

INSTALACIÓN GITHUB

ÍNDICE

1. INSTALACIÓN GIT EN 3 PASOS
2. CREAR CUENTA EN GITHUB
3. INSTALACIÓN VISUAL STUDIO CODE
4. MANDAR UN PROYECTO LOCAL AL REPOSITORIO
5. NOCIONES TEÓRICAS
6. USO DE COMANDOS
7. PRÁCTICA
8. INSTALACIÓN ENTORNO NODE.JS
9. INSTALACIÓN LENGUAJE TYPESCRIPT
10. PROYECTO TYPESCRIPT



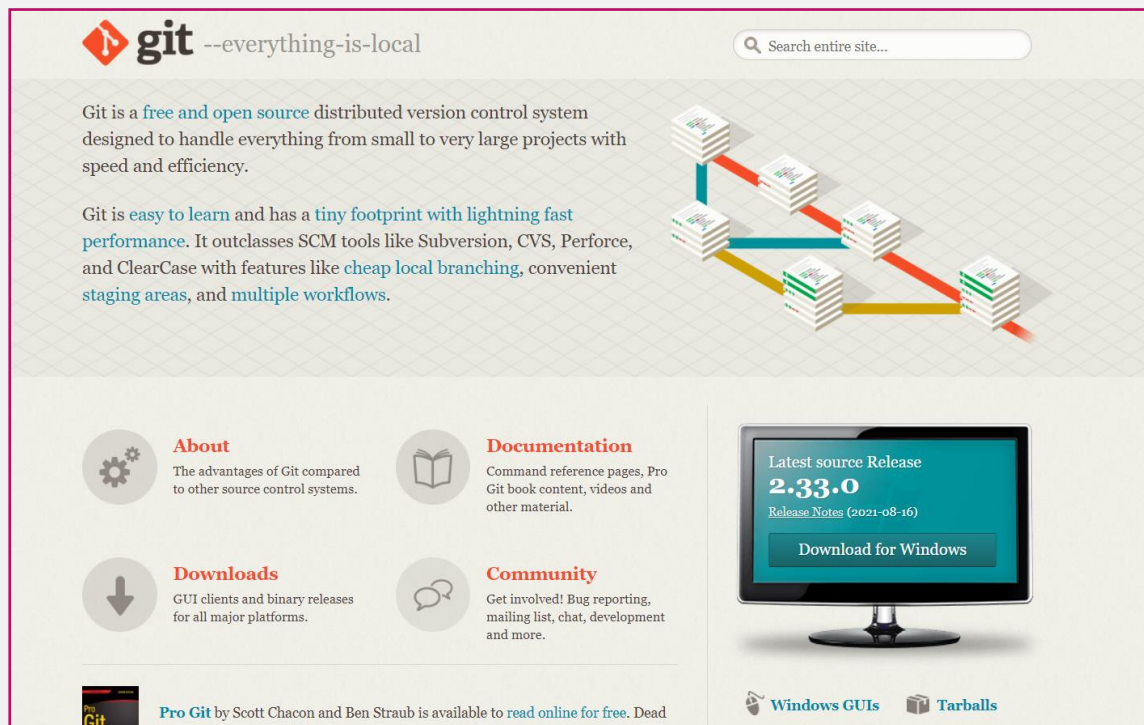
1. INSTALACIÓN GIT EN 3 PASOS

- **PASO 1. DESCARGAR GIT**

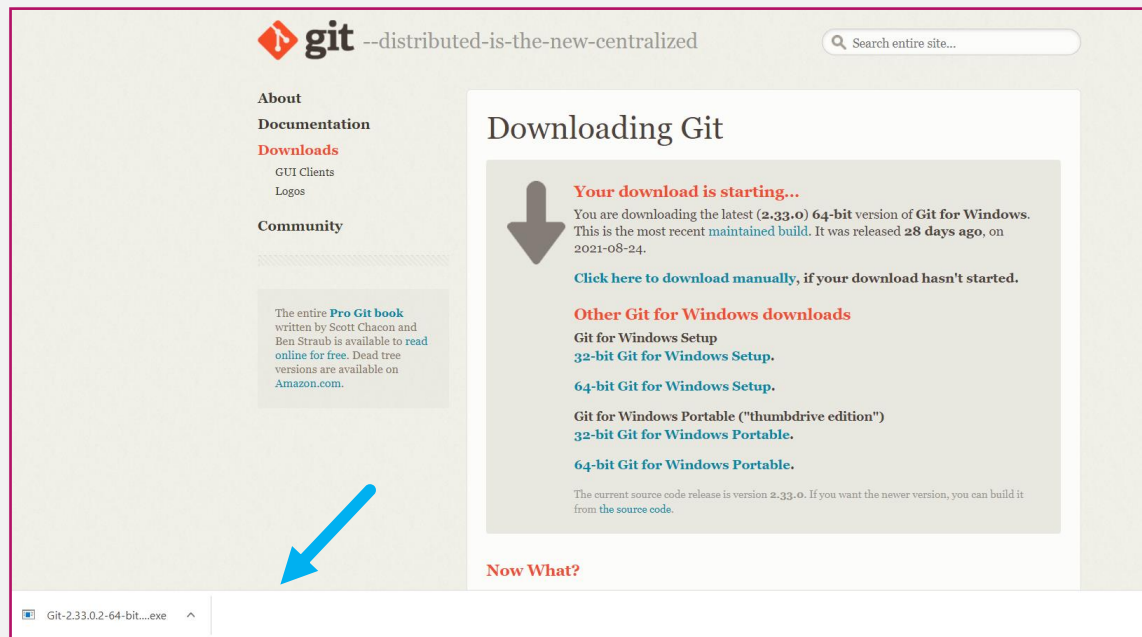
Para comenzar abriremos una nueva ventana en el navegador e iremos a la página de GIT, puedes acceder a ella a través del siguiente enlace:

<https://git-scm.com/>

Nos aparecerá una ventana como esta, donde nos desplazaremos hasta el apartado “[Downloads for Windows](#)”. Una vez clicado la descarga habrá comenzad



- **PASO 2. ARCHIVO EJECUTABLE**



Una vez descargado, el archivo aparecerá en la parte baja izquierda de la pantalla. Lo siguiente que vamos a hacer es abrir este archivo ejecutable.

- **PASO 3. SETUP**

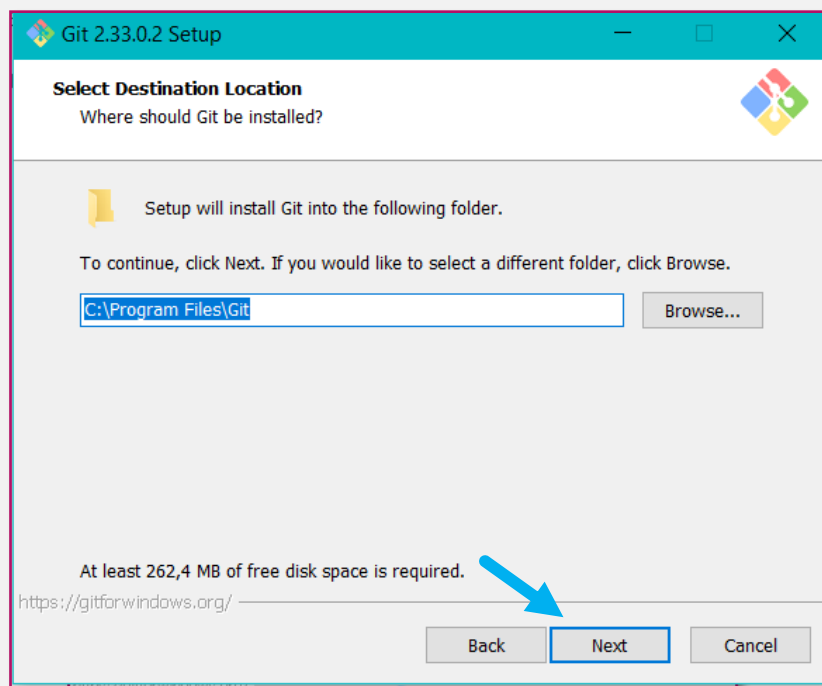
Al inicio preguntará por permisos para ejecutar el instalador, a lo cual debemos responder que sí.



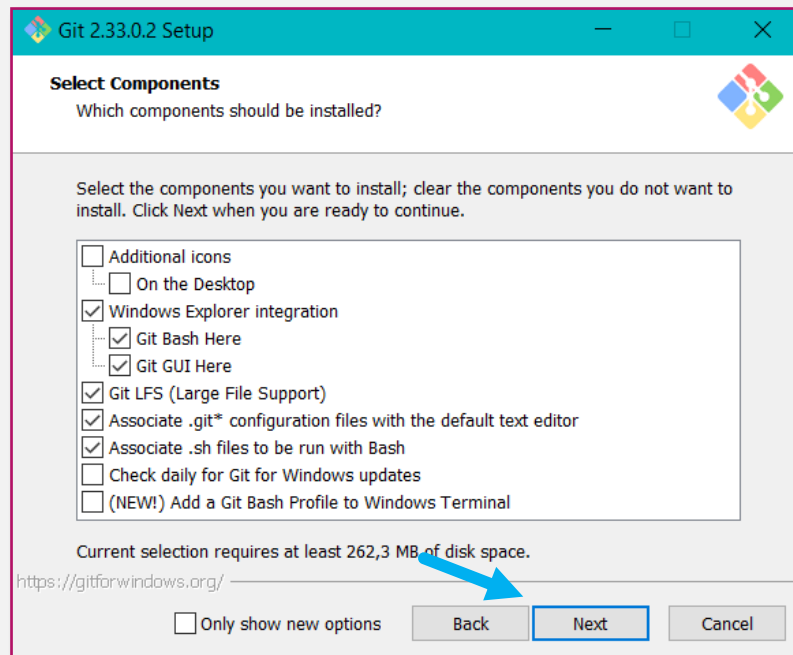
3.1 Pantalla inicial del instalador



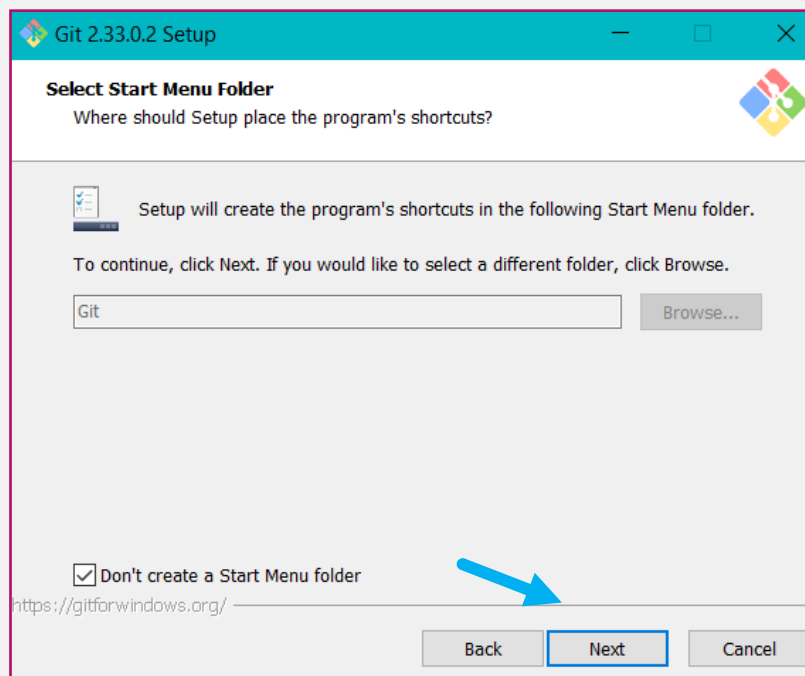
3.2 Ruta donde se instalará Git



3.3 Configuración de componentes que vienen disponibles con Git



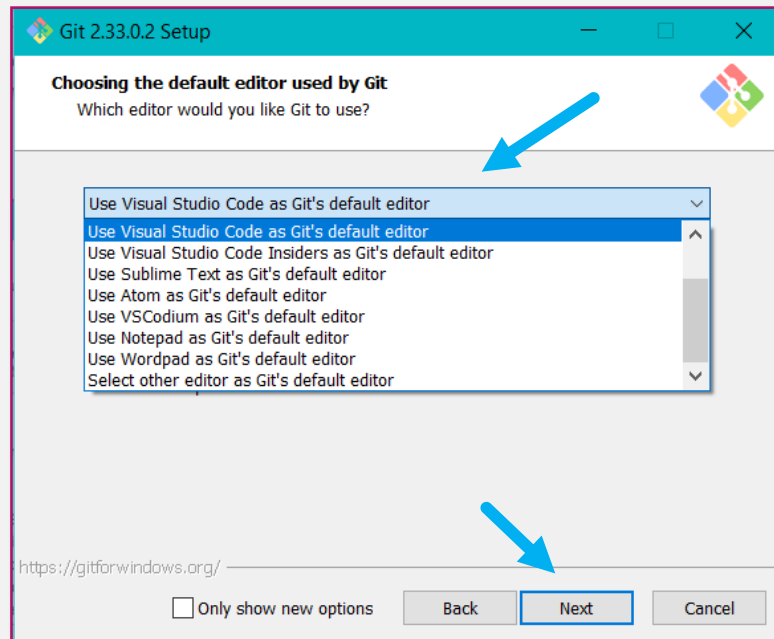
3.4 Nombre con el cual se puede buscar desde la lista de programas



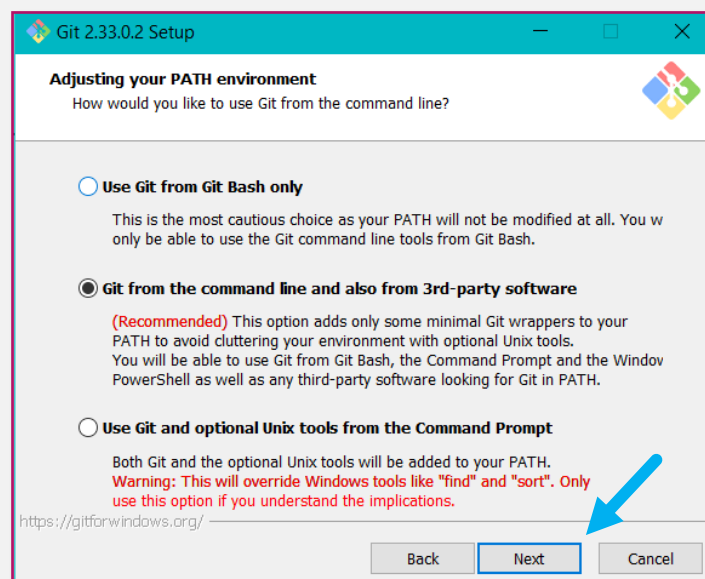
3.5 Editor predeterminado

En esta ventana nos pide que escojamos el editor predeterminado para Git, yo en mi caso usaré la primera opción que es Visual Studio Code, pero usted podrá escoger su editor de confianza, entre ellos están Notepad, Wordpad, Sublime...

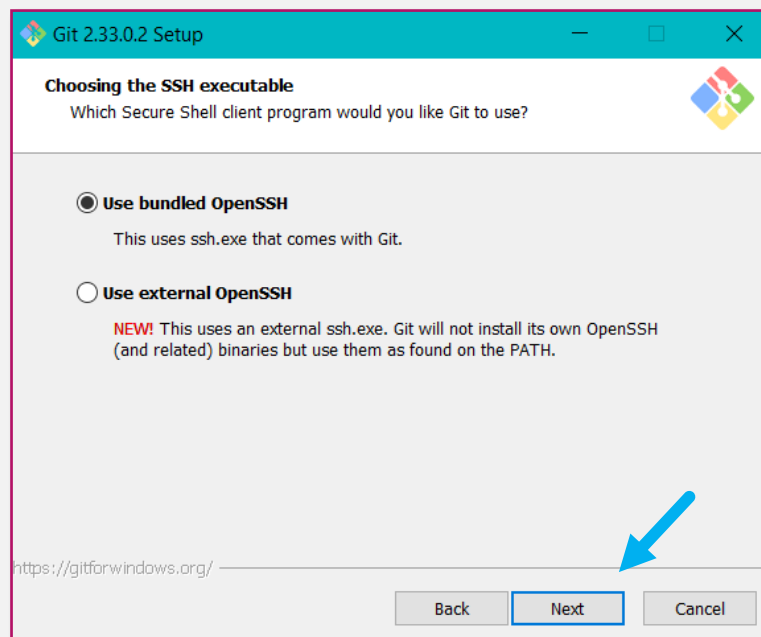
Después de la selección nos mantendremos en clicar "Next".



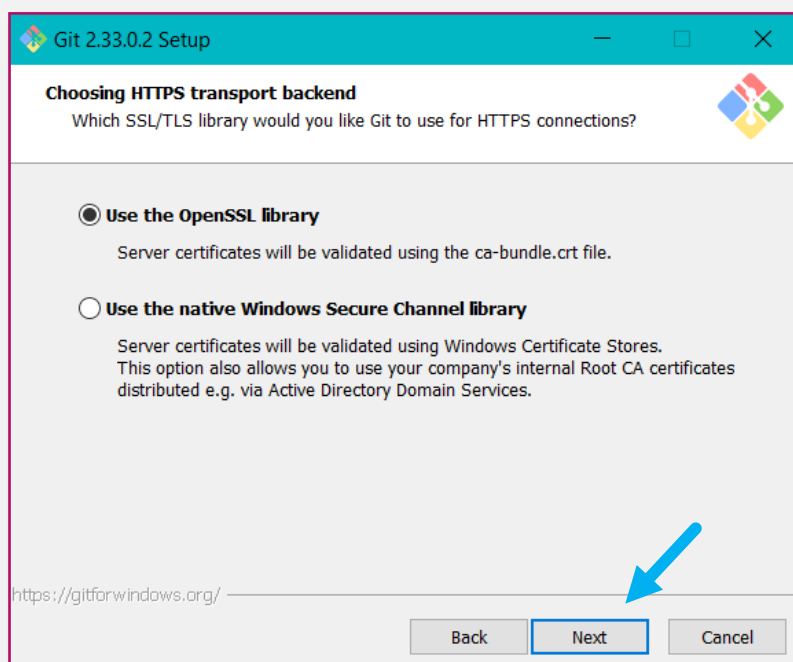
3.6 Terminales desde la cual se puede ejecutar Git



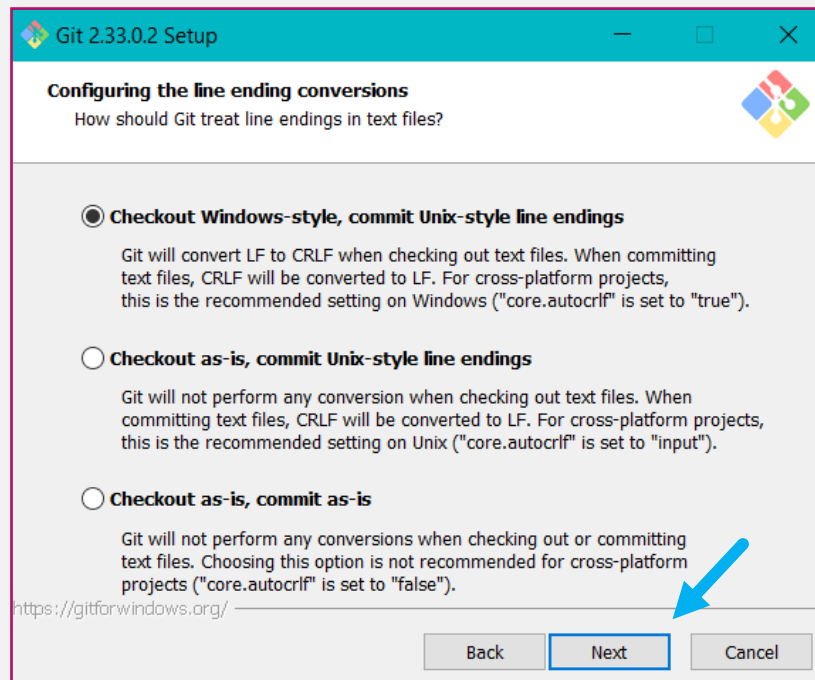
3.7 Escoger el ejecutable SSH



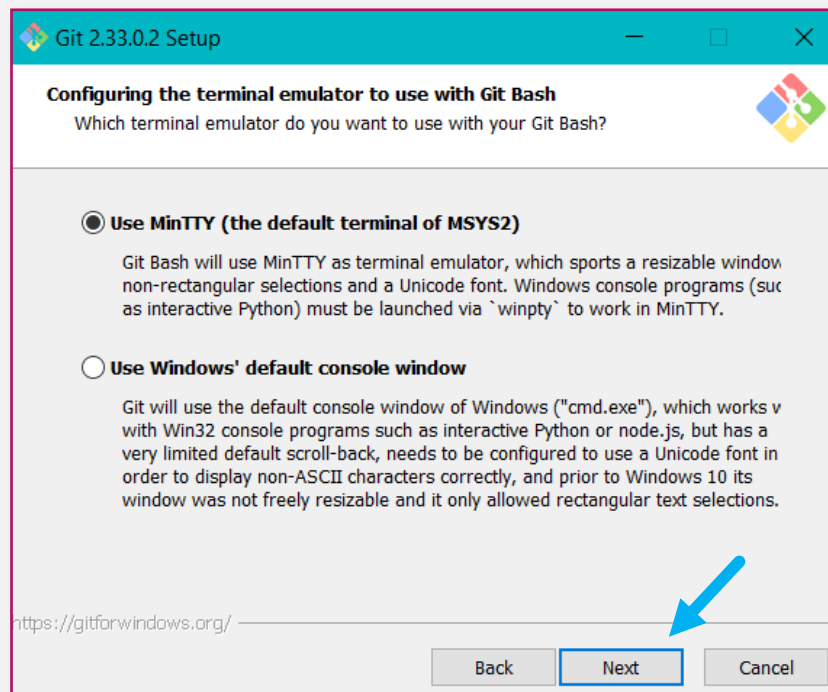
3.8 Transporte HTTPS



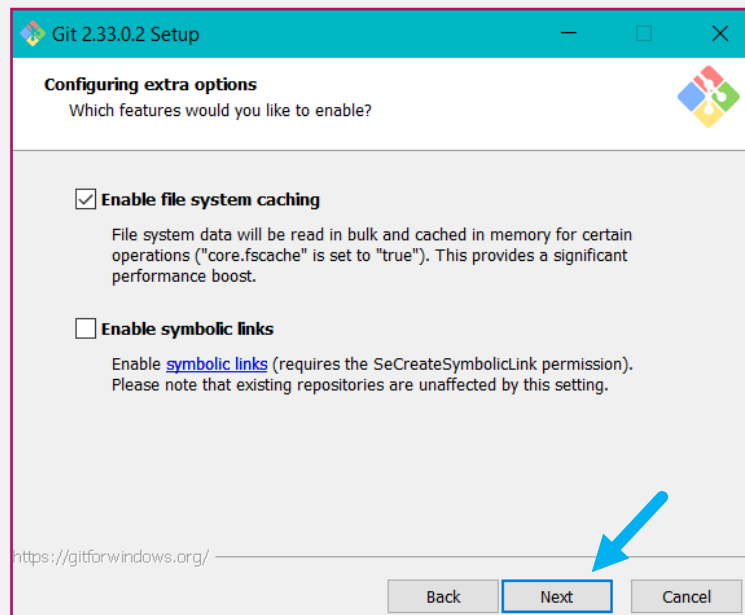
3.9 Configuración del formato para los finales de línea de los archivos.



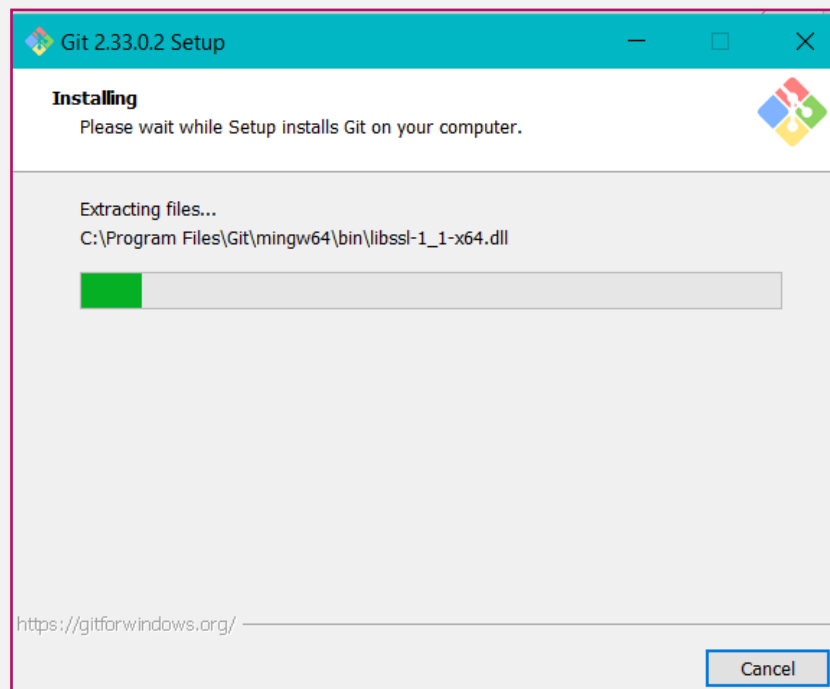
3.10 Tipo de emulador que usará Git Bash



3.11 Opciones extra de Git



3.12 Instalación

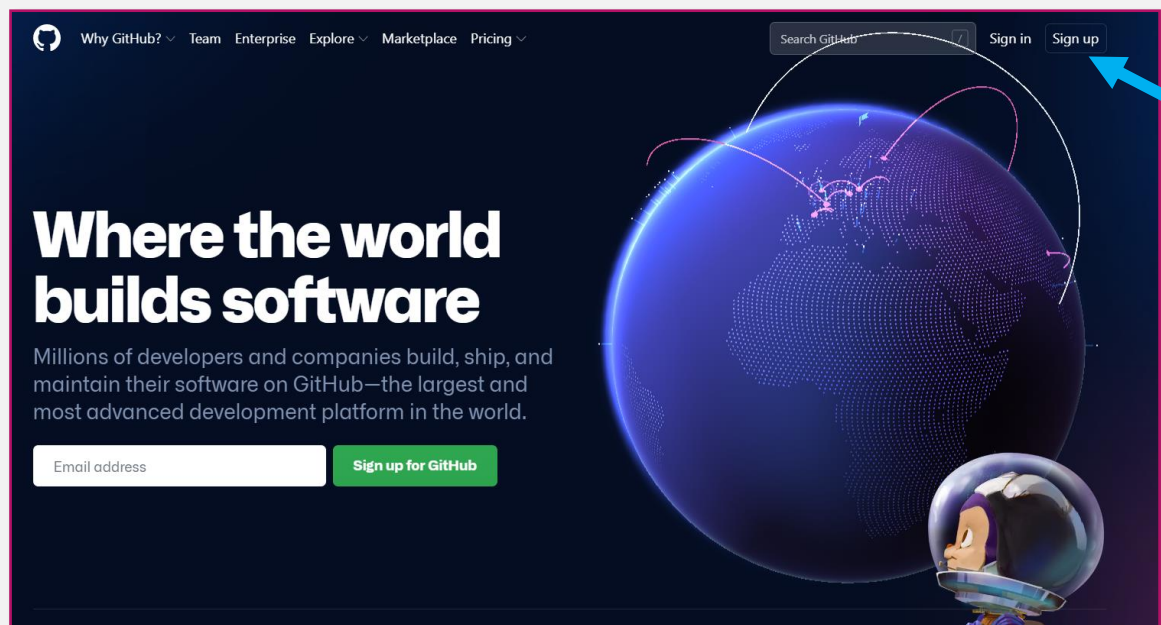


2. CREAR CUENTA DE GITHUB

El siguiente paso para poder tener nuestro repositorio donde almacenar nuestros proyectos es crear una cuenta en la plataforma GitHub, accederemos a ella a través del siguiente enlace:

<https://github.com/>

En la parte superior derecha de la pantalla encontraremos las opciones Sign in y Sign up, para iniciar sesión o crear una cuenta nueva, respectivamente.



A continuación, se nos abre un menú que nos pedirá ciertos datos, un correo electrónico, una contraseña y un nombre de usuario.

Luego nos preguntara si deseamos recibir información de actualizaciones por correo, yo en mi caso pondré "n" de no. En el caso contrario escribiremos "y".



Welcome to GitHub!
Let's begin the adventure

Enter your email

✓ `luciatest@gmail.com`

Create a password

✓ `.....`

Enter a username

✓ `luciatest`

Would you like to receive product updates and announcements via email?

Type "y" for yes or "n" for no

→ `n`

Continue

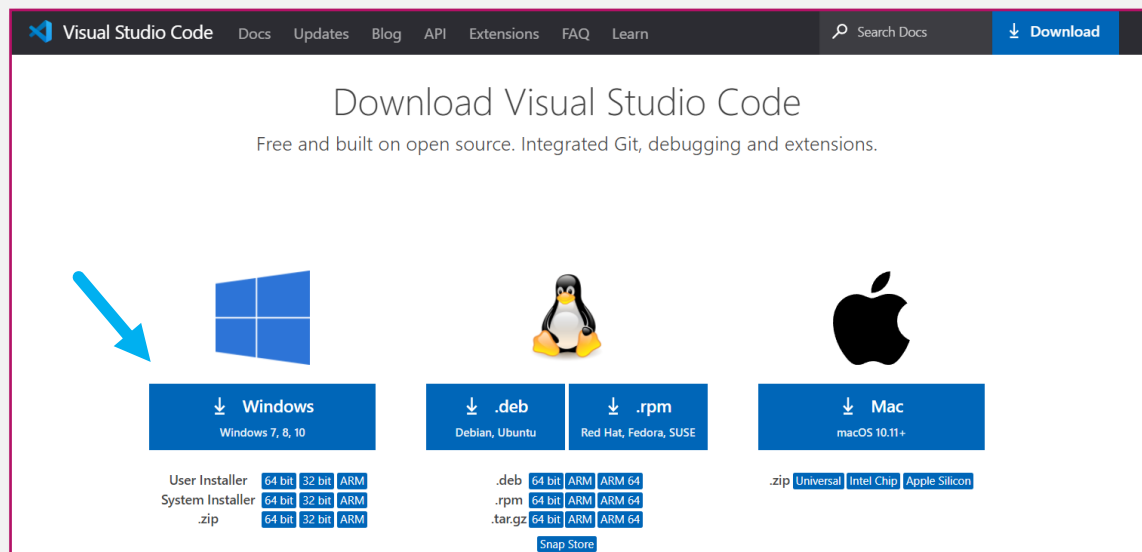


3. INSTALACIÓN VISUAL STUDIO CODE

Para la instalación de Visual Studio accederemos a su página principal, posible a través de este enlace:

<https://code.visualstudio.com/download>

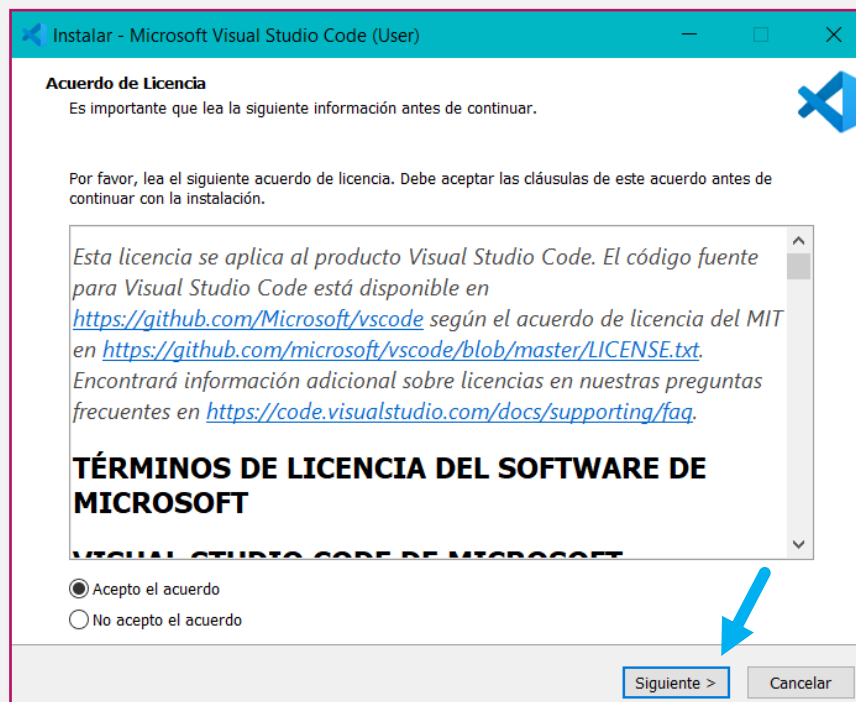
Una vez allí pincharemos en nuestro sistema operativo y la descarga comenzará automáticamente. En mi caso, Windows 10.



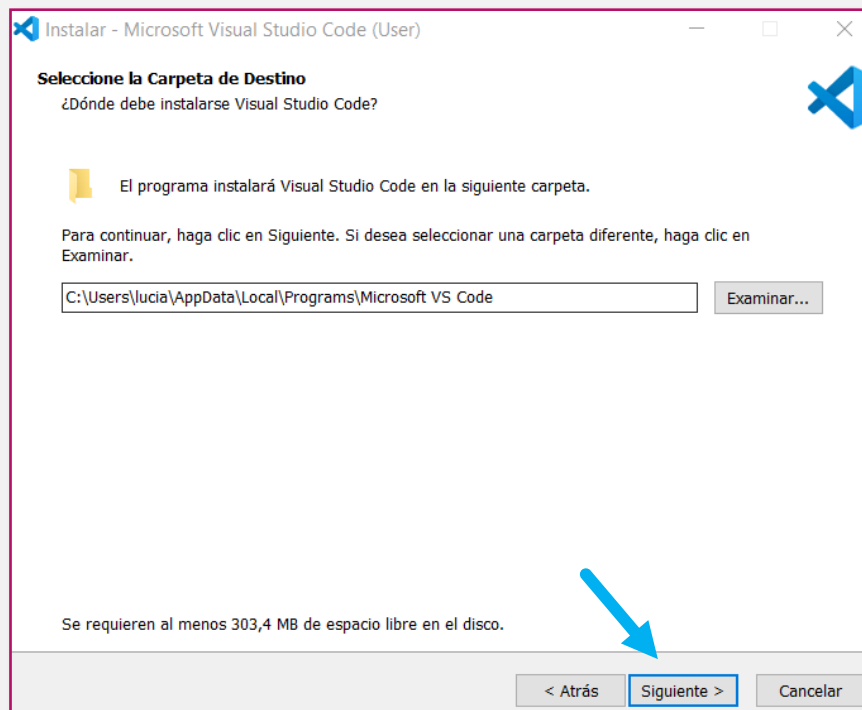
Abrimos el archivo ejecutable y comenzará la instalación.



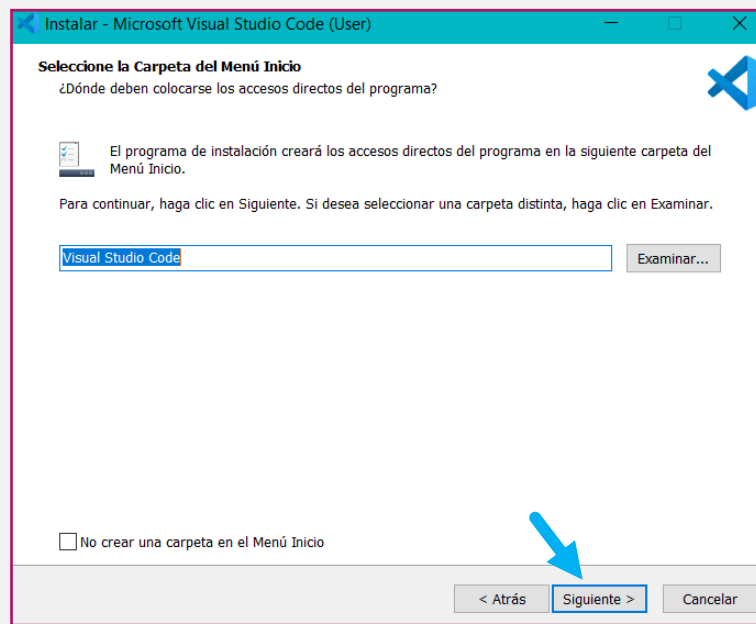
PASO 1. ACEPTAMOS LA LICENCIA



PASO 2. SELECCIONAMOS LA CARPETA DE DESTINO

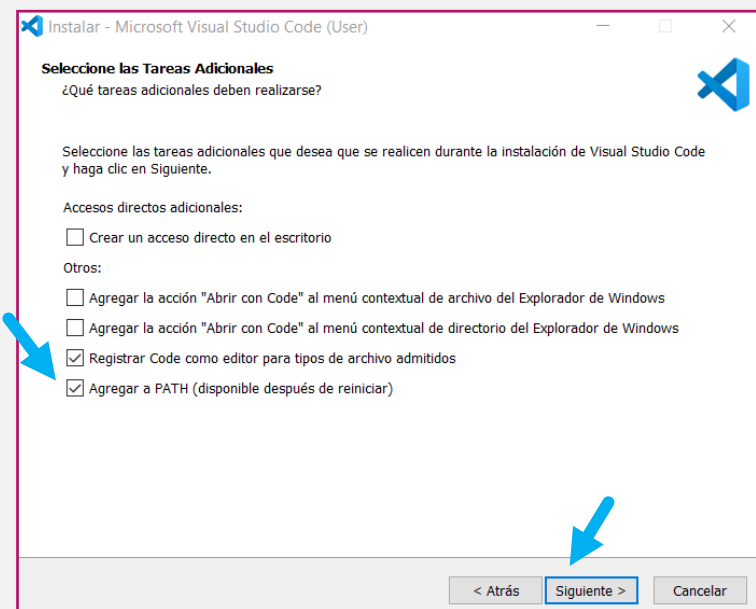


PASO 3. SELECCIONAMOS LA CARPETA DEL MENÚ DE INICIO

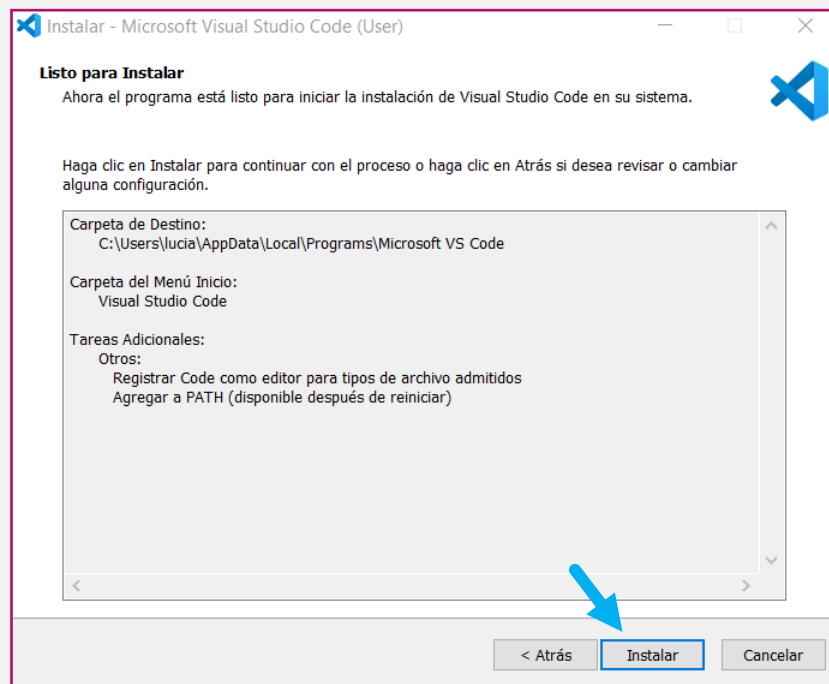


PASO 4. TAREAS ADICIONALES

¡IMPORTANTE! -> Marcar la casilla del PATH.

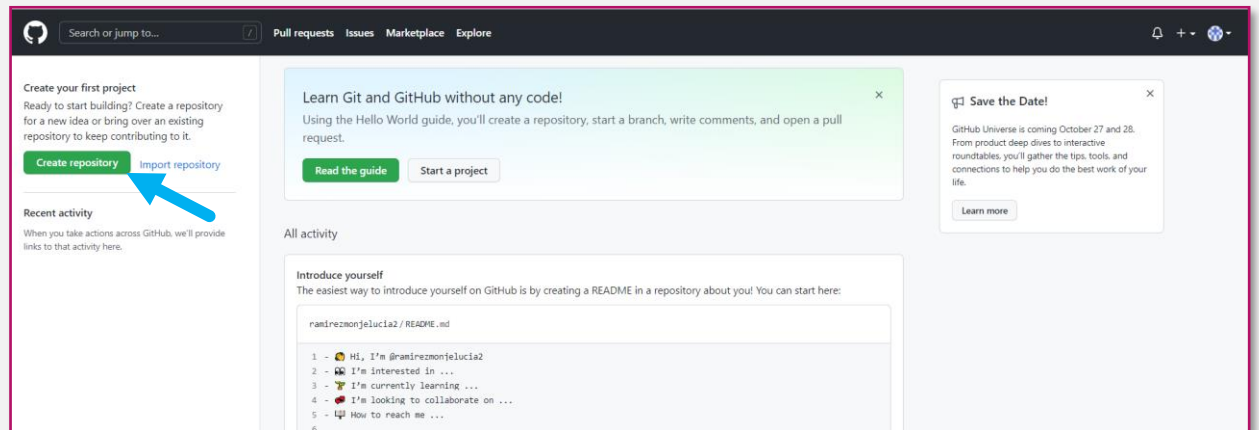


PASO 5. FIN DE LA INSTALACIÓN



4. MANDAR UN PROYECTO LOCAL AL REPOSITORIO

En primer lugar, vamos a crear el repositorio donde almacenaremos nuestro proyecto. Una vez iniciado sesión, en la pantalla principal de GitHub daremos clic en “[Create repository](#)”





Se nos abrirá una nueva pantalla donde deberemos rellenar únicamente un dato, el nombre del repositorio. Aunque hay varias opciones de personalización como por ejemplo si queremos que el repositorio sea privado o que sea visible para todo el mundo, o si queremos que el repositorio ya lleve añadido su archivo readme donde va un pequeño resumen del contenido o la carpeta .gitignore que será explicada detalladamente más tarde.



Create a new repository


A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)


Owner * **Repository name ***

 ramirezmonjelucia2 / pueba001 

Great repository names are short and memorable. Need inspiration? How about [turbo-giggle?](#)

Description (optional)

☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.


☐  **Private**
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

☐ **Add a README file**
This is where you can write a long description for your project. [Learn more.](#)

☐ **Add .gitignore**
Choose which files not to track from a list of templates. [Learn more.](#)




☐ **Choose a license**
A license tells others what they can and can't do with your code. [Learn more.](#)



Ya casi estamos cerca, una vez que tengamos creado nuestro repositorio éste estará vacío ya que nuestro proyecto lo tenemos en nuestra carpeta local.

Para llevar los archivos desde el local al repositorio vamos a seguir los pasos que nos da GitHub una vez creado el repositorio.

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

 Set up in Desktop or  HTTPS  SSH

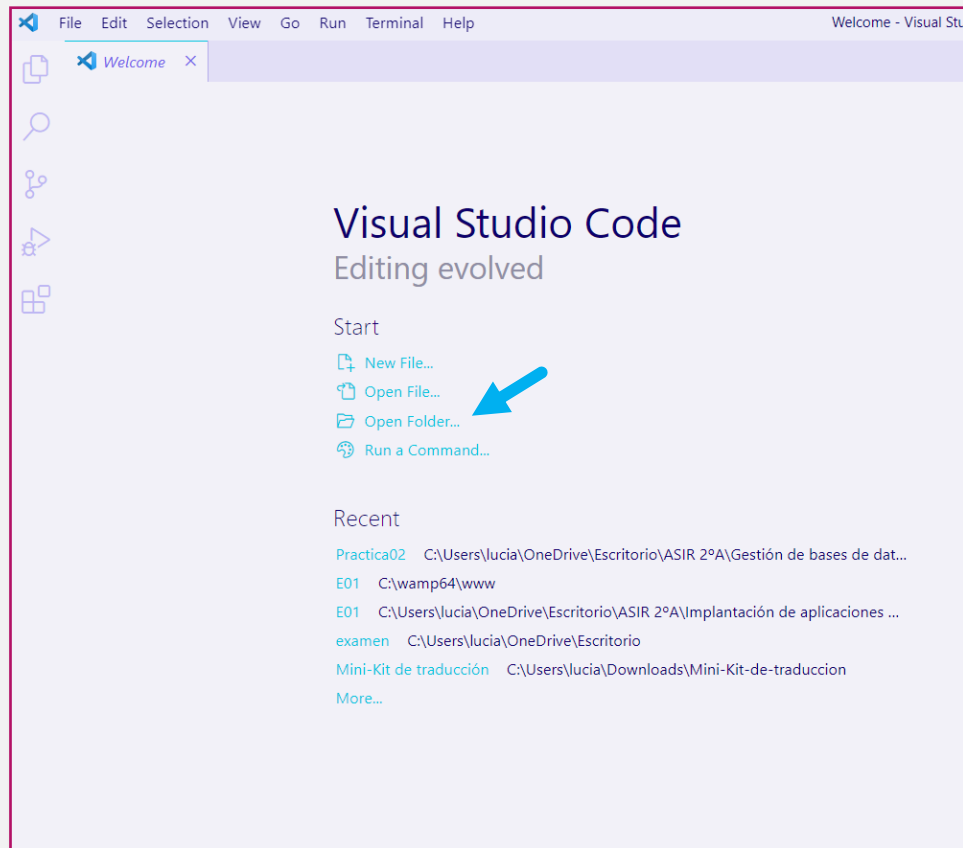
Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

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

```
echo "# EJERCICIO01" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/ramirezmonjelucia2/EJERCICIO01.git
git push -u origin main
```



Ya en este punto, lo siguiente que haremos será irnos a Visual Studio Code y abrir la carpeta donde tenemos nuestro proyecto.



A continuación, abriremos un terminal donde ejecutaremos los comandos que nos proporciona GitHub.

PASO 1: GIT INIT

Nos situaremos en la carpeta donde tenemos el proyecto, una vez ahí, ejecutamos el comando git init.

```
PS C:\Users\lucia\OneDrive\Escritorio\ASIR 2ºA\Gestión de bases de datos\Practica01> git init
Reinitialized existing Git repository in C:/Users/lucia/OneDrive/Escritorio/ASIR 2ºA/Gestión de bases de datos/Practica01/.git/
PS C:\Users\lucia\OneDrive\Escritorio\ASIR 2ºA\Gestión de bases de datos\Practica01>
```

Creamos una carpeta local en el repositorio con el comando git init, aunque podemos observar si ponemos git status que la carpeta está por crear y preparada para añadir la información.



PASO 2: AGREGAR LOS ARCHIVOS

¡IMPORTANTE! Para COMPROBAR que el proceso se está realizando correctamente tenemos a nuestra disposición el comando `git status`.

Continuamos con el comando `git add .`, este comando selecciona todos los archivos que hay en el directorio y los prepara para subirlos al repositorio.

```
PS C:\Users\lucia\OneDrive\Escritorio\ASIR 2ºA\Gestión de bases de datos\Practica01> git init
Initialized empty Git repository in C:/Users/lucia/OneDrive/Escritorio/ASIR 2ºA/Gestión de bases de datos/Practica01/.git/
No commits yet

(use "git add <file>..." to include in what will be committed)
"Guia de instalaci303\263n GITHUB borrador.docx"
"Guia de instalaci303\263n GITHUB.docx"
src/
"~$ia de instalaci303\263n GITHUB borrador.docx"

nothing added to commit but untracked files present (use "git add" to track)
PS C:\Users\lucia\OneDrive\Escritorio\ASIR 2ºA\Gestión de bases de datos\Practica01> git add .
PS C:\Users\lucia\OneDrive\Escritorio\ASIR 2ºA\Gestión de bases de datos\Practica01> git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   "Guia de instalaci303\263n GITHUB borrador.docx"
    new file:   "Guia de instalaci303\263n GITHUB.docx"
    new file:   src/estearchivosunaprueba.txt.txt
    new file:   "~$ia de instalaci303\263n GITHUB borrador.docx"
```

¡IMPORTANTE! Todos aquellos ficheros que no queramos subir al repositorio debemos meterlos en la carpeta `.gitignore`.

Para crear un archivo `.gitignore` local, crea un archivo de texto y asígnale el nombre `".gitignore"` (recuerda incluir el `.` al principio). Luego, edita este archivo según sea necesario. Cada nueva línea debe incluir un archivo o carpeta adicional que quieras que Git lo ignore.

```
PS C:\Users\lucia\OneDrive\Escritorio\ASIR 2ºA\Gestión de bases de datos\Practica01> git commit -m "Primera version"
[master (root-commit) d5cd799] Primera version
4 files changed, 1 insertion(+)
 create mode 100644 "Guia de instalaci303\263n GITHUB borrador.docx"
 create mode 100644 "Guia de instalaci303\263n GITHUB.docx"
 create mode 100644 src/estearchivosunaprueba.txt.txt
 create mode 100644 "~$ia de instalaci303\263n GITHUB borrador.docx"
PS C:\Users\lucia\OneDrive\Escritorio\ASIR 2ºA\Gestión de bases de datos\Practica01> git branch -M main
PS C:\Users\lucia\OneDrive\Escritorio\ASIR 2ºA\Gestión de bases de datos\Practica01> git remote add origin https://github.com/ramirezmonjelucia2/EJERCICIO01.git
PS C:\Users\lucia\OneDrive\Escritorio\ASIR 2ºA\Gestión de bases de datos\Practica01> git push -u origin main
```

```
PS C:\Users\lucia\OneDrive\Escritorio\ASIR 2ºA\Gestión de bases de datos\Practica01> git push -u origin main
info: please complete authentication in your browser...
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (7/7), 3.23 MiB | 3.44 MiB/s, done.
Total 7 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/ramirezmonjelucia2/EJERCICIO01.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
PS C:\Users\lucia\OneDrive\Escritorio\ASIR 2ºA\Gestión de bases de datos\Practica01> █
```



5. NOCIONES TEÓRICAS

¿Qué es GitHub y para qué sirve?

Se podría hablar de Github como la red social pensada para desarrolladores, es un repositorio online gratuito que permite gestionar proyectos y controlar versiones de código.

Es muy utilizado por desarrolladores para almacenar sus trabajos dando así la oportunidad a millones de personas de todo el mundo a cooperar en ellos.

Podemos seguir e interactuar con personas interesadas en un tipo de proyecto en concreto, dando a conocer los nuestros o cooperando en el proyecto de terceros.

¿Qué es un control de versiones?

Un control de versiones permite a los desarrolladores administrar cambios en un software a la vez que el proyecto evoluciona.

En el caso de que un desarrollador quisiera trabajar en un proyecto, sería arriesgado realizar cambios sobre el código original. El control de versiones permite duplicar una parte de un proyecto de forma aislada y trabajar sobre ella sin que se modifique el repositorio original.

Una vez comprobado que el cambio se ha realizado con éxito, el desarrollador podrá crear una nueva versión del proyecto.

Esta nueva versión registra los cambios realizados sobre la versión anterior para que se puedan testear por otros desarrolladores.



¿Qué es un repositorio?

Un repositorio es la ubicación o ruta en la que se almacena toda la información de un proyecto como imágenes, código, carpetas, documentos, etc.

Cada proyecto contaría con su propio repositorio único, por lo que la ruta de acceso será exclusiva para el proyecto.

¿Qué es Git?

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

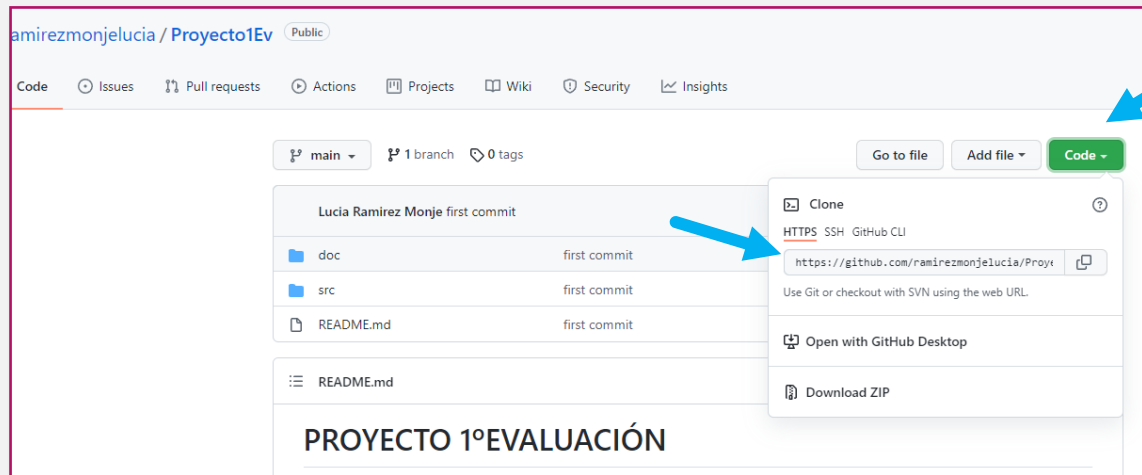


6. CLONACIÓN

Este apartado consistirá en un ejercicio práctico que nos puede suceder en cualquier empresa en la que estemos trabajando con un proyecto en la nube.

Para empezar, cogeremos la dirección del repositorio en el que se encuentra.

Hacemos clic en el botón verde "Code" y luego copiamos la URL que aparece.



A continuación, abriremos una terminal, puede ser Windows Power Shell o en mi caso, Visual Studio Code. Nos situamos en el directorio donde queremos que se guarde la clonación y por último ejecutamos el siguiente comando:

```
git clone https://github.com/URL
```

De esta forma podemos ver que empezará la clonación en nuestro directorio local.

```
PS C:\Users\lucia\OneDrive\Escritorio\ASIR 2ºA\Gestión de bases de datos> git clone https://github.com/ramirezmonjelucia/Proyecto1Ev.git
Cloning into 'Proyecto1Ev'...
remote: Enumerating objects: 40, done.
remote: Counting objects: 100% (40/40), done.
remote: Compressing objects: 100% (33/33), done.
remote: Total 40 (delta 10), reused 36 (delta 6), pack-reused 0
Receiving objects: 100% (40/40), 1.23 MiB | 4.18 MiB/s, done.
Resolving deltas: 100% (10/10), done.
```



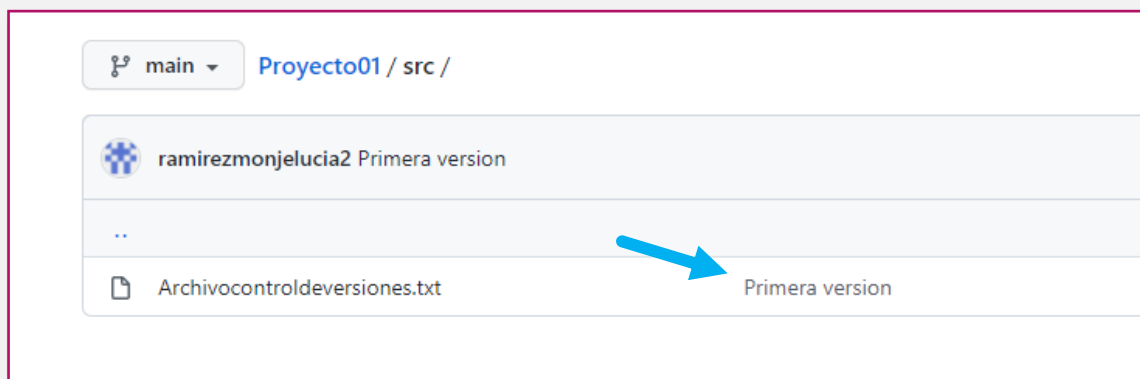
7. PRÁCTICA

Consistirá en:

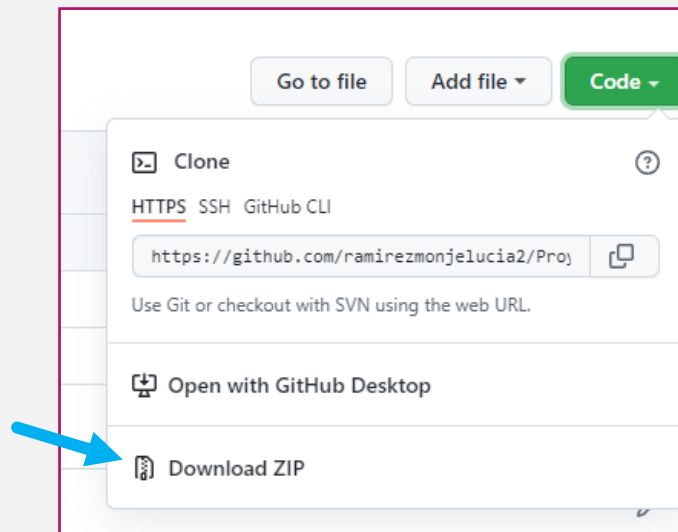
Tendremos dos ordenadores diferentes, imaginemos que estamos en una empresa trabajando conjuntamente y usamos el mismo repositorio para nuestro proyecto. Lo que haremos será crear una situación real.

Yo como informática de la empresa voy a extraer el repositorio donde el a trabajado y al finalizar los cambios lo volveré a subir, pero con las nuevas modificaciones y su versión 2.

Podemos ver que en el repositorio ya existe su primera versión:



En primer lugar, nos descargaremos el repositorio en el local. En mi caso he usado el comando git pull URL. Pero se puede hacer haciendo una clonación (explicada en el apartado 6) o descargando el ZIP.



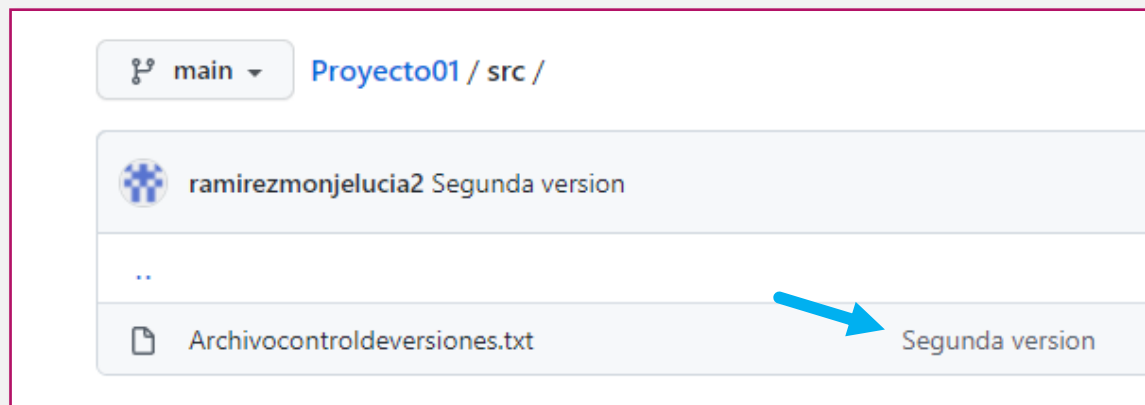
Una vez que ya tenemos el proyecto en la primera versión, vamos a modificar los archivos que sean necesarios para continuar con el proyecto.

```
PS C:\Users\lucia\OneDrive\Escritorio\ASIR 2ºA\Gestión de bases de datos\Proyecto01> git pull https://github.com/ramirezmonjelucia2/Proyecto01.git
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 810 bytes | 14.00 KiB/s, done.
From https://github.com/ramirezmonjelucia2/Proyecto01
* branch      HEAD      -> FETCH_HEAD
Updating 77f8f9c..38f5a1c
Fast-forward
 actualizacion.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 actualizacion.txt
```

Una vez que se han realizado los cambios procedentes se subirá otra vez al repositorio en su versión 2.

```
PS C:\Users\lucia\OneDrive\Escritorio\ASIR 2ºA\Gestión de bases de datos\Proyecto01> git add .
PS C:\Users\lucia\OneDrive\Escritorio\ASIR 2ºA\Gestión de bases de datos\Proyecto01> git commit -m "Segunda version"
[main 80b064a] Segunda version
 2 files changed, 3 insertions(+), 1 deletion(-)
 delete mode 100644 actualizacion.txt
PS C:\Users\lucia\OneDrive\Escritorio\ASIR 2ºA\Gestión de bases de datos\Proyecto01> git push -u origin main
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 374 bytes | 187.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/ramirezmonjelucia2/Proyecto01.git
 38f5a1c..80b064a  main -> main
```

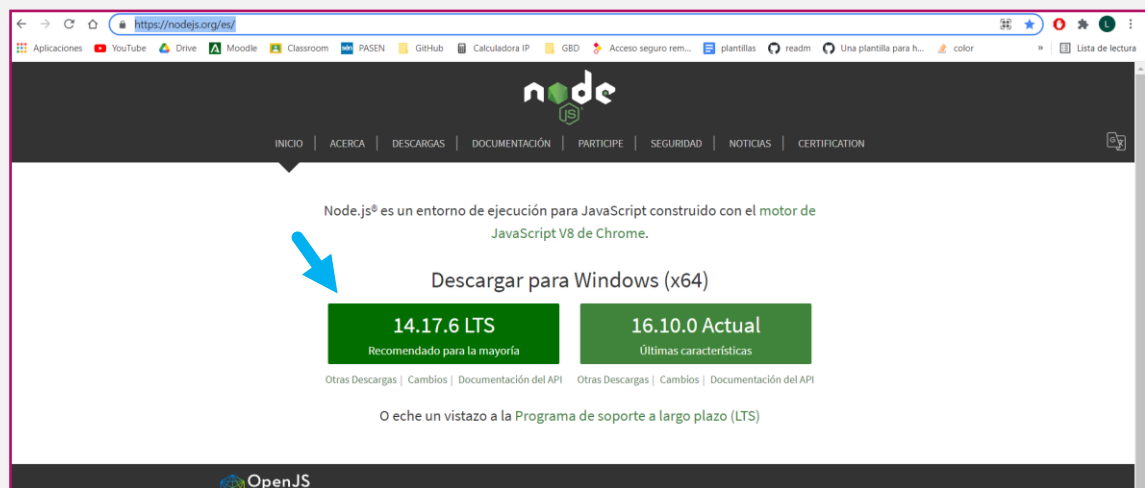




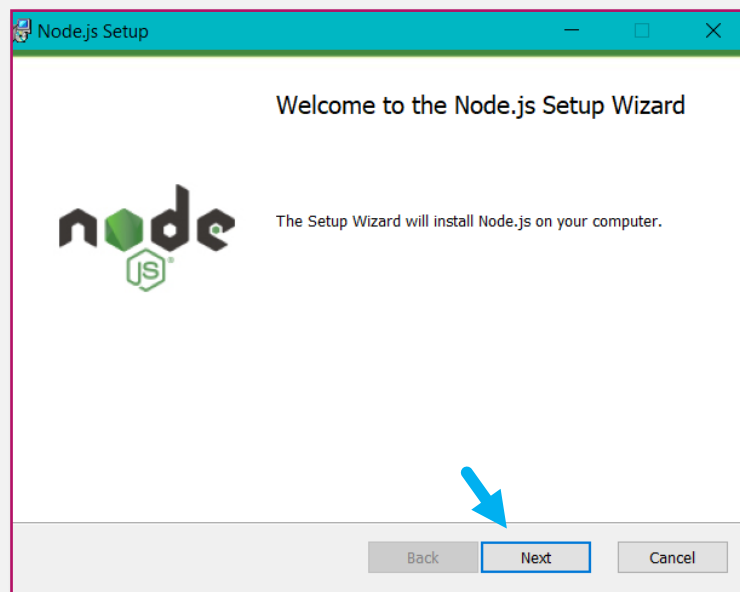
8. INSTALACIÓN ENTORNO NODE.JS

Vamos a proceder con la descarga yendo a la pagina principal de Node.js o directamente desde el siguiente enlace: <https://nodejs.org/es/>

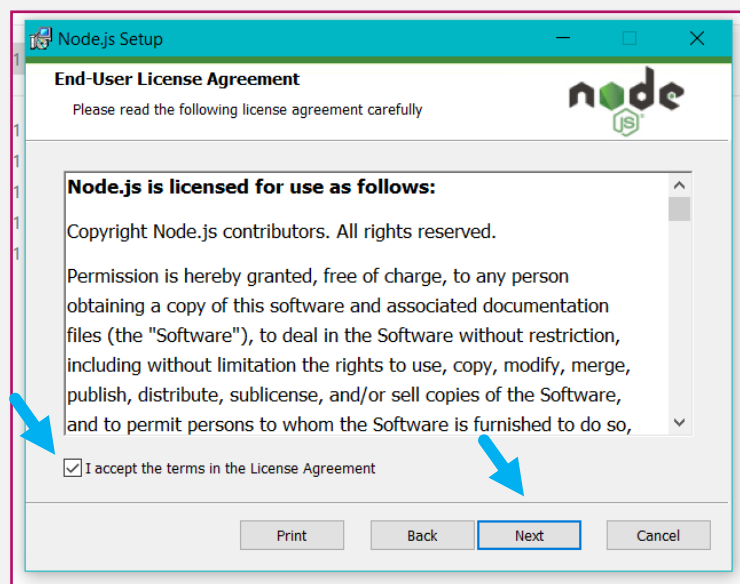
En mi caso usaré la versión 14.17.6 LTS.



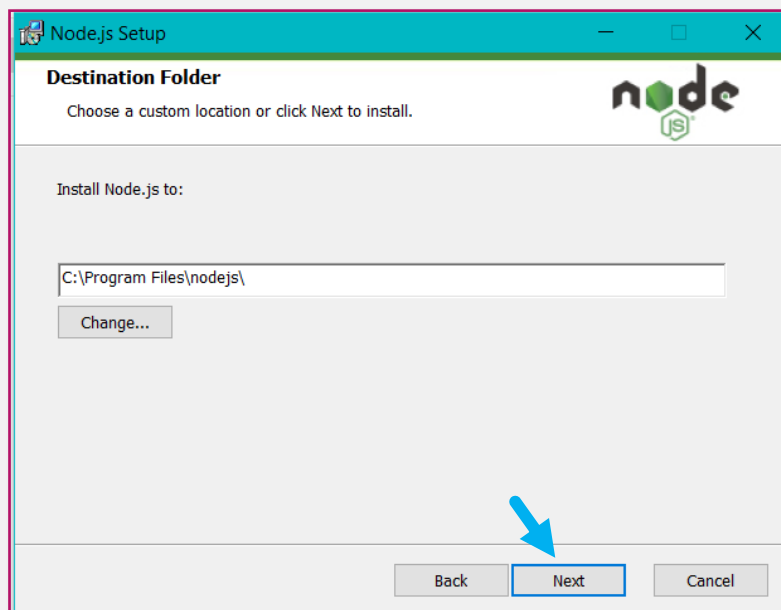
8.1 Abrir el archivo de instalación



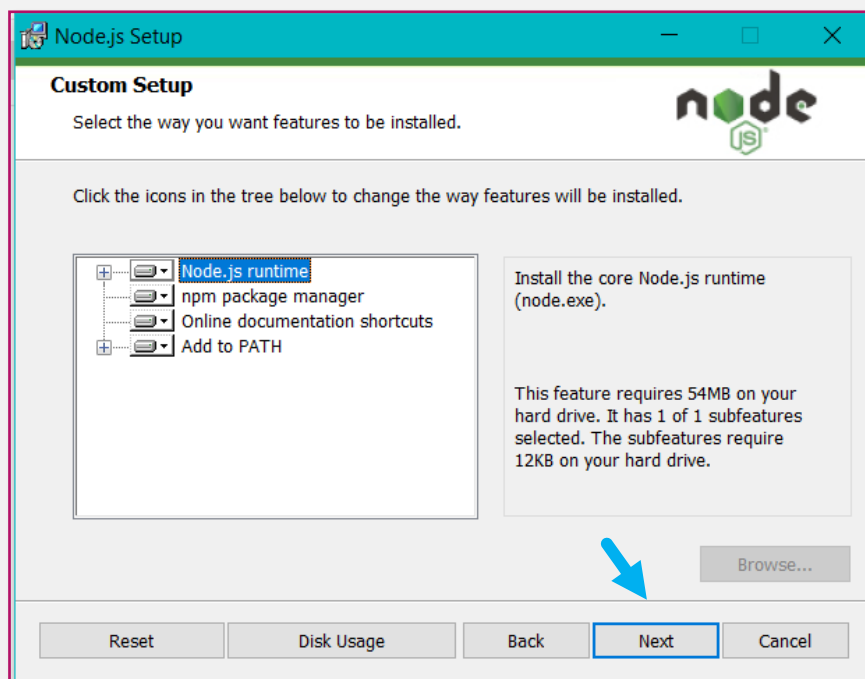
8.2 Aceptar la licencia



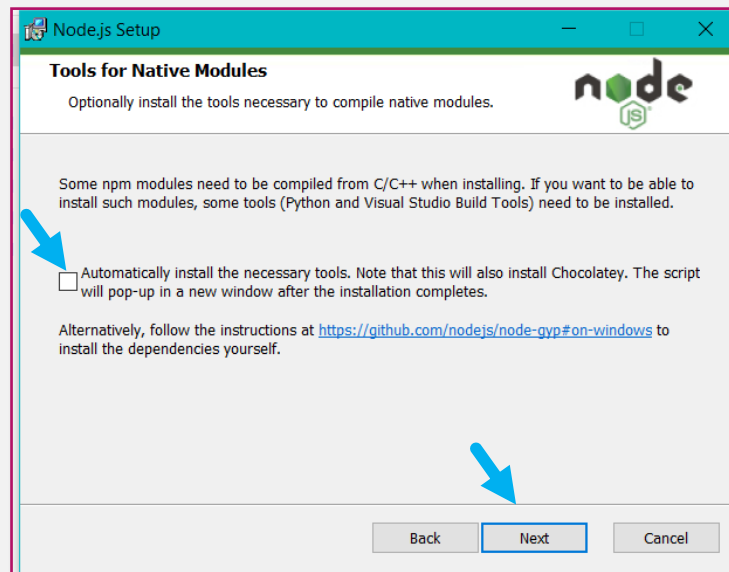
8.3 Elegimos la carpeta de destino



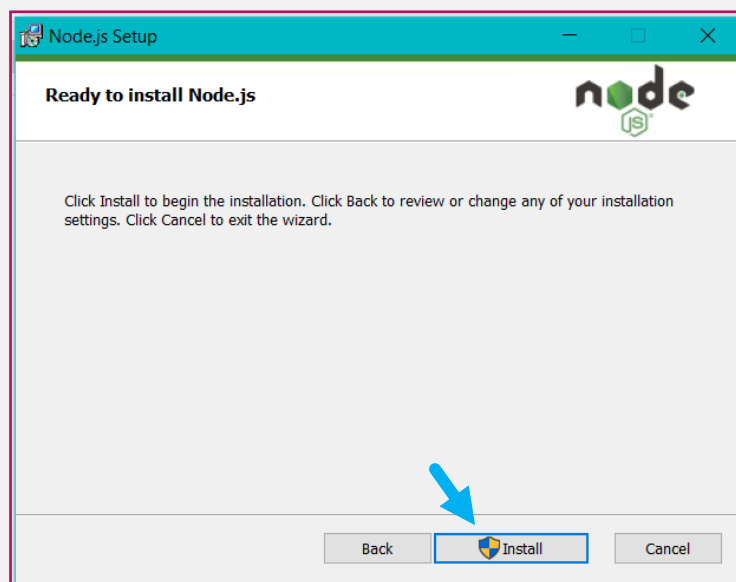
8.4 Elementos que deseas instalar (por defecto todos)



8.5 Herramientas opcionales (sin marcar)



8.6 Listo para instalar



Para ver si la instalación ha sido correcta nos vamos a un terminal nuevo y usaremos el siguiente comando:

```
PS C:\Users\lucia> node --version  
v14.17.6  
PS C:\Users\lucia> node -v  
v14.17.6
```

Node te permite usar una SHELL PROPIA, para usarla escribiremos el comando:

```
PS C:\Users\lucia> node  
Welcome to Node.js v14.17.6.  
Type ".help" for more information.
```

También usaremos NPM, un gestor de paquetes de NODE.JS:

<https://www.npmjs.com/package/typescript>



9. INSTALACIÓN LENGUAJE TYPESCRIPT

Para instalar typescript usaremos una terminal y el siguiente comando:

```
npm install -g typescript
```

g de global, repercute en todos los proyectos, no solo para el proyecto que estemos ejecutando.

```
PS C:\Users\lucia> npm install -g typescript
C:\Users\lucia\AppData\Roaming\npm\tsserver -> C:\Users\lucia\AppData\Roaming\npm\node_modules\typescript\bin\tsserver
C:\Users\lucia\AppData\Roaming\npm\tsc -> C:\Users\lucia\AppData\Roaming\npm\node_modules\typescript\bin\tsc
+ typescript@4.4.3
added 1 package from 1 contributor in 2.82s
PS C:\Users\lucia>
```

Node sirve para ejecutar un fichero con contenido javascript

Primera compilación

Usaremos un archivo nuevo que he creado, llamado E01.ts

Para compilar ejecutamos el comando tsc + archivo.ts y esto nos generará en el directorio donde estemos el archivo compilado con extensión .js

```
PS C:\Users\lucia\OneDrive\Escritorio\ASIR 2ºA\Gestión de bases de datos\Practica02> tsc E01.ts
PS C:\Users\lucia\OneDrive\Escritorio\ASIR 2ºA\Gestión de bases de datos\Practica02> dir

Directorio: C:\Users\lucia\OneDrive\Escritorio\ASIR 2ºA\Gestión de bases de datos\Practica02

Mode                LastWriteTime         Length Name
----                -
-a----             22/09/2021   9:07             31 E01.js
-a----             22/09/2021   8:53             28 E01.ts
-a----             22/09/2021   8:53             28 E02.js

PS C:\Users\lucia\OneDrive\Escritorio\ASIR 2ºA\Gestión de bases de datos\Practica02>
```



10. PROYECTO TYPESCRIPT

10.1 PASOS PREVIOS

Vamos a separar este apartado en “dos partes” la primera, en la creación del archivo package.json, y la segunda en la creación del archivo tsconfig.json.

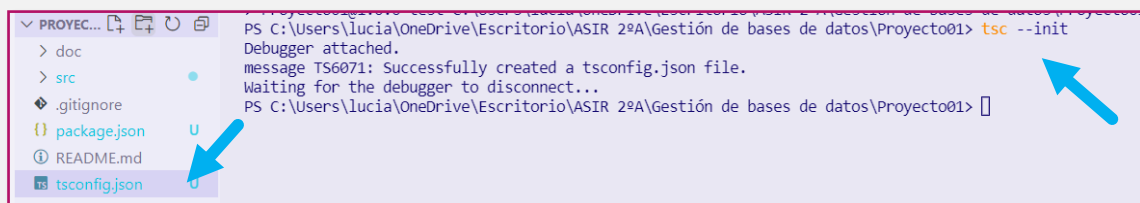
Al crear un nuevo proyecto con `npm init -y`, se lanzará un asistente que tras algunas preguntas, crea un archivo llamado package.json en la carpeta raíz del proyecto, donde coloca toda la información que se conoce sobre el mismo. Este archivo es un simple fichero de texto, en formato JSON que incorpora a través de varios campos información muy variada.



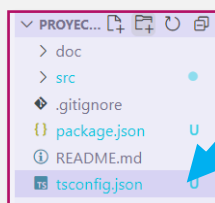
```
PS C:\Users\lucia\OneDrive\Escritorio\ASIR 2ºA\Gestión de bases de datos\Proyecto01> npm init -y
Debugger attached.
Wrote to C:\Users\lucia\OneDrive\Escritorio\ASIR 2ºA\Gestión de bases de datos\Proyecto01\package.json:

{
  "name": "Proyecto01",
  "version": "1.0.0",
  "description": " Realizado por Lucía Ramírez Monje 2ºAsir A_",
  "main": "index.js",
  "directories": {
    "doc": "doc"
  },
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "type": "git",
  "url": "git+https://github.com/ramirezmonjelucia2/Proyecto01.git",
  "keywords": [],
  "author": "",
  "license": "ISC",
  "bugs": {
    "url": "https://github.com/ramirezmonjelucia2/Proyecto01/issues"
  },
  "homepage": "https://github.com/ramirezmonjelucia2/Proyecto01#readme"
}
```

El archivo tsconfig.json es el que indica en un proyecto que se está trabajando con TypeScript. Lo colocas en la raíz de carpetas del proyecto y en él situamos un JSON con todas las configuraciones de trabajo para el transpilador de TypeScript. Se crea con el comando `tsc --init`



```
PS C:\Users\lucia\OneDrive\Escritorio\ASIR 2ºA\Gestión de bases de datos\Proyecto01> tsc --init
Debugger attached.
message TS6071: Successfully created a tsconfig.json file.
Waiting for the debugger to disconnect...
PS C:\Users\lucia\OneDrive\Escritorio\ASIR 2ºA\Gestión de bases de datos\Proyecto01> 
```



Ya para finalizar la "instalación" iremos a este último archivo tsconfig.json y cambiaremos los siguientes caracteres:

```
{
  "compilerOptions": {
    /* Visit https://aka.ms/tsconfig.json to read more about this file */

    /* Projects */
    // "incremental": true,           /* Enable incremental compilation */
    // "composite": true,           /* Enable constraints that allow a TypeScript compiler to be used by multiple compilers */
    // "tsBuildInfoFile": "./",     /* Specify the folder for .tsbuildinfo files */
    // "disableSourceOfProjectReferenceRedirect": true, /* Disable preferring source files instead of declaration files */
    // "disableSolutionSearching": true, /* Opt a project out of multi-project reference checking */
    // "disableReferencedProjectLoad": true, /* Reduce the number of projects loaded automatically by TypeScript */

    /* Language and Environment */
    "target": "es6",                /* Set the JavaScript language version for emitted JavaScript and include compatible library declarations to the compilation */
    // "lib": [],                   /* Specify a set of bundled library declaration files that describe the target runtime environment */
  }
}
```

```
// "sourceMap": true,
// "outFile": "./",
  "outDir": "./dist",
// "removeComments": true,
// "noEmit": true,
// "importHelpers": true,
// "importsNotUsedAsValues": "remove",
// "downlevelIteration": true,
// "sourceRoot": "",
// "mapRoot": "",
// "inlineSourceMap": true,
// "inlineSources": true,
// "emitBOM": true,
// "newline": "crlf",
// "stripInternal": true,
// "noEmitHelpers": true,
// "noEmitOnError": true,
// "preserveConstEnums": true,
// "declarationDir": ""
```



10.2 PRÁCTICA

Vamos a crear un archivo dentro de la carpeta src llamado basico.ts.

Dentro escribiremos lo siguiente:

```
src > TS basico.ts
1 console.log("Hola mundo")
```

Una vez que lo hayamos escrito compilamos o bien con el comando tsc o tsc -w

-w es el atributo para que escuche a ver si encuentra fallos a la vez que estamos trabajando

```
[19:11:41] Starting compilation in watch mode...
[19:11:42] Found 0 errors. Watching for file changes.
```

En el terminal podemos hacer la invocación al ejecutable JavaScript que se encontrará en la carpeta dist

node dist/básico.js

RESULTADO FINAL

```
PS C:\Users\lucia\OneDrive\Escritorio\ASIR 2ºA\Gestión de bases de datos\Proyecto01> node dist/basico.js
Debugger attached.
Hola mundo
Waiting for the debugger to disconnect...
```

```
PS C:\Users\lucia\OneDrive\Escritorio\ASIR 2ºA\Gestión de bases de datos\Proyecto01> tsc
PS C:\Users\lucia\OneDrive\Escritorio\ASIR 2ºA\Gestión de bases de datos\Proyecto01> node dist/index.js
Hola mundo
```

