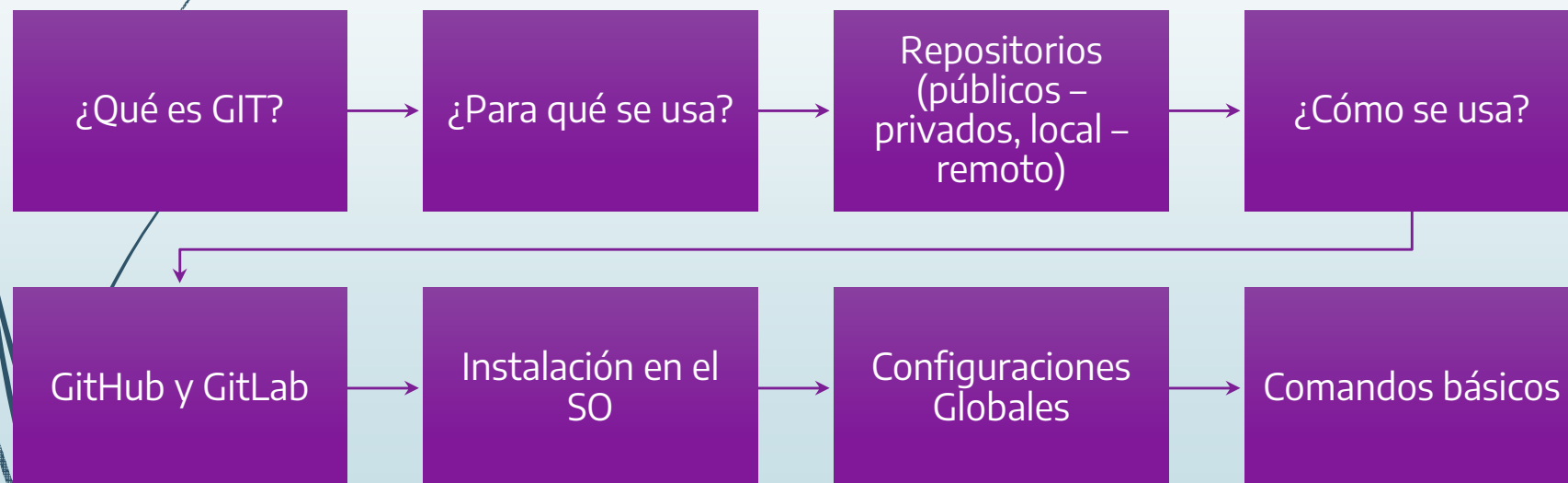


A decorative graphic on the left side of the slide. It features a thick, dark blue vertical bar. To its right, there are several thin, dark blue curved lines that sweep upwards and to the right. A dark grey arrow points horizontally to the right, positioned in the middle of the vertical bar and overlapping the curved lines.

# GIT - Introducción

# Agenda



## ¿Qué es GIT?

Git es un herramienta de ***control de versiones*** (o sistema de versionado).

Una herramienta de control de versiones lleva adelante la gestión de los diversos cambios que se realizan sobre los elementos de algún código fuente.

## ¿Qué es GIT?

Un ***Sistema de Control de Versiones*** es un sistema que registra los cambios realizados en un archivo o conjunto de archivos a lo largo del tiempo, de modo que se puedan recuperar versiones específicas más adelante.

## ¿Para qué se usa?

- Compartir código con otras personas.
- Tener un historial de los cambios realizados en el código.

## ¿Para qué se usa?

Git nos permite:

- Tener un historial de cambios
- Saber quién los hizo y cuándo
- Resolver los conflictos que surjan cuando dos o más personas modifiquen el mismo archivo (merge)



# Repositorios

- Un repositorio es un espacio, en la nube, que tenemos asignado para poder alojar todos los archivos de nuestro proyecto.
- Un repositorio es el lugar donde van a estar todos los commits que forman parte del historial del proyecto.

## Repositorios locales y remotos

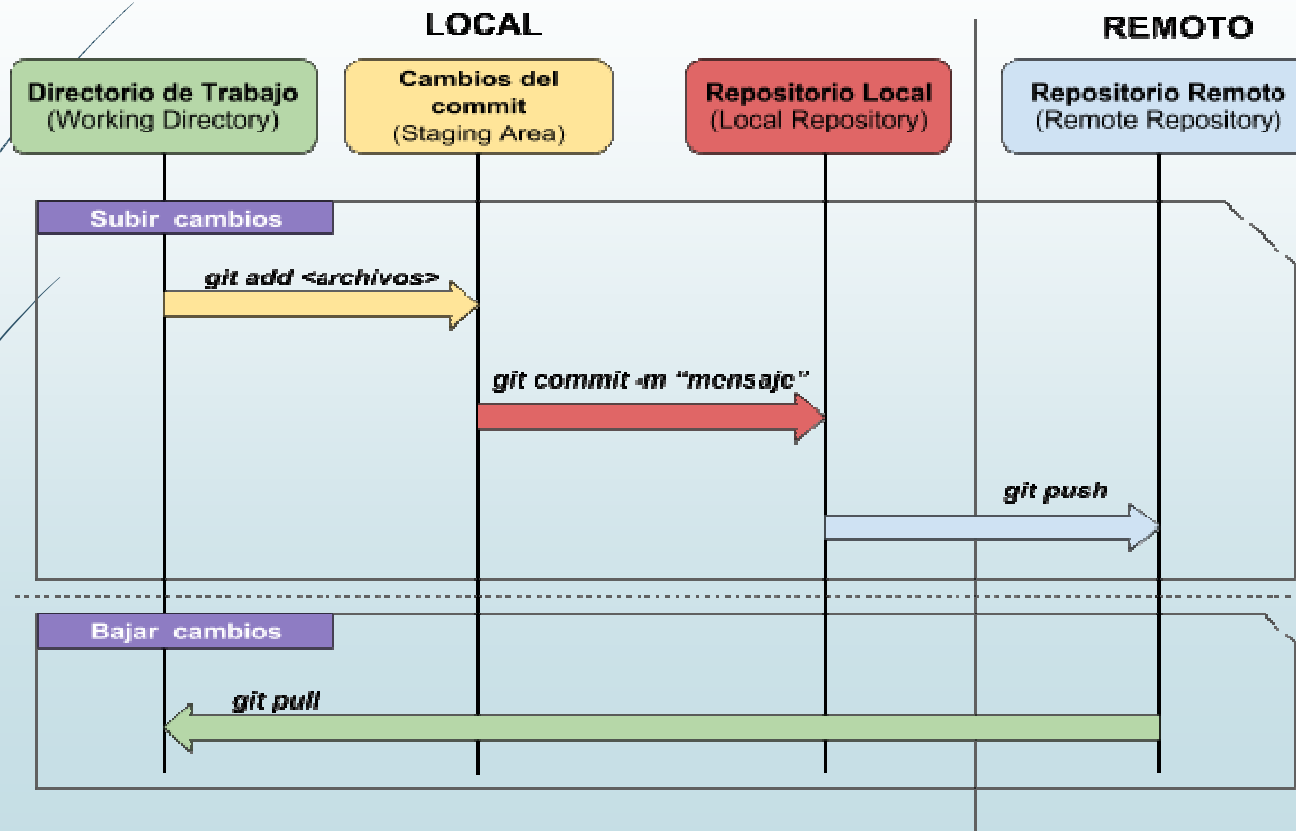
- GIT trabaja con un repositorio local que está en nuestro equipo, donde iremos agregando nuestros commits.
- También trabaja con uno remoto en el cual podemos subir nuestros commits o del cual podemos bajarnos los commits que haya subido alguien.



## ¿Cómo se usa? (Alto nivel)

1. Creamos/borramos/modificamos archivos en una carpeta asociada a un repositorio (localmente, en nuestro S.O.)
2. Seleccionamos los archivos que van a ser parte de un commit.
3. Confirmamos el commit para agregarlo al repositorio.
4. Subimos los commits de nuestro repositorio local al repositorio remoto.

## ¿Cómo se usa? (Alto nivel)



# GITHUB y GITLAB

Existen diferentes plataformas web que implementan y brindan un Sistema de Versionado de Archivos basados en GIT.

Entre ellos se encuentran:

- GitHub
- GitLab
- Bitbucket
- Gogs

## GITHUB y GITLAB

- También existe la posibilidad de instalar nuestro propio servidor de GIT, con nuestra propia interfaz gráfica personalizada.
- Muchas empresas realizan y usan sus propias instalaciones de plataformas basadas en GIT, para que el código alojado esté 100% en su poder y no “en manos de otros”.

# GITHUB y GITLAB

- GitHub y GitLab son dos de las plataformas más utilizadas, junto a Bitbucket.
- Tanto GitHub como GitLab ofrecen repositorios públicos y privados gratuitos.
- Ambas plataformas cuentan con funcionalidades exclusivas por las cuales hay que pagar. Es por ello que ofrecen tres posibles suscripciones: free, team (GitHub)/premium (GitLab) y Enterprise (GitHub)/Ultimate (GitLab)



## GITHUB y GITLAB

Lo cierto es que para trabajar en un proyecto “chico/mediano”, cualquiera de las dos opciones se acomoda a las necesidades.

## Instalación en el sistema operativo

- Para poder utilizar GIT en nuestro equipo necesitamos tener instalado un cliente de dicho sistema.
- Tanto GitHub como GitLab ofrecen repositorios públicos y privados gratuitos.



## Instalación en el sistema operativo

- “GIT” es un cliente de GIT que nos permite acceder desde la línea de comandos de cualquier terminal a las funcionalidades brindadas por dicho sistema de versionado.
- También instala su propia terminal llamada Git Bash.
- Además, trae un GUI Client muy sencillo y básico.



## Instalación en el sistema operativo

El link a su sitio de descargas es:

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



## Instalación en el sistema operativo

- Un GUI Client recomendado por sencillez y facilidad de uso es **GitHub Desktop**.
- GitHub Desktop tiene soporte para Windows y macOS.

## Instalación en el sistema operativo

El link a su sitio de descargas es:

**<https://desktop.github.com/>**

# Configuraciones globales

¿Luego de haber instalado algún cliente de GIT en nuestro equipo, debemos realizar mínimamente las siguientes dos configuraciones:

- `git config --global user.name "TU NOMBRE"`
- `git config --global user.email "TU DIRECCION DE EMAIL"`

# Comandos básicos

## GIT clone

```
git config --global user.name "TU NOMBRE"
```

- Permite clonarnos un repositorio remoto.
- En el caso de que el repositorio sea privado, debemos tener el correspondiente acceso y permiso para descargarlo (debemos tener el rol de “propietario” o formar parte de los “colaboradores”).

# GIT status

## `git status`

- Debe ejecutarse en una terminal situados en la raíz de un repositorio clonado (o descargado).
- Permite saber si se realizaron cambios sobre archivos que aún no fueron commiteados.

## GIT add

```
git add <filename>
```

```
git add .
```

- Debe ejecutarse en una terminal situados en la raíz de un repositorio clonado (o descargado).
- Permite agregar uno o varios archivos al área de staging (repositorio local), para que luego éstos sean commiteados (en un mismo commit) y subidos al repositorio remoto.



## GIT commit

```
git commit -m 'mensaje'
```

- Debe ejecutarse en una terminal situados en la raíz de un repositorio clonado (o descargado).
- Permite armar y guardar un commit con los archivos que se encuentran en el área de staging (los que previamente fueron agregados con git add).
- Debe considerarse que un commit es un “punto de cambio del proyecto”, situado cronológicamente.

## GIT log

```
git log --online
```

- Debe ejecutarse en una terminal situados en la raíz de un repositorio clonado (o descargado).
- Muestra las líneas los commit realizados con los comentarios que hubiéramos realizado cada cual con su identificador.

## GIT reset

```
git reset --hard identificdor
```

- Debe ejecutarse en una terminal situados en la raíz de un repositorio clonado (o descargado).
- Retrocede al commit especificado en el identificador referenciado, eliminando todos los cambios futuros que se hubieren realizado.



# **Comandos**

## **Repositorio remoto**

# GitHub

- En la página github.com
  - ✓ Crea un usuario
  - ✓ Crear un nuevo repositorio (nombreMiProyecto)
  - ✓ Definir si es público o privado
- En la terminal de nuestra consola
  - ✓ Transferir a nuestro archivos a la nube escribiendo el en la terminal el siguiente código:  
`Git remote add origin https://github.com/bluuweb/nombreMiProyecto.git`

## GIT push – subida a repositorio remoto

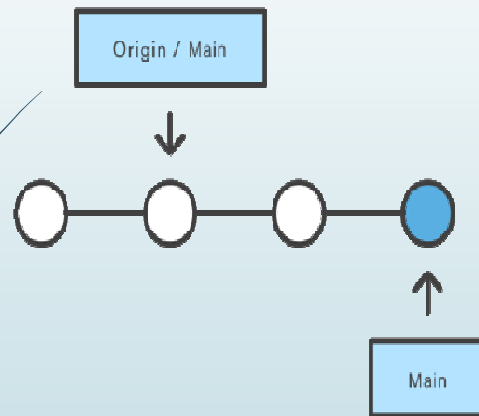
```
git push <remote> <branch>
```

- Debe ejecutarse en una terminal situados en la raíz de un repositorio clonado (o descargado).
- Permite subir los commits realizados en nuestro repositorio local al repositorio remoto para que éstos puedan ser descargados por el resto del equipo de trabajo.

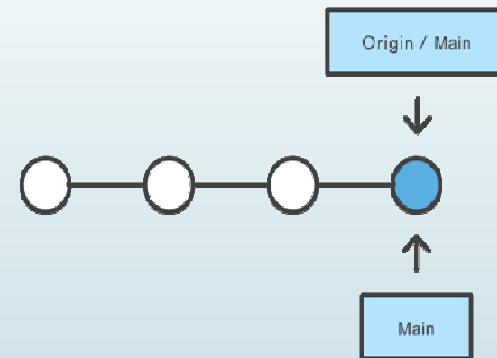
Git push -u origin master

# Git push

Before Pushing



After Pushing



## GIT pull – descarga a repositorio local

`git pull`

- Debe ejecutarse en una terminal situados en la raíz de un repositorio clonado (o descargado).
- Permite descargar los cambios (commits) desde el repositorio remoto y actualizar al instante el repositorio local para reflejar ese contenido.



## GIT tags – versiones (release)

```
git tag nombreVersion -m "descripción"
```

- Crea un tag que contiene un nombre de versión

```
git log --online
```

- Despliega el nombre del tag: versión

```
git push --tags
```

- Subir los tags locales al repositorio remoto

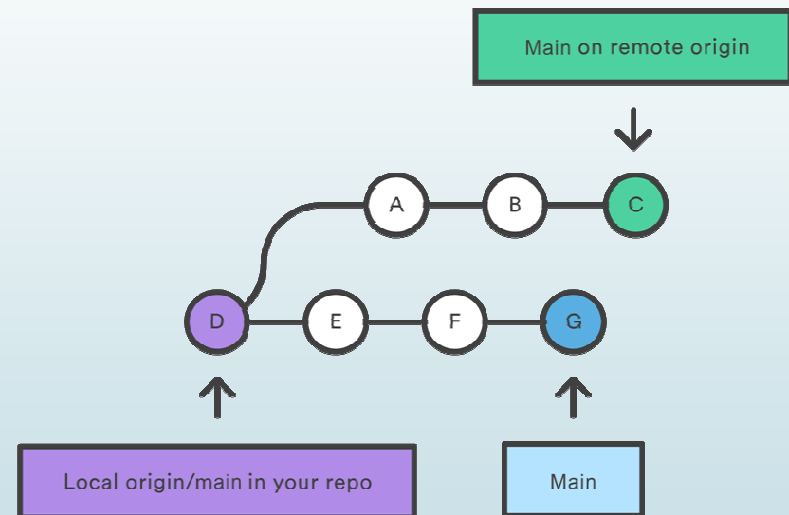
## GIT pull

En este caso, ***git pull*** descargará todos los cambios desde el punto de separación de la rama local y la rama principal.

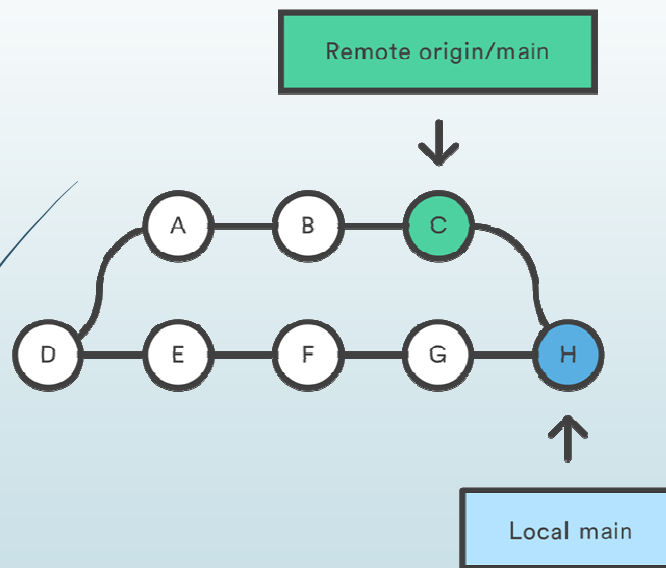
En el ejemplo, ese punto es **E**.

El comando ***git pull*** recuperará las confirmaciones remotas divergentes, que son **A**, **B** y **C**.

A continuación, el proceso de incorporación de cambios creará otra confirmación de fusión local que incluya el contenido de las nuevas confirmaciones remotas divergentes.



## GIT pull



En este diagrama, podemos ver la nueva confirmación **H**, que es una confirmación de fusión nueva que incluye el contenido de las confirmaciones remotas **A**, **B** y **C**, y tiene un mensaje de registro combinado.

## GIT branch - ramas

```
git log --online
```

- **Despliega** el nombre de la rama, por ejemplo master

```
Git branch nuevaRama
```

- **Crea** una rama

```
Git checkout nuevaRama
```

- Nos **posiciona** en nuevaRama

```
Git branch
```

- **Muestra** cuantas ramas hay y en cual estamos posicionados

## Git merge – unión de ramas

Paso 1. `Git checkout master`

- Debemos **posicionarnos** en el master

Paso 2. `Git merge nuevaRama`

- **Une** la rama NombreRama al master

Paso 3. `Git branch -d nuevaRama`

- **Eliminamos** la rama creada nuevaRama dado que ya no la necesitamos al haber sido unificada a la rama master.

# GIT - Conflictos

Un conflicto se presenta cuando se modifica el mismo objeto en dos ramas diferentes.

El comando **Git merge** es el encargado de detectar el conflicto y desplegarlo dando las diferentes opciones de solución .

- Aceptar cambio actual
- Aceptar cambio entrante
- Aceptar ambos cambios
- Comparar cambios

## GIT – settings - pages

Se introduce el concepto de REPOSITORIO que contiene un conjunto de archivos de diferentes tipos.

Es conveniente crear una carpeta dentro del repositorio DOCS que contenga el archivo index.html

**Git** **pages** nos permite crear un dominio gratis por cada uno de nuestros repositorios (proyecto) utilizando el index.html contenido en la carpeta DOCS

# GIT – Repositorio - PAGES

Paso 1. Crear repositorio en github,

- El nombre debe ser el nombreDeUsuario.github

Paso 2. Subir los archivos

Git init

Git add .

Git commit -m "creando el "index"

Git remote add origin <http://github.com/bluuweb/nombreDeUsuariogithub.io.git>

Paso 3. Setting / Github pages

- Despliega el hosting que ya esta habilitado para su uso



A decorative graphic on the left side of the slide. It features a thick, dark blue vertical bar. To its right, there are several thin, dark blue curved lines that sweep upwards and to the right. A dark brown, arrow-shaped element points to the right, partially overlapping the vertical bar and the curved lines.

Gracias!