




EXPLICACION DE GIT

- Git SCM

Git es un **software de control de versiones desantralizado** diseñado por Linus Torvalds.

O sea, un software que sirve básicamente para gestionar las versiones por las que va pasando un archivo o un conjunto de archivos. De esta forma a futuro se pueden recuperar versiones específicas y posibilita el trabajo colaborativo a travez de plataformas publicas.

PLATAFORMAS

Las plataformas más utilizadas actualmente son *Github* , *GitLab*  y *Bitbucket* . En este seminario elegimos **GitLab** por su licencia de código abierto que permite hostear una instancia en un servidor propio, como es el caso de gitlab.catedras.info.unlp.edu.ar

Podemos instalar Git accediendo a este [link](#)

En caso de tener problemas con la instalación, se puede recurrir a la [documentación oficial](#)

Configurar el nombre y mail que sale en los commits

```
git config --global user.name <nombre-de-usuario>
git config --local user.email <email-de-la-cuenta>

# con --global se usa en todos los repositorio del sistema
# con --local solo se setea en el repositorio en que se ejecuta
```

Es importante que el email sea el mismo que el que usamos en la plataforma remota.

Hay 2 posibles puntos de partida a la hora de comenzar a trabajar con un repositorio:

1. Clonándolo desde un repositorio remoto (git clone).

Es el caso de la cursada, ya que el repositorio existe y esta inicializado

2. Inicializándolo desde una carpeta local (git init).

Esto seria para crear un repositorio de git local, para poder tenerlo en una plataforma hay que configurar el remote del repositorio

Al clonar un repositorio remoto, creamos una copia local que estará vinculada al mismo. Existen 2 formas de clonar el repositorio

HTTPS (no lo usamos)

```
git clone https://gitlab.catedras.linti.unlp.edu.ar/js/grupos-2023/grupo-<xx>.git
# este metodo pide usuario y contraseña cada vez que se realiza una operacion el repositorio remoto
# esto se puede evitar diciendo que nos guarde las credenciales
git config --local credential.helper store
```

SSH (si lo usamos, más seguro)

```
git clone git@gitlab.catedras.linti.unlp.edu.ar:js/grupos-2023/grupo-<xx>.git

# para usar SSH deben generar un par de claves https://gitlab.catedras.linti.unlp.edu.ar/help/user/ssh.md
# pero el comando recomendado es
ssh-keygen -o -t rsa -b 4096 -C "mail.en@gitlab.catedras"
# Generating public/private rsa key pair.
# Enter file in which to save the key (/home/msosa/.ssh/id_rsa): /home/msosa/.ssh/gitlab-catedras #donde y con que nombre guarda
# Enter passphrase (empty for no passphrase): #aca puse enter si pones una clave te va a pedir cada vez que la uses
# Enter same passphrase again: #repetir el enter o la misma password anterior
# Your identification has been saved in /home/msosa/.ssh/gitlab-catedras --> parte privada, nunca se comparte
# Your public key has been saved in /home/msosa/.ssh/gitlab-catedras.pub --> parte publica, es la que se comparte
# The key fingerprint is: ****

cat /home/msosa/.ssh/gitlab-catedras.pub # esto imprime el contenido de la llave publica,
```


Iniciamos GIT en la carpeta donde esta el proyecto

```
git init
```

Luego creamos un repositorio remoto en alguna plataforma. Una vez creado el repositorio remoto, debemos pararnos en la raíz de nuestro repositorio local y vincular ambos:

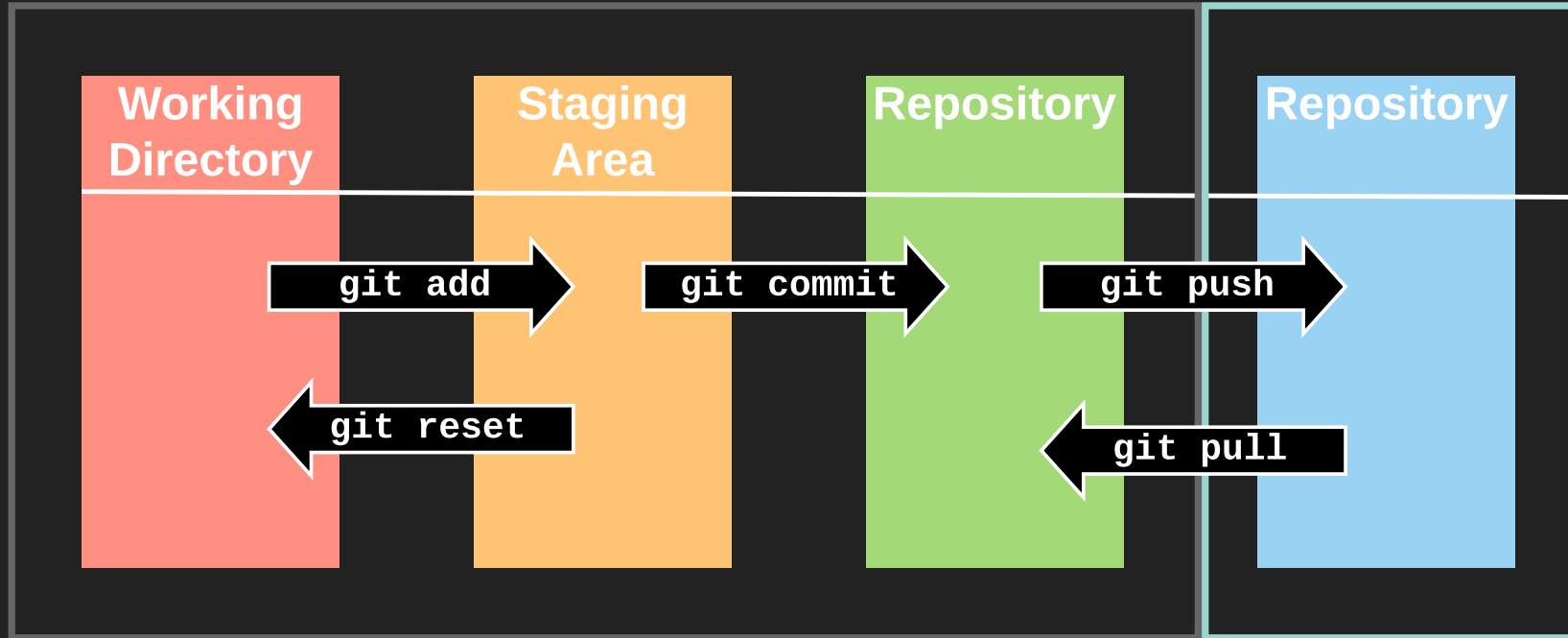
```
git remote add origin <url-del-repositorio>
```

tener en cuenta que en el primer push debemos agregar la opción "--set-upstream" que vinculará nuestras ramas main.

```
git push --set-upstream <url-del-repositorio> main
```

LOCAL

REMOTE



GIT

El comando `git add` añade un cambio del directorio de trabajo en el entorno de stage. Indica a Git que quieres incluir actualizaciones en uno o varios archivos en el próximo commit, pero este trabajo aún sigue en tu máquina. Puedes añadir tus archivos uno por uno, todos a la vez e incluso especificar reglas:

```
# Añadir todos los archivos
git add .

# Añadir un archivo concreto
git add [filename]

# Añadir todos los archivos omitiendo los nuevos
git add --all

# Añadir todos los archivos con una extensión específica
git add *.txt

# Añadir todos los archivos dentro de un directorio
git add docs/

# Añadir todos los archivos dentro de un directorio y sus subdirectorios
git add -r docs/
```

El comando `git commit` captura una instantánea de los cambios preparados en ese momento del proyecto (los que hayamos añadido con `git add` anteriormente). Las instantáneas confirmadas pueden considerarse como versiones «seguras» de un proyecto: Git no las cambiará nunca a no ser que se lo pidas expresamente.

Los commits son puntos de control a través de los cuales podemos viajar para ir a versiones anteriores en nuestro desarrollo. Lo que se comitea aún no está subido al repositorio remoto. Su sintaxis es:

```
git commit -m "Descriptive text for this commit"

# Podemos añadir los ficheros modificados y hacer el commit en un único paso del siguiente modo:
git commit -am "Descriptive text for this commit"

# Podemos sobrescribir el commit más reciente (sólo recomendable si el último commit no se ha subido al repositorio remoto).

git commit --amend
```

El comando git pull se emplea para extraer y descargar contenido desde un repositorio remoto y actualizar al instante el repositorio local para reflejar ese contenido.

```
git pull
```

El comando git pull es, en realidad, una combinación de dos comandos, git fetch seguido de git merge. En la primera etapa de la operación git pull ejecutará un git fetch en la rama local a la que apunta HEAD. Una vez descargado el contenido, git pull entrará en un flujo de trabajo de fusión. Se creará una nueva confirmación de fusión y se actualizará HEAD para que apunte a la nueva confirmación.

Este comando se utiliza para subir nuestros commits al repositorio remoto y compartirlos con el resto del equipo. Su sintaxis es:

```
# Push de tus commits la primera vez, si quieres enviar tu rama al repositorio remoto
git push -u origin [branch_name]

# Push "normal"
git push [remote] [branch_name]
```

Los argumentos `remote` y `branch_name` son opcionales, git ya tiene la información del remoto y de la rama actual. Sin embargo son obligatorios la primera vez (además habrá que utilizar el flag `-u`) que vayas a realizar un push del repositorio, o de una rama que hayas creado en local, de este modo creará la misma rama en el repositorio remoto.

El comando `git status` muestra el estado del directorio de trabajo y del área del entorno de ensayo. Permite ver los cambios que se han preparado, los que no y los archivos en los que Git no va a realizar el seguimiento.

```
git status
```

El resultado del estado no muestra ninguna información relativa al historial del proyecto.

Para acceder al histórico de commits utilizaremos el comando git log.

```
git log

# Mostrar información en una sola línea
git log --oneline

# Ver un número limitado de commits
git log -[n]

# Ver información extendida del commit
git log -p

# Ver información de los logs a color y en una sola línea
git log --all --decorate --oneline --graph
```

El listado de commits aparece invertido, es decir, los últimos realizados aparecen los primeros. De un commit podemos ver diversas informaciones básicas como: Identificador del commit, Autor, Fecha de realización, Mensaje escrito