Cluster Genesis User Guide Documentation

Release 1.4

Ray Harrington

Nov 02, 2017

Contents:

Version 1.4

Date 2017-09-22

Document Owner OpenPOWER Cluster Genesis Team

Authors Irving Baysah, Rolf Brudeseth, Jay Carman, Ray Harrington, Doug Lehr Hoa Ngo, Nilesh Shah, Jorge Yanez,

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.	

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	TBD
1.1	Castor	2017-02-24	TBD
1.2	Denebola	2017-04-15	TBD
1.3	Electra	2017-06-26	TBD
1.4	Fafnir	2017-09-22	TBD

CHAPTER 3

Introduction

OpenPOWER Cluster Genesis (OPCG) 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 OPCG supports Ethernet networking with separate data and management networks. OPCG can configure simple flat networks for typical HPC environments or more advanced networks with VLANS and bridges for OpenStack environments. OPCG also configures the switches in the cluster. Currently Mellanox SX1410 is supported for the data network and the Lenovo G8052 is supported for the management network.

3.1 Overview

OPCG is designed to be easy to use. If you are implementing one of the supported architectures with supported hardware, OPCG eliminates the need for custom scripts or programming. It does this via a configuration file (config.yml) which drives the cluster configuration. The configuration file is a yaml text file which the user edits. Example YAML files are included. The configuration process is driven from a "deployer" node which does not need to remain in the cluster when finished. The 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.
3. Install the OpenPOWER Cluster Genesis software on the deployer node.
4. Edit an existing config.yml file.
5. Run the OPCG software
6. Power on the cluster compute nodes.

When finished, OPCG generates a YAML formatted inventory file which 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 OpenPOWER Cluster Genesis software supports clusters of servers interconnected with Ethernet. The servers must support IPMI and PXE boot. Currently single racks with single or redundant data switches (with MLAG) are supported. Multiple racks can be interconnected with traditional two tier access-aggregation networking. In the future we plan to support two tier leaf-spine networks with L3 interconnect capable of supporting VXLAN.

3.1.2 Networking

The data network is implemented using the Mellanox SX1410 10 Gb switch. Currently OPCG supports up to four ethernet interfaces. These interfaces can be bonded in pairs with support for LAG or MLAG.

Templates are used to define multiple network configurations in the config.yml file. These can be physical ports, bonded ports, Linux bridges or vLANS. Physical ports can be renamed to ease installation of additional software stack elements.

Management and/or Data switches can be set to "passive" mode to allow deployment without supplying login credentials to the switch management interfaces. This mode requires the user to manually write switch MAC address tables to file and to configure the management and/or data switch in accordance with the defined networks. The client networks will still be configured by Cluster Genesis.

3.1.3 Compute Nodes

OPCG 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
- S822LC
- Tyan servers derived from the above 2 nodes are generally supported.
- SuperMicro OpenPOWER servers

x86 Compute Nodes;

- Lenovo x3550
- Lenovo x3650

Switches

For information on adding additional switch support using Genesis' switch class API, (see *Developer Guide*)

Data Switches;

- Mellanox SX1410
- Mellanox SX1710

Support for Lenovo G8264 is planned

Management Switches;

- Lenovo G8052

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 compute node connections is available and verified. Note that every node to be deployed, must have a BMC and PXE connection to a management switch. (see the example cluster in Appendix-D)
- Cable the deployer node to the cluster management network. It is required that the deployer node be connected directly to the management switch. For large cluster deployments, a 10 Gb connection is recommended. The deployer node must also have access to the public internet (or site) network for accessing software and operating system image files. If the cluster management network does not have external access, an alternate connection with external access must be provided such as the cluster data network, or wireless etc.
- 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 also configured to obtain an ip address via DHCP. On OpenPOWER servers, this is typically done using the Petitboot menus.
- 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 *4 Creating the config.yml File*)

Configuring the Cluster Switches

If your switches are a supported model, Genesis can fully configure them. (See *Supported Hardware* for a list of supported switches.) Even if your switch models are not supported by Cluster Genesis, you can still use Cluster Genesis to deploy and configure your cluster compute nodes. Genesis supports a 'passive' switch mode which enables this. (See : *Preparing for Passive Mode*)

Initial configuration of data switch(es)

For out of box installation, it is usually easiest 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 address specified in the config.yml file (keyname: ipaddr-data-switch:) and be in a *different* subnet than your cluster management subnet used for BMC and PXE communication.*
- assign netmask. This must match the netmask of the subnet the deployer will use 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 file (keyword: password-data-switch:). Note that all data switches in the cluster must have the same userid and password.
- disable spanning tree (typical industry standard commands; *enable, configure terminal, no spanning-tree* or 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 the network template section of the config.yml file. 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 Genesis 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 Genesis 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. The diagram below shows the intitial switch setup and the corresponding config file entries;

In this example, the management switch has an in-band management interface. The initial setup requires an 'externally' accessible address on an in-band interface of all management switches. ('Externally' accessible is used here to mean external to the cluster. ie on the customers' management intranet) Cluster

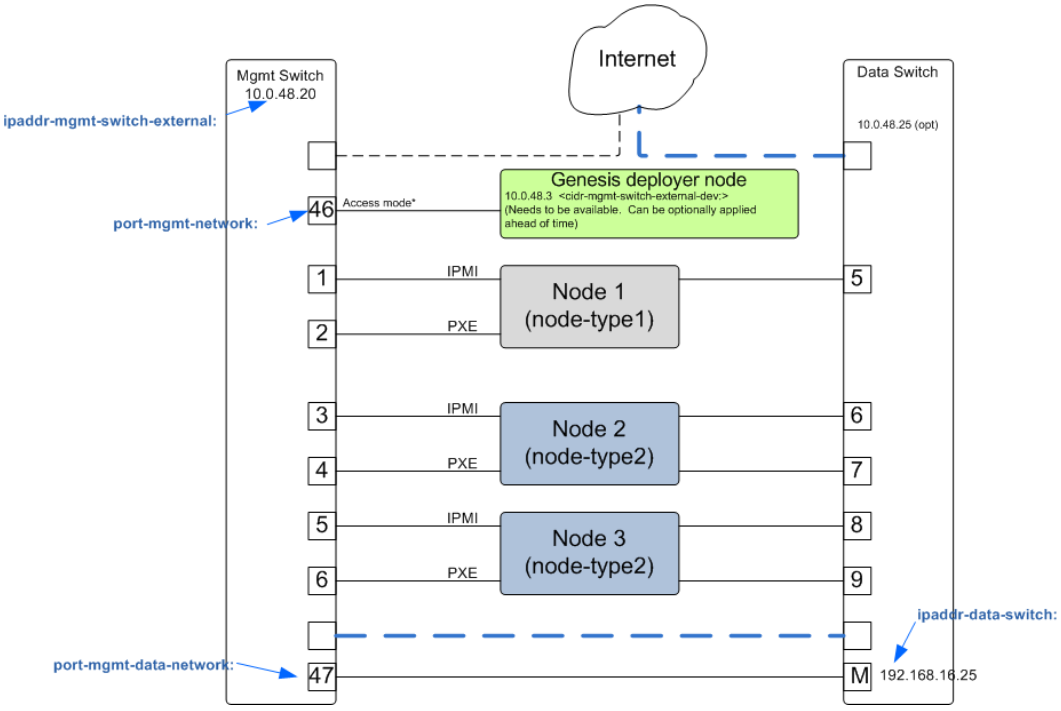

Fig. 4.1: Initial switch setup

genesis uses this address along with the provided userid and password credentials to access the management switch initially. Cluster genesis will create a vlan isolated management network for accessing the management interfaces of the switches in your cluster. A new management interface is created on the management switch in the vlan indicated by the config.yml file. The 'externally' accessible inerface is left unchanged and is available for external monitoring or other purposes. In addition, a vlan is created on the management switches for isolating access to the pxe and BMC interfaces of all node in the cluster.

The following entries in the config.yml file relate to initial switch setup;

- cidr-mgmt-switch-external-dev: 10.0.48.3/20 # example address

  Address on the deployer node for access to the customers external management network. Used by Cluster Genesis for initial management switch access. It is optional to configure this address on an interface on the deployer. If it is not configured, Genesis will configure it temporarily and then remove it when it has finished configuring the management network.

- **ipaddr-mgmt-switch-external:** rack1: 10.0.48.20 # example address

  Address of the management switch on the customers external management network. Used by Cluster Genesis for initial management switch access.

- port-mgmt-network: 46

  Specifies the port on the management switch that the deployer is connected to.

- ipaddr-mgmt-network: 192.168.16.0/24

  Defines the private network that Genesis creates for access to the management interfaces of switches in the cluster. Although the user is free to change this, it is usually not necessary as Genesis will vlan isolate this network so that it will not conflict with existing networks in the customer environment.

- **ipaddr-data-switch:** rack1: 192.168.16.25

Address on the data switch in the private network that genesis creates. Currently the user needs to set up this address on the data switches before running Cluster Genesis. In the future, Genesis will automatically create this address. This address must be within the subnet defined by the ipaddr-mgmt-network: value. Optionally, the customer may also set up a management interface in his external subnet for monitoring or other management purposes.

- **port-mgmt-data-network:** rack1: - 45

  Ports on the management switch which connect to management ports on the data switches.

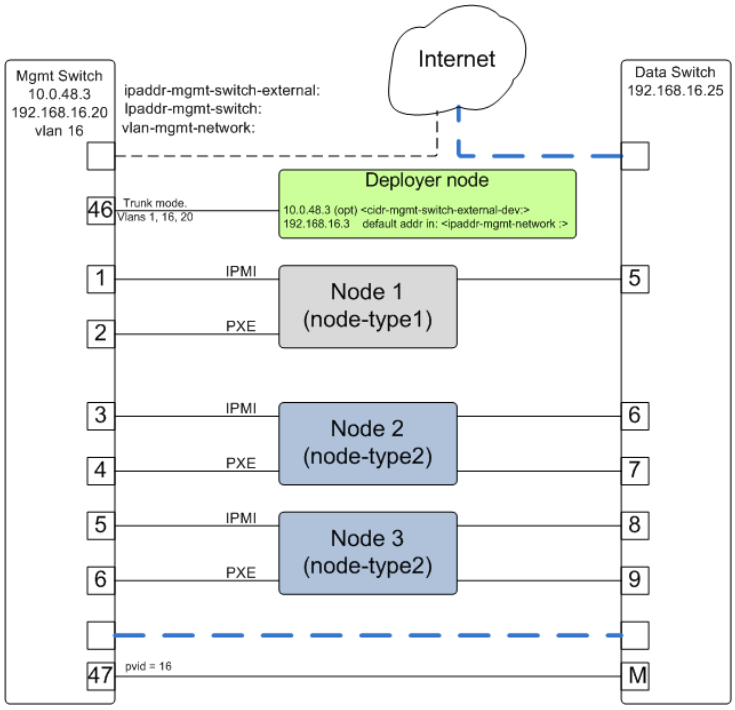

Fig. 4.2: Genesis setup of the switch management network

Management switch setup commands. (for 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 the address specified in the config.yml file (keyname: ipaddr-mgmt-switch-external:) and be in a *different* subnet than your cluster management subnet:

```
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       (User selectable, usually default␣
↪vlan 1 is used)
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 file (keyword: password-mgmt-switch:). Note that all management switches in the cluster must have the same userid and password. The following command is interactive:

```
access user administrator-password
```

- disable spanning tree (for Lenovo switches *enable, configure terminal, spanning-tree mode disable*):

```
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 Genesis initial configuration. **Preparing for Passive Mode**

In passive mode, Genesis configures the cluster compute nodes without requiring any management communication with the cluster switches. This facilitates the use of Genesis 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 Genesis in passive mode, the user is responsible for configuring the cluster switches. The user must also provide the Cluster Genesis software with MAC address tables collected from the cluster switches during the Genesis process. For passive mode, the cluster management switch must be fully programmed before beginning cluster genesis, while the data switch should be configured after Genesis runs.

Configuring the management switch(es)

- The port connected to the deployer node must be put in trunk mode with allowed vlans *vlan-mgmt-network* and *vlan-mgmt-client-network* added. (see *Appendix - B The System Configuration File* for a description of these config file keys)
- The ports on the management switch which connect to the management ports of cluster data switches must be in access mode and have their PVID (Native VLAN) value set to *vlan-mgmt-network*
- The ports on the management switch which connect to cluster node BMC ports or PXE ports must be in access mode and have their PVID (Native VLAN) set to *vlan-mgmt-client-network*

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 Genesis 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

Requirements; 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
    - 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).

ie:

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

Genesis 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.

The following keys are used to provide initial access to the switches in the cluster and must be assigned in the config.yml file

- *ipaddr-mgmt-switch*
- *ipaddr-data-switch*
- *vlan-mgmt-network*
- *ipaddr-mgmt-switch-external*
- *cidr-mgmt-switch-external-dev*
- *port-mgmt-data-network*

For a detailed description of these keys, see *Appendix - B The System Configuration File* and *Genesis setup of the switch management network*.

There are two options for configuring network setup on the deployer. With the first option, Genesis will attempt to discover the deployer port connected to the management switch and configure a temporary address on it for accessing the management switches. For the second option, the user can optionally assign the *label-mgmt-switch-external-dev* key in the config file to skip the auto discovery. In this case, the user must configure the specified port so that it can access the management switches on the 'external' management network.

CHAPTER 5

Creating the config.yml File

The config.yml file drives the creation of the cluster. It uses YAML syntax which is stored as readable text. As config.yml is a Linux file, lines must terminate with a line feed character (/n). If using a windows editor to create or edit the file, be sure to use an editor such as Open Office which supports saving text files with new line characters or use dos2unix to convert the windows text file to linux format.

YAML files support data structures such as lists, dictionaries and scalars. A complete definition of the config.yml file along with detailed documentation of the elements used are given in appendix B.

The config.yml file has 5 main sections. These are;

1. General Settings
2. Cluster definition
3. Network templates
4. Node templates
5. Post Genesis activities

Notes:

- Usually it is easier to start with an existing config.yml file rather than create one from scratch.
- YAML files use spaces as part of syntax. This means for example that elements of the same list must have the exact same number of spaces preceeding them. When editing a .yml file pay careful attention to spaces at the start of lines. Incorrect spacing can result in failure to load messages during genesis.

5.1 General Settings

The top part of the config.yml file contains a group of key value pairs that define general settings. **Config file version**:

```
version: 1.1
```

Release Branch	Supported Config File Version
release-0.9	version: 1.0
release-1.x	version: 1.1
release-2.x	version: 2.0 (planned)

Default log level:

```
log_level: debug
```

Introspection:

Introspection consists of loading a lightweight in-memory OS (linux buildroot) on all client nodes prior to OS installation on disk. This feature can be enabled via the 'introspection-enabled' key in 'config.yml' to a boolean value. If omitted or set to 'false' the introspection components will not be run. Initially it is only supported on clusters with all ppc64el deployer and client nodes.:

```
introspection-enabled: true   # Introspection Mode Enabled
introspection-enabled: false  # Introspection Mode Disabled
```

Write switch configuration to flash memory

The manangement and data switches can automatically write the configuration to flash memory using the 'write-switch-memory' key.:

```
write-switch-memory: true   # Write Switch Memory Enabled
write-switch-memory: false  # Write Switch Memory Disabled
```

Deployment Environment

The 'deployment-environment' key in 'config.yml' can be used to define environment variables (as key: values) to be set during deployment:

```
deployment-environment:
    https_proxy: "http://192.168.1.2:3128"
    http_proxy: "http://192.168.1.2:3128"
    no_proxy: "localhost,127.0.0.1"
```

This was implemented to enable http/https proxy configuration but could be used for anything that utilized environment variables. The 'deployment-environment' dictionary is copied into 'playbooks/group_vars/all' as 'deployment_environment' (note the "-" is changed to "_").

5.2 Cluster definition

The next section of the config.yml file contains a group of key value pairs that define the overall cluster layout. Each rack in a cluster is assumed to have a management switch and one or two data switches. 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")

The following keys must be included in the cluster definition section:

```
ipaddr-mgmt-network: a.b.c.d/n
ipaddr-mgmt-client-network: a.b.e.f/n
vlan-mgmt-network: 16
vlan-mgmt-client-network: 20
port-mgmt-network: 1
ipaddr-mgmt-switch:
    rackname: a.b.c.d
```

```
ipaddr-data-switch:
    rackname: a.b.c.d
redundant-network: false #  "true" for redundant network (future release)
userid-default: joeuser
password-default: passw0rd
userid-mgmt-switch: admin
password-mgmt-switch: admin
userid-data-switch: admin
password-data-switch: admin
```

Notes:

- OpenPOWER Cluster Genesis creates two VLANs on the management switch(es) in your cluster. These are used to isolate access of the management interfaces on the cluster switches from the BMC and PXE ports of the cluster nodes. The VLAN in which the switch management interfaces reside is defined by the vlan-mgmt-network: keyword. The VLAN in which the cluster BMC and PXE ports reside in is defined by the vlan-mgmt-client-network: keyword.
- The ipaddr-mgmt-network: keyword defines the subnet that the PXE and BMC ports for your cluster nodes will reside in. addresses a.b.c.1 and a.b.c.2 are reserved for use by the linux container on the deployer node. Cluster node address assignements will begin at a.b.c.100.
- The ipaddr-mgmt-client-network: keyword defines the subnet that the BMC and PXE ports of the cluster nodes reside in.
- The management ip addresses for the management switch and the data switch must not reside in the same subnet as the nodes management network.
- It is permitted to include addititonal application specific key value pairs at the end of the cluster definition section. Additional keys will be copied to the inventory.yml file which can be read by software stack installation scripts.
- a.b.c.d is used above to represent any ipv4 address. The user must supply a valid ipv4 address. a.b.c.d/n is used to represent any valid ipv4 address in CIDR format.

Passive Switch Mode:

Cluster Genesis can deal with management and/or data switches in "passive" mode to allow deployments without requiring access to the switch management interfaces. This mode requires the user to manually configure the switches and to write switch MAC address tables to files.

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

Passive Management Switch Mode:

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

To configure passive switches simply omit 'userid-mgmt-switch' from 'config.yml'. The 'ipaddr-mgmt-switch' dictionary still needs to be defined in order to be used as a switch identifier. In place of IP addresses anything may be used as long as each switch has a unique value. These unique values will be used by Cluster Genesis to identify the files containing MAC address information.

Passive management switch example configuration:

```
ipaddr-mgmt-switch:
    base-rack: passive_mgmt_1
    rack2: passive_mgmt_2
```

```
    rack3: passive_mgmt_3
ipaddr-data-switch:
    base-rack: passive_data_1
    rack2: passive_data_2
    rack3: passive_data_3
```

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 by Cluster Genesis.

To configure passive switches simply omit 'userid-data-switch' from 'config.yml'. The 'ipaddr-data-switch' dictionary still needs to be defined in order to be used as a switch identifier. In place of IP addresses anything may be used as long as each switch has a unique value. These unique values will be used by Cluster Genesis to identify the files containing MAC address information.

Passive data switch example configuration:

```
ipaddr-mgmt-switch:
    base-rack: 192.168.16.5
    rack2: 192.168.16.6
    rack3: 192.168.16.7
ipaddr-data-switch:
    base-rack: passive1
    rack2: passive2
    rack3: passive3
```

5.3 Network Templates

The network template section of the config.yml file defines the cluster networks. The OpenPower cluster configuration software can configure multiple network interfaces, bridges and vlans on the cluster nodes. vlans setup on cluster nodes will be configured on the data switches also. Network templates are called out in compute templates to create the desired networks on your cluster.

The network template section of the config file begins with the following key:

```
networks:
```

This key is then followed by the name of an individual interface or bridge definitions. Users are free to use any name for a network template. Bridge definitions may optionally include vlans, in which case a virtual vlan port will be added to the specified interface and attached to the bridge. There may be as many network definitions as desired.

5.3.1 Simple static ip address assignement

The following definition shows how to specify a simple static ip address assignement to ethernet port 2:

```
external1: your-ifc-name
   description: Organization site or external network
   addr: a.b.c.d/n
   broadcast: a.b.c.e
   gateway: a.b.c.f
   dns-nameservers: e.f.g.h
   dns-search: your.search.domain
   method: static
   eth-port: eth2
```

Note: Addresses to be assigned to cluster nodes can be entered in the config file as individual addresses or multiple ranges of addresses.

5.3.2 Bridge creation

The following definition shows how to create a bridge with a VLAN attached to the physical port eth2 defined above:

```
mybridge:
   description: my-bridge-name
   bridge: br-mybridge
   method: static
   tcp_segmentation_offload: off
   addr: a.b.c.d/n
   vlan: n
   eth-port: eth2
```

The above definition will cause the creation of a bridge called br-mybridge with a connection to a virtual vlan port eth2.n which is connected to physical port eth2.

5.4 Node Templates

5.4.1 Renaming Interfaces

The *name-interfaces:* 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 as specified. In the example below, the first node has a pxe port cabled to management switch port 1. The genesis code reads the MAC address attached to that port from the management switch and then changes the name of the physical port belonging to that MAC address to the name specified. (in this case "eth15"). Note also that the key pairs under name-interfaces: must correlate to the interfaces names listed under "ports:" ie "mac-pxe" correlates to "pxe" etc.

In the example compute node template below, the node ethernet ports connected to management switch ports 1 and 3 (the pxe ports) will be renamed to eth15, the node ethernet ports connected to management switch ports 5 and 7 (the eth10 ports) will be renamed to eth10:

```
compute:
    hostname: compute
    userid-ipmi: ADMIN
    password-ipmi: ADMIN
    cobbler-profile: ubuntu-14.04.4-server-amd64.sm
    os-disk: /dev/sda
    name-interfaces:
        mac-pxe: eth15
        mac-eth10: eth10
    ports:
        pxe:
            rack1:
                - 1
                - 3
        ipmi:
            rack1:
                - 2
                - 4
```

```
        eth10:
            rack1:
                - 5
                - 7
```

5.4.2 Node Template Definition

The node templates section of the config file starts with the following key:

```
node-templates:
```

Template definitions begin with a user chosen name followed by the key values which define the node:

```
compute:
    hostname: compute
    userid-ipmi: ADMIN
    password-ipmi: ADMIN
    cobbler-profile: ubuntu-14.04.4-server-amd64.sm
    os-disk: /dev/sda
    name-interfaces:
        mac-pxe: eth15
        mac-eth10: eth10
        mac-eth11: eth11
    ports:
        pxe:
            rack1:
                - 1
                - 3
        ipmi:
            rack1:
                - 2
                - 4
        eth10:
            rack1:
                - 5
                - 7
        eth11:
            rack1:
                - 6
                - 8
    networks:
        - external1
        - mybridge
```

Notes:

- The order of ports under the "ports:" dictionary are important and must be in order for each node. In the above example, the first node's pxe, ipmi, eth10 and eth11 ports are connected to the data switch ports 1, 2, 5 and 6.
- 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 insconsistent 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

5.5 Post Genesis Activities

The section of the config.yml file allows you to execute additional commands on your cluster nodes after Genesis completes. These can perform various additional configuration activities or bootstrap additional software package installation. Commands can be specified to run on all cluster nodes or only specific nodes specified by the compute template name.

The following config.yml file entries run the "apt-get update" command on all cluster nodes and then runs the "apt-get upgrade -y" command on the first compute node and runs "apt-get install vlan" on all controller nodes:

```
software-bootstrap:
    all: apt-get update
    compute[0]: |
        apt-get update
        apt-get upgrade -y
    controllers:
        apt-get install vlan
```

CHAPTER 6

OpenPOWER reference design recipes

Many OpenPOWER reference design recipes are available on github. These recipes include bill of materials, system diagrams and config.yml files;

- openstack-recipes
- acclerated-db

OpenPOWER reference designs

CHAPTER 7

Running the OpenPOWER Cluster Configuration Software

7.1 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.

7.2 Passive Switch Mode Special Instructions

Deploying operating systems to your cluster nodes with passive management switches

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

```
$ 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
```

7.3 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 8

Developer Guide

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

8.1 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.

8.2 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.

8.3 Commit Message Rules

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

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

```
test:      Tests change
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
```

8.4 Unit Tests and Linters

8.4.1 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
pep8
ansible-lint
```

Run only 'pep8' test environment:

```
cluster-genesis$ tox -e pep8
```

8.4.2 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
```

8.4.3 Linters

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

- Bash: bashate
- Python: pep8/flake8
- Ansible: ansible-lint

Linters can be run through tox:

```
cluster-genesis$ tox -e bashate
cluster-genesis$ tox -e pep8
cluster-genesis$ tox -e ansible-lint
```

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

8.4.4 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
```

8.5 Mock Inventory Generation

Upon completion, Cluster-Genesis provides an inventory of the cluster (saved locally on the deployer at /var/oprc/inventory.yml). This inventory is used to generate an Ansible dynamic inventory. It can also be consumed by other post-deployment services.

A 'mock' inventory can be generated from any config.yml file. A tox environment is provided to automatically create a python virtual environment with all required dependencies. By default the 'config.yml' file in the cluster-genesis root directory will be used as the input:

```
cluster-genesis$ tox -e mock_inventory

usage: mock_inventory.py [-h] [config_file] [inventory_file]

positional arguments:
  config_file     Input config.yml to process
  inventory_file  Output inventory.yml path

optional arguments:
  -h, --help      show this help message and exit
```

CHAPTER 9

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).

9.1 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'

9.1.1 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.

9.2 Run Time

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

9.3 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 10

Appendix - A Using the 'gen' Program

The 'gen' program is the primary interface to the OpenPOWER Cluster Genesis software. Help can be accessed by typing:

```
gen -h
or
gen --help
```

Usage; gen [–help | -h] <command> [<args>]

Auto completion is enabled for the gen functions.

The gen program provides the following functions;

- log Displays the log file associated with Genesis network setup and container install.
- logc Displays the Genesis container log which logs activities associated with OS deployment and cluster node configuration.
- loga [<-f | +F>] Displays the log file generated by Genesis' ansible playbooks. Normally displays the end of the file. By using the -f or +F options, you or another user can monitor (ie tail) the progress of the Genesis installation from another window.
- status Displays information about the status of the Genesis installation including information about the Genesis container, the bridges Genesis creates in the deployer and information about the state of the DHCP server (including leased addresses) and information about the cobbler program including operating systems deployed and in progress.
- deploy [<-p>] This command runs all of the ansible playbooks necessary to configure the management switch, create the container for Genesis' deploy functions to run in and deploy operating systems to the cluster nodes. When run with the -p option, Genesis will prompt you if you want to continue after each playbook runs. The playbooks run are;
  - setup
  - enable-mgmt-switch
  - lxc-create

  - install_1
  - install_2

  Note that these playbooks can be run individually. ie; gen enable-mgmt-switch. This can be useful when debugging or if you do not have time to complete the entire deploy process for instance.

- deploy-passive [<-p>] This command performs the same functions as the deploy command, but does not access the management switches.
- deploy-passive-retry [<-p>] If deploy-passive fails due to incomplete MAC address table(s) this will reset all servers (power off / set bootdev pxe / power on) to allow the user another chance to collect MAC address tables.
- enable-mgmt-switch Runs the enable-mgmt-switch ansible playbook. Prepares the management switch for use by Genesis.
- lxc-create Runs the lxc-create ansible playbook. Creates the container for Genesis to run in and installs the needed software.
- install_1 Runs the install_1 ansible playbook which performs the first phase of OS deployment. Node discovery and mac address association is performed during this phase. If introspection is enabled, it is run during this step.
- install_2 Runs the install_2 ansible playbook which performs the second phase of OS deployment. Actual OS deployment occurs during this phase.
- config.yml Displays the config.yml file.
- inventory Displays the inventory.yml file created by Cluster Genesis.
- show_mgmt_switches Displays select configuration information of a cluster management switch. Information includes display of management interfaces, configuration of the port connecting to the Genesis deployer node and vlan information for the Genesis VLANs. If multiple management switches are defined, a list of switches is displayed and the user is prompted to select a switch.
- –help or -h Displays help for the gen program
- post-deploy [<-p>] This command runs all of the ansible playbooks which perform post OS deploy activities. These activities include configuration of network interfaces on cluster nodes, copying of SSH keys to cluster nodes, configuring VLANs on data switches and running 'bootstrap' scripts on cluster nodes. When run with the -p option, Genesis will prompt you if you want to continue after each playbook runs. The following playbooks are run;
  - ssh_keyscan
  - gather_mac_addresses
  - set_data_switch_config
  - configure_operating_systems
- ssh_keyscan Runs the ssh_keyscan ansible playbook. Gathers hostkeys from all client nodes and appends the hostkeys to the known_hosts file on each client node.
- gather_mac_addresses Runs the gather_mac_addresses ansible playbook. Gathers mac addresses from data switches for all client node interfaces. Genesis uses this information to accurately rename client node interfaces.
- set_data_switch_config Runs the set_data_switch_config ansible playbook. Configures the cluster data switches including LAG, MLAG and VLANs.
- configure_operating_systems Runs the configure_operating_systems ansible playbook. Configures client node network interfaces, transfers SSH keys to client nodes, copies the inventory.yml file to select cluster nodes and runs bootstrap scripts on specified cluster nodes.
- post-deploy-passive [<-p>] This command performs the same functions as the post-deploy command, but does not access the data switches.

CHAPTER 11

Appendix - B The System Configuration File

Genesis of the OpenPOWER Cloud Reference Config is controlled by the opcr.cfg.yml file. This file is stored in YAML format. The definition of the fields and the YAML file format are documented below.

11.1 config.yml Field Definitions (incomplete)

Keyword	Description	Required/ Optional	Format	Example
cidr-mgmt-switch-external-dev	If the *label-mgmt-switch-external-dev* key is not present in the config file and there is not an existing route to the addresses listed in *ipaddr-mgmt-switch-ext*, Genesis will temporarily configure this address on each 'up' interface on the deployer node in turn looking for one which can communicate with the management switch(es). After Genesis configures the management switch with the address it will use, it will remove this interface adddress.	O		
deployment-environment	Set deployer environment variables to be set during deployment. See *config file deployment environment*.	O	(dictionary)	
introspection-enabled	Enable introspection mode. See *config file introspection*.	O	(boolean)	true
ipaddr-data-switch	This is a list of ipv4 addresses of the management ports of the data switches. This address must be manually configured on the data switches before genesis begins. Users should also plan to allocate one or more additional ip addresses for each pair of data switches. These addresses are used by the switches for inter-switch communication. All of the management interfaces for the management switches and the data switches must reside in one subnet. This subnet must be different than the subnet used for the cluster management network.	R	(string)	192.168.80.36
ipaddr-mgmt-client-network	Cluster node management network address in CIDR format. This is the network that the PXE and BMC ports will reside in. This network will reside in the vlan specified by the vlan-mgmt-client-network key. Note that the management ports of all switches will reside in a different subnet and vlan.	R	(string)	192.168.16.0/20
ipaddr-mgmt-switch	List of IP v4 address to be used for the interface Genesis will create on the management switches in the cluster. These will be in the vlan specified by the vlan-mgmt-network key. The subnet mask to be used on the created interface is defined by the ipaddr-mgmt-network key. Depending on the switch, this interface may be able to exist on the same physical port as the interface on which ipaddr-mgmt-switch-ext is defined.	R	(dictionary)	192.168.16.20
ipaddr-mgmt-switch-ext	List of externally accessible ipv4 addresses of the management interfaces for the management switches in the cluster. Here, externally is used to indicate that these addresses are visible from outside the cluster (ie on the user's intranet) and available for monitoring or other management purposes. These ip addresses must be manually configured on the management switches before genesis begins. The OpenPOWER cluster genesis will look for management switches at the specified addresses. Genesis will create an additional interface on the management switch in the vlan specified by the vlan-mgmt-network key in the config.yml file. Usually, one management switch would be physically located in each rack or with each cell. Note that all of the management interfaces for the management switch and the data switches must reside in one subnet. This subnet must be different than the subnet used for the cluster node network.	R	(dictionary)	10.0.1.2
log_level	Sets the level for Genesis logging. Valid levels are DEBUG, INFO, WARNING, ERROR and CRITICAL. See *config file default log level*. When omitted, log_level defaults to "DEBUG".	O	(string)	DEBUG
label-mgmt-switch-	This is the device name of the physical port on the deployer which connects to the management switch. If included in the config file,	O	(string)	enp1s0f0

11.2 config.yml YAML File format:

```
---
# Copyright 2017 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.
# 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")

# 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: 192.168.16.0/20
ipaddr-mgmt-client-network: 192.168.20.0/24
vlan-mgmt-network: 16
vlan-mgmt-client-network: 20
port-mgmt-network: 1
# NOTE: The "_rack:" keywords must match the the corresponding rack keyword
# under the keyword;
# node-templates:
#     _node name:
#         ports:
port-mgmt-data-network:
    _rack1: 47
ipaddr-mgmt-switch:
    _rack1: 192.168.16.20
ipaddr-data-switch:
    _rack1: 192.168.16.25
redundant-network: false
userid-default: user
password-default: passw0rd
# An encrypted password hash can also be provided using the following format:
# password-default-crypted: $6$STFB8U/AyA$sVhg5a/2RvDiXof9EhADVcUm/7Tq8T4m0dcdHLFZkOr.
↪pCjJr2eH8RS56W7ZUWw6Zsm2sKrkcS4Xc8910JMOw.
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
```

```
# Rack information is optional (not required to be present)
racks:
    - rack-id: rack1
      data-center: dataeast
      room: room33
      row: row1
networks:
    _external1:
        description: Organization site or external network
        addr: 9.3.89.0/24
        available-ips:
            - 9.3.89.14            # single address
            - 9.3.89.18 9.3.89.22  # address range
            - 9.3.89.111 9.3.89.112
            - 9.3.89.120
        broadcast: 9.3.89.255
        gateway: 9.3.89.1
        dns-nameservers: 9.3.1.200
        dns-search: your.dns.com
        method: static
        eth-port: eth10
        mtu: 9000
    _external2:
        description: Interface for eth11
        method: manual
        eth-port: eth11
    _pxe-dhcp:
        description: Change pxe port(eth15) to dhcp
        method: dhcp
        eth-port: eth15
    _standalone-bond0:
        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:
            - eth0
            - eth1
        # 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: 1
        # 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
    _manual-bond1:
        description: bond network to be used by future bridges
        bond: bond1
        method: manual
        bond-mode: balance-rr
        bond-interfaces:
            - eth10
            - eth11
    _cluster-mgmt:
        description: Cluster Management Network
        bridge: br-mgmt
        method: static
        tcp_segmentation_offload: "off"  # on/off values need to be enclosed in quotes
        addr: 172.29.236.0/22
        vlan: 10
        eth-port: eth10
        bridge-port: veth-infra  # add a veth pair to the bridge
    _vm-vxlan-network:
        description: vm vxlan Network
        bridge: br-vxlan
        method: static
        addr: 172.29.240.0/22
        vlan: 30
        eth-port: eth11
    _vm-vlan-network:
        description: vm vlan Network
        bridge: br-vlan
        method: static
        addr: 0.0.0.0/1  # Host nodes do not get IPs assigned in this network
        eth-port: eth11  # No specified vlan.  Allows use with untagged vlan
        bridge-port: veth12
node-templates:
    _node-name:
        hostname: controller
        userid-ipmi: userid
        password-ipmi: password
        cobbler-profile: ubuntu-14.04.4-server-amd64
        os-disk: /dev/sda
        users:
            - name: user1
              groups: sudo
            - name: testuser1
              groups: testgroup
        groups:
            - name: testgroup
        name-interfaces:
            mac-pxe: eth15    # This keyword is paired to ports: pxe: keyword
            mac-eth10: eth10  # This keyword is paired to ports: eth10: keyword
            mac-eth11: eth11  # This keyword is paired to ports: eth11: 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
                    - 2
                    - 3
            ipmi:
                _rack1:
                    - 4
                    - 5
                    - 6
            eth10:
                _rack1:
                    - 1
                    - 2
                    - 3
            eth11:
                _rack1:
                    - 4
                    - 5
                    - 6
        networks:
            - _cluster-mgmt
            - _vm-vxlan-network
            - _vm-vlan-network
            - _external1
            - _external2
            - _pxe-dhcp
            - _manual-bond1
            - _standalone-bond0
    _compute:
        hostname: compute
        userid-ipmi: userid
        password-ipmi: password
        cobbler-profile: ubuntu-14.04.4-server-amd64
        name-interfaces:
            mac-pxe: eth15
            mac-eth10: eth10
            mac-eth11: eth11
        # 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 cabled.
        # To add or remove hosts for this node-template you add or remove
        # switch port numbers to these ports.
        ports:
            pxe:
                _rack1:
                    - 7
                    - 8
                    - 9
            ipmi:
                _rack1:
                    - 10
                    - 11
                    - 12
            eth10:
```

```
                _rack1:
                    - 7
                    - 8
                    - 9
            eth11:
                _rack1:
                    - 10
                    - 11
                    - 12
        networks:
            - _cluster-mgmt
            - _vm-vxlan-network
            - _vm-vlan-network
            - _external1
            - _external2
            - _pxe-dhcp
            - _manual-bond1
            - _standalone-bond0

software-bootstrap:
    all: apt-get update
    compute[0]: |
        apt-get update
        apt-get upgrade -y
# Additional key/value pairs are not processed by Genesis, but are copied into
# the inventory.yml file and made available to post-Genesis scripts and/or
# playbooks.
```

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

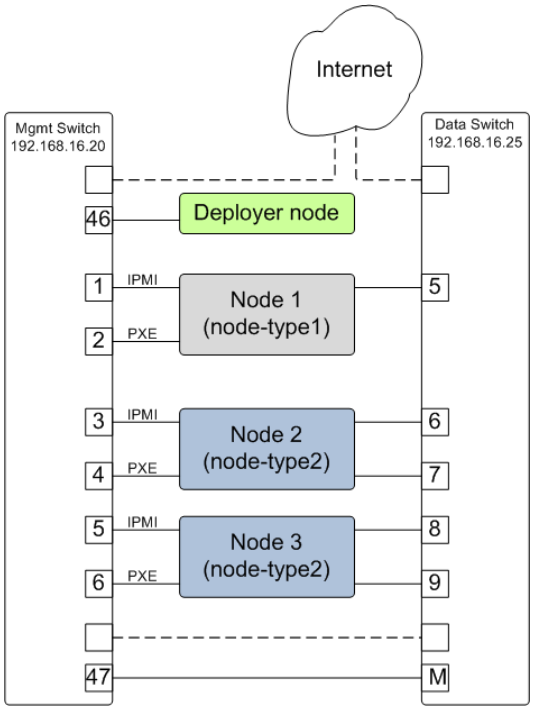

Fig. 13.1: A simple flat cluster with two node types

A Sample config.yml file;

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. Internet access must then be provided via one of the dotted line paths shown in the figure above or alternately via a wireless or dedicated interface.

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: 192.168.16.0/20
ipaddr-mgmt-client-network: 192.168.20.0/24
vlan-mgmt-network: 16
vlan-mgmt-client-network: 20
port-mgmt-network: 46
# NOTE: The "_rack:" keywords must match the the corresponding rack keyword
# under the keyword;
# node-templates:
#     _node name:
#         ports:
port-mgmt-data-network:
    _rack1: 47
ipaddr-mgmt-switch:
    _rack1: 192.168.16.20
ipaddr-data-switch:
    _rack1: 192.168.16.25
redundant-network: false
userid-default: user
password-default: passw0rd
# An encrypted password hash can also be provided using the following format:
# password-default-crypted: $6$STFB8U/AyA$sVhg5a/2RvDiXof9EhADVcUm/7Tq8T4m0dcdHLFZkOr.
↪pCjJr2eH8RS56W7ZUWw6Zsm2sKrkcS4Xc8910JMOw.
userid-mgmt-switch: user        # applies to all mgmt switches
password-mgmt-switch: passw0rd  # applies to all mgmt switches
userid-data-switch: user
password-data-switch: passw0rd
# Rack information is optional (not required to be present)
racks:
    - rack-id: rack1
      data-center: dataeast
      room: room33
      row: row1
networks:
    _external1:
        description: Organization site or external network
        addr: 10.3.89.0/24
        available-ips:
            - 10.3.89.14            # single address
            - 10.3.89.18 10.3.89.22  # address range
            - 10.3.89.111 10.3.89.112
            - 10.3.89.120
        broadcast: 10.3.89.255
        gateway: 10.3.89.1
        dns-nameservers: 8.8.8.8
        dns-search: your.dns.com
        method: static
        eth-port: eth10
        mtu: 9000
```

```
    _external2:
        description: Interface for eth11
        method: manual
        eth-port: eth11
        mtu: 9000
    _pxe-dhcp:
        description: Change pxe port(eth15) to dhcp
        method: dhcp
        eth-port: eth15
    _cluster-bridge:
        description: Cluster Management Network
        bridge: br-clst
        method: static
        tcp_segmentation_offload: "off"  # on/off values need to be enclosed in quotes
        addr: 172.29.236.0/22
        vlan: 10
        eth-port: eth10
        bridge-port: veth-infra  # add a veth pair to the bridge
node-templates:
    _node-type1:
        hostname: charlie
        userid-ipmi: userid
        password-ipmi: password
        cobbler-profile: ubuntu-14.04.4-server-amd64
        os-disk: /dev/sda
        users:
            - name: user1
              groups: sudo
            - name: testuser1
              groups: testgroup
        groups:
            - name: testgroup
        name-interfaces:
            mac-pxe: eth15    # This keyword is paired to ports: pxe: keyword
            mac-eth10: eth10  # This keyword is paired to ports: eth10: keyword
            mac-eth11: eth11  # This keyword is paired to ports: eth11: 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:
                    - 2
            ipmi:
                _rack1:
                    - 1
            eth10:
                _rack1:
                    - 5
        networks:
            - _cluster-mgmt
            - _external1
            - _external2
            - _pxe-dhcp
    _node-type2:
        hostname: compute
```

```
        userid-ipmi: userid
        password-ipmi: password
        cobbler-profile: ubuntu-14.04.4-server-amd64
        name-interfaces:
            mac-pxe: eth15
            mac-eth10: eth10
            mac-eth11: eth11
        # 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 cabled.
        # To add or remove hosts for this node-template you add or remove
        # switch port numbers to these ports.
        ports:
            pxe:
                _rack1:
                    - 4
                    - 6
            ipmi:
                _rack1:
                    - 3
                    - 5
            eth10:
                _rack1:
                    - 6
                    - 8
            eth11:
                _rack1:
                    - 7
                    - 9
        networks:
            - _cluster-mgmt
            - _external1
            - _external2
            - _pxe-dhcp

software-bootstrap:
    all: apt-get update
#   _node-type2[0]: |
#       export GIT_BRANCH=master
#       URL="https://raw.githubusercontent.com/open-power-ref-design/openstack-
↪recipes/${GIT_BRANCH}/scripts/bootstrap-solution.sh"
#       wget ${URL}
#       chmod +x bootstrap-solution.sh
#       ./bootstrap-solution.sh
```

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

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
```

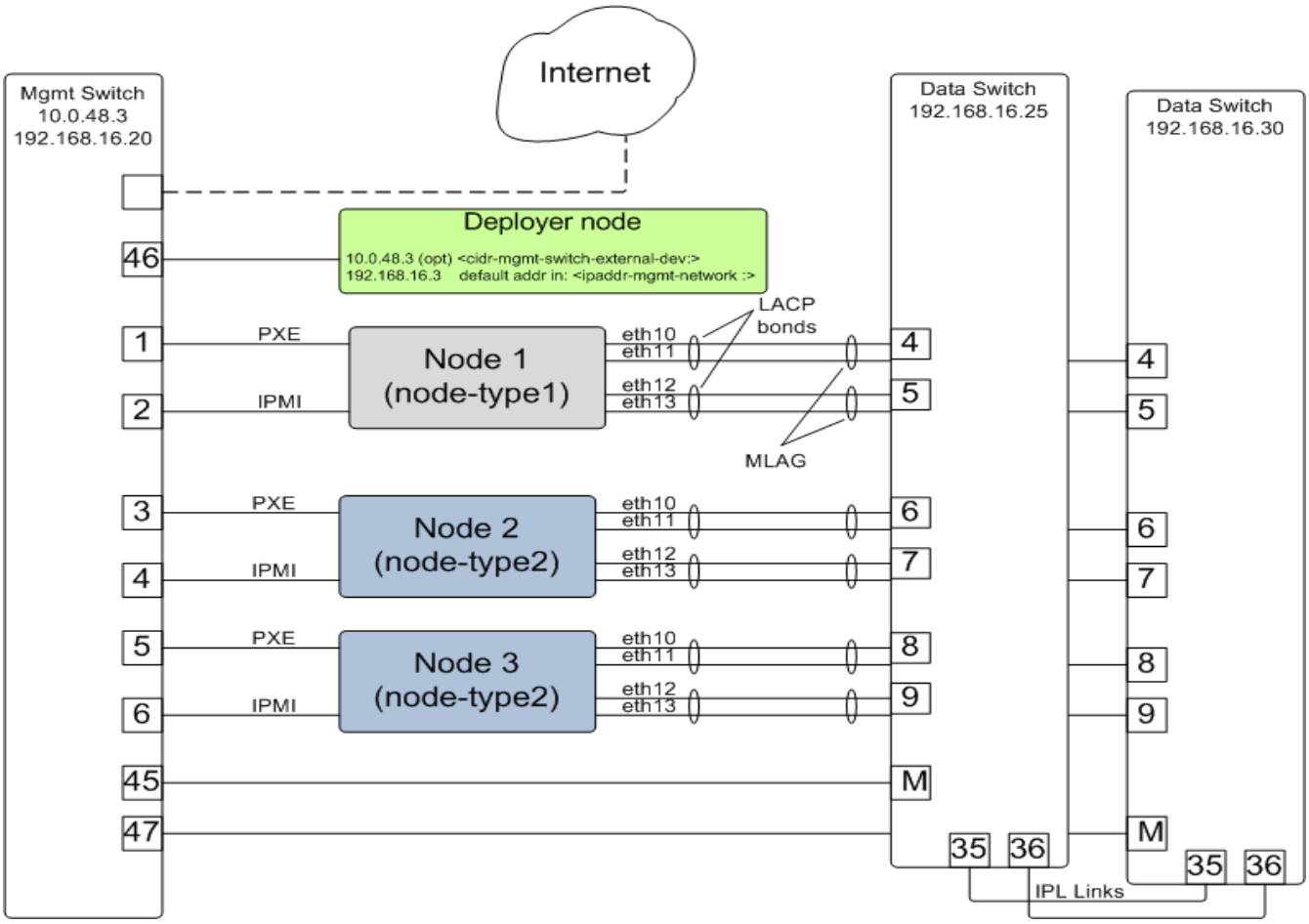

Fig. 14.1: High Availability Network using MLAG

```
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:
        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
            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
            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 2017, 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 2017 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