Introduction to Git

Drills for the Course

References

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

Git Flow Theory: http://nvie.com/posts/a-successful-git-branching-model/

Git Flow Exercises: http://jeffkreeftmeijer.com/2010/why-arent-you-using-git-flow/

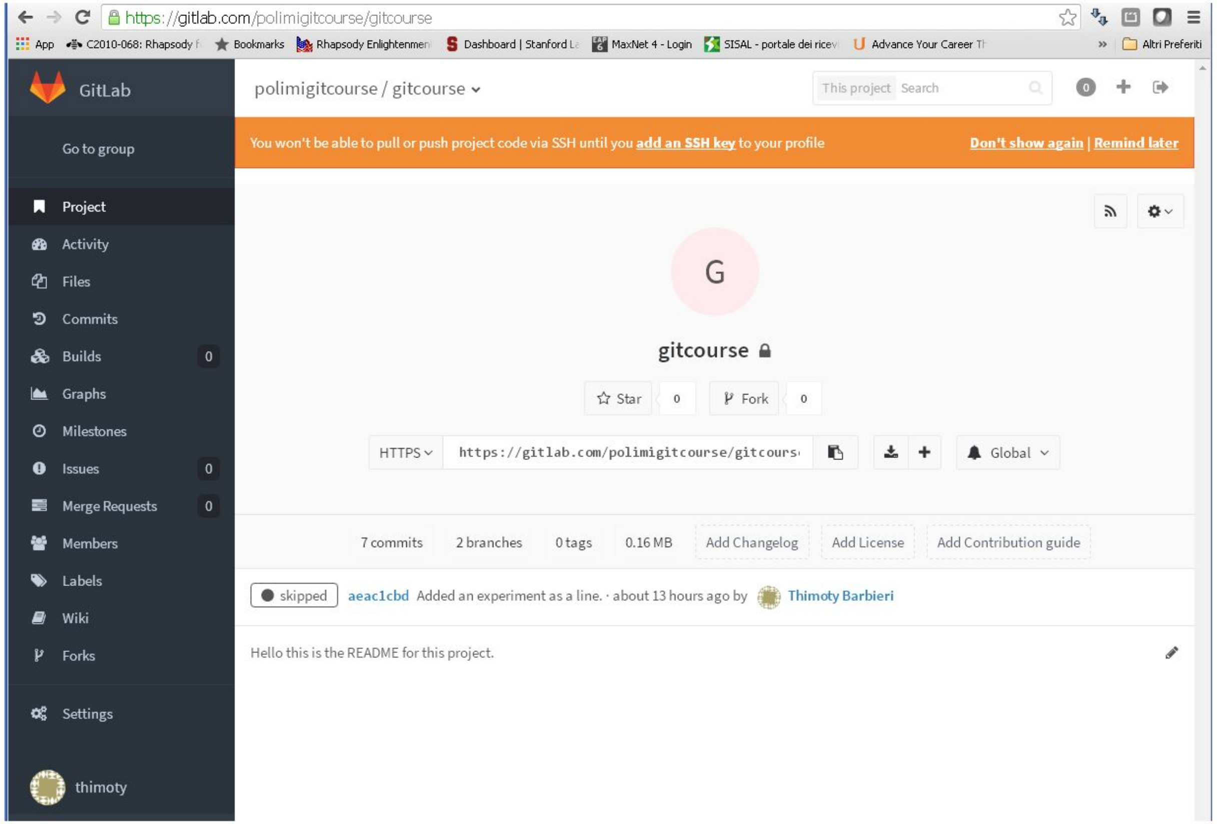
First time Git Setup

- Set up your name globally

```
$ git config --global user.name "John Doe"
$ git config --global user.email johndoe@example.com
```

- Explore Git Bash
- Obtain help
- Explore Git GUI
- Explore EGit

Create a GitLab Account



https://gitlab.com/polimigitcourse/gitcourse

polimigitcourse / gitcourse

You won't be able to pull or push project code via SSH until you **add an SSH key** to your profile **Don't show again** | **Remind later**

gitcourse

https://gitlab.com/polimigitcourse/gitcours

7 commits 2 branches 0 tags 0.16 MB

skipped aeac1cbd Added an experiment as a line. · about 13 hours ago by Thimoty Barbieri

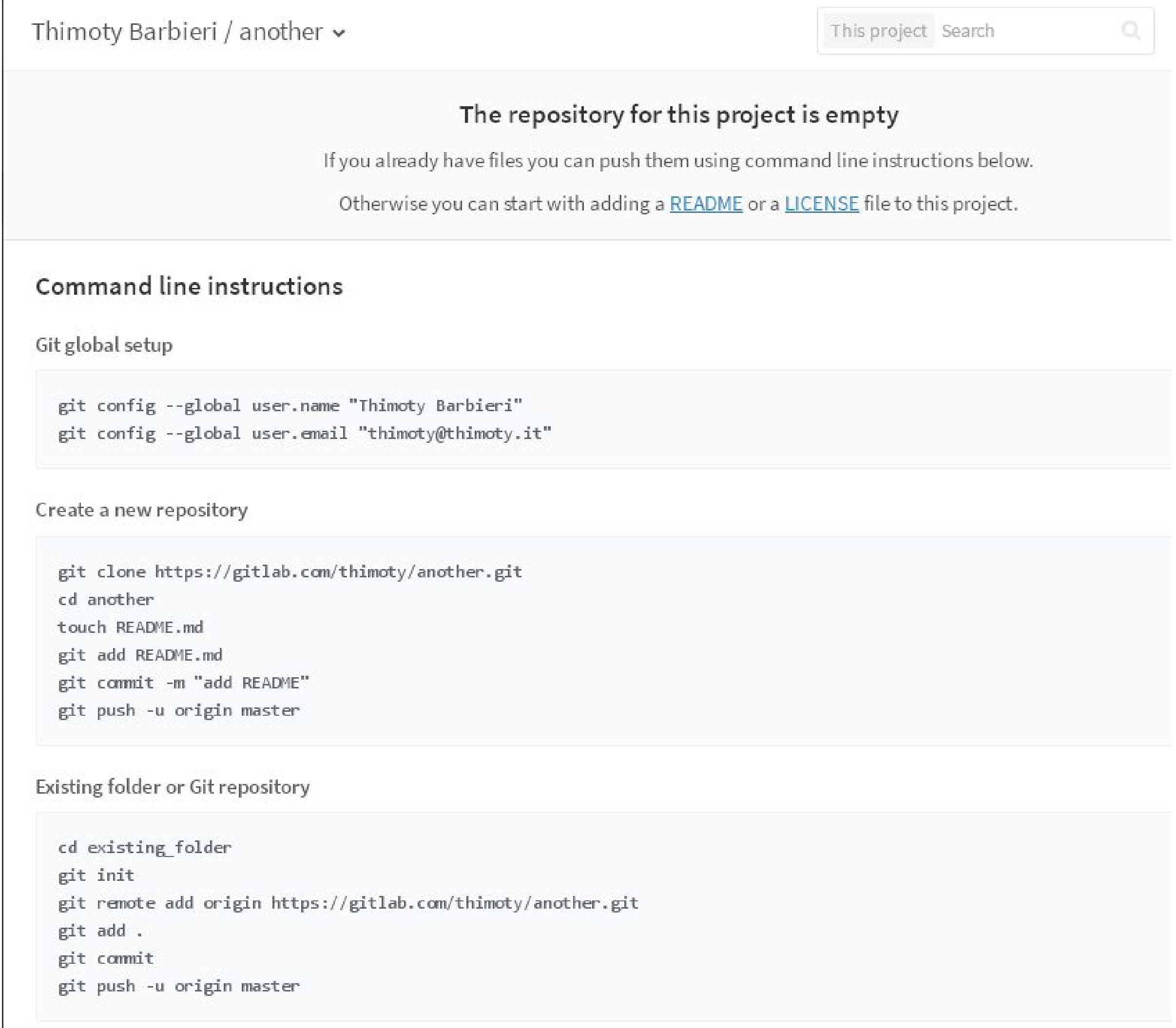
Hello this is the README for this project.

Starting with a Repository (create)

The Repo is attached to a new GitLab Project

Follow the instructions in GitLab to init and create your new GitLab Repo

This will be your REPO1



Thimoty Barbieri / another

This project Search

The repository for this project is empty

If you already have files you can push them using command line instructions below.

Otherwise you can start with adding a README or a LICENSE file to this project.

Command line instructions

Git global setup

```
git config --global user.name "Thimoty Barbieri"
git config --global user.email "thimoty@thimoty.it"
```

Create a new repository

```
git clone https://gitlab.com/thimoty/another.git
cd another
touch README.md
git add README.md
git commit -m "add README"
git push -u origin master
```

Existing folder or Git repository

```
cd existing_folder
git init
git remote add origin https://gitlab.com/thimoty/another.git
git add .
git commit
git push -u origin master
```

Creating and Staging Some Files

Create some file and stage them

Explore the status

Modify a file that you have already staged

Explore diff

Commit the staged modifications

Explore the log, use some query criteria

Push your master to your upstream branch

Attach Gitk and EGit to your local Repo

Start Gitk via Git GUI and attach your local Repo

Browse the current Branch Tree

Start Egit and attach to your local Repo

Look at the staging area, and at the branch tree

Undoing

- Amend a commit
- Unstage a modification
- Discard entirely a modification

Starting with a parallel repository (clone)

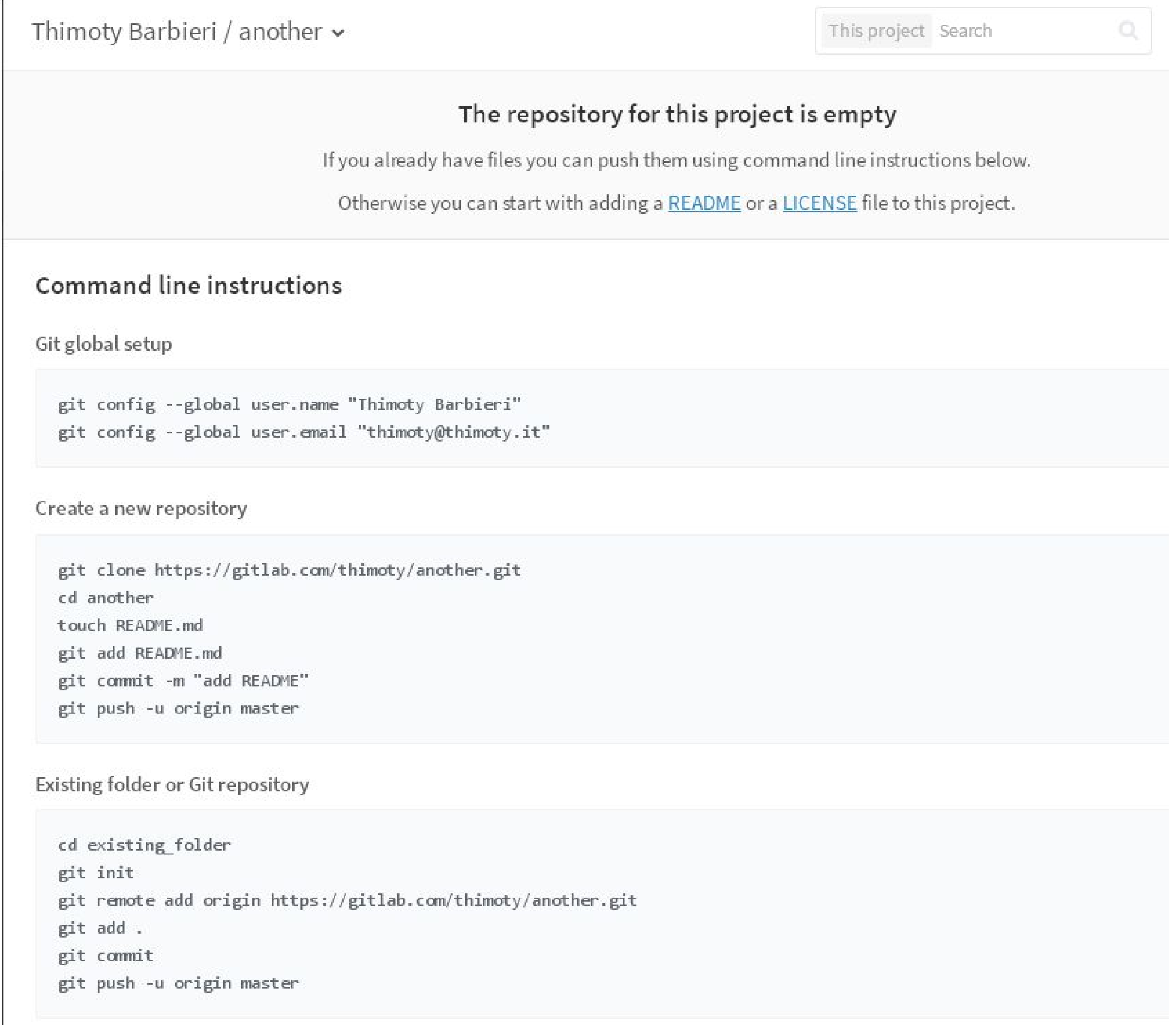
Create a new local repository

Clone from the main repository

Now you can check the log and the branches

You can now use two different local repos to simulate a collaboration

This will be your REPO2



The repository for this project is empty

If you already have files you can push them using command line instructions below.

Otherwise you can start with adding a README or a LICENSE file to this project.

Command line instructions

Git global setup

```
git config --global user.name "Thimoty Barbieri"
git config --global user.email "thimoty@thimoty.it"
```

Create a new repository

```
git clone https://gitlab.com/thimoty/another.git
cd another
touch README.md
git add README.md
git commit -m "add README"
git push -u origin master
```

Existing folder or Git repository

```
cd existing_folder
git init
git remote add origin https://gitlab.com/thimoty/another.git
git add .
git commit
git push -u origin master
```

Using remotes

- Show remotes and tracking status of repository 1
- Push from repository 1
- Inspect the remote (show remote origin)
- Fetch into repository 2
- Inspect your repository 2. What are you missing?
- Pull into repository 2
- Inspect now

Tagging

- Create an annotated tag on repository 1
- Push the tag
- Do an additional commit on repository 1
- Pull the tag in repository 2
- Explore the master branch in repository 2
- Checkout again the tag in repository 1

Basic Branching

- Create a branch from master
- Switch to the new branch
- Modify a file and commit to the branch
- Switch back to master
- Explore the branches in git bash, gitk, egit
- Merge the new branch into master (Check it does FF)
- Explore the branches
- Delete the branch

Recursive Branching

- Create a branch from master
- Switch to the new branch
- Modify a file and commit to the branch
- Switch back to master
- Modify the same file (in a different place)
- Commit to master
- Explore the branches in git bash, gitk, egit
- Merge the new branch into master (Check it does Recursive)
- Explore the branches
- Delete the branch

Merging

- Create a branch from master
- Switch to the new branch
- Modify a file and commit to the branch
- Switch back to master
- Modify the same file (in the same place)
- Commit to master
- Explore the branches in git bash, gitk, egit
- Merge the new branch into master (Check it does Recursive)
- Check the status to verify the conflict
- Solve the Merge, and commit
- Check the status again
- Explore the branches
- Delete the branch

Tracking Branches

- Create a branch on repo1
- Switch the new branch
- Commit something new
- Push the branch upstream
- Go to repo2
- Inspect remote branches
- Fetch and track the remote branch on repo2
- Inspect log of the branch in repo2

Rebase

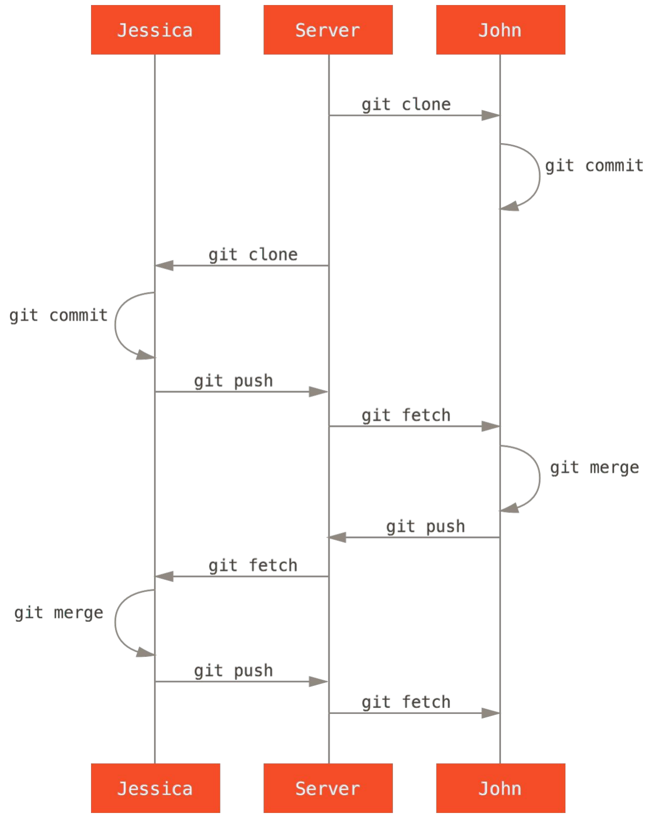
- Go on master branch
- Create a switch to new branch
- Do a commit
- Get back to master
- Do another commit
- Check branches on gitk
- Rebase the new branch on master
- Merge the new branch onto master
- Check branches on gitk
- Take a colleague and explain him/her the GOLDEN RULE OF REBASE

Stashing

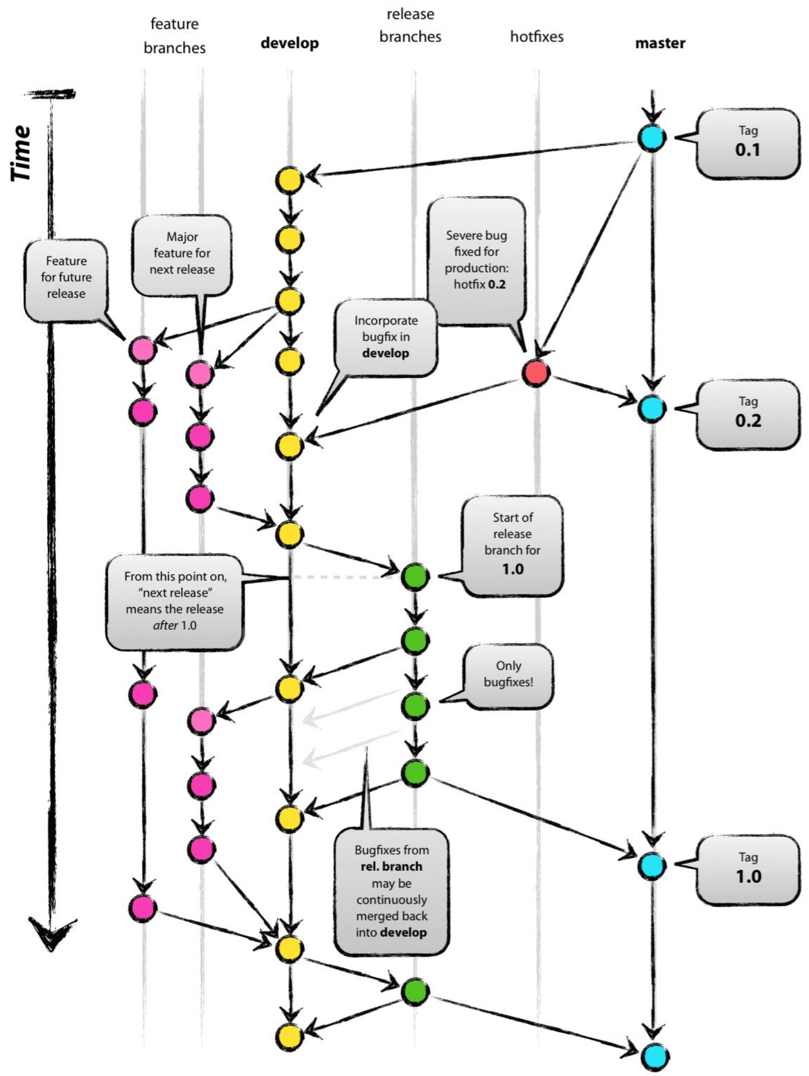
- start some work and leave some files staged and uncommitted
- check status
- do a stash
- check status
- switch branch
- check status
- switch back to original branch
- reapply stash
- finish work and commit
- drop the stash

Private Small Team Flow

- Pair with a colleague
    - Colleague 1 will be John,
    - Colleague 2 will be Jessica
- Define a common repo on GitLab and share it
- Perform at least two rounds of the
    - Private Small Team Flow



Git Flow


feature branches
develop
release branches
hotfixes
master
Time
Tag 0.1
Major feature for next release
Feature for future release
Severe bug fixed for production: hotfix **0.2**
Incorporate bugfix in **develop**
Tag 0.2
Start of release branch for 1.0
From this point on, "next release" means the release *after* 1.0
Only bugfixes!
Bugfixes from **rel. branch** may be continuously merged back into **develop**
Tag 1.0

Init a Repo with Git Flow

- Create develop branch if not existing
- Perform init flow on your existing repo
- Configure init flow

Git Flow: Perform a Bugfix / Feature on Develop

- Use Bugfix start
- Perform a Bugfix and Commit
- Use Bugfix finish
- inspect log and tags on gitk
- push your bugfix upstream

Git Flow: Perform a Release

- use git flow start release for a new version 0.2
- write your release number in one of your files
- commit
- git flow finish release
- inspect branches and tags on gitk
- push your release upstream

Git Flow: Perform a Hotfix

- use git flow start hotfix for a hotfix
- perform the hotfix
- commit everything
- git flow finish hotfix
- inspect branches and tags on gitk. Check develop and master.
- push the hotfix upstream