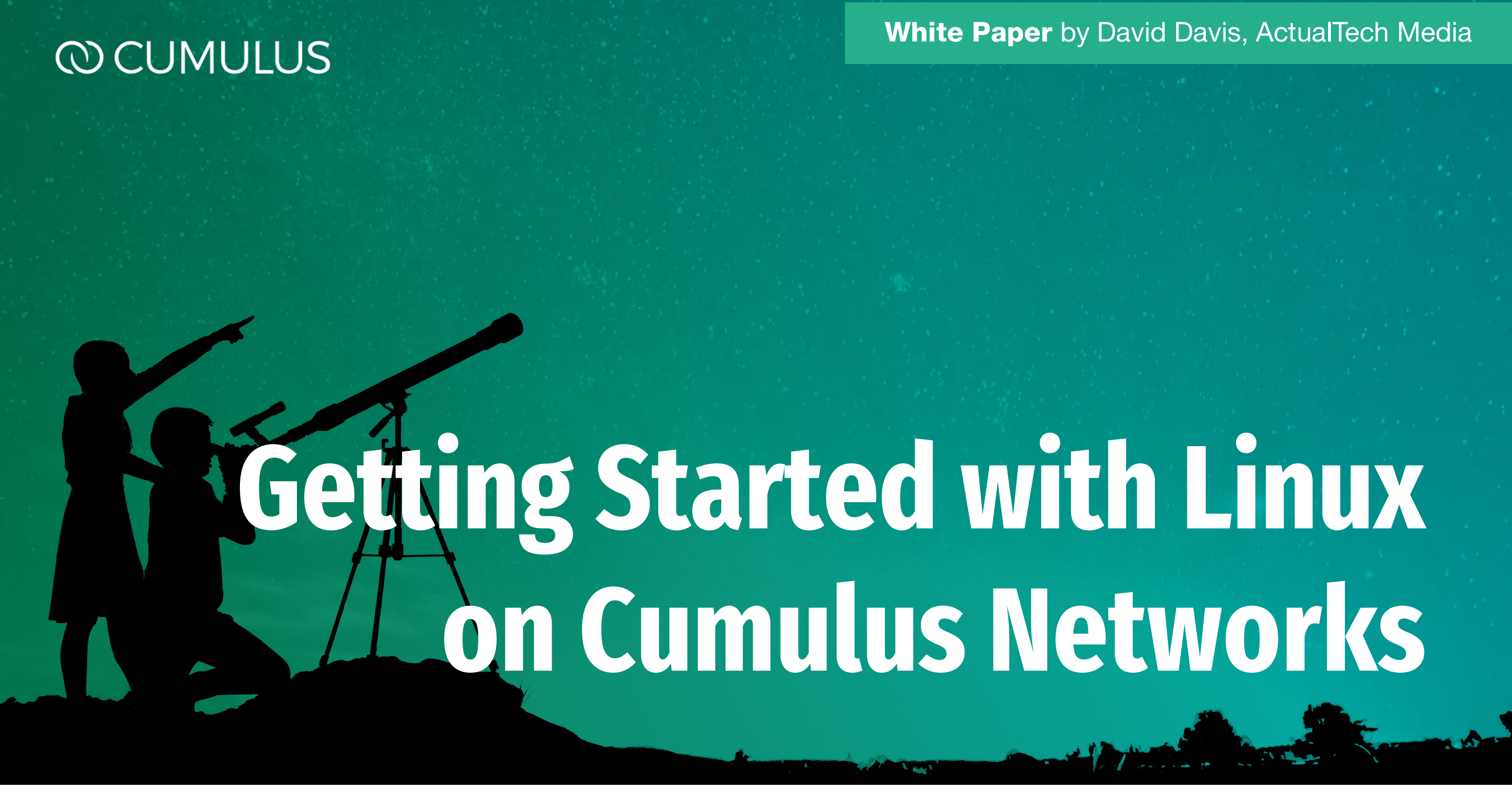
White Paper by David Davis, ActualTech Media



Getting Started with Linux on Cumulus Networks

In this Paper

Introduction

Cumulus Linux is a Linux distribution that is focused on layer 2 and layer 3 internetworking — switching and routing, respectively — using off-the-shelf whitebox switching hardware to accelerate the packet processing that would be done by software on traditional servers (see Figure 1). This allows networking hardware controlled by Cumulus Linux to achieve packet processing rates and functionality on par with traditional switching and routing vendors such as Cisco Systems, Juniper Networks, and others.

It's important to know that the high-performance switched Ethernet interfaces in Cumulus Linux are accelerated by hardware, and Cumulus has made the decision to prefix these devices with "swp" (short for *switch port*). Even if you went to another Linux distribution and renamed the "eth" ports with "swp," they wouldn't be accelerated because they're missing Cumulus' secret sauce.

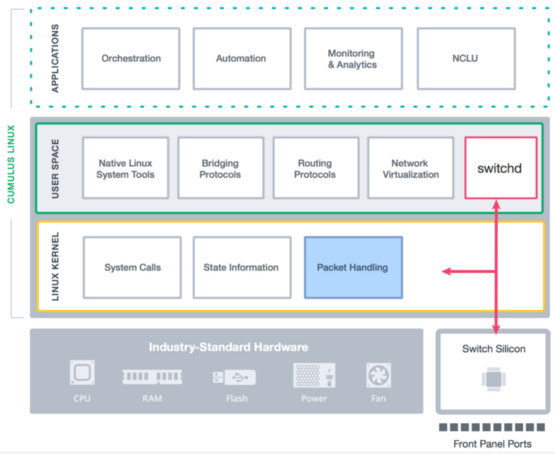


Figure 1. The Cumulus Linux architecture

Linux at the Core

Cumulus Linux makes complete use of the Linux networking model that is widely used by Linux hosts and other devices, around the world. All of the standard Linux networking tools work on Cumulus Linux systems. The Cumulus Linux distribution stays on the leading edge of Linux networking, contributing to the rapid advancement of Linux kernel networking functionality.

Latest and Greatest Networking Protocols

Internetworking requires protocols to interact with network peers and provide the services for the underlying network model. For example, if you want a Linux system to act as a router without having to painfully and manually manage what could be thousands of static routing rules, you need to run a routing protocol, such as BGP, on the Linux system so that you can automatically share IP routes with the rest of the network. Cumulus Linux uses the following:

- **FRRouting**. OSPF, BGP, PIM routing protocols for layer 3 internetworking
- **mstpd**. 802.1 spanning tree protocols for layer 2 internetworking
- **Linux kernel LACP**. For link aggregation
- **MLAG**. For multi-chassis link aggregation

Hardware acceleration

While Linux networking can work on just about any hardware, Cumulus Linux is best run on commodity bare-metal switches that are hardware accelerated. The “hardware acceleration” portion of that means that the switches contain hardware called *ASICs*, specially designed to switch frames and route packets, similar to how a graphics card is specially designed for graphics. These ASICs are what make routers and switches different from regular servers and allow them to process hundreds of gigabits or even terabits of network traffic per second.

Network Command Line Utility (NCLU)

Linux networking offers numerous commands, many of which seemingly perform similar functions, and numerous text files that can be edited to manage network configurations.

This complexity occurs because baseline Linux networking relies on the wide variety of tools. However, the differences between these Linux networking tools can be daunting, even for experienced Linux administrators. Imagine now just how overwhelming they may be for those who are early in their Linux networking journey.

Instead of trying to administer a Linux-powered network with hundreds of command and configuration files, Cumulus Linux includes a command line utility as part of the NCLU package that is invoked by the **`net`** command to provide a consistent and helpful user interface.

As we explore internetworking use cases throughout this paper, I will lean on net and show you how to leverage it while using Cumulus Linux. One of the most useful facilities in NCLU is the built-in help and examples (as shown below):

```
$ net help
Usage:
    # net <COMMAND> [<ARGS>] [help]
    #
    # net is a command line utility for networking on Cumulus Linux switches.
    #
    # COMMANDS are listed below and have context specific arguments which can
    # be explored by typing "<TAB>" or "help" anytime while using net.
    #
    # Use 'man net' for a more comprehensive overview.
    net abort
    net commit [verbose] [confirm] [description <wildcard>]
    net commit delete (<number>|<number-range>)
    net commit permanent <wildcard>
    net del all
    net help [verbose]
    net pending [json]
    net rollback (<number>|last)
    net rollback description <wildcard-snapshot>
    net show commit (history|<number>|<number-range>|last)
    net show rollback (<number>|last)
    net show rollback description <wildcard-snapshot>
    net show configuration [commands|files|acl|bgp|multicast|ospf|ospf6|interface
<interface>]
Options:
    # Help commands
    help     : context sensitive information; see section below
    example  : detailed examples of common workflows
    # Configuration commands
    add      : add/modify configuration
    del      : remove configuration
    # Commit buffer commands
    abort    : abandon changes in the commit buffer
    commit   : apply the commit buffer to the system
    pending  : show changes staged in the commit buffer
    rollback : revert to a previous configuration state
    # Status commands
    show     : show command output
    clear    : clear counters, BGP neighbors, etc
```

Building a Better Bridge

One of the most basic networking use cases is a single transparent bridge. In our example, we'll put the interfaces named swp1, swp2, and swp3 into a transparent bridge with swp3 connecting back into our layer 2 bridge infrastructure (Figure 2).

```
net add bridge bridge ports swp1,swp2,swp3
net commit
```

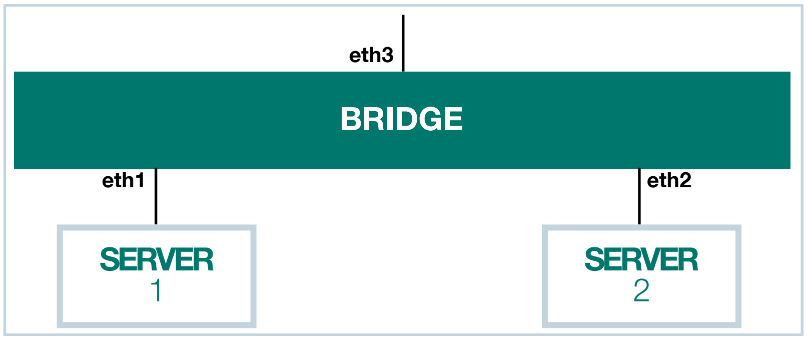


Figure 2. Transparent bridge using spanning tree

This simple example has a few noteworthy things going on. The first is that we don't need to use the **`sudo`** command for privileged access. NCLU makes sure that the user has permission to invoke privileged commands (or belongs to a group that has permission). The second is that **`net`** puts commands into a "commit buffer" so that you can issue a bunch of commands, review them in a pending state (with **`net pending`**), correct them as needed, and then "commit" them to the system with **`net commit`**.

Two Links Are Better Than One

A very typical layer 2 edge use case is using two switches to act like one in a bond to a server. This provides for link- and switch-level redundancy. These types of connections are called *multi-chassis link aggregations* (MLAG), and they are typical of server connections in just about any server deployment.

Figure 3 shows such a deployment with swp1 and swp2 connected to servers (each is part of an 802.1ad bond on the server side), swp3 and swp4 connected back to the network core, and swp5 and swp6 acting as "peer links" between the two switches that form the redundant pair. In the example, 100 VLANs are trunked to each of the servers. Try **`net example clag`** for a few MLAG use cases and **`net example clag l2-with-server-vlan-trunks`** for something close to what is described here:

```
$ net add clag peer sys-mac 44:38:39:FF:00:01 interface swp5,swp6 primary
$ net add vlan 100-199
$ net add clag port bond bond-to-spines interface swp3-4 clag-id 500
$ net add clag port bond host-01 interface swp1 clag-id 1
$ net add clag port bond host-02 interface swp1 clag-id 2
$ net commit
```

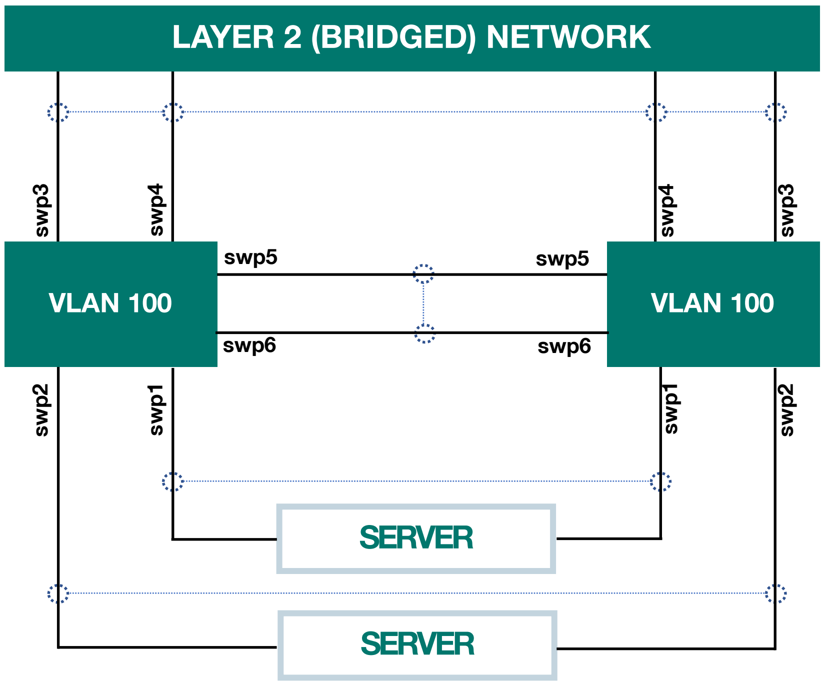


Figure 3. Two switches connected with MLAG

IP Fabrics Are Easy

A recent trend in modern network architecture (especially in data centers) is to reduce the size of the broadcast domains and use layer 3 (IP routed) internetworking to create *fabrics*. A fabric is a simple, high-speed, layer 3 network. The motivation behind this trend is that IP networks scale better than layer 2 networks and behave better in the face of unfortunate misconfigurations and failures.

Traditionally, layer 3 fabrics have been complex to configure because every interface on a switch/router needs to exist on an IP subnet with its link peer — a painstaking undertaking. Recent implementations of BGP and OSPF, such as FRRouting in Cumulus Linux, include the ability to connect routers via point-to-point links using "unnumbered" interfaces.

Figure 4 shows the configuration of a leaf switch in a layer 3 leaf-spine network built using BGP unnumbered. The leaf switch has a bridge with swp1-4 that has the 10.0.0.0/24 IPv4 subnets. Swp5 through swp8 are connected to spines using BGP unnumbered, advertising reachability of the bridge subnets to the rest of the network.

BGP unnumbered uses automatically assigned IPv6 addresses on the unnumbered interfaces. There is no requirement for the loopback interface but it is recommended in order to uniquely identify the network device that is putting routes into BGP (the BGP Router-ID).

```
$ net add bgp autonomous-system 65001
$ net add loopback lo ip address 10.1.0.1/32
$ net add bgp ipv4 unicast network 10.1.0.1/32
$ net add vlan 1 ip address 10.0.0.1/24
$ net add bgp ipv4 unicast network 10.0.0.1/24
$ net add bgp neighbor swp5-8 interface remote-as external
$ net add bgp ipv4 unicast neighbor swp5-8 activate
$ net add bridge bridge ports swp1-4
$ net commit
```

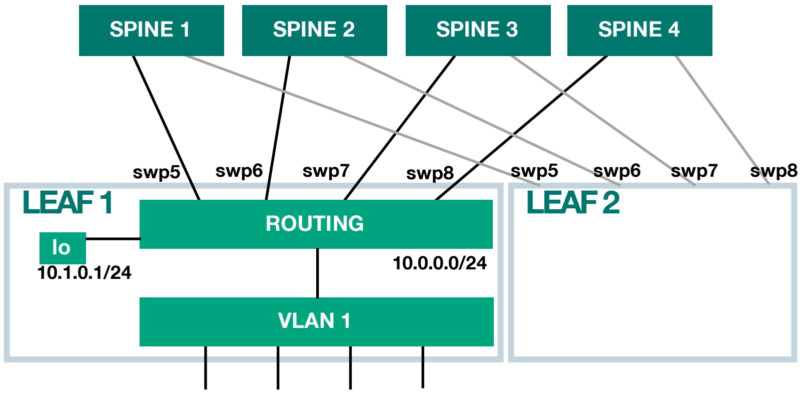


Figure 4. Creating a Layer 3 fabric with BGP

BGP EVPN: L3 Network Virtualization for Network Engineers

Many networks have the scale that requires layer 3 internetworking; however, some applications still require layer 2 peering over the layer 3 fabric. One example of where this can be extremely useful is VMware's vMotion. The Ethernet Virtual Private Networks facilities built into FRRouting's BGP daemon allows us to use BGP to build both the IP fabric as well as any distributed layer 2 overlays that are needed to support your applications. BGP EVPN will take any MAC address learned and advertise it to the remote EVPN peers. This allows each leaf in the network to know where to send the Layer 2 VxLAN traffic without flooding or the need for spanning tree.

This example builds on the network defined in the last section on IP fabrics. This time, VLAN 100 provides layer 2 connectivity via an overlay network. You'll add a Virtual Tunnel EndPoint (VTEP) to VLAN 100, send that VLAN tagged to all the hosts on the bridge, and advertise all the layer 2 hosts to the rest of the network with EVPN.

The details of the configuration show a leaf switch in a layer 3 leaf-spine network built using BGP unnumbered. The leaf switch has a default VLAN (1) with swp1-4 that has the 10.0.0.0/24 IPv4 subnet and a second VLAN (100) on swp1-4 that is tagged. Swp5 through swp8 are connected to spines using BGP unnumbered, advertising reachability of the bridge subnets to the rest of the network. VLAN 100 also has a VTEP and is advertised via BGP EVPN. (See Figure 5.)

```
$ net add bgp autonomous-system 65001
$ net add loopback lo ip address 10.1.0.1/32
$ net add bgp ipv4 unicast network 10.1.0.1/32
$ net add vlan 1 ip address 10.0.0.1/24
$ net add bgp ipv4 unicast network 10.0.0.1/24
$ net add interface swp1-4 bridge trunk vlans 100
$ net add vxlan vtep100 vxlan id 100
$ net add vxlan vtep100 vxlan local-tunnelip 10.1.0.1
$ net add vxlan vtep100 bridge access 100
$ net add vxlan vtep100 bridge learning off
$ net add vxlan vtep100 mtu 9216
$ net add bgp neighbor swp5-8 interface remote-as external
$ net add interface swp5-8 mtu 9216
$ net add bgp neighbor swp5-8 interface remote-as external
$ net add bgp ipv4 unicast neighbor swp5-8 activate
$ net add bgp evpn neighbor swp5-8 activate
$ net add bgp evpn advertise-all-vni
$ net commit
```

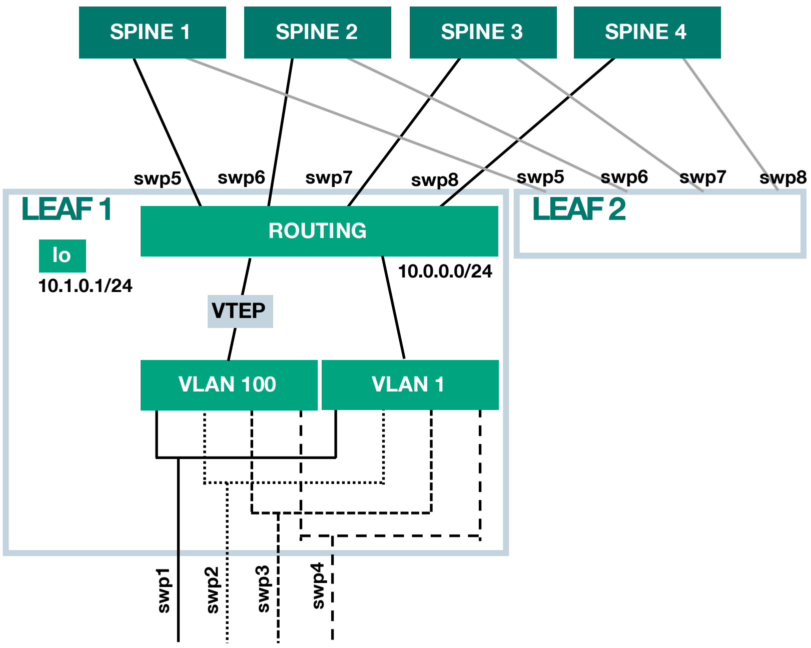


Figure 5. Creating a Layer 3 Fabric with EVPN

Note: The VXLAN header used to build the layer 2 network in EVPN makes Ethernet frames larger than the default of 1518, so you need to include the *maximum transmission unit* (MTU). In this case, set it to 9216 (large enough to support "jumbo" frames) so that you don't have to worry about it ever again.

These use cases are just four examples of how Linux networking can be easy, efficient, and powerful. If you'd like to try out more use cases and commands, we recommend downloading Cumulus VX, a free prototyping environment where you can test out your new Linux networking skills.

Your Cumulus Linux Action Plan

Tons of excellent resources are available in the Linux community, including blogs, documentation, and videos. As a part of that community, Cumulus Networks offers a plethora of learning resources as well. Here's your action plan for taking the next step with Cumulus Linux. Here's my recommendation:

Step 1: Gain Access to Cumulus Linux. You can do so in two different ways:

- **Access Cumulus in the Cloud.** This is a pre-built virtual Cumulus lab environment available at no cost. You can sign up to access Cumulus in the Cloud at https://cumulusnetworks.com/products/cumulus-in-the-cloud/

OR

- **Download the Cumulus virtual appliance, Cumulus VX.** Cumulus VX is available for VMware, VirtualBox, KVM, and Vagrant. The Getting Started Guide will walk you through how to deploy it, power it on, log in, and start configuring all the Linux networking that was demonstrated in this paper. Download Cumulus VX at https://cumulusnetworks.com/products/cumulus-vx/

Step 2: Check out the Cumulus Learning Resources at https://cumulusnetworks.com/learn/web-scale-networking-resources/

Here you'll find case studies, videos, validated designs, and white papers that will show how Cumulus Linux is being used in real data centers around the world.

Step 3: Join the conversations about Cumulus Linux in the Cumulus Slack Community at https://cumulusnetworks.slack.com/