




Git

Control de versiones
y
desarrollo en equipo



Definición

- DVCS, o sistema de control de versiones distribuido.
- Diseñado para el desarrollo del kernel de Linux.
- Desde abril del 2005. Su última versión es de febrero de 2016. v2.7.2
- Programado en C, Bourne Shell, Perl.
- Código abierto bajo licencia GPLv2.

¿Por qué control de versiones?

Desarrollo de software

Buenas prácticas

Trabajo en equipo

Evitar el envío de los ficheros de un proyecto

¿Qué permite?

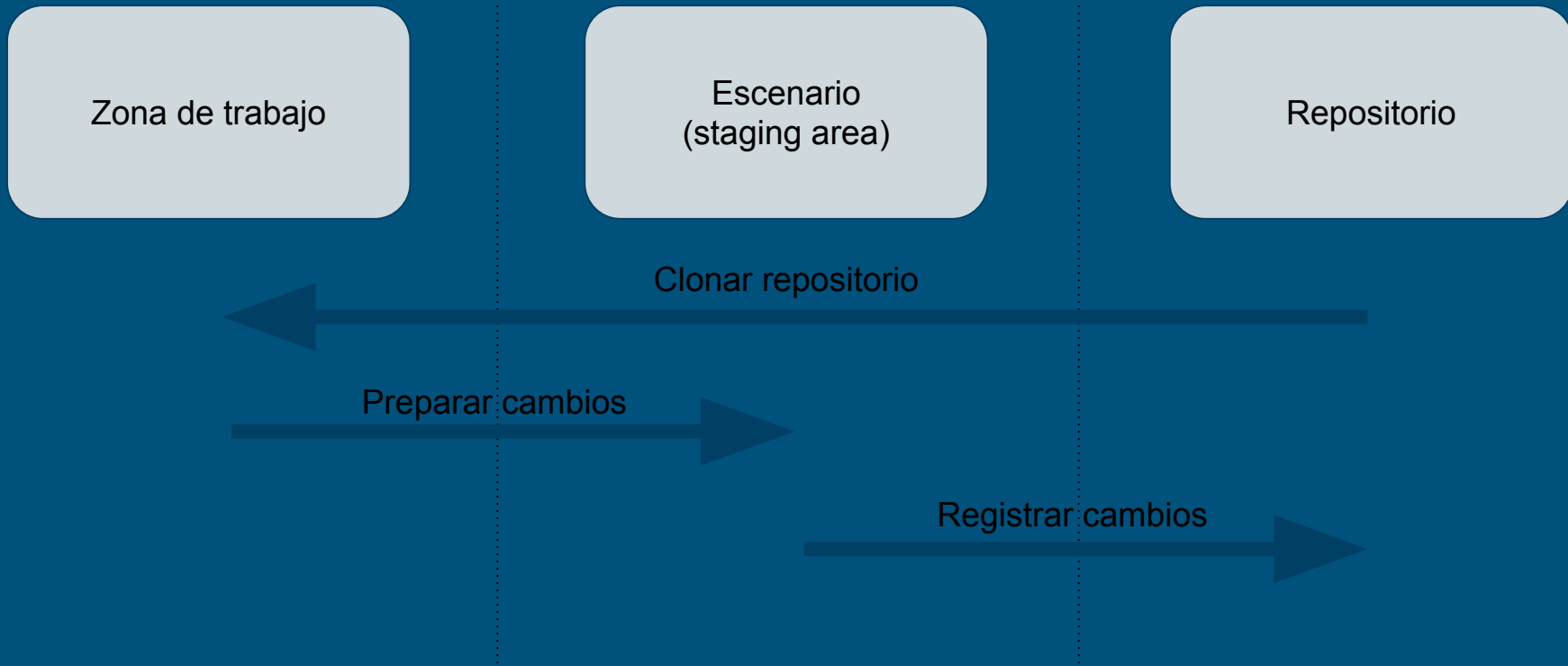
Registrar los cambios que escribes en el código

Ver una evolución del código

Volver a versiones anteriores

Trabajo paralelo entre compañeros sin problemas

Escenarios en un repositorio



Instalación

Multiplataforma

Distribuido para la mayoría de sistemas operativos

Guía de instalación:

<http://git-scm.com>

Configuración inicial

```
$ git config <--global> user.name  
    "First Name"
```

```
$ git config <--global> user.email  
    "my@email.com"
```

Comandos iniciales

Inicialización de repositorio:

```
$ git init
```

Registro de cambios:

```
$ git log
```

Clonación de un repositorio:

```
$ git clone "url"
```


Preparando cambios

Estado del repositorio:

```
$ git status
```

Añadiendo ficheros al cambio:

```
$ git add <--all | . | ruta/>
```

Eliminando ficheros del cambio:

```
$ git rm <--cached> <--all | . |  
ruta/>
```

Control de los ficheros y directorios del repositorio:

```
.gitignore
```

Ejemplo de contenido:

```
.idea/tab.xml  
bower_components/  
*.pyc
```

Registrando cambios

Registro de cambios:

```
$ git commit <-a|-m "Mensaje...">
```

Resetear estado del repositorio:

```
$ git reset <--soft|--mixed|--hard>  
HEAD<^|^^|...> <css/style.css>
```

Revertir cambios:

```
$ git revert <hash>
```

Branch - Ramas alternativas

Todas las ramas:

```
$ git branch
```

Crear:

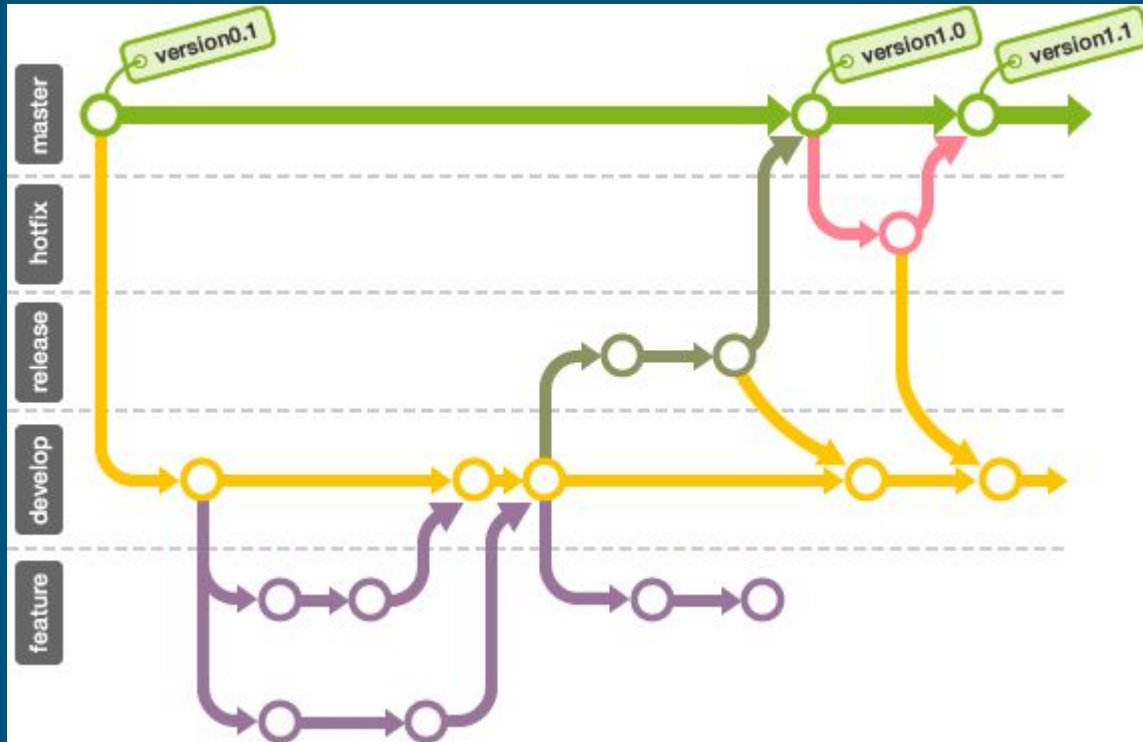
```
$ git checkout -b <branch name>
```

Eliminar:

```
$ git checkout -d <branch name>
```

Pasar:

```
$ git checkout <branch name>
```



Fusión de ramas y aparición de conflictos

Fusión:

```
$ git merge <--no-ff> <branch name>
```

Forma de un conflicto:

```
El número de planetas son
```

```
<<<<<< HEAD
```

```
nueve
```

```
=====
```

```
ocho
```

```
>>>>>> <branch_name>
```

```
en el sistema solar.
```

Trabajando en remoto

Añadiendo repositorios remotos:

```
$ git remote add <name> <url>
```

Eliminando repositorios remotos:

```
$ git remote rm <name>
```

Cambiando de nombres:

```
$ git remote rename <old> <new>
```

Revisar rama remota:

```
$ git checkout -b <rama> <repositorio>/<rama_remota>
```

Crear rama remota:

```
$ git push <repositorio> <rama>
```

Eliminando rama remota:

```
$ git push <repositorio> --delete <rama>
```

```
$ git push <repositorio> :<rama>
```

Manteniendo actualizados los cambios

Descargando del repositorio:

```
$ git fetch [--all] [<options>] <repository>
```

Actualizando cambios en local:

```
$ git pull [options] <repository>
```

Actualizando cambios en remoto:

```
$ git push [options] <repository>
```

Como resolver un **Pull request** en local
(fusionar origen en destino):

Sincronizamos la rama origen con el remoto:

```
$ git fetch <repositorio>
$ git checkout -b <rama_origen> <repositorio>/<rama_origen>
$ git merge <rama_destino>
```

Resolvemos conflictos si existen, y fusionamos:

```
$ git checkout <rama_destino>
$ git merge --no-ff <rama_origen>
$ git push origin <rama_destino>
```

Cambios en un limbo temporal

Para guardar los cambios actuales en nuestro espacio de trabajo sin necesidad de registrarlos mediante un commit, y así poder cambiarnos de rama para trabajar en otros aspectos de nuestro proyecto sin problema, como movernos de rama.

Listar guardados temporales

```
$ git stash list
```

Mostar los cambios de un stash

```
git stash show stash
```

Añadir un estado de stars

```
$ git stash add
```

Recuperar los cambios

```
$ git stash (pop | apply) [<stash>]
```

Eliminar un stash

```
$ git drop <stash>
```

Preguntas

Autor:

Ismael J. Taboada



@ismtabo



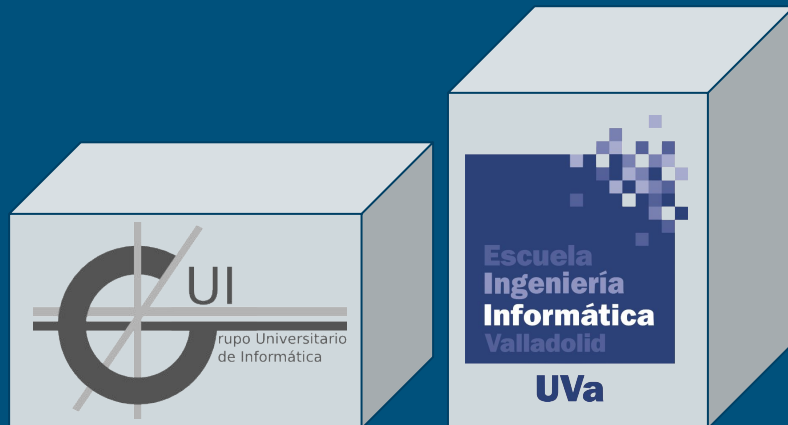
@isma_jtr

Esta obra está bajo una

[Licencia Creative Commons Atribución-CompartirIgual 4.0 Internacional.](#)

Basado en las transparencias de

Javier Provecho (GIT Talk UVa)



Contacto GUI:



www.gui.uva.es



foro.gui.uva.es



[@gui_uva](https://twitter.com/gui_uva)



telegram.me/gui_uva