Introducción a Git

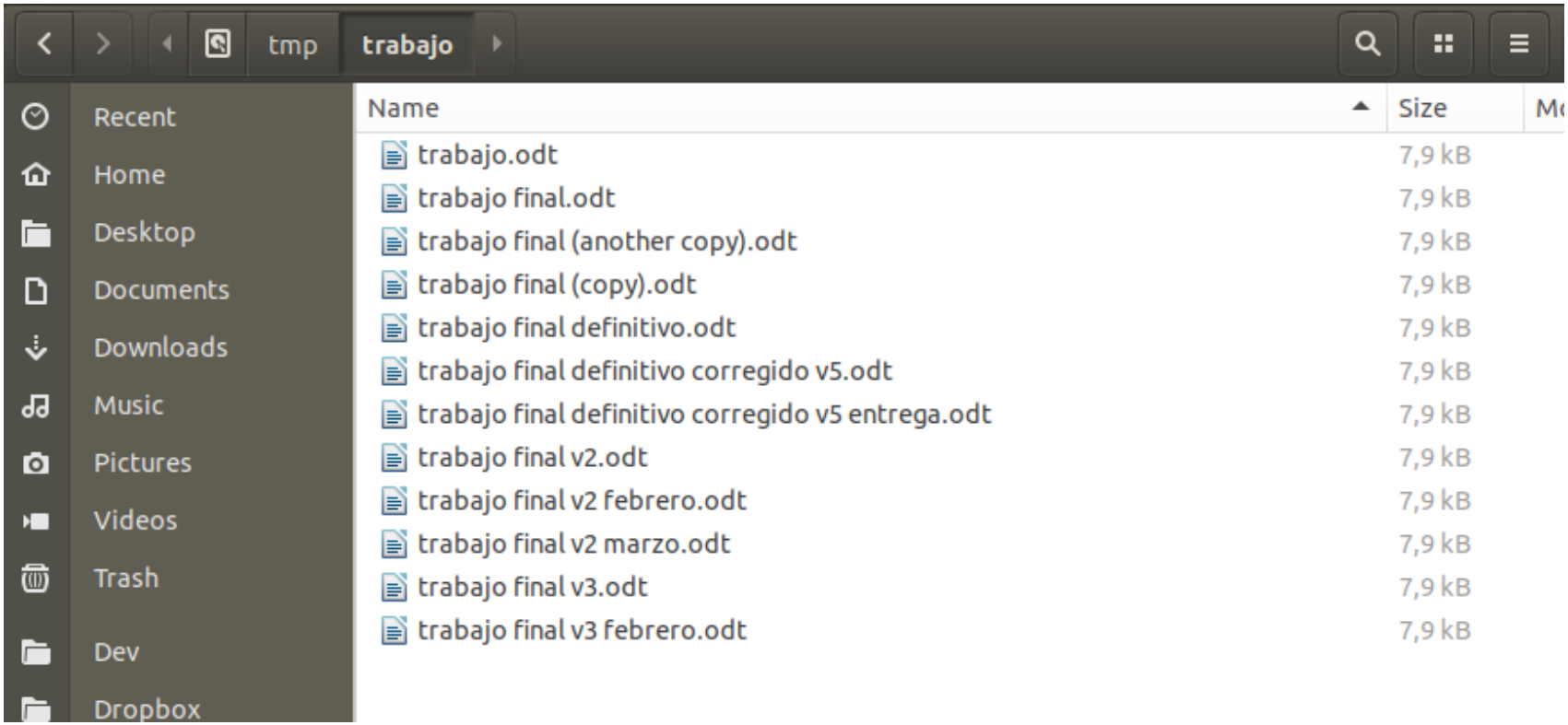
Rubén Crespo Cano

@rcrespocano

¿Qué es git?

- Sistema de control de versiones (VCS)
- Sistema distribuido
- Creado por Linus Torvalds para Linux (2005)
- Interfaz de línea de comandos (aunque también existen interfaces gráficas)

Sobre control de versiones

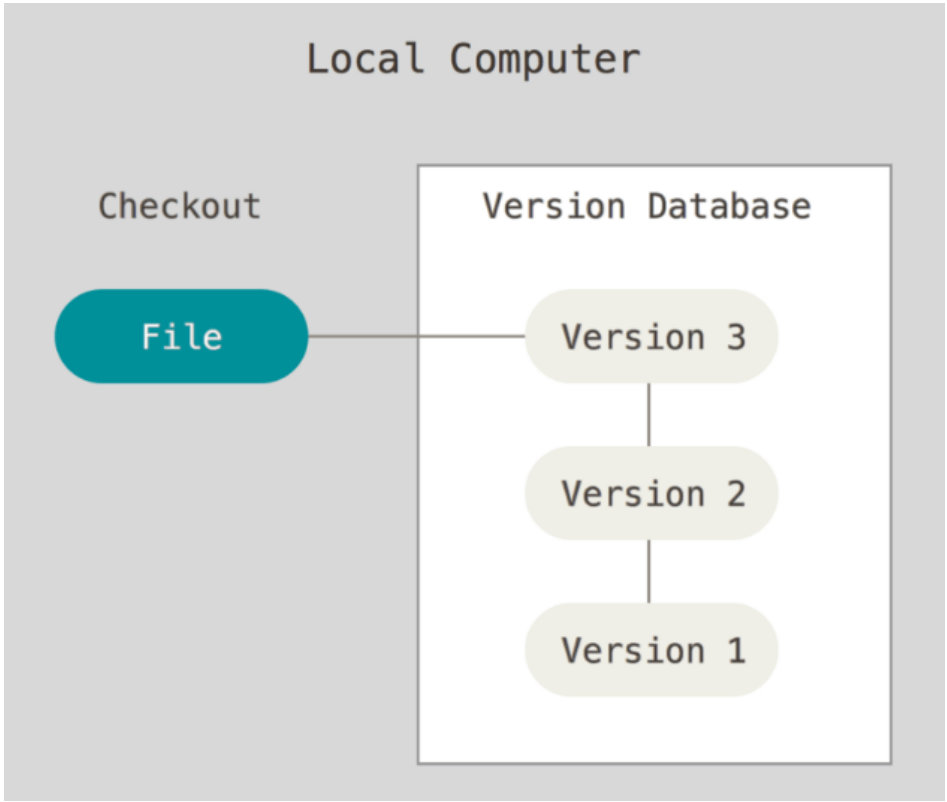


Sobre control de versiones

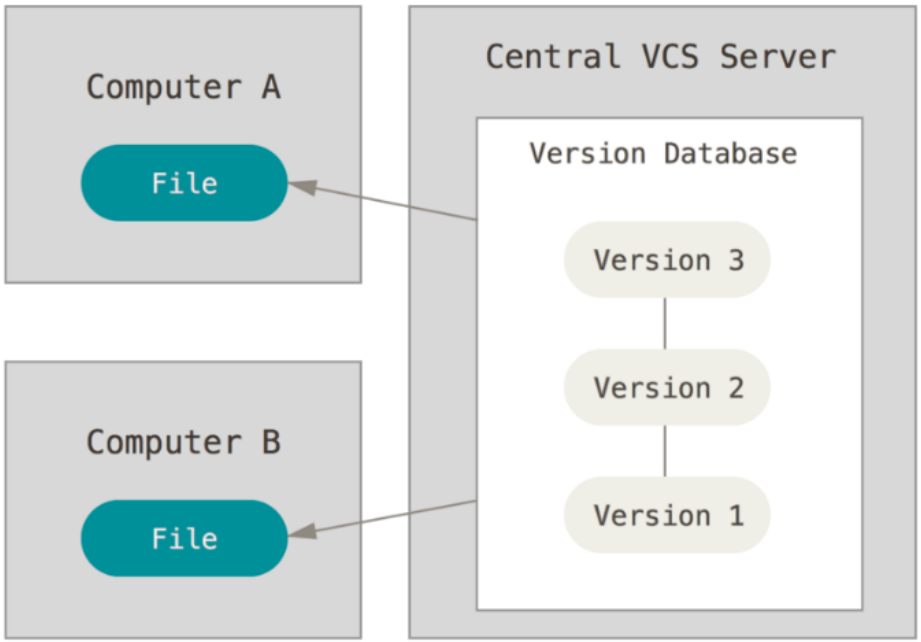
Un sistema de control de versiones registra y almacena cambios en ficheros a lo largo del tiempo para mejorar la gestión y poder recuperar versiones específicas en cualquier momento.

- Sistema de control de versiones local
- Sistema de control de versiones centralizado
- Sistema de control de versiones distribuido

Sistema de control de versiones local



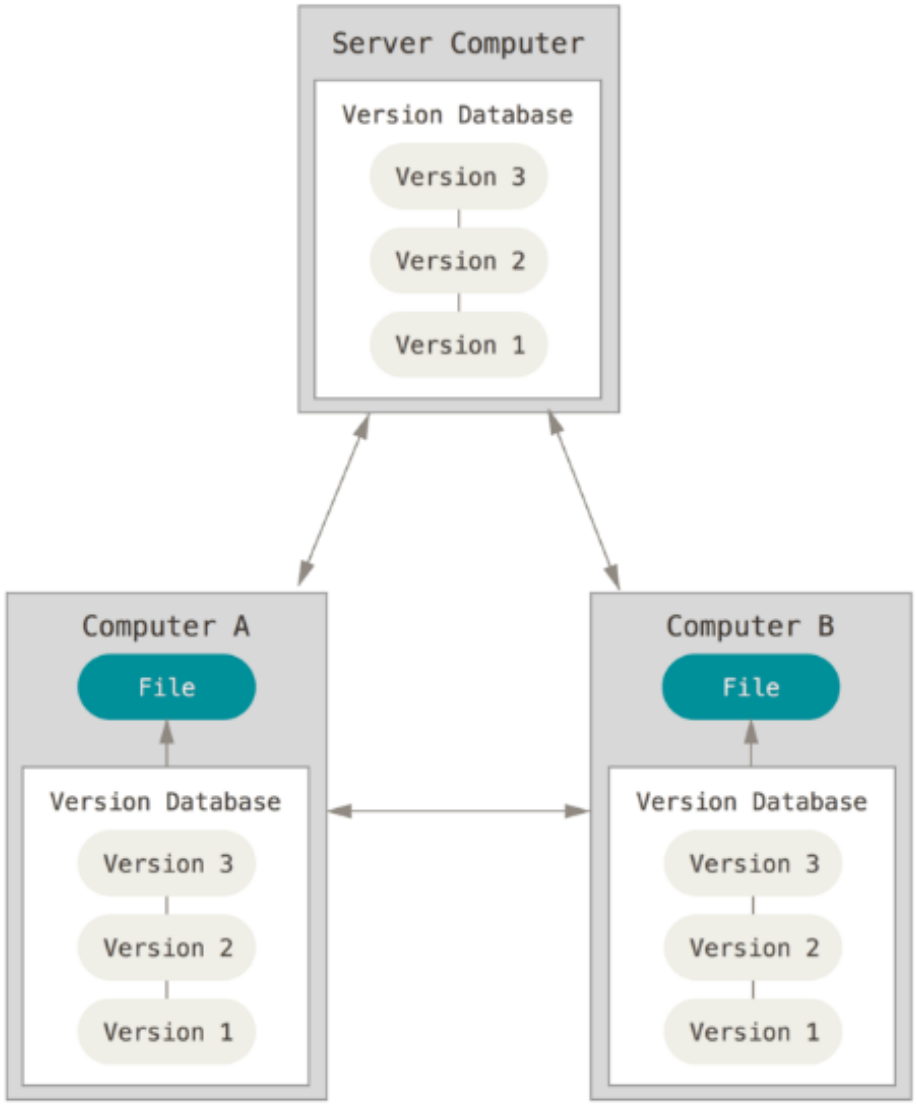
Sistema de control de versiones centralizado



Sistema de control de versiones centralizado

- Un servidor centralizado contiene todos los ficheros versionados y los metadatos.
- Ventajas: colaboración, administración y sencillez
- Inconvenientes: centralización, disponibilidad, copias de seguridad, velocidad, etc.

Sistema de control de versiones distribuido



Sistema de control de versiones distribuido

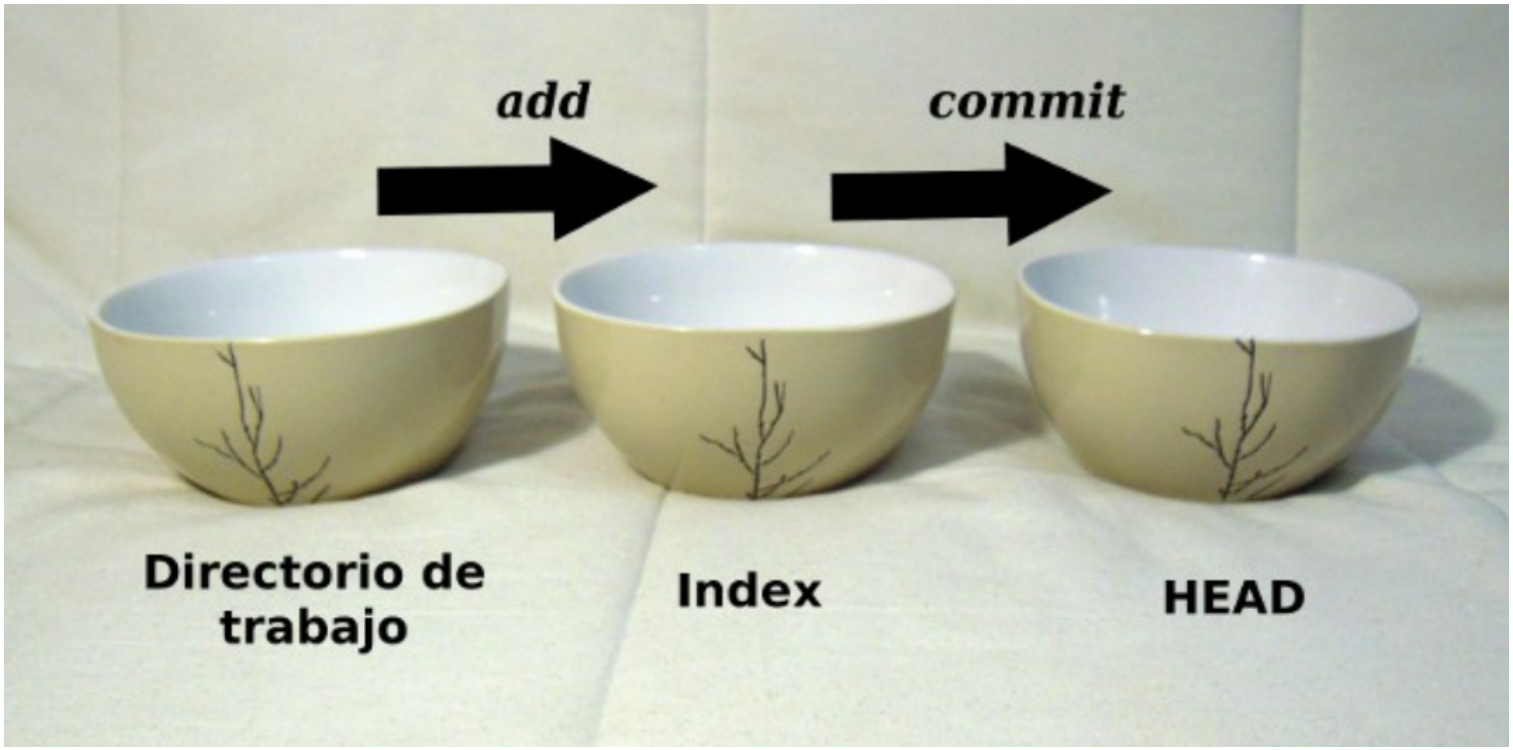
- Cada cliente tiene una réplica exacta del repositorio (ficheros, metadatos, historial, etc.)
- Si el servidor muere cualquier copia puede reemplazar al servidor central
- Colaboración con distintos grupos de forma descentralizada
- Ventajas
  - Velocidad
  - Diseño simple
  - Soporte para desarrollo no lineal (cientos de ramas paralelas)
  - Totalmente distribuido
  - Capaz de gestionar grandes proyectos de forma eficiente (velocidad y tamaño de datos)
- Inconvenientes: curva aprendizaje

¿Quién utiliza git?

Microsoft, Amazon, LinkedIn, Cisco, IBM, Accenture, Facebook, Yahoo, Apple, T-Mobile, Lenovo, Atlassian, y muchas otras.

- https://git-scm.com/
- https://www.quora.com/What-companies-use-Git

Flujo de trabajo



Instalación

GNU/Linux/Unix

https://git-scm.com/download/linux

Mac OS X

https://git-scm.com/download/mac

Windows

https://git-scm.com/download/win

Configuración básica

Nombre de usuario

```
git config --global user.name "Obi-Wan Kenobi"
```

Email

```
git config --global user.email "ninjacoder@email.com"
```

Activación colores

```
git config --global color.ui true
```

Configuración SSH

Generar el par clave pública/privada

```
ssh-keygen
```

Añadir la clave al ssh-agent

```
eval `ssh-agent`
ssh-add ~/.ssh/<private_key_file>
```

Añadir la clave pública al servidor centralizado

```
cat ~/.ssh/id_rsa.pub
```

Cómo crear un nuevo repositorio

Para crear un nuevo repositorio de git, crea un directorio nuevo, accede a él y ejecuta el siguiente comando:

Comando

```
git init
```

Ayuda

Para obtener más información sobre un comando, ejecuta el siguiente comando:

Comando

```
git --help <command>
```

Cómo añadir un origen remoto

Para añadir un origen remoto, ejecuta el siguiente comando:

Comando

```
git remote add <remote-name> <remote-URL>
```

Parámetros:

- Nombre remoto, por ejemplo, *origin*
- URL remota, por ejemplo, *https://github.com/user/repo.git*

Ejemplo

```
git remote add origin https://github.com/user/repo.git
```

Cómo clonar un repositorio

Para realizar una copia local de un repositorio, ejecuta el siguiente comando:

Comando

```
git clone </path/to/repository>
```

Para clonar un repositorio remoto, ejecuta el siguiente comando:

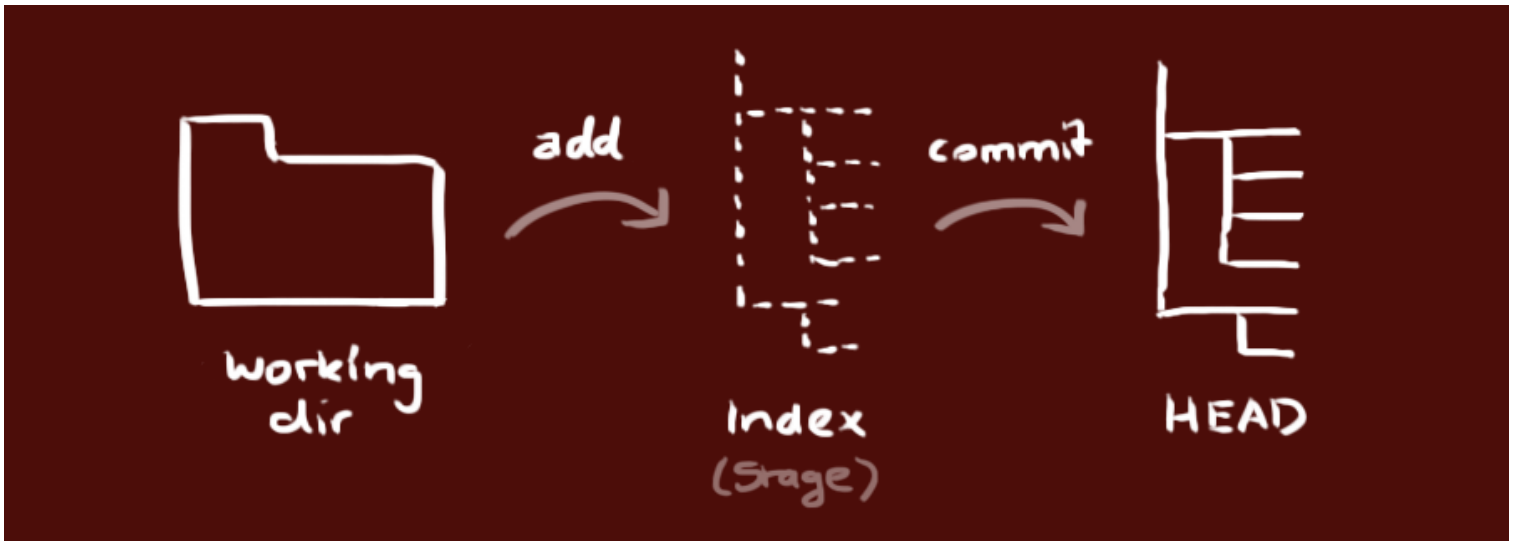
Comando

```
git clone <URL>
```

Flujo de trabajo

Cada repositorio local está compuesto por tres *árboles* administrados por git:

- Directorio de trabajo
- Index
- HEAD



Cómo registrar cambios en el repositorio

Para registrar nuevos cambios y añadirlos al **Index**, ejecuta el siguiente comando:

Comando(s)

```
git add <filename>
git add <pattern.*>
git add <folder>
git add <.>
```

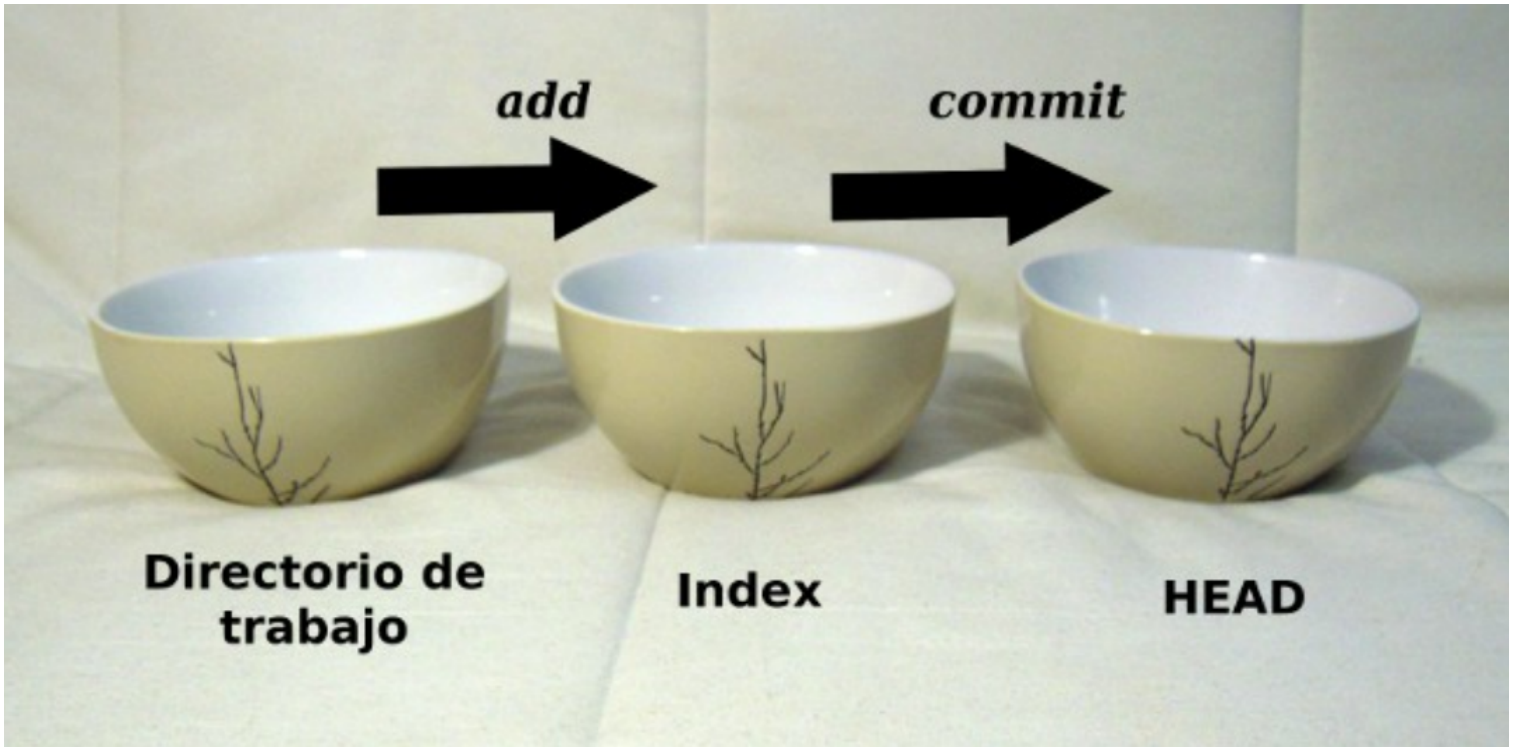
Para guardar dichos cambios en el repositorio, ejecuta el siguiente comando:

Comando

```
git commit -m ‘‘Commit message’’
```

Tras este proceso los cambios estarán en el **HEAD**, pero todavía no estarán en el repositorio remoto

Flujo de trabajo



Cómo borrar ficheros

Para borrar un fichero, ejecuta el siguiente comando:

Comando

```
git rm <file>
```

Para borrar un fichero del **Index**, ejecuta el siguiente comando:

Comando

```
git rm --cached <file>
```

Cómo mover o renombrar ficheros

Para mover o renombrar un fichero, ejecuta el siguiente comando:

Comando

```
git mv <file>
```

Deshacer cambios en un archivo

Para recuperar la versión del HEAD, ejecuta el siguiente comando:

Comando

```
git checkout -- <file>
```

Cómo conocer el estado actual

Para conocer el estado actual del repositorio, ejecuta el siguiente comando:

Comando

```
git status
```

Para ver las diferencias, ejecuta el siguiente comando:

Comando(s)

```
git diff
git diff <file>
```

Historial

Para mostrar el historial de todos los cambios efectuados, ejecuta el siguiente comando:

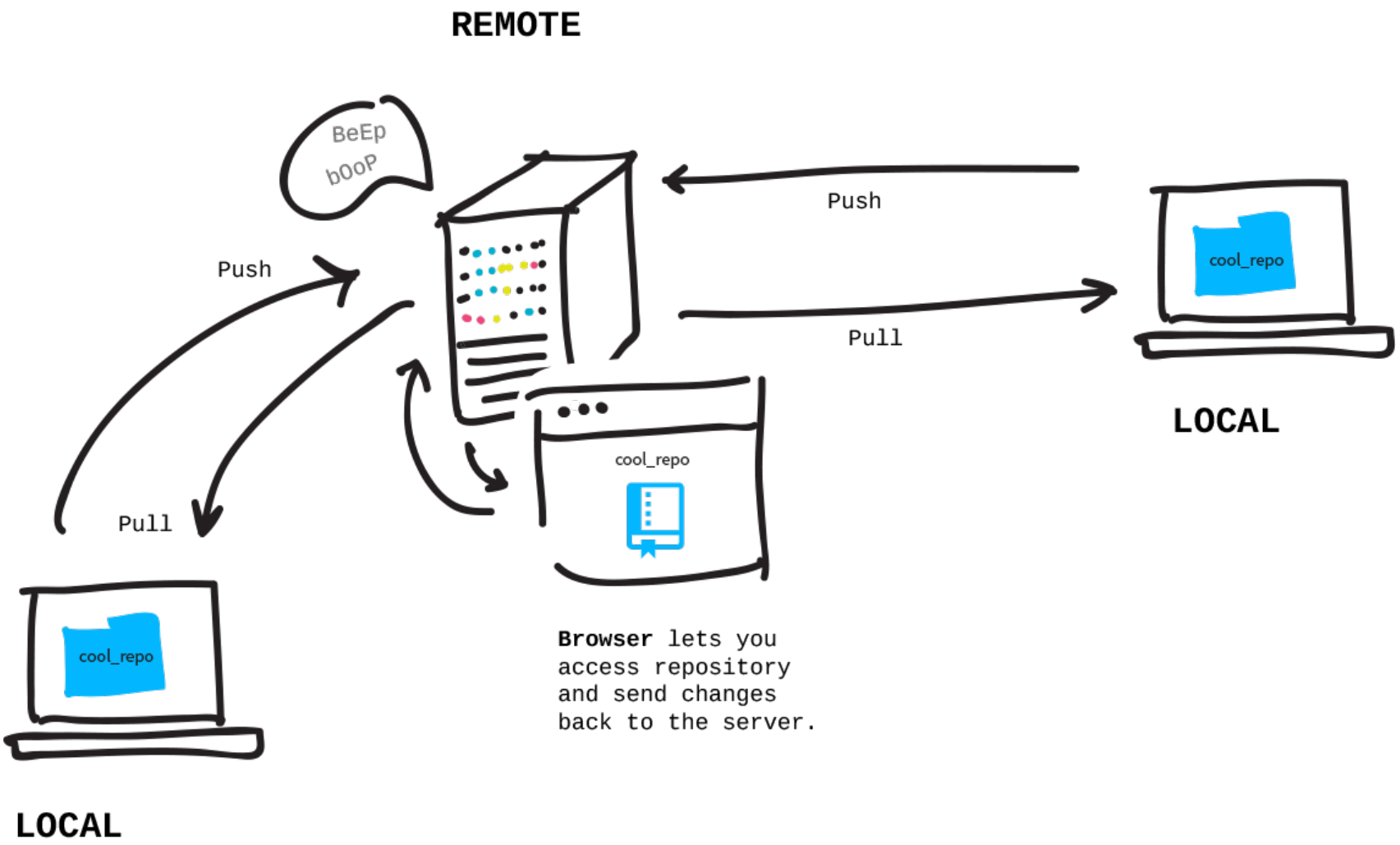
Comando

```
git log [-N]
```

Otros Parámetros:

- –oneline
- –graph
- -p

Cómo sincronizar con un servidor remoto



Cómo enviar los cambios al servidor remoto

Tras registrar los cambios en el **HEAD** de la copia local, para enviar los cambios al repositorio remoto ejecuta el siguiente comando:

Comando(s)

```
git push origin master
git push
```

Si se desea registrar los cambios de otra rama, reemplaza *master* por la rama deseada.

Comando(s)

```
git push origin <branch>
```

Cómo actualizar el repositorio local

Para actualizar el repositorio local, ejecuta el siguiente comando:

Comando(s)

```
git pull origin master
git pull origin <branch>
git pull
```

Cómo gestionar el conflicto entre repositorios

- Git informa detalladamente del problema
- El usuario deberá arreglar el problema
- El usuario deberá hacer *commit* y *push*

Etiquetas

Para listar las etiquetas, ejecuta el siguiente comando:

Comando

```
git tag
```

Para crear una etiqueta, ejecuta el siguiente comando:

Comando

```
git tag v0.0.2
```

Para crear una etiqueta con una anotación, ejecuta el siguiente comando:

Comando

```
git tag -a v0.0.2 -m ‘‘New software version.’’
```

Archivo .gitignore

A menudo, hay archivos que no se desean que sean añadidos al repositorio: passwords, temporales, binarios compilados, archivos de configuración local, etc.

- Git permite utilizar archivos en los que se indica qué archivos debe ignorar
- Archivo: **.gitignore**
- Uso de expresiones regulares y comentarios

Ejemplo

```
# Byte-compiled / optimized / DLL files
__pycache__/
*.py[cod]
```

Ramas

Para crear una nueva rama, ejecuta el siguiente comando:

Comando

```
git branch <new-branch-name>
```

Para moverse a la rama creada, ejecuta el siguiente comando:

Comando

```
git checkout <new-branch-name>
```

Para volver a la rama principal, ejecuta el siguiente comando:

Comando

```
git checkout master
```

Ramas

Para borrar una rama, ejecuta el siguiente comando:

Comando

```
git branch -d <new-branch-name>
```

Para enviar una rama al repositorio remoto, ejecuta el siguiente comando:

Comando

```
git push origin <new-branch-name>
```

Ramas

Para fusionar una rama a la rama activa, ejecuta el siguiente comando:

Comando

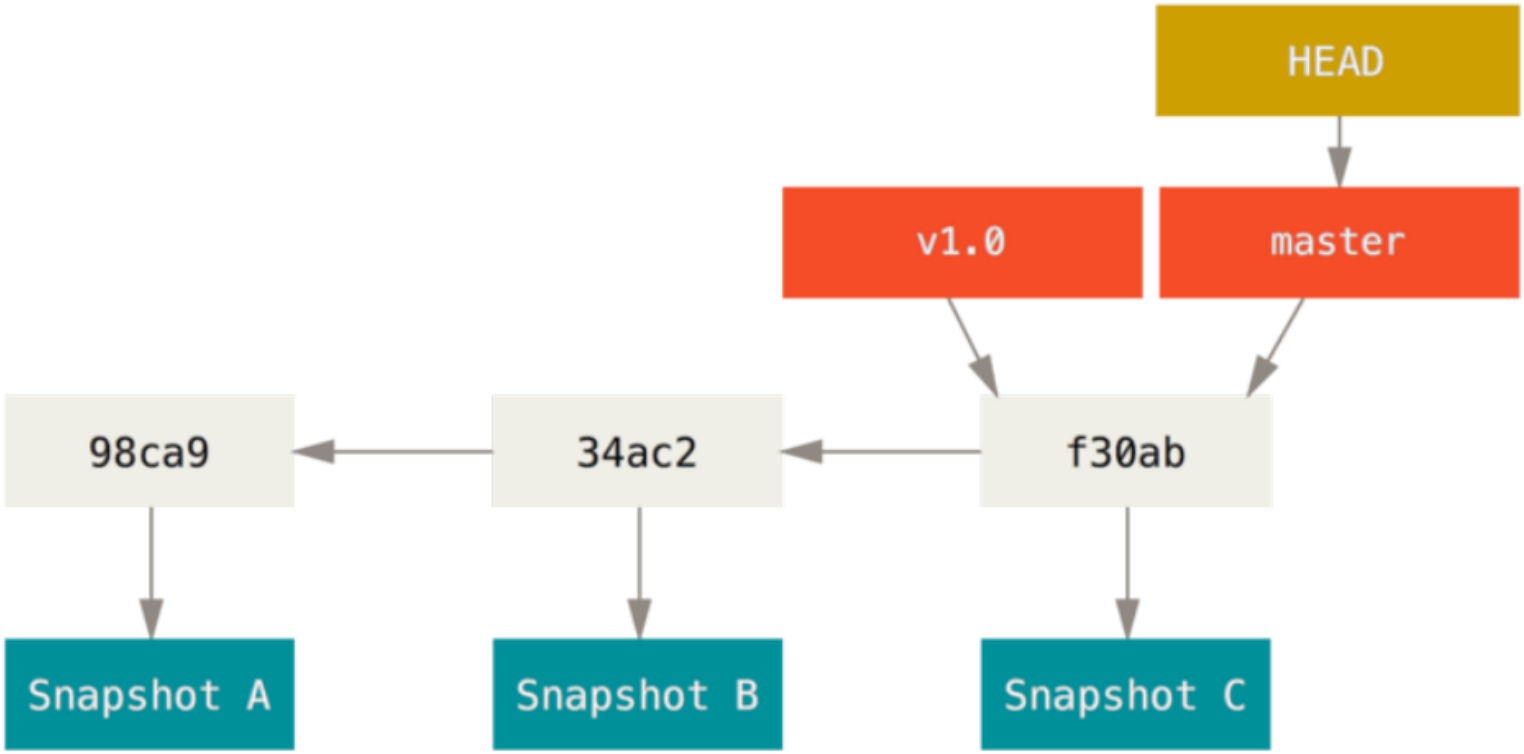
```
git merge <new-branch-name>
```

Antes de ejecutar ese comando se recomienda comparar los cambios. Para ello, ejecuta el siguiente comando:

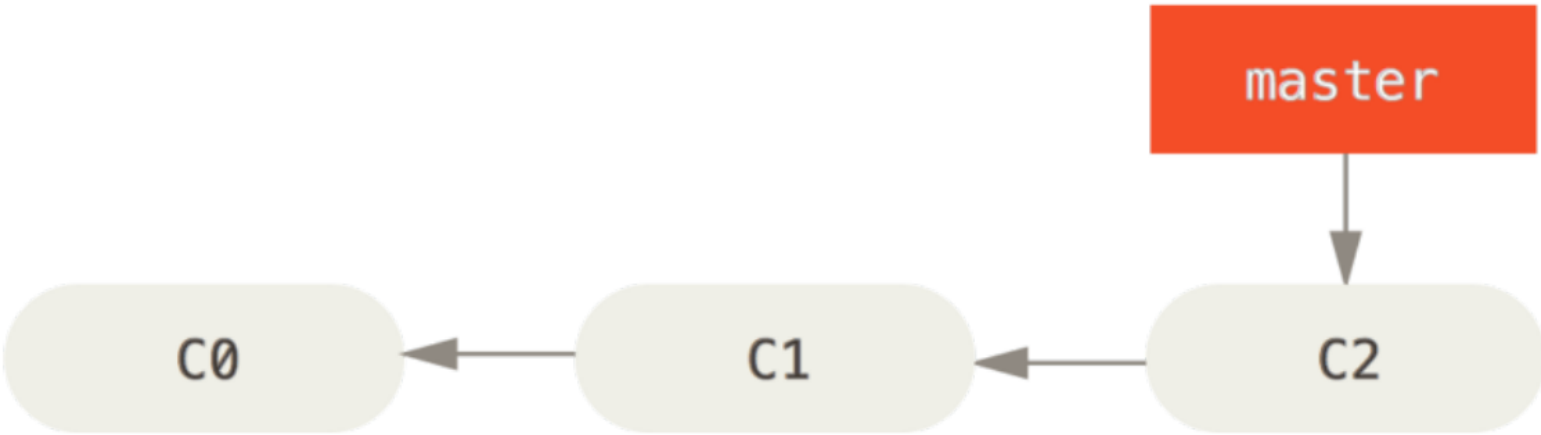
Comando

```
git diff <source-branch> <target-branch>
```

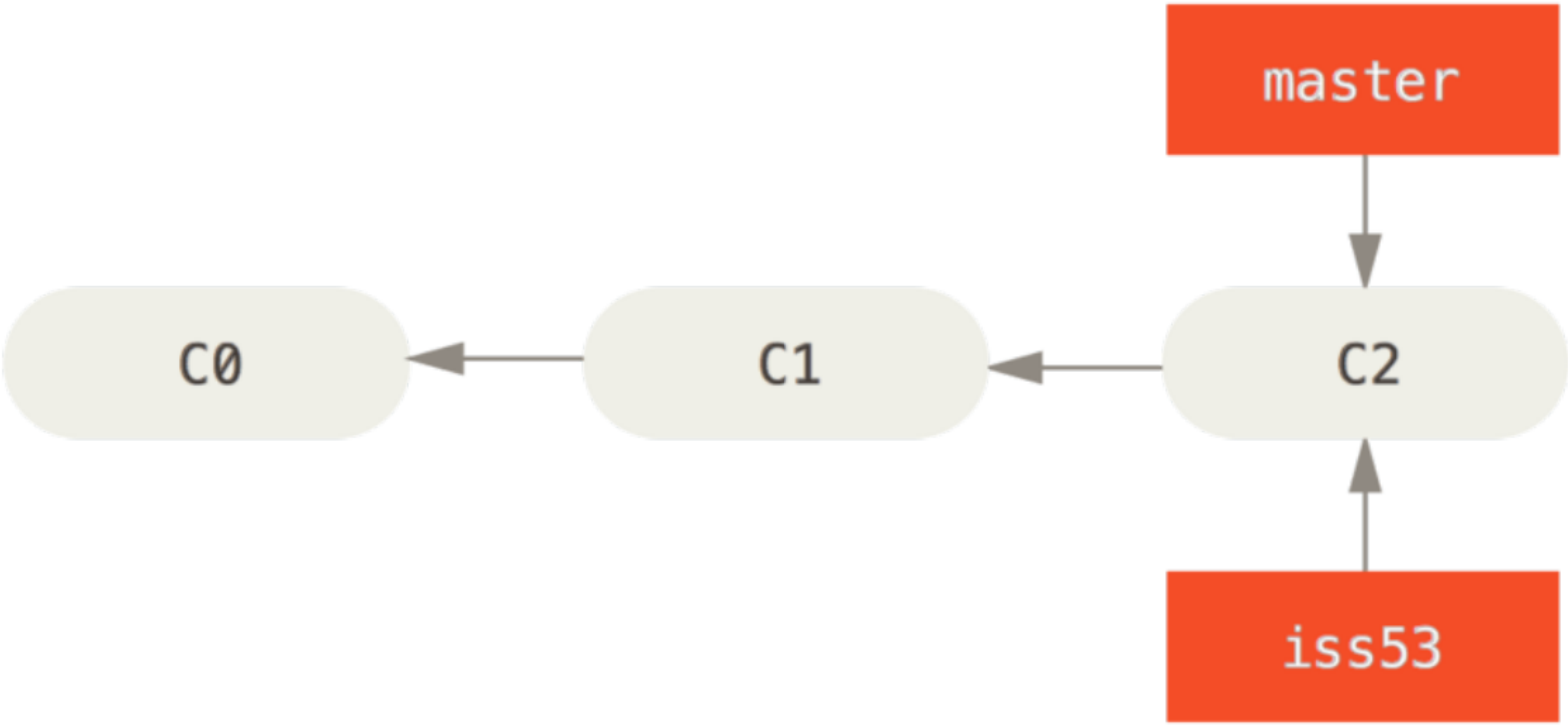
Ramas: Manejo básico



Ramas: Manejo básico



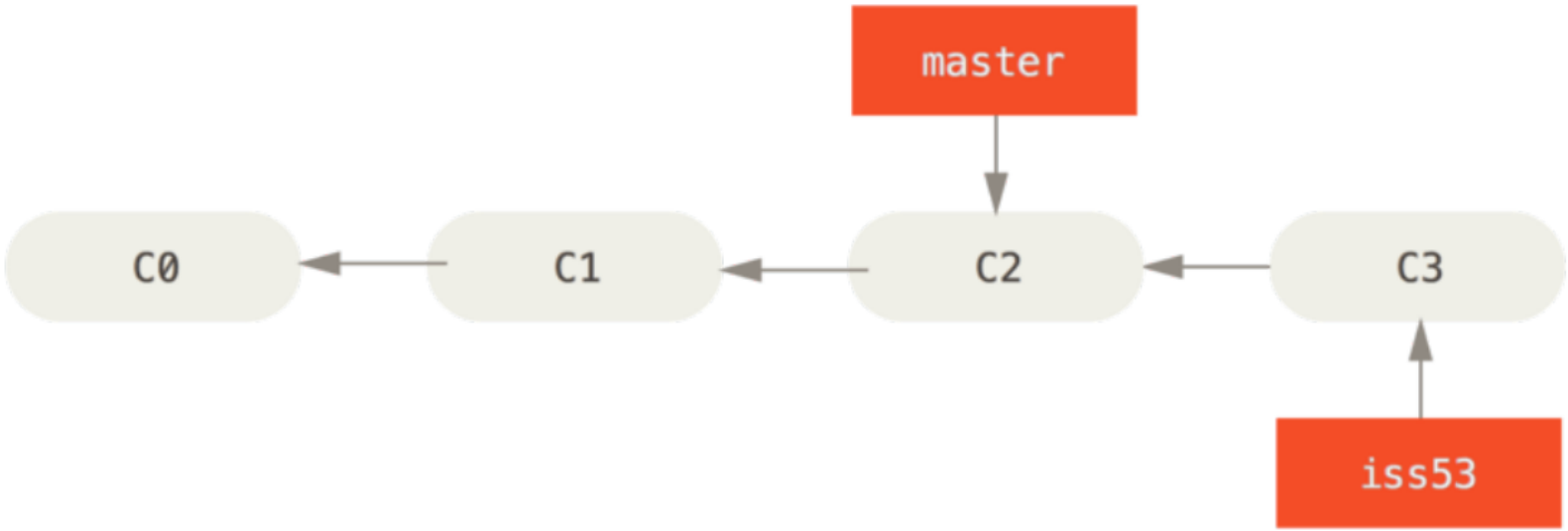
Ramas: Manejo básico



Comando(s)

```
git branch iss53
git checkout iss53
```

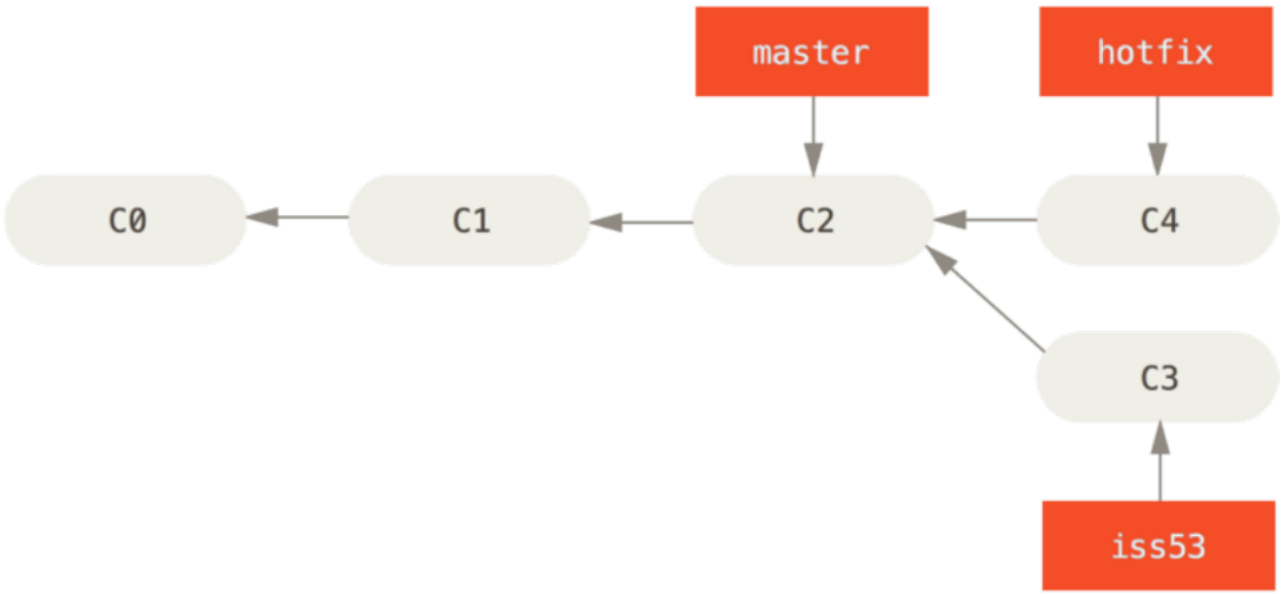
Ramas: Manejo básico



Comando(s)

```
vim index.html
git commit -a -m 'added a new footer [issue 53]'
```

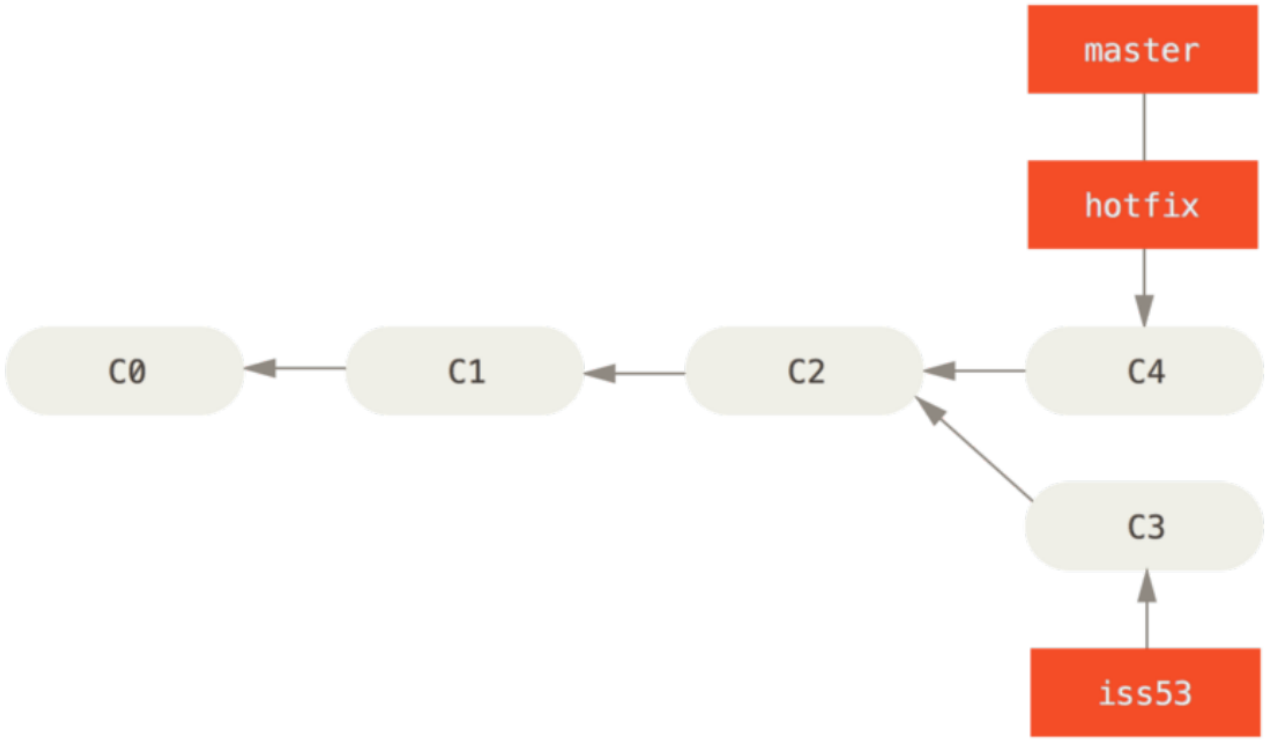
Ramas: Manejo básico



Comando(s)

```
git checkout master
git branch hotfix
git checkout hotfix
vim index.html
git commit -a -m 'fixed the broken email address'
```

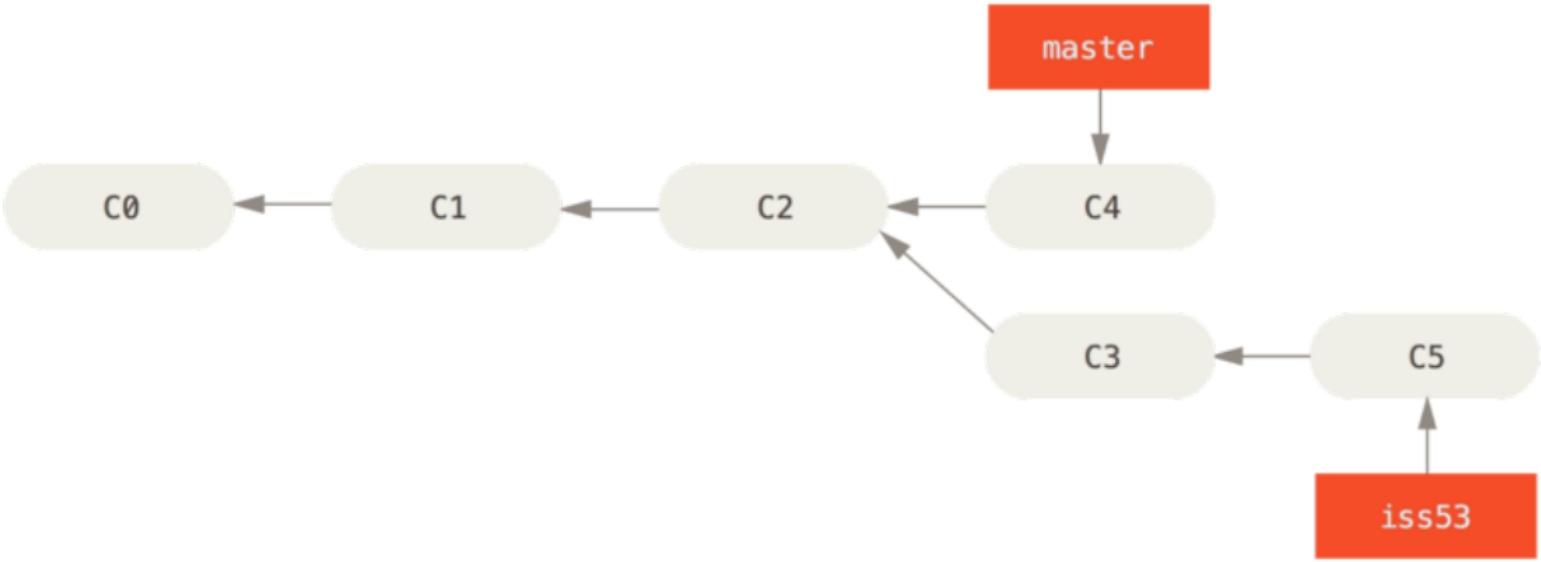
Ramas: Manejo básico



Comando(s)

```
git checkout master
git merge hotfix
```

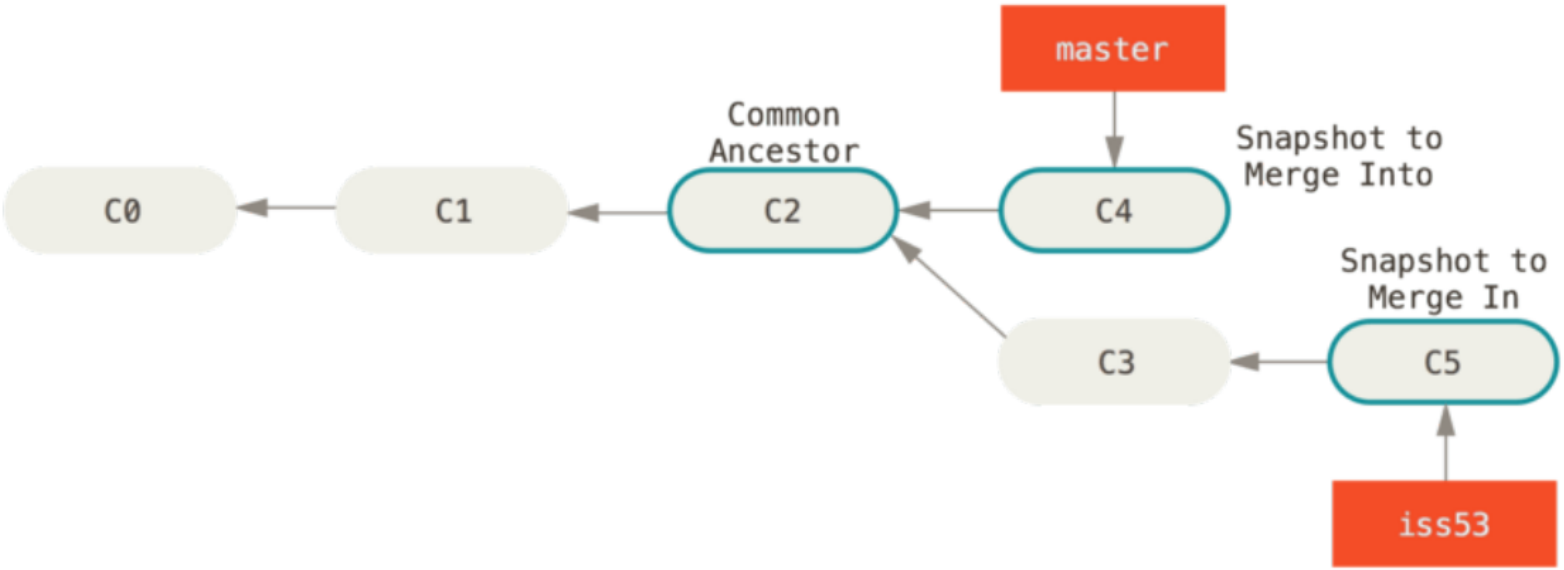
Ramas: Manejo básico



Comando(s)

```
git branch -d hotfix
---
git checkout iss53
vim index.html
git commit -a -m 'finished the new footer [issue 53]'
```

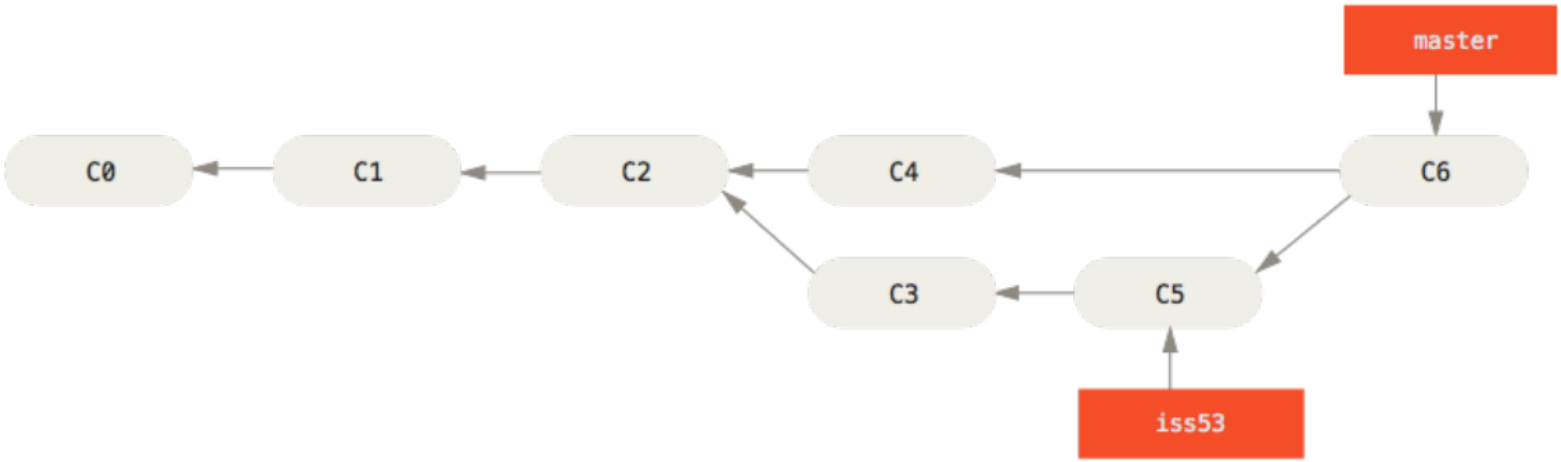
Ramas: Manejo básico



Comando(s)

```
git checkout master
git merge iss53
```

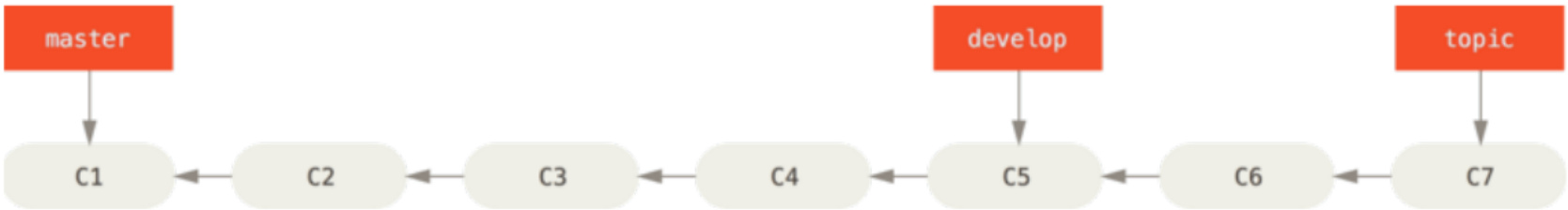
Ramas: Manejo básico

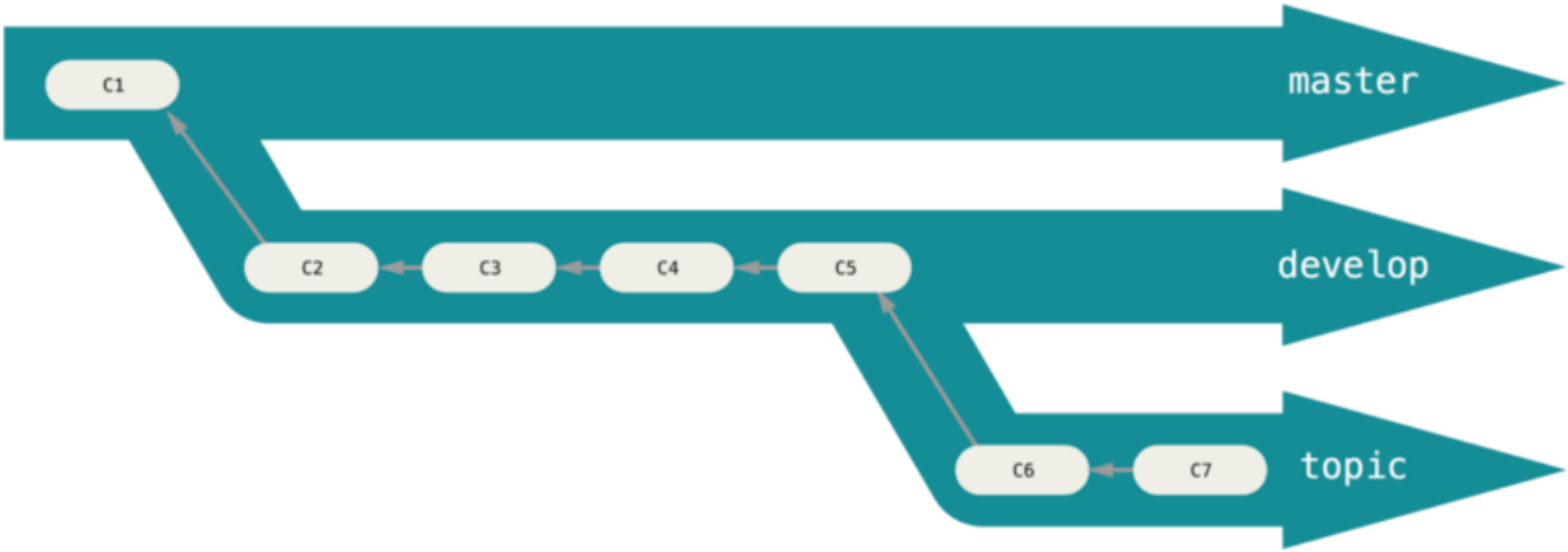


Comando(s)

```
git checkout master
git merge iss53
```

Ramas: Flujo de trabajo





Referencias

Scott Chacon y Ben Straub

Pro Git

https://git-scm.com/book/en/v2

Pablo Hinojosa y JJ Merelo

Aprende git

https://github.com/JJ/aprende-git

Angel Pablo Hinojosa Gutiérrez

El Zen de git

http://www.psicobyte.com/descargas/ZenDeGit.pdf

https://www.youtube.com/watch?v=P4RcOZycZBM

Mark Lodato

A Visual Git Reference

http://marklodato.github.io/visual-git-guide/index-en.html

Roger Dudler

Git - la guía sencilla

http://rogerdudler.github.io/git-guide/index.es.html

Víctor Suárez García

Introducción a Git

http://slides.com/zerasul/git/

Rafael Rodriguez

My Work From FreeCodeCamp

https://github.com/Rafase282/My-FreeCodeCamp-Code

¡Muchas gracias!

Rubén Crespo Cano

@rcrespocano