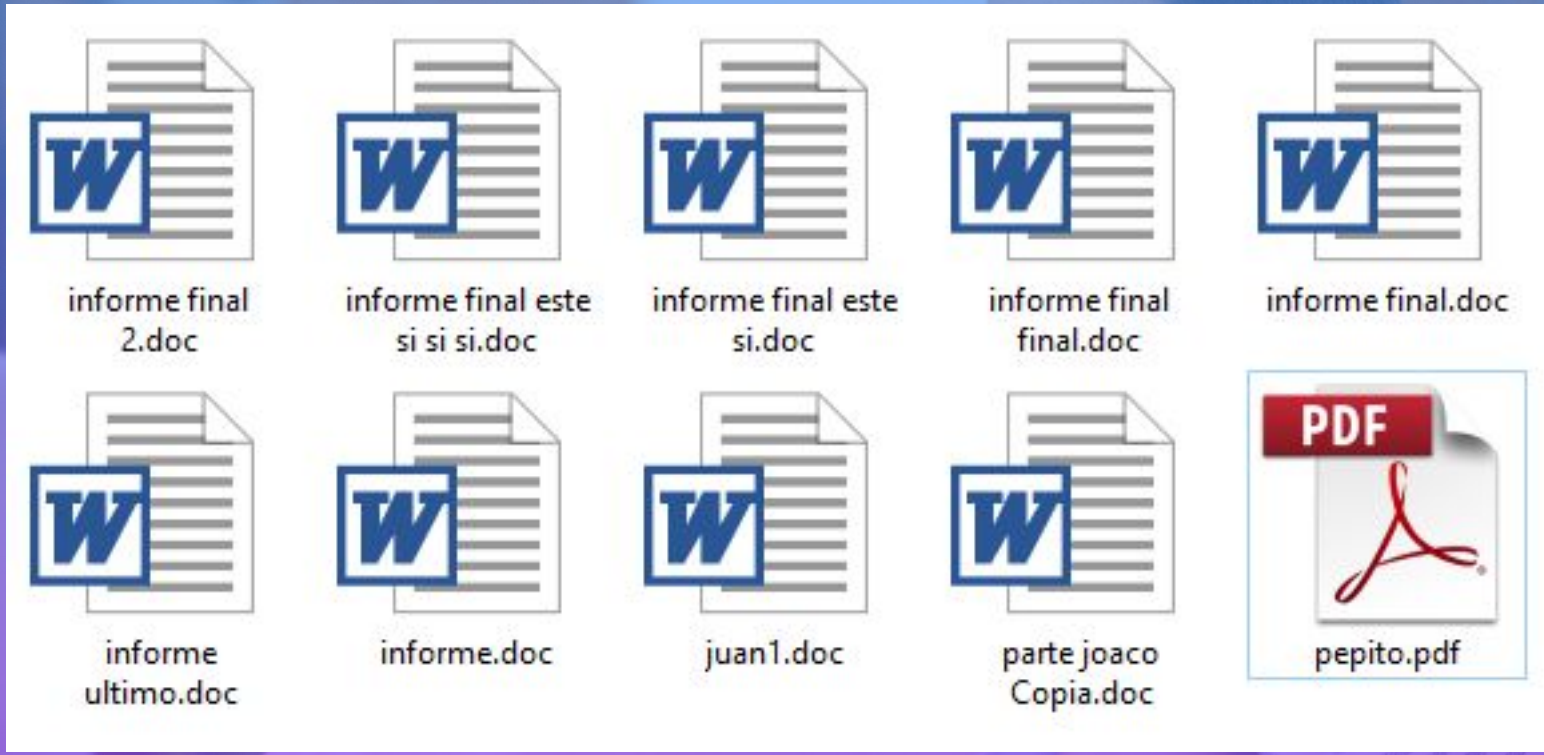
DESARROLLO WEB FULL STACK

Git y GitHub

Un viaje a través de los cambios

Cor-4


informe final 2.doc
informe final este si si si.doc
informe final este si.doc
informe final final.doc
informe final.doc
informe ultimo.doc
informe.doc
juan1.doc
parte joaco Copia.doc
PDF
pepito.pdf

Sistema de Control de Versiones

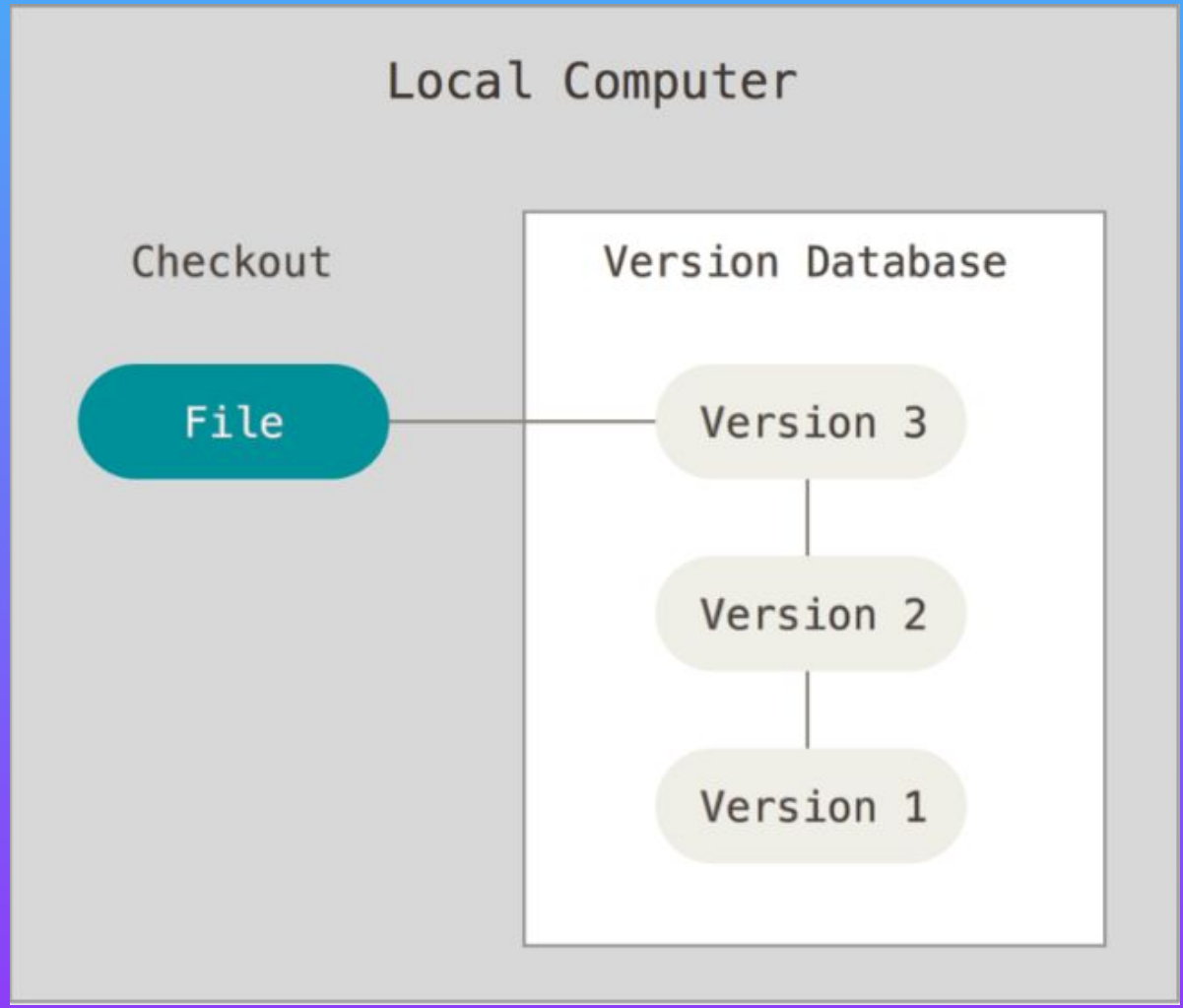
VCS (Sistema de Control de Versiones)

Es un sistema que **registra los cambios** de uno o más archivos a través del tiempo y así poder volver a una versión cuando lo necesitemos.

- Permite revertir cambios en un archivo o un proyecto a un estado anterior.
- Comparar cambios entre versiones.
- Ver quien y que modificó que pueda estar causando problemas.

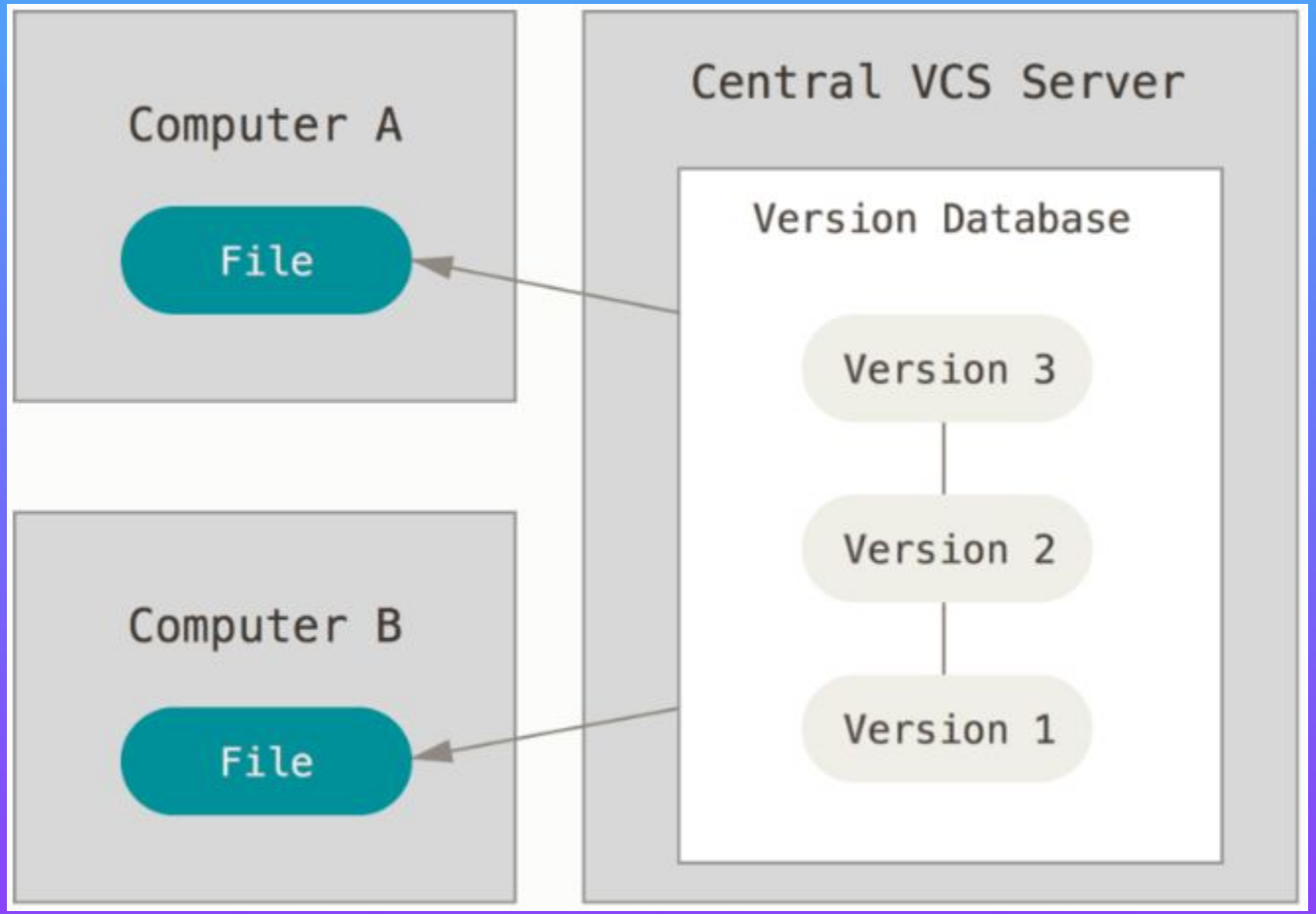
VCS Local

Estos sistemas generan una base de datos local en la que se guardan los cambios realizados al archivo o proyecto.


Local Computer
Checkout
File
Version Database
Version 3
Version 2
Version 1

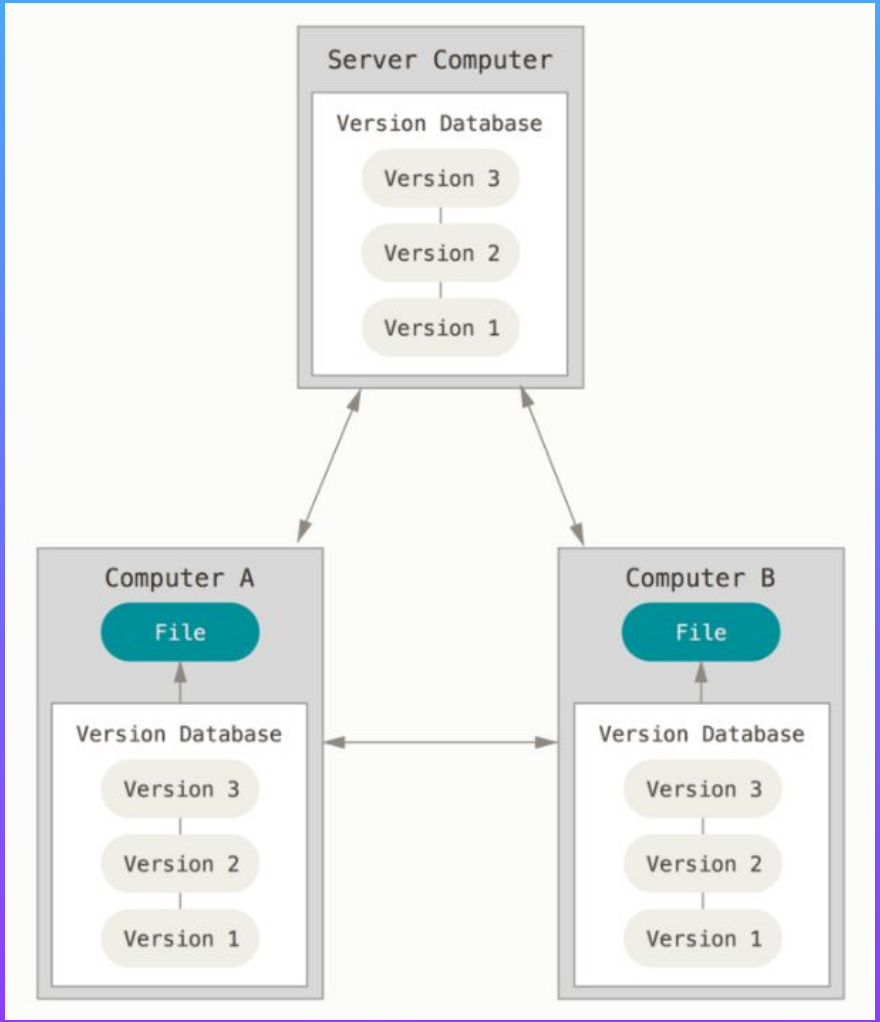
CVCSs (Sistema Centralizado de Control de Versiones)

Es un tipo de CVS donde los cambios registrados son alojados en un servidor central.


Computer A
File
Computer B
File
Central VCS Server
Version Database
Version 3
Version 2
Version 1

DVCS (Sistema de Control de Versiones Distribuidos)

Los repositorios dejan de estar centralizados y pueden ser clonados por los clientes.


Server Computer
Version Database
Version 3
Version 2
Version 1
Computer A
File
Version Database
Version 3
Version 2
Version 1
Computer B
File
Version Database
Version 3
Version 2
Version 1

Git

Git

Es uno de los Sistemas de Control de Versiones Distribuido más utilizados en el mundo.

Características

- Performance
- Seguridad
- Flexibilidad

Basado en snapshots

A diferencia de otros VCS que registran las diferencias de un archivo, Git crea una “foto” de ese archivo registrando así como se ve en ese momento nuestro archivo y guarda esa referencia.

Casi todas las operaciones son locales

En Git la mayoría de las operaciones que podemos realizar, son locales. Al tener una copia del proyecto y de su historial en nuestra computadora, podemos registrar y movernos entre versiones de manera instantánea y sin necesidad de tener conexión a internet.

Los 3 estados

- Modified
- Staged
- Committed

Modified

Este estado indica que existe una modificación pero que aún no ha sido agregada al registro de cambios.

Staged

Indica que el archivo fue marcado en la versión actual para ir a la próxima entrega de cambios.

Committed

Indica que un archivo fue entregado y guardado en nuestro repositorio de cambios.

El flujo de trabajo más básico en Git es:

1. Modificas tus archivos.
2. Seleccionas los cambios a marcar como entrega.
3. Realizas la entrega para guardarlos permanentemente en el repositorio.

Comenzando con Git

Crear un repositorio

Existen dos posibilidades:

- Creamos un repositorio de cero
- Clonamos un repositorio existente

Git init

Comenzando de cero:

Dentro del directorio de nuestro proyecto, **inicializamos** el repositorio.

Esto va a crear una carpeta **.git** donde se encuentra el esqueleto de nuestro repositorio.

```
$ git init
```

Git clone

Git también nos permite clonar un repositorio con una estructura ya definida.

```
$ git clone https://github.com/jorgito/miProyecto
```

Viendo el estado de nuestro proyecto

Git status

Este comando nos permite ver en qué estado se encuentra nuestro proyecto.

- Modificado
- Preparado para entregar(staged)
- Entregado(committed)

```
$ git status
```

Agregando nuestros cambios

Git add

El parámetro **add** nos permite elegir que archivos agregar para ser entregados.

```
$ git add texto.txt

//Para agregar más archivos simplemente los pasamos
//como parametros

$ git add texto.txt texto1.txt

//Si queremos agregar todos los archivos dentro de
//un directorio

$ git add .
```

Entregando nuestros cambios

Git commit

El parámetro **commit** registra nuestros cambios en el repositorio local de manera permanente.

Es común pasar un flag como parámetro para ingresar un mensaje descriptivo de nuestros cambios.

```
$ git commit -m “Se agrego la funcionalidad sumar()”
```

Agregar los cambios al repositorio remoto

Git push origin

Indicando el **repositorio remoto** y la **rama** donde estamos trabajando al comando **git push**, los cambios pasan a subirse al repositorio externo.

```
$ git push origin master
```

Actualizar el repositorio local

Git pull

Cuando se agregan cambios en el repositorio externos, nuestro repositorio local queda desactualizado.
Con el comando **pull** traemos todos los cambios hechos en el repositorio remoto a nuestro repositorio local.

```
$ git pull
```

Branches

Branches

Los ***branches*** son ramificaciones que crean a partir del archivo o proyecto principal, lo que nos permite trabajar sobre el proyecto sin arruinar el proyecto. Se crea una copia del proyecto en la cual se registra como un branch en nuestro repositorio. El branch por defecto es **master.**

Git branch

Cuando creamos un branch nuevo, este apunta al mismo commit donde estamos parados. El comando **branch** genera una “copia” sobre la cual podemos trabajar sin modificar el “original”.

```
$ git branch miBranch
```

Moviendonos entre branches

Muchas veces necesitamos movernos entre diferentes branches, para esto existe un parámetro de Git para poder cambiar a otro branch que existe dentro del repositorio.

```
$ git checkout miBranch
```

Uniendo cambios

Merging

Un merge indica que queremos **“fundir”** los cambios que realizamos en nuestro branch al proyecto original.

Git merge

Si queremos llevar nuestros cambios al branch principal (master) nos “paramos” en el branch moviéndonos con el comando **checkout** y luego realizamos un merge del branch **miBranch.**

```
$ git checkout master

$ git merge miBranch
```

Conflictos

Conflictos

Al *mergear* a veces sucede que los cambios suceden en el mismo archivo y en la misma línea. Esto genera un **conflicto** que debe ser solucionado manualmente.

Git merge

Si queremos llevar nuestros cambios al branch principal (master) nos “paramos” en el branch moviéndonos con el comando **checkout** y luego realizamos un merge del branch **miBranch.**

```
$ git checkout master

$ git merge miBranch
```

Actividad

Clonar un repositorio remoto. Iniciar el proyecto “Acamica & Beer”. Cada equipo deberá subir sus cambios.