Yason Documentation

Release 0.1.0

Konstantin Taletskiy

Oct 07, 2021

Contents:

CHAPTER 1

yason: Jupyter Notebook Scheduler for Argo

JupyterHub deployed on Kubernetes allows teams to do data analysis in the browser and efficiently share computational resources. But when it comes to Jupyter Notebooks, it is not easy to run them remotely and non-interactively or schedule them. Fortunately, with Papermill and Argo Workflows it is now possible to do just that. Yason is Python package and CLI for scheduling the remote execution of Jupyter Notebooks.

- Free software: MIT license
- Documentation: https://yason.readthedocs.io.

1.1 Features

- Check the status of all scheduled notebook jobs
- Schedule a notebook for execution
- Get resulting notebook after execution
- Delete scheduled notebook job

1.2 Coming soon

- Including multiple files/folders together with Notebook for execution
- Integration with Argo events for scheduling

1.3 Prerequisites

Yason is intended to run on JupyterLab pods spawned by JupyterHub deployed on Kubernetes. Yason also requires Argo Workflows to be deployed on the same cluster in the namespace *argo*. S3 bucket is required for intermediate storage of Notebooks before and after their execution.

See more details and instructions in the full documentation https://yason.readthedocs.io

1.4 Usage

To use Yason in Python project:

```
from yason import yason
```

1.5 CLI Tool

To use Yason as CLI tool, type:

```
yason COMMAND [ARGS]
```

To see the list of all scheduled Notebooks for your JupyterHub username, type:

```
yason list
```

To schedule and execute Notebook immediately, type:

```
yason run <Notebook Name>.ipynb
```

To get the resulting Notebook after execution, type:

```
yason get <Workflow ID> <Destination>
```

i.e.:

```
yason get 25fe9753bc854148aac26ff7d97ba128 My_Notebook_result.ipynb
```

To delete the scheduled Notebook from your list, type:

```
yason delete <Workflow ID>
```

i.e.:

```
yason delete 25fe9753bc854148aac26ff7d97ba128
```

1.6 Name

Yason (or Jason) in Greek mythology is the leader of Argonauts. Yason brings your Jupyter Notebooks on board of Argo (Workflows).

1.7 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

1.7.1 Requirements

Yason is intended to be used as a part of JupyterLab deployment on Kubernetes that also has Argo workflows and access to S3 bucket (in my case S3 bucket is provided by Rook/Ceph) deployed on the same cluster.

1.7.2 Installation

Stable release

To install Yason, run this command in your terminal:

```
$ pip install yason
```

This is the preferred method to install yason, 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.

From sources

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

You can either clone the public repository:

```
$ git clone git://github.com/ktaletsk/yason
```

Or download the tarball:

```
$ curl  -OL https://github.com/ktaletsk/yason/tarball/master
```

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

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

1.7.3 Usage

To use yason in a project:

```
import yason
```

1.7.4 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:

Types of Contributions

Report Bugs

Report bugs at https://github.com/ktaletsk/yason/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.

Fix Bugs

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

Implement Features

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

Write Documentation

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

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/ktaletsk/yason/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 :)

Get Started!

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

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

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

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv yason
$ cd yason/
$ python setup.py develop
```

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 yason tests
$ python setup.py test or py.test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

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.

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 2.7, 3.4, 3.5 and 3.6, and for PyPy. Check https://travis-ci.org/ktaletsk/yason/pull_requests and make sure that the tests pass for all supported Python versions.

Tips

To run a subset of tests:

```
$ py.test tests.test_yason
```

Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bumpversion patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

1.7.5 Credits

Development Lead

- Konstantin Taletskiy <konstantin@taletskiy.com>

Contributors

None yet. Why not be the first?

1.7.6 History

0.1.0 (2019-05-22)

- First release on PyPI.

CHAPTER 2

Indices and tables

- genindex
- modindex
- search