VLAN

Virtual LANs give you the ability to sub-divide a LAN. Linux can accept **VLAN** tagged traffic and presents each **VLAN ID** as a different network interface (eg: eth0.100 for **VLAN ID** 100)

This article explains how to configure a VLAN using iproute2 (ht tps://www.archlinux.org/packages/?name=iproute2) and systemd-networkd or netctl.

Related articles

Network Configuration

systemd-networkd

Netctl

Contents

- 1 Configuration
 - 1.1 Create the VLAN device
 - 1.2 Add an IP
 - 1.3 Turning down the device
 - 1.4 Removing the device
 - 1.5 Starting at boot
 - 1.5.1 systemd-networkd single interface
 - 1.5.2 systemd-networkd bonded interface

- 1.5.3 netct1
- **2** Troubleshooting
 - 2.1 udev renames the virtual devices

Configuration

Previously Arch Linux used the vconfig command to setup VLANs. This had been superseded by the ip command. Make sure you have iproute2 (https://www.archlinux.org/packages/?name=iproute2) installed.

In the following examples, lets assume the **interface** is eth0, the assigned **name** is eth0.100 and the **vlan id** is 100.

Create the VLAN device

Add the VLAN with the following command:

ip link add link eth0 name eth0.100 type vlan id 100

Run ip link to confirm that it has been created.

https://wiki.archlinux.org/index.php/VLAN 2/12

This interface behaves like a normal interface. All traffic routed to it will go through the master interface (in this example, eth0) but with a VLAN tag. Only VLAN aware devices can accept them if configured correctly else the traffic is dropped.

Using a **name** like eth0.100 is just convention and not enforced; you can alternatively use eth0_100 or something descriptive like IPTV. To see the VLAN ID on an interface, in case you used an unconventional name:

```
# ip -d link show eth0.100
```

The -d flag shows full details on an interface:

```
# ip -d addr show
4: eno1.100@eno1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 96:4a:9c:84:36:51 brd ff:ff:ff:ff:ff promiscuity 0
    vlan protocol 802.1Q id 100 <REORDER_HDR>
    inet6 fe80::944a:9cff:fe84:3651/64 scope link
    valid_lft forever preferred_lft forever
```

Add an IP

Now add an IPv4 address to the just created vlan link, and activate the link:

```
# ip addr add 192.168.100.1/24 brd 192.168.100.255 dev eth0.100
# ip link set dev eth0.100 up
```

https://wiki.archlinux.org/index.php/VLAN 3/12

Turning down the device

To cleanly shutdown the setting before you remove the link, you can do:

```
# ip link set dev eth0.100 down
```

Removing the device

Removing a VLAN interface is significantly less convoluted

```
# ip link delete eth0.100
```

Starting at boot

systemd-networkd single interface

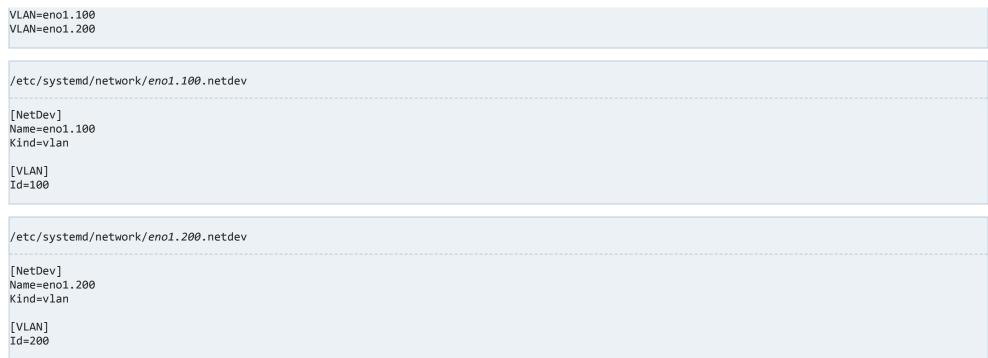
Use the following configuration files (Remember that systemd config files are case sensitive!):

```
/etc/systemd/network/eno1.network

[Match]
Name=eno1

[Network]
DHCP=ipv4
;these are arbitrary names, but must match the *.netdev and *.network files
```

https://wiki.archlinux.org/index.php/VLAN 4/12



You'll have to have associated .network files for each .netdev to handle addressing and routing. For example, to set the eno1.100 interface with a static IP and the eno1.200 interface with DHCP (but ignoring the supplied default route), use:

```
/etc/systemd/network/eno1.100.network

[Match]
Name=eno1.100

[Network]
DHCP=no

[Address]
Address=192.168.0.25/24

/etc/systemd/network/eno1.200.network
```

https://wiki.archlinux.org/index.php/VLAN 5/12

[Match]
Name=eno1.200

[Network]
DHCP=yes

[DHCP]
UseRoutes=false

Then enable systemd-networkd.service. See systemd-networkd for details.

systemd-networkd bonded interface

Similar to above, you're just going to stack more of the concepts in place. You'll want to ensure that you've got a bond set up in your switch and also make sure its a trunk with tagged vlans corresponding to what you create below. Convention would be to create a bond interface with the name bond, however there is a known issue where the bonding module, when loaded, creates a bond device of the name bond which systemd then refuses to configure (as systemd tries to respectfully leave alone any device it did not create).

Tip: To prevent the bonding module to create an initial bond0 interface, set the max_bonds option of the bonding module to 0 (default value is 1):

/etc/modprobe.d/bonding.conf

options bonding max bonds=0

See Kernel modules#Setting module options and Linux Ethernet Bonding Driver HOWTO (Kernel Documentation) (https://www.kernel.org/doc/Documentation/networ

king/bonding.txt) for details.

For the purposes of this write up, we are going to use **bondname** and you can make the choice yourself.

First, we create the bond device:

/etc/systemd/network/bondname.netdev

[NetDev]
Name=bondname
Kind=bond

[Bond]
Mode=802.3ad
LACPTransmitRate=fast

Now create a .network directive that references the vlans and interface carriers. In this case we'll use the convention for a dual port fiber module:

/etc/systemd/network/bondname.network

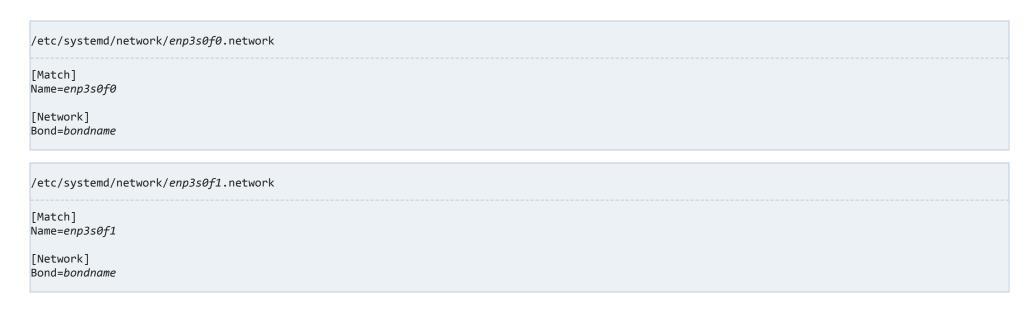
[Match]
Name=bondname

[Network]
VLAN=vlan10
VLAN=vlan20
VLAN=vlan30
BindCarrier=enp3s0f0 enp3s0f1

https://wiki.archlinux.org/index.php/VLAN 7/12

We're using the vlan<number> naming convention here, you can use something else but realize that this is a named reference so you'll have to have a corresponding set of files with the same name.

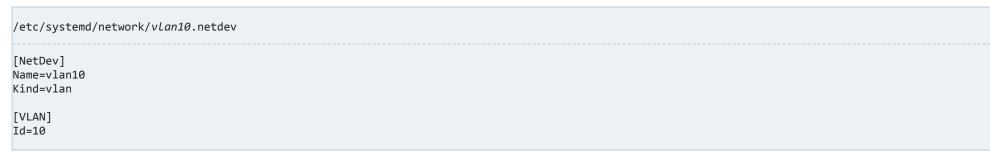
We'll now set up the physical network interfaces:



At this time you could reboot, and likely should, because the bonded interface is created at boot time. Restarting systemd-networkd will consume changes from these files typically, but device creation seems to occur at startup.

We will now set up the VLANs. You should be aware that having multiple VLANs can result in a situation where your machine has multiple default routes, so you'll need to specify a Destination directive in the network directives to ensure that only one VLAN is being used for a default route. In this case we'll use the VLAN with an ID of 10 as our default route.

https://wiki.archlinux.org/index.php/VLAN 8/12



Now create the associated network directive to set an address:

/etc/systemd/network/vlan10.network

[Match]
Name=vlan10

[Network]
VLAN=vlan10

[Address]
Address=10.10.10.2/24

[Route]
Destination=0.0.0/0
Gateway=10.10.10.1

We'll create a similar pair of files for the VLAN with an ID of 20:

/etc/systemd/network/vlan20.netdev

[NetDev]
Name=vlan20
Kind=vlan

[VLAN]
Id=20

https://wiki.archlinux.org/index.php/VLAN 9/12

/etc/systemd/network/vlan20.network

[Match]
Name=vlan20

[Network]
VLAN=vlan20

[Address]
Address=10.10.20.2/24

[Route]
Destination=10.10.20.0/24

Gateway=10.10.20.1

And again for the VLAN with an ID of 30:

/etc/systemd/network/vlan30.netdev

[NetDev]
Name=vlan30
Kind=vlan

[VLAN]
Id=30

/etc/systemd/network/*vlan30*.network

[Match] Name=vlan30

[Network] VLAN=vlan30

[Address] Address=10.10.30.2/24

[Route]

Destination=10.10.30.0/24

Gateway=10.10.30.1

https://wiki.archlinux.org/index.php/VLAN 10/12

Note that the Destination on vlan10 is set to 0.0.0.0/0, which will match all outbound, becoming the default route.

netctl

You can use **netctl** for this purpose, see the self-explanatory example profiles in {{ic|/etc/netctl/examples/vlan-{dhcp,static} }}.

Troubleshooting

udev renames the virtual devices

An annoyance is that **udev** may try to rename virtual devices as they are added, thus ignoring the **name** configured for them (in this case eth0.100).

For instance, if the following commands are issued:

```
# ip link add link eth0 name eth0.100 type vlan id 100
# ip link show
```

This could generate the following output:

```
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
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP qlen 1000
```

https://wiki.archlinux.org/index.php/VLAN 11/12

```
link/ether aa:bb:cc:dd:ee:ff brd ff:ff:ff:ff:ff
3: rename1@eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state DOWN
link/ether aa:bb:cc:dd:ee:ff brd ff:ff:ff:ff:ff
```

udev has ignored the configured virtual interface name eth0.100 and autonamed it **rename1**.

The solution is to edit /etc/udev/rules.d/network_persistent.rules and append **DRIVERS=="?*"** to the end of the physical interface's configuration line.

For example, for the interface **aa:bb:cc:dd:ee:ff** (eth0):

```
/etc/udev/rules.d/network_persistent.rules

SUBSYSTEM=="net", ATTR{address}=="aa:bb:cc:dd:ee:ff", NAME="eth0", DRIVERS=="?*"
```

A reboot should mean that VLANs configure correctly with the names assigned to them.

Retrieved from "https://wiki.archlinux.org/index.php?title=VLAN&oldid=506211"

- This page was last edited on 5 January 2018, at 22:16.
- Content is available under GNU Free Documentation License 1.3 or later unless otherwise noted.

https://wiki.archlinux.org/index.php/VLAN 12/12