MicroCloud

Canonical Group Ltd

Jul 19, 2024

CONTENTS

MicroCloud allows you to deploy your own fully functional cloud in minutes.

The MicroCloud snap automatically configures LXD, Ceph, and OVN across a set of servers. MicroCloud relies on mDNS to automatically detect other servers on the network, making it possible to set up a complete cluster by running a single command on one of the machines.

This way, MicroCloud creates a small footprint cluster of compute nodes with distributed storage and secure networking, optimised for repeatable, reliable remote deployments.

MicroCloud is aimed at edge computing, and anyone in need of a small-scale private cloud.

CHAPTER

ONE

IN THIS DOCUMENTATION

Tutorial

Start here: a hands-on introduction to MicroCloud for new users

How-to guides

Step-by-step guides covering key operations and common tasks

Reference

Technical information - specifications, APIs, architecture

Explanation

Discussion and clarification of key topics

CHAPTER

TWO

PROJECT AND COMMUNITY

MicroCloud is a member of the Ubuntu family. It's an open source project that warmly welcomes community projects, contributions, suggestions, fixes and constructive feedback.

- MicroCloud snap
- Contribute
- Get support
- Thinking about using MicroCloud for your next project? Get in touch!

2.1 Get started with MicroCloud

MicroCloud is quick to set up. Once *installed*, you can start using MicroCloud in the same way as a regular LXD cluster.

This tutorial guides you through installing MicroCloud in a confined environment and then starting some instances to see what you can do with MicroCloud. It uses virtual machines in LXD, so you don't need any extra hardware to follow the tutorial.

> **Tip**
>
> In this tutorial, we use four virtual machines in LXD for the MicroCloud cluster members. You can use a different number of machines if you want, but the minimum number of required machines is three.
>
> We limit each virtual machine to 2 GiB of RAM, which is less than the recommended hardware requirements. In the context of the tutorial, this amount of RAM is sufficient. However, in a production environment, make sure to use machines that fulfil the *Hardware requirements*.

2.1.1 1. Install and initialise LXD

> **Note**
>
> You can skip this step if you already have a LXD server installed and initialised. However, you should make sure that you have a storage pool set up that is big enough to store four virtual machines. We recommend a storage pool size of at least 40 GiB for that.

Complete the following steps to install and initialise LXD:

1. Install `snapd`:

1. Run **snap version** to find out if snap is installed on your system:

   `user@host:~$ snap version  snap 2.59.4snapd 2.59.4series 16ubuntu 22.04kernel 5.15.0-73-generic` If you see a table of version numbers, snap is installed. If the version for `snapd` is 2.59 or later, you are all set and can continue with the next step of installing LXD.

2. If the version for `snapd` is earlier than 2.59, or if the **snap version** command returns an error, run the following commands to install the latest version of `snapd`:

   ```
   sudo apt update
   sudo apt install snapd
   ```

2. Enter the following command to install LXD:

   ```
   sudo snap install lxd
   ```

3. Enter the following command to initialise LXD:

   ```
   lxd init
   ```

   Accept the default values except for the following questions:

   - `Size in GiB of the new loop device (1GiB minimum)`

     Enter `40GiB`.

   - `Would you like the LXD server to be available over the network? (yes/no)`

     Enter `yes`.

2.1.2 2. Provide storage disks

MicroCloud supports both local and remote storage. For local storage, you need one disk per cluster member. For remote storage, you need at least three disks that are located on different cluster members.

In this tutorial, we'll set up four cluster members, which means that we need a minimum of seven disks (four for local storage and three for remote storage).

Complete the following steps to create the required disks in a LXD storage pool:

1. Create a ZFS storage pool called `disks`:

   ```
   lxc storage create disks zfs size=100GiB
   ```

2. Configure the default volume size for the `disks` pool:

   ```
   lxc storage set disks volume.size 10GiB
   ```

3. Create four disks to use for local storage:

   ```
   lxc storage volume create disks local1 --type block
   lxc storage volume create disks local2 --type block
   lxc storage volume create disks local3 --type block
   lxc storage volume create disks local4 --type block
   ```

4. Create three disks to use for remote storage:

```
lxc storage volume create disks remote1 --type block size=20GiB
lxc storage volume create disks remote2 --type block size=20GiB
lxc storage volume create disks remote3 --type block size=20GiB
```

5. Check that the disks have been created correctly:

```
root@micro1:~# lxc storage volume list disks+--------+---------+-------------+--------------+-----
TYPE | NAME | DESCRIPTION | CONTENT-TYPE | USED BY |+--------+---------+-------------+--------------
custom | local1 | | block | 0 |+--------+---------+-------------+--------------+---------+|
custom | local2 | | block | 0 |+--------+---------+-------------+--------------+---------+|
custom | local3 | | block | 0 |+--------+---------+-------------+--------------+---------+|
custom | local4 | | block | 0 |+--------+---------+-------------+--------------+---------+|
custom | remote1 | | block | 0 |+--------+---------+-------------+--------------+---------+|
custom | remote2 | | block | 0 |+--------+---------+-------------+--------------+---------+|
custom | remote3 | | block | 0 |+--------+---------+-------------+--------------+---------+
```

2.1.3 3. Create a network

MicroCloud requires an uplink network that the cluster members can use for external connectivity. See *Networking* for more information.

Complete the following steps to set up this network:

1. Create a bridge network without any parameters:

```
lxc network create microbr0
```

2. Enter the following commands to find out the assigned IPv4 and IPv6 addresses for the network and note them down:

```
lxc network get microbr0 ipv4.address
lxc network get microbr0 ipv6.address
```

2.1.4 4. Create and configure your VMs

Next, we'll create the VMs that will serve as the MicroCloud cluster members.

Complete the following steps:

1. Create the VMs, but don't start them yet:

```
lxc init ubuntu:22.04 micro1 --vm --config limits.cpu=2 --config limits.memory=2GiB
lxc init ubuntu:22.04 micro2 --vm --config limits.cpu=2 --config limits.memory=2GiB
lxc init ubuntu:22.04 micro3 --vm --config limits.cpu=2 --config limits.memory=2GiB
lxc init ubuntu:22.04 micro4 --vm --config limits.cpu=2 --config limits.memory=2GiB
```

> **Tip**
>
> LXD downloads the image the first time you use it to initialise a VM. Therefore, the **`init`** command will take longer to complete on the first run. For subsequent runs, LXD uses the cached image.
>
> Therefore, you shouldn't run these commands in parallel.

2. Attach the disks to the VMs:

```
lxc storage volume attach disks local1 micro1
lxc storage volume attach disks local2 micro2
lxc storage volume attach disks local3 micro3
lxc storage volume attach disks local4 micro4
lxc storage volume attach disks remote1 micro1
lxc storage volume attach disks remote2 micro2
lxc storage volume attach disks remote3 micro3
```

3. Create and add network interfaces that use the dedicated MicroCloud network to each VM:

```
lxc config device add micro1 eth1 nic network=microbr0 name=eth1
lxc config device add micro2 eth1 nic network=microbr0 name=eth1
lxc config device add micro3 eth1 nic network=microbr0 name=eth1
lxc config device add micro4 eth1 nic network=microbr0 name=eth1
```

4. Start the VMs:

```
lxc start micro1
lxc start micro2
lxc start micro3
lxc start micro4
```

2.1.5 5. Install MicroCloud on each VM

Before you can create the MicroCloud cluster, you must install the required snaps on each VM. In addition, you must configure the network interfaces so they can be used by MicroCloud.

Complete the following steps on each VM (`micro1`, `micro2`, `micro3`, and `micro4`):

1. Access the shell in the VM. For example, for `micro1`:

```
lxc exec micro1 -- bash
```

> **Tip**
>
> If you get an error message stating that the LXD VM agent is not currently running, the VM hasn't fully started up yet. Wait a while and then try again. If the error persists, try restarting the VM (`lxc restart micro1`).

2. Configure the network interface connected to `microbr0` to not accept any IP addresses (because MicroCloud requires a network interface that doesn't have an IP address assigned):

```
echo 0 > /proc/sys/net/ipv6/conf/enp6s0/accept_ra
```

> **Note**
>
> `enp6s0` is the name that the VM assigns to the network interface that we previously added as `eth1`.

3. Bring the network interface up:

```
ip link set enp6s0 up
```

4. Install the required snaps:

```
snap install microceph --channel=quincy/stable --cohort="+"
snap install microovn --channel=22.03/stable --cohort="+"
snap install microcloud --channel=latest/stable --cohort="+"
```

> **Note**
>
> The `--cohort="+"` flag in the command ensures that the same version of the snap is installed on all machines. See *Keep cluster members in sync* for more information.

5. The LXD snap is already installed. Refresh it to the latest version:

```
snap refresh lxd --channel=5.21/stable --cohort="+"
```

2.1.6 6. Initialise MicroCloud

After installing all snaps on all VMs, you can initialise MicroCloud. This initialisation is done on one of the machines only. We use `micro1`, but you can choose another machine.

Complete the following steps:

1. Access the shell in `micro1`:

```
lxc exec micro1 -- bash
```

2. Start the initialisation process:

```
microcloud init
```

> **Tip**
>
> In this tutorial, we initialise MicroCloud interactively. Alternatively, you can use a preseed file for *Non-interactive configuration*.

3. Answer the questions:
    1. As the address for MicroCloud's internal traffic, select the listed IPv4 address.
    2. Select `yes` to limit the search for other MicroCloud servers to the local subnet.
    3. Select all listed servers (these should be `micro2`, `micro3`, and `micro4`).
    4. Select `yes` to set up local storage.
    5. Select the listed local disks (`local1`, `local2`, `local3`, and `local4`).

> **Tip**
>
> Type `local` to display only the local disks. The table is filtered by the characters that you type.

6. You don't need to wipe any disks (because we just created them).
7. Select `yes` to set up distributed storage.
8. Select `yes` to confirm that there are fewer disks available than machines.
9. Select all listed disks (these should be `remote1`, `remote2`, and `remote3`).
10. You don't need to wipe any disks (because we just created them).
11. You don't need to encrypt any disks to get started.
12. Select `yes` to optionally configure the CephFS distributed file system.
13. Leave the question empty for the IPv4 or IPv6 CIDR subnet address used for the Ceph internal network.
14. Select `yes` to configure distributed networking.
15. Select all listed network interfaces (these should be `enp6s0` on the four different VMs).
16. Specify the IPv4 address that you noted down for your `microbr0` network as the IPv4 gateway.
17. Specify an IPv4 address in the address range as the first IPv4 address. For example, if your IPv4 gateway is `192.0.2.1/24`, the first address could be `192.0.2.100`.
18. Specify a higher IPv4 address in the range as the last IPv4 address. As we're setting up four machines only, the range must contain a minimum of four addresses, but setting up a bigger range is more fail-safe. For example, if your IPv4 gateway is `192.0.2.1/24`, the last address could be `192.0.2.254`.
19. Specify the IPv6 address that you noted down for your `microbr0` network as the IPv6 gateway.

MicroCloud will now initialise the cluster. See *About the initialisation process* for more information.

See the full initialisation process here:

```
root@micro1:~# microcloud init  Select an address for MicroCloud's internal traffic:Space
to select; enter to confirm; type to filter results.Up/down to move; right to
select all; left to select none. +----------------------+--------+ | ADDRESS |
IFACE | +----------------------+--------+> [X] | 203.0.113.169 | enp5s0 | [ ] |
2001:db8:d:100::169 | enp5s0 | +----------------------+--------+  Using address
"203.0.113.169" for MicroCloud Limit search for other MicroCloud servers to 203.
0.113.169/24? (yes/no) [default=yes]:  yesScanning for eligible servers ...Space
to select; enter to confirm; type to filter results.Up/down to move; right to
select all; left to select none. +---------+--------+---------------+ | NAME |
IFACE | ADDR | +---------+--------+---------------+> [x] | micro3 | enp5s0 | 203.
0.113.171 | [x] | micro2 | enp5s0 | 203.0.113.170 | [x] | micro4 | enp5s0 | 203.
0.113.172 | +---------+--------+---------------+  Selected "micro3" at "203.0.
113.171" Selected "micro2" at "203.0.113.170" Selected "micro4" at "203.0.113.
172" Would you like to set up local storage? (yes/no) [default=yes]:  yesSelect
exactly one disk from each cluster member:Space to select; enter to confirm; type
to filter results.Up/down to move; right to select all; left to select none.
+----------+--------------+----------+------+-------------------------------------------------------+
| LOCATION | MODEL | CAPACITY | TYPE | PATH | +----------+---------------+----------+------+-----------
[x] | micro1 | QEMU HARDDISK | 10.00GiB | scsi | /dev/disk/by-id/
scsi-0QEMU_QEMU_HARDDISK_lxd_local1 | [ ] | micro1 | QEMU HARDDISK | 20.00GiB | scsi |
/dev/disk/by-id/scsi-0QEMU_QEMU_HARDDISK_lxd_remote1 | [x] | micro2 | QEMU HARDDISK |
10.00GiB | scsi | /dev/disk/by-id/scsi-0QEMU_QEMU_HARDDISK_lxd_local2 | [ ] | micro2 |
QEMU HARDDISK | 20.00GiB | scsi | /dev/disk/by-id/scsi-0QEMU_QEMU_HARDDISK_lxd_remote2
| [x] | micro3 | QEMU HARDDISK | 10.00GiB | scsi | /dev/disk/by-id/
scsi-0QEMU_QEMU_HARDDISK_lxd_local3 | [ ] | micro3 | QEMU HARDDISK | 20.00GiB |
scsi | /dev/disk/by-id/scsi-0QEMU_QEMU_HARDDISK_lxd_remote3 |> [x] | micro4 | QEMU
HARDDISK | 10.00GiB | scsi | /dev/disk/by-id/scsi-0QEMU_QEMU_HARDDISK_lxd_local4 |
```

```
+----------+---------------+----------+------+-------------------------------------------------------+
Select which disks to wipe:Space to select; enter to confirm; type to
filter results.Up/down to move; right to select all; left to select none.
+----------+---------------+----------+------+-------------------------------------------------------+
| LOCATION | MODEL | CAPACITY | TYPE | PATH | +----------+---------------+----------+------+-----------
[ ] | micro1 | QEMU HARDDISK | 10.00GiB | scsi | /dev/disk/by-id/
scsi-0QEMU_QEMU_HARDDISK_lxd_local1 | [ ] | micro2 | QEMU HARDDISK | 10.00GiB | scsi |
/dev/disk/by-id/scsi-0QEMU_QEMU_HARDDISK_lxd_local2 | [ ] | micro3 | QEMU HARDDISK |
10.00GiB | scsi | /dev/disk/by-id/scsi-0QEMU_QEMU_HARDDISK_lxd_local3 | [ ] | micro4 |
QEMU HARDDISK | 10.00GiB | scsi | /dev/disk/by-id/scsi-0QEMU_QEMU_HARDDISK_lxd_local4 |
+----------+---------------+----------+------+-------------------------------------------------------+
Using "/dev/disk/by-id/scsi-0QEMU_QEMU_HARDDISK_lxd_local3" on "micro3" for local storage
pool Using "/dev/disk/by-id/scsi-0QEMU_QEMU_HARDDISK_lxd_local4" on "micro4" for local
storage pool Using "/dev/disk/by-id/scsi-0QEMU_QEMU_HARDDISK_lxd_local1" on "micro1"
for local storage pool Using "/dev/disk/by-id/scsi-0QEMU_QEMU_HARDDISK_lxd_local2"
on "micro2" for local storage pool Would you like to set up distributed
storage? (yes/no) [default=yes]:  yesUnable to find disks on some systems.
Continue anyway? (yes/no) [default=yes]:  yesSelect from the available
unpartitioned disks:Space to select; enter to confirm; type to filter
results.Up/down to move; right to select all; left to select none.
+----------+---------------+----------+------+-------------------------------------------------------+
| LOCATION | MODEL | CAPACITY | TYPE | PATH | +----------+---------------+----------+------+-----------
[x] | micro1 | QEMU HARDDISK | 20.00GiB | scsi | /dev/disk/by-id/
scsi-0QEMU_QEMU_HARDDISK_lxd_remote1 | [x] | micro2 | QEMU HARDDISK | 20.00GiB |
scsi | /dev/disk/by-id/scsi-0QEMU_QEMU_HARDDISK_lxd_remote2 | [x] | micro3 | QEMU
HARDDISK | 20.00GiB | scsi | /dev/disk/by-id/scsi-0QEMU_QEMU_HARDDISK_lxd_remote3 |
+----------+---------------+----------+------+-------------------------------------------------------+
Select which disks to wipe:Space to select; enter to confirm; type to
filter results.Up/down to move; right to select all; left to select none.
+----------+---------------+----------+------+-------------------------------------------------------+
| LOCATION | MODEL | CAPACITY | TYPE | PATH | +----------+---------------+----------+------+-----------
[ ] | micro1 | QEMU HARDDISK | 20.00GiB | scsi | /dev/disk/by-id/
scsi-0QEMU_QEMU_HARDDISK_lxd_remote1 | [ ] | micro2 | QEMU HARDDISK | 20.00GiB |
scsi | /dev/disk/by-id/scsi-0QEMU_QEMU_HARDDISK_lxd_remote2 | [ ] | micro3 | QEMU
HARDDISK | 20.00GiB | scsi | /dev/disk/by-id/scsi-0QEMU_QEMU_HARDDISK_lxd_remote3 |
+----------+---------------+----------+------+-------------------------------------------------------+
Using 1 disk(s) on "micro1" for remote storage pool Using 1 disk(s) on "micro2"
for remote storage pool Using 1 disk(s) on "micro3" for remote storage pool Do you
want to encrypt the selected disks? (yes/no) [default=no]:  noWould you like to
set up CephFS remote storage? (yes/no) [default=yes]:  yesConfigure distributed
networking? (yes/no) [default=yes]:  yesSelect an available interface per system
to provide external connectivity for distributed network(s):Space to select;
enter to confirm; type to filter results.Up/down to move; right to select all;
left to select none. +----------+--------+----------+ | LOCATION | IFACE | TYPE |
+----------+--------+----------+> [x] | micro2 | enp6s0 | physical | [x] | micro3 |
enp6s0 | physical | [x] | micro1 | enp6s0 | physical | [x] | micro4 | enp6s0 | physical
| +----------+--------+----------+  Using "enp6s0" on "micro3" for OVN uplink Using
"enp6s0" on "micro1" for OVN uplink Using "enp6s0" on "micro2" for OVN uplink Using
"enp6s0" on "micro4" for OVN uplink Specify the IPv4 gateway (CIDR) on the uplink network
(empty to skip IPv4):  192.0.2.1/24Specify the first IPv4 address in the range to use on
the uplink network:  192.0.2.100Specify the last IPv4 address in the range to use on the
uplink network:  192.0.2.254Specify the IPv6 gateway (CIDR) on the uplink network (empty
to skip IPv6):  2001:db8:d:200::1/64Specify the DNS addresses (comma-separated IPv4 /
IPv6 addresses) for the distributed network (default:  192.0.2.1,2001:db8:d:200::1):
```

```
Initializing a new cluster Local MicroCloud is ready Local LXD is ready Local MicroOVN is
ready Local MicroCeph is readyAwaiting cluster formation ... Peer "micro2" has joined the
cluster Peer "micro3" has joined the cluster Peer "micro4" has joined the clusterCluster
initialization is completeMicroCloud is ready
```

2.1.7 7. Inspect your MicroCloud setup

You can now inspect your cluster setup.

> **Tip**
>
> You can run these commands on any of the cluster members. We continue using `micro1`, but you will see the same results on the other VMs.

1. Inspect the cluster setup:

```
root@micro1:~# lxc cluster list+--------+----------------------------+------------------+---------
NAME | URL | ROLES | ARCHITECTURE | FAILURE DOMAIN | DESCRIPTION | STATE | MESSAGE
|+--------+----------------------------+------------------+-------------+----------------+--------
micro1 | https://203.0.113.169:8443 | database-leader | x86_64 |
default | | ONLINE | Fully operational || | | database | | | | |
|+--------+----------------------------+------------------+-------------+----------------+--------
micro2 | https://203.0.113.170:8443 | database | x86_64 | default | | ONLINE | Fully
operational |+--------+----------------------------+------------------+-------------+-------------
micro3 | https://203.0.113.171:8443 | database | x86_64 | default | | ONLINE | Fully
operational |+--------+----------------------------+------------------+-------------+-------------
micro4 | https://203.0.113.172:8443 | database-standby | x86_64 | default | | ONLINE
| Fully operational |+--------+----------------------------+------------------+-------------+-----
root@micro1:~# microcloud cluster list+--------+--------------------+----------+-----------------
NAME | ADDRESS | ROLE | FINGERPRINT | STATUS |+--------+--------------------+----------+-----------
micro1 | 203.0.113.169:9443 | voter | 47a74cb2ed8b844544ce71f45e96acb2c8021d4c1ffc2f1f449cdbf2f6898
| ONLINE |+--------+--------------------+----------+-----------------------------------------------
micro2 | 203.0.113.170:9443 | voter | 56bee3adbd5e1de2186dd22788baffd5e1358e408ec3d9b713ed930741a33
| ONLINE |+--------+--------------------+----------+-----------------------------------------------
micro3 | 203.0.113.171:9443 | voter | aabdd5f64d4c2796a50d6ce9d91939f248bfeb27195426158dff05d660f93
| ONLINE |+--------+--------------------+----------+-----------------------------------------------
micro4 | 203.0.113.172:9443 | stand-by | 649ec21815135104f1faa5fca099daddf995f554119c6e34706a2b3168
| ONLINE |+--------+--------------------+----------+-----------------------------------------------
root@micro1:~# microceph cluster list+--------+--------------------+----------+-----------------
NAME | ADDRESS | ROLE | FINGERPRINT | STATUS |+--------+--------------------+----------+-----------
micro1 | 203.0.113.169:7443 | voter | a2b370cce1deb02437b583aa73be5e5c519aed75f02f4b98f6df150fd62c0
| ONLINE |+--------+--------------------+----------+-----------------------------------------------
micro2 | 203.0.113.170:7443 | voter | e37ea1acd14b984152cac4cb861cbe35ac438151233b9d0ee606c44c2e27c
| ONLINE |+--------+--------------------+----------+-----------------------------------------------
micro3 | 203.0.113.171:7443 | voter | 152ccf372ecc93faffa8a6801cedd5eca49d977eea72e3f2239245cc22965
| ONLINE |+--------+--------------------+----------+-----------------------------------------------
micro4 | 203.0.113.172:7443 | stand-by | 9b75b396f6d59481b8c14221942d775cff4d27c5621b0b541eb5ba3245
| ONLINE |+--------+--------------------+----------+-----------------------------------------------
root@micro1:~# microovn cluster list+--------+--------------------+----------+-----------------
NAME | ADDRESS | ROLE | FINGERPRINT | STATUS |+--------+--------------------+----------+-----------
micro1 | 203.0.113.169:6443 | voter | a552d316c159a50a4e11253c36a1cd25a3902bee50e24ed1e073ee7728be0
| ONLINE |+--------+--------------------+----------+-----------------------------------------------
micro2 | 203.0.113.170:6443 | voter | 2c779eb10409576a33fa01a29cede39abea61f7cd6a07837c369858b515ec
```

```
| ONLINE |+--------+--------------------+----------+-------------------------------------------
micro3 | 203.0.113.171:6443 | voter | 7f76cddfdbbe3d768c343b1a5f402842565c25d0e4e3ebbc8514263fc14ea
| ONLINE |+--------+--------------------+----------+-------------------------------------------
micro4 | 203.0.113.172:6443 | stand-by | 5d62b2a63dec514c45c07b24ff93e2bd83ad8b9af4ab774aad3d2ac51e
| ONLINE |+--------+--------------------+----------+-------------------------------------------
```

2. Inspect the storage setup:

```
root@micro1:~# lxc storage list+-----------+--------+-------------------------------------------+
NAME | DRIVER | DESCRIPTION | USED BY | STATE |+-----------+--------+---------------------------
local | zfs | Local storage on ZFS | 8 | CREATED |+-----------+--------+---------------------------
remote | ceph | Distributed storage on Ceph | 1 | CREATED |+-----------+--------+-----------------
remote-fs | cephfs | Distributed file-system storage using Ceph | 1 | CREATED
|+-----------+--------+-------------------------------------------+---------+---------+
root@micro1:~# lxc storage info local      info:  description:  Local storage on ZFS
driver:  zfs name:  local space used:  747.00KiB total space:  9.20GiBused by:
volumes:  - backups (location "micro1") - backups (location "micro2") - backups
(location "micro3") - backups (location "micro4") - images (location "micro1")
- images (location "micro2") - images (location "micro3") - images (location
"micro4")   root@micro1:~# lxc storage info remote   info:  description:  Distributed
storage on Ceph driver:  ceph name:  remote space used:  25.41KiB total space:  29.
67GiBused by:  profiles:  - default  root@micro1:~# lxc storage info remote-fs  info:
description:  Distributed file-system storage using CephFS driver:  cephfs name:
remote-fs space used:  0B total space:  29.67GiBused by:  {}
```

3. Inspect the OVN network setup:

```
root@micro1:~# lxc network list+---------+----------+---------+-----------------+-----------------
NAME | TYPE | MANAGED | IPV4 | IPV6 | DESCRIPTION | USED BY | STATE
|+---------+----------+---------+-----------------+-------------------------+-------------+-------
UPLINK | physical | YES | | | | 1 | CREATED |+---------+----------+---------+-----------------+----
br-int | bridge | NO | | | | 0 | |+---------+----------+---------+-----------------+--------------
default | ovn | YES | 198.51.100.1/24 | 2001:db8:d960:91cf::1/64 | | 1 | CREATED
|+---------+----------+---------+-----------------+-------------------------+-------------+-------
enp5s0 | physical | NO | | | | 0 | |+---------+----------+---------+-----------------+------------
enp6s0 | physical | NO | | | | 1 | |+---------+----------+---------+-----------------+------------
lxdovn1 | bridge | NO | | | | 0 | |+---------+----------+---------+-----------------+--------------
root@micro1:~# lxc network show default            config:  bridge.mtu:  "1442" ipv4.
address:  198.51.100.1/24 ipv4.nat:  "true" ipv6.address:  2001:db8:d960:91cf::1/
64 ipv6.nat:  "true" network:  UPLINK volatile.network.ipv4.address:  192.0.2.100
volatile.network.ipv6.address:  2001:db8:e647:610d:216:3eff:fe96:ed5cdescription:
""name:  defaulttype:  ovnused_by:- /1.0/profiles/defaultmanaged:  truestatus:
Createdlocations:- micro1- micro3- micro2- micro4
```

4. Make sure that you can ping the virtual router within OVN. You can find the IPv4 and IPv6 addresses of the virtual router under `volatile.network.ipv4.address` and `volatile.network.ipv6.address`, respectively, in the output of `lxc network show default`.

```
root@micro1:~# ping 192.0.2.100        PING 192.0.2.100 (192.0.2.100) 56(84) bytes of
data.64 bytes from 192.0.2.100:  icmp_seq=1 ttl=253 time=2.05 ms64 bytes from 192.
0.2.100:  icmp_seq=2 ttl=253 time=2.01 ms64 bytes from 192.0.2.100:  icmp_seq=3
ttl=253 time=1.78 ms^C--- 192.0.2.100 ping statistics ---4 packets transmitted,
3 received, 25% packet loss, time 3005msrtt min/avg/max/mdev = 1.777/1.945/2.
052/0.120 ms    root@micro1:~# ping6 -n 2001:db8:e647:610d:216:3eff:fe96:ed5c    PING
2001:db8:e647:610d:216:3eff:fe96:ed5c(2001:db8:e647:610d:216:3eff:fe96:ed5c) 56
data bytes64 bytes from 2001:db8:e647:610d:216:3eff:fe96:ed5c:  icmp_seq=1 ttl=253
time=1.61 ms64 bytes from 2001:db8:e647:610d:216:3eff:fe96:ed5c:  icmp_seq=2 ttl=253
```

```
time=1.99 ms64 bytes from 2001:db8:e647:610d:216:3eff:fe96:ed5c:  icmp_seq=3 ttl=253
time=15.7 ms^C--- 2001:db8:e647:610d:216:3eff:fe96:ed5c ping statistics ---3 packets
transmitted, 3 received, 0% packet loss, time 2004msrtt min/avg/max/mdev = 1.606/6.
432/15.704/6.558 ms
```

5. Inspect the default profile:

```
root@micro1:~# lxc profile show default    config:  {}description:  ""devices:  eth0:
name:  eth0 network:  default type:  nic root:  path:  /  pool:  remote type:
diskname:  defaultused_by:  []
```

2.1.8 8. Launch some instances

Now that your MicroCloud cluster is ready to use, let's launch a few instances:

1. Launch an Ubuntu container with the default settings:

```
lxc launch ubuntu:22.04 u1
```

2. Launch another Ubuntu container, but use the local storage instead of the remote storage that is the default:

```
lxc launch ubuntu:22.04 u2 --storage local
```

3. Launch an Ubuntu VM:

```
lxc launch ubuntu:22.04 u3 --vm
```

4. Check the list of instances. You will see that the instances are running on different cluster members.

```
root@micro1:~# lxc list+------+---------+---------------------+----------------------------------
NAME | STATE | IPV4 | IPV6 | TYPE | SNAPSHOTS | LOCATION |+------+---------+---------------------+-
u1 | RUNNING | 198.51.100.2 (eth0) | 2001:db8:d960:91cf:216:3eff:fe4e:9642 (eth0) |
CONTAINER | 0 | micro1 |+------+---------+---------------------+----------------------------------
u2 | RUNNING | 198.51.100.3 (eth0) | 2001:db8:d960:91cf:216:3eff:fe79:6765 (eth0) |
CONTAINER | 0 | micro3 |+------+---------+---------------------+----------------------------------
u3 | RUNNING | 198.51.100.4 (eth0) | 2001:db8:d960:91cf:216:3eff:fe66:f24b (eth0) |
VIRTUAL-MACHINE | 0 | micro2 |+------+---------+---------------------+----------------------------
```

5. Check the storage. You will see that the instance volumes are located on the specified storage pools.

```
root@micro1:~# lxc storage volume list remote+-----------------+----------------------------------
TYPE | NAME | DESCRIPTION | CONTENT-TYPE | USED BY | LOCATION
|+-----------------+-------------------------------------------------------------------+-----------
container | u1 | | filesystem | 1 | |+-----------------+-----------------------------------------
image | 17fbc145125c659b7ef926b2de5e5304370083e28846f084a0d514c7a96777bc | | block |
1 | |+-----------------+-------------------------------------------------------------------+--------
image | 45613e262f8a5fc9467330f679862147c289516f045e3edc313e07ebcb0aab4a | |
filesystem | 1 | |+-----------------+-----------------------------------------------------------------
virtual-machine | u3 | | block | 1 | |+-----------------+-----------------------------------------
root@micro1:~# lxc storage volume list local+-----------+-------------------------------------
TYPE | NAME | DESCRIPTION | CONTENT-TYPE | USED BY | LOCATION
|+-----------+-------------------------------------------------------------------+------------+----
container | u2 | | filesystem | 1 | micro3 |+-----------+-----------------------------------------
custom | backups | | filesystem | 1 | micro2 |+-----------+-----------------------------------------
custom | backups | | filesystem | 1 | micro3 |+-----------+-----------------------------------------
custom | backups | | filesystem | 1 | micro4 |+-----------+-----------------------------------------
custom | backups | | filesystem | 1 | micro1 |+-----------+-----------------------------------------
```

```
custom | images | | filesystem | 1 | micro2 |+-----------+----------------------------------------
custom | images | | filesystem | 1 | micro3 |+-----------+----------------------------------------
custom | images | | filesystem | 1 | micro4 |+-----------+----------------------------------------
custom | images | | filesystem | 1 | micro1 |+-----------+----------------------------------------
image | 45613e262f8a5fc9467330f679862147c289516f045e3edc313e07ebcb0aab4a | |
filesystem | 1 | micro3 |+-----------+-------------------------------------------------------------------
```

2.1.9 9. Inspect your networking

The instances that you have launched are all on the same subnet. You can, however, create a different network to isolate some instances from others.

1. Check the list of instances that are running:

```
root@micro1:~# lxc list+------+---------+---------------------+----------------------------------
NAME | STATE | IPV4 | IPV6 | TYPE | SNAPSHOTS | LOCATION |+------+---------+---------------------+-
u1 | RUNNING | 198.51.100.2 (eth0) | 2001:db8:d960:91cf:216:3eff:fe4e:9642 (eth0) |
CONTAINER | 0 | micro1 |+------+---------+---------------------+----------------------------------
u2 | RUNNING | 198.51.100.3 (eth0) | 2001:db8:d960:91cf:216:3eff:fe79:6765 (eth0) |
CONTAINER | 0 | micro3 |+------+---------+---------------------+----------------------------------
u3 | RUNNING | 198.51.100.4 (eth0) | 2001:db8:d960:91cf:216:3eff:fe66:f24b (eth0) |
VIRTUAL-MACHINE | 0 | micro2 |+------+---------+---------------------+----------------------------
```

2. Access the shell in `u1`:

```
lxc exec u1 -- bash
```

3. Ping the IPv4 address of `u2`:

```
root@u1:~# ping 198.51.100.3 PING 198.51.100.3 (198.51.100.3) 56(84) bytes of data.64
bytes from 198.51.100.3:  icmp_seq=1 ttl=64 time=1.33 ms64 bytes from 198.51.100.3:
icmp_seq=2 ttl=64 time=1.74 ms64 bytes from 198.51.100.3:  icmp_seq=3 ttl=64 time=0.
985 ms^C--- 198.51.100.3 ping statistics ---3 packets transmitted, 3 received, 0%
packet loss, time 2004msrtt min/avg/max/mdev = 0.985/1.352/1.739/0.308 ms
```

4. Ping the IPv6 address of `u3`:

```
root@u1:~# ping6 -n 2001:db8:d960:91cf:216:3eff:fe66:f24b                      PING
2001:db8:d960:91cf:216:3eff:fe66:f24b(2001:db8:d960:91cf:216:3eff:fe66:f24b) 56
data bytes64 bytes from 2001:db8:d960:91cf:216:3eff:fe66:f24b:  icmp_seq=1 ttl=64
time=16.8 ms64 bytes from 2001:db8:d960:91cf:216:3eff:fe66:f24b:  icmp_seq=2 ttl=64
time=3.41 ms64 bytes from 2001:db8:d960:91cf:216:3eff:fe66:f24b:  icmp_seq=3 ttl=64
time=3.86 ms^C--- 2001:db8:d960:91cf:216:3eff:fe66:f24b ping statistics ---3 packets
transmitted, 3 received, 0% packet loss, time 2004msrtt min/avg/max/mdev = 3.407/8.
012/16.774/6.197 ms
```

5. Confirm that the instance has connectivity to the outside world:

```
root@u1:~# ping www.example.com     PING www.example.com (93.184.216.34) 56(84) bytes
of data.64 bytes from 93.184.216.34 (93.184.216.34):  icmp_seq=1 ttl=49 time=111
ms64 bytes from 93.184.216.34 (93.184.216.34):  icmp_seq=2 ttl=49 time=95.2 ms64
bytes from 93.184.216.34 (93.184.216.34):  icmp_seq=3 ttl=49 time=96.2 ms^C--- www.
example.com ping statistics ---3 packets transmitted, 3 received, 0% packet loss,
time 2018msrtt min/avg/max/mdev = 95.233/100.870/111.165/7.290 ms
```

6. Log out of the `u1` shell:

```
exit
```

7. Create an OVN network with the default settings:

```
lxc network create isolated --type=ovn
```

   There is only one UPLINK network, so the new network will use this one as its parent.

8. Show information about the new network:

```
root@micro1:~# lxc network show isolated  config:  bridge.mtu:  "1442" ipv4.address:
198.51.100.201/24 ipv4.nat:  "true" ipv6.address:  2001:db8:452a:32b2::1/64 ipv6.
nat:  "true" network:  UPLINK volatile.network.ipv4.address:  192.0.2.101 volatile.
network.ipv6.address:  2001:db8:e647:610d:216:3eff:feef:6361description:  ""name:
isolatedtype:  ovnused_by:  []managed:  truestatus:  Createdlocations:- micro1-
micro3- micro2- micro4
```

9. Check that you can ping the volatile.network.ipv4.address:

```
root@micro1:~# ping 192.0.2.101  PING 192.0.2.101 (192.0.2.101) 56(84) bytes of data.
64 bytes from 192.0.2.101:  icmp_seq=1 ttl=253 time=1.25 ms64 bytes from 192.0.2.
101:  icmp_seq=2 ttl=253 time=1.04 ms64 bytes from 192.0.2.101:  icmp_seq=3 ttl=253
time=1.68 ms^C--- 192.0.2.101 ping statistics ---3 packets transmitted, 3 received,
0% packet loss, time 2002msrtt min/avg/max/mdev = 1.042/1.321/1.676/0.264 ms
```

10. Launch an Ubuntu container that uses the new network:

```
lxc launch ubuntu:22.04 u4 --network isolated
```

11. Access the shell in u4:

```
lxc exec u4 -- bash
```

12. Confirm that the instance has connectivity to the outside world:

```
root@u4:~# ping www.example.com     PING www.example.com (93.184.216.34) 56(84) bytes
of data.64 bytes from 93.184.216.34 (93.184.216.34):  icmp_seq=1 ttl=49 time=95.
6 ms64 bytes from 93.184.216.34 (93.184.216.34):  icmp_seq=2 ttl=49 time=118 ms64
bytes from 93.184.216.34 (93.184.216.34):  icmp_seq=3 ttl=49 time=94.6 ms^C--- www.
example.com ping statistics ---3 packets transmitted, 3 received, 0% packet loss,
time 2004msrtt min/avg/max/mdev = 94.573/102.587/117.633/10.646 ms
```

13. Ping the IPv4 address of u2:

```
root@u4:~# ping 198.51.100.3   PING 198.51.100.3 (198.51.100.3) 56(84) bytes of data.
^C--- 198.51.100.3 ping statistics ---14 packets transmitted, 0 received, 100%
packet loss, time 13301ms
```

   You will see that u2 is not reachable, because it is on a different OVN subnet.

2.1.10 10. Access the UI

Instead of managing your instances and your LXD setup from the command line, you can also use the LXD UI. See How to access the LXD web UI for more information.

1. Check the LXD cluster list to determine the IP addresses of the cluster members:

```
root@micro1:~# lxc cluster list+--------+----------------------------+------------------+--------
NAME | URL | ROLES | ARCHITECTURE | FAILURE DOMAIN | DESCRIPTION | STATE | MESSAGE
|+--------+----------------------------+----------------+-------------+---------------+-------
micro1 | https://203.0.113.169:8443 | database-leader | x86_64 |
default | | ONLINE | Fully operational || | | database | | | | |
|+--------+----------------------------+----------------+-------------+---------------+-------
micro2 | https://203.0.113.170:8443 | database | x86_64 | default | | ONLINE | Fully
operational |+--------+----------------------------+----------------+-------------+-------------
micro3 | https://203.0.113.171:8443 | database | x86_64 | default | | ONLINE | Fully
operational |+--------+----------------------------+----------------+-------------+-------------
micro4 | https://203.0.113.172:8443 | database-standby | x86_64 | default | | ONLINE
| Fully operational |+--------+----------------------------+----------------+-------------+-----
```

2. In your web browser, navigate to the URL of one of the machines. For example, for `micro1`, navigate to `https://203.0.113.169:8443`.
3. By default, MicroCloud uses a self-signed certificate, which will cause a security warning in your browser. Use your browser's mechanism to continue despite the security warning.
4. You should now see the LXD UI, prompting you to set up a certificate.

> **Note**
>
> Since LXD 5.21, the LXD UI is enabled by default.
>
> If you don't see the certificate screen, you might have an older version of LXD (run `snap info lxd` to check). In this case, run the following commands on the machine that you're trying to access (for example, `micro1`) to enable the UI:
>
> ```
> snap set lxd ui.enable=true
> systemctl reload snap.lxd.daemon
> ```

5. Follow the instructions in the UI to set up the certificates.

> **Tip**
>
> If you create a new certificate, you must transfer it to one of the cluster members to add it to the trust store.
>
> To do this, use the file push command. For example:
>
> ```
> lxc file push lxd-ui.crt micro1/root/lxd-ui.crt
> ```
>
> You can then access the shell on that cluster member and add the certificate to the trust store:
>
> ```
> lxc exec micro1 -- bash
> lxc config trust add lxd-ui.crt
> ```

6. You can now browse the UI and inspect, for example, the instances you created and the networks and storage that MicroCloud set up.

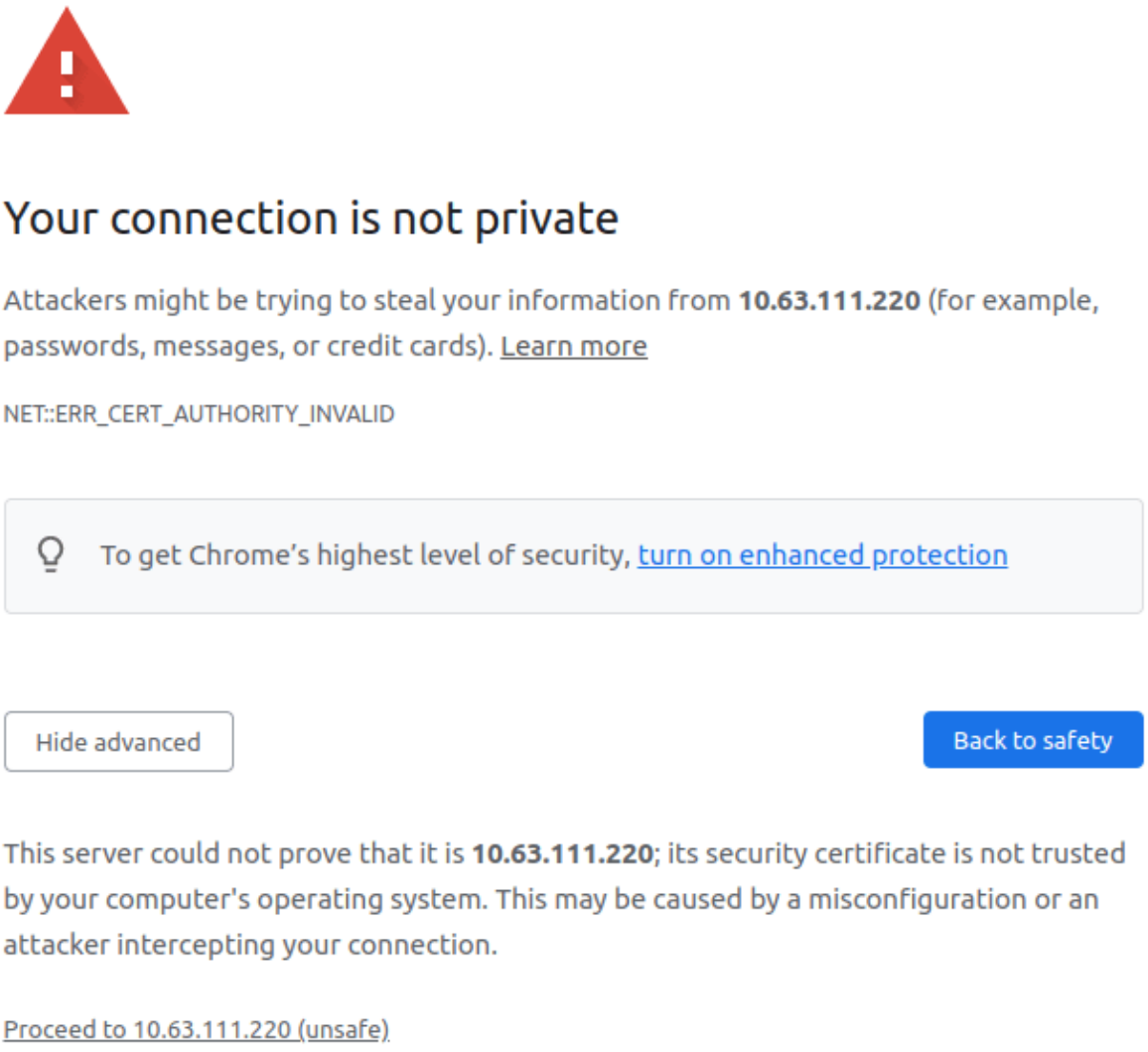

Fig. 1: Example for a security warning in Chrome

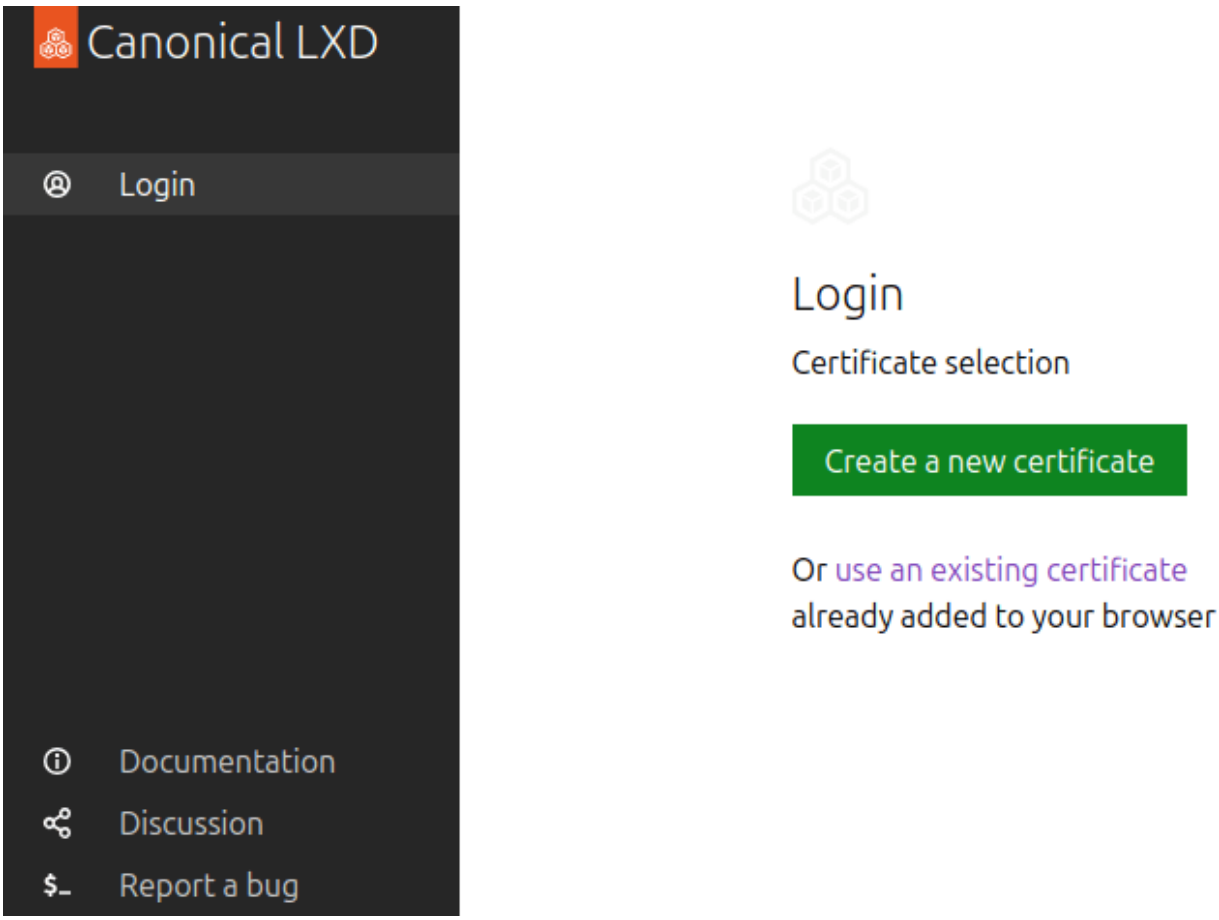

Fig. 2: Certificate selection in the LXD UI

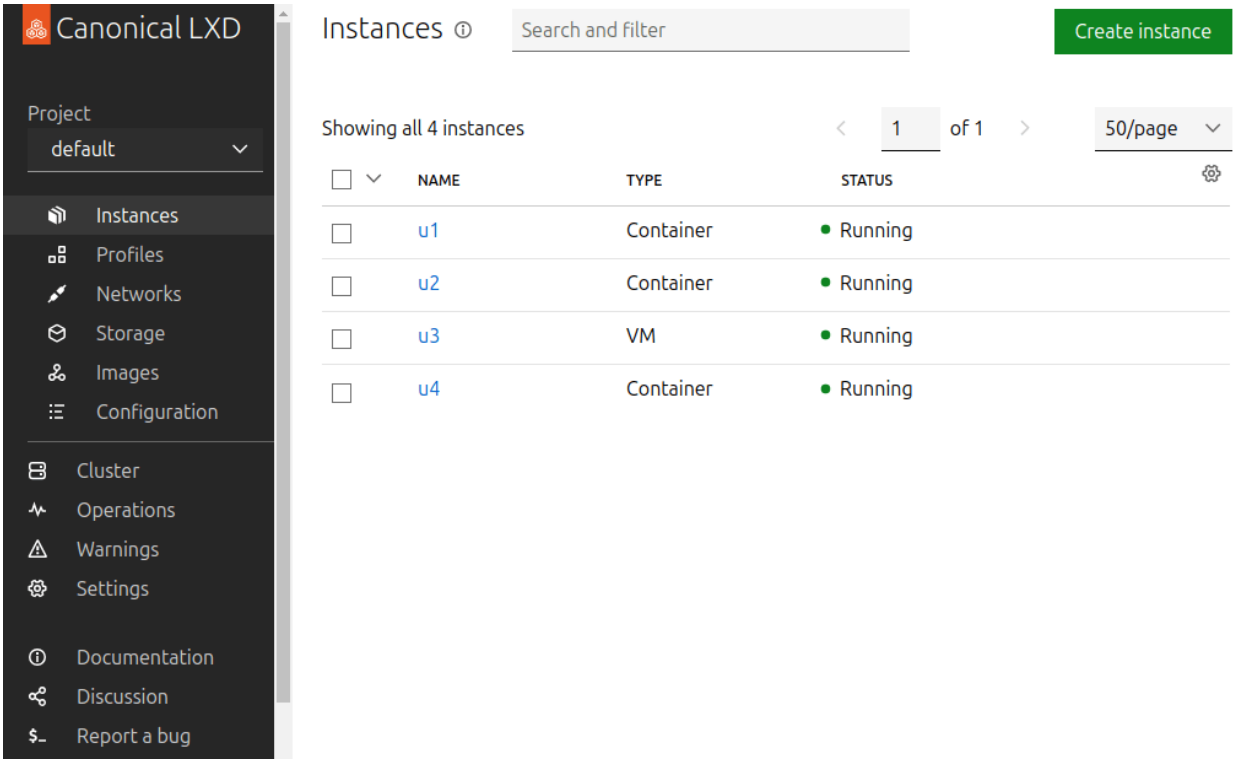

Fig. 3: Instances view in the LXD UI

2.1.11 Next steps

Now that your MicroCloud is up and running, you can start using it!

If you're already familiar with LXD, see *How to work with MicroCloud (command cheat sheet)* for a reference of the most common commands.

If you're new to LXD, check out the First steps with LXD tutorial to familiarise yourself with what you can do in LXD. You can skip the first section about installing and initialising LXD, because LXD is already operational as part of your MicroCloud setup.

2.2 How-to guides

These how-to guides cover key operations and processes in MicroCloud.

2.2.1 How to install MicroCloud

To install MicroCloud, install all required *Snaps* on all machines that you want to include in your cluster.

To do so, enter the following commands on all machines:

```
sudo snap install lxd --channel=5.21/stable --cohort="+"
sudo snap install microceph --channel=quincy/stable --cohort="+"
sudo snap install microovn --channel=22.03/stable --cohort="+"
sudo snap install microcloud --channel=latest/stable --cohort="+"
```

> **Note**
>
> Make sure to install the same version of the snaps on all machines. See *How to manage the snaps* for more information.
>
> If you don't want to use MicroCloud's full functionality, you can install only some of the snaps. However, this is not recommended.

2.2.2 How to manage the snaps

MicroCloud is distributed as a snap. The benefit of packaging MicroCloud as a snap is that it makes it possible to include the required dependencies, and that it allows MicroCloud to be installed on many different Linux distributions. The snap ensures that MicroCloud runs in a consistent environment.

Because MicroCloud uses a set of *other snaps*, you must make sure to have suitable versions of these snaps installed on all machines of your MicroCloud cluster. The installed snap versions must be compatible with one another, and for each of the snaps, the same version must be installed on all machines.

Choose the right channel and track

Snaps come with different channels that define which release of a snap is installed and tracked for updates. See Channels and tracks in the snap documentation for detailed information.

MicroCloud currently provides only the `latest` track.

> **Tip**
>
> In general, you should use the default channels for all snaps required to run MicroCloud.
>
> See *How to get support* for a list of supported channels that are orchestrated to work together.

When installing a snap, specify the channel as follows:

```
sudo snap install <snap_name> --channel=<channel>
```

For example:

```
sudo snap install microcloud --channel=latest/stable
```

To see all available channels of a snap, run the following command:

```
snap info <snap_name>
```

Control updates

By default, snaps are updated automatically. In the case of MicroCloud, this can be problematic because the related snaps must always use compatible versions, and because all machines of a cluster must use the same version of each snap.

Therefore, you should schedule your updates and make sure that all cluster members are in sync regarding the snap versions that they use.

Schedule updates

There are two methods for scheduling when your snaps should be updated:

- You can hold snap updates for a specific time, either for specific snaps or for all snaps on your system. After the duration of the hold, or when you remove the hold, your snaps are automatically refreshed.
- You can specify a system-wide refresh window, so that snaps are automatically refreshed only within this time frame. Such a refresh window applies to all snaps.

Hold updates

You can hold snap updates for a specific time or forever, for all snaps or for specific snaps.

Which strategy to choose depends on your use case. If you want to fully control updates to your MicroCloud setup, you should put a hold on all related snaps until you decide to update them.

Enter the following command to indefinitely hold all updates to the snaps needed for MicroCloud:

```
sudo snap refresh --hold lxd microceph microovn microcloud
```

When you choose to update your installation, use the following commands to remove the hold, update the snaps, and hold the updates again:

```
sudo snap refresh --unhold lxd microceph microovn microcloud
sudo snap refresh lxd --cohort="+"
sudo snap refresh microceph --cohort="+"
sudo snap refresh microovn --cohort="+"
sudo snap refresh microcloud --cohort="+"
sudo snap refresh --hold lxd microceph microovn microcloud
```

See Hold refreshes in the snap documentation for detailed information about holding snap updates.

Specify a refresh window

Depending on your setup, you might want your snaps to update regularly, but only at specific times that don't disturb normal operation.

You can achieve this by specifying a refresh timer. This option defines a refresh window for all snaps that are installed on the system.

For example, to configure your system to update snaps only between 8:00 am and 9:00 am on Mondays, set the following option:

```
sudo snap set system refresh.timer=mon,8:00-9:00
```

You can use a similar mechanism (setting `refresh.hold`) to hold snap updates as well. However, in this case the snaps will be refreshed after 90 days, irrespective of the value of `refresh.hold`.

See Control updates with system options in the snap documentation for detailed information.

Keep cluster members in sync

The cluster members that are part of the MicroCloud deployment must always run the same version of the snaps. This means that when the snaps on one of the cluster members are refreshed, they must also be refreshed on all other cluster members before MicroCloud is operational again.

Snap updates are delivered as progressive releases, which means that updated snap versions are made available to different machines at different times. This method can cause a problem for cluster updates if some cluster members are refreshed to a version that is not available to other cluster members yet.

To avoid this problem, use the `--cohort="+"` flag when refreshing your snaps:

```
sudo snap refresh lxd --cohort="+"
sudo snap refresh microceph --cohort="+"
sudo snap refresh microovn --cohort="+"
sudo snap refresh microcloud --cohort="+"
```

This flag ensures that all machines in a cluster see the same snap revision and are therefore not affected by a progressive rollout.

Use a Snap Store Proxy

If you manage a large MicroCloud deployment and you need absolute control over when updates are applied, consider installing a Snap Store Proxy.

The Snap Store Proxy is a separate application that sits between the snap client command on your machines and the snap store. You can configure the Snap Store Proxy to make only specific snap revisions available for installation.

See the Snap Store Proxy documentation for information about how to install and register the Snap Store Proxy.

After setting it up, configure the snap clients on all cluster members to use the proxy. See Configuring snap devices for instructions.

You can then configure the Snap Store Proxy to override the revisions for the snaps that are needed for MicroCloud:

```
sudo snap-proxy override lxd <channel>=<revision>
sudo snap-proxy override microceph <channel>=<revision>
sudo snap-proxy override microovn <channel>=<revision>
sudo snap-proxy override microcloud <channel>=<revision>
```

2.2.3 How to initialise MicroCloud

The *initialisation process* bootstraps the MicroCloud cluster. You run the initialisation on one of the machines, and it configures the required services on all machines.

Interactive configuration

If you run the initialisation process in interactive mode (the default), you are prompted for information about your machines and how you want to set them up. The questions that you are asked might differ depending on your setup; for example, if you do not have the MicroOVN snap installed, you will not be prompted to configure your network, and if your machines don't have local disks, you will not be prompted to set up local storage.

The following instructions show the full initialisation process.

> **Tip**
>
> During initialisation, MicroCloud displays tables of entities to choose from.
>
> To select specific entities, use the `Up` and `Down` keys to choose a table row and select it with the `Space` key. To select all rows, use the `Right` key. You can filter the table rows by typing one or more characters.
>
> When you have selected the required entities, hit `Enter` to confirm.

Complete the following steps to initialise MicroCloud:

1. On one of the machines, enter the following command:

   ```
   sudo microcloud init
   ```

2. Select the IP address that you want to use for MicroCloud's internal traffic (see *Network interface for intra-cluster traffic*). MicroCloud automatically detects the available addresses (IPv4 and IPv6) on the existing network interfaces and displays them in a table.

   You must select exactly one address.

3. Decide if you want to limit the search for other machines.

   If you accept the default (`yes`), MicroCloud will automatically detect machines in the local subnet. Otherwise, it will detect all available machines, which might include duplicates (if machines are available both on IPv4 and on IPv6).

   See *Automatic server detection* for more information.

4. Select the machines that you want to add to the MicroCloud cluster.

   MicroCloud displays all machines that it detects. This list will periodically update as new machines are detected. Make sure that all machines that you select have the required snaps installed.

5. Select whether you want to set up local storage.

   > **Note**
   >
   > To set up local storage, each machine must have a local disk. The disks must not contain any partitions.

   If you choose `yes`, configure the local storage:

   1. Select the disks that you want to use for local storage.

      You must select exactly one disk from each machine.

   2. Select whether you want to wipe any of the disks. Wiping a disk will destroy all data on it.

6. Select whether you want to set up distributed storage (using MicroCeph).

> **Note**
>
> To set up distributed storage, you need at least three additional disks on at least three different machines. The disks must not contain any partitions.

If you choose `yes`, configure the distributed storage:

1. Select the disks that you want to use for distributed storage.

   You must select at least three disks.

2. Select whether you want to wipe any of the disks. Wiping a disk will destroy all data on it.

3. Select whether you want to encrypt any of the disks. Encrypting a disk will store the encryption keys in the Ceph key ring inside the Ceph configuration folder.

   > **Note**
   >
   > Encryption requires a kernel with `dm_crypt` enabled. See how full disk encryption works in the MicroCeph documentation for more information.

4. You can choose to optionally set up a CephFS distributed file system.

7. Select either an IPv4 or IPv6 CIDR subnet for the Ceph internal traffic. You can leave it empty to use the default value, which is the MicroCloud internal network (see *How to configure Ceph networking* for how to configure it).

8. Select whether you want to set up distributed networking (using MicroOVN).

   If you choose `yes`, configure the distributed networking:

   1. Select the network interfaces that you want to use (see *Network interface to connect to the uplink network*).

      You must select one network interface per machine.

   2. If you want to use IPv4, specify the IPv4 gateway on the uplink network (in CIDR notation) and the first and last IPv4 address in the range that you want to use with LXD.

   3. If you want to use IPv6, specify the IPv6 gateway on the uplink network (in CIDR notation).

9. MicroCloud now starts to bootstrap the cluster. Monitor the output to see whether all steps complete successfully. See *Bootstrapping process* for more information.

   Once the initialisation process is complete, you can start using MicroCloud.

See an example of the full initialisation process in the *Get started with MicroCloud* tutorial.

Excluding MicroCeph or MicroOVN from MicroCloud

If the MicroOVN or MicroCeph snap is not installed on the system that runs **`microcloud init`**, you will be prompted with the following question:

```
MicroCeph not found. Continue anyway? (yes/no) [default=yes]:

MicroOVN not found. Continue anyway? (yes/no) [default=yes]:
```

If you choose `yes`, only existing services will be configured on all systems. If you choose `no`, the setup will be cancelled.

All other systems must have at least the same set of snaps installed as the system that runs **microcloud init**, otherwise they will not be available to select from the list of systems. Any questions associated to these systems will be skipped. For example, if MicroCeph is not installed, you will not be prompted for distributed storage configuration.

Reusing an existing MicroCeph or MicroOVN with MicroCloud

If some of the systems are already part of a MicroCeph or MicroOVN cluster, you can choose to reuse this cluster when initialising MicroCloud when prompted with the following question:

```
"micro01" is already part of a MicroCeph cluster. Do you want to add this cluster to␣
↪MicroCloud? (add/skip) [default=add]:

"micro01" is already part of a MicroOVN cluster. Do you want to add this cluster to␣
↪MicroCloud? (add/skip) [default=add]:
```

If you choose `add`, MicroCloud will add the remaining systems selected for initialisation to the pre-existing cluster. If you choose `skip`, the respective service will not be set up at all.

If more than one MicroCeph or MicroOVN cluster exists among the systems, the MicroCloud initialisation will be cancelled.

Non-interactive configuration

If you want to automate the initialisation process, you can provide a preseed configuration in YAML format to the **microcloud init** command:

```
cat <preseed_file> | microcloud init --preseed
```

The preseed YAML file must use the following syntax:

```
# `lookup_subnet` limits the subnet when looking up systems with mDNS.
lookup_subnet: 10.0.0.1/24
# `lookup_interface` limits the interface when looking up systems with mDNS.
lookup_interface: eth0

# `systems` lists the systems we expect to find by their host name.
#   `name` represents the host name
#   `ovn_uplink_interface` is optional and represents the name of the interface reserved␣
↪for use with OVN.
#   `storage` is optional and represents explicit paths to disks for each system.
systems:
- name: micro01
  ovn_uplink_interface: eth1
- name: micro02
  ovn_uplink_interface: eth1
  storage:
    local:
      path: /dev/nvme5n1
      wipe: true
    ceph:
      - path: /dev/nvme4n1
        wipe: true
      - path: nvme3n1
```

(continues on next page)

(continued from previous page)

```
        wipe: true
        encrypt: true
- name: micro03
  ovn_uplink_interface: eth1
- name: micro04
  ovn_uplink_interface: eth1

# `ceph` is optional and represents the Ceph global configuration
ceph:
  internal_network: 10.0.1.0/24
  public_network: 10.0.2.0/24

# `ovn` is optional and represents the OVN & uplink network configuration for LXD.
ovn:
  ipv4_gateway: 192.0.2.1/24
  ipv4_range: 192.0.2.100-192.0.2.254
  ipv6_gateway: 2001:db8:d:200::1/64
  dns_servers: 192.0.2.1,2001:db8:d:200::1

# `storage` is optional and is used as basic filtering logic for finding disks across all␣
↪systems.
# Filters are checked in order of appearance.
# The names and values of each key correspond to the YAML field names for the `api.
↪ResouresStorageDisk`
# struct here:
# https://github.com/canonical/lxd/blob/c86603236167a43836c2766647e2fac97d79f899/shared/
↪api/resource.go#L591
# Supported operands: &&, ||, <, >, <=, >=, ==, !=, !
# String values must not be in quotes unless the string contains a space.
# Single quotes are fine, but double quotes must be escaped.
# `find_min` and `find_max` can be used to validate the number of disks each filter finds.
# `cephfs: true` can be used to optionally set up a CephFS file system alongside Ceph␣
↪distributed storage.
storage:
  cephfs: true
  local:
    - find: size > 10GiB && size < 50GiB && type == nvme
      find_min: 1
      find_max: 1
      wipe: true
    - find: size > 10GiB && size < 50GiB && type == hdd && block_size == 512 && model ==
↪'Samsung %'
      find_min: 3
      find_max: 3
      wipe: false
  ceph:
    - find: size > 10GiB && size < 50GiB && type == nvme
      find_min: 1
      find_max: 2
      wipe: true
    - find: size > 10GiB && size < 50GiB && type == hdd && partitioned == false && block_
↪size == 512 && model == 'Samsung %'
```

(continues on next page)

(continued from previous page)

```
    find_min: 3
    find_max: 8
    wipe: false
```

2.2.4 How to configure Ceph networking

When running **microcloud init**, you are asked if you want to provide a custom subnet for the Ceph cluster. Here is the question you will be asked:

```
What subnet (either IPv4 or IPv6 CIDR notation) would you like your Ceph internal traffic
on? [default:  203.0.113.0/24]:  <answer>
```

You can choose to skip this question (just hit `Enter`) and use the default value which is the subnet used for the internal MicroCloud traffic. This is referred to as a *usual* Ceph networking setup.

Sometimes, you want to be able to use different network interfaces for some Ceph related usages. Let's imagine you have machines with network interfaces that are tailored for high throughput and low latency data transfer, like 100 GbE+ QSFP links, and other ones that might be more suited for management traffic, like 1 GbE or 10 GbE links.

In this case, it would probably be ideal to set your Ceph internal (or cluster) traffic on the high throughput network interface. This is referred to as a *partially disaggregated* Ceph networking setup.

To use a partially disaggregated Ceph networking setup with your MicroCloud, specify the corresponding subnets during the MicroCloud initialisation process.

The following instructions build on the *Get started with MicroCloud* tutorial and show how you can test setting up a MicroCloud with disaggregated Ceph networking inside a LXD setup.

1. Create the dedicated network for Ceph:
    1. First, just like when you created an uplink network for MicroCloud so that the cluster members could have external connectivity, you will need to create a dedicated network for the Ceph cluster members to communicate with each other. Let's call it `cephbr0`:

        ```
        lxc network create cephbr0
        ```

    2. Enter the following commands to find out the assigned IPv4 and IPv6 addresses for the networks and note them down:

        ```
        lxc network get cephbr0 ipv4.address
        lxc network get cephbr0 ipv6.address
        ```

2. Create the network interfaces that will be used for the Ceph networking setup for each VM:
    1. Add the network device for the `cephbr0` network:

        ```
        lxc config device add micro1 eth2 nic network=cephbr0 name=eth2
        lxc config device add micro2 eth2 nic network=cephbr0 name=eth2
        lxc config device add micro3 eth2 nic network=cephbr0 name=eth2
        lxc config device add micro4 eth2 nic network=cephbr0 name=eth2
        ```

3. Now, just like in the tutorial, start the VMs.
4. On each VM, bring the network interfaces up and give them an IP address within their network subnet:

1. For the `cephbr0` network, do the following for each VM:

```
# If the `cephbr0` gateway address is `10.0.1.1/24` (subnet should be `10.0.1.0/
↪24`)
ip link set enp7s0 up
# `X` should be a number between 2 and 254, different for each VM
ip addr add 10.0.1.X/24 dev enp7s0
```

5. Now, you can start the MicroCloud initialisation process and provide the subnets you noted down in step 1.c when asked for the Ceph networking subnets.
6. We will use `cephbr0` for the Ceph internal traffic. In a production setup, you'd choose the fast subnet for the internal Ceph traffic:

```
What subnet (either IPv4 or IPv6 CIDR notation) would you like your Ceph internal␣
↪traffic on? [default: 203.0.113.0/24]: 10.0.1.0/24

Interface "enp7s0" ("10.0.1.3") detected on cluster member "micro2"
Interface "enp7s0" ("10.0.1.4") detected on cluster member "micro3"
Interface "enp7s0" ("10.0.1.2") detected on cluster member "micro1"
```

7. The MicroCloud initialisation process will now continue as usual and the Ceph cluster will be configured with the networking setup you provided.
8. You can now inspect the Ceph network setup:
   1. Inspect the Ceph configuration file:

      ```
      root@micro1:~# microceph.ceph config dump                    WHO MASK LEVEL OPTION
      VALUE ROglobal advanced cluster_network 10.0.1.0/24 *global advanced
      osd_pool_default_crush_rule 2
      ```

   2. Inspect your Ceph-related network traffic:

      ```
      root@micro1:~# lxc launch ubuntu:22.04 u5 -s remote Creating c1Starting c1
      ```

   3. At the same time, observe the Ceph traffic on the `enp7s0` (or `enp8s0` in a fully disaggregated setup) interface (on any cluster member) using `tcpdump`:

      ```
      root@micro2:~# tcpdump -i enp7s0 17:48:48.600971 IP 10.0.1.4.6804 > micro1.48746:
      Flags [P.], seq 329386555:329422755, ack 245889462, win 24576, options [nop,
      nop,TS val 3552095031 ecr 3647909539], length 3620017:48:48.601012 IP micro1.
      48746 > 10.0.1.4.6804:  Flags [.], ack 329386555, win 24317, options [nop,
      nop,TS val 3647909564 ecr 3552095031], length 017:48:48.600971 IP 10.0.1.4.
      6804 > micro1.48746:  Flags [P.], seq 329422755:329451715, ack 245889462, win
      24576, options [nop,nop,TS val 3552095031 ecr 3647909563], length 2896017:48:48.
      601089 IP 10.0.1.4.6804 > micro1.48746:  Flags [P.], seq 329451715:329516875,
      ack 245889462, win 24576, options [nop,nop,TS val 3552095031 ecr 3647909563],
      length 6516017:48:48.601089 IP 10.0.1.4.6804 > micro1.48746:  Flags [P.],
      seq 329516875:329582035, ack 245889462, win 24576, options [nop,nop,TS val
      3552095031 ecr 3647909563], length 6516017:48:48.601089 IP 10.0.1.4.6804 >
      micro1.48746:  Flags [P.], seq 329582035:329624764, ack 245889462, win 24576,
      options [nop,nop,TS val 3552095031 ecr 3647909563], length 4272917:48:48.601204
      IP micro1.48746 > 10.0.1.4.6804:  Flags [.], ack 329624764, win 23357, options
      [nop,nop,TS val 3647909564 ecr 3552095031], length 017:48:48.601206 IP 10.0.
      1.4.6803 > micro1.33328:  Flags [P.], seq 938255:938512, ack 359644195, win
      24576, options [nop,nop,TS val 3552095031 ecr 3647909540], length 25717:48:48.
      601310 IP micro1.48746 > 10.0.1.4.6804:  Flags [P.], seq 245889462:245889506,
      ```

```
ack 329624764, win 24576, options [nop,nop,TS val 3647909564 ecr 3552095031],
length 4417:48:48.602839 IP micro1.48746 > 10.0.1.4.6804:  Flags [P.], seq
245889506:245889707, ack 329624764, win 24576, options [nop,nop,TS val
3647909566 ecr 3552095031], length 20117:48:48.602947 IP 10.0.1.4.6804 > micro1.
48746:  Flags [.], ack 245889707, win 24576, options [nop,nop,TS val 3552095033
ecr 3647909564], length 017:48:48.602975 IP 10.0.1.4.6804 > micro1.48746:  Flags
[P.], seq 329624764:329624808, ack 245889707, win 24576, options [nop,nop,TS
val 3552095033 ecr 3647909564], length 4417:48:48.603028 IP 10.0.1.4.6803 >
micro1.33328:  Flags [P.], seq 938512:938811, ack 359644195, win 24576, options
[nop,nop,TS val 3552095033 ecr 3647909540], length 29917:48:48.603053 IP micro1.
33328 > 10.0.1.4.6803:  Flags [.], ack 938811, win 1886, options [nop,nop,TS val
3647909566 ecr 3552095031], length 017:48:48.604594 IP micro1.33328 > 10.0.1.4.
6803:  Flags [P.], seq 359644195:359709355, ack 938811, win 1886, options [nop,
nop,TS val 3647909568 ecr 3552095031], length 6516017:48:48.604644 IP micro1.
33328 > 10.0.1.4.6803:  Flags [P.], seq 359709355:359774515, ack 938811, win
1886, options [nop,nop,TS val 3647909568 ecr 3552095031], length 6516017:48:48.
604688 IP micro1.33328 > 10.0.1.4.6803:  Flags [P.], seq 359774515:359839675,
ack 938811, win 1886, options [nop,nop,TS val 3647909568 ecr 3552095031],
length 6516017:48:48.604733 IP micro1.33328 > 10.0.1.4.6803:  Flags [P.], seq
359839675:359904835, ack 938811, win 1886, options [nop,nop,TS val 3647909568
ecr 3552095031], length 6516017:48:48.604751 IP 10.0.1.4.6803 > micro1.33328:
Flags [.], ack 359709355, win 24317, options [nop,nop,TS val 3552095035 ecr
3647909568], length 017:48:48.604757 IP micro1.33328 > 10.0.1.4.6803:  Flags
[P.], seq 359904835:359910746, ack 938811, win 1886, options [nop,nop,TS val
3647909568 ecr 3552095035], length 591117:48:48.604797 IP micro1.33328 > 10.0.
1.4.6803:  Flags [P.], seq 359910746:359975906, ack 938811, win 1886, options
[nop,nop,TS val 3647909568 ecr 3552095035], length 65160
```

2.2.5 How to add a machine

If you want to add a machine to the MicroCloud cluster after the initialisation, use the **`microcloud add`** command:

```
sudo microcloud add
```

Answer the prompts to add the machine. Alternatively, you can add the `--auto` flag to accept the default configuration instead of an interactive setup. You can also add the `--wipe` flag to automatically wipe any disks you add to the cluster.

2.2.6 How to get support

We recommend using the following channels for the snaps required to run MicroCloud:

- For MicroCloud: `latest/[stable|candidate|edge]`
- For LXD: `5.21/[stable|candidate|edge]`
- For MicroCeph: `quincy/[stable|candidate|edge]`
- For MicroOVN: `22.03/[stable|candidate|edge]`

> **Note**
>
> Currently, there is no LTS version of MicroCloud.

Support and community

The following channels are available for you to interact with the MicroCloud community:

- You can file bug reports and feature requests as GitHub issues.
- To ask questions, go to the MicroCloud section of our discussion forum.

Commercial support

Commercial support for MicroCloud is available through Ubuntu Pro (Ubuntu Pro (Infra-only) or full Ubuntu Pro). The support will cover all LTS versions for five years starting from the day of the release.

See the full Ubuntu Pro service description for detailed information about what support Ubuntu Pro provides.

Documentation

See the MicroCloud documentation for official product documentation.

You can find additional resources on the website and in the discussion forum.

2.2.7 How to contribute to MicroCloud

The MicroCloud team appreciates contributions to the project, through pull requests, issues on the GitHub repository, or discussions or questions on the forum.

Check the following guidelines before contributing to the project.

Code of Conduct

When contributing, you must adhere to the Code of Conduct. MicroCloud follows the Ubuntu Code of Conduct.

Licence and copyright

By default, any contribution to this project is made under the AGPLv3 licence. See the licence in the MicroCloud GitHub repository for detailed information.

All contributors must sign the Canonical contributor licence agreement, which gives Canonical permission to use the contributions. The author of a change remains the copyright holder of their code (no copyright assignment).

Pull requests

Propose your changes to this project as pull requests on GitHub.

Proposed changes will then go through review there and once approved, be merged in the main branch.

Commit structure

Use separate commits for every logical change, and for changes to different components. Prefix your commits with the component that is affected, using the code tree structure. For example, prefix a commit that updates the MicroCloud service with `microcloud/service:`.

This structure makes it easier for contributions to be reviewed and also greatly simplifies the process of porting fixes to other branches.

Developer Certificate of Origin

To improve tracking of contributions to this project we use the DCO 1.1 and use a "sign-off" procedure for all changes going into the branch.

The sign-off is a simple line at the end of the explanation for the commit which certifies that you wrote it or otherwise have the right to pass it on as an open-source contribution.

```
Developer Certificate of Origin
Version 1.1

Copyright (C) 2004, 2006 The Linux Foundation and its contributors.
660 York Street, Suite 102,
San Francisco, CA 94110 USA

Everyone is permitted to copy and distribute verbatim copies of this
license document, but changing it is not allowed.

Developer's Certificate of Origin 1.1

By making a contribution to this project, I certify that:

(a) The contribution was created in whole or in part by me and I
    have the right to submit it under the open source license
    indicated in the file; or

(b) The contribution is based upon previous work that, to the best
    of my knowledge, is covered under an appropriate open source
    license and I have the right under that license to submit that
    work with modifications, whether created in whole or in part
    by me, under the same open source license (unless I am
    permitted to submit under a different license), as indicated
    in the file; or

(c) The contribution was provided directly to me by some other
    person who certified (a), (b) or (c) and I have not modified
    it.

(d) I understand and agree that this project and the contribution
    are public and that a record of the contribution (including all
    personal information I submit with it, including my sign-off) is
    maintained indefinitely and may be redistributed consistent with
    this project or the open source license(s) involved.
```

An example of a valid sign-off line is:

```
Signed-off-by: Random J Developer <random@developer.org>
```

Use a known identity and a valid e-mail address, and make sure that you have signed the Canonical contributor licence agreement. Sorry, no anonymous contributions are allowed.

We also require each commit be individually signed-off by their author, even when part of a larger set. You may find git **`commit -s`** useful.

Contribute to the documentation

We want MicroCloud to be as easy and straight-forward to use as possible. Therefore, we aim to provide documentation that contains the information that users need to work with MicroCloud, that covers all common use cases, and that answers typical questions.

You can contribute to the documentation in various different ways. We appreciate your contributions!

Typical ways to contribute are:

- Add or update documentation for new features or feature improvements that you contribute to the code. We'll review the documentation update and merge it together with your code.
- Add or update documentation that clarifies any doubts you had when working with the product. Such contributions can be done through a pull request or through a post in the discussion forum. Tutorials and other documentation posted in the forum will be considered for inclusion in the docs (through a link or by including the actual content).
- To request a fix to the documentation, open a documentation issue on GitHub. We'll evaluate the issue and update the documentation accordingly.
- Post a question or a suggestion on the discussion forum. We'll monitor the posts and, if needed, update the documentation accordingly.

Documentation framework

MicroCloud's documentation is built with Sphinx and hosted on Read the Docs.

It is written in Markdown with MyST extensions. For syntax help and guidelines, see the MyST style guide and the documentation cheat sheet (source).

For structuring, the documentation uses the Diátaxis approach.

Build the documentation

To build the documentation, run `make doc` from the root directory of the repository. This command installs the required tools and renders the output to the `doc/_build/` directory. To update the documentation for changed files only (without re-installing the tools), run `make doc-html`.

Before opening a pull request, make sure that the documentation builds without any warnings (warnings are treated as errors). To preview the documentation locally, run **`make doc-serve`** and go to `http://localhost:8000` to view the rendered documentation.

When you open a pull request, a preview of the documentation output is built automatically. To see the output, view the details for the `docs/readthedocs.com:canonical-microcloud` check on the pull request.

Automatic documentation checks

GitHub runs automatic checks on the documentation to verify the spelling, the validity of links, and the use of inclusive language.

You can (and should!) run these tests locally as well with the following commands:

- Check the spelling: `make doc-spelling`
- Check the validity of links: `make doc-linkcheck`
- Check for inclusive language: `make doc-woke`

2.2.8 How to work with MicroCloud (command cheat sheet)

This guide lists the commands you need to know to do common operations in MicroCloud. This command list is not meant to be exhaustive, but it gives a general overview and serves as an entry point to working with MicroCloud.

Make sure to also check the LXD documentation. Most commands you use in MicroCloud are actually LXD client commands and are documented in more detail in the LXD documentation. There, you can also find the man pages for the `lxc` command.

The sections of the command list provide direct links to specific pages containing more information about the respective topics.

Create and manage instances

See Instances.

Check available images

See How to use remote images.

List remotes	`lxc remote list`
Switch remotes	`lxc remote switch <remote>`
List images	`lxc image list [<remote>:]`

Create instances

See How to create instances and How to access the console.

Create a container (without starting it)	`lxc init [<remote>:]<image> [<name>] [flags]`
Create and start a container	`lxc launch [<remote>:]<image> [<name>] [flags]`
Create a VM (without starting it)	`lxc init [<remote>:]<image> [<name>] --vm [flags]`
Create and start a VM and connect to its VGA console	`lxc launch [<remote>:]<image> [<name>] --vm --console=vga [flags]`

Manage instances

See How to manage instances.

List instances	`lxc list`
Show status information about an instance	`lxc info <instance>`
Start an instance	`lxc start <instance> [flags]`
Stop an instance	`lxc stop <instance> [flags]`
Force-stop an instance	`lxc stop <instance> --force`
Delete an instance	`lxc delete <instance> [--force|--interactive]`
Copy an instance	`lxc copy <instance> <new_name> [flags]`

Access instances

See How to run commands in an instance, How to access the console, and How to access files in an instance.

Run a command inside an instance	`lxc exec <instance> -- <command>`
Get shell access to an instance (if `bash` is installed)	`lxc exec <instance> -- bash`
Get console access to an instance	`lxc console <instance> [flags]`
Pull a file from an instance	`lxc file pull <instance>/<instance_filepath> <local_filepath>`
Push a file to an instance	`lxc file pull <local_filepath> <instance>/<instance_filepath>`

Configure instances

See How to configure instances, How to use profiles, and Instance configuration (and sub-pages).

Show the configuration of an instance	`lxc config show <instance>`
Show the configuration of an instance, including configurations inherited from a profile	`lxc config show <instance> --expanded`
Set some configuration options for an instance (this example limits memory and CPU usage)	`lxc config set <instance> limits.memory=8GiB limits.cpu=4` **Tip** See Instance options for all available instance options.
Override some device options for an instance (this example sets the root disk size)	`lxc config device override <instance> root size=10GiB` **Tip** See Devices for the device options that are available for each device type.
Edit the full configuration of an instance	`lxc config edit <instance>`
Apply a profile to an instance	`lxc profile add <instance> <profile>`

Use `cloud-init`

See How to use cloud-init.

For example, to import an SSH key:

1. Create a profile: **`lxc profile create <profile>`**
2. Run **`lxc profile edit <profile>`** and add the following configuration to the profile:

```
config:
  cloud-init.user-data: |
    #cloud-config
    ssh_authorized_keys:
      - <public_key>
```

3. Launch the instance using that profile (in addition to the `default` profile): **`lxc launch <image> [<name>] --profile default --profile <profile>`**

Manage instance snapshots

See Use snapshots for instance backup.

Create a snapshot	**`lxc snapshot <instance> [<snapshot_name>] [flags]`**
View information about a snapshot	**`lxc config show <instance>/<snapshot_name>`**
View all snapshots of an instance	**`lxc info <instance>`**
Restore a snapshot	**`lxc restore <instance> <snapshot_name> [--stateful]`**
Delete a snapshot	**`lxc delete <instance>/<snapshot_name>`**
Create an instance from a snapshot	**`lxc copy <instance>/<snapshot_name> <new_instance>`**

Configure storage

See How to manage storage volumes.

To create a storage pool, see How to manage storage pools. However, keep in mind that for MicroCloud to be able to use the storage pool, it must be created for the cluster and not only for one machine. Therefore, the following example commands use the `remote` storage pool that is automatically set up in MicroCloud.

Create a custom storage volume of content type `filesystem` in the `remote` storage pool	**`lxc storage volume create remote <volume>`**
Create a custom storage volume of content type `block` in the `remote` storage pool	**`lxc storage volume create remote <volume> --type=block`**
Attach a custom storage volume of content type `filesystem` to an instance	**`lxc storage volume attach remote <volume> <instance> <location>`**
Attach a custom storage volume of content type `block` to an instance	**`lxc storage volume attach remote <volume> <instance>`**

Configure networking

See Networking (and sub-pages).

Create a network	`lxc network create <network> --type=<type> [options]` **Tip** See How to create a network for detailed information.
Attach an instance to a network	`lxc network attach <network> <instance> [<device>] [<interface>]` **Tip** See Attach a network to an instance for detailed information.

Create and apply a network ACL to an instance	`lxc network acl rule add <ACL> ingress|egress [properties]`
	`lxc network set <network> security.acls="<ACL>"`
	Tip
	See How to configure network ACLs for detailed information.

Expose an instance on an external IP	`lxc network forward <network> create <external_IP> target_address=<instance_IP>` **Tip** See How to configure network forwards for detailed information.

Use projects

See About projects and Projects (and sub-pages).

Create a project	`lxc project create <project> [--config <option>]`
Configure a project	`lxc project set <project> <option>`
Switch to a project	`lxc project switch <project>`

Configure the LXD server

See How to configure the LXD server.

Configure server options	`lxc config set <key> <value>` **Tip** See Server configuration for all available server options.

Enable GUI access to the LXD cluster	`lxc config set core.https_address :8443`
	Then enable the UI on the snap and reload the snap:
	`snap set lxd ui.enable=true`
	`sudo systemctl reload snap.lxd.daemon`
	Tip
	See How to access the LXD web UI for detailed information.

Manage the MicroCloud cluster

See How to manage instances in a cluster and Evacuate and restore cluster members.

Inspect the cluster status	`microcloud cluster list`
	`lxc cluster list`
	`microceph cluster list`
	`microovn cluster list`

Move an instance to a different cluster member	`lxc move <instance> --target <member>`
Copy an instance from a different LXD server	Add one of the MicroCloud cluster members as a remote on the different LXD server and copy or move the instance from that server. `lxc copy <instance> <remote>` **Tip** See How to move existing LXD instances between servers for details.
Evacuate a cluster member	`lxc cluster evacuate <member>`
Restore a cluster member	`lxc cluster restore <member>`

2.2.9 How to recover a MicroCloud cluster

Note

Each MicroCloud service uses the Dqlite distributed database for highly-available storage. While the cluster recovery process is similar for each service, this document only covers cluster recovery for the `microcloudd` daemon. For cluster recovery procedures for LXD, MicroCeph and MicroOVN, see:

- LXD Cluster Recovery
- MicroOVN Launchpad Bug
- MicroCeph Issue

MicroCloud requires a majority of the database voters (a quorum) to be accessible in order to perform database operations. If a cluster has less than a quorum of voters up and accessible, then database operations will no longer be possible on the entire cluster.

If the loss of quorum is temporary (e.g. some members temporarily lose power), database operations will be restored when the offline members come back online.

This document describes how to recover database access if the offline members have been lost without the possibility of recovery (e.g. disk failure).

Recovery procedure

1. Shut down all cluster members before performing cluster recovery:

   ```
   sudo snap stop microcloud
   ```

2. Once all cluster members are shut down, determine which Dqlite database is most up to date. Look for files in `/var/snap/microcloud/common/state/database` whose filenames are two numbers separated by a dash (i.e. `0000000000056436-0000000000056501`). The largest second number in that directory is the end index of the most recently closed segment (56501 in the example). The cluster member with the highest end index is the most up to date.

3. On the most up-to-date cluster member, use the following command to reconfigure the Dqlite roles for each member:

   ```
   sudo microcloud cluster recover
   ```

4. As indicated by the output of the above command, copy `/var/snap/microcloud/common/state/recovery_db.tar.gz` to the same path on each cluster member.

5. Restart MicroCloud. The recovered database tarball will be loaded on daemon startup. Once a quorum of voters have been started, the MicroCloud database will become available.

   ```
   sudo snap start microcloud
   ```

Backups

MicroCloud creates a backup of the database directory before performing the recovery operation to ensure that no data is lost. The backup tarball is created in `/var/snap/microcloud/common/state/`. If the cluster recovery operation fails, use the following commands to restore the existing database:

```
cd /var/snap/microcloud/common/state
sudo mv database broken_db
sudo tar -xf db_backup.TIMESTAMP.tar.gz
```

2.3 Reference

The reference material in this section provides technical descriptions of MicroCloud.

2.3.1 Requirements

Hardware requirements

MicroCloud requires a minimum of three machines. It supports up to 50 machines.

Each machine must have at least 8 GiB of RAM (more depending on the connected disks). You can mix different processor architectures within the same MicroCloud cluster.

If you want to add further machines after the initial initialisation, you can use the **`microcloud add`** command.

To use local storage, each machine requires a local disk. To use distributed storage, at least three additional disks (not only partitions) for use by Ceph are required, and these disks must be on at least three different machines.

Also see Ceph's hardware recommendations.

Networking requirements

For networking, MicroCloud requires two dedicated network interfaces: one for intra-cluster communication and one for external connectivity. To allow for external connectivity, MicroCloud requires an uplink network that supports broadcast and multicast. See *Networking* for more information.

The IP addresses of the machines must not change after installation, so DHCP is not supported.

Software requirements

MicroCloud requires snapd version 2.59 or newer.

Also see LXD's Requirements and Ceph's OS Recommendations.

2.3.2 Snaps

To run MicroCloud, you must install the following snaps:

- MicroCloud snap
- LXD snap
- MicroCeph snap
- MicroOVN snap

2.4 Explanation

The explanatory guides in this section introduce you to the concepts used in MicroCloud and help you understand how things fit together.

2.4.1 About MicroCloud

The MicroCloud snap drives three other snaps (LXD, MicroCeph, and MicroOVN), enabling automated deployment of a highly available LXD cluster for compute with Ceph as the storage driver and OVN as the managed network.

During initialisation, MicroCloud detects the other servers and then prompts you to add disks to Ceph and configure the networking setup.

At the end of this, you'll have an OVN cluster, a Ceph cluster, and a LXD cluster. LXD itself will have been configured with both networking and storage suitable for use in a cluster.

LXD cluster

MicroCloud sets up a LXD cluster. You can use the **`microcloud cluster`** command to show information about the cluster members, or to remove specific members.

Apart from that, you can use LXD commands to manage the cluster. See Clustering in the LXD documentation for more information.

Networking

By default, MicroCloud uses MicroOVN for networking, which is a minimal wrapper around OVN (Open Virtual Network).

For external connectivity, MicroOVN requires an uplink network. This uplink network must support broadcast and multicast traffic (so that IP adverts and packets can flow between the OVN virtual router and the uplink network). Proper Ethernet networks generally fulfil these requirements, but virtual cloud environments often don't.

Each machine in the MicroCloud cluster should have at least two available network interfaces (which can be connected to the same network or to different networks):

Network interface for intra-cluster traffic
: MicroCloud requires one network interface that is pre-configured with an IP address that is within the same subnet as the IPs of the other cluster members. The network that it is connected to must support multicast.

  This network interface can be, for example, a dedicated physical network interface, a VLAN, or a virtual function on an SR-IOV (Single root I/O virtualisation)-capable network interface. It serves as the dedicated network interface for MicroOVN and is used for multicast discovery (during setup) and all internal traffic between the MicroCloud, OVN, and Ceph members.

Network interface to connect to the uplink network
: MicroCloud requires one network interface for connecting OVN to the uplink network. This network interface must either be an unused interface that does not have an IP address configured, or a bridge.

  MicroCloud configures this interface as an uplink interface that provides external connectivity to the MicroCloud cluster.

  You can specify a different interface to be used as the uplink interface for each cluster member. MicroCloud requires that all uplink interfaces are connected to the uplink network, using the gateway and IP address range information that you provide during the MicroCloud initialisation process.

If you have a network interface that is configured as a Linux bridge, you can use it for both network interfaces.

See OVN network in the LXD documentation for more information.

If you decide to not use MicroOVN, MicroCloud falls back on the Ubuntu fan for basic networking. MicroCloud will still be usable, but you will see some limitations:

- When you move an instance from one cluster member to another, its IP address changes.
- Egress traffic leaves from the local cluster member (while OVN provides shared egress). As a result of this, network forwarding works at a basic level only, and external addresses must be forwarded to a specific cluster member and don't fail over.
- There is no support for hardware acceleration, load balancers, or ACL functionality within the local network.

Dedicated internal network for Ceph

You can set up a dedicated network for Ceph to separate the internal Ceph traffic from the rest of the MicroCloud cluster traffic.

See *How to configure Ceph networking* for how to set up a dedicated internal network for Ceph.

Storage

You have two options for storage in MicroCloud: local storage or distributed storage.

Local storage is faster, but less flexible and not fail-safe. To use local storage, each machine in the cluster requires a local disk. Disk sizes can vary.

For distributed storage, MicroCloud uses MicroCeph, which is a lightweight way of deploying a Ceph cluster. To use distributed storage, you must have at least three disks (attached to at least three different machines).

Troubleshooting

MicroCloud does not manage the services that it deploys. After the deployment process, the individual services are operating independently. If anything goes wrong, each service is responsible for handling recovery.

So, for example, if `lxc cluster list` shows that a LXD cluster member is offline, follow the usual steps for recovering an offline cluster member (in the simplest case, restart the LXD snap on the machine). The same applies to MicroOVN and MicroCeph.

2.4.2 About the initialisation process

See *How to initialise MicroCloud* for instructions on how to set up MicroCloud.

Automatic server detection

MicroCloud uses mDNS (multicast DNS) to automatically detect other servers on the network. This method works in physical networks, but it is usually not supported in a cloud environment.

The scan can be limited to the default local subnet of the network interface you select.

MicroCloud will display all servers that it detects and periodically update the list. You can select the servers you want to add to the MicroCloud cluster.

Bootstrapping process

After you provide the required information to *initialise MicroCloud*, MicroCloud starts bootstrapping the cluster.

The bootstrapping process consists of the following steps:

1. MicroCloud initialises the first server (the one where you run the **`microcloud init`** command) and creates the MicroCloud cluster.
2. MicroCloud creates the LXD cluster, the OVN cluster, and the Ceph cluster.
3. MicroCloud issues join tokens for the other servers that are to be added to the cluster.
4. MicroCloud sends the join tokens over the network to the other servers.
5. The other servers initialise their services and join the MicroCloud cluster, the OVN cluster, and the Ceph cluster.

   This step of forming the cluster can take several minutes, mainly because of the initialisation of MicroCeph and adding disks to the Ceph cluster.
6. When the cluster is formed, MicroCloud configures LXD. It sets up networking and storage pools and configures the default profile to use the created OVN network and the distributed storage (if available).

After the initialisation is complete, you can look at the LXD configuration to confirm the setup.

```
user@micro01:~$ lxc cluster list+---------+-------------------------------------+----------------+-----
NAME | URL | ROLES | ARCHITECTURE | FAILURE DOMAIN | DESCRIPTION | STATE | MESSAGE
|+---------+-------------------------------------+----------------+--------------+----------------+-----
micro01 | https://[2001:db8:d:100::169]:8443 | database-leader | aarch64
| default | | ONLINE | Fully operational || | | database | | | | |
|+---------+-------------------------------------+----------------+--------------+----------------+-----
micro02 | https://[2001:db8:d:100::170]:8443 | database | aarch64 | default | | ONLINE |
Fully operational |+---------+-------------------------------------+----------------+--------------+----
micro03 | https://[2001:db8:d:100::171]:8443 | database | aarch64 | default | | ONLINE |
Fully operational |+---------+-------------------------------------+----------------+--------------+----
user@micro01:~$ lxc storage list+-----------+--------+-----------------------------------------+----
NAME | DRIVER | DESCRIPTION | USED BY | STATE |+-----------+--------+---------------------------------
local | zfs | Local storage on ZFS | 10 | CREATED |+-----------+--------+---------------------------------
remote | ceph | Distributed storage on Ceph | 7 | CREATED |+-----------+--------+-----------------------
remote-fs | cephfs | Distributed file-system storage using Ceph | 7 | CREATED
|+-----------+--------+---------------------------------------------+---------+---------+
user@micro01:~$ lxc network list+----------+----------+---------+----------------+-------------------
NAME | TYPE | MANAGED | IPV4 | IPV6 | DESCRIPTION | USED BY | STATE
|+----------+----------+---------+----------------+---------------------------+-------------+----------
UPLINK | physical | YES | | | | 2 | CREATED |+----------+----------+---------+----------------+-------
default | ovn | YES | 10.123.123.1/24 | fd42:1234:1234:1234::1/64 | | 5 | CREATED
|+----------+----------+---------+----------------+---------------------------+-------------+----------
eth0 | physical | NO | | | | 0 | |+----------+----------+---------+----------------+------------------
eth0.200 | vlan | NO | | | | 1 | |+----------+----------+---------+----------------+------------------
```

```
user@micro01:~$ lxc profile show default          config:  {}description:  ""devices:eth0:
name:  eth0 network:  default type:  nicroot:  path:  /  pool:  remote type:  diskname:
defaultused_by:  []
```