Oracle Linux 7

Setting Up Networking

F32794-08
October 2022

Oracle Linux 7 Setting Up Networking,

F32794-08

Contents

7 Configuring High Availability Features

8 Configuring Load Balancing

9 Configuring the VNC Service

Preface

Oracle® Linux 7: Setting Up Networking provides information about configuring networking for Oracle Linux 7 systems.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	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/.

For information about the accessibility of the Oracle Help Center, see the Oracle Accessibility Conformance Report at https://www.oracle.com/corporate/accessibility/templates/t2-11535.html.

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.

1

Configuring the System's Network

This chapter describes how to configure a system's network interfaces and network routing.

About Network Interface Names

Each physical and virtual network device on an Oracle Linux system has an associated configuration file named `ifcfg-`*`interface`* in the `/etc/sysconfig/network-scripts` directory, where *interface* is the name of the interface. For example:

```
cd /etc/sysconfig/network-scripts
ls ifcfg-*

ifcfg-em1  ifcfg-em2  ifcfg-lo
```

In this example, there are two configuration files for motherboard-based Ethernet interfaces, `ifcfg-em1` and `ifcfg-em2`, and one for the loopback interface, `ifcfg-lo`. The system reads the configuration files at boot time to configure the network interfaces.

On your system, you might see other names for network interfaces. See About Network Interface Names.

The following are sample entries from an `ifcfg-em1` file for a network interface that obtains its IP address using the Dynamic Host Configuration Protocol (DHCP):

```
DEVICE="em1"
NM_CONTROLLED="yes"
ONBOOT=yes
USERCTL=no
TYPE=Ethernet
BOOTPROTO=dhcp
DEFROUTE=yes
IPV4_FAILURE_FATAL=yes
IPV6INIT=no
NAME="System em1"
UUID=5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03
HWADDR=08:00:27:16:C3:33
PEERDNS=yes
PEERROUTES=yes
```

If the interface is configured with a static IP address, the file contains entries such as the following:

```
DEVICE="em1"
NM_CONTROLLED="yes"
ONBOOT=yes
USERCTL=no
TYPE=Ethernet
BOOTPROTO=none
DEFROUTE=yes
IPV4_FAILURE_FATAL=yes
IPV6INIT=no
NAME="System em1"
```

```
UUID=5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03
HWADDR=08:00:27:16:C3:33
IPADDR=192.168.1.101
NETMASK=255.255.255.0
BROADCAST=192.168.1.255
PEERDNS=yes
PEERROUTES=yes
```

The following configuration parameters are typically used in interface configuration files:

BOOTPROTO
How the interface obtains its IP address:

bootp
Bootstrap Protocol (BOOTP).

dhcp
Dynamic Host Configuration Protocol (DHCP).

none
Statically configured IP address.

BROADCAST
IPv4 broadcast address.

DEFROUTE
Whether this interface is the default route.

DEVICE
Name of the physical network interface device (or a PPP logical device).

GATEWAY *N*
IPv4 gateway address for the interface. As an interface can be associated with several combinations of IP address, network mask prefix length, and gateway address, these are numbered starting from 0.

HWADDR
Media access control (MAC) address of an Ethernet device.

IPADDR *N*
IPv4 address of the interface.

IPV4_FAILURE_FATAL
Whether the device is disabled if IPv4 configuration fails.

IPV6_DEFAULTGW
IPv6 gateway address for the interface. For example:
`IPV6_DEFAULTGW=2001:0daa::2%em1.`

IPV6_FAILURE_FATAL
Whether the device is disabled if IPv6 configuration fails.

IPV6ADDR
IPv6 address of the interface in CIDR notation, including the network mask prefix length. For example: `IPV6ADDR="2001:0db8:1e11:115b::1/32"`

IPV6INIT
Whether to enable IPv6 for the interface.

MASTER
Specifies the name of the primary bonded interface, of which this interface is backup.

NAME
Name of the interface as displayed in the Network Connections GUI.

NETWORK
IPV4 address of the network.

NM_CONTROLLED
Whether the network interface device is controlled by the network management daemon, `NetworkManager`.

ONBOOT
Whether the interface is activated at boot time.

PEERDNS
Whether the `/etc/resolv.conf` file used for DNS resolution contains information obtained from the DHCP server.

PEERROUTES
Whether the information for the routing table entry that defines the default gateway for the interface is obtained from the DHCP server.

PREFIX *N*
Length of the IPv4 network mask prefix for the interface.

SLAVE
Specifies that this interface is a backup of a bonded interface.

TYPE
Interface type.

USERCTL
Whether users other than `root` can control the state of this interface.

UUID
Universally unique identifier for the network interface device.

About Network Interface Names

Network interface names are based on information derived from the system BIOS or alternatively from a device's firmware, system path, or MAC address. This feature ensures that interface names persist across system reboots, hardware reconfiguration, and updates to device drivers and the kernel.

If you enable the `biosdevname` boot option (`biosdevname=1`), the `biosdevname` plugin to the udev device manager assigns names to network interfaces as follows:

- Ethernet interfaces on the motherboard are named `em` *N*, where *N* is the number of the interface starting from 1.
- Network interfaces on a PCI card are named `p` *S* `p` *P*, where *S* is the slot number and *P* is the port number.

- Virtual interfaces are named `p` *S* `p` *P* `_` *V*, where *S* is the slot number, *P* is the port number, and *V* is the virtual interface number.

If `biosdevname` is set to 0 (the default), `systemd` naming assigns the prefixes, `en`, `wl`, and `ww` to Ethernet, wireless LAN, and wireless WAN interfaces respectively. The prefix is followed by a suffix based on the hardware configuration, system bus configuration, or MAC address of the device:

`o` *N*
Onboard device with index number *N*.

`p`*B*`s`*S*`[f`*F*`][d`*D*`]
PCI device with bus number *B*, slot number *S*, function number *F*, and device ID *D*.

`p`*B*`s`*S*`[f`*F*`][u`*P*`]...[c`*C*`][i`*I*`]
USB device with bus number *B*, slot number *S*, function number *F*, port number *P*, configuration number *C*, and interface number *I*.

`s`*S*`[f`*F*`][d`*D*`]
Hot-plug device with slot number *S*, function number *F*, and device ID *D*.

`x` *M*
Device with MAC address *M*.

For example, an Ethernet port on the motherboard might be named `eno1` or `em1`, depending on whether the value of `biosdevname` is 0 or 1.

The kernel assigns a legacy, unpredictable network interface name (`eth` *N* and `wlan` *N*) only if it cannot discover any information about the device that would allow it to disambiguate the device from other such devices. You can use the `net.ifnames=0` boot parameter to reinstate the legacy naming scheme.

> **Caution:**
>
> Using the `net.ifnames` or `biosdevname` boot parameters to change the naming scheme can rendering existing firewall rules invalid. Changing the naming scheme can also affect other software that refers to network interface names.

About Network Configuration Files

The following sections describe additional network configuration files that you might need to configure on a system.

About the /etc/hosts File

The `/etc/hosts` file associates host names with IP addresses. It allows the system to look up (`resolve`) the IP address of a host given its name, or the name given the IP address. Most networks use DNS (Domain Name Service) to perform address or name resolution. Even if your network uses DNS, it is usual to include lines in this file that specify the IPv4 and IPv6 addresses of the loopback device, for example:

```
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
```

The first and second column contains the IP address and host name. Additional columns contain aliases for the host name.

For more information, see the `hosts(5)` manual page.

About the /etc/nsswitch.conf File

The `/etc/nsswitch.conf` file configures how the system uses various databases and name resolution mechanisms. The first field of entries in this file identifies the name of the database. The second field defines a list of resolution mechanisms in the order in which the system attempts to resolve queries on the database.

The following example hosts definition from `/etc/nsswitch.conf` indicates that the system first attempts to resolve host names and IP addresses by querying `files` (that is, `/etc/hosts`) and, if that fails, next by querying a DNS server, and last of all, by querying NIS+ (NIS version 3) :

```
hosts:      files dns nisplus
```

For more information, see the `nsswitch.conf(5)` manual page.

About the /etc/resolv.conf File

The `/etc/resolv.conf` file defines how the system uses DNS to resolve host names and IP addresses. This file usually contains a line specifying the search domains and up to three lines that specify the IP addresses of DNS server. The following entries from `/etc/resolv.conf` configure two search domains and three DNS servers:

```
search us.mydomain.com mydomain.com
nameserver 192.168.154.3
nameserver 192.168.154.4
nameserver 10.216.106.3
```

If your system obtains its IP address from a DHCP server, it is usual for the system to configure the contents of this file with information also obtained using DHCP.

For more information, see the `resolv.conf(5)` manual page.

About the /etc/sysconfig/network File

The `/etc/sysconfig/network` file specifies additional information that is valid to all network interfaces on the system. The following entries from `/etc/sysconfig/network` define that IPv4 networking is enabled, IPv6 networking is not enabled, the host name of the system, and the IP address of the default network gateway:

```
NETWORKING=yes
NETWORKING_IPV6=no
HOSTNAME=host20.mydomain.com
GATEWAY=192.168.1.1
```

> **Note:**
>
> In previous releases of Oracle Linux, the host name of the system was defined in `/etc/sysconfig/network`. The host name is now defined in `/etc/hostname` and can be changed by using the `hostnamectl` command. The host name must be a fully qualified domain name (FQDN), for example, `host20.mydomain.com`, instead of a simple short name.
>
> Additionally, system-wide default localization settings such as the default language, keyboard, and console font were defined in `/etc/sysconfig/i18n`. These settings are now defined in `/etc/locale.conf` and `/etc/vconsole.conf`.
>
> For more information, see the `hostname(5)`, `hostnamectl(1)`, `locale.conf(5)`, and `vconsole.conf(5)` manual pages.

Command-Line Network Configuration Interfaces

If the `NetworkManager` service is running, you can use the `nmcli` command to display the state of the system's physical network interfaces, for example:

```
sudo nmcli device status

DEVICE  TYPE      STATE
em1     ethernet  connected
em2     ethernet  connected
lo      loopback  unmanaged
```

You can use the `ip` command to display the status of an interface, for debugging, or for system tuning. For example, to display the status of all active interfaces:

```
sudo ip addr show

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: em1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
qlen 1000
    link/ether 08:00:27:16:c3:33 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global em1
    inet6 fe80::a00:27ff:fe16:c333/64 scope link
       valid_lft forever preferred_lft forever
```

For each network interface, the output shows the current IP address, and the status of the interface. To display the status of a single interface such as `em1`, specify its name as shown here:

```
sudo ip addr show dev em1

2: em1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
qlen 1000
    link/ether 08:00:27:16:c3:33 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global em1
```

```
    inet6 fe80::a00:27ff:fe16:c333/64 scope link
       valid_lft forever preferred_lft forever
```

You can also use `ip` to set properties and activate a network interface. The following example sets the IP address of the `em2` interface and activates it:

```
sudo ip addr add 10.1.1.1/24 dev em2
sudo ip link set em2 up
```

> **Note:**
>
> You might be used to using the `ifconfig` command to perform these operations. However, `ifconfig` is considered obsolete and will eventually be replaced altogether by the `ip` command.

Any settings that you configure for network interfaces using `ip` do not persist across system reboots. To make the changes permanent, set the properties in the `/etc/sysconfig/network-scripts/ifcfg-`*`interface`* file.

Any changes that you make to an interface file in `/etc/sysconfig/network-scripts` do not take effect until you restart the network service or bring the interface down and back up again. For example, to restart the network service:

```
sudo systemctl restart network
```

To restart an individual interface, you can use the `ifup` or `ifdown` commands, which invoke the script in `/etc/sysconfig/network-scripts` that corresponds to the interface type, for example:

```
sudo ifdown em1
sudo ifup em1

Connection successfully activated
(D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/5)
```

Alternatively, you can use the `ip` command to stop and start network activity on an interface without completely tearing down and rebuilding its configuration:

```
sudo ip link set em1 down
sudo ip link set em1 up
```

The `ethtool` utility is useful for diagnosing potentially mismatched settings that affect performance, and allows you to query and set the low-level properties of a network device. Any changes that you make using `ethtool` do not persist across a reboot. To make the changes permanent, modify the settings in the device's `ifcfg-`*`interface`* file in `/etc/sysconfig/network-scripts`.

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

Configuring Network Interfaces Using Graphical Interfaces

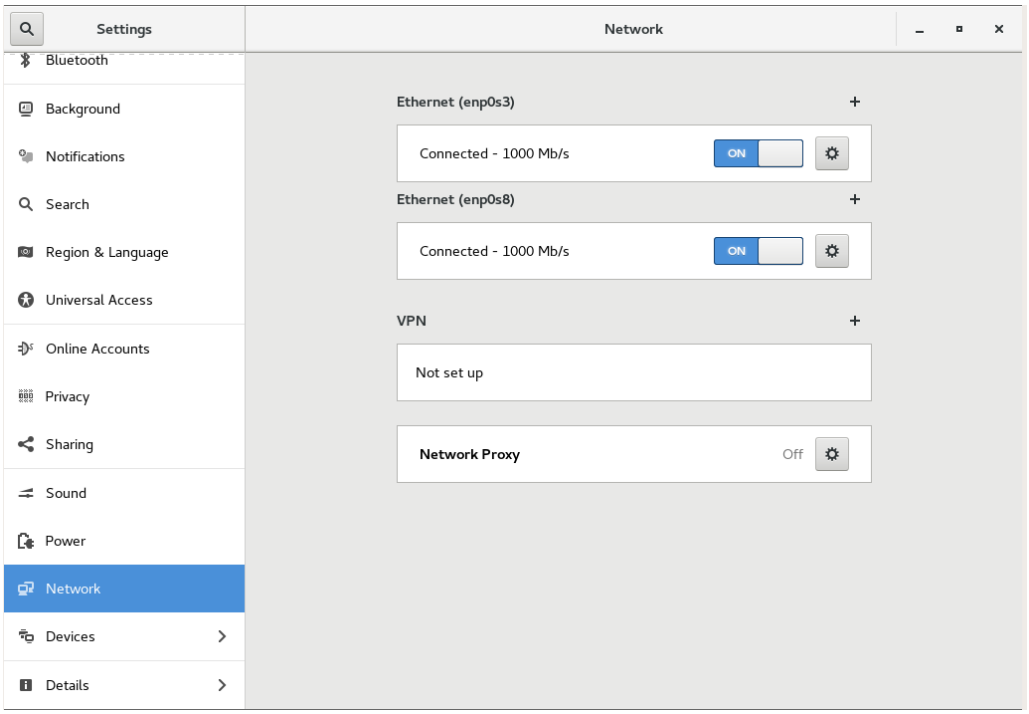
> **Note:**
>
> The `NetworkManager` service and the `nmcli` command are included in the `NetworkManager` package. The Network Connections editor is included in the `nm-connection-editor` package.

The `NetworkManager` service dynamically detects and configures network connections. You can click on the network icon in the GNOME notification area to obtain information about the status of the network interfaces and to manage network connections:

- To enable or disable a network interface from the pull-down menu, use the On/Off toggle.
- To display the Settings window, select **Network Settings** from the drop-down menu.

Figure 1-2 shows the Network Settings editor.

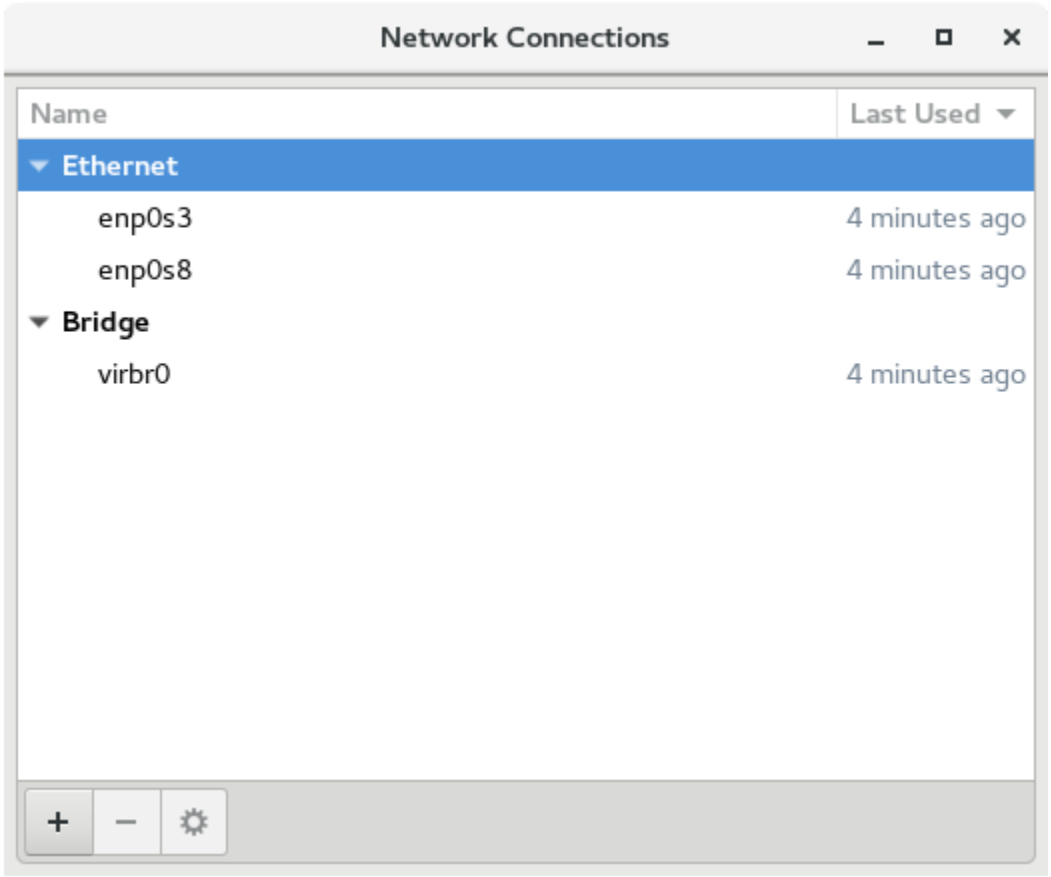
Figure 1-1 Network Settings Editor



To edit an existing interface, select it from the list and click the gear icon. You can add a profile to any interface to provide alternate configurations that you can use at any point in time. You can equally use this window to configure a network proxy or add an enable a Virtual Private Network (VPN) connection.

To perform more complex configuration and to add additional connection types, use the Network Connections editor. This tool allows you to configure wired, wireless, mobile broadband, VPN, Digital Subscriber Link (DSL), and virtual (bond, bridge, team, and VLAN) interfaces. You can open this window by using the `nm-connection-editor` command. Figure 1-2 shows the Network Connections editor.

Figure 1-2 Network Connections Editor



To create a new network interface, click the **+** icon, select the type of interface (hardware, virtual, or VPN) and click **Create**. To edit an existing interface, select it from the list and click the gear icon. To remove a selected interface, click the **-** icon.

You can also use the `nmcli` command to manage network connections through `NetworkManager`. For more information, see the `nmcli(1)` manual page.

About Network Interface Bonding

Network interface bonding combines multiple network connections into a single logical interface. A bonded network interface can increase data throughput by load balancing or can provide redundancy by allowing failover from one component device to another. By default, a bonded interface appears like a normal network device to the kernel, but it sends out network packets over the available secondary devices by using a simple 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.

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`) do not 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 modify the hardware address while the device is active. For more information see `/usr/share/doc/iputils-*/README.bonding`.

Configuring Network Interface Bonding

The bonding driver that is provided with the Oracle Linux kernel allows you to aggregate multiple network interfaces, such as `em1` and `em2`, into a single logical interface such as `bond0`. You can use the Network Settings editor to create the bond and then add network interfaces to this bond. Alternatively, you can use the `nmcli` command to create and configure the bond.

To create and configure a bonded interface from the command line:

1. Create the bond:

   ```
   sudo nmcli con add type bond con-name bond0 ifname bond0 mode balance-rr
   ```

   This example sets the name of the bond to `bond0` and its mode to `balance-rr`. For more information about the available options for load balancing or ARP link monitoring, see `/usr/share/doc/iputils-*/README.bonding` and the `nmcli(1)` manual page.

2. Add each interface to the bond:

   ```
   sudo nmcli con add type bond-slave ifname em1 master bond0
   sudo nmcli con add type bond-slave ifname em2 master bond0
   ```

   These commands add the `em1` and `em2` interfaces to `bond0`.

3. Restart the `NetworkManager` service:

   ```
   sudo systemctl restart NetworkManager
   ```

   After restarting the service, the bonded interface is available for use.

About Network Interface Teaming

> **Note:**
>
> Network interface teaming requires Unbreakable Enterprise Kernel Release 3 (UEK R3) Quarterly Update 7 or later.

Network interface teaming is similar to network interface bonding and provides a way of implementing link aggregation that is relatively maintenance-free, as well as being simpler to modify, expand, and debug as compared with bonding.

A lightweight kernel driver implements teaming and the `teamd` daemon implements load-balancing and failover schemes termed *runners*. The following standard runners are defined:

activebackup
Monitors the link for changes and selects the active port that is used to send packets.

broadcast
Sends packets on all member ports.

lacp
Provides load balancing by implementing the Link Aggregation Control Protocol 802.3ad on the member ports.

loadbalance
In passive mode, uses the BPF hash function to select the port that is used to send packets. In active mode, uses a balancing algorithm to distribute outgoing packets over the available ports.

random
Selects a port at random to send each outgoing packet.

> **Note:**
>
> UEK R3 does not currently support this runner mode.

roundrobin
Transmits packets over the available ports in a round-robin fashion.

For specialized applications, you can create customized runners that `teamd` can interpret. The `teamdctl` command allows you to control the operation of `teamd`.

For more information, see the `teamd.conf(5)` manual page.

Configuring Network Interface Teaming

You can configure a teamed interface by creating JSON-format definitions that specify the properties of the team and each of its component interfaces. The `teamd` daemon then interprets these definitions. You can use the JSON-format definitions to create a team interface by starting the `teamd` daemon manually, by editing interface definition files in `/etc/sysconfig/network-scripts`, by using the `nmcli` command, or by using the Network Configuration editor (`nm-connection-editor`). This section describes the first of these methods.

To create a teamed interface by starting `teamd` manually:

1. Create a JSON-format definition file for the team and its component ports. For sample configurations, see the files under `/usr/share/doc/teamd-*/example_configs/`.

   The following example, which is based on the contents of the file `activebackup_ethtool_1.conf`, defines an active-backup configuration where `em4` is configured as the primary port and `em3` as the backup port and these ports are monitored by `ethtool`.

   ```
   {
           "device":       "team0",
           "runner":       {"name": "activebackup"},
           "link_watch":   {"name": "ethtool"},
           "ports":        {
   ```

```
                "em3": {
                        "prio": -10,
                        "sticky": true
                },
                "em4": {
                        "prio": 100
                }
        }
}
```

2. Use the `ip` command to bring down the component ports:

   ```
   sudo ip link set em3 down
   sudo ip link set em4 down
   ```

   Active interfaces cannot be added to a team.

3. Start an instance of the `teamd` daemon and have it create the teamed interface by reading the configuration file (in this example, `/root/team_config/team0.conf`):

   ```
   sudo teamd -g -f /root/team_config/team0.conf -d

   Using team device "team0".
   Using PID file "/var/run/teamd/team0.pid"
   Using config file "/root/team_config/team0.conf"
   ```

   The `-g` option displays debugging messages and can be omitted.

4. Use the `ip` command to set the IP address and network mask prefix length of the teamed interface:

   ```
   sudo ip addr add 10.0.0.5/24 dev team0
   ```

For more information, see the `teamd(8)` manual page.

Adding Ports to and Removing Ports from a Team

To add a port to a team, use the `teamdctl` command, for example:

```
sudo teamdctl team0 port add em5
```

To remove a port from a team:

```
sudo teamdctl team0 port remove em5
```

For more information, see the `teamdctl(8)` manual page.

Changing the Configuration of a Port in a Team

You can use the `teamdctl` command to update the configuration of a constituent port of a team, for example:

```
sudo teamdctl team0 port config update em3 '{"prio": -10, "sticky": false}'
```

Enclose the JSON-format definition in single quotes and do not split it over multiple lines.

For more information, see the `teamdctl(8)` manual page.

Removing a Team

To remove a team, use the following command to kill the `teamd` daemon:

```
# teamd -t team0 -k
```

For more information, see the `teamd(8)` manual page.

Displaying Information About Teams

To display the network state of the teamed interface, use the `ip` command:

```
sudo ip addr show dev team0

7: team0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
    link/ether 08:00:27:15:7a:f1 brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.5/24 scope global team0
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe15:7af1/64 scope link
       valid_lft forever preferred_lft forever
```

You can use the `teamnl` command to display information about the component ports of the team:

```
sudo teamnl team0 ports

 5: em4: up 1000Mbit FD
 4: em3: up 1000Mbit FD
```

To display the current state of the team, use the `teamdctl` command, for example:

```
sudo teamdctl team0 state

setup:
  runner: activebackup
ports:
  em3
    link watches:
      link summary: down
      instance[link_watch_0]:
        name: ethtool
        link: down
  em4
    link watches:
      link summary: up
      instance[link_watch_0]:
        name: ethtool
        link: up
runner:
  active port: em4
```

You can also use `teamdctl` to display the JSON configuration of the team and each of its constituent ports:

```
sudo teamdctl team0 config dump

{
    "device": "team0",
```

```
    "link_watch": {
        "name": "ethtool"
    },
    "mcast_rejoin": {
        "count": 1
    },
    "notify_peers": {
        "count": 1
    },
    "ports": {
        "em3": {
            "prio": -10,
            "sticky": true
        },
        "em4": {
            "prio": 100
        }
    },
    "runner": {
        "name": "activebackup"
    }
}
```

For more information, see the `teamdctl(8)` and `teamnl(8)` manual pages.

Configuring VLANs with Untagged Data Frames

A virtual local area network (VLAN) consists of a group of machines that can communicate as if they were attached to the same physical network. A VLAN allows you to 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 ports that you assign with this PVID are in a single broadcast domain. Broadcasts between devices in the same VLAN are not visible to other ports with a different VLAN, even if they exist on the same switch.

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

To create a VLAN device from the command line, enter:

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

This example 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 do not 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.

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 em1-pvid5 ifname em1-pvid5 dev em1 id 5
```

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

Using the ip Command to Create VLAN Devices

The `ip` command provides an alternate method of creating VLAN devices. However, such devices do not persist across system reboots.

To create a VLAN interface `em1.5` for `em1` with a PVID of 5:

```
sudo ip link add link em1 name em1.5 type vlan id 5
```

For more information, see the `ip(8)` manual page.

Configuring Network Routing

A system uses its routing table to determine which network interface to use when sending packets to remote systems. If a system has only a single interface, it is sufficient to configure the IP address of a gateway system on the local network that routes packets to other networks.

To create a default route for IPv4 network packets, include an entry for GATEWAY in the `/etc/sysconfig/network` file. For example, the following entry configures the IP address of the gateway system:

```
GATEWAY=192.0.2.1
```

If your system has more than one network interface, you can specify which interface should be used:

```
GATEWAY=192.0.2.1
GATEWAYDEV=em1
```

A single statement is usually sufficient to define the gateway for IPv6 packets, for example:

```
IPV6_DEFAULTGW="2001:db8:1e10:115b::2%em1"
```

Any changes that you make to `/etc/sysconfig/network` do not take effect until you restart the network service:

```
sudo systemctl restart network
```

To display the routing table, use the `ip route show` command, for example:

```
sudo ip route show

10.0.2.0/24 dev em1  proto kernel  scope link  src 10.0.2.15
default via 10.0.2.2 dev em1  proto static
```

This example shows that packets destined for the local network (10.0.2.0/24) do not use the gateway. The default entry means that any packets destined for addresses outside the local network are routed via the gateway 10.0.2.2.

> **Note:**
>
> You might be used to using the `route` command to configure routing. However, `route` is considered obsolete and will eventually be replaced altogether by the `ip` command.

You can also use the `netstat -rn` command to display this information:

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
10.0.2.0        0.0.0.0         255.255.255.0   U         0 0          0 em1
0.0.0.0         10.0.2.2        0.0.0.0         UG        0 0          0 em1
```

To add or delete a route from the table, use the `ip route add` or `ip route del` commands. For example, to replace the entry for the static default route:

```
sudo ip route del default
sudo ip route show

10.0.2.0/24 dev em1  proto kernel  scope link  src 10.0.2.15

sudo ip ro add default via 10.0.2.1 dev em1 proto static
sudo ip route show

10.0.2.0/24 dev em1  proto kernel  scope link  src 10.0.2.15
default via 10.0.2.1 dev em1  proto static
```

To add a route to the network 10.0.3.0/24 via 10.0.3.1 over interface `em2`, and then delete that route:

```
sudo ip route add 10.0.4.0/24 via 10.0.2.1 dev em2
sudo ip route show

10.0.2.0/24 dev em1  proto kernel  scope link  src 10.0.2.15
10.0.3.0/24 via 10.0.3.1 dev em2
default via 10.0.2.2 dev em1  proto static

sudo ip route del 10.0.3.0/24
sudo ip route show

10.0.2.0/24 dev em1  proto kernel  scope link  src 10.0.2.15
default via 10.0.2.2 dev em1  proto static
```

The `ip route get` command is a useful feature that allows you to query the route on which the system will send packets to reach a specified IP address, for example:

```
sudo ip route get 23.6.118.140

23.6.118.140 via 10.0.2.2 dev em1  src 10.0.2.15
    cache  mtu 1500 advmss 1460 hoplimit 64
```

In this example, packets to 23.6.118.140 are sent out of the `em1` interface via the gateway 10.0.2.2.

Any changes that you make to the routing table using `ip route` do not persist across system reboots. To permanently configure static routes, you can configure them by creating a `route-`*`interface`* file in `/etc/sysconfig/network-scripts` for the interface.

For example, you would configure a static route for the `em1` interface in a file named `route-em1`. An entry in these files can take the same format as the arguments to the `ip route add` command.

For example, to define a default gateway entry for `em1`, create an entry such as the following in `route-em1`:

```
default via 10.0.2.1 dev em1
```

The following entry in `route-em2` would define a route to 10.0.3.0/24 via 10.0.3.1 over `em2`:

```
10.0.3.0/24 via 10.0.3.1 dev em2
```

Any changes that you make to a `route-`*`interface`* file do not take effect until you restart either the network service or the interface.

For more information, see the `ip(8)` and `netstat(8)` manual pages.

2

Configuring Network Addressing

This chapter describes how to configure a DHCP server, DHCP client, and Network Address Translation.

About the Dynamic Host Configuration Protocol

The Dynamic Host Configuration Protocol (DHCP) enables client systems to obtain network configuration information from a DHCP server each time that they connect to the network. The DHCP server is configured with a range of IP addresses and other network configuration parameters that clients need.

When you configure an Oracle Linux system as a DHCP client, the client daemon, `dhclient`, contacts the DHCP server to obtain the networking parameters. As DHCP is broadcast-based, the client must be on the same subnet as either a server or a relay agent. If a client cannot be on the same subnet as the server, a DHCP relay agent can be used to pass DHCP messages between subnets.

The server provides a lease for the IP address that it assigns to a client. The client can request specific terms for the lease, such as the duration. You can configure a DHCP server to limit the terms that it can grant for a lease. Provided that a client remains connected to the network, `dhclient` automatically renews the lease before it expires. You can configure the DHCP server to provide the same IP address to a client based on the MAC address of its network interface.

The advantages of using DHCP include:

- centralized management of IP addresses
- ease of adding new clients to a network
- reuse of IP addresses reducing the total number of IP addresses that are required
- simple reconfiguration of the IP address space on the DHCP server without needing to reconfigure each client

For more information about DHCP, see RFC 2131.

Configuring a DHCP Server

To configure an Oracle Linux system as a DHCP server:

1. Install the `dhcp` package:

   ```
   sudo yum install dhcp
   ```

2. Edit the `/etc/dhcp/dhcpd.conf` file to store the settings that the DHCP server can provide to the clients.

   The following example configures the domain name, a range of client addresses on the 192.168.2.0/24 subnet from 192.168.2.101 through 192.168.2.254 together with the IP addresses of the default gateway and the DNS server, the default and maximum lease

times in seconds, and a static IP address for the application server `svr01` that is identified by its MAC address:

```
option domain-name "mydom.org";
option domain-name-servers 192.168.2.1, 10.0.1.4;
option broadcast-address 192.168.2.255;
option routers 192.168.2.1;

subnet 192.168.2.0 netmask 255.255.255.0 {
  range 192.168.2.101 192.168.2.254;
  default-lease-time 10800;
  max-lease-time 43200;
}

host svr01 {
  hardware ethernet 80:56:3e:00:10:00;
  fixed-address 192.168.2.100;
  max-lease-time 86400;
}
```

The DHCP server sends the information in the `option` lines to each client when it requests a lease on an IP address. An option applies only to a subnet if you define it inside a `subnet` definition. In the example, the options are global and apply to both the `subnet` and `host` definitions. The `subnet` and `host` definitions have different settings for the maximum lease time.

> **Note:**
>
> In Oracle Linux 7, the DHCP server no longer reads its configuration from `/etc/sysconfig/dhcpd`. Instead, it reads `/etc/dhcp/dhcpd.conf` to determine the interfaces on which it should listen.

For more information and examples, see `/usr/share/doc/dhcp-`*`version`*`/dhcpd.conf.sample` and the `dhcpd(8)` and `dhcp-options(5)` manual pages.

3. Touch the `/var/lib/dhcpd/dhcpd.leases` file, which stores information about client leases:

   ```
   sudo touch /var/lib/dhcpd/dhcpd.leases
   ```

4. Enter the following commands to start the DHCP service and ensure that it starts after a reboot:

   ```
   sudo systemctl start dhcpd
   sudo systemctl enable dhcpd
   ```

For information about configuring a DHCP relay, see the `dhcrelay(8)` manual page.

Configuring a DHCP Client

To configure an Oracle Linux system as a DHCP client:

1. Install the `dhclient` package:

   ```
   sudo yum install dhclient
   ```

2. Edit `/container/name/rootfs/etc/sysconfig/network-scripts/ifcfg-iface` , where *iface* is the name of the network interface, and change the value of `BOOTPROTO` to read as:

   ```
   BOOTPROTO=dhcp
   ```

3. Edit `/etc/sysconfig/network` and verify that it contains the following setting:

   ```
   NETWORKING=yes
   ```

4. To specify options for the client, such as the requested lease time and the network interface on which to request an address from the server, create the file `/etc/dhclient.conf` containing the required options.

   The following example specifies that the client should use the `em2` interface, request a lease time of 24 hours, and identify itself using its MAC address:

   ```
   interface "em2" {
     send dhcp-lease-time 86400;
     send dhcp-client-identifier 80:56:3e:00:10:00;
   }
   ```

   For more information, see the `dhclient.conf(5)` manual page.

5. Restart the network interface or the network service to enable the client, for example:

   ```
   sudo systemctl restart network
   ```

   When the client has requested and obtained a lease, information about this lease is stored in `/var/lib/dhclient/dhclient-interface.leases`.

For more information, see the `dhclient(8)` manual page.

About Network Address Translation

Network Address Translation (NAT) assigns a public address to a computer or a group of computers inside a private network with a different address scheme. The public IP address masquerades all requests as going to one server rather than several servers. NAT is useful for limiting the number of public IP addresses that an organization must finance, and for providing 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.

> **Note:**
>
> If your want a system to be able to route packets between two of its network interfaces, you must turn on IP forwarding:
>
> ```
> echo 1 > /proc/sys/net/ipv4/ip_forward
> ```

You can use the Firewall Configuration GUI (`firewall-config`) to configure masquerading and port forwarding.

3
Configuring the Name Service

This chapter describes how to use BIND to set up a DNS name server.

About DNS and BIND

The Domain Name System (DNS) is a network-based service that maps (*resolves*) domain names to IP addresses. For a small, isolated network, you could use entries in the /etc/hosts file to provide the mapping, but most networks that are connected to the Internet use DNS.

DNS is a hierarchical and distributed database, where each level of the hierarchy is delimited by a period (`.`). Consider the following fully qualified domain name (FQDN):

```
wiki.us.mydom.com.
```

The root domain, represented by the final period in the FQDN, is usually omitted, except in DNS configuration files:

```
wiki.us.mydom.com
```

In this example, the top-level domain is `com`, `mydom` is a subdomain of `com`, `us` is a subdomain of `mydom`, and `wiki` is the host name. Each of these domains are grouped into zones for administrative purposes. A DNS server, or *name server*, stores the information that is needed to resolve the component domains inside a zone. In addition, a zone's DNS server stores pointers to the DNS servers that are responsible for resolving each subdomain.

If a client outside the `us.mydom.com` domain requests that its local name server resolve a FQDN such as `wiki.us.mydom.com` into an IP address for which the name server is not authoritative, the name server queries a root name server for the address of a name server that is authoritative for the `com` domain. Querying this name server returns the IP address of a name server for `mydom.com`. In turn, querying this name server returns the IP address of the name server for `us.oracle.com`, and querying this final name server returns the IP address for the FQDN. This process is known as a recursive query, where the local name server handles each referral from an external name server to another name server on behalf of the resolver.

Iterative queries rely on the resolver being able to handle the referral from each external name server to trace the name server that is authoritative for the FQDN. Most resolvers use recursive queries and so cannot use name servers that support only iterative queries. Fortunately, most

Oracle Linux provides the Berkeley Internet Name Domain (BIND) implementation of DNS. The `bind` package includes the DNS server daemon (`named`), tools for working with DNS such as `rndc`, and a number of configuration files, including the following:

`/etc/named.conf`
Contains settings for `named` and lists the location and characteristics of the zone files for your domain. Zone files are usually stored in `/var/named`.

/etc/named.rfc1912.zones
Contains several zone sections for resolving local loopback names and addresses.

/var/named/named.ca
Contains a list of the root authoritative DNS servers.

About Types of Name Servers

You can configure several types of name server using BIND, including:

Master name server
Authoritative for one or more domains, a primary (master) name server maintains its zone data in several database files, and can transfer this information periodically to any backup (slave) name servers that are also configured in the zone. An organization might maintain two primary name servers for a zone: one primary server outside the firewall to provide restricted information about the zone for publicly accessible hosts and services, and a hidden or *stealth* primary server inside the firewall that holds details of internal hosts and services.

Slave name server
Acting as a backup to a primary name server, a backup name server maintains a copy of the zone data, which it periodically refreshes from the primary server's copy.

Stub name server
A primary name server for a zone might also be configured as a stub name server that maintains information about the primary and backup name servers of child zones.

Caching-only name server
Performs queries on behalf of a client and stores the responses in a cache after returning the results to the client. It is not authoritative for any domains and the information that it records is limited to the results of queries that it has cached.

Forwarding name server
Forwards all queries to another name server and caches the results, which reduces local processing, external access, and network traffic.

In practice, a name server can be a combination of several of these types in complex configurations.

About DNS Configuration Files

Domains are grouped into zones and zones are configured through the use of zone files. Zone files store information about domains in the DNS database. Each zone file contains directives and resource records. Optional directives apply settings to a zone or instruct a name server to perform certain tasks. Resource records specify zone parameters and define information about the systems (*hosts*) in a zone.

For examples of BIND configuration files, see `/usr/share/doc/bind-version/sample/` .

/etc/named.conf

The main configuration file for `named` is `/etc/named.conf`, which contains settings for `named` and the top-level definitions for zones, for example:

```
include "/etc/rndc.key";

controls {
    inet 127.0.0.1 allow { localhost; } keys { "rndc-key"; }
};

zone "us.mydom.com" {
    type master;
    file "master-data";
    allow-update { key "rndc-key"; };
    notify yes;
};

zone "mydom.com" IN {
    type slave;
    file "sec/slave-data";
    allow-update { key "rndc-key"; };
    masters {10.1.32.1;};
};

zone "2.168.192.in-addr.arpa" IN {
    type master;
    file "reverse-192.168.2";
    allow-update { key "rndc-key"; };
    notify yes;
};
```

The `include` statement allows external files to be referenced so that potentially sensitive data such as key hashes can be placed in a separate file with restricted permissions.

The `controls` statement defines access information and the security requirements that are necessary to use the `rndc` command with the `named` server:

`inet`
Specifies which hosts can run `rndc` to control named. In this example, `rndc` must be run on the local host (127.0.0.1).

`keys`
Specifies the names of the keys that can be used. The example specifies using the key named `rndc-key`, which is defined in `/etc/rndc.key`. Keys authenticate various actions by `named` and are the primary method of controlling remote access and administration.

The `zone` statements define the role of the server in different zones.

The following zone options are used:

`type`
Specifies that this system is the primary name server for the zone `us.mydom.com` and a backup server for `mydom.com`. `2.168.192.in-addr.arpa` is a reverse zone for resolving IP addresses to host names. See About Resource Records for Reverse-name Resolution.

`file`
Specifies the path to the zone file relative to `/var/named`. The zone file for `us.mydom.com` is stored in `/var/named/master-data` and the transferred zone data for `mydom.com` is cached in `/var/named/sec/slave-data`.

allow-update
Specifies that a shared key must exist on both the primary and backup name servers for a zone transfer to take place from the primary server to the backup. The following is an example record for a key in `/etc/rndc.key`:

```
key "rndc-key" {
    algorithm hmac-md5;
    secret "XQX8NmM41+RfbbSdcqOejg==";
};
```

You can use the `rndc-confgen -a` command to generate a key file.

notify
Specifies whether to notify the backup name servers when the zone information is updated.

masters
Specifies the primary name server for a backup name server.

The next example is taken from the default `/etc/named.conf` file that is installed with the `bind` package, and which configures a caching-only name server.

```
options {
    listen-on port 53 { 127.0.0.1; };
    listen-on-v6 port 53 { ::1; };
    directory       "/var/named";
    dump-file       "/var/named/data/cache_dump.db";
    statistics-file "/var/named/data/named_stats.txt";
    memstatistics-file "/var/named/data/named_mem_stats.txt";
    allow-query { localnets; };
    recursion yes;

    dnssec-enable yes;
    dnssec-validation yes;
    dnssec-lookaside auto;

    /* Path to ISC DLV key */
    bindkeys-file "/etc/named.iscdlv.key";

    managed-keys-directory "/var/named/dynamic";
};

logging {
    channel default_debug {
        file "data/named.run";
        severity dynamic;
    };
};

zone "." IN {
    type hint;
    file "named.ca";
};

include "/etc/named.rfc1912.zones";
include "/etc/named.root.key";
```

The `options` statement defines global server configuration options and sets defaults for other statements.

listen-on
The port on which `named` listens for queries.

directory
Specifies the default directory for zone files if a relative pathname is specified.

dump-file
Specifies where `named` dumps its cache if it crashes.

statistics-file
Specifies the output file for the `rndc stats` command.

memstatistics-file
Specifies the output file for `named` memory-usage statistics.

allow-query
Specifies which IP addresses may query the server. `localnets` specifies all locally attached networks.

recursion
Specifies whether the name server performs recursive queries.

dnssec-enable
Specifies whether to use secure DNS (DNSSEC).

dnssec-validation
Whether the name server should validate replies from DNSSEC-enabled zones.

dnssec-lookaside
Whether to enable DNSSEC Lookaside Validation (DLV) using the key in `/etc/named.iscdlv.key` defined by `bindkeys-file`.

The `logging` section enables logging of messages to `/var/named/data/named.run`. The `severity` parameter controls the logging level, and the `dynamic` value means that this level can be controlled by using the `rndc trace` command.

The `zone` section specifies the initial set of root servers using a hint zone. This zone specifies that `named` should consult `/var/named/named.ca` for the IP addresses of authoritative servers for the root domain (`.`).

For more information, see the `named.conf(5)` manual page and the BIND documentation in `/usr/share/doc/bind-`*`version`*`/arm`.

About Resource Records in Zone Files

A resource record in a zone file contains the following fields, some of which are optional depending on the record type:

Name
Domain name or IP address.

TTL (time to live)
The maximum time that a name server caches a record before it checks whether a newer one is available.

Class
Always `IN` for Internet.

Type
Type of record, for example:

`A` (address)
IPv4 address corresponding to a host.

`AAAA` (address)
IPv6 address corresponding to a host.

`CNAME` (canonical name)
Alias name corresponding to a host name.

`MX` (mail exchange)
Destination for email addressed to the domain.

`NS` (name server)
Fully qualified domain name of an authoritative name server for a domain.

`PTR` (pointer)
Host name corresponding to an IP address for address to name lookups (reverse-name resolution).

`SOA` (start of authority)
Authoritative information about a zone, such as the primary name server, the email address of the domain's administrator, and the domain's serial number. All records following a `SOA` record relate to the zone that it defines up to the next `SOA` record.

Data
The information that the record stores, such as an IP address in an `A` record, or a host name in a `CNAME` or `PTR` record.

The following example shows the contents of a typical zone file such as `/var/named/master-data`:

```
$TTL 86400        ; 1 day
@ IN SOA dns.us.mydom.com. root.us.mydom.com. (
            57 ; serial
            28800 ; refresh (8 hours)
            7200 ; retry (2 hours)
            2419200 ; expire (4 weeks)
            86400 ; minimum (1 day)
            )
              IN  NS      dns.us.mydom.com.

dns           IN  A       192.168.2.1
us.mydom.com  IN  A       192.168.2.1
svr01         IN  A       192.168.2.2
www           IN  CNAME   svr01
host01        IN  A       192.168.2.101
host02        IN  A       192.168.2.102
host03        IN  A       192.168.2.103
...
```

A comment on a line is preceded by a semicolon (`;`).

The `$TTL` directive defines the default time-to-live value for all resource records in the zone. Each resource record can define its own time-to-live value, which overrides the global setting.

The `SOA` record is mandatory and included the following information:

`us.mydom.com`
The name of the domain.

`dns.us.mydom.com.`
The fully qualified domain name of the name server, including a trailing period (`.`) for the root domain.

`root.us.mydom.com.`
The email address of the domain administrator.

serial
A counter that, if incremented, tells `named` to reload the zone file.

refresh
The time after which a primary name server notifies backup name servers that they should refresh their database.

retry
If a refresh fails, the time that a backup name server should wait before attempting another refresh.

expire
The maximum elapsed time that a backup name server has to complete a refresh before its zone records are no longer considered authoritative and it will stop answering queries.

minimum
The minimum time for which other servers should cache information obtained from this zone.

An `NS` record declares an authoritative name server for the domain.

Each `A` record specifies the IP address that corresponds to a host name in the domain.

The `CNAME` record creates the alias `www` for `svr01`.

For more information, see the BIND documentation in `/usr/share/doc/bind-`*`version`*`/arm`.

About Resource Records for Reverse-name Resolution

Forward resolution returns an IP address for a specified domain name. Reverse-name resolution returns a domain name for a specified IP address. DNS implements reverse-name resolution by using the special `in-addr.arpa` and `ip6.arpa` domains for IPv4 and IPv6.

The characteristics for a zone's `in-addr.arpa` or `ip6.arpa` domains are usually defined in `/etc/named.conf`, for example:

```
zone "2.168.192.in-addr.arpa" IN {
    type master;
    file "reverse-192.168.2";
    allow-update { key "rndc-key"; };
    notify yes;
};
```

The zone's name consists of `in-addr.arpa` preceded by the network portion of the IP address for the domain with its dotted quads written in reverse order.

If your network does not have a prefix length that is a multiple of 8, see RFC 2317 for the format that you should use instead.

The `PTR` records in `in-addr.arpa` or `ip6.arpa` domains define host names that correspond to the host portion of the IP address. The following example is taken from the `/var/named/reverse-192.168.2` zone file:

```
$TTL 86400        ;
@ IN SOA dns.us.mydom.com. root.us.mydom.com. (
            57 ;
            28800 ;
            7200 ;
            2419200 ;
            86400 ;
            )
              IN  NS      dns.us.mydom.com.

1             IN  PTR     dns.us.mydom.com.
1             IN  PTR     us.mydom.com.
2             IN  PTR     svr01.us.mydom.com.
101           IN  PTR     host01.us.mydom.com.
102           IN  PTR     host02.us.mydom.com.
103           IN  PTR     host03.us.mydom.com.
...
```

For more information, see the BIND documentation in `/usr/share/doc/bind-`*`version`*`/arm`.

Configuring a Name Server

By default, the BIND installation allows you to configure a caching-only name server using the configuration settings that are provided in `/etc/named.conf` and files that it includes. This procedure assumes that you will either use the default settings or configure new `named` configuration and zone files.

To configure a name server:

1. Install the bind package:

   ```
   sudo yum install bind
   ```

2. If `NetworkManager` is enabled on the system, edit the `/etc/sysconfig/network-scripts/ifcfg-`*`interface`* file, and add the following entry:

   ```
   DNS1=127.0.0.1
   ```

   This line causes `NetworkManager` to add the following entry to `/etc/resolv.conf` when the network service starts:

   ```
   nameserver 127.0.0.1
   ```

   This entry points the resolver at the local name server.

   If you have disabled `NetworkManager`, edit `/etc/resolv.conf` to include the `nameserver 127.0.0.1` entry.

3. If required, modify the `named` configuration and zone files.

4. Configure the system firewall to allow incoming TCP connections to port 53 and incoming UDP datagrams on port 53:

```
sudo firewall-cmd --zone=zone --add-port=53/tcp --add-port=53/udp
sudo firewall-cmd --permanent --zone=zone --add-port=53/tcp --add-port=53/udp
```

5. Restart the `network` service, restart the `named` service, and configure `named` to start following system reboots:

```
sudo systemctl restart network
sudo systemctl start named
sudo systemctl enable named
```

Administering the Name Service

The `rndc` command allows you to administer the `named` service, either locally or from a remote machine (if permitted in the `controls` section of the `/etc/named.conf` file). To prevent unauthorized access to the service, `rndc` must be configured to listen on the selected port (by default, port 953), and both named and `rndc` must have access to the same key. To generate a suitable key, use the `rndc-confgen` command:

```
# rndc-confgen -a
wrote key file "/etc/rndc.key"
```

To ensure that only `root` can read the file:

```
# chmod o-rwx /etc/rndc.key
```

To check the status of the `named` service:

```
# rndc status
number of zones: 3
debug level: 0
xfers running: 0
xfers deferred: 0
soa queries in progress: 0
query logging is OFF
recursive clients: 0/1000
tcp clients: 0/100
server is up and running
```

If you modify the `named` configuration file or zone files, `rndc reload` instructs `named` to reload the files:

```
# rndc reload
server reload successful
```

For more information, see the `named(8)`, `rndc(8)` and `rndc-confgen(8)` manual pages.

Performing DNS Lookups

The `host` utility is recommended for performing DNS lookups. Without any arguments, `host` displays a summary of its command-line arguments and options. For example, look up the IP address for `host01`:

```
host host01
```

Perform a reverse lookup for the domain name that corresponds to an IP address:

```
host 192.168.2.101
```

Query DNS for the IP address that corresponds to a domain:

```
host dns.us.mydoc.com
```

Use the `-v` and `-t` options to display verbose information about records of a certain type:

```
host -v -t MX  www.mydom.com

Trying "www.mydom.com"
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 49643
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;www.mydom.com.            IN    MX

;; ANSWER SECTION:
www.mydom.com.        135    IN    CNAME    www.mydom.com.acme.net.
www.mydom.com.acme.net. 1240 IN    CNAME    d4077.c.miscacme.net.

;; AUTHORITY SECTION:
c.miscacme.net.    2000    IN    SOA    m0e.miscacme.net.
hostmaster.misc.com. ...

Received 163 bytes from 10.0.0.1#53 in 40 ms
```

The `-a` option (equivalent to `-v -t ANY`) displays all available records for a zone:

```
host -a www.us.mydom.com

Trying "www.us.mydom.com"
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 40030
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;www.us.mydom.com.            IN    ANY

;; ANSWER SECTION:
www.us.mydom.com.        263    IN    CNAME    www.us.mydom.acme.net.

Received 72 bytes from 10.0.0.1#53 in 32 ms
```

For more information, see the `host(1)` manual page.

4
Configuring Network Time

This chapter describes how to configure a system to use the chrony, Network Time Protocol (NTP), or Precision Time Protocol (PTP) daemons for setting the system time.

About the chronyd Daemon

The `chrony` package provides a `chronyd` service daemon and `chronyc` utility that enable mobile systems and virtual machines to update their system clock after a period of suspension or disconnection from a network.

The `chronyd` service is primarily designed to allow mobile systems and virtual machines to update their system clock after a period of suspension or disconnection from a network. However, you can also use it to implement a simple NTP client or a NTP server. When used as an NTP server, `chronyd` can synchronise with higher stratum NTP servers or it can act as a stratum 1 server using time signals received from the Global Positioning System (GPS) or radio broadcasts such as DCF77, MSF, or WWVB.

You can use the `chronyc` command to manage the `chronyd` service.

> **Note:**
>
> `chronyd` uses NTP version 3 (RFC 1305), whose features are compatible with NTP version 4 (RFC 5905). However, `chronyd` does not support several important features of NTP version 4 nor does it support the use of PTP.

Configuring the chronyd Service

To configure the `chronyd` service on a system:

1. Install the `chrony` package.

   ```
   sudo yum install chrony
   ```

2. Edit `/etc/chrony.conf` to set up the configuration for `chronyd`.

   > **Note:**
   >
   > The default configuration assumes that the system has network access to public NTP servers with which it can synchronise. The firewall rules for your internal networks might well prevent access to these servers but instead allow access to local NTP servers.

   The following example shows a sample configuration for a system that can access three NTP servers:

```
server NTP_server_1
server NTP_server_2
server NTP_server_3
driftfile /var/lib/chrony/drift
keyfile /etc/chrony.keys
commandkey 1
generatecommandkey
```

The `commandkey` directive specifies the `keyfile` entry that `chronyd` uses to authenticate both `chronyc` commands and NTP packets. The `generatecommandkey` directive causes `chronyd` to generate an SHA1-based password automatically when the service starts.

To configure `chronyd` to act as an NTP server for a specified client or subnet, use the `allow` directive, for example:

```
server NTP_server_1
server NTP_server_2
server NTP_server_3
allow 192.168.2/24
driftfile /var/lib/chrony/drift
keyfile /etc/chrony.keys
commandkey 1
generatecommandkey
```

If a system has only intermittent access to NTP servers, the following configuration might be appropriate:

```
server NTP_server_1 offline
server NTP_server_2 offline
server NTP_server_3 offline
driftfile /var/lib/chrony/drift
keyfile /etc/chrony.keys
commandkey 1
generatecommandkey
```

If you specify the `offline` keyword, `chronyd` does not poll the NTP servers until it is told that network access is available. You can use the `chronyc -a online` and `chronyc -a offline` command to inform `chronyd` of the state of network access.

3. If remote access to the local NTP service is required, configure the system firewall to allow access to the NTP service in the appropriate zones, for example:

```
sudo firewall-cmd --zone=zone --add-service=ntp

success

sudo firewall-cmd --zone=zone --permanent --add-service=ntp

success
```

4. Start the `chronyd` service and configure it to start following a system reboot.

```
sudo systemctl start chronyd
sudo systemctl enable chronyd
```

You can use the `chronyc` command to display information about the operation of `chronyd` or to change its configuration, for example:

```
sudo chronyc -a
```

```
chrony version version
...
200 OK

chronyc> sources

210 Number of sources = 4
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
^+ service1-eth3.debrecen.hp     2   6    37    21  -2117us[-2302us] +/-   50ms
^* ns2.telecom.lt                2   6    37    21   -811us[ -997us] +/-   40ms
^+ strato-ssd.vpn0.de            2   6    37    21   +408us[ +223us] +/-   78ms
^+ kvm1.websters-computers.c     2   6    37    22  +2139us[+1956us] +/-   54ms

chronyc> sourcestats

210 Number of sources = 4
Name/IP Address            NP  NR  Span  Frequency  Freq Skew  Offset  Std Dev
==============================================================================
service1-eth3.debrecen.hp   5   4   259     -0.394     41.803  -2706us   502us
ns2.telecom.lt              5   4   260     -3.948     61.422   +822us   813us
strato-ssd.vpn0.de          5   3   259      1.609     68.932   -581us   801us
kvm1.websters-computers.c   5   5   258     -0.263      9.586  +2008us   118us

Reference ID    : 212.59.0.2 (ns2.telecom.lt)
Stratum         : 3
Ref time (UTC)  : Tue Sep 30 12:33:16 2014
System time     : 0.000354079 seconds slow of NTP time
Last offset     : -0.000186183 seconds
RMS offset      : 0.000186183 seconds
Frequency       : 28.734 ppm slow
Residual freq   : -0.489 ppm
Skew            : 11.013 ppm
Root delay      : 0.065965 seconds
Root dispersion : 0.007010 seconds
Update interval : 64.4 seconds
Leap status     : Normal
chronyc> exit

chronyc> tracking

Reference ID    : 212.59.0.2 (ns2.telecom.lt)
Stratum         : 3
Ref time (UTC)  : Tue Sep 30 12:33:16 2014
System time     : 0.000354079 seconds slow of NTP time
Last offset     : -0.000186183 seconds
RMS offset      : 0.000186183 seconds
Frequency       : 28.734 ppm slow
Residual freq   : -0.489 ppm
Skew            : 11.013 ppm
Root delay      : 0.065965 seconds
Root dispersion : 0.007010 seconds
Update interval : 64.4 seconds
Leap status     : Normal

chronyc> exit
```

Using the `-a` option to `chronyc` is equivalent to entering the `authhash` and `password` subcommands, and avoids you having to specify the hash type and password every time that you use `chronyc`:

```
sudo cat /etc/chrony.keys

1 SHA1 HEX:4701E4D70E44B8D0736C8A862CFB6B8919FE340E
# chronyc
...
chronyc> authhash SHA1
chronyc> password HEX:4701E4D70E44B8D0736C8A862CFB6B8919FE340E
200 OK
```

For more information, see the `chrony(1)` and `chronyc(1)` manual pages, `/usr/share/doc/chrony-`*`version`*`/chrony.txt`, or use the `info chrony` command.

About the NTP Daemon

The `ntpd` daemon can synchronise the system clock with remote NTP servers, with local reference clocks, or with GPS and radio time signals. `ntpd` provides a complete implementation of NTP version 4 (RFC 5905) and is also compatibility with versions 3 (RFC 1305), 2 (RFC 1119), and 1 (RFC 1059).

You can configure `ntpd` to run in several different modes, as described at http://doc.ntp.org/4.2.6p5/assoc.html, using both symmetric-key and public-key cryptography, as described at http://doc.ntp.org/4.2.6p5/authopt.html.

Configuring the ntpd Service

To configure the `ntpd` service on a system:

1. Install the `ntp` package.

   ```
   sudo yum install ntp
   ```

2. Edit `/etc/ntp.conf` to set up the configuration for `ntpd`.

   > **Note:**
   >
   > The default configuration assumes that the system has network access to public NTP servers with which it can synchronise. The firewall rules for your internal networks might well prevent access to these servers but instead allow access to local NTP servers.

   The following example shows a sample NTP configuration for a system that can access three NTP servers:

   ```
   server NTP_server_1
   server NTP_server_2
   server NTP_server_3
   server  127.127.1.0
   fudge   127.127.1.0 stratum 10
   driftfile /var/lib/ntp/drift
   restrict default nomodify notrap nopeer noquery
   ```

   The `server` and `fudge` entries for 127.127.1.0 cause `ntpd` to use the local system clock if the remote NTP servers are not available. The `restrict` entry allows remote systems only to synchronise their time with the local NTP service.

For more information about configuring `ntpd`, see http://doc.ntp.org/4.2.6p5/manyopt.html.

3. Create the drift file.

```
touch /var/lib/ntp/drift
```

4. If remote access to the local NTP service is required, configure the system firewall to allow access to the NTP service in the appropriate zones, for example:

```
sudo firewall-cmd --zone=zone --add-service=ntp

success

sudo firewall-cmd --zone=zone --permanent --add-service=ntp

success
```

5. Start the `ntpd` service and configure it to start following a system reboot.

```
sudo systemctl start ntpd
sudo systemctl enable ntpd
```

You can use the `ntpq` and `ntpstat` commands to display information about the operation of `ntpd`, for example:

```
sudo ntpq -p

     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
*ns1.proserve.nl 193.67.79.202    2 u   21   64  377   31.420   10.742   3.689
-pomaz.hu        84.2.46.19       3 u   22   64  377   59.133   13.719   5.958
+server.104media 193.67.79.202    2 u   24   64  377   32.110   13.436   5.222
+public-timehost 193.11.166.20    2 u   28   64  377   57.214    9.304   6.311

sudo ntpstat

synchronised to NTP server (80.84.224.85) at stratum 3
   time correct to within 76 ms
   polling server every 64
```

For more information, see the `ntpd(8)`, `ntpd.conf(5)`, `ntpq(8)`, and `ntpstat(8)` manual pages and http://doc.ntp.org/4.2.6p5/.

About PTP

PTP allows you to synchronise system clocks on a local area network to a higher accuracy than NTP. Provided that network drivers support either hardware or software time stamping, a PTP clock can use the time stamps in PTP messages to compensate for propagation delays across a network. Software time stamping allows PTP to synchronise systems to within a few tens of microseconds. With hardware time stamping, PTP can synchronise systems to within a few tenths of a microsecond. If you require high-precision time synchronization of systems, use hardware time stamping. Both the UEK R3 and RHCK kernels support PTP version 2 as defined in IEEE 1588.

A typical PTP configuration on an enterprise local area network consists of:

- One or more *grandmaster clock* systems.

  A grandmaster clock is typically implemented as specialized hardware that can use high-accuracy GPS signals or lower-accuracy code division multiple access (CDMA) signals,

radio clock signals, or NTP as a time reference source. If several grandmaster clocks are available, the best master clock (BMC) algorithm selects the grandmaster clock based on the settings of their `priority1`, `clockClass`, `clockAccuracy`, `offsetScaledLogVariance`, and `priority2` parameters and their unique identifier, in that order.

- Several *boundary clock* systems.

  Each boundary clock is backed up to a grandmaster clock on one subnetwork and relays PTP messages to one or more additional subnetworks. A boundary clock is usually implemented as a function of a network switch.

- Multiple *slave clock* systems.

  Each slave clock on a subnetwork is backed up to a boundary clock, which acts as the *master clock* for that slave clock.

A simpler configuration is to set up a single grandmaster clock and multiple slave clocks on the same network segment, which removes any need for an intermediate layer of boundary clocks.

Grandmaster and slave clock systems that use only one network interface for PTP are termed *ordinary clocks*.

Boundary clocks require at least two network interfaces for PTP: one interface acts a slave to a grandmaster clock or a higher-level boundary clock; the other interfaces act as masters to slave clocks or lower-level boundary clocks.

Synchronization of boundary and slave clock systems is achieved by sending time stamps in PTP messages. By default, PTP messages are sent in UDPv4 datagrams. It is also possible to configure PTP to use UDPv6 datagrams or Ethernet frames as its transport mechanism.

To be able to use PTP with a system, the driver for at least one of the system's network interfaces must support either software or hardware time stamping. To find out whether the driver for a network interface supports time stamping, use the `ethtool` command as shown in the following example:

```
sudo ethtool -T em1

Time stamping parameters for em1:
Capabilities:
    hardware-transmit     (SOF_TIMESTAMPING_TX_HARDWARE)
    software-transmit     (SOF_TIMESTAMPING_TX_SOFTWARE)
    hardware-receive      (SOF_TIMESTAMPING_RX_HARDWARE)
    software-receive      (SOF_TIMESTAMPING_RX_SOFTWARE)
    software-system-clock (SOF_TIMESTAMPING_SOFTWARE)
    hardware-raw-clock    (SOF_TIMESTAMPING_RAW_HARDWARE)
...
```

The output from `ethtool` in this example shows that the `em1` interface supports both hardware and software time stamping capabilities.

With software time stamping, `ptp4l` synchronises the system clock to an external grandmaster clock.

If hardware time stamping is available, `ptp4l` can synchronise the PTP hardware clock to an external grandmaster clock. In this case, you use the `phc2sys` daemon to synchronise the system clock with the PTP hardware clock.

Configuring the PTP Service

To configure the PTP service on a system:

1. Install the `linuxptp` package.

   ```
   sudo yum install linuxptp
   ```

2. Edit `/etc/sysconfig/ptp4l` and define the start-up options for the `ptp4l` daemon.

   Grandmaster clocks and slave clocks require that you define only one interface.

   For example, to use hardware time stamping with interface `em1` on a slave clock:

   ```
   OPTIONS="-f /etc/ptp4l.conf -i em1 -s"
   ```

   To use software time stamping instead of hardware time stamping, specify the `-S` option:

   ```
   OPTIONS="-f /etc/ptp4l.conf -i em1 -S -s"
   ```

   > **Note:**
   >
   > The `-s` option specifies that the clock operates only as a slave (`slaveOnly` mode). Do not specify this option for a grandmaster clock or a boundary clock.

   For a grandmaster clock, omit the `-s` option, for example:

   ```
   OPTIONS="-f /etc/ptp4l.conf -i em1"
   ```

   A boundary clock requires that you define at least two interfaces, for example:

   ```
   OPTIONS="-f /etc/ptp4l.conf -i em1 -i em2"
   ```

   You might need to edit the file `/etc/ptp4l.conf` to make further adjustments to the configuration of `ptp4l`, for example:

   - For a grandmaster clock, set the value of the `priority1` parameter to a value between 0 and 127, where lower values have higher priority when the BMC algorithm selects the grandmaster clock. For a configuration that has a single grandmaster clock, a value of 127 is suggested.
   - If you set the value of `summary_interval` to an integer value *N* instead of 0, `ptp4l` writes summary clock statistics to `/var/log/messages` every 2^N seconds instead of every second ($2^0 = 1$). For example, a value of 10 would correspond to an interval of 2^{10} or 1024 seconds.
   - The `logging_level` parameter controls the amount of logging information that `ptp4l` records. The default value of `logging_level` is 6, which corresponds to `LOG_INFO`. To turn off logging completely, set the value of `logging_level` to 0. Alternatively, specify the `-q` option to `ptp4l`.

   For more information, see the `ptp4l(8)` manual page.

3. Configure the system firewall to allow access by PTP event and general messages to UDP ports 319 and 320 in the appropriate zone, for example:

   ```
   sudo firewall-cmd --zone=zone --add-port=319/udp --add-port=320/udp
   ```

```
success

sudo firewall-cmd --permanent --zone=zone --add-port=319/udp --add-
port=320/udp

success
```

4. Start the `ptp4l` service and configure it to start following a system reboot.

```
sudo systemctl start ptp4l
sudo systemctl enable ptp4l
```

5. To configure `phc2sys` on a clock system that uses hardware time stamping:

   a. Edit `/etc/sysconfig/phc2sys` and define the start-up options for the `phc2sys` daemon.

      On a boundary clock or slave clock, synchronise the system clock with the PTP hardware clock that is associated with the slave network interface, for example:

      ```
      OPTIONS="-c CLOCK_REALTIME -s em1 -w"
      ```

      > **Note:**
      >
      > The slave network interface on a boundary clock is the one that it uses to communicate with the grandmaster clock.

      The `-w` option specifies that `phc2sys` waits until `ptp4l` has synchronised the PTP hardware clock before attempting to synchronise the system clock.

      On a grandmaster clock, which derives its system time from a reference time source such as GPS, CDMA, NTP, or a radio time signal, synchronise the network interface's PTP hardware clock from the system clock, for example:

      ```
      OPTIONS="-c em1 -s CLOCK_REALTIME -w"
      ```

      For more information, see the `phc2sys(8)` manual page.

   b. Start the `phc2sys` service and configure it to start following a system reboot.

      ```
      sudo systemctl start phc2sys
      sudo systemctl enable phc2sys
      ```

You can use the `pmc` command to query the status of `ptp4l` operation. The following example shows the results of running `pmc` on a slave clock system that is directly connected to the grandmaster clock system without any intermediate boundary clocks:

```
sudo pmc -u -b 0 'GET TIME_STATUS_NP'

sending: GET TIME_STATUS_NP
    080027.fffe.7f327b-0 seq 0 RESPONSE MANAGEMENT TIME_STATUS_NP
        master_offset              -98434
        ingress_time               1412169090025854874
        cumulativeScaledRateOffset +1.000000000
        scaledLastGmPhaseChange    0
        gmTimeBaseIndicator        0
        lastGmPhaseChange          0x0000'0000000000000000.0000
        gmPresent                  true
        gmIdentity                 080027.fffe.d9e453
```

```
sudo pmc -u -b 0 'GET CURRENT_DATA_SET'


sending: GET CURRENT_DATA_SET
    080027.fffe.7f327b-0 seq 0 RESPONSE MANAGEMENT CURRENT_DATA_SET
        stepsRemoved     1
        offsetFromMaster  42787.0
        meanPathDelay    289207.0
```

Useful information in this output includes:

`gmIdentity`
The unique identifier of the grandmaster clock, which is based on the MAC address of its network interface.

`gmPresent`
Whether an external grandmaster clock is available. This value is displayed as `false` on the grandmaster clock itself.

`meanPathDelay`
An estimate of how many nanoseconds by which synchronization messages are delayed.

`offsetFromMaster`
The most recent measurement of the time difference in nanoseconds relative to the grandmaster clock.

`stepsRemoved`
The number of network steps between this system and the grandmaster clock.

For more information, see the `phc2sys(8)`, `pmc(8)`, and `ptp4l(8)` manual pages, https://www.zhaw.ch/en/engineering/institutes-centres/ines/downloads/documents.html, and IEEE 1588.

Using PTP as a Time Source for NTP

To make the PTP-adjusted system time on an NTP server available to NTP clients, include the following entries in `/etc/ntp.conf` on the NTP server to define the local system clock as the time reference:

```
server    127.127.1.0
fudge     127.127.1.0 stratum 0
```

> **Note:**
>
> Do not configure any additional `server` lines in the file.

For more information, see Configuring the ntpd Service.

5
Configuring the Apache HTTP Web Service

This chapter describes how to configure a basic HTTP server.

About the Apache HTTP Server

Oracle Linux provides the Apache HTTP Server, which is an open-source web server developed by the Apache Software Foundation. The Apache server hosts web content, and responds to requests for this content from web browsers such as Firefox.

Installing the Apache HTTP Server

To install the Apache HTTP server:

1. Enter the following command:

   ```
   sudo yum install httpd
   ```

2. Start the server, and configure it to start after system reboots:

   ```
   sudo apachectl start
   sudo systemctl enable httpd
   ```

3. Check for configuration errors:

   ```
   sudo apachectl configtest
   ```

4. Create firewall rules to allow access to the ports on which the HTTP server listens, for example:

   ```
   sudo firewall-cmd --zone=zone --add-service=http
   sudo firewall-cmd --permanent --zone=zone --add-service=http
   ```

Configuring the Apache HTTP Server

> **Note:**
>
> Any changes that you make to the configuration of the Apache HTTP server do not take effect until you restart the server:
>
> ```
> sudo apachectl restart
> ```

The main configuration file for the Apache HTTP server is `/etc/httpd/conf/httpd.conf`. You can modify the directives in this file to customize Apache for your environment.

The directives include:

`Allow from client [client ...] | all`
Specifies a list of clients that can access content or `all` to serve content to any client. The `Order` directive determines the order in which `httpd` evaluates `Allow` and `Deny` directives.

`Deny from client [client ...] | all`
Specifies a list of clients that cannot access content or `all` to disallow all clients. The `Order` directive determines the order in which `httpd` evaluates `Allow` and `Deny` directives.

`DocumentRoot directory-path`
The top level directory for Apache server content. The `apache` user requires read access to any files and read and execute access to the directory and any of its sub-directories. Do not place a slash at the end of the directory path.
For example:

```
DocumentRoot /var/www/html
```

If you specify a different document root or link to content that is not under `/var/www/html` and SELinux is enabled in enforcing mode on your system, change the default file type of the directory hierarchy that contains the content to `httpd_sys_content_t`:

1. Use the `semanage` command to define the default file type of the content directory as `httpd_sys_content_t`:

   ```
   sudo /usr/sbin/semanage fcontext -a -t httpd_sys_content_t " content_dir (/.*)?"
   ```

2. Use the `restorecon` command to apply the file type to the entire content directory hierarchy.

   ```
   sudo /sbin/restorecon -R -v content_dir
   ```

`ErrorLog filename | syslog[:facility]`
If set to a file name, specifies the file, relative to `ServerRoot`, to which `httpd` sends error messages.
If set to `syslog`, specifies that `httpd` send errors to `rsyslogd`. A *facility* argument specifies the `rsyslogd` facility. The default facility is `local7`.
For example:

```
ErrorLog logs/error_log
```

`Listen [IP_address:]port`
Accept incoming requests on the specified port or IP address and port combination. By default, the `httpd` server accepts requests on port 80 for all network interfaces. For a port number other than 80, HTTP requests to the server must include the port number.
For example:

```
Listen 80
Listen 192.168.2.1:8080
```

`LoadModule module path`
The Apache HTTP server can load external modules (dynamic shared objects or DSOs) to extend its functionality. The *module* argument is the name of the DSO, and `filename` is the path name of the module relative to `ServerRoot`.
For example:

```
LoadModule auth_basic_module modules/mod_auth_basic.so
```

`Order deny,allow | allow,deny`
Specifies the order in which `httpd` evaluates Allow and Deny directives.
For example, permit access only to clients from the `mydom.com` domain:

```
Order deny,allow
Deny from all
Allow from .mydom.com
```

The following directives would not permit access by any client:

```
Order allow,deny
Deny from all
Allow from .mydom.com
```

`ServerName FQDN[:port]`
Specifies the fully qualified domain name or IP address of the `httpd` server and an optional port on which the server listens. The FQDN must be resolvable to an IP address. If you do not specify a FQDN, the server performs a reverse-name lookup on the IP address. If you do not specify a port, the server uses the port corresponding to the incoming request.
For example:

```
ServerName www.mydom.com:80
```

`ServerRoot directory-path`
The top of the directory hierarchy where the `httpd` server keeps its configuration, error, and log files. Do not place a slash at the end of the directory path.
For example:

```
ServerRoot /etc/httpd
```

`Timeout seconds`
Specifies the number of seconds that `httpd` waits for network operations to finish before reporting a timeout error. The default value is 60 seconds.

`UserDir directory-path ... | disabled [user ...] | enabled user ...`
If set to `disabled`, disallows users identified by the space-separated *user* argument to publish content from their home directories. If no users are specified, all users are disallowed.
If set to `enabled`, allows users identified by the space-separated *user* argument to publish content from their home directories, provided that they are not specified as an argument to `disabled`.
directory-path is the name of a directory from which `httpd` publishes content. A relative path is assumed to be relative to a user's home directory. If you specify more than one directory path, `httpd` tries each alternative in turn until find a web page. If *directory-path* is not defined, the default is *~/public_html*. Do not place a slash at the end of the directory path.
For example:

```
UserDir disabled root guest
UserDir enabled oracle alice
UserDir www http://www.mydom.com/
```

The `root` and `guest` users are disabled from content publishing. Assuming that `ServerName` is set to `www.mydom.com`, browsing `http://www.example.com/~alice` displays `alice`'s web page, which must be located at `~alice/www` or `http://www.example.com/alice` (that is, in the directory `alice` relative to `ServerRoot`).

> **Note:**
>
> You would usually change the settings in the `<IfModule mod_userdir.c>` container to allow users to publish user content.

For more information, see https://httpd.apache.org/docs/current/mod/directives.html.

Testing the Apache HTTP Server

To test that an Apache HTTP server is working:

- From the local system, direct a browser on the local system to `http://localhost`.
- From a remote system, direct a browser to `http://` followed by the value of the `ServerName` directive specified in the configuration file (`/etc/httpd/conf/httpd.conf`).

If the browser displays the Apache 2 Test Page, the server is working correctly.

To test that the server can deliver content, create an HTML file named `index.html` in the directory specified by the `DocumentRoot` directive (by default, `/var/www/html`). After reloading the page, the browser should display this HTML file instead of the Apache 2 Test Page.

Configuring Apache Containers

Apache containers are special directives that group other directives, often to create separate web directory hierarchies with different characteristics. A container is delimited by the XML-style tags `<type>` and `</type>`, where *type* is the container type.

The following are examples of container types:

`<Directory directory-path>`
Applies the contained directives to directories under *directory-path*. The following example applies the `Deny`, `Allow`, and `AllowOverride` directives to all files and directories under `/var/www/html/sandbox`.

```
<Directory /var/www/html/sandbox>
  Deny from all
  Allow from 192.168.2.
  AllowOverride All
</Directory>
```

The `AllowOverride` directive is only used in `Directory` containers and specifies which classes of directives are allowed in `.htaccess` files. (`.htaccess` configuration files typically contain user authentication directives for a web directory.) The directive classes control such aspects as authorization, client access, and directory indexing. You can specify the argument `All` to permit all classes of directives in `.htaccess` files, a space-separated list of directive classes to permit only those classes, or `None` to make the server ignore `.htaccess` files altogether.

> **Note:**
>
> If SELinux is enabled on the system, you must change the default file type if the file system hierarchy specified by `<Directory>` is not under `/var/www/html`.

`<IfModule [!]module>`
Applies directives if the specified module has been loaded, or, when the exclamation point (`!`) is specified, if the module has not been loaded.
The following example disallows user-published content if `mod_userdir.c` has been loaded:

```
<IfModule mod_userdir.c>
  UserDir disabled
</IfModule>
```

`<Limit method ...>`
Places limits on the specified HTTP methods (such as GET, OPTIONS, POST, and PUT) for use with a Uniform Resource Identifier (URI).
The following example limits systems in `mydom.com` to using only the `GET` and `PUT` methods to perform HTTP downloads and uploads:

```
<Limit GET PUT>
  Order deny,allow
  Deny from all
  Allow from .example.com
</Limit>
```

Systems outside `mydom.com` cannot use `GET` and `PUT` with the URI.

`<LimitExcept method ...>`
Places limits on all except the specified HTTP methods for use with a Uniform Resource Identifier (URI).
The following example disallows any system from using any method other than `GET` and `POST`:

```
<LimitExcept GET POST>
  Order deny,allow
  Deny from all
</Limit>
```

`VirtualHost IP_address:port ...`
Specifies a group of directives that define a container for a virtual host. See Configuring Apache Virtual Hosts.

About Nested Containers

The following example illustrates how you can nest containers, using `<Limit>` and `<LimitExcept>` containers to permit `GET`, `POST`, and `OPTIONS` to be used with user directories under `/home/*/public_html`.

```
<Directory /home/*/public_html>
  AllowOverride FileInfo AuthConfig Limit
  Options MultiViews Indexes SymLinksIfOwnerMatch \
  IncludesNoExec
  <Limit GET POST OPTIONS>
    Order allow,deny
    Allow from all
```

```
  </Limit>
  <LimitExcept GET POST OPTIONS>
    Order deny,allow
    Deny from all
  </LimitExcept>
</Directory>
```

In the example, the `AllowOverride` directive specifies the following directive classes:

`AuthConfig`
Permits the use of the authorization directives.

`FileInfo`
Permits the use of directives that control document types.

`Limit`
Permits the use of directives that control host access.

The `Options` directive controls the features of the server for the directory hierarchy, for example:

`FollowSymLinks`
Follow symbolic links under the directory hierarchy.

`Includes`
Permits server-side includes.

`IncludesNoExec`
Prevents the server from running `#exec cmd` and `#exec cgi` server-side includes.

`Indexes`
Generates a web directory listing if the `DirectoryIndex` directive is not set.

`MultiViews`
Allows the server to determine the file to use that best matches the client's requirements based on the MIME type when several versions of the file exist with different extensions.

`SymLinksIfOwnerMatch`
Allows the server to follow a symbolic link if the file or directory being pointed to has the same owner as the symbolic link.

For more information, see https://httpd.apache.org/docs/current/mod/directives.html.

Configuring Apache Virtual Hosts

The Apache HTTP server supports virtual hosts, meaning that it can respond to requests that are directed to multiple IP addresses or host names that correspond to the same host machine. You can configure each virtual host to provide different content and to behave differently.

You can configure virtual hosts in two ways:

- *IP-based Virtual Hosts (host-by-IP)*

  Each virtual host has its own combination of IP address and port. The server responds to the IP address with which the host name resolves. Host-by-IP is

needed to server HTTPS requests because of restrictions in the SSL (Secure Sockets Layer) protocol.

- *Name-based Virtual Hosts (host-by-name)*

  All virtual hosts share a common IP address. Apache responds to the request by mapping the host name in the request to `ServerName` and `ServerAlias` directives for the virtual host in the configuration file.

To configure a virtual host, you use the `<VirtualHost hostname>` container. You must also divide all served content between the virtual hosts that you configure.

The following example shows a simple name-based configuration for two virtual hosts:

```
NameVirtualHost *:80

<VirtualHost *:80>
  ServerName websvr1.mydom.com
  ServerAlias www.mydom-1.com
  DocumentRoot /var/www/http/websvr1
  ErrorLog websvr1.error_log
</VirtualHost>

<VirtualHost *:80>
  ServerName websvr2.mydom.com
  ServerAlias www.mydom-2.com
  DocumentRoot /var/www/http/sebsvr2
  ErrorLog websvr2.error_log
</VirtualHost>
```

For more information, see https://httpd.apache.org/docs/2.2/vhosts/.

6
Email Service Configuration

This chapter describes email programs and protocols that are available with Oracle Linux, and how to set up a basic Sendmail client.

About Email Programs

A Mail User Agent is an email client application that allows you to create and read email messages, set up mailboxes to store and organize messages, and send outbound messages to a Mail Transfer Agent (MTA). Many MUAs can also retrieve email messages from remote servers using the Post Office Protocol (POP) or Internet Message Access Protocol (IMAP).

A Mail Transfer Agent (MTA) transports email messages between systems by using the Simple Mail Transport Protocol (SMTP). The mail delivery services from the client program to a destination server possibly traverses several MTAs in its route. Oracle Linux offers two MTAs, Postfix and Sendmail, and also includes the special purpose MTA, Fetchmail for use with SLIP and PPP.

A Mail Delivery Agent (MDA) performs the actual delivery of an email message. The MTA invokes an MDA, such as Procmail, to place incoming email in the recipient's mailbox file. MDAs distribute and sort messages on the local system that email client application can access.

About Email Protocols

Several different network protocols are required to deliver email messages. These protocols work together to allow different systems, often running different operating systems and different email programs, to send, transfer, and receive email.

About SMTP

The Simple Mail Transfer Protocol (SMTP) is a transport protocol that provides mail delivery services between email client applications and servers, and between the originating server and the destination server. You must specify the SMTP server when you configure outgoing email for an email client application.

SMTP does not require authentication. Anyone can use SMTP to send email, including junk email and unsolicited bulk email. If you administer an SMTP server, you can configure relay restrictions that limit users from sending email through it. Open relay servers do not have any such restrictions. Both Postfix and Sendmail are SMTP server programs that use SMTP. Unless you own a domain in which you want to receive email, you do not need to set up an SMTP server.

About POP and IMAP

The Post Office Protocol (POP) is an email access protocol that email client applications use to retrieve email messages from the mailbox on a remote server, typically maintained by an

Internet Service Provider (ISP). POP email clients usually delete the message on the server when it has been successfully retrieved or within a short time period thereafter.

The Internet Message Access Protocol (IMAP) is an email access protocol that email client applications use to retrieve email messages from a remote server, typically maintained by their organization. The entire message is downloaded only when you open it, and you can delete messages from the server without first downloading them. Email is retained on the server when using IMAP.

Both POP and IMAP allow you to manage mail folders and create multiple mail directories to organize and store email.

The `dovecot` package provides the `dovecot` service that implements both an IMAP server and a POP server.

By default, the dovecot service runs IMAP and POP together with their secure versions that use Secure Socket Layer (SSL) encryption for client authentication and data transfer sessions. The IMAP and POP servers provided by `dovecot` are configured to work as installed. It is usually unnecessary to modify the configuration file, `/etc/dovecot.conf`.

For more information, see the `dovecot(1)` manual page and `/usr/share/doc/dovecot-`*`version`* .

About the Postfix SMTP Server

Postfix is configured as the default MTA on Oracle Linux. Although Postfix does not have as many features as Sendmail, it is easier to administer than Sendmail and its features are sufficient to meet the requirements of most installations. You should only use Sendmail if you want to use address re-writing rules or mail filters (*milters*) that are specific to Sendmail. Most mail filters function correctly with Postfix. If you do use Sendmail, disable or uninstall Postfix to avoid contention over network port usage.

Postfix has a modular design that consists of a primary daemon and several smaller processes. Postfix stores its configuration files in the `/etc/postfix` directory, including:

`access`
Specifies which hosts are allowed to connect to Postfix.

`main.cf`
Contains global configuration options for Postfix.

`master.cf`
Specifies how the Postfix master daemon and other Postfix processes interact to deliver email.

`transport`
Specifies the mapping between destination email addresses and relay hosts.

By default, Postfix does not accept network connections from any system other than the local host. To enable mail delivery for other hosts, edit `/etc/postfix/main.cf` and configure their domain, host name, and network information.

Restart the Postfix service after making any configuration changes:

```
sudo systemctl restart postfix
```

For more information, see `postfix(1)` and other Postfix manual pages, Forwarding Email, `/usr/share/doc/postfix-`*`version`* , and http://www.postfix.org/documentation.html.

About the Sendmail SMTP Server

Sendmail is highly configurable and is the most commonly used MTA on the Internet. Sendmail is mainly used to transfer email between systems, but it is capable of controlling almost every aspect of how email is handled.

Sendmail is distributed in the following packages:

`procmail`
Contains Procmail, which acts as the default local MDA for Sendmail. This package is installed as a dependency of the `sendmail` package.

`sendmail`
Contains the Sendmail MTA.

`sendmail-cf`
Contains configuration files for Sendmail.

To install the Sendmail packages, enter:

```
sudo yum install sendmail sendmail-cf
```

For more information, see the `sendmail(8)` manual page .

About Sendmail Configuration Files

The main configuration file for Sendmail is `/etc/mail/sendmail.cf`, which is not intended to be manually edited. Instead, make any configuration changes in the `/etc/mail/sendmail.mc` file.

If you want Sendmail to relay email from other systems, change the following line in `sendmail.mc`:

```
DAEMON_OPTIONS(`Port=smtp,Addr=127.0.0.1, Name=MTA')dnl
```

so that it reads:

```
dnl # DAEMON_OPTIONS(`Port=smtp,Addr=127.0.0.1, Name=MTA')dnl
```

The leading `dnl` stands for *delete to new line*, and effectively comments out the line.

After you have edited `sendmail.mc`, restart the `sendmail` service to regenerate `sendmail.cf`:

```
sudo systemctl restart sendmail
```

Alternatively, you can use the `make` script in `/etc/mail`:

```
sudo /etc/mail/make all
```

However, Sendmail does not use the regenerated configuration file until you restart the server.

Other important Sendmail configuration files in `/etc/mail` include:

access
Configures a relay host that processes outbound mail from the local host to other systems. This is the default configuration:

```
Connect: localhost.localdomain      RELAY
Connect: localhost                  RELAY
Connect: 127.0.0.1                  RELAY
```

To configure Sendmail to relay mail from other systems on a local network, add an entry such as the following:

```
Connect: 192.168.2                    RELAY
```

mailertable
Configures forwarding of email from one domain to another. The following example forwards email sent to the `yourorg.org` domain to the SMTP server for the `mydom.com` domain:

```
yourorg.org        smtp:[mydom.com]
```

virtusertable
Configures serving of email to multiple domains. Each line starts with a destination address followed by the address to which Sendmail forwards the email. For example, the following entry forwards email addressed to any user at `yourorg.org` to the same user name at `mydom.com`:

```
@yourorg.org           %1@mydom.com
```

Each of these configuration files has a corresponding database (`.db`) file in `/etc/mail` that Sendmail reads. After making any changes to any of the configuration files, restart the `sendmail` service. To regenerate the database files, run the `/etc/mail/make all` command. As for `sendmail.cf`, Sendmail does not use the regenerated database files until you restart the server.

Forwarding Email

You can forward incoming email messages with the Postfix `local` delivery agent or with Sendmail by configuring the `/etc/aliases` file. Entries in this file can map inbound addresses to local users, files, commands, and remote addresses.

The following example redirects email for `postmaster` to `root`, and forwards email sent to `admin` on the local system to several other users, including `usr04`, who is on a different system:

```
postmaster:   root
admin:        usr01, usr02, usr03, usr04@another-system.com
```

To direct email to a file, specify an absolute path name instead of the destination address. To specify a command, precede it with a pipe character (`|`). The next example erases email sent to `nemo` by sending it to `/dev/null`, and runs a script named `aggregator` to process emails sent to `fixme`:

```
nemo:         /dev/null
fixme:        |/usr/local/bin/aggregator
```

After changing the file, run the command `newaliases` to rebuild the indexed database file.

For more information, see the `aliases(5)` manual page.

Configuring a Sendmail Client

A Sendmail client sends outbound mail to another SMTP server, which is typically administered by an ISP or the IT department of an organization, and this server then relays the email to its destination.

To configure a Sendmail client:

1. If the account on the SMTP server requires authentication:
   a. Create an `auth` directory under `/etc/mail` that is accessible only to `root`:

      ```
      sudo mkdir /etc/mail/auth
      sudo chmod 700 /etc/mail/auth
      ```

   b. In the `auth` directory, create a file `smtp-auth` that contains the authentication information for the SMTP server, for example:

      ```
      sudo echo 'AuthInfo:smtp.isp.com: "U:username" "P:password"' > /etc/mail/auth/
      smtp-auth
      ```

      In the previous command, *smtp.isp.com* is the FQDN of the SMTP server, and *username* and *password* are the name and password of the account.

   c. Create the database file from `smtp-auth`, and make both files read-writable only by `root`:

      ```
      cd /etc/mail/auth
      makemap hash smtp-auth < smtp-auth
      chmod 600 smtp-auth smtp-auth.db
      ```

2. Edit `/etc/mail/sendmail.mc`, and change the following line:

   ```
   dnl define('SMART_host', 'smtp.your.provider')dnl
   ```

   to read:

   ```
   define('SMART_host', 'smtp.isp.com')dnl
   ```

   In the previous command, *smtp.isp.com* is the FQDN of the SMTP server.

3. If the account on the SMTP server requires authentication, add the following lines after the line that defines `SMART_host`:

   ```
   define('RELAY_MAILER_ARGS', 'TCP $h port')dnl
   define('confAUTH_MECHANISMS', 'EXTERNAL GSSAPI DIGEST-MD5 CRAM-MD5 LOGIN PLAIN')dnl
   FEATURE('authinfo','hash /etc/mail/auth/smtp-auth.db')dnl
   define(`confAUTH_OPTIONS', `A p y')dnl
   ```

   In the previouis command, *port* is the port number that is used by the SMTP server (for example, 587 for SMARTTLS or 465 for SSL/TLS).

4. Edit `/etc/sysconfig/sendmail` and set the value of `DAEMON` to `no`:

   ```
   DAEMON=no
   ```

   This entry disables `sendmail` from listening on port 25 for incoming email.

5. Restart the `sendmail` service:

   ```
   sudo systemctl restart sendmail
   ```

To test the configuration, send email to an account in another domain.

This configuration does not receive or relay incoming email. You can use a client application to receive email via POP or IMAP.

7

Configuring High Availability Features

This chapter describes how to configure the Pacemaker and Corosync technologies to create an HA cluster that delivers continuous access to services running across multiple nodes.

More information and documentation on Pacemaker and Corosync can also be found at https://clusterlabs.org/pacemaker/doc/.

About Oracle Linux High Availability Services

Oracle Linux high availability services comprises several open-source packages, including Corosync and Pacemaker, to provide the tools to achieve high availability for applications and services running on Oracle Linux. You may download Corosync, Pacemaker and the functional sub packages from the Unbreakable Linux Network at https://linux.oracle.com or the Oracle Linux yum server at https://yum.oracle.com.

Corosync is an open source cluster engine that includes an API to implement a number of high availability features, including an availability manager that can restart a process when it fails, a configuration and statistics database and a quorum system that can notify applications when quorum is achieved or lost.

Corosync is installed in conjunction with Pacemaker, an open source high availability cluster resource manager responsible for managing the life-cycle of software deployed on a cluster and for providing high availability services. High availability services are achieved by detecting and recovering from node and resource level failures via the API provided by the cluster engine.

Pacemaker also ships with the Pacemaker Command Shell (`pcs`) that can be used to access and configure the cluster and its resources. The pcs daemon runs as a service on each node in the cluster, making it possible to synchronize configuration changes across all of the nodes in the cluster.

Oracle provides support for Corosync and Pacemaker used for an active-passive 2-node (1:1) cluster configuration on Oracle Linux 7.3 or higher. Support for clustering services does not imply support for Oracle products clustered using these services.

Oracle also provides Oracle Clusterware for high availability clustering with Oracle Database. You can find more information at https://www.oracle.com/database/technologies/rac/clusterware.html.

Installing Pacemaker and Corosync

On each node in the cluster, install the `pcs` and `pacemaker` software packages along with all available resource and fence agents from the Oracle Linux yum server or from the Unbreakable Linux Network.

```
sudo yum install pcs pacemaker resource-agents fence-agents-all
```

If you are running `firewalld`, you should add the `high-availability` service on each of the nodes, so that the service components are able to communicate across the network. This

step typically enables TCP ports 2224 (used by the pcs daemon), 3121 (for Pacemaker Remote nodes), 21064 (for DLM resources); and UDP ports 5405 (for Corosync clustering) and 5404 (for Corosync multicast, if this is configured).

```
sudo firewall-cmd --permanent --add-service=high-availability
sudo firewall-cmd --add-service=high-availability
```

To use the `pcs` command to configure and manage your cluster, a password must be set on each node for the `hacluster` user. It is helpful if the password that you set for this user is the same on each node. Use the `passwd` command on each node to set the password:

```
sudo passwd hacluster
```

To use the `pcs` command, the `pcsd` service must be running on each of the nodes in the cluster. You can set this service to run and to start at boot using the following commands:

```
sudo systemctl start pcsd.service
sudo systemctl enable pcsd.service
```

Configuring an Initial Cluster and Service

In the following example, a cluster is configured across two nodes hosted on systems with the resolvable hostnames of `node1` and `node2`. Each system is installed and configured using the instructions provided in Installing Pacemaker and Corosync.

The cluster is configured to run a service, `Dummy`, that is included in the `resource-agents` package that you should have installed along with the pacemaker packages. This tool simply keeps track of whether it is running or not. We configure Pacemaker with an interval parameter that determines how long it should wait between checks to determine whether the `Dummy` process has failed.

We manually stop the `Dummy` process outside of the Pacemaker tool to simulate a failure and use this to demonstrate how the process is restarted automatically on an alternate node.

Creating the Cluster

1. Authenticate the `pcs` cluster configuration tool for the `hacluster` user on each node in your configuration. To do this, run the following command on one of the nodes that will form part of the cluster:

   ```
   sudo pcs cluster auth node1node2 -u hacluster
   ```

   Replace *node1* and *node2* with the resolvable hostnames of the nodes that will form part of the cluster. The tool will prompt you to provide a password for the `hacluster` user. You should provide the password that you set for this user when you installed and configured the Pacemaker software on each node.

2. To create the cluster, use the `pcs cluster setup` command. You must specify a name for the cluster and the resolvable hostnames for each node in the cluster:

   ```
   sudo pcs cluster setup --name pacemaker1 node1 node2
   ```

   Replace *pacemaker1* with an appropriate name for the cluster. Replace *node1* and *node2* with the resolvable hostnames of the nodes in the cluster.

3. Start the cluster on all nodes. You can do this manually using the `pcs` command:

   ```
   sudo pcs cluster start --all
   ```

   You can also do this by starting the `pacemaker` and `corosync` services from systemd:

   ```
   sudo systemctl start pacemaker.service
   sudo systemctl start corosync.service
   ```

   Optionally, you can enable these services to start at boot time, so that if a node reboots it automatically rejoins the cluster:

   ```
   sudo systemctl enable pacemaker.service
   sudo systemctl enable corosync.service
   ```

   Some users prefer not to enable these services, so that a node failure resulting in a full system reboot can be properly debugged before it rejoins the cluster.

Setting Cluster Parameters

1. Fencing is an advanced feature that helps protect your data from being corrupted by nodes that may be failing or unavailable. Pacemaker uses the term `stonith` (shoot the other node in the head) to describe fencing options. Since this configuration depends on particular hardware and a deeper understanding of the fencing process, we recommend disabling the fencing feature for this example.

   ```
   sudo pcs property set stonith-enabled=false
   ```

   Fencing is an important part of setting up a production level HA cluster and is disabled in this example to keep things simple. If you intend to take advantage of `stonith`, see Fencing Configuration for more information.

2. Since this example is a two-node cluster, you can disable the no-quorum policy, as quorum requires a minimum of three nodes to make any sense. Quorum is only achieved when more than half of the nodes agree on the status of the cluster. In this example, quorum can never be reached, so configure the cluster to ignore the quorum state:

   ```
   sudo pcs property set no-quorum-policy=ignore
   ```

3. Configure a migration policy. In this example we configure the cluster to move the service to a new node after a single failure:

   ```
   sudo pcs resource defaults migration-threshold=1
   ```

Creating a Service and Testing Failover

Creating a service and testing failover

Services are created and are usually configured to run a resource agent that is responsible for starting and stopping processes. Most resource agents are created according to the OCF (Open Cluster Framework) specification defined as an extension for the Linux Standard Base (LSB). There are many handy resource agents for commonly used processes included in the `resource-agents` packages, including a variety of heartbeat agents that track whether commonly used daemons or services are still running.

In this example we set up a service that uses a Dummy resource agent created precisely for the purpose of testing Pacemaker. We use this agent because it requires the least possible configuration and does not make any assumptions about your environment or the types of services that you intend to run with Pacemaker.

1. To add the service as a resource, use the `pcs resource create` command. Provide a name for the service. In the example below, we use the name *dummy_service* for this resource:

   ```
   sudo pcs resource create dummy_service ocf:pacemaker:Dummy op monitor
   interval=120s
   ```

   To invoke the Dummy resource agent, a notation (`ocf:pacemaker:Dummy`) is used to specify that it conforms to the OCF standard, that it runs in the pacemaker namespace and that the Dummy script should be used. If you were configuring a heartbeat monitor service for an Oracle Database, you might use the `ocf:heartbeat:oracle` resource agent.

   The resource is configured to use the monitor operation in the agent and an interval is set to check the health of the service. In this example we set the interval to 120s to give the service some time to fail while you are demonstrating failover. By default, this is usually set to 20 seconds, but may be modified depending on the type of service and your own environment.

2. As soon as you create a service, the cluster attempts to start the resource on a node using the resource agent's start command. You can see the resource start and run status by running the `pcs status` command:

   ```
   sudo pcs status

   Cluster name: pacemaker1
   Stack: corosync
   Current DC: node1 (version 1.1.16-12.el7-94ff4df) - partition with quorum
   Last updated: Wed Jan 17 06:35:18 2018
   Last change: Wed Jan 17 03:08:00 2018 by root via cibadmin on node1

   2 nodes configured
   1 resource configured

   Online: [ node2 node1 ]

   Full list of resources:

    dummy_service   (ocf::pacemaker:Dummy): Started node2

   Daemon Status:
     corosync: active/enabled
     pacemaker: active/enabled
     pcsd: active/enabled
   ```

3. Simulate service failure by force stopping the service directly, using `crm_resource`, so that the cluster is unaware that the service has been manually stopped.

   ```
   sudo crm_resource --resource dummy_service --force-stop
   ```

4. Run `crm_mon` in interactive mode so that you can wait until you see the node fail and a `Failed Actions` message is displayed. You should see the service restart on the alternate node.

   ```
   sudo crm_mon

   Stack: corosync
   Current DC: node1 (version 1.1.16-12.el7-94ff4df) - partition with quorum
   Last updated: Wed Jan 17 06:41:04 2018
   Last change: Wed Jan 17 06:39:02 2018 by root via cibadmin on node1
   ```

```
2 nodes configured
1 resource configured

Online: [ node2 node1 ]

Active resources:

dummy_service    (ocf::pacemaker:Dummy): Started node1

Failed Actions:
* dummy_service_monitor_120000 on node2 'not running' (7): call=16,
status=complete, exitreason='none',
    last-rc-change='Wed Jan 17 06:41:02 2018', queued=0ms, exec=0ms
```

You can use the `Ctrl-C` key combination to exit out of `crm_mon` at any point.

5. You can try to reboot the node where the service is running to see that failover also occurs in the case of node failure. Note that if you have not enabled the corosync and pacemaker services to start on boot, you may need to start the service on the node that you have rebooted, manually. For example:

```
sudo pcs cluster start node1
```

Fencing Configuration

Fencing or `stonith` is used to protect data when nodes become unresponsive. If a node fails to respond, it may still be accessing data. To be sure that your data is safe, you can use fencing to prevent a live node from having access to the data until the original node is truly offline. To do this, you must configure a device that can ensure that a node is taken offline. There are a number of fencing agents available that can be configured for this purpose. In general, `stonith` relies on particular hardware and service protocols that can force reboot or shutdown nodes physically to protect the cluster.

In this section, different configurations using some of the available fencing agents are presented as examples. Note that these examples make certain presumptions about hardware and assume that you are already aware of how to set up, configure and use the hardware concerned. The examples are provided for basic guidance and it is recommended that you also refer to upstream documentation to familiarize yourself with some of the concepts presented here.

Before proceeding with any of these example configurations, you must ensure that `stonith` is enabled for your cluster configuration:

```
sudo pcs property set stonith-enabled=true
```

After you have configured `stonith`, you can check your configuration to ensure that it is set up correctly by running the following commands:

```
sudo pcs stonith show --full
sudo pcs cluster verify -V
```

To check the status of your `stonith` configuration, run:

```
sudo pcs stonith
```

To check the status of your cluster, run:

```
sudo pcs status
```

IPMI LAN Fencing

Intelligent Platform Management Interface (IPMI) is an interface to a subsystem that provides management features of the host system's hardware and firmware and includes facilities to power cycle a system over a dedicated network without any requirement to access the system's operating system. The `fence_ipmilan` fencing agent can be configured for the cluster so that `stonith` can be achieved across the IPMI LAN.

If your systems are configured for IPMI, you can run the following commands on one of the nodes in the cluster to enable the `ipmilan` fencing agent and to configure `stonith` for both nodes:

```
sudo pcs stonith create ipmilan_n1_fencing fence_ipmilan pcmk_host_list=node1
delay=5 \
ipaddr=203.0.113.1 login=root passwd=password lanplus=1 op monitor interval=60s

sudo pcs stonith create ipmilan_n2_fencing fence_ipmilan pcmk_host_list=node2 \
ipaddr=203.0.113.2 login=root passwd=password lanplus=1 op monitor interval=60s
```

In the above example, the host named *node1* has an IPMI LAN interface configured on the IP *203.0.113.1*. The host named *node2* has an IPMI LAN interface configured on the IP *203.0.113.2*. The root user password for the IPMI login on both systems is specified here as *password*. In each instance, you should replace these configuration variables with the appropriate information to match your own environment.

Note that the delay option should only be set to one node. This helps to ensure that in the rare case of a fence race condition only one node is killed and the other continues to run. Without this option set, it is possible that both nodes believe they are the only surviving node and simultaneously reset each other.

> **NOT_SUPPORTED:**
>
> The IPMI LAN agent exposes the login credentials of the IPMI subsystem in plain text. Your security policy should ensure that it is acceptable for users with access to the Pacemaker configuration and tools to also have access to these credentials and the underlying subsystems concerned.

SCSI Fencing

The SCSI Fencing agent is used to provide storage level fencing. This protects storage resources from being written to by two nodes at the same time, using SCSI-3 PR (Persistent Reservation). Used in conjunction with a watchdog service, a node can be reset automatically via `stonith` when it attempts to access the SCSI resource without a reservation.

To configure an environment in this way, install the watchdog service on *both* nodes and copy the provided `fence_scsi_check` script to the watchdog configuration before enabling the service:

```
sudo yum install watchdog
sudo cp /usr/share/cluster/fence_scsi_check /etc/watchdog.d/
sudo systemctl enable --now watchdog
```

To use this fencing agent, you must also enable the iscsid service provided in the `iscsi-initiator-utils` package on *both* nodes:

```
sudo yum install -y iscsi-initiator-utils
sudo systemctl enable --now iscsid
```

Once both nodes are configured with the watchdog service and the iscsid service, you can configure the `fence_scsi` fencing agent on one of the cluster nodes to monitor a shared storage device, such as an iSCSI target. For example:

```
sudo pcs stonith create scsi_fencing fence_scsi pcmk_host_list="node1 node2" \
 devices="/dev/sdb" meta provides="unfencing"
```

In the example, *node1* and *node2* are the hostnames of the nodes in the cluster and */dev/sdb* is the shared storage device. You should replace these variables with the appropriate information to match your own environment.

SBD Fencing

Storage Based Death (SBD) is a daemon that can run on a system and monitor shared storage and that can use a messaging system to track cluster health. SBD can trigger a reset in the event that the appropriate fencing agent determines that `stonith` should be implemented.

To set up and configure SBD fencing, stop the cluster by running the following command on one of the nodes:

```
sudo pcs cluster stop --all
```

On each node, install and configure the SBD daemon:

```
sudo yum install sbd
```

Edit `/etc/sysconfig/sbd` to set the `SBD_DEVICE` parameter to identify the shared storage device. For example, if your shared storage device is available on */dev/sdc*, edit the file to contain the line:

```
SBD_DEVICE="/dev/sdc"
```

Enable the SBD service in systemd:

```
sudo systemctl enable --now sbd
```

On one of the nodes, create the SBD messaging layout on the shared storage device and confirm that it is in place. For example, to set up and verify messaging on the shared storage device at */dev/sdc*, run the following commands:

```
sudo sbd -d /dev/sdc create
sudo sbd -d /dev/sdc list
```

Finally, start the cluster and configure the fence_sbd fencing agent for the shared storage device. For example, to configure the shared storage device, */dev/sdc*, run the following commands on one of the nodes:

```
sudo pcs cluster start --all
sudo pcs stonith create sbd_fencing fence_sbd devices=/dev/sdc
```

IF-MIB Fencing

IF-MIB fencing takes advantage of SNMP to access the IF-MIB on an Ethernet network switch and to shutdown the port on the switch to effectively take a host offline. This leaves the host running, but disconnects it from the network. It is worth bearing in mind that any FibreChannel or InfiniBand connections could remain intact, even after the Ethernet connection has been terminated, which could mean that data made available on these connections could still be at risk. As a result, it is best to configure this as a fallback fencing mechanism. See Configuring Fencing Levels for more information on how to use multiple fencing agents together to maximise `stonith` success.

To configure IF-MIB fencing, ensure that your switch is configured for SNMP v2c at minimum and that SNMP SET messages are enabled. For example, on an Oracle Switch, via the ILOM CLI, you could run:

```
sudo set /SP/services/snmp/ sets=enabled
sudo set /SP/services/snmp/ v2c=enabled
```

On one of the nodes in your cluster, configure the `fence_ifmib` fencing agent for each node in your environment. For example:

```
sudo pcs stonith create ifmib_n1_fencing fence_ifmib pcmk_host_list=node1 \
ipaddr=203.0.113.10 community=private port=1 delay=5 op monitor interval=60s

sudo pcs stonith create ifmib_n2_fencing fence_ifmib pcmk_host_list=node2 \
ipaddr=203.0.113.10 community=private port=2 op monitor interval=60s
```

In the above example, the switch SNMP IF-MIB is accessible at the IP address *203.0.113.10*. The host *node1* is connected to port *1* on the switch. The host *node2* is connected to port *2* on the switch. You should replace these variables with the appropriate information to match your own environment.

Configuring Fencing Levels

If you have configured multiple fencing agents, you may want to set different fencing levels. Fencing levels allow you to prioritize different approaches to fencing and can provide a valuable mechanism to provide fallback options should a default fencing approach fail.

Each fencing level is attempted in ascending order starting from level 1. If the fencing agent configured for a particular level fails, the fencing agent from the next level is attempted instead.

For example, you may wish to configure IPMI-LAN fencing at level 1, but fallback to IF-MIB fencing as a level 2 option. Using the example configurations from IPMI LAN Fencing and IF-MIB Fencing, you could run the following commands on one of the nodes to set the fencing levels for each configured agent:

```
sudo pcs stonith level add 1 node1 ipmilan_n1_fencing
sudo pcs stonith level add 1 node2 ipmilan_n2_fencing
sudo pcs stonith level add 2 node1 ifmib_n1_fencing
sudo pcs stonith level add 2 node2 ifmib_n2_fencing
```

8
Configuring Load Balancing

This chapter describes how to configure the Keepalived and HAProxy technologies for balancing access to network services while maintaining continuous access to those services.

About HAProxy

HAProxy is an application layer (Layer 7) load balancing and high availability solution that you can use to implement a reverse proxy for HTTP and TCP-based Internet services.

The configuration file for the `haproxy` daemon is `/etc/haproxy/haproxy.cfg`. This file must be present on each server on which you configure HAProxy for load balancing or high availability.

For more information, see http://www.haproxy.org/#docs, the `/usr/share/doc/haproxy-`*`version`* documentation, and the `haproxy(1)` manual page.

Installing and Configuring HAProxy

To install HAProxy:

1. Install the `haproxy` package on each front-end server:

   ```
   sudo yum install haproxy
   ```

2. Edit `/etc/haproxy/haproxy.cfg` to configure HAProxy on each server. See About the HAProxy Configuration File.

3. Enable IP forwarding and binding to non-local IP addresses:

   ```
   echo "net.ipv4.ip_forward = 1" >> /etc/sysctl.conf
   echo "net.ipv4.ip_nonlocal_bind = 1" >> /etc/sysctl.conf
   sysctl -p

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

4. Enable access to the services or ports that you want HAProxy to handle.

   For example, to enable access to HTTP and make this rule persist across reboots, enter the following commands:

   ```
   sudo firewall-cmd --zone=zone --add-service=http

   success

   sudo firewall-cmd --permanent --zone=zone --add-service=http

   success
   ```

   To allow incoming TCP requests on port 8080:

   ```
   sudo firewall-cmd --zone=zone --add-port=8080/tcp
   ```

```
success

sudo firewall-cmd --permanent --zone=zone --add-port=8080/tcp

success
```

5. Enable and start the `haproxy` service on each server:

```
sudo systemctl enable haproxy
ln -s '/usr/lib/systemd/system/haproxy.service' '/etc/systemd/system/multi-
user.target.wants/haproxy.service'
sudo systemctl start haproxy
```

If you change the HAProxy configuration, reload the `haproxy` service:

```
sudo systemctl reload haproxy
```

About the HAProxy Configuration File

The `/etc/haproxy/haproxy.cfg` configuration file is divided into the following sections:

`global`
Defines global settings such as the `syslog` facility and level to use for logging, the maximum number of concurrent connections allowed, and how many processes to start in daemon mode.

`defaults`
Defines default settings for subsequent sections.

`listen`
Defines a complete proxy, implicitly including the `frontend` and `backend` components.

`frontend`
Defines the ports that accept client connections.

`backend`
Defines the servers to which the proxy forwards client connections.

For examples of how to configure HAProxy, see:

- Configuring Simple Load Balancing Using HAProxy
- Making HAProxy Highly Available Using Keepalived
- Making HAProxy Highly Available Using Oracle Clusterware

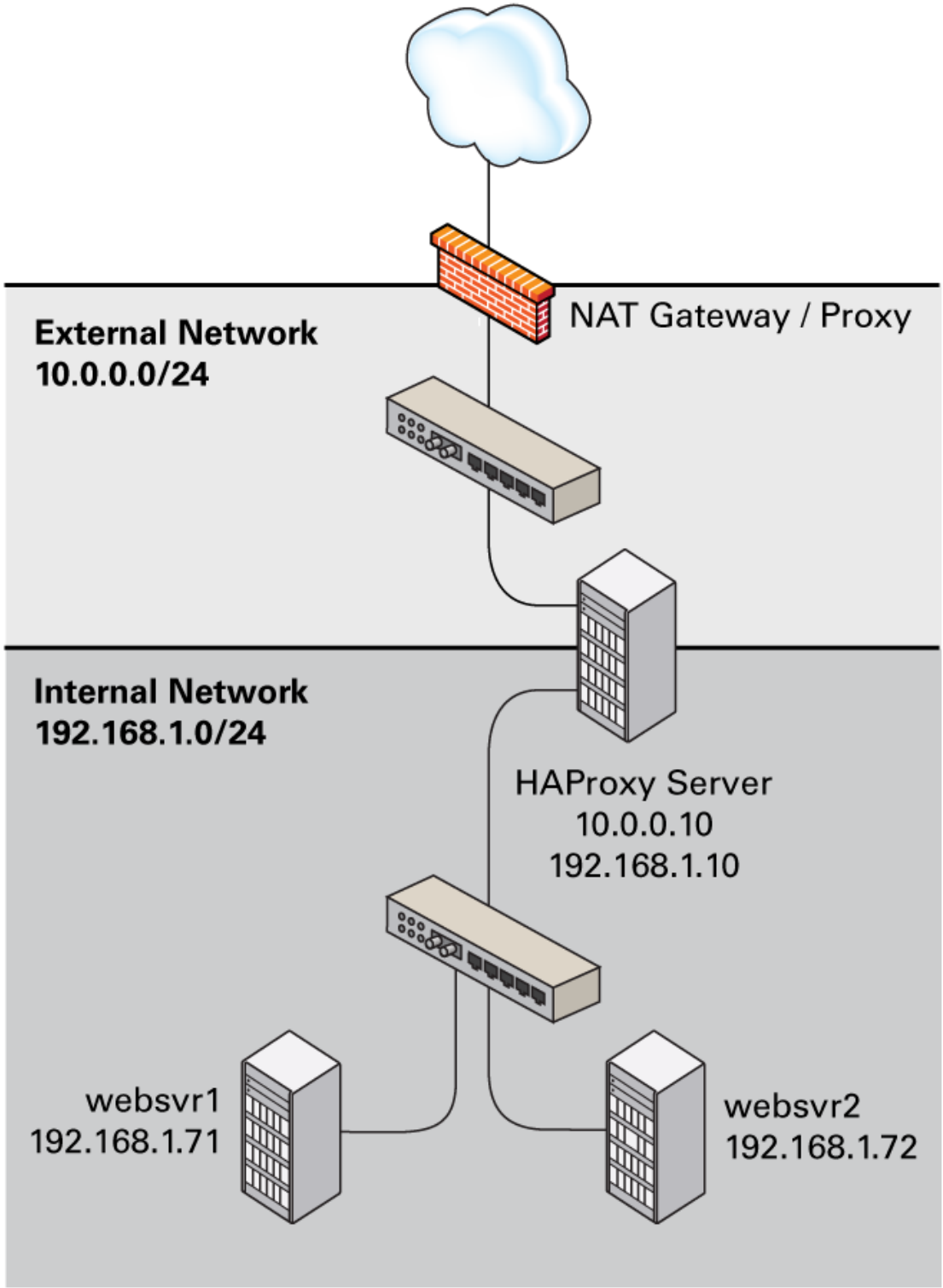
Configuring Simple Load Balancing Using HAProxy

The following example uses HAProxy to implement a front-end server that balances incoming requests between two back-end web servers, and which is also able to handle service outages on the back-end servers.

Figure 8-1 shows an HAProxy server (`10.0.0.10`), which is connected to an externally facing network (`10.0.0.0/24`) and to an internal network (`192.168.1.0/24`). Two web servers, `websvr1` (`192.168.1.71`) and `websvr2` (`192.168.1.72`), are accessible on the internal network. The IP address 192.168.1.10 is in the private address range `192.168.1.0/24`, which cannot be routed on the Internet. An upstream network

address translation (NAT) gateway or a proxy server provides access to and from the Internet.

Figure 8-1 Example HAProxy Configuration for Load Balancing



You might use the following configuration in `/etc/haproxy/haproxy.cfg` on the server:

```
global
    daemon
    log 127.0.0.1 local0 debug
    maxconn 50000
    nbproc 1

defaults
    mode http
    timeout connect 5s
    timeout client 25s
    timeout server 25s
    timeout queue 10s

# Handle Incoming HTTP Connection Requests
listen  http-incoming
    mode http
    bind 10.0.0.10:80
# Use each server in turn, according to its weight value
    balance roundrobin
# Verify that service is available
    option httpchk OPTIONS * HTTP/1.1\r\nHost:\ www
# Insert X-Forwarded-For header
    option forwardfor
# Define the back-end servers, which can handle up to 512 concurrent connections
each
    server websvr1 192.168.1.71:80 weight 1 maxconn 512 check
    server websvr2 192.168.1.72:80 weight 1 maxconn 512 check
```

This configuration balances HTTP traffic between the two back-end web servers `websvr1` and `websvr2`, whose firewalls are configured to accept incoming TCP requests on port 80.

After implementing simple `/var/www/html/index.html` files on the web servers and using `curl` to test connectivity, the following output demonstrate how HAProxy balances the traffic between the servers and how it handles the `httpd` service stopping on `websvr1`:

```
$ while true; do curl http://10.0.0.10; sleep 1; done
This is HTTP server websvr1 (192.168.1.71).
This is HTTP server websvr2 (192.168.1.72).
This is HTTP server websvr1 (192.168.1.71).
This is HTTP server websvr2 (192.168.1.72).
...
This is HTTP server websvr2 (192.168.1.72).
<html><body><h1>503 Service Unavailable</h1>
No server is available to handle this request.
</body></html>
This is HTTP server websvr2 (192.168.1.72).
This is HTTP server websvr2 (192.168.1.72).
This is HTTP server websvr2 (192.168.1.72).
...
This is HTTP server websvr2 (192.168.1.72).
This is HTTP server websvr2 (192.168.1.72).
This is HTTP server websvr2 (192.168.1.72).
This is HTTP server websvr1 (192.168.1.71).
This is HTTP server websvr2 (192.168.1.72).
This is HTTP server websvr1 (192.168.1.71).
...
```

```
^C
$
```

In this example, HAProxy detected that the `httpd` service had restarted on `websvr1` and resumed using that server in addition to `websvr2`.

By combining the load balancing capability of HAProxy with the high availability capability of Keepalived or Oracle Clusterware, you can configure a backup load balancer that ensures continuity of service in the event that the primary load balancer fails. See Making HAProxy Highly Available Using Keepalived and Making HAProxy Highly Available Using Oracle Clusterware.

See Installing and Configuring HAProxy for details of how to install and configure HAProxy.

Configuring HAProxy for Session Persistence

Many web-based application require that a user session is persistently served by the same web server.

If you want web sessions to have persistent connections to the same server, you can use a `balance` algorithm such as `hdr`, `rdp-cookie`, `source`, `uri`, or `url_param`.

If your implementation requires the use of the `leastconn`, `roundrobin`, or `static-rr` algorithm, you can implement session persistence by using server-dependent cookies.

To enable session persistence for all pages on a web server, use the `cookie` directive to define the name of the cookie to be inserted and add the `cookie` option and server name to the `server` lines, for example:

```
cookie WEBSVR insert
server websvr1 192.168.1.71:80 weight 1 maxconn 512 cookie 1 check
server websvr2 192.168.1.72:80 weight 1 maxconn 512 cookie 2 check
```

HAProxy includes an additional `Set-Cookie:` header that identifies the web server in its response to the client, for example: `Set-Cookie: WEBSVR=`*`N`*`; path=`*`page_path`* . If a client subsequently specifies the `WEBSVR` cookie in a request, HAProxy forwards the request to the web server whose `server cookie`value matches the value of `WEBSVR`.

The following example demonstrates how an inserted cookie ensures session persistence:

```
while true; do curl http://10.0.0.10; sleep 1; done

This is HTTP server websvr1 (192.168.1.71).
This is HTTP server websvr2 (192.168.1.72).
This is HTTP server websvr1 (192.168.1.71).

^CC
curl http://10.0.0.10 -D /dev/stdout

HTTP/1.1 200 OK
Date: ...
Server: Apache/2.4.6 ()
Last-Modified: ...
ETag: "26-5125afd089491"
Accept-Ranges: bytes
Content-Length: 38
Content-Type: text/html; charset=UTF-8
Set-Cookie: WEBSVR=2; path=/
```

```
This is HTTP server svr2 (192.168.1.72).

while true; do curl http://10.0.0.10 --cookie "WEBSVR=2;"; sleep 1; done

This is HTTP server websvr2 (192.168.1.72).
This is HTTP server websvr2 (192.168.1.72).
This is HTTP server websvr2 (192.168.1.72).

^C
```

To enable persistence selectively on a web server, use the `cookie` directive to specify that HAProxy should expect the specified cookie, usually a session ID cookie or other existing cookie, to be prefixed with the `server cookie` value and a `~` delimiter, for example:

```
cookie SESSIONID prefix
server websvr1 192.168.1.71:80 weight 1 maxconn 512 cookie 1 check
server websvr2 192.168.1.72:80 weight 1 maxconn 512 cookie 2 check
```

If the value of `SESSIONID` is prefixed with a `server cookie` value, for example: `Set-Cookie: SESSIONID=`*`N~Session_ID`*`;`, HAProxy strips the prefix and delimiter from the `SESSIONID` cookie before forwarding the request to the web server whose `server cookie` value matches the prefix.

The following example demonstrates how using a prefixed cookie enables session persistence:

```
while true; do curl http://10.0.0.10 --cookie "SESSIONID=1~1234;"; sleep 1; done

This is HTTP server websvr1 (192.168.1.71).
This is HTTP server websvr1 (192.168.1.71).
This is HTTP server websvr1 (192.168.1.71).

^C
```

A real web application would usually set the session ID on the server side, in which case the first HAProxy response would include the prefixed cookie in the `Set-Cookie:` header.

About Keepalived

Keepalived uses the IP Virtual Server (IPVS) kernel module to provide transport layer (Layer 4) load balancing, redirecting requests for network-based services to individual members of a server cluster. IPVS monitors the status of each server and uses the Virtual Router Redundancy Protocol (VRRP) to implement high availability.

The configuration file for the `keepalived` daemon is `/etc/keepalived/keepalived.conf`. This file must be present on each server on which you configure Keepalived for load balancing or high availability.

For more information, see https://www.keepalived.org/documentation.html, the `/usr/share/doc/keepalive-`*`version`* documentation, and the `keepalived(8)` and `keepalived.conf(5)` manual pages.

Installing and Configuring Keepalived

To install Keepalived:

1. Install the `keepalived` package on each server:

   ```
   sudo yum install keepalived
   ```

2. Edit `/etc/keepalived/keepalived.conf` to configure Keepalived on each server. See About the Keepalived Configuration File.

3. Enable IP forwarding:

   ```
   sudo echo "net.ipv4.ip_forward = 1" >> /etc/sysctl.conf
   sudo sysctl -p

   net.ipv4.ip_forward = 1
   ```

4. Add firewall rules to allow VRRP communication using the multicast IP address 224.0.0.18 and the VRRP protocol (112) on each network interface that Keepalived will control, for example:

   ```
   sudo firewall-cmd --direct --permanent --add-rule ipv4 filter INPUT 0 \
   --in-interface enp0s8 --destination 224.0.0.18 --protocol vrrp -j ACCEPT

   success

   sudo firewall-cmd --direct --permanent --add-rule ipv4 filter OUTPUT 0 \
   --out-interface enp0s8 --destination 224.0.0.18 --protocol vrrp -j ACCEPT

   success

   sudo firewall-cmd --reload

   success
   ```

5. Enable and start the `keepalived` service on each server:

   ```
   sudo systemctl enable keepalived
   sudo ln -s '/usr/lib/systemd/system/keepalived.service' '/etc/systemd/system/multi-
   user.target.wants/keepalived.service'
   sudo systemctl start keepalived
   ```

   If you change the Keepalived configuration, reload the `keepalived` service:

   ```
   sudo systemctl reload keepalived
   ```

About the Keepalived Configuration File

The `/etc/keepalived/keepalived.conf` configuration file is divided into the following sections:

`global_defs`
Defines global settings such as the email addresses for sending notification messages, the IP address of an SMTP server, the timeout value for SMTP connections in seconds, a string that identifies the host machine, the VRRP IPv4 and IPv6 multicast addresses, and whether SNMP traps should be enabled.

`static_ipaddress`
`static_routes`
Define static IP addresses and routes, which VRRP cannot change. These sections are not required if the addresses and routes are already defined on the servers and these servers already have network connectivity.

`vrrp_sync_group`
Defines a VRRP synchronization group of VRRP instances that fail over together.

`vrrp_instance`
Defines a moveable virtual IP address for a member of a VRRP synchronization group's internal or external network interface, which accompanies other group members during a state transition. Each VRRP instance must have a unique value of `virtual_router_id`, which identifies which interfaces on the primary and backup servers can be assigned a given virtual IP address. You can also specify scripts that are run on state transitions to `BACKUP`, `MASTER`, and `FAULT`, and whether to trigger SMTP alerts for state transitions.

`vrrp_script`
Defines a tracking script that Keepalived can run at regular intervals to perform monitoring actions from a `vrrp_instance` or `vrrp_sync_group` section.

`virtual_server_group`
Defines a virtual server group, which allows a real server to be a member of several virtual server groups.

`virtual_server`
Defines a virtual server for load balancing, which is composed of several real servers.

For examples of how to configure Keepalived, see:

- Configuring Simple Virtual IP Address Failover Using Keepalived
- Configuring Load Balancing Using Keepalived in NAT Mode
- Configuring Load Balancing Using Keepalived in DR Mode
- Making HAProxy Highly Available Using Keepalived

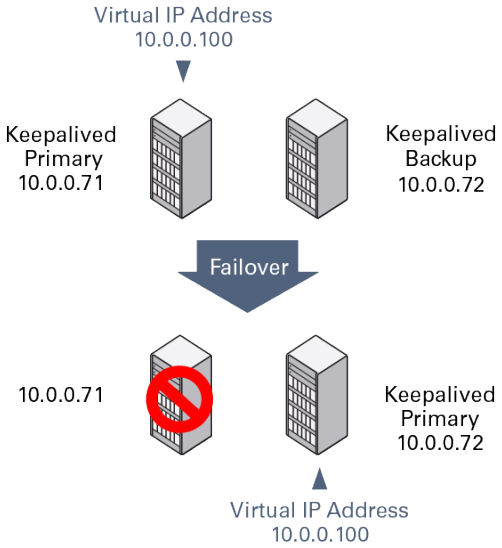
Configuring Simple Virtual IP Address Failover Using Keepalived

A typical Keepalived high-availability configuration consists of one primary server and one or more backup servers. One or more virtual IP addresses, defined as *VRRP instances*, are assigned to the primary server's network interfaces so that it can service network clients. The backup servers listen for multicast VRRP advertisement packets that the primary server transmits at regular intervals. The default advertisement interval is one second. If the backup nodes fail to receive three consecutive VRRP advertisements, the backup server with the highest assigned priority takes over as the primary server and assigns the virtual IP addresses to its own network interfaces. If several backup servers have the same priority, the backup server with the highest IP address value becomes the primary server.

The following example uses Keepalived to implement a simple failover configuration on two servers. One server acts as the primary server and the other acts as a backup. The primary server has a higher priority than the backup server.

The following figure shows how the virtual IP address `10.0.0.100` is initially assigned to the primary server (`10.0.0.71`). When the primary server fails, the backup server (`10.0.0.72`) becomes the new primary server and is assigned the virtual IP address `10.0.0.100`.

Figure 8-2 Example Keepalived Configuration for Virtual IP Address Failover



You might use the following configuration in `/etc/keepalived/keepalived.conf` on the primary (master) server:

```
global_defs {
   notification_email {
     root@mydomain.com
   }
   notification_email_from svr1@mydomain.com
   smtp_server localhost
   smtp_connect_timeout 30
}

vrrp_instance VRRP1 {
    state MASTER
#   Specify the network interface to which the virtual address is assigned
    interface enp0s8
#   The virtual router ID must be unique to each VRRP instance that you define
    virtual_router_id 41
#   Set the value of priority higher on the primary server than on a backup server
    priority 200
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1066
    }
    virtual_ipaddress {
        10.0.0.100/24
    }
}
```

The configuration of the backup server is the same, except for the `notification_email_from`, `state`, `priority`, and possibly `interface` values, if the system hardware configuration is different:

```
global_defs {
   notification_email {
     root@mydomain.com
   }
   notification_email_from svr2@mydomain.com
   smtp_server localhost
   smtp_connect_timeout 30
}

vrrp_instance VRRP1 {
    state BACKUP
#   Specify the network interface to which the virtual address is assigned
    interface enp0s8
    virtual_router_id 41
#   Set the value of priority lower on the backup server than on the primary
server
    priority 100
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1066
    }
    virtual_ipaddress {
        10.0.0.100/24
    }
}
```

In the event that the primary server (`svr1`) fails, `keepalived` assigns the virtual IP address `10.0.0.100/24` to the `enp0s8` interface on the backup server (`svr2`), which becomes the primary server.

To determine whether a server is acting as the primary server, you can use the `ip` command to see whether the virtual address is active, for example:

```
sudo ip addr list enp0s8

3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
qlen 1000
    link/ether 08:00:27:cb:a6:8d brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.72/24 brd 10.0.0.255 scope global enp0s8
    inet 10.0.0.100/24 scope global enp0s8
    inet6 fe80::a00:27ff:fecb:a68d/64 scope link
       valid_lft forever preferred_lft forever
```

Alternatively, search for Keepalived messages in `/var/log/messages` that show transitions between states, for example:

```
...51:55 ... VRRP_Instance(VRRP1) Entering BACKUP STATE
...
...53:08 ... VRRP_Instance(VRRP1) Transition to MASTER STATE
...53:09 ... VRRP_Instance(VRRP1) Entering MASTER STATE
...53:09 ... VRRP_Instance(VRRP1) setting protocol VIPs.
...53:09 ... VRRP_Instance(VRRP1) Sending gratuitous ARPs on enp0s8 for
10.0.0.100
```

> **Note:**
>
> Only one server should be active as the primary (master) server at any time. If more than one server is configured as the primary server, it is likely that there is a problem with VRRP communication between the servers. Check the network settings for each interface on each server and check that the firewall allows both incoming and outgoing VRRP packets for multicast IP address `224.0.0.18`.

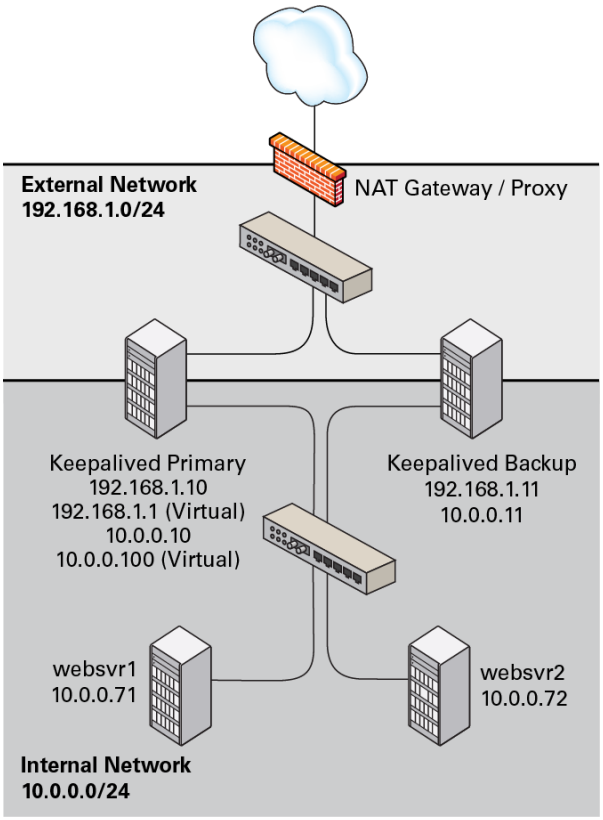
See Installing and Configuring Keepalived for details of how to install and configure Keepalived.

Configuring Load Balancing Using Keepalived in NAT Mode

The following example uses Keepalived in NAT mode to implement a simple failover and load balancing configuration on two servers. One server acts as the primary server and the other acts as a backup. The primary server has a higher priority than the backup server. Each of the servers has two network interfaces, where one interface is connected to the side facing an external network (`192.168.1.0/24`) and the other interface is connected to an internal network (`10.0.0.0/24`) on which two web servers are accessible.

The following figure shows that the Keepalived primary server has the network addresses `192.168.1.10`, `192.168.1.1` (virtual), `10.0.0.10`, and `10.0.0.100` (virtual). The Keepalived backup server has the network addresses `192.168.1.11` and `10.0.0.11`. The web servers, `websvr1` and `websvr2`, have the network addresses `10.0.0.71` and `10.0.0.72`, respectively.

Figure 8-3 Example Keepalived Configuration for Load Balancing in NAT Mode



You might use the following configuration in `/etc/keepalived/keepalived.conf` on the primary server:

```
global_defs {
   notification_email {
     root@mydomain.com
   }
   notification_email_from svr1@mydomain.com
   smtp_server localhost
   smtp_connect_timeout 30
}

vrrp_sync_group VRRP1 {
#    Group the external and internal VRRP instances so they fail over together
     group {
         external
         internal
         }
}

vrrp_instance external {
     state MASTER
     interface enp0s8
     virtual_router_id 91
     priority 200
     advert_int 1
     authentication {
         auth_type PASS
```

```
        auth_pass 1215
    }
#   Define the virtual IP address for the external network interface
    virtual_ipaddress {
        192.168.1.1/24
    }
}

vrrp_instance internal {
    state MASTER
    interface enp0s9
    virtual_router_id 92
    priority 200
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1215
    }
#   Define the virtual IP address for the internal network interface
    virtual_ipaddress {
        10.0.0.100/24
    }
}

# Define a virtual HTTP server on the virtual IP address 192.168.1.1
virtual_server 192.168.1.1 80 {
    delay_loop 10
    protocol TCP
#   Use round-robin scheduling in this example
    lb_algo rr
#   Use NAT to hide the back-end servers
    lb_kind NAT
#   Persistence of client sessions times out after 2 hours
    persistence_timeout 7200

    real_server 10.0.0.71 80 {
        weight 1
        TCP_CHECK {
          connect_timeout 5
          connect_port 80
        }
    }

    real_server 10.0.0.72 80 {
        weight 1
        TCP_CHECK {
          connect_timeout 5
          connect_port 80
        }
    }
}
```

The previous configuration is similar to the configuration shown in Configuring Simple Virtual IP Address Failover Using Keepalived, with the additional definition of a `vrrp_sync_group` section so that the network interfaces are assigned together on failover, as well as a `virtual_server` section to define the real back-end servers that Keepalived uses for load balancing. The value of `lb_kind` is set to NAT mode, which means that the Keepalived server handles both inbound and outbound network traffic for the client on behalf of the back-end servers.

The configuration of the backup server is the same, except for the `notification_email_from`, `state`, `priority`, and possibly `interface` values, if the system hardware configuration is different:

```
global_defs {
   notification_email {
     root@mydomain.com
   }
   notification_email_from svr2@mydomain.com
   smtp_server localhost
   smtp_connect_timeout 30
}

vrrp_sync_group VRRP1 {
#    Group the external and internal VRRP instances so they fail over together
    group {
        external
        internal
        }
}

vrrp_instance external {
    state BACKUP
    interface enp0s8
    virtual_router_id 91
    priority 100
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1215
    }
#    Define the virtual IP address for the external network interface
    virtual_ipaddress {
        192.168.1.1/24
    }
}

vrrp_instance internal {
    state BACKUP
    interface enp0s9
    virtual_router_id 92
    priority 100
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1215
    }
#    Define the virtual IP address for the internal network interface
    virtual_ipaddress {
        10.0.0.100/24
    }
}

# Define a virtual HTTP server on the virtual IP address 192.168.1.1
virtual_server 192.168.1.1 80 {
    delay_loop 10
    protocol TCP
#    Use round-robin scheduling in this example
    lb_algo rr
#    Use NAT to hide the back-end servers
    lb_kind NAT
```

```
#   Persistence of client sessions times out after 2 hours
    persistence_timeout 7200

    real_server 10.0.0.71 80 {
        weight 1
        TCP_CHECK {
          connect_timeout 5
          connect_port 80
        }
    }

    real_server 10.0.0.72 80 {
        weight 1
        TCP_CHECK {
          connect_timeout 5
          connect_port 80
        }
    }
}
```

Two further configuration changes are required:

- Configure firewall rules on each Keepalived server (primary and backup) that you configure as a load balancer as described in Configuring Firewall Rules for Keepalived NAT-Mode Load Balancing.
- Configure a default route for the virtual IP address of the load balancer's internal network interface on each back-end server that you intend to use with the Keepalived load balancer as described in Configuring Back-End Server Routing for Keepalived NAT-Mode Load Balancing.

See Installing and Configuring Keepalived for details of how to install and configure Keepalived.

Configuring Firewall Rules for Keepalived NAT-Mode Load Balancing

If you configure Keepalived to use NAT mode for load balancing with the servers on the internal network, the Keepalived server handles all inbound and outbound network traffic and hides the existence of the back-end servers by rewriting the source IP address of the real back-end server in outgoing packets with the virtual IP address of the external network interface.

To configure a Keepalived server to use NAT mode for load balancing:

1. Configure the firewall so that the interfaces on the external network side are in a different zone from the interfaces on the internal network side.

   The following example demonstrates how to move interface `enp0s9` to the `internal` zone while interface `enp0s8` remains in the `public` zone:

   ```
   sudo firewall-cmd --get-active-zones

   public
     interfaces: enp0s8 enp0s9

   sudo firewall-cmd --zone=public --remove-interface=enp0s9

   success

   sudo firewall-cmd --zone=internal --add-interface=enp0s9
   ```

```
success

sudo firewall-cmd --permanent --zone=public --remove-interface=enp0s9

success

sudo firewall-cmd --permanent --zone=internal --add-interface=enp0s9

success

sudo firewall-cmd --get-active-zones

internal
  interfaces: enp0s9
public
  interfaces: enp0s8
```

2. Configure NAT mode (masquerading) on the external network interface, for example:

```
sudo firewall-cmd --zone=public --add-masquerade

success

sudo firewall-cmd --permanent --zone=public --add-masquerade

success

sudo firewall-cmd --zone=public --query-masquerade

yes

sudo firewall-cmd --zone=internal --query-masquerade

no
```

3. If not already enabled for your firewall, configure forwarding rules between the external and internal network interfaces, for example:

```
sudo firewall-cmd --direct --permanent --add-rule ipv4 filter FORWARD 0 \
-i enp0s8 -o enp0s9 -m state --state RELATED,ESTABLISHED -j ACCEPT

success

sudo firewall-cmd --direct --permanent --add-rule ipv4 filter FORWARD 0 \
-i enp0s9 -o enp0s8 -j ACCEPT

success

sudo firewall-cmd --direct --permanent --add-rule ipv4 filter FORWARD 0 \
-j REJECT --reject-with icmp-host-prohibited

success

sudo firewall-cmd --reload
```

4. Enable access to the services or ports that you want Keepalived to handle.

   For example, to enable access to HTTP and make this rule persist across reboots, enter the following commands:

```
sudo firewall-cmd --zone=public --add-service=http
```

```
success

sudo firewall-cmd --permanent --zone=public --add-service=http

success
```

Configuring Back-End Server Routing for Keepalived NAT-Mode Load Balancing

On each back-end real servers that you intend to use with the Keepalived load balancer, ensure that the routing table contains a default route for the virtual IP address of the load balancer's internal network interface.

For example, if the virtual IP address is `10.0.0.100`, you can use the `ip` command to examine the routing table and to set the default route:

```
sudo ip route show

10.0.0.0/24 dev enp0s8  proto kernel  scope link  src 10.0.0.71

sudo ip route add default via 10.0.0.100 dev enp0s8
sudo ip route show

default via 10.0.0.100 dev enp0s8
10.0.0.0/24 dev enp0s8  proto kernel  scope link  src 10.0.0.71
```

To make the default route for `enp0s8` persist across reboots, create the file `/etc/sysconfig/network-scripts/route-enp0s8`:

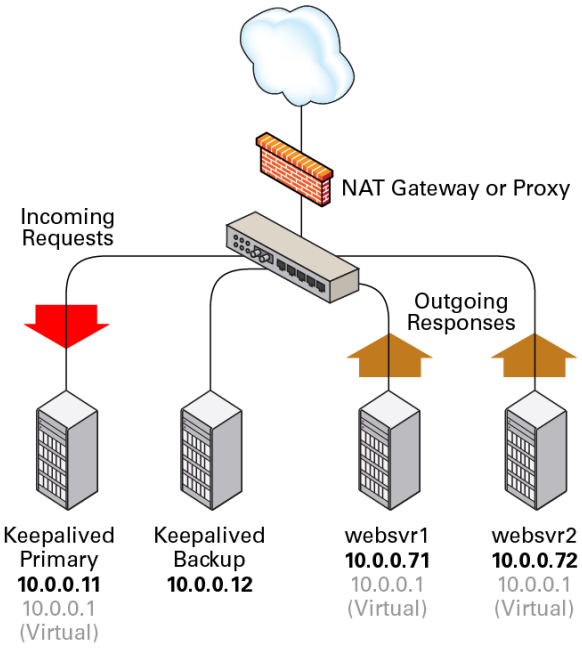
```
sudo echo "default via 10.0.0.100 dev enp0s8" > /etc/sysconfig/network-scripts/route-enp0s8
```

Configuring Load Balancing Using Keepalived in DR Mode

The following example uses Keepalived in direct routing (DR) mode to implement a simple failover and load balancing configuration on two servers. One server acts as the primary server and the other acts as a backup. The primary server has a higher priority than the backup server. Each of Keepalived servers has a single network interface and the servers are connected to the same network segment (10.0.0.0/24) on which two web servers are accessible.

Figure 8-4 shows that the Keepalived primary server has network the addresses `10.0.0.11` and `10.0.0.1` (virtual). The Keepalived backup server has the network address `10.0.0.12`. The web servers, `websvr1` and `websvr2`, have the network addresses `10.0.0.71` and `10.0.0.72`, respectively. In addition, both web servers are configured with the virtual IP address `10.0.0.1` so that they accept packets with that destination address. Incoming requests are received by the primary server and redirected to the web servers, which respond directly.

Figure 8-4 Example Keepalived Configuration for Load Balancing in DR Mode



You might use the following configuration in `/etc/keepalived/keepalived.conf` on the primary server:

```
global_defs {
   notification_email {
     root@mydomain.com
   }
   notification_email_from svr1@mydomain.com
   smtp_server localhost
   smtp_connect_timeout 30
}

vrrp_instance external {
    state MASTER
    interface enp0s8
    virtual_router_id 91
    priority 200
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1215
    }
    virtual_ipaddress {
        10.0.0.1/24
    }
}

virtual_server 10.0.0.1 80 {
    delay_loop 10
    protocol TCP
    lb_algo rr
#   Use direct routing
    lb_kind DR
    persistence_timeout 7200
```

```
    real_server 10.0.0.71 80 {
        weight 1
        TCP_CHECK {
          connect_timeout 5
          connect_port 80
        }
    }

    real_server 10.0.0.72 80 {
        weight 1
        TCP_CHECK {
          connect_timeout 5
          connect_port 80
        }
    }
}
```

The virtual server configuration is similar to that given in Configuring Load Balancing Using Keepalived in NAT Mode except that the value of `lb_kind` is set to `DR` (Direct Routing), which means that the Keepalived server handles all inbound network traffic from the client before routing it to the back-end servers, which reply directly to the client, bypassing the Keepalived server. This configuration reduces the load on the Keepalived server but is less secure as each back-end server requires external access and is potentially exposed as an attack surface. Some implementations use an additional network interface with a dedicated gateway for each web server to handle the response network traffic.

The configuration of the backup server is the same, except for the `notification_email_from`, `state`, `priority`, and possibly `interface` values, if the system hardware configuration is different:

```
global_defs {
   notification_email {
     root@mydomain.com
   }
   notification_email_from svr2@mydomain.com
   smtp_server localhost
   smtp_connect_timeout 30
}

vrrp_instance external {
    state BACKUP
    interface enp0s8
    virtual_router_id 91
    priority 100
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1215
    }
    virtual_ipaddress {
        10.0.0.1/24
    }
}

virtual_server 10.0.0.1 80 {
    delay_loop 10
    protocol TCP
    lb_algo rr
#   Use direct routing
```

```
    lb_kind DR
    persistence_timeout 7200

    real_server 10.0.0.71 80 {
        weight 1
        TCP_CHECK {
          connect_timeout 5
          connect_port 80
        }
    }

    real_server 10.0.0.72 80 {
        weight 1
        TCP_CHECK {
          connect_timeout 5
          connect_port 80
        }
    }
}
```

Two further configuration changes are required:

- Configure firewall rules on each Keepalived server (primary and backup) that you configure as a load balancer as described in Configuring Firewall Rules for Keepalived DR-Mode Load Balancing.
- Configure the `arp_ignore` and `arp_announce` ARP parameters and the virtual IP address for the network interface on each back-end server that you intend to use with the Keepalived load balancer as described in Configuring the Back-End Servers for Keepalived DR-Mode Load Balancing.

See Installing and Configuring Keepalived for details of how to install and configure Keepalived.

Configuring Firewall Rules for Keepalived DR-Mode Load Balancing

Enable access to the services or ports that you want Keepalived to handle.

For example, to enable access to HTTP and make this rule persist across reboots, enter the following commands:

```
sudo firewall-cmd --zone=public --add-service=http

success

sudo firewall-cmd --permanent --zone=public --add-service=http

success
```

Configuring the Back-End Servers for Keepalived DR-Mode Load Balancing

The example configuration requires that the virtual IP address is configured on the primary Keepalived server and on each back-end server. The Keepalived configuration maintains the virtual IP address on the primary Keepalived server.

Only the primary Keepalived server should respond to ARP requests for the virtual IP address. You can set the `arp_ignore` and `arp_announce` ARP parameters for the

network interface of each back-end server so that they do not respond to ARP requests for the virtual IP address.

To configure the ARP parameters and virtual IP address on each back-end server:

1. Configure the ARP parameters for the primary network interface, for example `enp0s8`:

```
sudo echo "net.ipv4.conf.enp0s8.arp_ignore = 1" >> /etc/sysctl.conf
sudo echo "net.ipv4.conf.enp0s8.arp_announce = 2" >> /etc/sysctl.conf
sudo sysctl -p

net.ipv4.conf.enp0s8.arp_ignore = 1
net.ipv4.conf.enp0s8.arp_announce = 2
```

2. To define a virtual IP address that persists across reboots, edit `/etc/sysconfig/network-scripts/ifcfg-`*`iface`* and add `IPADDR1` and `PREFIX1` entries for the virtual IP address, for example:

```
...
NAME=enp0s8
...
IPADDR0=10.0.0.72
GATEWAY0=10.0.0.100
PREFIX0=24
IPADDR1=10.0.0.1
PREFIX1=24
...
```

   This example defines the virtual IP address `10.0.0.1` for `enp0s8` in addition to the existing real IP address of the back-end server.

3. Reboot the system and verify that the virtual IP address has been set up:

```
sudo ip addr show enp0s8

2: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
qlen 1000
    link/ether 08:00:27:cb:a6:8d brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.72/24 brd 10.0.0.255 scope global enp0s8
    inet 10.0.0.1/24 brd 10.0.0.255 scope global secondary enp0s8
    inet6 fe80::a00:27ff:fecb:a68d/64 scope link
       valid_lft forever preferred_lft forever
```

Configuring Keepalived for Session Persistence and Firewall Marks

Many web-based application require that a user session is persistently served by the same web server.

If you enable the load balancer in Keepalived to use persistence, a client connects to the same server provided that the timeout period (`persistence_timeout`) has not been exceeded since the previous connection.

Firewall marks are another method for controlling session access so that Keepalived forwards a client's connections on different ports, such as HTTP (80) and HTTPS (443), to the same server, for example:

```
sudo firewall-cmd --direct --permanent --add-rule ipv4 mangle PREROUTING 0 \
-d virtual_IP_addr/32 -p tcp -m multiport --dports 80,443 -j MARK --set-mark 123
```

```
success

sudo firewall-cmd --reload
```

These commands set a firewall mark value of 123 on packets that are destined for ports 80 or 443 at the specified virtual IP address.

You must also declare the firewall mark (`fwmark`) value to Keepalived by setting it on the virtual server instead of a destination virtual IP address and port, for example:

```
virtual_server fwmark 123 {
   ...
}
```

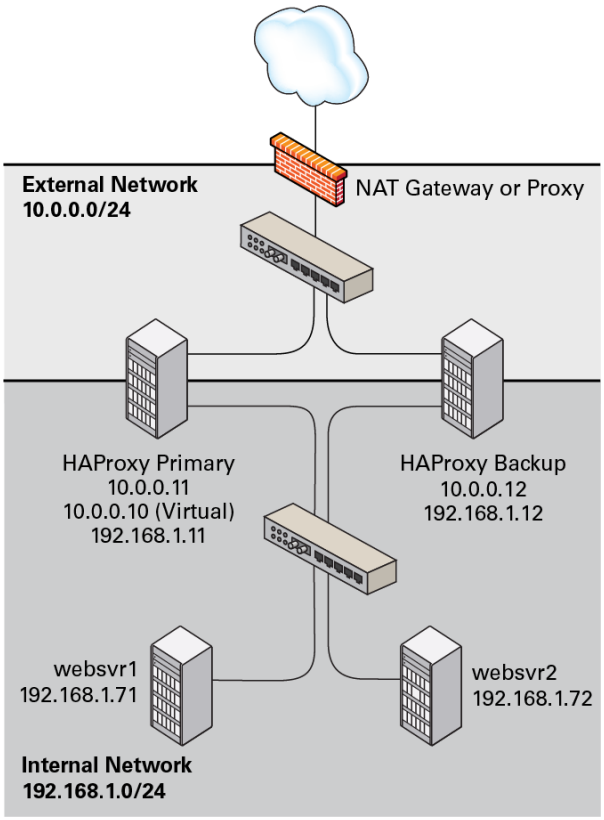
This configuration causes Keepalived to route the packets based on their firewall mark value rather than the destination virtual IP address and port. When used in conjunction with session persistence, firewall marks help ensure that all ports used by a client session are handled by the same server.

Making HAProxy Highly Available Using Keepalived

The following example uses Keepalived to make the HAProxy service fail over to a backup server in the event that the primary server fails.

The following figure shows two HAProxy servers, which are connected to an externally facing network (`10.0.0.0/24`), as `10.0.0.11` and `10.0.0.12`, and to an internal network (`192.168.1.0/24`), as `192.168.1.11` and `192.168.1.12`. One HAProxy server (`10.0.0.11`) is configured as a Keepalived primary server with the virtual IP address `10.0.0.10` and the other (`10.0.0.12`) is configured as a Keepalived backup server. Two web servers, `websvr1` (`192.168.1.71`), and `websvr2` (`192.168.1.72`), are accessible on the internal network. The IP address `10.0.0.10` is in the private address range `10.0.0.0/24`, which cannot be routed on the Internet. An upstream network address translation (NAT) gateway or a proxy server provides access to and from the Internet.

Figure 8-5 Example of a Combined HAProxy and Keepalived Configuration with Web Servers on a Separate Network



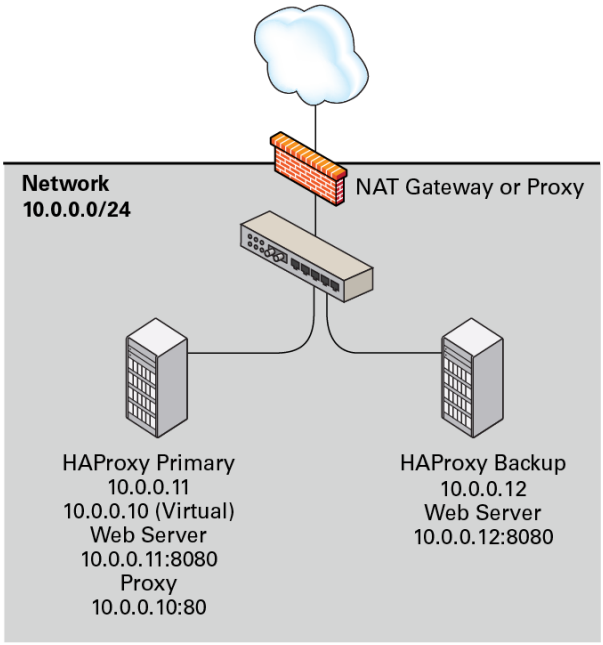
The HAProxy configuration on both `10.0.0.11` and `10.0.0.12` is very similar to Configuring Simple Load Balancing Using HAProxy. The IP address on which HAProxy listens for incoming requests is the virtual IP address that Keepalived controls.

```
global
    daemon
    log 127.0.0.1 local0 debug
    maxconn 50000
    nbproc 1

defaults
    mode http
    timeout connect 5s
    timeout client 25s
    timeout server 25s
    timeout queue 10s

# Handle Incoming HTTP Connection Requests on the virtual IP address controlled by
Keepalived
listen  http-incoming
    mode http
    bind 10.0.0.10:80
# Use each server in turn, according to its weight value
    balance roundrobin
# Verify that service is available
    option httpchk OPTIONS * HTTP/1.1\r\nHost:\ www
# Insert X-Forwarded-For header
```

```
    option forwardfor
# Define the back-end servers, which can handle up to 512 concurrent connections
each
    server websvr1 192.168.1.71:80 weight 1 maxconn 512 check
    server websvr2 192.168.1.72:80 weight 1 maxconn 512 check
```

It is also possible to configure HAProxy and Keepalived directly on the web servers as shown in the following figure. As in the previous example, one HAProxy server (`10.0.0.11`) is configured as the Keepalived primary server, with the virtual IP address `10.0.0.10`, and the other (`10.0.0.12`) is configured as a Keepalived backup server. The HAProxy service on the primary server listens on port 80 and forwards incoming requests to one of the `httpd` services, which listen on port 8080.

Figure 8-6 Example of a Combined HAProxy and Keepalived Configuration with Integrated Web Servers



The HAProxy configuration is the same as the previous example, except for the IP addresses and ports of the web servers.

```
...
    server websvr1 10.0.0.11:8080 weight 1 maxconn 512 check
    server websvr2 10.0.0.12:8080 weight 1 maxconn 512 check
```

The firewall on each server must be configured to accept incoming TCP requests on port 8080.

The Keepalived configuration for both example configurations is similar to that given in Configuring Simple Virtual IP Address Failover Using Keepalived.

The primary (master) server has the following Keepalived configuration:

```
global_defs {
   notification_email {
     root@mydomain.com
   }
```

```
   notification_email_from haproxy1@mydomain.com
   smtp_server localhost
   smtp_connect_timeout 30
}

vrrp_instance VRRP1 {
    state MASTER
#   Specify the network interface to which the virtual address is assigned
    interface enp0s8
#   The virtual router ID must be unique to each VRRP instance that you define
    virtual_router_id 41
#   Set the value of priority higher on the primary server than on a backup server
    priority 200
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1066
    }
    virtual_ipaddress {
        10.0.0.10/24
    }
}
```

The configuration of the backup server is the same, except for the `notification_email_from`, `state`, `priority`, and possibly `interface` values, if the system hardware configuration is different:

```
global_defs {
   notification_email {
     root@mydomain.com
   }
   notification_email_from haproxy2@mydomain.com
   smtp_server localhost
   smtp_connect_timeout 30
}

vrrp_instance VRRP1 {
    state BACKUP
#   Specify the network interface to which the virtual address is assigned
    interface enp0s8
    virtual_router_id 41
#   Set the value of priority lower on the backup server than on the primary server
    priority 100
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1066
    }
    virtual_ipaddress {
        10.0.0.10/24
    }
}
```

In the event that the primary server (`haproxy1`) fails, `keepalived` assigns the virtual IP address `10.0.0.10/24` to the `enp0s8` interface on the backup server (`haproxy2`), which then becomes the primary server.

See Installing and Configuring HAProxy and Installing and Configuring Keepalived for details on how to install and configure HAProxy and Keepalived.

About Keepalived Notification and Tracking Scripts

Notification scripts are executable programs that Keepalived invokes when a server changes state. You can implements notification scripts to perform actions such as reconfiguring a network interface or starting, reloading or stopping a service.

To invoke a notification script, include one the following lines inside a `vrrp_instance` or `vrrp_sync_group` section:

`notify` *program_path*
Invokes *program_path* with the following arguments:

`$1`
Set to `INSTANCE` or `GROUP`, depending on whether Keepalived invoked the program from `vrrp_instance` or `vrrp_sync_group`.

`$2`
Set to the name of the `vrrp_instance` or `vrrp_sync_group`.

`$3`
Set to the end state of the transition: `BACKUP`, `FAULT`, or `MASTER`.

`notify_backup` *program_path*
`notify_backup"`*program_patharg* `...``"`
Invokes *program_path* when the end state of a transition is `BACKUP`. *program_path* is the full pathname of an executable script or binary. If a program has arguments, enclose both the program path and the arguments in quotes.

`notify_fault` *program_path*
`notify_fault"`*program_patharg* `...``"`
Invokes *program_path* when the end state of a transition is `FAULT`.

`notify_master` *program_path*
`notify_master"`*program_patharg* `...``"`
Invokes *program_path* when the end state of a transition is `MASTER`.

The following executable script could be used to handle the general-purpose version of `notify`:

```
#!/bin/bash

ENDSTATE=$3
NAME=$2
TYPE=$1

case $ENDSTATE in
    "BACKUP") # Perform action for transition to BACKUP state
              exit 0
              ;;
    "FAULT")  # Perform action for transition to FAULT state
              exit 0
              ;;
    "MASTER") # Perform action for transition to MASTER state
              exit 0
              ;;
    *)        echo "Unknown state ${ENDSTATE} for VRRP ${TYPE} ${NAME}"
```

```
            exit 1
            ;;
esac
```

Tracking scripts are programs that Keepalived runs at regular intervals according to a `vrrp_script` definition:

```
vrrp_script script_name {
  script       "program_path
                  arg ..."
  interval i  # Run script every i seconds
  fall f      # If script returns non-zero f times in succession, enter FAULT state
  rise r      # If script returns zero r times in succession, exit FAULT state
  timeout t   # Wait up to t seconds for script before assuming non-zero exit code
  weight w    # Reduce priority by w on fall
}
```

In the example, *program_path* is the full pathname of an executable script or binary.

You can use tracking scripts with a `vrrp_instance` section by specifying a `track_script` clause, for example:

```
vrrp_instance instance_name {
  state MASTER
  interface enp0s8
  virtual_router_id 21
  priority 200
  advert_int 1
  virtual_ipaddress {
    10.0.0.10/24
  }
  track_script {
    script_name
    ...
  }
}
```

If a configured script returns a non-zero exit code *f* times in succession, Keepalived changes the state of the VRRP instance or group to `FAULT`, removes the virtual IP address `10.0.0.10` from `enp0s8`, reduces the priority value by *w* and stops sending multicast VRRP packets. If the script subsequently returns a zero exit code *r* times in succession, the VRRP instance or group exits the `FAULT` state and transitions to the `MASTER` or `BACKUP` state depending on its new priority.

If you want a server to enter the `FAULT` state if one or more interfaces goes down, you can also use a `track_interface` clause, for example:

```
  track_interface {
    enp0s8
    enp0s9
  }
```

A possible application of tracking scripts is to deal with a potential split-brain condition in the case that some of the Keepalived servers lose communication. For example, a script could track the existence of other Keepalived servers or use shared storage or a backup communication channel to implement a voting mechanism. However, configuring Keepalived to avoid a split brain condition is complex and it is difficult to avoid corner cases where a scripted solution might not work.

For an alternative solution, see Making HAProxy Highly Available Using Oracle Clusterware.

Making HAProxy Highly Available Using Oracle Clusterware

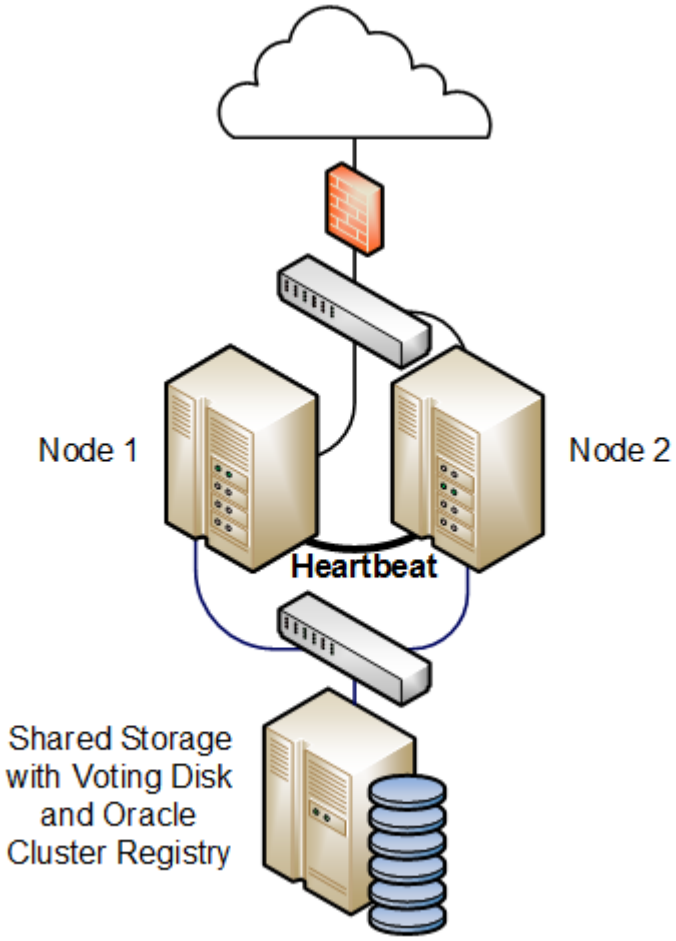
When Keepalived is used with two or more servers, loss of network connectivity can result in a split-brain condition, where more than one server acts as the primary server and which can result in data corruption. To avoid this scenario, Oracle recommends that you use HAProxy in conjunction with a *shoot the other node in the head* (STONITH) solution such as Oracle Clusterware to support virtual IP address failover in preference to Keepalived.

Oracle Clusterware is a portable clustering software solution that allow you to configure independent servers so that they cooperate as a single cluster. The individual servers within the cluster cooperate so that they appear to be a single server to external client applications.

The following example uses Oracle Clusterware with HAProxy for load balancing to HTTPD web server instances on each cluster node. In the event that the node running HAProxy and an HTTPD instance fails, the services and their virtual IP addresses fail over to the other cluster node.

The follow figure shows two cluster nodes that are connected to an externally facing network. The nodes are also linked by a private network that is used for the cluster heartbeat. The nodes have shared access to certified SAN or NAS storage that holds the voting disk and Oracle Cluster Registry (OCR) in addition to service configuration data and application data.

Figure 8-7 Example of an Oracle Clusterware Configuration with Two Nodes



For a high availability configuration, Oracle recommends that the network, heartbeat, and storage connections are multiply redundant and that at least three voting disks are configured.

The following steps describe how to configure this type of cluster:

1. Install Oracle Clusterware on each system that will serve as a cluster node.
2. Install the `haproxy` and `httpd` packages on each node.
3. Use the `appvipcfg` command to create a virtual IP address for HAProxy and a separate virtual IP address for each HTTPD service instance. For example, if there are two HTTPD service instances, you would need to create three different virtual IP addresses.
4. Implement cluster scripts to start, stop, clean, and check the HAProxy and HTTPD services on each node. These scripts must return 0 for success and 1 for failure.
5. Use the shared storage to share the configuration files, HTML files, logs, and all directories and files that the HAProxy and HTTPD services on each node require to start.

   If you have an Oracle Linux Support subscription, you can use OCFS2 or ASM/ACFS with the shared storage as an alternative to NFS or other type of shared file system.

6. Configure each HTTPD service instance so that it binds to the correct virtual IP address. Each service instance must also have an independent set of configuration, log, and other required files, so that all of the service instances can coexist on the same server if one node fails.
7. Use the `crsctl` command to create a cluster resource for HAProxy and for each HTTPD service instance. If there are two or more HTTPD service instances, binding of these instances should initially be distributed amongst the cluster nodes. The HAProxy service can be started on either node initially.

You can use Oracle Clusterware as the basis of a more complex solution that protects a multi-tiered system consisting of front-end load balancers, web servers, database servers and other components.

For more information, see the Oracle Clusterware 11g Administration and Deployment Guide and the Oracle Clusterware 12c Administration and Deployment Guide.

9
Configuring the VNC Service

This chapter describes how to enable a Virtual Network Computing (VNC) server to provide remote access to a graphical desktop.

About VNC

Virtual Network Computing (VNC) is a system for sharing a graphical desktop over a network. A VNC client (the "viewer") connects to, and can control, a desktop that is shared by a VNC server on a remote system. Because VNC is platform independent, you can use any operating system with a VNC client to connect to a VNC server. VNC makes remote administration using graphical tools possible.

By default, all communication between a VNC client and a VNC server is not secure. You can secure VNC communication by using an SSH tunnel. Using an SSH tunnel also reduces the number of firewall ports that need to be open. Oracle recommends that you use SSH tunnels.

Configuring a VNC Server

To configure a VNC server:

1. Install the `tigervnc-server` package:

   ```
   sudo yum install tigervnc-server
   ```

2. Create the VNC environment for the VNC users.

   Each VNC desktop on the system runs a VNC server as a particular user. This user must be able to log in to the system with a user name and either a password or an SSH key (if the VNC desktop is to be accessed through an SSH tunnel).

   Use the `vncpasswd` command to create a password for the VNC desktop. The password must be created by the user that runs the VNC server and not `root`, for example:

   ```
   su - vncuser
   vncpasswd
   Password: password
   Verify: password
   ```

   The password must contain at least six characters. If the password is longer than eight characters, only the first eight characters are used for authentication. An obfuscated version of the password is stored in `$HOME/.vnc/passwd` unless the name of a file is specified with the `vncpasswd` command.

3. Create a service unit configuration file for each VNC desktop that is to be made available on the system.

   a. Copy the `vncserver@.service` template file, for example:

      ```
      cp /lib/systemd/system/vncserver@.service /etc/systemd/system/
      vncserver@\:display.service
      ```

In the previous command, *display* is the unique display number of the VNC desktop, starting from `1`. Use a backslash character (`\`) to escape the colon (`:`) character.

Each VNC desktop is associated with a user account. For ease of administration if you have multiple VNC desktops, you can include the name of the VNC user in the name of the service unit configuration file, for example:

```
cp /lib/systemd/system/vncserver@.service /etc/systemd/system/vncserver-
vncuser@\:display.service
```

b. Edit the service unit configuration files.

The following sections in the configuration file should resemble the sample entries. Replace *vncuser* with the actual VNC user name.

```
[Service]
Type=forking
WorkingDirectory=/home/vncuser
User=vncuser
Group=vncuser

# Clean any existing files in /tmp/.X11-unix environment
ExecStartPre=/bin/sh -c '/usr/bin/vncserver -kill %i > /dev/null 2>&1
|| :'
ExecStart=/usr/bin/vncserver %i
PIDFile=/home/vncuser/.vnc/%H%i.pid
ExecStop=/usr/bin/vncserver -kill %i
```

Optionally, you can add command-line arguments for the VNC server. In the following example, the VNC server only accepts connections from `localhost`, which means the VNC desktop can only be accessed locally or through an SSH tunnel; and the size of the window has been changed from the default 1024x768 to 640x480 using the `geometry` flag:

```
ExecStart=/usr/bin/vncserver %i -localhost -geometry 640x480
PIDFile=/home/vncuser/.vnc/%H%i.pid
```

4. Start the VNC desktops.

a. Make `systemd` reload its configuration files:

```
sudo systemctl daemon-reload
```

b. For each VNC desktop, start the service, and configure the service to start after a system reboot. Remember to use the username and the display number that you specified in the service unit configuration file to be associated with that service. For example:

```
sudo systemctl start vncserver-vncuser@\:display.service
sudo systemctl enable vncserver-vncuser@\:display.service
```

> **Note:**
>
> If you make any changes to a service unit configuration file, you must reload the configuration file and restart the service.

5. Configure the firewall to allow access to the VNC desktops.

If users will access the VNC desktops through an SSH tunnel and the SSH service is enabled on the system, you do not need to open additional ports in the firewall. SSH is enabled by default. For information on enabling SSH, see Oracle® Linux: Connecting to Remote Systems With OpenSSH.

If users will access the VNC desktops directly, you must open the required port for each desktop. The required ports can be calculated by adding the VNC desktop service display number to 5900 (the default VNC server port). So if the display number is 1, the required port is 5901 and if the display number is 67, the required port is 5967.

To open ports 5900 to 5903, you can use the following commands:

```
sudo firewall-cmd --zone=zone --add-service=vnc-server
sudo firewall-cmd --zone=zone --add-service=vnc-server --permanent
```

To open additional ports, for example port 5967, use the following commands:

```
sudo firewall-cmd --zone=zone --add-port=5967/tcp
sudo firewall-cmd --zone=zone --add-port=5967/tcp --permanent
```

6. Configure the VNC desktops.

   By default, the VNC server runs the user's default desktop environment. This is controlled by the VNC user's `$HOME/.vnc/xstartup` file, which is created automatically when the VNC desktop service is started.

   If you did not install a desktop environment when you installed the system (for example because you selected Minimal Install as the base environment), you can install one with the following command:

   ```
   sudo yum groupinstall "server with gui"
   ```

   When the installation is complete, use the `systemctl get-default` command to check that the default system state is `multi-user.target` (multi-user command-line environment). Use the `systemctl set-default` command reset the default system state or to change it to the `graphical.target` (multi-user graphical environment) if you prefer.

   The `$HOME/.vnc/xstartup` file is a shell script that specifies the X applications to run when the VNC desktop is started. For example, to run a KDE Plasma Workspace, you could edit the file as follows:

   ```
   #!/bin/sh
   unset SESSION_MANAGER
   unset DBUS_SESSION_BUS_ADDRESS
   #exec /etc/X11/xinit/xinitrc
   startkde &
   ```

   If you make any changes to a user's `$HOME/.vnc/xstartup` file, you must restart the VNC desktop for the changes to take effect:

   ```
   sudo systemctl restart vncserver-vncuser@\:display.service
   ```

See the `vncserver(1)`, `Xvnc(1)`, and `vncpasswd(1)` manual pages for more information.

Connecting to VNC Desktop

You can connect to a VNC desktop on an Oracle Linux 7 system using any VNC client. The following example instructions are for the TigerVNC client. Adapt the instructions for your client.

1. Install the TigerVNC client (`vncviewer`).

   ```
   sudo yum install tigervnc
   ```

2. Start the TigerVNC client and connect to a desktop.

   To connect directly to a VNC desktop, you can start the TigerVNC client and enter *`host`*`:`*`display`* to specify the host name or IP address of the VNC server and the display number of the VNC desktop to connect to. Alternatively, you can specify the VNC desktop as an argument for the `vncviewer` command. For example:

   ```
   vncviewer myhost.example.com:1
   ```

   To connect to a VNC desktop through an SSH tunnel, use the `-via` option for the `vncviewer` command to specify the user name and host for the SSH connection, and use `localhost:`*`display`* to specify the VNC desktop. For example:

   ```
   vncviewer -via vncuser@myhost.example.com localhost:67
   ```

   See the `vncviewer(1)` manual page for more information.