

MEELMO

Community Edge VM Setup Guide

Version 0.3.1-rc · 13 October 2026

Contents

How to get the free Meelmo Community Edge virtual machine running on your own PC: the host settings it needs, importing it, the one-time setup at its console, and what to check when something does not come up. For the edition's limits and what it includes, see *COMMUNITY_EDITION.md*; for everything you can do once it is running, the *USER_MANUAL.md*.

The VM is a complete Meelmo workstation on Ubuntu 24.04: the node, the web HMI and Studio, and the Desktop, with a simulated production line already feeding it. It is one person's workstation: one account, one live viewer at a time, 32 tags.

1. Before you start

What you need

Hypervisor	VMware Workstation Pro or Player 17, or ESXi/vSphere 7 or later, with the OVA. QEMU/KVM, Proxmox VE and VirtualBox with the QCOW2 (section 7).
Host	x86-64 with hardware virtualisation (Intel VT-x or AMD-V) enabled in the BIOS.
Memory and CPU for the VM	4 GB RAM and 2 vCPUs (what the VM is set up with).
Disk	About 15 GB free: the download is about 3 GB and the VM's disk grows to 12 GB.
Network	A wired or Wi-Fi network for the host; internet access is recommended for time sync, not required.

Get the host's clock right first

The VM makes its own certificate authority when it first starts, from its clock. If that clock is wrong and is corrected afterwards, the VM's certificates can be refused as *not yet valid* (or *expired*) by the Desktop and by browsers.

- **Set the host's time automatically.** On Windows: **Settings ► Time & language ► Date & time ► Set time automatically** on, and **Set time zone automatically** on (or your zone chosen by hand). Press **Sync now** once.
- **Why a Windows host matters.** Windows keeps the PC's hardware clock in *local* time; Linux expects it in UTC. The VM is set up to be handed UTC (`rtc.diffFromUTC = 0`) and to wait for a time sync before it makes its certificates, and Meelmo's certificates start a day early, so a corrected clock no longer breaks them. Builds before 0.3.1-rc had none of this: a host in Rome (UTC+2) gave the VM a clock two hours fast, and its certificates were refused for two hours after setup.
- **Keep VMware's time sync on.** The VM ships with **Synchronize guest time with host** enabled (VM ► Settings ► Options ► VMware Tools). Leave it on.

Choose the network mode

Mode	Use it when	What works
Bridged (recommended)	Other PCs on your network should reach the HMI	The VM gets its own address on your network; <code>https://meelmo-edge.local:8443/</code> works from other PCs (and <code>meelmo-edge</code> from Windows).
NAT	Only this PC uses the VM	The VM can reach the internet; other PCs cannot reach the HMI without port forwarding of 8443. Use the VM's address, not its name.
Host-only	Isolated testing	No internet, so no time sync; only this PC reaches the VM.

You can change the mode later in **VM ► Settings ► Network Adapter**; the VM picks up its new address at the next boot (the console shows it).

2. Download and check

Your personal download page (the link Meelmo emails you) offers the same VM in two formats:

Format	File	For
OVA	<code>Meelmo-Community-Edge-0.3.1-rc.ova</code> , about 3 GB	VMware Workstation, Player, ESXi and vSphere (sections 3 to 6)
QCOW2	<code>Meelmo-Community-Edge-0.3.1-rc-amd64.qcow2</code> , about 4.4 GB	QEMU/KVM and Proxmox, or to convert for VirtualBox (section 7)

The page shows each file's SHA-256. Check it before importing:

- Linux or macOS: `sha256sum Meelmo-Community-Edge-0.3.1-rc.ova`
- Windows PowerShell: `Get-FileHash .\Meelmo-Community-Edge-0.3.1-rc.ova -Algorithm SHA256`

and compare the result with the page. Interrupted downloads resume; the link allows a limited number of downloads and expires on the date the page shows.

If you took the VM from a build instead (a zip with `SHA256SUMS` inside), extract it and run `sha256sum -c SHA256SUMS` in that folder.

A hash that does not match means a damaged or altered download: download it again.

3. Import into VMware

Workstation or Player

1. **File ► Open**, choose the `.ova` , give the VM a name and a folder, **Import**.
2. If Workstation says the file failed a specification check, choose **Retry**: it relaxes the check and imports the VM as made.
3. Before the first start, open **VM ► Settings**:

- **Network Adapter: Bridged** (section 1), with **Replicate physical network connection state** ticked on a laptop.
- **Options ► Advanced ► Firmware type: UEFI**, Secure Boot off (already so; check it if the VM shows a firmware screen or "no operating system").
- Leave **Memory** at 4 GB or more and **Processors** at 2 or more.

ESXi or vSphere

Deploy OVF Template (or **Create/Register VM ► Deploy a virtual machine from an OVF or OVA file**), map `VM Network` to the network the HMI should be reachable on, and keep the firmware at EFI.

4. First start: the console setup

Power the VM on and stay at its console. The first start asks two things, once.

1. **Your account.** A username and a password. This one account is both the Linux login (with `sudo`) and the **Meelmo administrator** for the Desktop and the web HMI.
 - The password needs at least 12 characters and three of: lowercase, uppercase, digit, symbol. Type it twice.
 - `meelmo` is reserved for the product's own service; choose another username.
 - There is no default password and no other account.
2. **Your time zone**, as a name such as `Europe/Rome` or `America/Monterrey` (any capitalisation). **Enter** keeps UTC. Clocks, logs and the HMI show this zone; data is always recorded in UTC. The time zone does not change the clock itself.
3. Nothing else: there is no setup page or key for the Community VM.

The VM then sets itself up. On its console you see one line per step:

Step	What it does
<code>phase=clock</code>	Waits up to 90 seconds for a time sync, then prints the clock in UTC and whether it was synchronised.
<code>phase=setup</code>	Installs Meelmo (node, web, Desktop). This takes a few minutes.
<code>MEELMO-VERIFY: ... =ok</code>	Checks the installation.
<code>phase=pki</code>	Makes the site's certificate authority and the node's certificate.
<code>phase=network-config</code> , <code>hmi-start</code> , <code>root-fingerprint</code>	Opens the HMI on the network and prints the root fingerprint and the HMI address.
<code>phase=idle-sign-out</code>	Turns Meelmo's idle sign-out off (one person's workstation).
<code>phase=hmi-example</code>	Installs the <i>Example line</i> operator screen for everyone.
<code>phase=desktop-sign-in</code>	Signs the Desktop in with your account, so it does not ask again.

Write down the root fingerprint (`root-sha256 Fingerprint=...`) and the HMI address the console shows; you need them to trust the VM from other PCs.

After the last step the VM shows the graphical login.

If `phase=clock` says `synchronized=no`, the VM had no time source (no internet, or host-only network). Setup goes on. Check the clock on the desktop against a real clock once you are signed in; if it is wrong, see section 6.

5. Using the VM

Sign in and open Meelmo

1. At the graphical login, sign in with the account you created.
2. On the desktop:
 - **Meelmo HMI** opens the web HMI in the VM's browser, at `https://127.0.0.1:8443/`. Sign in with the same username and password. It opens on **Example line**: state, rate, count, temperature, pressure, level, motor load, rejects, vibration, trends and an alarm list, all from the simulated line.
 - **Meelmo Community** opens the Desktop engineering client, already signed in to the node.
3. Build your own screens in the **Studio** (from the web HMI), and your own connections under **Administration ► I/O**.

From another PC

1. Open `https://meelmo-edge.local:8443/trust` (bridged network) or `https://<VM-address>:8443/trust`. The browser warns the first time; that is expected, because it does not know the VM's certificate authority yet.
2. Compare the fingerprint on the page with the one the VM's console showed. **Only if they match**, download the root certificate and install it as a trusted root:
 - Windows: double-click it ► **Install Certificate ► Local Machine ► Place all certificates in the following store ► Trusted Root Certification Authorities**.
 - macOS: open it in **Keychain Access**, then set **When using this certificate** to **Always Trust**.
 - Firefox keeps its own list: **Settings ► Privacy & Security ► Certificates ► View Certificates ► Authorities ► Import**.
3. Restart the browser and open `https://meelmo-edge.local:8443/`.

Limits of the Community edition

- **One live viewer.** Live values stream to one browser or Desktop at a time. A second one shows the page with its values marked *Disconnected*, and takes the stream over by itself within about a minute of the first one closing.
- **32 enabled tags.** The simulated line uses ten.
- **History is off** until you enable the embedded historian under **Administration ► Historian**.
- No idle sign-out on this VM. Closing the browser or the Desktop still signs you out; to set a timeout, use **Administration ► Users & permissions**.

6. When something is wrong

What you see	Why	What to do
Desktop: <i>certificate is not yet valid (compare this machine's clock with the real time)</i> ; browser in the VM: <i>This Connection is Not Secure</i> at 127.0.0.1:8443 ; console: web-profile unchanged (URLError)	The VM's clock was corrected backwards after it made its certificates (builds before 0.3.1-rc, typically on a Windows host whose zone is ahead of UTC).	Wait: the certificate becomes valid on its own once the clock catches up (two hours for UTC+2). The first-boot steps that failed (Desktop sign-in, example screen, idle sign-out) are not redone; sign in by hand. Better: delete the VM and import a 0.3.1-rc or later OVA.
Desktop: <i>certificate has expired</i>	The VM's clock is far ahead of the real time.	Fix the host clock (section 1), keep VMware time sync on, reboot the VM.
Values marked <i>Disconnected</i>	Another browser or the Desktop holds the one live stream.	Close the other viewer; this one recovers within about a minute.
meelmo-edge.local does not open from another PC	The VM is behind NAT, or the network blocks multicast name lookup.	Use the address the console shows, or switch to bridged networking.
The browser inside the VM shows a blank page	An old build without a desktop session for the browser.	Import a current OVA.
A setup page opens at 127.0.0.1:8443	Builds before 0.3.1-rc, when nobody had yet signed in to the web.	Sign in once with your account, or import a current OVA.
Firmware screen or "no operating system"	The VM is set to BIOS.	VM ► Settings ► Options ► Advanced ► Firmware type: UEFI , Secure Boot off.
The console stops at MEELMO-HANDOFF: failed phase=...	That step failed.	Reboot the VM: the console asks for your password again and retries.

Checking and fixing the clock

In a terminal on the VM (**Applications ► Terminal**):

```
timedatectl          # "System clock synchronized: yes" and the right local time
sudo timedatectl set-ntp true
```

Repairing or redoing the setup

Sign in on the text console (**Ctrl+Alt+F2**) with your account and run `sudo meelmo-commission`. It reinstalls Meelmo over the existing data and keeps the site's certificate authority. To start completely afresh, delete the VM and import the OVA again.

7. QEMU/KVM, Proxmox and VirtualBox

Take the **QCOW2** from your download page. It is the same validated disk as the OVA.

Whatever the hypervisor, the VM needs **UEFI firmware with Secure Boot off**, 2 vCPUs, 4 GB RAM and a bridged network, and its hardware clock in **UTC**.

- **QEMU/KVM (virt-manager): New VM ► Import existing disk image**, choose the QCOW2, operating system *Ubuntu 24.04*, then tick **Customize configuration before install** and set **Overview ► Firmware** to a UEFI firmware *without* Secure Boot (for example `OVMF_CODE_4M.fd`, not `...secboot...`). Network: your bridge.
- **Proxmox VE:** create a VM with **BIOS: OVMF (UEFI)**, machine **q35**, an EFI disk with **Pre-Enroll keys unticked** (that is what keeps Secure Boot off), no CD, 2 cores, 4096 MB and network on `vmbr0`. Delete the empty hard disk the wizard made, copy the QCOW2 to the Proxmox host and import it:

```
qm importdisk <vmid> Meelmo-Community-Edge-0.3.1-rc-amd64.qcow2 <storage>
```

then attach the imported *Unused Disk* as **SATA** or **VirtIO SCSI** and put it first under **Options ► Boot Order**.

- **VirtualBox:** convert it first, then attach the VDI to a SATA controller of a new *Ubuntu (64-bit)* VM with **Enable EFI** ticked, 2 vCPUs, 4 GB RAM and a bridged adapter. Tick **Hardware Clock in UTC Time** (System ► Motherboard).

```
qemu-img convert -f qcow2 -O vdi Meelmo-Community-Edge-0.3.1-rc-amd64.qcow2 Meelmo-Community.vdi
```

The image is validated under QEMU/OVMF at every build, which is what Proxmox runs too. VMware is the setup used in practice; Proxmox and VirtualBox carry the same disk but have not been through acceptance of their own.