
MicroOVN

Canonical Group Ltd

May 16, 2024

CONTENTS

| | | |
|----------|-------------------------------------|----------|
| 1 | In this documentation | 3 |
| 2 | Project and community | 5 |
| 2.1 | How-to guides | 5 |
| 2.2 | Tutorials | 12 |
| 2.3 | Reference | 14 |
| 2.4 | Developer’s documentation | 15 |

MicroOVN is a snap-based distribution of OVN - [Open Virtual Network](#).

It allows users to deploy an OVN cluster with just a few commands. Aside from the regular OVN packages, MicroOVN comes bundled with a CLI utility (`microovn`) that facilitates operational management. In particular, it simplifies the task of adding/removing cluster members and incorporates status checking out of the box.

Besides the ease of deployment and a convenient CLI tool, another benefit of MicroOVN is in its self-contained nature: it is distributed as a [strictly confined snap](#). This means that it can be easily upgraded/downgraded/removed without affecting the host system.

MicroOVN caters to a wide range of user and environment types. It lowers the barrier of entry to OVN for people that are less familiar with it by automating much of the deployment process. It also provides a fully fledged, unrestricted OVN deployment that is suitable for both development and production environments.

IN THIS DOCUMENTATION

Tutorial

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

How-to guides

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

Reference

Technical information - specifications, APIs, architecture

PROJECT AND COMMUNITY

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

- We follow the Ubuntu community [Code of conduct](#)
- Contribute to the project on [GitHub](#) (documentation contributions go under the docs directory)
- GitHub is also used as our bug tracker
- To speak with us, you can find us in our [MicroOVN Discourse](#) category. Use the [Support](#) sub-category for technical assistance.
- Optionally enable [Ubuntu Pro](#) on your OVN nodes. This is a service that provides the [Livepatch Service](#) and the [Expanded Security Maintenance](#) (ESM) program.

2.1 How-to guides

These How-to guides will cover a variety of operations and configurations that are possible with MicroOVN. They do however assume general Linux knowledge and a basic understanding of OVN.

2.1.1 Working with TLS

Starting with snap revision 111, new deployments of MicroOVN use TLS encryption by default. A self-signed CA certificate is used to issue certificates to all OVN services that require it. They provide authentication and encryption for OVSDB communication. The CA certificate is generated during cluster initialisation (**cluster bootstrap** command).

In the current implementation, self-provisioned certificates are the only mode available. Future releases may include support for externally provided certificates.

Warning: The certificate and private key generated for the self-provisioned CA are currently stored unencrypted in the database on every cluster member. If an attacker gains access to any cluster member, they can use the CA to issue valid certificates that will be accepted by other cluster members.

Certificates CLI

MicroOVN exposes a few commands for basic interaction with TLS certificates.

List certificates

To list currently used certificates:

```
microovn certificates list
```

Example output:

```
[OVN CA]
/var/snap/microovn/common/data/pki/cacert.pem (OK: Present)

[OVN Northbound Service]
/var/snap/microovn/common/data/pki/ovnnb-cert.pem (OK: Present)
/var/snap/microovn/common/data/pki/ovnnb-privkey.pem (OK: Present)

[OVN Southbound Service]
/var/snap/microovn/common/data/pki/ovnsb-cert.pem (OK: Present)
/var/snap/microovn/common/data/pki/ovnsb-privkey.pem (OK: Present)

[OVN Northd Service]
/var/snap/microovn/common/data/pki/ovn-northd-cert.pem (OK: Present)
/var/snap/microovn/common/data/pki/ovn-northd-privkey.pem (OK: Present)

[OVN Chassis Service]
/var/snap/microovn/common/data/pki/ovn-controller-cert.pem (OK: Present)
/var/snap/microovn/common/data/pki/ovn-controller-privkey.pem (OK: Present)
```

This command does not perform any certificate validation, it only ensures that if a service is available on the node, the file that should contain a certificate is in place.

Re-issue certificates

The **certificates reissue** command is used to interact with OVN services on the local host; it does not affect peer cluster members.

Important: Services must be running in order to be affected by the **certificates reissue** command. For example, running **certificates reissue ovnnb** on a member that does not run this service is expected to fail.

To re-issue a certificate for a single service:

```
microovn certificates reissue <ovn_service_name>
```

To re-issue certificates for all services, the **all** argument is supported:

```
microovn certificates reissue all
```

Valid service names can be discovered with the **--help** option:

```
microovn certificates reissue --help
```

Regenerate PKI for the cluster

The **certificates regenerate-ca** command is used to issue a new CA certificate and new certificates for every OVN service in the cluster:

```
microovn certificates regenerate-ca
```

This command replaces the current CA certificate and notifies all cluster members to re-issue certificates for all their services. The command's output will include evidence of successfully issued certificates for each cluster member.

Warning: A new certificate must be issued successfully for every service on every member. Any failure will result in subsequent communication errors for that service within the cluster.

Certificate lifecycle

Certificates that are automatically provisioned by MicroOVN have the following lifespans:

- CA certificate: 10 years
- OVN service certificate: 2 years

MicroOVN runs daily checks for certificate lifespan validity. When a certificate is within 10 days of expiration, it will be automatically renewed.

Upgrade from plaintext to TLS

Plaintext communication is used when MicroOVN is initially deployed with a snap revision of less than 111, and there's no way to automatically convert to encrypted communication. The following manual steps are needed to upgrade from plaintext to TLS:

1. ensure that all MicroOVN snaps in the cluster are upgraded to, at least, revision 111
2. run `microovn certificates regenerate-ca` on one of the cluster members
3. run `sudo snap restart microovn.daemon` on **all** cluster members. Allow commands to complete before proceeding to the next step.
4. run `sudo snap restart microovn.ovn-northd` on **all** cluster members

Once this is done, OVN API services throughout the cluster will start listening on TLS-secured ports. However, the process is not complete yet because OVN Southbound and Northbound database clusters themselves are not capable of automatically switching to TLS communication in existing clusters.

Manually switch OVN Northbound and Southbound clusters to TLS

Both database clusters need to be manually switched over by individually removing cluster members that use `tcp` connection and reconnecting them with `ssl`. This process technically replaces every member in the original cluster, but because we are doing it gradually, cluster data remains intact.

Let's assume that we have a 3 node cluster. We'll start with switching over the OVN Northbound cluster.

Preparation: We will be running commands on multiple nodes throughout this process, it is recommended to open a separate shell on each node and keep it open with following variables exported:

```
CONTROL_SOCKET=/var/snap/microovn/common/run/ovn/ovnnb_dbctl
DB=OVN_Northbound
DB_FILE=/var/snap/microovn/common/data/central/db/ovnnb_db.db
PORT=6643
```

1. Leave cluster on the node 1:

```
microovn.ovn-appctl -t $CONTROL_SOCKET cluster/leave $DB
```

2. Make sure that member properly left the cluster by inspecting cluster status on nodes 2 and 3 and ensuring that node 1 is no longer part of the cluster:

```
microovn.ovn-appctl -t /var/snap/microovn/common/run/ovn/ovnnb_dbctl cluster/status OVN_
↪Northbound
```

3. Clean up remaining DB files on node 1:

```
snap stop microovn.central
rm $DB_FILE
```

4. Rejoin the cluster with node 1, using `ssl` as protocol for local listening port. Notice that we will still use `tcp` as a protocol for remote cluster connection because no other node listens on `ssl` yet. This will get fixed automatically when other cluster members switch to `ssl`:

```
microovn.ovsdb-tool join-cluster $DB_FILE $DB ssl:<local_ip>:$PORT tcp:<node_2_ip>:$PORT
snap restart microovn.central
```

5. Monitor cluster, from node 1, as it converges to stable state. Use following command to monitor cluster until it indicates three members and field `Entries not yet applied` reaches 0:

```
microovn.ovn-appctl -t $CONTROL_SOCKET cluster/status $DB
```

Now that node 1 successfully transitioned to TLS we can repeat the same steps on node 2 and then on node 3. The only difference is in **4. step** where we will use protocol `ssl` and IP of a node 1 as last arguments for `microovn.ovsdb-tool` command. To save you some searching and replacing, here are the revised commands for the **4. step** to be used on node 2 and 3:

```
microovn.ovsdb-tool join-cluster $DB_FILE $DB ssl:<local_ip>:$PORT ssl:<node_1_ip>:$PORT
snap restart microovn.central
```

After all three nodes transitioned to TLS usage, you can once again inspect cluster status on any node:

```
microovn.ovn-appctl -t $CONTROL_SOCKET cluster/status $DB
```

to verify that all three cluster members are using `ssl` as their connection protocol.

This whole process needs to be repeated again for OVN Southbound cluster. Steps and commands are the same, just with different set of variables configured in the **Preparation** step:

```
CONTROL_SOCKET=/var/snap/microovn/common/run/ovn/ovnsb_db.ctl
DB=OVN_Southbound
DB_FILE=/var/snap/microovn/common/data/central/db/ovnsb_db.db
PORT=6644
```

Common issues

This section contains some well known or expected issues that you can encounter.

I'm getting failed to load certificates error

If you run commands like **microovn.ovn-sbctl** and you get complaints about missing certificates while the rest of the commands seem to work fine.

Example:

```
microovn.ovn-sbctl show
```

Example output:

```
2023-06-14T15:09:31Z|00001|stream_ssl|ERR|SSL_use_certificate_file:
↪error:80000002:system library::No such file or directory
2023-06-14T15:09:31Z|00002|stream_ssl|ERR|SSL_use_PrivateKey_file: error:10080002:BIO
↪routines::system lib
2023-06-14T15:09:31Z|00003|stream_ssl|ERR|failed to load client certificates from /var/
↪snap/microovn/common/data/pki/cacert.pem: error:0A080002:SSL routines::system lib
Chassis microovn-0
  hostname: microovn-0
  Encap geneve
    ip: "10.5.3.129"
    options: {csum="true"}
```

This likely means that your MicroOVN snap got upgraded to a version that supports TLS, but it requires some manual upgrade steps. See section [Upgrade from plaintext to TLS](#).

2.1.2 Downscaling the cluster

Impact

Downscaling can have an adverse effect on the availability and resiliency of the cluster, especially when a member is being removed that runs an OVN central service (OVN SB, OVN NB, OVN Northd).

OVN uses the [Raft consensus algorithm](#) for cluster management, which has a fault tolerance of up to $(N-1)/2$ members. This means that fault resiliency will be lost if a three-node cluster is reduced to two nodes.

Monitoring

You can watch logs on the departing member for indications of removal failures with:

```
snap logs -f microovn.daemon
```

Any issues that arise during the removal process will need to be resolved manually.

Remove a cluster member

To remove a cluster member:

```
microovn cluster remove <member_name>
```

The value of <member_name> is taken from the **Name** column in the output of the **cluster list** command.

Any chassis components (ovn-controller and ovs-vswitchd) running on the member will first be stopped and disabled (prevented from starting). For a member with central components present (microovn.central), the Northbound and Southbound databases will be gracefully removed.

Verification

Upon removal, check the state of OVN services to ensure that the member was properly removed.

```
# Check status of OVN SB cluster
microovn.ovn-appctl -t /var/snap/microovn/common/run/central/ovnsb_dbctl cluster/status_
↪ OVN_Southbound

# Check status of OVN NB cluster
microovn.ovn-appctl -t /var/snap/microovn/common/run/central/ovnnb_dbctl cluster/status_
↪ OVN_Northbound

# Check registered chassis
microovn.ovn-sbctl show
```

Data preservation

MicroOVN will back up selected data directories into the timestamped location /var/snap/microovn/common/backup_<timestamp>/. These backups will include:

- logs
- OVN database files
- OVS database file
- issued certificates and keys

2.1.3 Accessing logs

The *MicroOVN services* provide logs as part of their normal operation.

By default they are provided through the systemd journal, and can be accessed through the use of the `journalctl` or `snap logs` commands.

This is how you can access the logs of the `microovn.chassis` service using the `snap logs` command:

```
snap logs microovn.chassis
```

and using the `journalctl` command:

```
journalctl -u snap.microovn.chassis
```

This is how you can view a live log display for the same service using the `snap logs` command:

```
snap logs -f microovn.chassis
```

and using the `journalctl` command:

```
journalctl -f -u snap.microovn.chassis
```

Log files

Inside the `/var/snap/microovn/common/logs` directory you will find files for each individual service, however these will either be empty or not contain updated information, this is intentional.

On a fresh install the files are created, as a precaution, in the event a need arises for enabling *debug logging*. When upgrading MicroOVN, existing files will be retained, but not updated.

Debug logging

The Open vSwitch (OVS) and Open Virtual Network (OVN) daemons have a rich set of debug features, one of which is the ability to specify log levels for individual modules at run time.

A list of modules can be acquired through the `microovn.ovs-appctl` and `microovn.ovn-appctl` commands.

This is how to enable debug logging for the Open vSwitch `vswitchd` module:

```
microovn.ovs-appctl vlog/set vswitchd:file:dbg
```

This is how to enable debug logging for the Open Virtual Network `reconnect` module:

```
microovn.ovn-appctl vlog/set reconnect:file:dbg
```

For more details on how to configure logging, see [ovs-appctl manpage](#).

2.2 Tutorials

These tutorials provide step-by-step instructions for common goals. They do not assume special Linux knowledge nor any particular understanding of OVN.

2.2.1 Single-node

This tutorial shows how to install MicroOVN in the simplest way possible.

Caution: A single-node OVN cluster does not have any redundancy (service failover).

Install the software

Install MicroOVN on the designated node with the following command:

```
sudo snap install microovn
```

Initialise the cluster

```
microovn cluster bootstrap
```

Manage the cluster

You can interact with OVN using its native commands prefaced with the string `microovn..` For example, to show the contents of the OVN Southbound database:

```
microovn.ovn-sbctl show
```

2.2.2 Multi-node

This tutorial shows how to install a 3-node MicroOVN cluster.

One big advantage of a multi-node cluster is that it provides redundancy (service failover). A 3-node deployment can tolerate up to one node failure.

Requirements

You will need three (virtual or physical) machines that can communicate with each other over the network. They will be known here as `node-1`, `node-2`, and `node-3`.

Install the software

Install MicroOVN on **each** of the designated nodes with the following command:

```
sudo snap install microovn
```

Initialise the cluster

On **node-1**, initialise the cluster:

```
microovn cluster bootstrap
```

Generate access tokens

On **node-1**, generate access tokens for the other two nodes (cluster members). These will be needed to join these nodes to the cluster.

Let this token be for node-2:

```
microovn cluster add node-2
```

The output will be a special string such as: eyJuYW11Ijoibm9kZS0yIiwic2VjcmV0IjoiMzBlM...

Let this token be for node-3:

```
microovn cluster add node-3
```

Similarly, a string will be sent to the screen: eyJuYW11Ijoibm9kZS0zIiwic2VjcmV0IjoiZmZhY...

Complete the cluster

Join node-2 and node-3 to the cluster using their assigned access tokens.

On **node-2**:

```
microovn cluster join eyJuYW11Ijoibm9kZS0yIiwic2VjcmV0IjoiMzBlM...
```

On **node-3**:

```
microovn cluster join eyJuYW11Ijoibm9kZS0zIiwic2VjcmV0IjoiZmZhY...
```

Now all three nodes are joined to the cluster.

Manage the cluster

You can interact with OVN using its native commands prefaced with the string `microovn..` For example, to show the contents of the OVN Southbound database:

```
microovn.ovn-sbctl show
```

The cluster can be managed from any of its nodes.

2.3 Reference

MicroOVN reference material is specific to the MicroOVN project. It does not cover upstream OVN/OVS topics.

2.3.1 MicroOVN services

This page presents a list of all MicroOVN services. Their descriptions are for reference only - the user is not expected to interact directly with these services.

The status of all services is displayed by running:

```
snap services microovn
```

`microovn.central`

Warning: The `microovn.central` service is deprecated and will be removed in a future release.

This is a transitional service. Starting this service will start and enable multiple services:

- `microovn.ovn-ovsdb-server-nb`
- `microovn.ovn-ovsdb-server-sb`
- `microovn.ovn-northd`

However this service is not capable of stopping these child services so its usage is strongly discouraged. Users should use individual services instead.

`microovn.chassis`

This service maps directly to the `ovn-controller` daemon.

`microovn.daemon`

The main MicroOVN service/process that manages all the other processes. It also handles communication with other MicroOVN cluster members and provides an API for the `microovn` client command.

`microovn.ovn-ovsdb-server-nb`

This service maps directly to the OVN Northbound database/service.

`microovn.ovn-northd`

This service maps directly to the `ovn-northd` daemon.

`microovn.ovn-ovsdb-server-sb`

This service maps directly to the OVN Southbound database/service.

`microovn.refresh-expiring-certs`

This service is a recurring process that runs once a day between 02:00 and 02:30. It triggers TLS certification reissue for certificates that are nearing the expiration. For more information see the [certificates lifecycle](#).

`microovn.switch`

This services maps directly to the `ovs-vswitchd` daemon.

2.4 Developer's documentation

This section is dedicated to documentation targeted at MicroOVN developers and covers topics useful for code or documentation contributors.

2.4.1 Build and install MicroOVN from source

This how-to contains steps needed for building MicroOVN from its source code. This is useful, for example, if you want to contribute to the MicroOVN and you want to test your changes locally.

Build requirements

MicroOVN is distributed as a snap and the only requirements for building it are `Make` and `snappy`. You can install them with:

```
sudo apt install make
sudo snap install snappy --classic
```

Snappy requires LXD to build snaps. So if your system does not have LXD installed and initiated, you can check out either [LXD getting started guides](#) or go with following default setup:

```
sudo snap install lxd
lxd init --auto
```

Build MicroOVN

To build MicroOVN, go into the repository's root directory and run:

```
make
```

This will produce the `microovn.snap` file that can be then used to install MicroOVN on your system.

Install MicroOVN

Using the `microovn.snap` file created in the previous section, you can install MicroOVN in this way:

```
sudo snap install --dangerous ./microovn.snap
```

Note: If you are building latest MicroOVN from the main branch, it's possible that it's using a non-stable core snap as its base. In that case, you may get a message like this:

```
Ensure prerequisites for "microovn" are available (cannot install snap base "core24": no
↪snap revision available as specified)
```

In such a case, you will need to install the required core snap manually from the edge risk level. For example:

```
snap install core24 --edge
```

Then repeat the installation step.

You will also need to manually connect required plugs, as `snapped` won't do it automatically for locally installed snaps.

```
for plug in firewall-control \
    hardware-observe \
    hugepages-control \
    network-control \
    openvswitch-support \
    process-control \
    system-trace; do \
    sudo snap connect microovn:$plug;done
```

To verify that all the required plugs are correctly connected to their slots, you can run:

```
snap connections microovn
```

An example of correctly connected connected plugs would look like this:

| Interface | Plug | Slot | Notes |
|-------------------|----------------------------|---------------------------|--------|
| content | - | microovn:ovn-certificates | - |
| content | - | microovn:ovn-chassis | - |
| content | - | microovn:ovn-env | - |
| firewall-control | microovn:firewall-control | :firewall-control | manual |
| hardware-observe | microovn:hardware-observe | :hardware-observe | manual |
| hugepages-control | microovn:hugepages-control | :hugepages-control | manual |
| microovn | - | microovn:microovn | - |
| network | microovn:network | :network | - |

(continues on next page)

(continued from previous page)

| | | | |
|---------------------|------------------------------|----------------------|--------|
| network-bind | microovn:network-bind | :network-bind | - |
| network-control | microovn:network-control | :network-control | manual |
| openvswitch-support | microovn:openvswitch-support | :openvswitch-support | manual |
| process-control | microovn:process-control | :process-control | manual |
| system-trace | microovn:system-trace | :system-trace | manual |

And if the plugs are not connected, the output would look like this:

| Interface | Plug | Slot | Notes |
|---------------------|------------------------------|---------------------------|-------|
| content | - | microovn:ovn-certificates | - |
| content | - | microovn:ovn-chassis | - |
| content | - | microovn:ovn-env | - |
| firewall-control | microovn:firewall-control | - | - |
| hardware-observe | microovn:hardware-observe | - | - |
| hugepages-control | microovn:hugepages-control | - | - |
| microovn | - | microovn:microovn | - |
| network | microovn:network | :network | - |
| network-bind | microovn:network-bind | :network-bind | - |
| network-control | microovn:network-control | - | - |
| openvswitch-support | microovn:openvswitch-support | - | - |
| process-control | microovn:process-control | - | - |
| system-trace | microovn:system-trace | - | - |

2.4.2 Run MicroOVN tests

MicroOVN has two types of tests, linter checks and functional tests and this page will show how to run them.

Linter checks

Linting is currently applied only to the tests themselves, not the main codebase. The prerequisites for running linter are:

- make
- shellcheck

You can install them with:

```
sudo apt install make shellcheck
```

To perform linting, go into the repository's root directory and run:

```
make check-lint
```

Functional tests

These tests build the MicroOVN snap and use it to deploy the OVN cluster in LXD containers. This cluster is then used for running functional test suites.

Satisfy the test requirements

There is no need to run tests in dedicated VMs or in isolated environments as all functional tests run inside containers and no changes are made to the host running them.

MicroOVN needs to be built prior to running the functional tests. See the [Build MicroOVN](#) page.

Secondly, ensure that you have installed [Bash Automated Testing System \(BATS\)](#), a software dependency. Due to the reliance on its latest features, MicroOVN uses BATS directly from its source. If you cloned the MicroOVN repository with submodules (using `--recurse-submodules` flag), you are all set and you will have the following **non-empty** directories:

- `.bats/bats-assert/`
- `.bats/bats-core/`
- `.bats/bats-support/`

If they are empty, you can fetch the submodules with:

```
git submodule update --init --recursive
```

Run functional tests

Once you have your environment set up, running tests is just a matter of invoking the appropriate make target. To run all available test suites, use the `check-system` make target:

```
make check-system
```

To run individual test suites you can execute:

```
make tests/<name_of_the_test_suite>.bats
```

Tip: If your hardware can handle it, you can run test suites in parallel by supplying `make` with `-j` argument (e.g. `make check-system -j4`). To avoid interleaving output from these parallel test suites, you can specify the `-O` argument as well.

Clean up

Functional test suites will attempt to clean up their containers. However, if a test crashes, or if it's forcefully killed, you may need to do some manual cleanup.

If you suspect that tests did not clean up properly, you can list all containers with:

```
lxc list
```

Any leftover containers will be named according to: `microovn-<test_suite_name>-<number>`. You can remove them with:

```
lxc delete --force <container_name>
```