fades Documentation

Release 4

Facundo Batista, Nicolás Demarchi

March 01, 2016

Contents

CHAPTER 1

Contents:

1.1 fades

fades is a system that automatically handles the virtualenvs in the cases normally found when writing scripts and simple programs, and even helps to administer big projects.

1.1.1 What does it do?

fades will automagically create a new virtualenv (or reuse a previous created one), installing the necessary dependencies, and execute your script inside that virtualenv, with the only requirement of executing the script with *fades* and also marking the required dependencies.

(If you don't have a clue why this is necessary or useful, I'd recommend you to read this small text about Python and the Management of Dependencies *.)*

The first non-option parameter (if any) would be then the child program to execute, and any other parameters after that are passed as is to that child script.

fades can also be executed without passing a child script to execute: in this mode it will open a Python interactive interpreter inside the created/reused virtualenv (taking dependencies from `--dependency` or `--requirement` options).

1.2 How to use it?

When you write an script, you have to take two special measures:

- need to execute it with *fades* (not *python*)
- need to mark those dependencies

At the moment you execute the script, fades will search a virtualenv with the marked dependencies, if it doesn't exists fades will create it, and execute the script in that environment.

1.2.1 How to execute the script with fades?

You can always call your script directly with fades:

```
fades myscript.py
```

However, for you to not forget about fades and to not execute it directly with python, it's better if you put at the beggining of the script the indication for the operating system that it should be executed with fades...

```
#!/usr/bin/fades
```

...and also set the executable bit in the script:

```
chmod +x yourscript.py
```

1.2.2 How to mark the dependencies to be installed?

The procedure to mark a module imported by the script as a *dependency to be installed by fades* is by using a comment.

This comment will normally be in the same line of the import (recommended, less confusing and less error prone in the future), but it also can be in the previous one.

The simplest comment is like:

```
import somemodule    # fades.pypi
from somepackage import othermodule     # fades.pypi
```

The `fades.pypi` is mandatory, it may allow more options in the future.

With that comment, *fades* will install automatically in the virtualenv the `somemodule` or `somepackage` from PyPI.

Also, you can indicate a particular version condition, examples:

```
import somemodule    # fades.pypi == 3
import somemodule    # fades.pypi >= 2.1
import somemodule    # fades.pypi >=2.1,<2.8,!=2.6.5
```

Sometimes, the project itself doesn't match the name of the module; in these cases you can specify the project name (optionally, before the version):

```
import bs4    # fades.pypi beautifulsoup4
import bs4    # fades.pypi beautifulsoup4 == 4.2
```

1.2.3 Other ways to specify dependencies

Apart of marking the imports in the source file, there are other ways to tell *fades* which dependencies to install in the virtualenv.

One way is through command line, passing the `--dependency` parameter. This option can be specified multiple times (once per dependency), and each time the format is `repository::dependency`. The dependency may have versions specifications, and the repository is optional (defaults to 'pypi').

Other way is to specify the dependencies in a text file, one dependency per line, with each line having the format previously described for the `--dependency` parameter. This file is then indicated to fades through the `--requirement` parameter.

In case of multiple definitions of the same dependency, command line overrides everything else, and requirements file overrides what is specified in the source code.

1.2.4 How to control the virtualenv creation and usage?

You can influence several details of all the virtualenv related process.

The most important detail is which version of Python will be used in the virtualenv. Of course, the corresponding version of Python needs to be installed in your system, but you can control exactly which one to use.

No matter which way you're executing the script (see above), you can pass a `-p` or `--python` argument, indicating the Python version to be used just with the number (`2.7`), the whole name (`python2.7`) or the whole path (`/usr/bin/python2.7`).

Other detail is the verbosity of *fades* when telling what is doing. By default, *fades* only will use stderr to tell if a virtualenv is being created, and to let the user know that is doing an operation that requires an active network connection (e.g. installing a new dependency).

If you call *fades* with `-v` or `--verbose`, it will send all internal debugging lines to stderr, which may be very useful if any problem arises. On the other hand if you pass the `-q` or `--quiet` parameter, *fades* will not show anything (unless it has a real problem), so the original script stderr is not polluted at all.

Sometimes, you want to run a script provided by one of the dependencies installed into the virtualenv. *fades* supports this via the `-x` (or `--exec` argument).

If you want to use IPython shell you need to call *fades* with `-i` or `--ipython` option. This option will add IPython as a dependency to *fades* and it will launch this shell instead of the python one.

You can also use `--system-site-packages` to create a venv with access to the system libs.

1.2.5 How to deal with packages that are upgraded in PyPI

When you tell *fades* to create a virtualenv using one dependency and don't specify a version, it will install the latest one from PyPI.

For example, you do `fades -d foobar` and it installs foobar in version 7. At some point, there is a new version of foobar in PyPI, version 8, but if do `fades -d foobar` it will just reuse previously created virtualenv, with version 7, not using the new one!

You can tell fades to do otherwise, just do:

```
fades -d foobar --check-updates
```

...and *fades* will search updates for the package on PyPI, and as it will found version 8, will create a new virtualenv using the latest version.

You can even use this parameter when specifying the package version. Say you call `fades -d foobar==7`, *fades* will install version 7 no matter which one is the latest. But if you do:

```
fades -d foobar==7 --check-updates
```

...it will still use version 7, but will inform you that a new version is available!

1.2.6 Under the hood options

For particular use cases you can send specifics arguments to `virtualenv` or `pip`. using the `--virtuaenv-options` and `--pip-options`. You have to use that argument for each argument sent.

Examples:

```
fades -d requests --virtualenv-options="--always-copy" --virtualenv-options="--extra-search
```

```
fades -d requests --pip-options="--index-url="http://example.com"
```

1.2.7 Setting options using config files

You can also configure fades using *.ini* config files. fades will search config files in */etc/fades/fades.ini*, the path indicated by *xdg* for your system (for example *~/config/fades/fades.ini*) and *.fades.ini*.

So you can have different settings at system, user and project level.

With fades installed you can get your config dir running:

```
python -c "from fades.helpers import get_confdir; print(get_confdir())"
```

The config files are in *.ini* format. (configparser) and fades will search for a *[fades]* section.

You have to use the same configurations that in the CLI. The only difference is with the config options with a dash, it has to be replaced with a underscore.:

```
[fades]
ipython=true
verbose=true
python=python3
check_updates=true
dependency=requests;django>=1.8  # separated by semicolon
```

There is a little difference in how fades handle these settings: "dependency", "pip-options" and "virtualenv-options". In these cases you have to use a semicolon separated list.

The most important thing is that these options will be merged. So if you configure in */etc/fades/fades.ini* "dependency=requests" you will have requests in all the virtualenvs created by fades.

1.2.8 How to clean up old virtualenvs?

When using *fades* virtual environments are something you should not have to think about. *fades* will do the right thing and create a new virtualenv that matches the required dependencies. There are cases however when you'll want to do some clean up to remove unnecessary virtual environments from disk.

By running *fades* with the `--rm` argument, *fades* will remove the virtualenv matching the provided uuid if such a virtualenv exists.

1.2.9 Some command line examples

`fades foo.py --bar`

Executes `foo.py` under *fades*, passing the `--bar` parameter to the child program, in a virtualenv with the dependencies indicated in the source code.

`fades -v foo.py`

Executes `foo.py` under *fades*, showing all the *fades* messages (verbose mode).

`fades -d dependency1 -d dependency2>3.2 foo.py --bar`

Executes `foo.py` under *fades* (passing the `--bar` parameter to it), in a virtualenv with the dependencies indicated in the source code and also `dependency1` and `dependency2` (any version > 3.2).

`fades -d dependency1`

Executes the Python interactive interpreter in a virtualenv with `dependency1` installed.

`fades -r requirements.txt`

Executes the Python interactive interpreter in a virtualenv after installing there all dependencies taken from the `requirements.txt` file.

```
fades -d django -x django-admin.py startproject foo
```

Uses the `django-admin.py` script to start a new project named `foo`, without having to have django previously installed.

```
fades --rm 89a2bf83-c280-4918-a78d-c35506efd69d
```

Removes a virtualenv matching the given uuid from disk and cache index.

1.3 How to install it

Several instructions to install `fades` in different platforms.

1.3.1 Simplest way

In some systems you can install `fades` directly, no needing to install previously any dependency.

If you are in debian unstable or testing, just do:

    sudo apt-get install fades

For Arch linux:

    yaourt -S fades

Else, keep reading to know how to install the dependencies first, and `fades` in your system next.

1.3.2 Dependencies

Fades depends on the `pkg_resources` package, that comes in with `setuptools`. It's installed almost everywhere, but in any case, you can install it in Ubuntu/Debian with:

```
apt-get install python3-setuptools
```

And on Archlinux with:

```
pacman -S python-setuptools
```

It also depends on `python-xdg` package. This package should be installed on any GNU/Linux OS wiht a freedesktop.org GUI. However it is an **optional** dependency.

You can install it in Ubuntu/Debian with:

```
apt-get install python3-xdg
```

And on Archlinux with:

```
pacman -S python-xdg
```

Fades also needs the *virtualenv* <*https://virtualenv.pypa.io/en/latest/*> package to support different Python versions for child execution. (see *–python* argument.)

1.3.3 For others debian and ubuntu

If you are NOT in debian unstable or testing (if you are, see above for better instructions), you can use this .deb.

Download it and install doing:

```
sudo dpkg -i fades-latest.deb
```

1.3.4 Using pip if you want

```
pip3 install fades
```

1.3.5 Multiplatform tarball

Finally you can always get the multiplatform tarball and install it in the old fashion way:

```
wget http://taniquetil.com.ar/fades/fades-latest.tar.gz
tar -xf fades-latest.tar.gz
cd fades-*
sudo ./setup.py install
```

1.3.6 Can I try it without installing it?

Yes! Branch the project and use the executable:

```
git clone https://github.com/PyAr/fades.git
cd fades
bin/fades your_script.py
```

1.4 Get some help, give some feedback

You can ask any question or send any recommendation or request to the mailing list.

Come chat with us on IRC. The #fades channel is located at the Freenode network.

Also, you can open an issue here (please do if you find any problem!).

Thanks in advance for your time.

1.5 Development

See the documentation for detailed instructions about how to setup everything and develop fades.

1.6 How to develop fades itself

Quick guide to get you up and running in fades development.

1.6.1 Getting the code

Clone the project:

```
git clone git@github.com:PyAr/fades.git
```

1.6.2 Install dependencies

fades manages it's own dependencies, so there is nothing extra you need to install.

1.6.3 How to run the tests

When starting development, at all times, and specially before wrapping up a new branch, you need to be sure that all tests pass ok.

This is very simple, actually, just run:

```
./test
```

That will not only check test cases, but also that the code complies with aesthetic recommendations, and that the README document has a proper format.

If you want to run *one* particular test, just specify it. Example:

```
./test tests.test_main:DepsMergingTestCase.test_two_different
```

1.6.4 Development process

Just pick an issue from the list.

Develop, assure `./test` is happy, commit, push, create a pull request, etc.

Please, if you aim for creating a Pull Request with new code (functionality or fixes), include tests for your changes.

Thanks! Enjoy.

CHAPTER 2

Indices and tables

- genindex
- modindex
- search