How to Deploy OpenStack Swift on OpenPOWER

Version 1.3

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 the OpenStack Swift (Object Storage) Service. This provides a standalone configuration for OpenStack Swift. For an OpenStack Swift configuration that is integrated with a private compute cloud, see the private compute cloud documentation. The standalone OpenStack Swift configuration may include distinct nodes for the controller, proxy, metadata, and object storage.

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 this step, a standalone OpenStack Swift 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/standalone-swift/blob/master/docs/design.pdf

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

https://github.com/open-power-ref-design/standalone-swift/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&disable-Cookie=Yes

The Bill of Materials lists three possible configurations for your OpenStack Swift solution: "Swift Small", "Swift Medium", and "Swift Large". Use the guidelines in the Bill of Materials to choose which configuration to build.

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 your 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 files for each swift configuration:

Swift Small

https://github.com/open-power-ref-design/standalone-swift/blob/master/config-small.yml

Swift Medium

https://github.com/open-power-ref-design/standalone-swift/blob/master/config-medium.yml

Swift Large

https://github.com/open-power-ref-design/standalone-swift/blob/master/config-large.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 deployment 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
- 3 Network Interface connections: IPMI, 10G (high-speed), 1G (Mgmt.).
- Ubuntu 16.04 LTS before beginning with deployment.

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 OpenStack Swift solution. Optimal server resources have been selected to attain the right balance between compute, storage and network I/O.

The five classes of servers in a standalone OpenStack Swift solution are:

Controller Nodes: These servers host the OpenStack Control services and Operational Management services. The design is triple redundant and allows for scaling to larger (intra-rack) clusters. 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'. These nodes are present in the 'Swift Small', 'Swift Medium', and 'Swift Large' configurations.

Swift Proxy Nodes: These servers host the OpenStack Swift Proxy services. The 'Swift Medium' and 'Swift Large' configurations for this solution have 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 'Proxy'. In the 'Swift Small' configuration, the Swift Proxy services run inside an lxc container on the Controller Nodes so there is no dedicated hardware for 'Proxy' nodes in the rack.

Swift Metadata Nodes: These servers, along with their built in storage, host the OpenStack Swift Metadata. The 'Swift Large' configuration for this solution has three Stratton Power S812LC MTM 8001-12C nodes with SSDs (solid state drives) to serve this role. These nodes are referenced in the rest of the document with a prefix of 'Meta'. In the 'Swift Small' and 'Swift Medium' configurations for this solution, the Metadata services and Object storage services are converged onto the same hardware (see Swift Object and Metadata Nodes below) so there is no dedicated hardware for 'Meta' nodes in the rack.

Swift Object Nodes: These servers, together with their attached JBOD (just a bunch of disks) storage drawers, provide Object storage services. The 'Swift Large' configuration for this solution has three Stratton Power S812LC MTM 8001-12C nodes with attached SMC 4U90 disk drawers serving this role. These nodes are referenced in the rest of the document with a prefix of 'Obj'. In the 'Swift Small' and 'Swift Medium' configurations for this solution, the Metadata services and Object storage services are converged onto the same hardware (see Swift Object and Metadata Nodes below) so there is no dedicated hardware for 'Obj' nodes in the rack.

Swift Object and Metadata Nodes: These servers, together with their attached JBOD (just a bunch of disks) storage drawers, provide Metadata and Object storage services. The 'Swift Small' and 'Swift Medium' configurations for this solution have three Stratton Power S812LC MTM 8001-12C nodes serving this role. The Swift Metadata resides on SSDs (solid state drives). The Swift Object storage resides on HDDs (hard disk drives) in the attached (JBOD) SMC 4U90 disk drawers. These nodes are referenced in the rest of the document with a prefix of 'Obj+Meta'. In the 'Swift Large' configuration for this solution, the 'Obj' and 'Meta' nodes are on separate dedicated hardware so there are no converged 'Obj+Meta' nodes in the rack.

Racking the components

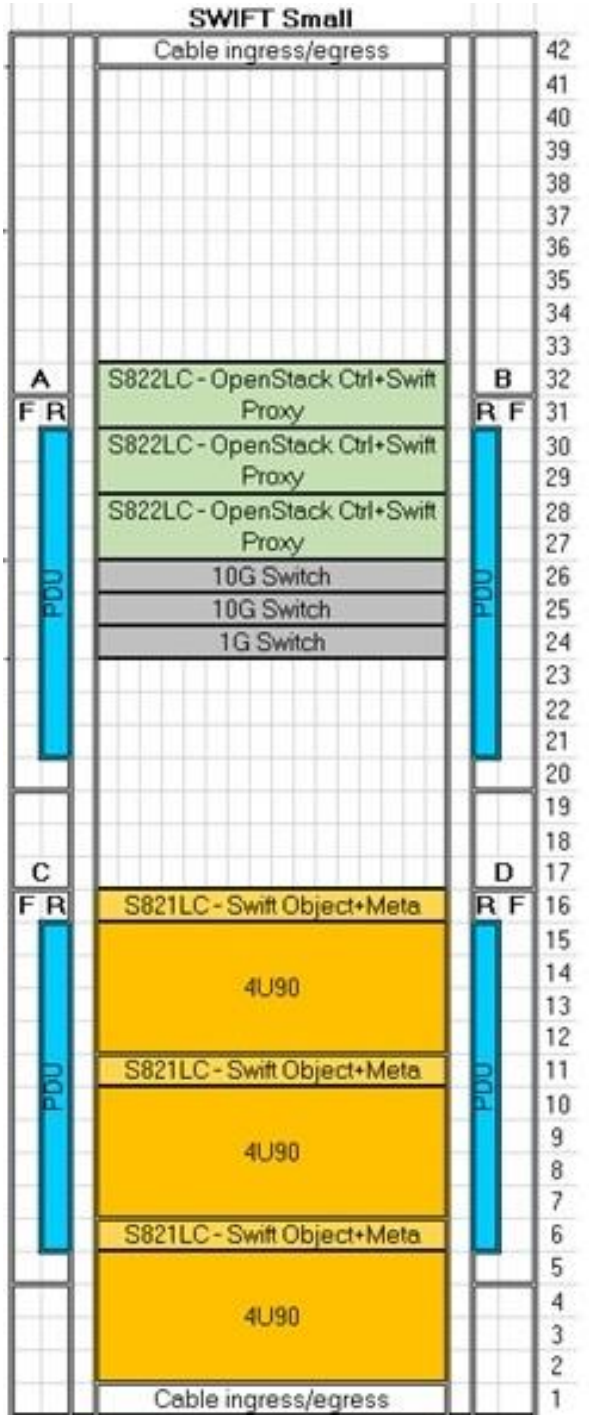
Place the intra-rack (leaf) network switches in **U24**-**U26** as follows:

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

Place the Obj + Meta nodes with the attached Storage drawers at the bottom of the rack. Place the OpenStack Ctrl nodes (which include Swift Proxy services) above the switches.

If more than 4 PDUs are needed, place 2 horizontal PDUs in 40U and 41U. Spine switches take priority over additional PDUs. If 40U and 41U are occupied by Spine Switches, place horizontal PDUs in next available slots.

RESULTING EXAMPLE: 6 NODE OPENSTACK SWIFT SMALL CLUSTER



Racking the components

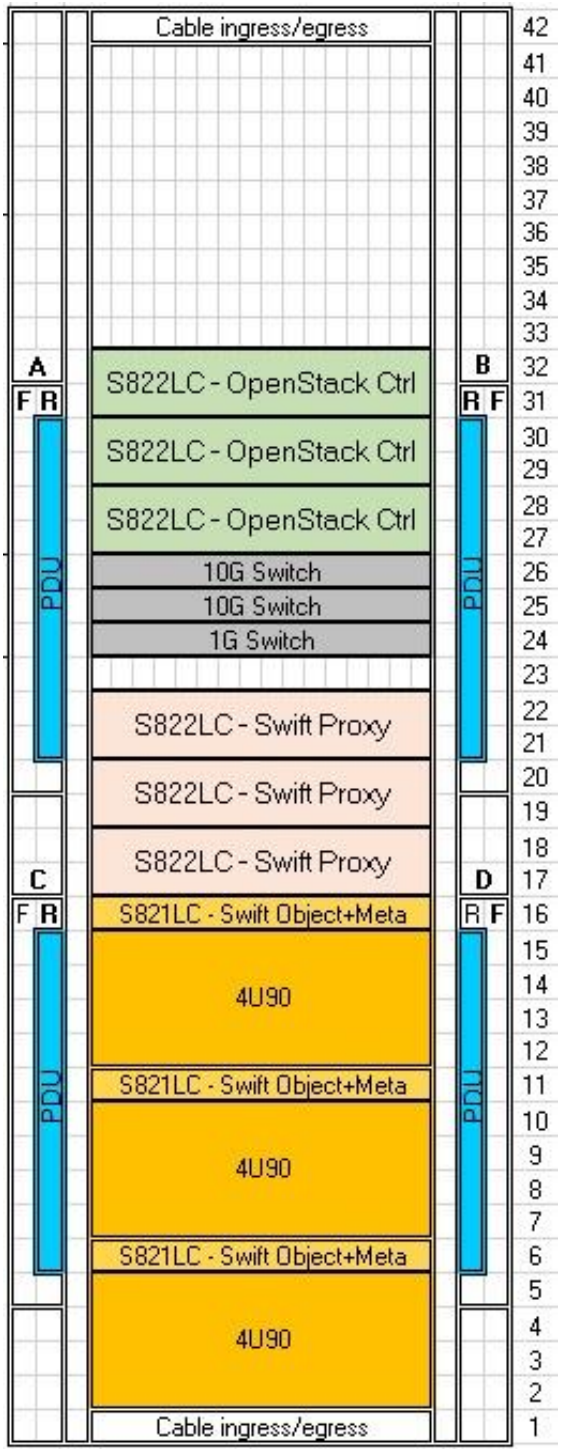
Place the intra-rack (leaf) network switches in **U24**-**U26** as follows:

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

Place the Obj + Meta nodes with the attached Storage drawers at the bottom of the rack. Place the Proxy nodes above the Obj + Meta nodes. Place the OpenStack Ctrl nodes above the switches.

If more than 4 PDUs are needed, place 2 horizontal PDUs in 40U and 41U. Spine switches take priority over additional PDUs. If 40U and 41U are occupied by Spine Switches, place horizontal PDUs in next available slots.

RESULTING EXAMPLE: 9 NODE OPENSTACK SWIFT MEDIUM CLUSTER



Racking the components

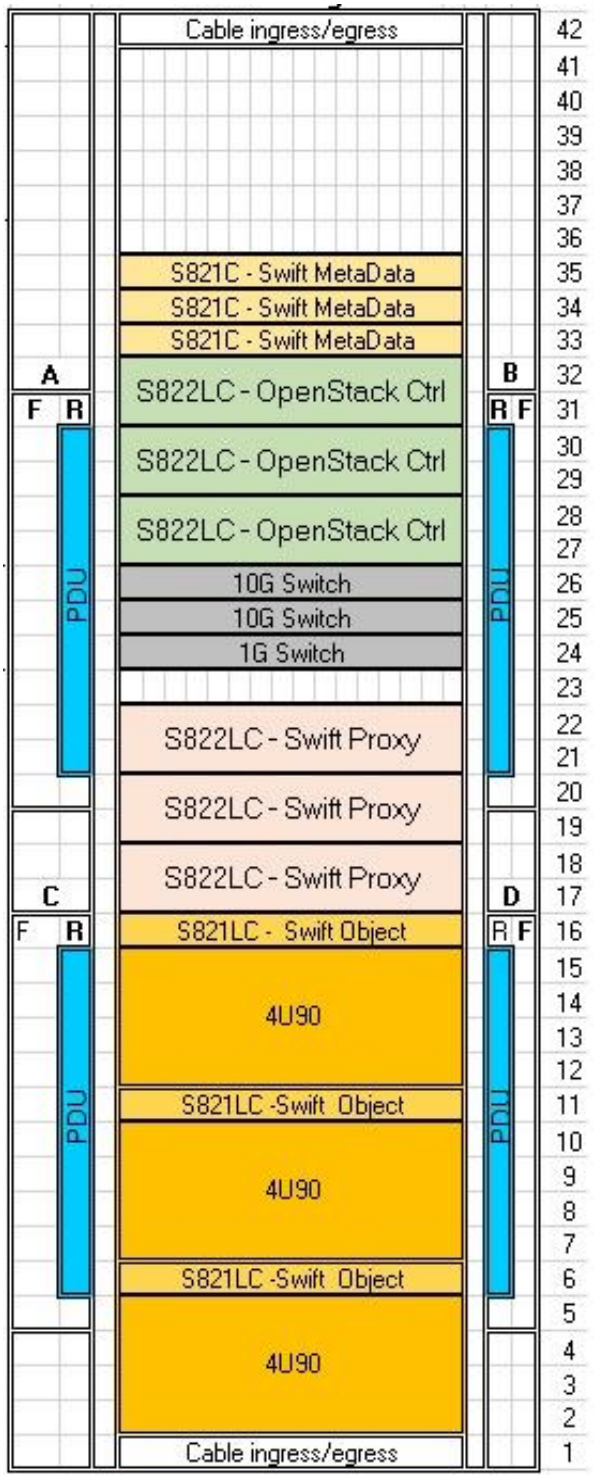
Place the intra-rack (leaf) network switches in **U24**-**U26** as follows:

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

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

If more than 4 PDUs are needed, place 2 horizontal PDUs in 40U and 41U. Spine switches take priority over additional PDUs. If 40U and 41U are occupied by Spine Switches, place horizontal PDUs in next available slots.

RESULTING EXAMPLE: 12 NODE OPENSTACK SWIFT LARGE CLUSTER



STEP 5: CABLE THE SYSTEMS

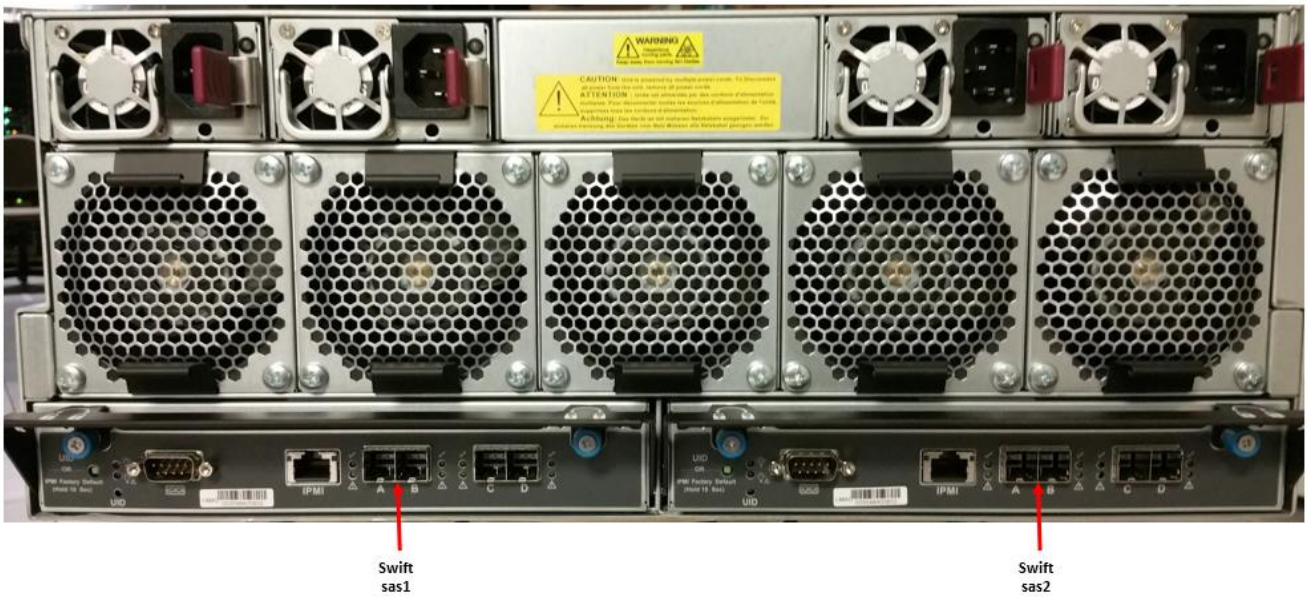
Cabling of an OpenStack Swift solution consists of these distinct efforts. One is the storage cabling, another is the network cabling, and finally powering the systems and switches. Presented first is the storage cabling.

In this step connect the Swift Object nodes (or Swift Object + Metadata Nodes) to the storage drawers, using the SAS cables.

EXAMPLE: SWIFT OBJECT NODE

Rear view of Swift Object node connections with labels.

Rear view of Swift Storage drawer with labels.



Repeat the above step for all the Swift Object Server nodes.

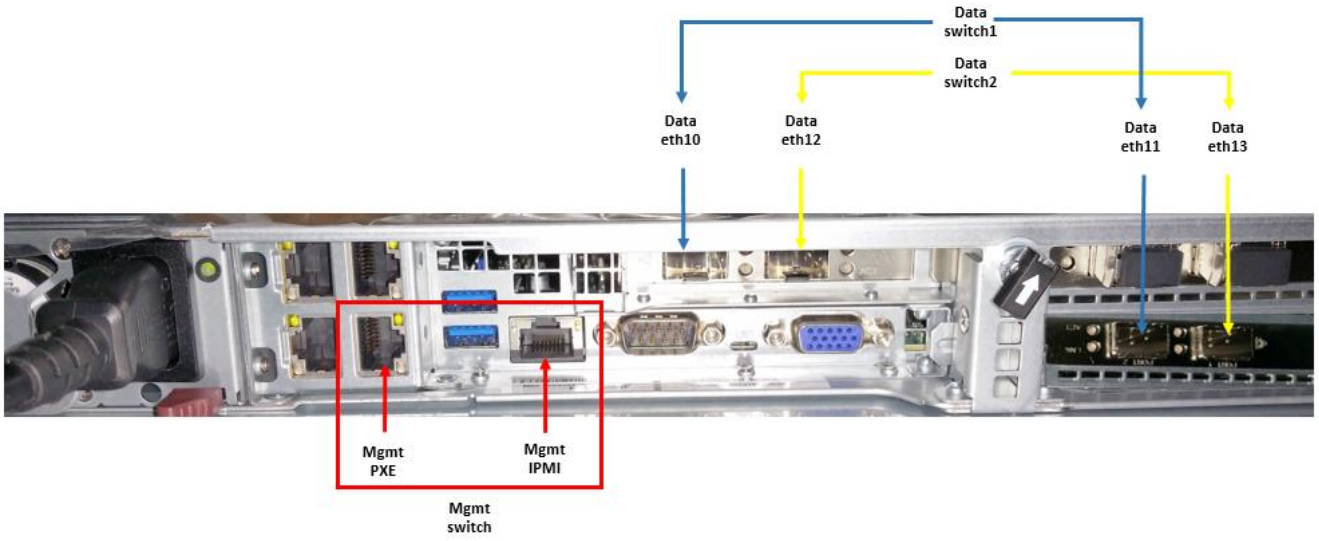
Next we will provide an example of cabling up the network. There are three networks in the OpenStack Swift solution. There is a management network with a single dedicated switch and two data networks using two high speed data switches.

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 high speed network cables and connect the ports marked with Data on the server to the data switches 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.

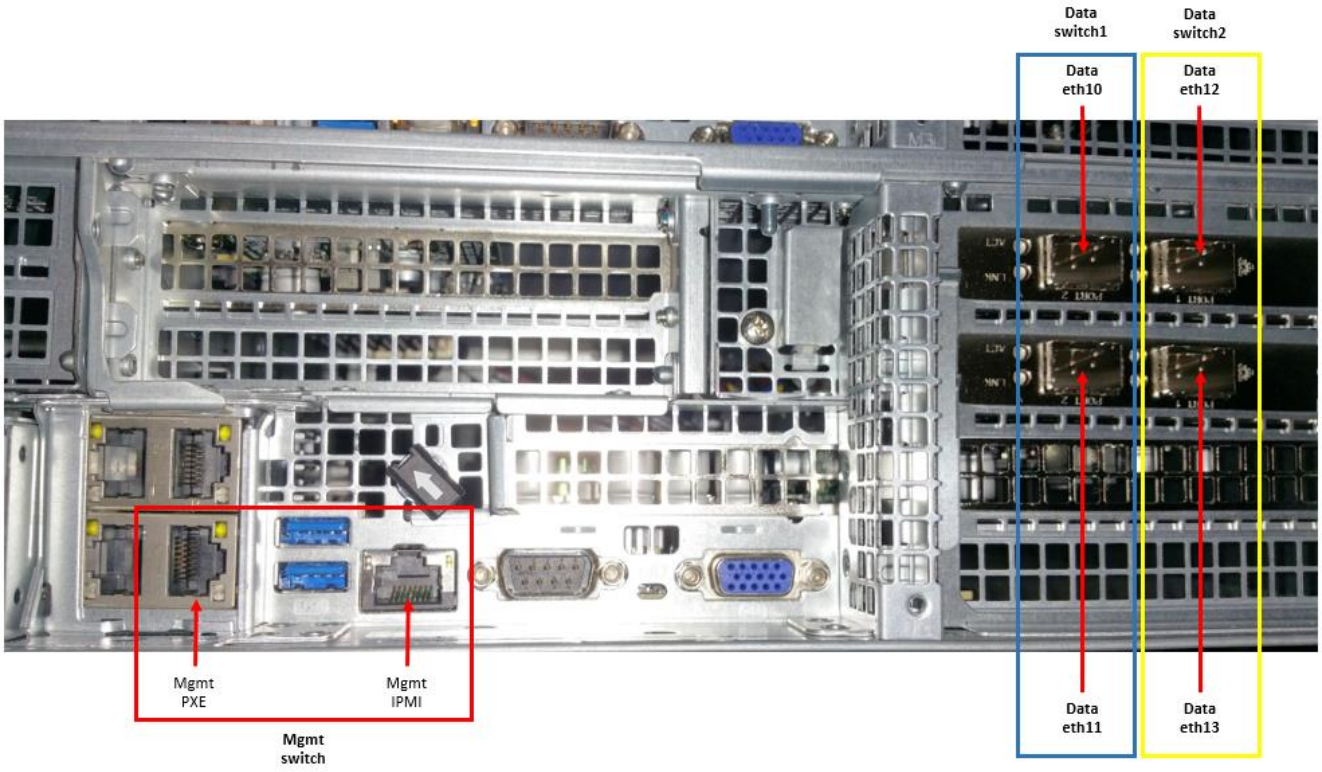
EXAMPLE: STRATTON NODE

Rear view of Stratton server with labels



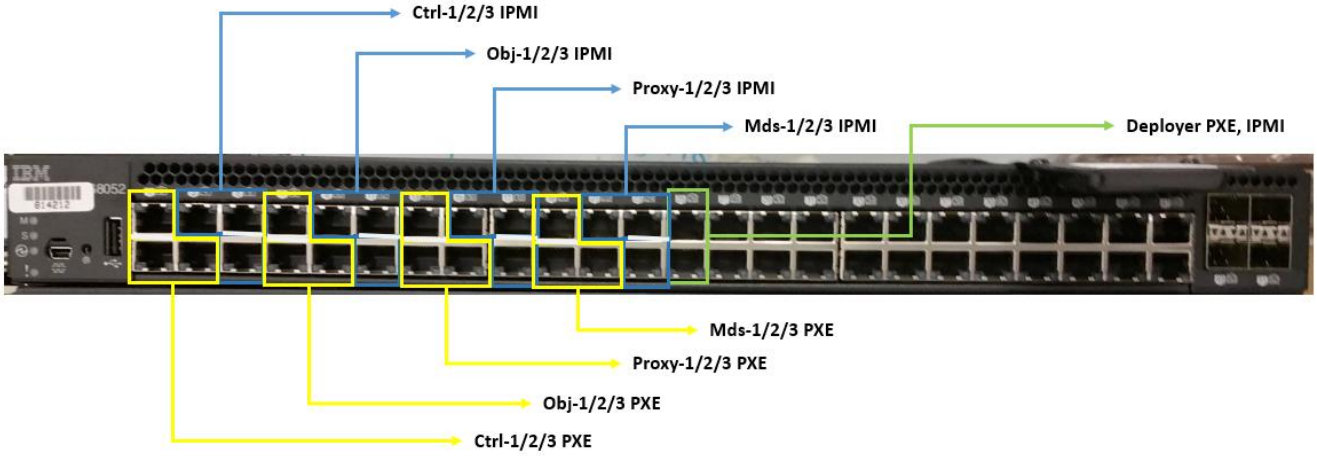
EXAMPLE: BRIGGS NODE

Rear view of Briggs server with labels



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

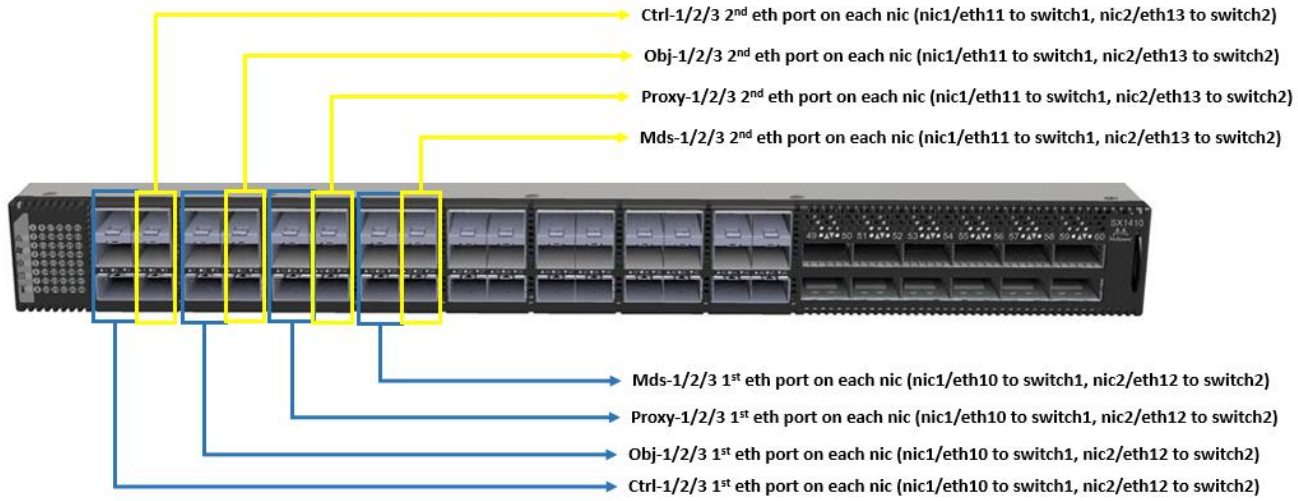
Rear-view of management network switch.



Wiring of the management switch depends on the Swift configuration used: small, medium or large. Start with the control nodes (Ctrl), 3 at a time and use the initial ports of the switch – first PXE and then IPMI networks. Next wire similarly the first 3 Swift object (Obj), then Swift

proxies (Proxy) and Swift metadata (Mds) - if your configuration has them. Towards the end of the switch you will also need a couple of ports to wire the deployer node.

Rear-view of Data Network Switch.



Wiring of the data switches depends on the Swift configuration used: small, medium or large. The wiring must alternate NICs in each server and the switches it connects to provide high availability. For example, The first port on NIC1 goes to Switch1 and the first port on NIC2 goes to Switch2, then second port on NIC1 goes to Switch2 and second port on NIC2 goes to Switch1. This guarantees that if either a NIC or a Switch fails the server will still have a data connection to operate (degraded, but available). Start with the control nodes (Ctrl), 3 at a time, then proceed to wire the Swift object nodes (Obj) and the Swift proxies (Proxy) or metadata (Mds) nodes - if your configuration has them.

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. To ensure that the Swift services are always available, servers of a given type should not all be connected to the same pair of PDUs.

STEP 6: DEPLOY THE CLUSTER

This section covers the power on, initialization, configuration and installation of an Openstack Swift 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 Swift Small configuration file.

https://github.com/open-power-ref-design/standalone-swift/blob/master/config-small.yml

Go here for a copy of the Swift Medium configuration file.

https://github.com/open-power-ref-design/standalone-swift/blob/master/config-medium.yml

Go here for a copy of the Swift Large configuration file.

https://github.com/open-power-ref-design/standalone-swift/blob/master/config-large.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

```
~ ~ ~ ~ ~ ~ ~ bunch of licensing comment and 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/standalone-swift

$ cd standalone-swift
$ 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 ../standalone-swift/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 initial 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

CONFIGURING OPENSTACK SERVICES

The provisioning step above runs for about 2 hours 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.	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. 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:
  - "172.29.236.1,172.29.236.50"
  - "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 Swift 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/private-cloud-newton/README.rst

VERIFY THE DEPLOYMENT

The create-cluster provisioning step above will run for several hours (~3). After its completion to verify that the OpenStack Swift cluster is operational, please follow the 'Verifying an install' section of the README.

https://github.com/open-power-ref-design/standalone-swift/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:

http://www.redbooks.ibm.com/abstracts/redp5406.html?Open

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

http://www.redbooks.ibm.com/abstracts/redp5407.html?Open

Go here for standalone-swift repository:

https://github.com/open-power-ref-design/standalone-swift

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 1.2
12 June 2017