Version control system

git

Lic. Rodrigo Lugones

rlugones@df.uba.ar

Version control system

git

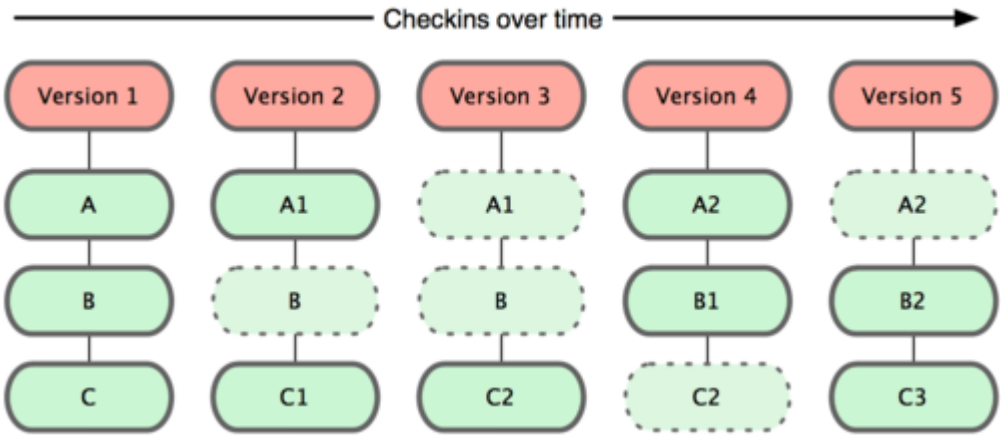
www.git-scm.com

¿Por qué usar un sistema de control de versiones?

Muchas veces nos enfrentamos a esta triste situación:

Program.f90
Program_v0.f90
Program_v0-original.f90
Program_original.f90

¿Por qué usar un sistema de control de versiones?

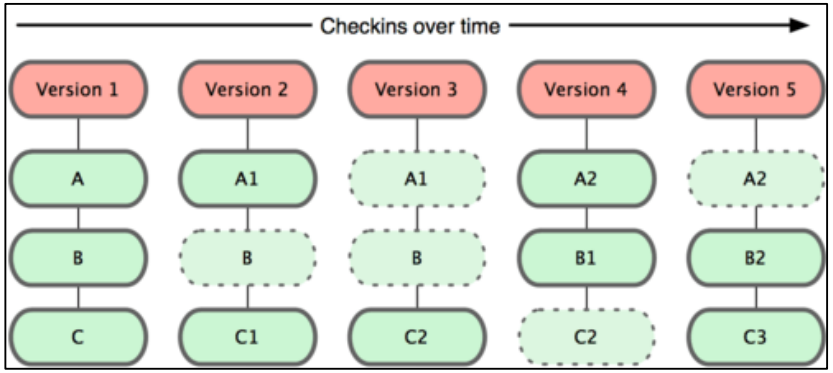


¿Qué querríamos de un sistema de control de versiones?

Backup de snapshots consistentes del proyecto

Documentar cambios

Compartir cambios



Distribuir el desarrollo para muchas personas y/o distintos lugares

Seguimiento de los *bugs* a través de la historia del desarrollo

WTPC
Workshop en técnicas de programación científica

Para todo esto (y más), existen los sistema de control de versiones

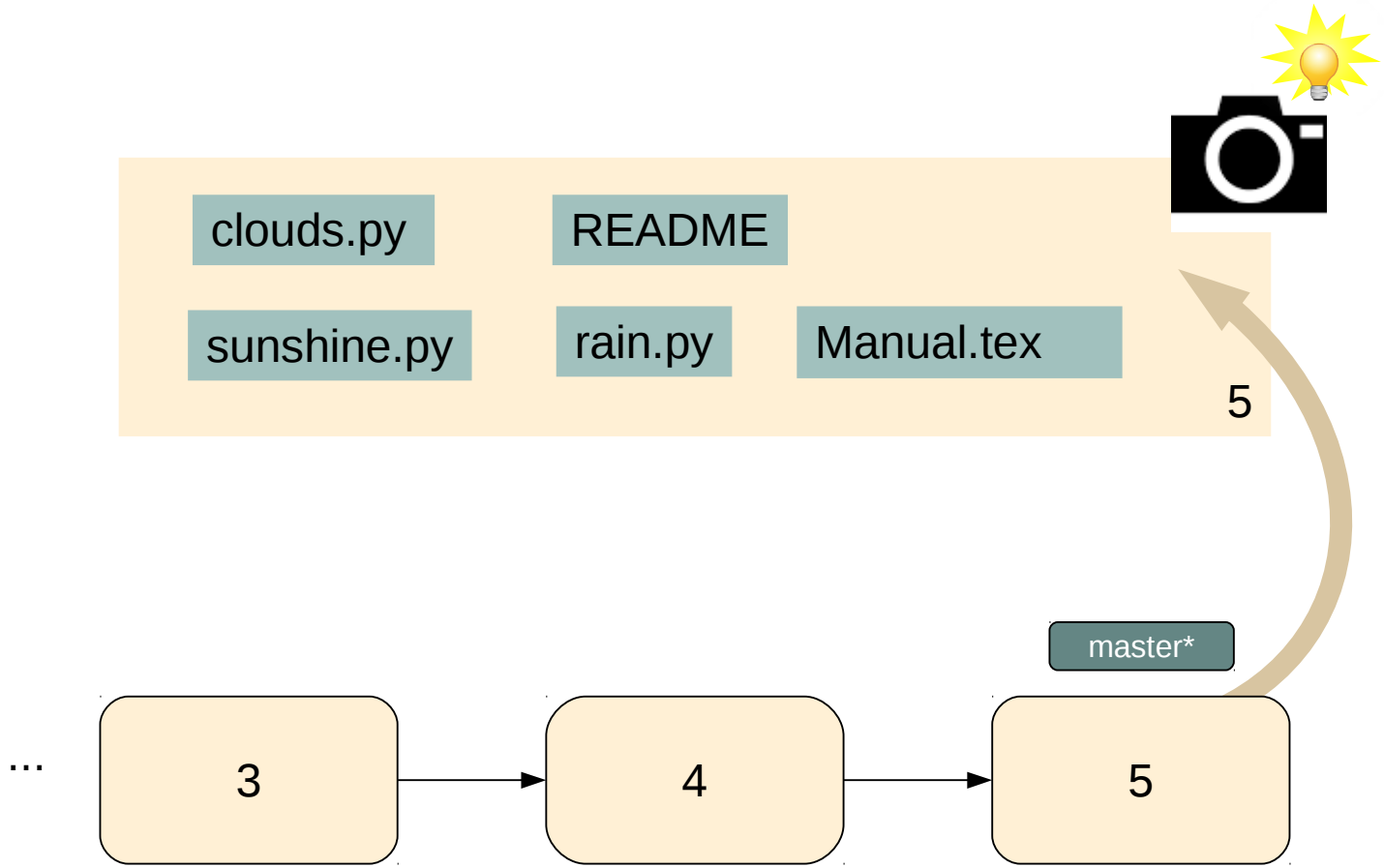
xkcd.com/1597/

¿Sirve para cualquier tipo de archivo?

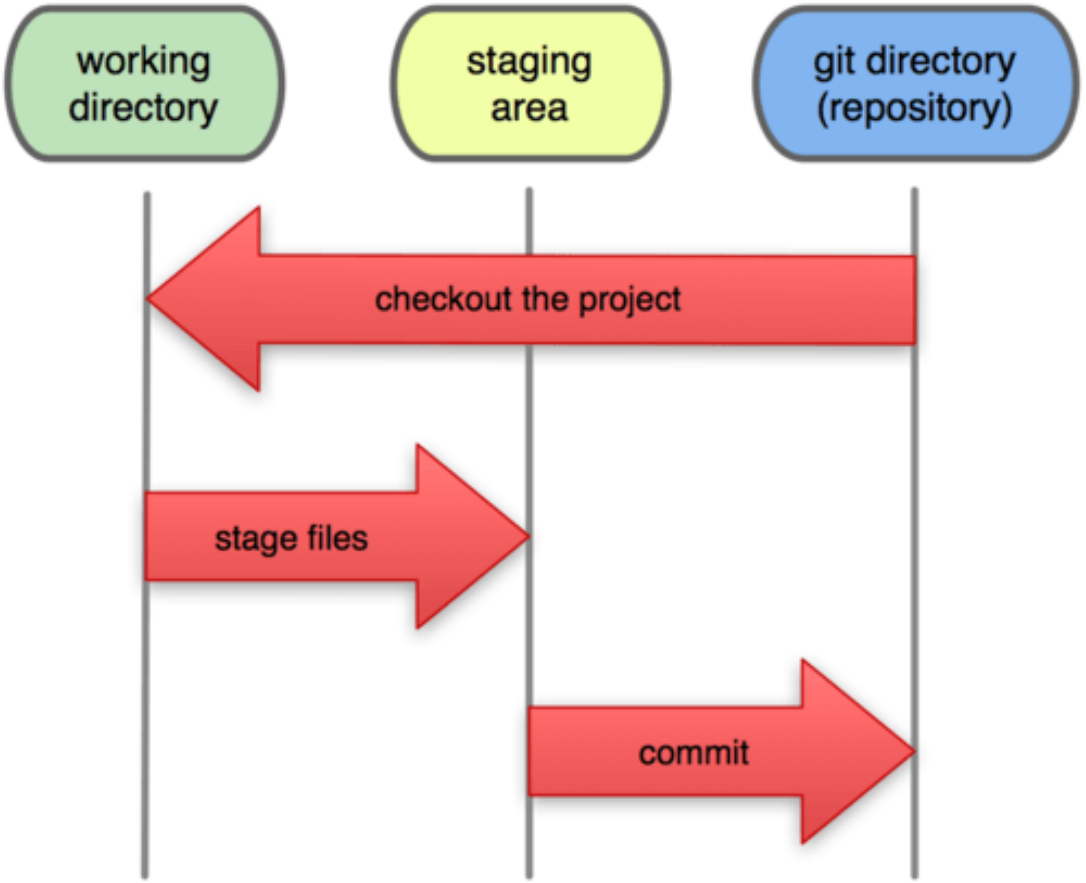
En principio sí, pero no es la idea.

Códigos fuente y cualquier archivo de texto en general son *mergeados* automáticamente.

git detecta cambios. Con archivos que no son texto plano guardará una copia nueva del archivo.


clouds.py
README
sunshine.py
rain.py
Manual.tex
5
master*
...
3
4
5

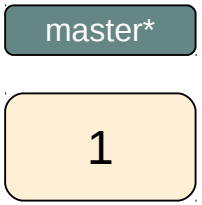
Local Operations

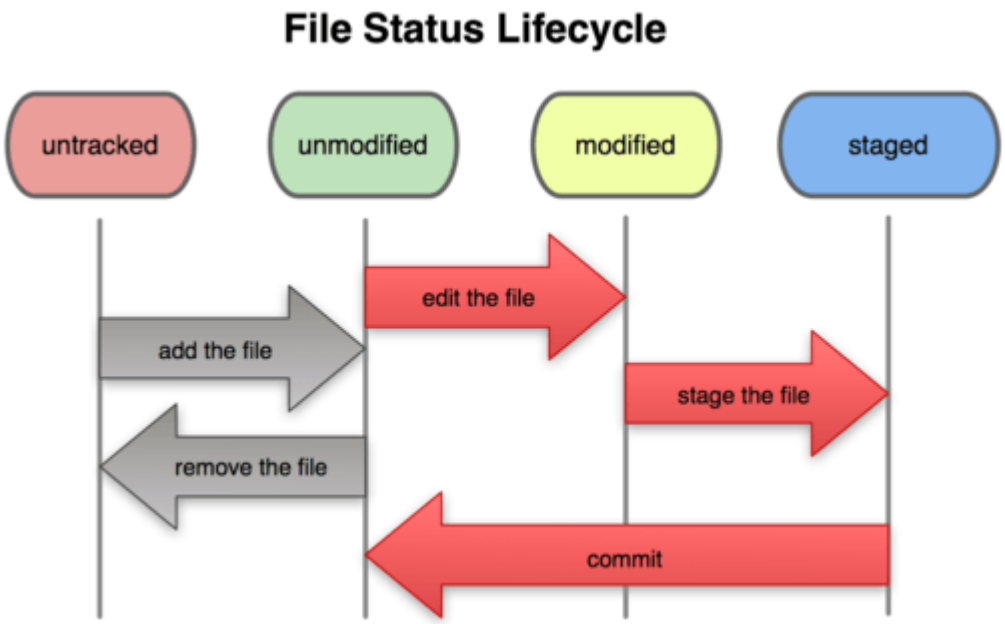


La ardua tarea de crear un repositorio

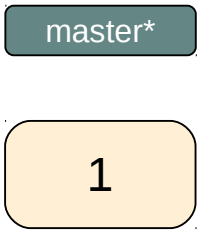
La ardua tarea de crear un repositorio

```
$ git init
```

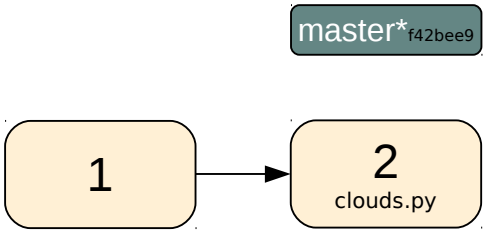



File Status Lifecycle
untracked
unmodified
modified
staged
edit the file
add the file
stage the file
remove the file
commit

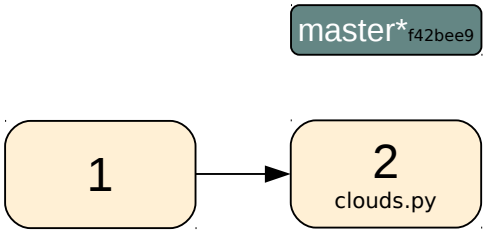
```
$ git status
On branch master

Initial commit

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



```
$ vim clouds.py
$ git add clouds.py
$ git commit -m 'test cases added'
[master (root-commit) f42bee9] test case
 1 file changed, 1 insertion(+)
 create mode 100644 clouds.py
```

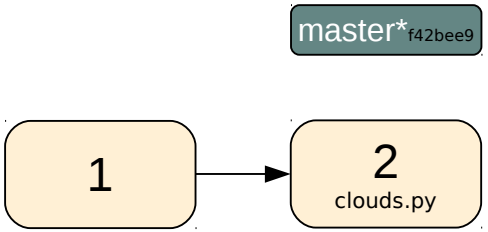


```
$ git log
commit f28ef8b10951a4ecc95dc911bbe7018e3c0225fc
Author: Rodrigo Lugones <rlugones@df.uba.ar>
Date:   Mon Oct 26 12:36:46 2015 -0300

    test cases added
```

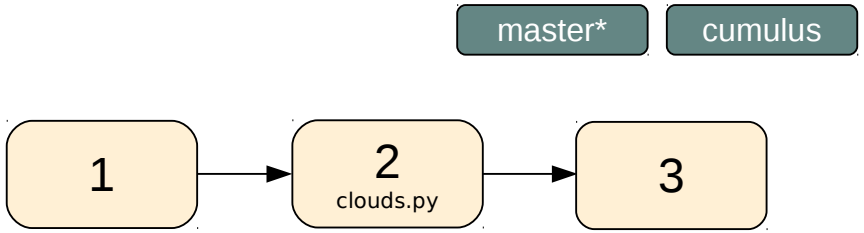


xkcd.com/1296/

```
$ vi wind.py
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

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

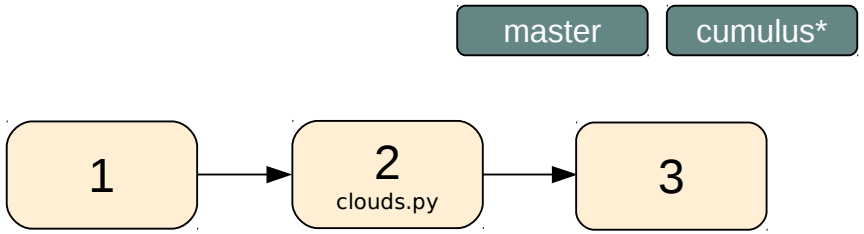


$ git branch cumulus



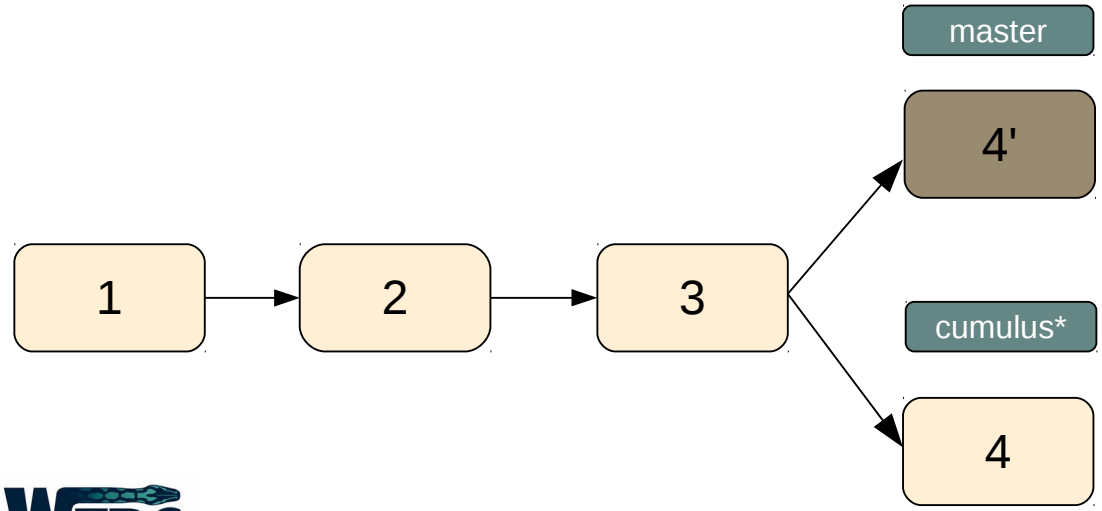
No copia el código
Sólo es un puntero

$ git checkout cumulus



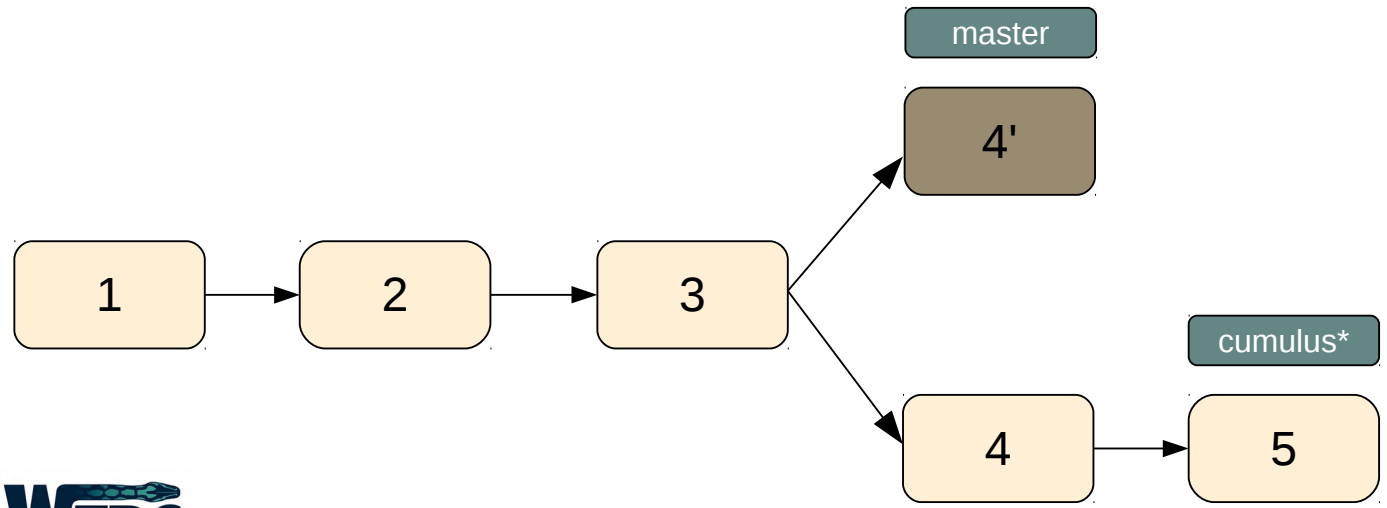
WTPC
workshop en técnicas de programación científica

```
$ vim clouds.py          #y modifico clouds.py
$ git add clouds.py
$ git commit -m 'added parametrization [cumulus]'

$ git commit -m 'added parametrization'
```



WTPC
Workshop en técnicas de programación científica

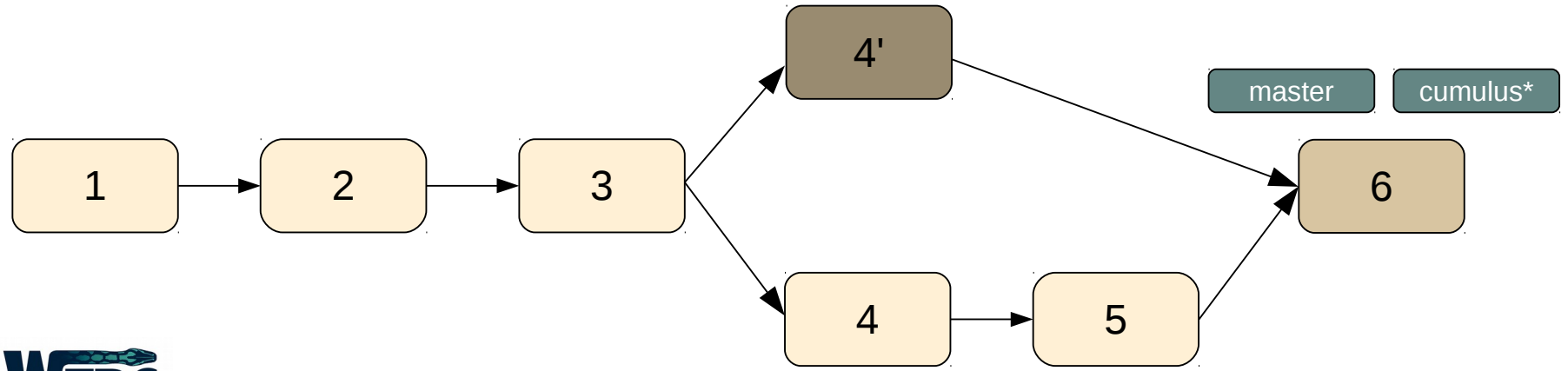
Estoy en la rama cumulus, y la quiero “unir”/*mergear* con la rama master.



WTPC
Workshop en técnicas de programación científica

Estoy en la rama cumulus, y la quiero “unir”/*mergear* con la rama master.

$ git merge cumulus master
o bien
$ git merge master



¿Y si quiero ignorar ciertos archivos?

```
$ cat .gitignore          #Creo (o edito) un archivo llamado .gitignore

# Un comentario. Esta línea es ignorada.
# Ningún archivo .a
*.a

# Pero trackeá lib.a, aunque esté ignorando los archivos .a
!lib.a

# Ignorar el archivo TODO en el directorio de raíz, pero no los subdir/TODO
/TODO

# Ignorar todos los archivos en el directorio build/
build/

# Ignorar los archivos doc/*.txt, pero no los doc/server/arch.txt
doc/*.txt

# Ignorar todos los archivos .txt en el directorio doc/ y los subdirectorios
doc/**/*.txt
```

¿Cómo miro qué cosas cambié respecto del último *commit*?

```
$ git diff
diff --git a/wind.py b/wind.py
index 653ac76..bbade15 100644
--- a/wind.py
+++ b/wind.py
@@ -1,2 +1,6 @@
 the wind

+some code
+
+another some code
+
```

con lo que **no** está en stage

```
$git diff --staged
```

con lo que **sí** está en stage

WTPC
Workshop en técnicas de programación científica

¿Cómo borro archivos?

```
$ rm wind.py
$ git status
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

      deleted:    wind.py

no changes added to commit (use "git add" and/or "git commit -a")
```

No está en el *stage*. Ahora, debería hacer un **git add** seguido de un **git commit**.

¿Cómo borro archivos?

```
$ git rm wind.py
rm 'wind.py'

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

      deleted:    wind.py

$ git commit -m 'remove old wind model'
```

Con esto, me ahorro un paso.

¿Cómo borro archivos?

```
$ git rm wind.py
rm 'wind.py'

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

      deleted:    wind.py

$ git commit -m 'remove old wind model'
```

Con esto, me ahorro un paso.

Flujo de trabajo

Abróchense los cinturones!

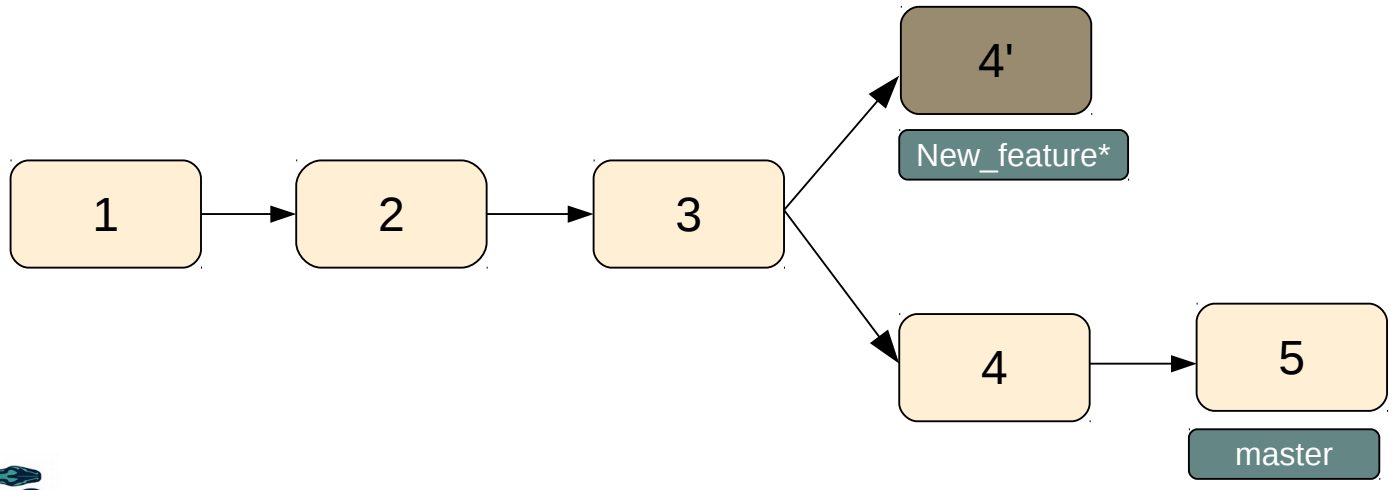
Flujo de trabajo
o
Como pensar el desarrollo del software

WTPC
workshop en técnicas de programación científica

Flujo de trabajo

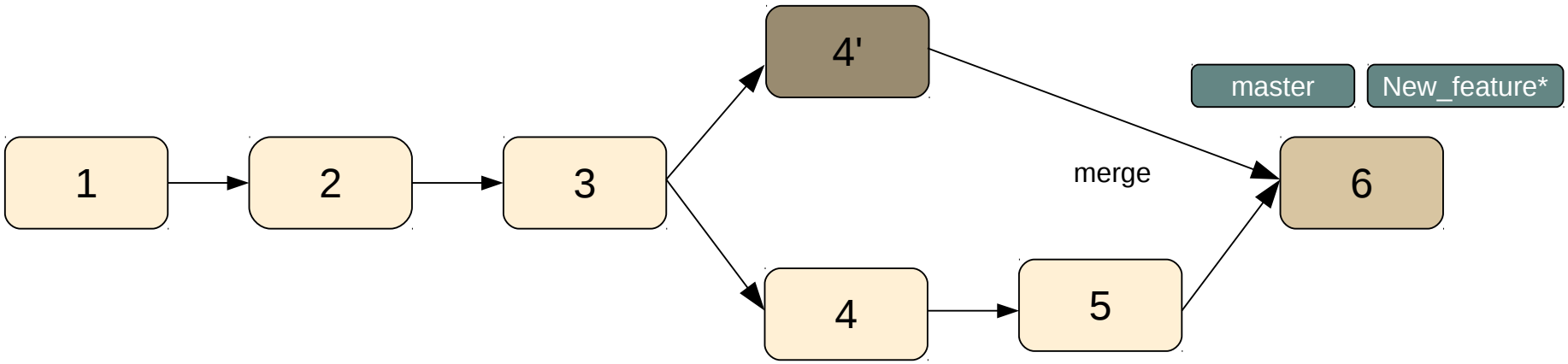
Pensar en el ciclo de vida de la *release*

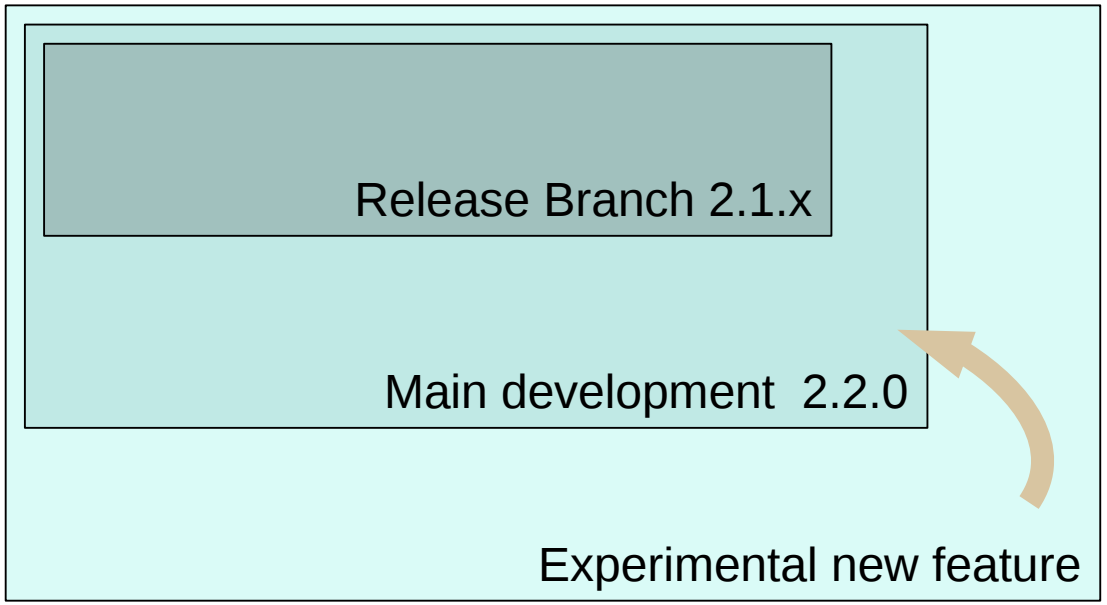
Típicamente se separan las *releases* y las nuevas funcionalidades

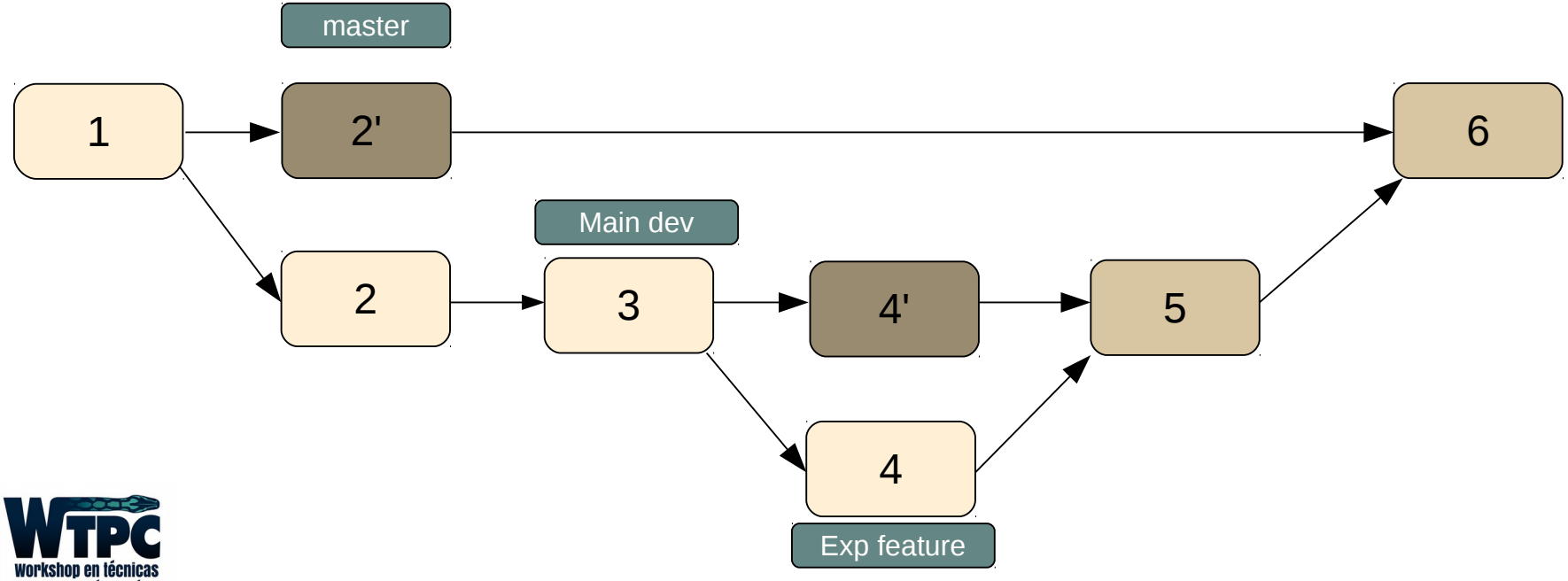


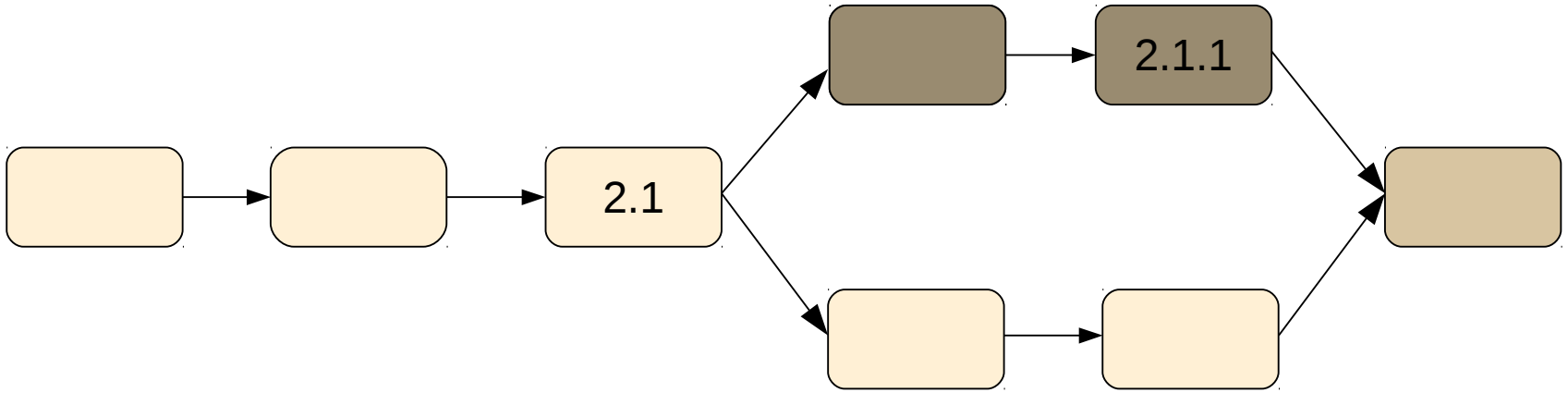
Flujo de trabajo

¡Cuidado con cómo se hacen los *merge*!

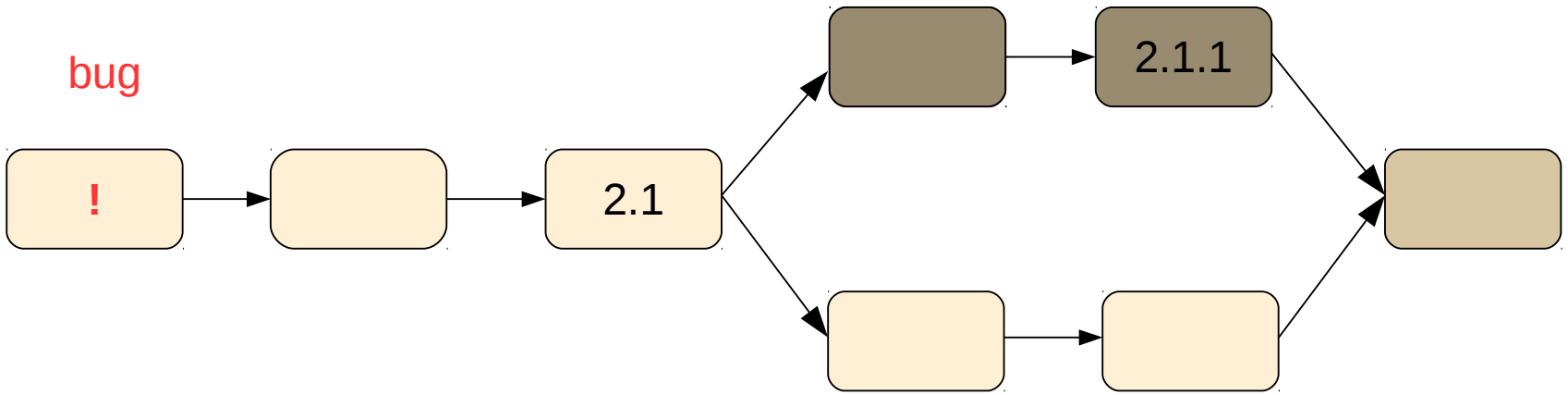



Release Branch 2.1.x
Main development 2.2.0
Experimental new feature

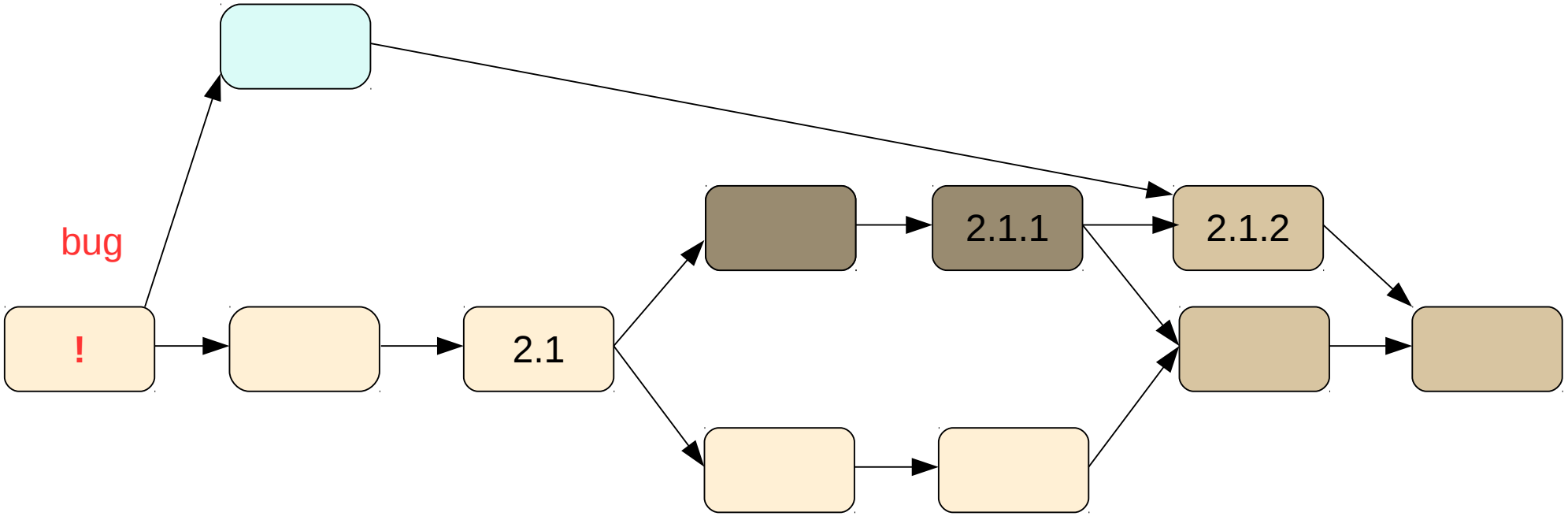

master
1
2'
6
Main dev
2
3
4'
5
4
Exp feature



...2.2


bug
!
2.1
2.1.1
...2.2

Branch nueva para corregir el *bug*



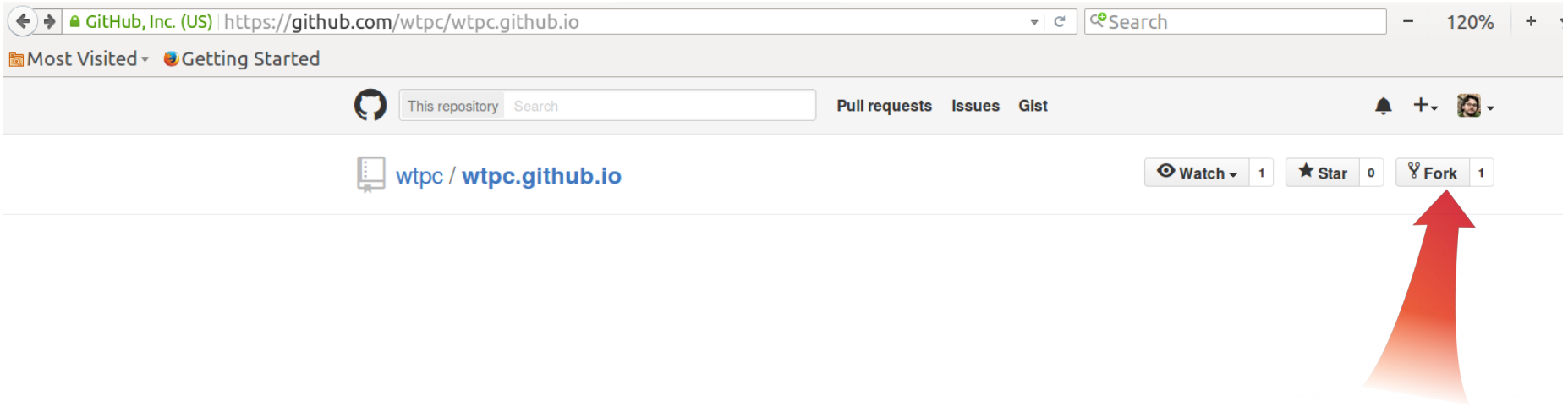
Repaso

- Workspace, staging area, local repository
- init #crea repositorio
- status #estado del repositorio
- add #agrega a *stage*
- commit #guarda en repositorio lo que está en *stage*
- branch #crea nueva rama
- checkout #cambia de rama
- diff #indica diferencias
- log #muestra los *commit* hechos

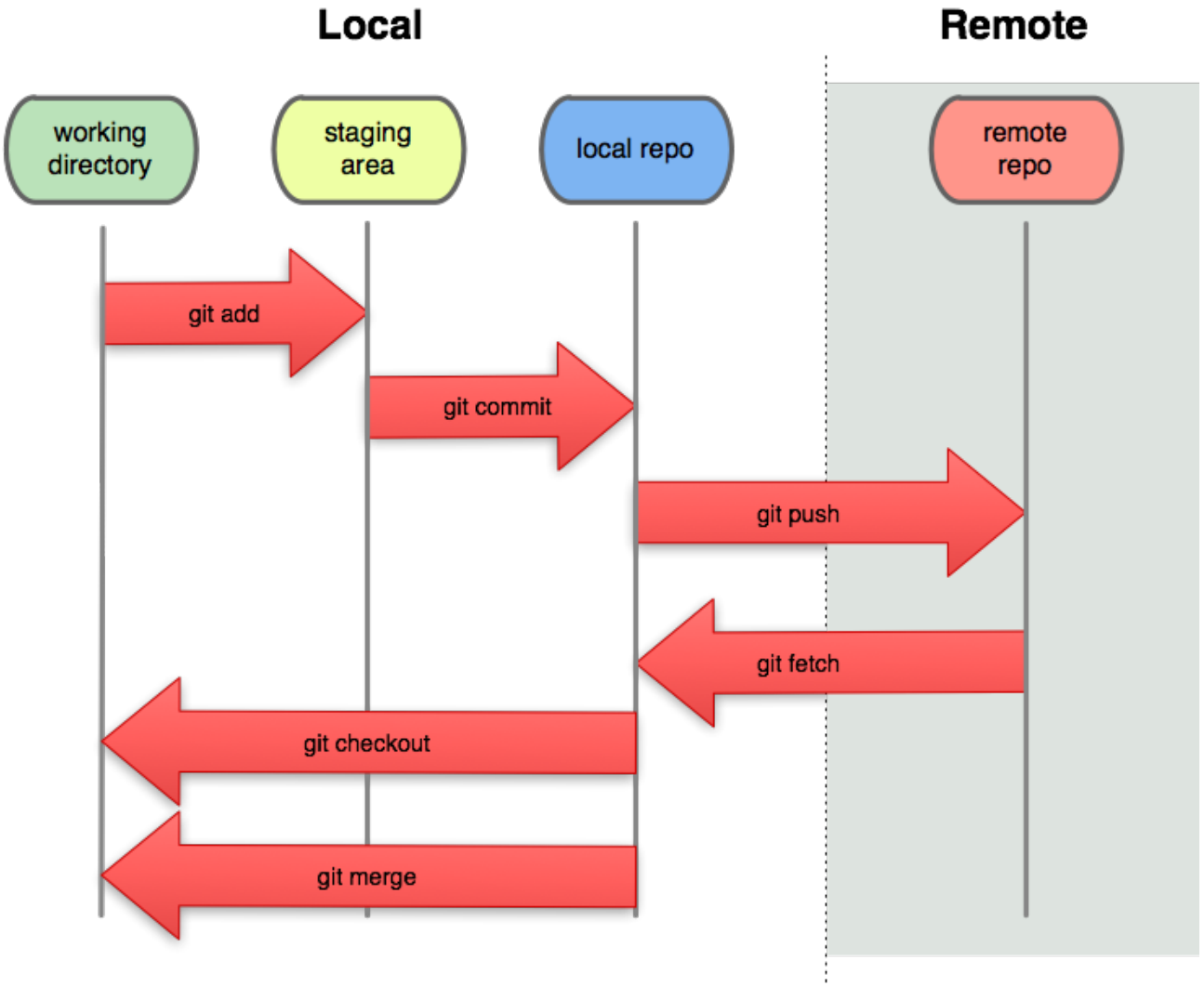
WTPC
Workshop en técnicas de programación científica

Forkear un proyecto

Forkear un proyecto



Trabajando remoto



Trabajando remoto

```
$ git clone https://github.com/<USUARIO>/HOgit.git
```

```
$ git push
```

```
$ git pull
```

git pull

Incorpora cambios de un repositorio remoto en la rama actual. En su implementación por defecto, **git pull** es simplemente **git fetch** seguido de **git merge FETCH_HEAD**.

WTPC
Workshop en técnicas de programación científica

Créditos y referencias

Distributed Version Control David Grellscheid
Workshop on Advanced Techniques for Scientific Programming and Management of Open Source Software Packages
ICPT SAIFR

GIT's hands on
Axel Kohlmeyer
https://sites.google.com/site/akohlmey/

Pagina web de git
https://git-scm.com/book/es/v1/Fundamentos-de-Git

WTPC workshop en técnicas de programación científica

¿Preguntas?