up Networking With NetworkManager

G14606-02
October 2025

Oracle Linux 10 Setting Up Networking With NetworkManager,

G14606-02

Contents

6 Network Address Translation

Preface

Oracle Linux 10: Setting Up Networking With NetworkManager provides information about configuring networking for Oracle Linux 10 systems.

1

Network Configuration Overview

To enable the system to connect to the network, transmit and receive traffic with other systems, you would need to configure the system to have identifiable names, IP addresses, routes, and so on. Depending on the system's available resources, you can further optimize the network configuration to attain high availability and improved performance by implementing added network technologies such as network bonds and multipathing.

In Oracle Linux 10, network configuration is managed by `NetworkManager`.

Network Configuration Tools

Different tools are available to configure the network. All of them typically perform the same functions. You can select any tool or a combination of tools to manage the network.

Web-Based Tools

Cockpit is a web-based configuration tool for managing network configuration, including network interfaces, bonds, bridges, virtual VLANs, and the firewall. See Oracle Linux: Using the Cockpit Web Console for detailed instructions about managing the network with Cockpit.

Graphical Tools

If you selected the default System With GUI installation profile or environment to install Oracle Linux, these tools are automatically included. For more information on installation profiles, see the Oracle Linux release's installation guide.

Tool	Details
GNOME Settings	The GNOME Settings application enables you to perform various system configurations, including networking. Access GNOME Settings in either of the following ways: • Click the network icon at the upper right of the desktop and select **Settings**. • On the desktop's menu bar, click **Activities**, select **Show Applications**, then select **Settings**. From the list on the left panel, select the type of configuration you want to do.
Network Connection Editor	The Network Connection Editor is a subset of the GNOME settings application which you can use to directly perform network configurations. To start the editor, run the `nm-connection-editor` command in a terminal window.

Command Line Tools

Use these `NetworkManager` command line tools if you didn't select the Server With GUI installation profile to install Oracle Linux.

Tool	Details
`nmcli`	`nmcli` is `NetworkManager`'s command line tool for managing network settings. Combine subcommands, options, and arguments, to complete network configurations in a single command syntax. To avoid entering long commands, you can also use `nmcli` in interactive mode. Other commands, such as `ip` and `ethtool`, complement `nmcli` for configuring and managing network settings.
`nmtui`	`nmtui` is `NetworkManager`'s text based user interface (TUI). Navigate through the interface by using keyboard keys instead of the mouse device. To start the TUI, run the `nmtui` command in a terminal window.

For more information, see the `nmcli(1)`, `ip(8)`, and `ethtool(8)` manual pages.

Network Interface Names

Traditionally, early kernel versions assigned names to network interface devices by assigning a prefix, which is typically based on the device driver, and a number, such as `eth0`. With the availability of different types of devices, this naming schema is no longer efficient. The names don't necessarily correspond to the chassis labels and the names themselves might be inconsistent across existing network interfaces. The inconsistency would affect embedded adapters on the system, including add-in adapters. Server platforms with several network adapters could have problems managing these interfaces.

Oracle Linux implements a consistent naming scheme for all network interfaces through the `udev` device manager. The scheme offers the following advantages:

- The names of the devices are predictable.
- Device names persist across system reboots or after changes are made to the hardware.
- Defective hardware can easily be identified and thus replaced.

The feature that implements consistent naming on devices is enabled in Oracle Linux by default. Network interface names are based on information that's derived from the system BIOS. Alternatively, they can be based on a device's firmware, system path, or MAC address.

Network interfaces are identified by a name that combines a prefix and a suffix. The prefix depends on the type of network interface:

Prefix	Description
`en`	Ethernet network interfaces.
`wl`	Wireless local area network (LAN) interfaces.
`ww`	Wireless wide area network (WAN) interfaces.

The suffix contains any of the following information:

Prefix	Description
o*n*	An on-board index number. Example: `eno0`
s*n*	A hot-plug slot index number. Example: `ens1` Other prefixes that might be included in the interface name include: • `f`—function • `d`—device-id
p*bus*s*n*	The bus and slot number. Example: `enp0s8` Other prefixes that might be included in the interface name include: • `f`—function • `d`—device-id
x*MAC-addr*	The MAC address. Example: `enx0217b08b` **Note:** This naming format isn't used by Oracle Linux by default. However, administrators can implement it as an option.

2

NetworkManager Connection Profiles

Each network connection configuration that you create becomes a `NetworkManager` connection profile on the system. In Oracle Linux 10, profiles can only be in the key file format. Because network scripts have been removed in Oracle Linux 10, the `ifcfg` format capability that manages these scripts has also been removed.

Depending on its purpose, a `NetworkManager` connection profile can be stored in one of the following locations:

- `/etc/NetworkManager/system-connections/`: Default location of persistent profiles that are created by the user. Profiles in this directory can also be edited.
- `/run/NetworkManager/system-connections/`: Location of temporary profiles that are automatically removed when you reboot the system.
- `/usr/lib/NetworkManager/system-connections/`: Location of predeployed and permanent connection profiles. If you edit one of these profiles by using the `NetworkManager` API, then the profile is copied either to the persistent or the temporary directory.

For more information about configuring `NetworkManager` connection profiles, see:

- Creating a keyfile Connection Profile Using the Network Connection Editor GUI
- Creating a keyfile Connection Profile Using `nmcli`
- Creating a `keyfile` Connection Profile Using `nmcli` in Offline Mode
- Creating a `keyfile` Connection Profile Manually
- Creating a keyfile Connection Profile Using the Text Based User Interface

Creating a keyfile Connection Profile Using the Network Connection Editor GUI

If not already installed, install the `nm-connection-editor` package.

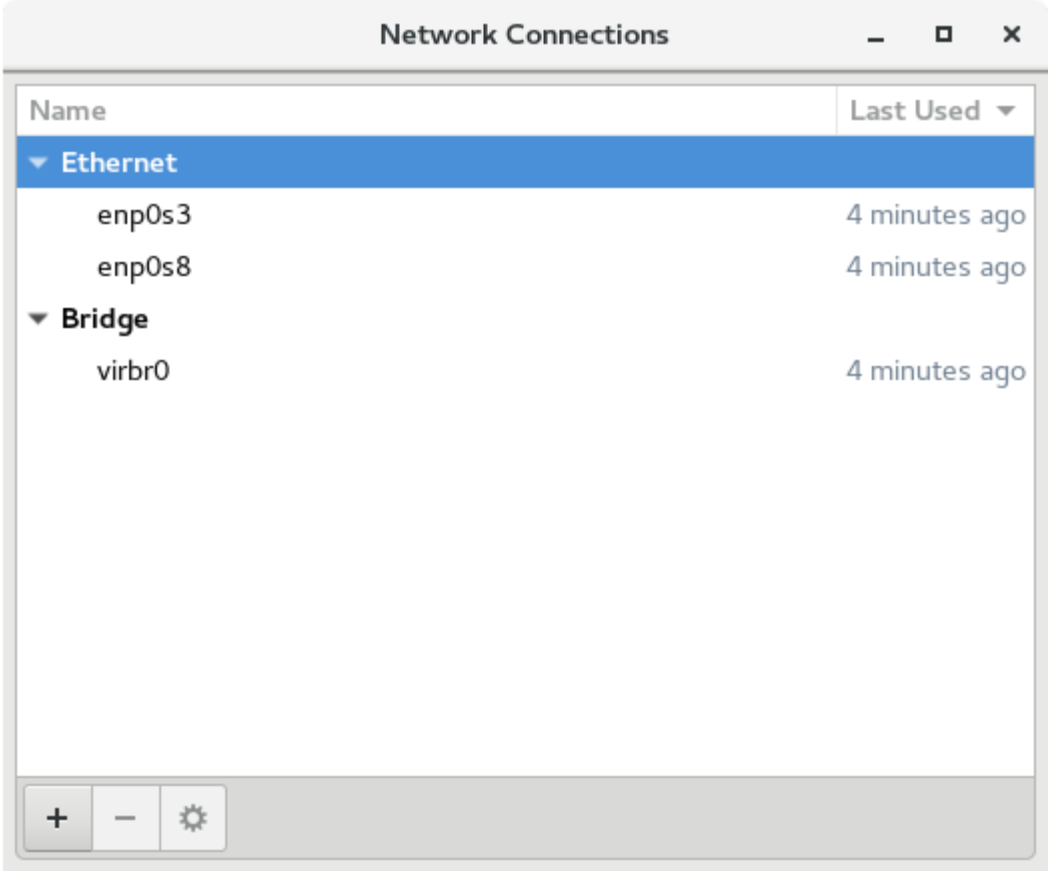
```
sudo dnf install -y nm-connection-editor
```

1. Start the editor:

   ```
   sudo nm-connection-editor
   ```

   The editor detects the network devices that are on the system and lists them and their current states:

Figure 2-1 Network Connections

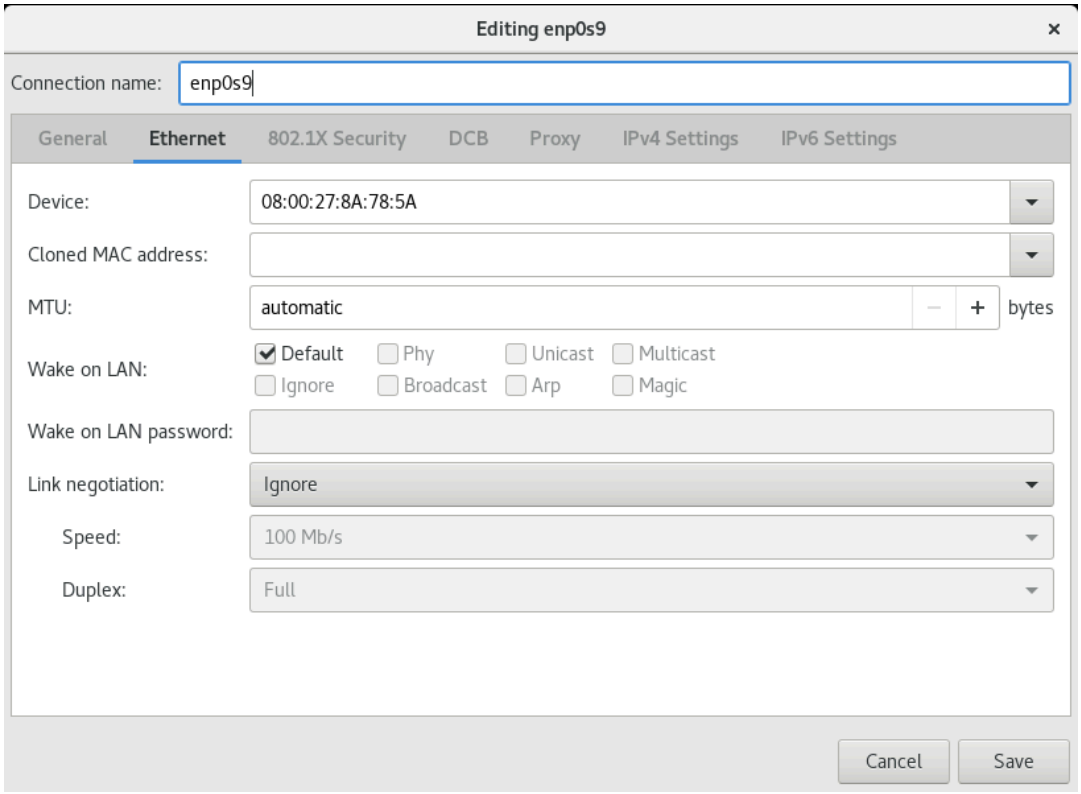


2. To add or remove a connection, use the plus (+) or minus (-) buttons at the bottom of the editor window.

   If you add a connection, a window that prompts you for the connection type opens. Select a type, such as Ethernet, from the drop down list, then click **Create**. The Interface Editor window opens.

> **Note**
>
> The same window opens if you edit an existing connection.

Figure 2-2 Interface Editor



3. Click each tab as needed and enter the required information about the interface.
4. Click **Save** after you have completed the configuration.

   You must specify all the required information. Otherwise, the settings can't be saved and the editor's background terminal window would display messages that indicate the errors.

Creating a keyfile Connection Profile Using `nmcli`

To illustrate the different uses of the `nmcli` command, this procedure describes an example of adding and configuring a new Ethernet connection for the `enp0s2` device. For more information about the command, see the `nmcli(1)` manual page.

> **Tip**
>
> Before adding the connection, prepare the information you would need for the configuration, such as the following:
>
> - Connection name, for example, `My Work Connection`. The `nmcli` command works by referring to the connection name rather than the device name. If you don't set a connection name, then the device's name is used as the connection name.
> - IP addresses (IPv4 and, if needed, IPv6)
> - Gateway addresses
> - Other relevant data you want to set for the connection

1. Display the network devices on the system.

```
sudo nmcli device status
```

```
DEVICE  TYPE      STATE          CONNECTION
enp0s1  ethernet  connected      enp0s1
enp0s2  ethernet  disconnected    --
lo      loopback  unmanaged
```

The command shows whether a device is connected or disconnected, and whether it is managed or unmanaged.

2. Display the connection information about the network devices.

```
sudo nmcli con show --active
```

```
NAME     UUID                                  TYPE      DEVICE
enp0s1   nn-nn-nn-nn-nn  ethernet  enp0s1
virbr0   nn-nn-nn-nn-nn  bridge    virbr0
mybond   nn-nn-nn-nn-nn  bond      bond0
```

The `con` subcommand is the short form of `connection`, and can be further shortened to `c`. Specifying the `--active` option displays only active devices.

> **Note**
>
> In the output, `NAME` represents the connection ID.

3. Add a new connection.

```
sudo nmcli con add type connection type {properties} [IP-info] [gateway-info
```

type
(Required) Specifies a known NetworkManager connection type.

For a list of allowed type values, see the `nmcli connection add` section in the `nmcli(1)` manual page.

properties
The connection name as specified by the `con-name` argument, and the interface name as specified by the `ifname` argument.

IP-info
The IPv4 or IPv6 address as specified by either the `ip4` or `ip6` argument. The address must be in the format `address/netmask`. The IPv4 address can be in CIDR form, for example, `1.2.3.4/24`.

gateway-info
The gateway IPv4 or IPv6 address as specified by either the `gw4` or `gw6` argument.

For example, to add the connection with the information at the beginning of this procedure, run the following command:

```
sudo nmcli con add type ethernet ifname enp0s2 con-name "My Work
Connection" ip4 192.168.5.10/24 gw4 192.168.5.2
```

The output would acknowledge that the connection is successfully completed.

4. Activate the interface.

```
sudo nmcli con up "My Work Connection"
```

5. Display the configuration properties of the new connection.

```
sudo nmcli -o con show "My Work Connection"
```

```
connection.id:                My Work Connection
connection.uuid:              nn-nn-nn-nn-nn
connection.type:              802-3-ethernet
connection.interface-name:    enp0s2
...
IP4.ADDRESS[1]:               192.168.5.10
IP4.GATEWAY:                  192.168.5.2
...
```

Specifying the `-o` option displays only properties that have configured values.

After you have created the connection, a corresponding profile is created. For more information on connection profiles, see NetworkManager Connection Profiles.

```
ls -lrt /etc/NetworkManager/system-connections/
```

```
...
-rw-r--r--. 1 root root 266 Aug  6 11:03 /etc/sysconfig/network-scripts/ifcfg-
My_Work_Connection
```

Creating a `keyfile` Connection Profile Using `nmcli` in Offline Mode

When creating or updating `NetworkManager` profile connections, we recommend using its CLI tool in offline mode (`nmcli --offline`). In offline mode, `nmcli` operates without the `NetworkManager` service, which offers user enhanced editing control and the ability to create various connection profiles in `keyfile` format. For example, you can create the following type of connection profiles in `keyfile` format:

- static Ethernet connection
- dynamic Ethernet connection
- network bond
- network bridge
- VLAN or any kind of enabled connections

Complete the following steps to create a `keyfile` connection profile using `nmcli` in offline mode:

1. Run the `nmcli --offline connection add` command and include property/value pairs for the settings you want to include in the connection profile.

   The `type` property is required. For a list of allowed type values, see the `nmcli connection add` section in the `nmcli(1)` manual page.

   For an exhaustive list of available properties and values, see the `nm-settings-nmcli(5)` manual page.

   The following example shows the syntax to use to create a keyfile for an Ethernet device with a manually assigned IPv4 address and DNS address.

   ```
   nmcli --offline connection add type ethernet con-name Example-Connection
   ipv4.addresses ###.#.#.#/# ipv4.dns ###.#.#.### ipv4.method manual > /etc/
   NetworkManager/system-connections/outputmconnection
   ```

   where:

   - `nmcli --offline` = instructs `nmcli` to operate in offline mode.
   - `connection add` = creates a connection profile.
   - `type ethernet` = specifies a connection type value (in this example: Ethernet).
   - `con-name` = connection name property, which saves the value to the `id` variable in the generated connection profile.
     When you manage this connection later, using `nmcli`, note the following `id` variable usages:
     - In cases where the `id` variable is provided, use the connection name. For example: `Example-Connection`.
     - In cases where the `id` variable is omitted, use the file name without the `.nmconnection` suffix, for example `output`.
   - `ipv4` properties = specify the IP address and name server to use on an IPv4 network without DHCP.
   - `> /etc/NetworkManager/system-connections/outputmconnection` = redirects output from `nmcli` to a new file in `/etc/NetworkManager/system-connections`, where NetworkManager expects connection profiles.

   > **Note**
   >
   > See the `nmcli-examples(7)` manual page for more keyfile examples.

2. Set permissions to the configuration file so that only the `root` user can read and update it.

   ```
   chmod 600 /etc/NetworkManager/system-connections/outputmconnection
   ```

   ```
   chown root:root /etc/NetworkManager/system-connections/outputmconnection
   ```

3. Start the `NetworkManager` service.

   ```
   systemctl start NetworkManager.service
   ```

4. If you set the `autoconnect` variable in the profile to `false`, activate the connection.

   ```
   nmcli connection up Example-Connection
   ```

5. Complete the following steps to verify the profile configuration:

   a. Verify that the `NetworkManager` service is running.

   ```
   systemctl status NetworkManager
   ```

   ```
   ● NetworkManager.service - Network Manager
      Loaded: loaded (/usr/lib/systemd/system/NetworkManager.service
   enabled vendor preset: enabled)
      Active: active (running) because Wed -03 13:08:32 CEST   ago
   ```

   b. Verify that `NetworkManager` can read the profile from the configuration file.

   ```
   nmcli -f TYPE,FILENAME,NAME connection
   ```

   ```
   TYPE      FILENAME
   NAME
   ethernet /etc/NetworkManager/system-connections/outputmconnection
   Example-Connection
   ethernet  /etc/sysconfig/network-scripts/ifcfg-enp0                enp0
   ```

   If the output doesn't display the newly created connection, verify that the `keyfile` permissions and the syntax used are correct.

   c. Run `nmcli connection show` to display the connection profile.

   ```
   nmcli connection show Example-Connection
   ```

   ```
   connection.id:                          Example-Connection
   connection.uuid:                        ce8d4422-9603-4d6f-
   b602-4f71992c49c2
   connection.stable-id:                   --
   connection.type:                        802-3-ethernet
   connection.interface-name:              --
   connection.autoconnect:                 yes
   ```

Creating a `keyfile` Connection Profile Manually

You can manually create a `NetworkManager` connection profile in a keyfile format.

> **Note**
>
> Manually creating or updating the configuration files can result in an unexpected network configuration. Another option would be to use `nmcli` in offline mode. See Creating a `keyfile` Connection Profile Using `nmcli` in Offline Mode.

1. If you're creating a profile for a hardware interface, such as Ethernet, display the hardware's MAC address.

   ```
   ip address show ens3
   ```

   ```
   2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc pfifo_fast state
   UP group default qlen 1000
       link/ether 02:00:17:03:b9:ae brd ff:ff:ff:ff:ff:ff
       ...
   ```

2. Use any text editor to create a connection profile that contains the network settings that you want to define for the connection.

   For example, if the connection uses DHCP, the profile would contain settings similar to the following example:

   ```
   [connection]
   id=myconnection
   type=ethernet
   autoconnect=true

   [ipv4]
   method=auto

   [ipv6]
   method=auto

   [ethernet]
   mac-address=02:00:17:03:b9:ae
   ```

3. Save the profile to `/etc/NetworkManager/system-connections/`*`filname`*`.nmconnection`.

   In this current procedure, the profile would be `/etc/NetworkManager/system-connections/myconnection.nmconnection`.

   > **Note**
   >
   > The defined ID variable, such as `myconnection`, doesn't need to be identical with the profile's file name, for example `myethernet.nmconnection`. When you change the profile by using the `nmcli` command, you can identify the profile by the defined ID (`myconnection`) or by the file name, but excluding the file extension name (`myethernet`).

4. Restrict the permissions of the profile.

   ```
   sudo chown root:root /etc/NetworkManager/system-connections/
   myconnection.nmconnection
   sudo chown 600 /etc/NetworkManager/system-connections/
   myconnection.nmconnection
   ```

5. Reload the connection profiles.

   ```
   sudo nmcli connection reload
   ```

6. Verify that `NetworkManager` can read the profile.

```
sudo nmcli -f NAME,UUID,FILENAME connection
```

```
NAME           UUID       FILENAME
myconnection   uuid        /etc/NetworkManager/system-connections/
myconnection.nmconnection
```

7. If you specified `false` for the profile's `autoconnect` parameter, then activate the connection.

```
sudo nmcli connection up myconnection
```

Creating a keyfile Connection Profile Using the Text Based User Interface

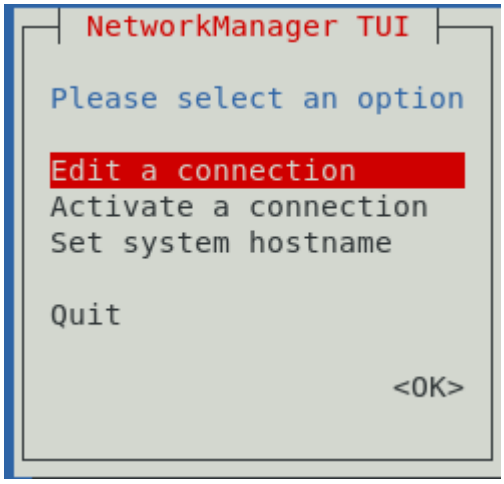
If not already installed, install the `NetworkManager-tui` package.

```
sudo dnf install -y NetworkManager-tui
```

1. Open `NetworkManager`'s text-based user interface.

```
sudo nmtui
```

Figure 2-3 TUI Main Menu

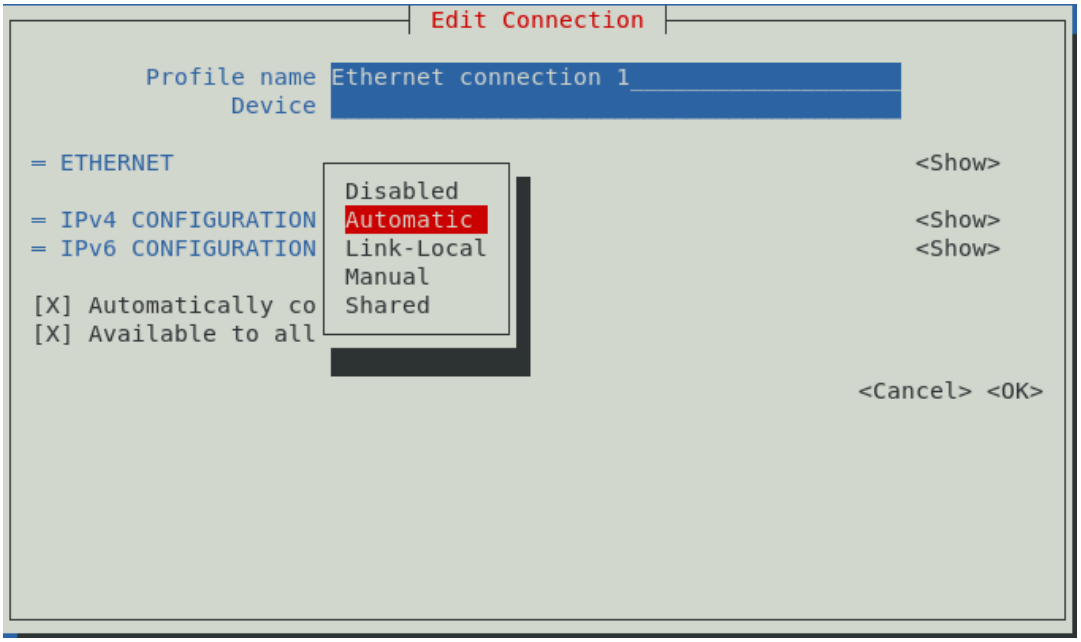


Navigate the tool using the up and down arrow keys, then press **Enter** to make a selection.

2. To add a connection, select **Edit a connection**, then select **Add**.
3. Select a connection type.

   The Edit Connection window opens.

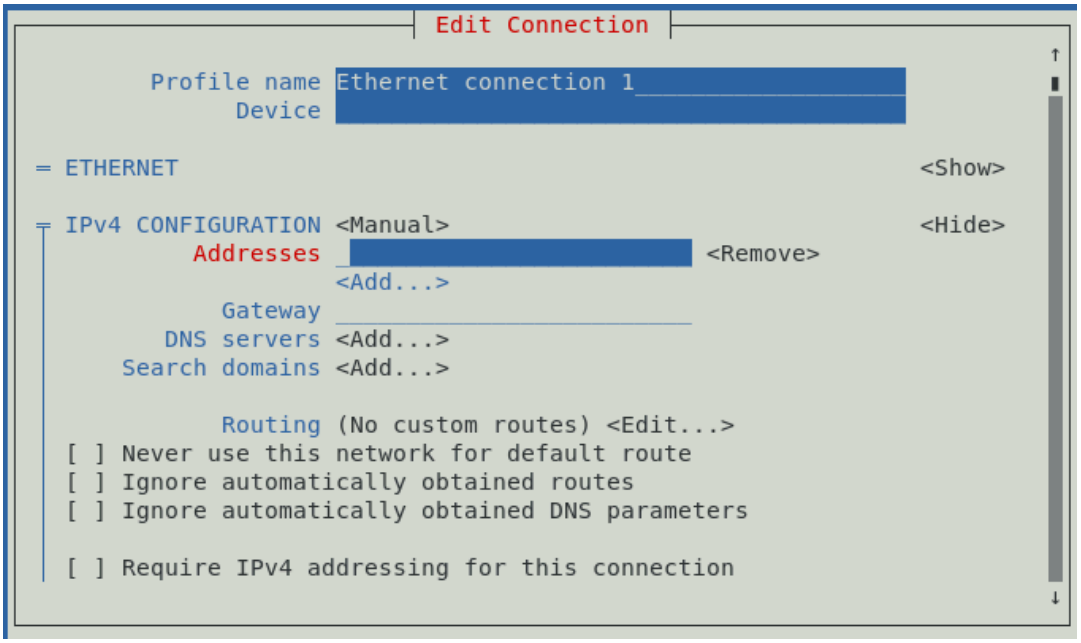
Figure 2-4 Edit Connection



4. As an option, specify a preferred profile name and the name of the device.
5. By default, IPv4 and IPv6 configurations are set to Automatic. To change the setting, select the **Automatic** field and press **Enter**. From the drop down list, select the type of IP configuration that you want to implement, such as Manual. Then, select the corresponding **Show** field.

   The fields that are displayed depend on the type of IP configuration that's selected. For example, to manually configure an IP address, selecting **Show** displays an address field, where you would enter an IP addresses for the interface, as the following figure illustrates.

Figure 2-5 Adding IP Addresses



6. Navigate through all the fields on the screen to ensure that the required information is specified.
7. After you have edited the connection, select **OK**.

3
Network Routing

A system uses its routing table to identify which network interface to use when sending packets to remote systems. For a system with only a single interface, configuring the IP address of a gateway system on the local network suffices to route packets to other networks. For example, see the image Figure 2-5, which shows a field where you can enter the IP address of the default gateway.

On systems that have several IP interfaces, you can define static routes so that traffic for a special host or network is forwarded to that network through the default gateway. You use the same tools to configure routing as you do to configure network interfaces.

Configuring a Static Route Using the Network Connection Editor

To create a static route to the 192.0.2.0/24 network through the gateway 198.51.100.1, ensure first that the default gateway 198.51.100.1 is reachable on the interface. Then, complete the following steps:

1. Start the editor.

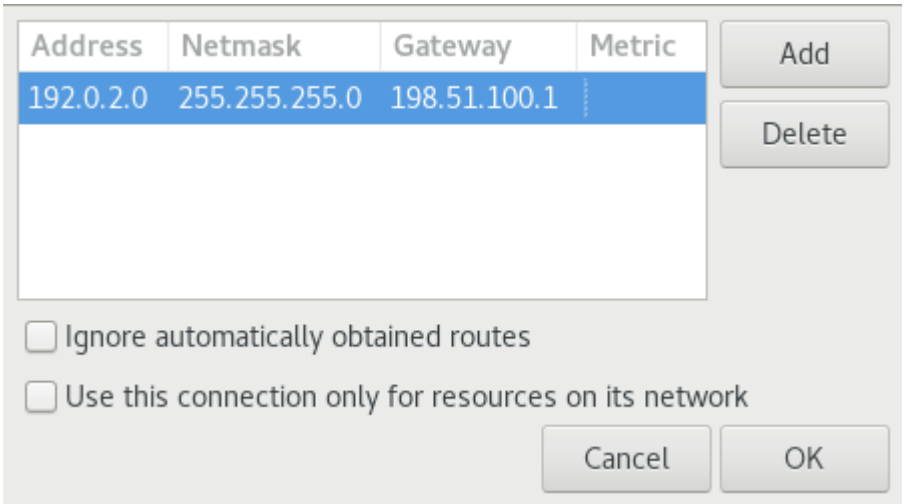
   ```
   nm-connection-editor
   ```

2. From the list of connections, select the device under the connection name for which you want to create a static route.

   For example, under `myconnection`, you would select the device `ens3`.

3. Click the settings icon (gear wheel) to edit the connection settings.
4. Click the IPv4 Settings tab.
5. Click **Routes**.
6. Click **Add**.
7. Enter the network's address and netmask for which the route is created, and specify the gateway IP address through which the route is established.

   You can optionally enter a metric value and select the other available options on display.



8. Click **OK** and then **Save**.
9. Back at the terminal window, restart the connection.

   This step causes the connection to temporarily drop.

   ```
   sudo nmcli connection up myconnection
   ```

10. Verify that the new route is active.

    ```
    ip route
    ```

    ```
    ...
    192.0.2.0/24 via 198.51.100.1 dev myconnection proto static metric 100
    ```

Configuring a Static Route Using the Command Line

To configure static routes with the `nmcli` command, use the following syntax:

```
nmcli connection modify connection_name +ipv4.routes "ip[/prefix] options(s)
attribute(s)"[next_hop] [metric] [attribute=value] [attribute=value] ..."
```

+ipv4.routes
The plus (+) sign indicates that you're creating an IPv4 route. Without the sign, the command changes an existing IPv4 setting.

connection-name
Connection name or label for which you're creating a static route.

***ip*[*/prefix*]**
IP address of the static route that you're creating. The IP address can also be in CIDR notation.

options
Options include next hop addresses and optional route metrics. These options are separated by spaces. For more information, see the `nm-settings-nmcli(5)` manual pages.

attributes
Attributes are entered as *attribute*=*value* and are also separated by spaces. Some attributes are `mtu`, `src`, `type`, `cwnd`, and so on. For more information, see the `nm-settings-nmcli(5)` manual pages.

Suppose that you have the following configurations:

- Name of the connection: `myconnection`
- Default gateway address: 198.51.100.1
- Network to which you want to create a static route: 192.0.2.0/24

To create the route, ensure first that the default gateway for the route is directly reachable on the interface. Then, do the following:

1. Create the static route.

   ```
   sudo nmcli connection modify myconnection +ipv4.routes "192.0.2.0/24
   198.51.100.1"
   ```

   To create several static routes in a single command, separate the *route gateway* entries with commas, for example:

   ```
   sudo nmcli connection modify myconnection +ipv4.routes "192.0.2.0/24
   198.51.100.1, 203.0.113.0/24 198.51.100.1"
   ```

2. Verify the new routing configuration.

   ```
   nmcli connection show myconnection


   --
   ipv4.routes:   { ip = 192.0.2.0/24, nh = 198.51.100.1 }
   --
   ```

3. Restart the network connection.

   This step causes the connection to temporarily drop.

   ```
   sudo nmcli connection up myconnection
   ```

4. Verify that the new route is active.

   ```
   ip route


   ...
   192.0.2.0/24 via 198.51.100.1 dev example proto static metric 100
   ```

Configuring a Static Route Using the Command Line in Interactive Mode

You can use the `nmcli` command in interactive mode to configure network settings, including configuring static routes. When in interactive mode, the `nmcli>` prompt appears where you can run commands to configure static routes for a specific connection profile.

This procedure assumes the following network settings for creating the static route:

- Name of the connection: `myconnection`
- Default gateway address: 198.51.100.1
- Network to which you want to create a static route: 192.0.2.0/24

To create the route, ensure first that the default gateway for the route is directly reachable on the interface. Then, do the following:

1. Start `nmcli` in interactive mode.

   ```
   sudo nmcli connection modify myconnection


   nmcli>
   ```

2. Create the static route.

   ```
   nmcli> set ipv4.routes 192.0.2.0/24 198.51.100.1
   ```

3. Display the new configuration.

   ```
   nmcli> print


   ...
   ipv4.routes:        { ip = 192.0.2.1/24, nh = 198.51.100.1 }
   ...
   ```

4. Save the configuration.

   ```
   nmcli> save persistent
   ```

5. Restart the network connection.

   This step causes the connection to temporarily drop.

   ```
   nmcli> activate myconnection
   ```

6. Exit the interactive mode.

   ```
   nmcli> quit
   ```

7. Verify that the new route is active.

```
ip route


...
192.0.2.0/24 via 198.51.100.1 dev example proto static metric 100
```

4

Running Scripts When Network Events Occur

You can configure the system to respond to network events by providing scripts for `NetworkManager-dispatcher` to run. Use a script, for example, to mount a remote file system when a device is brought up or send a notification when a device loses connectivity.

1. Save an executable script in `/etc/NetworkManager/dispatcher.d/` or one of its subdirectories.

   When `NetworkManager-dispatcher` runs a script, it passes two arguments to the script:

   - The name of the interface on which an action occurred.
   - The action that occurred.

   In addition, environment variables related to the network are available for you to use in the script.

   See the `NetworkManager-dispatcher (8)` manual page for a full list of actions and environment variables you can use in a script.

   Subdirectories within `/etc/NetworkManager/dispatcher.d/` provide special handling of scripts.

Subdirectory	Details
`pre-up.d`	Place or symbolically link scripts responding to `pre-up` or `vpn-pre-up` actions in this subdirectory.
`pre-down.d`	Place or symbolically link scripts responding to `pre-down` or `vpn-pre-down` actions in this subdirectory.
`no-wait.d`	Create a symbolic link from the script to this subdirectory if you want the script to run immediately when an action occurs. These scripts run in parallel, and don't wait for scripts already running to end.

   > **Note**
   >
   > Depending on the SELinux policies and security contexts configured on the system, SELinux might prevent scripts from running if they are symbolic links into a subdirectory in `/etc/NetworkManager/dispatcher.d/`. To ensure that scripts run, either place scripts directly in the subdirectory, or update the SELinux configuration. For more information, see Oracle Linux: Administering SELinux.

   Because `NetworkManager-dispatcher` runs scripts in `/etc/NetworkManager/dispatcher.d/` in alphabetical order, you can prefix the file name with a number to enforce the execution order of scripts. For example: `/etc/NetworkManager/dispatcher.d/10-myscript`.

2. Set ownership and permissions for the file.

The following settings are required:

a. Change the file owner to `root`.

```
sudo chown root /etc/NetworkManager/dispatcher.d/10-myscript
```

b. Disable write access for group and other, and disable `setuid`.

```
sudo chmod 0700 /etc/NetworkManager/dispatcher.d/10-myscript
```

When network actions occur, `NetworkManager.service` starts `NetworkManager-dispatcher.service`, which runs the scripts in `/etc/NetworkManager/dispatcher.d/` following these parameters:

- One script runs at a time.
- Scripts run in sequence based on the order in which network events occur.
- After the dispatcher service queues a script, the script always runs, even if a later action makes the script unnecessary.
- NetworkManager terminates scripts if they run for too long.

5

High Availability for Network Services

For systems that provide network services to clients inside the network, network availability becomes a priority to ensure that the services are continuous and interruptions are prevented. With virtual local area networks (VLANs), you can also organize the network such that systems with similar functions are grouped together as though they belong to their own virtual networks. This feature improves network management and administration.

For a system to avail of these advanced features, it must have several NICs. The more NICs, the better assurances of network availability that a server can provide.

Network Bonding

A system's physical network interfaces that are connected to a network switch can be grouped together into a single logical interface to provide better throughput or availability. This grouping, or aggregation, of physical network interfaces is known as a network bond.

A bonded network interface can increase data throughput by load balancing or can provide redundancy by activating failover from one component device to another. By default, a bonded interface appears similar to a normal network device to the kernel, but it sends out network packets over the available secondary devices by using a round-robin scheduler. You can configure bonding module parameters in the bonded interface's configuration file to alter the behavior of load-balancing and device failover.

The network bonding driver within the kernel can be used to configure the network bond in different modes to take advantage of different bonding features, depending on the requirements and the available network infrastructure. For example, the `balance-rr` mode can be used to provide basic round-robin load-balancing and fault tolerance across a set of physical network interfaces; while the `active-backup` mode provides basic fault tolerance for high availability configurations. Some bonding modes, such as `802.3ad`, or dynamic link aggregation, require particular hardware features and configuration on the switch that the physical interfaces connect to. Basic load-balancing modes (`balance-rr` and `balance-xor`) work with any switch that supports EtherChannel or trunking. Advanced load-balancing modes (`balance-tlb` and `balance-alb`) don't impose requirements on the switching hardware, but do require that the device driver for each component interfaces implement certain specific features such as support for `ethtool` or the ability to change the hardware address while the device is active.

For more information on the kernel bonding driver, see the upstream documentation at https://www.kernel.org/doc/Documentation/networking/bonding.txt or included at `/usr/share/doc/iputils-*/README.bonding`.

> **Note**
>
> For network configurations where systems are directly cabled together for high availability, a switch is required to support certain network interface bonding features such as automatic failover. Otherwise, the mechanism might not work.

Configuring Network Bonding Using the Command Line

Set up a network bond using the `nmcli` command line tool.

1. Add a bond interface using the `nmcli connection add` command.

   ```
   sudo nmcli connection add type bond con-name "Bond Connection 1" ifname
   bond0 bond.options "mode=active-backup"
   ```

   Take note to set the bond connection name, the bond interface name, and, importantly, the bond mode option. In this example, the mode is set to `active-backup`. If you don't set the bond connection name, then the bond interface name is also used as the connection name.

2. Optional: Configure the IP address for the bond interface using the `nmcli connection modify` command.

   By default the interface is configured to use DHCP, but if you require static IP addressing, manually configure the address. For example, to configure IPv4 settings for the bond, type:

   ```
   sudo nmcli connection modify "Bond Connection 1" \
   ipv4.addresses '192.0.2.2/24' \
   ipv4.gateway '192.0.2.1' \
   ipv4.dns '192.0.2.254' \
   ipv4.method manual
   ```

3. Add the physical network interfaces to the bond as secondary-type interfaces using the `nmcli connection add` command.

   For example:

   ```
   sudo nmcli connection add type ethernet slave-type bond con-name bond0-if1
   ifname enp1s0 master bond0
   ```

   ```
   sudo nmcli connection add type ethernet slave-type bond con-name bond0-if2
   ifname enp2s0 master bond0
   ```

   Give each secondary a connection name, and select the interface name for each interface that you want to add. You can get a list of available interfaces by running the `nmcli device` command. Specify the interface name of the bond to which you want to attach the secondary network interfaces.

4. Start the bond interface.

   ```
   sudo nmcli connection up "Bond Connection 1"
   ```

5. Verify that the network interfaces have been added to the bond correctly.

You can check this by looking at the device list again.

```
sudo nmcli device
```

```
...
enp1s0   ethernet  connected  bond0-if1
enp2s0   ethernet  connected  bond0-if2
```

Configuring Network Bonding Using the Network Connections Editor

Set up a network bond using the `nm-connection-editor` graphical interface.

1. Start the editor:

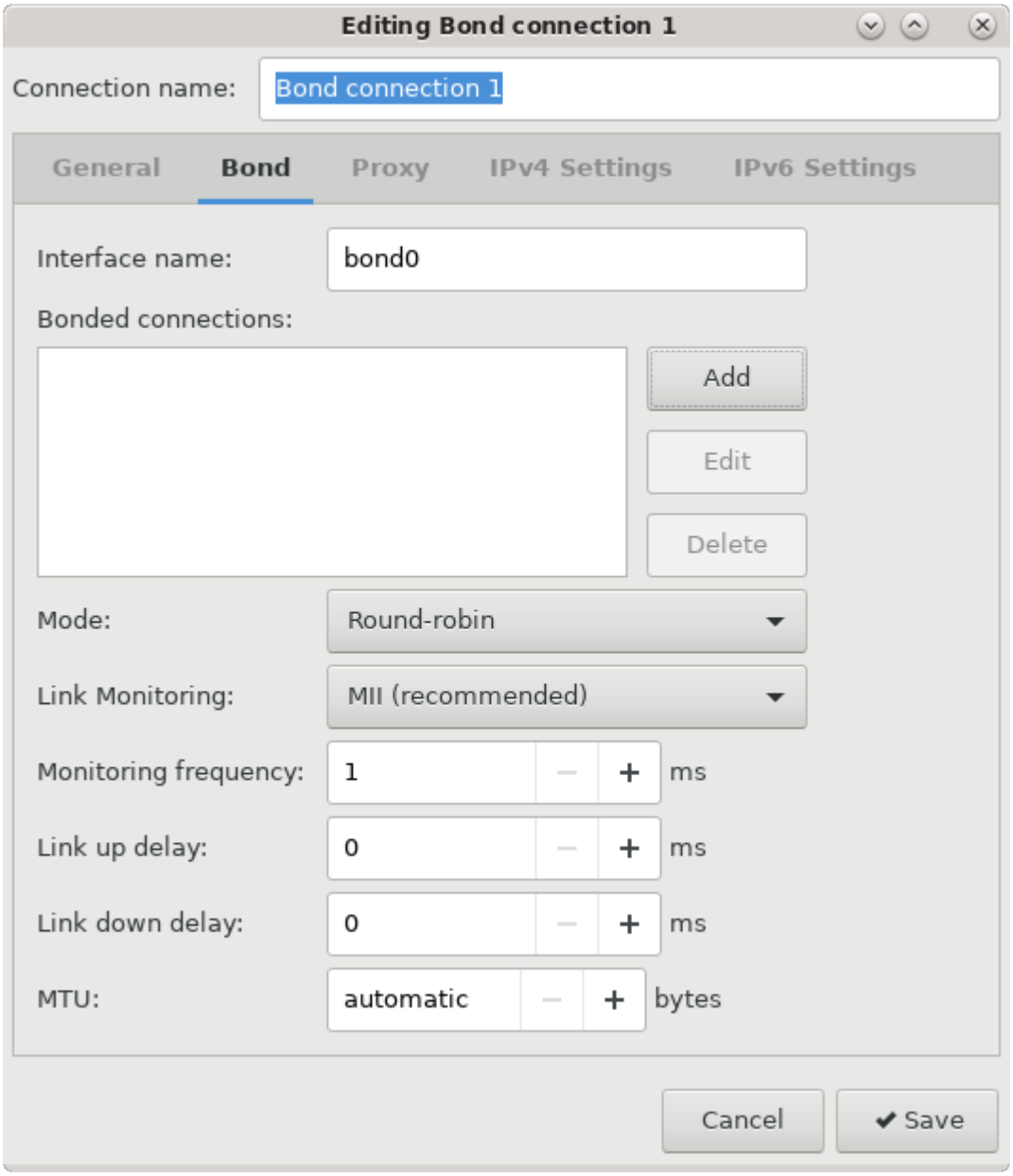
   ```
   sudo nm-connection-editor
   ```

   The Network Connections window opens.
2. To add a connection, click the plus (+) button at the bottom of the window.

   Another window opens that prompts you for the type of connection to create.
3. From the window's drop down list and under the Virtual section, select **Bond**, then click **Create**.

   The Network Bond Editor window opens.

   Network Bond Editor



4. Optional: Configure a connection name for the bond.
5. For each physical network interface you want to add to the network bond, complete the following steps:
   a. Click the **Add** button.

      A new window opens where you can select the type of physical interface to add to the network bond.
   b. Select the type of connection you want to create, then click **Create** to configure the secondary interface.

      For example, you can select the **Ethernet** type to add an Ethernet interface to the network bond.
   c. Optional: Configure a name for the secondary interface.
   d. In the **Device** field, select the physical network interface to add as a secondary to the bond.

> **Note**
>
> If a device is already configured for networking, it's not listed as available to configure within the bond.

e. Click **Save** to add the secondary device to the network bond.

6. In the **Mode** drop-down list, select the bonding mode that you want to use for the network bond.

> **Note**
>
> Some modes might require more configuration on the network switch.

7. Configure other bond parameters such as link monitoring as required if you don't want to use the default settings.

   If you don't intend to use DHCP for network bond IP configuration, set the IP addressing by clicking on the **IPv4** and **IPv6** tabs.

8. Click **Save** to save the configuration and create the network bond.

Verifying the Network Bond Status

1. Run the following command to obtain information about the network bond with device name *bond0*:

   ```
   cat /proc/net/bonding/bond0
   ```

   The output shows the bond configuration and status, including which bond secondaries are active. The output also provides information about the status of each secondary interface.

2. Temporarily disconnect the physical cable that's connected to one of the secondary interfaces.

   No other reliable method is available to test link failure.

3. Check the status of the bond link as shown in the initial step for this procedure.

   The status of the secondary interface would indicate that the interface is down and a link failure has occurred.

VLANs and Untagged Data Frames

A VLAN is a group of machines that can communicate as though they're attached to the same physical network. With a VLAN, you can group systems regardless of their actual physical location on a LAN.

In a VLAN that uses untagged data frames, you create the broadcast domain by assigning the ports of network switches to the same permanent VLAN ID or PVID (other than 1, which is the default VLAN). All the ports that you assign with this PVID are in a single broadcast domain. Broadcasts between devices in the same VLAN aren't visible to other ports with a different VLAN, even if they exist on the same switch.

Creating VLAN Devices by Using the nmcli Command

You can use the `nmcli` command to create a VLAN device for an Ethernet interface.

- Run the `nmcli` command with the following arguments:

  ```
  sudo nmcli con add type vlan con-name new-name ifname new-ifname dev
  device-name id new-id-number
  ```

  Running the following command sets up the VLAN device `bond0-pvid10` with a PVID of 10 for the bonded interface `bond0`. In addition to the regular interface, `bond0`, which uses the physical LAN, you now have a VLAN device, `bond0-pvid10`, which can use untagged frames to access the virtual LAN.

  > **Note**
  >
  > You don't need to create virtual interfaces for the component interfaces of a bonded interface. However, you must set the PVID on each switch port to which they connect.

  ```
  sudo nmcli con add type vlan con-name bond0-pvid10 ifname bond0-pvid10 dev
  bond0 id 10
  ```

  You can also use the command to set up a VLAN device for a non bonded interface, for example:

  ```
  sudo nmcli con add type vlan con-name en1-pvid5 ifname en1-pvid5 dev en1
  id 5
  ```

To obtain information about the configured VLAN interfaces, view the files in the `/proc/net/vlan` directory.

Creating VLAN Devices by Using the ip Command

You can create a VLAN device using the `ip` utility.

> **Note**
>
> VLAN devices created using `ip` don't persist across system reboots.

- Run the `ip` command with the following arguments:

  ```
  ip link add link device name name-for-device type vlan id id-for-vlan
  ```

The following example shows the input that would create a VLAN device on the interface `en1` named `en1.5` with a PVID of `5`:

```
sudo ip link add link eth1 name eth1.5 type vlan id 5
```

For more information, see the following manual pages:

- `ip(8)`
- `ip-link(8)`

6

Network Address Translation

Network Address Translation (NAT) is a process that assigns a public address to a computer or a group of computers inside a private network by using a different address scheme. The public IP address masquerades all the requests as though they're going to one server, rather than several servers. NAT is useful for limiting the number of public IP addresses that an organization must finance. NAT also provides extra security by hiding the details of internal networks.

The `netfilter` kernel subsystem provides the `nat` table to implement NAT, in addition to its tables for packet filtering. The kernel consults the `nat` table whenever it handles a packet that creates a new incoming or outgoing connection.

By default, IP forwarding is enabled and the system can route packets among configured network interfaces. To check whether IP forwarding is enabled, use the following command:

```
sudo sysctl -a | grep ip_forward
```

In the ensuing output, a value of `1` for `net.ipv4.ip_forward` indicates that IP forwarding is enabled.

You can change the status of IP forwarding on the system by using the following command:

```
sudo sysctl -w net.ipv4.ip_forward=[0|1]
```

The new status is displayed when you run the command. To make the change persist across system reboots, copy the command output line and add it to the `/etc/sysctl.conf` file.

You can also use the Firewall Configuration GUI (`firewall-config`) to configure masquerading and port forwarding. See Oracle Linux 10: Configuring the Firewall for more information.