Healthchecks Decorator

Daniel del Valle

Sep 25, 2023

CONTENTS

A simple python decorator for healthchecks.io.

CHAPTER ONE

FEATURES

The *healthchecks-decorator* library provides the following features:

- **Easy to use:** Simply decorate your function with @*healthcheck* to enable health checks.
- **Execution time measurement:** Supports sending */start* signals to measure job execution times.
- **Exception handling:** Automatically sends */failure* signals when jobs produce exceptions.
- **Auto-provisioning:** Supports automatic provisioning of new health checks by adding *?create=1* to the ping URL.
- **Diagnostics information:** Send diagnostics information to help diagnose issues.
- **Flexible endpoint support:** Supports both SaaS and self-hosted endpoints.

CHAPTER TWO

REQUIREMENTS

- None - just pure python .

CHAPTER THREE

INSTALLATION

You can install *Healthchecks Decorator* via pip from PyPI:

```
$ pip install healthchecks-decorator
```

CHAPTER

FOUR

USAGE

4.1 Basic usage

```python
from healthchecks_decorator import healthcheck

@healthcheck(url="https://hc-ping.com/<uuid1>")
def job():
   """Job with a success healthcheck signal when done"""
   pass


@healthcheck(url="https://hc-ping.com/<uuid2>", send_start=True)
def job_with_start():
   """Send also a /start signal before starting"""
   pass


@healthcheck(url="https://hc-ping.com/<uuid3>")
def job_with_exception():
   """This will produce a /fail signal"""
   raise Exception("I'll be propagated")


@healthcheck(url="https://hc-ping.com/<uuid4>", send_diagnostics=True)
def job_with_diagnostics():
   """Send the returned value in the POST body.
   The returned value must be a valid input for `urllib.parse.urlencode`.
   Otherwise, nothing will be sent."""
   return {"temperature": -7}
```

4.2 Environment variables

It is possible to set options through environment variables. Each option has a corresponding environment variable defined by the option name in *upper snake case* with the `HEALTHCHECK_` prefix.

For example, setting:

- `HEALTHCHECK_URL=http://fake-hc.com/uuid`
- `HEALTHCHECK_SEND_DIAGNOSTICS=TRUE`
- `HEALTHCHECK_SEND_START=1`

will allow having the most minimalist usage:

```
@healthcheck
def job():
    """Url, send_diagnostics and send_start are grabbed from environment."""
    pass
```

Note: Boolean options will be parsed as `True` if the env var is set to the word 'true' (in any case) or '1'. Otherwise, the option is set to `False`.

Note: Explicit values take precedence over environment variables.

Please see the Documentation for details.

CHAPTER

FIVE

CONTRIBUTING

Contributions are very welcome. To learn more, see the Contributor Guide.

CHAPTER SIX

LICENSE

Distributed under the terms of the MIT license, *Healthchecks Decorator* is free and open source software.

CHAPTER SEVEN

ISSUES

If you encounter any problems, please file an issue along with a detailed description.

CHAPTER EIGHT

CREDITS

- healthchecks.io.
- This project was generated from @cjolowicz's Hypermodern Python Cookiecutter template.

8.1 Reference

8.1.1 healthchecks_decorator

Healthchecks Decorator.

`healthchecks_decorator.`**`healthcheck`**(*func=None*, *, *url=None*, *send_start=None*, *send_diagnostics=None*)

Healthcheck decorator.

Parameters

- **`func`** (`t.Union[WrappedFn, None], optional`) – The function to decorate. Defaults to None.
- **`url`** (`str`) – The ping URL (e.g.: "https://hc-ping.com/<uuid>"). Must start with *http*.
- **`send_start`** (`bool, optional`) – Whether to send a '/start' signal. Defaults to False.
- **`send_diagnostics`** (`bool, optional`) – When enabled, send the wrapped function returned value as diagnostics information. Defaults to False.

Returns

A wrapped function.

Return type

t.Union[WrappedFn, t.Callable[[WrappedFn], WrappedFn]]

8.2 Contributor Guide

Thank you for your interest in improving this project. This project is open-source under the MIT license and welcomes contributions in the form of bug reports, feature requests, and pull requests.

Here is a list of important resources for contributors:

- Source Code
- Documentation
- Issue Tracker

8.2.1 How to report a bug

Report bugs on the Issue Tracker.

When filing an issue, make sure to answer these questions:

- Which operating system and Python version are you using?
- Which version of this project are you using?
- What did you do?
- What did you expect to see?
- What did you see instead?

The best way to get your bug fixed is to provide a test case, and/or steps to reproduce the issue.

8.2.2 How to request a feature

Request features on the Issue Tracker.

8.2.3 How to set up your development environment

You need Python 3.7+ and the following tools:

- Poetry
- Nox
- nox-poetry

Install the package with development requirements:

```
$ poetry install
```

You can now run an interactive Python session, or the command-line interface:

```
$ poetry run python
$ poetry run healthchecks-decorator
```

8.2.4 How to test the project

Run the full test suite:

```
$ nox
```

List the available Nox sessions:

```
$ nox --list-sessions
```

You can also run a specific Nox session. For example, invoke the unit test suite like this:

```
$ nox --session=tests
```

Unit tests are located in the `tests` directory, and are written using the pytest testing framework.

8.2.5 How to submit changes

Open a pull request to submit changes to this project.

Your pull request needs to meet the following guidelines for acceptance:

- The Nox test suite must pass without errors and warnings.
- Include unit tests. This project maintains 100% code coverage.
- If your changes add functionality, update the documentation accordingly.

Feel free to submit early, though—we can always iterate on this.

To run linting and code formatting checks before committing your change, you can install pre-commit as a Git hook by running the following command:

```
$ nox --session=pre-commit -- install
```

It is recommended to open an issue before starting work on anything. This will allow a chance to talk it over with the owners and validate your approach.

8.3 MIT License

PYTHON MODULE INDEX

INDEX