Cumulus® VX™ Lab Guide

Intro to Cumulus VX

For Use with Cumulus VX

Contents

Version 1.5
December 10, 2015

Overview

Objective

This lab guide provides you with a basic run-through of the functionality and concepts within Cumulus Linux, using the Cumulus VX virtual machine.

Lab Environment

The lab exercises are designed to be performed on your own laptop, using a two-switch topology on the Cumulus VX setup in VirtualBox. The architecture and topology is shown below in Figure 1.

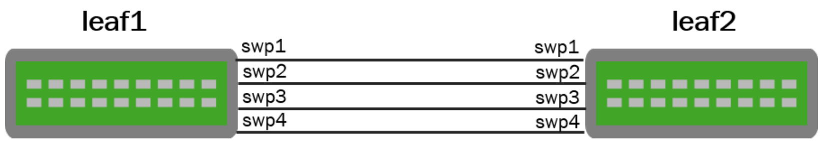


Figure 1. Two-Switch Cumulus VX Environment Overview

Each Cumulus VX environment includes two interconnected switches, called **leaf1** and leaf2. The lab automation script sets up the VMs and opens console terminals for each switch, which can then be used to perform the following exercises.

Prerequisites

This lab guide assumes you have a basic understanding of networking and Linux operating systems, including how to use nano to edit files in Linux.

If you are new to Linux administration, there are introduction to Linux videos available at https://cumulusnetworks.com/technical-videos/.

About Cumulus VX

LEARN TEST DRIVE PROTOTYPE DEVELOP

Cumulus VX is a virtual appliance that helps potential customers and partners familiarize themselves with Cumulus Networks' technology, while providing existing customers and partners with a platform to prototype network operations and develop custom applications prior to deploying into production environments. Without the need for a bare metal switch or specialized hardware, Cumulus VX runs on many popular hypervisors, making traditional networking protocols such as BGP and MLAG, and Cumulus Networks-specific technologies such as Prescriptive Topology Manager (PTM), available for testing and configuration.

Data center cloud administrators and network engineers can leverage Cumulus VX. Some use cases for Cumulus VX include (but are not limited to):

- **Learn:** Cumulus VX will help IT and Network professionals get familiar with Open Networking and Cumulus Linux.
- **Test drive:** Shorten the evaluation period by testing Cumulus Linux features and functionality in your environment within your own timeframe, without needing to invest in any hardware platform.
- **Prototype & Pre-Production Rollout:** Existing customers can leverage Cumulus VX to experiment and design rollouts, minimizing errors and time to production.
- **Development:** Customers and partners can leverage Cumulus VX to develop custom, portable applications that you can seamlessly migrate to your production Cumulus Linux or RMP deployments.

Cumulus VX can run in the following environments:

Hypervisor	Supported Configurations
VirtualBox	• Standalone • With GNS3 • With Vagrant
KVM	• Standalone • With GNS3
VMware	• Workstation • Fusion • ESXi

For more information, read the [Getting Started Guide] and visit the [Cumulus VX Community] to ask any questions or share your experiences.

Cumulus VX Features

Cumulus VX has the same foundation as Cumulus Linux and Cumulus RMP. It includes all the control plane elements of Cumulus Linux or Cumulus RMP, but does not have an actual ASIC or NPU for line rate performance or hardware acceleration. Essentially, switchd is not a part of Cumulus VX, and tools that interact with switchd, like cl-cfg, and other hardware management tools, are not available.

Thus, Cumulus VX is not a production-ready virtual switch or virtual router. It isn't meant to run on production switches or carry production data traffic.

However, you can use tools like jdoo in Cumulus VX to monitor the virtual switch, the same automation and zero touch provisioning tools, as well as security and QoS tools. And while you cannot upgrade the Cumulus VX operating system using apt-get upgrade|update or use ONIE, you can use apt-get to install additional software packages, whether they are Cumulus Linux-specific or Debian-specific.

The following table outlines the similarities and differences between Cumulus VX and other Cumulus Networks operating systems.

Feature or Functionality	Cumulus VX Behavior
Installation and Upgrade	No ONIE, new images available with every GA release, no upgrade path.
Hardware Acceleration	No switchd. Data path forwarding is dependent on the choice of hypervisor and VM resources.
Licensing	No license required since switchd is not present.
Hardware management	None.
Hardware limitations	None. Dependent on hypervisor and VM resources. Certain features such as route-table-size could accommodate more routes than are supported in hardware (32K routes), given available memory.
Production-ready	No.
Linux extensibility	Yes.
Layer 2 features	Yes, hypervisor/topology manager dependent.
Layer 3 features	Yes.
Network virtualization	Yes (software forwarding).
OS management (ZTP, ifupdown2, third party packages)	Yes.
Automation, monitoring, troubleshooting	Yes (excluding ONIE and hardware dependencies).
Security	Yes.
QoS	Yes.

Lab 1: Setting up and Accessing Your Switches

Objective

Import the VirtualBox OVA and start the 2 leaf VMs.

Goals

- Install VirtualBox
- Import the OVA file
- Start the VMs

Procedure

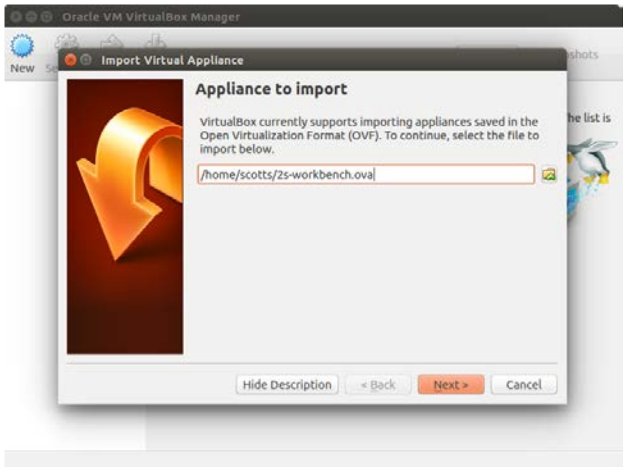
1. Download the required VirtualBox files from https://www.virtualbox.org/wiki/Downloads

2. Install VirtualBox using the installation method for your chosen OS.

3. Once installed, launch the VirtualBox program.

4. If you have not done so, download the 2s-workbench.ova file from https://cumulusnetworks.box.com/vx2s

5. Begin the import. Click **File → Import Appliance** and follow the wizard to define your file location to import the OVA file.



Appliance to import

VirtualBox currently supports importing appliances saved in the Open Virtualization Format (OVF). To continue, select the file to import below.

/home/scotts/2s-workbench.ova

6. When the import finishes, you see two VMs, one for each switch in a stopped state.

7. Double-click each VM to start them. This opens console windows for each VM.

8. Click to select one, then press Enter to get a login prompt.

   leaf1 login: **cumulus**
   Password: **CumulusLinux!**

   leaf2 login: **cumulus**
   Password: **CumulusLinux!**

   To free the mouse focus from the console window, you may need to press the right <Ctrl> key (Windows) or the left <Command> key (Mac).

This completes Lab 1.

Lab 2: Basic Switch Port Configuration

Objective

Define and activate switch port interfaces and basic Layer 2 constructs.

The switches in this Cumulus VX environment have the first four front panel ports, swp1 through swp4, connected to each other respectively, so swp1 on leaf1 is connected to swp1 on leaf2, and so forth. The cabling is illustrated in Figure 2.

The link state for port swp1 through swp4 is down until each pair of ports is defined and configured on each switch, and after both corresponding ports are brought up.

Goals

- Verify ports swp1 through swp4 on each switch and verify connectivity.
- Create an LACP-bonded interface using ports swp1 and swp2.
- Create a bridge with two untagged bridge members (access switch ports).
- Show interface statistics.

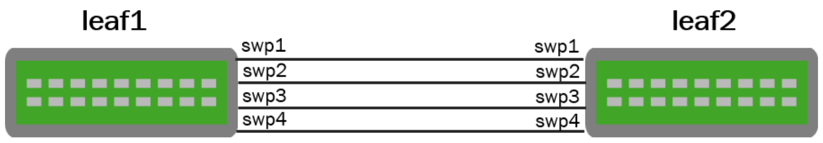
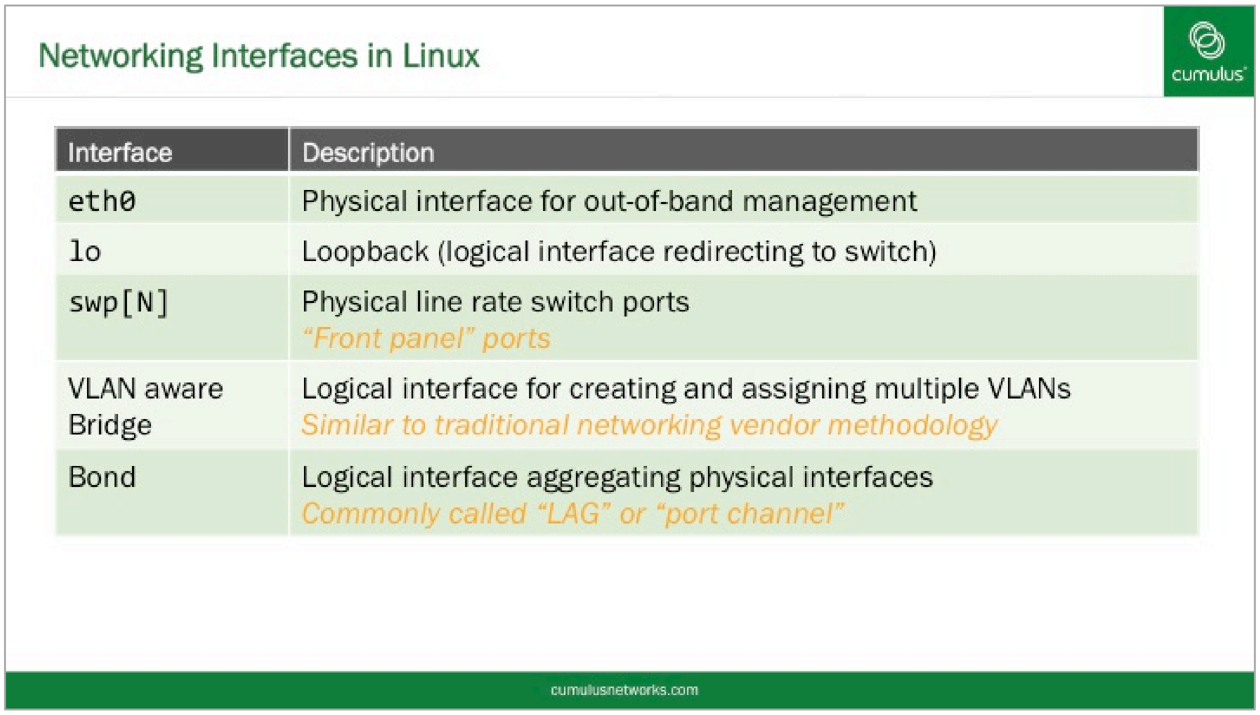


Figure 2. Lab Switch Port Cabling

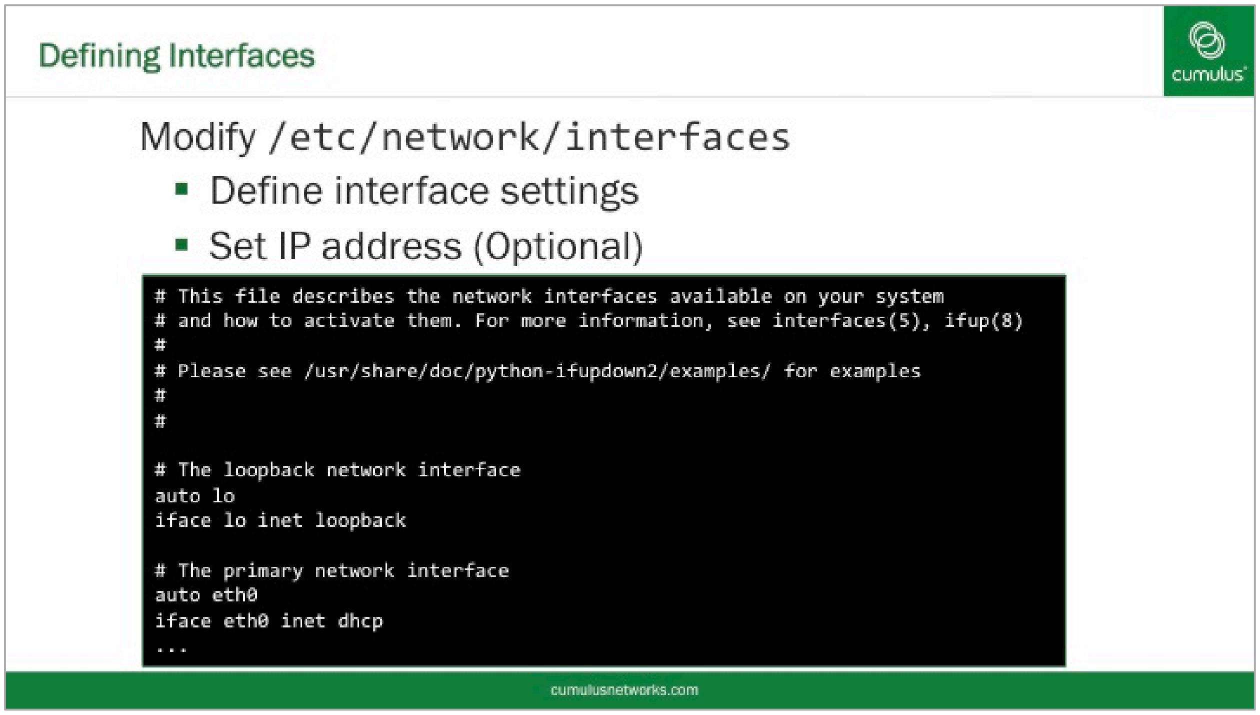
Key Concepts



Networking Interfaces in Linux

Interface	Description
`eth0`	Physical interface for out-of-band management
`lo`	Loopback (logical interface redirecting to switch)
`swp[N]`	Physical line rate switch ports *"Front panel" ports*
VLAN aware Bridge	Logical interface for creating and assigning multiple VLANs *Similar to traditional networking vendor methodology*
Bond	Logical interface aggregating physical interfaces *Commonly called "LAG" or "port channel"*

cumulusnetworks.com



Defining Interfaces

Modify `/etc/network/interfaces`

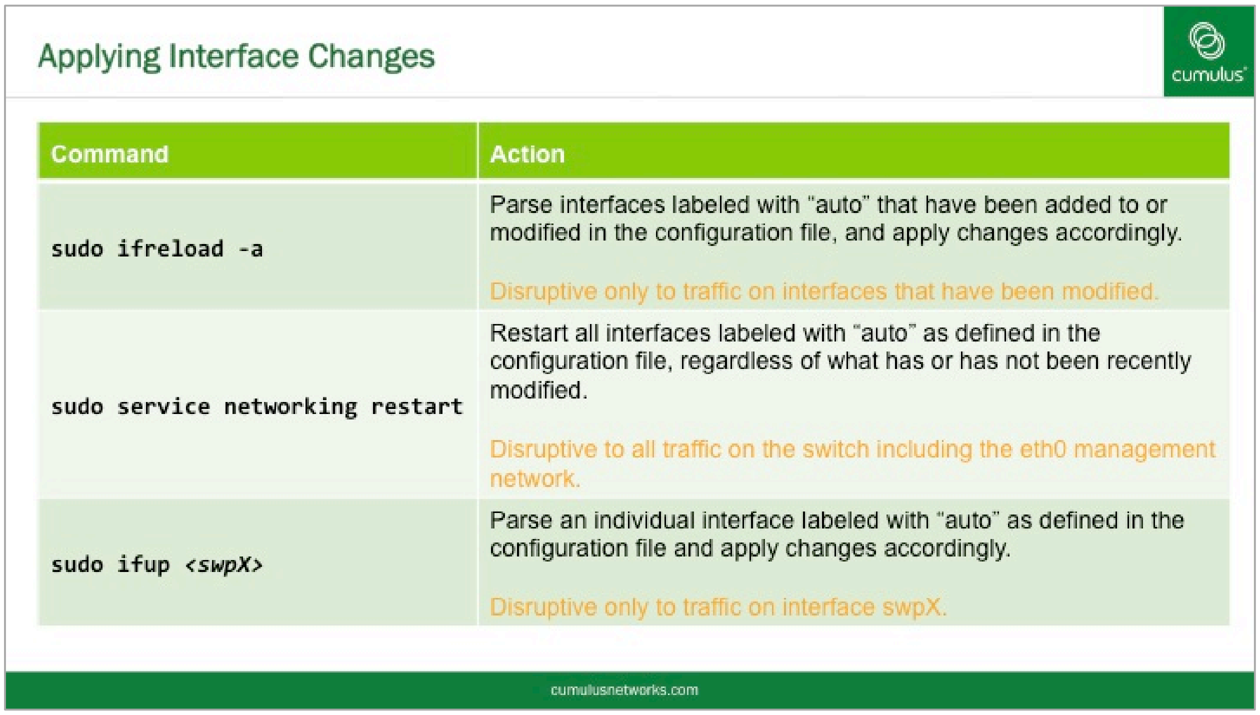
- Define interface settings
- Set IP address (Optional)

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5), ifup(8)
#
# Please see /usr/share/doc/python-ifupdown2/examples/ for examples
#
#

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet dhcp
...
```

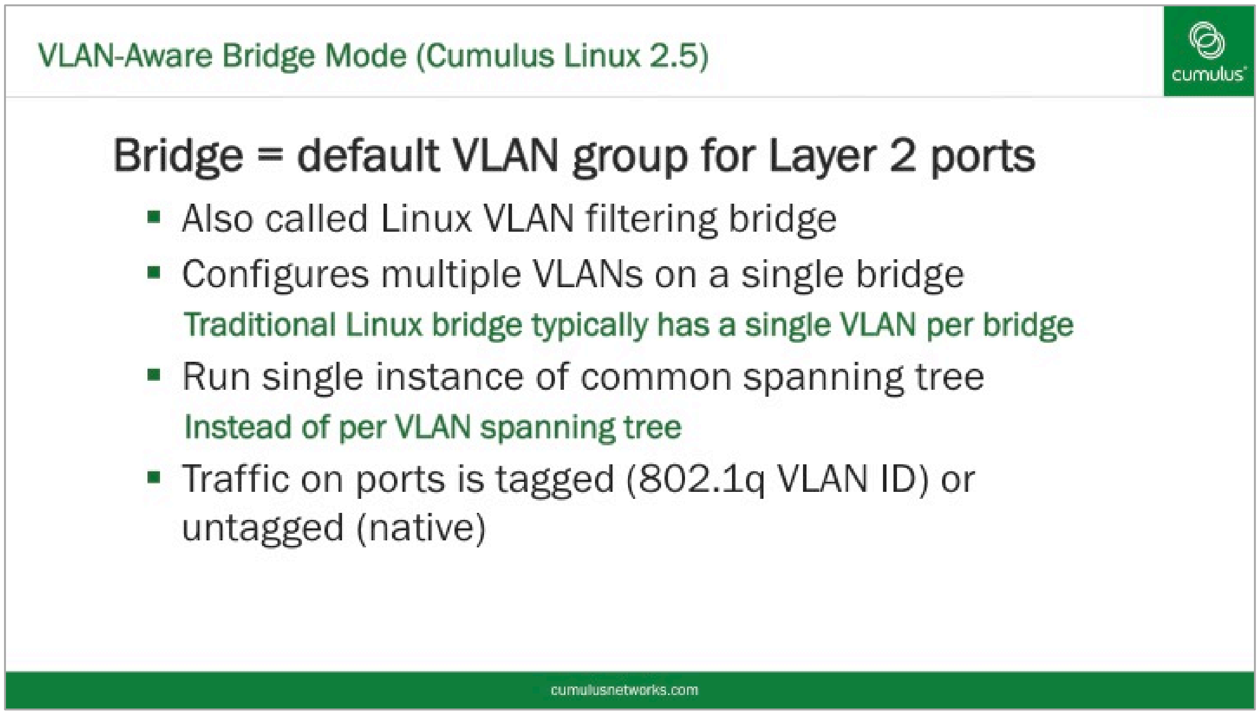
cumulusnetworks.com



Applying Interface Changes

Command	Action
`sudo ifreload -a`	Parse interfaces labeled with "auto" that have been added to or modified in the configuration file, and apply changes accordingly. Disruptive only to traffic on interfaces that have been modified.
`sudo service networking restart`	Restart all interfaces labeled with "auto" as defined in the configuration file, regardless of what has or has not been recently modified. Disruptive to all traffic on the switch including the eth0 management network.
`sudo ifup <swpX>`	Parse an individual interface labeled with "auto" as defined in the configuration file and apply changes accordingly. Disruptive only to traffic on interface swpX.

cumulusnetworks.com



VLAN-Aware Bridge Mode (Cumulus Linux 2.5)

Bridge = default VLAN group for Layer 2 ports

- Also called Linux VLAN filtering bridge
- Configures multiple VLANs on a single bridge
  Traditional Linux bridge typically has a single VLAN per bridge
- Run single instance of common spanning tree
  Instead of per VLAN spanning tree
- Traffic on ports is tagged (802.1q VLAN ID) or untagged (native)

cumulusnetworks.com

Procedure

1. Review interface configuration concepts in Figure 3.

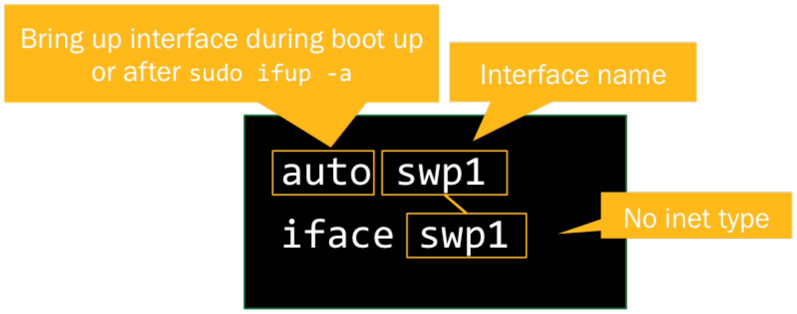


Figure 3. Basic Switch Port Configuration

2. Configure and activate two switch ports. Verify connectivity between switches.

Define swp1 and swp2 on leaf1. Do the same on leaf2.

```
leaf1:~$ sudo nano /etc/network/interfaces
```

Add the following lines for swp1 and swp2:

```
auto swp1
iface swp1

auto swp2
iface swp2
```

Save the `/etc/network/interfaces` file.

On leaf1, bring up swp1:

```
leaf1:~$ sudo ifup swp1

[   75.115763] 8021a: adding VLAN 0 to HW filter on device swp1
```

Go to leaf2, configure swp1 and swp2 similarly as above, and bring up swp1. This activates the link.

```
leaf2:~$ sudo ifup swp1

[  316.216799] 8021a: adding VLAN 0 to HW filter on device swp1
```

```
leaf1:~$ ip link show dev swp1

3: swp1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
mode DEFAULT qlen 500
    link/ether 08:9e:01:f8:95:0d brd ff:ff:ff:ff:ff:ff
```

Similarly, bring up swp2 on both leaf1 and leaf2.

```
leaf1:~$ sudo ifup swp2
```

```
leaf2:~$ sudo ifup swp2
```

3. **Create an LACP port bond (bundle/channel).**
   Assign ports swp1 and swp2 to a bonded interface. This creates a topology as shown below in Figure 4.

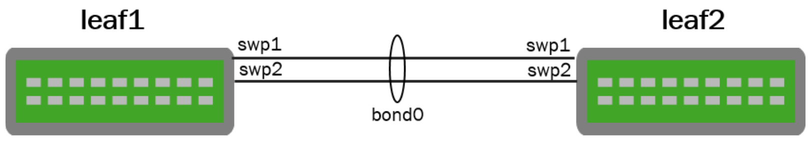


Figure 4. LACP Port Bond

Add the following stanzas to the `/etc/network/interfaces` file. Make sure this is done on both switches.

```
cumulus@leaf1$ sudo nano /etc/network/interfaces
```

Add the following lines for bond0 on both leaf1 and leaf2:

```
auto bond0
iface bond0
  bond-slaves swp1 swp2
  bond-mode 802.3ad
  bond-miimon 100
  bond-lacp-rate 1
  bond-min-links 1
  bond-xmit-hash-policy layer3+4
```

4. Bring up the bonded interface on both leaf1 and leaf2:

```
~$ sudo ifup bond0
bonding: bond0 is being created...
bonding: bond0: setting xmit hash policy to layer3+4 (1).
bonding: bond0: Setting MII monitoring interval to 100.
bonding: bond0: setting mode to 802.3ad (4).
bonding: bond0: Setting LACP rate to fast (1).
bonding: bond0: Setting min links value to 1
bonding: bond0: Adding slave swp1
bonding: bond0: enslaving swp1 as a backup interface with an up link.
bonding: bond0: Adding slave swp2
bonding: bond0: enslaving swp2 as a backup interface with an up link.
ADDRCONF(NETDEV_UP): bond0: link is not ready

...
```

Once bond0 has been brought up on both switches, the bond becomes active:

```
ADDRCONF(NETDEV_UP): bond0: link becomes ready
```

5. Check the bond interface status on both leaf1 and leaf2:

```
~$ cat /proc/net/bonding/bond0 | less
Ethernet Channel Bonding Driver: v3.7.1 (April 27, 2011)

Bonding Mode: IEEE 802.3ad Dynamic link aggregation
Transmit Hash Policy: layer3+4 (1)
MII Status: up
MII Polling Interval (ms): 100
Up Delay (ms): 0
Down Delay (ms): 0

802.3ad info
LACP rate: fast
Min links: 1
Aggregator selection policy (ad_select): stable
System Identification: 65535 6c:64:1a:00:2a:91
Active Aggregator Info:
        Aggregator ID: 1
        Number of ports: 2
        Actor Key: 17
        Partner Key: 17
        Partner Mac Address: 08:9e:01:f8:88:37
LACP Bypass Info:
        Allowed: 0
        Timeout: 0

Slave Interface: swp1
MII Status: up
Speed: 1000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 08:9e:01:f8:95:0d
Aggregator ID: 1
LACP bypass priority: 0
Slave queue ID: 0

Slave Interface: swp2
MII Status: up
Speed: 1000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 08:9e:01:f8:95:0e
Aggregator ID: 1
LACP bypass priority: 0
Slave queue ID: 0
```

To quit the `less` command, type q.

6. Configure a bridge containing two untagged members (switch access ports).

The next steps apply to both leaf 1 and leaf 2.

On both switches, configure ports swp3 and swp4 to be members of a bridge called *bridge* by adding the following stanzas to the `/etc/network/interfaces` file. This creates the topology illustrated in Figure 5.

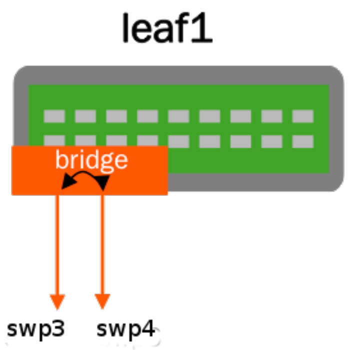


Figure 5. Bridge with Untagged Members

```
~$ sudo nano /etc/network/interfaces
```

```
auto swp3
iface swp3

auto swp4
iface swp4

auto bridge
iface bridge
  bridge-vlan-aware yes
  bridge-ports swp3 swp4
  bridge-vids 1-2000
  bridge-pvid 1
  bridge-stp on
```

```
~$ sudo ifreload -a
```

a.
Configure the bridge on both switches before continuing with this lab.

7. Show the bridge configuration.

```
~$ brctl show

bridge name     bridge id               STP enabled     interfaces
bridge          8000.089e01f89511       yes             swp3
                                                        swp4
```

All hosts connected to either port swp3 or swp4, which are members of bridge, should have connectivity to each other at Layer 2 through the bridge. As there is no gateway specified on these segments, the hosts cannot route outside the subnet.

```
~$ mstpctl showport bridge

*   swp3  8.001 down 8.000.08:9E:01:F8:95:0F 8.000.08:9E:01:F8:95:0F 0.000 Disa
*   swp4  8.002 down 8.000.08:9E:01:F8:95:0F 8.000.08:9E:01:F8:95:0F 0.000 Disa
```

Show the spanning tree topology, like the root port, blocked ports, and so forth.

```
~$ mstpctl showbridge

bridge CIST info
  enabled         yes
  bridge id       8.000.08:9E:01:F8:95:0F
  designated root 8.000.08:9E:01:F8:95:0F
  regional root   8.000.08:9E:01:F8:95:0F
  root port       none
  path cost     0          internal path cost   0
  max age       20         bridge max age       20
  forward delay 15         bridge forward delay 15
  tx hold count 6          max hops             20
  hello time    2          ageing time          300
  force protocol version     rstp
  time since topology change 262s
  topology change count      1
  topology change            no
  topology change port       swp3
  last topology change port  None
```

Show what VLANs are available on which ports.

```
~$ bridge vlan show

port    vlan ids
swp3     1 PVID Egress Untagged
         2-2000

swp4     1 PVID Egress Untagged
         2-2000

bridge   None
```

Sample Answers for Lab 2

Here are sample **`/etc/network/interfaces`** files you can use to check your work.

switch 1:

```
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet dhcp

auto swp1
iface swp1

auto swp2
iface swp2

auto bond0
iface bond0
  bond-slaves swp1 swp2
  bond-mode 802.3ad
  bond-miimon 100
  bond-lacp-rate 1
  bond-min-links 1
  bond-xmit-hash-policy layer3+4

auto swp3
iface swp3

auto swp4
iface swp4

auto bridge
iface bridge
  bridge-vlan-aware yes
  bridge-ports swp3 swp4
  bridge-vids 1-2000
  bridge-pvid 1
  bridge-stp on
```

switch 2:

```
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet dhcp

auto swp1
iface swp1

auto swp2
iface swp2

auto bond0
iface bond0
  bond-slaves swp1 swp2
  bond-mode 802.3ad
  bond-miimon 100
  bond-lacp-rate 1
  bond-min-links 1
  bond-xmit-hash-policy layer3+4

auto swp3
iface swp3

auto swp4
iface swp4

auto bridge
iface bridge
  bridge-vlan-aware yes
  bridge-ports swp3 swp4
  bridge-vids 1-2000
  bridge-pvid 1
  bridge-stp on
```

Bonus Exercise: Using netshow

1. **Install `netshow`.**

   `netshow` is a strong package of support tools built on the Python language.

   Install the needed packages:

   ```
   cumulus@leaf1$ sudo apt-get update
   ```

   ```
   cumulus@leaf1$ sudo apt-get install netshow
   ```

2. **Show networks configurations with `netshow`.**

   In the Cumulus VX environment, you cannot run all of the **`netshow`** options, but here are two that work.

```
~$ netshow lldp
...
Local Port    Speed    Mode                Remote Port    Remote Host
------------  -------  ----------  ----  -------------  --------------
swp1          10G      BondMember  ====  swp1           cumulus
swp2          10G      BondMember  ====  swp2           cumulus
swp3          10G      Trunk/L2    ====  swp4           cumulus
                                   ====  swp3           cumulus
                                   ====  swp4           cumulus
swp4          10G      Trunk/L2    ====  swp3           cumulus
                                   ====  swp3           cumulus
                                   ====  swp4           cumulus
```

```
~$ netshow interface
...
    Name    Speed    MTU    Mode        Summary
--  ------  -------  -----  ----------  -------------------
UP  bond0   20G      1500   Bond/L3     Bond Members: swp1(UP), swp2(UP)
UP  bridge  N/A      1500   Bridge/L2   Untagged Members: swp3-4
                                        802.1q Tag: Untagged
                                        STP: RootSwitch(32768)
                                        Vlan Aware Bridge
UP  eth0    10G      1500   Mgmt        IP:  10.0.2.15/24(DHCP)
UP  lo      N/A      16436  Mgmt        IP:  127.0.0.1/8,  ::1/128
UP  swp1    10G      1500   BondMember  Master: bond0(UP)
UP  swp2    10G      1500   BondMember  Master: bond0(UP)
UP  swp3    10G      1500   Trunk/Lw    Vlans: 1-2000
                                        Native: 1
UP  swp4    10G      1500   Trunk/Lw    Vlans: 1-2000
                                        Native: 1
```

This completes Lab 2.

Other Great Items Available to Extend the Usage of the Cumulus VX VM

GitHub

The Cumulus Networks Customer Engineering team has been hard at work to provide easy-to-use demos to help end users get demos and test environments set up quickly.

https://github.com/CumulusNetworks/cumulus-vx-vagrant/

In this repo you have access to help you build the following demos:

- clos-bgp
- clos-ospf unnumbered
- MLAG

Note: Because you need Ansible to run these demos, they are currently not accessible on Windows.

Cumulus Networks Open Networking Community

Our Cumulus Networks Community is here to assist in any way we can. It provides a place for users to ask questions, get information, and share experiences with others in our community.