# Python linux iso Documentation

*Release 0.1*

**Jean-Charles Naud**

**Dec 04, 2018**

# Contents

Contents:

CHAPTER 1

# Introduction

## 1.1 Resume

This programme provide a way to **download**, **custom** and **deploy**, full **automatically**, an OS ISO. The second goal of this project is to never use linux root access to make this.

**This programme have tree modules:**

- download
- custom
- virtualbox
- (FUTURE DEVELOPPEMENT) pxe

**Actually, you can custom this offical ISO:**

- debian 9
- debian 9 with raid 1 (UNDER DEVELOPPEMENT)
- unbuntu 16 (UNDER DEVELOPPEMENT)
- unbuntu 17 (UNDER DEVELOPPEMENT)

**Each module can be use independently in different way like:**

- Command Line Interface (CLI)
- python module

**This programme is run with python:**

- 3.6 (UNDER DEVELOPPEMENT)
- 3.5

## 1.2 Installation

This programme works only on **linux distribution**.

To avoid using root access, we need some tools for mount, unmount and build ISO.

### 1.2.1 Linux package

For example, on debian, install theses paquages:

```
sudo apt-get install fuseiso isolinux xorriso virtualbox
```

**optional**: You can also install virtualbox gui:

```
sudo apt-get install virtualbox-qt
```

### 1.2.2 Python

A strongly advice you to use **virtualenv**.

Install virtualenv:

```
sudo apt-get install virtualenv

cd python-linux-iso/
virtualenv -p /usr/bin/python3 venv
source venv/bin/activate
pip install -r requirements.txt
deactivate
```

## 1.3 Getting started

Use configuration file, many of them are in **examples** directory. For example, use **examples/1_debian_simple/settings.yml**

**Change some parameters to avoid warning.:**

- general.dir_input: <directory where offical iso are (if empty, use default/dir_input directory)>
- general.dir_isocustom: <directory to put custom iso (if empty, use default/dir_isocustom directory)>
- general.dir_build: <temp directory where we build iso (if empty, use default/dir_build directory)>
- virutalbox.vms.myhostname.interface_name: <interface name used by virtualbox to connect the vm (if empty auto detect default interface)>

Now typical work flow is write in **examples/1_debian_simple/commands.sh**:

Download debian ISO:

```
cd examples/1_debian_simple
../../scripts/downloadcli --config settings.yaml --download debian-9.6.0-strech-amd64-
↪netinst.iso
```

Custom this debian iso with recipe:

```
../../scripts/customcli --config settings.yaml --create myhostname.iso
```

Deploy on virtualbox and run it:

```
../../scripts/virtualboxcli --config settings.yaml --create myhostname
../../scripts/virtualboxcli --config settings.yaml --run myhostname
```

You have more commands examples in **examples/1_debian_simple/commands.sh** and you can run it with:

```
cd example/1_debian_simple
./commands.sh
```

Or deploy on USB KEY (cf. documentation)

Or deploy on PXE (FUTURE DEVELOPPEMENT)

## 1.4 Architecture

**Life cycle:**

- Download
- Custom
- **Deploy**
  - on usb key
  - on localhost vitualbox
  - on localhost PXE (FUTURE DEVELOPPEMENT)

Project structure:

```
├── docs/            # Sphinx documentation deploy on **readthedoc.io**
├── examples/
├── linuxiso/        # Source code
    ├── __init__.py
    ├── __main__.py
    ├── conf/
    ├── download.py    # Download module part
    ├── custom/        # Custom module part
    ├── virtualbox.py  # Vituralbox module part
    ├── ressources     # Generique function (Ex: logging, load conf, ...)
    └── scripts        # Code for command line interface support for all modules
├── scripts/  # User entry point for command line interface for all modules
├── tests/    # Test (pytest+coverage) deployed on **travis-ci.org** and␣
↪**codeclimate.com**
│
├── README.rst
├── LICENSE.txt
├── requirements-dev.txt # Python dependencies for develop (build doc, run tests, ...)
├── requirements.txt     # Python dependencies for production
└── setup.py
```

## 1.5 Run unit test

First install developpement dependency:

```
pip install -r requirements-dev.txt
```

Secondly, execute all test using **pytest**:

```
pytest tests
```

## 1.6 Compile documentation

This documentation is generated with sphinx.

First install developpement dependency:

```
pip install -r requirements-dev.txt
```

Secondly, compile the documentation with sphinx:

```
cd docs
make html
```

The entry point of the documentation is in **docs/build/html/index.html**.

## 1.7 Compile distribution package (UNDERDEVELOPPEMENT)

Compile distribution package from source:

```
python setup.py sdist
```

The distribution package are in the **dist** directory

## 1.8 Run tests with coverage

The calcul of tests coverage is make with **pytest-cov**.

First install developpement dependency:

```
pip install -r requirements-dev.txt
```

Run tests with coverage:

```
py.test --cov=linuxiso tests
```

## 1.9 Links

Usefull link to understand Iso custumisation

Debian wiki for Raspbian: https://wiki.debian.org/RaspberryPi/qemu-user-static

Mount all kind of *.img: https://www.suse.com/c/accessing-file-systems-disk-block-image-files/

CHAPTER 2

# Download

**class** linuxiso.download.**Download**(*conf=None*)

Bases: object

Class manage download and verify iso.

**Parameters** **conf** (*dict*) – Configuration

You need to provide configuration with all info of iso managed.

**The typical use is:**

- chose a config that containt all info about iso
- **list** iso managed
- get the **status** of one or all iso (**status_all**)
- do operation on iso like **download**, **download_all**, **remove** or **remove_all**

```
>>> download = Download(conf)
>>> download.list()
>>> download.status("debian-9.6.0-strech-amd64-netinst.iso")
>>> download.download("debian-9.6.0-strech-amd64-netinst.iso")
```

**download**(*iso*)

Download one iso

**Parameters** **iso** (*str*) – Name of iso used

```
>>> download.download("debian-9.6.0-strech-amd64-netinst.iso")
```

**download_all**()

Download all iso

```
>>> download.download_all()
```

**list**()

Get list of iso managed.

**Returns** List of iso managed

**Return type** list

```
>>> download.list()
[
    "centos-7-x86-amd64-desktop.iso",
    "debian-10-buster-amd64-netinst-testing.iso",
    "debian-9.6.0-strech-amd64-netinst.iso",
    "kde-18.3-destop-kde.iso",
    "linuxmint-18.2-Sonya-amd64-desktop-cinnamon.iso",
    "raspbian-9-strech-lite.img",
    "ubuntu-16.04.4-LTS-Xenial_Xerus-amd64-desktop-live.iso",
    "ubuntu-16.04.4-LTS-Xenial_Xerus-amd64-server.iso",
    "ubuntu-17.10.1-Artful_Aardvark-amd64-server.iso"
]
```

**remove**(*iso*)

Remove one iso

**Parameters** **iso** (*str*) – Name of iso used

```
>>> download.remove("debian-9.6.0-strech-amd64-netinst.iso")
```

**remove_all**()

Remove all iso

```
>>> download.remove_all()
```

**status**(*iso*)

Get one iso status.

**Parameters** **iso** (*str*) – Name of iso used

**Returns** Status of the iso

**Return type** dict

**Test :**

- if the url to download the iso exist
- if the iso already downloaded
- if the checksum is good (if the iso is download)

```
>>> download.status("debian-9.6.0-strech-amd64-netinst.iso")
{
    "is_downloaded": true,
    "is_hash_valid": true,
    "is_url_exist": true
}
```

**status_all**()

Get all iso status.

**Returns** status of all iso managed

**Return type** dict

Test if the url to download exist, if the iso already downloaded and if the checksum is good

```
>>> download.status_all()
```

# CHAPTER 3

# Custom

**class** linuxiso.custom.core.**Custom**(*conf=None*)

Bases: object

Custom iso

**create**(*file_iso*)

Create custom iso/image from a other normal iso params file_iso : Name iso used

**list**()

Get list of cusotm iso.

**remove**(*iso*)

Delete custom iso/image from a other normal iso params file_iso : Name iso used

```
>>> custom.remove("Custom-FullAuto-Debian-9-strech-amd64-netinst-server.iso")
```

**remove_all**()

Remove all iso

```
>>> download.remove_all()
```

**static render**(*path_template_file*, *context*)

Simple function to use jinja2 template with file

**status**(*iso*)

Check custom iso/image status return : dict with status

**status_all**()

Check custom iso/image status return : dict with status

# CHAPTER 4

# Virtualbox

**class** linuxiso.virtualbox.**Virtualbox**(*conf=None*)

Bases: object

Class manage virtualbox with VBoxManage command

**The typical use is:**

- chose a config that containt all info about iso
- **list** iso managed
- get the **status** of one or all iso (**status_all**)
- do operation on iso like **download**, **download_all**, **remove** or **remove_all**

```
>>> virtualbox = Virtualbox(conf)
>>> virtualbox.list_vms()
>>> virtualbox.create(
...     hostname=hostname,
...     recipe='Debian-amd64-standard',
...     iso=./path/iso/debian.iso)
>>> virtualbox.run('Debian VM')
```

**create**(*vm_name*)

Create virtualbox vm >>> from linuxiso.virtualbox import Virtualbox >>> virtualbox = Virtualbox(conf) >>> virtualbox.run('Debian VM') >>> virtualbox.create( ... hostname=hostname, ... recipe='Debian-amd64-standard', ... iso=./path/iso/debian.iso)

**static get_machine_folder**()

Get machine folder

**static list_ostypes**()

Get list ostypes return l_ostypes : dict result

**static list_vms**()

Get list vms return l_vm : dict result

**remove**(*name_or_uid*)
Remove virtualbox vm

```
>>> virtualbox = Virtualbox(conf)
>>> virtualbox.remove('Debian VM')
```

**static run**(*hostname*)
Run existing vm

```
>>> virtualbox = Virtualbox(conf)
>>> virtualbox.run('Debian VM')
```

CHAPTER 5

# Download CLI

Program manage download of iso/image

```
usage: downloadcli [-h] [-f CONF_FILE]
                   [-l | -s ISO_NAME | -S | -d ISO_NAME | -a | -r ISO_NAME | -k]
                   [-v | -q]
```

## 5.1 Named Arguments

| | |
|---|---|
| **-f, --config-file** | load personnal configuration file (defaut: settings.yaml) |
| **-l, --list** | list iso managed |
| | Default: False |
| **-s, --status** | status of the iso/image |
| **-S, --status-all** | status of all iso/image |
| | Default: False |
| **-d, --download** | download the iso/image |
| **-a, --download-all** | download all iso/image |
| | Default: False |
| **-r, --remove** | remove the iso/image |
| **-k, --remove-all** | remove all iso/image |
| | Default: False |
| **-v, -vv, --verbose** | enable verbosity: -v = INFO, -vv = DEBUG |
| | Default: 0 |

**-q, --quiet** quiet mode

Default: False

Example of standard usage:

./downloadcli –config ../examples/0_debian_simple/settings.yaml –list ./downloadcli –status debian-9.6.0-strech-amd64-netinst.iso ./downloadcli –download debian-9.6.0-strech-amd64-netinst.iso

CHAPTER 6

# Custom CLI

Program custom iso/image

```
usage: custumcli [-h] [-f CONF_FILE]
                 [-l | -s ISO_NAME | -S | -r ISO_NAME | -k | -c ISO_NAME]
                 [-v | -q]
```

## 6.1 Named Arguments

| | |
|---|---|
| **-f, --config-file** | load personnal configuration file (defaut: settings.yaml) |
| **-l, --list** | list custom iso status |
| | Default: False |
| **-s, --status** | status of the custom iso/image |
| **-S, --status-all** | status of all custom iso/image |
| | Default: False |
| **-r, --remove** | remove the custom iso/image |
| **-k, --remove-all** | remove all custom iso/image |
| | Default: False |
| **-c, --create** | create custom iso/image |
| **-v, -vv, --verbose** | enable verbosity: -v = INFO, -vv = DEBUG |
| | Default: 0 |
| **-q, --quiet** | quiet mode |
| | Default: False |

Example of standard usage:

./customcli –list ./customcli –status Custom-FullAuto-Debian-9-strech-amd64-netinst-server.iso ./customcli –create Custom-FullAuto-Debian-9-strech-amd64-netinst-server.iso

–context context.yaml

CHAPTER 7

# Virtualbox CLI

Program manage virtualbox

```
usage: virtualboxcli [-h]
                     (-l | --list-ostypes | -c VM_NAME | -x VM_NAME | -r VM_NAME_OR_
↪UID)
                     [-f CONF_FILE] [-v | -q]
```

## 7.1 Named Arguments

| | |
|---|---|
| **-l, --list** | list curent VMs status |
| | Default: False |
| **--list-ostypes** | list os type supported by Virtualbox |
| | Default: False |
| **-c, --create** | create new VM (and mount an iso/image) |
| **-x, --run** | run virtualbox VM |
| **-r, --remove** | delete VM |
| **-f, --config-file** | load personnal configuration file (defaut: settings.yaml) |
| **-v, -vv, --verbose** | enable verbosity: -v = INFO, -vv = DEBUG |
| | Default: 0 |
| **-q, --quiet** | quiet mode |
| | Default: False |

Example of standard usage:

./virtualboxcli –list ./virtualboxcli –create myhostname

CHAPTER 8

# Indices and tables

- genindex
- modindex
- search

# Index