whisk Documentation

Release 0.1.24

Derek Haynes

May 13, 2020

Contents:

CHAPTER 1

Installation

1.1 Stable release

To install whisk, run this command in your terminal:

```
$ pip install whisk
```

This is the preferred method to install whisk, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

1.2 From sources

The sources for whisk can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/itsderek23/whisk
```

Or download the tarball:

```
$ curl -OJL https://github.com/itsderek23/whisk/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER 2

Usage

To use whisk in a project:

```
import whisk
```

CHAPTER 3

whisk

3.1 whisk package

3.1.1 Submodules

3.1.2 whisk.cli module

3.1.3 whisk.whisk module

3.1.4 Module contents

Top-level package for whisk.

whisk.**data_dir = None**
: The location of the artifacts directory. This is set from the Project instance and made available as a top-level attribute since it is frequently used.

whisk.**project = None**
: The location of the data directory. This is set from the Project instance and made available as a top-level attribute since it is frequently used.

CHAPTER 4

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/whiskml/whisk/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

4.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

4.1.4 Write Documentation

whisk could always use more documentation, whether as part of the official whisk docs, in docstrings, or even on the web in blog posts, articles, and such.

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/whiskml/whisk/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

4.2 Get Started!

Ready to contribute? Here's how to set up *whisk* for local development.

1. Fork the *whisk* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/whisk.git
```

3. Install your local copy into a venv. This is how you set up your fork for local development:

```
$ cd whisk/
$ python3 -m venv venv
$ source venv/bin/activate
$ python setup.py develop
$ pip install -r requirements_dev.txt
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 whisk tests
$ pytest -s --ignore=whisk/template
$ tox
```

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

4.3 Creating a demo project

To create a demo project from your local fork of whisk:

1. Set the following environment variables:

   PROJECT_DEMO_DIR # The directory where the demo project will be created
   PROJECT_DEMO_NAME # The name of the project

2. Then run the make task:

   $ make create-demo

make create-demo deletes the existing demo project (if it exists) and creates a new project.

4.4 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.6, 3.7 and 3.8, and for PyPy. Check https://app.circleci.com/pipelines/github/whisk-ml/whisk and make sure that the tests pass for all supported Python versions.

4.5 Tips

To run a subset of tests:

```
$ pytest tests/test_whisk.py
```

4.6 Testing the source code vs. the package

When running *pytest*, you are testing the source code in the current *venv*. When running *tox*, you are testing the package generated by *python setup.py sdist*. It's important to run *tox* as it runs test against the package other users will install. *tox* can fail even if *pytest* succeeds because of an incorrect *MANIFEST.in* file or missing dependencies within the *setup.py install_requires* argument.

4.7 Checking MANIFEST.in

It's easy to add files to version control but forget to include in the MANIFEST.in file. After committing changes, run the following to see if any files are missing:

$ check-manifest

4.8 Updating the getting started notebook

The project contains a notebook to help orientate new users. You can modify this notebook in the demo project and update the template with:

$ make update-notebook

4.9 Deploying

Make sure all your changes are pushed (including an entry in HISTORY.rst) and pass CI tests. Then run:

```
$ bump2version patch # possible: major / minor / patch
$ git push
$ git push --tags
$ make release
```

We intended to have CircleCI perform the release, but it's having issues with git tags. https://discuss.circleci.com/t/jobs-triggered-by-annotated-tags-fail-when-using-built-in-git-client/34486

If doing a small patch, you can just run:

$ make bump-push release

4.10 CI Setup - Required environment variables

GIT_AUTHOR_EMAIL GIT_AUTHOR_NAME GIT_COMMITTER_EMAIL GIT_COMMITTER_NAME PYPI_PASSWORD PYPI_USERNAME

CHAPTER 5

History

5.1 0.1.0 (2020-05-05)

- First release on PyPI.

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Python Module Index

W

Index