Mid Sized Cloud Model

Contents

Mid Sized Cloud Model

This section describes the configuration of the Swift services in the example mid sized cloud model.

Node Type

There are two type of nodes that is associated with Swift services. They are as follows:

- **Proxy, Container, Account (PAC) Node**: This node runs the Swift-proxy, Swift-account, and Swift-container services. The Swift-proxy service processes API requests and directs them to the Swift-account, Swift-container or Swift-object services for processing. The Swift-account and Swift-container handle requests to accounts and containers respectively.
- **Object (OBJ) Node**: This node runs the Swift-object service. The Swift-object service handles requests for objects.

Configuration of Swift Services in the Mid Sized Cloud Model

In the example mid sized cloud model, the Swift services are configured in the following `yml` files:

- data/control_plane.yml
  - The Proxy, Container, Account node type is assigned to a dedicated *cluster of nodes* as specified in the **swpac** cluster in `yml` file. These nodes are dedicated only to Swift services.
  - The Object node type is assigned to a dedicated *resource nodes* group as specified in the **swobj** group in `yml` file.
  - Request to Swift are directed to a virtual IP address (VIP) that is managed by a cluster as specified in the **core** cluster in data/control_plane.yml file. The requests are then directed on the MGMT network to the Swift-proxy service on one of the Proxy, Container, Account nodes.
- data/disks_swpac.yml

  The Proxy, Container, Account node type uses two disk drives to store account and container databases which is specified in this file.
- data/disks_swobj.yml

  The Object node type uses two disk drives to store account and container databases which is specified in this file.
- config/swift/rings.yml

  Swift account, container, and object storage are managed by Swift using a data structure known as a *ring*. This `yml` file provides the specification of the rings. For more information on ring specification, refer to *ring specification.*
- data/network_groups.yml

  This `yml` file specifies the MGMT and Swift networks. The Swift-proxy service uses the Swift network to communicate with the other Swift services and among themselves.
- The Swift-proxy service uses other cloud services as follows. Both of these are configured in the example to run on the **core** cluster:
  - Swift validates token by making requests to the Keystone service
  - Swift caches tokens and other data using the memcached service

Allocating Server

In the `data/baremetal.yml` file, you specify the roles for servers. For example, a Swift Proxy, Container, Account (PAC) node is specified as follows:

```
baremetal_servers:
.
.
.
- node_name: swpac1
  node_type: ROLE-SWPAC
  pxe_mac_addr: 26:67:3e:49:5a:a7
  pxe_interface: eth2
  pxe_ip_addr: 192.168.10.12
  ilo_ip: 192.168.9.12
  ilo_user: admin
  ilo_password: password
.
.
.
```

In the above example a **node_type** is assigned as **ROLE-SWPAC**, which indicates that this node is Swift Proxy, Container, Account (PAC) node.

Note: If you don't use Cobbler to install your servers then you must create `data/servers.yml` and configure nodes on the file.

The number of servers is specified in the **swpac** cluster in the `data/control_plane.yml` file. In the following example, the number of servers is set to 3.

```
control-planes:
    - name: ccp
      region-name: region1
      common-service-components:
        - logging-producer
        - monasca-agent
        - stunnel
      clusters:
.
.
.
- id: "4"
          name: swpac
          server-role: ROLE-SWPAC
          member-count: 3
          service-components:
            - ntp-client
            - swift-proxy
            - swift-account
            - swift-container
            - swift-ring-builder
            - swift-client
.
.
.
```

You can increase the **member-count** value so that more Proxy, Account, Container nodes are used. When you increase the **member-count** you must also increase the number of nodes with the `ROLE-SWPAC` role in the `data/baremetalConfig.yml` (or `data/servers.yml`) file. <**if we dont increase the nodes in the data/baremetalconfig.yml file then what error message do we get?**>

A Swift Object (OBJ) node is specified in `data/baremetalConfig.yml` file as shown in the following example:

```
baremetal_servers:
.
.
.
- node_name: swobj1
  node_type: ROLE-SWOBJ
  pxe_mac_addr: 8b:f6:9e:ca:3b:78
  pxe_interface: eth2
  pxe_ip_addr: 192.168.10.20
  ilo_ip: 192.168.9.20
  ilo_user: admin
  ilo_password: password
.
.
.
```

In the above example a `node_type` is assigned as `ROLE_SWOBJ`, which indicates that this node is Swift Object node.

To specify the number of Swift Object servers add more servers of type `ROLE_SWOBJ` to the `data/baremetalConfig.yml` (or `data/servers.yml`) file.

Note: For Swift Object, the `member-count` is automatically determined. It is recommended not to change or increment the `member count`.

Allocating Disk Drives

The disk model describes the number of disk present on a particular server and its usage. The examples include several disk models. The disk model used by any given server is determined as follows:

- The `node_type` of server as specified in `data/baremetalConfig.yml` (or `data/servers.yml`) determines the role of the server.
- There is a specification for each sever role in the `data/server_roles.yml` file. This specifies a disk model for a given server role. For example, for the `ROLE-SWOBJ`, the disk model is called `DISK_SET_SWOBJ`.

  `data/server_roles.yml` file specifies the disk model as follows:

  ```
  server-roles:


      - name: ROLE-SWPAC 16
        interface-model: INTERFACE_SET_SWPAC 17
        disk-model: DISK_SET_SWPAC 18


      - name: ROLE-SWOBJ 24
        interface-model: INTERFACE_SET_SWOBJ 25
        disk-model: DISK_SET_SWOBJ
  ```

- The Object server uses the `DISK_SET_SWOBJ` disk model in the `data/disks_swobj.yml` file.
- The Proxy, Account, Container servers use the `DISK_SET_SWPAC` disk model in the `data/disks_swpac.yml` file.

The structure of the `DISK_SET_SWOBJ` and `DISK_SET_SWPAC` disk models are similar. The `data/disks_swpac.yml` file specifies the disk mode as follows:

```
disk-models:
- name: DISK_SET_SWPAC
  volume-groups:
    - name: hlm-vg
```

```
      physical-volumes:
        - /dev/sda5
      ...
      consumer:
         name: os

  device-groups:
    - name: swiftpac
      devices:
        - name: /dev/sdb
        - name: /dev/sdc
      consumer:
        name: swift
        attrs:
          rings:
            - account
            - container
```

In the above example, Swift uses `/dev/sdb` and `/dev/sdc` disk drives and operating system uses `/dev/sda` disk drives.

The disk model has the following fields:

- **device-groups**: There can be several device groups. This allows different sets of disks to be used for different purposes. In this example, there is only one entry in the device groups list.
- **name**: This is an arbitrary name for the device group. The name must be unique, but is not otherwise used.
- **devices**: This is a list of devices allocated to the device group. For Swift, these devices must meet the criteria as explained in *Requirements for a disk device*
- **consumer**: This specifies the service that uses the device group. A `name` field containing **swift** indicates that the device group is used by Swift. If the name field contains other values, the Swift system will ignore the device group.
- **attrs.rings**: This lists the rings that the devices are allocated to. In this example, the disk model is used by Proxy, Account, Container nodes, so the **account** and **container** rings are listed. In the `DISK_SET_SWOBJ` disk model, `object-0` ring is listed. It would be an error to add the `object-0` ring to the `DISK_SET_SWPAC` disk model because the Swift-object service does not run on PAC node.

You should review the disk devices in the disk model before making any changes to the existing model. See *Making Changes to a Swift Disk Model*.

Requirements for a Disk Device

Disk devices listed in the **devices** list must meet the following criteria:

- The disk device must exist on the server. For example, if you add `/dev/sdx` to a server with only 3 devices then the deploy process fails.
- The disk device must be unpartitioned or have a single partition that uses the whole drive.
- Do not have a label on the partition. <**is there any example we can provide?**>
- Do not have an XFS filesystem that contains a filesystem label. <**is there any example we can provide?**>
- If the disk drive is already labeled as described above, the `swiftlm-drive-provision` process will assume that the drive has valuable data and refuses to use or modify the drive. <**any procedure we should include in the troubleshooting section so that the user can clean the drives and run install the process?** >

Making Changes to a Swift Disk Model

There are several reasons for changing the disk model as follows:

- It is advisable that if you have additional drives available then you can add the drives to the devices list.
- The disk devices listed in the example disk model have different names on your servers. This may be due different harware drives. Edit the disk model (`DISK_SET_SWPAC` or `DISK_SET_SWOBJ`) and change the device names to the correct names.

- If you prefer a different disk drive than the one listed in the model. For example, if `/dev/sdb` and `/dev/sdc` are slow hard drives and you have SDD drives available in `/dev/sdd` and `/dev/sde`. In this case, delete `/dev/sdb` and `/dev/sdc` and replace with `/dev/sdd` and `/dev/sde`.

Allocating Network

It is recommended not to change a network configurations.

Ring Specifications

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 ring works similarly. Ring-builder is a utility which builts and manually manages the rings. The ring-builder also keeps its own builder file with the ring information and additional data required to build future rings.

The rings are specified in the input model using the **ring-specifications** key. There is an entry for each distinct Swift system – hence the Keystone region name is specified in the **region-name** key of the input model.

In the example, a ring-specification is mentioned in `config/swift/rings.yml` file. The sample file of ring-specification is as follows:

```
ring-specifications:
    - region-name: region1
      rings:
        - name: account
          display-name: Account Ring
          min-part-time: 16
          partition-power: 17
          replication-policy:
            replica-count: 3
        - name: container
          display-name: Container Ring
          min-part-time: 16
          partition-power: 17
          replication-policy:
            replica-count: 3
        - name: object-0
          display-name: General
          default: yes
          min-part-time: 16
          partition-power: 17
          replication-policy:
            replica-count: 3
```

The above smaple file 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** - It is also known as Storage Policy. 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. In the example, the storage policy is called **General**.
- The swift-ring-builder will enforce a minimum of 16 hours between ring rebuilds.
- **Replica count**: The recommended value for the `replica-count` attribute is 3. Three copies of data are kept to provide resiliency and availability.

  The purpose of the **min-part-time, partition-power, replication-policy** and**replica-count** are described in the following section.

Making Changes to the Ring Specifications

The ring specification provided in the example model contains adequate configuration details to deploy Swift. However, there may be reasons for changing the model as follows:

- Your system has more servers and disks than in the mid sized cloud example. Therefore, you may require to pick a different value for `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. Hence, the `partition-power` value should be the same. For more information about selecting an appropriate value, refer to *Selecting a Partition Power*.
- You require high resiliency or availability and want a higher replica count. The `replica-count` is normally 3 (i.e., Swift will keep three copies of accounts and containers). It is **recommended** not to change the value to lower than 3.
- You require a different name for the Storage Policy. If so, edit the `display-name` of the object-0 ring. This must be a single word (character such as underscore or dash are allowed). This is visible to your end users.
- Your system has more servers and disks than in the mid sized cloud example and the time to replicate data during a ring rebuild is longer than 16 hours. However, this time is system-dependent so you will be unable to figure out the appropriate value for `min-part-power` until you have more experience with your system.

  Therefore, during the initial deployment of Swift, we suggest not to make changes in **min-part-time**.

Selecting a Partition Power

- Partition power is used to distribute the data uniformly across drives in a Swift nodes. It also defines the storage cluster capacity. You must set the part partition power value based on the total amount of storage you expect your entire ring to use.

  When storing an object, the object storage system hashes the name. This hash results in a hit on a partition (so a number of different object names result in the same partition number). Generally, the partition is mapped to available disk drives. With a replica count of 3, each partition is mapped to three different disk drives. The hashing algorithm used hashes over a fixed number of partitions. The partition-power attribute determines the number of partitions you have.

Select a partition power for a given ring that is appropriate to the number of disk drives you allocate to the ring for the following reasons:

- If you use a high partition power and have a few disk drives, each disk drives will have 1000s of partitions. With too many partitions, audit and other processes in the Object Storage system cannot "walk" the partitions in a reasonable time and updates will not occur in a timely manner.
- If you use a low partition power and have many disk drives, you will have 10s (or maybe only one) partition on a drive. The Object Storage system does not use size when hashing to a partition – it hashes the name.

  With many partitions on a drive, a large partition is cancelled out by a smaller partition so the overall drive usage is similar. However, with very small numbers of partitions, the uneven distribution of sizes can be reflected in uneven disk drive utilization (so one drive becomes full while a neighboring drive is empty).

An ideal number of partitions per drive is 100. If you know the number of drives, select part power approximately to 100 partition per drive. Usually, you install a system with a specific number of drives and add drives as your required for storage grows. However, you cannot change the value of the partition power. Hence you must select a value that is a compromise between current and planned capacity.

Important: If you are installing a small capacity system and you need to grow to a very large capacity but you cannot fit within any of the ranges in the table, please seek help from Professional Services to plan your system.

There are a few additional factors that can help or mitigate the fixed nature of the partition power, as follows:

- Account and container storage represents a small fraction (typically 1%) of your object storage needs. Hence, you can select a smaller partition power (relative to object ring partition power) for the account and container rings.
- For object storage, you can add additional storage policies (i.e., another object ring). When you have reached capacity in an existing storage policy, you can add a new storage policy with a higher partition power <**can we add new storage policy in Beta as we claim to support one?**> (because you now have more disk drives in your system). This means that you can install your system with a small partition power appropriate to a small number

of initial disk drives. Later when you have many disk drives, the new storage policy can have a higher value appropriate to the larger number of drives.

However, when you continue to add storage capacity you should note that existing containers continue to use their original storage policy. Hence, the additional objects must be added to new containers to take advantage of the new storage policy.

Use the following table to select an appropriate partition power for each ring. The partition power of a ring cannot be changed so it is important to select an appropriate value. This table is based on a replica count of 3. But, your replica count is different or you are unable to find your system in the table then refer to *Calculating Numbers of Partitions* for information of selecting a partition power.

The table assumes that when you first deploy the system (**by system- are we referring Swift here?**) you have a small number of drives (the minimum column in the table) and later you add drives.

Note:

- You use the total number of drives. For example, if you have 3 servers, each with two drives, the total number of drives is 6.
- The lookup should be done separably for each of the account, container and object rings. Since account and containers represent approximately 1-2% of object storage, you will probably use fewer drives for the account and container rings (i.e., you will have fewer Proxy, Account, Container (PAC) servers) – hence you your object rings may have a higher partition power.
- The largest anticipated number of drives imposes a limit in how a few drives (the minimum) you can have. See *Calculating Numbers of Partitions*. This means that if you anticipate significant growth, your initial system can be small, but under certain limit.
- The table assumes that disk drives are the same size. The actual size of a drive is not significant.

Partition Power

Number of drive during deployment (minimum)	Number of drives in largest anticipated system (maximum)	Recommended Part Power
6	5,000	14
12	10,000	15
36	40,000	17
72	80,000	18
128	160,000	19
256	300,000	20
512	600,000	21
1024	1,200,00	22
2048	2,500,000	23
5120	5,000,000	24

Calculating Numbers of Partitions

The Object Storage system hashes a given name into a specific *partition*. For each partition, for a replica count of 3, there are three *partition directories*. The partition directories are then evenly scattered over all drives. We can calculate the number of partition directories as follows:

```
number-partition-directories-per-drive = ( (2 ** partition-power) * replica-
count ) / number-of-drives
```

An ideal number of partition directories per drive is 100. However, the system operate normally with a wide range of number of partition directories per drive. The table ***Partition Power*** is based on the following:

- A replica count of 3
- For a given partition power, the minimum number of drives results in approximately 10,000 partition directories per drive. With more directories on a drive results in performance issues.
- For a given partition power, the maximum number of drives results in approximately 10 partition directories per drive. With fewer directories per drive results in an uneven distribution of space utilization.

It is easy to select an appropriate partition power if your system is a fixed size. Select a value that gives the closest value to 100 partition directories per drive. If your system starts smaller and then grows, the issue is more complicated. The following two techniques may help:

- Start with a system that is closer to your final anticipated system size – this means you can use a high partition power that suites your final system.
- You can add additional storage policies as the system grows. These storage policies can have a higher partition power because there will be more drives in a larger system. Note that this does not help account and container rings – storage policies are only applicable to object rings.