FACULTAD
DE INGENIERIA
Universidad de Buenos Aires

Introducción al uso de Git

Guía rápida para los conceptos básicos de Git V1.0

Agenda

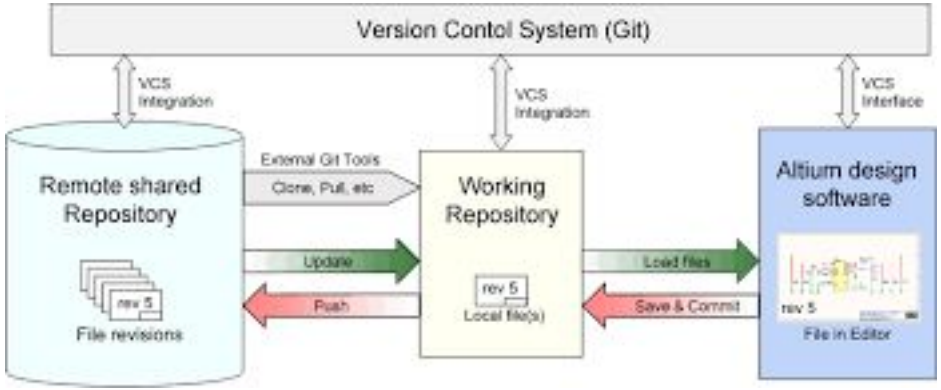
- Qué es Git?
- Qué es Github?
- Qué es un versionador?
- Ventajas de utilizar un versionador
- Ventajas de Git sobre otros versionadores
- Comandos Básicos
  - Configuración general de git (*config*)
  - Crear un repositorio (*init*)
  - Descargar un repositorio existente (*clone*)
  - Agregar archivos para ser versionados (*add*)
  - Guardar Versión de código local (*commit*)
  - Generar un Repositorio desde una carpeta local (*remote add origin*)
  - Abrir una rama (*branch/checkout*)
  - Actualizar la versión local con la versión del repositorio (*pull*)
  - Actualizar la versión del código en el repositorio (*push*)
- Otros archivos importantes
- Preguntas?
- Links útiles

Qué es Git?

Git es un sistema de Control de Versiones (VCS).

Este tipo de sistemas guardan una versión centralizada de la información y permiten hacer seguimiento a los cambios realizados por aquellos usuarios con permiso para modificar el contenido.

Generalmente se utiliza para código fuente de aplicaciones.



Qué es Github?

Github es el sitio más popular para tener un repositorio de código con Git.

El código alojado en github puede ser público o privado (para empresas).

Las cuentas gratuitas tienen la limitación de tener repositorio ***público***

Ventajas de usar un versionador

- Resguardar el código
- Volver parte o la totalidad del código al estado que uno necesita
- Dividir el trabajo asegurando que cada integrante utiliza la última versión publicada de cada parte
- Integración simple con las herramientas de Desarrollo: IDEs, Consola.
- Portabilidad: Es sencillo sumar integrantes nuevos, y proveerles la última versión del código

Ventajas de Git sobre otros versionadores

- Es gratuito y abierto
- Se puede utilizar de forma simple en todas las plataformas
- Es el más utilizado en el ambiente de desarrollo
- Se puede regenerar el repositorio desde cualquiera de las copias
- Es rápido de utilizar
- Integra varios niveles de seguridad
    - User/Pass
    - 2FA (Two Phase Authentication - Autenticación en 2 etapas)
    - SSH/GPG
    - Signed Commits

Comandos Básicos: *git config*

El Comando “git config”, se utiliza para configurar la información de git. Aquí lo utilizaremos para configurar la información global

```
git config --global user.name "Tu nombre aquí"
```

```
git config --global user.email "tu_email_aquí@example.com"
```

Este comando guarda las opciones “globales” del usuario con el nombre y con el email.

Comandos Básicos: *git init*

El Comando “git init”, se utiliza para informar a git que nuestra carpeta contendrá un repositorio.
sintaxis:

```
git init <Mi_Proyecto>
```

Este comando genera una carpeta oculta“.git”, en la cual git almacena la información para el manejo de versiones.

Comandos Básicos: *git clone*

El Comando “git clone”, se utiliza para generar una copia de un repositorio a partir de la carpeta actual.

sintaxis:

```
git clone “URL”
```

Este comando genera una carpeta, nombrada de igual forma que el repositorio yotra oculta“.git”, en la cual git almacena la información para el manejo de versiones

Comandos Básicos: *git add*

El Comando “git add”, se utiliza para agregar archivos al tesoro (stash) donde se almacenan hasta ser sincronizados

sintaxis:

```
git add .
```

```
git add mi_primer_programa.py
```

Este comando todos los archivos desde el lugar donde uno se encuentra al stash, para luego ser sincronizados.

La segunda línea muestra cómo utilizar el comando para sumar unicamente el archivo “mi_primer_programa.py”

Comandos Básicos: *git commit*

El Comando “git commit”, se utiliza para agregar marcar un grupo de cambios juntos, preferentemente con un mensaje (-m) o especificar que el commit sea firmado “-S”

sintaxis:

```
git commit -m “Mi primer commit”

git commit -S -m “Mi Primer commit firmado”
```

Este comando todos los archivos del stash, se marcaron con su respectiva versión, y agruparon bajo un commit.

La segunda línea muestra cómo utilizar el modificador “-S” para especificar que se firme el commit, con la clave de firma que uno haya configurado. Esto es una medida de seguridad para evitar que alguna persona no autorizada suba cambios al repositorio.

Comandos Básicos: *git remote add origin*

El Comando “git remote add origin”, se utiliza para iniciar un repositorio con nuestro código de proyecto actual

sintaxis:

```
git remote add origin “URL”
```

Este comando copiará el status de nuestro proyecto (último Commit) al repositorio (URL) donde se centralizará (por ejemplo: github.com/mi_proyecto)

Comandos Básicos: *git checkout*

El Comando “git checkout”, se utiliza para cambiar la rama de trabajo actual por otra. Si no estabamos en ninguna rama de trabajo, estábamos en la rama principal o “master”

sintaxis:

```
git checkout “mi_rama_de_trabajo”
```

```
git checkout -b “mi_rama_de_trabajo”
```

Este comando nos permite cambiar a la rama de trabajo y preservar la rama anterior (o la principal) y que los cambios que hagamos, estén solamente en nuestra rama.

La segunda línea, al no existir la rama, la crea, con el nombre “mi_rama_de_trabajo” y nos copia todos los archivos de la rama actual con su versión local.

Las modificaciones locales a los archivos, son mantenidas, por lo que un archivo actualizado, mantendrá nuestros cambios

Comandos Básicos: *git branch*

El Comando “git branch”, se utiliza para cambiar de rama (o saltar de rama) a la rma de trabajo en la que queremos estar. Esto nos permite trabajar en distintas versiones de producto, sin afectar el código de la rama principal (master)

sintaxis:

```
git branch --list
```

```
git branch “mi_rama”
```

Este comando lista todas las ramas existentes en la especificación de la copia local del proyecto. Si hemos sincronizado nuestro proyecto, veremos todas las ramas que existen en el proyecto centralizado (remoto)

La segunda línea cómo podemos hacer para crear una rama nueva (otra alternativa a “git checkout -b ‘mi_rama’ ”).

Comandos Básicos: *git pull*

El Comando “git pull”, se utiliza para traer la última versión del repositorio centralizado a nuestra versión local

sintaxis:

```
git pull
```

Este comando nos trae los cambios de la rama actual, e informa de las diferencias encontradas con las versiones locales.

Es muy importante empezar a trabajar utilizando este comando para tomar todos los cambios “antes de empezar a modificar el código”

Comandos Básicos: *git push*

El Comando “git push”, se utiliza para subir los cambios de nuestra rama de la versión local al repositorio.

sintaxis:

```
git push
```

Este comando sincroniza los commits almacenados con el repositorio remoto, en la rama en la cual estamos trabajando.

Otros Archivos importantes

Existen algunos archivos simples y de utilidad para configurar en un repositorio Git

- .gitignore
  - Sirve para guardar los archivos, directorios o máscaras de aquellos elementos que deben versionarse (ignorar)
- README.md
  - Es el archivo donde se suele dejar una explicación de que contiene el repositorio y como debe ser utilizado/ejecutado
- CHANGELOG.md
  - Es el archivo en el cual se suelen guardar los cambios introducidos en la version que se sube al repositorio

Links útiles

- https://git-scm.com/
- https://github.com/
- https://www.diegocmartin.com/tutorial-git/

Preguntas?