

Git

September 29, 2022

Git

Git is a Version Control System!



Linus Torvalds invented Git in 2005 for managing the Linux Kernel source code.

Git is a version control system that uses DAG and Merkle trees!

VCS Recap: glossary

- **Working copy:** All project files (on your PC) tracked by the VCS.
- **Commit:** a “snapshot” of the repository at a specific moment in time.
- **Branch:** a line of development. It’s an ordered set of commits.
- **Merge:** the action of fusing two or more branches.
- **Tag:** a custom label attached to a commit.
- **Repository:** a set of commits, branches, and tags (usually for the same project).
- **Fork:** a new repository that is a copy of an existing one.
- **Pull/Merge request:** a request to merge code from a fork or branch back to the parent repository/branch.

Commit

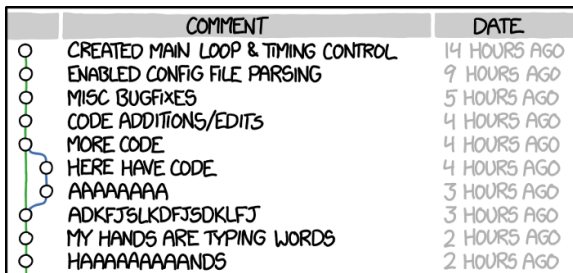


A snapshot of the project in a given moment. It can include some or all files (it's up to the user to decide).

Each commit has one or more parents. In rare case, zero!

It **requires** a meaningful message.

Commit



	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
○	AAAAAAA	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.

Image credits by XKCD - Randall Munroe

Please, don't.

Your future self will thank you.

What a git commit contains

- **Commit message**
- **Committer** and commit date
- **Author** and author date
- **Tree** (hash of all files in the commit, sort of)
- **Parent** commits

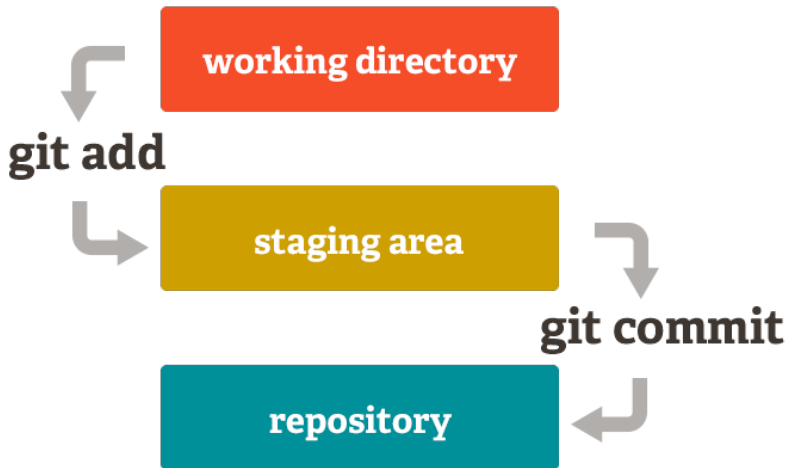
The commit ID is the SHA1 of the content.

“Orphan”

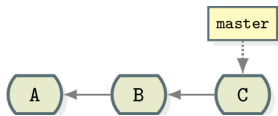
Sometimes commits have no parents. In that case, the commit is called “orphan”.

Example: the first commit in an empty repository.

Git staging area



Branch

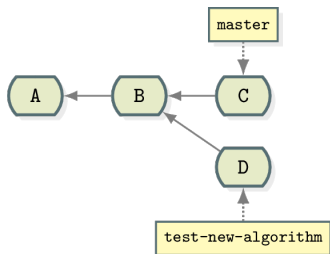


A “branch” is a line of development — an ordered set of commits.

A branch has a name and starts with a commit. Part of the branch history is shared with other branches.

Usually ends up merging again with the main line.

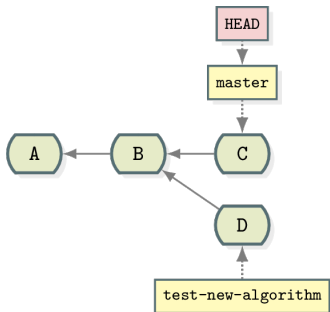
Why branching?



To develop a new feature or do experiments without interfering with others.

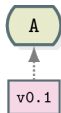
Note: the main line of development is a branch itself - nothing special!

The HEAD



- The HEAD is a pointer to the “current” commit/branch/tag.
- In other words: files in the working copy are from the commit/branch/tag pointed by HEAD.
- Special case: “detached HEAD” is when HEAD is not a branch: we can’t commit.

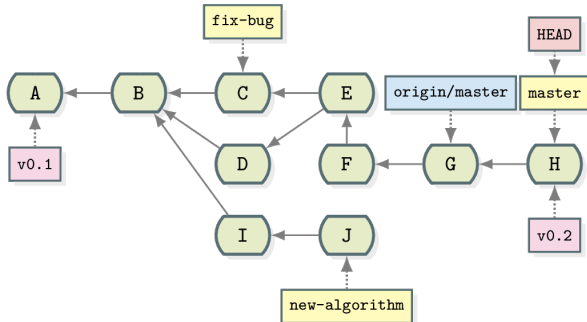
Tags



Commits can be tagged. It makes it easier to look at the repo history.

You can remove a tag, although you shouldn't.

Commit history and branches

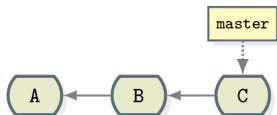


When a branch is completed and should be added to the main line: merge!

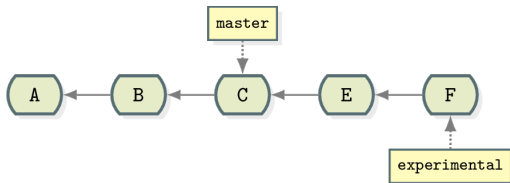
There are different ways of merging branches ("strategies"):

- Fast-forward
- Non-fast-forward
- Rebase

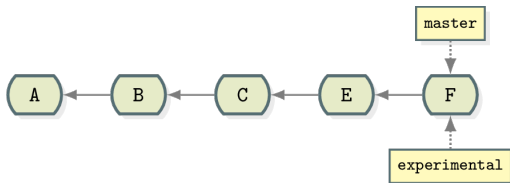
Merge - fast-forward - before



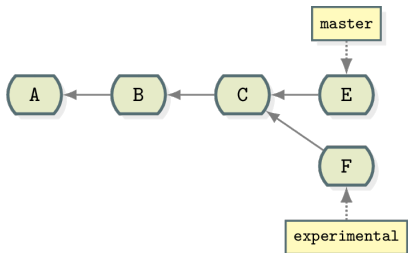
Merge - fast-forward



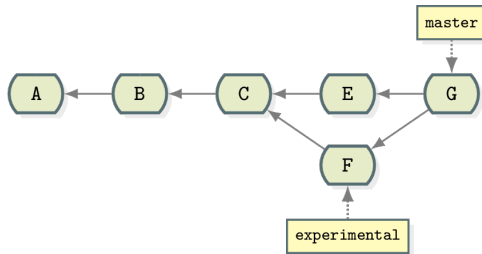
Merge - fast-forward - after



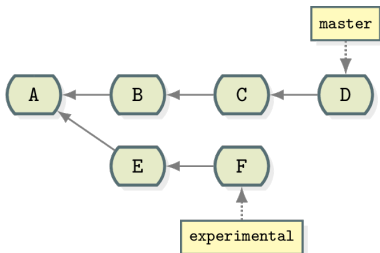
Merge - non-fast-forward



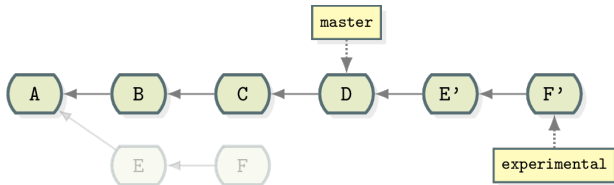
Merge - non-fast-forward - after



Rebase



Rebase - after



Merge conflicts

Merge conflicts

A “merge conflict” may happen when we merge branches using non-fast-forward or rebase strategies.

(or in some advanced functions)

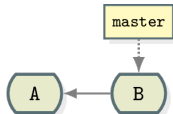
```
package main

import "fmt"

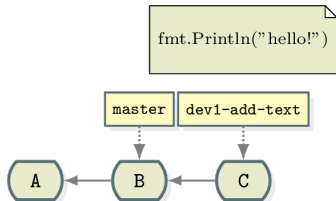
func main() {
    fmt.Println("")
}
```

Suppose that two developers change the message in parallel to two different texts.

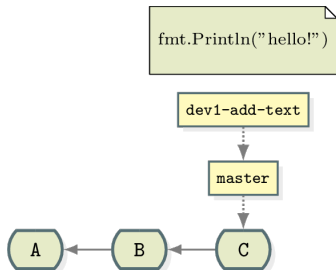
The calm before the storm



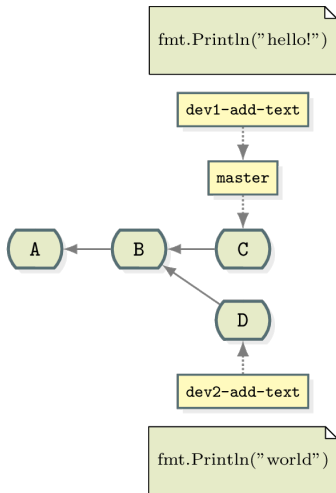
The 1st developer commits



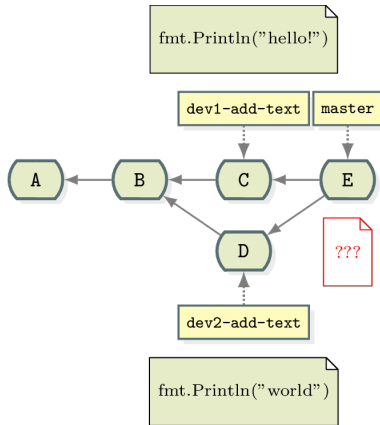
Fast-forward merge is ok...



...however 2nd dev committed in the meantime



Oops!



example.go during conflict

```
package main

import "fmt"

func main() {
<<<<<< HEAD
    fmt.Println("hello!")
=====
    fmt.Println("world")
>>>>>> dev2-add-text
}
```


How to resolve conflicts?

Three ways to resolve:

- Edit the file manually
- Keep the already-merged **file** from dev1 ("ours")
- Use the new **file** from dev2 ("theirs")

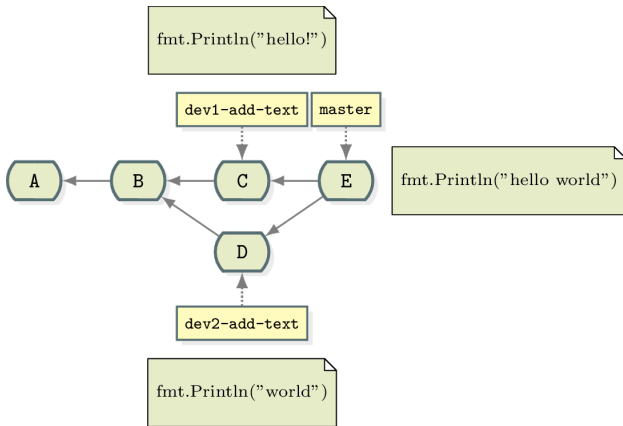
Let's fix example.go manually

```
package main

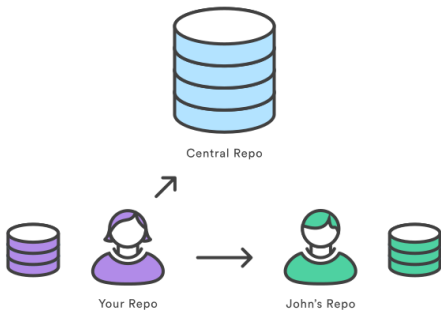
import "fmt"

func main() {
    fmt.Println("hello world!")
}
```

We edit the file and commit



Remote git repositories



Git can retrieve and send commits, tags, and branches to remote hosts (named remotes).

The default remote is named origin.

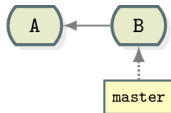
“Clone” creates a local copy of the repository, starting from a URL.

Git will download all commits, branches, and tags.

The provided URL will be the origin remote.

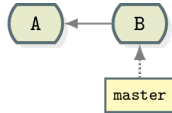
Clone

Remote



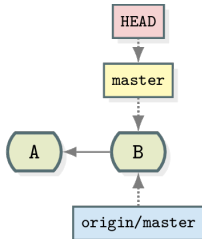
Clone

Remote



Git clone

Local



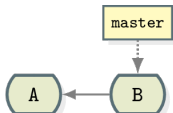
Send commits/branches/tags

You can send any commit, branch, or tag to any remote.

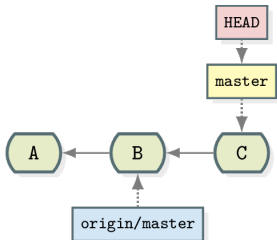
The action is named **push**.

Send commits/branches/tags

Remote

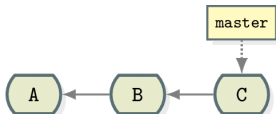


Local



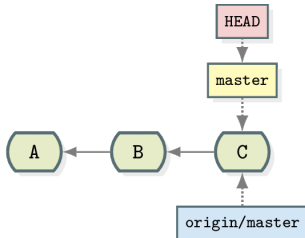
Send commits/branches/tags

Remote



Git push

Local



Retrieve commits/branches/tags

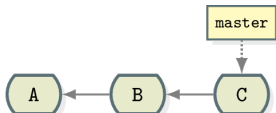
You can retrieve any commit, branch, or tag from any remote.

There are two main actions: **fetch** and **pull**.

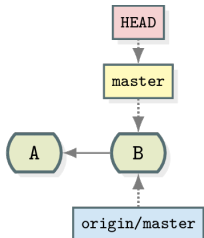
- Fetch retrieves remote commits, branches, and tags
- Pull is fetch plus a merge!

Retrieve commits/branches/tags: fetch

Remote

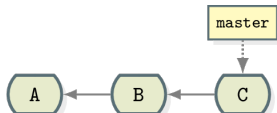


Local



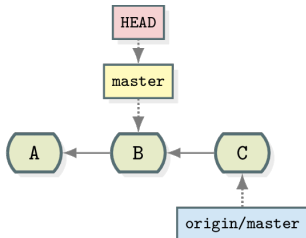
Retrieve commits/branches/tags: fetch

Remote



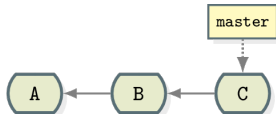
Git fetch

Local



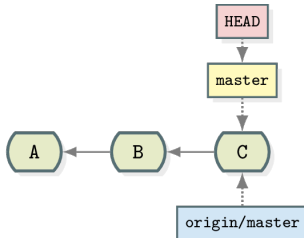
Retrieve commits/branches/tags: fetch

Remote



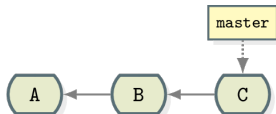
Git merge fast-forward

Local

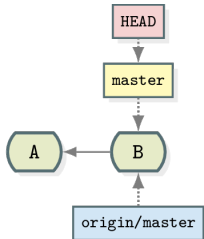


Retrieve commits/branches/tags: pull

Remote

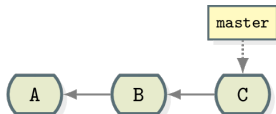


Local



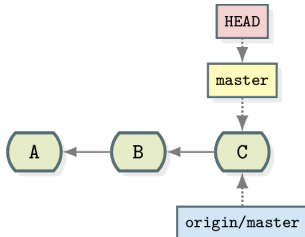
Retrieve commits/branches/tags: pull

Remote



Git pull

Local



Forges

“forges”?

A Git remote can be a server running sshd. Or even a USB key that you share with your colleagues.

However, specific platforms ease collaboration - they're named “git forges”.

Git forges: hosting services



GitLab



Gitea

- Always online git remote
- Bug tracking
- Documentation wikis
- Manage forks and pull/merge requests
- Collaborate with others
- Advanced: CI/CD, integrated IDEs, etc

Project management

The screenshot shows the GitLab web interface for the 'GitLab' project. At the top, there is a dark blue navigation bar with the GitLab logo, a search bar, and various utility icons. Below the navigation bar, the project name 'GitLab' is displayed with its Project ID (278964). The interface includes statistics for commits (302,453), branches (11,741), tags (2,096), project storage (35.8 TB), and releases (128). A description states that GitLab is an open source end-to-end software development platform. A progress bar shows the status of various pipelines, including 'pipeline running', 'Ruby Coverage unknown', and 'JS Coverage 71.75%'. Below the progress bar, there are options to view files, use the Web IDE, download, or clone the repository. A recent merge commit is shown, along with a list of project files and folders such as .github, .gitlab, .rubocop_todo, .theia, .vscode, and app.

GitLab.org > GitLab

GitLab Project ID: 278964

🔔 Star 3997 Fork 6891

🏠 302,453 Commits 11,741 Branches 2,096 Tags 35.8 TB Project Storage 128 Releases

GitLab is an open source end-to-end software development platform with built-in version control, issue tracking, code review, CI/CD, and more. Self-host GitLab on your own servers, in a container, or on a cloud provider.

pipeline running Ruby Coverage unknown JS Coverage 71.75%

master /

Merge branch 'release-tools/update-gitally' into 'master' 5722ae69

release-tools approver bot authored 53 minutes ago

Name	Last commit	Last update
.github	Rename GitLab CE to FOSS in GitHub issue t...	3 years ago
.gitlab	Fix QA image name in FOSS pipelines	2 days ago
.rubocop_todo	Re-generate RuboCop TODOs for RSpec/Cap...	3 days ago
.theia	Revert "Move host authorization to to gitpod ...	1 year ago
.vscode	Add GitLab Workflow as recommended VS C...	11 months ago
app	Merge branch '367103-migrate-card-admin-u...	1 hour ago

Project management

The screenshot displays the GitLab web interface for a project. The top navigation bar includes the GitLab logo, a search bar, and various utility icons. A left sidebar provides navigation for project information, repository, issues (45,191), merge requests (1,401), CI/CD, deployments, packages, monitor, analytics, and snippets.

The main content area shows the repository name 'gitalab' with a search bar and a 'Clone' button. Below this, a commit history table is visible, showing the most recent commit by 'approver bot' 53 minutes ago. The commit message is 'h 'release-tools/update-gitaly' into 'master''. Below the commit message, there are buttons for 'CHANGELOG', 'CONTRIBUTING', and 'CI/CD configuration'.

Last commit	Last update
Rename GitLab CE to FOSS in GitHub issue t...	3 years ago
Fix QA image name in FOSS pipelines	2 days ago
Re-generate RuboCop TODOs for RSpec/Cap...	3 days ago
Revert "Move host authorization to to gitpod ...	1 year ago
Add GitLab Workflow as recommended VS C...	11 months ago
Merge branch '367103-migrate-card-admin-u...	1 hour ago

Project management: Issues

The screenshot shows a GitHub issue page for the repository 'GitLab.org / GitLab'. The issue title is 'Pods: wrong db_config used by Rails after modifying code and model reload triggered', created 4 days ago by Pedro Pombeiro. The issue is currently open and assigned to Omar Qunsul. It is categorized as a 'bug' and has several labels: 'devops', 'verify', 'group runner', 'pods', 'active', 'section ops', 'type bug', 'workflow', 'in review', and 'Category Pods'. The description explains that when using the GDK, a worrying behavior occurs where Rails starts using the 'main' database instead of the intended 'ci' database. This happens when the code is changed or 'reload!' is called from the GDK console. The issue provides steps to reproduce the problem and lists associated files: '.envrc', 'gdk.yml', 'database.yml', and 'development.log.zip'. The issue was edited 4 days ago by Pedro Pombeiro. The right sidebar shows various metadata fields: Assignee (Omar Qunsul), Epic (None), Labels (as listed), Milestone (15.5), Iteration (None), Weight (None), Due date (None), Time tracking (No estimate or time spent), and Health status (None).

Next Search GitLab

GitLab.org > GitLab > Issues > #374753

Open Issue created 4 days ago by Pedro Pombeiro

Pods: wrong db_config used by Rails after modifying code and model reload triggered

When using the GDK, I noticed a worrying behavior where Rails started using the `main` database for all `Ci::ApplicationRecord` models.

This happens when either the code is changed, or `reload!` is called from the GDK console. CI models will start pointing to the `main` database. Running `ActiveSupport::Reloader.after_class_unload { Gitlab::Database::LoadBalancing::Setup.new(Ci::ApplicationRecord).setup }` will fix the problem, so probably this should be run automatically when the models are reloaded.

Steps to reproduce

On the GDK console, run `Ci::Runner.first` and notice the `db_config_name:ci` comment. After running `reload!`; `Ci::Runner.first`, the model will point to `db_config_name:main`.

Files

- [.envrc](#)
- [gdk.yml](#)
- [database.yml](#)
- [development.log.zip](#)

Edited 4 days ago by Pedro Pombeiro

Drag your designs here or [click to upload](#).

Tasks (0)

Add a to do

Assignee
Omar Qunsul

Epic
None

Labels
devops verify group runner
pods active section ops type bug
workflow in review Category Pods

Milestone
15.5

Iteration
None

Weight
None

Due date
None

Time tracking
No estimate or time spent

Health status
None

Project management: Pull/Merge request

The screenshot shows a GitLab Merge Request (MR) page. At the top, the navigation bar includes the GitLab logo, a search bar, and user avatars. The breadcrumb trail is: GitLab.org > GitLab > Merge requests > 198910. The main heading is "Draft: Add JH dir in pipeline rules" with a "Code" dropdown menu. Below the heading, it says "Open" and "Song Huang requested to merge gitlab-jh/jh-team/gitlab:f... into master 3 minutes ago". There are tabs for Overview (1), Commits (1), Pipelines (1), and Changes (1). The "What does this MR do and why?" section contains the text: "Hello, this is JIHu team, the config/feature_flags directory also exists in the JIHu edition, so we need to add the jh dir to the pipeline rules like this MR did." The "MR acceptance checklist" section has a checked box: "I have evaluated the MR acceptance checklist for this MR." Below this, it says "/cc @daveliu" and "Edited 2 minutes ago by Song Huang". A light blue box states: "Members who can merge are allowed to add commits." A pipeline status bar shows "Merge request pipeline #650590377 running for 5faec629" with a progress indicator. A box indicates "Requires 1 approval from Code Owners." At the bottom, there is a "View eligible approvers" link. On the right sidebar, the "Assignee" is Song Huang, "Reviewers" are none, "Labels" include workflow, in dev, Community contribution, and JIHu contribution, "Milestone" is none, "Time tracking" has no estimate, and "3 participants" are listed.

Project management: Pull/Merge request

The screenshot shows a GitLab Merge Request (MR) interface. At the top, the navigation bar includes the GitLab logo, a search bar, and user avatars. The breadcrumb trail reads: GitLab.org > GitLab > Merge requests > 198910. The main heading is "Draft: Add JH dir in pipeline rules". A blue "Code" button is visible in the top right. Below the heading, it states "Song Huang requested to merge gitlab-jh/jh-team/gitlab:f... into master 3 minutes ago".

The "Changes" tab is selected, showing a diff for the file `.gitlab-ci/rules.gitlab-ci.yml`. The diff shows a change between the `latest version` and `master`. The changes are as follows:

Line	Old	New
519	519	@@ -519,7 +519,7 @@
520	520	- "vendor/assets/javascripts/**/*"
521	521	.feature-flag-development-config-patterns: &feature-flag-development-config-patterns
522	522	- - "{ee/}config/feature_flags/{development,ops}/*.yml"
522		+ - "{ee/,jh}config/feature_flags/{development,ops}/*.yml"
523	523	
524	524	#####
525	525	# Conditions set #

Resources

Links

- <https://git-scm.com/book>
- <https://www.atlassian.com/git/tutorials>
- <https://threesaplings.co/articles/explaining-basic-concepts-git-and-github/>
- <https://github.com/RomuloOliveira/commit-messages-guide>
- <https://nitaym.github.io/ourstheirs/>