El futuro digital es de todos

MinTIC

GIT Y GITHUB

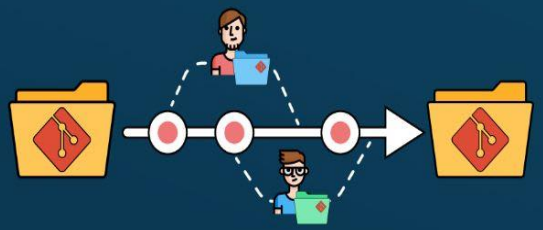
Mg. Richard E. Mendoza G.

unab VIGILADA MINEDUCACIÓN

Mision TIC 2022

¿SABES QUÉ ES GIT?

Es un sistema de **control de versiones**. Lleva un registro de todos los **cambios y avances** de tu proyecto.



v1 — v2 — v3

Funciona como una máquina de tiempo, puedes ir al pasado de tu código o volver al presente.

Github es un servicio que te ayuda **a almacenar tu proyecto en la nube**, además existen otros servicios como **Gitlab** o **Bitbucket.**

GIT TRABAJA CON RAMAS

Ayuda a que **varias personas trabajen** en un mismo proyecto y pueden realizar **modificaciones sin afectar a los demás** archivos. Una vez que estén listos los cambios se **fusionan con la rama principal.**

Prof. Beto Quiroga

Todo desarrollador sin importar el lenguaje **debe dominar Git.**

Aprende a trabajar en equipo con Git en:

ed.team/cursos/git-workflow

EDteam

¿CÓMO TRABAJAR EN EQUIPO CON GIT?

RAMA MASTER

(No puedes trabajar aquí): Es donde está el proyecto principal, nunca debes tocarla.

RAMA DEV

(Aquí es donde puedes trabajar): Saca las ramas que quieras y trabaja sin preocupaciones.

1. Cada desarrollador puede sacar su propia rama de la "rama dev" y trabajar en lo que le corresponde.

2. Luego estas ramas se vuelven a unir a través de Merge.

3. Si hay conflictos se deben corregir (Líder de proyecto o integración continua).

3. Se integran los cambios a la rama Master.

Y Master está actualizado con todos los cambios hechos por los programadores y la app funcionando.

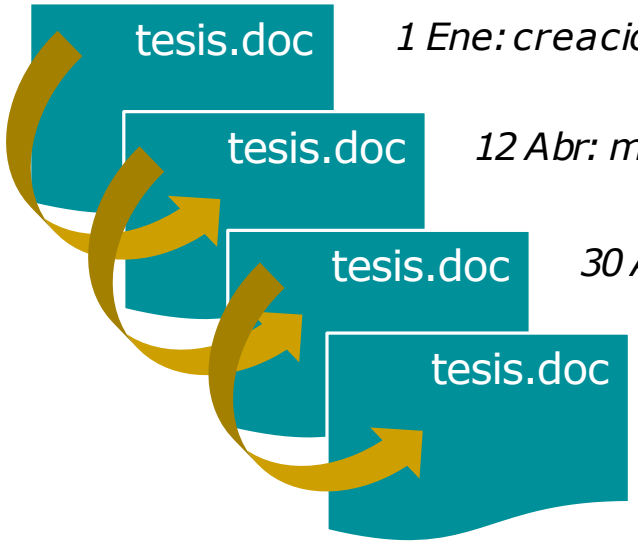
Aprende cómo usan Git los equipos de desarrollo de software en:

ed.team/cursos/git-workflow

EDteam


https://www.youtube.com/watch?v=iNFtX2ctExM
https://www.youtube.com/watch?v=4XpnKHJAok8

EL PROBLEMA



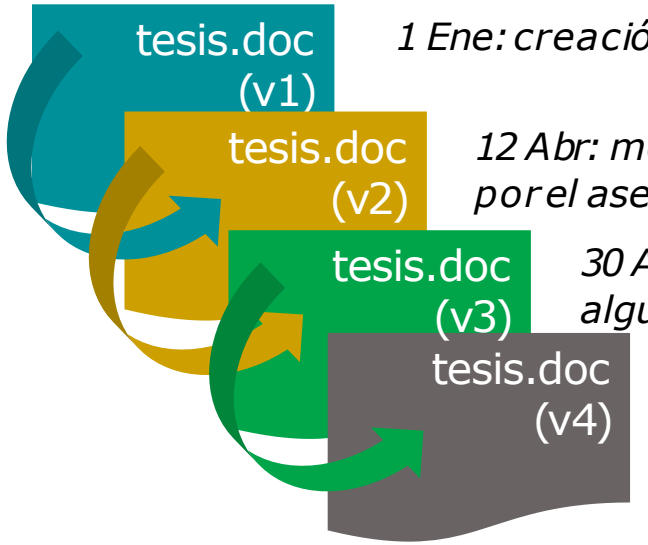
1 Ene: creación del archivo

12 Abr: modificaciones solicitadas por el asesor

30 Abr: corrección de algunos errores

21 Ago: ajustes de formato

2 Nov: necesito un texto eliminado anteriormente



1 Ene: creación del archivo

12 Abr: modificaciones solicitadas por el asesor

30 Abr: corrección de algunos errores

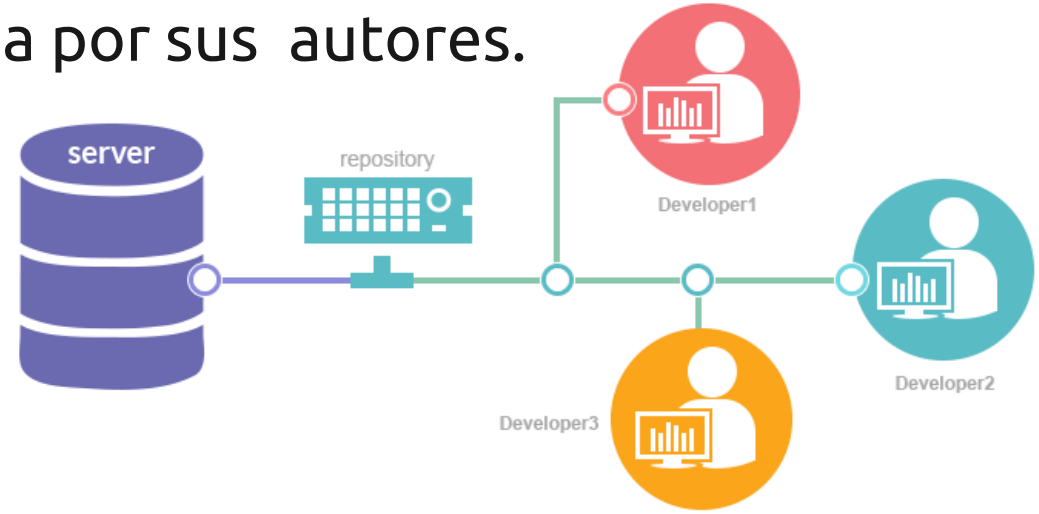
21 Ago: ajustes de formato

No tengo copias pasadas


git

CONTROL DE VERSIONES

- Es un sistema que registra los cambios sobre un conjunto de archivos a lo largo del tiempo, de modo que se pueda recuperarse.
- Estos sistemas utilizan un almacenamiento especial (Repositorio) para cada archivo y cada modificación hecha por sus autores.



El futuro digital es de todos | MinTIC

VENTAJAS DEL VCS

- Permite revertir archivos a un estado anterior.
- Comparar cambios a lo largo del tiempo.
- Ver quién es responsable de las modificaciones.
- Es un medio alternativo de backup del código fuente.
- Permite el desarrollo colaborativo.

git

SUBVERSION

monotone

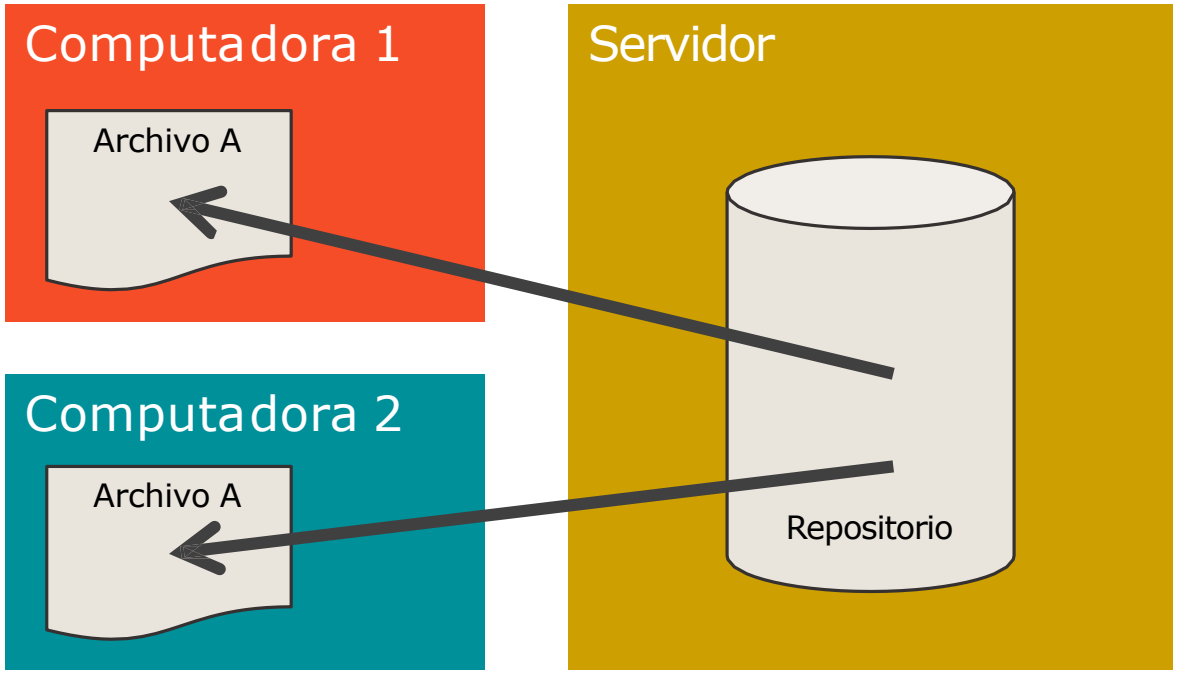
mercurial

unab
VIGILADA MINEDUCACIÓN

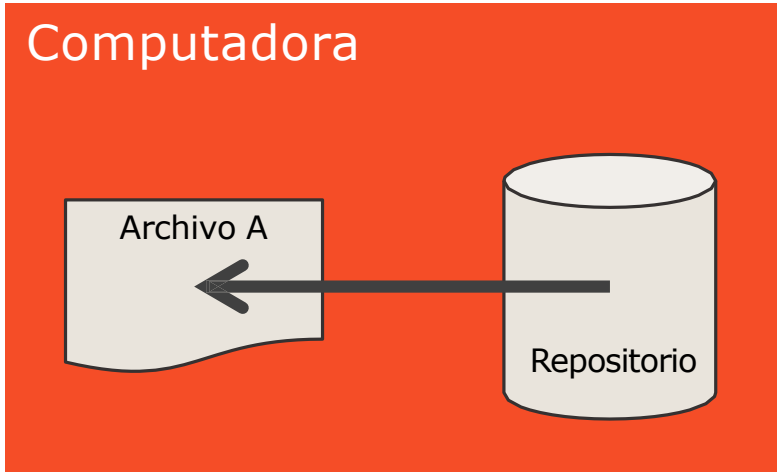
MisiónTIC2022

TIPOS DE VCS

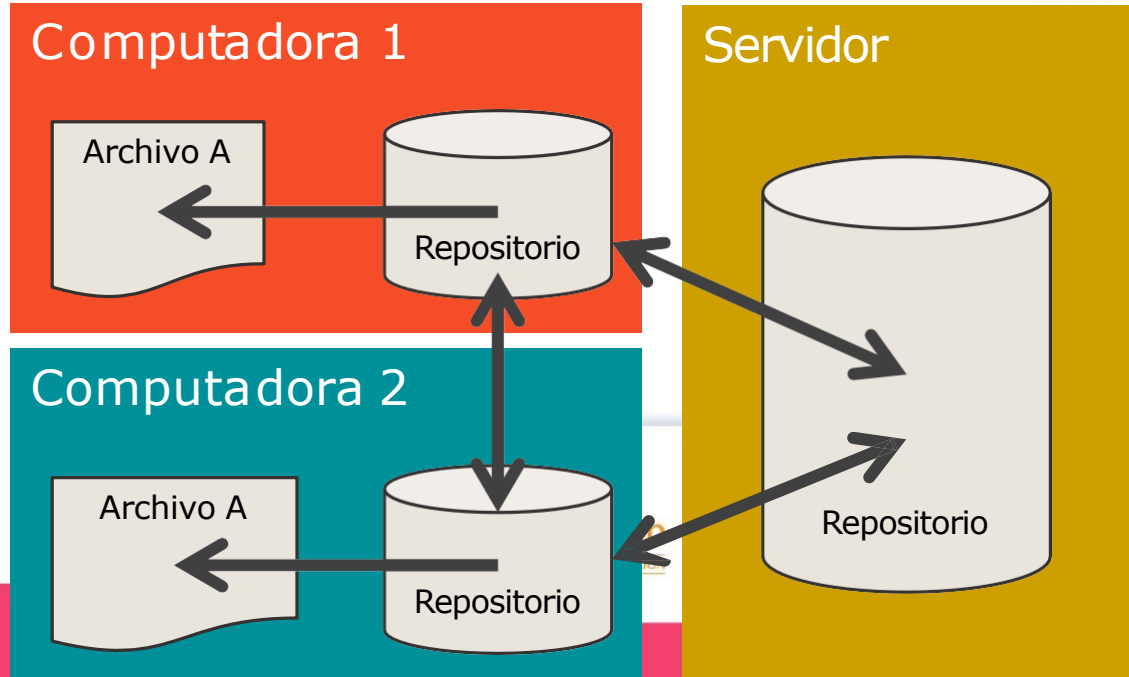
Centralizado



Local



Distribuido



¿Qué es GIT?

GIT es un software diseñado por Linus Torvalds que permite la gestión y el control de versiones de una aplicación Software. Esta se encuentra disponible en dos versiones:

Versión CLIENTE.

Versión SERVIDOR.

¿Qué es GITHUB?

- Servicio gratuito de almacenamiento de código fuente en la nube
- Fomenta la colaboración de proyectos abiertos
- Dispone planes para proyectos privados
- Dispone de herramientas como seguimiento de errores, wiki, etc.
- Utilizado por varios proyectos en la Web

[Actualización] Microsoft por fin adquiere Github por un total de 7,500 millones de dólares.

CARACTERISTICAS DE GITHUB

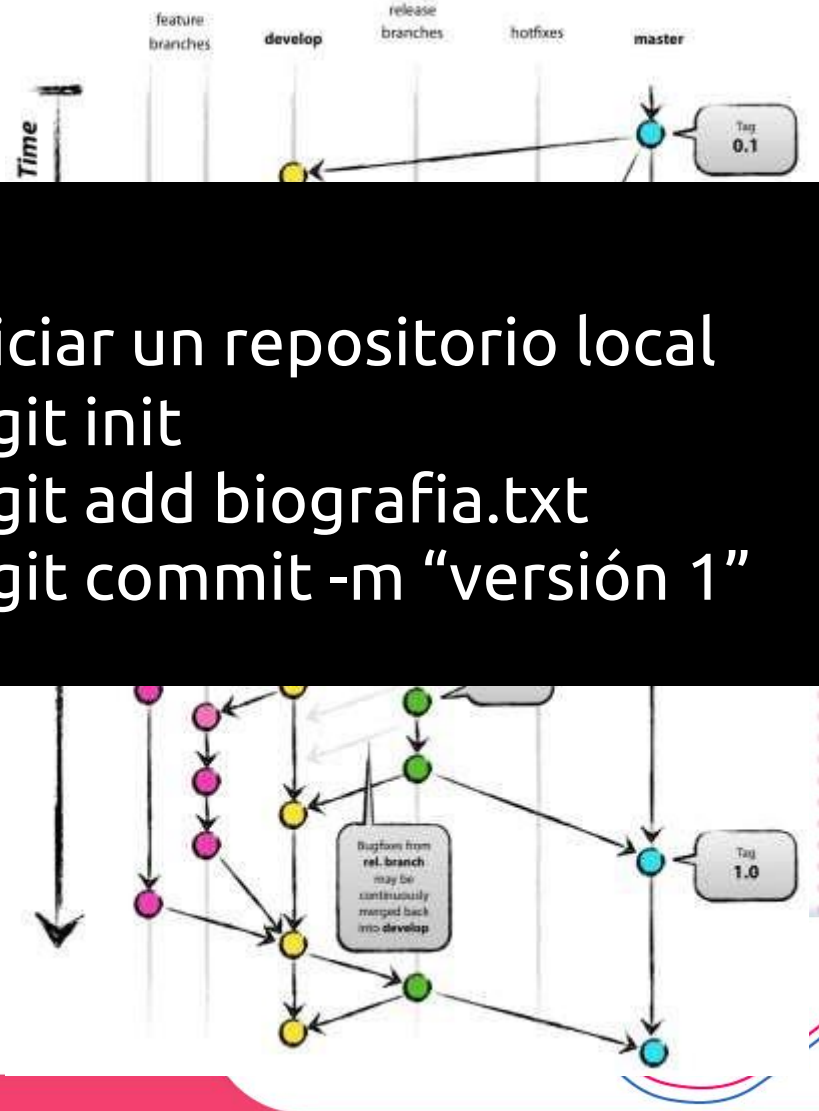
- Issue Tracking (Seguimiento de Incidencias).
- Soporte para Milestones (Hitos) y Labels (Etiquetas).
- Soporte para palabras claves en Commits (closes,fixes).
- Soporte para discusiones detalladas acerca de todos y cada uno de los commits realizados (por línea y por commit en su totalidad).

CONCEPTOS BASICOS

- **Repositorio:** Se denomina al sitio donde se almacenan los archivos del proyecto en forma centralizada.
- **Commit:** Consignación de un conjunto de cambios. Un commit genera una nueva versión. La misma tiene asociado un conjunto de cambios.
- **Branch:** Es una ramificación de un proyecto. De forma ideal (los proyectos open-source de hecho lo implementan) todo proyecto debería tener cuatro bifurcaciones:

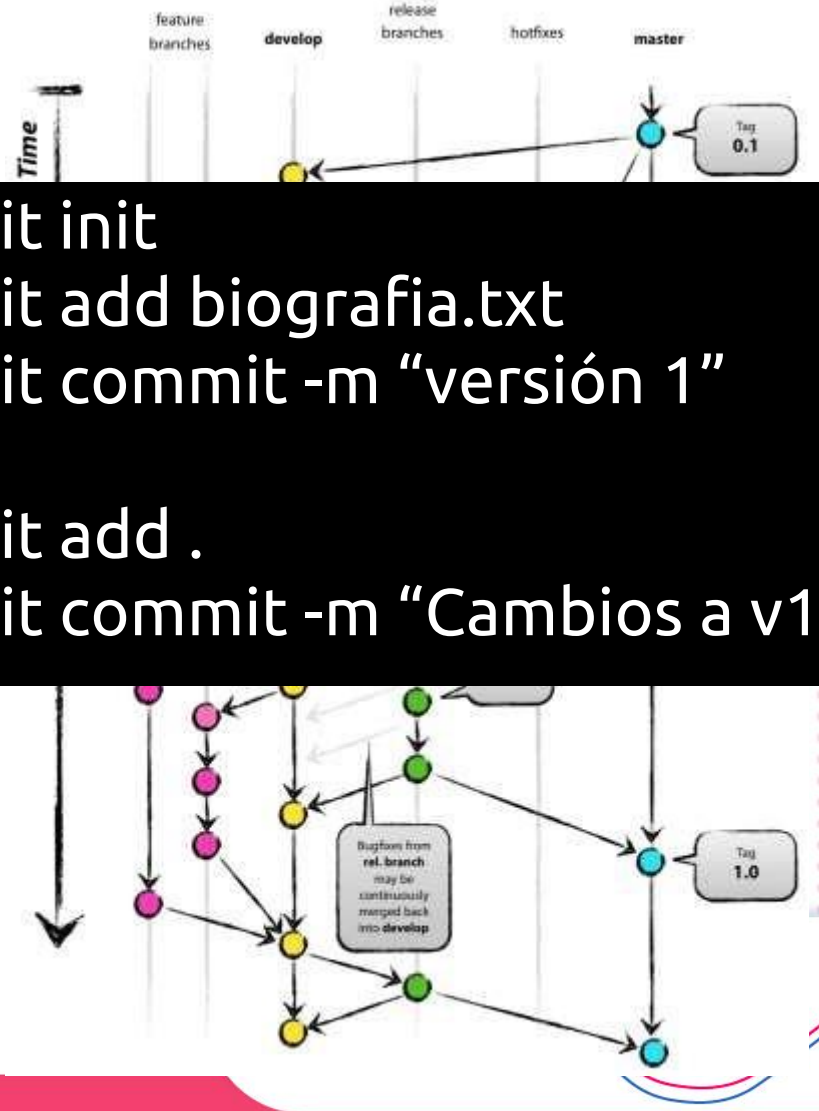
Master, Development, Features y Hotfixes

```
Iniciar un repositorio local
$ git init
$ git add biografia.txt
$ git commit -m “versión 1”
```



CONCEPTOS BASICOS

- **Master:** Es la rama principal. Contiene el repositorio que contiene la versión de la aplicación que se encuentra en producción, por lo que debe estar siempre en un estado “estable”.
- **Development:** Es un ramificación de master. Es la rama de integración de todas las nuevas funcionalidades. Luego que se realice la integración y se corrijan los errores (en caso de haber alguno), es decir que la rama se encuentre en un estado “estable”, se puede hacer una fusión entre las ramas de development y la rama master.

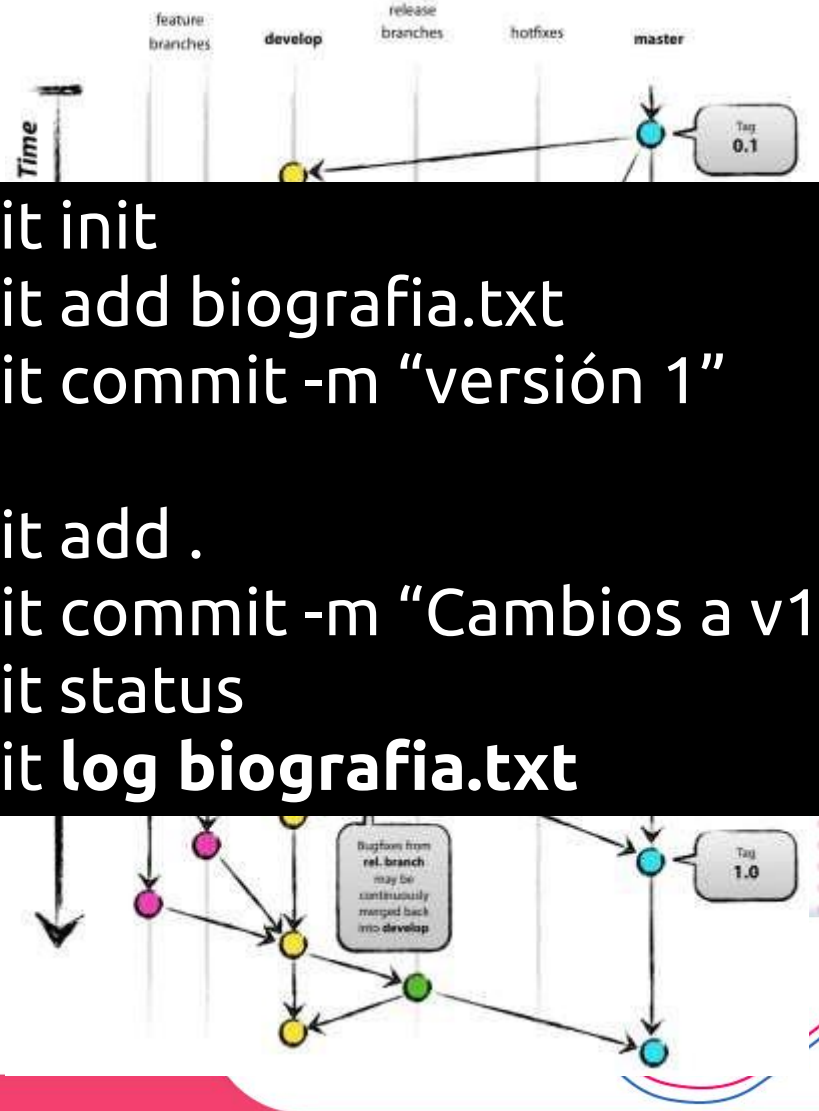



```
$ git init
$ git add biografia.txt
$ git commit -m “versión 1”

$ git add .
$ git commit -m “Cambios a v1”
```

CONCEPTOS BASICOS

- **Features:** Cada nueva funcionalidad se debe realizar en una rama nueva, específica para esa funcionalidad. Estas se deben sacar de development. Una vez que la funcionalidad se encuentre “cocinada”, se hace un fusión de la rama especifica de dicha funcionalidad sobre la rama development, donde se integrará con las demás funcionalidades.
- **Hotfix:** Esta rama se utiliza para solucionar bugs que surgen en la aplicación que se encuentra en producción, por lo que se deben arreglar y publicar de forma urgente.

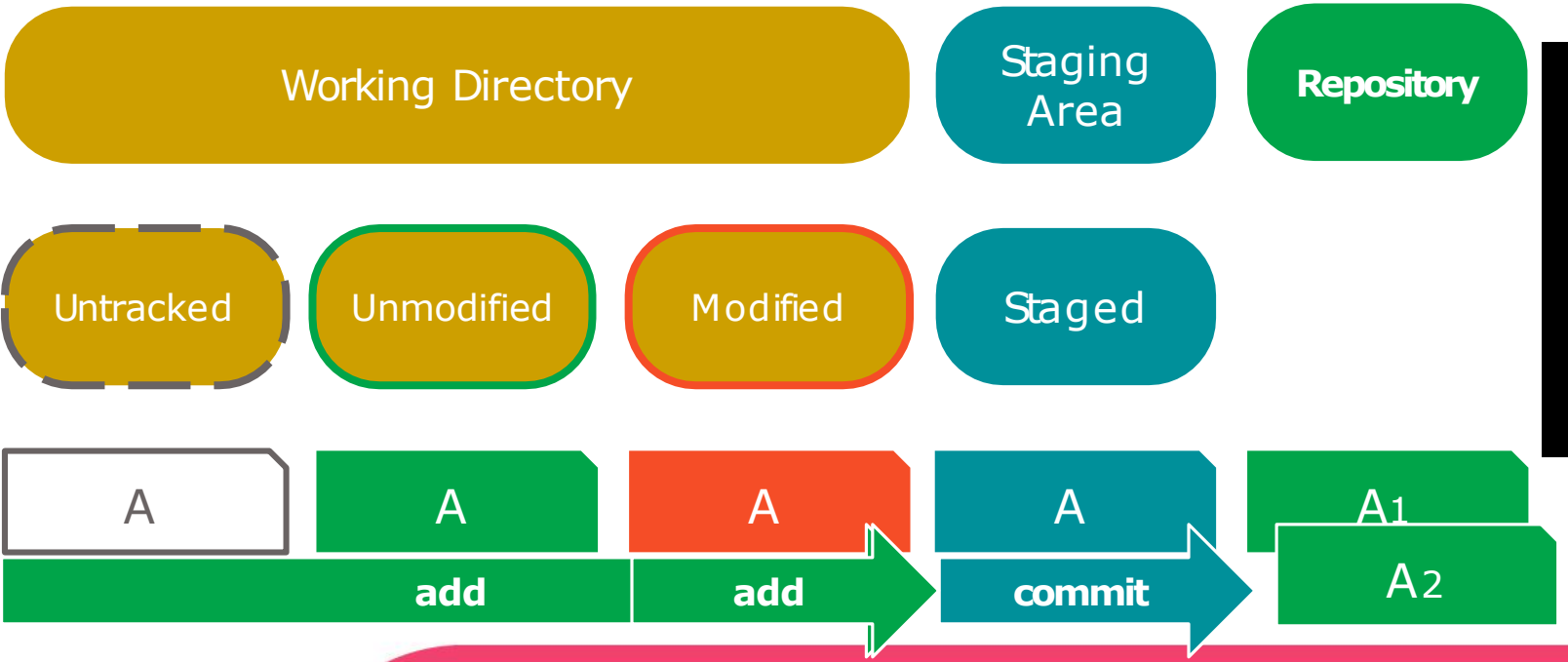



```
$ git init
$ git add biografia.txt
$ git commit -m “versión 1”

$ git add .
$ git commit -m “Cambios a v1”
$ git status
$ git log biografia.txt
```

CONCEPTOS BASICOS

- **Merge:** Es la fusión o mezcla de dos ramas del proyecto.
- **Staging Area:** Área de preparado de cambios.
- **Snapshot:** Imagen/Fotocopia del estado actual de un proyecto.
- **HEAD:** Cabecera que apunta al último snapshot (commit) realizado.



Configurar datos del usuario:

git config --global user.email "fperez@ejemplo.com"

git config --global user.name "Fulanito Perez"

CONCEPTOS BASICOS

- **clone:** Clonar un repositorio.
- **init:** Crear un repositorio en forma local.
- **add:** Agregar un documento a un área de preparación de cambios (Staging area).
- **commit:** Consignar un conjunto de cambios.
- **touch:** Crear un archivo.
- **reset:** Cambiar el estado del repositorio a un estado anterior.
- **checkout:** Crear una nueva rama/bifurcación del proyecto (repositorio) actual.

Iniciar un repositorio local:

cd ruta/a/mi/proyecto

git init

Sincronizar repositorio remoto en el actual:

git remote add origin http://url/del/repositorio/git.git

Se utiliza archivo .gitignore

touch .gitignore

Listar archivos ready to stage:

git status

Agregar archivos staging area.

git add mi_archivo.txt

git commit -m “Este es el primer commit.”

CONCEPTOS BASICOS

- **push:** Subir al servidor el/los cambios realizados (Una nueva versión).
- **pull:** Descargar y actualizar loscambios realizados en el repositorio remoto.
- **diff:** Ver los cambios realizados entre dos versiones (Línea por línea).
- **log:** Ver un log de los cambios realizados.
- **merge:** Unificar, mezclar cambios realizados en dos ramas y/o bifurcaciones del proyecto.
- **branch:** Lista, crear o eliminar ramas y/o bifurcaciones del proyecto.

Listar la lista de commits realizados.

git push -u origin <<nombre_branch>>

Se sincroniza en el branch “master”.

git push -u origin master

Crear un branch (Ramificación):

git branch <<nombre_branch>>

Crear un branch y cambiar al mismo:

git checkout -b <<nombre_branch>>

Listar todas las ramas locales:

git branch -a //Locales y remotas.

git branch -d <nombre_rama>//Eliminar rama

Fusionar las ramas master y hotfix.

git checkout master //cambiar a la master.

git merge hotfix //Fusionar con master

Overview Release Notes Help

GitHub Desktop

Focus on what matters instead of fighting with Git. Whether you're new to Git or a seasoned user, GitHub Desktop simplifies your development workflow.

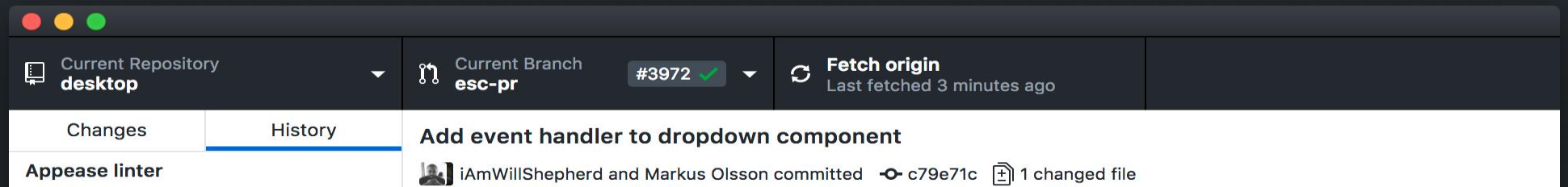
Download for macOS

Download the native macOS build for Apple silicon machines. If you don't know whether your machine has an Apple or Intel chip, see the Apple docs.

Download for Windows

By downloading, you agree to the Open Source Applications Terms.

https://docs.github.com/es/desktop/installing-and-configuring-github-desktop/installing-and-authenticating-to-github-desktop/installing-github-desktop



The entire **Pro Git book** written by Scott Chacon and Ben Straub is available to read online for free. Dead tree versions are available on Amazon.com.

Downloads

macOS Windows

Linux/Unix

Older releases are available and the Git source repository is on GitHub.

GUI Clients

Git comes with built-in GUI tools (**git-gui**, **gitk**), but there are several third-party tools for users looking for a platform-specific experience.

View GUI Clients →

Logos

Various Git logos in PNG (bitmap) and EPS (vector) formats are available for use in online and print projects.

View Logos →

Git via Git

https://git-scm.com/downloads

If you already have Git installed, you can get the latest development version via Git itself:

```
git clone https://github.com/git/git
```

You can also always browse the current contents of the git repository using the web interface.