Table of Contents

Installazione e configurazione del servizio Object Storage (Swift)

Installazione e configurazione di Storage node

- Installare i pacchetti di Storage node

```
# yum install openstack-swift-account openstack-swift-container openstack-swift-object xfs
```

- Creare la directory di configurazione e settate i proprietari:

```
# mkdir -p /etc/swift
# chown -R swift:swift /etc/swift/
```

- Creare il file /etc/swift/swift.conf:

```
[swift-hash]
# random unique string that can never change (DO NOT LOSE)
swift_hash_path_suffix = fLIbertYgibbitZ
```

Creare e montare un volume logico attraverso i seguenti passi:

1. Creare un volume logico di 800 Gigabyte sul server:

   ```
   # lvcreate -L 800G -n lv_swift <VOLUME_NAME>
   ```

   dove <VOLUME_NAME> è il nome del volume fisico sul quale si intende creare il volume logico.
2. Formattare la partizione appena creata su filesystem xfs con blocchi da 1024 byte:

   ```
   # mkfs.xfs -i size=1024 <LV_PATH>
   ```

   dove <LV_PATH> è il percorso del volume logico ricavabile tramite il comando `lvdisplay`.
3. Creare la directory dove si intende montare il volume appena creato:

   ```
   # mkdir -p /srv/node/swift
   ```
4. Per consentire che il volume logico sia montato all'avvio del server, editare il file `/etc/fstab` aggiungendo la seguente riga:

   ```
   <LV_PATH>       /srv/node/swift       xfs    noatime,nodiratime,nobarrier,logbufs=8    0 0
   ```
5. Montare il volume:

   ```
   # mount  a
   ```
6. Settare i proprietari della directory superiore:

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

- Creare il file di configuarazione per rsync /etc/rsyncd.conf:

```
uid = swift
gid = swift
log file = /var/log/rsyncd.log
pid file = /var/run/rsyncd.pid
address = <STORAGE_LOCAL_NET_IP>

[account]
max connections = 2
path = /srv/node/
read only = false
lock file = /var/lock/account.lock

[container]
max connections = 2
```

```
path = /srv/node/
read only = false
lock file = /var/lock/container.lock

[object]
max connections = 2
path = /srv/node/
read only = false
lock file = /var/lock/object.lock
```

Dove <STORAGE_LOCAL_NET_IP> è l'indirizzo IP dello Storage node.

- Editare la seguente riga nel file /etc/default/rsync:

```
RSYNC_ENABLE = true
```

- Creare uno script /etc/rc.d/init.d/rsyncd che possa far partire il servizio rsync:

```
#!/bin/bash

# Source function library.
. /etc/rc.d/init.d/functions

[ -f /usr/bin/rsync ] || exit 0

case "$1" in
start)
action "Starting rsyncd: " /usr/bin/rsync --daemon
;;
stop)
action "Stopping rsyncd: " killall rsync
;;
*)
echo "Usage: rsyncd {start|stop}"
exit 1
esac
exit 0
```

- Settare il default SELinux file context per lo script appena creato:

```
# restorecon -R -v /etc/rc.d/init.d/rsyncd
```

- Lanciare rsyncd attraverso il nuovo script creato:

```
# service rsyncd start
```

Nota bene: non è possibile usare il comando `chkconfig` per far partire il servizio rsyncd in automatico all'avvio del server, quindi è necessario farlo partire manualmente ad ogni riavvio.

- Creare il file /etc/swift/account-server.conf:

```
[DEFAULT]
bind_ip = <STORAGE_LOCAL_NET_IP>
workers = 1

[pipeline:main]
pipeline = account-server

[app:account-server]
use = egg:swift#account

[account-replicator]
```

```
[account-auditor]

[account-reaper]
```

- Creare il file /etc/swift/container-server.conf:

```
[DEFAULT]
bind_ip = <STORAGE_LOCAL_NET_IP>
workers = 1

[pipeline:main]
pipeline = container-server

[app:container-server]
use = egg:swift#container

[container-replicator]

[container-updater]

[container-auditor]

[container-sync]
```

- Creare il file /etc/swift/object-server.conf:

```
[DEFAULT]
bind_ip = <STORAGE_LOCAL_NET_IP>
workers = 1

[pipeline:main]
pipeline = object-server

[app:object-server]
use = egg:swift#object

[object-replicator]

[object-updater]

[object-auditor]

[object-expirer]
```

Installazione e configurazione di Proxy server

- Installare i pacchetti del Proxy server

```
# yum install openstack-swift-proxy memcached
```

- Installare i pacchetti di python-keystone:

```
# yum install python-keystone python-keystoneclient
```

- Creare un certificato auto-firmato per SSL:

```
# cd /etc/swift
# openssl req -new -x509 -nodes -out cert.crt -keyout cert.key
```

- Far partire il servizio memcached

```
# service memcached restart
```

  Fare in modo che memcached parta all'avvio del server:

```
# chkconfig memcached on
```

- Create /etc/swift/proxy-server.conf:

```
[DEFAULT]
bind_port = 8080
user = <SWIFT_USER_NAME>

[pipeline:main]
pipeline = catch_errors healthcheck cache authtoken keystone proxy-server

[app:proxy-server]
use = egg:swift#proxy
account_autocreate = true

[filter:keystone]
paste.filter_factory = keystone.middleware.swift_auth:filter_factory
operator_roles = admin, swiftoperator

[filter:authtoken]
paste.filter_factory = keystone.middleware.auth_token:filter_factory
# Delaying the auth decision is required to support token-less
# usage for anonymous referrers ('.r:*').
delay_auth_decision = 1
service_port = 5000
service_host = <KEYSTONE_HOSTNAME>
auth_port = 35357
auth_host = <KEYSTONE_HOSTNAME>
auth_token = <ADMIN_TOKEN>
admin_token = <ADMIN_TOKEN>
auth_protocol = http

[filter:cache]
use = egg:swift#memcache
set log_name = cache

[filter:catch_errors]
use = egg:swift#catch_errors

[filter:healthcheck]
use = egg:swift#healthcheck
```

- Creare i ring per account, container e oggetti:

```
# cd /etc/swift
# swift-ring-builder account.builder create 7 1 24
# swift-ring-builder container.builder create 7 1 24
# swift-ring-builder object.builder create 7 1 24
```

Dove:

- ♦ Il primo valore numerico (7 nell'esempio) è calcolato come segue: considerata la potenza in base 2 appena superiore al valore della partizione di storage in Gigabyte diviso 10, il valore da riportare è l'esponente di tale potenza. Nell'esempio riportato la partizione è di 800 GB, 2^7 (128) è la potenza di 2 appena superiore a 800/10=80.
- ♦ Il secondo valore numerico (1 nell'esempio) è il numero di repliche di ogni oggetto. **Nota bene:** tale numero non può essere superiore al numero di nodi storage.
- ♦ Il terzo valore numerico (24 nell'esempio) indica che la prtizione può essere spostata una volta ogni 24 ore.

- **Su ognuno degli Storage node** lanciare i seguenti comandi:

```
swift-ring-builder account.builder add z<ZONE>-<STORAGE_LOCAL_NET_IP>:6002/<DEVICE> <NUM>
swift-ring-builder container.builder add z<ZONE>-<STORAGE_LOCAL_NET_IP_1>:6001/<DEVICE> <N
swift-ring-builder object.builder add z<ZONE>-<STORAGE_LOCAL_NET_IP_1>:6000/<DEVICE> <NUM>
```

Per esempio sul nodo openstack-04.cnaf.infn.it si setta uno storage node con una partizione in Zona 1 sull'IP del server (131.154.100.113). Il mount point di questa partizione è /srv/node/swift, la partizione è il volume logico "swift", e 80 è il numero di Terabyte della partizione moltiplicato per 100. I comandi da lanciare sono:

```
# swift-ring-builder account.builder add z1-131.154.100.113:6002/swift 80
Device z1-131.154.100.113:6002/swift_"" with 80.0 weight got id 0
# swift-ring-builder container.builder add z1-131.154.100.113:6001/swift 80
Device z1-131.154.100.113:6001/swift_"" with 80.0 weight got id 0
# swift-ring-builder object.builder add z1-131.154.100.113:6000/swift 80
Device z1-131.154.100.113:6000/swift_"" with 80.0 weight got id 0
```

- Verificare il contenuto di ogni ring:

```
# swift-ring-builder account.builder
account.builder, build version 1
128 partitions, 1 replicas, 1 zones, 1 devices, 100.00 balance
The minimum number of hours before a partition can be reassigned is 24
Devices:    id  zone      ip address  port      name weight partitions balance meta
             0     1 131.154.100.113  6002     swift  80.00          0 -100.00

# swift-ring-builder container.builder
container.builder, build version 1
128 partitions, 1 replicas, 1 zones, 1 devices, 100.00 balance
The minimum number of hours before a partition can be reassigned is 24
Devices:    id  zone      ip address  port      name weight partitions balance meta
             0     1 131.154.100.113  6001     swift  80.00          0 -100.00

# swift-ring-builder object.builder
object.builder, build version 1
128 partitions, 1 replicas, 1 zones, 1 devices, 100.00 balance
The minimum number of hours before a partition can be reassigned is 24
Devices:    id  zone      ip address  port      name weight partitions balance meta
             0     1 131.154.100.113  6000     swift  80.00          0 -100.00
```

- Ri-bilanciare i ring:

```
# swift-ring-builder account.builder rebalance
Reassigned 128 (100.00%) partitions. Balance is now 0.00.

# swift-ring-builder container.builder rebalance
Reassigned 128 (100.00%) partitions. Balance is now 0.00.

# swift-ring-builder object.builder rebalance
Reassigned 128 (100.00%) partitions. Balance is now 0.00.
```

- Verificare nuovamente il contenuto di ogni ring:

```
# swift-ring-builder container.builder
container.builder, build version 1
128 partitions, 1 replicas, 1 zones, 1 devices, 0.00 balance
The minimum number of hours before a partition can be reassigned is 24
Devices:    id  zone      ip address  port      name weight partitions balance meta
             0     1 131.154.100.113  6001     swift  80.00        128    0.00

# swift-ring-builder object.builder
object.builder, build version 1
128 partitions, 1 replicas, 1 zones, 1 devices, 0.00 balance
The minimum number of hours before a partition can be reassigned is 24
Devices:    id  zone      ip address  port      name weight partitions balance meta
             0     1 131.154.100.113  6000     swift  80.00        128    0.00

# swift-ring-builder account.builder
account.builder, build version 1
128 partitions, 1 replicas, 1 zones, 1 devices, 0.00 balance
The minimum number of hours before a partition can be reassigned is 24
Devices:    id  zone      ip address  port      name weight partitions balance meta
             0     1 131.154.100.113  6002     swift  80.00        128    0.00
```

- Copiare i file account.ring.gz, container.ring.gz e object.ring.gz **su ognuno dei server Proxy e Storage** node in /etc/swift.

- Assicurarsi che i file di configurazione abbiano i seguenti proprietari:

```
chown -R swift:swift /etc/swift
```

- **Su tutti gli Storage node** far partire i servizi di Swift:

```
# swift-init object-server start
# swift-init object-replicator start
# swift-init object-updater start
# swift-init object-auditor start
# swift-init container-server start
# swift-init container-replicator start
# swift-init container-updater start
# swift-init container-auditor start
# swift-init account-server start
# swift-init account-replicator start
# swift-init account-auditor start
```

Si possono usare i seguenti comandi compatti:

```
# swift-init main start
# swift-init rest start
```

Fare in modo che i servizi di swift partano all'avvio del server:

```
# chkconfig openstack-swift-account on
# chkconfig openstack-swift-container on
# chkconfig openstack-swift-object on
```

- **Sul server Proxy** far partire il servizio proxy:

```
# swift-init proxy start
```

Fare in modo che i servizi di swift partano all'avvio del server:

```
# chkconfig openstack-swift-proxy on
```

Troubleshooting

- Da linea di comando usare il comando `swift` per mostrare informationi su account, container o oggetti.

```
# swift -V 2 -A http://<KEYSTONE_HOSTNAME>:5000/v2.0 -U <SERVICE_TENANT>:<SWIFT_USER> -K <
```

  Dove:

  - <KEYSTONE_HOSTNAME> è l'hostname del server Keystone (nel prototipo in esempio è openstack-01.cnaf.infn.it)
  - <SERVICE_TENANT> è il tenant del DB di Keystone che racchiude gli utenti dei servizi OpenStack
  - <SWIFT_USER> è il nome dell'utente del servizio Swift nel DB di Keystone
  - <ADMIN_PASSWORD> è la password dell'utente del servizio Swift nel DB di Keystone

- Caricare un file (viene creato automaticamente il container):

```
# swift -V 2 -A http://<KEYSTONE_HOSTNAME>:5000/v2.0 -U <SERVICE_TENANT>:<SWIFT_USER> -K <
```

  Dove

  - <CONTAINER_NAME> è il nome del container che si intende creare
  - <FILE_NAME> è il nome del file che si intende caricare

- Scaricare un file:

```
# swift -V 2 -A http://<KEYSTONE_HOSTNAME>:5000/v2.0 -U <SERVICE_TENANT>:<SWIFT_USER> -K <
```

- Mostrare i container esistenti:

```
# swift -V 2 -A http://<KEYSTONE_HOSTNAME>:5000/v2.0 -U <SERVICE_TENANT>:<SWIFT_USER> -K <
```

-- EnricoFattibene - 2012-10-12

This topic: MarcheCloudPilotaCNAF > InstallingAndConfiguringSwift
Topic revision: r9 - 2012-10-22 - EnricoFattibene