Tools I use daily and can contribute to

Tom Rochette <tom.rochette@coreteks.org>

January 31, 2020 — 33893357

1 Question

What are the tools I use daily that I could contribute to?

2 Answer

2.1 General

- Visual Studio Code My text editor of choice when I don't need an IDE. I've implemented a few plugins for VSC and use it daily to write personal notes as well as blog articles. I sometimes make use of its diff tool to merge changes from Sourcetree.
- Pycharm My IDE of choice when I write python. Very powerful, easy to get used to if you've used other Jetbrains IDE (I've used PHPStorm for more than 5 years). Extremely useful to run a specific unit test using pytest or to debug a complex issue by putting breakpoints and investigating the internal state of the program.
- Docker I use docker at work to containerize all of our dependencies so that it is somewhat easy to deploy what we develop in "any" environment, that is, an environment where docker (or similar, such as Kubernetes) is installed.
- Drone CI We use Drone CI at work to do continuous integration and I consider this tool to be an essential part of my daily work. When people push code to github and it fails on Drone CI, I can use this information to help them fix their issues. We also use it has part of our PR process to ensure that the PR passes all the expected tests so that we do not introduce faulty code into our master branch.
- Dependabot A few months ago I had introduced dependabot into my dependency management practices. It was highly useful to get automated PRs with updates to libraries we depended on. However, since the release of poetry 1.0.0, dependabot has not been able to update my python dependencies and has been left unused. I've created a PR which I hope will move this issue forward and get dependabot working again with poetry.
- Plotly I used to use the highcharts plotting library until someone at work introduced me to plotly. Plotly.js is open source software and released under the MIT license, which makes it an ideal library to use in personal as well as commercial software.

2.2 Python specific

- Pandas I do machine learning development for a living nowadays and I depend highly on pandas. I don't think there's a single work day that goes by in which I don't whip out at least one `pd.DataFrame`.
- Scikit-learn Similar to my dependency on pandas, my dependency on Scikit-learn is on a daily basis. Unlike the DummyRegressor documentation suggests, I use it for real problems and it's definitely useful!
- Dask / Distributed In order to scale both horizontally and vertically machine learning problems I've leaned on dask and distributed. Their use of delayed and Futures has made it simple to migrate simple for loops code into highly distributed tasks which can be monitored through a bokeh dashboard.
- pytest Who writes code without testing it? Pytest is the PHPUnit of python for me, an essential component that is used daily to ensure that code doesn't regress more than it needs to.

- mypy Python typing system is pretty weak in my opinion. I miss using PHP typing system, as well as its visibility system. Mypy is similar to doing a code compilation pass and verifying that the types specified in a function signature are the types of the arguments given to that function. It is useful in order to detect mistakes in the arguments being passed to a function.
- isort I'm a tidy man. I like when my imports are ordered alphabetically. That's what isort is there for.
- black I don't particularly like discussing code style with others because everyone has their own quirks and creating a code style that everyone agrees on is as difficult as agreeing on whether tabs or spaces should be used. black is highly opinionated and doesn't allow for much to be tweaked, while it also has a sensible style that sometimes can make you crazy.
- prospector Prospector allows you to run a variety of linters on your code, which is quite useful when you like your code to be as standard and pretty as I like it. Some of the tools also look for code complexity, which helps you identify nightmares before they're in the master branch.
- poetry I've used pip, I've used pipenv, requirements.txt, setup.py, etc. I didn't like the setup.py because since I've used composer (for PHP), I've always seen dependency management as something that shouldn't require code to define. I didn't like the requirements.txt/.lock variants because it was never clear how those were generated and if they were kept up to date together since you could use the requirements.txt as soft dependencies and requirements.lock as hard dependencies that you had to freeze yourself (which many people didn't know about). pipenv Pipfile was alright, but adding dependencies seemed to take longer and longer, which wasn't a pleasant experience, especially when the package you wanted to add didn't want to play nice with the other packages. poetry was the closest experience I got to composer.