Canonical

MicroCeph Deployment Guide

Contents

1. Getting started with MicroCeph

This tutorial will guide you through your first steps with MicroCeph. We will deploy a Ceph cluster on a single node using MicroCeph and store a JPEG image in an S3 bucket managed by MicroCeph.

1.1. How you'll do It

You will install MicroCeph, initialise the cluster, and add storage. Then, you will enable the S3-compatible Ceph Object Gateway (RGW) on your node and create an S3 bucket. Finally, you will upload an image to the bucket, consuming the storage via RGW.

As we progress, you will also interact with your cluster by checking its health, adding disks, and enabling RGW.

By the end of this tutorial, after successfully using MicroCeph to store an image, you will have a foundational understanding of how MicroCeph works, and be ready to explore more advanced use cases.

1.2. What you'll need

- The latest Ubuntu LTS version. Find Ubuntu release information here[1].
- 2 CPU cores
- 4 GiB RAM
- 12GiB disk space
- An Internet connection

1.3. Install MicroCeph

First, install MicroCeph as a snap package from the Snap Store:

```
sudo snap install microceph
```

Disable the default automatic Snap upgrades to prevent MicroCeph from being updated automatically:

```
sudo snap refresh --hold microceph
```

Caution:

Failing to set this option may result in unintended upgrades, which could critically impact your deployed cluster. To prevent this, all subsequent MicroCeph upgrades must be performed manually.

1.4. Initialise your cluster

Next, bootstrap your new Ceph storage cluster:

[1] https://ubuntu.com/about/release-cycle

```
sudo microceph cluster bootstrap
```

This process takes 3 to 5 seconds.

Check the cluster status:

```
sudo microceph status
```

The output should look somewhat as shown below:

```
user@host:~$

MicroCeph deployment summary:
- ubuntu (10.246.114.49)
 Services: mds, mgr, mon
    Disks: 0
```

Your cluster deployment summary contains your node's hostname (IP address). In our case, it's `ubuntu` (`10.246.114.49`), along with information about the services running and available storage. You'll notice that the cluster is healthy with one node and three services running, but no storage has been allocated yet.

Now that the cluster is initialised, we'll add some storage to the node.

1.5. Add storage

Let's add storage disk devices to the node.

We will use loop files, which are file-backed Object Storage Daemons (OSDs) convenient for setting up small test and development clusters. Three OSDs are required to form a minimal Ceph cluster.

Execute the following command:

```
sudo microceph disk add loop,4G,3
```

```
user@host:~$

+-----------+---------+
|   PATH    | STATUS  |
+-----------+---------+
| loop,4G,3 | Success |
+-----------+---------+
```

Success! You have added three OSDs with 4GiB storage to your node.

Recheck the cluster status:

```
sudo microceph status
```

```
user@host:~$
```

```
MicroCeph deployment summary:
- ubuntu (10.246.114.49)
Services: mds, mgr, mon, osd
Disks: 3
```

You have successfully deployed a Ceph cluster on a single node.

Remember that we had three services running when the cluster was bootstrapped. Note that we now have four services running, including the newly added `osd` service.

1.6. Enable RGW

As mentioned before, we will use the Ceph Object Gateway to interact with the object storage cluster we just deployed.

1.6.1. Enable the RGW daemon on your node

```
sudo microceph enable rgw
```

Note:

By default, the `rgw` service uses port 80, which may not always be available. If port 80 is occupied, you can specify an alternative port, such as 8080, by adding the `--port <port-number>` parameter.

Run the status check again to confirm that the `rgw` service is reflected in the status output.

```
sudo microceph status
```

```
user@host:~$

MicroCeph deployment summary:
- ubuntu (10.246.114.49)
Services: mds, mgr, mon, rgw, osd
Disks: 3
```

1.6.2. Create an RGW user

MicroCeph is packaged with the standard `radosgw-admin` tool that manages the `rgw` service and users. We will now use this tool to create an RGW user called `user`, with the display name `user`.

```
sudo radosgw-admin user create --uid=user --display-name=user
```

The output should include user details as shown below, with auto-generated access and secret keys.

```
user@host:~$
```

```
 {
"user_id": "user",
"display_name": "user",
"email": "",
"suspended": 0,
"max_buckets": 1000,
"subusers": [],
"keys": [
    {
        "user": "user",
        "access_key": "NJ7YZ3LYI45M4Q1A08OS",
        "secret_key": "H7OTclVbZIwhd2o0NLPu0D7Ass8ouSKmtSewuYwK",
        "active": true,
        "create_date": "2024-11-28T13:07:41.561437Z"
    }
],
...
```

1.6.3. Set user secrets

Let's define secrets for this user, setting `access_key` to `foo`, and `--secret-key` to `bar`.

```
sudo radosgw-admin key create --uid=user --key-type=s3 --access-key=foo --secret-key=bar
```

```
user@host:~$

 ...
 [
     {
         "user": "user",
         "access_key": "NJ7YZ3LYI45M4Q1A08OS",
         "secret_key": "H7OTclVbZIwhd2o0NLPu0D7Ass8ouSKmtSewuYwK",
         "active": true,
         "create_date": "2024-11-28T13:07:41.561437Z"
     },
     {
         "user": "user",
         "access_key": "foo",
         "secret_key": "bar",
         "active": true,
         "create_date": "2024-11-28T13:54:36.065214Z"
     }
 ],
...
```

1.7. Consuming the storage

1.7.1. Access RGW

Before attempting to consume the object storage in the cluster, validate that you can access RGW by running **`curl`** on your node.

Find the IP address of the node running the `rgw` service:

```
sudo microceph status
```

```
user@host:~$

MicroCeph deployment summary:
- ubuntu (10.246.114.49)
Services: mds, mgr, mon, rgw, osd
Disks: 3
```

Then, run **`curl`** from this node.

```
curl http://10.246.114.49
```

```
user@host:~$

<?xml version="1.0" encoding="UTF-8"?><ListAllMyBucketsResult xmlns="http://
s3.amazonaws.com/doc/2006-03-01/"><Owner><ID>anonymous</ID></Owner><Buckets>
</Bucket
```

1.7.2. Create an S3 bucket

You have verified that your cluster is accessible via RGW. To interact with S3, we need to make sure that the `s3cmd` utility is installed and configured.

Install and configure `s3cmd`

To install `s3cmd`, run the following command:

```
sudo apt-get install s3cmd
```

To configure the s3cmd tool, create a file named `.s3cfg` in your home directory. This should be an INI-style configuration file with a single `[default]` section and key-value pairs for configuration.

Run the below command to create the file and configure s3cmd:

```
user@host:~$
```

```
cat > ~/.s3cfg <<EOF
[default]
access_key = foo
secret_key = bar
host_base = ubuntu
host_bucket = ubuntu/%(bucket)
check_ssl_certificate = False
check_ssl_hostname = False
use_https = False
EOF
```

Instead of running this command, you can of course also set up the configuration file using your favourite editor.

This configuration will do the following:

- Configure secret and access key that we had set earlier.
- Configure the host to contact. We have named our host `ubuntu`, so this is what we will set here.
- Configure the host bucket template. The host bucket scheme allows users to specify virtual hosting style access or other access modes. For our uses, we will set it to the host name, followed by the bucket name.
- Finally, we did not configure SSL/TLS for our endpoint, so we are disabling it for s3cmd as well.

As a good security practice, it should also be ensured that the `.s3cfg` file is only readable by the user as it does contain the secret key. Run chmod like this:

```
user@host:~$

chmod 0600 ~/.s3cfg
```

Create a bucket

You have verified that your cluster is accessible via RGW. Now, let's create a bucket using the `s3cmd` tool:

```
s3cmd mb -P s3://mybucket
```

Note:

The `-P` flag ensures that the bucket is publicly visible, enabling you to access stored objects easily via a public URL.

```
user@host:~$

Bucket 's3://mybucket/' created
```

Our bucket is successfully created.

1.7.3. Upload an image into the bucket

```
s3cmd put -P image.jpg s3://mybucket
```

```
user@host:~$

upload: 'image.jpg' -> 's3://mybucket/image.jpg'  [1 of 1]
66565 of 66565   100% in    0s     4.52 MB/s  done
Public URL of the object is: http://ubuntu/mybucket/image.jpg
```

The output shows that your image is stored in a publicly accessible S3 bucket. You can now click on the public object URL in the output to view the image in your browser.

1.8. Cleaning up resources

If you want to remove MicroCeph, you can purge the snap from your machine using:

```
sudo snap remove microceph --purge
```

This command stops all running services and removes the MicroCeph snap, along with your cluster and all its contained resources.

Note:

Note: the `--purge` flag will remove all persistent state associated with MicroCeph. The `--purge` flag deletes all files associated with the MicroCeph package, meaning it will remove the MicroCeph snap without saving any data snapshots. Running the command without this flag will not fully remove MicroCeph; the persistent state will remain intact.

Tip:

Skipping the **`purge`** option is useful if you intend to re-install MicroCeph, or move your configuration to a different system.

```
user@host:~$

2024-11-28T19:44:29+03:00 INFO Waiting for "snap.microceph.rgw.service" to
stop.
2024-11-28T19:45:00+03:00 INFO Waiting for "snap.microceph.mds.service" to
stop.
microceph removed
```

1.9. Next steps

You have deployed a healthy Ceph cluster on a single-node and enabled RGW on it. Even better, you have consumed the storage in that cluster by creating a bucket and storing an image object in it. Curious to see what else you can do with MicroCeph?

See our how-to guides, packed with instructions to help you achieve specific goals with MicroCeph.

Or, explore our Explanation and Reference sections for additional information and quick references.