```
[ceph: root@clienta /]# rbd pool init benchpool
```

4. Open a second terminal and log in to the `clienta` node as the `admin` user. Use the first terminal to generate a workload and use the second terminal to collect metrics. Run a write test to the RBD pool `benchpool`. This might take several minutes to complete.

> **Note**
> This step requires sufficient time to complete the write OPS for the test. Be prepared to run the `osd pref` command in the second terminal immediately after starting the `benchpool` command in the first terminal.

4.1. Open a second terminal. Log in to `clienta` as the `admin` user and use `sudo` to run the `cephadm` shell.

```
[student@workstation ~]$ ssh admin@clienta
[admin@clienta ~]$ sudo cephadm shell
[ceph: root@clienta /]#
```

4.2. In the first terminal, generate the workload.

```
[ceph: root@clienta /]# rados -p benchpool bench 30 write
hints = 1
Maintaining 16 concurrent writes of 4194304 bytes to objects of size 4194304 for
 up to 30 seconds or 0 objects
Object prefix: benchmark_data_clienta.lab.example.com_50
  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        58        42   167.988       168    0.211943    0.322053
    2      16       112        96   191.982       216    0.122236    0.288171
    3      16       162       146   194.643       200    0.279456    0.300593
    4      16       217       201   200.975       220    0.385703    0.292009
...output omitted...
```

4.3. In the second terminal, collect performance metrics. The `commit_latency` data is the time for the OSD to write and commit the operation to its journal. The `apply_latency` data is the time to apply the write operation to the OSD file system back end. Note the OSD ID where the heavy load is occurring. Your OSD output might be different in your lab environment.

```
[ceph: root@clienta /]# ceph osd perf
osd  commit_latency(ms)  apply_latency(ms)
osd  commit_latency(ms)  apply_latency(ms)
  7                  94                 94
  8                 117                117
  6                 195                195
  1                  73                 73
  0                  72                 72
  2                  80                 80
```