VersionClimber Documentation

Release 1.1.0

Christophe Pradal, Dennis Shasha

May 08, 2023

Contents

Authors : Christophe Pradal, Sarah Cohen-Boulakia, Patrick Valduriez, Dennis Shasha

Institutes : CIRAD / INRIA / NYU

Status : Python package

License : Cecill-C

Release 1.1.0

Date May 08, 2023

CHAPTER 1

About

1.1 Description

Version Climber is a system primarily aimed at Data Scientists to make package evolution efficient and automatic.

VersionClimber will help you upgrade a multi-package software system to a configuration that works.

CHAPTER 2

Installation

2.1 Installation

2.1.1 Installation with Miniconda

Conda installation

You should already have installed Miniconda or Anaconda for Python 2.7.

Using conda on Linux, Mac or Windows

Create virtual environment and activate it

```
conda create --name vclimber python
source activate vclimber
```

Version Climber install

```
conda install -c versionclimber versionclimber
```

2.1.2 Installation with pip

```
pip install git+https://github.com/VersionClimber/VersionClimber
```

2.1.3 Installation from source code

Source code Requirements (conda env):

```
conda install path.py gitpython pyyaml python six requests pyzmq
```

or (in a virtualenv):

```
pip install path.py gitpython pyyaml
```

Install from source VersionClimber:

```
python setup.py install
```

CHAPTER 3

Tutorials

3.1 Tutorials

Release 1.1.0

Date May 08, 2023

3.1.1 Case Study 1: two python packages from github

This case study is defined here. To run this case study, first create a new conda environment, and install VersionClimber:

```
conda create -n tutorial_usecase1 python=2 -y
source activate tutorial_usecase1
conda install versionclimber -c versionclimber -y
```

We install also some dependencies we want to fix, such as NumPy, SciPy, Cython and six:

```
conda install numpy scipy cython six -y
```

Now you are ready to define a configuration file for VersionClimber in a directory.

Classical layout of a project

To reproduce an execution by using VersionClimber, you will create a directory containing two files. Let's name this directory tutorial. It will contain:

- config.yaml: the VersionClimber configuration file
- test_function.py: a executable script (python file or another script) to test the validity of one configuration

Definition of a simple configuration file

VersionClimber uses the declarative configuration file to indicate which packages have to be tested and how.

In this section you are going to define a configuration file that uses two well-knowned scientific Python packages, namely Scikit-Learn and Scikit-Image.

The configuration file config.yaml is as follows (in this example, scikit-image has a higher priority than scikit-learn so scikit-image is first):

```
packages:
    - name      : scikit-image
      vcs       : git
      url       : https://github.com/scikit-image/scikit-image
      cmd       : pip install --no-index --no-deps -U
      version   : v0.11.0
      hierarchy : patch
      directory : .vclimb

    - name      : scikit-learn
      vcs       : git
      url       : https://github.com/scikit-learn/scikit-learn
      cmd       : pip install --no-index --no-deps -U
      version   : 0.16.0
      hierarchy : patch

run:
    - python test_function.py
```

It is divided into two sections, namely **packages** and **run**:

- **packages:** list the different packages, their location (e.g. git repository), how to build them and which git commit or tags will be considered (in hierarchy, as explained below).
- **run:** indicate how to test the different packages together to know if one combination is valid. Typically (as in this example), this will be the name of a driver file.

Packages

The *packages* section list the different packages that will be tested by the run command:

- **name** is the name of the package
- **vcs** define which type of version control system the package use (i.e. git or svn).

If we want to consider binary packages rather than source one, we can define a package repository (i.e. pypi or conda)

- **url** is the address where the package will be cloned or checkout
- **cmd** is the command to build the package
- **conda** is an optional argument to indicate if the package is managed by conda (*True*) or pip (*False*)
- **recipe** is the local path where the conda recipe is defined
- **channels** is a list of priority channels to consider when installing with conda
- **hierarchy** is the strategy use to select the different versions of the package from the *vcs*.

If *hierarchy* is *major*, *minor*, or *patch*, the versions of the tags will be selected for that indentation level and higher. Otherwise, (*commit*) all the commits of the origin or master branch will be tested by VersionClimber. In this example, because minor packages are of the form x.y, VersionClimber will take the most recent patch associated with each x.y. So, if a package is identified as 5.4.3 and there is no higher patch number among the patches that begin with 5.4, then VersionClimber will select 5.4.3.

Run command in *config.yaml*

This is the script (usually) after run: in that file

This script (*test_function.py*) extract HOG features of each digits of the MNIST database of handwritten digits using scikit-image and train a Linear SVM classifier to recognise hand-written digits.

Invocation of VersionClimber

vclimb – will fetch the packages from git, retrieve all the versions, install each configuration (set of package-version pairs) suggested by the Version Climber software, then invoke the run part of the config.yaml on that installed configuration. The output is configuration that works sorted based on the priorities in config.yaml

If we want to vary all the main dependencies of **scikit-learn** and **scikit-image**, we can extend the config.yaml file with other packages obtained from PyPi:

```
packages:
    - name      : scikit-image
      vcs       : git
      url       : https://github.com/scikit-image/scikit-image
      cmd       : pip install --no-index --no-deps -U
      version   : v0.11.0
      hierarchy : patch
      directory : .vclimb

    - name      : scikit-learn
      vcs       : git
      url       : https://github.com/scikit-learn/scikit-learn
      cmd       : pip install --no-index --no-deps -U
      version   : 0.16.0
      hierarchy : patch

    - name      : scipy
      vcs       : pypi
      version   : 0.13.0
      hierarchy : minor

    - name      : numpy
      vcs       : pypi
      version   : 0.9.6
      hierarchy : minor

run:
    - python test_function.py
```

All the minor versions of numpy and scipy will be considered (0.19, 0.18, ...). In this case, if wheels are availables, they will be installed in priority (thanks to pip).

3.1.2 Case Study 2: Two binary packages available from conda

In this example, we consider the same packages that in the previous case study (i.e. scikit-learn and scikit-image), but conda binary versions of the packages will be assembled rather than building the packages from github.

```
packages:
    - name      : scikit-image
      vcs       : conda
      cmd       : conda install -y
      channels  :
        - conda-forge
      hierarchy : patch

    - name      : scikit-learn
      vcs       : conda
      cmd       : conda install -y
      channels  :
        - conda-forge
      hierarchy : patch

run:
    - python test_function.py
```

In this example, the set of versions of each package is retrieved from anaconda default channel and the conda-forge (ref TODO) one. You can explore the available versions using the command

```
vclimb -v

--------------------------------------------------------------------------------
Versions of scikit-image scikit-learn


Versions of  scikit-image
-------------------------
0.7.2
0.8.0
0.8.2
0.9.1
0.9.3
0.10.0
0.10.1
0.11.0
0.11.2
0.11.3
0.12.3
0.13.0


Versions of  scikit-learn
-------------------------
0.11
0.12.1
0.13
0.13.1
0.14.1
0.15.0
0.15.0b1
0.15.0b2
```

(continues on next page)

(continued from previous page)

```
0.15.1
0.15.2
0.16.0
0.16.1
0.17
0.17.1
0.18
0.18.1
0.18.2
```

As in the previous case study, we can extend the configuration file by adding numpy and scipy packages, but installed from conda.

```
packages:
    - name      : scikit-image
      vcs       : conda
      cmd       : conda install -y
      channels  :
        - conda-forge
      hierarchy : patch

    - name      : scikit-learn
      vcs       : conda
      cmd       : conda install -y
      channels  :
        - conda-forge
      hierarchy : patch

    - name      : scipy
      vcs       : conda
      cmd       : conda install -y
      channels  :
        - conda-forge
      hierarchy : minor

    - name      : numpy
      vcs       : conda
      cmd       : conda install -y
      channels  :
        - conda-forge
      hierarchy : minor

run:
    - python test_function.py
```

3.1.3 Case Study 3: OpenAlea

In this case study, we want to find a valid configuration of various packages from OpenAlea, a scientific project developed to study multiscale plant modeling.

Packages in OpenAlea are implemented in different languages (mainly, C++, Python and R). First, we will consider PlantGL (ref TODO), a large 3D C++ library with various dependencies. Then we will explore an example obtained from the combina

What happens?

- First, the different packages are checked out in the folder `.vclimb`
- Then, all the package versions are retrieved from git, PyPi or svn
- The cmd (run) is tested on several configurations (combinations of packages)
- The log is written in a file names versionclimber.log

CHAPTER 4

User Documentation

4.1 User Documentation

Write here the documentation.

CHAPTER 5

Developper Documentation

5.1 References

Release 1.1.0

Date May 08, 2023

Version Climber is a system for package evolution for Data Science

5.1.1 API Reference

The exact API of all functions and classes, as given by the docstrings. The API documents expected types and allowed features for all functions, and all parameters available for the algorithms.

Configuration

`Package`
`load_config`

Utilitaries

`pypi_versions`
`conda_versions`
`git_versions`
`svn_versions`
`sh`

Environment

MyEnv
YAMLEnv

CHAPTER 6

Indices and tables

- genindex
- search