

# NODE.JS

# FUENTE DE CONTROL

# GIT

wawiwa



# Fuente de control

1. Un sistema que guarda todo el código del proyecto
2. Contiene cambios históricos a su código y documentación
  - a. Guarda todas las versiones que publicamos
3. Permite a múltiples programadores trabajar en el código
  - a. Puedes ver quien es responsable para cada parte del código
  - b. Previene colisiones y ayuda a lidiar con ellas
4. Te permite simultáneamente desarrollar diferentes capacidades

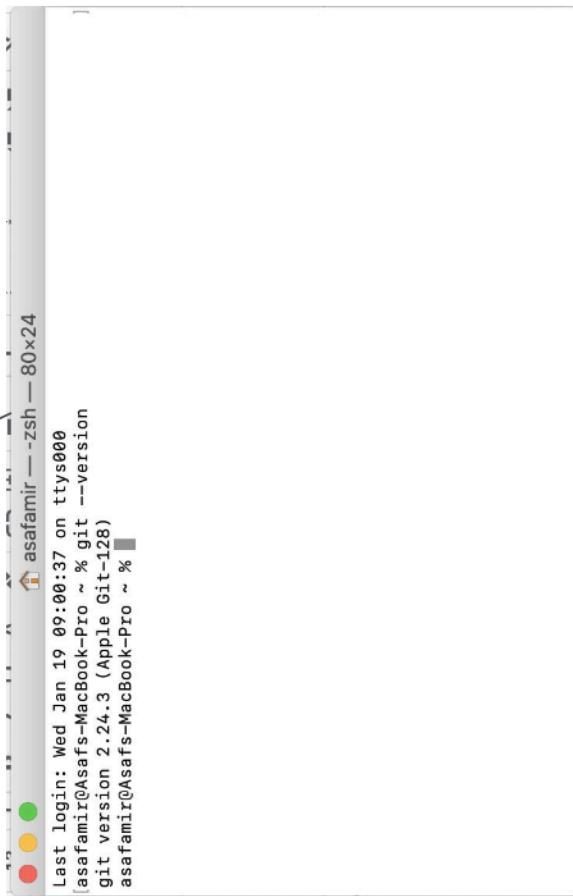
<https://git-scm.com>



# Git -- versión

Ve a la terminal y verifica que git ya na siao instalado

Escribe en la terminal : `git --version`



```
Last login: Wed Jan 19 09:00:37 on ttys000
asafamir@Asafs-MacBook-Pro ~ % git --version
git version 2.24.3 (Apple Git-128)
asafamir@Asafs-MacBook-Pro ~ %
```

A screenshot of a terminal window titled "asafamir — zsh — 80x24". The window shows the user's last login details and the command "git --version" being run. The output indicates that git version 2.24.3 is installed on the system.

# Instalar Git

Git es un sistema de control de versiones para rastrear cambios en cualquier conjunto de archivos y coordinar el trabajo entre programadores durante el desarrollo de software.

Antes de instalar la CLI de heroku, también necesita instalar git en su computadora.

<https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>

A continuación, abre la terminal y mete config git con **tus propias** credenciales:

```
git config --global user.name "John Doe"
```

```
git config --global user.email johndoe@example.com
```

# Trabajando con git

¿Cómo trabajar con fuente de control?  
Usando git y github

## ¿Por qué es bueno trabajar con git?

- Supón que desarrollamos un sistema y le queremos agregar alguna capacidad y cambiamos algo en el código y ahora dejó de funcionar
- Agregar una nueva funcionalidad al sistema
- ¿Qué se puede hacer?

# ¿Por qué es bueno trabajar con git?

Si trabajamos con git podemos manejar versiones e ir atrás a versiones previas en el código.

Ejemplo:

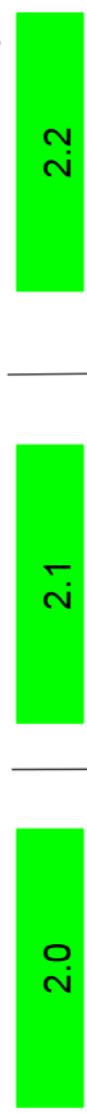
Empezamos a trabajar en un nuevo módulo en el código y encontramos un “bug” en el código y no está funcionando. Podemos actualizar el código en una versión atrás y regresar al código del empleado, al mismo tiempo corregir el código y actualizarlo de nuevo.

## ¿Por qué es bueno trabajar con git?

Usualmente programando proyectos, más de un programador trabaja en el código. Usando git podemos sincronizar entre todos los programadores.

# Manejo de versiones

Piensa en las versiones de desarrollo del sistema operativo Android.



2.2  
Android : 2.21

Android : 2.22

Android : 2.23

Android : 2.11

Android : 2.12

Android : 2.13

Android : 2.01

Android : 2.02

## Fuente de control

- Un sistema que contiene todo el código del proyecto
- El sistema contiene la historia de los cambios de todo el código, incluyendo documentación.
- Varios programadores pueden trabajar en el mismo sistema y desarrollar diferentes productos simultáneamente

**wawiwa**

# CONCEPTOS BÁSICOS DE FUENTE DE CONTROL

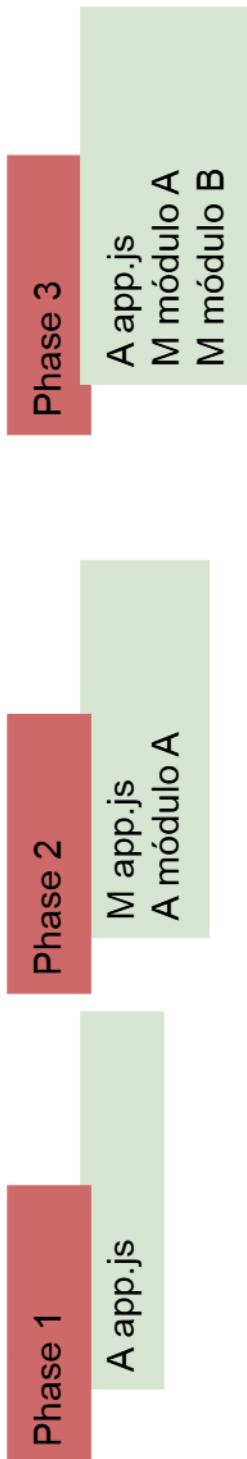


# Repositorio

- Para cada proyecto se va a crear un repositorio
- El repositorio va a contener la historia de cambios en el código. Que incluye la operación de versiones,ejes paralelos de desarrollo

# changeset

Colección de cambios entre dos subtareas  
Los cambios pueden ser - agregar, actualizar, editar o borrar archivos.



- Cada conjunto de cambios tiene un número llamado revisión y es único:
  - Y cada revisión tiene un nombre llamado tag

# Repositorio

1. Código manejado
2. Para cada proyecto - vamos a crear un **Repositorio**
  - a. En una compañía con 10 proyectos - Vamos a tener 10 repositorios
3. El **repositorio** contiene la historia de cambios en el código
  - a. Todos los cambios (borrados / creados / cambiados) en archivos
  - b. Quién hizo el cambio
  - c. Versiones publicadas
  - d. Ejes de desarrollo pasadas y presentes (**branches** - “ramificaciones”)

# Tipos de fuentes de control

1. Hay dos tipos de Fuentes de Control:
  1. Fuente de Control centralizado
    - a. Por ejemplo: cvs, svn
    - b. Toda la historia de cambios son guardados en el servidor (y solo ahí)
    - c. Cada operación será ejecutada directamente frente al servidor
    - d. Este trabajo requiere de nuestra parte una conexión activa en todo momento.
      - i. Cada acción que es ejecutada frente a la red
      - ii. Esto puede alentar el trabajo

# Tipos de fuentes de control

## 2. Fuente de control distribuida

- Cada usuario tiene una copia entera del repositorio
- Trabaja frente una base de datos local en la PC
- Al final del trabajo, actualiza en el servidor
  - Enviar a un servidor central
- Para recibir los cambios - actualiza del servidor
  - Ejecuta un “pull” del servidor principal
- Jugadores principales:
  - git y el servicio de almacenamiento de código Github
  - Los servicios de almacenamiento de código Mercurial y Bitbucket

# git

- Versión de control distribuida (Distributed VC)
- El uso se realiza mediante la herramienta git.
- Existen un número de interfaces gráficas desarrolladas
- Interfaz gráfica recomendada:
  - SourceTree (Recomendada!)
  - Tortoise git
- Existe una integración de git con la mayoría de los IDEs (eclipse, visual studio,...)

# Comandos comunes

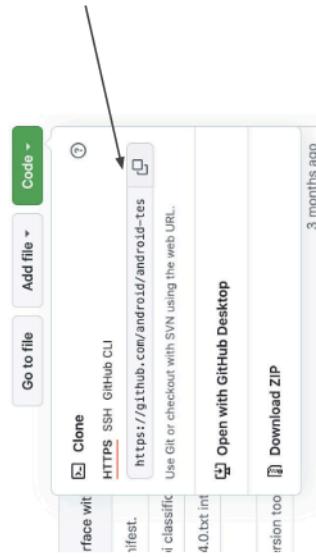
1. Clone
2. Commit
3. Add
4. Commit
5. Push
6. Pull
7. Update
8. merge
9. branch

# Comandos comunes

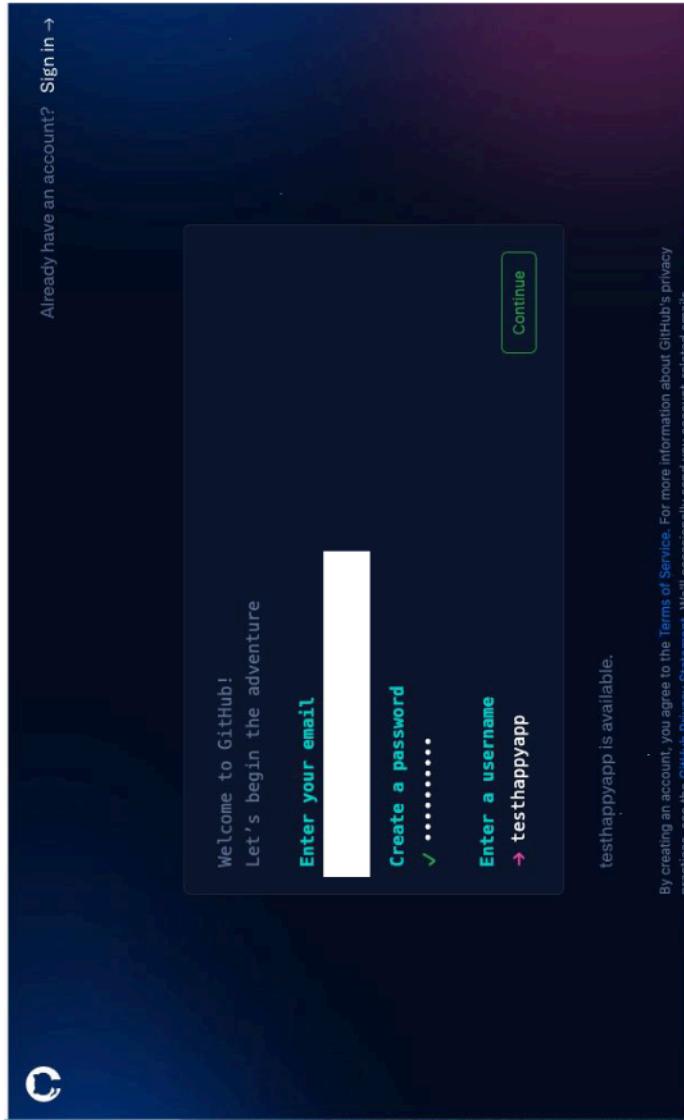
- touch <filename> : crea archivo
- rm <filename>: borra archivo
- **git add .**
- **git commit -m "first"**
- git diff - diferencias con la última versión
- git status
- git log

# git clone

- Cada usuario tiene una copia completa del repositorio
- La operación “clone” crea la copia en tu computadora
- El comando creará una carpeta en la computadora y dentro de ella habrá una imagen reflejada completa de todo el código más reciente.
- Estructura del comando:
  - `git clone https://user@server.com/repo-name`



# Register en github.com



testhappyapp is available.

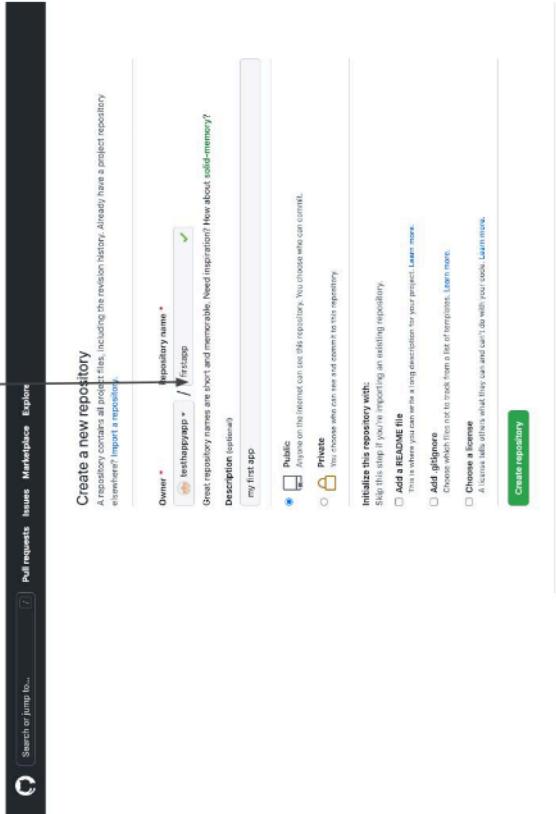
# Crea tu primer repositorio github

The screenshot shows the GitHub interface with a dark theme. At the top, there's a navigation bar with 'Search or jump to...', 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. Below this, a large callout box on the left says 'Create your first project' and 'Ready to start building? Create a repository for a new idea or bring over an existing repository to keep contributing to it.' It features two buttons: 'Create repository' (green) and 'Import repository' (grey). To the right of this, a main content area has a heading 'Learn Git and GitHub without any code!' with a 'Start a project' button. Another section titled 'Introduce yourself' explains that creating a README is the easiest way to introduce yourself on GitHub. A list of 6 items follows:

- 1 - 🌟 Hi, I'm @testhappyapp
- 2 - 🚀 I'm interested in ...
- 3 - 🌱 I'm currently learning ...
- 4 - 💬 I'm looking to collaborate on ...
- 5 - 📩 How to reach me ...
- 6 - 🛡️

At the bottom right of the main content area is a 'Dismiss this' link and a green 'Continue' button.

# First app



description

public

The screenshot shows a GitHub repository page for 'testhappyapp / firstapp'. The top navigation bar includes links for Pull requests, Issues, Marketplace, and Explore. A search bar at the top left contains the placeholder 'Search or jump to...'. On the right side of the header are buttons for Pin, Unwatch, Fork, and Settings.

**Quick setup — if you've done this kind of thing before**

Set up in Desktop or HTTPS SSH https://github.com/testhappyapp/firstapp.git

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

**...or create a new repository on the command line**

```
echo "# firstapp" >> README.md
git init
git add README.md
git commit -m "First commit"
git branch -M main
git remote add origin https://github.com/testhappyapp/firstapp.git
git push -u origin main
```

**...or push an existing repository from the command line**

```
git remote add origin https://github.com/testhappyapp/firstapp.git
git branch -M main
git push -u origin main
```

**...or import code from another repository**

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code

## Crea una nueva carpeta en Visual Studio

1. Desde Visual Studio crea una nueva carpeta y selecciona el nombre de la carpeta.
2. Desde la terminal, cd a la carpeta

# En la terminal escribe el comando

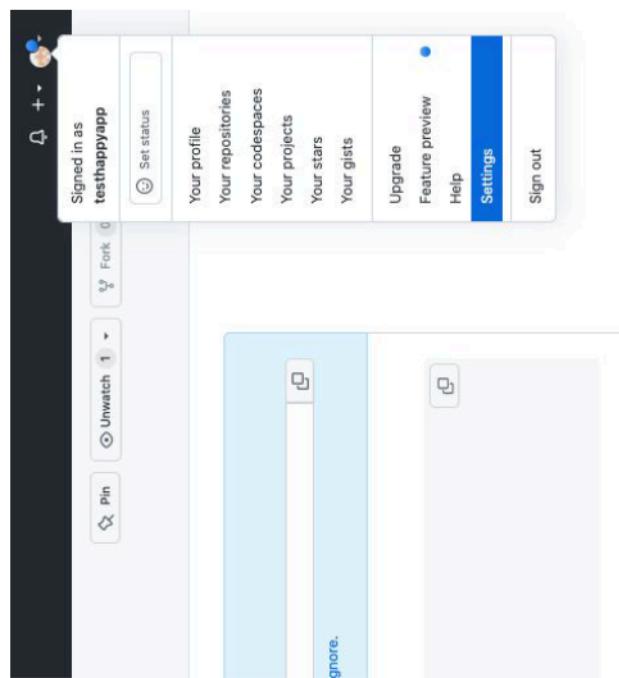
```
echo "# test" >> README.md  
git init  
git add README.md  
git commit -m "first commit"  
git branch -M main  
git remote add origin  
https://github.com/testhappyapp/first\_app.git  
git push -u origin main
```

README.md  
archivo creado

✓ chap17-firstapp  
| README.md

Si no puedes hacer push, tienes que crear un token

# Create a token - ve a GitHub



# Crea un token

New personal access token

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

Note

test

What's this token for?

Expiration \*

30 days

The token will expire on Tue, Mar 1 2022

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes.](#)

<input type="checkbox"/> <b>repo</b>	Full control of private repositories
<input type="checkbox"/> <b>repo_status</b>	Access comment status
<input type="checkbox"/> <b>repo_deployment</b>	Access deployment status
<input type="checkbox"/> <b>public_repo</b>	Access public repositories
<input type="checkbox"/> <b>read_private</b>	Access repository invitations
<input type="checkbox"/> <b>security_events</b>	Read and write security events
<input checked="" type="checkbox"/> <b>workflow</b>	Update GitHub Action workflows
<input checked="" type="checkbox"/> <b>write_packages</b>	Upload packages to GitHub Package Registry
<input type="checkbox"/> <b>read_packages</b>	Download packages from GitHub Package Registry
<input checked="" type="checkbox"/> <b>delete_packages</b>	Delete packages from GitHub Package Registry
<input checked="" type="checkbox"/> <b>admin:org</b>	Full control of orgs and teams; read and write org projects
<input type="checkbox"/> <b>read:org</b>	Read and write org and team membership, read and write org projects
<input type="checkbox"/> <b>write:org</b>	Read org and team membership, read org projects
<input type="checkbox"/> <b>read_org</b>	
<input checked="" type="checkbox"/> <b>admin:public_key</b>	Full control of user public keys

# Crea un token

Settings / Developer settings

New personal access token

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

Note

test

What's this token for?

Expiration \*

30 days The token will expire on Tue, Mar 1 2022

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes.](#)

<input type="checkbox"/> <b>repo</b>	Full control of private repositories
<input type="checkbox"/> <b>repostatus</b>	Access commit status
<input type="checkbox"/> <b>repo_deployment</b>	Access deployment status
<input type="checkbox"/> <b>public_repo</b>	Access public repositories
<input type="checkbox"/> <b>repomanage</b>	Access repository invitations
<input type="checkbox"/> <b>security_events</b>	Read and write security events
<input checked="" type="checkbox"/> <b>workflow</b>	Update GitHub Action workflows
<input checked="" type="checkbox"/> <b>write_packages</b>	Upload packages to GitHub Package Registry
<input type="checkbox"/> <b>read_packages</b>	Download packages from GitHub Package Registry
<input checked="" type="checkbox"/> <b>delete_packages</b>	Delete packages from GitHub Package Registry
<input checked="" type="checkbox"/> <b>admin:org</b>	Full control of orgs and teams, read and write org projects
<input type="checkbox"/> <b>write:org</b>	Read and write org and team membership, read and write org projects
<input type="checkbox"/> <b>read:org</b>	Read org and team membership, read org projects
<input checked="" type="checkbox"/> <b>admin:public_key</b>	Full control of user public keys

# Copia el token

Some of the scopes you've selected are included in other scopes. Only the minimum set of necessary scopes has been saved.

X

Settings / Developer settings

## Personal access tokens

Generate new token

Revoke all

Tokens you have generated that can be used to access the GitHub API.

Personal access tokens

Make sure to copy your personal access token now. You won't be able to see it again!

✓ ghp\_50xt0YKJ1tQcSP6c1V3MSNCID10w2uR48p0vF 

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

© 2022 GitHub, Inc. Terms Privacy Security Status Docs Contact GitHub Pricing API Training Blog About

# En la terminal escribe el comando

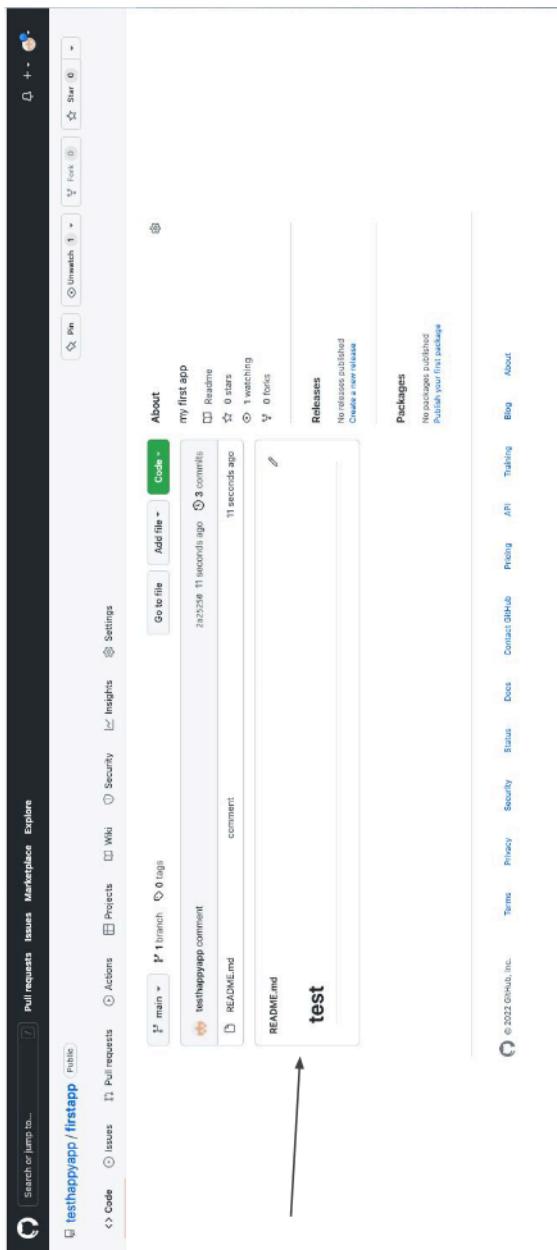
En la terminal :

git remote set-url origin <https://<TOKEN>@github.com/<USERNAME>/<REPOSITORYNAME>> git

```
git remote set-url origin https://ghp_50xt0YkjltqcsP6cIv3MSNDI0w2UR48pQvF@github.com/testhappyapp/firstapp.git  
Now you can...  
git push -u origin main
```

```
asafamir@Asafs-MBP chap17-firstapp % git push -u origin main  
s: 6, done.  
Counting objects: 100% (6/6), done.  
Delta compression using up to 16 threads  
Compressing objects: 100% (2/2), done.  
Writing objects: 100% (6/6), 436 bytes | 436.00 KiB/s, done.  
Total 6 (delta 0), reused 0 (delta 0)  
To https://github.com/testhappyapp/firstapp.git  
 * [new branch]  main -> main  
Branch 'main' set up to track remote branch 'main' from 'origin'.  
asafamir@Asafs-MBP chap17-firstapp % █
```

# Cambio en [github.com](#)



El archivo que  
creamos

# Cambia el contenido en el archivo README.md

```
git add .
git commit -am "second commit"
git push -u origin main
```

```
[asafamir@Asafs-MBP chap17-firstapp % git add .
asafamir@Asafs-MBP chap17-firstapp % git commit -am "second commit"
```

```
[main 7ad3140] second commit
 1 file changed, 1 insertion(+), 1 deletion(-)
asafamir@Asafs-MBP chap17-firstapp % git push -u origin main
```

Enumerating objects: 5, done.

Counting objects: 100% (5/5), done.

Writing objects: 100% (3/3), 263 bytes | 263.00 KiB/s, done.

Total 3 (delta 0), reused 0 (delta 0)

To https://github.com/testhappyapp/firstapp.git

2a25250..7ad3140 main -> main

Branch 'main' set up to track remote branch 'main' from 'origin'.

```
asafamir@Asafs-MBP chap17-firstapp %
```

```
 README.md x
chap17-firstapp > README.md > # test hello world
```

```
1 | # test hello world
```

**testhappyapp / firstapp** Public

<> Code ⚡ Issues 🌐 Pull requests ⏪ Actions 📂 Projects 📖 Wiki

1 branch 0 tags 0 pull requests 0 issues

Go to file Add

testhappyapp second commit 7ad3140 3 minutes ago

README.md

Después haz push a  
github

**Repitamos lo que hicimos**

**en la terminal :**

```
echo "# test" >> README.md
```

**Create readme file**



The screenshot shows a terminal window with several tabs at the top: OUTPUT, DEBUG CONSOLE, and TERMINAL. The TERMINAL tab is active. The command `echo "# test" >> README.md` is typed into the terminal and is being processed. The output shows the command itself followed by the message `[1]: bash`. The terminal interface includes standard navigation keys like arrow keys, a search bar, and a copy/paste button.

```
Asaf's-MacBook-Pro-397:myproject asafmirit$ echo "# test" >> README.md
[1]: bash
```

# En la terminal :

git init  
Inicializa git



The screenshot shows a terminal window with the following text:

```
Asaf's-MacBook-Pro-397:myproject asafamir$ echo "# test" >> README.md
Asaf's-MacBook-Pro-397:myproject asafamir$ git init
Initialized empty Git repository in /Users/asafamir/Desktop/myproject/.git/
Asaf's-MacBook-Pro-397:myproject asafamir$
```

A red arrow points from the text "Inicializa git" in the previous slide to the "git init" command in the terminal output.

## En la terminal :

```
git add README.md
```

Aggrega el archivo solamente en tu computadora

OUTPUT	DEBUG CONSOLE	TERMINAL
		Asafs-MacBook-Pro-397:myproject asafamir\$ git add README.md
		Asafs-MacBook-Pro-397:myproject asafamir\$ git commit -m "first commit"
		[master (root-commit) a52092b] first commit
		1 file changed, 1 insertion(+)
		create mode 100644 README.md
		Asafs-MacBook-Pro-397:myproject

Above the terminal window, there is a red arrow pointing downwards towards the commit message "first commit".

# En la terminal

git commit -m "first commit"

Agrega el conjunto de cambios

The screenshot shows a terminal window with several tabs at the top: OUTPUT, DEBUG CONSOLE, and TERMINAL. The TERMINAL tab is active. The command `git commit -m "first commit"` is entered and executed. A red arrow points to the output line where the commit message is displayed. The terminal also shows the creation of a new file named `README.md`.

```
Asafs-MacBook-Pro-397:myproject asafamir$ git commit -m "first commit"
[master (root-commit) a52092b] first commit
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
Asafs-MacBook-Pro-397:myproject git branch -M main
Asafs-MacBook-Pro-397:myproject
```

git branch -M main

## En la terminal

git branch -M main

```
OUTPUT DEBUG CONSOLE TERMINAL
[master (root-commit) a52092b] first commit
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
Asafs-MacBook-Pro-397:myproject
Asafs-MacBook-Pro-397:myproject
Asafs-MacBook-Pro-397:myproject
```

git branch -M main  
git remote add origin https://github.com/appschool/test.git



## O configúralo con el token

```
" " " " " "
git remote set-url origin https://ghp_50xt0YkjltqcsP6cIw3MSNCDI0w2UR48p0vF@github.com/testhappyapp/firstapp.git
```

# Git status

El comando “git status” muestra todos los cambios que haz hecho desde la versión pasada. Los cambios no van a ser válidos hasta que hagamos el “commit”. El comando “commit” guarda nuestros cambios de git a la computadora.

# Git add

Cuando creamos un archivo debemos usar el comando “git add”. El comando va a agregar todos los archivos.

```
Asafs-MacBook-Pro-397:git-lesson asafamir$ git add index.html  
Asafs-MacBook-Pro-397:git-lesson asafamir$ git status  
On branch main  
Your branch is up-to-date with 'origin/main'.  
  
Changes to be committed:  
(use "git reset HEAD <file>..." to unstage)
```

new file: `index.html`

```
Asafs-MacBook-Pro-397:git-lesson asafamir$ █
```



# Git commit

Cuando terminemos de trabajar en un archivo, podemos y queremos darle a nuestra colección de cambios un nombre. La colección de cambios When we finish working on a file, we can, we want to give the collection of changes a name. El recordatorio de recopilación de cambios es **changeset**

Ejemplo:

```
Changes to be committed:  
(use "git reset HEAD <file>..." to unstage)
```

```
new file: index.html
```

```
Asafs-MacBook-Pro-397:git-lesson asafamir$ git commit -am"description 1"  
[main 660858b] description 1  
 1 file changed, 5 insertions(+)  
  create mode 100644 index.html  
Asafs-MacBook-Pro-397:git-lesson asafamir$ █
```

# Git rm

El comando va a borrar un archivo  
Ejemplo:

```
git rm index.html
```

```
Changes to be committed:  
(use "git reset HEAD <file>..." to unstage)  
  
new file: index.html
```

```
Asafs-MacBook-Pro-397:git-lesson asafamir$ git commit -am "description 1"  
[main 660858b] description 1  
 1 file changed, 5 insertions(+)  
 create mode 100644 index.html  
Asafs-MacBook-Pro-397:git-lesson asafamir$ git rm index.html  
rm 'index.html'  
Asafs-MacBook-Pro-397:git-lesson asafamir$ █
```

# Git log

Ve el historial de cambios  
Ejemplo:

git log

```
Asafs-MacBook-Pro-397:git-lesson asafamir$ git log
commit 660858b1bd0bee6f261c2544f6d7d09a0fbdb12db (HEAD -> main)
Author: yaya games <yayagames100@gmail.com>
Date: Mon Nov 2 07:39:22 2020 +0200

description 1
```

```
commit 0dce20a6a68ac8c5f87ecbd0a457062c1c89c342 (origin/main)
Author: yaya games <yayagames100@gmail.com>
Date: Mon Nov 2 07:33:40 2020 +0200

first commit
```

```
Asafs-MacBook-Pro-397:git-lesson asafamir$ █
```

# Git diff

Muestra los cambios en los archivos

Ejemplo:

```
Asafs-MacBook-Pro-397:git-lesson asafamir$ git diff
diff --git a/index.html b/index.html
index 427f49b..97d009e 100644
--- a/index.html
+++ b/index.html
@@ -1,5 +1,5 @@
<html>
<body>
- hello
+ hello, hi
</body>
</html>
\ No newline at end of file
Asafs-MacBook-Pro-397:git-lesson asafamir$ █
```

## Más comandos

<https://github.com/joshnh/Git-Commands>

wawiwa

¿Preguntas?