Assignment 05

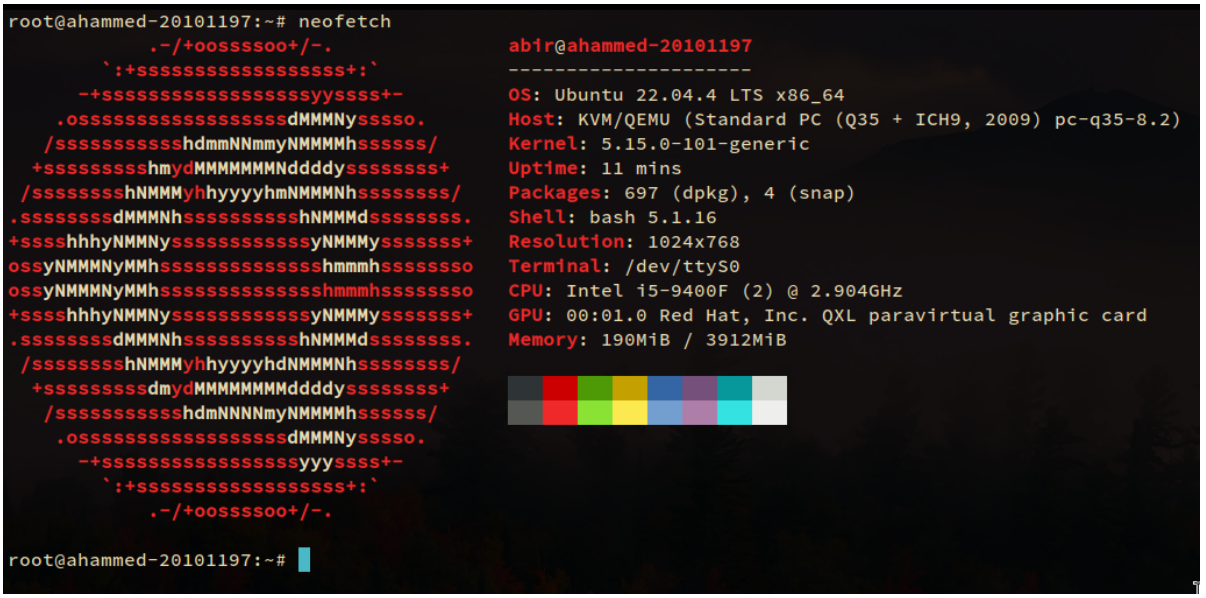
Abir Ahammed Bhuiyan
ID: 20101197
Dept. of Computer
Science and Engineering

Section : 01
Subject : CSE484
Semester : Spring-2024
Date : 16.04.2024

Installing OpenStack Swift SAIO

Some things to note before the installation:

- [x] 3 virtual hard disks were used each having 200MB
- [x] Part power is 3 because we have to create 8 partitions and replica count is also 3
- [x] `Ubuntu Server 22.04 LTS` was used for this experiment
- [x] Everything will be done under `root` or `superuser`



Installation

For installing `Swift SAIO` some packages must need to be in the system. Install them using the below command.

```
apt install curl gcc memcached rsync sqlite3 xfsprogs \
                    git-core libffi-dev python3-setuptools \
                    liberasurecode-dev libssl-dev python3-pip
```

```
root@ahammed-20101197:~# apt install curl gcc memcached rsync sqlite3 xfsprogs \
                    git-core libffi-dev python3-setuptools \
                    liberasurecode-dev libssl-dev
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Note, selecting 'git' instead of 'git-core'
xfsprogs is already the newest version (5.13.0-1ubuntu2).
xfsprogs set to manually installed.
curl is already the newest version (7.81.0-1ubuntu1.16).
curl set to manually installed.
git is already the newest version (1:2.34.1-1ubuntu1.10).
git set to manually installed.
python3-setuptools is already the newest version (59.6.0-1.2ubuntu0.22.04.1).
python3-setuptools set to manually installed.
rsync is already the newest version (3.2.7-0ubuntu0.22.04.2).
rsync set to manually installed.
The following additional packages will be installed:
  cpp cpp-11 gcc-11 gcc-11-base libasan6 libatomic1 libc-dev-bin libc-devtools libc6-d
  liblsan0 libmpc3 libnsl-dev libquadmath0 libtirpc-dev libtsan0 libubsan1 libxpm4 lin
Suggested packages:
  cpp-doc gcc-11-locales gcc-multilib make autoconf automake libtool flex bison gdb gc
  libanyevent-perl libcache-memcached-perl libmemcached libyaml-perl sqlite3-doc
The following NEW packages will be installed:
  cpp cpp-11 gcc gcc-11 gcc-11-base libasan6 libatomic1 libc-dev-bin libc-devtools lib
  libgcc-11-dev libgd3 libisl23 libitm1 liblsan0 libmpc3 libnsl-dev libquadmath0 libss
  sqlite3
0 upgraded, 34 newly installed, 0 to remove and 2 not upgraded.
Need to get 50.3 MB of archives.
After this operation, 163 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

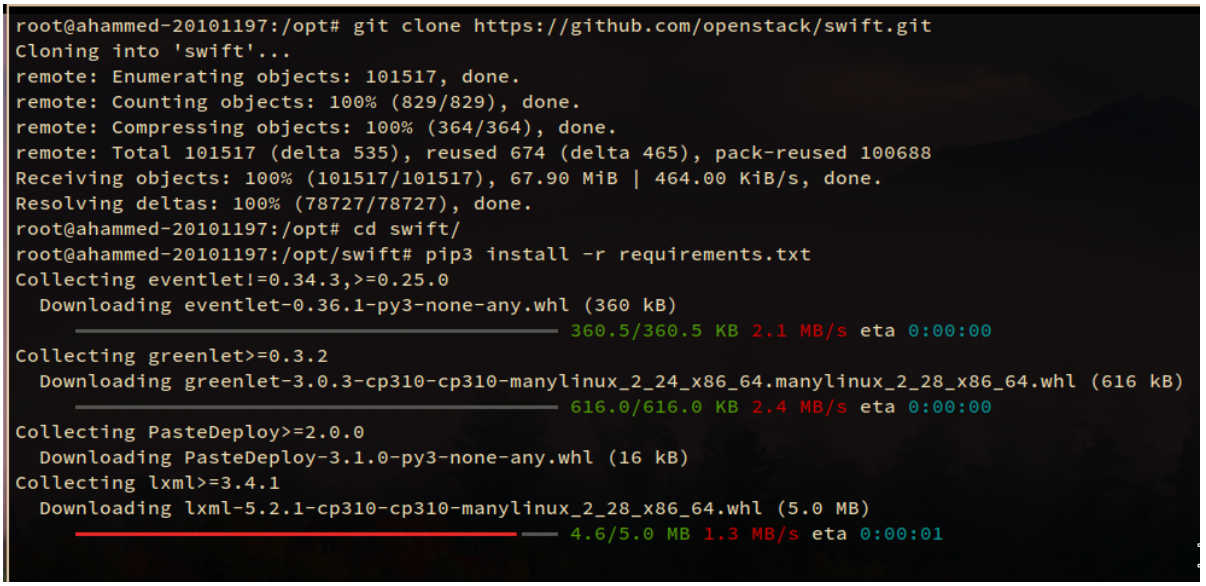
Next go to the `/opt` directory and clone the `python-swiftclient`. Go to `python-swiftclient` and install the `python3` dependencies and just install using the below commands.

```
cd /opt
git clone https://github.com/openstack/python-swiftclient.git
cd /opt/python-swiftclient
pip3 install -r requirements.txt
python3 setup.py install
cd ..
```

```
root@ahammed-20101197:/opt# git clone https://github.com/openstack/python-swiftclient.git
Cloning into 'python-swiftclient'...
remote: Enumerating objects: 6266, done.
remote: Counting objects: 100% (899/899), done.
remote: Compressing objects: 100% (281/281), done.
remote: Total 6266 (delta 646), reused 866 (delta 614), pack-reused 5367
Receiving objects: 100% (6266/6266), 3.36 MiB | 1.20 MiB/s, done.
Resolving deltas: 100% (4309/4309), done.
root@ahammed-20101197:/opt# cd python-swiftclient/
root@ahammed-20101197:/opt/python-swiftclient# pip3 install -r requirements.txt
Requirement already satisfied: requests>=2.4.0 in /usr/lib/python3/dist-packages (from -r requ
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behav
ttps://pip.pypa.io/warnings/venv
root@ahammed-20101197:/opt/python-swiftclient# python3 setup.py install
```

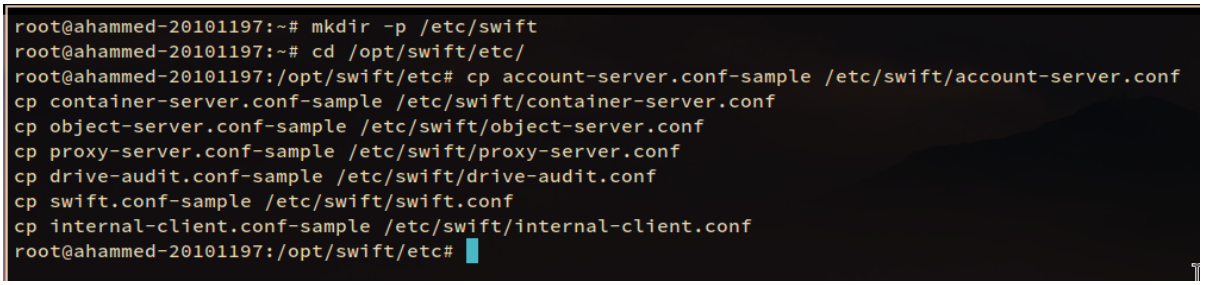
Then again clone `swift` under `/opt` and install the dependencies and install the program likewise.

```
cd /opt
git clone https://github.com/openstack/swift.git
cd /opt/swift
pip3 install -r requirements.txt
python3 setup.py install
cd ..
```



After installing `python-swiftclient` and `swift` we will create a directory `/etc/swift/`, in that directory we have to copy some configuration file from `/opt/swift/etc/` and paste it. Be careful to remove the `*-sample` extension of the copied files, every file in `/etc/swift/` should end with `*.conf`.

```
mkdir -p /etc/swift
cd /opt/swift/etc
cp account-server.conf-sample /etc/swift/account-server.conf
cp container-server.conf-sample /etc/swift/container-server.conf
cp object-server.conf-sample /etc/swift/object-server.conf
cp proxy-server.conf-sample /etc/swift/proxy-server.conf
cp drive-audit.conf-sample /etc/swift/drive-audit.conf
cp swift.conf-sample /etc/swift/swift.conf
cp internal-client.conf-sample /etc/swift/internal-client.conf
```



Formatting and mounting of virtual hard disks

Using `lsblk` we can see the hard disk connected and their names (in our case it is `vdb` `vdc` `vdd`). Format the hard disks using the below commands. [Note that the file system must be `xfs`]

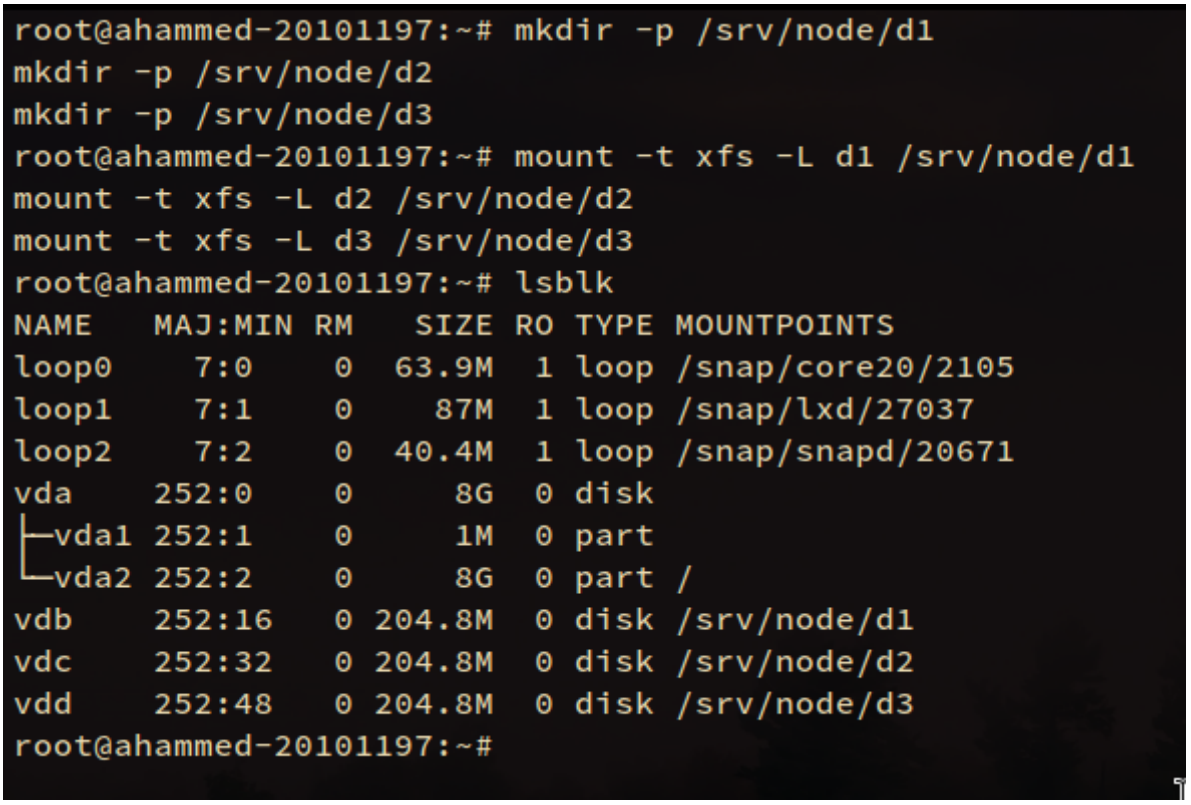
```
lsblk
mkfs.xfs -f -L d1 /dev/vdb
mkfs.xfs -f -L d2 /dev/vdc
mkfs.xfs -f -L d3 /dev/vdd
```

```
root@ahammed-20101197:~# lsblk
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
loop0    7:0    0  63.9M  1 loop /snap/core20/2105
loop1    7:1    0    87M  1 loop /snap/lxd/27037
loop2    7:2    0  40.4M  1 loop /snap/snapd/20671
vda    252:0    0     8G  0 disk
├─vda1 252:1    0     1M  0 part
└─vda2 252:2    0     8G  0 part /
vdb    252:16   0 204.8M  0 disk
vdc    252:32   0 204.8M  0 disk
vdd    252:48   0 204.8M  0 disk
root@ahammed-20101197:~# mkfs.xfs -f -L d1 /dev/vdb
mkfs.xfs -f -L d2 /dev/vdc
mkfs.xfs -f -L d3 /dev/vdd
meta-data=/dev/vdb               isize=512    agcount=4, agsize=13108 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=1, sparse=1, rmapbt=0
         =                       reflink=1    bigtime=0 inobtcount=0
data     =                       bsize=4096   blocks=52429, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0, ftype=1
log      =internal log           bsize=4096   blocks=1368, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
Discarding blocks...Done.
meta-data=/dev/vdc               isize=512    agcount=4, agsize=13108 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=1, sparse=1, rmapbt=0
         =                       reflink=1    bigtime=0 inobtcount=0
data     =                       bsize=4096   blocks=52429, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0, ftype=1
log      =internal log           bsize=4096   blocks=1368, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
Discarding blocks...Done.
meta-data=/dev/vdd               isize=512    agcount=4, agsize=13108 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=1, sparse=1, rmapbt=0
         =                       reflink=1    bigtime=0 inobtcount=0
data     =                       bsize=4096   blocks=52429, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0, ftype=1
log      =internal log           bsize=4096   blocks=1368, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
Discarding blocks...Done.
root@ahammed-20101197:~#
```

Next create three folder in `/srv/node` like shown below and mount the formatted hard disks on those directory.

```
mkdir -p /srv/node/d1
mkdir -p /srv/node/d2
mkdir -p /srv/node/d3
```

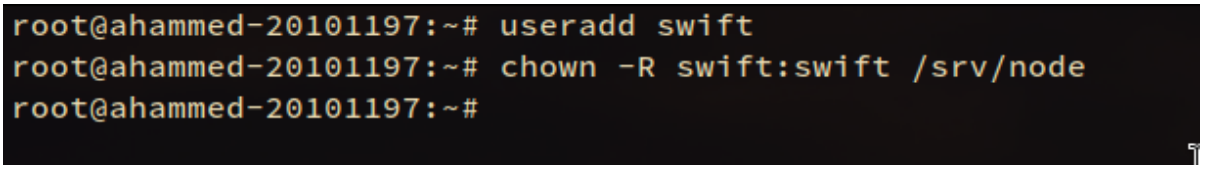
```
mount -t xfs -L d1 /srv/node/d1
mount -t xfs -L d2 /srv/node/d2
mount -t xfs -L d3 /srv/node/d3
```



Creation of a new user `swift`

For our installed `swift` program to work the `/srv/node` directories ownership should be `swift`. Because of that reason we have to create a new user `swift`. After creating `swift` user let's change ownership to `swift:swift` each and every directory under `/srv/node` using the below commands.

```
useradd swift
chown -R swift:swift /srv/node
```



Configurations of Swift

The first important thing after installation and setting up hard disk is to create a ring builder. We should create ring builder file for `account` `container` and `object` seperately.

> swift-ring-builder (account|container|object|object-n).builder create <part_power> <min_part_hours>

```
cd /etc/swift
swift-ring-builder account.builder create 3 3 1
swift-ring-builder container.builder create 3 3 1
swift-ring-builder object.builder create 3 3 1
```

After creating the builder files we need to add our mounted drives to the builder files.

> swift-ring-builder (account|container|object).builder add -:/

One thing to remember is that the default port for account, container and object should be the values shown in the table [if you try to use any arbitrary port then the port must also be changed in the `*.conf` files located at `/etc/swift/`],

Name	Port
account	6202
container	6201
object	6200

```
swift-ring-builder account.builder add r1z1-127.0.0.1:6202/d1 100
swift-ring-builder container.builder add r1z1-127.0.0.1:6201/d1 100
swift-ring-builder object.builder add r1z1-127.0.0.1:6200/d1 100

swift-ring-builder account.builder add r1z2-127.0.0.1:6202/d2 100
swift-ring-builder container.builder add r1z2-127.0.0.1:6201/d2 100
swift-ring-builder object.builder add r1z2-127.0.0.1:6200/d2 100

swift-ring-builder account.builder add r1z3-127.0.0.1:6202/d3 100
swift-ring-builder container.builder add r1z3-127.0.0.1:6201/d3 100
swift-ring-builder object.builder add r1z3-127.0.0.1:6200/d3 100
```

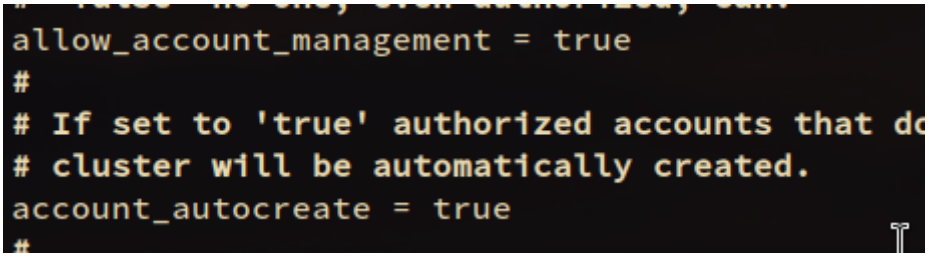
At last, we can finish the ring building by using `rebalance` command.

```
swift-ring-builder account.builder rebalance
swift-ring-builder container.builder rebalance
swift-ring-builder object.builder rebalance
```

```
root@ahammed-20101197:/etc/swift# swift-ring-builder account.builder create 3 3 1
swift-ring-builder container.builder create 3 3 1
swift-ring-builder object.builder create 3 3 1
root@ahammed-20101197:/etc/swift# swift-ring-builder account.builder add r1z1-127.0.0.1:6202/d1 100
swift-ring-builder container.builder add r1z1-127.0.0.1:6201/d1 100
swift-ring-builder object.builder add r1z1-127.0.0.1:6200/d1 100

swift-ring-builder account.builder add r1z2-127.0.0.1:6202/d2 100
swift-ring-builder container.builder add r1z2-127.0.0.1:6201/d2 100
swift-ring-builder object.builder add r1z2-127.0.0.1:6200/d2 100

swift-ring-builder account.builder add r1z3-127.0.0.1:6202/d3 100
swift-ring-builder container.builder add r1z3-127.0.0.1:6201/d3 100
swift-ring-builder object.builder add r1z3-127.0.0.1:6200/d3 100
Device d0r1z1-127.0.0.1:6202R127.0.0.1:6202/d1_"" with 100.0 weight got id 0
Device d0r1z1-127.0.0.1:6201R127.0.0.1:6201/d1_"" with 100.0 weight got id 0
Device d0r1z1-127.0.0.1:6200R127.0.0.1:6200/d1_"" with 100.0 weight got id 0
Device d1r1z2-127.0.0.1:6202R127.0.0.1:6202/d2_"" with 100.0 weight got id 1
Device d1r1z2-127.0.0.1:6201R127.0.0.1:6201/d2_"" with 100.0 weight got id 1
Device d1r1z2-127.0.0.1:6200R127.0.0.1:6200/d2_"" with 100.0 weight got id 1
Device d2r1z3-127.0.0.1:6202R127.0.0.1:6202/d3_"" with 100.0 weight got id 2
Device d2r1z3-127.0.0.1:6201R127.0.0.1:6201/d3_"" with 100.0 weight got id 2
Device d2r1z3-127.0.0.1:6200R127.0.0.1:6200/d3_"" with 100.0 weight got id 2
root@ahammed-20101197:/etc/swift# swift-ring-builder account.builder rebalance
swift-ring-builder container.builder rebalance
swift-ring-builder object.builder rebalance
Reassigned 24 (300.00%) partitions. Balance is now 0.00.  Dispersion is now 0.00
Reassigned 24 (300.00%) partitions. Balance is now 0.00.  Dispersion is now 0.00
Reassigned 24 (300.00%) partitions. Balance is now 0.00.  Dispersion is now 0.00
root@ahammed-20101197:/etc/swift#
```

Another important thing is that we have to uncomment and change the values of the following to `true` in `/etc/swift/proxy-server.conf`.

```
allow_account_management = true
account_autocreate = true
```



Configuration of the Swift log

By running the below commands and restarting the `rsyslog` daemon again we can have our log for `swift` program that will help us to monitor the programs internal behaviour.

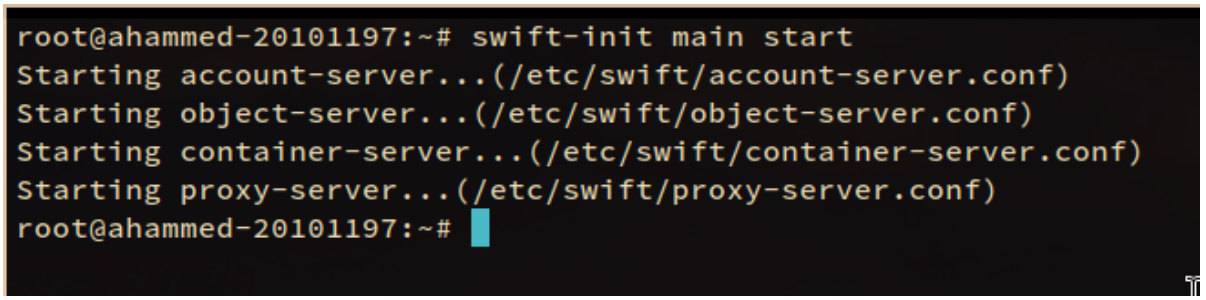
```
echo local0.* /var/log/swift/all0.log > /etc/rsyslog.d/0-swift.conf
mkdir /var/log/swift
chown -R syslog.adm /var/log/swift
chmod -R g+w /var/log/swift
systemctl restart rsyslog
```

```
root@ahammed-20101197:/etc/swift# echo local0.* /var/log/swift/all0.log > /etc/rsyslog.d/0-swift.conf
root@ahammed-20101197:/etc/swift# mkdir /var/log/swift
root@ahammed-20101197:/etc/swift# chown -R syslog.adm /var/log/swift
root@ahammed-20101197:/etc/swift# chmod -R g+w /var/log/swift
root@ahammed-20101197:/etc/swift# systemctl restart rsyslog
root@ahammed-20101197:/etc/swift#
```

Starting the servers

By using the below command all the main server processes will start and we will be able to use the `swift` object storage.

```
swift-init main start
```



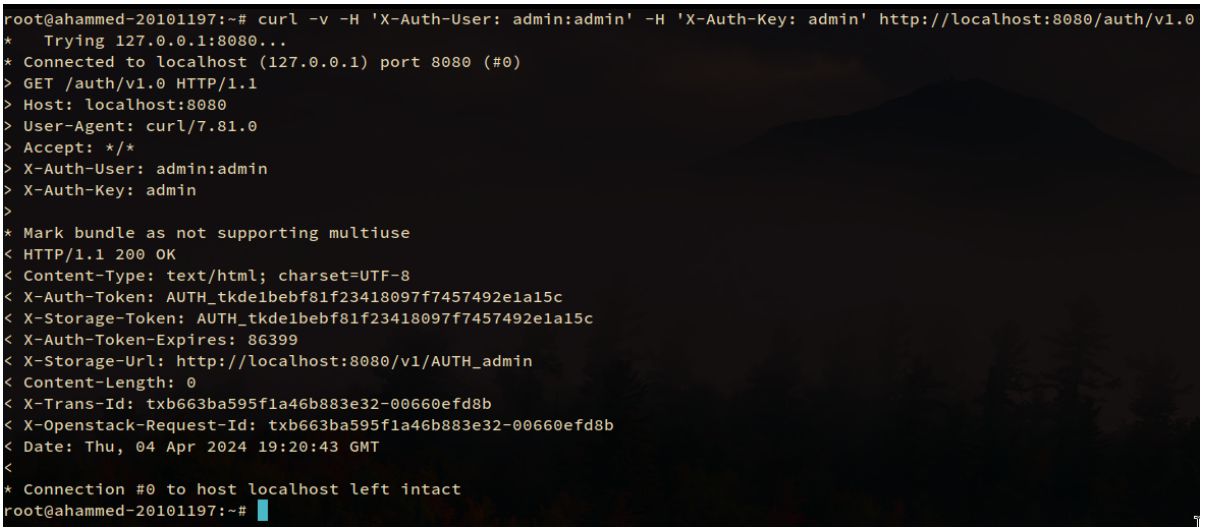
Test your authorization, authentication, and upload and download of a object successfully (Using both Curl command and Swift Client)

Using `cURL`

Authentication

For our use case we will be using `user=admin` `password=admin`. To authorize the admin we can fire a command like below. After this request the `proxy-server` will respond with `X-Auth-Token` and `X-Storage-Url`.

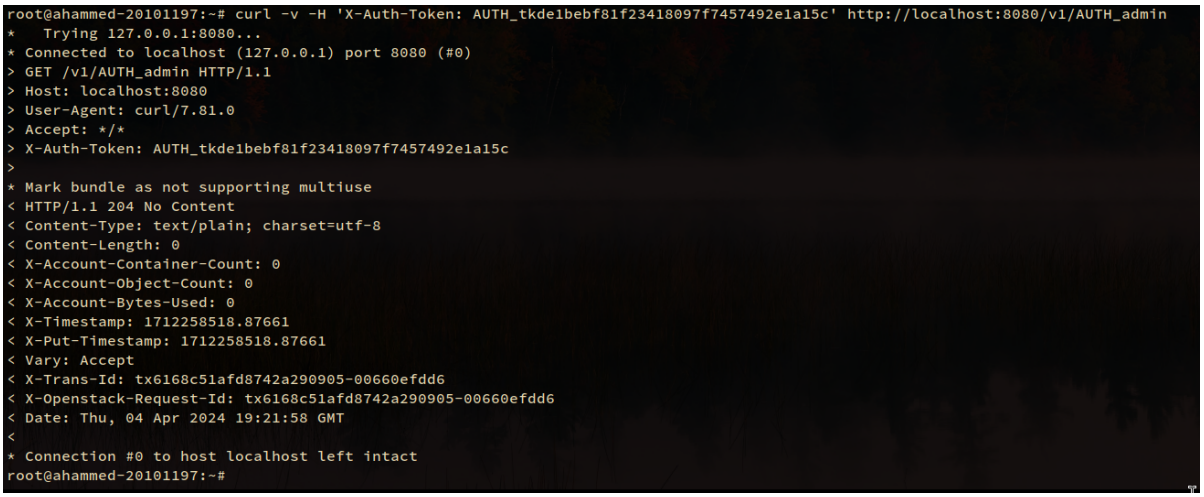
```
 curl -v -H 'X-Auth-User: admin:admin' -H 'X-Auth-Key: admin'
http://localhost:8080/auth/v1.0
```



Authorization

For authorization we must pass the `X-Auth-Token` in every request (GET, PUT etc). In the below command we are tyring to get the current state of the account database and we are getting `204 No Content`, this means the authentication is working.

```
curl -v -H 'X-Auth-Token: AUTH_tkde1bebf81f23418097f7457492e1a15c'
http://localhost:8080/v1/AUTH_admin
```

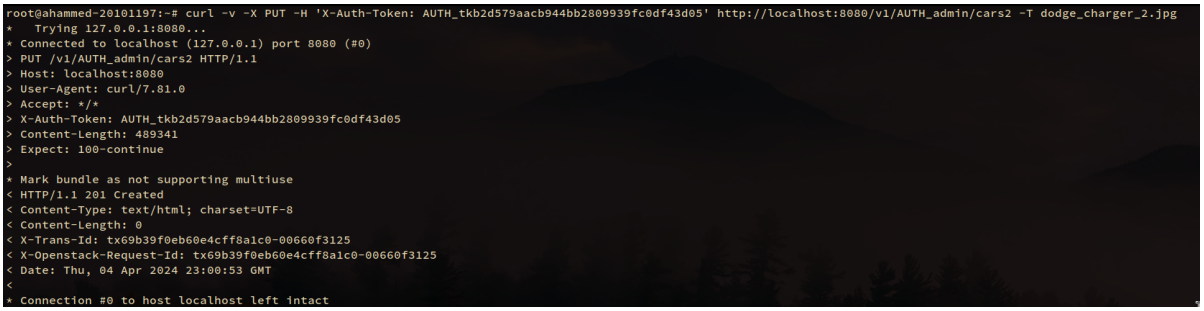


Upload

For uploading a file we can use `PUT` method along with the `X-Auth-Token`. Moreover, we have to specify the file we are trying to upload using `-T` flag and where the file will be uploaded (/account/container/object). In our case, the upload location is `AUTH_admin/cars2/dodge_charger_2.jpg`.

```
curl -v -X PUT -H 'X-Auth-Token: AUTH_tkb2d579aacb944bb2809939fc0df43d05'
http://localhost:8080/v1/AUTH_admin/cars2/dodge_charger_2.jpg -T
dodge_charger_2.jpg
```

After the request we will get `201 Created` response.



Download

For downloading a object we should use `GET` method along with `X-Auth-Token`. Also, we have to use `-o` file to save the output in a file.

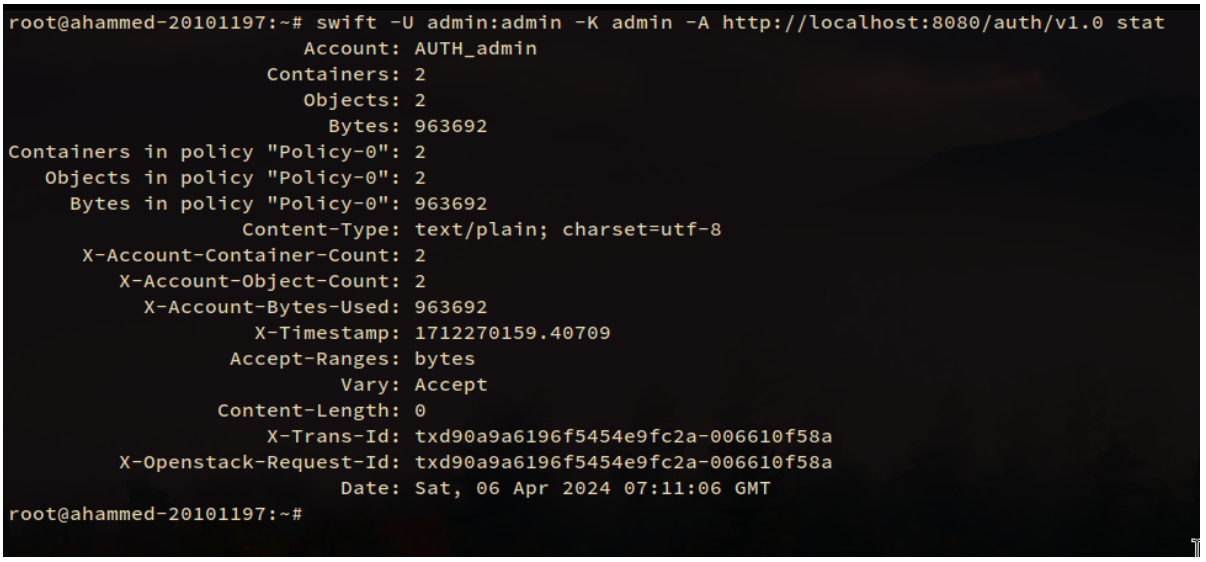
```
curl -v -X GET -H 'X-Auth-Token: AUTH_tk08d50ac1d6b74c139d667da72ed2eeb7'
http://localhost:8080/v1/AUTH_admin/cars2/dodge_charger_2.jpg -o hello.jpg
```

Using `Swift Client`

Authentication & Authorization

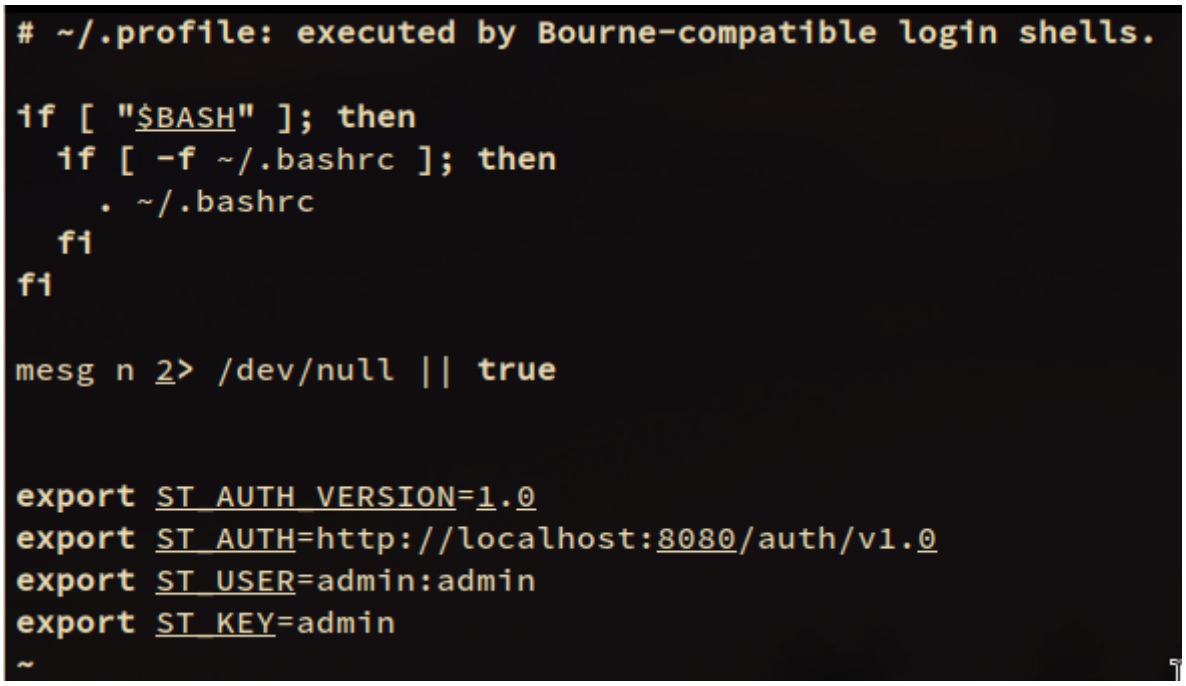
The `swift` command is popular is because it provides users with human-friendly verbs (`upload` instead of `PUT`) to use when communicating with a cluster. But one drawback with the Swift CLI is that is requires you to pass in your authentication information with each command (unlike `curl` we cannot pass in any token for authorization.

```
swift -U admin:admin -K admin -A http://localhost:8080/auth/v1.0 stat
```



With every command we have to pass `-U` and `-K` and `-A` flag, we can skip doing this by exporting some environment variable for `python-swiftclient` in `~/.profile`.

```
export ST_AUTH_VERSION=1.0
export ST_AUTH=http://localhost:8080/auth/v1.0
export ST_USER=admin:admin
export ST_KEY=admin
```



Upload

For uploading an object we can use a command like below.

```
swift upload cars dodge_charger_1.jpg
```

```
root@ahammed-20101197:~# swift stat
                        Account: AUTH_admin
                     Containers: 1
                        Objects: 1
                          Bytes: 489341
Containers in policy "Policy-0": 1
   Objects in policy "Policy-0": 1
     Bytes in policy "Policy-0": 489341
                   Content-Type: text/plain; charset=utf-8
      X-Account-Container-Count: 1
         X-Account-Object-Count: 1
           X-Account-Bytes-Used: 489341
                    X-Timestamp: 1712270159.40709
                  Accept-Ranges: bytes
                           Vary: Accept
                 Content-Length: 0
                     X-Trans-Id: tx177d1b5366a94b80aec75-00661133be
         X-Openstack-Request-Id: tx177d1b5366a94b80aec75-00661133be
                           Date: Sat, 06 Apr 2024 11:36:30 GMT
root@ahammed-20101197:~# swift upload cars dodge_charger_1.jpg
dodge_charger_1.jpg
root@ahammed-20101197:~# swift stat
                        Account: AUTH_admin
                     Containers: 2
                        Objects: 1
                          Bytes: 489341
Containers in policy "Policy-0": 2
   Objects in policy "Policy-0": 1
     Bytes in policy "Policy-0": 489341
                   Content-Type: text/plain; charset=utf-8
      X-Account-Container-Count: 2
         X-Account-Object-Count: 1
           X-Account-Bytes-Used: 489341
                    X-Timestamp: 1712270159.40709
                  Accept-Ranges: bytes
                           Vary: Accept
                 Content-Length: 0
                     X-Trans-Id: txcda3c0897ab44f5b8bbaa-00661133cb
         X-Openstack-Request-Id: txcda3c0897ab44f5b8bbaa-00661133cb
                           Date: Sat, 06 Apr 2024 11:36:43 GMT
root@ahammed-20101197:~#
```

Download

```
swift download cars dodge_charger_1.jpg
```

```
root@ahammed-20101197:/opt# swift download cars dodge_charger_1.jpg
dodge_charger_1.jpg [auth 0.003s, headers 0.016s, total 0.018s, 32.032 MB/s]
root@ahammed-20101197:/opt# ls
dodge_charger_1.jpg  python-swiftclient  swift
root@ahammed-20101197:/opt# file dodge_charger_1.jpg
dodge_charger_1.jpg: JPEG image data, JFIF standard 1.01, aspect ratio, density 1x1, segment length 16, commen
ion 8, 1920x1200, components 3
root@ahammed-20101197:/opt#
```

Test Other commands such as head/get/post etc. Test all the commands of Swift Client.

`python-swiftclient` upload, download are equivalent to get and post request method, those have been shown in the previous section, in this section we will focus on the rest of the commands.

command	HTTP method	details
`delete`	`DELETE`	Delete a container or objects within a container
`download`	`GET`	Download objects from containers
`list`	`GET`	List containers in an account or objects in a container.
`post`	`POST`	Update metadata for account, container, or object (may also be used to create a container)
`stat`	`HEAD`	Display header information for account, container, or object
`upload`	`PUT`	Upload files or directories to a container
`capabilities`	`GET`	List cluster capabilities
`auth`	`GET`	Display auth related environment variables

`stat`

Displays information for the account, container, or object depending on the arguments given (if any). In verbose mode, the storage URL and the authentication token are displayed as well.

```
root@ahammed-20101197:~# swift stat
                     Account: AUTH_admin
                  Containers: 2
                     Objects: 2
                       Bytes: 963692
Containers in policy "Policy-0": 2
   Objects in policy "Policy-0": 2
     Bytes in policy "Policy-0": 963692
                Content-Type: text/plain; charset=utf-8
      X-Account-Container-Count: 2
         X-Account-Object-Count: 2
           X-Account-Bytes-Used: 963692
                  X-Timestamp: 1712270159.40709
              Accept-Ranges: bytes
                       Vary: Accept
             Content-Length: 0
                 X-Trans-Id: tx03e60f9336474be383920-00661138a3
         X-Openstack-Request-Id: tx03e60f9336474be383920-00661138a3
                       Date: Sat, 06 Apr 2024 11:57:23 GMT
root@ahammed-20101197:~#
```

`list`

Lists the containers for the account or the objects for a container.

```
root@ahammed-20101197:~# swift list
cars
cars2
root@ahammed-20101197:~# swift list cars2
dodge_charger_2.jpg
root@ahammed-20101197:~# swift list cars
dodge_charger_1.jpg
root@ahammed-20101197:~#
```

`auth`

Display authentication variables in shell friendly format.

```
root@ahammed-20101197:~# swift auth
export OS_STORAGE_URL=http://localhost:8080/v1/AUTH_admin
export OS_AUTH_TOKEN=AUTH_tk3634197e703a4a7eb81fbd3063e106b7
root@ahammed-20101197:~#
```

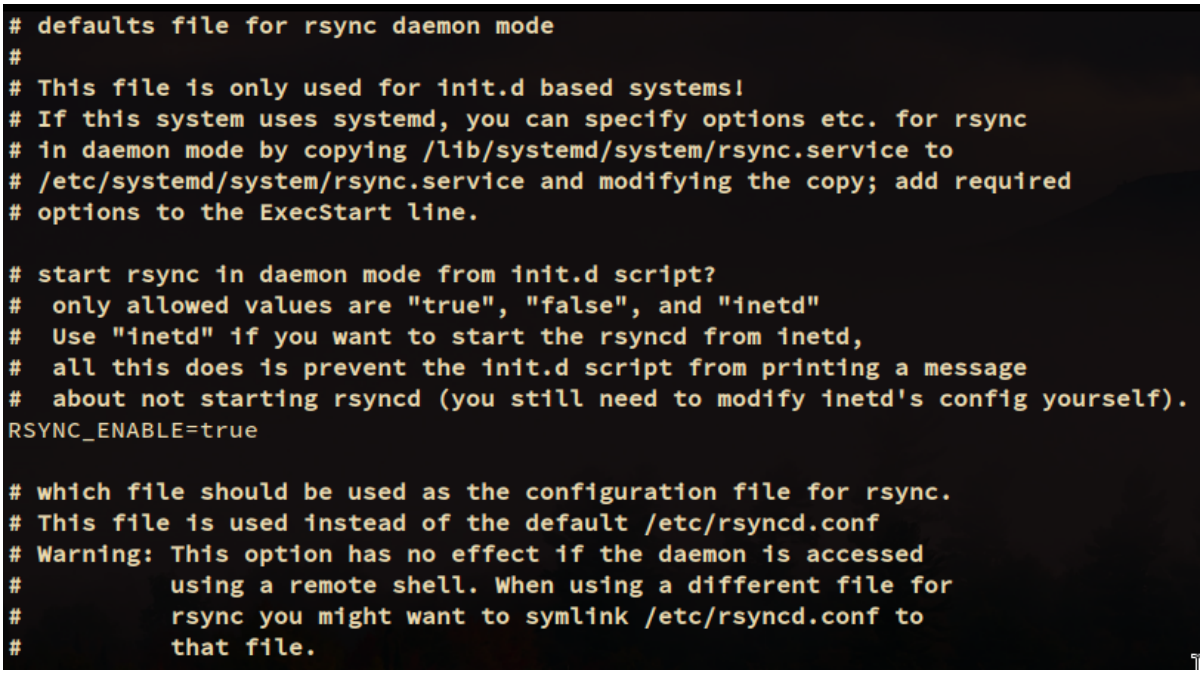
`capabilities`

Displays cluster capabilities. The output includes the list of the activated Swift middlewares as well as relevant options for each ones.

```
root@ahammed-20101197:/opt# swift capabilities
Core: swift
 Options:
  account_autocreate: True
  account_listing_limit: 10000
  allow_account_management: True
  container_listing_limit: 10000
  extra_header_count: 0
  max_account_name_length: 256
  max_container_name_length: 256
  max_file_size: 5368709122
  max_header_size: 8192
  max_meta_count: 90
  max_meta_name_length: 128
  max_meta_overall_size: 4096
  max_meta_value_length: 256
  max_object_name_length: 1024
  policies: [{'name': 'Policy-0', 'aliases': 'Policy-0, ye
  strict_cors_mode: True
  version: 2.34.0.dev14
Additional middleware: account_quotas
Additional middleware: bulk_delete
 Options:
  max_deletes_per_request: 10000
  max_failed_deletes: 1000
Additional middleware: bulk_upload
 Options:
  max_containers_per_extraction: 10000
  max_failed_extractions: 1000
Additional middleware: container_quotas
Additional middleware: container_sync
 Options:
  realms: {}
Additional middleware: ratelimit
 Options:
  account_ratelimit: 0.0
  container_listing_ratelimits: []
  container_ratelimits: []
  max_sleep_time_seconds: 60.0
root@ahammed-20101197:/opt# swift capabilities
Core: swift
 Options:
  account_autocreate: True
  account_listing_limit: 10000
  allow_account_management: True
  container_listing_limit: 10000
  extra_header_count: 0
```

Test replication is working in your system

Replication to work first we have to make the value of `RSYNC_ENABLE` to `true` in `/etc/default/rsync`.



```
# defaults file for rsync daemon mode
#
# This file is only used for init.d based systems!
# If this system uses systemd, you can specify options etc. for rsync
# in daemon mode by copying /lib/systemd/system/rsync.service to
# /etc/systemd/system/rsync.service and modifying the copy; add required
# options to the ExecStart line.

# start rsync in daemon mode from init.d script?
#  only allowed values are "true", "false", and "inetd"
#  Use "inetd" if you want to start the rsyncd from inetd,
#  all this does is prevent the init.d script from printing a message
#  about not starting rsyncd (you still need to modify inetd's config yourself).
RSYNC_ENABLE=true

# which file should be used as the configuration file for rsync.
# This file is used instead of the default /etc/rsyncd.conf
# Warning: This option has no effect if the daemon is accessed
#          using a remote shell. When using a different file for
#          rsync you might want to symlink /etc/rsyncd.conf to
#          that file.
```

After that create a file `/etc/rsyncd.conf` and paste the below configuration snippet there.

```
uid = swift
gid= swift
log file = /var/log/rsyncd.log
pid file = /var/run/rsyncd.pid
[account]
max connections = 25
path = /srv/node/
read only = false
lock file = /var/lock/account.lock
[container]
max connections = 25
path = /srv/node/
read only = false
lock file = /var/lock/container.lock
[object]
max connections = 25
path = /srv/node/
read only = false
lock file = /var/lock/object.lock
```

After that just restart the `rsync` using the below command.

```
systemctl restart rsync
```

Use the below command to test if the `rsync` is working fine or not.

```
rsync localhost::
```

```
root@ahammed-20101197:~# sudo systemctl restart rsync
root@ahammed-20101197:~# sudo systemctl status rsync
● rsync.service - fast remote file copy program daemon
     Loaded: loaded (/lib/systemd/system/rsync.service; disabled; vendor preset: enabled)
     Active: active (running) since Thu 2024-04-04 22:24:42 UTC; 9s ago
       Docs: man:rsync(1)
             man:rsyncd.conf(5)
   Main PID: 1009 (rsync)
      Tasks: 1 (limit: 4558)
     Memory: 1.3M
        CPU: 14ms
     CGroup: /system.slice/rsync.service
             └─1009 /usr/bin/rsync --daemon --no-detach

Apr 04 22:24:42 ahammed-20101197 systemd[1]: Started fast remote file copy program daemon.
root@ahammed-20101197:~# rsync localhost::
account
container
object
root@ahammed-20101197:~#
```

Use the below command to start all the `server-processes` and `consistencey-processes`.

```
swift-init all start
```

```
root@ahammed-20101197:~# swift-init all start
Starting container-sharder...(/etc/swift/container-server.conf)
Starting account-server...(/etc/swift/account-server.conf)
Starting container-replicator...(/etc/swift/container-server.conf)
Starting container-updater...(/etc/swift/container-server.conf)
Starting container-auditor...(/etc/swift/container-server.conf)
Starting object-reconstructor...(/etc/swift/object-server.conf)
Starting object-expirer...(/etc/swift/object-server.conf)
Starting proxy-server...(/etc/swift/proxy-server.conf)
Starting object-auditor...(/etc/swift/object-server.conf)
Unable to locate config for container-reconciler
Starting object-replicator...(/etc/swift/object-server.conf)
Starting account-reaper...(/etc/swift/account-server.conf)
Starting container-sync...(/etc/swift/container-server.conf)
Starting object-updater...(/etc/swift/object-server.conf)
Starting account-replicator...(/etc/swift/account-server.conf)
Starting object-server...(/etc/swift/object-server.conf)
Starting account-auditor...(/etc/swift/account-server.conf)
Starting container-server...(/etc/swift/container-server.conf)
root@ahammed-20101197:~#
```

To test if the repllication is working or not, we can do an experiment like the picture below. There we have first enlisted all the available `objects` or `*.data`. After, that we have just removed everything inside the `/srv/node/d1/`. Then again we enlist and found that `./d1` directory along with objects are missing. But after a few seconds if we try to enlist again we will see that `./d1` is back with the data similar to `./d2` and `./d3`. This proves our replication is working as it should.

BONUS

a) After installation of the object storage, setup/find out the way to measure different performance metrics. What are the measurement metrics?

After setting up OpenStack Swift object storage, we can measure various performance metrics to assess the object storage's performance and health. Here are some key measurement metrics we can consider,

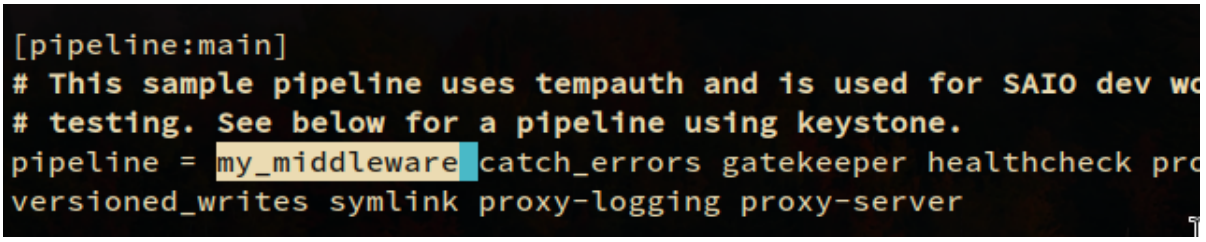
- **Latency**: Measure the time taken for requests to be processed by the Swift storage system. This includes the time taken for read, write, delete, and other operations. Lower latency indicates faster performance.
- **Throughput**: Calculate the rate at which data can be transferred to and from the Swift storage system. Throughput is usually measured in `bytes per second (Bps)` or `operations per second (ops/sec)`.
- **Concurrency**: Evaluate the system's ability to handle multiple simultaneous requests or users. Measure the number of concurrent connections or operations the system can support without a significant decrease in performance.
- **Availability**: Monitor the uptime and availability of the Swift storage system. Calculate the percentage of time the system is operational and accessible to users. Availability is typically measured as a percentage over a specific time period (e.g., 99.9% uptime).
- **Durability**: Assess the reliability and durability of data stored in the Swift storage system. Measure data durability by calculating the probability of data loss or corruption over time. Swift uses replication and erasure coding for data durability.
- **Scalability**: Determine the system's ability to scale horizontally and vertically to accommodate increasing storage capacity and workload demands. Measure scalability by monitoring performance under varying load levels and increasing storage requirements.
- **Error Rates**: Track the frequency of errors and failures occurring in the Swift storage system. Monitor error rates for operations such as read/write failures, server errors, network errors, and authentication errors.
- **Storage Efficiency**: Evaluate the efficiency of storage utilization and optimization techniques employed by Swift, such as compression, deduplication, and tiering. Measure storage efficiency by comparing the actual storage used to the total storage capacity allocated.

- **Network Performance**: Assess the performance of the network infrastructure supporting the Swift storage system. Measure network latency, bandwidth utilization, packet loss, and throughput between clients and Swift nodes.
- **API Performance**: Evaluate the performance of the Swift API endpoints used for interacting with the storage system. Measure API response times, throughput, and error rates for various API operations.

b) Write a new middleware in OpenStack Swift that can do something new!!!

We are going to code a middleware that will add this `X-Hello: Hello from eniac00!!!` header in the response body of every requests.

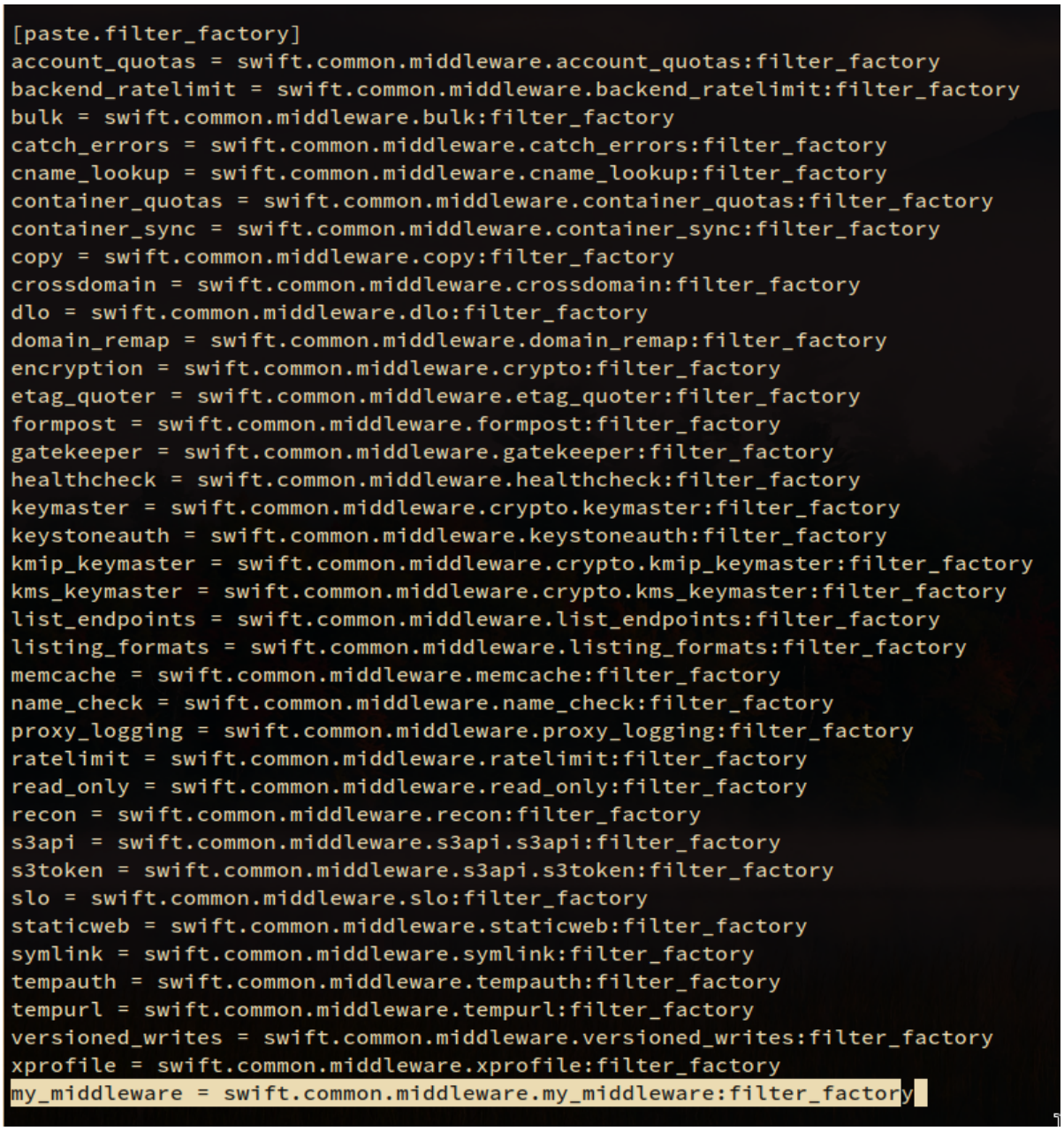
To achieve that, we have to append the `pipeline` with our middleware name in `/etc/swift/proxy-server.conf`. In our case our middleware name is `my_middleware` and we have to appended that name in the `pipeline` section like the image below.



Moreover, in `/etc/swift/proxy-server.conf` we have to add a `filter` section like below.

```
[filter:my_middleware]
use = egg:swift#my_middlewared
```

After doing that we have to let the `entry_points.txt` know of our newly created middleware, for this we have to add a line under `[paste.filter_factory]` section located in `/usr/local/lib/python3.10/dist-packages/swift-2.34.0.dev14.egg-info/entry_points.txt` file like below.



After that we have to code our middleware. We should create our middleware in `/usr/local/lib/python3.10/dist-packages/swift/common/middleware` directory. Let's create a file in that directory and the name of the file should be `my_middleware.py`. Now we have to write the below code in that `my_middleware.py` and save the file.

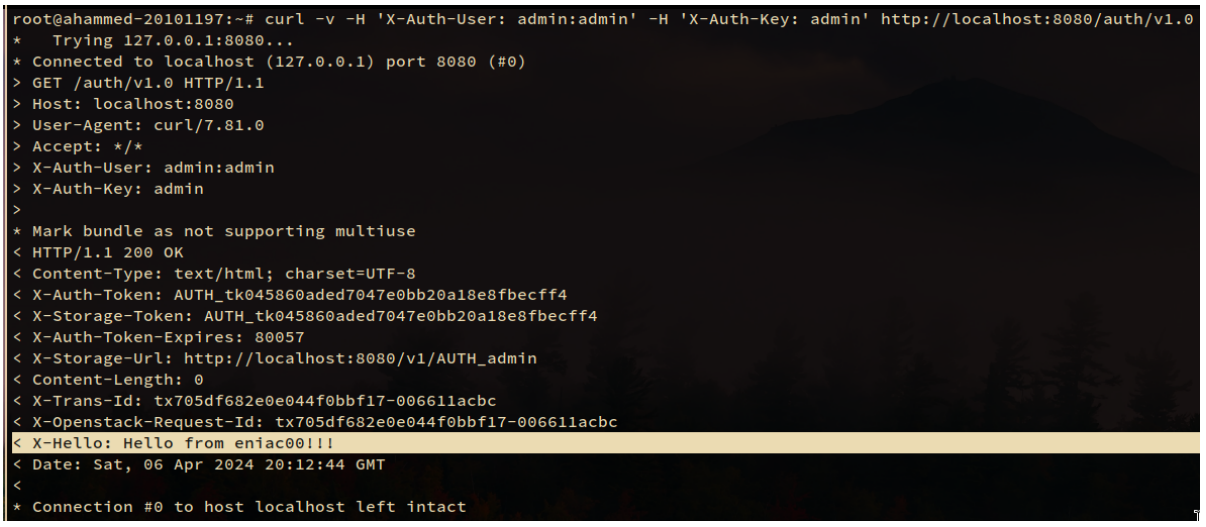
```
from swift.common.swob import Response
from swift.common.swob import wsgify

class SwiftSampleMiddlware(object):
    def __init__(self, app, conf):
        self.app = app

    @wsgify
    def __call__(self, req):
        resp = req.get_response(self.app)
        resp.headers['X-Hello'] = "Hello from eniac00!!!"
        return resp

def filter_factory(global_conf, **local_conf):
    conf = global_conf.copy()
```

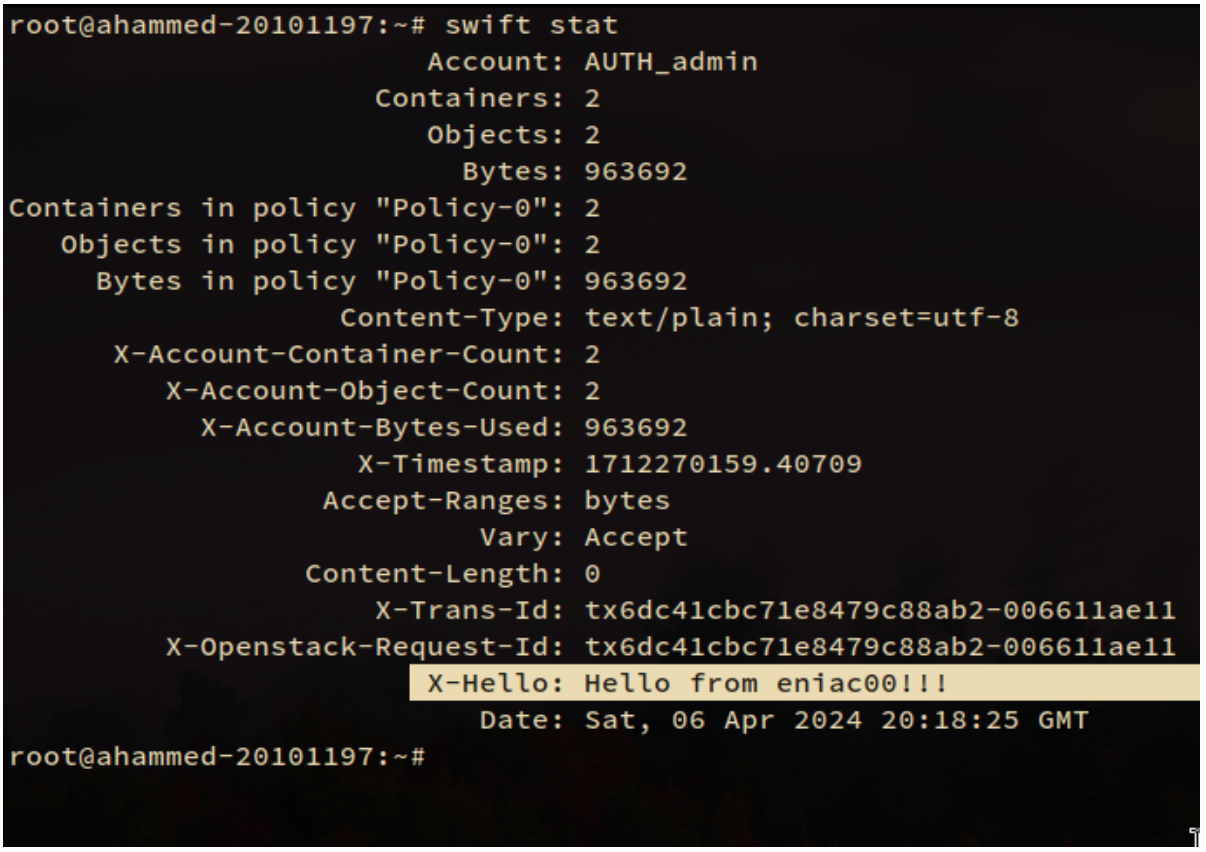
```
    conf.update(local_conf)

    def sample_filter(app):
        return SwiftSampleMiddlware(app, conf)
    return sample_filter
```

Next, we have to restart the whole swift processes using the below command.

```
swift-init all restart
```

Let's see if our middleware is functioning or not by firing any request command using `cURL` or `python-swiftclient`.





From the images above we are seeing that both of the responses contain a header named `X-Hello: Hello from eniac00!!!`. This means our middleware is working as it should.