

Qubes Salt Beginner's Guide

[salt](#), [customization](#), [administration](#), [configuration](#), [backup](#)

[leo](#) 1 July 31, 2023, 8:55pm

Qubes Salt Beginner's Guide

Part 1: Creating our first qubes

As a beginner, [Salt](#) seemed daunting to me at first. It took some effort to learn but it was worth it! I'm writing this guide for beginners who enjoy an hands-on introduction with examples.

1.1 Creating personal state configuration directories

Our journey starts with a file found in the base Salt configuration directory in dom0: `/srv/salt/qubes/README.rst` ([GitHub link](#)). In this file we can read:

qubes.user-dirs

Install and maintain user salt and pillar directories for personal state configurations:

```
/srv/user_salt  
/srv/user_pillar
```

User defined scripts will not be removed on removal of qubes-mgmt-salt by design nor will they be modified on any updates, other than permissions being enforced.

We can activate `qubes.user-dirs` to create personal state configuration directories. What is `qubes.user-dirs`, and how do we activate it? This is what we call a *state configuration*. It is a configuration file that tells Salt what to do to reach a particular state.

To activate `qubes.user-dirs`, we can follow the instructions found in its configuration file, `/srv/salt/qubes/user-dirs.sls` ([GitHub link](#)):

qubes.user-dirs

=====

Install and maintain user salt and pillar directories for personal state configurations:

Includes a simple locale state file

User defined scripts will not be removed on removal of qubes-mgmt-salt by design nor will they be modified on any updates, other than permissions being enforced.

Execute:

```
qubesctl state.sls qubes.user-dirs
```

We run the command `sudo qubesctl state.sls qubes.user-dirs`. Salt applies the corresponding state, and tells us that some files and directories were created. Among these directories we can find `/srv/user_salt/`: this is the main directory where we'll place our own state configuration files.

1.2 The top file and running highstate

Running the state `qubes.user-dirs` also created the file `/srv/user_salt/top.sls`. Here is what this file looks like before we modify it ([GitHub link](#)):

```
# vim: set syntax=yaml ts=2 sw=2 sts=2 et :
#
# 1) Intial Setup: sync any modules, etc
# --> qubesctl saltutil.sync_all
#
# 2) Initial Key Import:
# --> qubesctl state.sls salt.gnupg
#
# 3) Highstate will execute all states
# --> qubesctl state.highstate
#
# 4) Highstate test mode only. Note note all states seem to conform to test
# mode and may apply state anyway. Needs more testing to confirm or not
# --> qubesctl state.highstate test=True

# === User Defined Salt States =====
#user:
#  '*':
#    - locale
```

This file is called the *top file*.

In the future, when we have many state configuration files, it will become quite tedious to run each state one by one with the command `sudo qubesctl state.sls my-custom-state`. The top file solves that. If we write in this file how to run each state, we get the ability to run all of them with a single command: `sudo qubesctl state.highstate`. We call this “running highstate”.

1.3 Targeting qubes

There are three lines that are commented out at the end of the top file `/srv/user_salt/top.sls`:

```
user:
  '*':
    - locale
```

If we were to uncomment those lines and run highstate, Salt would run in *all* targeted qubes (this is what is meant by the `*` character) the state `locale`, for which the state configuration file can

either be `/srv/user_salt/locale.sls` or `/srv/user_salt/locale/init.sls`.

How do we target a qube? By default, the commands `qubesctl state.sls my-custom-state` and `qubesctl state.highstate` only target dom0. To make Salt target additional qubes, we can give their names to the `--targets` argument:

- `sudo qubesctl --targets=fedora-38 state.sls my-custom-state` will run `my-custom-state` targeting dom0 and fedora-38.
- `sudo qubesctl --skip-dom0 --targets=debian-12,untrusted state.highstate` will run highstate targeting the qubes debian-12 and untrusted but not dom0.

1.4. Creating a qube with Salt

We have a [template](#) called fedora-38. We would like Salt to create a purple qube named “salty” based on this template. We write the state configuration file `/srv/user_salt/salty.sls` as follows:

```
salty--create-qube:
  qvm.vm:
    - name: salty
    - present:
      - template: fedora-38
      - label: purple
    - prefs:
      - label: purple
```

That’s it! Running `sudo qubesctl state.sls salty saltenv=user` will make Salt create a purple qube named salty. If salty is already present, Salt will just make sure it’s purple but won’t do anything else.

▼ Note on "saltenv=user"

Note that we always need to add the extra argument `saltenv=user` to the command `sudo qubesctl state.sls my-custom-state` when we run individual states from the user directory `/srv/user_salt/`.

To make things easier, we would like to automatically run this state when we run highstate. We add the following to the top file `/srv/user_salt/top.sls`:

```
user:
  dom0:
    - salty
```

Great! Now, the command `sudo qubesctl state.highstate` will automatically create salty.

1.5 Creating a disconnected qube

We have a template called debian-11. We would like Salt to create a green qube named “disconnected” based on this template, but that has no web browser and no internet access. We write the state configuration file `/srv/user_salt/disconnected.sls` as follows:

```
disconnected--create-qube:
  qvm.vm:
```

```
- name: disconnected
- present:
  - template: debian-11
  - label: green
- prefs:
  - label: green
  - netvm: none
- features:
  - set:
    - menu-items: org.gnome.Terminal.desktop org.gnome.Nautilus.desktop
```

Perfect! We can now make Salt create this qube with the command `sudo qubesctl state.sls disconnected saltenv=user`.

To make things easier, we would like to automatically run this state when we run highstate. We add the following lines to the top file `/srv/user_salt/top.sls`:

```
user:
  dom0:
    - disconnected
```

Great! Now, the command `sudo qubesctl state.highstate` will automatically create our disconnected qube.

▼ Tip: How to make Salt create both "salty" and "disconnected" when we run highstate?

We can write the top file `/srv/user_salt/top.sls` as follows:

```
user:
  dom0:
    - salty
    - disconnected
```

1.6 Useful links

I hope this was clear. Here are some links if you'd like to go further:

- [Qubes docs on Salt](#)
- [Qubes configuration management](#), extensive docs by [@gonzalo-bulnes](#)
- [Salt user guide](#), a complete overview of Salt targeted at beginners
- [Salt docs](#), great when looking for something specific but intimidating to start with
- [@unman's notes](#), a good reference on Salt written by a developer of Qubes OS
- [qvm commands](#), list of all qubes-specific salt commands and what they do
- [Qubes state configuration files](#) used to create the first qubes when installing Qubes OS

The next part of this guide will be about creating new templates and installing packages in them.

Part 2: Apps and templates

In this part we'll learn how use Salt to make qubes with [new software](#) (including apps that are not in the official repositories!), and create new templates.

2.1 Activating pre-installed apps

We have a [template](#) called `debian-12`. We would like Salt to create a “vault” qube based on `debian-12` that is never connected to the internet, and that we will only use for the app `KeepassXC`.

Luckily, `KeepassXC` comes pre-installed in the template `debian-12`, so we can simply tell Salt to make it available in the app menu. We write our state configuration file `/srv/user_salt/vault.sls` as follows:

```
vault--create-qube:
  qvm.vm:
    - name: vault
    - present:
      - template: debian-12
      - label: black
    - prefs:
      - label: black
      - netvm: none
    - features:
      - set:
        - menu-items: org.KeepassXC.KeePassXC.desktop org.gnome.Terminal.desktop
```

As a result, running `sudo qubesctl state.sls vault saltenv=user` will make Salt create the vault qube if it’s not there, and make sure that it has `KeePassXC` in its app menu.

To make things easier, we would like Salt to automatically take care of the vault when we run `highstate`. We write the following in the top file `/srv/user_salt/top.sls`:

```
user:
  dom0:
    - vault
```

The command `sudo qubesctl state.highstate` will now automatically run the “vault” state.

2.2 Installing new apps

We would like to have a “messaging” qube for communicating with our friends through an app called `Telegram`. However, `Telegram` is not part of the `debian-12` template, so we’ll have to install it.

Luckily, `Telegram` is available in the official repository. We can therefore tell Salt to create the “messaging” qube and make sure that `Telegram` is installed in the `debian-12` template by writing the state configuration file `/srv/user_salt/messaging.sls` as follows:

```
{% if grains['id'] == 'dom0' %}

messaging--create-qube:
  qvm.vm:
    - name: messaging
    - present:
      - template: debian-12
      - label: yellow
    - prefs:
      - label: yellow
```

```

- features:
- set:
  - menu-items: org.telegram.desktop.desktop org.gnome.Nautilus.desktop

{% elif grains['id'] == 'debian-12' %}

messaging--install-apps-in-template:
  pkg.installed:
    - pkgs:
      - telegram-desktop

{% endif %}

```

There you go! As we need to run the install process in the debian-12 template, we have to [target](#) debian-12 when we make Salt execute this state: `sudo qubesctl --targets=debian-12 --show-output state.sls messaging saltenv=user`.

▼ Note on the "{% ... %}" syntax

This state configuration file has two parts. In the first part, we wrote the instructions that Salt has to execute while running in the admin qube dom0, while the second part is about installing Telegram, which must be executed in the template debian-12. To have everything in the same file but ensure that the right part gets executed in the right qube, we decided to use a Jinja [“if statement”](#) to modify the state configuration file depending on what qube Salt is currently running the instructions for.

Similarly as in the previous section, we can have Salt apply this state while targeting both dom0 and debian-12 when running highstate. We add the following to the top file `/srv/user_salt/top.sls`:

```

user:
  dom0 or debian-12:
    - messaging

```

This makes the command `sudo qubesctl --targets=debian-12 --show-output state.highstate` automatically create a messaging qube with Telegram as part of its app menu.

2.3 Creating a “non-free” template

We would like to create a “conferencing” qube with the software Skype to communicate with our family. Skype, however, is not available from the official debian-12 repository because it is distributed under a proprietary software licence: we will have to add an external repository to be able to install it.

As Skype is not in the official repository, we consider that there is a non-zero risk that it compromises the security of the template during its installation process. Because we want to [trust our default templates](#), we decide to create a new “nonfree” template to install this proprietary software.

We write our state configuration file `/srv/user_salt/conferencing.sls` as follows:

```

{% if grains['id'] == 'dom0' %}

```

```

conferencing--create-nonfree-template:
  qvm.clone:
    - name: nonfree
    - source: debian-12

conferencing--create-app-qube:
  qvm.vm:
    - name: conferencing
    - present:
      - template: nonfree
      - label: yellow
    - prefs:
      - label: yellow
    - features:
      - set:
        - menu-items: skypeforlinux.desktop org.gnome.Nautilus.desktop
    - require:
      - qvm: conferencing--create-nonfree-template

{% elif grains['id'] == 'nonfree' %}

conferencing--download-key:
  cmd.run:

```

Running this state makes Salt create a “conferencing” app qube based on a new template called “nonfree”, in which Salt makes sure that Skype is installed from its external repository. To run this state, we target our new “nonfree” template with the command:

```
sudo qubesctl --targets=nonfree --show-output state.sls conferencing saltenv=
```

Let’s add this state to the top file, so that it is applied automatically when we run highstate! At the end of the top file `/srv/user_salt/top.sls`, we write:

```

user:
  dom0 or nonfree:
    - conferencing

```

We can now run highstate with the command:

```
sudo qubesctl --targets=nonfree --show-output state.highstate
```

▼ Tip: Here is what our top file would look like if we would include all the states from this guide so far.

```

user:
  dom0:
    - salty
    - disconnected
    - vault
  dom0 or debian-12:
    - messaging

```

```
dom0 or nonfree:
- conferencing
```

With this top file, we would run highstate with:

```
sudo qubesctl --targets=debian-12,nonfree --show-output state.highstate
```

We could also directly target all our templates with:

```
sudo qubesctl --templates --show-output state.highstate
```

2.4 Useful links

I tried to be as concise as I could. Please let me know if you have any further questions! Here are some related links.

- [Qubes docs on installing software](#)
- [Qubes docs on trusting your templates](#)
- [Basic Salt syntax](#), an extremely useful primer from the official Salt documentation
- [extrepo-data](#), a list of external repositories for many apps that are not in the official repositories

In the final part of this guide, we will learn how to make Salt perform automated backups of our qubes with [Wyng](#).

Part 3: Backing up Qubes

One of the biggest advantages of using Salt with Qubes OS is that we would just need to copy and run our state configuration files to completely recreate our system. Saving those files would however not be sufficient for a backup, because that would not contain any our personal data.

We could use the official [Qubes Backup](#) tool to create full backups of our qubes, but [as of now](#) this tool does not support creating [incremental backups](#), which make the backup process much more performant.

In this part of the guide, we are going to create Salt states that can make fast incremental backups of our qubes.

3.1 Downloading Wyng

[Wyng](#) is a backup software written by [@tasket](#) that we can use to make incremental backups of the contents of our qubes in a very efficient manner. Its project page gives a detailed explanation of how it works.

It should be noted that a tool called [wyng-util-qubes](#) is also currently being developed and makes it very easy to use Wyng with Qubes OS. One of the advantages of using this tool is that, on top of backing up the contents of our qubes with Wyng, this tool also backs up their configuration. At the time of writing, however, wyng-util-qubes does not yet support installations of Qubes OS with a BTRFS partitioning scheme. For this reason, and because the configuration of our qubes is already saved in our Salt state configuration files, we are going to use Wyng directly instead of wyng-util-qubes.

To use Wyng, we first have to copy it to dom0. To do so, can we download and extract the main branch of its repository in a trusted qube called “disp8265” with the command:


```
curl --location https://github.com/tasket/wyng-backup/archive/refs/heads/main
```

▼ Note on the version of Wyng used in this guide.

The above command downloads the latest recommended version of Wyng, which is Wing v0.8beta at the time of writing. It might be that some parts of this guide stop working with a more recent version of Wyng, in which case the guide will have to be updated.

The entire Wyng program is contained in the file `wyng-backup-main/src/wyng`, which is in the directory that we just created. We can follow the instructions shown on [the project page](#) to verify the authenticity of this file. Once we trust this file, we can [copy it to dom0](#) by opening a dom0 terminal and running:

▼ Make sure to understand the risks of copying files to dom0 before executing this command.

```
qvm-run --pass-io disp8265 'cat wyng-backup-main/src/wyng' > wyng
```

The file `wyng` is now in our home directory in dom0. We could mark this file as executable with the command `chmod +x wyng` and run Wyng directly from the command line, but we won't: we are going to use Salt! But first, we have to create a BTRFS subvolume.

3.2 Creating a BTRFS subvolume

When we installed Qubes OS on our machine, we decided to use BTRFS instead of the default partition scheme. To be able to use Wyng with BTRFS, the files that contain the filesystems of our qubes must be located inside of a BTRFS subvolume. This is not the case by default: the files that contain the filesystems of our qubes are located under `/var/lib/qubes`, which is not a BTRFS subvolume but a regular directory.

We use the following commands to create a BTRFS subvolume:

```
qvm-shutdown --all --wait --force
sudo mv /var/lib/qubes /var/lib/qubes-old
sudo btrfs subvolume create /var/lib/qubes
shopt -s dotglob
sudo mv /var/lib/qubes-old/* /var/lib/qubes
sudo rmdir /var/lib/qubes-old
```

▼ Tip: There is a longer and riskier method that does not involve moving files across subvolumes.

1. Shut down all qubes with `qvm-shutdown --all --wait --force`
2. Rename `/var/lib/qubes` to `/var/lib/qubes-old`
3. Create a snapshot of the root subvolume at the location `/var/lib/qubes`:

```
sudo btrfs subvolume snapshot / /var/lib/qubes
```

4. Delete everything in the new subvolume `/var/lib/qubes` except the directory `/var/lib/qubes/var/lib/qubes-old`
5. Move the contents of `/var/lib/qubes/var/lib/qubes-old` to `/var/lib/qubes` (don't forget the hidden files)
6. Delete the now empty directory `/var/lib/qubes/var/lib/qubes-old` and its empty parents

Once this is done, the output of the command `sudo btrfs subvolume list /` should contain a line that ends with `/var/lib/qubes`, and our system should function normally.

3.3 Making local backups

We would like to configure Wyng to create encrypted incremental backups that we will save on our machine in a [service qube](#) called “sys-backup”. The idea is that we’ll back up our qubes regularly and, from time to time, we’ll copy the backup archive in sys-backup to some external storage, or upload it to a remote server.

We create the following files:

▼ `/srv/user_salt/backup/map.jinja`

```
{# This file holds the backup passphrase and general configuration #}

{# The passphrase must not contain any single quote character (') #}

{% set backup = {
    'passphrase': 'my-backup-passphrase',
    'qubes': ['vault', 'messaging'],
    'volumes': ['appvms/vault/private.img', 'appvms/messaging/private.img'],
    'source': '/var/lib/qubes',
    'destination': '/home/user/qubes.backup',
} %}
```

▼ `/srv/user_salt/backup/init.sls`

```
# Install Wyng and create the service qube "sys-backup" for storing backups

{% from 'backup/map.jinja' import backup %}

backup--install-dependencies:
  pkg.installed:
    - pkgs:
      - python3-pycryptodomex
      - python3-zstd

backup--install-wyng:
  file.managed:
    - names:
      - /usr/local/bin/wyng:
        - source: salt://backup/files/wyng
        - mode: 755
      - /etc/wyng/wyng.ini:
        - source: salt://backup/files/wyng.ini
        - template: jinja
        - context: {{ backup|yaml }}
        - makedirs: True
    - require:
      - pkg: backup--install-dependencies
```

```
backup--create-service-qube:
```

▼ /srv/user_salt/backup/configure.sls

```
# Create a backup archive and add the backed-up qubes to its configuration

{% from 'backup/map.jinja' import backup %}

backup.configure--create-archive:
  cmd.run:
    - name: |
        echo '{{ backup.passphrase }}' | wyng arch-init --unattended
    - unless:
        - qvm-run sys-backup test -d {{ backup.destination }}

backup.configure--add-volumes:
  cmd.run:
    - name: |
        echo '{{ backup.passphrase }}' | wyng add {{ backup.volumes|join(' ') }}
    - require:
        - cmd: backup.configure--create-archive
```

▼ /srv/user_salt/backup/clear-cache.sls

```
# Remove the directory /home/user/.cache in each of the backed-up qubes

{% from 'backup/map.jinja' import backup %}

backup.clear-cache--shutdown-app-qubes:
  qvm.shutdown:
    - names: {{ backup.qubes|yaml }}

backup.clear-cache--clear-cache:
  qvm.vm:
    - names: {{ backup.qubes|yaml }}
    - actions:
        - run
        - shutdown
    - run:
        - cmd: rm --recursive --force /home/user/.cache
    - shutdown: []
    - require:
        - qvm: backup.clear-cache--shutdown-app-qubes
```

▼ /srv/user_salt/backup/send.sls

```
# Create a backup and copy it to the archive in "sys-backup"

{% from 'backup/map.jinja' import backup %}

include:
  - backup.configure
```

```

- backup.clear-cache

backup.send--send-backup:
  cmd.run:
    - name: |
        echo '{{ backup.passphrase }}' | wyng send --all --unattended
    - require:
      - cmd: backup.configure--add-volumes
      - qvm: backup.clear-cache--clear-cache

backup.send--shutdown-service-qube:
  qvm.shutdown:
    - name: sys-backup
    - require:
      - cmd: backup.send--send-backup

```

▼ /srv/user_salt/backup/receive.sls

```

# Overwrite current data with the most recent backup from the archive

{% from 'backup/map.jinja' import backup %}

include:
  - backup.configure

backup.receive--shutdown-app-qubes:
  qvm.shutdown:
    - names: {{ backup.qubes|yaml }}
    - require:
      - cmd: backup.configure--add-volumes

backup.receive--receive-backup:
  cmd.run:
    - name: |
        echo '{{ backup.passphrase }}' | wyng receive --all --unattended
    - require:
      - qvm: backup.receive--shutdown-app-qubes

backup.receive--shutdown-service-qube:
  qvm.shutdown:
    - name: sys-backup
    - require:
      - cmd: backup.receive--receive-backup

```

▼ /srv/user_salt/backup/files/wyng

This is Wyng's executable file which we copied to dom0 in [section 3.1](#).

▼ /srv/user_salt/backup/files/wyng.ini

```
{#- This file configures the default options for Wyng commands -#}
```

```
[var-global-default]
local = {{ source }}
dest = qubes://sys-backup{{ destination }}
```

This gives us a general configuration file `map.jinja` where we can enter our password and the list of qubes that we want to include in our backups, as well as the following Salt states:

- The initial state `init.sls` makes sure that Wyng and the python libraries that are required for encryption and compression are installed in dom0, and creates the “sys-backup” service qube. **We must run this state at least once before using the other states.** We can run this state with:

```
sudo qubesctl state.sls backup saltenv=user
```

- The state `configure.sls` creates the backup archive in sys-backup if it doesn't exist, and ensures that all the backed-up qubes specified in the configuration file `map.jinja` are in the archive configuration. This state is automatically included in other states that require it, but we can also run it manually with:

```
sudo qubesctl state.sls backup.configure saltenv=user
```

- The state `clear-cache.sls` shuts down all the backed-up qubes to make sure that there have no program running, then starts them one by one and removes their cache directory `/home/user/.cache` before shutting them down once again. Because it is important to clear the cache before making backups, this step is automatically included as part of the backup process when using the state `send.sls` to make backups. Nevertheless, if needed we can run this state manually with the command:

```
sudo qubesctl state.sls backup.clear-cache saltenv=user
```

- The state `send.sls` first applies the state `configure.sls`, then clears the cache of all backed-up qubes by applying the state `clear-cache.sls`, and finally creates a new backup in the archive in “sys-backup”. **This is the state that we'll run the most.** We can run it with:

```
sudo qubesctl state.sls backup.send saltenv=user
```

- The state `receive.sls` automatically applies the state `configure.sls`, then shuts down all the backed-up qubes and overwrites their data with the data contained in the latest backup found in the archive. It can be run with:

```
sudo qubesctl state.sls backup.receive saltenv=user
```

▼ Tip: For very long operations, we can run Wyng manually to see progress in real-time.

Making backups manually requires performing more actions than simply applying the `send.sls` state file, but it is useful because it shows the output of Wyng in real time during the backup process. It can be done by running the following command (preceded by `sudo qubesctl state.sls backup.configure saltenv=user` in case the backup archive is not yet configured):

```
sudo qubesctl state.sls backup.clear-cache saltenv=user && sudo wyng send --a
```

It is also possible to restore a backup manually by running the following command after shutting down all the backed-up qubes:

```
sudo wyng receive --all
```

3.4 Making remote backups

We would like to configure Wyng to create encrypted incremental backups that we will directly upload to a remote [Nextcloud Server](#). To do so, we will use [rclone](#) to mount the cloud storage in a [service qube](#) called “sys-backup”.

This method is not recommended for backing up large qubes, because each backup has to be copied to a [virtual file system](#) in the “sys-backup” qube before it can be uploaded to the remote. This means that the “sys-backup” qube must be large enough to hold an entire incremental backup, which may require too much disk space.

We create the following files:

▼ /srv/user_salt/backup/map.jinja

```
{# This file holds the backup passphrase and general configuration #}

{# The passphrase must not contain any single quote character (') #}

{% import 'templates.jinja' as templates %}

{% set backup = {
    'passphrase': 'my-backup-passphrase',
    'qubes': ['archive', 'personal', 'vault'],
    'volumes': ['appvms/archive/private.img', 'appvms/personal/private.img',
    'source': '/var/lib/qubes',
    'directory': 'qubes.backup',
    'remote': ':webdav,url="https://cloud.example.com/remote.php/webdav",vend
    'domain': 'cloud.example.com',
    'timeout': 3600,
} %}
```

▼ /srv/user_salt/backup/init.sls

```
# Install Wyng and rclone and create the service qube "sys-backup"

{% from 'backup/map.jinja' import backup %}

{% if grains['id'] == 'dom0' %}

backup--install-dependencies:
    pkg.installed:
        - pkgs:
            - python3-pycryptodomex
            - python3-zstd
```

```

backup--install-wyng:
  file.managed:
    - names:
      - /usr/local/bin/wyng:
          - source: salt://backup/files/wyng
          - mode: 755
      - /etc/wyng/wyng.ini:
          - source: salt://backup/files/wyng.ini
          - template: jinja
          - context: {{ backup|yaml }}
          - makedirs: True
    - require:
      - pkg: backup--install-dependencies

```

▼ /srv/user_salt/backup/configure.sls

```

# Create a backup archive and add the backed-up qubes to its configuration

{% from 'backup/map.jinja' import backup %}

backup.configure--create-mount-point:
  cmd.run:
    - name: qvm-run sys-backup 'sudo mkdir /mnt/remote && sudo chown user:use

backup.configure--mount-remote:
  cmd.run:
    - name: |
        qvm-run --quiet sys-backup rclone mount --rc --vfs-cache-mode=writes
    - bg: True
    - require:
      - cmd: backup.configure--create-mount-point

backup.configure--wait-mounted:
  loop.until_no_eval:
    - name: cmd.run
    - expected: 'true'
    - args:
      - qvm-run --pass-io sys-backup 'rclone rc vfs/list | jq ".vfses | lengt
    - require:
      - cmd: backup.configure--mount-remote

backup.configure--create-archive:

```

▼ /srv/user_salt/backup/clear-cache.sls

This is identical to the homonymous state in [section 3.3](#).

▼ /srv/user_salt/backup/send.sls

```

# Create a backup and upload it to the archive through "sys-backup"

{% from 'backup/map.jinja' import backup %}

```

```

include:
  - backup.configure
  - backup.clear-cache

backup.send--send-backup:
  cmd.run:
    - name: |
        echo '{{ backup.passphrase }}' | wyng send --all --unattended
    - require:
        - loop: backup.configure--wait-volumes-sync
        - qvm: backup.clear-cache--clear-cache

backup.send--wait-backup-sync:
  loop.until_no_eval:
    - name: cmd.run
    - expected: 'true'
    - timeout: {{ backup.timeout }}
    - args:
        - qvm-run --pass-io sys-backup 'rclone rc vfs/stats | jq "all(.diskCach
    - require:
        - cmd: backup.send--send-backup

```

▼ /srv/user_salt/backup/receive.sls

```

# Overwrite current data with the most recent backup from the archive

{% from 'backup/map.jinja' import backup %}

include:
  - backup.configure

backup.receive--shutdown-app-qubes:
  qvm.shutdown:
    - names: {{ backup.qubes|yaml }}
    - require:
        - loop: backup.configure--wait-volumes-sync

backup.receive--receive-backup:
  cmd.run:
    - name: |
        echo '{{ backup.passphrase }}' | wyng receive --all --unattended
    - require:
        - qvm: backup.receive--shutdown-app-qubes

backup.receive--wait-backup-sync:
  loop.until_no_eval:
    - name: cmd.run
    - expected: 'true'
    - timeout: {{ backup.timeout }}
    - args:

```


▼ /srv/user_salt/backup/files/wyng

This is Wyng’s executable file which we copied to dom0 in [section 3.1](#).

▼ /srv/user_salt/backup/files/wyng.ini

```
{#- This file configures the default options for Wyng commands -#}  
  
[var-global-default]  
local = {{ source }}  
dest = qubes://sys-backup/mnt/remote/{{ directory }}
```

▼ /srv/user_salt/backup/files/firewall.xml

```
{#- This file sets up basic firewall rules for "sys-backup" -#}  
  
<firewall version="2">  
  <rules>  
    <rule>  
      <properties>  
        <property name="action">accept</property>  
        <property name="dsthost">{{ domain }}</property>  
      </properties>  
    </rule>  
    <rule>  
      <properties>  
        <property name="action">accept</property>  
        <property name="specialtarget">dns</property>  
      </properties>  
    </rule>  
    <rule>  
      <properties>  
        <property name="action">accept</property>  
        <property name="proto">icmp</property>  
      </properties>  
    </rule>  
    <rule>  
      <properties>  
        <property name="action">drop</property>  
      </properties>  
    </rule>  
  </rules>  
</firewall>
```

This gives us a general configuration file `map.jinja` where we can enter our encryption password as well as the list of qubes that we want to include in our backups, the rclone remote specification as a [connection string](#), and how much time we are willing to wait for each upload operation (in seconds). We also wrote the following Salt states:

- The initial state `init.sls` makes sure that Wyng and the python libraries that are required for encryption and compression are installed in dom0, and installs rclone in the template “debian-12”. It also creates the “sys-backup” service qube, and adds a firewall rule to prevent this qube from connecting to any IP address that does not correspond to the domain of the cloud storage. **We must run this state at least once before using the other states.** We can run this state with:

```
sudo qubesctl --show-output --targets=debian-12 state.sls backup saltenv=us
```

- The state `configure.sls` mounts the cloud storage in “sys-backup”, creates the backup archive if it doesn’t exist, and ensures that all the backed-up qubes specified in the configuration file `map.jinja` are in the archive configuration. This state is automatically included in other states that require it, but we can also run it manually with:

```
sudo qubesctl state.sls backup.configure saltenv=user
```

- The state `clear-cache.sls` shuts down all the backed-up qubes to make sure that there have no program running, then starts them one by one and removes their cache directory `/home/user/.cache` before shutting them down once again. Because it is important to clear the cache before making backups, this step is automatically included as part of the backup process when using the state `send.sls` to make backups. Nevertheless, if needed we can run this state manually with the command:

```
sudo qubesctl state.sls backup.clear-cache saltenv=user
```

- The state `send.sls` first applies the state `configure.sls`, then clears the cache of all backed-up qubes by applying the state `clear-cache.sls`, and finally uploads a new backup to the archive through “sys-backup”. **This is the state that we’ll run the most.** We can run it with:

```
sudo qubesctl state.sls backup.send saltenv=user
```

- The state `receive.sls` automatically applies the state `configure.sls`, then shuts down all the backed-up qubes and overwrites their data with the data downloaded from the latest backup found in the archive. It can be run with:

```
sudo qubesctl state.sls backup.receive saltenv=user
```

▼ Tip: For very long operations, we can run Wyng manually to see progress in real-time.

Making backups manually requires performing more actions than simply applying the `send.sls` state file, but it is useful because it shows the output of Wyng in real time during the backup process. It can be done by running the following command (preceded by `sudo qubesctl state.sls backup.configure saltenv=user` in case the backup archive is not yet configured):

```
sudo qubesctl state.sls backup.clear-cache saltenv=user && sudo wyng send --a
```

We must then run the following command periodically to see how many uploads are still in progress/queued:

```
qvm-run --pass-io sys-backup rclone rc vfs/stats
```

The backup is complete when the properties “uploadsQueued” and “uploadsInProgress” returned by this command are both equal to zero.

It is also possible to restore a backup manually by running the following command after shutting down all the backed-up qubes:

```
sudo wyng receive --all
```

3.5 Useful links

- [Wyng repository](#), it gives detailed explanations for each command and their arguments
- [wyng-util-qubes](#), a tool that makes it easy to use Wyng with Qubes OS
- [docs for rclone](#), a tool to manage files in the cloud supporting many storage providers
- [Jinja templating docs](#), all the syntax and semantics of the Jinja templating engine
- [Salt docs on Jinja](#), they include descriptions of all the built-in filters such as `regex_replace`

It took me quite a while to write this final part of the guide but I'm quite happy with the result. I tried to write it in a way that is easy to adapt to another Qubes install. Of course, feel free to ask if something needs clarification or doesn't work. Have a nice day.

Appendix: Salt resources made by Qubes users

Guides:

- [Qubes configuration management](#), extensive docs by [@gonzalo-bulnes](#)
- [@unman's notes](#), a good reference on Salt written by a developer of Qubes OS

Formulas:

- [Shaker](#) by [@unman](#)
- [Salt & Pepper](#), drkhsh's configuration for automated template deployments
- [qubes-salt-examples](#) by [@zaz](#) and [@kenosen](#)
- [Qubes SaltStack configuration of Videos Playback VM \(didactic\)](#) by [@bruno_schroeder](#)
- [qusal: Salt Formulas for Qubes OS](#) by [@ben-grande](#)
- [qubes-for-journalists](#), a simple journalist-oriented salt repo and wiki by [@kenosen](#)

Videos:

- [How to architect your Qubes OS with SaltStack](#) by [@ben-grande](#)

42 Likes

[Incremental backup possibilities](#)

[Odd errors while starting out with salt](#)

[Are there release notes for dom0 updates?](#)

[ProtonVPN App 4.2 setup guide](#)

[Questions about Wyng](#)

[VMs snapshots for revert in time](#)

[Salt: automating NVIDIA GPU passthrough: fedora 41](#)

[Wrapping my head around salt](#)

[Audio qube and Salt](#)

[Incremental backup possibilities](#)

[ANN: Wyng incremental backup, new version!](#)

[Audio qube](#)

[Persistant application changes within template across all vm types?](#)

[leo](#) 2 August 4, 2023, 11:31pm

I just published [Part 2: Apps and templates.](#) 😊

7 Likes

[Qubes for journalists \(a salt and wiki repository\)](#)

[solene](#) 3 August 5, 2023, 8:13am

This is a very good guide, thank you very much

1 Like

[quber](#) 4 August 5, 2023, 9:01am

Thank you, very good. A good example for SaltStack configuration is this:

<https://git.drkhsh.at/salt-n-pepper/file/README.md.html>

It is not mine, I just found it and use it.

1 Like

[nona](#) 5 August 8, 2023, 11:43am

There's so much to learn... 🤔

I'm adding this to my todo list for the next days, thanks a lot! 😊

1 Like

[leo](#) 6 August 26, 2023, 4:42pm

I finally wrote [Part 3: Backing up Qubes.](#) 😊

4 Likes

[tasket](#) 7 August 27, 2023, 1:28am

Thanks! Although I have a long familiarity with Qubes, I never really learned the salt parts... this helps a lot in that regard.

For the Wyng part, I suggest linking to the main project URL instead of tags since 'main' features the most recommended version. Currently the new v0.8beta is recommended as generally better just on account of the extra internal checks, making it a more solid choice than v0.3. The new version is also much easier to use.

I would also suggest dropping the link to the announcement for the older version, as it starts out with examples that are a bit out of date

This part sounds a little dicey...

A malicious file or qube could compromise your system through this command!

...so I added my gpg pubkey and signature files to the project, along with verification [instructions](#). That should help facilitate wording in this guide that sounds more sure-footed; the question then shifts to whether you trust my public key, and doing the verify probably in dom0 after unzipping in the dispvm.

Creating a subvolume (3.2): These instructions seem shorter and don't involve removing files:

```
cd /var/lib
qvm-shutdown --all --force --wait
sudo mv qubes qubes-old
sudo btrfs subvolume create qubes
sudo mv qubes-old/* qubes
```

The above should work fine: The `mv` command will see the src and dest as being the same device because of the way Linux VFS handles nested subvolumes.

Last suggestion: It should mention that Wyng directly deals with only volumes themselves, not qube definitions or VM settings, so only the VM content is backed up this way. A wrapper called [wyng-util-qubes](#) exists which performs send/receive functions and makes sure the Qubes settings are included in the process; it also includes the right combination of volumes for each qube and you only need to know the qube names not any special volume names. Much simpler! The only problem is it doesn't yet handle Btrfs volume paths, so right now it only works for Thin LVM storage.

And thanks again!

3 Likes

[leo](#) 8 August 27, 2023, 2:00pm

Thanks a lot for creating Wyng, and thanks for taking the time to reply!

I added your improvements to the guide.

I have a question: if there is would be a breaking change in the future, some parts of the guide might stop working for readers who are downloading newer versions of Wyng from the main branch. Wouldn't it be safer if the guide would recommend downloading a specific tagged version instead? Another possibility would be to add a disclaimer such as "this guide was written for Wyng v0.8beta, it might not work with more recent versions"...

Thanks again.

Edit: I added the line `shopt -s dotglob` to your instructions for creating the BTRFS subvolume, because otherwise the file `/var/lib/qubes/.qubes-exclude-block-devices` would not be included in the transfer on my machine.

1 Like

[gonzalo-bulnes](#) 9 August 27, 2023, 4:33pm

leo:

Another possibility would be to add a disclaimer such as "this guide was written for Wyng v0.8beta, it might not work with more recent versions"...

I would recommend this instead of instructing to install a specific version of the package because:

- it is likely better for folks to use new versions of the package as they come out (realistically, some folks won't notice the version they installed may be outdated, or date changing the version of the package if it is explicitly specified in the instructions)
- the guide may well be working fine with later versions for a while
- if the guide stops working with a future version of the package, then the disclaimer offers a good prompt for folks to ask for the instructions to be updated (a good reminder for you or whoever is able to update the guide in the future)

2 Likes

[leo](#) 10 August 27, 2023, 5:10pm

Thanks for you advice. I added a disclaimer in that section.

1 Like

[Szewcu](#) 11 August 30, 2023, 2:47pm

Is it really necessary to download the key in `dvm` and copy it to `dom0`? I think better idea will be using proxy inside the template somehow, as we are doing it manually.

1 Like

[leo](#) 12 August 30, 2023, 3:11pm

I think that's a great idea, using a proxy should work but it is [not yet documented](#) in the official Salt docs. It was however already [proposed in a GitHub issue](#) by marmarek so should be a safe thing to do.


Maybe adding something like this would be sufficient:

```
conferencing--set-proxy:
  environ.setenv:
    - value:
      http_proxy: http://127.0.0.1:8082
      https_proxy: http://127.0.0.1:8082
```

Edit: [It isn't.](#)

[Szewcu](#) 13 August 30, 2023, 3:14pm

Don't know how much actual it is but:

[pkgrepo.managed should support proxy option](#)

Documentation


Bug

severity-low

P4

stale

time-estimate-sprint

opened 03:09PM - 05 Jul 18 UTC  [Poil](#)

Hi,

With yum it's sometime useful to configure the proxy in the repo configur [...](#)

1 Like

[Szewcu](#) 14 August 30, 2023, 5:55pm

```
[ERROR ] State 'qvm.clone' was not found in SLS 'debian-google-template'
Reason: 'qvm.clone' is not available.

local:
-----
      ID: clone-template
Function: qvm.clone
      Name: debian-12-google
      Result: False
      Comment: State 'qvm.clone' was not found in SLS 'debian-google-template'
```

```
Reason: 'qvm.clone' is not available.  
Changes:
```

I got such error when try to use qvm.clone in user saltenv... Anyone have an idea what is wrong?

[leo](#) 15 August 30, 2023, 7:01pm

I think this type of error can happen when you try to use qvm commands while [targeting other qubes than dom0](#).

Did you try to apply your Salt state with the following command?

```
sudo qubesctl --targets=debian-12 --show-output state.sls debian-google-templ
```

If so, this tells Salt to target both debian-12 and dom0 to execute the instructions found in the state configuration file. If the file contains qvm commands, this will not work because qvm commands are not available in debian-12, they are only available in dom0.

The following should work:

```
sudo qubesctl state.sls debian-google-template saltenv=user
```

[Szewcu](#) 16 August 30, 2023, 8:28pm

Still the same.

[leo](#) 17 August 30, 2023, 8:31pm

Can you share your state configuration file and the command you use to apply the state?

[Szewcu](#) 18 August 30, 2023, 9:23pm

I must broke something in my salt since recipes in regular salt folder also stop working. Just reinstalled mgmt-salt packages and everything works again. I also wrote recipe that downloading keys and adding external repo today but I will share it after some polishing.

[leo](#) 19 August 31, 2023, 6:33am

I updated the guide to download the key directly in the template. Thanks a lot for your contribution!

1 Like

[Szewcu](#) 20 August 31, 2023, 10:07am

One more thing I still can't figure out. How to download a key from keyserver using proxy. I remember that there was some bug in proxy implementation in gpg while ago.

[leo](#) 21 August 31, 2023, 4:54pm

You could use something like this:

```
conferencing --download-key:
cmd.run:
- name: curl --output /etc/apt/keyrings/skype.asc https://keyserver.ubuntu.com/keys/0x17AD272BB5D734439A0160891428A50170080000
- env:
- https_proxy: http://127.0.0.1:8082
- creates:
- /etc/apt/keyrings/skype.asc
```

This is similar to [unman's recommendation](#) on this topic.

[Szewcu](#) 22 August 31, 2023, 5:31pm

```
leo:
- https_proxy: http://127.0.0.1:8082
```

don't know why this not work form me. Which work is `curl -x`.

[Szewcu](#) 23 August 31, 2023, 5:33pm

```
leo:
- key_url: /etc/apt/keyrings/skype.asc
```

this gives me an error. Deleting this line (since we provided key file externally) make everything work.

[leo](#) 24 August 31, 2023, 6:05pm

Szewcu:

leo:

```
- https_proxy: http://127.0.0.1:8082
```

don't know why this not work form me. Which work is `curl -x`.

It could be that you tried to download something from a HTTP URL while using a proxy with HTTPS. To avoid this issue you could always set both environment variables:

```
conferencing--download-key:
```

```
cmd.run:
```

```
- name: curl --output /etc/apt/keyrings/skype.asc https://keyserver.ubuntu.com/pks/lookup?search=Skype&fingerprint=0x17924F83917B2F2D07C184E077EAC5A4ED770232
- env:
  - http_proxy: http://127.0.0.1:8082
  - https_proxy: http://127.0.0.1:8082
- creates:
  - /etc/apt/keyrings/skype.asc
```

Using curl's `--proxy` option like you did also works.

Szewcu:

leo:

```
- key_url: /etc/apt/keyrings/skype.asc
```

this gives me an error. Deleting this line (since we provided key file externally) make everything work.

That's great! I will leave this line in the guide tough, because it takes care of transforming Skype's GPG key from an ASCII-armored format to a binary format, and this is necessary to use the key with the repository.

[Szewcu](#) 25 August 31, 2023, 6:08pm

ahhh ok, I transform them in place using `gpg --dearmor`. So I got error that they are both the same.

[Szewcu](#) 26 August 31, 2023, 7:34pm

Do you maybe have an idea how to deal with selinux policies in fedora-38? I want to install Threema but I can't. I remember that I probably disable, selinux, reboot, install reboot and again enable selinux...

[leo](#) 27 August 31, 2023, 7:44pm

I haven't got any experience with selinux policies, but with Salt I guess you could use [the selinux interface](#) to manage them.

[leo](#) 28 October 3, 2023, 12:22pm

A small update for anyone interested in making remote backups with Wyng:

After a few weeks of testing, I wrote [section 3.4: Making remote backups](#). Using Wyng with [rclone mount](#) is however quite experimental, so I would recommend using the [local backup method](#) instead. This can be complemented with [rclone sync](#) for example to upload the local backup files to some cloud storage.

Nevertheless, I hope that these are interesting examples of what can be done with Salt 😊

[zaz](#) 29 January 11, 2024, 7:32am

Thanks a lot for writing this!

Ken Rosen and I are working on [adding more example SALT files](#) to Qubes.

I also hope to see a user SALT directory created by default, [backed up by default](#), and documented.

After we get some very minimal examples set up, some of your example states may be good to include!

4 Likes

[Insurgo](#) 30 January 13, 2024, 5:22pm

Most probably doing something stupid but some collaboration would be needed to ease having qubes-builder-v2 available and ready to use.

I tried to replicate [Distribute the build environment for qubes builder \(e.g., template or Salt recipe\) · Issue #8774 · QubesOS/qubes-issues · GitHub](#) but failed. Any of you willing to put this on github and make this mature, usable and ready to use?

[Insurgo](#) 31 January 14, 2024, 5:40am

Salt practitioners, one bug is unfixed upstream and some of that salt configuring a quebes-builder after bug manually fixed under q4.2 is totally skipped. If you can contribute that would be awesome [Distribute the build environment for qubes builder \(e.g., template or Salt recipe\) · Issue #8774 · QubesOS/qubes-issues · GitHub](#)

Goal would be to propose the final recipe as PR so that qubesos deploys salt recipes to deploy qubes builder for anyone wanting to build quebes. That would be needed.

Otherwise if you know how to fix this here works less then on github but that would be better than nothing 😊

[leo](#) 32 January 14, 2024, 7:03am

Would you or anyone be able to check whether [the patch to user-dirs.sls from DemiMarie](#) fixes the issue, and report the result in [issue #8491](#)?

I think nobody was able to test it so far, and this may be what's necessary to solve this bug 😊

2 Likes

[solene](#) 33 January 14, 2024, 1:31pm

Haha, thanks for the reminder, I forgot I was pinged to try the patch 😞

1 Like

[augsch](#) 34 January 14, 2024, 3:40pm

Wow really, I upgraded from R4.2.0 alpha so I didn't even notice that qubes.user-dirs state was broken in later version. Perhaps that's why [@Insurgo](#) cannot use my recipe.

2 Likes

[Insurgo](#) 35 January 14, 2024, 5:07pm

leo:

Would you or anyone be able to check whether [the patch to user-dirs.sls from DemiMarie](#) fixes the issue, and report the result in [issue #8491](#) ?

I think nobody was able to test it so far, and this may be what's necessary to solve this bug

Thank you for that!!!

Done under [Ensure that qubes.user-dirs state actually works by DemiMarie · Pull Request #11 · QubesOS/qubes-mgmt-salt-base-config · GitHub](#)

The fix works!

augusch:

Wow really, I upgraded from R4.2.0 alpha so I didn't even notice that qubes.user-dirs state was broken in later version. Perhaps that's why [@Insurgo](#) cannot use my recipe.

But not the salt recipe to deploy qubes-builder-v2 as reported with logs [there](#) and next reply. Will try other [qubes-builder-v2 salt deployment recipe](#) proposed which is co-dependent on other salt recipes and is not as lean as this salt could be... When personal free time permits it.

Reported PR as fixing user salt dir under [Ensure that qubes.user-dirs state actually works by DemiMarie · Pull Request #11 · QubesOS/qubes-mgmt-salt-base-config · GitHub](#)

1 Like

[SteveC](#) 36 January 14, 2024, 7:58pm

As it so happens, I just tried a clean 4.2 install and did set up user dirs there. Friday and yesterday (since I had to redo the install for other reasons). I saw no problem. (I have no idea what this problem was, in fact.)

[solene](#) 37 January 14, 2024, 10:24pm

Does it work reliably when you run highstate? I found it works a third of the time without error

[Insurgo](#) 38 January 14, 2024, 10:45pm

I'm not sure I follow pas two replies. There is a bug, and a pr to fix that bug at [Ensure that qubes.user-dirs state actually works by DemiMarie · Pull Request #11 · QubesOS/qubes-mgmt-salt-base-config · GitHub](#).

Are you both saying you can't replicate success nor bug with/without fix? Can you please replicate locally qubes builder v2 salt recipes with without fix and comment on both places? The more votes for fixes and bug replication, the faster it is fixed because raising attention. My salt Fu is not good enough to state where thing break, but from my traces posted one can easily see where things go wrong and what is fixed with pr. It needs replication and confirmation.

[solene](#) 39 January 14, 2024, 10:48pm

Insurgo:

Are you both saying you can replicate success nor bug with fix?

I didn't try the fix yet, but the changes applies through salt the fix I reported to work in my original issue, aka the symbolic link required to make it work fine.

I would be surprised if [@SteveC](#) doesn't see the issue at all, because I've been able to reproduce it, even two weeks ago again.

1 Like

[Insurgo](#) 40 January 14, 2024, 10:49pm

solene:

I would be surprised if [@SteveC](#) doesn't see the issue at all, because I've been able to reproduce it, even two weeks ago again.

That is also my point.

1 Like

[SteveC](#) 41 January 15, 2024, 1:26am

I know that yesterday after installing 4.2, I was able to create the user_pillar, user_salt and user_formula directories with that command.

I did have to go into /srv and change ownership of the directories (recursively), then I had to soft link /srv/user_pillar/init.top into /srv/pillar/... (and I needed to do sudo su to be able to do that). If one of those is the bug in question, then there's your answer.

1 Like

[Insurgo](#) 42 January 15, 2024, 5:44am

Wow. The qusal project is simply amazing to read and understand salt recipes. Will test tomorrow [qusal/salt/qubes-builder/README.md at main · ben-grande/qusal · GitHub](#)

Do not skip dom0 requirements at [qusal/docs/BOOTSTRAP.md at main · ben-grande/qusal · GitHub](#)

3 Likes

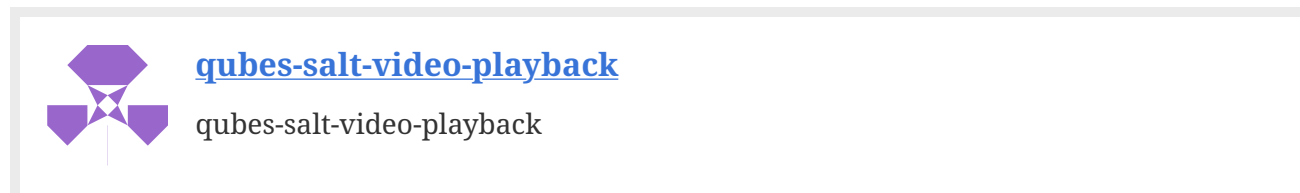
[Bees and brtfs deduplication](#)

[bruno_schroeder](#) 43 January 23, 2024, 12:54pm

Trying to help those whostruggle with SaltStack, and have more Salt code and less visual/bash step-by-steps in the forum, I have just created an MIT License repo for the community.

It's a didactic small and simple config files and step-by-step commands for a VM for video playback with mpv, fzf, smplayer, and vlc inspired by this [forum topic](#).

The repo:



If it fits well, please update in the initial message.

3 Likes

[leo](#) 44 January 23, 2024, 2:44pm

Thanks a lot! I made an [appendix](#) and added a link to your repository. By the way, this topic is a wiki so anyone should feel free to edit the first post 😊

1 Like

[galthop](#) 45 March 6, 2024, 8:13pm

I'm trying to use this but keep getting errors because salt/dotfiles is empty. I see that directory is a git submodule and I think it is empty because I dont have access to the git repo. Do you know where I can get default files for this?

[kenosen](#) 46 March 25, 2024, 6:28am

Insurgo:

Wow. The qusal project is simply amazing to read and understand salt recipes.

Agreed. It is a work of art and the dev is quick to respond. Most comprehensive yet and most states work as intended out of box.

1 Like

[Insurgo](#) 47 March 26, 2024, 12:59am

The project needs testing and bug report. Did open a lot and hope others will do the same.

[edsiot](#) 48 April 4, 2024, 12:54pm

hello everyone,

I'm kinde new there, and can't proceed very far.

I'm stuck on the commands using 'qvm-appmenus' that 'fail with error code: 1'

The error occurs in every codes of 'disconnected' 'messaging' and 'vault' the vm-create-qube part works but fails for the vm-update-app-menu function .

when I run for example 'qvm-appmenus --update salty' directly in terminal I have no errors

If any one know where does the problem come from thanks in advance

best regards ,

edsiot

1 Like

[Cerberus](#) 49 April 5, 2024, 6:21am

A very good example at The Guardian:

When security matters: working with Qubes OS at the Guardian [When security matters: working with Qubes OS at the Guardian | | The Guardian](#)

3 Likes

[parulin](#) 50 April 5, 2024, 6:24am

Hi, could you run the salt state again, with `--show-output` option and copy the output?

I remember having an error while trying to remove the following line:

```
{% set gui_user = salt['cmd.shell']('groupmems --list --group qubes') %}
```

1 Like

[edsiot](#) 51 April 5, 2024, 1:15pm

Hi, I've figured it out;

I'm on Qubes 4.2, don't know if that make a difference but changed the syntax of the code; I write present directly as said by qubes doc and dont repeat the label tag, don't know if it's just for me but now it works, thanks;

```
{% if grains['id'] == 'dom0' %}

{% set gui_user = salt['cmd.shell']('groupmems --list --group qubes') %}

messaging--create-qube:
  qvm.present:
    - name: messaging
    - template: debian-12
    - label: yellow
    - features:
      - set:
        - menu-items: org.telegram.desktop.desktop org.gnome.Nautilus.desktop

messaging--update-app-menu:
  cmd.run:
    - name: qvm-appmenus --update messaging
    - runas: {{ gui_user }}
    - require:
      - qvm: messaging--create-qube

{% elif grains['id'] == 'debian-12' %}

messaging--install-apps-in-template:
  pkg.installed:
    - pkgs:
      - telegram-desktop
```

3 Likes

[leo](#) 52 April 5, 2024, 2:38pm

That's interesting. As I understand it, the state configuration using the state command [qvm.vm](#),

```

messaging--create-qube:
  qvm.vm:
    - name: messaging
    - present:
      - template: debian-12
      - label: yellow
    - prefs:
      - label: yellow
    - features:
      - set:
        - menu-items: org.telegram.desktop.desktop org.gnome.Nautilus.desktop

```

is equivalent to using the [qvm.present](#), [qvm.prefs](#) and [qvm.features](#) separately:

```

messaging--create-qube:
  qvm.present:
    - name: messaging
    - template: debian-12
    - label: yellow

messaging--set-prefs:
  qvm.prefs:
    - name: messaging
    - label: yellow

messaging--set-features:
  qvm.features:
    - name: messaging
    - set:
      - menu-items: org.telegram.desktop.desktop org.gnome.Nautilus.desktop

```

Maybe `qvm.prefs` or `qvm.features` is causing the issue.

Edit: It seems that since [commit 8ed18dc](#) to [qubes-desktop-linux-common](#), the issue [#8494](#) is fixed and a call to `qvm-appmenus` is not longer necessary to refresh the list of applications after changing it with `qvm.features`. In this case, the parts

```
{% set gui_user = salt['cmd.shell']('groupmems --list --group qubes') %}
```

and

```

cmd.run:
  - name: qvm-appmenus --update name-of-qube
  - runas: {{ gui_user }}

```

can be removed from all the Salt configuration files.

Has anyone tested whether the list of applications is refreshed when omitting those parts? If so, I'll update the guide.

1 Like

[parulin](#) 53 April 6, 2024, 9:42am

leo:

Has anyone tested whether the list of applications is refreshed when omitting those parts? If so, I'll update the guide.

I just tried this state file:

```
create-test-vm:
  qvm.vm:
    - name: test-appmenu
    - present:
      - template: debian-12-xfce
      - label: red
    - features:
      - set:
        - menu-items: org.xfce.mousepad-settings.desktop xarchiver.desktop
```

And the appmenu is refreshed (even when changing the `menu-items` list) so I need to update my salt states too 😊

EDIT: I think this guide is much better with this tip, thanks! Please check if I removed too much code, I hope not.

2 Likes

[leo](#) 54 April 7, 2024, 8:22pm

Many thanks!

1 Like

[nokke](#) 55 May 21, 2024, 1:21pm

Trying to create a non-free template, I went for `file.managed` instead of an explicit curl command.

Unfortunately, I couldn't find a way to pass the update proxy variable, so abandoned this in the end. Everything else I tried resulted in a DNS resolution error.

This is an [upstream issue](#) that they usually resolve by setting the proxy in the minion config. That's not a great solution for how Qubes does it (single minion, config/proxy would be shared by all targets), so it looks like `cmd.run` is the way to go.

1 Like

[kenosen](#) 56 May 21, 2024, 5:15pm

nokke:

I couldn't find a way to pass the update proxy variable

Try:

```
curl --proxy http://127.0.0.1:8082/ --tlsv1.2 --proto =https --max-time 180 -
```

1 Like

[nokke](#) 57 May 21, 2024, 7:23pm

That's in a shell context, how would you pass it to `file.managed?` `env` doesn't do it, and setting it earlier with `environ.setenv` also doesn't work.

[Solomon1732](#) 58 October 31, 2024, 5:29pm

For my setup there are a few packages which often come together. Instead of copy-pasting them for each template, I wish to use a YAML file to group certain packages together in a bundle, as it were. This will let me write the name of each group instead of specifying the members of each group in each template.

I tried to do it in the past. However, I have no idea how to do it even though I looked into it.

Is it possible to add such a section?

1 Like

[quber](#) 59 October 31, 2024, 5:53pm

yaml file: groups.yml

```
group_one:
  - test
  - testt
group_two:
  - test
  - testt
```

then you import:

```
{% import_yaml slspath + "/groups.yaml" as groups %}
```

```
install-pkg:
  pkg.installed:
    - pkgs:
      - {{ groups.group_one }}
      - {{ groups.group_two }}
```

2 Likes

[Solomon1732](#) 60 October 31, 2024, 6:05pm

Thanks a bunch!

1 Like