CEPH FileSystem

Reported by: **An Pham**
Course: **Cluster - Grid - Cloud Computing**
ITMO University - St. Petersburg, Russia
Instructed by: **Prof. Andrey Y Shevel**
June 7, 2018

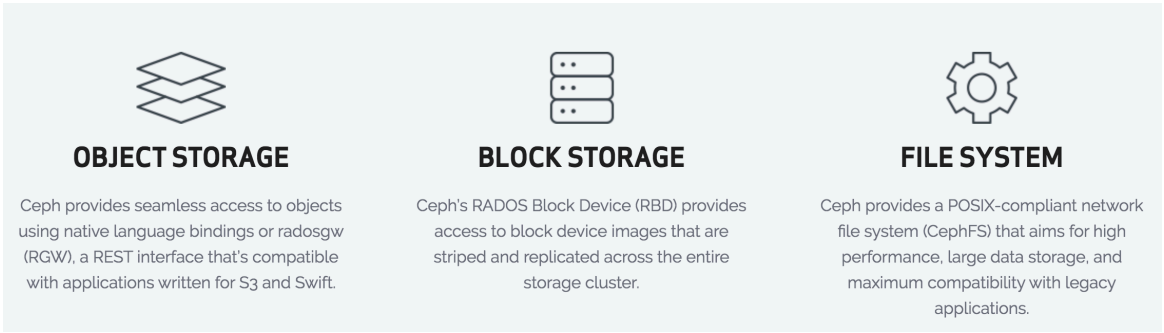
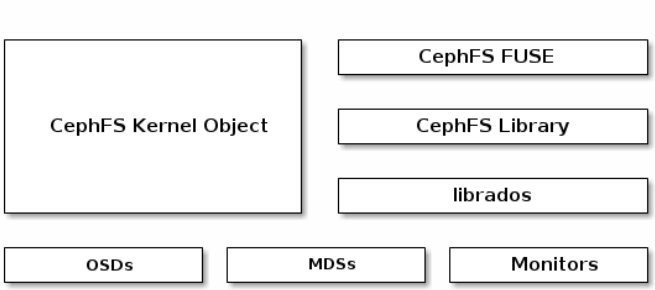


Table of Content

CEPH - DISTRIBUTED FILE SYSTEM

Introduction

As the size and performance requirements of storage systems have increased, file system designers have looked to new architectures to facilitate system scalability. **Ceph** is a fully **open source distributed file system** supporting block, object and file based storage, designed for reliability, performance, and scalability from terabytes to exabytes. It consists of **MON** nodes, **OSD** nodes and optionally an **MDS** node. All of these are fully distributed, and may run on the same set of servers. Clients directly interact with all of them.

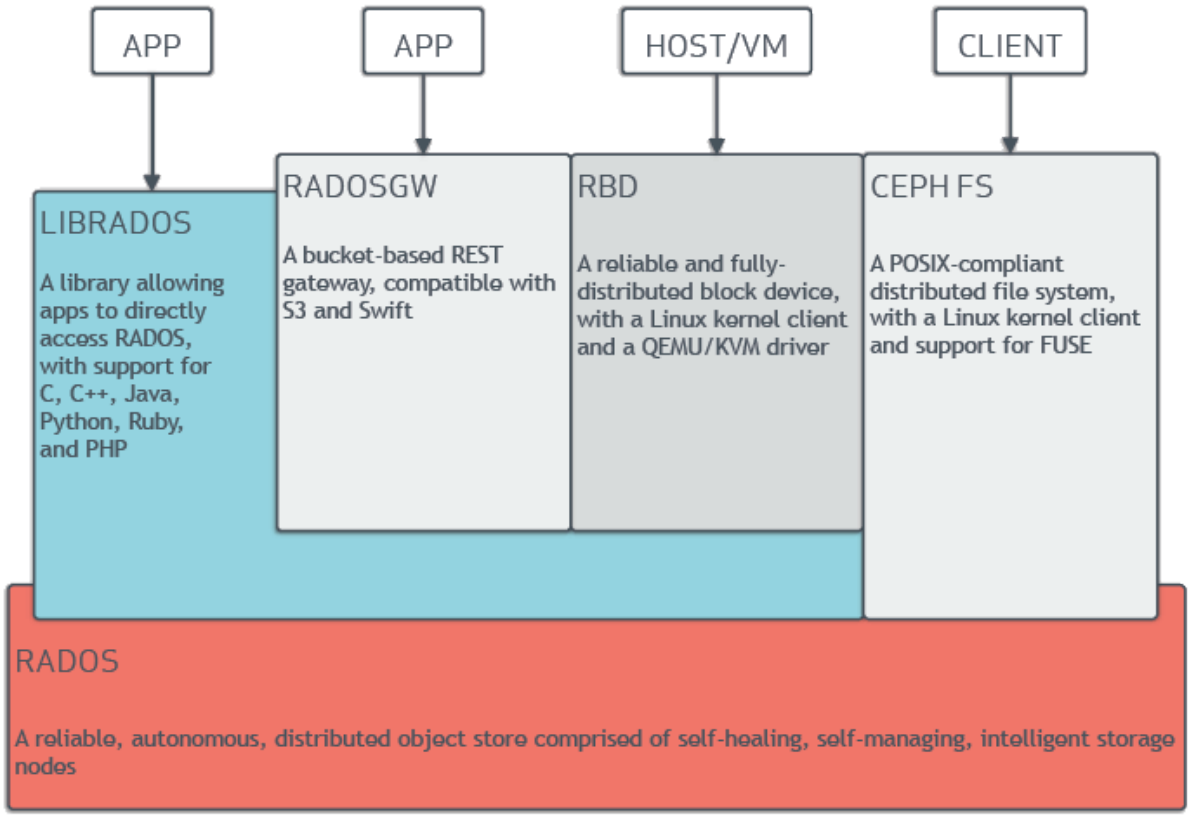


Ceph utilizes a novel placement algorithm (CRUSH), active storage nodes, and peer-to-peer gossip protocols to avoid the scalability and reliability problems associated with centralized controllers and lookup tables. Ceph is part of a tremendous and growing ecosystem where it is integrated in virtualization platforms (Proxmox), Cloud platforms (OpenStack, CloudStack, OpenNebula), containers (Docker), and big data (Hadoop, as a meted server for HDFS).

- **Monitors**: A Ceph Monitor maintains maps of the cluster state, including the monitor map, manager map, the OSD map, and the CRUSH map. These maps are critical cluster state required for Ceph daemons to coordinate with each other. Monitors are also responsible for managing authentication between daemons and clients. At least three monitors are normally required for redundancy and high availability.
- **Managers**: A Ceph Manager daemon (`ceph-mgr`) is responsible for keeping track of runtime metrics and the current state of the Ceph cluster, including storage utilization, current performance metrics, and system load. At least two managers are normally required for high availability.
- **Ceph OSDs**: A Ceph OSD (object storage daemon, `ceph-osd`) stores data, handles data replication, recovery, rebalancing, and provides some monitoring information to Ceph Monitors and Managers by checking other Ceph OSD Daemons for a heartbeat. At least 3 Ceph OSDs are normally required for redundancy and high availability.
- **MDSs**: A Ceph Metadata Server (MDS, `ceph-mds`) stores metadata on behalf of the Ceph Filesystem (i.e., Ceph Block Devices and Ceph Object Storage do not use MDS). Ceph Metadata Servers allow POSIX file system users to execute basic commands (like `ls`, `find`, etc.) without placing an enormous burden on the Ceph Storage Cluster.

Fault tolerance is a key challenge for both system design and operations. Ceph is designed to be both highly available and elastic. In large clusters, disk, host, and even network failures are the norm rather than the exception, hardware is heterogeneous and incrementally deployed or de-provisioned, and availability must be continuous.

ARCHITECTURE



CRUSH Introduction

Ceph Clients and Ceph OSD Daemons both use the CRUSH algorithm to efficiently compute information about object location, instead of having to depend on a central lookup table. CRUSH provides a better data management mechanism compared to older approaches, and enables massive scale by cleanly distributing the work to all the clients and OSD daemons in the cluster. CRUSH uses intelligent data replication to ensure resiliency, which is better suited to hyper-scale storage. The following sections provide additional details on how CRUSH works.

The CEPH Storage Cluster

Ceph uniquely delivers object, block, and file storage in one unified system. Ceph provides an infinitely scalable **Ceph Storage Cluster** based upon **RADOS** (Reliable Autonomic Distributed Object Store)

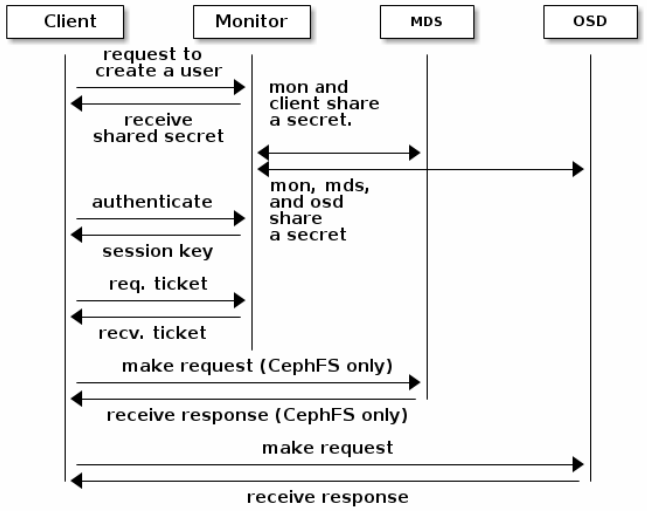
> **Storing Data**: The Ceph Storage Cluster receives data from Ceph Clients–whether it comes through a Ceph Block Device, Ceph Object Storage, the Ceph Filesystem or a custom implementation you create using librados – and it stores the data as objects.
>
> **Scalability and High Availability**: In traditional architectures, clients talk to a centralized component (e.g., a gateway, broker, API, facade, etc.), which acts as a single point of entry to a complex subsystem. This imposes a limit to both performance and scalability, while introducing a single point of failure (i.e., if the centralized component goes down, the whole system goes down, too). Ceph eliminates the centralized gateway to enable clients to interact with Ceph OSD Daemons directly. Ceph OSD Daemons create object replicas on other Ceph Nodes to ensure data safety and high

availability. Ceph also uses a cluster of monitors to ensure high availability. To eliminate centralization, Ceph uses an algorithm called CRUSH.

High Availability Monitors: Before Ceph Clients can read or write data, they must contact a Ceph Monitor to obtain the most recent copy of the cluster map. For added reliability and fault tolerance, Ceph supports a cluster of monitors. In a cluster of monitors, latency and other faults can cause one or more monitors to fall behind the current state of the cluster. For this reason, Ceph must have agreement among various monitor instances regarding the state of the cluster. Ceph always uses a majority of monitors and the Paxos[1] algorithm to establish a consensus among the monitors about the current state of the cluster.

High Availability Authentication:

To identify users and protect against man-in-the-middle attacks, Ceph provides its ***cephx*** authentication system to authenticate users and daemons. The authentication process is described in detail in the following graph. Cephx uses shared secret keys for authentication, meaning both the client and the monitor cluster have a copy of the client's secret key. The authentication protocol is such that both parties are able to prove to each other they have a copy of the key without actually revealing it. This provides mutual authentication, which means the cluster is sure the user possesses the secret key, and the user is sure that the cluster has a copy of the secret key. More details are found in Ceph documentation [1].



Smart Daemons Enable Hyper scale: In many clustered architectures, the primary purpose of cluster membership is so that a centralized interface knows which nodes it can access. Then the centralized interface provides services to the client through a double dispatch – which is a **huge** bottleneck at the petabyte-to-exabyte scale. Ceph eliminates the bottleneck: Ceph's OSD Daemons AND Ceph Clients are cluster aware. Like Ceph clients, each Ceph OSD Daemon knows about other Ceph OSD Daemons in the cluster. This enables Ceph OSD Daemons to interact directly with other Ceph OSD Daemons and Ceph monitors. Additionally, it enables Ceph Clients to interact directly with Ceph OSD Daemons.

Dynamic Cluster Management: The Ceph storage system supports the notion of 'Pools', which are logical partitions for storing objects. Ceph Clients retrieve a Cluster Map from a Ceph Monitor, and write objects to pools. Each pool has a number of placement groups. CRUSH maps PGs to OSDs dynamically. When a Ceph Client stores objects, CRUSH will map each object to a placement group. With a copy of the cluster map and the CRUSH algorithm, the client can compute exactly which OSD to use when reading or writing a particular object.

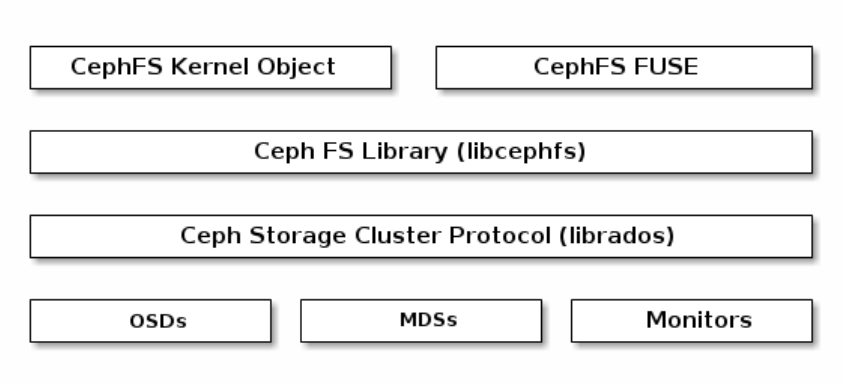
[1] Paxos: http://en.wikipedia.org/wiki/Paxos_(computer_science)

CEPH FILE SYSTEM (CEPH FS)

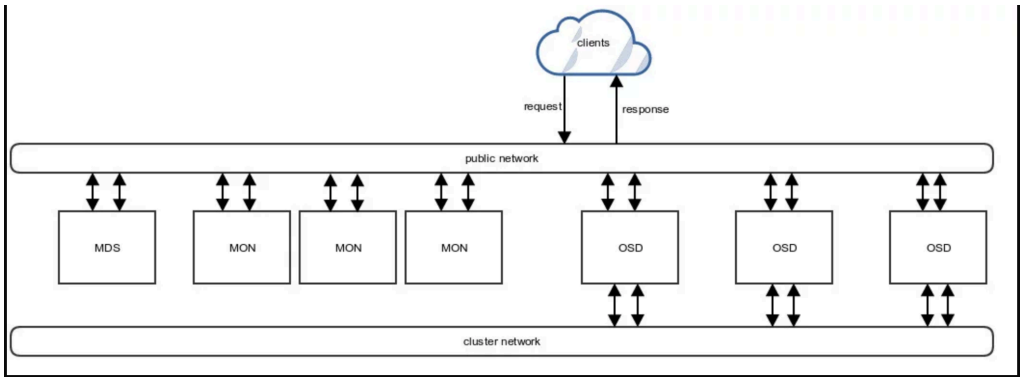
Structure

Ceph's file system (CephFS) runs on top of the same object storage system that provides object storage and block device interfaces. The Ceph metadata server cluster provides a service that maps the directories and file names of the file system to objects stored within RADOS clusters. The metadata server cluster can expand or contract, and it can rebalance the file system dynamically to distribute data evenly among cluster hosts. This ensures high performance and prevents heavy loads on specific hosts within the cluster.

The *Ceph Filesystem* (Ceph FS) is a POSIX-compliant filesystem that uses a Ceph Storage Cluster to store its data. The Ceph filesystem uses the same Ceph Storage Cluster system as Ceph Block Devices, Ceph Object Storage with its S3 and Swift APIs, or native bindings (librados).



CEPH Filesystem structure



CEPH Data Transfering

Pros

- Scalable storage system: as an object-based storage system, CEPH has the possibility to build much **larger storage clusters**: scalable, reliable and fault-tolerate.
- Big Community: CEPH is an open source system and managed by Red Hat - the world leading multinational software company providing open-source software products to the enterprise community. Thus, CEPH is supported and contributed by big community of organizations and companies as well as open source developer.
- **Cloud Platform compatibility**: Third party cloud provisioning platforms such as **OpenStack**, CloudStack, OpenNebula, ProxMox, etc.

- Multi type storage: CEPH offers file-, block- and object-based storage. The Ceph Storage Cluster receives data from Ceph Clients no matter where the data comes from - a Ceph Block Device, Ceph Object Storage, the Ceph Filesystem or a custom implementation you create using librados - CEPH stores the data as objects.
- Better transfer speed and lower latency if compared to other file system (e.g: Swift in this case) – because traffic to and from the Swift cluster goes through proxy servers which slow it down.

Cons

- Design to be highly scalable distributed system, thus, setting up and deploying CEPH storage cluster is time-consuming and requires deep understanding and engineer skills. Although CEPH has big community, troubleshooting tips are not widely available.
- Ceph's multi-region support – which is usually touted as an advantage - is in a master slave configuration, but as replication is only possible from master to slave, in a deployment with more than 2 regions you can get uneven load distribution.
- There can also be a security issue as RADOS clients on the Cloud compute node communicate directly with the RADOS servers over the same network Ceph uses for unencrypted replication traffic. So, potentially, if Ceph client node is compromised, the attacker can see all traffic on the storage network.

Pricing

As an open source project, CEPH is free to download and set up. However, in order to maintain the storage cluster and multiple OSDs, users usually choose cloud platform as a service base. For example, in ubuntu supporting store, CEPH is included as part of OpenStack consulting package, thus, the price to use is included as an integration. [2]

OpenStack consulting packages

Canonical delivers turnkey OpenStack software solutions, on-site, through its field consulting team. Two plans are offered: the Foundation Cloud, which is based on our award-winning reference architecture, and Foundation Plus, which packs in a complete set of enterprise features and architectural flexibility.

	Foundation Cloud	Foundation Cloud Plus
One-time price	$75,000	$150,000
Expert delivery of an Ubuntu OpenStack cloud	Reference architecture	Custom architecture
On-site workshop and training	2-days	4-days
Hypervisor options	KVM and LXD	KVM, LXD, and Hyper-V
Software Defined Networking options	OVS (GRE and/or VXLAN)	OVS with Provider Networks, BGP, DVR *
Software Defined Storage options	Ceph	Ceph, Swift **
Encryption		Control plane and storage

[2] https://www.ubuntu.com/support/plans-and-pricing#openstack

Use cases / Reference Architectures

As mentioned above, CEPH is currently used by many big company or organizations. Here are some highlight best practice or use case implementations of Ceph Software and supporting hardware by commercial providers and experts in the field.

- [2018] – Red Hat Ceph Storage on Intel Processors and SSDs
- [2017] – Red Hat Ceph Storage on QCT Servers: Object Storage Performance and Sizing Guide
- [2017] – Understanding a Multi-Site Ceph Gateway Installation
- [2017] – Hyper Converged Red Hat OpenStack Platform 10 and Red Hat Ceph Storage 2
- [2017] – Dell EMC DSS 7000 with Red Hat Ceph Storage 2
- [2016] – High-Performance Cluster Storage for IOPS Workloads
- [2016] – Ceph on NetApp E-Series
- [2016] – MySQL Databases on Red Hat Ceph Storage
- [2016] – Red Hat Ceph Storage on Dell Poweredge R730XD
- [2016] – Red Hat Ceph Storage on Supermicro Storage Servers
- [2016] – Cisco UCS C3160 high Density Rack Server with Red Hat Ceph Storage
- [2016] – Accelerating Ceph for Database Workloads With an All PCIe SSD Cluster
- [2016] – Red Hat Ceph Storage on QCT Servers
- [2016] – Cisco UCS Integrated Infrastructure with RHEL OSP and Red Hat Ceph Storage
- [2014] – Ceph@Home: The domestication of a wild cephalopod

HANDS-ON LAB

In order to know CEPH better, I followed a hands-on lab hosted by Redhat, the purpose is to reuse a pre-setup online environment provided by Redhat to skip all the painful preflight configuration to focus on CEPH deploying process only. The lab environment has 6 nodes in total and you will be using `ceph-admin` node most of the time.

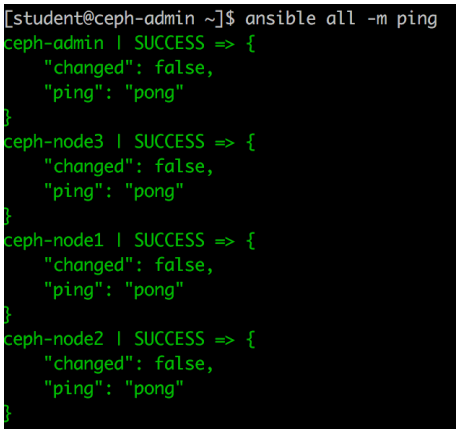
Node Name	Function
ceph-admin	Ceph (Ansible + Metrics + RGW + Client)
ceph-node1	Ceph (MON + Filestore OSD)
ceph-node2	Ceph (MON + Filestore OSD)
ceph-node3	Ceph (MON + Filestore OSD)
ceph-node4	Ceph (Bluestore OSD)
client-node1	Ceph Client

Deploying CEPH Cluster

- In this module we will be deploying Red Hat Ceph Storage 3 cluster across 3 nodes using Ansible based deployer called **`ceph-ansible.`** Ensure that Ansible can reach to all the cluster nodes.

```
ansible all -m ping
```

You should see the result like in this screenshot:



Configuring Ceph-Ansible Settings

- Visit `ceph-ansible` main configuration directory.

```
cd /usr/share/ceph-ansible/group_vars/
```

- all.yml configuration file most importantly configures
  - Ceph repository, path to RHCS ISO
  - Ceph Monitor network interface ID, public network
  - Ceph OSD backend as `filestore`
  - Ceph RGW port, threads and interface
  - Ceph configuration settings for pools

```
cat all.yml
cat osds.yml
cat clients.yml
```

- all.yml

```
---
dummy:
fetch_directory: ~/ceph-ansible-keys
ceph_repository_type: iso
ceph_origin: repository
ceph_repository: rhcs
ceph_rhcs_version: 3
ceph_rhcs_iso_path: "/home/student/rhceph-3.0-rhel-7-x86_64.iso"

monitor_interface: eth0
mon_use_fqdn: true
public_network: 10.100.0.0/16
osd_objectstore: filestore

radosgw_civetweb_port: 80
radosgw_civetweb_num_threads: 512
radosgw_civetweb_options: "num_threads={{ radosgw_civetweb_num_threads }}"
radosgw_interface: eth0
radosgw_dns_name: "ceph-admin"

ceph_conf_overrides:
  global:
    osd pool default pg num: 64
    osd pool default pgp num: 64
    mon allow pool delete: true
    mon clock drift allowed: 5
    rgw dns name: "ceph-admin"
```

- osds.yml

```
[student@ceph-admin group_vars]$ cat osds.yml
---
dummy:
copy_admin_key: true
osd_auto_discovery: true
osd_scenario: collocated
```

- clients.yml

```
[student@ceph-admin group_vars]$ cat clients.yml
---
dummy:
copy_admin_key: True
```

Deploying RHCS Cluster

- To start deploying RHCS cluster, switch to `ceph-ansible` root directory

```
cd /usr/share/ceph-ansible
```

- Run `ceph-ansible` playbook

```
time ansible-playbook site.yml
```

- This should usually take 10-12 minutes to complete. Once its done, play recap should look similar to below. Make sure play recap does not show any host run failed.

```
PLAY RECAP *********************************************************************
ceph-admin                 : ok=120  changed=27   unreachable=0    failed=0
ceph-node1                 : ok=165  changed=33   unreachable=0    failed=0
ceph-node2                 : ok=158  changed=31   unreachable=0    failed=0
ceph-node3                 : ok=160  changed=33   unreachable=0    failed=0
```

- Finally check the status of your cluster.

```
sudo ceph -s
```

```
[student@ceph-admin ceph-ansible]$ sudo ceph -s
  cluster:
    id:     dad99667-c0ca-4527-b1a7-3ff6b99085c0
    health: HEALTH_OK

  services:
    mon: 3 daemons, quorum ceph-node1,ceph-node2,ceph-node3
    mgr: ceph-admin(active)
    osd: 12 osds: 12 up, 12 in

  data:
    pools:   0 pools, 0 pgs
    objects: 0 objects, 0 bytes
    usage:   1290 MB used, 1138 GB / 1139 GB avail
    pgs:
```

Success —> At this point you should have a healthy RHCS cluster up and running with 3 x Ceph Monitors, 3 x Ceph OSDs (12 x OSDs), 1 x Ceph Manager.

Deploying Ceph Metrics Dashboard

- Edit Ansible inventory file

```
sudo vim /etc/ansible/hosts
```

- Add the name of the host you like use for Ceph Metrics, in our case we will use `ceph-admin` node. Add the following content to `/etc/ansible/hosts` file.

```
[ceph-grafana]
ceph-admin ansible_connection=local
```

- Switch to Ceph Metrics Ansible directory

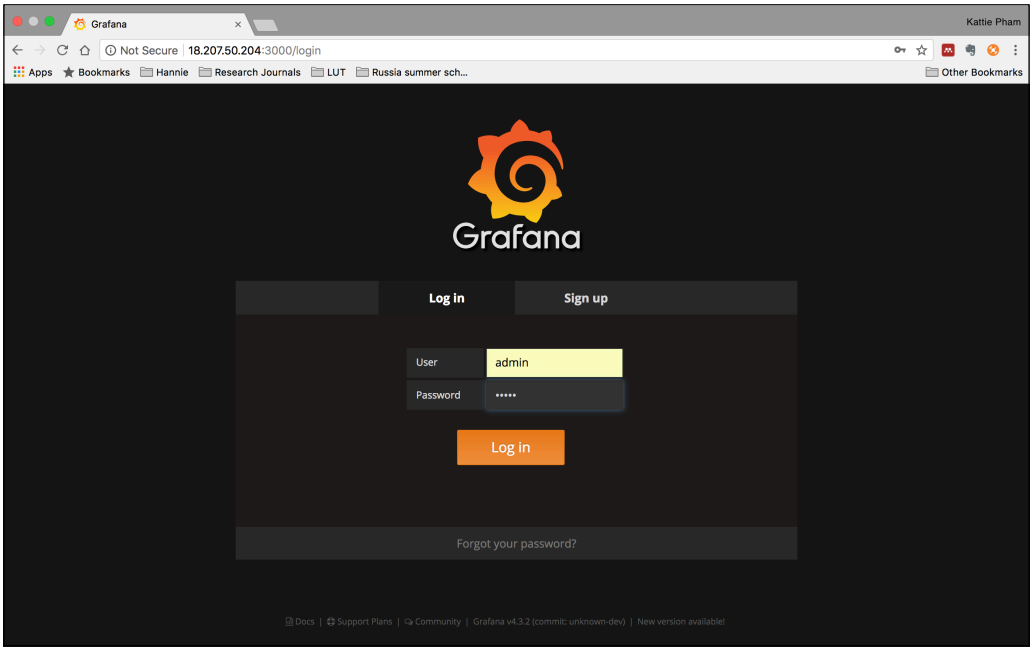
```
cd /usr/share/cephmetrics-ansible
```

- Deploy Ceph Metrics

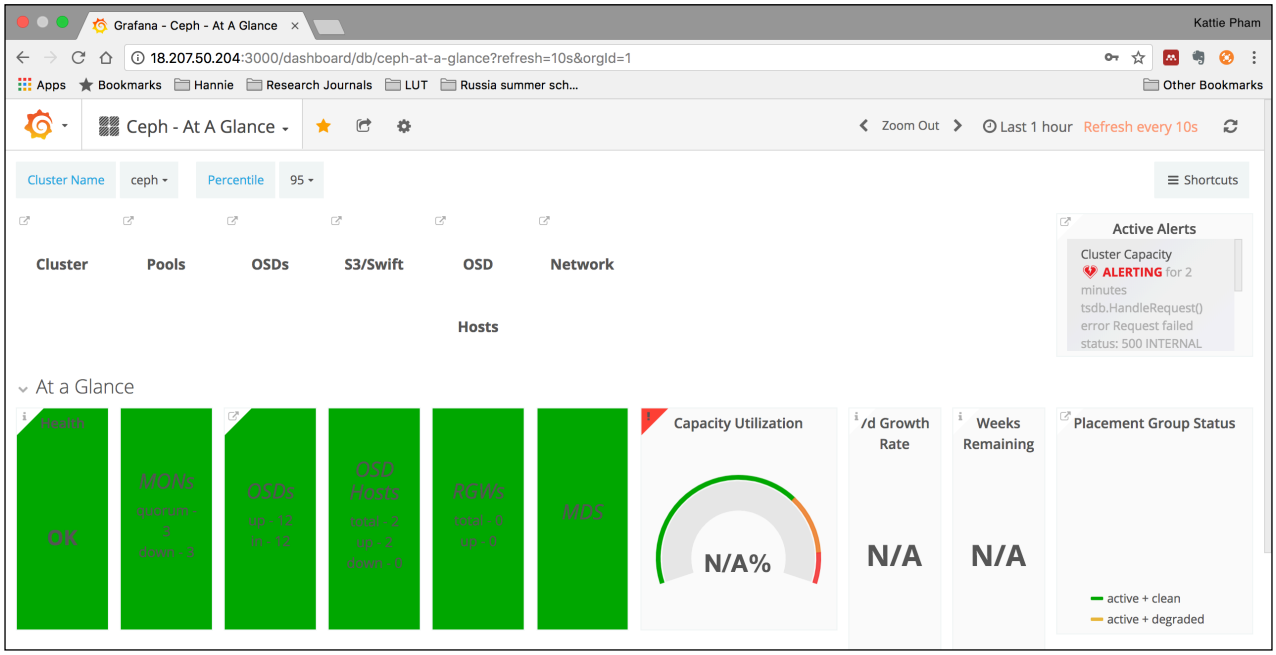
```
time ansible-playbook playbook.yml
```

Working with CEPH Metrics Dashboard

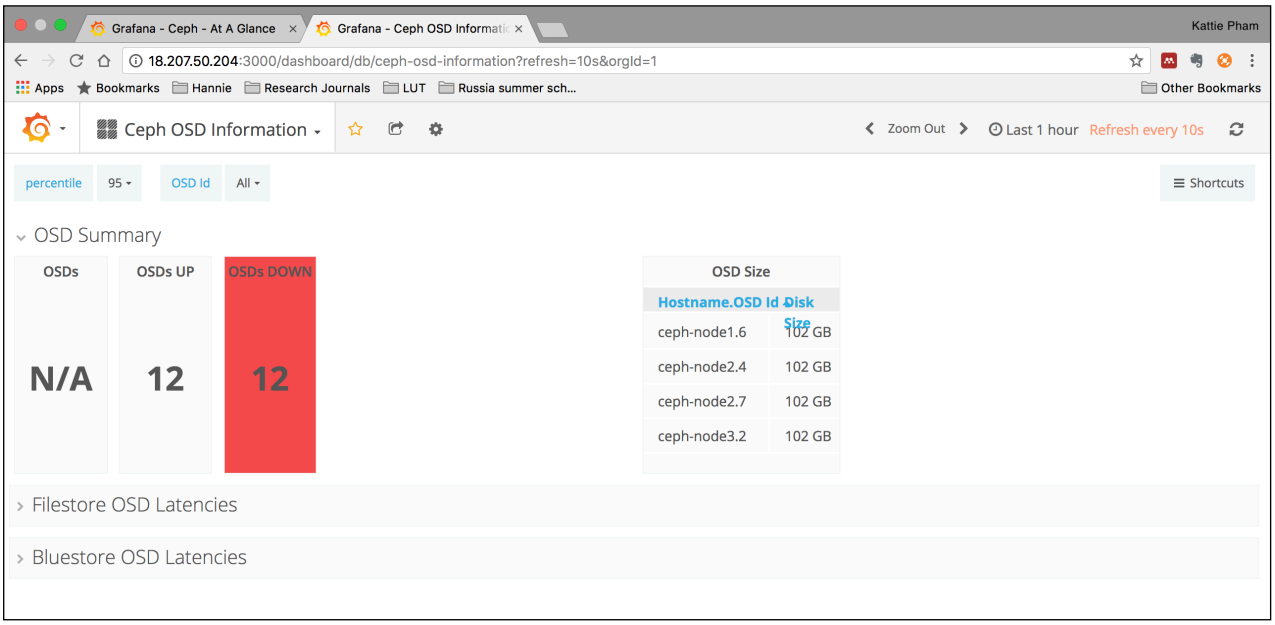
- CEPH metric dashboard will be up and running on port 3000. In my case, it should be: http://18.207.50.204:3000/. You will be asked to login using these credential:
  - User Name : `admin`
  - Password : `admin`



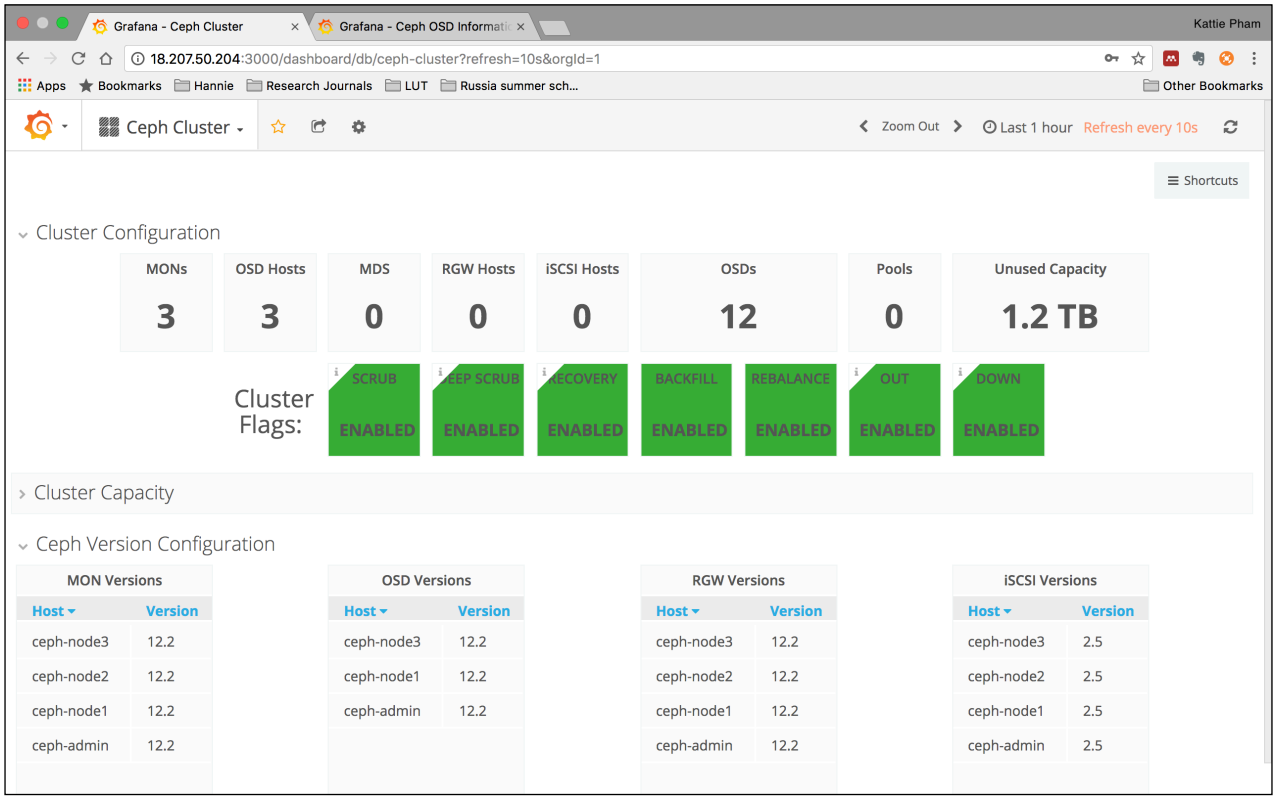
- The Ceph Metrics Dashboard should look like this. You can see info about MONs, OSDs, MDS…



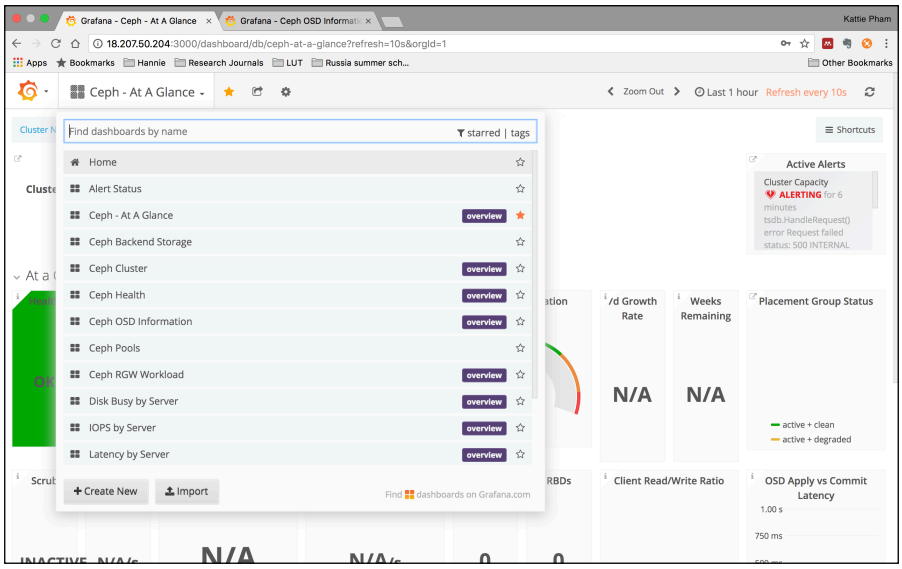
- CEPH OSD information



- CEPH Cluster information



- Under CEPH - At a Glance Menu, you can see a lot of choices for more information



INTERACTING WITH YOUR CEPH CLUSTER

- Check cluster status:

```
ceph -s
```

- Check cluster health

```
ceph health detail
```

- Check Ceph OSD stats and tree view of OSDs in cluster

```
ceph osd stat
ceph osd tree
```

- Check Ceph monitor status

```
ceph mon stat
```

```
[student@ceph-admin cephmetrics-ansible]$ ceph osd stat
12 osds: 12 up, 12 in
[student@ceph-admin cephmetrics-ansible]$ ceph osd tree
ID CLASS WEIGHT  TYPE NAME           STATUS REWEIGHT PRI-AFF
-1       1.11237 root default
-5       0.37079     host ceph-node1
 0   ssd 0.09270         osd.0           up  1.00000 1.00000
 3   ssd 0.09270         osd.3           up  1.00000 1.00000
 6   ssd 0.09270         osd.6           up  1.00000 1.00000
 9   ssd 0.09270         osd.9           up  1.00000 1.00000
-3       0.37079     host ceph-node2
 1   ssd 0.09270         osd.1           up  1.00000 1.00000
 4   ssd 0.09270         osd.4           up  1.00000 1.00000
 7   ssd 0.09270         osd.7           up  1.00000 1.00000
10   ssd 0.09270         osd.10          up  1.00000 1.00000
-7       0.37079     host ceph-node3
 2   ssd 0.09270         osd.2           up  1.00000 1.00000
 5   ssd 0.09270         osd.5           up  1.00000 1.00000
 8   ssd 0.09270         osd.8           up  1.00000 1.00000
11   ssd 0.09270         osd.11          up  1.00000 1.00000
[student@ceph-admin cephmetrics-ansible]$ ceph mon stat
e1: 3 mons at {ceph-node1=10.100.0.11:6789/0,ceph-node2=10.100.0.12:6789/0,ceph-node
3=10.100.0.13:6789/0}, election epoch 4, leader 0 ceph-node1, quorum 0,1,2 ceph-node
1,ceph-node2,ceph-node3
[student@ceph-admin cephmetrics-ansible]$
```

- At this point, we know how to deploy and interact with a CEPH cluster and ready to dig deeper.

REFERENCES

[1] http://docs.ceph.com/docs/giant/
[2] https://ceph.com/
[3] https://www.suse.com/c/ceph-vs-swift-openstack-object-storage-pros-vs-cons-approach-evaluation-flawed-analysis/
[4] http://sdn.ifmo.ru/education/perccom/cluster-grid-clouds/2017/student-reports/Openstack-Ceph%20report%20Gofurov%20F.pdf
[5] https://ezplain.com/2017/09/04/how-to-install-ceph-on-virtualbox-and-deploy-a-sandbox-cloud-cluster/
[6] https://docs.google.com/presentation/d/1AdgxaX5iqROeSuOEHqNz4HwQCKnZGrTK8BcpAYdPdq0/edit?usp=sharing
[7] https://redhat.qwiklab.com/