

qemu-web-desktop

qemu-web-desktop is a remote desktop service that launches virtual machines in the cloud, and displays them in your browser. These machines can be used for e.g. scientific data treatment. The Data Analysis Remote Treatment Service (DARTS) is an implementation of **qemu-web-desktop** at Synchrotron SOLEIL.

Table of Contents

- Installation and Configuration
 - What is provided by this service
 - Usage
 - How it works
 - Credits
-

Installation and configuration

- Installation instructions: `INSTALL.md`
 - Installation instructions for GPUs: `GPU.md`
 - Configuration of the service: `CONFIGURE.md`
 - Add/handle virtual machines: `VIRTUAL_MACHINES.md`
-

What is provided by this service

The service allows launch a remote virtual machine, and display it in a browser window. No additional software installation is needed on the client side. This project has been developed on a Debian-class system, and is thus suited for it, but should install on basically any Linux system. Each session usually starts from the same state, all further modifications are made within snapshots, and thus not permanent.

Once installed (see `INSTALL.md`), connect to:

- `http://localhost/qemu-web-desktop`

Basically a user can enter some login/password or email. The user can click on “Create”. It is also possible to select the type of machine (system), and what resources are needed (cpu, memory).

The user credentials can be tested against nothing (all can connect), local accounts, IMAP, SMTP, and LDAP. In this case, a user ID (login name) and password are required. Authentication can also be achieved using an email sent with connection information.

When authentication is successful, a virtual machine is launched and can be displayed in the browser window.

Just click the link, and you will access the virtual machine remote desktop. Use full screen, adapt screen resolution and keyboard layout, and you'll be good to go !

From the service landing page (login, machine specifications), you may also access a monitoring page to list running sessions.

Features

- Supports all common virtual machine formats (VDI, VMDK, VHD, VHDX, QCOW2, RAW), as well as ISO live images, for `x86_64` and `aarch64` (arm64) host servers.
- Supports authentication using local, SMTP, IMAP, LDAP and email (or no authentication). Supports MFA via LDAP and email. See `CONFIG-URE.md`.
- Checks the server load to avoid DoS. See “customize” section in `CONFIG-URE.md`.
- The browser can be closed and re-connected later while the session is still running. The connection information can also be shared to allow multiple users to collaborate. It is important to remember the session URL. :warning: Beware: all users will have mouse/keyboard control, so that friendly collaboration rules must be set in place.
- No need to install anything on the client side.
- The rendering of the web service is responsive design. It adapts to the browser window size.
- Can monitor running sessions.
- Can mount host volumes.
- Automatically clean-up sessions that have passed their life-time.
- Can optionally assign physical GPU to sessions (see `GPU.md`).
- Can optionally insert and execute scripts in the virtual machine boot process.
- Can optionally distribute the sessions work-load over a farm of servers.

Usage: as a web service

First make sure the service has been installed in the `html/desktop` root level of the host, and the `cgi-bin/qemu-web-desktop.pl` e.g. in the `/usr/lib/cgi-bin`.

Open a browser and go to:

- `http://localhost/qemu-web-desktop/`

Customize your server with the usual settings (edit configuration with `sudo qwdctl edit config`):

- the user credentials. The protocol used for checking depends on the `/etc/qemu-web-desktop/config.pl` (and `config.d/*.pl`) settings (default is ‘no check’).
- the system (virtual machine) to boot.
- the number of cores (`#CPU`).
- the amount of memory.
- the life-time.

Optionally (when un-commenting sections in the web form `/usr/share/qemu-web-desktop/html/desktop/index.html`, e.g. use `sudo qwdctl edit web`)

- a GPU request (the system must have been configured as explained in `GPU.md`).
- the auto-start script to execute at boot. This script may contain any set of commands (installations via `apt`, `pip` or `conda`, configuration, start-up of a service or application, ...). The symbols `@USER@`, `@PW@`, `@SESSION_NAME@` and `@VM@` are replaced by the user name/pw, the session ID, and the virtual machine name. The script is executed with administrator privileges in the guest virtual machine. In case the server is protected by a proxy, the `service_proxy` must be set in the `/etc/qemu-web-desktop/config.pl` file (and `config.d/*.pl`) to access external resources. This option is not recommended for high security systems. The script may be:
 - a path to a script on the server
 - a URL to a distant file (URL such as on `github.com` or `gitlab.com` - make sure to provide a raw content). We provide as an example the script `https://gitlab.com/soleil-data-treatment/soleil-software-projects/trunk-in-my-car/-/raw/main/start-script.sh`
 - a string starting with `exec:` or `bash:` and followed by shell commands separated by ‘;’ or EOL. An example would be `exec: touch /tmp/hello`.
- the one-shot button, which creates virtual machines allowing only a single connection.

Then press the **Create** button. After about 10 seconds, information is displayed. Follow instructions, click the given link or scan the QR code to connect the display. You can of course access the service remotely if the server is on a network.

Connect within a browser to the displayed IP, such as:

- `http://localhost:6080/vnc.html?resize=remote&path=?token=jNIjYTUn`

Once done with the session, make sure you shut-down the remote desktop session. Do not just close the browser, suspend or logout. This is to free the resources for others once you do not need the session anymore.

You can close the browser any time, and reconnect later: the session remains active, any calculation will proceed. To reconnect you may click again on the link. If you have lost this link, first click on the **Manage sessions** button

(or select the same item in the machine list), then find the relevant session in the table, and select the CONNECT item. You may as well stop the session prematurely with the STOP button.

Last, it is possible to send the session link to your colleagues (or the QR code), so that you all see the same desktop and work together. You should however notice that rules must be adopted to share your multiple keyboards and mice.

Once the maximum life-time is over, the session is automatically stopped and cleaned-up. There is no way to recover a cleared session.

Usage: local (for testing)

It is possible to test that all works with:

```
cd qemu-web-desktop/src
make test
```

A URL will be displayed, and the session will then be closed after 30 seconds.

The `qemu-web-desktop.pl` script can be used as a command with additional arguments. The full list of supported options is obtained with:

```
qemu-web-desktop.pl --help
```

You may also launch any virtual machine manually with:

```
VM=TinyCore-current.iso
VM_DIR=/var/lib/qemu-web-desktop/machines
/usr/lib/cg-bin/qemu-web-desktop.pl --dir_snapshots=/tmp --dir_cfg=/tmp --dir_machines=$VM_DIR
```

or

```
qwdctl start VM ...
```

Additional arguments override the default configuration, such as:

qwdctl start VM [option...]	Description
--snapshot_alloc_cpu=VALUE	Number of CPU cores to use. Default: 1 (qemu equivalent: <code>-smp VALUE</code>)
--snapshot_alloc_disk=VALUE	Disk size to create for ISO's, in GB. Default: 10
--snapshot_alloc_mem=VALUE	Memory to allocate to session, in GB. Default: 4 (qemu equivalent: <code>-m VALUE*1024</code>)

<code>qwdctl start VM [option...]</code>	Description
<code>--snapshot_use_master=VALUE</code>	When 1, do NOT create a snapshot, so that all changes are written to the master VM (not for ISO's). Default: 0 (changes are lost)

A URL is shown. Open a browser to view the session.

```
$ qwdctl start /var/lib/qemu-web-desktop/machines/TinyCore-current.iso
http://localhost:6005/vnc.html?resize=scale&autoconnect=true&host=localhost&port=6005
```

:warning: All sessions are started in one-shot mode, i.e. closing the browser will end the session. Except when starting with the `--snapshot_use_master=1` option (not for ISO's), the session and the associated snapshot/qcow2 file will be lost. To keep it, copy the qcow2 file before ending the session. There is no support for GPU pass-through with this manual launch.

You can force a local session to stop with any of:

```
qemu-web-desktop.pl --dir_snapshots=/tmp --dir_cfg=/tmp --session_stop=/path/to/json
qwdctl stop PID|TOKEN
```

And you can stop and clear all local sessions with any of:

```
qemu-web-desktop.pl --dir_snapshots=/tmp --dir_cfg=/tmp --service_purge=1
qwdctl stop
```

Last, you can monitor all running sessions, with:

```
qwdctl status
```

or

```
qemu-web-desktop.pl --dir_snapshots=/tmp --dir_cfg=/tmp --service_monitor=1 > /tmp/mon.html
firefox /tmp/mon.html
```

which generates an HTML file and renders it in a browser.

How it works

A static HTML page with an attached style sheet (handling responsive design), calls a perl CGI on the Apache server. This CGI creates a snapshot of the selected virtual machine (so that local changes by the user do not affect the master VM files). A `qemu` command line is assembled, typically (here 4 SMP cores and 8 GB memory):

```
qemu-system-x86_64 -m 8192 -smp 4 -hda machine1-snapshot.qcow2 -device ich9-ahci,id=ahci -e
```

The integrated QEMU VNC server is also launched, so that we can access the VM display. As indicated, we also use the `virtio-balloon` device, which allows to share the unused memory when multiple VM's are launched. When IOMMU/VFIO GPU are available, their PCI slot is passed to QEMU with the `virtio-pci` option.

A websocket is attached to the QEMU VNC, and redirected to a noVNC port, so that we can display the VM screen in a browser.

A monitoring page is also handled by the CGI script, to display the server load and running sessions. These can be killed one-by-one, or all at once.

The single perl CGI script that does all the job fits in 2300 lines. It comes with a helper `qwdctl` command.

Credits

(c) 2020- Emmanuel Farhi - GRADES - Synchrotron Soleil. AGPL3.

- Farhi, E., (2023). DARTS: The Data Analysis Remote Treatment Service. Journal of Open Source Software, 8(90), 5562, <https://doi.org/10.21105/joss.05562>
- <https://gitlab.com/soleil-data-treatment/soleil-software-projects/remote-desktop>

This project has received support from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957189 (BIG-MAP project).

This work was supported by the Paris Ile-de-France Region through the DIM-MAP CAIMAN Map InStoRe project.

We have benefited from the following web resources.

Debian/Ubuntu documentation

- <https://doc.ubuntu-fr.org/vfio> (in French)
- <https://alpha.lordran.net/posts/2018/05/12/vfio/> (in French)
- <https://passthroughpo.st/gpu-debian/>
- <https://wiki.debian.org/VGAPassthrough>
- <https://davidyat.es/2016/09/08/gpu-passthrough/>
- <https://heiko-sieger.info/low-end-kvm-virtual-machine/>

Other documentation

- <https://mathiashueber.com/windows-virtual-machine-gpu-passthrough-ubuntu/>
- https://wiki.archlinux.org/index.php/PCI_passthrough_via_OVMF

- <https://neg-serg.github.io/2017/06/pci-pass/> (ARCH linux)
- https://wiki.gentoo.org/wiki/GPU_passthrough_with_libvirt_qemu_kvm (Gentoo)
- <https://medium.com/@calerogers/gpu-virtualization-with-kvm-qemu-63ca98a6a172>

VirtualBox documentation

- <https://docs.oracle.com/en/virtualization/virtualbox/6.0/admin/pcipassthrough.html>