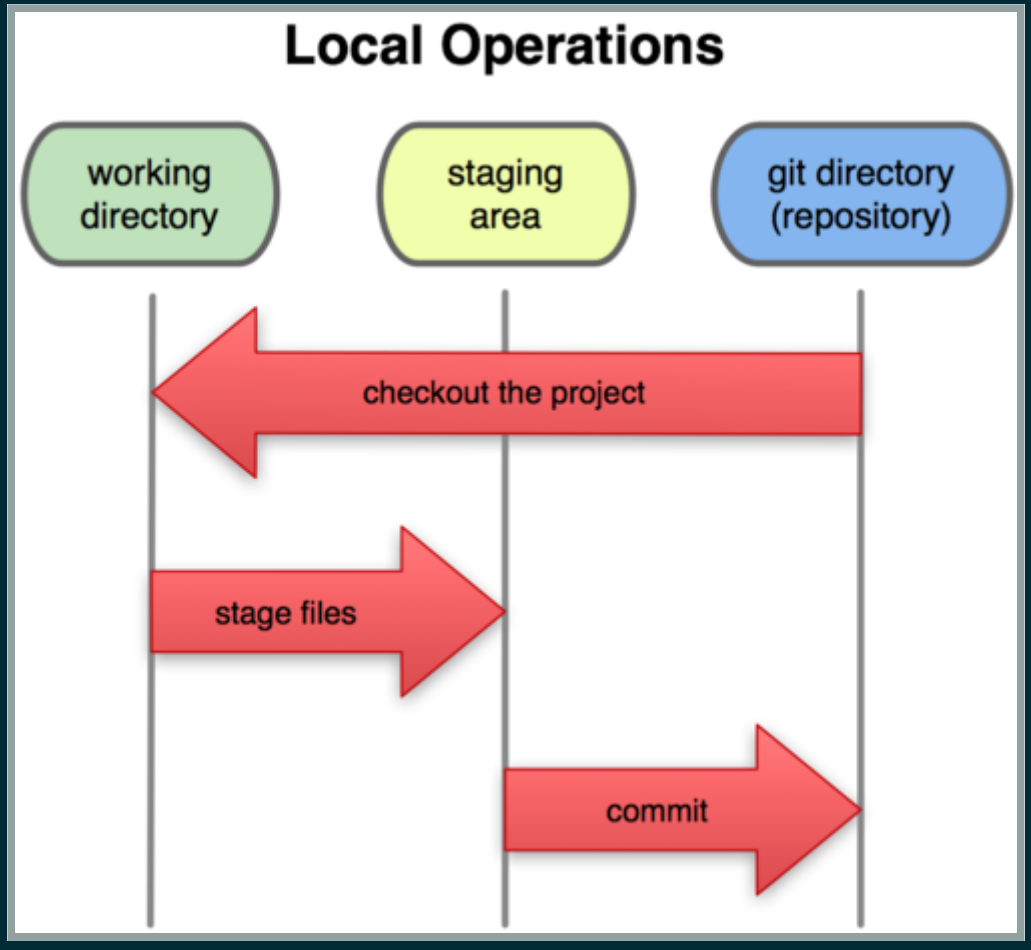
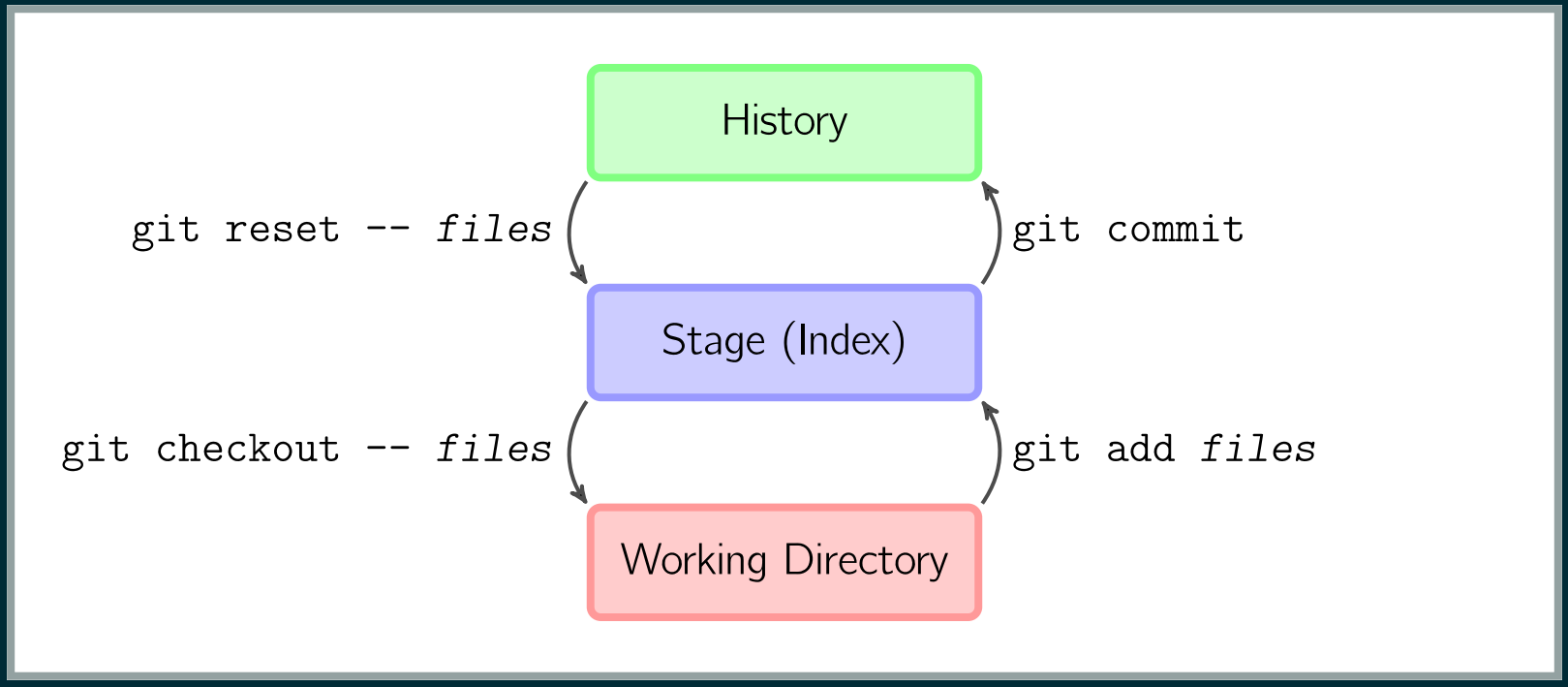
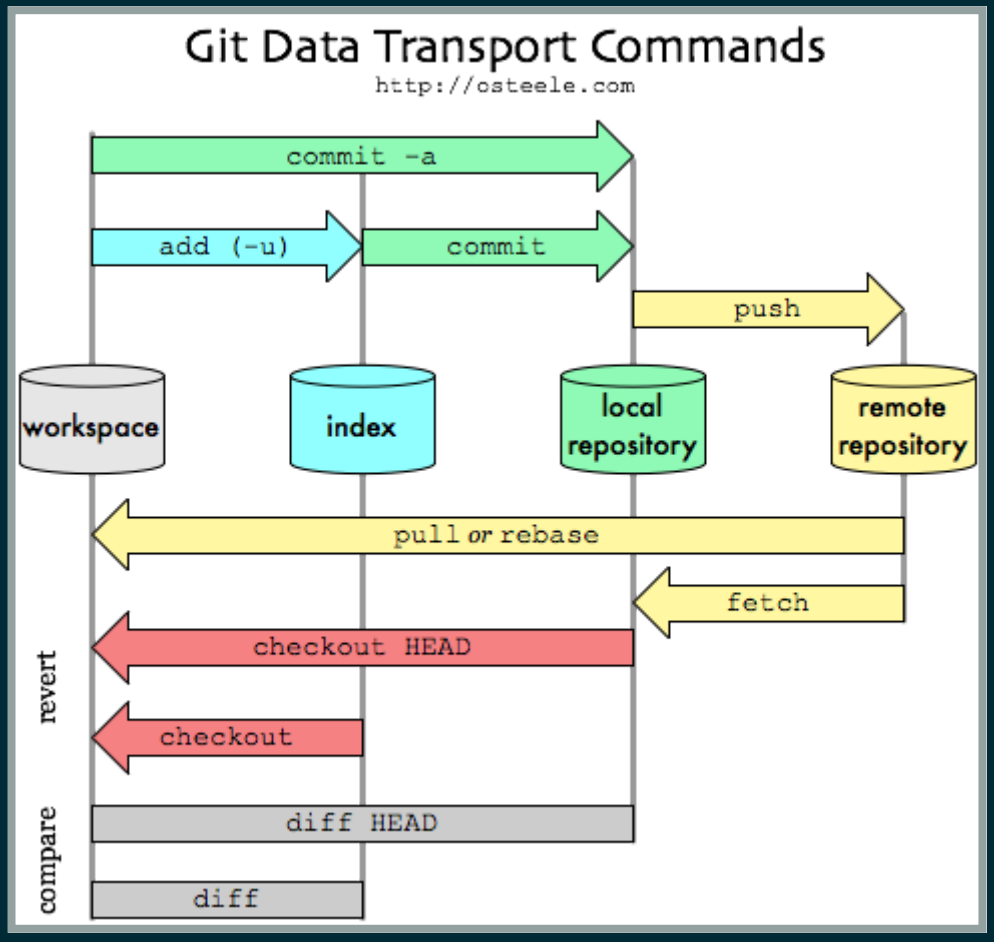
GIT


THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

GIT AREAS

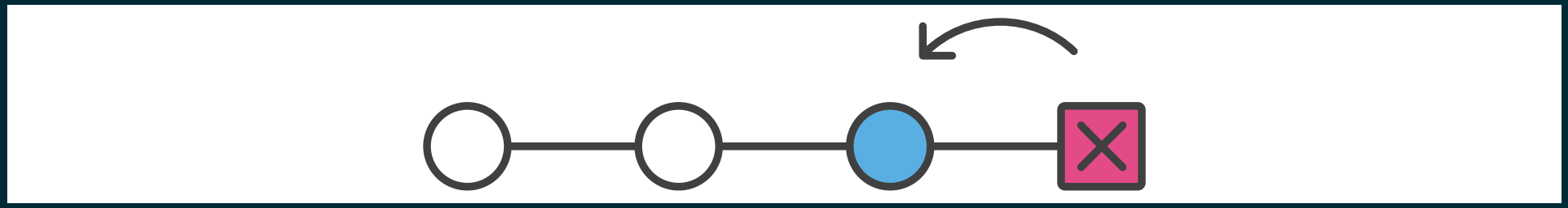

Local Operations
working directory
staging area
git directory (repository)
checkout the project
stage files
commit


History
git reset -- files
git commit
Stage (Index)
git checkout -- files
git add files
Working Directory


Git Data Transport Commands
http://osteele.com
commit -a
add (-u)
commit
push
workspace
index
local repository
remote repository
pull or rebase
fetch
checkout HEAD
checkout
diff HEAD
diff
revert
compare

NOW YOU SEE IT, NOW YOU DON'T

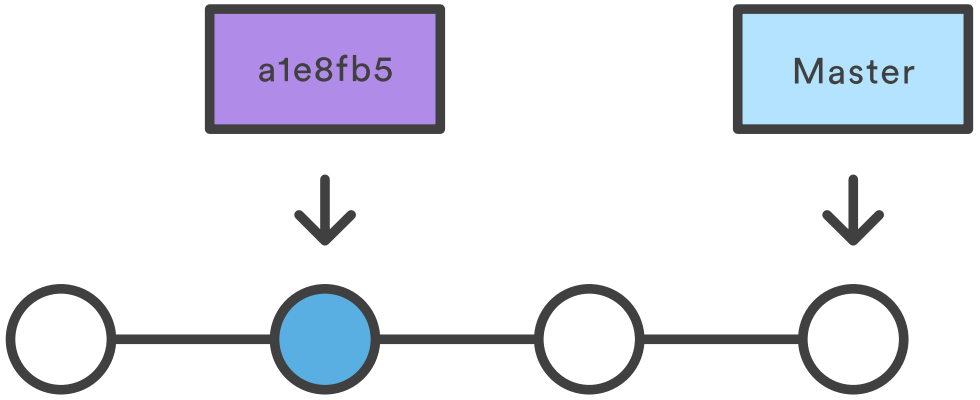
(O DESHACIENDO CON GIT)



La idea detras de GIT es mantener copias "seguras" de un proyecto, para no preocuparnos de romper nuestro código.

Con *git checkout* nos movemos de una copia a otra.

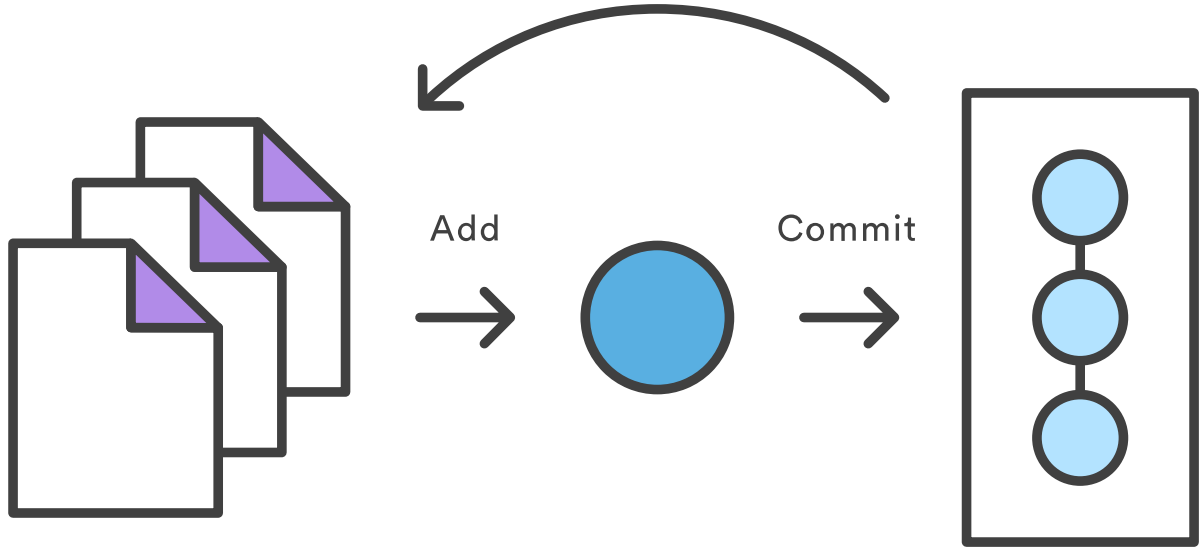
Checking out a previous commit



Checkout es una operación de sólo lectura. No provocará ningún daño, pero se puede volver a committear una version anterior, y esto afecta al repositorio. Ademas de hacer checkout a una rama, podemos hacerlo a un commit específico:

```
git checkout commit
```

Checking out a previous version of a file



y también a una versión anterior de un archivo solo

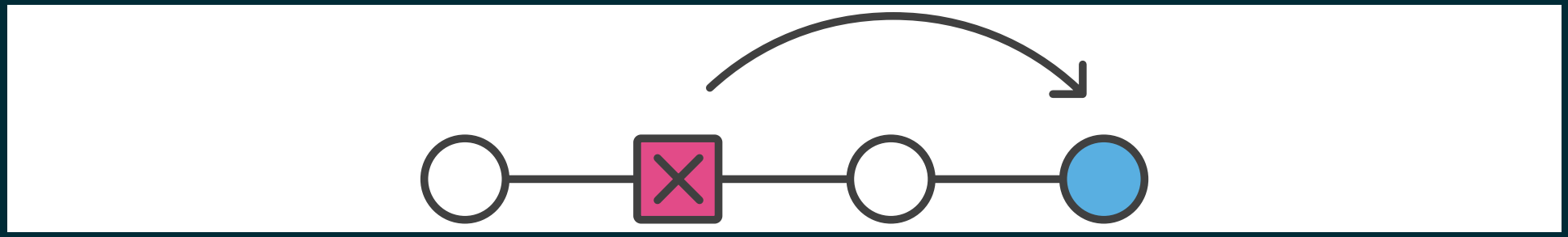
```
git checkout commit file.php
```

o la versión sin modificar de un archivo

```
$ git checkout -- file.py
```

REVERTIR UN COMMIT

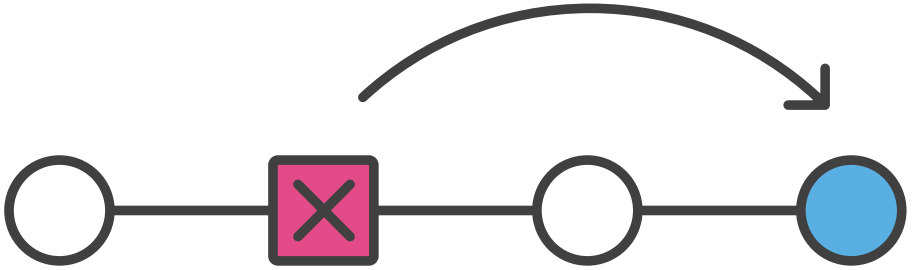
```
$ git revert 1776f5
```



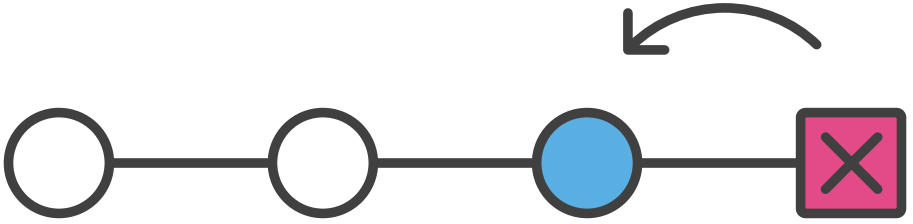
Revert crea un nuevo commit que deshace cambios de un commit previo

Es importante saber que *revert* revierte UN solo commit. No a un estado anterior del proyecto quitando todos los commits desde entonces. En GIT, este proceso se conoce como *reset*.

Reverting



Resetting



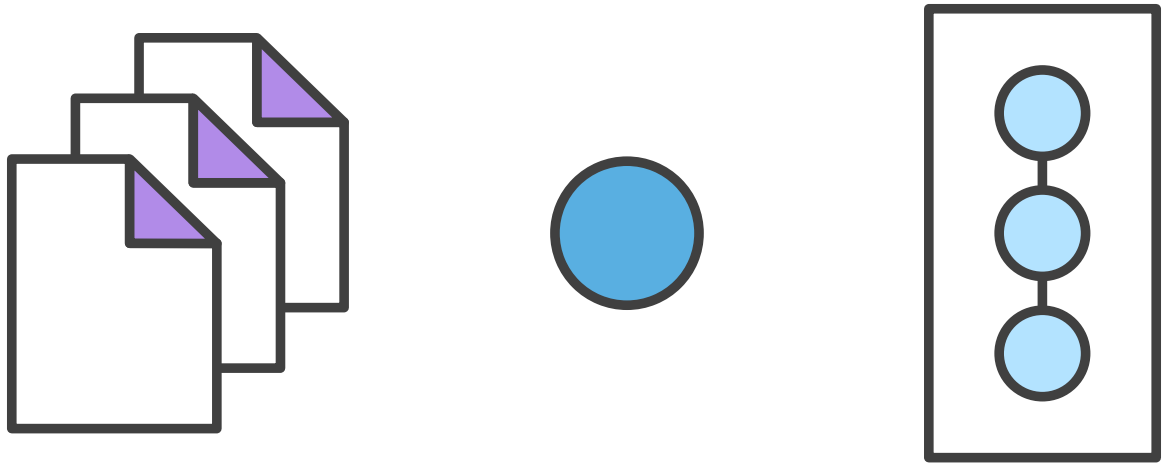
Revert es más seguro, ya que no borra información, sino que genera un nuevo commit deshaciendo el anterior.

Reset por el contrario, tiene el potencial de perder información. Debe usarse localmente porque cambia la historia de git

(kinda sorta)

RESET

The main components of a Git repository



Los tres componentes de un repositorio GIT

RESET EN NIVEL "COMMIT"

```
$ git reset HEAD~2
```

Vuelve 2 commits hacia atras. Esos dos commits, ahora estan sin referencia y serán borrados por GIT en la proxima recoleccion de basura.

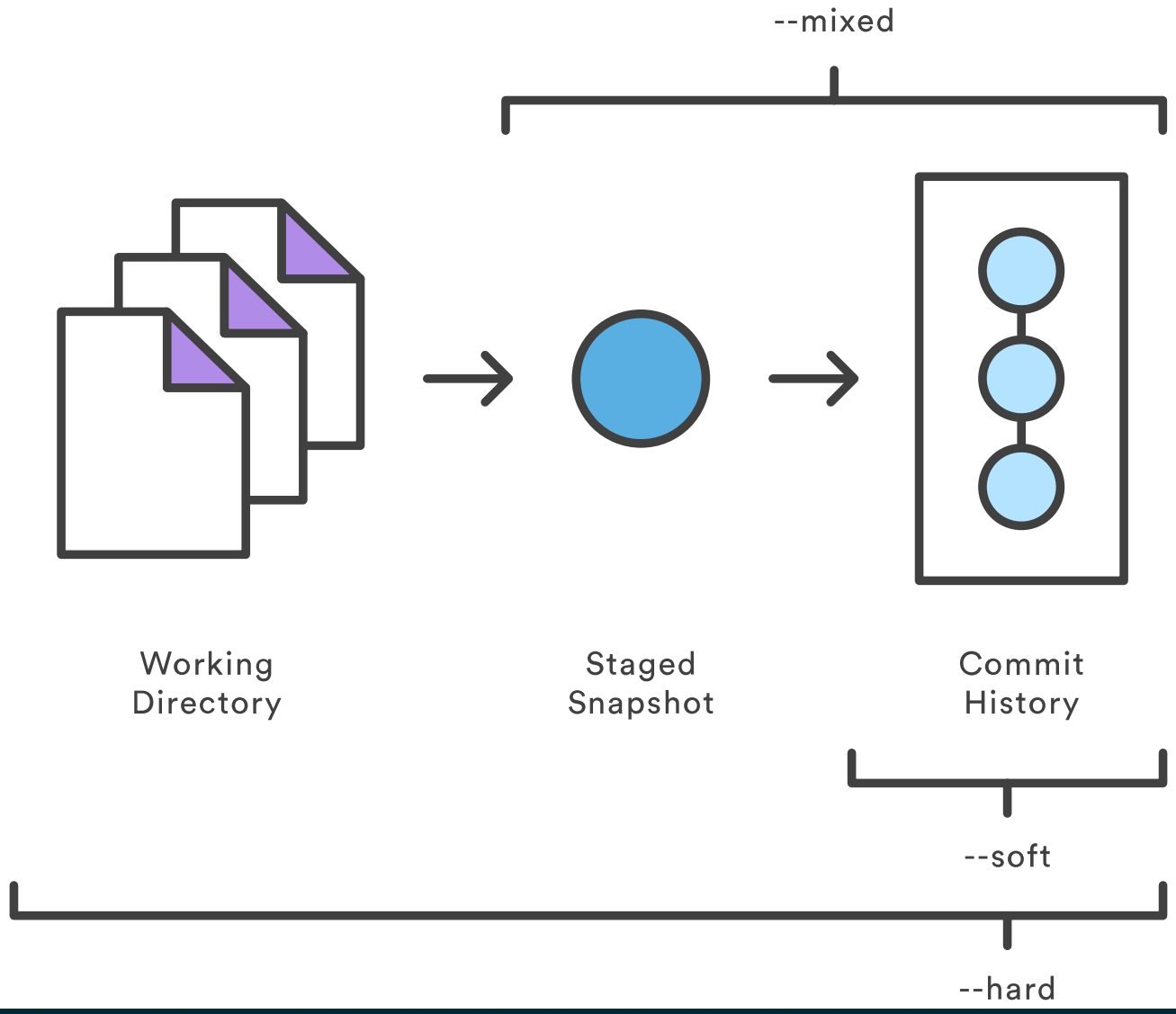
Reset tambien puede modificar el area de staging y el directorio de trabajo, usando estas opciones:

> *--soft – Staging y el directorio de trabajo no se alteran.*

> *--mixed – (default) Staging se actualiza al commit especificado, pero el directorio de trabajo no se afecta.*

> *--hard – Staging y el directorio de trabajo se modifican al commit especificado.*

The scope of git reset's modes



usos más comunes:

```
$ git reset --mixed HEAD
```

Afecta al area de staging, pero mantiene los cambios en el directorio de trabajo

```
$ git reset --hard HEAD
```

Borra todos lo cambios, de staging y el directorio de trabajo

Caveat emptor: usar con cuidado cuando se trabaja en ramas publicas.

RESET, REVERT Y CHECKOUT

Comando	Alcance	Uso más comun
`git reset`	Commit-level	Descarta commits en una rama privada o borra cambios sin comitear
`git reset`	File-level	Quita un archivo de staging
`git checkout`	Commit-level	Cambia de rama o inspecciona commits
`git checkout`	File-level	Descarta cambios en el directorio de trabajo
`git revert`	Commit-level	Deshace commits en una rama pública
`git revert`	File-level	No tiene aplicación

REWRITING HISTORY

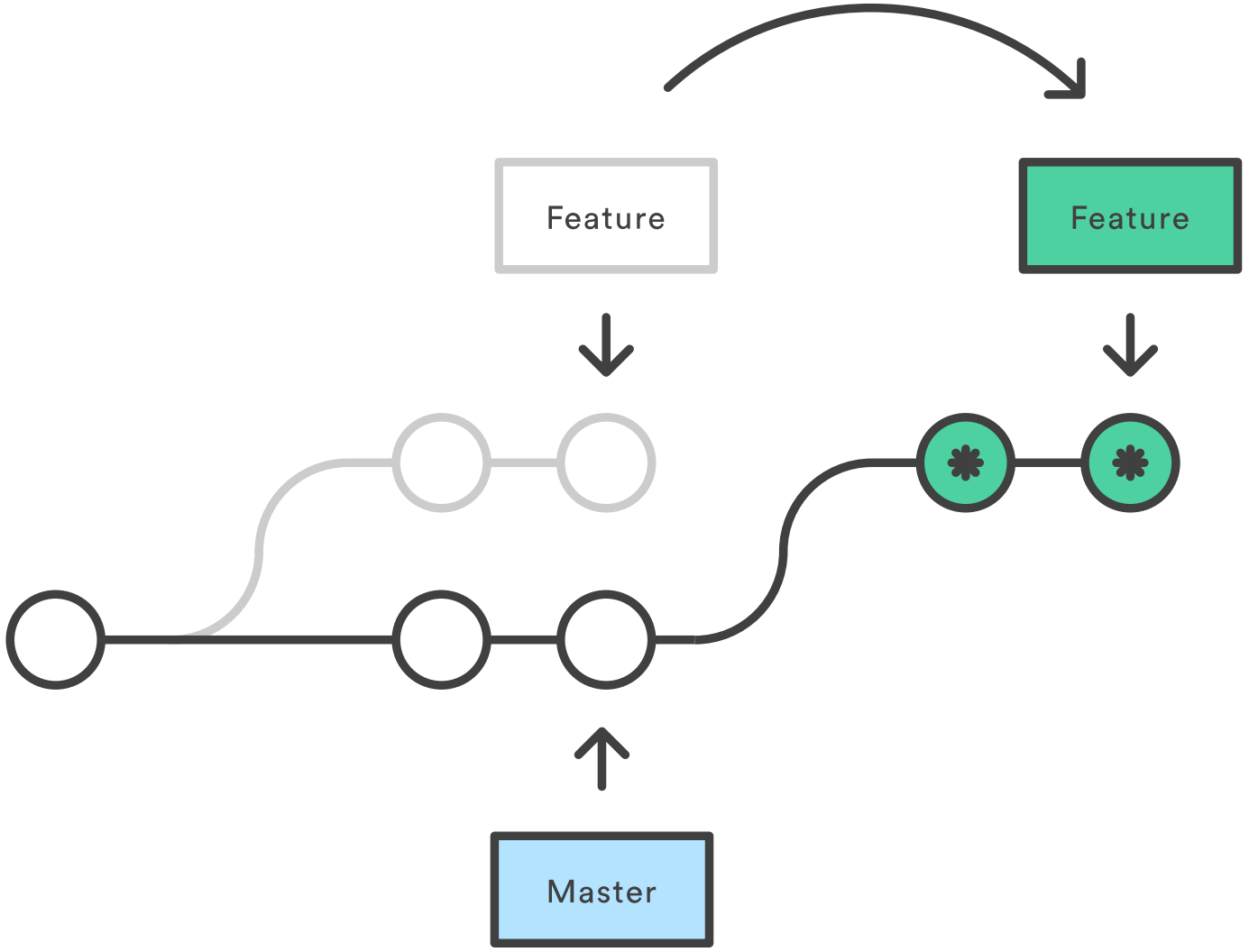
git ofrece herramientas para modificar la historia, pero esto puede ser peligroso.

```
$ git commit --amend
```

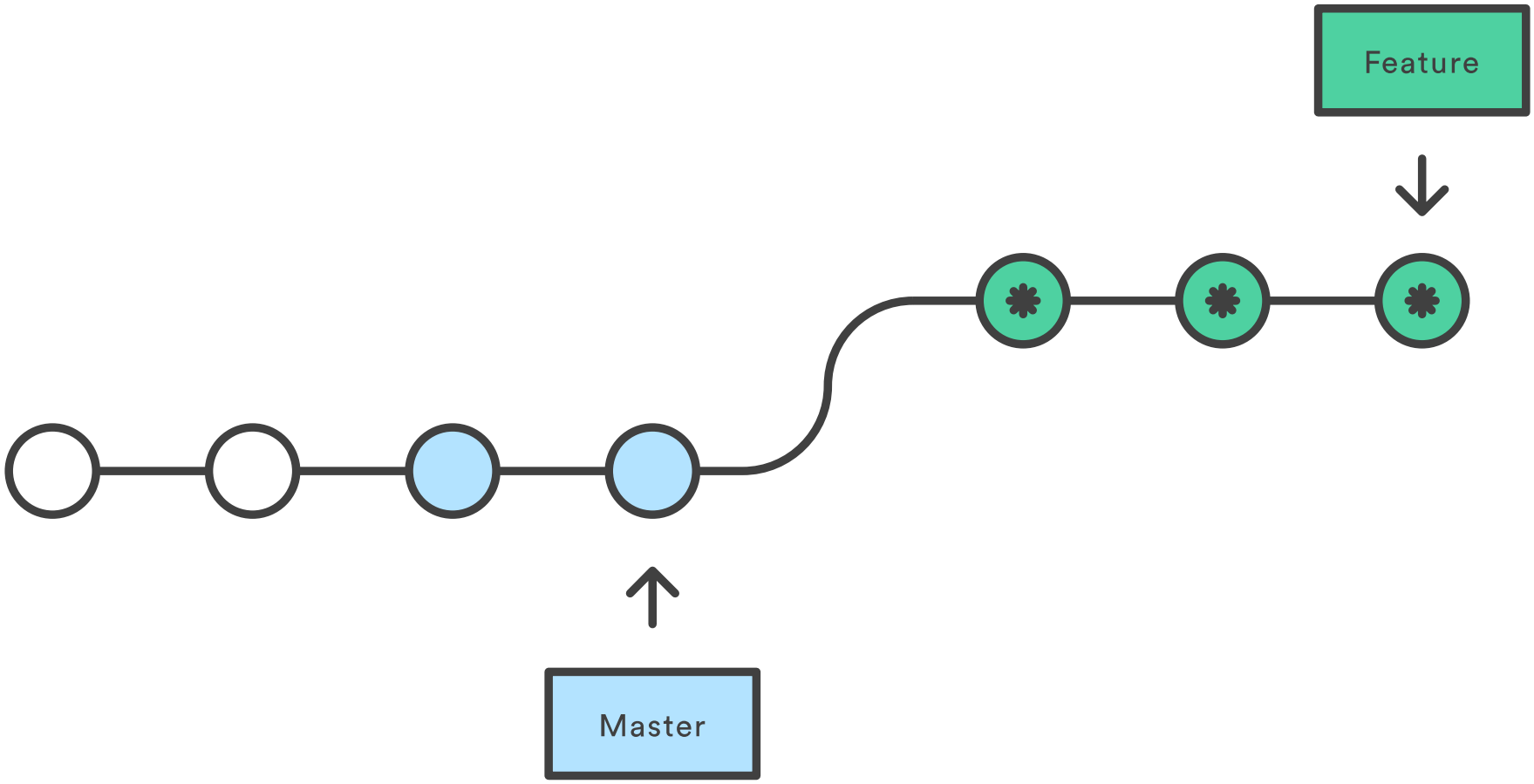
sirve para editar el ultimo commit, y debe hacerse siempre en un commit no publico.

```
$ git rebase
```

Sirve para actualizar una rama con los cambios de otra.


Feature
Feature
Master
Brand New Commits

Rebasing the feature branch onto master



REBASE INTERACTIVO

```
$ git rebase -i base
```

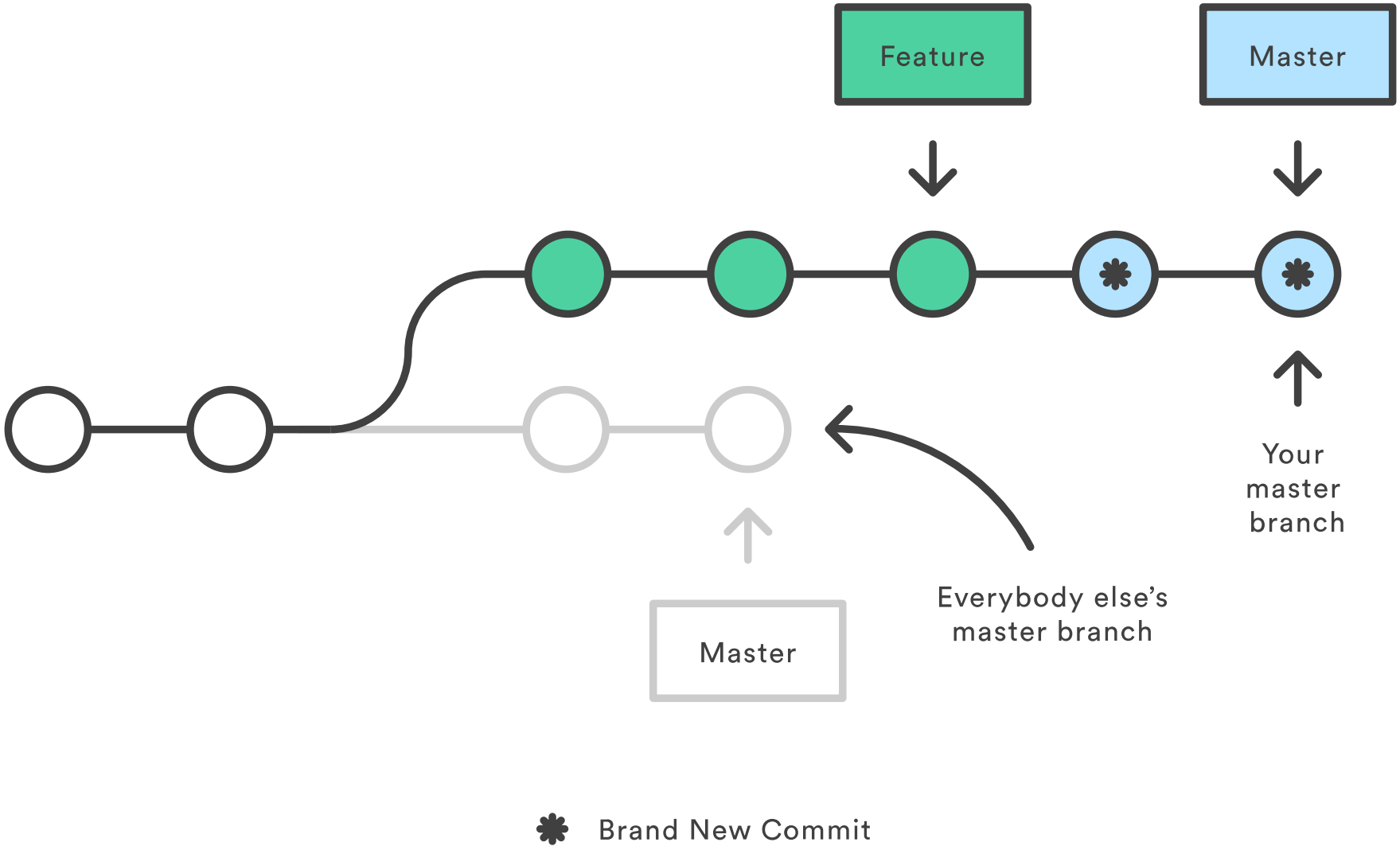
Inicia una sesión interactiva de rebasing
Nos da la oportunidad de limpiar y ordenar la historia.
Permite quitar, dividir y alterar commits existentes.

It's like git commit --amend on steroids.

THE GOLDEN RULE OF REBASING

NUNCA usar en ramas públicas

Rebasing the master branch



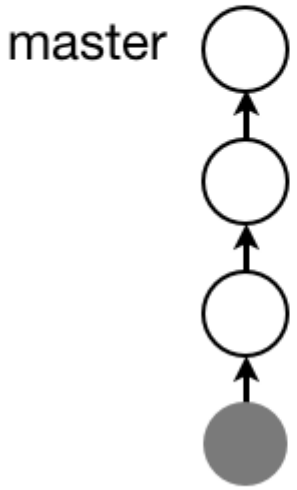
FAST FORWARDING

```
git merge --no-ff
```

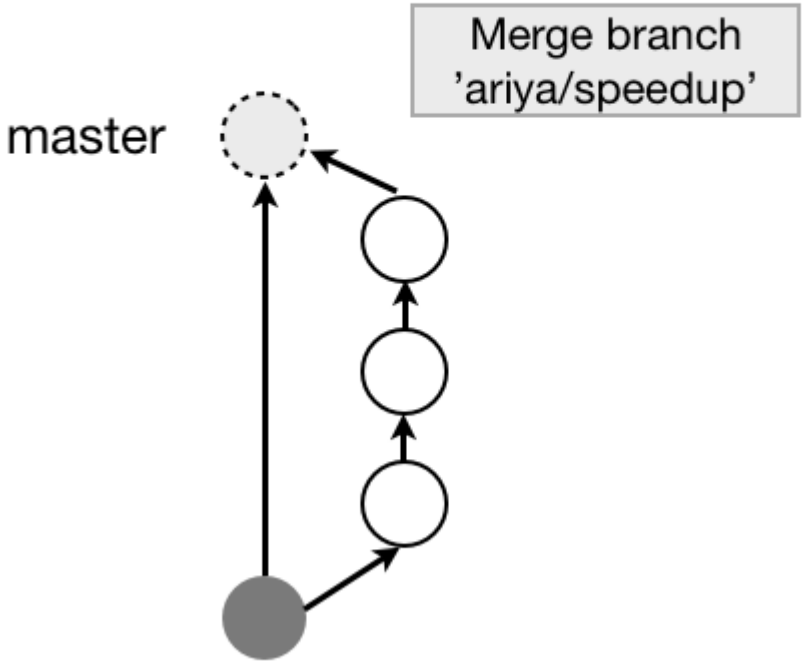
Ocurre cuando se hace git pull sin tener ningun cambio local.
Git simplifica las cosas avanzando el puntero, ya que no hay ningún otro trabajo divergente a fusionar.

Con --no-ff se genera un commit adicional enfatizando el merge

```
git fetch ariya
git merge ariya/speedup
```




```
git fetch ariya
git merge -no-ff ariya/speedup
```



CHERRY PICK

Genera un nuevo commit con commits elegidos de otra rama

Paso 1.

```
git checkout rama-a-la-que-queremos-agregar-commits
```

Paso 2.

```
git cherry-pick c90fd66
```

FORMATEANDO Y FILTRANDO GIT LOG

git log is ugly :(

```
commit 3f1c1a58f9fb660a1446eea99528543e35fde1de
Author: kilinkis
Date:   Fri Feb 10 17:26:24 2017 -0300

    Fix margin iOS and Android landing pages

commit 50127ce2ec77b1013773d6f074a6e95c509115ce
Author: kilinkis
Date:   Fri Feb 10 17:14:27 2017 -0300

    Fix: Drupal landing page margin

commit bef189d53c0e83c4b9e1d855411d0ad974159ab3
Author: kilinkis
Date:   Fri Feb 10 16:53:18 2017 -0300
```

```
git log --oneline
```

--oneline imprime un commit por linea

```
3f1c1a5 Fix margin iOS and Android landing pages
50127ce Fix: Drupal landing page margin
bef189d Fix margin in landing page for Internal Solutions
96ab0fa Plugin added: Redirect unattached images
734ce00 Merge remote-tracking branch 'origin/fs-19' into fs-19
dceba5f Fix: landing page Internal Solutions
68dec55 Merge branch 'develop' into fs-19
```

```
git shortlog
```

shortlog lista los mensajes de los commits

```
Juan Incaurgarat (33):
     Initial work on custom landing page posts with Advanced Cust
     Merge branch 'custom-landing-page-posts' into feature-sprint
     Merge pull request #5 from kilinkis/custom-landing-page-post
     Hotfix for custom landing page template
     Merge pull request #6 from kilinkis/case-nalc
```

```
git log --graph
```

Grafica actividad en el repositorio

```
| * |   commit cbc02e2bf96971fc359569f089f525d734fcac94
| | |\ \  Merge: 6c950ef c45068d
| | | |/  Author: Juan Manuel Incaurgarat
| | | |   Date:   Fri Dec 23 17:12:25 2016 -0300
| | | |
| | | |       Merge pull request #62 from kilinkis/feature-sprint
| | | |
| | | |       Feature sprint 14
| | | |
| | | * commit c45068dd87baf37fd28622efbc952d94bb86679e
| | | | Author: kilinkis
| | | | Date:   Fri Dec 23 13:50:29 2016 -0300
| | | |
| | | |     Fix cropped form in small devices
| | | |
| | | |     https://intridea.atlassian.net/browse/KILINKIS-363
```

```
git log --pretty=format:"%cn commited %h on %cd"
```

Dando formato al git log

```
kilinkis commited 3f1c1a5 on Fri Feb 10 17:26:24 2017 -0300
kilinkis commited 50127ce on Fri Feb 10 17:14:27 2017 -0300
kilinkis commited bef189d on Fri Feb 10 16:53:18 2017 -0300
kilinkis commited 96ab0fa on Fri Feb 10 14:01:03 2017 -0300
```

Filtrando por fechas

```
git log --after="2016-12-30" --before="yesterday"
```

LIMPIANDO EL .GITIGNORE

IGNORAR ARCHIVOS QUE YA FUERON COMMITEADOS AL REPO

Paso 1. Aplicar al .gitignore los patrones de los archivos que queremos ignorar.
Paso 2. Commitear (o stash) cualquier cambio local.
Paso 3:

```
// git rm --cached /path/to.file
$ git rm -r --cached .
$ git add .
$ git commit -m "Clean up ignored files"
```

QUITAR ARCHIVOS DE UN COMMIT

```
git reset --soft HEAD^ // o git reset --soft HEAD~1
```

Resetear el archivo no deseado para dejarlo fuera del commit:

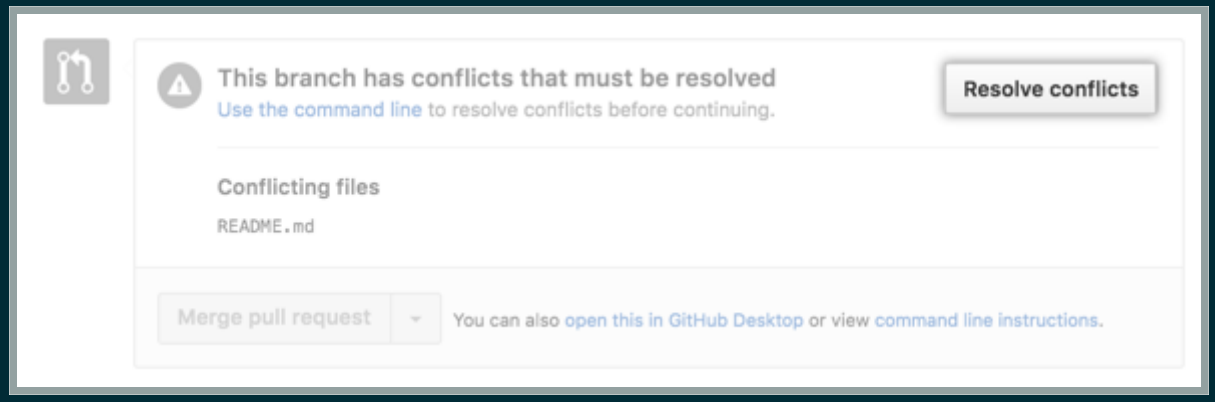
```
git reset HEAD path/to/unwanted_file
```

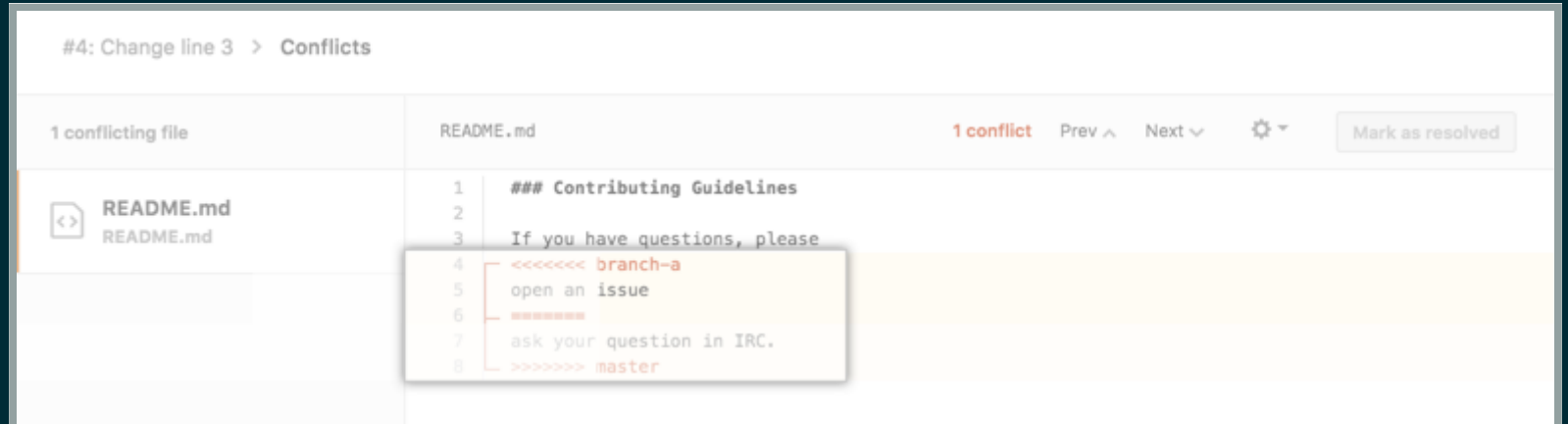
Commitear (incluso se puede usar el mismo mensaje):

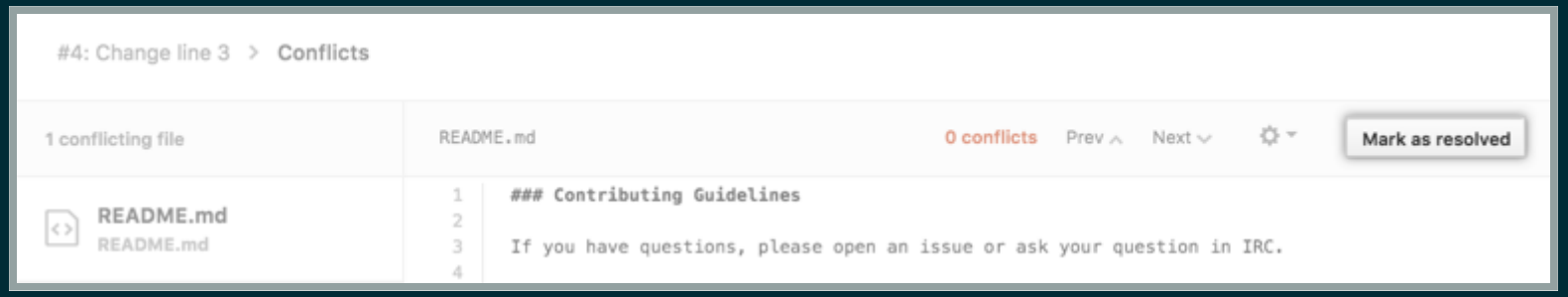
```
git commit -c ORIG_HEAD
```

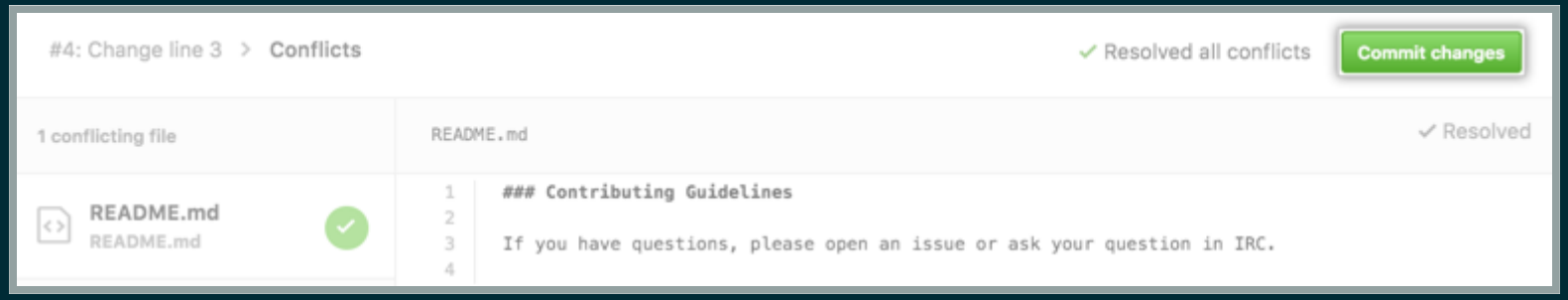
RESOLVER CONFLICTOS ONLINE

Si en un Pull Request hay un conflicto:






#4: Change line 3 > Conflicts
1 conflicting file
README.md
README.md
README.md
0 conflicts
Prev
Next
Mark as resolved
1 ### Contributing Guidelines
2
3 If you have questions, please open an issue or ask your question in IRC.
4


#4: Change line 3 > Conflicts
Resolved all conflicts
Commit changes
1 conflicting file
README.md
README.md
README.md
Resolved
1 ### Contributing Guidelines
2
3 If you have questions, please open an issue or ask your question in IRC.
4

GIT EN NUESTRO PROPIO SERVIDOR

Paso 1. Instalar GIT en ambas máquinas:

```
sudo apt-get install git-core
```

Paso 2. Agregar user y generar llaves SSH (privada y pública).

```
sudo useradd git
passwd git
ssh-keygen -t rsa
```

Paso 3. Copiamos las claves al server

```
cat ~/.ssh/id_rsa.pub | ssh git@remote-server "mkdir -p ~/.ssh &&
```

Paso 4. Crear un directorio para el proyecto de git en la maquina remota

```
git@server:~ $ mkdir -p /home/username/project-1.git && cd "$_"
```

Paso 5. Iniciar el repo

```
git@server:~ $ git init --bare
//Initialized empty Git repository in /home/username/project-1.gi
```

Paso 6. Creamos el proyecto localmente

```
mkdir -p /home/username/git/project && cd "$_"
git init
//Initialized empty Git repository in /home/username/git/project
```

Paso 7. Crear y agregar un archivo

```
touch file.php
git add .
```

Paso 8. Commitear

```
git commit -m "message" -a
[master e517b10] message
1 file changed, 1 insertion(+)
```

Paso 9. Agregamos el origen remoto

```
git remote add origin ssh://git@remote-server/repo->path-on-serve
```

Paso 10. Ahora ya podemos pushear (y pullear) nuestros cambios.

```
git push origin master
```

And Presto!
Otros colaboradores pueden clonar el proyecto

```
git clone git@remote-server:/home/username/project.git
```

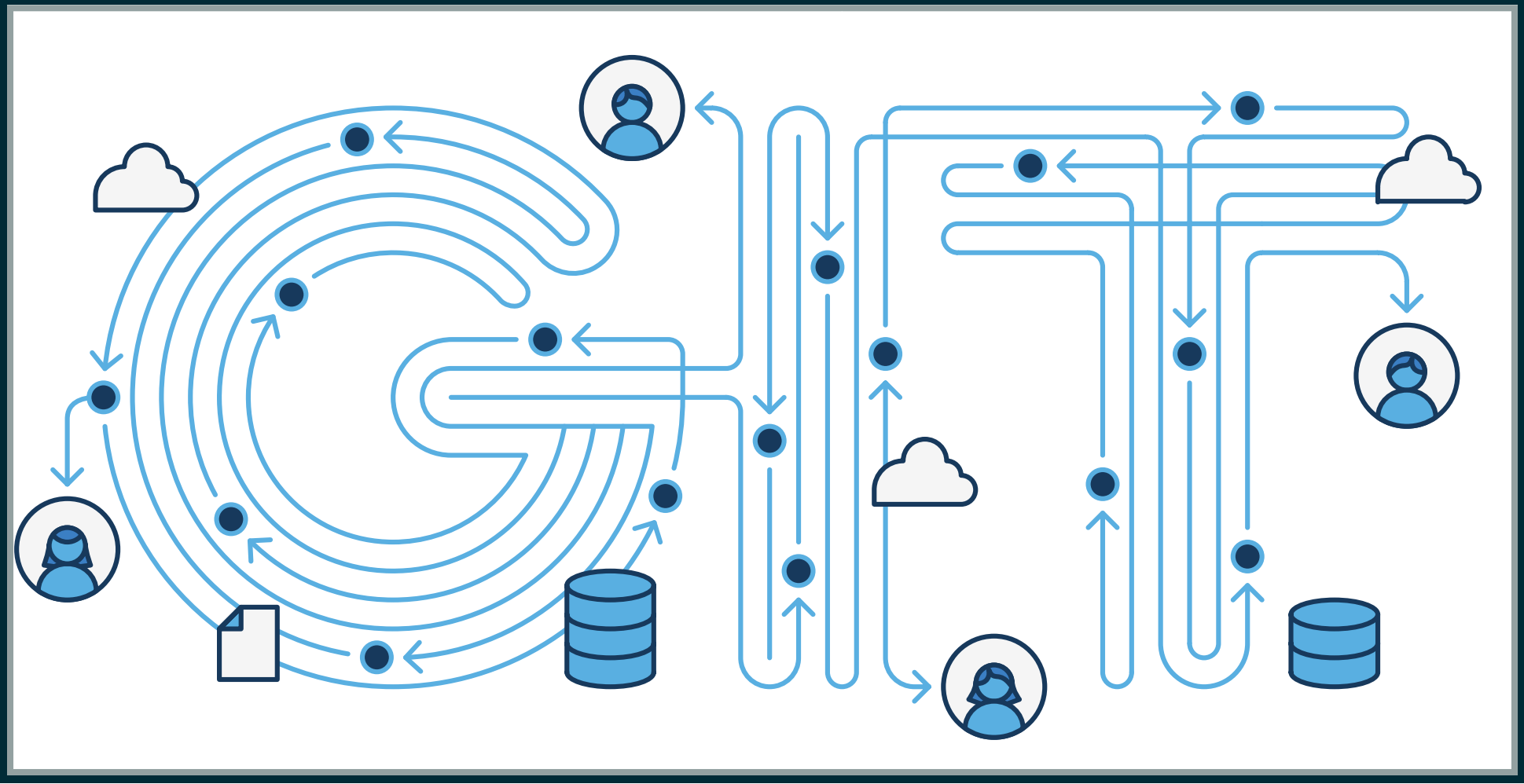
CONCLUSIONES:

GIT es una herramienta muy versatil que permite trabajar de forma distruibuida en la que cada desarrollador tiene una copia del proyecto. Se puede llegar al mismo resultado de distintas maneras. No tiene un unico modo de uso.

Distintos equipos lo usan de distintas formas, pero es solamente "how it's done here". El GIT workflow esta abierto a mejoras, modificaciones y colaboraciones.

PREGUNTAS?

FUENTES, LECTURAS Y EJERCITACIÓN



- atlassian.com/git
- Visual guide to GIT
- GIT official documentation
- GIT tower
- Rewrite history with rebase
- Reset vs Checkout vs Reverting
- Merge vs Rebase
- Run GIT on your server
- GIT on a server
- Resolving a merge conflict on github

- Hooks
- A successful git branching model
- GIT cheatsheet
- .gitignore templates
- GIT interactive cheatsheet
- GIT workflows

- git how to
- GIT immersion

THANK YOU!