

Taller de Python

Científico

Introducción a git



Departamento de

Ciencias de la Atmósfera y los Océanos

Facultad de Ciencias Exactas y Naturales - Universidad de Buenos Aires

¿Necesito control de versiones?

Comencemos un nuevo proyecto

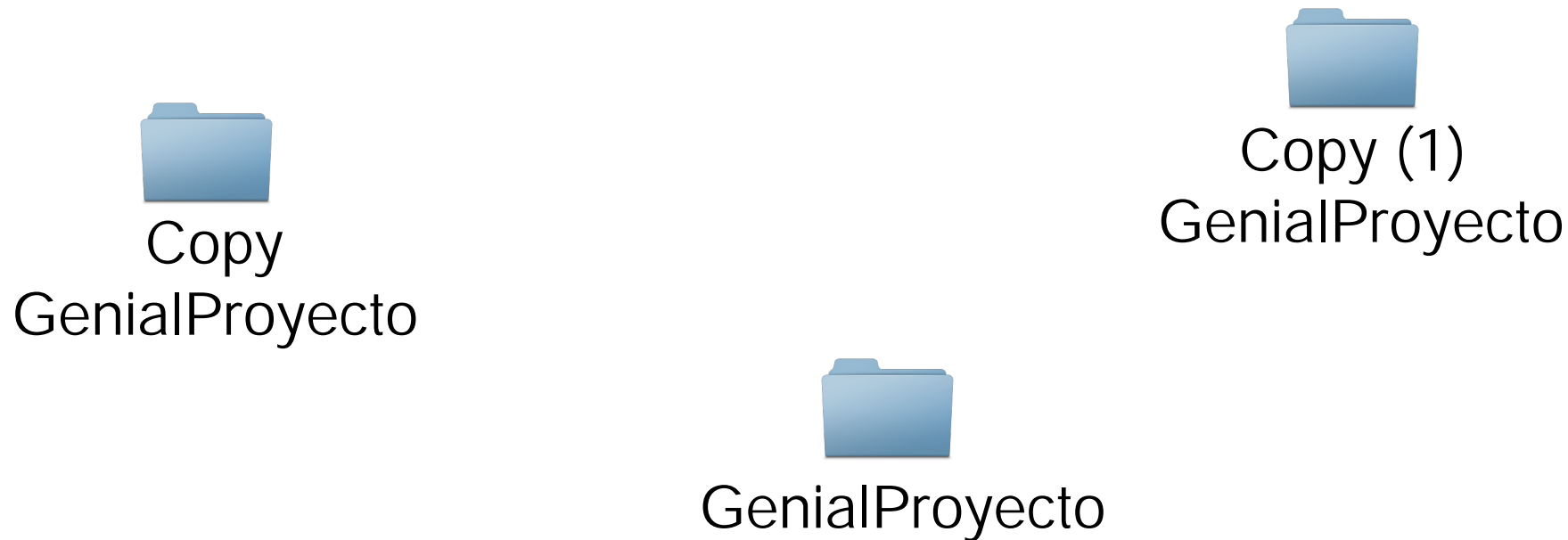


GenialProyecto

¡Todo bien! (por ahora)

¿Necesito control de versiones?

El proyecto va creciendo y necesitamos hacer algunos "back-ups"



Ya comienza a crecer y a complicarse los directorios

¿Necesito control de versiones?

Ya no nos acordamos qué modificación realizamos en cada copia y les ponemos nombre



No parece algo muy simple de mantener...
Crece rápidamente el tamaño de disco usado

¿Necesito control de versiones?

¿Que pasa si desarrollo en la pc del trabajo y quiero sincronizar con mi notebook?

- * Usamos un pendrive (copia completa de todo el proyecto)
- * DropBox (no puede escalar en los integrantes)
- * Trabajamos en una sola computadora

¿Necesito control de versiones?

Y ahora un colega quiere colaborar en el desarrollo de nuestro software

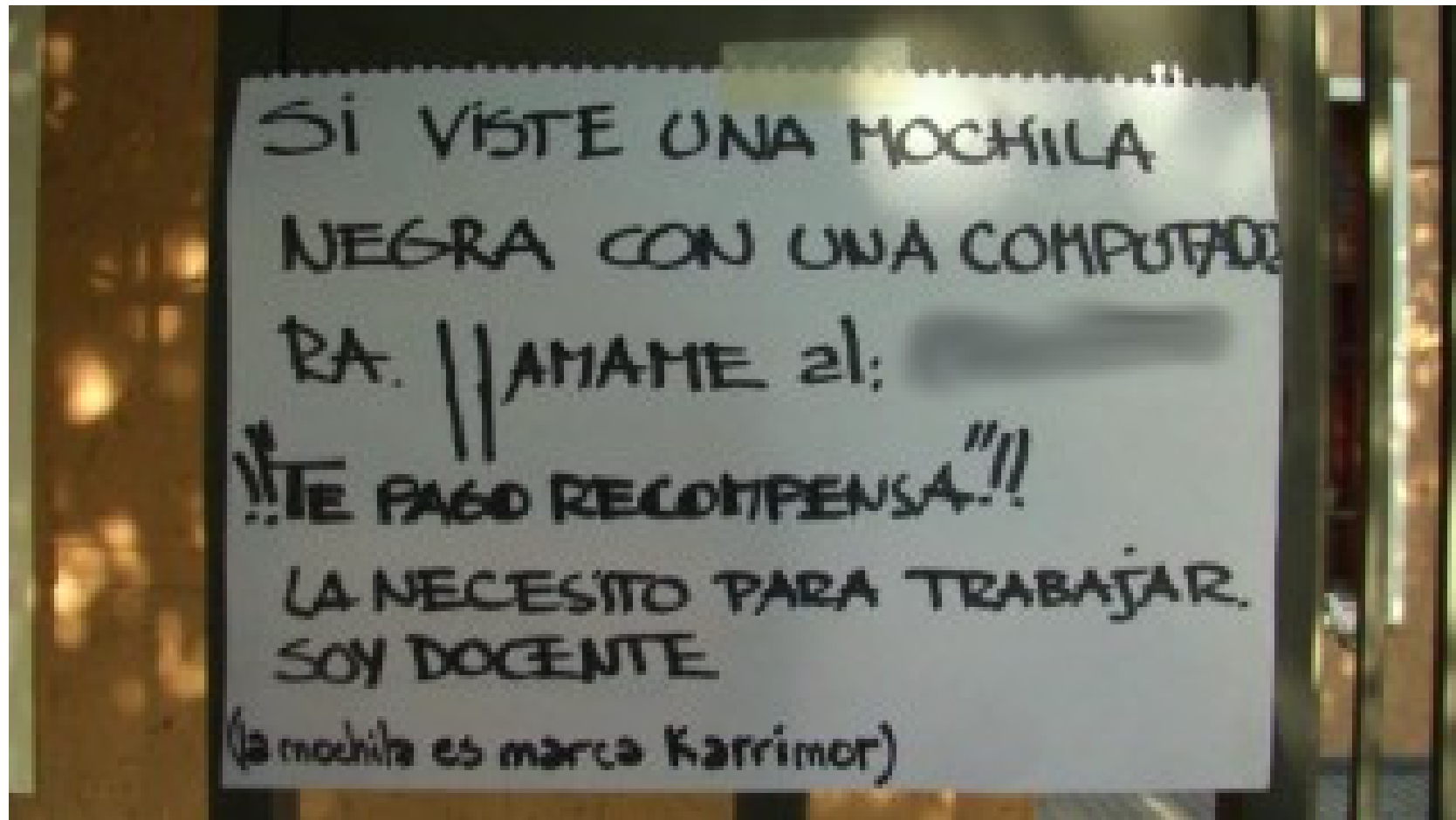
From: <JonSnow@Winterfell.com>
To: Ned <NedStark@Winterfell.com>
Subject: Trabajo

Ned, acá te mando el zip con la última versión (creo) con mi parte del trabajo.

From: <NedStark@Winterfell.com>
To: Jon <JonSnow@Winterfell.com>
Subject: Re: Trabajo

Jon, me olvidé de avisarte que yo ya había modificado el integrador numérico.

¿Necesito control de versiones?



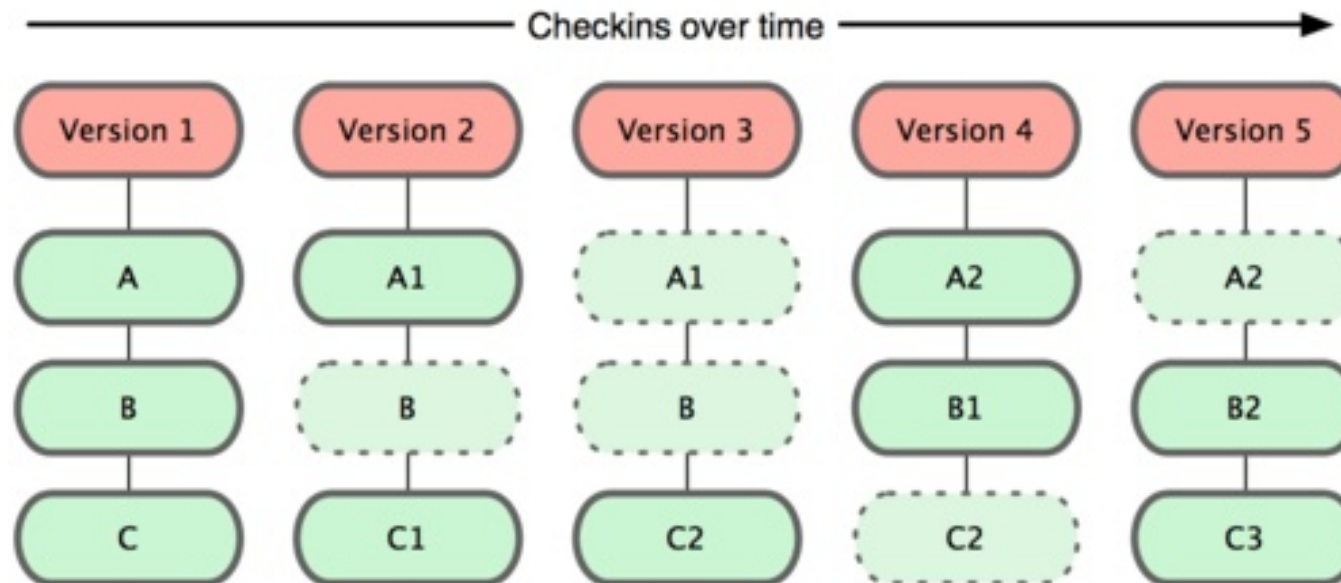
Dos años de trabajo perdido...

¿Qué es Git?

Es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente.

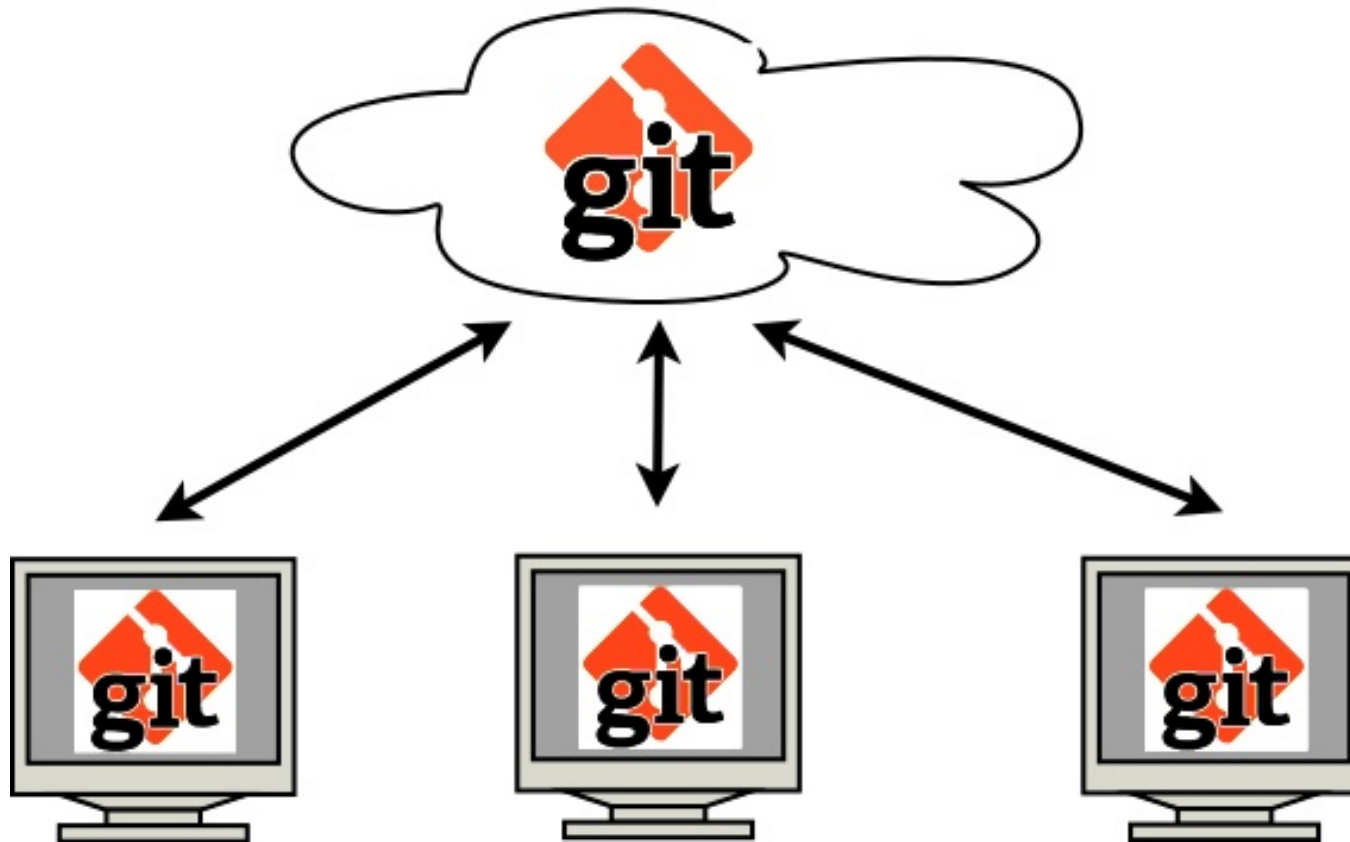
Algunas características de Git

- * Backup de snapshots del proyecto (de cualquier tipo de archivo).
- * Mensaje de las modificaciones.
- * Múltiples autores.
- * Múltiples ramas de trabajo.



Algunas características de Git

- * Distribuido
 - * Todos los usuarios tienen la historia completa.
 - * Múltiples servidores



Algunas características de Git

- * Eficiente forma de almacenar la historia.
- * Log de modificaciones.
- * Rápido.

```
commit 662b951963ce3258b15ad738df6c578ef60b3de2
Author: Franco N. Bellomo <fnbellomo@gmail.com>
Date:   Fri Jan 22 20:10:16 2016 -0300

    update notebook

commit 881901f134ba2a424ca313401be21df888bcd649
Author: Franco N. Bellomo <fnbellomo@gmail.com>
Date:   Fri Jan 22 20:08:32 2016 -0300

    calculo de tau con trampas

commit 16835fce3456c55e090e6f43dd5e457584fcbad3
Author: Lucas Esequiel Bellomo <lbellomo@gmail.com>
Date:   Thu Dec 10 01:55:08 2015 -0300

    add notebook 9 and 10

commit cf830329bc65c1063959cdee68864c2a9f4f060d
Author: Lucas Esequiel Bellomo <lbellomo@gmail.com>
Date:   Thu Dec 10 01:53:01 2015 -0300

    remove data TPP

commit e5190c389cc55a1e626d5c83975e3b64cfa007a5
Author: Lucas Esequiel Bellomo <lbellomo@gmail.com>
Date:   Tue Dec 1 00:47:12 2015 -0300

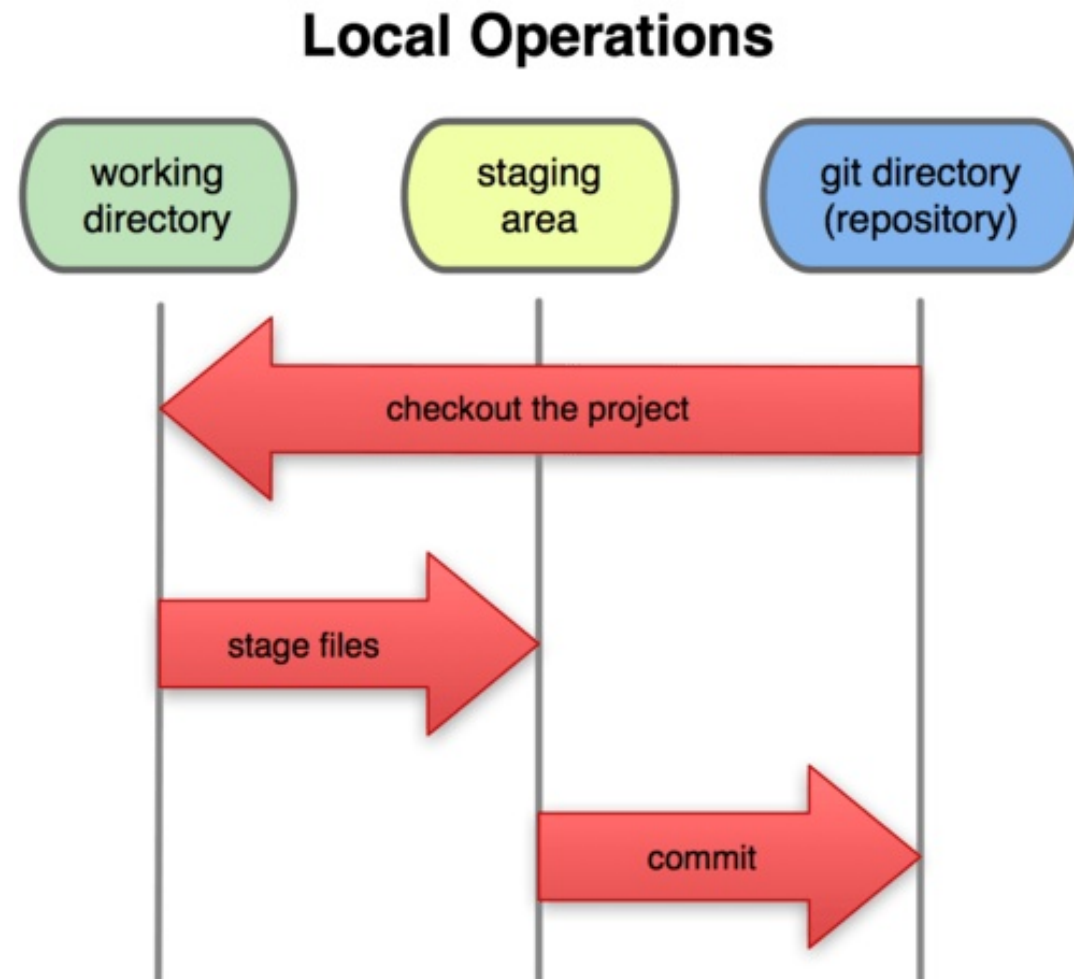
    add speed grafic

:
```

Los tres estados

Git tiene tres estados principales en los que se pueden encontrar los archivos:

- * Confirmado (committed)
- * Modificado (modified)
- * Preparado (staged)



Comencemos

Primero, configuremos git

```
$ git config --global user.name "Jon Snow"  
$ git config --global user.email jonsnow@Winterfell.com  
$ git config --global core.editor emacs
```

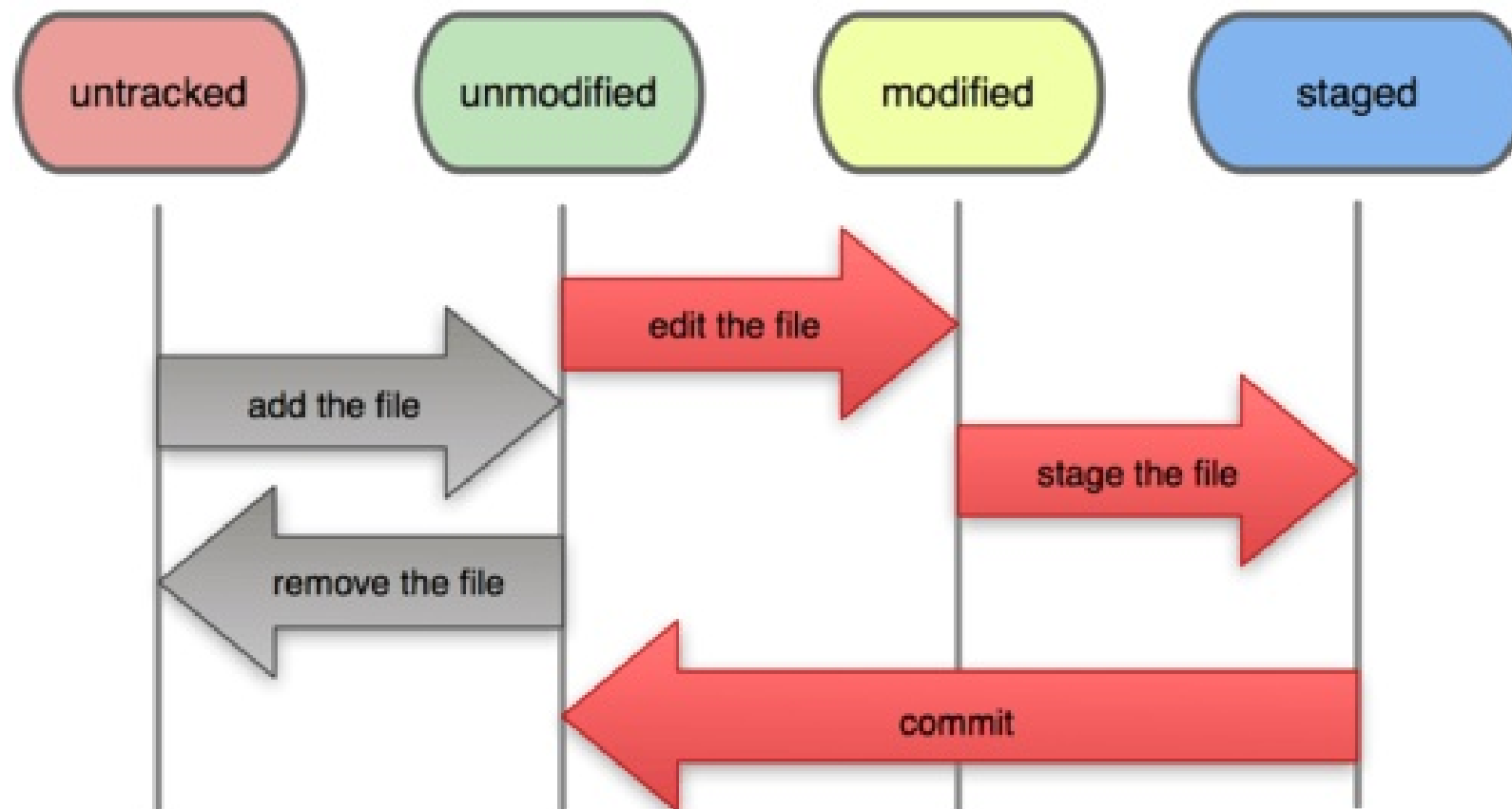
La ardua tarea de crear un repo

```
$ mkdir newProyect  
$ cd newProyect  
$ git init
```

```
$ git status  
On branch master  
Initial commit  
nothing to commit (create/copy files and use "git add" to track)
```

Estado de los archivos

File Status Lifecycle



Agregando archivos

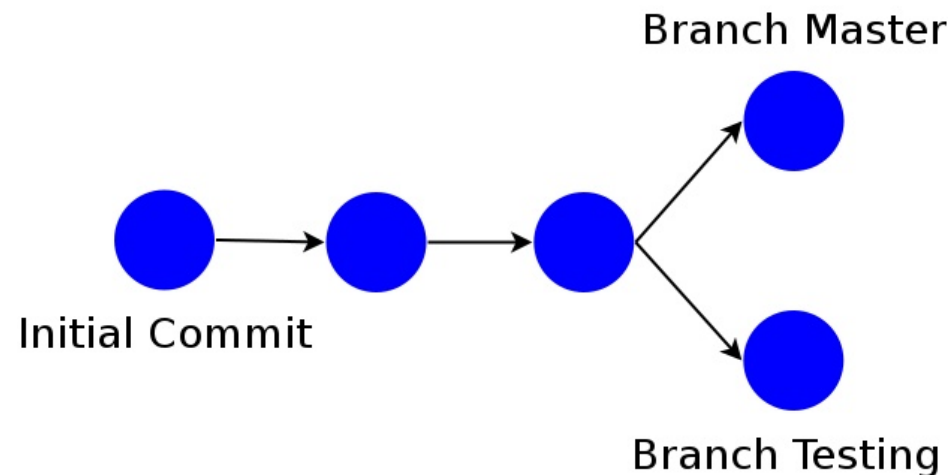
```
$ emacs README.md
$ git add README.md
$ git commit -m 'initial commit'
[master (root-commit) a7d19cd] initial commit
1 file changed, 1 insertion(+)
create mode 100644 README.md
```

```
$ emacs 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)
```


Branch

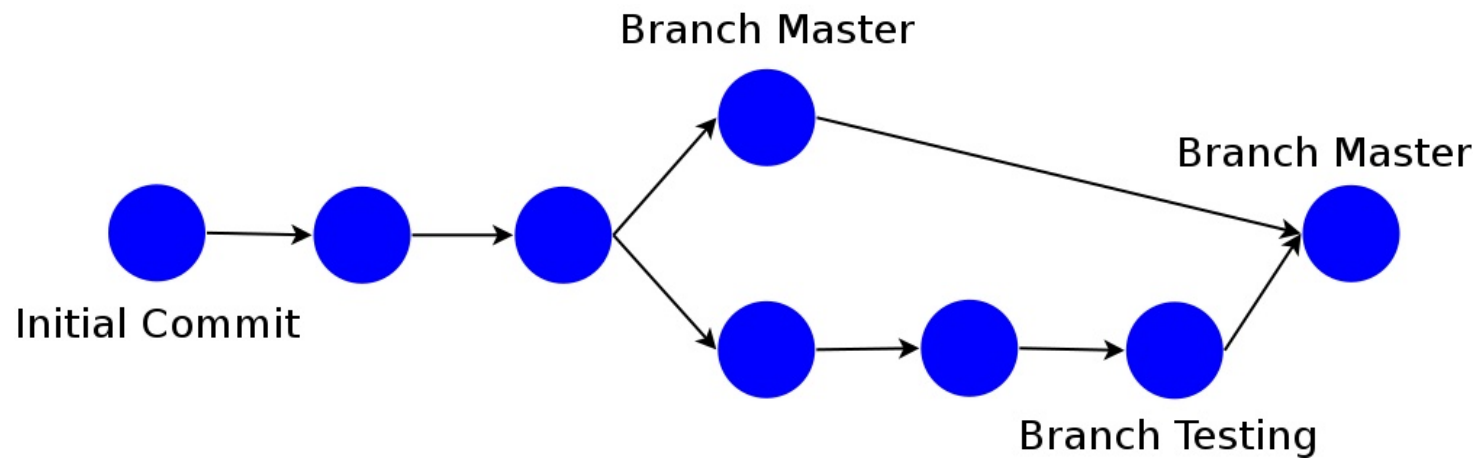
Nuestro repositorio ha crecido bastante y queremos probar algunas modificaciones al código, pero no queremos romper lo que ya está andando.

```
$ git checkout -b test  
Switched to a new branch 'test'  
$ ls  
README.md  wind.py
```



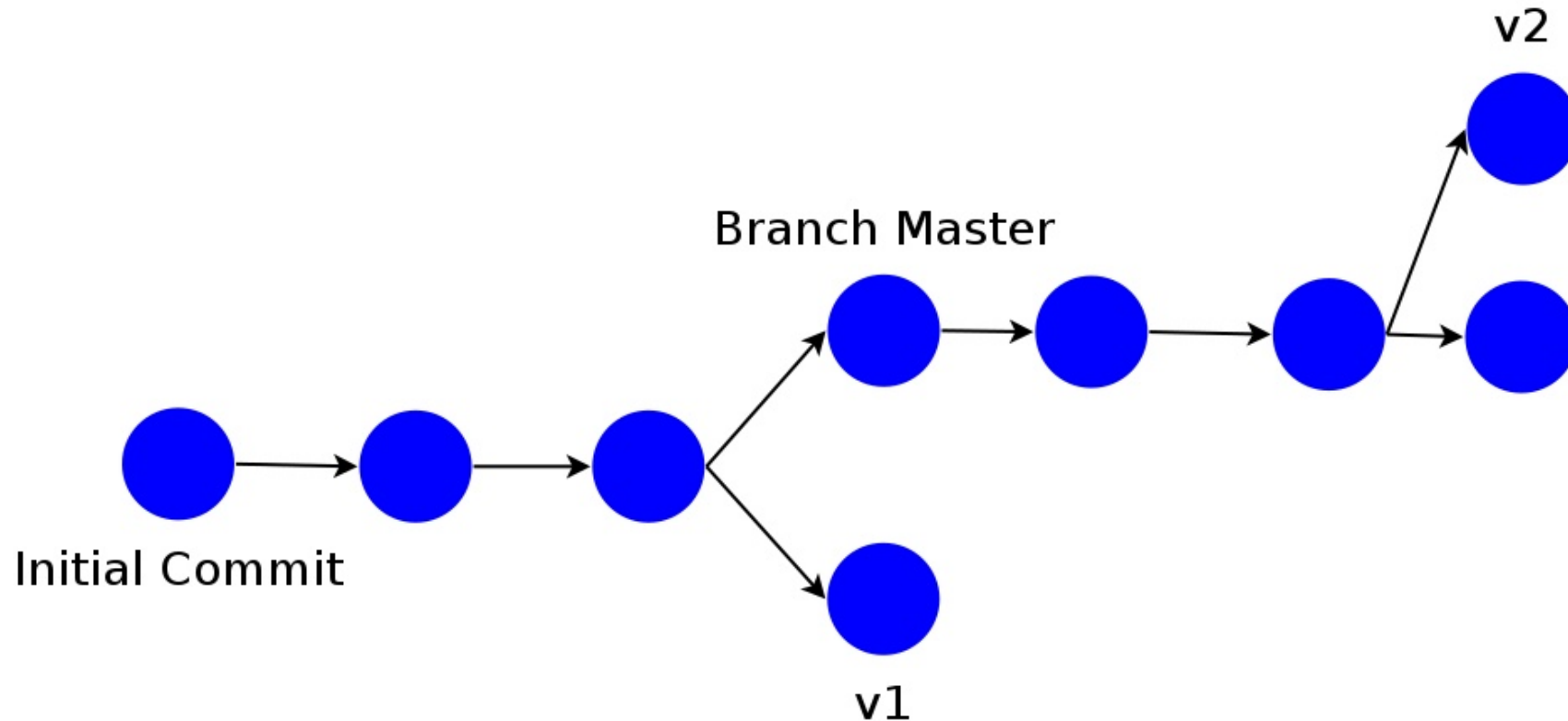
Merge

```
$ git checkout master  
Switched to branch 'master'  
$ git merge test
```

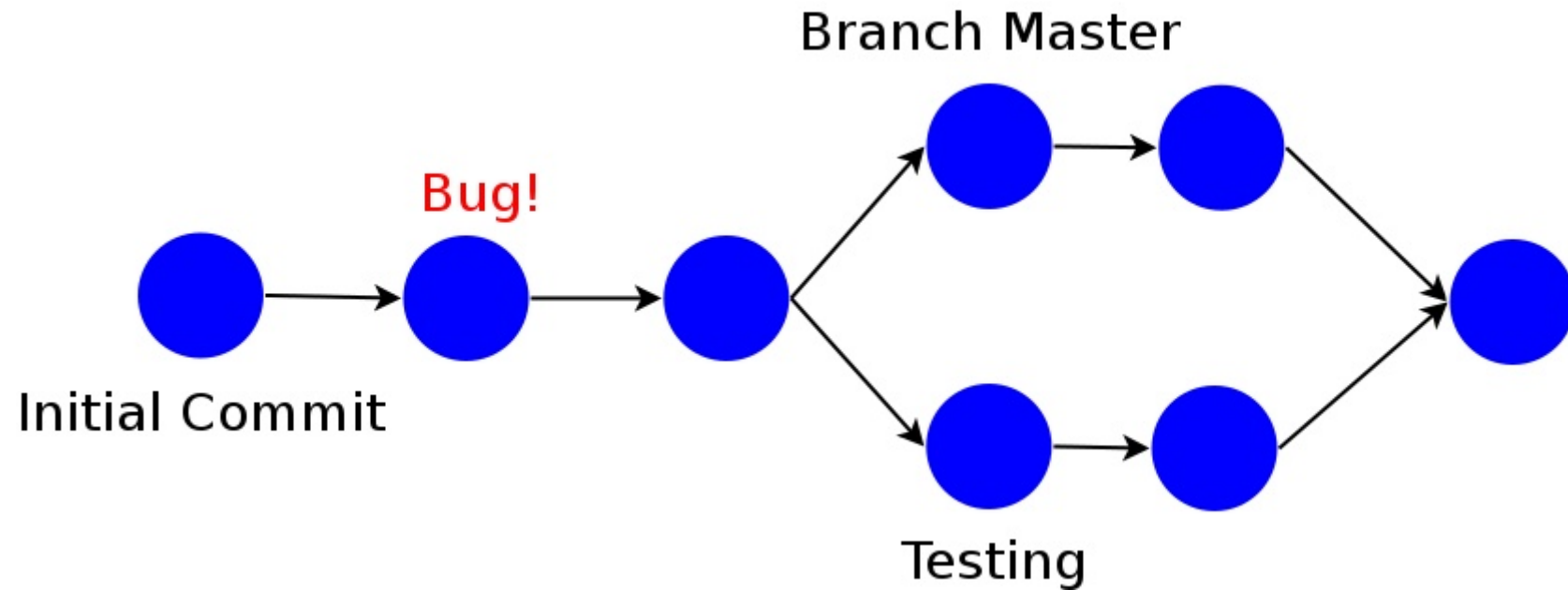


Buena práctica

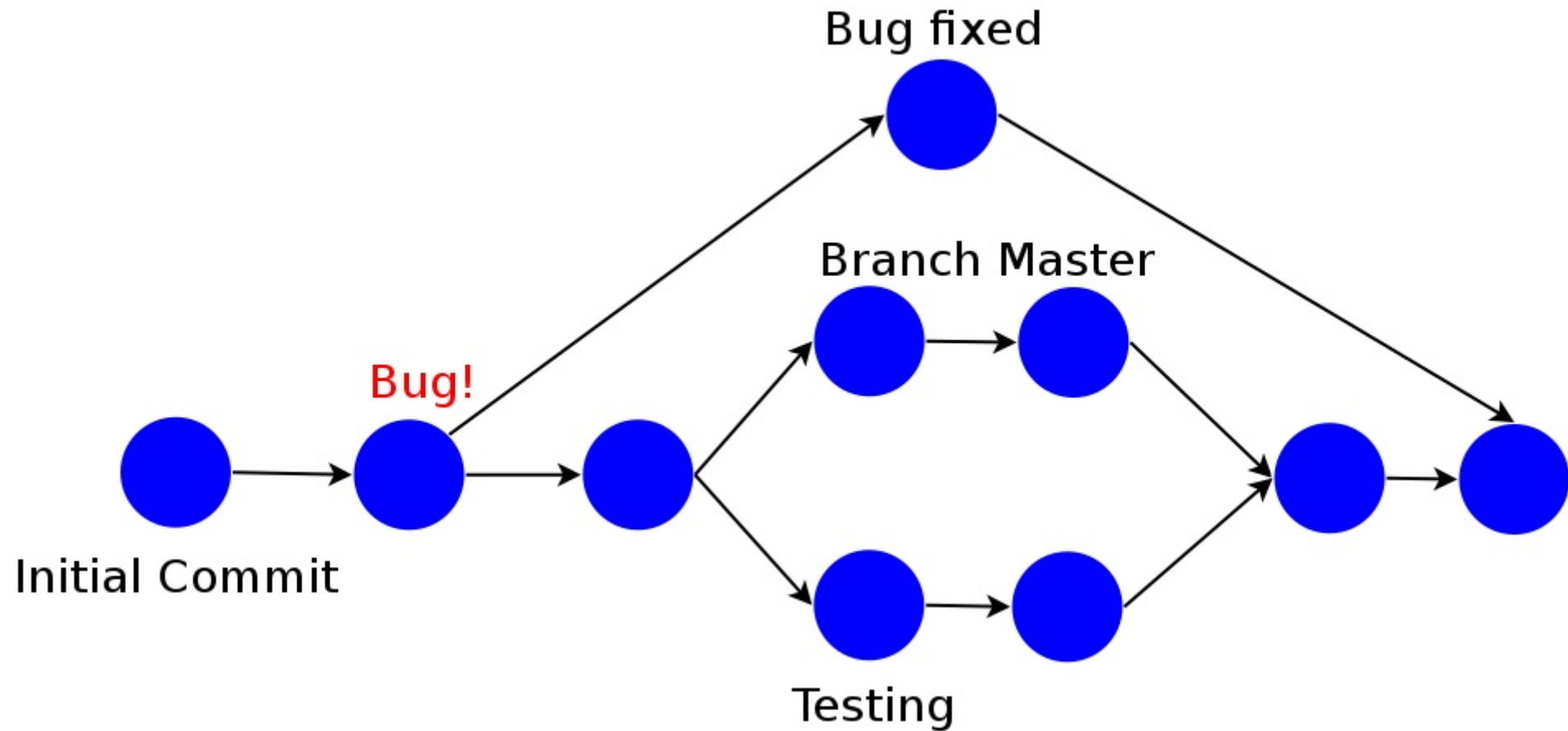
Es una buena práctica dejar las versiones release del código en un branch propio.



Arreglando la historia



Arreglando la historia



Fork

GitHub repository page for **pewen / UVLM**. The **Fork** button is circled in red. The repository has 50 commits, 2 branches, 0 releases, and 2 contributors. The file list shows UVLM, dat, .gitignore, and README.md.

Repository: **pewen / UVLM**

Buttons: Unwatch (3), Star (0), **Fork (0)**

Navigation: Code, Issues (0), Pull requests (0), Wiki, Pulse, Graphs, Settings

Description: No description or website provided. — Edit

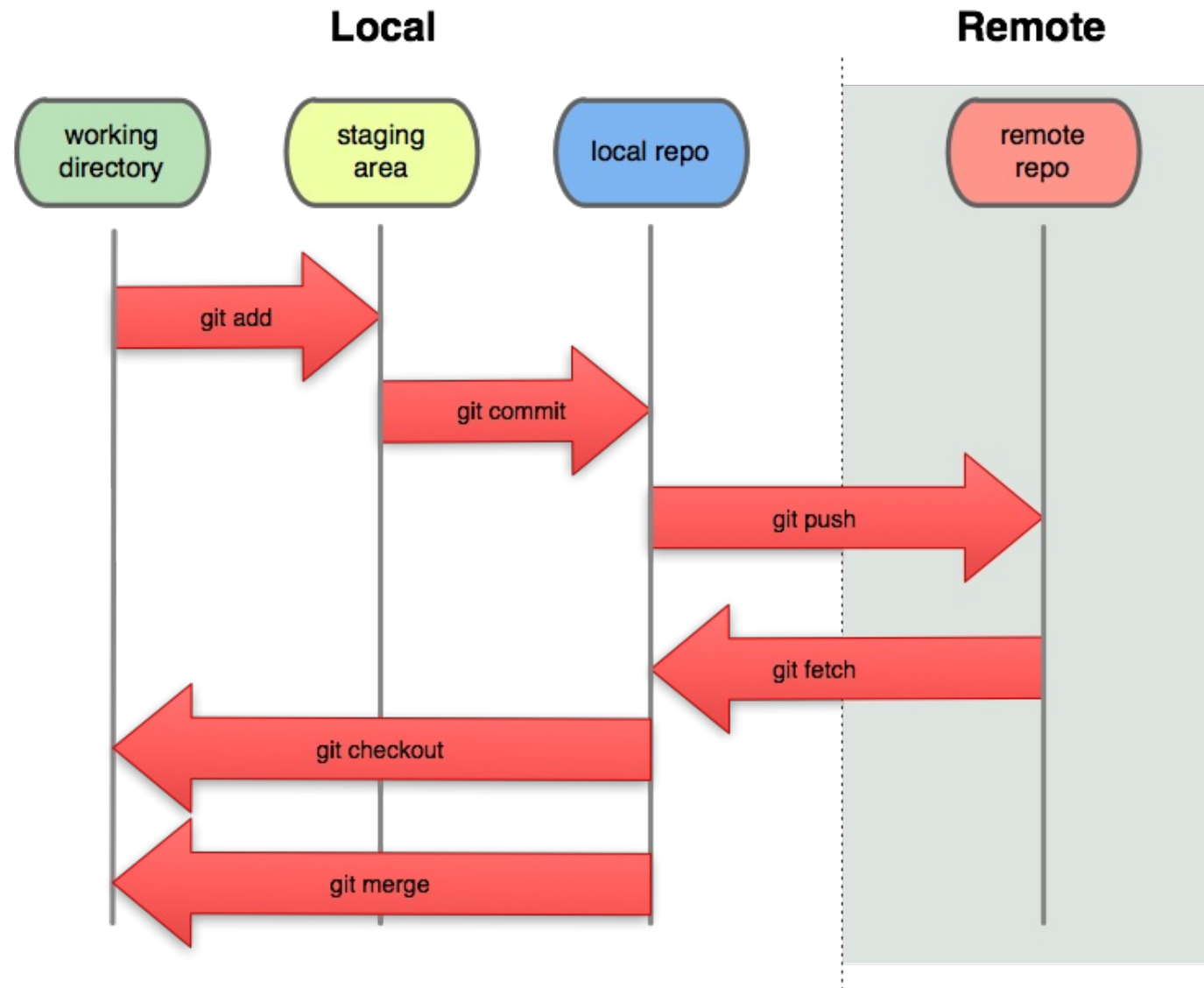
Statistics: 50 commits, 2 branches, 0 releases, 2 contributors

Actions: Branch: master, New pull request, Create new file, Upload files, Find file, Clone or download

Commit history:

| Commit | Message | Time |
|----------------|--|---------------------------------|
| franco bellomo | join Convect_I and Convect_II in Convect | Latest commit 3235ba5 on May 27 |
| UVLM | join Convect_I and Convect_II in Convect | 2 months ago |
| dat | reordenando estructura de dir | 4 months ago |
| .gitignore | se ignora ./bin | 4 months ago |
| README.md | first commit | a year ago |

Trabajando remotamente



Comandos

```
$ git clone https://github.com/numpy/numpy.git
```

```
$ git push
```

```
$ git pull
```


Referencias

Documentación oficial de Git

<https://git-scm.com/book/es/v1>

Servicios gratuitos

<https://github.com>

<https://gitlab.com>

<https://bitbucket.org>