pyqubes Documentation

Release 0.0.1

Tom Milligan

May 14, 2017

Contents:

CHAPTER 1

Installation

Install with pip:

```
pip install pyqubes
```

Use

Todo

General use on AppVMs

dom0

Todo

Special use on dom0 without installation

CHAPTER 2

Examples

Todo

Examples

- pythonic
- direct

CHAPTER 3

API

This API documentation is automatically generated.

Pythonic Classes

VM

The top level `VM` object holds common methods for VMs.

It should not be instanciated directly - use the lower level `TemplateVM` and `AppVM` objects instead.

class `pyqubes.vm.`**`VM`**(*name*, *proactive=False*, *operating_system='fedora-23'*)

The VM object represants a QubesOS VM. Its methods are common accross both AppVMs and TemplateVMs.

VM should not be instanciated directly - use TemplateVM or AppVM.

By default, all VMs are Fedora 23 based. Other values are listed in `pyqubes.constants`

`animate`()

Magically start and stop the VM:

```
vm = TemplateVM('foo')
# Template is not started on instanciation
with vm.animate():
    # Template is now running
    vm.update()
# VM is shut down automatically
```

`enact`(*args*)

Enact a list of command arguments using the VM's `enact_function`

Any one of the functions in `pyqubes.qubes`,''pyqubes.qubesdb'' or `pyqubes.qvm` will return arguments in the correct format.

firewall(***kwargs*)
Edit the VM firewall

firewall_close()
Can be explicity called to close the VM firewall to 'deny'.

firewall_open()
Can be explicity called to open the VM firewall to 'allow'.

In most cases you should use''with vm.internet''.

info(*info*)
Echo information from pyqubes using the VM's enact method

internet()
Magically open and close the VM firewall:

```
vm = TemplateVM('foo')
with vm.animate:
    # Templates are offline be default
    with vm.internet:
        # Template now has unrestricted internet access
        vm.run('curl http://ipecho.net/plain')
    # Firewall is restored automatically
    # This will now fail
    vm.run('curl http://ipecho.net/plain')
```

remove(***kwargs*)
Remove the VM

run(*command*, *quote=True*, ***kwargs*)
Run a command on the VM.

Please note: * `--pass-io` is always set, to run commands synchronously * Commands are automatically encapsulated in single quotes:

```
vm = TemplateVM('spam')
vm.run('echo "foo bar"')
# produces: qvm-run spam 'echo "foo bar"'
```

Parameters **quote** (*bool*) – By default command is single quoted - set `False` to disable

shutdown(***kwargs*)
Shutdown the

Please note: * `--wait` is always set, to run commands synchronously

start(***kwargs*)
Start the VM explicitly.

In most cases you should use''with vm.animate()''.

TeamplateVM & AppVM

These represent the actual VMs within QubesOS.

Methods mentioned here are specific to the VM type.

class `pyqubes.vm.`**TemplateVM**(**args*, ***kwargs*)
TemplateVM - for installing apps

clone(*clone_name*, ***kwargs*)
Clone the TemplateVM and return a new TemplateVM

Parameters **clone_name** (*string*) – Name of the new VM

Returns The new `TemplateVM` instance

create_app(*app_name*, ***kwargs*)
Create and return a new AppVM based on the TemplateVM.

Please note: * If `label` is not set, it will default to `red`

Parameters **app_name** (*string*) – Name of the new VM

Returns The new `AppVM` instance

update()
Smartly runs the relevant package manager updates for the TemplateVM

class pyqubes.vm.**AppVM**(**args*, ***kwargs*)
AppVM - for running apps

Helper Classes

Direct command wrapping

qubes- commands

qubesdb- commands

qvm- commands

pyqubes.qvm.**qvm_clone**(*vm_name*, *clone_name*, *quiet=False*, *path=''*)
qvm-clone

pyqubes.qvm.**qvm_create**(*vm_name*, *template=''*, *label=''*, *proxy=False*, *net=False*, *hvm=False*, *hvm_template=False*, *root_move_from=''*, *root_copy_from=''*, *standalone=False*, *mem=0*, *vcpus=0*, *internal=False*, *force_root=False*, *quiet=False*)
qvm-create

pyqubes.qvm.**qvm_firewall**(*vm_name*, *list_view=False*, *add_rule=''*, *del_rule=''*, *set_policy=''*, *set_icmp=''*, *set_dns=''*, *set_yum_proxy=''*, *numeric=False*)
qvm-firewall

pyqubes.qvm.**qvm_remove**(*vm_name*, *quiet=False*, *just_db=False*, *force_root=False*)
qvm-remove

pyqubes.qvm.**qvm_run**(*vm_name*, *command*, *quiet=False*, *auto=False*, *user=''*, *tray=False*, *all_vms=False*, *exclude=[]*, *wait=False*, *shutdown=False*, *pause=False*, *unpause=False*, *pass_io=False*, *localcmd=''*, *force=False*)
qvm-run

pyqubes.qvm.**qvm_shutdown**(*vm_name*, *quiet=False*, *force=False*, *wait=False*, *all_vms=False*, *exclude=[]*)
qvm-shutdown

pyqubes.qvm.**qvm_start**(*vm_name*, *quiet=False*, *no_guid=False*, *console=False*, *dvm=False*, *custom_config=''*)
qvm-start

Utilities

Compile

The `compile` module converts instructions from pythonic data structures into flat lists.abs

These may require further processing before being passed to the `enact` module for action.

`pyqubes.compile.`**`flags_boolean`**(*flags*)
Return a list of string values, corresponding to the given keys whose values evaluate to True

All keys and values will be converted to strings.

Parameters **`flags`** (*`dict`*) – A dictionary in the form `{'--flag': boolean}`, where `boolean` is used to determine whether `--flag` is included in the output.

Returns A flat list of strings

`pyqubes.compile.`**`flags_store`**(*flags*)
Return a list of string values, corresponding to the given keys and values whose values evaluate to True

The output is a flat list of all strings.

All keys and values will be converted to strings.

Parameters **`flags`** (*`dict`*) – A dictionary in the form `{'--flag': value}`, where `value` is used to determine whether the entry is included in the output.

Returns A flat list of strings

`pyqubes.compile.`**`flags_store_iterable`**(*flags*)
Calls `flags_store` for each value within each key in `flags`.

e.g. {'–fruits': ['apple', 'pear']} results in ['–fruits', 'apple', '–fruits', 'pear']

Parameters **`flags`** (*`dict`*) – A dictionary in the form `{'--flag': value}`, where `value` is an iterable.

Returns A flat list of strings

Raises `TypeError` if values are not iterable

`pyqubes.compile.`**`info`**(*info*, *quote=True*, *style=True*)
Returns the given string `info` as a set of echo arguments.

Optionally provides quoting and terminal styling.

Parameters

- **`info`** (*`string`*) – Info string to add to script
- **`quote`** (*`bool`*) – By default quote given sting in single quotes
- **`style`** (*`bool`*) – By default add

Returns A flat list of strings

Constants

Enact

The enact module contains functions that act on a list of command-line arguments,

The two most important ones are: * Direct execution with `call` (proactive mode) * Echoing an execution-ready script with `echo` (reactive mode)

`pyqubes.enact.`**`call`**(*args*, ***kwargs*)
Thin wrapper for builtin subprocess.call

`pyqubes.enact.`**`call_quiet`**(*args*)
Uses the `call` function, but throws away stdout and stderr.

Should be used for internal unit tests wherever possible.

`pyqubes.enact.`**`echo`**(*args*, *file=None*)
Echo a list of arguments (as given to `subprocess.call`) to the given stream.

This defaults to `stdout`, but can be changed to any stream-like object such as a file handle.

Parameters

- **`args`** – A string or list of strings
- **`file`** – A file-like object to stream output to. Defaults to `sys.stdout`

Utils

Utility functions for pyqubes

Utilities have no dependencies.

`pyqubes.utils.`**`assert_list_items_equal_in_nested`**(*actual_nested*, *expected_list*)
Assert that the given `expected_list` matches one pf the lists within `actual_nested`, using the comparison `sorted(list)`

Parameters

- **`actual_nested`** (*`list`*) – A list of lists (usually generated by test)
- **`expected_list`** (*`list`*) – A list of expected values

`pyqubes.utils.`**`flatten_list`**(*nested_list*)
Flatten a nested list one level.

Parameters **`nested_list`** (*`list`*) – A nested list

Return type list

`pyqubes.utils.`**`object_fullname`**(*obj*)
Returns the full absolute name of the object provided

`pyqubes.utils.`**`object_logger`**(*obj*)
Returns a correctly named logger for the given object.

Call as self.logger = object_logger(self)

Validate

Validate functions for pyqubes

These will return the original value if validation passes. Otherwise, `ValueError` will be raised

`pyqubes.validate.`**`firewall_policy`**(*policy*)
qvm-firewall policy string should match `^(allow|deny)$`

Parameters **`policy`** (*`string`*) – Policy string to check

Returns `policy` if valid, else `ValueError`

`pyqubes.validate.`**`label_color`**(*color*)
VM label color should be one of: * red * orange * yellow * green * blue * purple * black * gray

Parameters **`color`** (*`string`*) – Label color string to check

Returns `color` if valid, else `ValueError`

`pyqubes.validate.`**`linux_hostname`**(*hostname*)
Linux hostnames are recommended to match `^[a-zA-Z0-9-]+(\.[a-zA-Z0-9-]+)*$`

Parameters **`hostname`** (*`string`*) – Hostname to check

Returns `hostname` if valid, else `ValueError`

`pyqubes.validate.`**`linux_username`**(*username*)
Linux usernames are recommended to match `^[a-z_][a-z0-9_-]*\$?$`

Parameters **`username`** (*`string`*) – Username to check

Returns `username` if valid, else `ValueError`

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index