



# Git/github

Sistema de Control de Versiones  
Red Social de Código



# 1 Configuración de Git

- Descargar: <http://git-scm.com/download/>
- Registrarnos: [www.github.com](http://www.github.com)
- Crear repositorio “nombrerepo”

Ejecutamos en la linea de comando lo siguiente:

```
>> git config --global user.name "John Doe"
```

```
>> git config --global user.email johndoe@example.com
```

```
>> git config --list
```



# Clonar Repositorio Remoto Git

Si creamos un repositorio remoto con github podemos descargarlo(clonarlo) en nuestro equipo como un repositorio local el cual podemos editar y llenar de información, después podemos hacerle un **push** del repositorio local al remoto.

```
>> git clone git://github.com/user/nombrerepo.git
```

```
>> git clone https://github.com/user/nombrerepo.git
```



# 1.1 ¿Quieres aprender Git?

Git permite a grupos de personas trabajar en los mismos documentos (con frecuencia código fuente), al mismo tiempo. Es un sistema de control de versiones distribuido.

Nuestra línea de comandos se encuentra actualmente en un directorio llamado **nombrerepo**.

Para inicializar un repositorio **Git** aquí, escriba el siguiente comando:

```
>> git init
```



## 1.2 Comprobar el estado

Nuestro directorio `nombrerepo` ahora tiene un repositorio vacío en `/.git/`. El repositorio es un directorio oculto donde Git funciona.

El siguiente paso, vamos a escribir el comando `git status` para ver cuál es el estado actual de nuestro proyecto:

```
>> git status
```



## 1.3 Añadir (Add) y Cometer(commit)

He creado un archivo llamado `leer.txt` en el repositorio `nombrerepo`.

Debe ejecutar el comando `git status` de nuevo para ver cómo el estado del repositorio ha cambiado:

```
>> git status
```



## 1.4 Adicionar Cambios

Observe cómo **Git** dice **leer.txt** "Esta sin seguimiento" Esto significa que **Git** ve que **leer.txt** es un nuevo archivo.

Para decirle a Git que inicie el seguimiento de los cambios realizados en **leer.txt**, primero tenemos que agregarlo a la zona de ensayo mediante **git add**.

```
>> git add leer.txt
```



# 1.5 Comprobar Cambios

Git ahora está rastreando nuestro archivo `leer.txt`.

Vamos a ejecutar `git status` de nuevo para ver los cambios:

```
>> git status
```



## 1.6 (Committing)

Los archivos que aparecen aquí están en el área de ensayo, y no se encuentran en el repositorio todavía.

Podríamos añadir o eliminar archivos antes de guardarlos en el repositorio.

Para almacenar los cambios por etapas corremos el comando `commit` con un mensaje que describe lo que hemos cambiado. Vamos a hacerlo ahora, escribiendo:

```
>> git commit -m "Agregar archivo leer.txt"
```



## 1.7 Adicionando todos los cambios

También puede utilizar comodines si desea añadir varios archivos del mismo tipo. Tenga en cuenta que se ha añadido un montón de archivos **.txt** en el directorio.

Hay algunos archivos en el directorio **css/** y algunos otros terminaron en la raíz de nuestra **nombrerepo**. Por suerte, podemos añadir todos los archivos nuevos usando un comodín con **git add**.

```
>> git add .
```



## 1.8 (Commiting) todos los cambios

Usted ha añadido todos los archivos de texto en el área de preparación. Ejecute `git status` para ver lo que está a punto de cometer(`commit`).

Ejecutar:

```
>> git commit -m 'Adicionar todos los archivos txt'
```



# 1.9 Historial

Por suerte para nosotros, Existe un registro de **Git**. El registro de **Git** es como un diario que recuerda todos los cambios que hemos realizado hasta ahora, en el orden en que se hicieron los **commits**. Trate de ejecutar ahora:

```
>> git log
```



# 1.10 Repositorios Remotos

Hemos creado un nuevo repositorio GitHub vacío para su uso con **nombrerepo**. Puedes comprobar que funciona, pero te darás cuenta de que nada está allí todavía. Para empujar (**push**) nuestro repositorio local al servidor GitHub tendremos que añadir un repositorio remoto.

Este comando tiene un nombre y una dirección URL remota del repositorio, que en su caso es `git@github.com:user/nombrerepo.git`.

Vamos a ejecutar `git add remote` con las siguientes opciones:

```
>> git remote add origin git@github.com:user/nombrerepo.git
```



## 1.11 Empujando(pushing) remotamente

El comando `push` le dice a `Git` dónde poner nuestros `commits` cuando estemos listos. Así que vamos a empujar nuestros cambios locales en nuestro repositorio `origin` (en `GitHub`).

El nombre de la rama remota es `origin` y el nombre predeterminado de la rama local es `master`. Con `-u` le dice a `Git` que recuerde los parámetros, por lo que la próxima vez simplemente puede ejecutar `git push` y `Git` sabrá qué hacer.

```
>> git push -u origin master
```



# 1.12 Halando(pulling) remotamente

Ahora vaya a **GitHub** y eche un vistazo a su repositorio. Usted debe notar que está lleno de archivos.

Vamos a suponer que ha pasado algún tiempo. Hemos invitado a otras personas a nuestro proyecto de **github** y han empujado (**push**) algunos cambios.

Podemos ver si hay cambios en nuestro repositorio de GitHub y jalar (**pull**) hacia abajo los nuevos cambios ejecutando:

```
>> git pull origin master
```



# Deshacer git add/commit/push

Ejecutamos en la linea de comando lo siguiente:

Eliminar git add:

```
>> git reset
```

```
>> git reset .
```

Eliminar git commit:

```
>> git reset --hard
```

Eliminar git push:

```
git push --delete origin nombre_rama
```

Revertir git push:

```
git revert {commit_id}
```



# Administrar Ramas (branches)

Ejecutamos en la linea de comando lo siguiente:

Crear nueva rama:

```
>> git checkout -b nombre_rama
```

Cambiar de rama en nuestro repositorio:

```
>> git checkout nombre_rama
```

Realizar push a la rama en github:

```
>> git push origin nombre_rama
```

Ver todas las ramas:

```
>> git branch
```

Eliminar rama (local):

```
>> git branch -d nombre_rama
```

Eliminar rama (github):

```
>> git push origin :nombre_rama
```



# Configuración de Proxy

Ejecutamos en la línea de comando lo siguiente:

Adicionar Proxy:

```
>> git config --global http.proxy http://proxy2.sena.edu.co:80
```

Quitar Proxy:

```
>> git config --global --unset http.proxy
```

Verificar Configuración:

```
>> git config --list
```



# Pasos en Clase

```
>> git --version
```

```
>> cd Desktop/
```

```
>> git config user.name "user"
```

```
>> git config user.email "correo@gmail.com"
```

```
>> git config --list
```

```
>> git clone https://github.com/user/nombrerepo.git
```

```
>> cd nombrerepo/
```

```
>> git status
```

```
>> git add .
```

```
>> git commit -m "Mensaje"
```

```
>> git push origin master
```



# Pasos con Proxy

```
>> cd Desktop/  
>> git config --global http.proxy http://proxy2.sena.edu.co:80  
>> git clone https://github.com/user/nombrerepo.git  
>> cd nombrerepo/  
>> git add .  
>> git status  
>> git commit -m "Mensaje"  
>> git remote set-url origin  
>> git push origin master
```



# Pasos sin Proxy

```
>> cd Desktop/  
>> git config --global --unset http.proxy  
>> git clone https://github.com/user/nombrerepo.git  
>> cd nombrerepo/  
>> git add .  
>> git status  
>> git commit -m "Mensaje"  
>> git remote set-url origin  
>> git push -u origin master
```