# Everyone

Please take some time to review our [code of conduct](CODE-OF-CONDUCT.md) to help guide your interactions with others on this project.

# User

Before you open a new issue, please search for older ones that cover the same issue.

In general "me too" comments/issues are frowned upon.

You can add a :+1: emoji reaction to the issue if you want to express interest in this.

# Developer

We use `clang-format` version `3.8` to consistently format the code base. There is a helper script under `scripts/format.sh`.

The format is automatically checked by the `mason-linux-release` job of a Travis CI build.

To save development time a local hook `.git/hooks/pre-push`

```

#!/bin/sh

remote="$1"

if [ x"$remote" = xorigin ] ; then

if [ $(git rev-parse --abbrev-ref HEAD) = master ] ; then

echo "Rejected push to $remote/master" ; exit 1

fi

./scripts/format.sh && ./scripts/error\_on\_dirty.sh

if [ $? -ne 0 ] ; then

echo "Unstaged format changes" ; exit 1

fi

fi

```

could check code format, modify a local repository and reject push due to unstaged formatting changes.

Also `pre-push` hook rejects direct pushes to `origin/master`.

?? `scripts/format.sh` checks all local files that match `\*.cpp` or `\*.hpp` patterns.

In general changes that affect the API and/or increase the memory consumption need to be discussed first.

Often we don't include changes that would increase the memory consumption a lot if they are not generally usable (e.g. elevation data is a good example).

## Pull Request

Every pull-request that changes the API needs to update the docs in `docs/http.md` and add an entry to `CHANGELOG.md`.

Breaking changes need to have a BREAKING prefix. See the [releasing documentation](docs/releasing.md) on how this affects the version.

Early feedback is also important.

You will see that a lot of the PR have tags like `[not ready]` or `[wip]`.

We like to open PRs as soon as we are starting to work on something to make it visible to the rest of the team.

If your work is going in entirely the wrong direction, there is a good chance someone will pick up on this before it is too late.

Everyone is encouraged to read PRs of other people and give feedback.

For every significant code change we require a pull request review before it is merged.

If your pull request modifies the API this need to be signed of by a team discussion.

This means you will need to find another member of the team with commit access and request a review of your pull request.

Once your pull request is reviewed you can merge it! If you don't have commit access, ping someone that has commit access.

If you do have commit access there are in general two accepted styles to merging:

1. Make sure the branch is up to date with `master`. Run `git rebase master` to find out.

2. Once that is ensured you can either:

- Click the nice green merge button (for a non-fast-forward merge)

- Merge by hand using a fast-forward merge

Which merge you prefer is up to personal preference. In general it is recommended to use fast-forward merges because it creates a history that is sequential and easier to understand.

# Maintainer

## Doing a release

There is an in-depth guide around how to push out a release once it is ready [here](docs/releasing.md).

## The API

Changes to the API need to be discussed and signed off by the team. Breaking changes even more so than additive changes.

## Milestones

If a pull request or an issue is applicable for the current or next milestone, depends on the target version number.

Since we use semantic versioning we restrict breaking changes to major releases.

After a Release Candidate is released we usually don't change the API anymore if it is not critical.

Bigger code changes after a RC was released should also be avoided.