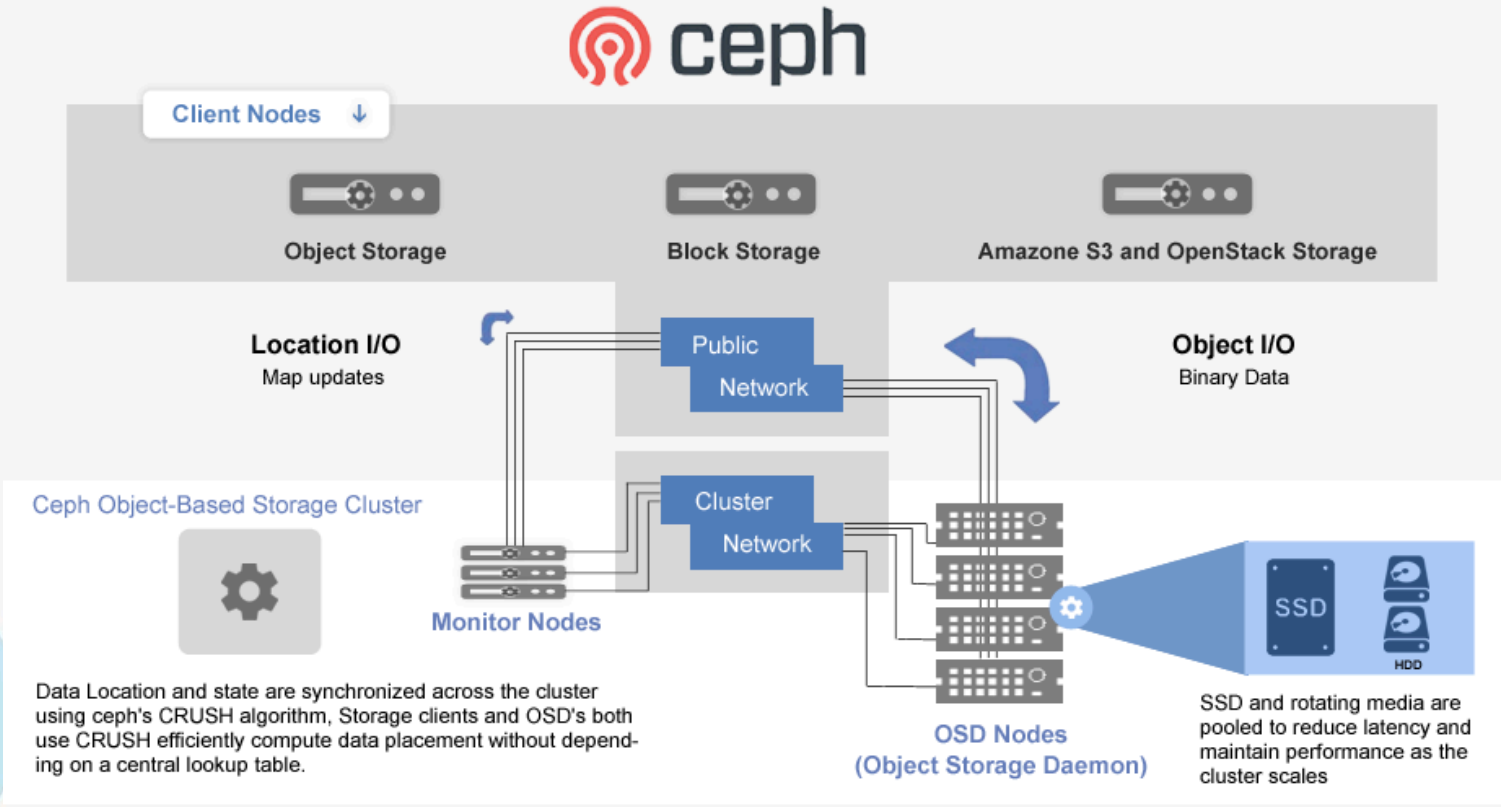
Ceph



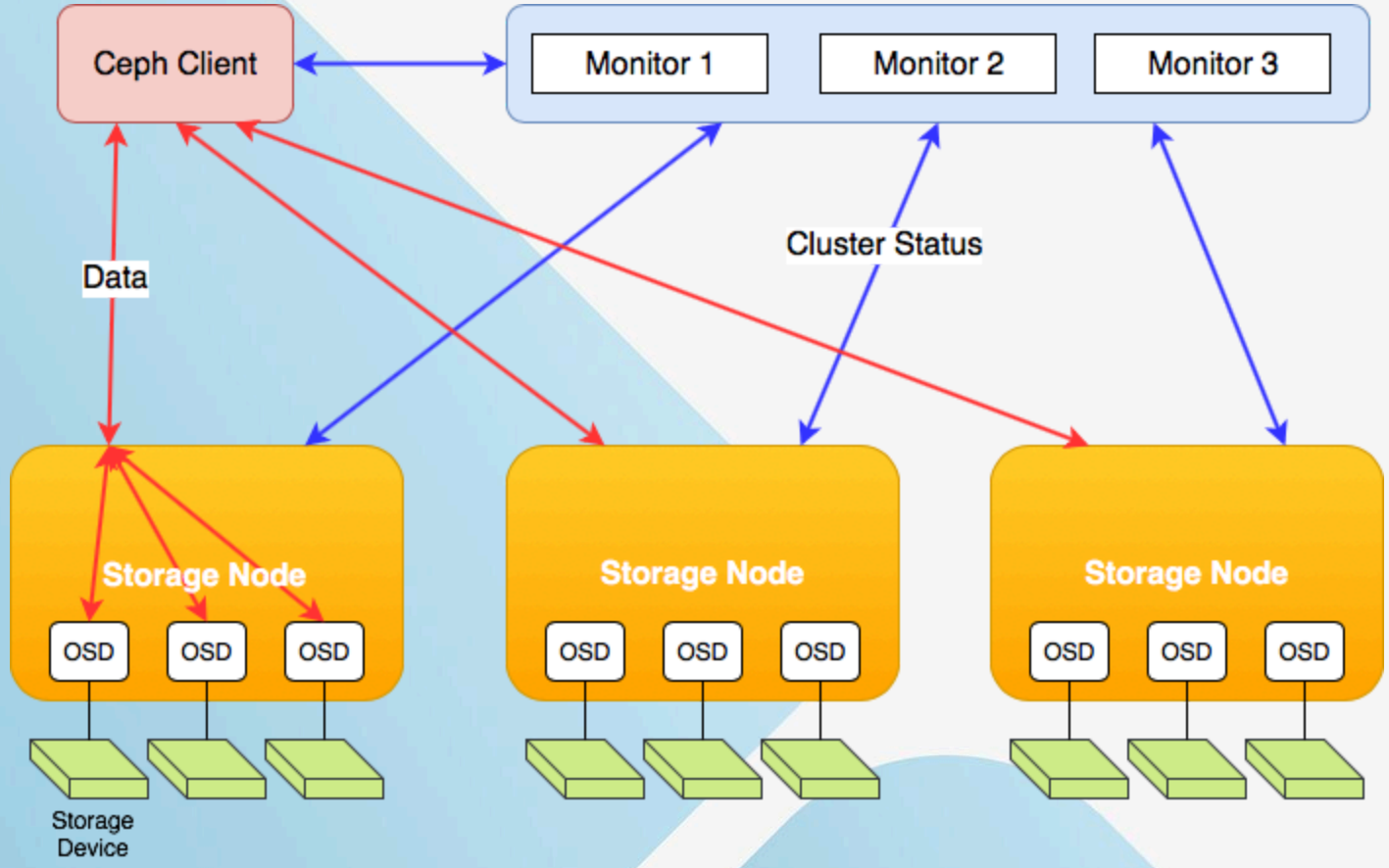
Targets

- Deployment
- Operations
- Networking
- Monitoring
- Troubleshooting
- Testing

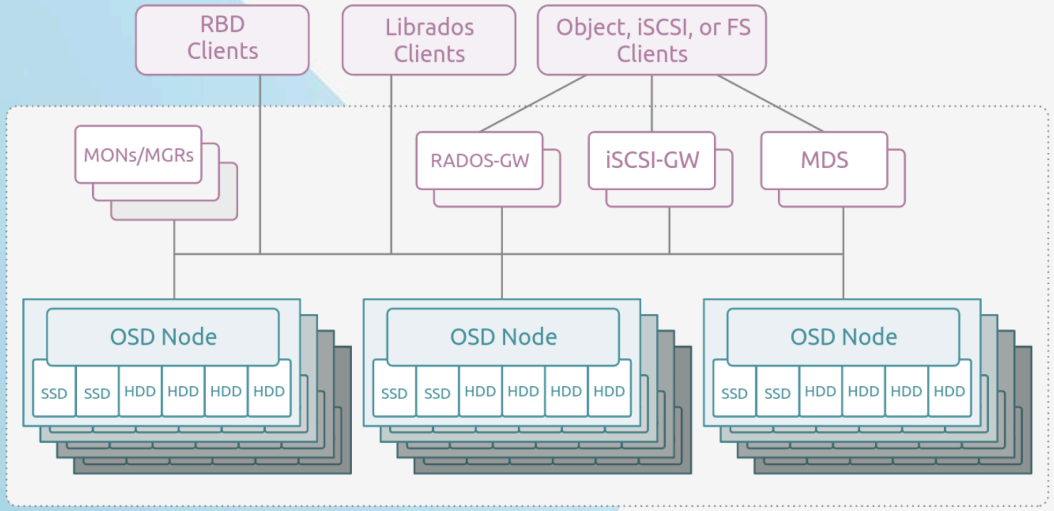
Architecture



Architecture

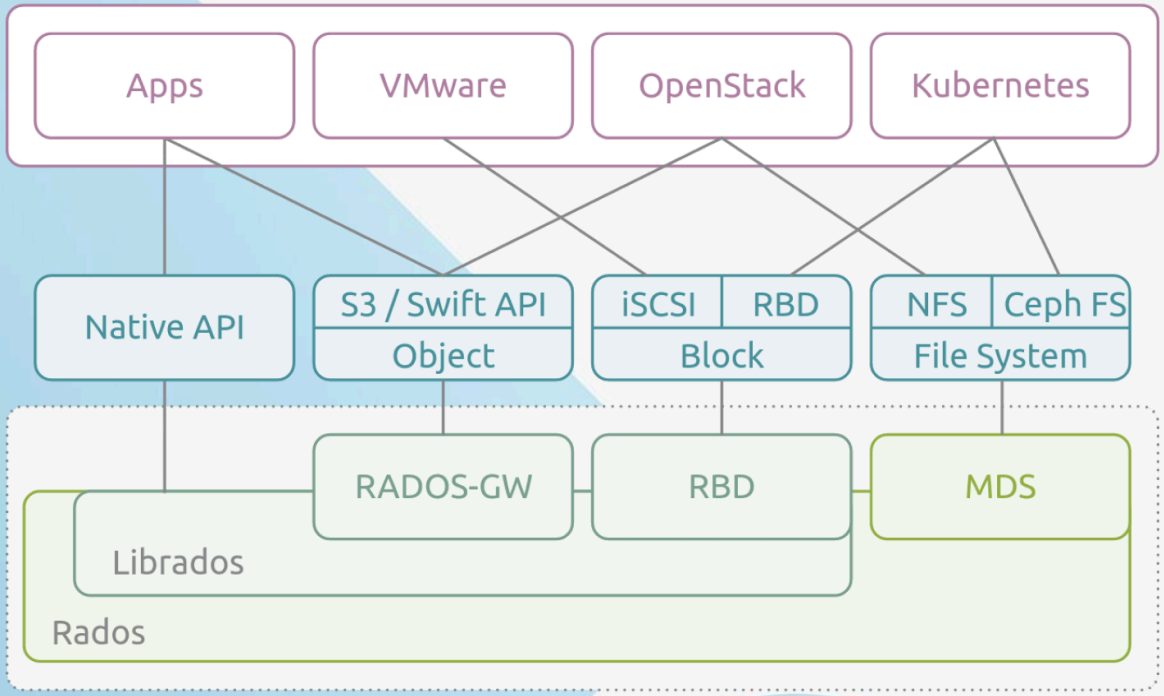


Architecture





Architecture



Users



Deutsche Telekom

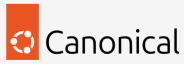
Bloomberg



DreamHost

DigitalOcean

Contributors













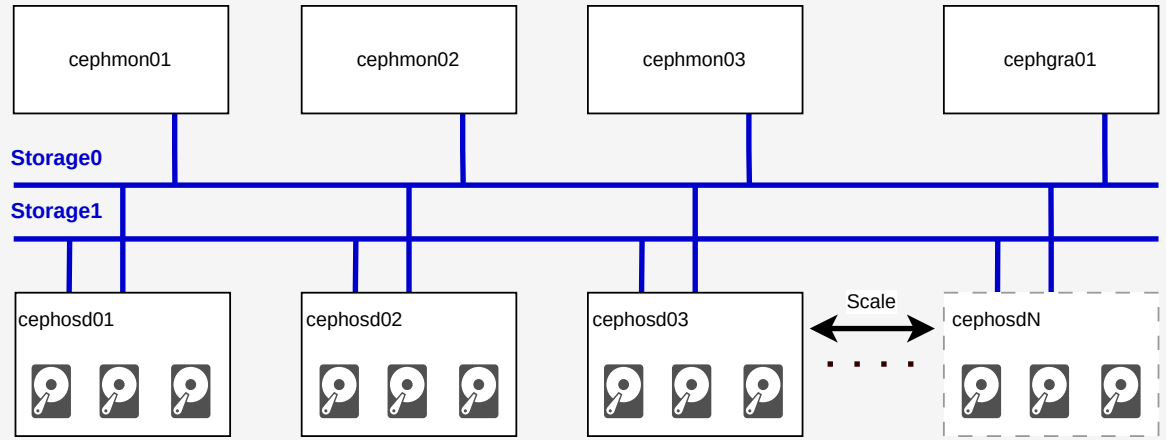


Ceph Deployment Approaches

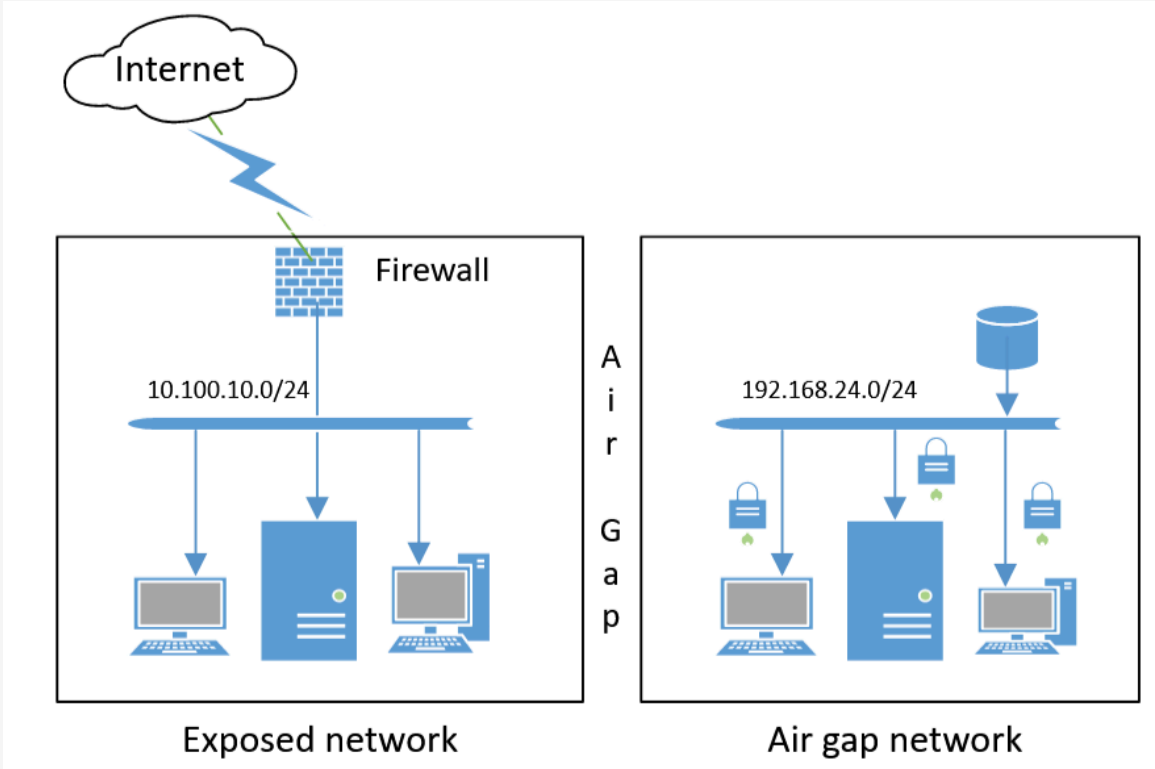
- cephadm (official)
- rook

- ceph-ansible
- ceph-salt
- ceph-deploy (deprecated)
- juju
- manually

Ceph Network



Air-Gapped Environment

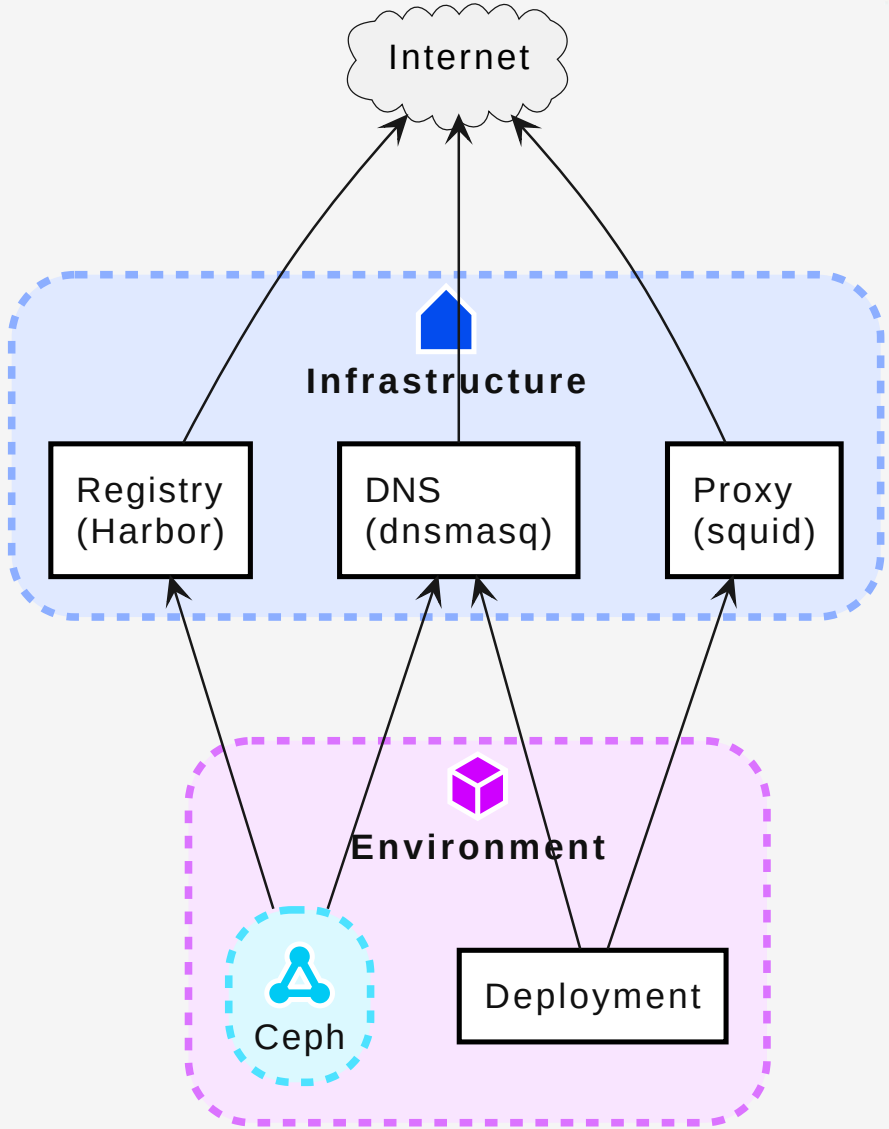


Workshop Hardware and VMs

- Hardware: 20 vCPUs, 40 GB RAM, 1 TB disk
- Deployment VM
- Infra VMs: registry, proxy, dns
- Ceph VMs: 3x mon, 3(+)x osd, 1x Monitoring

Infrastructure

Overview



Deployment Node

- Python virtual envs
- cephadm

Deploy Ceph with Cephadm

```
# as deploy on deployment host
sudo apt install cephadm ceph-common ceph-fuse python3-jinja2 -y

# create ssh keypair for ceph
sudo ssh-keygen -t rsa -N '' -f /etc/ceph/id_rsa

# copy the pub key to ceph nodes
for i in cephmon0{1..3} cephosd0{1..3} cephgra01 deployment
do
  ssh  $i.mgmt 'sudo cp /home/deploy/.ssh/authorized_keys /root/.ssh/authorized_keys'
  ssh-copy-id -f -i /etc/ceph/id_rsa root@$i.mgmt
done
```



Deploy Ceph with Cephadm

```
#  as deploy on deployment host
CEPH_VERSION="19.2.1"
CEPH_FSID="1959570b-e2e3-4017-a5dc-c7606c5068fd"
CEPH_CONFIG="$HOME/ceph/initial-ceph.conf"
CEPH_SPEC="$HOME/ceph/spec.yaml"
CEPH_MON_IP=$(ip -br a | grep strg0 | awk '{print $NF}' | cut -d / -f 1)
CEPH_PUBLIC_NET="10.20.21.0/24"
CEPH_CLUSTER_NET="10.20.22.0/24"
CEPH_IMAGE="registry.lab.nfgd.net/quay/ceph/ceph:v$CEPH_VERSION"
SPEC="$HOME/ceph/spec.yaml"

cd ~/ceph
sudo cephadm --image "$CEPH_IMAGE"                    \
  bootstrap                                           \
    --fsid                      $CEPH_FSID            \
    --config                    $CEPH_CONFIG          \
    --mon-ip                    $CEPH_MON_IP          \
    --cluster-network           $CEPH_CLUSTER_NET     \
    --ssh-private-key           /etc/ceph/id_rsa      \
    --ssh-public-key            /etc/ceph/id_rsa.pub  \
    --initial-dashboard-password  p@ssw0rd            \
    --dashboard-password-noupdate                     \
    --allow-fqdn-hostname                             \
    --skip-firewalld                                  \
    --single-host-defaults

# NOTE: --no-cleanup-on-failure to debug if the bootsrap step fails
#       --verbose
#       --retry <NR>

# wait a moment, check health
sudo ceph -s
```



Deploy Ceph with Cephadm

```
# Ceph cluster was created, but is not running on all target nodes yet

# Review the state on deploy node

sudo docker ps

sudo ceph orch host ls

# Review $CEPH_SPEC file and apply
sudo ceph orch apply -i $CEPH_SPEC

sudo chown deploy:deploy -R /etc/ceph/
```



Deploy Ceph with Cephadm

Create client keys for OpenStack services

```
# as root on deployment node, create the .keyring files
cat > /etc/ceph/ceph.client.glance.keyring <<EOF
[client.glance]
        key = AQBqB/JpuJQKDRAACBv/LtaMfeBa4cG3s/5WCg==
        caps mon = "profile rbd"
        caps osd = "profile rbd pool=volumes, profile rbd pool=images"
EOF
cat > /etc/ceph/ceph.client.cinder.keyring <<EOF
[client.cinder]
        key = AQB6B/JpY78nKRAA9esk98rCEzyNbCPga83DOQ==
        caps mon = "profile rbd"
        caps osd = "profile rbd pool=volumes, profile rbd pool=vms, profile rbd pool=images"
EOF
cat > /etc/ceph/ceph.client.cinder-backup.keyring <<EOF
[client.cinder-backup]
        key = AQCMB/JprCiDDRAAPWyW0wv/yQ2UQrf4M+7zZA==
        caps mon = "profile rbd"
        caps osd = "profile rbd pool=backups"
EOF
cat > /etc/ceph/ceph.client.nova.keyring <<EOF
[client.nova]
        key = AQCgB/JpjgdbKRAA3bYIrDC+PnmvojeEpXClrg==
        caps mon = "profile rbd"
        caps osd = "profile rbd pool=volumes, profile rbd pool=vms, profile rbd pool=images"
EOF
cat > /etc/ceph/ceph.client.openstack.keyring <<EOF
[client.openstack]
        key = AQCvB/JpqAKsLRAAJHTMBMMNpIwD9DbOFx4e/g==
        caps mon = "profile rbd"
        caps osd = "profile rbd pool=images, profile rbd pool=vms, profile rbd pool=volumes, profile rbd pool=backups"
EOF
# Import the .keyring clients
for f in glance cinder cinder-backup nova openstack
do
  chown 0600          /etc/ceph/ceph.client.${f}.keyring
  ceph  auth import -i /etc/ceph/ceph.client.${f}.keyring
done
```



Prepare for OpenStack

Create pools for OpenStack services

```
# as root on deployment host
for p in volumes images vms backups
do
  ceph osd pool create $p
  ceph osd pool application enable $p rbd
  ceph osd pool set $p crush_rule replicated_rule
done
```



Check it out

```
# On deployment node (as root or with sudo)
ceph orch ls
ceph orch host ls
ceph orch ps
ceph orch device ls

ceph -s
ceph osd tree
```



State Overview

- Ceph/Dashboard
- Grafana/Monitoring
- Prometheus/Monitoring
- Alertmanager



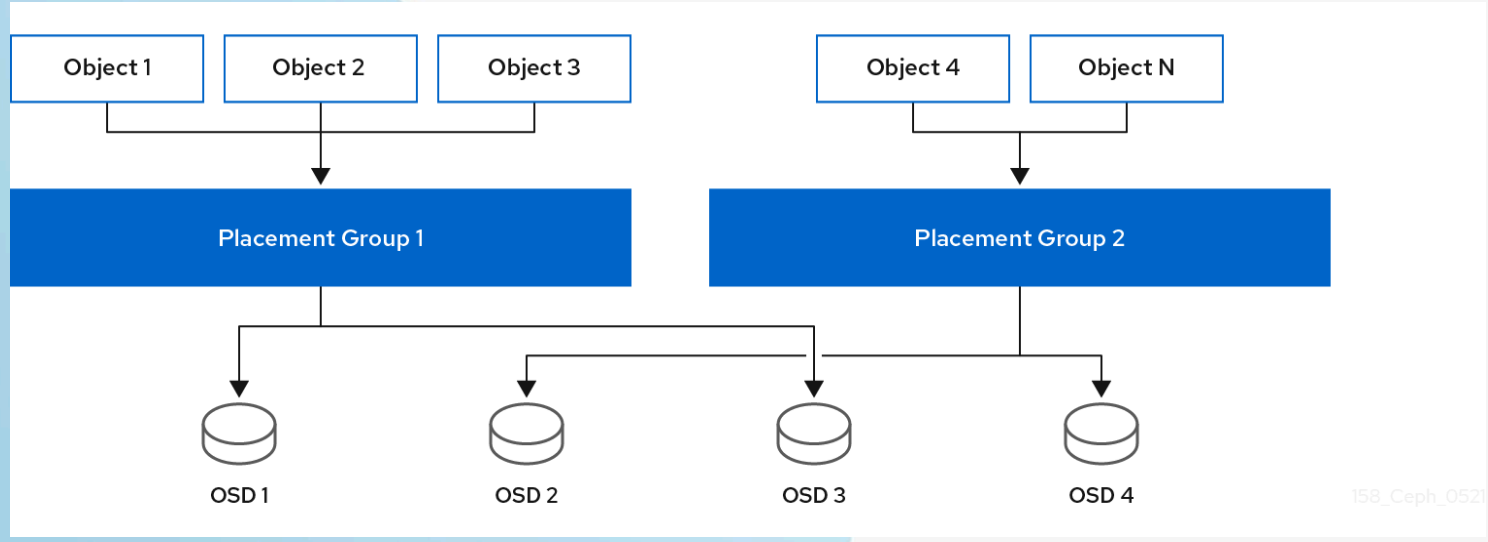
Ceph Basics



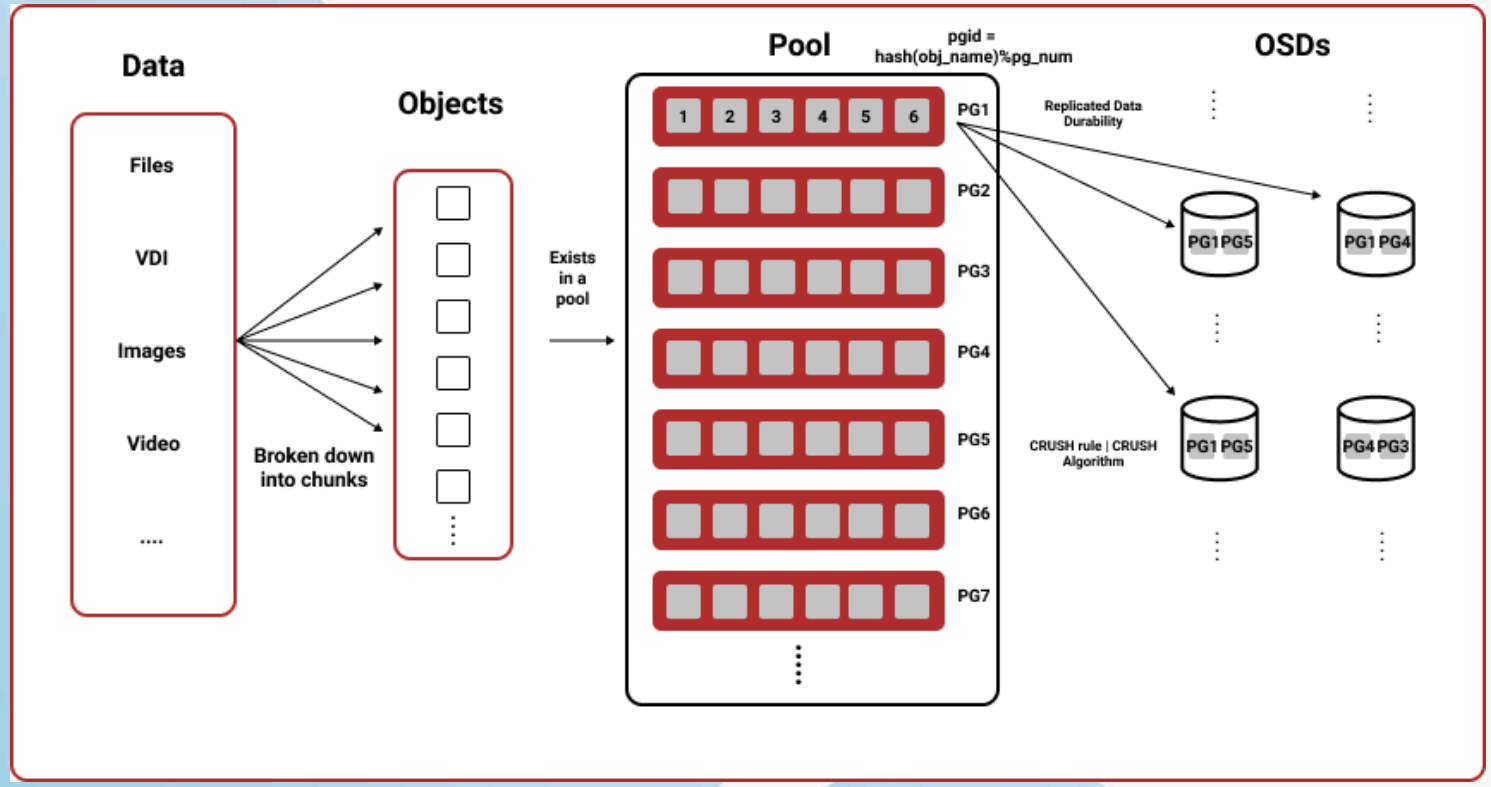
Crush

- map
- rules
- rulesets

Placement Groups

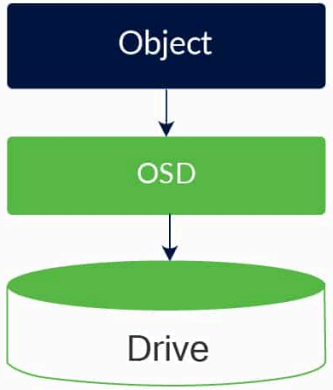


Placement Groups



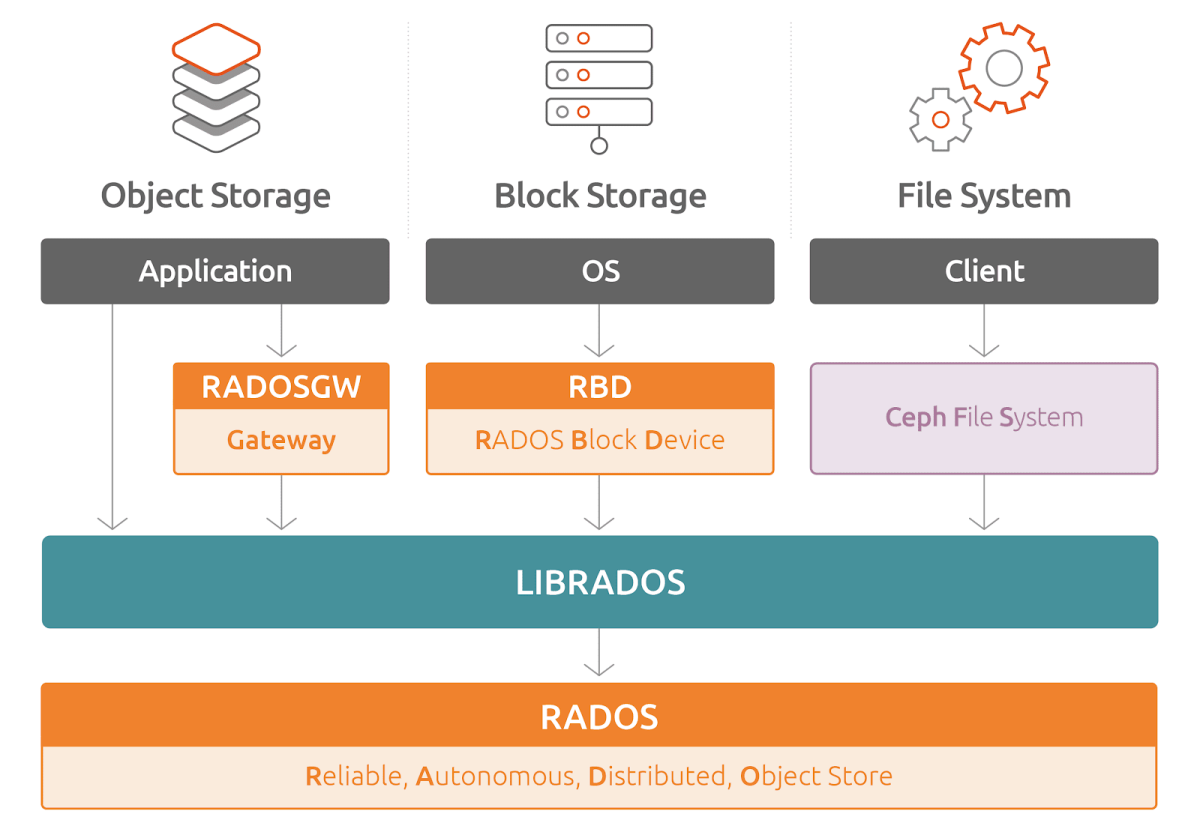
Object Storage Daemon (OSD)

Ceph OSDs store data on behalf of Ceph clients. Additionally, Ceph OSDs utilize the CPU, memory and networking of Ceph nodes to perform data replication, erasure coding, rebalancing, recovery, monitoring and reporting functions.



Pools

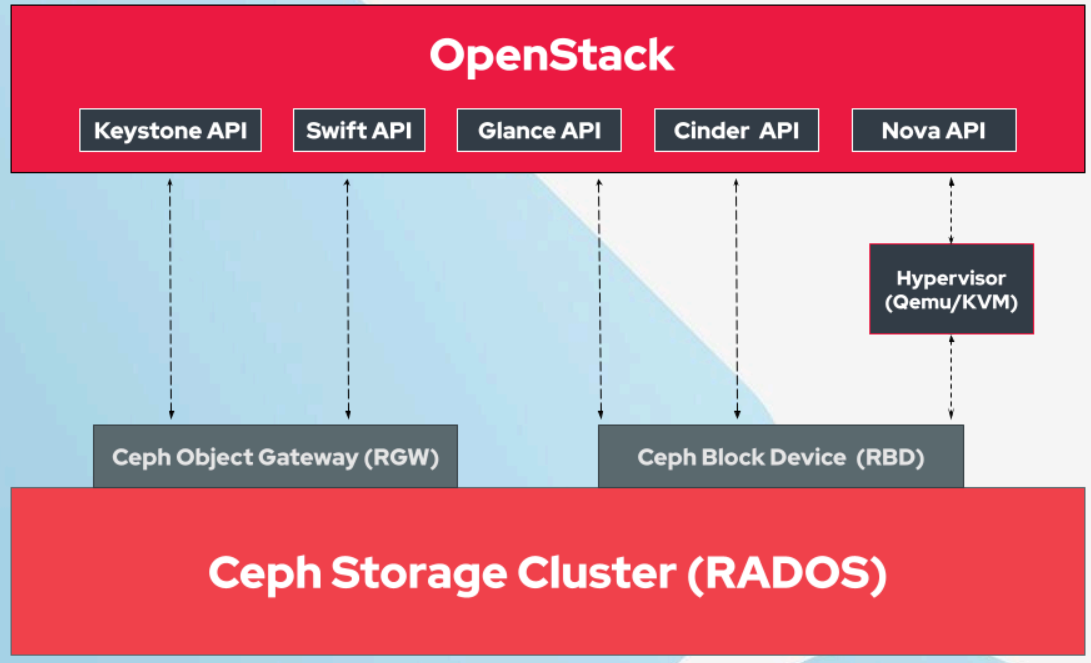
Interfaces





infraguide.org | Questions or need a guided workshop? → ping@infraguide.org | Imprint · Privacy

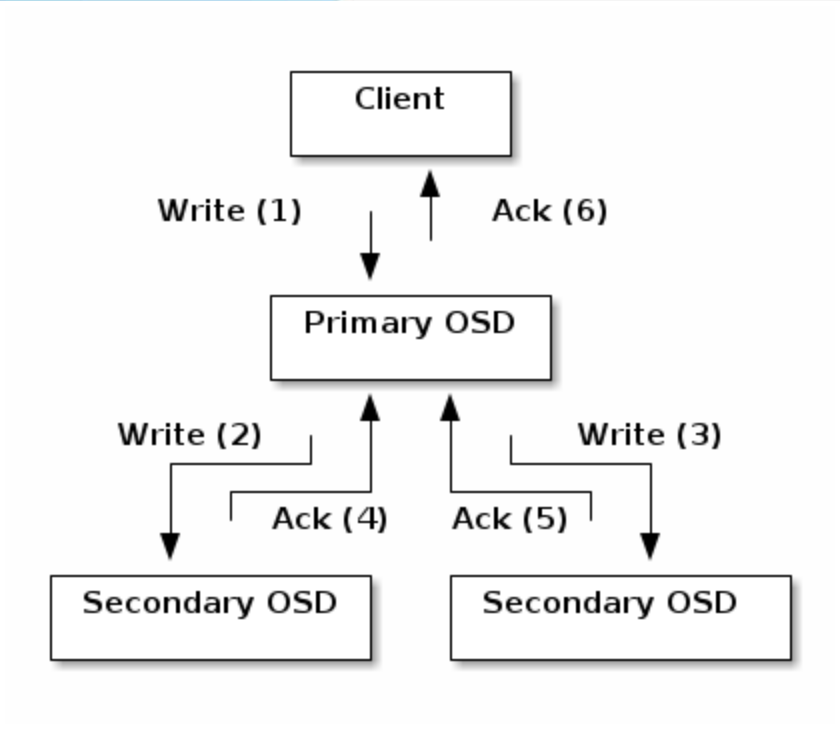
How OpenStack uses Ceph



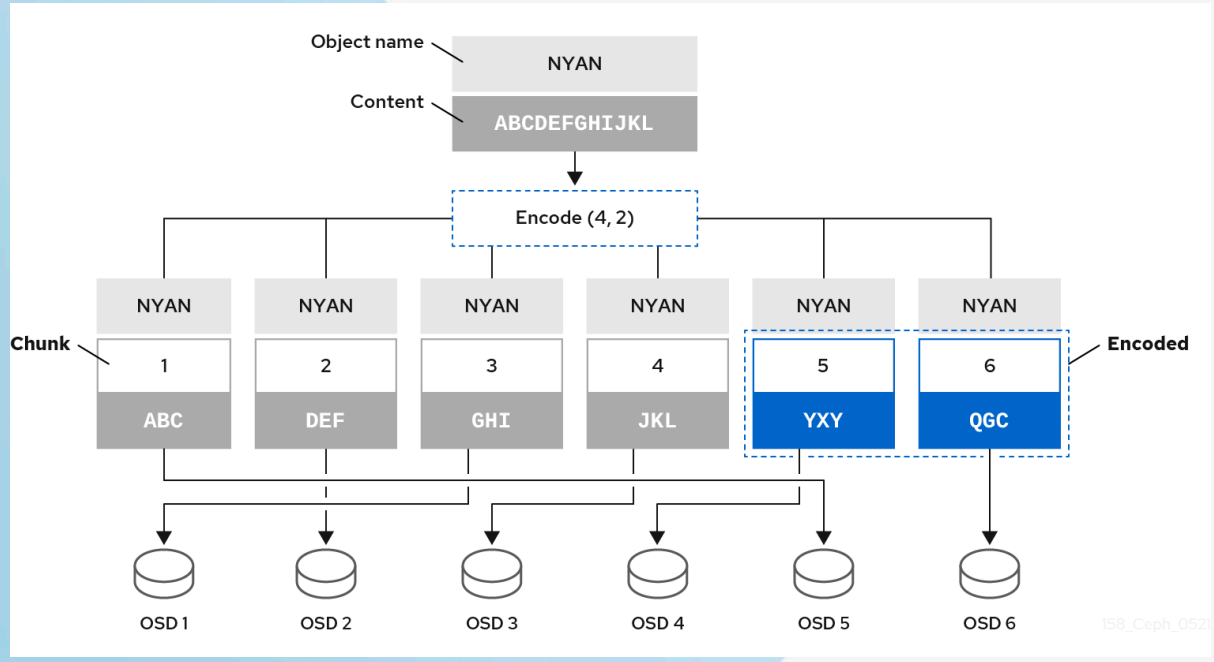


infraguide.org | Questions or need a guided workshop? → ping@infraguide.org | Imprint · Privacy

Replication



Erasure Coding





Cluster Components

Monitor — daemon responsible for maintaining a master copy of the cluster map. A quorum of 3 is needed at minimum to ensure high availability.

OSD (Object Storage Daemon) — daemon that handles all data storage, replication and data recovery operations.

Rados Gateway – The Rados Gateway delivers an API for S3 or Swift to connect directly with Ceph.

Metadata Server — MDS handles all file operations and uses RADOS objects to store data and file system attributes.

Ceph Manager — runs alongside the monitor daemons to provide additional monitoring and interfaces to external monitoring and management systems.

infraguide.org | Questions or need a guided workshop? → ping@infraguide.org | Imprint · Privacy

Cluster Map

Ceph maintains all cluster topology, which includes five maps called the "Cluster Map":

Monitor Map: Contains the fsid of the cluster, the position, the name, the address and the port of each monitor. It also indicates the current time, when the map was created and the last time it was changed. To view a monitor map, run ceph mon dump.

OSD Map: Contains the cluster fsid, when the map was last created and modified, a list of pools, replica sizes, PG numbers, a list of OSDs and their status (for example, up, in and down). To view an OSD map, run ceph osd dump.

Cluster Map

PG Map: Contains the PG version, its timestamp, the last epoch of the OSD map, the complete proportions and details of each placement group, such as PG ID, Up Set, Active, PG status (for example, active + clean) and data usage statistics for each pool.

CRUSH Map: Contains a list of storage devices, the fault domain hierarchy (for example, device, host, rack, row, room, etc.) and rules for traversing the hierarchy when storing data.
You can view the decompiled map in a text editor.

Cluster Map

MDS Map (CephFS): Contains the current timestamp of the MDS map, when the map was created and the last time it was changed. It also contains the pool to store metadata, a list of metadata servers and which metadata servers are active and available. To view an MDS map, run ceph mds dump.

Authentication

Ceph uses a **cephx authentication system** similar to Kerberos to authenticate users and daemons. Both the clients and the monitors in the cluster have a copy of the client's secret key.



```
[client.cinder]
        key = AQB6B/JpY78nKRAA9esk98rCEzyNbCPga83DOQ==
        caps mon = "profile rbd"
        caps osd = "profile rbd pool=volumes, profile rbd pool=vms, profile rbd pool=images"
```

Review the cluster

```
# Cluster status
ceph -s
ceph -w
ceph health [detail]

# cephadm status
ceph orch ls
ceph orch ps
ceph orch ls osd --export
ceph orch host ls

# Versions used
ceph tell mon.* version
ceph tell osd.* version
ceph tell mgr.* version
```



OSDs

```
# Review the state of OSDs
ceph osd status
ceph osd tree
ceph osd df
ceph osd info osd.5
ceph osd crush class ls
ceph osd versions
```



Pools

```
# Review the ods pools
ceph osd pool ls
ceph osd pool get noautoscale
ceph osd pool get-quota volumes
ceph osd pool stats     volumes

ceph osd pool rename ...

ceph osd pool repair [POOL]
ceph osd pool scrub  [POOL]

ceph osd pool get <POOL> size
ceph osd pool get <POOL> crush_rule
ceph osd pool set ...
```



Placement Groups

```
# Review placement groups
ceph pg ls
ceph pg ls-by-pool vms
ceph pg dump
ceph pg dump_stuck unclean
```



Clients

```
# List all entities
ceph auth ls

ceph auth get-or-create client.testuser \
  mon 'allow r' \
  osd 'allow rw pool=test'

ceph auth get-key client.testuser

ceph auth get client.testuser

ceph auth rm client.testuser
```



RBD Volumes

```
# Create a pool named 'vms' for our block devices
ceph osd pool create test 32 32

# Enable the RBD application on that pool
ceph osd pool application enable test rbd

# Create a 10 Gigabyte RBD volume named 'demo-disk'
rbd create test/demo-disk --size 10G
rbd ls test/

# Inspect
rbd info test/demo-disk
rbd map  test/demo-disk
rbd device list
lsblk

# Use
ls -lh /dev/rbd0  /dev/rbd/test/demo-disk
mkfs.ext4 /dev/rbd/test/demo-disk
mount     /dev/rbd/test/demo-disk /mnt/
echo "Hello from Ceph RBD Storage!" > /mnt/test.txt; sync
```



RBD Snapshots

```
# Create a new snapshot
rbd snap create test/demo-disk@snap1
rbd snap list test/demo-disk
rm /mnt/test.txt

# Unmount and unmap the disk to safely roll back
umount /mnt
rbd unmap /dev/rbd/test/demo-disk
ls -lh /dev/rbd*

# Roll back the volume to the snapshot state
rbd snap rollback test/demo-disk@snap1

# Remap and mount to verify the data is back
rbd map test/demo-disk
mount /dev/rbd/test/demo-disk /mnt/
cat /mnt/test.txt

# Unmount, unmap
umount /mnt
rbd unmap /dev/rbd/test/demo-disk
```



File Systems

```
# Create the metadata pool (needs high-speed disks like SSDs if possible)
ceph osd pool create testfs_metadata 32 32

# Create the data pool (where your files actually sit)
ceph osd pool create testfs_data 32 32

# Create new filesystem
ceph fs new testfs testfs_metadata testfs_data
ceph fs status

# Create a user named 'client.guest' with read-write access
ceph auth get-or-create \
  client.guest mon 'allow r' mds 'allow rws' osd 'allow rw pool=testfs_data' \
  -o /etc/ceph/ceph.client.guest.keyring
ceph auth get client.guest

# Mount
ceph-fuse -n client.guest -k /etc/ceph/ceph.client.guest.keyring /mnt/ --client_mds_namespace=testfs
```



File Systems and Snapshots

```
# Create a working project folder
mkdir -p /mnt/project1

# Add a sample document
echo "Version 1.0 - Stable Build" > /mnt/project1/document.txt

# Take a snapshot (the mkdir trick)
mkdir /mnt/project1/.snap/snapshot_backup_v1
cat   /mnt/project1/.snap/snapshot_backup_v1/document.txt

# Overwrite our working live file
echo "Version 2.0 - Broken Build!" > /mnt/project1/document.txt

# Add junk data
echo "Temporary junk" > /mnt/project1/garbage.txt

# Restore the file from snapshot
cp /mnt/project1/.snap/snapshot_backup_v1/document.txt /mnt/project1/document.txt

# NOTE: will not work with rm -rf
rmdir  /mnt/project1/.snap/snapshot_backup_v1
```



RadosGW / S3

```
# Create a user to access S3
radosgw-admin user create --uid=demo-user --display-name="Demo S3 User"

# Search for access_key and secret_key in the output
radosgw-admin user list
radosgw-admin user info --uid  demo-user
```



aws-cli example

```
# Install and configure awscli
python3 -m venv ~/venv/ceph-tools
source ~/venv/ceph-tools/bin/activate
pip3 install -U pip
pip3 install awscli awscli-plugin-endpoint


mkdir ~/.aws
cat > ~/.aws/config <<EOF
[plugins]
endpoint = awscli_plugin_endpoint
[profile default]
s3 =
  endpoint_url = http://s3.strg.lab.nfgd.net
  signature_version = s3v4
  addressing_style = auto
s3api =
  endpoint_url = http://s3.strg.lab.nfgd.net
EOF

cat > ~/.aws/credentials <<EOF
[default]
aws_access_key_id = {ACCESS_KEY}
aws_secret_access_key = {SECRET_KEY}
EOF

# List buckets
aws s3 ls s3:///

# Create a new bucket
aws s3 mb s3://demo-bucket

# Upload a file
echo "This is object data stored in Ceph RGW." > test.txt
aws s3 cp test.txt s3://demo-bucket/project1/test.txt

# Review
aws s3 ls s3://demo-bucket/
aws s3 ls s3://demo-bucket/project1/

# Download
aws s3 cp s3://demo-bucket/project1/test.txt /tmp/test.txt
cat  /tmp/test.txt
```



mc example

```
# Install and configure mc client
TAG=$(
  curl -fsSL --proxy proxy.lab.nfgd.net:3128 \
      https://api.github.com/repos/minio/mc/releases/latest | jq -Mr .tag_name
)

curl -fsSL -o mc "https://github.com/minio/mc/releases/download/${TAG}/mc.linux-amd64.${TAG}"
chmod +x mc && sudo mv mc /usr/local/bin/


# Configure
mc alias set s3 http://s3.strg.lab.nfgd.net {ACCESS_KEY} {SECRET_KEY}
mc alias ls
mc --autocompletion
source ~/.bashrc

# Test
mc ls  s3
mc cat s3/demo-bucket/project1/test.txt
```



Operations

** Make the cluster alive*

```
ceph osd pool create benchmark

rbd create  --size 10G benchmark/test
rbd config image set   benchmark/test rbd_qos_bps_limit 1048576 # 1 MiB/s

# start the benchmark process
rbd bench --io-type write --io-pattern rand --io-size 4K --io-threads 4 --io-total 1T benchmark/test

# to remove the limit excute
rbd config image remove benchmark/test rbd_qos_bps_limit
```



Operations

Update system packages and reboot

```
# Stop ceph services on cephosd03 as root
docker ps
systemctl stop ceph.target
docker ps

# Check status on deployment as root
ceph status
ceph health detail
ceph osd tree
ceph pg dump
ceph pg dump_stuck
ceph pg dump_stuck unclean

ceph osd set noout

# Update and reboot

# On cephosd03, start ceph services
systemctl start  ceph.target
docker ps

# On deployment check the status
ceph status
ceph health detail
ceph osd tree
ceph pg dump_stuck

ceph osd unset noout
```



Operations

Add an OSD Node

```
# From deployment node as deploy user, prepare the new osd host
ssh  <OSD_NODE> 'sudo cp /home/deploy/.ssh/authorized_keys /root/.ssh/authorized_keys'
ssh-copy-id -f -i /etc/ceph/id_rsa root@<OSD_NODE>

# Add cephosd04 host to spec
vim $SPEC

# List the known hosts in cluster
sudo ceph orch host ls

# apply the spec
sudo ceph orch apply -i $CEPH_SPEC

# Watch
sudo ceph orch host ls
sudo ceph -s
```



Operations

Update system packages and reboot

```
# The same with ceph orch
ceph orch host maintenance enter <OSD_NODE>  --force --yes-i-really-mean-it

# Upgrade and reboot the host

ceph orch host maintenance exit <OSD_NODE>

ceph status
```



Operations

Upgrade

Cheat sheet to check



Operations

Upgrade

```
# Upgrade ceph cluster
VERSION="19.2.3"
IMAGE=registry.lab.nfgd.net/quay/ceph/ceph:v$VERSION

# check
ceph health

# order
# mgr -> mon -> crash -> osd -> mds -> rgw -> rbd-mirror -> cephfs-mirror -> ceph-exporter -> iscsi -> nfs -> nvmeof

# Review versions installed
ceph versions

# Review the releases page in browser
# https://docs.ceph.com/en/latest/releases/

# List available versions
ceph orch upgrade ls

# Start an upgrade for mgr, mon
ceph orch upgrade start --image $IMAGE --daemon-types mgr,mon

# status
ceph orch upgrade status
ceph -W cephadm
ceph versions

# stop / continue
# ceph orch upgrade stop
# ceph orch upgrade start --ceph-version $VERSION

# wait for 'in_progress' to become 'false'
ceph orch upgrade status

# continue to upgrade crash
ceph orch upgrade start --image $IMAGE --daemon-types crash

# wait for 'in_progress' to become 'false'
ceph orch upgrade status

# continue to upgrade osd
ceph orch upgrade start --image $IMAGE --daemon-types osd
# status
ceph -s
ceph health
ceph orch upgrade status

# continue with the rest
ceph orch upgrade start --image $IMAGE

# Check available upgrade version for ceph packages on deployment node
apt-cache policy ceph-common
apt install -y cephadm-<VERSION> ceph-common-<VERSION>
```



Operations

Replace a Mon Node

```
# 🚧 on deployment node remove cephmon03 from the cluster
for l in _admin mds mgr mon nfs rbd-mirror rgw
do
  ceph orch host label rm cephmon03 $l
done
ceph orch host drain cephmon03
# wait a bit
ceph -s
ceph orch host rm cephmon03

# 🚧 on cephmon03
systemctl stop ceph.target
rm -rf  /etc/ceph/ /var/lib/ceph/* /var/log/ceph/* /etc/systemd/system/ceph*
systemctl daemon-reload
docker image ls | awk '{print $3}' | xargs docker rmi

# 🩺 on deployment node check status
ceph -s
ceph orch host ls
ceph orch ls
ceph orch ps

# 🚚 Redeploy
ceph orch apply -i spec.yaml

# 🩺 check status on deployment node
ceph -s
ceph orch ps
```



Operations

Replace an OSD

```
# List all the OSDs in the cluster, take the last one, check the devices behind 📋
ceph osd tree
ceph osd metadata <OSD_NR> | grep device

# 💥 Let's break it, on <OSD_NODE>
dd if=/dev/zero of=<DEVICE> bs=1024 count=$((1024*10))

# 🩺 check status
ceph osd tree
ceph status
ceph health detail
ceph device ls

# 🛠️ Tell the cluster not to try to recover
ceph osd set noout
ceph osd set norecover

# 🚧 Rebuild
ceph orch ls osd --export
ceph orch set-unmanaged   osd.ssd
ceph orch ls osd
ceph osd tree

ceph orch osd rm --zap <OSD_NR> --force

# use with caution if you need to force
# ceph osd purge <OSD_NR> --yes-i-really-mean-it

ceph orch osd rm status
ceph status
ceph health detail

ceph orch daemon rm osd.<OSD.NR> --force

# on osdhost check if the device was zapped
lsblk
```



Operations

Replace an OSD

```
# 🚧 Set initial weight to 0 to control recovery
ceph config get osd osd_crush_initial_weight
ceph config set osd osd_crush_initial_weight 0

# 🚚 Redeploy
ceph orch set-managed   osd.ssd
ceph orch ls osd
ceph osd tree
ceph device ls

# 🩺 Check
ceph osd tree
ceph osd df
ceph pg ls-by-osd <OSD_NR>

# 🚚 Allow data distribution again
ceph osd crush reweight <OSD> <WEIGHT>

ceph config set osd osd_crush_initial_weight -1
ceph config get osd osd_crush_initial_weight
ceph osd unset noout
ceph osd unset norecover
```



Operations

Replace an OSD

Workarounds

```
# If --zap didn't work, the following tools can help to clean up

# List block devices
lsblk

# List device names
dmsetup ls

# Remove stale logical volume
dmsetup remove ceph--<ID>

# Clean up device
wipefs --all --force <DEVICE>
```



Operations

Repair a broken OSD Node

```
# 🚧 on the broken osd node clean up
systemctl stop ceph.target

systemctl list-units | grep ceph
rm -rf  /etc/ceph/ /var/lib/ceph/* /var/log/ceph/* /etc/systemd/system/ceph*
systemctl daemon-reload
docker image ls | awk '{print $3}' | xargs docker rmi
reboot
# After reboot continue
dmsetup ls
dmsetup ls | awk '{print $1}'| xargs -I{} sh -c 'echo Removing {}; dmsetup remove {};'

# 🛠️ Clean up device
pvs
pvs --noheadings | awk '{print $1}'| xargs -I{} sh -c 'echo Cleaning {}; wipefs --all --force {};'
pvs

# 🛠️ on the deployment node clean up
ceph orch host maintenance enter <OSD_NODE>  --force --yes-i-really-mean-it
ceph orch host ls
ceph osd tree
ceph orch host drain <OSD_NODE>
ceph orch host rm    <OSD_NODE> --force

# 🛠️ remove old OSDs, when still present
ceph orch host ls
ceph osd purge <OSD_NR> --yes-i-really-mean-it
ceph osd purge <OSD_NR> --yes-i-really-mean-it
ceph osd purge <OSD_NR> --yes-i-really-mean-it

ceph osd crush remove <OSD_NODE>

# 🩺 check before reguidling
ceph orch host ls
ceph osd tree

ceph orch apply -i /home/deploy/ceph/spec.yaml

# 🩺 check for problems on the osd node
tail -f /var/log/ceph/cephadm.log

# deployment deplyment node, check status
ceph -s
ceph orch host ls
ceph osd tree
```



Operations

Rebuild an OSD Node

```
# Prepare the rebuild
ceph osd set noout
ceph osd set norecover

# 🩺 check if OSDs can be stopped
ceph osd tree
ceph osd ok-to-stop <OSD_NR> <OSD_NR> <OSD_NR> | jq . |  head -n 25

# 🚧 Drain the Ceph OSD host
ceph orch host drain <hostname> [--zap-osd-devices]
ceph orch host drain status
ceph orch device ls  [hostname]
ceph orch osd  rm --zap <OSD_NR> --force
ceph orch host rm     <hostname> --force

# Reboot the host

# Join the cluster again
ceph orch set-managed osd.ssd
ceph orch apply -i spec.yaml

# Rebuild the host
# ⚠️ Add the ssh key for the ceph deployment in authorized_keys on the host

# 🩺 Check the status
ceph status
ceph health detail

# ⚠️ You may want to control the recovery process – check available options

ceph osd unset noout
ceph osd unset norecover
```



Operations

Remove an OSD Node

```
# Start maintenance for the host
ceph orch host maintenance enter cephosd04  --force --yes-i-really-mean-it

# Drain/Remove
ceph orch host drain  cephosd04
ceph orch host rm     cephosd04 [--force]

# Help if needed
# ceph orch osd  rm --zap <OSD_NR> --force
# use with caution if you need to force
# ceph osd purge <OSD_NR> --yes-i-really-mean-it

ceph status

# Remove the host from the crush map
ceph osd tree
ceph osd crush remove cephosd04
```



Modify and activate a spec

```
# modify the spec
vim $SPEC

# apply
sudo ceph orch apply -i $CEPH_SPEC
```



Operations

Modify crush map

Failure Domain

```
# Bucket types
ceph osd crush dump  | jq .types

# Create racks
ceph osd crush add-bucket rack-01 rack
ceph osd crush add-bucket rack-02 rack
ceph osd crush add-bucket rack-03 rack

# Move the racks under the root
ceph osd crush move rack-01 root=default
ceph osd crush move rack-02 root=default
ceph osd crush move rack-03 root=default

# Move the hosts into the racks
ceph osd crush move cephosd01 rack=rack-01
ceph osd crush move cephosd02 rack=rack-02
ceph osd crush move cephosd03 rack=rack-03

# Create a new replicated rule and assign it to a volume
ceph osd crush rule create-replicated replicated_rack_rule default rack
ceph osd crush rule ls
ceph osd crush rule dump replicated_rack_rule
ceph osd pool set <POOL> crush_rule replicated_rack_rule
```



Benchmarking

```
# Benchmark a specific OSD
ceph tell osd.2 bench
ceph --format plain tell osd.0 bench
# Benchmark all OSDs
ceph tell osd.* bench
ceph --format plain tell osd.* bench

# Benchmark on newly created pool + cleanup
ceph osd pool create benchmark 64
rados bench -p benchmark 30 write
rados -p benchmark bench -b 4096 30 write # object size 4096
# Read benchmark
rados -p benchmark bench 30 write --no-cleanup
rados -p benchmark bench 30 rand
ceph tell mon.\* injectargs '--mon-allow-pool-delete=true'
ceph osd pool delete benchmark benchmark --yes-i-really-really-mean-it

# Benchmark on a newly created volume + cleanup
rbd create  --size 1024 benchmark/test
rbd bench-write benchmark/test --io-size 1048576  --io-threads 16 --io-pattern rand
rbd rm  benchmark/test

# Benchmark on a newly created volume + cleanup
rbd create  --size 32 benchmark/test
fio --size=100M --ioengine=rbd --invalidate=0 --direct=1 --numjobs=10 --rw=write --name=fiojob --blocksize_range=4K-512k --iodepth=1 --pool=benchmark --rbdname=test
rbd rm  benchmark/test
```



Performance tests

Simple - Standard tests with dd

```
# Big file
dd if=/dev/random of=./testfile bs=1G count=1 oflag=direct
# Many small files
dd if=/dev/zero of=./testfile bs=512 count=1000 oflag=dsync
```



Performance tests

Sequential Throughput (1MB)

Simulates: Large file transfers, VM migrations, backups.

```
# dd if=/dev/zero of=testfile bs=1M count=2048 oflag=direct
fio                 \
--name=baseline_seq  \
--ioengine=libaio    \
--direct=1           \
--bs=1M              \
--size=2G            \
--rw=write           \
--iodepth=8          \
--numjobs=1          \
--group_reporting    \
--runtime=60         \
--time_based
```



Performance tests

Random IOPS (4KB)

Simulates: General OS responsiveness and small metadata operations.

```
# dd if=testfile of=/dev/null bs=4k count=262144 iflag=direct
fio                 \
--name=baseline_iops \
--ioengine=libaio    \
--direct=1           \
--bs=4k              \
--size=1G            \
--rw=randrw          \
--rwmixread=100      \
--iodepth=32         \
--numjobs=1          \
--group_reporting    \
--runtime=60         \
--time_based
```



Performance tests

PostgreSQL (8KB Blocks)

```
#
fio               \
--name=db_postgres \
--ioengine=libaio  \
--direct=1         \
--bs=8k            \
--size=2G          \
--rw=randrw        \
--rwmixread=70     \
--iodepth=16       \
--numjobs=2        \
--group_reporting  \
--runtime=120
```



Performance tests

MySQL/MariaDB (16KB Blocks)

```
#
fio              \
--name=db_mysql   \
--ioengine=libaio \
--direct=1        \
--bs=16k          \
--size=2G         \
--rw=randrw       \
--rwmixread=70    \
--iodepth=16      \
--numjobs=2       \
--group_reporting \
--runtime=120
```



Links

- Ceph - Docs
- Ceph - Releases
- Ceph - Dashboard
- Ceph - Grafana / Monitoring
- Ceph - Prometheus / Monitoring
- Ceph - Alertmanager

- Cephadm - Guide

- Harbor - Registry

Feedback and contributions

Found an error or want to contribute? Open a PR on GitHub.

More labs and cheat sheets: infraguide.org

Questions or ideas: ping@infraguide.org