Git

¿Qué es?

Git es un **sistema de control de versiones distribuidas** de código abierto y gratuito diseñado para manejar todo, desde proyectos pequeños a muy grandes, con velocidad y eficiencia.

¿Qué es el control de versiones, y por qué debería importarte? El control de versiones es un sistema que registra los cambios realizados sobre un archivo o conjunto de archivos a lo largo del tiempo, de modo que puedas recuperar versiones específicas más adelante. Cualquier tipo de archivo que encuentres en un ordenador puede ponerse bajo control de versiones.

Si eres diseñador gráfico o web, y quieres mantener cada versión de una imagen o diseño (algo que sin duda quieres), un sistema de control de versiones (Version Control System o **VCS** en inglés) es una elección muy sabia. Te permite revertir archivos a un estado anterior, revertir el proyecto entero a un estado anterior, comparar cambios a lo largo del tiempo, ver quién modificó por última vez algo que puede estar causando un problema, quién introdujo un error y cuándo, y mucho más. Usar un VCS también significa generalmente que si fastidias o pierdes archivos, puedes recuperarlos fácilmente. Además, obtienes todos estos beneficios a un coste muy bajo.

Estructura de git

Git suele usarse para realizar un control de versiones entre un repositorio local y uno remoto. Vamos haciendo cambios en nuestro repositorio local y en cualquier momento podemos enviarlos al repositorio remoto para que los demás participantes en el proyecto puedan disponer de los cambios realizados. Por tanto, vemos que hay (o puede haber) dos zonas:

- La local **siempre va a existir** ya que es donde reside mi proyecto
- La remota *puede no existir*, por ejemplo en el caso de que sea un proyecto personal

Zona local

Aquí es donde vas realizando los cambios en el proyecto. Todos estos cambios no se reflejarán en el remoto hasta que tú decidas.

Flujo de trabajo *normal*

Tu zona de trabajo está compuesta por tres "árboles" administrados por git. El primero es tu **Directorio de trabajo** que contiene los archivos (con todos los cambios que vas haciendo), el segundo es el **Index** (**área de preparación o stage area**) que actúa como una zona intermedia, y el último es el **HEAD** o repositorio local que apunta al último `commit` (ya veremos que es esto) realizado.

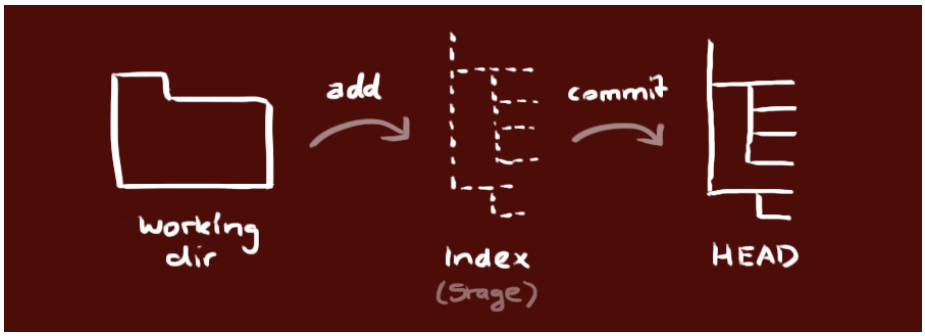


Figure 1: Git Flow

Cuando realizas cambios en un archivo local llega un momento que decides que quieres guardar una copia del mismo (para poder volver a la misma en cualquier momento). Para ello debes pasarlo a tu zona *stage* (ensayo).

Ahora mismo tienes una copia del archivo. Si decides que esa copia que has pasado a estado *staged* ya es candidata para pasarla al repositorio remoto, debes moverla a la zona *Head*.

Ten en cuenta que esto sólo pasa en local.

Es decir:

- **Working Directory**, mantiene los archivos en su estado actual
- **Index**, mantiene los archivos en un estado intermedio
- **Head**, mantiene los archivos candidatos a subir al repositorio remoto

Si una versión concreta de un archivo está en el directorio de HEAD, se considera confirmada (**committed**). Si ha sufrido cambios desde que se obtuvo del repositorio, pero ha sido añadida al área de preparación, está preparada (**staged**). Y si ha sufrido cambios desde que se obtuvo del repositorio, pero no se ha preparado, está modificada (**modified**). Si un archivo está en tu directorio de trabajo pero no está gestionado por git, está en estado **untracked**

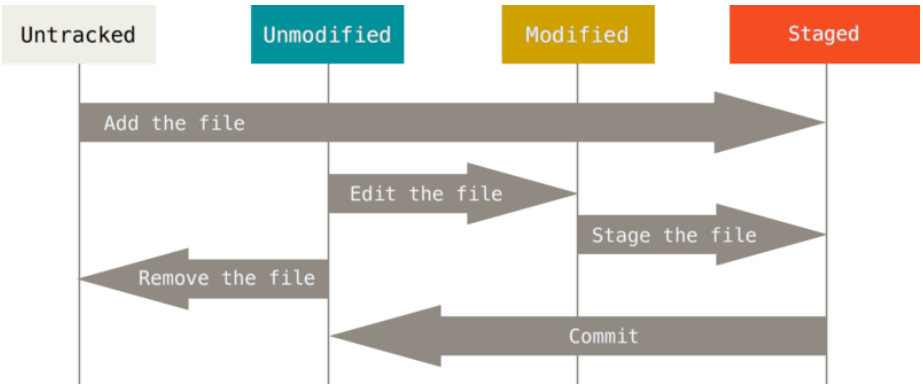


Figure 2: Life Cycle

Zona Remota

La parte remota sólo está compuesta por el repositorio remoto y refleja el estado actual del proyecto (ten en cuenta que en él pueden participar muchos desarrolladores).

Cuando decides que los cambios que has hecho en local ya pueden pertenecer al repositorio remoto, debes subirlos.

Instalar git

Como siempre,

```
sudo apt-get install git
```

Configurar git por primera vez

Lo primero que deberás hacer cuando instales Git es establecer tu nombre de usuario y dirección de correo electrónico. Esto es importante porque los "commits" de Git usan esta información, y es introducida de manera inmutable en los commits que envías:

```
$ git config --global user.name "Víctor Ponz"
$ git config --global user.email "victor.ponz@ieselcaminas.org"
```

Más información en el manual de git

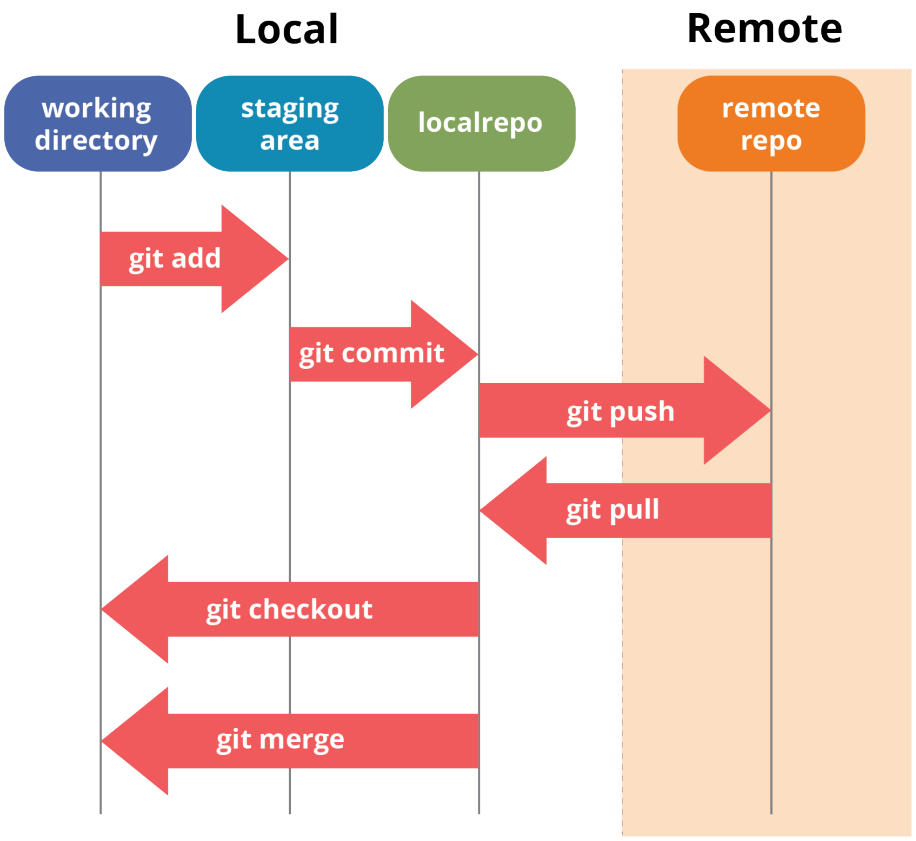


Figure 3: Local

Obtener ayuda

Si alguna vez necesitas ayuda usando `Git`, existen tres formas de ver la página del manual (`manpage`) para cualquier comando de Git:

```
$ git help <verb>
$ git <verb> --help
$ man git <verb>;
```

Por ejemplo, puedes ver la página del manual para el comando `config` ejecutando

```
$ git help config
```

Comandos básicos

Para añadir un proyecto al control de versiones de git

```
git init
## Dentro del directorio del proyecto
```

Para comprobar el estado del repositorio local

```
git status
```

El siguiente diagrama muestra el flujo de trabajo junto con los comandos git:

Para añadir un archivo al área de ensayo (*staged area*)

```
git add archivo.txt /* Se pueden usar wildchars */
```

El archivo ahora estará en el área de ensayo, preparado para formar parte del historial del proyecto.

Para registrar cambios en el historial:

```
git commit -m "Comentario asociado a este commit"
```

Para confirmar los cambios del último `commit` y guardarlos en el repositorio `main`.

```
git push origin main
```

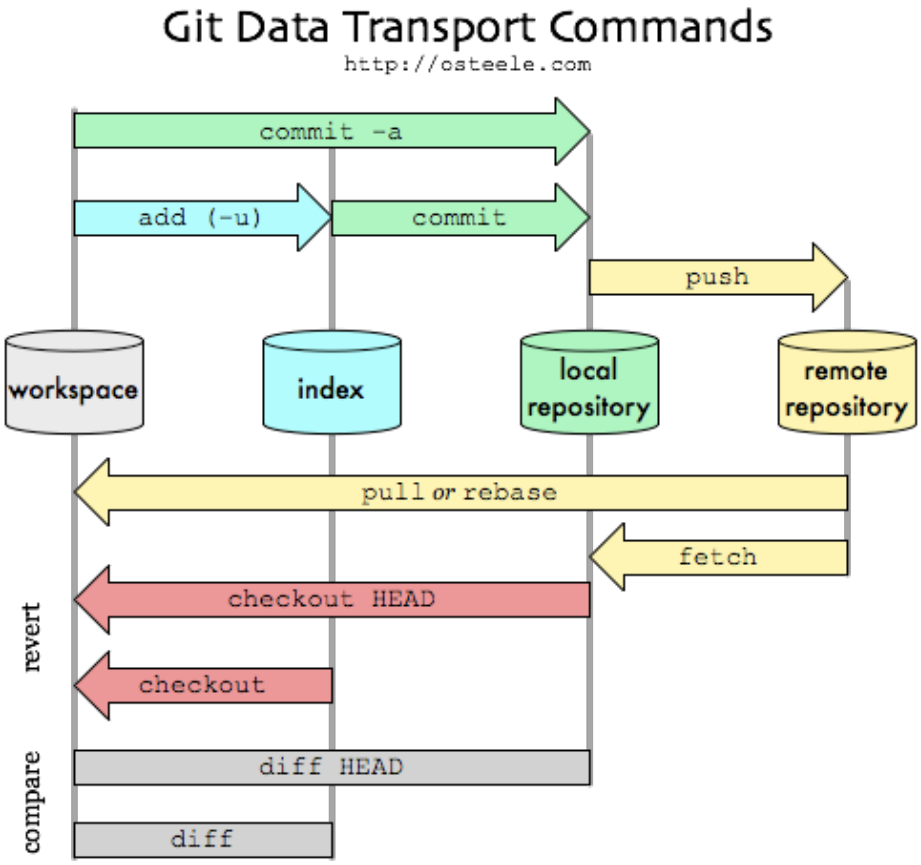


Figure 4: Git Data Transport Commands

Para traer los cambios realizados en `main` o `master`, (depende de settigns globales de GitHub) a tu copia local

```
git pull origin main
```

Para volver al estado original de un archivo cuando todavía no se ha hecho add

```
git checkout nombre-de-archivo
```

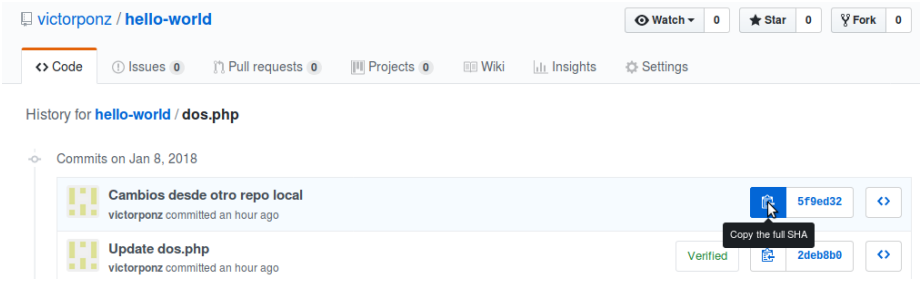
Si ya se ha hecho el `add` y quiero volver a una versión anterior

```
git reset HEAD nombre-de-archivo
git checkout nombre-de-archivo
```

Volver a una versión concreta

```
git checkout hash-de-la-revisión -- nombre-de-archivo nombre-de-archivo
```

Para obtener el `hash` en GitHub debemos seleccionar un archivo, hacer clic en History y luego en el botón que dice "*Copy the full SHA*"



Para actualizar mi zona local con los cambios realizados en el repositorio remoto

```
git pull
```

Esto te puede ser útil para sincronizar los cambios entre clase y tu casa. Por ejemplo, si haces cambios en clase, en casa ejecuta `git pull` para que te traiga todos los cambios. Recuerda que primero has de hacer un `git push` en clase.

GitHub

Para poder trabajar correctamente con `git`, es necesario tener un repositorio donde alojar nuestros proyectos.

Para ello vamos a usar GitHub que nos permite tener repositorios gratuitos.

> **Nota importante**.
>
> El Instituto dispone de cuentas para todo el alumnado, así que es mejor que uséis dicha cuenta. Si no la conocéis preguntad.

Además, el Instituto pertenece al Programa GitHub for Education lo que permite disfrutar al alumnado de una serie de herramientas agrupadas en Gimage-20201110183906307itHub Student Developer Pack,

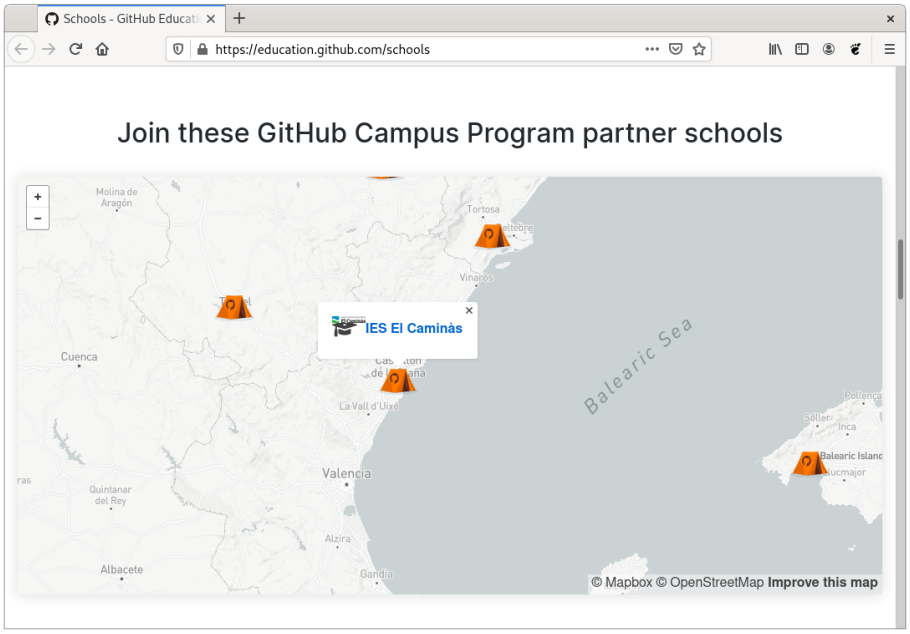


Figure 5: GitHub Campus Program

Crear un repositorio

Una vez registrados, haced clic en el botón "*Read the guide*" que os guiará para crear un nuevo repositorio en `github`.

Una vez creado, el aspecto de dicho repositorio será parecido a esto:

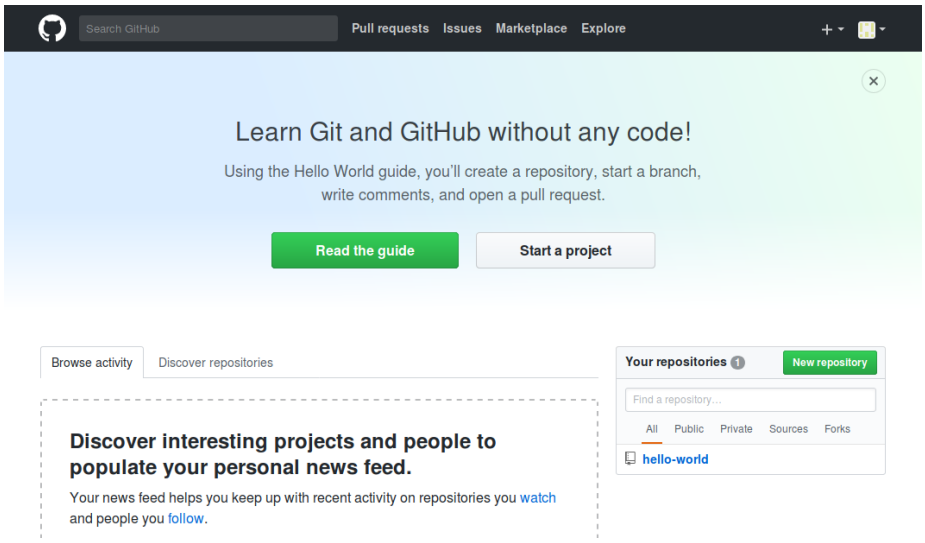
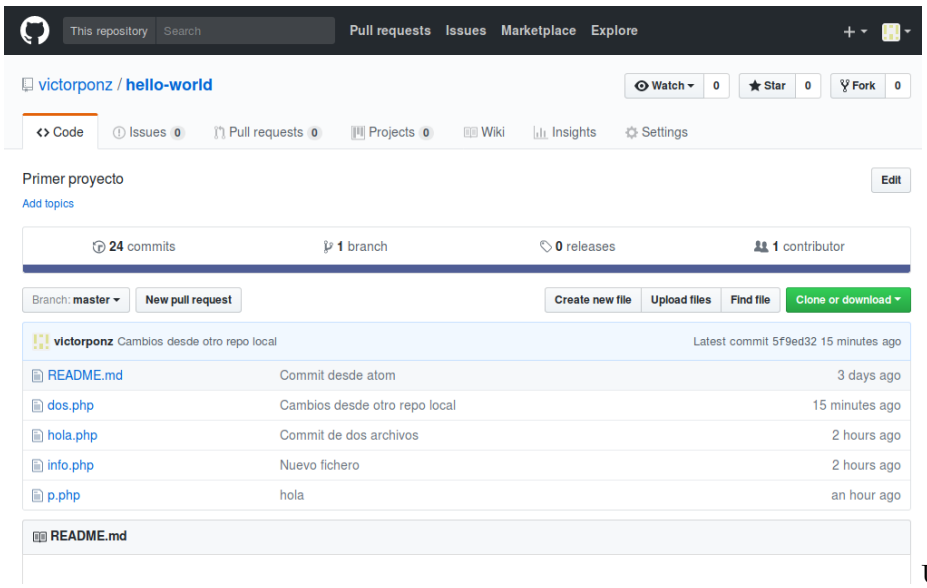


Figure 6: Crear repositorio



Una vez creado el repositorio en `github` vamos a clonarlo para usarlo en nuestro ordenador.

El comando es:

```
git clone [url]
```

Donde [url] es la url del repositorio a clonar. Para ver la url, haced clic en el

botón *"Clone or download"*

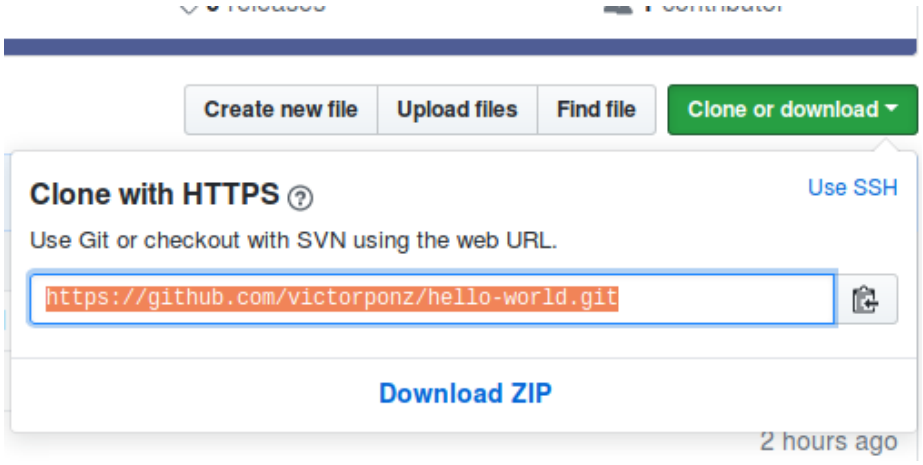


Figure 7: Clonar Repo

Por tanto, para clonar este repositorio hay que cambiar al directorio donde queramos clonarlo y escribir el comando:

```
git clone https://github.com/victorponz/hello-world.git
```

Al hacerlo, creará un directorio `.git` oculto donde irá almacenado toda la información relativa al estado de la copia local.

Bitbucket

El proceso es muy parecido en Bitbucket

Crear un nuevo repositorio remoto a partir de uno local

1. El primer paso será crear un nuevo repositorio remoto **vacío** y copiar la url del mismo.
2. En el proyecto local inicializarlo para usar git

   ```
   git init
   ```

3. Ahora añadir todos los archivos

   ```
   git add *
   ```

4. Hacer el primer `commit`

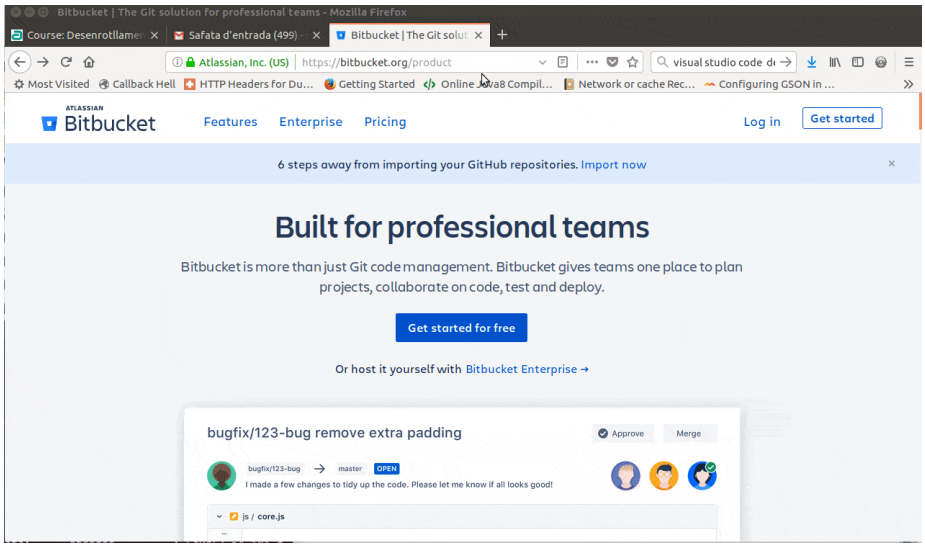


Figure 8: Bitbucket

```
git commit -m "First commit"
```

5. Ahora añadir la url del repositorio remoto

   ```
   git remote add origin URL-del-repositorio-remoto
   ```

6. Subir los cambios al repositorio remoto

   ```
   git push origin main //o master, dependiendo de la configuración
   ```

Aprende git online

Antes de empezar a trabajar con un repositorio real, visitad la página GitInit en Codecademy.com y seguid el curso.

También podéis usar el siguiente curso en PluralSight

Otro enlace muy interesante es el dedicado a aprender git en Atlassian.

Tarea - Crear cuenta en GitHub

Crea una cuenta en **GitHub** si no la tienes ya creada. Usa, si es posible, el correo del instituto

Adaptado de los siguientes materiales

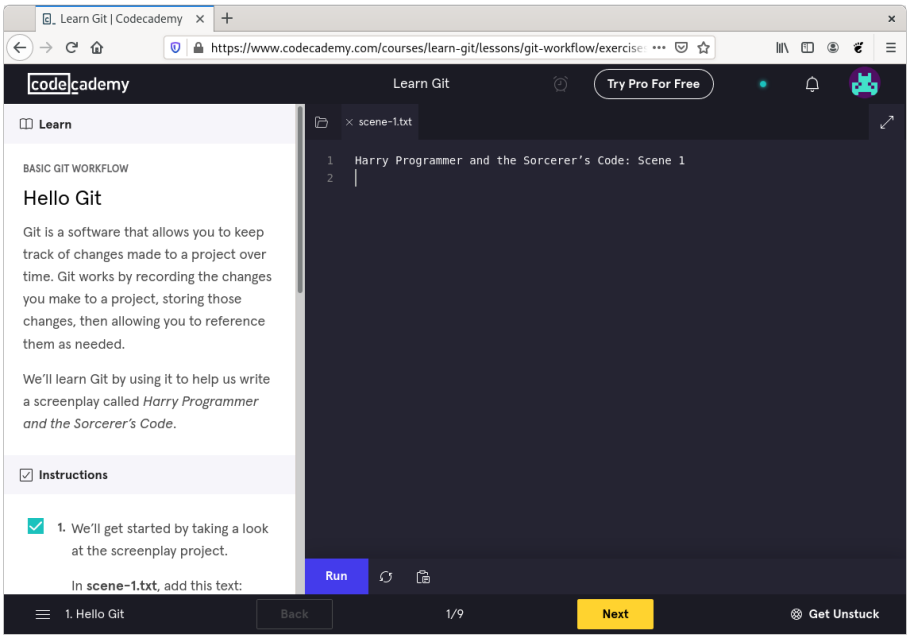


Figure 9: Codecademy

- https://git-scm.com/book/es/v2/
- https://www.quora.com/Whats-the-difference-between-committing-and-pushing-in-Git
- https://blog.osteele.com/2008/05/my-git-workflow/
- http://rogerdudler.github.io/git-guide/

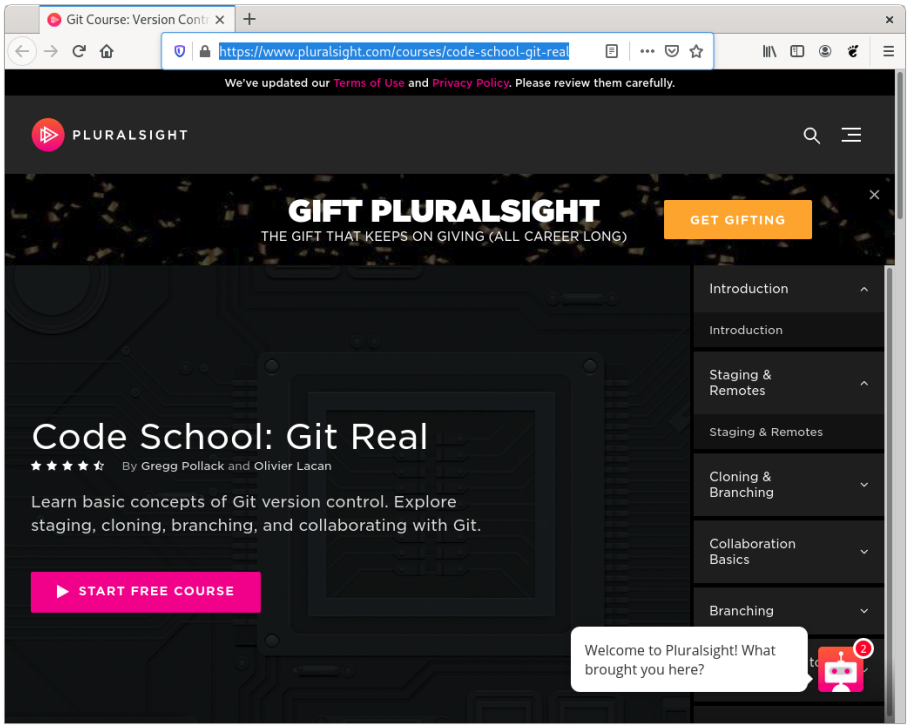


Figure 10: image-20201110183200149