pyvenvwrapper Documentation

Release 1.0.0

Nikita Solovyev

Sep 16, 2018

Contents

CHAPTER 1

Contents

1.1 Introduction

pyvenvwrapper is a small and lightweight set of Bash script functions, that enhance the use of Python standard library venv module for management of virtual environments. Between Python 3.2 and Python 3.6 venv module was wrapped in a pyvenv script, that is now officially deprecated in favor of direct usage of venv module. pyvenvwrapper functions allow to create and manipulate virtual environments and corresponding projet folders in a convenient way using only their names. Additional feature is automatic activation/deactivation of virtual environment when changing current working directory in the shell. Since venv and virtualenv use similar technics for virtual environments, pyvenvwrapper can be used for both, though main aim is venv.

pyvenvwrapper can be used to manage virtual environments and corresponding project folders or only virtual environments. In former case it assumes that the same name is used for virtual environment folder and the project folder which uses this virtual environment. The directories containing this folders are configured using special variables.

The idea to create pyvenvwrapper is inspired by using virtualenvwrapper, which at that moment didn't have support for pyvenv and venv virtual environment management. *pyvenvwrapper code is in no way related to virtualenvwrapper code.*

1.1.1 Compatibility

pyvenvwrapper functions are written and tested for Bash shell, however they might work with other Bash-like shells. pyvenvwrapper is pure shell script with calls to common system tools, so it doesn't care much on what Python version is used, therefore it should work with any Python 2 or Python 3 version. Some features will require '*pip*'. pyvenvwrapper originally was intended to be used with '*pyvenv*' tool ('*venv*' module in Python standard library), but it supports '*virtualenv*' tool too.

1.1.2 Support

Any questions or issues can be reported via GitHub Issues.

1.1.3 Changelog

1.0.0

- Added support for creation of virtual environments using venv module (*python -m venv*). It is now preferred by default with a fallback to *pyvenv* and *virtualenv*. Using *venv* directly is officially recommended for creating virtual environments since Python 3.5. *pyvenv* script is officially deprecated in Python 3.6.
- Added new option *-b/–python* to *mkvenv* command, that allows to specify which Python interpreter executable to use for new virtual environment. This option works with *venv* and *virtualenv*.

0.1.0

- Initial version with all the main features.

1.1.4 License

1.2 Installation

By defautl the following instructions show how to enable pyvenvwrapper for particular user, if you wish to enable it system wide, see *Manual installation*.

1.2.1 Automated installation

To install **pyvenvwrapper**:

1. Run '*pip install pyvenvwrapper*', this will download and install required files on your machine.
2. Run '*pyvenvwrapper_enable*', this will enable pyvenvwrapper for current user by adding the following lines to user's *.bashrc* file:

```
source [path_to_pyvenvwrapper]/pyvenvwrapper_settings
source [path_to_pyvenvwrapper]/pyvenvwrapper
```

There's also '*pyvenvwrapper_disable*' command, which disables pyvenvwrapper for current user by removing those lines.

3. Reboot your shell or run '*source ~/.bashrc*'.
4. Run '*pyvenvwrapper*' to see available commands and start using **pyvenvwrapper** or see *Settings* to customize its behavior first.

1.2.2 Manual installation

To install **pyvenvwrapper** manually:

1. Run '*pip install pyvenvwrapper*', this will download and install required files on your machine.
2. Find where *pyvenvwrapper* package is installed. Usually somewhere in *site-packages* or *dist-packages*, i.e. */usr/lib/python3/site-packages/*, */usr/local/lib/python2.7/site-packages/*.
3. Open current user's *.bashrc* file in text editor, i.e. '*vim ~/.bashrc*', and add the following lines to the end of the file, substituting *[path_to_pyvenvwrapper_package]* with actual path from step 2:

```
source [path_to_pyvenvwrapper_package]/pyvenvwrapper_settings
source [path_to_pyvenvwrapper_package]/pyvenvwrapper
```

   If you wish to enable pyvenvwrapper system wide, then consider adding the lines above to the end of */etc/bash.bashrc* file, or adding symlinks for specified files to */etc/profile.d/* directory.

4. Reboot your shell or run '*source ~/.bashrc*'.
5. Run '*pyvenvwrapper*' to see available commands and start using **pyvenvwrapper** or see *Settings* to customize its behavior first.

1.3 Settings

The following settings are defined in *pyvenvwrapper_settings* file in *pyvenvwrapper* package directory, which is sourced in user's *.bashrc*. The settings have sane defaults, but can be redefined directly in *pyvenvwrapper_settings* or in the end of user's *.bashrc* file. For changes to take effect the shell has to be rebooted or user's *.bachrc* has to be sourced by running '*source ~/.bashrc*'.

PYVENVWRAPPER_ENV_DIR (=*~/.virtualenvs*) Directory to keep virtual environments. No symlinks allowed. **The only setting that must be defined in order to make pyvenvwrapper work.**

PYVENVWRAPPER_PROJ_DIR (=*~/projects*) Directory to keep project folders. No symlinks allowed. If this setting is undefined, then pyvenvwrapper will silently not perform any actions, that assume existence of project folders related to virtual environments. Therefore not defining this option makes pyvenvwrapper work only with virtual environments. However if any command is called with explicit option related to project folder when this option is undefined, the command will be aborted with error.

PYVENVWRAPPER_CD_ON_WORKON (=*true*) Enables/Disables directory change to corresponding project directory after virtual environment activation with workon command. Possible values: true/false. Requires *PYVENVWRAPPER_PROJ_DIR* to be set in order to work.

PYVENVWRAPPER_CD_ON_DEACT (=*true*) Enables/Disables directory change to the one used at the time of workon execution after virtual environment deactivation with deact call. Possible values: true/false.

PYVENVWRAPPER_ACTIVATE_ON_CD (=*true*) Enables/Disables redefinition of *cd*, *popd*, *pushd* commands in oreder to activate virtual environment if directory changed to one of virtual environments' or corresponding projects' directory, otherwise do nothing or deactivate active virtual environment. Possible values: true/false. Requires shell reboot after changing or sourcing user's *.bashrc*.

Note on *PYVENVWRAPPER_ACTIVATE_ON_CD*: redefinition of commands is intended to be transparent, so argumetns of original built-in functions are not affected in any way, return value are always that of wrapped built-in and no additional output related to added behavior is introduced.

1.4 Command reference

Usage and possible options for each command can be displayed in the shell by calling a command with *-h* or *–help* option.

All commands support auto-completion of virtual environment names.

All commands return:

- '*0*' exit code on successful execution;
- '*1*' exit code when an error occurres;
- '*2*' exit code on invocation syntax errors.

1.4.1 mkvenv

mkvenv command is a wrapper for venv/pyvenv/virtualenv and pip install

Usage: mkvenv [OPTIONS] VENV_NAME

mkvenv command creates new virtual environment with the name of VENV_NAME in directory specified by PYVENVWRAPPER_ENV_DIR and new project directory with the same name in directoy specified by PYVENVWRAPPER_PROJ_DIR, if this variable is set. Additional options, that modify this command's behavior are described below.

Mandatory arguments to long options are mandatory for short options too. Combined options are not supported, i.e. instead of '-aj' use '-a -j'.

-o, --options <options> Options to provide to underlying tool for virtual environment creation. See additional information below.

-i, --install <requirements> Install packages listed in requirements using pip after virtual environment is created. <requirements> should be quoted string in "pip install" requirement specifier format. mkvenv will automatically try to install pip if it isn't already available.

-r, --requirements <file> Install packages listed in requirements file using pip after virtual environment is created. <file> should be path pointing to a file containing requirement specifications in "pip install -r" requirements file format. mkvenv will automatically try to install pip if it isn't already available.

-u, --util <util name> Specify the name of utility to use for virtual environment creation. By default mkvenv tries to use "venv" first, if it's not available mkvenv tries to use "pyvenv" and then "virtualenv".

-b, --python <python executable> Use provided Python executable for new virtual environment. <python executable> should be a path pointing to a Python interpreter executable file. Works only with "venv" and "virtualenv". By default the system's "python" is used.

-p, --pip Install pip after virtual environment is created.

-t, --template <template dir path> Copy files and directories from template directory to newly created project directory. Precludes use of -n option.

-n, --no-project	Don't create project directory. Precludes use of -t, -j options.
-a, --activate	Activate virtual environment after it is created.
-e, --env	Change current directory to virtual environment directory after it is created. Precludes use of -j option.
-j, --project	Change current directory to project directory after it is created. Precludes use of -n, -e options.

1.4.2 workon

Usage: workon [-n] VENV_NAME

workon command is a wrapper for VIRTUAL_ENV/bin/activate

workon command activates existing virtual environment with the name of VENV_NAME from directory specified by PYVENVWRAPPER_ENV_DIR, and changes current working directory to corresponding project directory if PYVENVWRAPPER_PROJ_DIR is specified and PYVENVWRAPPER_CD_ON_WORKON is set to "true".

-n, --no-cd	Don't change current working directory to corresponding project directory after virtual environment activation.

1.4.3 deact

Usage: deact

deact command is a wrapper for deactivate

deact command deactivates active virtual environment, and changes current working directory back to its value at the time of virtual environment activation if PYVENVWRAPPER_CD_ON_DEACT is set to "true".

1.4.4 lsvenv

Usage: lsvevn [OPTIONS] [VENV_NAME]

lsvenv command list existing virtual environments in the directory specified by PYVENVWRAPPER_ENV_DIR. If used with existing virtual environment name as optional argument VENV_NAME, then lsvenv lists packages installed in this virtual environment in requirements format (alias to "pip freeze"). Additional options, that modify this command's behavior are described below.

Combined options are not supported, i.e. instead of '-se' use '-s -e'.

-l, --local	If virtual environment has global access, do not list globally-installed packages. Has no meaning if VENV_NAME is not provided.
-s, --simple	Use simple output format instead of requirements format (alias to "pip list"). Has no meaning if VENV_NAME is not provided.
-e, --extended	Show additional information.

1.4.5 cdvenv

Usage: cdvenv [OPTIONS] VENV_NAME

cdvenv command changes current working directory to directory of virtual environment specified by VENV_NAME argument. Additional options, that modify this command's behavior are described below.

-s, --site	Change current working directory to virtual environment's site-packages directory instead. Precludes use of -p option.
-p, --project	Change current working directory to virtual environment's related project directory instead. Precludes use of -s option.

1.4.6 cpvenv

Usage: cpvenv [OPTIONS] SRC_VENV_NAME DST_VENV_NAME

cpvenv command creates a copy of virtual environment. It copies all contents of SRC_VENV_NAME virtual environment directory to a new directory for virtual environment with the name specified by DST_VENV_NAME. If PYVENVWRAPPER_PROJ_DIR is set, cpvenv also creates a new project directory related to new virtual environment with DST_VENV_NAME. cpvenv will not overwrite any existing data in DST_VENV_NAME virtual environment directory (and related project directory) if it already exists and is not empty, unless -f option is provided. Additional options, that modify this command's behavior are described below.

Note: Depending on the name of source virtual environment destination virtual environment might be broken after copy. This is due to renaming in destination virtual environment which has to take place because of how virtual environments work. Source virtual environment will not be affected in any way. This should normally not happen if the name is unique and not anything more generic like simple "if", "var", etc..

Combined options are not supported, i.e. instead of '-fp' use '-f -p'.

-f, --force	Overwrite data in DST_VENV_NAME virtual environment directory (and related project directory) if it already exists and is not empty.
-p, --project	Copy contents of project directory related to SRC_VENV_NAME virtual environment to new project directory related to DST_VENV_NAME virtual environment. Precludes use of -n option.
-n, --no-project	Don't create project directory. Precludes use of -p option.

1.4.7 rmvenv

Usage: rmvenv [OPTIONS] VENV_NAME

rmvenv command removes virtual environment directory with the name specified by VENV_NAME. Additional options, that modify this command's behavior are described below.

Combined options are not supported, i.e. instead of '-fp' use '-f -p'.

Be cautious when using options!

-f, --force	Don't prompt for any confirmations.
-p, --project	Also remove related project directory with all contents.

1.5 Hooks

If there's a need for added behavior on any command execution, it can be provided via custom scripts, that can be assigned to the hook variables. The script provided will be sourced, which means that its commands will be called in the same process and any changes, ie. directory changes, global variables, will be kept in current shell session after sourcing. There're hooks that will be sourced before and after each command.

Custom hook script will be sourced:

- for **PRE** command - before any actions are taken, but after command line options and arguments are parsed and verified;
- for **POST** command - after all actions are taken, as last instructions, but only if no errors occured.

For convenience every script defined for hook variables will get "*venv=VENV_NAME*" as first argument and all the arguments from command line as subsequent arguments. Special cases are:

- **LSVENV** might be called without *VENV_NAME*, in this case "*venv=*" will be provided;
- **CPVENV** will get "*venv=SRC_VENV*" and "*dst=DST_VENV*" as first and second arguments and all the arguments from command line as subsequent arguments;
- **DEACT** will not get any arguments, as it doesn't use any. (Active virtual evironment path is kept in *VIRTUAL_ENV* environment variable, so it can be used.)

VENV_NAME, *SRC_VENV*, *DST_VENV* will be the actual virtual environments names provided as argument to corresponding command.

Custom script should return '*0*' in the end if no errors occured. If the sourced script will return any return code other than '*0*', then the command will be aborted with error.

Provide a path to a custom script file as a value for the following variables directly in *pyvenvwrapper_settings* in *pyvenvwrapper* package or in the end of user's *.bashrc* file to define hooks (i.e. *PYVENVWRAPPER_POST_MKVENV=~/custom_sript*). Fot changes to take effect you'll have to reboot the shell or run '*source ~/.bashrc*'.

- **Sourced before and after *mkvenv*: PYVENVWRAPPER_PRE_MKVENV PYVENVWRAPPER_POST_MKVENV**
- **Sourced before and after *lsvenv*: PYVENVWRAPPER_PRE_LSVENV PYVENVWRAPPER_POST_LSVENV**
- **Sourced before and after *cdvenv*: PYVENVWRAPPER_PRE_CDVENV PYVENVWRAPPER_POST_CDVENV**
- **Sourced before and after *rmvenv*: PYVENVWRAPPER_PRE_RMVENV PYVENVWRAPPER_POST_RMVENV**
- **Sourced before and after *cpvenv*: PYVENVWRAPPER_PRE_CPVENV PYVENVWRAPPER_POST_CPVENV**
- **Sourced before and after *workon*: PYVENVWRAPPER_PRE_WORKON PYVENVWRAPPER_POST_WORKON**
- **Sourced before and after *deact*: PYVENVWRAPPER_PRE_DEACT PYVENVWRAPPER_POST_DEACT**
- **Sourced before and after virtual environment activation on directory change if *PYVENVWRAPPER_ACT_ON_CD* setting PYVENVWRAPPER_PRE_ACT_ON_CD PYVENVWRAPPER_POST_ACT_ON_CD**

Note for *PYVENVWRAPPER_PRE_ACT_ON_CD* and *PYVENVWRAPPER_POST_ACT_ON_CD*: If *cd* to directory not related in any way to any virtual environment, hooks are not called. If *cd* to directory related to virtual environment, even if there's any already active virtual environment, the *PRE* hook will be source before currently active environment deactivation. For these hook scripts any output to console will be suppresed.