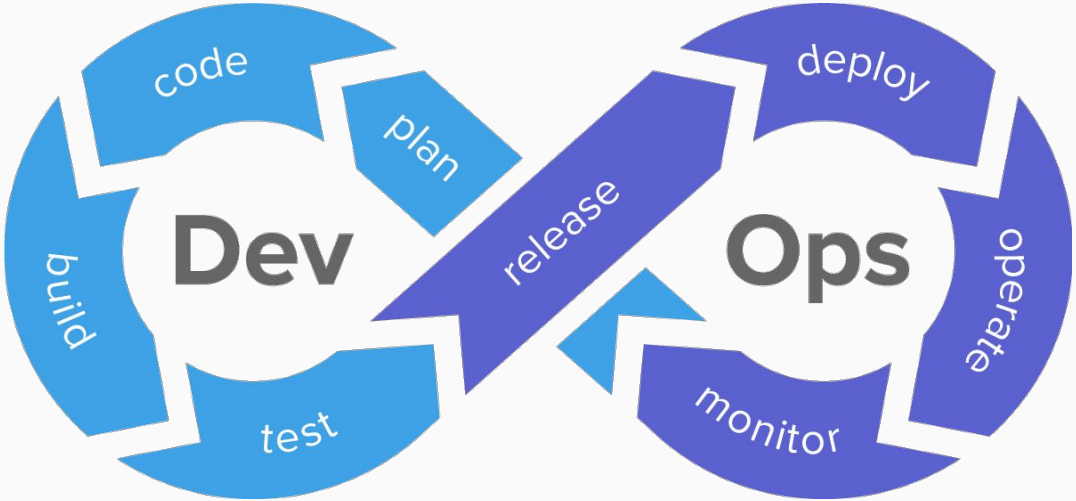
Fundamentos Desarrollo de Experiencias Multimedia para la Web

Juan José Cardona Quiroz

Devops


code
plan
build
test
Dev
release
deploy
operate
monitor
Ops

Pequeñas conquistas

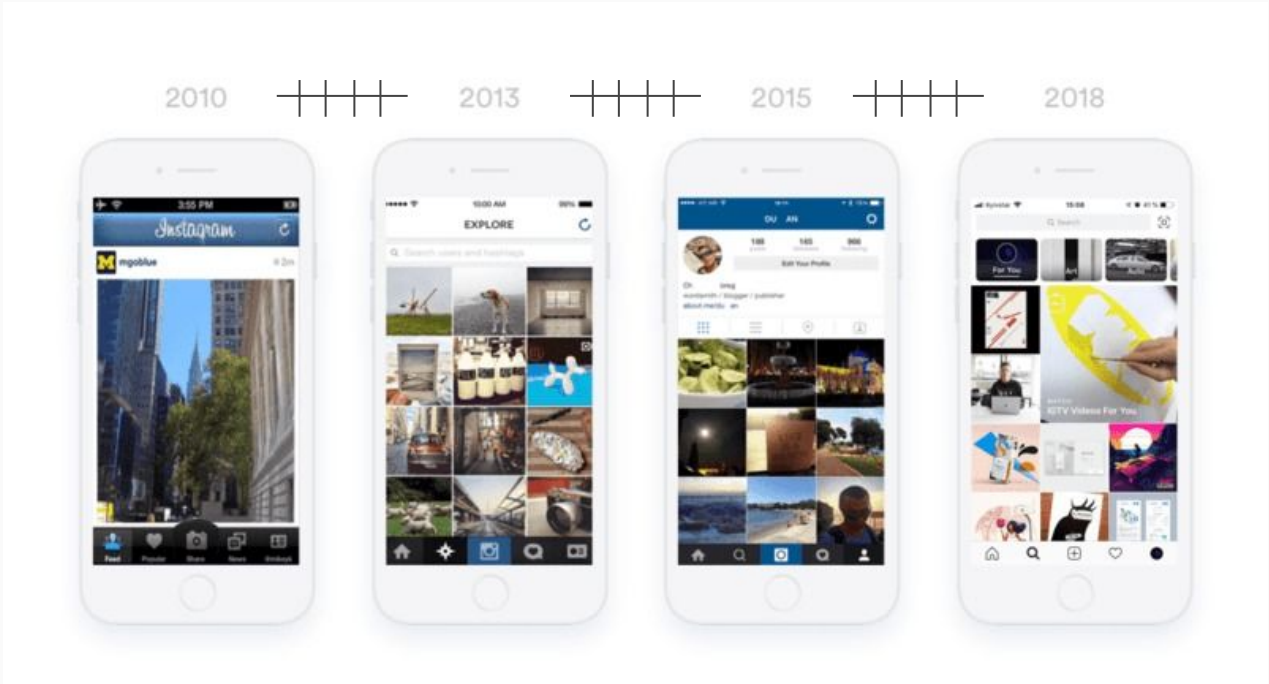
La implementación de pequeños lotes de funcionalidades llevará al proyecto a un proceso de mejora continua.

+150

Archivos de código asociados al proyecto

¿Cómo podemos implementar un proceso de mejora continua?

Manejo de versiones

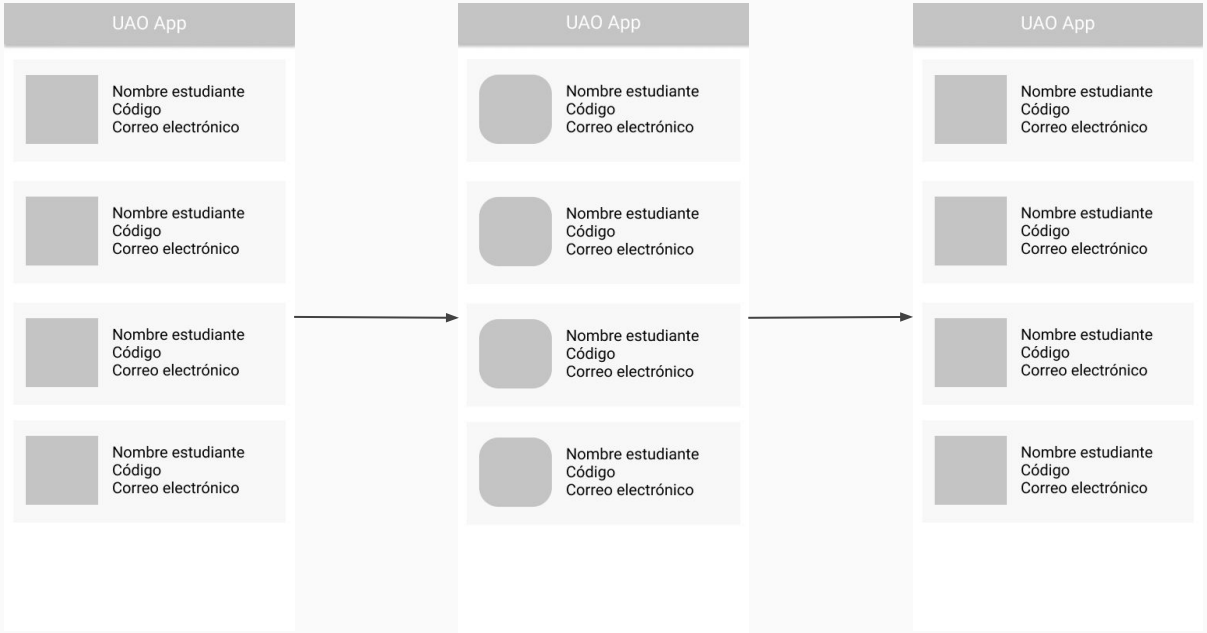


Versión A

Todos los archivos del proyecto con la imágen cuadrada

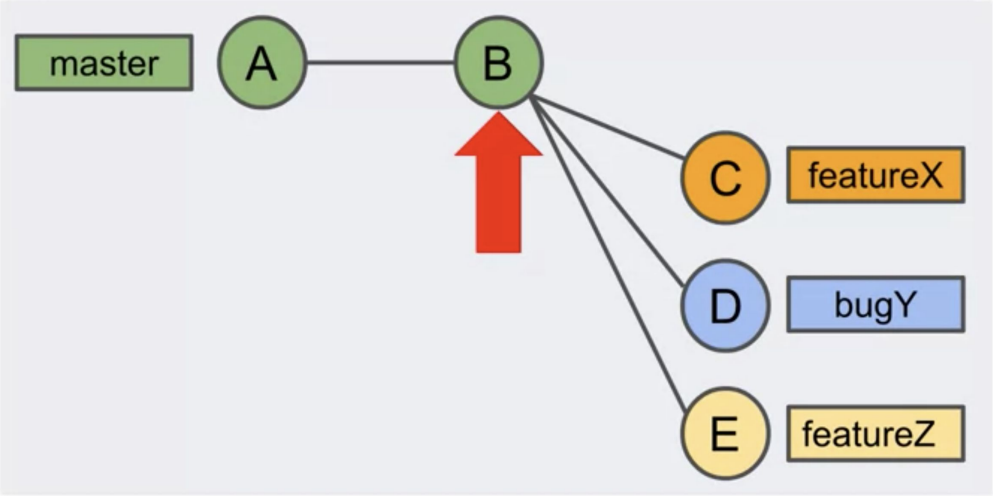
Versión B

Todos los archivos del proyecto con la imágen redondeada

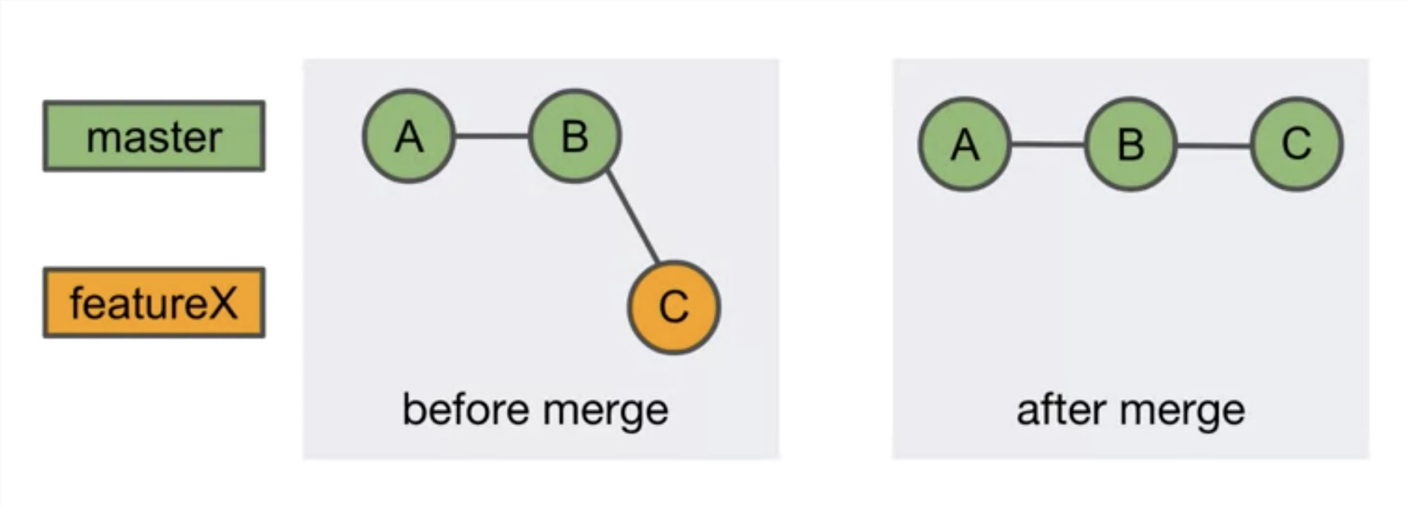


Versión A **Versión B** **Versión C**

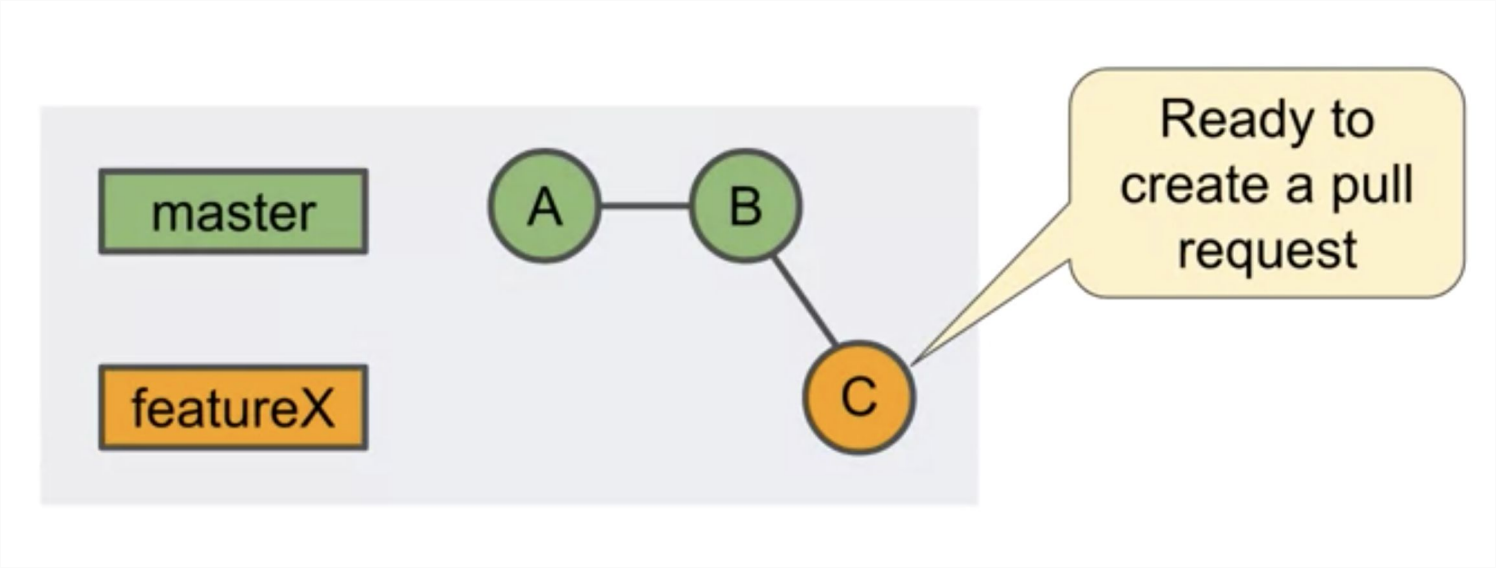
Manejo de ramas



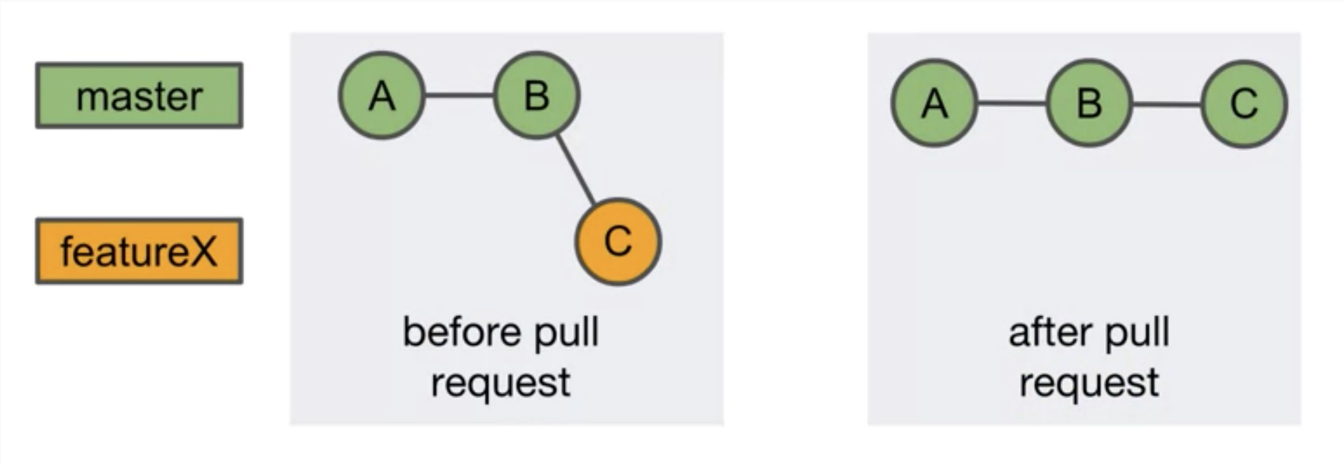
Manejo de ramas



Manejo de ramas



Manejo de ramas

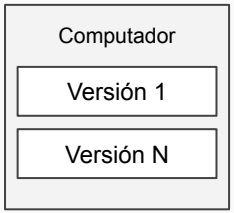


¿Qué tipo de contenido podríamos tener en un Sistema de control de versiones?

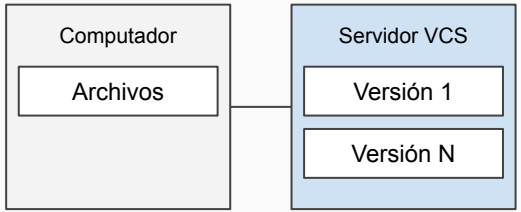
- Código fuente
- Pruebas automáticas
- Configuración de servidor
- Documentación
- Libro
- Contenido de un sitio web

Tipos de sistemas de control de versiones

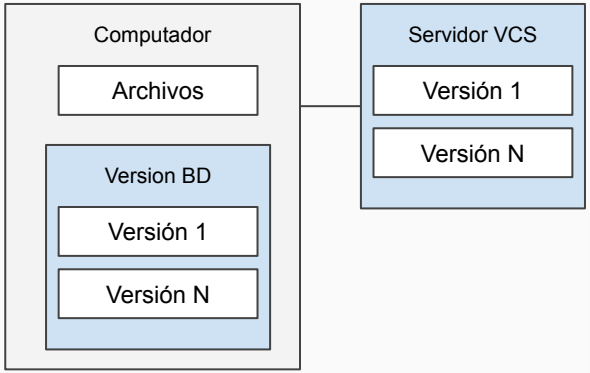
- Locales



- Centralizados



- Distribuidos



Git es un sistema de control de versiones

Es un software libre de uso (Open Source Software OSS) con una comunidad bastante robusta

Git

Línea de comandos

- Son habilidades valoradas en la industria
- Automatizable
- Rápido

```
git --version
```

Instalación y configuración

Validación de instalación

git --version

Para instalar

https://www.atlassian.com/git/tutorials/install-git

Sintáxis de Git

Estructura básica de un comando en Git

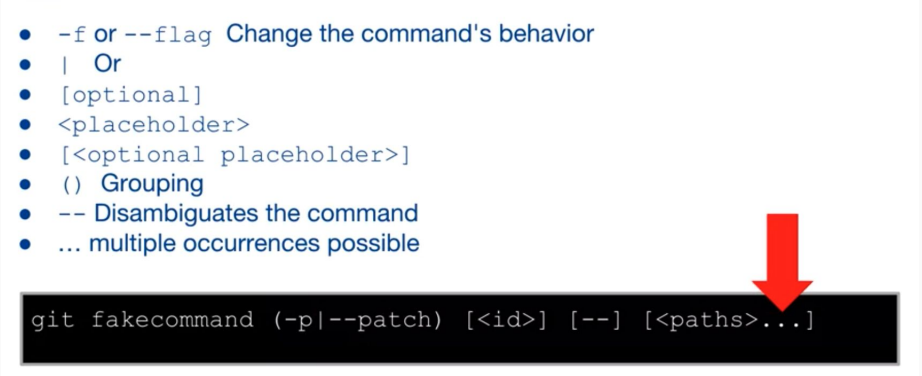
git [command] [--flags] [arguments]

Ejemplo:

git status

git status --short

git add file.txt



Configurar nombre del usuario y correo

`git config [--local|--global|--system] <key> [<value>]`

- The `--system` flag applies to every repository for all users on your computer
- The `--global` flag applies to every repository that you use on your computer
- No flag or `--local` applies only to the current repository (highest precedence)

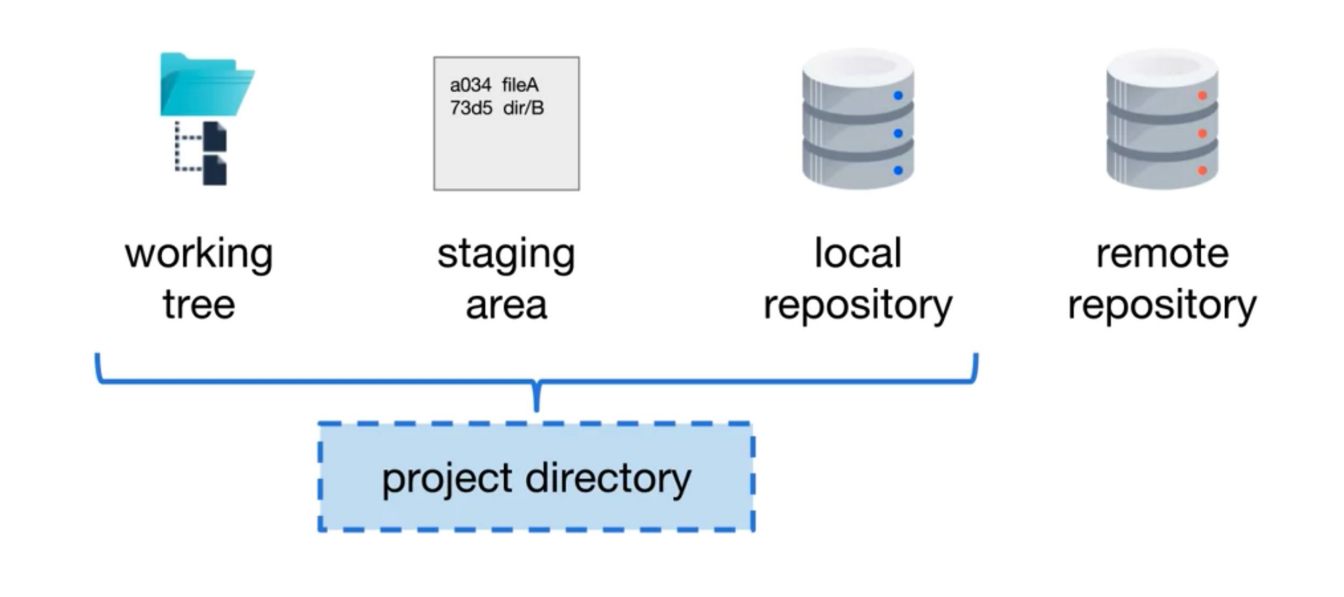
```
# set user name and email
$ git config --global user.name "Pat"
$ git config --global user.email "pat@example.com"
```

Configurar nombre del usuario y correo

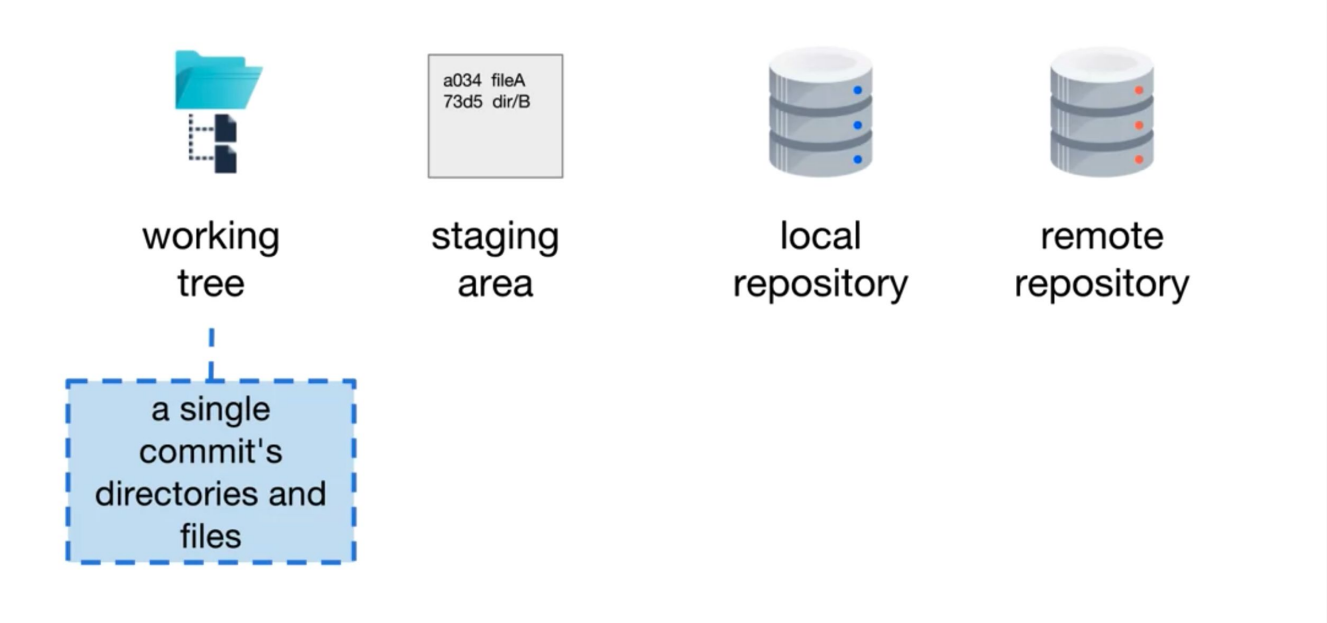
`git config <key>`

- The current value of `<key>` will be returned

Git Locations

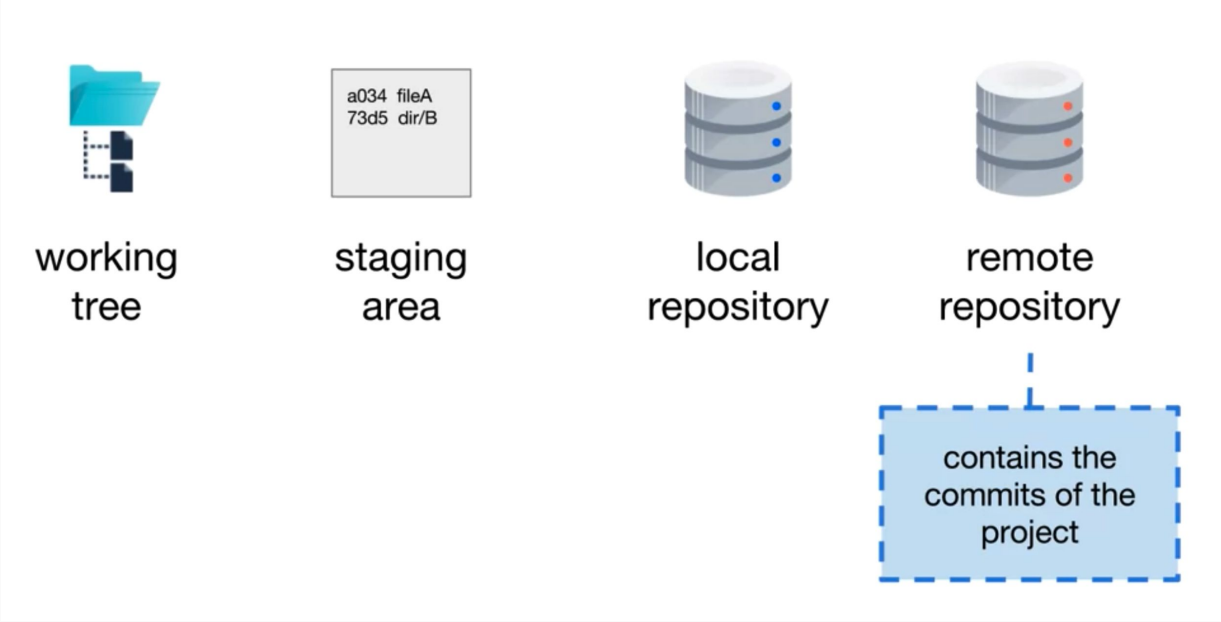


Git Locations



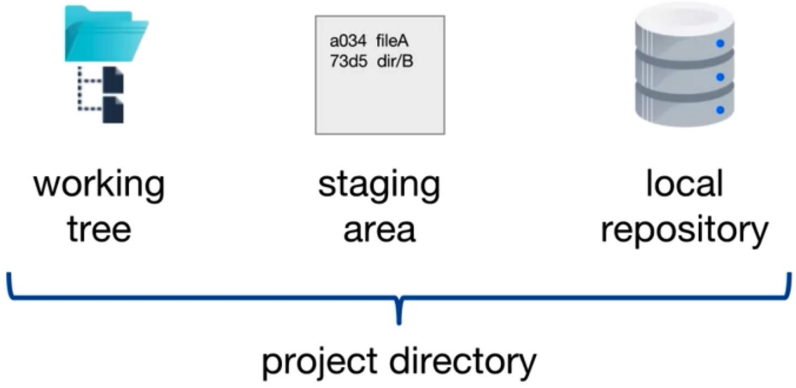
Git Locations

Git Locations



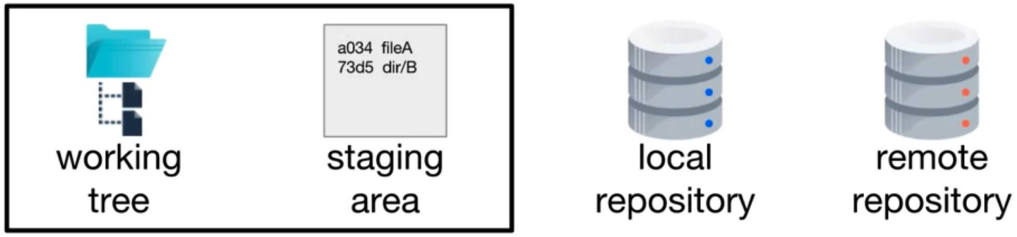
Creación de un repositorio Local

```
myproj $ git init
Initialized empty Git repository in myproj/.git/
```



Git Status

Use `git status` to view the status of files in the working tree and staging area




```
myproj$ git status
On branch master
nothing to commit, working tree clean
```

Git Add

```
git add <file-or-directory>
```


```
myproj$ git add fileA.txt
On branch master
No commits yet
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

    new file:   fileA.txt
```

Git Add Directories

Add directories with `git add <directory>`

```
(create dirA and dirA/fileA.txt, dirA/fileB.txt)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    dirA/
nothing added to commit but untracked files present (use "git add" to
track)
$ git add dirA
$ git status
On branch master
No commits yet
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   dirA/fileA.txt
    new file:   dirA/fileB.txt
```

Git Modified File

Add directories with `git add <directory>`

```
(create dirA and dirA/fileA.txt, dirA/fileB.txt)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    dirA/
nothing added to commit but untracked files present (use "git add" to
track)
$ git add dirA
$ git status
On branch master
No commits yet
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   dirA/fileA.txt
    new file:   dirA/fileB.txt
```

Git Commit

Adds staged content to the local repository as a commit

- Previously committed files are also included
- Creates a snapshot of the entire project

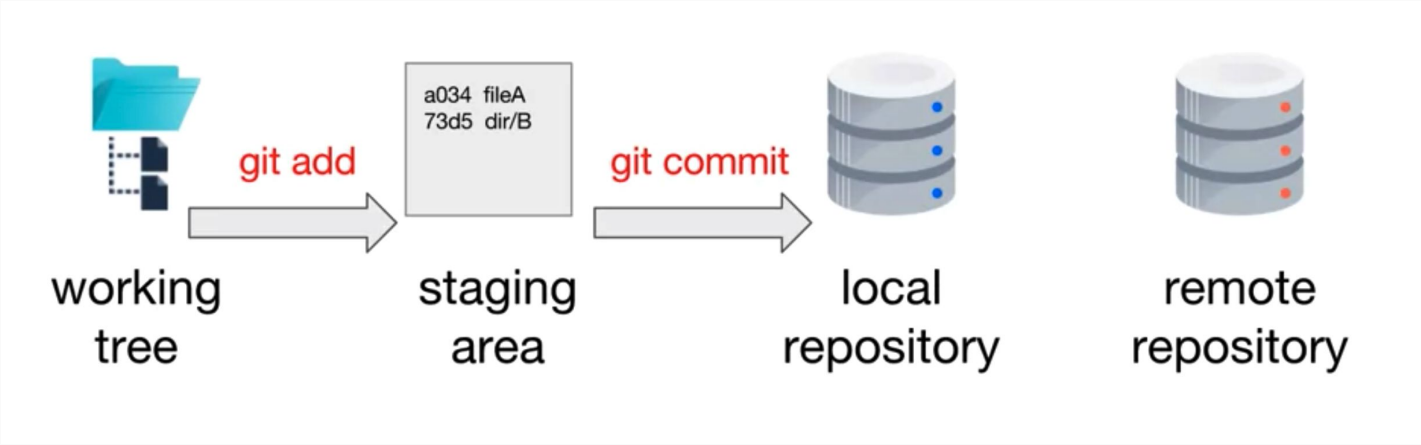
Git Commit

```
$ git commit -m "initial commit"
[master (root-commit) 725c95a] initial commit
 1 file changed, 1 insertion(+)
 create mode 100644 fileA
$ git status
On branch master
nothing to commit, working tree clean
```

Git Log

```
$ git log
commit 725c95ae156e5f10d4f99735b2fa8698a9860128 (HEAD -> master)
Author: Pat Smith <pat@example.com>
Date:   Sat Sep 9 14:15:04 2017 -0700

    initial commit
```

Proceso dentro del repositorio local



Repositorio remoto

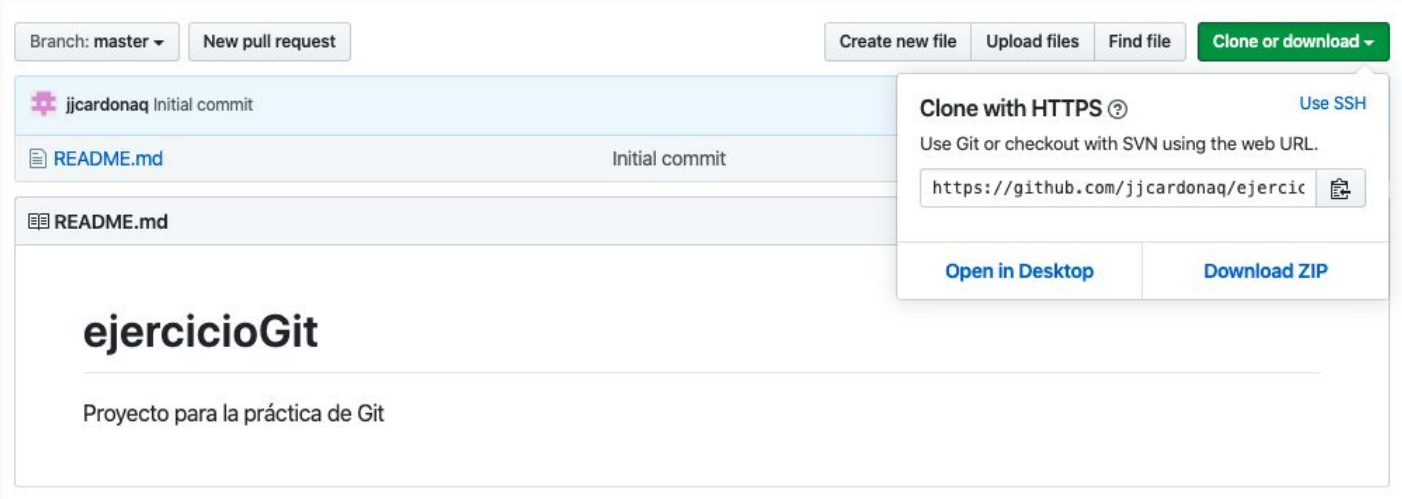
- Gestión profesional
- Fuente oficial del proyecto
- Interoperable con otros sistemas

Posibles escenarios

Have a local repository?	Task
no	*clone* the remote
yes	*add* the remote

Si **NO** tengo un repositorio local

Clonar repositorio remoto



Información del repositorio remoto

`git remote --verbose`

- Displays information about remote repositories associated with the local repository

```
repos$ git clone https://bitbucket.org/atlassian_tutorial/helloworld.git
Cloning into 'helloworld'...
$ cd helloworld
helloworld$ git remote -v
origin   https://bitbucket.org/atlassian_tutorial/helloworld.git (fetch)
origin   https://bitbucket.org/atlassian_tutorial/helloworld.git (push)
```

TENIENDO un repositorio local

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

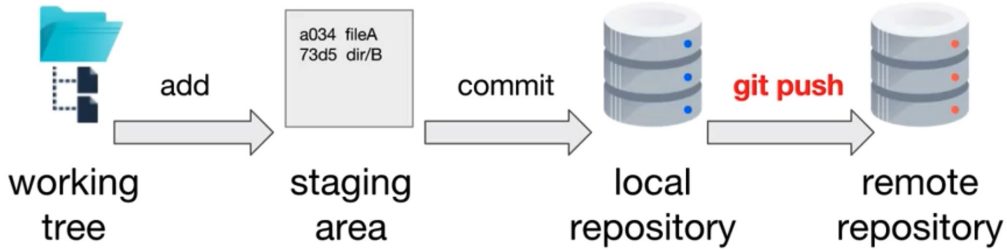
```
(create a remote Bitbucket repository named repoa)
repoa$ git remote add origin https://me@bitbucket.org/me/repoa.git
repoa$ git remote -v
origin   https://bitbucket.org/me/repoa.git (fetch)
origin   https://bitbucket.org/me/repoa.git (push)
```

Creación de versiones

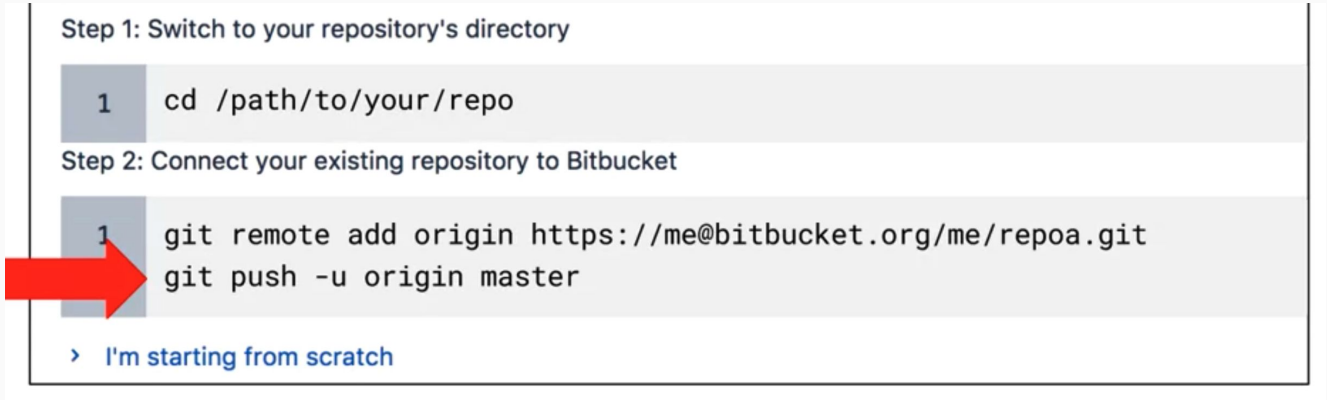
- All commits belong to a *branch*
- By default, there is a single branch and it is called *master*

Creación de versiones

`git push` writes commits for a branch to a remote repository



Creación de versiones



Pushing commits al repositorio remoto

`git push [-u] [<repository>] [<branch>]`

- `<repository>` can be a name (shortcut) or URL
- `-u` track this branch (`--set-upstream`)

```
repoa$ git remote -v
origin    https://bitbucket.org/me/repoa.git (fetch)
origin    https://bitbucket.org/me/repoa.git (push)
repoa$ git push -u origin master
```

Resumen

Have a local repository?	Command
no	`git clone`
yes	`git remote add`

You can then push commits to the remote repository

Git Objects

1. **Commit**- A small text file
2. **Annotated tags**- A permanent reference to a commit
3. Tree- Directories and filenames in the project
4. Blob- The content of a file in the project

Git ID

- The **name** of a Git object
- 40-character hexadecimal string
- Also known as *object ID, SHA-1, hash* and *checksum*

```
$ git log
commit e8d41c0322e629aaf7d7aa8a9a1335bff2f76f72 (HEAD -> master)
...
```

Git Log con Shortname

Four or more characters of the beginning of a Git ID

```
$ git log --oneline
483d057 (HEAD -> master) add README.md
$ git log
commit 483d0572148a7bfed924a1f7bee20de6d9364942 (HEAD -> master)
Author: ...
$ git show 483d
commit 483d0572148a7bfed924a1f7bee20de6d9364942 (HEAD -> master)
Author: ...
```

Referencias

User-friendly name that points to:

- a commit SHA-1 hash
- another reference
  - known as a *symbolic reference*

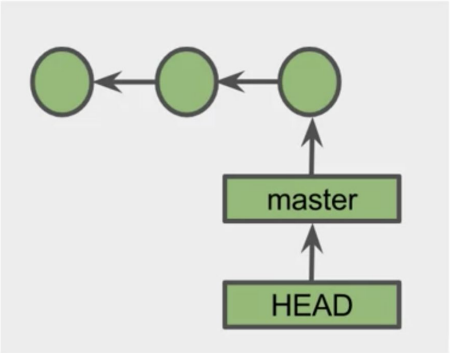
Referencias

Use references instead of SHA-1 hashes

```
$ git log
commit 1ef16ac15dbd60a779a6c8b5f3cc1ce39383fa2c (HEAD -> master)
...
$ git show HEAD
commit 1ef16ac15dbd60a779a6c8b5f3cc1ce39383fa2c (HEAD -> master)
Author: Pat <pat@example.com>
Date:   Tue Sep 19 16:50:07 20xx -0700
    changed fileA.txt to version 2
diff --git a/fileA.txt b/fileA.txt
...
```

¿Qué es HEAD?

- A reference to the current commit
- Usually points to the branch label of the current branch
- One *HEAD* per repository




```
$ git log --oneline -1
483d057 (HEAD -> master) add feature 2
```

“Parentezco” entre versiones

Refers to a prior commit

- ~ or ~1 = parent
- ~2 or ~~ = parent's parent

```
$ git log --oneline --graph
* e0cb6c5 (HEAD -> master, tag: v2.0) added feature 2 to fileA.txt
* 1ef16ac changed fileA.txt to version 2
* e8d41c0 updated fileA.txt feature 1
* 14b97e6 added fileA.txt
$ git show HEAD
(shows commit info for e0cb6c5)
$ git show HEAD~  # same as HEAD~1
(shows commit info for 1ef16ac)
$ git show master~3
(shows commit info for 14b97e6)
```

Creación de etiquetas

To tag a commit with a lightweight tag:

- `git tag <tagname> [<commit>]`
- `<commit>` defaults to `HEAD`

```
$ git tag v1.0 # tag the current commit
$ git tag # view tags
v1.0
$ git tag v0.1 HEAD^ # tag the previous commit
$ git tag
v0.1
v1.0
```

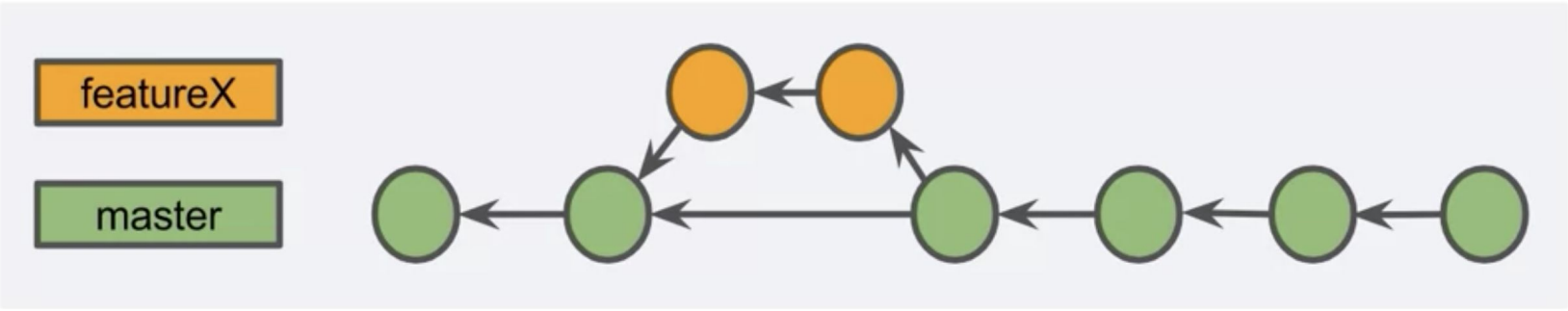
Creación de etiquetas

- To tag a commit with an annotated tag:
  - `git tag -a [-m <msg> | -F <file>] <tagname> [<commit>]`
  - `<commit>` defaults to `HEAD`
- `git show` displays the tag object information followed by the commit information

```
$ git tag -a -m "includes feature 2" v2.0
$ git show v2.0
tag v2.0
Tagger: Pat <pat@example.com>
Date:   Thu Sep 21 18:39:58 20XX-0700
includes feature 2

commit e0cb6c5eb3ebaceb30a860c713eaeae3901fa79e (HEAD -> master, tag: v2.0)
Author: …
(commit information)
```

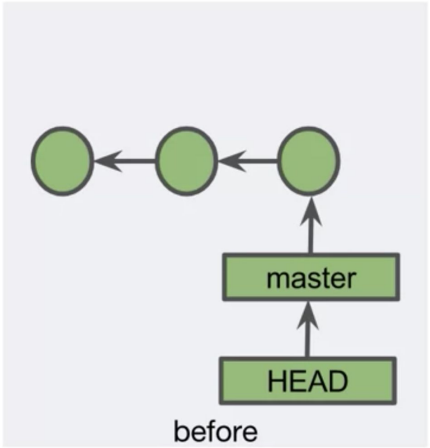
Ramas en Git

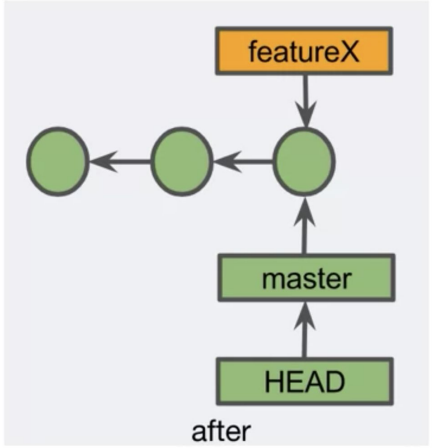
- Fáciles de crear
- Permiten experimentar
- Permiten el trabajo en equipo
- Permite el soporte de múltiples versiones



Ramas en un repositorio

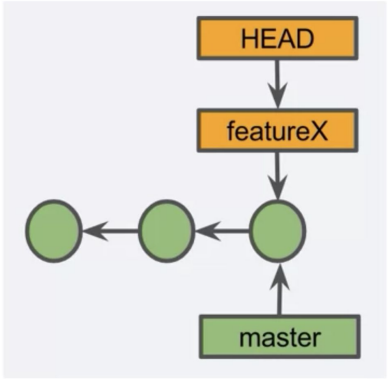
Crear una rama

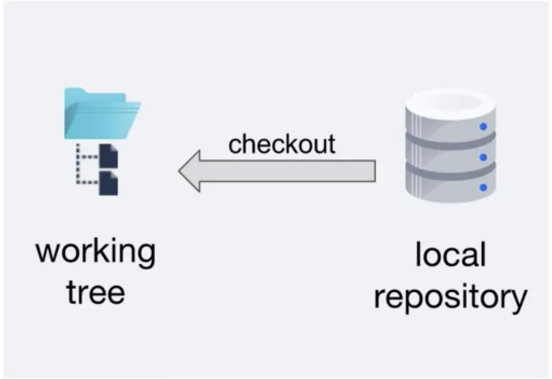





```
# create a branch
$ git branch featureX
$ git branch
* master
  featureX
```

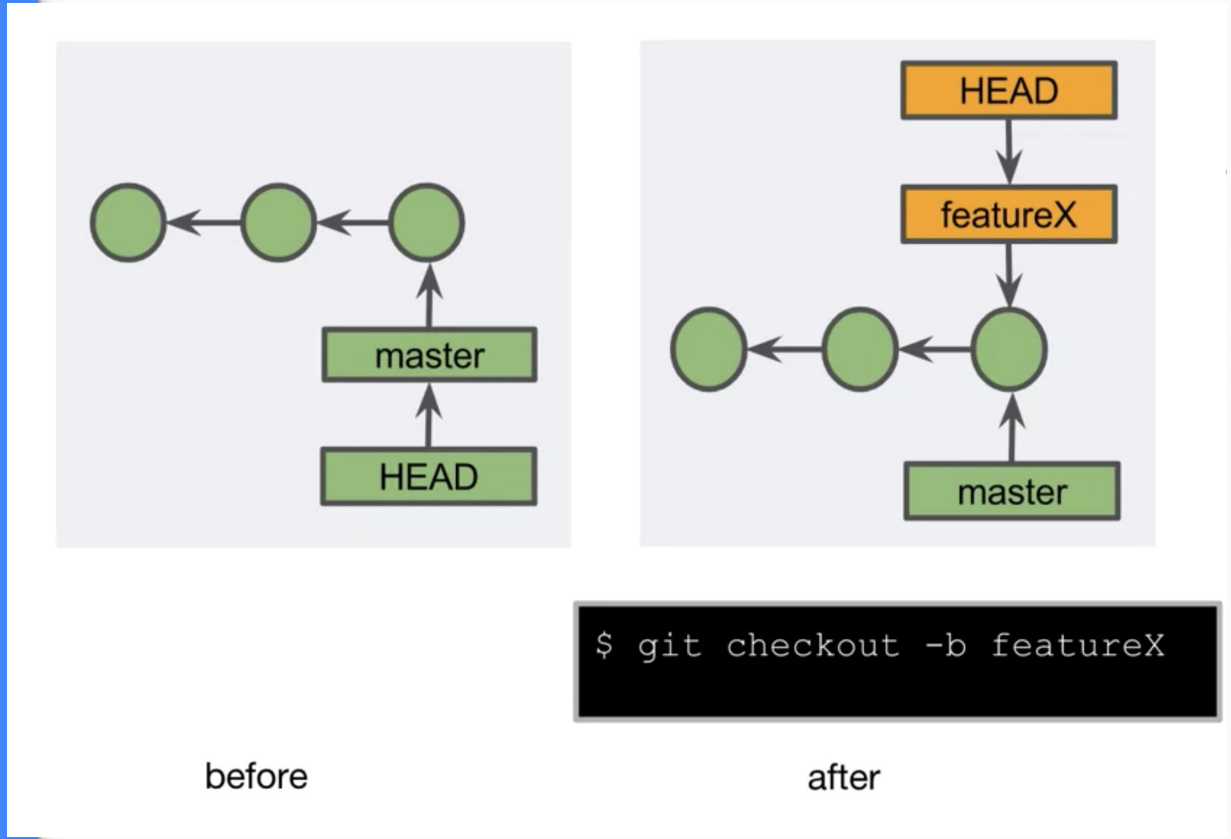
Cambiar HEAD



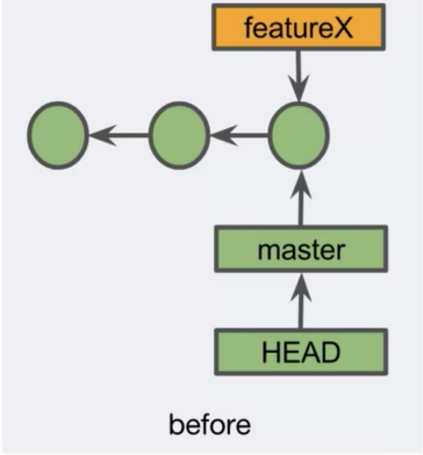


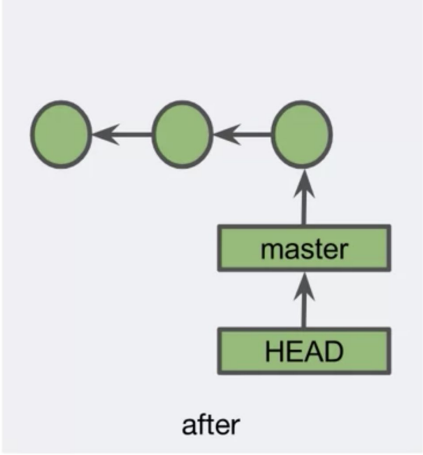

```
# switch to a branch
$ git checkout featureX
$ git branch
  master
* featureX
```

Atajos cuando se crea un branch



Borrar Branch



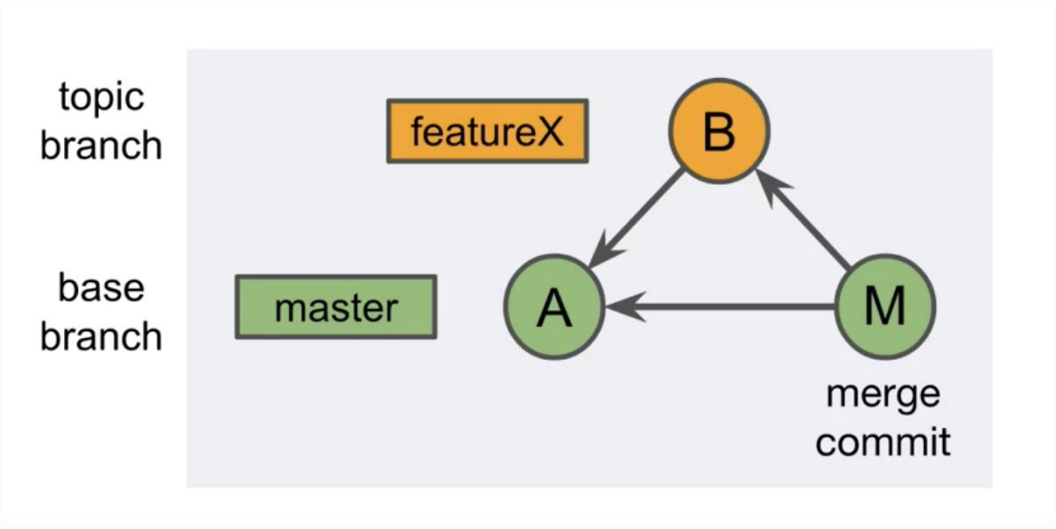



```
$ git branch -d featureX
Deleted branch featureX (was 010226b)
$ git branch
* master
```

Devolverse a una versión que se está borrando

```
$ git branch -D featureX
Deleted branch featureX (was 434dfa0)

$ git reflog
942c36f (HEAD -> master) HEAD@{0}: checkout: moving from featureX to master
434dfa0 HEAD@{1}: commit: added featureX
942c36f (HEAD -> master) HEAD@{2}: checkout: moving from master to featureX
…

$ git checkout -b featureX 434dfa0
Switched to a new branch 'featureX'
```

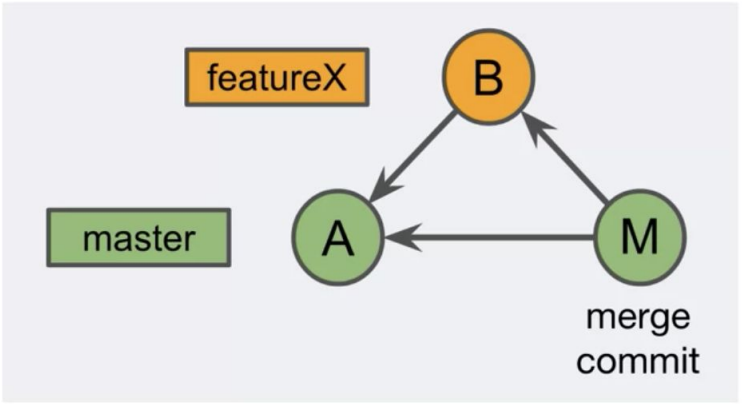
Merge



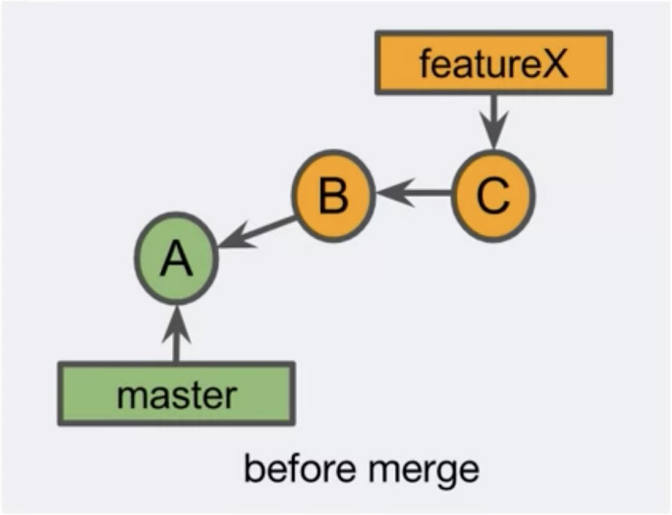
Formas de hacer Merge

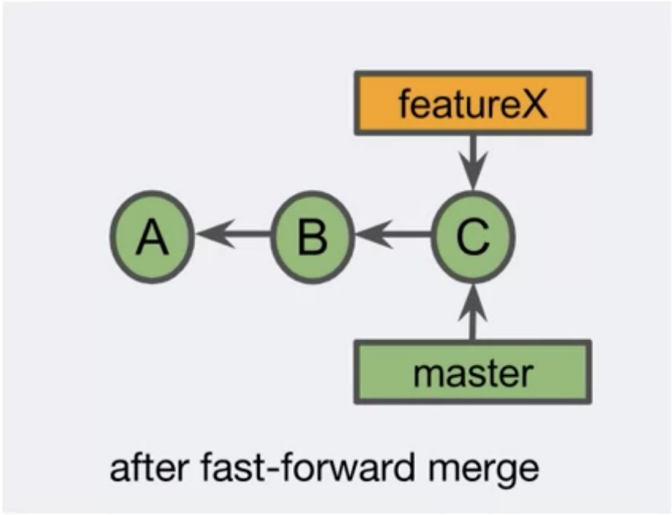
Main types of merges:

1. Fast-forward merge
2. Merge commit
3. Squash merge*
4. Rebase*

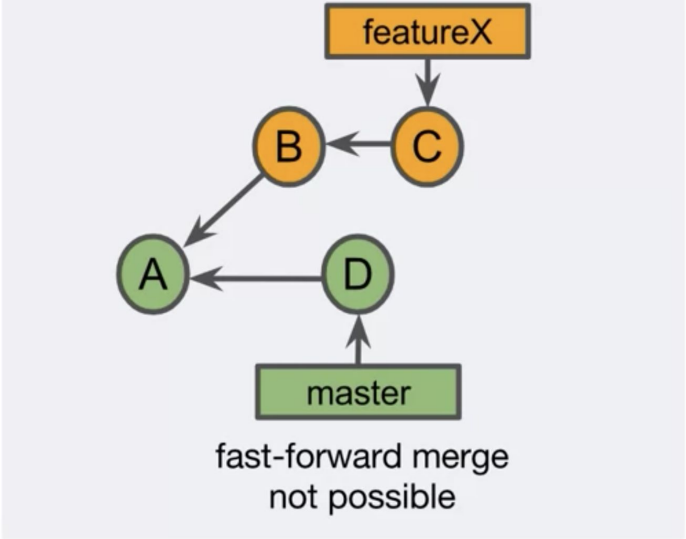


1. Fast forward Merge





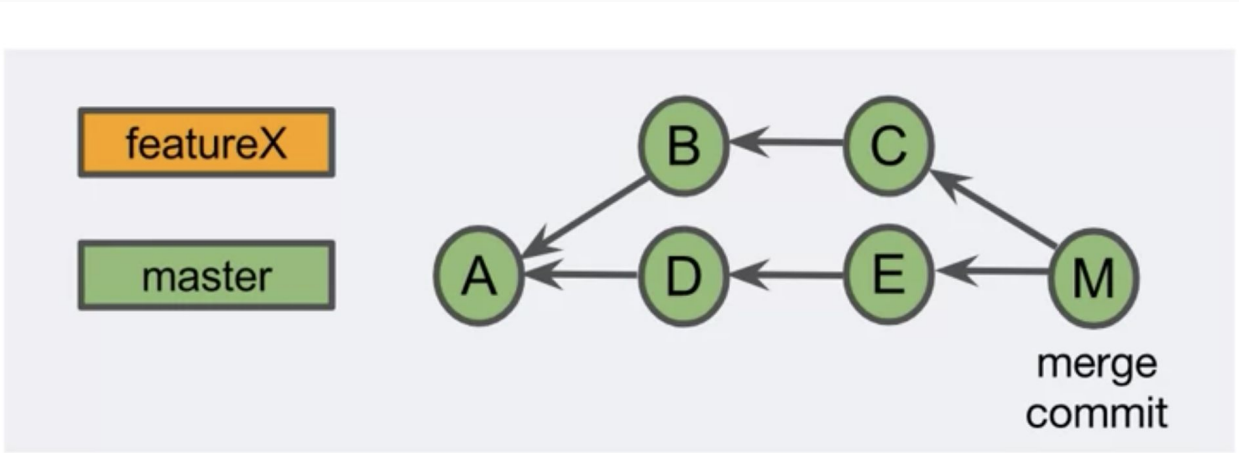
1. Fast forward Merge



1. Fast forward Merge

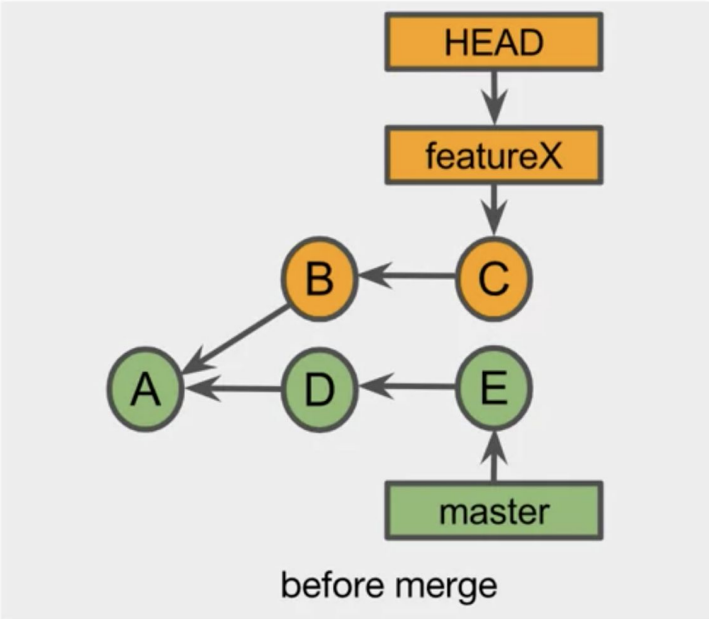
```
$ git log --oneline --graph --all
* 5046c75 (HEAD -> feature2) added feature 2
* fbadd75 (master) added feature 1
$ git checkout master
Switched to branch 'master'
$ git merge feature2
Updating fbadd75..5046c75
Fast-forward
 fileA.txt | 1 +
 1 file changed, 1 insertion(+)
$ git log --oneline --graph --all
* 5046c75 (HEAD -> master, feature2) added feature 2
* fbadd75 added feature 1
$ git branch -d feature2
Deleted branch feature2 (was 5046c75).
$ git log --oneline --graph --all
* 5046c75 (HEAD -> master) added feature 2
* fbadd75 added feature 1
```

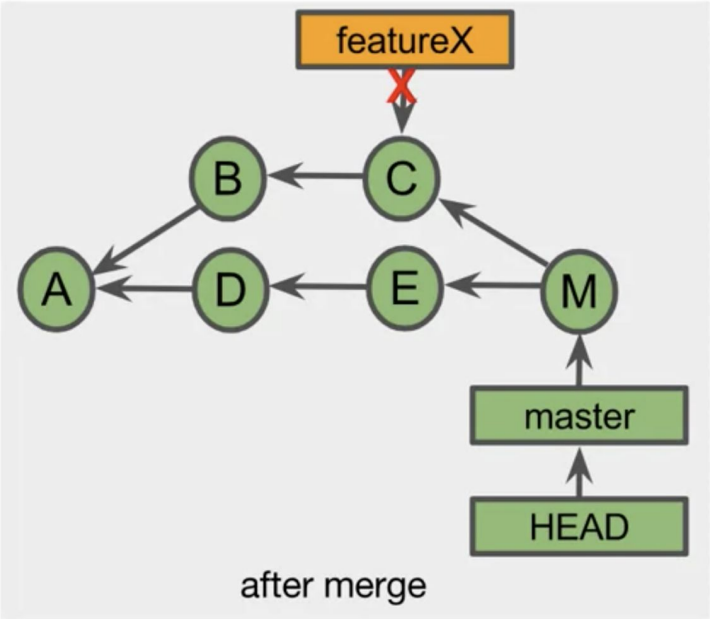
2. Merge commit



2. Merge commit

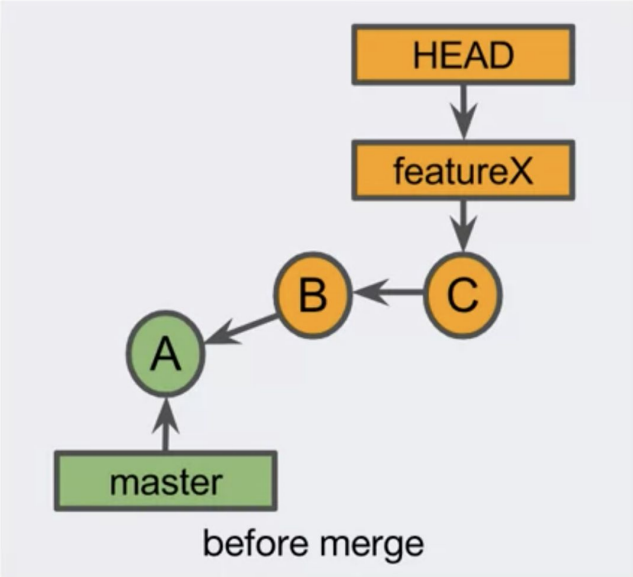
1. `git checkout master`
2. `git merge featureX`
   - accept or modify the **merge message**
3. `git branch -d featureX`



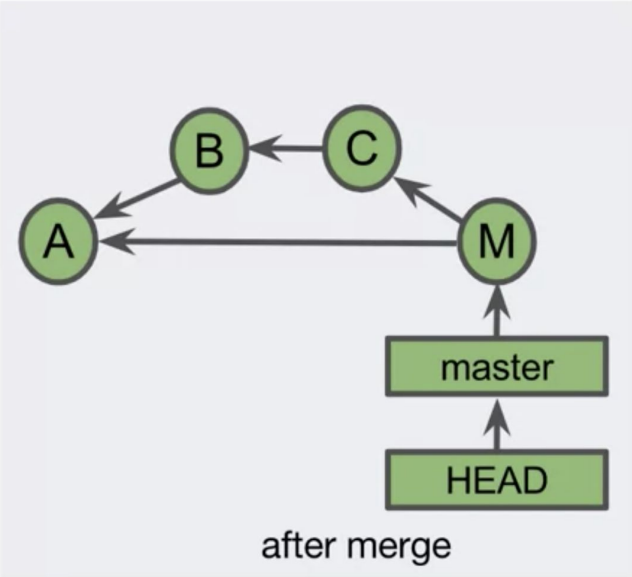


2. Merge commit

1. `git checkout master`
2. `git merge --no-ff featureX`
   - accept or modify the merge message
3. `git branch -d featureX`



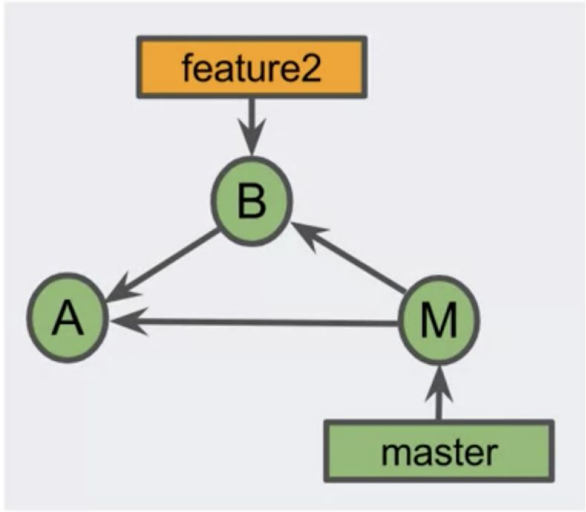
before merge



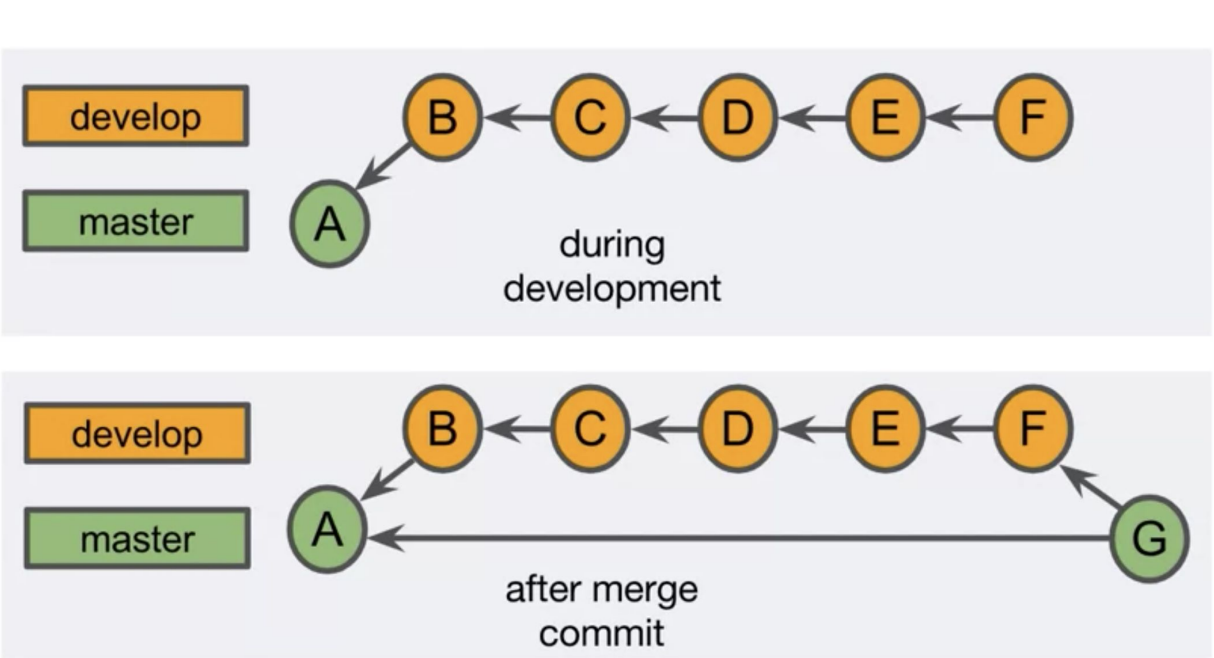
after merge

2. Merge commit

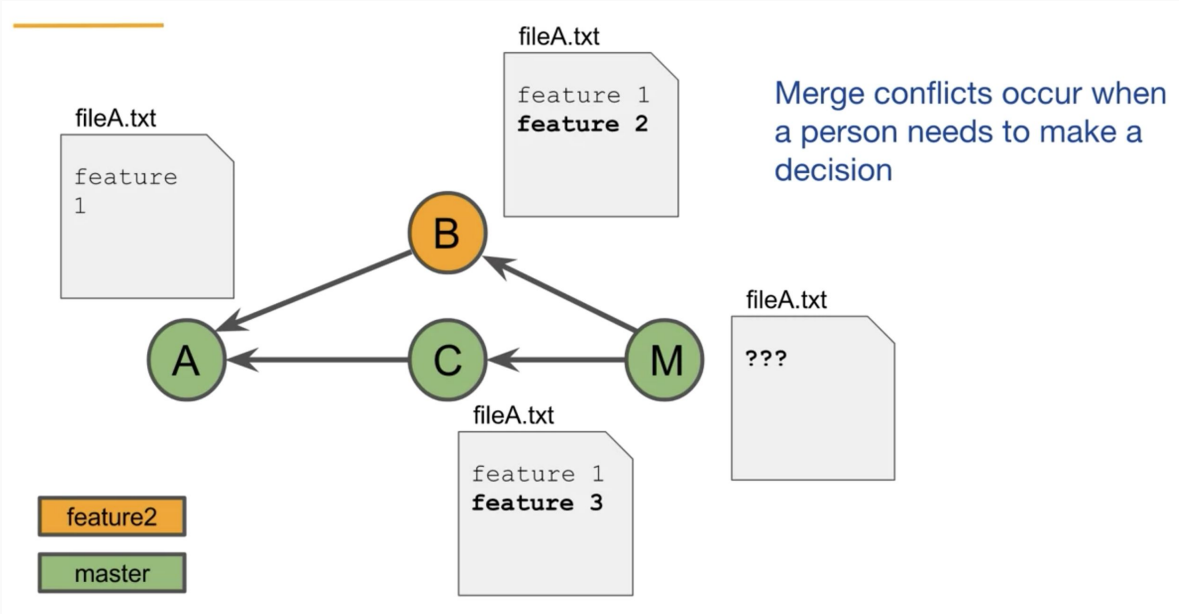
```
$ git log --oneline --graph --all
* 804772b (HEAD -> feature2) added feature 2
* 9232a1d (master) added feature 1
$ git checkout master
Switched to branch 'master'
$ git merge --no-ff feature2
# opens editor to edit merge message
Merge made by the 'recursive' strategy.
 fileA.txt | 1 +
 1 file changed, 1 insertion(+)
$ git log --oneline --graph --all
*   efaa6ff (HEAD -> master) Merge branch 'feature2'
|\
| * 804772b (feature2) added feature 2
|/
* 9232a1d added feature 1
$ git branch -d feature2
Deleted branch feature2 (was 804772b).
```



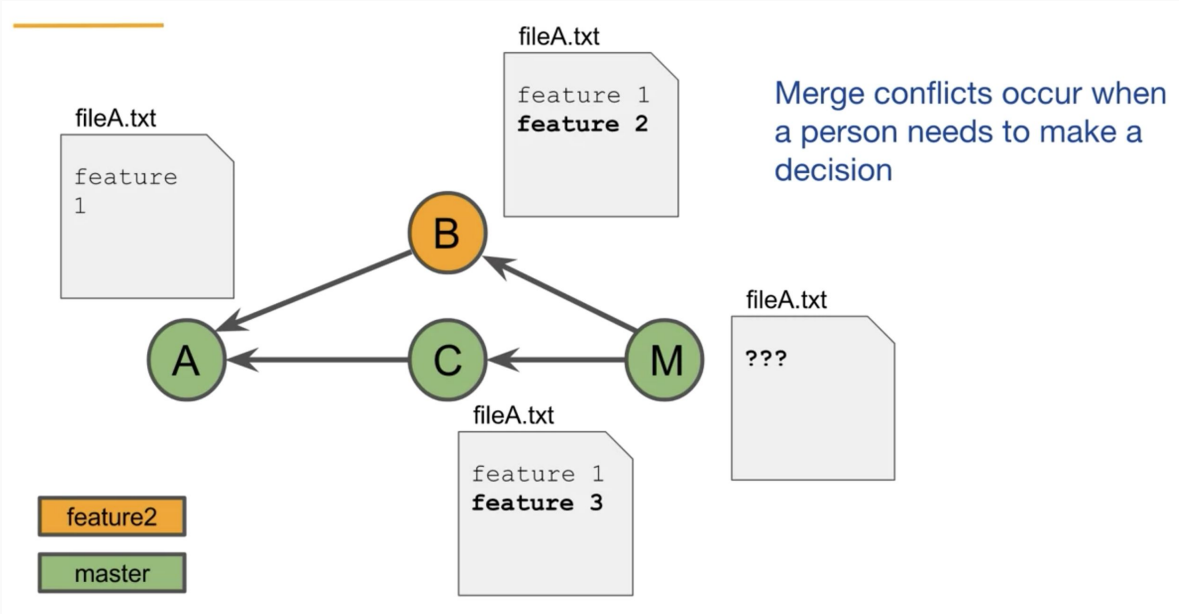
Resumen


develop
master
A
B
C
D
E
F
during
development
develop
master
A
B
C
D
E
F
G
after merge
commit

Resolver conflictos

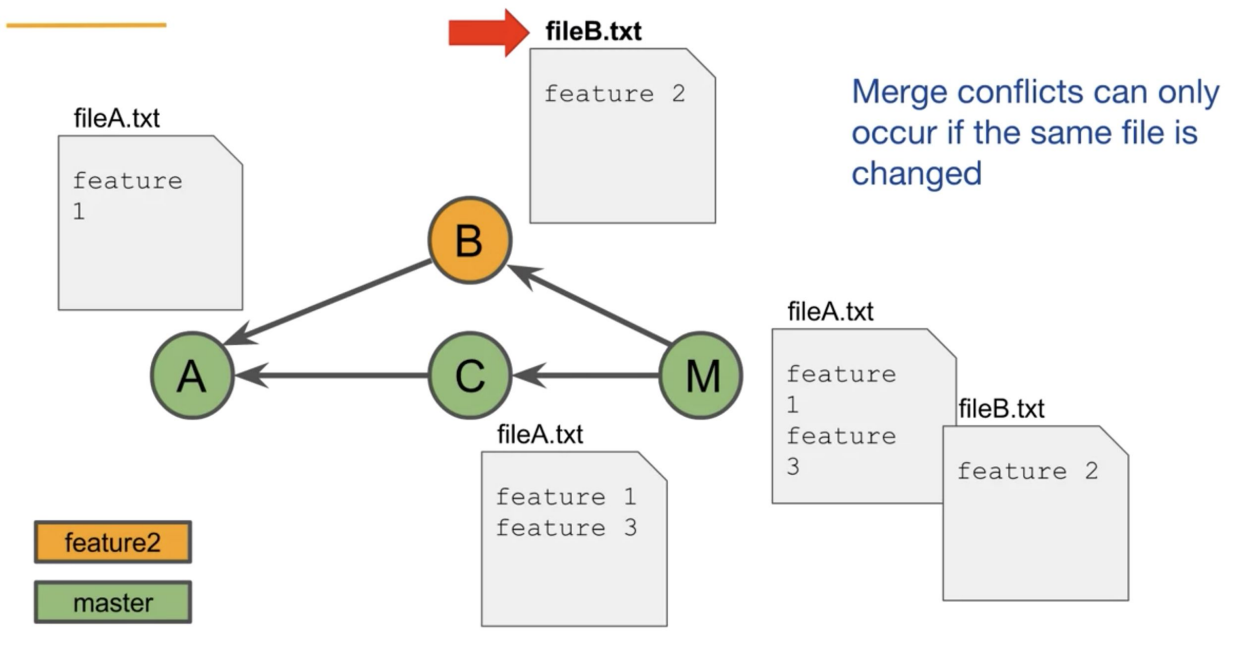


Resolver conflictos

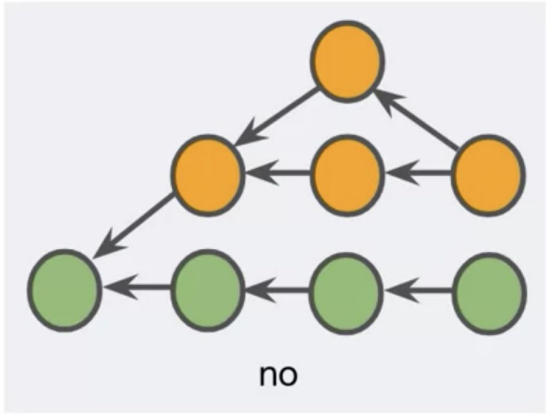


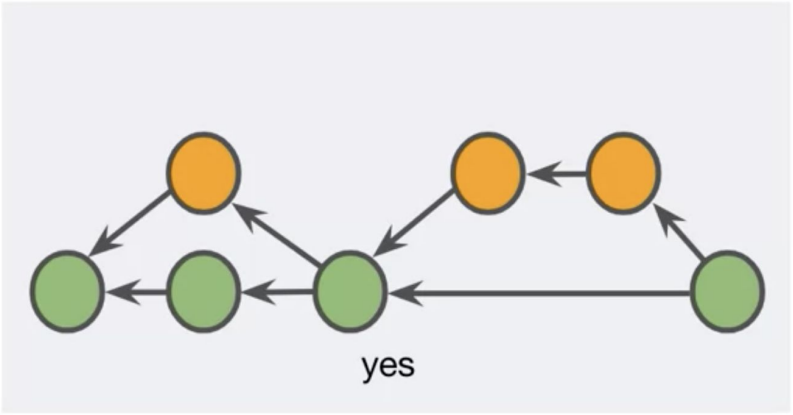
Resolver conflictos

Solo ocurre en los casos en que se modifica una misma línea en un mismo archivo



Recomendación en el manejo de ramas





Resolver conflictos

1. Checkout master
2. Merge featureX
   a. CONFLICT- Both modified fileA.txt
3. Fix fileA.txt
4. Stage fileA.txt
5. Commit the merge commit
6. Delete the featureX branch label

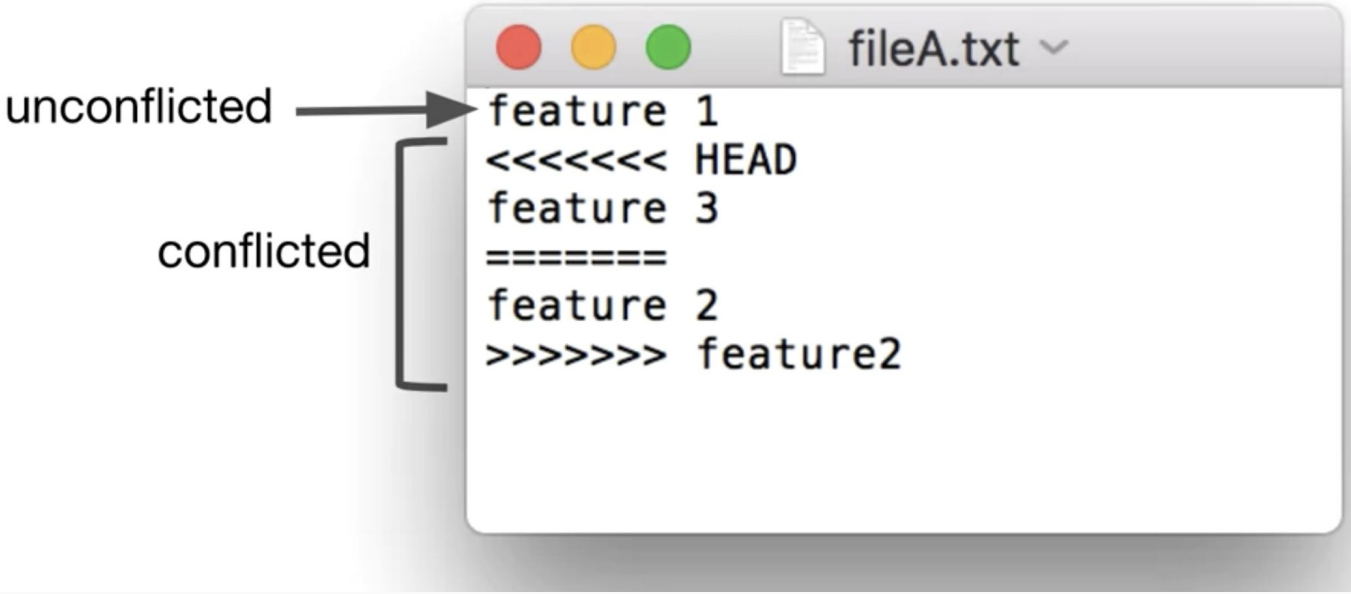
When attempting a merge, files with conflicts are **modified by Git** and placed in the working tree

Resolver conflictos

```
$ git log --oneline --graph --all
* c1633f9 (HEAD -> master) added feature 3
| * 942b91e (feature2) added feature 2
|/
* c431e4b added feature 1
$ git merge feature2
Auto-merging fileA.txt
CONFLICT (content): Merge conflict in fileA.txt
Automatic merge failed; fix conflicts and then commit the result.
$ git status
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)

    both modified:   fileA.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

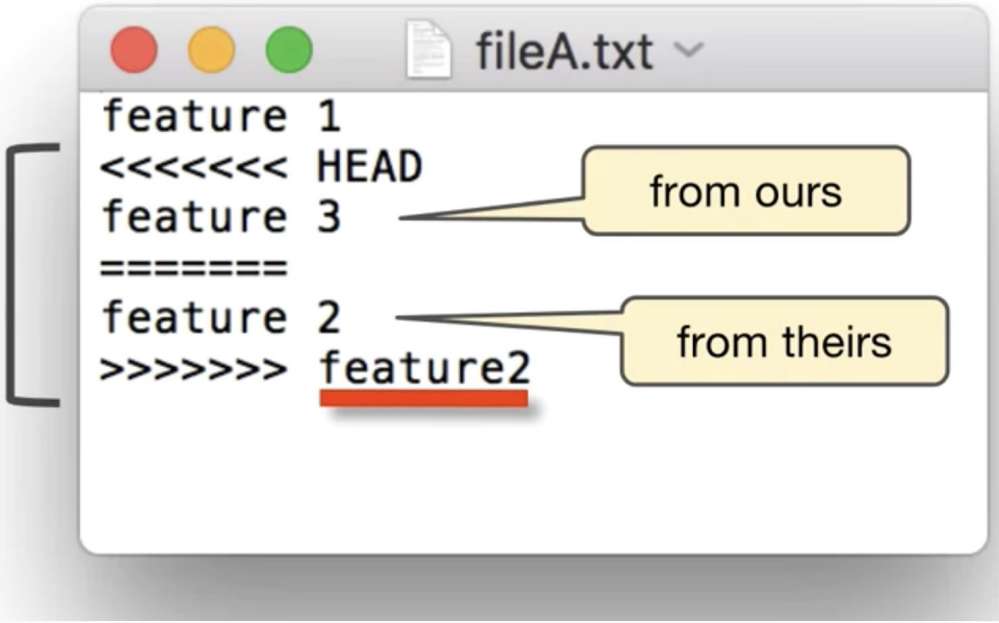
Resolver conflictos

Conflicted hunks are surrounded by conflict markers <<<<<<< and >>>>>>>



Resolver conflictos

- Text from the `HEAD` commit is between `<<<<<<<` and `=======`
- Text from the branch to be merged is between `=======` and `>>>>>>>`



Resolver conflictos

```
$ cat fileA.txt
feature 1
feature 2
feature 3
$ git add fileA.txt
$ git commit
(edit merge message if desired)
[master a8899d8] Merge branch 'feature2'
$ git log --oneline --graph --all
*   a8899d8 (HEAD -> master) Merge branch 'feature2'
|\
| * 942b91e (feature2) added feature 2
* | c1633f9 added feature 3
|/
* c431e4b added feature 1
$ git branch -d feature2
Deleted branch feature2 (was 942b91e).
```