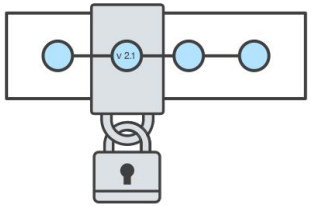
Git Basics

Universidad de SanAndrés

Control de Versiones



Control de versiones (en inglés, “*version control*”), es la práctica de seguimiento y administración de cambios en el código.

Control de Versiones

- Programas de control de versiones **mantienen un seguimiento de toda modificación hecha en el código**—en un tipo de almacenamiento especial.
- El control de versiones **protege el código fuente** tanto de una catástrofe como de una equivocación, y sus consecuencias no intencionadas.
- Ante alguna equivocación en las versiones, es posible volver hacia atrás en la **historia**.
- Es muy útil—seguramente necesario—en equipos de desarrollo donde cada integrante está trabajando en una pequeña parte del código fuente, pudiendo integrar el trabajo hecho sin mayores disrupciones.
- Entre otras cosas permite:
    - Conocer el autor y el momento de la contribución hecha.
    - Mantener varias versiones del código fuente al mismo tiempo.

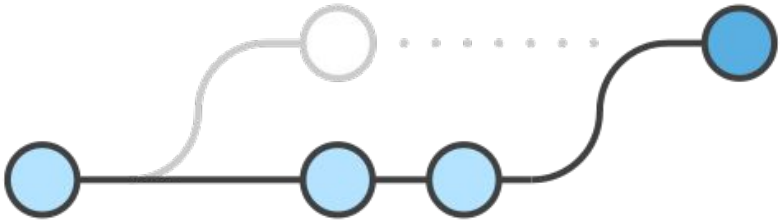
Beneficios de Sistemas de Control de Versiones

- Sistemas de Control de Versiones, en inglés *"Version Control Systems (VCS)"* son también llamados herramientas de *"Source Code Management (SCM)"* o *"Revision Control System (RCS)"*.
- El más popular es Git, el cual es gratis y open source.
- Entre muchos de los beneficios están:
    - Un historia completa a largo plazo de cada modificación de cada uno de los archivos.
    - *Branching* y *Merging* para el trabajo colaborativo en un mismo proyecto sin entorpecer el trabajo de otros, ni la salud del programa en desarrollo.
    - La posibilidad de trazar los cambios hechos y conectarlo con otros plataformas de seguimiento de problemas y tareas.

La pregunta no es si debemos usar un VCS, sino cuál VCS usar. No usarlo implica un gran riesgo para el proyecto.

Buenas prácticas de los SCM

- Realizar *commits* frecuentemente.
- Asegurarse de estar trabajando en la última versión.
- Hacer notas detallada del trabajo en cada *commit*.
- Revisar los cambios antes de crear un *commit*.
- Usar ramas (*branches*).
- Respetar el flujo de trabajo acordado en el equipo.



¿Qué es Git?

- Git es el sistema de control de versiones moderno más popular en el mundo hasta el día de hoy.
- Fue creado en 2005 por Linus Torvalds, quién también creó el famoso sistema operativo Linux.
- Muchas compañías dependen del control de versiones a partir de Git como así también muchos proyectos open-source.
- Se lo conoce como un software *cross-platform*, funciona en los sistemas operativos más famosos como Windows, macOS, y Linux.
- Git es un sistema distribuido donde cada desarrollador puede tener una copia de trabajo con la historia completa del proyecto en cuestión.
- Git fue diseñado con criterios de performance, seguridad y flexibilidad.

Instalar Git

Mac OS X

- Hay muchas maneras de instalar Git en Mac OS X. Puede que ya esté instalado por defecto si se instaló XCode.
- Para ello abrir la Terminal (Command + Space, escribir Terminal), y ejecutar el siguiente comando.

```
$ git --version
git version 2.9.2
```

Instalar Git

Git para Mac Installer

1. Descargar la última versión de Git para Mac Installer.
2. Seguir el prompt para instalar Git.
3. Para verificar la instalación, ejecutar, en una terminal, el comando:

```
$ git --version
git version 2.9.2
```

Instalar Git

Git con Homebrew

1. Abrir la Terminal.
2. Ejecutar el comando de homebrew:

```
$ brew install git
```

3. Verificar la instalación de Git con el comando:

```
$ git --version
git version 2.9.2
```

Instalar Git

Windows

1. Descargar el último Git para Windows Installer.
2. Una vez ejecutado exitosamente el instalador, se verá el wizard para el set up de Git. Seguir los **Next** y el **Finish** para completar la instalación. La opciones predeterminadas generalmente funciona para los usuarios comunes.
3. Abrir el Command Prompt (o Git Bash si durante la instalación se eligió no utilizar el Command Prompt de Windows).
4. Es posible verificar la versión de Git con el siguiente comando:

```
$ git --version
git version 2.35.1.windows.2
```

Instalar Git

Linux

1. Abrir la terminal con Ctrl+Alt+T.
2. Actualizar APT.

```
$ sudo apt update
```

3. Instalar Git con APT

```
$ sudo apt install git
```

4. Verificar versión

```
$ git --version
git version 2.9.2
```

Configurar Usuario de Git

1. Agregar nombre.

```
$ git config --global user.name "Guillermina Paris"
```

2. Agregar email.

```
$ git config --global user.email "gparis@udesa.edu.ar"
```

Git con GUI

Muchos usuarios no utilizan la Terminal de Linux o Mac OS X, y tampoco utilizan el Command Prompt de Windows. Simplemente por una cuestión de gustos.

Existen varias alternativas de clientes GUI:

- Git Kraken
- SourceTree
- Github Desktop
- TortoiseGit
- Y muchos mas...

Para ver más alternativas, sigan este link.

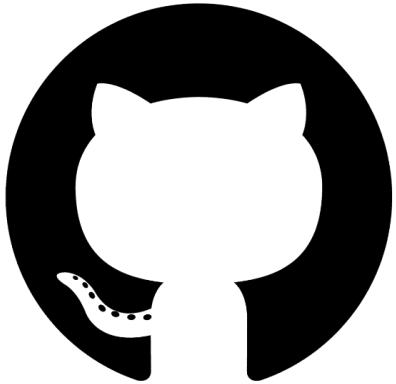
Aplicaciones en base a Git

La mayoría de los programadores/desarrolladores no utilizan Git directamente (a veces sí). Todos nosotros utilizamos productos hechos en base a Git que ofrecen muchas otras funcionalidades sobre Git para un mejor trabajo en equipo y para un mejor seguimiento de las proyectos.

Estos proyectos son Web-based, es decir, que se acceden por el navegador.

Ejemplos de estos son:

- Github
- Gitlab
- Bitbucket
- AWS CodeCommit
- Y muchísimas más...



GitHub

Repositorios

- Un repositorio es un espacio donde nuestros proyectos viven.
- Los repositorios vamos a manejarlos desde el navegador como así también desde la terminal o Command Line Interface (CLI).
    - Crear un repositorio.
    - Clonar un repositorio.
    - Hacer un *fork* de un repositorio.
    - Subir código.
    - Bajar código.
    - Borrar un repositorio.
    - Agregar algunos permisos.
    - Autenticación.

Para más información sobre estos tópicos seguir la documentación oficial de Github en: https://docs.github.com/en/repositories.

Autenticación

Para manejar repositorios **privados**, es necesario estar autenticados con la cuenta que tiene acceso a los repositorios.

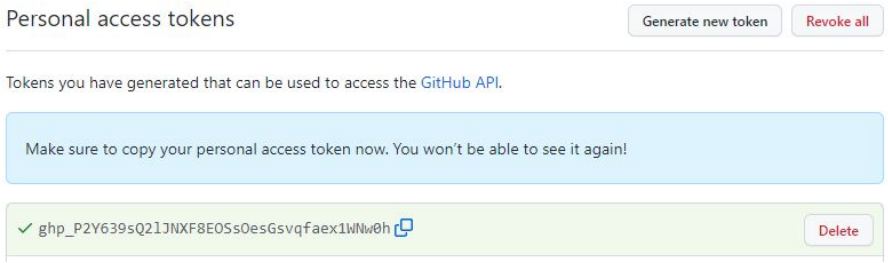
La mejor forma es utilizar Github CLI. Una vez instalado es necesario realizar los siguientes pasos:

1. Generar un Personal Access Token (PAT).
2. Autenticarse con Github CLI.

Personal Access Token

La generación del access token se realiza en la cuenta de Github.

1. Dirigirse a ***`Profile > Settings`***.
2. Ingresar en ***`“Developer Settings”`***.
3. Luego, en ***`Personal Access Tokens`***, tocar el boton ***`Generate New Token`***.
4. Elegir, ***`Expiration > No Expiration`***.
5. Añadir una nota, y los permisos ***`‘repo’`***, ***`‘read:org’`***, ***`‘workflow’`***.
6. Guardar el token generado en algún lugar seguro (no se puede volver a ver en Github).



Github CLI

Para autenticarse con Github CLI:

1. Abrir el Command Prompt de Windows.
2. Ejecutar el comando:

```
$ gh auth login
```

3. Presionar ***enter*** hasta que aparezca la pregunta: *"How would you like to authenticate Github CLI?"*. Elegir *"Paste an authentication token"*.
4. Pegar el token generado y presionar ***enter***.

Más Temas...

- Commits.
- Branching y Merging.
- Rebasing.
- Pull Requests.
- Remotes.
- Historia.
- Formato Markdown.
- README.