When writing a blog post with Microsoft Word – the filename becomes the title. In this case the file name is “2020-01-01-Microsoft-Word-Example-Post.docx”.

There is minimal support for Word documents in fastpages compared to Jupyter notebooks. Some known limitations:

* [alt text](https://support.office.com/en-us/article/add-alternative-text-to-a-shape-picture-chart-smartart-graphic-or-other-object-44989b2a-903c-4d9a-b742-6a75b451c669) in Word documents are not yet supported by fastpages, and will break links to images.
* You can only specify [front matter](https://jekyllrb.com/docs/front-matter/) for Word documents globally. See [the README](https://github.com/fastai/fastpages#writing-blog-posts-with-microsoft-word) for more details.

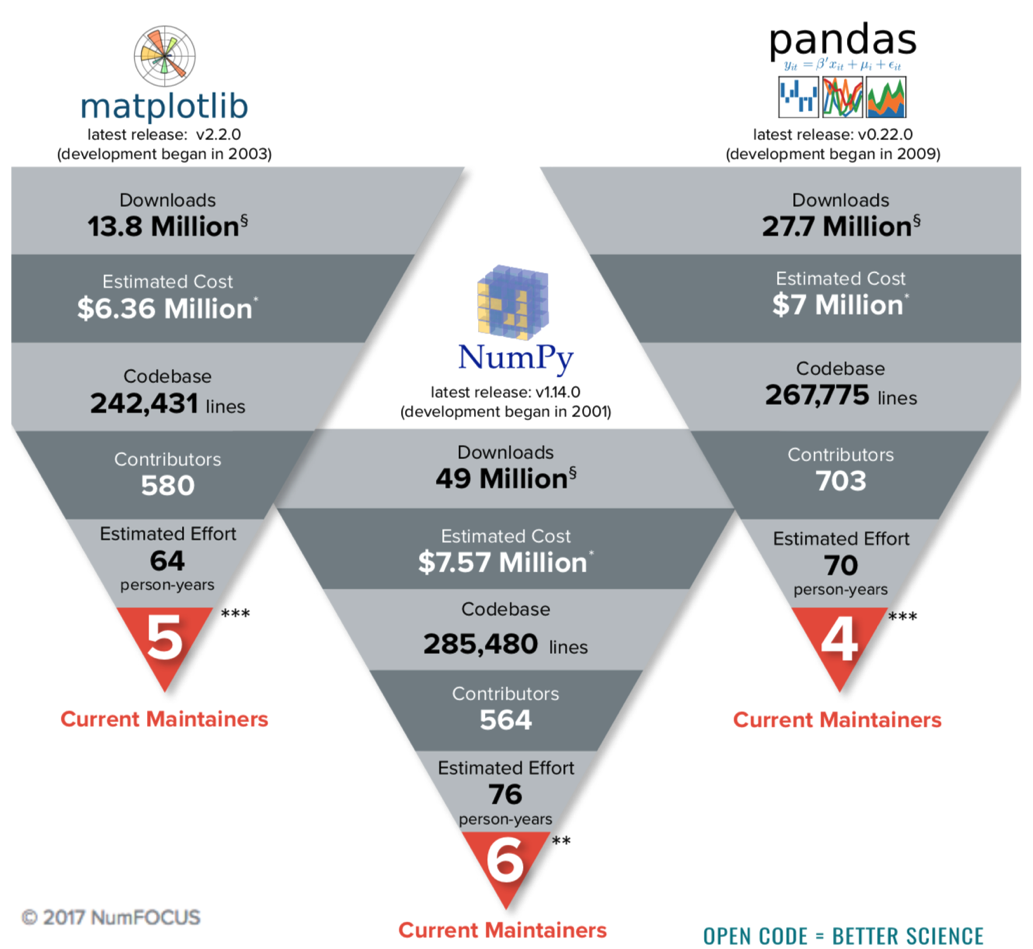
For greater control over the content produced in word documents, you will need to convert word documents to markdown files manually. You can follow the steps this [blog post](https://www.fast.ai/2020/01/18/gitblog/), which walk you through how to use [pandoc](https://pandoc.org/installing.html) to do the conversion. **Note**: If you wish to customize your Word-generated blog post in markdown, make sure you delete your word document from the \_word directory so your markdown file doesn’t get overwritten!

*If your primary method of writing blog posts is Word documents, and you plan on always manually editing markdown files converted from Word, you are probably better off using* [*fast\_template*](https://github.com/fastai/fast_template) *instead of* ***fastpages*.**

The material below is a reproduction of [this blog post](https://medium.com/kubeflow/reducing-maintainer-toil-on-kubeflow-with-github-actions-and-machine-learning-f8568374daa1), and serves as an illustrative example.

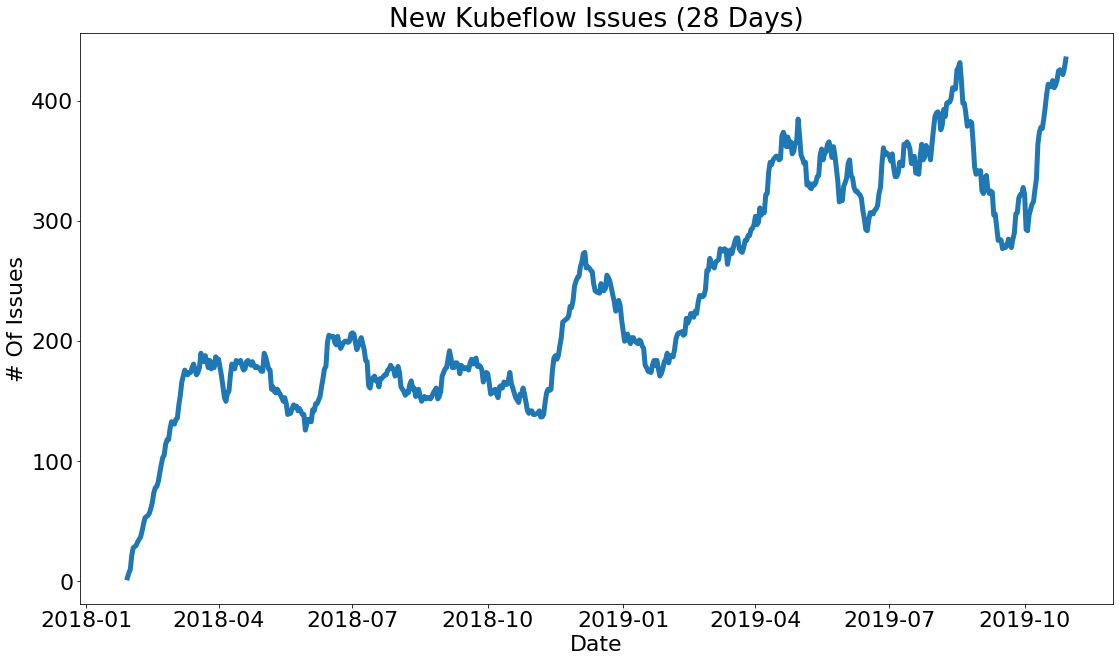
Maintaining a healthy open source project can entail a huge amount of toil. Popular projects often have orders of magnitude more users and episodic contributors opening issues and PRs than core maintainers capable of handling these issues.

Consider this graphic prepared by the NumFOCUS foundation showing the number of maintainers for three widely used scientific computing projects:



We can see that across these three projects, there is a very low ratio maintainers to users. Fixing this problem is not an easy task and likely requires innovative solutions to address the economics as well as tools.

Due to its recent momentum and popularity, Kubeflow suffers from a similar fate as illustrated by the growth of new issues opened:



Source: “[TensorFlow World 2019, Automating Your Developer Workflow With ML](http://bit.ly/tf-github)”

Coincidentally, while building out end to end machine learning [examples](https://github.com/kubeflow/examples) for Kubeflow, we built two examples using publicly available GitHub data: [GitHub Issue Summarization](https://github.com/kubeflow/examples/tree/master/github_issue_summarization) and [Code Search](https://github.com/kubeflow/examples/tree/master/code_search). While these tutorials were useful for demonstrating components of Kubeflow, we realized that we could take this a step further and build concrete data products that reduce toil for maintainers.

This is why we started the project [kubeflow/code-intelligence](https://github.com/kubeflow/code-intelligence), with the goals of increasing project velocity and health using data driven tools. Below are two projects we are currently experimenting with :

1. [Issue Label Bot](https://github.com/marketplace/issue-label-bot): This is a bot that automatically labels GitHub issues using Machine Learning. This bot is a GitHub App that was originally built for Kubeflow but is now also used by several large open source projects. The [current version](https://github.com/machine-learning-apps/Issue-Label-Bot) of this bot only applies a very limited set of labels, however we are currently [A/B testing new models](https://twimlai.com/twiml-talk-313-machine-learning-at-github-with-omoju-miller/) that allow personalized labels. Here is a [blog post](https://towardsdatascience.com/mlapp-419f90e8f007) discussing this project in more detail.
2. [Issue Triage GitHub Action](https://github.com/kubeflow/code-intelligence/tree/master/Issue_Triage/action): to compliment the Issue Label Bot, we created a GitHub Action that automatically adds / removes Issues to the Kubeflow project board tracking issues needing triage.

Together these projects allow us to reduce the toil of triaging issues. The GitHub Action makes it much easier for the Kubeflow maintainers to track issues needing triage. With the label bot we have taken the first steps in using ML to replace human intervention. We plan on [using features extracted by ML](https://github.com/kubeflow/code-intelligence/tree/master/Issue_Embeddings) to automate more steps in the triage process to further reduce toil.

## Building Solutions with GitHub Actions

One of the premises of Kubeflow is that a barrier to building data driven, ML powered solutions is getting models into production and integrated into a solution. In the case of building models to improve OSS project health, that often means integrating with GitHub where the project is hosted.

We are really excited by GitHub’s newly released feature [GitHub Actions](https://github.com/features/actions) because we think it will make integrating ML with GitHub much easier.

For simple scripts, like the issue triage script, GitHub actions make it easy to automate executing the script in response to GitHub events without having to build and host a GitHub app.

To automate adding/removing issues needing triage to a Kanban board we wrote a simple [python](https://github.com/kubeflow/code-intelligence/blob/master/py/issue_triage/triage.py) script that interfaces with GitHub’s [GraphQL API](https://developer.github.com/v4/) to modify issues.

As we continue to iterate on ML Models to further reduce toil, GitHub Actions will make it easy to leverage Kubeflow to put our models into production faster. A number of prebuilt GitHub Actions make it easy to create Kubernetes resources in response to GitHub events. For example, we have created [GitHub Actions to launch Argo Workflows](https://github.com/marketplace?utf8=%E2%9C%93&type=actions&query=argo). This means once we have a Kubernetes job or workflow to perform inference we can easily integrate the model with GitHub and have the full power of Kubeflow and Kubernetes (eg. GPUs). We expect this will allow us to iterate much faster compared to building and maintaining GitHub Apps.

**Call To Action**

We have a lot more work to do in order to achieve our goal of reducing the amount of toil involved in maintaining OSS projects. If your interested in helping out here's a couple of issues to get started:

* Help us create reports that pull and visualize key performance indicators (KPI). <https://github.com/kubeflow/code-intelligence/issues/71>
  + We have defined our KPI here: [issue #19](https://github.com/kubeflow/code-intelligence/issues/19)
* Combine repo specific and non-repo specific label predictions: <https://github.com/kubeflow/code-intelligence/issues/70>

In addition to the aforementioned issues we welcome contributions for these [other issues](https://github.com/kubeflow/code-intelligence/issues) in our repo.