

git & devops2

phil

whoami

phil

okay knowledge of git

a bit of experience in devops

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

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/e83c5163316f89fbde7d9ab23ca2e25604af290>

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

```
git config --global user.name "Philippe Loctaux"
```

```
git config --global user.email p@philippeloctaux.com
```

```
git config --global core.editor emacs
```

```
git config --global color.ui auto
```

CLI vs GUI

CLI: Command Line Interface

GUI: Graphical User Interface

Learn with the CLI first, because it forces you to understand what you are doing

Once you are comfortable with the CLI, use a GUI (because it's faster)

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

	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAAAAAA	3 HOURS AGO
○	ADKFJSLKDFJSDKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

git commit message

first line: single short summary of the change

second line: blank

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

explain what you did

“Like your math professor, show me your work, show the individual steps of what you did”

Your commit message can be used for documentation

Useful when something breaks and you need to get some context

Summarize changes in around 50 characters or less

More detailed explanatory text, if necessary. Wrap it to about 72 characters or so. In some contexts, the first line is treated as the subject of the commit and the rest of the text as the body. The blank line separating the summary from the body is critical (unless you omit the body entirely); various tools like ``log``, ``shortlog`` and ``rebase`` can get confused if you run the two together.

Explain the problem that this commit is solving. Focus on why you are making this change as opposed to how (the code explains that). Are there side effects or other unintuitive consequences of this change? Here's the place to explain them.

Further paragraphs come after blank lines.

- Bullet points are okay, too
- Typically a hyphen or asterisk is used for the bullet, preceded by a single space, with blank lines in between, but conventions vary here

If you use an issue tracker, put references to them at the bottom, like this:

Resolves: #123
See also: #456, #789

git commit

Don't use `git commit -m`

Use `git commit` instead

If you forgot something or made a typo in your commit: `git commit --amend`

git log

```
git log
```

```
git show sha
```

```
git blame path/to/file
```

SHA?

Secure Hash Algorithms

Used to keep data safe (such as passwords)

Can be used for integrity of content (in case of git)

Git has a hash of everything: commits, trees, blobs, etc

In git you will find what you want easily it's hash

git back in time

```
git checkout <sha>
```

```
git reset HEAD~1
```

```
git reset --hard HEAD~1
```

```
git revert <sha>
```

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

git branch

```
git branch my-feature
```

```
git switch my-feature
```

```
git merge my-feature
```

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

```
git push origin --delete 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 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 diagram illustrates the mapping of format specifiers in the custom log format to their corresponding Git log output fields. The format string is: `"%C(magenta)%h %C(red)(%an) %C(yellow)%ar %C(green)%s %Creset%d"`. The mappings are as follows:

- `%h` (magenta) maps to `hash (abbreviated)`
- `(%an)` (red) maps to `author name`
- `%ar` (yellow) maps to `date (relative)`
- `%s` (green) maps to `message`
- `%d` (reset) maps to `decorations`

git more

<https://ohshitgit.com>

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

<https://education.github.com/git-cheat-sheet-education.pdf>

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: get code and build it

Continuous Delivery: publish easily

providers



GitLab

GitHub

GitHub actions

free for public repos, 2000 minutes per month for private repos

free for epitech projects (epitech pays for us)

actions triggered by events (ex: push on branch)

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

feedback on builds

The screenshot shows a GitHub Actions workflow for the repository 'x4m3 / arcade'. The workflow is named 'C++ build' and is triggered on a 'push' event. The workflow is currently in a 'Completed' state, having succeeded 29 days ago in 38s. The workflow steps are:

- Set up job
- Run actions/checkout@v2
- Install libraries
- make
 - Run make
 - CXX c++
 - CXXFLAGS -Wall -Wextra -I./core -I./games -I./lib -std=c++11
 - CXXFLAGS_LIB -Wall -Wextra -I./core -I./games -I./lib -std=c++11 -fpic
 - LDFLAGS -ldl
 - LDFLAGS_SFML -lsfml-graphics -lsfml-window -lsfml-system
 - LDFLAGS_SDL2 -lSDL2 -lSDL2_ttf
 - LDFLAGS_NCURSES -lncurses
 - CXX core/main.cpp
 - CXX core/Exception/exception.cpp
 - CXX core/Parser/Parser.cpp
 - CXX core/libloader/libloader.cpp
 - core/libloader/libloader.cpp: In member function 'void arcade::Libloader::close()':
 - core/libloader/libloader.cpp:33:12: warning: deleting object of abstract class type 'GameModule' which has delete game;
 - 18
 - BUILD arcade
 - CXX lib/SFML/SFML.cpp
 - lib/SFML/SFML.cpp: In constructor 'SFML::SFML()':
 - lib/SFML/SFML.cpp:20:21: warning: statement has no effect [-Wunused-value]
 - 23 EXIT_FAILURE;
 - 24
 - BUILD lib/lib_arcade_sfml.so
 - CXX lib/SDL2/SDL2.cpp
 - CXX lib/SDL2/menu.cpp
 - BUILD lib/lib_arcade_sdl2.so
 - CXX lib/ncurses/ncurses.cpp
 - BUILD lib/lib_arcade_ncurses.so
- Post actions/checkout@v2
- Complete job

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



Philippe Loctaux notifications@github.com

2 recipients · x4m3/arcade · 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

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?

- build code
- check coding style
- unit testing
- code statistics
- security tests
- package application
- deploy to production

the sky's the limit!

how to write

`.yaml` files in `.github/workflows` in your repo

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

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
```

let's go

1. compile code
2. check epitech coding style
3. run unit tests

compile code

use `make re` to build your code like the mouli

code of the workflow job:

<https://gist.github.com/x4m3/7f9a1601845024a849fd641e65d383a9>

check coding style

WARNING: does not work in a Epitech repository

[clang-format](#): tool to format code

[epitech-clang-format](#): epitech rules

place the `.clang-format` file at the root of your repo

code of the workflow job: <https://gist.github.com/x4m3/a582dae33af7da46ee084e7de73e6bb7>

run unit tests

Criterion: lib used at tek for unit testing

Learn how to write and run unit tests

Makefile: rule `tests_run` to build and run tests

code of the workflow job:

<https://gist.github.com/x4m3/a08900beda2ebe4217ccc4e36c760bc1>

thank you

<https://x4m3.rocks/talks/git-devops2.pdf>