# How to report issues

Before reporting an issue with GitHub, be sure that:

\* you are using the git version of radare2

\* you are using a clean installation

\* the issue was not already reported

When the above conditions are satisfied, feel free to submit an issue

while trying to be as precise as possible. If you can, provide the problematic

binary, the steps to reproduce the error and a backtrace in case of SEGFAULTs.

Any information will help to fix the problem.

# How to contribute

There are a few guidelines that we need contributors to follow so that we can

try to keep the codebase consistent and clean.

## Getting Started

\* Make sure you have a GitHub account and solid ability to use `git`.

\* Fork the repository on GitHub.

\* Create a topic branch from master. Please avoid working directly on the `master` branch.

\* Make commits of logical units.

\* Check for coding style issues with:

git diff master..mybranch | ./sys/clang-format-diff.py -p1

and be sure to follow the CODINGSTYLE (more on this in [DEVELOPERS.md][]).

\* Submit the Pull Request(PR) on Github.

\* Prefix the PR title with `WIP:` if it's not yet ready to be merged

\* When relevant, write a test in [test/](test).

## Rebasing onto updated master

Every so often, your PR will lag behind `master` and get conflicts.

To "update" your branch `my-awesome-feature`, you \*rebase\* it onto

the latest `radareorg/master`, and \*force-push\* the result into your fork.

#### Step 1: Switch to `master` branch.

$ git checkout master

#### Step 2: Pull new commits published to radareorg repo.

$ git pull https://github.com/radareorg/radare2

#### Step 3: Switch back to `my-awesome-feature` branch.

$ git checkout my-awesome-feature

#### Step 4: Rebase the `my-awesome-feature` branch.

$ git rebase master

Optionally, use the alternative mode "interactive rebase". It allows

to `squash` your commits all into one, reorder, reword them, etc.

$ git rebase -i master

Follow git instructions when conflicts arise.

#### Step 5: publish your updated local branch.

$ git push -f

This `-f` \*force-flag\* is needed because git commits are immutable: rebasing

creates newer versions of them. git needs to confirm the destruction of

previous incarnations.

When afraid to touch force and risk losing your work (do backups!..),

try \*merging master into your branch\* instead of rebasing onto it.

This is discouraged, as it produces ugly hard-to-maintain commit history.

## Commit message rules

When commiting your changes into the repository you may want to follow some

rules to make the git history more readable and consistent:

\* Start the message capitalized (only the first character must be in uppercase)

\* Be short and concise, the whole concept must fit one line

\* If a command is inlined, use backticks

\* Add a double-hashtag if the change matters for the changelog (See below)

\* If the commit fixes a bug start with 'Fix #number - '

\* For extra details, add an empty line and use asterisk item list below

\* Use present simple grammar tense (Add vs Added, Fix vs Fixed/Fixes)

### Commit message hashtag list:

\* ##anal - analysis related

\* ##asm - assembler

\* ##bin - binary parsing

\* ##build - build fixes/changes

\* ##config - config var changes/additions/renamings

\* ##cons - console/terminal-related

\* ##crypto - cryptography

\* ##debug - debugger stuff

\* ##diff - diffing code, strings, basic blocks, ...

\* ##disasm - disassembler

\* ##doc - documentation

\* ##egg - the `r\_lang` compiler

\* ##emu - emulation, including esil

\* ##graph - basic block graph, callgraph, ...

\* ##io - related to the `r\_io` library

\* ##json - json fixes/changes

\* ##lang - bindings

\* ##meta - metadata handling other than printing

\* ##optimization-space/time optimizations

\* ##port - portability (new OS/archs)

\* ##print - printing data, structures, strings, tables, types ..

\* ##projects - saving/loading state

\* ##refactor - improve code quality

\* ##remote - r2 over tcp, http, rap, serial .. including collaboration

\* ##search - rafind2, / command, ..

\* ##shell - commandline, newshell, ..

\* ##signatures-searching/generating them

\* ##test - testing infrastructure fixes/changes

\* ##tools - r2pm, rarun2, rax2 ... that don't fit in other categories

\* ##util - core apis

\* ##visual - visual ui, including panels

# Additional resources

\* [README.md][]

\* [DEVELOPERS.md][]

[README.md]: https://github.com/radareorg/radare2/blob/master/README.md

[DEVELOPERS.md]: https://github.com/radareorg/radare2/blob/master/DEVELOPERS.md

If you need more confidence in your git skills, check out this quick guide:

<https://learnxinyminutes.com/docs/git/>