POWER-Up User Guide Documentation

Release 2.0

Ray Harrington

Oct 16, 2020

Contents:

Version 2.0

Date 2018-03-26

Document Owner OpenPOWER POWER-Up Team

Authors Irving Baysah, Rolf Brudeseth, Jay Carman, Ray Harrington, Hoa Ngo, Nilesh Shah

CHAPTER 1

Document Preface and Scope

This document is a User's guide for the OpenPOWER POWER-Up toolkit. It is targeted at all users of the toolkit. Users are expected to have a working knowledge of Linux and Ethernet networking.

1.1 Document Control

Upon initial publication, this document will be stored on Github

1.2 Revision History

0.9	11 Oct 2016	Beta release	
1.0	24 Jan 2017	initial external release	
1.0	4 Feb 2017	Fixes and updates	
1.1	24 Feb 2017	Release 1.1 with LAG and MLAG support	
1.2	14 Apr 2017	Release 1.2 with introspection and support for 4 ports and 2 bonds	
1.3	26 Jun 2017	Release 1.3 Passive switch mode and improved introspection support.	
1.4	22 Sep 2017	Release 1.4 Cisco passive mode support.	
2.0b	7 Mar 2018	Release 2.0 New config file format and validation. Add hardware discovery and validation. Add Cisco (NX-OS) switch support	

Table 1: Revision History

1.3 Related Documentation

Document Name	Location / Owner
Lenovo Application Guide For Networking OS 8.3	http://systemx.lenovofiles.com/help/topic/com.lenovo.rackswitch.g8052.doc/G8052_AG_8-3.pdf
Mellanox MLNX-OS® User Manual for Ethernet	See instructions for access at https://community.mellanox.com/docs/DOC-2188

CHAPTER 2

Release Table

Release	Code Name	Release Date	End of Life Date
0.9	Antares	2016-10-24	2017-04-15
1.0	Betelgeuse	2017-01-25	2018-03-07
1.1	Castor	2017-02-24	2018-03-07
1.2	Denebola	2017-04-15	2018-03-07
1.3	Electra	2017-06-26	TBD
1.4	Fafnir	2017-06-26	TBD
2.0	Grumium	2018-03-26	TBD
2.1	Helvetios	TBD	TBD

CHAPTER 3

Introduction

Cluster POWER-Up enables greatly simplified configuration of clusters of bare metal OpenPOWER servers running Linux and/or installation of software to clusters of servers. It leverages widely used open source tools such as Cobbler, Ansible and Python. Because it relies solely on industry standard protocols such as IPMI and PXE boot, hybrid clusters of OpenPOWER and x86 nodes can readily be supported. Currently Cluster POWER-Up supports Ethernet networking. POWER-Up can configure simple flat networks for typical HPC environments or more advanced networks with VLANS and bridges for OpenStack environments. Complex heterogeneous clusters can be easily deployed using POWER-Up's interface and node templates. POWER-Up configures the switches in the cluster with support for multiple switch vendors. Software installation allows for 'pluggable' software install modules which can be user created. Utilities are provided to facilitate the creation of repositories and installation of the nginx web server.

3.1 Overview

Cluster POWER-Up is designed to be easy to use. If you are implementing one of the supported architectures with supported hardware, it eliminates the need for custom scripts or programming. It does this via a text configuration file (config.yml) which drives the cluster configuration. The configuration file is a YAML text file which the user edits. Several example config files are included docs directory. The configuration process is driven from a "deployer" node which can be removed from the cluster when finished. The POWER-Up process is as follows;

1. Rack and cable the hardware.
2. Initialize hardware.
    - initialize switches with static IP address, userid and password.
    - insure that all cluster compute nodes are set to obtain a DHCP address on their BMC ports and they are configured to support PXE boot on one of their network adapters.
3. Install the Cluster POWER-Up software on the deployer node.
4. Edit an existing config.yml file to drive the configuration.
5. Run the POWER-Up software

When finished, Cluster POWER-Up generates a YAML formatted inventory file with detailed information about your cluster nodes. This file can be read by operational management software and used to seed configuration files needed for installing a solution software stack.

3.1.1 Hardware and Architecture Overview

The POWER-Up software supports clusters of servers interconnected with Ethernet. The servers must support IPMI and PXE boot. Multiple racks can be configured with traditional two tier access-aggregation networking. POWER-Up configures both a management and data network. In simple / cost sensitive setups, the management and data networks can be configured on the same physical switch. Power-Up can configure VLANs and bonded networks with as many ports as the hardware supports. Redundant data switches (ie MLAG) are also supported. (Currently only implemented on Mellanox switches.)

3.1.2 Networking

Cluster POWER-Up supports Cisco, Mellanox and Lenovo switches. Not all functionality is enabled on all switch types. Currently redundant networking (MLAG) is only implemented on Mellanox switches. Port channel support is only implemented on Cisco (NX-OS) and Mellanox switches. POWER-Up can configure any number of node interfaces on cluster nodes. To facilitate installation of higher level software, interfaces can be optionally renamed.

Interface templates are used to define network configurations in the config.yml file. These can be physical ports, bonded ports, Linux bridges or VLANS. Interface templates can be entered using Ubuntu or Red Hat network configuration syntax. Once defined, interface templates can be applied to any node template. Node interfaces can optionally be configured with static IP addresses. These can be assigned sequentially or from a list.

3.1.3 Compute Nodes

Cluster POWER-Up supports clusters of heterogeneous compute nodes. Users can define any number of node types by creating templates in a config file. Node templates can include any network templates defined in the network templates section. The combination of node templates and network templates allows great flexibility in building heterogeneous clusters with nodes dedicated to specific purposes.

3.1.4 Supported Hardware

Compute Nodes

OpenPOWER Compute Nodes;

- S812LC
- S821LC
- S822LC (Minsky)
- SuperMicro OpenPOWER servers

x86 Compute Nodes;

- Lenovo x3550
- Lenovo x3650

Many other x86 nodes should work, but we have only tested with Lenovo and some Supermicro nodes.

Switches

For information on adding additional switch support using POWER-Up's switch class API, (see *Developer Guide*)

Supported Switches;

- Mellanox SX1410
- Mellanox SX1710
- Cisco 5K (FEXes supported)
- Lenovo G8052, G7028, G7052 (bonding not currently supported)

Note Other Mellanox switches may work but have not been tested Lenovo G8264 has not been tested Other Cisco NX-OS based switches may work but have not been tested

CHAPTER 4

Prerequisite Hardware Setup

4.1 Setting up the Deployer Node

It is recommended that the deployer node have at least one available core of a XEON class processor, 16 GB of memory free and 64 GB available disk space. When using the POWER-Up software installation capabilities, it is recommended that 100 GB of disk space be available and that there be at least 40 GB of free disk space in the partition holding the /srv directory. For larger clusters, additional cores, memory and disk space are recommended. A 4 core XEON class processor with 32 GB memory and 320 GB disk space is generally adequate for clusters up to several racks.

The deployer node requires internet access for setup and installation of the POWER-UP software and may need internet access for creation of any repositories needed for software installation. This can be achieved through the interface used for connection to the management switch (assuming the management switch has a connection to the internet) or through another interface. Internet access requirements for software installation depends on the software installation module. Internet access is required when running cluster deployments.

4.1.1 Operating Sytem and Package setup of the Deployer Node

- **Deployer OS Requirements:**
  - **Ubuntu (Software installation is not yet supported under Ubuntu)**
    - Release 14.04LTS or 16.04LTS
    - sudo privileges
  - **RHEL (Software installation is supported with POWER-Up vs 2.1. Cluster deployment is not yet supported und**
    - Release 7.2 or later
    - Extra Packages for Enterprise Linux (EPEL) repository enabled (https://fedoraproject.org/wiki/EPEL)
    - sudo privileges

        * **Enable Red Hat 'optional' and 'extra' repository channels or enable the repository on the RHEL installat**

            - **Power8:**

              $ sudo subscription-manager repos –enable=rhel-7-for-power-le-optional-rpms

              $ sudo subscription-manager repos –enable=rhel-7-for-power-le-extras-rpms

            - **Power9:**

              $ sudo subscription-manager repos –enable=rhel-7-for-power-9-optional-rpms

              $ sudo subscription-manager repos –enable=rhel-7-for-power-9-extras-rpms

- **Optional:**
  - Assign a static, public ip address to the BMC port to allow external control of the deployer node.
  - Enable ssh login

4.1.2 Network Configuration of the Deployer Node

For Software Installation

Use of the POWER-Up software installer requires that an interface on the installer node be pre-configured with access to the cluster nodes. If the cluster was not deployed by POWER-Up, this needs to be done manually. If the cluster has been deployed by POWER-Up, the PXE network will be automatically configured and can be used for software installation.

Although a routed connection to the cluster can be used for software installs, It is preferable that the interface used have an IP address in the subnet of the cluster network to be used for installation.

For Bare Metal Deployments

For bare metal deployments the deployer port connected to the management switch must be defined in /etc/network/interfaces (Ubuntu) or the ifcfg-eth# file (RedHat). e.g.:

```
auto eth0      # example device name
iface eth0 inet manual
```

POWER-Up can set up a subnet and optionally a vlan for it's access to the switches in the cluster. It is recommended that the deployer be provided with a direct connection to the management switch to simplify the overall setup. If this is not possible, the end user must insure that tagged vlan packets can be communicated between the deployer and the switches in the cluster. The interface used for PXE and IPMI can have additional IP addresses on it, but they should not be in the PXE or IPMI subnet. Similarly, this interface can have existing tagged vlans configured on it, but they should not be the vlans to be used by the PXE and IPMI networks.

An example of the config file parameters used to configure initial access to the switches is given above with *POWER-Up setup of the switch management network*. For a detailed description of these keys see *deployer 'mgmt' networks*, *'switches: mgmt:'* and *'switches: data:'* in the *Cluster Configuration File Specification*.

4.2 Hardware initialization

- Insure the cluster is cabled according to build instructions and that a list of all switch port to node physical interface connections is available and verified. Note that every node must have a physical connection from both BMC and PXE ports to a management switch. (see the example cluster in *Appendix-D*)

- Cable the deployer node directly to a management switch. For large cluster deployments, a 10 Gb connection is recommended. The deployer node must have access to the public internet (or site) network for retrieving software and operating system image files. If the cluster management network does not have external access an alternate connection must be provided, such as the cluster data network.
- Insure that the BMC ports of all cluster nodes are configured to obtain an IP address via DHCP.
- If this is a first time OS install, insure that all PXE ports are configured to obtain an IP address via DHCP. On OpenPOWER servers this is typically done using the Petitboot menus, e.g.:

```
Petitboot System Configuration
──────────────────────────────────────────────────────────────────────────────

 Boot Order      (0) Any Network device
                 (1) Any Device:

                 [    Add Device:     ]
                 [  Clear & Boot Any  ]
                 [       Clear        ]

 Timeout:        10    seconds


 Network:        (*) DHCP on all active interfaces
                 ( ) DHCP on a specific interface
                 ( ) Static IP configuration
```

- Acquire any needed public and or site network addresses.
- Insure you have a config.yml file to drive the cluster configuration. If necessary, edit / create the config.yml file (see section *Creating the Config File*)

Configuring the Cluster Switches

POWER-Up can configure supported switch models (See *Supported Hardware*). If automated switch configuration is not desired 'passive' switch mode can be used with any switch model (See *Preparing for Passive Mode*)

Initial configuration of cluster switch(es)

In order to configure your cluster switches, Cluster POWER-Up needs management access to all your cluster switches. This management network can be vlan isolated but for most applications a non-isolated management network is suitable and simpler to setup. To prepare for a non-isolated management network, you need to create management interfaces on all your cluster switches. The IP addresses for these management interfaces all need to be in the same subnet. The deployer will also need an IP address in this subnet. You will also need to know a userid and password for each switch and each switch will need to be enabled for SSH access. One of the management switches in your cluster must have a data port accessible to the deployer. This can be a routed connection supporting tagged vlans, but it is recommended that there be a direct connection between the deployer and one management switch.

For out of box installation, it is usually easiest to configure switches using a serial connection. Alternately, if the switch has a connection to a network without a DHCP server running, you may be able to access the switch at a default IP address. If the switch has a connection to a network with a DHCP server running, you may be able to reach it at the assigned IP address. See the switches installation guide. For additional info on Lenovo G8052 specific commands, see *Appendix-G* and the *Lenovo RackSwitch G8052 Installation guide*).

In this simple cluster example, the management switch has an in-band management interface. The initial setup requires a management interface on all switches configured to be accessible by the deployer node. The configured ip address must be provided in the 'interfaces:' list within each *'switches: mgmt:'* and *'switches: data:'* item. Cluster POWER-Up uses this address along with the provided userid and password credentials to access the management switch. Any additional switch 'interfaces' will be configured automatically along with *deployer 'mgmt' networks*.

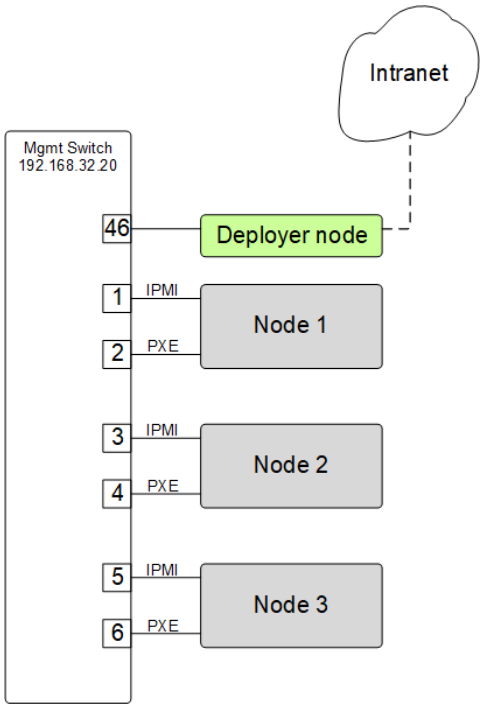

Fig. 1: POWER-Up setup of the switch management network

The following snippets are example config.yml entries for the diagram above:

- Switch config file definition:

```
switches:
    mgmt:
        - label: mgmt_switch
          userid: admin
          password: abc123
          class: lenovo
          interfaces:
              - type: inband
                ipaddr: 192.168.32.20
          links:
              - target: deployer
                ports: 46
```

- Deployer 'mgmt' networks:

```
deployer:
    networks:
        mgmt:
            - device: enp1s0f0
              interface_ipaddr: 192.168.32.95
              netmask: 255.255.255.0
```

Note that the deployer mgmt interface_ipaddress is in the same subnet as the management switches ipaddr. (192.168.32.0 netmask: 255.255.255.0)

As an example, management switch setup commands for the Lenovo G8052 are given below. For other supported switches consult the switch documentation.

- Enable configuration of the management switch:

```
enable
configure terminal
```

- Enable IP interface mode for the management interface:

```
RS G8052(config)# interface ip 1
```

- assign a static ip address, netmask and gateway address to the management interface. This must match one of the switch 'interfaces' items specified in the config.yml *'switches: mgmt:'* list:

```
RS G8052(config-ip-if)# ip address 192.168.32.20  # example IP address
RS G8052(config-ip-if)# ip netmask 255.255.255.0
RS G8052(config-ip-if)# vlan 1  # default vlan 1 if not specified
RS G8052(config-ip-if)# enable
RS G8052(config-ip-if)# exit
```

- admin password. This must match the password specified in the config.yml corresponding *'switches: mgmt:'* list item. The following command is interactive:

```
access user administrator-password
```

- disable spanning tree:

```
spanning-tree mode disable
```

- enable secure https and SSH login:

```
ssh enable
ssh generate-host-key
access https enable
```

- Save the config. For additional information, consult vendor documentation):

```
copy running-config startup-config
```

Adding additional management and data switch(es)

For out of box installation, it is usually necessary to configure the switch using a serial connection. See the switch installation guide. As an example, for Mellanox switches, a configuration wizard can be used for initial configuration:

- assign hostname
- set DHCP to no for management interfaces
- set zeroconf on mgmt0 interface: to no
- do not enable ipv6 on management interfaces
- assign static ip address. This must match the corresponding interface 'ipaddr' specified in the config.yml file *'switches: data:'* list, and be in a *deployer 'mgmt' network*.
- assign netmask. This must match the netmask of the *deployer 'mgmt' network* that will be used to access the management port of the switch.
- default gateway
- Primary DNS server
- Domain name

- Set Enable ipv6 to no
- admin password. This must match the password specified in the config.yml corresponding *'switches: data:'* list item.
- disable spanning tree. Typical industry standard commands:

```
enable
configure terminal
no spanning-tree
```

- enable SSH login:

```
ssh server enable
```

- Save config. In switch config mode:

```
configuration write
```

- If using redundant data switches with MLAG or vPC, connect only a single inter switch peer link (IPL) between switches or leave the IPL links disconnected until Cluster POWER-Up completes. (This avoids loops)
- Add the additional switches to the config.yml. A data switch is added as shown below:
  - Switch config file definition:

```
switches:
    .
    .
    data:
        - label: data_switch
          userid: admin
          password: abc123
          class: cisco
          interfaces:
              - type: inband
                ipaddr: 192.168.32.25
          links:
              - target: mgmt_switch
                ports: mgmt
```

This completes normal POWER-Up initial configuration. For additional information and examples on preparing cluster hardware, see the sample configurations in the appendices.

Preparing for Passive Mode

In passive mode, POWER-Up configures the cluster compute nodes without requiring any management communication with the cluster switches. This facilitates the use of POWER-Up even when the switch hardware is not supported or in cases where the end user does not allow 3rd party access to their switches. When running POWER-Up in passive mode, the user is responsible for configuring the cluster switches. The user must also provide the Cluster POWER-Up software with MAC address tables collected from the cluster switches during the POWER-Up process. For passive mode, the cluster management switch must be fully programmed before beginning cluster POWER-Up, while the data switch should be configured after POWER-Up runs.

Configuring the management switch(es)

- The port(s) connected to the deployer node must be put in trunk mode with allowed vlans associated with each respective device as defined in the deployer *'mgmt'* and *'client'* networks.

- The ports on the management switch which connect to cluster node BMC ports or PXE interfaces must be in access mode and have their PVID (Native VLAN) set to the respective 'type: ipmi' and 'type: pxe' 'vlan' values set in the *'deployer client networks'*.

Configuring the data switch(es)

Configuration of the data switches is dependent on the user requirements. The user / installer is responsible for all configuration. Generally, configuration of the data switches should occur after Cluster POWER-Up completes. In particular, note that it is not usually possible to acquire complete MAC address information once vPC (AKA MLAG or VLAG) has been configured on the data switches.

CHAPTER 5

Installing the POWER-Up Software

1. Verify that all the steps in *Setting up the Deployer Node* have been executed.
2. Login to the deployer node.
3. Install git
   - Ubuntu:

     ```
     $ sudo apt-get install git
     ```

   - RHEL:

     ```
     $ sudo yum install git
     ```

4. From your home directory, clone POWER-Up:

   ```
   $ git clone https://github.com/open-power-ref-design-toolkit/power-up
   ```

5. Install the remaining software packages used by Power-Up and setup the environment:

   ```
   $ cd power-up
   $ ./scripts/install.sh

   (this will take a few minutes to complete)

   $ source scripts/setup-env
   ```

   NOTE: The setup-env script will ask for permission to add lines to your .bashrc file which modify the PATH environment variable. It is recommended that you allow this so that the POWER-Up environment is restored if you need to re-open the window or open and additional window.

CHAPTER 6

Creating the Config File

The config file drives the creation of the cluster. It is in YAML format which is stored as readable text. The lines must be terminated with a newline character (\n). When creating or editing the file on the Microsoft Windows platform be sure to use an editor, such as *LibreOffice*, which supports saving text files with the newline terminating character or use *dos2unix* to convert the windows text file to unix format.

Sample config files can be found in the *power-up/sample-configs* directory. Once a config file has been created, rename it to *config.yml* and move it to the project root directory. YAML files support data structures such as lists, dictionaries and scalars. The *Cluster Configuration File Specification* describes the various fields.

See *Cluster Configuration File Specification*.

YAML files use spaces as part of its syntax. For example, elements of the same list must have the exact same number of spaces preceding them. When editing the config file pay careful attention to spaces at the start of lines. Incorrect spacing can result in failure to parse the file.

Schema and logic validation of the config file can be performed with the *pup.py* command:

```
$ cd power-up
$ source pup-venv/bin/activate
$ ./scripts/python/pup.py validate --config-file
```

6.1 Switch Mode

6.1.1 Active Switch Mode

This mode allows the switches to be automatically configured during deployment.

6.1.2 Passive Switch Mode

This mode requires the user to manually configure the switches and to write switch MAC address tables to file.

Passive management switch mode and passive data switch mode can be configured independently, but passive and active switches of the same classification cannot be mixed (i.e. all data switches must either be active or passive).

See *Config Specification - Globals Section*.

Passive Management Switch Mode:

Passive management switch mode requires the user to configure the management switch *before* initiating a deploy. The client network must be isolated from any outside servers. IPMI commands will be issued to any system BMC that is set to DHCP and has access to the client network.

Passive Data Switch Mode:

Passive data switch mode requires the user to configure the data switch in accordance with the defined networks. The node interfaces of the cluster will still be configured.

6.2 Networks

The network template section defines the networks or groups of networks and will be referenced by the *Node Template* members.

See *Config Specification - Networks Section*.

6.3 Node Templates

The order of the individual ports under the *ports* list is important since the index represents a node and is referenced in the list elements under the *pxe* and *data* keys.

See *Config Specification - Node Templates Section*.

6.3.1 Renaming Interfaces

The *rename* key provides the ability to rename ethernet interfaces. This allows the use of heterogeneous nodes with software stacks that need consistent interface names across all nodes. It is not necessary to know the existing interface name. The cluster configuration code will find the MAC address of the interface cabled to the specified switch port and change it accordingly.

6.3.2 Install Device

The *install_device* key is the disk to which the operating system will be installed. Specifying this disk is not always obvious because Linux naming is inconsistent between boot and final OS install. For OpenPOWER S812LC, the two drives in the rear of the unit are typically used for OS install. These drives should normally be specified as */dev/sdj* and */dev/sdk*.

6.4 Post POWER-Up Activities

Once deployment has completed it is possible to launch additional commands or scripts specified in the *Software Bootstrap* section. These can perform configuration actions or bootstrap install of additional software packages. Commands can be specified to run on all cluster nodes or only specific nodes determined by the compute template name.

See *Config Specification - Software Bootstrap Section*.

CHAPTER 7

Running the POWER-Up Cluster Deployment Software

7.1 Installing and Running the POWER-Up code. Step by Step Instructions

1. Verify that all the steps in section *4 Prerequisite Hardware Setup* have been executed. POWER-Up can not run if addresses have not been configured on the cluster switches and recorded in the config.yml file.
2. Login to the deployer node.
3. Install git
   - Ubuntu:

     ```
     $ sudo apt-get install git
     ```

   - RHEL:

     ```
     $ sudo yum install git
     ```

4. From your home directory, clone POWER-Up:

   ```
   $ git clone https://github.com/open-power-ref-design-toolkit/power-up
   ```

5. Install the remaining software packages used by Power-Up and setup the environment:

   ```
   $ cd power-up
   $ ./scripts/install.sh

   (this will take a few minutes to complete)

   $ source scripts/setup-env
   ```

   NOTE: The setup-env script will ask for permission to add lines to your .bashrc file. It is recommended that you allow this so that the POWER-Up environment is restored if you open a new window. These lines can be removed using the "teardown" script.

6. If introspection is enabled then follow the instructions in Building Necessary Config Files to set the 'IS_BUILDROOT_CONFIG' and 'IS_KERNEL_CONFIG' environment variables. (Introspection NOT YET ENABLED for POWER-Up 2.0)

7. Copy your config.yml file to the ~/power-up directory (see section *4 Creating the config.yml File* for how to create the config.yml file)

8. Copy any needed os image files (iso format) to the '~/power-up/os-images' directory. Symbolic links to image files are also allowed.

   NOTE: Before beginning the next step, be sure all BMCs are configured to obtain a DHCP address then reset (reboot) all BMC interfaces of your cluster nodes. As the BMCs reset, the POWER-Up DHCP server will assign new addresses to them.

   One of the following options can be used to reset the BMC interfaces;

   - Cycle power to the cluster nodes. BMC ports should boot and wait to obtain an IP address from the deployer node.
   - Use ipmitool run as root local to each node; ipmitool bmc reset warm OR ipmitool mc reset warm depending on server
   - Use ipmitool remotely such as from the deployer node. (this assumes a known ip address already exists on the BMC interface):

     ```
     ipmitool -I lanplus -U <username> -P <password> -H <bmc ip address> mc reset␣
     ↪cold
     ```

   If necessary, use one of the following options to configure the BMC port to use DHCP;

   - From a local console, reboot the system from the host OS, use the UEFI/BIOS setup menu to configure the BMC network configuration to DHCP, save and exit.
   - use IPMItool to configure BMC network for DHCP and reboot the BMC

9. Copy your config.yml file to the ~/power-up directory.

10. To validate your config file:

    ```
    $ pup validate --config-file
    ```

    Note: **Most of POWER-Up's capabilities are accessed using the 'pup' program.** For a complete overview of the pup program, see *Appendix-A*.

11. To deploy operating systems to your cluster nodes:

    ```
    $ pup deploy
    ```

    Note: If running with passive management switch(es) follow special instructions in *Passive Switch Mode Special Instructions* instead. (NOTE: passive management switches are not yet supported in POWER-Up 2.0)

12. This will create the management networks, install the container that runs most of the POWER-Up functions and then optionally launch the introspection OS and then install OS's on the cluster nodes. This process can take as little as 40 minutes or as much as multiple hours depending on the size of the cluster, the capabilities of the deployer and the complexity of the deployment.

    - To monitor progress of the deployment, open an additional terminal session into the deployment node and run the pup program with a status request. (Running POWER-Up utility functions in another terminal window will not work if you did not allow POWER-Up to make updates to your .bashrc file):

      ```
      $ pup util --status (NOT yet implemented in POWER-Up 2.0)
      ```

After a few minutes POWER-Up will have initialized and will start discovering and validating your cluster hardware. During discovery and validation, POWER-Up will first verify that it can communicate with all of the switches defined in the config file. Next it will create a DHCP server attached to the IPMI network and wait for all of the cluster nodes defined in the config file to request a DHCP address. After several minutes, a list of responding nodes will be displayed. (display order will match the config file order). If there are missing nodes, POWER-Up will pause so that you can take corrective actions. You will then be given the option to continue discovering the nodes or to continue on. POWER-Up will also verify that all nodes respond to IPMI commands. Next, POWER-Up will verify that all cluster nodes are configured to request PXE boot. POWER-Up will set the boot device to PXE on all discovered nodes, cycle power and then wait for them to request PXE boot. Note that POWER-Up will not initiate PXE boot at this time, it is only verifying that all the nodes are configured to request PXE boot. After several minutes all nodes requesting PXE boot will be listed (again in the same order that they are entered in the config file) POWER-Up will again pause to give you an opportunity to make any necessary corrections or fixes. You can also choose to have POWER-Up re-cycle power to nodes that have not yet requested PXE boot. For nodes that are missing, verify cabling and verify the config.yml file. See "Recovering from POWER-Up Issues" in the appendices for additional debug help. You can check which nodes have obtained IP addresses, on their BMC's and or PXE ports by executing the following from another window:

```
$ pup util --scan-ipmi (not yet implemented in POWER-Up 2.0)
$ pup util --scan-pxe  (not yet implemented in POWER-Up 2.0)
```

NOTES: The DHCP addresses issued by POWER-Up during discovery and validation have a short 5 minute lease and POWER-Up dismantles the DHCP servers after validation. You will lose the ability to scan these networks within a few minutes after validation ends. After deploy completes, you will again be able to scan these networks.

Note that cluster validation can be re-run as often as needed. Note that if cluster validation is run after deploy, the cluster nodes will be power cycled which will of course interrupt any running work.

After discovery and validation complete, POWER-Up will create a container for the POWER-Up deployment software to run in. Next it installs the deployment software and operating system images in the container and then begins the process of installing operating systems to the cluster nodes. Operating system install happens in parallel and overall install time is relatively independent of the number of nodes up to tens of nodes.

13. Introspection (NOT yet enabled in POWER-Up 2.0)

    If introspection is enabled then all client systems will be booted into the in-memory OS with ssh enabled. One of the last tasks of this phase of POWER-Up will print a table of all introspection hosts, including their IP addresses and login / ssh private key credentials. This list is maintained in the 'power-up/playbooks/hosts' file under the 'introspections' group. POWER-Up will pause after the introspection OS deployment to allow for customized updates to the cluster nodes. Use ssh (future: or Ansible) to run custom scripts on the client nodes.

14. To continue the POWER-Up process after introspection, press enter.

    Again, you can monitor the progress of operating system installation from an additional terminal window:

    ```
    $ pup util --status
    ```

    It will usually take several minutes for all the nodes to load their OS. If any nodes do not appear in the cobbler status, see "Recovering from POWER-Up Issues" in the Appendices

    POWER-Up creates logs of it's activities. A file (gen) external to the POWER-Up container is written in the power-up/log directory.

    An additional log file is created within the deployer container. This log file can be viewed:

    ```
    $ pup util --log-container  (NOT yet implemented in POWER-Up 2.0)
    ```

Configuring networks on the cluster nodes

Note: If running with passive data switch(es) follow special instructions in *post-deploy-passive* instead.

After completion of OS installation, POWER-Up will pause and wait for user input before continuing. You can press enter to continue on with cluster node and data switch configuration or stop the POWER-Up process. After stopping, you can readily continue the node and switch configuration by entering:

```
$ pup post-deploy
```

During post-deploy, POWER-Up performs several additional activities such as setting up networking on the cluster nodes, setting up SSH keys and copying them to cluster nodes, and configures the data switches.

If data switches are configured with MLAG verify that;

- Only one IPL link is connected. (Connecting multiple IPL links before configuration can cause loop problems)
- No ports used by you cluster nodes are configured in port channels. (If ports are configured in port channels, MAC addresses can not be acquired, which will prevent network configuration)

7.2 Passive Switch Mode Special Instructions

Deploying operating systems to your cluster nodes with passive management switches

When prompted, it is advisable to clear the mac address table on the management switch(es).

When prompted, write each switch MAC address table to file in the 'power-up/passive' directory. The files should be named to match the unique switch label values set in the 'config.yml' 'switches:' dictionary. For example, for the following management switch definitions:

```
switches:
    mgmt:
        - label: passive_mgmt_1
          userid: admin
          password: abc123
          interfaces:
            :
            :
            :
    mgmt:
        - label: passive_mgmt_2
          userid: admin
          password: abc123
          interfaces:
```

The user would need to write two files:

1. 'power-up/passive/passive_mgmt_1'
2. 'power-up/passive/passive_mgmt_2'

If the user has ssh access to the switch management interface, writing the MAC address table to file can be readily accomplished by redirecting stdout. Here is an example of the syntax for a Lenovo G8052:

```
$ ssh <mgmt_switch_user>@<mgmt_switch_ip> \
'show mac-address-table' > ~/power-up/passive/passive_mgmt_1
```

Note that this command would need to be run for each individual mgmt switch, writing to a separate file for each. It is recommended to verify each file has a complete table for the appropriate interface configuration and only one mac address entry per interface.

See *MAC address table file formatting rules* below.

After writing MAC address tables to file press enter to continue with OS installation. *Resume normal instructions*.

If deploy-passive fails due to incomplete MAC address table(s) use the following command to reset all servers (power off / set bootdev pxe / power on) and attempt to collect MAC address table(s) again when prompted:

```
$ pup util --cycle-power-pxe (NOT yet implemented)
```

Configuring networks on the cluster nodes with passive data switches

When prompted, it is advisable to clear the mac address table on the data switch(es). This step can be skipped if the operating systems have just been installed on the cluster nodes and the mac address timeout on the switches is short enough to insure that no mac addresses remain for the data switch ports connected to cluster nodes. If in doubt, check the acquired mac address file (see below) to insure that each data port for your cluster has only a single mac address entry.:

```
$ pup post-deploy
```

When prompted, write each switch MAC address table to file in 'power-up/passive'. The files should be named to match the unique label values set in the 'config.yml' 'switches:' dictionary. For example, take the following data switch definitions:

```
switches:
      :
      :
    data:
        - label: passive1
          class: cisco
          userid: admin
          password: passw0rd
      :
      :
        - label: passive2
          class: cisco
          userid: admin
          password: passw0rd
      :
      :
        - label: passive3
          class: cisco
          userid: admin
          password: passw0rd
```

The user would need to write three files:

1. '~/power-up/passive/passive1'
2. '~/power-up/passive/passive2'
3. '~/power-up/passive/passive3'

If the user has ssh access to the switch management interface writing the MAC address table to file can easily be accomplished by redirecting stdout. Here is an example of the syntax for a Mellanox SX1400 / SX1710:

```
$ ssh <data_switch_user>@<data_switch_ip> \
'cli en "conf t" "show mac-address-table"' > ~/power-up/passive/passive1
```

For a Cisco NX-OS based switch:

```
$ ssh <data_switch_user>@<data_switch_ip> \
'conf t ; show mac address-table' > ~/power-up/passive/passive1
```

Note that this command would need to be run for each individual data switch, writing to a separate file for each. It is recommended to verify each file has a complete table for the appropriate interface configuration and only one mac address entry per interface.

See *MAC address table file formatting rules* below.

MAC Address Table Formatting Rules

Each file must be formatted according to the following rules:

- **MAC addresses and ports are listed in a tabular format.**
  - Columns can be in any order
  - Additional columns (e.g. vlan) are OK as long as a header is provided.
- If a header is provided and it includes the strings "mac address" and "port" (case insensitive) it will be used to identify column positions. Column headers must be delimited by at least two spaces. Single spaces will be considered a continuation of a single column header (e.g. "mac address" is one column, but "mac address vlan" would be two).
- If a header is provided, it must include a separator row consisting of dashes '-' to delineate columns. One or more spaces or plus symbols '+' are to be used to separate columns.
- If a header is not provided then only MAC address and Port columns are allowed.
- **MAC addresses are written as (case-insensitive):**
  - Six pairs of hex digits delimited by colons (:) [e.g. 01:23:45:67:89:ab]
  - Six pairs of hex digits delimited by hyphens (-) [e.g. 01-23-45-67-89-ab]
  - Three quads of hex digits delimited by periods (.) [e.g. 0123.4567.89ab]
- **Ports are written either as:**
  - An integer
  - A string starting with 'Eth1/' followed by one or more numeric digits without white space. (e.g. "Eth1/25" will be saved as "25")
  - A string starting with 'Eth' and containing multiple numbers separated by "/". The 'Eth' portion of the string will be removed) removed. (e.g. "Eth100/1/5" will be saved as "100/1/5").

Cisco, Lenovo and Mellanox switches currently supported by POWER-Up follow these rules. An example of a user generated "generic" file would be:

```
mac address        Port
-----------------  ----
0c:c4:7a:20:0d:22    38
0c:c4:7a:76:b0:9b    19
0c:c4:7a:76:b1:16    9
0c:c4:7a:76:c8:ec    37
40:f2:e9:23:82:ba    18
40:f2:e9:23:82:be    17
40:f2:e9:24:96:5a    22
40:f2:e9:24:96:5e    21
5c:f3:fc:31:05:f0    13
5c:f3:fc:31:06:2a    12
5c:f3:fc:31:06:2c    11
```

(continues on next page)

(continued from previous page)

```
5c:f3:fc:31:06:ea    16
5c:f3:fc:31:06:ec    15
6c:ae:8b:69:22:24    2
70:e2:84:14:02:92    5
70:e2:84:14:0f:57    1
```

7.3 SSH Keys

The OpenPOWER POWER-Up Software will generate a passphrase-less SSH key pair which is distributed to each node in the cluster in the /root/.ssh directory. The public key is written to the authorized_keys file in the /root/.ssh directory and also to the /home/userid-default/.ssh directory. This key pair can be used for gaining passwordless root login to the cluster nodes or passwordless access to the userid-default. On the deployer node, the key pair is written to the ~/.ssh directory as gen and gen.pub. To login to one of the cluster nodes as root from the deployer node:

```
ssh -i ~/.ssh/gen root@a.b.c.d
```

As root, you can log into any node in the cluster from any other node in the cluster as:

```
ssh root@a.b.c.d
```

Where a.b.c.d is the IP address of the port used for pxe install. These addresses are stored under the key name *ipv4-pxe* in the inventory file. The inventory file is stored on every node in the cluster at /var/oprc/inventory.yml. The inventory file is also stored on the deployer in the deployer container in the /opt/power-up directory. A symbolic link to this inventory file is created in the ~/power-up directory as 'inventorynn.yml', where nn is the number of the pxe vlan.

Note that you can also log into any node in the cluster using the credentials specified in the config.yml file (key names *userid-default* and *password-default*)

CHAPTER 8

Running the POWER-Up Software Installation Software

Under development. This functionality is not yet supported in the master branch of POWER-Up. Development of this function is in the dev-software-install branch.

- Verify that all the steps in *Installing the POWER-Up Software* have been executed.
- Copy or download the software install module to be used to the power-up/software directory. POWER-Up currently ships with the installer module for PowerAI Enterprise vs 5.2. (paie52). See *Running the PowerAI Enterprise Software Install Module*
- Consult the README for the specific software install module for the names of any tar files, binaries or other files needed for the specific software installation. Copy these to the installer node before running the software install module. Installation files can be copied anywhere on the installer node but will be located more quickly if located in directories under a /home directory.

Run the prep phase:

```
$ pup software --prep <install module name>
```

After successful completion, run the init or install phase. (Install will run the init phase prior to installation phase):

```
$ pup software --init-clients <install module name>

$ pup software --install <install module name>
```

POWER-Up provides a simple framework for running user provided software install modules. See *Creating Software Install Modules* for guidance on how to create these modules.

CHAPTER 9

Creating Software Install Modules

POWER-Up provides a simple framework for running user provided software install modules. Software install modules are Python modules which reside in the power-up/software directory. The module may be given any valid Python module name. A POWER-Up software install module can contain any user provided code, but it must implement a class named 'software' and the software class must implement the following methods;

- README
- prep
- init_client
- install
- status

The prep method is generally intended to provide setup of repositories and directories and installation and configuration of a web server. POWER-Up provides support for setting up an EPEL mirror and supports installation of the nginx web server.

In order to facilitate software installation to clusters without internet access, the prep method is intended to be able to run without requiring access to the cluster nodes. This allows preloading of required software onto a laptop or other node prior to being connected to the cluster.

The init_client method should provide for license accept activities and setting up client nodes to access the POWER-Up node for any implemented repositories.

The install method needs to implement the logic for installing the desired software packages and binaries on the cluster nodes. POWER-Up includes Ansible. The install method may make use of any Ansible modules or POWER-Up provided playbooks.

CHAPTER 10

Running the PowerAI Enterprise Software Install Module

10.1 Overview

The PowerAI Enterprise software installation can be automated using the POWER-Up software. POWER-Up can run on OpenPOWER or x_86 architecture.

The PowerAI Enterprise Software Install Module provides for rapid installation of the PowerAI Enterprise software or PowerAI Enterprise evaluation software to a cluster of POWER8 or POWER9 servers. The install module creates a web based software installation server on one of the cluster nodes or another node with access to the cluster. The software server is populated with repositories and files needed for installation of PowerAI Enterprise. Once the software server is setup, installation scripts orchestrate the software installation to one or more client nodes. Note that the software installer node requires access to several open source repositories during the 'preparation' phase. During the preparation phase, packages which PowerAI Enterprise is dependent on are staged on the POWER-Up installer node. After completion of the preparation phase, the installation requires no further access to the open source repositories and can thus enable installation to servers which do not have internet access.

The POWER-Up software installer does not currently support installation of PowerAI Enterprise onto the node running the POWER-Up software installer. If it is necessary to install PowerAI Enterprise onto the node running the POWER-Up software, this can be done manually or can be accomplished by running the POWER-Up software on an additional node in the cluster. Hint: A second POWER-Up server can be quickly prepared by replicating the repositories from the first POWER-Up server.

10.2 Support

Questions regarding the PowerAI Enterprise installation software, installation, or suggestions for improvement can be posted on IBM's developer community forum at https://developer.ibm.com/answers/index.html with the PowerAI tag.

Answered questions regarding PowerAI can be viewed at https://developer.ibm.com/answers/topics/powerai/

10.3 Set up of the POWER-Up Software Installer Node

POWER-Up Node Prerequisites;

1. The POWER-Up software installer currently runs under RHEL 7.4 or above.
2. The user account used to run the POWER-Up software needs sudo privileges.
3. Enable access to the Extra Packages for Enterprise Linux (EPEL) repository. (https://fedoraproject.org/wiki/EPEL#Quickstart)
4. Enable the common, optional and extras repositories.

   On POWER8:

   ```
   $ sudo subscription-manager repos --enable=rhel-7-for-power-le-rpms --
   ↪enable=rhel-7-for-power-le-optional-rpms --enable=rhel-7-for-power-le-
   ↪extras-rpms
   ```

   On POWER9:

   ```
   $ sudo subscription-manager repos --enable=rhel-7-for-power-9-rpms --
   ↪enable=rhel-7-for-power-9-optional-rpms --enable=-enable=rhel-7-for-
   ↪power-9-extras-rpms
   ```

5. Insure that there is at least 45 GB of available disk space in the partition holding the /srv directory:

   ```
   $ df -h /srv
   ```

6. Install the version of POWER-Up software appropriate for the version of PowerAI Enterprise you wish to install. The versions listed in the table below are the versions tested with the corresponding release of PowerAI Enterprise;

PowerAI Enterprise Release	POWER-Up software installer vs	Notes	EOL date
1.1.0	software-install-b2.5		14 Sep 2018
1.1.1	software-install-b2.9		
1.1.1	software-install-b2.10	Updates for x86 based installer node	
1.1.1	software-install-b2.11	Enable cuda replicate on x86 installer node	
1.1.2	software-install-b2.12	Support for installation of PAIE 1.1.2	

From your home directory install the POWER-Up software and initialize the environment. For additional information see *Installing the POWER-Up Software*:

```
$ sudo yum install git

$ git clone https://github.com/open-power-ref-design-toolkit/power-up -b software-
↪install-b2.n

$ cd power-up

$ ./scripts/install.sh

$ source scripts/setup-env
```

NOTES:

- The latest functional enhancements and defect fixes can be obtained by cloning the software installer without specifying the tag release. Generally, you should use a release level specified in the table above unless you are experiencing problems.:

```
git clone https://github.com/open-power-ref-design-toolkit/power-up -b software-
↪install-b2
```

- Multiple users can install and use the PAIE installer software, however there is only one software server created and there are no safeguards built in to protect against concurrent modifications of the software server content, data files or client nodes.
- Each user of the PAIE installer software must install the POWER-Up software following the steps above.

10.4 Installation of PowerAI Enterprise

Installation of the PowerAI Enterprise software involves the following steps;

1. Preparation of the client nodes
2. Preparation of the software server
3. Initialization of the cluster nodes
4. Installation of software on the cluster nodes

10.4.1 Preparation of the client nodes

Before beginning automated installation, you should have completed the 'Setup for automated installer steps' at https://www.ibm.com/support/knowledgecenter/SSFHA8_1.1.1/enterprise/powerai_auto_install_setup.html

Before proceeding with preparation of the POWER-Up server, you will need to gather the following information;

- Fully qualified domain name (FQDN) for each client node
- Userid and password or private ssh key for accessing the client nodes. Note that for running an automated installation, the same user id and password must exist on all client nodes and must be configured with sudo access.

10.4.2 Preparation of the POWER-Up Software Server

Before beginning installation of PowerAI Enterprise, the files listed below need to be copied onto the software server node. The files can be copied anywhere, but the POWER-Up software can locate them quicker if the files are under a subdirectory of one of the /home/ directories or the /root directory. Note that the PAIE installer will stop searching for installation files if required files are found under one of the directories mentioned above.

- PowerAI Enterprise binary file. (powerai-enterprise-*_ppc64le.bin)
- Cuda cudnn (cudnn-9.2-linux-ppc64le-v7.*.tgz)
- Cuda nccl v2 (nccl_*-1+cuda9.2_ppc64le.tgz)

In addition, the POWER-Up software server needs access to the following repositories during the preparation phase;

- Red Hat 'common', 'optional' and 'extras'
- Extra Packages for Enterprise Linux (EPEL)

- Cuda Toolkit
- Anaconda

These can be accessed using the public internet (URL's are provided) or via an alternate web site such as an intranet mirror repository, another POWER-Up server or from a mounted USB key. Because the software installer can run on x_86 architecture, a laptop can be used as an installer node, allowing preparation at a location with internet access and installation at a location without internet access.

Before beginning, extract the contents of the powerai-enterprise-*_ppc64le.bin file and accept the license by running the following on the installer node:

```
$ sudo bash ./powerai-enterprise-*_ppc64le.bin
```

NOTES:

- Extraction and license acceptance of PowerAI Enterprise must be performed on an OpenPOWER node. If you are running the POWER-Up installer software on an x_86 node, you must first extract the files on an OpenPOWER node and then copy all of the extracted contents to the POWER-Up installer node.
- If running the PowerAI Enterprise installer from an x_86 node, you must download the Red Hat dependent packages on a Power node and copy them to a directory on the x_86 installer node. A utility script is included to facilitate this process. To use the script, insure you have ssh access with sudo privileges to a Power node which has a subscription to the Red Hat 'common', 'optional' and 'extras' channels. (One of the cluster nodes or any other suitable Power node can be used for this purpose). To run the script from the power-up directory on the installer node:

  ```
  $ ./software/get-dependent-packages.sh userid hostname
  ```

The hostname can be a resolvable hostname or ip address. The get-dependent-packages script will download the required packages on the specified Power node and then move them to the ~/tempdl directory on the installer node. After running the script, run/rerun the –prep phase of installation. For dependent packages, choose option D (Create from files in a local Directory) and enter the full absolute path to the /tempdl directory.

Status of the Software Server

At any time, you can check the status of the POWER-Up software server by running:

```
$ pup software --status paie*
```

To use the automated installer with the evaluation version of PowerAI Enterprise, include the –eval switch in all pup commands. ie:

```
$ pup software --status --eval paie*
```

Note: The POWER-Up software installer runs python installation modules. Inclusion of the '.py' in the software module name is optional. ie For PowerAI Enterprise version 1.1.1, paie111 or paie111.py are both acceptable.

Hint: The POWER-Up command line interface supports tab autocompletion.

Preparation is run with the following POWER-Up command:

```
$ pup software --prep paie*
```

Preparation is interactive and may be rerun if needed. Respond to the prompts as appropriate for your environment. Note that the EPEL, Cuda, dependencies and Anaconda repositories can be replicated from the public web sites or from alternate sites accessible on your intranet environment or from local disk (ie from a mounted USB drive). Most other files come from the local file system except for the Anaconda package which can be downloaded from the public internet during the preparation step.

10.4.3 Initialization of the Client Nodes

During the initialization phase, you will need to enter a resolvable hostname for each client node. Optionally you may enter the path of a private ssh key file. If one is not available, an ssh key pair will be automatically generated. You will also be prompted for a password for the client nodes.

To initialize the client nodes and enable access to the POWER-Up software server:

```
$ pup software --init-clients paie*
```

Note: Initialization of client nodes can be rerun if needed.

10.4.4 Installation

To install the PowerAI Enterprise software and prerequisites:

```
$ pup software --install paie*
```

NOTES:

- **During the installation phase you will be required to provide values for certain environment variables needed by Spectrum**
  - If left blank, the CLUSTERADMIN variable will be automatically populated with the cluster node userid provided during the init-client phase of installation.
  - The DLI_SHARED_FS environment variable should be the full absolute path to the shared file system mount point. (eg DLI_SHARED_FS: /mnt/my-mount-point). The shared file system and the client node mount points need to be configured prior to installing PowerAI Enterprise.
  - If left blank, the DLI_CONDA_HOME environment variable will be automatically populated. If entered, it should be the full absolute path of the install location for Anaconda. (ie DLI_CONDA_HOME: /opt/anaconda2)
- Installation of PowerAI Enterprise can be rerun if needed.

After completion of the installation of the PowerAI Enterprise software, you must configure Spectrum Conductor Deep Learning Impact and apply any outstanding fixes. Go to https://www.ibm.com/support/knowledgecenter/SSFHA8, choose your version of PowerAI Enterprise and then use the search bar to search for 'Configure IBM Spectrum Conductor Deep Learning Impact'.

10.4.5 Additional Notes

You can browse the content of the POWER-Up software server by pointing a web browser at the POWER-Up installer node. Individual files can be copied to client nodes using wget or curl if desired.

Dependent software packages The PowerAI Enterprise software is dependent on additional open source software that is not shipped with PowerAI Enterprise. Some of these dependent packages are downloaded to the POWER-Up software server from enabled yum repositories during the preparation phase and are subsequently available to the client nodes during the install phase. Additional software packages can be installed in the 'dependencies' repo on the POWER-Up software server by listing them in the power-up/software/dependent-packages.list file. Entries in this file can be delimited by spaces or commas and can appear on multiple lines. Note that packages listed in the dependent-packages.list file are not automatically installed on client nodes unless needed by the PowerAI software. They can be installed on a client node explicitly using yum on the client node (ie yum install pkg-name). Alternatively, they can be installed on all client nodes at once using Ansible (run from within the power-up/playbooks/ directory):

```
$ ansible all -i software_hosts -m yum -a "name=pkg-name"
```

or on a subset of nodes (eg the master nodes)

```
$ ansible master -i software_hosts -m yum -a "name=pkg-name"
```

10.5 Uninstalling the POWER-Up Software

To uninstall the POWER-Up software and remove the software repositories, follow the instructions below;

1. Stop and remove the nginx web server:

```
$ sudo nginx -s stop
$ sudo yum erase nginx -y
```

2. If you wish to remove the http service from the firewall on this node:

```
$ sudo firewall-cmd --permanent --remove-service=http
$ sudo firewall-cmd --reload
```

3. If you wish to stop and disable the firewall service on this node:

```
$ sudo systemctl stop firewalld.service
$ sudo systemctl disable firewalld.service
```

4. Remove the yum.repo files created by the PAIE installer:

```
$ sudo rm /etc/yum.repos.d/cuda-local.repo
$ sudo rm /etc/yum.repos.d/cuda.repo
$ sudo rm /etc/yum.repos.d/dependencies-local.repo
$ sudo rm /etc/yum.repos.d/dependencies.repo
$ sudo rm /etc/yum.repos.d/epel-ppc64le-local.repo
$ sudo rm /etc/yum.repos.d/epel-ppc64le.repo
$ sudo rm /etc/yum.repos.d/power-ai-local.repo
$ sudo rm /etc/yum.repos.d/nginx.repo
```

5. Remove the software server content and repositories:

```
$ sudo rm -rf /srv/anaconda
$ sudo rm -rf /srv/cuda-dnn
$ sudo rm -rf /srv/cuda-nccl2
$ sudo rm -rf /srv/power-ai
$ sudo rm -rf /srv/powerai-enterprise-license
$ sudo rm -rf /srv/spectrum-dli
$ sudo rm -rf /srv/spectrum-conductor
$ sudo rm -rf /srv/repos
```

6. Remove the yum cache data:

```
$ sudo rm -rf /var/cache/yum/ppc64le/7Server/cuda/
$ sudo rm -rf /var/cache/yum/ppc64le/7Server/cuda-local/
$ sudo rm -rf /var/cache/yum/ppc64le/7Server/dependencies/
$ sudo rm -rf /var/cache/yum/ppc64le/7Server/dependencies-local/
$ sudo rm -rf /var/cache/yum/ppc64le/7Server/epel-ppc64le/
$ sudo rm -rf /var/cache/yum/ppc64le/7Server/epel-ppc64le-local/
```

(continues on next page)

(continued from previous page)

```
$ sudo rm -rf /var/cache/yum/ppc64le/7Server/power-ai-local/
$ sudo rm -rf /var/cache/yum/ppc64le/7Server/nginx/
```

7. Uninstall the PowerAI Enterprise license program from the installer node. If you extracted the PowerAI Enterprise package on this node and accepted the enterprise license:

```
$ sudo yum erase powerai-enterprise-license -y
```

8. **Uninstall the PowerUp Software**

   - Assuming you installed from your home directory, execute:

```
$ sudo rm -rf ~/power-up
```

CHAPTER 11

Cluster Configuration File Specification

Specification Version: v2.0

Deployment of the OpenPOWER Cloud Reference Cluster is controlled by the 'config.yml' file. This file is stored in YAML format. The definition of the fields and the YAML file format are documented below.

Each section represents a top level dictionary key:

version:
globals:
location:
deployer:
switches:
interfaces:
networks:
node_templates:
software_bootstrap:

11.1 version:

Element	Example(s)	Description	Required
`version:`	`version: v2.0`	Config file version. Release Branch Supported Config File Version release-2.x version: v2.0 release-1.x version: 1.1 release-0.9 version: 1.0	**yes**

11.2 globals:

```
globals:
    introspection:
    env_variables:
    switch_mode_mgmt:
    switch_mode_data:
    dhcp_lease_time:
```

Element	Example(s)	Description	Required
globals: introspection: ...	introspection: true	Introspection shall be enabled. Evaluates to *false* if missing. *false* *true*	no
globals: env_variables: ...	env_variables: https_proxy: ↪http://192.168.1.↪2:3128 http_proxy: ↪http://192.168.1.↪2:3128 no_proxy: ↪localhost,127.0.↪0.1	Apply environmental variables to the shell. The example to the left would give the following result in bash: export https_proxy="http://192.168.1.2:3128" export http_proxy="http://192.168.1.2:3128" export no_proxy="localhost,127.0.0.1"	no
globals: switch_mode_↪mgmt: ...	switch_mode_mgmt: ↪active	Sets POWER-Up management switch mode. Evaluates to *active* if missing. *passive* *active*	no
globals: switch_mode_↪data: ...	switch_mode_data: ↪active	Sets POWER-Up data switch mode. Evaluates to *active* if missing. *passive* *active*	no
globals: dhcp_lease_time: ...	dhcp_lease_time: ↪15m dhcp_lease_time: 1h	Sets DHCP lease time given to client nodes. Value can be in seconds, minutes (e.g. "15m"), hours (e.g. "1h") or "infinite" (lease does not expire).	no

11.3 location:

```
location:
    time_zone:
    data_center:
    racks:
        - label:
```

(continues on next page)

(continued from previous page)

```
          room:
          row:
          cell:
```

Element	Example(s)	Description	Required

`location:`
`  time_zone:`
`  ...`
	`time_zone: UTC`
	
	`time_zone: America/Chicago`
		Cluster time zone in tz database format.	no

`location:`
`  data_center:`
`  ...`
	`data_center: East Coast`
	
	`data_center: Austin, TX`
		Data center name to be associated with cluster inventory.	no

`location:`
`  racks:`
`  - label:`
`  room:`
`  row:`
`  cell:`
`  ...`
	`racks:`
	`  - label: rack1`
	`  room: lab41`
	`  row: 5`
	`  cell: B`
	`  - label: rack2`
	`  room: lab41`
	`  row: 5`
	`  cell: C`
		List of cluster racks.
		
		Required keys:
		label - Unique label used to reference this rack elsewhere in the config file.
		
		Optional keys:
		room - Physical room location of rack.
		row - Physical row location of rack.
		cell - Physical cell location of rack.
			yes

11.4 deployer:

```
deployer:
    gateway:
    networks:
        mgmt:
            - device:
              interface_ipaddr:
              container_ipaddr:
              bridge_ipaddr:
              vlan:
              netmask:
              prefix:

        client:
```

(continues on next page)

(continued from previous page)

```
            - type:
              device:
              container_ipaddr:
              bridge_ipaddr:
              vlan:
              netmask:
              prefix:
```

Element	Example(s)	Description	Required
deployer: gateway: ...	gateway: true	Deployer shall act as cluster gateway. Evaluates to *false* if missing. *false* *true* The deployer will be configured as the default gateway for all client nodes. Configuration includes adding a 'MASQUERADE' rule to the deployer's 'iptables' NAT chain and setting the 'dnsmasq' DHCP service to serve the deployer's client management bridge address as the default gateway. Note: Specifying the 'gateway' explicitly on any of the data networks will override this behaviour.	no
deployer: networks: mgmt: - device: interface_ipaddr: container_ipaddr: bridge_ipaddr: vlan: netmask: prefix:	mgmt: - device: enp1s0f0 interface_ipaddr: 192.168.1.2 netmask: 255.255.255.0 - device: enp1s0f0 container_ipaddr: 192.168.5.2 bridge_ipaddr: 192.168.5.3 vlan: 5 prefix: 24	Management network interface configuration. Required keys: *device* - Management network interface device. Optional keys: *vlan* - Management network vlan (tagged). IP address must be defined with: *interface_ipaddr* - Management interface IP address (non-tagged). — or — *container_ipaddr* - Container management interface IP address (tagged). *bridge_ipaddr* - Deployer management bridge interface IP address (tagged).	**yes**

11.5 switches:

```
switches:
    mgmt:
        - label:
          hostname:
          userid:
          password:
          ssh_key:
          class:
          rack_id:
          rack_eia:
          interfaces:
              - type:
                ipaddr:
                vlan:
                port:
          links:
              - target:
                ipaddr:
                vip:
                netmask:
                prefix:
                ports:
    data:
        - label:
          hostname:
          userid:
          password:
          ssh_key:
          class:
          rack_id:
          rack_eia:
          interfaces:
              - type:
                ipaddr:
                vlan:
                port:
          links:
              - target:
                ipaddr:
                vip:
                netmask:
                prefix:
                ports:
```

Element	Example(s)	Description	Required
switches: mgmt: - label: hostname: userid: password: class: rack_id: rack_eia: ␣ ↪interfaces: -␣ ↪type: ␣ ↪ipaddr: ␣ ↪vlan: ␣ ↪port: links: -␣ ↪target: ␣ ↪ports: ...	mgmt: - label: mgmt_ ↪switch hostname:␣ ↪switch23423 userid: admin password:␣ ↪abc123 class: lenovo rack_id:␣ ↪rack1 rack_eia: 20 interfaces: - type:␣ ↪outband ␣ ↪ipaddr: 192.168. ↪1.10 port:␣ ↪mgmt0 - type:␣ ↪inband ␣ ↪ipaddr: 192.168. ↪5.20 port:␣ ↪15 links: -␣ ↪target: deployer ports:␣ ↪1 -␣ ↪target: data_ ↪switch ports:␣ ↪2	Management switch configuration. Each physical switch is defined as an item in the *mgmt:* list. Required keys: *label* - Unique label used to reference this switch elsewhere in the config file. Required keys in "active" switch mode: *userid* - Userid for switch management account. *password*[1] - Plain text password associated with *userid*. *ssh_key*[1] - Path to SSH private key file associated with *userid*. Required keys in "passive" switch mode: *class* - Switch class (lenovo/mellanox/cisco/cumulus). Optional keys: *hostname* - Hostname associated with switch management network interface. *rack_id* - Reference to rack *label* defined in the *locations: racks:=* element. *rack_eia* - Switch position within rack. *interfaces* - See *interfaces*. *links* - See *links*.	**yes**
switches: data: - label: hostname: userid: password: class:	example #1: data: - label: data_ ↪switch_1 hostname:␣ ↪switch84579 userid: admin password:␣	Data switch configuration. Each physical switch is defined as an item in the *data:* list. Key/value specs are identical to *mgmt switches*.	**yes**

11.6 interfaces:

```
interfaces:
    - label:
      description:
      iface:
      method:
      address_list:
      netmask:
      broadcast:
      gateway:
      dns_search:
      dns_nameservers:
      mtu:
      pre_up:
      vlan_raw_device:
    - label:
      description:
      DEVICE:
      BOOTPROTO:
      ONBOOT
      ONPARENT
      MASTER
      SLAVE
      BONDING_MASTER
      IPADDR_list:
      NETMASK:
      BROADCAST:
      GATEWAY:
      SEARCH:
      DNS1:
      DNS2:
      MTU:
      VLAN:
```

[1] Either *password* or *ssh_key* shall be specified, but not both.

Element	Example(s)	Description	Required
interfaces: - ... - ...		List of OS interface configuration definitions. Each definition can be formatted for either *Ubuntu* or *RHEL*.	no
interfaces: - label: description: iface: method: address_list: netmask: broadcast: gateway: dns_search: dns_nameservers: mtu: pre_up: vlan_raw_device:	- label: manual1 description: manual network 1 iface: eth0 method: manual - label: dhcp1 description: dhcp interface 1 iface: eth0 method: dhcp - label: static1 description: static interface 1 iface: eth0 method: static address_list: - 9.3.89.14 - 9.3.89.18-9.3.89.22 - 9.3.89.111-9.3.89.112 - 9.3.89.120 netmask: 255.255.255.0 broadcast: 9.3.89.255 gateway: 9.3.89.1 dns_search: your.dns.com dns_nameservers: 9.3.1.200 9.3.1.201 mtu: 9000 pre_up: command - label: vlan1 description: vlan interface 1 iface: eth0.10 method: manual - label: vlan2 description: vlan interface 2 iface: myvlan.20 method: manual vlan_raw_device: eth0	Ubuntu formatted OS interface configuration. Required keys: *label* - Unique label of interface configuration to be referenced within *networks:* *node_templates:* *interfaces:*. Optional keys: *description* - Short description of interface configuration to be included as a comment in OS config files. *address_list* - List of IP address to assign client interfaces referencing this configuration. Each list element may either be a single IP address or a range (formatted as <*start_address*>-<*end_address*>). *address_start* - Starting IP address to assign client interfaces referencing this configuration. Addresses will be assigned to each client interface incrementally. Optional “drop-in” keys: The following key names are derived directly from the Ubuntu *interfaces* configuration file (note that all “-” characters are replaced with “_”). Values will be	no
	- label: bridge1 description: bridge interface 1 iface: br1		

11.7 networks:

```
networks:
    - label:
      interfaces:
```

Element	Example(s)	Description	Required
networks: - label: interfaces:	interfaces: - label:␣ ↪example1 ... - label:␣ ↪example2 ... - label:␣ ↪example3 ... networks: - label: all_ ↪nets interfaces: -␣ ↪example1 -␣ ↪example2 -␣ ↪example3 - label: group1 interfaces: -␣ ↪example1 -␣ ↪example2 - label: group2 interfaces: -␣ ↪example1 -␣ ↪example3	The 'networks' list defines groups of interfaces. These groups can be assigned to items in the *node_templates:* list. Required keys: *label* - Unique label of network group to be referenced within a *node_templates:* item's 'networks:' value. *interfaces* - List of interfaces assigned to the group.	no

11.8 node_templates:

```
node_templates:
    - label:
      ipmi:
          userid:
          password:
      os:
          hostname_prefix:
          domain:
          profile:
```

(continues on next page)

(continued from previous page)

```
        install_device:
        users:
            - name:
              password:
        groups:
            - name:
        kernel_options:
        redhat_subscription:
    physical_interfaces:
        ipmi:
            - switch:
              ports:
        pxe:
            - switch:
              interface:
              rename:
              ports:
        data:
            - switch:
              interface:
              rename:
              ports:
    interfaces:
    networks:
    roles:
```

Element	Example(s)	Description	Required
node_templates: - label: ipmi: os: physical_interfaces: interfaces: networks: roles:	- label: controllers ipmi: userid: admin password: pass os: hostname_prefix: ctrl domain: ibm.com profile: ubuntu-14.04-server-ppc64el install_device: /dev/sda kernel_options: quiet physical_interfaces: ipmi: - switch: mgmt_switch_1 ports: - 1 - 3 - 5 pxe: - switch: mgmt_switch_1 ports: - 2 - 4 - 6	Node templates define client node configurations. Existing IPMI credentials and network interface physical connection information must be given to allow Cluster POWER-Up to connect to nodes. OS installation characteristics and post install network configurations are also defined. Required keys: *label* - Unique label used to reference this template. *ipmi* - IPMI credentials. See *node_templates: ipmi*. *os* - Operating system configuration. See *node_templates: os*. *physical_interfaces* - Physical network interface port mappings. See *node_templates: physical_interfaces*. Optional keys: *interfaces* - Post-deploy interface assignments. See *node_templates: interfaces*. *networks* - Post-deploy network (interface group) assignments. See *node_templates: networks*. *roles* - Ansible group assignment. See *node_templates: roles*.	**yes**
11.8. node_templates:			**55**
node_templates:	- label: ppc64el	Client node IPMI credentials. Note that IPMI cre-	**yes**

11.9 software_bootstrap:

```
software_bootstrap:
    - hosts:
      executable:
      command:
```

Element	Example(s)	Description	Required

`software_bootstrap:`
`  - hosts:`
`  executable:`
`  command:`
	`software_bootstrap:`
	`  - hosts: all`
	`  command: apt-get update`
	`  - hosts: openstackservers`
	`  executable: /bin/bash`
	`  command: |`
	`  set -e`
	`  apt update`
	`  apt upgrade -y`
		Software bootstrap defines commands to be run on client nodes after POWER-Up completes. This is useful for various additional configuration activities, such as bootstrapping additional software package installations.
		
		Required keys:
		hosts - Hosts to run commands on. The value can be set to 'all' to run on all hosts, node_template labels, or role/group names.
		command - Command to run.
		
		Optional keys:
		executable - Path to shell used to execute the command.
			no

CHAPTER 12

Cluster Inventory File Specification

Specification Version: v2.0

TODO: Short description of *inventory.yml* and how it should be used.

Each section represents a top level dictionary key:

12.1 version:

Element	Example(s)	Description	Required
`version:`	`version: v2.0`	Inventory file version. Release Branch Supported Inventory File Version release-2.x version: v2.0 release-1.x version: 1.0 release-0.9 version: 1.0	**yes**

12.2 location:

See *Config Specification - Location Section*.

12.3 switches:

See *Config Specification - Switches Section*.

12.4 nodes:

```
nodes:
    - label:
      hostname:
      rack_id:
      rack_eia:
      ipmi:
          switches:
          ports:
          userid:
          password:
          ipaddrs:
          macs:
      pxe:
          switches:
          ports:
          devices:
          ipaddrs:
          macs:
          rename:
      data:
          switches:
          ports:
          devices:
          macs:
          rename:
      os:
      interfaces:
```

Element	Example(s)	Description	Required

`nodes:`
`  label:`
`  ...`
	`label: ubuntu-servers`	Type.	**yes**

`nodes:`
`  hostname:`
`  ...`
	`hostname: server-1`	Hostname.	**yes**

`nodes:`
`  rack_id:`
`  ...`
	`rack_id: rack_1`	Rack ID.	no

`nodes:`
`  rack_eia:`
`  ...`
	`rack_eia: U10`	Rack EIA.	no

`nodes:`
`  ipmi:`
`  switches:`
`  ports:`
`  ipaddr:`
`  mac:`
`  userid:`
`  password:`
`  ...`
	`nodes:`
	`  ipmi:`
	`  switches:`
	`  - mgmt_1`
	`  - mgmt_2`
	`  ports:`
	`  - 1`
	`  - 11`
	`  ipaddrs:`
	`  - 10.0.0.1`
	`  - 10.0.0.2`
	`  macs:`
	`  - 01:23:45:67:89:AB`
	`  - 01:23:45:67:89:AC`
	`  userid: user`
	`  password: passw0rd`
		IPMI related parameters.
		
		Required keys:
		switches - Management switches.
		ports - Management ports.
		ipaddrs - IPMI interface ipaddrs.
		macs - IPMI interface MAC addresses.
		userid - IPMI userid.
		password - IPMI password.
		
		List items are correlated by index.
			yes

`nodes:`
`  pxe:`
`  switches:`
`  ports:`
`  devices:`
`  ipaddrs:`
`  macs:`
`  rename:`
`  ...`
	`nodes:`
	`  pxe:`
	`  switches:`
	`  - mgmt_1`
	`  - mgmt_2`
	`  ports:`
	`  - 2`
	`  - 12`
	`  devices:`
	`  - eth16`
	`  - eth17`
	`  ipaddrs:`
	`  - 10.0.1.1`
	`  - 10.0.1.2`
	`  macs:`
	`  - 01:23:45:67:89:AD`
	`  - 01:23:45:67:89:AE`
		PXE related parameters.
		
		Required keys:
		switches - Management switches.
		ports - Management ports.
		devices - Network devices.
		ipaddrs - Interface ipaddrs.
		macs - Interface MAC addresses.
		rename - Interface rename flags.
		
		List items are correlated
			yes

CHAPTER 13

Multiple Tenant Support

POWER-Up has the ability to segment a physical cluster into multiple isolated groups of nodes, allowing multiple users / tenants to use the cluster at the same time while maintaining complete isolation between tenants.

The process of sub-dividing a cluster into multiple groups is simple. You create a config.yml file for each group of nodes and deploy the groups one at a time. Each group must have a unique PXE and IPMI subnet and vlan number. The mgmt network can be common for all groups. POWER-Up creates a container and isolated networks on the deployer for each tenant in the cluster. A symbolic link to the inventory.yml file for each group is created in the power-up directory with the name inventoryn.yml where n is the number of the pxe vlan for the group.

As an example, the figure above shows a basic cluster with four nodes. To configure these into two groups of two nodes, create a config file for each group. Edit the deployer section of each config file and under the client subsection, specify a unique container_ipaddr, bridge_ipaddr and vlan for the ipmi and pxe networks for each group of nodes.

For example, the two groups could be configured as below;

Group 1:

```
deployer:
    networks:
        mgmt:
            - device: enP10p1s0f0
              interface_ipaddr: 192.168.16.3
              netmask: 255.255.255.0
        client:
            - device: enP10p1s0f0
              type: ipmi
              container_ipaddr: 192.168.30.2
              bridge_ipaddr: 192.168.30.3
              netmask: 255.255.255.0
              vlan: 30
            - device: enP10p1s0f0
              type: pxe
              container_ipaddr: 192.168.40.2
```

(continues on next page)

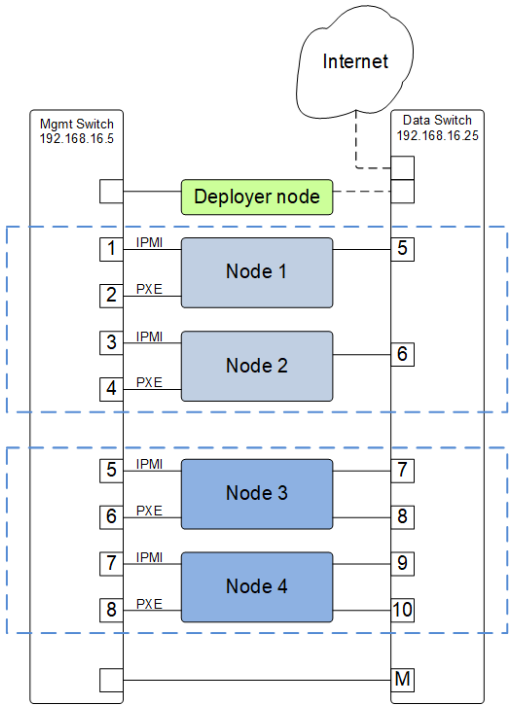

Fig. 1: POWER-Up Support for multiple tenants

(continued from previous page)

```
            bridge_ipaddr: 192.168.40.3
            netmask: 255.255.255.0
            vlan: 40
```

Group 2:

```
deployer:
    networks:
        mgmt:
            - device: enP10p1s0f0
              interface_ipaddr: 192.168.16.3
              netmask: 255.255.255.0
        client:
            - device: enP10p1s0f0
              type: ipmi
              container_ipaddr: 192.168.31.2
              bridge_ipaddr: 192.168.31.3
              netmask: 255.255.255.0
              vlan: 31
            - device: enP10p1s0f0
              type: pxe
              container_ipaddr: 192.168.41.2
              bridge_ipaddr: 192.168.41.3
              netmask: 255.255.255.0
              vlan: 41
```

Next, edit the switch ports list in the node_templates section of each config file;

Group 1:

```
node_templates:
    - label: ubuntu1604-node
      ipmi:
          userid: ADMIN
          password: admin
      os:
          profile: ubuntu-16.04-server-ppc64el
          users:
              - name: user1
                password: $6$Utk.IILMG9.$EepS/sIgD4aA.qYQ3voZL9yI3/5Q4vv.
↪p2s4sSmfCLAJlLAuaEmXDizDaBmJYGqHpobwpU2l4rJW.uUY4WNyv.
                groups: sudo
          install_device: /dev/sdj
      physical_interfaces:
          ipmi:
              - switch: mgmt1
                ports:
                    - 1
                    - 3
          pxe:
              - switch: mgmt1
                interface: pxe-ifc
                rename: true
                ports:
                    - 2
                    - 4
          data:
              - switch: data1
                interface: static_1
                rename: true
                ports:
                    - 5
                    - 6
```

Group 2:

```
node_templates:
    - label: ubuntu1604-node
      ipmi:
          userid: ADMIN
          password: admin
      os:
          profile: ubuntu-16.04-server-ppc64el
          users:
              - name: user1
                password: $6$Utk.IILMG9.$EepS/sIgD4aA.qYQ3voZL9yI3/5Q4vv.
↪p2s4sSmfCLAJlLAuaEmXDizDaBmJYGqHpobwpU2l4rJW.uUY4WNyv.
                groups: sudo
          install_device: /dev/sdj
      physical_interfaces:
          ipmi:
              - switch: mgmt1
                ports:
                    - 5
                    - 7
          pxe:
              - switch: mgmt1
```

(continues on next page)

(continued from previous page)

```
                interface: pxe-ifc
                rename: true
                ports:
                    - 6
                    - 8
        data:
            - switch: data1
              interface: static_1
              rename: true
              ports:
                    - 7
                    - 9
        data:
            - switch: data1
              interface: static_2
              rename: true
              ports:
                    - 8
                    - 10
```

For a complete config file for a basic cluster, See *Appendix-D*

Assuming your two config files are named config-T1.yml and config.T2.yml and residing in the power-up directory, to deploy the two groups:

```
pup deploy config-T1.yml
```

After the first deploy completes:

```
pup deploy config-T2.yml
```

Note

POWER-Up does not currently support the execution of two deploys at the same time. When deploying multiple groups of nodes, the groups must be deployed sequentially.

Note that if you move a node from an already deployed group to a new group, it can take up to one hour for it's IPMI IP lease to expire. If the node is moved to a new subnet before the lease expires you will not be able to access the nodes IPMI system until it renews it's IP lease in the new subnet. To avoid this, you can manually cycle power to the node. Alternately, you can use the ipmitool to reset the BMC of the node to be moved:

```
ipmitool -I lanplus -H 192.168.30.21 -U ADMIN -P admin mc reset cold
```

then immediately run:

```
pup config --mgmt-switches new-group-config.yml
```

CHAPTER 14

Developer Guide

POWER-Up development is overseen by a team of IBM engineers.

14.1 Git Repository Model

Development and test is orchestrated within the *master* branch. Stable *release-x.y* branches are created off *master* and supported with bug fixes. Semantic Versioning is used for release tags and branch names.

14.2 Coding Style

Code should be implemented in accordance with PEP 8 – Style Guide for Python Code.

14.3 Commit Message Rules

- **Subject line**
  - First line of commit message provides a short description of change
  - Must not exceed 50 characters
  - First word after tag must be capitalized
  - Must begin with one of the follwoing subject tags:

```
feat:      New feature
fix:       Bug fix
docs:      Documentation change
style:     Formatting change
refactor:  Code change without new feature
test:      Tests change
```

(continues on next page)

(continued from previous page)

```
chore:     Miscellaneous no code change
Revert     Revert previous commit
```

- **Body**
  - Single blank line seperates subject line and message body
  - Contains detailed description of change
  - Lines must not exceed 72 characters
  - Periods must be followed by single space

Your Commit message can be validated within the tox environment (see below for setup of the tox environment):

```
power-up$ tox -e commit-message-validate
```

14.4 Unit Tests and Linters

14.4.1 Tox

Tox is used to manage python virtual environments used to run unit tests and various linters.

To run tox first install python dependencies:

```
power-up$ ./scripts/install.sh
```

Install tox:

```
power-up$ pip install tox
```

To run all tox test environments:

```
power-up$ tox
```

List test environments:

```
power-up$ tox -l
py36
bashate
flake8
ansible-lint
commit-message-validate
verify-copyright
file-format
```

Run only 'flake8' test environment:

```
power-up$ tox -e flake8
```

14.4.2 Unit Test

Unit test scripts reside in the *power-up/tests/unit/* directory.

Unit tests can be run through tox:

```
power-up$ tox -e py36
```

Or called directly through python (be mindful of your python environment!):

```
power-up$ python -m unittest discover
```

14.4.3 Linters

Linters are required to run cleanly before a commit is submitted. The following linters are used:

- Bash: bashate
- Python: pycodestyle/flake8/pylint
- Ansible: ansible-lint

Linters can be run through tox:

```
power-up$ tox -e bashate
power-up$ tox -e flake8
power-up$ tox -e ansible-lint
```

Or called directly (again, be mindful of your python environment!)

Pylint and *pycodestyle* validation is not automatically launched when issuing the *tox* command. They need to be called out explicitly:

```
power-up$ tox -e pycodestyle
power-up$ tox -e pylint
power-up$ tox -e pylint-errors
```

14.4.4 File Format Validation

Ensure that each text file is in *unix* mode where lines are terminated by a linefeed:

```
power-up$ tox -e file-format
```

14.4.5 Copyright Date Validation

If any changed files include a copyright header the year must be current. This rule is enforced within a tox environment:

```
power-up$ tox -e verify-copyright
```

CHAPTER 15

Building the Introspection Kernel and Filesystem

Note: Introspection is not yet supported in POWER-Up 2.0

Introspection enables the clients to boot a Linux mini-kernel and filesystem prior to deployment. This allows POWER-Up to extract client hardware resource information and provides an environment for users to run configuration scripts (e.g. RAID volume management).

15.1 Building

1. By default, the introspection kernel is built automatically whenever one of the following commands are executed, and the introspection option is enabled in the config.yml file

```
cd power-up/playbooks
ansible_playbook -i hosts lxc-create.yml -K
ansible_playbook -i hosts lxc-introspect.yml -K
ansible_playbook -i hosts introspection_build.yml -K

or

gen deploy #if introspection was specified in the config.yml file
```

2. Wait for introspection_build.yml playbook to complete. If the rootfs.cpio.gz and vmlinux images already exist, the playbook will not rebuild them.
3. The final kernel and filesystem will be copied from the deployer container to the host filesystem under 'power-up/os-images/introspection'

15.1.1 Buildroot Config Files

Introspection includes a default buildroot and linux kernel config files.

These files are located in introspection/configs directory under power-up.

If there are any additional features or packages that you wish to add to the introspection kernel, they can be added to either of the configs prior to setup.sh being executed.

15.2 Run Time

Average load and build time on a POWER8 Server(~24 mins)

15.3 Public Keys

To append a public key to the buildroot filesystem

1. Build.sh must have been run prior
2. Execute add_key.sh <key.pub>
3. The final updated filesystem will be placed into output/rootfs.cpio.gz

CHAPTER 16

Appendix - A Using the 'pup' Program

The 'pup' program is the primary interface to the Cluster POWER-Up software. Help can be accessed by typing:

```
pup -h
or
pup --help
```

Help is context sensitive and will give help appropriate for the argument. For example, 'pup setup -h' will provide help on the setup function.

Usage;

pup [command] [<args>] [options] [–help | -h]

Cluster POWER-Up has extensive logging capabilities. Logging can take place to the screen and a log file (power-up/logs/gen) and the logging level can be set individually for the screen and file. By default, file logging is set to debug and screen logging is set to info.

To enable detailed logging to the screen, add the -p debug option. For additional log level help, enter -h at the end of a pup command. (ie pup setup -h)

Auto completion is enabled for the pup program. At any level of command entry, a single tab will complete the current command if it is distinguishable. Double tabbing will list all available options for that level of command input.

The following top level commands are provided;

- config
- deploy
- post-deploy
- setup
- software
- utils
- validate

16.1 Bare Metal Deployment

The deploy command deploys your cluster;

pup deploy [config-file-name]

For bare metal deploy, POWER-Up goes through the following steps when you enter pup deploy;

- validate the config file
- sets up interfaces and networks on the deployer node
- configures the management switches
- discovers and validates the cluster hardware
- creates a container for hosting the rest of the POWER-Up software
- deploys operating systems to your cluster node
- sets up ssh keys and user accounts on your cluster nodes
- configures networking on your cluster nodes
- configures your data switches

After installing the operating systems, POWER-Up will pause and wait for input before executing the last 3 steps above. This provides a convenient place to check on the cluster hardware before proceeding. If desired, you can stop POWER-Up at that point and re-start later by entering 'pup post-deploy'.

It is sometimes useful when first bringing up the cluster hardware to be able to run the initial steps above individually. The following commands can be used to individually run / re-run the first four steps above:

```
pup validate --config-file [config-file-name]
pup setup --networks [config-file-name]
pup config --mgmt-switches [config-file-name]
pup validate --cluster-hardware [config-file-name]
```

Note that the above steps must initially be run in order. After successfully completing the above steps in order, they can be re-run individually. When isolating cluster hardware issues, it is useful to be able to re-run pup validate –cluster-hardware. pup validate –config-file may be run any time as often as needed.

16.2 Software Installation

POWER-Up provides the ability to install software to a cluster of nodes.

To deploy software;

pup software [{–prep, –install}] [software-name]

Software installation is broken into two phases. The 'prep' phase copies / downloads packages and binaries and syncs any specified repositories to the POWER-Up node. The nginx web server is installed and software is moved to the /srv directory and made available via the web server. The install phase creates linkages on the client nodes to repositories on the POWER-Up node and then installs and configures the software.

CHAPTER 17

Appendix - D Example system 1 - Basic Flat Cluster

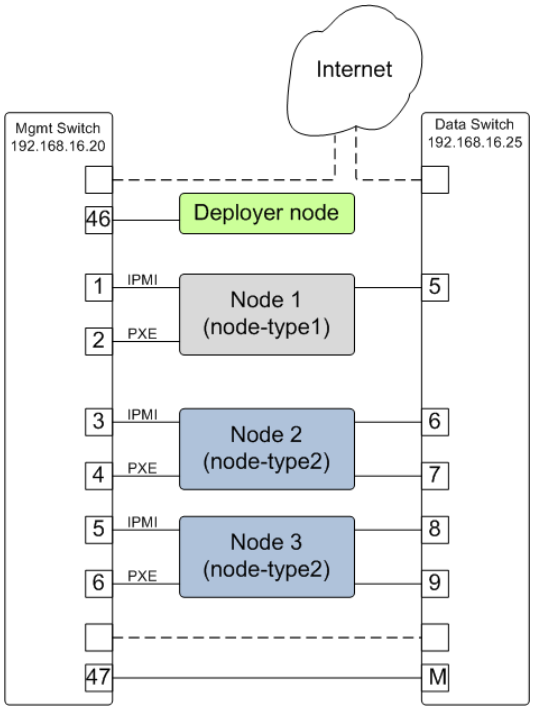

Fig. 1: A basic flat cluster with two node types

A Sample config.yml file for a basic flat cluster

The config file below defines two compute node templates with multiple network interfaces. The deployer node needs to have access to the internet which shown via one of the dotted line paths in the figure above or alternately via a wireless or dedicated interface.

```
---
# Copyright 2018 IBM Corp.
#
# All Rights Reserved.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

version: v2.0

globals:
    introspection: False
    switch_mode_mgmt: active

location:
    racks:
        - label: rack1

deployer:
    networks:
        mgmt:
            - device: enP10p1s0f0
              interface_ipaddr: 192.168.16.3
              netmask: 255.255.255.0
        client:
            - device: enP10p1s0f0
              type: ipmi
              container_ipaddr: 192.168.30.2
              bridge_ipaddr: 192.168.30.3
              netmask: 255.255.255.0
              vlan: 30
            - device: enP10p1s0f0
              type: pxe
              container_ipaddr: 192.168.40.2
              bridge_ipaddr: 192.168.40.3
              netmask: 255.255.255.0
              vlan: 40

switches:
    mgmt:
        - label: mgmt1
          class: lenovo
          userid: admin
          password: passw0rd
          interfaces:
              - type: outband
                ipaddr: 192.168.16.20
                port: 1
```

(continues on next page)

(continued from previous page)

```
          links:
              - target: deployer
                ports: 46
    # Note that there must be a data switch defined in the config file. In this
    # case the data and mgmt switch are the same physical switch
    data:
        - label: data1
          class: lenovo
          userid: admin
          password: passw0rd
          interfaces:
              - type: outband
                ipaddr: 192.168.16.25
          links:
              - target: deployer
                ports: 47

interfaces:
    - label: pxe-ifc
      description: pxe interface
      iface: eth0
      method: dhcp

    - label: static_1
      description: static network 1
      iface: eth1
      method: static
      address_list:
          - 192.168.1.2
          - 192.168.1.3
          - 192.168.1.4
      netmask: 255.255.255.0
      broadcast: 192.168.1.255
      gateway: 192.168.1.1

    - label: static_2
      description: static network 2
      iface: eth2
      method: static
      address_list:
          - 192.168.2.2
          - 192.168.2.3
          - 192.168.2.4
      netmask: 255.255.255.0
      broadcast: 192.168.2.255
      gateway: 192.168.2.1

networks:
    - label: static-ifc1
      interfaces:
          - static_1

node_templates:
    - label: node-type1
      ipmi:
          userid: ADMIN
          password: admin
```

(continues on next page)

(continued from previous page)

```
      os:
          profile: ubuntu-16.04-server-ppc64el
          users:
              - name: user1
                password: $6$Utk.IILMG9.$EepS/sIgD4aA.qYQ3voZL9yI3/5Q4vv.
↪p2s4sSmfCLAJlLAuaEmXDizDaBmJYGqHpobwpU2l4rJW.uUY4WNyv.
                groups: sudo
          install_device: /dev/sdj
      physical_interfaces:
          ipmi:
              - switch: mgmt1
                ports:
                    - 1
          pxe:
              - switch: mgmt1
                interface: pxe-ifc
                rename: true
                ports:
                    - 2
          data:
              - switch: data1
                interface: static_1
                rename: true
                ports:
                    - 5
    - label: node-type2
      ipmi:
          userid: ADMIN
          password: admin
      os:
          profile: ubuntu-16.04-server-ppc64el
          users:
              - name: user1
                password: $6$Utk.IILMG9.$EepS/sIgD4aA.qYQ3voZL9yI3/5Q4vv.
↪p2s4sSmfCLAJlLAuaEmXDizDaBmJYGqHpobwpU2l4rJW.uUY4WNyv.
                groups: sudo
          install_device: /dev/sdj
      physical_interfaces:
          ipmi:
              - switch: mgmt1
                ports:
                    - 3
                    - 5
          pxe:
              - switch: mgmt1
                interface: pxe-ifc
                rename: true
                ports:
                    - 4
                    - 6
          data:
              - switch: data1
                interface: static_1
                rename: true
                ports:
                    - 6
                    - 8
```

(continues on next page)

(continued from previous page)

```
            - switch: data1
              interface: static_2
              rename: true
              ports:
                  - 7
                  - 9
```

CHAPTER 18

Appendix - E Example system 2 - Basic Cluster with High Availability Network

The config file below defines two compute node templates and multiple network templates. The sample cluster can be configured with the provided config.yml file. The deployer node needs to have access to the internet for accessing packages.

Various OpenPOWER nodes can be used such as the S821LC. The deployer node can be OpenPOWER or alternately a laptop which does not need to remain in the cluster. The data switch can be Mellanox SX1700 or SX1410.

```
---
# Copyright 2018 IBM Corp.
#
# All Rights Reserved.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

version: v2.0

globals:
    introspection: False
    switch_mode_mgmt: active

location:
    time_zone: America/Chicago
    racks:
```

(continues on next page)

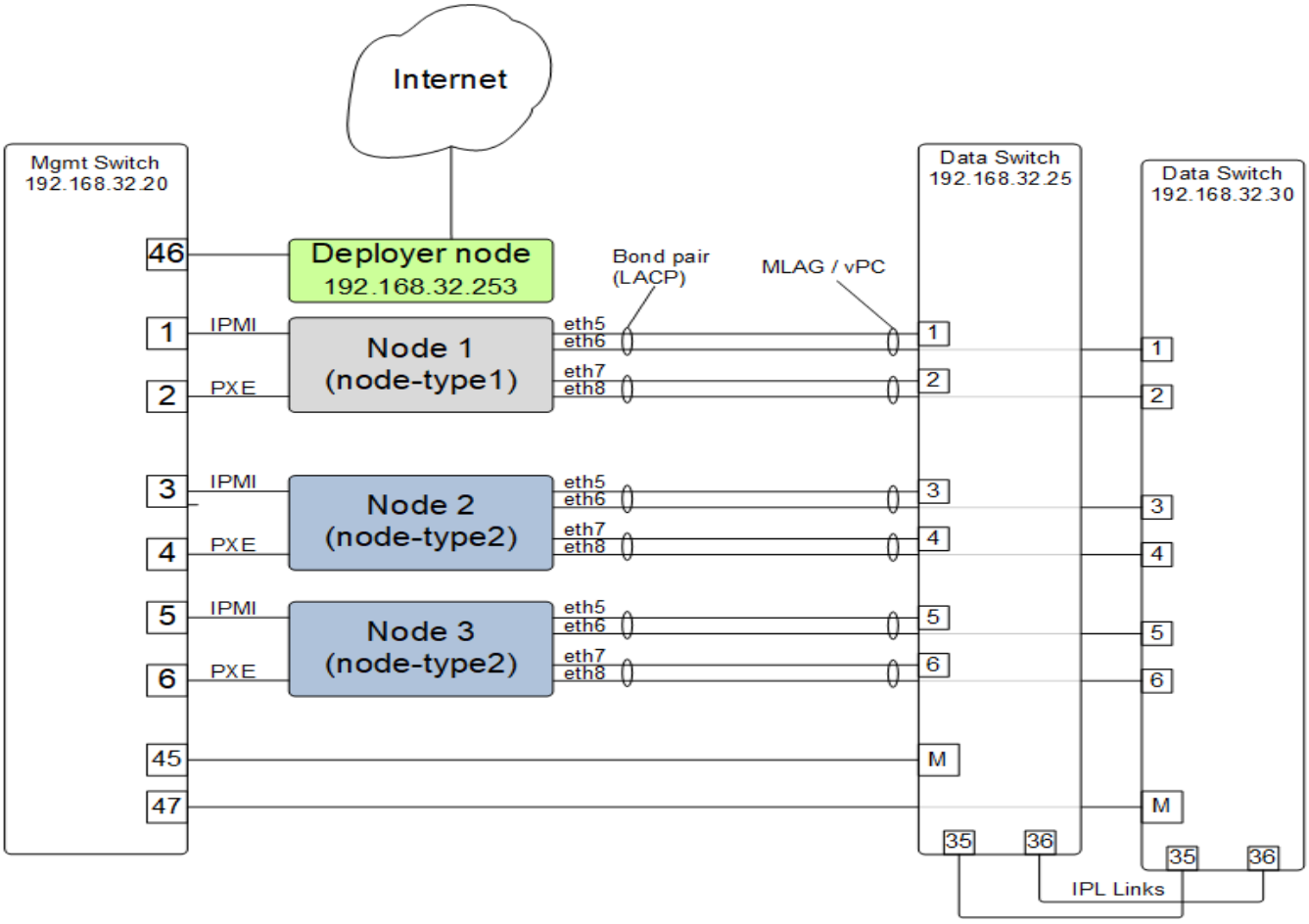

Fig. 1: High Availability Network using MLAG / vPC

(continued from previous page)

```
        - label: rack1

deployer:
    networks:
        mgmt:
            - device: enP1p10s0f0
              interface_ipaddr: 192.168.32.253
              prefix: 24
        client:
            - device: enP1p10s0f0
              type: pxe
              container_ipaddr: 192.168.10.2
              bridge_ipaddr: 192.168.10.3
              netmask: 255.255.255.0
              vlan: 10
            - device: enP1p10s0f0
              type: ipmi
              container_ipaddr: 192.168.12.2
              bridge_ipaddr: 192.168.12.3
              prefix: 24
              vlan: 12

switches:
    mgmt:
        - label: mgmt_1
          class: lenovo
          userid: admin
          password: passw0rd
          rack_id: rack1
          interfaces:
              - type: outband
                ipaddr: 192.168.32.20
                port: mgmt0
          links:
              - target: deployer
                ports:
                    - 1
              - target: data1_1
                ports:
                    - 2
              - target: data1_2
                ports:
                    - 3

    data:
        - label: data_1_1
          class: mellanox
          userid: admin
          password: passw0rd
          rack_id: rack1
          interfaces:
              - type: outband
                ipaddr: 192.168.32.25
                port: mgmt0
          links:
              - target: mgmt_1
                ports:
```

(continues on next page)

(continued from previous page)

```
                    - mgmt0
                - target: data1_2
                  ipaddr: 10.0.0.1
                  prefix: 24
                  vlan: 4000
                  ports:
                      - 35
                      - 36
          - label: data_1_2
            class: mellanox
            userid: admin
            password: passw0rd
            rack_id: rack1
            interfaces:
                - type: outband
                  ipaddr: 192.168.32.30
                  port: mgmt0
            links:
                - target: mgmt_1
                  ports: mgmt0
                - target: data1_1
                  ipaddr: 10.0.0.2
                  netmask: 255.255.255.0
                  vlan: 4000
                  ports:
                      - 35
                      - 36

interfaces:
    - label: pxe-ifc
      description: pxe interface
      iface: eth0
      method: dhcp

    - label: bond1_interface1
      description: primary interface for bond1
      iface: eth1
      method: manual
      bond_master: bond1
      bond_primary: eth0

    - label: bond1_interface2
      description: secondary interface for bond1
      iface: eth2
      method: manual
      bond_master: bond1

    - label: bond1
      description: bond interface 1
      iface: bond1
      bond_mode: active-backup
      bond_miimon: 100
      bond_slaves: none

    - label: bond1_vlan10
      description: vlan10 interface off bond1
      iface: bond1.10
```

(continues on next page)

(continued from previous page)

```
      method: manual

    - label: bond1_br10
      description: bridge interface off bond1 vlan10
      iface: br10
      method: static
      address_start: 172.16.10.1
      netmask: 255.255.255.0
      bridge_ports: bond1.10
      bridge_stp: off

    - label: bond1_vlan20
      description: vlan20 interface off bond1
      iface: bond1.20
      method: manual

    - label: bond1_br20
      description: bridge interface off bond1 vlan20
      iface: br20
      method: static
      address_start: 172.16.20.1
      netmask: 255.255.255.0
      bridge_ports: bond1.20
      bridge_stp: off

networks:
    - label: bond1_br10
      interfaces:
          - bond1_interface1
          - bond1_interface2
          - bond1
          - bond1_vlan10
          - bond1_br10

    - label: bond1_br20
      interfaces:
          - bond1_interface1
          - bond1_interface2
          - bond1
          - bond1_vlan20
          - bond1_br20

    - label: bond1_br10_br20
      interfaces:
          - bond1_interface1
          - bond1_interface2
          - bond1
          - bond1_vlan10
          - bond1_br10
          - bond1_vlan20
          - bond1_br20

node_templates:
    - label: controllers
      ipmi:
          userid: ADMIN
          password: admin
```

(continues on next page)

(continued from previous page)

```
      os:
          profile: ubuntu-16.04-server-ppc64el
          users:
              - name: user1
                password: $6$Utk.IILMG9.$EepS/sIgD4aA.qYQ3voZL9yI3/5Q4vv.
↪p2s4sSmfCLAJlLAuaEmXDizDaBmJYGqHpobwpU2l4rJW.uUY4WNyv.
                groups: sudo
          install_device: /dev/sdj
      physical_interfaces:
          ipmi:
              - switch: mgmt_1
                ports:
                    - 10
                    - 12
          pxe:
              - switch: mgmt_1
                interface: pxe-ifc
                rename: true
                ports:
                    - 11
                    - 13
          data:
              - switch: data_1_1
                interface: bond1_interface1
                rename: true
                ports:
                    - 18
                    - 19
              - switch: data_1_2
                interface: bond1_interface2
                rename: true
                ports:
                    - 18
                    - 19
      interfaces:

      networks:
          - bond1_br10_br20

    - label: compute
      ipmi:
          userid: ADMIN
          password: admin
      os:
          profile: ubuntu-16.04-server-ppc64el
          users:
              - name: user1
                password: $6$Utk.IILMG9.$EepS/sIgD4aA.qYQ3voZL9yI3/5Q4vv.
↪p2s4sSmfCLAJlLAuaEmXDizDaBmJYGqHpobwpU2l4rJW.uUY4WNyv.
                groups: sudo
          install_device: /dev/sdj
      physical_interfaces:
          ipmi:
              - switch: mgmt_1
                ports:
                    - 14
                    - 16
```

(continues on next page)

(continued from previous page)

```
          pxe:
              - switch: mgmt_1
                interface: pxe-ifc
                rename: true
                ports:
                    - 15
                    - 17
          data:
              - switch: data_1_1
                interface: bond1_interface1
                rename: true
                ports:
                    - 20
                    - 21
              - switch: data_1_2
                interface: bond1_interface2
                rename: true
                ports:
                    - 20
                    - 21
      interfaces:

      networks:
          - bond1_br10

    - label: storage
      ipmi:
          userid: ADMIN
          password: admin
      os:
          profile: ubuntu-16.04-server-ppc64el
          users:
              - name: user1
                password: $6$Utk.IILMG9.$EepS/sIgD4aA.qYQ3voZL9yI3/5Q4vv.
↪p2s4sSmfCLAJlLAuaEmXDizDaBmJYGqHpobwpU2l4rJW.uUY4WNyv.
                groups: sudo
          install_device: /dev/sdj
      physical_interfaces:
          ipmi:
              - switch: mgmt_1
                ports:
                    - 18
                    - 20
          pxe:
              - switch: mgmt_1
                interface: pxe-ifc
                rename: true
                ports:
                    - 19
                    - 21
          data:
              - switch: data_1_1
                interface: bond1_interface1
                rename: true
                ports:
                    - 22
                    - 23
```

(continues on next page)

(continued from previous page)

```
            - switch: data_1_2
              interface: bond1_interface2
              rename: true
              ports:
                  - 22
                  - 23
    interfaces:

    networks:
        - bond1_br20
```

CHAPTER 19

Appendix - F Detailed POWER-Up Flow (needs update)

This section not yet completed for POWER-Up 2.0.

CHAPTER 20

Appendix - G Configuring Management Access on the Lenovo G8052 and Mellanox SX1410

For the Lenovo G8052 switch, the following commands can be used to configure management access on interface 1. Initially the switch should be configured with a serial cable so as to avoid loss of communication with the switch when configuring management access. Alternately you can configure a second management interface on a different subnet and vlan.

Enable configuration mode and create vlan:

```
RS 8052> enable
RS 8052# configure terminal
RS 8052 (config)# vlan 16     (sample vlan #)
RS G8052(config-vlan)# enable
RS G8052(config-vlan)# exit
```

Enable IP interface mode for the management interface:

```
RS 8052 (config)# interface ip 1
```

Assign a static ip address, netmask and gateway address to the management interface. This must match the address specified in the config.yml file (keyname: ipaddr-mgmt-switch:) and be in a *different* subnet than your cluster management subnet. Place this interface in the above created vlan:

```
RS 8052 (config-ip-if)# ip address 192.168.16.20 (example IP address)
RS 8052 (config-ip-if)# ip netmask 255.255.255.0
RS 8052 (config-ip-if)# vlan 16
RS 8052 (config-ip-if)# enable
RS 8052 (config-ip-if)# exit
```

Configure the default gateway and enable the gateway:

```
ip gateway 1 address 192.168.16.1  (example ip address)
ip gateway 1 enable
```

Note: if you are SSH'd into the switch on interface 1, be careful not to cut off access if changing the ip address. If needed, additional management interfaces can be set up on interfaces 2, 3 or 4.

For the Mellanox switch, the following commands can be used to configure the MGMT0 management port;

switch (config) # no interface mgmt0 dhcp

switch (config) # interface mgmt0 ip address <IP address> <netmask>

For the Mellanox switch, the following commands can be used to configure an in-band management interface on an existing vlan ; (example vlan 10)

switch (config) # interface vlan 10

switch (config interface vlan 10) # ip address 10.10.10.10 /24

To check the config;

switch (config) # show interfaces vlan 10

CHAPTER 21

Appendix - H Recovering from POWER-Up Issues (needs update)

This section not yet updated for POWER-Up 2.0

CHAPTER 22

Appendix - I Using the 'teardown' Program

The 'teardown' program allows for select 'tear down' of the POWER-Up environment on the deployer node and cluster switches. It is primarily used when redeploying your cluster for test purposes, after taking corrective action after previous deployment failures or for removing the POWER-Up environment from the deployer node.

Similar to the pup program, teardown has built in help and supports tab completion.

Usage:

```
teardown <command> [<args>] [options] [-help | -h]
```

The teardown program can perform the following functions;

- Destroy the container associated with the current config.yml file. $ teardown deployer –container
- Undo the deployer network configuration associated with the current config.yml file $ teardown deployer –networks
- Undo the configuration of the data switches associated with the current config.yml file. $ teardown switches –data

NOTE: teardown actions are driven by the current config.yml file. If you wish to make changes to your cluster configuration, be sure to teardown the existing cluster configuration before changing your config.yml file.

For a typical re-deploy where the POWER-Up software does not need updating, you should teardown the deployer container and the data switches configuration.

CHAPTER 23

Indices and tables

- genindex
- modindex
- search