

Git y GitHub

DEV.F
DESARROLLAMOS(PERSONAS);

dev

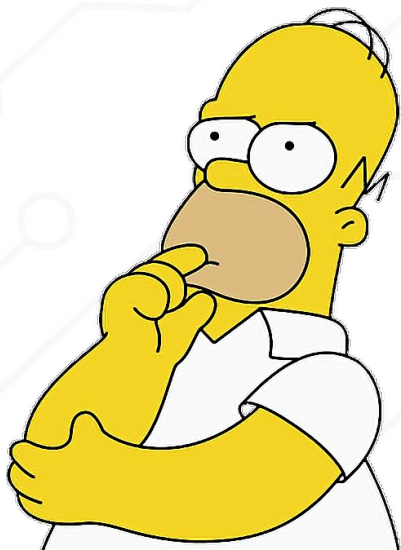




Git es un Sistema de Control de Versiones

**Github es un
repositorio remoto
(almacenamiento en internet)**

¿Que significa controlar una versión?



- segunda revision >
- tesis-corregida >
- tesis-corregida copy >
- tesis-revisada >
- tesis-v-finañ >
- tesis1 >

¿Que significa controlar una versión?

- segunda revision >
- tesis-corregida >
- tesis-corregida copy >
- tesis-revisada >
- tesis-v-finañ >
- tesis1 >



Sistema de control de versiones (VCS)

Un **sistema de control de versiones (VCS - Version control system)** es aquel que nos permite llevar un historial y control de cambios a medida que las personas y los equipos colaboran en proyectos juntos.

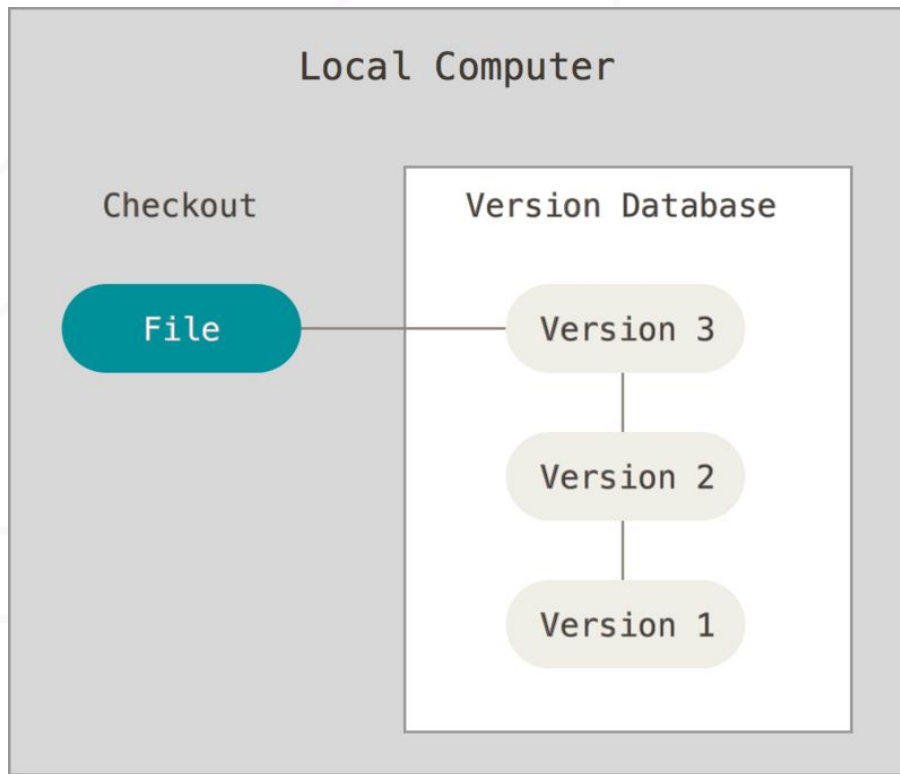
¿Que cambios se hicieron?

¿Quién hizo los cambios?

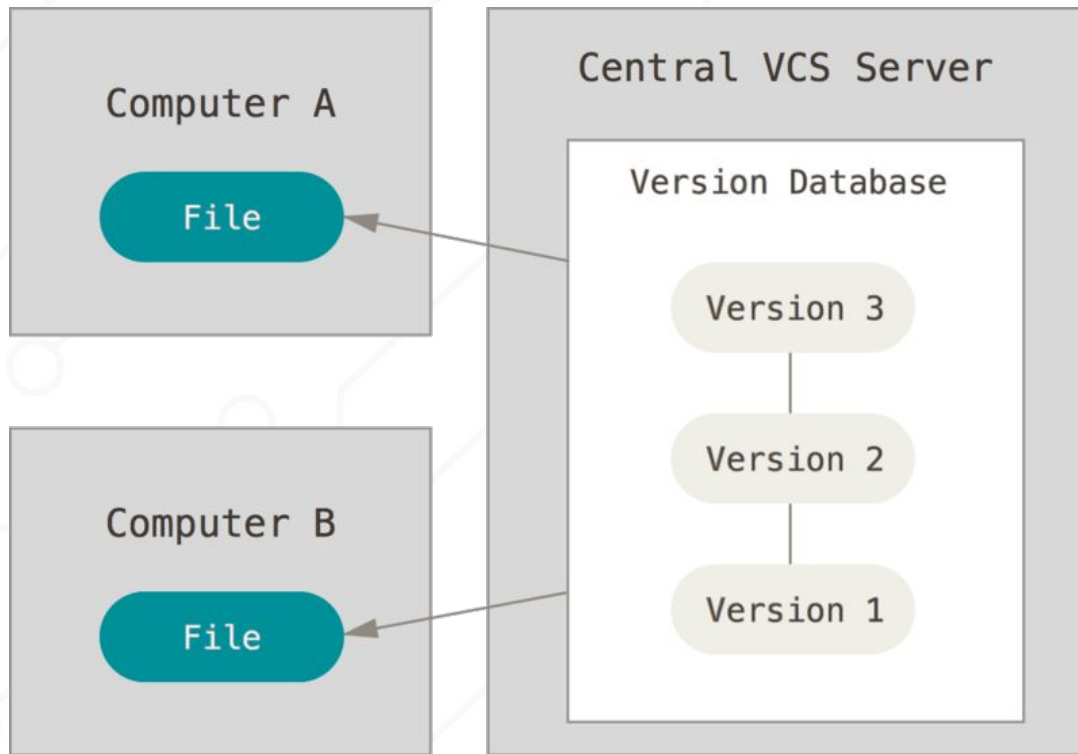
¿Cuando se hicieron los cambios?

¿Por qué fueron requeridos los cambios?

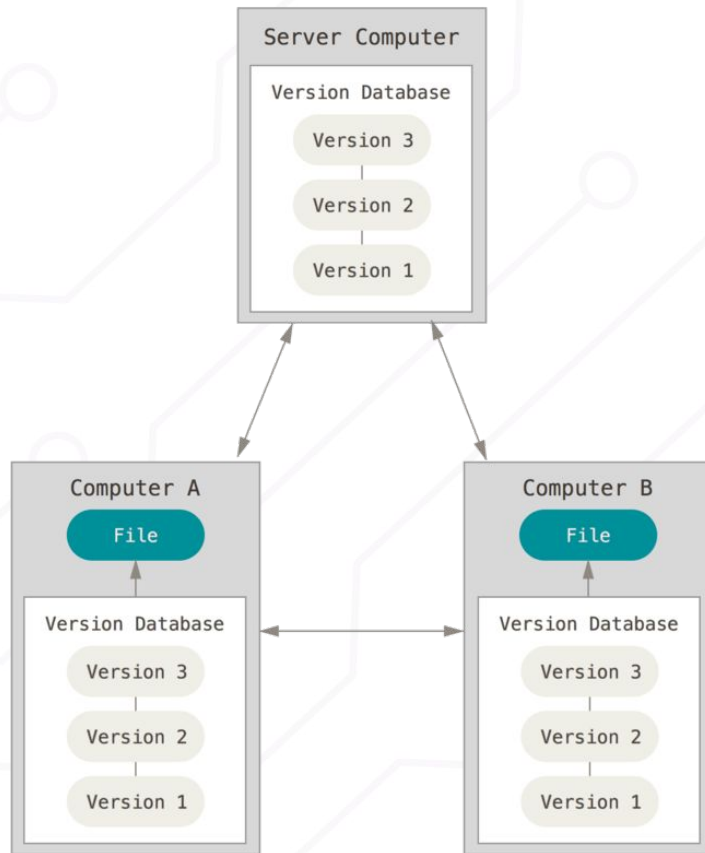
Tipos de VCS (Local)



Tipos de VCS (Centralizado)



Tipos de VCS (Distribuido)





Git es un (VCS) de tipo **distribuido** de código abierto y actualmente el más usado por los desarrolladores gracias a sus beneficios para individuos y equipos de trabajo como:

- Acceso detallado a la historia del proyecto.
- Colaboración en cualquier momento y lugar.

Su uso principal es mediante Interfaz de línea de comandos (**CLI - Command line interface**)

¿Qué es un repositorio?



Un **repositorio** es un espacio de almacenamiento donde se organiza, mantiene y difunde información.

En el caso de **Git** el **repositorio** viene siendo el proyecto donde estará la colección de archivos y carpetas junto al historial de cambios.



Instalación y manejo de Git

DEV.F
DESARROLLAMOS(PERSONAS);

dev

Windows

<https://medium.com/@GalarnykMichael/install-git-on-windows-9acf2a1944f0>

OS

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

Linux

<https://openwebinars.net/blog/como-instalar-git-en-ubuntu/>

Configuración inicial de Git

- Desde consola, se puede acceder a la configuración de **Git** con el comando:

```
git config
```

- Se recomienda establecer una identidad en **Git**, para ello se usan los comandos:

```
git config user.name
```

```
git config user.email
```

- Usando el flag “**--global**” podemos establecer la configuración de forma global y realizarla una sola vez.

```
git config --global user.name “devf”
```



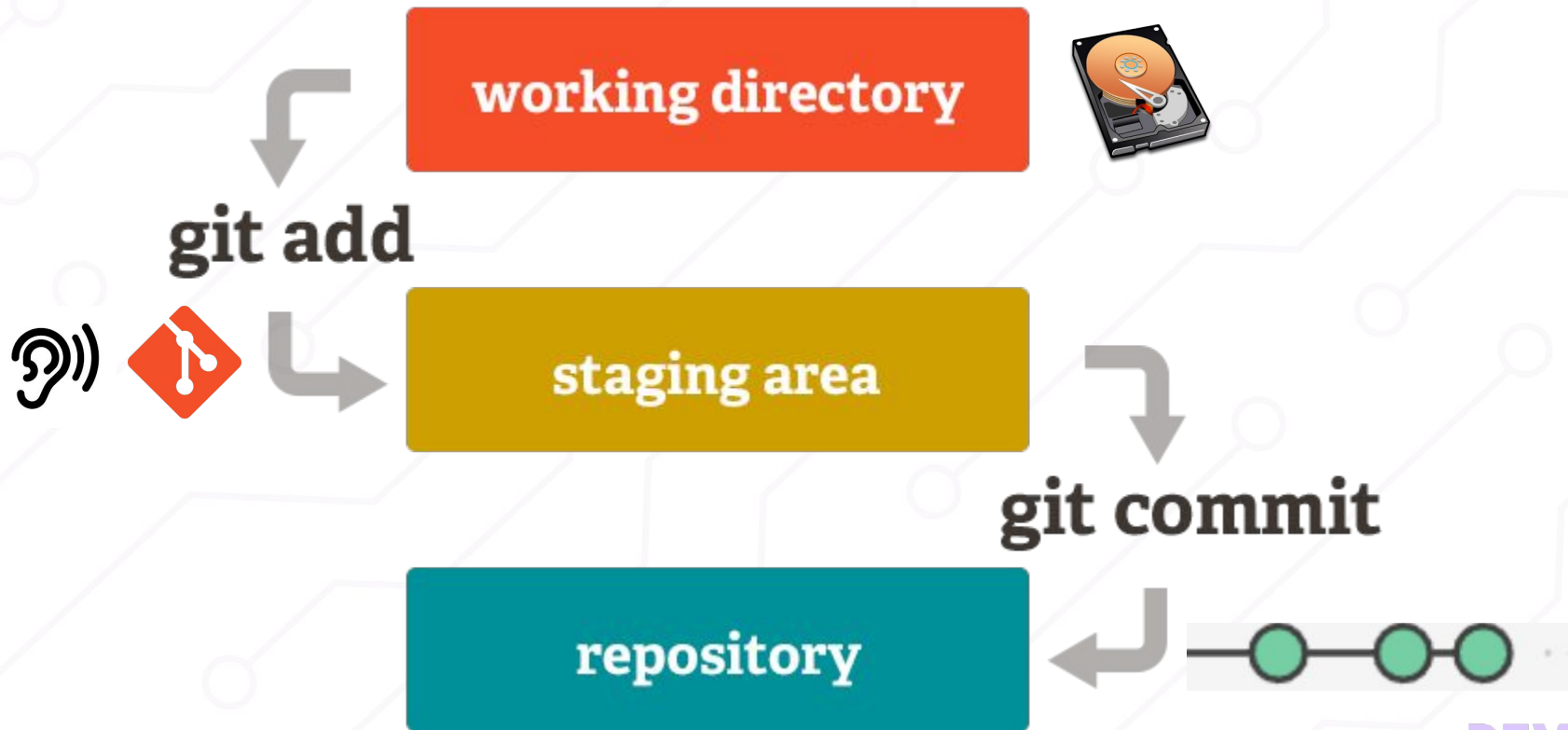
Trabajando con el repositorio local



Repositorio local

- **Working directory:** Nuestro disco duro o sistema de archivos.
- **Staging area:** Lo que está listo para agregarse al historial (Área de indexado de git).
- **Repo local:** Es lo que se encuentra en el historial del commit.

Estados de Git





Comandos básicos parte 1

git config --global user.name -> Poner un nombre de usuario global a la configuración de nuestro git.

git config --global user.email -> Poner un nombre de usuario global a la configuración de nuestro git.

git init -> Inicializamos repositorio.

Comandos básicos parte 1

`git status` -> Nos muestra el estado de working y staging area..

`git add .` -> Agregamos todos los archivos al staging area.

`git add archivo.txt` -> Agregamos el archivo.txt al staging area.

`git commit -m "Comentario"` -> Se crea un punto en la historia con un mensaje.

`git commit -am "Comentario"` -> Agregamos el archivo.txt al staging area.

Volver en el tiempo y uso branches

DEV.F
DESARROLLAMOS(PERSONAS);

dev

Regresar en el tiempo



Regresar en el tiempo

Viajar entre los commit que
hemos realizado.



Ramas (Branch)

Las **ramas (Branch)** son bifurcaciones o variantes de un repositorio, estas pueden contener diferentes archivos y carpetas o tener todo igual excepto por algunas líneas de código.



Comandos básicos parte 2

`git branch` -> Mostrar las ramas que tenemos.

`git branch newBranchName` -> Creamos una nueva rama.

`git checkout nombre` -> Nos cambiamos a la rama nombre.

`git checkout -b nombre` -> Crear y cambiarse a una nueva rama.

`git checkout hash` -> Cambiarnos a un commit en específico.

`git checkout .` -> Regresar al commit más reciente de la rama actual.

`git merge sourceBranch` -> Unimos cambios de una rama.

Trabajando con el repositorio remoto

DEV.F
DESARROLLAMOS(PERSONAS);

dev

Repositorios remotos



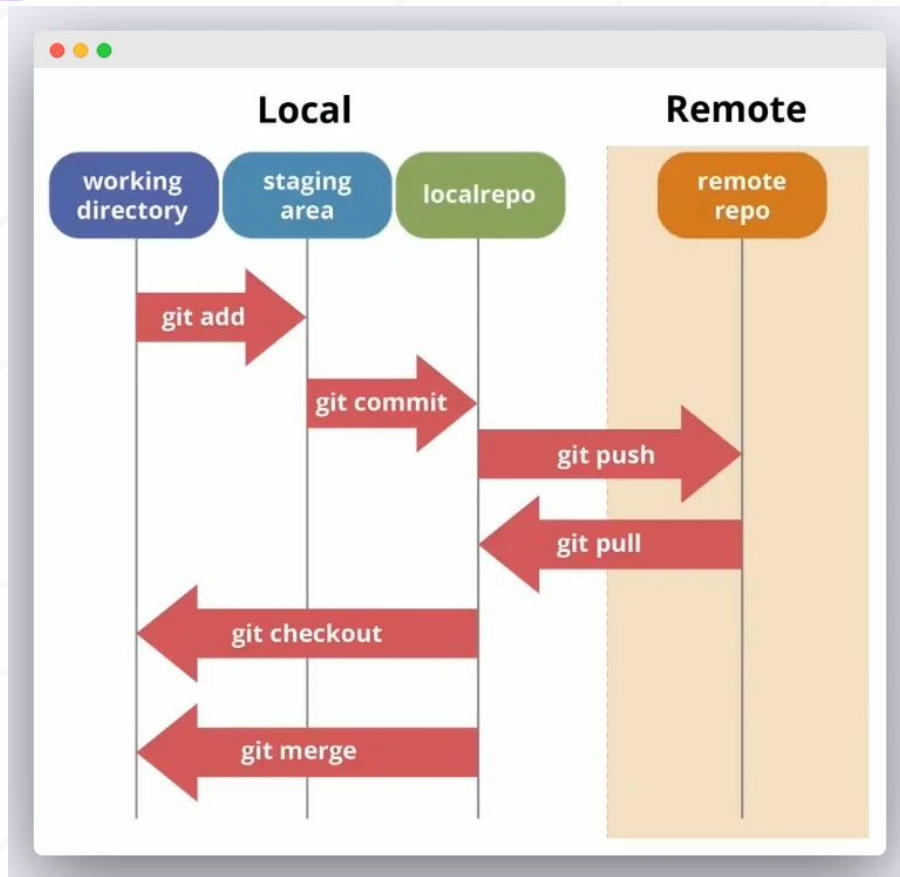
Conectar un repositorio local a uno remoto

Mediante el comando `git remote add origin`.

Diferencias entre el repo remoto y local

- El historial de cambios es gestionado por GIT.
- Los repos remotos son el respaldo de nuestro local.

Flujo con repositorio remoto



Comandos básicos parte 3

`git remote add origin url` -> Agregar la conexión de nuestro repo local al remoto.

`git remote set-url aliasName myNewUrl` -> Agregar la conexión de nuestro repo local al remoto.

`git clone url` -> Clonar repositorio existente.

`git push alias branch` -> Enviamos cambios a repositorio remoto.

`git push --all origin` -> Subir todas las ramas desde local a remoto.

`git pull alias branch` -> Obtenemos cambios más recientes de la rama.

Gitflow

DEV.F
DESARROLLAMOS(PERSONAS);

dev

Flujo merge

1. Cambiarse a la rama de destino.

`git checkout main`

2. Ejecutar el comando merge en la rama destino

`git merge develop`

Flujo pull request

1. Hacer un commit.

```
git add .
```

```
git commit -m "Comentario"
```

2. Enviar al repositorio remoto.

```
git push alias branch
```

3. Crear la pull request en github (rama base y rama destino) y agregar revisores.
4. Los revisores aceptan la PR (Pull request) y se hace el merge.
5. Obtener los cambios mediante git pull origin develop.