



# git



# Git

## Agenda

1. Introducción a Git
2. Ramas y Tags
3. Sincronización
4. Github Flow




# Introducción a git

“”

**git** es un  
sistema **distribuido**  
de **control de versiones**

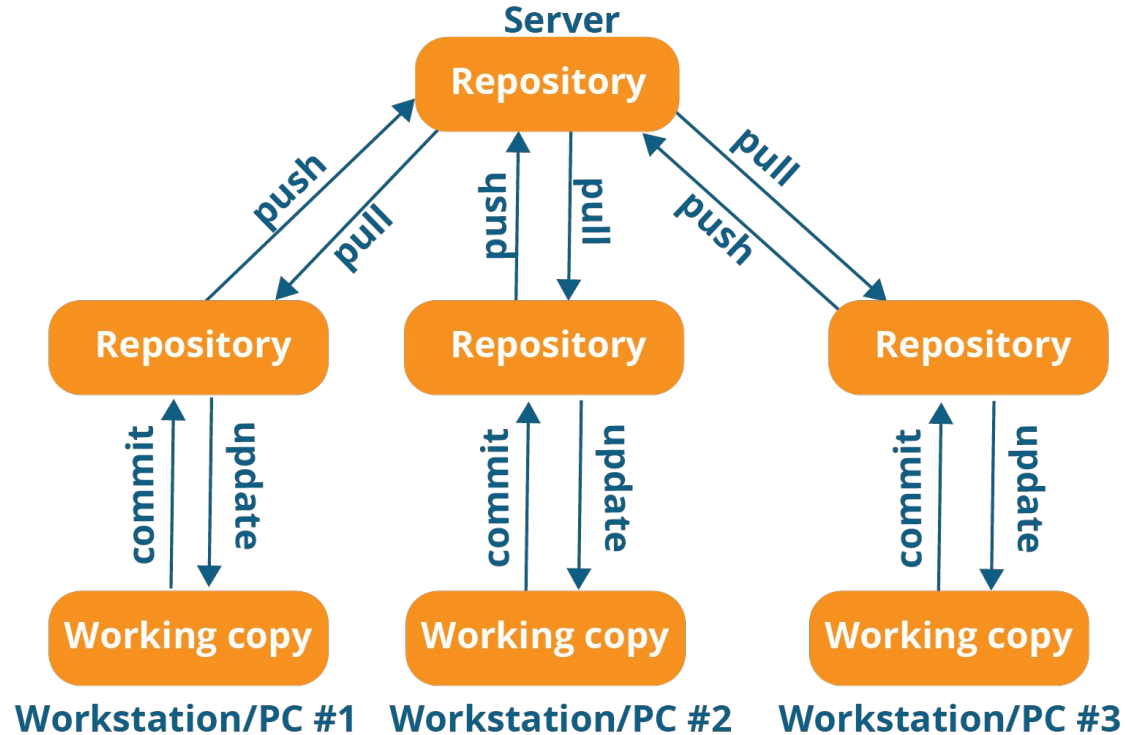
<https://git-scm.com/>

A photograph of two men in a dimly lit office environment, focused on a large computer monitor. The man in the foreground, wearing glasses and a dark shirt, is looking intently at the screen with his hand near his chin. The man behind him is also looking at the screen. The desk is cluttered with papers, a keyboard, and a mug. A desk lamp provides a warm, focused light on the workspace. The background is blurred, showing other office equipment and lights.

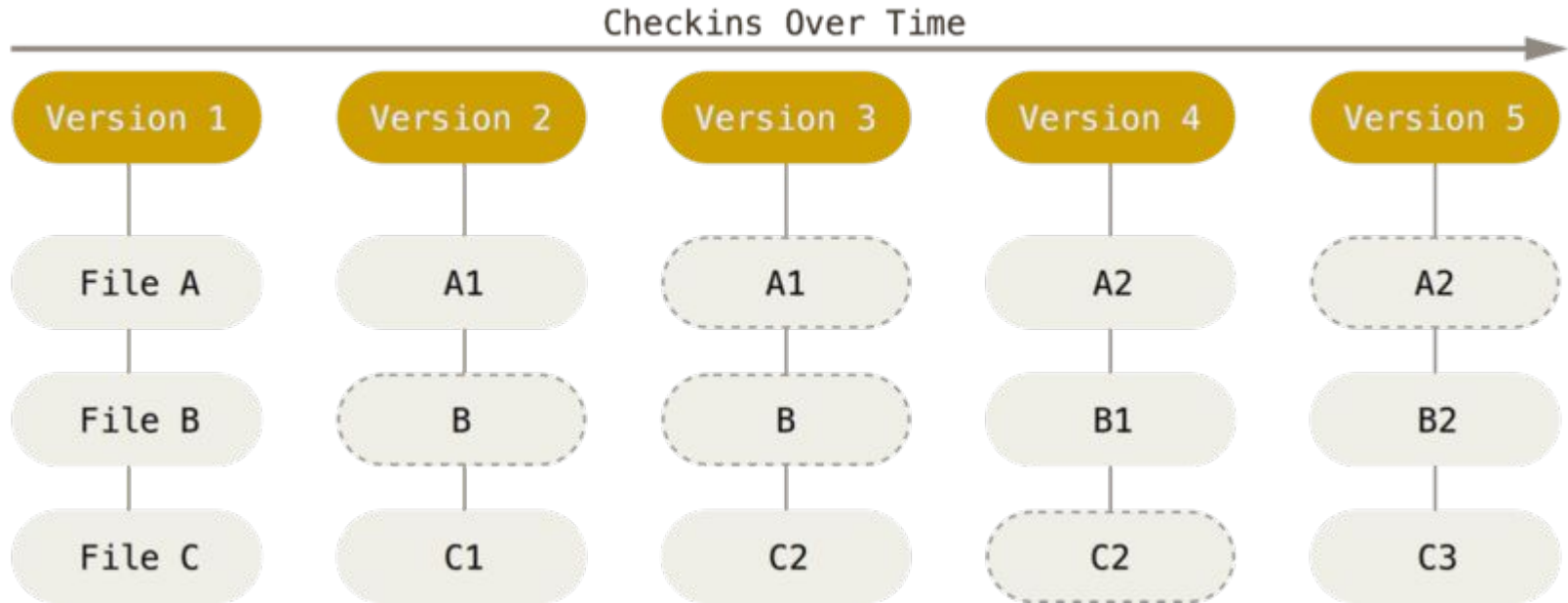
**Control de  
versiones**

es la **gestión**  
de los **cambios** que  
se realizan en un  
proyecto

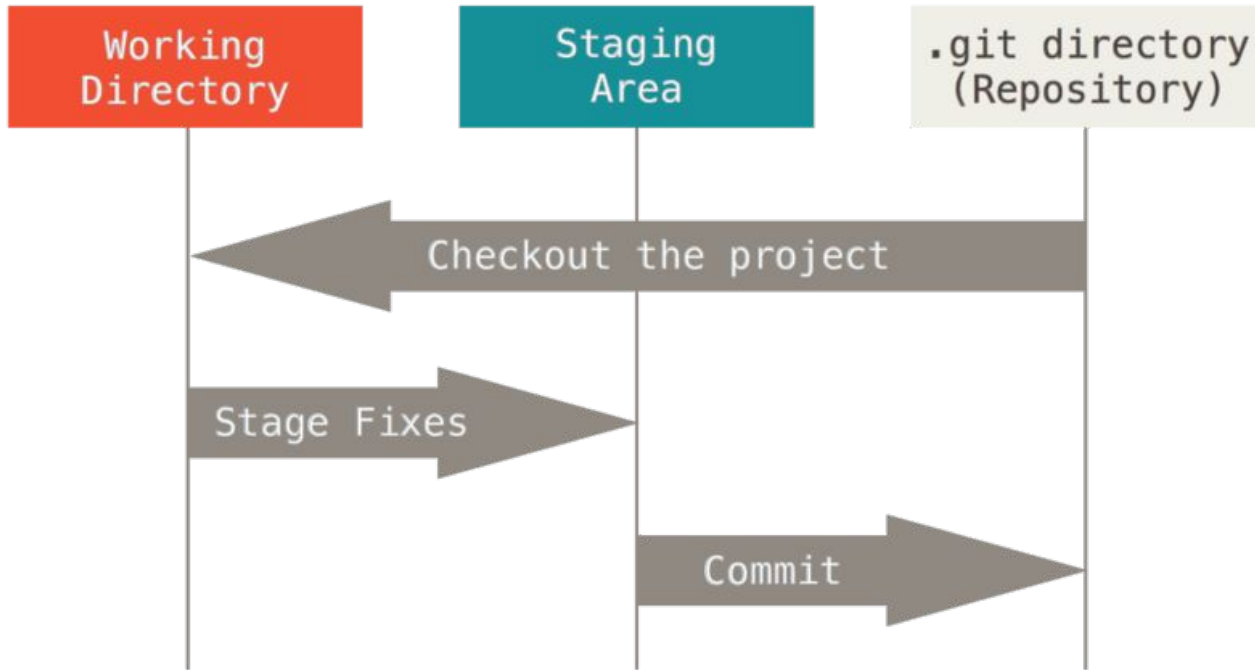
# Sistema distribuido de control de versiones



# Snapshots de cada versión

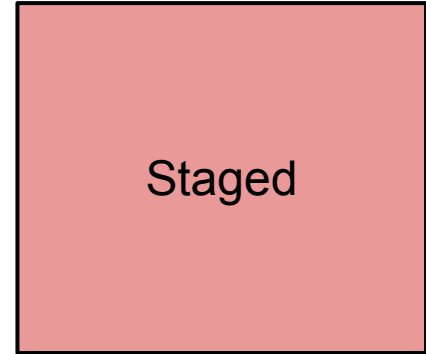
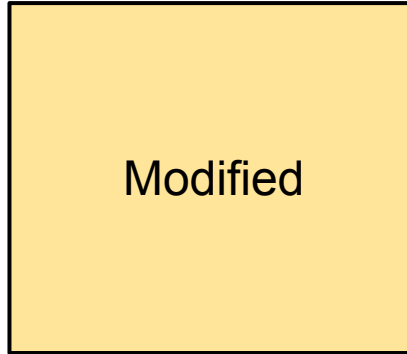
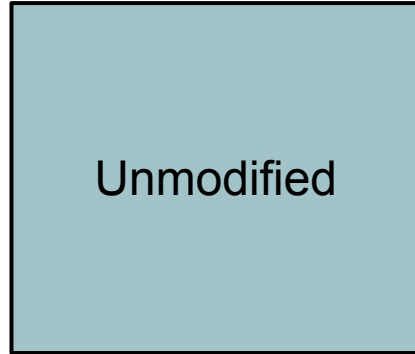


## 3 áreas locales en el repositorio

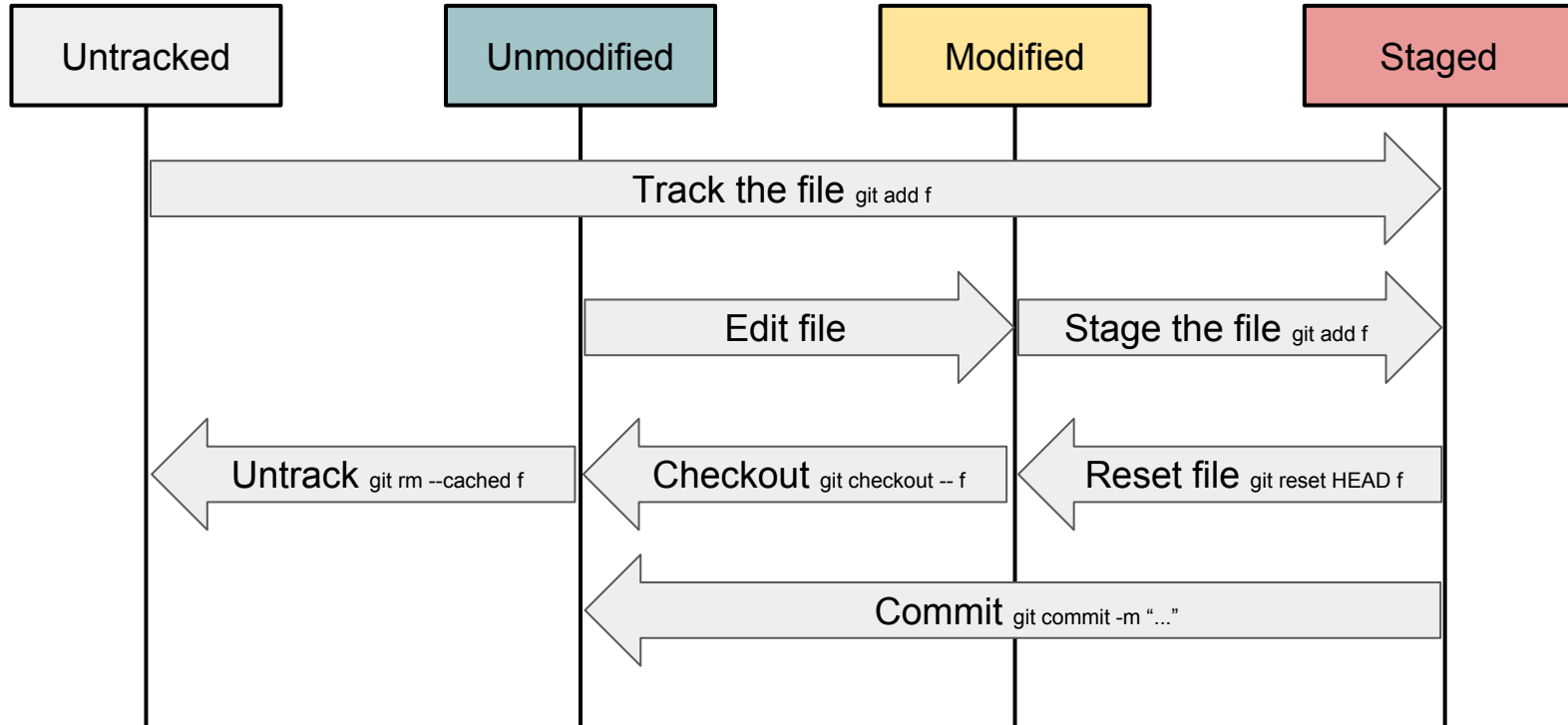




# 3 estados de cada archivo



# 3 estados de cada archivo



# Comandos cotidianos

`git help`

`git init` ⇒ Inicializar un nuevo repositorio en el directorio actual

`git status` ⇒ Estado actual del repositorio y el directorio de trabajo

`git add file` ⇒ Agregar archivo no-tracked/no-modificado al área de preparación

`git commit -m "mensaje"` ⇒ Confirmar los cambios preparados

`git diff [file]` ⇒ Ver diferencias con el repositorio

# Más comandos cotidianos

`git diff --cached` ⇒ Ver los cambios preparados hasta el momento

`git reset HEAD file` ⇒ Quitar los cambios del área de preparación para un archivo

`git rm file` ⇒ Borrar un archivo en el área de preparación y también en el directorio de trabajo

`git rm --cached file` ⇒ Borrar un archivo solamente en el área de preparación

`git log` ⇒ Ver un historial de los commits realizados

`git commit --amend` ⇒ Agregar cambios al último commit o arreglar el mensaje

`git stash` ⇒ Guardar los cambios actuales en la pila “stash” (general para el repositorio)

`git stash pop` ⇒ Quitar los cambios que hay en el tope de la pila “stash” y aplicarlos al WD

# Archivo .gitignore

# ignore all .a files

**\*.a**

# but do track lib.a, even though you're ignoring .a files above

**!lib.a**

# only ignore the TODO file in the current directory, not subdir/TODO

**/TODO**

# ignore all files in the build/ directory

**build/**

# ignore doc/notes.txt, but not doc/server/arch.txt

**doc/\*.txt**

# ignore all .pdf files in the doc/ directory and any of its subdirectories

**doc/\*\*/\*pdf**

# Documentación

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

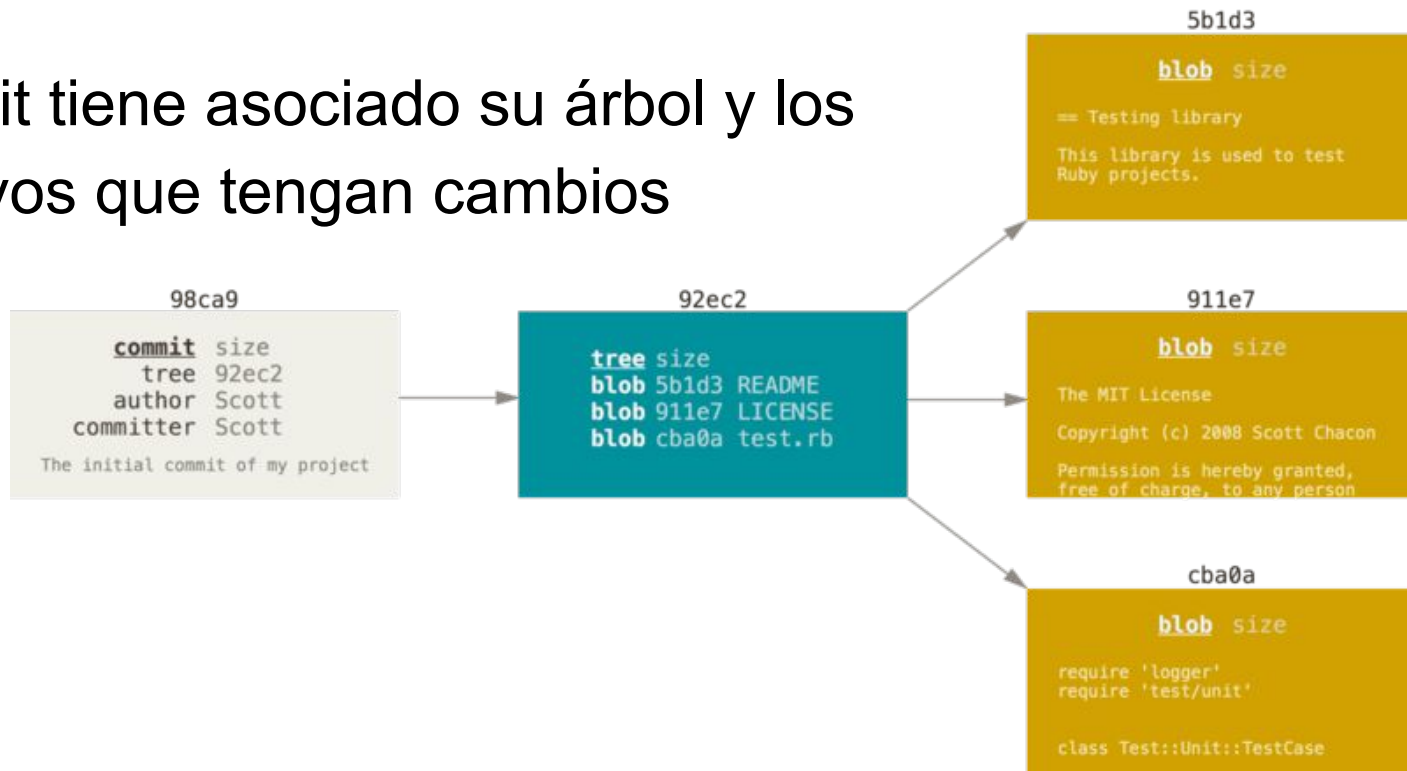
<https://git-scm.com/book/en/v2/Git-Basics-Git-Aliases>



# Ramas y Tags

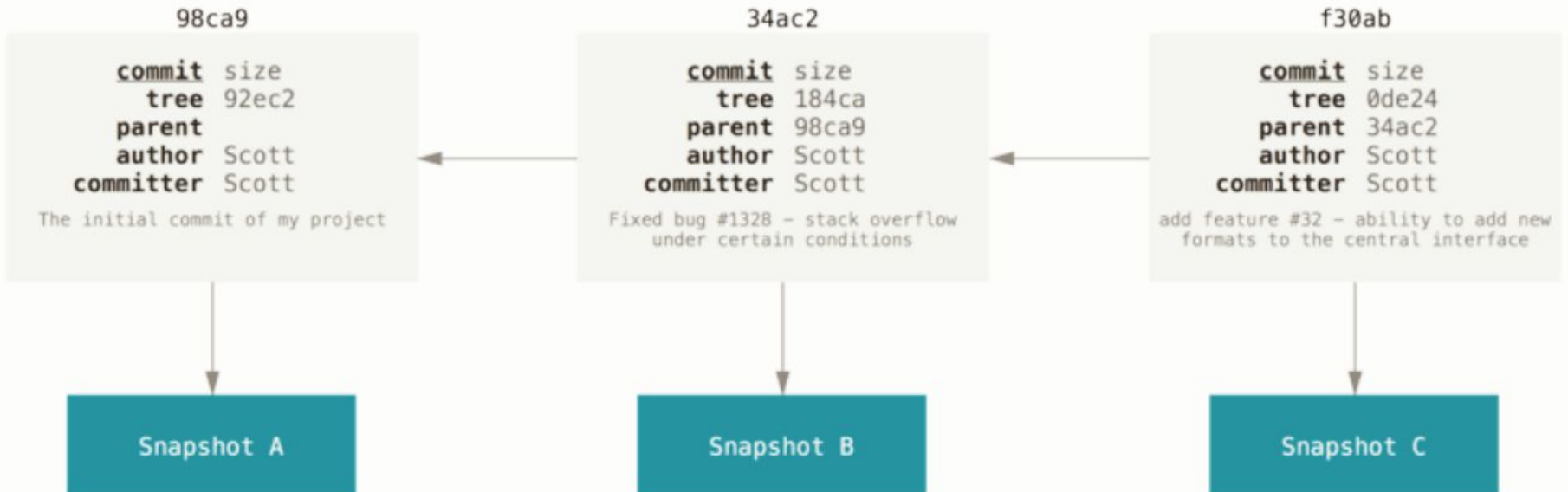
# Git almacena la información como snapshots

Cada commit tiene asociado su árbol y los archivos que tengan cambios





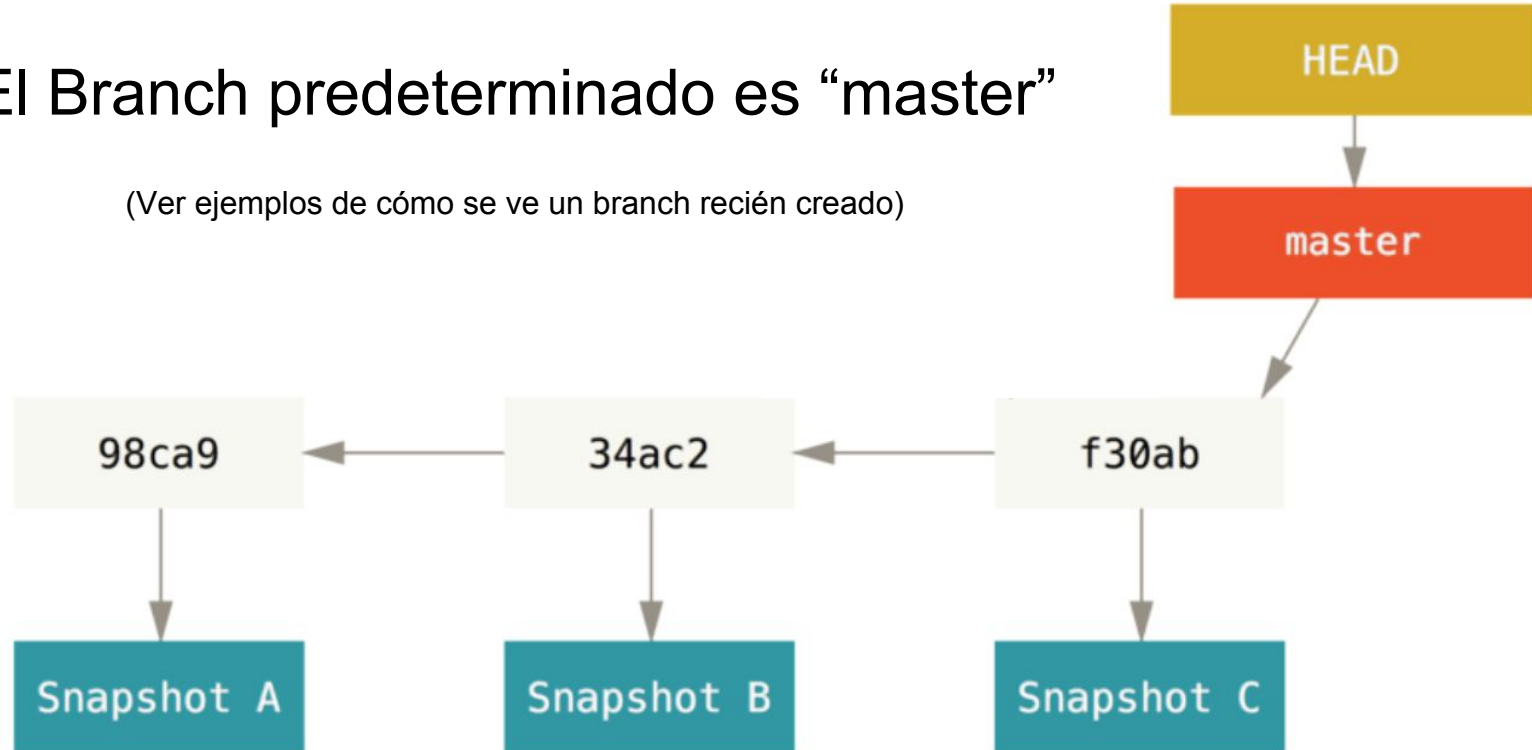
# Cada commit conoce al predecesor



# Un Branch es un puntero móvil a un commit

El Branch predeterminado es “master”

(Ver ejemplos de cómo se ve un branch recién creado)



# Comandos vinculados a ramas

`git branch` ⇒ Listar todos los branch locales

`git branch -a` ⇒ Listar todos los branches (incluyendo los remotos)

`git checkout BranchName` ⇒ Posicionar HEAD en un Branch Determinado

`git branch NewBranchName` ⇒ Crear un branch

`git checkout -b NewBranchName` ⇒ Crear un branch y mover HEAD al mismo

`git branch -d BranchName` ⇒ Eliminar un branch

# Branching & Merging

Estamos trabajando en un sitio web y creamos un **branch** “iss53” y trabajamos en esta rama. Nos llaman por un error en producción.

1

Cambiamos al  
branch  
“master”

2

Creamos un  
branch  
“hotfix”

3

Realizamos el  
fix, testeamos  
y subimos a  
producción

4

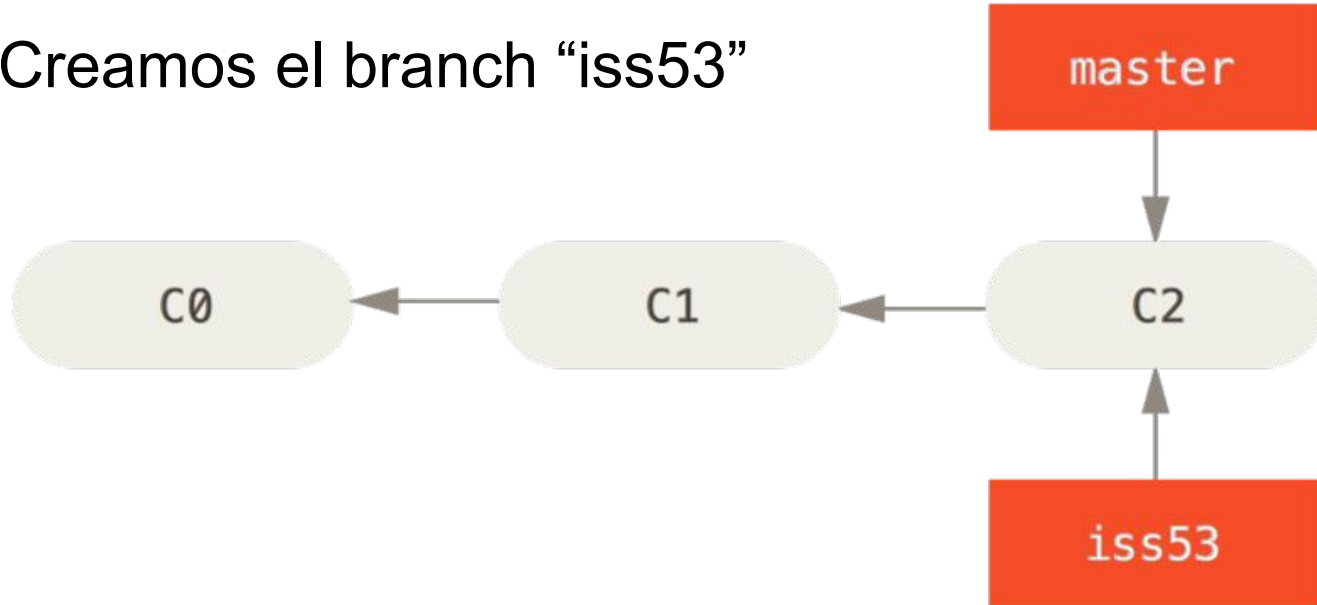
Mergeamos el  
branch  
“hotfix” en  
“master”

5

Cambiamos  
nuevamente  
al branch  
“iss53” en el  
que  
estábamos  
trabajando

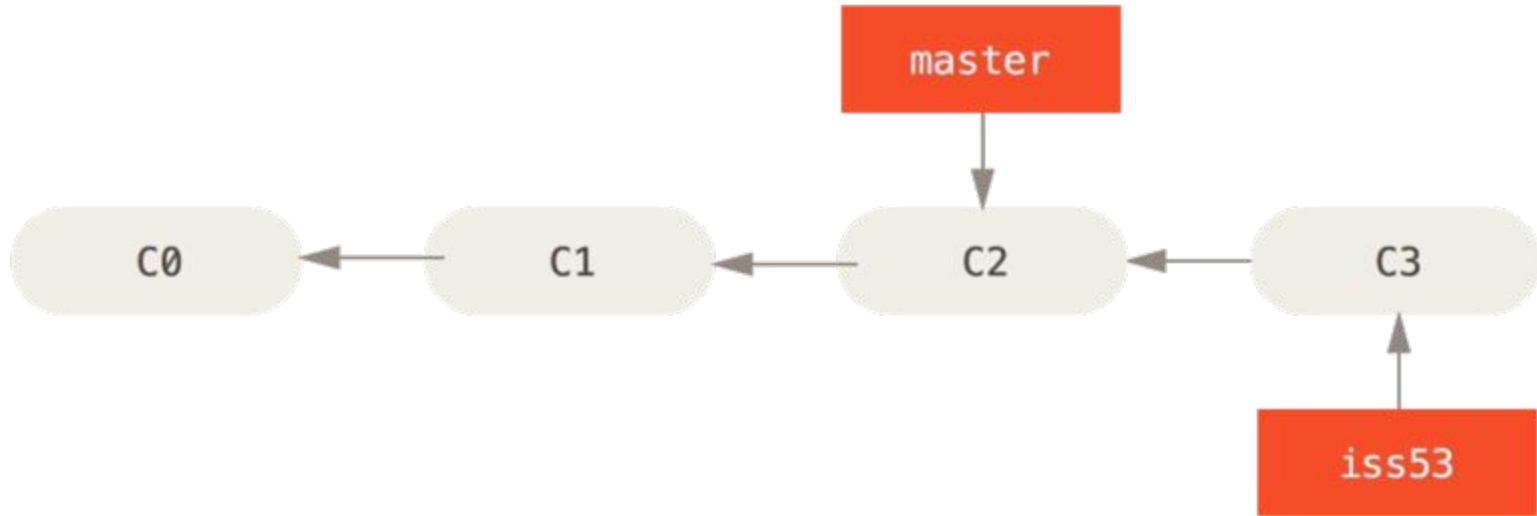
# Branching & Merging (1)

Creamos el branch “iss53”

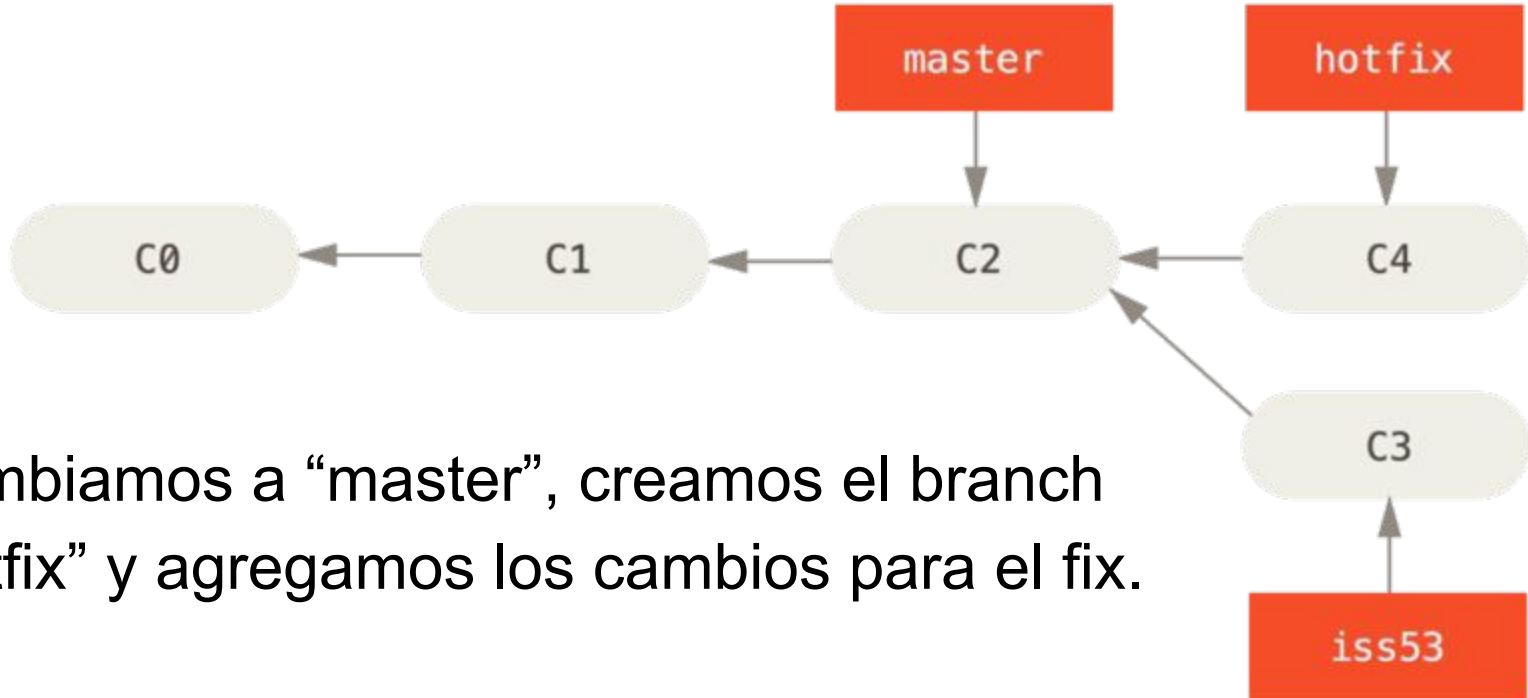


## Branching & Merging (2)

Hacemos algo de trabajo en el branch recién creado



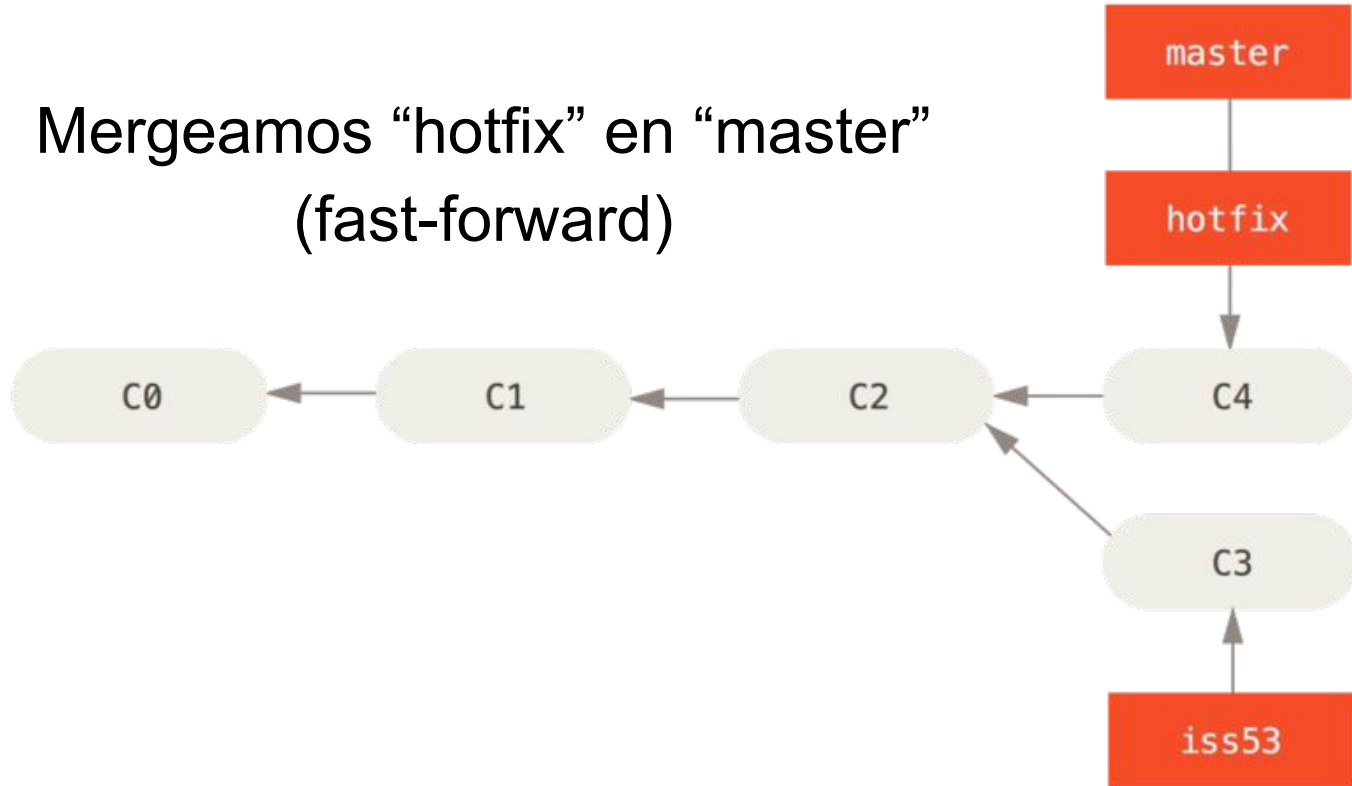
## Branching & Merging (3)



Cambiamos a “master”, creamos el branch “hotfix” y agregamos los cambios para el fix.

# Branching & Merging (4)

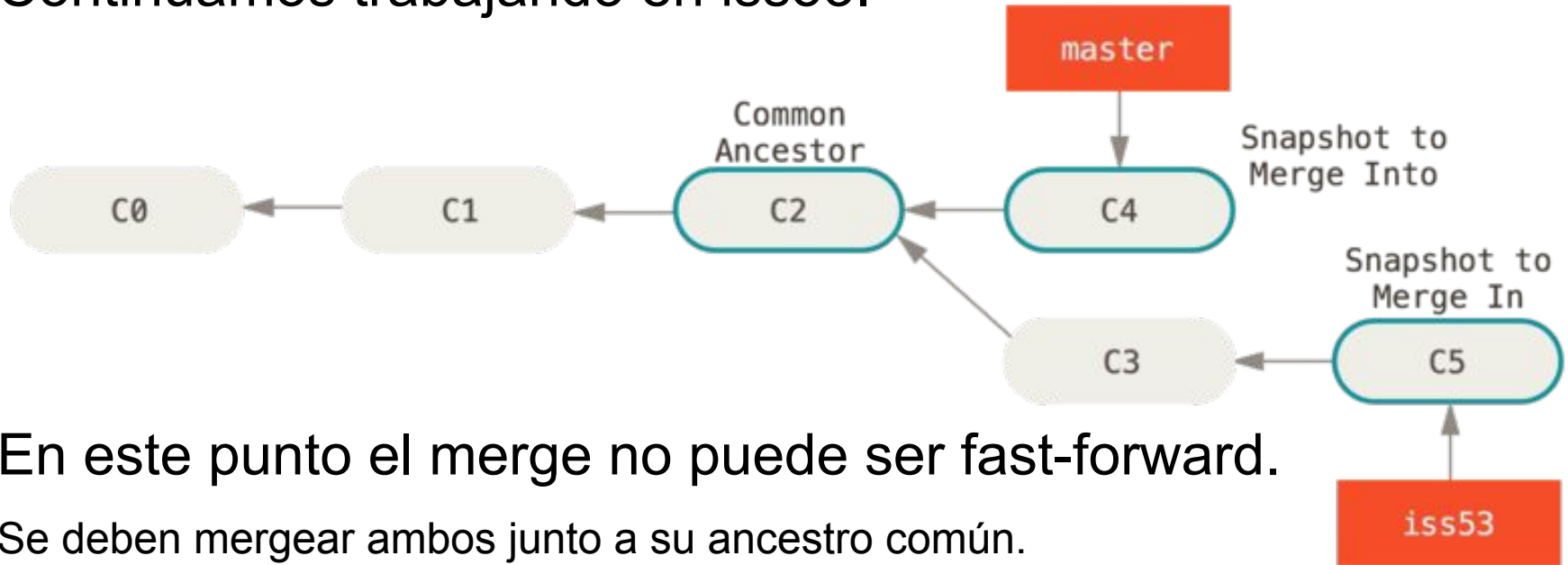
Mergeamos “hotfix” en “master”  
(fast-forward)





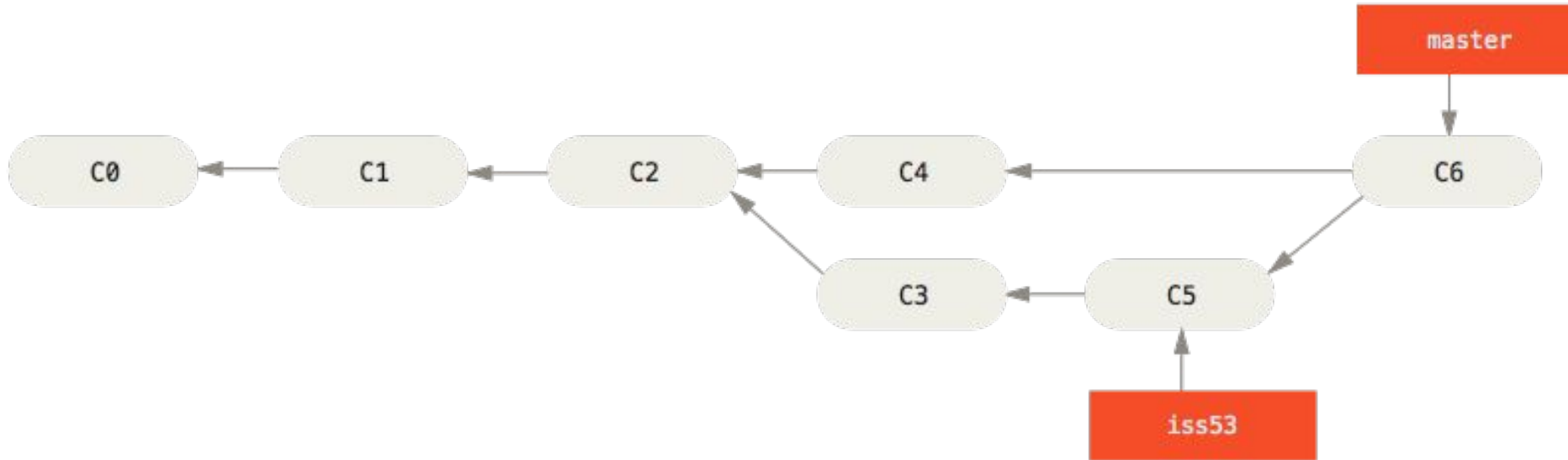
# Branching & Merging (5)

Continuamos trabajando en iss53.



# Branching & Merging (6)

El resultado es un commit de merge (que tiene 2 predecesores)



# Resolución de conflictos

Los **conflictos** son cambios que git no puede mezclar automáticamente.

Ejemplo: Hay dos cambios sobre un mismo archivo y hay que decidir cuál dejar, o si dejar ambos.

# Pasos en la resolución de conflictos

1. `git merge myBranch` *## conflicts found*
2. *## Fix conflicts manually*
3. `git add .`
4. `git commit` *## allows to edit default merge message*

```
<<<<<< HEAD:index.html
<div id="footer">contact : email.support@github.com</div>
=====
<div id="footer">
  please contact us at support@github.com
</div>
>>>>>> iss53:index.html
```

# Mergetool y otras herramientas

El comando **git mergetool** nos permite utilizar una herramienta que nos ayuda a resolver conflictos visualmente.

```
git config merge.tool meld  
git mergetool
```

# Tags

Etiquetas usadas para marcar puntos específicos de la historia (commits).

Típicamente se usan para marcar deploys.

# Comandos vinculados a tags

`git tag -l v1.*` ⇒ Ver una lista de todos los tags o filtrar por un patrón

`git tag v1.5` ⇒ Crear un nuevo tag en el commit actual

`git log --pretty=oneline` ⇒ Ver el log para obtener el hash de un commit a etiquetar

`git tag -a v1.2 9fceb02` ⇒ Etiquetar un commit pasado

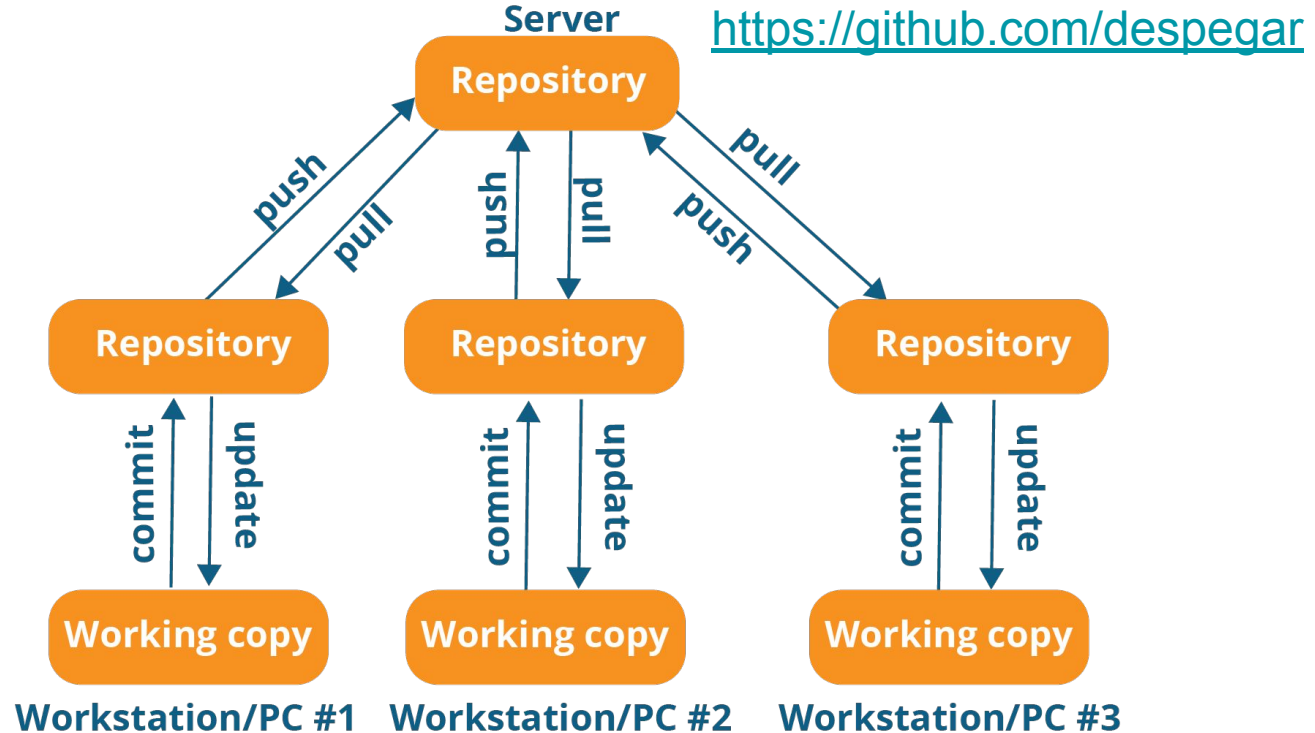
`git checkout 9fceb02` ⇒ Posicionar HEAD en un tag específico (detached HEAD).

The background is a dark blue surface with a faint, light blue line drawing. The drawing depicts a document or form with several rectangular boxes, some of which contain wavy lines. There are also several checkboxes, some of which are marked with an 'X'. The overall style is that of a hand-drawn sketch.

# Sincronización



# Github



# Comandos sincronización

`git remote -v` ⇒ Ver la lista de remotos

`git clone repositoryURL` ⇒ Clonar repo  
(git@github.com:despegar/jav-2018-q2-lp-nombreusuario.git)

`git remote add origin repositoryURL` ⇒ Vincular un remoto a un repo existente

`git push [origin master]` ⇒ Subir al remoto los datos locales del repositorio

`git pull [origin master]` ⇒ Traer al repositorio local los datos actualizados del remoto

`git push [origin master] --tags` ⇒ Subir al remoto las etiquetas locales del repositorio



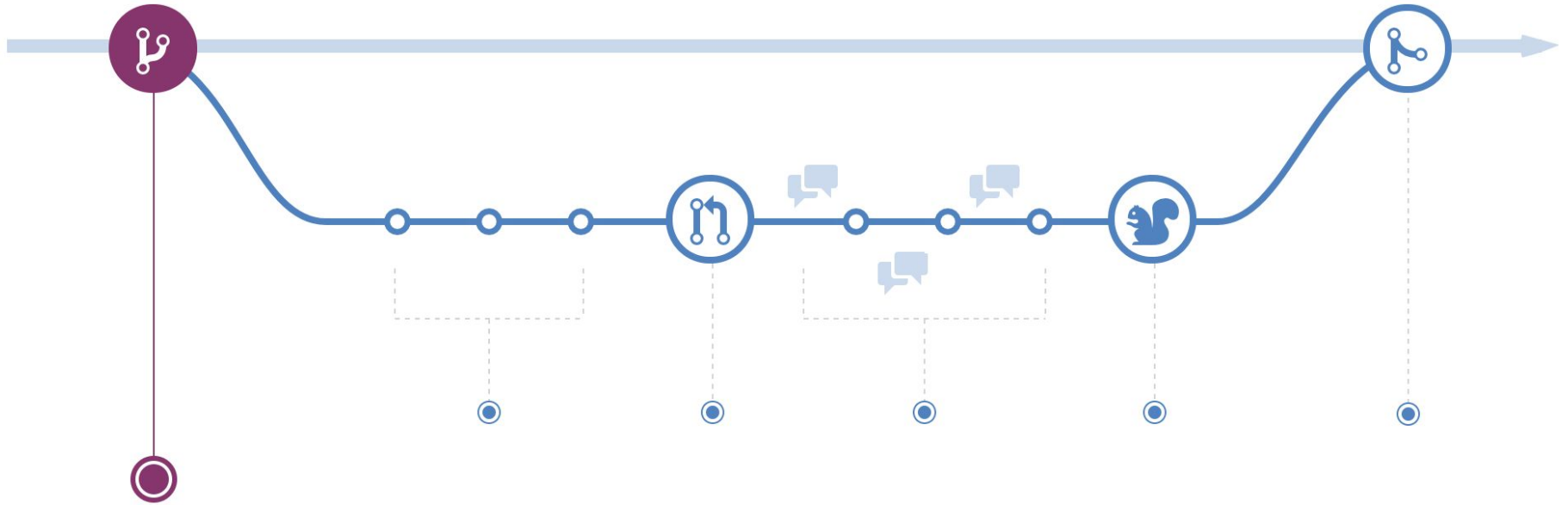
# Github flow

# Github flow

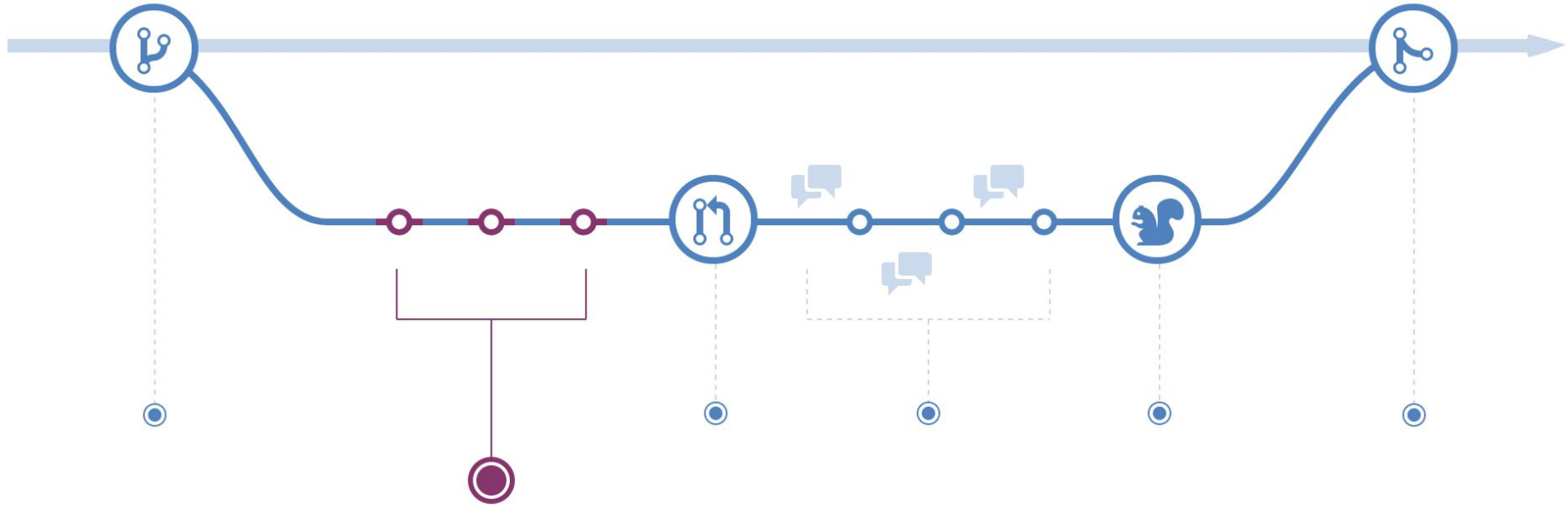
Modalidad de trabajo basada en ramas que ayuda a proyectos y equipos que hagan deploys frecuentemente.

<https://guides.github.com/introduction/flow/>

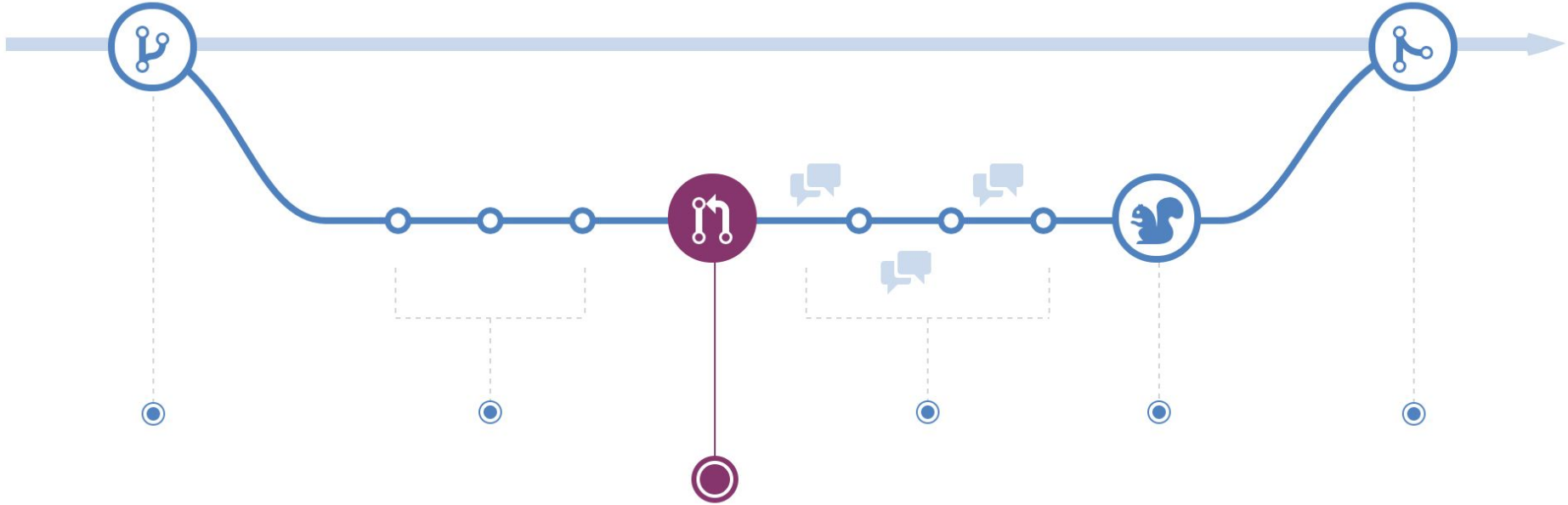
# 1. Crear un branch



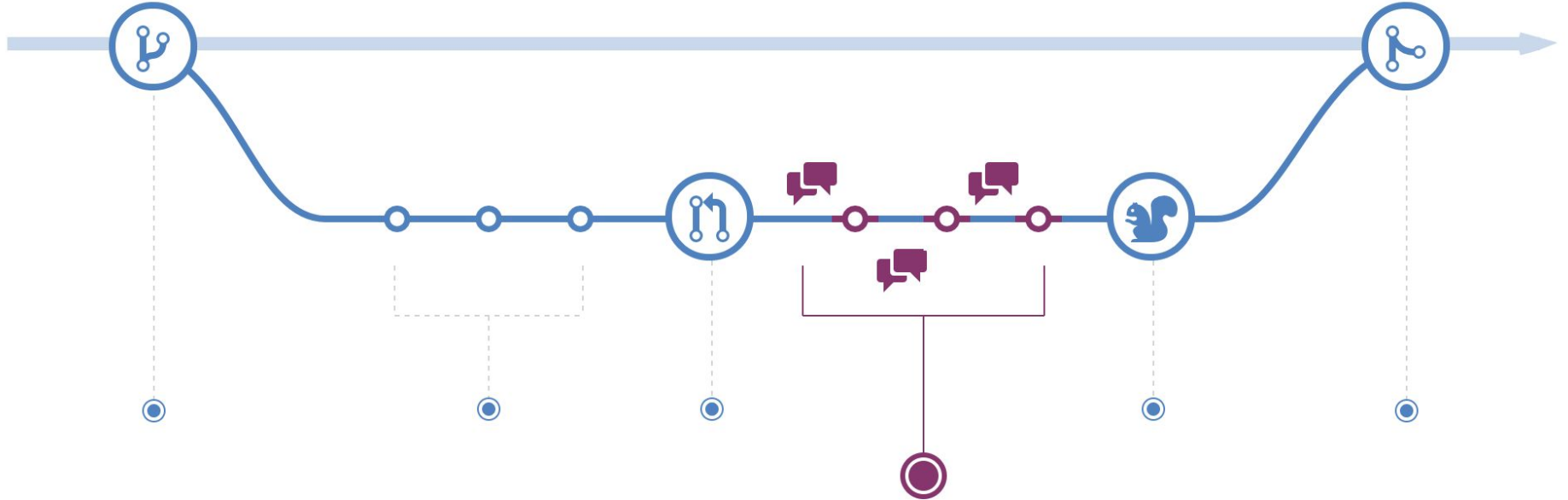
## 2. Agregar commits



### 3. Crear un Pull Request

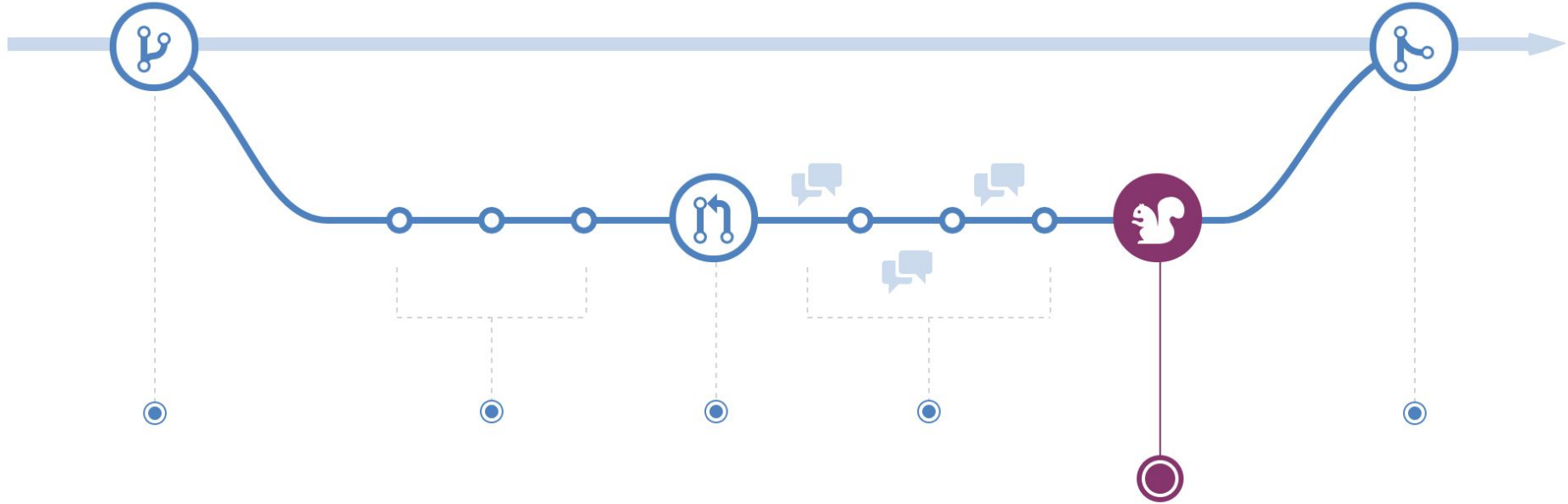


## 4. Revisión de código

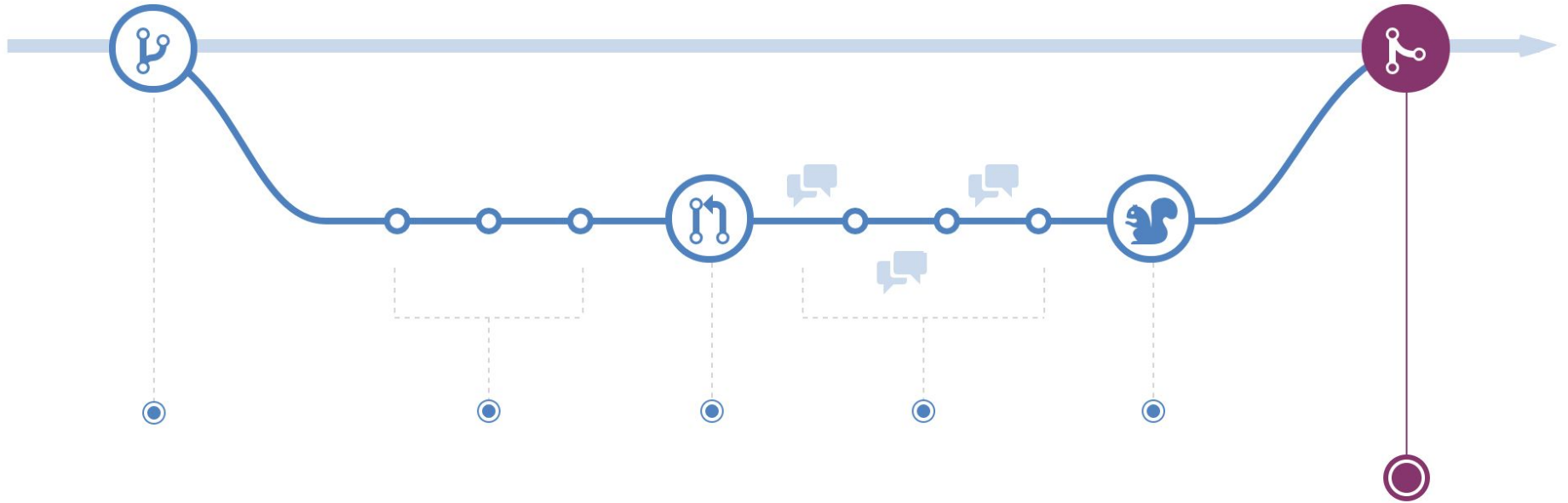




## 5. Deploy y prueba de funcionalidad



## 6. Squash & Merge



The background is a solid blue color with a faint, light blue line drawing. The drawing depicts a document or form with several rectangular boxes, some of which contain wavy lines representing text. There are also several checkboxes, some of which are marked with an 'X'. The overall style is that of a hand-drawn sketch.

# Resumen

# Resumen

## Estados

Modified-Staged-Committed, .gitignore

## Áreas

Staging - Repository - WD

## Ramas y tags

HEAD, master, v1.2

## Merging

ff & 3-way

## Sincronización

pull - push

## Github Flow

Pull Requests

# ¡Gracias!

