HYPER SCALERS

2022

Ceph Pacific Appliance Deployment

HYPERSCALERS

Contents

Ceph Storage deployment to Bare metal Machines

About Ceph - Ceph is a software-defined storage solution designed to address the object, block, and file storage needs of data centres adopting open source as the new norm for high-growth block storage, object stores and data lakes. Ceph provides enterprise scalable storage while keeping CAPEX and OPEX costs in line with underlying bulk commodity disk – SSD and HDD prices.

Ceph (16.2.7/ Pacific) [1] is a freely available storage platform that implements object storage on a single distributed computer cluster and provides interfaces for object-, block- and file-level storage. Ceph aims primarily for completely distributed operation without a single point of failure. Ceph storage manages data replication and is generally quite fault tolerant. As a result of its design, the system is both self-healing and self-managing.

Hyperscalers provides support for Ceph Pacific.

Terminologies of a Ceph cluster

Before we dive into the actual deployment process, let's see what we'll need to understand a few terminologies in our Ceph cluster.

There are three services that form the backbone of the cluster [2]

- **ceph monitors** (ceph-mon) maintain maps of the cluster state and are also responsible for managing authentication between daemons and clients
- **managers** (ceph-mgr) are responsible for keeping track of runtime metrics and the current state of the Ceph cluster
- **object storage daemons** (ceph-osd) store data, handle data replication, recovery, rebalancing, and provide some ceph monitoring information.

Additionally, we can add further parts to the cluster to support different storage solutions

- **metadata servers** (ceph-mds) store metadata on behalf of the Ceph Filesystem
- **rados gateway** (ceph-rgw) is a Hypertext Transfer Protocol server for interacting with a Ceph Storage Cluster that provides interfaces compatible with OpenStack Swift and Amazon S3.

There are multiple ways of deploying these services. We'll check two of them:

- first, using the `ceph/deploy` tool,
- then docker

Ceph Setup

For the proof-of-concept infrastructure, we'll set up the bare minimum managers, monitors. object storage drives needed to achieve full functionality of the ceph storage cluster

You should not run multiple different Ceph daemons on the same host, but for the sake of simplicity, we'll only use 4 machines for the whole cluster.

In the case of object storage drives, you can run multiple of them on the same host but using the same storage drive for multiple instances is a bad idea as the disk's Input/Output speed might limit the object storage drive daemons' performance.

We've created four (three (initial deployment) plus one (expansion of cluster)) for Ceph and one admin node. For ceph-deploy to work, the admin node requires passwordless secure shell access to the nodes and that secure shell user has to have passwordless sudo privileges.

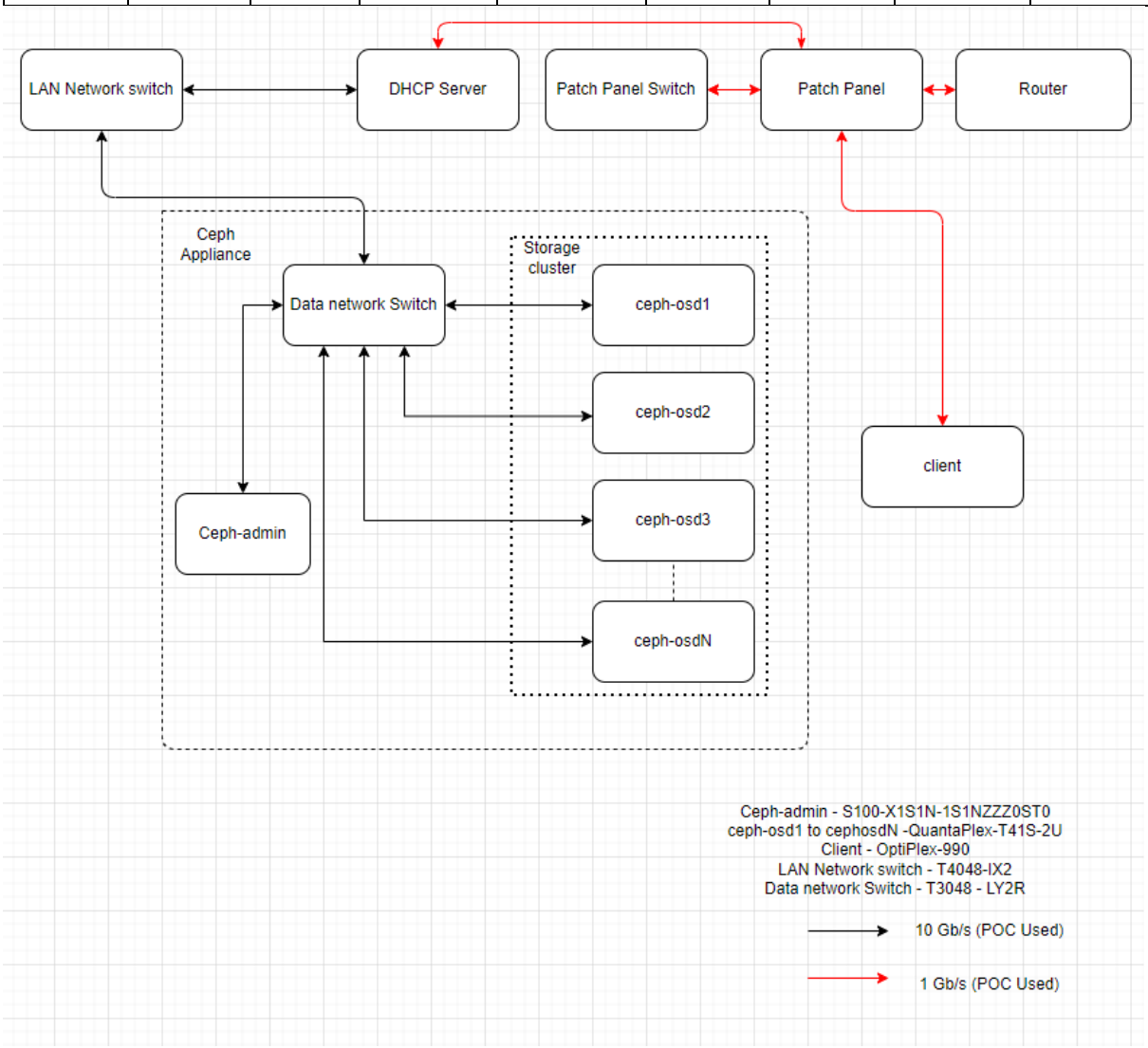
Before you deploy Ceph, firewall settings or other resources have to be adjusted to open these ports

- 22 for secure shell
- 6789 for monitors
- 6800:7300 for object storage drives, managers, and metadata servers
- 8080 for dashboard
- 7480/80 for rados object gateway

Tested Hardware environment and deployed architecture

Server	Number of nodes	Nodes	CPU	RAM	NIC Mezz	Storage card	PCIe NIC	Storage Drives	OS
S5S [3]/S100	4 + 1	Mgmt (S100-X1S1N)	E3-1220 v3 x1	8/1600x2	Null	Null	Intel 82599ES 10GBe	240 GB Samsung NVMe	Ubuntu 18.04
		Node 1	Intel Scalable Processor Family	16/2933x4 units	Intel 82599ES x8	3108 (IT/JBOD mode)	Solarflare 9120 10GBe	120 GB Intel NVMe, 1.96 TB HGST x2	Ubuntu 18.04
		Node 2	Intel Scalable Processor Family	16/2933x4 units	Intel 82599ES x8	3108 (IT/JBOD mode)	Solarflare 9120 10GBe	120 GB Intel NVMe, 1.96 TB HGST x2	Ubuntu 18.04
		Node 3	Intel Scalable Processor Family	16/2933x4 units	Intel 82599ES x8	3108 (IT/JBOD mode)	Solarflare 9120 10GBe	120 GB Intel NVMe, 1.96 TB HGST x2	Ubuntu 18.04

		Node 4	Intel Scalable Processor Family	16/2933x4 units	Intel 82599ES x8	3108 (IT/JBOD mode)	Intel 82599ES 10GBe	240 GB Samsung NVMe, 1.96 TB HGST x2	Ubuntu 18.04



Ceph pacific is deployable across many types of servers and storage servers including S5S, S5P, S5X, S5K and many others across 25/40/100 Gb/s data network speeds [4].

S5X | D53X-1U - 3rd Gen Intel® Xeon Scalable processor Ice Lake empowered (hyperscalers.com)

S5P-T22P-4U (hyperscalers.com)

S5K | D43K-1U AMD EPYC 3rd Gen Milan (hyperscalers.com)

S5S+ TC | T42SP-2U (hyperscalers.com)

S5S TC | T42S-2U (hyperscalers.com)

Ceph Storage Deployment

Install prerequisites on all machines

The following are the requirements within the operating system (Ubuntu 18.04) needed for deployment of Ceph storage cluster [5]

- Python 3
- Systemd
- Podman or Docker for running containers [6]
- Time synchronization (such as chrony or network time protocol)
- Logical Volume Manager 2 for provisioning storage drives

```
$ sudo apt update
$ sudo apt -y install ntp python openssh-server
```

For Ceph to work seamlessly, we have to make sure the system clocks are synchronized. The suggested solution is to install network time protocol on all machines, and it will take care of the problem. While we're at it, let's install python, secure shell on all hosts as ceph-deploy depends on it being available on the target machines. It is preferable to use the clients and nodes as a root user. In the file at `/etc/ssh/sshd_config`, change permit root login [7] [8]

```
PermitRootLogin yes
```

Prepare the admin node

```
$ ssh -i ~/.ssh/id_rsa -A root@node-ip
```

As all the machines have public key added to `known_hosts`, we can use secure shell agent forwarding to access the Ceph machines from the admin node. The first line ensures that local secure shell agent has the proper key in use and the -A flag takes care of forwarding the key.

```
$ wget -q -O- 'https://download.ceph.com/keys/release.asc' | sudo apt-key
add -
echo deb https://download.ceph.com/debian-pacific/ $(lsb_release -sc) main
| sudo tee /etc/apt/sources.list.d/ceph.list
$ sudo apt update
$ sudo apt -y install ceph-deploy
```

We'll use the latest Pacific (16.2.7) release in this example. If you want to deploy a different version, just change the `debian-pacific` part to your desired release (luminous (12.2.13), mimic (13.2.10), etc.).

```
$ echo "StrictHostKeyChecking no" | sudo tee -a /etc/ssh/ssh_config >
/dev/null
```

OR

```
$ ssh-keyscan -H monitor-one-ip, monitor-two-ip, monitor-three-ip >>
~/.ssh/known_hosts
```

`ceph-deploy` uses secure shell connections to manage the nodes we provide. Each time you secure shell login to a machine that is not in the list of known_hosts (`~/.ssh/known_hosts`), you'll get prompted whether you want to continue connecting or not. This interruption does not mesh well with the deployment process, so we either have to use `ssh-keyscan` to grab the fingerprint of all the target machines or disable the strict host key checking outright.


```
monitor-one-ip cephosd1-QuantaPlex-T41S-2U

monitor-two-ip  cephosd2-QuantaPlex-T41S-2U

monitor-three-ip cephosdd3-QuantaPlex-T41S-2U

monitor-four-ip cephosd4-QuantaPlex-T41S-2U

client-one-ip administrator-OptiPlex-990 (client)
```

Even though the target machines are in the same subnet as our admin and they can access each other, we have to add them to the hosts file (/etc/hosts) for `ceph-deploy` to work properly. `ceph-deploy` creates monitors by the provided hostname, so make sure it matches the actual hostname of the machines otherwise the monitors won't be able to join the quorum and the deployment fails. Don't forget to reboot the admin node for the changes to take effect.

```
$ mkdir ceph-deploy
$ cd ceph-deploy
```

As a final step of the preparation, let's create a dedicated folder as ceph-deploy will create multiple config and key files during the process.

Deploy resources

```
$ ceph-deploy new cephosd1-QuantaPlex-T41S-2U cephosd2-QuantaPlex-T41S-2U
cephosdd3-QuantaPlex-T41S-2U
```

The command `ceph-deploy newly` creates the necessary files for the deployment, passes it to the hostnames of the **monitor** nodes, and it will create `ceph.conf` and `ceph.mon.keyring` along with a log file [9].

The `ceph.conf` should look something like this

```
[global]

fsid = a72e77b0-7265-4e9f-ba17-5dffe87e4e8a

mon_initial_members = cephosd1-QuantaPlex-T41S-2U, cephosd2-QuantaPlex-
T41S-2U, cephosdd3-QuantaPlex-T41S-2U

mon_host = monitor-one-ip, monitor-two-ip, monitor-three-ip

auth_cluster_required = cephx

auth_service_required = cephx
```

```
auth_client_required = cephx
```

It has a unique ID called `fsid`, the monitor hostnames and addresses and the authentication modes. Ceph provides two authentication modes: none (anyone can access data without authentication) or cephx (key based authentication).

The other file, the monitor keyring is another important piece of the puzzle, as all monitors must have identical keyrings in a cluster with multiple monitors. `ceph-deploy` takes care of the propagation of the key file during virtual deployments.

```
$ ceph-deploy install --release pacific cephosd1-QuantaPlex-T41S-2U
cephosd2-QuantaPlex-T41S-2U cephosdd3-QuantaPlex-T41S-2U
```

As you might have noticed so far, we haven't installed ceph on the target nodes yet. We could do that one-by-one, but a more convenient way is to let `ceph-deploy` take care of the task. Don't forget to specify the release of your choice, otherwise you might run into a mismatch between your admin and targets.

```
$ ceph-deploy mon create-initial
```

Finally, the first piece of the cluster is up and running, `create-initial` will deploy the monitors specified in `ceph.conf` we generated previously and also gather various key files. The command will only complete successfully if all the monitors are up and, in the quorum, [9].

```
$ ceph-deploy admin cephosd1-QuantaPlex-T41S-2U cephosd2-QuantaPlex-T41S-2U
cephosdd3-QuantaPlex-T41S-2U
```

Executing `ceph-deploy admin` will push a Ceph configuration file and the `ceph.client.admin.keyring` to the `/etc/ceph` directory of the nodes, so we can use the ceph command line interface without having to provide the `ceph.client.admin.keyring` each time to execute a command.

At this point, we can check the status of our cluster. Let's secure shell login into a target machine (we can do it directly from the admin node thanks to agent forwarding) and run `sudo ceph status`.

```
$ sudo ceph status
  cluster:
        id:     a72e77b0-7265-4e9f-ba17-5dffe87e4e8a
        health: HEALTH_OK

  services:
        mon: 3 daemons, cephosd1-QuantaPlex-T41S-2U, cephosd2-QuantaPlex-
T41S-2U, cephosdd3-QuantaPlex-T41S-2U (age 110m)
mgr: no daemons active
osd: 0 osds: 0 up, 0 in

  data:
        pools:   0 pools, 0 pgs
objects: 0 objects, 0 B
```

```
        usage:   0 B used, 0 B / 0 B avail
pgs:
```

Here we get a quick overview of what we have so far. Our cluster seems to be healthy, and all three monitors are listed under services. Let's go back to the admin and continue adding pieces.


```
$ ceph-deploy mgr create cephosd1-QuantaPlex-T41S-2U cephosdd3-QuantaPlex-T41S-2U
```

For Luminous (12.2.13) and above builds a manager daemon is required. It's responsible for monitoring the state of the Cluster [9] and manages modules/plugins.

We have all the management in place, let's add some storage to the cluster.

First, we have to find out (on each target machine) the label of the drive we want to use. To fetch the list of available disks on a specific node [9], run

```
$ ceph-deploy disk list <host-name>

$ ceph-deploy osd create --data /dev/sdb cephosd1-QuantaPlex-T41S-2U
$ ceph-deploy osd create --data /dev/sdb cephosd2-QuantaPlex-T41S-2U
$ ceph-deploy osd create --data /dev/sdb cephosdd3-QuantaPlex-T41S-2U
```

In this case the label was `sdb` on all three machines, so to add object storage drives to the cluster we just ran these three commands.

At this point, our cluster is basically ready. We can run `ceph status` to see that our monitors, managers and OSDs are up and running. But nobody wants to secure shell login into a machine every time to check the status of the cluster. Luckily there's a pretty neat dashboard that comes with Ceph, we just have to enable it.

Ceph Dashboard

The dashboard was introduced in Luminous (12.2.13) release and was further improved in Mimic (13.2.10). However, currently we're deploying Pacific (16.2.7), the latest version of Ceph. After trying the usual way of enabling the dashboard via a manager

We have to install it first.

```
$ sudo apt install -y ceph-mgr-dashboard

$ sudo apt install -y python-routes

$ sudo ceph mgr module enable dashboard
```

In Pacific (16.2.7), the dashboard package is no longer installed by default. We can check the available modules by running

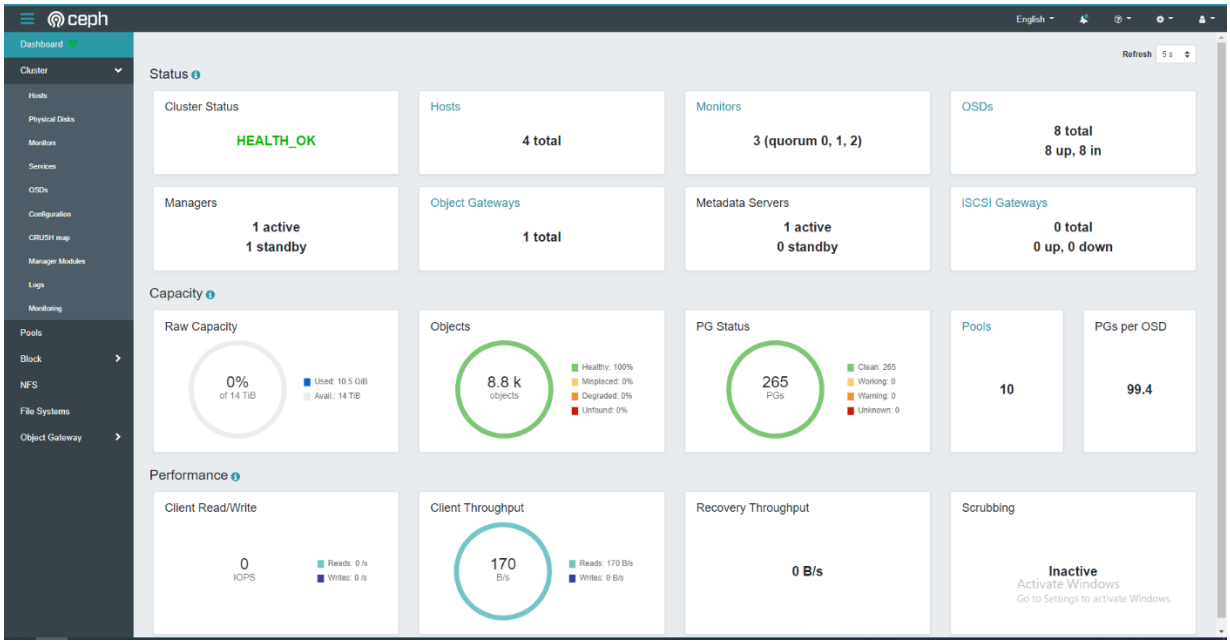
```
$ sudo ceph mgr module ls
```

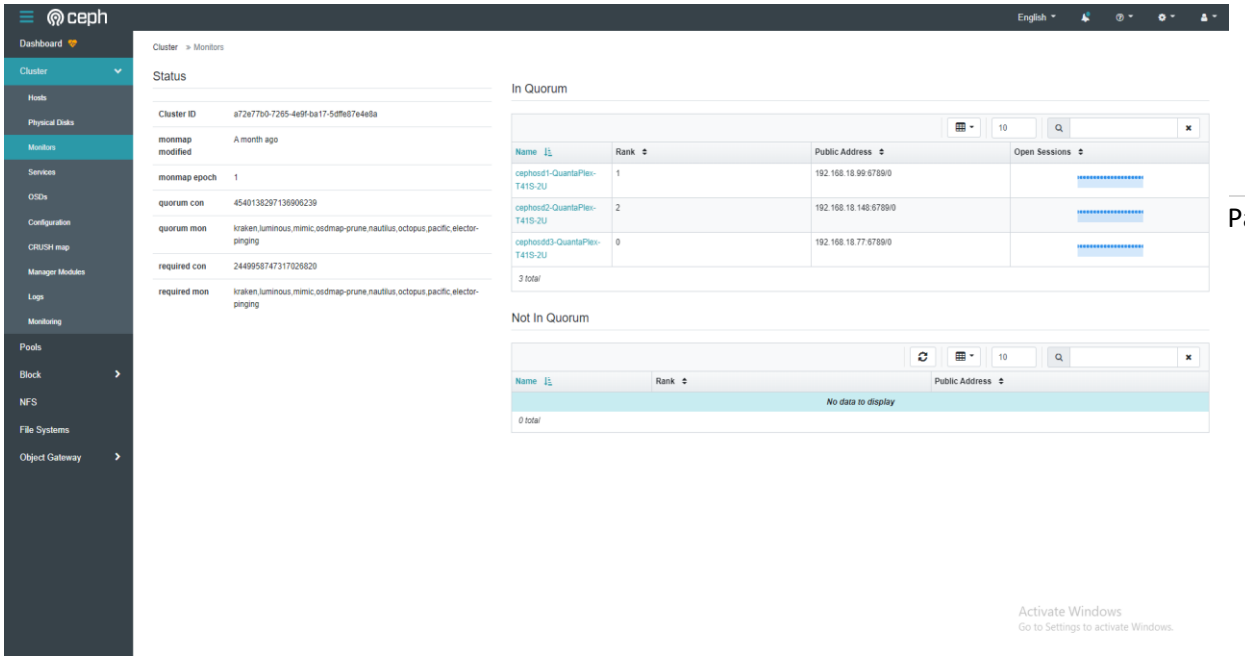
and as expected, dashboard is not there, it comes in a form a separate package. So

We're all set to enable the dashboard module now. As it's a public-facing page that requires login, we should set up a cert for SSL. [10]

```
sudo ceph mgr module enable dashboard

ceph dashboard create-self-signed-cert

ceph dashboard set-ssl-certificate -i dashboard.crt

ceph dashboard set-ssl-certificate-key -i dashboard.key

sudo ceph dashboard ac-user-create admin secret administrator
```


By default, the dashboard is available on the host running the manager on port 8080. After logging in, we get an overview of the cluster status, and under the cluster menu, we get really detailed overviews of each running daemon.





If we try to navigate to the `Filesystems` or `Object Gateway` tabs, we get a notification that we haven't configured the required resources to access these features. Our cluster can only be used as a block storage right now.

Using the Ceph filesystem

Going back to our admin node, running

```
$ ceph-deploy mds create cephosd1-QuantaPlex-T41S-2U cephosd2-QuantaPlex-T41S-2U cephosdd3-QuantaPlex-T41S-2U
```

will create metadata servers, that will be inactive for now, as we haven't enabled the feature yet. First, we need to create two RADOS pools, one for the actual data and one for the metadata. [11]

```
$ sudo ceph osd pool create cephfs_data 8
$ sudo ceph osd pool create cephfs_metadata 8
```

After creating the required pools, we're ready to enable the filesystem feature

```
$ sudo ceph fs new cephfs cephfs_metadata cephfs_data
```

The MDS daemons will now be able to enter an active state, and we are ready to mount the filesystem. We have two options to do that, via the kernel driver or as FUSE with `ceph-fuse` [11].

Before we continue with the mounting, let's create a user keyring that we can use in both solutions for authorization and authentication as we have cephx enabled. There are multiple restrictions that can be set up when creating a new key specified in the docs [12]. For example:

```
$ sudo ceph auth get-or-create client.user mon 'allow r' mds 'allow r, allow
rw    path=/home/cephfs'    osd    'allow    rw    pool=cephfs_data'    -o
/etc/ceph/ceph.client.user.keyring
```

will create a new client key with the name `user` and output it into `ceph.client.user.keyring`. It will provide write access for the MDS only to the `/home/cephfs` directory, and the client will only have write access within the `cephfs_data` pool.

Mounting with the kernel

Now let's create a dedicated directory and then use the key from the previously generated keyring to mount the filesystem with the kernel. [11]

```
$ sudo mkdir /mnt/mycephfs
$    sudo    mount    -t    ceph    192.168.18.99:6789:/    /mnt/mycephfs    -o
name=user,secret=AQBxnDFdS5atIxAAV0rL9klnSxwy6EFpR/EFbg==
```

Using the RADOS gateway

To enable the S3 management feature of the cluster, the rados gateway has to be created. [13]

```
$ ceph-deploy rgw create cephosd1-QuantaPlex-T41S-2U
```

For the dashboard, it's required to create a `radosgw-admin` user with the `system` flag to enable the Object Storage management interface. We also have to provide the user's `access_key` and `secret_key` to the dashboard before we can start using it.

```
$ sudo radosgw-admin user create --uid=rg_wadmin --display-name=rgw_admin -
-system
$ sudo ceph dashboard set-rgw-api-access-key <access_key>
$ sudo ceph dashboard set-rgw-api-secret-key <secret_key>
```

Using the Ceph Object Storage is really easy as RGW provides an interface identical to S3. You can use your existing S3 requests and code without any modifications, just have to change the connection string, access, and secret keys.

Ceph Storage Monitoring

The dashboard we've deployed shows a lot of useful information about our cluster, but monitoring is not its strongest suit. Ceph comes with a Prometheus module. After enabling it by running:

```
$ sudo ceph mgr module enable prometheus
```

A wide variety of metrics will be available on the given host on port 9283 by default [14]. To make use of these exposed data, we'll have to set up a prometheus instance.

There are multiple ways of firing up Prometheus, probably the most convenient is with docker. After installing docker [6] on your machine, create a `prometheus.yml` file to provide the endpoint where it can access our Ceph metrics [14] [10].

```
# /etc/prometheus.yml

scrape_configs:
  - job_name: 'ceph'
    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.
    static_configs:
    - targets: ['<target-ip>:9283']
```


Then launch the container itself by running:

```
$ sudo docker run -p 9090:9090 -v
/etc/prometheus.yml:/etc/prometheus/prometheus.yml prom/prometheus
```

Prometheus will start scraping our data, and it will show up on its dashboard. We can access it on port `9090` on its host machine. Prometheus dashboard is great but does not provide a very eye-pleasing dashboard. That's the main reason why it's usually used in pair with Graphana, which provides awesome visualizations for the data provided by Prometheus [10]. It can be launched with docker as well.

```
$ sudo docker run -d -p 3000:3000 grafana/grafana
```

Grafana is fantastic when it comes to visualizations but setting up dashboards can be a daunting task. To make our lives easier, we can load one of the pre-prepared dashboards [15]

Migrating to cephadm

In the earlier deployment no orchestrator is being installed making the expansion of the cluster a tedious process. The following steps can be implemented to migrate the existing and

expansion cluster to cephadm based orchestration [16] by adding another object storage drive node (`cephosd4-QuantaPlex-T41S-2U`) [17].

1. Install cephadm in every node of the existing cluster.

   1.1. `curl --silent --remote-name --location https://github.com/ceph/ceph/raw/pacific/src/cephadm/cephadm`

   1.2. `chmod +x cephadm`

   1.3. `./cephadm add-repo --release <release-name>`

   1.4. `./cephadm install`

2. `cephadm prepare-host`

   The above command ensures that all the prerequisites of the cluster are being met.

3. `cephadm ls`

   The above command lists the daemons that are running in the cluster for which the migration procedures need to be performed.

4. `ceph config assimilate-conf -i /etc/ceph/ceph.conf`

5. `ceph config dump`

   The above command lists the monitors, managers that are actively running in the cluster which helps us to migrate them to cephadm.

6. `cephadm adopt --style legacy --name mon.<hostname>`

   Adapt every monitor into cephadm

7. `cephadm adopt --style legacy --name mgr.<hostname>`

   Adopt every manager into cephadm

8. `ceph mgr module enable cephadm`

9. `ceph orch set backend cephadm`

   Enabling the cephadm orchestrator.

10. `ceph cephadm generate-key`

11. `ceph cephadm get-pub-key > ~/ceph.pub`

12. `ssh-copy-id -f -i ~/ceph.pub root@<host>`

    Generate private and public keys and copy it to every ceph node in the cluster.

13. `ceph orch host add <hostname> [ip-address]`

    Adds a new host (`cephosd4-QuantaPlex-T41S-2U`)

14. `ceph orch ps`

    The above command helps to verify the proper functionality of monitor and manager daemons

15. cephadm adopt --style legacy --name <name> (example name: osd.1) # Migrates object storage drives (performed in nodes where the drives are present)
16. ceph fs ls #lists the filesystems in the cluster
17. ceph orch apply mds <fs-name> [--placement=<placement>] #migrates the existing filesystem (FS)
18. ceph orch ps --daemon-type mds # verification of migration of FS
19. systemctl stop ceph-mds.target #stopping earlier daemon
20. rm -rf /var/lib/ceph/mds/ceph-* # removes earlier files related to FS
21. ceph orch apply rgw <realm> <zone> [--subcluster=<subcluster>] [--port=<port>] [--ssl] [--placement=<placement>] # migrates rados gateway
22. systemctl stop ceph-rgw.target # stopping earlier daemon
23. rm -rf /var/lib/ceph/radosgw/ceph-* #removes earlier files related to rados
24. ceph health detail # check for any stray daemons

Removing CephFS and CephFS Pools

The following commands are executed in manager node.

1. ceph tell mon.* injectargs '--mon-allow-pool-delete=true'
2. ceph fs fail cephfs
3. ceph fs rm cephfs --yes-i-really-mean-it
4. ceph osd pool delete cephfs_data cephfs_data --yes-i-really-really-mean-it
5. ceph osd pool delete cephfs_metadata cephfs_metadata --yes-i-really-really-mean-it

Resetting Rados Gateway upon restart of Rados Gateway deployed node

The following commands are executed in manager node.

1. ceph osd pool delete default.rgw.log default.rgw.log --yes-i-really-really-mean-it
2. ceph osd pool delete default.rgw.log default.rgw.log --yes-i-really-really-mean-it
3. ceph osd pool delete default.rgw.control default.rgw.control --yes-i-really-really-mean-it
4. ceph osd pool delete default.rgw.meta default.rgw.meta --yes-i-really-really-mean-it

5. `ceph osd pool delete .rgw.root .rgw.root --yes-i-really-really-mean-it`

After these actions, `.rgw.root` will reinitialize into the UI and all other corresponding pools will come up subsequently. Verify that rgw-hosted-node-ip:7480 is reachable with output something like [13]


```
<ListAllMyBucketsResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
<Owner>
<ID>anonymous</ID>
<DisplayName/>
</Owner>
<Buckets/>
</ListAllMyBucketsResult>
```

Recreate the admin user and add the access and secret keys to the ceph dashboard

Safely removing Object storage drive and remounting

The following are done in the node where the Object storage drive is present

1. `systemctl stop ceph-osd@x`
2. `ceph osd out osd.x`
3. `ceph osd down osd.x`
4. `ceph osd rm osd.x`
5. `ceph osd crush rm osd.x`
6. `ceph auth del osd.x`
7. `while ! ceph osd safe-to-destroy osd.x ; do sleep 10 ; done`
8. `ceph osd destroy x --yes-i-really-mean-it`

Upon these commands the cluster will rebalance itself to number-of-drives minus one in the object storage drives cluster and become healthy.

- `ceph-volume lvm zap /dev/sdX`

If the above command fails with `wipefs probing initialization failed`, execute

- `wipefs -af /dev/sdX`

Remove the OSD from the node and replace it, note down the updated /dev/sdX mount location, in admin node execute,

- `ceph-deploy osd create --data /dev/sdX node-name`

Creating a Rados Block Device (RBD)

In admin node, [18]

1. `rbd pool init <pool-name>`

In client node,

2. `rbd create <pool-name> --size <pool-size> --image-feature layering -m mon-ip -k /path/to/ceph.client.admin.keyring`
3. `rbd map <pool-name> --name client.admin -m monitor-ip -k /path/to/ceph.client.admin.keyring`
4. `mkfs.ext4 -m0 /dev/rbdX`

Ceph Storage cluster benchmark

Create a benchmark pool [19]

`ceph osd pool create scbench <placement-group-num> <placement-group-for-placement-purpose-number>` [20]

`rados bench -p <pool-name> <time-in-seconds> <write|seq|rand> -t <number-of-threads|default=16>`

Scenario

1. Write benchmark – bandwidth - 899.196 MB/s; Average IOPS – 224
2. Sequential read benchmark – bandwidth – 1424.54 MB/s; Average IOPS – 356
3. Random read benchmark – bandwidth – 1452.63 MB/s; Average IOPS – 363

Write benchmark

```
root@cephosd1-QuantaPlex-T41S-2U:/home/cephosd1# rados bench -p scbench 10 write --no-cleanup
hints = 1
Maintaining 16 concurrent writes of 4194304 bytes to objects of size 4194304 for up to 10 seconds or 0 o
bjects
Object prefix: benchmark_data_cephosd1-QuantaPlex-T41S-2U_960027
  sec Cur ops   started  finished  avg MB/s  cur MB/s last lat(s)  avg lat(s)
    0       0         0         0         0         0           -           0
    1      16       233       217   867.861       868     0.12278   0.0699251
    2      16       451       435   869.871       872   0.0464751   0.0715526
    3      16       676       660   879.874       900   0.0411569   0.0716654
    4      16       903       887   886.872       908   0.0542586   0.0713965
    5      16      1124      1108   886.273       884   0.0626679   0.0714274
    6      16      1344      1328   885.203       880   0.0490622   0.0719086
    7      16      1564      1548    884.44       880    0.108266   0.0718224
    8      16      1798      1782   890.865       936    0.051938   0.0715028
    9      16      2028      2012   894.084       920    0.127798   0.0712524
   10      15      2260      2245   897.863       932   0.0793156   0.0710174
Total time run:         10.0534
Total writes made:      2260
Write size:             4194304
Object size:            4194304
Bandwidth (MB/sec):     899.196
Stddev Bandwidth:       24.9622
Max bandwidth (MB/sec): 936
Min bandwidth (MB/sec): 868
Average IOPS:           224
Stddev IOPS:            6.24055
Max IOPS:               234
Min IOPS:               217
Average Latency(s):     0.0709566
Stddev Latency(s):      0.0325125
Max latency(s):         0.240095
Min latency(s):         0.0277656
```

Sequential read Benchmark

```
root@cephosd1-QuantaPlex-T41S-2U:/home/cephosd1# rados bench -p scbench 10 seq
hints = 1
  sec Cur ops   started  finished  avg MB/s  cur MB/s last lat(s)  avg lat(s)
    0       0         0         0         0         0           -           0
    1      16       371       355   1419.57      1420   0.0876289   0.0428356
    2      16       724       708   1415.66      1412   0.0323933   0.0437681
    3      16      1087      1071   1427.69      1452   0.0288853   0.0435786
    4      15      1457      1442   1441.72      1484   0.0318756   0.0433583
    5      16      1813      1797   1437.34      1420   0.0611507   0.0436626
    6      16      2161      2145   1429.76      1392   0.0380427   0.0439118
Total time run:       6.34593
Total reads made:     2260
Read size:            4194304
Object size:          4194304
Bandwidth (MB/sec):   1424.54
Average IOPS:         356
Stddev IOPS:          8.19146
Max IOPS:             371
Min IOPS:             348
Average Latency(s):   0.0440899
Max latency(s):       0.189666
Min latency(s):       0.00910259
```


Random read rados Benchmark

```
root@cephosd1-QuantaPlex-T41S-2U:/home/cephosd1# rados bench -p scbench 10 rand
hints = 1
  sec Cur ops   started  finished  avg MB/s  cur MB/s last lat(s)  avg lat(s)
    0       0         0         0         0         0           -           0
    1      16       380       364   1455.53      1456   0.0261242   0.0414834
    2      15       733       718   1435.55      1416   0.0750217   0.0432696
    3      16      1095      1079    1438.3      1444   0.0341771   0.0433615
    4      16      1480      1464   1463.66      1540   0.0651599   0.0427036
    5      16      1848      1832   1465.29      1472   0.0839621   0.0427691
    6      16      2193      2177   1451.04      1380   0.0992841    0.043223
    7      16      2551      2535   1448.29      1432    0.055503   0.0433095
    8      16      2925      2909   1454.22      1496   0.0316212   0.0431631
    9      16      3290      3274   1454.84      1460   0.0332063   0.0431945
   10      14      3649      3635   1453.56      1444   0.0095314   0.0431925
Total time run:       10.048
Total reads made:     3649
Read size:            4194304
Object size:          4194304
Bandwidth (MB/sec):   1452.63
Average IOPS:         363
Stddev IOPS:          10.9062
Max IOPS:             385
Min IOPS:             345
Average Latency(s):   0.0432777
Max latency(s):       0.214364
Min latency(s):       0.00371021
```

Speed test with Rados Block Device

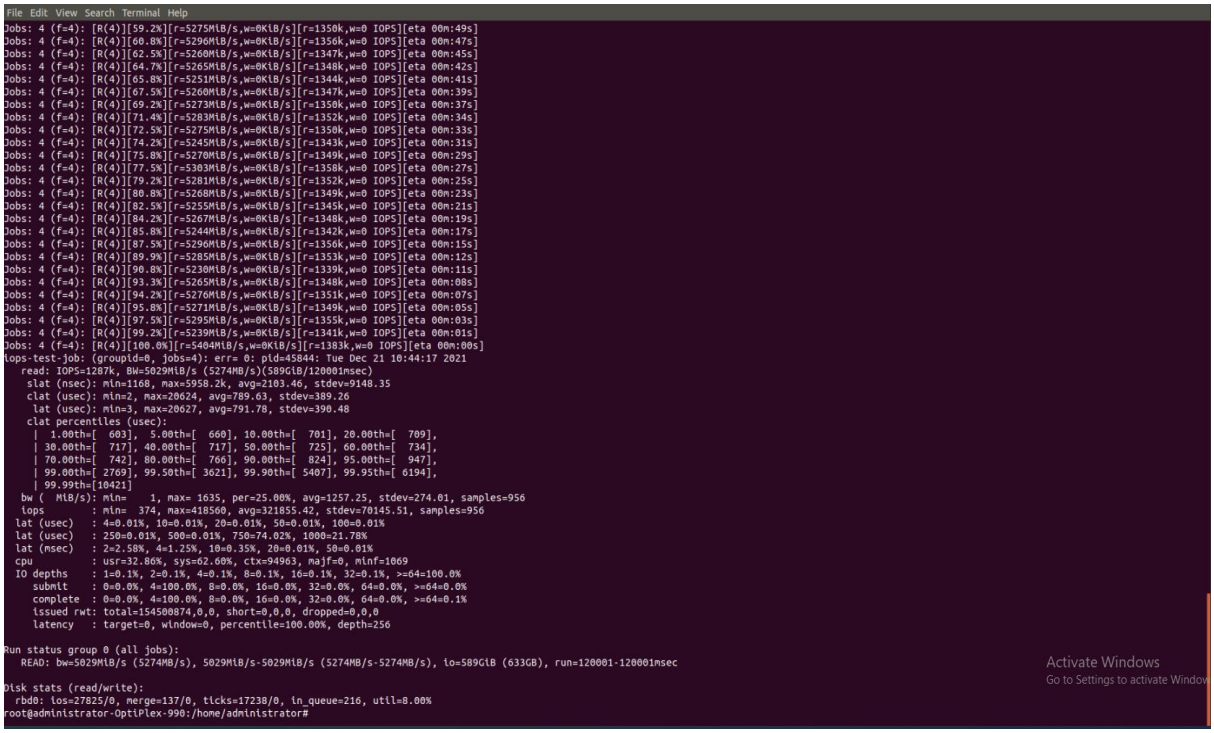
The speed test on rados block device was performed using FIO tool [21].

```
fio --filename=/dev/rbdX --rw=read --bs=4k --ioengine=libaio --iodepth=256
--runtime=120 --numjobs=4 --time_based --group_reporting --name=iops-test-
job --eta-newline=1 -readonly
```

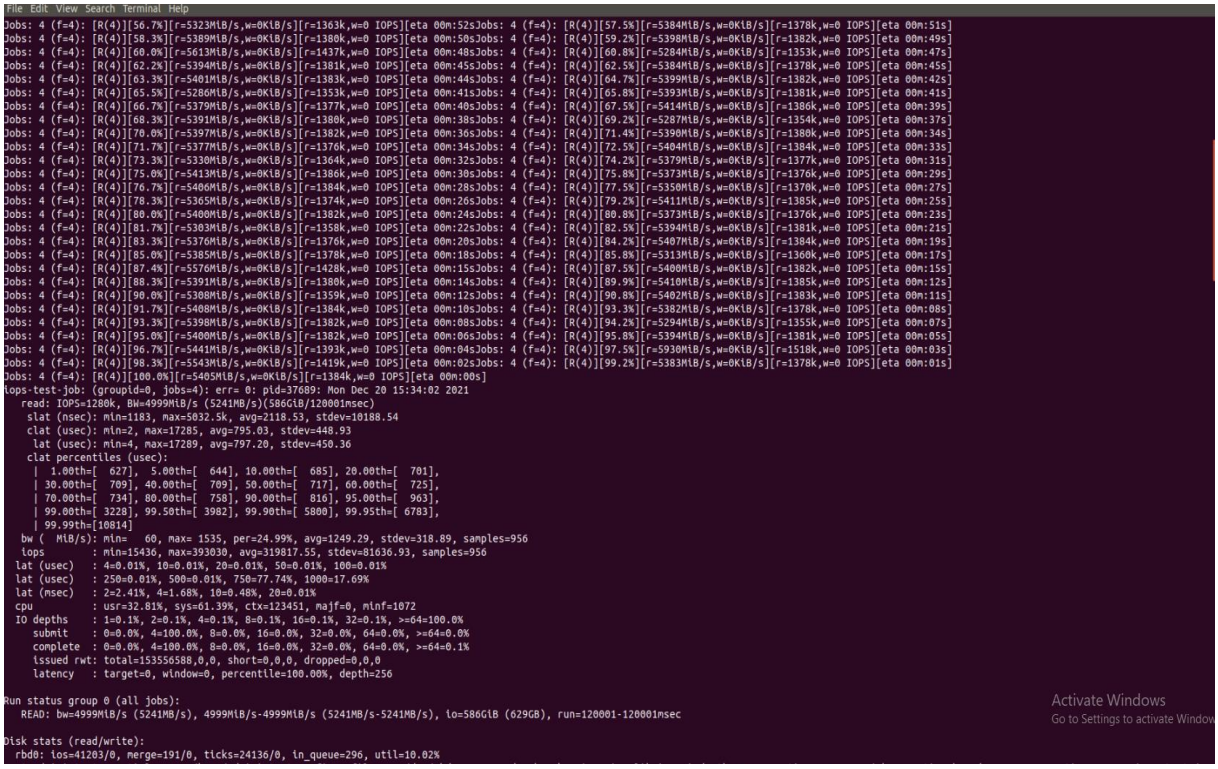
Scenario:

1. Object storage drives working normally – read bandwidth **– 5029 MB/s;** IOPS – 1287k; network – 1 Gb/s; num-jobs = 4
2. Failure of two object storage drives – read bandwidth – **4999 MB/s**; IOPS – 1280k; network – 1 Gb/s; num-jobs = 4
3. Speed test with 1 TB volume (block storage) and 1 MB block size – read bandwidth – **2813 MB/s**; IOPS – 2880; network – 1 Gb/s; num-jobs = 4
4. Speed test with 12 TB volume (block storage) and 4 KB block size – read bandwidth – **5231 MB/s**; IOPS – 1339k; network – 1 Gb/s; num-jobs = 32
5. Speed test with 1 TB volume (block storage) and 4 KB block size – read bandwidth – **2778 MB/s**; IOPS – 711k; network – 10 Gb/s; num-jobs = 4
6. Speed test with 1 TB volume (block storage) and 4 KB block size – read bandwidth – **4913 MB/s**; IOPS – 1258k; network – 10 Gb/s; num-jobs = 8

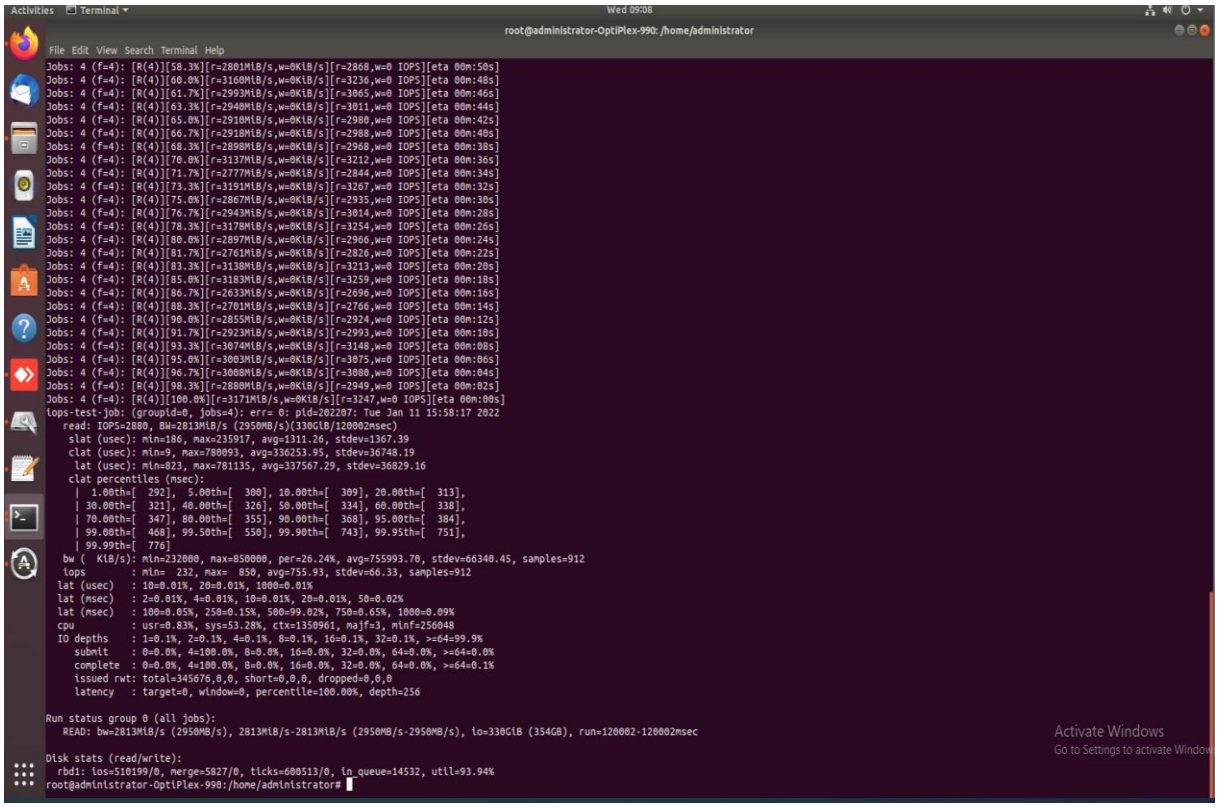
All Object storage drives working normally,



Simulating failure of 2 Object storage drives,



Speed test with 1TB volume (block storage) and 1 MB block size



Volume=12TB, block size=4k, num-jobs=32 (reducing num jobs reduces speed)

```
Jobs: 32 (f=32): [R(32)][70.0%][r=5372MiB/s,w=0KiB/s][r=1375k,w=0 IOPS][eta 00m:36s]
Jobs: 32 (f=32): [R(32)][71.7%][r=5251MiB/s,w=0KiB/s][r=1344k,w=0 IOPS][eta 00m:34s]
Jobs: 32 (f=32): [R(32)][73.3%][r=5149MiB/s,w=0KiB/s][r=1318k,w=0 IOPS][eta 00m:32s]
Jobs: 32 (f=32): [R(32)][75.0%][r=5121MiB/s,w=0KiB/s][r=1311k,w=0 IOPS][eta 00m:30s]
Jobs: 32 (f=32): [R(32)][76.7%][r=5204MiB/s,w=0KiB/s][r=1332k,w=0 IOPS][eta 00m:28s]
Jobs: 32 (f=32): [R(32)][78.3%][r=5346MiB/s,w=0KiB/s][r=1368k,w=0 IOPS][eta 00m:26s]
Jobs: 32 (f=32): [R(32)][80.0%][r=5329MiB/s,w=0KiB/s][r=1364k,w=0 IOPS][eta 00m:24s]
Jobs: 32 (f=32): [R(32)][81.7%][r=5229MiB/s,w=0KiB/s][r=1339k,w=0 IOPS][eta 00m:22s]
Jobs: 32 (f=32): [R(32)][83.3%][r=5225MiB/s,w=0KiB/s][r=1338k,w=0 IOPS][eta 00m:20s]
Jobs: 32 (f=32): [R(32)][85.0%][r=5403MiB/s,w=0KiB/s][r=1383k,w=0 IOPS][eta 00m:18s]
Jobs: 32 (f=32): [R(32)][86.7%][r=5318MiB/s,w=0KiB/s][r=1361k,w=0 IOPS][eta 00m:16s]
Jobs: 32 (f=32): [R(32)][88.3%][r=5290MiB/s,w=0KiB/s][r=1354k,w=0 IOPS][eta 00m:14s]
Jobs: 32 (f=32): [R(32)][90.0%][r=5272MiB/s,w=0KiB/s][r=1350k,w=0 IOPS][eta 00m:12s]
Jobs: 32 (f=32): [R(32)][91.7%][r=5187MiB/s,w=0KiB/s][r=1328k,w=0 IOPS][eta 00m:10s]
Jobs: 32 (f=32): [R(32)][93.3%][r=5356MiB/s,w=0KiB/s][r=1371k,w=0 IOPS][eta 00m:08s]
Jobs: 32 (f=32): [R(32)][95.0%][r=5384MiB/s,w=0KiB/s][r=1378k,w=0 IOPS][eta 00m:06s]
Jobs: 32 (f=32): [R(32)][96.7%][r=5179MiB/s,w=0KiB/s][r=1326k,w=0 IOPS][eta 00m:04s]
Jobs: 32 (f=32): [R(32)][98.3%][r=5253MiB/s,w=0KiB/s][r=1345k,w=0 IOPS][eta 00m:02s]
Jobs: 32 (f=32): [R(32)][100.0%][r=5299MiB/s,w=0KiB/s][r=1357k,w=0 IOPS][eta 00m:00s]
Jobs: 1 (f=1): [_(30),f(1),_(1)][100.0%][r=696MiB/s,w=0KiB/s][r=178k,w=0 IOPS][eta 00m:00s]
iops-test-job: (groupid=0, jobs=32): err= 0: pid=498466: Thu Jan 13 12:15:42 2022
   read: IOPS=1339k, BW=5231MiB/s (5486MB/s)(613GiB/120002msec)
    slat (nsec): min=1191, max=1442.6M, avg=17244.33, stdev=835293.05
    clat (usec): min=2, max=1444.2k, avg=3037.94, stdev=9915.28
     lat (usec): min=3, max=1444.2k, avg=3055.55, stdev=9948.11
    clat percentiles (usec):
     |  1.00th=[   515],  5.00th=[   562], 10.00th=[   586], 20.00th=[   619],
     | 30.00th=[   635], 40.00th=[   660], 50.00th=[   668], 60.00th=[   693],
     | 70.00th=[   717], 80.00th=[  1303], 90.00th=[  8586], 95.00th=[ 19268],
     | 99.00th=[ 32637], 99.50th=[ 36439], 99.90th=[ 46924], 99.95th=[ 52691],
     | 99.99th=[101188]
   bw (  KiB/s): min= 4600, max=845274, per=3.15%, avg=168941.76, stdev=28681.91, samples=7620
   iops        : min= 1150, max=211318, avg=42235.20, stdev=7170.47, samples=7620
  lat (usec)   : 4=0.01%, 10=0.01%, 20=0.01%, 50=0.01%, 100=0.01%
  lat (usec)   : 250=0.01%, 500=0.63%, 750=73.74%, 1000=4.57%
  lat (msec)   : 2=2.52%, 4=3.86%, 10=5.91%, 20=3.99%, 50=4.70%
  lat (msec)   : 100=0.06%, 250=0.01%, 500=0.01%, 750=0.01%, 1000=0.01%
  lat (msec)   : 2000=0.01%
  cpu          : usr=7.39%, sys=14.93%, ctx=448161, majf=0, minf=4415
  IO depths    : 1=0.1%, 2=0.1%, 4=0.1%, 8=0.1%, 16=0.1%, 32=0.1%, >=64=100.0%
     submit    : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
     complete  : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.1%
     issued rwt: total=160712766,0,0, short=0,0,0, dropped=0,0,0
     latency   : target=0, window=0, percentile=100.00%, depth=128

Run status group 0 (all jobs):
   READ: bw=5231MiB/s (5486MB/s), 5231MiB/s-5231MiB/s (5486MB/s-5486MB/s), io=613GiB (658GB), run=120002-120002msec

Disk stats (read/write):
  rbd1: ios=88472/0, merge=205/0, ticks=67947/0, in_queue=2628, util=91.78%
```

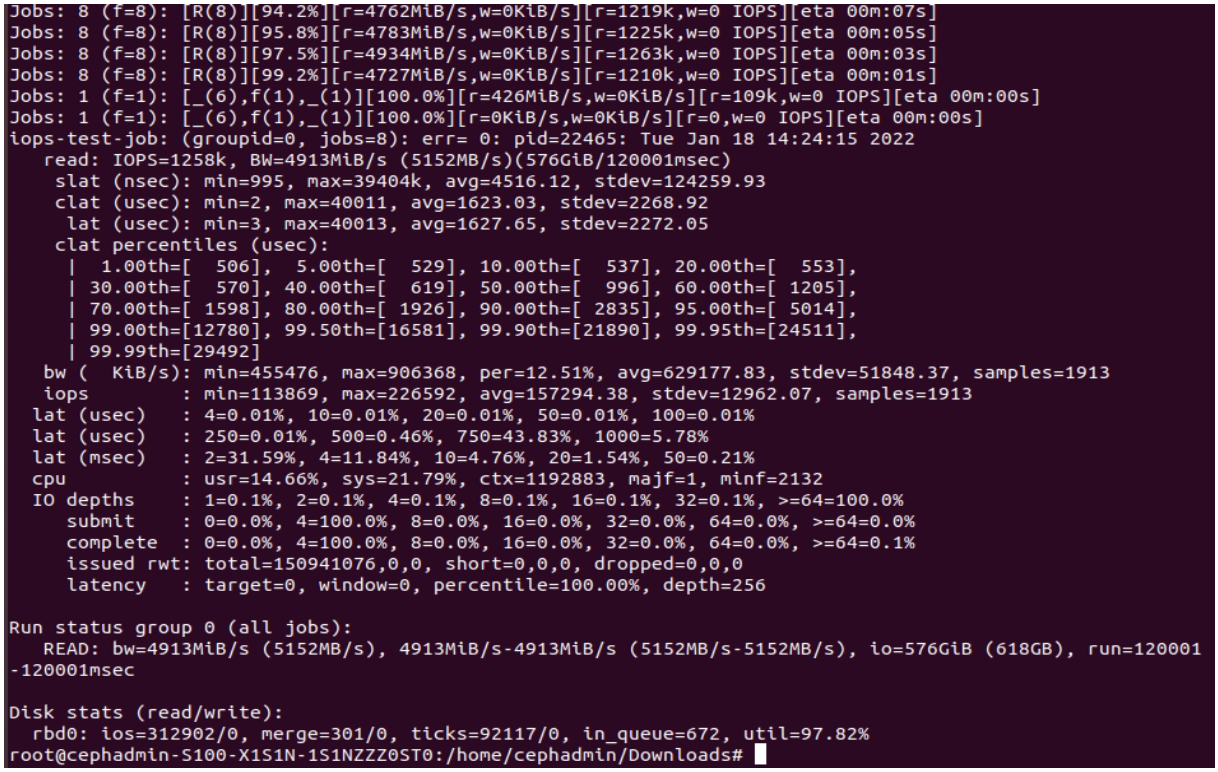
All the tests above are conducted in OptiPlex-990 with 1G network.

Client testing with 10G network

Volume=1TB, block size=4k, num-jobs=4

```
Jobs: 4 (f=4): [R(4)][97.5%][r=2765MiB/s,w=0KiB/s][r=708k,w=0 IOPS][eta 00m:03s]
Jobs: 4 (f=4): [R(4)][99.2%][r=2815MiB/s,w=0KiB/s][r=721k,w=0 IOPS][eta 00m:01s]
Jobs: 1 (f=1): [_(1),f(1),_(2)][100.0%][r=269MiB/s,w=0KiB/s][r=68.8k,w=0 IOPS][eta 00m:00s]
Jobs: 1 (f=1): [_(1),f(1),_(2)][100.0%][r=0KiB/s,w=0KiB/s][r=0,w=0 IOPS][eta 00m:00s]
iops-test-job: (groupid=0, jobs=4): err= 0: pid=22234: Tue Jan 18 14:08:47 2022
   read: IOPS=711k, BW=2778MiB/s (2913MB/s)(326GiB/120001msec)
    slat (nsec): min=966, max=31287k, avg=4541.68, stdev=58653.87
    clat (usec): min=2, max=31869, avg=1435.10, stdev=956.99
     lat (usec): min=85, max=31871, avg=1439.70, stdev=958.56
    clat percentiles (usec):
     |  1.00th=[  502],  5.00th=[  529], 10.00th=[  553], 20.00th=[  758],
     | 30.00th=[  988], 40.00th=[ 1139], 50.00th=[ 1287], 60.00th=[ 1450],
     | 70.00th=[ 1614], 80.00th=[ 1844], 90.00th=[ 2245], 95.00th=[ 2835],
     | 99.00th=[ 5211], 99.50th=[ 6456], 99.90th=[10028], 99.95th=[12125],
     | 99.99th=[16909]
   bw (  KiB/s): min=605720, max=865160, per=25.00%, avg=711103.75, stdev=34500.79, samples=956
   iops        : min=151430, max=216290, avg=177775.90, stdev=8625.20, samples=956
  lat (usec)   : 4=0.01%, 100=0.01%, 250=0.01%, 500=1.01%, 750=18.65%
  lat (usec)   : 1000=11.09%
  lat (msec)   : 2=54.18%, 4=12.95%, 10=2.01%, 20=0.10%, 50=0.01%
  cpu          : usr=16.63%, sys=25.07%, ctx=1325298, majf=1, minf=1071
  IO depths    : 1=0.1%, 2=0.1%, 4=0.1%, 8=0.1%, 16=0.1%, 32=0.1%, >=64=100.0%
     submit    : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
     complete  : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.1%
     issued rwt: total=85331289,0,0, short=0,0,0, dropped=0,0,0
     latency   : target=0, window=0, percentile=100.00%, depth=256

Run status group 0 (all jobs):
   READ: bw=2778MiB/s (2913MB/s), 2778MiB/s-2778MiB/s (2913MB/s-2913MB/s), io=326GiB (350GB), run=120001
-120001msec

Disk stats (read/write):
  rbd0: ios=399614/0, merge=614/0, ticks=119751/0, in_queue=2124, util=99.89%
```

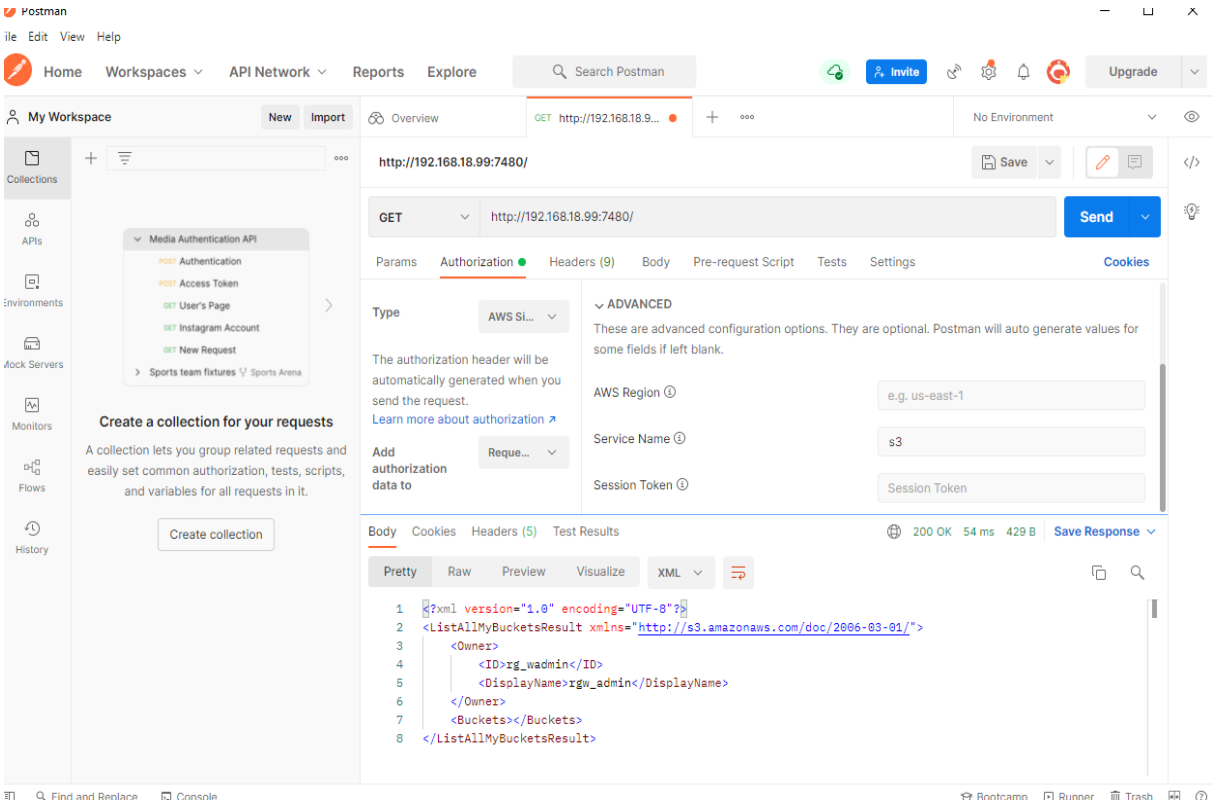
Volume=1TB, block size=4k, num-jobs=8



Object Storage test

Default connectivity to S3

Postman application provides means to connect and test Application Programming Interface (API). In Ceph, the object gateway is can be accessed based on Representational State transfer (RESTful) API and Swift API [22]. The postman application was installed from [23] in a Windows OS machine for the connectivity tests.



Test with S3

Installation

Download the stable release from warp [24]

```
sudo dpkg -i /path/to/warp_0.5.5_Linux_x86_64.deb
```

Benchmarking

General statistics

```
warp stat --host=<object-gateway-ip:port> --access-key=<access-key> --secret-key=<secret-key> --autoterm --concurrent=<concurrent-jobs> --obj.size=<object-size>
```

Mixed workload

```
warp mixed --host=<object-gateway-ip:port> --access-key=<access-key> --secret-key=<secret-key> --autoterm --concurrent=<concurrent-jobs> --obj.size=<object-size>
```

Results

1. General statistics benchmark from client node – object size = 4KB; PUT 110.02 MB/s, 28.17 obj/s; STAT 12409.92 obj/s; network =1 Gb/s
2. Mixed workload benchmark from client node – object size = 4KB; Cluster total 149.13 MB/s 63.7 obj/s; network = 1 Gb/s

3. General statistics benchmark from client node – object size = 4KB; PUT 664.95 MB/s, 170.23 obj/s; STAT 13687.53 obj/s; network =10 Gb/s
4. Mixed workload benchmark from client node – object size = 4KB; Cluster total 917.5 MB/s 391.34 obj/s; network = 10 Gb/s

General statistics benchmark from client node

```
root@administrator-OptiPlex-990:/home/administrator/Downloads# warp stat --host=192.168.18.99:80 --access-key=896BDPIMUFOUD3Q3XKFV
L --autoterm --concurrent=20 --obj.size=4096000
Throughput 11924.3 objects/s within 7.500000% for 10.08s. Assuming stability. Terminating benchmark.
warp: Benchmark data written to "warp-stat-2022-01-18[102543]-XPcv.csv.zst"

----------------------------------------
Operation: PUT
* Average: 110.02 MiB/s, 28.17 obj/s

Throughput, split into 353 x 1s:
 * Fastest: 111.6MiB/s, 28.58 obj/s
 * 50% Median: 110.0MiB/s, 28.17 obj/s
 * Slowest: 108.8MiB/s, 27.85 obj/s

----------------------------------------
Operation: STAT
* Average: 12409.92 obj/s

Throughput, split into 36 x 1s:
 * Fastest: 12905.50 obj/s
 * 50% Median: 12527.79 obj/s
 * Slowest: 11477.89 obj/s
warp: Cleanup Done.root@administrator-OptiPlex-990:/home/administrator/Downloads#
```

Mixed workload benchmark from client node

```
root@administrator-OptiPlex-990:/home/administrator/Downloads# warp mixed --host=192.168.18.99:80
nL --autoterm --concurrent=20 --obj.size=4096000
Throughput 18.9 objects/s within 7.500000% for 36.099s. Assuming stability. Terminating benchmark.
warp: Benchmark data written to "warp-mixed-2022-01-18[101036]-3CUR.csv.zst"
Mixed operations.
Operation: DELETE, 10%, Concurrency: 20, Ran 2m9s.
 * Throughput: 6.41 obj/s

Operation: GET, 45%, Concurrency: 20, Ran 2m9s.
 * Throughput: 111.88 MiB/s, 28.64 obj/s

Operation: PUT, 15%, Concurrency: 20, Ran 2m9s.
 * Throughput: 37.59 MiB/s, 9.62 obj/s

Operation: STAT, 30%, Concurrency: 20, Ran 2m9s.
 * Throughput: 19.10 obj/s

Cluster Total: 149.13 MiB/s, 63.70 obj/s over 2m9s.
warp: Cleanup Done.root@administrator-OptiPlex-990:/home/administrator/Downloads#
```

General statistics benchmark from client node with 10G network connection

```
root@cephadmin-S100-X1S1N-1S1NZZZ0ST0:/home/cephadmin/Downloads# warp stat --
ess-key=896BDPIMUFOUD3Q3XKFV --secret-key=PLQDuDvPsCFH3RGJ6e98kWfh1ZFlCisCDxA
=4096000
Throughput 14211.1 objects/s within 7.500000% for 10.08s. Assuming stability.
warp: Benchmark data written to "warp-stat-2022-01-18[103309]-Yz60.csv.zst"

----------------------------------------
Operation: PUT
* Average: 664.95 MiB/s, 170.23 obj/s

Throughput, split into 58 x 1s:
 * Fastest: 679.5MiB/s, 173.94 obj/s
 * 50% Median: 667.1MiB/s, 170.77 obj/s
 * Slowest: 622.0MiB/s, 159.24 obj/s

----------------------------------------
Operation: STAT
* Average: 13687.53 obj/s

Throughput, split into 36 x 1s:
 * Fastest: 14343.09 obj/s
 * 50% Median: 13789.26 obj/s
 * Slowest: 12684.35 obj/s
warp: Cleanup Done.root@cephadmin-S100-X1S1N-1S1NZZZ0ST0:/home/cephadmin/Down
```

Mixed workload benchmark from client node with 10G network connection

```
root@cephadmin-S100-X1S1N-1S1NZZZ0ST0:/home/cephadmin/Downloads# warp mixed --host=192.168.18.99:80 --ac
cess-key=896BDPIMUFOUD3Q3XKFV --secret-key=PLQDuDvPsCFH3RGJ6e98kWfh1ZFlCisCDxAByKnL --autoterm --obj.siz
e=4096000
Throughput 698.8MiB/s within 7.500000% for 10.269s. Assuming stability. Terminating benchmark.
warp: Benchmark data written to "warp-mixed-2022-01-18[104251]-d3eR.csv.zst"
Mixed operations.
Operation: DELETE, 10%, Concurrency: 20, Ran 36s.
 * Throughput: 39.02 obj/s

Operation: GET, 45%, Concurrency: 20, Ran 36s.
 * Throughput: 688.82 MiB/s, 176.34 obj/s

Operation: PUT, 15%, Concurrency: 20, Ran 36s.
 * Throughput: 230.10 MiB/s, 58.91 obj/s

Operation: STAT, 30%, Concurrency: 20, Ran 36s.
 * Throughput: 117.50 obj/s

Cluster Total: 917.50 MiB/s, 391.34 obj/s over 37s.
```

Test with swift

Installation

The following commands are to be executed in a ceph manager node [19]

1. `sudo radosgw-admin user create --uid="benchmark" --display-name="benchmark"`
2. `sudo radosgw-admin subuser create --uid=benchmark --subuser=benchmark:swift --access=full`
3. `sudo radosgw-admin key create --subuser=benchmark:swift --key-type=swift --secret=guessme`
4. `radosgw-admin user modify --uid=benchmark --max-buckets=0`

The following commands are executed in the client node

swift-bench using *pip install swift && pip install swift-bench*

Create a swift configuration file with the details of gateway (swift.conf)

```
[bench]

auth = http://gateway-ip/auth/v1.0

user = benchmark:swift

key = <key>

auth_version = 1.0
```


Benchmarking

```
swift-bench -c <concurrent-jobs> -s <object-size> -n <num-put> -g <num-get>
/path/to/swift.conf
```

Results

1. When concurrent jobs = 64; object size = 4KB; num-put =1000; num-get = 100; PUTS = 258.7 obj/s; GETS = 590.7 obj/s; DEL = 722.1 obj/s; network = 1 Gb/s
2. When concurrent jobs = 128; object size = 4KB; num-put =1000; num-get = 1000; PUTS = 629.8 obj/s; GETS = 644.5 obj/s; DEL = 694.0 obj/s; network = 1 Gb/s
3. When concurrent jobs = 128; object size = 10MB; num-put =1000; num-get = 1000; PUTS = 652.6 obj/s; GETS = 661.6 obj/s; DEL = 731.8 obj/s; network = 1 Gb/s
4. When concurrent jobs = 128; object size = 10MB; num-put =1000; num-get = 1000; PUTS = 609.8 obj/s; GETS = 619.8 obj/s; DEL = 696.7 obj/s; network = 1 Gb/s

```
root@administrator-OptiPlex-990:/home/administrator# swift-bench -c 64 -s 4096 -n 1000 -g 100 /home/administrator/swift.conf
swift-bench 2022-01-17 10:22:30,033 INFO Auth version: 1.0
swift-bench 2022-01-17 10:22:34,293 INFO Auth version: 1.0
swift-bench 2022-01-17 10:22:36,736 INFO 10 PUTS [0 failures], 4.1/s
swift-bench 2022-01-17 10:22:38,172 INFO 1000 PUTS **FINAL** [0 failures], 258.7/s
swift-bench 2022-01-17 10:22:38,172 INFO Auth version: 1.0
swift-bench 2022-01-17 10:22:38,352 INFO 100 GETS **FINAL** [0 failures], 590.7/s
swift-bench 2022-01-17 10:22:38,352 INFO Auth version: 1.0
swift-bench 2022-01-17 10:22:39,747 INFO 1000 DEL **FINAL** [0 failures], 722.1/s
swift-bench 2022-01-17 10:22:39,747 INFO Auth version: 1.0
```

```
root@administrator-OptiPlex-990:/home/administrator# swift-bench -c 128 -s 4096 -n 1000 -g 1000 /home/administrator/swift.conf
swift-bench 2022-01-17 10:25:49,740 INFO Auth version: 1.0
swift-bench 2022-01-17 10:25:50,117 INFO Auth version: 1.0
swift-bench 2022-01-17 10:25:51,724 INFO 1000 PUTS **FINAL** [0 failures], 629.8/s
swift-bench 2022-01-17 10:25:51,724 INFO Auth version: 1.0
swift-bench 2022-01-17 10:25:53,293 INFO 1000 GETS **FINAL** [0 failures], 644.5/s
swift-bench 2022-01-17 10:25:53,293 INFO Auth version: 1.0
swift-bench 2022-01-17 10:25:54,752 INFO 1000 DEL **FINAL** [0 failures], 694.0/s
swift-bench 2022-01-17 10:25:54,752 INFO Auth version: 1.0
```

```
root@administrator-OptiPlex-990:/home/administrator# swift-bench -c 64 -s 10240 -n 1000 -g 1000 /home/administrator/swift.conf
swift-bench 2022-01-17 10:58:09,539 INFO Auth version: 1.0
swift-bench 2022-01-17 10:58:09,897 INFO Auth version: 1.0
swift-bench 2022-01-17 10:58:11,438 INFO 1000 PUTS **FINAL** [0 failures], 652.6/s
swift-bench 2022-01-17 10:58:11,438 INFO Auth version: 1.0
swift-bench 2022-01-17 10:58:12,959 INFO 1000 GETS **FINAL** [0 failures], 661.6/s
swift-bench 2022-01-17 10:58:12,959 INFO Auth version: 1.0
swift-bench 2022-01-17 10:58:14,336 INFO 1000 DEL **FINAL** [0 failures], 731.8/s
swift-bench 2022-01-17 10:58:14,336 INFO Auth version: 1.0
```

```
root@administrator-OptiPlex-990:/home/administrator# swift-bench -c 128 -s 10240 -n 1000 -g 1000 /home/administrator/swift.conf
swift-bench 2022-01-17 10:26:32,374 INFO Auth version: 1.0
swift-bench 2022-01-17 10:26:32,742 INFO Auth version: 1.0
swift-bench 2022-01-17 10:26:34,401 INFO 1000 PUTS **FINAL** [0 failures], 609.8/s
swift-bench 2022-01-17 10:26:34,401 INFO Auth version: 1.0
swift-bench 2022-01-17 10:26:36,032 INFO 1000 GETS **FINAL** [0 failures], 619.8/s
swift-bench 2022-01-17 10:26:36,032 INFO Auth version: 1.0
swift-bench 2022-01-17 10:26:37,486 INFO 1000 DEL **FINAL** [0 failures], 696.7/s
swift-bench 2022-01-17 10:26:37,486 INFO Auth version: 1.0
root@administrator-OptiPlex-990:/home/administrator#
```


Although *swift-bench* measures performance in number of objects/secs, it's easy enough to convert this into MB/sec, by multiplying by the size of each object. However, you should be wary of comparing this directly with the baseline disk performance statistics you obtained earlier, since several other factors also influence these statistics, such as: [19]

- the level of replication (and latency overhead)
- full data journal writes (offset in some situations by journal data coalescing)
- fsync on the object storage drives to guarantee data safety
- metadata overhead for keeping data stored in RADOS
- latency overhead (network, ceph, etc) makes readahead more important

Simulating failures

The simulation involved 1000 iterations, simulating drive by drive failure in various orders of drives failing [25].

```
End of simulation: Out of 1000 double failures, 1718 caused a data loss incident

End of simulation: Out of 10 1 disk failures, 0 caused a data loss incident

End of simulation: Out of 10 2 disk failures, 0 caused a data loss incident

End of simulation: Out of 10 3 disk failures, 31 caused a data loss incident

End of simulation: Out of 10 4 disk failures, 109 caused a data loss incident

End of simulation: Out of 10 5 disk failures, 237 caused a data loss incident

End of simulation: Out of 10 6 disk failures, 477 caused a data loss incident

End of simulation: Out of 10 7 disk failures, 863 caused a data loss incident

End of simulation: Out of 10 8 disk failures, 1400 caused a data loss incident
```

Acknowledging crash warnings

Execute the following command in one of the manager nodes to remove crash warnings, [26]

```
ceph crash archive archive-all
```

References

[1] Ceph, "Ceph Homepage," [Online]. Available: https://ceph.com/en/. [Accessed 2022].

[2] Ceph, "Ceph Glossary," [Online]. Available: https://docs.ceph.com/en/pacific/glossary/. [Accessed 2022].

[3] Hyperscalers, "S2S Tier 1," [Online]. Available: https://www.hyperscalers.com.au/S2S-T1-server-compare-UCS-M4C240-HP-DL180-buy-T41S-2U. [Accessed 2022].

[4] Hyperscalers, "Storage servers," [Online]. Available: https://www.hyperscalers.com/storage/storage-servers. [Accessed 2022].

[5] Ceph, "Deploying a new Ceph cluster," [Online]. Available: https://docs.ceph.com/en/pacific/cephadm/install/#requirements. [Accessed 2022].

[6] Docker docs, "Get Docker," [Online]. Available: https://docs.docker.com/get-docker/.

[7] Liquid web, "Enable root login via ssh in Ubuntu," [Online]. Available: https://www.liquidweb.com/kb/enable-root-login-via-ssh/.

[8] Ceph, "Preflight Checklist," [Online]. Available: https://docs.ceph.com/en/mimic/start/quick-start-preflight/. [Accessed 2022].

[9] Ceph, "Storage cluster quickstart," [Online]. Available: https://docs.ceph.com/en/mimic/start/quick-ceph-deploy/. [Accessed 2022].

[10] Ceph, "Ceph Dashboard," [Online]. Available: https://docs.ceph.com/en/latest/mgr/dashboard/. [Accessed 2022].

[11] Ceph, "CephFS quickstart," [Online]. Available: https://docs.ceph.com/en/mimic/start/quick-cephfs/. [Accessed 2022].

[12] Ceph, "CephFS client capabilities," [Online]. Available: https://docs.ceph.com/en/nautilus/cephfs/client-auth/#cephfs-client-capabilities. [Accessed 2022].

[13] Ceph, "Object storage quickstart," [Online]. Available: https://docs.ceph.com/en/mimic/start/quick-rgw/. [Accessed 2022].

[14] Ceph, "Prometheus module," [Online]. Available: https://docs.ceph.com/en/latest/mgr/prometheus/. [Accessed 2022].

[15] Grafana Labs, "Ceph- cluster," [Online]. Available: https://grafana.com/grafana/dashboards/7056. [Accessed 2022].

[16] Ceph, "Orchestrator CLI," [Online]. Available: https://docs.ceph.com/en/latest/mgr/orchestrator/. [Accessed 2022].

[17] Ceph, "Converting an existing cluster to Cephadm," [Online]. Available: https://docs.ceph.com/en/pacific/cephadm/adoption/. [Accessed 2022].

[18] Ceph, "Block device quickstart," [Online]. Available: https://docs.ceph.com/en/mimic/start/quick-rbd/. [Accessed 2022].

[19] Ceph, "Benchmark Ceph Cluster Performance," [Online]. Available: https://tracker.ceph.com/projects/ceph/wiki/Benchmark_Ceph_Cluster_Performance. [Accessed 2022].

[20] Ceph, "what's the difference between pg and pgp?," [Online]. Available: http://lists.ceph.com/pipermail/ceph-users-ceph.com/2015-May/001610.html. [Accessed 2022].

[21] J. Axboe, "FIO Documentation," [Online]. Available: https://fio.readthedocs.io/en/latest/fio_doc.html. [Accessed 2022].

[22] Ceph, "Ceph Object gateway," [Online]. Available: https://docs.ceph.com/en/pacific/radosgw/index.html. [Accessed 2022].

[23] Postman, "Download Postman," [Online]. Available: https://www.postman.com/downloads/. [Accessed 2022].

[24] MinIO, "S3 Warp," [Online]. Available: https://github.com/minio/warp. [Accessed 2022].

[25] cernceph, "ceph-scripts," [Online]. Available: https://github.com/cernceph/ceph-scripts. [Accessed 2022].

[26] Ceph, "Crash Module," [Online]. Available: https://docs.ceph.com/en/latest/mgr/crash/. [Accessed 2022].