# Contributing to Chronograf

## Bug reports

Before you file an issue, please search existing issues in case it has already been filed, or perhaps even fixed. If you file an issue, please include the following.

\* Full details of your operating system (or distribution) e.g. 64-bit Ubuntu 14.04.

\* The version of Chronograf you are running

\* Whether you installed it using a pre-built package, or built it from source.

\* A small test case, if applicable, that demonstrates the issues.

\* A screenshot, when appropriate, to demonstrate any issues that are related to the UI

Remember the golden rule of bug reports: \*\*The easier you make it for us to reproduce the problem, the faster it will get fixed.\*\*

If you have never written a bug report before, or if you want to brush up on your bug reporting skills, we recommend reading [Simon Tatham's essay "How to Report Bugs Effectively."](http://www.chiark.greenend.org.uk/~sgtatham/bugs.html)

Please note that issues are \_not the place to file general questions\_ such as "How do I use Chronograf?" Questions of this nature should be sent to the [InfluxData Community Site](https://community.influxdata.com), not filed as issues. Issues like this will be closed.

## Feature requests

We really like to receive feature requests, as it helps us prioritize our work. Please be clear about your requirements, as incomplete feature requests may simply be closed if we don't understand what you would like to see added to Chronograf.

## Contributing to the source code

Chronograf is built using Go for its API backend and serving the front-end assets, and uses Dep for dependency management. The front-end visualization is built with React (JavaScript) and uses Yarn for dependency management. The assumption is that all your Go development are done in `$GOPATH/src`. `GOPATH` can be any directory under which Chronograf and all its dependencies will be cloned. For full details on the project structure, follow along below.

## Submitting a pull request

To submit a pull request you should fork the Chronograf repository, and make your change on a feature branch of your fork. Then generate a pull request from your branch against \_master\_ of the Chronograf repository. Include in your pull request details of your change -- the why \_and\_ the how -- as well as the testing your performed. Also, be sure to run the test suite with your change in place. Changes that cause tests to fail cannot be merged.

There will usually be some back and forth as we finalize the change, but once that completes it may be merged.

To assist in review for the PR, please add the following to your pull request comment:

```md

\* [ ] CHANGELOG.md updated

\* [ ] Rebased/mergable

\* [ ] Tests pass

\* [ ] Sign [CLA](https://influxdata.com/community/cla/) (if not already signed)

```

## Signing the CLA

If you are going to be contributing back to Chronograf please take a second to sign our CLA, which can be found

[on our website](https://influxdata.com/community/cla/).

## Installing & Using Node

You'll need to install Node 10.13.0 to run the frontend chronograf application.

\* [Install Node 10.13.0 LTS](https://nodejs.org/en/blog/release/v10.13.0/)

## Installing & Using Yarn

You'll need to install Yarn to manage the frontend (JavaScript) dependencies.

\* [Install Yarn](https://yarnpkg.com/en/docs/install)

To add a dependency via Yarn, for example, run `yarn add <dependency>` from within the `/chronograf/ui` directory.

## Installing Go

Chronograf requires Go 1.13 or higher.

## Installing & Using Dep

You'll need to install Dep to manage the backend (Go) dependencies.

\* [Install Dep](https://github.com/golang/dep)

To add a dependency via Dep, for example, run `dep ensure -add <dependency>` from within the `/chronograf` directory. \_Note that as of this writing, `dep ensure` will modify many extraneous vendor files, so you'll need to run `dep prune` to clean this up before committing your changes. Apparently, the next version of `dep` will take care of this step for you.\_

## Revision Control Systems

Go has the ability to import remote packages via revision control systems with the `go get` command. To ensure that you can retrieve any remote package, be sure to install the following rcs software to your system.

Currently the project only depends on `git`.

\* [Install Git](http://git-scm.com/book/en/Getting-Started-Installing-Git)

## Getting the source

Setup the project structure and fetch the repo like so:

```bash

mkdir $HOME/go

export GOPATH=$HOME/go

go get github.com/influxdata/chronograf

```

You can add the line `export GOPATH=$HOME/go` to your bash/zsh file to be set for every shell instead of having to manually run it everytime.

## Cloning a fork

If you wish to work with fork of Chronograf, your own fork for example, you must still follow the directory structure above. But instead of cloning the main repo, instead clone your fork. Follow the steps below to work with a fork:

```bash

export GOPATH=$HOME/go

mkdir -p $GOPATH/src/github.com/influxdata

cd $GOPATH/src/github.com/influxdata

git clone git@github.com:<username>/chronograf

```

Retaining the directory structure `$GOPATH/src/github.com/influxdata` is necessary so that Go imports work correctly.

## Build and Test

Make sure you have `go` and `yarn` installed and the project structure as shown above. We provide a `Makefile` to get up and running quickly, so all you'll need to do is run the following:

```bash

cd $GOPATH/src/github.com/influxdata/chronograf

make

```

The binaries will be located in `$GOPATH/bin`.

To run the tests, execute the following command:

```bash

cd $GOPATH/src/github.com/influxdata/chronograf

make test

```

## Continuous Integration testing

Chronograf uses CircleCI for continuous integration testing. To see how the code is built and tested, check out [this file](https://github.com/influxdata/chronograf/blob/master/Makefile). It closely follows the build and test process outlined above. You can see the exact version of Go Chronograf uses for testing by consulting that file.