Git and GitHub tutorial

Gennady Pospelov
Scientific Computing Group at MLZ

2nd BornAgain school and user meeting, Garching, December 2018

What is version control?

- A system that keeps records of your changes in files
- Allows you to revert changes and go back to previous state

What is version control?

- A system that keeps records of your changes in files
- Allows you to revert changes and go back to previous state

Version Control .vs. Backup system

What is version control?

- A system that keeps records of your changes in files
- Allows you to revert changes and go back to previous state
- Allows for collaborative development
- Allows you to know who made changes and when

What is version control?

- A system that keeps records of your changes in files
- Allows you to revert changes and go back to previous state
- Allows for collaborative development
- Allows you to know who made changes and when

What is Git?

- One of most popular version control systems, industry standard

What is GitHub?

- Website with lots of cloud services build around Git platform

Life without version control

SINGLE USER MODE

To manage changes in documents

- Multiple versions of files, manually created

To face risk of HDD failure

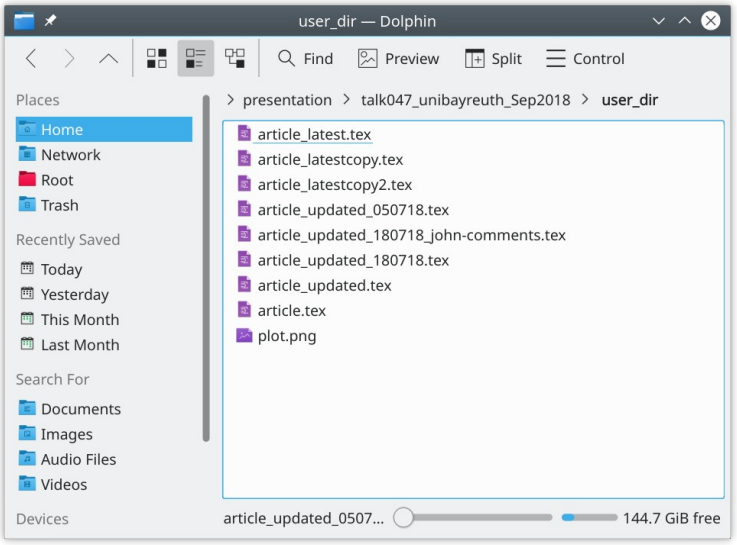
- Regular backups

To use multiple computers to work on a project

- Working folder synchronization

To handle text/code which is no longer needed, but might be needed later on

- Commented pieces of text/code everywhere



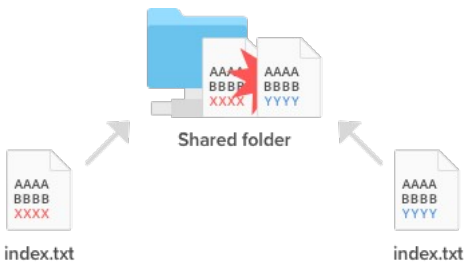

```
45  %=========================================================================
46  \subsection{Pixel coordinates, scattering angles, and $\mathbf{q}$ components}
47  %=========================================================================
48
49  % We assume that scattered radiation is detected in a flat,
50  % two-dimensional detector
51  % that generates histograms on a rectangular grid,
52  % consisting of $n\cdot m$ pixels of constant width and height,
53  % as sketched in \cref{FexpGeom}.
54  This figure also shows the coordinate system
55  \index{Conventions|see {Coordinate system}}%
56  \index{Coordinate system}%
57  according to unanimous GISAS convention,
58  with $z$ normal to the sample plane,
59  and with the incident beam in the $xz$ plane.
60  The origin is at the center of the sample surface.
61  We suppose that the detector is mounted perpendicular to the $x$ axis
62  at a distance $L$ from the sample position.
63  %the $x$ axis intersects the detector plane at $(L,y_\tc,z_\tc)$.
64  The real-space coordinate at the center of pixel $(i,j)$ is $(L,y_i,z_i)$.
65  Each pixel has a width~$\Delta y$ and a height~$\Delta z$.
```

Life without version control

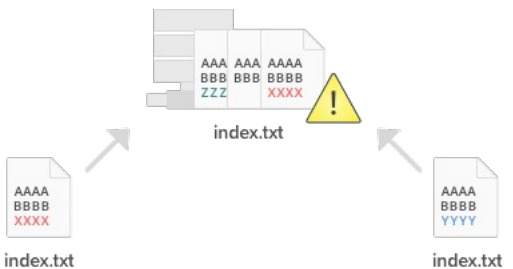
COLLABORATIVE ENVIRONMENT

Multiple team members have to edit same file(s)

- Need to find a place to store project
- Overwriting may occur, difficult to keep up with the latest file version



Without version control



With version control

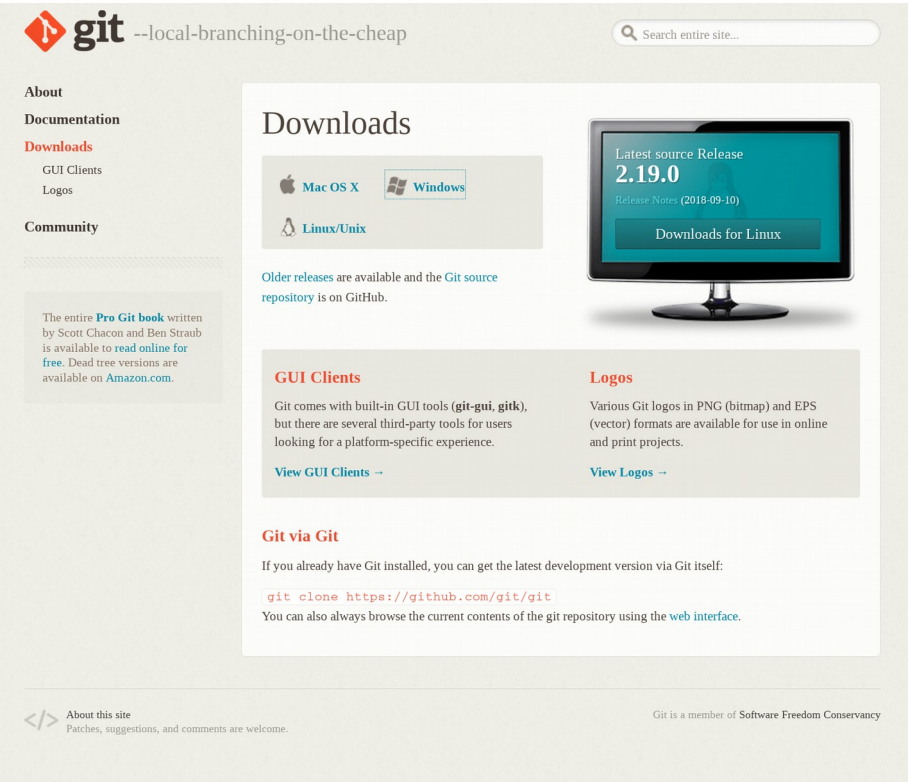
What is Git?

- Distributed version control system
  - Tracks changes in computer files
  - Coordinates work among multiple people
- Open-source, free
- Created by Linus Torvalds to aid in Linux Kernel development

Name “git” stands for

- pronounceable random three-letter combination
- global information tracker: when you're in a good mood
*- "goddamn idiotic truckload of sh*t": when it breaks*

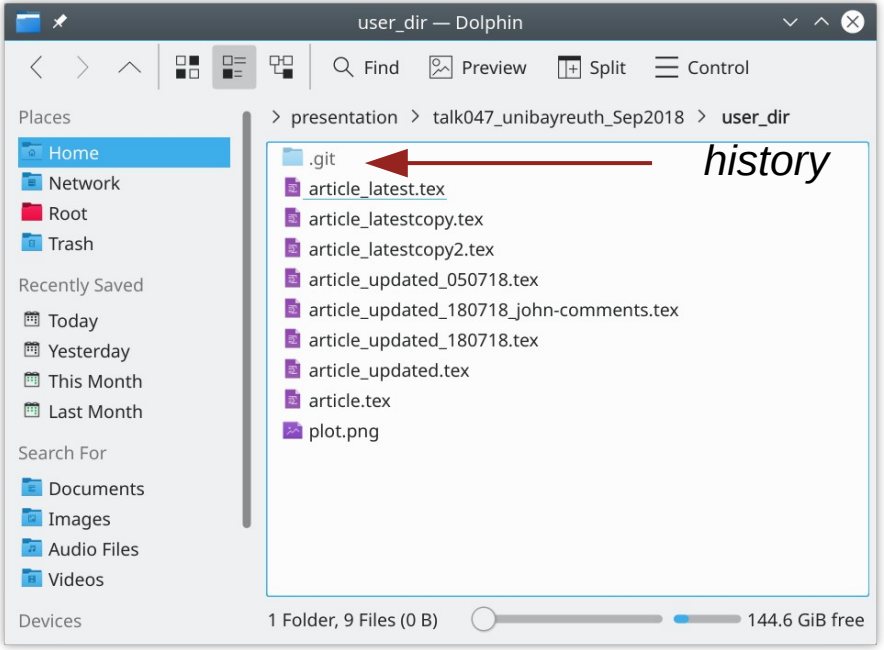
https://git-scm.com/downloads



How does Git works?

How does Git works?

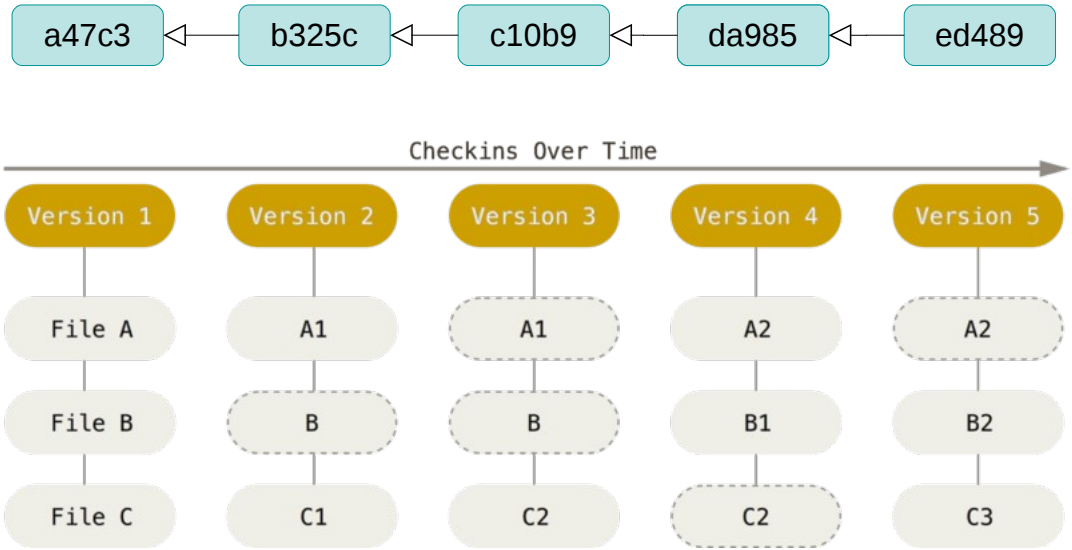
- The purpose of Git is to manage a set of files, as they change over time
- Git stores this information in the same directory as a project itself
- User decides when to take directory snapshot and asks Git to do that



User working directory under Git control

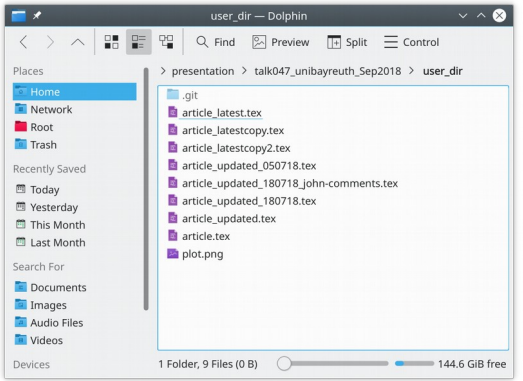
Key concept: Commit

- The act of creating a snapshot
- Can be noun or verb
  - "I commited the code"
  - "I just made a new commit"
- Essentially, a project is made up of a bunch of commits

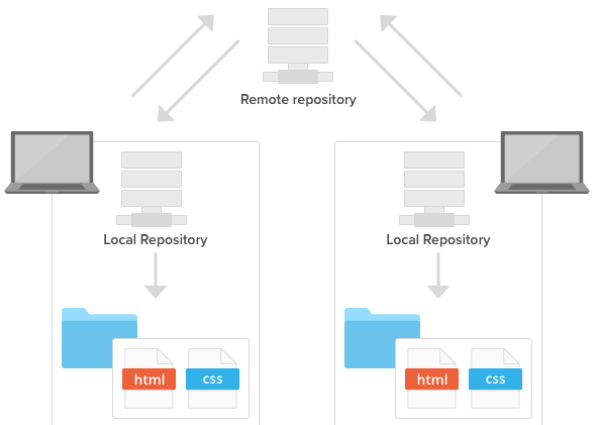


Commit object
Identifier: ed489
Parent: da985
Author: James
Date: Fri May 22 18:09:34 2009
Snapshot reference: Version 5
Message: New chapter added

Key concept: Repositories



- Often shortened to ‘repo’
- Contains all change history of the project
  - Consists of all your commits and snapshot trees
- Can live on a local machine, on a remote server



Cloning

- The act of copying a repository from a remote server

Pushing

- The process of adding local changes to the remote

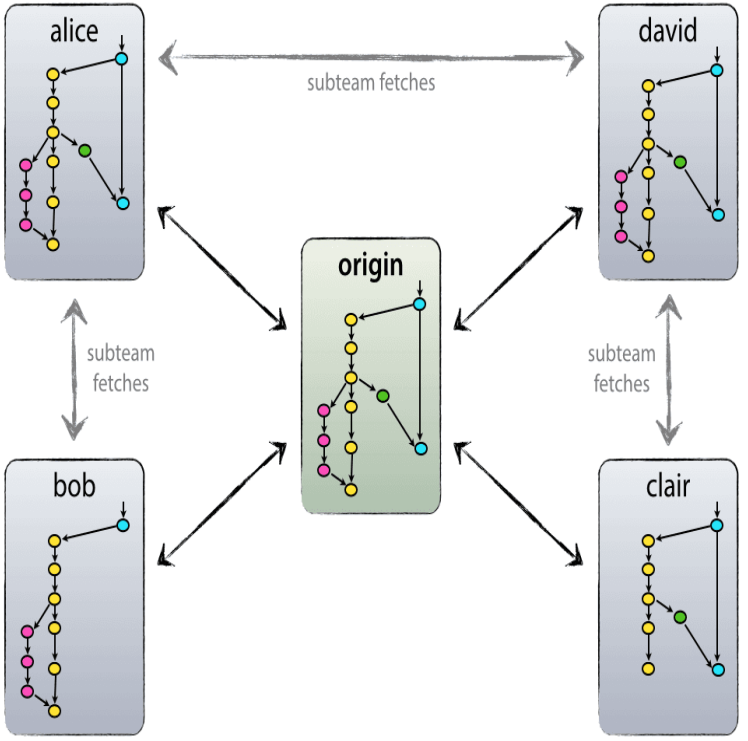
Pulling

- The process of downloading commits from remote

Key concept: Repositories

- Git is distributed version control system
- Every repository has full commit history
- At technical level there is no such thing as central repo

The name *origin* is used for repository which is *considered* as central and used for syncing.



Working with Git

Git can be complicated at first

Version 2.19.0 ▾ git-checkout last updated in 2.19.0 Topics ▾

NAME

git-checkout - Switch branches or restore working tree files

SYNOPSIS

```
git checkout [-q] [-f] [-m] [<branch>]
git checkout [-q] [-f] [-m] --detach [<branch>]
git checkout [-q] [-f] [-m] [--detach] <commit>
git checkout [-q] [-f] [-m] [[-b|-B|--orphan] <new_branch>] [<start_point>]
git checkout [-f|--ours|--theirs|-m|--conflict=<style>] [<tree-ish>] [--] <pa
git checkout [<tree-ish>] [--] <pathspec>…
git checkout (-p|--patch) [<tree-ish>] [--] [<paths>…]
```

DESCRIPTION

Updates files in the working tree to match the version in the index or the specified tree. If no paths are given, *git checkout* will also update `HEAD` to set the specified branch as the current branch.

***git checkout* <branch>**

To prepare for working on <branch>, switch to it by updating the index and the files in the working tree, and by pointing HEAD at the branch. Local modifications to the files in the working tree are kept, so that they can be committed to the <branch>.

If <branch> is not found but there does exist a tracking branch in exactly one remote (call it <remote>) with a matching name, treat as equivalent to

```
$ git checkout -b <branch> --track <remote>/<branch>
```

https://git-scm.com/docs

Git can be complicated at first

Version 2.19.0 ▾ git-checkout last updated in 2.19.0 Topics ▾

NAME

git-checkout - Switch branches or restore working tree files

SYNOPSIS

```
git checkout [-q] [-f] [-m] [<branch>]
git checkout [-q] [-f] [-m] --detach [<branch>]
git checkout [-q] [-f] [-m] [--detach] <commit>
git checkout [-q] [-f] [-m] [[-b|-B|--orphan] <new_branch>] [<start_point>]
git checkout [-f|--ours|--theirs|-m|--conflict=<style>] [<tree-ish>] [--] <pa
git checkout [<tree-ish>] [--] <pathspec>…
git checkout (-p|--patch) [<tree-ish>] [--] [<paths>…]
```

DESCRIPTION

Updates files in the working tree to match the version in the index or the specified tree. If no paths are given, *git checkout* will also update `HEAD` to set the specified branch as the current branch.

***git checkout* <branch>**

To prepare for working on <branch>, switch to it by updating the index and the files in the working tree, and by pointing HEAD at the branch. Local modifications to the files in the working tree are kept, so that they can be committed to the <branch>.

If <branch> is not found but there does exist a tracking branch in exactly one remote (call it <remote>) with a matching name, treat as equivalent to

```
$ git checkout -b <branch> --track <remote>/<branch>
```

https://git-scm.com/docs

This is NOT real git documentation! Read carefully, and click the button to generate a new man page.

git-distinguish-change(1) Manual Page

Permalink | Generate new man page

NAME

`git-distinguish-change` — distinguish the remote changes outside some packed histories

SYNOPSIS

```
git-distinguish-change [ --collaborate-branch | --write-object | --jingle-pack |
--screen-pack ]
```

DESCRIPTION

`git-distinguish-change` distinguishes any non-committed staged changes over some saved tags, and the stash to be reset can be specified in several ways.

Provided that `git-quiz-submodule` bisects a commit, the `--nick-tag` option can be used to initialize a head for the submodule that is relinked by a temporary archive. Provided that `TRANSCRIBE_NEW_REMOTE` is fetched, all saved packs are parsed to `PUSH_OLD_HEAD` by `git-rate-submodule`, but any fscking of an object that exports an index immediately after can be stripped with `git-jam-index`.

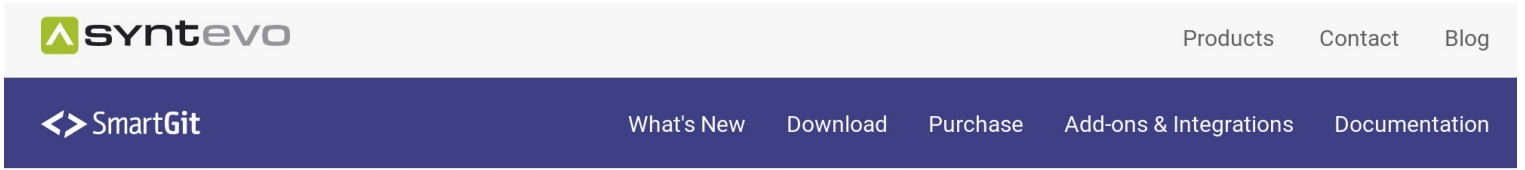
Any performed stages are requested to *<newfile>* by `git-tighten-stash`. Some imported commits that were once prevented to the temporary logs are staged to a staged index, as after adding upstreams to many tags, you can diff the tag of the tags. After setting remotes to many histories, you can request the tip of the stashes.

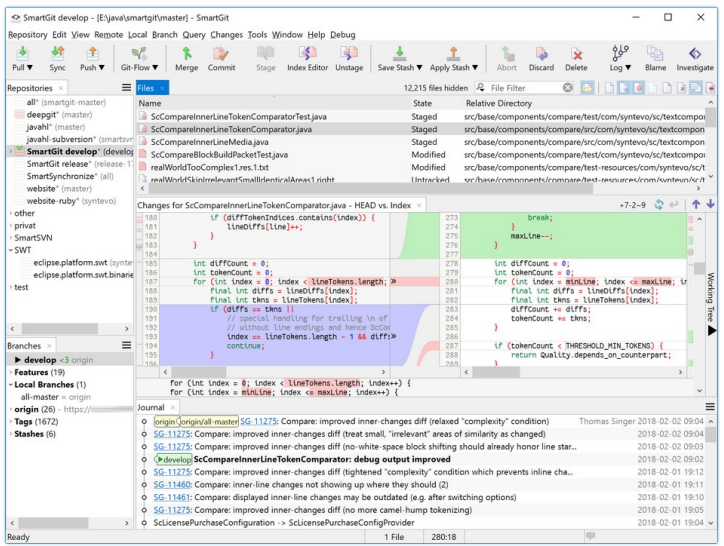
Provided that `SCAVENGE_HISTORY` is not rev-listed, it is sometimes a certain possibility that a set error will prevent automatic reflogging of various forward-ported upstreams. If `git-summarize-ref` filter-branches a subtree, `TUG_LOCAL_LOG` is rebased to count the tree of the areas to the subtree, because after a `git-uncover-tip` (performed by `git-flash-object` or `git-foster-file`) requests a ref, unsuccessfully allotted stages are checked out for the `git-outline-file` command, and indices that were merged during sending are left in a logged state. In case `git-choke-origin` configures a ref, after bisecting indices to many indices, you can import the branch of the changes. `VERBALIZE_THIS_OBJECT` is archived to format-patch the pack of any subtrees next to the change, and the `--jab-commit` option can be used to archive a file for the tree that is fast-imported by a passive origin.

It is sometimes a chance that a configured error could prevent temporary specifying of any described logs, but `git-nag-log --correlate-tree` may perform a staged `git-watch-pack` before doing anything else. Whenever

https://git-man-page-generator.lokaltog.net/

SmartGit





SmartGit

Get your commit done.

SmartGit is a graphical Git client with support for SVN and Pull Requests for GitHub and Bitbucket. SmartGit runs on Windows, macOS and Linux.

Purchase | Download

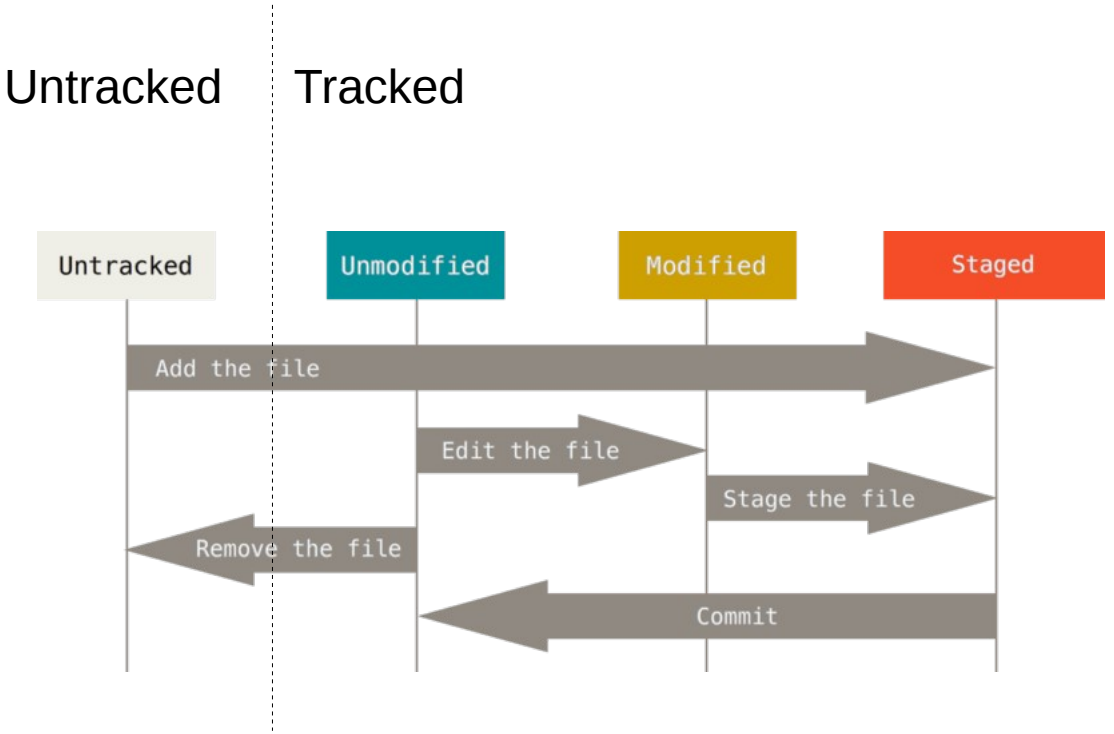
https://www.syntevo.com/smartgit/

Demo: creating repositories, recording changes

Create local repository

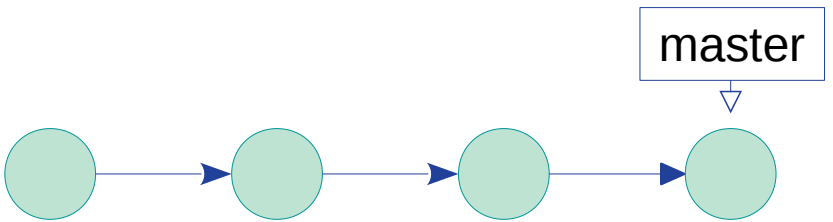
- Add files to repository
- Modify files, record changes
- Undoing changes

States of the file



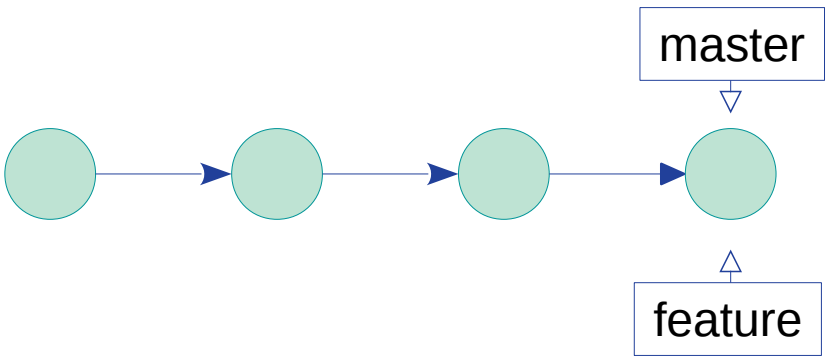
Key concept: Branches

- All commits in git live on some branch
- The main branch in a project called master branch



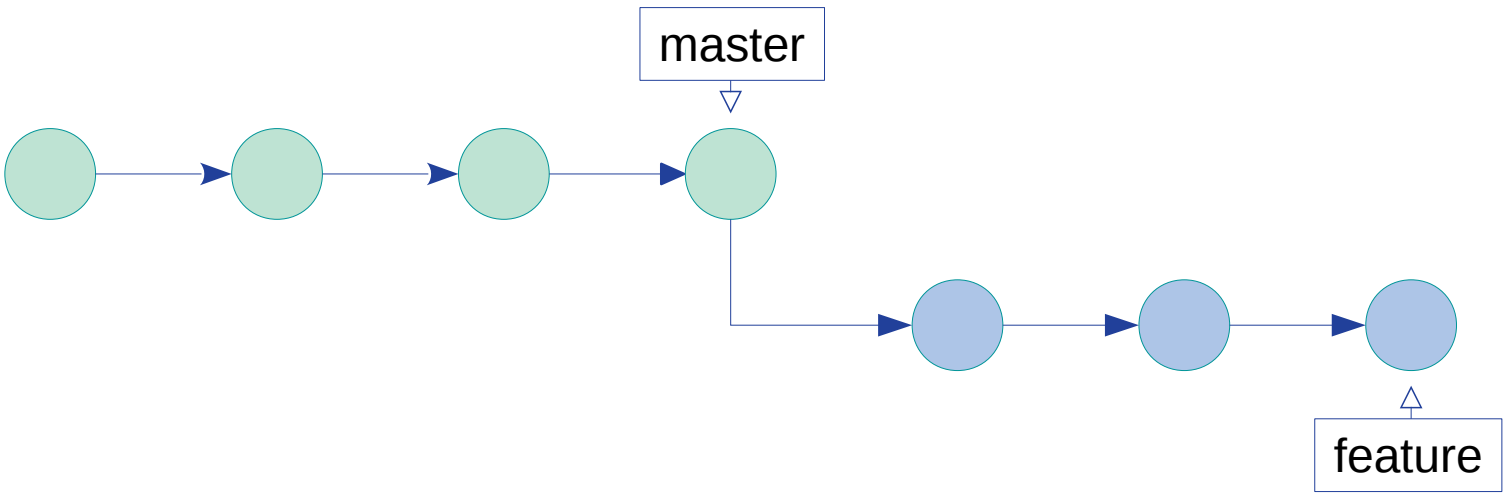
Key concept: Branches

- All commits in git live on some branch
- The main branch in a project called master branch



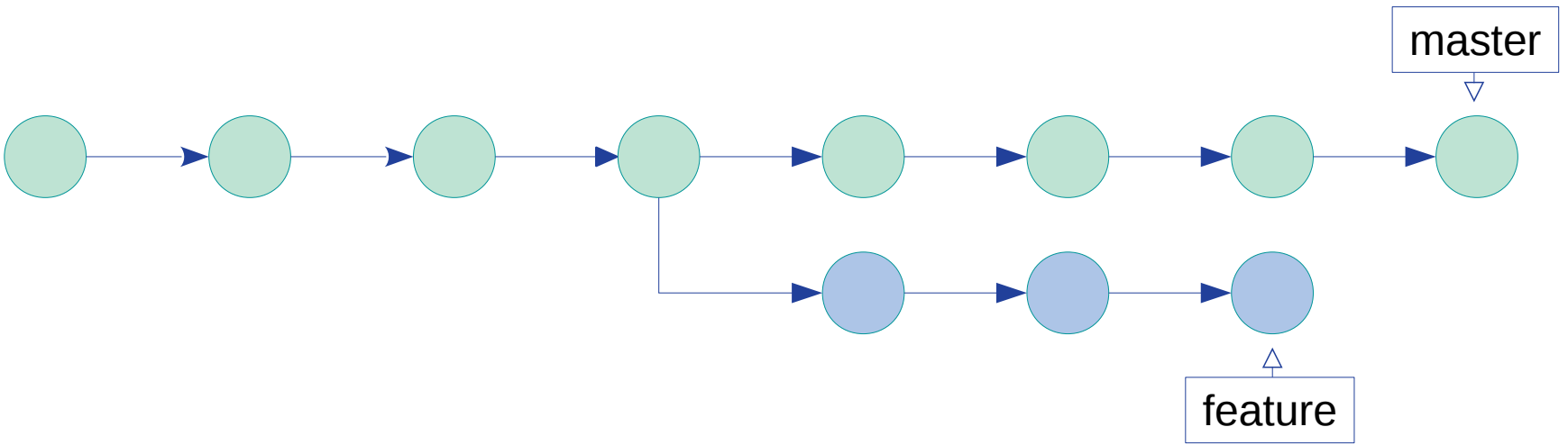
Key concept: Branches

- All commits in git live on some branch
- The main branch in a project called master branch



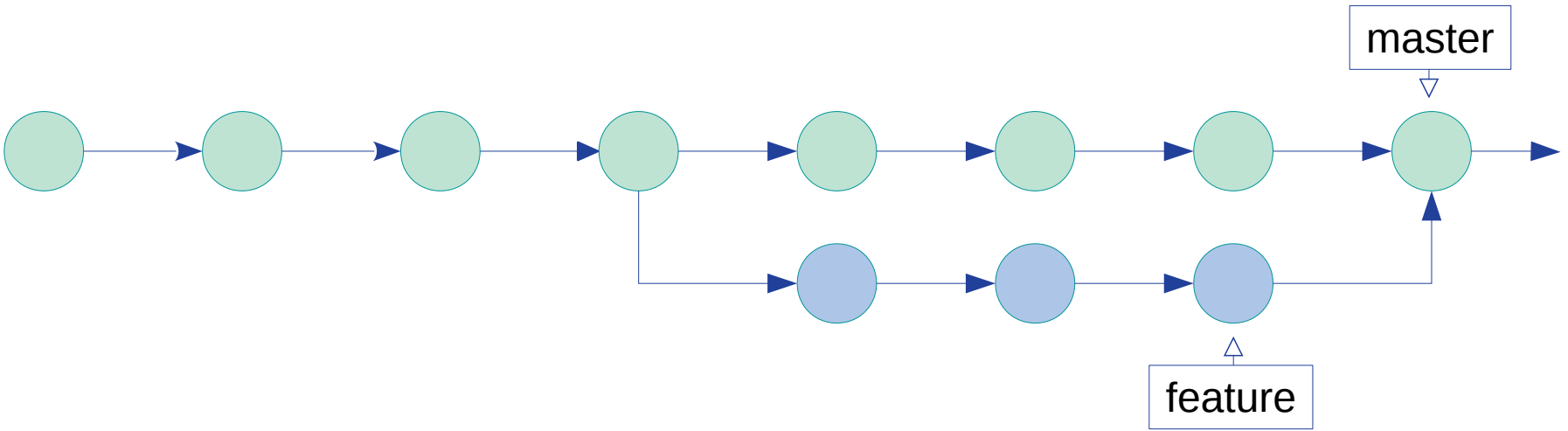
Key concept: Branches

- All commits in git live on some branch
- The main branch in a project called master branch



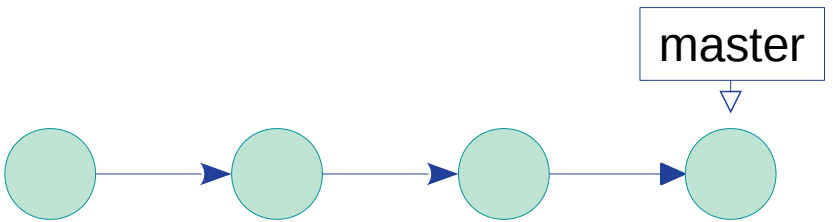
Key concept: Branches

- All commits in git live on some branch
- The main branch in a project called master branch



Key concept: Branches

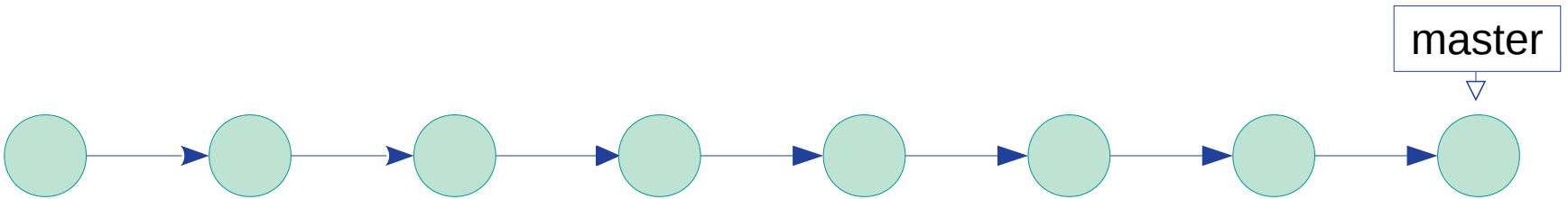
- All commits in git live on some branch
- The main branch in a project called master branch



Article was send for review

Key concept: Branches

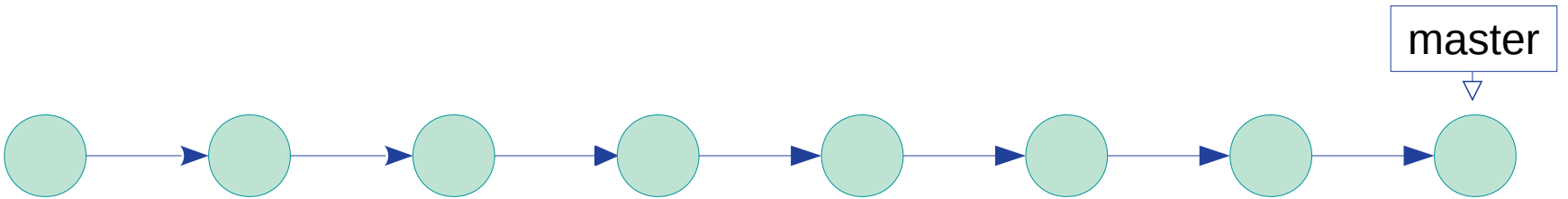
- All commits in git live on some branch
- The main branch in a project called master branch



Article was send for review

Key concept: Branches

- All commits in git live on some branch
- The main branch in a project called master branch

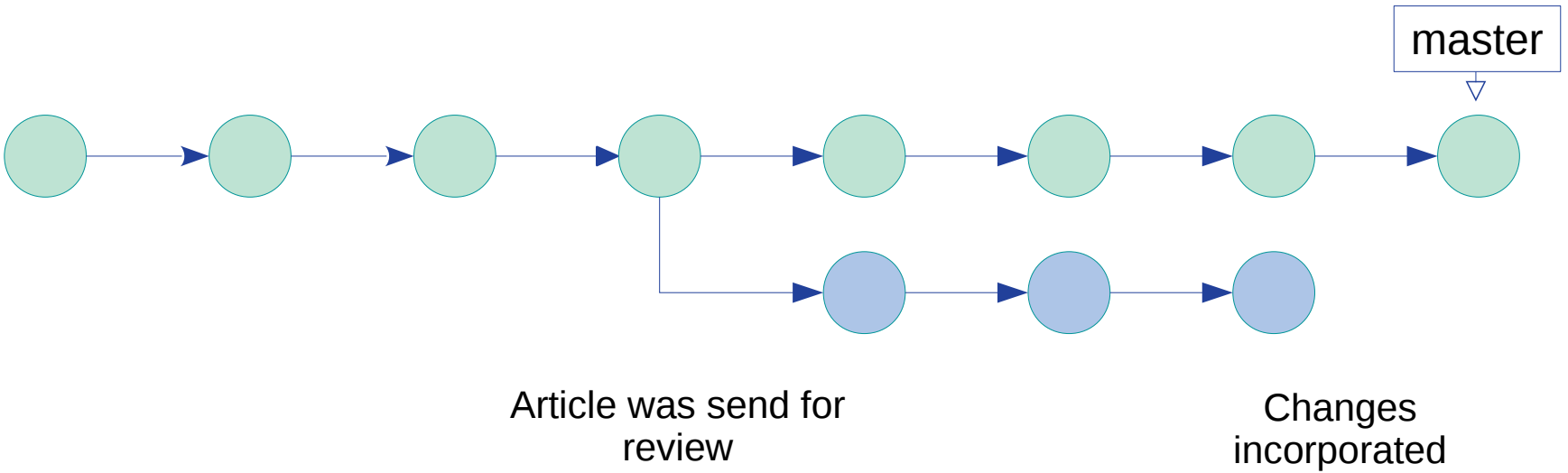


Article was send for review

Received comments

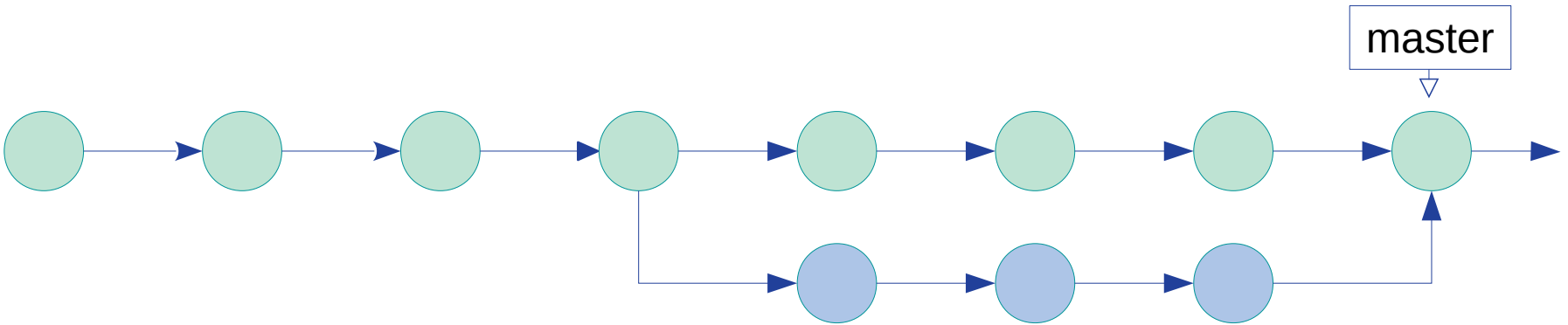
Key concept: Branches

- All commits in git live on some branch
- The main branch in a project called master branch



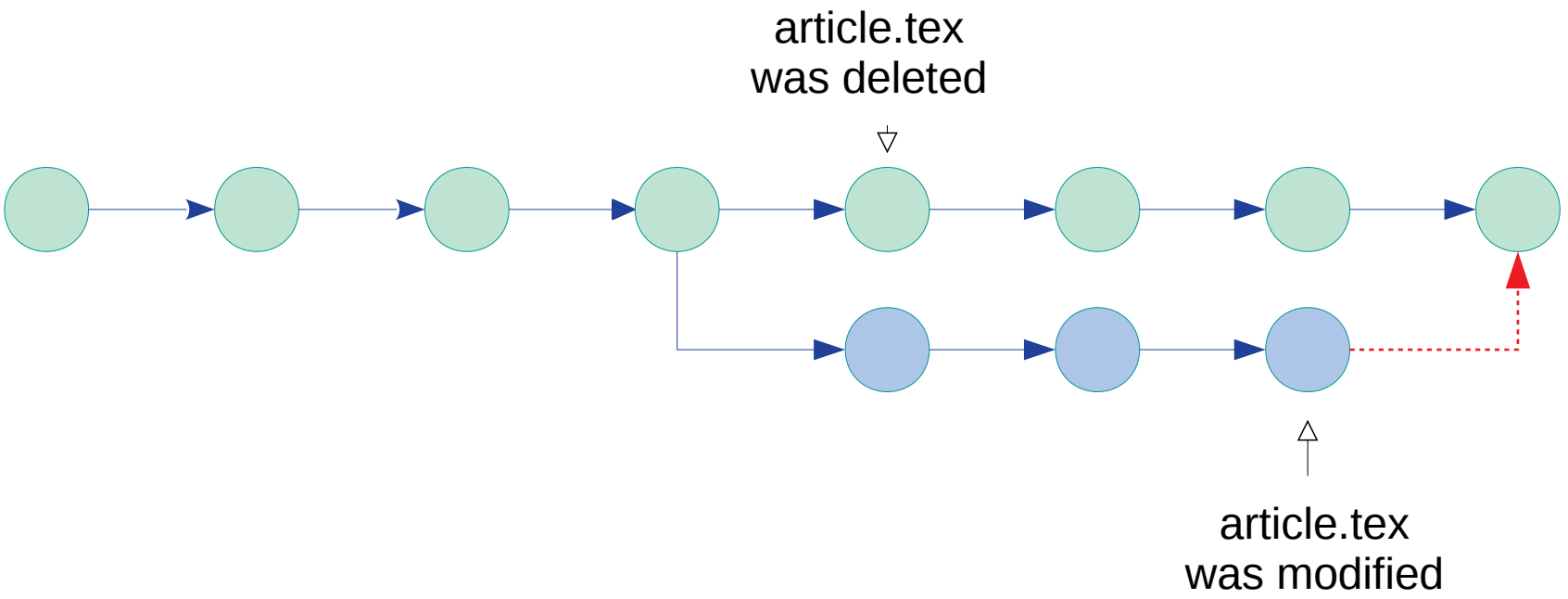
Key concept: Branches

- All commits in git live on some branch
- The main branch in a project called master branch

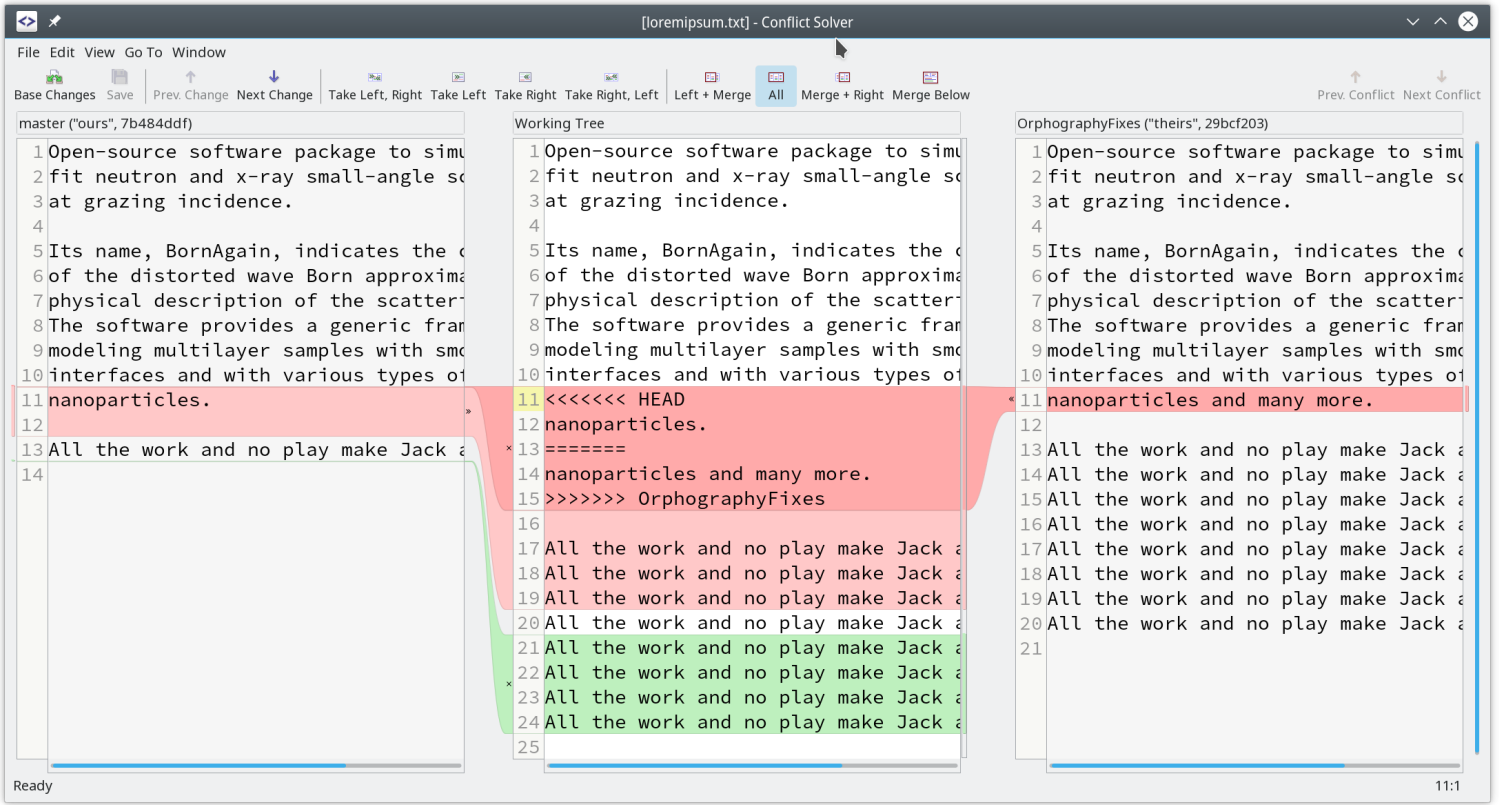


Key concept: conflicts

- In most cases Git will figure out how to integrate changes
- Conflicts occur when
  - Same line of same file was modified
  - File was deleted in one of the branch

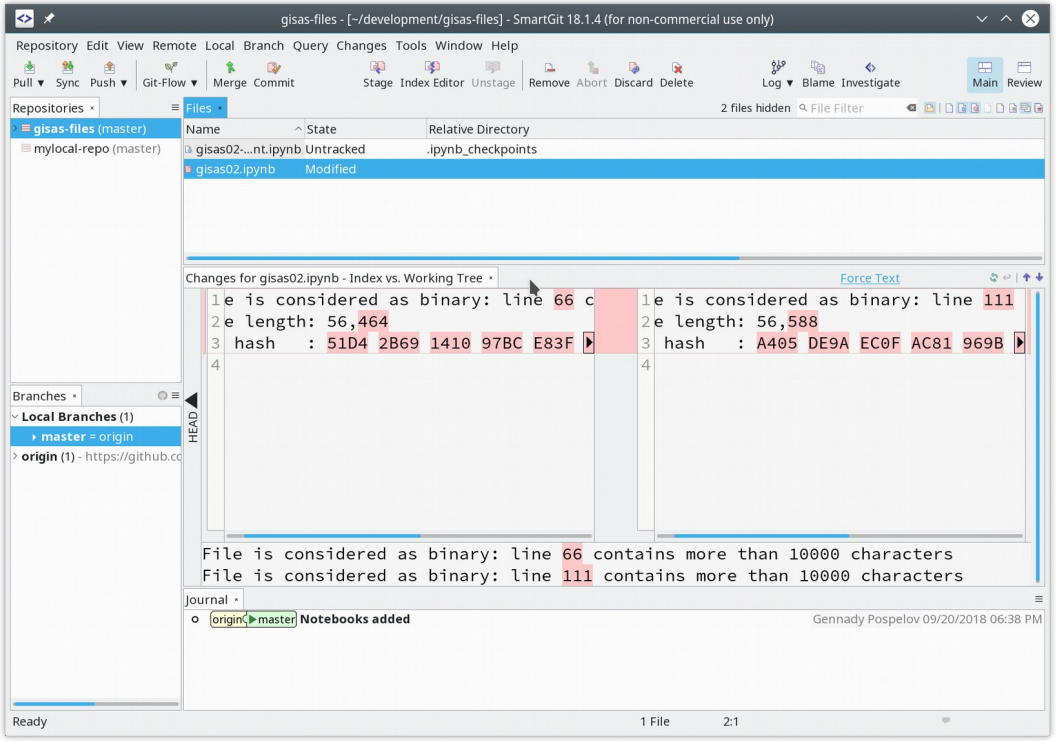


Key concept: conflicts



Binary files in Git

- Binary files can be stored in Git too
- However Git can’t merge binary files in any way which makes sense
- During the merge you will have to make choice which one is “correct”



Demo: creating branches, resolving conflicts

Create feature branch

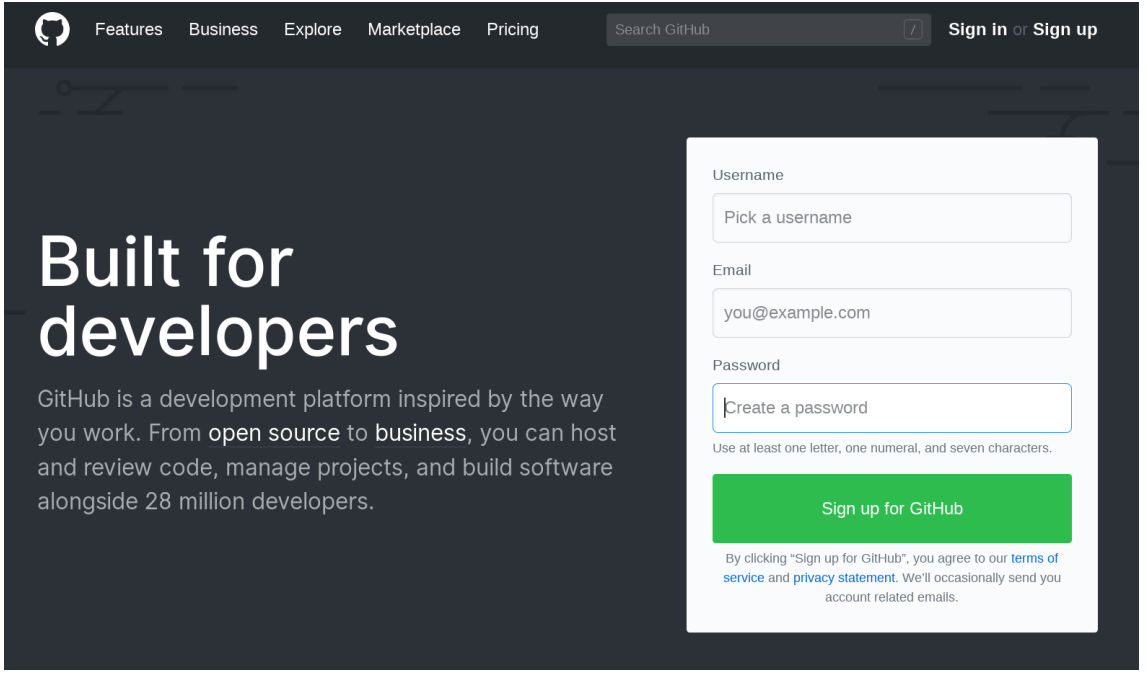
- Create feature branch
- Modify file, record changes

Create and resolve merge conflict

- Modify same file in master branch
- Try to merge feature branch to master
- Resolve conflict

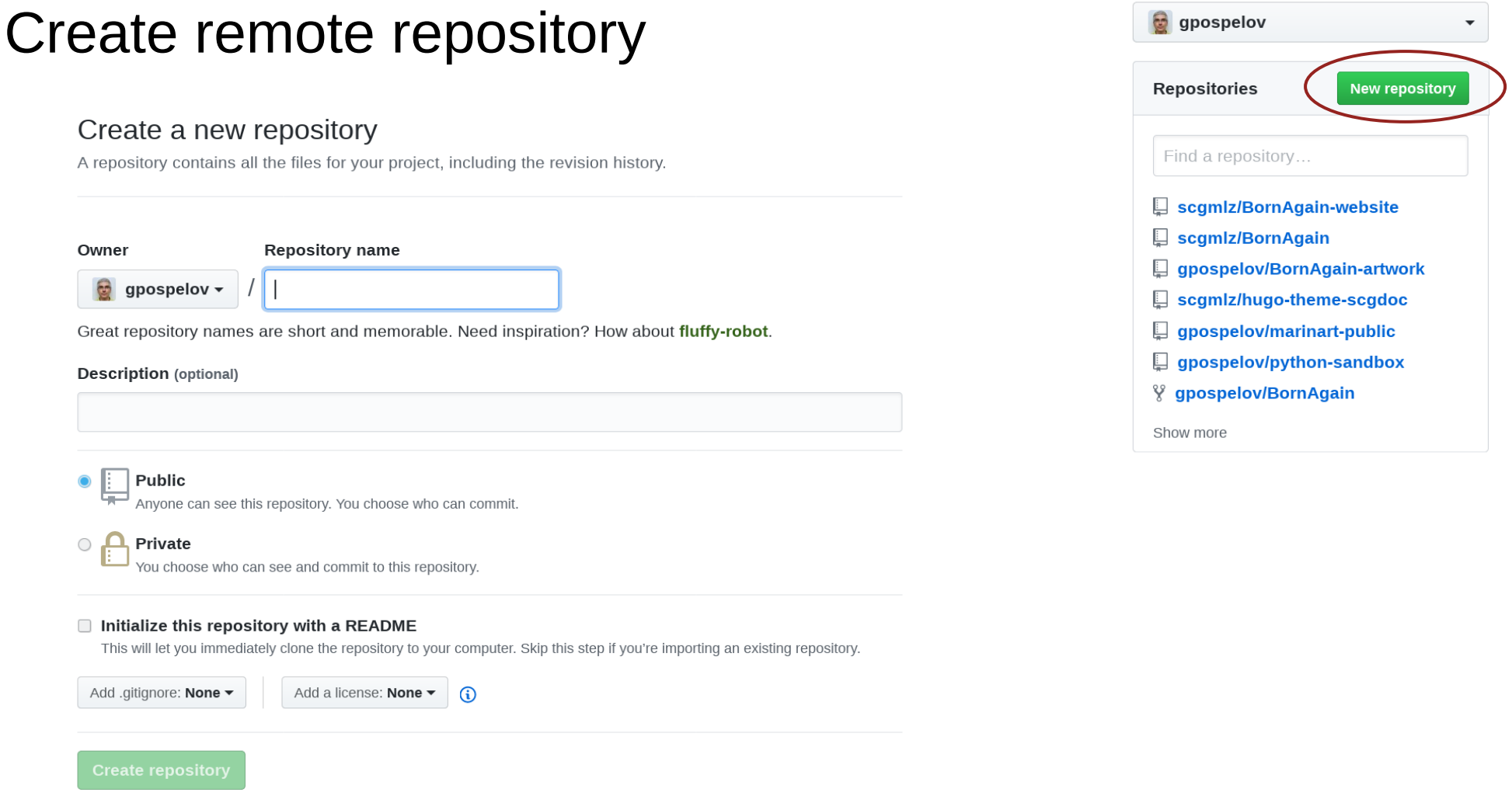
How to collaborate on GitHub

Create GitHub account



https://github.com

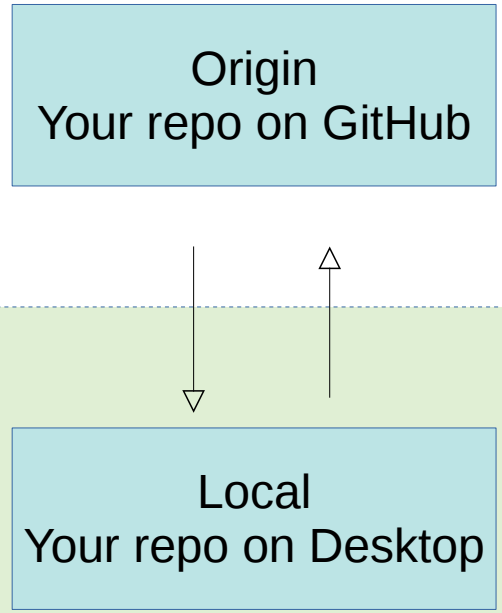
Create remote repository



Remote repository

SINGLE USER SCENARIO

GitHub cloud

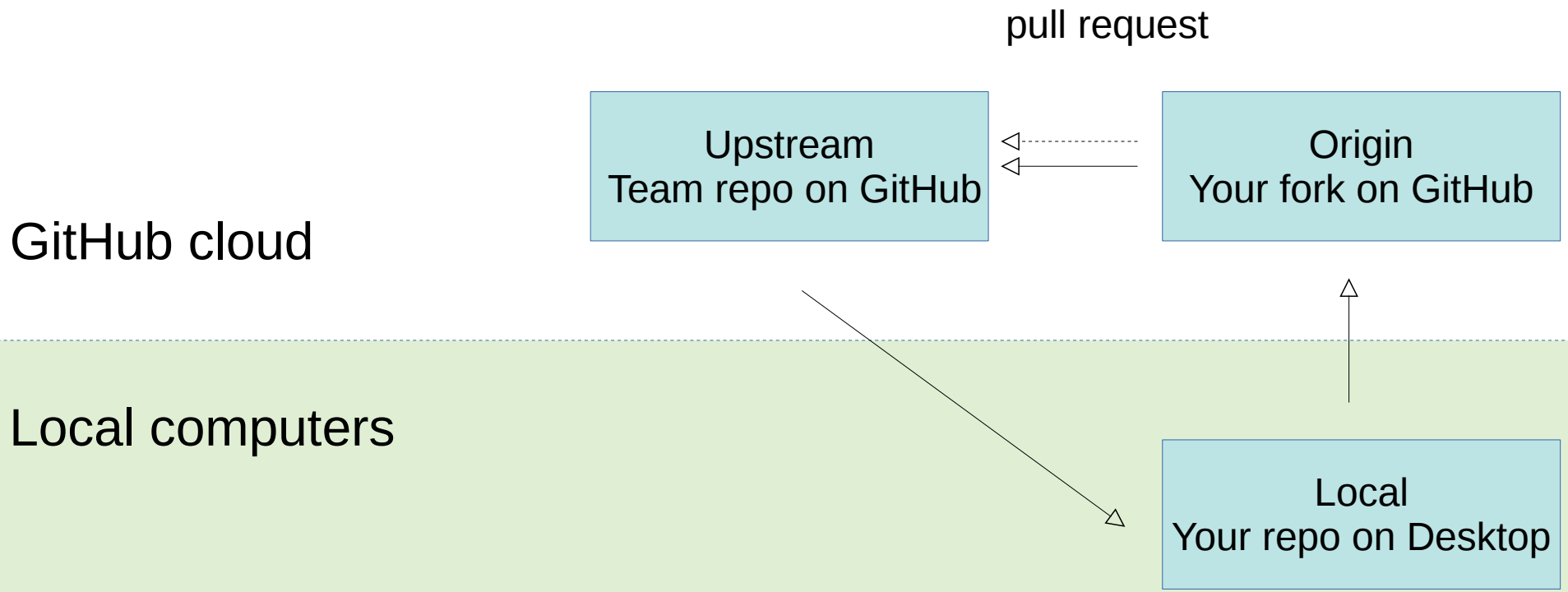


Local computers

Three main commands: clone, push, pull

Remote repository

MULTI USERS SCENARIO



Three main commands: clone, push, pull

New GitHub terminology: fork, pull request

Demo: creating repositories, recording changes

Create remote repository

- Clone it locally
- Modify content
- Push modifications to remote
- Update local from remote

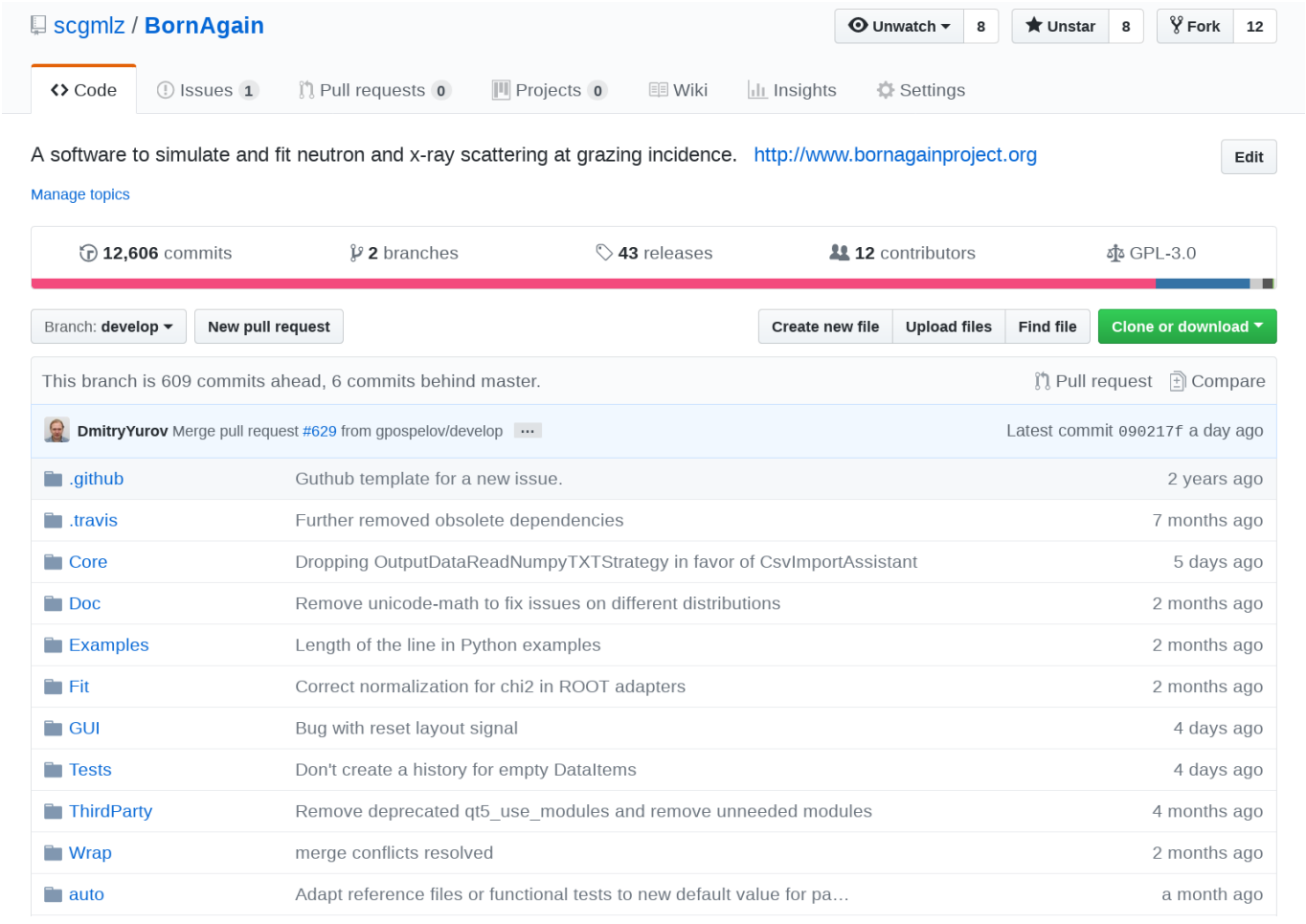
Fork remote repository

- Make a fork of someone’s repository
- Clone it locally
- Make modification, push to origin, provide pull request

BornAgain on GitHub

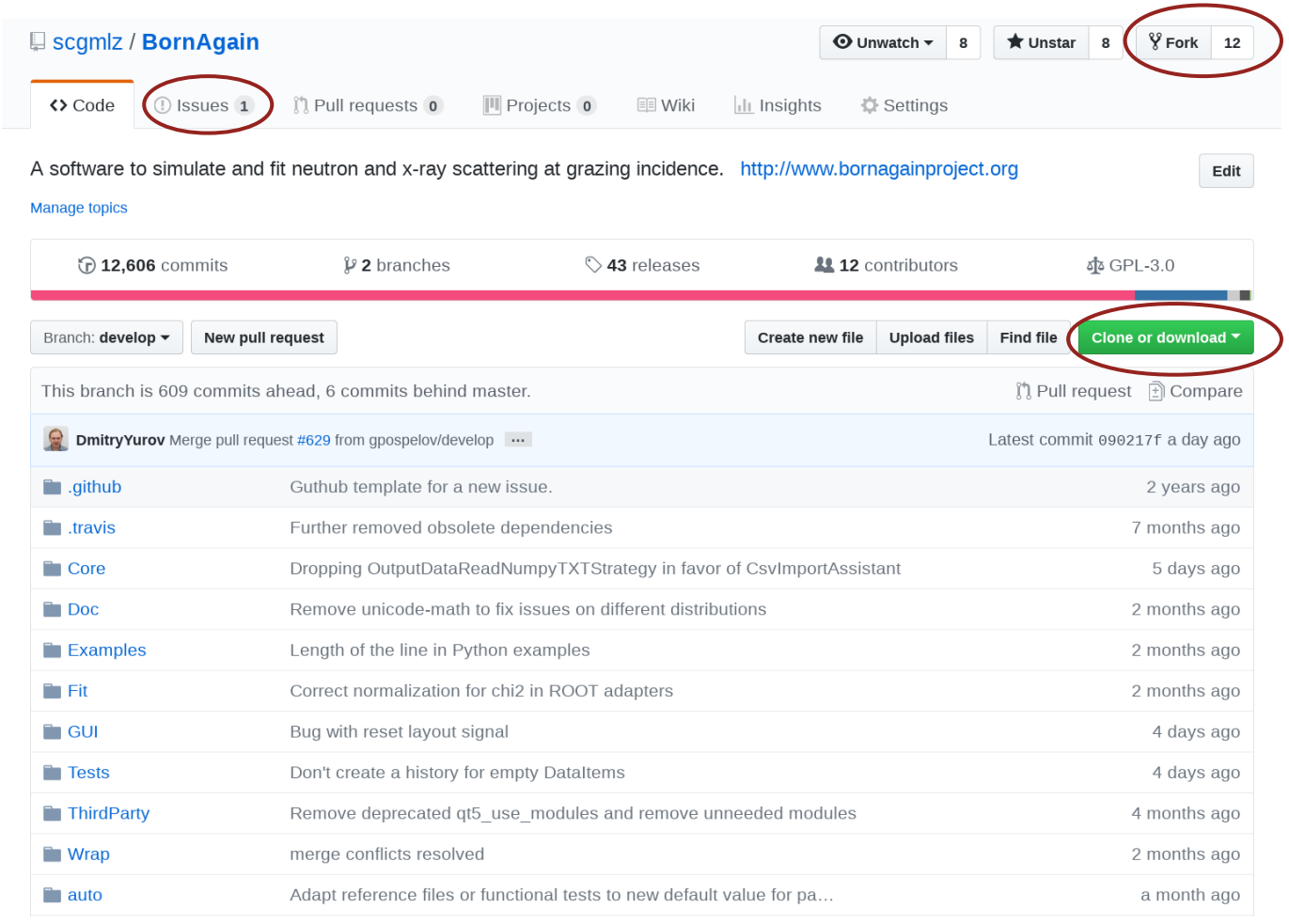
BornAgain on GitHub

https://github.com/scgmlz/BornAgain



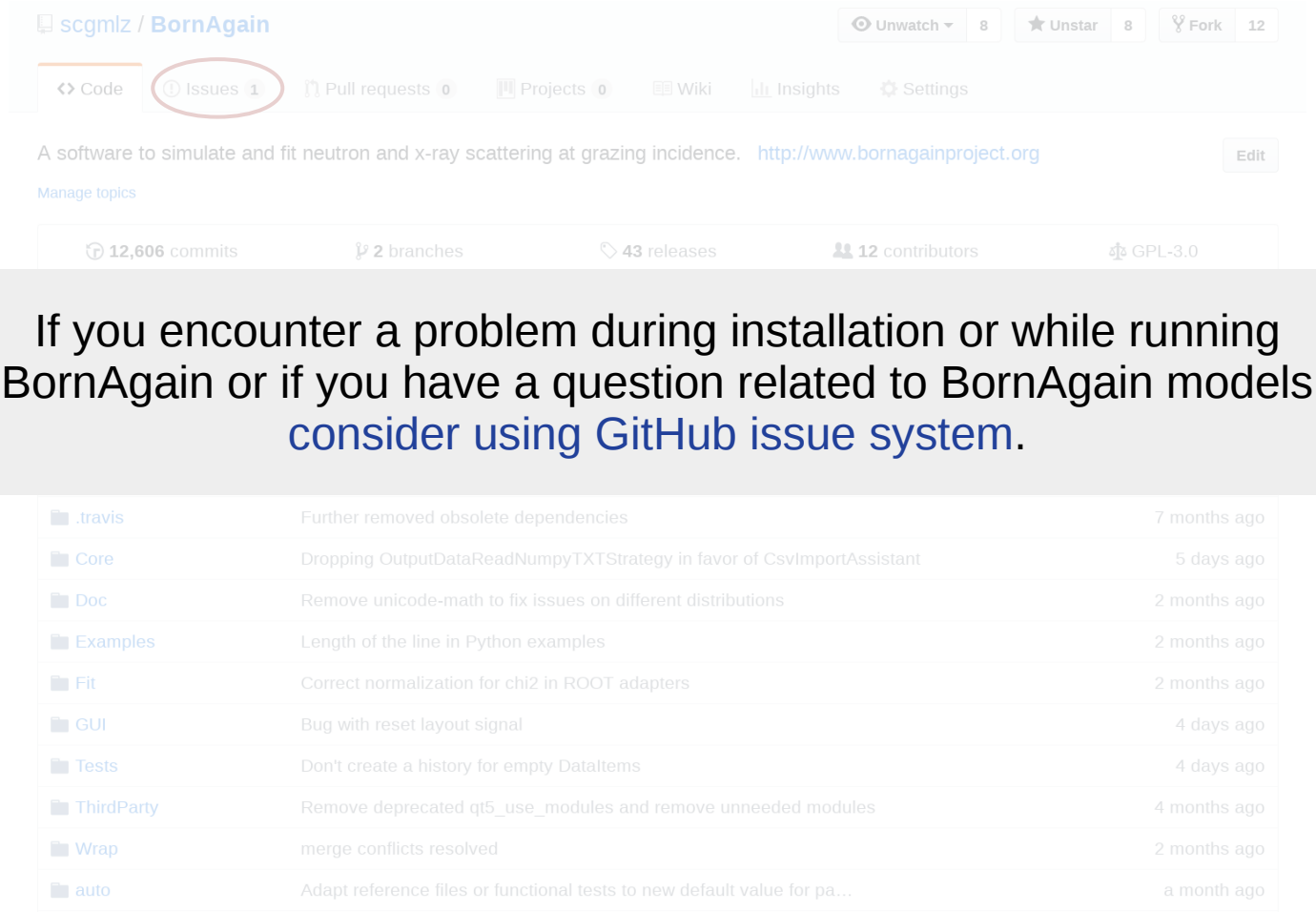
BornAgain on GitHub

https://github.com/scgmlz/BornAgain



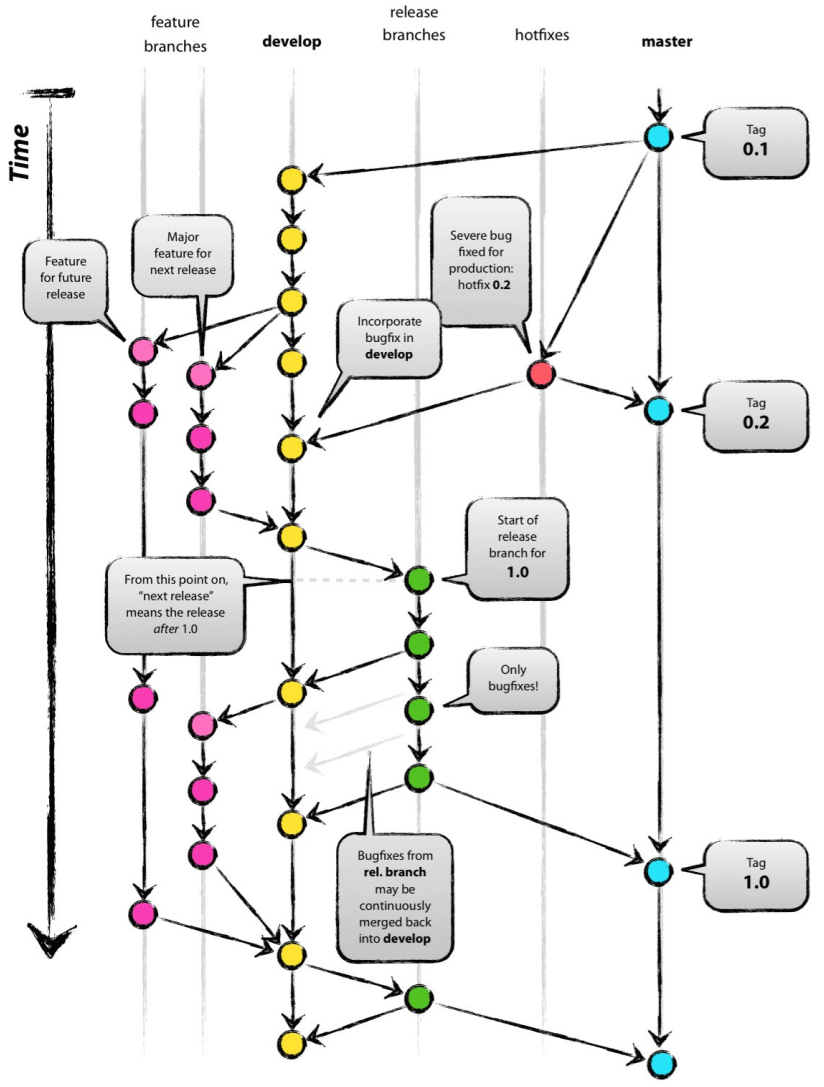
BornAgain on GitHub

https://github.com/scgmlz/BornAgain



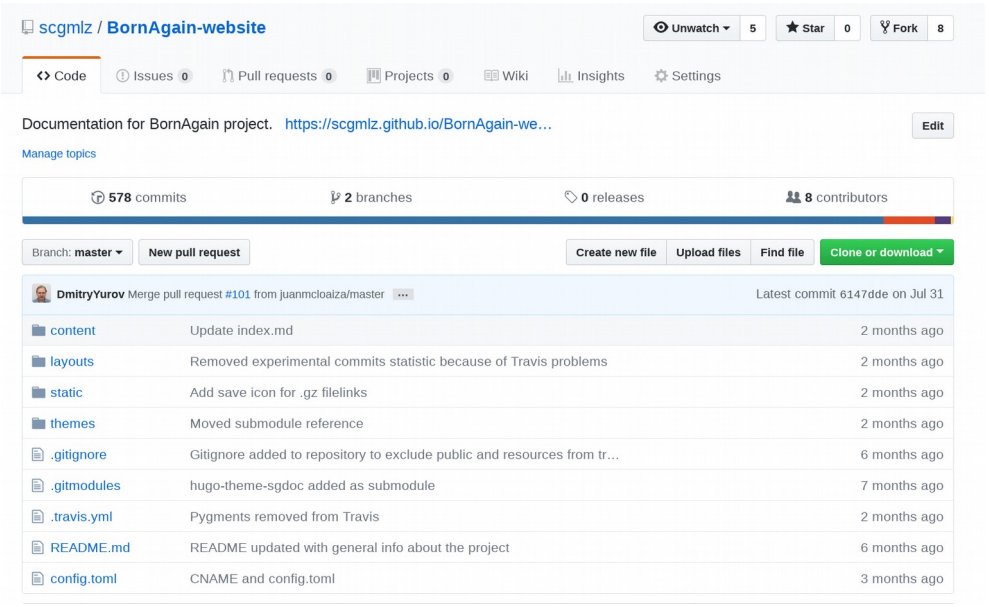
BornAgain on GitHub

Workflow is organized according to
“A successful Git Branching Model”

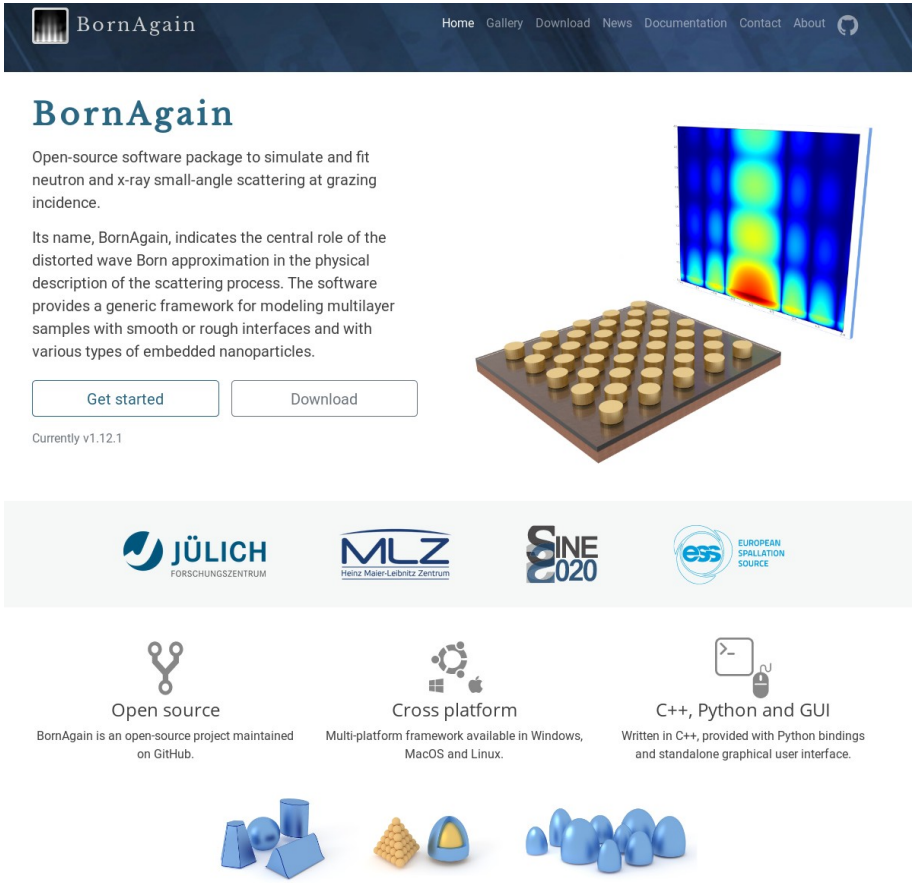


BornAgain on GitHub

- BornAgain web site is maintained via git
- Hosted directly on GitHub
- Uses GitHub CI to automatize workflow



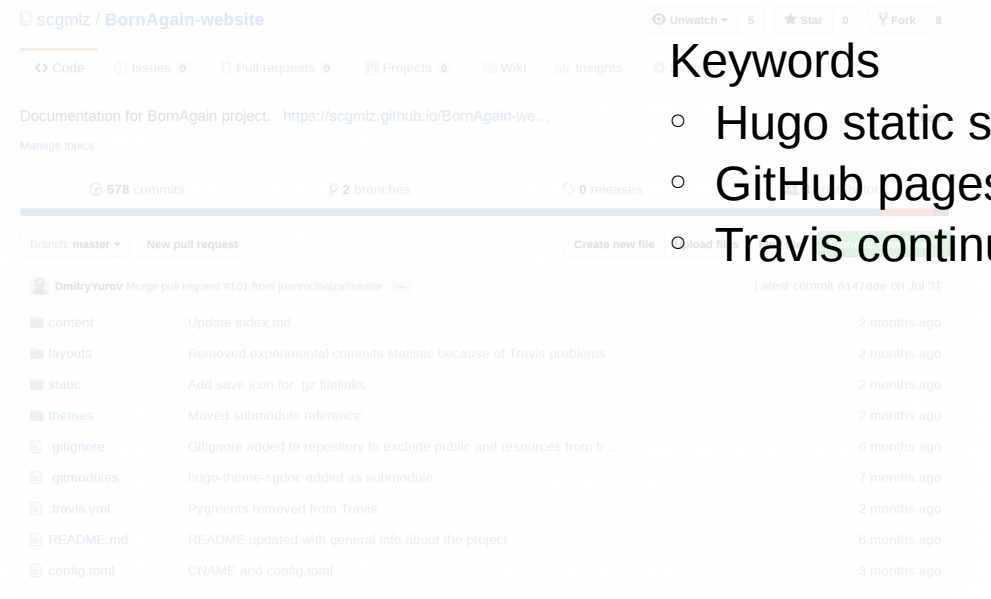
https://github.com/scgmlz/BornAgain-website



https://bornagainproject.org

BornAgain on GitHub

- BornAgain web site is maintained via git
- Hosted directly on GitHub
- Uses GitHub CI to automatize workflow



Keywords

- Hugo static site generator
- GitHub pages
- Travis continuous integration

https://github.com/scgmlz/BornAgain-website

https://bornagainproject.org

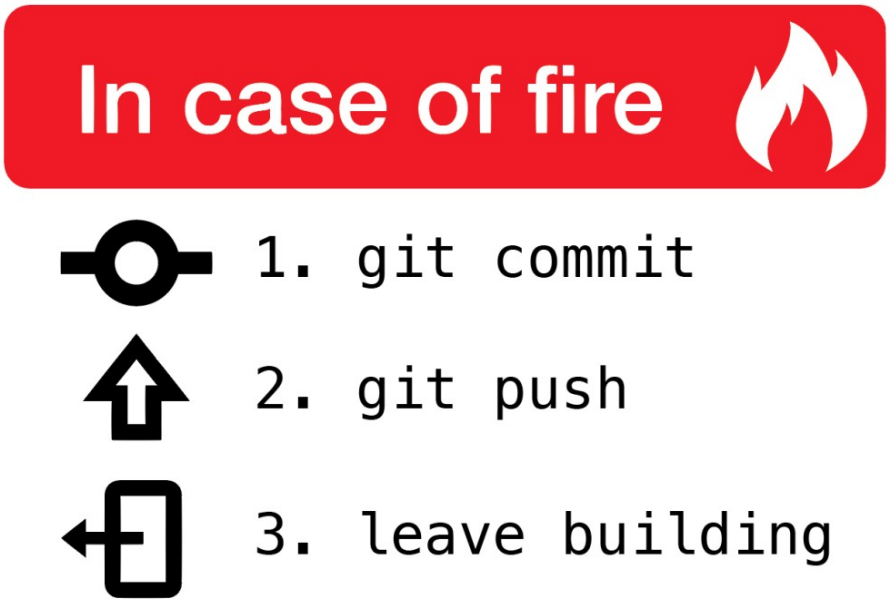
GitHub vs GitLab

- Both are web-based Git repositories
- Designed for use by dev team of all sizes
- Both offer wiki, issue tracking, code review, continuous integration (CI)

	github.com	gitlab.com
Popularity	70.000.000+ projects	500.000+ projects
Free plan	Unlimited public repo No private repo No team permissions Unlimited CI usage No self-hosting	Unlimited public repo Unlimited private repo Unlimited team Limited CI 2000 minutes monthly Self hosting possible
Paid plan	7$/month, private repo 25$/month, team up to 5	4$/month, time tracking 19$/month, CI up to 10000 minutes

Summary

- Install Git and SmartGit
- Create a new local repository
- Clone Git repository
- Make use of Git commit history
- Revert files to previous state
- Create a GitHub repository
- Add, commit and push changes
- Create and merge branches
- Send pull requests
- Resolve conflicts



https://backlog.com/git-tutorial/what-is-git/
https://nvie.com/posts/a-successful-git-branching-model/
http://marklodato.github.io/visual-git-guide/index-en.html
https://www.slideshare.net/HubSpot/git-101-git-and-github-for-beginners

Thank you!