Git per edizioni digitali collaborative su GitHub

Angelo Mario Del Grosso

angelo.delgrosso@ilc.cnr.it

CNR-ILC visiting VeDPH

Istituto di Linguistica Computazionale “A. Zampolli”,

25th November 2020

Outline

1. VCS Introduction

2. GitHub host platform

3. CLI - shell concepts

4. Git environment by command line interface

5. Git basic branching and merging

6. Conclusions and References

What is my work about

Digital and Computational Philology

Attività di ricerca per lo sviluppo di sistemi di linguistica e filologia digitale e computazionale volti alla produzione, rappresentazione, analisi, fruizione e interrogazione di testi di tradizione medievale, a stampa e di autori moderni e contemporanei.

Topic of the class

Topics of the class

12 ore: gio 26/11/2020 - ven 27/11/2020

- Version Control Systems (VCSs)
- GitHub hosting service
- Git usage through the main CLI commands
- Cloning, modifying, contributing, diffing, logging
- Introduction to remotes
- Branching model and merging
- Some advanced git tools

Topics of the class

12 ore: gio 26/11/2020 - ven 27/11/2020

- Version Control Systems (VCSs)
- GitHub hosting service
- Git usage through the main CLI commands
- Cloning, modifying, contributing, diffing, logging
- Introduction to remotes
- Branching model and merging
- Some advanced git tools

Topics of the class

12 ore: gio 26/11/2020 - ven 27/11/2020

- Version Control Systems (VCSs)
- GitHub hosting service
- Git usage through the main CLI commands
- Cloning, modifying, contributing, diffing, logging
- Introduction to remotes
- Branching model and merging
- Some advanced git tools

Topics of the class

12 ore: gio 26/11/2020 - ven 27/11/2020

- Version Control Systems (VCSs)
- GitHub hosting service
- Git usage through the main CLI commands
- Cloning, modifying, contributing, diffing, logging
- Introduction to remotes
- Branching model and merging
- Some advanced git tools

Topics of the class

12 ore: gio 26/11/2020 - ven 27/11/2020

- Version Control Systems (VCSs)
- GitHub hosting service
- Git usage through the main CLI commands
- Cloning, modifying, contributing, diffing, logging
- Introduction to remotes
- Branching model and merging
- Some advanced git tools

Topics of the class

12 ore: gio 26/11/2020 - ven 27/11/2020

- Version Control Systems (VCSs)
- GitHub hosting service
- Git usage through the main CLI commands
- Cloning, modifying, contributing, diffing, logging
- Introduction to remotes
- Branching model and merging
- Some advanced git tools

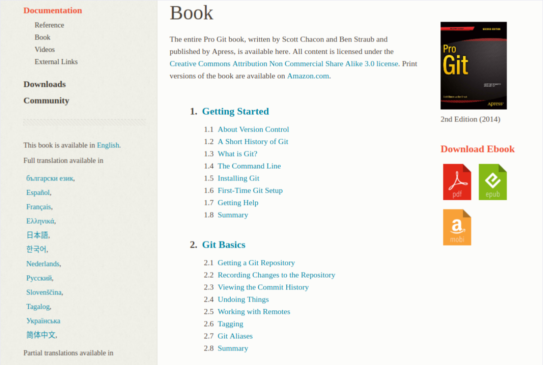
Topics of the class

12 ore: gio 26/11/2020 - ven 27/11/2020

- Version Control Systems (VCSs)
- GitHub hosting service
- Git usage through the main CLI commands
- Cloning, modifying, contributing, diffing, logging
- Introduction to remotes
- Branching model and merging
- Some advanced git tools

Reference Book

Reference book for this class



Documentation

- Reference
- Book
- Videos
- External Links

Downloads

Community

This book is available in English.

Full translation available in

български език,
Español,
Français,
Ελληνικά,
日本語,
한국어,
Nederlands,
Русский,
Slovenščina,
Tagalog,
Українська
简体中文,

Partial translations available in

Book

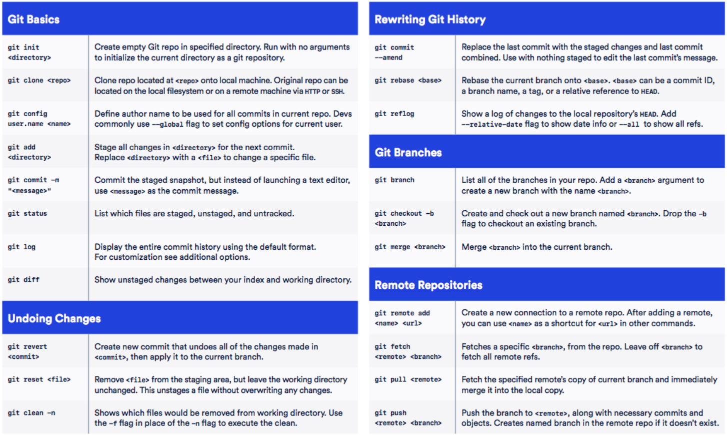
The entire Pro Git book, written by Scott Chacon and Ben Straub and published by Apress, is available here. All content is licensed under the Creative Commons Attribution Non Commercial Share Alike 3.0 license. Print versions of the book are available on Amazon.com.

2nd Edition (2014)

Download Ebook

https://git-scm.com/book/en/v2

Main git commands and concepts



Git Basics

`git init <directory>`	Create empty Git repo in specified directory. Run with no arguments to initialize the current directory as a git repository.
`git clone <repo>`	Clone repo located at `<repo>` onto local machine. Original repo can be located on the local filesystem or on a remote machine via HTTP or SSH.
`git config user.name <name>`	Define author name to be used for all commits in current repo. Devs commonly use `--global` flag to set config options for current user.
`git add <directory>`	Stage all changes in `<directory>` for the next commit. Replace `<directory>` with a `<file>` to change a specific file.
`git commit -m "<message>"`	Commit the staged snapshot, but instead of launching a text editor, use `<message>` as the commit message.
`git status`	List which files are staged, unstaged, and untracked.
`git log`	Display the entire commit history using the default format. For customization see additional options.
`git diff`	Show unstaged changes between your index and working directory.

Undoing Changes

`git revert <commit>`	Create new commit that undoes all of the changes made in `<commit>`, then apply it to the current branch.
`git reset <file>`	Remove `<file>` from the staging area, but leave the working directory unchanged. This unstages a file without overwriting any changes.
`git clean -n`	Shows which files would be removed from working directory. Use the `-f` flag in place of the `-n` flag to execute the clean.

Rewriting Git History

`git commit --amend`	Replace the last commit with the staged changes and last commit combined. Use with nothing staged to edit the last commit's message.
`git rebase <base>`	Rebase the current branch onto `<base>`. `<base>` can be a commit ID, a branch name, a tag, or a relative reference to HEAD.
`git reflog`	Show a log of changes to the local repository's HEAD. Add `--relative-date` flag to show date info or `--all` to show all refs.

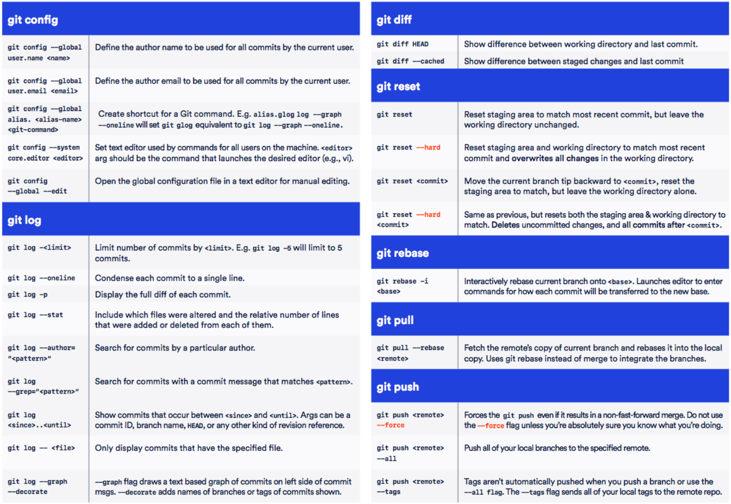
Git Branches

`git branch`	List all of the branches in your repo. Add a `<branch>` argument to create a new branch with the name `<branch>`.
`git checkout -b <branch>`	Create and check out a new branch named `<branch>`. Drop the `-b` flag to checkout an existing branch.
`git merge <branch>`	Merge `<branch>` into the current branch.

Remote Repositories

`git remote add <name> <url>`	Create a new connection to a remote repo. After adding a remote, you can use `<name>` as a shortcut for `<url>` in other commands.
`git fetch <remote> <branch>`	Fetches a specific `<branch>`, from the repo. Leave off `<branch>` to fetch all remote refs.
`git pull <remote>`	Fetch the specified remote's copy of current branch and immediately merge it into the local copy.
`git push <remote> <branch>`	Push the branch to `<remote>`, along with necessary commits and objects. Creates named branch in the remote repo if it doesn't exist.

Main git commands and concepts



git config

`git config --global user.name <name>`	Define the author name to be used for all commits by the current user.
`git config --global user.email <email>`	Define the author email to be used for all commits by the current user.
`git config --global alias. <alias-name> <git-command>`	Create shortcut for a Git command. E.g. `alias.glog log --graph --oneline` will set `git glog` equivalent to `git log --graph --oneline`.
`git config --system core.editor <editor>`	Set text editor used by commands for all users on the machine. `<editor>` arg should be the command that launches the desired editor (e.g., vi).
`git config --global --edit`	Open the global configuration file in a text editor for manual editing.

git log

`git log -<limit>`	Limit number of commits by `<limit>`. E.g. `git log -5` will limit to 5 commits.
`git log --oneline`	Condense each commit to a single line.
`git log -p`	Display the full diff of each commit.
`git log --stat`	Include which files were altered and the relative number of lines that were added or deleted from each of them.
`git log --author= "<pattern>"`	Search for commits by a particular author.
`git log --grep="<pattern>"`	Search for commits with a commit message that matches `<pattern>`.
`git log <since>..<until>`	Show commits that occur between `<since>` and `<until>`. Args can be a commit ID, branch name, `HEAD`, or any other kind of revision reference.
`git log -- <file>`	Only display commits that have the specified file.
`git log --graph --decorate`	`--graph` flag draws a text based graph of commits on left side of commit msgs. `--decorate` adds names of branches or tags of commits shown.

git diff

`git diff HEAD`	Show difference between working directory and last commit.
`git diff --cached`	Show difference between staged changes and last commit

git reset

`git reset`	Reset staging area to match most recent commit, but leave the working directory unchanged.
`git reset --hard`	Reset staging area and working directory to match most recent commit and **overwrites all changes** in the working directory.
`git reset <commit>`	Move the current branch tip backward to `<commit>`, reset the staging area to match, but leave the working directory alone.
`git reset --hard <commit>`	Same as previous, but resets both the staging area & working directory to match. **Deletes** uncommitted changes, and **all commits after** `<commit>`.

git rebase

`git rebase -i <base>`	Interactively rebase current branch onto `<base>`. Launches editor to enter commands for how each commit will be transferred to the new base.

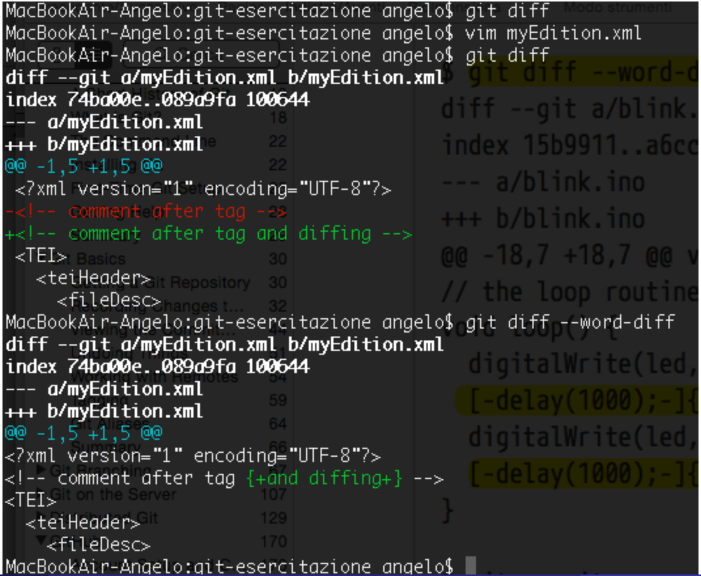
git pull

`git pull --rebase <remote>`	Fetch the remote's copy of current branch and rebases it into the local copy. Uses git rebase instead of merge to integrate the branches.

git push

`git push <remote> --force`	Forces the `git push` even if it results in a non-fast-forward merge. Do not use the `--force` flag unless you're absolutely sure you know what you're doing.
`git push <remote> --all`	Push all of your local branches to the specified remote.
`git push <remote> --tags`	Tags aren't automatically pushed when you push a branch or use the `--all flag`. The `--tags` flag sends all of your local tags to the remote repo.

Command Line Example



```
MacBookAir-Angelo:git-esercitazione angelo$ git diff
MacBookAir-Angelo:git-esercitazione angelo$ vim myEdition.xml
MacBookAir-Angelo:git-esercitazione angelo$ git diff
diff --git a/myEdition.xml b/myEdition.xml
index 74ba00e..089a9fa 100644
--- a/myEdition.xml
+++ b/myEdition.xml
@@ -1,5 +1,5 @@
 <?xml version="1" encoding="UTF-8"?>
-<!-- comment after tag -->
+<!-- comment after tag and diffing -->
 <TEI>
   <teiHeader>
     <fileDesc>
MacBookAir-Angelo:git-esercitazione angelo$ git diff --word-diff
diff --git a/myEdition.xml b/myEdition.xml
index 74ba00e..089a9fa 100644
--- a/myEdition.xml
+++ b/myEdition.xml
@@ -1,5 +1,5 @@
<?xml version="1" encoding="UTF-8"?>
<!-- comment after tag {+and diffing+} -->
<TEI>
  <teiHeader>
    <fileDesc>
MacBookAir-Angelo:git-esercitazione angelo$
```

Progress status

1 VCS Introduction

2 GitHub host platform

3 CLI - shell concepts

4 Git environment by command line interface

5 Git basic branching and merging

6 Conclusions and References

Git intro

Getting started with Git

VCS

A Version Control System (VCS) is a tool that **records changes** to a file or set of files over time so that you can **recall specific versions later**.

Git intro

Getting started with Git

Benefits

- It allows you to **revert** selected files back to a previous state
- **compare** changes over time
- **who** last modified something that might be causing a problem, **when**, **why**, etc
- . . .

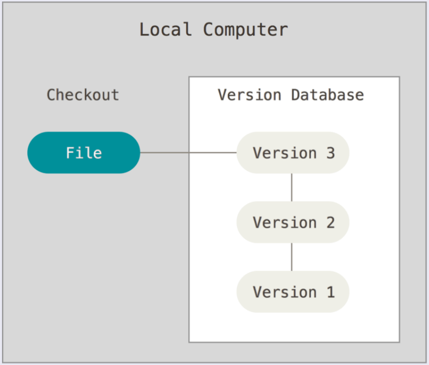
VCS Main feature

Using a VCS also generally means that if you **screw things up** or lose files, you can easily **recover**

Different VCS Architectures

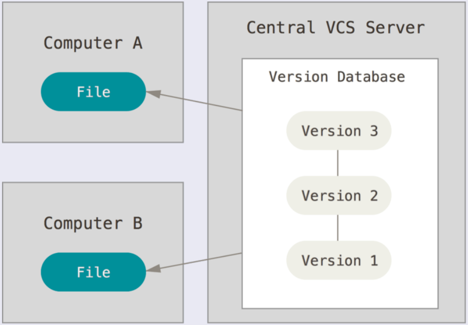
- **Local** Version Control System (*RCS*)
- **Centralized** Version Control System (*CVS, SVN*)
- **Distributed** Version Control System (*GIT, Mercurial*)

Local Version Control System



Database that kept all the changes to files under control

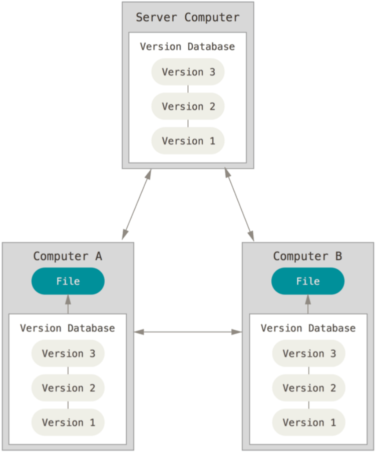
Centralized Version Control System



Need to collaborate: single server that contains all the files

Git and GitHub

Getting started with Git



Client repositories can be copied back up to the server to restore it

GIT DVCS

- Started by Linux community
- Fast and efficient
- Simple design
- non-linear development
- fully distributed
- handle large projects
- easy to use

Git and GitHub

Getting started with Git

GIT DVCS

With Git, every time you commit, or save the state of your project, Git basically **takes a picture of what all your files look like** at that moment and stores a **reference to that snapshot**.

GIT DVCS

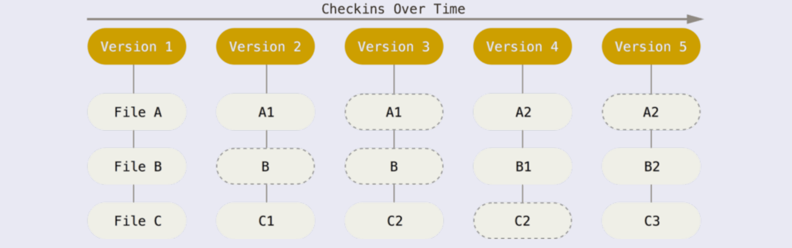
Everything in git is **checksummed before it is stored** and is then referred to by that checksum

GIT DVCS

40-character string composed of hexadecimal characters

a62bc012b405ee47d26b695708063a9f2ffad243

GIT DVCS


Checkins Over Time
Version 1
File A
File B
File C
Version 2
A1
B
C1
Version 3
A1
B
C2
Version 4
A2
B1
C2
Version 5
A2
B2
C3

Git has three main states that your files can reside in

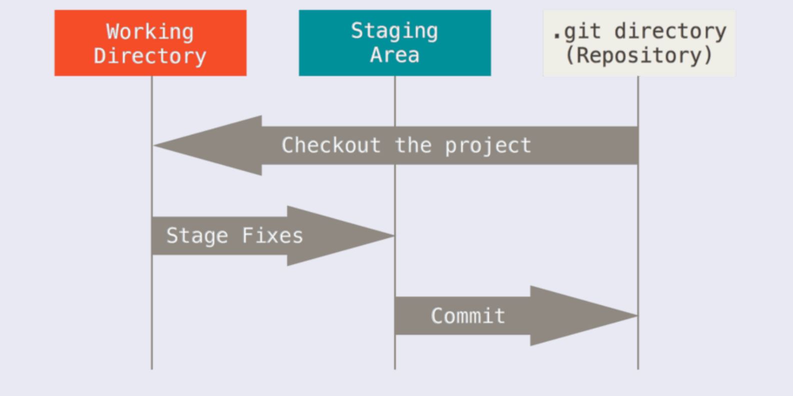
GIT DVCS

- committed
- modified
- staged

Git and GitHub

Getting started with Git

GIT Areas



Git has three main states that your files can reside in

GIT local workflow

- **modify** files in your working tree
- **stage** just those changes you want to be part of your next commit
- do a **commit** which stores that snapshot permanently to your git directory

Progress status

1 VCS Introduction

2 GitHub host platform

3 CLI - shell concepts

4 Git environment by command line interface

5 Git basic branching and merging

6 Conclusions and References

Github

Init a repository

Github platform

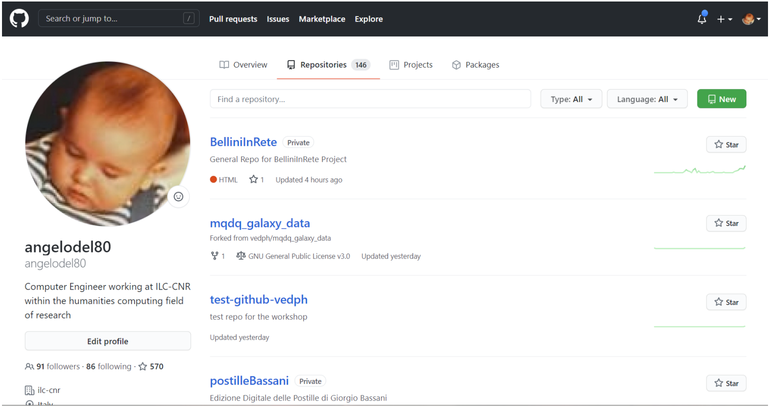
GitHub is the **largest host for git repositories**. It is a central point of **collaboration** among developers.

Github capabilities

Git *hosting*, *issue tracking*, *code review*, *project activities* and many other things

Github

Init a repository



Search or jump to... / Pull requests Issues Marketplace Explore

Overview Repositories 146 Projects Packages

angelodel80
angelodel80

Computer Engineer working at ILC-CNR within the humanities computing field of research

Edit profile

91 followers · 86 following · ☆ 570

ilc-cnr
Italy

Find a repository... Type: All Language: All New

BellinilnRete Private
General Repo for BellinilnRete Project
● HTML ☆ 1 Updated 4 hours ago

mqdq_galaxy_data
Forked from vedph/mqdq_galaxy_data
1 GNU General Public License v3.0 Updated yesterday

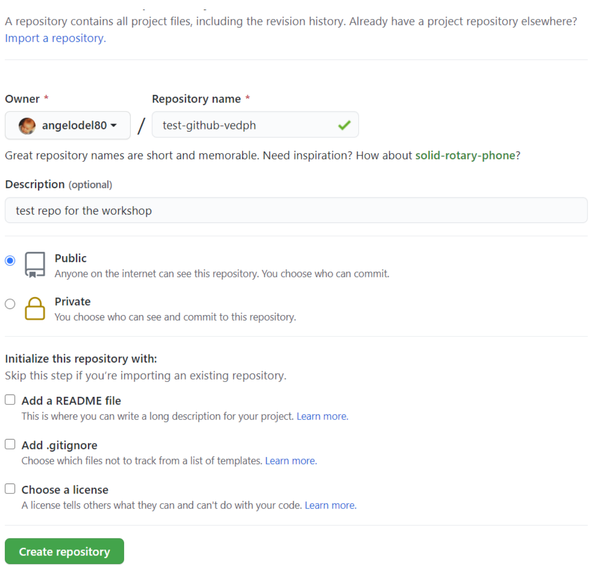
test-github-vedph
test repo for the workshop
Updated yesterday

postilleBassani Private
Edizione Digitale delle Postille di Giorgio Bassani

GitHub - personal profile and repositories

Github

Init a repository



A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner * / Repository name *

angelodel80 / test-github-vedph ✓

Great repository names are short and memorable. Need inspiration? How about solid-rotary-phone?

Description (optional)

test repo for the workshop

● 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.

Initialize this repository with:
Skip this step if you're importing an existing repository.

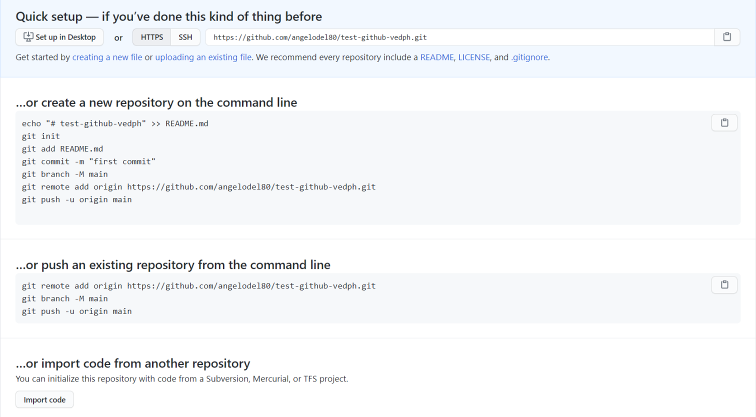
□ Add a README file
This is where you can write a long description for your project. Learn more.

□ Add .gitignore
Choose which files not to track from a list of templates. Learn more.

□ Choose a license
A license tells others what they can and can't do with your code. Learn more.

Github

Init a repository



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

```
https://github.com/angelodel80/test-github-vedph.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-github-vedph" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/angelodel80/test-github-vedph.git
git push -u origin main
```

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

```
git remote add origin https://github.com/angelodel80/test-github-vedph.git
git branch -M main
git push -u origin main
```

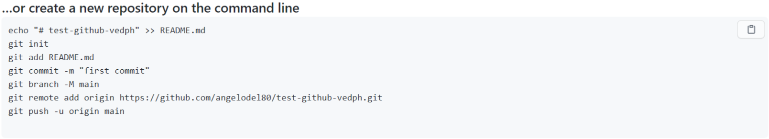
...or import code from another repository

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

GitHub - initialization of a new repository (command line)

Github

Init a repository



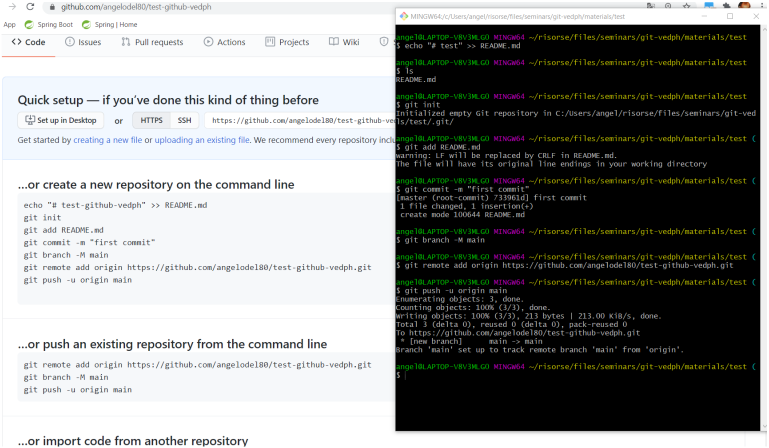
...or create a new repository on the command line

```
echo "# test-github-vedph" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/angelodel80/test-github-vedph.git
git push -u origin main
```

GitHub - focus on the initialization commands for a new repository, tracking resources with a local git node (command line)

Github

Init a repository


github.com/angelodel80/test-github-vedph
App
Spring Boot
Spring | Home
Code
Issues
Pull requests
Actions
Projects
Wiki
Set up in Desktop
or
HTTPS
SSH
MINGW64:/c/Users/angel/risorse/files/seminars/git-vedph/materials/test

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

```
https://github.com/angelodel80/test-github-ve
```

Get started by creating a new file or uploading an existing file. We recommend every repository inclu

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

```
echo "# test-github-vedph" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/angelodel80/test-github-vedph.git
git push -u origin main
```

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

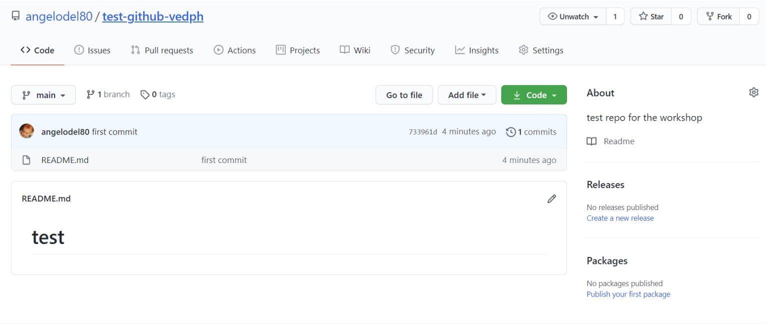
```
git remote add origin https://github.com/angelodel80/test-github-vedph.git
git branch -M main
git push -u origin main
```

...or import code from another repository

```
angel@LAPTOP-V8V3MLGO MINGW64 ~/risorse/files/seminars/git-vedph/materials/test
$ echo "# test" >> README.md

angel@LAPTOP-V8V3MLGO MINGW64 ~/risorse/files/seminars/git-vedph/materials/test
$ ls
README.md

angel@LAPTOP-V8V3MLGO MINGW64 ~/risorse/files/seminars/git-vedph/materials/test
$ git init
Initialized empty Git repository in C:/Users/angel/risorse/files/seminars/git-ved
ls/test/.git/

angel@LAPTOP-V8V3MLGO MINGW64 ~/risorse/files/seminars/git-vedph/materials/test (
$ git add README.md
warning: LF will be replaced by CRLF in README.md.
The file will have its original line endings in your working directory

angel@LAPTOP-V8V3MLGO MINGW64 ~/risorse/files/seminars/git-vedph/materials/test (
$ git commit -m "first commit"
[master (root-commit) 733961d] first commit
 1 file changed, 1 insertion(+)
 create mode 100644 README.md

angel@LAPTOP-V8V3MLGO MINGW64 ~/risorse/files/seminars/git-vedph/materials/test (
$ git branch -M main

angel@LAPTOP-V8V3MLGO MINGW64 ~/risorse/files/seminars/git-vedph/materials/test (
$ git remote add origin https://github.com/angelodel80/test-github-vedph.git

angel@LAPTOP-V8V3MLGO MINGW64 ~/risorse/files/seminars/git-vedph/materials/test (
$ git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 213 bytes | 213.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/angelodel80/test-github-vedph.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.

angel@LAPTOP-V8V3MLGO MINGW64 ~/risorse/files/seminars/git-vedph/materials/test (
$
```

Github

Init a repository



angelodel80 / test-github-vedph

angelodel80 first commit 733961d 4 minutes ago 1 commits

README.md first commit 4 minutes ago

README.md

test

About

test repo for the workshop

Readme

Releases

No releases published
Create a new release

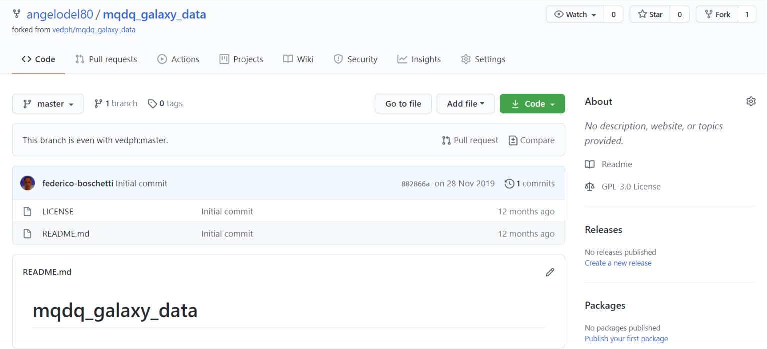
Packages

No packages published
Publish your first package

GitHub - a new repository just created

Github

Init a repository



angelodel80 / mqdq_galaxy_data

forked from vedph/mqdq_galaxy_data

This branch is even with vedph:master.

federico-boschetti Initial commit 882866a on 28 Nov 2019 1 commits

LICENSE	Initial commit	12 months ago
README.md	Initial commit	12 months ago

README.md

mqdq_galaxy_data

About

No description, website, or topics provided.

Readme

GPL-3.0 License

Releases

No releases published
Create a new release

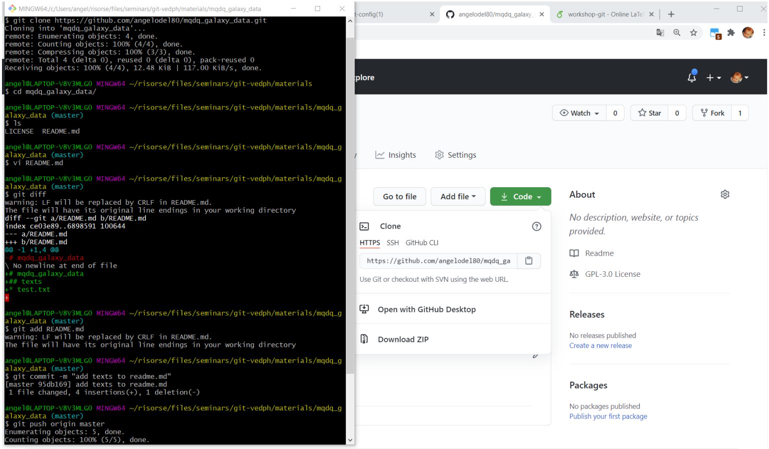
Packages

No packages published
Publish your first package

GitHub - New personal repository forked from an existing one

Github

Init a repository



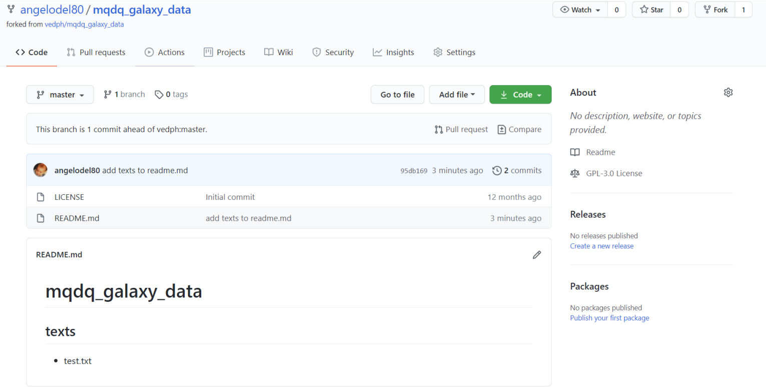
MINGW64:/c/Users/angel/risorse/files/seminars/git-vedph/materials/mqdq_galaxy_data

```
$ git clone https://github.com/angelodel80/mqdq_galaxy_data.git
Cloning into 'mqdq_galaxy_data'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), 12.48 KiB | 117.00 KiB/s, done.

angel@LAPTOP-V8V3MLGO MINGW64 ~/risorse/files/seminars/git-vedph/materials
$ cd mqdq_galaxy_data/

angel@LAPTOP-V8V3MLGO MINGW64 ~/risorse/files/seminars/git-vedph/materials/mqdq_g
alaxy_data (master)
$ ls
LICENSE  README.md

angel@LAPTOP-V8V3MLGO MINGW64 ~/risorse/files/seminars/git-vedph/materials/mqdq_g
alaxy_data (master)
$ vi README.md

angel@LAPTOP-V8V3MLGO MINGW64 ~/risorse/files/seminars/git-vedph/materials/mqdq_g
alaxy_data (master)
$ git diff
warning: LF will be replaced by CRLF in README.md.
The file will have its original line endings in your working directory
diff --git a/README.md b/README.md
index ce03e89..6898591 100644
--- a/README.md
+++ b/README.md
@@ -1 +1,4 @@
-# mqdq_galaxy_data
\ No newline at end of file
+# mqdq_galaxy_data
+## texts
+* test.txt
+

angel@LAPTOP-V8V3MLGO MINGW64 ~/risorse/files/seminars/git-vedph/materials/mqdq_g
alaxy_data (master)
$ git add README.md
warning: LF will be replaced by CRLF in README.md.
The file will have its original line endings in your working directory

angel@LAPTOP-V8V3MLGO MINGW64 ~/risorse/files/seminars/git-vedph/materials/mqdq_g
alaxy_data (master)
$ git commit -m "add texts to readme.md"
[master 95db169] add texts to readme.md
 1 file changed, 4 insertions(+), 1 deletion(-)

angel@LAPTOP-V8V3MLGO MINGW64 ~/risorse/files/seminars/git-vedph/materials/mqdq_g
alaxy_data (master)
$ git push origin master
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
```

GitHub - contributing to a forked repository

Github

Init a repository



angelodel80 / mqdq_galaxy_data

forked from vedph/mqdq_galaxy_data

This branch is 1 commit ahead of vedph:master.

angelodel80 add texts to readme.md 95db169 3 minutes ago 2 commits

LICENSE Initial commit 12 months ago

README.md add texts to readme.md 3 minutes ago

README.md

mqdq_galaxy_data

texts

- test.txt

About

No description, website, or topics provided.

Readme

GPL-3.0 License

Releases

No releases published

Create a new release

Packages

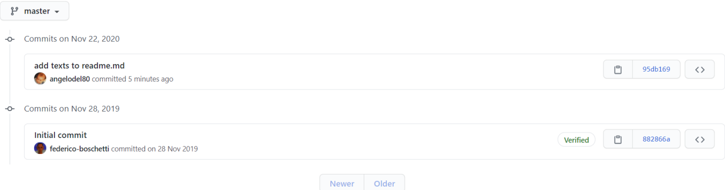
No packages published

Publish your first package

GitHub - contributing to a forked repository

Github

Init a repository



master

Commits on Nov 22, 2020

add texts to readme.md

angelodel80 committed 5 minutes ago

95db169

Commits on Nov 28, 2019

Initial commit

federico-boschetti committed on 28 Nov 2019

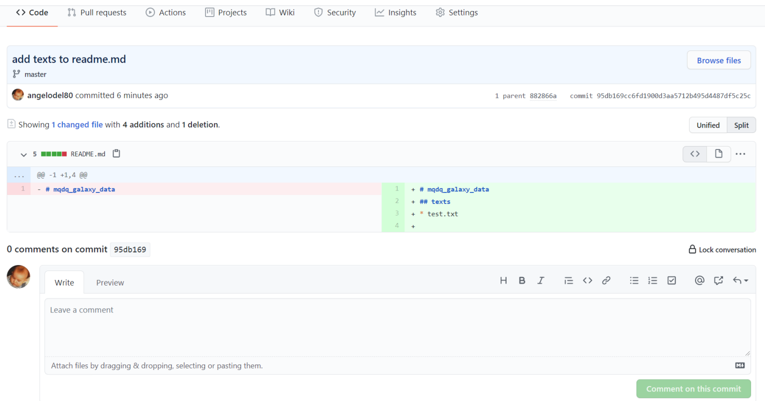
Verified

882866a

GitHub - history of commits in contributing to a forked repository

Github

Init a repository



add texts to readme.md

master

angelodel80 committed 6 minutes ago

1 parent 882866a commit 95db169cc6fd1900d3aa5712b495d4487df5c25c

Showing 1 changed file with 4 additions and 1 deletion.

```
5 ■■■■■ README.md
@@ -1 +1,4 @@
1  - # mqdq_galaxy_data      1  + # mqdq_galaxy_data
                            2  + ## texts
                            3  + * test.txt
                            4  +
```

0 comments on commit 95db169

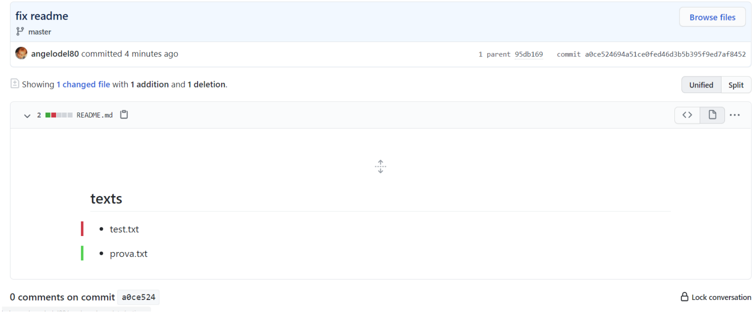
Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

GitHub - Comparing changes in contributing to a forked repository

Github

Init a repository



fix readme

master

angelodel80 committed 4 minutes ago

1 parent 95db169 commit a0ce524694a51ce0fed46d3b5b395f9ed7af8452

Showing 1 changed file with 1 addition and 1 deletion.

2 README.md

texts

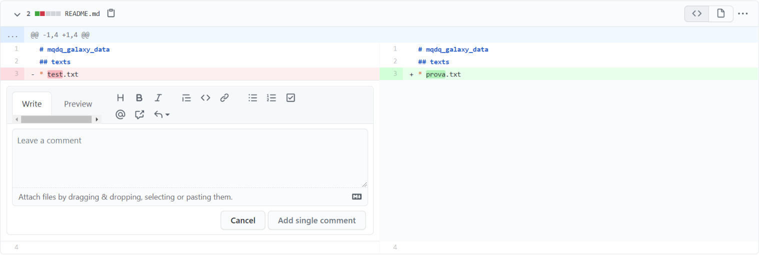
- test.txt
- prova.txt

0 comments on commit a0ce524

GitHub - Comparing changes in contributing to a forked repository

Github

Init a repository



```
2 README.md
@@ -1,4 +1,4 @@
  # mqdq_galaxy_data
  ## texts
- * test.txt
+ * prova.txt
```

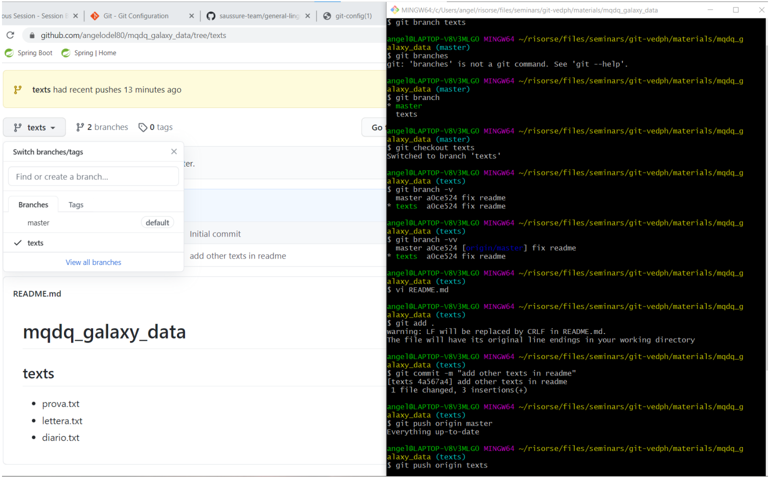
Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

GitHub - Commenting changes in contributing to a forked repository

Github

Init a repository



texts had recent pushes 13 minutes ago

Initial commit

add other texts in readme

README.md

mqdq_galaxy_data

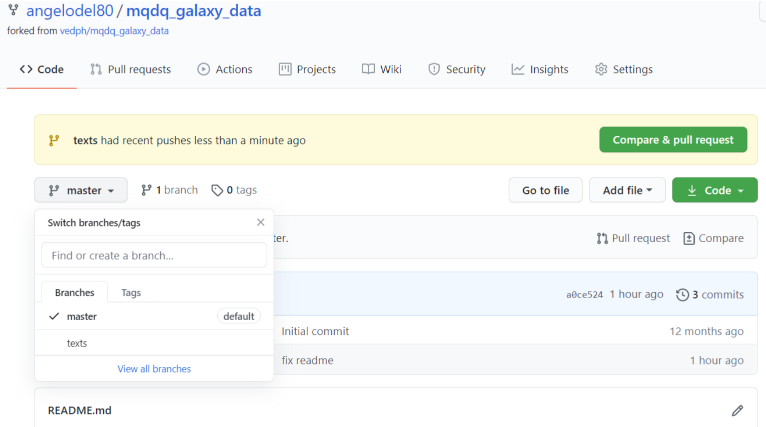
texts

- prova.txt
- lettera.txt
- diario.txt

MINGW64:/c/Users/angel/risorse/files/seminars/git-vedph/materials/mqdq_galaxy_data

```
$ git branch texts

angel@LAPTOP-V8V3MLGO MINGW64 ~/risorse/files/seminars/git-vedph/materials/mqdq_g
alaxy_data (master)
$ git branches
git: 'branches' is not a git command. See 'git --help'.

angel@LAPTOP-V8V3MLGO MINGW64 ~/risorse/files/seminars/git-vedph/materials/mqdq_g
alaxy_data (master)
$ git branch
* master
  texts

angel@LAPTOP-V8V3MLGO MINGW64 ~/risorse/files/seminars/git-vedph/materials/mqdq_g
alaxy_data (master)
$ git checkout texts
Switched to branch 'texts'

angel@LAPTOP-V8V3MLGO MINGW64 ~/risorse/files/seminars/git-vedph/materials/mqdq_g
alaxy_data (texts)
$ git branch -v
  master a0ce524 fix readme
* texts  a0ce524 fix readme

angel@LAPTOP-V8V3MLGO MINGW64 ~/risorse/files/seminars/git-vedph/materials/mqdq_g
alaxy_data (texts)
$ git branch -vv
  master a0ce524 [origin/master] fix readme
* texts  a0ce524 fix readme

angel@LAPTOP-V8V3MLGO MINGW64 ~/risorse/files/seminars/git-vedph/materials/mqdq_g
alaxy_data (texts)
$ vi README.md

angel@LAPTOP-V8V3MLGO MINGW64 ~/risorse/files/seminars/git-vedph/materials/mqdq_g
alaxy_data (texts)
$ git add .
warning: LF will be replaced by CRLF in README.md.
The file will have its original line endings in your working directory

angel@LAPTOP-V8V3MLGO MINGW64 ~/risorse/files/seminars/git-vedph/materials/mqdq_g
alaxy_data (texts)
$ git commit -m "add other texts in readme"
[texts 4a567a4] add other texts in readme
 1 file changed, 3 insertions(+)

angel@LAPTOP-V8V3MLGO MINGW64 ~/risorse/files/seminars/git-vedph/materials/mqdq_g
alaxy_data (texts)
$ git push origin master
Everything up-to-date

angel@LAPTOP-V8V3MLGO MINGW64 ~/risorse/files/seminars/git-vedph/materials/mqdq_g
alaxy_data (texts)
$ git push origin texts
```

GitHub - Branching in contributing to a forked repository



angelodel80 / mqdq_galaxy_data
forked from vedph/mqdq_galaxy_data

texts had recent pushes less than a minute ago

✓ master default
texts

a0ce524 1 hour ago 3 commits

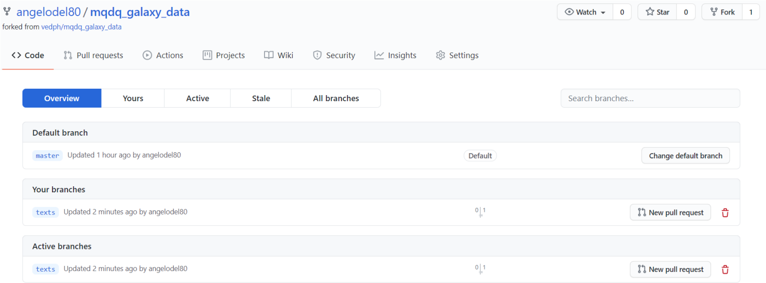
Initial commit	12 months ago
fix readme	1 hour ago

README.md

GitHub - Branching in contributing to a forked repository

Github

Init a repository


angelodel80 / mqdq_galaxy_data

forked from vedph/mqdq_galaxy_data

Default branch

master Updated 1 hour ago by angelodel80 Default

Your branches

texts Updated 2 minutes ago by angelodel80 0 | 1

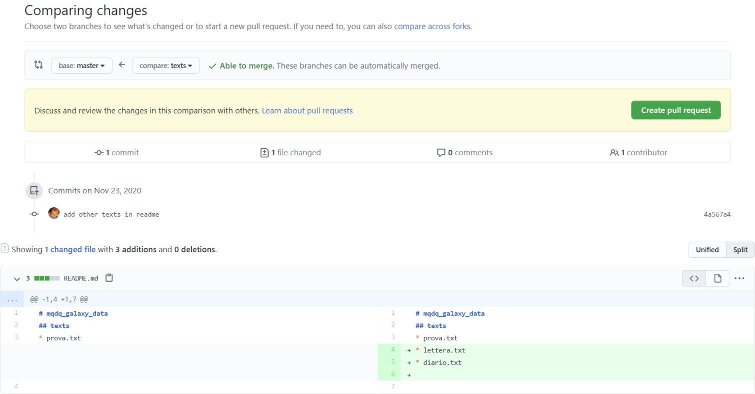
Active branches

texts Updated 2 minutes ago by angelodel80 0 | 1

GitHub - Branching in contributing to a forked repository

Github

Init a repository



Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also compare across forks.

base: master ← compare: texts ✓ **Able to merge.** These branches can be automatically merged.

Discuss and review the changes in this comparison with others. Learn about pull requests

1 commit **1** file changed **0** comments **1** contributor

Commits on Nov 23, 2020

add other texts in readme 4a567a4

Showing **1 changed file** with **3 additions** and **0 deletions**.

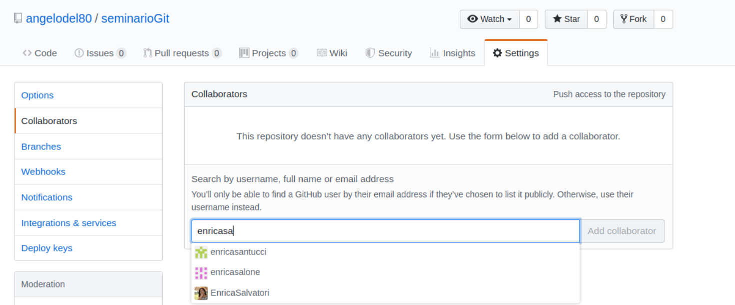
3 README.md

```
@@ -1,4 +1,7 @@
1   # mqdq_galaxy_data        1   # mqdq_galaxy_data
2   ## texts                  2   ## texts
3   * prova.txt               3   * prova.txt
                              4 + * lettera.txt
                              5 + * diario.txt
                              6 +
4                             7
```

GitHub - Comparing branches in contributing to a forked repository

Github

adding collaborators



This repository doesn't have any collaborators yet. Use the form below to add a collaborator.

Search by username, full name or email address

You'll only be able to find a GitHub user by their email address if they've chosen to list it publicly. Otherwise, use their username instead.

enricasa

enricasantucci

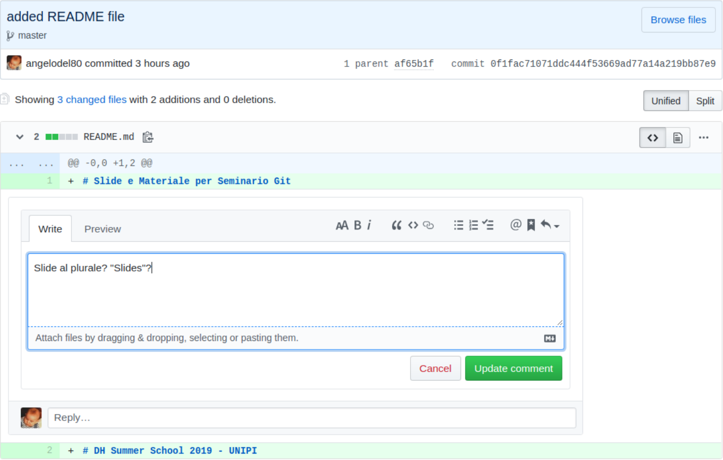
enricasalone

EnricaSalvatori

GitHub - Adding collaborators to a repository

Github

Comments to content lines



added README file

master

angelodel80 committed 3 hours ago

1 parent af65b1f commit 0f1fac71071ddc444f53669ad77a14a219bb87e9

Showing 3 changed files with 2 additions and 0 deletions.

```
2 README.md
@@ -0,0 +1,2 @@
1 + # Slide e Materiale per Seminario Git
```

Slide al plurale? "Slides"?

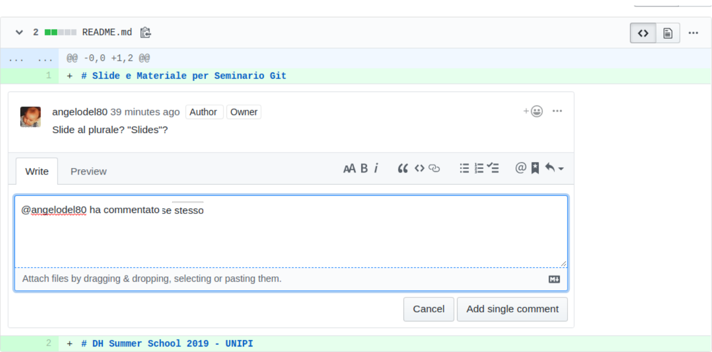
Attach files by dragging & dropping, selecting or pasting them.

```
2 + # DH Summer School 2019 - UNIPI
```

GitHub - Adding comments to a repository

Github

Comments to content lines



```
2 README.md
@@ -0,0 +1,2 @@
+ # Slide e Materiale per Seminario Git
```

angelodel80 39 minutes ago Author Owner

Slide al plurale? "Slides"?

@angelodel80 ha commentato se stesso

Attach files by dragging & dropping, selecting or pasting them.
Cancel
Add single comment

```
+ # DH Summer School 2019 - UNIPI
```

GitHub - Tagging collaborators to a comment

Progress status

1 VCS Introduction

2 GitHub host platform

3 CLI - shell concepts

4 Git environment by command line interface

5 Git basic branching and merging

6 Conclusions and References

Command Line - shell

Getting started with Git

Command Line

The shell is a program that takes keyboard commands and passes them to the operating system to carry out some task.

Terminal Emulator

We need another program called a terminal emulator able to give us access to the shell.

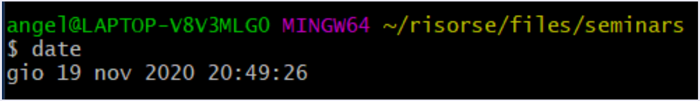
Command Line - shell

Getting started with Git

Command Line concepts

- shell prompt
- cursor
- command (with options and arguments)
- help, man
- file system
- exit

shell command line



```
angel@LAPTOP-V8V3MLGO MINGW64 ~/risorse/files/seminars
$ date
gio 19 nov 2020 20:49:26
```

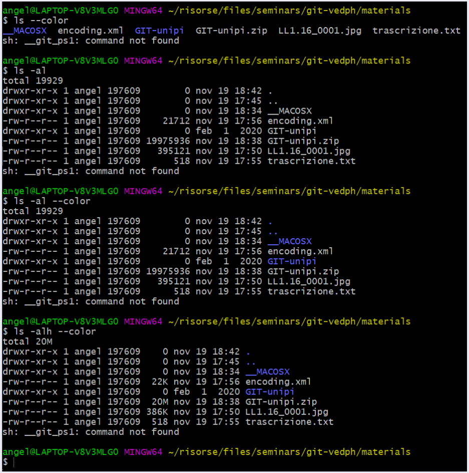
Command Line - shell

Getting started with Git

some useful commands

- navigate the file system *(pwd, cd, ls)*
- manipulate the system *(cp, mv, mkdir, rm, ln)*
- redirection *(cat, wc, head, tail, touch)*
- history *(clear, history, stat)*
- Multitasking *(ps, top, jobs, bg, fg, kill, killall)*
- search files *(locate, find, xargs)*
- archiving *(tar, zip, gzip, rsync)*
- text processing *(echo, sort, uniq, diff, tr, sed)*

ls command examples



```
angel@LAPTOP-V8V3MLGO MINGW64 ~/risorse/files/seminars/git-vedph/materials
$ ls --color
__MACOSX  encoding.xml  GIT-unipi  GIT-unipi.zip  LL1.16_0001.jpg  trascrizione.txt
sh: __git_ps1: command not found

angel@LAPTOP-V8V3MLGO MINGW64 ~/risorse/files/seminars/git-vedph/materials
$ ls -al
total 19929
drwxr-xr-x 1 angel 197609        0 nov 19 18:42 .
drwxr-xr-x 1 angel 197609        0 nov 19 17:45 ..
drwxr-xr-x 1 angel 197609        0 nov 19 18:34 __MACOSX
-rw-r--r-- 1 angel 197609    21712 nov 19 17:56 encoding.xml
drwxr-xr-x 1 angel 197609        0 feb  1  2020 GIT-unipi
-rw-r--r-- 1 angel 197609 19975936 nov 19 18:38 GIT-unipi.zip
-rw-r--r-- 1 angel 197609   395121 nov 19 17:50 LL1.16_0001.jpg
-rw-r--r-- 1 angel 197609      518 nov 19 17:55 trascrizione.txt
sh: __git_ps1: command not found

angel@LAPTOP-V8V3MLGO MINGW64 ~/risorse/files/seminars/git-vedph/materials
$ ls -al --color
total 19929
drwxr-xr-x 1 angel 197609        0 nov 19 18:42 .
drwxr-xr-x 1 angel 197609        0 nov 19 17:45 ..
drwxr-xr-x 1 angel 197609        0 nov 19 18:34 __MACOSX
-rw-r--r-- 1 angel 197609    21712 nov 19 17:56 encoding.xml
drwxr-xr-x 1 angel 197609        0 feb  1  2020 GIT-unipi
-rw-r--r-- 1 angel 197609 19975936 nov 19 18:38 GIT-unipi.zip
-rw-r--r-- 1 angel 197609   395121 nov 19 17:50 LL1.16_0001.jpg
-rw-r--r-- 1 angel 197609      518 nov 19 17:55 trascrizione.txt
sh: __git_ps1: command not found

angel@LAPTOP-V8V3MLGO MINGW64 ~/risorse/files/seminars/git-vedph/materials
$ ls -alh --color
total 20M
drwxr-xr-x 1 angel 197609    0 nov 19 18:42 .
drwxr-xr-x 1 angel 197609    0 nov 19 17:45 ..
drwxr-xr-x 1 angel 197609    0 nov 19 18:34 __MACOSX
-rw-r--r-- 1 angel 197609  22K nov 19 17:56 encoding.xml
drwxr-xr-x 1 angel 197609    0 feb  1  2020 GIT-unipi
-rw-r--r-- 1 angel 197609  20M nov 19 18:38 GIT-unipi.zip
-rw-r--r-- 1 angel 197609 386K nov 19 17:50 LL1.16_0001.jpg
-rw-r--r-- 1 angel 197609  518 nov 19 17:55 trascrizione.txt
sh: __git_ps1: command not found

angel@LAPTOP-V8V3MLGO MINGW64 ~/risorse/files/seminars/git-vedph/materials
$
```

Progress status

1 VCS Introduction

2 GitHub host platform

3 CLI - shell concepts

4 Git environment by command line interface

5 Git basic branching and merging

6 Conclusions and References

Git command line environment

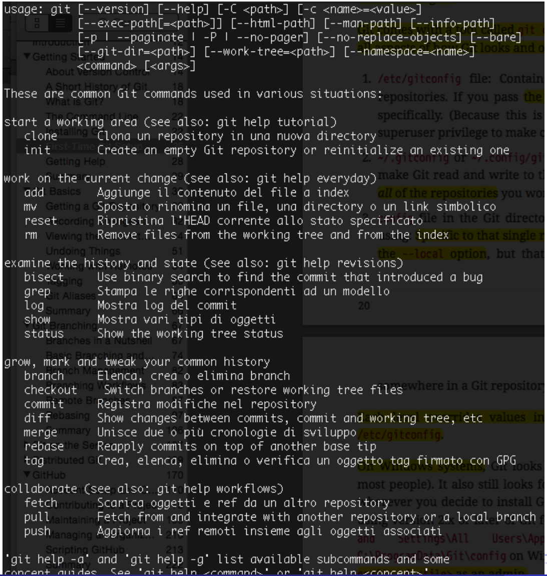
The command line is the only place you can run all Git commands.

GUIs environment

GUIs implement only a partial subset of Git functionality for simplicity

Git and GitHub

Getting started with Git



```
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
   clone      Clona un repository in una nuova directory
   init       Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
   add        Aggiunge il contenuto del file a index
   mv         Sposta o rinomina un file, una directory o un link simbolico
   reset      Ripristina l'HEAD corrente allo stato specificato
   rm         Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
   bisect     Use binary search to find the commit that introduced a bug
   grep       Stampa le righe corrispondenti ad un modello
   log        Mostra log del commit
   show       Mostra vari tipi di oggetti
   status     Show the working tree status

grow, mark and tweak your common history
   branch     Elenca, crea o elimina branch
   checkout    Switch branches or restore working tree files
   commit     Registra modifiche nel repository
   diff       Show changes between commits, commit and working tree, etc
   merge      Unisce due o più cronologie di sviluppo
   rebase     Reapply commits on top of another base tip
   tag        Crea, elenca, elimina o verifica un oggetto tag firmato con GPG

collaborate (see also: git help workflows)
   fetch      Scarica oggetti e ref da un altro repository
   pull       Fetch from and integrate with another repository or a local branch
   push       Aggiorna i ref remoti insieme agli oggetti associati

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
```

Git comes with a tool called **git config** that lets you get and set configuration variables that control all aspects of how Git looks and operates

git config

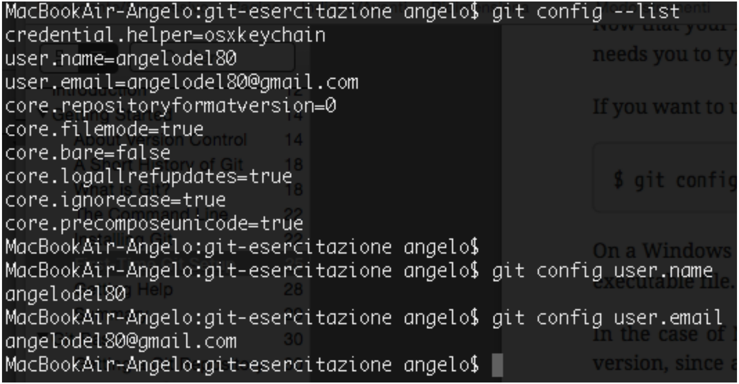
- system (all users, all repositories)
- global (all repositories, single user)
- local (single repository, single user)

The first thing you should do when you install Git is to set your **user** name and **email** address

git config

- `git config --global user.name "Angelo Mario Del Grosso"`
- `git config --global user.email "angelo.delgrosso@ilc.cnr.it"`

Checking Your Settings



```
MacBookAir-Angelo:git-esercitazione angelo$ git config --list
credential.helper=osxkeychain
user.name=angelodel80
user.email=angelodel80@gmail.com
core.repositoryformatversion=0
core.filemode=true
core.bare=false
core.logallrefupdates=true
core.ignorecase=true
core.precomposeunicode=true
MacBookAir-Angelo:git-esercitazione angelo$
MacBookAir-Angelo:git-esercitazione angelo$ git config user.name
angelodel80
MacBookAir-Angelo:git-esercitazione angelo$ git config user.email
angelodel80@gmail.com
MacBookAir-Angelo:git-esercitazione angelo$
```

Git and GitHub

Getting started with Git

help while using git

- `git help <verb>`
- `man git-<verb>`
- `git <verb> --help`
- `git <verb> -h`

fundamental capabilities

- configure and initialize a repository
- tracking files
- stage and commit changes
- ignore certain files and file patterns
- undo mistakes
- browse the history and view changes
- push and pull from remote repositories
- branching and merging

git repository

- local directory that is not under version control, and turn it into a git repository
- clone an existing Git repository from elsewhere

git repository init

- `git init`
- `git clone <URL> <DIR>`

Git and GitHub

Getting started with Git

git repository init

After **init** nothing in the project is **tracked** yet.
Need to begin **tracking** those files and do an initial commit.

specify the files you want to track

```
git add <FILE(S)>
git commit -m "<MESSAGE>"
```

Git repository with tracked files and an initial commit

Git and GitHub

Getting started with Git

git clone repository

Every version of every file for the history of the project is **pulled down** by default when you run `git clone`

git repository init

Each file in your working directory can be in one of two states

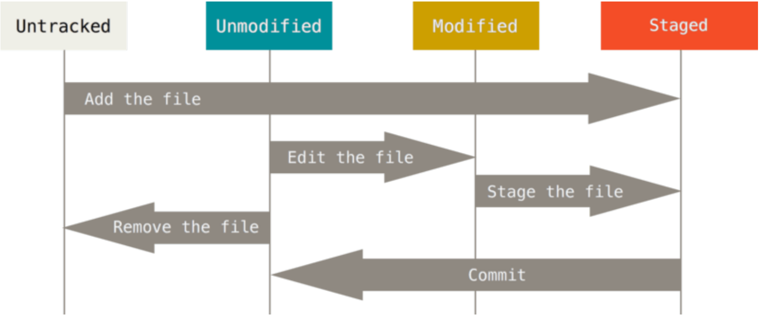
track files

- tracked
- untracked

Tracked files are files that were in the last snapshot; they can be unmodified, modified, *or* staged

Git and GitHub

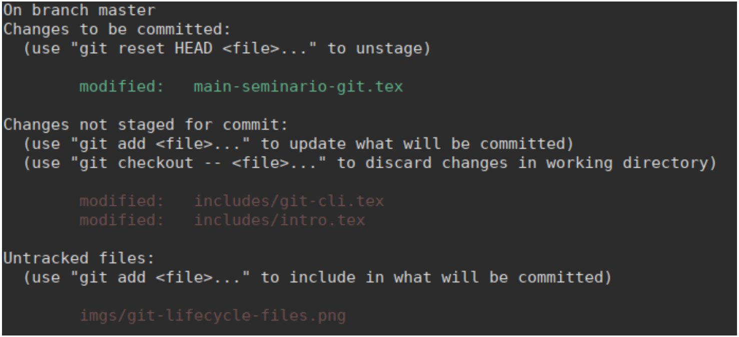
track files



Git and GitHub

status files

To determine which files are in which state: the git status command



```
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   main-seminario-git.tex

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   includes/git-cli.tex
        modified:   includes/intro.tex

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

        imgs/git-lifecycle-files.png
```

Git and GitHub

adding files

git add

In order to begin tracking a new file, you use the **command git add**

git add

file is now **tracked** and **staged** to be **committed**

The git add command takes a path name *for either a file or a directory*

Git and GitHub

adding files

git add

File that is tracked has been modified in the working directory but not yet staged

git add

To stage a modified tracked file, you have to run the **git add command** again.

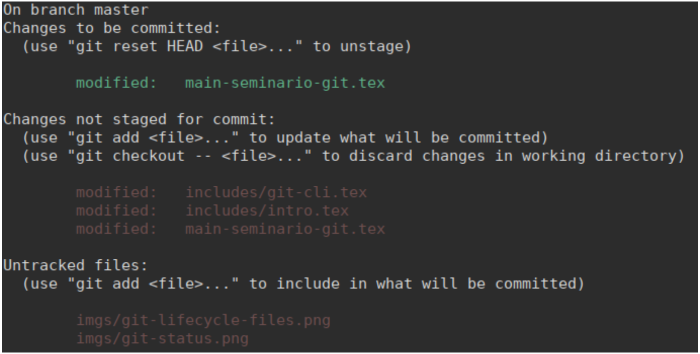
After git add, the files are staged and will go into your next commit

Git and GitHub

adding files

git add

If you modify a file after you run git add, you have to run git add again to stage the latest version of the file



```
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   main-seminario-git.tex

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   includes/git-cli.tex
        modified:   includes/intro.tex
        modified:   main-seminario-git.tex

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

        imgs/git-lifecycle-files.png
        imgs/git-status.png
```

Git and GitHub

ignoring files

.gitignore file

If you'll have a class of files that you don't want to track

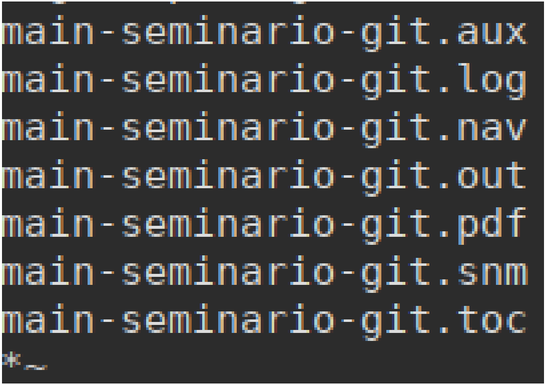
.gitignore file

you can create a file listing patterns to match them named **.gitignore**.

Git and GitHub

ignoring files

```
more .gitignore
```



```
main-seminario-git.aux
main-seminario-git.log
main-seminario-git.nav
main-seminario-git.out
main-seminario-git.pdf
main-seminario-git.snm
main-seminario-git.toc
*~
```

Git and GitHub

viewing files

git diff

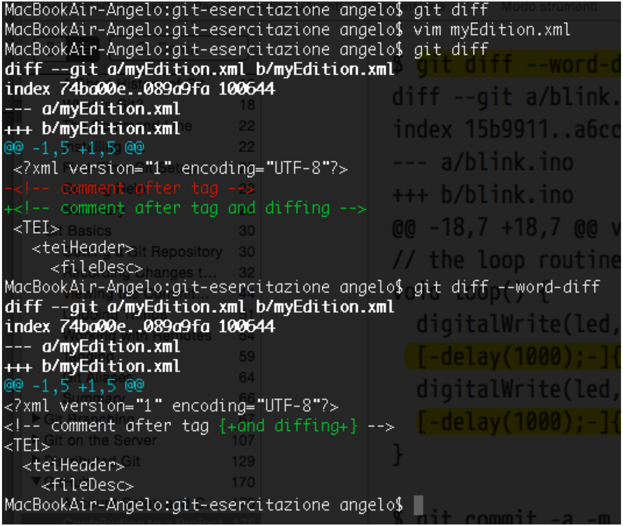
know exactly what you changed, not just which files were changed
by using the **git diff command**

git diff

- what have you changed but not yet staged (`git diff`)
- what have you staged that you are about to commit (`git diff --staged`)
- showing word-based changes by **using –word-diff** option

Git and GitHub

viewing files



```
MacBookAir-Angelo:git-esercitazione angelo$ git diff
MacBookAir-Angelo:git-esercitazione angelo$ vim myEdition.xml
MacBookAir-Angelo:git-esercitazione angelo$ git diff
diff --git a/myEdition.xml b/myEdition.xml
index 74ba00e..089a9fa 100644
--- a/myEdition.xml
+++ b/myEdition.xml
@@ -1,5 +1,5 @@
 <?xml version="1" encoding="UTF-8"?>
-<!-- comment after tag -->
+<!-- comment after tag and diffing -->
 <TEI>
   <teiHeader>
     <fileDesc>
MacBookAir-Angelo:git-esercitazione angelo$ git diff --word-diff
diff --git a/myEdition.xml b/myEdition.xml
index 74ba00e..089a9fa 100644
--- a/myEdition.xml
+++ b/myEdition.xml
@@ -1,5 +1,5 @@
<?xml version="1" encoding="UTF-8"?>
<!-- comment after tag {+and diffing+} -->
<TEI>
  <teiHeader>
    <fileDesc>
MacBookAir-Angelo:git-esercitazione angelo$
```

Git and GitHub

committing files

git commit

Any files you have created or modified that you haven't run git add on since you edited them — won't go into the commit.

git commit

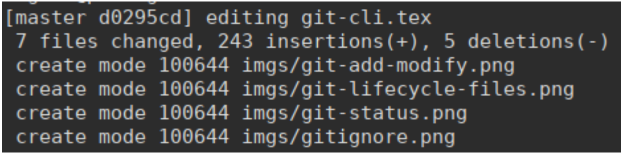
- the simplest way to commit is to type (`git commit`)
- type your commit message inline (`git commit -m "message"`)

Git and GitHub

committing files

git commit

Every time you perform a commit, you're recording a snapshot of your project that you can revert to or compare to later.



```
[master d0295cd] editing git-cli.tex
 7 files changed, 243 insertions(+), 5 deletions(-)
 create mode 100644 imgs/git-add-modify.png
 create mode 100644 imgs/git-lifecycle-files.png
 create mode 100644 imgs/git-status.png
 create mode 100644 imgs/gitignore.png
```

Git and GitHub

removing files

git rm

To remove a file from git, you have to remove it from your tracked files

git rm

```
git rm <FILE>
git rm -f <FILE>
git rm --cached <FILE>
```

Git and GitHub

moving files

git mv

If you rename a file in Git, no metadata is stored in Git that tells it you renamed the file

git mv

- `git mv <FILE-FROM> <FILE-TO>`

Git and GitHub

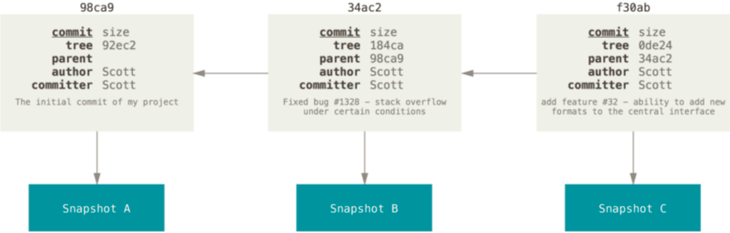
moving files

git mv

- `mv <FILE-FROM> <FILE-TO>`
- `git rm <FILE-FROM>`
- `git add <FILE-TO>`

Git and GitHub

History of commits



98ca9

```
   commit size
     tree 92ec2
   parent
   author Scott
committer Scott
```

The initial commit of my project

34ac2

```
   commit size
     tree 184ca
   parent 98ca9
   author Scott
committer Scott
```

Fixed bug #1328 - stack overflow under certain conditions

f30ab

```
   commit size
     tree 0de24
   parent 34ac2
   author Scott
committer Scott
```

add feature #32 - ability to add new formats to the central interface

Snapshot A

Snapshot B

Snapshot C

git log

git log lists the commits made in that repository in reverse chronological order, each commit with its checksum hash string, author's name and email, date, the commit message.

git log

- `git log <options>`

Git and GitHub

History of commits

git log

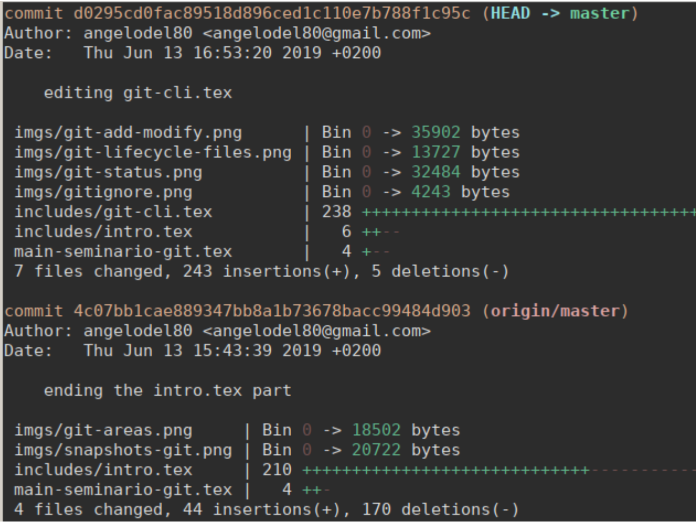
if you want to see some abbreviated stats for each commit, you can use **the –stat option**

git log

- `git log --stat`

Git and GitHub

History of commits



```
commit d0295cd0fac89518d896ced1c110e7b788f1c95c (HEAD -> master)
Author: angelodel80 <angelodel80@gmail.com>
Date:   Thu Jun 13 16:53:20 2019 +0200

    editing git-cli.tex

 imgs/git-add-modify.png       | Bin 0 -> 35902 bytes
 imgs/git-lifecycle-files.png | Bin 0 -> 13727 bytes
 imgs/git-status.png           | Bin 0 -> 32484 bytes
 imgs/gitignore.png            | Bin 0 -> 4243 bytes
 includes/git-cli.tex          | 238 ++++++++++++++++++++++++++++++++++
 includes/intro.tex            |   6 ++--
 main-seminario-git.tex        |   4 +--
 7 files changed, 243 insertions(+), 5 deletions(-)

commit 4c07bb1cae889347bb8a1b73678bacc99484d903 (origin/master)
Author: angelodel80 <angelodel80@gmail.com>
Date:   Thu Jun 13 15:43:39 2019 +0200

    ending the intro.tex part

 imgs/git-areas.png      | Bin 0 -> 18502 bytes
 imgs/snapshots-git.png | Bin 0 -> 20722 bytes
 includes/intro.tex      | 210 +++++++++++++++++++++++++++++-----------
 main-seminario-git.tex |   4 ++-
 4 files changed, 44 insertions(+), 170 deletions(-)
```

Git and GitHub

History of commits

git log options

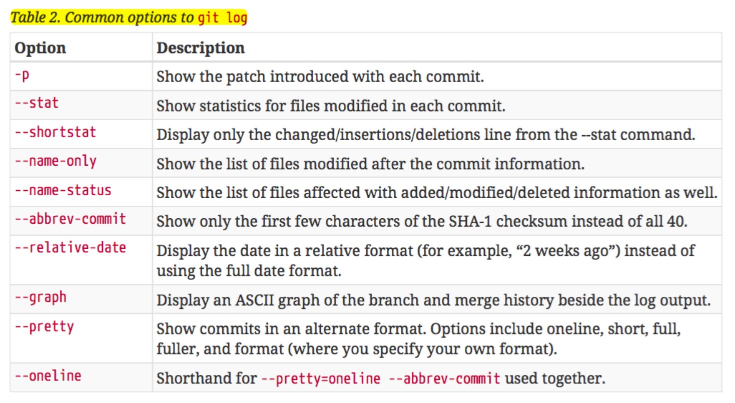


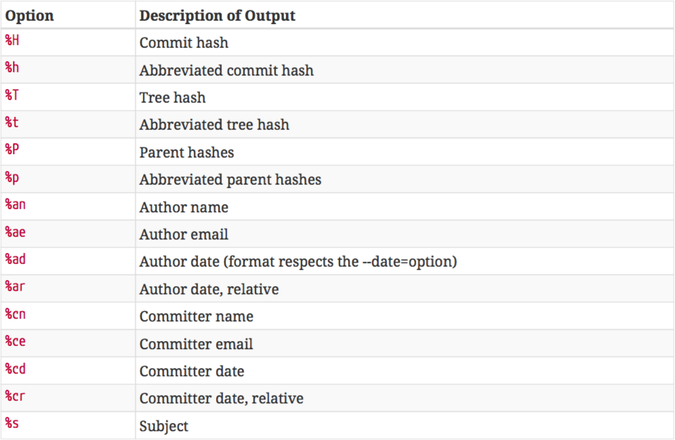
Table 2. Common options to `git log`

Option	Description
`-p`	Show the patch introduced with each commit.
`--stat`	Show statistics for files modified in each commit.
`--shortstat`	Display only the changed/insertions/deletions line from the --stat command.
`--name-only`	Show the list of files modified after the commit information.
`--name-status`	Show the list of files affected with added/modified/deleted information as well.
`--abbrev-commit`	Show only the first few characters of the SHA-1 checksum instead of all 40.
`--relative-date`	Display the date in a relative format (for example, “2 weeks ago”) instead of using the full date format.
`--graph`	Display an ASCII graph of the branch and merge history beside the log output.
`--pretty`	Show commits in an alternate format. Options include oneline, short, full, fuller, and format (where you specify your own format).
`--oneline`	Shorthand for `--pretty=oneline --abbrev-commit` used together.

Git and GitHub

git log –pretty

Table 1. Useful options for `git log --pretty=format`



Option	Description of Output
`%H`	Commit hash
`%h`	Abbreviated commit hash
`%T`	Tree hash
`%t`	Abbreviated tree hash
`%P`	Parent hashes
`%p`	Abbreviated parent hashes
`%an`	Author name
`%ae`	Author email
`%ad`	Author date (format respects the --date=option)
`%ar`	Author date, relative
`%cn`	Committer name
`%ce`	Committer email
`%cd`	Committer date
`%cr`	Committer date, relative
`%s`	Subject

Git and GitHub

History of commits

git log limit options

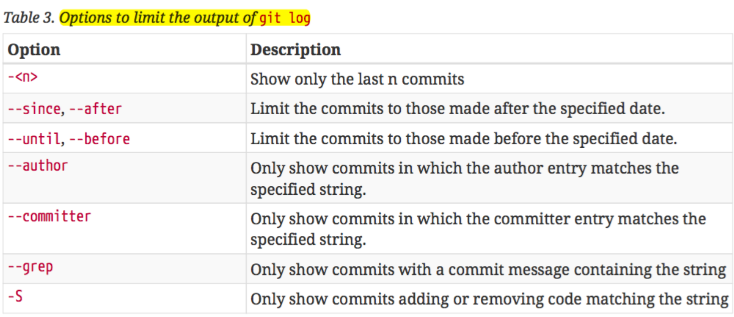


Table 3. Options to limit the output of `git log`

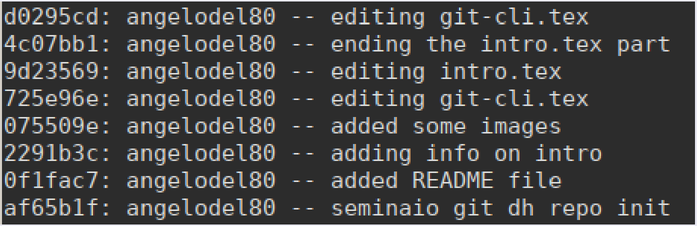
Option	Description
`-<n>`	Show only the last n commits
`--since`, `--after`	Limit the commits to those made after the specified date.
`--until`, `--before`	Limit the commits to those made before the specified date.
`--author`	Only show commits in which the author entry matches the specified string.
`--committer`	Only show commits in which the committer entry matches the specified string.
`--grep`	Only show commits with a commit message containing the string
`-S`	Only show commits adding or removing code matching the string

Git and GitHub

History of commits

```
git log --pretty="%h:  %an -- %s" --no-merges
```

git log –pretty



```
d0295cd: angelodel80 -- editing git-cli.tex
4c07bb1: angelodel80 -- ending the intro.tex part
9d23569: angelodel80 -- editing intro.tex
725e96e: angelodel80 -- editing git-cli.tex
075509e: angelodel80 -- added some images
2291b3c: angelodel80 -- adding info on intro
0f1fac7: angelodel80 -- added README file
af65b1f: angelodel80 -- seminaio git dh repo init
```

amend option

If you commit too early and possibly forget to add some files, make the additional changes you forgot, stage them, and **commit again using the –amend option**.
You end up with a single commit — the *second commit replaces the first one*.

git log

- `git commit --amend [-m "MESSAGE"]`

unstage and discard changes

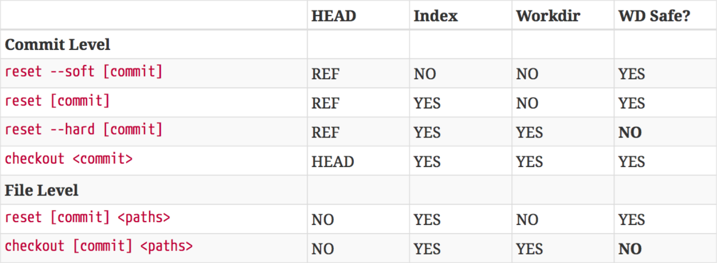
How can you unstage a file or revert it back to what it looked like when you last committed.

git reset and checkout

- `git reset HEAD <FILE>` (unstage file)
- `git checkout -- <FILE>` (discard changes)

Git and GitHub

Undoing things



	HEAD	**Index**	**Workdir**	**WD Safe?**
Commit Level				
`reset --soft [commit]`	REF	NO	NO	YES
`reset [commit]`	REF	YES	NO	YES
`reset --hard [commit]`	REF	YES	YES	**NO**
`checkout <commit>`	HEAD	YES	YES	YES
File Level				
`reset [commit] <paths>`	NO	YES	NO	YES
`checkout [commit] <paths>`	NO	YES	YES	**NO**

Remote Repositories

Remote repositories are versions of your project that are hosted on the Internet

Remote Repositories

Collaborating with others involves managing remote repositories.
This entails **pushing** and **pulling** data to and from remote repositories when you need to share data.

capabilities

- add remote repositories
- remove remotes
- manage various remote branches
- define them as being tracked or not
- pushing, pulling and fetching operations

remote repositories

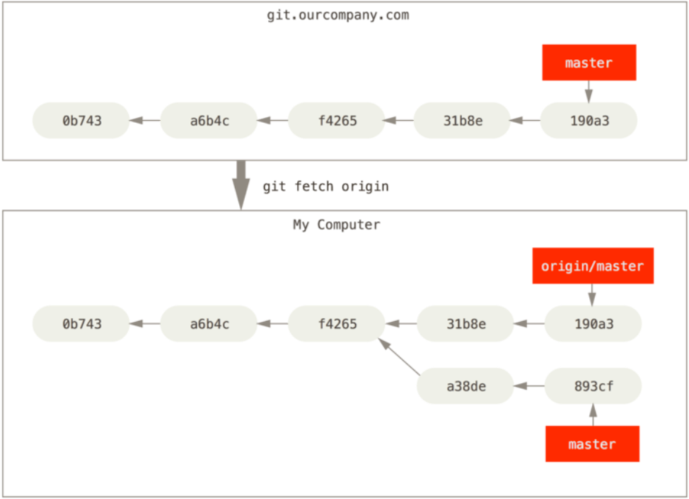
To see which remote servers you have configured, you can run the **git remote command**

git remote

- `git remote`
- `git remote -v`

Git Branching and Merging

Remote Branches



remote repositories

To add a new remote Git repository as a shortname you can reference easily, run **git remote add shortname url**:

git remote

```
git remote add upstream-edition
https://github.com/angelodel80/myEdition
```

If you clone a repository, the command automatically adds that remote repository under the name "origin"

Git and GitHub

Working with Remotes

remote repositories

to get data from your remote projects, you can run the **git fetch command**.
It's important to note that the git fetch command only downloads the data to your local repository — **it doesn't automatically merge it** with any of your work or modify what you're currently working on.

git remote

- `git fetch <remote>`

remote repositories

When you have your project at a point that you want to share, you have to **push it upstream**. This pushes any commits you've done back up to the server if you have write access and if nobody has pushed in the meantime.

git remote

- `git push <remote> <branch>`

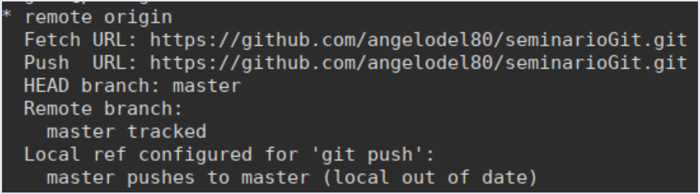
Git and GitHub

Working with Remotes

git remote

- `git remote show <remote>`

remote repositories



```
* remote origin
  Fetch URL: https://github.com/angelodel80/seminarioGit.git
  Push  URL: https://github.com/angelodel80/seminarioGit.git
  HEAD branch: master
  Remote branch:
    master tracked
  Local ref configured for 'git push':
    master pushes to master (local out of date)
```

Git and GitHub

Working with Remotes

remote repositories

You can run **git remote rename** to change a remote's shortname, if you want to remove a remote repository you can either use **git remote remove** command or **git remote rm** command.

git remote

- `git remote rename original upstream-edition`
- `git remote remove upstream-edition`

Tagging

tag specific points

Git has the ability to **tag specific points** in a *repository's history* as being important, e.g. mark release points. Git supports two types of tags: lightweight and annotated.

git tag

- `git tag [-l] [--list] <PATTERN>` (list tags)
- `git tag -a <TAG-NAME> -m "MESSAGGIO"` (create an annotated tag)
- `git show <TAG-NAME>` (show the tag data)
- `git push <REMOTE> <TAG-NAME>` (push tag)
- `git tag -d <TAG-NAME>` (delate locally)
- `git push <REMOTE> --delete <TAG-NAME>` (delate remotelly)

tag specific points

If you want to view the versions of files a tag is pointing to, you can do a git checkout of that tag.
This puts your repository in “detached HEAD” state, which has some ill side effects

git tag

- `git checkout <TAG-NAME>` (View the files in tag version)

Progress status

1 VCS Introduction

2 GitHub host platform

3 CLI - shell concepts

4 Git environment by command line interface

5 Git basic branching and merging

6 Conclusions and References

Git Branching and Merging

Branch

What branching means

Branching means you **diverge from the main line of development** and continue to do work without messing with that main line

Git Killer Feature

The way Git branches is incredibly lightweight

Branching often

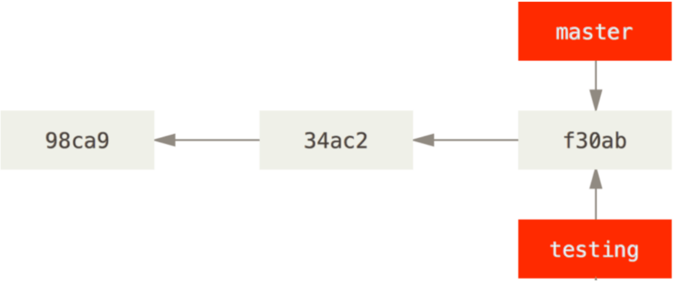
Git encourages workflows that branch and merge often, even multiple times in a day

A Git Branch is

A branch in Git is simply a lightweight movable pointer to one commits

Git Branching and Merging

Branch



Git Branching and Merging

Branch

Branching often

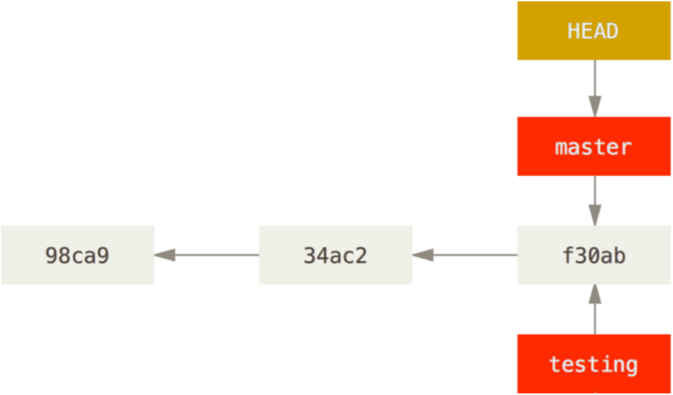
create a new branch creates a new pointer

new branch

```
git branch <NAME NEW BRANCH>
```

Git Branching and Merging

Branch



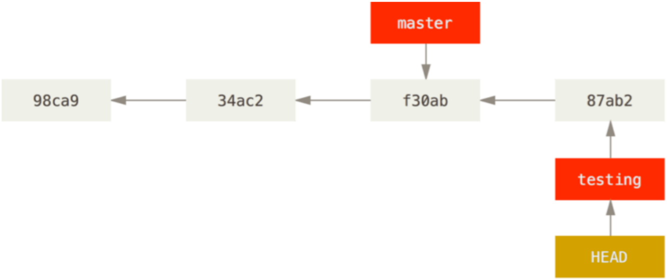
new branch

The git branch command only created a new branch.
To switch to an existing branch, you run the `git checkout` command

checkout

```
git checkout <NAME BRANCH>
```

This moves the special pointer HEAD to point to the new branch



files in your working directory will change and HEAD pointer moves.

Git Branching and Merging

Branch

branches

you can **switch back and forth** between the branches and **merge them together** when you're ready

checkout

If you run git `log --oneline --decorate --graph --all` it will print out the history of your commits

Git Branching and Merging

Branch

branches

To create a new branch and switch to it at the same time, you can run the git checkout command with the **-b switch**.

create branch and checkout

```
git checkout -b <NAME NEW BRANCH>
```

Git Branching and Merging

Branch

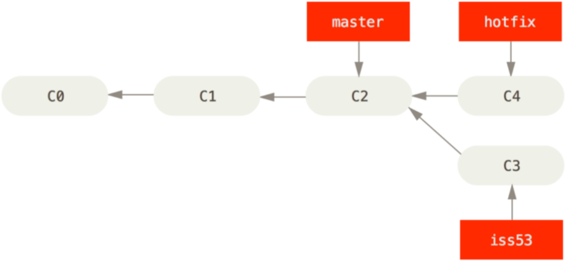
checkout

when you switch branches, Git resets your working directory to look like it did the last time you committed on that branch.

if your working directory or staging area has uncommitted changes that conflict with the branch you're checking out, Git won't let you switch branches

Git Branching and Merging

Merge



merge and delete branch

finally you merge the branch back into your master branch with the `git merge` command and delete the merged branch with the `git branch -d` option.

Git Branching and Merging

Merge and Three-Way Merge

Fast-Forward Merge

When you try to merge one commit with a commit that can be reached by following the **first commit's history**, Git simplifies things by moving the pointer forward because there is no divergent work to merge together — this is called a **fast-forward**.

Git Branching and Merging

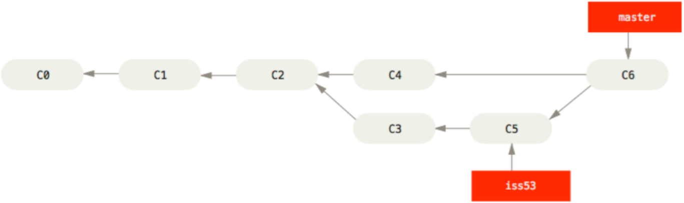
Merge and Three-Way Merge

Three-Way Merge

If your development history has diverged from some older point, the commit on the branch you're on **isn't a direct ancestor of the branch** you're merging in, Git creates a new snapshot that results from a **three-way merge** and automatically creates a new commit that points to it

Git Branching and Merging

Merge



merge and delete branch

using the two snapshots pointed to by the branch tips and the common ancestor of the two.

Git Branching and Merging

Merge

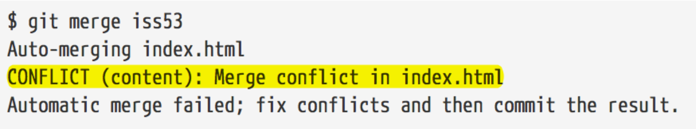
Merge Conflict

If the same part of the same file has been differently changed in the two branches you're merging, Git won't be able to merge them cleanly

GIT paused the process while you resolve the conflict.

Git Branching and Merging

Merge



```
$ git merge iss53
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
Automatic merge failed; fix conflicts and then commit the result.
```

finalize the merge

After you've resolved each of these sections in each conflicted file, **run git add on each file** to mark it as resolve and then type git commit to finalize the merge commit

Git Branching and Merging

branch command

list branches

If you run `git branch` with no arguments, you get a simple listing of your current branches. The branch that HEAD points to is signed by a star (*).

- `git branch`
- `git branch -v` (see the last commit)

Git Branching and Merging

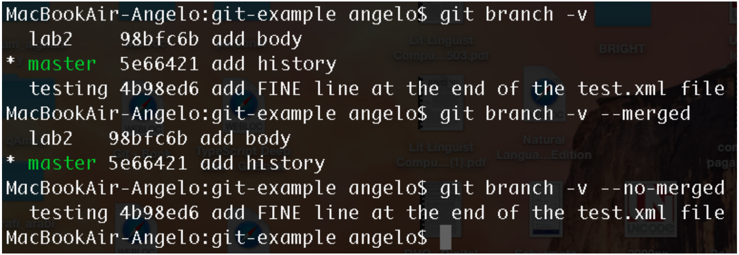
branch command

list branches

The useful **–merged and –no-merged options** can filter this list to branches that you have or have not yet merged into the branch you're currently on.

Git Branching and Merging

Branch



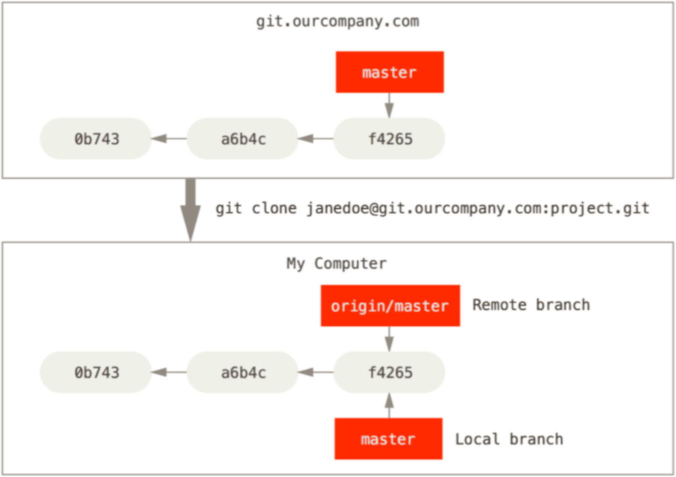
```
MacBookAir-Angelo:git-example angelo$ git branch -v
  lab2    98bfc6b add body
* master  5e66421 add history
  testing 4b98ed6 add FINE line at the end of the test.xml file
MacBookAir-Angelo:git-example angelo$ git branch -v --merged
  lab2   98bfc6b add body
* master 5e66421 add history
MacBookAir-Angelo:git-example angelo$ git branch -v --no-merged
  testing 4b98ed6 add FINE line at the end of the test.xml file
MacBookAir-Angelo:git-example angelo$
```

Some common workflows

- Long-Running Branches
- Topic Branches
- ...

Git Branching and Merging

Remote Branches



```
git clone janedoe@git.ourcompany.com:project.git
```

Git Branching and Merging

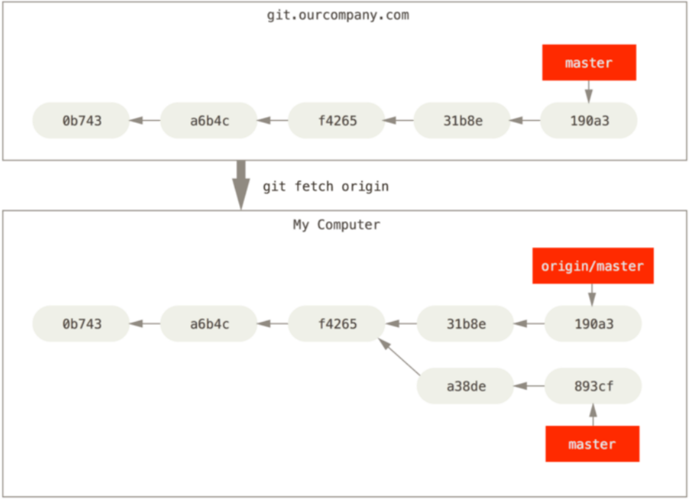
branch

remote branches

Remote-tracking branches are references to the state of remote branches

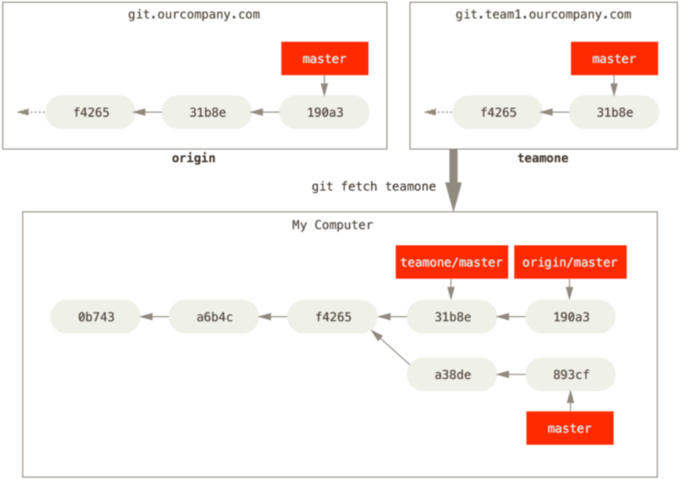
remote branches

Remote-tracking branch names take the form `<remote>/<branch>`.


git.ourcompany.com
master
0b743
a6b4c
f4265
31b8e
190a3
git fetch origin
My Computer
origin/master
0b743
a6b4c
f4265
31b8e
190a3
a38de
893cf
master

Git Branching and Merging

Remote Branches



Git Branching and Merging

branch

push to remote branches

When you want to share a branch, you need to **push it up to a remote branch** to which you have write access

push to remote branches

Local branches aren't automatically synchronized to the remotes

Git Branching and Merging

branch

push remote new branches

You can push new branches up the same way you pushed your default branch

push command

```
git push <remote> <branch>
git push <remote> <local-branch:remote-branch>
```

Git Branching and Merging

branch

remote branches

It's important to note that when you do a fetch that brings down new remote-tracking branches, you don't automatically have local, editable copies of them: **you have only a remote reference that you cannot modify**.

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

Git Branching and Merging

branch

remote branches

Checking out a local branch from a remote-tracking branch automatically creates what is called a **tracking branch**, remote branch is called **upstream branch**.

remote branches

If you're on a tracking branch and type `git pull`, Git automatically knows which server to fetch from and which branch to merge in.

Git Branching and Merging

branch

remote branches

If the branch name you're trying to checkout **(a) doesn't exist** and (b) **exactly matches a name on only one remote**, *Git will create a tracking branch*.

git command

```
git checkout <NAME BRANCH>
```

remote branches

If you already *have a local branch* and want to **set it to a remote branch** you just pulled down, or want to change the upstream branch you're tracking, you can `use the -u or --set-upstream-to option` to git branch to explicitly set it at any time.

git command

```
git branch -u <REMOTE>/<NAME BRANCH>
```

Git Branching and Merging

branch

remote branches

If you want to see what tracking branches you have set up, **you can use the -vv option to git branch**.
Further information such as what each branch is tracking and if your local branch is **ahead, behind or both**.

git command

```
git branch -vv
```

Git Branching and Merging

branch

git pull

`git pull` will look up what server and branch your current branch is tracking, **fetch from that server and then try to merge in that remote branch**.

git pull

Generally it's **better to simply use the fetch and merge commands explicitly** as the magic of git pull can often be confusing.

Git Branching and Merging

branch

delete branch

You can **delete a remote branch** removing the pointer from the server until a **garbage collection** runs.

delete branch

```
git push <REMOTE> --delete <NAME BRANCH>
```

Git Branching and Merging

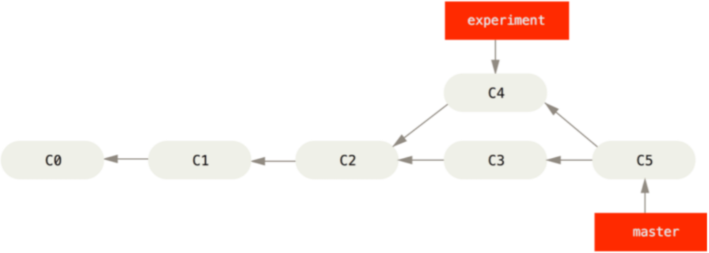
branch

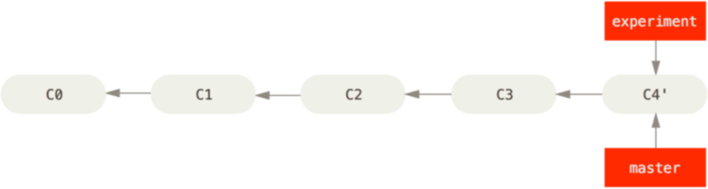
merge vs rebase

In Git, there are **two main ways to integrate changes** from one branch into another: the **merge** and the **rebase**.

Git Branching and Merging

Merge vs Rebase




experiment
C0
C1
C2
C3
C4'
master

merge vs rebase

Beside to the merge command there is another command in git called **rebase**: you can take all the changes that were committed on one branch and **replay them on a different branch**.

git rebasing

Rebasing works by going to the common ancestor of the two branches, getting the diff introduced by each commit of the branch you're on, saving those diffs to temporary files, **resetting the current branch to the same commit as the branch you are rebasing onto** and applying each change in turn.

rebase git command

```
git rebase <ONTO-BRANCH>
```

rebase git command

At this point, you can go back to the target branch and do a **fast-forward merge**

git rebasing

The snapshot pointed to by the **final commit** whether it's the last of the rebased commits for a rebase or the final merge commit after a merge, is **the same snapshot**. **It's only the history that is different**.

Git Branching and Merging

Advanced rebasing

rebase git command

- `git rebase --onto master <branchA> <branchB>`
- `git rebase master <NAME BRANCH>`

git rebasing

You can rebase a branch onto another branch without having to check it out first by running `git rebase <basebranch> <topicbranch>`

Git Branching and Merging

rebasing drawbacks

rebase issues

rebasing work that you've made public can cause problems.

git rebasing

When you rebase stuff, you're **abandoning existing commits and creating new ones** that are *similar but different*.

Git Branching and Merging

rebasing drawbacks

git rebasing

You can run `git pull --rebase` instead of a normal git pull. Or you could do it manually with a `git fetch` followed by a `git rebase teamone/master` in this case (**based on pathces and patch-ids**).

Git Branching and Merging

Merge vs Rebase

merging vs rebasing: matter of history

- repository's commit history is a record of what actually happened
- the commit history is the story of how your project was made

merging vs rebasing

Git is a powerful tool, and allows you to do many things to and with your history, but every team and every project is different.

Progress status

1 VCS Introduction

2 GitHub host platform

3 CLI - shell concepts

4 Git environment by command line interface

5 Git basic branching and merging

6 **Conclusions and References**

GIT

- basic understanding of what VCS and git are
- working version of Git on your system
- basic configuration set up

GIT and GITHUB

Conclusions

GIT basic command line tools

- all the basic local Git operations
- creating or cloning a repository
- making, staging and committing changes
- viewing the history of the changes
- branching model

GIT remote tool

- get a remote git repository up and running
- collaborate with others or share your work
- contributing to a project
- maintaining your own project
- integrating other users' contributions

GITHUB

- gitHub user
- how to create an account
- manage an organization
- create and push to repositories
- contribute to other people's projects
- accept contributions from others

References

Chacon, S., e B. Straub. 2014. Pro Git. Apress.

References

Pro Git book, written by Scott Chacon and Ben Straub, 2nd Edition (2014).
`https://git-scm.com/book/it/v2`