



An introduction to GIT

Xavier Meyer
10/04/2017

Plan

Version control

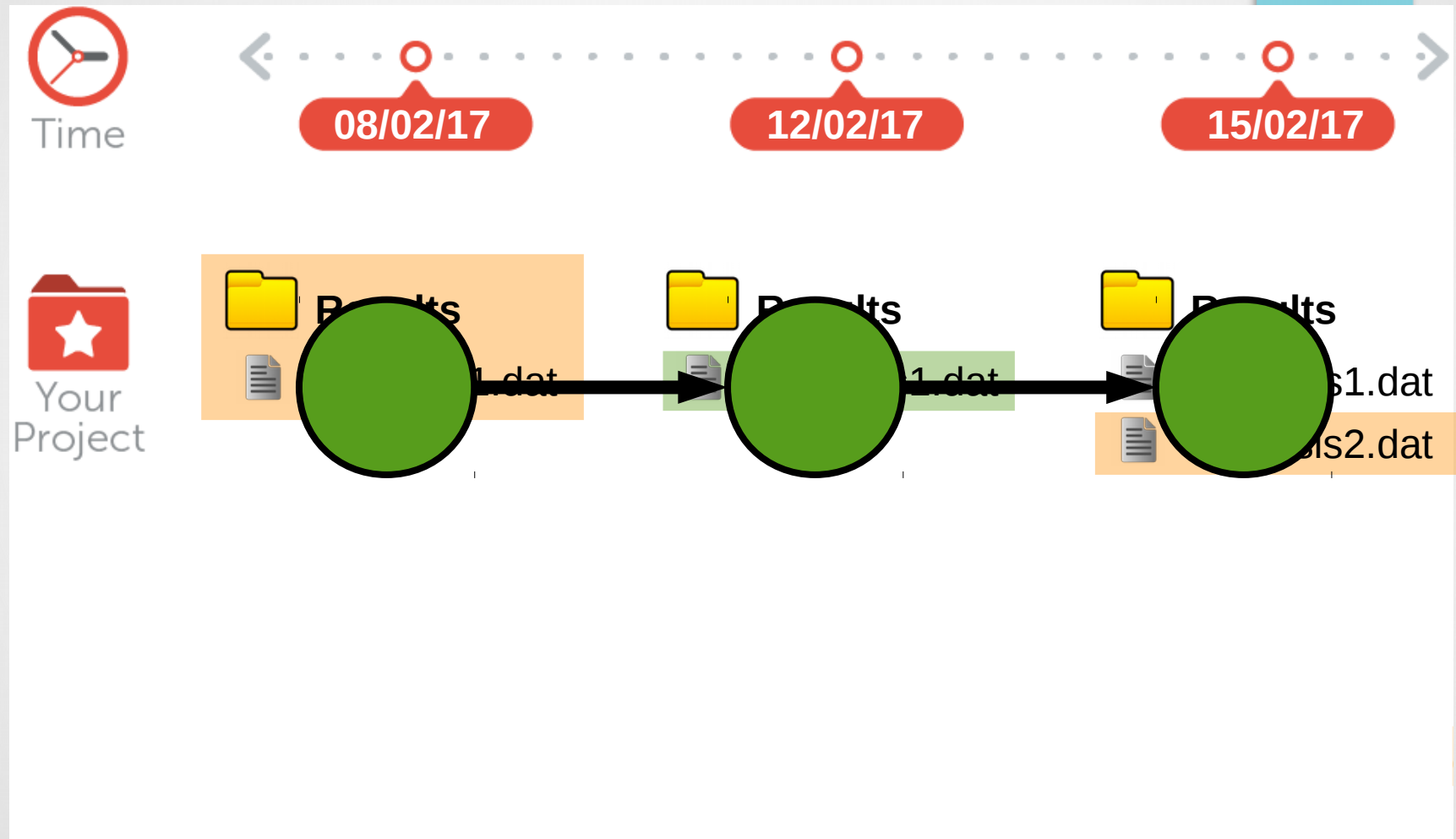
GIT concepts

- Local repository

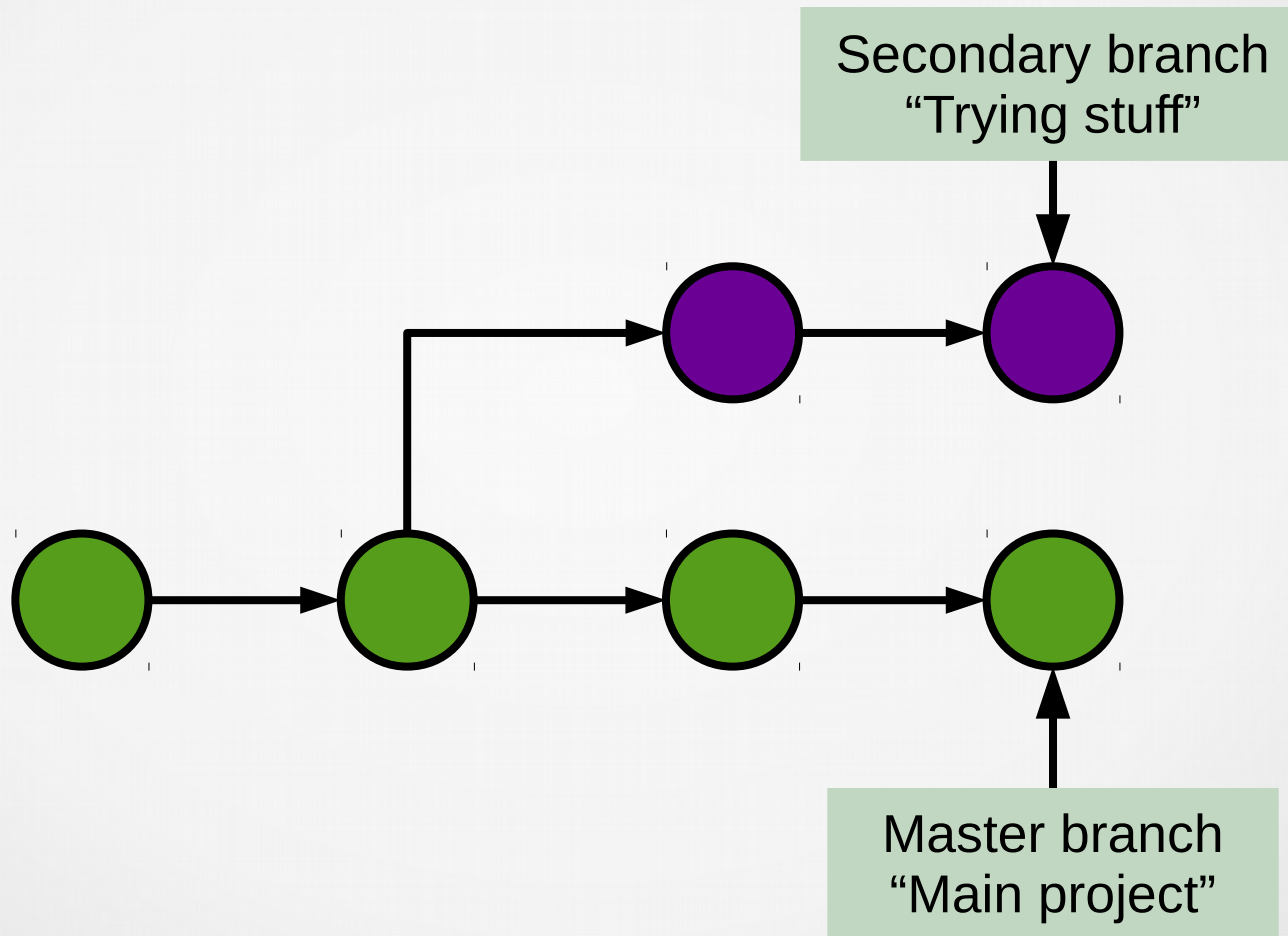
- Remote repository

Interactive example

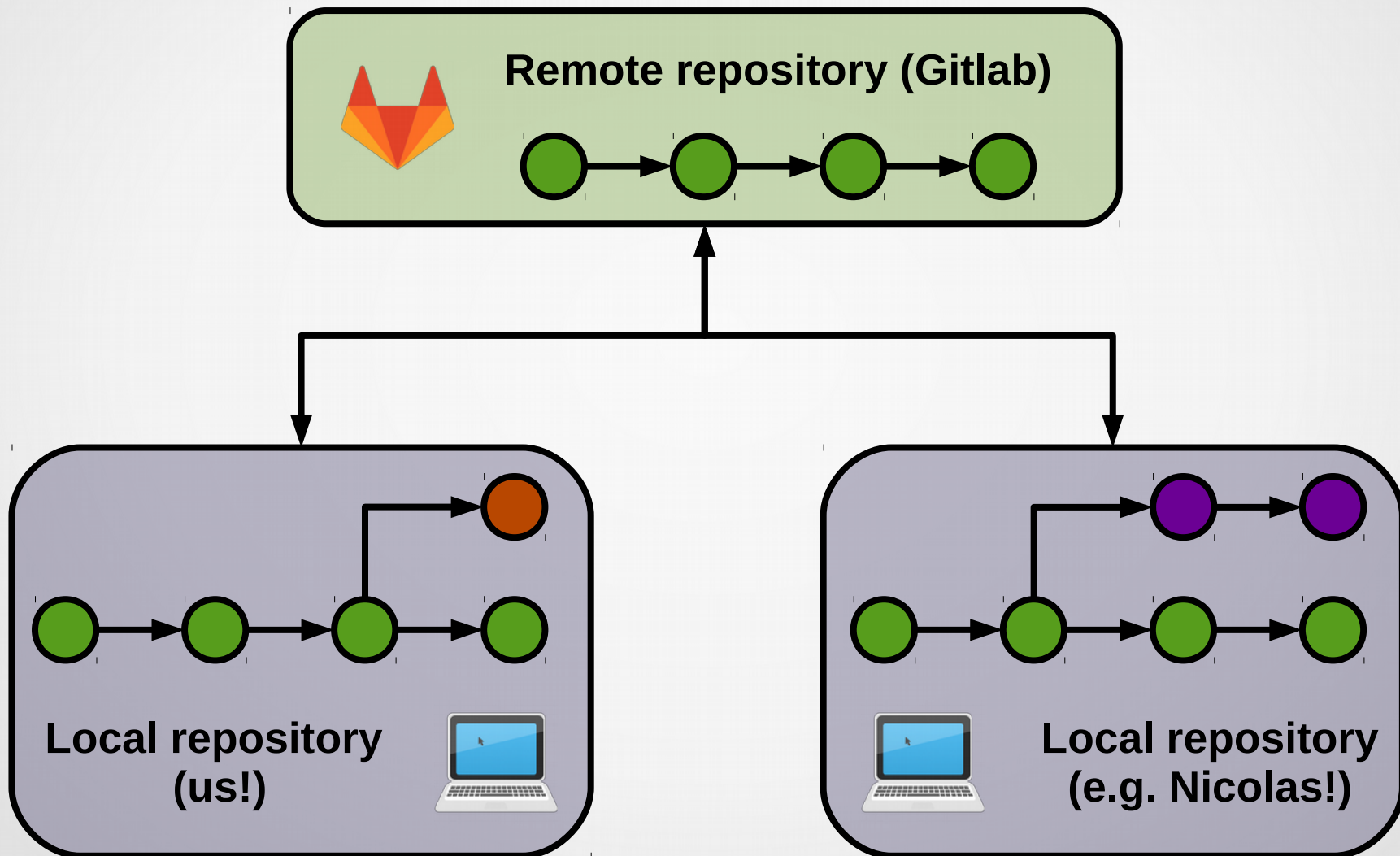
Version control



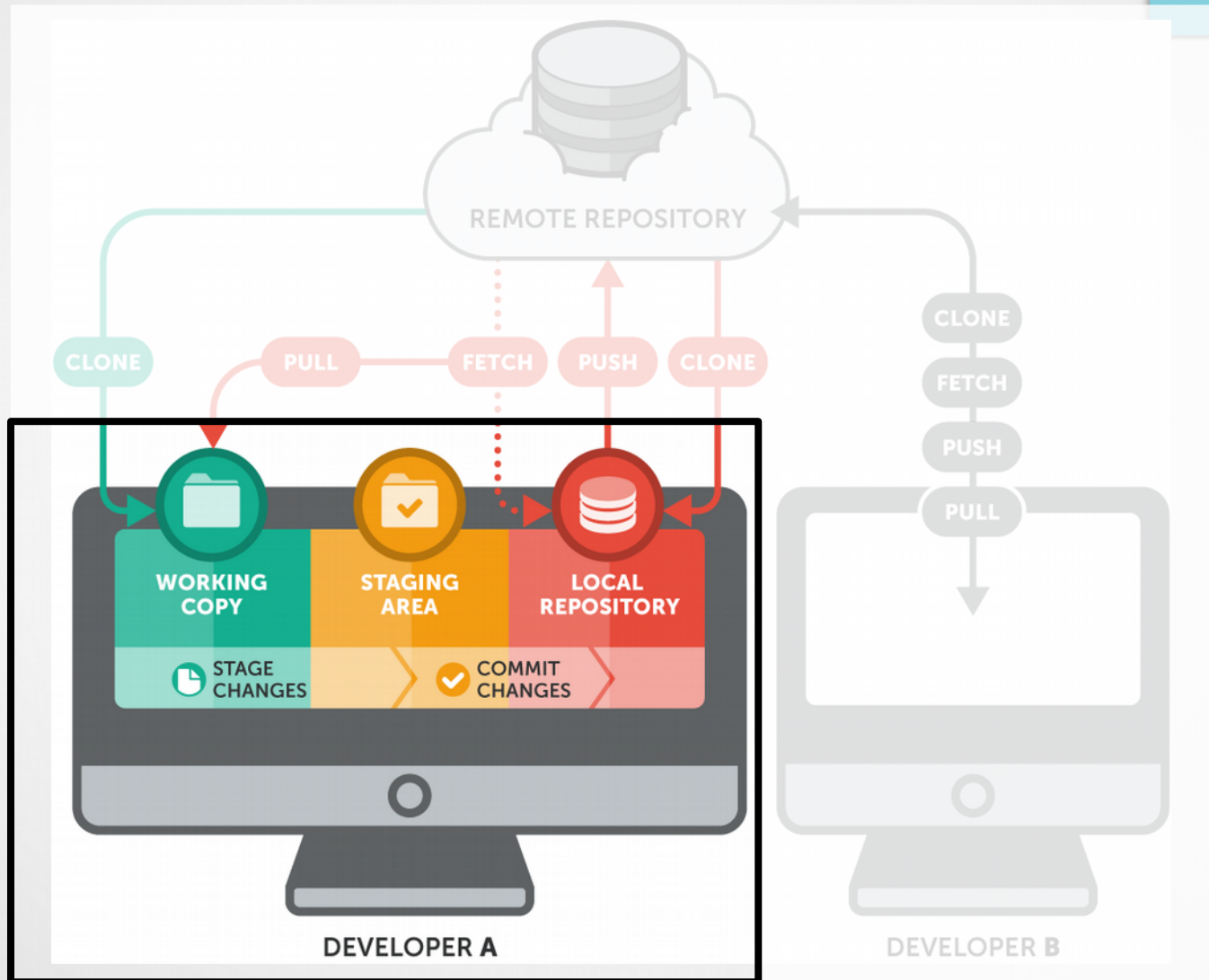
Branching, or let's try something new!



Use case : our setting!



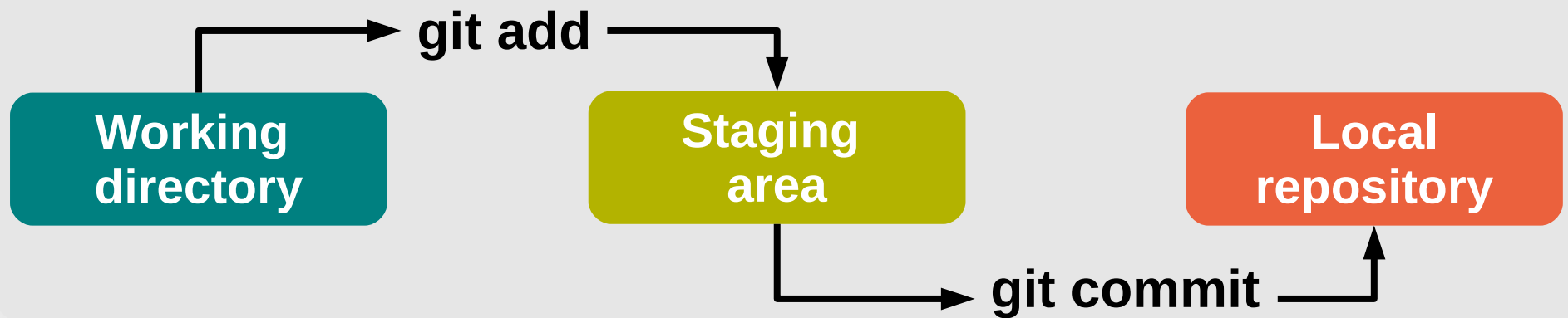
The GIT way



On our laptop



Git local flow



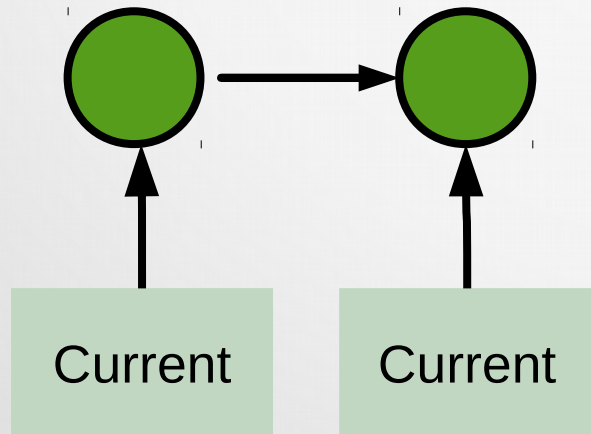
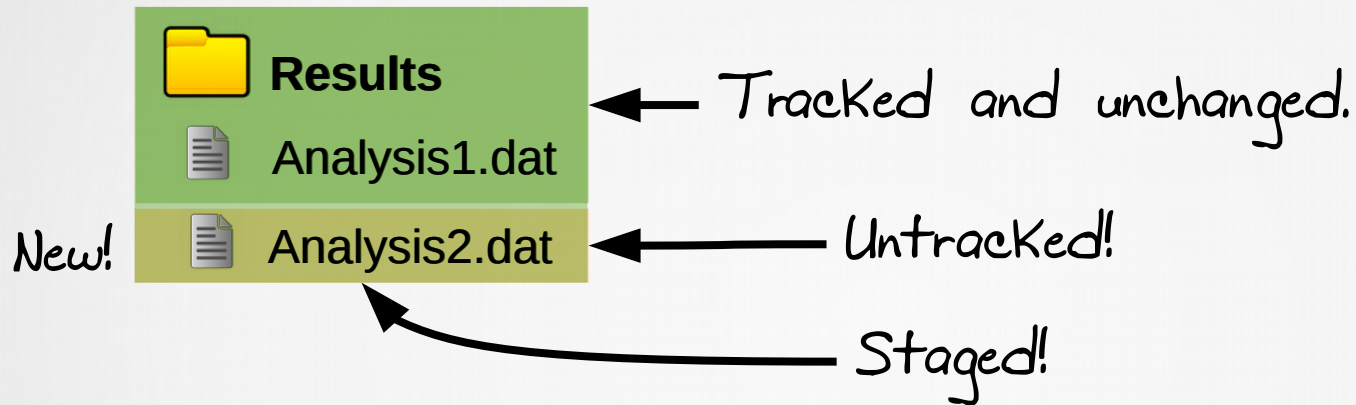
GIT file status

- Tracked and modified
- Untracked

- Staged

- Saved

Tracking/saving files



Terminal

```
~$ python someAnalysis.py  
~$ git add Results/Analysis2.dat  
~$ git commit -m "New file."
```


Managing our versions locally

- Staging files

`git add ...`

with `-A` : stage **all** untracked/modified files

with `< filenames >` : stage only files `< filenames >`

- Saving staged files

`git commit`

with `-m "A commit message."`

- Erasing an untracked files

`git checkout < filenames >`

File status within GIT

- What is modified/untracked ?

`git status`

- What has changed since the previous version ?

`git diff`

- What have been the last commits ?

`git log`

Moving through versions



Results



Analysis1.dat



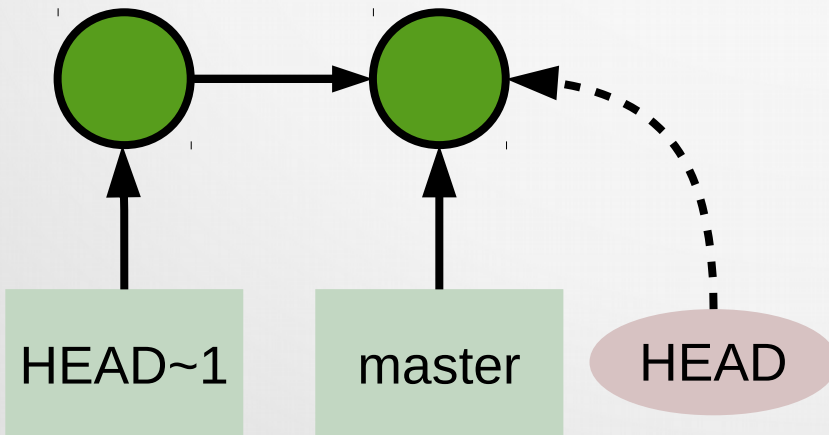
Results



Analysis1.dat



Analysis2.dat



Terminal

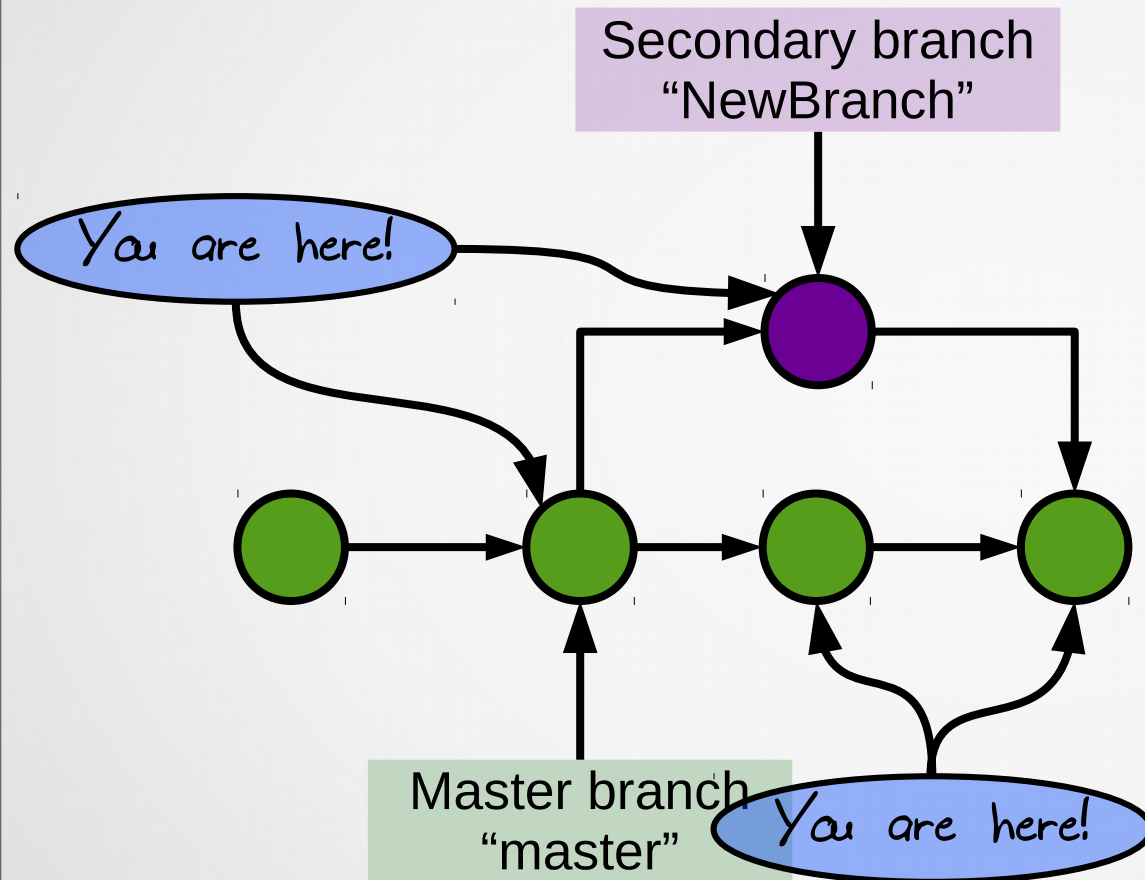
```
~$ git checkout HEAD~1  
~$ git checkout master
```

Moving through versions

git checkout

- with “HEAD~x”
 - Go back “x” to the xth previous version
- With commit **hashkey**
 - Go to said commit
- With “**master**”
 - Go to HEAD of master branch

Branching



Terminal

```
~$ git checkout -b NewBranch  
~$ ...  
~$ git checkout master  
~$ ...  
~$ git merge NewBranch
```

Git Branch/Merge

- Creating a new branch

`git checkout -b <branch_name>`

- Moving from branch to branch

`git checkout <branch_name>`

- Merging branches (NewBranch with master)

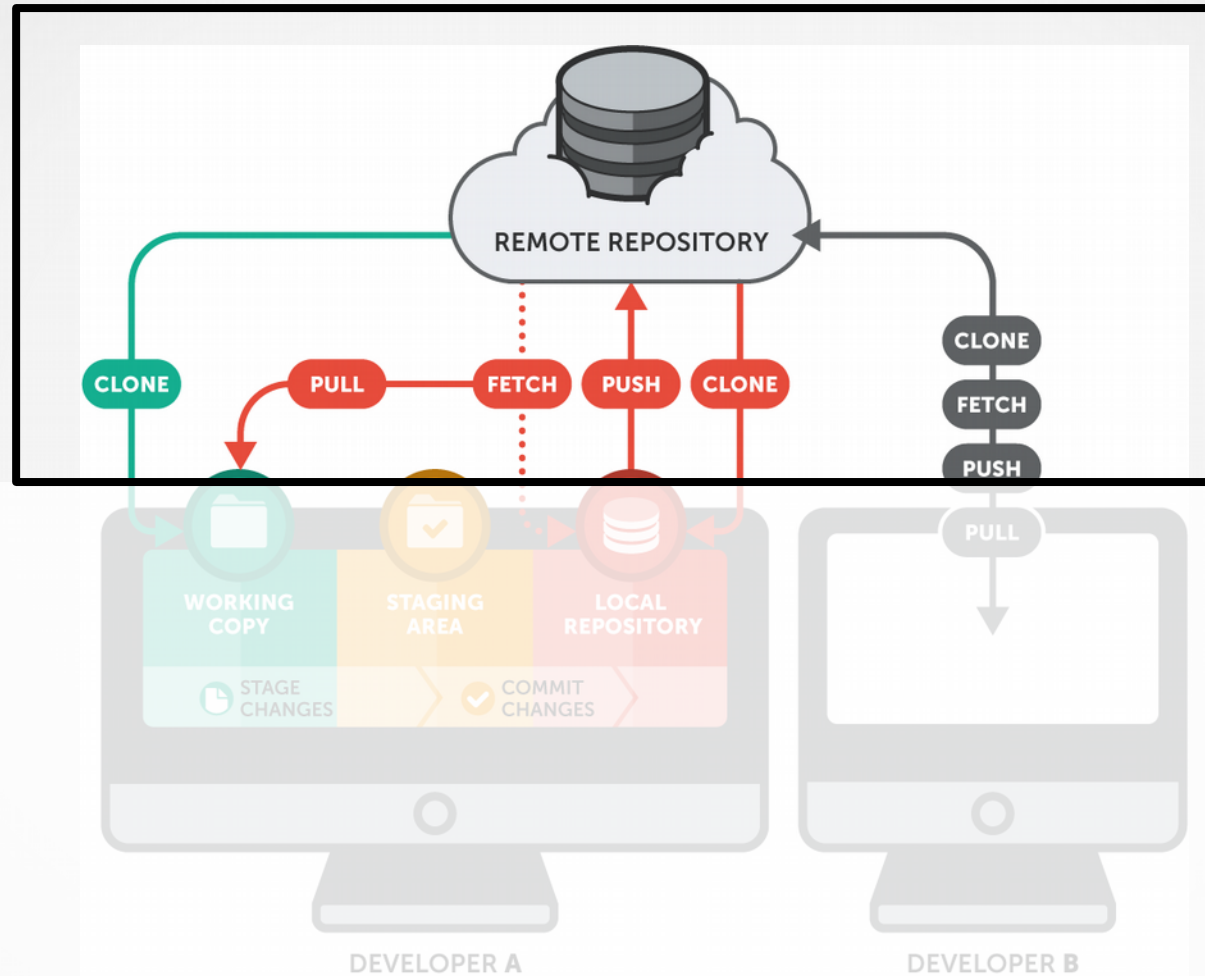
`git checkout master`

`git merge <branch_name>`

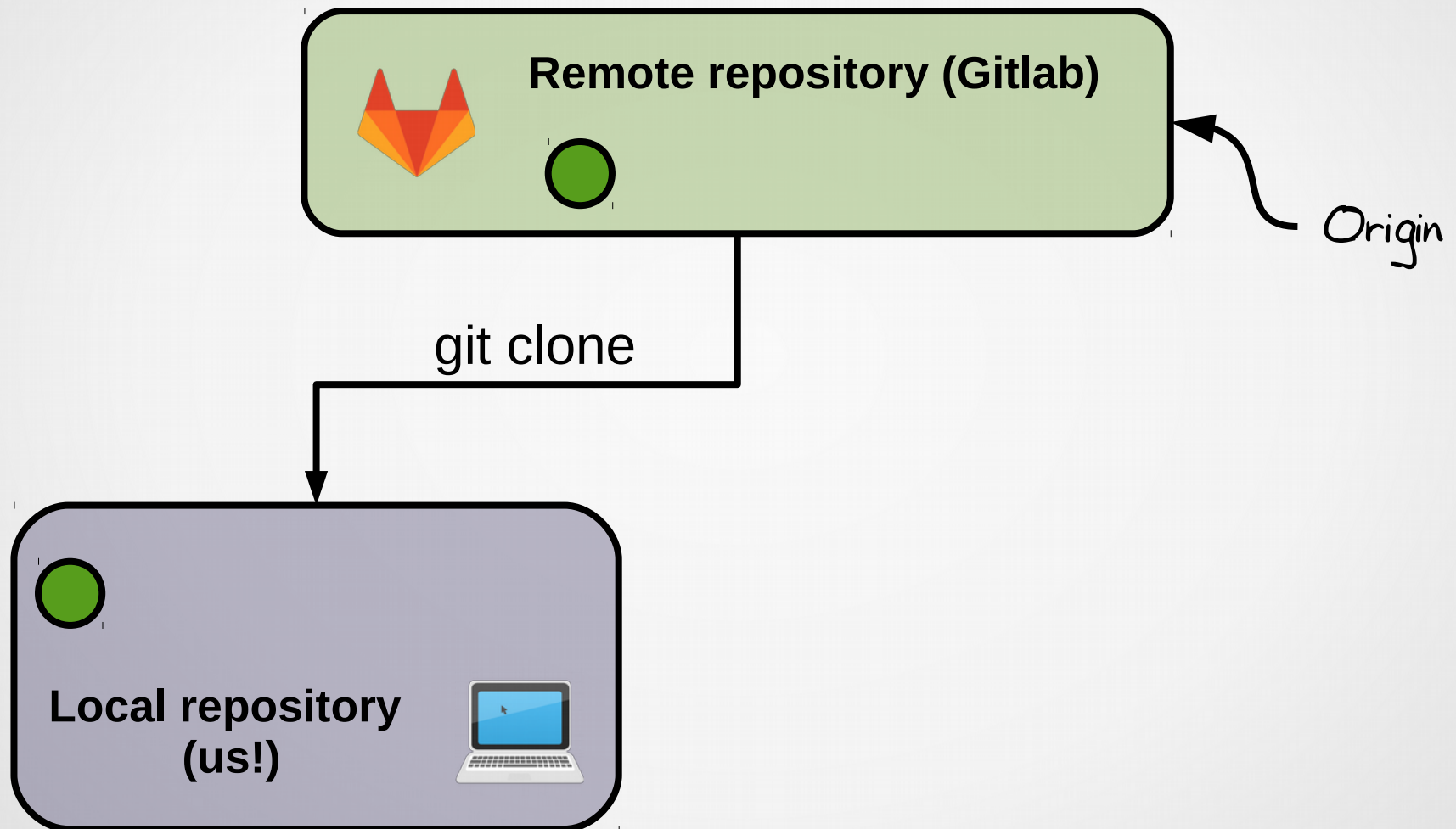
- Delete a branch

`git branch -d <branch_name>`

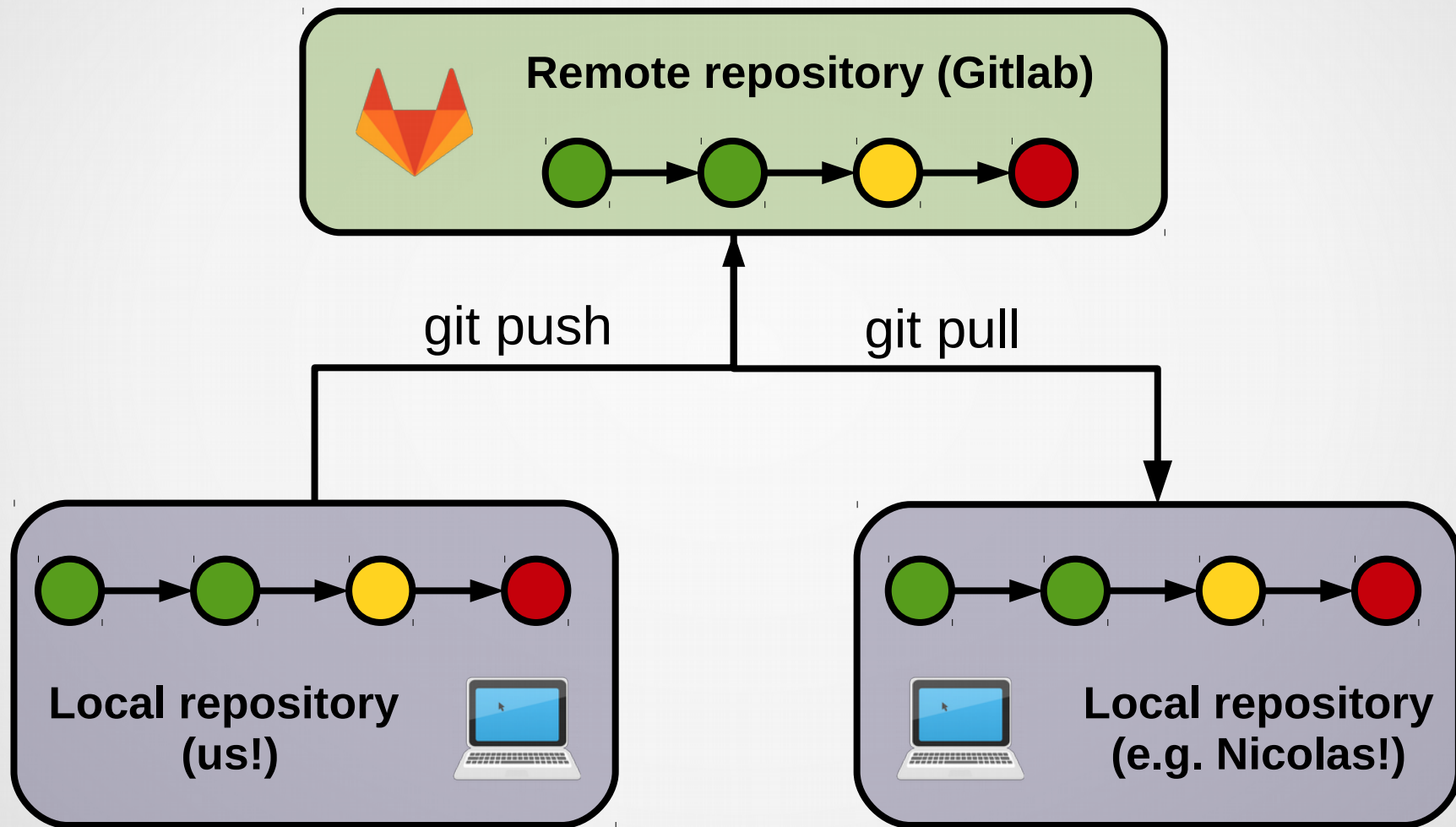
Git remote



Initialization



Synchronization



Remote commands

- Initialization

`git clone < GIT-URL > < local_path >`

- Pushing new commit from local to remote

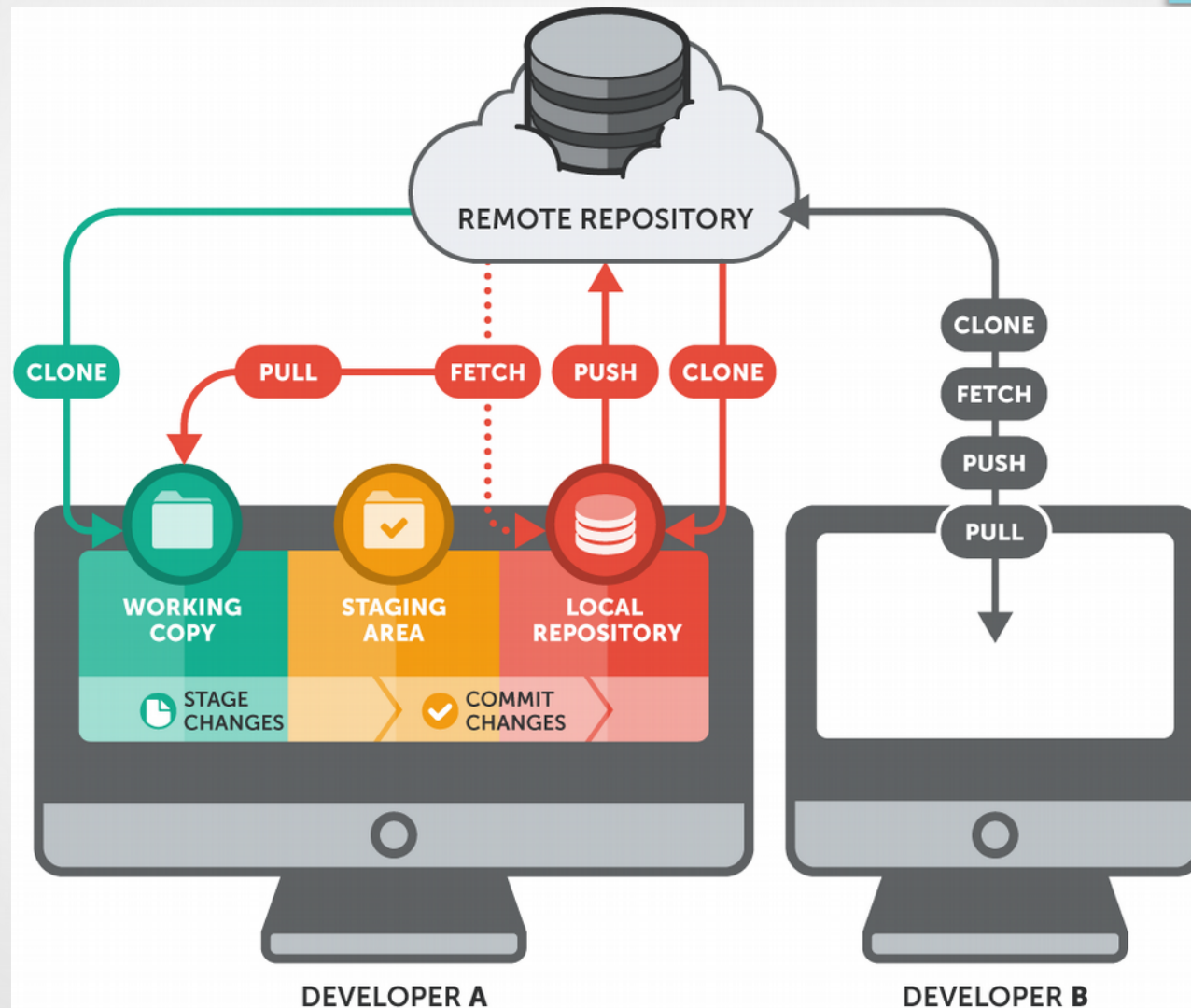
`git push < repository > < branch >`

e.g. `git push origin master`

- Pulling versions from remote to local

`git pull < repository > < branch >`

Summary



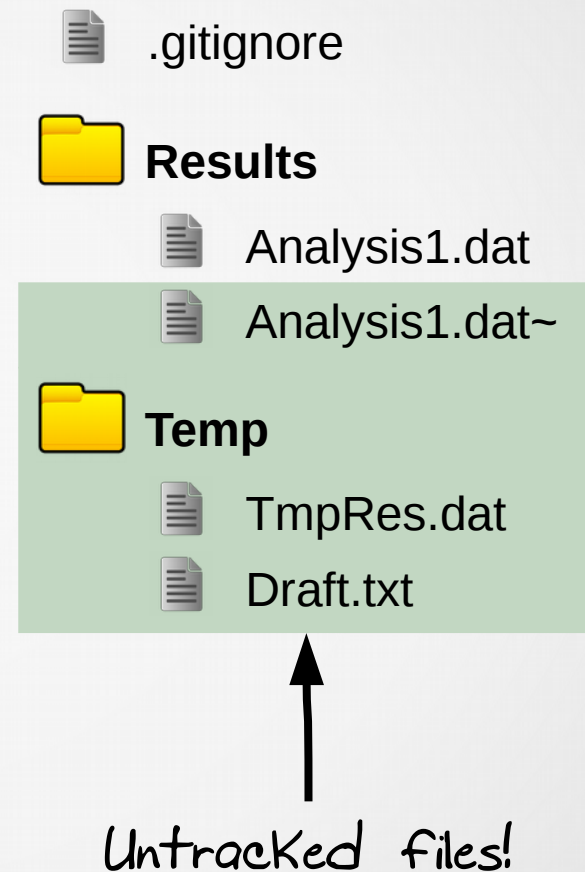
Useful tip #1 : Ignoring files

- Hiding files to GIT
 - Create **.gitignore** file
 - Put filename to hide in it
- Content of .gitignore, e.g

.gitignore

*~

Temp/



Useful tip #2 : Tagging

- Tagging commits
 - Giving a name to a commit
`git tag -a < tag_name > -m "Content of v1."`
 - Listing tags
`git tag`
 - Showing a tag
`git show < tag_name >`
- Sharing tag
`git push origin --tags`

Useful tip #3 : Avoiding bad situations

- Check that you are working on the correct version
 - e.g. branch = master / commit = head
- Check that nobody has updated the global repository
 - Synchronize frequently (git pull / push)
- Be careful when detaching from HEAD
 - Don't modify again this commit
 - Create a new branch from there

`git checkout -b < branch_name > < commit >`



Your turn!

Setting up GIT

- Clone the repository

```
git clone https://gitlab.isb-sib.ch/phylo/labTest.git  
cd labTest/Test
```

- Check the status

```
git status
```

- Create your own branch (with Username=Xavier, e.g.)

```
git checkout -b BranchName
```

- Check the status

Optional: the pandoc script

- Go to : <https://gitlab.isb-sib.ch/phylo/labTest/>
 - Wiki → Archiving and sharing → Git hook for docx
- Follow the steps described on the webpage
- Ask for help if you are lost!

New file

- Create a project folder

```
mkdir ProjectUsername/
```

- Create a file inside it

e.g. : `$~touch ProjectUsername/myFile.txt`

- Stage the file

```
git add ProjectUsername/myFile.txt
```

- Commit the file

```
git commit -m "Added myFile.txt."
```


Update the file

- Update your file (with your favorite editor)

`vim myFile.txt`

- Check the status and differences

`git status`

`git diff`

- Stage and commit the change

`git add -A`

`git commit -m "Updated myFile.txt"`

- Check the commit log

`git log`

Synchronize

- Push your branch

`git push origin BranchUsername`

- Check the effects

Go to <https://gitlab.isb-sib.ch/phylo/labTest>

- Pull your colleague branch

`git pull origin BranchColleague`

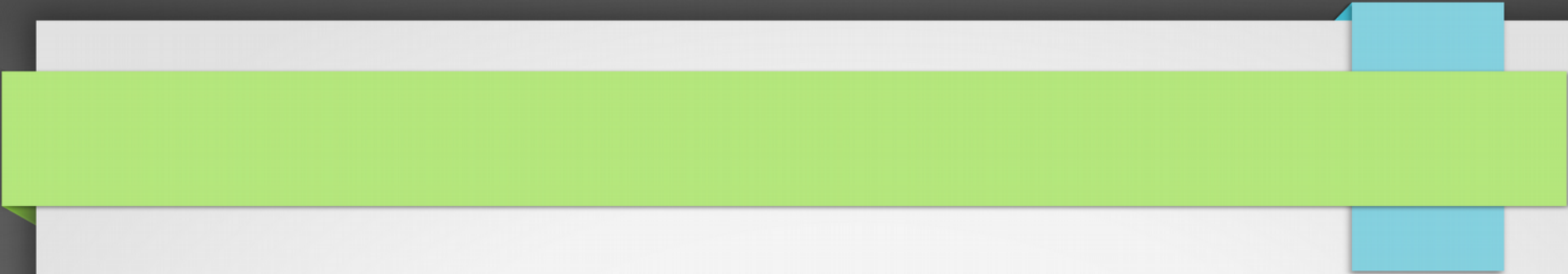
- Check the effects

`git checkout BranchColleague`

`git log --graph --all`

More...

- Try to add a doc file to test your pandoc installation
- Prepare your own GIT repository and project
- Try the .gitignore tips
- Try the git tag options
- Try to create conflicting commits with a colleague



You are now GIT experts!
Congrats!