Sistemas de control de versiones: GIT

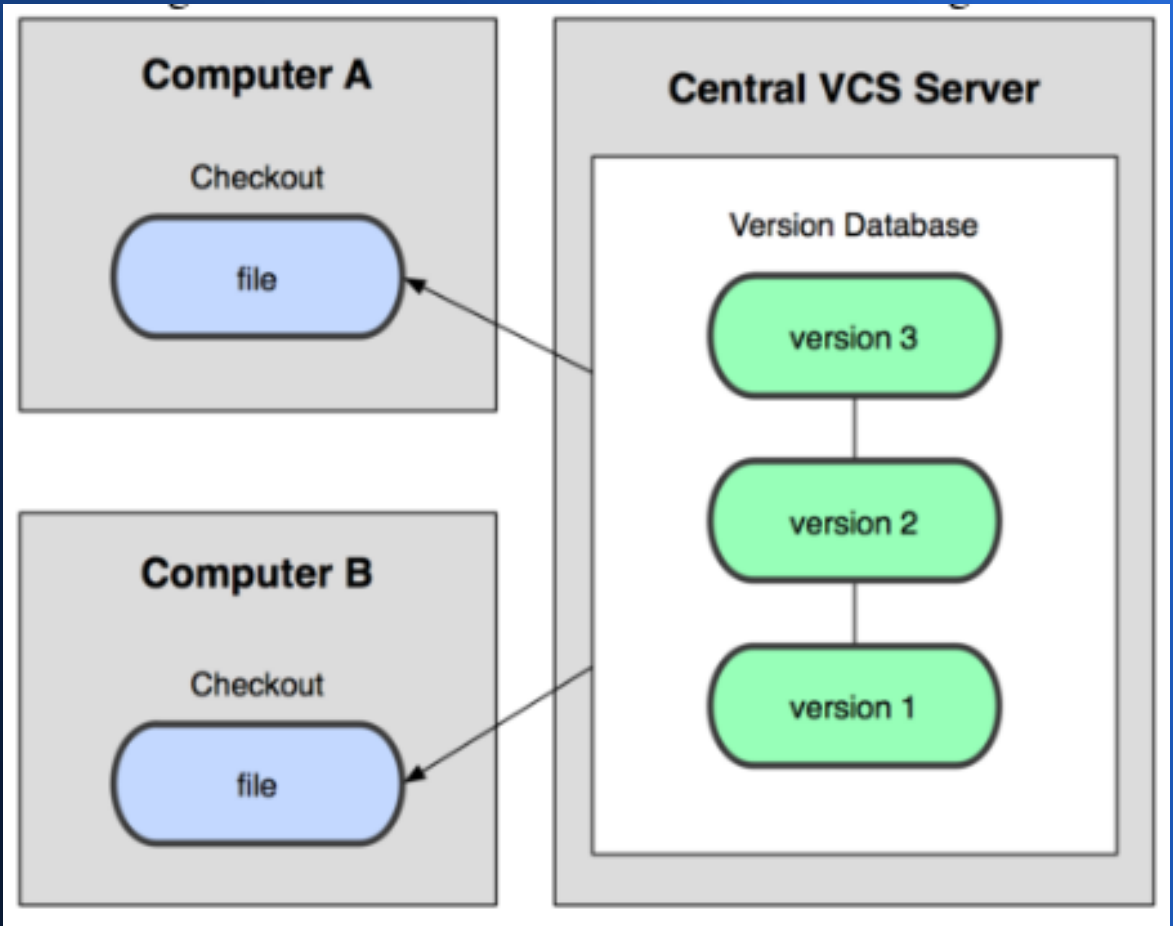
Definición

- Sistema para controlar los cambios en un proyecto.
- Internamente lleva una “base de datos” de los estados del proyecto.
- Similar conceptualmente a otros más antiguos (Subversion, VisualSourceSafe).

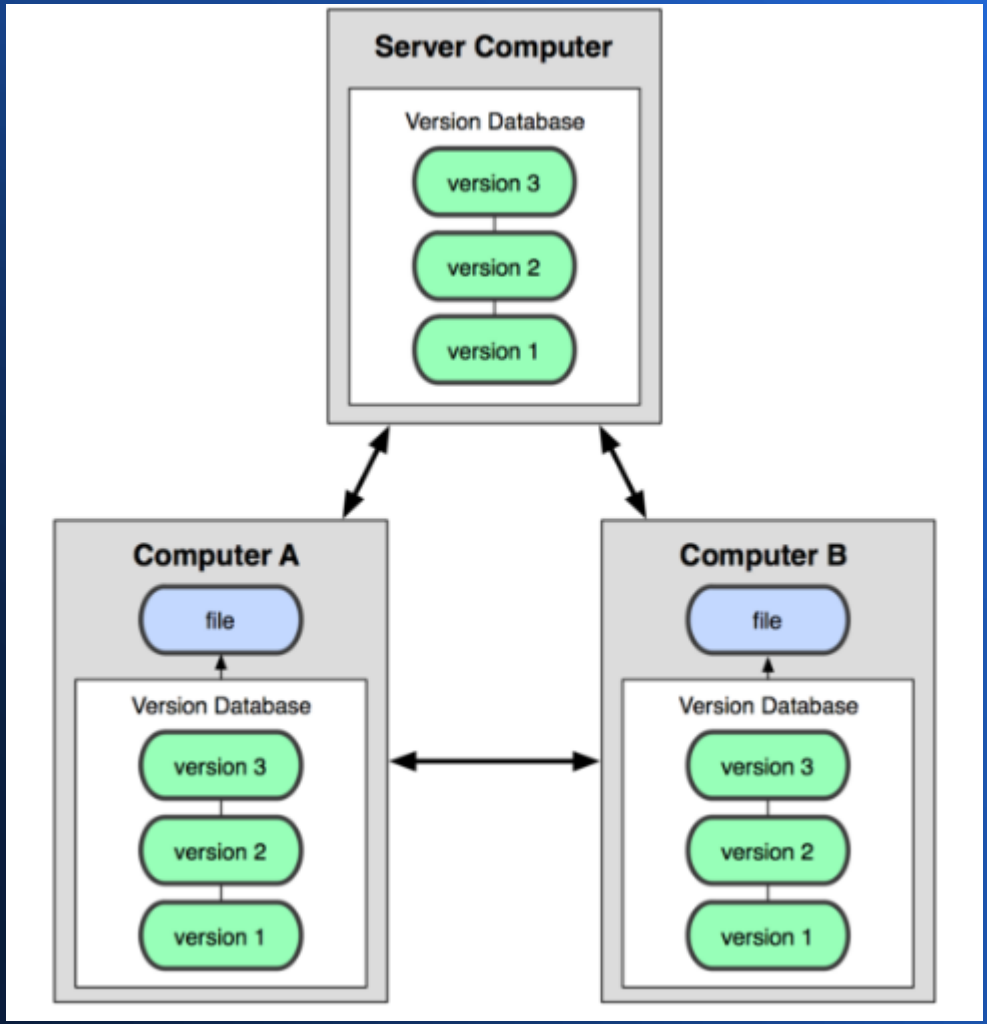
Utilidades

- Permite regresar a un estado anterior del proyecto.
- Comparar cambios hechos a lo largo del tiempo.
- Buscar el causante de problemas en el código.
- Recuperación del trabajo ante desastres.
- Permite mejor colaboración en equipos de trabajo.

Sistema centralizado



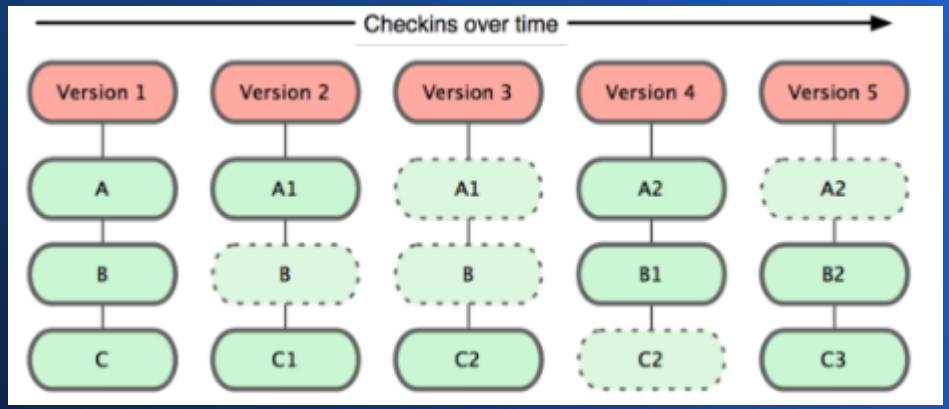
Sistema distribuido



Consideraciones de diseño de git

- Checksum (SHA-1) de contenido. El Hash se usa como identificador de la versión. Ejemplo: 24b9da6552252987aa493b52f8696cd6d3b00373
- Es simple, rápido (casi todas las operaciones son locales).
- Distribuido.
- Pensado para desarrollo no lineal (partes se llevan en paralelo).

Git


Checkins over time
Version 1
Version 2
Version 3
Version 4
Version 5
A
A1
A1
A2
A2
B
B
B
B1
B2
C
C1
C2
C2
C3

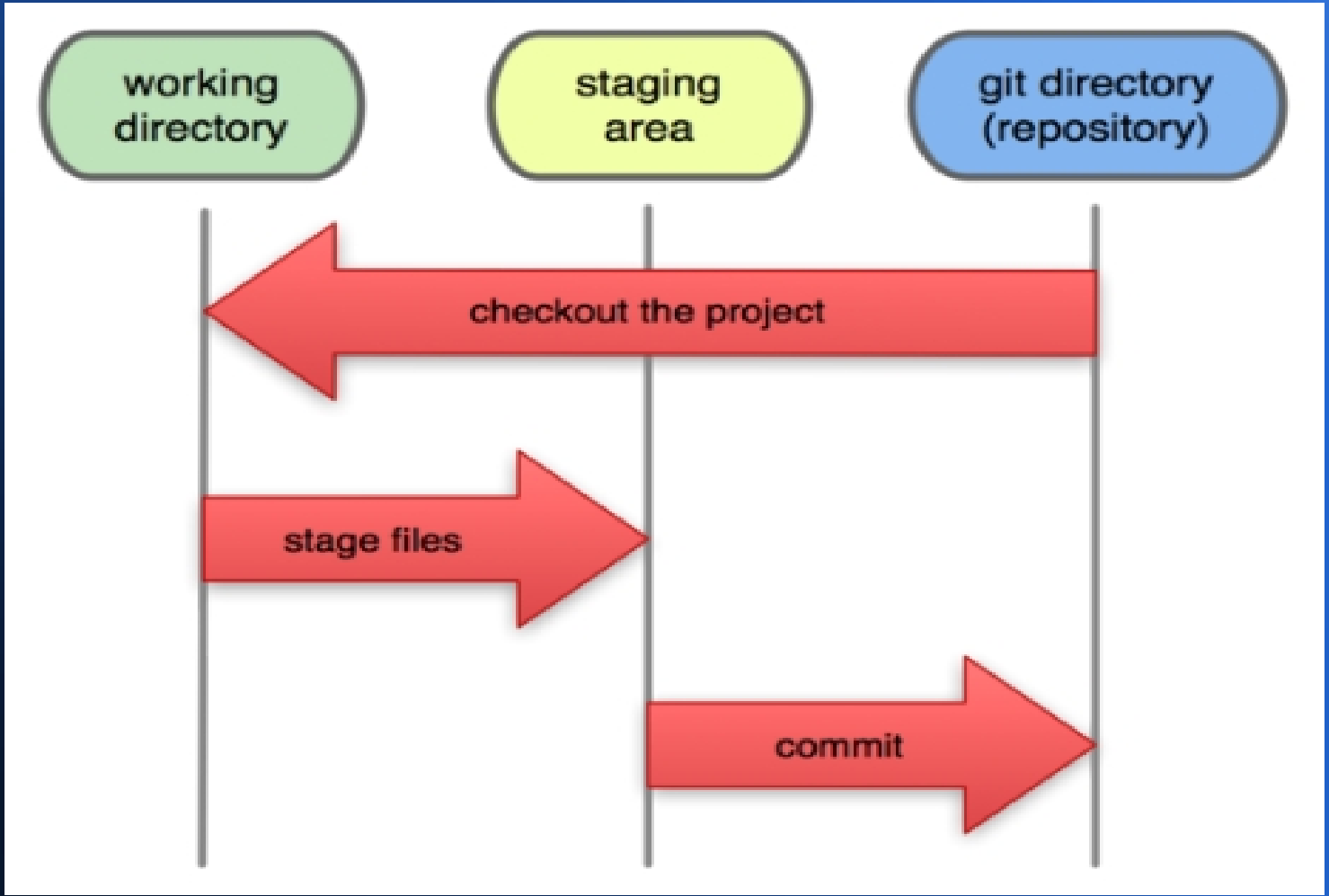
Git - Estados

- Committed: guardado en base de datos local.
- Modified: cambiado y aún no guardado en BD.
- Staged: marcado para que la versión actual sea guardado en próximo commit.

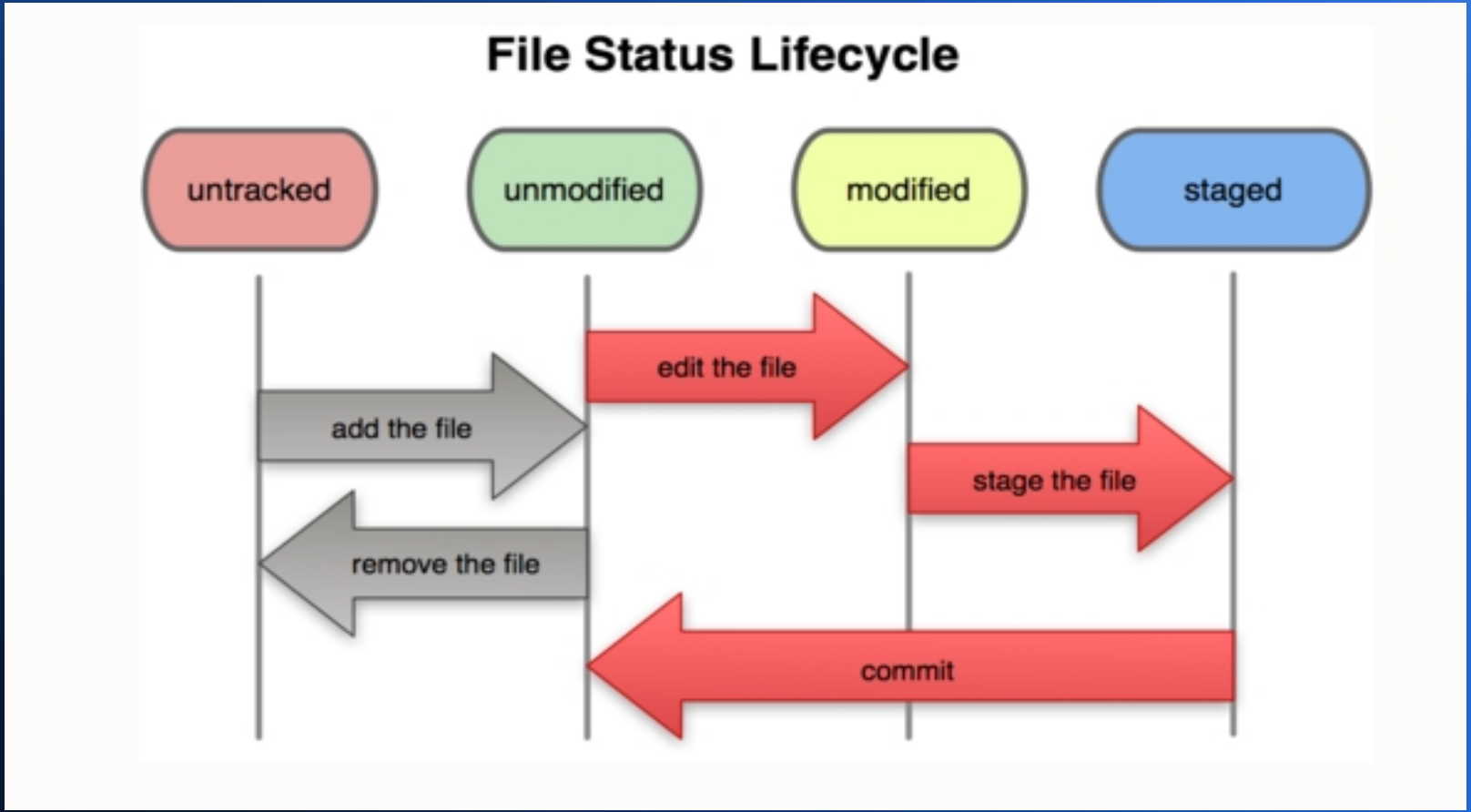
Git - directorios

- Git directory (metadatos y base de datos)
- Working directory: versión actual editable y descomprimida.
- Staging area: archivo con info de cambios que van a commit.

Git - directorios



Ciclo de vida de un archivo con git



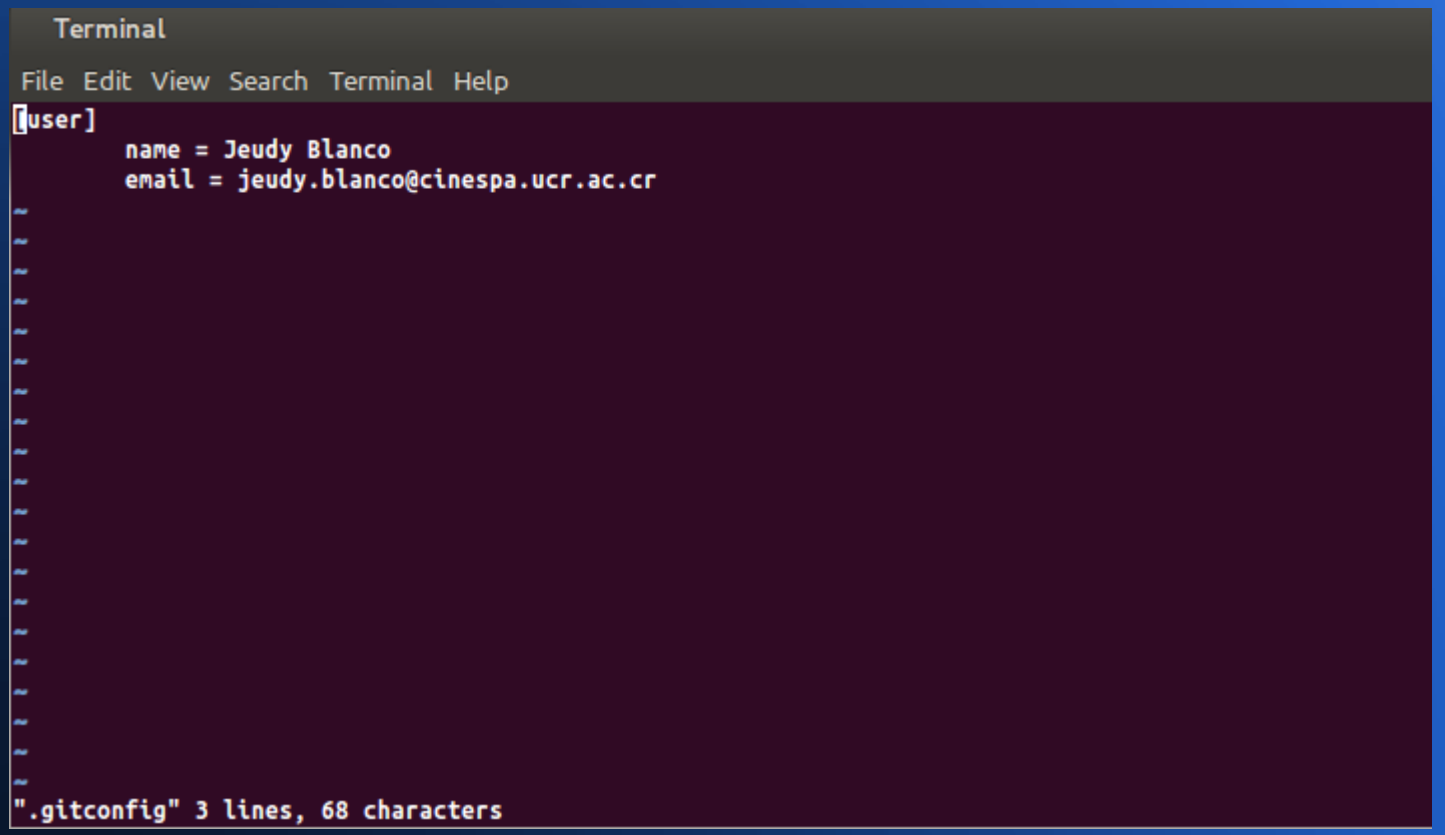
Instalación y Configuración

- apt-get install git-core
- ~/.gitconfig → configuración.
- .git/config en directorio

Configuración

- git config --global user.name “Juan Perez”
- git config --global user.email johndoe@example.com
- (sin –global para que sea solo a proyecto específico)
- git config --list
- git config user.name

~/.gitconfig


Terminal
File Edit View Search Terminal Help
[user]
name = Jeudy Blanco
email = jeudy.blanco@cinespa.ucr.ac.cr
".gitconfig" 3 lines, 68 characters

Comandos Git

- git clone
- git init
- git add
- git commit (--amend)
- git checkout
- git pull
- git push
- git status
- git diff
- git rm
- git log --stat, -p, --prety=online|full
- git fetch
- git blame
- git remote -v

Iniciando un repositorio

- Directorio existente.
- Clonar repositorio desde servidor.

Nuevo Repositorio Local

- Creamos directorio nuevo en home
- Lo inicializamos con git init
- Creamos un shell script simple.sh

```
#! /bin/bash

echo "El primer parametro recibido es: $1"
```

- Le damos permisos de ejecución.
- .gitignore (ejemplos *~)

Mostrar branch actual en consola

- https://github.com/jimeh/git-aware-prompt/blob/master/README.md
- mkdir ~/.bash
- cd ~/.bash
- git clone git://github.com/mikesten/git-aware-prompt.git
- Editar ~/.profile o ~/.bashrc:
  - export GITAWAREPROMPT=~/.bash/git-aware-prompt
  - source $GITAWAREPROMPT/main.sh
  - export PS1="\u@\h \w\[$txtcyn\]\$git_branch\[$txtylw\]\$git_dirty\[$txtrst\]\$ "

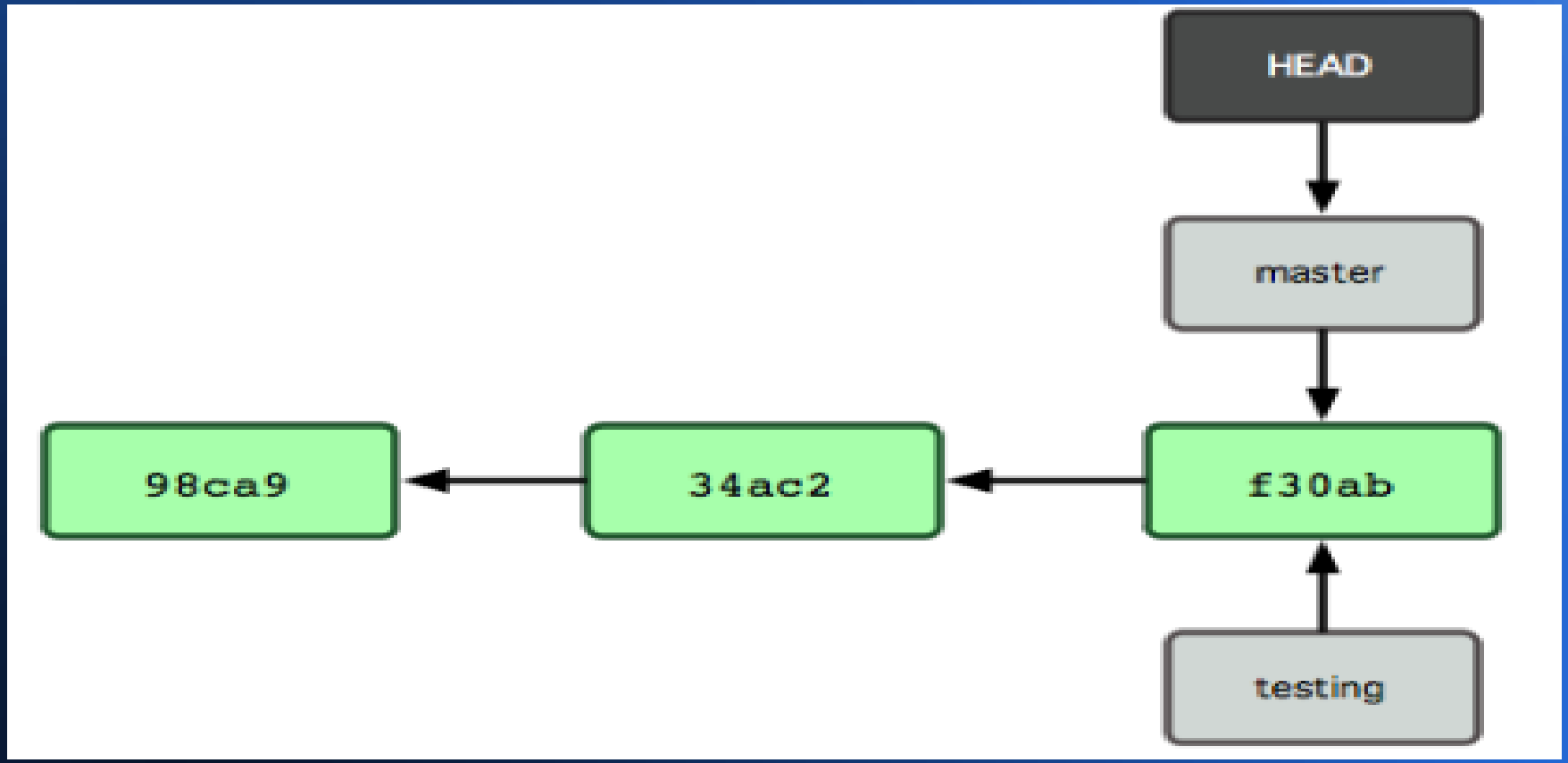
Branches

- Un branch es una “divergencia” respecto al código principal.
- Permite trabajar en cambios o correcciones sin afectar al resto del equipo o al código principal.
- Al terminar, se puede fusionar el branch con el código principal (merge con master).

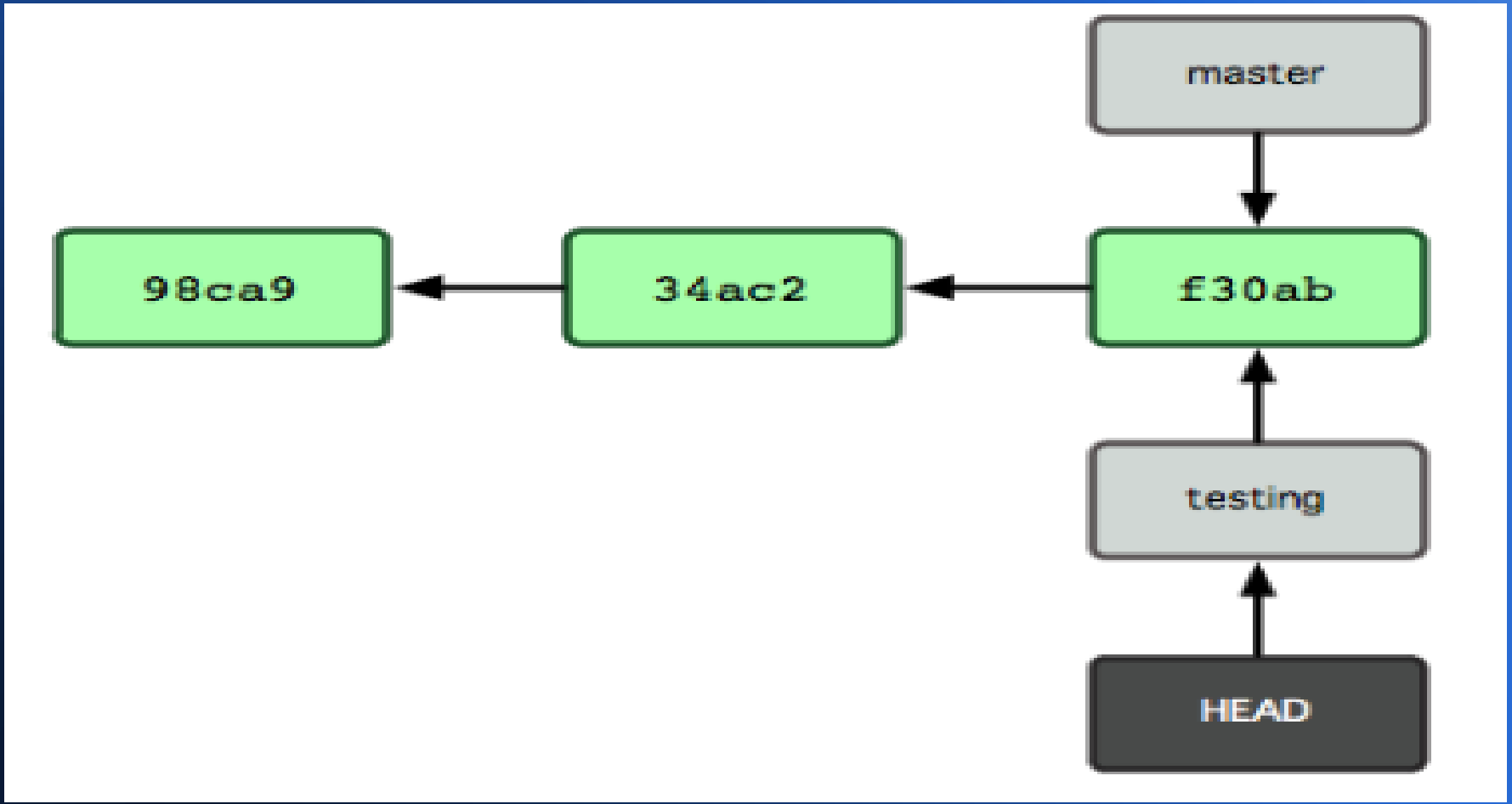
Ejercicio

- Crear nuevo branch
  - git branch <nombre>
  - git checkout <nombre>
- O bien, abreviado:
  - Git checkout -b <nombre>

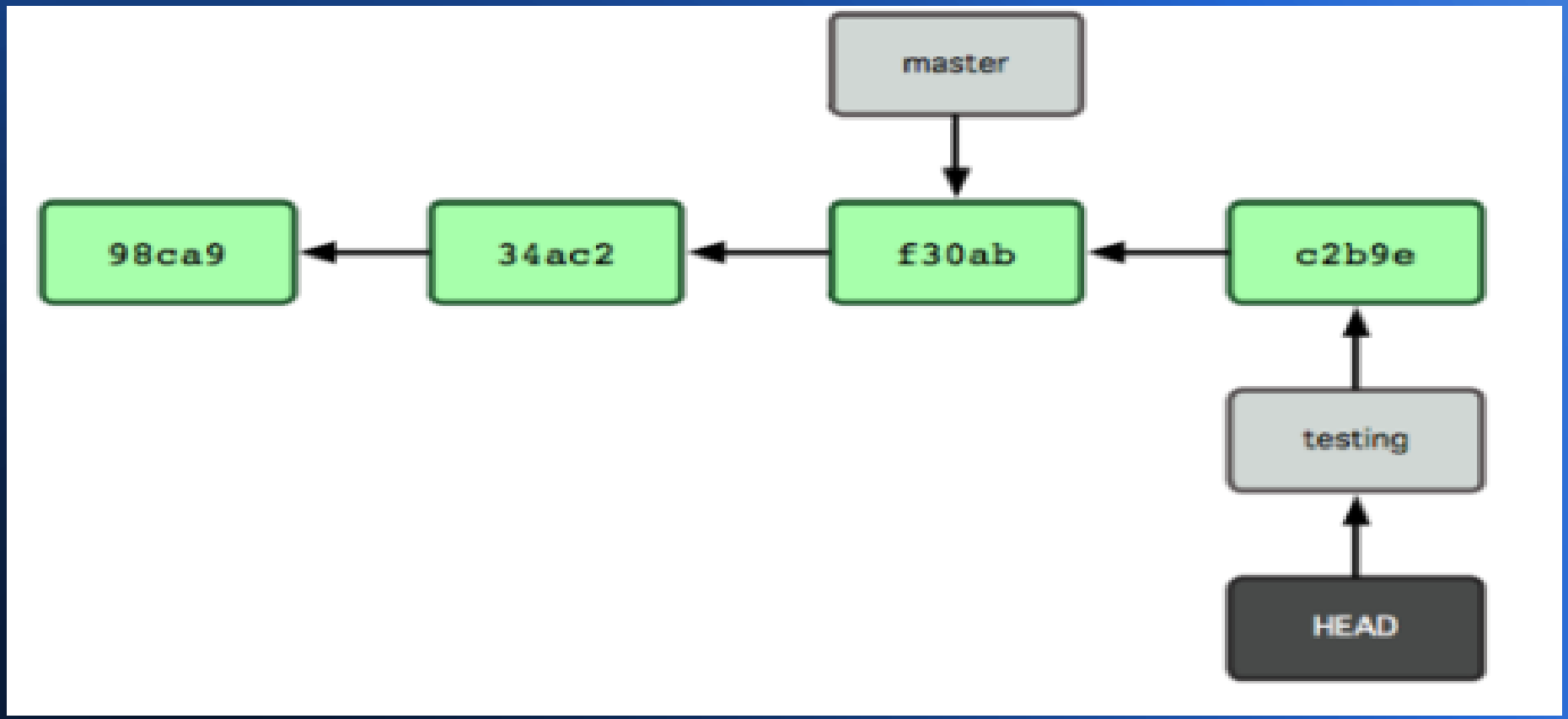
HEAD – Nuevo Branch



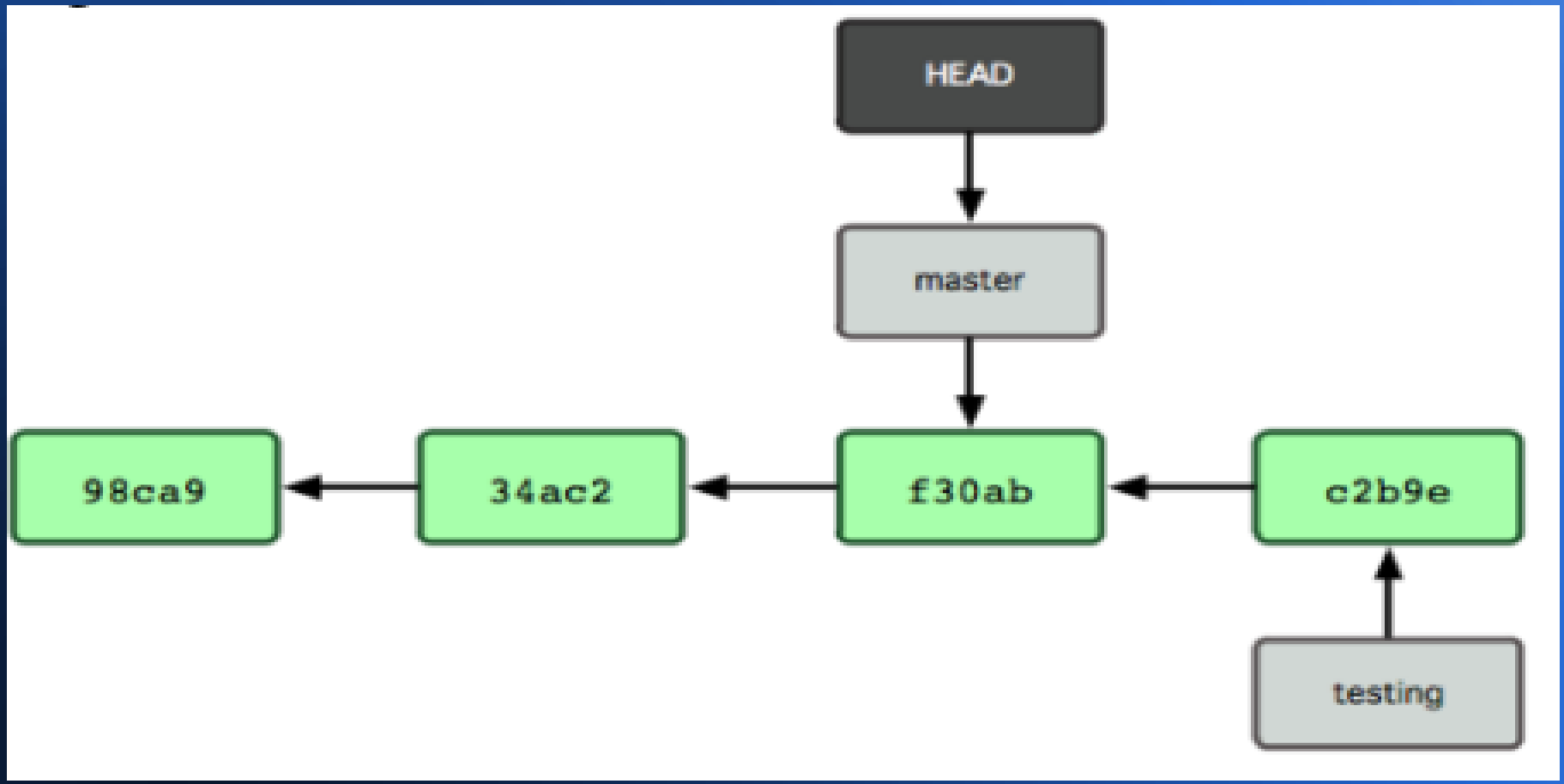
HEAD – checkout nuevo branch



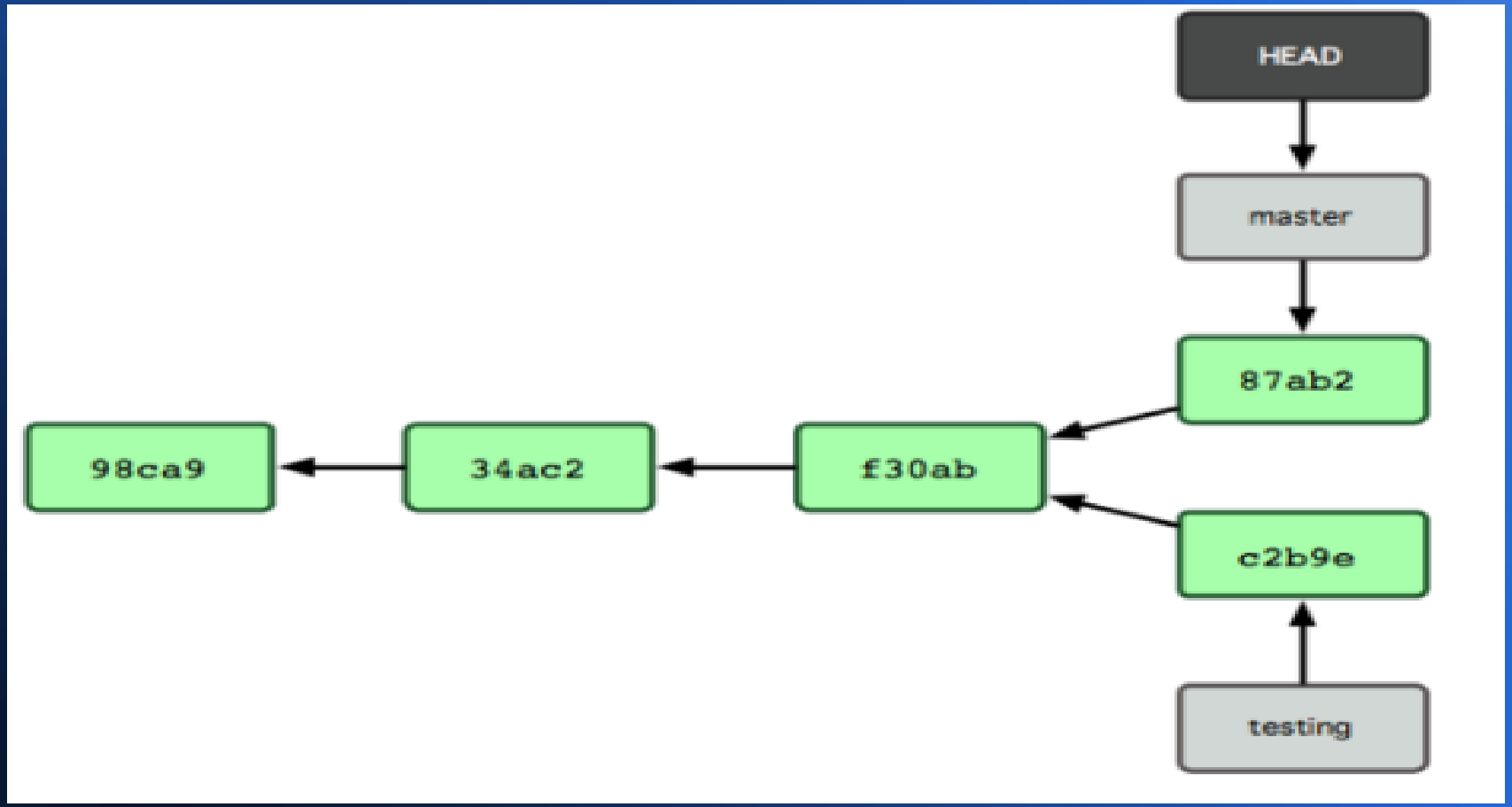
HEAD – nuevo commit



HEAD – checkout de master

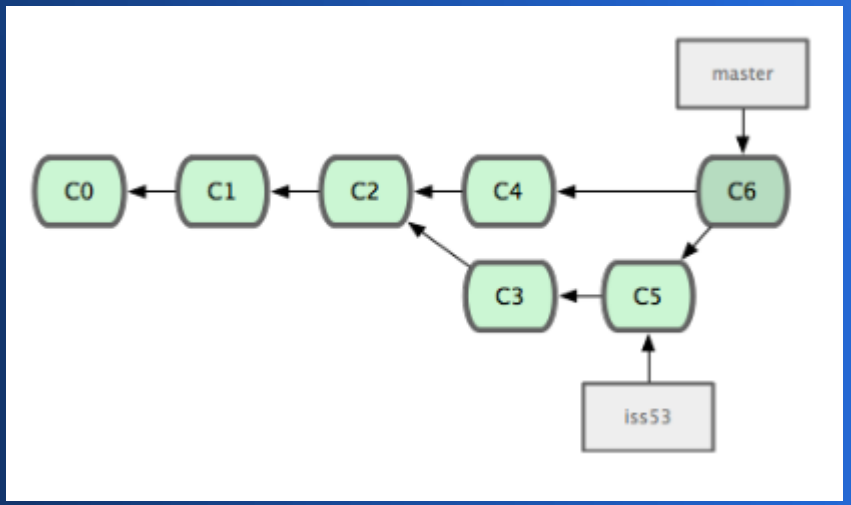


HEAD – nuevo commit en master



Ejercicio

- Del programa simple.sh, hacer checkout a master y ejecutarlo.
- Hacer merge del nuevo branch.
- Volver a ejecutarlo.



Conflictos en merge D:

- Cambios en misma línea en branches diferentes.
- Ejercicio:
  - Checkout de nuevo branch
  - Cambie el segundo echo y agregue linea, commit
  - Checkout a master
  - Cambio segundo echo, commit
  - Merge de nuevo branch.
  - Enjoy.

Resolver conflictos

```
1 #! /bin/bash
2 echo "El primer parametro es: $1"
3 <<<<<<< HEAD
4 echo "Fin del programa desde master"
5 =======
6 echo "La fecha actual es `date`."
7 echo "Ultima linea"
8 >>>>>>> trouble
9
```

- Se resuelve el conflicto.
- git add de archivo.
- git commit

Resolver conflictos

```
1 #! /bin/bash
2 echo "El primer parametro es: $1"
3 <<<<<<< HEAD
4 echo "Fin del programa desde master"
5 =======
6 echo "La fecha actual es `date`."
7 echo "Ultima linea"
8 >>>>>>> trouble
9
```

- Se resuelve el conflicto.
- git add de archivo.
- git commit

Regresar a commit anterior

- git reset --hard <SHA-1>
- git reset --hard HEAD~10

Próxima clase

- Git en la nube: GitHub y BitBucket
- Ciclo de vida del software
- Estructuras de datos
- Quiz :-)