Configuring Object Storage (Swift)

Contents

Configuring Object Storage (Swift)

This page provides detailed instructions on the configuration of the input model for HP Helion OpenStack Object Storage and validating the object storage cloud model.

Before starting the Swift deployment ensure that HP Helion OpenStack deployer node is successfully installed.

Setting up the Cloud Model

- *Hardware Requirement*
- *Allocate servers and disk drives*
- *Allocate networks*
- *Allocate a Ring -Builder Node*
- *Ring specifications*
- Making site-specific changes to configuration parameters (**can we exclude this for beta???**)

Hardware Requirement

It is recommended to have two **disk drives on 3 controller nodes** to run swift services. **How do we add extra drive ? is it in baremetal file etc??**

Allocate servers and disk drives

The disk model describes the number of disk present on a particular server and its usage. Disk are used by the operating system (for example: for the root, log crash filesystem) and swift storage.

A disk model specifies a number of disk drives/devices and then alloctes them for a specific purpose. The general syntax is as follows:

```
device-groups:

  - name: group_name
    devices:
      -  name: sdc
      -  name: sdd
      -  name: sde
    consumer:
        name: <subsystem>
        attrs:
            <attributes-sepecific-to-subsystem>
```

The following is a sample for object store. The **consumer** contains an item called **name**. Here the **subsystem (name)** is **swift**. The **attrs** must contain an item called **rings**. The **rings** item contains a list of ring names.

```
device-groups:

  - name: swiftdevices
    devices:
      -  name: sdc
      -  name: sdd
      -  name: sde
    consumer:
        name: swift
        attrs:
            rings:
              -  name: account
              -  name: container
```

```
              -  name: object-0
```

In the above example, a server using this model has at least three drives: `dev/sdc`, `/dev/sdd`, `/dev/sde`. Apart from the listed drives there may be other drives present in the server which are allocated to other subsystem or used as the boot device. If a device is not present in the disk mode, but exists on a server, it will not be used. Conversely, if a device is listed in a disk model and is not present in the server, the deployment fails. In this case you must either add the device or change the disk model so that the device is not mentioned.

Note: The **device-group** name is for information and has no effect on the drives usage of Object Storage.

The `sdc sdd`,and `sde` drives are used and dedicated to the Object Storage system. And, the devices are used by the account, container, and object-0 rings. Refer *ring specification* for more information on rings.

You can modify or edit the disk models to suite your usage of the system. For example, given the same set of drives, you can allocate devices to rings in a different topology as follows:

```
device-groups:

  - name: metadata
    devices:
      -  name: sdc
    consumer:
        name: swift
        attrs:
            rings:
              -  name: account
              -  name: container
 - name: data
 devices:
      -  name: sdd
      -  name: sde
    consumer:
        name: swift
        attrs:
            rings:
              -  name: object-0
              -  name: object-1
```

In the above example `/dev/sdc` is allocated to account and container rings, whereas `/dev/sdd` and`dev/sde` are devoted to object storage using two different storage policies (object-0 and object-1). **Can i remove object-1 from the storage policy as for beta??) As per *HPM-1151* we support only one storage ploicy??.**

Allocate networks

Information is required for beta 0?

Allocate a Ring -Builder Node

Information is required for beta 0?

Ring Specification

The ring maps the logical names of data to locations on a particular disks. There is a separate ring for account database, account database, and individual objects, but each rings works similarly. The rings are built and managed manually by a utility called the ring-builder. The ring-builder also keeps its own builder file with the ring information and additional data required to build future rings.

Using the ring -sepcification key you can specify the ring in the input model. Also, there is an entry for each distinct Swift system. Therefore, in the input model keystone region name in the specified in the **region-name**.

In the example files (**do we have any such file in build**??), a ring-specification is located in `config/swift/rings.yml` file.The sample of `rings.yml` is as follows:

```
ring-specifications:
    - region-name: region1
      rings:
        - name: account
          display-name: Account Ring
          min-part-time: 24
          partition-power: 17
          replication-policy:
            replica-count: 3
        - name: container
          display-name: Container Ring
          min-part-time: 24
          partition-power: 17
          replication-policy:
            replica-count: 3
        - name: object-0
          display-name: General
          default: yes
          min-part-time: 24
          partition-power: 17
          replication-policy:
            replica-count: 3
```

The above example shows that the **region1** has three rings as follows:

- **Account ring** - You must always specify a ring called "account". The account ring is used by Swift to store metadata about the projects in your system. In Swift, a Keystone project maps to a Swift account. The **display-name** is informational and not used.
- **Container ring** -You must always specify a ring called "container". The **display-name** is informational and not used.
- **Object ring** - You must always specify a ring called "object-0". It is possible to have multiple object rings, which is known as *storage policies*. But in this release we support only one storage policy, i.e. object-0. The **display-name** is the name of the storage policy and can be used by users of the Swift system when they create containers. It allows them to specify the storage policy that the container uses.

Ring Parameters

You can specify the ring parametes as follows:

- **name** - The **name** attribute must be "account" and "container" respectively. Also, you must have account and container ring. The display-name attribute can be set to any value and is not used by the system.
- **min-part-time**- This is the number of hours that the swift-ring-builder tool will enforce between ring rebuilds. On a small system, this can be as low as one hour. The value can be different for each ring.
- **partition-power** - The appropriate value is related to the number of disk drives you allocate for the account and container storage. We recommend that you use the same drives for both the account and container rings. Therefore, the **partition-power** value should be the same.
- **replication-policy** - The **replication-policy** attribute is used to specify that a ring uses replicated storage. The duplicate copies of the object are created and stored on different disk drives. All replicas are identical. If one is lost or corrupted, the system automatically copies one of the remaining replicas to restore the missing replica.
- **replica-count** - Defines the number of copies of objects created. The recommended value for the **replica-count** attribute is 3. That is, three copies of data are kept to ensure redundancy and continued access in the even of component failures.
- (Optional)**Swift-regions** - The swift-regions attribute applies to all rings in a given Keystone region.
- **swift-zones** - The swift-zones can be applied to all rings in a given region or can be specified for a specific ring

Once you have configured the object storage in the input model, you can verify the following configuration in ~/helion/my_cloud/definition/ on the deployer.

- Verify the configured ring-specification.
- Verify that all disk drives allocated to swift have specified a ring and has a corresponding ring in the **ring-specifications**.

  Note: It is an error to name a ring which does not have a specification.
- Verify that you have allocated the appropriate servers and their drives to the appropriate ring.

Object Storage Cloud Model Validation

Note: Before validating the objsct storage cloud model, run the configuration processor against your cloud model.

To validate object storage cloud model, execute the following command on the deployer :

```
ansible-playbook -i hosts/verb_hosts swift-ring-validate.yml
```

If any error occurs while executing the validation command, you can get the detailed information as follows:

- Logon to the node that has been designated as the Ring-Builder (**see "Allocate a Ring-Builder Node" above**).

  Note: You should run the swift-ring-validate playbook as mentioned above otherwise the files required to run the swiftlm-ring-supervisor will not be in place on the node.
- Execute the following command:

  ```
  sudo swiftlm-ring-supervisor --make-delta --report
  ```

The command reports an error or problem with the input model. It also prints a summanry of the ring create actions that are planned for the deploy phase of the process.

Examine the ring-delta file

Important: You should run the swift-ring-validate playbook as mentioned above otherwise the files required to run the swiftlm-ring-supervisor will not be in place on the node.

- Logon to the node that has been designated as the Ring-Builder (**see "Allocate a Ring-Builder Node" above**).
- Execute the following command to list all drives and the actions planned for the rings:

  ```
  sudo swiftlm-ring-supervisor --make-delta --report --detail detail
  ```

Validate the following information in the output:

- Drives are being added to all rings in the ring specifications.
- Servers are being used as expected (for example, you may have a different set of servers for the account/container rings than the object rings.
- The weight value is as expected. The weight value is the drive size expressed in GB (i.e., a 1 TB drive has a weight of 1024).