

# Git & Sourcetree

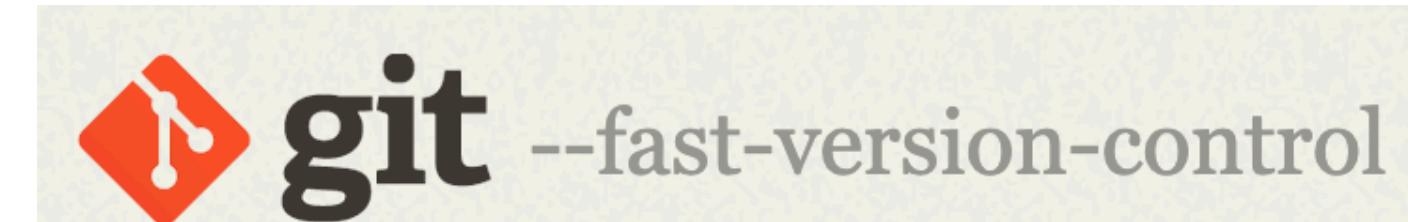
Git, Github &  
Sourcetree



A high level tour of git &  
Sourcetree

# Git Command line application

Create, update and manage local and remote repositories of code projects



Search entire site...

## About

## Documentation

## Downloads

GUI Clients

Logos

## Community

The entire [Pro Git book](#) written by Scott Chacon and Ben Straub is available to [read online for free](#). Dead tree versions are available on [Amazon.com](#).

# Downloads



Mac OS X



Windows



Linux/Unix

Older releases are available and the [Git source repository](#) is on GitHub.

## GUI Clients

Git comes with built-in GUI tools (`git-gui`, `gitk`), but there are several third-party tools for users looking for a platform-specific experience.

[View GUI Clients →](#)

## Git via Git

If you already have Git installed, you can get the latest development version via Git itself:

```
git clone https://github.com/git/git
```

You can also always browse the current contents of the git repository using the [web interface](#).



## Logos

Various Git logos in PNG (bitmap) and EPS (vector) formats are available for use in online and print projects.

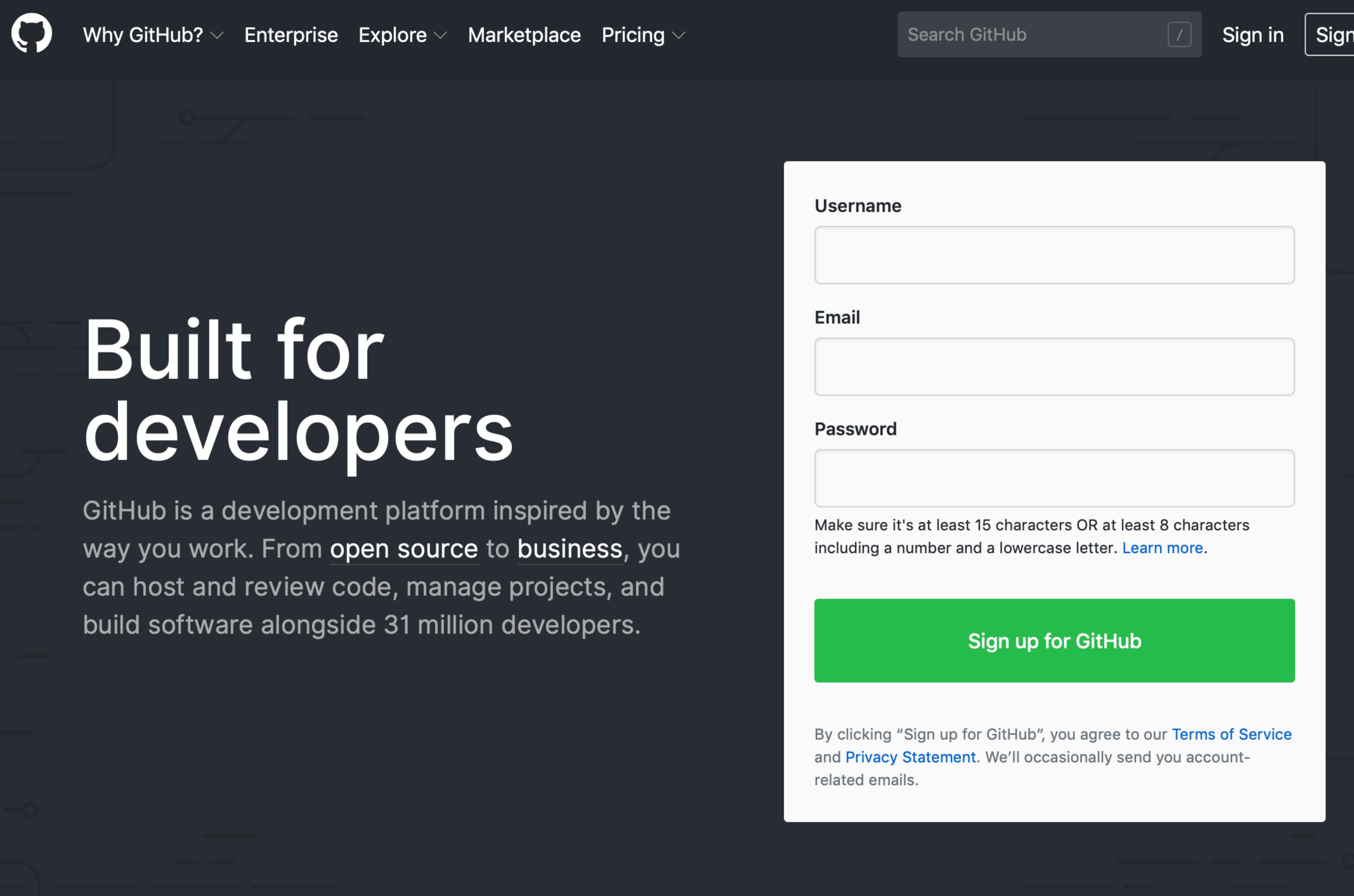
[View Logos →](#)

## Github Web Service

Host code  
projects. Support  
sharing and  
collaboration

# Built for developers

GitHub is a development platform inspired by the way you work. From open source to business, you can host and review code, manage projects, and build software alongside 31 million developers.



The image shows the GitHub sign-up interface. At the top, there's a navigation bar with links for "Why GitHub?", "Enterprise", "Explore", "Marketplace", and "Pricing". To the right of the navigation is a search bar with a placeholder "Search GitHub" and a magnifying glass icon. Further right are "Sign in" and "Sign up" buttons. Below the navigation, the main heading "Built for developers" is displayed. A descriptive paragraph follows, explaining GitHub's purpose as a development platform for both open source and business projects. To the right of this text is a sign-up form. The form consists of three input fields: "Username", "Email", and "Password". Below these fields is a note: "Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter." A "Learn more" link is also present. At the bottom of the form is a large green "Sign up for GitHub" button. Below the button, a small note states: "By clicking 'Sign up for GitHub', you agree to our [Terms of Service](#) and [Privacy Statement](#). We'll occasionally send you account-related emails."

Why GitHub? Enterprise Explore Marketplace Pricing

Search GitHub / Sign in Sign up

Username

Email

Password

Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. [Learn more](#).

Sign up for GitHub

By clicking "Sign up for GitHub", you agree to our [Terms of Service](#) and [Privacy Statement](#). We'll occasionally send you account-related emails.

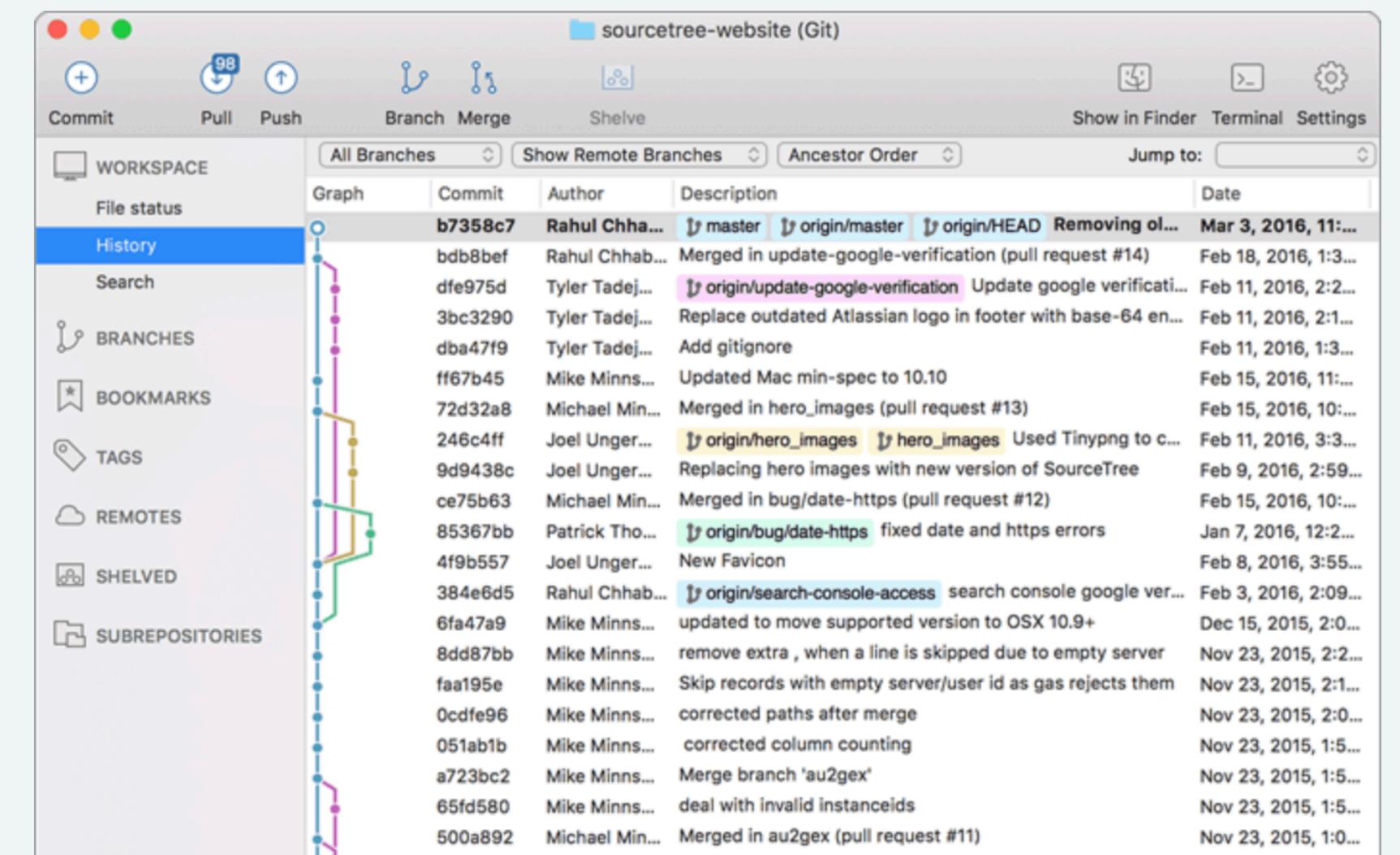
# Sourcetree

A graphical client application  
to compliment the git  
command line

**Simplicity and power  
in a beautiful Git GUI**

[Download for Mac OS X](#)

Also available for Windows



## A free Git client for Windows and Mac

Sourcetree simplifies how you interact with your Git repositories so you can focus on coding. Visualize and manage your repositories through Sourcetree's simple Git GUI.

```

docs/themes/sourcetree/templates/base.html
Hunk 1 : Lines 1-7
1 1 <!DOCTYPE html>
2 2 <html>
3 3   <head>
4 +   <title>SourceTree Help</title>
4 5     <link rel="stylesheet" href="{{ SITEURL }}/{{ THEME_STATIC_DIR }}/css/
5 6   </head>
6 7   <body>

Hunk 2 : Lines 15-24
14 15 <footer>
15 16   <a href="http://blog.sourcetreeapp.com/">Blog</a> | <a href="http://w
16 17   Copyright &copy; 2014
17 -   <a href="http://www.atlassian.com/"><img href="{{ SITEURL }}/{{ THEME
18 +   <div id="footer-right">
19 +     Copyright &copy; 2014

```

### Simple for beginners

Say goodbye to the command line - simplify distributed version control with a Git client and quickly bring everyone up to speed.

### Powerful for experts

Perfect for making advanced users even more productive. Review changesets, stash, cherry-pick between branches and more.

# git - the simple guide

just a simple guide for getting started with git. no deep s#@t ;)



by Roger Dudler

credits to @tfnico, @fhd and Namics

this guide in deutsch, español, français, indonesian, italiano, nederlands, polski, português, русский, тürkçe,

မြန်မာ, 日本語, 中文, 한국어 Vietnamese

please report issues on github

<http://rogerdudler.github.io/git-guide>

+ Sourcetree

# git command line

- Create
- Checkout
- Workflow
- Add & Commit
- Pushing Changes
- Branching
- Update & Merge
- Tagging
- Log
- Replace Local Changes



# Sourcetree

- Create
- Checkout
- Workflow
- Add & Commit
- Pushing Changes
- Tagging
- Log



# git command line

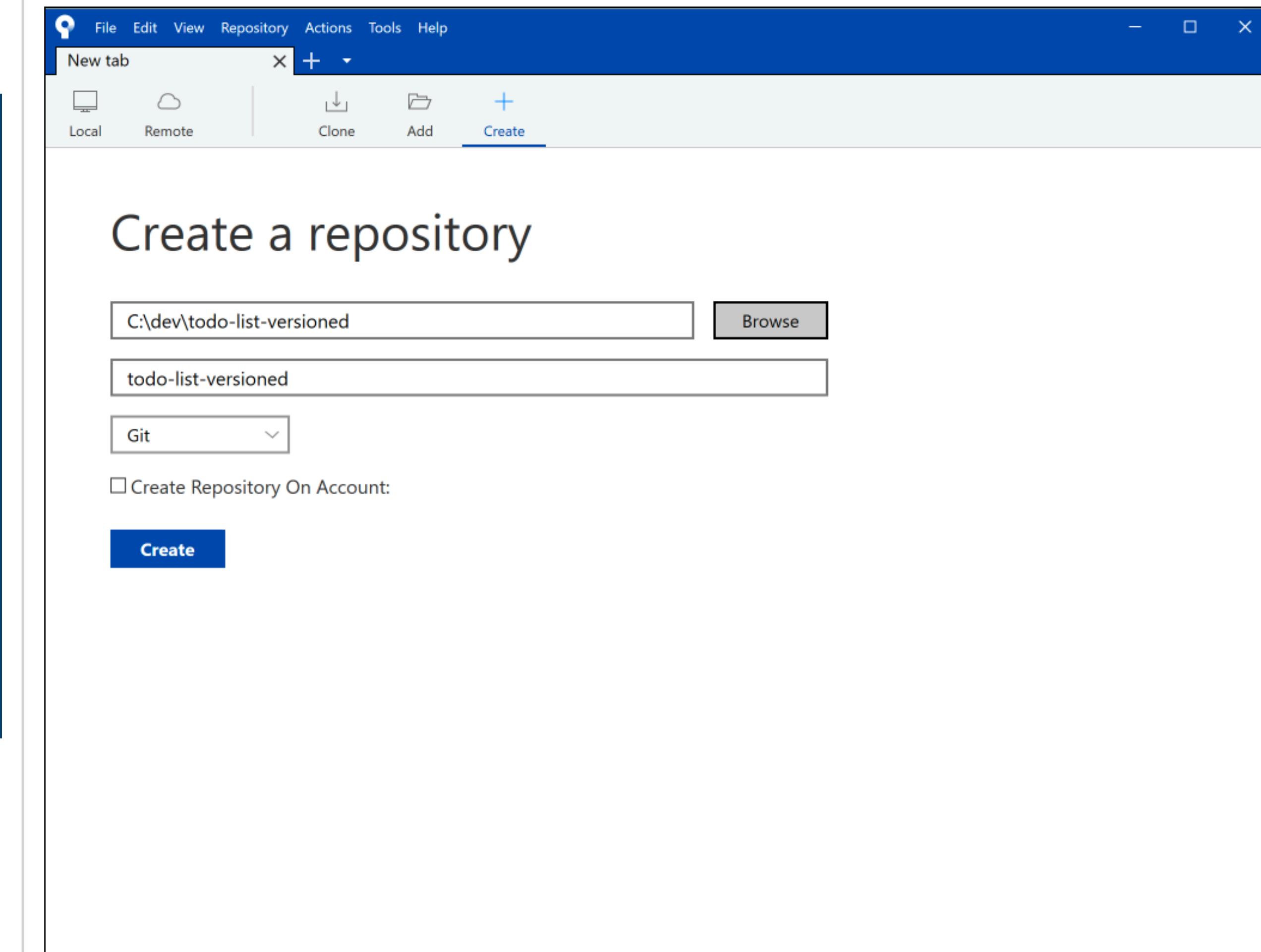
## create a new repository

create a new directory, open it and perform a

**git init**

to create a new git repository.

# Sourcetree



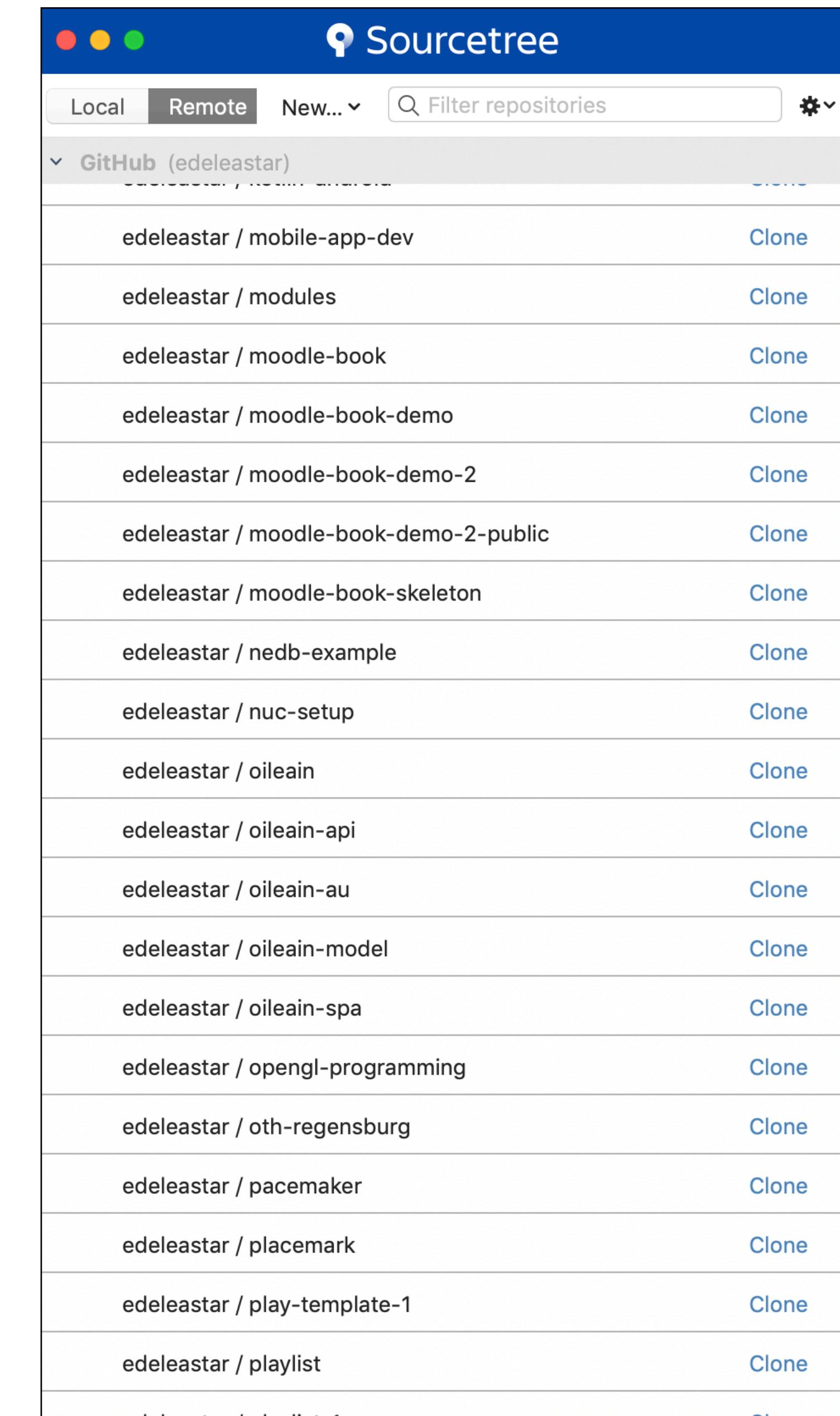
## checkout a repository

create a working copy of a local repository by running the command

```
git clone /path/to/repository
```

when using a remote server, your command will be

```
git clone username@host:/path/to/repository
```



# workflow

your local repository consists of three "trees" maintained by git. the first one is your **Working Directory** which holds the actual files. the second one is the **Index** which acts as a staging area and finally the **HEAD** which points to the last commit you've made.



# git command line

## add & commit

You can propose changes (add it to the **Index**) using

```
git add <filename>
```

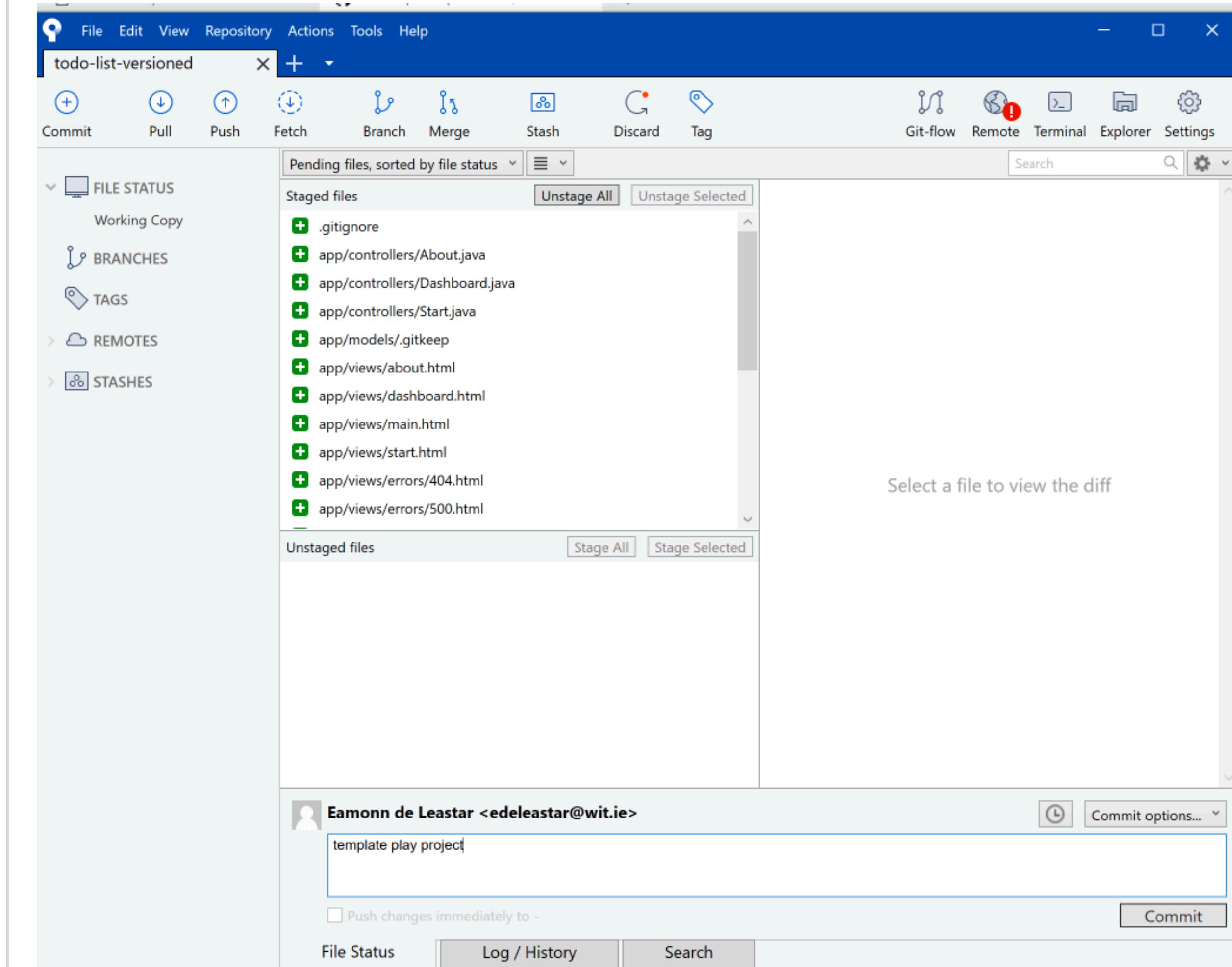
```
git add *
```

This is the first step in the [basic git workflow](http://rogerdudler.github.io/git-guide/). To actually commit these changes use

```
git commit -m "Commit message"
```

Now the file is committed to the **HEAD**, but not in your remote repository yet.

# Sourcetree



# git command line

## pushing changes

Your changes are now in the **HEAD** of your local working copy. To send those changes to your remote repository, execute

```
git push origin master
```

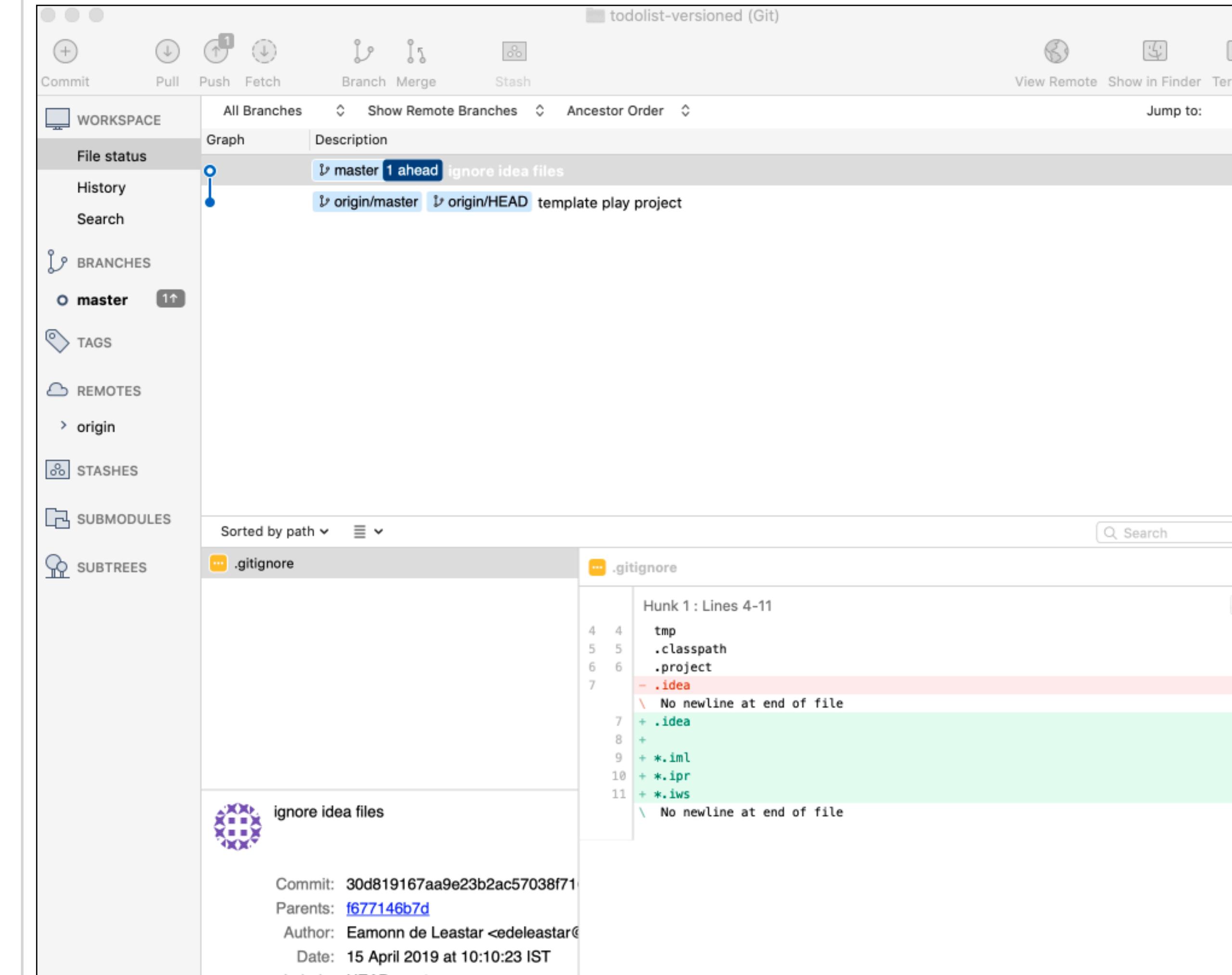
Change *master* to whatever branch you want to push your changes to.

If you have not cloned an existing repository and want to connect your repository to a remote server, you need to add it with

```
git remote add origin <server>
```

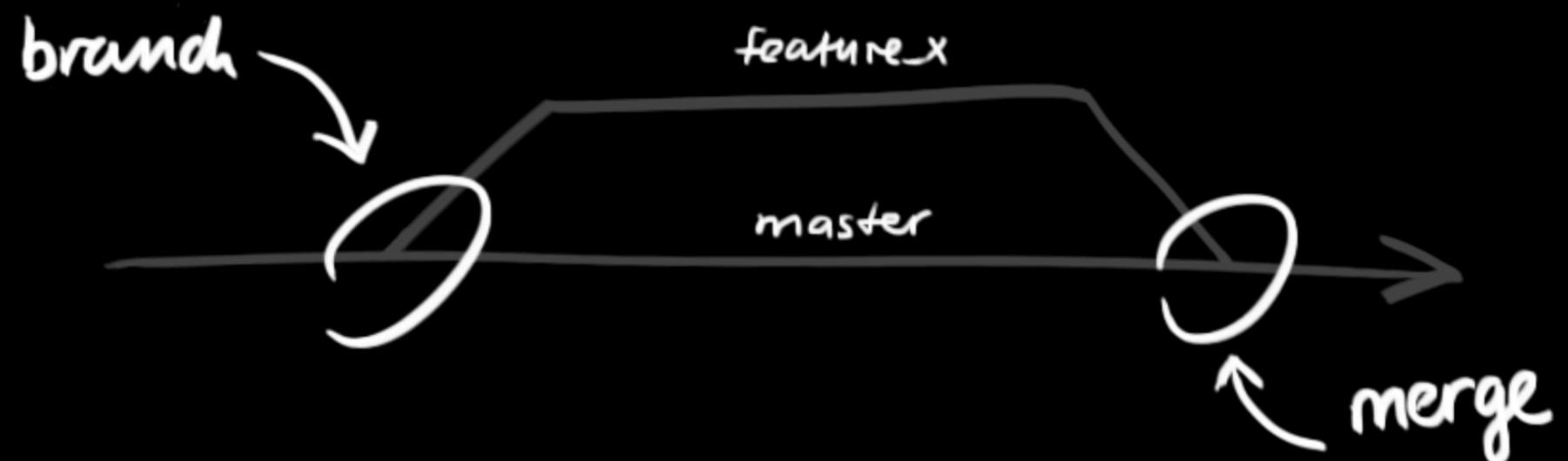
Now you are able to push your changes to the selected remote server

# Sourcetree



# branching

Branches are used to develop features isolated from each other. The *master* branch is the "default" branch when you create a repository. Use other branches for development and merge them back to the master branch upon completion.



create a new branch named "feature\_x" and switch to it using

```
git checkout -b feature_x
```

switch back to master

```
git checkout master
```

and delete the branch again

```
git branch -d feature_x
```

a branch is *not available to others* unless you push the branch to your remote repository

```
git push origin <branch>
```

# update & merge

to update your local repository to the newest commit, execute

```
git pull
```

in your working directory to *fetch* and *merge* remote changes.

to merge another branch into your active branch (e.g. master), use

```
git merge <branch>
```

in both cases git tries to auto-merge changes. Unfortunately, this is not always possible and results in *conflicts*. You are responsible to merge those *conflicts* manually by editing the files shown by git. After changing, you need to mark them as merged with

```
git add <filename>
```

before merging changes, you can also preview them by using

```
git diff <source_branch> <target_branch>
```

# git command line

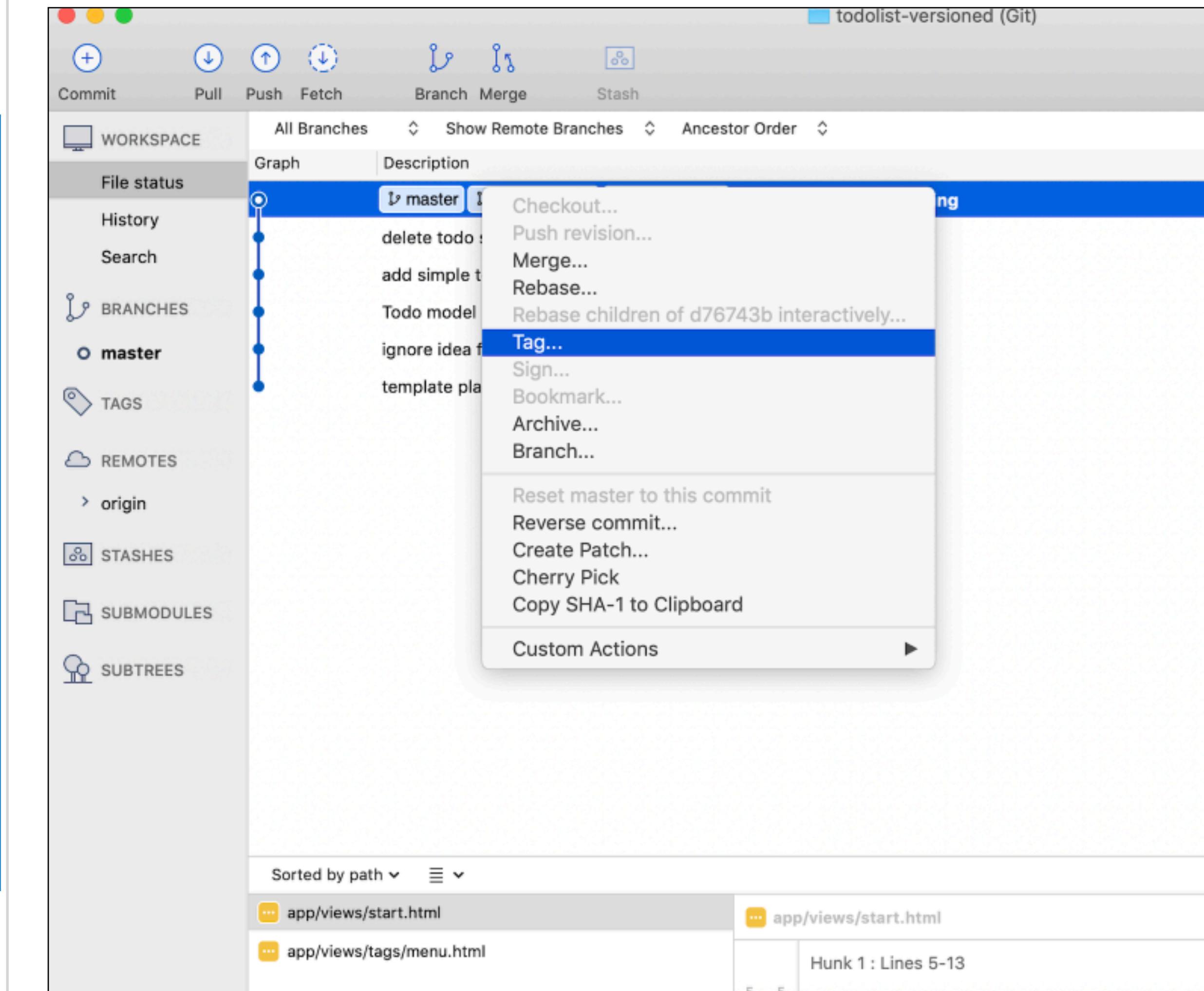
## tagging

it's recommended to create tags for software releases. this is a known concept, which also exists in SVN. You can create a new tag named *1.0.0* by executing

```
git tag 1.0.0 1b2e1d63ff
```

the *1b2e1d63ff* stands for the first 10 characters of the commit id you want to reference with your tag. You can get the commit id by looking at the...

# Sourcetree



# git command line

## log

in its simplest form, you can study repository history using.. `git log`

You can add a lot of parameters to make the log look like what you want.

To see only the commits of a certain author:

```
git log --author=bob
```

To see a very compressed log where each commit is one line:

```
git log --pretty=oneline
```

Or maybe you want to see an ASCII art tree of all the branches,  
decorated with the names of tags and branches:

```
git log --graph --oneline --decorate --all
```

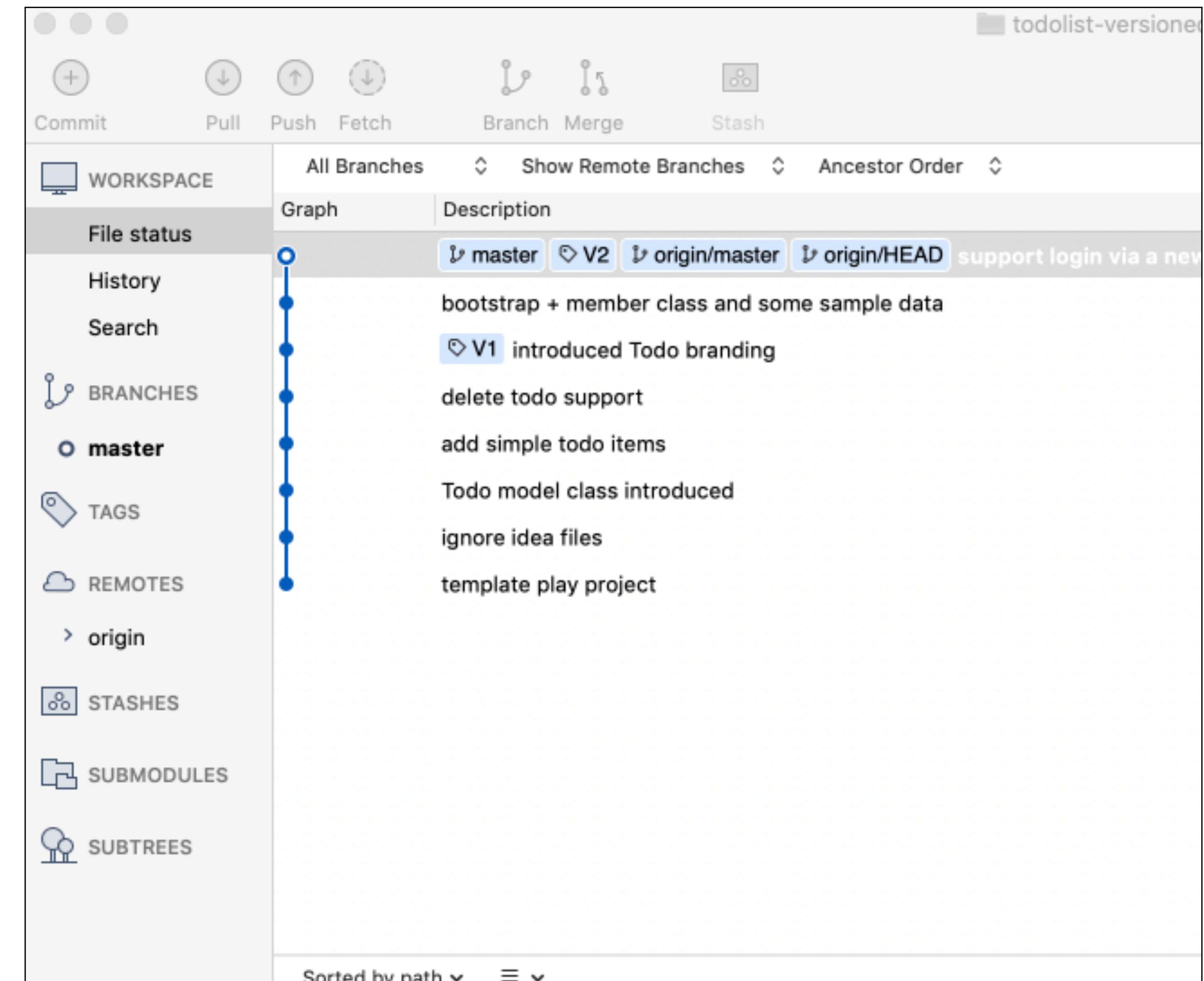
See only which files have changed:

```
git log --name-status
```

These are just a few of the possible parameters you can use. For more,

see `git log --help`

# Sourcetree



# replace local changes

In case you did something wrong, which for sure never happens ;), you can replace local changes using the command

```
git checkout -- <filename>
```

this replaces the changes in your working tree with the last content in HEAD. Changes already added to the index, as well as new files, will be kept.

If you instead want to drop all your local changes and commits, fetch the latest history from the server and point your local master branch at it like this

```
git fetch origin
```

```
git reset --hard origin/master
```

# useful hints

built-in git GUI

`gitk`

use colorful git output

`git config color.ui true`

show log on just one line per commit

`git config format.pretty oneline`

use interactive adding

`git add -i`

# links & resources

<http://rogerdudler.github.io/git-guide/>

## graphical clients

GitX (L) (OSX, open source)

Tower (OSX)

Source Tree (OSX & Windows, free)

GitHub for Mac (OSX, free)

GitBox (OSX, App Store)

## guides

Git Community Book

Pro Git

Think like a git

GitHub Help

A Visual Git Guide

## get help

Git User Mailing List

#git on irc.freenode.net