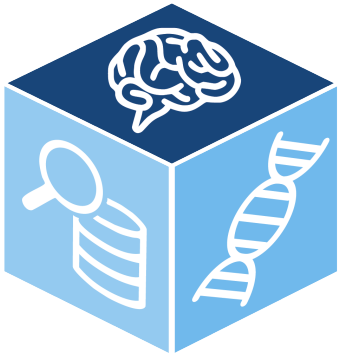

D3b

Center for Data-Driven Discovery in Biomedicine


# Developer Handbook

June 23, 2018

# Contents

# 1 Welcome

# 2 Creating Issues

Issues should be created If a repository has an *Issue Template*, make sure to follow it!

# 3 Finding Issues

## 3.1 Labels

Repositories across the D3b Github organization that have frequent user feedback and bug reports should use our standard set of labels to tag issues shown in Figure 3.1. Labels act as a means of filtering categories of issues and providing summaries of effort completed in release notes. Users that are not part of the organization may not add labels themselves, so please add labels as you see necessary to new issues. The help wanted label is a special label that is displayed in repository summaries on Github (See the *11 Issues need help* in Figure 8.3). Use it to mark issues that are self-contained and do not require in-depth knowledge of how the code base works. This tag is useful for guiding new members or external contributors looking to get involved to issues that will be easy for them to accomplish.

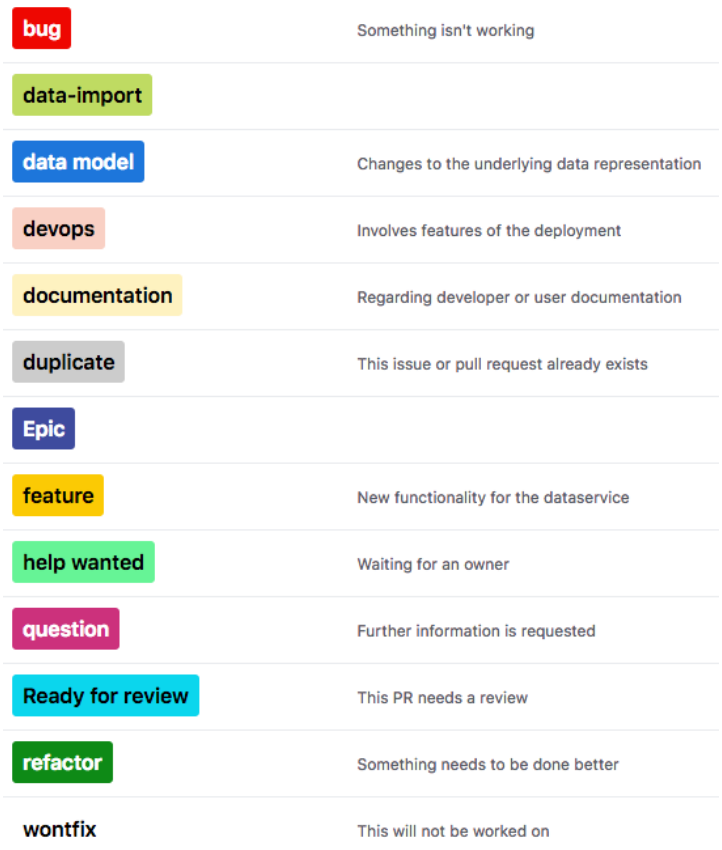


Figure 1: Standard Repository labels with descriptions and colors

# 4 Creating a Pull Request

## 4.1 Commit Messages

## 4.2 Pull Request Titles

## 4.3 Keeping the Commit Log Tidy

## 4.4 Labeling Pull Requests

## 4.5 Requesting Reviews

# 5 Review Process

## 5.1 Code Review

## 5.2 Testing

## 5.3 Status Checks

# 6 Software Release

# 7 New Repositories

## 7.1 Naming

## 7.2 Description

## 7.3 Tags

## 7.4 Protected Branches

## 7.5 README

## 7.6 License

## 7.7 Config Repository

In addition to creating the main code repository, each code repository that is to be deployed must have a second, *private* repository that contains the infrastructure and CI/CD strategy for that code base. The repository for the config *must match the name of the code repository suffixed with -config*. This is a rule of thumb that is followed throughout so that the configuration may be discovered automatically by other scripts. In addition to the naming standard, the repository should be *private*. It is best practice to keep the infrastructure obscured and prevent external contributions as the config contains code that is often run at elevated privileges in our environments.

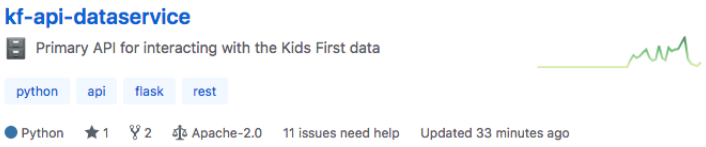


Figure 2: An exemplary repository summary including: proper naming scheme, emoji followed by short description, labels, the *Apache 2.0* license, and issues tagged with *help wanted*

# 8 Environments

Often, entire software stacks are deployed concurrently in many different *environments*. This provides us with a handful of benefits:

- Assurance that there is always a stable deployment for end-users
- Provides a safe sandbox to develop new features in isolation
- Insures the boilerplate to re-deploy a new environment exists

We divide our environments into three primary stages that are tightly couple to our Continuous Integration

## 8.1 dev

The development environment (dev) is intended as sandbox for new features and code bases. It is the most unstable environment with no guarantees on data availability. When new code is pushed to a feature branch, it automatically triggers its CI pipeline to test, build, and, deploy to the development environment. The branch must pass through testing and deployment successfully to before it is allowed to be merged to insure that it will successfully deploy to the next stage.

## 8.2 qa

The QA environment provides a mostly-stable environment where new features may be used, evaluated, and tested before being released. This is is where integration test-suites may be run to ensure that all services cooperate nicely. The data in QA should also be close to production quality to fully flex all features. New features enter QA after merging a feature branch into the master branch of a code base.

## 8.3 prd

The production environment (prd) is the end-user environment. This is critically stable as it must be exposed to the public.