

Git

and Open Source Distributed
Version Control System



Git is a Distributed Version **Control System**

- content tracker
- it can be used to store content
- it is mostly used to store code
- it is designed to store code

Git is a Distributed **Version Control System**

- when we save something, we are making a version
- when we make a new version we are not saving everything but the incremental changes from previous version

Git is a **Distributed Version Control System**

- git is stored in a server and in a local repository, the computer of the developer.
- it is not centralized in an unique server, so everything published in github, it's free, it's belong to the community.

Git is a Distributed **Version Control System**

- content tracker
- it can be used to store content
- it is mostly used to store code
- it is designed to store code

Download Git

Downloads



Mac OS X



Windows



Linux/Unix

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



Create a Github Account

Join GitHub

Create your account

Username *

Email address *

Password *

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

Email preferences

☐ Send me occasional product updates, announcements, and offers.

Install GithubDesktop or GitKraken

GithubDesktop

The screenshot shows the Github Desktop application. The top bar indicates the current repository is 'desktop', the current branch is 'esc-pr' (commit #3972), and the origin was last fetched 3 minutes ago. The main area is divided into a left sidebar with 'Changes' and 'History' tabs, and a central pane showing the commit history and a diff view. The commit history lists several commits, including 'Add event handler to dropdown component' by IAmWillIShepherd and Markus Olsson. The diff view shows changes to 'app/src/ui/t.../dropdown.tsx', with a table of line numbers and a preview of the code changes, such as adding a 'private get isOpen()' method and updating the 'onDropdownIcon' prop.

GitKraken

The screenshot shows the GitKraken application. The top bar indicates the current repository is 'vscode', the current branch is 'master', and the origin was last fetched 3 minutes ago. The main area is divided into a left sidebar with 'Repository', 'Branch', 'Pull Requests', 'Tags', 'Submodules', and 'GitHub Actions' tabs, and a central pane showing a commit history and a diff view. The commit history lists several commits, including 'Merge branch 'master' into electron-8.0.x' by Benjamin Pasero. The diff view shows changes to 'joh/concat-doc', with a table of line numbers and a preview of the code changes, such as adding a 'private get isOpen()' method and updating the 'onDropdownIcon' prop.

Open the terminal



Git Bash
(Windows)



terminal or iTerm
(Mac)

```
MINGW32~/git
Welcome to Git (version 1.8.3-preview20130601)

Run 'git help git' to display the help index.
Run 'git help <command>' to display help for specific commands.

Bacon@BACON ~
$ git clone https://github.com/msysgit/git.git
Cloning into 'git'...
remote: Counting objects: 177468, done.
remote: Compressing objects: 100% (52057/52057), done.
remote: Total 177468 (delta 133396), reused 166093 (delta 123576)
Receiving objects: 100% (177468/177468), 42.16 MiB | 1.84 MiB/s, done.
Resolving deltas: 100% (133396/133396), done.
Checking out files: 100% (2576/2576), done.

Bacon@BACON ~
$ cd git

Bacon@BACON ~/git (master)
$ git status
# On branch master
nothing to commit, working directory clean

Bacon@BACON ~/git (master)
$
```

```
Macintosh HD ~ bash — 80x24
bash-3.2$
Display all 1468 possibilities? (y or n)

osafolly.com
```

Check git version

```
git --version  
git version 2.28.0
```

Create a local repository

```
cd where/you/want/to/create/a/repo  
mkdir repo_name  
cd repo_name  
ls -la  
git init  
ls -la
```

Initialize git

```
cd where/you/want/to/create/a/repo
mkdir repo_name
cd repo_name
ls -la
.
..
git init
ls -la
.
..
.git
```

Check status

```
git status
```

```
On branch master
```

```
No commits yet
```

```
nothing to commit (create/copy files and  
use "git add" to track)
```

Create a file (make a change)

```
touch new_code.py
```

Check status again

```
git status
```

```
On branch master
```

```
No commits yet
```

```
Untracked files:
```

```
    (use "git add <file>..." to include in  
what will be committed)
```

```
    new_code.py
```

```
nothing added to commit but untracked files  
present (use "git add" to track)
```

Add changes to staging area

```
git add new_code.py
```


Check status

```
git add new_code.py
```

On branch master

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to
unstage)

new file: new_code.py

Check status

```
git add new_code.py
```

On branch master

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to
unstage)

new file: new_code.py

Check status

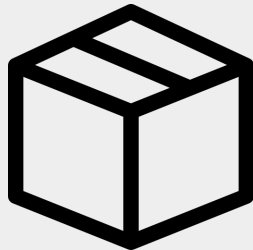
```
git add new_code.py
```

On branch master

No commits

Changes to
 (use "git
unstage)

new file:



The staging area is there to keep track of all the files which are to be committed. Any file which is not added to the staging area will not be committed. This gives the developer control over which files need to be committed.

Committing

```
git commit -m "message"  
[master (root-commit) 9d47752] message  
1 file changed, 0 insertions(+), 0  
deletions(-)  
create mode 100644 new_code.py
```

Committing

```
git commit -m "message"  
[master (root-commit) 9d47752] message  
1 file changed  
deletions(-)  
create mode
```



when we commit we are taking a snapshot of the code, we capture the state of a project at that point in time.

Check the log

```
git log
```

```
9d47752a506f80a630b4b601d3ffdc1c9707e888
```

```
(HEAD → master)
```

```
Author: FranDiego
```

```
<franciscojosediegoacosta@gmail.com>
```

```
Date:    Mon Jun 15 016:53:41 2020 +0200
```

Check the log

```
git log
```

```
9d47752a506f80a630b4b601d3ffdc1c9707e888
```

```
(HEAD → mas
```

```
Author: Fra
```

```
<franciscoj
```

```
Date: Mon
```

```
:q
```

Check the log

```
git log
```

```
9d47752a506f80a630b4b601d3ffdc1c9707e888
```

```
(HEAD → mas
```

```
Author: Fra
```

```
<franciscoj
```

```
Date: Mon
```

```
:q
```


Create a project

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

Owner *



frandiego ▾

/

Repository name *

Great repository names are short and memorable. Need inspiration? How about **jubilant-spork**?

Description (optional)



Public

Anyone on the internet can see this repository. You choose who can commit.




Private

You choose who can see and commit to this repository.

Create a project

Quick setup — if you've done this kind of thing before

 Set up in Desktop

or

HTTPS

SSH

git@github.com:frandiego/test.git



Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# test" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M master
git remote add origin git@github.com:frandiego/test.git
git push -u origin master
```



...or push an existing repository from the command line

```
git remote add origin git@github.com:frandiego/test.git
git branch -M master
git push -u origin master
```



...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code

Copy the git link

Quick setup — if you've done this kind of thing before



Set up in Desktop

or

HTTPS

SSH

git@github.com:frandiego/test.git



Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

Add a remote

```
git remote add origin  
git@github.com:frandiego/test.git
```

Push

```
git push origin master
```

```
Username for 'https://github.com':  
Password
```

Check your repo

 master ▾

 1 branch

 0 tags


Go to file

Add file ▾

 Code ▾



frandiego new code

d10fc4f 3 minutes ago  1 commits



new_code.py

new code

3 minutes ago

Help people interested in this repository understand your project by adding a README.

Add a README

Exercise

1. Create a folder called `code`.
2. Move the code to this new folder
3. Add to staging area
4. Commit
5. Push

Hints

```
mkdir
```

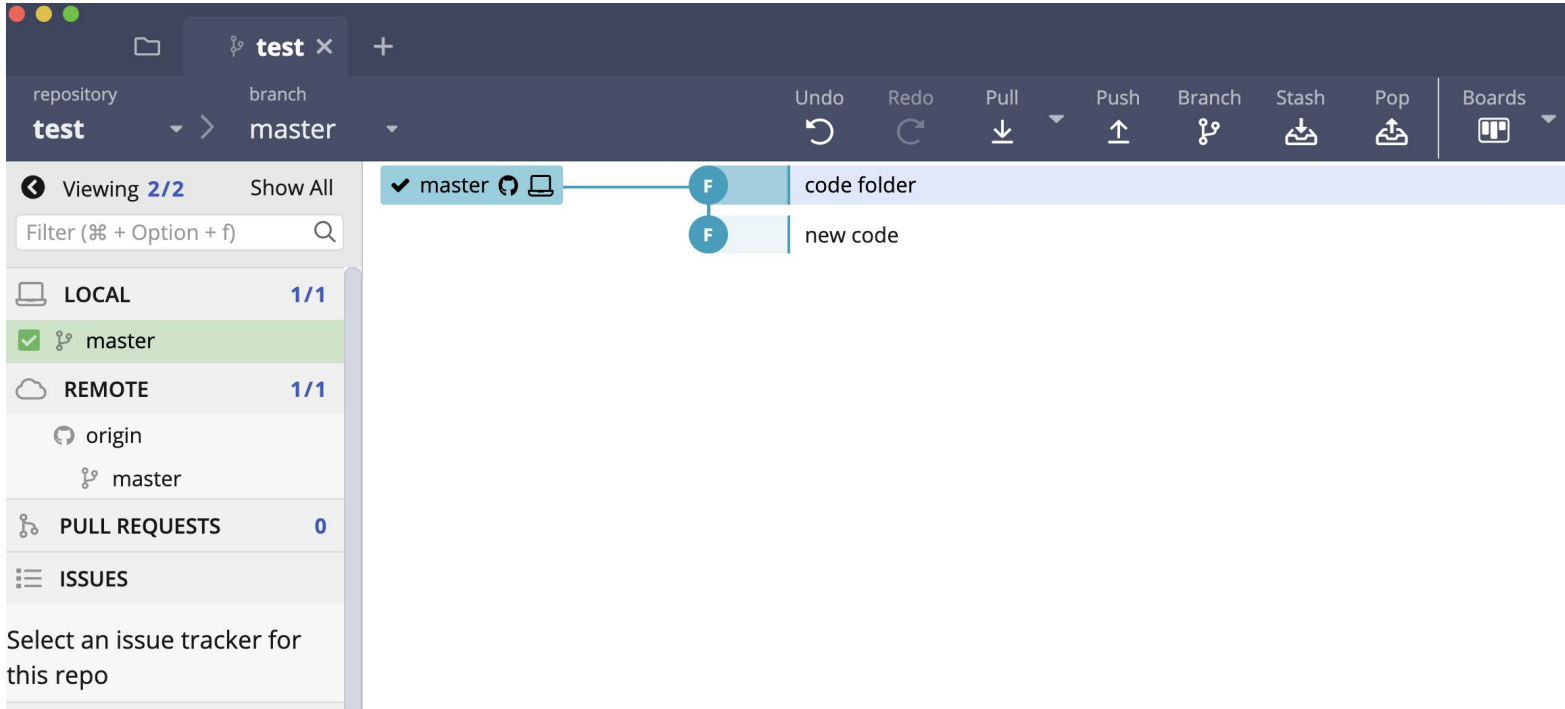
```
mv
```

```
git add
```

```
git commit
```

```
git push
```

Open the project with a UI

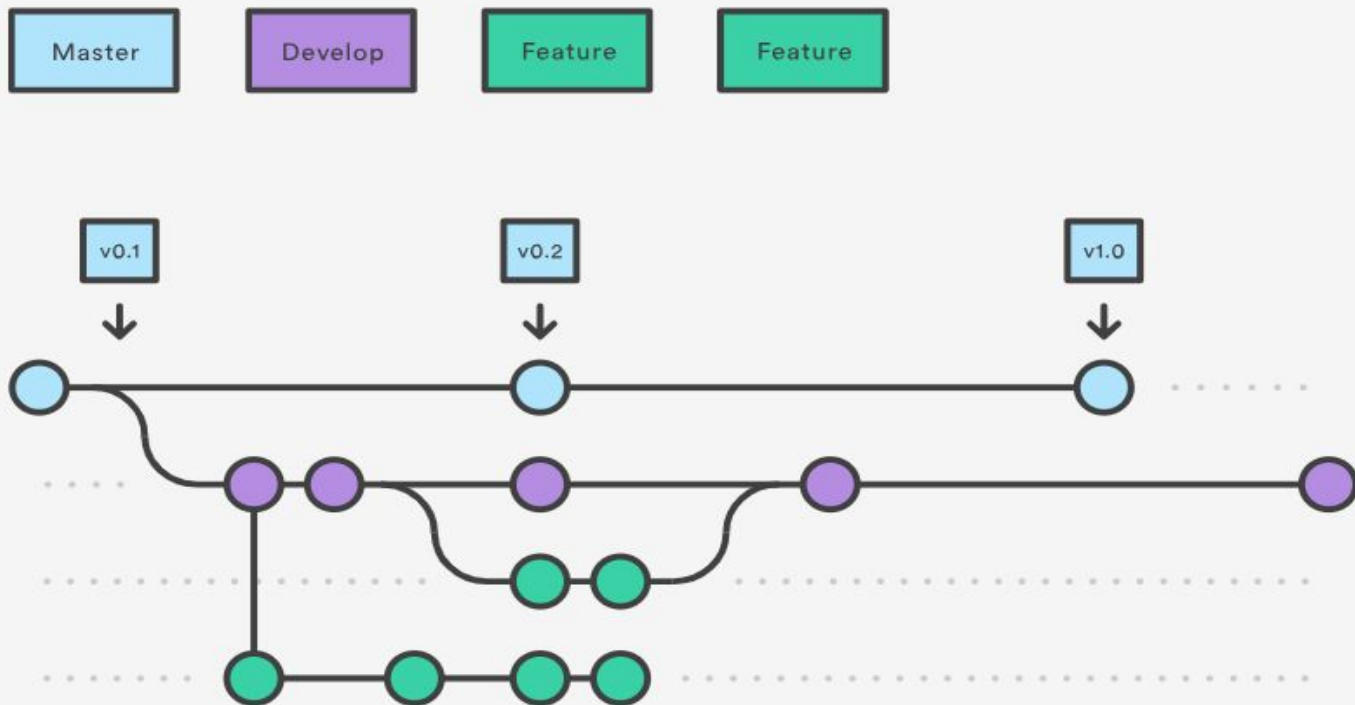


Exercise

1. Open `code/new_code.py`.
2. Write something

```
print("Hello git")
```
3. Add, Commit and Push using the UI

GIT FLOW



GIT FLOW



[Gig Branching](#)

