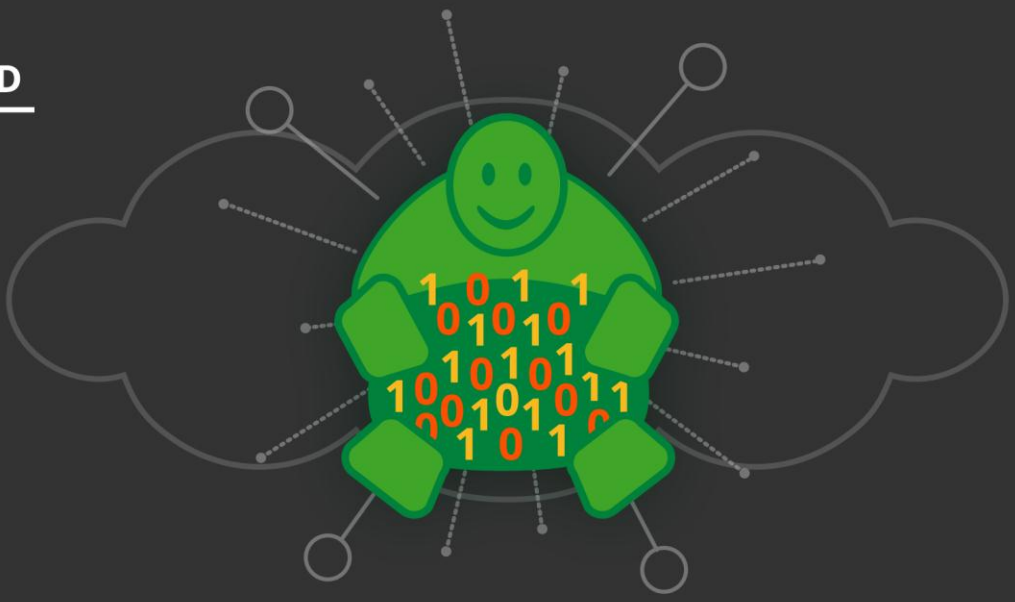
cumulus®

VALIDATED
DESIGN



Big Data and Cumulus® Linux® Validated Design Guide

Deploying Apache Hadoop with Network Switches Running Cumulus® Linux®

Contents

Version 1.0.1

June 24, 2016

About Cumulus Networks

Unleash the power of Open Networking with Cumulus Networks. Founded by veteran networking engineers from Cisco and VMware, Cumulus Networks makes the first Linux operating system for networking hardware and fills a critical gap in realizing the true promise of the software-defined data center. Just as Linux completely transformed the economics and innovation on the server side of the data center, Cumulus Linux is doing the same for the network. It is radically reducing the costs and complexities of operating modern data center networks for service providers and businesses of all sizes. Cumulus Networks has received venture funding from Andreessen Horowitz, Battery Ventures, Sequoia Capital, Peter Wagner and four of the original VMware founders. For more information visit cumulusnetworks.com or @cumulusnetworks.

Big Data with Cumulus Linux

Objective

This Validated Design Guide presents a design and implementation approach for deploying big data analytics on network switches running Cumulus Linux. This design uses the Apache™ Hadoop® project as an example implementation of a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models, scalable from single to thousands of machines, each offering local compute and storage.

Network Demands of Apache Hadoop

Big data is in some ways, **the** application that made the modern data center as we know it. Big data applications build resiliency into the application rather than rely on an infallible network, and require a communication medium that scales from a few nodes to tens or even hundreds of thousands of nodes. Specifically, big data applications use a lot of inter-server communication and are not tied to behaving as if all nodes are on the same subnet. In other words, big data applications are served well by Layer 3 fabrics built on a Clos topology.

Apache Hadoop is a software framework that supports large-scale distributed data analysis on commodity servers, typically on Linux-based compute nodes with local disks.

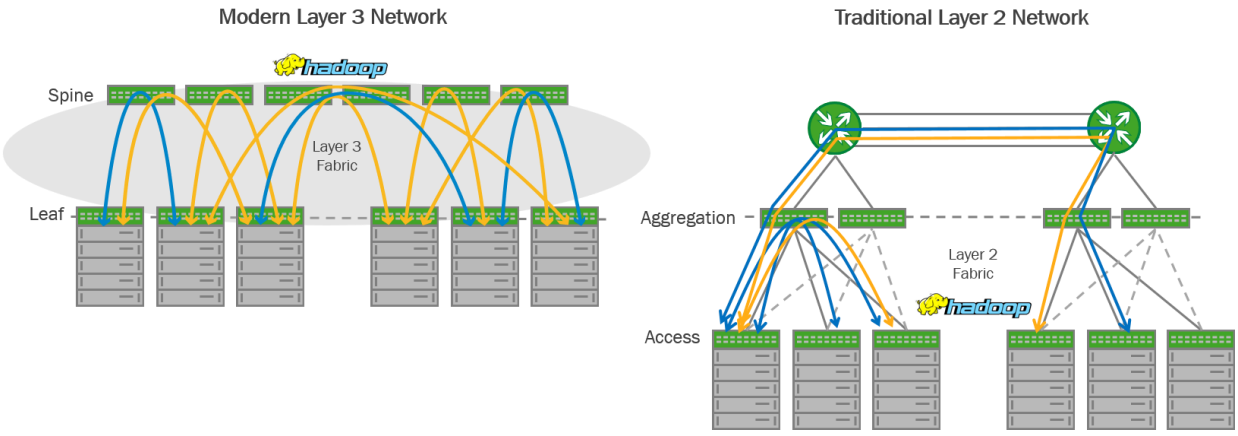


Figure 1. Network Traffic Patterns of Hadoop

Hadoop is a leading example of a modern data storage and processing platform, and is ideally deployed in conjunction with a modern data center infrastructure. In order to best leverage the scale-out capabilities of MapReduce and YARN available in Hadoop, compute and storage resources should be deployed using a high performance, Layer 3 Clos "leaf and spine" network fabric.

An example and popular distribution of Apache Hadoop is Hortonworks Data Platform (HDP), a 100% open source, enterprise grade Hadoop distribution. Hortonworks is a major contributor to open source initiatives (Apache Hadoop, HDFS, Pig, Hive, HBase, Zookeeper) and has extensive experience managing production-level Hadoop clusters.

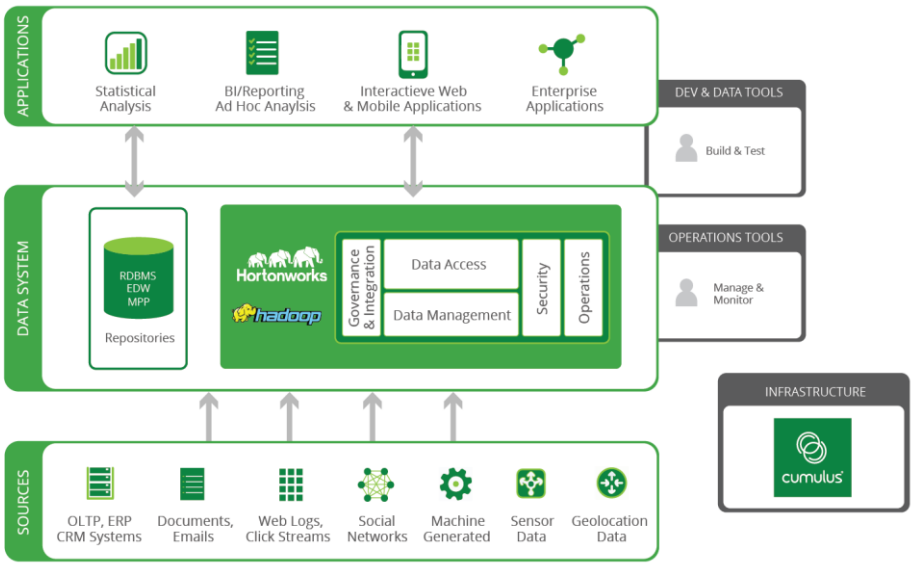


Figure 2. Hortonworks Data Platform and Cumulus Linux

The Power of Open Networking

The path to building a modern Layer 3 network is made easier and more affordable by Cumulus Linux. Rather than locking yourself into a proprietary solution from a specific vendor, you now have choice through open networking. Cumulus Linux gives you the choice of network hardware from a large HCL of vendors at an attractive and affordable price, allowing you to build a Layer 3 network and grow as you need. Cumulus Linux allows you to save capital expense while building larger, higher-performance leaf and spine network fabrics.

Since Hadoop is moving from being primarily batch focused to incorporating real-time and streaming data processing, a non-blocking network fabric powered by Cumulus Linux that can provide comprehensive statistics and programmability is beneficial.

Running Apache Hadoop with Cumulus Linux

The ideal network design for running Hadoop takes a Layer 3 routed model that can be scaled out by adding spines and leafs, or uplinks. The more spines and uplinks that are utilized provide more redundancy, more cross-sectional bandwidth and make the failure domains smaller. A Layer 3 Clos fabric can be designed to be non-blocking or oversubscribed.

From an East-West traffic standpoint, full bandwidth is available between any pair of servers. You can achieve faster recovery with plenty of core bandwidth to re-replicate after failure. The leaf and spine network architecture also provides a consistent distance across the cluster for a low and deterministic latency where every server is equidistant within a pod and equidistant to other pods.

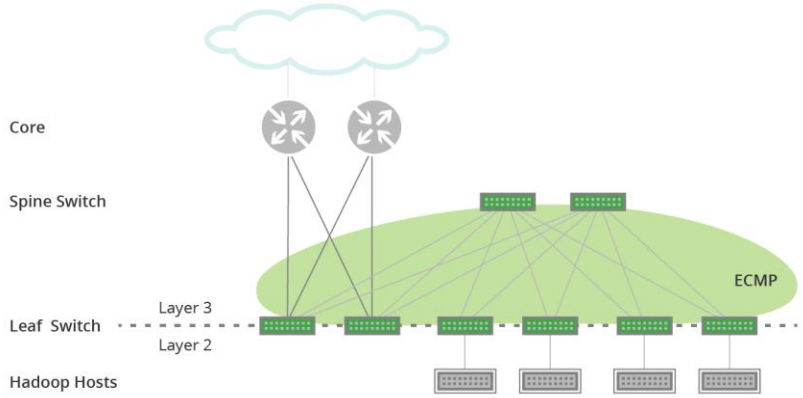


Figure 3. Modern Layer 3 Clos "Spine Leaf" Architecture

Equal-Cost Multipath or ECMP is used to send traffic across all available uplinks and spines. Standard routing protocols such as OSPF or BGP provide a simple failure detection mechanism, and route failures.

Traditional Layer 2 designs using VLANs should be avoided. These designs are brittle by nature with a coarse failure domain involving half of the fabric. A traditional Layer 2 fabric also often involves proprietary protocols. You may be required to send traffic across a core or backbone to another pod or cluster which will create non-deterministic latency and higher latency for some traffic.

To optimize network performance, you can try to run workloads locally where possible. You can leverage the Prescriptive Topology Manager (PTM) and LLDP in Cumulus Linux to map out a complete blueprint of all physical connectivity to eliminate the issues driven from manual cabling or unreachability concerns, as well as extract a rack-aware topology through a simple script for the NameNode. However, location becomes less important from a performance consideration when using a leaf and spine topology.

Scaling Out

The advantage of a Layer 3 Clos network architecture is that you can add additional spine switches as needed to scale horizontally. You can add up to 6 uplinks per leaf switch in a Layer 3 environment, whereas in a Layer 2 environment you are limited to 2 uplinks per access switch.

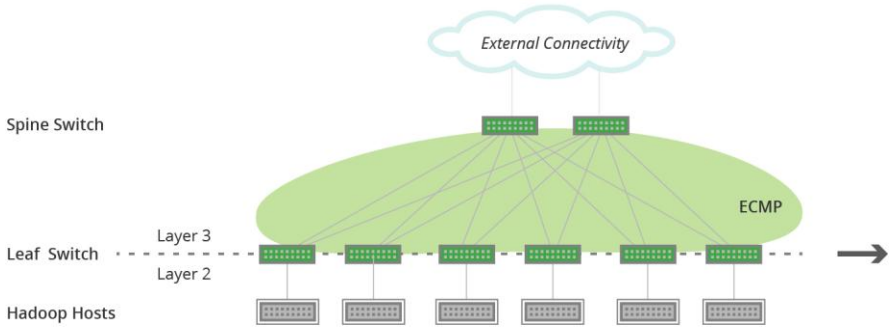


Figure 4. Adding Additional Switches

As you approach the limit for the number of spine switches, you can increase scale by adding additional tiers as shown in Figure 5.

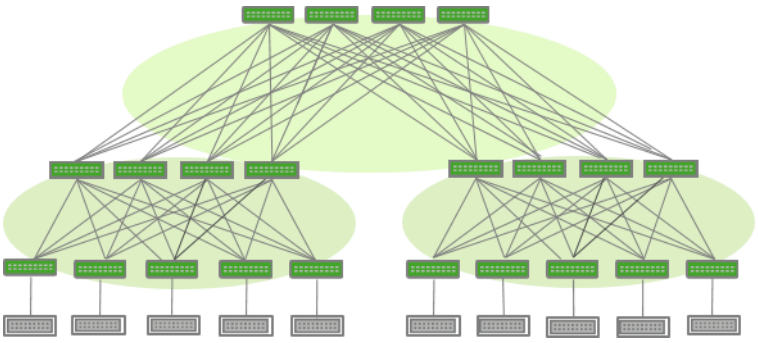


Figure 5. Adding Tiers

Converged Administration

Because Cumulus Linux is Linux and not simply a Linux-based network operating system, you also have the ability to leverage existing server automation tools for managing your switches. While big data users typically install open source solutions like Zookeeper and Ambari to deploy Hadoop clusters, you can also leverage open source and automation/orchestration tools of DevOps' choice such as: Ansible, Chef, Puppet, Salt, or CFEngine. These same tools are already used by many organizations to simplify server deployments, and modifying them to provision entire racks of Hadoop clusters, including both servers and network switches, becomes a simple task of converged administration.

Cumulus Linux also integrates with many open source monitoring tools available for both servers, such as OpenTSDB, Nagios, Ganglia, and Splunk. Converged administration can show both sever and switch data in a similar manner and make correlation easier. The ability to use these common tools unifies your data center operations and can lower operational expenses. The open source nature of Cumulus Linux and the inclusion of programming languages like Python, Perl, Ruby and Bash make it very extensible and easy to modify if desired. Several innovations such as networking plug-ins with Ambari APIs can also be easily built in the future.

Intended Audience for Network Design and Build

The intended audience for the rest of this white paper is a data center architect or administrator experienced with big data Hadoop clusters and familiar with Layer 3 networking for large-scale designs. The network architecture and build steps provided in this document can be used as a reference for planning and implementing big data analytics with Cumulus Linux in your environment. In addition to knowledge of Apache Hadoop configuration, a basic understanding of Linux commands is assumed, such as accessing a Linux installation, navigating the file system, and editing files.

If you are using this guide to help you set up your Cumulus Linux environment to support your big data implementation, we assume you have Cumulus Linux binaries, licenses, and switches from the *Cumulus Linux Hardware Compatibility List (HCL)* at cumulusnetworks.com/hcl. Additional information on Cumulus Linux software, licensing, and supported hardware may be found on cumulusnetworks.com or by contacting sales@cumulusnetworks.com.

Building a Data Center Network with Cumulus Linux to Run Apache Hadoop

The instructions that follow detail the steps needed for designing and building a representative network for Hadoop workloads with switches running Cumulus Linux. The following prerequisites are assumed:

- Users should be familiar with basic text editing, UNIX file permissions, and process monitoring. A variety of text editors are pre-installed, including `vi` and `nano`.
- You must have access to a Linux or UNIX shell. If you are running Windows, you should use a Linux environment like Cygwin as your command line tool for interacting with Cumulus Linux.
- Please review the Hardware Compatibility List (HCL) to confirm your switch model is supported by Cumulus Networks. The HCL is updated regularly, listing products by port configuration, manufacturer, and SKU part number.

Design Steps

The steps for designing a Hadoop network environment with Cumulus Linux switches are:

Step		Tasks
1.	**Diagram your network architecture.**	Specify spine and leaf switches. Specify how switches are connected.
2.	**Determine which IP address each node will use.**	Enumerate network nodes. Enumerate loopback IP addresses. Enumerate VLAN subnets.
3.	**Determine how your cluster will connect to your core network.**	Review documentation.
4.	**Determine the IP addresses that each server will use.**	Enumerate servers. Enumerate IPMI IP addresses. Enumerate static IP addresses.
5.	**Select systems to be master and backup.**	Designate master and backup. Apply DNS as needed.

1. Diagram your Network Architecture.

An example network architecture diagram is shown below in Figure 6.

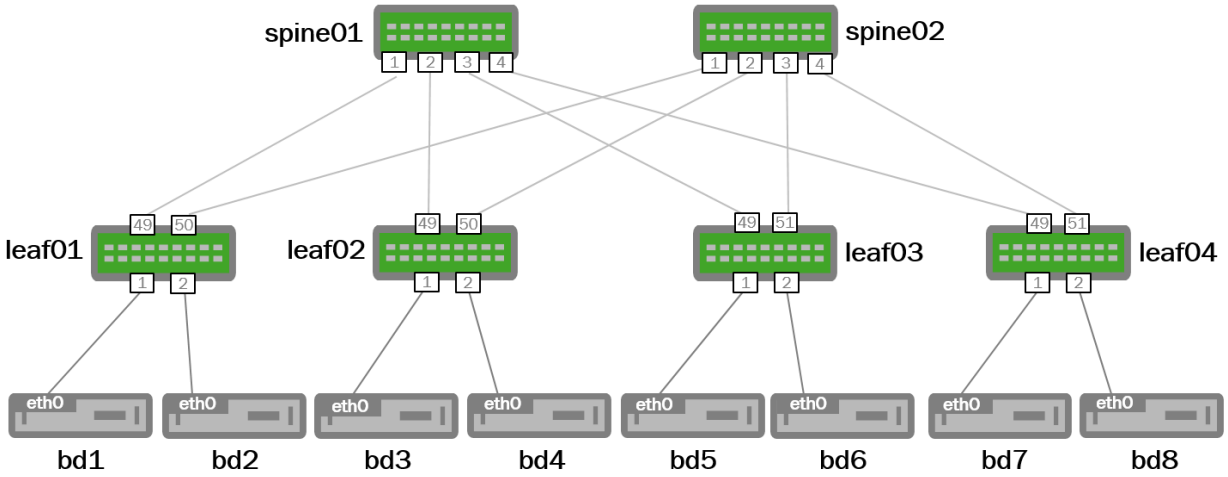


Figure 6. Example Network Architecture

2. Determine which IP Address Each Node Uses

For your site, enumerate your network nodes, their loopback IP addresses and the addresses to use on the leaf switches for VLANs. For example:

Node	Loopback	VLAN Subnet
spine01	10.2.1.3/32	N/A
spine02	10.2.1.4/32	N/A
leaf01	10.2.1.1/32	10.4.1.1/25
leaf02	10.2.1.2/32	10.4.2.1/25
leaf03	10.2.1.5/32	10.4.3.1/25
leaf04	10.2.1.6/32	10.4.4.1/25
edge-leaf01	10.2.1.7/32	N/A
edge-leaf02	10.2.1.8/32	N/A

3. Determining How Your Cluster Connects to Your Core Network

You need to determine which routing protocols and IP addresses your cluster will use to connect to your core network. The details of this are outside of the scope of this document, but you may need to reference the Cumulus Linux Documentation along with documentation for the network devices to which the edge-leaf switches will connect.

4. Determine the IP Address That Your Servers Use

If you are not going to leverage an existing DHCP setup, then you need to manually configure the IP addresses of your servers. Due to the static nature of the Hadoop configuration files, ensuring your master and backup systems have a stable IP address is critical, so it is suggested that they at a minimum have static IP addresses. The worker nodes within the cluster can have dynamic addresses, so long as DNS properly reflects this. The variety of configurations is outside of the scope of this document, so it will focus on a static IP address approach only.

For your site, enumerate your servers, their IPMI IP addresses and your planned static IP addresses. For example:

Server	IPMI IP Address	Static IP Address
bigdata1	10.0.0.1	10.4.1.2
bigdata2	10.0.0.2	10.4.1.3
bigdata3	10.0.0.3	10.4.2.2
bigdata4	10.0.0.4	10.4.2.3
bigdata5	10.0.0.5	10.4.3.2
bigdata6	10.0.0.6	10.4.3.3
bigdata7	10.0.0.7	10.4.4.2
bigdata8	10.0.0.8	10.4.4.3

5. Select Master and Backup Systems

Two nodes are required to run extra functions on them and so act as the master scheduler, HDFS name node, job control system and other functions within the Hadoop cluster. For availability reasons, nominating a backup is also suggested. These two systems should also have as few single common points of failure as possible, so they should be in different racks, connected to different leaf nodes and should have as independent power as is supported in your deployment.

If you choose to have local DNS, this should also be run on the master and backup systems for consistency.

For example, *bigdata1* is the designated master and *bigdata8* is the designated backup. Static configurations are leveraged for DNS and IP addresses.

Build Steps

The steps for building out a Hadoop network environment with Cumulus Linux switches are:

Step		Tasks
Physical Network		
1.	**Set up physical network.**	Rack and cable all network switches. Install Cumulus Linux. Verify all cables are properly connected as expected. Use ptmctl as needed.
2.	**Set up basic configuration of all switches.**	Configure basic items for management such as eth0, hostname and DNS.
Network Topology		
3.	**Configure leaf switches.**	Configure an individual switch and then automate for remaining switches where possible.
4.	**Configure spine switches.**	Configure an individual switch and then automate for remaining switches where possible.
5.	**Configure edge leaf switches.**	Configure switches individually and configuration for connections to core.
6.	**Verify configuration.**	Ensure your topology matches your planning.
7.	**Connect network fabric.**	Connect leaf switches to spine switches.
Host Setup		
8.	**Install base OS.**	Install the OS into each *bigdata#* node.
9.	**Ensure common services are set up.**	DNS, DHCP or other dynamic common services.
10.	**Begin installation of Hadoop.**	Follow the recommended process in the Hortonworks Hadoop guide.

In a green field environment, the order of configuring spine or leaf switches does not matter; that is, they may be reversed. In a brown field environment, start with leaf switches and finish with the connections to the core for minimal network service disruptions.

1. Set Up Physical Network

Rack, stack, and cable your network switches. Install the Cumulus Linux OS and license on each switch. Refer to the *Quick Start Guide* and *Getting Started* sections of the Cumulus Linux Documentation for more information.

2. Set Up Basic Configuration of All Switches

The default configuration for eth0, the management interface, is DHCP. To reconfigure eth0 to use a static IP address, edit the `/etc/network/interfaces` file by adding an IP address/mask and an optional gateway.

The default for the hostname of a switch running Cumulus Linux is *cumulus*. Change this to the appropriate name based on your network architecture diagram, like leaf01 or spine01, by modifying `/etc/hostname` and `/etc/hosts`.

Modify your DNS settings if needed. You can add your domain, search domain, and nameserver entries to the `/etc/resolv.conf` file.

3. Configure Leaf Switches

Define the interfaces you will use on each switch and their roles. This is all done in the `/etc/network/interfaces` file on each switch.

Open your `/etc/network/interfaces` file and add the following sections:

```
# Active ports
<%
  spine-ports = [49,50]
  client-ports =
[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,3
0,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48]
%>
```

Next, set up a policy to set up the client ports:

```
# Create strings for the bridge
<% portstr = ' '.join(client-ports) %>
```

Next, set the MTU on the client-facing ports, set the ports to *edge* ports within STP, and block on seeing BPDU frames:

```
% for i in client-ports:
auto swp${i}
iface swp${i}
  bridge-access 1
  mtu 9000
  mstpctl-portadminedge yes
  mstpctl-bpduguard yes
% endfor
```

Loopback Configuration

Create the global loopback for the switch:

```
auto lo
iface lo inet loopback
```

Assign an IP address to the loopback:

```
  address 10.2.1.1/32
```

Spine Port Configuration

Create a Mako configuration to set the uplink interfaces into an *unnumbered* configuration:

```
% for i in spine-ports:
auto swp${i}
iface swp${i}
  address 10.2.1.1/32
  mtu 9000
% endfor
```

VLAN Configuration

Create the global bridge called *bridge*:

```
auto bridge
iface bridge
```

Assign physical ports to the VLAN and enable STP:

```
  bridge-vlan-aware yes
  bridge-ports ${portstr}
  bridge-pvid 1
  bridge-stp on
```

Assign an IP address to VLAN 1 on the bridge:

```
auto bridge.1
iface bridge.1
  address 10.4.1.1/25
```

You can now exit out of editing your `/etc/network/interfaces` file.

Once these interfaces have been created, apply the configuration by having `ifreload` reread the entire configuration:

```
cumulus@switch:~$ sudo ifreload -a
```

Enable Quagga and Configure Routing

The first step is to enable the `zebra` and `ospf` daemons.

Open your `/etc/quagga/daemons` file and change the two highlighted daemon flags from their default of *no* to *yes*:

```
zebra=yes
bgpd=no
ospfd=yes
ospf6d=no
ripd=no
ripngd=no
```

```
isisd=no
babeld=no
```

Next, provide Quagga with its own configuration file. Create a new `/etc/quagga/Quagga.conf` file with the following:

```
password PASSWORD
enable password PASSWORD
!
interface swp49
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
interface swp50
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
interface bridge
ip ospf area 0.0.0.0
!
router ospf
  auto-cost reference-bandwidth  40000
  ospf router-id 10.2.1.1
  passive-interface default
  no passive-interface swp49
  no passive-interface swp50
```

Once these interfaces have been created, apply the configuration by restarting the network services or individually bringing up the new interfaces:

```
cumulus@switch:~$ sudo service quagga restart
```

4. Configure Spine Switches

Define the interfaces you will use on each switch and their roles.

Open your `/etc/network/interfaces` file and add the following sections:

```
# Active ports
<%
  spine-ports =
[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,3
0,31,32]
%>
```

Loopback Configuration

Create the global loopback:

```
auto lo
iface lo inet loopback
```

Assign an IP address to the loopback:

```
  address 10.2.1.3/32
```

Spine Port Configuration

Next, create all interfaces listed in the above Mako set.

```
% for i in spine-ports:
auto swp${i}
iface swp${i}
  address 10.2.1.3/32
  mtu 9000
% endfor
```

You can now exit out of editing your `/etc/network/interfaces` file.

Once these interfaces have been created, apply the configuration by having `ifreload` reread the entire configuration:

```
cumulus@switch:~$ sudo ifreload -a
```

Enable Quagga and Configure Routing

The first step is to enable the `zebra` and `ospf` daemons.

Open your `/etc/quagga/daemons` file and change the two highlighted daemon flags from their default of *no* to *yes*:

```
zebra=yes
bgpd=no
ospfd=yes
ospf6d=no
ripd=no
ripngd=no
isisd=no
babeld=no
```

Next, provide Quagga with its own configuration file.

Create a new `/etc/quagga/Quagga.conf` file with the following content:

```
password PASSWORD
enable password PASSWORD
!
interface swp1
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
interface swp2
```

```
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
interface swp3
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
interface swp4
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
interface swp5
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
interface swp6
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
interface swp7
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
interface swp8
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
interface swp9
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
interface swp10
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
interface swp11
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
interface swp12
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
interface swp13
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
interface swp14
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
```

```
!
interface swp15
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
interface swp16
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
interface swp17
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
interface swp18
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
interface swp19
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
interface swp20
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
interface swp21
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
interface swp22
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
interface swp23
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
interface swp24
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
interface swp25
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
interface swp26
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
```

```
interface swp27
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
interface swp28
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
interface swp29
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
interface swp30
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
interface swp31
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
interface swp32
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
router ospf
  auto-cost reference-bandwidth  40000
  ospf router-id 10.2.1.3
  passive-interface
  no passive-interface swp49
  no passive-interface swp50
```

Once these interfaces have been created, apply the configuration by restarting the network services or individually bringing up the new interfaces:

```
cumulus@switch:~$ sudo service quagga restart
```

5. Configure the Edge Leaf Switches

Define the interfaces you will use on each switch and their roles.

Open your `/etc/network/interfaces` file and add the following sections:

```
# Active ports
<%
  spine-ports = [1,2]
%>
```

Loopback Configuration

Create the global loopback:

```
auto lo
iface lo inet loopback
```

Assign an IP address to each bridge:

```
  address 10.2.1.7/32
```

Spine Port Configuration

Next, create all interfaces listed in the above Mako configuration:

```
% for i in spine-ports:
auto swp${i}
iface swp${i}
  address 10.2.1.7/32
  mtu 9000
% endfor
```

Core Port Configuration

Next, configure the core port interfaces.

Note: This configuration is dependent on the core devices you are connecting to. As such, the example is an incomplete reference that is dependent on your individual site setup.

```
auto swp31
iface swp31
  address [IP]/30
  mtu 9000

auto swp32
iface swp32
  address [IP]/30
  mtu 9000
```

You can now exit out of editing your `/etc/network/interfaces` file.

Once these interfaces have been created, apply the configuration by having `ifreload` reread the entire configuration:

```
cumulus@switch:~$ sudo ifreload -a
```

Enable Quagga and Configure Routing

The first step is to enable the `zebra` and `ospf` daemons.

Open your `/etc/quagga/daemons` file and change the two highlighted daemon flag from their default of *no* to *yes*.

Note: If you will be using BGP to talk to your core network then you will need to also enable `bgpd` here by changing its value from *no* to *yes*.

```
zebra=yes
bgpd=no
ospfd=yes
ospf6d=no
ripd=no
ripngd=no
isisd=no
babeld=no
```

Next, provide Quagga with its own configuration file.

Create a new `/etc/quagga/Quagga.conf` file with the following content:

```
password PASSWORD
enable password PASSWORD
!
interface swp49
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
interface swp50
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
router-id 10.2.1.7
router ospf
  auto-cost reference-bandwidth  40000
  ospf router-id 10.2.1.32
  passive-interface default
  no passive-interface swp49
  no passive-interface swp50
```

For all other protocols you are going to run to your core network, you will also need to add those to the Quagga config. Please see the Cumulus Linux Documentation for assistance here.

Once these interfaces have been created, apply the configuration by restarting the network services or individually bringing up the new interfaces:

```
cumulus@switch:~$ sudo service quagga restart
```

6. Verify Configuration

Prepare a basic configuration — hostname, DNS, out-of-band management. Use DHCP for eth0 for basic management connectivity. Run LLDP to check neighbors and connectivity. Here is some sample output for LLDP:

```
cumulus@leaf1:~$ sudo lldpctl
-------------------------------------------------------------------------------
LLDP neighbors:
-------------------------------------------------------------------------------
Interface:    swp49, via: LLDP, RID: 3, Time: 28 days, 19:33:36
  Chassis:
    ChassisID:    mac 6c:64:1a:00:51:1e
    SysName:      spine01
    SysDescr:     Cumulus Linux
    MgmtIP:       10.2.1.3
    Capability:   Bridge, off
    Capability:   Router, on
  Port:
    PortID:       ifname swp1
    PortDescr:    swp1
-------------------------------------------------------------------------------
Interface:    swp50, via: LLDP, RID: 2, Time: 28 days, 19:33:36
  Chassis:
    ChassisID:    mac 6c:64:1a:00:4f:9b
    SysName:      spine02
    SysDescr:     Cumulus Linux
    MgmtIP:       10.2.1.4
    Capability:   Bridge, off
    Capability:   Router, on
  Port:
    PortID:       ifname swp1
    PortDescr:    swp1
-------------------------------------------------------------------------------
```

To verify that OSPF is up and running correctly, run `cl-ospf neighbor`:

```
cumulus@leaf1:~$ sudo cl-ospf neighbor

Neighbor ID     Pri State           Dead Time Address         Interface
10.2.1.3          1 Full/DROther      33.787s 10.2.1.3        swp49:10.2.1.1
10.2.1.4          1 Full/DROther      33.788s 10.2.1.4        swp50:10.2.1.1
```

7. Connect Network Fabric

You can now connect up your Hadoop network to your core network. You should then verify that your cluster brings up the routing protocols and receives routes from the rest of your network. Please see the Cumulus Linux Documentation for details on how to do this.

8. Install Base OS for Hadoop Nodes

At this point you can begin installing your base OS on your *bigdata#* nodes. The steps may vary for your specific platform, but the general cases are:

- Download the ISO for the OS you are going to install.
  - For Ubuntu 14.04, please see the Ubuntu website.
- For the IPMI configuration supported by your server, connect to the virtual console of your server.
- Attach the ISO that you downloaded for your OS as a virtual device for your server.
- Boot the ISO.
- Set the appropriate data as the install runs:
  - Language
  - "Install Ubuntu Server"
  - Manual Network Configuration
    - Refer to the static IP address you selected previously for this node.
  - Hostname
  - Username and password
  - Do **not** encrypt home directories
  - Time zone
  - Partition
    - Your file system should be independent from your HDFS file system.
    - Partitioning depends on the number and size of drives that you have in your systems.
  - Enable Openssh server
  - Install Grub
- Once the install is complete, ensure you disconnect the ISO image so that the system will boot into your new OS.
- Repeat the above steps for each system in your cluster.

As your installs complete, verify you can reach each of them by connecting via SSH to their IP addresses and logging in with the user account you created.

9. Ensure Common Services Are Set Up

If you are leveraging DNS, DHCP or other dynamic common services in your cluster, you should configure them before proceeding.

- For setting up DNS, please see the ISC-Bind9 guides
- For setting up DHCP, please see the ISC-DHCP guides
- For setting up Dynamic DNS via DHCP, please see the ISC Dynamic DNS setup guide

10. Set Up Hadoop

Now that your network is active, all the OSes are installed, and you can reach outside resources, you can begin installing Hadoop. Please follow the steps in the Hadoop Manual Install Guide published by Hortonworks to bring the remaining software components of your big data cluster up.

Once you have completed your Hadoop cluster setup, you can run various tests to ensure the overall operation of the system is functional. Example test cases are:

- TeraGen
- TeraSort
- TeraValidate
- Calculate pi

- TestDFSIO - write
- TestDFSIO - read
- TestDFSIO - multiple write
- TestDFSIO - multiple read
- randomword
- wordcount

Conclusion

Summary

The efficiencies and optimizations of big data analytics are coupled with network demands that require a modern data center to fully realize. Thus, a modern Layer 3 Clos network is the best network architecture to run big data frameworks such as Apache Hadoop.

Cumulus Linux allows big data customers to build highly scalable and affordable Layer 3 network architectures to support the demands of their analytics. Because Cumulus Linux is Linux, customers are also able to achieve operational efficiencies from converged administration of their compute and network devices.

References

Article/Document	URL
Cumulus Linux Documentation • Quick Start Guide • Ethernet Bridging - VLANs • Layer 3 Features • Configuring Quagga • Open Shortest Path First (OSPF) Protocol • Link Layer Discovery Protocol • Prescriptive Topology Manager - PTM	http://docs.cumulusnetworks.com
Cumulus Linux KB Articles • Demos and Training, specifically Implementing the Big Data Design Guide in the Cumulus Workbench	https://support.cumulusnetworks.com/hc/en-us/sections/200398866 https://support.cumulusnetworks.com/hc/en-us/articles/203594808
Cumulus Linux Product Information • Software Pricing • Hardware Compatibility List (HCL)	http://cumulusnetworks.com/product/pricing/ http://cumulusnetworks.com/hcl
ISC • Bind9 • DHCP • Dynamic DNS	https://kb.isc.org/article/AA-01031 https://kb.isc.org/article/AA-00333 https://deepthought.isc.org/article/AA-01091/0/ISC-DHCP-support-for-Standard-DDNS.html
Hortonworks Hadoop Documentation	http://docs.hortonworks.com/HDPDocuments/HDP2/HDP-2.1-latest/bk_installing_manually_book/content/rpm-chap1.html
Ubuntu	http://www.ubuntu.com/download/server

Appendix: Network Setup Checklist

	Tasks	Considerations
1. Set up physical network.		
❑	Select network switches	Refer to the HCL and hardware guides at http://cumulusnetworks.com/hcl. Leaf switches: • Choose between at least a 48 port 10G switch or 32 port 40G switch with breakout cables. • Consider price and future-proofing. Spine switches: • Choose at least a 10G switch or a 40G switch; 40G for more traffic aggregation. • Consider price and future-proofing.
❑	Plan cabling	Refer to KB article, *Suggested Transceivers and Cables*: https://support.cumulusnetworks.com/hc/en-us/articles/202983783. Generally, higher number ports on a switch are reserved for uplink ports, so: • Assign downlinks or host ports to the lower end, like swp1, swp2 • Reserve higher number ports for network Connect all console ports.
❑	Install Cumulus Linux	Obtain the latest version of Cumulus Linux. Obtain license key, which is separate from Cumulus Linux OS distribution. To minimize variables and aid in troubleshooting, use identical versions across switches — same version X.Y.Z, packages, and patch levels. See the *Quick Start Guide* in the *Cumulus Linux Documentation*.
2. Set up basic configuration of all switches.		
❑	Reserve management space	Reserve pool of IP addresses. Define hostnames and DNS. RFC 1918 should be used where possible.
❑	Determine IP addressing	Enumerate: • Loopback IP addresses • VLAN subnets • IPMI IP addresses • Server static IP addresses (if not using DHCP)

	Tasks	Considerations
❑	Edit configuration files	Apply standards and conventions to promote similar configurations. For example, place stanzas in the same order in configuration files across switches and specify the child interfaces before the parent interfaces (so a bond member appears earlier in the file than the bond itself, for example). This allows for standardization and easier maintenance and troubleshooting, and ease of automation and the use of templates. Consider naming conventions for consistency, readability, and manageability. Doing so helps facilitate automation. For example, call your leaf switches leaf01 and leaf02 rather than leaf1 and leaf02. • Use all lowercase for names • Avoid characters that are not DNS-compatible.
3. Configure leaf switches.		
❑	Define switch ports (swp) in /etc/network/interfaces on a switch	Instantiate swp interfaces for using the `ifup` and `ifdown` commands.
❑	Define loopback interface	Loopback interface is created by default. Set IP address on loopback interface.
❑	Define spine port interfaces	Create connections to spine. Use Mako for shorthand, iterative definitions.
❑	Configure routing	Enable Quagga (not enabled by default). Configure Layer 3 routing. OSPF point-to-point is typical. Configure router-id.
❑	Set MTU	By default, MTU is set to 1500. Set to a high value, like 9000, to avoid packet fragmentation.
❑	Set speed and duplex	These settings are dependent on your network.
❑	Automate setup	Use automation for setting up many switches.
4. Configure spine switches.		
❑	Repeat steps for configuring leaf switches	Steps for spine switches are similar.
5. Configure edge leaf switches.		
❑	Repeat steps for configuring leaf switches	Steps for spine switches are similar.

6. Verify configuration.		
❑	Check neighbors	Use LLDP. Use Quagga and `show ip ospf neighbors`. Use `cl-ospf` for non-modal option.
7. Connect network fabric.		
❑	Connect to core.	Check MTU setting for the connection to the core. This depends on what the core needs. Determine how to handle the default route: originate or learn? Decide what IP address to use for the gateway. Typically this is either .1 or .254. Determine if routing will use static or dynamic routing. If dynamic: • Specify `router-id` and advertised networks. • Consider IPv4 and/or IPv6. • Determine protocol, OSPF or BGP.
8. Install base OS for Hadoop Nodes		
❑	Install Ubuntu	Server configuration Partitioning: • File system should be independent from HDFS file system. • Partitioning depends on number and size of drives in your systems. Enable Openssh server Install Grub
9. Ensure common services are set up		
❑	Set up common services	Configure the following as needed: • DHCP • DNS • Dynamic DNS
10. Set up Hadoop		
❑	Configure Hadoop Test	DFS up and running Smoke test run and complete MapReduce, TeraGen, TeraSort, TeraValidate DFS performance testing Environment operational