vmupdate Documentation

Release 0.3.0

Author

May 03, 2016

vmupdate is a command line utility used to keep your virtual machines up to date. It searches your computer for virtualizers, queries them for a list of VM's, and runs the appropriate update commands.

Head on over to *Getting Started* for more information.

CHAPTER 1

Features

1.1 Virtualizers

- **Windows**
  - VirtualBox

1.2 Guests

- Arch
- Debian
- Fedora
- Red Hat
- Ubuntu

1.2.1 Getting Started

Installation

The recommended installation tool is **pip**:

```
$ pip install vmupdate
```

Configuration

Create a custom configuration file *vmupdate.yaml*:

```
Credentials:
  Username: myuser
  Password: mypass
```

Note: This method is included for simplicity, but is not recommended due to the inherent insecurity of a plaintext password. See *Configuration* for more options.

Command

And pass the path to the utility:

```
$ vmupdate --config "/path/to/config/vmupdate.yaml"
```

1.2.2 Configuration

Configuration is at the root of **vmupdate** and as a user you can override virtually all of the utility's functionality to suit your needs. For most purposes setting up the *Credentials* will be sufficient. To override the configuration (including `Credentials`) for specific VM's see *Machines*.

You can pass a custom config file as follows:

```
$ vmupdate --config "/path/to/config/vmupdate.yaml"
```

Note: Nested keys will be merged, but values will be replaced. Thus, when modifying a list make sure to include any original list items that you wish to keep.

Specification

Credentials

The `Credentials` section is used for options relating to authentication and access. These options will be used for all VM's unless specifically overridden (see *Machines*).

```
Credentials:
  Username: myuser
  Password: mypass
  Use Keyring: true
  Run As Elevated: true
```

Username The username used to authenticate with the VM. Defaults to `root`.

Password The password used to authenticate with the VM. Defaults to `null`.

Use Keyring Whether to use the host's keyring to access the password. See *Using the Keyring* for more details. Defaults to `true`.

Run As Elevated Whether to use an elevated user mode when running commands against the VM. This will be required by most guest operating system configurations. Defaults to `true`.

Warning: Setting a password in plaintext is generally insecure. Use of the keyring is encouraged.

General

The `General` section is used for miscellaneous options.

```
General:
  Wait After Start: 30
  Wait Before Stop: 10
```

Wait After Start Time in seconds to wait after starting the VM. Defaults to `30`.

Wait Before Stop Time in seconds to wait before stopping the VM. Defaults to `10`.

Network

The `Network` section is used for options relating to SSH endpoints. These are advanced options and generally don't need to be modified.

```
Network:
  SSH:
    Guest:
      Port: 22
    Host:
      Ports:
        Min: 49152
        Max: 65535
```

SSH

Guest Port SSH port of the guest. Defaults to `22`.

Host Ports Range of ports to be used on the host for forwarding SSH to the guest. Defaults to `49152 - 65535`.

Package Managers

The `Package Managers` section is used for configuring package managers on guest operating systems. These are advanced options and generally don't need to be modified.

```
Package Managers:
  Ubuntu:
    apt-get:
      - update -y -u -q
      - upgrade -y -u -q
```

This example configures the utility to run `apt-get` with the `update` and `upgrade` commands on guests running `Ubuntu`.

Shells

The `Shells` section is used for configuring *`shells`* for communicating with the guest operating system. These are advanced options and generally don't need to be modified.

```
Shells:
  Ubuntu: Posix
```

This example configures the utility to use the *`Posix`* shell to communicate with guests running `Ubuntu`.

Machines

The `Machines` section is used for overriding the configuration for specific virtual machines.

```
Machines:
  My Machine:
    Username: myuser
    Password: mypass
```

```
    Use Keyring: true
    Run As Elevated: true
    Shell: Posix
    Ignore: false
```

Username The username used to authenticate with the VM.

Password The password used to authenticate with the VM.

Use Keyring Whether to use the host's keyring to access the password. See *Using the Keyring* for more details.

Run As Elevated Whether to use an elevated user mode when running commands against the VM. This will be required by most guest operating system configurations.

Shell Which shell to use for communicating with the guest operating system.

Ignore Whether to skip the machine for updating. Defaults to `false`.

`My Machine` is the name of the virtual machine as listed in the virtualizer.

Virtualizers

The `Virtualizers` section is used for configuring *`virtualizers`* that may be found on the host. These are advanced options and generally don't need to be modified.

```
Virtualizers:
  Windows:
    VirtualBox:
      - $PROGRAMW6432\Oracle\VirtualBox\VBoxManage.exe
      - $PROGRAMFILES\Oracle\VirtualBox\VBoxManage.exe
```

This example configures the utility to search for *`VirtualBox`* on `Windows` hosts at the listed paths. The first path found will be used.

Note: `$[ENVAR]` in the paths will be expanded using environment variables on the host.

Examples

Using the Keyring

The keyring of your host is the most secure place to store the password(s) used by the utility.

Note: Keyring lookup is by label and username. Both must match to retrieve the password.

General Credentials In your config file:

```
Credentials:
  Username: myuser
  Use Keyring: true
```

Then in your keyring provider, set the password using the label `vmupdate` and your chosen username. This will act as the default authentication profile for all virtual machine connections.

Machine Credentials You may have different credentials for a specific machine.

In your config file:

```
Machines:
  My Machine:
    Username: myuser
    Use Keyring: true
```

Then in your keyring provider, set the password with the label as your machine name (`My Machine` in the example). This will override the authentication profile for this machine.

1.2.3 Troubleshooting

SSH

SSH is used to communicate with VM's so you will need an SSH server enabled on each virtual machine. This is often the case by default with many **nix* installations, but may have to be installed separately.

Port Forwarding

An attempt will be made to forward the configured guest SSH port on each VM to a unique port on the host if such a port forwarding does not already exist. This only needs to be done once per virtual machine and can only occur if the VM is in a *stopped* state. If the automatic port forwarding fails, you can configure it yourself using your virtualizer.

Elevated User

Most guests will require elevated access (i.e. *sudo*) to run updates. Make sure the account you use can run as an elevated user.

PyCrypto Install

If you get a PyCrypto build error during installation please see the paramiko install docs.

1.2.4 Code

The code can be found on GitHub.

vmupdate.channel

Provide wrapper classes around virtual machine communication.

class `vmupdate.channel.`**`Channel`**(*hostname*, *port*)

Bases: `object`

Provide virtual machine communication.

Variables

- **`hostname`** (*`str`*) – name or IP of the virtual machine
- **`port`** (*`int`*) – port of the virtual machine

close()
Close connection and release resources.

connect(*username*, *password*)
Connect to the virtual machine.

Parameters

- **username** (*str*) – username for authentication
- **password** (*str*) – password for authentication

run(*args*)
Run command against the virtual machine and return a *ChannelCommand*.

Parameters **args** (*str or list*) – the command to be run

Return type *ChannelCommand*

class vmupdate.channel.**ChannelCommand**(*stdin*, *stdout*, *stderr*)
Bases: object

Contain pipes returned from executed command.

Variables

- **stdin** (*pipe*) – standard input
- **stdout** (*pipe*) – standard output
- **stderr** (*pipe*) – standard error

wait()
Wait for the command to complete and return the exit code.

Return type int

vmupdate.cli

Provide the main entry point and command line parsing.

vmupdate.cli.**main**()
Initialize environment and call *host.update_all_vms()*.

This is the main entry point for vmupdate.

Returns exitcode

Return type int

vmupdate.config

Provide a wrapper around configuration.

class vmupdate.config.**Config**
Bases: *vmupdate.config.ConfigSection*

Provide a wrapper for the merged configuration files.

credentials
Return the *Credentials* configuration section.

Return type *Credentials*

general
Return the *General* configuration section.

Return type *General*

load(*config_path=None*, *log_dir=None*)
Load the configuration files and configure logging.

Parameters

- **config_path** (*str*) – path to a user defined configuration file
- **log_dir** (*str*) – path to the directory where log files are to be stored

machines
Return the *Machines* configuration section.

Return type *Machines*

network
Return the *Network* configuration section.

Return type *Network*

pkgmgrs
Return the *Package Managers* configuration section.

Return type *PackageManagers*

shells
Return the *Shells* configuration section.

Return type *Shells*

virtualizers
Return the *Virtualizers* configuration section.

Return type *Virtualizers*

class vmupdate.config.**ConfigSection**(*data=None*)
Bases: object

Provide a base class for configuration sections.

This class wraps a dict.

get(*key*, *default=None*)
Return the value for key, else default.

items()
Return a copy of the config section's list of (key, value) pairs.

keys()
Return a copy of the config section's list of keys.

values()
Return a copy of the config section's list of values.

class vmupdate.config.**Credentials**(*data*)
Bases: *vmupdate.config.ConfigSection*

Provide a wrapper around the credentials configuration section.

password
Return the *Password* configuration.

Return type str

run_as_elevated
Return the *Run As Elevated* configuration.

Return type bool

use_keyring
Return the *Use Keyring* configuration.

Return type bool

username
Return the *Username* configuration.

Return type str

class vmupdate.config.**General**(*data*)
Bases: *vmupdate.config.ConfigSection*

Provide a wrapper around the general configuration section.

wait_after_start
Return the *Wait After Start* configuration.

Return type int

wait_before_stop
Return the *Wait Before Stop* configuration.

Return type int

class vmupdate.config.**Machine**(*data*)
Bases: *vmupdate.config.ConfigSection*

Provide a wrapper around the machine configuration section.

ignore
Return the *Ignore* configuration.

Return type bool

password
Return the *Password* configuration.

Return type str

run_as_elevated
Return the *Run As Elevated* configuration.

Return type bool

shell
Return the *Shell* configuration.

Return type str

use_keyring
Return the *Use Keyring* configuration.

Return type bool

username
Return the *Username* configuration.

Return type str

class vmupdate.config.**Machines**(*data*)
Bases: *vmupdate.config.ConfigSection*

Provide a wrapper around the machines configuration section.

This class wraps a dict of *Machine*.

class vmupdate.config.**Network**(*data*)
Bases: *vmupdate.config.ConfigSection*

Provide a wrapper around the network configuration section.

ssh
Return the *SSH* configuration section.

Return type *Ssh*

class vmupdate.config.**PackageManagers**(*data*)
Bases: *vmupdate.config.ConfigSection*

Provide a wrapper around the package managers configuration section.

class vmupdate.config.**Shells**(*data*)
Bases: *vmupdate.config.ConfigSection*

Provide a wrapper around the shells configuration section.

class vmupdate.config.**Ssh**(*data*)
Bases: *vmupdate.config.ConfigSection*

Provide a wrapper around the SSH configuration section.

guest_port
Return the guest port configuration.

Return type int

host_max_port
Return the host port maximum configuration, else 65,535.

Return type int

host_min_port
Return the host port minimum configuration.

Return type int

class vmupdate.config.**Virtualizers**(*data*)
Bases: *vmupdate.config.ConfigSection*

Provide a wrapper around the virtualizers configuration section.

vmupdate.constants

Provide constants for vmupdate.

vmupdate.constants.**OS_ARCH** = **'Arch'**
VM OS Arch

vmupdate.constants.**OS_DEBIAN** = **'Debian'**
VM OS Debian

vmupdate.constants.**OS_FEDORA** = **'Fedora'**
VM OS Fedora

vmupdate.constants.**OS_GENTOO = 'Gentoo'**
VM OS Gentoo

vmupdate.constants.**OS_LINUX = 'Linux'**
VM OS Linux

vmupdate.constants.**OS_MAC_OS_X = 'Mac OS X'**
VM OS Mac OS X

vmupdate.constants.**OS_MANDRIVA = 'Mandriva'**
VM OS Mandriva

vmupdate.constants.**OS_OPENSUSE = 'openSUSE'**
VM OS openSUSE

vmupdate.constants.**OS_ORACLE = 'Oracle'**
VM OS Oracle

vmupdate.constants.**OS_REDHAT = 'Red Hat'**
VM OS Red Hat

vmupdate.constants.**OS_TURBOLINUX = 'Turbolinux'**
VM OS Turbolinux

vmupdate.constants.**OS_UBUNTU = 'Ubuntu'**
VM OS Ubuntu

vmupdate.constants.**OS_UNKNOWN = 'Unknown'**
VM OS Unknown

vmupdate.constants.**OS_WINDOWS = 'Windows'**
VM OS Windows

vmupdate.constants.**OS_XANDROS = 'Xandros'**
VM OS Xandros

vmupdate.constants.**VM_PAUSED = 3**
VM State Paused

vmupdate.constants.**VM_RUNNING = 1**
VM State Running

vmupdate.constants.**VM_STOPPED = 0**
VM State Stopped

vmupdate.constants.**VM_SUSPENDED = 2**
VM State Suspended

vmupdate.constants.**VM_UNKNOWN = -1**
VM State Unknown

vmupdate.credentials

Provide functions for accessing credential information from the config and keyring.

vmupdate.credentials.**get_credentials**(*uid*)
Return the configured credentials for the virtual machine.

Parameters **uid** (*str*) – name of the virtual machine

Returns tuple of (username, password)

Return type (str, str)

vmupdate.credentials.**get_password**(*username*, *uid*)
Return the password for the username and virtual machine.

Parameters

- **username** (*str*) – username associated with the password
- **uid** (*str*) – name of the virtual machine

Returns password

Return type str

vmupdate.credentials.**get_run_as_elevated**(*uid*)
Return whether to run commands as an elevated user for virtual machine.

Parameters **uid** (*str*) – name of the virtual machine

Return type bool

vmupdate.credentials.**get_username**(*uid*)
Return the username for the virtual machine.

Parameters **uid** (*str*) – name of the virtual machine

Returns username

Return type str

vmupdate.errors

Provide application-specific error classes.

exception vmupdate.errors.**AppError**
Bases: exceptions.Exception

Provide base class for application-specific errors.

exception vmupdate.errors.**SshError**
Bases: *vmupdate.errors.AppError*

Provide class for SSH errors.

exception vmupdate.errors.**UpdateError**
Bases: *vmupdate.errors.AppError*

Provide class for update errors.

vmupdate.host

Provide functions to find and update VM's.

vmupdate.host.**update_all_vms**()
Update all virtual machines on the system.

Returns exitcode

Return type int

vmupdate.pkgmgr

Provide functions to query and command package managers.

vmupdate.pkgmgr.**get_pkgmgrs**(*vm*)
Return all package managers on the virtual machine.

Parameters **vm** (*VM*) – virtual machine to target

Returns list of tuples of (name, list of paths)

Return type list((str, list(str)))

vmupdate.pkgmgr.**run_pkgmgr**(*vm*, *pkgmgr*, *cmds*)
Run the package manager commands on the virtual machine in sequence.

Parameters

- **vm** (*VM*) – virtual machine to target
- **pkgmgr** (*str*) – name of the package manager to run
- **cmds** (*list(str)*) – list of commands to run in sequence

Raises **UpdateError** – if any command does not exit with 0

vmupdate.shells

Provide a transparent abstraction for interacting with shells.

class vmupdate.shells.**Posix**(*channel*)
Bases: *vmupdate.shells.Shell*

Represent a POSIX shell that communicates through a channel.

Variables ***channel*** (*Channel*) – channel used for virtual machine communication

command_exists(*command*)
Return whether the command exists in the shell.

Parameters **command** (*str*) – name of the command

Return type bool

run_as_elevated(*args*, *password*)
Run command against the virtual machine as an elevated user.

Parameters

- **args** (*str or list*) – the command to be run
- **password** (*str*) – password to be used for elevated authentication

Return type *ChannelCommand*

class vmupdate.shells.**Shell**
Bases: object

Abstract virtual machine shell that communicates through a channel.

This class must be inherited and cannot be used directly.

Variables ***channel*** (*Channel*) – channel used for virtual machine communication

close()
Close channel and release resources.

command_exists(*command*)
Return whether the `command` exists in the shell.

This is a shell-specific command and must be overridden.

Parameters **command** (*str*) – name of the command

Return type bool

run(*args*)
Run command against the virtual machine.

Parameters **args** (*str or list*) – the command to be run

Return type *ChannelCommand*

run_as_elevated(*args*, *password*)
Run command against the virtual machine as an elevated user.

This is a shell-specific command and must be overridden.

Parameters

- **args** (*str or list*) – the command to be run
- **password** (*str*) – password to be used for elevated authentication

Return type *ChannelCommand*

vmupdate.shells.**get_shell**(*name*, *channel*)
Return an instance of a shell.

The shell should extend *Shell*.

Parameters

- **name** (*str*) – name of the shell class to instantiate
- **channel** (*Channel*) – channel instance to pass to the constructor

vmupdate.virtualizers

Provide a transparent abstraction for interacting with virtualizers.

class vmupdate.virtualizers.**VirtualBox**(*manager_path*)
Bases: *vmupdate.virtualizers.Virtualizer*

Control the VirtualBox virtualizer.

enable_ssh(*uid*, *host_port*, *guest_port*)
Enable SSH port forwarding for the virtual machine.

Parameters

- **uid** (*str*) – identifier of the machine
- **host_port** (*int*) – the post on the host to forward to the guest
- **guest_port** (*int*) – SSH port of the guest

Returns exitcode

Return type int

get_ssh_info(*uid*, *ssh_port*)
Return the SSH connection information for the virtual machine.

Parameters

- **uid** (*str*) – identifier of the machine
- **ssh_port** (*int*) – expected SSH port of the guest

Returns tuple of (hostname, port)

Return type (str, int)

get_vm_os(*uid*)
Return the operating system of the virtual machine.

Possible values can be found in *constants*.

Parameters **uid** (*str*) – identifier of the machine

Return type str

get_vm_status(*uid*)
Return the status of the virtual machine.

Possible values can be found in *constants*.

Parameters **uid** (*str*) – identifier of the machine

Return type str

list_vms()
Return all virtual machines.

Returns list of tuple (name, id)

Return type list(str, str)

start_vm(*uid*)
Start the virtual machine.

Parameters **uid** (*str*) – identifier of the machine

Returns exitcode

Return type int

stop_vm(*uid*)
Stop the virtual machine.

Parameters **uid** (*str*) – identifier of the machine

Returns exitcode

Return type int

class vmupdate.virtualizers.**Virtualizer**
Bases: object

Abstract virtualizer control.

This class must be inherited and cannot be used directly.

enable_ssh(*uid*, *host_port*, *guest_port*)
Enable SSH port forwarding for the virtual machine.

This is a virtualizer-specific command and must be overridden.

Parameters

- **uid** (*str*) – identifier of the machine

- **host_port** (*int*) – the post on the host to forward to the guest
- **guest_port** (*int*) – SSH port of the guest

Returns exitcode

Return type int

get_ssh_info(*uid*, *ssh_port*)
Return the SSH connection information for the virtual machine.

This is a virtualizer-specific command and must be overridden.

Parameters

- **uid** (*str*) – identifier of the machine
- **ssh_port** (*int*) – expected SSH port of the guest

Returns tuple of (hostname, port)

Return type (str, int)

get_vm_os(*uid*)
Return the operating system of the virtual machine.

This is a virtualizer-specific command and must be overridden.

Possible values can be found in *constants*.

Parameters **uid** (*str*) – identifier of the machine

Return type str

get_vm_status(*uid*)
Return the status of the virtual machine.

This is a virtualizer-specific command and must be overridden.

Possible values can be found in *constants*.

Parameters **uid** (*str*) – identifier of the machine

Return type str

list_vms()
Return all virtual machines.

This is a virtualizer-specific command and must be overridden.

Returns list of tuple (name, id)

Return type list(str, str)

start_vm(*uid*)
Start the virtual machine.

This is a virtualizer-specific command and must be overridden.

Parameters **uid** (*str*) – identifier of the machine

Returns exitcode

Return type int

stop_vm(*uid*)
Stop the virtual machine.

This is a virtualizer-specific command and must be overridden.

Parameters **uid** (*str*) – identifier of the machine

Returns exitcode

Return type int

vmupdate.virtualizers.**get_virtualizer**(*name*, *path*)
Return an instance of a virtualizer.

The virtualizer should extend *Virtualizer*.

Parameters

- **name** (*str*) – name of the virtualizer class to instantiate
- **path** (*str*) – path of the virtualizer to pass to the constructor

vmupdate.vm

Provide a wrapper class around VM interactions.

class vmupdate.vm.**VM**(*virtualizer*, *uid*)
Bases: object

Provide virtual machine interface.

Variables

- **virtualizer** (*Virtualizer*) – virtualizer that the virtual machine runs under
- **uid** (*str*) – identifier of the virtual machine

connect()
Connect to the virtual machine and return a shell.

Return type *Shell*

enable_ssh(*host_port*)
Enable SSH port forwarding for the virtual machine.

Parameters **host_port** (*int*) – the post on the host to forward to the guest

Returns exitcode

Return type int

get_os()
Return the operating system of the virtual machine.

Possible values can be found in *constants*.

Return type str

get_ssh_info()
Return the SSH connection information for the virtual machine.

Returns tuple of (hostname, port)

Return type (str, int)

get_status()
Return the status of the virtual machine.

Possible values can be found in *constants*.

Return type str

shell_name
Return the name of the shell.

Return type str

ssh_port
Return the SSH port of the guest.

Return type int

start()
Start the virtual machine.

Returns exitcode

Return type int

stop()
Stop the virtual machine.

Returns exitcode

Return type int

V

O

P

R

S

U

V

W