

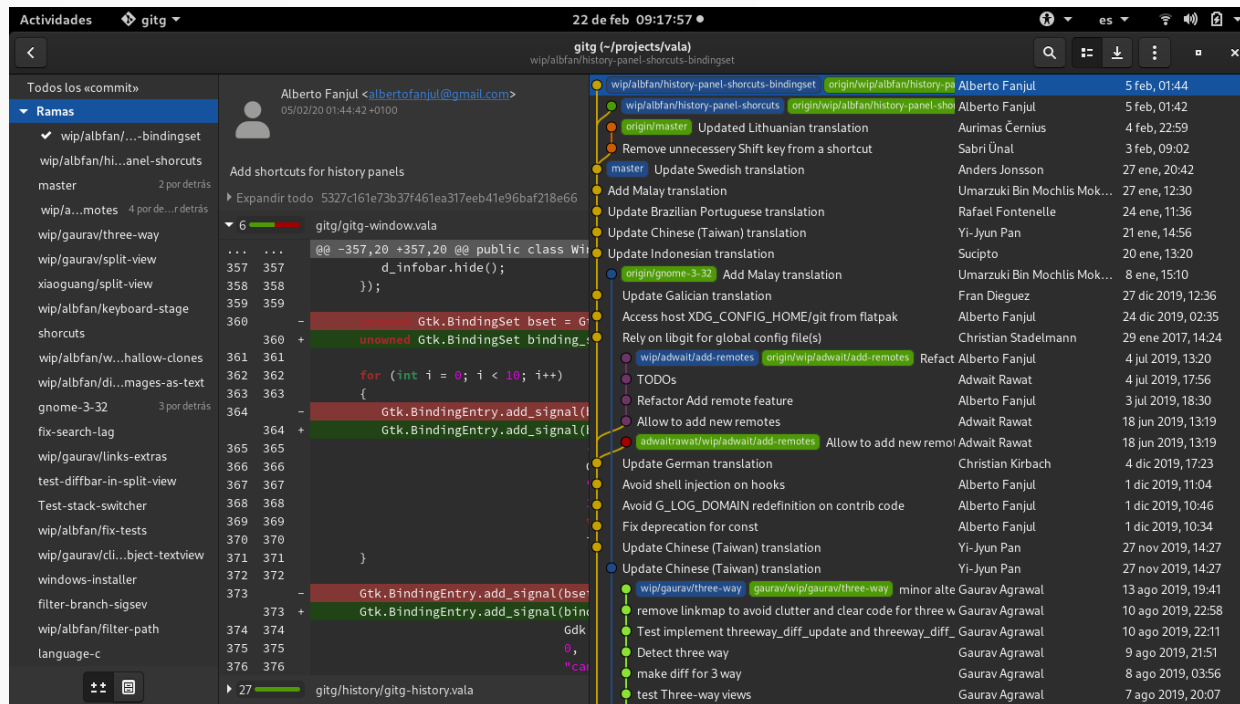


Cursillo básico git

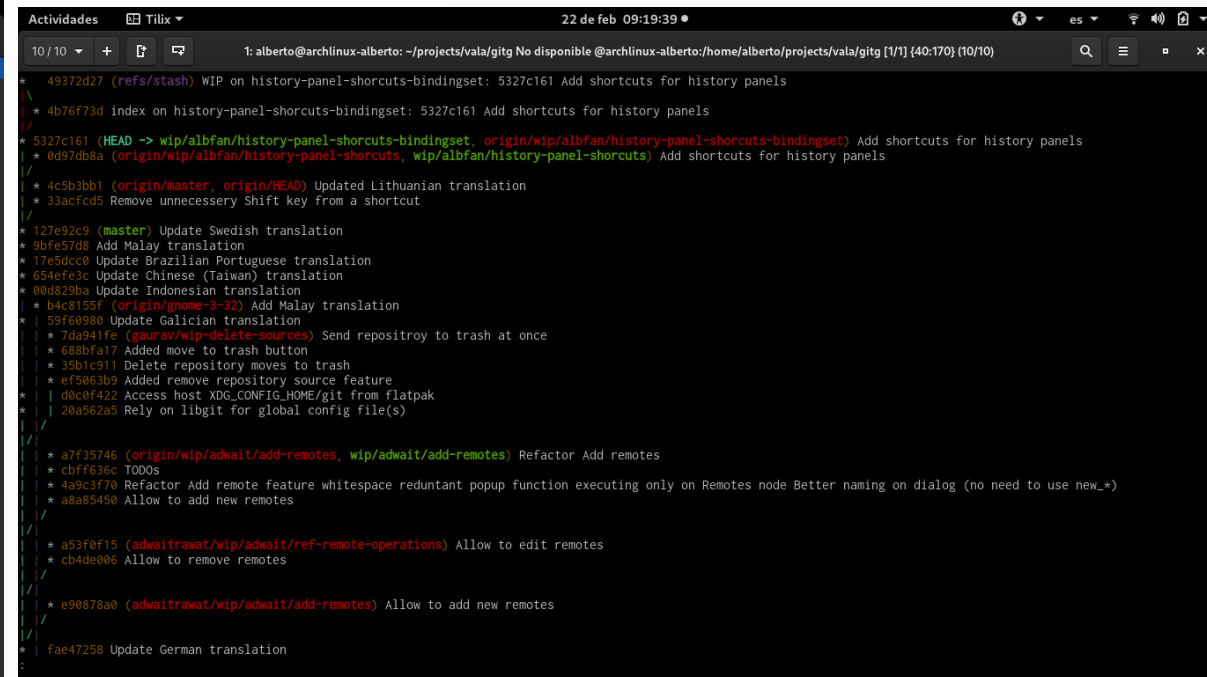
Alberto Fanjul Alonso
Responsable de gitg

Introducción al control de versiones git

- Visualizando un repositorio git



The screenshot shows the Git GUI interface. On the left, a list of branches is visible, including 'wip/albfan/...-bindingset', 'wip/albfan/hi...anel-shorcuts', 'master', 'wip/a...motes', 'wip/gaurav/three-way', 'wip/gaurav/split-view', 'xiaoguang/split-view', 'wip/albfan/keyboard-stage', 'shorcuts', 'wip/albfan/w...hallow-clones', 'wip/albfan/di...ages-as-text', 'gnome-3-32', 'fix-search-lag', 'wip/gaurav/links-extras', 'test-diffbar-in-split-view', 'Test-stack-switcher', 'wip/albfan/fix-tests', 'wip/gaurav/cli...bject-textview', 'windows-installer', 'filter-branch-sigsev', 'wip/albfan/filter-path', and 'language-c'. The main area displays a commit history graph with a vertical timeline of commits. The current commit is 'wip/albfan/history-panel-shorcuts-bindingset' by Alberto Fanjul, dated 5 Feb, 01:44. The commit message is 'Add shortcuts for history panels'. The diff view shows changes to 'gitg/gitg-window.vala' and 'gitg/history/gitg-history.vala'.



The screenshot shows a terminal window with the output of a 'git log' command. The output lists a series of commits, including the current commit '49372d27 (refs/stash) WIP on history-panel-shorcuts-bindingset: 5327c161 Add shortcuts for history panels'. The log shows the commit history, including the current commit and its parents, with commit hashes, branch names, and commit messages.

Qué es un control de versiones?

Se refiere a un sistema para almacenar los cambios que se realizan en cierta información. Normalmente se refiere a texto.

- Si tenemos un fichero saludos:

Hola,
Bienvenidos

- Y lo cambiamos a:

Hola, Bienvenidos al cursillo de git.
Tomad asiento, teneis enchufes disponibles y Wifi

- Tal vez queramos dejar constancia de este cambio como:

Introducción e información para asistentes

Eso es un control de versiones.

Qué es git?

git es un programa para realizar control de versiones. Lo hace a través de:

- Repositorio: Concepto que representa como cambia el contenido en el tiempo
- Commit: Representación de un cambio
- Historia: Relación entre commits

git permite realizar todas las operaciones esperables en un control de versiones:

- Añadir contenido
- Registrar cambios
- Bifurcar contenido (crear ramas)
- Fusionar contenidos en conflicto

Instalando git

Para instalar git en cualquier sistema operativo, solo hay que acudir a su página web:

- <https://git-scm.com/downloads>

Donde encontraremos instaladores para el sistema operativo que eligamos

Configurar git

Git identifica a la persona que realiza un cambio y a la persona que lo agrega al repositorio, para ello necesita que configures tu usuario y tu email.

- ¿Por qué el email? git esta basado en un sistema por el que unos desarrolladores trabajaban antes de existir git (se enviaban los cambios por correo) A su entender el email es una información importante sobre quien hace un cambio por eso se necesita.
- NOTA: En realidad el email no tiene que ser real, no se hace ninguna comprobación sobre él.

```
git config --global user.name Nombre
```

```
git config --global user.email mi@correo.com
```

Es suficiente para configurar git

Crear un commit

git es un control de versiones descentralizado. Eso significa que no hay que conectarse a ningún servicio, servidor o base de datos para almacenar los cambios.

¿Como iniciamos un repositorio donde guardar los cambios de git?

```
mkdir prueba
```

- ```
cd prueba
```
- ```
git init
```

- Y ya hemos creado un repositorio

```
echo hola > file1
```

- Y ya hemos creado contenido en nuestro espacio de trabajo

```
git status
```

- Veremos que git no sabe nada de ese fichero

```
git add .
```

- Ahora git lo ha marcado para almacenarlo

```
git commit -m "Primera versión"
```

- Ahora git ya ha registrado este cambio.
- Para poder ver este cambio podemos usar

```
git show  
git log
```

Hacer un cambio

Ya estamos en un repositorio git. Si cambiamos algún fichero, git se dará cuenta.

```
echo adios > file2  
echo buenos días >> file1
```

- Nosotros debemos indicarle a git que registre los cambios

```
git add file1
```

- Volvemos a hacer un commit

```
git commit -m "Mas saludos"
```

- Git permite agregar solo parte de los cambios

```
git status
```

- Para seleccionar parte de los cambios usaremos

```
git add -p
```

- Ahora git ha agregado esta información

```
git commit -m "Despedidas"
```

- De nuevo podemos ver los cambios que hemos hecho

```
git show  
git log
```


Ver un cambio

Git sabe lo que hemos cambiado en cada “foto” que hemos realizado. Como podemos verla?

```
git diff
```

- Que estamos viendo? Formato

```
diff --git a/file2 b/file2
new file mode 100644
index 0000000..3ec006e
--- /dev/null
+++ b/file2
@@ -0,0 +1 @@
+Adios
```

[Vista de un diff]

- Podemos pedirle a la vista de log que nos muestre los diff

```
git log -p
```

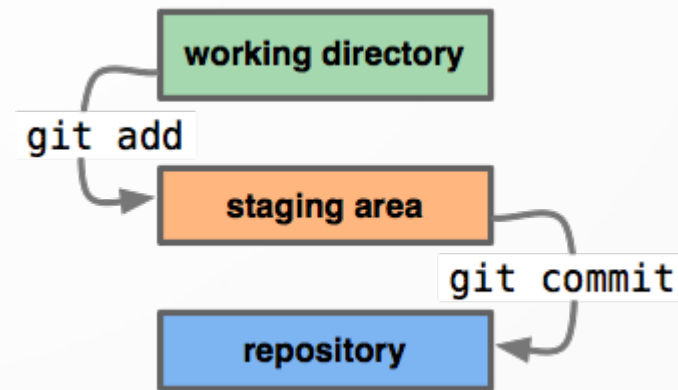
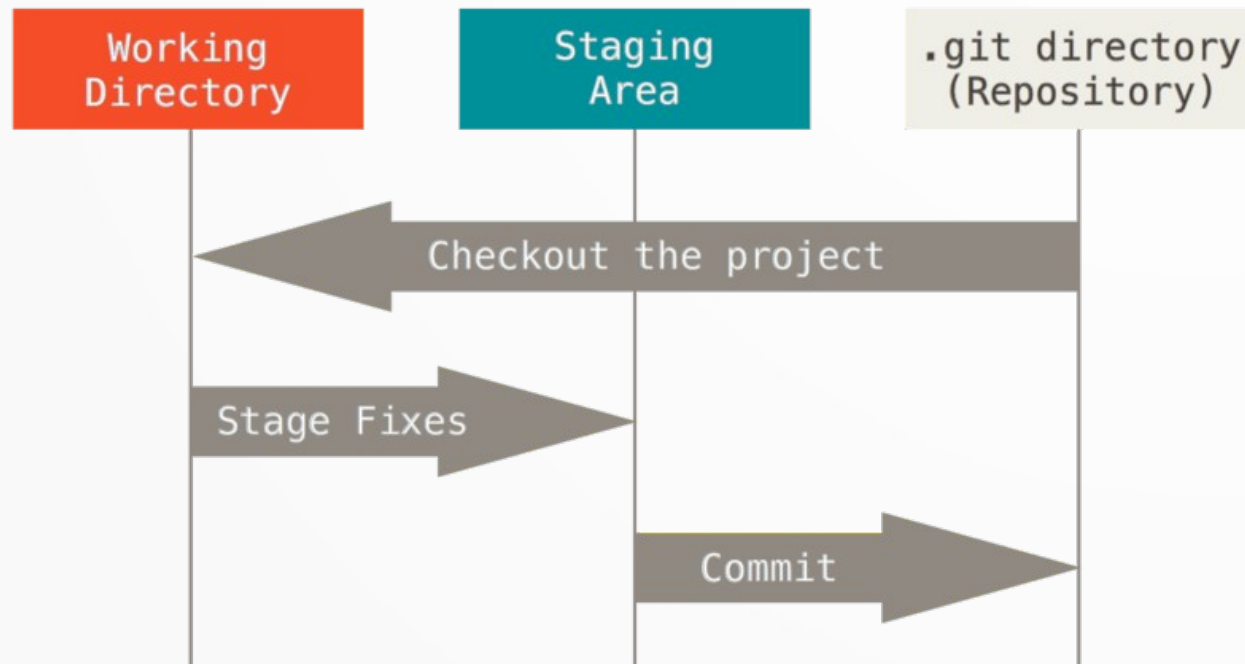
- Como ver cambios concretos? [RTFM]

```
man gitrevisions
```

- Si no nos aclaramos podemos verlo gráficamente

```
git difftool -y
git showtool -y
gitk
```

Un poco de teoría...

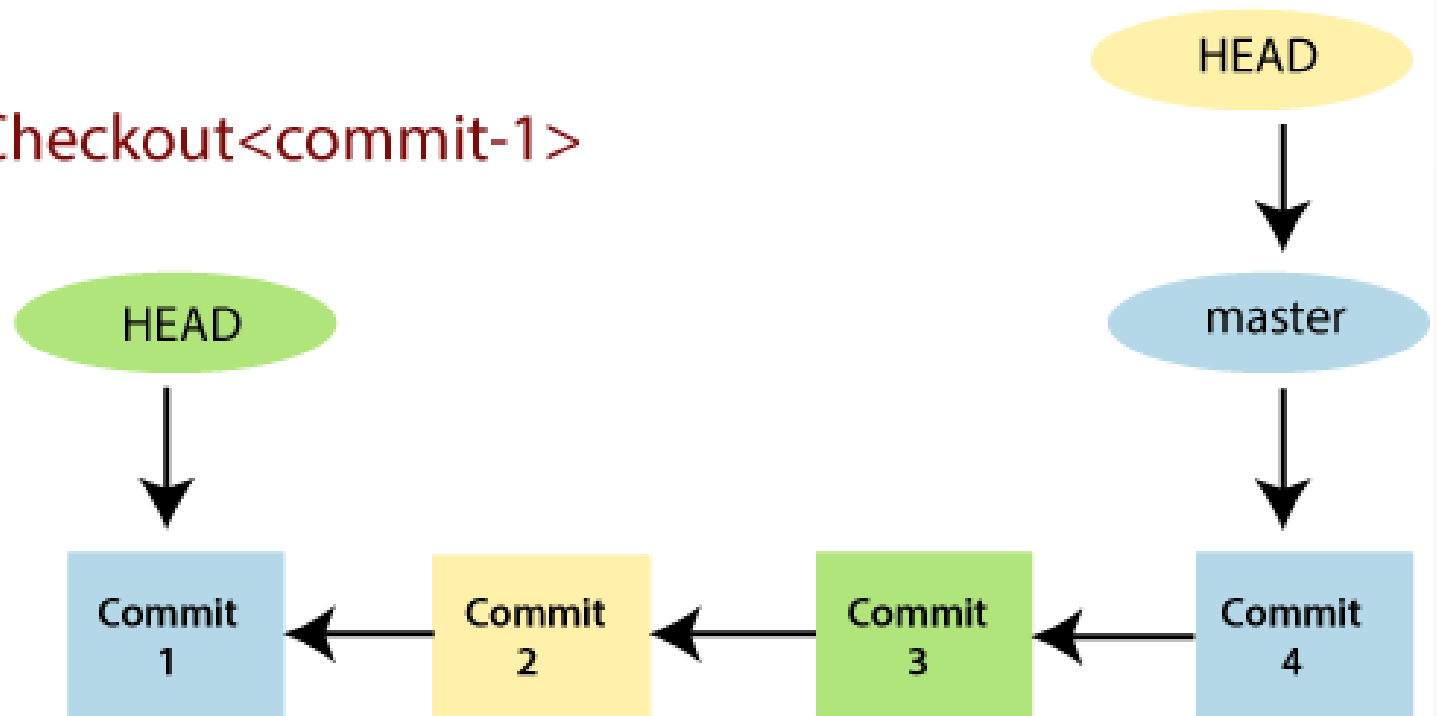


... un poco mas de teoría

git checkout

nos permite movernos
por la historia del repositorio

Checkout<commit-1>



Crear una rama

En un control de versiones la historia puede ramificarse, tener diferentes direcciones. Es lo que se llama una rama

```
git branch rama1
```

- Crear una rama no nos pone a trabajar en ello, para eso usamos

```
git checkout rama1
```

- Lo comprobamos con:

```
git log
```

- Esto es tan típico que git ofrece una opción para hacerlo a la vez

```
git checkout -b rama1
```

- Ya podemos hacer modificaciones en nuestra nueva rama

```
echo Hello >> file1  
echo Bye! >> file2  
git add .
```

- Y guardarlos en otro commit

```
git commit -m "saludos en inglés"
```

- Podemos visualizar esta información

```
git log --all --graph
```

- Como veis git permite muchas opciones, lo normal es salvarse atajos en un fichero de configuración:

```
vi $HOME/.gitconfig  
vi .git/config
```

- Este es el mío

<https://github.com/albfan/dotfiles>



Mas teoría...

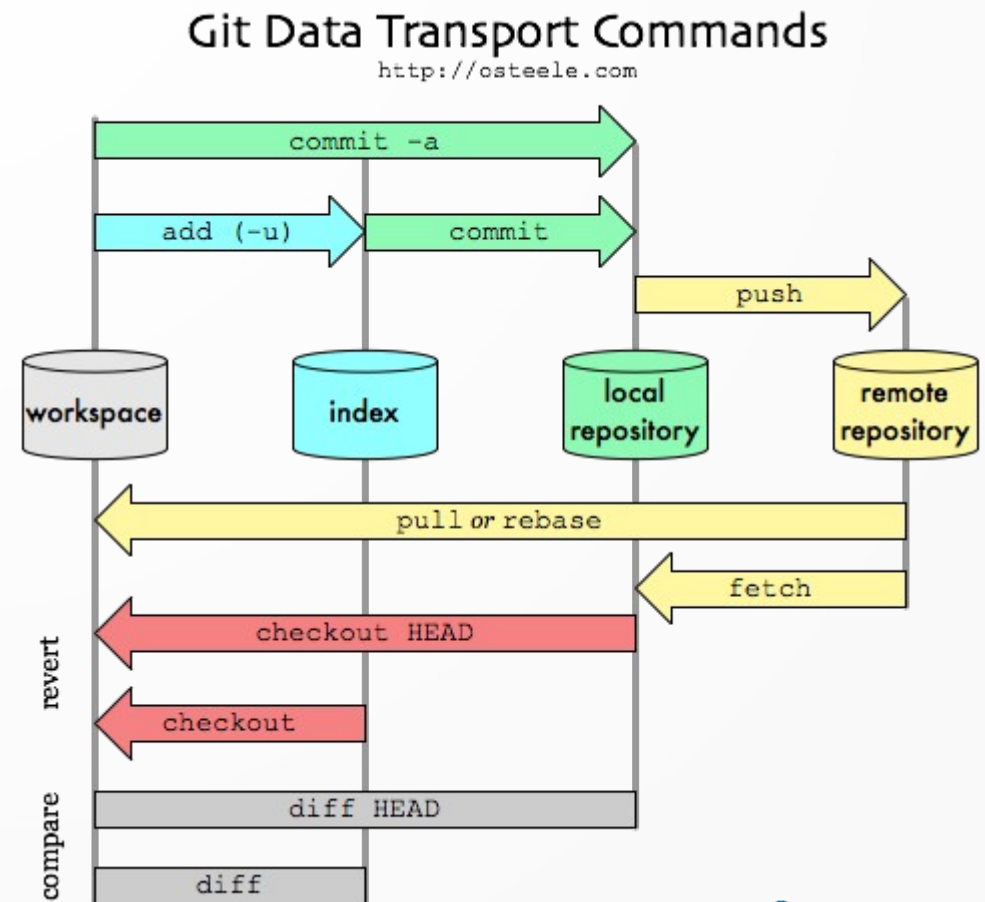
git remote

- Nos permite enlazar repositorios externos
- Así funciona git como repositorio descentralizado
- Si clonamos un repositorio externo

git clone

- Ya nos agrega el remote "origin". Para verlo

git remote -v



...lo último

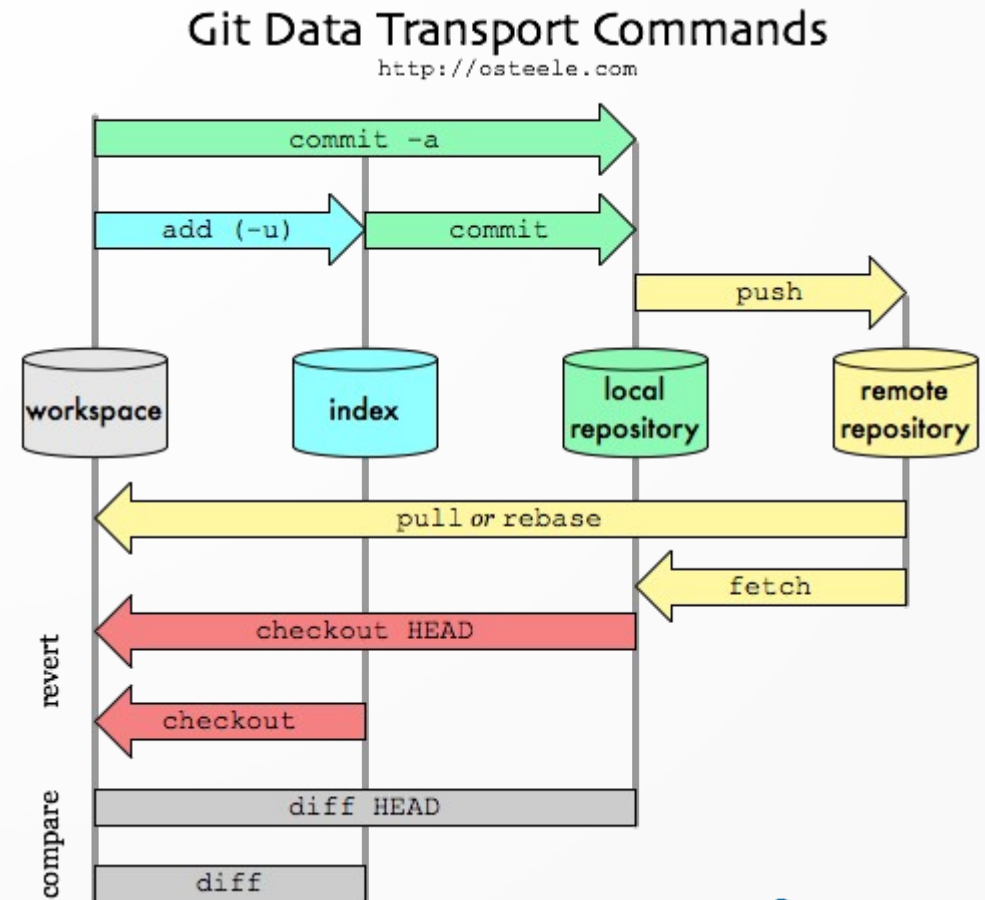
git fetch

- Nos permite traernos todos los cambios en un remote

git push

- nos permite subir cambios a un remote (tenemos que tener permiso)
- Vamos a intentar hacerlo sobre el ejemplo de este cursillo:

<https://github.com/albfan/saludos>



Dudas Preguntas