

Git for Contributors

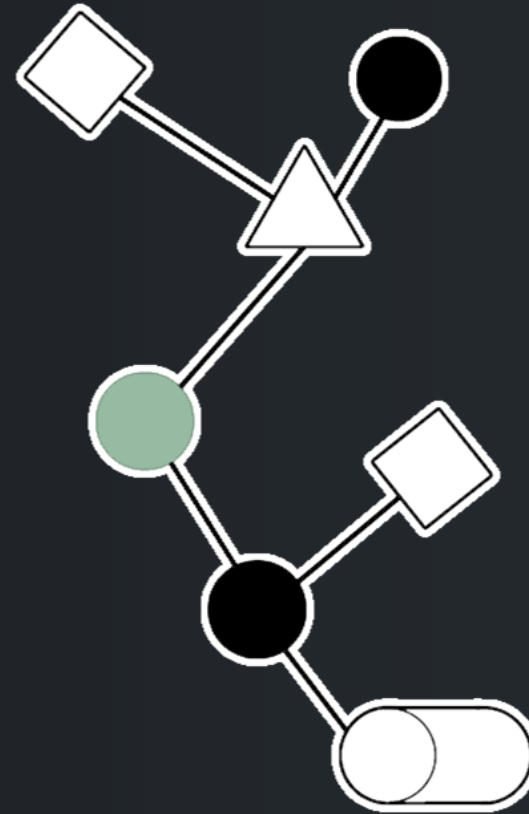
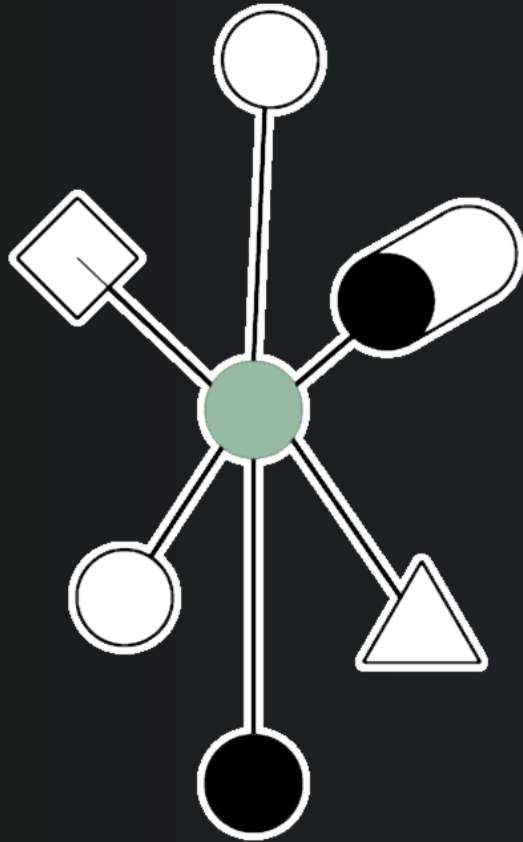
Learn how git works to better contribute to open source

Federico Grandi - @EndBug - federicograndi@duck.com

What is git?



What is git?

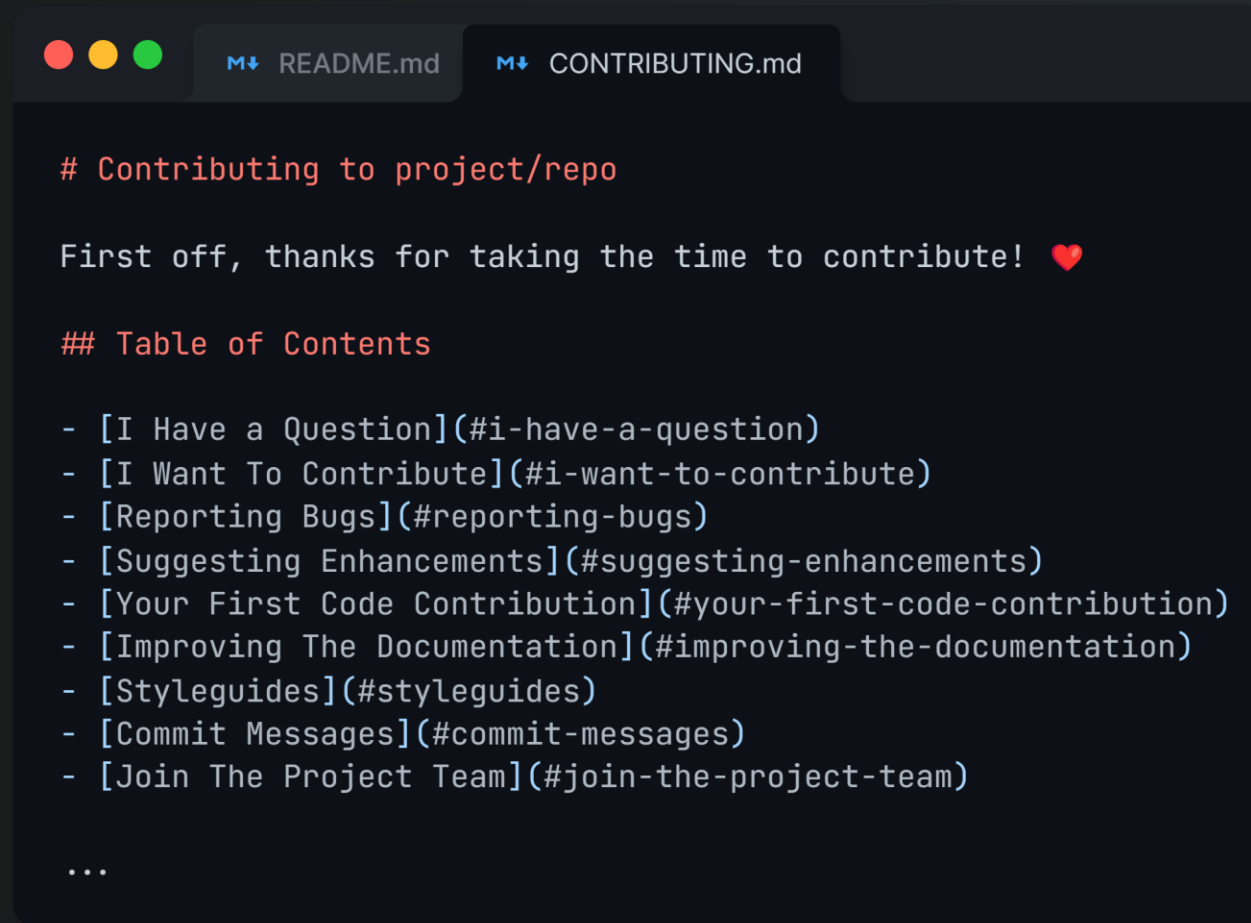


Open source projects



Contributing to an open source project

Contributing guidelines



```
# Contributing to project/repo

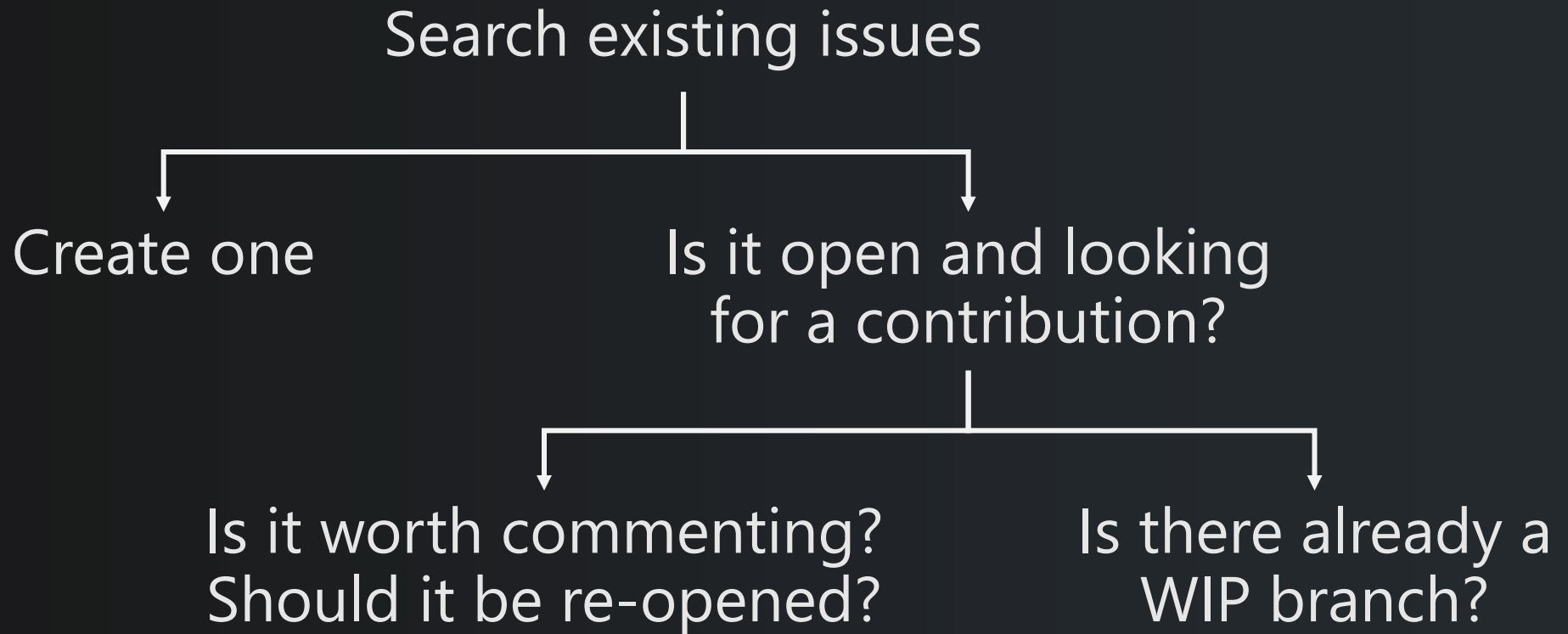
First off, thanks for taking the time to contribute! ❤️

## Table of Contents

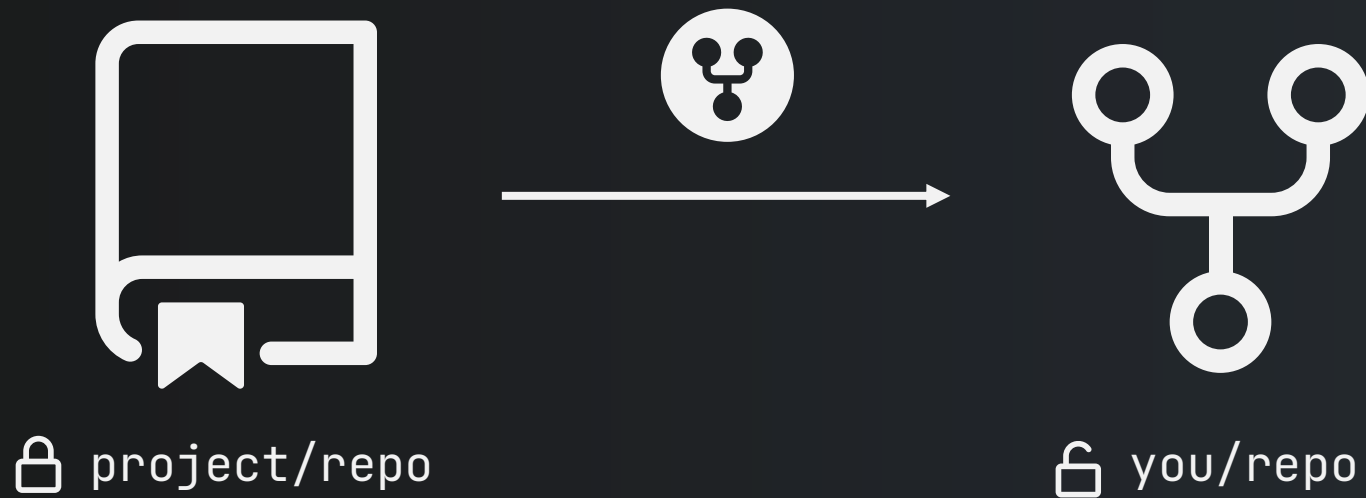
- [I Have a Question](#i-have-a-question)
- [I Want To Contribute](#i-want-to-contribute)
- [Reporting Bugs](#reporting-bugs)
- [Suggesting Enhancements](#suggesting-enhancements)
- [Your First Code Contribution](#your-first-code-contribution)
- [Improving The Documentation](#improving-the-documentation)
- [Styleguides](#styleguides)
- [Commit Messages](#commit-messages)
- [Join The Project Team](#join-the-project-team)

...
```

Issues

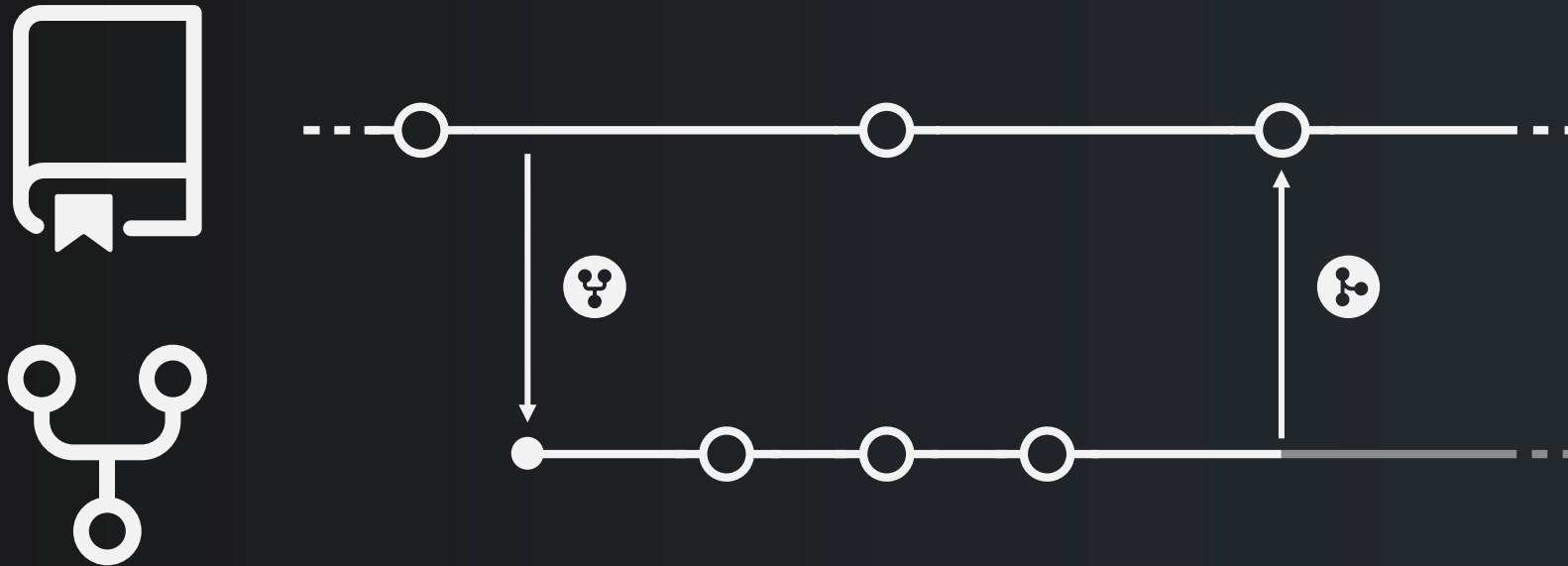


Forks



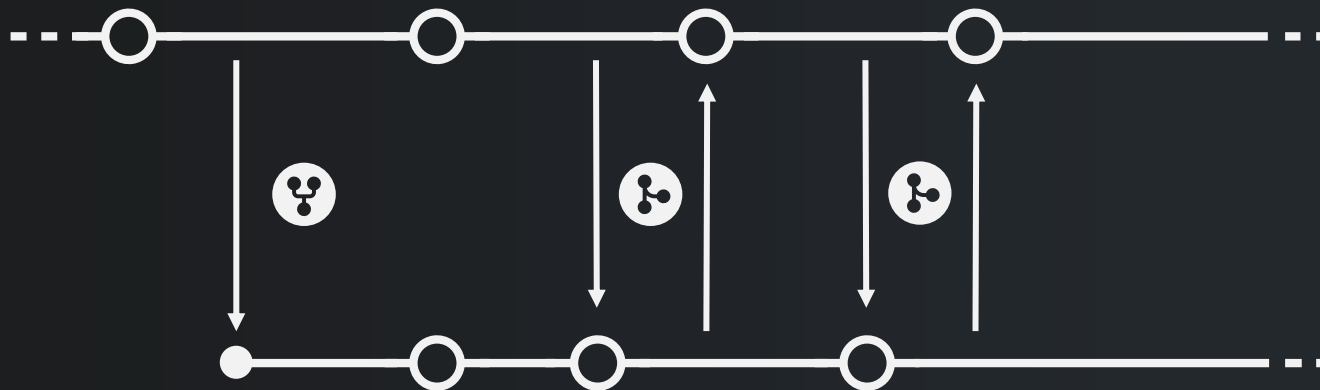
Forks

Timeline of a fork



Forks

Friendly forks



Branches

Branching and merging



Branches

Popular naming conventions

Persistent branches

main
master
dev
stable
v12.3.x
release-1.23

Feature branches

⦿ #123: Abc!
abc
123-abc
123-fix-abc
UserName/issue-123
UserName/123/fix-abc

Commits

Single Responsibility Principle



Atomic commits:

Minimal
Complete
Isolate

Commits

Commit messages

Short (max 50 chars), imperative, capitalized, no period

```
1  Add early return to isValid() method for clarity
2
3  When the conditional is false, we return the value immediately. Because no
4  further calculation is necessary in this situation, we can return the value
5  immediately. This doesn't change the behavior of the method but makes it
6  easier for a human to understand.
7
8  See #11
```

Wraps at 72~80 chars,
explains the thinking
behind the commit

Reference to issue/ticket

Blank lines to separate sections

Commits

conventionalcommits.org / Angular convention

The diagram shows a conventional commit message with labels pointing to its components:

- type**: points to `feat(api)!`
- scope**: points to `api`
- breaking changes indicator**: points to `!`
- description**: points to `send an email when a product is shipped`
- body**: points to the paragraph starting with `The previous notification system...`
- breaking changes footer**: points to `BREAKING CHANGE: web notifications won't work anymore`

```
1 feat(api)!: send an email when a product is shipped
2
3 The previous notification system was not reliable enough, and customers
4 were missing on important info about their shipments. This should make
5 notifications more consistent.
6
7 BREAKING CHANGE: web notifications won't work anymore
8
9 Refs: #123, #456
```

Commits

GPG signing



```
$ git commit -m "Not me!" --author "Somebody Else <their@mail.com>"  
$ git push  
  
> remote: OK!
```



Not me!

Somebody Else committed now



9674adb



Commits

GPG signing



Upload your public GPG key



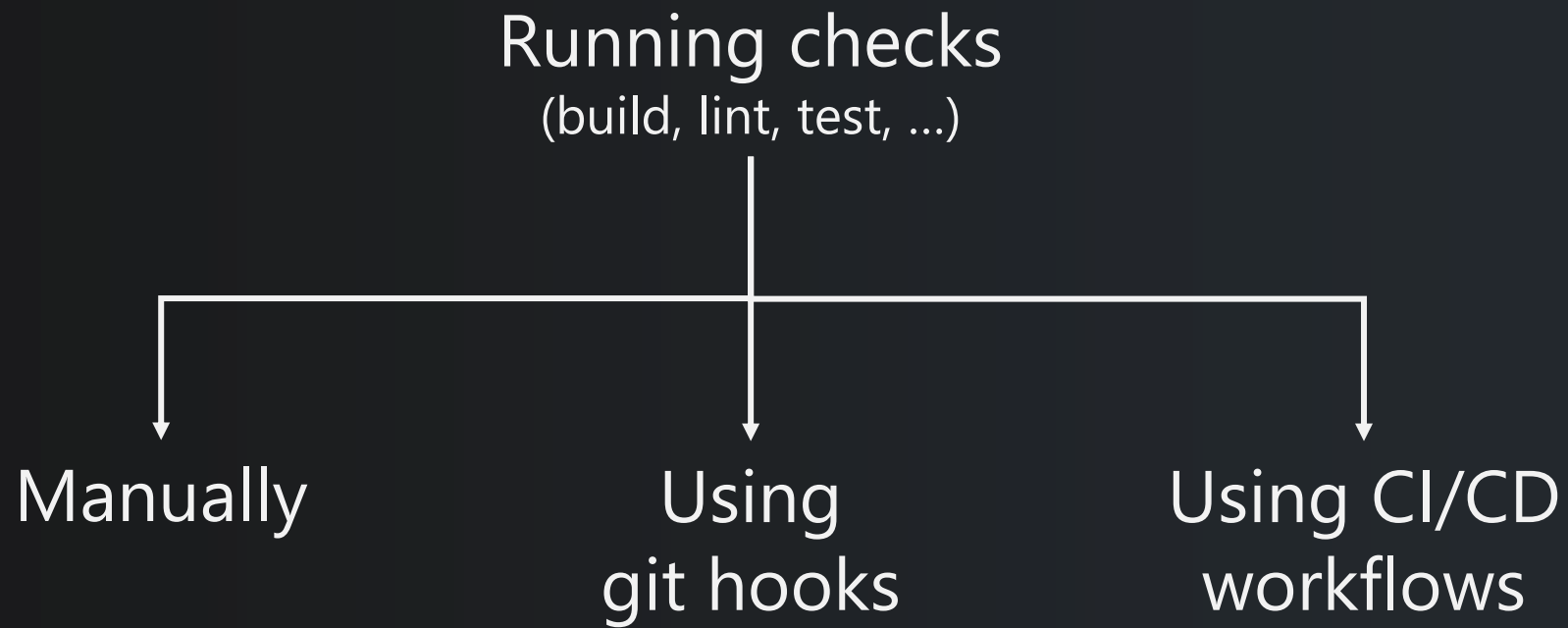
Sign your commit



Commits will now be marked as
either "Verified" or "Unverified"

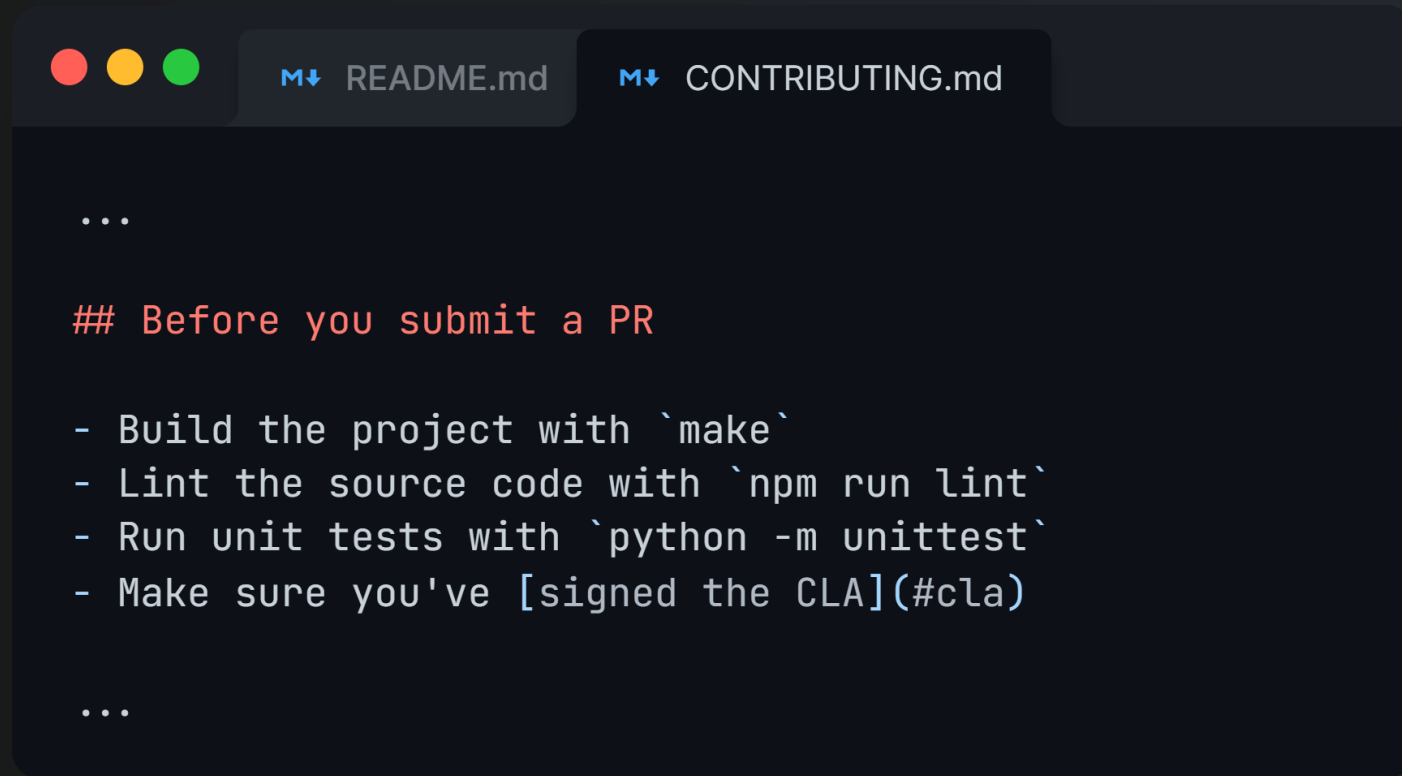


Checks



Checks

Running manual checks




```
...  
  
## Before you submit a PR  
  
- Build the project with `make`  
- Lint the source code with `npm run lint`  
- Run unit tests with `python -m unittest`  
- Make sure you've [signed the CLA](#cla)  
  
...
```

Checks

Using git hooks

pre-commit
commit-msg
post-commit
update
pre-push
pre-receive
...

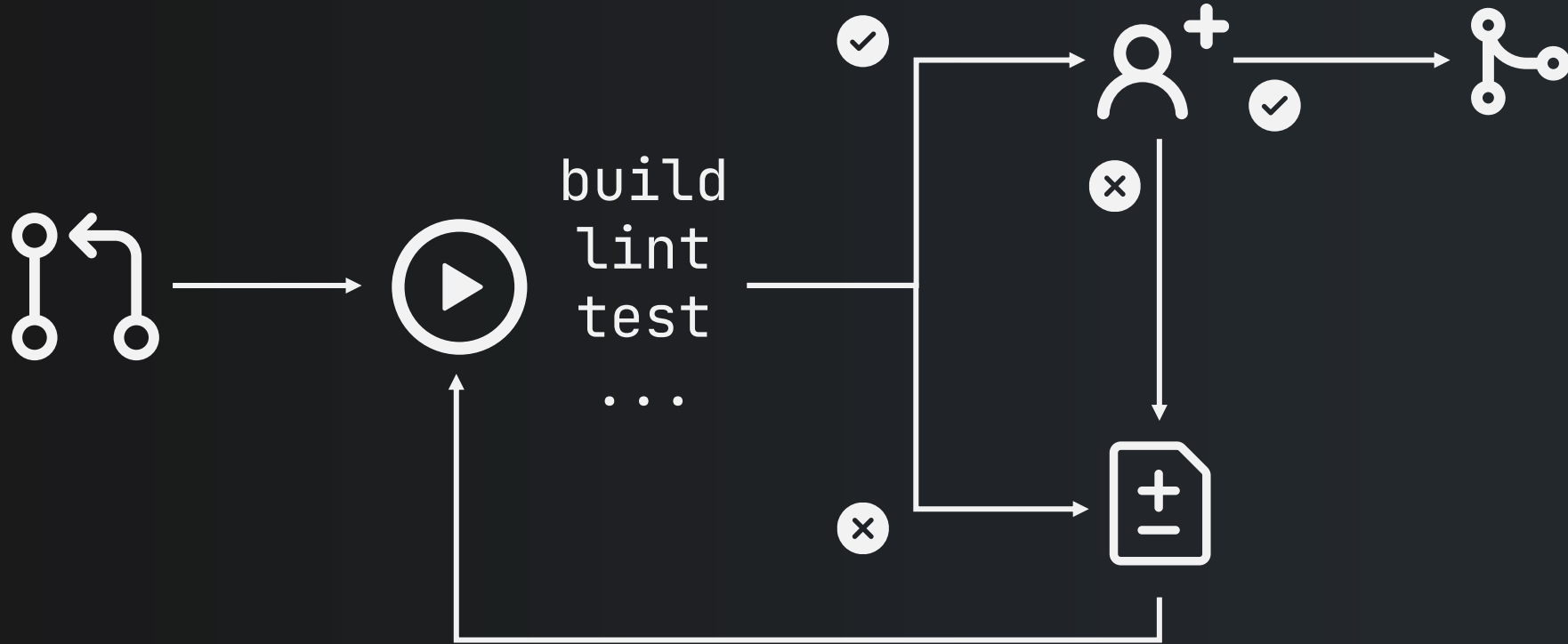


```
.git/hooks/pre-commit

1  #!/bin/sh
2
3  echo "You are about to commit" $(git diff --cached
  --name-only --diff-filter=ACM)
4  echo "to" $(git branch --show-current)
5
6  while : ; do
7      read -p "Do you really want to do this? [y/n] "
      RESPONSE < /dev/tty
8      case "${RESPONSE}" in
9          [Yy]* ) exit 0; break;;
10         [Nn]* ) exit 1;;
11     esac
12 done
```

Checks

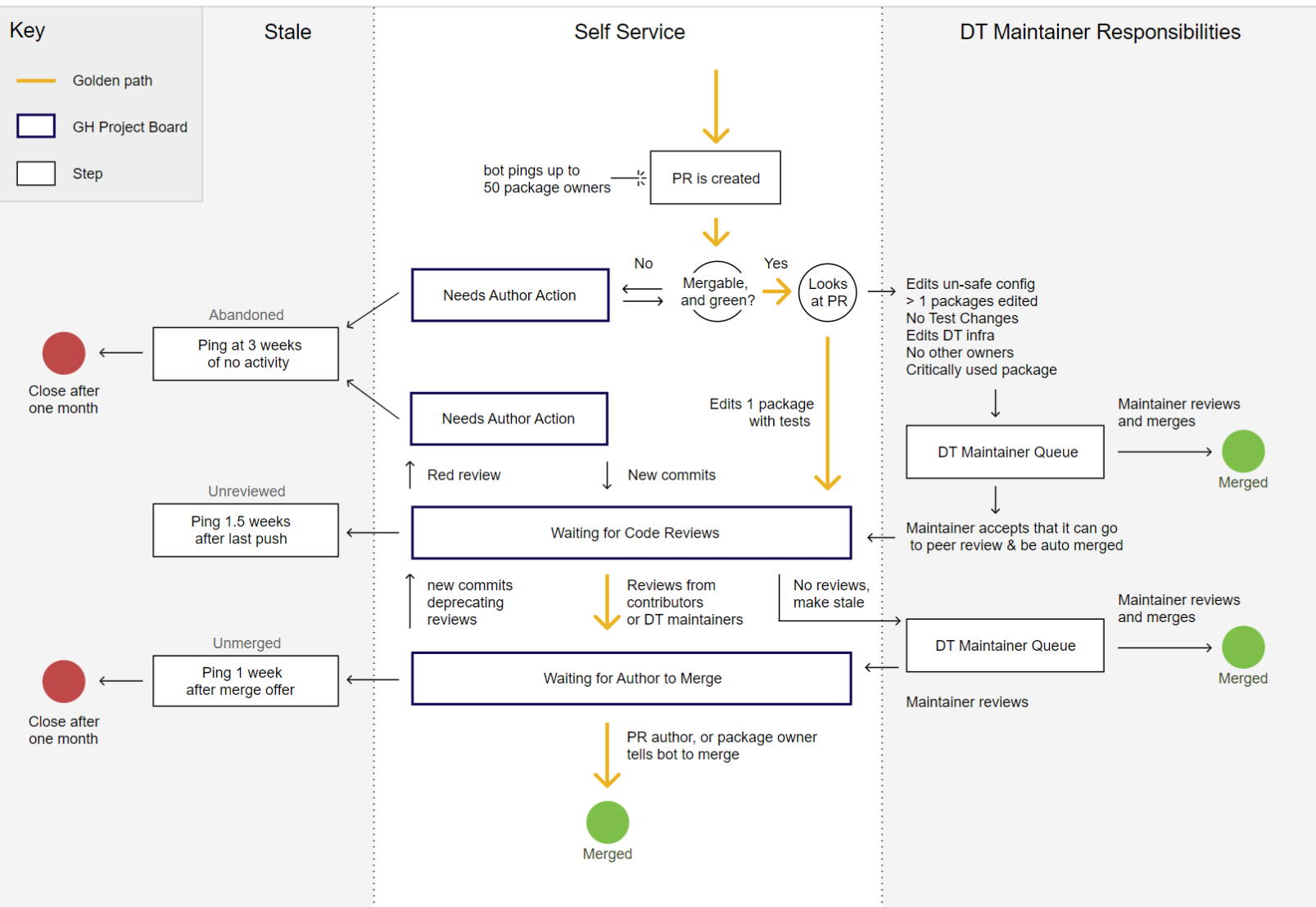
Using CI/CD workflows



Checks

Using CI/CD workflows

The lifecycle of a Pull Request to DefinitelyTyped/DefinitelyTyped



Times are approximated

Accurate as of December 2020

Based on DefinitelyTyped/dt-mergebot/src/compute-pr-actions.ts

Checks

Unit testing



Checks

Linting

```
1 function HelloWorld({greeting = "hello", greeted = "World", silent = false,
2 onMouseOver,}) {
3   if(!greeting){return null};
4
5   // TODO: Don't use random in render
6   let num = Math.floor(Math.random() * 1E+7).toString().replace(/\.d+/ig, "")
7
8   return <div className='HelloWorld' title={`You are visitor number ${ num }`}
9     onMouseOver={onMouseOver}>
10     <strong>{ greeting.slice( 0, 1 ).toUpperCase() +
11       greeting.slice(1).toLowerCase() }</strong>
12     <em>
13       { greeted }
14     </em>
15     { (silent)
16       ? "."
17       : "!"}
18
19   </div>;
20
21 }
```



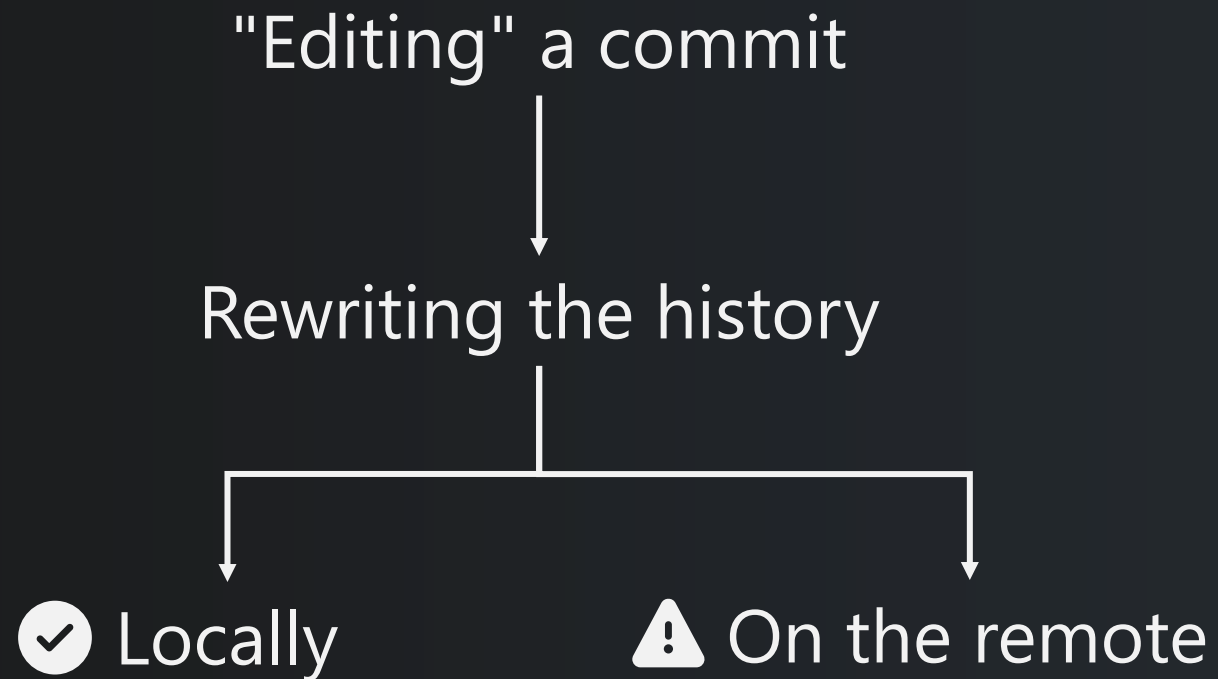
```
1 function HelloWorld({
2   greeting = "hello",
3   greeted = "World",
4   silent = false,
5   onMouseOver,
6 }) {
7   if (!greeting) {
8     return null;
9   }
10
11   // TODO: Don't use random in render
12   let num = Math.floor(Math.random() * 1e7)
13     .toString()
14     .replace(/\.d+/gi, "");
15
16   return (
17     <div
18       className="HelloWorld"
19       title={`You are visitor number ${num}`}
20       onMouseOver={onMouseOver}
21     >
22       <strong>
23         {greeting.slice(0, 1).toUpperCase() +
24         greeting.slice(1).toLowerCase()}
25       </strong>
26       {greeting.endsWith(",") ? (
27         " "
28       ) : (
29         <span style={{ color: "grey" }}>{greeting.endsWith(",") ? (
30           " "
31         ) : (
32           <span style={{ color: "grey" }}>{greeting.endsWith(",") ? (
33             " "
34           ) : (
35             <span style={{ color: "grey" }}>{greeting.endsWith(",") ? (
```


Pull/merge requests




Fixing mistakes

Disclaimer



Disclaimer

SHA generated with `git hash-object`



```
$ git cat-file -p 7f54a437d87cd1f241cfb893c4823bc7e60c19ec
tree 04b0192c1c88ac1c1a96f386e84e5388ef8a509a
parent 26349a25253f9b316db1a5d3c3f23c1ca5ca4e0e
author Author Name <their@email.com> 1651163875 +0200
committer Author Name <their@email.com> 1651163875 +0200

Commit message...
```

Commit
tree

Parent
commit

Commit
info

Editing the last commit

`git commit --amend`

```
$ git show -s --online
6773ad2 (HEAD → main, origin/main, origin/HEAD) A file is missing!

$ git add missing_file
$ git commit --amend -m "New commit message"
...

$ git show -s --online
98aysd7 (HEAD → main, origin/main, origin/HEAD) New commit message
```

Editing previous commits

`git rebase -i`

```
$ git log --oneline
6773ad2 (HEAD → main) docs(README): update examples
06d5ce7 release: v9.1.1
e401f2b chore(deps-dev): remove unused dependency
3a7c05e chore: remove changelog
385d2d6 docs: add ManuelRauber as a contributor for code (#441)
...
```

Editing previous commits

`git rebase -i`

```
$ git rebase -i HEAD~5
pick 385d2d6 docs: add ManuelRauber as a contributor for code (#441)
pick 3a7c05e chore: remove changelog
pick e401f2b chore(deps-dev): remove unused dependency
pick 06d5ce7 release: v9.1.1
pick 6773ad2 docs(README): update examples

# ...
```

Editing previous commits

`git rebase -i`

```
$ git rebase -i HEAD~5
drop    385d2d6 docs: add ManuelRauber as a contributor for code (#441)
pick    3a7c05e chore: remove changelog
squash  e401f2b chore(deps-dev): remove unused dependency
reword  06d5ce7 release: v9.1.1
edit    6773ad2 docs(README): update examples

# ...
```


Editing previous commits

```
git rebase -i
```

```
$ git log --oneline
87b0468 (HEAD → main) docs(README): update examples and add image
aa6654b v9.1.1
8b0e82e chore: remove changelog and unused dependencies
...
```

Cherry-picking

`git cherry-pick`

```
$ git branch
  feature-branch
* main

$ git log --oneline
a3e21e9 (HEAD → main) feat: feature commit
6773ad2 docs(README): update examples
...
```

Cherry-picking

git cherry-pick

```
$ git checkout feature-branch
Switched to branch 'feature-branch'

$ git cherry-pick a3e21e9
[feature-branch bfbadab] feat: feature commit
Date: Fri Oct 19 21:00:42 2022 +0200
1 file changed, 1 insertion(+)
create mode 100644 file.txt

$ git log --oneline
bfbadab (HEAD → feature-branch) feat: feature commit
...
```

Cherry-picking

git cherry-pick

```
$ git checkout main
Switched to branch 'main'

$ git reset --hard HEAD~1
HEAD is now at 6773ad2 docs(README): update examples

$ git log --oneline
6773ad2 (HEAD → main) docs(README): update examples
...
```

Restoring "lost" refs

git reflog

```
$ git log --oneline
1f40d85 (HEAD → main) fix: super important commit
6773ad2 docs(README): update examples
...

$ git reset --hard HEAD~1
HEAD is now at 6773ad2 docs(README): update examples
```

Restoring "lost" refs

git reflog

```
$ git reflog
6773ad2 (HEAD → main) HEAD@{0}: reset: moving to HEAD~1
1f40d85 HEAD@{1}: commit: fix: super important commit
...

$ git branch important-fix 1f40d85

$ git show important-fix --oneline
1f40d85 (important-fix) fix: super important commit
...
```

Tracking down bugs

git bisect

```
$ git bisect start
status: waiting for both good and bad commits

$ git bisect bad
status: waiting for good commit(s), bad commit known

$ git bisect good v9.0.0
Bisecting: 30 revisions left to test after this (roughly 5 steps)
[66ecef47895e388b5231ef2a3e841ba35d2a158a] 9.0.1

$ test
$ git bisect (good|bad)
...
```

Tracking down bugs

git bisect

```
...
$ git bisect bad
Bisecting: 0 revisions left to test after this (roughly 1 step)
[4250682bb5a58c16de35907e9c9380f5b93a8a05] chore: update lockfile

$ git bisect good
4c184a79ea0b56a2abd5dd2d812c147a5fd82c17 is the first bad commit
commit 4c184a79ea0b56a2abd5dd2d812c147a5fd82c17
Author: Federico Grandi <federicograndi@duck.com>
Date:   Fri Mar 18 11:22:57 2022 +0100

    chore: bump simple-git (#380)

...
```


Thanks!

Any questions?

Federico Grandi - @EndBug - federicograndi@duck.com