# CONTENTS

1 In this documentation 3

2 Project and community 5
   2.1 Get started with MicroCloud 5
   2.2 How-to guides 16
   2.3 Reference 30
   2.4 Explanation 31
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
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
- Announcement
- 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.

If you are already familiar with LXD, you are good to go! Otherwise, see the LXD documentation to get started.

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’ll 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.

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:
a. Run `snap version` to find out if snap is installed on your system:

```
user@host:~$ snap version
snap 2.59.4
snapd 2.59.4
series 16ubuntu22.04
kernel 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.

b. 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:

```
```

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:
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 to communicate. 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:
MicroCloud

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

```bash
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:

```bash
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:

   ```bash
   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):

   ```bash
   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:

   ```bash
   ip link set enp6s0 up
   ```

4. Install the required snaps:
5. The LXD snap is already installed. Refresh it to the latest version:

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

### 2.1.6 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:
   ```
   a. As the address for MicroCloud’s internal traffic, select the listed IPv4 address.
   b. Select yes to limit the search for other MicroCloud servers to the local subnet.
   c. Select all listed servers (these should be `micro2`, `micro3`, and `micro4`).
   d. Select yes to set up local storage.
   e. 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.

   ```
   f. You don’t need to wipe any disks (because we just created them).
   g. Select yes to set up distributed storage.
   h. Select yes to confirm that there are fewer disks available than machines.
   i. Select all listed disks (these should be `remote1`, `remote2`, and `remote3`).
   j. You don’t need to wipe any disks (because we just created them).
   k. Select yes to configure distributed networking.
   l. Select all listed network interfaces (these should be `enp6s0` on the four different VMs).
   m. Specify the IPv4 address that you noted down for your `microbr0` network as the IPv4 gateway.
   ```
n. 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.

o. 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.

p. 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 |
+----------------------+--------+
| 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 203.0.113/24? (yes/no) [default=yes]: yes
Scanning 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 |
+---------+--------+---------------+
| micro3 | enp5s0 | 203.0.113.171 |
| micro2 | enp5s0 | 203.0.113.170 |
| 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]: yes
Select 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 |
+----------+---------------+----------+------+------------------------------------------------------+
| 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 |
+----------+---------------+----------+------+------------------------------------------------------+

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]: yes
Unable to find disks on some systems.
Continue anyway? (yes/no) [default=yes]: yes
Select 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
Configure distributed networking? (yes/no) [default=yes]: yes
Select exactly one network interface 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 | 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):
Specify the first IPv4 address in the range to use with LXD:
Specify the last IPv4 address in the range to use with LXD:
Specify the IPv6 gateway (CIDR) on the uplink network (empty to skip IPv6):

Initializing a new cluster
Local MicroCloud is ready
Local LXD is ready
Local MicroOVN is ready
Local MicroCeph is ready
Awaiting cluster formation
Peer "micro2" has joined the cluster
Peer "micro3" has joined the cluster
Peer "micro4" has joined the cluster
Cluster initialization is complete
MicroCloud 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         | |
   +-------+------------------------+----------------+----------------+----------------+-------------+--------+-------------------+

2.1. Get started with MicroCloud
### Inspect the network setup:

| NAME  | ADDRESS | ROLE      | FINGERPRINT | STATUS |+--------+--------------------+----------+------------------------------------------------------------------+--------+|
|-------|---------|-----------|-------------|--------+------------------------------------------------------------------++--------|
| micro4 | https://203.0.113.172:8443 | database-standby | x86_64 | default | | ONLINE | Fully operational |+--------+----------------------------+------------------|--------------|----------------+-------------+--------+-------------------+|
| micro3 | https://203.0.113.171:8443 | database | x86_64 | default | | ONLINE | Fully operational |+--------+----------------------------+------------------|--------------|----------------+-------------+--------+-------------------+|
| micro2 | https://203.0.113.170:8443 | database | x86_64 | default | | ONLINE | Fully operational |+--------+----------------------------+------------------|--------------|----------------+-------------+--------+-------------------+|

### Inspect the storage setup:

```
NAME | DRIVER | DESCRIPTION | USED BY | STATE |
--- | --- | --- | --- | --- |
local | zfs | Local storage on ZFS | 8 | CREATED |
remote | ceph | Distributed storage on Ceph | 1 | CREATED |
```

```
NAME | DRIVER | DESCRIPTION | USED BY | STATE |
--- | --- | --- | --- | --- |
rroot@micro1:~# lxc storage info local info: description: Local storage on ZFS driver: zfs name: local space used: 747.00KiB total space: 9.20GiB used by: volumes: - backups (location "micro1") - backups (location "micro2") - backups (location "micro3") - images (location "micro1") - images (location "micro2") - images (location "micro3") - images (location "micro4")
rroot@micro1:~# lxc storage info remote info: description: Distributed storage on Ceph driver: ceph name: remote space used: 25.41KiB total space: 29.67GiB used by: profiles: - default |
```
MicroCloud

|+-------------+----------+---------+-----------------+--------------------------+-------------+---------+---------+|
|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 | |+---------+----------+---------+-----------------+--------------------------+-------------+---------+---------+|
|lxdovnl | bridge | NO | | | | 0 | |+---------+----------+---------+-----------------+--------------------------+-------------+---------+---------+|


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 ms 64 bytes from 192.0.2.100: icmp_seq=2 ttl=253 time=2.01 ms 64 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 3005ms rtt min/avg/max/mdev = 1.777/1.945/2.052/0.120 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: default used_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:

```
1xc launch ubuntu:22.04 u1
```

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

```
1xc launch ubuntu:22.04 u2 --storage local
```

3. Launch an Ubuntu VM:

```
1xc launch ubuntu:22.04 u3 --vm
```

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

2.1. Get started with MicroCloud
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 | ✓ | micro1 |
+-----------------+---------+--------------------+----------------------------------------------+-----------------+-----------+----------+
| u2 | RUNNING | 198.51.100.3 (eth0) | 2001:db8:d960:91cf:216:3eff:fe79:6765 (eth0) | CONTAINER | ✓ | micro3 |
+-----------------+---------+--------------------+----------------------------------------------+-----------------+-----------+----------+
| u3 | RUNNING | 198.51.100.4 (eth0) | 2001:db8:d960:91cf:216:3eff:fe66:f24b (eth0) | VIRTUAL-MACHINE | ✓ | 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 ms
64 bytes from 198.51.100.3: icmp_seq=2 ttl=64 time=1.74 ms
64 bytes from 198.51.100.3: icmp_seq=3 ttl=64 time=0.985 ms
--- 198.51.100.3 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
round-trip 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
bytes
64 bytes from 2001:db8:d960:91cf:216:3eff:fe66:f24b: icmp_seq=1 ttl=64 time=16.8 ms
64 bytes from 2001:db8:d960:91cf:216:3eff:fe66:f24b: icmp_seq=2 ttl=64 time=3.41 ms
64 bytes from 2001:db8:d960:91cf:216:3eff:fe66:f24b: icmp_seq=3 ttl=64 time=3.86 ms
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
round-trip 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 ms
64 bytes from 93.184.216.34 (93.184.216.34): icmp_seq=2 ttl=49 time=95.2 ms
64 bytes from 93.184.216.34 (93.184.216.34): icmp_seq=3 ttl=49 time=96.2 ms
--- www.example.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2018ms
round-trip 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

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 ms
64 bytes from 192.0.2.101: icmp_seq=2 ttl=253 time=1.04 ms
64 bytes from 192.0.2.101: icmp_seq=3 ttl=253 time=1.68 ms
--- 192.0.2.101 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
round-trip 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 images:ubuntu/22.04 u4 --network isolated

11. Access the shell in u4:

2.1. Get started with MicroCloud
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 ms
64 bytes from 93.184.216.34 (93.184.216.34): icmp_seq=2 ttl=49 time=118 ms
64 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 2004ms
rtt 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.

### 2.2 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 command on all machines:

```
sudo snap install lxd microceph microovn microcloud --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.

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:
MicroCloud

```
sudo snap refresh --unhold lxd microceph microovn microcloud
sudo snap refresh lxd microceph microovn 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 microceph microovn 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:
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. 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. 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.

7. 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.
      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).

8. 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.

---

**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

# `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
```

(continues on next page)
- name: micro02
  ovn_uplink_interface: eth1
  storage:
    local:
      path: /dev/nvme5n1
      wipe: true
    ceph:
      - path: /dev/nvme4n1
        wipe: true
      - path: nvme3n1
        wipe: true
- name: micro03
  ovn_uplink_interface: eth1
- name: micro04
  ovn_uplink_interface: eth1

# `ovn` is optional and represents the OVN & uplink network configuration for LXD.

ovn:
  ipv4_gateway: 10.0.0.1/24
  ipv4_range: 10.0.0.100-10.0.0.254
  ipv6_gateway: cafe::1/64

# `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.
# ResourcesStorageDisk`
# 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.

storage:
  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)
2.2.4 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.

2.2.5 How to get support

MicroCloud is not yet ready for production. We recommend using the default channels for all snaps required to run 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.

Documentation

See the MicroCloud documentation for official product documentation. You can find additional resources on the website and in the discussion forum.

2.2.6 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.

License and copyright

By default, any contribution to this project is made under the AGPL-3.0 license. See the license 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

(continues on next page)
(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 reStructuredText. For syntax help and guidelines, see our reStructuredText style guide. The repository also contains a documentation cheat sheet, which can be helpful as a quick reference.

For structuring, the documentation uses the Diátaxis approach.

Build the documentation

To build the documentation, go to the doc folder of the repository and run `make html`. This command installs the required tools and renders the output to the doc/_build/ directory. Subsequent builds only process files that have changed. To run a clean build of all files, but without re-installing all tools, run `make clean-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 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 spelling`
- Check the validity of links: `make linkcheck`
- Check for inclusive language: `make woke`

2.2.7 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.
MicroCloud

Create and manage instances

See Instances.

Check available images

See How to use remote images.

<table>
<thead>
<tr>
<th>List remotes</th>
<th>lxc remote list</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch remotes</td>
<td>lxc remote switch &lt;remote&gt;</td>
</tr>
<tr>
<td>List images</td>
<td>lxc image list [&lt;remote&gt;]:</td>
</tr>
</tbody>
</table>

Create instances

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

<table>
<thead>
<tr>
<th>Create a container (without starting it)</th>
<th>lxc init [&lt;remote&gt;]:&lt;image&gt; [&lt;name&gt;] [flags]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create and start a container</td>
<td>lxc launch [&lt;remote&gt;]:&lt;image&gt; [&lt;name&gt;] [flags]</td>
</tr>
<tr>
<td>Create a VM (without starting it)</td>
<td>lxc init [&lt;remote&gt;]:&lt;image&gt; [&lt;name&gt;] --vm [flags]</td>
</tr>
<tr>
<td>Create and start a VM and connect to its VGA console</td>
<td>lxc launch [&lt;remote&gt;]:&lt;image&gt; [&lt;name&gt;] --vm --console=vga [flags]</td>
</tr>
</tbody>
</table>

Manage instances

See How to manage instances.

<table>
<thead>
<tr>
<th>List instances</th>
<th>lxc list</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show status information about an instance</td>
<td>lxc info &lt;instance&gt;</td>
</tr>
<tr>
<td>Start an instance</td>
<td>lxc start &lt;instance&gt; [flags]</td>
</tr>
<tr>
<td>Stop an instance</td>
<td>lxc stop &lt;instance&gt; [flags]</td>
</tr>
<tr>
<td>Force-stop an instance</td>
<td>lxc stop &lt;instance&gt; --force</td>
</tr>
<tr>
<td>Delete an instance</td>
<td>lxc delete &lt;instance&gt; [--force</td>
</tr>
<tr>
<td>Copy an instance</td>
<td>lxc copy &lt;instance&gt; &lt;new_name&gt; [flags]</td>
</tr>
</tbody>
</table>

Access instances

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

<table>
<thead>
<tr>
<th>Run a command inside an instance</th>
<th>lxc exec &lt;instance&gt; -- &lt;command&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get shell access to an instance (if bash is installed)</td>
<td>lxc exec &lt;instance&gt; -- bash</td>
</tr>
<tr>
<td>Get console access to an instance</td>
<td>lxc console &lt;instance&gt; [flags]</td>
</tr>
<tr>
<td>Pull a file from an instance</td>
<td>lxc file pull &lt;instance&gt;/&lt;instance_filepath&gt; &lt;local_filepath&gt;</td>
</tr>
<tr>
<td>Push a file to an instance</td>
<td>lxc file pull &lt;local_filepath&gt; &lt;instance&gt;/ &lt;instance_filepath&gt;</td>
</tr>
</tbody>
</table>
Configure instances

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

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

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 AuthorizedKeys:
       - <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.
MicroCloud

| 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).
**MicroCloud**

<table>
<thead>
<tr>
<th>Create a network</th>
<th><code>lxc network create &lt;network&gt; --type=&lt;type&gt; [options]</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tip:</strong></td>
<td>See How to create a network for detailed information.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attach an instance to a network</th>
<th><code>lxc network attach &lt;network&gt; &lt;instance&gt; [&lt;device&gt;] [interface]</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tip:</strong></td>
<td>See Attach a network to an instance for detailed information.</td>
</tr>
</tbody>
</table>

| 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.   |

<table>
<thead>
<tr>
<th>Expose an instance on an external IP</th>
<th><code>lxc network forward &lt;network&gt; create &lt;external_IP&gt; target_address=&lt;instance_IP&gt;</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tip:</strong></td>
<td>See How to configure network forwards for detailed information.</td>
</tr>
</tbody>
</table>

---

**Use projects**

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

<table>
<thead>
<tr>
<th>Create a project</th>
<th><code>lxc project create &lt;project&gt; [--config &lt;option&gt;]</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure a project</td>
<td><code>lxc project set &lt;project&gt; &lt;option&gt;</code></td>
</tr>
<tr>
<td>Switch to a project</td>
<td><code>lxc project switch &lt;project&gt;</code></td>
</tr>
</tbody>
</table>
# Configure the LXD server

See [How to configure the LXD server](#).

<table>
<thead>
<tr>
<th>Configure server options</th>
<th><code>lxc config set &lt;key&gt; &lt;value&gt;</code></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Tip:</strong> See <a href="#">Server configuration</a> for all available server options.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Enable GUI access to the LXD cluster</th>
<th><code>lxc config set core.https_address :8443</code></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Then enable the UI on the snap and reload the snap:</td>
</tr>
<tr>
<td></td>
<td><code>snap set lxd ui.enable=true</code></td>
</tr>
<tr>
<td></td>
<td><code>sudo systemctl reload snap.lxd.daemon</code></td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> See <a href="#">How to access the LXD web UI</a> for detailed information.</td>
</tr>
</tbody>
</table>

## Manage the MicroCloud cluster

See [How to manage instances in a cluster](#) and [Evacuate and restore cluster members](#).

<table>
<thead>
<tr>
<th>Inspect the cluster status</th>
<th><code>microcloud cluster list</code></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>lxc cluster list</code></td>
</tr>
<tr>
<td></td>
<td><code>microceph cluster list</code></td>
</tr>
<tr>
<td></td>
<td><code>microovn cluster list</code></td>
</tr>
</tbody>
</table>

| Move an instance to a different cluster member | `lxc move <instance> --target <member>` |

<table>
<thead>
<tr>
<th>Copy an instance from a different LXD server</th>
<th>Add one of the MicroCloud cluster members as a remote on the different LXD server and copy or move the instance from that server.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>lxc copy &lt;instance&gt; &lt;remote&gt;</code></td>
</tr>
</tbody>
</table>

**Tip:** See [How to move existing LXD instances between servers](#) for details.

<table>
<thead>
<tr>
<th>Evacuate a cluster member</th>
<th><code>lxc cluster evacuate &lt;member&gt;</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Restore a cluster member</td>
<td><code>lxc cluster restore &lt;member&gt;</code></td>
</tr>
</tbody>
</table>

## 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.

**Networking requirements**

For networking, MicroCloud requires a dedicated network interface and an uplink network that is an actual L2 subnet. 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.

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 implementation of OVN (Open Virtual Network). If you decide to not use MicroOVN, MicroCloud falls back on the Ubuntu fan for basic networking. MicroOVN requires an uplink network that is an actual L2 subnet (which is usually not the case in a virtual cloud environment). In addition, MicroOVN requires its own dedicated network interface, 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. See OVN network in the LXD documentation for more information.

**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.

By default, the scan is limited to the local subnet.

MicroCloud will display all servers that it detects, and you can then select the ones 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 | 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 |
|+--------+--------+-----------------------------+---------+---------+|
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

2.4. Explanation
user@micro01:~$ lxc profile show default

config: {}
description: ""
devices:
  eth0:
    name: eth0
    network: default
    type: nic
    root: path: / pool: remote
type: disk
used_by: []