border$patrol Documentation$

Release unknown

Florian Wilhelm

Jul 23, 2022

CONTENTS

Border-Patrol logs all imported packages and their version to support you while debugging. In 95% of all cases when something suddenly breaks in production it is due to some different version in one of your requirements. Pinning down the versions of all your dependencies and dependencies of dependencies inside a virtual environment helps you to overcome this problem but is quite cumbersome and thus this method is not always applied in practice. Also sometimes, like when you are using PySpark, you might not be 100% sure which library versions are installed on some cluster nodes.

With Border-Patrol you can easily find the culprit by looking in the logs of the last working version and compare it to the failing one since Border-Patrol will list all imported packages and their corresponding version right at the end of your application, even if it crashed.

CHAPTER

ONE

USAGE

Border-Patrol is really simple to use, just install it with `pip install border-patrol` and import it before any other package, e.g.:

```
from border_patrol import with_print_stdout

import pandas as pd
```

If you run those lines in a script, you will get a similar output to this one:

```
Python version is 3.6.7 |Anaconda, Inc.| (default, Oct 23 2018, 14:01:38)
[GCC 4.2.1 Compatible Clang 4.0.1 (tags/RELEASE_401/final)]
Following packages were imported:
PACKAGE         VERSION   PATH
border_patrol   0.1       /Users/fwilhelm/Sources/border_patrol/src/border_patrol
cycler          0.10.0    /Users/fwilhelm/anaconda/envs/lib/python3.6/site-packages/
↪cycler.py
dateutil        2.7.5     /Users/fwilhelm/anaconda/envs/lib/python3.6/site-packages/
↪dateutil/__init__.py
matplotlib      2.2.3     /Users/fwilhelm/anaconda/envs/lib/python3.6/site-packages/
↪matplotlib/__init__.py
numpy           1.15.1    /Users/fwilhelm/anaconda/envs/lib/python3.6/site-packages/
↪numpy/__init__.py
pandas          0.23.4    /Users/fwilhelm/anaconda/envs/lib/python3.6/site-packages/
↪pandas/__init__.py
pyparsing       2.3.0     /Users/fwilhelm/anaconda/envs/lib/python3.6/site-packages/
↪pyparsing.py
pytz            2018.7    /Users/fwilhelm/anaconda/envs/lib/python3.6/site-packages/pytz/
↪__init__.py
six             1.11.0    /Users/fwilhelm/anaconda/envs/lib/python3.6/site-packages/six.
↪py
```

If you import `with_print_stdout`, Border-Patrol will use `print` as output function whereas `with_print_stderr` will print to standard error. Since most production applications will rather use the `logging` module, you can tell Border-Patrol to use it by importing `with_log_{error|warning|info|debug}`. For instance `from border_patrol import with_log_info` will log the final report by using the `INFO` logging level.

If you want even more fine grained control you can import the `BorderPatrol` class directly from the `border_patrol` package and use the `register()` and `unregister()` method to activate and deactivate it, respectively. At any point the tracking can be circumvented by using `border_patrol.builtin_import`.

CHAPTER

TWO

HOW DOES IT WORK?

Border-Patrol is actually quite simple. It overwrites the `__import__` function in Python's `builtins` package to track every imported module. For each module the corresponding package is determined and the version number is retrieved with the help of the `__version__` attribute which most professional libraries provide at the package level. If this fails the distribution name for the package is determined, e.g. `scikit-learn` is the distribution containing the `sklearn` package, with the help of `pkg_resources` which is a part of `setuptools`. Then the distribution name is used to determine the version number also using `pkg_resources` similar to how `pip` would do it.

Finally, Border-Patrol registers an `atexit` handler to be called when your application finishes and reports all imported modules. To avoid any problem registering these things more than once, Border-Patrol is implemented as a singleton and thus it is *not* thread-safe.

CHAPTER

THREE

NOTE

This project has been set up using PyScaffold 3.1. For details and usage information on PyScaffold see https://pyscaffold.org/.

CHAPTER

FOUR

CONTENTS

4.1 License

4.2 Contributors

- Florian Wilhelm <florian.wilhelm@inovex.de>
- Daniel Hepper <info@epicco.de>

4.3 Changelog

4.3.1 Version 1.0.1

- Added funding note to github
- Publishing via API token

4.3.2 Version 1.0

- After years of testing, let's call it stable ;-)

4.3.3 Version 0.3

- Remove problem when `__file__` returns `None`

4.3.4 Version 0.2

- switch back to default `repr` implementation
- have `template` as singleton attribute
- some more docs

4.3.5 Version 0.1

- First release

4.4 border_patrol

4.4.1 border_patrol package

Submodules

border_patrol.with_log_debug module

Import this module to let Border-Patrol use logging with level DEBUG

border_patrol.with_log_error module

Import this module to let Border-Patrol use logging with level ERROR

border_patrol.with_log_info module

Import this module to let Border-Patrol use logging with level INFO

border_patrol.with_log_warning module

Import this module to let Border-Patrol use logging with level WARNING

border_patrol.with_print_stderr module

Import this module to let Border-Patrol use plain print on stderr

border_patrol.with_print_stdout module

Import this module to let Border-Patrol use plain print

Module contents

Main module holding the actual functionality of Border-Patrol

class border_patrol.**BorderPatrol**(**args*, ***kwargs*)

Bases: object

Border-Patrol singleton class to track imports of packages.

Since BorderPatrol is a singleton, passing `None` for a value will keep the currently set value while passing a value will update the corresponding parameter.

Parameters

- **report_fun** (*callable*) – output function for reporting imports
- **ignore_std_lib** (*bool*) – ignore imports of Python's stdlib, default True
- **report_py** (*bool*) – also report the Python runtime version, default True

template

string template for the report

Type
str

at_exit()

Handler to be called at exit

register()

Registers/activates Border Patrol

Returns
Border-Patrol instance

Return type
self

report()

Reports currently imported libraries

Returns
list of package's (name, version, path)

Return type
list

track(*module*)

Tracks packages for later reporting

Parameters
module – module instance

unregister()

UnRegisters/deactivates Border Patrol

Returns
Border-Patrol instance

Return type
self

class border_patrol.**IdentityDict**

Bases: dict

Dictionary returning key by default

border_patrol.**get_package**(*module*)

Gets package part of module

Parameters
module – module instance

Returns
name of module's package

Return type
str

border_patrol.**get_pkg_to_dist_map**()

Generates mapping of packages to distributions

Returns
mapping of packages to distributions

Return type
dict

border_patrol.**package_path**(*package*)

Retrieves path of package

Parameters
package – module instance of package

Returns
path of package

Return type
str

border_patrol.**package_version**(*package*, *pkg_to_dist_map=None*)

Retrieves version string of package

Parameters

- **package** (*module*) – package as module instance
- **pkg_to_dist_map** (*dict*) – mapping of packages to their distributions. Avoids recalculation if passed. (optional)

Returns
version string of package

Return type
str

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

b

INDEX