Cluster Genesis User Guide Documentation

Release 2.0b

Ray Harrington

Mar 21, 2018

Contents:

Version 2.0b

Date 2018-03-07

Document Owner OpenPOWER Cluster Genesis 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 Cluster Genesis toolkit. It is targeted at all users of the toolkit. Users are expected to have a working knowledge of Ethernet networking and Linux.

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
2.0b	Gemini	2018-03-07	TBD

CHAPTER 3

Introduction

Cluster POWER-Up enables greatly simplified configuration of clusters of bare metal OpenPOWER servers running Linux. 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. Cluster POWER-Up can configure simple flat networks for typical HPC environments or more advanced networks with VLANS and bridges for OpenStack environments. Complex heterogenous clusters can be easily deployed using POWER-Up's interface and node templates. Cluster POWER-Up configures the switches in the cluster with support for multiple switch vendors.

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)

Notes; 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 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 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. Using the Mellanox configuration wizard:

- 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
```

  for Lenovo switches:

```
spanning-tree mode disable
```

- enable SSH login:

```
ssh server enable
```

- If this switch has been used previously, delete any existing vlans which match those specified in *config.yml 'interfaces:'*. This insures that only those nodes specified in the config file have access to the cluster. (for a brand new switch this step can be ignored)
  - login to the switch:

```
enable
configure terminal
show vlan
```

    note those vlans that include the ports of the nodes to be included in the new cluster and remove those vlans or remove those ports from existing vlans:

```
no vlan n
```

- Save config. In switch config mode:

```
configuration write
```

- If using redundant data switches with MLAG, Leave the interswitch peer links (IPL) links disconnected until Cluster POWER-Up completes. (This avoids loops)

Initial configuration of management switch(es)

For out of box installation, it is usually necessary to configure the switch using a serial connection. See the switch installation guide. For additional info on Lenovo G8052 specific commands, see *Appendix-G* and the *Lenovo RackSwitch G8052 Installation guide*)

In order for Cluster POWER-Up to access and configure the switches in your cluster it is necessary to configure management access on all switches and provide management access information in the config.yml file.

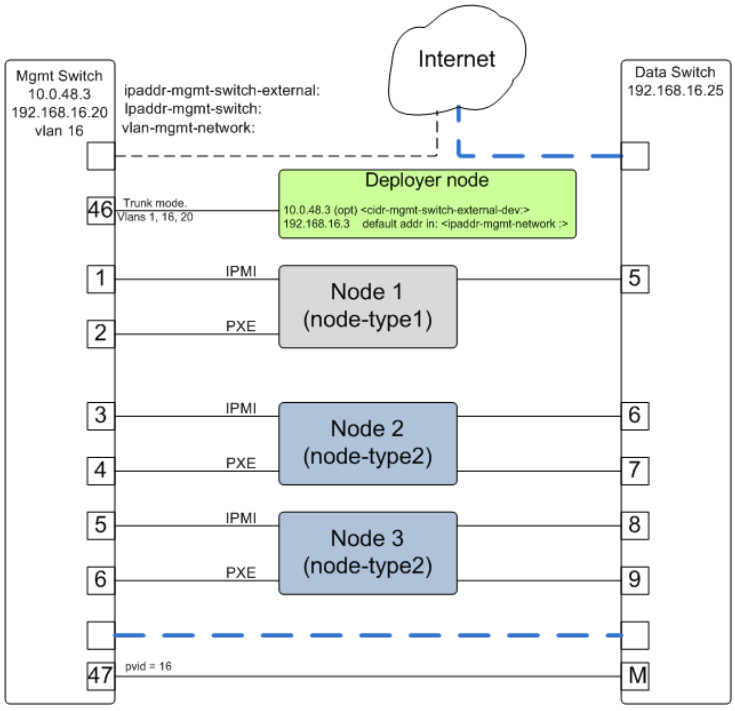

Fig. 1: POWER-Up setup of the switch management network

In this 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*.

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

- Switch IP Addresses:

```
switches:
    mgmt:
        - label: Mgmt_Switch
          userid: admin
          password: abc123
          interfaces:
              - type: inband
                ipaddr: 10.0.48.3
              - type: inband
                ipaddr: 192.168.16.20
                netmask: 255.255.255.0
                vlan: 16
          links:
              - target: deployer
                ports: 46
              - target: Data_Switch
                ports: 47
    data:
        - label: Data_Switch
          userid: admin
          password: abc123
          interfaces:
              - type: outband
                ipaddr: 192.168.16.25
                vlan: 16
                port: mgmt0
          links:
              - target: Mgmt_Switch
                ports: mgmt0
```

- Deployer 'mgmt' networks:

```
deployer:
    networks:
        mgmt:
            - device: enp1s0f0
              interface_ipaddr: 10.0.48.3
              netmask: 255.255.255.0
            - device: enp1s0f0
              container_ipaddr: 192.168.16.2
              bridge_ipaddr: 192.168.16.3
              netmask: 255.255.255.0
              vlan: 16
```

Management switch setup commands for the Lenovo G8052:

- 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 10.0.48.20  # example IP address
RS G8052(config-ip-if)# ip netmask 255.255.240.0
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
```

- Optionally configure a default gateway and enable the gateway:

```
RS G8052(config)# ip gateway 1 address 10.0.48.1  # example ip address
RS G8052(config)# ip gateway 1 enable
```

- 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
```

  For Lenovo switches:

```
enable
configure terminal
spanning-tree mode disable
```

- enable secure https and SSH login:

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

- Save the config (For Lenovo switches, enter config mode). For additional information, consult vendor documentation):

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

This completes normal POWER-Up initial configuration.

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 hardare 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 aquire complete MAC address information once vPC (AKA MLAG or VLAG) has been configured on the data switches.

4.2 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. For larger cluster deployments, 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 installations up to several racks.

The deployer node requires internet access. 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.

Operating Sytem and Package setup of the Deployer Node

- **Deployer OS Requirements:**
  - **Ubuntu**
    - Release 14.04LTS or 16.04LTS
    - SSH login enabled
    - sudo privileges
  - **RHEL**
    - Release 7.2 or later
    - Extra Packages for Enterprise Linux (EPEL) repository enabled (https://fedoraproject.org/wiki/EPEL)
    - SSH login enabled
    - sudo privileges
- Optionally, assign a static, public ip address to the BMC port to allow external control of the deployer node.
- login into the deployer and install the vim, vlan, bridge-utils and fping packages
  - Ubuntu:

```
$ sudo apt-get update
$ sudo apt-get install vim vlan bridge-utils fping
```

  - RHEL:

```
$ sudo yum install vim vlan bridge-utils fping
```

Network Configuration of the Deployer Node

Note: 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 sets up a vlan and subnet 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.

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*.

CHAPTER 5

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 *cluster-genesis/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 cluster-genesis
$ source deployenv/bin/activate
$ ./scripts/python/pup.py validate --config-file
```

5.1 Switch Mode

5.1.1 Active Switch Mode

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

5.1.2 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.

5.2 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*.

5.3 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*.

5.3.1 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.

5.3.2 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*.

5.4 Post Genesis 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 6

Cluster Configuration File Specification

Specification Version: v2.0

Genesis 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:

Additional key/value pairs are not processed by Cluster Genesis, but are copied into the inventory.yml file and made available to post-Genesis scripts and/or playbooks.

6.1 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**

6.2 globals:

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

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 Cluster Genesis management switch mode. Evaluates to *active* if missing. *passive* *active*	no
globals: switch_mode_ ↪data: ...	switch_mode_data: ↪active	Sets Cluster Genesis data switch mode. Evaluates to *active* if missing. *passive* *active*	no

6.3 location:

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

6.4 deployer:

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

        client:
            - type:
              device:
              container_ipaddr:
              bridge_ipaddr:
              vlan:
```

(continues on next page)

(continued from previous page)

```
            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: enp1s0f0`
`  interface_ipaddr: 192.168.1.2`
`  container_ipaddr: 192.168.5.2`
`  bridge_ipaddr: 192.168.5.3`
`  vlan: 5`
`  netmask: 24`
`  prefix: 24`
`  ...`
`  ...`
	`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

6.5 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**

6.6 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`		

6.7 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

6.8 node_templates:

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

(continues on next page)

(continued from previous page)

```
        users:
            - name:
              password:
        groups:
            - name:
        kernel_options:
    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 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**
6.8. node_templates:			**33**
node_templates:	- label: ppc64el	Client node IPMI credentials. Note that IPMI cre-	**yes**

6.9 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 Cluster Genesis 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 7

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:

7.1 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**

7.2 location:

See *Config Specification - Location Section*.

7.3 switches:

See *Config Specification - Switches Section*.

7.4 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 8

Running the OpenPOWER Cluster Configuration Software

8.1 Installing and Running the Genesis code. Step by Step Instructions

1. Verify that all the steps in section *4 Prerequisite Hardware Setup* have been executed. Genesis 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 Cluster Genesis:

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

5. Install the remaining software packages used by Cluster Genesis and setup the environment:

   ```
   $ cd cluster-genesis
   $ ./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. These lines can be removed using the "tear-down" 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.
7. copy your config.yml file to the ~/cluster-genesis 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 '/cluster-genesis/os_images' directory. Symbolic links to image files are also allowed.
9. For RHEL iso images, create a kickstart file having the same name as your iso image but with an extension of .ks. This can be done by copying the supplied kickstart file located in the /cluster-genesis/os_images/config directory. For example, if your RHEL iso is *RHEL-7.2-20151030.0-Server-ppc64le-dvd1.iso*, from within the */cluster-genesis/os_images/config directory*:

```
$ cp RHEL-7.x-Server.ks RHEL-7.2-20151030.0-Server-ppc64le-dvd1.ks
```

   (The cobbler-profile: key in your config.yml file should have a value of RHEL-7.2-20151030.0-Server-ppc64le-dvd1 (no .ks extension)*

   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 Cluster Genesis DHCP server will assign new addresses to the BMCs of all cluster nodes.

   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

   Most of Genesis' capabilities are accessed using the 'gen' program. For a complete overview of the gen program, see Appendix A.

10. To deploy operating systems to your cluster nodes:

```
$ gen deploy
```

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

11. This will create the management neworks, install the container that runs most of the Genesis functions and then optionally launch the introspection OS and then install OS's on the cluster nodes. This process can take as little as 30 minutes or as much as mutliple 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 gen program with a status request. (During install, you must allow Genesis to make updates to your .bashrc file in order to run gen functions from another terminal session):

```
$ gen status
```

After several minutes Cluster Genesis will have initialized and should display a list of cluster nodes which have obtained BMC addresses. Genesis will wait up to 30 minutes for the BMCs of all cluster nodes to reset and obtain an IP address. After 30 minutes, if there are nodes which have still not requested a DHCP address, Genesis will pause to give you an opportunity to make fixes. If any nodes are missing, verify cabling and verify the config.yml file. If necessary, recycle power to the missing nodes. See "Recovering from Genesis Issues" in the appendices for additional debug help. You can monitor which nodes have obtained ip addresses, by executing the following from another window:

```
$ gen status
```

After Genesis completes the assignment of DHCP addresses to the cluster nodes BMC ports, Genesis will interrogate the management switches and read the MAC addresses associated with the BMC and PXE ports and initialize Cobbler to assign specific IP addresses to the interfaces holding those MAC addresses.

After Genesis has assigned IP addresses to the BMC ports of all cluster nodes, it will display a list of all nodes. Genesis will wait up to 30 minutes for the PXE ports of all cluster nodes to reset and obtain an IP address. After 30 minutes, if there are nodes which have still not requested a DHCP address, Genesis will pause to give you an opportunity to make fixes.

After all BMC and PXE ports have been discovered Genesis will begin operating system deployment.

12. Introspection

    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 Cluster Genesis will print a table of all introspection hosts, including their IP addresses and login / ssh private key credentials. This list is maintained in the 'cluster-genesis/playbooks/hosts' file under the 'introspections' group. Genesis will pause after the introspection OS deployement to allow for customized updates to the cluster nodes. Use ssh (future: or Ansible) to run custom scripts on the client nodes.

13. To continue the Genesis process, press enter and/or enter the sudo password

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

    ```
    $ gen 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 Genesis Issues" in the Appendices

    Genesis creates logs of it's activities. A file (log.txt) external to the Genesis container is written in the cluster-genesis directory. This can be viewed:

    ```
    $ gen log
    ```

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

    ```
    $ gen logc
    ```

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, Genesis performs several additional activities such as setting up networking on the cluster nodes, setup SSH keys and copy to cluster nodes, and configure the data switches. From the host namespace, execute:

```
$ gen post-deploy
```

If data switches are configured with MLAG verify

- The switch IPL ports are disabled or are not plugged in.
- No port channels are defined.

8.2 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).:

```
$ gen deploy-passive
```

When prompted, write each switch MAC address table to file in 'cluster-genesis/passive'. The files should be named to match the unique values set in the 'config.yml' 'ipaddr-mgmt-switch' dictionary. For example, take the following 'ipaddr-mgmt-switch' configuration:

```
ipaddr-mgmt-switch:
    rack1: passive_mgmt_rack1
    rack2: passive_mgmt_rack2
```

The user would need to write two files:

1. 'cluster-genesis/passive/passive_mgmt_rack1'
2. 'cluster-genesis/passive/passive_mgmt_rack2'

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 Lenovo G8052:

```
$ ssh <mgmt_switch_user>@<mgmt_switch_ip> \
'show mac-address-table' > ~/cluster-genesis/passive/passive_mgmt_rack1
```

Note that this command would need to be run for each individual mgmt switch, writing to a seperate 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:

```
$ gen deploy-passive-retry
```

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.:

```
$ gen post-deploy-passive
```

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

```
ipaddr-data-switch:
    base-rack: passive1
    rack2: passive2
    rack3: passive3
```

The user would need to write three files:

1. 'cluster-genesis/passive/passive1'
2. 'cluster-genesis/passive/passive2'
3. 'cluster-genesis/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:

```
$ ssh <data_switch_user>@<data_switch_ip> \
'cli en show\ mac-address-table' > ~/cluster-genesis/passive/passive1
```

Note that this command would need to be run for each individual data switch, writing to a seperate 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 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 with a "/". The string up to and including the "/" will be removed. (e.g. "Eth1/5" will be saved as "5").

Both Lenovo and Mellanox switches currently supported by Cluster Genesis 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
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
```

8.3 SSH Keys

The OpenPOWER Cluster Genesis 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 keypair is written to the ~/.ssh directory as id_rsa_ansible-generated and id_rsa_ansible-generated.pub. To login to one of the cluster nodes as root from the deployer node:

```
ssh -i ~/.ssh/id_rsa_ansible-generated 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 keyname *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 /home/deployer/cluster-genesis directory.

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

CHAPTER 9

Developer Guide

Cluster Genesis development is overseen by a team of IBM engineers.

9.1 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.

9.2 Coding Style

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

It is requested that modules contain appropriate *__future__* imports to simplify future migration to Python3.

9.3 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
```

(continues on next page)

(continued from previous page)

```
refactor:  Code change without new feature
test:      Tests change
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):

```
cluster-genesis$ tox -e commit-message-validate
```

9.4 Unit Tests and Linters

9.4.1 Tox

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

To run tox first install python dependencies:

```
cluster-genesis$ ./scripts/install.sh
```

Install tox:

```
cluster-genesis$ pip install tox
```

To run all tox test environments:

```
cluster-genesis$ tox
```

List test environments:

```
cluster-genesis$ tox -l
py27
bashate
flake8
ansible-lint
commit-message-validate
verify-copyright
file-format
```

Run only 'flake8' test environment:

```
cluster-genesis$ tox -e flake8
```

9.4.2 Unit Test

Unit test scripts reside in the *cluster-genesis/tests/unit/* directory.

Unit tests can be run through tox:

```
cluster-genesis$ tox -e py27
```

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

```
cluster-genesis$ python -m unittest discover
```

9.4.3 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:

```
cluster-genesis$ tox -e bashate
cluster-genesis$ tox -e flake8
cluster-genesis$ 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:

```
cluster-genesis$ tox -e pycodestyle
cluster-genesis$ tox -e pylint
cluster-genesis$ tox -e pylint-errors
```

9.4.4 File Format Validation

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

```
cluster-genesis$ tox -e file-format
```

9.4.5 Copyright Date Validation

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

```
cluster-genesis$ tox -e verify-copyright
```

CHAPTER 10

Building the Introspection Kernel and Filesystem

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

10.1 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 cluster-genesis/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 'cluster-genesis/os_images/introspection'

10.1.1 Buildroot Config Files

Introspection includes a default buildroot and linux kernel config files.

These files are located in introspection/configs directory under cluster-genesis.

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.

10.2 Run Time

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

10.3 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 11

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. To enable detailed logging of activities, you can set the log level to debug. To enable detailed logging to the logs/gen file, add the -f debug option. To enable detailed display of log information, add -p debug. 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 after a space will list all available options for that level of command input.

The following five top level commands are provided;

- config
- deploy
- post-deploy
- setup
- validate

The deploy command deploys your cluster;

pup 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 you 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 proceding. 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
pup setup --networks
pup config --mgmt-switches
pup validate --cluster-hardware
```

Note that the above steps must initially be run in order. After succesfully 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.

CHAPTER 12

Appendix - C The System Inventory File (needs update)

The inventory.yml file is created by the system genesis process. It can be used by higher level software stacks installation tools to configure their deployment. It is also used to seed the system inventory information into the operations management environment.

12.1 inventory.yml File format:

—

userid-default: joedefault # default userid if no other userid is specified

password-default: joedefaultpassword

redundant-network: 0 # indicates whether the data network is redundant or not

ipaddr-mgmt-network: 192.168.16.0/20 #ipv4 address /20 provides 4096 addresses

ipaddr-mgmt-switch:

-rack1: 192.168.16.2 #ipv4 address of the management switch in the first rack or cell.

-rack2: 192.168.16.3

-rack3: 192.168.16.4

-rack4: 192.168.16.5

-rack5: 192.168.16.6

-aggregation: 192.168.16.18

userid-mgmt-switch: joemgmt # if not specified, the userid-default will be used

password-mgmt-switch: joemgmtpassword # if not specified, the password-default will be used.

ipaddr-data-switch:

-rack1: 192.168.16.20 # if redundant-network is set to 1, genesis will look for an additional switch at the next sequential address.

-rack2: 192.168.16.25

-rack3: 192.168.16.30

-rack4: 192.168.16.35

-rack5: 192.168.16.40

-spine: 192.168.16.45

userid-data-switch: joedata # if not specified, the userid-default will be used

password-data-switch: joedatapassword # if not specified, the password-default will be used.

userid-ipmi-new: userid

password-ipmi-new: password

Base Network information

openstack-mgmt-network:

addr: 172.29.236.0/22 #ipv4 openstack management network

vlan: 10

eth-port: eth10

openstack-stg-network:

addr: 172.29.244.0/22 #ipv4 openstack storage network

vlan: 20

eth-port: eth10

openstack-tenant-network:

addr: 172.29.240.0/22 #ipv4 openstack tenant network

vlan: 30 # vxlan vlan id

eth-port: eth11

ceph-replication-network:

addr: 172.29.248.0/22 # ipv4 ceph replication network

vlan: 40

eth-port: eth11

swift-replication-network:

addr: 172.29.252.0/22 # ipv4 ceph replication network

vlan: 50

eth-port: eth11

########## OpenStack Controller Node Section ################

userid-ipmi-ctrlr: userid

password-ipmi-ctrlr: password

hostname-ctrlr:

name-10G-ports-ctrlr:

-ifc1: [ifcname1, ifcname2] # 2nd ifcname is optional. Multiple ports are bonded.

-ifc2: [ifcname1, ifcname2]

list-ctrlr-ipmi-ports:

-rack1: [port1, port2, port3]

-rack2: [port1]

########## Compute Node Section ##############################

userid-ipmi-compute: userid

password-ipmi-compute: password

hostname-compute:

name-10G-ports-compute:

-ifc1: [ifcname1, ifcname2] # 2nd ifcname is optional. Multiple ports are bonded.

-ifc2: [ifcname1, ifcname2]

list-compute-ipmi-ports:

-rack1: [port1, port2, port3, port4]

-rack2: [port1, port2, port3, port4, port5]

-rack3: [port1, port2, port3, port4, port5]

-rack4: [port1, port2, port3, port4, port5]

-rack5: [port1, port2, port3, port4, port5]

########## Ceph OSD Node Section ############################

userid-ipmi-ceph-osd: userid

password-ipmi-ceph-osd: password

hostname-ceph-osd:

name-10G-ports-ceph-osd:

-ifc1: [ifcname1, ifcname2] # 2nd ifcname is optional. Multiple ports are bonded.

-ifc2: [ifcname1, ifcname2]

list-ceph-osd-ipmi-ports:

-rack1: [port1, port2, port3]

-rack2: [port1, port2, port3]

-rack3: [port1]

-rack4: [port1]

-rack5: [port1]

########## Swift Storage Node Section ######################

userid-ipmi-swift-stg: userid

password-ipmi-swift-stg: password

hostname-swift-stg:

name-10G-ports-swift-stg:

-ifc1: [ifcname1, ifcname2] # 2nd ifcname is optional. Multiple ports are bonded.

-ifc2: [ifcname1, ifcname2]

list-swift-stg-ipmi-ports:

-rack1: [port2, port3, port4]

-rack2: [port2, port3, port4]

-rack3: [port1, port2]

-rack4: [port1]

-rack5: [port1]

. . .

—

hardware-mgmt-network: 192.168.0.0/20 # 4096 addresses

ip-base-addr-mgmt-switches: 2 # 20 contiguous ip addresses will be reserved

ip-base-addr-data-switches: 21 # 160 contiguous ip addresses will be reserved

redundant-network: 1

dns:

- dns1-ipv4: address1
- dns2-ipv4: address2

userid-default: user

password-default: passw0rd

userid-mgmt-switch: user # applied to all mgmt switches

password-mgmt-switch: passw0rd # applied to all mgmt switches

userid-data-switch: user

password-data-switch: passw0rd

ssh-public-key: # key used for access to all node types

ssh-passphrase: passphrase

openstack-mgmt-network:

addr: 172.29.236.0/22 #ipv4 openstack management network

vlan: 10

eth-port: eth10

openstack-stg-network:

addr: 172.29.244.0/22 #ipv4 openstack storage network

vlan: 20

eth-port: eth10

openstack-tenant-network:

addr: 172.29.240.0/22 #ipv4 openstack tenant network

vlan: 30 # vxlan vlan id

eth-port: eth11

ceph-replication-network:

addr: 172.29.248.0/22 # ipv4 ceph replication network

vlan: 40

eth-port: eth11

swift-replication-network:

addr: 172.29.252.0/22 # ipv4 ceph replication network

vlan: 50

eth-port: eth11

racks:

- rack-id: rack number or name

data-center: data center name

room: room id or name

row: row id or name

- rack-id: rack number or name

data-center: data center name

room: room id or name

row: row id or name

switches:

mgmt:

- hostname: Device hostname

ipv4-addr: ipv4 address of the management port

userid: Linux user id for this controller

password: Linux password for this controller

rack-id: rack name or number

rack-eia: rack eia location

model: model # for this switch

serial-number: Serial number for this switch

- hostname: Device hostname

ipv4-addr: ipv4 address of the management port

userid: Linux user id for this controller

password: Linux password for this controller

rack-id: rack name or number

rack-eia: rack eia location

model: model # for this switch

serial-number: Serial number for this switch

leaf:

- hostname: Device hostname

ipv4-addr: ipv4 address of the management port

userid: Linux user id for this controller

password: Linux password for this controller

rack-id: rack name or number

rack-eia: rack eia location

model: model # for this switch

serial-number: Serial number for this switch

- hostname: Device hostname

ipv4-addr: ipv4 address of the management port

userid: Linux user id for this controller

password: Linux password for this controller

rack-id: rack name or number

rack-eia: rack eia location

model: model # for this switch

serial-number: Serial number for this switch

spine:

- hostname: Device hostname

ipv4-addr: ipv4 address of the management port

userid: Linux user id for this controller

password: Linux password for this controller

rack-id: rack name or number

rack-eia: rack eia location

model: model # for this switch

serial-number: Serial number for this switch

- hostname: Device hostname

ipv4-addr: ipv4 address of the management port

userid: Linux user id for this controller

password: Linux password for this controller

rack-id: rack name or number

rack-eia: rack eia location

model: model # for this switch

serial-number: Serial number for this switch

nodes:

controllers: # OpenStack controller nodes

- hostname: hostname #(associated with ipv4-addr below)

ipv4-addr: ipv4 address of this host # on the eth10 interface

userid: Linux user id for this controller

cobbler-profile: name of cobbler profile

rack-id: rack name or number

rack-eia: rack eia location

chassis-part-number: part number # ipmi field value

chassis-serial-number: Serial number # ipmi field value

model: system model number # ipmi field value

serial-number: system serial number # ipmi field value

ipv4-ipmi: ipv4 address of the ipmi port

mac-ipmi: mac address of the ipmi port

userid-ipmi: userid for logging into the ipmi port

password-ipmi: password for logging into the ipmi port

userid-pxe: userid for logging into the pxe port

password-pxe: password for logging into the pxe port

ipv4-pxe: ipv4 address of the ipmi port

mac-pxe: mac address of the ipmi port

openstack-mgmt-addr: 172.29.236.2/22

openstack-stg-addr: 172.29.244.2/22

openstack-tenant-addr: 172.29.240.2/22

- hostname: Linux hostname

ipv4-addr: ipv4 address of this host # on the eth10 interface

userid: Linux user id for this controller

cobbler-profile: name of cobbler profile

rack-id: rack name or number

rack-eia: rack eia location

chassis-part-number: part number # ipmi field value

chassis-serial-number: Serial number # ipmi field value

model: system model number # ipmi field value

serial-number: system serial number # ipmi field value

ipv4-ipmi: ipv4 address of the ipmi port

mac-ipmi: mac address of the ipmi port

userid-ipmi: userid for logging into the ipmi port

password-ipmi: password for logging into the ipmi port
userid-pxe: userid for logging into the pxe port
password-pxe: password for logging into the pxe port
ipv4-pxe: ipv4 address of the ipmi port
mac-pxe: mac address of the ipmi port
openstack-mgmt-addr: 172.29.236.3/22 #ipv4 mgmt network
openstack-stg-addr: 172.29.244.3/22 #ipv4 storage network
openstack-tenant-addr: 172.29.240.3/22 #ipv4 tenant network
compute: # OpenStack compute nodes

- hostname: Linux hostname

ipv4-addr: ipv4 address of this host # on the eth11 port???
userid: Linux user id for this controller
cobbler-profile: name of cobbler profile
rack-id: rack name or number
rack-eia: rack eia location
chassis-part-number: part number # ipmi field value
chassis-serial-number: Serial number # ipmi field value
model: system model number # ipmi field value
serial-number: system serial number # ipmi field value
ipv4-ipmi: ipv4 address of the ipmi port
mac-ipmi: mac address of the ipmi port
userid-ipmi: userid for logging into the ipmi port
password-ipmi: password for logging into the ipmi port
userid-pxe: userid for logging into the pxe port
password-pxe: password for logging into the pxe port
ipv4-pxe: ipv4 address of the ipmi port
mac-pxe: mac address of the ipmi port
openstack-mgmt-addr: 172.29.236.0/22 #ipv4 management network
openstack-stg-addr: 172.29.244.0/22 #ipv4 storage network
openstack-tenant-addr: 172.29.240.0/22 #ipv4 tenant network

- hostname: Linux hostname

ipv4-addr: ipv4 address of this host # on the eth11 port???
userid: Linux user id for this controller
cobbler-profile: name of cobbler profile
rack-id: rack name or number
rack-eia: rack eia location

chassis-part-number: part number # ipmi field value
chassis-serial-number: Serial number # ipmi field value
model: system model number # ipmi field value
serial-number: system serial number # ipmi field value
ipv4-ipmi: ipv4 address of the ipmi port
mac-ipmi: mac address of the ipmi port
userid-ipmi: userid for logging into the ipmi port
password-ipmi: password for logging into the ipmi port
userid-pxe: userid for logging into the pxe port
password-pxe: password for logging into the pxe port
ipv4-pxe: ipv4 address of the ipmi port
mac-pxe: mac address of the ipmi port
openstack-mgmt-addr: 172.29.236.0/22 #ipv4 management network
openstack-stg-addr: 172.29.244.0/22 #ipv4 storage network
openstack-tenant-addr: 172.29.240.0/22 #ipv4 tenant network
ceph-osd:

- hostname: nameabc #Linux hostname

ipv4-addr: ipv4 address of this host # on the eth10 interface
userid: Linux user id for this controller
cobbler-profile: name of cobbler profile
rack-id: rack name or number
rack-eia: rack eia location
chassis-part-number: part number # ipmi field value
chassis-serial-number: Serial number # ipmi field value
model: system model number # ipmi field value
serial-number: system serial number # ipmi field value
ipv4-ipmi: ipv4 address of the ipmi port
mac-ipmi: mac address of the ipmi port
userid-ipmi: userid for logging into the ipmi port
password-ipmi: password for logging into the ipmi port
userid-pxe: userid for logging into the pxe port
password-pxe: password for logging into the pxe port
ipv4-pxe: ipv4 address of the ipmi port
mac-pxe: mac address of the ipmi port
openstack-stg-addr: 172.29.244.0/22 #ipv4 storage network
ceph-replication-addr: 172.29.240.0/22 #ipv4 replication network

journal-devices:

- /dev/sdc
- /dev/sdd

osd-devices:

- /dev/sde
- /dev/sdf
- /dev/sdg
- /dev/sdh
- hostname: nameabc

ipv4-addr: ipv4 address of this host # on the eth11 port???

userid: Linux user id for this controller

cobbler-profile: name of cobbler profile

rack-id: rack name or number

rack-eia: rack eia location

chassis-part-number: part number # ipmi field value

chassis-serial-number: Serial number # ipmi field value

model: system model number # ipmi field value

serial-number: system serial number # ipmi field value

ipv4-ipmi: ipv4 address of the ipmi port

mac-ipmi: mac address of the ipmi port

userid-ipmi: userid for logging into the ipmi port

password-ipmi: password for logging into the ipmi port

userid-pxe: userid for logging into the pxe port

password-pxe: password for logging into the pxe port

ipv4-pxe: ipv4 address of the ipmi port

mac-pxe: mac address of the ipmi port

openstack-stg-addr: 172.29.244.0/22 #ipv4 storage network

ceph-replication-addr: 172.29.240.0/22 #ipv4 replication network

journal-devices:

- /dev/sdc
- /dev/sdd

osd-devices:

- /dev/sde
- /dev/sdf
- /dev/sdg
- /dev/sdh

swift-storage:

- hostname: Linux hostname

ipv4-addr: ipv4 address of this host # on the eth11 port???

userid: Linux user id for this controller

cobbler-profile: name of cobbler profile

rack-id: rack name or number

rack-eia: rack eia location

chassis-part-number: part number # ipmi field value

chassis-serial-number: Serial number # ipmi field value

model: system model number # ipmi field value

serial-number: system serial number # ipmi field value

ipv4-ipmi: ipv4 address of the ipmi port

mac-ipmi: mac address of the ipmi port

userid-ipmi: userid for logging into the ipmi port

password-ipmi: password for logging into the ipmi port

userid-pxe: userid for logging into the pxe port

password-pxe: password for logging into the pxe port

ipv4-pxe: ipv4 address of the ipmi port

mac-pxe: mac address of the ipmi port

openstack-mgmt-addr: 172.29.236.0/22 #ipv4 management network

openstack-stg-addr: 172.29.244.0/22 #ipv4 storage network

swift-replication-addr: 172.29.240.0/22 #ipv4 replication network

- hostname: Linux hostname

ipv4-addr: ipv4 address of this host # on the eth11 port???

userid: Linux user id for this controller

cobbler-profile: name of cobbler profile

rack-id: rack name or number

rack-eia: rack eia location

chassis-part-number: part number # ipmi field value

chassis-serial-number: Serial number # ipmi field value

model: system model number # ipmi field value

serial-number: system serial number # ipmi field value

ipv4-ipmi: ipv4 address of the ipmi port

mac-ipmi: mac address of the ipmi port

userid-ipmi: userid for logging into the ipmi port

password-ipmi: password for logging into the ipmi port

userid-pxe: userid for logging into the pxe port

password-pxe: password for logging into the pxe port

ipv4-pxe: ipv4 address of the ipmi port

mac-pxe: mac address of the ipmi port

openstack-mgmt-addr: 172.29.236.0/22 #ipv4 management network

openstack-stg-addr: 172.29.244.0/22 #ipv4 storage network

openstack-tenant-addr: 172.29.240.0/22 #ipv4 tenant network

CHAPTER 13

Appendix - D Example system 1 Simple Flat Cluster

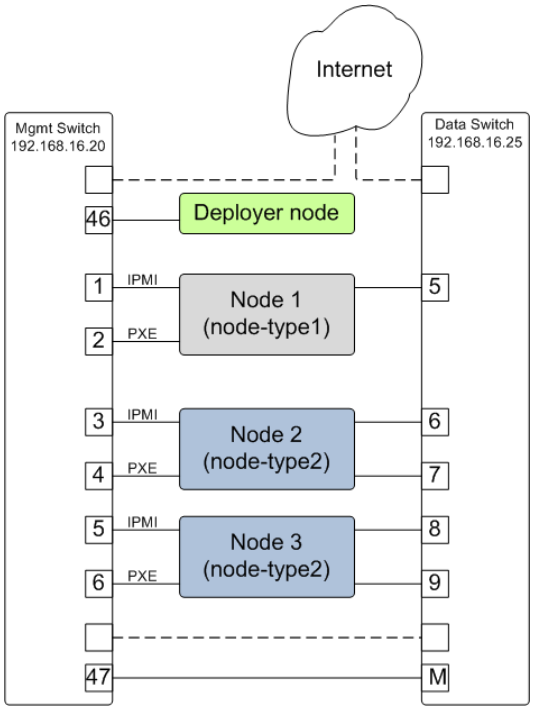

Fig. 1: A simple flat cluster with two node types

A Sample config.yml file

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: 192.168.5.2
              netmask: 255.255.255.0
              vlan: 5
        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: 192.168.5.10
```

(continues on next page)

(continued from previous page)

```
                port: 1
          links:
              - target: deployer
                ports: 2
    # 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: 192.168.5.10
                port: 1
          links:
              - target: deployer
                ports: 2

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

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

node_templates:
    - label: ubuntu1604-node
      ipmi:
          userid: ADMIN
          password: admin
      os:
          profile: ubuntu-16.04-server-ppc64el
          users:
              - name: user1
                password: passw0rd
                groups: sudo
          install_device: /dev/sdj
      physical_interfaces:
          ipmi:
              - switch: mgmt1
```

(continues on next page)

(continued from previous page)

```
            ports:
                - 10
                - 12
                - 14
        pxe:
            - switch: mgmt1
              interface: pxe-ifc
              rename: true
              ports:
                - 11
                - 13
                - 15
        data:
            - switch: data1
              interface: static_1
              rename: true
              ports:
                - 23
                - 25
                - 27
```

CHAPTER 14

Appendix - E Example system 2 - Simple 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. The management switch must be a Lenovo G8052 switch:

```
# This sample configuration file documents all of the supported key values
# supported by the genesis software.  It can be used as the basis for creating
# your own config.yml file.  Note that keywords with a leading underscore
# can be changed by the end user as appropriate for your application. (e.g.
# "_rack1" could be changed to "base-rack")

version: 1.1

ipaddr-mgmt-network: 192.168.16.0/24
ipaddr-mgmt-client-network: 192.168.20.0/24
vlan-mgmt-network: 16
vlan-mgmt-client-network: 20
port-mgmt-network: 19
# Note: The "_rack:" keywords must match the the corresponding rack keyword
# under the keyword;
# node-templates:
#     _node name:
#         ports:
port-mgmt-data-network:
    _rack1:
        - 45
        - 47
ipaddr-mgmt-switch:
    _rack1: 192.168.16.20
cidr-mgmt-switch-external-dev: 10.0.48.3/24
```

(continues on next page)

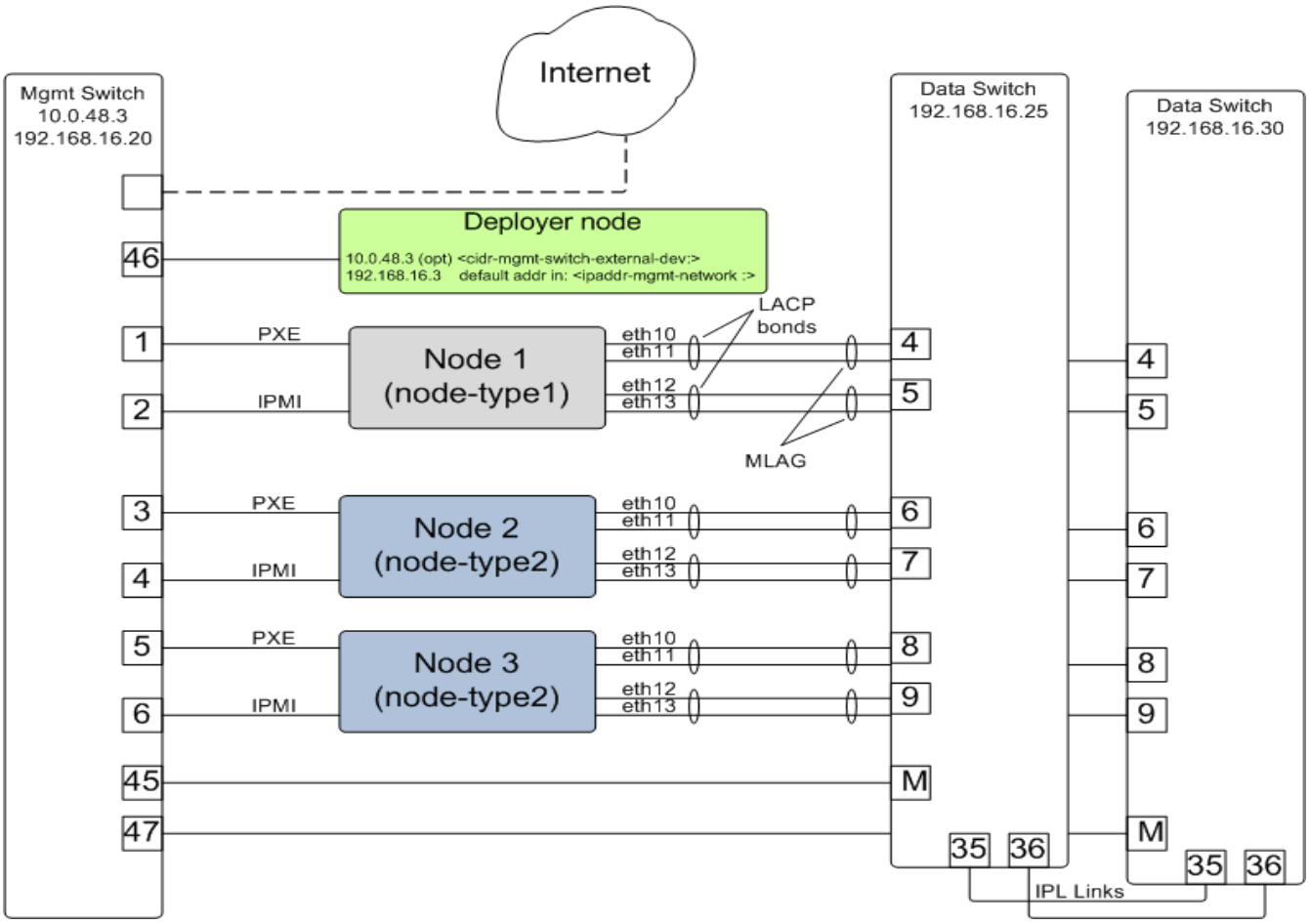

Fig. 1: High Availability Network using MLAG

(continued from previous page)

```
ipaddr-mgmt-switch-external:
    _rack1: 10.0.48.20      # must be present on the switch to start
ipaddr-data-switch: # With MLAG
    _rack1:
        - passmlagdsw1_192.168.16.25
        - passmlagdsw2_192.168.16.30
ipaddr-mlag-vip:
    _rack1: 192.168.16.254
cidr-mlag-ipl:
    _rack1:
        - 10.0.0.1/24
        - 10.0.0.2/24
mlag-vlan:
    _rack1: 4000
mlag-port-channel:
    _rack1: 6
mlag-ipl-ports:
    _rack1:
        -
            - 35
            - 36
        -
            - 35
            - 36
redundant-network: false
userid-default: ubuntu
password-default: passw0rd
userid-mgmt-switch: admin        # applies to all mgmt switches
password-mgmt-switch: admin      # applies to all mgmt switches
userid-data-switch: admin
password-data-switch: admin
networks:
    _external1:
        description: Interface for eth10
        method: manual
        eth-port: eth10
        mtu: 9000
    _external2:
        description: Interface for eth11
        method: manual
        eth-port: eth11
        mtu: 9000
    _external3:
        description: Interface for eth12
        method: manual
        eth-port: eth12
        mtu: 9000
    _external4:
        description: Interface for eth13
        method: manual
        eth-port: eth13
        mtu: 9000
    _pxe-dhcp:
        description: Change pxe port(eth15) to dhcp
        method: dhcp
        eth-port: eth15
    _standalone-bond0:
```

(continues on next page)

(continued from previous page)

```
        description: Multilink bond
        bond: mybond0
        addr: 10.0.16.0/22
        available-ips:
            - 10.0.16.150              # single address
            - 10.0.16.175 10.0.16.215  # address range
        broadcast: 10.0.16.255
        gateway: 10.0.16.1
        dns-nameservers: 10.0.16.200
        # dns-search: mycompany.domain.com
        method: static
        # name of physical interfaces to bond together.
        bond-interfaces:
            - eth10
            - eth11
        mtu: 9000
        # if necessary not all bond modes support a primary slave
        bond-primary: eth10
        # bond-mode, needs to be one of 7 types
        # either name or number can be used.
        # 0 balance-rr
        # 1 active-backup
        # 2 balance-xor
        # 3 broadcast
        # 4 802.3ad
        # 5 balance-tlb
        # 6 balance-alb
        # bond-mode: active-backup
        bond-mode: 4
        # there is a long list of optional bond arguments.
        # Specify them here and they will be added to end of bond definition
        optional-bond-arguments:
            bond-miimon: 100
            bond-lacp-rate: 1
    _standalone-bond1:
        description: bond network to be used by future bridges
        bond: mybond1
        method: manual
        bond-interfaces:
            - eth12
            - eth13
        mtu: 9000
        bond-primary: eth12
        bond-mode: 4
        optional-bond-arguments:
            bond-miimon: 100
            bond-lacp-rate: 1
node-templates:
    node-type1:
        hostname: gandalf
        userid-ipmi: ADMIN
        password-ipmi: admin
        cobbler-profile: ubuntu-16.04.2-server-ppc64el
        os-disk: /dev/sdj
        name-interfaces:
            mac-pxe: eth15    # This keyword is paired to ports: pxe: keyword
            mac-eth10: eth10  # This keyword is paired to ports: eth10: keyword
```

(continues on next page)

(continued from previous page)

```
            mac-eth11: eth11  # This keyword is paired to ports: eth11: keyword
            mac-eth12: eth12  # This keyword is paired to ports: eth12: keyword
            mac-eth13: eth13  # This keyword is paired to ports: eth13: keyword
        # Each host has one network interface for each of these ports and
        # these port numbers represent the switch port number to which the host
        # interface is physically cabled.
        # To add or remove hosts for this node-template you add or remove
        # switch port numbers to these ports.
        ports:
            pxe:
                _rack1:
                    - 1
            ipmi:
                _rack1:
                    - 2
            eth10:          # switch one, 1st bond
                _rack1:
                    - 4     # 1st node
            eth11:          # switch two, 1st bond
                _rack1:
                    - 4
            eth12:          # switch one, 2nd bond
                _rack1:
                    - 5
            eth13:          # switch two, 2nd bond
                _rack1:
                    - 5
        networks:
            - _external1
            - _external2
            - _external3
            - _external4
            - _pxe-dhcp
            - _standalone-bond0
            - _standalone-bond1
    node-type2:
        hostname: radagast
        userid-ipmi: ADMIN
        password-ipmi: admin
        cobbler-profile: ubuntu-16.04.2-server-ppc64el
        os-disk: /dev/sdj
        name-interfaces:
            mac-pxe: eth15
            mac-eth10: eth10
            mac-eth11: eth11
            mac-eth12: eth12
            mac-eth13: eth13
        # Each host has one network interface for each of these ports and
        # these port numbers represent the switch port number to which the host
        # interface is physically cabled.
        # To add or remove hosts for this node-template you add or remove
        # switch port numbers to these ports.
        ports:
            pxe:
                _rack1:
                    - 3
                    - 5
```

(continues on next page)

(continued from previous page)

```
            ipmi:
                _rack1:
                    - 4
                    - 6
            eth10:          # switch one, 1st bond
                _rack1:
                    - 6     # 1st node
                    - 8     # 2nd node
            eth11:          # switch two, 1st bond
                _rack1:
                    - 6
                    - 8
            eth12:          # switch one, 2nd bond
                _rack1:
                    - 7
                    - 9
            eth13:          # switch two, 2nd bond
                _rack1:
                    - 7
                    - 9
        networks:
            - _external1
            - _external2
            - _external3
            - _external4
            - _pxe-dhcp
            - _standalone-bond0
            - _standalone-bond1
```

CHAPTER 15

Appendix - F Detailed Genesis Flow (needs update)

Phase 1:

1. Apply power to the management and data switches.
2. All ports on the management switch will be enabled and added to a single LAN through genesis routines.
3. Power on the compute, storage and controller nodes.
   (a) Each BMC will automatically be assigned an arbitrary IP from the DHCP pool.
4. Genesis code accesses management switch to read MAC address table information. (MAC to port number mapping). This will include both BMC MAC addresses as well as PXE port MAC addresses.
5. Read BMC port list from the config file.
6. Read ip address assignement for BMC ports from the DHCP server
7. IPMI call will be issued to determine whether the BMC represents an x86_64 or PPC64 system.
8. Each BMC will be instructed to initiate a PXE install of a minimal OS, such as CoreOS or similar.
9. Genesis function will access CoreOS and correlate IPMI and PXE MAC addresses using internal IPMI call.
10. Each data network port on the client will be issues an 'UP' and checked for physical connectivity.
11.
12. Cobbler database will be updated. Need more detail.
13. Data switch will be configured.
    (a) VLANS.
14. verification
15. Inventory file will be updated with IPMI, PXE and data port details.
16. IPMI will be used to configure for OS reload and reboot.
17. OS and packages will be installed on the various systems
18. 10 Gb Network ports are renamed

19. Networks are configured on system nodes. There will be a unique config per role. Network configuration consists of modifying the interfaces file template for that role and copying it to the servers.
    - IP addresses
    - VLANS
    - Bridges created
1. Other post OS configuration (NTP)
2. reboot for network config to take effect
3. Deployer container is copied to the first controller node.
4. The inventory file is copied to the first controller node.

Phase 2:

1. Software installation orchestrator is installed on first controller node and given control. Genesis activity continues on first controller node.

CHAPTER 16

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 17

Appendix - H Recovering from Genesis Issues

17.1 Playbook "lxc-create.yml" fails to create lxc container.

- Verify python virtual environment is activated by running *which ansible-playbook*. This should return the path */cluster-genesis/deployenv/bin/ansible-playbook. If something else is returned (including nothing) cd into the cluster-genesis directory and re-run *source scripts/setup-env*.

Verify that the Cluster Genesis network bridges associated with the management and client vlans specified in the config.yml file are up and that there are two interfaces attached to each bridge. One of these interfaces should be a tagged vlan interface associated with the physical port to be used by by Cluster Genesis. The other should be a veth pair attached to the Cluster Genesis container:

```
$ gen status
```

Verify than both bridges have an ip address assigned:

```
ip address show brn  (n whould be the vlan number)
```

17.2 Switch connectivity Issues:

- Verify connectivity from deployer container to management interfaces of both management and data switches. Be sure to use values assigned to the [ipaddr,userid,password]-[mgmt,data]-switch keys in the config.yml. These switches can be on any subnet except the one to be used for your cluster management network, as long as they're accessible to the deployer system.
- Verify SSH is enabled on the data switch and that you can ssh directly from deployer to the switch using the ipaddr,userid, and password keys defined in the config.yml

17.3 Missing Hardware

Hardware can fail to show up for various reasons. Most of the time these are do to miscabling or mistakes in the config.yml file. The Node discovery process starts with discovery of mac addresses and DHCP hand out of ip addresses to the BMC ports of the cluster nodes. This process can be monitored by checking the DHCP lease table after booting the BMCs of the cluster nodes. During execution of the install_1.yml playbook, at the prompt;

"Please reset BMC interfaces to obtain DHCP leases. Press <enter> to continue"

After rebooting the BMCs and before pressing <enter>, you can execute from a second shell:

```
gen status
```

Alternately to see just the leases table, log into the deployer container:

```
$ ssh ~/.ssh/id_rsa_ansible-generated deployer@address
```

The address used above can be read from the 'gen status' display. It is the second address of the subnet specified by the ipaddr-mgmt-network: key in the config.yml file. After logging in:

```
deployer@ubuntu-14-04-deployer:~$ cat /var/lib/misc/dnsmasq.leases
```

*1471870835 a0:42:3f:30:61:cc 192.168.3.173 * 01:a0:42:3f:30:61:cc*

*1471870832 70:e2:84:14:0a:10 192.168.3.153 * 01:70:e2:84:14:0a:10*

*1471870838 a0:42:3f:32:6f:3f 192.168.3.159 * 01:a0:42:3f:32:6f:3f*

*1471870865 a0:42:3f:30:61:fe 192.168.3.172 * 01:a0:42:3f:30:61:fe*

To follow the progress continually you can execute;

deployer@ubuntu-14-04-deployer:~$ tail -f /var/lib/misc/dnsmasq.leases

You can also check what switch ports these mac addresses are connected to by logging into the management switch and executing;

RS G8052>show mac-address-table

- MAC address VLAN Port Trnk State Permanent Openflow*
- ————————— ———— ————- —- ——— ———— ————*
- 00:00:5e:00:01:99 1 48 FWD N *
- 00:16:3e:53:ae:19 1 20 FWD N *
- 0c:c4:7a:76:c8:ec 1 37 FWD N *
- 40:f2:e9:23:82:be 1 11 FWD N *
- 40:f2:e9:24:96:5e 1 1 FWD N *
- 5c:f3:fc:31:05:f0 1 15 FWD N *
- 5c:f3:fc:31:06:2a 1 18 FWD N *
- 5c:f3:fc:31:06:2c 1 17 FWD N *
- 5c:f3:fc:31:06:ec 1 13 FWD N *
- 70:e2:84:14:02:92 1 3 FWD N *

For missing mac addresses, verify that port numbers in the above printout match the ports specified in the config.yml file. Mistakes can be corrected by correcting cabling, correcting the config.yml file and rebooting the BMCs.

Mistakes in the config.yml file require a restart of the deploy process. (ie rerunning gen deploy.) Before doing so remove the existing Genesis container by running the 'tear-down' script and answering yes to the prompt to destroy the container and it's associated bridges.

Depending on the error, it may be possible to rerun the deploy playbooks individually:

```
$ gen install_1
$ gen install_2
```

Alternately, from the cluster-genesis/playbooks directory:

```
$ ansible-playbook -i hosts install_1.yml -K
$ ansible-playbook -i hosts install_2.yml -K
```

Before rerunning the above playbooks, make a backup of any existing inventory.yml files and then create an empty inventory.yml file:

```
$ mv inventory.yml inventory.yml.bak
$ touch inventory.yml
```

Once all the BMC mac addresses have been given leases, press return in the genesis execution window.

17.4 Common Supermicro PXE bootdev Failure

Supermicro servers often fail to boot PXE devices on first try. In order to get the MAC addresses of the PXE ports our code sets the bootdev on all nodes to pxe and initiates a power on. Supermicro servers do ***not* reliably boot pxe (usually will instead choose one of the disks). This *will usually show up as a python key error in the "container/inv_add_pxe_ports.yml" playbook. The only remedy is to retry the PXE boot until it's successful (usually **within* 2-3 tries). To retry use ipmitool from the deployer. The tricky part, however, is determining 1) which systems failed to PXE boot and 2) what the current BMC IP address is. **

To determine which systems have failed to boot, go through the following bullets in this section (starting with "Verify port lists…")

To determine what the corresponding BMC addresss is view the inventory.yml file. At this point the BMC ipv4 and mac address will already be populated in the inventory.yml within the container. To find out:

ubuntu@bloom-deployer: cluster-genesis/playbooks$ grep "^deployer" hosts

deployer ansible_user=deployer ansible_ssh_private_key_file=/home/ubuntu/.ssh/id_rsa_ansible-generated ansible_host=192.168.16.2

ubuntu@bloom-deployer:~/cluster-genesis/playbooks$ ssh -i /home/ubuntu/.ssh/id_rsa_ansible-generated deployer@192.168.16.2

Welcome to Ubuntu 14.04.4 LTS (GNU/Linux 4.2.0-42-generic x86_64)

- * Documentation: https://help.ubuntu.com/*

Last login: Mon Aug 22 12:14:17 2016 from 192.168.16.3

deployer@ubuntu-14-04-deployer:~$ grep -e hostname -e ipmi cluster-genesis/inventory.yml

- – hostname: mgmtswitch1*
- – hostname: dataswitch1*
- – hostname: controller-1*
- userid-ipmi: ADMIN*
- password-ipmi: ADMIN*
- port-ipmi: 29*
- mac-ipmi: 0c:c4:7a:4d:88:26*
- ipv4-ipmi: 192.168.16.101*
- – hostname: controller-2*
- userid-ipmi: ADMIN*
- password-ipmi: ADMIN*
- port-ipmi: 27*
- mac-ipmi: 0c:c4:7a:4d:87:30*
- ipv4-ipmi: 192.168.16.103*

~snip~

Verify port lists within cluster-genesis/config.yml are correct:

~snip~

node-templates:

controller1:

~snip~

- ports:*
- ipmi:*
- rack1:*
- – 9*
- – 11*
- – 13*
- pxe:*
- rack1:*
- – 10*
- – 12*
- – 14*
- eth10:*
- rack1:*
- – 5*
- – 7*
- – 3*

- eth11:*
- rack1:*
- – 6*
- – 8*
- – 4*

~snip~

On the management switch;

RS G8052>show mac-address-table

in the mac address table, look for the missing pxe ports. Also note the mac address for the corresponding BMC port. Use ipmitool to reboot the nodes which have not pxe booted succesfully.

17.5 *Stopping and resuming progress*

In general, to resume progress after a play stops on error (presumably after the error has been understood and corrected!) the failed playbook should be re-run and subsequent plays run as normal. In the case of "cluster-genesis/playbooks/install_1.yml" and "cluster-genesis/playbooks/install_2.yml" around 20 playbooks are included. If one of these playbooks fail then edit the .yml file and and comment plays that have passed by writing a "#" at the front of the line. Be sure *not* to comment out the playbook that failed so that it will re-run. Here's an example of a modified "cluster-genesis/playbooks/install.yml" where the user wishes to resume after a data switch connectivity problem caused the "container/set_data_switch_config.yml" playbook to fail:

- 1 —*
- 2 # Copyright 2018, IBM US, Inc.*
- 3 *

~ 4 #- include: lxc-update.yml

~ 5 #- include: container/cobbler/cobbler_install.yml

~ 6 #- include: pause.yml message="Please reset BMC interfaces to obtain DHCP leases. Press <enter> to continue"

- 7 - include: container/set_data_switch_config.yml log_level=info*
- 8 - include: container/inv_add_switches.yml log_level=info*
- 9 - include: container/inv_add_ipmi_ports.yml log_level=info*
- **10 - include: container/ipmi_set_bootdev.yml log_level=info** bootdev=network persistent=False*
- 11 - include: container/ipmi_power_on.yml log_level=info*
- 12 - include: pause.yml minutes=5 message="Power-on Nodes"*
- 13 - include: container/inv_add_ipmi_data.yml log_level=info*
- 14 - include: container/inv_add_pxe_ports.yml log_level=info*
- 15 - include: container/ipmi_power_off.yml log_level=info*
- 16 - include: container/inv_modify_ipv4.yml log_level=info*
- 17 - include: container/cobbler/cobbler_add_distros.yml*
- 18 - include: container/cobbler/cobbler_add_profiles.yml*
- 19 - include: container/cobbler/cobbler_add_systems.yml*

- 20 - include: container/inv_add_config_file.yml*
- 21 - include: container/allocate_ip_addresses.yml*
- 22 - include: container/get_inv_file.yml dest=/var/oprc*
- **23 - include: container/ipmi_set_bootdev.yml log_level=info** bootdev=network persistent=False*
- 24 - include: container/ipmi_power_on.yml log_level=info*
- 25 - include: pause.yml minutes=5 message="Power-on Nodes"*
- **26 - include: container/ipmi_set_bootdev.yml log_level=info** bootdev=default persistent=True*

17.6 Recovering from Wrong IPMI userid and /or password

If the userid or password for the ipmi ports are wrong, genesis will fail. To fix this, first correct the userid and or password in the config.yml file (~/cluster-genesis/config.yml in both the host OS and the container). Also correct the userid and or password in the container at ~/cluster-genesis/inventory.yml. Then modify the ~/cluster-genesis/playbooks/install.yml file, commenting out the playbooks shown below. Then rerstart genesis from step 15(rerun the install playbook)

—

 # 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.

#- include: lxc-update.yml

#- include: container/cobbler/cobbler_install.yml

- include: pause.yml message="Please reset BMC interfaces to obtain DHCP leases"

#- include: container/set_data_switch_config.yml

#- include: container/inv_add_switches.yml

#- include: container/inv_add_ipmi_ports.yml

- include: container/ipmi_set_bootdev.yml bootdev=network persistent=False

- include: container/ipmi_power_on.yml

- include: pause.yml minutes=20 message="Power-on Nodes"

- include: container/inv_add_ipmi_data.yml

- include: container/inv_add_pxe_ports.yml

- include: container/ipmi_power_off.yml

- include: container/inv_modify_ipv4.yml

- include: container/cobbler/cobbler_add_distros.yml

- include: container/cobbler/cobbler_add_profiles.yml

- include: container/cobbler/cobbler_add_systems.yml

- include: container/inv_add_config_file.yml

- include: container/allocate_ip_addresses.yml

- include: container/get_inv_file.yml dest=/var/oprc

- include: container/ipmi_set_bootdev.yml bootdev=network persistent=False

- include: container/ipmi_power_on.yml

- include: pause.yml minutes=5 message="Power-on Nodes"

- include: container/ipmi_set_bootdev.yml bootdev=default persistent=True

17.7 Recreating the Genesis Container

To destroy the Genesis container and restart Genesis from that point:

```
$ tear-down
```

Respond yes to prompts to destroy the container and remove it's associated bridges. Restart genesis from step 9 of the step by step instructions.

17.8 OpenPOWER Node issues

Specifying the target drive for operating system install;

In the config.yml file, the *os-disk* 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

PXE boot: OpenPOWER nodes need to have the Ethernet port used for PXE booting enabled for DHCP in petitboot.

Be sure to specify a disk configured for boot as the bootOS drive in the config.yml file.

When using IPMI, be sure to specify the right user id and password. IPMI will generate an "unable to initiate IPMI session errors" if the password is not correct.

ipmitool -I lanplus -H 192.168.x.y -U ADMIN -P ADMIN chassis power off

ipmitool -I lanplus -H 192.168.x.y -U ADMIN -P ADMIN chassis bootdev pxe
ipmitool -I lanplus -H 192.168.x.y -U ADMIN -P ADMIN chassis power on

ipmitool -I lanplus -H 192.168.x.y -U ADMIN -P ADMIN chassis power status

To monitor the boot window using the serial over lan capability;

ipmitool -H 192.168.0.107 -I lanplus -U ADMIN -P admin sol activate

Be sure to use the correct password.

You can press Ctrl-D during petit boot to bring up a terminal.

To exit the sol window, enter "~." enter (no quotes)

CHAPTER 18

Appendix - I Using the 'tear-down' Program

The 'tear-down' program allows for select 'tear down' of the Genesis 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 Cluster Genesis environment from the deployer node.

tear-down is completely interactive and only acts when you respond 'y' to prompts.

Usage:

```
tear-down
```

There are currently no arguments or options.

The tear-down program can perform the following functions;

- Backup the config.yml file. Backed up to ~/configbak directory. Config.yml files are date/time stamped.
- Backup the os-images directory
- Remove the Cluster Genesis created management interface from the management switch.
- Remove the Cluster Genesis created bridges from the deployer node.
- Remove the Genesis container. Removes the containers SSH key fom the deployers known_host file.
- Remove the Cluster Genesis software and the directory it is installed in.
- Remove entries made to the .bashrc file and undo changes made to the $PATH environment variable.
- Remove the SSH keys for cluster switches from the deployer known_host file.

For a typical redeploy where the Cluster Genesis software does not need updating, you should remove the cluster genesis container and it's associated bridges. You should also allow removal of all SSH keys from the known_hosts file.

CHAPTER 19

Appendix - J Transferring Deployement Container to New Host

Stil in Development

TODO: general description

19.1 Save Container Files

1. Note container name from LXC status:

```
user@origin-host:~$ sudo lxc-ls -f
```

2. Archive LXC files:

```
user@origin-host:cluster-genesis/scripts $ ./container_save.sh [container_name]
```

3. Save config.yml, inventory.yml, and known_hosts files:

```
origin-host:<cluster-genesis>/config.yml
origin-host:/var/oprc/inventory.yml
origin-host:<cluster-genesis>/playbooks/known_hosts
```

19.2 Prepare New Host

1. Install git
   - Ubuntu:

```
user@new-host:~$ sudo apt-get install git
```

   - RHEL:

```
user@new-host:~$ sudo yum install git
```

2. From your home directory, clone Cluster Genesis:

```
user@new-host:~$ git clone https://github.com/open-power-ref-design-toolkit/
↪cluster-genesis
```

3. Install the remaining software packages used by Cluster Genesis and setup the environment:

```
user@new-host:~$ cd cluster-genesis
user@new-host:~/cluster-genesis$ ./scripts/install.sh

(this will take a few minutes to complete)::

user@new-host:~/cluster-genesis$ source scripts/setup-env

**NOTE:** anytime you leave and restart your shell session, you need to
re-execute the set-env script. Alternately, (recommended) add the following
to your .bashrc file; *PATH=~/cluster-genesis/deployenv/bin:$PATH*

ie::

user@new-host:~$ echo "PATH=~/cluster-genesis/deployenv/bin:\$PATH" >> ~/.bashrc
```

4. Copy config.yml, inventory.yml, and known_hosts files from origin to new host:

```
new-host:<cluster-genesis>/config.yml
new-host:/var/oprc/inventory.yml
new-host:<cluster-genesis>/playbooks/known_hosts
```

5. If needed, modify config.yml and inventory.yml 'port-mgmt-network'. This value represents the port number that the deployer is connected to the management switch.

6. Append cluster-genesis host keys to user's known_hosts:

```
user@new-host:~/cluster-genesis$ cat playbooks/known_hosts >> ~/.ssh/known_hosts

**NOTE:** If user@new-host:~/.ssh/known_hosts already includes keys for
any of these host IP address this action will result in SSH refusing to
connect to the host (with host key checking enabled).
```

7. Make the ~/cluster-genesis/playbooks directory the current working directory:

```
user@new-host:~/cluster-genesis$ cd ~/cluster-genesis/playbooks/
```

8. Setup host networking:

```
user@new-host:~/cluster-genesis/playbooks$ ansible-playbook -i hosts lxc-create.
↪yml -K --extra-vars "networks_only=True"
```

9. Configure management switch:

```
user@new-host:~/cluster-genesis/playbooks$ ansible-playbook -i hosts container/
↪set_mgmt_switch_config.yml
```

19.3 Restore container from archive

1. Copy LXC file archive from origin to new host
2. Run 'container_restore.sh' script to install and start container:

```
user@new-host:cluster-genesis/scripts $ ./container_restore.sh container_archive␣
↪[new_container_name]
```

3. Use LXC status to verify container is running:

```
user@new-host:~$ sudo lxc-ls -f
```

CHAPTER 20

Indices and tables

- genindex
- modindex
- search