

3G. Sesión Tres

GitHub

Alfredo González Gaviña, Walter Alejandro Moreno Ramírez, María Susana Ávila García.
Taller 3G
DEM Yuriria 07/03/2018

Mi propio clon.

Clonaremos nuestro primer repositorio de github usando el siguiente comando:

```
$ git clone <url>
```

```
$ git clone https://github.com/Arthyom/Taller-3G.git
```

Construyamos un mejor software, juntos



GitHub Es una **plataforma de desarrollo colaborativo** para alojar proyectos utilizando el sistema de control de versiones Git. **GitHub aloja tu repositorio** y te brinda **herramientas** muy útiles para el **trabajo en equipo**, además puedes **contribuir a mejorar el software de los demás**.

¿Qué Wikis

herramientas


proporciona?

- Sistemas de seguimientos de problemas.
- Herramientas de revisión.
- Visores de ramas.



En línea.

[<https://github.com/login>]



Sign in to GitHub


Username or email address

Password

Forgot password?


Sign in


New to GitHub? [Create an account.](#)


 [Features](#) [Business](#) [Explore](#) [Marketplace](#) [Pricing](#) [Sign in](#) or [Sign up](#)

Join GitHub

The best way to design, build, and ship software.

 Step 1:
Create personal account

 Step 2:
Choose your plan

 Step 3:
Tailor your experience

Create your personal account

Username

This will be your username. You can add the name of your organization later.

Email address

We'll occasionally send updates about your account to this inbox. We'll never share your email address with anyone.

Password

Use at least one lowercase letter, one numeral, and seven characters.


By clicking on "Create an account" below, you are agreeing to the [Terms of Service](#) and the [Privacy Policy](#).


Create an account


You'll love GitHub

Unlimited collaborators

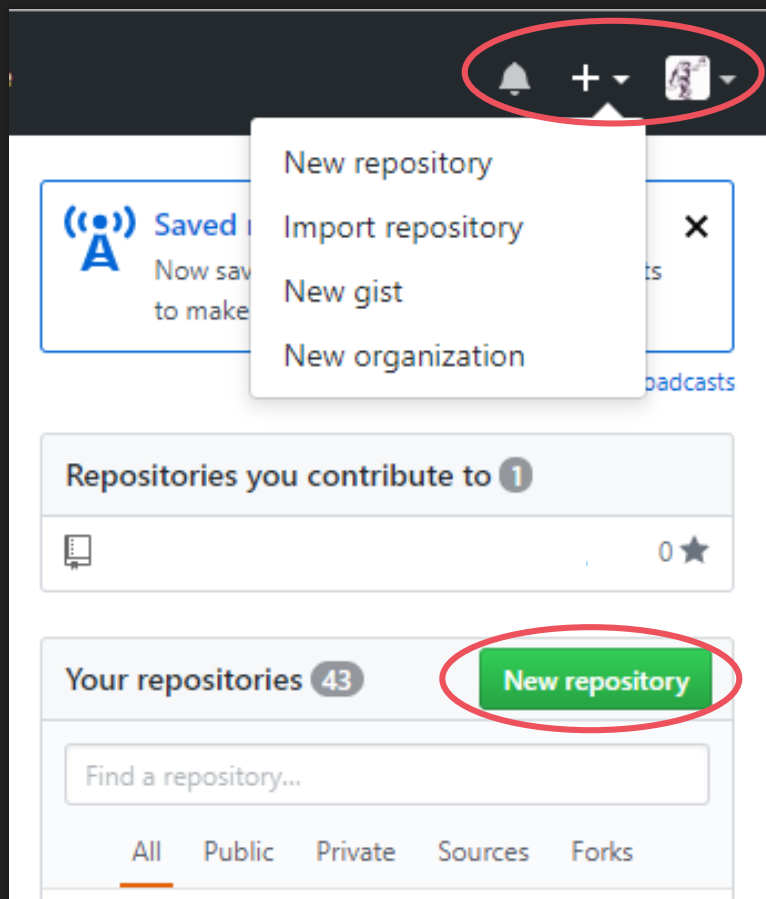
Unlimited public repositories

 Great communication

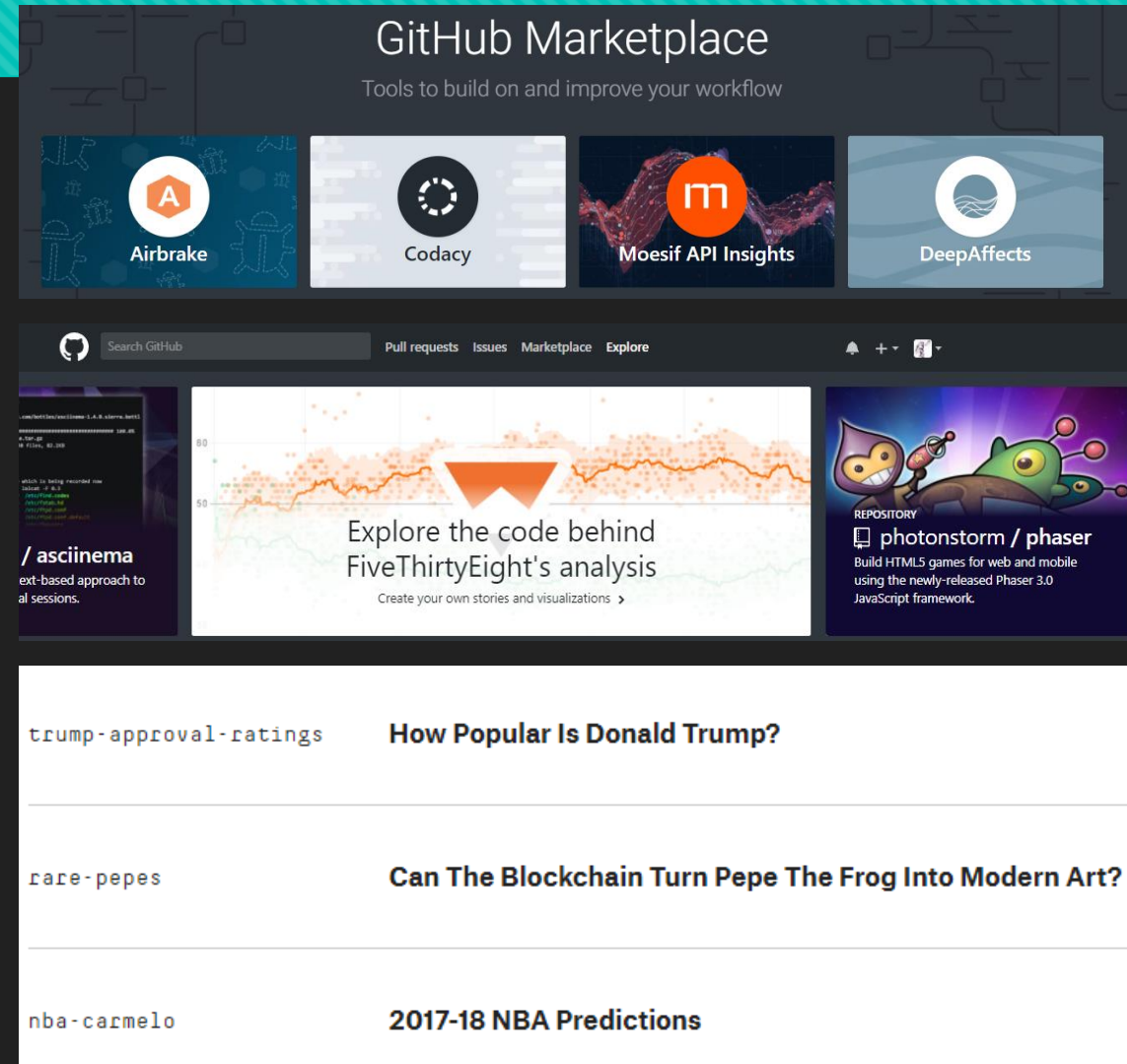
 Frictionless development

 Open source community

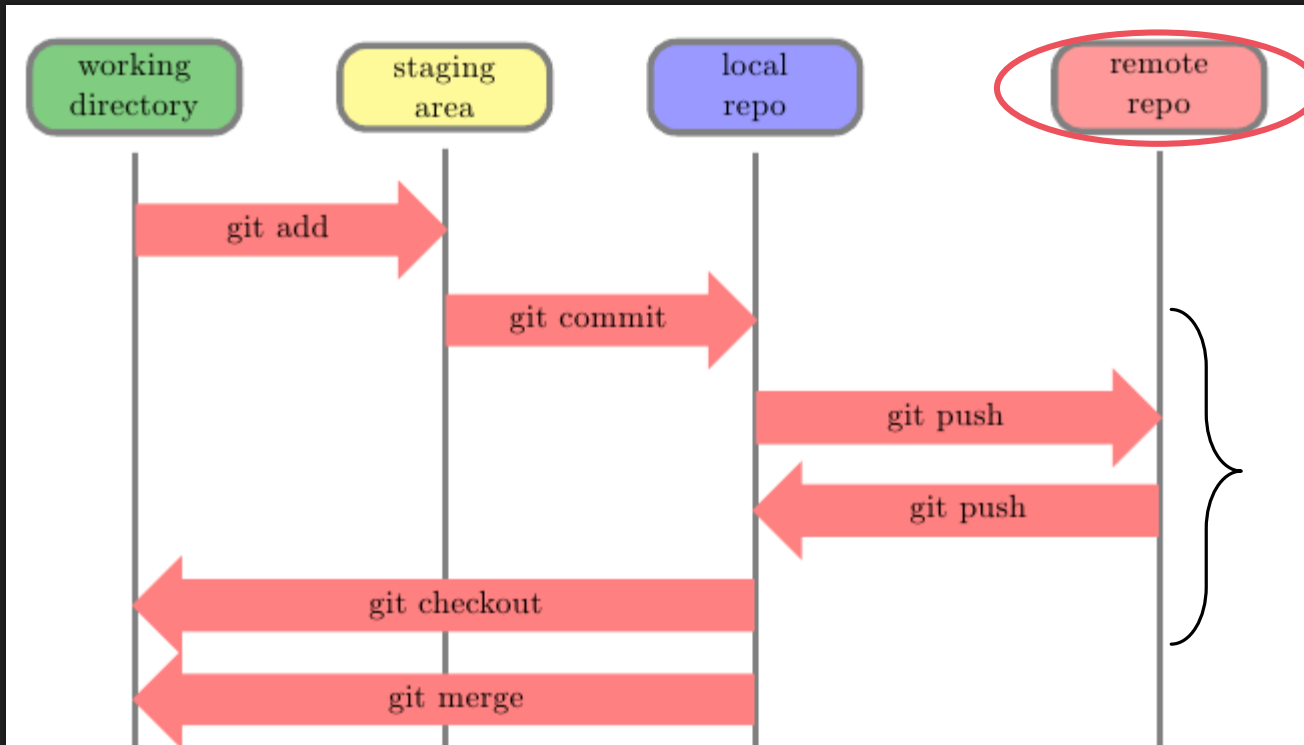
¿Qué sigue?



GitHub se centra en los repositorios y para comenzar a trabajar con github debes crear al menos un repositorio. Si no dispones de un repositorio tan solo podrás observar y tal vez contribuir a los proyectos de alguien más.



Jalar y Empujar



Para poner en línea los cambios locales que hemos trabajado en nuestro repositorio local primero **necesitamos** un “pedacito” de internet en **donde almacenar esos cambios**, este espacio es conocido como “**repositorio remoto**” o “**remoto**”. Después debemos tomar nuestros cambios y “empujarlos” hasta este espacio para que nuestros compañeros de proyecto puedan visualizarlos y posteriormente ellos pueda “Jalar” esos cambios y fusionarlos con su repositorio local.

Jalar y Empujar

```
$ git remote add <nombreRemoto> <URL>
```

```
$ git push -u <nombreRemoto> <nombreRama>
```

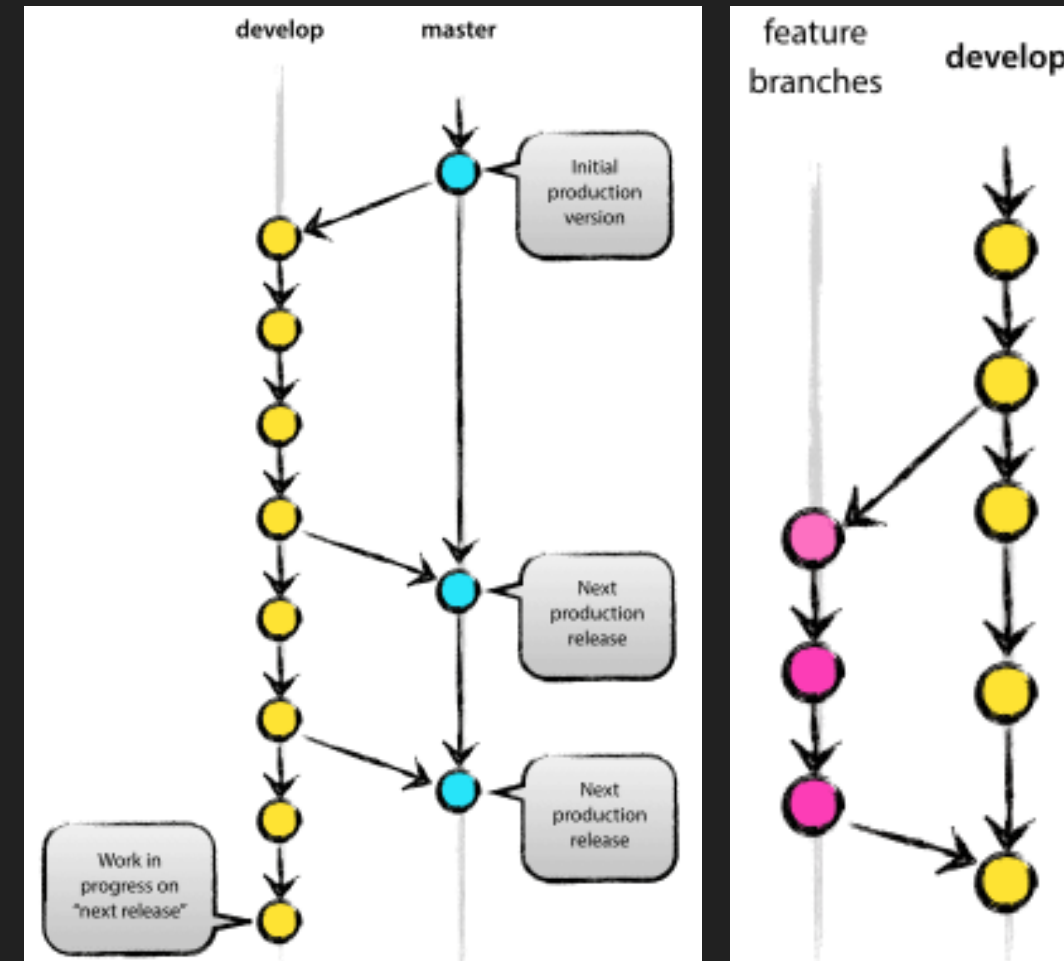
```
$ git pull <nombreRemoto> <nombreRama>
```


El modelo “*Feature Branch*”

En enero de 2010 **Vincent Driessen** creó un flujo de trabajo donde propone una serie de “**reglas**” para organizar el trabajo del equipo dentro de un repositorio, estas reglas se conocen como “**A successful Git branching model**” y ayudan a evitar el caos al trabajar con ramas dentro de un repositorio.

Master: Esta rama contiene la versión en producción. Cada vez que se incorpora algo a master tenemos una nueva versión.

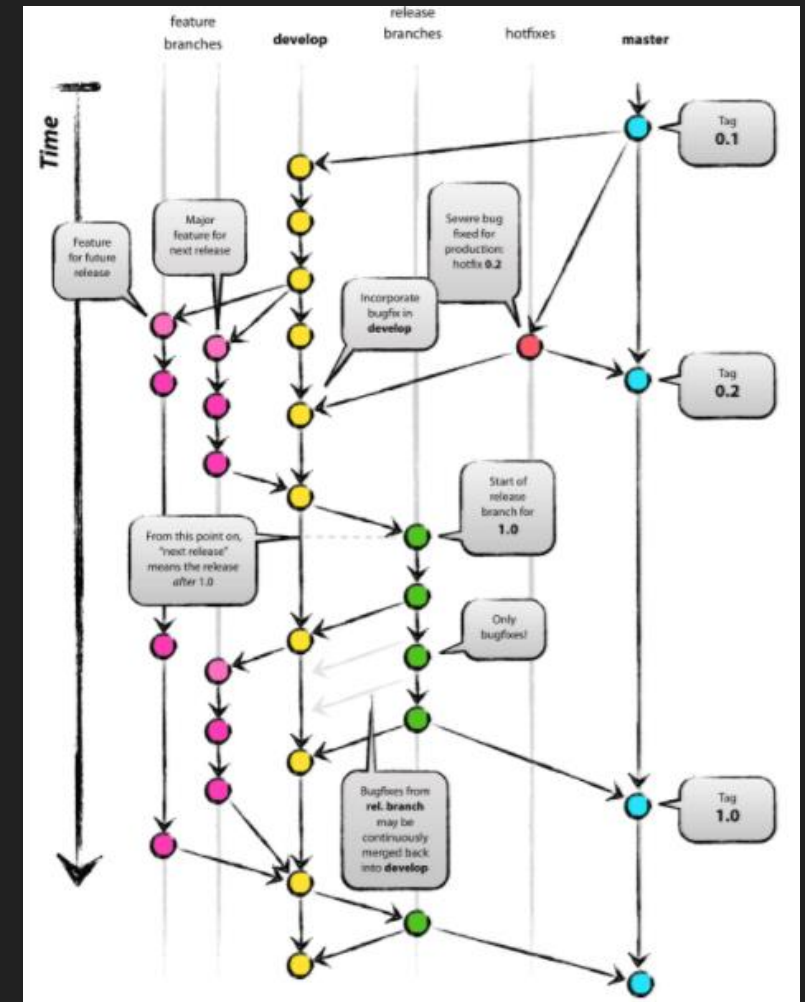
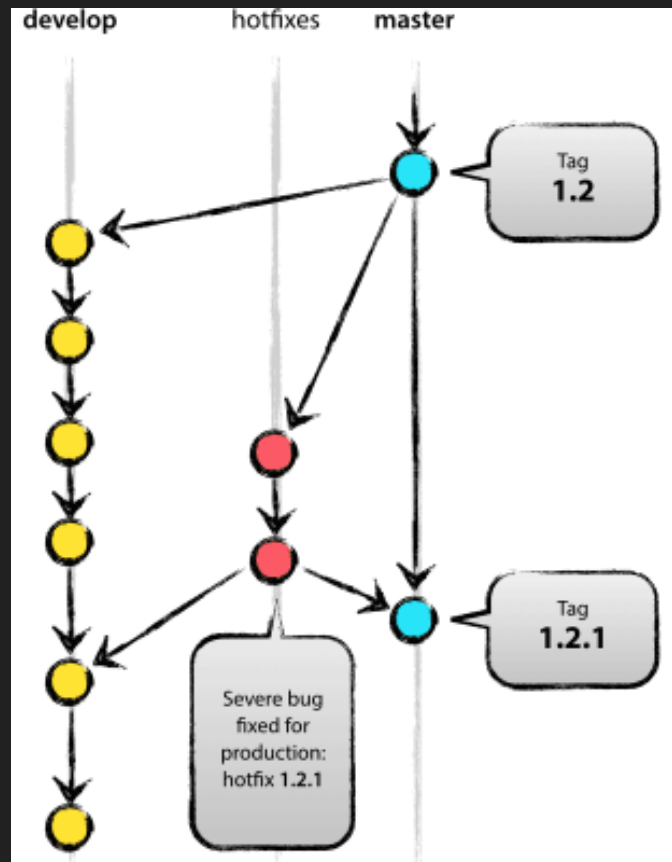
- **Develop:** En esta rama está el código que conformará la siguiente versión planificada del proyecto.
- **Feature o Topic Branches:** Se usan para desarrollar nuevas características de la aplicación.



El modelo “*Feature Branch*”

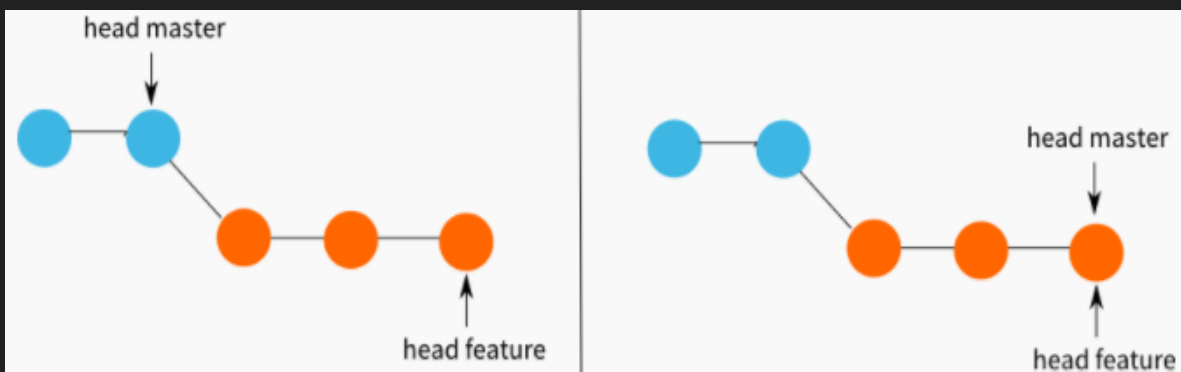
hotfix-* o **release-*** : Son cambios rápidos e imprevistos que se hacen sobre las versiones que van a salir o que ya están en producción, por lo general reparar problemas o detalles menores sin alterar la versión en producción.

Fuente [<http://nvie.com/posts/a-successful-git-branching-model/>]



La naturaleza del problema.

Fuente [<https://styde.net/ramas-y-resolucion-de-conflictos-en-git/>]

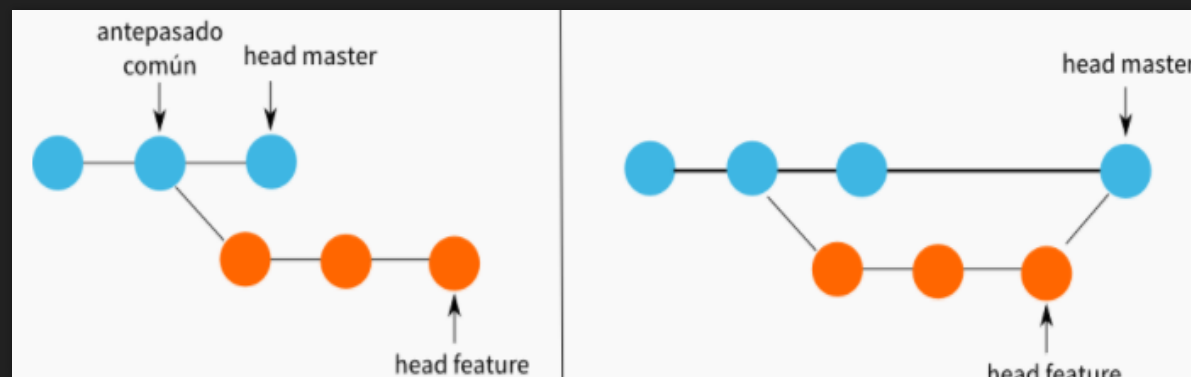


Fast-Foward

Cuando la **rama de destino no tiene ningún commit nuevo** y **el ultimo cambio es el antecesor directo** de los commits de otra rama entonces solamente se agregan los cambios que fueron hechos hacia “adelante”

Estrategia Recursiva

Cuando la **rama de destino tiene commits nuevos** y **el ultimo cambio NO ES ANCESTRO DIRECTO** de los commits de otra rama entonces se hace una fusión “**a tres bandas**” o “**por estrategia recursiva**”



Si los mundos chocan...



Git puede decidir como realizar una fusión, cuando es incapaz de esto ocurre un “**conflicto de fusión**” o “**colisión**”. El usuario decidirá cuales cambios son los correctos. Las colisiones ocurren cuando la rama actual y la rama a fusionar intentan modificar la misma parte del contenido, normalmente estos conflictos se producen cuando se desarrolla ignorando el modelo **feature branch**.

```
<!DOCTYPE HTML>
<html>
  <head>
<<<<<<< HEAD
    <title>Nuevo Titulo</title>
=====
    <title>Nuevo Titulo para la web</title>
>>>>>>> contenido
  </head>
  <body>
    <p>Contenido de la web</p>
    <p>Nuevo párrafo de la página</p>
  </body>
</html>
```

No todo esta perdido.

```
$ git merge --abort
```

```
$ git merge -s recursive -X theirs rama-a-fusionar
```

```
git merge -s recursive -X ours rama-a-fusionar
```

Ejercicio Practico y Preguntas.