BLUESTORE MIGRATION

Each OSD can run either BlueStore or FileStore, and a single Ceph cluster can contain a mix of both. Users who have previously deployed FileStore are likely to want to transition to BlueStore in order to take advantage of the improved performance and robustness. There are several strategies for making such a transition.

An individual OSD cannot be converted in place in isolation, however: BlueStore and FileStore are simply too different for that to be practical. “Conversion” will rely either on the cluster’s normal replication and healing support or tools and strategies that copy OSD content from an old (FileStore) device to a new (BlueStore) one.

DEPLOY NEW OSDS WITH BLUESTORE

Any new OSDs (e.g., when the cluster is expanded) can be deployed using BlueStore. This is the default behavior so no specific change is needed.

Similarly, any OSDs that are reprovisioned after replacing a failed drive can use BlueStore.

CONVERT EXISTING OSDS

MARK OUT AND REPLACE

The simplest approach is to mark out each device in turn, wait for the data to rereplicate across the cluster, reprovision the OSD, and mark it back in again. It is simple and easy to automate. However, it requires more data migration than should be necessary, so it is not optimal.

1. Identify a FileStore OSD to replace:

   ```
   ID=<osd-id-number>
   DEVICE=<disk-device>
   ```

   You can tell whether a given OSD is FileStore or BlueStore with:

   ```
   ceph osd metadata $ID | grep osd_objectstore
   ```

   You can get a current count of filestore vs bluestore with:

   ```
   ceph osd count-metadata osd_objectstore
   ```

2. Mark the filestore OSD out:

   ```
   ceph osd out $ID
   ```

3. Wait for the data to migrate off the OSD in question:

   ```
   while ! ceph osd safe-to-destroy $ID ; sleep 60 ; done
   ```

4. Stop the OSD:

   ```
   systemctl kill ceph-osd@$ID
   ```

5. Make note of which device this OSD is using:

   ```
   mount | grep /var/lib/ceph/osd/ceph-$ID
   ```

6. Unmount the OSD:

```
umount /var/lib/ceph/osd/ceph-$ID
```

7. Destroy the OSD data. Be *EXTREMELY CAREFUL* as this will destroy the contents of the device; be certain the data on the device is not needed (i.e., that the cluster is healthy) before proceeding.

```
ceph-volume lvm zap $DEVICE
```

8. Tell the cluster the OSD has been destroyed (and a new OSD can be reprovisioned with the same ID):

```
ceph osd destroy $ID --yes-i-really-mean-it
```

9. Reprovision a BlueStore OSD in its place with the same OSD ID. This requires you do identify which device to wipe based on what you saw mounted above. BE CAREFUL!

```
ceph-volume lvm create --bluestore --data $DEVICE --osd-id $ID
```

10. Repeat.

You can allow the refilling of the replacement OSD to happen concurrently with the draining of the next OSD, or follow the same procedure for multiple OSDs in parallel, as long as you ensure the cluster is fully clean (all data has all replicas) before destroying any OSDs. Failure to do so will reduce the redundancy of your data and increase the risk of (or potentially even cause) data loss.

Advantages:

- Simple.
- Can be done on a device-by-device basis.
- No spare devices or hosts are required.

Disadvantages:

- Data is copied over the network twice: once to some other OSD in the cluster (to maintain the desired number of replicas), and then again back to the reprovisioned BlueStore OSD.

WHOLE HOST REPLACEMENT

If you have a spare host in the cluster, or have sufficient free space to evacuate an entire host in order to use it as a spare, then the conversion can be done on a host-by-host basis with each stored copy of the data migrating only once.

First, you need have empty host that has no data. There are two ways to do this: either by starting with a new, empty host that isn’t yet part of the cluster, or by offloading data from an existing host that in the cluster.

USE A NEW, EMPTY HOST

Ideally the host should have roughly the same capacity as other hosts you will be converting (although it doesn’t strictly matter).

```
NEWHOST=<empty-host-name>
```

Add the host to the CRUSH hierarchy, but do not attach it to the root:

```
ceph osd crush add-bucket $NEWHOST host
```

Make sure the ceph packages are installed.

USE AN EXISTING HOST

If you would like to use an existing host that is already part of the cluster, and there is sufficient free space on that host so that all of its data can be migrated off, then you can instead do:

```
OLDHOST=<existing-cluster-host-to-offload>
ceph osd crush unlink $OLDHOST default
```

where “default” is the immediate ancestor in the CRUSH map. (For smaller clusters with unmodified configurations this will normally be “default”, but it might also be a rack name.) You should now see the host at the top of the OSD tree output with no parent:

```
$ bin/ceph osd tree
ID CLASS WEIGHT  TYPE NAME     STATUS REWEIGHT PRI-AFF
-5             0 host oldhost
10   ssd 1.00000     osd.10        up  1.00000 1.00000
11   ssd 1.00000     osd.11        up  1.00000 1.00000
12   ssd 1.00000     osd.12        up  1.00000 1.00000
-1       3.00000 root default
-2       3.00000     host foo
 0   ssd 1.00000         osd.0     up  1.00000 1.00000
 1   ssd 1.00000         osd.1     up  1.00000 1.00000
 2   ssd 1.00000         osd.2     up  1.00000 1.00000
...
```

If everything looks good, jump directly to the “Wait for data migration to complete” step below and proceed from there to clean up the old OSDs.

MIGRATION PROCESS

If you’re using a new host, start at step #1. For an existing host, jump to step #5 below.

1. Provision new BlueStore OSDs for all devices:

   ```
   ceph-volume lvm create --bluestore --data /dev/$DEVICE
   ```

2. Verify OSDs join the cluster with:

   ```
   ceph osd tree
   ```

   You should see the new host `$NEWHOST` with all of the OSDs beneath it, but the host should *not* be nested beneath any other node in hierarchy (like `root default`). For example, if `newhost` is the empty host, you might see something like:

   ```
   $ bin/ceph osd tree
   ID CLASS WEIGHT  TYPE NAME     STATUS REWEIGHT PRI-AFF
   -5             0 host newhost
   10   ssd 1.00000     osd.10        up  1.00000 1.00000
   11   ssd 1.00000     osd.11        up  1.00000 1.00000
   12   ssd 1.00000     osd.12        up  1.00000 1.00000
   -1       3.00000 root default
   -2       3.00000     host oldhost1
    0   ssd 1.00000         osd.0     up  1.00000 1.00000
    1   ssd 1.00000         osd.1     up  1.00000 1.00000
    2   ssd 1.00000         osd.2     up  1.00000 1.00000
   ...
   ```

3. Identify the first target host to convert

   ```
   OLDHOST=<existing-cluster-host-to-convert>
   ```

4. Swap the new host into the old host’s position in the cluster:

   ```
   ceph osd crush swap-bucket $NEWHOST $OLDHOST
   ```

At this point all data on `$OLDHOST` will start migrating to OSDs on `$NEWHOST`. If there is a difference in the total capacity of the old and new hosts you may also see some data migrate to or from other nodes in the cluster, but as long as the hosts are similarly sized this will be a relatively small amount of data.

5. Wait for data migration to complete:

```
while ! ceph osd safe-to-destroy $(ceph osd ls-tree $OLDHOST); do sleep 60 ; done
```

6. Stop all old OSDs on the now-empty `$OLDHOST`:

```
ssh $OLDHOST
systemctl kill ceph-osd.target
umount /var/lib/ceph/osd/ceph-*
```

7. Destroy and purge the old OSDs:

```
for osd in `ceph osd ls-tree $OLDHOST`; do
    ceph osd purge $osd --yes-i-really-mean-it
done
```

8. Wipe the old OSD devices. This requires you do identify which devices are to be wiped manually (BE CAREFUL!). For each device,:

```
ceph-volume lvm zap $DEVICE
```

9. Use the now-empty host as the new host, and repeat:

```
NEWHOST=$OLDHOST
```

Advantages:

- Data is copied over the network only once.
- Converts an entire host's OSDs at once.
- Can parallelize to converting multiple hosts at a time.
- No spare devices are required on each host.

Disadvantages:

- A spare host is required.
- An entire host's worth of OSDs will be migrating data at a time. This is like likely to impact overall cluster performance.
- All migrated data still makes one full hop over the network.

PER-OSD DEVICE COPY

A single logical OSD can be converted by using the `copy` function of `ceph-objectstore-tool`. This requires that the host have a free device (or devices) to provision a new, empty BlueStore OSD. For example, if each host in your cluster has 12 OSDs, then you'd need a 13th available device so that each OSD can be converted in turn before the old device is reclaimed to convert the next OSD.

Caveats:

- This strategy requires that a blank BlueStore OSD be prepared without allocating a new OSD ID, something that the `ceph-volume` tool doesn't support. More importantly, the setup of *dmcrypt* is closely tied to the OSD identity, which means that this approach does not work with encrypted OSDs.
- The device must be manually partitioned.
- Tooling not implemented!
- Not documented!

Advantages:

- Little or no data migrates over the network during the conversion.

Disadvantages:

- Tooling not fully implemented.
- Process not documented.
- Each host must have a spare or empty device.
- The OSD is offline during the conversion, which means new writes will be written to only a subset of the OSDs. This increases the risk of data loss due to a subsequent failure. (However, if there is a failure before conversion is complete, the original FileStore OSD can be started to provide access to its original data.)