

Enterprise Networking with RHEL 9

CINLUG
Central Indiana Linux Users Group

June 5 2024

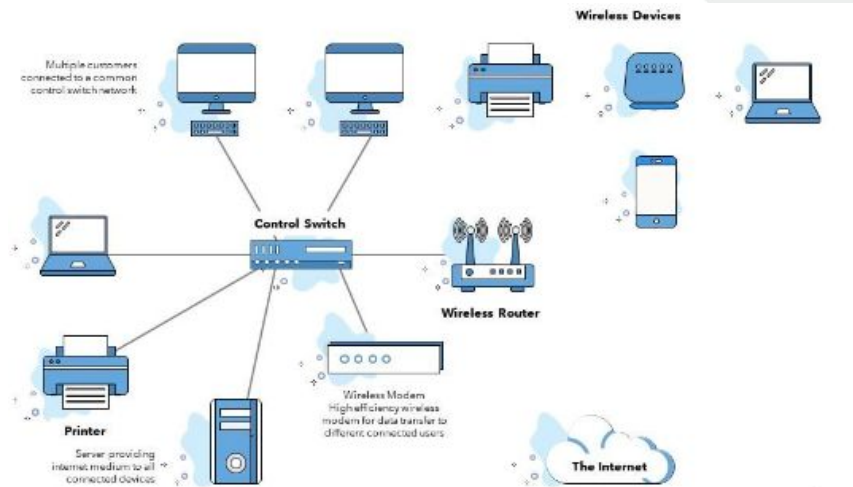


Greg Brown
Senior Technical Account Manager

Agenda

Goal: Bringing a client system online to an enterprise network on RHEL 9

- New for RHEL 9
- Architectures
 - Ifcfg
 - keyfiles
- Configuring the NIC
 - IP
 - Resolvers
 - HA/Aggregation
- Tools
 - Commandline. ip vs nmcli
 - System Roles
 - Portal Labs
 - Cockpit Web Console



New in RHEL 9



What's New ?



- By default, NetworkManager now uses the **keyfiles** to store new connection profiles. Note that the ifcfg format is still supported.
- The **teamd** service and the libteam library are deprecated. As a replacement, configure a bond instead of a network team.
- The **iptables** back end in firewalld is deprecated. Use the nftables framework to configure firewall rules.
- New core and IPv4-related networking **sysctl** kernel parameters. See `/usr/share/doc/ ... net.rst and ip-sysctl.rst` for details.
- Configuring **Multipath TCP** using NetworkManager is now fully supported
- [WireGuard VPN](#) is now in tech preview (9.3 and 9.4)!

RHEL Networking Architectures



Ifcfg



- `ip [show|link|address|route]*`
- `Ifconfig`
- `Ifup / ifdown`
- `route [add|delete]`
- `ss`
- `ethtool`
- `systemctl start|stop|restart|status network`
- `/sbin/init.d/network [start|stop]`
- `vim /etc/sysconfig/ifcfg-<ifname>`



Red Hat

*Command line operations do NOT survive a reboot

Structure of Ifcfg

- Called from /etc/init.d/network
 - uses parms from /etc/sysconfig/network-scripts
- Attributes of an ethernet connection
 - Interface - DEVICE=enp1s0
 - ip - IPADDR=10.0.1.27
 - subnet - PREFIX=24
 - gateway - if static GATEWAY0=192.168.1.1
 - resolver - dhcp vs static - BOOTPROTO=none
 - DNS - DNS1=ip-address
 - NM_CONTROLLED=yes|no
- Use vim to populate cfg files
- Supports IPv4 and IPv6
- Used in RHEL 6. Supported in RHEL 7. Deprecated in RHEL 8/9*

Network Manager



NetworkManager, which is a dynamic network control and configuration daemon to keep network devices and connections up and active when they are available.

NetworkManager ensures that network connectivity works. creates temporary connections when it detects that there is no network configuration

Scans and shows availability of interfaces. Has tools to completely build and manage interface connections.

Maintaining the state of devices after the reboot process and taking over interfaces which are set into managed mode during restart.

Network Manager uses the [ifcfg-rh plugin](#) to maintain support for ifcfg files and is defined in NetworkManager.conf.

Keyfile

- NetworkManager.service called from systemd
 - Uses parms from
/etc/NetworkManager/system-connections/XXX.nmconnection
- Supports IPv4 and IPv6
 - ipv4* variables
 - ipv6* variables
- Introduced in RHEL 7
- Designed to work in parallel with ifcfg scripts
- Takes precedence over ifcfg scripts
- Introduced to handle the myriad of new network types beginning with WiFi

Structure of Keyfiles

.INI format

[connection]

id=example_connection

type=ethernet

autoconnect=true ⇐ *On Boot or when resources become available*

[ipv4]

method=manual <= do not use *DHCP*

addresses=192.168.1.25

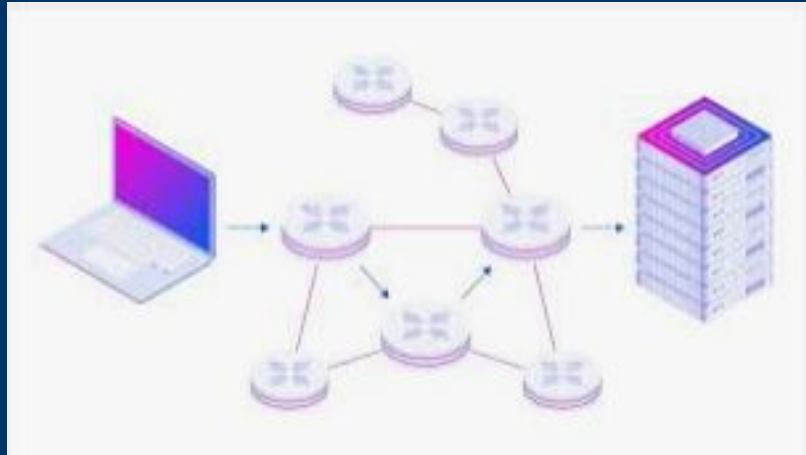
dns=192.168.1.10

[ethernet]

mac-address=00:53:00:8f:fa:66

mtu=1500

Addressing and Routing



Device Naming

- *Scheme 1*: Names incorporating Firmware or BIOS provided index numbers for on-board devices (example: **eno1**)
- *Scheme 2*: Names incorporating Firmware or BIOS provided **PCI Express** hotplug slot index numbers (example: **ens1**)
- *Scheme 3*: Names incorporating physical location of the connector of the hardware (example: **enp2s0**)
- *Scheme 4*: Names incorporating interface's MAC address (example: **enx78e7d1ea46da**) (Not default)
- *Scheme 5*: The traditional unpredictable kernel naming scheme, is used if all other methods fail (example: **eth0**)*

Device Naming

- Device names are stable across reboots.
- Device names stay fixed even if you add or remove hardware.
- Defective hardware can be seamlessly replaced.
- The network naming is stateless and does not require explicit configuration files.

By default, the udev device manager uses the **/usr/lib/systemd/network/99-default.link** file to determine which device naming policies to apply when it renames interfaces. The **NamePolicy** parameter in this file defines which policies udev uses and in which order:

```
NamePolicy=keep kernel database onboard slot path
```

```
Eth0    lo0    idrac   eno1    ens1    enp1s0
```

Define a network connection

Interface = device, such as eth0

Defined by *ifname*. Can have many connections (IPs, etc)

Connection - logical construct associated with a device

IP, netmask, gateway ...

\$ nmcli connection show

NAME	UUID	TYPE	DEVICE
enp1s0	507d9668-c6a3-4422-98c7-1991e66a2479	ethernet	enp1s0
virbr0	8c2453d9-b2c7-42e7-abfc-0f06d4a377a6	bridge	virbr0

Where....

NAME = Connection

DEVICE = Interface

All configurations associate with the connection.

Attributes of an ethernet connection

`== ifcfg format ==`

```
TYPE=Ethernet
DEVICE=enp1s0
UUID=0c8afb07-fe80-47bc-b61d-09ee85eefd6e

IPADDR=10.0.0.2
NETMASK=255.255.255.0
GATEWAY=10.0.0.1
PREFIX=24
NAME="System enp1s0"
DNS1=10.0.0.3
ONBOOT=yes
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
```

`== keyfile format ==`

```
[connection]
type=ethernet
interface-name=enp1s0
uuid=0c8afb07-fe80-47bc-b61d-09ee85eefd6e
[ipv4]
address1=10.0.0.2/24,10.0.0.1

id=enp1s0
dns=10.0.0.3;

autoconnect-retries=1
dns-search=example.org
may-fail=false
method=manual
```

DHCP

Attributes for DHCP

BOOTPROTO=dhcp

ONBOOT=yes

DHCP_HOSTNAME=hostname OR ..

DHCP_FQDN=fully.qualified.domain.name

GATEWAY0= # Defaults to **default**

DEFROUTE=no|yes

By default, NetworkManager calls the DHCP client, **dhclient**, when a profile has been set to obtain addresses automatically by setting BOOTPROTO to dhcp in an interface configuration file. If DHCP is required, an instance of dhclient is started for every Internet protocol, IPv4 and IPv6, on an interface. If NetworkManager is not running, or is not managing an interface, then the legacy network service will call instances of dhclient as required. By default, DHCP will assign the **GATEWAY**

DNS

Attributes for DNS (dynamic)

PEERDNS=no

DNS1=ip-address

DNS2=ip-address

Where ip-address is the address of a DNS server. This will cause the network service to update **/etc/resolv.conf** with the specified DNS servers specified. Only one DNS server address is necessary, the other is optional.

Static Routes

1) ip route add command line

2) Syntax

default via 192.168.0.1 dev enp1s0

10.10.10.0/24 via 192.168.0.10 dev enp1s0

172.16.1.10/32 via 192.168.0.10 dev enp1s0

3) /etc/sysconfig/network-scripts/route-enp1s0

ADDRESS0=10.10.10.0

NETMASK0=255.255.255.0

GATEWAY0=192.168.1.1

DEFROUTE=no|yes

Network Manager

- `nmcli` (1) - command-line tool for controlling NetworkManager
- `nmtui` (1) - Text User Interface for controlling NetworkManager
- `nm-connection-editor` (1) - network connection editor for NetworkManager
- `nm-settings` (5) - Description of settings and properties of NetworkManager
- `nm-settings-nmcli` (5) - Description of settings and properties of NetworkManager

Make it so!

```
# ip addr add 192.168.1.25/24 dev enp1s0  
# ip route add 192.168.1.1/24 via 10.0.0.1 enp1s0  
# vim ifcfg-enp1s0 to add in DHCP/DNS options
```



```
# nmcli con add \  
  con-name test_con \  
  ifname eth0 \  
  type ethernet \  
  ipv4.method manual \  
  ipv4.addresses 192.168.1.25/24 \  
  ipv4.dns 192.168.1.72
```

```
# In RHEL 7, and 8 update ifcfg-enp1s0  
# In RHEL 9 update enp1s0.nmconnection
```



Network Manager - nmcli

You only need to use just enough of a keyword to make it unique. For example:

Connection = con

Show = sho

Also, Tab completion is your friend!

```
# nmcli <tab tab>
```

```
agent    device    help    networking  
connection general  monitor  radio
```

```
# nmcli con <tab tab>
```

```
add      delete  edit   help   load   monitor show  
clone    down    export import modify reload up
```

Network Manager - nmcli

Attributes!

nmcli con **mod** test_con <tab tab>

Display all 292 possibilities? (y or n)

nmcli con **add** <tab tab>

autoconnect

con-name

connection.auth-retries

connection.autoconnect

connection.autoconnect-priority

connection.autoconnect-retries

connection.autoconnect-slaves

ipv4.ignore-auto-dns

ipv4.ignore-auto-routes

ipv4.may-fail

ipv4.method

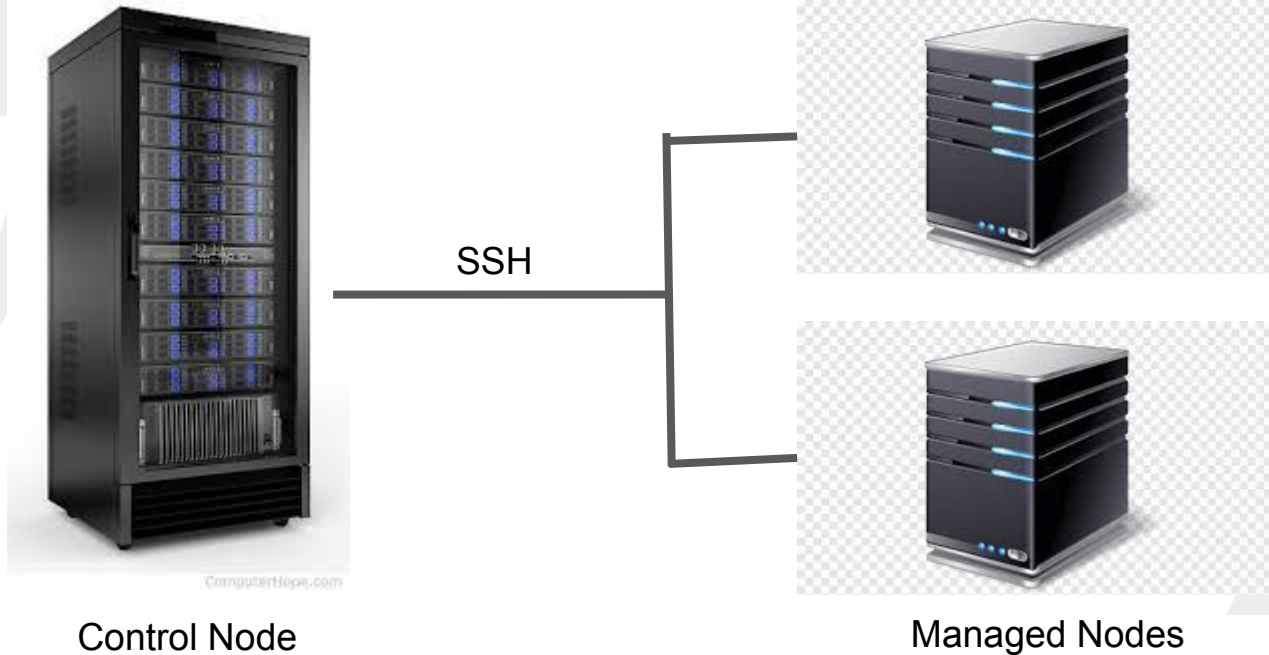
ipv4.never-default

ipv4.required-timeout

ipv4.route-metric

System Roles

Pre-configured Ansible modules designed to do rote sysadmin tasks.



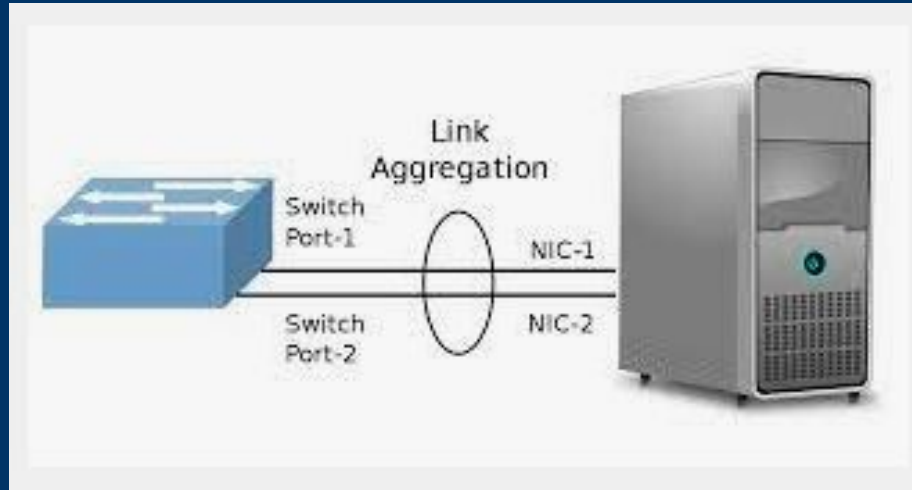
Networking System Role

```
---
- name: Configure an Ethernet connection with static IP
  hosts: node.example.com
  become: true
  tasks:
  - include_role:
      name: redhat.rhel_system_roles.network

  vars:
    network_connections:
      - name: enp7s0
        interface_name: enp7s0
        type: ethernet
        autoconnect: yes
        ip:
          address:
            - 192.0.2.1/24
            - 2001:db8:1::1/64
          gateway4: 192.0.2.254
          gateway6: 2001:db8:1::fffe
          dns:
            - 192.0.2.200
            - 2001:db8:1::ffbb
          dns_search:
            - example.com
```

- ▶ Configure ethernet interfaces with static or DHCP settings
- ▶ Configuring VLAN tagging
- ▶ Setup bridge and bonded interfaces
- ▶ Supports 802.1X network authentication
- ▶ Supports RHEL 9, RHEL 8, RHEL 7, and RHEL 6 hosts

HA and Link Aggregation



Bonds

A network bond is a method to combine or aggregate physical and virtual network interfaces to provide a logical interface with higher throughput or redundancy.

- Use *nmcli* to configure bond connections using the command line.
- Use our Red Hat Portal Lab.
- Use the RHEL *Web Console* to configure bond connections using a web browser.

E. Pluribus Unum

Bonding

Our bonding driver supports several modes.

- Active-backup - An HA solution for port/cable/switch failure
- Balance-tlb - The balance-tlb mode balances outgoing traffic by peer.
- Balance-alb - Adaptive load balancing includes Balance-tlb, but includes more robust algorithms for balancing traffic across ports.
- Round Robin - Requires static EtherChannel enabled, not Link Aggregation Control Protocol (LACP)-negotiated.
- LACP - Link Aggregation Control Protocol aka 802.3ad

Teaming

A network team is a method to combine or aggregate physical and virtual network interfaces to provide a logical interface with higher throughput or redundancy.

Bonding is handled exclusively in the **kernel**. **Teaming** includes a small set of kernel modules that provide an interface for teamd instances, but everything else is handled in **user space**.

Teaming has some advantages such as lower overhead, separate per-port link monitoring setup, and greater Extensibility.

Teaming has been deprecated in RHEL 9 in favor of bonds.

Teaming

Our teaming daemons support several modes.

- broadcast (data is transmitted over all ports)
- round-robin (data is transmitted over all ports in turn)
- active-backup (one port or link is used while others are kept as a backup)
- loadbalance (with active Tx load balancing and BPF-based Tx port selectors)
- lacp (implements the 802.3ad Link Aggregation Control Protocol)

Bonding with nmcli

```
# nmcli connection add type bond con-name bond0 ifname bond0  
bond.options "mode=active-backup"
```

```
# nmcli device status
```

DEVICE	TYPE	STATE	CONNECTION
enp7s0	ethernet	disconnected	--
enp8s0	ethernet	disconnected	--

Assign interfaces to the bond:

```
# nmcli connection add type ethernet slave-type bond con-name  
bond0-port1 ifname enp7s0 master bond0  
# nmcli connection add type ethernet slave-type bond con-name  
bond0-port2 ifname enp8s0 master bond0
```

Bonding with nmcli

Configure the IPv4 settings:

```
# nmcli connection modify bond0 ipv4.method disabled <= No DHCP  
# nmcli connection modify bond0 ipv4.addresses '192.0.2.1/24'  
ipv4.gateway '192.0.2.254' ipv4.dns '192.0.2.253' ipv4.dns-search  
'example.com' ipv4.method manual
```

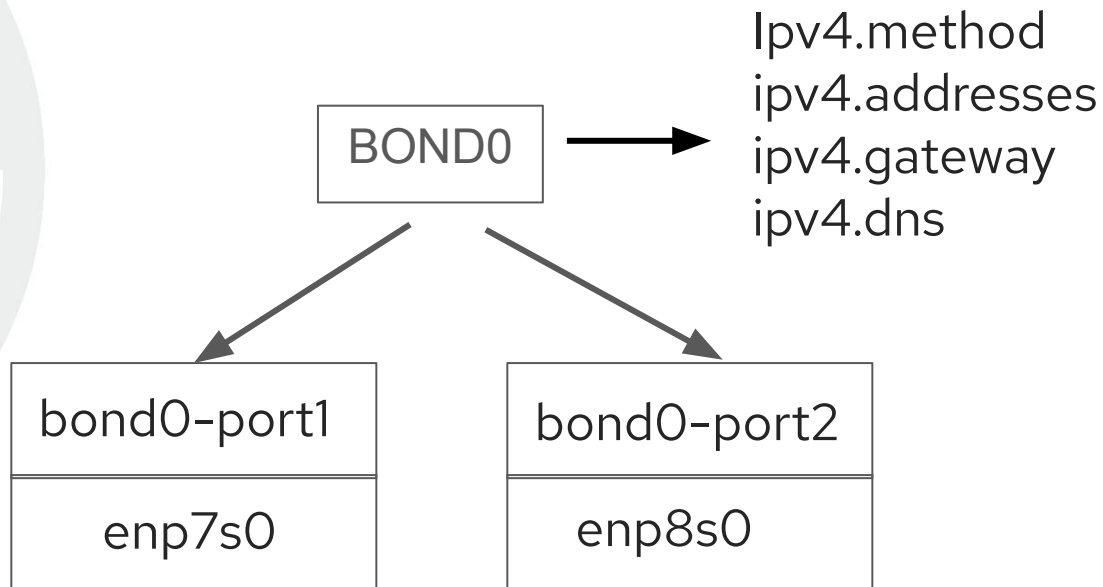
Make sure NM enables all ports automatically when the bond is enabled:

```
# nmcli connection modify bond0 connection.autoconnect-slaves 1
```

Activate the connection:

```
# nmcli connection up bond0
```

Bonding with nmcli



Bonding with nmcli

```
[root@grbrown-rhel9-vm ~]# nmcli con sho
```

NAME	UUID	TYPE	DEVICE
bond0	7a300f24-b92e-45c9-be46-1068400765c2	bond	bond0
enp1s0	507d9668-c6a3-4422-98c7-1991e66a2479	ethernet	enp1s0
virbr0	8c2453d9-b2c7-42e7-abfc-0f06d4a377a6	bridge	virbr0
test	5fa64268-90c9-4c3b-a70d-cb1d4c62d652	ethernet	--

con show

```
[root@grbrown-rhel9-vm ~]#
```

```
[root@grbrown-rhel9-vm ~]# nmcli con show bond0 | grep -E 'connection.id|connection.interface-name|connection.  
.autoconnect-slave|ipv4.method|bond.options'
```

```
connection.id:                bond0  
connection.interface-name:    bond0  
connection.autoconnect-slaves: 1 (yes)  
ipv4.method:                  auto  
bond.options:                 mode=active-backup,downdelay=0,miimon=100,primary=enp1s0,updelay=0
```

bond0
(master)

```
[root@grbrown-rhel9-vm ~]#
```

```
[root@grbrown-rhel9-vm ~]# nmcli con show enp1s0 | grep -E 'connection.id|connection.master|connection.slave-  
type|connection.autoconnect|ipv4.addresses'
```

```
connection.id:                enp1s0  
connection.autoconnect:       yes  
connection.autoconnect-priority: 0  
connection.autoconnect-retries: -1 (default)  
connection.master:            bond0  
connection.slave-type:        bond  
connection.autoconnect-slaves: -1 (default)
```

enp1s0
(slave)

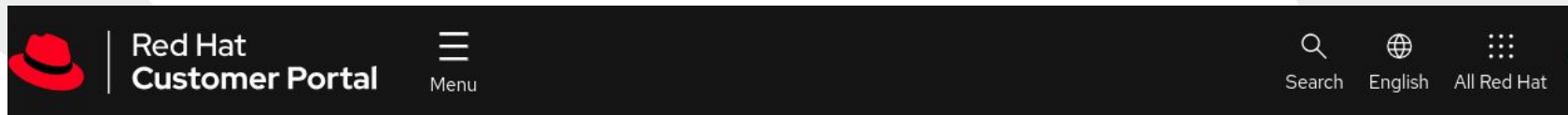
```
[root@grbrown-rhel9-vm ~]#
```

Portal Labs



Bonding with Portal Lab

<https://access.redhat.com/labs/networkbondinghelper/>



[Labs](#) > Network Bonding Helper

Network Bonding Helper

[SWITCH TO MANUAL MODE](#)



RHEL version

Choose your Red Hat Enterprise Linux version

Red Hat Enterprise Linux 9



NEXT >

Bonding with Portal Lab

<https://access.redhat.com/labs/networkbondinghelper/>

Walk through some guided questions.

- Multiple-switch or single-switch?
- Do you want to use LACP Link aggregation?
- Give it Bond and Device names.
- Give it an IP, Subnet mask and Gateway
- Do you want MII or ARP monitoring?

Bond Name

bond0

Device Names

ens192

ens193

Profile Names

ens192

ens193

Go!

GENERATE ▶

Bonding with Portal Lab

<https://access.redhat.com/labs/networkbondinghelper/>

You end up with a downloadable shell script to execute.

```
#!/bin/bash
nmcli connection add type bond con-name bond0 ifname bond0 bond.options
"mode=active-backup,primary=ens192,miimon=100"
nmcli connection modify bond0 ipv4.addresses '192.168.1.10/24'
nmcli connection modify bond0 ipv4.gateway '192.168.1.1'
nmcli connection modify bond0 ipv4.method manual
nmcli connection modify bond1 ipv6.method disabled
nmcli connection add type ethernet slave-type bond con-name ens192 ifname ens192 master
bond0
nmcli connection add type ethernet slave-type bond con-name ens193 ifname ens193 master
bond0
nmcli connection up ens192
nmcli connection up ens193
nmcli connection up bond0
```

Teaming with Portal Lab

<https://access.redhat.com/labs/teamdconfighelper/>

Network Teaming Helper

SWITCH TO MANUAL MODE

▶ **RHEL Version**

Choose Your Red Hat Enterprise Linux Version

Red Hat Enterprise Linux 8 ▼

NEXT ▶

Cockpit



Using the web console AKA Cockpit

```
# yum install cockpit  
# systemctl enable --now cockpit.socket  
# firewall-cmd --add-service=cockpit --permanent  
# firewall-cmd --reload
```

In a browser, load ...
<https://localhost:9090>

Using the web console AKA Cockpit

The screenshot shows the Red Hat Cockpit web console interface. The browser address bar displays `https://localhost:9090/network`. The left sidebar contains a navigation menu with the following items: `root@grbrown-rhel9-vm`, a search bar, `System`, `Overview`, `Logs`, `Storage`, `Networking` (highlighted with a red arrow), `Podman containers`, `Accounts`, `Services`, `Tools`, and `Applications`. The main content area displays network-related information. At the top, there are two graphs: `Bytes transmitting` and `Bytes receiving`. Below these, a `Firewall` section shows it is enabled with a toggle switch and a button to `Edit rules and zones`. The `Interfaces` section includes buttons for `Add bond`, `Add team`, `Add bridge`, and `Add VLAN`. A table lists the active interfaces:

Name	IP address	Sending	Receiving
<code>enp1s0</code> (with a red arrow pointing to it)	192.168.122.181/24	0 bps	416 bps
<code>virbr0</code>	192.168.124.1/24	No carrier	

Using the web console AKA Cockpit - IP Config

Storage

Networking

Podman containers

Accounts

Services

Tools

enp1s0

Red Hat, Inc. Virtio network device virtio_net

52:54:00:27:91:B6



Status

192.168.122.181/24, fe80:0:0:0:5054:ff:fe27:91b6/64

Carrier

Yes

General

☒ Connect automatically

IPv4

Automatic (DHCP)

edit

IPv6

Automatic

edit

MTU

Automatic

edit

Using the web console AKA Cockpit - Bonding

root@grbrown-rhel9-vm

Search

System

Overview

Logs

Storage

Networking

Podman containers

Accounts

Services

Tools

Applications

Help

Session

Bond settings

Name: bond0

Interfaces: ☒ enp1s0 ☒ virbr0

MAC:

Mode: Active backup

Primary: enp1s0

Link monitoring: MII (recommended)

Apply Cancel

Using the web console AKA Cockpit - Teaming

The screenshot shows a web browser at `https://localhost:9090/network` displaying the Red Hat Cockpit interface. The left sidebar shows the user is logged in as `root@grbrown-rhel9-vm` and the 'Networking' section is selected. The main area displays the 'Team settings' dialog for a team named 'team0'. The settings are as follows:

Setting	Value
Name	team0
Ports	☒ enp1s0 ☒ virbr0
Runner	Active backup
Link watch	Ethtool
Link up delay	0
Link down delay	0

At the bottom of the dialog are 'Apply' and 'Cancel' buttons.

Denouement



How to get heyah from theyah

My Leapp upgrade preserved my...

- [Legacy eth0](#) interface name.
You can change using either [udev rules](#) or [systemd](#) links.
- Old ifcfg-*ifname* definitions.
Use [nmcli migrate](#) to migrate network configurations to **keyfiles** under `/etc/NetworkManager/system-connections`
nmcli connection migrate eth0 OR
nmcli connection migrate <= all interfaces



How to get heyah from theyah

My Leapp upgrade preserved my...

- Network Team config
team2bond (1)

OR ...

- Do nothing. RHEL 9 will support your old configs

BUT

- New configs will use advanced udev naming,
- Network Manager, and
- keyfiles under
/etc/NetworkManager/system-connections



Summary

- Let the system assign udev names for your interfaces
- Configs will be stored under `/etc/NetworkManager/system-connections`
- Tab completion is your friend!
- Use bonds over teams
- You have a variety of tools for getting the job done. One method might look like ...
 - Use Network Manager (nmcli) for the daily care and feeding of your networks
 - Use Cockpit for big jobs, like setting up a bond
- Tools like System Roles and Portal labs are out there to help!

Dig yourselves! You got this!

For reference

- KB: [Why are scripts under /etc/sysconfig/network-scripts directory gone on rhel8/rhel9?](#)
- BLOG: [RHEL 9 networking: Say goodbye to ifcfg-files, and hello to keyfiles](#)
- If all else fails....
[Configuring and managing networking](#)
- For extra fun!
[Getting started with Multipath TCP](#)



Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.



linkedin.com/company/red-hat



youtube.com/user/RedHatVideos



facebook.com/redhatinc



twitter.com/RedHat