**HOS 2.0 Swift**

**Note: since we will be using ansible all the deployment procedure will change. So it is expected to have major change**

* Object Overview (include input model)
* Swift Architecture –description and diagram
* Swift Components:
  + Proxy server
  + Rings
  + Zone
  + Account and Container
  + Objects
  + Partitions
* Storage Policies and Policy zero ???
* Pre install –
  + Disk and Server model
  + Recommendation
  + Ring attributes
  + Other Attributes
  + Pre-requisite
* Install Swift
* Verify Deployed Swift nodes
* Post install
  + Adding servers
  + Adding drives to existing servers
  + Upgrading drives (typically to larger drives)
  + Adding storage policies (ie..a new object ring)
  + Removing servers
  + Removing drives from existing servers

Swift is highly scalable and durable object storage system that is designed to store large amount of unstructured data at low cost. Swift is horizontally scalable. It is highly durable architecture with no single point of failure.

Swift, offers cloud storage software so that you can store and retrieve lots of data with a simple API. It's built for scale and optimized for durability, availability, and concurrency across the entire data set. Swift is ideal for storing unstructured data that can grow without bound.

Swift allows you to store, retrieve and delete objects (with the associated metadata) in container through RESTful HTTP API.

**Swift characteristics**

* All object have URL.
* All objects have their own metadata
* RESTful HTTP API is medium of interaction with the object storage system
* Object data can be located anywhere in the cluster.
* The cluster scales by adding additional nodes.
* New nodes can be added to the cluster without downtime.
* Failed nodes and disks can be swapped out with no downtime

Swift is designed to have linear growth characteristics. The performance of the Swift does not degrade if the system grows its usage and the number of request increases. Swift is designed to grow as and when needed by adding storage nodes to increase storage capacity, adding proxy nodes requests increase, and growing network capacity where choke point are detected.

The objects are distributed in triplicate across the cluster to achieve the durability of swift. The object should be written in two of the three locations to be considered successful. Replicators runs to ensure that a sufficient number of copies are in the cluster. In case of failure, it ensures that three copies of the data is replicated throughout the cluster.

Another feature is the ability to define zones. Failure zones allow a cluster to be deployed across physical boundaries, each of which could individually fail. For example: a cluster could be deployed across several nearby data centers, enabling it to survive multiple datacenter failures.

Swift is designed ground up to handle failures so that reliability on the individual component level is less critical.

Swift URL: Swift has RESTful-API, hence all the communication are done over HTTP, using the HTTP verbs to signal the requested action.

Base: swift.example.com/v1/

Account: An account is determined by the authenticaton server when the account is created.

Container: Containers are namedapces used to group objects within an account.

Object: Objects are where the actual data is stored in swift. Object name may contain /, so pseudo-nested directories are possible.

To get a list of all containers in an account, use the GET command on the account:

**Swift components**

Following components enables swift to deliver high availability, high durability and high concurrency are:

**Proxy servers**: Handles all incoming AP1 requests.

* **Minimum two proxy server is deployed for redundancy purpose**

**Rings**: Maps logical names of data to location on particular disks.

The ring maintains this mapping using zones, devices, partitions, and replicas. Each partition in the ring is replicated three times by default across the cluster and the location for a partition are stored in the mapping maintained by the ring. The ring is also responsible for determining whch devices are used for handoff a failure occur.

**Zones**: Each zone isolates data from other zones. A failure in one zone doesn’t impact the rest of the cluster because data is replicated across the Zones.

**Account and Containers**: Each account and container are different data base that are distributed across the cluster. An account database contains the list of containers in that account. A container database contains the list of objects in the container.

**Objects**: The data itself

**Partitions**: A partition stores Objects, account databases and container databases. Its an intermediate bucket that help manage locations where data lives in the cluster.