

qemu-web-desktop: Managing Virtual machines

This documentation assumes you already have installed and configured the service. Refer to the `INSTALL.md` and `CONFIGURE.md`. You are also advised to look at the `USAGE.md` page for the different types of use for this service.

Table of contents:

1. Creating virtual machines
2. Adding virtual machines to the service
3. Automatic configuration of virtual machines via `qwdctl`
4. Manual configuration of virtual machines
5. Issues: missing QEMU Guest Agent

Creating virtual machines

It is possible to create a VM from an ISO, just like you would boot physically. An empty disk is first created (here with size 10GB).

```
qemu-img create -f qcow2 machine1.qcow2 10G
```

Then you should boot from an ISO file (here indicated as `file.iso`)

```
qemu-system-x86_64 -m 4096 -smp 4 -hda machine1.qcow2 -name MySuperbVM -boot d -cdrom file.iso
```

and install the system on the prepared disk. If you can not start `qemu/kvm` with a message `Could not access KVM kernel module: Permission denied;`
`qemu-system-x86_64: failed to initialize kvm: Permission denied,`
execute the command `sudo /sbin/adduser $USER kvm`, then `logout/login`.

You may also convert an existing VDI/VMDK file (VirtualBox and VMWare formats - here `file.vmdk`) into QCOW2 for QEMU (here `machine1.qcow2`) with command:

```
qemu-img convert -f vmdk -O qcow2 file.vmdk machine1.qcow2
```

In case you have an OVA file for VirtualBox, extract it, and convert it afterwards to a QCOW2 file (optional as QWD can use the vmdk files).

```
tar -xvf file.ova
```

```
qemu-img convert -f vmdk -O qcow2 file.vmdk machine1.qcow2
```

Last, you may dump an existing physical disk (with a functional system - here from device `dev/sda`) into a QCOW2 format:

```
qemu-img convert -O qcow2 /dev/sda machine1.qcow2
```

The QCOW2 format allows to resize disks, for instance with (increase by +50GB):

```
qemu-img resize machine1.qcow2 +50G
```

If a VM file gets too large, you can (re)compress it with command:

```
qemu-img convert -O qcow2 -c image.qcow2 image_compressed.qcow2
```

Adding virtual machines to the service

QWD is designed to run virtual machines with the host architecture, e.g. x86_64 on Intel/AMD and aarch64 on ARM. This choice provides the best performance.

The easiest is to make use of the `qwdctl` tool (see below). Other options are activated by un-commenting sections in the file `/usr/share/qemu-web-desktop/html/desktop/index.html` (can be done with `sudo qwdctl edit web`).

Common virtual machine image file formats are supported (QCOW2, VDI, VMDK, RAW, VHD/VHDX, QED), as well as ISO live images.

:exclamation: In order for the clipboard copy-paste to properly work between the host and the sessions, the following packages should be installed in the virtual machines:

- `qemu-guest-agent spice-vdagent`

Automatic configuration of virtual machines via `qwdctl` The best way to manage virtual machines is to use the `qwdctl` tool. It is possible to add new VM's, disable (hide) VM's, re-enable hidden VM's, purge unused VM files.

Command	Description	Example
<code>qwdctl status</code>	List running sessions, enabled, disabled and orphan VM.	<code>qwdctl</code>
<code>qwdctl edit VM</code>	Edit an existing VM entry.	<code>qwdctl edit slax.iso</code>
<code>qwdctl enable VM</code>	Add or (re)enable a VM. The VM can be given as a full file path, a URL or a VM file already present in <code>/var/lib/qemu-web-desktop/machines/</code> .	<code>qwdctl enable /path/to/VM.qcow2</code> <code>qwdctl enable http://url/to/VM.qcow2</code> <code>qwdctl enable slax.iso</code>
<code>qwdctl disable VM</code>	Disable a VM. The VM name must match an active entry.	<code>qwdctl disable slax.iso</code>
<code>qwdctl purge qwdctl purge VM</code>	Delete VM files that are not running, nor active. You may restrict the purge to a given name.	<code>qwdctl purge qwdctl</code> <code>qwdctl purge slax.iso</code> <code>qwdctl purge slax.iso --yes</code>
<code>qwdctl update qwdctl download</code>	Take into account the configuration and update the service landing page <code>/usr/share/qemu-web-desktop/html/desktop/index.html</code> .	<code>qwdctl update</code>

To enable an existing example, such as the `slax.iso` VM, simply use:

```
sudo qwdctl enable slax.iso
```

To add a new VM and make it available, use:

```
sudo qwdctl enable /path/to/VM.ext "Description"
```

The VM file can be given as a ISO, QCOW2, VDI, VMDK, RAW, VHD/VHDX file, as well as a URL. The “Description” can be omitted, in which case the VM name is used as descriptor. The same command can be used to re-enable a VM that has been disabled. New VM file descriptors are stored in `/etc/qemu-web-desktop/config.d/*.ini`. The VM files are stored e.g. in `/var/lib/qemu-web-desktop/machines/`.

To disable (i.e. hide) a VM, use:

```
sudo qwdctl disable VM
```

where the VM must match an existing file e.g. in `/var/lib/qemu-web-desktop/machines/`. The VM file is not removed, it is only made inactive in the service landing page. It can be re-enabled with `qwdctl enable VM`.

To edit the description for a VM (the message that appears on the web page), use e.g.:

```
sudo qwdctl edit VM
```

The configuration file holding the VM will be edited.

Last, to clean VM files which are both not running and disabled, use:

```
sudo qwdctl purge
```

An optional VM argument allows to specify a starting name for removal, e.g. `qwdctl purge slax` removes `slax*`. The command requests for confirmation, but you may add the `--yes` for a fully automatic configuration. This allows to definitely remove leftover VM files (orphan).

Semi-Automatic configuration of virtual machines via `qwdctl` Each entry in the configuration file `/etc/qemu-web-desktop/machines.conf` (and `/etc/qemu-web-desktop/config.d/*.ini`) spans on 2 or 3 lines:

- `[name.ext]`
- `url=[URL to ISO, QCOW2, VDI, VMDK, RAW, VHD/VHDX virtual machine disk]` (optional if the file is already present)
- `description=[description to be shown in the login page]`

Images listed in the configuration file *without* a `url=` parameter are expected to be downloaded by hand and installed into `/var/lib/qemu-web-desktop/machines/` by the local administrator. In this case, just specify the `[name.ext]` and `[description]`.

Images with a `[url]` line are downloaded (requires a configured network connection). Add ISO or virtual machine files for your architecture, and comment the others with `;`.

You may edit this file manually, or with the command:

```
sudo -E qwdctl edit machines
```

You may set the **EDITOR** environment variable to select the text editor to use.

:warning: In case the machine list does not appear in the landing page, first refresh the landing page, or change the `machines_insert=no` to `machines_insert=yes` in `/usr/bin/qwdctl`.

Then actually launch (done automatically after `sudo qwdctl edit machines`):

```
sudo -E qwdctl download
```

or, to only update existing machines, use:

```
sudo -E qwdctl refresh
```

To list running sessions, use:

```
qwdctl status
```

To stop active sessions matching a given TOKEN, use (with caution):

```
sudo qwdctl stop TOKEN
```

The command `qwdctl` alone will display all possible options.

Manual configuration of virtual machines You may also do this by hand. Place any ISO, QCOW2, VDI, VMDK, RAW, VHD/VHDX, QED virtual machine file in e.g. `/var/lib/qemu-web-desktop/machines`.

```
ls /var/lib/qemu-web-desktop/machines
```

```
slax.iso  slitaz-rolling.iso  TinyCore-current.iso  machine1.qcow2 ...
```

Then create/edit the `/usr/share/qemu-web-desktop/html/machines.html` (link from `/var/lib/qemu-web-desktop/machines.html`) web page and add entries to reflect the VM files in `html/machines`:

```
<option value="slax.iso">Slax (Debian)</option>
<option value="TinyCore-current.iso">TinyCore Linux</option>
...
<option value="machine1.qcow2">My superb VM</option>
...
```

You can also comment/uncomment sections (e.g. GPU, user script, one-shot) at will in the main web page `/usr/share/qemu-web-desktop/html/desktop/index.html` (you may use `sudo qwdctl edit web` for this purpose). Defaults will then be used.

:+1: This project provides minimal ISO's for testing (for `x86_64` in `html/desktop/machines`):

- Slax a very compact Debian-based system (265 MB)
- TinyCore a very compact, old-style Linux (24 MB - yes)
- SliTaz a very compact, old-style Linux (54 MB - yes)

We also recommend other small linux systems for `x86_64`:

- <https://puppylinux-woof-ce.github.io/index.html> (small, comes as Debian and Slackware base)
- <https://www.porteus.org/> (small, SlackWare based)
- <https://archbang.org/> (Arch based)
- <https://www.bodhilinux.com/> (Ubuntu/Debian based)

and for `arm64`:

- <https://cdimage.debian.org/cdimage/release/current/arm64/iso-dvd/>
- <https://www.armbian.com/uefi-arm64/>
- <https://archlinuxarm.org/>

There exists some virtual machine repositories, for instance:

- <https://marketplace.opennebula.systems/appliance>
- <https://www.osboxes.org>

Issues: missing QEMU Guest Agent

In some virtual machines or ISO's, you may get errors such as:

- Dependency failed for QEMU Guest Agent
- Time out waiting for device `/dev/virtio-ports/org.qemu.guest_agent.0`

In this case, try to boot the VM manually, and install the `qemu-guest-agent` and the `spice-vdagent` packages.