Red Hat OpenStack Platform - Swift

Sachin[*]

May 18, 2016

1 Theory

- Swift allows users to store unstructured data(objects) with canonical names containing *three* part:
  - `/account`: Think `account` as a storage location and NOT user account. `account` stores meta-data of that account plus list of all containers in that account. `account` is analogous to `/home` directory which may holds multiple users.
  - `/account/container`: Think of container as a root directory of user(analogous to `/home/USER/`). Account can have many containers with no limit.
  - `/account/container/object`: This is actual file. User may start storing a single file(files are stored in container as an object), or hierarchical data like `/photos/alaska/magic-bus/me.jpg` as an object. Swift stores multiple copies of single object across physical locations to ensure the data reliability and availability.
- Remember that user do not have to know the actual location of the data. In-fact he never knows. He always access the data in the form of `/account/container/object`.

2 Handy commands

- *Note*: My Swift cluster is running on `192.168.8.80:8080`. Following examples has `account` with the name *wasteland*, `container` with the name *keys* and file(`object`) with the name *mykey.pem*. My `AuthURL` is *http://192.168.8.80:8080/auth/v1.0* and `PubicURL/StorageURL` is *http://192.168.8.80:8080/v1/AUTH_wasteland*
- Fetching info about Swift proxy. If you have access to Swift server, `Pubic-URL/StorageURL`, `Auth-Token`, & `account` can be fetched using following command,

```
swift stat -v
#                    StorageURL: http://192.168.8.80:8080/v1/AUTH_wasteland
#                    Auth Token: AUTH_tk968b0ae7947640be874af6cd897a2b1e
#                       Account: AUTH_wasteland
#                    Containers: 0
#                       Objects: 0
#                         Bytes: 0
# Containers in policy "default": 1
#    Objects in policy "default": 0
#      Bytes in policy "default": 0
#                 Accept-Ranges: bytes
#                   X-Timestamp: 1463138224.81309
#                    X-Trans-Id: tx1fe3ecbeb9f04fdc92287-005735e92c
#                  Content-Type: text/plain; charset=utf-8
```

[*]psachin@redhat.com

- New user can be added as follows,

```
# --- /etc/swift/proxy-server.conf ---
[filter:tempauth]
use = egg:swift#tempauth
# user_ACCOUNT_USER = PASSWORD [GROUP] <storage URL:8080>
user_wasteland_psachin = psachin .admin .reseller_admin

[app:proxy-server]
use = egg:swift#proxy
allow_account_management = true
account_autocreate = true
# --- File ends here ---

# Restart servers and Proxy
swift-init account start
swift-init container start
swift-init object start
swift-init proxy restart
```

- Managing container and object using `swift` command

  Set following environmental variables

```
# --- ~/.profile ---
export ST_AUTH=http://192.168.8.80:8080/auth/v1.0
export ST_USER=wasteland:psachin
export ST_KEY=psachin
# --- File ends here ---
```

  Source the file before executing any command

```
source ~/.profile
```

  Most of the time, no configuration is needed, if Swift is enabled during packstack. You can actually start from here.

```
# --- Create container: 'keys' ---
swift post keys
# Verify/list containers
swift list
# --- Upload an object to container ---
# Create a file
echo "746c1c636cebe7a888fd77688dbfc252" > mykey.pem
# Upload object-'mykey.pem' to container-'keys'
swift upload keys mykey.pem
# Verify the object
swift list keys
# --- Download object ---
swift download keys mykey.pem
# Download object with different name
swift download keys mykey.pem -o mykey2.pem
```

- Managing container and object using APIs(`curl` command)

```
# --- Get token ---
# Set authURL and publicURL
export authURL="http://192.168.8.80:8080/auth/v1.0/"
export publicURL="http://192.168.8.80:8080/v1/AUTH_wasteland"

curl -v \
     -H "X-Auth-User: wasteland:psachin" \
     -H "X-Auth-Key: psachin" \
     $authURL
```

```
# *   Trying 192.168.8.80...
# * Connected to 192.168.8.80 (192.168.8.80) port 8080 (#0)
# > GET /auth/v1.0/ HTTP/1.1
# > Host: 192.168.8.80:8080
# > User-Agent: curl/7.43.0
# > Accept: */*
# > X-Auth-User: wasteland:psachin
# > X-Auth-Key: psachin
# >
# < HTTP/1.1 200 OK
# < X-Storage-Url: http://192.168.8.80:8080/v1/AUTH_wasteland
# < X-Auth-Token-Expires: 82975
# < X-Auth-Token: AUTH_tk968b0ae7947640be874af6cd897a2b1e
# < Content-Type: text/html; charset=UTF-8
# < X-Storage-Token: AUTH_tk968b0ae7947640be874af6cd897a2b1e
# < Content-Length: 0
# < X-Trans-Id: tx9c1bef9065754dd9b68ec-005735c49d
# < Date: Fri, 13 May 2016 12:12:13 GMT
# <
# * Connection #0 to host 192.168.8.80 left intact

export token="AUTH_tk968b0ae7947640be874af6cd897a2b1e"

# Verify account access
curl -v \
     -H "X-Storage-Token: $token" \
     $publicURL

# *   Trying 192.168.8.80...
# * Connected to 192.168.8.80 (192.168.8.80) port 8080 (#0)
# > GET /v1/AUTH_wasteland HTTP/1.1
# > Host: 192.168.8.80:8080
# > User-Agent: curl/7.43.0
# > Accept: */*
# > X-Storage-Token: AUTH_tk968b0ae7947640be874af6cd897a2b1e
# >
# < HTTP/1.1 204 No Content
# < Content-Length: 0
# < Accept-Ranges: bytes
# < X-Account-Object-Count: 0
# < X-Account-Storage-Policy-Default-Bytes-Used: 0
# < X-Account-Storage-Policy-Default-Object-Count: 0
# < X-Timestamp: 1463138224.81309
# < X-Account-Bytes-Used: 0
# < X-Account-Container-Count: 0
# < Content-Type: text/plain; charset=utf-8
# < X-Account-Storage-Policy-Default-Container-Count: 0
# < X-Trans-Id: tx95142c218202459399c88-005735cac1
# < Date: Fri, 13 May 2016 12:38:25 GMT
# <
# * Connection #0 to host 192.168.8.80 left intact

# --- Create a container: 'keys' ---
curl -v \
     -H "X-Storage-Token: $token" \
     -X PUT $publicURL/keys

# *   Trying 192.168.8.80...
# * Connected to 192.168.8.80 (192.168.8.80) port 8080 (#0)
# > PUT /v1/AUTH_wasteland/keys HTTP/1.1
# > Host: 192.168.8.80:8080
# > User-Agent: curl/7.43.0
# > Accept: */*
# > X-Storage-Token: AUTH_tk968b0ae7947640be874af6cd897a2b1e
# >
# < HTTP/1.1 201 Created
# < Content-Length: 0
# < Content-Type: text/html; charset=UTF-8
# < X-Trans-Id: tx39b7aee463b64127adfe2-005735cb92
# < Date: Fri, 13 May 2016 12:41:54 GMT
# <
# * Connection #0 to host 192.168.8.80 left intact

# Verify container
curl -v \
     -H "X-Storage-Token: $token" \
     -X GET $publicURL/keys

# *   Trying 192.168.8.80...
# * Connected to 192.168.8.80 (192.168.8.80) port 8080 (#0)
# > GET /v1/AUTH_wasteland/keys HTTP/1.1
# > Host: 192.168.8.80:8080
# > User-Agent: curl/7.43.0
# > Accept: */*
# > X-Storage-Token: AUTH_tk968b0ae7947640be874af6cd897a2b1e
# >
# < HTTP/1.1 204 No Content
# < Content-Length: 0
# < X-Container-Object-Count: 0
# < Accept-Ranges: bytes
# < X-Storage-Policy: default
# < X-Container-Bytes-Used: 0
# < X-Timestamp: 1463138224.83257
# < Content-Type: text/html; charset=UTF-8
# < X-Trans-Id: tx05408e3d41c246ea930f5-005735cc21
# < Date: Fri, 13 May 2016 12:44:17 GMT
# <
# * Connection #0 to host 192.168.8.80 left intact

# --- Upload object to container ---
# Create a file
echo "746c1c636cebe7a888fd77688dbfc252" > mykey.pem

# Upload object-'mykey.pem' to container-'keys'
curl -v \
     -H "X-Storage-Token: $token" \
     -X PUT $publicURL/keys/mykey.pem -T mykey.pem

# *   Trying 192.168.8.80...
# * Connected to 192.168.8.80 (192.168.8.80) port 8080 (#0)
# > PUT /v1/AUTH_wasteland/keys/mykey.pem HTTP/1.1
# > Host: 192.168.8.80:8080
# > User-Agent: curl/7.43.0
# > Accept: */*
# > X-Storage-Token: AUTH_tk968b0ae7947640be874af6cd897a2b1e
# > Content-Length: 43
# > Expect: 100-continue
# >
# < HTTP/1.1 100 Continue
# * We are completely uploaded and fine
# < HTTP/1.1 201 Created
# < Last-Modified: Fri, 13 May 2016 12:53:00 GMT
# < Content-Length: 0
# < Etag: 640ebd176639fb6ef9a3227770ee7b17
# < Content-Type: text/html; charset=UTF-8
# < X-Trans-Id: txf33923d6fbfe4523b4451-005735ce2b
# < Date: Fri, 13 May 2016 12:52:59 GMT
# <
# * Connection #0 to host 192.168.8.80 left intact

# Download an object
curl -v \
     -H "X-Storage-Token: $token" \
     -X GET $publicURL/keys/mykey.pem > mykey.pem

# *   Trying 192.168.8.80...
#   % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
#                                  Dload  Upload   Total   Spent    Left  Speed
    #   0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:-- 0* \
#                            Connected to 192.168.8.80 (192.168.8.80) port 8080 (#0)
# > GET /v1/AUTH_wasteland/keys/mykey.pem HTTP/1.1
# > Host: 192.168.8.80:8080
# > User-Agent: curl/7.43.0
# > Accept: */*
# > X-Storage-Token: AUTH_tk968b0ae7947640be874af6cd897a2b1e
# >
# < HTTP/1.1 200 OK
# < Content-Length: 43
# < Accept-Ranges: bytes
# < Last-Modified: Fri, 13 May 2016 12:53:00 GMT
# < Etag: 640ebd176639fb6ef9a3227770ee7b17
# < X-Timestamp: 1463143979.89953
# < Content-Type: application/octet-stream
# < X-Trans-Id: tx6b14a272331b4bc6937db-005735cef1
# < Date: Fri, 13 May 2016 12:56:17 GMT
# <
# { [43 bytes data]
# 100    43  100    43    0     0   2748      0 --:--:-- --:--:-- --:--:--  2866
# * Connection #0 to host 192.168.8.80 left intact
```

- Get statistics

  ```
  # To obtain Storage URL and Auth-Token
  swift stat -v

  # Get statistics of container and/or object
  swift stat [container]
  swift stat [container] [object]

  # Retrive capability of proxy
  swift capabilities

  # List container's details(Similar to 'ls -lh')
  swift list --lh [container]
  ```

- Object versioning

  When an object is overwritten, it's older version is lost, but there is a way we can store older version(s) of an object, no matter how many times is was overwritten.

  To enable object versioning, set `allow_versions` option to `true` in container configuration file.

```
# --- /etc/swift/container-server.conf ---
[app:container-server]
allow_versions = true
# --- File ends here ---

# Create 'archive' container to hold 'current' container's object versions
swift post archive

# Now create 'current' container with header 'X-Versions-Location'
# pointing to 'archive'
swift post current -H "X-Versions-Location: archive"

# --- Other similar ways(Optional) ---
# May also define content length at the time of creating a container
swift post archive -H "content-length: 0"
swift post current -H "content-length: 0" -H "X-Versions-Location: archive"

# And also specify Read ACL(World readable) during container creation
swift post -r ".r:*" archive -H "content-length: 0"
swift post -r ".r:*" current -H "content-length: 0" -H "X-Versions-Location: archive"
# --- xxx ---
```

3 Additional notes

Swift consistency processes:

- *Auditor*: Will walk through the storage nodes, read the data and the checksum, ensure the checksum matched with the database checksum. If the checksum didn't match, the data is moved to the Quarantine.
- *Replicator*: The replicator, will also scan each drive and ensures that the replicas of data is stored where is supposed to live. If it does not finds the data in that place(may be the data, due to corruption was moved to Quarantine), it will push the data to that place.

4 Slides notes

- Multiple HDD, where is my data store?
- HDD failure
- Storage problem
- Ownership of your data
- Access to data, HTTP, FTP, ReST
  - Mobile, Laptop..
- Swift
  - loosely tied to storage media
  - Scalable
  - Direct client access
- Terminology
  - Proxy: provides API access/ Coordinates requests to storage servers
  - Account: user namespace
  - Container: User defined segment of an account(root directory)
  - Object: Actual data
- Flow Proxy request -> Storage nodes(account, container, obj)

- Data placement
  - triple replication by default(as unique as possible)
  - Show Region/Zone pic
- Drive failures
  1. Umount failing drive
  2. Replicate/rebalance data
- Server failures
  1. Network, Power
  2. New data that is to be written will be placed elsewhere within a cluster/server
  3. Rebalancing happens
- Currupt data
  1. Stores checksum of the data with data itself
  2. Matches checksum of data periodically
     - If checksum doesnt match, the object is quarantined and the replication process rebalances the data/object
- Storage policies
  - Decide where you want to store data
    * Between swift clusters
    * Subset of hardware
  - Erasure coding <- Data availability policies
    * Based in frequency of access
- Erasure Codes
  - https://www.youtube.com/watch?v=kH3DXMKlEr8
  - https://www.youtube.com/watch?v=GDNK1S4FJBQ
- ACLs
  - Container ACL

    ```
    # World readable
    swift post -r ".r:*" photos

    # Allow .welcome.com but deny .noisy.com
    swift post -r ".r:*.welcome.com,.r:-noisy.com" photos

    # Enable object listing within a container
    swift post -r ".r:*,.rlistings" photos
    ```
  - Account ACL
- Hashing
  - Swift hashing function

```python
# Swift hashing is based on MD5
# hash(path) = md5(path + per-cluster suffix)

# Python snippet to know on which drive the object is stored. Say I
# have 4 drives
from hashlib import md5

m = md5()
m.update("/account/container/object")  # Hypothetical object path
```

```python
digest = m.hexdigest()
print(digest)

# hex to int
hex2int = int(digest, 16)
print(hex2int)
# digest % (number of drives) = Drive number
print(hex2int % 4)   # 2
```

5 Links

- https://gitlab.cee.redhat.com/psachin/bootcamp
- HTML version of this[1] doc is available at:
  https://gitlab.cee.redhat.com/psachin/bootcamp/blob/master/2016/scripts/notes.org
- Slides: https://redhat.slides.com/psachin/rhosp-swift-2016

[1]Made with Love, LaTeX, & GNU Emacs