Who is Leslie?

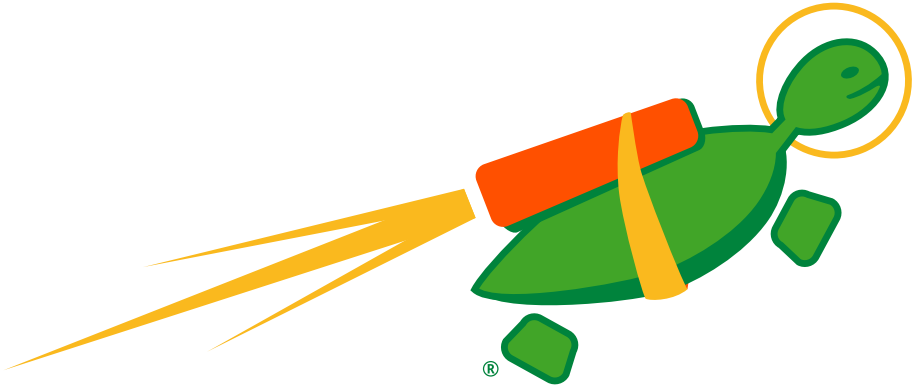
cumulus

Operations Engineer

- @lesliegeek
- Google
- Craigslist
- Twitter
- Wikimedia Foundation (Wikipedia)

Who is Leslie?

Cumulus Networks®



What is Cumulus Linux?

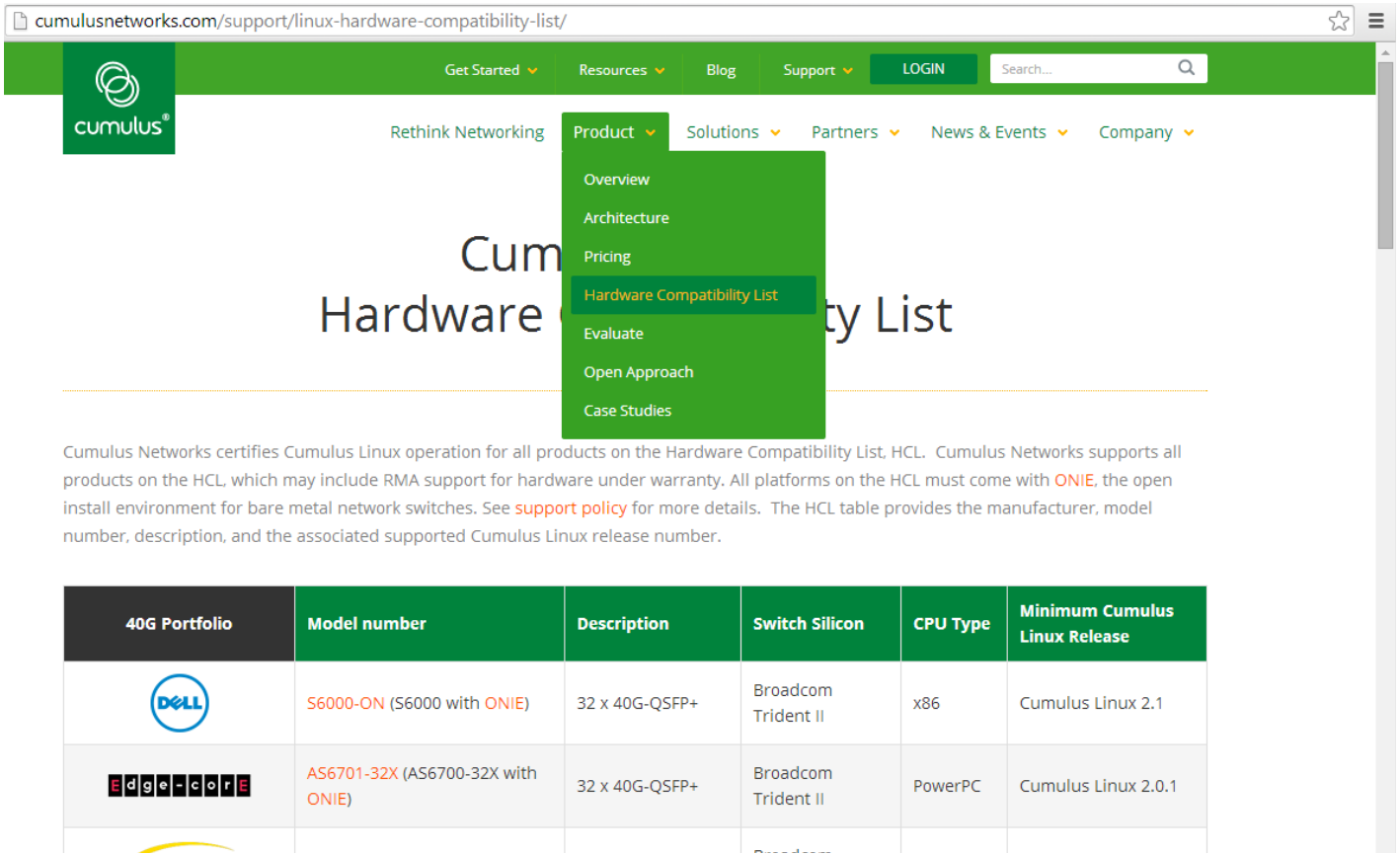
Cumulus® Linux®

- Debian based distribution for Network switches

Cumulus Philosophy

- Manage your network switch as a server
- Use existing linux tools to configure network switches
- Current release — Based on Debian wheezy

Cumulus® Linux® Hardware Compatibility List



Installation and Configuration

Traditional Network Gear Process

- Comes preinstalled with (old) software
- Telnet or serial into box
- TFTP new image
- Enable SSH (sometimes with passwords!)
- Copy/paste configuration
- Automation usually restricted to Perl, TCL, and expect scripts

Normal Server Installation

- Reboot and bios with PXE automatically catches
- TFTP boot image
- New image pulled over via normal means (usually webserver, sometimes TFTP)
- Pre/post installation scripts runs
- Automation software manages configuration and administration

ONIE

cumulus

Network OS installer discovery and execution

- Like a pre-installed BIOS, PXE, and kickstarter in one
- Implemented through Linux kernel with BusyBox

Donated to the Open Compute Project (OCP)

http://www.onie.org

OPEN
Compute Project

1 Look for installer (“discover”)

- Locally, e.g. USB if available
- Over the network on eth0

  DHCP, IPv6 neighbor, TFTP

2 Search for file name and execute

- `onie-installer-*`

ZTP

During the DHCP process over eth0 (management interface), Cumulus Linux will request DHCP option 239. This option is used to specify the custom provisioning script. It will also send the following headers:

Header	Value	Example
User-Agent		CumulusLinux-AutoProvision/0.4
CUMULUS-ARCH	CPU architecture	powerpc
CUMULUS-BUILD		1.5.1-5c6829a-201309251712-final
CUMULUS-LICENSE-INSTALLED	Either 0 or 1	1
CUMULUS-MANUFACTURER		dni
CUMULUS-PRODUCTNAME		et-7448bf
CUMULUS-SERIAL		XYZ123004
CUMULUS-VERSION		1.5.1
CUMULUS-PROV-COUNT		0
CUMULUS-PROV-MAX		32

cumulus

Script must contain `CUMULUS-AUTOPROVISIONING`

Can be in the following languages :

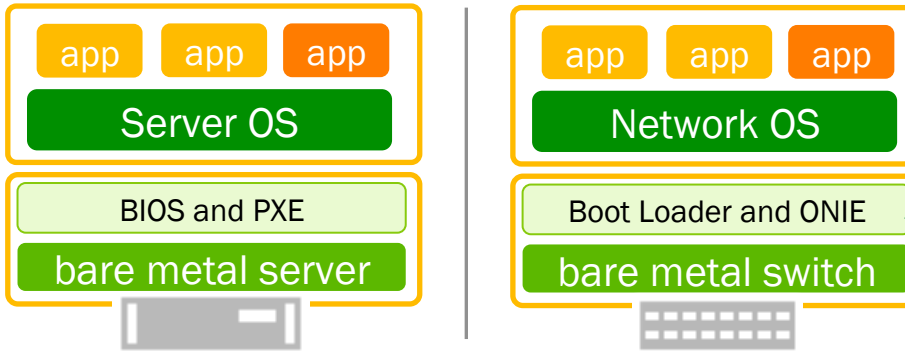
- Perl
- Python
- Ruby
- Shell

ZTP Example with Puppet

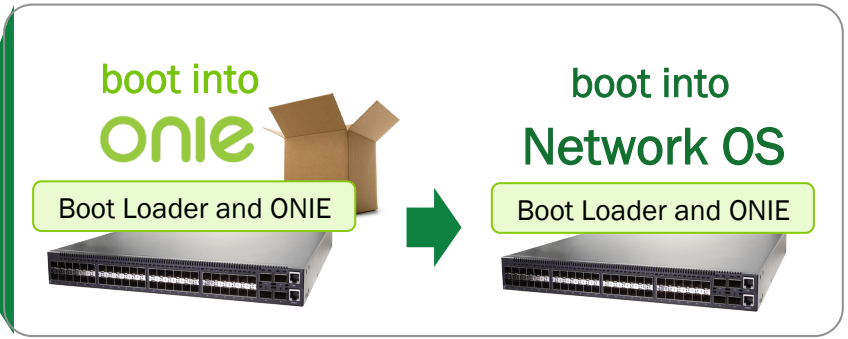
```
#!/bin/bash

function error() {
 echo -e "\e[0;33mERROR: Provisioning failed running $BASH_COMMAND at line $BASH_LINENO of $(basename $0) \e[0m" >&2
 exit 1
}
trap error ERR
# Allow Cumulus testing repo
sed -i /etc/apt/sources.list -e 's/^#\s*\(deb.*testing.*\)$/\1/g'

# Upgrade and install Puppet
apt-get update -y
apt-get upgrade -y
apt-get install puppet -y

echo "Configuring puppet" | wall -n
sed -i /etc/default/puppet -e 's/START=no/START=yes/'

service puppet restart

# CUMULUS-AUTOPROVISIONING

exit 0
```

Comparison

Similar to installing a server OS using PXE



ONIE looks for and installs network OS image



Video

cumulus

Because Debian based, we can do everything via puppet

- users
- interface configuration
- routing software (Quagga) configuration

Video!

cumulus®

How can we make your life better?

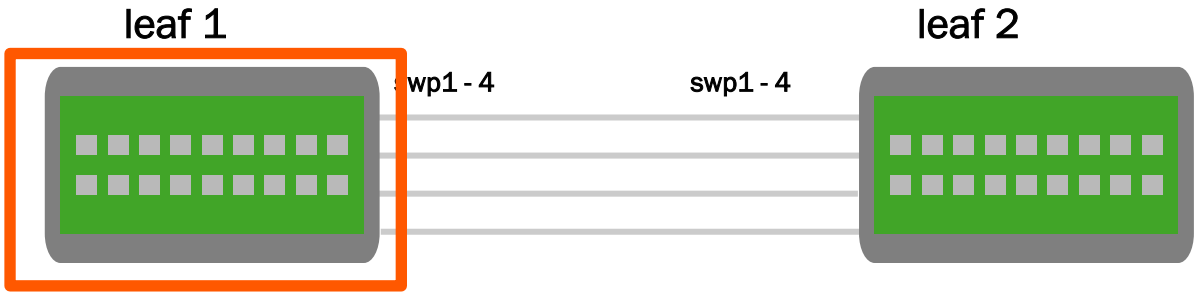
Interim Puppet Server!

Warning - small hard drive and limited processing power -- not recommended for long term

Scenario - installing new rack with limited connectivity

* make switch puppetmaster

Example Topology



Puppetmaster

Video

Transition

Existing Network Interface Managers

- Optimized for desktop and hypervisor environments
- Complexity increases with interface configuration scale
- Burden of network interface configuration dependency ordering is on the user
- Lack of support for incremental changes to network interfaces: minimal disruption
- Lack of tools to query and validate running interface configuration

cumulus

Switch networking characteristics:

- Large number of interfaces
- Switch ports, bridges, bonds, vlans
- Large number of interface attributes
- Addresses, bridge stp, mstp and igmp attributes
- Mostly static configuration

cumulus

Benefits

- Pluggable architecture
- Uses native Linux tools, enabling faster development
- Good user documentation, well known tool

Challenges

- No knowledge of interface configuration dependency (burden on the user)
- Large scale configuration results in large files or too many files
- No support for incremental configurations
- No support to query/validate running interface configuration
- literate program
- Bugs

cumulus

New implementation of ifupdown in Python

- Backward compatible with ifupdown interfaces format and commands
- Continues to use existing Linux native tools to configure network interfaces.
- Large number of interface attributes
- Pluggable architecture add-on python modules for interface configuration
- Meets some shortcomings seen with existing network interface managers on network switches
- ifreload acts like HUP instead of restart

ifupdown2 compare cli?

ifupdown

```
auto swp19
iface swp19 inet manual
  up link set $IFACE up
  down link set $IFACE down
  pre-up /sbin/ethtool -s $IFACE speed
  1000

auto swp19.100
iface swp19.100 inet manual
  up link set $IFACE up
  down link set $IFACE down

auto vlan100
iface vlan100 inet manual
bridge_ports swp19.100
mstpctl_stp on
```

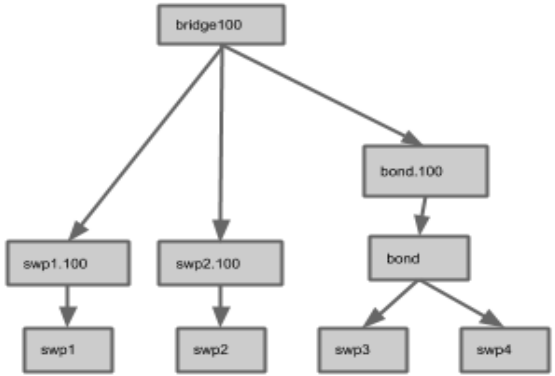
ifupdown2

```
auto swp19
iface swp19
  link-speed 1000

auto vlan100
iface vlan100
  bridge-ports swp19.100
  bridge-stp on
```

Network Interface Dependency

- Handles network interface dependency using dependency graphs
- Uses topological sort to order network interface configurations
- Provides options and tools to query and execute interface configuration in dependency order
- Built-in devices support

  ifupdown2 implicitly recognizes vlan and physical interfaces that appear as dependents and does the required minimal configuration to get them up: No need to specify `iface swp1.1000` in the example



sample network interface dependency graph showing switchports (swp1, swp2, ..), a bridge (bridge100), bond (bond) and vlan interfaces (swp1.100, swp3.100, bond.100)

sorted interface list: swp1, swp1.100, swp2, swp2.100, swp3, swp4, bond, bond.100, bridge100

ifupdown2 - Mako Templates

```
## Note that the "range" ends with '4'
## But will iterate only from 1 to 3
## See Python range() for more details
% for i in range(1, 4):
   auto swp${i}
   iface swp${i}
   % endfor
```

```
<%def name="interface_defaults()">
      mtu 9000
      link-speed 10000
      link-duplex full
      link-autoneg off
  </%def>

  % for i in range(3,7):
  auto swp${i}
  iface swp${i}
  ${interface_defaults()}
  % endfor

  auto default_bridge
  iface default_bridge
          bridge_ports glob swp3-6
          bridge-stp on
```

Find out more!

Ifupdown2

- https://github.com/CumulusNetworks/ifupdown2

PTM

- https://github.com/CumulusNetworks/ptm

Example Code

- https://github.com/LeslieCarr/puppet-presentation

Cumulus Open Source

- http://oss.cumulusnetworks.com

Twitter

- @lesliegeek

Thank You!

© 2014 Cumulus Networks. Cumulus Networks, the Cumulus Networks Logo, and Cumulus Linux are trademarks or registered trademarks of Cumulus Networks, Inc. or its affiliates in the U.S. and other countries. Other names may be trademarks of their respective owners. The registered trademark Linux® is used pursuant to a sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the mark on a world-wide basis.