# Welcome to libgit2!

We're making it easy to do interesting things with git, and we'd love to have

your help.

## Licensing

By contributing to libgit2, you agree to release your contribution under

the terms of the license. Except for the `examples` and the

`deps` directories, all code is released under the [GPL v2 with

linking exception](../COPYING).

The `examples` code is governed by the

[CC0 Public Domain Dedication](../examples/COPYING), so that you may copy

from them into your own application.

The bundled dependencies in the `deps` directories are governed

by the following licenses:

- http-parser is licensed under [MIT license](../deps/http-parser/COPYING)

- regex is governed by [LGPL v2.1+ license](../deps/regex/COPYING)

- winhttp is governed by [LGPL v2.1+](../deps/winhttp/COPYING.LGPL) and [GPL v2 with linking exception](../deps/winhttp/COPYING.GPL)

- zlib is governed by [zlib license](../deps/zlib/COPYING)

## Discussion & Chat

We hang out in the

[`#libgit2`](http://webchat.freenode.net/?channels=#libgit2)) channel on

irc.freenode.net.

Also, feel free to open an

[Issue](https://github.com/libgit2/libgit2/issues/new) to start a discussion

about any concerns you have. We like to use Issues for that so there is an

easily accessible permanent record of the conversation.

## Libgit2 Versions

The `master` branch is the main branch where development happens.

Releases are tagged

(e.g. [v0.21.0](https://github.com/libgit2/libgit2/releases/tag/v0.21.0) )

and when a critical bug fix needs to be backported, it will be done on a

`<tag>-maint` maintenance branch.

## Reporting Bugs

First, know which version of libgit2 your problem is in and include it in

your bug report. This can either be a tag (e.g.

[v0.17.0](https://github.com/libgit2/libgit2/releases/tag/v0.17.0)) or a

commit SHA

(e.g. [01be7863](https://github.com/libgit2/libgit2/commit/01be7863)).

Using [`git describe`](http://git-scm.com/docs/git-describe) is a

great way to tell us what version you're working with.

If you're not running against the latest `master` branch version,

please compile and test against that to avoid re-reporting an issue that's

already been fixed.

It's \*incredibly\* helpful to be able to reproduce the problem. Please

include a list of steps, a bit of code, and/or a zipped repository (if

possible). Note that some of the libgit2 developers are employees of

GitHub, so if your repository is private, find us on IRC and we'll figure

out a way to help you.

## Pull Requests

Our work flow is a [typical GitHub

flow](https://guides.github.com/introduction/flow/index.html), where

contributors fork the [libgit2 repository](https://github.com/libgit2/libgit2),

make their changes on branch, and submit a

[Pull Request](https://help.github.com/articles/using-pull-requests)

(a.k.a. "PR"). Pull requests should usually be targeted at the `master`

branch.

Life will be a lot easier for you (and us) if you follow this pattern

(i.e. fork, named branch, submit PR). If you use your fork's `master`

branch directly, things can get messy.

Please include a nice description of your changes when you submit your PR;

if we have to read the whole diff to figure out why you're contributing

in the first place, you're less likely to get feedback and have your change

merged in.

In addition to outlining your thought process in the PR's description, please

also try to document it in your commits. We welcome it if every commit has a

description of why you have been doing your changes alongside with your

reasoning why this is a good idea. The messages shall be written in

present-tense and in an imperative style (e.g. "Add feature foobar", not "Added

feature foobar" or "Adding feature foobar"). Lines should be wrapped at 80

characters so people with small screens are able to read the commit messages in

their terminal without any problem.

To make it easier to attribute commits to certain parts of our code base, we

also prefer to have the commit subject be prefixed with a "scope". E.g. if you

are changing code in our merging subsystem, make sure to prefix the subject with

"merge:". The first word following the colon shall start with an lowercase

letter. The maximum line length for the subject is 70 characters, preferably

shorter.

If you are starting to work on a particular area, feel free to submit a PR

that highlights your work in progress (and note in the PR title that it's

not ready to merge). These early PRs are welcome and will help in getting

visibility for your fix, allow others to comment early on the changes and

also let others know that you are currently working on something.

Before wrapping up a PR, you should be sure to:

\* Write tests to cover any functional changes

\* Update documentation for any changed public APIs

\* Add to the [`changelog.md`](changelog.md) file describing any major changes

## Unit Tests

We believe that our unit tests allow us to keep the quality of libgit2

high: any new changes must not cause unit test failures, and new changes

should include unit tests that cover the bug fixes or new features.

For bug fixes, we prefer unit tests that illustrate the failure before

the change, but pass with your changes.

In addition to new tests, please ensure that your changes do not cause

any other test failures. Running the entire test suite is helpful

before you submit a pull request. When you build libgit2, the test

suite will also be built. You can run most of the tests by simply running

the resultant `libgit2\_clar` binary. If you want to run a specific

unit test, you can name it with the `-s` option. For example:

libgit2\_clar -sstatus::worktree::long\_filenames

Or you can run an entire class of tests. For example, to run all the

worktree status tests:

libgit2\_clar -sstatus::worktree

The default test run is fairly exhaustive, but it will exclude some

unit tests by default: in particular, those that talk to network

servers and the tests that manipulate the filesystem in onerous

ways (and may need to have special privileges to run). To run the

network tests:

libgit2\_clar -ionline

In addition, various tests may be enabled by environment variables,

like the ones that write exceptionally large repositories or manipulate

the filesystem structure in unexpected ways. These tests \*may be

dangerous\* to run on a normal machine and may harm your filesystem. It's

not recommended that you run these; instead, the continuous integration

servers will run these (in a sandbox).

## Porting Code From Other Open-Source Projects

`libgit2` is licensed under the terms of the GPL v2 with a linking

exception. Any code brought in must be compatible with those terms.

The most common case is porting code from core Git. Git is a pure GPL

project, which means that in order to port code to this project, we need the

explicit permission of the author. Check the

[`git.git-authors`](https://github.com/libgit2/libgit2/blob/development/git.git-authors)

file for authors who have already consented.

Other licenses have other requirements; check the license of the library

you're porting code \*from\* to see what you need to do. As a general rule,

MIT and BSD (3-clause) licenses are typically no problem. Apache 2.0

license typically doesn't work due to GPL incompatibility.

If your pull request uses code from core Git, another project, or code

from a forum / Stack Overflow, then \*please\* flag this in your PR and make

sure you've given proper credit to the original author in the code

snippet.

## Style Guide

The public API of `libgit2` is [ANSI C](http://en.wikipedia.org/wiki/ANSI\_C)

(a.k.a. C89) compatible. Internally, `libgit2` is written using a portable

subset of C99 - in order to compile with GCC, Clang, MSVC, etc., we keep

local variable declarations at the tops of blocks only and avoid `//` style

comments. Additionally, `libgit2` follows some extra conventions for

function and type naming, code formatting, and testing.

We like to keep the source code consistent and easy to read. Maintaining

this takes some discipline, but it's been more than worth it. Take a look

at the [conventions file](conventions.md).

## Starter Projects

See our [projects list](projects.md).