qemu-web-desktop: Configuration

This documentation explains how to configure the service (directories, landing page, authentication, security, load-balancing and limits). You should have already installed the qemu-web-desktop/DARTS from INSTALL.md.

The main commands to use for the service configuration are:

- main configuration: `sudo qwdctl edit config`
- web page: `sudo qwdctl edit web`
- available virtual machines: `sudo -E qwdctl edit machines`

and in case you wish to make GPU's available for the virtual machines, refer to GPU.md which boils down to `sudo qwdctl gpu` (not supported on Apple/MacOS). To manage virtual machines, refer to VIRTUAL_MACHINES.md.

Table of contents:

1. Configuration: Location of files and directories
2. Configuration: Server settings
    - Configuration: Server: Using config_script
    - Configuration: Server: Security aspects
3. Configuration: User credential settings
    - Configuration: Custom authentication mechanism
    - IMAP/SSL encryption level
4. Configuration: Landing page and generated message layout
5. Configuration: Hiding the service port
6. Configuration: Distributing the sessions work-load (load-levelling, scale-up)
7. Configuration: List of all options
8. How it works
9. Common Issues

List of files

The service configuration resides in a set of files, as follows:

| File | Description | |———————|————-| |
`/etc/qemu-web-desktop/config.pl` | Main service configuration file. The `config_script` allows further customization of VM's when they boot. | |
`/etc/qemu-web-desktop/config.d/*.pl` | Additional configuration files. It is good practice to write your specific configuration here rather than in the main `config.pl` | | `/var/lib/qemu-web-desktop/html/index.html` | Landing page Formgenerated from the template. | |
`/etc/qemu-web-desktop/index.html.tmpl` | Template page to change appearance. Customize with the configuration `layout_` options. | |
`/etc/qemu-web-desktop/machines.conf` | List the available VM's to propose in the service. Run `sudo qwdctl download` after edits. | |
`/etc/qemu-web-desktop/config.d/*.ini` | List additional VM's to propose in the service. `.conf` extension is also supported. | |
`/etc/qemu-web-desktop/config.pl:check_user_custom` | A function reference that is executed when the user clicks on *Create* to e.g. perform further checks. | | `/etc/qemu-web-desktop/config.pl:config_script` | Auto start scripts. This allows to configure the session *during* its boot. An example can be `$config{config_script}=("https://gitlab.com/soleil-data-treatment/infra-config/-/raw/master/` |
You may edit the `/etc/qemu-web-desktop/config.pl` file with the command `sudo qwdctl edit config` (or better in `/etc/qemu-web-desktop/config.d/*.pl`), to:
- adapt location of files (esp. directories to `machines`,`snapshots`). - adapt the default specification of virtual machines (cpu, mem). - adapt the restrictions for using the service (number of connections, load limit). - adapt the user credential tests you wish to use. They are all tested one after the other, until one works.
If the VMs you wish to run use a different architecture than the host running the service, you also need to adapt the `$config{qemu_exec}` and `$config{qemu_exec_options}` variables to the architecture you wish to emulate. Note that you also need to install the appropriate variant of qemu-system-XXX. The default choices are `'auto'` that make use of native virtualization and best QEMU options.
Most options below can be changed in the configuration script, or overridden with command line argument `--name=value`.
Adapt the `$config{layout_*}` optuins to tune the appearance of the web pages. This file can be edited with `sudo qwdctl edit web`, but you may also enable/disable these features with: - `qwdctl enable|disable gpu`
Changes to the configuration are immediate, and there is no need to restart the server nor the web service.
We list below some of the common options.

Configuration: Location of files and directories

The main configuration file is `/etc/qemu-web-desktop/config.pl` (edit it with `sudo qwdctl edit config`). You may alternatively add specific configuration

settings as `*.pl` files in `/etc/qemu-web-desktop/config.d/`. These files override the main configuration.

Web pages are usually in `/var/lib/qemu-web-desktop/html` (edit the service web page with `sudo qwdctl edit web`). Virtual machines are usually in `/var/lib/qemu-web-desktop`. List of available virtual machines for `qwdctl` are in `/etc/qemu-web-desktop/machines.conf` (edit with `sudo -E qwdctl edit machines`) and in `/etc/qemu-web-desktop/config.d/*.ini` (and `.conf`). These settings should be kept to their default for an Apache web server.

Locations	Default	Description
`dir_html`	`/var/lib/qemu-web-desktop/html`	HTML root. Contains the `index.html` form
`dir_service`	`/var/lib/qemu-web-desktop`	Location of virtual machines
`dir_machines`	`/var/lib/qemu-web-desktop/machines`	Default path for machines (ISO,VM)
`dir_snapshots`	`/var/lib/qemu-web-desktop/snapshots`	Where snapshots are stored
`dir_cfg`	`/tmp`	Temporary files (JSON for sessions)
`dir_novnc`	`/usr/share/novnc`	Location of noVNC directory, must contain `vnc.html`
`dir_websockify`	`websockify`	Websockify executable
`dir_mounts`	`(/mnt,/media)`	Volumes from host to mount in guests. Use e.g. `mount -t 9p -o trans=virtio,access=client host_media /mnt/media` in guest. The last word of the mount path is used to build the 9p label `host_<last_word>`.
`service_logfile`	`/var/log/qemu-web-desktop.log`	Service log file. Must exist and be RW

Configuration: Server settings

Important options	Default	Description
`snapshot_lifetime`	86400	Maximum time in seconds above which sessions are stopped. The clean-up occurs upon a new service request (new session or monitoring).
`service_max_load`	0.8	Maximal load of the machine, in 0-1 where 1 means all CPU's are used
`service_max_session_nb`	10	Maximum number of simultaneous sessions
`service_max_cpu_fraction_nb_per_user`	0.3	Max CPU load fraction per user
`service_max_mem_fraction_nb_per_user`	0.3	Max memory fraction per user
`service_max_gpu_nb_per_user`	1	Max GPU number per user (when GPU is configured and activated)
`service_port`	6080	The port to which the display will be broadcast. This is shown in the client URL. When `service_port_multiple` is set, the port is chosen randomly in `[service_port:service_port+service_max_instan`
`service_port_vnc`	5901	The base internal VNC port to use
`service_port_multiple`	0	When true, use one websockify port per instance (e.g. :6080+rand). When false, a single port is used for all sessions with random tokens, except for one-shot sessions.

Important options	Default	Description
`service_proxy`	“”	A proxy URL to pass through when getting external scripts, e.g. “http://xxx.yy.z:port/”. This is used for the `config_script` and `fallback_servers` options.
`certificate_crt`	`/etc/apache2/certificate.crt`	A certificate (CERT) bundle in order to use HTTPS. The KEY must also be available. The web server should use the same certificates. After setting these, KILL any running websockify: `sudo killall websockify`
`certificate_key`	`/etc/apache2/certificate.key`	A certificate KEY in order to use HTTPS. The CERT must also be available. The web server should use the same certificates.
`fallback_servers`	“”	A comma-separated list of servers, e.g. `http://server1,server2,195.221.4.1`. URL, server names and IP are allowed. When the current server is overloaded (cpu,mem,GPU), the request is sent to the other servers.
`config_script`	(““)	An array of strings specifying scripts to execute at boot, as root (see below).
`layout_level`	1	Output level for the connect page (after Create). 0=minimalist; 1=normal; 2=verbose

Important options	Default	Description
`layout_title`	'DARTS'	The title you wish to give to the connect page (after Create), e.g. your service, lab, name, enterprise, ...
`layout_header`	service url and DARTS logo	The connect page header
`layout_footer`	service url	The connect page footer
`layout_resize`	`remote`	noVNC Screen rescale/resize. Can be 'scale' (fixed ratio) or 'remote' (scales to browser size)
`layout_cpu`	`[2, 4, 8]`	Number of CPU selectable by the user.
`layout_memory`	`[1, 2, 4]`	Amount of memory selectable by the user, in GB.
`layout_lifetime`	`[1,2,4]`	Life-time selectable by the user, in days (integers).
`layout_gpu`	0	Hide or show GPU section. Use `qwdctl gpu` to configure GPU.
`oneshot`	0	Hide or show/activate one-shot sessions.

The `layout` options are interesting to customize the appearance of the service to your needs, in addition with the landing page.

Configuration: Server: Using `config_script` :warning: This feature is not supported on Apple/MacOS (except `qemu:` options).

Each `config_script` string can be given as:

- a URL "http://some/url"
- a path "/some/local/path/to/script" (on the host server)
- a string starting with `exec:` followed by shell commands separated by EOL or `;`
- a string starting with `virt-customize:` followed by one-line commands separated by EOL `\n`. (see: https://libguestfs.org/virt-customize.1.html).
- a string starting with `qemu:` followed by a list of QEMU options separated by spaces.

The symbols `@USER@` `@SESSION_NAME@` and `@VM@` are replaced by the user name, the session ID, and the virtual machine name.

In addition, when the above script description is preceded by `if(EXPR):`, the given expression is evaluated (with Perl) and the script is only executed when result is True. The `EXPR` condition may use the `@...@` symbols above.

This way, it is possible to specify scripts that apply to given virtual machines with e.g.:

- `if("@VM@" =~ /debian/i): http://some/url` (only for *debian* VM's, case insensitive).
- `if("@VM@" =~ /debian/): exec: touch /tmp/my_script_is_executed` (only for *debian* VM's, case sensitive).
- `if("@VM@" =~ /unstable/i): /host/path/script.sh` for a file from the host.
- `if("@USER@" =~ /farhie/): http://some/url` (only for a given user).

Scripts can be specified in e.g. `/etc/qemu-web-desktop/config.pl: config{config_script}` (and better in `/etc/qemu-web-desktop/config.d/*.pl`).

Configuration: Server: Security aspects

:warning: Note about the used ports

Highest security: The default setting is `service_port_multiple=0` which indicates that a single port is used for all sessions. The 'one-shot' option in the `/var/lib/qemu-web-desktop/html/index.html` file should be left commented (disabled). The HTTPS certificates should as well be set (see above). The user scripts should also better be inactivated in the form, and the VM's should not allow administrator privileges (e.g. `sudo`). Last, you may activate MFA/2FA by setting `check_user_mfa=1` which sends an email with the session VNC password. The user email must be valid from e.g. ID (landing page) or LDAP. These are the recommended settings for a secured service. It is also recommended to hide the service port through a reverse proxy (see below).
Medium security: Optionally use `qwdctl enable one-shot` to un-comment the 'one-shot' option in the landing page. One specific port will be used for each session, and be closed as soon as the session browser tab is closed. Other sessions will use a single shared port when `service_port_multiple=0`.
Low security: When `service_port_multiple=1`, each session has its own communication port. You can use `qwdctl enable one-shot` to un-comment the 'one-shot' section in the `/var/lib/qemu-web-desktop/html/index.html` file. The ports `service_port` up to `service_port+service_max_instance_nb` must be allowed on the network.

In a high security level, it is highly recommended to configure a firewall in order to restrict e.g. the SSH connections from the running sessions to other local

infrastructure servers. For instance, one would use:

```
# allow SSH access to the QEMU host [10.0.2.2] (insert at top -I)
/sbin/iptables -I OUTPUT -d 10.0.2.2        -p tcp --dport ssh -j ACCEPT

# restrict SSH access to a local domain (append at end -A)
/sbin/iptables -A OUTPUT -d 192.168.0.0/16  -p tcp --dport ssh -j REJECT
/sbin/iptables -A OUTPUT -d 10.0.0.0/8      -p tcp --dport ssh -j REJECT
```

We also encourage to hide the service port, as detailed below. This is done automatically under Debian/Ubuntu.

Configuration: User credential settings

It is possible to activate more than one authentication mechanism, which are tested until one works. The details of the SMTP, IMAP and LDAP server settings should be set in the e.g. `/etc/qemu-web-desktop/config.pl` script (and better in `/etc/qemu-web-desktop/config.d/*.pl`). The preferred authentication is the local one.

WARNING: Without any authentication mechanism set, the access is granted by default.

User authentication	Default	Description
`check_user_with_local`	0	When set, the user ID/password is checked against local accounts (via ssh). This authenticator requires `ssh` to be available on the server (e.g. `apt install openssh-server`). The authentication will e.g. use PAM, so that any PAM-plugin authenticator can be used transparently.
`check_user_with_email`	0	When set and user ID is an email, a message with the connection information is sent as authentication

User authentication	Default	Description
`check_user_with_imap`	0	When set, the user ID/password is checked against specified IMAP server
`check_user_with_smtp`	0	When set, the user ID/password is checked against specified SMTP server
`check_user_with_ldap`	0	When set, the user ID/password is checked against specified LDAP server
`check_user_custom`	""	May point to a function reference to allow any identification mechanism (see below)
`check_user_mfa`	0	When set, a VNC password is required, which is sent via email. The user MUST have a valid email defined from the landing page (`check_user_with_email=1`) or e.g. the LDAP (`check_user_with_ldap=1`). The SMTP server must be defined.
`user_admin`	[]	A list of administrator accounts, which can monitor all sessions and view/abort them. Must be a valid user account for the service (with the above authentication mechanisms).

Configuration: Custom authentication mechanism It is possible to define custom authentication mechanisms via a user function that should get `(user, pw, authenticated, session_ref)` as arguments (see below) and return a string starting by "SUCCESS" or "FAILED". The default return value should be the previous authenticator results. Any "SUCCESS" in the returned string fully qualifies the authentication.

In practice, define such a function in e.g. `/etc/qemu-web-desktop/config.pl`

(and better in `/etc/qemu-web-desktop/config.d/*.pl`) as for instance:

```
sub check_user_func {
  my $user          = shift;
  my $pw            = shift;
  my $authenticated = shift; # previous authenticator results
  my $session_ref   = shift;

  if (not $session_ref) { return $authenticated; }
  my %session     = %{ $session_ref };
  my $res           = "";

  # choose state depending on $user and $pw, as well as previous $authenticated
  res = "SUCCESS: [Custom] $user authenticated.";

  # or when authentication fails
  res = "FAILED: [Custom] $user failed authentication.";

  # or if we skip tests, we may return the previous authentication message
  res = $authenticated;

  return "$res";
}

$config{check_user_custom} = \&check_user_func;
```

or directly as an anonymous function

```
 $config{check_user_custom} = sub { ... };
```

Such a function could be used as an independent authenticator when other authenticators have failed.

But the custom function could also be used as a further check, validating a previously successful authentication, such as in:

```
sub check_user_func {
  my $user          = shift;
  my $pw            = shift;
  my $authenticated = shift; # previous authenticator results
  my $session_ref   = shift;

  if (not $session_ref) { return $authenticated; }
  my %session     = %{ $session_ref };
  my $res = "";

  # any previous authenticator was successful
  if (index($authenticated, "SUCCESS") >= 0) {
    # make further checks on user credentials
```

```
    if((length($pw)<8) ||
      ($pw !~ /[A-Z]/)  ||
      ($pw !~ /[0-9]/)  ||
      ($pw !~ /[a-z]/)  ||
      ($pw !~ /[@#*=&%><~_\$\-\+.,;:!\?]/)){
      $res .= "FAILED: $user, your password is not strong. It must be at least 8 characters
    }
  }
  # must check for a valid email
  if (not Email::Valid->address($session{user_email})) {
    $res .= "FAILED: $user, your email address $session{user_email} is not valid. You need a
  }

  # default: return failure or previous authenticator
  if ($res) { return $res; }
  return $authenticated;
}
$config{check_user_custom} = \&check_user_func;
```

In this example, additional requirements are set, checking password strength and email availability.

It is also possible for instance to use this custom section for: - Central Authentication Service(CAS) via the Perl library `libauthen-cas-client-perl` - OAuth2 via the library `liblwp-authen-oauth2-perl` - SAML2.

IMAP/SSL encryption level :warning: The SSL encryption level of the IMAP server (for user credentials) should match that of the server running the remote desktop service. The current Debian rules are to use SSL v1.2 or v1.3. In case the user IMAP authentication brings errors such as:

```
IMAP error "Unable to connect to <server>: SSL connect attempt failed error:1425F102:SSL rou
```

which appears in the Apache2 error log (`/var/log/apache2/error.log`), then you may downgrade the SSL encryption requirement in the file `/etc/ssl/openssl.cnf` in a section such as:

```
[system_default_sect]
MinProtocol = TLSv1
CipherString = DEFAULT@SECLEVEL=1
```

Configuration: Landing page and generated message layout

The default landing page is generated from the `index.html.tmpl` template which is located e.g. in `/etc/qemu-web-desktop`. You may copy and edit your own version of this template in `/etc/qemu-web-desktop/config.d`. This includes e.g.:

- changing the logo, welcome text, etc.
- relocate the session choice sections (cpu, memory, life=time, GPU, ...).

Most options can be controlled from the configuration files `config.pl` as well as `config.d/*.pl`. The following placeholders are replaced as follows:

- `{{layout_machines_list}}`: list of enabled VM from `*.ini` and `*.conf` files. Use `qwdctl enable VM`. To order the available VM's, number the `.ini` files, e.g. `01-Debian-Trixie.ini`.
- `{{layout_cpu}}`: values extracted in order from `$config{layout_cpu}` in configuration files (`*.pl`).
- `{{layout_memory}}`: values extracted in order from `$config{layout_memory}` in configuration files (`*.pl`), in GB.
- `{{layout_lifetime}}`: values extracted in order from `$config{{{layout_lifetime}}}` in configuration files (`*.pl`), in days. Use 0 for infinite.

The GPU and one-shot sections can be made active/inactive with:

- `$config{oneshot}`=0 or 1. One-shot sessions are *not* recommended.
- `$config{layout_gpu}`=0 or 1. Use `qwdctl enable|disable gpu` to attach GPU's to the QWD service.

The service generates dynamic web pages to display e.g. logging and monitoring information.

- `$config{layout_level}` can be set as, 0,1,2 to tune the output verbosity.
- `$config{layout_title}` sets the generated page title (e.g. the service name).
- `$config{layout_header}` is displayed at the top of generated pages. It can be specified as a file path (e.g. `/etc/qemu-web-desktop/config.d/header.html`), or a string with HTML format, or left empty for default (e.g. `<a href=service_url><img src=DARTS_logo></a>`).
- `$config{layout_footer}` is displayed at the bottom of generated pages. It can be specified as a file path (e.g. `/etc/qemu-web-desktop/config.d/footer.html`), or a string with HTML format, or left empty for default (e.g. `<small><a href=service_url></small>`).
- `$config{layout_resize}` specifies the behavior of the VNC display when resizing the browser. The choice 'remote' rescales the inner session display when the browser page is resized. The 'scale' choice leaves the session untouched at fixed ratio.

Configuration: Hiding the service port

The default URL to connect to the sessions pass through an internal 'websocket' port such as `:6080` as defined in the e.g. `/etc/qemu-web-desktop/config.pl` (`$config{service_port}=6080;`) configuration file. The `6080` port is itself a websocket pointing to the internal VNC port, e.g. `5901`

(`$config{service_port_vnc}`). A reverse proxy is then set to only show a normal HTTP/HTTPS connection (`$config{service_reverse_proxy}`).

If case you wish to change the default settings, edit the QWD Apache configuration (e.g. `/etc/apache2/conf-available/qemu-web-desktop.conf`). Check the following block after the existing `Location` and `Directory` blocks. It must NOT be in a VirtualHost directive, to apply globally.

```
# /etc/apache2/conf-available/qemu-web-desktop.conf
# ...
# redirect QWD port 6080 to /darts/
# Applied globally to all <VirtualHost>
  RewriteEngine On
  RewriteCond %{HTTP:Upgrade} =websocket [NC]
  RewriteRule      /darts wss://127.0.0.1:6080/ [P,L]

  ProxyPass        /darts  ws://127.0.0.1:6080/
  ProxyPassReverse /darts  ws://127.0.0.1:6080/
```

where the port must be adapted to that of `$config{service_port}` (here 6080, the default), and the path `/darts` must match `$config{service_reverse_proxy}`.

Adapt the QWD configuration with `sudo qwdctl edit config` to modify e.g. the `/etc/qemu-web-desktop/config.pl` file accordingly (or edit a `.pl` file in `config.d/`):

```
$config{service_port}             = 6080;     # must match 6080     in apache config.
$config{service_port_multiple}    = 0;
$config{service_reverse_proxy}    = "/darts"; # must match '/darts' in apache config.
```

Then activate the new configuration:

```
sudo a2enmod proxy proxy_http proxy_wstunnel rewrite
sudo systemctl restart apache2
```

The URL to connect to the VM is pure HTTP/HTTPS, without the need to open other ports for the outside world. - https://server/darts/vnc.html?resize=remote&autoconnect=true&path=?tok

:warning: NOTE: In case you wish to directly use the exotic port `$config{service_port}` without the reverse proxy, set `$config{service_reverse_proxy} = "";`. The VNC websocket will then serve all connections, and not be restricted to localhost. The resulting URL is then e.g.: - https://server:6080/vnc.html?resize=remote&autoconnect=true&path=?token=6IuLRDsA

Configuration: Distributing the sessions work-load (load-levelling, scale-up)

It is possible to distribute the sessions over a set of servers. Each server has its own settings (load, GPU, ...). When the current server is overloaded (number

of sessions, cpu, mem, GPU), the request is sent to other servers in the list, until one can provide the service. A session must still fit on a single server (it can not be split into parts on different servers).

The only requirements are:

- install DARTS/qemu-web-desktop on all servers.
- configure each server with their own settings.
- install the same virtual machines on all servers (must have same name).
- set `$config{fallback_servers}` to a comma-separated list of servers, e.g. `http://server1,server2,195.221.4.1`. Any of URL, server names and IP are allowed. The URL should be preferred as it indicates the protocol to use (http or https).
- make sure fallback servers are reachable. However when a server is down, it is ignored, so that the computing infrastructure can cope with failures.

The list of fallback servers may be the same for all servers in the farm, so that the workload is fully shared and distributed equally, whatever be the used entry point. You may favour one entry point, and distribute the load to other servers. But you may as well define specific fallback lists on various servers, to allow different entry points and workload distributions. For instance you may group servers providing GPU's and similar resources. You may as well redirect to shared servers when local private servers are filled, but not the other way round, to secure some resources. Only the `$config{fallback_servers}` list has to be specified to scale-up your infrastructure.

There is no need to install any other complex load-leveller system. The DARTS system is a decentralized cluster. Any of its elements is an entry point to the service, and distributes the load automatically when needed. Following the above procedure provides a very fast way to scale-up a compute infrastructure. Just install a new computer with DARTS/qemu-web-desktop, and add its name to the other nodes. It will immediately be callable.

List of all options

The following list is obtained with the `qwdctl start` and the `qemu-web-desktop.pl -h` commands. All options can be specified as input argument to `qwdctl start VM ...` and in e.g. the `/etc/qemu-web-desktop/config.pl` file (and in `config.d/*.pl`).

```
cgi-bin/qemu-web-desktop.pl: launch a QEMU/KVM machine in a browser window. Version 26.03.19

Usage: cgi-bin/qemu-web-desktop.pl --option1=value1 ...

Valid options are:
  --boot_delay=VALUE [5]
  --certificate_crt=VALUE []
```

```
--certificate_key=VALUE []
--check_user_custom=VALUE []
--check_user_mfa=VALUE [0]
--check_user_with_email=VALUE [0]
--check_user_with_imap=VALUE [0]
--check_user_with_ldap=VALUE [0]
--check_user_with_local=VALUE [0]
--check_user_with_smtp=VALUE [0]
--config_script=VALUE [ARRAY(0x561a23660d08)]
--dir_cfg=VALUE [/tmp]
--dir_html=VALUE [/var/lib/qemu-web-desktop/html]
--dir_machines=VALUE [/var/lib/qemu-web-desktop/machines]
--dir_mounts=VALUE [ARRAY(0x561a2539a960)]
--dir_novnc=VALUE [/usr/share/novnc]
--dir_service=VALUE [/var/lib/qemu-web-desktop]
--dir_snapshots=VALUE [/var/lib/qemu-web-desktop/snapshots]
--dir_websockify=VALUE [websockify]
--email_from=VALUE []
--email_method=VALUE [auto]
--email_passwd=VALUE []
--fallback_servers=VALUE []
--gpu_blacklist=VALUE []
--gpu_model=VALUE [ARRAY(0x561a25387d78)]
--gpu_name=VALUE [ARRAY(0x561a24eee420)]
--gpu_pci=VALUE [ARRAY(0x561a236a5b98)]
--imap_port=VALUE [993]
--imap_server=VALUE []
--layout_footer=VALUE []
--layout_header=VALUE []
--layout_level=VALUE [1]
--layout_resize=VALUE [remote]
--layout_title=VALUE [Data Analysis Remote Treatment Service]
--ldap_domain=VALUE [EXP]
--ldap_port=VALUE [389]
--ldap_server=VALUE []
--machine=VALUE [-]
--oneshot=VALUE [0]
--qemu_exec=VALUE [auto]
--qemu_exec_options=VALUE [auto]
--service=VALUE [qemu-web-desktop]
--service_logfile=VALUE [/var/log/qemu-web-desktop.log]
--service_max_cpu_fraction_nb_per_user=VALUE [0.3]
--service_max_gpu_nb_per_user=VALUE [1]
--service_max_load=VALUE [0.8]
--service_max_mem_fraction_nb_per_user=VALUE [0.3]
--service_max_script_length=VALUE [65535]
```

```
--service_max_session_nb=VALUE [10]
--service_max_session_nb_per_user=VALUE [3]
--service_monitor=VALUE [0]
--service_port=VALUE [6080]
--service_port_multiple=VALUE [0]
--service_port_vnc=VALUE [5901]
--service_proxy=VALUE []
--service_purge=VALUE [0]
--service_reverse_proxy=VALUE [/darts]
--session_stop=VALUE []
--smtp_port=VALUE [587]
--smtp_server=VALUE []
--smtp_use_ssl=VALUE [starttls]
--snapshot_alloc_cpu=VALUE [1]
--snapshot_alloc_disk=VALUE [30]
--snapshot_alloc_mem=VALUE [4]
--snapshot_lifetime=VALUE [345600]
--snapshot_use_master=VALUE [0]
--user_admin=VALUE [ARRAY(0x561a236619f8)]
--version=VALUE [26.03.19]
```

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.

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, boot the VM manually, and install therein the `qemu-guest-agent` and the `spice-vdagent` packages. These packages also bring the shared clipboard for copy-paste between the host and the sessions.

Issues: Machine list does not appear in service page

You should refresh the landing page.

Issues: STOP and +24h buttons fail or uncrypted data warnings

If you get strange behaviours with the 'STOP' and '+24h' buttons ("ERROR: Can not find virtual machine") or warnings claiming that the data will be sent uncrypted, this is probably an encryption issue with SSL certificate mismatch.

Check that: - Apache configuration `SSLCertificateFile` should match QWD `$config{certificate_crt}` - Apache configuration `SSLCertificateKeyFile` should match QWD `$config{certificate_key}`

Issues: System specificity

The Arch-type and Fedora-type systems only support the `$config{service_port_multiple} = 1`. This means a different port number is requested for each new session. To secure the sessions, a VNC connection password is requested, which is indicated in the login page.

:warning: Currently, the web-service **fails under RedHat/Fedora**, but the `qwdctl start VM` properly works.