

DOVECOT ON CEPH CLUSTER

Document created on 2013-09-10

Updated on 2013-09-25

Nicolas Bondier

[\[pdf\]](#)[\[doc\]](#)[\[htm\]](#)

* * *



Copyright © 2013 by Switzernet

Contents

Introduction.....	3
Prerequisites.....	3
Ceph cluster.....	3
Create Ceph clustered block device shared with OCFS2.....	4
Rados Block device creation.....	4
OCFS2 shared file system	4
Mount the block device.....	8
Automatic mounting of the block device at start-up	9
Dovecot installation.....	11
Configure Dovecot.....	11
Create users and test.....	12
Annexes	14
Special network configuration for testing.....	14
Links.....	15

Introduction

This document present the installation of Dovecot servers on multiple Ceph clients sharing the same Rados Block Device.

The goal of this research is to build an IMAP cluster using a Ceph cluster as shared storage and to bring high availability (with DNS SRV for example) for the IMAP clients.

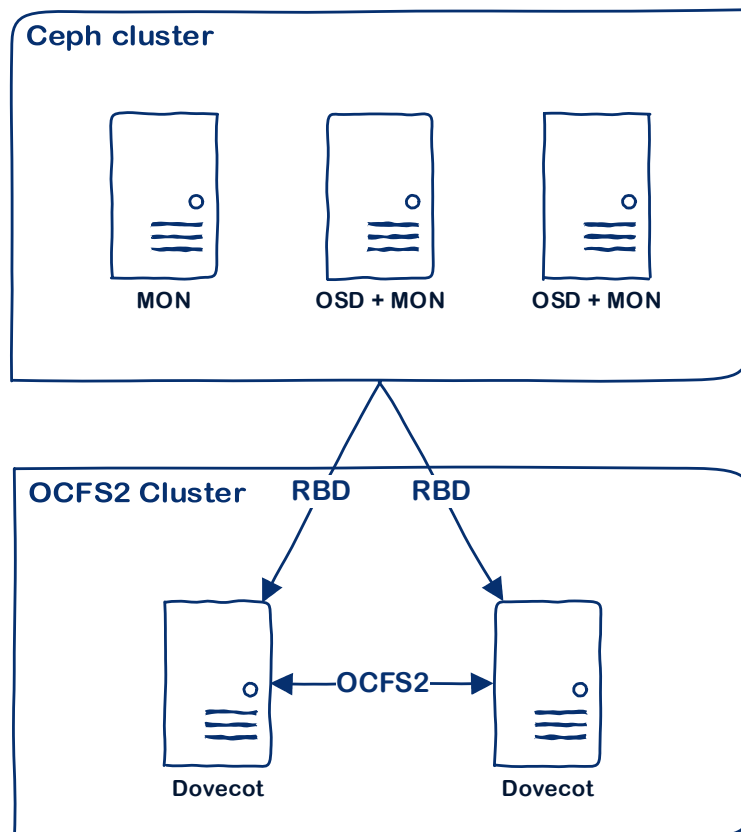
Prerequisites

Ceph cluster

The Ceph cluster is assumed to be running. For this document, we use a three nodes testing cluster. It is composed of 2 servers containing OSD daemons and Monitors, and a third server with a standalone Monitor server (it is only an admin server with a monitor daemon for monitor high-availability).

This document present the installation of two Ceph clients. They access to the Ceph cluster with RBD. This document doesn't explain how to install the nodes and RBD. The document explaining how-to install a complete Ceph cluster (servers and clients) is available under this link:

<http://switzernet.com/3/public/130925-ceph-cluster/>



You can find additional information about basic installation of Ceph cluster directly on Ceph web site: <http://ceph.com/docs/master/start/quick-ceph-deploy/>

Create Ceph clustered block device shared with OCFS2

Let's begin to prepare the storage media for Dovecot emails data.

Rados Block device creation

The first step is to create the block device inside the cluster. Here we prepare a 10 GB block device.

All Ceph configuration files are stored in the Ceph RBD clients (under `/etc/ceph/`). It automatically connect to the cluster without needing to provide hosts and admin keyring for each commands on the Ceph cluster.

From the first client, create a pool. We will call it "dovecot" for storing all the block devices used by Dovecot.

```
root@debian-test-1:~# ceph osd pool create dovecot 8 8
pool 'dovecot' created
root@debian-test-1:~#
```

Then we will create the block device directly in this pool on the Ceph cluster.

```
root@debian-test-1:~# rbd create dovecot-rbd-1 --size 10240 --pool dovecot
root@debian-test-1:~#
```

When created, you can simply list the block devices inside the pool with the command:

```
root@debian-test-1:~# rbd ls dovecot
dovecot-rbd-1
root@debian-test-1:~#
```

Map the new Block Device to the local machine:

```
root@debian-test-1:~# sudo rbd map dovecot-rbd-1 --pool dovecot --name
client.admin
root@debian-test-1:~#
```

The link to new mapped Block Device is located under `/dev/rbd/{pool-name}/{block-device-name}`.

```
root@debian-test-1:~# ls /dev/rbd/dovecot/dovecot-rbd-1
/dev/rbd/dovecot/dovecot-rbd-1
root@debian-test-1:~#
```

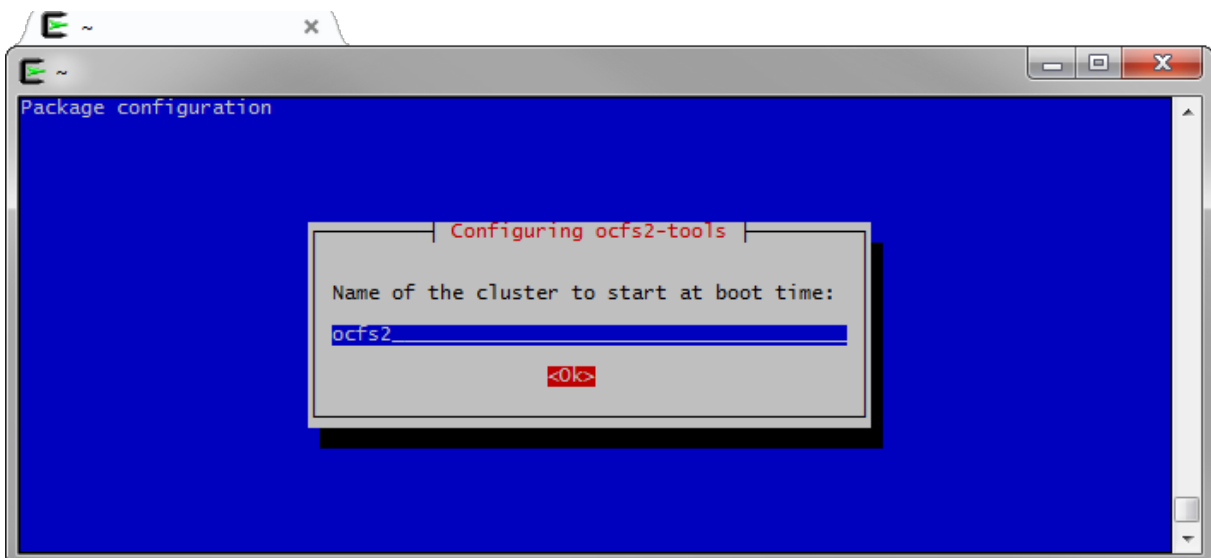
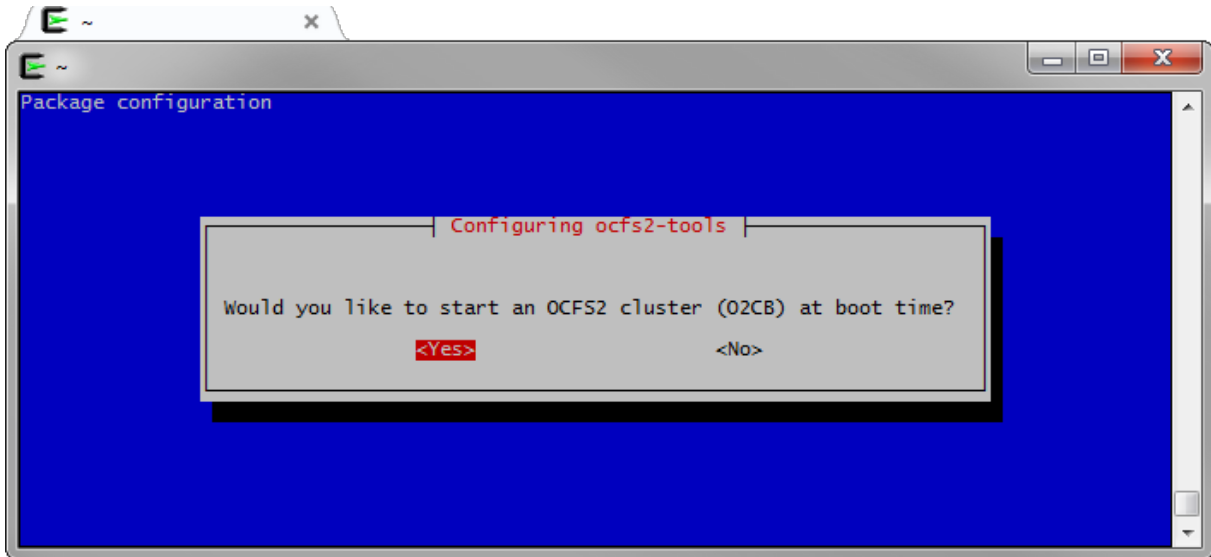
OCFS2 shared file system

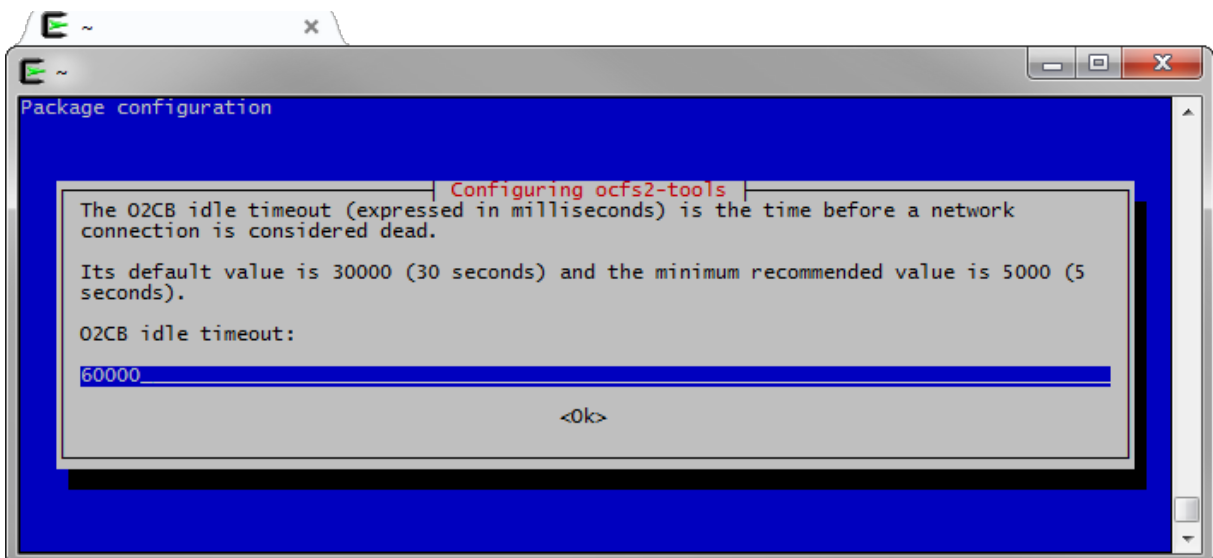
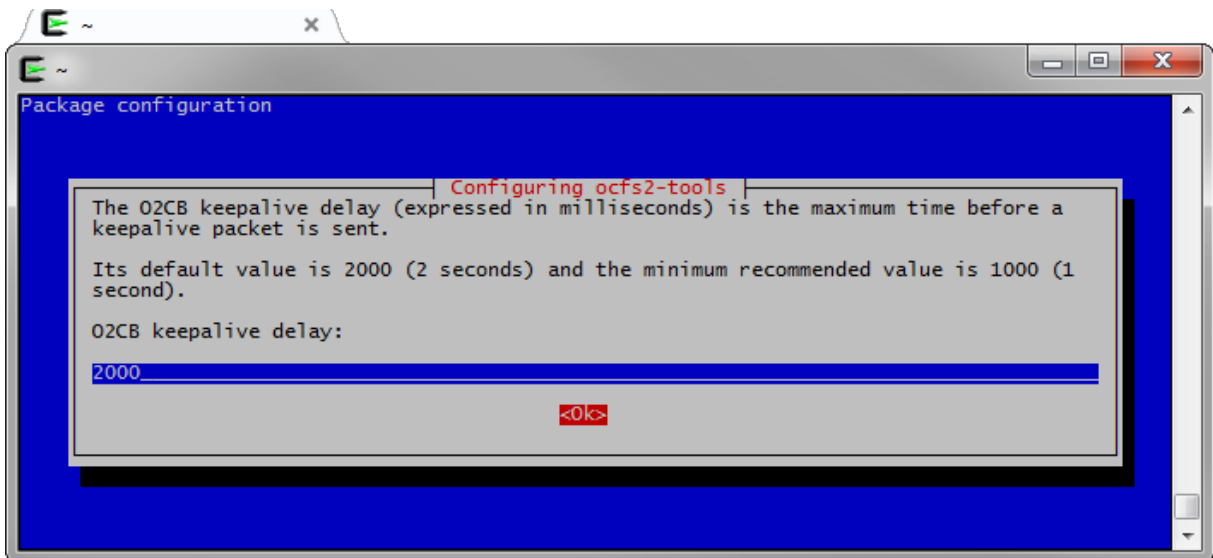
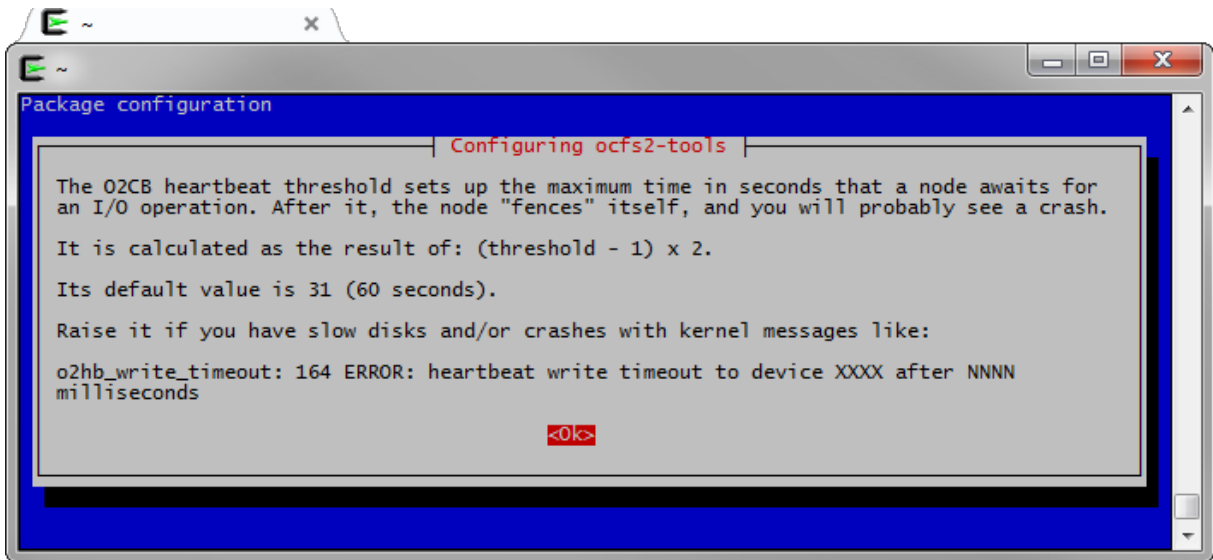
Install `ocfs2-tools` to create the `ocfs2` cluster above Ceph Block Devices.

```
root@debian-test-1:~# aptitude install ocfs2-tools
```

On each node, I will recommend to edit the `/etc/default/o2cb` file in order to prevent heartbeat automatically reboot the system in case of latency of ocfs2. This was necessary on [our network configuration](#). See the documentation at https://oss.oracle.com/projects/ocfs2/dist/documentation/v1.2/ocfs2_faq.html#TIMEOUT

Run `'dpkg-reconfigure ocfs2-tools'` command to edit the configuration file and increase the values as in the examples below:





Another setting that will help your clients to not encountering kernel panic during heavy io is to add 'elevator=deadline' option to your kernel boot command in your grub.cfg.

In recent debian distributions this can be done by editing '/etc/default/grub' and adding 'elevator=deadline' to the GRUB_CMDLINE_LINUX_DEFAULT variable.

```
root@debian-test-2:~# nano /etc/default/grub
...
GRUB_CMDLINE_LINUX_DEFAULT="elevator=deadline quiet"
...
```

When this is done, simply run the 'update-grub' command and verify your grub.cfg file.

```
root@debian-test-2:~# update-grub
Generating grub.cfg ...
Found background image: /usr/share/images/desktop-base/desktop-grub.png
Found linux image: /boot/vmlinuz-3.2.0-4-amd64
Found initrd image: /boot/initrd.img-3.2.0-4-amd64
done
root@debian-test-2:~# cat /boot/grub/grub.cfg
...
    linux    /boot/vmlinuz-3.2.0-4-amd64 root=UUID=733db890-811a-4fc5-
982f-f0326b3613fc ro elevator=deadline quiet
...
```

On each client, create the /etc/ocfs2/cluster.conf

```
root@debian-test-1:~# cat /etc/ocfs2/cluster.conf
node:
    ip_port = 7777
    ip_address = 192.168.2.163
    number = 0
    name = debian-test-1
    cluster = ocfs2
node:
    ip_port = 7777
    ip_address = 192.168.2.165
    number = 1
    name = debian-test-2
    cluster = ocfs2
cluster:
    node_count = 2
    name = ocfs2
root@debian-test-1:~#
```

Load o2cb service:

```
root@debian-test-1:~# service o2cb load
```

And make the OCFS2 service online on all the clients:

```
root@debian-test-1:~# service o2cb online
```

On the first node, format the partition. Here we use the mail type, which is faster for little size files:

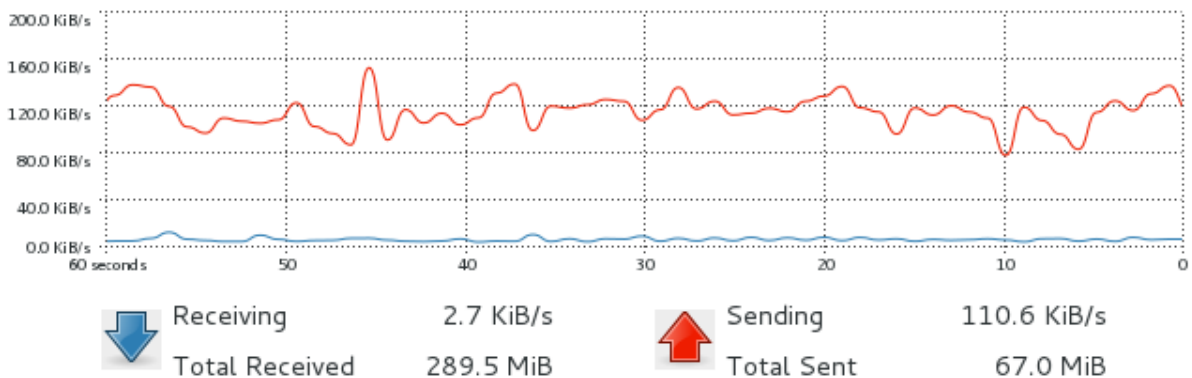
```

root@debian-test-1:~# mkfs.ocfs2 -T mail /dev/rbd/dovecot/dovecot-rbd-1
mkfs.ocfs2 1.6.4
Cluster stack: classic o2cb
Overwriting existing ocfs2 partition.
Proceed (y/N): y
Filesystem Type of mail
Label:
Features: sparse backup-super unwritten inline-data strict-journal-super
xattr
Block size: 4096 (12 bits)
Cluster size: 4096 (12 bits)
Volume size: 10737418240 (2621440 clusters) (2621440 blocks)
Cluster groups: 82 (tail covers 8704 clusters, rest cover 32256 clusters)
Extent allocator size: 4194304 (1 groups)
Journal size: 268435456
Node slots: 8
Creating bitmaps: done
Initializing superblock: done
Writing system files: done
Writing superblock: done
Writing backup superblock: 2 block(s)
Formatting Journals: done
Growing extent allocator: done
Formatting slot map: done
Formatting quota files: done
Writing lost+found: done
mkfs.ocfs2 successful

```

This step can take a long time, as the formatting is done over the network. The above chart is a screenshot from the gnome system monitor on the client and represents the traffic during the formatting of the block device.

Network History



Mount the block device

Once it is formatted, mount the device in a new folder of your machine:

```

root@debian-test-1:~# mkdir /mnt/dovecot-data
root@debian-test-1:~# mount /dev/rbd/dovecot/dovecot-rbd-1 /mnt/dovecot-
data

```


On the second client, the device is not mapped. Map and mount it:

```
root@debian-test-2:~# rbd map dovecot-rbd-1 --pool dovecot --name
client.admin
root@debian-test-2:~# mkdir /mnt/dovecot-data
root@debian-test-2:~# mount /dev/rbd/dovecot/dovecot-rbd-1 /mnt/dovecot-
data
```

Your two clients are now sharing the same block device and can read and write inside simultaneously.

Automatic mounting of the block device at start-up

It can be useful to connect the block device at start-up, to avoid manual mounting if the server restart for any reason.

The following steps must be done on all clients.

First, open your `/etc/fstab` and add the line corresponding to the block device at the end:

```
/dev/rbd/dovecot/dovecot-rbd-1      /mnt/dovecot-data      ocfs2
_netdev      0      0
```

- `/dev/rbd/dovecot/dovecot-rbd-1`: Link to the device.
- `/mnt/dovecot-data`: Mount point.
- `ocfs2`: File system type
- `_netdev`: This is a network device and it must be mounted when the network is available.
- `0`: File system must not be saved by dump utility
- `0`: No fsck check

With the current `fstab`, the network filesystem will currently not be mounted at start-up because the device is not mapped in the system until we launch the `'rbd map ...'` command. The line of the `fstab` will be used later in our own `init.d` scripts by a `'mount -a'` command when the block device is mapped and the `ocfs2` cluster is running.

Create the `/etc/init.d/rbdmap` script with the content of the file [[/etc/init.d/rbdmap](#)], and change the permissions.

```
root@debian-test-2:~# chmod 0755 /etc/init.d/rbdmap
```

The script use a simple configuration file, with `/etc/rbdmap` as path. Copy the following content to `/etc/rbdmap`.

```
# RbdDevice Parameters
#poolname/imagename [parameter=value[,parameter=value[,...]]]
dovecot/dovecot-rbd-1 name=client.admin
# End of the configuration file (keep this line)
```

Test if the script works:

```

root@debian-test-2:~# /etc/init.d/rbdmap start
Cluster ocfs2 already online
Starting RBD Mapping: dovecot/dovecot-rbd-1.
Mounting all filesystems...
root@debian-test-2:~#

```

And finally add at the end of the booting sequence:

```

root@debian-test-2:~# update-rc.d rbdmap defaults 80

```

You can now reboot to verify the automatic mounting of the file system. Here is an example of 'df -h' after a reboot:

```

root@debian-test-2:~# df -h

```

Filesystem	Size	Used	Avail
Use% Mounted on			
rootfs	323M	213M	94M
70% /			
udev	10M	0	10M
0% /dev			
tmpfs	50M	648K	50M
2% /run			
/dev/disk/by-uuid/733db890-811a-4fc5-982f-f0326b3613fc	323M	213M	94M
70% /			
tmpfs	5.0M	4.0K	5.0M
1% /run/lock			
tmpfs	304M	220K	304M
1% /run/shm			
/dev/sda9	7.9G	436M	7.1G
6% /home			
/dev/sda8	368M	11M	339M
3% /tmp			
/dev/sda5	7.4G	4.3G	2.9G
60% /usr			
/dev/sda6	2.8G	664M	2.0G
25% /var			
/dev/rbd0	10G	2.1G	8.0G
21% /mnt/dovecot-data			

Dovecot installation

Configure Dovecot

On each of the two clients, install Dovecot:

```
root@debian-test-1:~# aptitude update
root@debian-test-1:~# aptitude install dovecot-imapd
```

Dovecot configuration files are stored under `/etc/dovecot` and we will store all emails under the new file system `/mnt/dovecot-data`. If you wish to prevent Dovecot from starting when the block device is not mounted, I would suggest to store all configuration files directly in the block device.

Do this on one of the two clients:

```
root@debian-test-1:~# mkdir /mnt/dovecot-data/config
root@debian-test-1:~# mkdir /mnt/dovecot-data/mailboxes
root@debian-test-1:~# cp -r /etc/dovecot/* /mnt/dovecot-data/config/
```

On each client, remove the original `/etc/dovecot` directory and create a link to the configuration folder inside the mounted partition.

```
root@debian-test-1:~# rm -r /etc/dovecot
root@debian-test-1:~# ln -s /mnt/dovecot-data/config /etc/dovecot
root@debian-test-1:~# service dovecot restart
Restarting IMAP/POP3 mail server: dovecot.
```

```
root@debian-test-2:~# rm -r /etc/dovecot
root@debian-test-2:~# ln -s /mnt/dovecot-data/config /etc/dovecot
root@debian-test-2:~# service dovecot restart
Restarting IMAP/POP3 mail server: dovecot.
```

We will use UNIX users for testing authentication. This should be done with ldap or any other centralized authentication system in production.

Then on one of the clients, copy the configuration file [[/etc/dovecot/dovecot.conf](#)] and past its content to `/etc/dovecot/dovecot.conf`.

On each client, copy the following lines to your `/etc/pam.d/dovecot`:

```
auth    required    pam_unix.so
account required    pam_unix.so
```

Restart the Dovecot server:

```
root@debian-test-1:~# service dovecot restart
```

```
root@debian-test-2:~# service dovecot restart
```

Change the permission of the root mailboxes folder `/mnt/dovecot-data/mailboxes`.

```
root@debian-test-1:~# chmod 777 /mnt/dovecot-data/mailboxes
```

Create users and test

On both servers, create the same UNIX user with the same password:

```
root@debian-test-1:~# groupadd nicolas.bondier -g 1005;
root@debian-test-1:~# useradd nicolas.bondier -u 1005 -g 1005
root@debian-test-1:~# passwd nicolas.bondier
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

```
root@debian-test-2:~# groupadd nicolas.bondier -g 1005;
root@debian-test-2:~# useradd nicolas.bondier -u 1005 -g 1005
root@debian-test-2:~# passwd nicolas.bondier
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

You can now try to login using telnet on both servers:

```
root@debian-test-1:~# telnet 192.168.2.165 143
Trying 192.168.2.165...
Connected to 192.168.2.165.
Escape character is '^]'.
* OK [CAPABILITY IMAP4rev1 LITERAL+ SASL-IR LOGIN-REFERRALS ID ENABLE IDLE
AUTH=PLAIN AUTH=LOGIN] Dovecot ready.
> a login nicolas.bondier 123456
a OK [CAPABILITY IMAP4rev1 LITERAL+ SASL-IR LOGIN-REFERRALS ID ENABLE IDLE
SORT SORT=DISPLAY THREAD=REFERENCES THREAD=REFS MULTIAPPEND UNSELECT
CHILDREN NAMESPACE UIDPLUS LIST-EXTENDED I18NLEVEL=1 CONDSTORE QRESYNC
ESEARCH ESORT SEARCHRES WITHIN CONTEXT=SEARCH LIST-STATUS SPECIAL-USE]
Logged in
> a LIST "" *
* LIST (\HasNoChildren) "/" "INBOX"
* LIST (\HasNoChildren) "/" "Drafts"
* LIST (\HasNoChildren) "/" "Sent"
* LIST (\HasNoChildren) "/" "Junk"
* LIST (\HasNoChildren) "/" "Trash"
a OK List completed.
> a LOGOUT
* BYE Logging out
a OK Logout completed.
Connection closed by foreign host.
root@debian-test-1:~#
```

```
root@debian-test-1:~# telnet 192.168.2.163 143
Trying 192.168.2.163...
Connected to 192.168.2.163.
Escape character is '^]'.
* OK [CAPABILITY IMAP4rev1 LITERAL+ SASL-IR LOGIN-REFERRALS ID ENABLE IDLE
AUTH=PLAIN AUTH=LOGIN] Dovecot ready.
a login nicolas.bondier aaa000
a OK [CAPABILITY IMAP4rev1 LITERAL+ SASL-IR LOGIN-REFERRALS ID ENABLE IDLE
SORT SORT=DISPLAY THREAD=REFERENCES THREAD=REFS MULTIAPPEND UNSELECT
CHILDREN NAMESPACE UIDPLUS LIST-EXTENDED I18NLEVEL=1 CONDSTORE QRESYNC
ESEARCH ESORT SEARCHRES WITHIN CONTEXT=SEARCH LIST-STATUS SPECIAL-USE]
Logged in
a LIST "" *
* LIST (\HasNoChildren) "/" "INBOX"
* LIST (\HasNoChildren) "/" "Drafts"
* LIST (\HasNoChildren) "/" "Sent"
* LIST (\HasNoChildren) "/" "Junk"
* LIST (\HasNoChildren) "/" "Trash"
a OK List completed.
a LOGOUT
* BYE Logging out
a OK Logout completed.
Connection closed by foreign host.
root@debian-test-1:~#
```

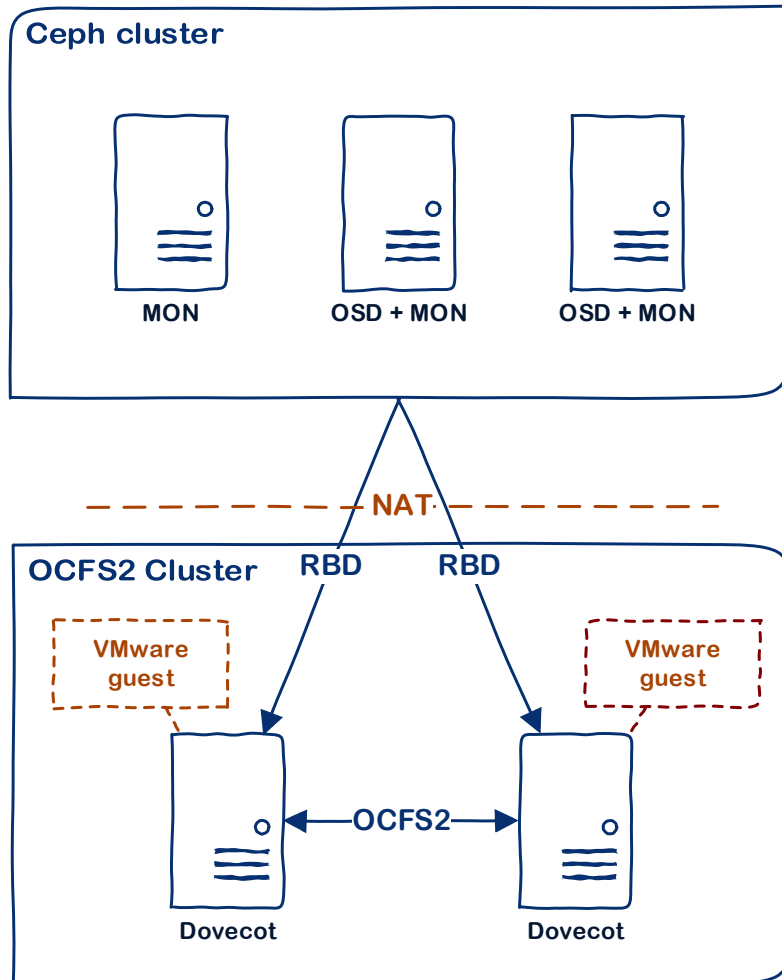
Configure two mail clients on each of the Dovecot server with the same credentials. You will this way test the way ocfs2 and Ceph RBD replicate the data between the both servers.

Annexes

Special network configuration for testing

I used a special network configuration, which is not explained in the document because of it doesn't influence the configuration.

The Ceph cluster is hosted with public addresses and is accessible from the internet, but the client/dovecot servers are VMware guests hosted on servers in a local network behind the router of our internet access provider.



Links

This document: <http://switzernet.com/3/public/130910-ceph-dovecot/>

Install a Ceph cluster: <http://switzernet.com/3/public/130925-ceph-cluster/>

Ceph Home Page: <http://ceph.com/>

Dovecot: <http://www.dovecot.org/>

Dovecot + NFS (for shared file system settings): <http://wiki.dovecot.org/NFS>

Ceph + OCFS2: <http://ceph.com/docs/next/rbd/rbd-config-ref/>

* * *



Copyright © 2013 by Switzernet