OSD Scenarios

The following are all of the available options for the `osd_scenario` config setting. Defining an `osd_scenario` is mandatory for using `ceph-ansible`.

collocated

This OSD scenario uses `ceph-disk` to create OSDs with collocated journals from raw devices.

Use `osd_scenario: collocated` to enable this scenario. This scenario also has the following required configuration options:

- `devices`

This scenario has the following optional configuration options:

- `osd_objectstore`: defaults to `filestore` if not set. Available options are `filestore` or `bluestore`. You can only select `bluestore` if the ceph release is Luminous or greater.
- `dmcrypt`: defaults to `false` if not set.

This scenario supports encrypting your OSDs by setting `dmcrypt: True`.

If `osd_objectstore: filestore` is enabled both 'ceph data' and 'ceph journal' partitions will be stored on the same device.

If `osd_objectstore: bluestore` is enabled 'ceph data', 'ceph block', 'ceph block.db', 'ceph block.wal' will be stored on the same device. The device will get 2 partitions:

- One for 'data', called 'ceph data'
- One for 'ceph block', 'ceph block.db', 'ceph block.wal' called 'ceph block'

Example of what you will get:

```
[root@ceph-osd0 ~]# blkid /dev/sda*
/dev/sda: PTTYPE="gpt"
/dev/sda1: UUID="9c43e346-dd6e-431f-92d8-cbed4ccb25f6" TYPE="xfs" PARTLABEL="ceph data
/dev/sda2: PARTLABEL="ceph block" PARTUUID="e6ca3e1d-4702-4569-abfa-e285de328e9d"
```

An example of using the `collocated` OSD scenario with encryption would look like:

```
osd_scenario: collocated
dmcrypt: true
devices:
  - /dev/sda
  - /dev/sdb
```

non-collocated

This OSD scenario uses `ceph-disk` to create OSDs from raw devices with journals that exist on a dedicated device.

Use `osd_scenario: non-collocated` to enable this scenario. This scenario also has the following required configuration options:

- `devices`

This scenario has the following optional configuration options:

- `dedicated_devices`: defaults to `devices` if not set
- `osd_objectstore`: defaults to `filestore` if not set. Available options are `filestore` or

`bluestore`. You can only select `bluestore` with the ceph release is Luminous or greater.
- `dmcrypt`: defaults to `false` if not set.

This scenario supports encrypting your OSDs by setting `dmcrypt: True`.

If `osd_objectstore: filestore` is enabled 'ceph data' and 'ceph journal' partitions will be stored on different devices: - 'ceph data' will be stored on the device listed in `devices` - 'ceph journal' will be stored on the device listed in `dedicated_devices`

Let's take an example, imagine `devices` was declared like this:

```
devices:
  - /dev/sda
  - /dev/sdb
  - /dev/sdc
  - /dev/sdd
```

And `dedicated_devices` was declared like this:

```
dedicated_devices:
  - /dev/sdf
  - /dev/sdf
  - /dev/sdg
  - /dev/sdg
```

This will result in the following mapping:

- /dev/sda will have /dev/sdf1 as journal
- /dev/sdb will have /dev/sdf2 as a journal
- /dev/sdc will have /dev/sdg1 as a journal
- /dev/sdd will have /dev/sdg2 as a journal

> ## Note:
>
> On a containerized scenario we only support A SINGLE journal for all the OSDs on a given machine. If you don't, bad things will happen This is a limitation we plan to fix at some point.

If `osd_objectstore: bluestore` is enabled, both 'ceph block.db' and 'ceph block.wal' partitions will be stored on a dedicated device.

So the following will happen:

- The devices listed in `devices` will get 2 partitions, one for 'block' and one for 'data'. 'data' is only 100MB big and do not store any of your data, it's just a bunch of Ceph metadata. 'block' will store all your actual data.
- The devices in `dedicated_devices` will get 1 partition for RocksDB DB, called 'block.db' and one for RocksDB WAL, called 'block.wal'

By default `dedicated_devices` will represent block.db

Example of what you will get:

```
[root@ceph-osd0 ~]# blkid /dev/sd*
/dev/sda: PTTYPE="gpt"
/dev/sda1: UUID="c6821801-2f21-4980-add0-b7fc8bd424d5" TYPE="xfs" PARTLABEL="ceph data
/dev/sda2: PARTLABEL="ceph block" PARTUUID="ea454807-983a-4cf2-899e-b2680643bc1c"
/dev/sdb: PTTYPE="gpt"
/dev/sdb1: PARTLABEL="ceph block.db" PARTUUID="af5b2d74-4c08-42cf-be57-7248c739e217"
/dev/sdb2: PARTLABEL="ceph block.wal" PARTUUID="af3f8327-9aa9-4c2b-a497-cf0fe96d126a"
```

There is more device granularity for Bluestore ONLY if `osd_objectstore: bluestore` is

enabled by setting the `bluestore_wal_devices` config option.

By default, if `bluestore_wal_devices` is empty, it will get the content of `dedicated_devices`. If set, then you will have a dedicated partition on a specific device for block.wal.

Example of what you will get:

```
[root@ceph-osd0 ~]# blkid /dev/sd*
/dev/sda: PTTYPE="gpt"
/dev/sda1: UUID="39241ae9-d119-4335-96b3-0898da8f45ce" TYPE="xfs" PARTLABEL="ceph data
/dev/sda2: PARTLABEL="ceph block" PARTUUID="bff8e54e-b780-4ece-aa16-3b2f2b8eb699"
/dev/sdb: PTTYPE="gpt"
/dev/sdb1: PARTLABEL="ceph block.db" PARTUUID="0734f6b6-cc94-49e9-93de-ba7e1d5b79e3"
/dev/sdc: PTTYPE="gpt"
/dev/sdc1: PARTLABEL="ceph block.wal" PARTUUID="824b84ba-6777-4272-bbbd-bfe2a25cecf3"
```

An example of using the `non-collocated` OSD scenario with encryption, bluestore and dedicated wal devices would look like:

```
osd_scenario: non-collocated
osd_objectstore: bluestore
dmcrypt: true
devices:
  - /dev/sda
  - /dev/sdb
dedicated_devices:
  - /dev/sdc
  - /dev/sdc
bluestore_wal_devices:
  - /dev/sdd
  - /dev/sdd
```

lvm

This OSD scenario uses `ceph-volume` to create OSDs from logical volumes and is only available when the ceph release is Luminous or newer.

> ## Note:
>
> The creation of the logical volumes is not supported by `ceph-ansible`, `ceph-volume` only creates OSDs from existing logical volumes.

`lvm_volumes` is the config option that needs to be defined to configure the mappings for devices to be deployed. It is a list of dictionaries which expects a volume name and a volume group for logical volumes, but can also accept a partition in the case of `filestore` for the `journal`.

This scenario supports encrypting your OSDs by setting `dmcrypt: True`. If set, all OSDs defined in `lvm_volumes` will be encrypted.

The `data` key represents the logical volume name, raw device or partition that is to be used for your OSD data. The `data_vg` key represents the volume group name that your `data` logical volume resides on. This key is required for purging of OSDs created by this scenario.

> ## Note:
>
> Any logical volume or logical group used in `lvm_volumes` must be a name and not a path.

> ## Note:

You can not use the same journal for many OSDs.

filestore

There is filestore support which can be enabled with:

```
osd_objectstore: filestore
```

To configure this scenario use the `lvm_volumes` config option. `lvm_volumes` is a list of dictionaries which expects a volume name and a volume group for logical volumes, but can also accept a parition in the case of `filestore` for the `journal`.

The following keys are accepted for a `filestore` deployment:

- `data`
- `data_vg` (not required if `data` is a raw device or partition)
- `journal`
- `journal_vg` (not required if `journal` is a partition and not a logical volume)
- `crush_device_class` (optional, sets the crush device class for the OSD)

The `journal` key represents the logical volume name or partition that will be used for your OSD journal.

For example, a configuration to use the `lvm` osd scenario would look like:

```
osd_objectstore: filestore
osd_scenario: lvm
lvm_volumes:
  - data: data-lv1
    data_vg: vg1
    journal: journal-lv1
    journal_vg: vg2
    crush_device_class: foo
  - data: data-lv2
    journal: /dev/sda
    data_vg: vg1
  - data: data-lv3
    journal: /dev/sdb1
    data_vg: vg2
  - data: /dev/sda
    journal: /dev/sdb1
  - data: /dev/sda1
    journal: journal-lv1
    journal_vg: vg2
```

For example, a configuration to use the `lvm` osd scenario with encryption would look like:

```
osd_objectstore: filestore
osd_scenario: lvm
dmcrypt: True
lvm_volumes:
  - data: data-lv1
    data_vg: vg1
    journal: journal-lv1
    journal_vg: vg2
    crush_device_class: foo
```

bluestore

This scenario allows a combination of devices to be used in an OSD. `bluestore` can work just with a single “block” device (specified by the `data` and optionally `data_vg`) or additionally with a `block.wal` and `block.db` (interchangeably)

The following keys are accepted for a `bluestore` deployment:

- `data` (required)
- `data_vg` (not required if `data` is a raw device or partition)
- `db` (optional for `block.db`)
- `db_vg` (optional for `block.db`)
- `wal` (optional for `block.wal`)
- `wal_vg` (optional for `block.wal`)
- `crush_device_class` (optional, sets the crush device class for the OSD)

A `bluestore` lvm deployment, for all four different combinations supported could look like:

```
  osd_objectstore: bluestore
  osd_scenario: lvm
  lvm_volumes:
    - data: data-lv1
      data_vg: vg1
      crush_device_class: foo
    - data: data-lv2
      data_vg: vg1
      wal: wal-lv1
      wal_vg: vg2
    - data: data-lv3
      data_vg: vg2
      db: db-lv1
      db_vg: vg2
    - data: data-lv4
      data_vg: vg4
      db: db-lv4
      db_vg: vg4
      wal: wal-lv4
      wal_vg: vg4
    - data: /dev/sda
```