

VSM High Level Design

The main target of VSM project is to integrate Solid State Storage in Compute and Storage Nodes. This document mainly aims to describe our concept about VSM module. It contains:

- ☐ General Architecture
- ☐ Capacity Management
- ☐ Cluster Management
- ☐ CAS for VSM
- ☐ Status monitor of VSM cluster
- ☐ Web UI
- ☐ Developing Environment

1.1 General Architecture

Here, we mainly introduce the Architecture of VSM module. The design of VSM module is completely according to the architecture design of OpenStack. To design this architecture, we obey some principles:

- ☐ Easy to integrate with OpenStack modules.
- ☐ Fully de-coupled with other OpenStack modules.
- ☐ Ease to merge into other Cloud Platforms.
- ☐ With high reliability, high availability for large scale data center.
- ☐ Easy to use, deploy and manage.

1.1.1 Architecture

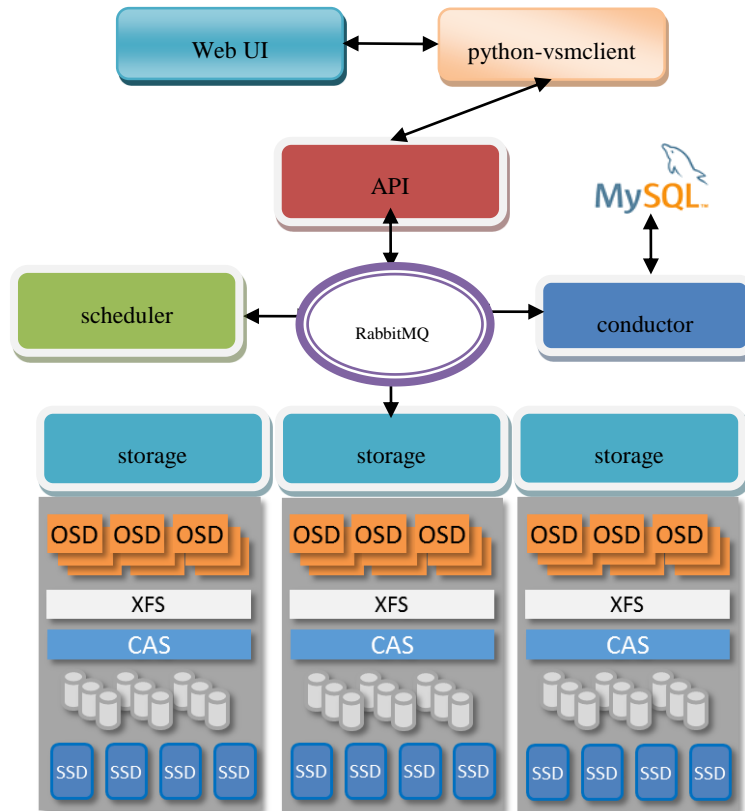


Figure 1.1 High level design of VSM module.

From figure 1.1, we can get that there are several modules. Every module in VMS has its own contribution. We'll introduce this architecture top-down.

1.1.2. Web UI

Our Web UI is built on horizon¹ which is a component of OpenStack. Here, we may simply introduce the design of horizon. From that, it's easy to understand why we choose horizon.

Horizon is built on Django which is a popular project written in Python. Actually, horizon does not provide Web UI to end user. Horizon just defines and provides basic elements such as tables, panels and templates.

The Web UI accessed by end user is provided by Dashboard which organizes basic elements provided by horizon. Dashboard is grouped by several panels such as "Project", "Admin" which are provided defiantly. It's notable that Dashboard is a subproject in horizon.

¹ <http://docs.openstack.org/developer/horizon/>.

The relationship of Panels, Dashboard, and horizon is list as below:

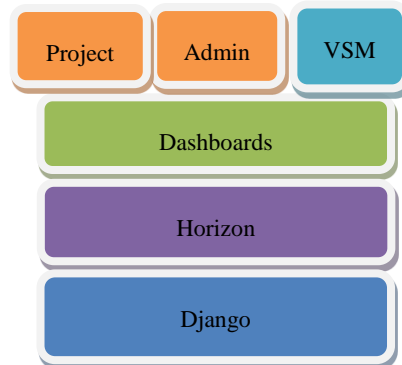


Figure 1.2 Relationships between modules in Horizon

Figure 1.2 shows that VSM's Web UI has been merged with OpenStack. The Web UI looks like:

OpenStack Dashboard

Project Admin **VSM**

capacity Management
Pools management
Presenting pools
Monitoring pools

cluster management
Ceph server management
Maintaining OSDs
Fault-tolerance processing
Creating recipes

Ceph server management Logged in as: admin Settings Help Sign Out

Add/remove server Add server

| <input type="checkbox"/> | host | server type | id | IP | status | create time | action |
|--------------------------|----------|-------------|------|---------------|--------|-------------------|--------|
| <input type="checkbox"/> | mon-host | MON | mona | 192.168.123.1 | up | 2013-9-4 14:50:32 | remove |
| <input type="checkbox"/> | osd-host | OSD | osdb | 192.168.123.1 | down | 2013-9-4 14:50:32 | remove |
| <input type="checkbox"/> | mds-host | MDS | mdsc | 192.168.123.1 | up | 2013-9-4 14:50:32 | remove |

server up/down

| <input type="checkbox"/> | host | server type | id | IP | status | create time | action |
|--------------------------|----------|-------------|------|---------------|--------|-------------------|--------|
| <input type="checkbox"/> | mon-host | MON | mona | 192.168.123.1 | up | 2013-9-4 14:50:32 | down |
| <input type="checkbox"/> | osd-host | OSD | osdb | 192.168.123.1 | down | 2013-9-4 14:50:32 | up |
| <input type="checkbox"/> | mds-host | MDS | mdsc | 192.168.123.1 | up | 2013-9-4 14:50:32 | down |

Figure 1.3 VSM's Web UI merged with OpenStack.

If we want to use VSM's Web UI individually, we can change the setting file¹ in Horizon. Find out the settings:

```
'dashboards': ('project', 'admin', 'settings', 'vsm'),
```

change it to be:

```
'dashboards': ('settings', 'vsm'),
```

After that, the architecture of Web UI looks as below:

¹ horizon/openstack_dashboard/settings.py

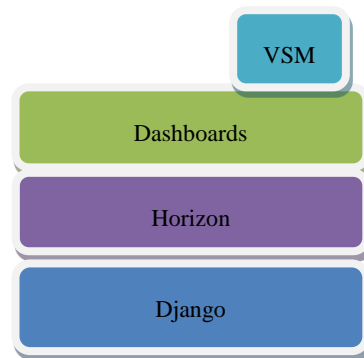


Figure 1.4 VSM's Web UI

Then the VSM's Web UI in browser looks like:

Ceph server management

Logged in as: admin Settings Help Sign Out

Add/remove server

| | host | server type | id | IP | status | create time | action |
|--------------------------|----------|-------------|------|---------------|--------|-------------------|--------|
| <input type="checkbox"/> | mon-host | MON | mona | 192.168.123.1 | up | 2013-9-4 14:50:32 | remove |
| <input type="checkbox"/> | osd-host | OSD | osdb | 192.168.123.1 | down | 2013-9-4 14:50:32 | remove |
| <input type="checkbox"/> | mds-host | MDS | mdsc | 192.168.123.1 | up | 2013-9-4 14:50:32 | remove |

server up/down

| | host | server type | id | IP | status | create time | action |
|--------------------------|----------|-------------|------|---------------|--------|-------------------|--------|
| <input type="checkbox"/> | mon-host | MON | mona | 192.168.123.1 | up | 2013-9-4 14:50:32 | down |
| <input type="checkbox"/> | osd-host | OSD | osdb | 192.168.123.1 | down | 2013-9-4 14:50:32 | up |
| <input type="checkbox"/> | mds-host | MDS | mdsc | 192.168.123.1 | up | 2013-9-4 14:50:32 | down |

Figure 1.5 VSM's Web UI individually¹

1.1.3. python-vsmclient

Web UI is just a pure UI (even without database). When Web UI needs some information from VSM, it uses python-vsmclient to connect with VSM-api module.

Just likes other components in OpenStack², python-vsmclient is the client tool of VSM. Python-vsmclient is used to send RESTful request to VSM api server.

With python-vsmclient, Web UI is fully de-coupled with VSM API. In the future, if we want to add VSM into other cloud computing platforms such as CloudStack. It just needs to send

¹ As your wish, we may remove the OpenStack's logo in the Web UI.

² For example, Keystone has python-keystoneclient. Nova module has python-novaclient.

RESTful API request to VSM.

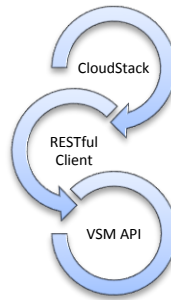


Figure 1.6 Add VSM into CloudStack.

Although CloudStack is not written in python, but it's very easy to find one client tool to instead python-vsmclient. By this strategy, VSM is de-coupled with specific cloud computing platforms. It can be merged into different cloud platforms with flexibility.

1.1.4. API

API¹ module in VSM provides RESTful API for VSM project. It receives RESTful request from python-vsmclient, curl, and java RESTful client tools².

In python, it's easy to build RESTful API for VSM by PasteDeploy³ project. As we know, after API receives the request sent by client tools, there are a lot of procedures to process. With PasteDeploy, these procedures are split into atomic operations. The request processing pipeline is constructed by these atomic operations.

1. Work with Keystone

For example, we may make up a simple pipeline for VSM. In the configuration file⁴, we can define a pipeline as below:

```
keystone = faultwrap authtoken ratelimit vsmapi_v1
```

The request processing procedure looks like:

¹ In VSM, we just call it API. To make different with other API modules in OpenStack, we may call it VSM-API or vsm-api. In this paper, if we say API, it refers to VSM-API.

² API just receives the request with no sense about what the client tools is.

³ <http://pythonpaste.org/deploy/>.

⁴ /etc/vsm/api-paste.ini

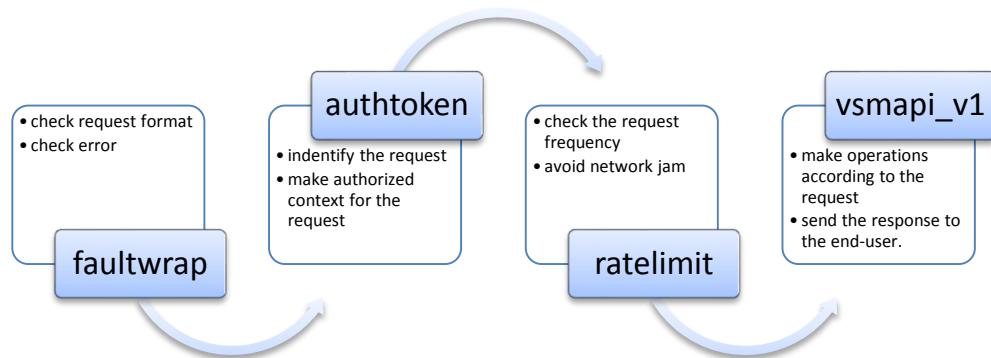


Figure 1.7 An example of pipeline.

From Figure 1.7, faultwrap will handle the request at first, and then transfer the request to authtoken. At last, vsmapi_v1 will receive the request if the request passed atomic operations forehead. The unauthorized and illegal request would not handle by vsmapi_v1. If authtoken find any illegal information in the request, authtoken will send response with “Unauthorized”. Then the request will not pass to other atomic operations in the pipeline.

If we want to use Keystone project to make user authentication, we can set authtoken as below:

```
keystone = faultwrap authtoken ratelimit vsmapi_v1
authtoken = keystoneclient.middleware.auth_token
```

“keystoneclient.middleware.auth_token” is a class in Keystone’s source code. It’s very convenient that we do not need to merge Keystone’s source code in our VSM project. We just need to change the **configuration file** with “authtoken” which refers to Keystone’s code. If we use Keystone in VSM, the architecture of VSM would looks as below:

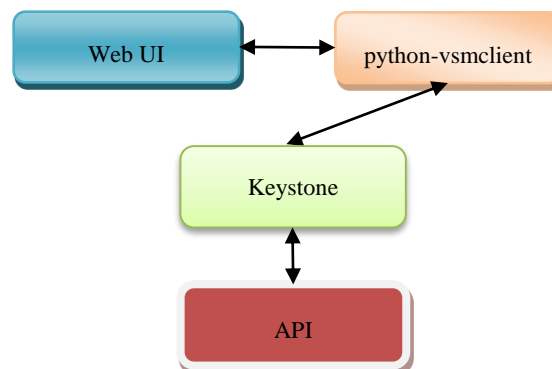


Figure 1.8 Use Keystone in VSM

2. Without Keystone

If we do not want to use Keystone user identification, we may change the configuration file as:

```
keystone = faultwrap ratelimit vsmapi_v1
```

Then the pipeline can be list as below:

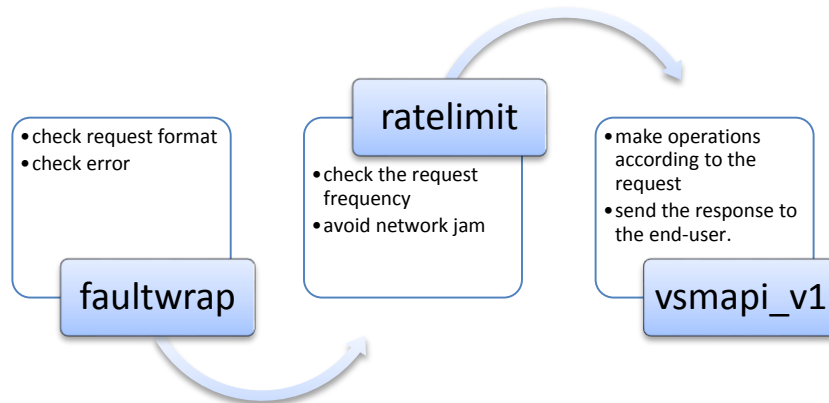


Figure 1.9 the pipeline without Keystone.

3. Use other Authentication Package

As described, with PasteDeploy, we can use the third part user authenticate package easily. For example, we want to merge VSM into another cloud platform. We have to use their user authentication strategy. We may change the configuration file as below:

```
name = faultwrap cloudstackauth ratelimit vsmapi_v1
cloudstackauth = cloudstack.auth.auth_token
```

Sometimes, there are no class named "cloudstack.auth.auth_token" for us to use. We can write one as below:

```
class auth_token:
    def __call__(self, request):
        resp, ret = self._cloudstack_client.auth(request)
        if resp == 404:
            return 404, "Unthorized"
        else:
            return self.application
```

We just write wrapper classes which just call the cloudstack's authenticate client tools.

1.1.5. Scheduler

After API received the request from end user, API module just transfer these request to other

modules, such as conductor¹, scheduler. The API module send the request to other modules by in different conditions:

- ❑ Conductor: Get/Update information in data base.
- ❑ Scheduler: Actions need to be done. Actions may be “Create/Delete/Snapshot” of block storage.

Scheduler module is used to make decision. Take the creation of volume as an example as below:

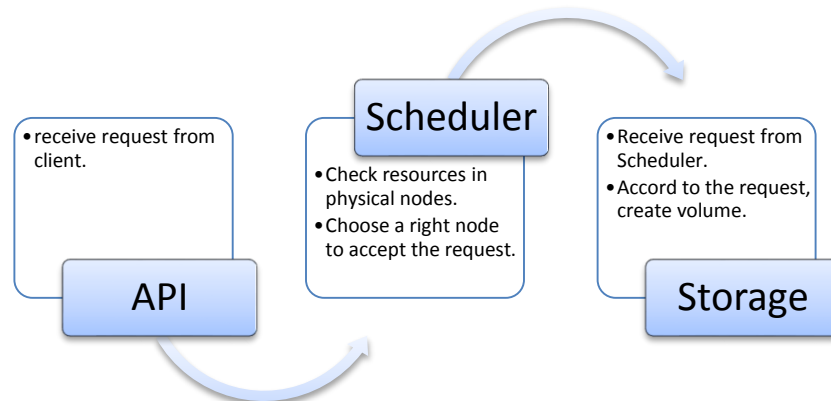


Figure 1.10 The usage of Scheduler module.

We can add different driver for Scheduler. Then we can select physical nodes by different strategy.

1.1.6. Conductor

Conductor provides data base services for VSM project. With conductor module, VSM project is de-coupled with DataBase such as MySQL, Oracle, and DB2. Actually, Conductor is built on python-sqlalchemy² which is a DB toolkit written in Python. Then we can use different DB which is supported by python-sqlalchemy.

1.1.7. Storage

Storage is the most important part of VSM. To support Ceph, CAS, and other features, Storage module can be designed as below:

¹ Conductor is used as data base service. Usually, conductor use MySQL as its DB.

² http://docs.sqlalchemy.org/en/rel_0_8/.

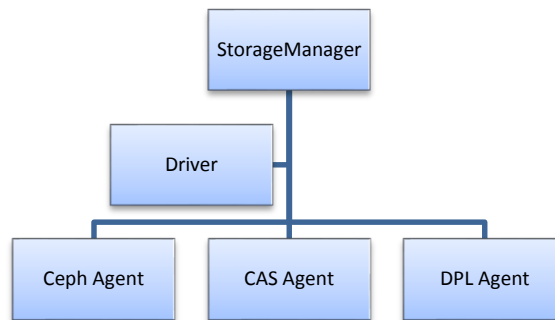


Figure 1.11 the design of Storage module.

The StorageManager is used to receive request sent by Scheduler or other modules. The Driver implements interfaces defined in StorageManager with different Agent. With this design, it's very easy to add more agents for Storage module.

1.1.8. RabbitMQ

Between python-vsmclient and API, we use RESTful API to communication. For the internal modules, how do they communicate with others?

In OpenStack, they choose RabbitMQ services to implement RPC (remote procedure call) between internal modules. In VSM project, we also use RabbitMQ which is very easy to use and support more modules.

1.2 Management

Here we will introduce several parts of management which includes:

- ❑ Capacity Management
- ❑ Cluster Management

1.2.1. Capacity Management

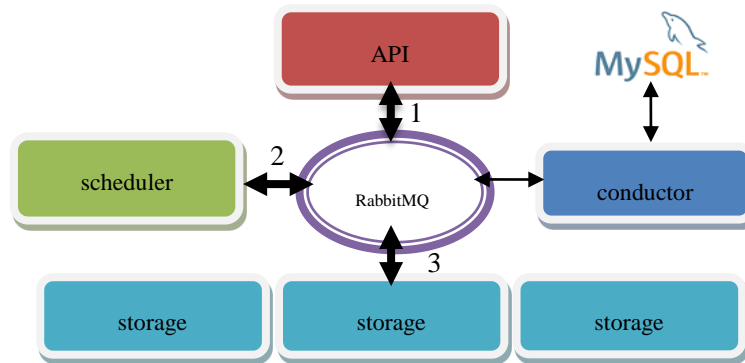


Figure 2.1 the procedure of capacity management.

From Figure 2.1, we can see the procedure of Capacity Management:

- ❑ Api module receives clients' requests.
- ❑ Scheduler receives request, chooses the right physical node. Then transfer the request to it.
- ❑ Storage node receives the request, and then manages the capacity.

1. Storage pools

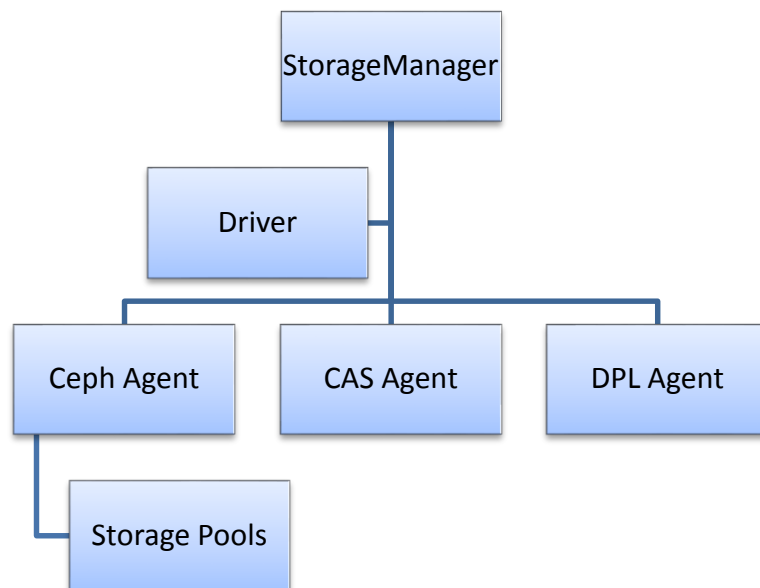


Figure 2.2 Ceph pools' management

To manage ceph pools in Ceph, we add these functions in the Ceph Agent. There are several aspects of Storage Pools Management. Here we just describe our high level design.

1. Storage Pool Recipes

Before we create storage pools, we may need to define Storage Pool Recipes. Although recipes are related with Storage, the definition of recipe should be stored in data base. So the conductor module will handle recipes' operation such as define, undefine, list and query.

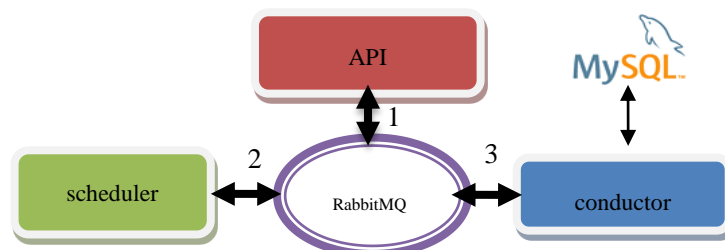


Figure 2.3 the definition of recipes.

Recipes' operations should follow the procedure:

- ❑ API receives requests about from clients. Then transfer the request to Scheduler.
- ❑ Scheduler receives the request from API by RabbitMQ. It finds out one running conductor service and then transfer this request to the chosen conductor.
- ❑ After Conductor receives the request sent by Scheduler, it will record information about recipes in DB.

2. Create Storage Pools

Creating storage pools has more steps, this operation need every module to participate in.

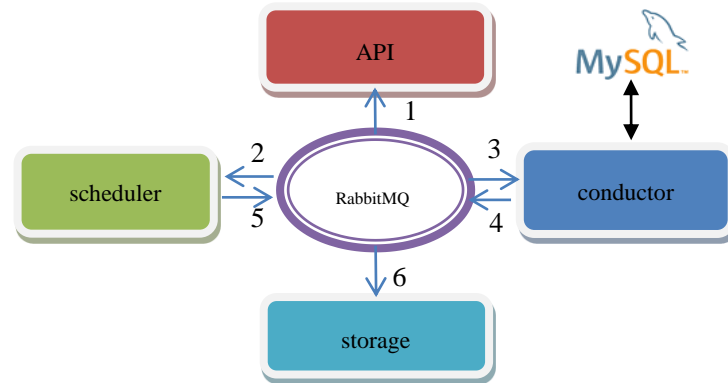


Figure 2.4 Steps to create storage pools

Different modules in VSM have different function:

- ❑ Step 1: API receives requests about from clients. Send the authenticated request to RabbitMQ.
- ❑ Step 2: Scheduler receives requests which are sent by API.
- ❑ Step 3: In order to get more information about recipes, Scheduler transfer request to Conductor by RabbitMQ.
- ❑ Step 4: Conductor collects information about recipes from DB, and then transfers details about recipes back to Scheduler.
- ❑ Step 5: Scheduler receives details about recipes sent by Conductor, and then finds out the appropriate Storage to send the request.
- ❑ Step 6: Storage module gets the request, and then begins to create storage pools.

When storage module gets the request, it sends the request to Ceph Agent. Ceph agent will use Ceph to manage storage pools. For creating storage pools, there are two methods provided by Ceph:

✧ Command line¹:

```
ceph osd pool create {pool-name} {pg-num} [{pgp-num}]
```

✧ Librados API

¹ <http://ceph.com/docs/master/rados/operations/pools/>.

```

intrados_pool_create(rados_t cluster, const char *pool_name)
intrados_pool_create_with_auid(rados_t cluster, const char *pool_name, uint64_t auid)
intrados_pool_create_with_crush_rule(rados_t cluster, const char *pool_name, __u8
intrados_pool_create_with_all(rados_t cluster, const char *pool_name, uint64_t auid, __u8

```

3. Access Right to Storage Pools

In our high level design, we use Keystone to control access right to storage pools and other resources. Why we choose Keystone? There are several reasons:

- ✧ With Keystone, it's very easy to merge into OpenStack.
- ✧ As we know, Keystone is an independent module. We can use Keystone in VSM without efforts to change any code. We just need to change configuration file.
- ✧ With PasteDeploy package, it's easy to use other user access right controlling packages. With this flexibility, we can easily add VSM project into other cloud computing platform with little efforts to use their Authentication module.

Ceph provides two strategies to the control access right to storage pools:

- ✧ No controlling. Gives out this function to upper level applications such as Keystone.
- ✧ Cephx: Ceph provides user authentication in a manner similar to Kerberos¹. Such as:

```

$ ceph auth get-or-create client.{username} {daemon1} {cap1} {daemon2} {cap2} ...
$ ceph auth get-or-create client.foo osd 'allow rw' mon 'allow r' > keyring.foo

```

The best way to control access right of lower level resources might be use Keystone and Cephx both.

¹ <http://ceph.com/docs/master/rados/operations/authentication/#add-a-key>

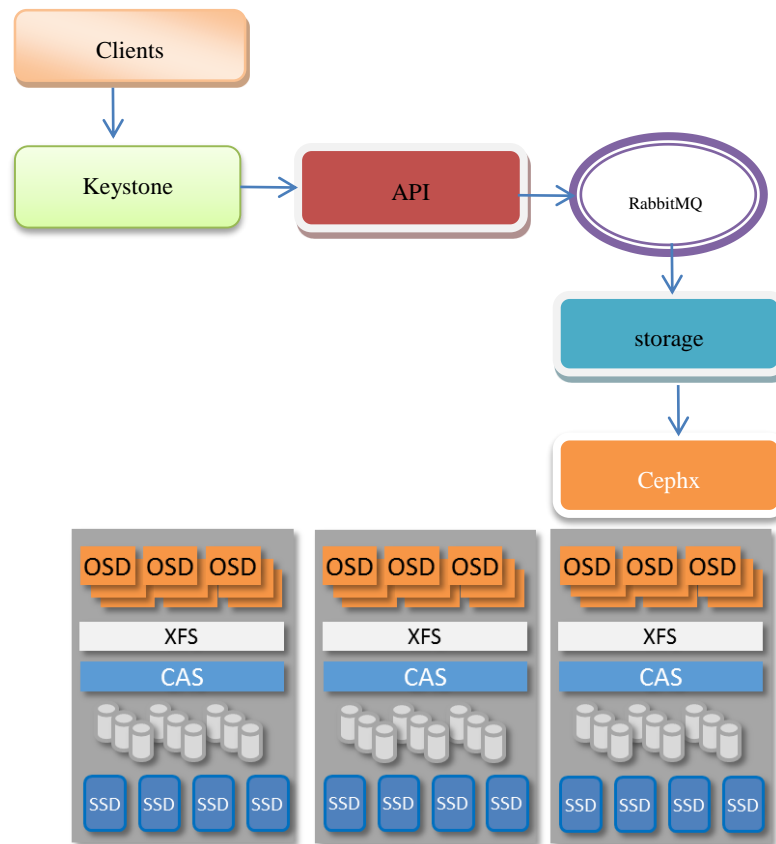


Figure 2.5 Use Keystone and Cephx to control the access right.

Keystone is to authenticate the user's request. The illegal request will be rejected. But why we also use Cephx?

For Storage module, we create Admin user in Cephx. After user's request passed the authentication of Keystone, Storage module will use Admin user to manage resources in Ceph system.

For flexibility, we can change configuration file to use or unuse Keystone in VSM. In Storage module, we can also write one option in configuration file:

```

use_cephx=Yes/No
cephx_user=
  
```

If we set "use_cephx=No", in Storage module of VSM, cephx will not be used. So, we can use/unused Keystone and Cephx in VSM by changing configuration file.

4. Storage Pools for OpenStack

Ceph supports three kinds of usage of storage pools:

✧ Ceph File System

With Ceph File System provided by VSM, OpenStack can use it in this way:

- ✚ Glance can use storage pools to storage virtual machines' images.
- ✚ Nova can use CephFS to storage virtual machines' disks.
- ✧ Ceph block device

Virtual Machines can attach/detach Ceph block devices with Ceph qemu rbd driver¹.
- ✧ Ceph object gateway

OpenStack can use CephGW as below:

 - ✚ Provides storage devices for swift.
 - ✚ Because CephGW supports S3², Glance services in OpenStack can use this interface to storage VM's images.

5. Increasing the Capacity of Storage Pools

Increasing the capacity of storage pools has the same procedure with creating storage pools. The differences are that several steps have different jobs.

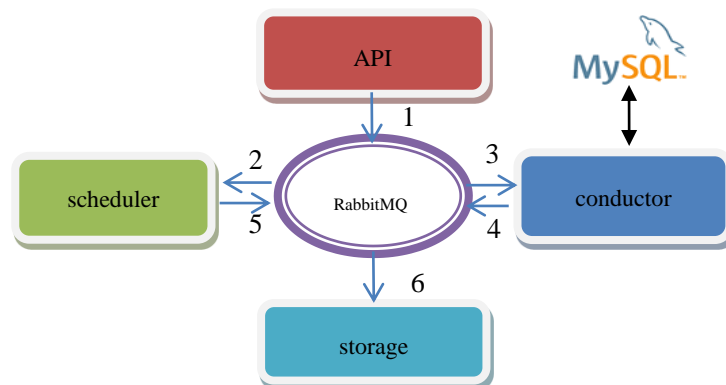


Figure 2.6 Steps to increase the capacity of storage pools.

Steps to increase the capacity of Storage Pools:

- ❑ Step 1: API receives requests about from clients. Send the authenticated request to RabbitMQ.
- ❑ Step 2: Scheduler receives requests which are sent by API.
- ❑ Step 3: In order to get more information about storage pools, Scheduler transfer request to Conductor by RabbitMQ.
- ❑ Step 4: Conductor collects information about storage pools from DB, and then transfers details about storage pools back to Scheduler.
- ❑ Step 5: Scheduler receives details about storage pools sent by Conductor, and then finds out the appropriate Storage to send the request.
- ❑ Step 6: Storage module gets the request, and then begins to increase the capacity

¹ If you want VMs to use RBD devices provided by Ceph, you have to install Qemu tools provided by Ceph.

² <http://ceph.com/docs/master/radosgw/s3/>.

storage pools.

After the capacity of storage pool is changed in Step 6, conductor will update the information about storage pools.

6. Update information of Storage Pools

In order to monitor pool utilization, performance and other details, we have to update information about storage pools periodically.

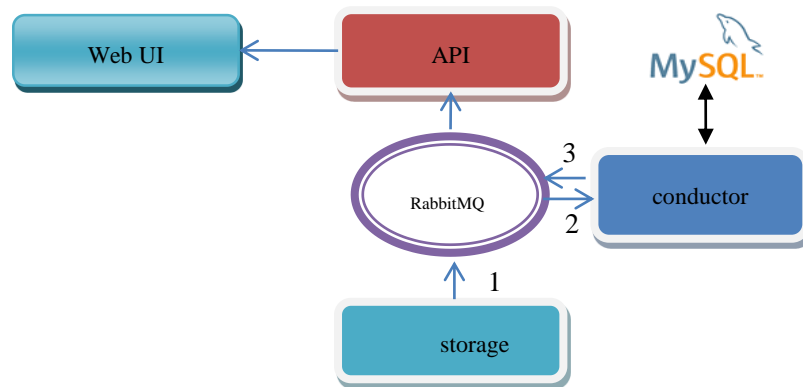


Figure 2.7 Steps to update information about storage pools.

Steps to update information about storage pools:

- ❑ Step 1: periodic threads monitor information about storage pools. If it's valuable to update into DB, Storage module send these information into RabbitMQ. In Storage module, there are two methods to monitor information:

- i. command line

```
$ radosdf
```

- ii. librados

```
intrados_ioctx_pool_stat(rados_ioctx_tio, structrados_pool_stat_t *stats)
```

- ❑ Step 2: Conductor receives information sent by Storage from RabbitMQ, and then write these information into DB.

In order to show the information in Web UI, we will use Ajax in Web UI which can update information from API. When API receives request from Web UI, it will collect information from conductor then response to Web UI.

This is strategy is also used in **Monitor of Cluster**.

1.2.2. Cluster Management

With our high level design, cluster management is the same with Storage Pools management.

The only difference is actions in Storage module. So we can extend Storage module to support cluster operations.

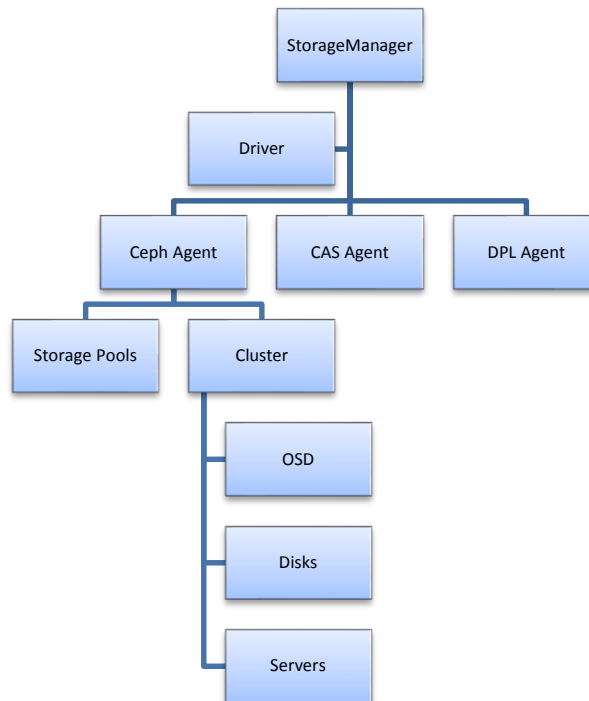


Figure 2.8 Cluster management in Ceph Agent.

For OSD, Disk and Servers, we have several operations:

- ✧ Add
- ✧ Delete
- ✧ Maintain

For monitoring of cluster's status, we can also use ganglia¹ and nagios² which are very easy to merge into our Web UI.

1.2.3. CAS Management

Because CAS is running on single node, so we run Storage module in each node. With CAS agent in Storage module, we can manage CAS in VSM. The design architecture is the same with Ceph Agent; so details will not be introduced here.

¹ <http://ganglia.sourceforge.net/>.

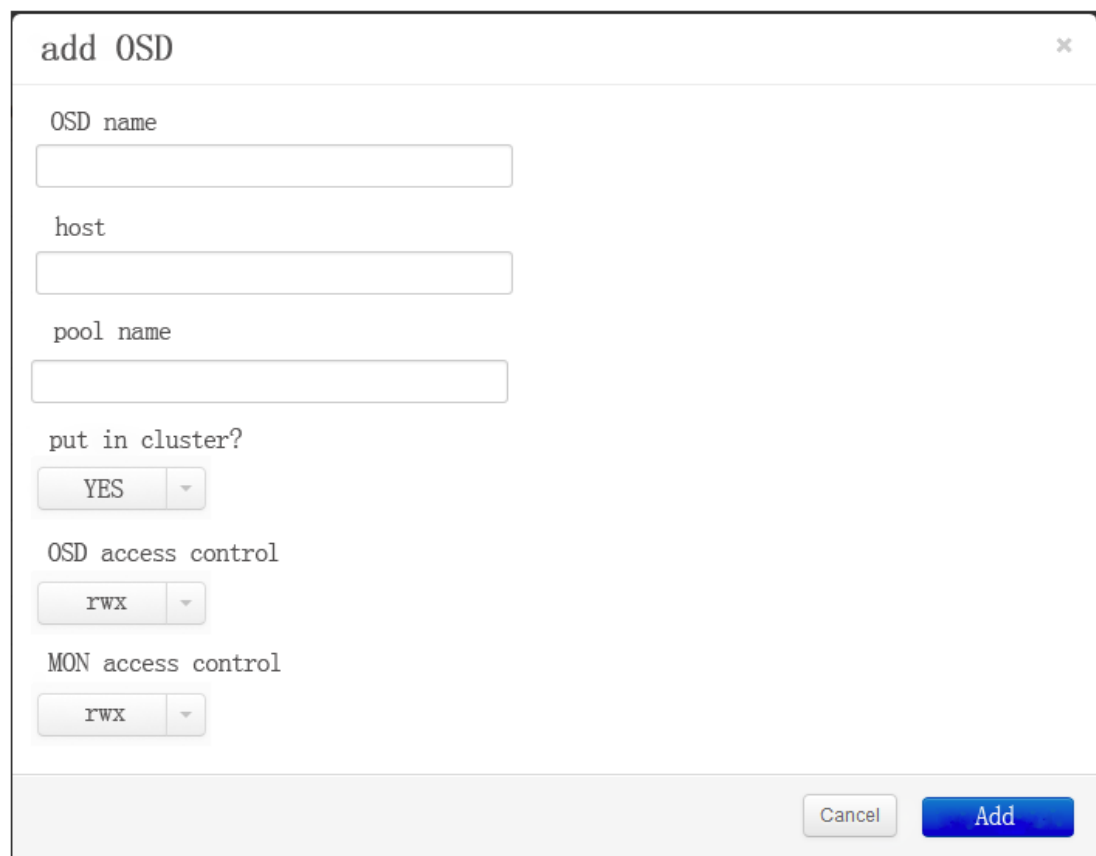
² <http://www.nagios.org/>.

1.3 Web UI

Here we design some examples of Web UI. There are several parts of our Web UI:

- ☐ OSD
- ☐ Server
- ☐ Storage Pools

1.3.1. OSD

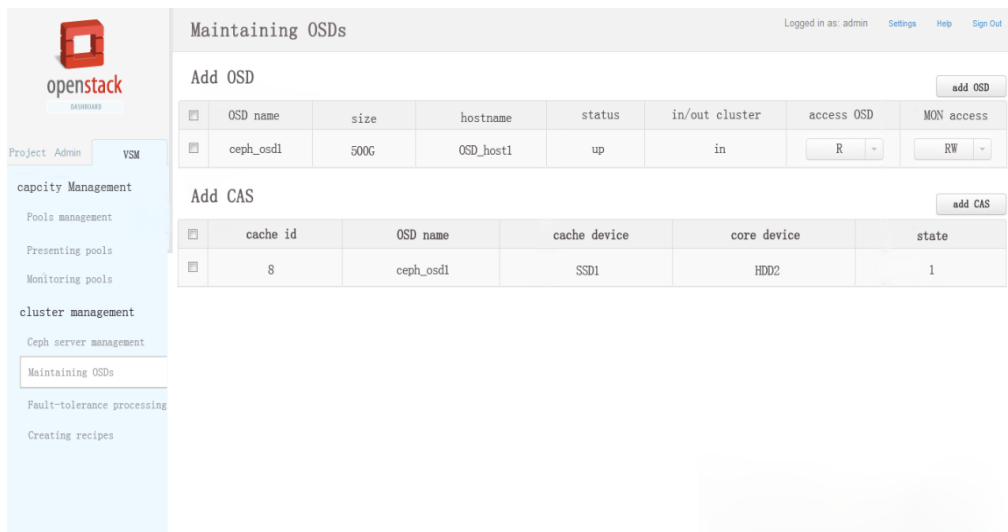


The screenshot shows a web UI form titled "add OSD" with a close button (X) in the top right corner. The form contains the following fields and controls:

- OSD name:** A text input field.
- host:** A text input field.
- pool name:** A text input field.
- put in cluster?:** A dropdown menu with "YES" selected.
- OSD access control:** A dropdown menu with "rwx" selected.
- MON access control:** A dropdown menu with "rwx" selected.

At the bottom right of the form, there are two buttons: "Cancel" and "Add".

Figure 3.1 Add OSD in Web UI.



Maintaining OSDs Logged in as: admin [Settings](#) [Help](#) [Sign Out](#)

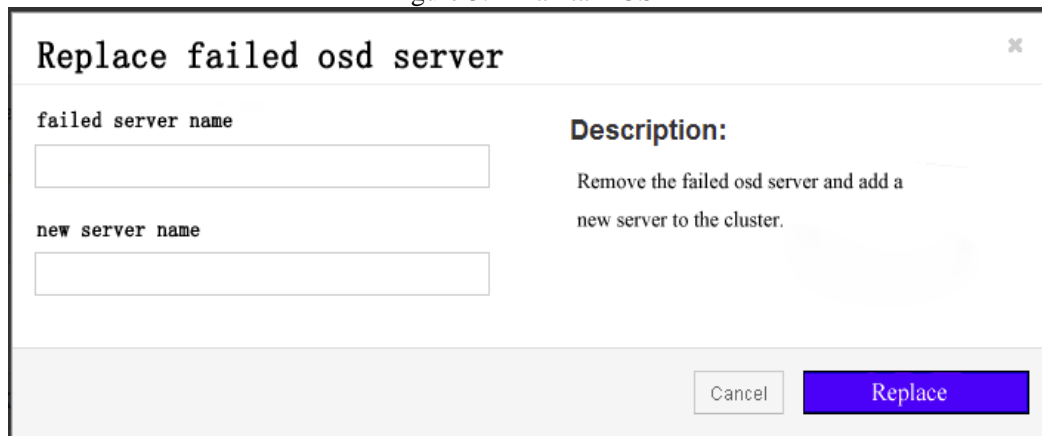
Add OSD add OSD

| <input type="checkbox"/> | OSD name | size | hostname | status | in/out cluster | access OSD | MON access |
|--------------------------|-----------|------|-----------|--------|----------------|------------|------------|
| <input type="checkbox"/> | ceph_osd1 | 500G | OSD_host1 | up | in | R - | RW - |

Add CAS add CAS

| <input type="checkbox"/> | cache id | OSD name | cache device | core device | state |
|--------------------------|----------|-----------|--------------|-------------|-------|
| <input type="checkbox"/> | 8 | ceph_osd1 | SSD1 | HDD2 | 1 |

Figure 3.2 Maintain OSD



Replace failed osd server ✕

failed server name

new server name

Description:

Remove the failed osd server and add a new server to the cluster.

Cancel Replace

Figure 3.3 Replace failed OSD.

1.3.1. Ceph Servers

add server

server type

OSD server

ID

host

IP

port

Cancel

add

Figure 3.4 Add Ceph Servers.

openstack

dashboard

Project Admin VSM

capacity Management

Pools management

Presenting pools

Monitoring pools

cluster management

Ceph server management

Maintaining OSDs

Fault-tolerance processing

Creating recipes

Ceph server management

Logged in as: admin Settings Help Sign Out

Add/remove server

Add server

| | host | server type | id | IP | status | create time | action |
|--------------------------|----------|-------------|------|---------------|--------|-------------------|--------|
| <input type="checkbox"/> | mon-host | MON | mona | 192.168.123.1 | up | 2013-9-4 14:50:32 | remove |
| <input type="checkbox"/> | osd-host | OSD | osdb | 192.168.123.1 | down | 2013-9-4 14:50:32 | remove |
| <input type="checkbox"/> | mds-host | MDS | mdsc | 192.168.123.1 | up | 2013-9-4 14:50:32 | remove |

server up/down

| | host | server type | id | IP | status | create time | action |
|--------------------------|----------|-------------|------|---------------|--------|-------------------|--------|
| <input type="checkbox"/> | mon-host | MON | mona | 192.168.123.1 | up | 2013-9-4 14:50:32 | down |
| <input type="checkbox"/> | osd-host | OSD | osdb | 192.168.123.1 | down | 2013-9-4 14:50:32 | up |
| <input type="checkbox"/> | mds-host | MDS | mdsc | 192.168.123.1 | up | 2013-9-4 14:50:32 | down |

Figure 3.5 Ceph servers management.

1.3.1. Storage Pools

The 'create pools' dialog box contains the following fields and controls:

- pools name:** A text input field.
- cluster:** A text input field.
- auid:** A text input field.
- crush rule num:** A text input field.
- use recipes?:** A text input field.
- get snapshot:** A button located to the right of the 'use recipes?' field.
- Cancel:** A button at the bottom right.
- create:** A blue button at the bottom right.

Figure 3.6 Create Storage Pools

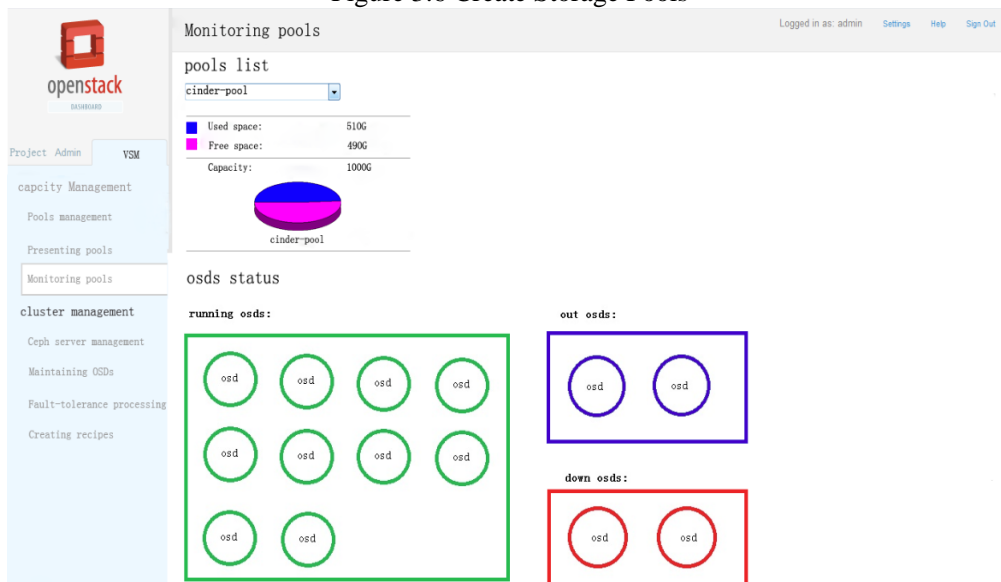
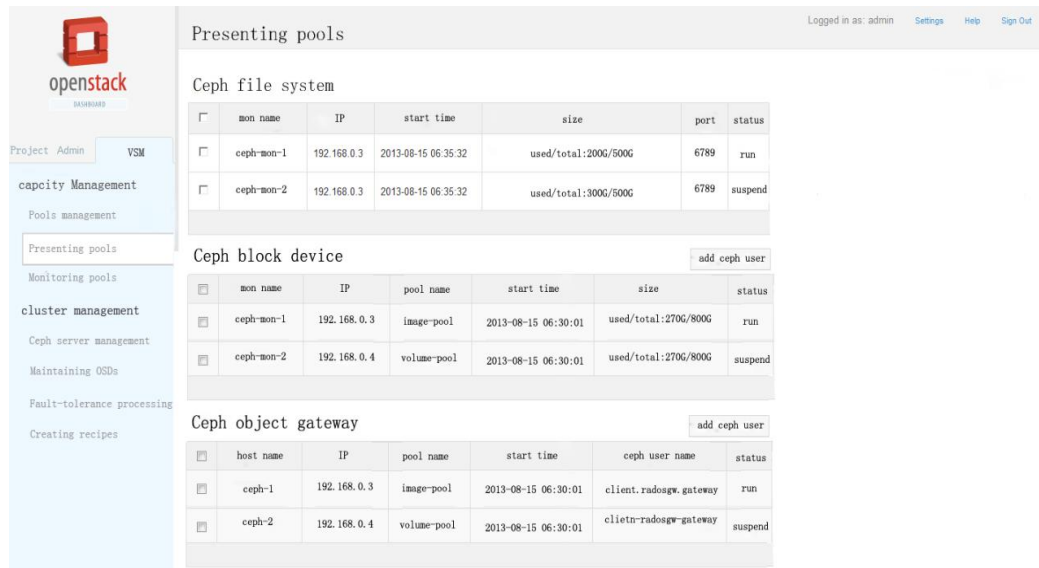


Figure 3.7 Monitor Storage Pools



Presenting pools

Logged in as: admin [Settings](#) [Help](#) [Sign Out](#)

Ceph file system

| <input type="checkbox"/> | mon name | IP | start time | size | port | status |
|--------------------------|------------|-------------|---------------------|----------------------|------|---------|
| <input type="checkbox"/> | ceph-mon-1 | 192.168.0.3 | 2013-08-15 06:35:32 | used/total:200G/500G | 6789 | run |
| <input type="checkbox"/> | ceph-mon-2 | 192.168.0.3 | 2013-08-15 06:35:32 | used/total:300G/500G | 6789 | suspend |

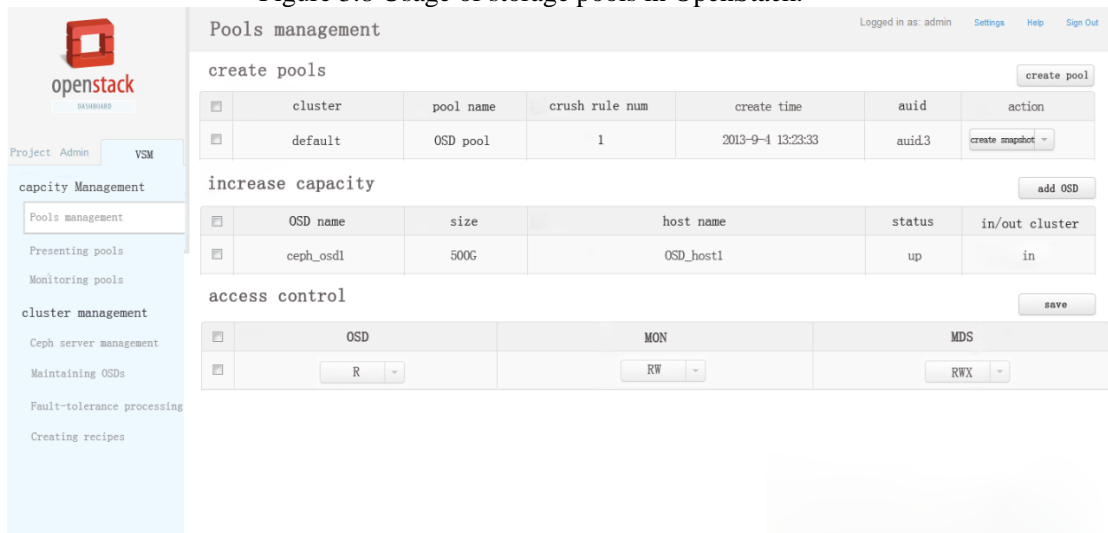
Ceph block device [add ceph user](#)

| <input type="checkbox"/> | mon name | IP | pool name | start time | size | status |
|--------------------------|------------|-------------|-------------|---------------------|----------------------|---------|
| <input type="checkbox"/> | ceph-mon-1 | 192.168.0.3 | image-pool | 2013-08-15 06:30:01 | used/total:270G/800G | run |
| <input type="checkbox"/> | ceph-mon-2 | 192.168.0.4 | volume-pool | 2013-08-15 06:30:01 | used/total:270G/800G | suspend |

Ceph object gateway [add ceph user](#)

| <input type="checkbox"/> | host name | IP | pool name | start time | ceph user name | status |
|--------------------------|-----------|-------------|-------------|---------------------|-------------------------|---------|
| <input type="checkbox"/> | ceph-1 | 192.168.0.3 | image-pool | 2013-08-15 06:30:01 | client.radosgw.gateway | run |
| <input type="checkbox"/> | ceph-2 | 192.168.0.4 | volume-pool | 2013-08-15 06:30:01 | clientn.radosgw.gateway | suspend |

Figure 3.8 Usage of storage pools in OpenStack.



Pools management

Logged in as: admin [Settings](#) [Help](#) [Sign Out](#)

create pools [create pool](#)

| <input type="checkbox"/> | cluster | pool name | crush rule num | create time | aid | action |
|--------------------------|---------|-----------|----------------|-------------------|-------|---------------------------------|
| <input type="checkbox"/> | default | OSD pool | 1 | 2013-9-4 13:23:33 | aid.3 | create snapshot |

increase capacity [add OSD](#)

| <input type="checkbox"/> | OSD name | size | host name | status | in/out cluster |
|--------------------------|-----------|------|-----------|--------|----------------|
| <input type="checkbox"/> | ceph_osd1 | 500G | OSD_host1 | up | in |

access control [save](#)

| <input type="checkbox"/> | OSD | MON | MDS |
|--------------------------|-------------------|--------------------|---------------------|
| <input type="checkbox"/> | R | RW | RWX |

Figure 3.9 The management of storage pools.