Git Project Manager / Commission Open Source

Yann Esposito

<2018-10-25 Thu>

Code from this talk → Git Project Manager

`git` is a Distributed Concurrent Versions System

GitHub is a Centralized `git` host

Can we do without Github™?

Betteridge's law

Betteridge's law of headlines is an adage that states:

> Any headline that ends in a question mark can be answered by the word no.

. . .

Here the answer is **YES**!

- Linux (only mail)
- GHC used a self hosted instance of phabricator + trac
- many others I don't know

All right, we can, but *should we*?

```
__   _______ ____  _
\ \ / / ____/ ___|| |
 \ V /|  _| \___ \| |
  | | | |___ ___) |_|
  |_| |_____|____/(_)
```

Short History

- Internet (decentralized, email, bbs, usenet, etc...)
- P2P -> no business

- Centralized -> business, steal data!!!!
- Decentralized again!
    - cryptobulshit: crash business `$$$$ -> #!@*!`
    - bio blockchains: sustainable business `$$`
    - old fashionned style: ???

GitHub™

GitHub™: Social Network

- user management & trust
- discoverability

GitHub™: its free!

> If you're not paying for it, you're the product being sold.

$$: Pay for private repositories

GitHub™: Features!!!

From their website (in that order):

- *Code Review*: comment diffs, approve, refuse, etc...
- *Project Management*: issues, milestones, dashboard, etc...
- *Integrations*: travis, slack, etc...
- *Team Management*: access rights, community guidelines, etc...
- *Social Coding*: follow, explore, share, etc...
- *Documentation*: github pages, wiki, ...à
- *Code Hosting*: all your code in one place, tree view, blame view, etc...

GitHub™: Metas

Most GitHub™ features put data in their own internal closed representation:

- Issues
- Comments
- Pages
- Pull Request & review
- Wiki

Note there are tools to export them. Ex: migrate to Gitlab

GitHub™ is great today but can suck tomorrow

- *bloatware* remember digg, readitlater?
- *downgradeware* Swagger-UI v3 (v2 is neat), reddit new redesign (looks better, but slow)

- *payware* You rely on our feature, but now, we want you to move or to pay. Fair ;)
- *crapware* Nothing works as expected unless you pay: Twitter streaming API?
- *dieware* Remember Friendfeed? Google Reader™?
- etc...

GitHub™ force all your team member to use GitHub™

Were you already forced to:

- use PowerPoint? Excel? Word?
- code in PHP? in Java?
- work on windows? Harder need to ssh to UNIX machines?
- use Eclipse instead of vim/emacs?
- use a super complex GUI instead of a few command line tools?
- etc...

REAL STORY @WORK: github dashboard is slow & terrible for the manager.

Why you shouldn't rely too much on GitHub™?

- Github™ is great to get you started:
  - nothing to install
  - only high level interface
  - everything explained with nice docs
  - github is really a great product
- **The hidden price to pay**:
  - use closed source services
  - give freely many private infos
  - you must TRUST github for privacy, private account

Why not self hosted Gitlab then?

Gitlab is a better alternative but:

- You still keep the metas of your project in the Gitlab server in some DB.
- You still force all the member of your team to use your Gitlab version, with your Gitlab plugins, with your Gitlab settings, etc...
- Gitlab push a big warn so you are pushed to upgrade (new features & anti-features)

`git clone`

☒ code

☒ web pages

- ☐ **issues**
- ☐ **reviews**
- ☐ **comments**
- ☐ **wiki/doc**
- ☐ **hooks**

Git Project Manager

Problems

- can't clone everything
- big dependence on private tooling (that could change or being interrupted)
- force same tooling choices accross your team members

Solution

- put metas in git branches **CLONE ALL THE THINGS!!!!**

. . .

- use text files for everything **DO NOT FORCE ANY TOOL**

. . .

- only rely on conventions, better on standardized conventions **HELP TO WRITE SPECIFIC OPEN SOURCE TOOLS**

Git Project Manager `gpm`

- command line tool
- integrate your project management metas in your git repo
- automate a few common tasks
- follow a few conventions

Tool freedom

- people on the team don't need to install or use `gpm`
- they just need to follow a minimal set of conventions
- want to use other conventions? Write yourself a `gpm` in a few hours.
- but really there are *very few* conventions `gpm` follows

`gpm` conventions

- `git` as DCSV
- text files
- Project Management metas goes in the branch `gpm`

Encouraged but not enforced `gpm` conventions

- encourage to use `org-mode` format but you can change
- issues goes in `issues.org` file
- reviews goes in `reviews/` with name `<branch>-<reviewer>.org`
- docs goes in `wiki.org`
- serve goes in your XDG data dir (standard)

git is awesome!

battery included:

- `git` hooks
- `git instaweb`
- `git daemon`

org-mode is awesome

- TODO list oriented document convention
- Extremely versatile:
  - issues, bug tracking, comments
  - handling with minimal friction code reviews `org-annotate-file`
  - workflows:
    - basic trello (TODO, IN-PROGRESS, DONE)
    - scrum (EPIC / USER-STORY / etc...)
    - kanban:
      - EPIC with different statuses (prep, specified,etc..), comments
      - user stories with evaluation, different status, comments
      - QA status
      - Ops status

DEMO

Create a git project

```
mkdir -p /tmp/gpm-playground/testprj
cd /tmp/gpm-playground/testprj
echo "Hello GPM" > README
git init .
git add README
git commit -m "Initial commit"
```

`gpm init` (1)

```
gpm init
```

```
GPM -- Git Project Manager
==========================
Create a new branch gpm (be sure the branch gpm doesn't already exists)
    git checkout --orphan gpm
Switched to a new branch 'gpm'
cleanup the branch
    git rm --cached -r .
    git clean -fd
```

gpm init (2)

```
* issue.org
    git add issues.org
    git add templat
* wiki.org
    git add wiki.or
* reviews.org
    create some example review for inspiration
      reviews/write-contributing-yogsototh.org
    git add reviews
    create some review templates
      templates/new-review.org
    git add templates
```

gpm init (3)

```
* hooks/
    Copyings default hooks into the hooks directory
    git add hoo
* server init
create dir: /Users/yaesposi/.local/share/gpm/public
    git init .
    git rev-parse --show-toplevel
    git rev-parse --show-toplevel
    git clone --mirror /tmp/gpm-playground/testprj
               ~/.local/share/gpm/public/testprj.git
Cloning into bare repository '.../testprj.git'...
done.
    git update-server-info
    git commit -m 'gpm initialized'
    git checkout master
Switched to branch 'master'
```

The `gpm` branch

```
> git checktout gpm
> tree
```

```
.
├── hooks
│   ├── applypatch-msg.sample
│   ├── commit-msg.sample
│   └── ...
├── issues.org
├── reviews
│   └── write-contributing-yogsototh.org
├── templates
│   ├── new-issue.org
│   └── new-review.org
└── wiki.org

3 directories, 16 files
```

Hooks

```
> gpm hooks
Usage: gpm hooks sync
  Handle hooks for this git repository

Available options:
  -h,--help                Show this help text

Available commands:
  sync                     Synchronize hooks from gpm branch
```

`issues.org` Basic

```
#+TODO: TODO(t) STARTED(s) WAITING(w) | DONE(d) CANCELLED(c)
* Basic Usages
** TODO Do thing 3
** STARTED Do thing 2
** DONE Do thing 1
```

`issues.org` Complex workflow, review

```
 #+PROPERTY: ASSIGNEE
 #+PROPERTY: REVIEWER
 #+TODO: REVIEW(i) | MERGED(m)
 #+TODO: ACCEPTED(a) CHANGE_REQUESTED(c) QUESTION(q) FEEDBACK(f) | REFUSED(r)
 ** REVIEW Basic review process
  :PROPERTIES:
  :BRANCH:   explain-review-process
  :ASSIGNEE: yogsototh
  :END:
```

```
*** ACCEPTED Review finished
   :PROPERTIES:
   :REVIEWER: shubby
   :END:
```

`issues.org` Full Professional Usage

```
#+TAGS: epic(e) user_story(u) task(t) qa(q) ops(o)

 * Some Title                                                               :epic:
 ** Some User Story                                                        :story:
 *** Dev Task                                                           :task:dev:
 *** Document Task                                                      :task:doc:
 *** QA Task                                                             :task:qa:
 *** Ops Task                                                           :task:ops:
```

`gpm new-issue`

```
> gpm new-issue -i
```

`gpm serve`

- web interface: `git instaweb` (port 1234)
- git server: `git daemon` (port 9418)

```
Usage: gpm serve (start | stop | update | path)
  Serve the git to the web

Available options:
  -h,--help                Show this help text

Available commands:
  start                    Start to serve all gpm tracked repositories
  stop                     Stop to serve all gpm tracked repositories
  update                   Update the served git repository
  path                     Show the path of the bare repository
```

`gpm review`: classical workflow

1. dev create a new feature branch
2. reviewer review the branch
3. dev pull the `gpm` branch and `gpm retrieve` the reviews
4. dev take feedbacks into account
5. goto 2 until reviewer accept the branch
6. integration manager/dictator/lieutenant merge the branch

`gpm review`: reviewer (step 2 of previous slide)

1. reviewer pull the remote feature branch
2. gpm review start: create a local file
3. write the review: `org-annotate-file` FTW!
4. stop the review: copy the local file in gpm branch and commit it
5. `gpm update` to serve the updated `gpm` branch

Conclusion

Proof of concept

- `gpm` is a proof of concept but so simple its already usable
- git clone should provide most of your projects data
- don't enforce tooling on your team, use text files
- I advise you to use org-mode it is awesome! **REALLY!**
  - vimer? ☒ spacemacs or doom-emacs
  - IDE? ☒ switch to spacemacs eat the bullet!
  - you still can edit org-mode with notepad

Lot of things already done

- git-scm.org has plenty of resources
- git instaweb
- git daemon
- how to serve git with apaches, if you want to use another non decentralized workflow, or share hosting with a few peers

Going further: Decentralized Web

- the Internet was thought to be decentralized
- centralization of services made lot of things easy, it was fair at first
- but made us dependant and the balance is no more fair
- it is time to re-decentralize the Internet and take back control
- we shouldn't be dependant of private services
- we should pay private service, but they should adapt to us, not the other way around

Decentralized Authentication: IndieAuth

- one of your online identities = one domaine name
- serve a page with all your online identity providers and username
  - google
  - twitter
  - etc...
  - but also your GPG keys (see keybase)

Mainly you OWN & CONTROL your identity and the informations about it.

Decentralized Comments: webmention

- you host your comment
- a 3rd party website can decide to show it in its comment section

Decentralized Web

- Your content is yours (prevent site death, change it, delete it...)
- Better connection:
  - messages can go to all your services
  - use open standards
- You are in control
  - post anything, any format, no monitoring, share links.

Follow:

- https://indiewebify.me
- https://indieweb.org