

git & devops

phil

whoami

phil

experience with git: 5 years

experience with devops: a little

the internet is your best friend when you don't have an answer

slides: <https://x4m3.rocks/talks/git-devops.pdf>



on today's program

- git
- devops

wtf is git?

scm: source control management

- bazaar (ubuntu)
- cvs (netbsd, openbsd)
- git (linux kernel, a billion projects)
- mercurial (mozilla, facebook)
- apache subversion (webkit)

git history

created by Linus Torvalds to maintain the kernel in april 2005 (**15 years ago!**)

- fast
- distributed (not centralized)
- no corruptions

2 weeks to get something working

1 month later: kernel 2.6.12 released with git

<https://github.com/git/git/tree/e83c5163316f89bfbde7d9ab23ca2e25604af290>

Git Source Code Mirror - This is a publish-only repository and all pull requests are ignored. Please follow Documentation/SubmittingPatches procedure for any of your improvements.

c shell

1 commit

5 branches

0 packages

768 releases

1,356 contributors

View license

Tree: e83c516331

New pull request

Create new file

Upload files

Find file

Clone or download



Linus Torvalds

Initial revision of "git", the information manager from hell

234

Latest commit e83c516 on Apr 8, 2005



Makefile

Initial revision of "git", the information manager from hell

15 years ago



README

Initial revision of "git", the information manager from hell

15 years ago



cache.h

Initial revision of "git", the information manager from hell

15 years ago



cat-file.c

Initial revision of "git", the information manager from hell

15 years ago



commit-tree.c

Initial revision of "git", the information manager from hell

15 years ago



init-db.c

Initial revision of "git", the information manager from hell

15 years ago



read-cache.c

Initial revision of "git", the information manager from hell

15 years ago



read-tree.c

Initial revision of "git", the information manager from hell

15 years ago



show-diff.c

Initial revision of "git", the information manager from hell

15 years ago



update-cache.c

Initial revision of "git", the information manager from hell

15 years ago



write-tree.c

Initial revision of "git", the information manager from hell

15 years ago

README

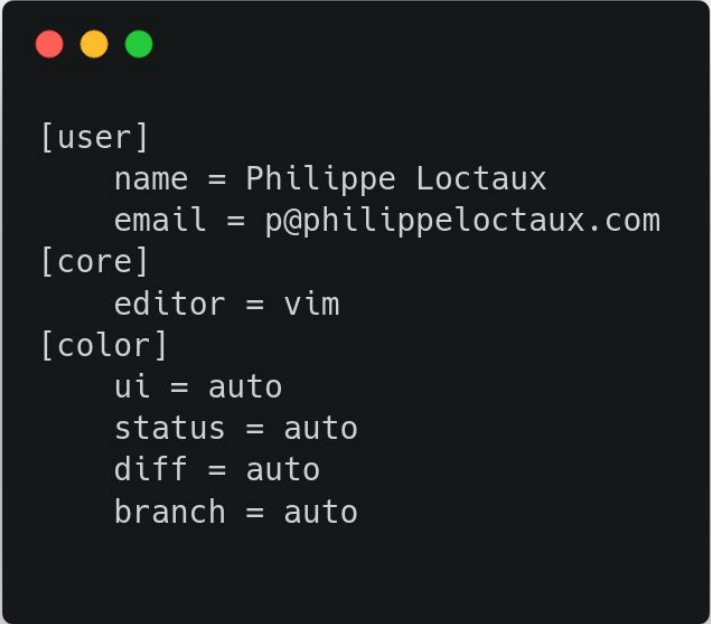
GIT - the stupid content tracker

"git" can mean anything, depending on your mood.

- random three-letter combination that is pronounceable, and not actually used by any common UNIX command. The fact that it is a mispronunciation of "get" may or may not be relevant.
- stupid, contemptible and despicable. simple. Take your pick from the dictionary of slang.
- "global information tracker": you're in a good mood, and it actually works for you. Angels sing, and a light suddenly fills the room.
- "goddamn idiotic truckload of sh*t": when it breaks

git config

~/.gitconfig



```
[user]
  name = Philippe Loctaux
  email = p@philippeloctaux.com
[core]
  editor = vim
[color]
  ui = auto
  status = auto
  diff = auto
  branch = auto
```

gui > cli

- gitg 👍
- gitkraken
- gitk
- github desktop
- vscode
- jetbrains

gui is bad for you

gui < cli

you learn by doing it manually

once you are comfortable with cli, move on with gui

git commit

commits help you **keep track** of your work

regular and **small** commits are important to see what you've done

with a nice message you know what you did

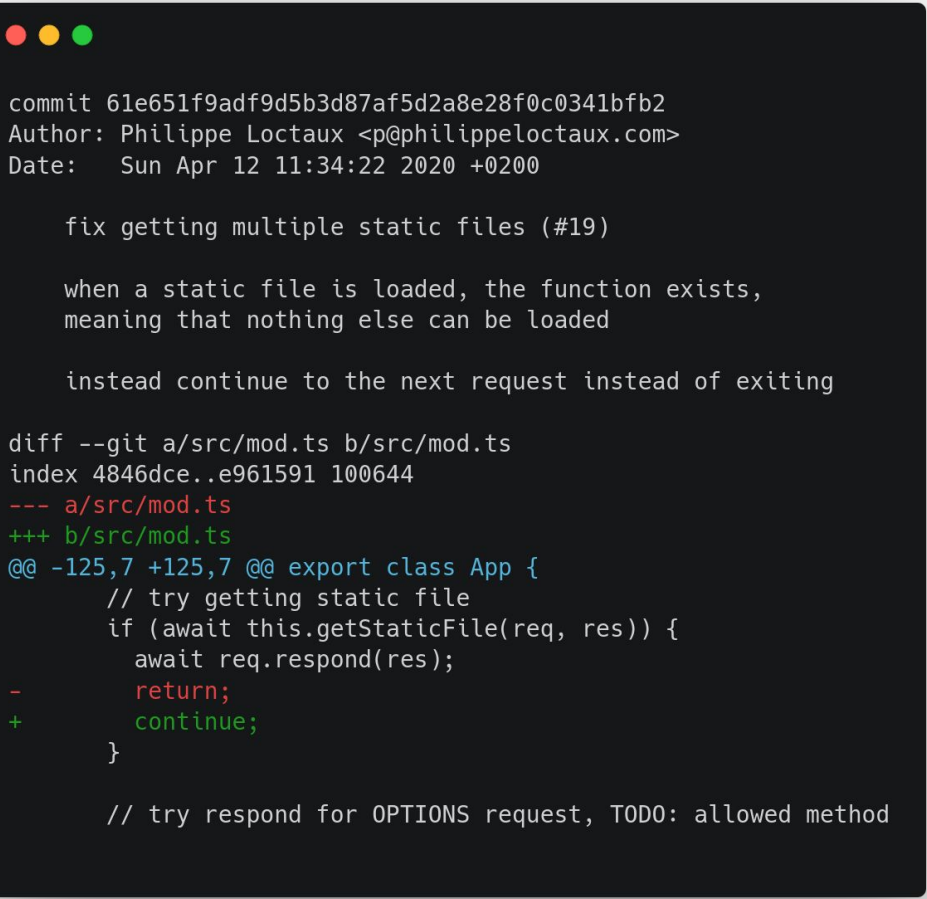
useful if you need to go back in time to fix a nasty bug

git commit message

first line: single short summary of the change

second line: blank

rest: description of the change, explain why you did that



```
commit 61e651f9adf9d5b3d87af5d2a8e28f0c0341bfb2
Author: Philippe Loctaux <p@philippeloctaux.com>
Date: Sun Apr 12 11:34:22 2020 +0200
```

```
fix getting multiple static files (#19)
```

```
when a static file is loaded, the function exists,
meaning that nothing else can be loaded
```

```
instead continue to the next request instead of exiting
```

```
diff --git a/src/mod.ts b/src/mod.ts
index 4846dce..e961591 100644
--- a/src/mod.ts
+++ b/src/mod.ts
@@ -125,7 +125,7 @@ export class App {
    // try getting static file
    if (await this.getStaticFile(req, res)) {
      await req.respond(res);
-     return;
+     continue;
    }

    // try respond for OPTIONS request, TODO: allowed method
```

git commit commands

```
git commit
```

```
git commit file1 file2
```

```
git commit -m <msg>
```

```
git commit --amend --no-edit
```

git back in time

```
git checkout <sha>
```

```
git reset HEAD^
```

```
git reset HEAD^ --hard
```

<https://github.blog/2015-06-08-how-to-undo-almost-anything-with-git/>

git branch

```
git branch
```

```
git branch my-feature
```

```
git branch -D my-feature
```

```
git merge my-feature
```

```
git branch -d my-feature
```

<https://learngitbranching.js.org/>

git diff

```
git diff
```

```
git diff master..my-feature
```

```
git diff sha..sha
```

```
git diff --staged
```


git stash

```
git stash
```

```
git stash pop
```

```
git stash list
```

```
git stash apply
```

```
git stash drop
```

git log

git log

git show sha

git blame path/to/file

git pretty log

<https://raw.githubusercontent.com/x4m3/point/master/git/gitconfig>



```
[pretty]
  custom = "%C(magenta)%h %C(red)(%an) %C(yellow)%ar %C(green)%s %Creset%d"
  #
  #
  #
  #
  #
  #
[alias]
  l = log --graph --pretty=custom
```

The image shows a terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. The terminal displays the output of the `git config` command for the `pretty` and `alias` sections. The `custom` format string is color-coded to match the output it produces: magenta for the hash, red for the author name, yellow for the relative date, green for the message, and reset for the commit count. Vertical lines and labels are used to map parts of the format string to their corresponding output: `%h` to hash (abbreviated), `(%an)` to author name, `%ar` to date (relative), `%s` to message, and `%Creset%d` to decorations.

git more

<https://ohshitgit.com/>

<https://git-scm.com/book/en/v2>

if it breaks `rm -rf` and start again

`man git`

on today's program

- — git
- **devops**

wtf is devops?

software development (**dev**) + information technology operations (**ops**)

1. **coding**
2. **building**
3. **testing**
4. packaging
5. releasing
6. configuring
7. monitoring

wtf is ci/cd?

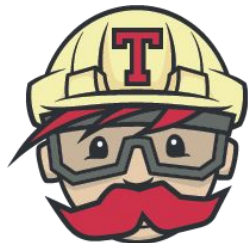
Continuous Integration: make sure it builds and passes the tests

Continuous Delivery: publish easily

continuous integration



GitHub



Travis CI



GitHub actions

free for open source, 2000 minutes for free accounts

3000 minutes for pro accounts (free for epitech students)

actions triggered by events (push on branch, pull request, etc)

works with nodejs, python, c, c++, java, php, rust, android, ios, etc

feedback on builds

The screenshot shows a GitHub Actions workflow run for the repository 'x4m3 / arcade'. The workflow is named 'updated window resolution: aligning with the interface'. The job 'C++ build' is shown as successful, with a duration of 4 minutes and 22.0 seconds. The job steps include 'Set up job', 'Run actions/checkout@v2', 'Install libraries', and 'make'. The 'make' step shows the compilation of various C++ files, including 'core/main.cpp', 'core/Exception/exception.cpp', 'core/Parser/Parser.cpp', 'core/libloader/libloader.cpp', and 'lib/SFML/SFML.cpp'. The output of the 'make' step is visible, showing the compilation of 'lib/SFML/SFML.cpp' and the creation of 'lib/lib_arcade_sfml.so'.

[x4m3/arcade] Run failed: C++ build - master (a2807ec)

Philippe Lactaux notifications@github.com
2 recipients: x4m3/arcade Cc: CI activity

Run failed for master (a2807ec)

Repository: x4m3/arcade
Workflow: C++ build
Duration: 4 minutes and 22.0 seconds
Finished: 2020-04-01 09:00:14 UTC

[View results](#)

Jobs:

- ca395085-040a-526b-2ce8-bdc85f692774 failed (1 annotation)

You are receiving this because this workflow ran on your branch.
Manage your GitHub Actions notifications [here](#).



[Click here to Reply](#), [Reply to all](#) or [Forward](#)

get actions

GitHub Marketplace

<https://github.com/marketplace?type=actions>

<https://github.com/sdras/awesome-actions>

Marketplace / Search results

Types

Apps

Actions ×

Categories

API management

Chat

Code quality

Code review

Continuous integration

Dependency management

Deployment

IDEs

Learning

Localization

Mobile

Monitoring

Project management

Publishing

Recently added

Security

Support

Testing

Utilities

Q Search for apps and actions

Actions

An entirely new way to automate your development workflow.

3399 results filtered by Actions ×



Assignee to reviewer

By abinoda

Automatically create review requests based on assignees

116 stars



Automatic Revert

By srt32

Automatically revert a commit on '/revert' comment

86 stars



ClearlyNoticed Action

By dabutvin

Maintain a NOTICE file based on your package-lock.json

14 stars



Hugo Actions

By srt32

Commands to help with building Hugo based sites

56 stars



Node App Helper Actions

By guahanweb

Provides some helper scripts to aid in basic Node.js app delivery

29 stars



Publish to Rubygems

By cadwallion

Build and publish your gem to Rubygems

23 stars



Python Style Checker

By andymckay

Run PyCodeStyle on your Python

41 stars



Rubocop checks

By gimenete

Lint your Ruby code in parallel to your builds

68 stars



Release Notifier Action

By bitoiu

Notifies developers on release with release notes via e-mail

64 stars



Snyk CLI Action

By clarkio

Run the Snyk CLI

20 stars



Actions for Discord

By Ilshidur

Outputs a message to Discord

103 stars



Bump Git Submodules

By domdere

Bump git submodules on '/submodules' comment

28 stars

what can you do?

- compile code
- multiple architectures
- different operating systems
- coding style
- unit tests
- code statistics
- security tests
- package application
- deploy to production
- push to another repo

example of workflow

1. compile code
2. check for epitech coding style
3. run unit tests
4. if everything passes push to **git.epitech.eu**
5. send message to discord / telegram / teams

let's go custom

`.yaml` or `.yml` files in `.github/workflows`

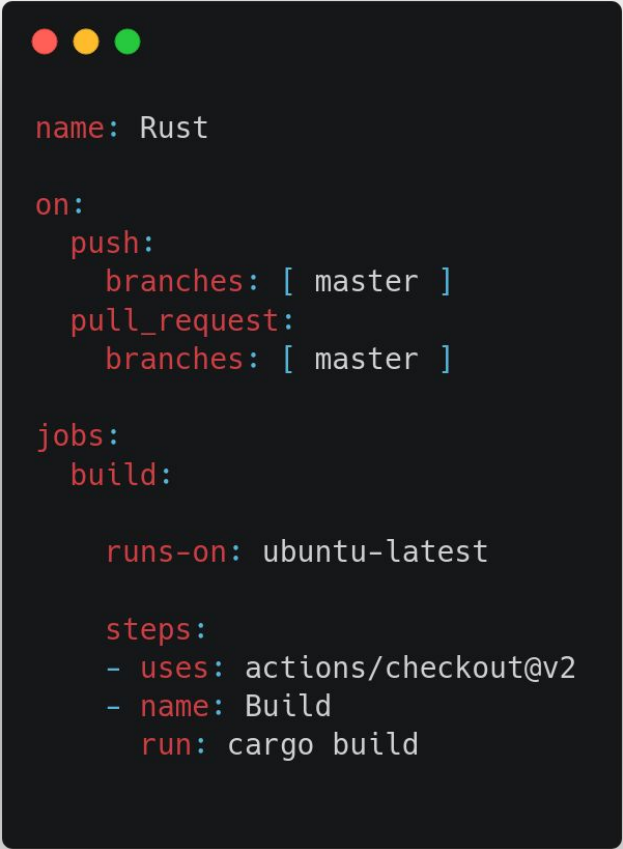
<https://help.github.com/en/actions/reference/workflow-syntax-for-github-actions>

git remote

```
git remote add github git@github.com:x4m3/repo.git
```

```
vim .git/config
```

example of workflow



```
name: Rust

on:
  push:
    branches: [ master ]
  pull_request:
    branches: [ master ]

jobs:
  build:

    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v2
      - name: Build
        run: cargo build
```

The image shows a GitHub Actions workflow definition for a Rust project. It is displayed in a dark-themed code editor window with three colored window control buttons (red, yellow, green) in the top-left corner. The workflow is named 'Rust' and is triggered on 'push' or 'pull_request' events for the 'master' branch. It consists of a single job named 'build' that runs on 'ubuntu-latest'. The job has two steps: 'checkout' using 'actions/checkout@v2' and 'Build' which runs the command 'cargo build'.

let's go

1. compile code
2. check for epitech coding style
3. run unit tests
4. if everything passes push to **git.epitech.eu**
5. send message to discord / telegram / teams

thanks

<https://x4m3.rocks/talks/git-devops.pdf>