

Control de versiones con Git

Por Karla García
Github & Gitlab: @kaarla

¿Editar archivos es una tarea destructiva?

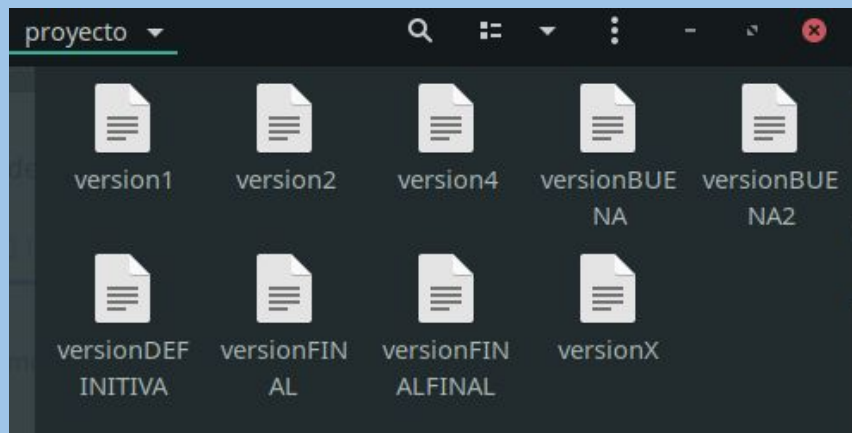


- Muchas veces eliminamos o modificamos componentes de un archivo que ya existían en alguna "versión" previa.
- Es complicado mantener versiones (anteriores) estables mientras realizamos nuevas modificaciones.

Control de versiones

Necesidad: mantener una organización en las versiones de un archivo o conjunto de archivos (proyecto).

Objetivo: Preservar una versión que ya funciona, antes de comprometernos a modificarla.



Controlador de Versiones



mercurial

Permiten:

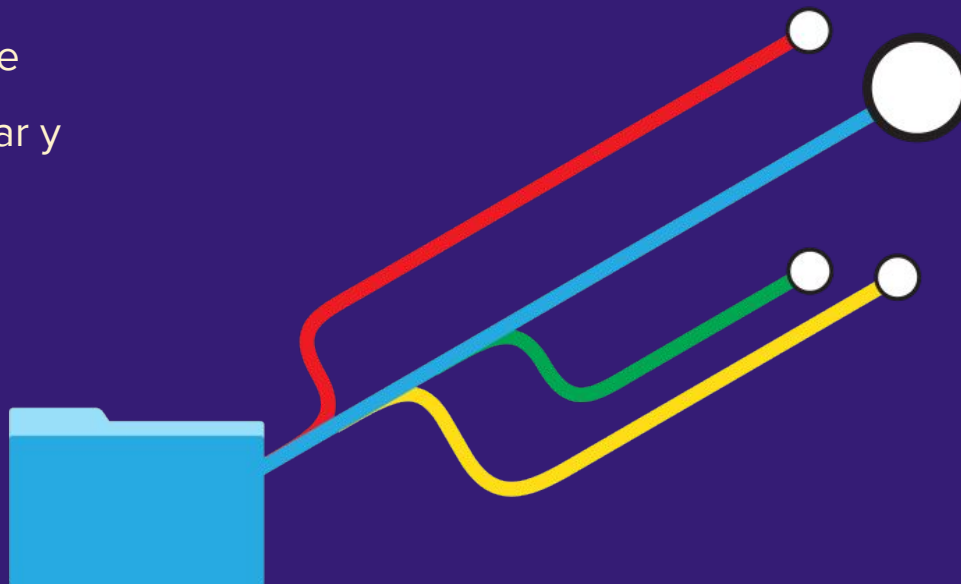
- Tener todas las versiones, guardadas, disponibles en todo momento.
- Despreocuparse por cometer errores en versiones posteriores causando pérdidas de avances importantes.
- Facilita el trabajo colaborativo en línea. Resulta sencillo organizar las versiones de cada integrante acerca del proyecto.

Repositorio

Un repositorio es el directorio donde se encuentra el proyecto con el historial de versiones listo para descargarse, realizar y actualizar cambios desde la misma plataforma.

Remoto: En una plataforma en línea.

Local: En una computadora física.



Un poco de historia...

- **1972**: El primer sistema de control de versiones para archivos fue Source Code Control System (**SCCS**)
- **1982**: **SCCS** fue, en general, reemplazado por el Revision Control System (**RCS**).
- En **1990** Dick Grune lanzó Concurrent Versions System (**CVS**) que ya podía trabajar con directorios (carpetas).



Otro poco de historia...

- En los 90's, básicamente, todos los proyectos de software libre en y para Linux utilizaron CVS.
- BitMoover comenzó el desarrollo de BitKeeper orientado a las necesidades de Linus Torvalds y, por lo tanto, de Linux.
- La ventaja de este sistema es que era descentralizado, a diferencia de CVS.



¿Sabías que... Linus Torvalds recibía las contribuciones al kernel de Linux por correo electrónico.

Origen de git

- La resistencia a BitKeeper, aunado a que no era software libre, resultaron en la creación de varios sistemas de control de versiones distribuidos.
- En 2005 BitMover retira la versión gratuita de BitKeeper que usaba la comunidad de desarrolladores del kernel de Linux.
- Linus Torvalds decidió tomarse unas semanas para escribir su propio sistema de control de versiones distribuido.



Tip: investiga a qué se dedica Andrew Tridgell y qué tuvo que ver en esto.

¡El nacimiento de una nueva era!



Resumiendo...

¿Qué es git?

Es un sistema de control de versiones distribuido...

que utiliza criptografía para toda la historia de un repositorio.

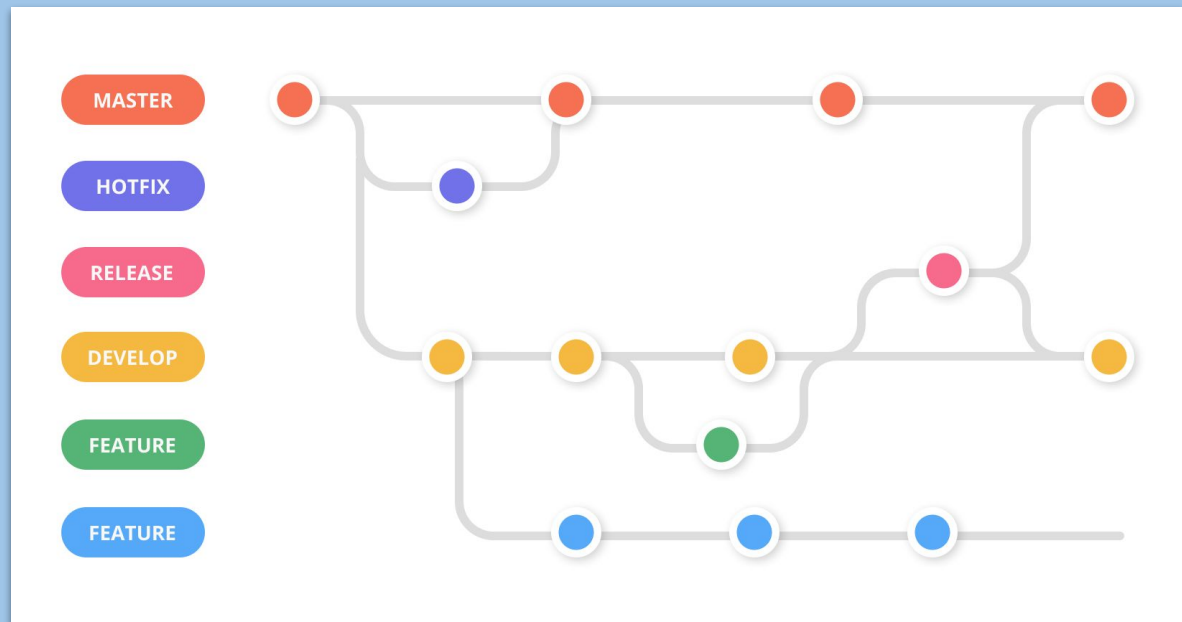
pensado para hacer bifurcaciones de manera muy rápida,

Y es, además, un proyecto de Software Libre.



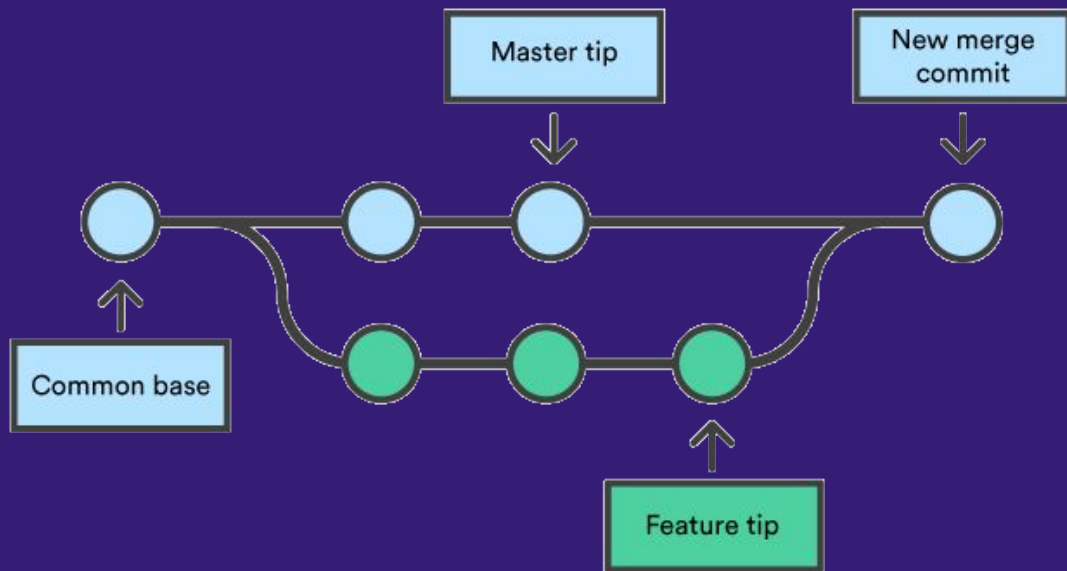
Ramas

- Las bifurcaciones en Git son llamadas "ramas", las cuales nos permiten trabajar sobre diferentes versiones de un proyecto de forma simultánea.
- Tendremos una rama principal llamada "master" o "main" sobre la que mantendremos la versión con los cambios "aceptados" del proyecto.



Ramas

- En las demás ramas podemos trabajar en el desarrollo de diferentes componentes del proyecto.
- Al considerar terminada la tarea de la rama la uniremos con “master” de manera que permanezca actualizada con respecto a la información de las ramas alternas.



Instalación

- UBUNTU

```
~ >>> sudo apt update
```

```
~ >>> sudo apt install git
```

- FEDORA

```
~ >>> sudo yum update
```

```
~ >>> sudo yum install git
```

- ARCH

```
~ >>> sudo pacman -Syy
```

```
~ >>> sudo pacman -S git
```

Verificar versión

- Cualquier distribución de Linux

```
~ >>> git --version  
git version 2.28.0  
~ >>>
```

Git

Comandos básicos

Por Karla García
Github & Gitlab: @kaarla

Estructura de un comando de Git

```
~ >>> git comando -opcion1 -opcionX
```

1. Palabra "git"

2. Comando principal

3. Opciones del comando

Creación de un repositorio

Crear un repositorio es posible desde un directorio existente:

```
~ >>> mkdir ejemplo
~ >>> cd ejemplo
~/ejemplo >>> git init .
Initialized empty Git repository in /home/karla/ejemplo/.git/
~/ejemplo >>> █
```

... o “desde cero” creando el directorio al tiempo que se inicializa el repositorio:

```
~ >>> git init ejemplo

Initialized empty Git repository in /home/karla/ejemplo/.git/
~ >>> cd ejemplo

~/ejemplo >>> █ ±[master]
```


Estado inicial del repositorio

El directorio con la configuración del repositorio se llama “.git”, recuerda que la bandera “-a” sirve para ver los archivos y directorios ocultos (los cuales tienen “.” al inicio de su nombre).

```
~/ejemplo >>> ls -a
.  ..  .git
~/ejemplo >>> ls .git
branches  description  hooks  objects
config    HEAD         info    refs
~/ejemplo >>> 
```

En Linux...

- es el directorio actual
- es el directorio “padre” del actual
- / es el directorio raíz del sistema
- ~ es el alias para “/home/user/”

Estado inicial de un repositorio

Por ahora el repositorio está vacío, podemos verificar su estado en cualquier momento con el comando **status**

```
~/ejemplo >>> git status ±[master]  
On branch master  
  
No commits yet  
  
nothing to commit (create/copy files and use "git add" to track)
```

Pero si creamos un archivo, estaremos generando un cambio...

```
~/ejemplo >>> touch nuevoArchivo.txt ±[master]  
~/ejemplo >>> git status ±[●][master]  
On branch master  
  
No commits yet  
  
Untracked files:  
  (use "git add <file>..." to include in what will be committed)  
    nuevoArchivo.txt  
  
nothing added to commit but untracked files present (use "git add" to track)
```

Cambios en Git

- **M - Modified** para los archivos existentes que tuvieron cambios.
- **U - Untracked** para los archivos que fueron creados entre la último commit y el momento actual.
- **D - Deleted** para los archivos que fueron eliminados entre el último commit y el momento actual.

En nuestro ejemplo anterior, el cambio fue el de un archivo creado

```
~/ejemplo >>> touch nuevoArchivo.txt                                     ±[master]
~/ejemplo >>> git status                                                ±[●][master]
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    nuevoArchivo.txt

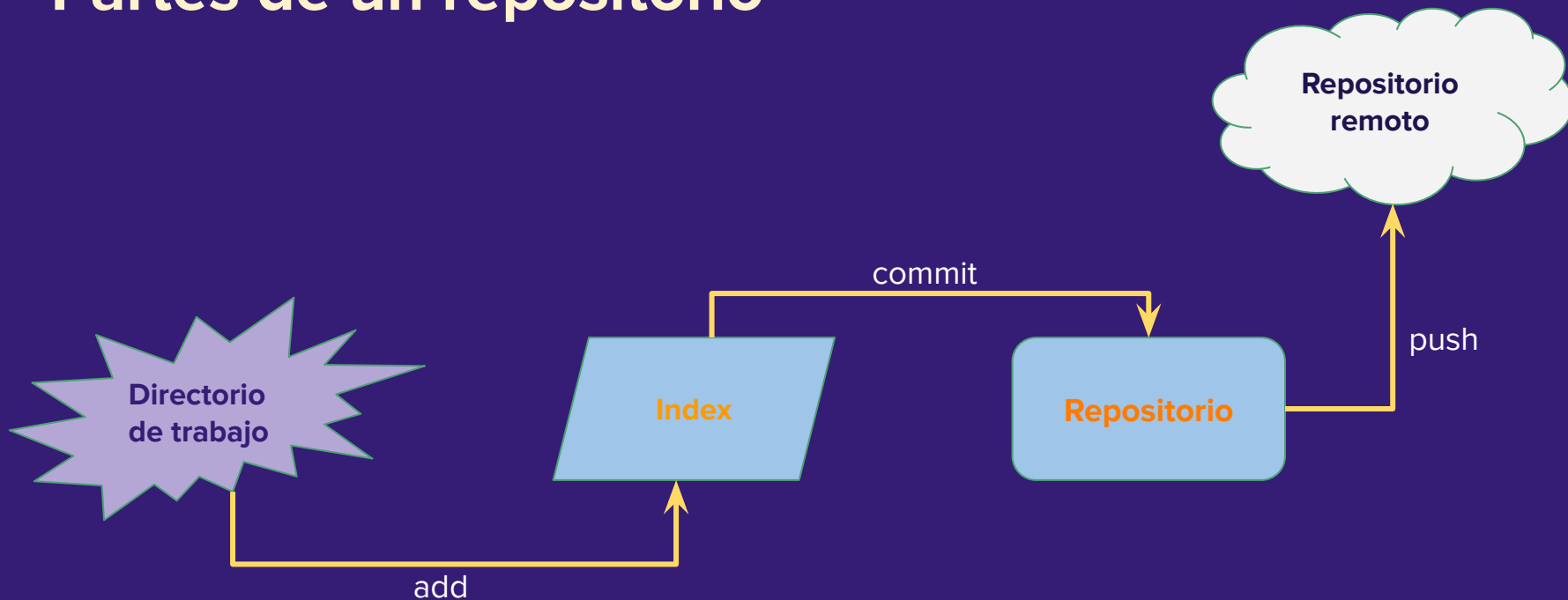
nothing added to commit but untracked files present (use "git add" to track)
```

Qué es un “commit”

Git permite guardar versiones de un proyecto cuando decidimos que tiene cambios que deseamos preservar; cada una de estas versiones se llama "commit", lo cual es un comprimido de los cambios realizados en esta versión hecho con un método de criptografía, lo cual le atribuye un identificador alfanumérico único.



Partes de un repositorio



Cómo hacer un commit

En resumen...

0. Guardar los cambios como normalmente lo haríamos.
1. **git add** (del directorio de trabajo a index)
2. **git commit** (de index al repositorio)
3. **git push** (**spoiler alert:** del repositorio local al repositorio remoto)

Cómo hacer un commit

1. **git add** para organizar en “index” los cambios para la nueva versión.

```
~/ejemplo >>> git add nuevoArchivo.txt
~/ejemplo >>> git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   nuevoArchivo.txt
```

Cómo hacer un commit - atajos de **git add**

- **git add -A** para agregar eliminaciones, creaciones y modificaciones.
- **git add .** para agregar las creaciones de archivos y modificaciones.
- **git add *** para agregar modificaciones y eliminaciones de archivos.
- **git add <nombre de archivo>** para agregar los cambios de un archivo en particular.



Cómo hacer un commit

2. **git commit** para crear la nueva versión del proyecto.

```
~/ejemplo >>> git commit -m "creación de archivo para ejemplo"
[master (root-commit) 1169f5c] creación de archivo para ejemplo
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 nuevoArchivo.txt
```

Se muestra la nomenclatura del tipo de cambio de cada archivo agregado en esta versión.

Se enlista la información sobre las líneas modificadas, insertadas y eliminadas.

Se crea un nuevo commit con identificador alfanumérico y el mensaje asignado.

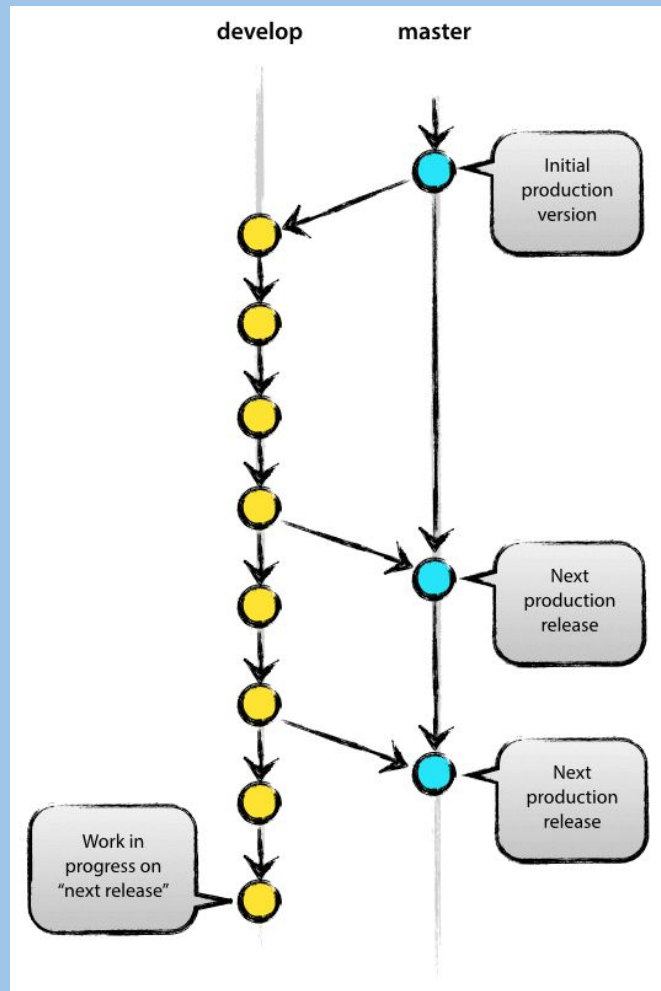
Git

Ramas y repositorios remotos

Por Karla García
Github & Gitlab: @kaarla

Ramas

Git nos permite trabajar en diferentes líneas de tiempo sobre las que es posible trabajar sobre versiones "simultáneas" de un proyecto. Estas bifurcaciones son llamadas ramas.



Creación de una nueva rama

Opción 1

- `git branch <nueva rama>` Para crear una rama
- `git checkout <nueva rama>` Para que HEAD apunte a esta rama

Opción 2

- `git checkout -b <nueva rama>`

Para crear una rama y hacer que HEAD apunte a ella, en un solo paso.

```
~/ejemplo >>> git checkout -b work ±[master]  
Switched to a new branch 'work'  
~/ejemplo >>> ±[work]
```

Nuevo cambio sobre rama work

```
~/ejemplo >>> echo "inicio del archivo nuevo en work" > archivoNuevo.txt
~/ejemplo >>> git status ±[●][work]
On branch work
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    archivoNuevo.txt

nothing added to commit but untracked files present (use "git add" to track)
~/ejemplo >>> git add -A ±[●][work]
~/ejemplo >>> git commit -m "inicio de trabajo en archivoNuevo" ±[●][work]
[work fdbf277] inicio de trabajo en archivoNuevo
1 file changed, 1 insertion(+)
create mode 100644 archivoNuevo.txt
~/ejemplo >>> git log ±[work]
```

1. Revisión de logs

```
commit fdbf277e0e8b5ba2e55eafa13c19da1a8dcda1b5 (HEAD -> work)
```

```
Author: kaarla <sgakarla@ciencias.unam.mx>
```

```
Date: Mon Sep 28 00:29:02 2020 -0500
```

inicio de trabajo en archivoNuevo

```
commit 1169f5c1e09555a955f6b931a1d4b27517fa8b27 (origin/master, master)
```

```
Author: kaarla <sgakarla@ciencias.unam.mx>
```

```
Date: Thu Sep 24 00:42:03 2020 -0500
```

creación de archivo para ejemplo

2. Cambio a commit previo

```
~/ejemplo >>> git checkout 1169f5c
```

```
±[work]
```

```
Note: switching to '1169f5c'.
```

You are in 'detached HEAD' state. You can look around, make experimental changes and commit them, and you can discard any commits you make in this state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may do so (now or later) by using `-c` with the switch command. Example:

```
git switch -c <new-branch-name>
```

Or undo this operation with:

```
git switch -
```

Turn off this advice by setting config variable `advice.detachedHead` to false

```
HEAD is now at 1169f5c creación de archivo para ejemplo
```

Verificamos checkout y revertimos

HEAD apunta al commit deseado

```
commit 1169f5c1e09555a955f6b931a1d4b27517fa8b27 (HEAD, origin/master, master)
Author: kaarla <sgakarla@ciencias.unam.mx>
Date: Thu Sep 24 00:42:03 2020 -0500

    creación de archivo para ejemplo

~
```

Revertimos:

```
~/ejemplo >>> git checkout - ±[master]
Previous HEAD position was 1169f5c creación de archivo para ejemplo
Switched to branch 'work'
~/ejemplo >>> +[work]
```


Git como principiantes

Git puede parecer difícil de usar al inicio, pero con práctica es posible dominarlo y aprovechar todas sus bondades.

Te animamos a usar sus funcionalidades básicas desde etapas tempranas de tu formación, ya que es una herramienta cotidiana de la vida profesional de las personas dedicadas al desarrollo de software.



Plataformas para manejar repositorios remotos



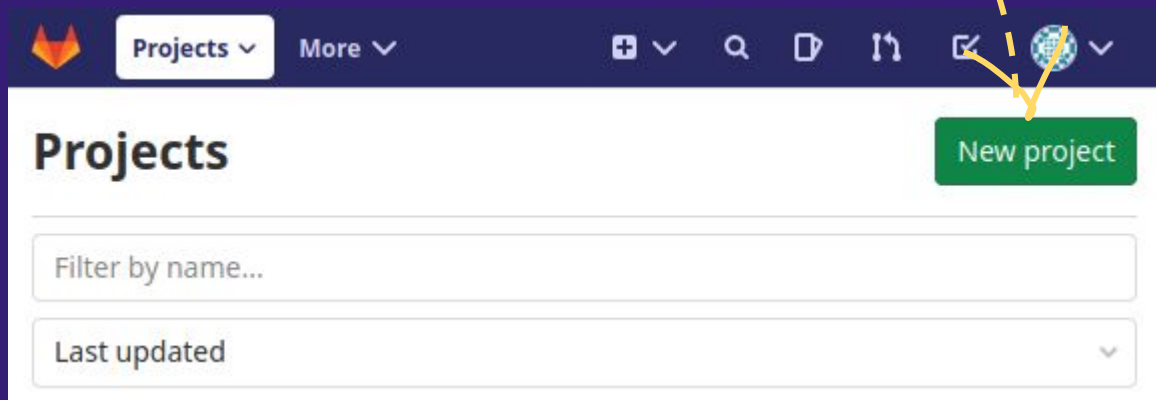
GitLab



GitHub

Ligando un repo local con uno remoto

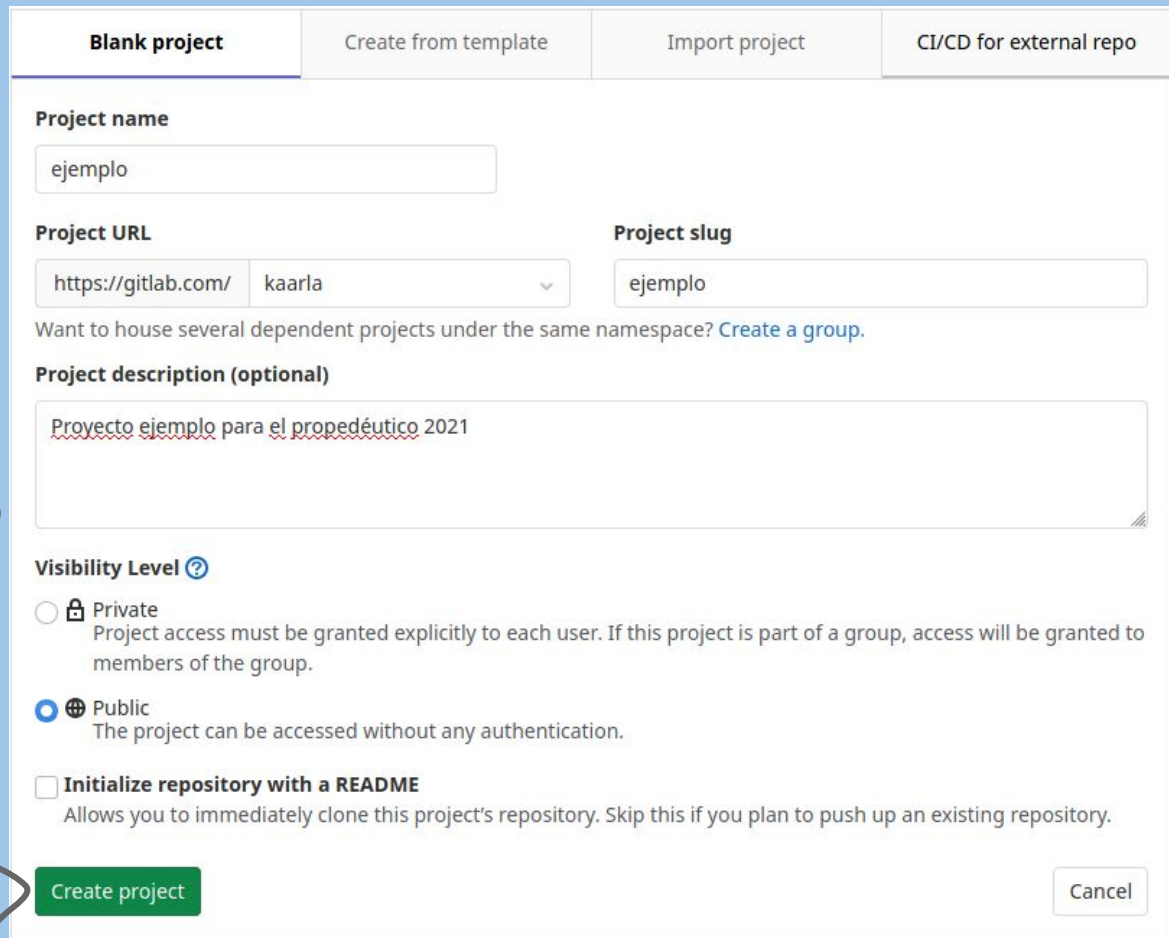
1. Crea una cuenta en gitlab.com
2. Da clic en New Project o Nuevo Proyecto



Ligando un repo local con uno remoto

3. Asigna un nombre y el resto de la configuración

4. Da clic en Crear Proyecto o Create Project



Blank project Create from template Import project CI/CD for external repo

Project name
ejemplo

Project URL **Project slug**
https://gitlab.com/ kaarla ejemplo

Want to house several dependent projects under the same namespace? [Create a group](#).

Project description (optional)
Proyecto ejemplo para el propedéutico 2021

Visibility Level

☐ **Private**
Project access must be granted explicitly to each user. If this project is part of a group, access will be granted to members of the group.

☒ **Public**
The project can be accessed without any authentication.

☐ **Initialize repository with a README**
Allows you to immediately clone this project's repository. Skip this if you plan to push up an existing repository.

Create project Cancel

Ligando un repo local con uno remoto

5. Ejecuta el comando git remote add

6. Ejecuta el comando git push con los parámetros correspondientes

🕒 Project 'ejemplo' was successfully created.

Command line instructions

You can also upload existing files from your computer using the instructions below.

Git global setup

```
git config --global user.name "Karla García"
git config --global user.email "sgakarla@ciencias.unam.mx"
```

Create a new repository

```
git clone https://gitlab.com/kaarla/ejemplo.git
cd ejemplo
touch README.md
git add README.md
git commit -m "add README"
git push -u origin master
```

Push an existing folder

```
cd existing_folder
git init
git remote add origin https://gitlab.com/kaarla/ejemplo.git
git add .
git commit -m "Initial commit"
git push -u origin master
```

Push an existing Git repository

```
cd existing_repo
git remote rename origin old-origin
git remote add origin https://gitlab.com/kaarla/ejemplo.git
git push -u origin --all
git push -u origin --tags
```

Comando push

Para usar el comando push, debemos indicar cuál es el destino en el que queremos subir nuestros cambios. Por ejemplo al agregar un nuevo commit en master lo haríamos de la siguiente forma:

```
$ git push origin master
```

origin indica el alias del repositorio remoto al que está ligado el local.

master es el nombre de la rama.

Ligando un repo local con uno remoto

Finalmente.

Actualiza el sitio de tu repositorio y verás todos tus cambios.

master

ejemplo / +

History


Find file

Web IDE

↓


↓



Clone



creación de segundo archivo en master
Karla García authored 6 minutes ago

09679b2f



Name	Last commit	Last update
 archivo2.txt	creación de segundo archivo en master	6 minutes ago
 nuevoArchivo.txt	creación de archivo para ejemplo	4 days ago

Visualización del historial

Adicionalmente, Gitlab te permite ver una representación gráfica del historial de versiones, en las que puedes ver los commits en los que nacen bifurcaciones y rastrear gráficamente los cambios de un proyecto.

