Git for Version Control

By **Corentin Domken**

Contact:corentin.domken@henallux.be

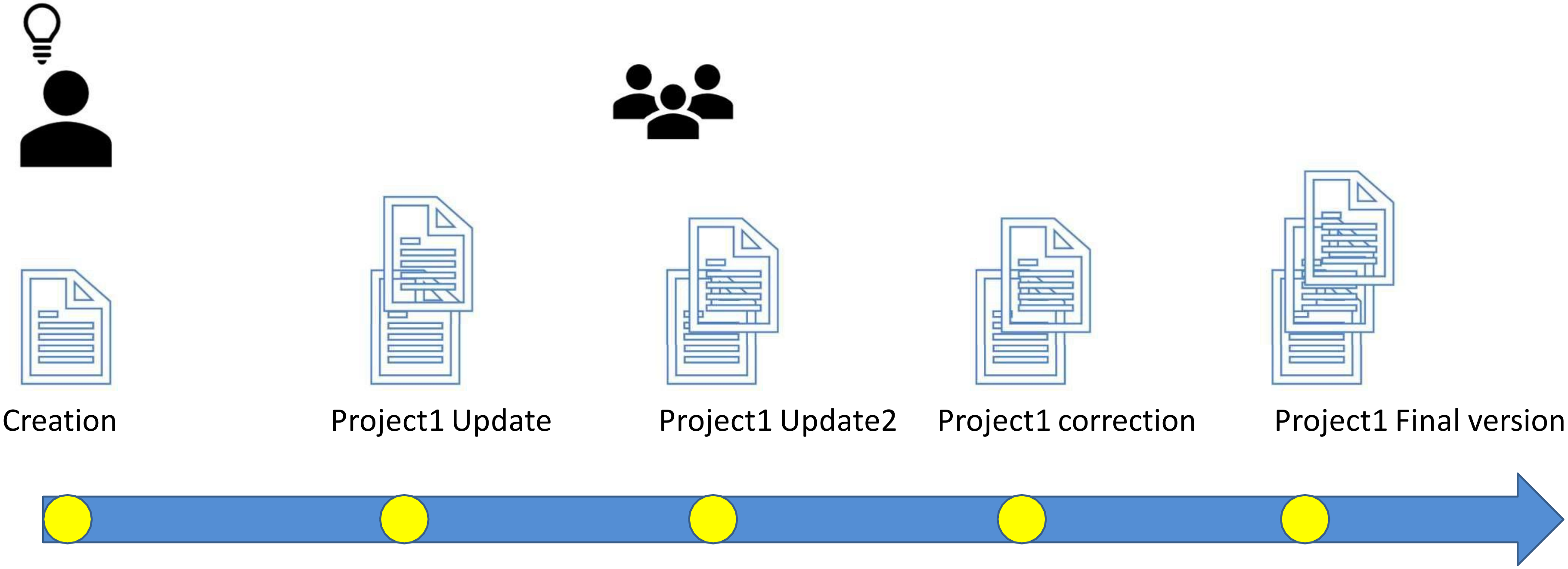
Outline

- Introduction
- The basic Git model (local)
- Branching on git
- Git interaction with a remote repository (remote)

Introduction

classic workflow of a file

Classic workflow of a file



Creation

Project1 Update

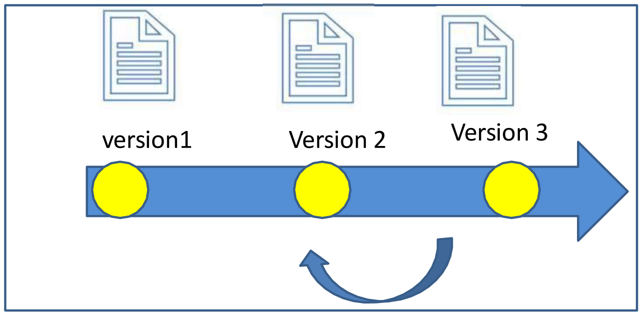
Project1 Update2

Project1 correction

Project1 Final version

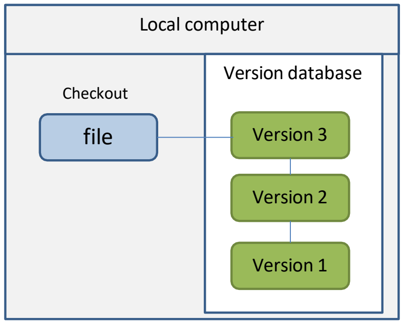
What is Git?

- *Git is a distributed version-control system DVCS*
- *version-control system :* is a system that records changes to a file or set of files over time so that you can recall specific versions later
- It allows you to:
  - Save different states of the project
  - Compare changes over time
  - Revert files to previous state
  - See who modified what? And much more…



Types of version control system

- Local Model



Local computer

Checkout

file

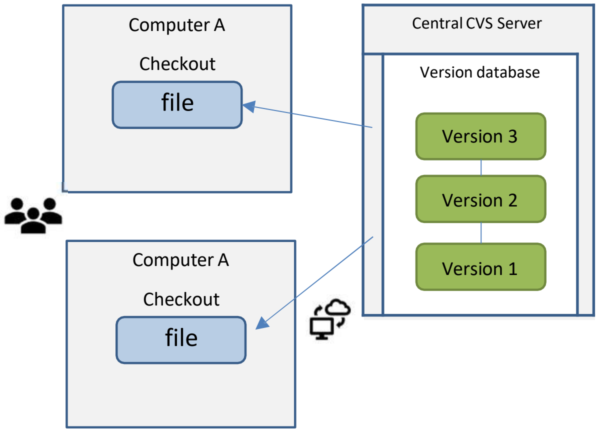
Version database

Version 3

Version 2

Version 1

- Centralized Model



Computer A

Checkout

file

Central CVS Server

Version database

Version 3

Version 2

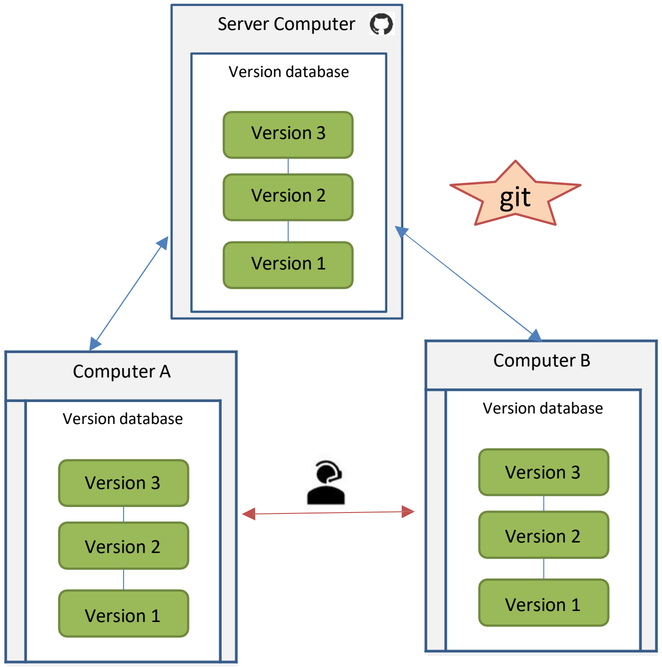
Version 1

Computer A

Checkout

file

- Distributed Model : **DVCS**



Server Computer

Version database

Version 3

Version 2

Version 1

git

Computer A

Version database

Version 3

Version 2

Version 1

Computer B

Version database

Version 3

Version 2

Version 1

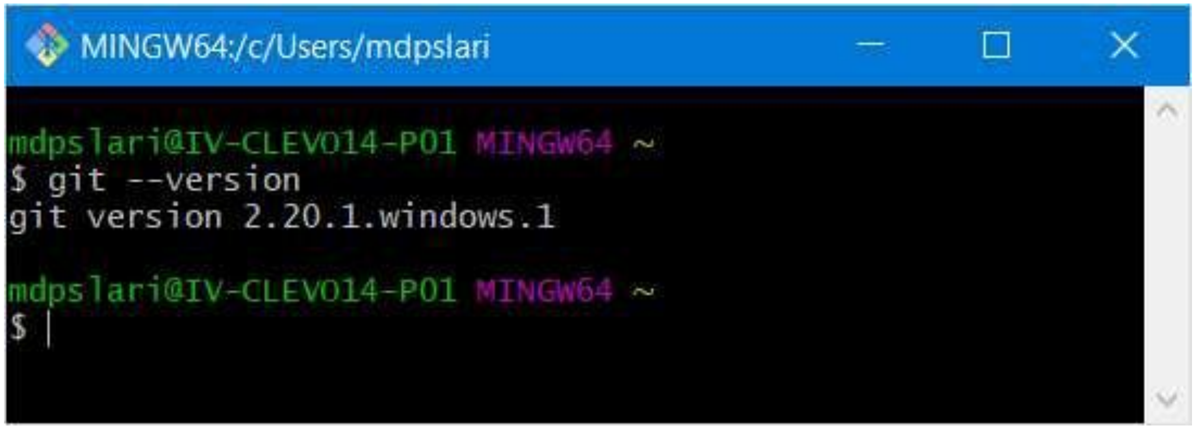
About Git

- From 1991 to 2002, the Linux kernel was developed without using a versioning system.
- From 2002, the community started using BitKeeper, a proprietary DVCS.

- In 2005, after an incident, BitKeeper withdraws the possibility of using its product for free. Linus Torvalds launches the development of Git and after just a few months of development, Git hosts the development of the Linux kernel.

Git install

- Install git (Git website: http://git-scm.com/)
- Once installed check Git version (on the Git Bash) : $git --version



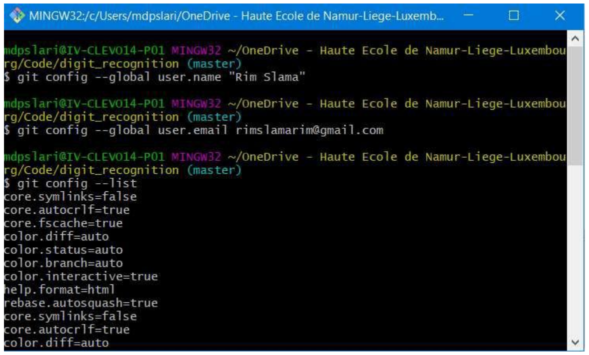
```
mdpslari@IV-CLEV014-P01 MINGW64 ~
$ git --version
git version 2.20.1.windows.1

mdpslari@IV-CLEV014-P01 MINGW64 ~
$ |
```

Git ressources

- At the command line: (where verb = config, add, commit, etc.)

  $ git help <verb>

  $ man git <verb>


MINGW32:/c/Users/mdpslari/OneDrive - Haute Ecole de Namur-Liege-Luxemb...

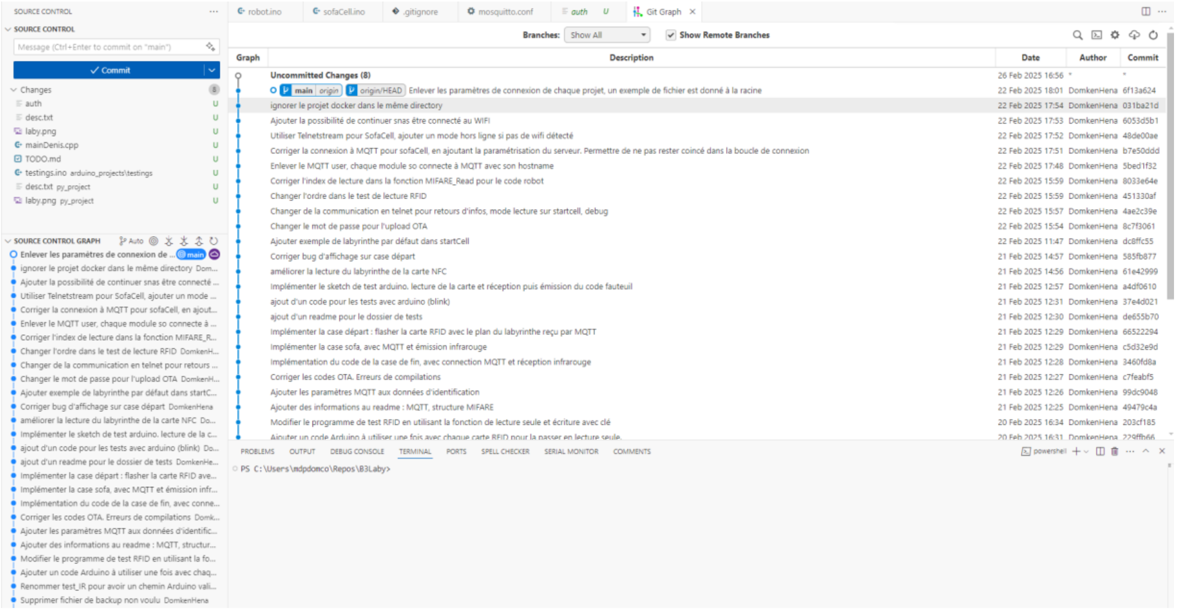
```
mdpslari@IV-CLEV014-P01 MINGW32 ~/OneDrive - Haute Ecole de Namur-Liege-Luxembou
rg/Code/digit_recognition (master)
$ git config --global user.name "Rim Slama"

mdpslari@IV-CLEV014-P01 MINGW32 ~/OneDrive - Haute Ecole de Namur-Liege-Luxembou
rg/Code/digit_recognition (master)
$ git config --global user.email rimslamarim@gmail.com

mdpslari@IV-CLEV014-P01 MINGW32 ~/OneDrive - Haute Ecole de Namur-Liege-Luxembou
rg/Code/digit_recognition (master)
$ git config --list
core.symlinks=false
core.autocrlf=true
core.fscache=true
color.diff=auto
color.status=auto
color.branch=auto
color.interactive=true
help.format=html
rebase.autosquash=true
core.symlinks=false
core.autocrlf=true
color.diff=auto
```

- Free on-line book: http://git-scm.com/book
- Git tutorial: http://schacon.github.com/git/gittutorial.html
- Reference page for Git: http://gitref.org/index.html
- Git for Computer Scientists (http://eagain.net/articles/git-for-computer-scientists/)

Git software

- Git GUIs : git website

- Preferably, use VSCode extension



Enlever les paramètres de connexion de ...
ignorer le projet docker dans le même directory Dom...
Ajouter la possibilité de continuer snas être connecté ...
Utiliser Telnetstream pour SofaCell, ajouter un mode ...
Corriger la connexion à MQTT pour sofaCell, en ajout...
Enlever le MQTT user, chaque module so connecte à ...
Corriger l'index de lecture dans la fonction MIFARE_R...
Changer l'ordre dans le test de lecture RFID DomkenH...
Changer de la communication en telnet pour retours ...
Changer le mot de passe pour l'upload OTA DomkenH...
Ajouter exemple de labyrinthe par défaut dans startC...
Corriger bug d'affichage sur case départ DomkenHena
améliorer la lecture du labyrinthe de la carte NFC Do...
Implémenter le sketch de test arduino. lecture de la c...
ajout d'un code pour les tests avec arduino (blink) Do...
ajout d'un readme pour le dossier de tests DomkenHe...
Implémenter la case départ : flasher la carte RFID ave...
Implémenter la case sofa, avec MQTT et émission infr...
Implémentation du code de la case de fin, avec conne...
Corriger les codes OTA. Erreurs de compilations Domk...
Ajouter les paramètres MQTT aux données d'identific...
Ajouter des informations au readme : MQTT, structur...
Modifier le programme de test RFID en utilisant la fo...
Ajouter un code Arduino à utiliser une fois avec chaq...
Renommer test_IR pour avoir un chemin Arduino vali...
Supprimer fichier de backup non voulu DomkenHena

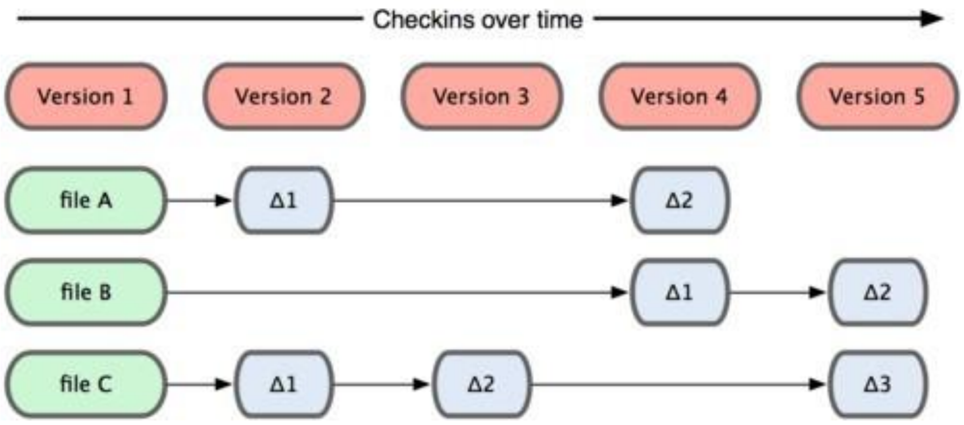
Graph	Description	Date	Author	Commit
	Uncommitted Changes (8)	26 Feb 2025 16:56	*	*
	main origin origin/HEAD Enlever les paramètres de connexion de chaque projet, un exemple de fichier est donné à la racine	22 Feb 2025 18:01	DomkenHena	6f13a624
	ignorer le projet docker dans le même directory	22 Feb 2025 17:54	DomkenHena	031ba21d
	Ajouter la possibilité de continuer snas être connecté au WIFI	22 Feb 2025 17:53	DomkenHena	6053d5b1
	Utiliser Telnetstream pour SofaCell, ajouter un mode hors ligne si pas de wifi détecté	22 Feb 2025 17:52	DomkenHena	48de00ae
	Corriger la connexion à MQTT pour sofaCell, en ajoutant la paramétrisation du serveur. Permettre de ne pas rester coincé dans la boucle de connexion	22 Feb 2025 17:51	DomkenHena	b7e50ddd
	Enlever le MQTT user, chaque module so connecte à MQTT avec son hostname	22 Feb 2025 17:48	DomkenHena	5bed1f32
	Corriger l'index de lecture dans la fonction MIFARE_Read pour le code robot	22 Feb 2025 15:59	DomkenHena	8033e64e
	Changer l'ordre dans le test de lecture RFID	22 Feb 2025 15:59	DomkenHena	451330af
	Changer de la communication en telnet pour retours d'infos, mode lecture sur startcell, debug	22 Feb 2025 15:57	DomkenHena	4ae2c39e
	Changer le mot de passe pour l'upload OTA	22 Feb 2025 15:54	DomkenHena	8c7f3061
	Ajouter exemple de labyrinthe par défaut dans startCell	22 Feb 2025 11:47	DomkenHena	dc8ffc55
	Corriger bug d'affichage sur case départ	21 Feb 2025 14:57	DomkenHena	585fb877
	améliorer la lecture du labyrinthe de la carte NFC	21 Feb 2025 14:56	DomkenHena	61e42999
	Implémenter le sketch de test arduino. lecture de la carte et réception puis émission du code fauteuil	21 Feb 2025 12:57	DomkenHena	a4df0610
	ajout d'un code pour les tests avec arduino (blink)	21 Feb 2025 12:31	DomkenHena	37e4d021
	ajout d'un readme pour le dossier de tests	21 Feb 2025 12:30	DomkenHena	de655b70
	Implémenter la case départ : flasher la carte RFID avec le plan du labyrinthe reçu par MQTT	21 Feb 2025 12:29	DomkenHena	66522294
	Implémenter la case sofa, avec MQTT et émission infrarouge	21 Feb 2025 12:29	DomkenHena	c5d32e9d
	Implémentation du code de la case de fin, avec connection MQTT et réception infrarouge	21 Feb 2025 12:28	DomkenHena	3460fd8a
	Corriger les codes OTA. Erreurs de compilations	21 Feb 2025 12:27	DomkenHena	c7feabf5
	Ajouter les paramètres MQTT aux données d'identification	21 Feb 2025 12:26	DomkenHena	99dc9048
	Ajouter des informations au readme : MQTT, structure MIFARE	21 Feb 2025 12:25	DomkenHena	49479c4a
	Modifier le programme de test RFID en utilisant la fonction de lecture seule et écriture avec clé	20 Feb 2025 16:34	DomkenHena	203cf185
	Ajouter un code Arduino à utiliser une fois avec chaque carte RFID pour la passer en lecture seule.	20 Feb 2025 16:31	DomkenHena	229ffb66

```
PS C:\Users\mdpdomco\Repos\B3Laby>
```

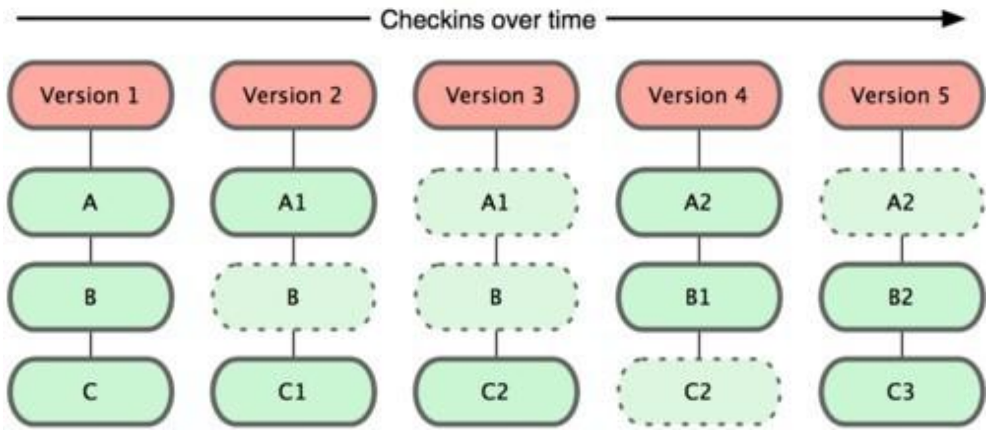
Git basics with local repo

Snapshots, not differences

- Other systems tend to store data as changes to a base version of each file



- Git stores data as snapshots of the project over time

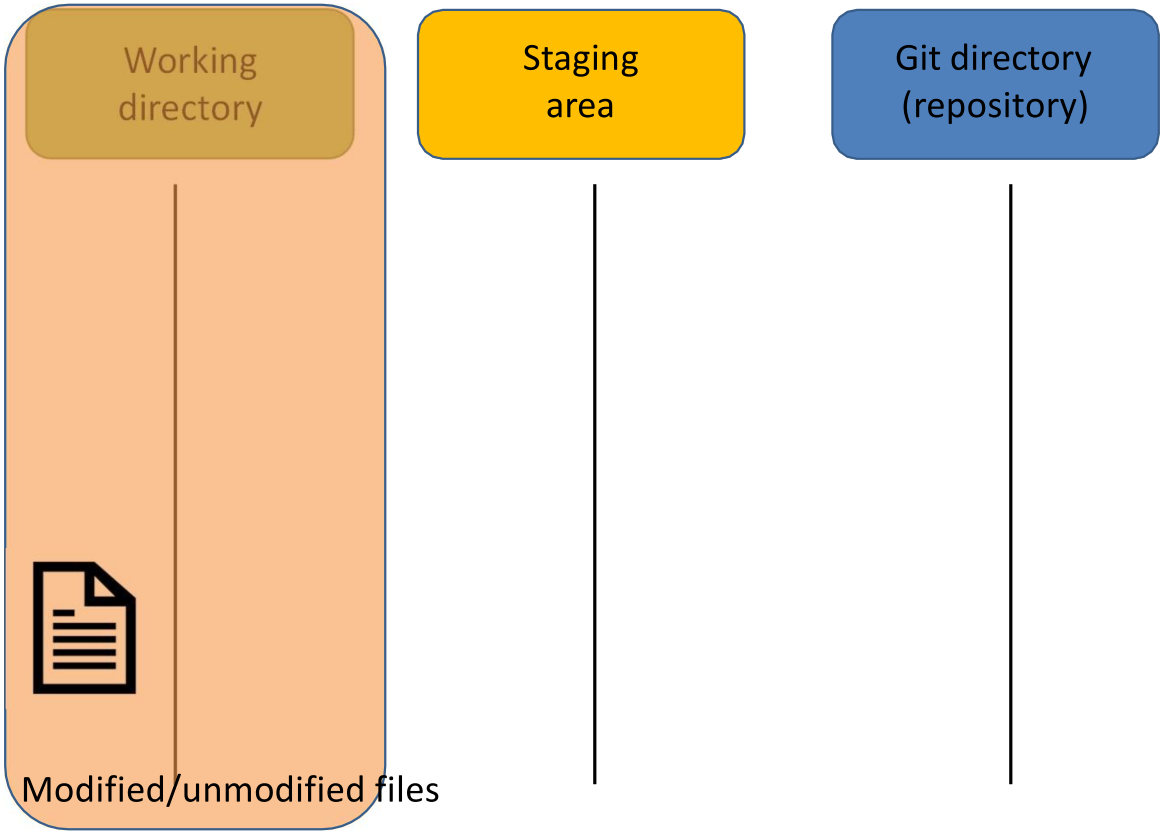


Git workflow: The three states

- In a Git repository your file can reside in three main states:
  - Modified
  - Staged
  - Committed

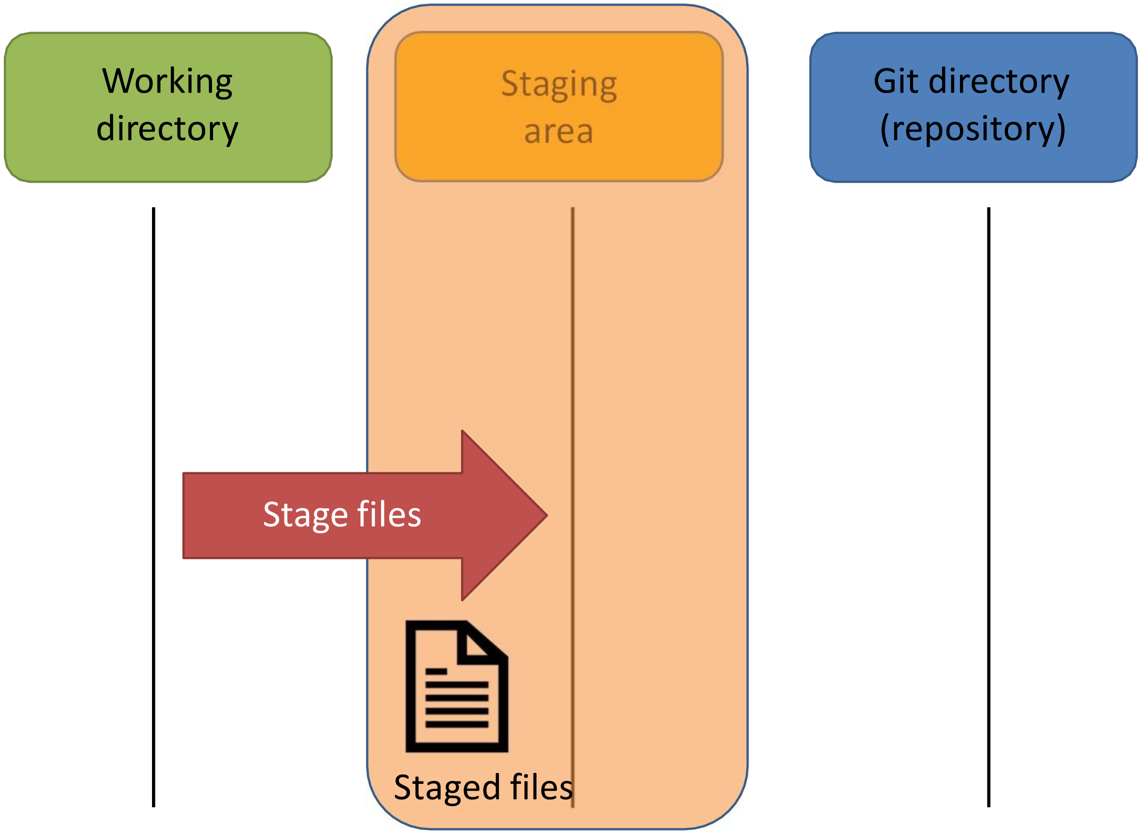
What does this mean?

Git workflow: The three states



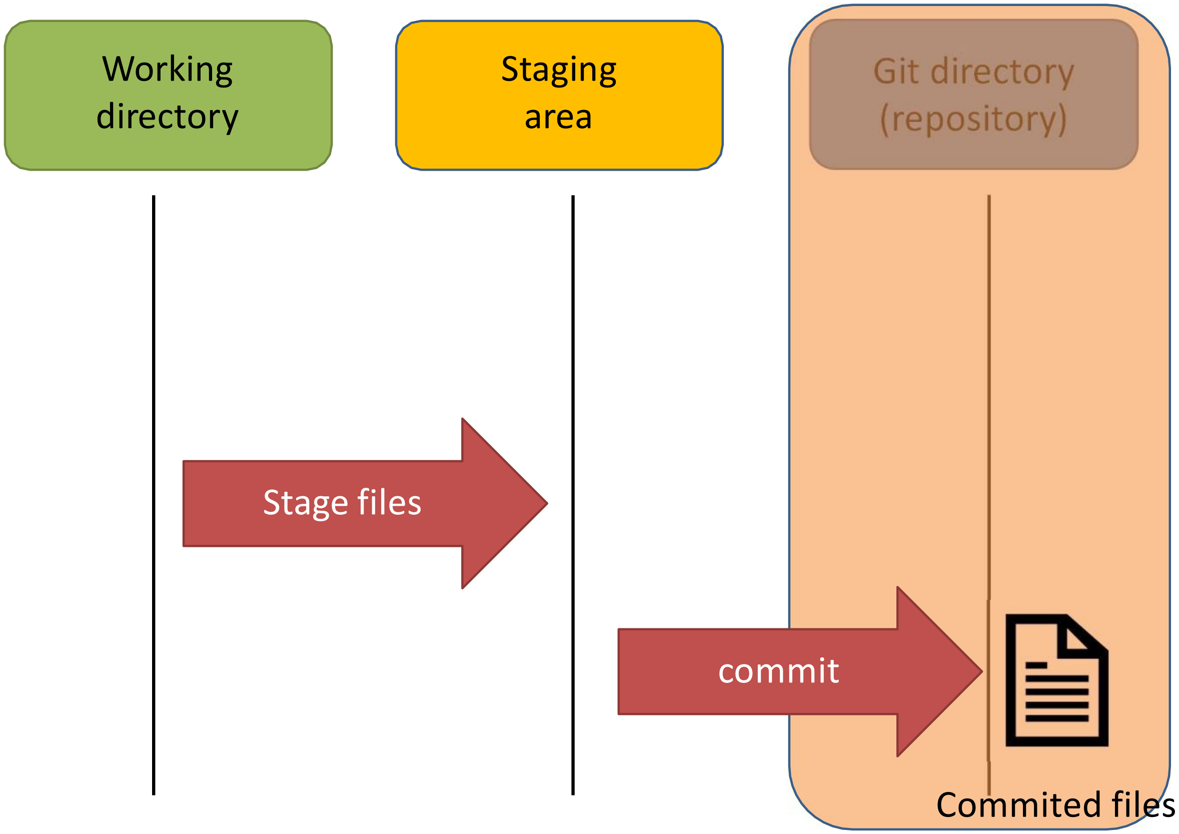
- You modify files in your working directory

Git workflow: The three states



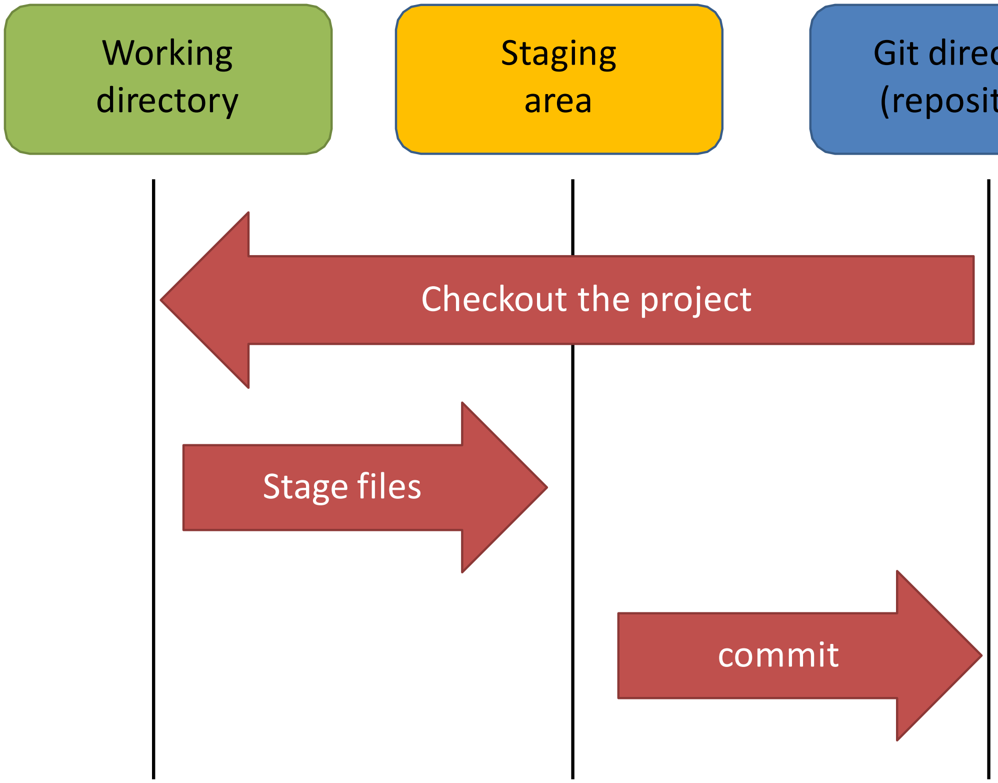
- You stage the files, adding snapshots of them to your staging area

Git workflow: The three states



- You do a commit that stores snapshots permanently to your Git directory

Git workflow: The three states



Working directory

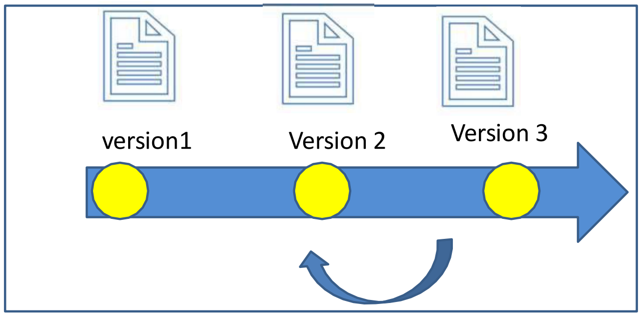
Staging area

Git directory (repository)

Checkout the project

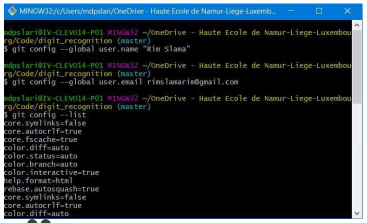
Stage files

commit



- Then, you can checkout any existing version, make changes, stage them and commit

Get ready to use git!


MINGW32:/c/Users/mdpslari/OneDrive - Haute Ecole de Namur-Liege-Luxemb...

```
mdpslari@IV-CLEVO14-P01 MINGW32 ~/OneDrive - Haute Ecole de Namur-Liege-Luxembou
rg/Code/digit_recognition (master)
$ git config --global user.name "Rim Slama"

mdpslari@IV-CLEVO14-P01 MINGW32 ~/OneDrive - Haute Ecole de Namur-Liege-Luxembou
rg/Code/digit_recognition (master)
$ git config --global user.email rimslamarim@gmail.com

mdpslari@IV-CLEVO14-P01 MINGW32 ~/OneDrive - Haute Ecole de Namur-Liege-Luxembou
rg/Code/digit_recognition (master)
$ git config --list
core.symlinks=false
core.autocrlf=true
core.fscache=true
color.diff=auto
color.status=auto
color.branch=auto
color.interactive=true
help.format=html
rebase.autosquash=true
core.symlinks=false
core.autocrlf=true
color.diff=auto
```

1. First time Git setup!

- Set the name and email for Git to use when you commit:

```
$ git config --global user.name “Corentin Domken”
$ git config --global user.email corentin.domken@henallux.be
```

- You can also set some global features:

```
$ git config --global alias.co checkout
$ git config --global alias.br branch
$ git config --global alias.ci commit
$ git config --global alias.st status
```

- Checking your settings

```
$  git config --list
```

Create a local copy of a repo

2. Two common scenarios: (only do one of these)

   a) To **clone an already existing repo** to your current directory:

   ```
   $ git clone <url> [local dir name]
   ```

   This will create a directory named *local dir name*, containing a working copy of the files from the repo, and a `.git` directory (used to hold the staging area and your actual repo)

   b) To **create a new local Git repo** in your current directory:

   ```
   $ git init
   ```

   This will create a `.git` directory in your current directory.

Add and commit a file

- Create 2 files a files **`README.md hello.py`** and write inside some text
- The first time we ask a file to be tracked, *and* **every time before we commit a file** we must add it to the staging area:

```
$ git status
$ git add README.md hello.py
```

This takes a snapshot of these files at this point in time and adds it to the staging area.

- To move staged changes into the repo we commit:

```
$ git commit –m “Add initial code”
```

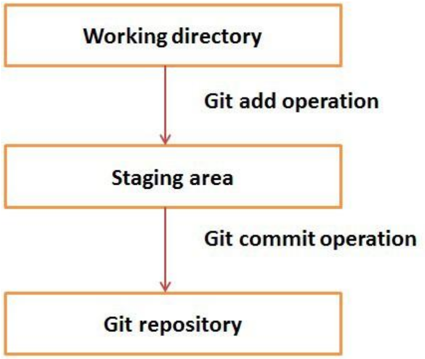
Note: To unstage a change on a file before you have committed it:

```
$ git reset HEAD  --  filename
```

Note: To unmodify a modified file:

```
$ git checkout  --  filename
```

Note: These commands are just acting on **your local version of repo**.


Working directory
Git add operation
Staging area
Git commit operation
Git repository

Viewing changes

- To view the **status** of your files in the working directory and staging are

`$ git status` or `$ git status -s`

(`-s` shows a short one line version similar to svn)

- compare the working directory with index :

`$ git diff [filename]`

- To see staged changes:

`$ git diff –cached [filename]`

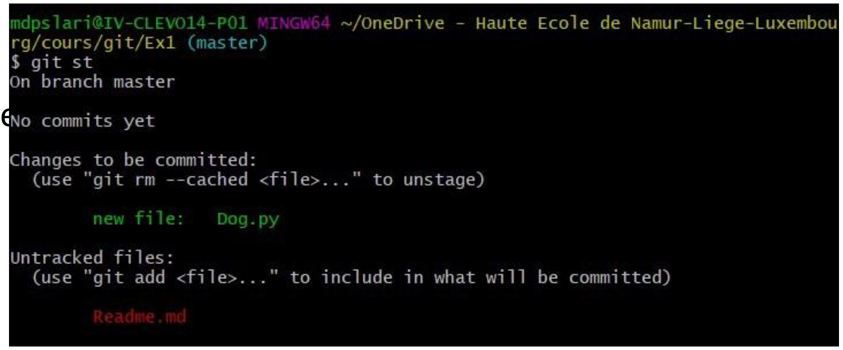
- Compare the working directory with local repo:

`$ git diff HEAD [filename]`

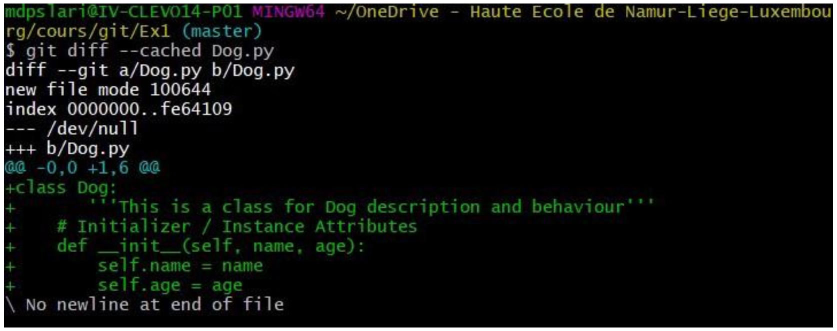
- To see a log of all changes in your local repo:

`$ git log` or

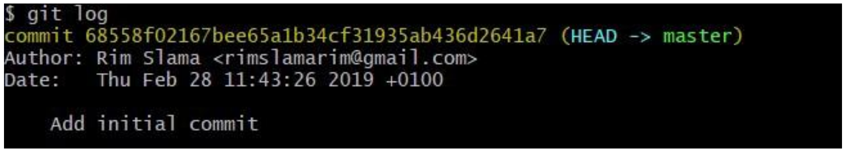
`$ git log --oneline` (to show a shorter version)



```
mdpslari@IV-CLEVO14-P01 MINGW64 ~/OneDrive - Haute Ecole de Namur-Liege-Luxembou
rg/cours/git/Ex1 (master)
$ git st
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   Dog.py

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

        Readme.md
```



```
mdpslari@IV-CLEVO14-P01 MINGW64 ~/OneDrive - Haute Ecole de Namur-Liege-Luxembou
rg/cours/git/Ex1 (master)
$ git diff --cached Dog.py
diff --git a/Dog.py b/Dog.py
new file mode 100644
index 0000000..fe64109
--- /dev/null
+++ b/Dog.py
@@ -0,0 +1,6 @@
+class Dog:
+       '''This is a class for Dog description and behaviour'''
+    # Initializer / Instance Attributes
+    def __init__(self, name, age):
+        self.name = name
+        self.age = age
\ No newline at end of file
```



```
$ git log
commit 68558f02167bee65a1b34cf31935ab436d2641a7 (HEAD -> master)
Author: Rim Slama <rimslamarim@gmail.com>
Date:   Thu Feb 28 11:43:26 2019 +0100

    Add initial commit
```

Undoing what is done

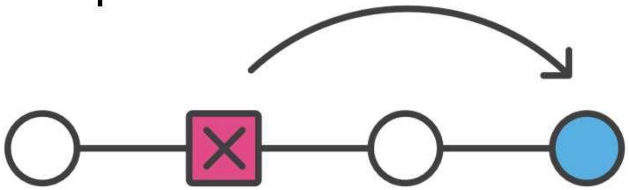
To unstage a change on a file before you have committed it:

```
$ git reset HEAD  --  filename
```

To unmodify a modified file:

```
$ git checkout  --  filename
```

- git revert
  - Reverts a commit
  - Does not delete the commit object, just applies a patch
  - Reverts can themselves be reverted!
- Git never deletes a commit object
  - It is very hard to shoot yourself in the foot!



Ignore?

- Create a file called .gitignore

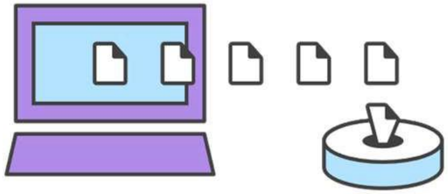
```
$ touch .gitignore
```

- Add files to ignore in this file

```
$ echo debug.log >> .gitignore
```

- Commit the gitignore file

```
$ git commit -m "Start ignoring debug.log"
```



Branching and merging

To create a branch called experimental:

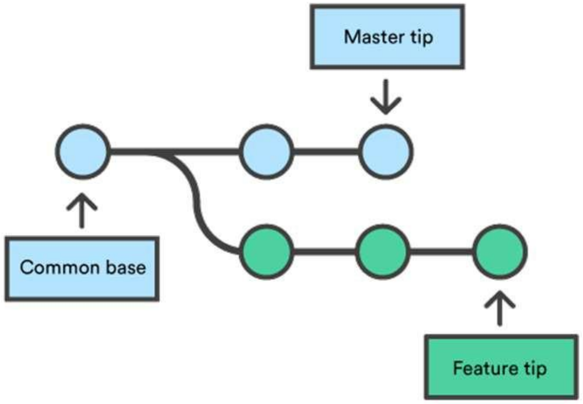
- `$ git branch experimental`

To list all branches: (* shows which one you are currently on)

- `$ git branch`

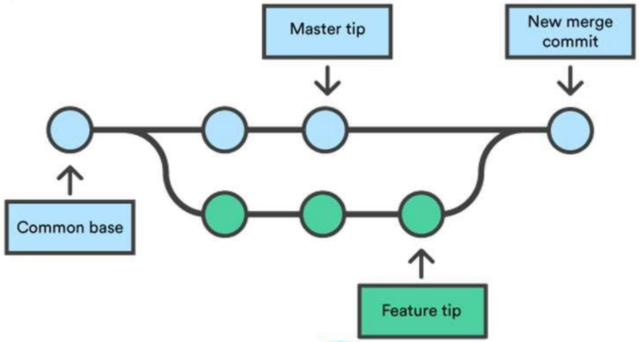
To switch to the experimental branch:

- `$ git checkout experimental`



Later on, changes between the two branches differ, to merge changess from experimental into the master:

- `$ git checkout master`
- `$ git merge experimental`

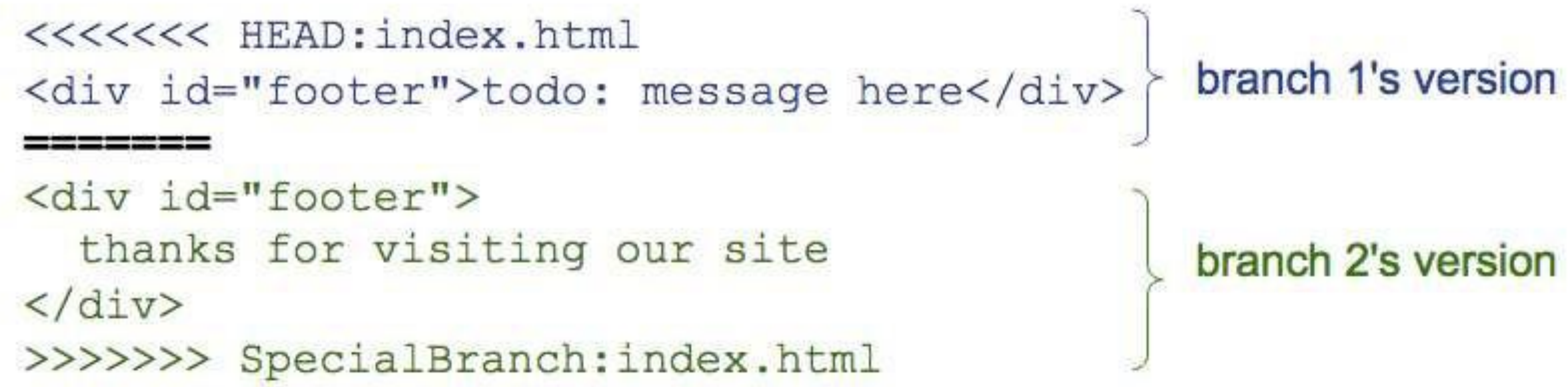


Note: `git log --graph` can be useful for showing branches.

Note: These branches are in *your local repo*!

Merge conflicts

- The conflicting file will contain <<< and >>> sections to indicate where Git was unable to resolve a conflict:



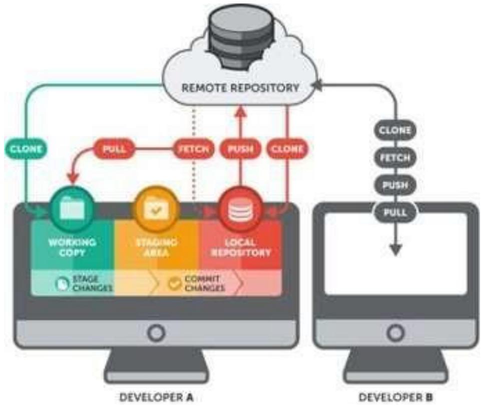
```
<<<<<<< HEAD:index.html
<div id="footer">todo: message here</div>
========
<div id="footer">
  thanks for visiting our site
</div>
>>>>>>> SpecialBranch:index.html
```

- Find all such sections, and edit them to the proper state (whichever of the two versions is newer / better / more correct).

Interaction with remote repo

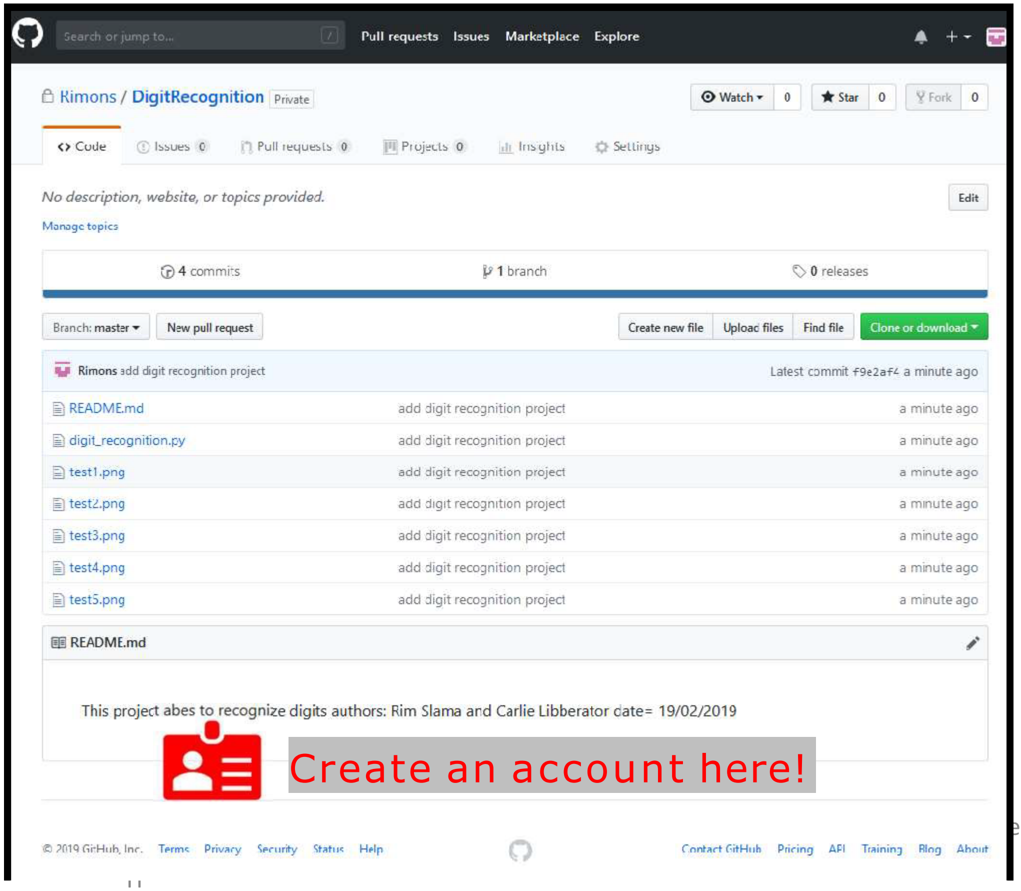
GitHub

- GitHub.com is a site for online storage of Git repositories.
  - You can create a remote repo there and push code to it.
  - Many open source projects use it, such as the Linux kernel
  - You can get free space for open source projects, or you can pay for private projects.
  - Free private repos for educational use: github.com/edu



Online storage for Git repos

- Github https://github.com/
- Bitbucket https://bitbucket.org



Rimons / DigitRecognition Private

No description, website, or topics provided.

4 commits | 1 branch | 0 releases

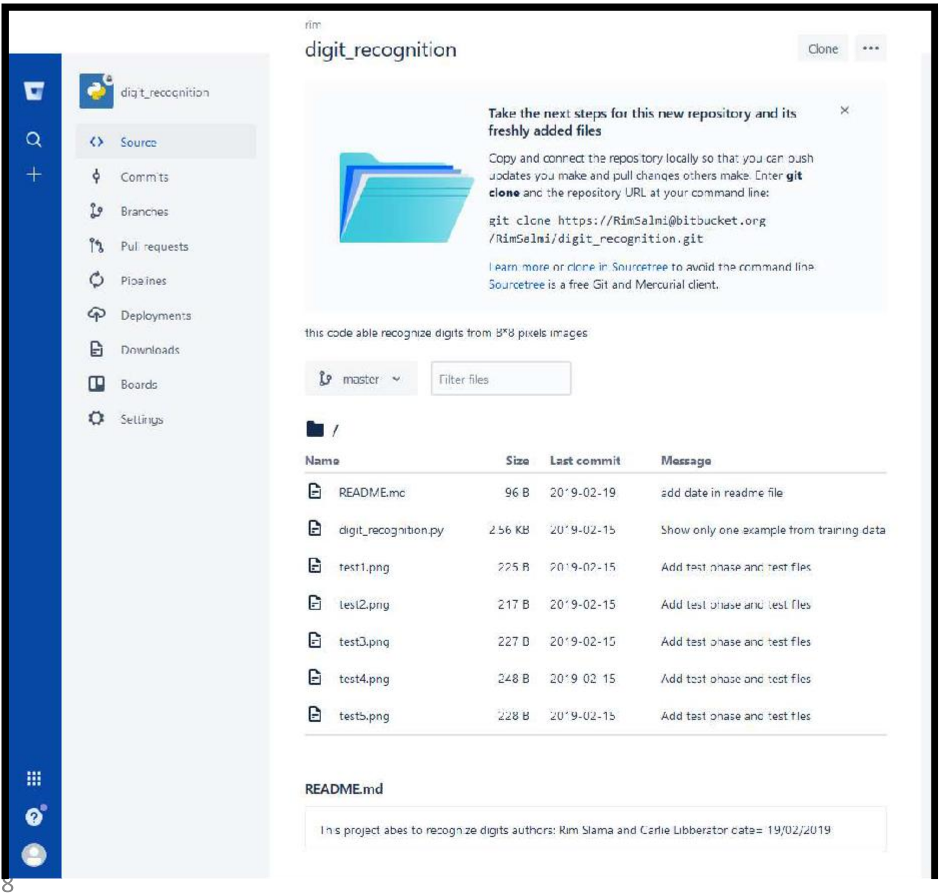
Rimons add digit recognition project — Latest commit f9e2af4 a minute ago

README.md	add digit recognition project	a minute ago
digit_recognition.py	add digit recognition project	a minute ago
test1.png	add digit recognition project	a minute ago
test2.png	add digit recognition project	a minute ago
test3.png	add digit recognition project	a minute ago
test4.png	add digit recognition project	a minute ago
test5.png	add digit recognition project	a minute ago

README.md

This project abes to recognize digits authors: Rim Slama and Carlie Libberator date= 19/02/2019

Create an account here!



rim

digit_recognition

Take the next steps for this new repository and its freshly added files

Copy and connect the repository locally so that you can push updates you make and pull changes others make. Enter **git clone** and the repository URL at your command line:

```
git clone https://RimSalmi@bitbucket.org
/RimSalmi/digit_recognition.git
```

Learn more or clone in Sourcetree to avoid the command line. Sourcetree is a free Git and Mercurial client.

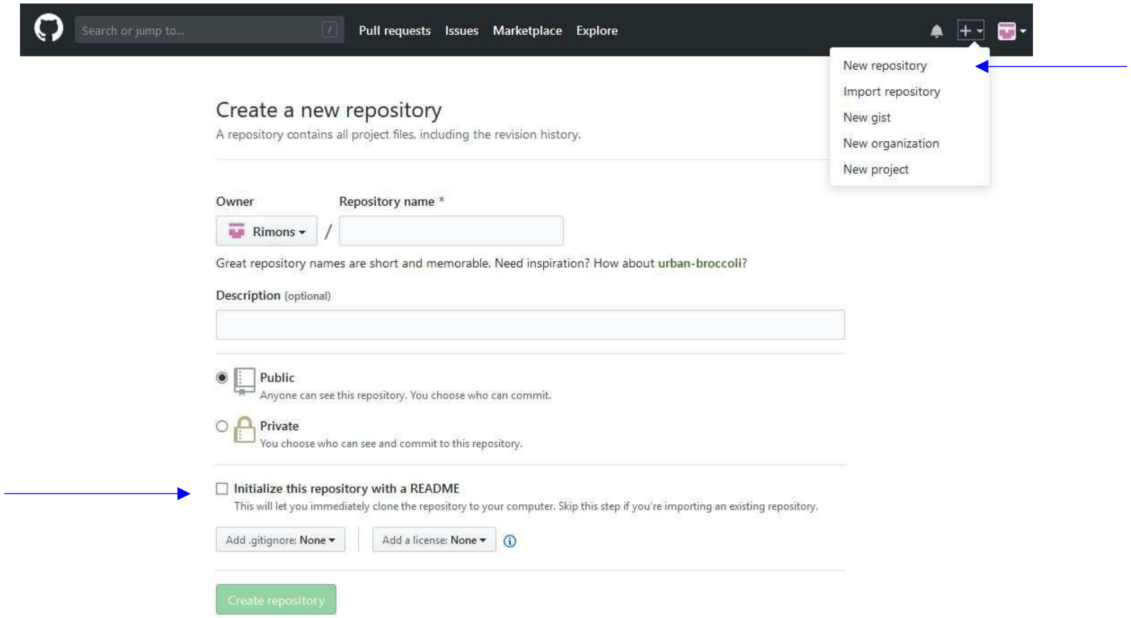
this code able recognize digits from 8*8 pixels images

Name	Size	Last commit	Message
README.md	96 B	2019-02-19	add date in readme file
digit_recognition.py	2.56 KB	2019-02-15	Show only one example from training data
test1.png	225 B	2019-02-15	Add test phase and test files
test2.png	217 B	2019-02-15	Add test phase and test files
test3.png	227 B	2019-02-15	Add test phase and test files
test4.png	248 B	2019-02-15	Add test phase and test files
test5.png	228 B	2019-02-15	Add test phase and test files

README.md

This project abes to recognize digits authors: Rim Slama and Carlie Libberator date= 19/02/2019

Push existing local repo to remote



Create a new repository

A repository contains all project files, including the revision history.

Owner: Rimons ▾ / Repository name *

Great repository names are short and memorable. Need inspiration? How about urban-broccoli?

Description (optional)

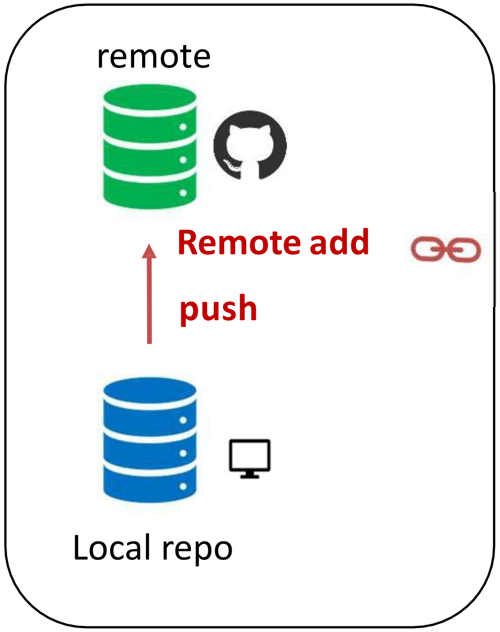
● Public
Anyone can see this repository. You choose who can commit.

○ Private
You choose who can see and commit to this repository.

□ Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

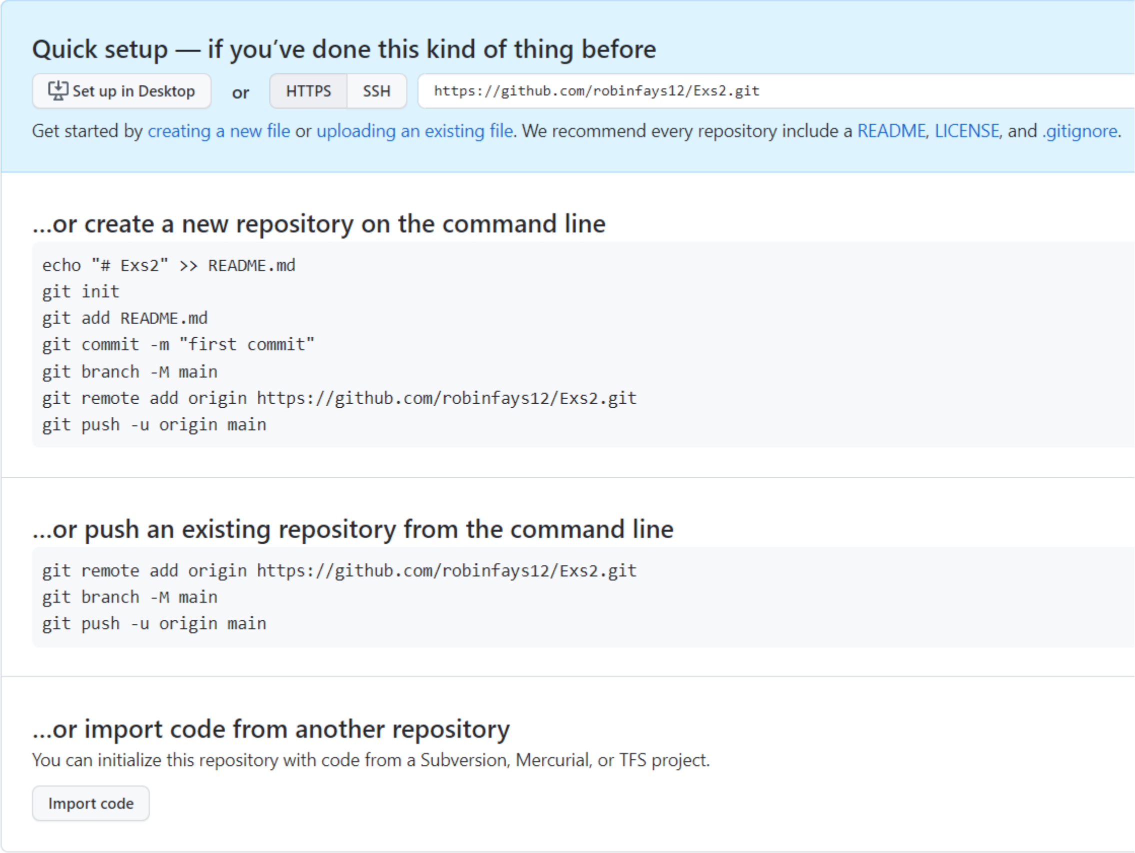
Add .gitignore: None ▾ | Add a license: None ▾

Create repository



```
$ git remote add origin https://github.com/DomkenHena/GIT-exercise
$ git branch –M main
$ git push –u origin master
```

Push existing local repo to remote



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

https://github.com/robinfays12/Exs2.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 "# Exs2" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/robinfays12/Exs2.git
git push -u origin main
```

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

```
git remote add origin https://github.com/robinfays12/Exs2.git
git branch -M main
git push -u origin main
```

...or import code from another repository

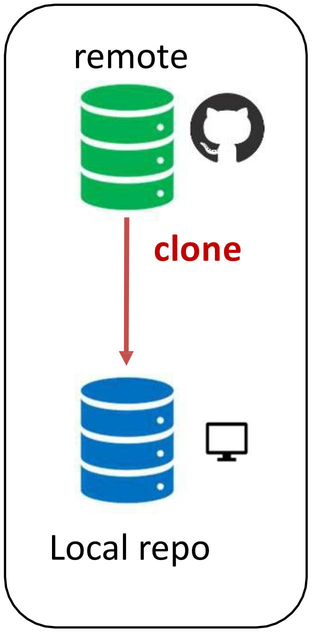
You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Clone from remotes

- To <u>**clone an already existing repo**</u> to your current directory:

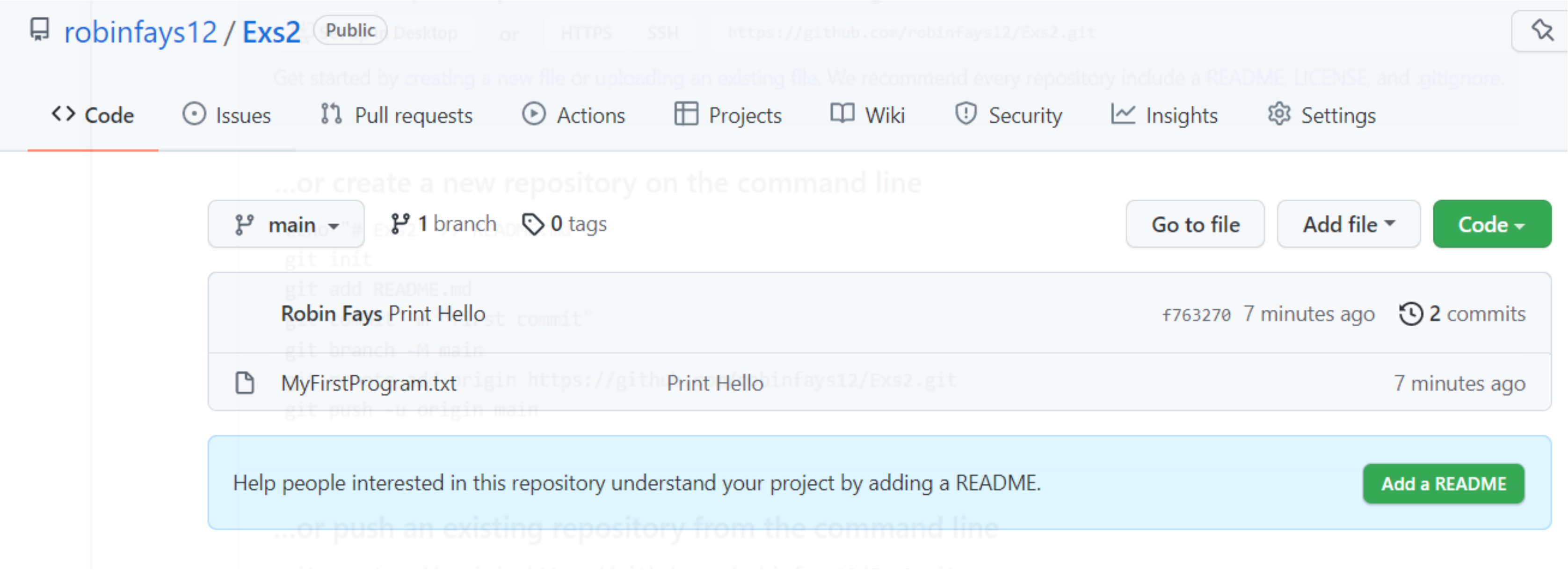
$ git clone <url> [local dir name]

- This will create a directory named *local dir name*, containing a working copy of the files from the repo,



Clone from remotes

```
$ git clone https://github.com/DomkenHena/GIT-exercise
```



Robin Fays Print Hello f763270 7 minutes ago 2 commits

MyFirstProgram.txt Print Hello 7 minutes ago

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

Pulling and pushing

- To fetch the most recent updates from the remote repo into your local repo, and put them into your working directory:

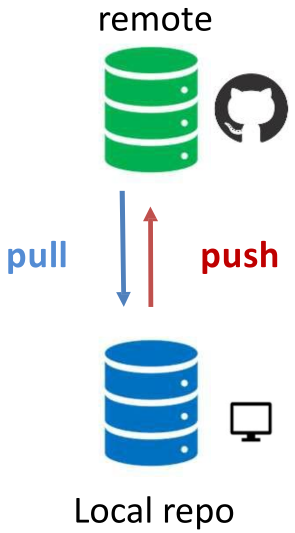
```
$ git pull origin master
```

- To push your changes from your local repo to the remote repo:

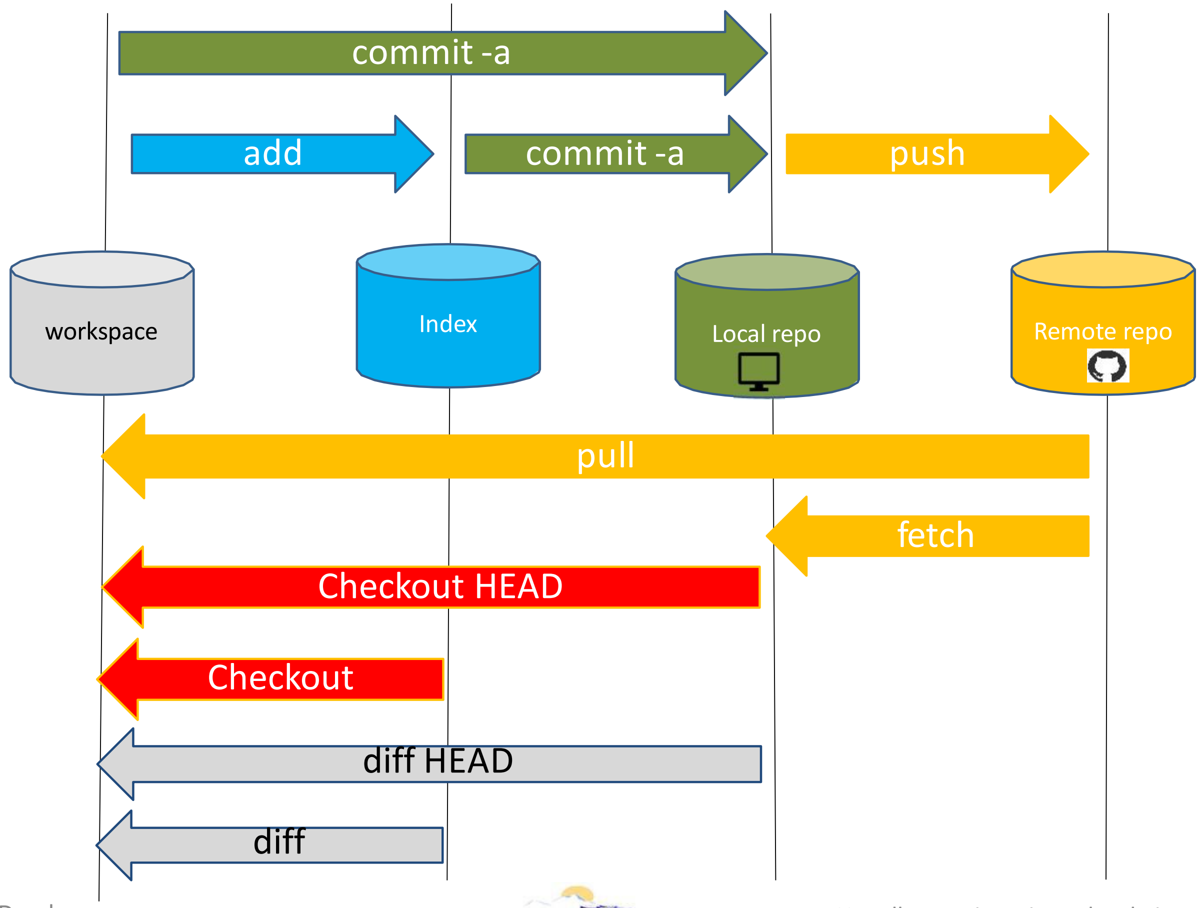
```
$ git push origin master
```

Notes: `origin` = an alias for the URL you cloned from

`master` = the remote branch you are pulling from/pushing to,
(the local branch you are pulling to/pushing from is your current branch)



Git data transport commands



Git and github: resume

- Question: Do I always have to use GitHub to use Git?
  - Answer: No! You can use Git locally for your own purposes.
  - Or you or someone else could set up a server to share files.
  - Or you could share a repo with users on the same file system, as long everyone has the needed file permissions).

Practice

Practice

- Install git (Git website: http://git-scm.com/)
- Once installed check Git version : $git --version

1. **$ git config --global user.name “Your Name”**
2. **$ git config --global user.email youremail@whatever.com**

Local

- Create an account on github: https://github.com/

Remote

Practice 1

1. Create a new repository you call :ProjStudent1.
2. Create a new local git repo: $ git init
3. Create a file named *userID*.txt (e.g. test.txt), add a pdf file and ignore it
4. Get the status of git: $ git status, $ git status -s
5. Add the file: **$ git add *userID*.txt**
6. Get the status of git: $ git status, $ git status -s
7. Commit the file to your local repo:
   $ git commit -m “added test.txt **file”**
   $ git status, $ git status -s, $ git log --oneline

*WAIT, DO NOT GO ON TO THE NEXT STEPS UNTIL YOU ARE TOLD TO!!

1. Add new repo to github you call ProjStudent1 (do not select initial readme!)
2. Push the project on the remote

```
$ git remote add origin https://github.com/username/ProjStudent1.git
$ git push -u origin master
```

Add more files and commit them, then

1. Pull from remote repo: **$git pull origin master**
2. Push to remote repo: **$git push origin master**

Practice 2

1. **Clone the existing repo Ex1 using the URL: https://github.com/DomkenHena/GIT-exercise**

```
$ git clone https://github.com/DomkenHena/GIT-exercise
```

Then try:

2. Have a look on all commits

```
$ git log, $ git log -oneline
```

3. Create a new branch and checkout it:

```
$ git branch experimental

  $ git checkout experimental
```

1. Create 1 new file named *userID*.txt (e.g. rea.txt)
2. Modify existing file
3. Ceck the status of git

```
$ git status, $ git status -s
```

4. Commit the files you modified to local repo:

```
$ git add userID.txt
$ git commit -m “added rea.txt file”
$ git status, $ git status -s, $ git log -oneline
```

Chechout master branch and merge the 2 branchs

Questions

References

- http://git.or.cz/
  - http://git.or.cz/course/cvs.html (For CVS users)
  - http://git.or.cz/course/svn.html (For SVN users)
- http://www.kernel.org/pub/software/scm/git/docs/user-manual.html
- http://jonas.iki.fi/git_guides/HTML/git_guide/
- https://github.com/praqma-training/git-katas
- https://medium.freecodecamp.org/follow-these-simple-rules-and-youll-become-a-git-and-github-master-e1045057468f