How to Deploy Open Database as a Service on OpenPOWER

Version 1.6

INTRODUCTION

This document along with referenced links describes a comprehensive set of instructions, rules, and automation tools for building an OpenPOWER-based platform for hosting Open Database as a Service. This platform could host a variety of open source databases, such as MongoDB, Redis, Cassandra, MariaDB, Couchbase, Postgres and MySQL etc. The databases are hosted on an OpenStack based infrastructure, with distinct compute, controller, and block storage and object storage nodes.

The build process is broken out into a series of Steps listed below. Some of these steps are preparatory in nature. The final step is fully automated. At the completion of these steps, an Open Database as a Service cluster should be operational.

HIGH LEVEL DEPLOYMENT STEPS

	EACH STEP BELOW IS DESCRIBED IN MORE DETAIL BELOW
1	Acquire the hardware
2	Choose deployment parameters
3	Prepare the deployment node
4	Rack the servers, switches and JBODs
5	Cable the systems
6	Deploy the cluster

STEP 1: ACQUIRE THE HARDWARE

Go here to view the Design Proposal for recipe required hardware.

https://github.com/open-power-ref-design/dbaas/blob/master/docs/design.pdf

Go here for the Bill of Materials list of required parts:

https://github.com/open-power-ref-design/dbaas/blob/master/docs/bom.pdf

If you do not already have the needed parts, please contact an IBM representative to assist you.

https://www-01.ibm.com/marketing/iwm/dre/signup?source=MAIL-power&disableCookie=Yes

STEP 2: CHOOSE YOUR DEPLOYMENT PARAMETERS

To facilitate faster automated configuration of the overall solution, collect together the following parameters before you start. This data will be edited into a configuration file which will in turn be used to automatically tune the infrastructure and add other needed software.

Parameter	Description	Example
Domain Name	The local domain where the cluster is being built.	mycompany.domain.com
Upstream DNS Servers	While a DNS server is configured within the cluster, upstream DNS servers need to be defined for names that cannot otherwise be resolved.	*4.4.4.4, 8.8.8.8 as default public upstream DNS servers
Deployment Node Host Name	What do you want to call the deployment node?	depnode
Management network IP address range	Management for the cluster happens on its own internal network. Labeled *ipaddr-mgmt-network* in the config.yml example below.	192.168.16.0/24

Data network IP address range	Private data network internal to the cluster. Labeled *external1* in the config.yml example below.	10.0.16.0/22
Management switch IP address	IP address of the management switch. Labeled *ipaddr-mgmt-switch* in config.yml in example below	192.168.16.20
Data switch IP address	IP address of the data switch on the rack. Labeled *ipaddr-data-switch* in config.yml in example below	192.168.16.25
External floating IP address	Floating ip-address that can be used to view the cluster GUI (Horizon) externally	10.0.16.50
Default login data	Both IDs and passwords	BMC network, OS Mgmt. network
Client-OS-Level	What operating system to install on the nodes	ubuntu-16.04.1-server-ppc64el

Go here to see more options in the config.yml file.
https://github.com/open-power-ref-design/dbaas/blob/master/config.yml

An alternate starter 3-node configuration yml file also exists and is published here.
https://github.com/open-power-ref-design/dbaas/blob/master/config-starter.yml

STEP 3: PREPARE THE DEPLOYER NODE

The deployer node is used to obtain the latest software and deployment tools from GitHub and populate the cluster. The deployer node is not included in the Bill of Materials (BOM). You can establish the deployer node as a temporary or a permanent server. It can be any Power8-LC or x86 server with the following minimum characteristics:

- 2 cores and 32G RAM
- 1 Network Interface connection: 1G (Mgmt.).
- Ubuntu 16.04 LTS.

If you do not already have Ubuntu installed on the deployer node, you can obtain it from the following locations:

- Power8-LC servers: https://www.ubuntu.com/download/server/power8
- For x86 servers: http://releases.ubuntu.com/16.04.1/

STEP 4: RACK THE SERVERS, SWITCHES AND STORAGE

There are various compute, storage and networking components in an Open Database as a Service solution. Optimal server resources have been selected to attain the right balance between compute, storage and network I/O.

The four classes of servers in an Open Database as a Service solution are:

Compute Nodes: These servers host the database instances. The database instances are hosted in Virtual Machines. This solution has two Stratton Power S812LC MTM 8001-12C nodes that serve this role. These nodes are referenced in the rest of the document with a prefix of 'Comp'. For example Comp-1 is the first compute node.

Controller Nodes: These servers host the OpenStack and OpsMgr Control services. The design is triple redundant and allows for scaling of the cluster to larger (intra-rack) configurations. This solution has three Briggs Power S822LC MTM 8001-22C nodes serving this role. These nodes are referenced in the rest of the document with a prefix of 'Ctrl'. For example Ctrl-1 is the first controller node.

Ceph OSD Nodes: These servers with their built in storage provide block storage for the Virtual Machines hosted on the Compute Nodes. This solution has three Briggs Power S822LC MTM 8001-22C nodes with additional hard-drives and SSDs to serve this role. These nodes are referenced in the rest of the document with a prefix of 'Ceph'. For example Ceph-1 is the first Ceph OSD node.

Swift Object Nodes: These servers together with their attached JBOD (just a bunch of disks) storage drawers provide Object storage services. This solution has three Stratton Power S812LC MTM 8001-12C nodes with attached IBM 5U84 JBOD disk drawers serving this role. These nodes are referenced in the rest of the document with a prefix of 'Swift'. For example Swift-1 is the first Swift Object node.

Racking the components

This version of the solution supports redundant data and non-redundant management switches.

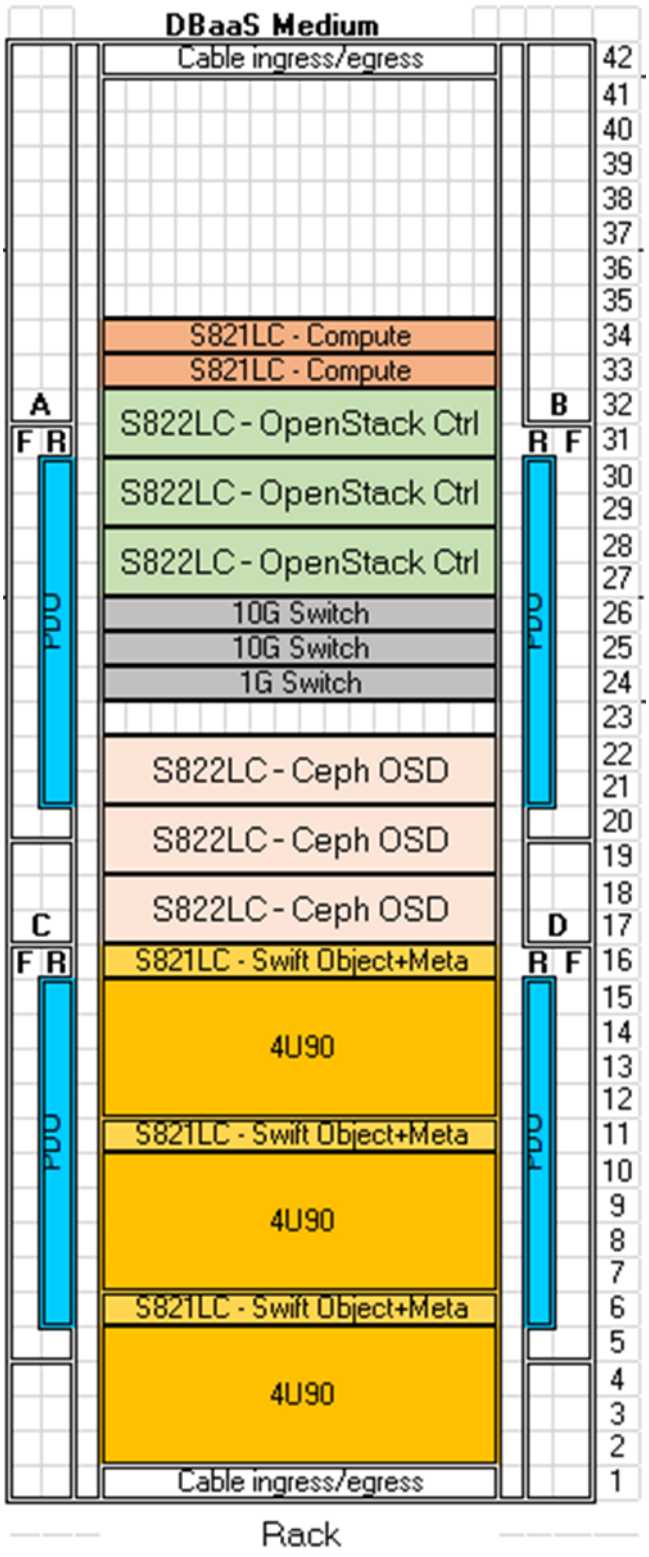
Place the intra-rack (leaf) network switches in **U26**-**U29** as follows:

- 10G data plane switches in **U25 and U26** (part 8831-S48)
- 1G management plane switch in **U24**. (part 7120-48E)

Place the Swift nodes with the attached Storage drawers at the bottom of the rack. Place the Ceph nodes above the Swift nodes. Place the OpenStack Controller nodes above the switches. Place the Compute nodes at the top of the Controller nodes, leaving some room for expansion (i.e. additional compute nodes).

If more than 4 PDUs are needed, place 2 horizontal PDUs in 40U and 41U

RESULTING EXAMPLE: 11 NODE OPEN DBAAS CLUSTER



Racking the components

This version of the solution is a low cost option for trying the cluster out before a larger investment. It uses single switches for non-redundant networking. Redundancy on the data switches can be added at a later time.

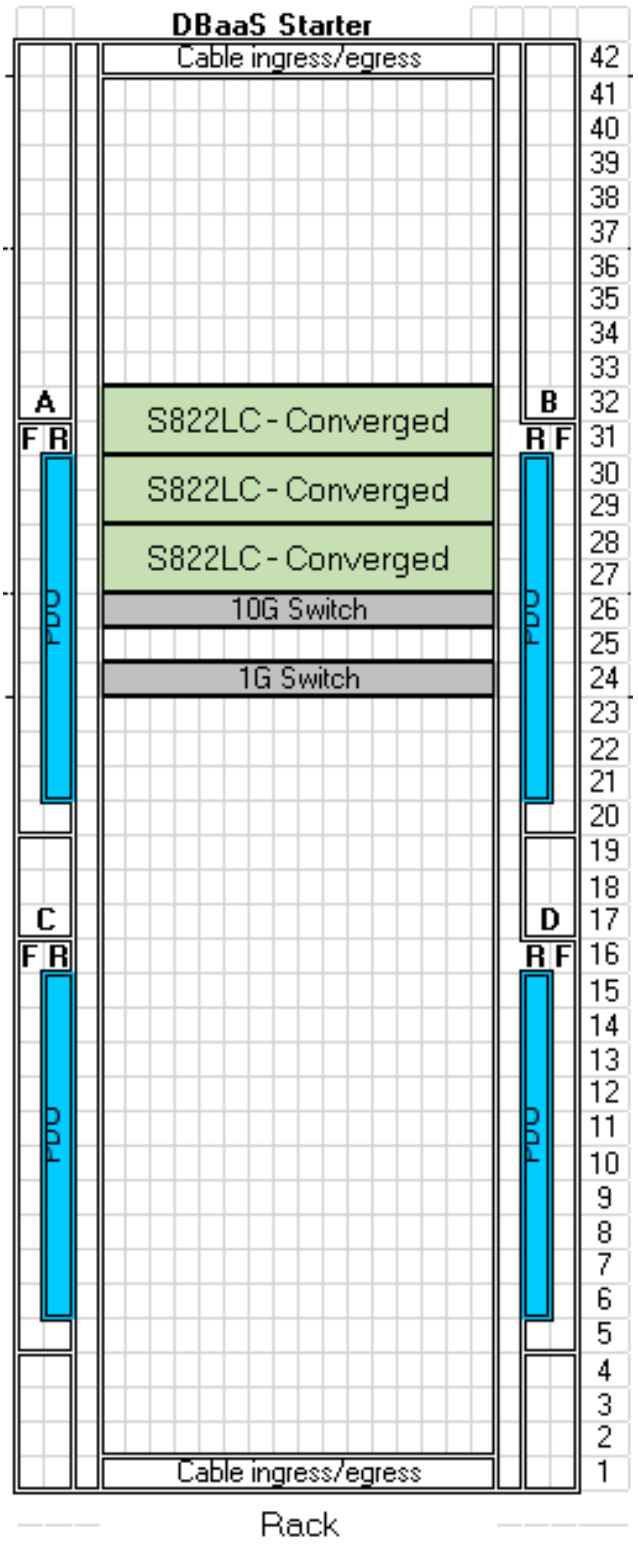
Place the intra-rack (leaf) network switches in **U26**-**U29** as follows:

- 10G data plane switch in **U26** (part 8831-S48)
- 1G management plane switch in **U24**.(part 7120-48E)

Place the Converged nodes above the switches. These nodes will serve in control, compute and storage roles.

If more than 4 PDUs are needed, place 2 horizontal PDUs in 40U and 41U

RESULTING EXAMPLE: 3 NODE OPEN DBAAS STARTER CLUSTER



STEP 5: CABLE THE SYSTEMS

Cabling of an Open Database as a Service solution consists of these distinct efforts. One is the storage cabling, another is the network cabling, and finally powering the systems and switches. Presented first is the storage cabling.

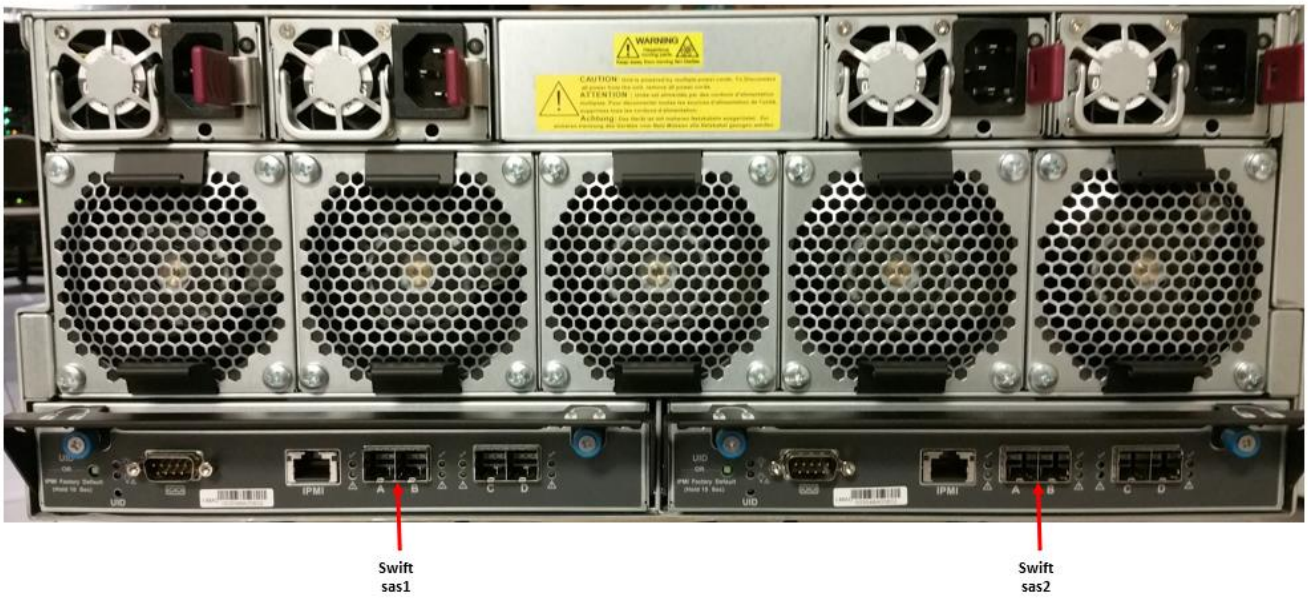
CABLE THE STORAGE DRAWERS

In this Step connect the Swift nodes to the storage drawers, using the SAS cables. Repeat this step for all the Swift Object Server nodes.

EXAMPLE: SWIFT OBJECT NODE

Rear view of Swift Object node connections with labels.

Rear view of Swift Storage drawer with labels.



CABLE THE NETWORK

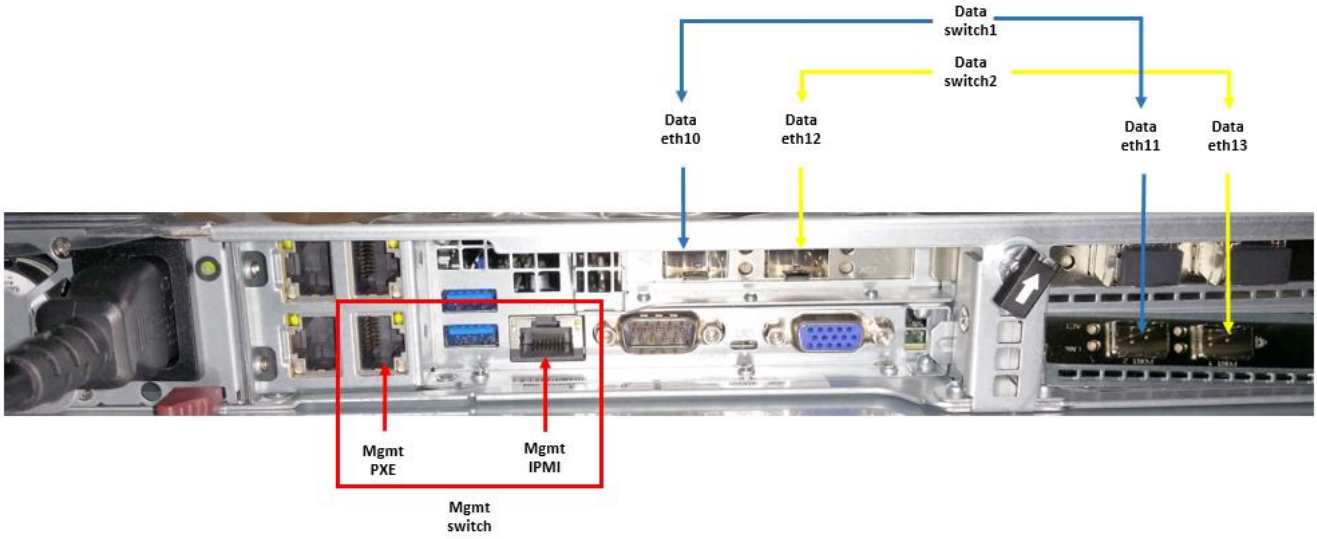
There are two networks in the Open DBaaS solution. There is a management network and a data network each with its own dedicated switch. The ports marked IPMI and PXE ports go to the 1G management network/switch, and the ports marked Data go to the 10G data network/switch.

Take the network cables and connect the IPMI port on the server to the management switch at the location marked on the switch for that node. Perform the same step next with the port marked PXE. Finally take a third network cable and connect the port marked with Data on the server to the data switch at the location marked for that node.

Since the S822LC (8001-22C) "Briggs" and S821LC (8001-12C) "Stratton" server nodes are not identical we include below examples depicting ports for both models.

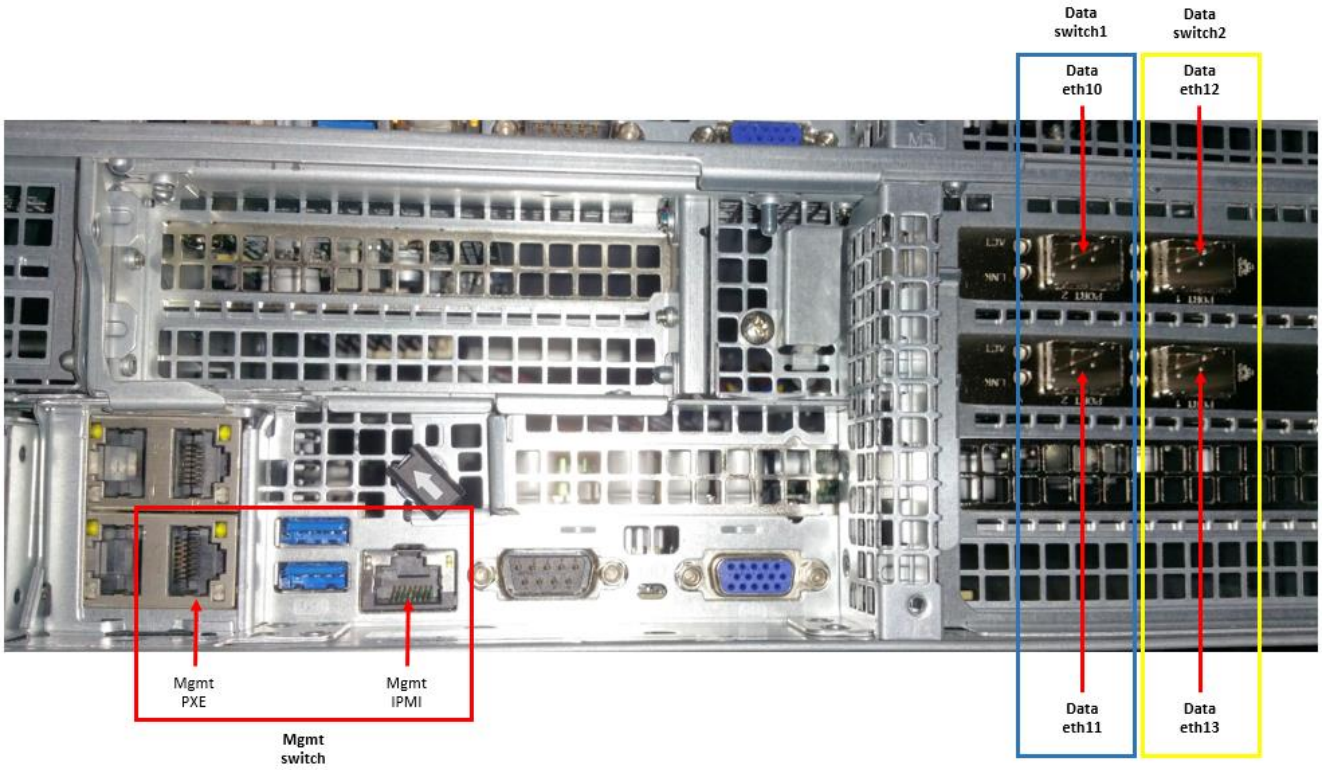
EXAMPLE: STRATTON NODE

Rear view of Stratton server with labels



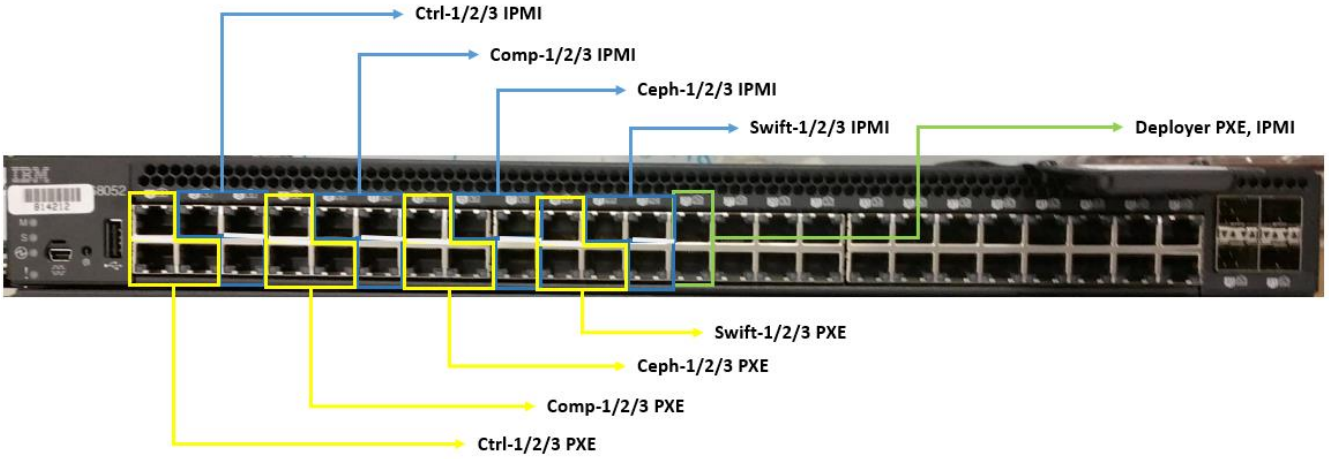
EXAMPLE: BRIGGS NODE

Rear view of Briggs server with labels

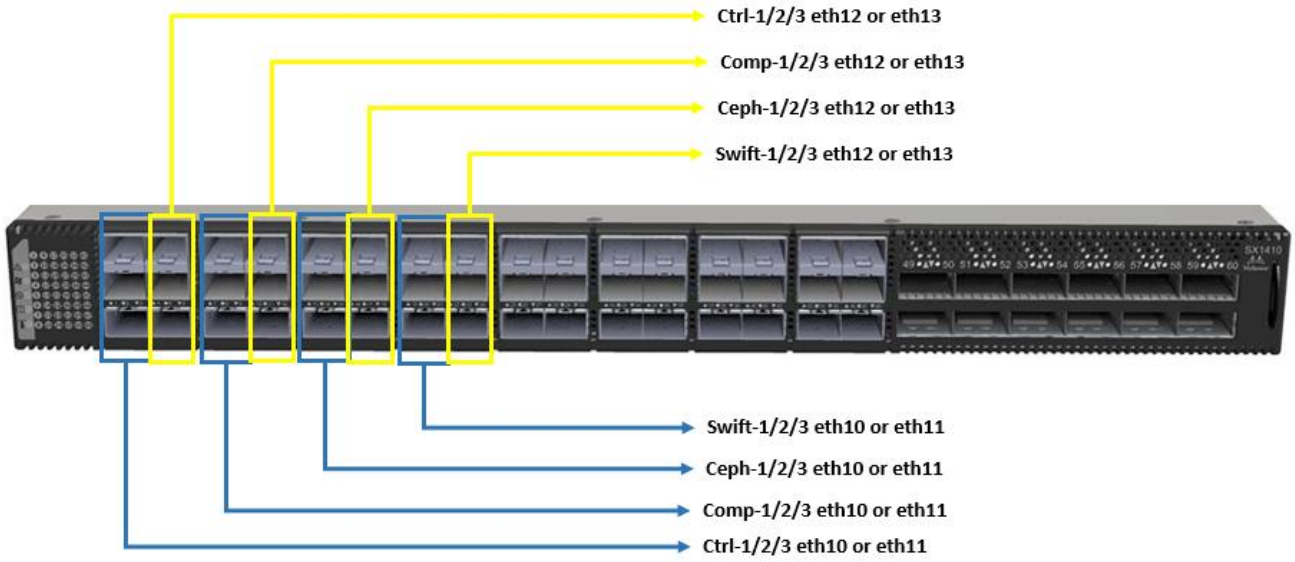


Once the server nodes are cabled please complete cabling connections on the Mgmt and Data switches.

Rear-view of management network switch.



Rear-view of Data Network Switch.



Repeat the above steps all existing nodes, depending on their server node types – Controllers, Compute, Ceph and Swift.

POWER CONSIDERATIONS

Careful consideration is required in cabling up the Power for the servers. Servers, storage drawers and switches should be distributed among different PDUs The PDUs have to be provided independent Power drops to ensure that a single PDU failure or a single power line will not disrupt any of the services. For example say PDU1 and PDU3 are attached to the first power line and PDU2 and PDU4 are attached to the second power line. To ensure that the Ceph services are always available, Ceph-1, Ceph-2 and Ceph-3 nodes should not all be connected to PDU1 and PDU3. Instead Ceph-1 could be connected to PDU1, Ceph-2 to PDU2 and Ceph-3 to PDU3. For servers with redundant power supplies, each supply should be connected to a separate path (PDU/line). The same goes for the redundant network switches. One should be connected to PDU1/PDU3 and the other to PDU2/PDU4. This will ensure the highest availability.

STEP 6: DEPLOY THE CLUSTER

This section covers the power on, initialization, configuration and installation of Open Database as a Service cluster. This deployment kit provides an automated method to quickly and more predictably go from assembly to a tuned operational state of the cluster's infrastructure.

THE STEPS INVOLVED IN DEPLOYING A CLUSTER INCLUDE:

A	Obtain the default configuration file
B	Tailor the configuration file for your environment
C	Validate the configuration file
D	Provision the cluster
E	Configure OpenStack services.
F	Verify the deployment

The orchestration of the above steps is performed by a tool called cluster-genesis. Please refer to the user guide here for details on cluster-genesis.

http://cluster-genesis.readthedocs.io/en/latest/

OBTAIN THE DEFAULT CONFIGURATION FILE

The deployment automation (cluster-genesis) uses a config file to specify the target cluster configuration. The deployment tooling uses this YAML text file to specify the IP address locations of the managed switches and the system nodes attached to the switches as well as other useful details for the deployment process.

Go here for a copy of the Open Database as a Service (DBaaS) Small Cluster config file.

https://github.com/open-power-ref-design/dbaas/blob/master/config.yml

TAILOR THE CONFIGURATION FILE FOR YOUR ENVIRONMENT

The config.yml file contains a lot of configuration information. To enable a cluster tailored to your environment, edit the YAML file with the configuration parameters you collected in Step 2, replacing the **RED** text with your data. The image below zooms in on only those lines to edit.

Editable Portions of the config.yml file

```
~ ~ ~ ~ ~ ~ ~  licensing comments and series of YAML ~ ~ ~ ~ ~
ipaddr-mgmt-network: 192.168.16.0/24   ← Type your management network range here.
ipaddr-mgmt-switch:
   rack1: 192.168.16.20                ← Type your management switch IP address here.
ipaddr-data-switch:
   rack1:
      - 192.168.16.25                  ← Type your data switch1 IP address here.
      - 192.168.16.30                  ← Type your data switch2 IP address here.
external-floating-ipaddr: 10.0.16.50   ← Type your external floating IP address here.

 ~ ~ ~ ~ ~ ~ ~ ~  series of YAML and comments ~ ~ ~ ~ ~
networks:
   external1:
      description: Organization site or external network
      addr: 10.0.16.0/22                   ← Type your External Data IP address range here.
      broadcast: 10.0.19.255               ← Type your External network broadcast address here.
      gateway: 10.0.16.1                   ← Type your external router address here.
      dns-nameservers: 10.0.16.200         ← Type your DNS server IP address here.
      dns-search: mycompany.domain.com     ← Type your DNS domain name here
      method: static
      eth-port: osbond0
 ~ ~ ~ ~ ~ ~ ~ ~  series of YAML and comments ~ ~ ~ ~ ~
node-templates:
   controllers:
      hostname: ctrl-                  ← Type your controller-1 hostname here.
      userid-ipmi: ADMIN               ← Type your IPMI user  here.
```

password-ipmi: admin ← *Type your IPMI password here.*

cobbler-profile: ubuntu-16.04.1-server-ppc64el ← *Type your OS Level here*

~ ~ ~ ~ ~ ~ ~ continuation of YAML and comments ~ ~ ~ ~ ~

VALIDATE THE CONFIGURATION FILE

To ensure that the format of the modified configuration file is valid, it is recommended that it gets validated by following these steps:

```
$ git clone git://github.com/open-power-ref-design/dbaas

$ cd dbaas
$ TAG=$(git describe --tags $(git rev-list --tags --max-count=1))
$ cd ..
$ apt-get install python-pip
$ pip install pyyaml
$ git clone git://github.com/open-power-ref-design-toolkit/os-ser-
vices
$ cd os-services
$ git checkout $TAG
$ ./scripts/validate_config.py --file ../dbaas/config.yml
$ cd ..
```

Before proceeding with the Provisioning step below, you will need to ensure that all the nodes in the cluster will have direct access to the internet. If the cluster is being configured in a private network without direct internet access, then it is recommended that the deployer node be provided internet access, and it acts as a NAT host to route all the nodes in the cluster.

PROVISION THE CLUSTER

Once the deployment cluster's configuration is finalized, the cluster can be provisioned by kicking off the automation tool (cluster-genesis). The steps are:

```
$ git clone git://github.com/open-power-ref-design-toolkit/cluster-
genesis
$ cd cluster-genesis
$ TAG=$(git describe --tags $(git rev-list --tags --max-count=1))
$ git checkout $TAG
$ ./scripts/install.sh
$ source scripts/setup-env
$ gen deploy
```

If custom install images are required, more details of the steps to follow are found here:

http://cluster-genesis.readthedocs.io/en/latest/OPCG_running_OPCG.html

CONFIGURE OPENSTACK SERVICES

The provisioning step above, runs for about 2 hours and completes the installation of the operating system on all nodes. Upon completion of the installation it then prepares the cluster (bootstrap) as well. The bootstrap step is automatically triggered when cluster-genesis is completed. At this point various OpenStack parameters need to be configured. To prepare for this phase, please collect the following information:

Parameter	Description	Example
Keystone Password	The password that will be used for the OpenStack authentication services	passw0rd
VRRP ID	Virtual Router ID that will be in the range of 1-255 and has to be unique across the network	202
USED IPs	The range of IP addresses in the private networks that are already taken, and cannot be assigned to nodes in the cluster. This includes the addresses assigned by cluster-genesis and those reserved for use by Trove. In this example, 172.29.236.100 thru 200 are reserved for Trove.	172.29.236.100..172.29.236.200 172.29.236.1..172.29.236.50 172.29.240.1..172.29.240.50 172.29.244.1..172.29.244.50

For complete information on all the various options to configure the OpenStack deployment, please refer to the openstack-ansible documentation available here.

https://docs.openstack.org/project-deploy-guide/openstack-ansible/newton

The minimal set of configuration options are provided below. To proceed with this, login to the first controller node (ctrl-1).

```
$ ssh ctrl-1
```

Edit the keystone stanza and set the desired keystone password, in the /etc/openstack_deploy/user_secrets.yml file.

```
$ vi /etc/openstack_deploy/user_secrets.yml
```

```
## Keystone Options
keystone_container_mysql_password:
keystone_auth_admin_token:
keystone_auth_admin_password: passw0rd
keystone_service_password:
keystone_rabbitmq_password:
```

The valid range for the next parameter (external virtual router ID) is 1-255 and it must be unique for each cluster. Edit the external virtual router ID, in the /etc/openstack_deploy/user_variables.yml file.

```
$ vi /etc/openstack_deploy/user_variables.yml
```

```
# Defines the default VRRP id used for keepalived with haproxy.
# Overwrite it to your value to make sure you don't overlap
# with existing VRRPs id on your network.  Default is 10 for the
# external and 11 for the internal VRRPs
haproxy_keepalived_external_virtual_router_id: 202
# haproxy_keepalived_internal_virtual_router_id:
```

Next edit the /etc/openstack_deploy/openstack_user_config.yml file to reserve ip addresses generally for the 172 networks: .1-.50. For the 172.29.236 network the .100-.200 range is also reserved. This is done with the "used_ips" field described in /etc/openstack_deploy/openstack_user_config.yml.example. The following values can be placed just above the "global_overrides" field.

```
$ vi /etc/openstack_deploy/openstack_user_config.yml
```

```
used_ips:
  - "172.29.236.100,172.29.236.200"
  - "172.29.236.1,172.29.236.50"
  - "172.29.240.1,172.29.240.50"
  - "172.29.244.1,172.29.244.50"
```

Now the cluster is ready to complete the final step of deployment:

```
$ cd os-services
$ ./scripts/create-cluster.sh 2>&1 | tee -a /root/create-cluster.out
```

Monitor the /root/create-cluster.out file for progress and indication of completion.

(OPTIONAL) CONFIGURE OPERATIONAL MANAGEMENT SERVICES

Part of the software stack deployed in the Controller nodes is a set of services that are collectively called "Operational Management" (or OpsMgr for short). This complements OpenStack with popular DevOps tools that provide additional function to monitor availability and health of your cluster and to collect log information from all elements of the cluster and present key metrics that are useful for continued operations. These tools are, respectively, Nagios Core and the Elastic Stack. In addition, Operational Management also offers Hardware inventory and integration services as an extension to the OpenStack Horizon user interface. From that extension users can visualize all their hardware devices and launch to all integreated DevOps tools to perform further operational tasks.

For more information on Operational Management and to learn about optional configuration of its installed services and tools please consult the associated README.

https://github.com/open-power-ref-design-toolkit/opsmgr/blob/master/recipes/privatecloud-newton/README.rst

VERIFY THE DEPLOYMENT

The Configuring OpenStack step above will run for a couple of hours (~4). After its completion to verify that the Open DBaaS cluster is operational, please follow the 'Verifying an install' section of the README.

https://github.com/open-power-ref-design/dbaas/blob/master/README.rst

APPENDIX

APPENDIX A - USING AN INTERNET PROXY DURING DEPLOYMENT

If the use of a network proxy is needed to access the internet then proxy settings must be specified in the 'deployment-environment' setting of the Genesis config.yml file.

```
## Deployment environment variables
deployment-environment:
      http_proxy: "http://1.2.3.4:3128"
      https_proxy: "http://1.2.3.4:3128"
      no_proxy: "localhost,127.0.0.1"
```

Change the example values to suite the actual deployment environment. Additional "no_proxy" values may be needed.

These presense of these variables will force the use of a network proxy during deployment on all the downstream components.

If errors are encountered during the automated Genesis deployment and if deployment needs to be resumed manually on the bootstrap phase these variables must be set as Shell variables before the corresponding bootstrap scripts are executed. These scripts are bootstrap-solution.sh (genesis) or bootstrap-cluster.sh (os-services). This is normally a step that Genesis automates, but doing it manually avoids having to rerun Genesis from scratch. The variables can be set like this:

```
$ export http_proxy="http://1.2.3.4:3128"

$ export https_proxy="http://1.2.3.4:3128"

$ export no_proxy="localhost,127.0.0.1"
```

At this point you should be able to re-run the bootstrap scripts in the same shell where you exported these variables.

REFERENCE LINKS

Go here for the Stratton Power S821LC MTM 8001-12C Redbook.

https://www.redbooks.ibm.com/Redbooks.nsf/Redbook-Abstracts/redp5406.html

Go here for the Briggs Power S822LC MTM 8001-22C Redbook:

http://www.redbooks.ibm.com/redpapers/pdfs/redp5407.pdf

Go here for the DBaaS repository:

https://github.com/open-power-ref-design/dbaas

Go here for the openstack-ansible documentation:

https://docs.openstack.org/developer/openstack-ansible/newton

Printed in the United States of America June 2017

IBM Systems
294 Route 100, Building SOM4
Somers, NY 10589-3216

The IBM home page can be found at **ibm.com®**.

Version 1.6
12 June 2017