Access logs

Logs are our first go-to when something goes wrong. Multipass is comprised of a daemon process (service) and the CLI and GUI clients, each of them reporting on their own health.

The `multipass` command accepts the `--verbose` option (`-v` for short), which can be repeated to go from the default (*error*) level through *warning*, *info*, *debug* up to *trace*.

> See also: Logging levels, Configure Multipass's default logging level

We use the underlying platform's logging facilities to ensure you get the familiar behaviour wherever you are.

Linux macOS Windows

On Windows, the Event system is used and Event Viewer lets you access them. Our logs are currently under "Windows Logs/Application", where you can filter by "Multipass" Event source. You can then export the selected events to a file.

Logs from the installation and uninstall process can be found under `%APPDATA%\Local\Temp`. Sort the contents of the directory by "Date Modified" to bring the newest files to the top. The name of the file containing the logs follows the pattern `MSI[0-9a-z].LOG`.

The Multipass GUI produces its own logs, that can be found under `%APPDATA%\com.canonical\Multipass GUI\multipass_gui.log`

Last updated on Feb 23, 2025

↑ Back to top

Access logs

Logs are our first go-to when something goes wrong. Multipass is comprised of a daemon process (service) and the CLI and GUI clients, each of them reporting on their own health.

The `multipass` command accepts the `--verbose` option (`-v` for short), which can be repeated to go from the default (*error*) level through *warning*, *info*, *debug* up to *trace*.

> See also: Logging levels, Configure Multipass's default logging level

We use the underlying platform's logging facilities to ensure you get the familiar behaviour wherever you are.

Linux macOS Windows

On macOS, log files are stored in `/Library/Logs/Multipass`, where `multipassd.log` has the daemon messages. You will need `sudo` to access it.

The Multipass GUI produces its own logs, that can be found under `~/Library/Application\ Support/com.canonical.multipassGui/multipass_gui.log`

Last updated on Feb 23, 2025

↑ Back to top

Access logs

Logs are our first go-to when something goes wrong. Multipass is comprised of a daemon process (service) and the CLI and GUI clients, each of them reporting on their own health.

The `multipass` command accepts the `--verbose` option (`-v` for short), which can be repeated to go from the default (*error*) level through *warning*, *info*, *debug* up to *trace*.

> See also: Logging levels, Configure Multipass's default logging level

We use the underlying platform's logging facilities to ensure you get the familiar behaviour wherever you are.

Linux macOS Windows

On Linux, `systemd-journald` is used, integrating with the de-facto standard for this on modern Linux systems.

To access the daemon (and its child processes') logs:

```
journalctl --unit 'snap.multipass*'
```

The Multipass GUI produces its own logs, that can be found under `~/snap/multipass/current/data/multipass_gui/multipass_gui.log`

Last updated on Feb 23, 2025

Add a network to an existing instance

> See also
>
> networks, get, set, local.<instance-name>.bridged

This guide explains how to bridge an existing Multipass instance with the available networks.

> Caution
>
> This feature is available starting from Multipass version 1.14.

First, you need to select a Multipass-wide preferred network to bridge with (you can always change it later). To do so, list all available networks using the `multipass networks` command. The output will be similar to the following:

```
Name      Type       Description
br-eth0   bridge     Network bridge with eth0
br-eth1   bridge     Network bridge with eth1
eth0      ethernet   Ethernet device
eth1      ethernet   Ethernet device
lxdbr0    bridge     Network bridge
mpbr0     bridge     Network bridge for Multipass
virbr0    bridge     Network bridge
```

Set the preferred network (for example, `eth0`) using the `set` command:

```
multipass set local.bridged-network=eth0
```

Before bridging the network, you need to stop the instance (called `ultimate-grosbeak` in our example) using the `stop` command:

```
multipass stop ultimate-grosbeak
```

You can now ask Multipass to bridge your preferred network using the `local.<instance-name>.bridged` setting:

```
multipass set local.ultimate-grosbeak.bridged=true
```

To add further networks, update the preferred bridge and repeat:

```
multipass set local.bridged-network=eth1
multipass set local.ultimate-grosbeak.bridged=true
```

Use the `get` command to check whether an instance is bridged with the currently configured preferred network:

```
multipass get local.ultimate-grosbeak.bridged
```

After following the recipe above, the result should be `true`.

Now, `start` the instance.

```
multipass start ultimate-grosbeak
```


Last updated on Apr 17, 2025

Authenticate clients with the Multipass service

> See also: `authenticate` , local.passphrase, Service

Multipass requires clients to be authenticated with the service before allowing commands to complete. The installing user is automatically authenticated.

Setting the passphrase

The administrator needs to set a passphrase for clients to authenticate with the Multipass service. The client setting the passphrase will need to already be authenticated.

There are two ways to proceed:

- Set the passphrase with an echoless interactive entry, where the passphrase is hidden from view:

  ```
  multipass set local.passphrase
  ```

  The system will then prompt you to enter a passphrase:

  ```
  Please enter passphrase:
  Please re-enter passphrase:
  ```

- Set the passphrase in the command line, where the passphrase is visible:

  ```
  multipass set local.passphrase=foo
  ```

Authenticating the client

A client that is not authorised to connect to the Multipass service will fail when running `multipass` commands. An error will be displayed when this happens.

For example, if you try running the `multipass list` command:

```
list failed: The client is not authenticated with the Multipass service.
Please use 'multipass authenticate' before proceeding.
```

At this time, the client will need to provide the previously set passphrase. This can be accomplished in two ways:

- Authenticate with an echoless interactive entry, where the passphrase is hidden from view:

  ```
  multipass authenticate
  ```

  The system will prompt you to enter the passphrase:

  ```
  Please enter passphrase:
  ```

- Authenticate in the command line, where the passphrase is visible:

  ```
  multipass authenticate foo
  ```

Troubleshooting

Here you can find solutions and workarounds for common issues that may arise.

The client cannot be authorised and the passphrase cannot be set

It is possible that another client that is privileged to connect to the Multipass socket will connect first and make it seemingly impossible to set the `local.passphrase` and also `authorize` the client with the service. This usually occurs when Multipass is installed as root/admin but the client is run as another user, or vice versa.

If this is the case, you will see something like the following when you run:

- `multipass list`

```
list failed: The client is not authenticated with the Multipass service.
Please use 'multipass authenticate' before proceeding.
```

- `multipass authenticate`

```
Please enter passphrase:
authenticate failed: Passphrase is not set. Please `multipass set
local.passphrase` with a trusted client.
```

- `multipass set local.passphrase`

```
Please enter passphrase:
Please re-enter passphrase:
set failed: The client is not authenticated with the Multipass service.
Please use 'multipass authenticate' before proceeding.
```

This may not even work when using `sudo`.

The following workaround should help get out of this situation:

```
cat ~/snap/multipass/current/data/multipass-client-certificate/multipass_ce
snap restart multipass
```

You may need `sudo` with this last command: `sudo snap restart multipass`.

At this point, your client should be authenticated with the Multipass service.

Build Multipass images with Packer

> **Note**
>
> Custom images are only supported on Linux.

Packer is a utility that lets you (re)build images to run in a variety of environments. Multipass can run those images too, provided some requirements are met (namely, the image has to boot on the hypervisor in use, and cloud-init needs to be available).

Setting up the build directory

The easiest way is to start from an existing Ubuntu Cloud Image, and the base project setup follows (you can click on the filenames to see their contents, `meta-data` is empty on purpose):

```
├── cloud-data
|   ├── meta-data
|   └── <a href="https://paste.ubuntu.com/p/6vbtNXttqZ/">user-data</a>
└── <a href="https://pastebin.ubuntu.com/p/nJsGtWk2N3/">template.json</a>

1 directory, 3 files
```

To specify a different image or image type, modify the `iso_checksum` and `iso_url` fields within `template.json`

Building the image

You will need to install QEMU and Packer (e.g. `sudo apt install qemu packer`), and from there you can run the following commands:

```
packer build template.json
multipass launch file://$PWD/output-qemu/packer-qemu --disk 5G
```

Now, shell into the new instance that was created, for example:

```
multipass shell tolerant-hammerhead
```

Customising the image

Now the above works for you, delete the test instance with `multipass delete --purge tolerant-hammerhead` and edit the following section in the `template.json` file:

```
        {
            "type": "shell",
            "inline": ["echo Your steps go here."]
        },
```

Anything you do here will be reflected in the resulting image. You can install packages, configure services, anything you can do on a running system. You'll need `sudo` (passwordless) for anything requiring admin privileges, or you could add this to the above provisioner to run the whole script privileged:

```
            "execute_command": "sudo sh -c '{{ .Vars }} {{ .Path }}'",
```

Next steps

Go to Packer's documentation to learn about the QEMU builder, other provisioners and their configurations as well as everything else that might come in handy. Alternatively, you could extend `user-data` with other `cloud-init` modules to provision your image.

Join the discussion on the Multipass forum and let us know about your images!

Command-line interface

> See also: Instance, Service

The **`multipass`** CLI (command line interface) client is used to communicate with the Multipass service to create, manage, and interact with Multipass instances using various sub-commands.

You can use `multipass help <command>` to display more information on the purpose and available options of each command.

- alias
- aliases
- authenticate
- clone
- delete
- exec
- find
- get
- help
- info
- launch
- list
- mount
- networks
- prefer
- purge
- recover
- restart
- restore
- set
- shell
- snapshot
- start
- stop
- suspend
- transfer
- umount
- unalias
- version

Back to top

Last updated on Feb 20, 2025

Configure Multipass's default logging level

> See also: Logging levels

This document demonstrates how to configure the default logging level of the Multipass service. Changing the logging level can be useful, for example, if you want to decrease the size of logging files or get more detailed information about what the daemon is doing. Logging levels can be set to one of the following: `error`, `warning`, `info`, `debug`, or `trace`, with case sensitivity.

Changing the default logging level

Linux macOS Windows

First, stop the Multipass daemon:

```
sudo snap stop multipass
```

After that, create the override config file, replacing `<level>` with your desired logging level:

```
sudo mkdir /etc/systemd/system/snap.multipass.multipassd.service.d/
sudo tee /etc/systemd/system/snap.multipass.multipassd.service.d/override.c
[Service]
ExecStart=
ExecStart=/usr/bin/snap run multipass.multipassd --verbosity <level>
EOF
sudo systemctl daemon-reload
```

Finally, start the Multipass daemon:

Last updated on Feb 19, 2025

latest

Configure static IPs

> See also: Instance

This guide explains how to create instances with static IPs in a new network, internal to the host. With this approach, instances get an extra IP that does not change with restarts. By using a separate, local network we avoid any IP conflicts. Instances retain the usual default interface with a DHCP-allocated IP, which gives them connectivity to the outside.

Step 1: Create a Bridge

The first step is to create a new bridge/switch with a static IP on your host.

This is beyond the scope of Multipass but, as an example, here is how this can be achieved with NetworkManager on Linux (e.g. on Ubuntu Desktop):

```
nmcli connection add type bridge con-name localbr ifname localbr \
    ipv4.method manual ipv4.addresses 10.13.31.1/24
```

This will create a bridge named `localbr` with IP `10.13.31.1/24`. You can see the new device and address with `ip -c -br addr show dev localbr`. This should show:

```
localbr          DOWN           10.13.31.1/24
```

You can also run `multipass networks` to confirm the bridge is available for Multipass to connect to.

Step 2: Launch an instance with a manual network

Next we launch an instance with an extra network in manual mode, connecting it to this bridge:

```
multipass launch --name test1 --network name=localbr,mode=manual,mac="52:54
```

You can also leave the MAC address unspecified (just `--network name=localbr,mode=manual`). If you do so, Multipass will generate a random MAC for you, but you will need to retrieve it in the next step.

Step 3: Configure the extra interface

We now need to configure the manual network interface inside the instance. We can achieve that using Netplan. The following command plants the required Netplan configuration file in the instance:

```
multipass exec -n test1 -- sudo bash -c 'cat << EOF > /etc/netplan/10-custo
network:
    version: 2
    ethernets:
        extra0:
            dhcp4: no
            match:
                macaddress: "52:54:00:4b:ab:cd"
            addresses: [10.13.31.13/24]
EOF'
```

The IP address needs to be unique and in the same subnet as the bridge. The MAC address needs to match the extra interface inside the instance: either the one provided in Step 2 above or the one Multipass generated (you can find it with `ip link`).

If you want to set a different name for the interface, you can add a `set-name` property.

Step 4: Apply the new configuration

We now tell Netplan to apply the new configuration inside the instance:

```
multipass exec -n test1 -- sudo netplan apply
```

You may also use `netplan try`, to have the outcome reverted if something goes wrong.

Step 5: Confirm that it works

You can confirm that the new IP is present in the instance with Multipass:

```
multipass info test1
```

The command above should show two IPs, the second of which is the one we just configured (`10.13.31.13`). You can use `ping` to confirm that it can be reached from the host:

```
ping 10.13.31.13
```

Conversely, you can also ping from the instance to the host:

```
multipass exec -n test1 -- ping 10.13.31.1
```

Step 6: More instances

If desired, repeat steps 2-5 with different names/MACs/IP terminations (e.g. `10.13.31.14`) to launch other instances with static IPs in the same network. You can ping from one instance to another to confirm that they are connected. For example:

```
multipass exec -n test1 -- ping 10.13.31.14
```

Conclusion

You have now created a small internal network, local to your host, with Multipass instances that you can reach on the same IP across reboots. Instances still have the default NAT-ed network, which they can use to reach the outside world. You can combine this with other networks if you want to (e.g. for bridging).

Last updated on Feb 20, 2025

Configure where Multipass stores external data

This document demonstrates how to configure the location where Multipass stores instances, caches images, and other data. Configuring a new storage location can be useful, for example, if you need to free up storage space on your boot partition.

Configuring a new storage location

> **Caution**
>
> **Caveats:**
>
> - Multipass will not migrate your existing data; this article explains how to do it manually. If you do not transfer the data, you will have to re-download any Ubuntu images and reinitialise any instances that you need.
> - When uninstalling Multipass, the uninstaller will not remove data stored in custom locations, so you'll have to deleted it manually.

Linux macOS Windows

First, stop the Multipass daemon:

```
sudo snap stop multipass
```

Since Multipass is installed using a strictly confined snap, it is limited on what it can do or access on your host. Depending on where the new storage directory is located, you will need to connect the respective interface to the Multipass snap. Because of snap confinement, this directory needs to be located in either `/home` (connected by default) or one of the removable mounts points (`/mnt` or `/media`). To connect the removable mount points, use the command:

```
sudo snap connect multipass:removable-media
```

Create the new directory in which Multipass will store its data:

```
mkdir -p <path>
sudo chown root <path>
```

After that, create the override config file, replacing `<path>` with the absolute path of the directory created above.

```
sudo mkdir /etc/systemd/system/snap.multipass.multipassd.service.d/
sudo tee /etc/systemd/system/snap.multipass.multipassd.service.d/override.co
[Service]
Environment=MULTIPASS_STORAGE=<path>
EOF
```

The output at this point will be:

```
[Service]
Environment=MULTIPASS_STORAGE=<path>
```

Then, instruct `systemd` to reload the daemon configuration files:

```
sudo systemctl daemon-reload
```

Now you can transfer the data from its original location to the new location:

```
sudo cp -r /var/snap/multipass/common/data/multipassd <path>/data
sudo cp -r /var/snap/multipass/common/cache/multipassd <path>/cache
```

You also need to edit the following configuration files so that the specified paths point to the new Multipass storage directory, otherwise your instances will fail to start:

- `multipass-vm-instances.json` : Update the absolute path of the instance images in the “arguments” key for each instance.
- `vault/multipassd-instance-image-records.json` : Update the “path” key for each instance.

Finally, start the Multipass daemon:

```
sudo snap start multipass
```

You can delete the original data at your discretion, to free up space:

```
sudo rm -rf /var/snap/multipass/common/data/multipassd
sudo rm -rf /var/snap/multipass/common/cache/multipassd
```

Reverting back to the default location

Linux macOS Windows

Stop the Multipass daemon:

```
sudo snap stop multipass
```

Although not required, to make sure that Multipass does not have access to directories that it shouldn't, you can disconnect the respective interface depending on where the custom storage location was set (see Configuring a new storage location above). For example, to disconnect the removable mounts points (`/mnt` or `/media`), run:

```
sudo snap disconnect multipass:removable-media
```

Then, remove the override config file:

```
sudo rm /etc/systemd/system/snap.multipass.multipassd.service.d/override.co
sudo systemctl daemon-reload
```

Now you can transfer your data from the custom location back to its original location:

```
sudo cp -r <path>/data /var/snap/multipass/common/data/multipassd
sudo cp -r <path>/cache /var/snap/multipass/common/cache/multipassd
```

Finally, start the Multipass daemon:

```
sudo snap start multipass
```

You can delete the data from the custom location at your discretion, to free up space:

```
sudo rm -rf <path>
```

Create an instance

> See also: launch , Instance

This document demonstrates various ways to create an instance with Multipass. While every method is a one-liner involving the command `multipass launch` , each showcases a different option that you can use to get exactly the instance that you want.

Create an instance

> See also: launch , info

To create an instance with Multipass, run the command `multipass launch` . This launches a new instance, which is randomly named; for example:

```
...
Launched: keen-yak
```

In particular, when we run `multipass info keen-yak` , we find out that it is an Ubuntu LTS release, namely 18.04, with 1GB RAM, 1 CPU, 5GB of disk:

```
Name:           keen-yak
State:          RUNNING
IPv4:           10.140.94.253
Release:        Ubuntu 18.04.1 LTS
Image hash:     d53116c67a41 (Ubuntu 18.04 LTS)
CPU(s):         1
Load:           0.00 0.12 0.18
Disk usage:     1.1G out of 4.7G
Memory usage:   71.6M out of 985.4M
```

Create an instance with a specific image

> See also: `find`, `launch <image>`, `info`

To find out which images are available, run `multipass find`. Here's a sample output:

```
Image                       Aliases           Version          Description
20.04                       focal             20240821         Ubuntu 20.04
22.04                       jammy             20240912         Ubuntu 22.04
24.04                       noble,lts         20240911         Ubuntu 24.04

Blueprint                   Aliases           Version          Description
anbox-cloud-appliance                         latest           Anbox Cloud
charm-dev                                     latest           A developmen
docker                                        0.4              A Docker env
jellyfin                                      latest           Jellyfin is
minikube                                      latest           minikube is
ros-noetic                                    0.1              A developmen
ros2-humble                                   0.1              A developmen
```

To launch an instance with a specific image, include the image name or alias in the command, for example `multipass launch jammy`:

```
...
Launched: tenacious-mink
```

`multipass info tenacious-mink` confirms that we've launched an instance of the selected image.

```
Name:           tenacious-mink
State:          Running
Snapshots:      0
IPv4:           192.168.64.22
Release:        Ubuntu 22.04.5 LTS
Image hash:     e898c1c93b32 (Ubuntu 22.04 LTS)
CPU(s):         1
Load:           0.00 0.02 0.01
Disk usage:     1.6GiB out of 4.8GiB
Memory usage:   149.5MiB out of 962.2MiB
Mounts:         --
```

Create an instance with a custom name

> See also: `launch --name`

To launch an instance with a specific name, add the `--name` option to the command line; for example `multipass launch kinetic --name helpful-duck`:

```
...
Launched: helpful-duck
```

Create an instance with custom CPU number, disk and RAM

> See also: `launch --cpus --disk --memory`

You can specify a custom number of CPUs, disk and RAM size using the `--cpus`, `--disk` and `--memory` arguments, respectively. For example:

```
multipass launch --cpus 4 --disk 20G --memory 8G
```

Create an instance with primary status

> See also: `launch --name primary`

An instance can obtain the primary status at creation time if its name is `primary`:

```
multipass launch kinetic --name primary
```

For more information, see How to use the primary instance.

Create an instance with multiple network interfaces

> See also: `launch --network`

Multipass can create instances with additional network interfaces using the `multipass launch` command with the `--network` option. This is complemented by the `networks` command, that you can use to find available host networks to bridge with.

This feature is only supported for images with `cloud-init` support for v2 network config, which in turn requires netplan to be installed, meaning that you'll require Ubuntu 17.10 and Ubuntu Core 16 (except `snapcraft:core16`) or later. More specifically, this feature is only supported in the following scenarios:

- on Linux, with QEMU (*from Multipass 1.15 onward*)
- on Windows, with both Hyper-V and VirtualBox
- on macOS, with the QEMU and VirtualBox drivers

The `--network` option can be given multiple times to request multiple network interfaces beyond the default one, which is always present. Each time you add the `--network` option you also need to provide an argument specifying the properties of the desired interface:

- `name` - This is the only required value, used to identify the host network to connect the instance's device to (see `networks` for possible values).
- `mode` - Either `auto` (default) or `manual`; with `auto`, the instance will attempt to configure the network automatically.
- `mac` - Custom MAC address to use for the device.

These properties can be specified in the format `<key>=<value>` but a simpler form with only `<name>` is available for the most common use case. Here is an example:

```
multipass launch --network en0 --network name=bridge0,mode=manual
```

You can inspect the IP addresses assigned to the network interfaces of the new instance ("upbeat-whipsnake) on the system using the command:

```
multipass exec upbeat-whipsnake -- ip -br address show scope global
```

Sample output:

```
enp0s3           UP             10.0.2.15/24
enp0s8           UP             192.168.1.146/24
enp0s9           DOWN
```

Last, you can run `ping -c1 192.168.1.146` from the same network to verify that the IP can be reached:

```
PING 192.168.1.146 (192.168.1.146): 56 data bytes
64 bytes from 192.168.1.146: icmp_seq=0 ttl=64 time=0.378 ms
...
```

In the example above, we got the following interfaces inside the instance:

- `enp0s3` - The default interface, that the instance can use to reach the outside world and that Multipass uses to communicate with the instance.
- `enp0s8` - The interface that is connected to `en0` on the host and that is automatically configured.
- `enp0s9` - The interface that is connected to `bridge0` on the host, ready for manual configuration.

Routing

Extra interfaces are configured with a higher metric (200) than the default one (100). So, by default the instance will only route through them if they're a better match for the destination IP (see Wikipedia | Longest_prefix-match.

For example, if the command `multipass exec upbeat-whipsnake -- ip route` returns the following routing table:

```
default via 10.0.2.2 dev enp0s3 proto dhcp src 10.0.2.15 metric 100
default via 192.168.1.1 dev enp0s8 proto dhcp src 192.168.1.146 metric 200
10.0.2.0/24 dev enp0s3 proto kernel scope link src 10.0.2.15
10.0.2.2 dev enp0s3 proto dhcp scope link src 10.0.2.15 metric 100
192.168.1.0/24 dev enp0s8 proto kernel scope link src 192.168.1.146
192.168.1.1 dev enp0s8 proto dhcp scope link src 192.168.1.146 metric 200
```

you can then explore how specific IPs are routed:

```
multipass exec upbeat-whipsnake -- ip route get 91.189.88.181
```

In this case, for example:

```
91.189.88.181 via 10.0.2.2 dev enp0s3 src 10.0.2.15 uid 1000
    cache
```

Bridging

On Linux, when trying to connect an instance network to an ethernet device on the host, Multipass will offer to create the required bridge.

First, run the `multipass networks` command; for example:

```
Name            Type      Description
eth0            ethernet  Ethernet device
lxdbr0          bridge    Network bridge
mpbr0           bridge    Network bridge for Multipass
virbr0          bridge    Network bridge
```

Then, select an ethernet device and launch a new instance requesting to connect to it, for example via `multipass launch --network eth0`. The output will be:

```
Multipass needs to create a bridge to connect to eth0.
This will temporarily disrupt connectivity on that interface.

Do you want to continue (yes/no)?
```

However, Multipass requires `NetworkManager` to achieve this. On installations that do not have `NetworkManager` installed (e.g. Ubuntu Server), the user can still create a bridge by other means and pass that to Multipass. For instance, this configuration snippet achieves that with Netplan:

```
network:
  bridges:
    mybridge:
      dhcp4: true
      interfaces:
        - eth0
```

That goes somewhere in `/etc/netplan/` (e.g. `/etc/netplan/50-custom.yaml`). After a successful `netplan try` or `netplan apply`, Multipass will show the new bridge with the `networks` command and instances can be connected to it:

```
multipass launch --network mybridge
```

Another option is to install `NetworkManager`, but other network handlers need to be deactivated to avoid conflicts and make the new bridges permanent.

Create an instance with a custom DNS

In some scenarios the default of using the system-provided DNS will not be sufficient. When that's the case, you can use the `--cloud-init` option to the `launch` command, or modify the networking configuration after the instance started.

The `--cloud-init` approach

> See also: `launch --cloud-init`

To use a custom DNS in your instances, you can use this `cloud-init` snippet:

```
#cloud-config
bootcmd:
- printf "[Resolve]\nDNS=8.8.8.8" > /etc/systemd/resolved.conf
- [systemctl, restart, systemd-resolved]
```

Replace `8.8.8.8` with whatever your preferred DNS server is. You can then launch the instance using the following command:

```
multipass launch --cloud-init systemd-resolved.yaml
```

The Netplan approach

After the instance booted, you can modify the `/etc/netplan/50-cloud-init.yaml` file, adding the `nameservers` entry:

```
network:
  ethernets:
    ens3:
      dhcp4: true
      match:
        macaddress: 52:54:00:fe:52:ee
      set-name: ens3
      nameservers:
        search: [mydomain]
        addresses: [8.8.8.8]</b>
```

You can then test it with the command `sudo netplan try`:

```
Do you want to keep these settings?

Press ENTER before the timeout to accept the new configuration

Changes will revert in 120 seconds
...
```

Last updated on Apr 23, 2025

GUI client

> See also: Instance, Service, Settings

> **Caution**
>
> The GUI was introduced in Multipass version 1.14. It is still in its MVP (Minimum Viable Product) stage, so it is likely to see some changes in design as it evolves.

The Multipass GUI (Graphical User Interface) is an application that acts as a client for interacting with the Multipass service. It aims to make the process of managing instances easier for users who do not want to interact with the CLI (Command Line Interface) client. You can launch the GUI either using your system's application launcher or by running `multipass.gui` in a terminal.

> For more information on GUI logs, see How to access logs.

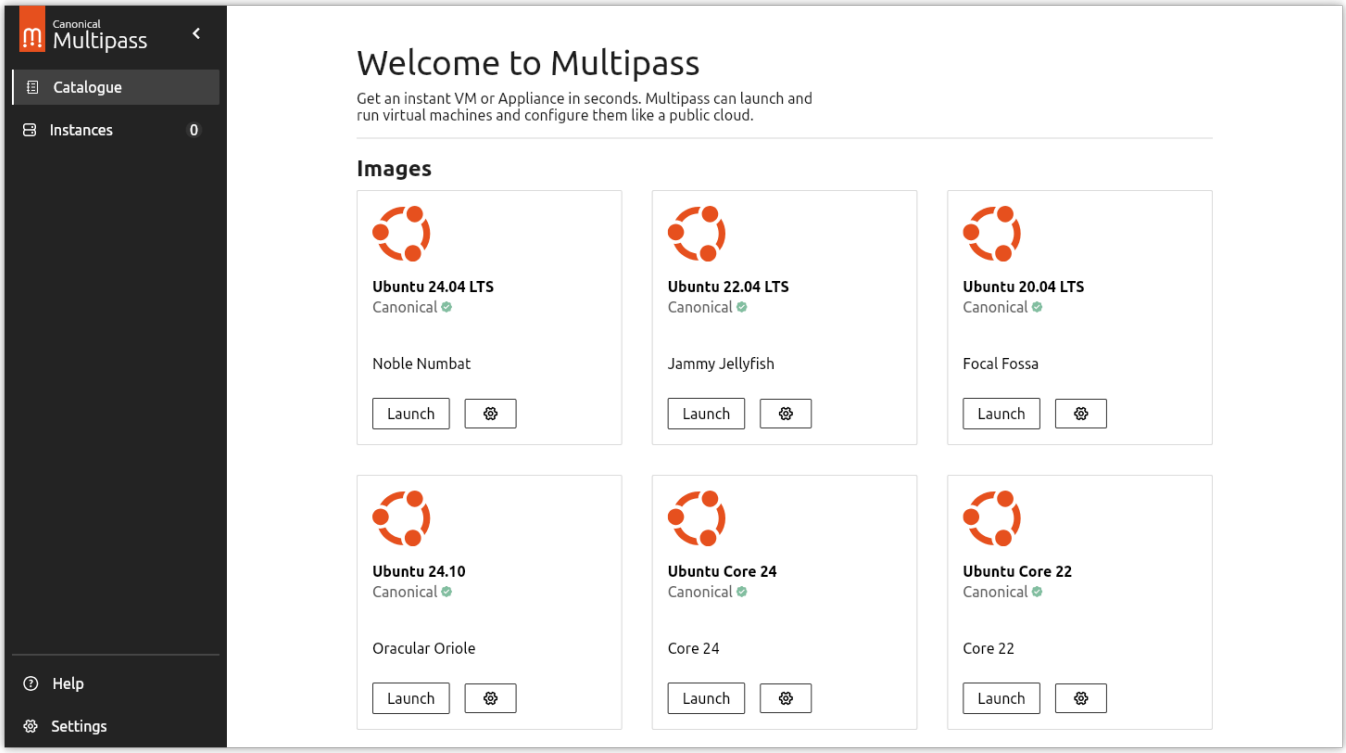
As of now, the Multipass GUI provides the following set of capabilities, grouped under four tabs:

- Catalogue tab
- Instances tab
- Settings tab

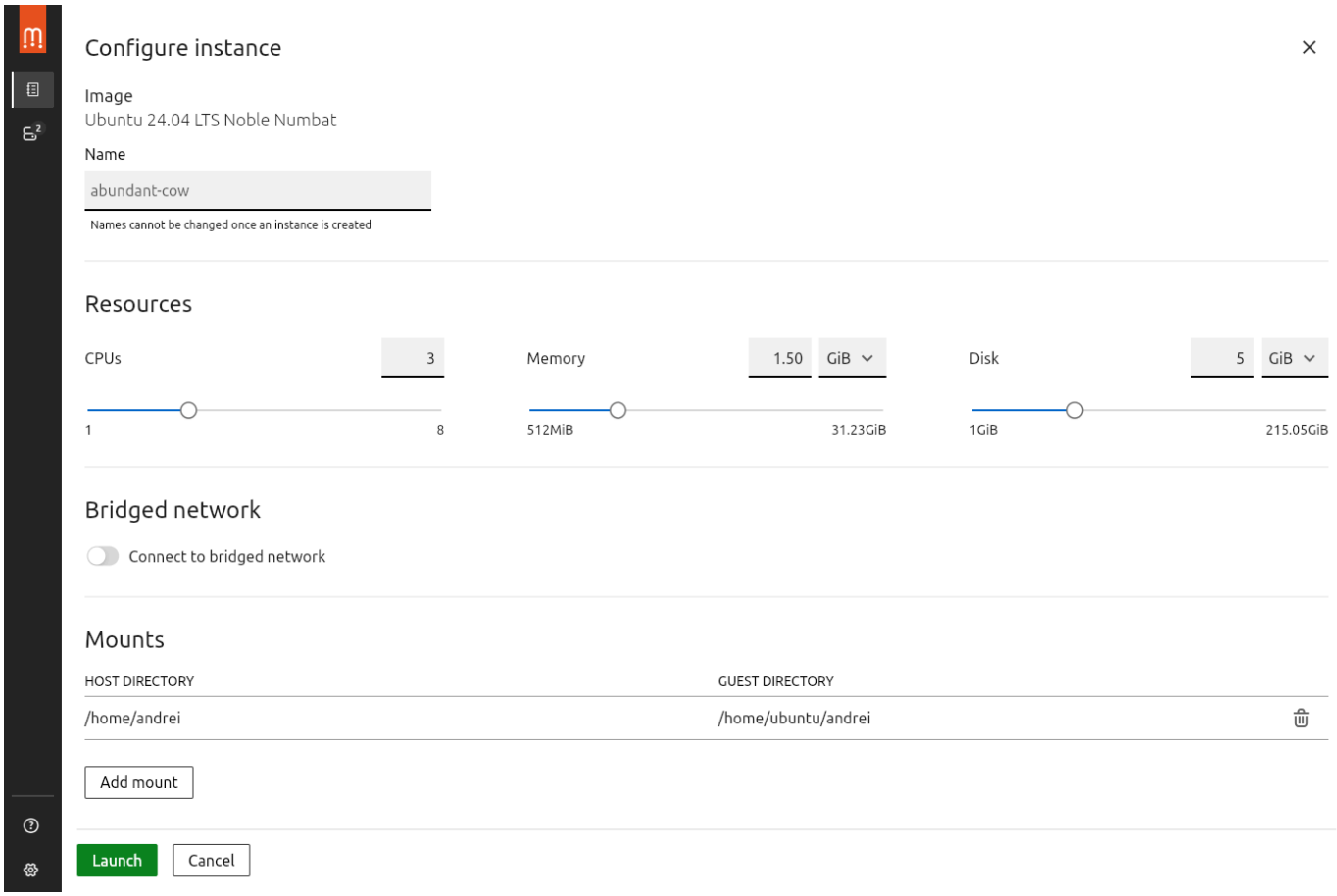
as well as a Tray icon menu.

Catalogue tab

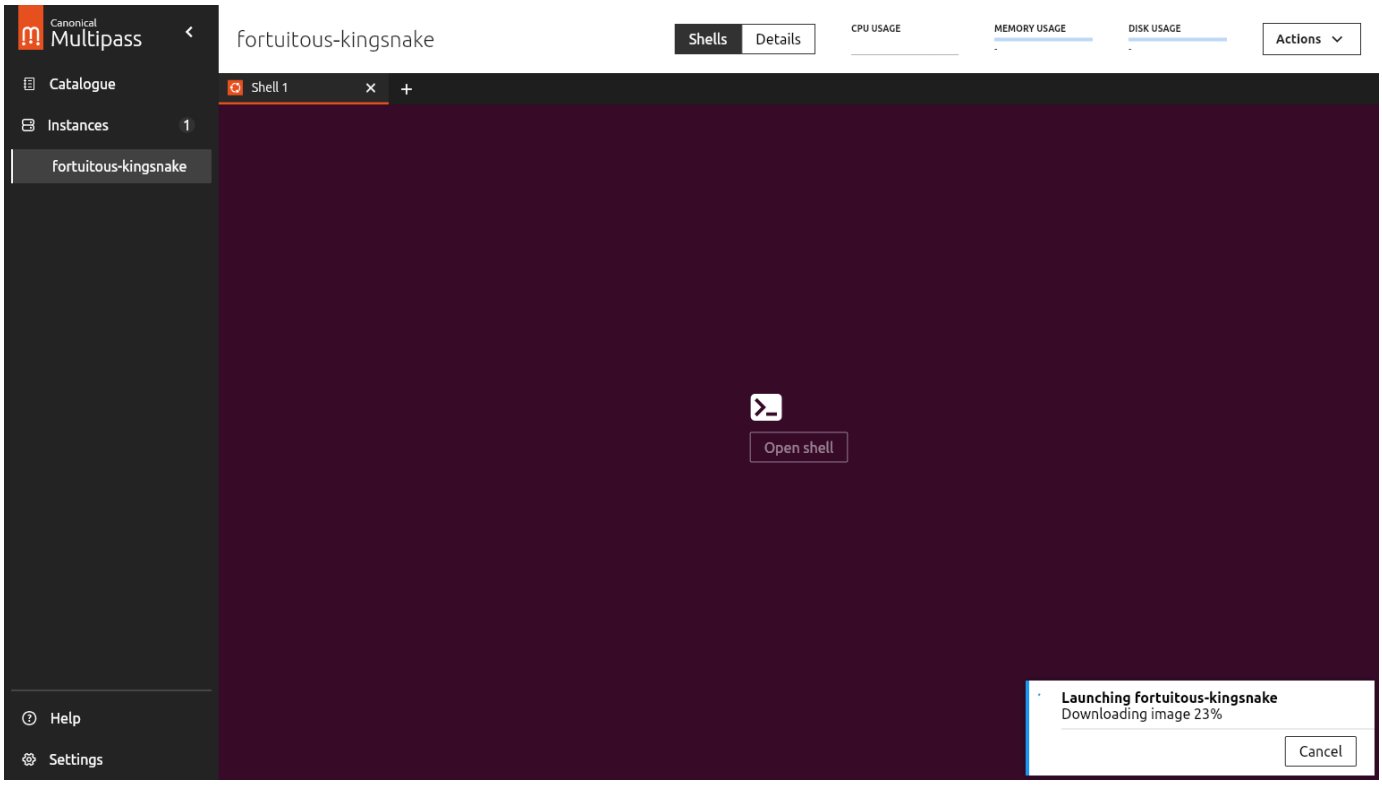
Here you can browse the available Ubuntu images. The output is equivalent to the one given by `multipass find --only-images`.



You can configure an instance's launch options, specifying parameters such as its name, allocated resources and mounts.

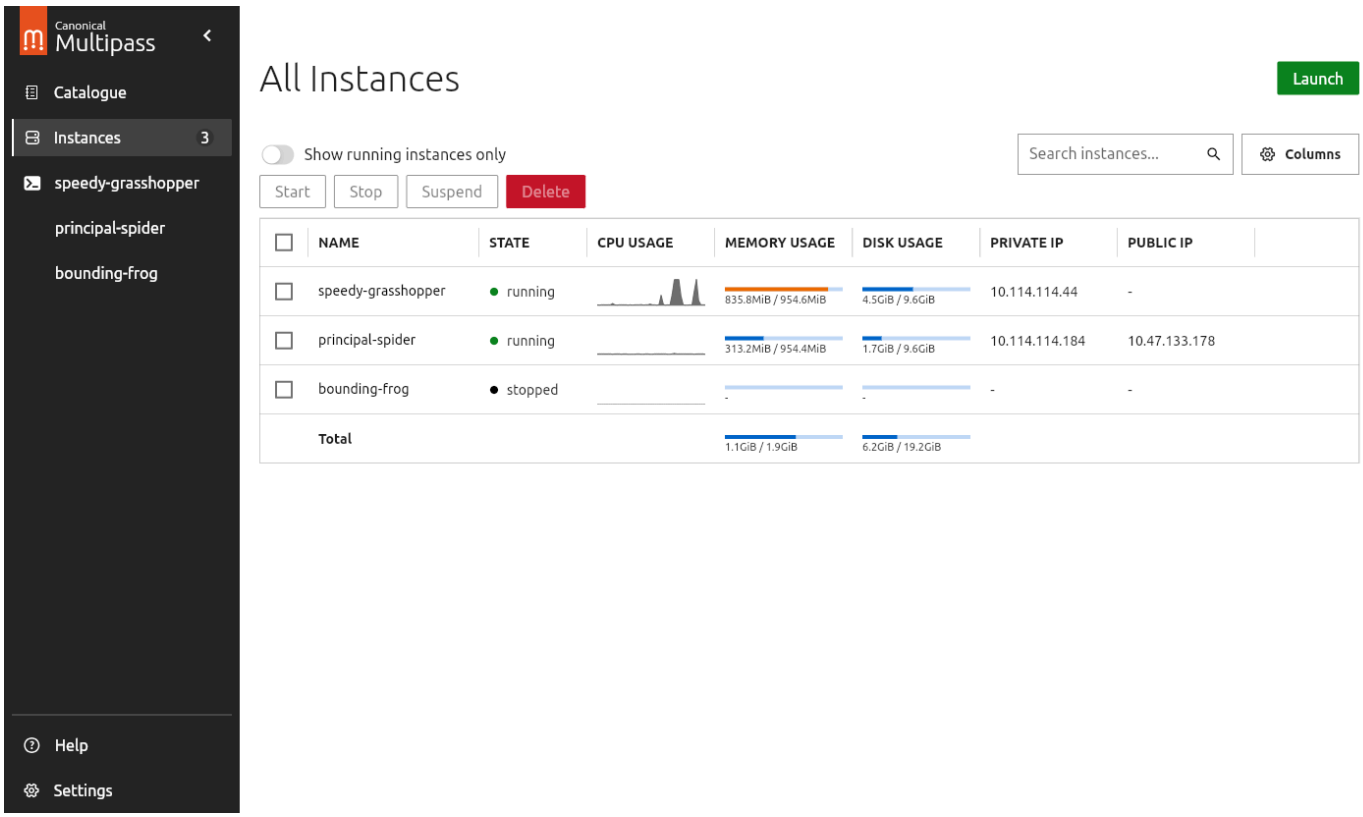


When you launch a VM, you can see details on all the steps taken and be notified of errors.

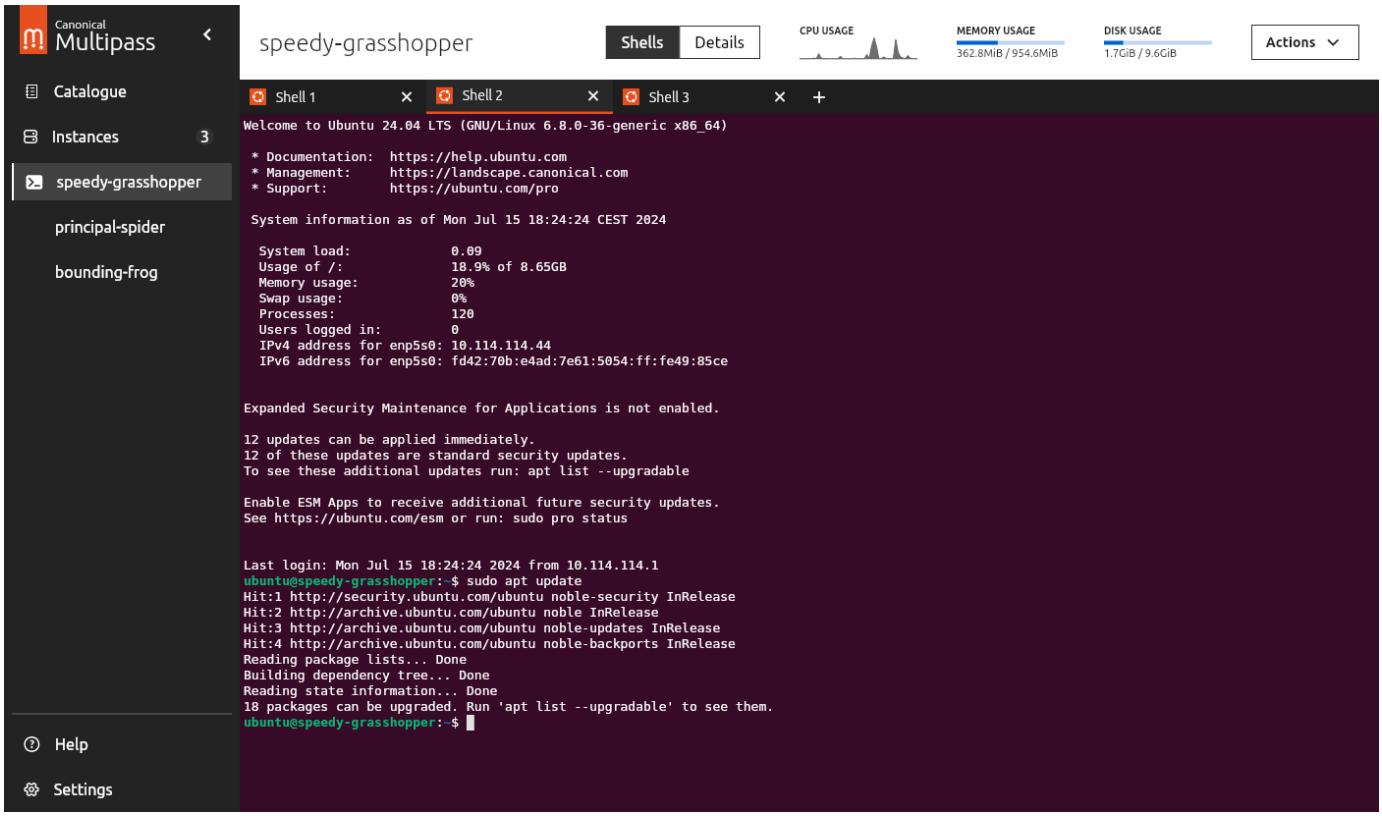


Instances tab

Here you can see an overview of all your instances and perform bulk actions such as starting, stopping, suspending or deleting the selected ones. You can also filter instances by name and by state (“running” or “stopped”).

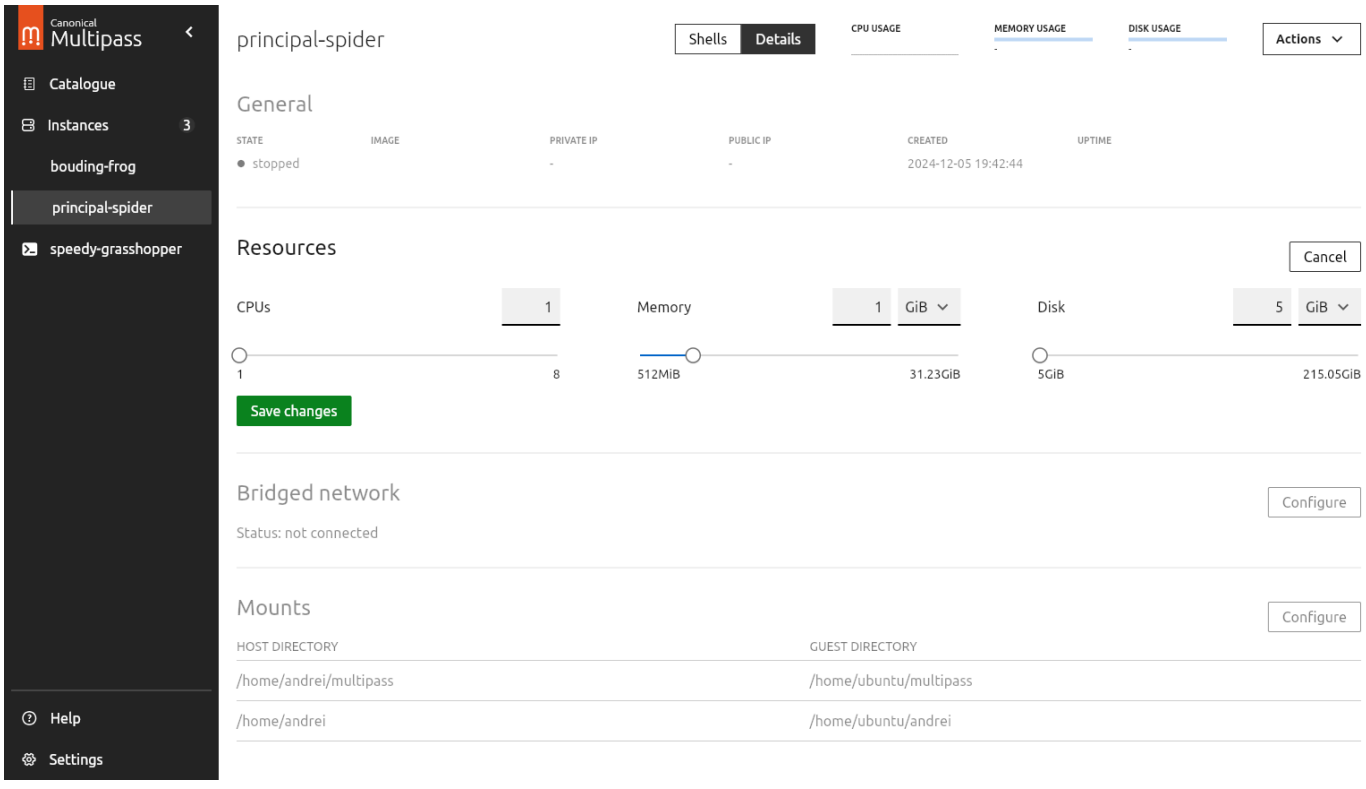


You can perform actions on an individual instance, such as starting, stopping, suspending or deleting it. You can also open shells within a running instance, where you can do all of your work that is specific to that instance.



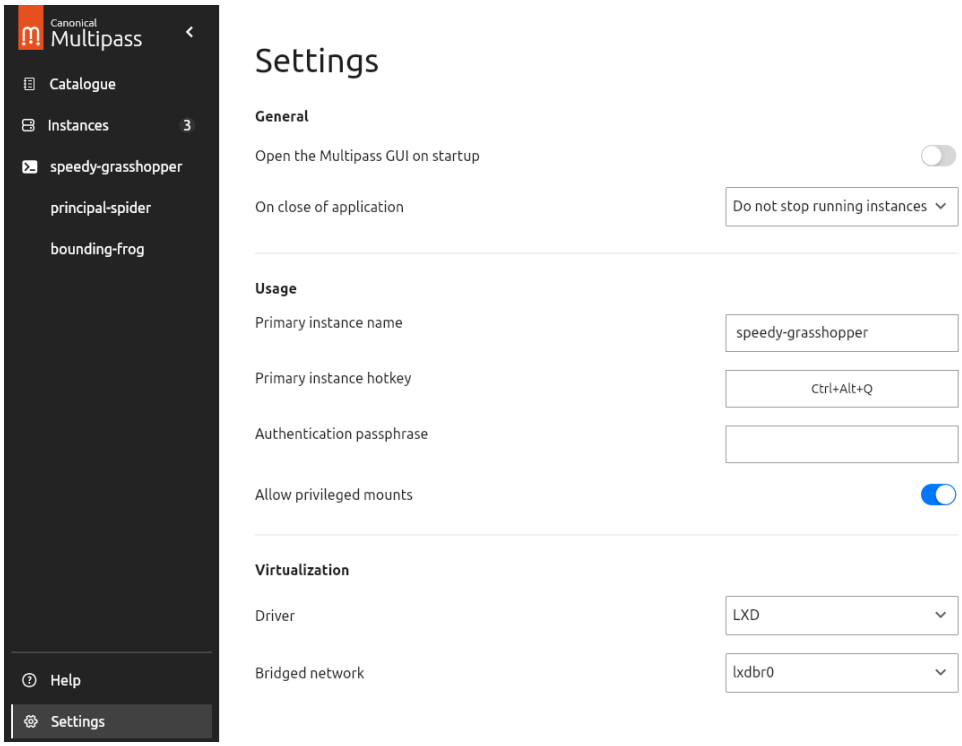
> **Caution**
>
> Please note that when you delete an instance using the GUI client, Multipass removes it permanently and they cannot be recovered. This behaviour is equivalent to running the `multipass delete --purge` command.

You can also edit an instance; in particular, you can change its allocated resources, connect it to a bridged network or edit its mounts. Some of these settings require the instance to be stopped before they can be applied.



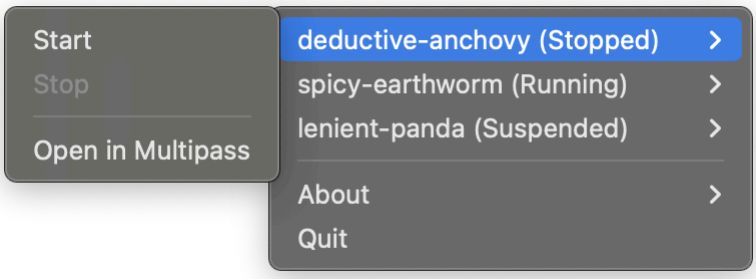
Settings tab

Here you can change various Multipass settings, although not all settings that are available in the CLI are present in the GUI and vice versa.



Tray icon

You can manage your instances using the tray icon menu as well.



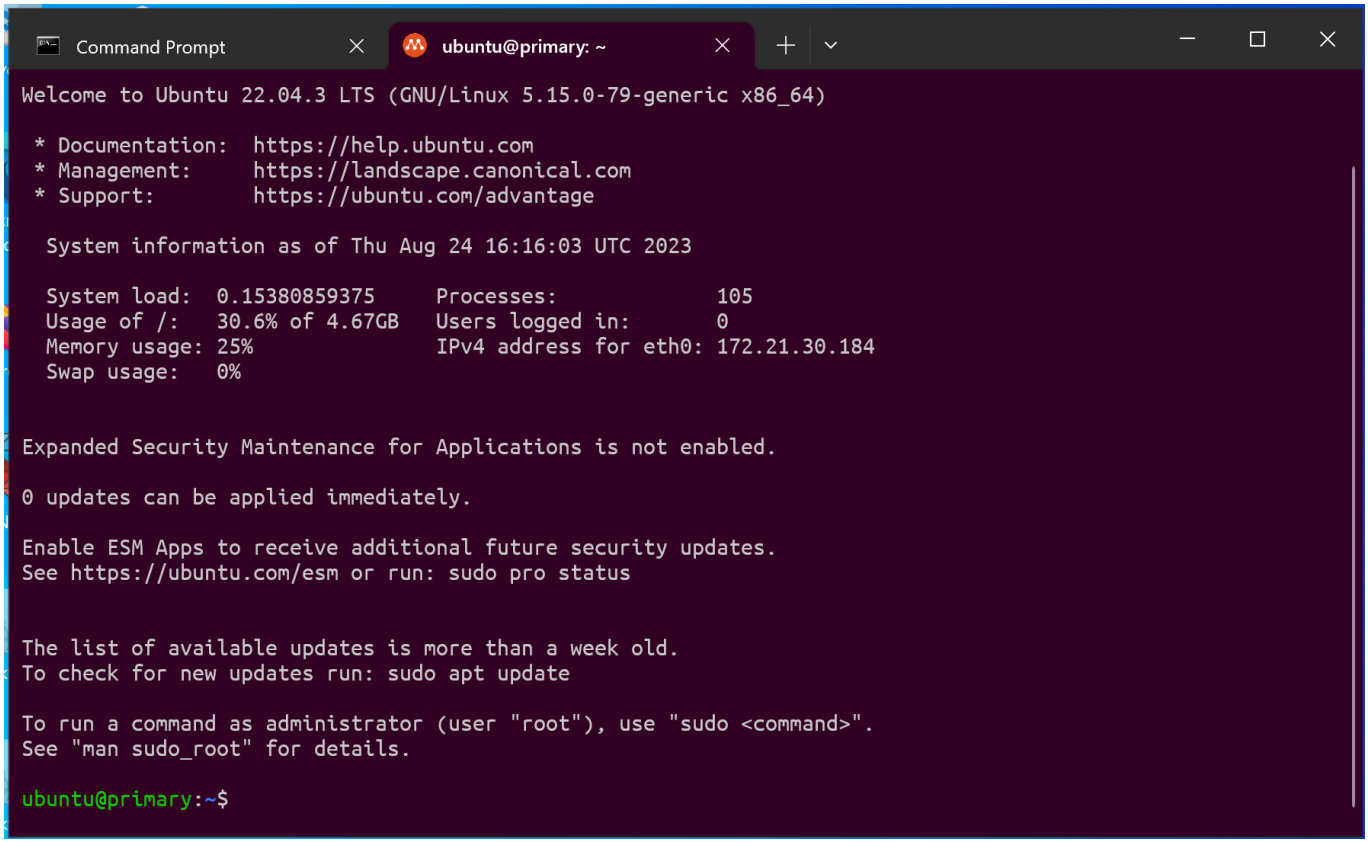
Last updated on Apr 07, 2025

How to integrate with Windows Terminal

If you are on Windows and you want to use Windows Terminal, Multipass can integrate with it to offer you an automatic `primary` profile.

Multipass profile

Currently, Multipass can add a profile to Windows Terminal for the Primary instance. When you open a Windows Terminal tab with this profile, you'll automatically find yourself in a primary instance shell. Multipass automatically starts or launches the primary instance if needed.



Install Windows Terminal

The first step is to install Windows Terminal. Once you have it along Multipass, you can enable the integration

Enable integration

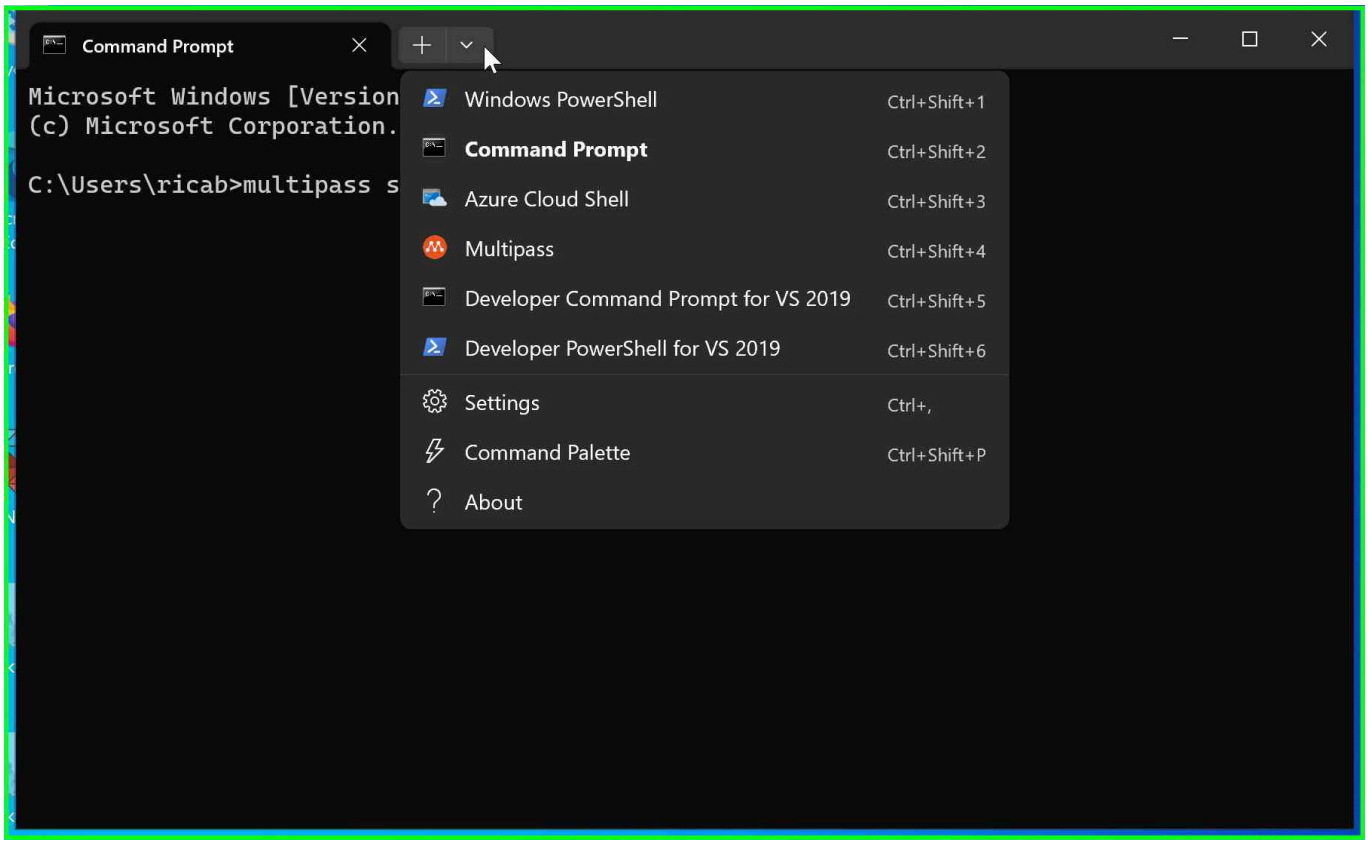
Open a terminal (Windows Terminal or any other) and enable the integration with the following command:

```
multipass set client.apps.windows-terminal.profiles=primary
```

For more information on this setting, see `client.apps.windows-terminal.profiles`. Until you modify it, Multipass will try to add the profile if it finds it missing. To remove the profile see Revert below.

Open a Multipass tab

You can now open a “Multipass” tab to get a shell in the primary instance. That can be achieved by clicking the new-tab drop-down and selecting the Multipass profile:



That’s it!

Revert

If you want to disable the profile again, you can do so with:

```
client.apps.windows-terminal.profiles=none
```

Multipass will then remove the profile if it exists.

Last updated on Feb 24, 2025

How to set up the driver

> See also: Driver, local.driver

This document demonstrates how to choose, set up, and manage the drivers behind Multipass. Multipass already has sensible defaults, so this is an optional step.

Default driver

Linux macOS Windows

By default, Multipass on Linux uses the `qemu` driver.

Install an alternative driver

Linux macOS Windows

> **Warning**
>
> Support for libvirt driver will be deprecated and removed in a future release.

If you want more control over your VMs after they are launched, you can also use the experimental libvirt driver.

To install libvirt, run the following command (or use the equivalent for your Linux distribution):

```
sudo apt install libvirt-daemon-system
```

Now you can switch the Multipass driver to libvirt. First, enable Multipass to use your local libvirt by connecting to the libvirt interface/plug:

```
sudo snap connect multipass:libvirt
```

Then, stop all instances and tell Multipass to use libvirt, running the following commands:

```
multipass stop --all
multipass set local.driver=libvirt
```

All your existing instances will be migrated and can be used straight away.

> **Note**
>
> You can still use the `multipass` client and the tray icon, and any changes you make to the configuration of the instance in libvirt will be persistent. They may not be represented in Multipass commands such as `multipass info`, though.

Use the driver to view Multipass instances

Linux macOS Windows

You can view instances with libvirt in two ways, using the `virsh` CLI or the `virt-manager` GUI.

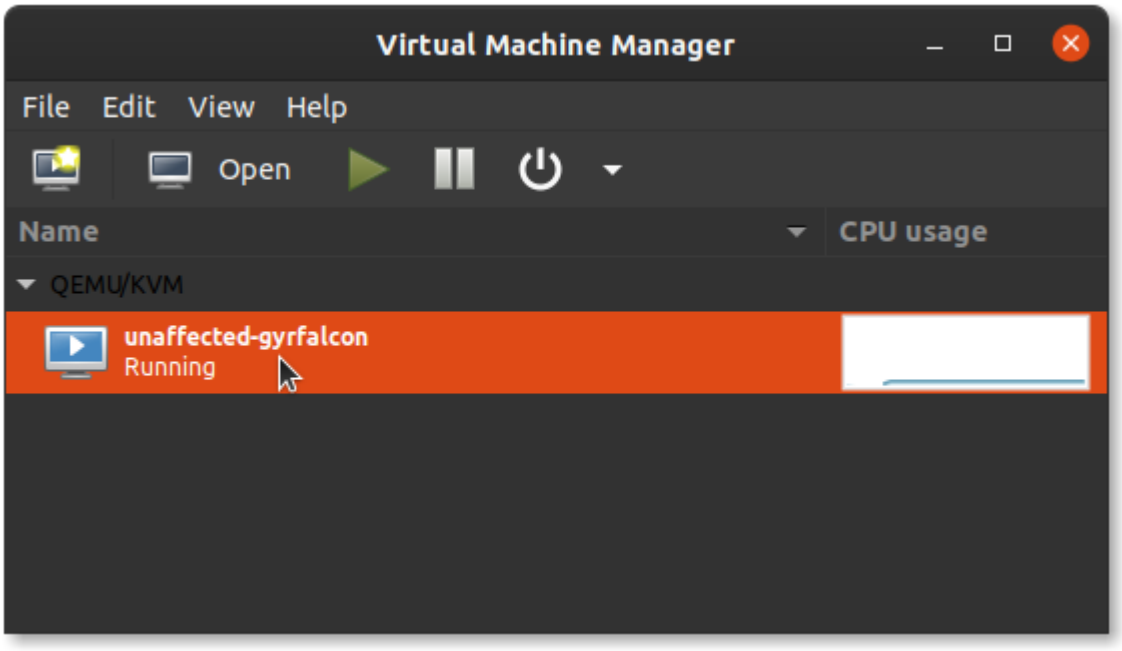
To use the `virsh` CLI, launch an instance and then run the command `virsh list` (see `man virsh` for a command reference):

```
virsh list
```

The output will be similar to the following:

```
 Id   Name                   State
-------------------------------------
 1    unaffected-gyrfalcon   running
```

Alternatively, to use the `virt-manager` GUI, …



Use VirtualBox to set up port forwarding for a Multipass instance

Linux macOS Windows

This option only applies to macOS and Windows systems.

Use VirtualBox to set up network bridging for a Multipass instance

Linux macOS Windows

This option only applies to macOS systems.

Switch back to the default driver

> See also: stop, local.driver

Linux macOS Windows

To switch back to the default `qemu` driver, first you need to stop all instances again:

```
multipass stop --all
multipass set local.driver=qemu
```

Here, too, existing instances will be migrated.

> **Note**
>
> This will make you lose any customisations you made to the instance in `libvirt`. [/note]

Last updated on Apr 09, 2025

Install Multipass

This guide explains how to install and manage Multipass on your system.

> **Note**
>
> Select the tab corresponding to your operating system (e.g. Linux) to display the relevant content in each section. Your decision will stick until you select another OS from the drop-down menu.

Check prerequisites

Linux macOS Windows

Multipass for Linux is published as a snap package, available on the Snap Store. Before you can use it, you need to install snapd. snapd is included in Ubuntu by default.

Install

Linux macOS Windows

To install Multipass, run the following command:

```
snap install multipass
```

You can also use the edge channel to get the latest development build:

```
snap install multipass --edge
```

Make sure you're part of the group that Multipass gives write access to its socket (`sudo` in this case, but it may also be `wheel` or `admin` , depending on your distribution).

1. Run this command to check which group is used by the Multipass socket:

   ```
   ls -l /var/snap/multipass/common/multipass_socket
   ```

   The output will be similar to the following:

   ```
   srw-rw---- 1 root sudo 0 Dec 19 09:47 /var/snap/multipass/common/multipas
   ```

2. Run the `groups` command to make sure you are a member of that group (in our example, "sudo"):

   ```
   groups | grep sudo
   ```

   The output will be similar to the following:

   ```
   adm cdrom sudo dip plugdev lpadmin
   ```

You can view more information on the snap package using the `snap info` command:

```
snap info multipass
```

For example:

```
name:      multipass
summary:   Instant Ubuntu VMs
publisher: Canonical✓
store-url: https://snapcraft.io/multipass
contact:   https://github.com/canonical/multipass/issues/new
license:   GPL-3.0
description: |
  Multipass is a tool to launch and manage VMs on Windows, Mac and Linux tha
  environment with support for cloud-init. Get Ubuntu on-demand with clean
  and version control on your native platform.
  ...
commands:
  - multipass.gui
  - multipass
services:
  multipass.multipassd: simple, enabled, active
snap-id:      mA11087v6dR3IEcQLgICQVjuvhUUBUKM
tracking:     latest/candidate
refresh-date: 5 days ago, at 10:13 CEST
channels:
  latest/stable:    1.3.0                2020-06-17 (2205) 228MB -
  latest/candidate: 1.3.0                2020-06-17 (2205) 228MB -
  latest/beta:      1.3.0-dev.17+gf89e1db 2020-04-28 (2019) 214MB -
  latest/edge:      1.4.0-dev.83+g149f10a 2020-06-17 (2216) 228MB -
installed:          1.3.0                          (2205) 228MB -
```

Alternatively, you can also check your preferred package manager to see if it provides Multipass, although this option is not officially supported.

Run

Linux macOS Windows

You've installed Multipass. Time to run your first commands! Use `multipass version` to check your version or `multipass launch` to create your first instance.

Upgrade

Linux macOS Windows

As the installation happened via snap, you don’t need to worry about upgrading—it will be done automatically.

Uninstall

Linux macOS Windows

To uninstall Multipass, run the following command:

```
snap remove multipass
```


Last updated on Apr 30, 2025

Instance states

> See also: Command-line interface

Instances in Multipass can be in a number of different states:

STATE	DESCRIPTION
Running	The instance is currently running and is ready to be used.
Stopped	The instance has been intentionally stopped and is not currently consuming resources. It can be started when needed.
Deleted	The instance has been marked for deletion. The instance can either be recovered or purged.
Starting	The instance is in the process of being started up and initialised. It will transition to the *Running* state once fully started.
Restarting	The instance is undergoing a restart. This involves stopping the instance and then starting it again.
Delayed shutdown	The instance has been sent a shutdown signal and will be stopped after a specified delay. This allows for any ongoing processes to be completed before shutdown.
Suspending	This instance is in the process of being suspended. The instance's state and memory will be saved, allowing it to be resumed from where it left off.
Suspended	The instance has been suspended, meaning its state and memory have been saved. It can be resumed from this state to continue its operation.
Unknown	The state of the instance cannot be determined or retrieved. This might occur due to unexpected errors or issues with Multipass.

These instance states reflect the various stages an instance can be in while using Multipass. Instances in different states can accept different commands. See Command-line interface for more information on which commands can be used and when.

Last updated on Feb 23, 2025

latest

Launch customized instances with Multipass and cloud-init

You can set up instances with a customized environment or configuration using the launch command along with a custom cloud-init YAML file and an optional post-launch health check to ensure everything is working correctly.

Below are some common examples of using `cloud-init` with Multipass to create customized instances. The `cloud-init` file is provided by the Multipass team, but users are free to create and use their own personal `cloud-init` configurations.

anbox-cloud-appliance

Launch with:

```
multipass launch \
  --name anbox-cloud-appliance \
  --cpus 4 \
  --memory 4G \
  --disk 50G \
  --timeout 900 \
  --cloud-init https://raw.githubusercontent.com/canonical/multipass/refs/h
```

charm-dev

Launch with:

```
multipass launch 24.04 \
  --name charm-dev \
  --cpus 2 \
  --memory 4G \
  --disk 50G \
  --timeout 1800 \
  --cloud-init https://raw.githubusercontent.com/canonical/multipass/refs/h
```

Health check:

```
multipass exec charm-dev -- bash -c "
 set -e
 charmcraft version
 mkdir -p hello-world
 cd hello-world
 charmcraft init
 charmcraft pack
"
```

docker

Launch with:

```
multipass launch 24.04 \
  --name docker \
  --cpus 2 \
  --memory 4G \
  --disk 40G \
  --cloud-init https://raw.githubusercontent.com/canonical/multipass/refs/h
```

Health check:

```
multipass exec docker -- bash -c "docker run hello-world"
```

You can also optionally add aliases:

```
multipass prefer docker
multipass alias docker:docker docker
multipass alias docker:docker-compose docker-compose
multipass prefer default
multipass aliases
```

> See also: How to use command aliases

jellyfin

Launch with:

```
multipass launch 22.04 \
  --name jellyfin \
  --cpus 2 \
  --memory 4G \
  --disk 40G \
  --cloud-init https://raw.githubusercontent.com/canonical/multipass/refs/h
```

minikube

Launch with:

```
multipass launch \
  --name minikube \
  --cpus 2 \
  --memory 4G \
  --disk 40G \
  --timeout 1800 \
  --cloud-init https://raw.githubusercontent.com/canonical/multipass/refs/h
```

Health check:

```
multipass exec minikube -- bash -c "set -e
  minikube status
  kubectl cluster-info"
```

ros2-humble

Launch with:

```
multipass launch 22.04 \
  --name ros2-humble \
  --cpus 2 \
  --memory 4G \
  --disk 40G \
  --timeout 1800 \
  --cloud-init https://raw.githubusercontent.com/canonical/multipass/refs/h
```

Heath check:

```
multipass exec ros2-humble -- bash -c "
  set -e

  colcon --help
  rosdep --version
  ls /etc/ros/rosdep/sources.list.d/20-default.list
  ls /home/ubuntu/.ros/rosdep/sources.cache

  ls /opt/ros/humble
"
```

ros2-jazzy

Launch with:

```
multipass launch 24.04 \
  --name ros2-jazzy \
  --cpus 2 \
  --memory 4G \
  --disk 40G \
  --timeout 1800 \
  --cloud-init https://raw.githubusercontent.com/canonical/multipass/refs/h
```

Health check:

```
multipass exec ros2-jazzy -- bash -c "
  set -e

  colcon --help
  rosdep --version
  ls /etc/ros/rosdep/sources.list.d/20-default.list
  ls /home/ubuntu/.ros/rosdep/sources.cache

  ls /opt/ros/jazzy
"
```


Last updated on Apr 23, 2025

Logging levels

> See also: Configure Multipass's default logging level, How to access logs

In Multipass, a hierarchy of logging levels is used to convey severity and improve visibility of important events. Multipass uses the following levels, ranked from most severe to least severe, for its background daemon and child processes.

Error

Indicates a failure that prevents the intended operation from being accomplished in its entirety. If there is a corresponding CLI command, it should exit with an error code.

Warning

Indicates an event or fact that might not correspond to the user's intentions/desires/beliefs, or a problem that is light enough that it does not prevent main goals from being accomplished. If there is a corresponding CLI command, it should exit with a success code.

Info

Indicates information that may be useful for the user to know, learn, etc.

Debug

Indicates information that is useful for developers and troubleshooting.

Trace

Indicates information that may be helpful for debugging, but which would clutter logs unreasonably if enabled by default.

Last updated on Feb 24, 2025

latest

Migrate from Hyperkit to QEMU on macOS

> See also: set , local.driver, Driver, How to set up the driver

As of Multipass 1.12, the Hyperkit driver is being deprecated. New installs will start with the QEMU driver set by default, but existing installs will retain the previous driver setting. Multipass will warn Hyperkit users of the deprecation and ask them to move to QEMU. To facilitate that, Multipass 1.12 will migrate Hyperkit instances to QEMU.

To migrate from Hyperkit to QEMU and bring your instances along, simply stop them and set the driver to QEMU:

```
multipass stop --all
multipass set local.driver=qemu
```

If you already had QEMU instances, they are not affected by the migration. Instances whose name is taken on the QEMU side are not migrated.

Repeated driver switches

The original Hyperkit instances are retained until explicitly deleted. You can achieve that by temporarily moving back to Hyperkit and using the delete command:

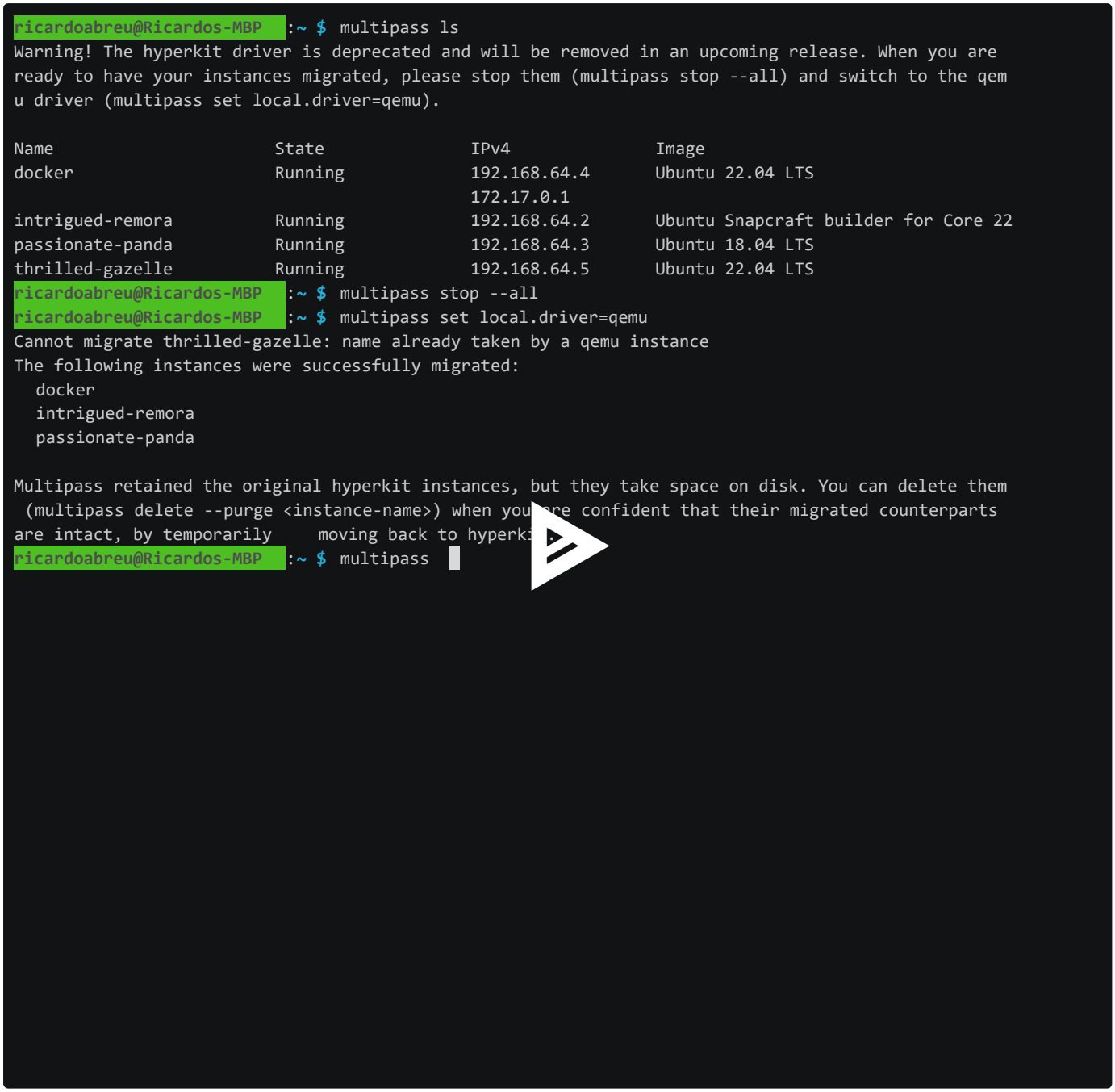
```
multipass set local.driver=hyperkit
multipass delete [-p] <instance> [...]
multipass set local.driver=qemu
```

When switching to QEMU again, migrated instances are not overwritten. If, for any reason, you want to repeat a migration, you can achieve that by deleting the QEMU counterpart first.

You can choose a convenient time to do any of this and you can set the driver to Hyperkit and move back and forth as many times as you want. Apart from the deprecation warning, ains the same until the driver is removed entirely. When that happens, it will ible to migrate Multipass (unless you downgrade to versio

Demo

Here is a video demonstration of the migration:



Last updated on Feb 23, 2025

Modify an instance

> See also: Instance, `launch`, `set`, Settings

This document shows further ways to customise an instance outside of the `launch` command using the Multipass settings.

Set the CPUs , RAM or disk space of an instance

> See also: `local.<instance-name>.cpus`, `local.<instance-name>.disk)`, `local.<instance-name>.memory`

You can set instance properties at `launch` time, but you can also update some of them after the instance has been created. Specifically, an instance's memory, disk space, and the number of its CPUs are exposed via daemon settings: `local.<instance-name>.(cpus|disk|memory)`.

To modify one of this properties, first stop the instance and then issue the `set` command. For example:

```
multipass stop handsome-ling
multipass set local.handsome-ling.cpus=4
multipass set local.handsome-ling.disk=60G
multipass set local.handsome-ling.memory=7G
```

Note

The disk size can only be increased.

Caution

When increasing the disk size of an instance, the partition may not expand automatically to use the new available space. This usually happens if the partition was already full when trying to increase the disk size.

In such cases, you need to expand the partition manually. To do so, `shell` into the instance and run the following command:

```
/dev/sda resizepart 1 100%
```

The system will guide you through the configuration steps:

```
Warning: Not all of the space available to /dev/sda appears to be used,
you can fix the GPT to use all of the space (an extra 4194304 blocks)
or continue with the current setting.
Fix/Ignore? fix
Partition number? 1
Warning: Partition /dev/sda1 is being used.
Are you sure you want to continue? Yes/No? yes
```

When done, run `sudo resize2fs /dev/sda1`.

```
You can view these properties using the `get` command, without the need to

```{note}
You can only update the properties of `Stopped`, non-deleted instances. If

On the other hand, it's always possible to fetch properties for all instanc
```

> **Note**
>
> Modifying instance settings is not supported when using the Hyperkit driver, which has been deprecated in favour of QEMU. The QEMU and VirtualBox drivers on Intel-based macOS hosts do support instance modification.

Set the status of an instance to primary

> See also: client.primary-name

This section demonstrates how to set the status of an instance to primary. This is convenient because it makes this instance the default argument for several commands, such as `shell`, `start`, `stop`, `restart` and `suspend`, and also automatically mounts your $HOME directory into the instance.

To grant a regular instance the primary status, assign its name to the `client.primary-name`:

```
multipass set client.primary-name=<instance name>
```

This setting allows transferring primary status among instances. The primary instance's name is changed independently of whether instances with the old and new primary status exist, and they gain primary status accordingly.

This provides a means of (de)selecting an existing instance as primary.
```

Example

Assign the primary status to an instance called “first”:

```
multipass set client.primary-name=first
```

This instance is picked up automatically by `multipass start`. When you run this command, the primary instance also automatically mounts the user’s home directory into a directory called `Home`:

```
...
Launched: first
Mounted '/home/ubuntu' into 'first:Home'
```

Run `multipass stop` to stop the primary instance, and then launch another instance named “second”:

```
multipass launch --name second
```

Now, change the primary instance to the existing “second” instance:

```
multipass set client.primary-name=second
```

From now on, the “second” instance will be used as the *primary* instance, so for example it will be used by default when you run the command `multipass suspend`.

When listing instances, the primary one is displayed first. For example, if you run `multipass list` now, you’ll see:

```
Name                    State             IPv4             Image
second                  Suspended         --               Ubuntu 18.04 LTS
first                   Stopped           --               Ubuntu 18.04 LTS
```


Last updated on Feb 19, 2025

Mount an encrypted home folder

> See also: `mount` , Instance

When you create the Primary instance using `multipass start` or `multipass shell` without additional arguments, Multipass automatically mounts your home directory into it.

On Linux, if your local home folder is encrypted using `fscrypt` , snap confinement prevents you from accessing its contents from a Multipass mount due the peculiar directory structure (`/home/.ecryptfs/<user>/.Private/`). This also applies to the primary instance, where the home folder is mounted automatically.

A workaround is mounting the entire `/home` folder into the instance, using the command:

```
multipass mount /home primary
```

By doing so, the home folder's contents will be mounted correctly.

Last updated on Feb 19, 2025

latest

Remove an instance

> See also: Instance

This guide demonstrates how to remove an instance, either temporarily or permanently.

Move an instance to the recycle bin

> See also: `delete`, `recover`

To mark an instance as deleted, run:

```
multipass delete keen-yak
```

Now, if you run `multipass list` to list the instances, you will see that it is actually just marked for deletion (or to put it in other words, moved to the recycle bin):

```
Name                    State             IPv4             Release
keen-yak                DELETED           --               Not Available
```

You can move all instances to the recycle bin at once using the `--all` option:

```
multipass delete --all
```

Instances that have been marked as deleted can later be recovered; for example:

```
multipass recover keen-yak
```

If you try `multipass list` again, you'll see that the instance is no longer marked for deletion:

```
Name                    State             IPv4             Release
                        STOPPED           --               Ubuntu 18.04 LTS
```

Remove an instance permanently

> See also: `delete`, `purge`

If you want to get rid of all instances in `Deleted` status for good, you can purge them:

```
multipass delete keen-yak
multipass purge
```

> **Caution**
>
> The `purge` command does not take an argument. It will permanently remove all instances marked as `Deleted`.

You can also use the `--purge` option to permanently delete an instance in a single command; for example:

```
multipass delete --purge keen-yak
```


Last updated on Feb 19, 2025

Run a Docker container in Multipass

> **Warning**
>
> Blueprints are deprecated and will be removed in a future release. You can achieve similar effects with cloud-init and other launch options. Find out more at: Launch customized instances with Multipass and cloud-init

Overview

Multipass has a Docker blueprint that gives its users access to out-of-the-box Docker on any platform. This new blueprint makes it easy to develop and test Docker containers locally on macOS, Windows, or Linux.

In this tutorial, you will see how to get started with the Docker blueprint by creating a blog in a Docker container in Multipass.

What you'll learn

- How to use Docker on macOS or Windows with Multipass
- How to alias the `docker` command to your host command line
- How to use Portainer to launch a Docker container in Multipass.

What you'll need

- Any computer with an internet connection

Install Multipass

Duration: 3 minutes

Let's start by installing Multipass on your machine, following the steps in How to install Multipass.

Launch a Docker VM

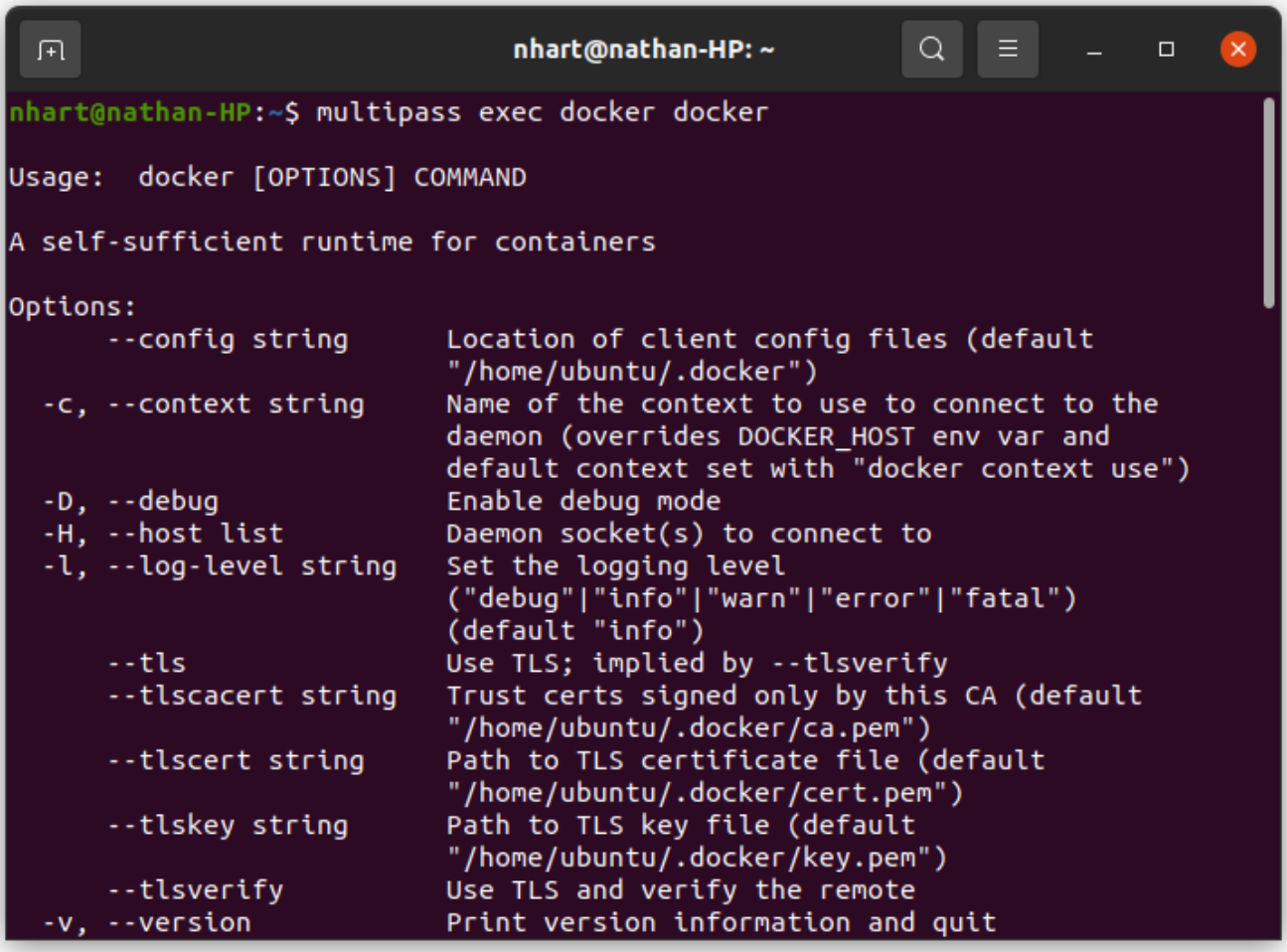
Duration: 1 minute

Now that Multipass is installed, you can create a VM running Docker very simply. Open up a terminal and type

```
multipass launch docker
```

This command will create a virtual machine running the latest version of Ubuntu, with Docker and Portainer installed. You can now use Docker already! Try the command below to see for yourself!

```
multipass exec docker docker
```



Alias of the Docker commands

Duration: 1 minute

The Docker blueprint creates automatically two aliases, that is, two commands which can be run from the host to use commands in the instance as if they were in the host. In particular, the host `docker` command executes `docker` in the instance, and the host `docker-compose` command executes `docker-compose` in the instance.

In order for these to work, you just need to add them to the path so that you can use them directly from your command line. If this was not done before, launching the Docker blueprint will return instructions showing how to add the aliases to your path. Simply copy and paste the command shown. It will likely be of this form:

```
PATH="$PATH:/home/<user>/snap/multipass/common/bin"
```

Run the command:

```
multipass launch docker
```

Sample output:

```
You'll need to add this to your shell configuration (.bashrc, .zshrc or so)
aliases to work without prefixing with `multipass`:

PATH="$PATH:/home/nhart/snap/multipass/common/bin"
```

You can now use `docker` straight from the command line. To try it out, run

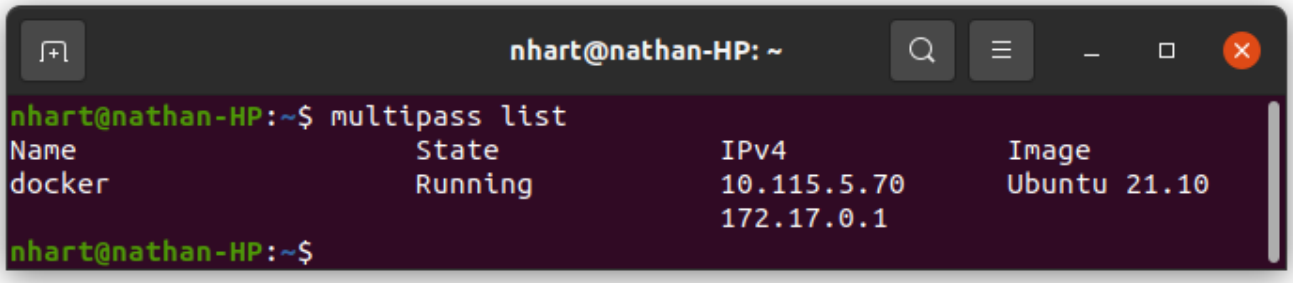
```
docker run hello-world
```

Using Portainer

Duration: 5 minutes

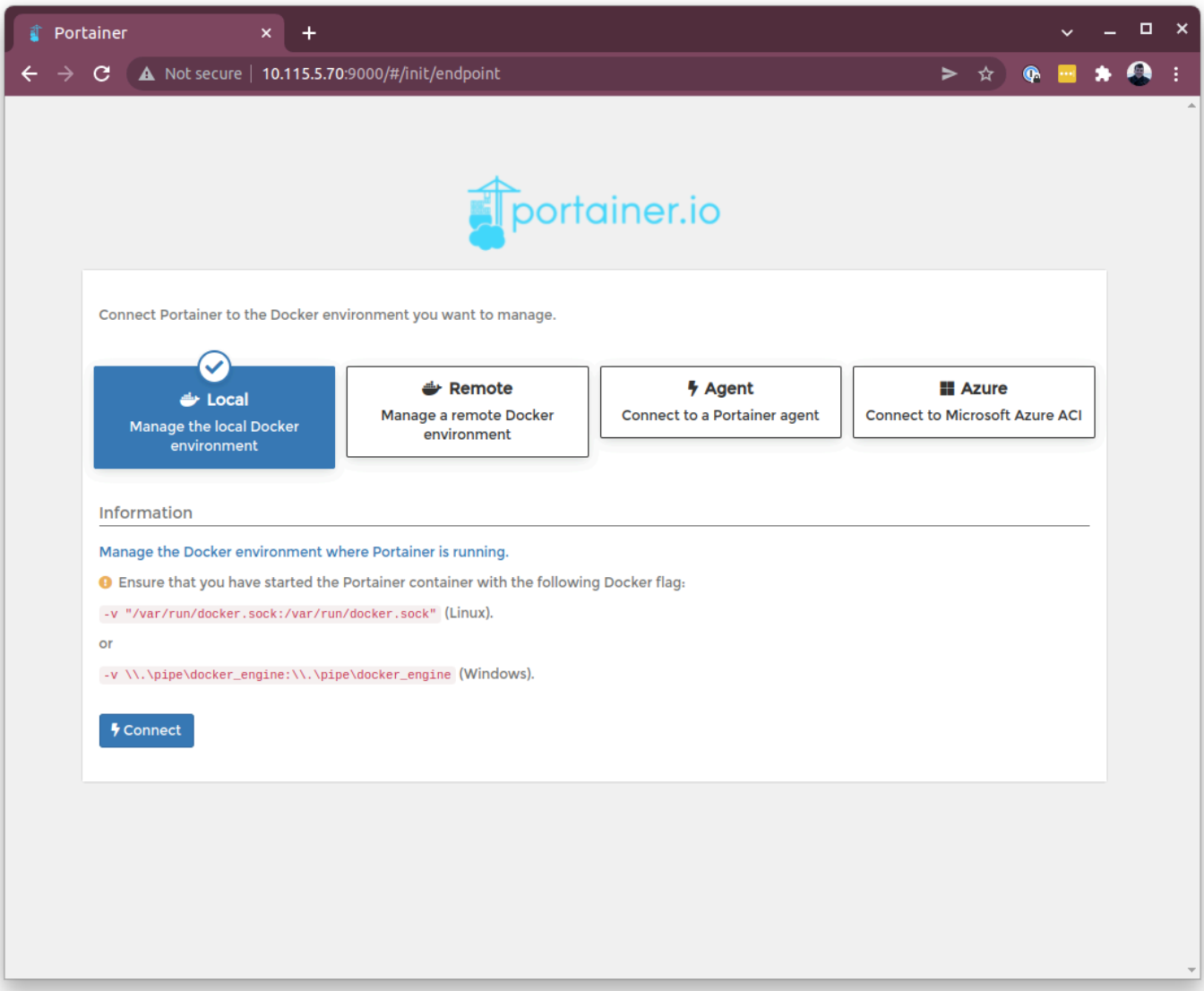
Let’s now go one step further, with Portainer. The Docker blueprint comes with Portainer installed, which gives an easy-to-use graphical interface for managing your Docker containers. To access Portainer, you will first need its IP address. The following command will show the IP addresses associated with the Docker VM you created in the previous steps:

```
multipass list
```

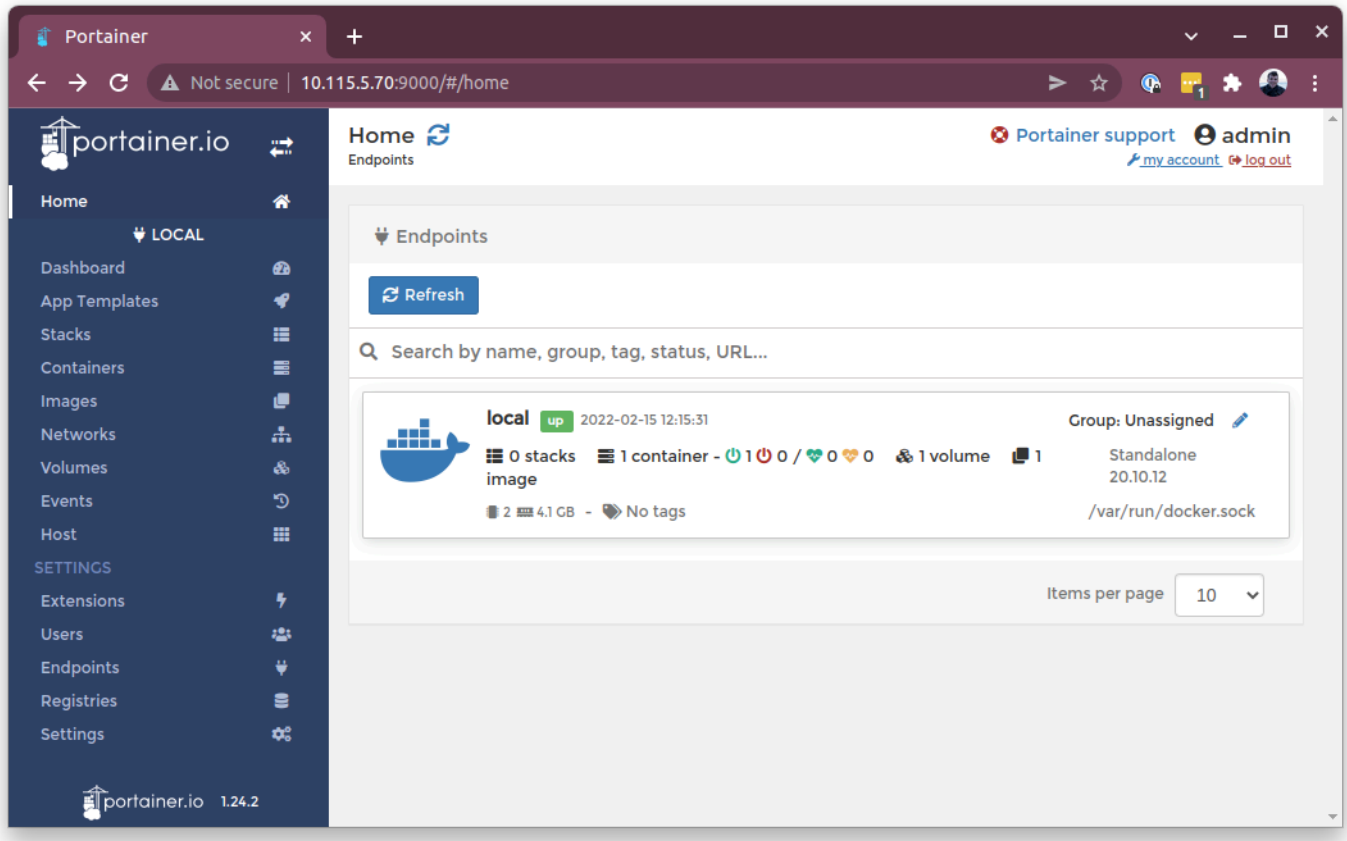


There should be two IP addresses listed, one for the Docker instance, the other for Portainer. The Portainer IP should start with a 10.

In a web browser, enter the Portainer IP address from the previous step followed by the Portainer port, 9000, like this: ":9000". Set up a username and password at the prompt, then select the option for managing a *local* Docker environment and click *connect*.



Click on the newly created "Local" environment to manage the Docker instance on your local VM.

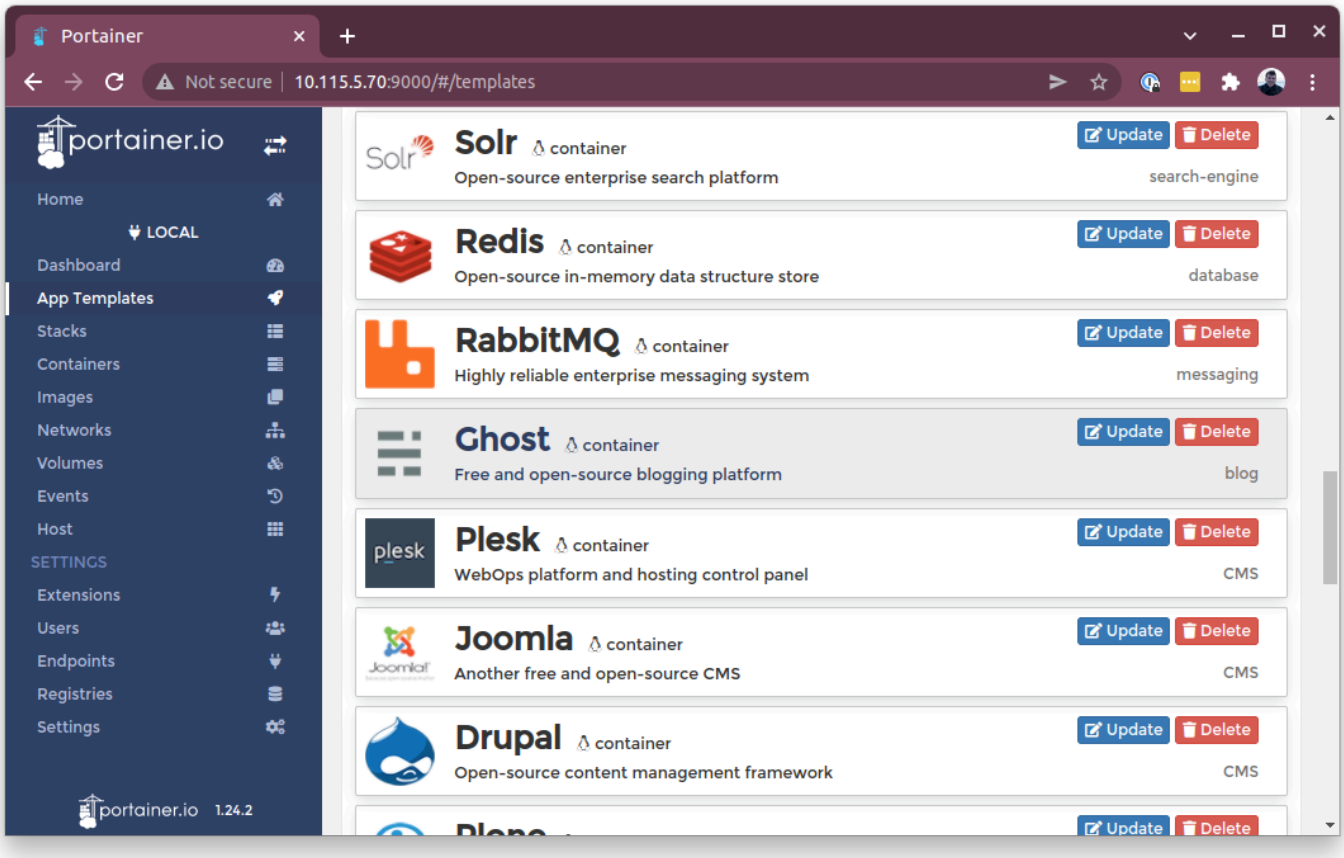


Launching a container

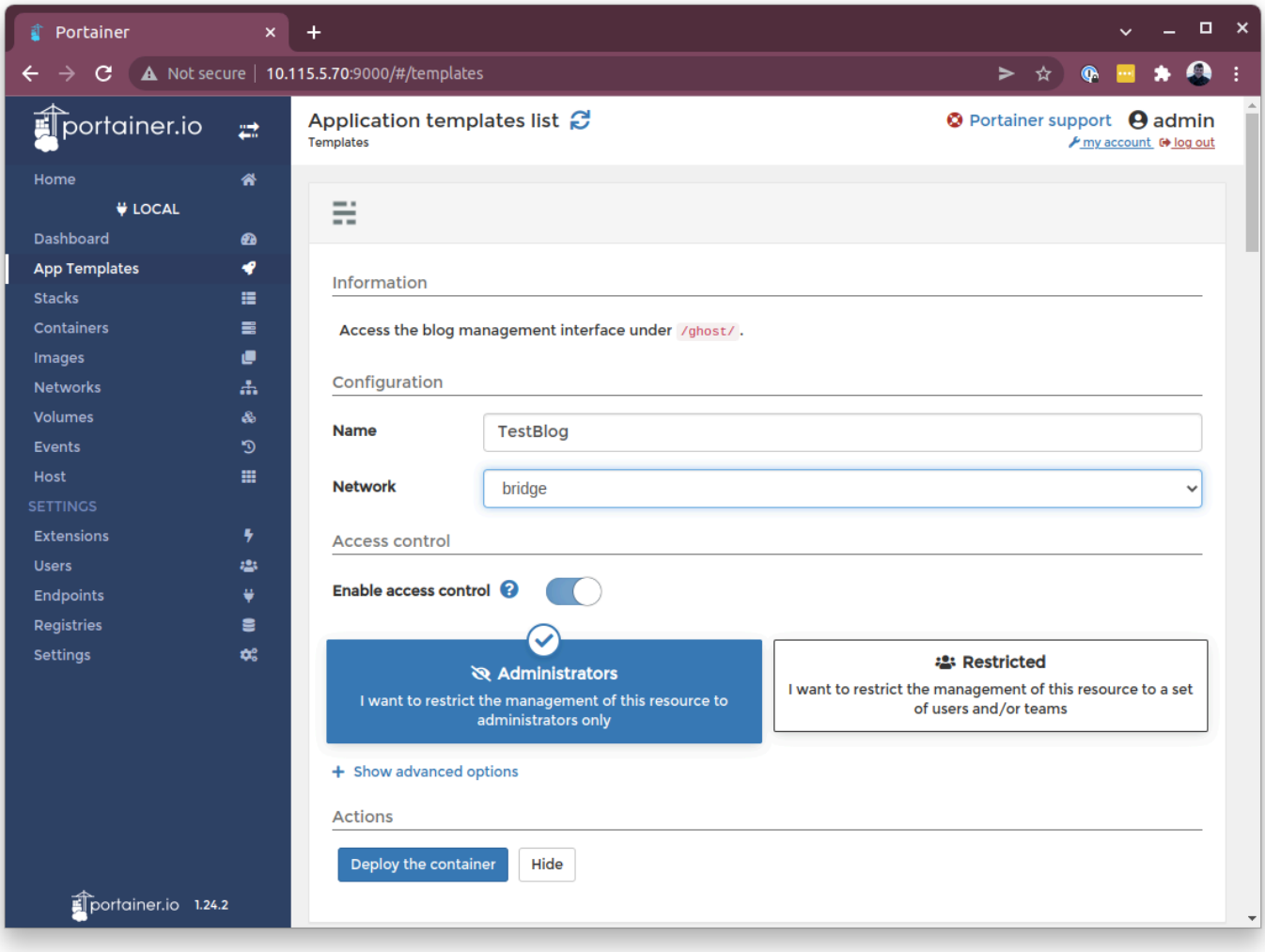
Duration: 5 minutes

For this tutorial, you will be creating a blog using the Ghost template in Portainer. Portainer has many other app templates if you are looking for more ideas. If you want more selection, you can launch containers from the Docker hub from Portainer or from the command line.

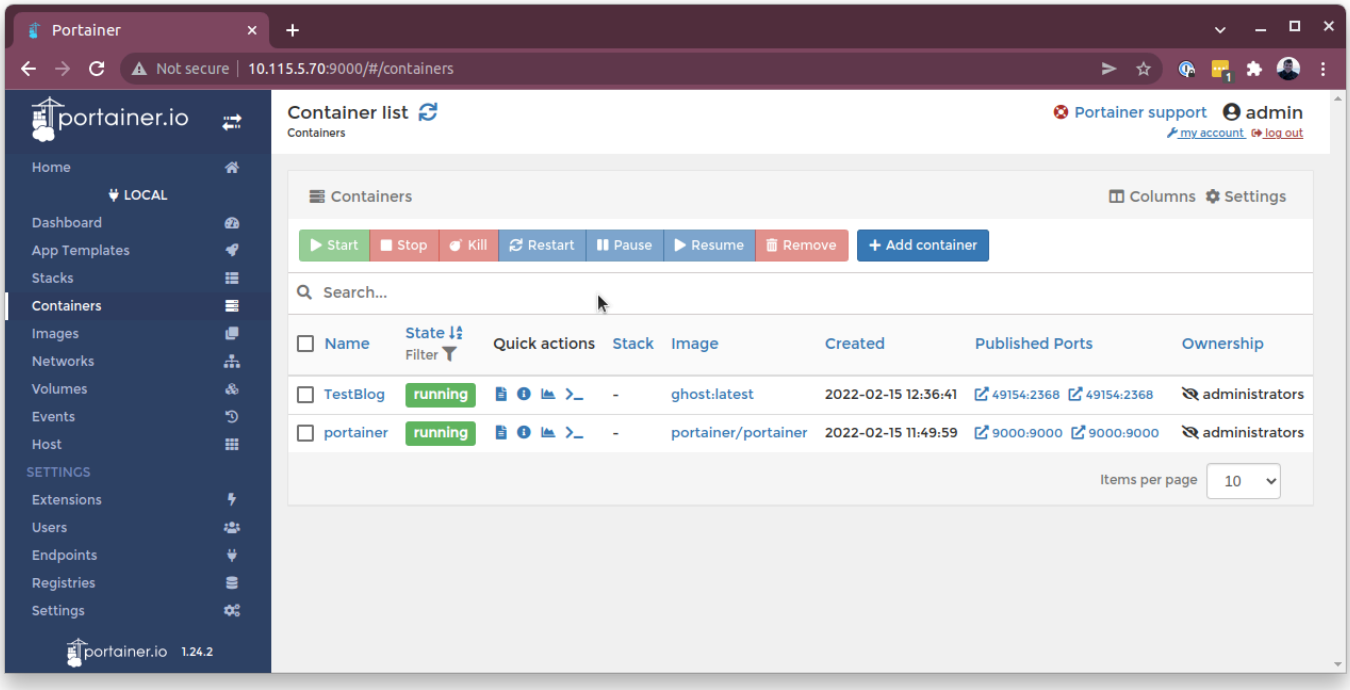
Inside Portainer, click on **App Templates** in the left toolbar, and scroll down to the **Ghost** template.



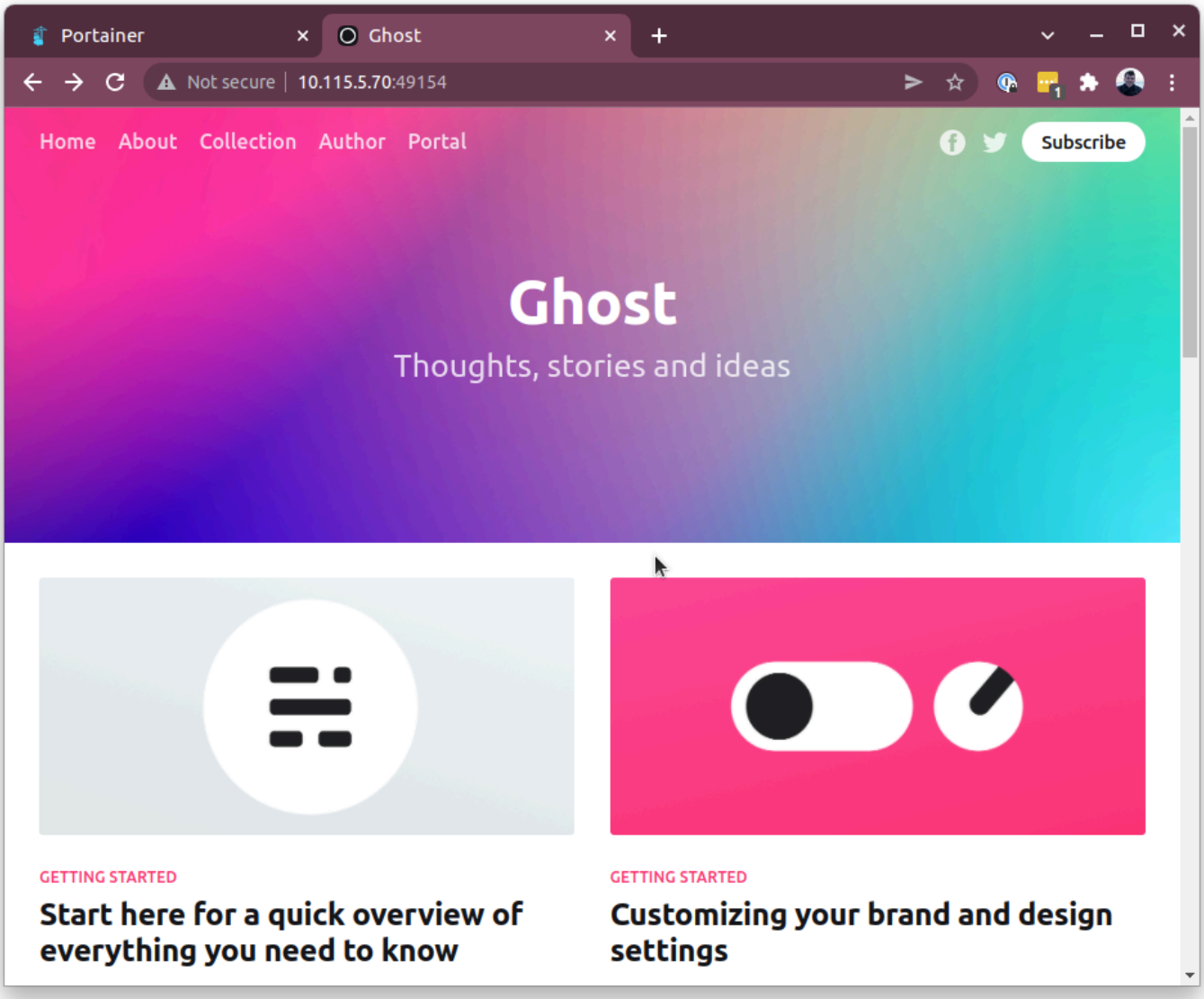
Now, you can configure and deploy the template. Enter a name and click deploy. The **bridge** network is the default and correct option.



On the **Containers** page, you should now see two containers running. One containing Ghost, and the other containing Portainer itself.



You can now access your Ghost blog by going to the published port indicated in the Containers page, i.e., **<VM IP Address>:<Ghost Port>**.



There it is, your blog running within a Docker container inside Multipass!

For next steps, try out Portainer's other App Templates (Step 5), or check out Docker Hub for more containers to try. If you want to try out container orchestration, Microk8s or Multipass' Minikube blueprint are great places to start.

Last updated on Apr 23, 2025

Set up a graphical interface

You can display the graphical desktop in various ways. In this document, we describe two options: RDP (Remote Display Protocol) and plain X11 forwarding. Other methods include VNC and running a Mir shell through X11 forwarding, as described in A simple GUI shell for a Multipass VM.

Using RDP

The images used by Multipass do not come with a graphical desktop installed. For this reason, you will have to install a desktop environment (here we use `ubuntu-desktop` but there are as many other options as flavours of Ubuntu exist) along with the RDP server (we will use `xrdp` but there are also other options such as `freerdp`).

To do this, first you need to log into a running Multipass instance. Start by listing your instances:

```
multipass list
```

Sample output:

```
Name                    State             IPv4             Image
headbanging-squid       Running           10.49.93.209     Ubuntu 22.04 LTS
```

Next, open a shell into the running instance:

```
multipass shell headbanging-squid
```

Once inside the instance, run the following commands to install `ubuntu-desktop` and `xrdp`:

```
sudo apt update
sudo apt install ubuntu-desktop xrdp
```

Now we need a user with a password to log in. One possibility is setting a password for the default `ubuntu` user:

```
sudo passwd ubuntu
```

You will be asked to enter and re-enter a password.

You are done on the server side!

Quit the Ubuntu shell on the running instance with the `exit` command, and take note of the IP address to connect to. You can find the instance's IP address in the output of `multipass list` from the first step above, or you can use the `multipass info` command as well.

```
multipass info headbanging-squid
```

Sample output:

```
Name:           headbanging-squid
State:          Running
Snapshots:      0
IPv4:           10.49.93.209
Release:        Ubuntu 22.04 LTS
Image hash:     2e0c90562af1 (Ubuntu 22.04 LTS)
CPU(s):         4
Load:           0.00 0.00 0.00
Disk usage:     1.8GiB out of 5.7GiB
Memory usage:   294.2MiB out of 3.8GiB
Mounts:         --
```

In this example, we will use the IP address `10.49.93.209` to connect to the RDP server on the instance.

> **Note**
>
> If the IP address of the instance is not displayed in the output of `multipass list`, you can obtain it directly from the instance, with the command `ip addr`.

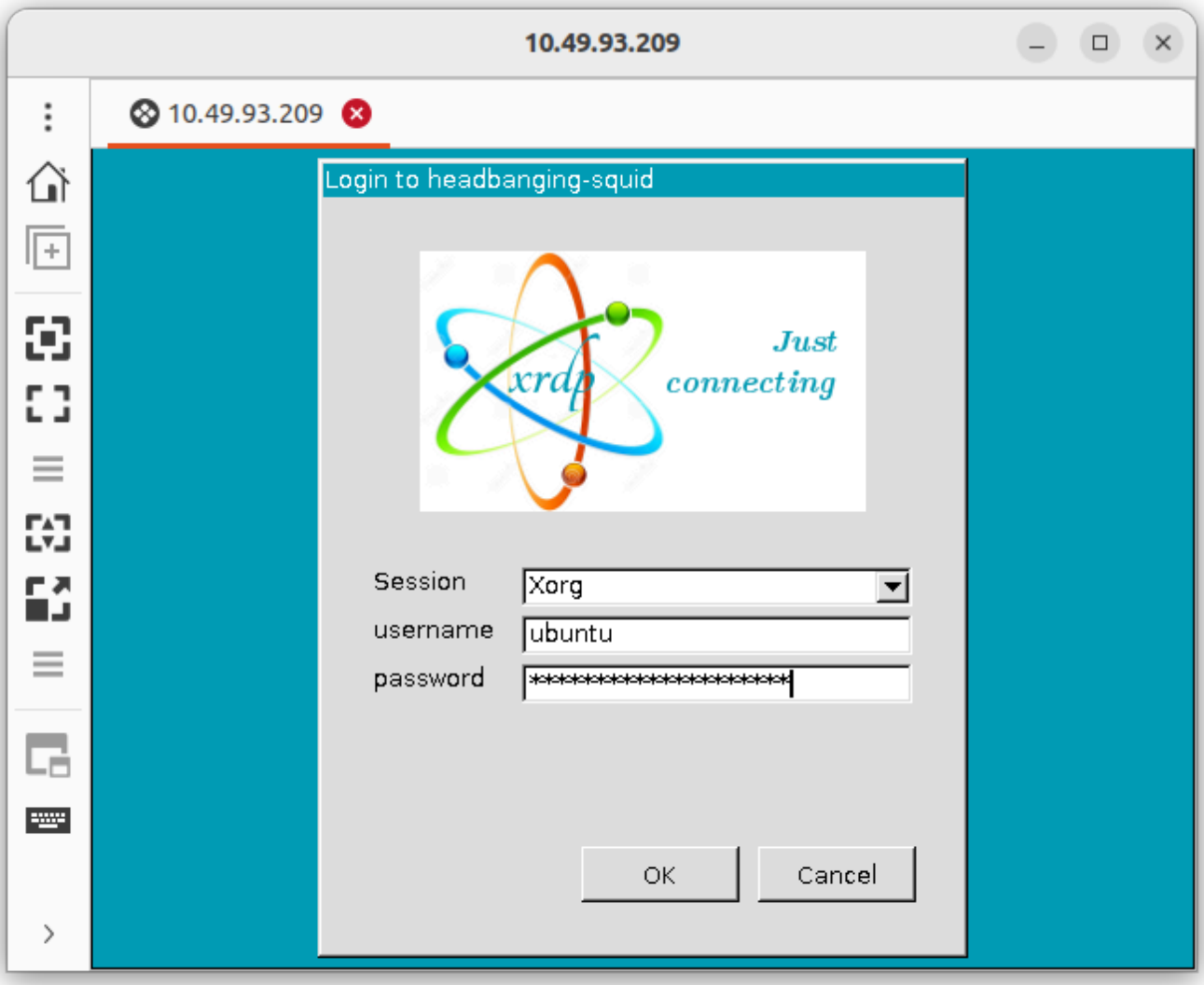
Linux macOS Windows

On Linux, there are applications such as Remmina to visualise the desktop (make sure the package `remmina-plugin-rdp` is installed in your host along with `remmina`).

To directly launch the client, run the following command:

```
remmina -c rdp://10.49.93.209
```

The system will ask for a username (`ubuntu`) and the password set above, and then the Ubuntu desktop on the instance will be displayed.



And we are done… a graphical desktop!

Using X11 forwarding

It might be the case that we only want Multipass to launch one application and to see only that window, without having the need for a complete desktop. It turns out that this setup is simpler than the RDP approach, because we do not need the Multipass instance to deploy a full desktop. Instead, we can use X11 to connect the applications in the instance with the graphical capabilities of the host.

Linux macOS Windows

Linux runs X by default, so no extra software in the host is needed.

On Linux, we can use authentication in X forwarding to add a bit more security. However, we will forward through SSH to avoid struggling with `xauth`. Our user in the host will log in to the Multipass instance through SSH, so that we can pass extra parameters to it.

To make this possible, copy your public key, stored in `~/.ssh/id_rsa.pub` , to the list of authorised keys of the instance, into the file `~/.ssh/authorized_keys` . Remember to replace the instance name used in the example with yours:

```
multipass exec headbanging-squid -- bash -c "echo `cat ~/.ssh/id_rsa.pub` >>
```

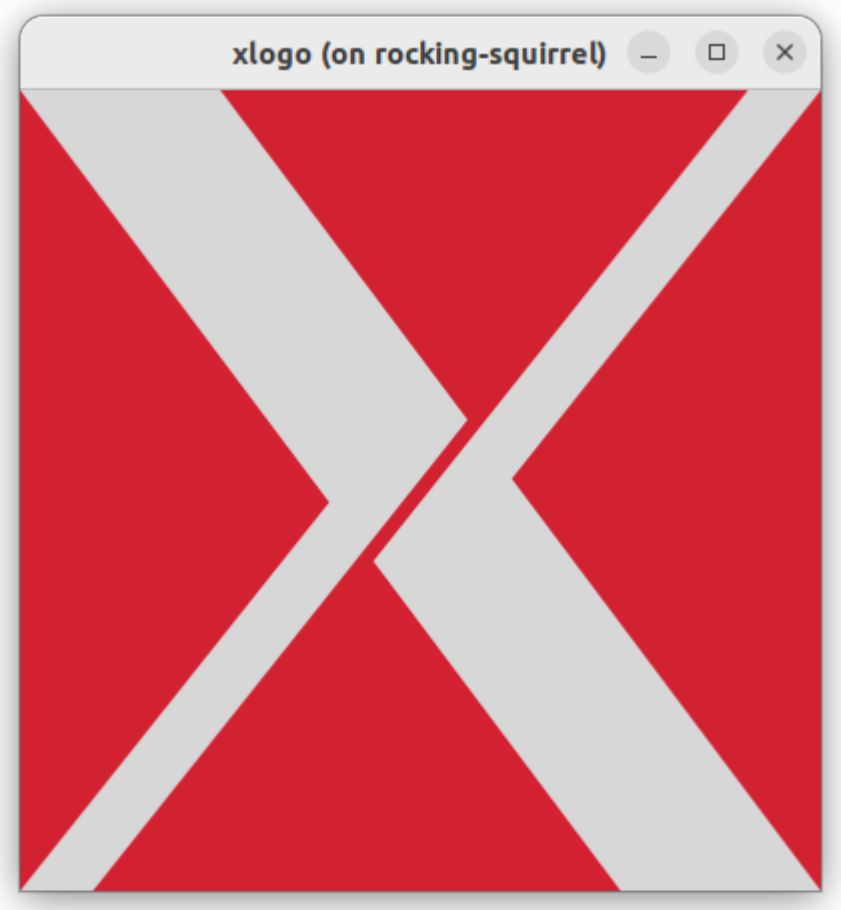
> **Note**
>
> If the file `~/.ssh/id_rsa.pub` does not exist, it means that you need to create your SSH keys. Use `ssh-keygen` to create them and then run the previous command again.

Check the IP address of the instance, using `multipass info headbanging-squid` Finally, log in to the instance using X forwarding using the command (replace `xx.xx.xx.xx` with the IP address obtained above):

```
ssh -X ubuntu@xx.xx.xx.xx
```

Test the setting running a program of your choice on the instance; for example:

```
sudo apt -y install x11-apps
xlogo &
```



A small window containing the X logo will show up. Done!

Last updated on Feb 19, 2025

Settings

> See also: Settings keys and values, `get`, `set`, GUI client

Multipass can be configured with a number of **settings** that are read and written by the `get` and `set` CLI commands, respectively. Some settings are also available in the GUI client.

Available settings

At any given time, the available settings depend on the state of the system. Some settings are only available on some platforms, while daemon settings can only be accessed when the Multipass daemon itself can be reached.

Some instance properties are also exposed as settings.

> See also: Set the CPUs , RAM or disk space of an instance

The command `multipass get --keys` shows what settings are available at any given time.

As of now, this is the total set of settings available:

- client.apps.windows-terminal.profiles
- client.primary-name
- local.bridged-network
- local.driver
- local.<instance-name>.bridged
- local.<instance-name>.cpus
- local.<instance-name>.disk
- local.<instance-name>.memory
- local.<instance-name>.<snapshot-name>.comment
- local.<instance-name>.<snapshot-name>.name
- local.passphrase
- local.privileged-mounts

Skip to content

> **Caution**
>
> Starting from Multipass version 1.14, the following settings have been removed from the CLI and are only available in the GUI client:
>
> - `client.gui.autostart`
> - `client.gui.hotkey`

Last updated on Feb 23, 2025

Share data with an instance

> See also: Instance, Mount, ID mapping, `launch`, `mount`, `umount`, `transfer`

This guide explains how to share data between your host and an instance. There are two ways to accomplish this:

- the `mount` command, that maps a local folder to a new or existing folder in the instance's filesystem
- the `transfer` command, that copies files to and from an instance

Using `mount`

You can use the `mount` command to share data between your host and an instance, by making specific folders in your host's filesystem available in your instance's filesystem, with read and write permissions. Mounted paths are persistent, meaning that they will remain available until they are explicitly unmounted.

The basic syntax of the `mount` command is:

```
multipass mount <local path> <instance name>
```

For example, to map your local home directory on a Linux system (identified as $HOME) into the `keen-yak` instance, run this command:

```
multipass mount $HOME keen-yak
```

You can check the result running `multipass info keen-yak`:

```
...
Mounts:          /home/michal => /home/michal
```

From this point the local home directory `/home/michal` will be available inside the instance.

To mount a local directory to a different path in your instance, you can specify the paths as follows:

```
multipass mount $HOME keen-yak:/some/path
```

> **Caution**
>
> If the `/some/path` directory already exists in the instance's filesystem, its contents will be temporarily hidden ("overlaid") by the mounted directory, but not overwritten. The original folder remains intact and will be revealed if you unmount.
>
> For this reason, it is not possible to mount an external folder path over the instance's $HOME directory, because it also contains the SSH keys required to access the instance: by hiding them, you would no longer be able to shell into the instance.

You can also define mounts when you create an instance, using the `launch` command with the `--mount` option:

```
multipass launch --mount /local/path:/instance/path
```

Unmounting shared directories

To unmount previously mounted paths, use the `umount` command.

You can specify the folder path to unmount:

```
multipass umount keen-yak:/home/michal
```

or, if you don't specify any paths, unmount all shared folders at once:

```
multipass umount keen-yak
```

Using `transfer`

You can also use the `transfer` command to copy files from your local filesystem to the instance's filesystem, and vice versa.

To indicate that a file is inside an instance, prefix its path with `<instance name>:`.

For example, to copy the `crontab` and `fstab` files from the `/etc` directory on the `keen-yak` instance to the `/home/michal` folder in the host's filesystem:

```
multipass transfer keen-yak:/etc/crontab keen-yak:/etc/fstab /home/michal
```

The files will be copied with the correct user mapping, as you'll see running the `ls -l` command:

```
...
-rw-r--r-- 1 michal michal 722 Oct 18 12:13 /home/michal/crontab
-rw-r--r-- 1 michal michal  82 Oct 18 12:13 /home/michal/fstab
...
```

The other way around, if you want to copy these files from your local filesystem into the instance, run the command:

```
multipass transfer /etc/crontab /etc/fstab keen-yak:/home/michal
```

In this case, the output of the `ls -l /home/michal` command on the instance will be:

```
...
-rw-rw-r-- 1 ubuntu ubuntu 722 Oct 18 12:14 crontab
-rw-rw-r-- 1 ubuntu ubuntu  82 Oct 18 12:14 fstab
...
```

See also ID mapping for more information on how the mount command maps user and group IDs between the host and the instance.

Troubleshoot launch/start issues

This topic addresses common issues when launching or starting instances, such as *timeouts or "unknown state" errors*.

These problems can occur for a few different reasons. Since Multipass relies on instances having an IP address on the default interface to establish an SSH connection, they are often (but not always) linked to IP assignment or connectivity issues.

The possible reasons that can lead the `launch` or `start` commands to fail are:

1. When you launch a new instance, it fails to load the required image due to stale network cache.
2. The VM didn't manage to boot properly and didn't get to the point where it requests an IP address.
3. The VM requested an IP address, but didn't obtain one.
4. The VM obtained an IP address, but:
   a. Multipass can't find it.
   b. Multipass finds an IP that doesn't match the one that was assigned to the instance.
5. SSH doesn't function properly in the VM, or Multipass is blocked from accessing it.
6. When you launch a new instance, it times out waiting for initialisation to complete.

Diagnose your issue

Follow these steps to diagnose your issue and identify the most likely scenario:

↑ Back to top

1. If the `multipass launch` command fails with the message “Downloaded image hash does not match”, see Stale network cache.
2. *(Windows, Hyper-V driver)* Inspect the file `C:\WINDOWS\System32\drivers\etc\hosts.ics` and see if there is more than one entry with your instance name in it. If that’s the case, see [Windows, with Hyper-V] Stale internet connection sharing lease.
3. *(Linux/macOS, QEMU driver)* Inspect the Multipass logs and look for a message mentioning `NIC_RX_FILTER_CHANGED`. This message indicates that the network interface has been initialised.
   - If you don’t find it, it means that the VM didn’t manage to bring up the interface; see VM boot failure.
   - If the message is present, proceed to check DHCP traffic in the next step.
4. *(Linux/macOS, QEMU driver)* Check DHCP traffic from your host to the instance, to find out if there are requests and replies. Adapt and run the following command *right after starting/launching* the instance:

   ```
   sudo tcpdump -i <bridge> udp port 67 and port 68
   ```

   You will need to replace `<bridge>` with `mpqemubr0` on Linux and with `bridge100` on macOS.

   > **Note**
   >
   > Note that, on macOS, `bridge100` is a virtual network interface that only appears when at least a VM is running.

   - If you see `NIC_RX_FILTER_CHANGED`, you should also see DHCP requests. If you don’t, see VM boot failure and please let us know.
   - If you see a DHCP request, but no reply, it means that the VM is still waiting for an IP address to be assigned; see No IP assigned.
   - If you see DHCP requests and replies, continue to the next step.
5. Look for messages regarding SSH in Multipass’s logs. The instance may have obtained an IP and/or be properly connected, but still refuse Multipass when it tries to SSH into it.
6. Look for the message in the CLI or GUI spinner. Once it reads “Waiting for initialisation to complete”, Multipass will have succeeded SSH-ing into the instance but remain waiting for cloud-init to finish.

Troubleshooting steps

VM boot failure

↑ Back to top

To find out if something is failing during boot, you'd need to attach to the VM's console/serial and observe the output and try to find out where the VM is getting stuck. Here is how you can do that, depending on the driver:

- *(Linux/macOS, QEMU driver)* Relaunch QEMU manually:
  1. Look for the `qemu-system-*` command line corresponding to the failing VM in Multipass logs.
  2. Copy it to an editor and modify it:
     a. Remove `-serial chardev:char0 -nographic`.
     b. Escape any spaces in paths (e.g. `Application Support` should become `Application\ Support`).
  3. Run the edited line in a terminal, with `sudo`. Here is an example:

```
/Library/Application\ Support/com.canonical.multipass/bin/qemu-system-aar
```

     This will open a QEMU window where you can see the boot output. You may need to select the correct display output (Serial or VGA) from the QEMU menu.

- *(macOS/Windows, VirtualBox driver)* Observe the output in the VirtualBox GUI:
  1. Run the VirtualBox GUI as admin/root:
     - [macOS] `sudo VirtualBox`
     - [Windows] Run with `psexec.exe` as explained in [Windows, VirtualBox driver] Running VirtualBox with the system account.
  2. Start or launch the instance with `multipass start|launch`.
  3. Select and attach to the VM in the VirtualBox GUI and observe the boot output. If it eventually arrives at a login screen, it means that the instance should've started correctly.

- *(Windows, Hyper-V driver)*
  1. Open the Hyper-V Manager GUI (look for it in your Start menu).
  2. Start or launch the instance with `multipass start|launch`.
  3. Select the VM in Hyper-V manager and click "Connect" on the Actions pane, at the right-hand side. Observe the boot output.

VM image corruption

Boot failures are often caused by VM image corruption, which can happen when the VM is killed without a proper shutdown.

Here are some options to attempt recovery:

- If you took a snapshot before incurring this issue, you could try to restore it. However, snapshots are typically stored layers ag[Back to top]al image file, so they may not be enough.
- **Run `fsck` in the Serial Console:**

  The `fsck` tool (short for “file system consistency check”) is used to scan the file system for errors and attempt repairs.

To use it, access the VM's console as described above and follow these steps:

1. **Access the VM's Console**
   - Use the method appropriate for your setup to access the VM's console, as described in the VM boot failure section.
2. **Interrupt the Boot Process**
   - As the VM starts booting, interrupt the boot process to access the GRUB menu:
     - Press the `Esc` key repeatedly during the VM's startup until the GRUB menu appears.
     - On some systems, you might need to hold down the `Shift` key instead.
     - The key needs to be pressed at just the right time, after UEFI loading (to avoid getting into the UEFI screen), but before Ubuntu starts booting (to trigger GRUB)
3. **Enter Recovery Mode**
   - In the GRUB menu:
     - Use the arrow keys to select **Advanced options for Ubuntu** (or your distribution's equivalent) and press `Enter`.
     - Select a kernel version with `(recovery mode)` appended and press `Enter`.
4. **Run `fsck` from Recovery Menu**
   - Once in the recovery menu:
     - Use the arrow keys to highlight **`fsck`** and press `Enter`.
     - You will be prompted to remount the filesystem in read/write mode. Select **Yes**.
     - The system will run `fsck` and attempt to repair any detected issues.
5. **Alternatively, Drop to a Root Shell**
   - If you prefer to run `fsck` manually:
     - From the recovery menu, select **`root`** to drop to a root shell prompt.
     - At the prompt, run the following commands:

       ```
       mount -o remount,ro /
       fsck -f /
       ```

     - After `fsck` completes, remount the filesystem in read/write mode:

       ```
       mount -o remount,rw /
       ```

     - Type `exit` to return to the recovery menu.
6. **Resume Normal Boot**
   - In the recovery menu, select **`resume`** to continue with the normal boot process.
   - The system should now boot normally if `fsck` was able to repair the filesystem.

- *(Linux/macOS)* Alternatively, run `fsck` over a mounted image on the host (see Reading data from a QCOW2 image).
- Run `qemu-img check -r` on the ima
  - `qemu-img`, shipped with Multipass, can also be used to check and repair disk images.
  - See Locating Multipass binaries below.
  - See Locating Multipass images below.
  - For example:

```
/Library/Application\ Support/com.canonical.multipass/bin/qemu-img check
```

- If none of the above works, you can still try to mount the image manually to recover data (see Reading data from a QCOW2 image).

↑ Back to top

No IP assigned

Sometimes VMs request an IP address, but don't obtain one. That can happen because of interference from other software, VPNs, network misconfiguration and, firewall settings.

[macOS, QEMU driver] Firewall blocks `bootp`

The macOS firewall is known to cause `vmnet` to malfunction, because it blocks Apple's own `bootp` from giving out IPs. The effect of this problem on Multipass is tracked in this issue, which we internally call the *dreaded firewall issue*.

You may be able to work around it by disabling the firewall entirely, or executing

```
/usr/libexec/ApplicationFirewall/socketfilterfw --add /usr/libexec/bootpd
/usr/libexec/ApplicationFirewall/socketfilterfw --unblock /usr/libexec/boot
```

We are aware that this requires administrative privileges, which managed Macs won't have. We unfortunately don't have a better fix for those cases. We continue hoping that Apple will eventually fix the problem which, to the best of our knowledge, affects all products using `vmnet`. Chances of that happening will probably increase if enough people report it to them.

> See also: How to troubleshoot networking.

[macOS, QEMU driver] `bootp` not coming up

The DHCP server should be launched automatically when there is a request, but you can also launch it manually if needed. To do so, run:

```
sudo launchctl start com.apple.bootpd
```

If that doesn't work for you, try :

```
sudo launchctl load -w /System/...nchDaemons/bootps.plist
```

↑ Back to top

[Windows, with Hyper-V] Stale internet connection sharing lease

Another possible reason for instance timeouts is a problem with the `Internet Connection Sharing` hosts file. This file sometimes gets corrupted, with jumbled or incomplete text. Other times, it contains duplicate or stale IP addresses.

Using Administrator privileges, edit the file `C:\WINDOWS\System32\drivers\etc\hosts.ics` and look for any corruption or entries that have your instance name in it. If there is more than one entry, remove any of them except for the first one listed. Save the file and try again. If that does not work, stop any running instances, delete the file, and reboot.

SSH issues

If SSH doesn't function properly in the VM, or Multipass is blocked from accessing it, your instance may need to be reconfigured or repaired.

- If the default user is not `ubuntu` , Multipass cannot connect. If you used a custom cloud-init config file, make sure that the default user is `ubuntu` .
- if SSH keys are missing or incorrect, you will have to add your public SSH key from `~/.ssh/id_rsa.pub` on the host to `~/.ssh/authorized_keys` in the instance. To do so you may need to gain access to the instance through a method besides SSH.

To gain access to an instance without SSH you can try the following methods.

- Mount the instance's image file on your host (see Reading data from a QCOW2 image) and make necessary changes through the filesystem.
- Run the instance VM directly. This will require a username and password to log in. The username is the default user, `ubuntu` , and the password is what was set in cloud-init if you used a custom cloud-init config. If you do not have a password you can modify the instance's `cloud-init-config.iso` file to change it. One way to do so is as follows.
  1. Back up your existing `cloud-init-config.iso` .
  2. Make a new instance by running `multipass launch --cloud-init config.yaml` , the contents `config.yaml` are shown below.
  3. Replace your existing `cloud-init-config.iso` with the newly generated `cloud-init-config.iso` .
  4. Restart the VM and use the password `ubuntu` . The instance's password will remain `ubuntu` unless it is changed again
  5. Make necessary changes.

```
#cloud-config
password: ubuntu
chpasswd: { expire: false }
```

↑ Back to top

Cloud-init tarries during an instance launch

- When launching a new instance, once Multipass obtains an SSH session to the instance, it will wait for cloud-init to complete. During this phase, the CLI/GUI spinner reads “Waiting for initialisation to complete”.
- At this point, the initialisation continues in the background, even if you interrupt the launch command or if it times out.
- So if you wait for a little while longer, your instance may eventually finish setting up. When it does, it will have this file: `/var/lib/cloud/instance/boot-finished`.
  - Consider passing a longer timeout to the `launch` command. For example, `multipass launch --timeout 1000` sets the launch timeout to 1000 seconds.
- You can use `multipass shell` to get a shell in the instance when Multipass is waiting for cloud-init to finish. To diagnose problems, inspect cloud-init’s logs in `/var/log/cloud-init*log`.

[Windows, VirtualBox driver] Running VirtualBox with the system account

To run the VirtualBox GUI with the system account, you will need a Windows tool called PsExec:

1. Install PsExec.
2. Add it to your `PATH`.
3. Run PowerShell as Administrator.
4. Execute `psexec.exe -s -i "C:\Program Files\Oracle\VirtualBox\VirtualBox.exe"` (adapt the path to the VirtualBox executable as needed).

When successful, you should see Multipass’s instances in VirtualBox

Stale network cache

This can be caused by a known Qt bug (see issue #1714 on our GitHub).

A workaround to resolve this issue is to run the command `multipass find --force-update`, which forces downloading the image information from the network. As a result, if the download is successful, the `network-cache` will be overwritten.

Alternatively, try deleting the `network-cache` folder and restart the Multipass service:

- *(on Linux)*

↑ Back to top

```
sudo snap stop multipass
sudo rm -rf /var/snap/multipass/common/cache/multipassd/network-cache/
sudo snap start multipass
```

- *(on macOS)*

```
sudo launchctl unload /Library/LaunchDaemons/com.canonical.multipassd.pli
sudo rm -rf /System/Volumes/Data/private/var/root/Library/Caches/multipas
sudo launchctl load /Library/LaunchDaemons/com.canonical.multipassd.plist
```

- *(on Windows)* Remove `C:\ProgramData\Multipass\cache\network-cache` and restart the Multipass service.

Reading data from a QCOW2 image

The images that Multipass uses with the QEMU driver follow a standard format — QCOW2 — which other tools can read.

One example is `qemu-nbd`, which allows mounting the image. This tool is not shipped with Multipass, so you would need to install it manually.

Once you have it, you can search the web for recipes to mount a QCOW2 image. For example, here is a a recipe.

Locating Multipass binaries

You may need to locate where Multipass is installed. There are several ways to do so, depending on your platform:

↑ Back to top

- *(on Linux)*
  - Run the command `which multipass` or `whereis multipass`.
  - By default, Multipass is installed in the `/snap/bin` folder.
- *(on Windows)*
  - Run the command `where.exe multipass`.
  - Right-click a shortcut to Multipass in your files or Start menu and select “Open file location”.
  - By default, Multipass is installed in the `C:\Program Files\Multipass\bin` folder.
- *(on macOS)*
  - Run the command `readlink -f $(which multipass)`
  - By default, Multipass is installed in the `/Library/Application\ Support/com.canonical.multipass/bin/` folder.

Locating Multipass images

You may need to locate where Multipass is storing instances. The location changes depending on your platform:

- *(Linux)* `/root/.local/share/multipassd/vault/instances/<instance/<img>`
- *(Windows)* `C:\ProgramData\Multipass\data\vault\instances\<instance>\<img>`
- *(macOS)* `/var/root/Library/Application\ Support/multipassd/qemu/vault/instances/<instance>/<img>`

Last updated on Feb 24, 2025

Troubleshoot networking

This document demonstrates how to troubleshoot various known Multipass networking issues on macOS and Windows.

Troubleshoot networking on macOS

Architecture

On macOS, the QEMU driver employs the Hypervisor.framework. This framework manages the networking stack for the instances.

On creation of an instance, the Hypervisor framework on the host uses macOS's **Internet Sharing** mechanism to:

1. Create a virtual switch and connect each instance to it (subnet 192.168.64.*).
2. Provide DHCP and DNS resolution on this switch at 192.168.64.1 (via bootpd & mDNSResponder services running on the host). This is configured by an auto-generated file (`/etc/bootpd.plist`), but editing this is pointless as macOS regenerates it as it desires.

Note that, according to **System Preferences > Sharing**, the **Internet Sharing** service can appear disabled. This is fine—in the background, it will still be enabled to support instances.

Tools known to interfere with Multipass

- VPN software can be aggressive at managing routes and may route 192.168.64 subnet through the VPN interface, instead of keeping it locally available.
  - Possible culprits: OpenVPN, F5, Dell SonicWall, Cisco AnyConnect, Citrix/Netscaler Gateway, Jupiter Junos Pulse / Pulse Secure.
  - Tunnelblick doesn't cause problems.
- Cisco Umbrella Roaming Client it binds to localhost:53 which clashes with Internet Sharing, breaking the instance's DNS.
- dnscrypt-proxy/dnscrypt-wrapper/cloudflared-proxy
  The default configuration binds to localhost port 53, clashing with Internet Sharing.
- Another `dnsmasq` process bound to localhost port 53
- Custom DHCP server bound to port 67? ("sudo lsof -iUDP:67 -n -P" should show launchd &
- macOS updates can make changes to the firewall and leave instances in unknown state; see Issues caused by macOS update below.

Problem class

- `multipass launch` fails
  - see Generic networking problems
- `multipass shell <instance>` fails
  - see Network routing problems
- `multipass shell <instance>` works but the instance cannot connect to the internet
  - see DNS problems
- extra IPs not reachable between instances
  - see ARP problems

Generic networking problems

`Unable to determine IP address` usually implies some networking configuration is incompatible, or there is interference from a Firewall or VPN.

> See also: How to troubleshoot launch/start issues.

Troubleshooting

1. Firewall
   a. Is Firewall enabled?
   b. If so it must not “Block all incoming connections”
   - Blocking all incoming connections prevents a DHCP server from running locally, to give an IP to the instance.
   - It’s OK to block incoming connections to “multipassd” however.
2. VPN
3. Little Snitch - defaults are good, it should permit mDNSResponder and bootpd access to BPF
   If you’re having trouble downloading images and/or see `Unknown error` s when trying to `multipass launch -vvv` , Little Snitch may be interfering with `multipassd` ’s network access (ref. #1169)
4. Internet Sharing - doesn’t usually clash

Network routing problems

You could try:

```
sudo route -nv add -net 192.168.64.0/24 -interface bridge100
```

Skip to content xists” error, maybe delete and retry? latest

```
sudo route -nv delete -net 192.168.64.0/24
sudo route -nv add -net 192.168.64.0/24 -interface bridge100
```

Maybe `-static` route helps?

If using Cisco AnyConnect, try using OpenConnect (`brew install openconnect`) instead as it messes with routes less (but your company sysadmin/policy may not permit/authorise this).

- It monitors the routing table so may prevent any customisation. Here is a very hacky workaround.

Does your VPN software provide a “split connection” option, where VPN sysadmin can designate a range of IP addresses to **not** be routed through the VPN?

- Cisco does
- Pulse Secure / Jupiter Junos Pulse do

Potential workaround for VPN conflicts

This was reported on GitHub (issue #495).

After the `nat ...` line (if there is one, otherwise at the end) in `/etc/pf.conf`, add this line:

```
nat on utun1 from bridge100:network to any -> (utun1)
```

and reload PF with the command:

```
sudo pfctl -f /etc/pf.conf
```

Configure Multipass to use a different subnet

Edit `/Library/Preferences/SystemConfiguration/com.apple.vmnet.plist` to change the “Shared_Net_Address” value to something other than `192.168.64.1 -`.

- it works if you edit the plist file and stay inside 192.168 range, as Multipass hardcoded for this

> **Note**
>
> If you change the subnet and launch an instance, it will get an IP from that new subnet. But if you try changing it back, the change is reverted on next instance start. It appears that the DHCP server reads the last IP in `/var/db/dhcpd_leases`, decides the subnet from that, and updates Shared_Net_Address to match. So, the only way to really revert this change is to edit or delete `/var/db/dhcpd_leases`.

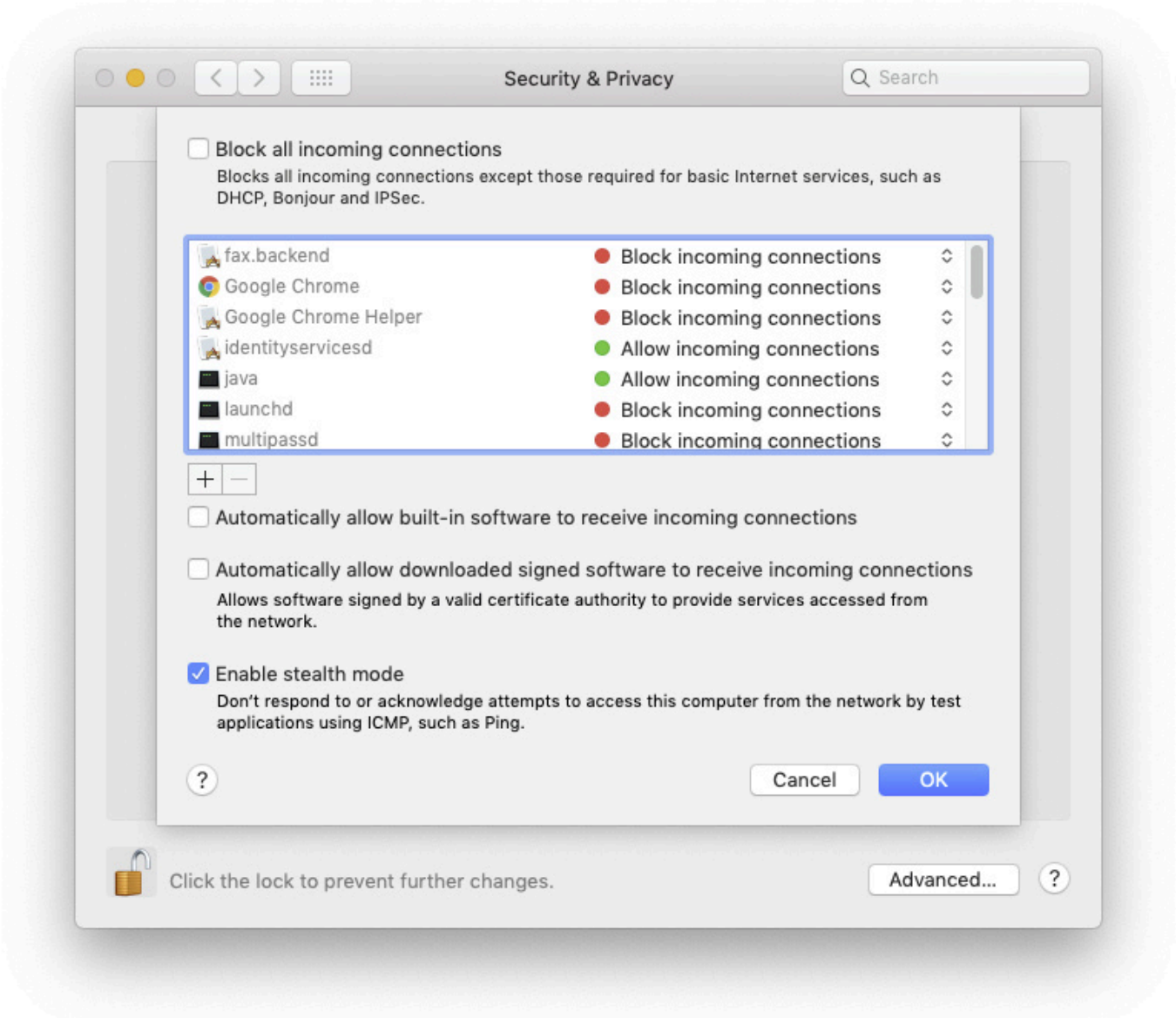
Skip to content
ddresses?
latest

```
ping 1.1.1.1
```

If not, the output will be similar to the following:

```
PING 1.1.1.1 (1.1.1.1) 56(84) bytes of data.
^C
--- 1.1.1.1 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2030ms
```

Note that macOS's firewall can block the ICMP packets that `ping` uses, which will interfere with this test. Make sure you disable **Stealth Mode** in **System Preferences > Security & Privacy > Firewall** just for this test.



If you try again, it should work:


```
ping 1.1.1.1
```

The output will be similar to the following:

```
PING 1.1.1.1 (1.1.1.1) 56(84) bytes of data.
64 bytes from 1.1.1.1: icmp_seq=1 ttl=53 time=7.02 ms
64 bytes from 1.1.1.1: icmp_seq=2 ttl=53 time=5.91 ms
64 bytes from 1.1.1.1: icmp_seq=3 ttl=53 time=5.12 ms
^C
--- 1.1.1.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2143ms
rtt min/avg/max/mdev = 5.124/6.020/7.022/0.781 ms
```

This means the instance can indeed connect to the internet, but DNS resolution is broken. You can test DNS resolution using the `dig` tool:

```
dig @192.168.64.1 google.ie
```

If broken, the output will be similar to:

```
; <<>> DiG 9.10.3-P4-Ubuntu <<>> google.ie
;; global options: +cmd
;; connection timed out; no servers could be reached
```

On the other hand, if it works correctly the output will be similar to:

```
; <<>> DiG 9.10.3-P4-Ubuntu <<>> google.ie
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 48163
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;google.ie.                     IN      A

;; ANSWER SECTION:
google.ie.              15      IN      A       74.125.193.94

;; Query time: 0 msec
;; SERVER: 192.168.64.1#53(192.168.64.1)
;; WHEN: Thu Aug 01 15:17:04 IST 2019
;; MSG SIZE  rcvd: 54
```

To test further, try supplying an explicit DNS server:

```
dig @1.1.1.1 google.ie
```

If it works correctly, the output will be similar to:

```
; <<>> DiG 9.10.3-P4-Ubuntu <<>> @1.1.1.1 google.ie
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 11472
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1452
;; QUESTION SECTION:
;google.ie.                IN    A

;; ANSWER SECTION:
google.ie.        39    IN    A    74.125.193.94

;; Query time: 6 msec
;; SERVER: 1.1.1.1#53(1.1.1.1)
;; WHEN: Thu Aug 01 15:16:27 IST 2019
;; MSG SIZE  rcvd: 54
```

This implies the problem is with the macOS **Internet Sharing** feature—for some reason, its built-in DNS server is broken.

The built-in DNS server should be “mDNSResponder”, which binds to localhost on port 53.

If using Little Snitch or another per-process firewall, ensure mDNSResponder can establish outgoing connections. The macOS’s built-in firewall should not interfere with it.

Check what is bound to that port on the host with:

```
sudo lsof -iTCP:53 -iUDP:53 -n -P
```

The sample output below shows the correct state while a instance is running:

```
sudo lsof -iTCP:53 -iUDP:53 -n -P
COMMAND    PID           USER   FD   TYPE             DEVICE SIZE/OFF NOD
mDNSRespo  191 _mdnsresponder   17u  IPv4 0xa89d451b9ea11d87      0t0  UDP *:
mDNSRespo  191 _mdnsresponder   25u  IPv6 0xa89d451b9ea1203f      0t0  UDP *:
mDNSRespo  191 _mdnsresponder   50u  IPv4 0xa89d451b9ea8b8cf      0t0  TCP *:
mDNSRespo  191 _mdnsresponder   55u  IPv6 0xa89d451b9e2e200f      0t0  TCP *:
```

If no instance is running (and **Internet Sharing** is disabled in **System Preferences**), the command should return nothing.

Any other command appearing in that output means a process is conflicting with **Internet Sharing** and thus will break DNS in the instance.

Possible workarounds

1. Configure DNS inside the instance to use an external working DNS server. Can do so by appending this line to /etc/resolv.conf manually:

   ```
   nameserver 1.1.1.1
   ```

   “1.1.1.1” is a free DNS service provided by CloudFlare, but you can use your own.

2. Use a custom cloud-init to set /etc/resolv.conf for you on first boot.

ARP problems

The macOS bridge by Multipass filters packets so that only the IP address originally assigned to the VM is allowed through. If you add an additional address (e.g. an IP alias) to the VM, the ARP broadcast will get through but the ARP response will be filtered out.

This means that applications that rely on additional IP addresses, such as metallb under microk8s, will not work.

Issues caused by macOS update

When upgrading macOS to 12.4 (this might happen however also when upgrading to other versions), macOS makes changes to the firewall. If the instances are not stopped before the update, it is possible the connection to the instances are blocked by the macOS firewall. We cannot know what is exactly the change introduced to the firewall, it seems the Apple’s `bootpd` stops replying DHCP requests.

There are some procedures that can help to overcome this issue (see issue #2387 on the Multipass GitHub repo for a discussion on this and some alternative solutions). First, you can try to:

- Reboot the computer.
- Disable and then re-enable Internet sharing and/or the firewall.
- Configure the driver (QEMU) and Multipass in the firewall to allow incoming connections.

Troubleshoot networking on Windows

This sections contains troubleshooting considerations that are specific to Windows systems.

Architecture

Multipass uses the native “Hyper-V” hypervisor on Windows, along with the “Default Switch” created for it. That, in turn, uses the “Internet Sharing” functionality, providing DHCP (IP addresses) and DNS (domain name resolution) to the instances.

Known issues

Here you can find more details on known issues affecting Windows systems.

Default switch going awry

Unfortunately the default switch is known to be quirky and Windows updates often put it in a weird state. This may result in new instances failing to launch and existing ones timing out to start.

The broken state also persists over reboots. The one approach that has helped is removing the network sharing from the default switch:

```
Get-HNSNetwork | ? Name -Like "Default Switch" | Remove-HNSNetwork
```

and then rebooting the system:

```
Restart-Computer
```

Hyper-V will recreate it on next boot.

Anti-virus / security software blocking instances

Anti-virus and network security software are not necessarily virtualisation-aware. If you’re having issues with connectivity, temporarily disabling this software to test can result in a positive outcome. Examples of this software are Symantec, ESET, Kaspersky and Malware Bytes.

Last updated on Feb 27, 2025

latest

Tutorial

Multipass is a flexible and powerful tool that can be used for many purposes. In its simplest form, you can use it to quickly create and destroy Ubuntu VMs (instances) on any host machine. But you can also use Multipass to build a local mini-cloud on your laptop, to test and develop multi-instance or container-based cloud applications.

This tutorial will help you understand how Multipass works, and the skills you need to use its main features.

Install Multipass

Multipass is available for Linux, macOS and Windows. To install it on the OS of your choice, please follow the instructions provided in How to install Multipass.

> **Note**
>
> Select the tab corresponding to your operating system (e.g. Linux) to display the relevant content in each section. Your decision will stick until you select another OS from the drop-down menu.

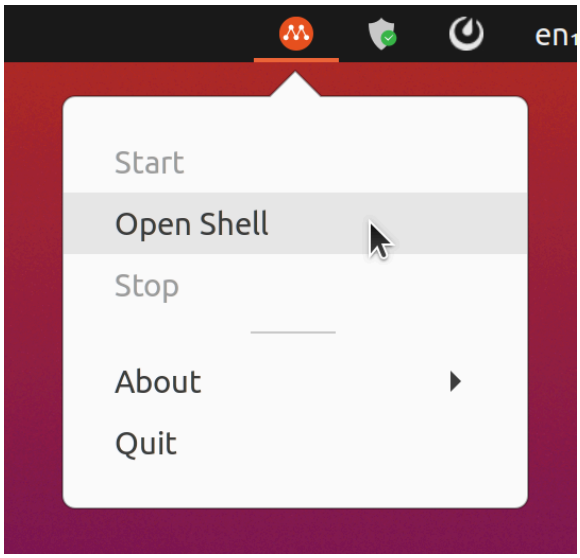
Create and use a basic instance

Linux macOS Windows

Start Multipass from the application launcher. In Ubuntu, press the super key and type “Multipass”, or find Multipass in the Applications panel on the lower left of the desktop.

After launching the application, you should see the Multipass tray icon on the upper right section of the screen.

Click on the Multipass icon and select **Open Shell**.



Clicking this button does many things in the background:

- It creates a new virtual machine (instance) named `primary`, with 1GB of RAM, 5GB of disk, and 1 CPU. See also: Primary instance
- ost recent Ubuntu LTS release on that instance.
- `$HOME` directory in the instance.
- It opens a shell to the instance, announced by the command prompt `ubuntu@primary`.

> **Caution**
>
> If your local home folder is encrypted using `fscrypt` and you are having trouble accessing its contents when it is automatically mounted inside your primary instance, see Mount an encrypted home folder.

You can see elements of this in the printout below:

```
Launched: primary
Mounted '/home/<user>' into 'primary:Home'
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-57-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Thu Jan 26 08:06:22 PST 2023

  System load:  0.0               Processes:             95
  Usage of /:   30.2% of 4.67GB   Users logged in:       0
  Memory usage: 21%               IPv4 address for ens3: 10.110.66.242
  Swap usage:   0%

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8
   just raised the bar for easy, resilient and secure K8s cluster deployment

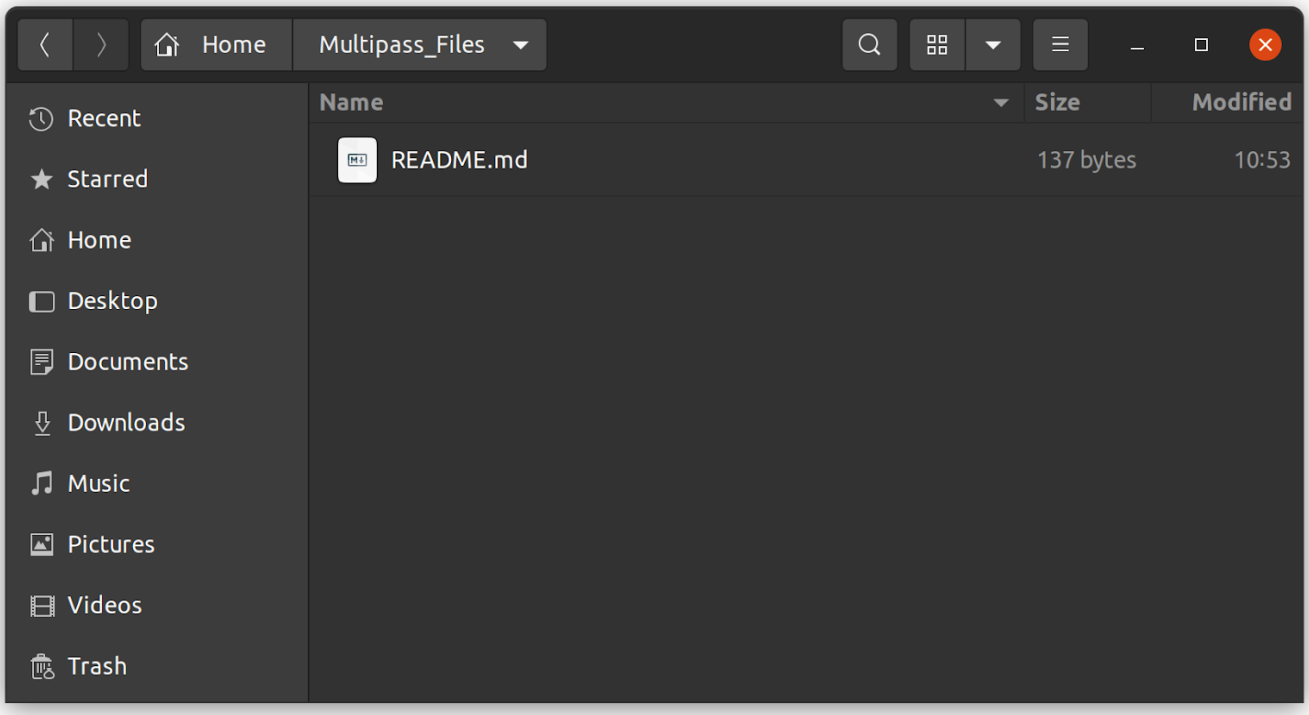
   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

0 updates can be applied immediately.


The list of available updates is more than a week old.
To check for new updates run: sudo apt update

ubuntu@primary:~$
```

Let's test it. As you've just learnt, the previous step automatically mounted your `$HOME` directory in the instance. Use this to share data with your instance. More concretely, create a new folder called `Multipass_Files` in your `$HOME` directory:



As you can see, a `README.md` file has been added to the shared folder. Check for the folder and read the file from your new instance:

```
cd ./Home/Multipass_Files/
cat README.md
```

Sample output:

```
## Shared Folder

This folder could be a great place to keep files that need to be accessed by
```

Congratulations, you've got your first instance!

This instance is great for when you just need a quick Ubuntu VM, but let's say you want a more customised instance, how can you do that? Multipass has you covered there too.

Optional Exercises >

Create a customised instance

Skip to content eat feature to help you get started with creating customis latest

the `multipass find` command. The result shows a list of all images you can currently launch through Multipass.

```
$ multipass find
Image                       Aliases           Version          Description
snapcraft:core18            18.04             20201111         Snapcraft bu
snapcraft:core20            20.04             20210921         Snapcraft bu
snapcraft:core22            22.04             20220426         Snapcraft bu
snapcraft:devel                               20230126         Snapcraft bu
core                        core16            20200818         Ubuntu Core
core18                                        20211124         Ubuntu Core
core20                                        20230119         Ubuntu Core
core22                                        20230119         Ubuntu Core
18.04                       bionic            20230112         Ubuntu 18.04
20.04                       focal             20230117         Ubuntu 20.04
22.04                       jammy,lts         20230107         Ubuntu 22.04
22.10                       kinetic           20230112         Ubuntu 22.10
daily:23.04                 devel,lunar       20230125         Ubuntu 23.04
appliance:adguard-home                        20200812         Ubuntu AdGua
appliance:mosquitto                           20200812         Ubuntu Mosqu
appliance:nextcloud                           20200812         Ubuntu Nextc
appliance:openhab                             20200812         Ubuntu openH
appliance:plexmediaserver                     20200812         Ubuntu Plex
anbox-cloud-appliance                         latest           Anbox Cloud
charm-dev                                     latest           A developmen
docker                                        0.4              A Docker env
jellyfin                                      latest           Jellyfin is
minikube                                      latest           minikube is
```

Launch an instance running Ubuntu 22.10 (“Kinetic Kudu”) by typing the `multipass launch kinetic` command.

Linux macOS Windows

Now, you have an instance running and it has been named randomly by Multipass. In this case, it has been named “coherent-trumpetfish”.

```
$ multipass launch kinetic
Launched: coherent-trumpetfish
```

You can check some basic info about your new instance by running the following command:

Skip to content

```
coherent-trumpetfish -- lsb_release -a
```

latest

This tells multipass to run the command `lsb_release -a` on the “coherent-trumpetfish” instance.

```
$ multipass exec coherent-trumpetfish -- lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 22.10
Release:        22.10
Codename:       kinetic
```

Perhaps after using this instance for a while, you decide that what you really need is the latest LTS version of Ubuntu, with a more informative name and a little more memory and disk. You can delete the “coherent-trumpetfish” instance by running the following command:

```
multipass delete coherent-trumpetfish
```

You can now launch the type of instance you need by running this command:

```
multipass launch lts --name ltsInstance --memory 2G --disk 10G --cpus 2
```

Manage instances

You can confirm that the new instance has the specs you need by running `multipass info ltsInstance`.

Linux macOS Windows

```
$ multipass info ltsInstance
Name:           ltsInstance
State:          Running
IPv4:           10.110.66.139
Release:        Ubuntu 22.04.1 LTS
Image hash:     3100a27357a0 (Ubuntu 22.04 LTS)
CPU(s):         2
Load:           1.11 0.36 0.12
Disk usage:     1.4GiB out of 9.5GiB
Memory usage:   170.4MiB out of 1.9GiB
Mounts:         --
```

 d deleted quite a few instances. It is time to run `multipas` instances you currently have.

```
$ multipass list
Name                    State             IPv4             Image
primary                 Running           10.110.66.242    Ubuntu 22.04 LTS
coherent-trumpetfish    Deleted           --               Not Available
ltsInstance             Running           10.110.66.139    Ubuntu 22.04 LTS
```

The result shows that you have two instances running, the “primary” instance and the LTS machine with customised specs. The “coherent-trumpetfish” instance is still listed, but its state is “Deleted”. You can recover this instance by running `multipass recover coherent-trumpetfish`. But for now, delete the instance permanently by running `multipass purge`. Then run `multipass list` again to confirm that the instance has been permanently deleted.

```
$ multipass list
Name                    State             IPv4             Image
primary                 Running           10.110.66.242    Ubuntu 22.04 LTS
ltsInstance             Running           10.110.66.139    Ubuntu 22.04 LTS
```

You’ve now seen a few ways to create, customise, and delete an instance. It is time to put those instances to work!

Put your instances to use

Let’s see some practical examples of what you can do with your Multipass instances:

- Run a simple web server
- Launch from a Blueprint to run Docker containers (deprecated)

Run a simple web server

One way to put a Multipass instance to use is by running a local web server in it.

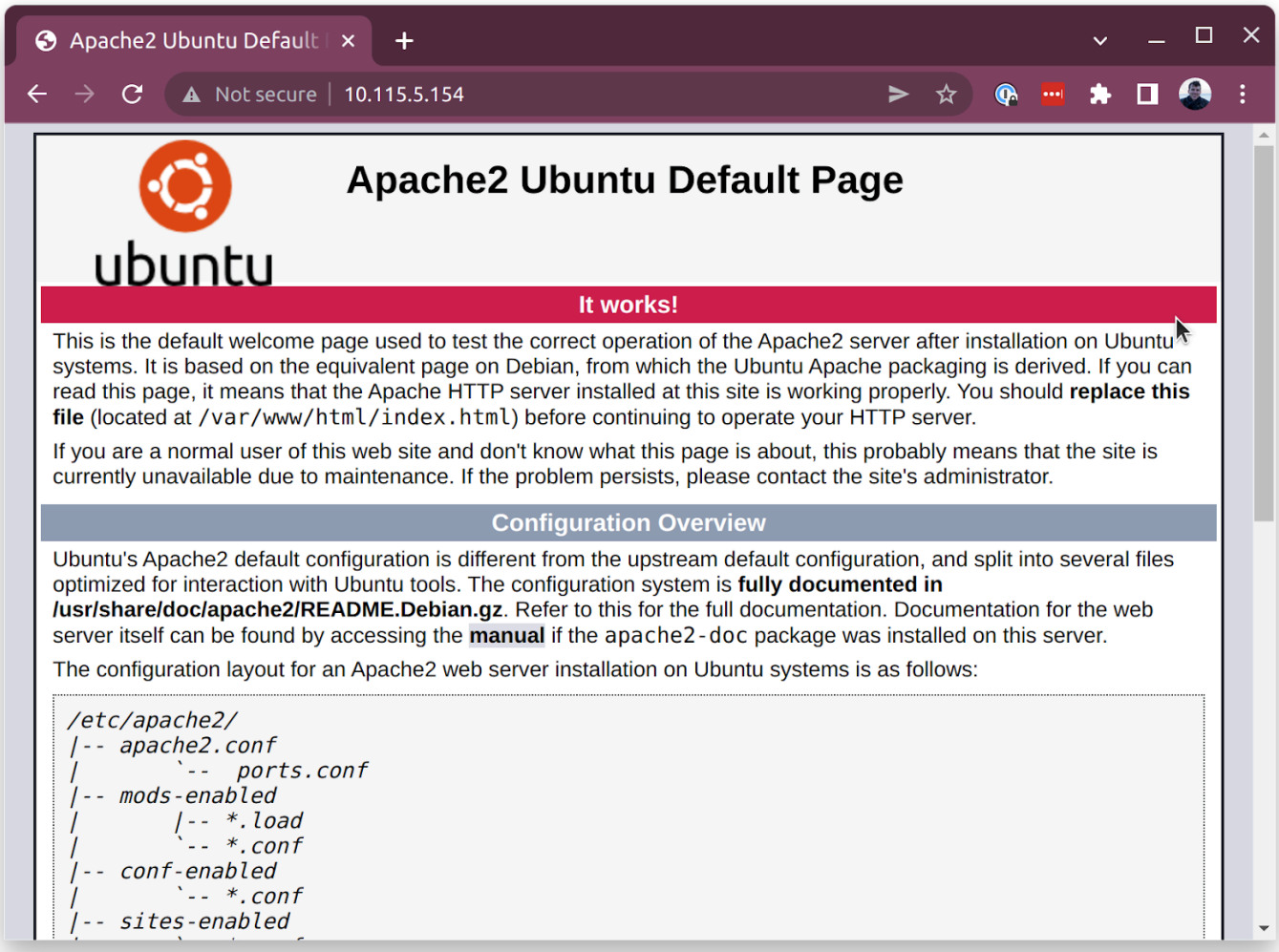
Return to your customised LTS instance. Take note of its IP address, which was revealed when you ran `multipass list`. Then run `multipass shell ltsInstance` to open a shell in the instance.

From the shell, you can run:

```
sudo apt update
sudo apt install apache2
```

Open a browser and type in the IP address of your instance into the address bar. You should now see the default Apache homepage.

Linux macOS Windows



Just like that, you've got a web server running in a Multipass instance!

You can use this web server locally for any kind of local development or testing. However, if you want to access this web server from the internet (for instance, a different computer), you need an instance that is exposed to the external network.

Launch from a Blueprint to run Docker containers (deprecated)

> **Warning**
>
> Blueprints are deprecated and will be removed in a future release. You can achieve similar effects with cloud-init and other launch options.

ts require a lot of configuration and setup. Multipass Blue

with a deep level of customisation. For example, the Docker Blueprint is a pre-configured Docker environment with a Portainer container already running.

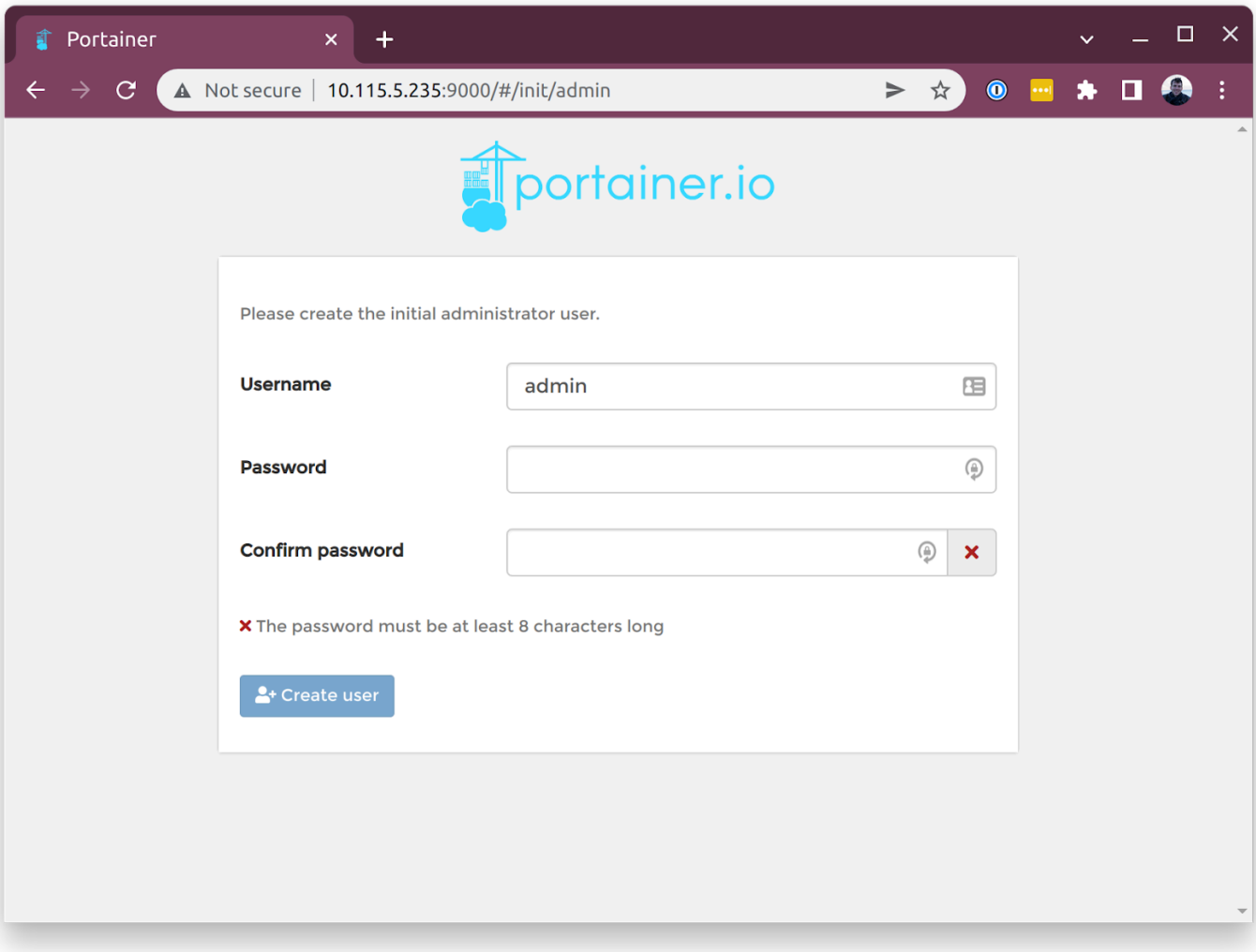
You can launch an instance using the Docker Blueprint by running `multipass launch docker --name docker-dev`.

Once that's done, run `multipass info docker-dev` to note down the IP of the new instance.

Linux macOS Windows

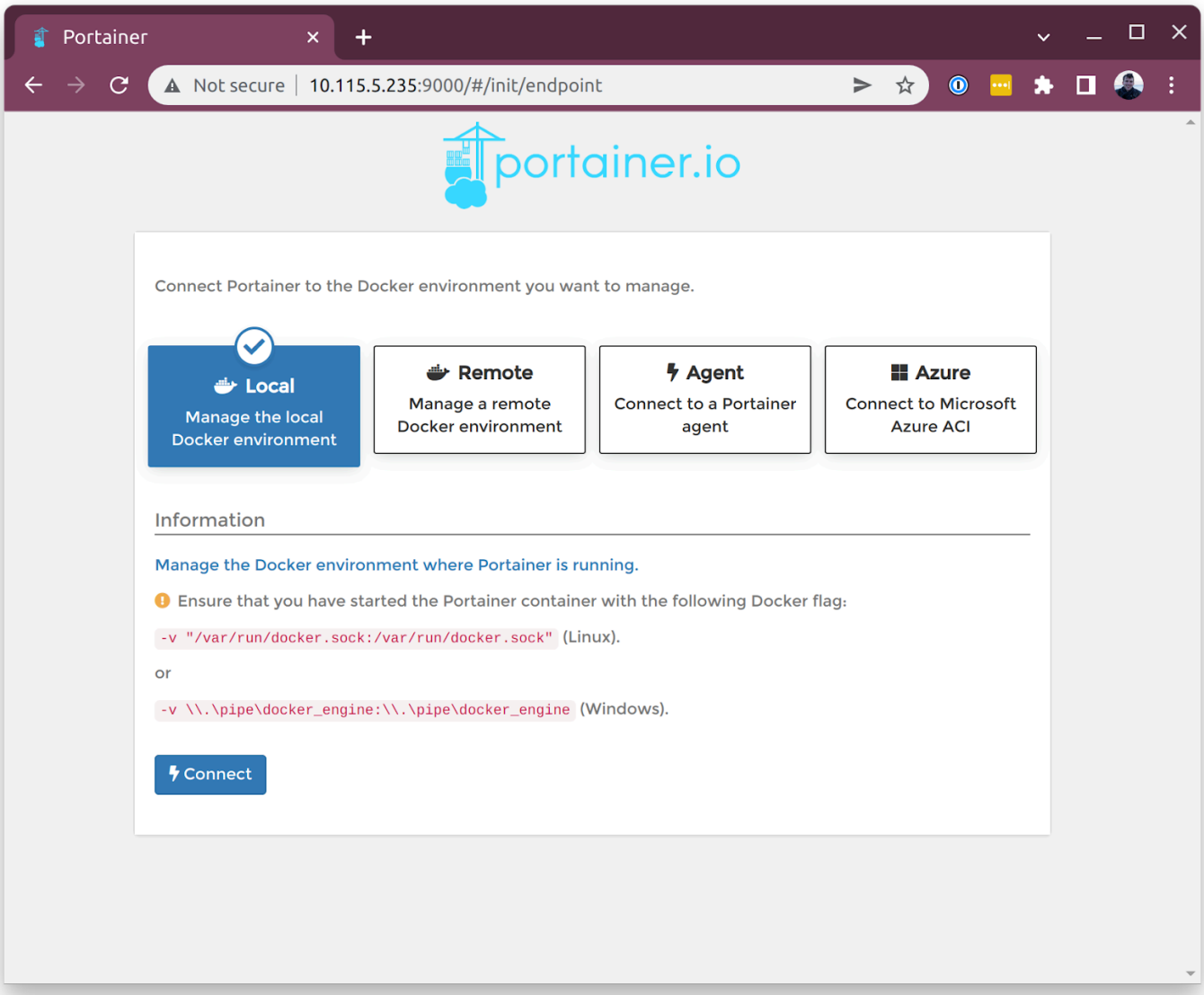
```
$ multipass launch docker --name docker-dev
Launched: docker-dev
$ multipass info docker-dev
Name:           docker-dev
State:          Running
IPv4:           10.115.5.235
                172.17.0.1
Release:        Ubuntu 22.04.1 LTS
Image hash:     3100a27357a0 (Ubuntu 22.04 LTS)
CPU(s):         2
Load:           1.50 2.21 1.36
Disk usage:     2.6GiB out of 38.6GiB
Memory usage:   259.7MiB out of 3.8GiB
Mounts:         --
```

Copy the IP address starting with "10" and paste it into your browser, then add a colon and the Portainer default port, 9000. It should look like this: 10.115.5.235:9000. This will take you to the Portainer login page where you can set a username and password.



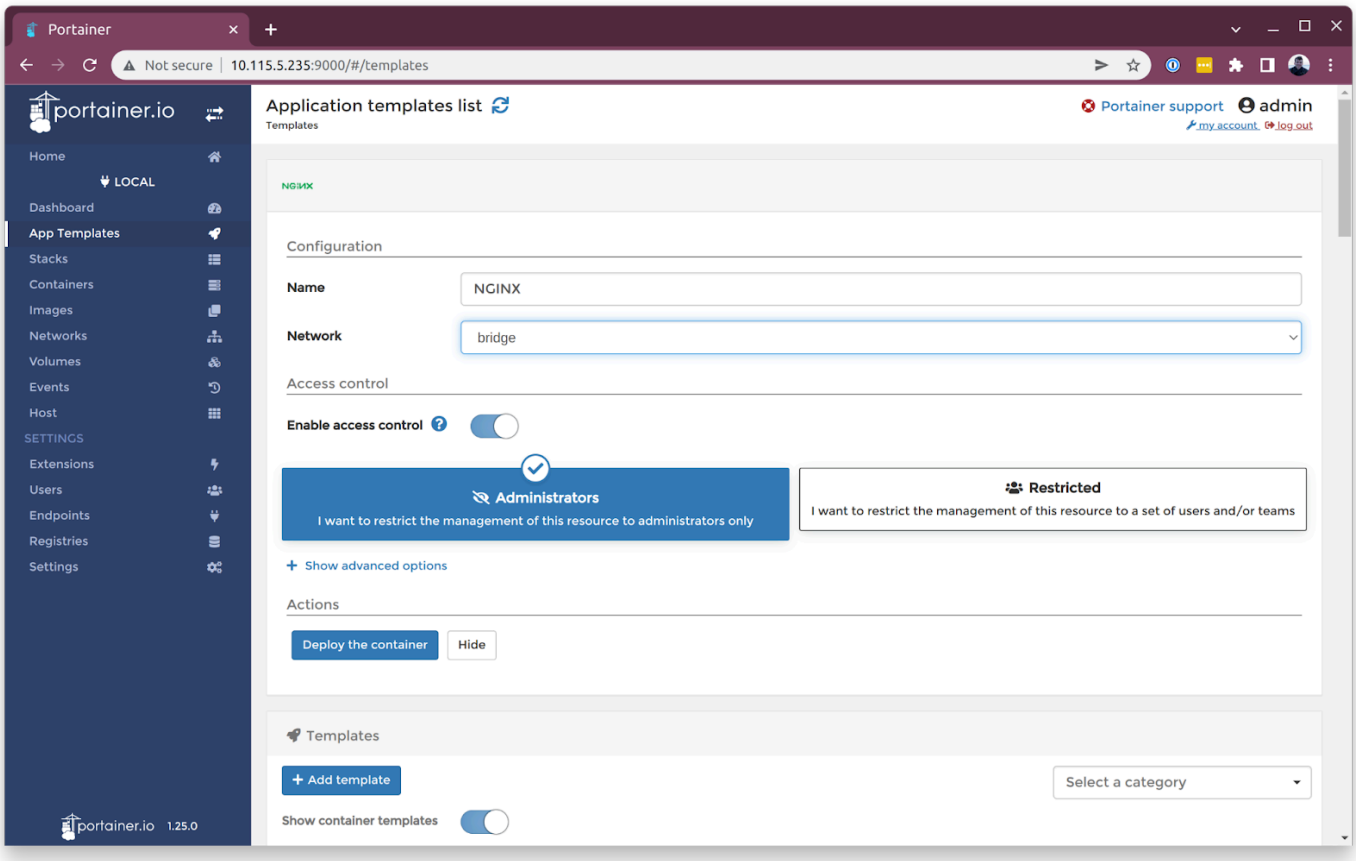
From there, select **Local** to manage a local Docker environment.

Skip to content latest

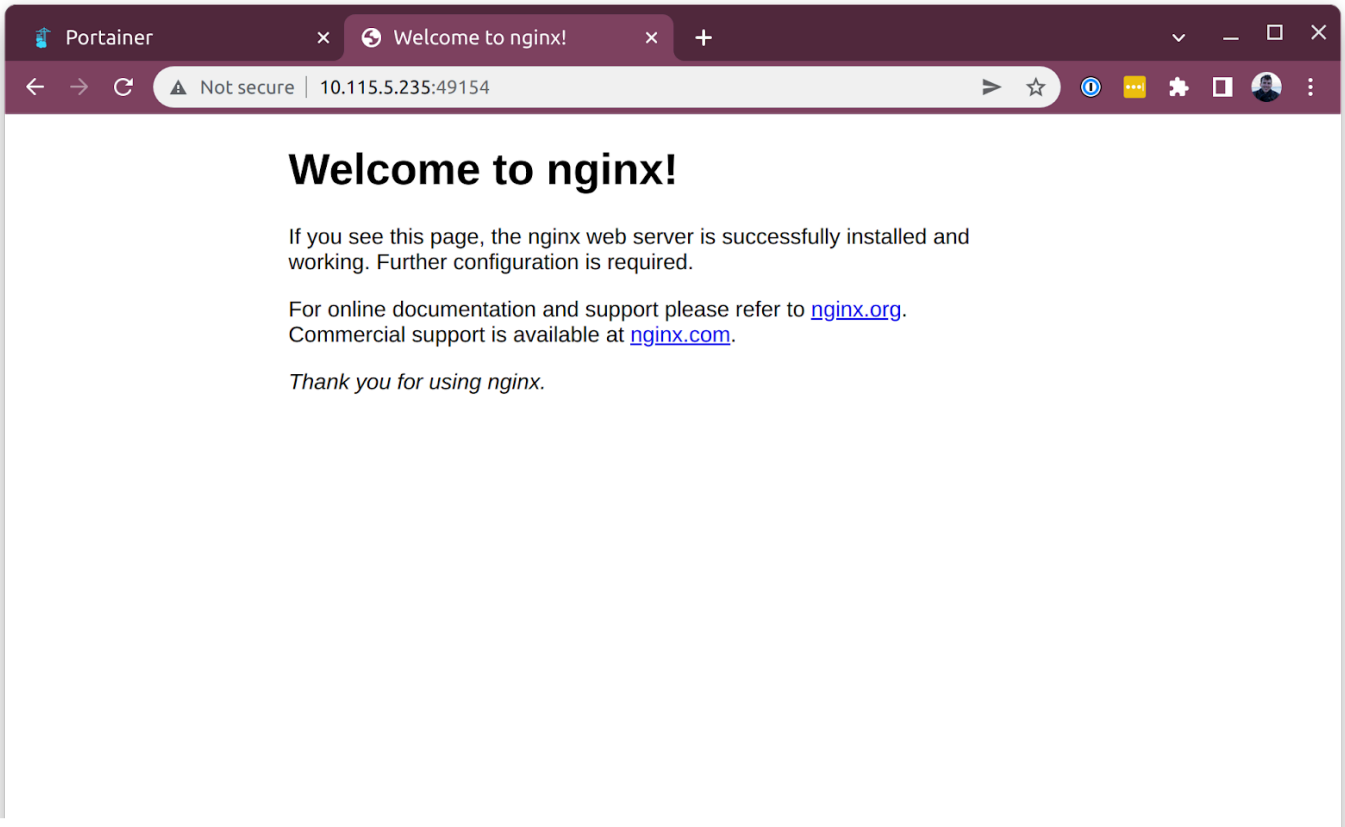


Inside the newly selected local Docker environment, locate the sidebar menu on the page and click on **App Templates**, then select **NGINX**.

Skip to content latest



From the Portainer dashboard, you can see the ports available on nginx. To verify that you have nginx running in a Docker container inside Multipass, open a new web page and paste the IP address of your instance followed by one of the port numbers.



Next steps

Congratulations! You can now use Multipass proficiently.

There's more to learn about Multipass and its capabilities. Check out our How-to guides for ideas and help with your project. In our Explanation and Reference pages you can find definitions of key concepts, a complete CLI command reference, settings options and more.

Join the discussion on the Multipass forum and let us know what you are doing with your instances!

Last updated on Apr 14, 2025

Use a blueprint (deprecated)

> **Warning**
>
> Blueprints are deprecated and will be removed in a future release. You can achieve similar effects with cloud-init and other launch options. Find out more at: Launch customized instances with Multipass and cloud-init

> See also: Blueprint

Blueprints provide a shortcut to initialising Multipass instances for a variety of applications.

To see what blueprints are available, run

```
multipass find --only-blueprints
```

> See also: `find`

To use a blueprint run:

```
multipass launch <blueprint name>
```

Launching a blueprint can take the same arguments as launching any other type of instance. If no further arguments are specified, the instance will launch with the defaults defined in the blueprint. Here's an example of creating an instance from the Docker blueprint with a few more parameters specified:

```
multipass launch docker --name docker-dev --cpus 4 --memory 8G --disk 50G -
```

This command will create an instance based on the Docker blueprint, with 4 CPU cores, 8GB of RAM, 50 GB of disk space, and connect that instance to the (predefined) bridged network.

Blueprints also provide a way of exchanging files between the host and the instance. For this, a folder named `multipass/<instance name>` is created in the user's home directory on the host and mounted in `<instance name>` in the user's home directory on the instance.

> See more: `multipass launch`

Last updated on Apr 23, 2025

Use a different terminal from the system icon

> See also: How to install Multipass, `shell`.

If you want, you can change the terminal application used by the Multipass system menu icon.

> **Note**
>
> Currently available only for macOS

To do so, you need to tell macOS which terminal to use for the `.command` file type. This document presents two ways of achieving this.

Using `duti`

`duti` is a small helper application that can modify the default application preferences. It's also available from `brew`.

Find out your preferred terminal's bundle identifier using `mdls`:

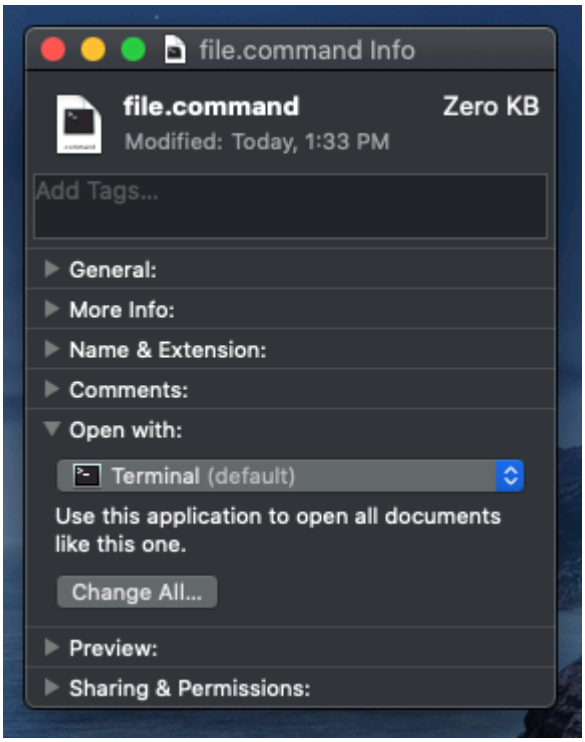
```
mdls /Applications/iTerm.app/ | grep BundleIdentifier
kMDItemCFBundleIdentifier              = "com.googlecode.iterm2"
```

And make it the default for script files using `duti`:

```
duti -s com.googlecode.iTerm2 com.apple.terminal.shell-script shell
```

Using Finder

Create an empty file with a `.command` extension and find it in Finder. Select the file and press `⌘I`. You should be presented with an information pane like this:



Expand the “Open with:” section, select your preferred terminal application and click on “Change All…”.

Last updated on Feb 19, 2025

Use an instance

> See also: Instance

This document demonstrates various ways to use an instance.

Open a shell prompt inside an instance

> See also: `shell`

To open a shell prompt on an existing instance (e.g. `loving-duck`), run the command `multipass shell loving-duck`. The output will be similar to the following:

```
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.4.0-109-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Tue May 31 14:26:40 -03 2022

  System load:  0.0               Processes:             113
  Usage of /:   28.8% of 4.67GB   Users logged in:       0
  Memory usage: 21%               IPv4 address for ens3: 10.49.93.241
  Swap usage:   0%


1 update can be applied immediately.
To see these additional updates run: apt list --upgradable


The list of available updates is more than a week old.
To check for new updates run: sudo apt update

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@loving-duck:~$
```

If the instance `loving-duck` is `Stopped` or `Suspended`, it will be started automatically.

If no argument is given to the `shell` command, Multipass will open a shell prompt on the primary instance (and also create it, if it doesn't exist).

As shown in the example above, an Ubuntu prompt is displayed as a result of the `shell` command, where you can run commands inside the instance.

To end the session, use `logout`, `exit`, or the `Ctrl-D` shortcut.

> **Note**
>
> This is also available on the GUI.

Run a command inside an instance

> See also: exec

To run a single command inside an instance, you don't need to open a shell. The command can be directly called from the host using `multipass exec`. For example, the command `multipass exec loving-duck -- pwd` returns:

```
/home/ubuntu
```

In the example, `/home/ubuntu` is the output of invoking the `pwd` command on the `loving-duck` instance.

Start an instance

> See also: `start`

An existing instance that is in `Stopped` or `Suspended` status can be started with the `multipass start` command; for example:

```
multipass start loving-duck
```

You can start multiple instances at once, specifying the instance names in the command line:

```
multipass start loving-duck devoted-lionfish sensible-shark
```

To start all existing instances at once, use the `--all` option:

```
multipass start --all
```

If no options are specified, the `multipass start` command starts the primary instance, creating it if needed.

> **Note**
>
> This is also available on the GUI.

Suspend an instance

> See also: `suspend`

An instance can be suspended with the command:

```
multipass suspend loving-duck
```

You can suspend multiple instances at once, specifying the instance names in the command line:

```
multipass suspend loving-duck devoted-lionfish sensible-shark
```

To suspend all running instances at once, use the `--all` option:

```
multipass suspend --all
```

If no options are specified, the `multipass suspend` command suspends the primary instance, if it exists and is running.

Stop an instance

> See also: `stop`

A running, not suspended instance is stopped with the command:

```
multipass stop loving-duck
```

You can stop multiple instances at once, specifying the instance names in the command line:

```
multipass stop loving-duck devoted-lionfish sensible-shark
```

To stop all running instances at once, use the `--all` option:

```
multipass stop --all
```

If no options are specified, the `multipass stop` command stops the primary instance, if it exists and is running.

Stop an instance forcefully

If the `multipass stop` command doesn't work, you can use the `--force` argument to force the instance to shut down immediately. This is particularly useful when the virtual machine is in a non-responsive, unknown or suspended state.

```
multipass stop --force
```

> **Caution**
>
> The `stop --force` command is analogous to unplugging the power cord from a physical machine – it immediately halts all computing activities. This may be necessary under certain circumstances but can potentially lead to data loss or corruption.

> **Note**
>
> This command is also available on the Multipass GUI.

Last updated on Feb 19, 2025

Use instance command aliases

> See also: Alias, How to use command aliases, Instance.

This guide demonstrates how to create, list, run and remove aliases for commands running inside an instance.

Create an alias

> See also: `alias`

To create an alias that runs a command on a given instance, use the `alias` command. The code below uses this command to create an alias `lscc` that will run the command `ls` inside an instance `crazy-cat`:

```
multipass alias crazy-cat:ls lscc
```

After running this command, the alias `lscc` is defined as running the command `ls` on the instance `crazy-cat`. If the alias name `lscc` is omitted, the alias name defaults to the name of the command to run (`ls` in this case).

Working directory mapping

By default, if the host folder on which you are running an alias is mounted on the instance, the working directory on the instance is changed to the mounted directory. To avoid this behaviour, use the option `--no-map-working-directory` when defining the alias; for example:

```
multipass alias crazy-cat:pwd pwdcc --no-map-working-directory
```

Alias contexts

> See also: `prefer`

Contexts are sets of aliases. While one can safely work with one context, named `default`, contexts can be useful in some scenarios; for example, to define aliases with the same name in different instances.

You can switch to using another context with `multipass prefer secondary`. Then, defining an alias in the new context is done in the usual way. The name of the alias can coincide with an already defined one. For example, `multipass alias cozy-canary:ls lscc`.

List the existing aliases

> See also: `aliases`

To see the list of aliases defined so far, use the `multipass aliases` command.

The output will be similar to the following:

```
Alias   Instance     Command   Context      Working directory
lscc    crazy-cat    ls        default      map
pwdcc   crazy-cat    pwd       default      default
lscc    cozy-canary  ls        secondary*   map
```

The current context is marked with an asterisk.

The column `Working directory` tells us on which directory of the host the alias will be run. The value `default` means that the alias will be run in the instance default working directory (normally, `/home/ubuntu`). The value `map` means that, in case the host directory from which the user calls the alias is mounted on the instance, the alias will be run on the mounted directory on the instance. The value will be `default` only if the `--no-map-working-directory` argument is present at alias creation.

Run an alias

There are two ways to run an alias:

- `multipass <alias>`
- `<alias>`

`multipass <alias>`

The first way of running an alias is invoking it with `multipass <alias>`, for example:

This command opens a shell into the instance `cozy-canary`, runs `ls` and returns to the host command line, as if it was an `exec` command. Since we have defined two aliases with the same name, `lscc`, the one in the current context will be run.

If you want to run an alias outside the current context, you can use a fully-qualified alias name:

```
multipass default.lscc
```

Arguments are also supported, provided you separate any options with `--`:

```
multipass lscc -- -l
```

or:

```
multipass default.lscc -- -l
```

`<alias>`

The second way of running an alias is a two-step process:

1. Add the Multipass alias script folder to the system path.
2. Run the alias.

Add the Multipass alias script folder to the system path

The instructions to add the Multipass alias script folder to the system path are displayed the first time you create an alias, and vary for each platform. For instance, when you run the command:

```
multipass alias crazy-cat:ls lscc
```

the following output is displayed:

```
You'll need to add this to your shell configuration (.bashrc, .zshrc or so)
aliases to work without prefixing with `multipass`:

PATH="$PATH:/home/user/snap/multipass/common/bin"
```

Linux macOS Windows

Skip to content

ve to edit the shell configuration file. In most Linux distrib

latest

used by default is `bash`. You can configure this option by editing the file `.bashrc` in the user's home directory using a text editor; for example:

```
nano ~/.bashrc
```

You can modify the path by appending a line to the `.bashrc` file, such as:

```
export PATH="$PATH:/home/user/snap/multipass/common/bin"
```

> **Note**
>
> Remember to replace the correct folder, as indicated in the output of the Multipass command above, and to restart the shell when done.
>
> If your shell is `zsh` and not `bash`, the file to modify is `.zshrc` instead of `.bashrc`. The procedure is the same.

Run the alias

Once you've added the alias folder to the system path, you can run it directly from the command line, without the need to also mention `multipass`; for example:

```
lscc
```

or:

```
default.lscc
```

Since the path has been added to the system path, this command is equivalent to `multipass lscc`. Arguments are also supported, without the need for `--`:

```
lscc -l
```

Remove an alias

> See also: `unalias`

Finally, to remove the alias `lscc`, run the following command:

```
multipass unalias lscc
```



```
multipass unalias default.lscc
```

The `unalias` command accepts many arguments, specifying more than one alias to remove. For example, you can remove both aliases `lscc` and `pwdcc` at once:

```
multipass unalias lscc pwdcc
```

You can also use the `--all` option to remove all the defined aliases in the current context at once:

```
multipass unalias --all
```

> **Note**
>
> Aliases are also removed when the instance for which they were defined is deleted and purged. This means that `multipass delete crazy-cat --purge` will also remove the aliases `lscc` and `pwdcc`.

Last updated on Feb 20, 2025

Use the Docker blueprint (deprecated)

> **Warning**
>
> Blueprints are deprecated and will be removed in a future release. You can achieve similar effects with cloud-init and other launch options. Find out more at: Launch customized instances with Multipass and cloud-init

The Docker blueprint gives Multipass users an easy way to create Ubuntu instances with Docker installed. It is based on the latest LTS release of Ubuntu, and includes docker engine and Portainer. The Docker blueprint automatically aliases the `docker` and `docker-compose` commands to your host, and creates a workspace that is shared between the host and the instance.

To use the Docker blueprint, run `multipass launch docker`, which will launch an instance with default parameters.

Next, follow the instructions in the output to add the aliased command to your path, it should look something like this:

```
You'll need to add this to your shell configuration (.bashrc, .zshrc or so)

aliases to work without prefixing with `multipass`:

PATH="$PATH:/home/user/snap/multipass/common/bin"
```

> **Note**
>
> Running `which docker` from your host command line should confirm that you are running Docker inside Multipass.

To access Portainer, run `multipass ls` and copy the IP address of the multipass instance (the first in the list), then enter it into your browser followed by a colon and Portainer's port number, 9000 (something like this: 10.21.145.191:9000). This gives you Portainer's web interface for visually managing your containers.

You can mount files into this instance as with any Multipass instance, but the default shared workspace is an easy way to edit your `dockerfiles` and `docker-compose.yaml` files from your host. With working directory mappi[illegible]n the `docker-compose` command from your host inside the shared directory, and have it run within that same directory in your Multipass instance.

↑ Back to top

Last updated on Apr 23, 2025

Use the primary instance

> See also: Instance, client.primary-name, `shell`, `mount`

Multipass offers a quick way to access an Ubuntu instance via a simple `multipass shell` command. This is achieved via the so-called Primary instance that is also automatically created (if it doesn't exist) when the user runs the `multipass start` or `multipass shell` commands without any arguments.

When automatically created, the primary instance gets the same properties as if `launch` was used with no arguments, except for the name (`primary` by default). In particular, this means that the instance will use the latest Ubuntu LTS image and will have the default CPU, disk and memory configuration.

You can also launch the primary instance with additional parameters, as you would do for any other instance. This provides one way to fine-tune its properties (e.g. `multipass launch --name primary --cpus 4 lts`). Alternatively, you can set another instance as primary, as explained in Changing the primary instance below.

There can be only one primary instance at any moment. If it exists, it is always listed first in the output of `list` and `info`.

Steering the primary instance

The primary instance can also be controlled from the Tray icon

In the command line, it is used as the default when no instance name is specified in the `shell`, `start`, `stop`, `restart` and `suspend` commands. When issuing one of these commands with no positional arguments, the primary instance is targeted. Its name can still be given explicitly wherever an instance name is expected (e.g. `multipass start primary`).

Automatic home mount

When launching the primary instance, whether implicitly or explicitly, Multipass automatically mounts the user's home inside it, in the folder `Home`. As with any other mount, you can unmount it with `multipass umount`. For instance, `multipass umount primary` will

 its made by Multipass inside `primary`, including the auto-

> **Note**
>
> On Windows mounts are disabled by default for security reasons. See `set` and local.privileged-mounts for information on how to enable them if needed.

Changing the primary instance

The primary instance is identified as such by its name. The name that designates an instance as the primary one is determined by a configuration setting with the key `client.primary-name`. In other words, while `primary` is the default name of the primary instance, you can change it with `multipass set client.primary-name=<custom_name>`.

This setting allows transferring primary status among instances. You can configure the primary name independently of whether instances with the old and new names exist. If they do, they lose and gain primary status accordingly.

This provides a means of (de)selecting an existing instance as primary. For example, after `multipass set client.primary-name=first`, the primary instance would be called `first`. A subsequent `multipass start` would start `first` if it existed, and launch it otherwise.

Example

Here is a long-form example of how Multipass handles the primary instance.

Set the name of the primary instance and start it:

```
multipass set client.primary-name=first
multipass start
```

Sample output:

```
Launched: first
Mounted '/home/ubuntu' into 'first:Home'
```

Now, stop the primary and launch another instance:

```
multipass stop
multipass launch eoan
```

Sample output:

Change the `client.primary-name` setting to the newly launched instance, and review the list of existing instances:

```
multipass set client.primary-name=calm-chimaera
multipass list
```

Sample output:

```
Name                    State             IPv4             Image
calm-chimaera           Running           10.122.139.63    Ubuntu 19.04 LTS
first                   Stopped           --               Ubuntu 18.04 LTS
```


Last updated on Feb 20, 2025

alias

> See also: Alias, How to use command aliases.

The `alias` command makes Multipass create a persistent alias to run commands on a given instance. Its syntax is the following:

```
multipass alias instance:command [name]
```

If `name` is omitted, the alias name defaults to `command`.

After running this command, a new alias is defined as running the `command` on the given instance.

By default, if the host folder where the alias is being executed is mounted on the instance, the instance's working directory is changed to the mounted directory. This behaviour can be avoided by defining the alias with the option `--no-map-working-directory`.

The full `multipass help alias` output explains the available options:

```
Usage: multipass alias [options] <definition> [<name>]
Create an alias to be executed on a given instance.

Options:
  -h, --help                        Displays help on commandline options
  -v, --verbose                     Increase logging verbosity. Repeat the 'v
                                    the short option for more detail. Maximum
                                    verbosity is obtained with 4 (or more) v'
                                    i.e. -vvvv.
  -n, --no-map-working-directory    Do not automatically map the host executi
                                    path to a mounted path

Arguments:
  definition                        Alias definition in the form
                                    <instance>:<command>
  name                              Name given to the alias being defined,
                                    defaults to <command>
```


Last updated on Feb 19, 2025

aliases

> See also: Alias, How to use command aliases

The `aliases` command shows the aliases defined for all the instances.

```
multipass aliases
```

The output will be similar to the following:

```
Alias   Instance          Command  Working directory
lsrm    rewarded-merlin   ls       default
topfp   flying-pig        top      map
```

The `Working directory` column tells us the directory of the host where the alias will be run. The value `default` means that the alias will be run in the instance's default working directory (normally, `/home/ubuntu`). The value `map` means that, if the host directory from which the user calls the alias is mounted on the instance, the alias will be run on the mounted directory on the instance.

> **Note**
>
> The value will be `default` only if the alias was created with the `--no-map-working-directory` option.

The command can be used in conjunction with the `--format` or `-f` options to specify the desired output format: `csv`, `json`, `table` or `yaml`.

The full `multipass help aliases` output explains the available options:

```
Usage: multipass aliases [options]
List available aliases

Options:
  -h, --help         Displays help on commandline options
  -v, --verbose      Increase logging verbosity. Repeat the 'v' in the shor
                     option for more detail. Maximum verbosity is obtained
                     4 (or more) v's, i.e. -vvvv.
  --format <format>  Output list in the requested format. Valid formats are
                     table (default), json, csv and yaml. The output workin
                     directory states whether the alias runs in the instanc
                     default directory or the alias running directory shoul
                     to be mapped to a mounted one.
```


Last updated on Feb 19, 2025

authenticate

> See also: Authentication, How to authenticate clients with the Multipass service, `local.passphrase`

The `authenticate` command is used to authenticate a client with the Multipass service. Once authenticated, the client can issue commands such as `list`, `launch`, etc.

To help reduce the amount of typing for `authenticate`, one can also use `multipass auth` as an alias:

```
multipass auth foo
```

If no passphrase is specified in the `multipass authenticate` command line, you will be prompted to enter it.

The full `multipass help authenticate` output explains the available options:

```
Usage: multipass authenticate [options] [<passphrase>]
Authenticate with the Multipass service.
A system administrator should provide you with a passphrase
to allow use of the Multipass service.

Options:
  -h, --help     Displays help on commandline options
  -v, --verbose  Increase logging verbosity. Repeat the 'v' in the short op
                 for more detail. Maximum verbosity is obtained with 4 (or
                 v's, i.e. -vvvv.

Arguments:
  passphrase     Passphrase to register with the Multipass service. If omit
                 a prompt will be displayed for entering the passphrase.
```

Last updated on Feb 19, 2025

client.apps.windows-terminal.profiles

> See also: `get`, `set`, How to integrate with Windows Terminal

Key

`client.apps.windows-terminal.profiles`

Description

Which profiles should be enabled in Windows Terminal.

Possible values

The following values are supported:

- `primary` to enable a profile for the Primary instance. Note that this value is independent of what primary name is configured.
- `none` to disable any profiles.

Examples

- `multipass set client.apps.windows-terminal.profiles=none`

Default value

`primary` (a profile for `primary` is added when Windows Terminal is found)

Last updated on Feb 19, 2025

latest

client.gui.autostart

> **Caution**
>
> This setting has been removed from the CLI starting from Multipass version 1.14. It is now only available in the GUI client.

> See also: `get`, `set`, GUI client

Key

`client.gui.autostart`

Description

Whether or not the Multipass GUI should start automatically on startup.

Possible values

Any case variations of `on|off`, `yes|no`, `1|0` or `true|false`.

Examples

```
multipass set client.gui.autostart=true
```

Default value

`true` (on Linux and macOS it only takes effect after the client (CLI or GUI) is run for the first time)

Last updated on Feb 19, 2025

latest

client.gui.hotkey

> **Caution**
>
> This setting has been removed from the CLI starting from Multipass version 1.14. It is now only available in the GUI client.

> See also: `get`, `set`, GUI client

Key

`client.gui.hotkey`

Description

A keyboard shortcut for the GUI to open a shell into the Primary instance.

Possible values

A single case-insensitive sequence of keys, containing:

- zero or more modifiers (such as `Ctrl`, `Alt`, `Cmd`, `Command`, `Opt`, etc.)
- one non-modifier key (such as `u`, `4`, `.`, `Space`, `Tab`, `Pause`, `F3`). When key names have multiple words, quote and use spaces (for example: `"Print Screen"`).
- (on macOS) alternatively, UTF-8 characters for Mac keys (such as ⌘, ⌥, ⇧, ^)
- a plus (`+`) sign separating each alphabetic word (but not key symbols) from the next
- the empty string (`""`) to disable the hotkey

> **Caution**
>
> **Caveats:**
>
> - There are some limitations on what keys and combinations are supported, depending on multiple factors such as keyboard, mapping, and OS (e.g. `AltGr`, numpad, or media keys may or may not work; `shift+enter` is rejected).
> - Some combinations may be grabbed by the OS before they reach multipass (e.g. `meta+a` may open the activities, `ctrl+alt+f3` may move ttys).

Examples

- `multipass set client.gui.hotkey="Ctrl+Print Screen"`.
- `multipass set client.gui.hotkey="^⇧Y"`.
- `multipass set client.gui.hotkey=option+space`.
- `multipass set client.gui.hotkey=""`

Default value

- `Ctrl+Alt+U` on Linux and Windows
- `⌥⌘U` on macOS

Last updated on Feb 19, 2025

client.primary-name

> See also: Instance, `get`, `set`, `launch`

Key

`client.primary-name`

Description

The name of the instance that is launched/recognised as primary.

> See also: Primary instance.

Possible values

Any valid instance name or the empty string (" ") to disable primary.

Examples

- `multipass set client.primary-name=asdf`
- `multipass set client.primary-name=`

Default value

`primary`

Last updated on Feb 19, 2025

latest

clone

The `multipass clone` command creates a clone of an instance. A cloned instance is a standalone instance that is a replica of the original instance in terms of its configuration, installed software, and data at the time of cloning. Cloning an instance can be useful for backup or testing purposes, or to create identical VMs from a working template.

> **Caution**
>
> The `clone` command is available since Multipass version 1.15.0.

Currently, only instances that are in the `Stopped` state can be cloned.

You can run the `clone` command on a source instance without any additional options. For example, `multipass clone natty-nilgai` will produce the following output:

```
...
Cloned from natty-nilgai to natty-nilgai-clone1.
```

By default, the `multipass clone` command automatically generates a name for the cloned instance using the format *<source_vm_name>-cloneN*, where *N* represents the number of instances cloned from that specific source instance, starting at 1. In the example, the name of the source VM is “natty-nilgai” and the automatically generated name for its clone is “natty-nilgai-clone1”.

Alternatively, you can specify a custom name for the cloned instance using `--name` option, following the standard instance name format. For example:

```
multipass clone natty-nilgai --name custom-clone
```

The full `multipass help clone` output explains the available options:

```
Usage: multipass clone [options] <source_name>
Create an independent copy of an existing (stopped) instance.

Options:
  -h, --help                      Displays help on commandline options
  -v, --verbose                   Increase logging verbosity. Repeat the 'v'
                                  the short option for more detail. Maximum
                                  verbosity is obtained with 4 (or more) v's
                                  i.e. -vvvv.
  -n, --name <destination-name>   An optional custom name for the cloned
                                  instance. The name must follow the usual
                                  validity rules (see "help launch"). Defaul
                                  "<source_name>-cloneN", where N is the Nth
                                  cloned instance.

Arguments:
  source_name                     The name of the source instance to be clon
```


Last updated on Feb 20, 2025

delete

> See also: `recover`, `purge`

The `multipass delete` command deletes the instances or snapshots that are specified as arguments.

You can provide multiple arguments in the same delete command, including both instances and snapshots; for example:

```
multipass delete --purge legal-takin calm-squirrel.snapshot2
```

Deleted instances are marked as such and removed from use, but you can still recover them using the `multipass recover` command, unless you used the `-p`/`--purge` option to delete them permanently.

To completely destroy instances and release the disk space they take up, use the `--purge` option or the `purge` command.

> **Caution**
>
> When you delete a snapshot, or when you delete an instance using the GUI client, Multipass removes them permanently (even if you didn't use the `--purge` option) and they cannot be recovered.

The `--all` option will delete all instances and their snapshots. Take care if using this option.

The output of `multipass help delete` explains the available options:

```
Usage: multipass delete [options] <instance>[.snapshot] [<instance>[.snapsh
Delete instances and snapshots. Instances can be purged immediately or late
with the "purge" command. Until they are purged, instances can be recovered
with the "recover" command. Snapshots cannot be recovered after deletion an

Options:
  -h, --help     Displays help on commandline options
  -v, --verbose  Increase logging verbosity. Repeat the 'v' in the short op
                 for more detail. Maximum verbosity is obtained with 4 (or
                 v's, i.e. -vvvv.
  --all          Delete all instances and snapshots
  -p, --purge    Permanently delete specified instances and snapshots
                 immediately

Arguments:
  name           Names of instances and snapshots to delete
```


Last updated on Feb 20, 2025

exec

> See also: Multipass exec and shells, How to use instance command aliases

The `exec` command runs the given commands inside the instance. The first argument is the instance to run the commands on, `--` optionally separates the `multipass` options from the rest - the command to run itself:

```
multipass exec primary -- uname -r
```

Sample output:

```
4.15.0-48-generic
```

You can pipe standard input and output to/from the command; for example:

```
multipass exec primary -- lsb_release -a | grep ^Codename:
```

Sample output:

```
No LSB modules are available.
Codename:       bionic
```

The `--` separator is required if you want to pass options to the command being run. Options to the `exec` command itself must be specified before `--`.

You can specify on which instance directory the command must be run in three different ways.

The first one is `--working-directory <dir>`, which tells Multipass that the command must be run in the folder `<dir>`. For example:

```
multipass exec arriving-pipefish --working-directory /home -- ls -a
```

The `ls -la` command shows the contents of the `/home` directory, because it was run from


```
.  ..  ubuntu
```

The second option to specify the working directory is to look through the mounted folders first. In case we are running the alias on the host from a directory which is mounted on the instance, the command will be run on the instance from there. If the working directory is not mounted on the instance, the command will be run on the default directory on the instance. This is the default behaviour and no parameter must be specified for this mapping to happen.

The third option is to directly run the command in the default directory in the instance (usually, it is `/home/ubuntu`. The parameter to force this behaviour is `--no-map-working-directory`.

The full `multipass help exec` output explains the available options:

```
Usage: multipass exec [options] <name> [--] <command>
Run a command on an instance

Options:
  -h, --help                        Displays help on commandline options
  -v, --verbose                     Increase logging verbosity. Repeat the 'v
                                    the short option for more detail. Maximum
                                    verbosity is obtained with 4 (or more) v'
                                    i.e. -vvvv.
  -d, --working-directory <dir>     Change to <dir> before execution
  -n, --no-map-working-directory    Do not map the host execution path to a
                                    mounted path

Arguments:
  name                              Name of instance to execute the command o
  command                           Command to execute on the instance
```

find

The `multipass find` command without any argument lists the images and blueprints Multipass can use to run instances with `launch` on your system and associated version information. For example:

```
Image                       Aliases           Version          Description
core                        core16            20200818         Ubuntu Core
core18                                        20211124         Ubuntu Core
core20                                        20230119         Ubuntu Core
core22                                        20230717         Ubuntu Core
20.04                       focal             20240129.1       Ubuntu 20.04
22.04                       jammy,lts         20240126         Ubuntu 22.04
23.10                       mantic            20240206         Ubuntu 23.10
daily:24.04                 noble,devel       20240129         Ubuntu 24.04
appliance:adguard-home                        20200812         Ubuntu AdGua
appliance:mosquitto                           20200812         Ubuntu Mosqu
appliance:nextcloud                           20200812         Ubuntu Nextc
appliance:openhab                             20200812         Ubuntu openH
appliance:plexmediaserver                     20200812         Ubuntu Plex

Blueprint                   Aliases           Version          Description
anbox-cloud-appliance                         latest           Anbox Cloud
charm-dev                                     latest           A developmen
docker                                        0.4              A Docker env
jellyfin                                      latest           Jellyfin is
minikube                                      latest           minikube is
ros-noetic                                    0.1              A developmen
ros2-humble                                   0.1              A developmen
```

The output is separated in two sections: one for the images and one for the blueprints. Restricting the output to only one of these two categories can be done, respectively, with the `--only-images` and `--only-blueprints` options.

Launch aliases, version information and a brief description are shown next to each name in the

Skip to content latest

Launching an image using its name or its alias gives the same result. For example, `multipass launch 23.10` and `multipass launch mantic` will launch the same image.

The aliases `lts` and `devel` are dynamic, that is, they change the image they alias from time to time, depending on the Ubuntu release.

The available aliases are:

- `default` - the default image on that remote
- `lts` - the latest Long Term Support image
- `devel` - the latest development series image (only on `daily:`)
- <codename> - the code name of a Ubuntu series
- <c> - the first letter of the code name
- <XX.YY> - the version number of a series

The list of available images and blueprints is updated periodically. The option `--force-update` forces an immediate update of the list from the servers, before showing the output.

The option `--show-unsupported` includes old Ubuntu images, which were available at some point but are not supported anymore. This means that some features of Multipass might now work on these images and no user support is given. However, they are still available for testing.

The command also supports searching through available images. For example, `multipass find mantic` returns:

```
Image                       Aliases           Version          Description
mantic                                        20240206         Ubuntu 23.10
daily:mantic                                  20240206         Ubuntu 23.10
```

The full `multipass help find` output explains the available options:

```
Usage: multipass find [options] [<remote:>][<string>]
Lists available images matching <string> for creating instances from.
With no search string, lists all aliases for supported Ubuntu releases.

Options:
  -h, --help          Displays help on commandline options
  -v, --verbose       Increase logging verbosity. Repeat the 'v' in the sho
                      option for more detail. Maximum verbosity is obtained
                      4 (or more) v's, i.e. -vvvv.
  --show-unsupported  Show unsupported cloud images as well
  --only-images       Show only images
  --only-blueprints   Show only blueprints
  --format <format>   Output list in the requested format.
                      Valid formats are: table (default), json, csv and yam
  --force-update      Force the image information to update from the networ

Arguments:
  string              An optional value to search for in [<remote:>]<string
                      format, where <remote> can be either ‘release’ or ‘da
                      If <remote> is omitted, it will search ‘release‘ firs
                      and if no matches are found, it will then search ‘dai
                      <string> can be a partial image hash or an Ubuntu rel
                      version, codename or alias.
```


Last updated on Feb 19, 2025

get

> See also: `set` , Settings

The `get` command retrieves the value of a single setting specified by a key argument. This key takes the form of a dot-separated path in a hierarchical settings tree; for example:

```
multipass get client.gui.autostart
```

You can use the `--keys` option to get a list of all available settings keys:

```
multipass get --keys
```

Sample output:

```
client.primary-name
local.bridged-network
local.driver
local.image.mirror
local.passphrase
local.privileged-mounts
```

To set the value of a particular setting, see `set` .

You can read more about the available settings in the Settings reference page.

The full `multipass help get` output explains the available options:

```
Usage: multipass get [options] [<arg>]
Get the configuration setting corresponding to the given key, or all settin
(Support for multiple keys and wildcards coming...)

Some common settings keys are:
  - client.gui.autostart
  - local.driver
  - local.privileged-mounts

Use `multipass get --keys` to obtain the full list of available settings at

Options:
  -h, --help     Displays help on commandline options
  -v, --verbose  Increase logging verbosity. Repeat the 'v' in the short op
                 for more detail. Maximum verbosity is obtained with 4 (or
                 v's, i.e. -vvvv.
  --raw          Output in raw format. For now, this affects only the
                 representation of empty values (i.e. "" instead of "<empty
  --keys         List available settings keys. This outputs the whole list
                 currently available settings keys, or just <arg>, if provi
                 and a valid key.

Arguments:
  arg            Setting key, i.e. path to the intended setting.
```

help

The `multipass help` command without an argument will list all the available commands with a brief explanation. It accepts a `<command>` argument to get detailed help for the command in question.

Sample output:

```
Usage: multipass [options] <command>
Create, control and connect to Ubuntu instances.

This is a command line utility for multipass, a
service that manages Ubuntu instances.

Options:
  -h, --help     Displays help on commandline options
  -v, --verbose  Increase logging verbosity. Repeat the 'v' in the short op
                 for more detail. Maximum verbosity is obtained with 4 (or
                 v's, i.e. -vvvv.

Available commands:
...
```

See our CLI reference for information on the available commands.

The `--verbose` option will increase its output verbosity and you can repeat it (up to three times) to get even more output. By default, only errors will be displayed, with `-v` warnings would be added, `-vv` enables informational messages and `-vvv` will print out debugging information.

Last updated on Feb 19, 2025

↑ Back to top

info

The `multipass info` command shows properties of instances and snapshots.

For example, `multipass info calm-squirrel`:

```
Name:           calm-squirrel
State:          RUNNING
Snapshots:      2
IPv4:           10.218.69.109
Release:        Ubuntu 16.04.6 LTS
Image hash:     2461b36d86ac (Ubuntu 22.04 LTS)
CPU(s):         1
Load:           0.49, 0.70, 0.71
Disk usage:     849.4M out of 4.8G
Memory usage:   41.9M out of 992.0M
Mounts:         /home/user => Home
                    UID map: 1000:default
                    GID map: 1001:default
                /home/user/multipass => mp
                    UID map: 1000:default
                    GID map: 1001:default
```

or, in case of a snapshot, `multipass info relative-lion.snapshot3`:

```
Snapshot:       snapshot3
Instance:       relative-lion
CPU(s):         1
Disk space:     5.0GiB
Memory size:    1.0GiB
Mounts:         --
Created:        2023-10-30T13:31:37.239Z
Parent:         snapshot1
Children:       --
Comment:        Before restoring snapshot2
```

To obtain information on all the snapshots of an instance, you can specify the name of the instance and add the `--snapshots` option to the command.

With no positional arguments, `multipass info` displays information on all the instances. If you add the `--snapshots` flag and no positional arguments, the command displays information on all the snapshots of all ins

Back to top

The `--format` option allows to choose how the output is formatted. The available options are a human-readable `table`, or machine-readable formats: `json`, `yaml` or `csv`.

For example, the command `multipass info --format yaml calm-squirrel` produces the following output:

```
errors:
  - ~
calm-squirrel:
  - state: RUNNING
    snapshot_count: 2
    image_hash: 2461b36d86ac524e755c56e25cdc1537c92aec8bac6d3e1795fa92b105b
    image_release: 22.04 LTS
    release: Ubuntu 22.04.3 LTS
    cpu_count: 1
    load:
      - 0.00
      - 0.01
      - 0.00
    disks:
      - sda1:
          used: 905637888
          total: 5136297984
    memory:
      usage: 31199232
      total: 1040195584
    ipv4:
      - 10.218.69.109
    mounts: ~
```

Details available for instances and snapshots

Here you can find a comprehensive list of the details that `multipass info` provides for instances and snapshots, respectively.

Instance details

↑ Back to top

ENTRY	DESCRIPTION
`Name`	Name of the instance that the information pertains to
`State`	Current state of the instance (see also: Instance states)
`Snapshots`	Number of snapshots of the instance
`IPv4`	A list of IPs through which a machine on the same network as the instance can reach the instance. Adding IP addresses besides the one automatically generated by Multipass is described in Create an instance with multiple network interfaces.
`Release`	Title of the release of the image installed on this instance
`Image hash`	The SHA-256 hash of the aforementioned image. This can be used to verify the validity of the image.
`CPU(s)`	Number of cores that the instance can use, as defined by the `--cpus` option of the `launch` command or as modified by the `local.<instance-name>.cpus` key of the `set` command. Depending on the used driver, this value can exceed the number of CPUs on the host machine.
`Load`	A triplet of values describing the average number of processes running on the CPU or in queue waiting for CPU time, sampled over the last 1 minute, 5 minutes and 15 minutes respectively. The values are normalised to the number of CPU cores, so for a machine with N cores, a load average of N indicates that the CPU is utilised at maximum capacity over the respective time interval, while a value below or above N indicates that the CPU is either under-utilised or over-utilised.
`Disk usage`	Amount of disk space used by the instance, as defined by the `--disk` option of the `launch` command or as modified by the `local.<instance-name>.disk` key of the `set` command.
`Memory usage`	Amount of memory used by the instance, as defined by the `--memory` option of the `launch` command or as modified by the `local.<instance-name>.memory` key of the `set` command. Depending on the used driver, the total value can exceed the amount of memory available on the host machine, but the used value cannot.
`Mounts`	List of directories on the host machine that are accessible through their respective directories from inside the instance. Each directory mapping has a list of UID and GID mappings, representing how the ownership of their respective host directories is translated the ownership of their respective instance directories. See also: Mount.

Snapshot details

↑ Back to top

ENTRY	DESCRIPTION
`Snapshot`	Name of the snapshot that the following details pertain to, as defined by the `--name` option of the `snapshot` command or by the `local.<instance>.<snapshot>.name` key on the `set` command.
`Instance`	Name of the instance that snapshot belongs too
`CPU(s)`	Number of cores that were assigned to the instance when the snapshot was taken, as defined by the `--cpus` option of the `launch` command or as modified by the `local.<instance-name>.cpus` key of the `set` command.
`Disk space`	Size of the instances disk when the snapshot was taken, as defined by the `--disk` option of the `launch` command or as modified by the `local.<instance-name>.disk` key of the `set` command.
`Memory size`	Amount of memory that was assigned to the instance when the snapshot was taken, as defined by the `--memory` option of the `launch` command or as modified by the `local.<instance-name>.memory` key of the `set` command.
`Mounts`	List of directories on the host machine that were mounted inside the instance at the time that the snapshot was taken. See also: Mount.
`Created`	Timestamp containing the date and time when the snapshot was created
`Parent`	The snapshot that was last in effect when this snapshot was taken, relatively to which the current snapshot records changes
`Children`	A list of snapshots that have this snapshot as a parent
`Comment`	The comment that is associated with the snapshot, as defined by the `--comment` option of the `snapshot` command or by the `local.<instance>.<snapshot>.comment` key on the `set` command.

The full `multipass help info` output explains the available options:

```
Usage: multipass info [options] <instance>[.snapshot] [<instance>[.snapshot
Display information about instances or snapshots

Options:
  -h, --help         Displays help on commandline options
  -v, --verbose      Increase logging verbosity. Repeat the 'v' in the shor
                     option for more detail. Maximum verbosity is obtained
                     4 (or more) v's, i.e. -vvvv.
  --snapshots        Display detailed information about the snapshots of
                     specified instances. This option has no effect on snap
                     arguments. Omit instance/snapshot arguments to obtain
                     detailed information on all the snapshots of all insta
  --format <format>  Output info in the requested format.
                     Valid formats are: table (default), json, csv and yaml

Arguments:
  instance/snapshot  Names of instances or snapshots to display information
                     about
```

launch

The `multipass launch` command without any argument will create and start a new instance based on the default image, using a random generated name; for example:

```
...
Launched: relishing-lionfish
```

You can then shell into an instance by its name:

```
multipass shell relishing-lionfish
```

The only, optional, positional argument is the image to launch an instance from. See `find` for how to find information on the available images.

It's also possible to provide a full URL to the image (use `file://` for an image available on the host running `multipassd`).

You can change the resources made available to the instance by passing the `--cpus`, `--disk` or `--memory` options, changing the allocated CPU cores, disk space or RAM, respectively.

If you want your instance to have a name of your choice, use the `--name` option. The instance name has to be unique and conform to a specific instance name format.

By passing a filename or an URL to `--cloud-init`, you can provide user data to `cloud-init` to customise the instance on first boot. See cloud-init documentation for examples.

Use the `--network` option to Create an instance with multiple network interfaces.

Passing `--bridged` and `--network bridged` are shortcuts to `--network <name>`, where `<name>` is configured via `multipass set local.bridged-interface`.

You can also mount folders in the instance after it is launched using the `--mount` option. It can be specified multiple times, with different mount paths.

Use the `--timeout` option to change how long Multipass waits for the machine to boot and initialise.

The full `multipass help launch` output explains the available options:

```
Usage: multipass launch [options] [[<remote:>]<image> | <url>]
Create and start a new instance.
```

↑ Back to top

```
Options:
  -h, --help                          Displays help on commandline option
  -v, --verbose                       Increase logging verbosity. Repeat
                                      'v' in the short option for more de
                                      Maximum verbosity is obtained with
                                      more) v's, i.e. -vvvv.
  -c, --cpus <cpus>                   Number of CPUs to allocate.
                                      Minimum: 1, default: 1.
  -d, --disk <disk>                   Disk space to allocate. Positive
                                      integers, in bytes, or decimals, wi
                                      M, G suffix.
                                      Minimum: 512M, default: 5G.
  -m, --memory <memory>               Amount of memory to allocate. Posit
                                      integers, in bytes, or decimals, wi
                                      M, G suffix.
                                      Minimum: 128M, default: 1G.
  -n, --name <name>                   Name for the instance. If it is
                                      'primary' (the configured primary
                                      instance name), the user's home
                                      directory is mounted inside the new
                                      launched instance, in 'Home'.
                                      Valid names must consist of letters
                                      numbers, or hyphens, must start wit
                                      letter, and must end with an
                                      alphanumeric character.
  --cloud-init <file> | <url>         Path or URL to a user-data cloud-in
                                      configuration, or '-' for stdin
  --network <spec>                    Add a network interface to the
                                      instance, where <spec> is in the
                                      "key=value,key=value" format, with
                                      following keys available:
                                       name: the network to connect to
                                      (required), use the networks comman
                                      a list of possible values, or use
                                      'bridged' to use the interface
                                      configured via `multipass set
                                      local.bridged-network`.
                                       mode: auto|manual (default: auto)
                                       mac: hardware address (default:
                                      random).
```

```
                                        You can also use a shortcut of "<na
                                        to mean "name=<name>".
  --bridged                             Adds one `--network bridged` networ
  --mount <local-path>:<instance-path>  Mount a local directory inside the
                                        instance. If <target> is omitted,
                                        the mount point will be under
                                        /home/ubuntu/<source-dir>, where
                                        <source-dir> is the name of the
                                        <source> directory.
  --timeout <timeout>                   Maximum time, in seconds, to wait f
                                        the command to complete. Note that
                                        background operations may continue
                                        beyond that. By default, instance
                                        startup and initialisation is limit
                                        5 minutes each.

Arguments:
  image                                 Optional image to launch. If omitte
                                        then the default Ubuntu LTS will be
                                        used.
                                        <remote> can be either ‘release’ or
                                        ‘daily‘. If <remote> is omitted,
                                        ‘release’ will be used.
                                        <image> can be a partial image hash
                                        an Ubuntu release version, codename
                                        alias.
                                        <url> is a custom image URL that is
                                        https://, https://, or file:// form
```


Last updated on Feb 20, 2025

list

> See also: info, launch, snapshot

The multipass list command lists available instances or snapshots. With no options, it presents a generic view of instances, with some of their properties; for example:

```
Name                    State             IPv4             Release
primary                 SUSPENDED         --               Ubuntu 18.04 LTS
calm-squirrel           RUNNING           10.218.69.109    Ubuntu 16.04 LTS
```

You can also call it with the --snapshots flag to get an overview of available snapshots. Here's a sample output of multipass list --snapshots:

```
Instance        Snapshot    Parent      Comment
calm-squirrel   snapshot1   --          --
calm-squirrel   snapshot3   snapshot1   Before restoring snapshot2
```

The multipass list command will truncate long snapshot comments, as well as those containing newlines. You can use info to view them in full.

You can also use the --format option to get machine-readable output (CSV, JSON, or YAML). For example, multipass list --format yaml:

```
primary:
  - state: SUSPENDED
    ipv4:
      - ""
    release: 18.04 LTS
calm-squirrel:
  - state: RUNNING
    ipv4:
      - 10.218.69.109
    release: 16.04 LTS
```

The full `multipass help list` output explains the available options:

```
Usage: multipass list [options]
List all instances or snapshots which have been created.

Options:
  -h, --help         Displays help on commandline options
  -v, --verbose      Increase logging verbosity. Repeat the 'v' in the shor
                     option for more detail. Maximum verbosity is obtained
                     4 (or more) v's, i.e. -vvvv.
  --snapshots        List all available snapshots
  --format <format>  Output list in the requested format.
                     Valid formats are: table (default), json, csv and yaml
```


Last updated on Feb 19, 2025

local.<instance-name>.<snapshot-name>.comment

> See also: `get`, `set`, `snapshot`

Key

`local.<instance-name>.<snapshot-name>.comment`

Where `<instance-name>` is the name of a Multipass instance and `<snapshot-name>` is the current name of one of its snapshots.

Description

An optional free piece of text that is associated with the snapshot. This can be used to describe the snapshot, recognise its role, or any other purpose.

Possible values

Any string that your terminal can encode.

Examples

```
multipass set local.relative-lion.snaphot2.comment="Got it! 🧐"
```

Default value

The comment that was assigned to the snapshot when it was taken.

Last updated on Feb 19, 2025

local.<instance-name>.<snapshot-name>.name

> See also: `get`, `set`, `snapshot`, Instance

Key

`local.<instance-name>.<snapshot-name>.name`

where `<instance-name>` is the name of a Multipass instance and `<snapshot-name>` is the current name of one of its snapshots.

Description

The name that identifies the snapshot in the context of its instance. Snapshot names cannot be repeated.

Possible values

Any name that follows the same rules as instance names and is not currently in use by another instance.

Examples

```
multipass set local.relative-lion.snaphot2.name=primed
```

Default value

The name that was assigned to the snapshot when it was taken.

Last updated on Feb 19, 2025

local.<instance-name>.bridged

> See also: `get`, `set`, `local.bridged-network`, How to add networks to existing instances

Key

`local.<instance-name>.bridged`

where `<instance-name>` is the name of a Multipass instance.

Description

Whether or not the virtual machine is connected to the preferred bridge that is currently defined by `local.<instance-name>.bridged`.

Setting this to `true` will cause the instance to be bridged to that host network interface.

Removing a bridged network from an instance is currently not supported.

Possible values

This setting can have a boolean value (`true` or `false`). However, at this time, it can only be manually set to `true`, but not to `false`.

The value of this setting depends on the value of `local.bridged-network`; that is, it varies dynamically according to the configured preferred network and the networks that have been added to the instance so far.

Examples

```
multipass set local.ultimate-grosbeak.bridged=true
```

If the instance `ultimate-grosbeak` was launched with the command `multipass launch --network eth0`, the result of `multipass get local.ultimate-grosbeak.bridged` will be `true` for as long as the value of `local.bridged-network` is `eth0`.

If you run the command `multipass set local.bridged-network=eth1`, the result of `multipass get local.ultimate-grosbeak.bridged` will become `false`. At that point, you could run the command `multipass set local.ultimate-grosbeak.bridged=true` to bridge `ultimate-grosbeak` with `eth1`.

Bridged connection to a physical network interface

On some platforms/backends, Multipass cannot connect instances directly to physical network interface controllers (NICs). In this case, Multipass offers to create a bridge/switch on the host connecting to that NIC. The instance is then connected to the bridge/switch, achieving the same effect as if it had been connected to the physical NIC directly. An instance that is indirectly connected with a physical NIC in this fashion is also considered to be bridged with that NIC by Multipass.

For example, if the preferred network is `eth1` and the instance `ultimate-grosbeak` is connected with a bridge `br-eth1` that is linked with `eth1`, then `multipass get local.ultimate-grosbeak.bridged` will return `true`.

Default value

`true` if the instance is bridged with the preferred network, `false` otherwise.

Last updated on Feb 19, 2025

local.<instance-name>.cpus

> See also: `get`, `set`, `launch`.

Key

`local.<instance-name>.cpus`

where `<instance-name>` is the name of a Multipass instance.

Description

The number of CPUs to simulate on the virtual machine. This establishes a limit on how many host threads the instance can simultaneously use, at most.

Possible values

A positive integer number. Multipass does not impose an upper limit on the possible values, but the underlying backend may do so.

Examples

```
multipass set local.handsome-ling.cpus=4
```

Default value

The number of CPUs that the instance was launched with.

Last updated on Feb 19, 2025

local.<instance-name>.disk

> See also: `get`, `set`, `launch`

Key

`local.<instance-name>.disk`

where `<instance-name>` is the name of a Multipass instance.

Description

The size of the instance's disk.

> **Caution**
>
> The disk size can only be increased. Decreasing the disk size is not permitted since it could cause data loss and the VM might break.

Allowed values

A size no smaller than the current disk size. This size can be specified with a positive integer or decimal number, optionally followed by a unit. Any case variations of the following suffixes are accepted as units:

- B, to designate one byte.
- KiB, KB, or K, to designate 1024 bytes.
- MiB, MB, or M, to designate 1024 x 1024 = 1048576 bytes
- GiB, GB, or G, to designate 1024 x 1024 x 1024 = 1073741824 bytes

> **Note**
>
> Decimal bytes (e.g. 1.1B) are refused, unless specified with larger units (>= KiB), in which case the amount is floored (e.g. 1.2KiB is interpreted as 1228B, even though 1.2 x 1024 = 1228.8).


```
multipass set local.handsome-ling.disk=7.5G
```

Default value

The size of the disk that the instance was launched with.

Last updated on Feb 19, 2025

local.<instance-name>.memory

> See also: `get`, `set`, `launch`

Key

`local.<instance-name>.memory`

where `<instance-name>` is the name of a Multipass instance.

Description

The amount of RAM that is allocated to the instance.

> **Note**
>
> Hypervisors may impose additional rounding on the total memory that is given to the instance. Furthermore, the value that is set here does not correspond exactly to the memory size that is available in userspace inside the instance (e.g. reported by `free -b`), because the guest kernel claims some for its own. Total memory can be inspected inside the instance with `lshw` (e.g. `sudo lshw -json -class memory`).

> **Note**
>
> Memory on the host is only allocated as the instance uses it, not right away. Once used, it is not released until the instance is shutdown or restarted.

Allowed values

A positive memory size, specified with a positive integer or decimal number, optionally followed by a unit. Any case variations of the following suffixes are accepted as units:

- B, to designate one byte
- KiB, KB, or K, to designate 1024 bytes
- MiB, MB, or M, to designate 1024 x 1024 = 1048576 bytes
- GiB, GB, or G, to designate 1024 x 1024 x 1024 = 1073741824 bytes

> **Note**
>
> Decimal bytes (e.g. 1.1B) are refused, unless specified with larger units (>= KiB), in which case the amount is floored (e.g. 1.2KiB is interpreted as 1228B, even though 1.2 x 1024 = 1228.8).

Examples

```
multipass set local.handsome-ling.memory=4G
```

Default value

The amount of memory that the instance was launched with.

Last updated on Feb 19, 2025

local.bridged-network

> See also: `get`, `set`, `launch`, `networks`, Create an instance with multiple network interfaces, Add a network to an existing instance, `local.<instance-name>.bridged`

Key

`local.bridged-network`

Description

The name of the interface to connect the instance to when the shortcut `launch --bridged` is issued.

Possible values

Any name from the output of `multipass networks`.

Validation is deferred to the `launch` command.

Examples

```
multipass set local.bridged-network=en0
```

Default value

`<empty> ("")`.

Last updated on Feb 20, 2025

latest

local.driver

> See also: `get`, `set`, Driver, How to set up the driver

Key

`local.driver`

Description

A string identifying the hypervisor back-end in use.

Possible values

- `qemu`, `libvirt` (deprecated) and `lxd` (deprecated) on Linux
- `hyperv` and `virtualbox` on Windows
- `qemu` and `virtualbox` on macOS 10.15+
- *(deprecated)* `hyperkit` on Intel macOS 10.15+

Default values

- `qemu` on macOS and Linux
- `hyperv` on Windows

Last updated on May 31, 2025

local.passphrase

> See also: `get`, `set`, `authenticate`

Key

`local.passphrase`

Description

Passphrase used by clients requesting access to the Multipass service via the `authenticate` command.

The passphrase can be given directly in the command line; if omitted, the user will be prompted to enter it.

Possible values

Any string

Examples

```
multipass set local.passphrase
```

Default value

Not set (expressed as `false` by `multipass get`)

Last updated on Feb 20, 2025

local.privileged-mounts

> See also: `get`, `set`, `mount`, Mount

Key

`local.privileged-mounts`

Description

Controls whether the `mount` command is allowed.

Possible values

Any case variations of `on|off`, `yes|no`, `1|0` or `true|false`.

Examples

```
multipass set local.privileged-mounts=Yes
```

Default value

- `true` on Linux and macOS
- `false` on Windows

Last updated on Feb 20, 2025

latest

mount

> See also: How to share data with an instance, Mount, `umount`

The `multipass mount` command maps a local directory from the host to an instance, with the possibility to specify the mount type (classic or native) and define group or user ID mappings.

For example, here's the syntax for mapping a local directory to your virtual machine using the classic mount:

```
multipass mount --type=classic /host/path <instance name>:/instance/path
```

Use the `multipass umount` command to undo the mapping.

See Mount to learn more on the difference between "classic" and "native" mounts.

See How to share data with an instance for examples of how you can use the `multipass mount` command to share data between your host and an instance.

The full `multipass help mount` output explains the available options:

```
Usage: multipass mount [options] <source> <target> [<target> ...]
Mount a local directory inside the instance. If the instance is
not currently running, the directory will be mounted
automatically on next boot.

Options:
  -h, --help                          Displays help on commandline options
  -v, --verbose                       Increase logging verbosity. Repeat the '
                                      in the short option for more detail. Max
                                      verbosity is obtained with 4 (or more) v
                                      i.e. -vvvv.
  -g, --gid-map <host>:<instance>     A mapping of group IDs for use in the mo
                                      File and folder ownership will be mapped
                                      <host> to <instance> inside the instance
                                      be used multiple times. Mappings can onl
                                      specified as a one-to-one relationship.
  -u, --uid-map <host>:<instance>     A mapping of user IDs for use in the mou
                                      File and folder ownership will be mapped
                                      <host> to <instance> inside the instance
                                      be used multiple times. Mappings can onl
                                      specified as a one-to-one relationship.
  -t, --type <type>                   Specify the type of mount to use.
                                      Classic mounts use technology built into
                                      Multipass.
                                      Native mounts use hypervisor and/or plat
                                      specific mounts.
                                      Valid types are: 'classic' (default) and
                                      'native'

Arguments:
  source                              Path of the local directory to mount
  target                              Target mount points, in <name>[:<path>]
                                      format, where <name> is an instance name
                                      and optional <path> is the mount point.
                                      If omitted, the mount point will be unde
                                      /home/ubuntu/<source-dir>, where <source
                                      is the name of the <source> directory.
```

Skip to content latest

Last updated on Feb 19, 2025

latest

networks

> See also: Driver (backend), How to set up the driver

The `multipass networks` command lists network interfaces that multipass can connect instances to. The result depends both on the platform and the driver in use.

At this time, `multipass networks` can only find interfaces in the following scenarios:

- on Linux, with QEMU (*from Multipass 1.15 onward*) and LXD (deprecated)
- on Windows, with both Hyper-V and VirtualBox
- on macOS, with the QEMU and VirtualBox drivers

For example, on Windows with Hyper-V the `multipass networks` command returns:

```
Name            Type    Description
Default Switch  switch  Virtual Switch with internal networking
ExternalSwitch  switch  Virtual Switch with external networking via "Red Ha
InternalSwitch  switch  Virtual Switch with internal networking
PrivSwitch      switch  Private virtual switch
```

Like `list`, `networks` supports the `--format` option.

Another example, running the command `multipass networks --format yaml` on macOS with VirtualBox returns:

```
bridge0:
  - type: bridge
    description: Network bridge with en1, en2
bridge2:
  - type: bridge
    description: Empty network bridge
en0:
  - type: wifi
    description: Wi-Fi (Wireless)
en1:
  - type: thunderbolt
    description: Thunderbolt 1
en2:
  - type: thunderbolt
    description: Thunderbolt 2
```

See `launch` and Create an instance with multiple network interfaces for instructions on how to use these.

The `multipass help networks` command explains the available options:

```
Usage: multipass networks [options]
List host network devices (physical interfaces, virtual switches, bridges)
available to integrate with using the `--network` switch to the `launch`
command.

Options:
  -h, --help         Displays help on commandline options
  -v, --verbose      Increase logging verbosity. Repeat the 'v' in the shor
                     option for more detail. Maximum verbosity is obtained
                     4 (or more) v's, i.e. -vvvv.
  --format <format>  Output list in the requested format.
                     Valid formats are: table (default), json, csv and yaml
```


Last updated on Apr 07, 2025

latest

prefer

> See also: Alias, How to use command aliases

The `prefer` command, introduced in Multipass release 1.11.0, is used to switch alias contexts. It accepts a single argument, specifying the name of the context to switch to.

For example, the command `multipass prefer secondary` makes `secondary` the current context. If the given context does not exist, Multipass creates it.

Subsequent calls to `alias` , `unalias` and running any aliases from the command line will use the aliases defined in the given context.

The full `multipass help prefer` output explains the available options:

```
Usage: multipass prefer [options] <name>
Switch the current alias context. If it does not exist, create it before sw

Options:
  -h, --help     Displays help on commandline options
  -v, --verbose  Increase logging verbosity. Repeat the 'v' in the short op
                 for more detail. Maximum verbosity is obtained with 4 (or
                 v's, i.e. -vvvv.

Arguments:
  name           Name of the context to switch to
```

Last updated on Feb 19, 2025

purge

> See also: delete, recover

The `multipass purge` command will permanently remove all instances deleted with the `multipass delete` command. This will destroy all the traces of the instance, and cannot be undone.

The full `multipass help purge` output explains the available options:

```
Usage: multipass purge [options]
Purge all deleted instances permanently, including all their data.

Options:
  -h, --help     Displays help on commandline options
  -v, --verbose  Increase logging verbosity. Repeat the 'v' in the short op
                 for more detail. Maximum verbosity is obtained with 4 (or
                 v's, i.e. -vvvv.
```

Last updated on Feb 20, 2025

recover

> See also: `delete`, `purge`

The `multipass recover` command will revive an instance that was previously removed with `multipass delete`. For this to be possible, the instance cannot have been purged with `multipass purge` nor with `multipass delete --purge`.

Use the `--all` option to recover all deleted instances at once:

```
multipass recover --all
```

The full `multipass help restart` output explains the available options:

```
Usage: multipass recover [options] <name> [<name> ...]
Recover deleted instances so they can be used again.

Options:
  -h, --help     Display this help on commandline options
  -v, --verbose  Increase logging verbosity. Repeat the 'v' in the short op
                 for more detail. Maximum verbosity is obtained with 4 (or
                 v's, i.e. -vvvv.
  --all          Recover all deleted instances

Arguments:
  name           Names of instances to recover
```

Last updated on Feb 20, 2025

restart

The `multipass restart` command without any argument will restart the Primary instance (and fail, if it doesn't exist). You can also pass one or more instance names or the `--all` option to restart more instances at the same time.

> **Note**
>
> Only instances in `Running` status can be restarted.

For example:

```
multipass restart desirable-earwig
```

The full `multipass help restart` output explains the available options:

```
Usage: multipass restart [options] [<name> ...]
Restart the named instances. Exits with return
code 0 when the instances restart, or with an
error code if any fail to restart.

Options:
  -h, --help            Displays help on commandline options
  -v, --verbose         Increase logging verbosity. Repeat the 'v' in the sh
                        option for more detail. Maximum verbosity is obtaine
                        with 4 (or more) v's, i.e. -vvvv.
  --all                 Restart all instances
  --timeout <timeout>   Maximum time, in seconds, to wait for the command to
                        complete. Note that some background operations may
                        continue beyond that. By default, instance startup a
                        initialisation is limited to 5 minutes each.

Arguments:
  name                  Names of instances to restart. If omitted, and witho
                        the --all option, 'primary' will be
```

Skip to content latest

Last updated on Feb 19, 2025

restore

> See also: `snapshot`, `list`, `info`, `delete`

The `multipass restore` command restores an instance to the state that it was in when the given snapshot was taken.

For example, when you run the command:

```
multipass restore relative-lion.snapshot2
```

the system will ask you if you want to save a snapshot of your instance before proceeding:

```
Do you want to take a snapshot of relative-lion before discarding its curre
```

If you confirm, the output will be similar to the following:

```
Snapshot taken: relative-lion.snapshot3
Snapshot restored: relative-lion.snapshot2
```

As shown in the example, with no further options, the command will offer to take another snapshot. This automatic snapshot saves the instance's current state before it is thrown away. It will be named following the `snapshot` command's default naming convention, and it will have an automatic comment to indicate its purpose.

In our example, if you run:

```
multipass info relative-lion.snapshot3 | grep Comment
```

you'll find the comment:

```
Comment:        Before restoring snapshot1
```


`--destructive` (or `-d`) option to skip the question and di[illegible] and is run non-interactively (i.e. with either standard input

being redirected), this flag is required, since there is no way to query the user for confirmation.

The full `multipass help restore` output explains the available options:

```
Usage: multipass restore [options] <instance>.<snapshot>
Restore an instance to the state of a previously taken snapshot.

Options:
  -h, --help         Displays help on commandline options
  -v, --verbose      Increase logging verbosity. Repeat the 'v' in the shor
                     option for more detail. Maximum verbosity is obtained
                     4 (or more) v's, i.e. -vvvv.
  -d, --destructive  Discard the current state of the instance

Arguments:
  instance.snapshot  The instance to restore and snapshot to use, in
                     <instance>.<snapshot> format, where <instance> is the
                     of an instance, and <snapshot> is the name of a snapsh
```


Last updated on Feb 20, 2025

set

> See also: `get` , Settings

The `multipass set` command takes an argument in the form `<key>=<value>` to configure a single setting. The *key* part is a dot-separated path identifying the setting in a hierarchical settings tree. The *value* part is what it should be set to:

```
multipass set client.gui.autostart=false
```

To find what the configured value is at any moment, see `get` .

You can read more about the available settings in the Settings reference page.

The full `multipass help set` output explains the available options:

```
Usage: multipass set [options] <key>[=<value>]
Set, to the given value, the configuration setting corresponding to the giv

Some common settings keys are:
  - client.gui.autostart
  - local.driver
  - local.privileged-mounts

Use `multipass get --keys` to obtain the full list of available settings at

Options:
  -h, --help     Displays help on commandline options
  -v, --verbose  Increase logging verbosity. Repeat the 'v' in the short op
                 for more detail. Maximum verbosity is obtained with 4 (or
                 v's, i.e. -vvvv.

Arguments:
  keyval         A key, or a key-value pair. The key specifies a path to th
                 setting to configure. The value is its intended value. If
                 the key is given, the value will be prompted for.
```


Last updated on Feb 19, 2025

shell

The `multipass shell` command will open a shell prompt on an instance. Without any arguments, it will open the shell prompt of the Primary instance (and also create it, if it doesn't exist). You can also pass the name of an existing instance. If the instance is not running, it will be started automatically.

If you run `multipass shell` you'll find yourself in the *primary* instance:

```
Launched: primary
Mounted '/Users/giulia' into 'primary:Home'
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-44-generic aarch64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Wed Sep 18 15:11:15 UTC 2024

  System load:             0.47
  Usage of /:              40.2% of 3.80GB
  Memory usage:            22%
  Swap usage:              0%
  Processes:               109
  Users logged in:         0
  IPv4 address for enp0s1: 192.168.64.21
  IPv6 address for enp0s1: fdec:5598:6ba4:a940:5054:ff:fe88:498b


Expanded Security Maintenance for Applications is not enabled.

23 updates can be applied immediately.
23 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status
```

Skip to content latest

To open a shell in an existing instance, run the command:

```
multipass shell <instance name>
```

The full `multipass help shell` output explains the available options:

```
Usage: multipass shell [options] [<name>]
Open a shell prompt on the instance.

Options:
  -h, --help            Displays help on commandline options
  -v, --verbose         Increase logging verbosity. Repeat the 'v' in the sh
                        option for more detail. Maximum verbosity is obtaine
                        with 4 (or more) v's, i.e. -vvvv.
  --timeout <timeout>   Maximum time, in seconds, to wait for the command to
                        complete. Note that some background operations may
                        continue beyond that. By default, instance startup a
                        initialisation is limited to 5 minutes each.

Arguments:
  name                  Name of the instance to open a shell on. If omitted,
                        'primary' (the configured primary instance name) wil
                        assumed. If the instance is not running, an attempt
                        made to start it (see `start` for more info).
```


Last updated on Feb 19, 2025

snapshot

> See also: `restore`, `list`, `info`, `delete`

The `multipass snapshot` command takes a snapshot of an instance; for example:

```
multipass snapshot maximal-stag
```

The output will be similar to the following:

```
...
Snapshot taken: maximal-stag.snapshot1
```

The snapshot will record all the information that is required to later restore the instance to the same state. For the time being, the `snapshot` command can only operate on instances in `Stopped` status.

You have the option to specify a snapshot name using the `--name` option, following the same format as the instance name format.

If you don't specify a name, the snapshot will be named "snapshotN", where N is the total number of snapshots that were ever taken of that instance. You can also add an optional comment with `--comment`. Both of these properties can be later modified with the `set` command, via the snapshot settings keys documented in Settings.

The full `multipass help snapshot` output explains the available options:

```
Usage: multipass snapshot [options] instance
Take a snapshot of an instance that can later be restored to recover the cu

Options:
  -h, --help                    Displays help on commandline options
  -v, --verbose                 Increase logging verbosity. Repeat the 'v' i
                                the short option for more detail. Maximum
                                verbosity is obtained with 4 (or more) v's,
                                -vvvv.
  -n, --name <name>             An optional name for the snapshot, subject t
                                the same validity rules as instance names (s
                                `help launch`). Default: "snapshotN", where
                                one plus the number of snapshots that were e
                                taken for <instance>.
  --comment, -c, -m <comment>   An optional free comment to associate with t
                                snapshot. (Hint: quote the text to avoid spa
                                being parsed by your shell)

Arguments:
  instance                      The instance to take a snapshot of.
```


Last updated on Feb 19, 2025

start

The `multipass start` command without any argument will start the Primary instance (and also create it, if it doesn't exist). You can also pass one or more instance names or the `--all` option to start more instances at the same time.

For example, the command `multipass start` will produce the following output:

```
Configuring primary \
Launching primary |
...
```

Only instances in `Stopped` or `Suspended` status can be started. Running instances can be restarted with `multipass restart`, stopped with `multipass stop`, and suspended with `multipass suspend`.

The full `multipass help start` output explains the available options:

```
Usage: multipass start [options] [<name> ...]
Start the named instances. Exits with return code 0
when the instances start, or with an error code if
any fail to start.

Options:
  -h, --help            Displays help on commandline options
  -v, --verbose         Increase logging verbosity. Repeat the 'v' in the sh
                        option for more detail. Maximum verbosity is obtaine
                        with 4 (or more) v's, i.e. -vvvv.
  --all                 Start all instances
  --timeout <timeout>   Maximum time, in seconds, to wait for the command to
                        complete. Note that some background operations may
                        continue beyond that. By default, instance startup a
                        initialisation is limited to 5 minutes each.

Arguments:
  name                  Names of instances to start. If omitted, and without
                        --all option, 'primary' (the configured primary inst
                        name) will be assumed. If 'primary' does not exist b
                        included in a successful start command either implic
                        or explicitly), it is launched automatically (see
                        `launch` for more info).
```


Last updated on Feb 19, 2025

stop

The `multipass stop` command without any argument will stop the Primary instance (and fail, if it doesn't exist). You can also pass one or more instance names or the `--all` option to stop more instances at the same time.

> **Note**
>
> Only instances in `Running` status can be stopped.

`Stopped` instances can be started again with `multipass start`.

You can use the `--force` option (*available starting from Multipass version 1.14*) to stop the instance when it is in a non-responsive, unknown or suspended state. It can also be used with the `--all` option to stop all instances forcefully.

The full `multipass help stop` output explains the available options:

```
Usage: multipass stop [options] [<name> ...]
Stop the named instances. Exits with return code 0
if successful.

Options:
  -h, --help          Displays help on commandline options
  -v, --verbose       Increase logging verbosity. Repeat the 'v' in the shor
                      option for more detail. Maximum verbosity is obtained
                      4 (or more) v's, i.e. -vvvv.
  --all               Stop all instances
  -t, --time <time>   Time from now, in minutes, to delay shutdown of the
                      instance
  -c, --cancel        Cancel a pending delayed shutdown
  --force             Force the instance to shut down immediately. Warning:
                      could potentially corrupt a running instance, so use w
                      caution.

Arguments:
  name                Names of instances to stop. If omitted, and without th
                      --all option, 'primary' will be assumed.
```


Last updated on Feb 19, 2025

suspend

The `multipass suspend` command without any argument will suspend the Primary instance (and fail, if it doesn't exist). You can also pass one or more instance names or the `--all` option to suspend more instances at the same time.

> **Note**
>
> Only instances in `Running` status can be suspended.

For example:

```
multipass stop boisterous-tortoise
multipass suspend boisterous-tortoise
```

If check your instances with `multipass list`, you'll see that its status is now set to `Suspended`:

```
Name                    State             IPv4             Image
boisterous-tortoise     Suspended         --               Ubuntu 22.04 LTS
```

Suspended instances can be resumed with the `multipass start` command.

The full `multipass help suspend` output explains the available options:

```
Usage: multipass suspend [options] [<name> ...]
Suspend the named instances, if running. Exits with
return code 0 if successful.

Options:
  -h, --help     Display this help
  -v, --verbose  Increase logging verbosity. Repeat the 'v' in the short op
                 for more detail. Maximum verbosity is obtained with 4 (or
                 v's, i.e. -vvvv.
  --all          Suspend all instances

Arguments:
  name           Names of instances to suspend. If omitted, and without the
                 --all option, 'primary' will be assumed.
```


Last updated on Feb 19, 2025

transfer

The `multipass transfer` command copies files between host and instance, without the need of mounting a folder on the instance.

For example, to copy a local file `local_file.txt` to the default home folder of the instance `good-prawn`, use the command:

```
multipass transfer local_file.txt good-prawn:.
```

Conversely, to copy a file `instance_file.txt` from the default home folder of the `ample-pigeon` instance to the current working folder, use the command:

```
multipass transfer ample-pigeon:remote_file.txt .
```

The source file can be the host standard input, in which case the stream will be written on the destination file on the instance. In the same way, the destination can be the standard output of the host and the source a file on the instance. In both cases, the standard input and output are specified with `-`.

If the target path does not exist, you can use the `--parents` option, that will create any missing parent directories. For example, if you run the command:

```
multipass transfer local_file.txt ample-pigeon:non/existent/path/remote_fil
```

and the system informs you that:

```
[2022-10-11T13:07:25.789] [error] [sftp] remote target does not exist
```

you can use the `--parents` option to create the missing parent folders:

```
multipass transfer --parents local_file.txt ample-pigeon:non/existent/path/
```

You can also copy an entire directory tree using the `--recursive` option:

Skip to content latest

```
multipass transfer --recursive ample-pigeon:dir .
```

{caution}Symbolic links are not followed during recursive transfer.

The full `multipass help transfer` output explains the available options:

```
Usage: multipass transfer [options] <source> [<source> ...] <destination>
Copy files and directories between the host and instances.

Options:
  -h, --help       Displays help on commandline options
  -v, --verbose    Increase logging verbosity. Repeat the 'v' in the short
                   option for more detail. Maximum verbosity is obtained wi
                   (or more) v's, i.e. -vvvv.
  -r, --recursive  Recursively copy entire directories
  -p, --parents    Make parent directories as needed

Arguments:
  source           One or more paths to transfer, prefixed with <name:> for
                   paths inside the instance, or '-' for stdin
  destination      The destination path, prefixed with <name:> for a path
                   inside the instance, or '-' for stdout
```


Last updated on Feb 19, 2025

umount

> See also: `mount` , Mount, How to share data with an instance.

The `umount` command without any options unmounts all previously defined mappings of local directories from the host to an instance.

You can also unmount only a specific mount if you specify the desired path.

The full `multipass help umount` output explains the available options:

```
Usage: multipass umount [options] <mount> [<mount> ...]
Unmount a directory from an instance.

Options:
  -h, --help     Displays help on commandline options
  -v, --verbose  Increase logging verbosity. Repeat the 'v' in the short op
                 for more detail. Maximum verbosity is obtained with 4 (or
                 v's, i.e. -vvvv.

Arguments:
  mount          Mount points, in <name>[:<path>] format, where <name> are
                 instance names, and optional <path> are mount points. If
                 omitted, all mounts will be removed from the named instanc
```

Last updated on Feb 19, 2025

unalias

> See also: `alias` , Alias, How to use command aliases

The `multipass unalias` command removes a previously defined alias.

```
multipass unalias name
```

This will remove the given alias `name` , returning an error if the alias is not defined.

> **Note**
>
> If an instance is deleted and purged, it is not necessary to run `unalias` for the aliases defined on that instance, as they are automatically removed.

You can remove multiple aliases in a single command:

```
multipass unalias name1 name2 name3
```

Or, use the argument `--all` to remove all the defined aliases:

```
multipass unalias --all
```

The full `multipass help unalias` output explains the available options:

```
Usage: multipass unalias [options] <name> [<name> ...]
Remove aliases

Options:
  -h, --help     Displays help on commandline options
  -v, --verbose  Increase logging verbosity. Repeat the 'v' in the short op
                 for more detail. Maximum verbosity is obtained with 4 (or
                 v's, i.e. -vvvv.
  --all          Remove all aliases from current context

Arguments:
  name           Names of aliases to remove
```


Last updated on Feb 19, 2025

version

The `multipass version` command without an argument will display the client and daemon versions of Multipass; for example:

```
multipass  1.0.0
multipassd 1.0.0
```

If there is an update to Multipass available, it will be printed out in addition to the standard output; for example:

```
multipass  1.0.0
multipassd 1.0.0

###############################################################################
Multipass 1.0.1 release
Bugfix release to address a crash

Go here for more information: https://github.com/canonical/multipass/releas
###############################################################################
```

The full `multipass help version` output explains the available options:

```
Usage: multipass version [options]
Display version information about the multipass command
and daemon.

Options:
  -h, --help         Displays help on commandline options
  -v, --verbose      Increase logging verbosity. Repeat the 'v' in the shor
                     option for more detail. Maximum verbosity is obtained
                     4 (or more) v's, i.e. -vvvv.
                     ormat>  Output version information in the requ
                     Valid formats are: table (default), js
```

Skip to content latest

Last updated on Feb 19, 2025