

# PUESTA EN MARCHA DE NUESTRO ENTORNO DE TRABAJO

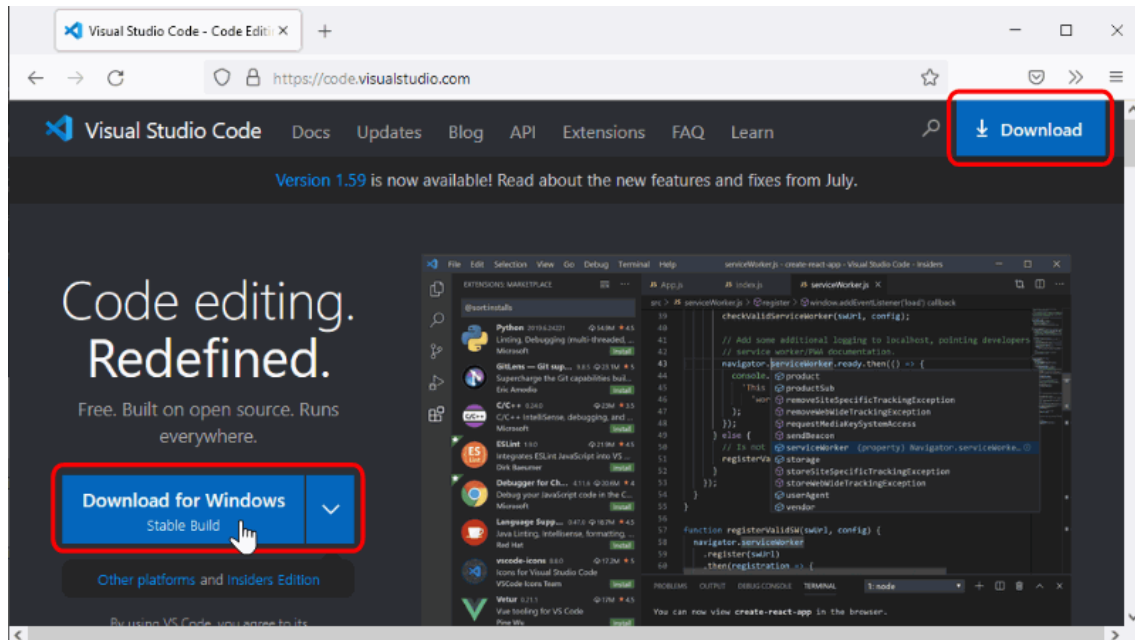
## CONTENIDO

Visual Studio Code. Instalación .....	2
Conseguir Visual Studio Code.....	2
Instalar Visual Studio Code en Windows .....	3
Visual Studio Code. Uso.....	8
Visual Studio Code. Configuración recomendada para el curso Introducción a la programación con Python .....	14
Tema de color recomendado para el curso de Python .....	14
Área de trabajo.....	15
Preferencias de configuración recomendadas para el curso de Python .....	16
Extensiones recomendadas para el curso de Python.....	19
multi-command .....	19
Python extension for Visual Studio Code .....	19
Instalación de pylint .....	19
Configuración de pylint .....	20
Instalación de módulo de autoformato.....	21
Creación de la primera aplicación Python .....	22
Seleccione y active un entorno .....	22
Crear un nuevo archivo de Python y agregar código .....	27
Ejecutar programa.....	28
Otras Extensiones.....	29
Trabajar con repositorios git .....	32
Instalar Git en Windows .....	33
GitHub. Crear cuenta.....	43
Crear un repositorio en GitHub .....	47

## VISUAL STUDIO CODE. INSTALACIÓN

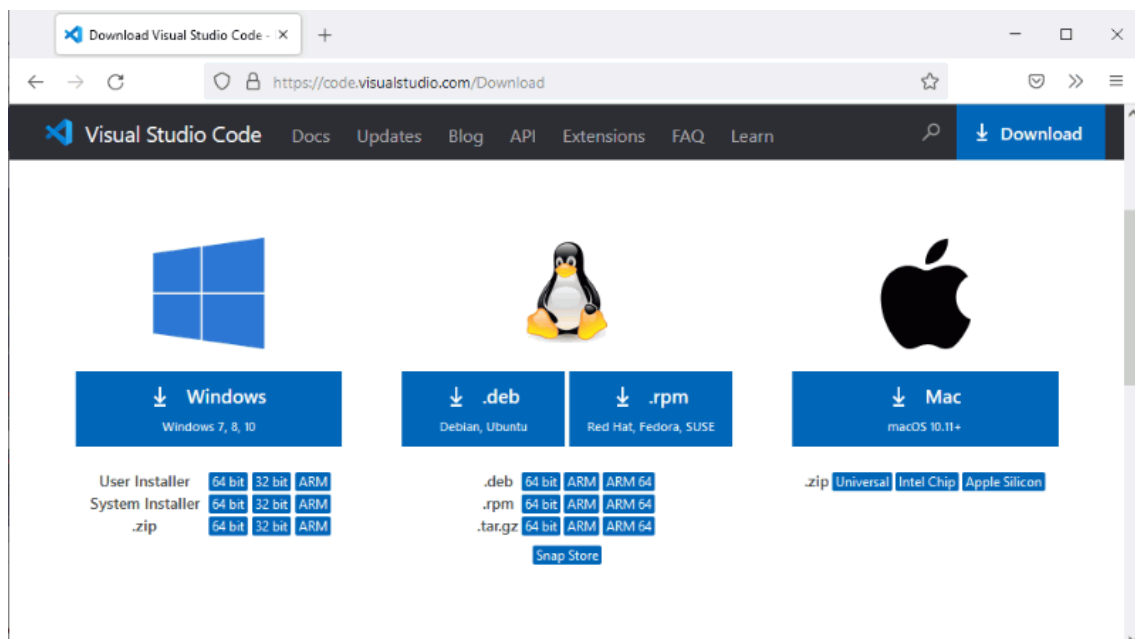
### CONSEGUIR VISUAL STUDIO CODE

La página oficial de Visual Studio Code es <https://code.visualstudio.com/>. Desde la página principal se puede descargar la última versión estable (mediante el cuadro verde grande situado a la izquierda) o acceder a la [página de descargas](#) (mediante el cuadro verde más pequeño situado arriba a la derecha):



Desde [la página de descargas](#) se pueden descargar las versiones para diferentes sistemas operativos (32 / 64 bits, Windows / GNU/Linux / Mac). En Windows hay además disponibles versiones *System installer*, que se instalan en la carpeta de Archivos de programa, y versiones *User installer*, que se instalan en la carpeta de usuario. Desde el verano de 2018, Microsoft recomienda la versión *User installer*.

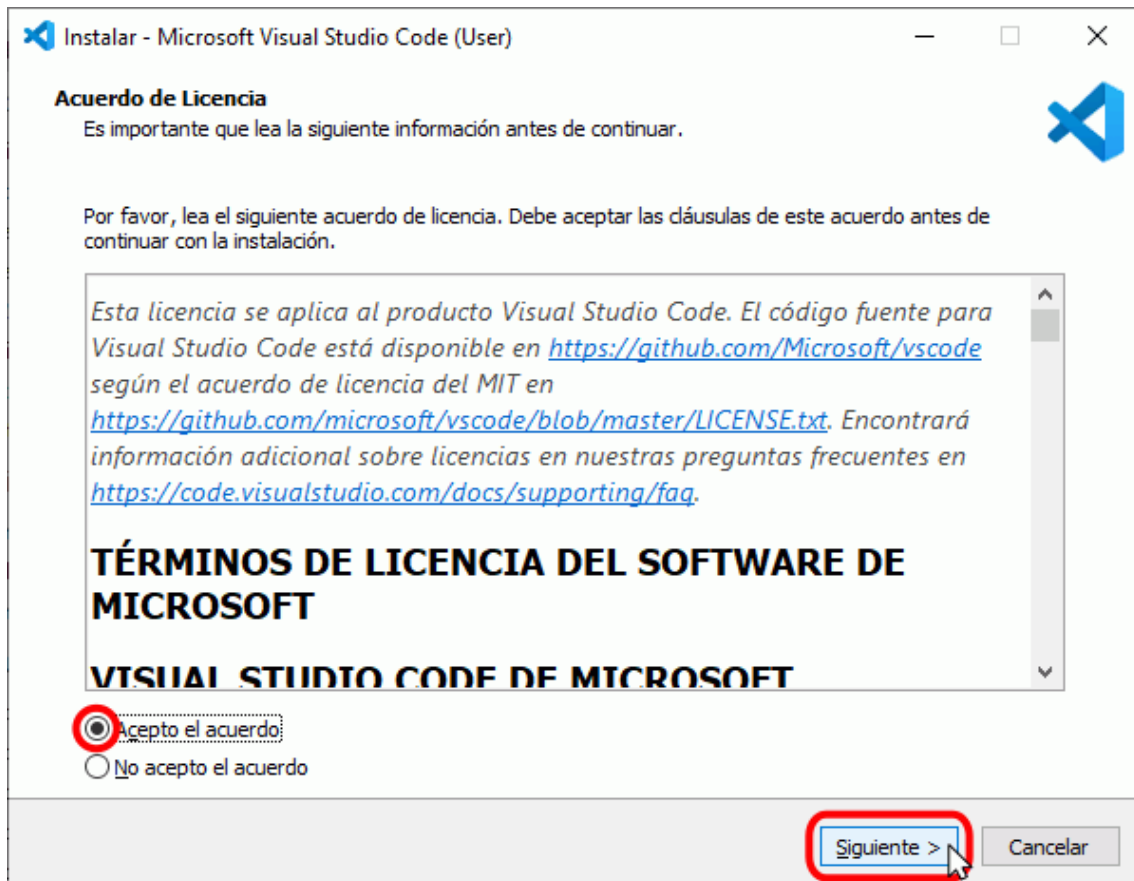
Enlaces de descarga para Windows: [Visual Studio Code \(64 bits, recomendado\)](#) - [Visual Studio Code \(32 bits\)](#)



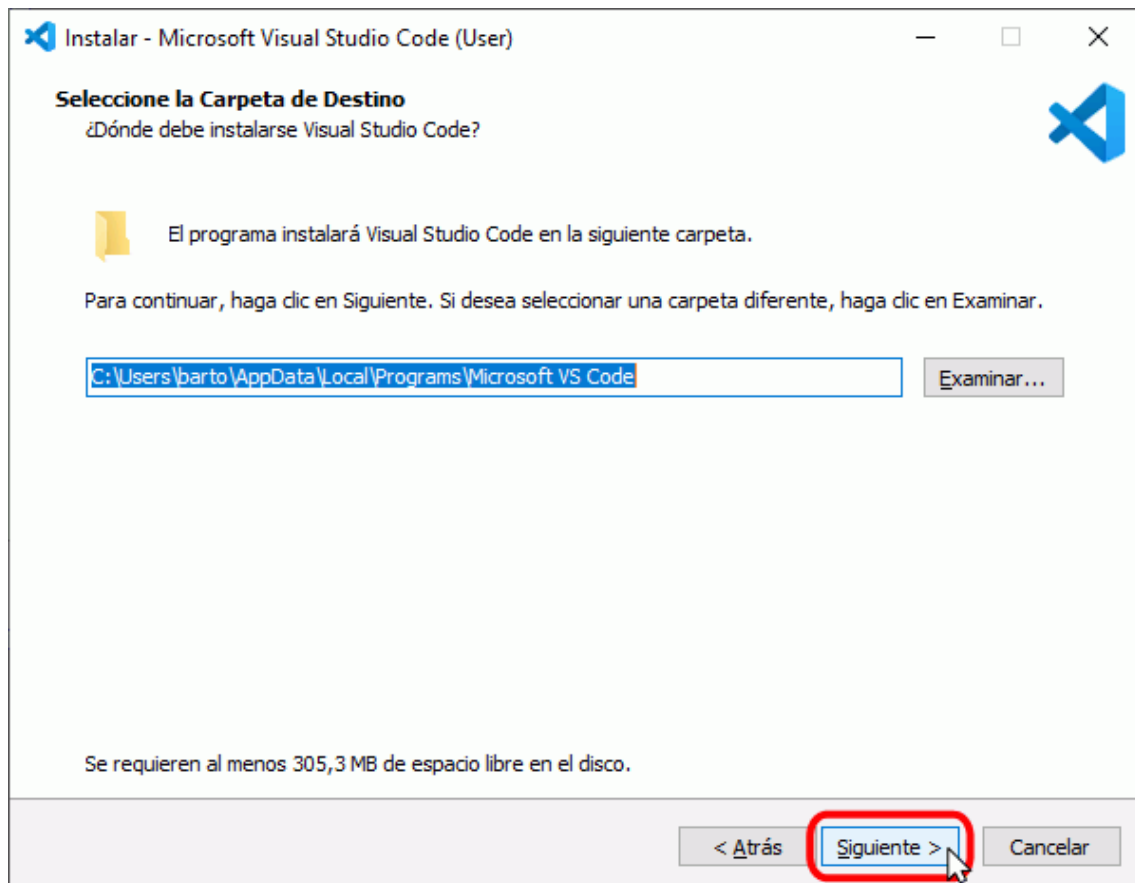
## INSTALAR VISUAL STUDIO CODE EN WINDOWS

**Nota:** Las capturas siguientes corresponden a Visual Studio Code 1.59 (*User installer*) en Windows 10 de 64 bits. Versiones posteriores pueden ser ligeramente diferentes.

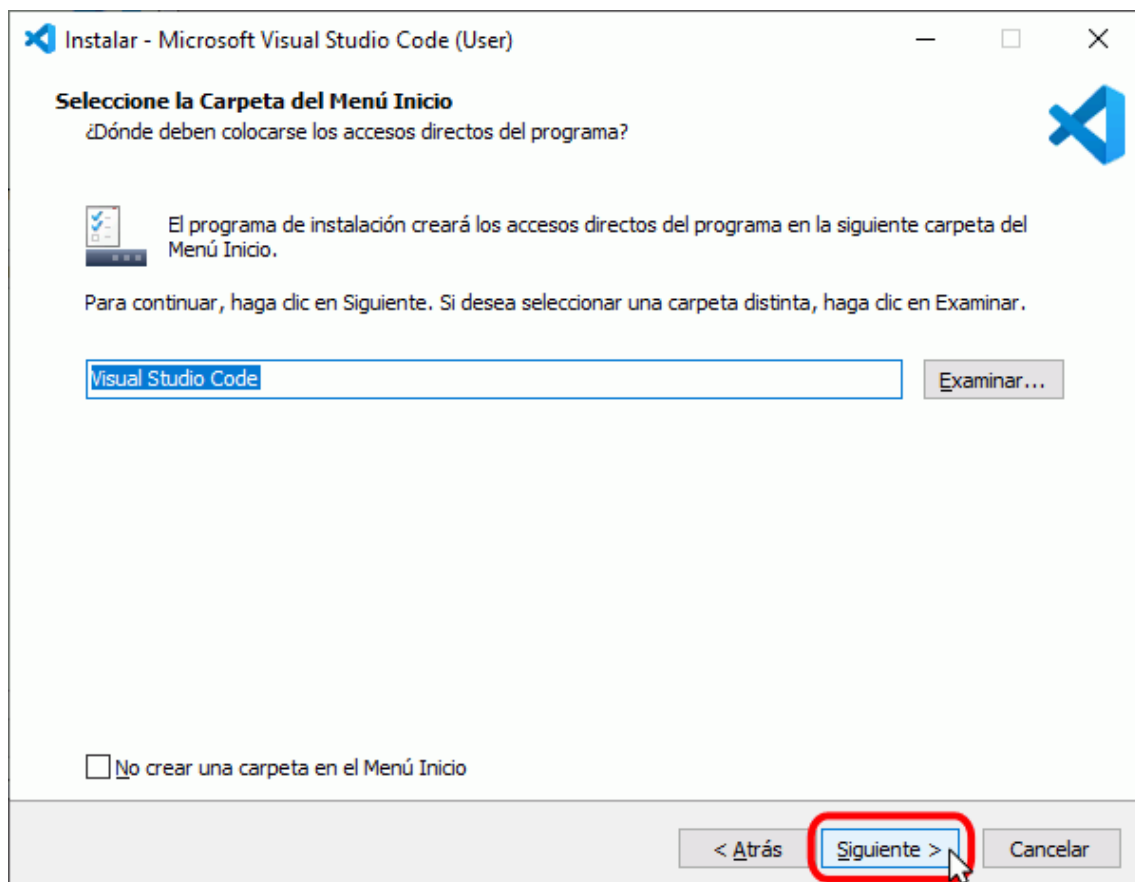
1. Haga doble clic sobre el instalador de Visual Studio Code para poner en marcha el asistente de instalación.
2. La primera pantalla exige aceptar la licencia de Visual Studio Code para continuar la instalación:



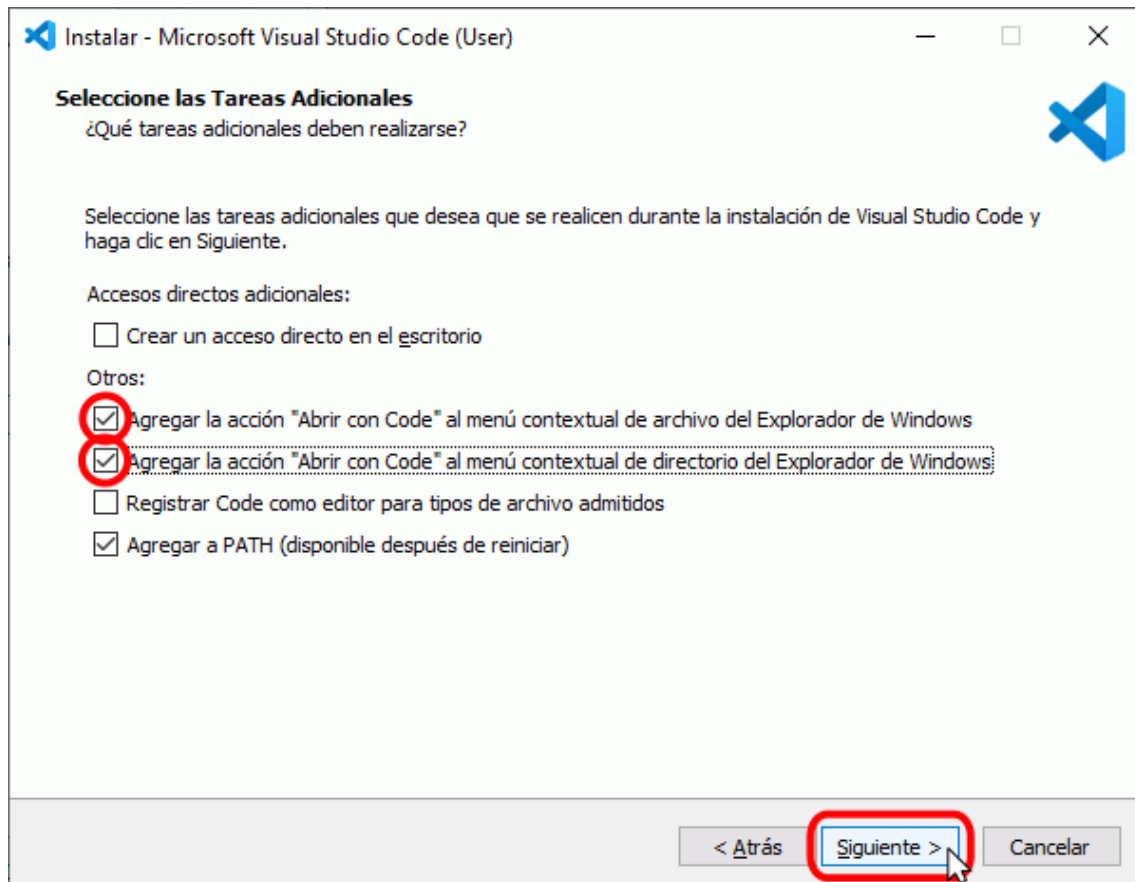
3. La segunda pantalla permite elegir el directorio de instalación (por tratarse de la versión *User installer*, el directorio de instalación está en la carpeta de usuario, no en Archivos de programa):



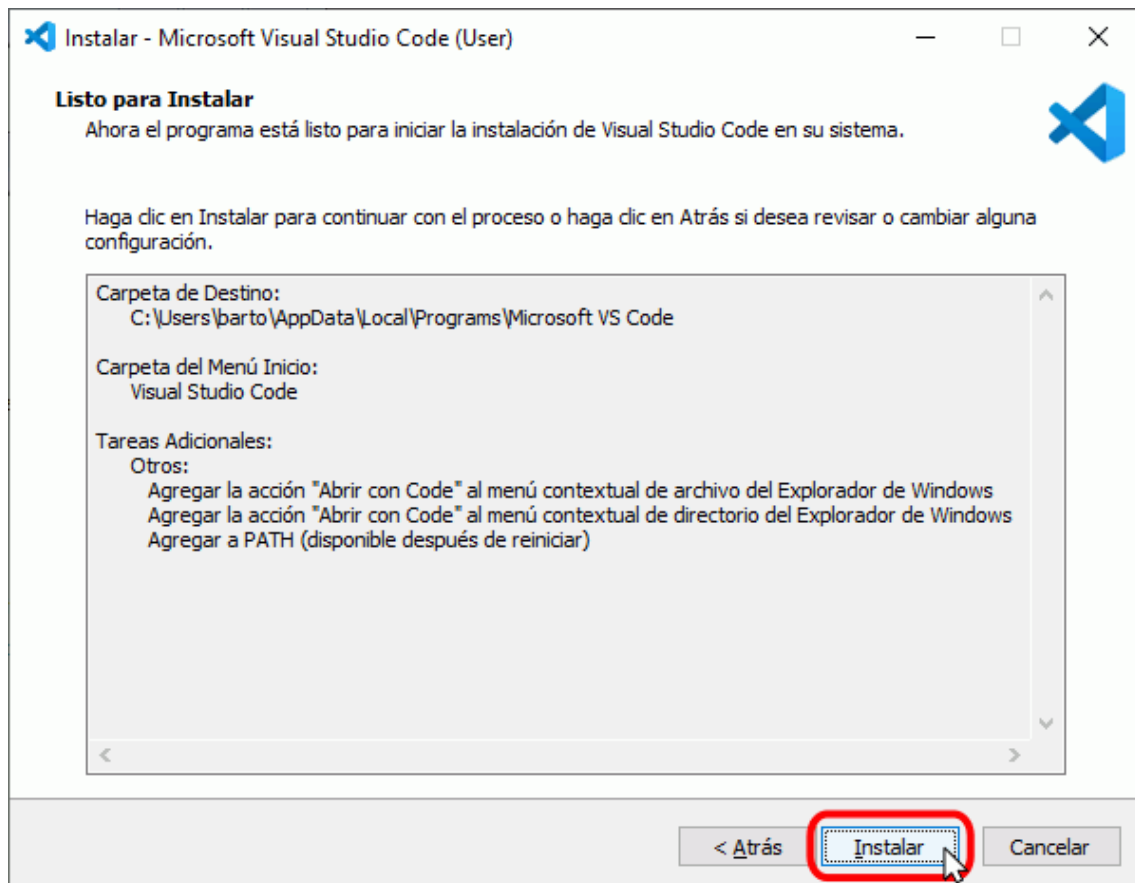
4. La tercera pantalla permite elegir el nombre de la carpeta del menú de inicio:



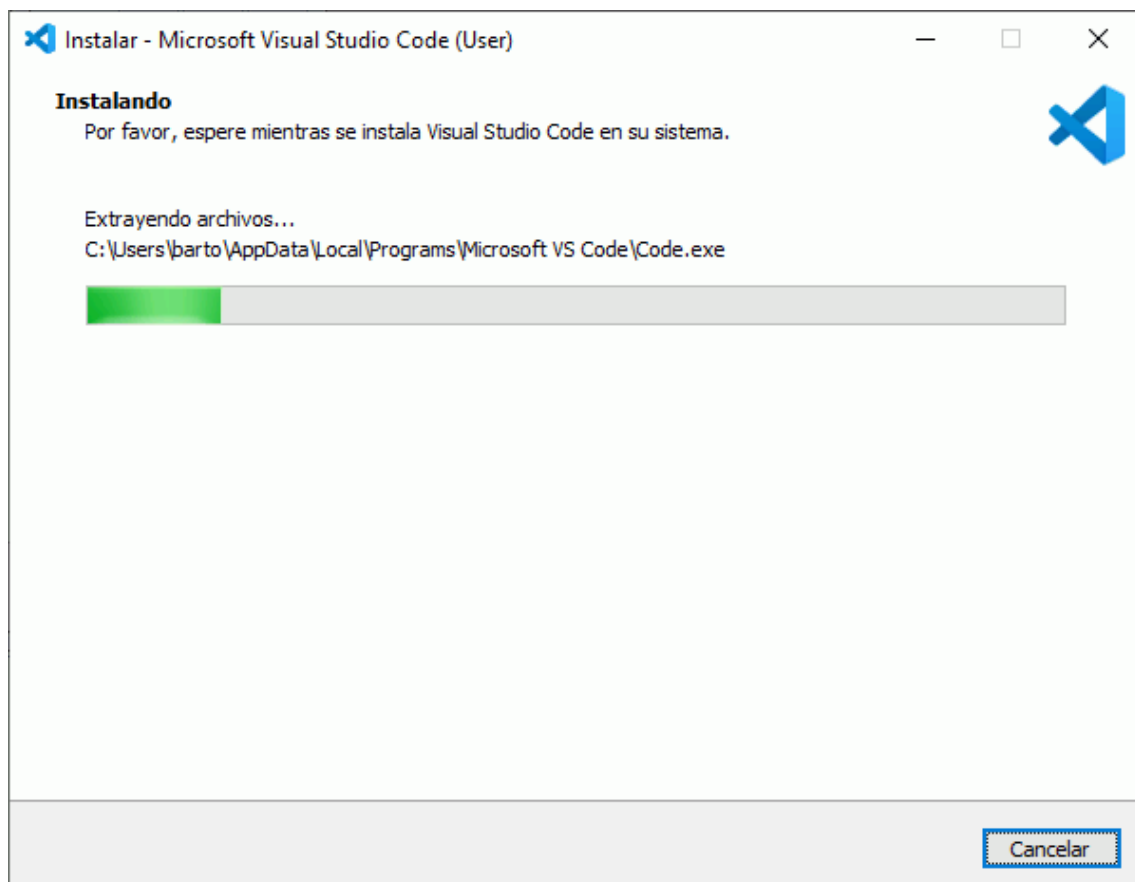
5. La cuarta pantalla permite elegir algunas tareas adicionales tras la instalación. Personalmente, aconsejo marcar las casillas "Agregar la acción ...":



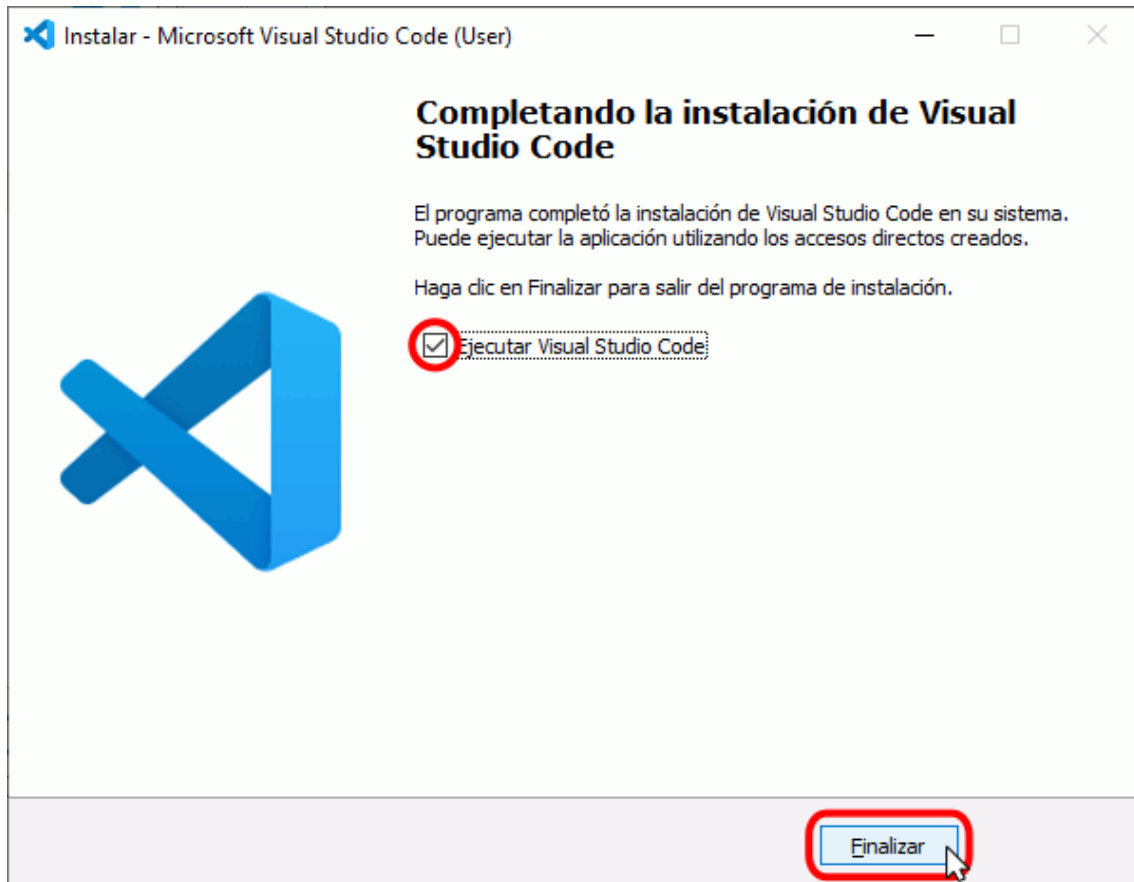
6. Finalmente se muestran las opciones elegidas en las pantallas anteriores. Para iniciar la instalación, haga clic en Instalar.



7. A continuación, se instalará Visual Studio Code.

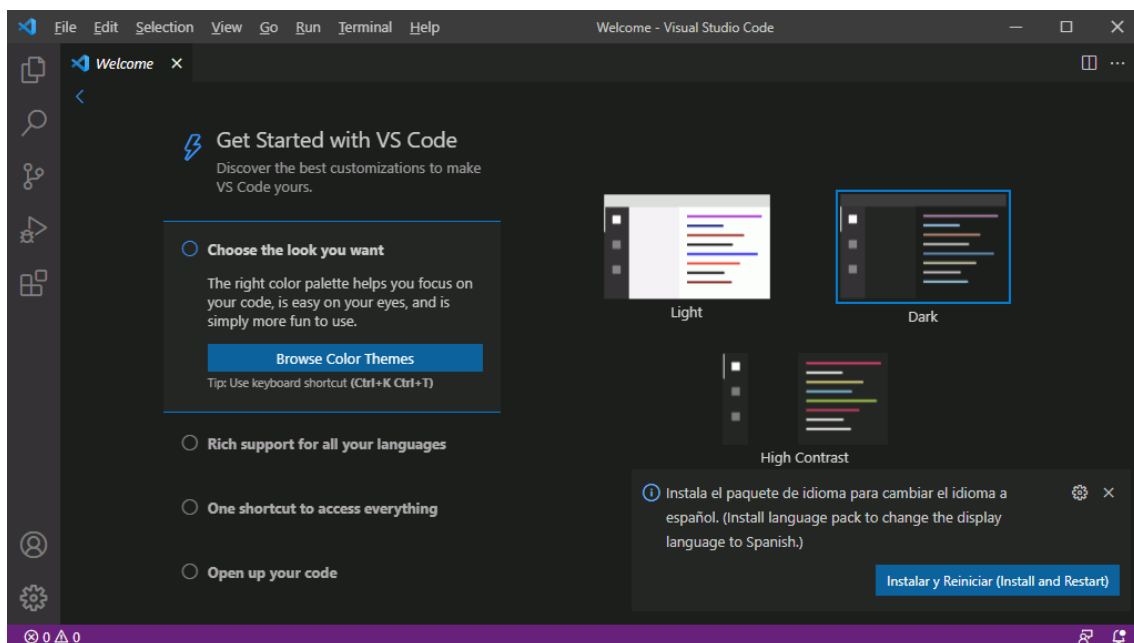


8. Una vez completada la instalación, se muestra la pantalla final. Si va a utilizar [Git](#) con Visual Studio Code, desmarque la casilla "Ejecutar Visual Studio Code", haga clic en Finalizar e [instale Git](#).



### Primera ejecución

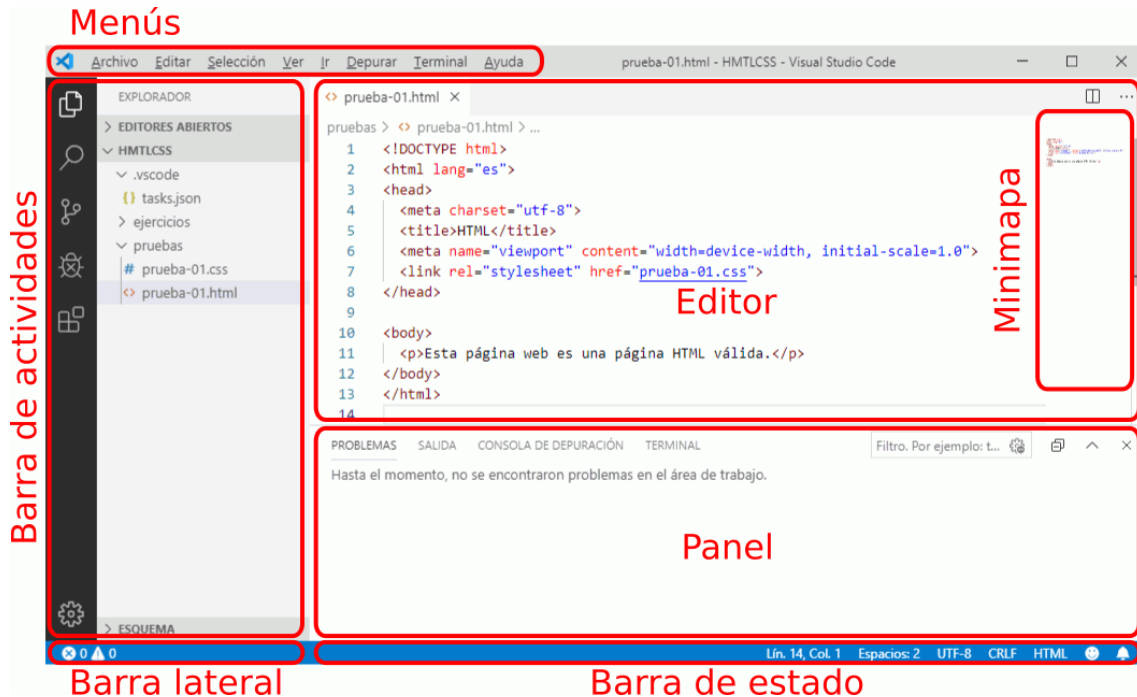
La primera vez que se abre Visual Studio Code tras la instalación, se muestra una página de bienvenida al programa:



## VISUAL STUDIO CODE. USO

### Interfaz de Visual Studio Code

Estos son los elementos principales del interfaz de Visual Studio Code:



Referencia: [Visual Studio Code Docs: User Interface](#)

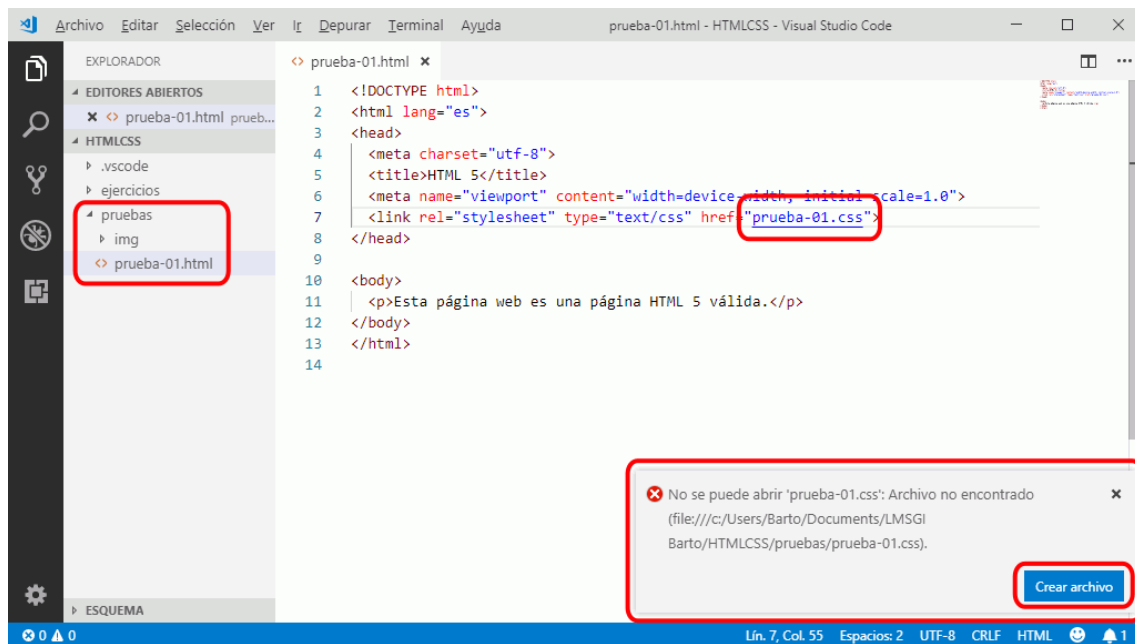
### Abrir enlace

Para abrir un documento enlazado, haga **Ctrl+clic** en la URL. Si es un documento local, el documento se abrirá en Visual Studio Code en una nueva pestaña. Si es un documento externo, el documento se abrirá en el navegador predeterminado.

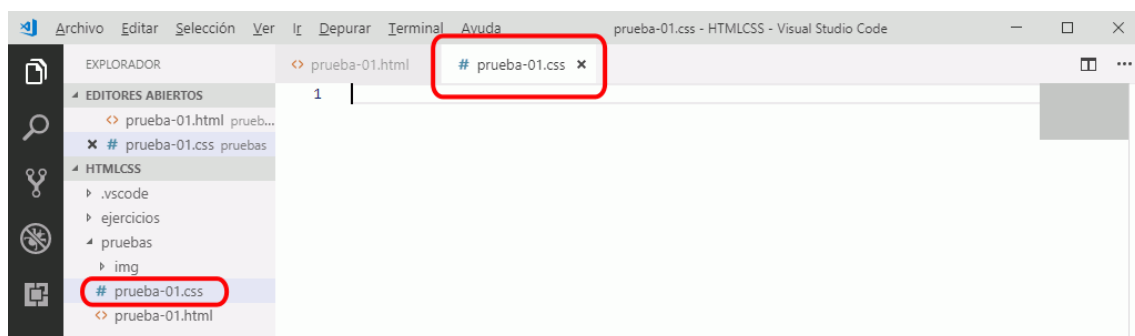
Si el documento no existe, Visual Studio Code ofrece la posibilidad de crearlo, como muestra el ejemplo siguiente.

- La página web del ejemplo contiene un enlace a la hoja de estilo, que no existe. Al hacer **Ctrl+clic** en el nombre de la hoja de estilo, Visual Studio Code muestra un aviso de error que permite la posibilidad de crear el archivo:





- Al hacer clic en "Crear archivo", el archivo se crea automáticamente y se abre en una nueva pestaña:

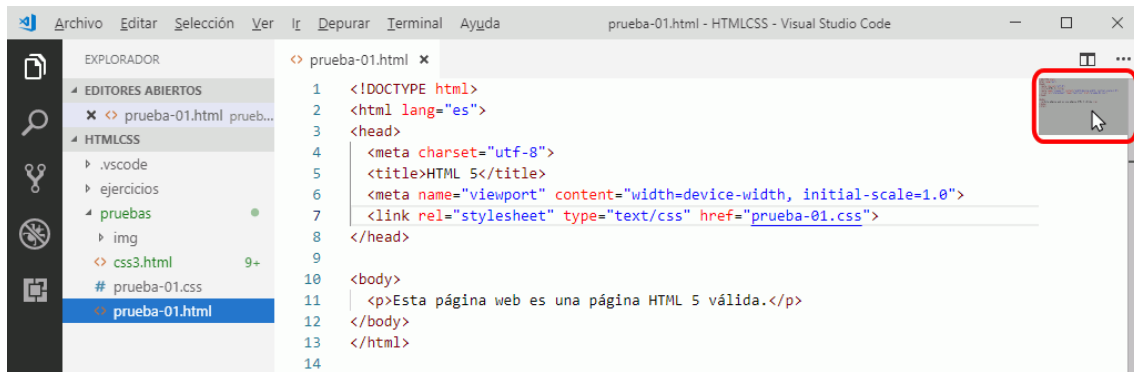


## Ocultar / mostrar minimapa

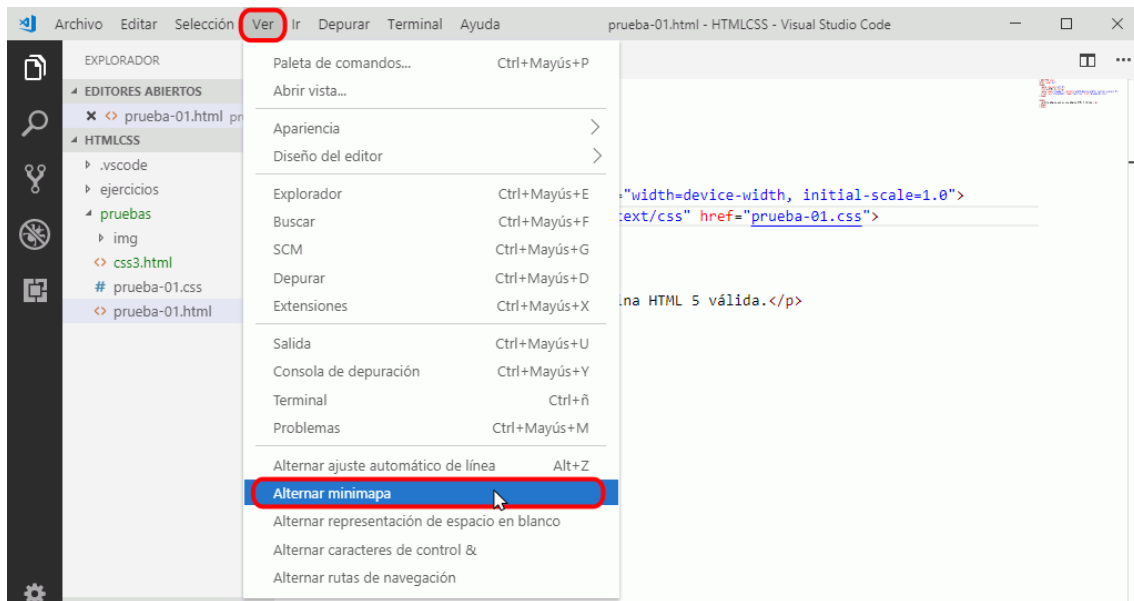
El minimapa es una imagen reducida de la página web que se muestra en la parte derecha de Visual Studio Code y que permite desplazarse rápidamente a lo largo de un documento.



Si el documento no es muy largo (como ocurre en los ejercicios de este curso), el minimapa no resulta demasiado útil, pero ocupa espacio de pantalla.



Para ocultar (o mostrar si está oculto) el minimapa, elija la opción de menú Ver > Alternar minimapa:

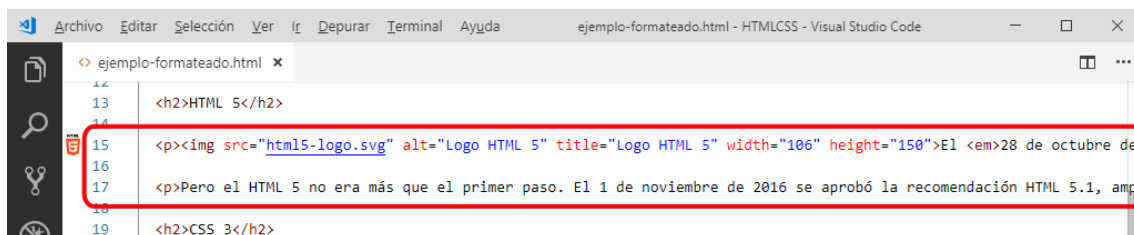


Una vez oculto el minimapa, la ventana de edición se extenderá hasta el borde derecho:

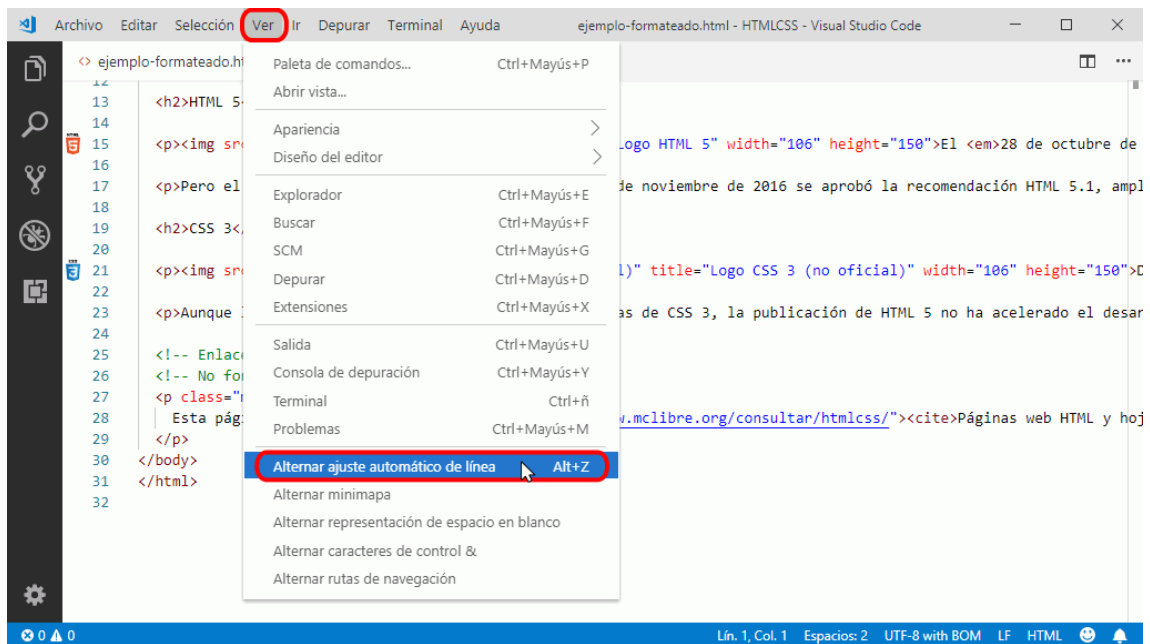


## Ajuste de línea

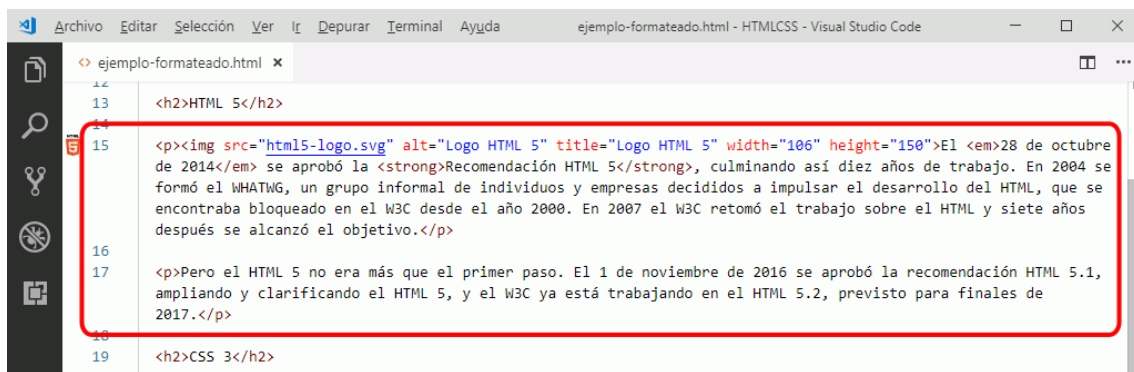
Inicialmente, Visual Studio Code no ajusta las líneas, por lo que las líneas largas no se ven completas.



Para que Visual Studio Code ajuste automáticamente las líneas, elija la opción de menú Ver > Alternar ajuste automático de línea:



Con el ajuste de líneas, las líneas largas se verán completas. El número de línea situado a la izquierda se mantiene para la toda la línea:



## Formatear código

Para formatear código, pulse el atajo de teclado **Alt+Shift+f**.

Referencia: [Visual Studio Code Docs: Formatear fragmentos](#)

## Pegar código

Al pegar código, Visual Studio Code añade un sangrado que puede resultar molesto en caso de que un formato automático posterior no lo corrija (por ejemplo, si el código pegado incluye un elemento `<pre>`).

La solución es pulsar **Ctrl+z** (deshacer) inmediatamente después de pegar el código. De esa manera se eliminará el sangrado añadido.

[VSC issue #6392: Copy/Cut then Paste doesn't preserve or fix indentation](#)

## Sangrado incorrecto

A veces el tamaño del sangrado no coincide con el establecido en la configuración.

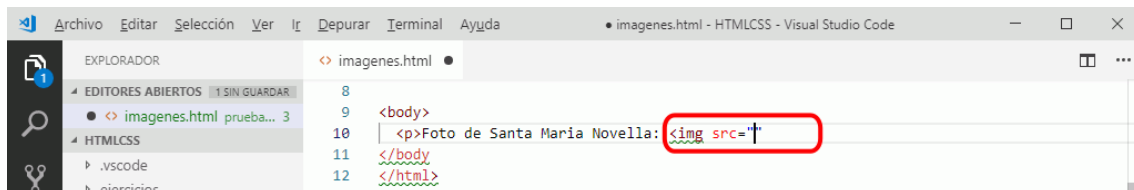
Referencia: [Pregunta en Stack Overflow](#)

## Insertar rutas de archivos

Cuando se escribe una ruta relativa en algún atributo (en una imagen, en un enlace, etc), se muestra un selector con los archivos y directorios disponibles. Al seleccionar uno de ellos, el nombre se añade a la ruta. Si con la lista visible escribimos uno o más caracteres, la lista muestra únicamente los archivos y directorios que comienzan por esos caracteres.

Si las imágenes se encuentran en un subdirectorio del directorio actual:

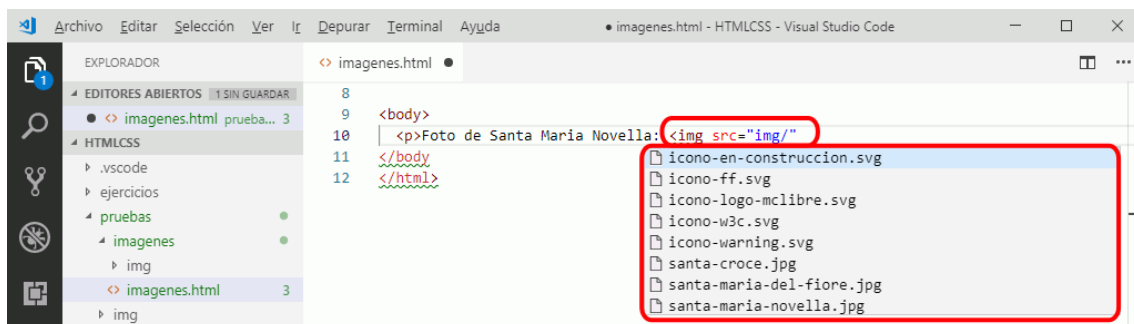
1. Al escribir las comillas en el atributo src ...:



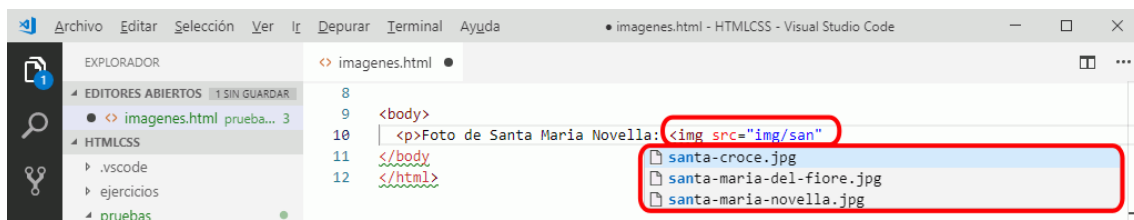
2. ... se mostrará un cuadro con el contenido del directorio actual con uno de los elementos resaltado en azul claro:



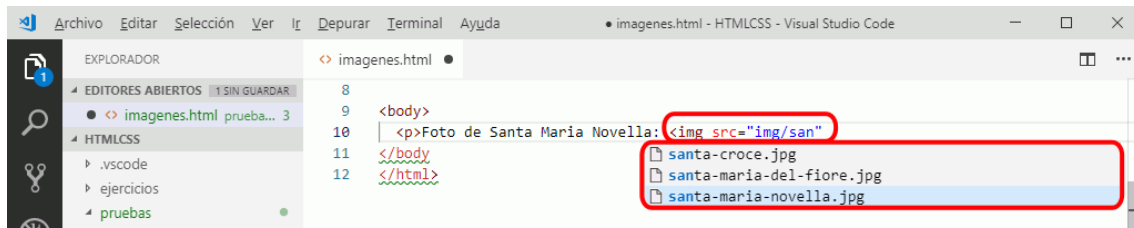
3. Si la ruta que desea escribir es precisamente la que está resaltada en azul claro, pulse **Intro**. La ruta se añadirá automáticamente y si se trata de un directorio (como en el ejemplo), se mostrará el contenido:



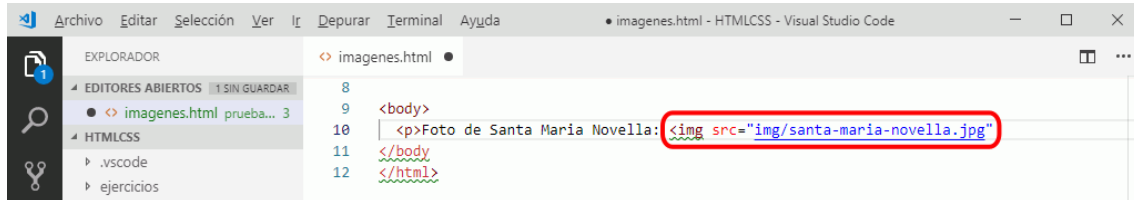
4. Si escribe caracteres, el cuadro mostrará únicamente los ficheros cuyo nombre coincida con lo escrito:



5. En vez de escribir el nombre completo, en cualquier momento puede desplazar con las teclas de flecha arriba o abajo la franja azul hasta el nombre deseado ...:



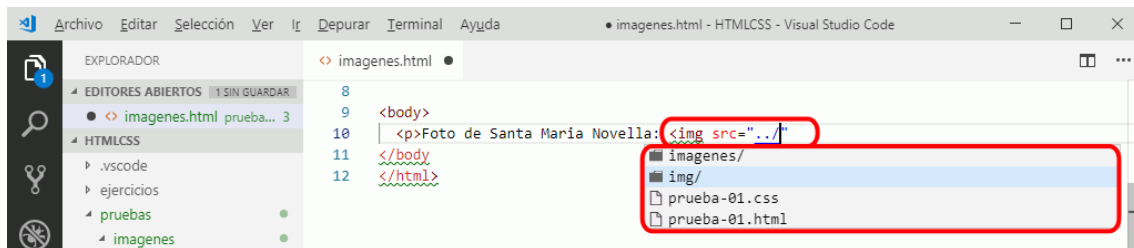
6. ... y pulsar **Intro** para que se escriba en la página el nombre seleccionado en azul:



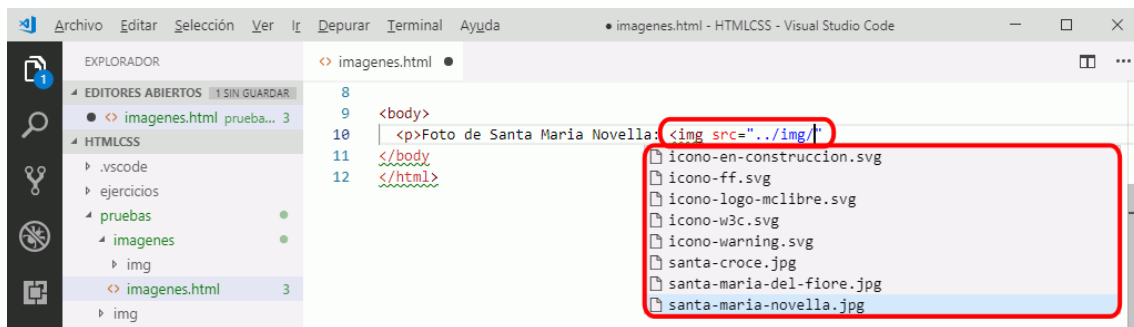
Si en algún momento no se muestra el cuadro con los nombres de los ficheros, pulse **Ctrl+Espacio** para mostrarlo.

Si las imágenes se encuentran en un directorio superior al directorio actual:

1. Escriba **../** para indicar que el camino empieza subiendo al directorio superior y se mostrarán los directorios y ficheros contenidos en el directorio superior:



2. A partir de ahí elija o escriba el nombre del directorio (o **../** si fuera necesario subir otro directorio) y se mostrará su contenido como en el ejemplo anterior:



## Seleccionar bloques

Se pueden seleccionar bloques de etiquetas completos con los atajos de teclado **Alt+Shift+FlechaDerecha** y **Alt+Shift+FlechaIzquierda**:

Cada vez que se pulsa **Alt+Shift+FlechaDerecha** se van seleccionando los elementos de nivel superior a la selección actual. Cada vez que se pulsa **Alt+Shift+FlechaIzquierda** se vuelve a la selección de nivel inferior anterior.

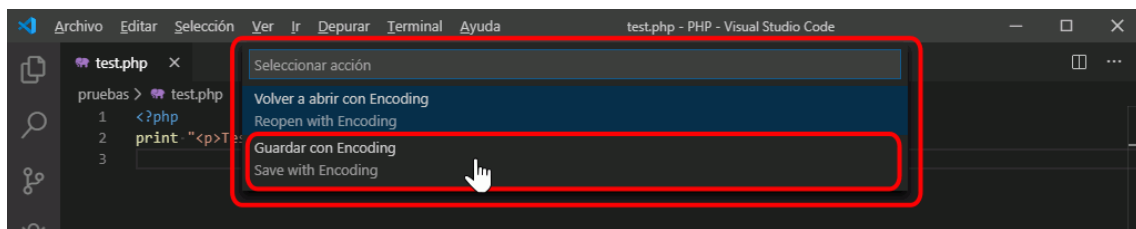
## Modificar el juego de caracteres

Visual Studio Code permite modificar el juego de caracteres de un documento. El ejemplo siguiente muestra cómo cambiar el juego de caracteres de UTF-8 con BOM a UTF (es decir, eliminar la marca de orden de bytes de UTF-8):

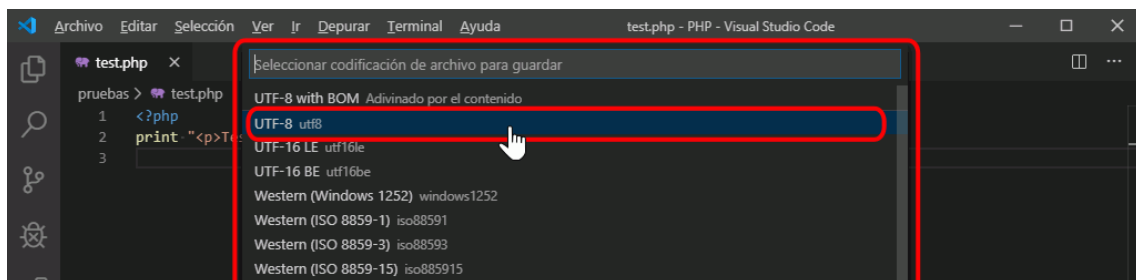
1. Visual Studio Code detecta el juego de caracteres de un documento y muestra el juego de caracteres en la barra de estado:



2. Haga clic en el nombre del juego de caracteres de la barra de estado y se abrirá un menú en la ventana de comandos. Elija la opción Guardar con Encoding:



3. Se abrirá un menú con los juegos de caracteres disponibles. El juego de caracteres actual del documento se mostrará el primero de la lista. Si quiere cambiar el juego de caracteres a UTF-8 (sin BOM), haga clic en la opción UTF-8:



4. Automáticamente el documento se guardará en el juego de caracteres elegido. El nuevo juego de caracteres se mostrará en la barra de estado:



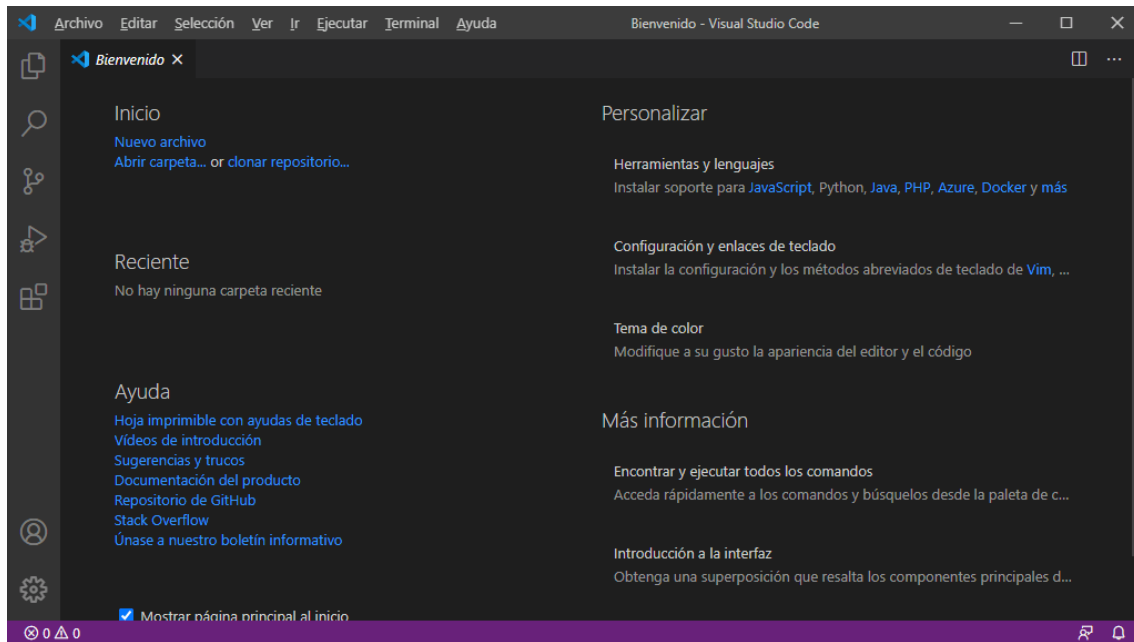
## VISUAL STUDIO CODE. CONFIGURACIÓN RECOMENDADA PARA EL CURSO INTRODUCCIÓN A LA PROGRAMACIÓN CON PYTHON

Esta es la configuración de Visual Studio Code que se recomienda para seguir este curso.

### TEMA DE COLOR RECOMENDADO PARA EL CURSO DE PYTHON

El tema de color es un aspecto muy personal por lo que se deja a elección del alumno el tema de color empleado.

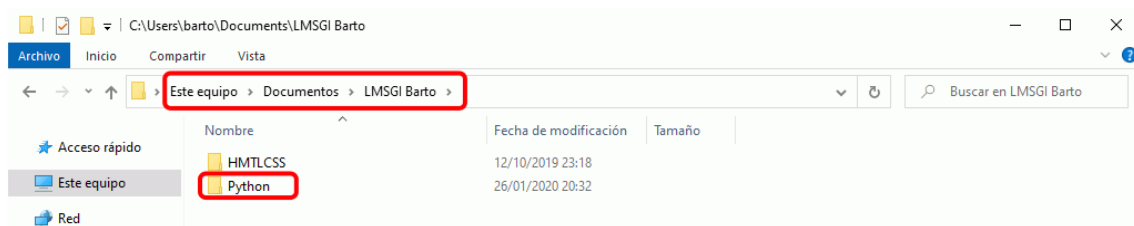
En las capturas de los apuntes se empleará el Tema Dark+ (default dark).



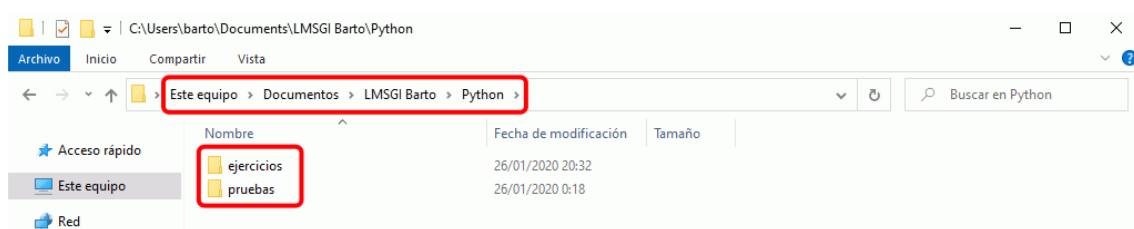
## ÁREA DE TRABAJO

Esta es el área de trabajo recomendada para seguir este curso.

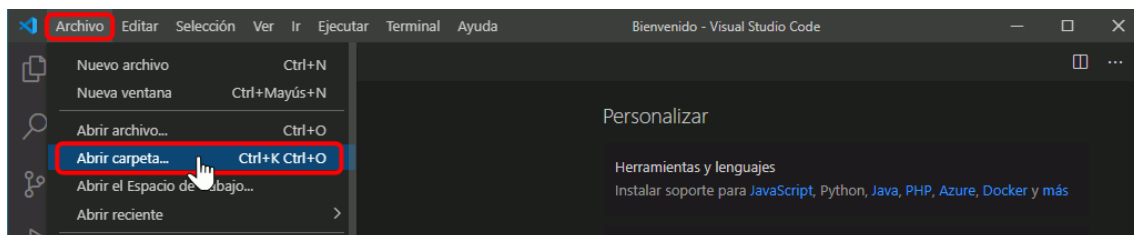
### 1. Cree la carpeta Documentos > LMSGI > Python:



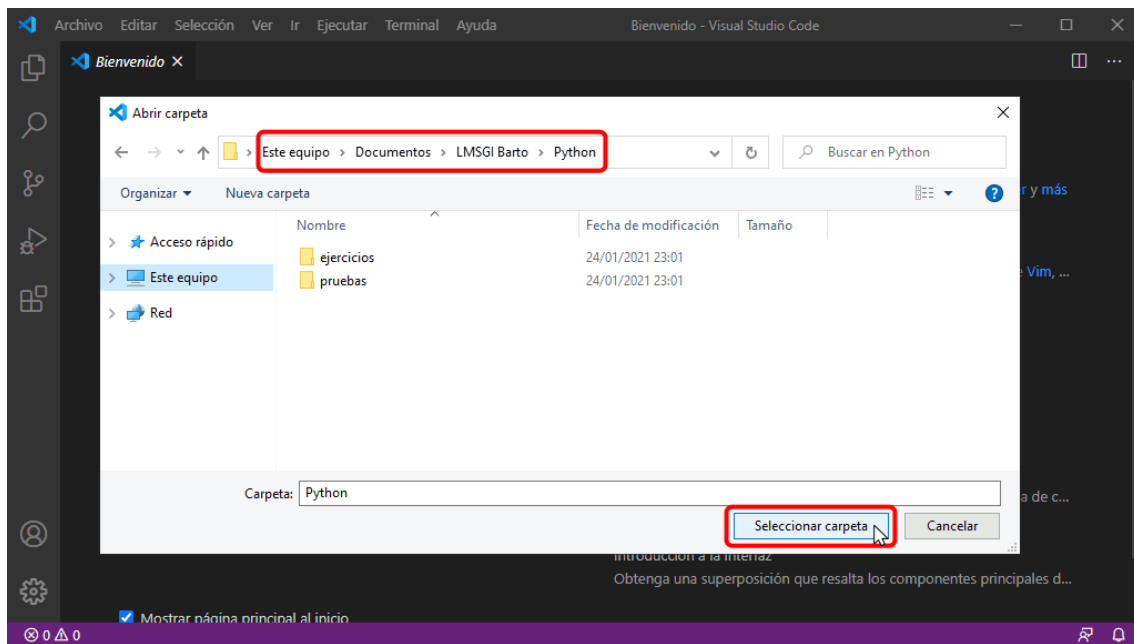
### 2. Cree las carpeta Documentos > LMSGI > Python > ejercicios y Documentos > LMSGI > Python > pruebas:



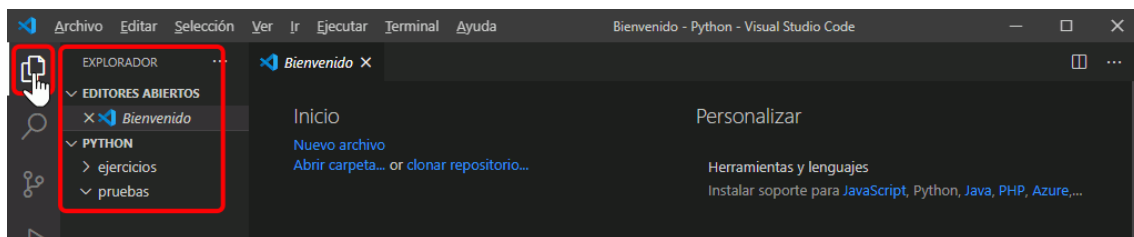
### 3. Elija la opción de menú Archivo > Abrir carpeta ...:



### 4. Elija la carpeta Documentos > LMSGI > Python:



5. Se abrirá en Explorador en la barra de actividades para ver el contenido de la carpeta elegida:



## PREFERENCIAS DE CONFIGURACIÓN RECOMENDADAS PARA EL CURSO DE PYTHON

Estas son las preferencias de configuración que se aconseja establecer para seguir este curso.

Las preferencias propuestas relacionadas con extensiones requieren la instalación de las extensiones correspondientes.

### Cambiar preferencias de configuración

Estos son los pasos a realizar para editar el archivo de preferencias de configuración:

1. Elija la opción de menú Archivo > Preferencias > Configuración, o bien haga clic en el icono Administrar (con forma de rueda dentada) situado en la esquina inferior izquierda y elija la opción de menú Configuración:

```
{
// Configuraciones recomendadas para el curso Introducción a la programación con Python
//
// Files: Insert Final Newline
// Si se habilita, inserte una nueva línea al final del archivo cuando lo guarde.
"files.insertFinalNewline": true,
```



```
//
// Extensión multi-command
"multiCommand.commands": [
{
  "command": "multiCommand.clearTerminal",
  "sequence": [
    {
      "command": "workbench.action.terminal.sendSequence",
      "args": {
        "text": "clear\u000D"
      }
    },
    "workbench.action.focusActiveEditorGroup",
  ],
},
{
  "command": "multiCommand.pythonExecuteInTerminal",
  "sequence": [
    "workbench.action.terminal.focus",
    {
      "command": "workbench.action.terminal.sendSequence",
      "args": {
        "text": "cd '${fileDirname}'\u000Dclear\u000Dpy '${fileBasename}'\u000D"
      }
    },
    "workbench.action.focusActiveEditorGroup",
  ],
},
{
  "command": "multiCommand.pythonExecuteAndFocusInTerminal",
  "sequence": [
    "workbench.action.terminal.focus",
    {
```

```

        "command": "workbench.action.terminal.sendSequence",
        "args": {
            "text": "cd '${fileDirname}'\u000Dclear\u000Dpy '${fileBasename}'\u000D"
        }
    },
],
},
{
    "command": "multiCommand.pythonMptcInTerminal",
    "sequence": [
        "workbench.action.terminal.focus",
        {
            "command": "workbench.action.terminal.sendSequence",
            "args": {
                "text": "cd '${fileDirname}'\u000Dclear\u000Dmptc '${fileBasename}' "
            }
        }
    ],
},
],
//
// Extensi3n pylint
"python.linting.pylintArgs": [
    "--extension-pkg-whitelist=pygame",
    "--disable=C0103, C0114, C0115, C0116",
    // Pylint features. Pylint global options and switches
    // http://pylint.pycqa.org/en/2.4/technical_reference/features.html
    // http://pylint.pycqa.org/en/latest/technical_reference/features.html
    // C0103: invalid-name - Used when the name doesn't conform to naming rules associated to its
    type (constant, variable, class...).
    // C0114: missing-module-docstring - Used when a module has no docstring. Empty modules do
    not require a docstring.
    // C0115: missing-class-docstring - Used when a class has no docstring. Even an empty class must
    have a docstring.

```

*// C0116: missing-function-docstring - Used when a function or method has no docstring. Some special methods like `__init__` do not require a docstring.*

```
],  
  
//  
  
}
```

## EXTENSIONES RECOMENDADAS PARA EL CURSO DE PYTHON

Estas son las dos extensiones que se aconseja instalar para seguir este curso.

- multi-command
- Python extension for Visual Studio Code

### MULTI-COMMAND

Autor: ryuta46

Marketplace: [multi-command](#)

Repositorio GitHub: [multi-command](#)

---

Esta extensión permite crear comandos que ejecuten a su vez varios comandos de Visual Studio Code.

### PYTHON EXTENSION FOR VISUAL STUDIO CODE



Marketplace: [Python](#)

Repositorio GitHub: [Python](#)

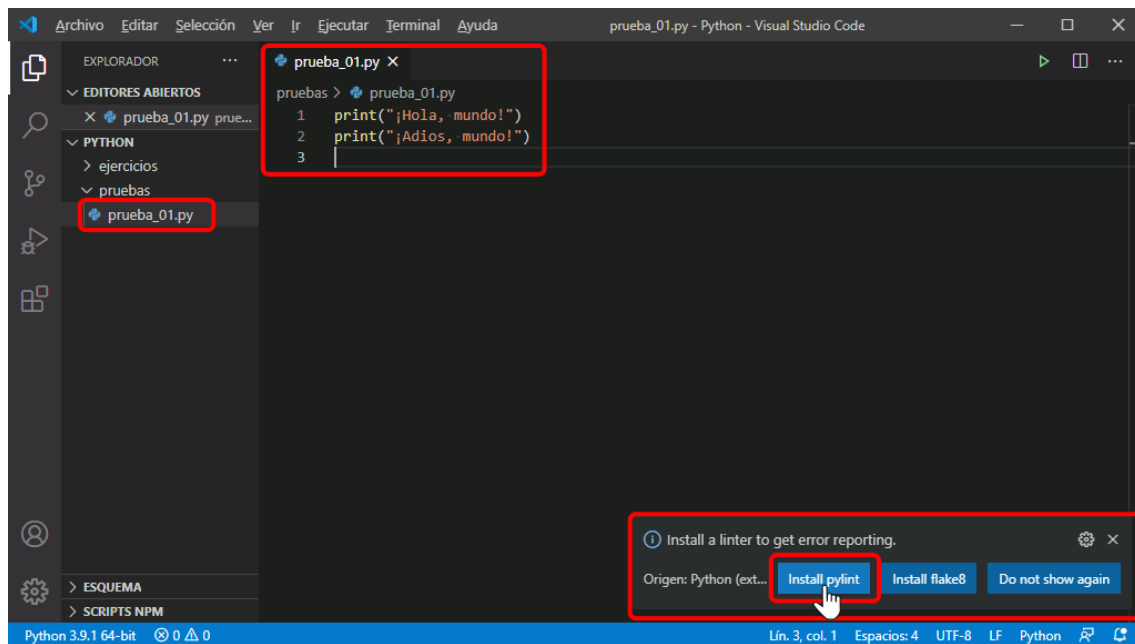
---

Al instalar la extensión Python en Visual Studio Code, se debe instalar el módulo `pylint` y es conveniente instalar un módulo de autoformato.

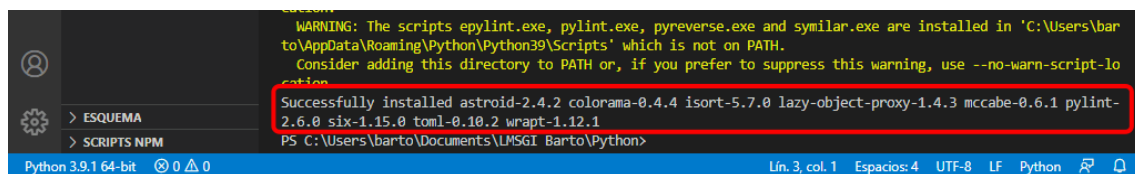
### INSTALACIÓN DE PYLINT

Al guardar por primera vez un programa de Python, Visual Studio Code mostrará un aviso indicando que no hay ningún `linter` instalado y ofrecerá la opción de instalar `pylint` automáticamente.

1. Abra o escriba un programa cualquiera y guárdelo.
2. Se mostrará el aviso indicando que no hay ningún `linter` instalado. Haga clic en "Install pylint" para instalar `pylint`.



3. Se abrirá en la parte inferior una ventana de terminal en la que se ejecutará la instalación de pylint y de sus dependencias. Si la instalación se realiza correctamente, se mostrará el aviso correspondiente. Puede cerrar la ventana haciendo clic en el icono de cierre.



Nota: Como puede verse en la captura, se muestra un aviso en color amarillo indicando que los paquetes se han instalado en una carpeta que no está en la variable de entorno de Windows *PATH*. Eso se debe a que VSCode instala los paquetes con la opción `--user`, por lo que los paquetes se instalan en la carpeta `..\AppData\Roaming\Python\...`, mientras que Python está instalado en `..\AppData\Local\Programs\Python\...` En principio no es necesario añadir ninguna carpeta al *PATH* de Windows.

## CONFIGURACIÓN DE PYLINT

La configuración inicial de Pylint en VSCode es una configuración mínima ([documentación de VSCode](#)) que conviene modificar, sobre todo si trabajamos con pygame, para evitar que se muestren muchos avisos irrelevantes.

La lista de mensajes de error y avisos de Pylint se encuentra en el apartado [Pylint Features de la documentación de Pylint](#).

Nota: Estas preferencias están incluidas en el apartado anterior de preferencias de configuración. No es necesario añadirlas si ya se han añadido anteriormente.

//

// Extensión pylint

"python.linting.pylintArgs": [

```

"--extension-pkg-whitelist=pygame",

"--disable=C0103, C0114, C0115, C0116",

// Pylint features. Pylint global options and switches

// http://pylint.pycqa.org/en/2.4/technical_reference/features.html

// http://pylint.pycqa.org/en/latest/technical_reference/features.html

// C0103: invalid-name - Used when the name doesn't conform to naming rules associated to its
type (constant, variable, class...).

// C0114: missing-module-docstring - Used when a module has no docstring.Empty modules do not
require a docstring.

// C0115: missing-class-docstring - Used when a class has no docstring.Even an empty class must
have a docstring.

// C0116: missing-function-docstring - Used when a function or method has no docstring.Some
special methods like __init__ do not require a docstring.

],

//

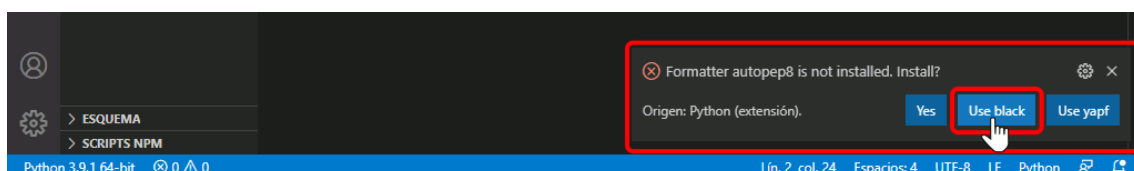
//

```

## INSTALACIÓN DE MÓDULO DE AUTOFORMATO

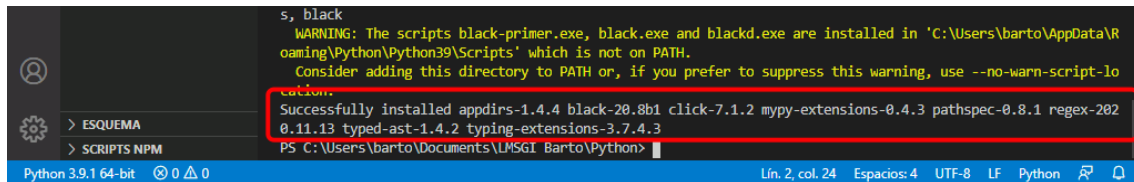
Al formatear código por primera vez después de instalar la extensión Python, Visual Studio Code mostrará un aviso indicando que no hay ningún formateador de código de Python instalado y que le ofrecerá la opción de instalar algún módulo de autoformato. Actualmente (enero de 2021), los módulos de autoformato que ofrece son:

- [autopep8](#)
  - [black](#)
  - [yapf](#)
1. Abra o escriba un programa cualquiera y teclee el atajo Shift+Alt+f para formatear el código.
  2. Se mostrará el aviso indicando que no hay ningún formateador instalado y ofreciendo la instalación de autopep8, pero en estos apuntes se recomienda el uso de black. Haga clic en "Use black" para instalarlo.



Nota: En estos apuntes se recomienda el módulo black, pero se podría utilizar otro módulo. El formato aplicado no afecta al resultado ni a la velocidad de ejecución de los programas.

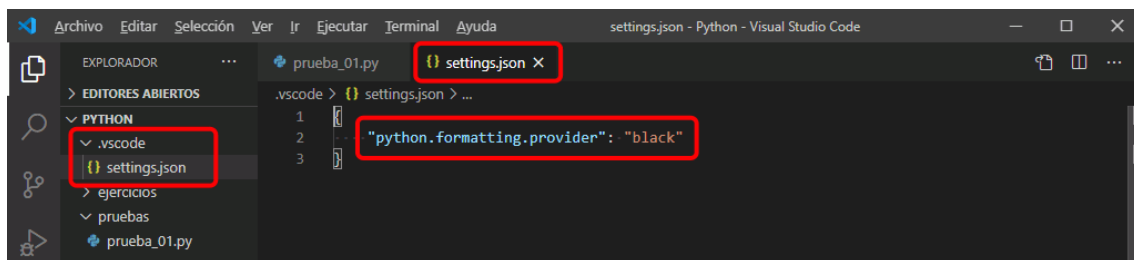
3. Se abrirá en la parte inferior una ventana de terminal en la que se ejecutará la instalación de black y de sus dependencias. Si la instalación se realiza correctamente, se mostrará el aviso correspondiente. Puede cerrar la ventana haciendo clic en el icono de cierre.



```
s, black
WARNING: The scripts black-primer.exe, black.exe and blackd.exe are installed in 'C:\Users\barto\AppData\Roaming\Python\Python39\Scripts' which is not on PATH.
Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed appdirs-1.4.4 black-20.8b1 click-7.1.2 mypy-extensions-0.4.3 pathspec-0.8.1 regex-2020.11.13 typed-ast-1.4.2 typing-extensions-3.7.4.3
PS C:\Users\barto\Documents\LM5GI Barto\Python>
```

Nota: Como puede verse en la captura, se muestra un aviso en color amarillo indicando que los paquetes se han instalado en una carpeta que no está en la variable de entorno de Windows *PATH*. Eso se debe a que VSCode instala los paquetes con la opción *--user*, por lo que los paquetes se instalan en la carpeta *..\AppData\Roaming\Python\...*, mientras que Python está instalado en *..\AppData\Local\Programs\Python\...* En principio no es necesario añadir ninguna carpeta al *PATH* de Windows.

4. Se creará una carpeta *.vscode* con el archivo de configuración del área de trabajo *settings.json*. Este archivo contiene la preferencia de configuración que indica a Visual Studio Code que queremos utilizar el formateador black.



## CREACIÓN DE LA PRIMERA APLICACIÓN PYTHON

Con Python y las herramientas instaladas, ahora puede crear su primera aplicación de Python. Creará un directorio vacío, lo abrirá en Visual Studio Code y, a continuación, creará la primera aplicación.

### Paso 2: Crear un nuevo archivo de Python y agregar código

Con Visual Studio Code abierto y la carpeta vacía creada, ahora creará un archivo de Python para mostrar el mensaje de clase Hola mundo.

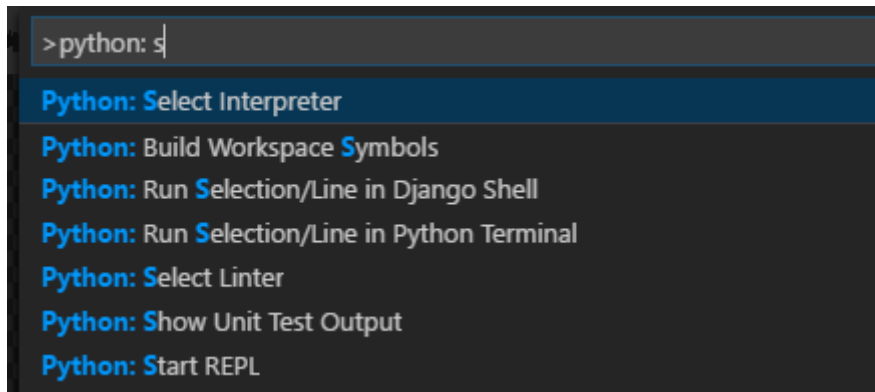
Es posible que vea un mensaje para instalar PYLINT, que puede instalar seleccionando Instalar.

<https://pylint.org/>

Si Visual Studio Code no DETECTA AUTOMÁTICAMENTE UN INTÉRPRETE DE PYTHON, puede que se le pida que SELECCIONE UNO eligiendo la opción predeterminada.

## SELECCIONE Y ACTIVE UN ENTORNO

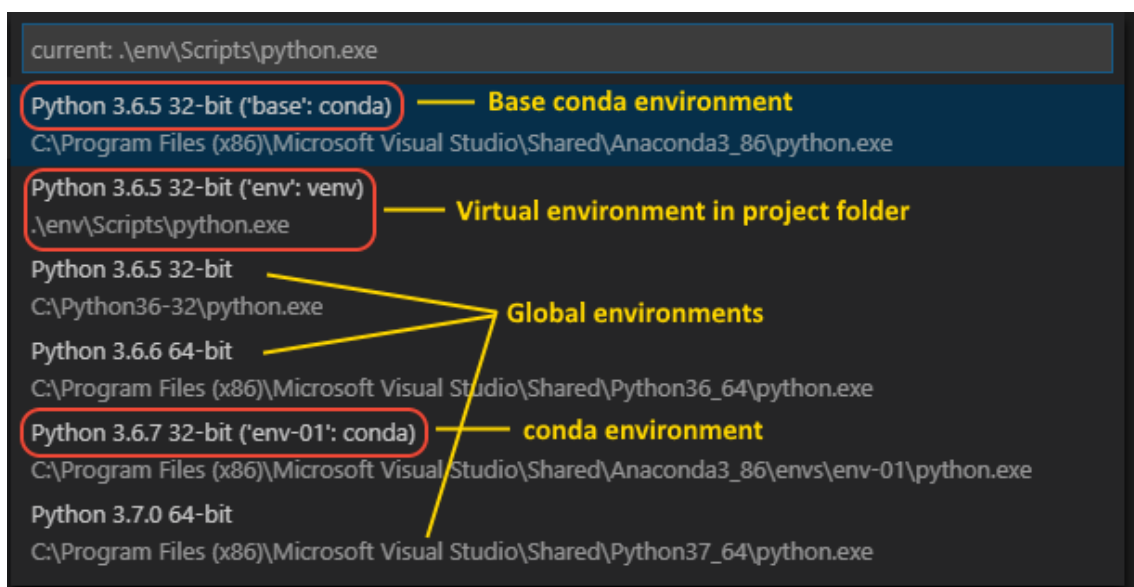
De forma predeterminada, la extensión de Python busca y usa el primer intérprete de Python que encuentra en la ruta del sistema. Para seleccionar un entorno específico, use el comando **Python: Seleccionar intérprete** de la **paleta de comandos** ( Ctrl + Shift + P ).



**Nota** : Si la extensión de Python no encuentra un intérprete, emite una advertencia. En macOS, la extensión también emite una advertencia si está utilizando el intérprete de Python instalado en el sistema operativo, porque normalmente desea utilizar un intérprete que instala directamente. En cualquier caso, puede deshabilitar estas advertencias estableciendo `python.disableInstallationCheck` a `true` en su [configuración de usuario](#) .

Puede cambiar de entorno en cualquier momento; El cambio de entornos le ayuda a probar diferentes partes de su proyecto con diferentes intérpretes o versiones de biblioteca según sea necesario.

El comando **Python: Seleccionar intérprete** muestra una lista de entornos globales, entornos conda y entornos virtuales disponibles. (Consulte la sección [Dónde busca entornos la extensión](#) para obtener detalles, incluidas las distinciones entre estos tipos de entornos). La siguiente imagen, por ejemplo, muestra varias instalaciones de Anaconda y CPython junto con un entorno conda y un entorno virtual ( env ) que se encuentra dentro la carpeta del espacio de trabajo:



**Nota:** En Windows, VS Code puede tardar un poco en detectar los entornos de conda disponibles. Durante ese proceso, es posible que vea "(en caché)" antes de la ruta a un entorno. La etiqueta indica que VS Code está trabajando actualmente con información almacenada en caché para ese entorno.

Si tiene una carpeta o un espacio de trabajo abierto en VS Code y selecciona un intérprete de la lista, la extensión de Python almacenará esa información internamente para que se use el mismo intérprete una vez que vuelva a abrir el espacio de trabajo.

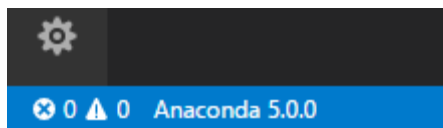
La extensión Python usa el entorno seleccionado para ejecutar código Python (usando el comando **Python: Run Python File in Terminal** ), proporcionando servicios de lenguaje (autocompletar, verificación de sintaxis, linting, formateo, etc.) cuando tiene un .pyarchivo abierto en el editor y abriendo una terminal con el comando **Terminal: Create New Terminal** . En este último caso, VS Code activó automáticamente el entorno seleccionado.

**Consejo** : para evitar la activación automática de un entorno seleccionado, agréguelo "python.terminal.activateEnvironment": false a su settings.jsonarchivo (se puede colocar en cualquier lugar como hermano de la configuración existente).

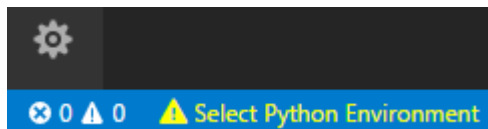
**Sugerencia** : si el comando de activación genera el mensaje "Activate.ps1 no está firmado digitalmente. No puede ejecutar este script en el sistema actual", entonces debe cambiar temporalmente la política de ejecución de PowerShell para permitir que se ejecuten los scripts (consulte [Acerca de las políticas de ejecución](#) en la documentación de PowerShell): Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope Process

**Nota** : De forma predeterminada, VS Code utiliza el intérprete seleccionado para su espacio de trabajo al depurar el código. Puede anular este comportamiento especificando una ruta diferente en la pythonpropiedad de una configuración de depuración. Consulte [Elegir un entorno de depuración](#) .

La barra de estado siempre muestra el intérprete actual.



La barra de estado también refleja cuando no se selecciona ningún intérprete.



En cualquier caso, hacer clic en esta área de la barra de estado es un atajo conveniente para el comando **Python: Seleccionar intérprete** .

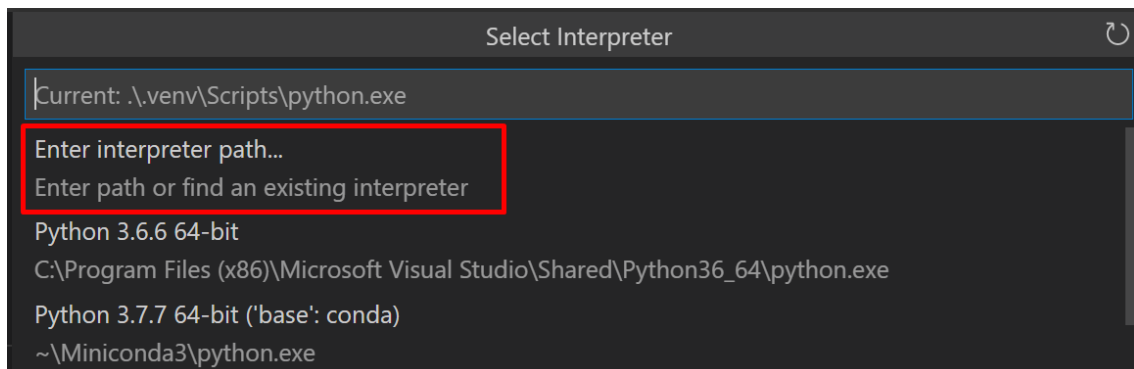
**Consejo** : Si tiene algún problema con VS Code para reconocer un entorno virtual, [presente un problema](#) en el repositorio de extensiones para que podamos ayudar a determinar la causa.

Especifique manualmente un intérprete <#>

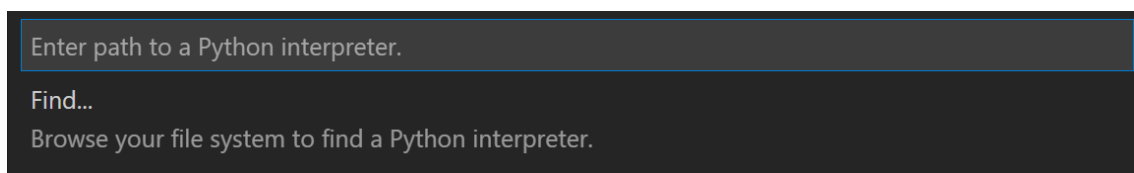
Si VS Code no localiza automáticamente un intérprete que desea utilizar, puede buscar el intérprete en su sistema de archivos o proporcionar la ruta manualmente.

Puede hacerlo ejecutando el comando **Python: Seleccionar intérprete** y haciendo clic en la opción **Ingresar ruta del intérprete ...** que se muestra en la parte superior de la lista de intérpretes:





Luego puede ingresar la ruta completa del intérprete de Python directamente en el cuadro de texto (por ejemplo, ".venv / Scripts / python.exe"), o puede hacer clic en el botón **Buscar ...** y buscar en su sistema de archivos para buscar el ejecutable de Python que desea seleccionar.



Si desea especificar manualmente un intérprete predeterminado que se utilizará una vez que abra su espacio de trabajo por primera vez, puede crear o modificar una entrada para `python.defaultInterpreterPath` en su espacio de trabajo `settings.json` con la ruta completa al ejecutable de Python.

Por ejemplo:

Ventanas:

```
{  
  "python.defaultInterpreterPath": "c:/python39/python.exe"  
}
```

macOS / Linux:

```
{  
  "python.defaultInterpreterPath": "/home/python39/python"  
}
```

También puede utilizar `python.defaultInterpreterPath` para apuntar a un entorno virtual, por ejemplo:

Ventanas:

```
{  
  "python.defaultInterpreterPath": "c:/dev/ala/venv/Scripts/python.exe"  
}
```

macOS / Linux:

```
{  
  
  "python.defaultInterpreterPath": "/home/abc/dev/ala/venv/bin/python"  
  
}
```

**Nota** : Los cambios en la python.defaultInterpreterPath configuración no se recogen después de que ya se haya seleccionado un intérprete para un espacio de trabajo; cualquier cambio en la configuración se ignorará una vez que se seleccione un intérprete inicial para el espacio de trabajo.

Además, si desea configurar un intérprete predeterminado para todas sus aplicaciones de Python, puede agregar una entrada python.defaultInterpreterPath manualmente dentro de su Configuración de usuario. Para hacerlo, abra la paleta de comandos ( Ctrl + Shift + P ) e ingrese **Preferencias: Abrir configuración de usuario** . Luego configure python.defaultInterpreterPath, que se encuentra en la sección de extensión Python de Configuración de usuario, con el intérprete apropiado.

También puede utilizar una variable de entorno en la configuración de la ruta utilizando la sintaxis \${env:VARIABLE}. Por ejemplo, si ha creado una variable nombrada PYTHON\_INSTALL\_LOC con una ruta a un intérprete, puede usar el siguiente valor de configuración:

```
"python.defaultInterpreterPath": "${env:PYTHON_INSTALL_LOC}",
```

**Nota** : La sustitución de variables solo se admite en archivos de configuración de VS Code, no funcionará en .env archivos de entorno.

Al usar una variable de entorno, puede transferir fácilmente un proyecto entre sistemas operativos donde las rutas son diferentes, solo asegúrese de establecer primero la variable de entorno en el sistema operativo.

Entornos y ventanas de terminal <#>

Después de usar **Python: seleccione Intérprete** , ese intérprete se aplica al hacer clic con el botón derecho en un archivo y seleccionar **Python: Ejecutar archivo Python en la terminal** . El entorno también se activa automáticamente cuando utiliza el comando **Terminal: Crear nueva terminal** a menos que cambie la python.terminal.activateEnvironment configuración a false.

Sin embargo, ejecutar VS Code desde un shell en el que se activa un determinado entorno de Python no activa automáticamente ese entorno en la Terminal integrada predeterminada. Utilice el comando **Terminal: Create New Terminal** después de que VS Code se esté ejecutando.

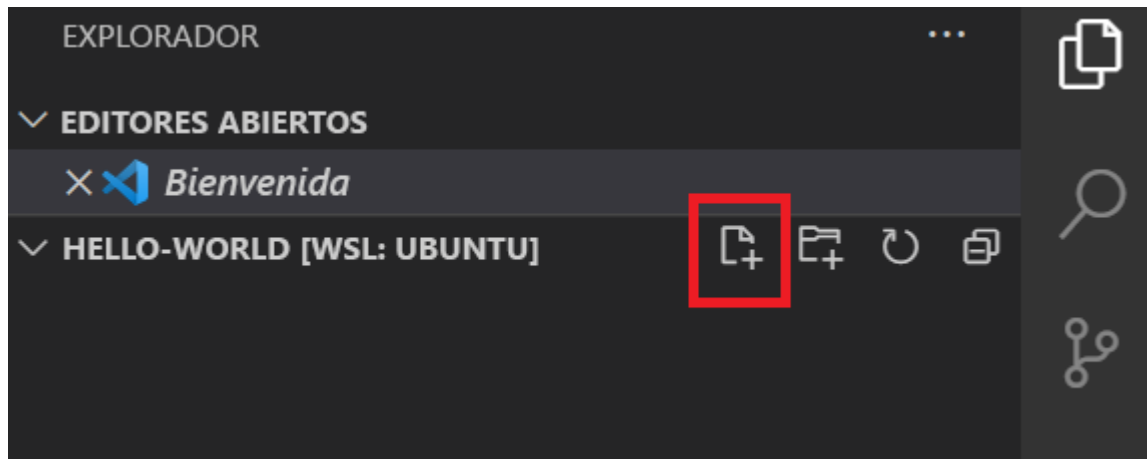
**Nota:** los entornos conda no se pueden activar automáticamente en el terminal integrado si PowerShell está configurado como shell integrado. Consulte [Terminal integrado - Configuración](#) para [saber](#) cómo cambiar el shell.

Cualquier cambio que realice en un entorno activado dentro de la terminal es persistente. Por ejemplo, usar conda install <package> desde la terminal con un entorno conda activado instala el paquete en ese entorno de forma permanente. De manera similar, usar pip installen una terminal con un entorno virtual activado agrega el paquete a ese entorno.

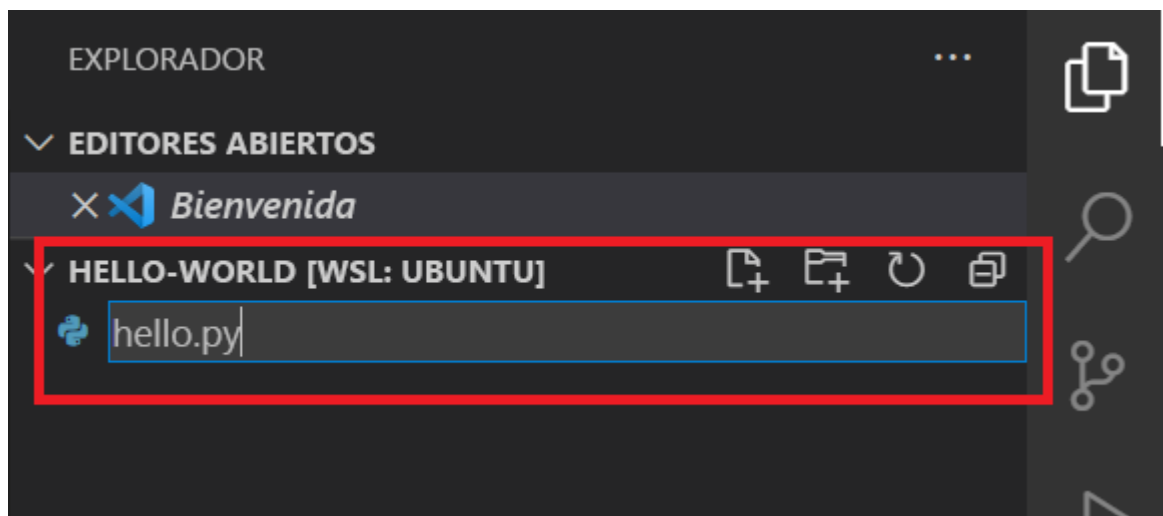
Cambiar de intérpretes con el comando **Python: Seleccionar intérprete** no afecta a los paneles de terminales que ya están abiertos. Por lo tanto, puede activar entornos separados en una terminal dividida: seleccione el primer intérprete, cree una terminal para él, seleccione un intérprete diferente y luego use el botón dividir ( Ctrl + Shift + 5 ) en la barra de título de la terminal.

## CREAR UN NUEVO ARCHIVO DE PYTHON Y AGREGAR CÓDIGO

Dentro de la ventana **Explorador**, seleccione **Nuevo archivo**.



Asigne el nombre `hello.py` al nuevo archivo; para ello, escríbalo en el nuevo cuadro de texto y presione ENTRAR.



Escriba el siguiente código de Python en la ventana del editor, que usa la función `print` para mostrar el texto `Hola, mundo` cuando se ejecuta la aplicación.

PYTHONCopiar

```
print('Hello, World!')
```

Guarde el archivo seleccionando **Archivo** y **Guardar**.

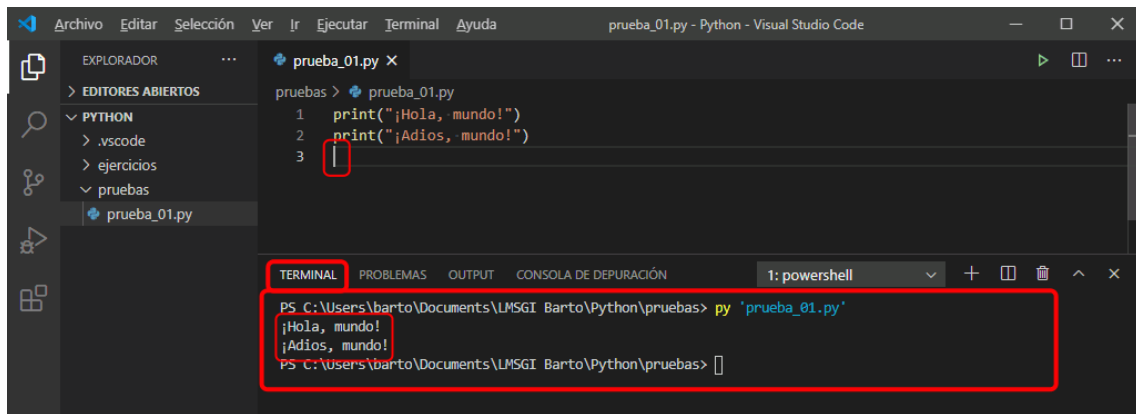
## EJECUTAR PROGRAMA

Los dos atajos (Ctrl+Alt+F5 y Ctrl+Alt+Shift+F5) permiten ejecutar el programa de Python que se esté editando. El programa se ejecuta en la ventana de terminal PowerShell de Visual Studio Code. Se ha elegido la tecla F5 ya que esta es la tecla que permite ejecutar programas en IDLE.

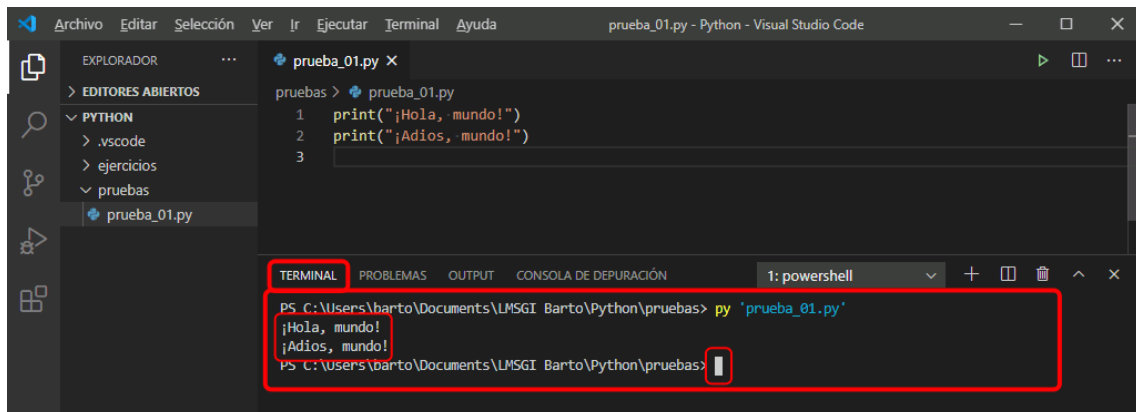
Ambos atajos cambian automáticamente al directorio en el que se encuentra el programa y lo ejecutan en ese directorio.

La diferencia entre ambos atajos es que:

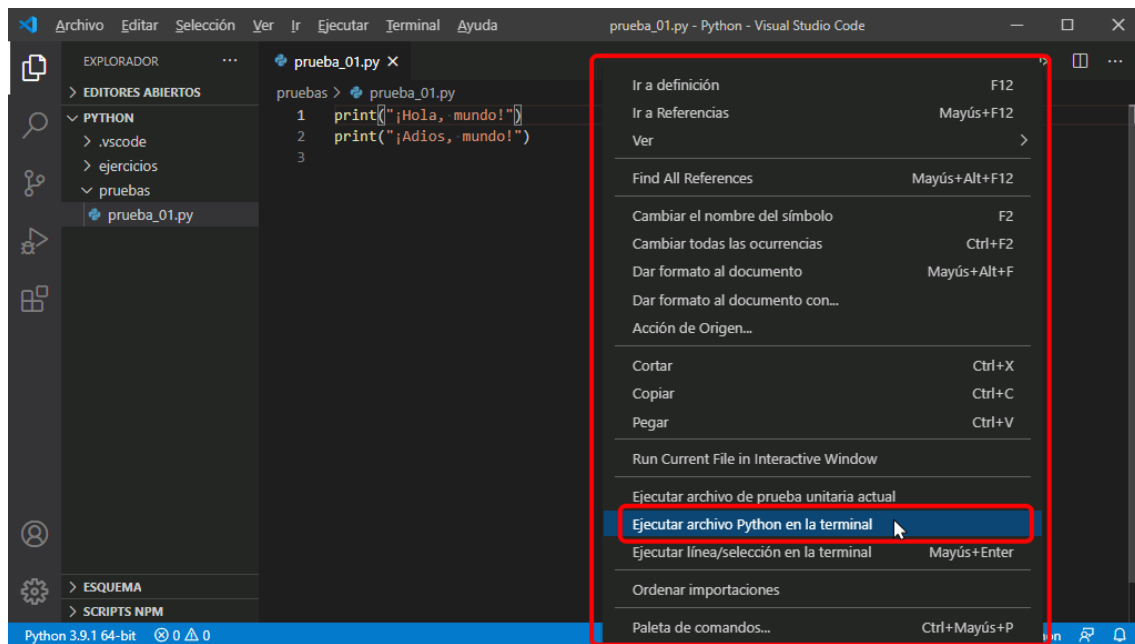
el atajo Ctrl+Alt+F5 mantiene el foco en el editor. Este atajo es útil cuando el programa a ejecutar no pide datos al usuario.



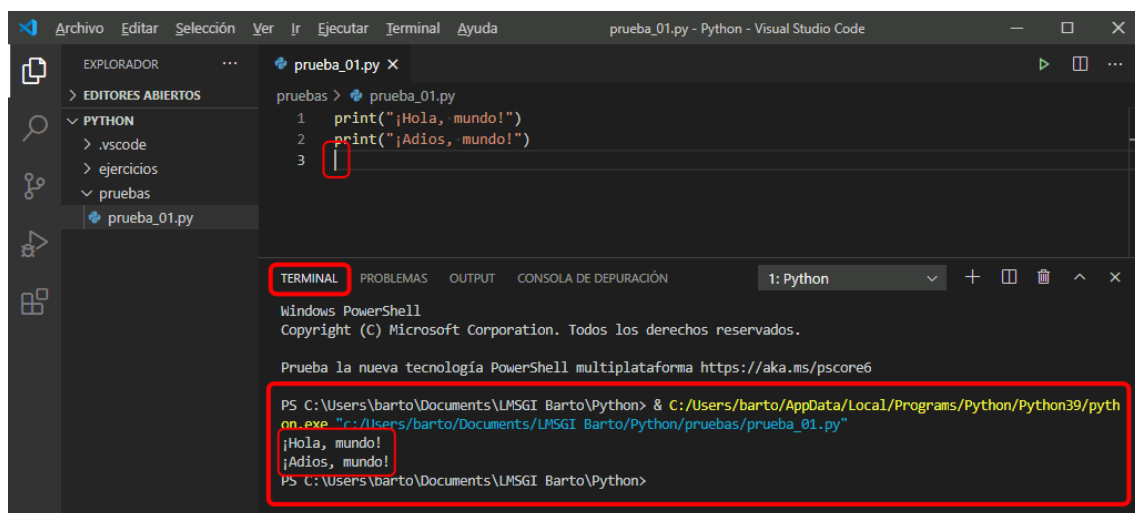
el atajo Ctrl+Alt+Shift+F5 cambia el foco al terminal. Este atajo es útil cuando el programa a ejecutar pide datos al usuario. Una vez ejecutado el programa, para volver al editor se puede utilizar el atajo Ctrl+Alt+.-.



Si no se define el atajo de teclado, el programa se puede ejecutar igualmente haciendo clic derecho en el cuerpo del programa y eligiendo la opción "Ejecutar archivo Python en la terminal".



En este caso, el terminal se abre en el directorio raíz del área de trabajo y el foco se mantiene en el editor.



## OTRAS EXTENSIONES

### [IntelliCode de Visual Studio](#)

La extensión Visual Studio IntelliCode proporciona funciones de desarrollo asistidas por IA para desarrolladores de Python, TypeScript / JavaScript y Java en Visual Studio Code, con información basada en la comprensión del contexto de su código combinado con el aprendizaje automático.

Necesitará Visual Studio Code de octubre de 2018 versión 1.29.1 o posterior para usar esta extensión. Para cada idioma admitido, consulte la sección "Introducción" a continuación para comprender cualquier otro requisito previo que deberá instalar y configurar para obtener finalizaciones de IntelliCode.

Para obtener compatibilidad con C #, C ++, TypeScript / JavaScript y XAML en el IDE de Visual Studio, consulte la extensión IntelliCode en Visual Studio Marketplace .

#### [Python by Microsoft](#)

Esta es la extensión indispensable para toda desarrolladora en Python, pues incluye un linter, debugger, IntelliSense, navegación por el código, formateo efectivo, refactoring, unit testing, snippets, descripción de tu código y puedes crear notebooks de JuPyter.

##### **¿Por qué usarlo?**

Instalarlo te hará la vida muchísimo más fácil sin importar que seas novato o veterano. Así que no dudes de incluirlo ya mismo en tu editor.

#### [Python Docstring Generator](#)

Mantener una buena documentación en tu código es valioso y esta extensión te ayudará a cumplir con tu cometido.

##### **¿Por qué usarlo?**

Pues al colocar comillas dobles dentro de una clase, método o función generará una plantilla para tu docstring y así proveer de la información necesaria de lo que estás creando.



```
class Pizza(models.Model):
    """[summary] You, a few seconds ago • Uncommitted cha

    Args:
        models ([type]): [description]

    Returns:
        [type]: [description]
    """
    name = models.CharField(max_length = 150)
    date_added = models.DateTimeField(auto_now_add = True)

    def __str__(self):
        """Return a string representation of the model."""
        return self.name
```

#### [Code Runner](#)

Es natural que mientras escribimos código lo depuramos mediante su ejecución y para esto mismo es Code Runner. Podrá ejecutar tu código directamente desde la misma ventana sin tener que pasar a la terminal.

##### **¿Por qué usarlo?**

Este plugin es grandioso no solo para Python, pues es compatible con **más de 20 lenguajes de programación distintos**.

#### [4. Trailing Spaces](#)

Otro plugin de uso general es Trailing Spaces, el cual permite ver aquellos espacios vacíos al final o entre líneas para eliminarlos.

##### **¿Por qué usarlo?**

Bastante práctico considerando que Python utiliza la indentación como base de su estructura.

```
class Pizza(models.Model):
    """A pizza included in the restaurant menu"""
    name = models.CharField(max_length = 150)
    # ...
    # ...
    date_added = models.DateTimeField(auto_now_add = True)

    def __str__(self):
        """Return a string representation of the model."""
        return self.name
```

### [Indent Rainbow](#)

Hablando de indentación, hay que estar muy atentos no solo a los espacios en blanco sino también a los ocupados.

#### ¿Por qué usarlo?

Por lo que a través de colores distintos esta extensión nos mostrará la “profundidad” que llevan nuestras líneas de código.

```
class Pizza(models.Model):
    """A pizza included in the restaurant menu"""
    # ...
    # ...
    #* Información importante a revisar
    #! Esto es una alerta, cuidado
    #? Debería hacer X por Z?
    # TODO: Refactorizar clase
    # ...
```

### [Better Comments](#)

Al igual que los docstrings también podemos colocar comentarios que sean efectivos y con este plugin puedes utilizar un formato de color especial para datos importantes, alertas, tareas pendientes o consideraciones.

#### ¿Por qué usarlo?

Better Comments es compatible también con múltiples lenguajes.

```
class Pizza(models.Model):
    """A pizza included in the restaurant menu"""
    ...
    #**Información importante a revisar
    #·!·Esto es una alerta, cuidado
    #·?·Debería hacer X por Z?
    #·TODO: Refactorizar clase
    ...
    name = models.CharField(max_length=150)
    date_added = models.DateTimeField(auto_now_add=True)

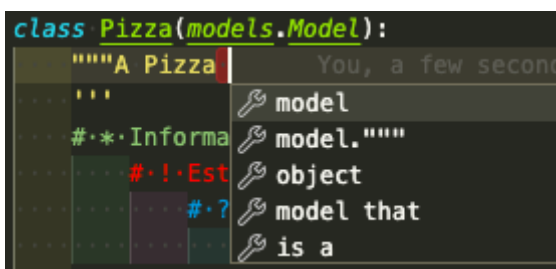
    def __str__(self):
        """Return a string representation of the model."""
        return self.name
```

### [TabNine](#)

Con el poder de la inteligencia artificial este plugin leerá tu archivo .gitignore y basado en ello brindará sugerencias de contenidos. Resulta bastante útil cuando conocemos la estructura de nuestro proyecto o sabemos que vamos a escribir y llevarlo a cabo con un TAB.

#### ¿Por qué usarlo?

Toma en cuenta que a veces su precisión no es la mejor y que puedes cometer errores al aceptar alguna de las recomendaciones sin revisar a detalle. Esto puede implicar trabajo adicional en ocasiones, algo que en su versión de paga se corrige.



```
class Pizza(models.Model):
    """A Pizza
    ...
    #**Informa
    #·!·Est
    #·?·
    ...
```

The suggestions shown are:

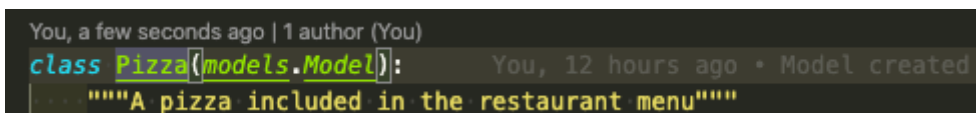
- model
- model."""
- object
- model that
- is a

### [GitLens](#)

Git es muy útil por sí solo, pero con este añadido tendrás toda la información que necesites en tu editor al colocar el selector sobre alguna línea.

#### ¿Por qué usarlo?

Podrás ver quien hizo la última modificación, cuando e incluso navegar entre los cambios.



```
You, a few seconds ago | 1 author (You)
class Pizza(models.Model):
    """A pizza included in the restaurant menu"""
```

The commit history shown is:

- You, 12 hours ago · Model created

## TRABAJAR CON REPOSITORIOS GIT

En esta lección se explica cómo empezar a trabajar en un repositorio git desde Visual Studio Code. Aunque podemos crear el repositorio con Visual Studio Code en nuestro ordenador y después subirlo a GitHub, en esta lección se explica cómo clonar en nuestro ordenador un repositorio ya existente en GitHub.



## INSTALAR GIT EN WINDOWS

Una vez descargado el instalador, haga doble clic en él para iniciar el proceso de instalación.

Personalmente, yo recomiendo cambiar un par de opciones predeterminadas:

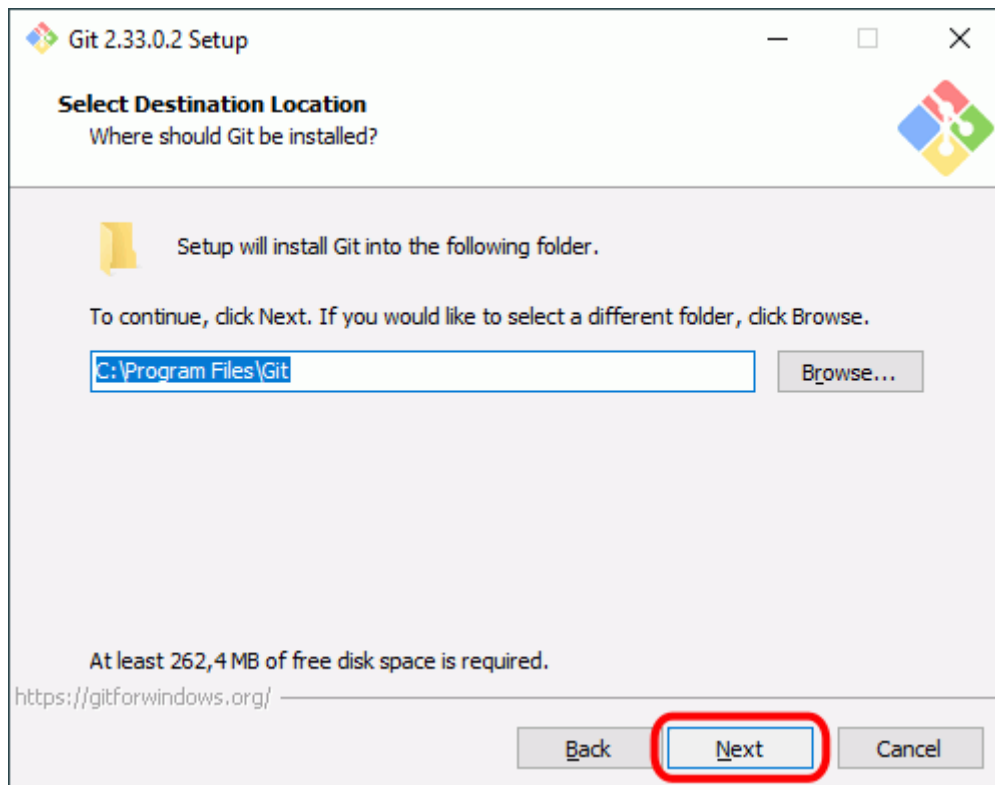
- que Git busque actualizaciones diariamente
- que utilice Visual Studio Code como editor.

Estos son los pasos del proceso de instalación de la versión 2.33, similares a los de otras versiones:

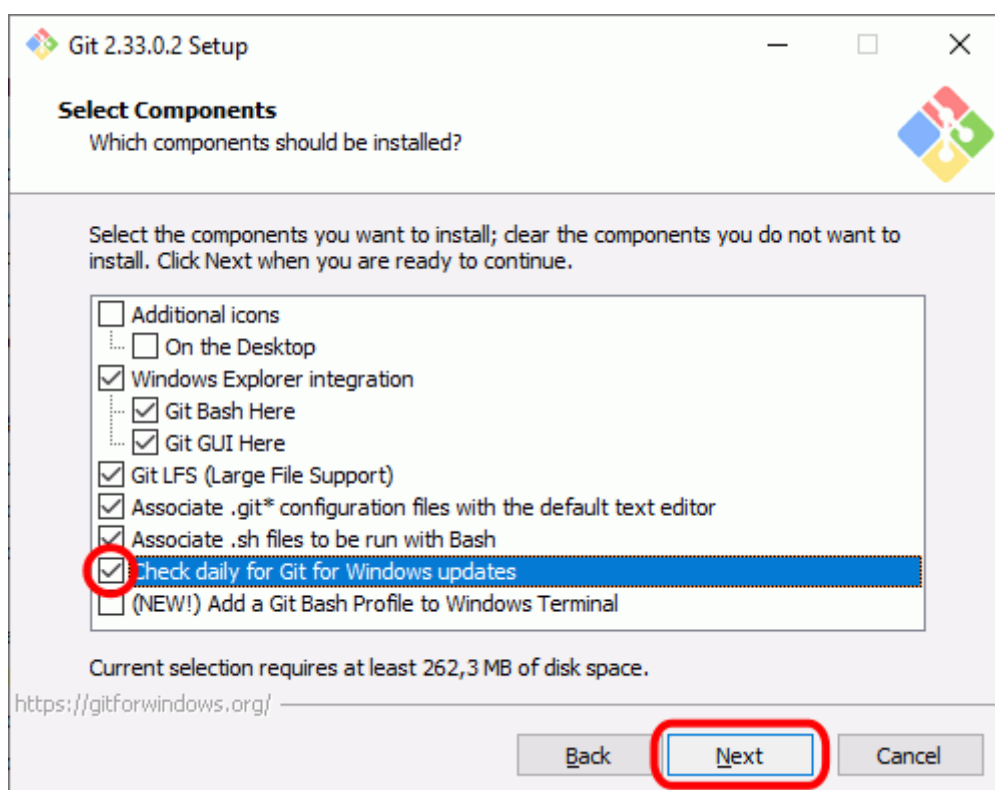
1. La primera pantalla muestra la licencia de Git ([GPL 2](#), como el kernel Linux):



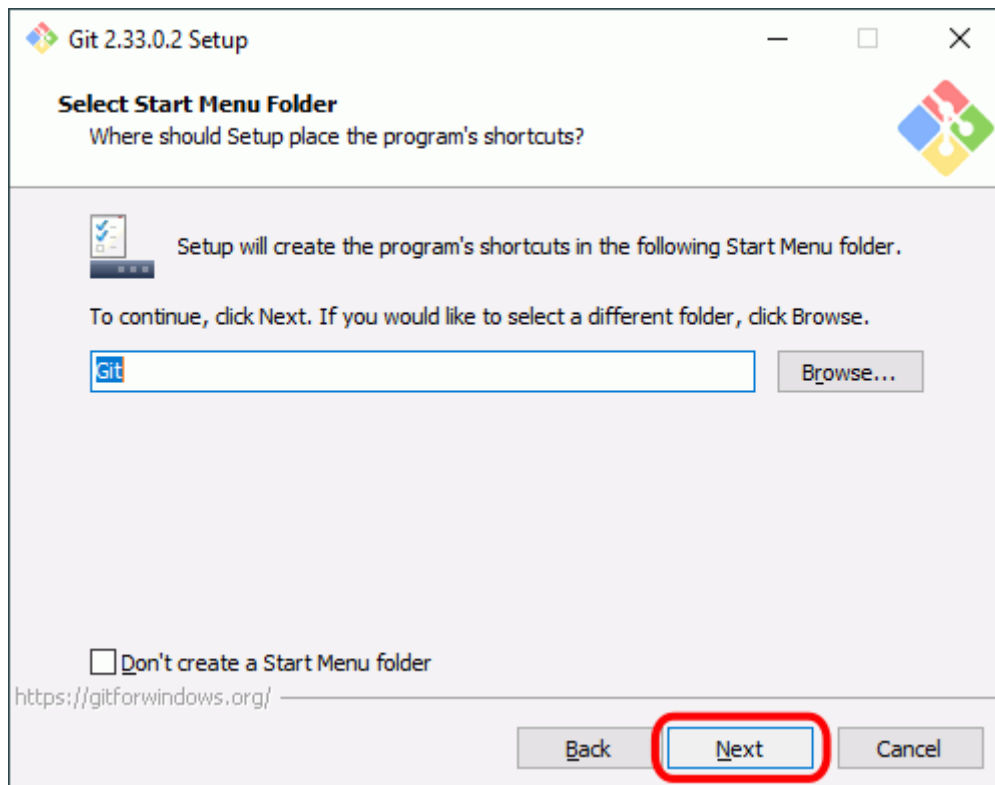
2. La siguiente pantalla indica el directorio de instalación:



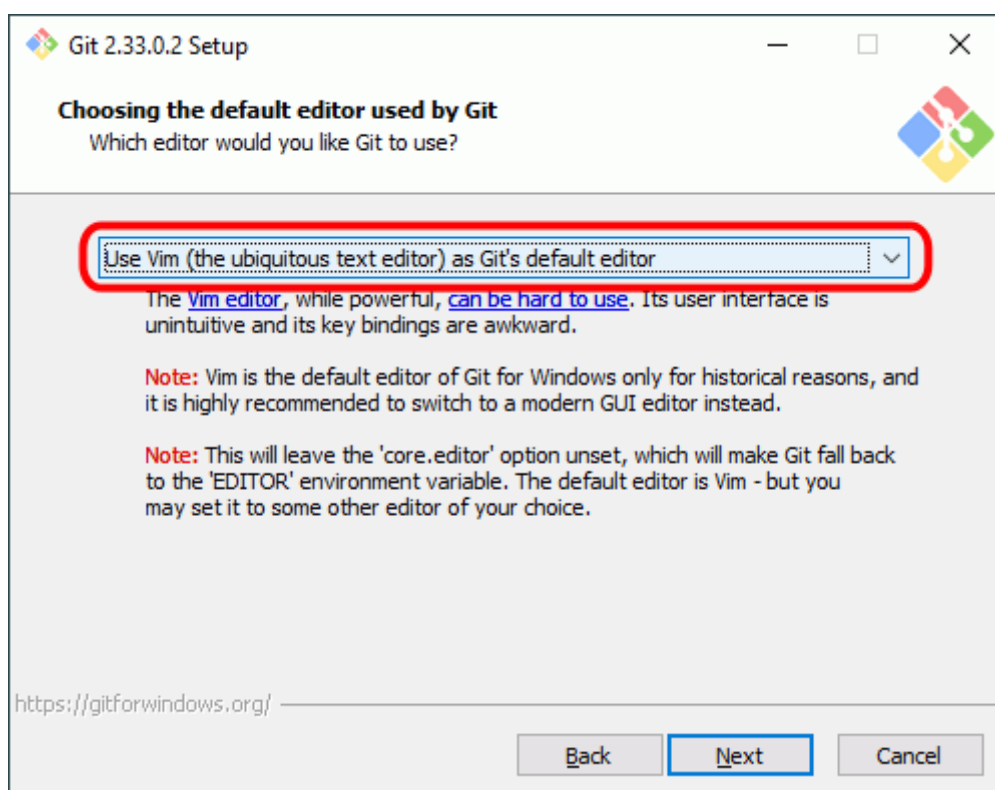
3. La siguiente pantalla indica los componentes a instalar. Marque la casilla "Check daily for Git for Windows updates".



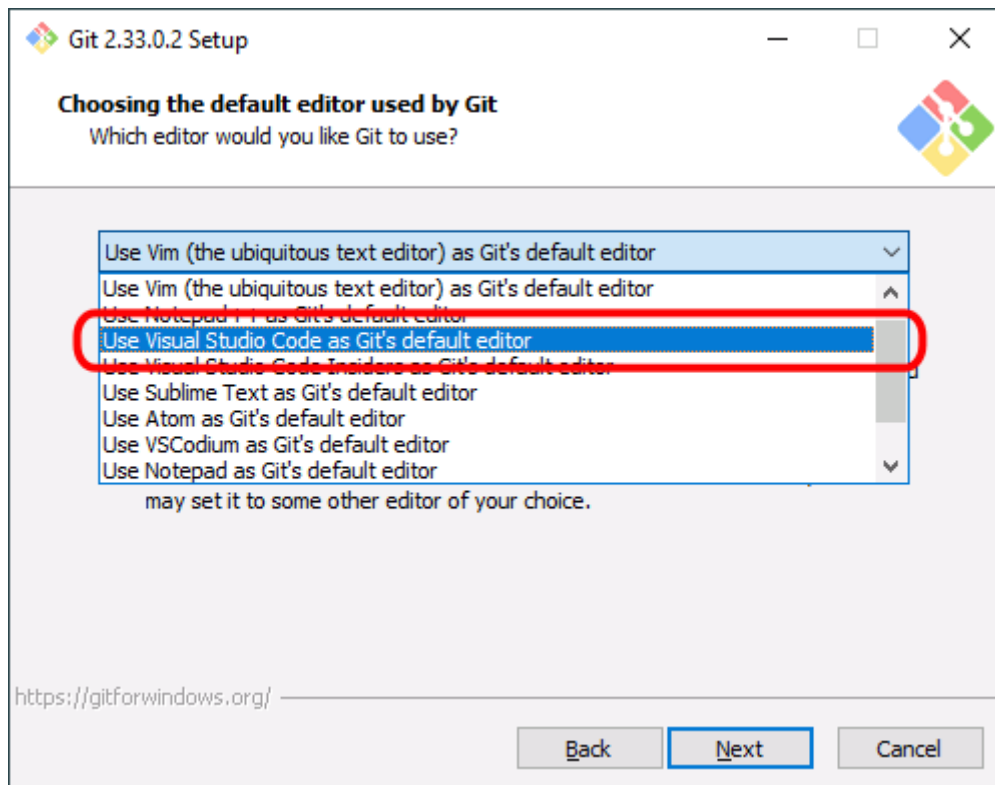
4. La siguiente pantalla indica el nombre de la carpeta en el menú de Inicio:



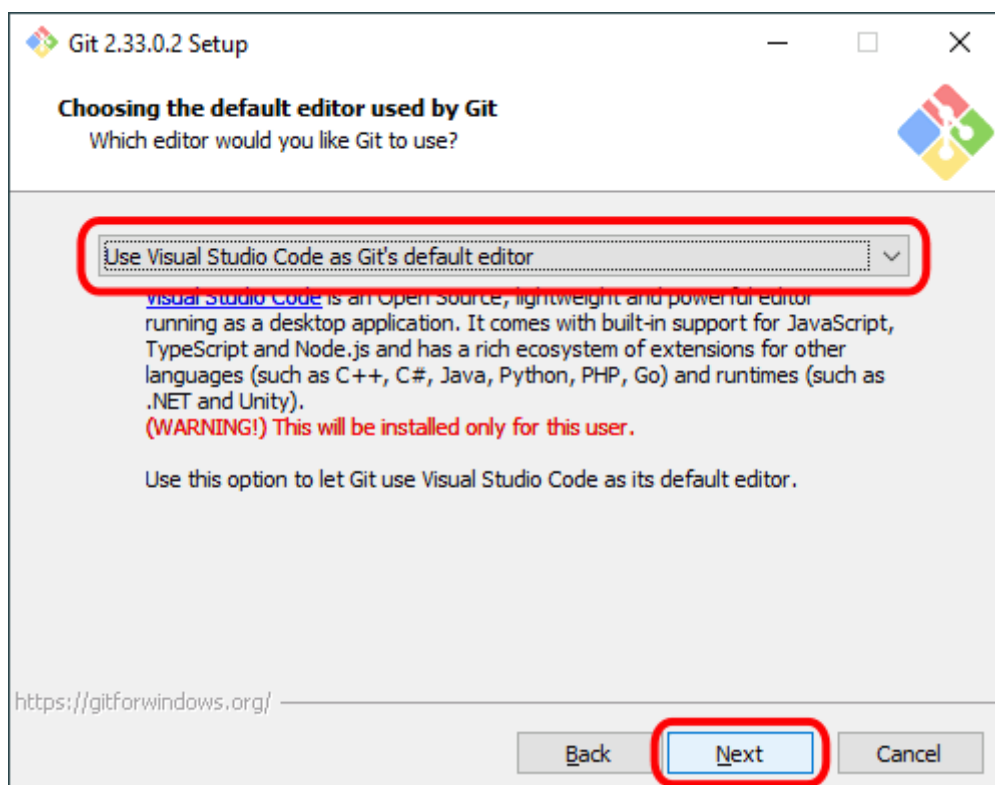
5. La siguiente pantalla permite elegir el editor predeterminado. De forma predeterminada Git ofrece Vim, pero el propio instalador recomienda elegir un navegador gráfico.



6. Despliegue la caja de lista para elegir otro editor. En estos apuntes se recomienda elegir Visual Studio Code ([Notepad++](#) es otro editor aconsejable).

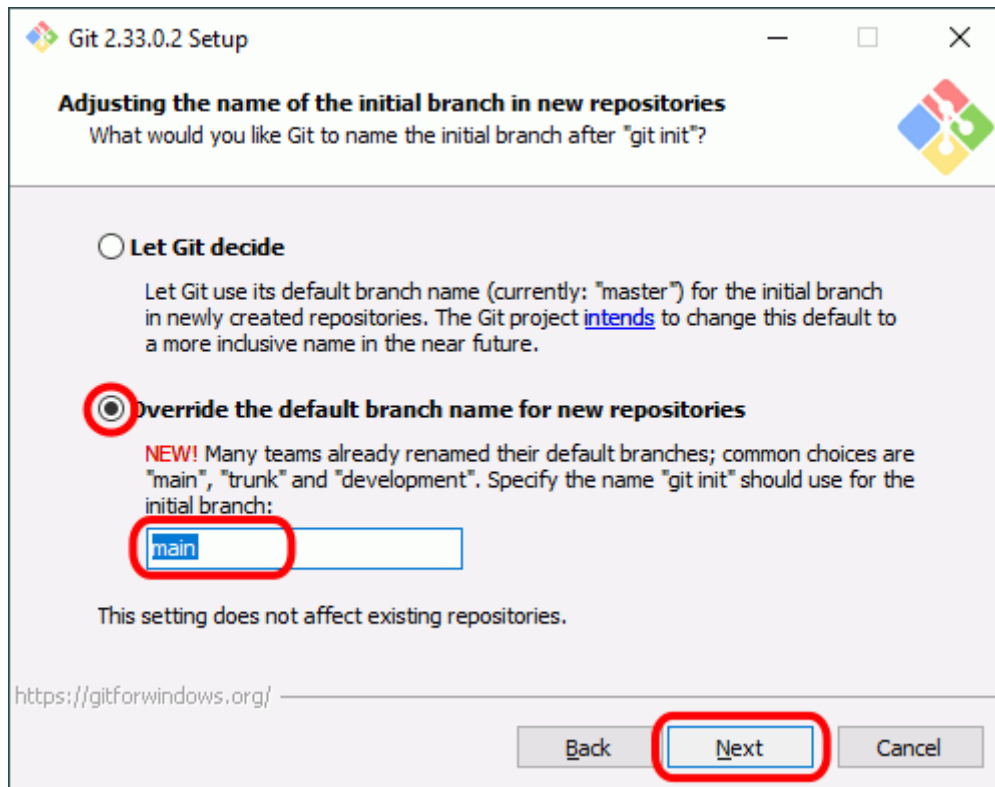


7. Una vez elegido el editor, haga clic en "Next":

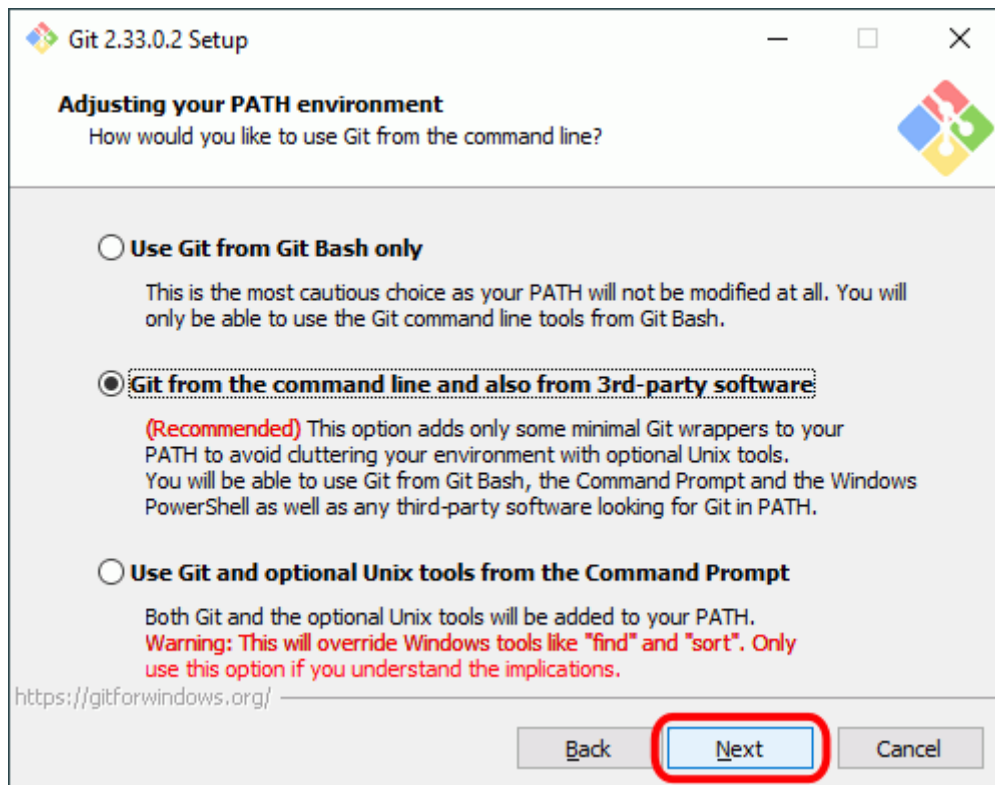


8. La siguiente pantalla permite elegir el nombre de la rama principal de los nuevos repositorios. Tradicionalmente la rama principal recibía el nombre de *master*, pero desde hace años este término se ha visto envuelto en polémica ya que hay personas que lo consideran racista (ya que los términos *master/slave* se pueden traducir como amo/esclavo). La verdad es que en este caso esta opinión es discutible (por un lado, la relación amo/esclavo

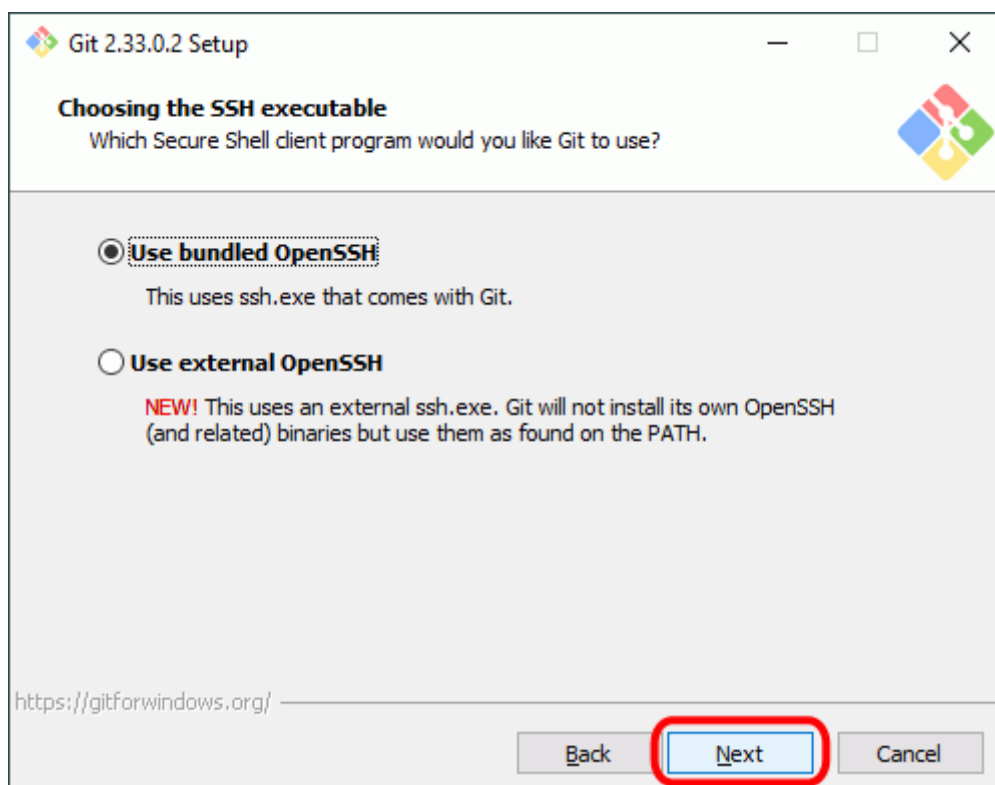
es independiente de la raza, y por otro lado, la palabra *master* en inglés tiene también el significado de "original", que es el que tiene sentido en este caso (*master record* es "grabación original") pero, de todas formas, muchos proyectos están modificando la denominación de *master*. Git todavía no ha decidido el cambio, pero GitHub ha empezado a utilizar el término *main*, que es el que elegiremos aquí.



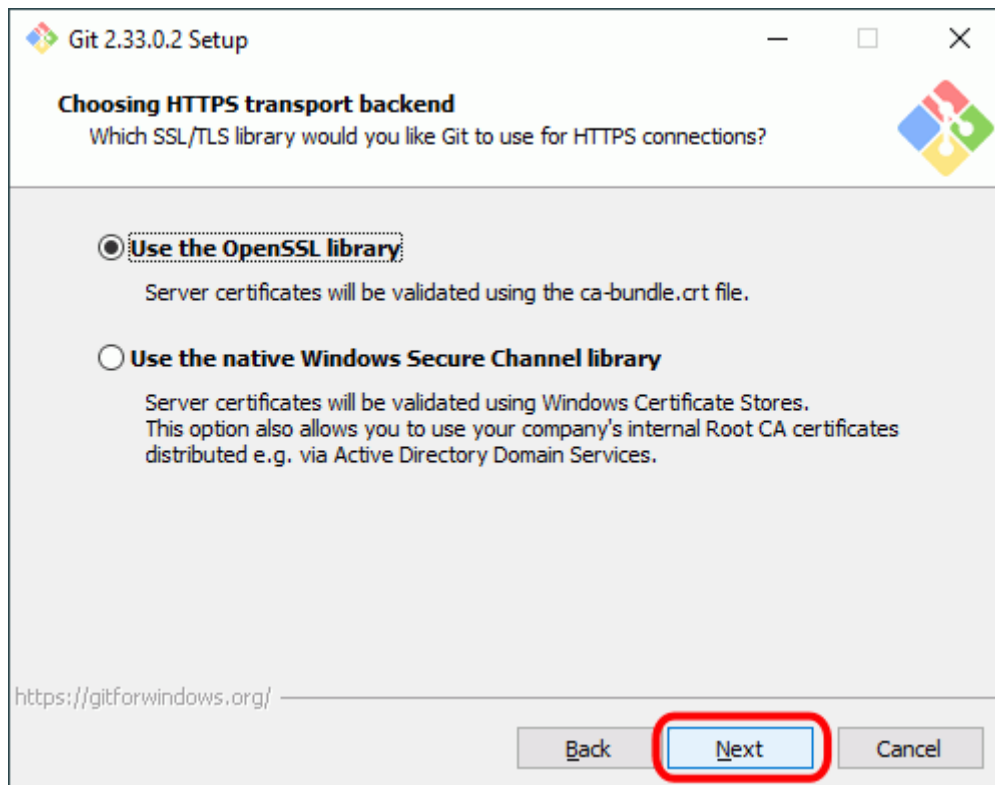
9. La siguiente pantalla permite elegir el tipo de ventanas de terminal desde las que se podrá usar Git:



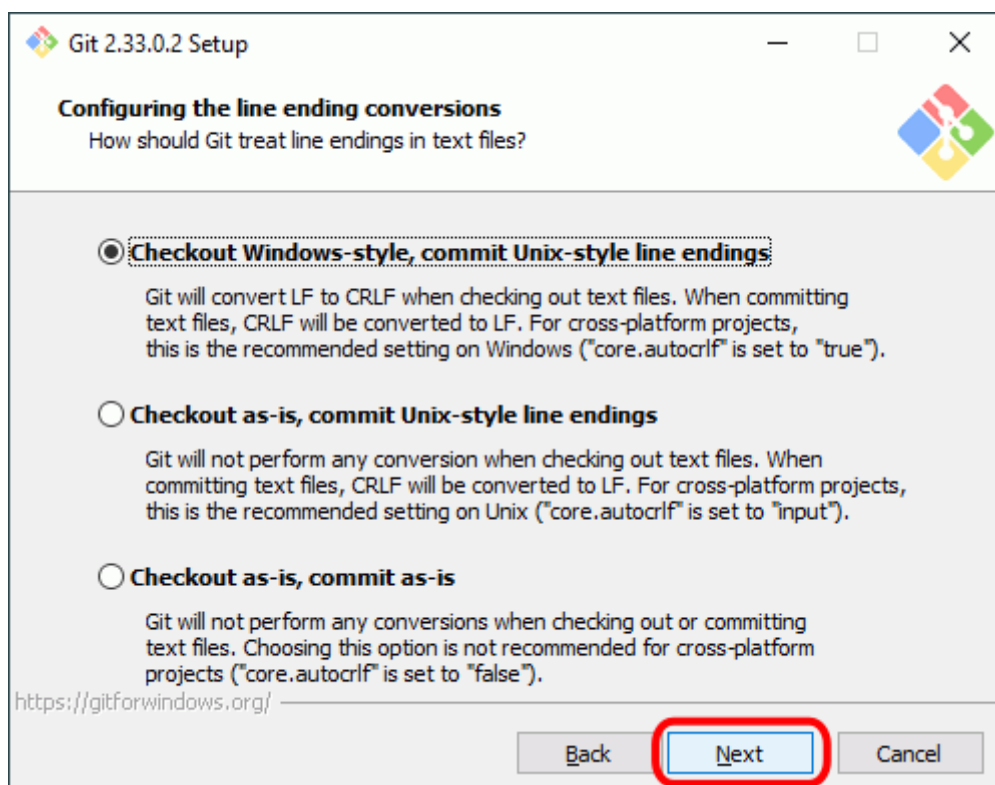
10. La siguiente pantalla permite elegir el ejecutable SSH:



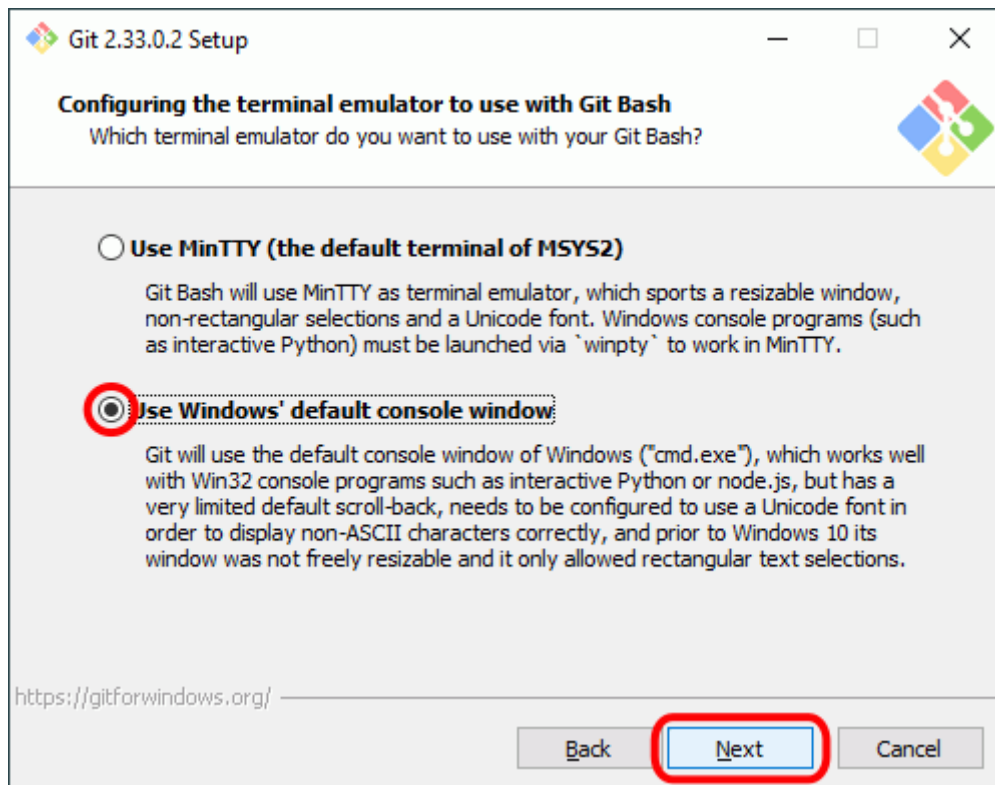
11. La siguiente pantalla permite elegir la biblioteca SSL/TLS:



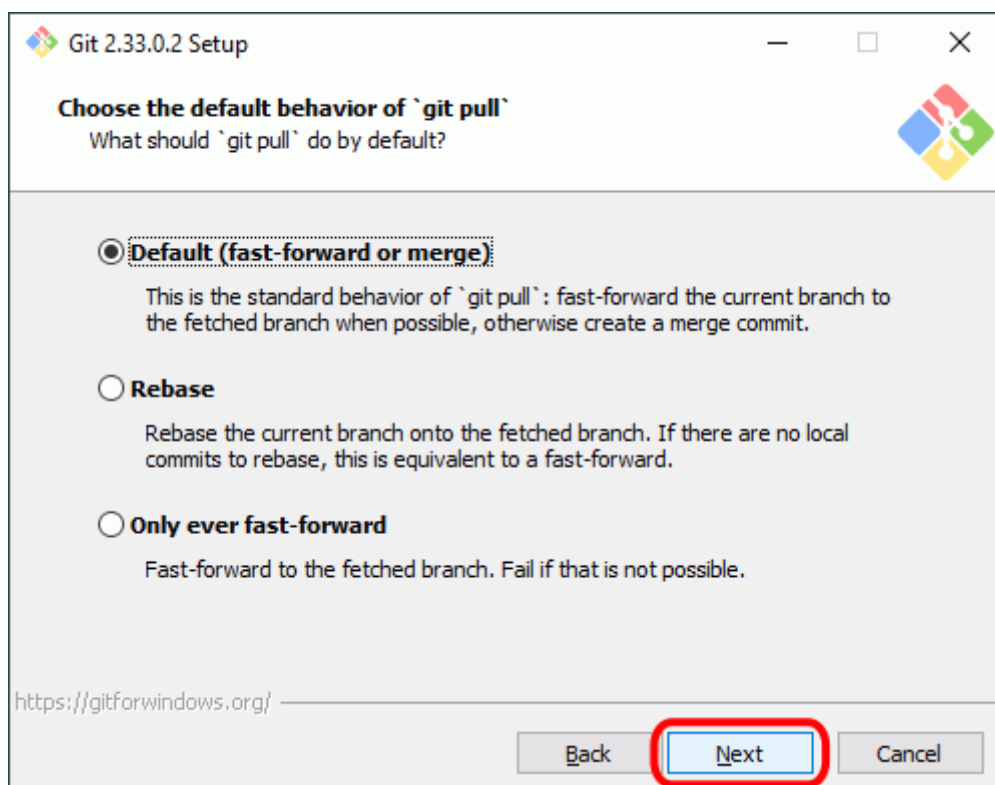
12. La siguiente pantalla permite elegir el carácter de final de línea:



13. La siguiente pantalla permite elegir el tipo de terminal de Git:

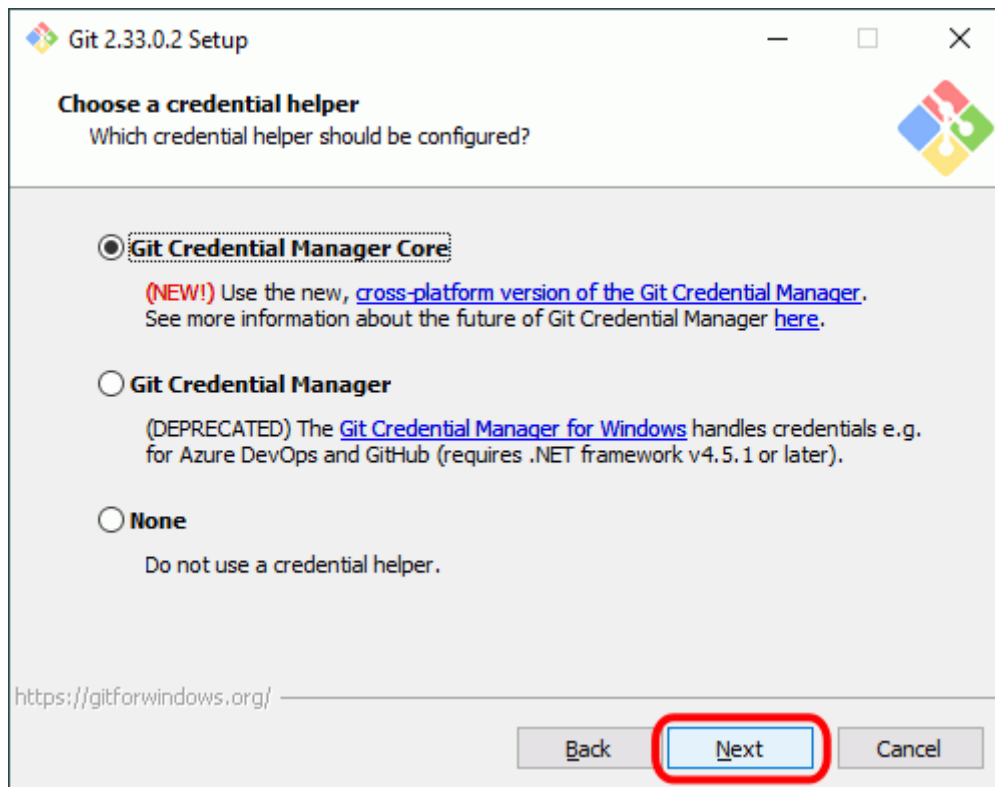


14. La siguiente pantalla permite elegir el comportamiento de la orden "git pull":

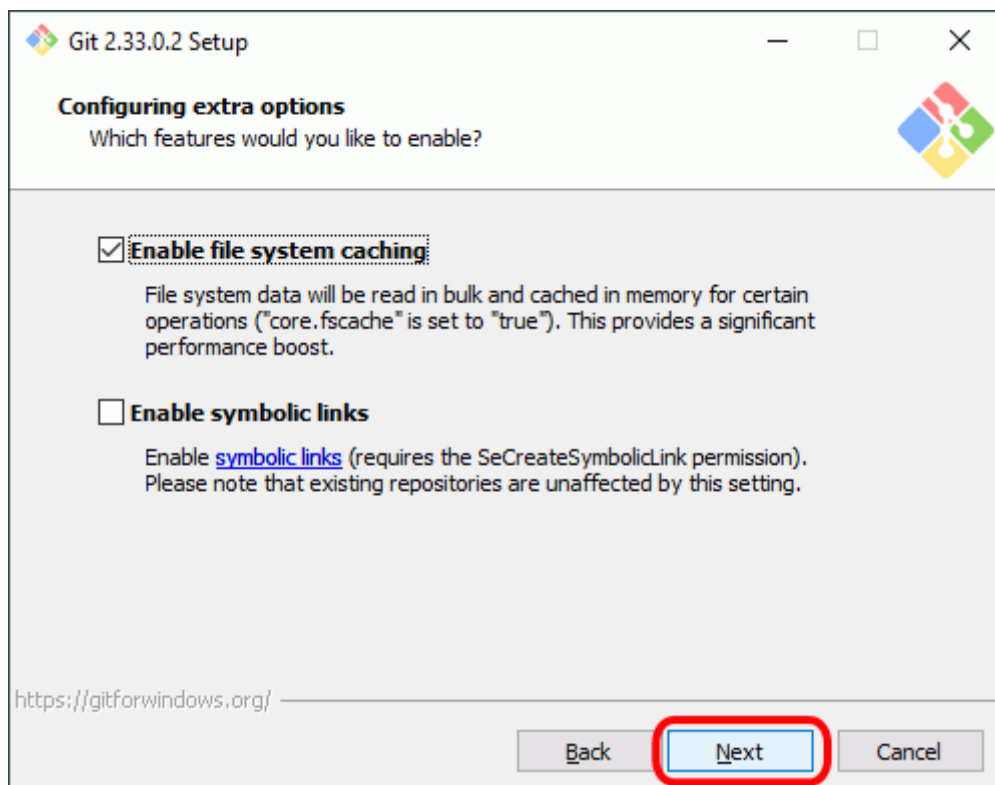


15. La siguiente pantalla permite elegir el gestor de credenciales:

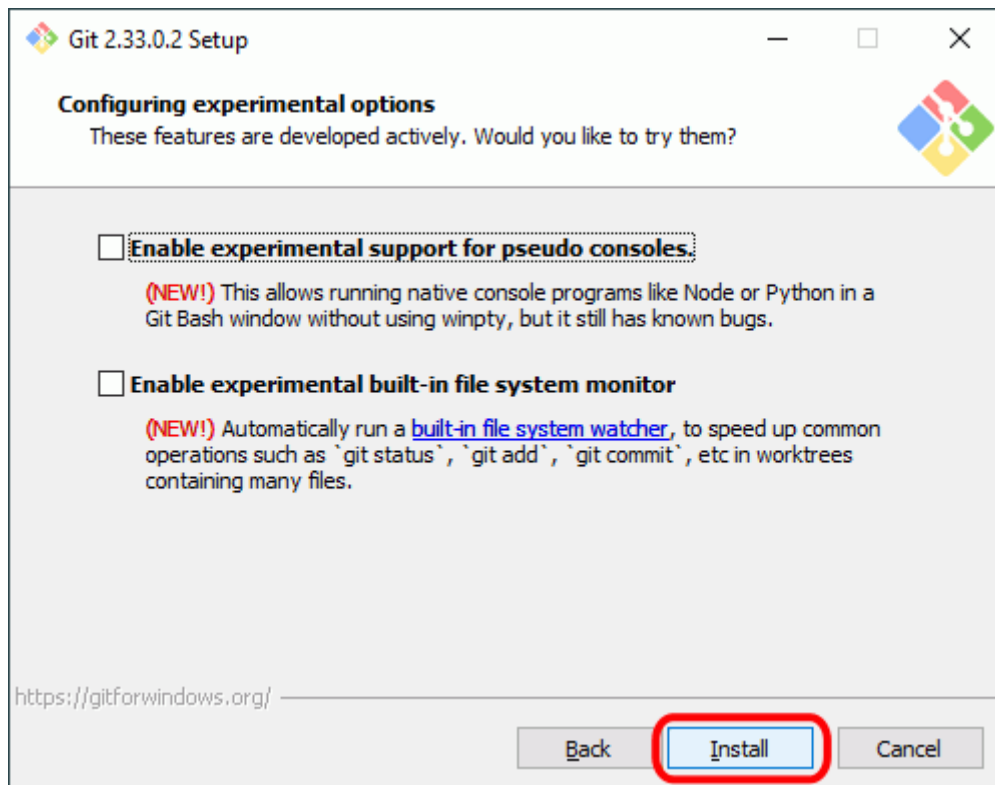




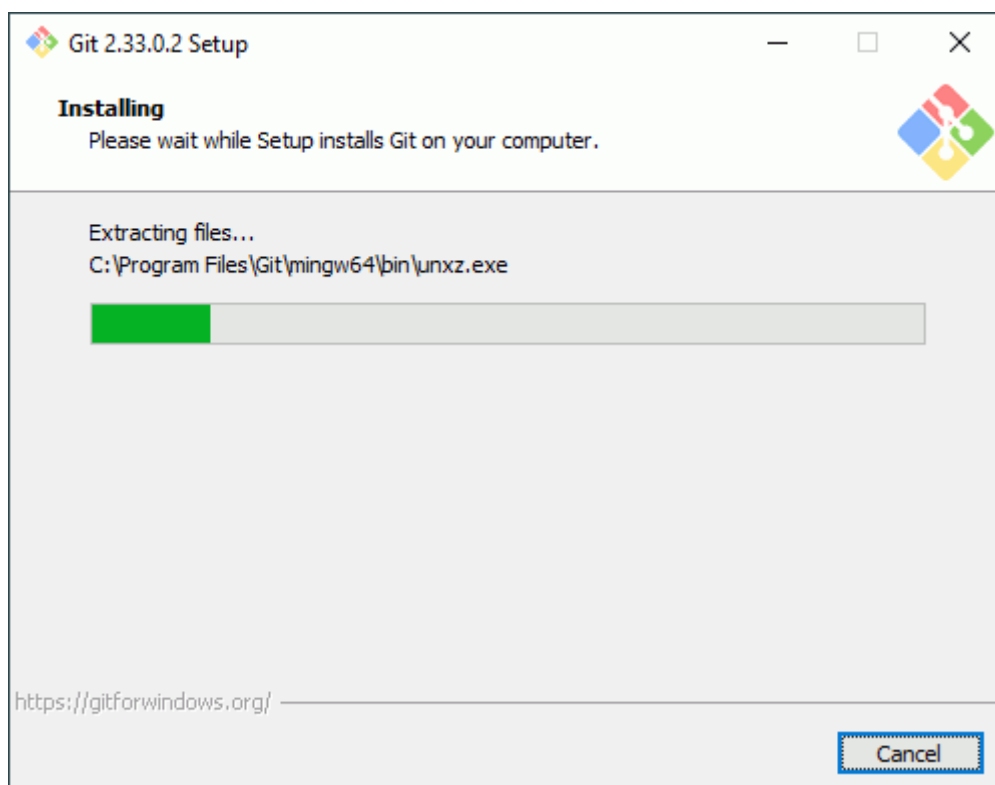
16. La siguiente pantalla permite elegir algunas características extra:



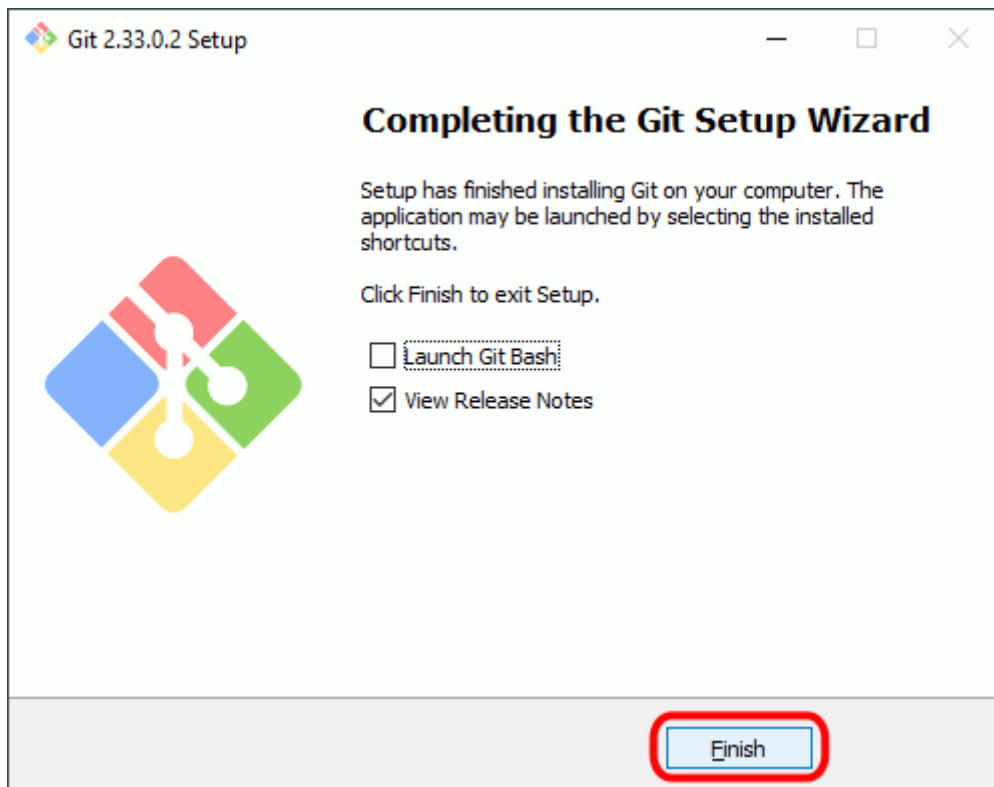
17. La siguiente pantalla permite elegir algunas características experimentales (en algunas versiones no aparece esta pantalla):



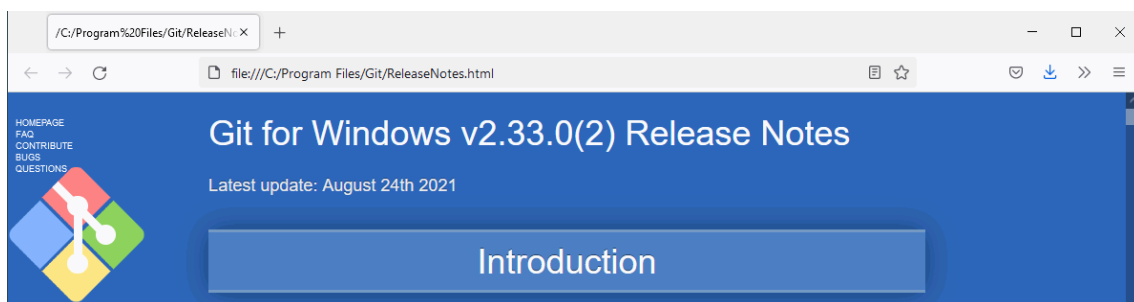
18. A continuación se realizará la instalación, que durará unos segundos:



19. Cuando termine la instalación se mostrará la pantalla final. Haga clic en Finish para cerrar el programa de instalación:



20. Al cerrarse el programa de instalación, si se ha dejado marcada la casilla "View Release Notes" en la pantalla final, se mostrarán en el navegador las notas de la versión:



## GITHUB. CREAR CUENTA

**GitHub** ofrece básicamente alojamiento web de repositorios git públicos o privados, pero también toda una serie de servicios asociados a ellos (alojamiento, issues, documentación, etc.). Los repositorios públicos son gratuitos y los privados son de pago (salvo en los casos de cuentas especiales como cuentas de estudiantes).

### Crear una cuenta en GitHub

Crear una cuenta de GitHub es muy sencillo. Tan sólo se necesita disponer previamente de una cuenta de correo electrónico (gmail, etc.):

1. Abra la página <https://github.com>.
2. Rellene los campos:

# Built for developers

GitHub is a development platform inspired by the way you work. From **open source** to **business**, you can host and review code, manage projects, and build software alongside millions of other developers.

Username

Pick a username

Email

you@example.com

Password

Create a password

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

Sign up for GitHub

By clicking "Sign up for GitHub", you agree to our [terms of service](#) and [privacy policy](#). We'll occasionally send you account related emails.

3. En la pantalla siguiente, elija el tipo de cuenta (que se puede cambiar posteriormente):

## Welcome to GitHub

You've taken your first step into a larger world, @NumaNigerio.

✓ Completed Set up a personal account	🔧 Step 2: Choose your plan	⚙️ Step 3: Tailor your experience
--	-------------------------------	--------------------------------------

### Choose your personal plan

☒ Unlimited public repositories for free.

☐ Unlimited private repositories for \$7/month. [\(view in EUR\)](#)

Don't worry, you can cancel or upgrade at any time.

☐ Help me set up an organization next  
Organizations are separate from personal accounts and are best suited for businesses who need to manage permissions for many employees.  
[Learn more about organizations](#)

☐ Send me updates on GitHub news, offers, and events  
Unsubscribe anytime in your email preferences. [Learn more](#)

Continue

### Both plans include:

- ✓ Collaborative code review
- ✓ Issue tracking
- ✓ Open source community
- ✓ Unlimited public repositories
- ✓ Join any organization

4. En la pantalla siguiente, conteste si lo desea unas preguntas simples sobre sus intereses en GitHub:

## Welcome to GitHub

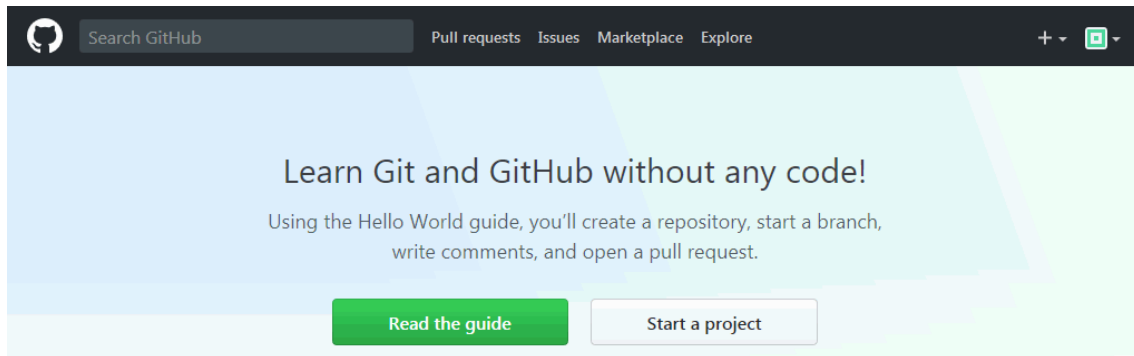
You'll find endless opportunities to learn, code, and create, @NumaNigerio.

✓ Completed Set up a personal account	🔧 Step 2: Choose your plan	⚙️ Step 3: Tailor your experience
--	-------------------------------	--------------------------------------

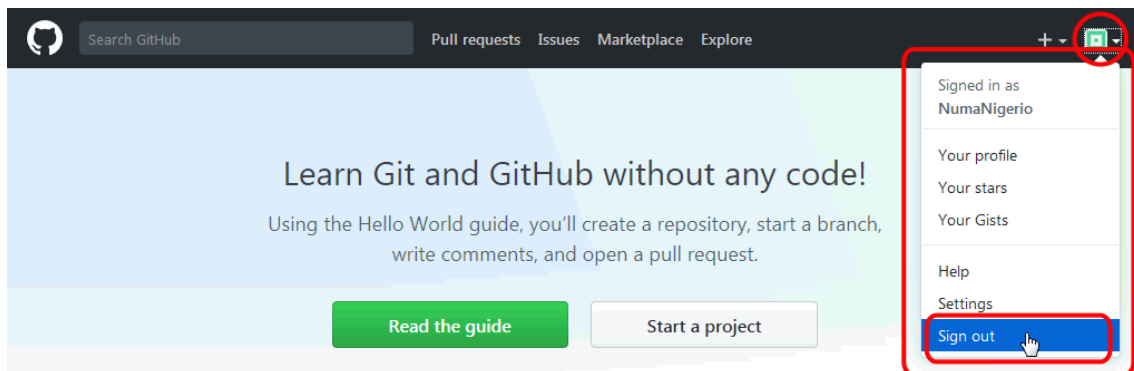
How would you describe your level of programming experience?

☐ Totally new to programming    ☐ Somewhat experienced    ☐ Very experienced

5. La pantalla final ofrece la posibilidad de seguir un tutorial o crear directamente un repositorio:



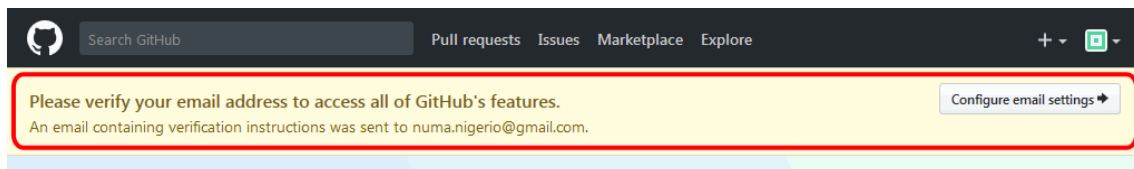
6. Para salir de GitHub, haga clic en el icono superior derecho y elija la opción "Sign out":



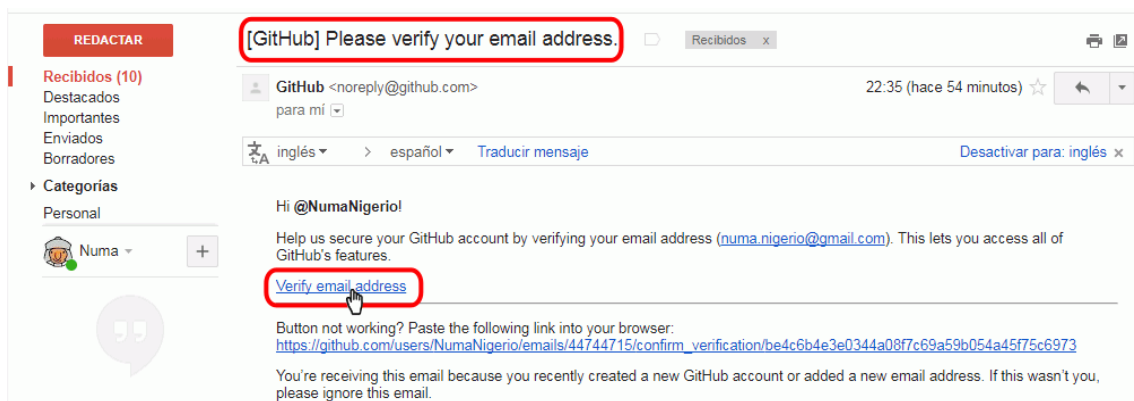
### Verificar cuenta de correo

Al crear la cuenta, GitHub envía un correo electrónico a su cuenta de correo para confirmar que la cuenta de correo es correcta.

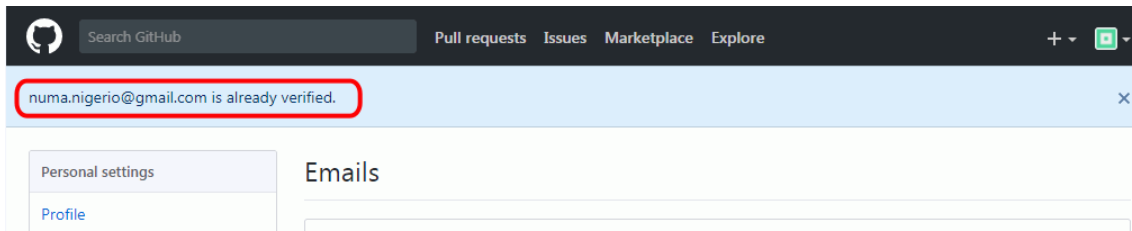
1. Si no verifica la cuenta de correo, GitHub le recordará que debe hacerlo:



2. Abra su cuenta de correo y en el correo remitido por GitHub, haga clic en el enlace "Verify email address":

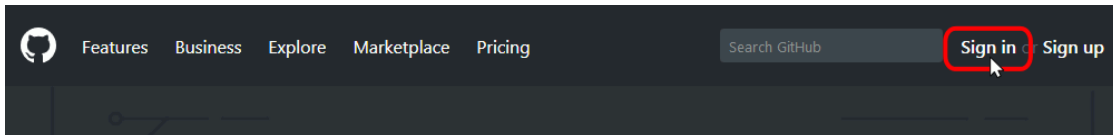


3. Se abrirá una nueva pestaña en la sección de cuentas de correo de la página de configuración de la cuenta de GitHub y se mostrará un mensaje indicando que se ha verificado la cuenta:



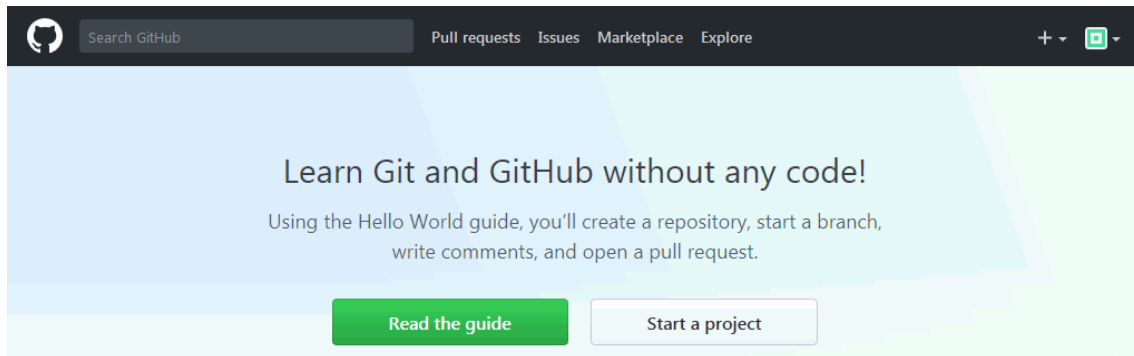
Entrar en una cuenta de GitHub ya existente

1. Abra la página <https://github.com>.
2. Haga clic en "Sign in", situado arriba a la derecha:



3. Escriba la dirección de correo y la contraseña de su cuenta y haga clic en "Sign in":

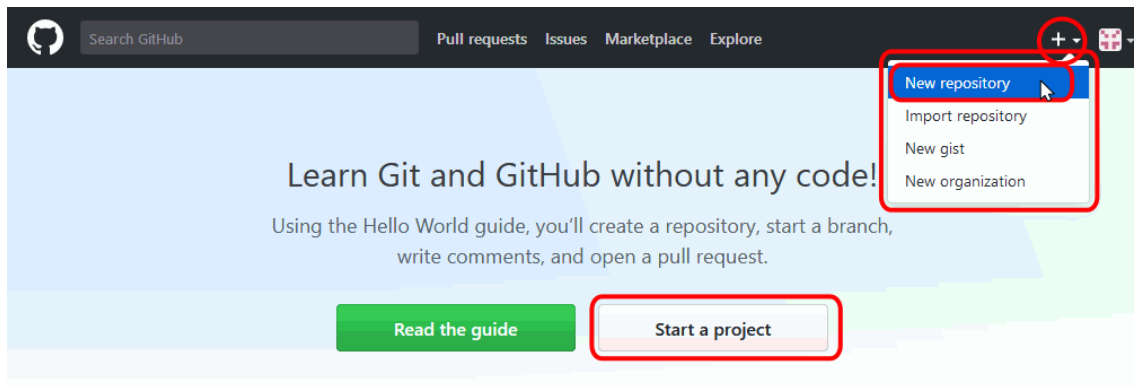
4. Se mostrará su página de GitHub (en este caso, la página inicial, sin repositorios creados):



## CREAR UN REPOSITORIO EN GITHUB

Para crear un nuevo repositorio en GitHub:

1. Haga clic en el icono en forma de cruz situado arriba a la derecha y elija la opción "New repository". Si no ha creado todavía ningún repositorio, en la página web se puede mostrar un botón "Start a project", que también permite crear un nuevo repositorio.



2. Indique el nombre del repositorio, su descripción, si es público o privado. Se recomienda incluir en el repositorio un fichero README, pero no es necesario crear el fichero .gitignore (salvo si se trata de un proyecto de software de uno de los tipos disponibles). Si se trata de un proyecto de software libre, elija la licencia que desee emplear. Haga clic en Create repository:

## Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: NumaNigerio

Repository name:

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

Description (optional):

☒ Public  
Anyone can see this repository. You choose who can commit.

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

☒ Initialize this repository with a README  
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore:

Add a license:

3. Inmediatamente se mostrará el repositorio recién creado. En este caso únicamente contiene el fichero `Readme.md`:

This repository Search Pull requests Issues Marketplace Explore

NumaNigerio / **test-01** Watch 0 Star 0 Fork 0

<> Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

Repositorio de prueba Edit

Add topics

1 commit 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download

NumaNigerio Initial commit Latest commit 84a9c31 Jan 25, 2018

README.md Initial commit just now

README.md

test-01

Repositorio de prueba

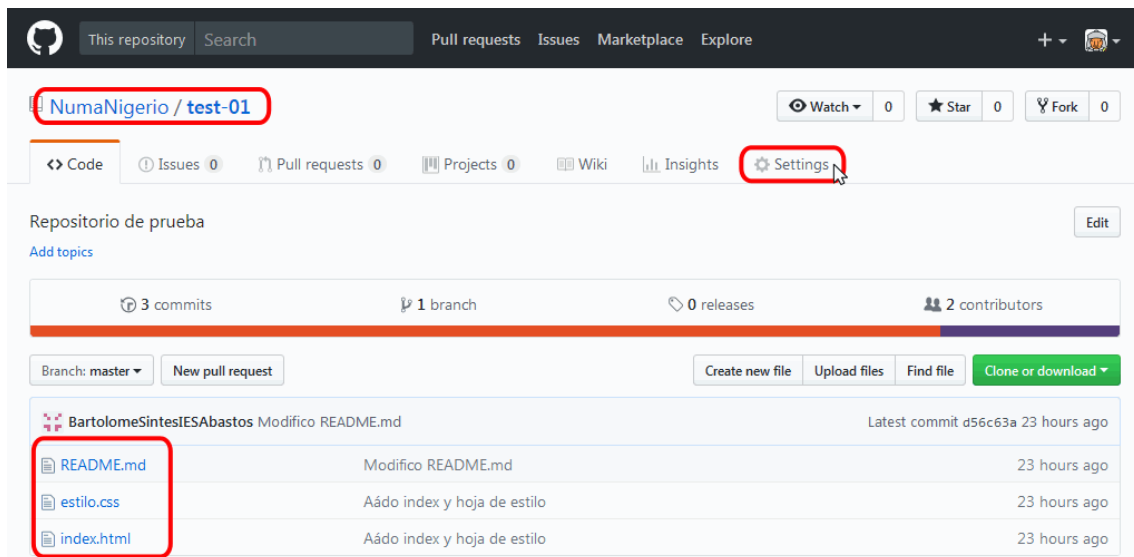
### Publicar repositorios web

[GitHub Pages](#) es un servicio de GitHub que permite publicar en la web el contenido de un repositorio.

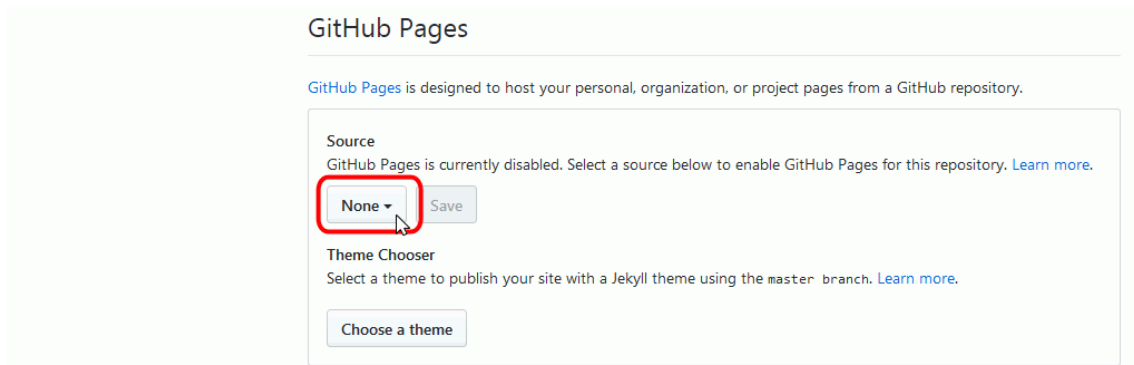
Activar este servicio es muy sencillo:

- Si el repositorio contiene únicamente las páginas web que queremos publicar, haga clic en el enlace **Settings** (configuración):

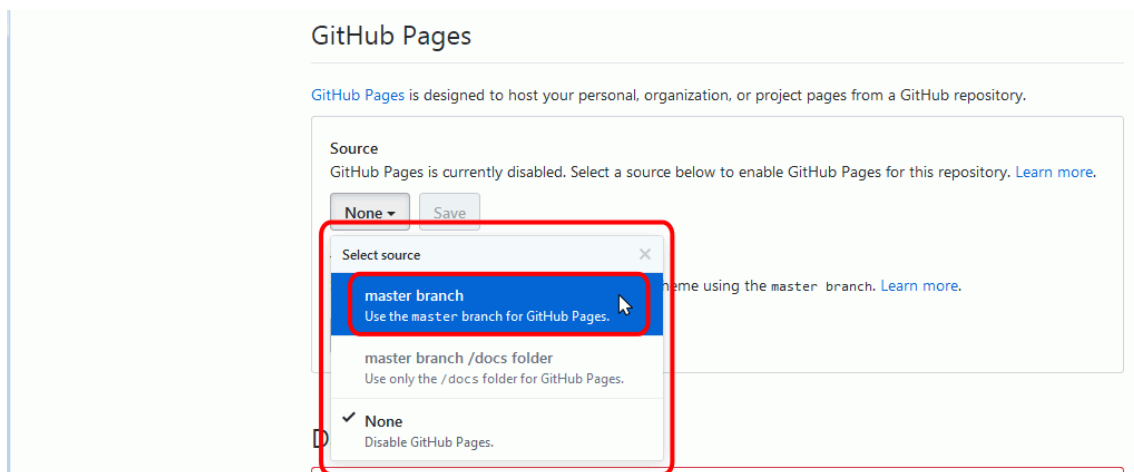




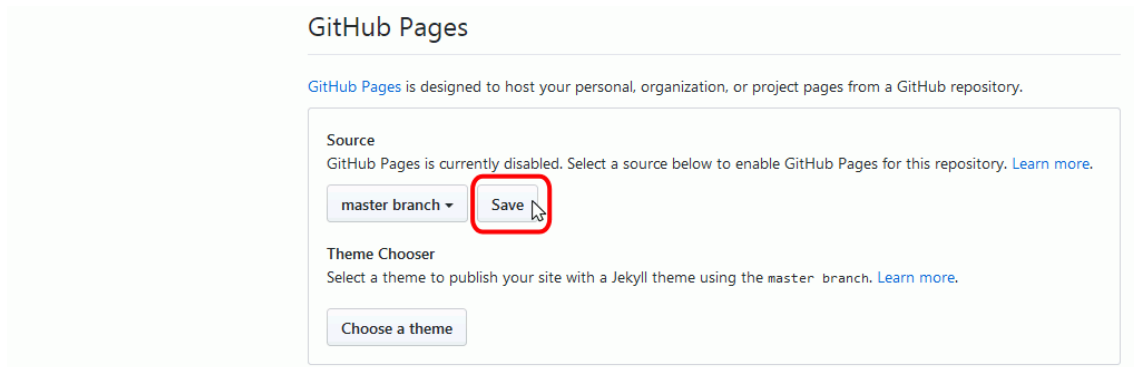
- Casi al final de la página de configuración se encuentra el apartado correspondiente a GitHub Pages. Haga clic en el botón que ahora dice None ...



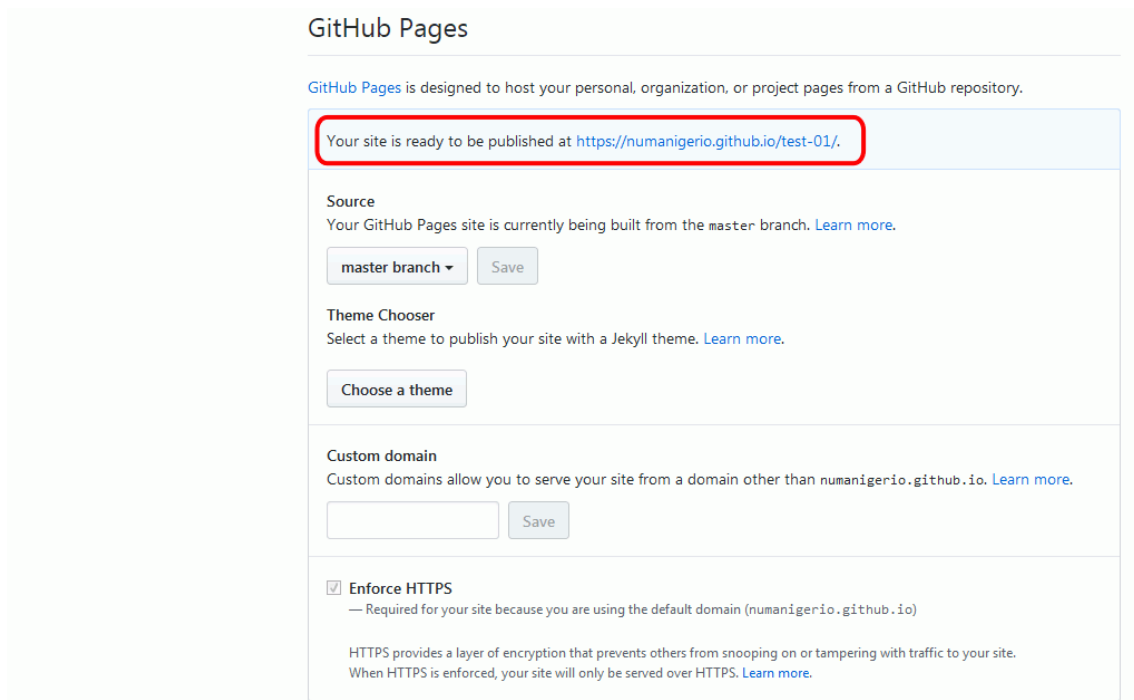
- ... cambie a "master branch" ...:



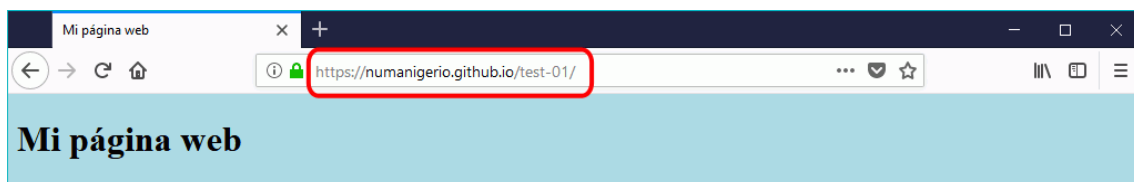
- ... y haga clic en "Save":



- La página se actualizará y le mostrará la URL pública:



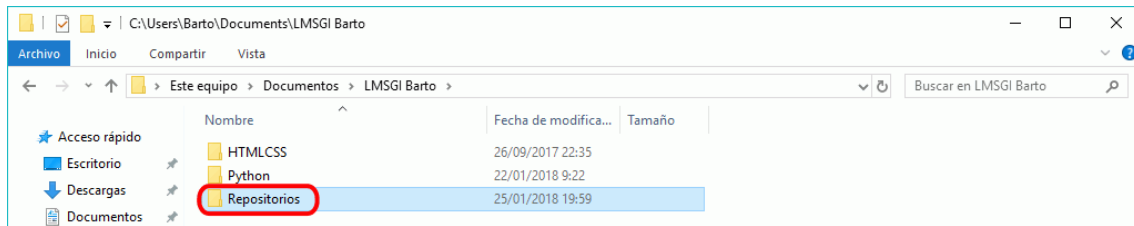
- En unos segundos la página estará disponible::



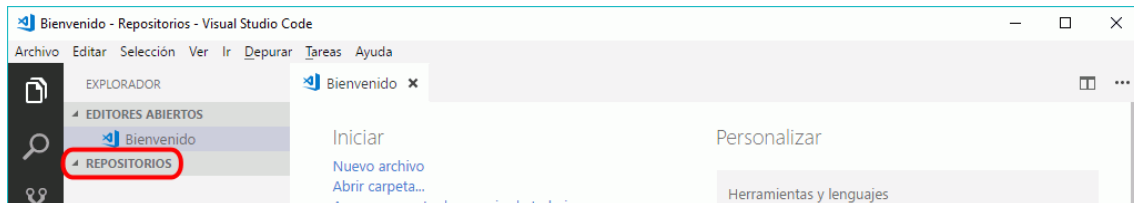
## Clonar un repositorio de GitHub localmente

Para clonar localmente con Visual Studio Code un repositorio de GitHub:

1. Cree un directorio donde se guardarán los repositorios:



2. Abra el directorio en Visual Studio Code (menú Archivo > Abrir carpeta ...):

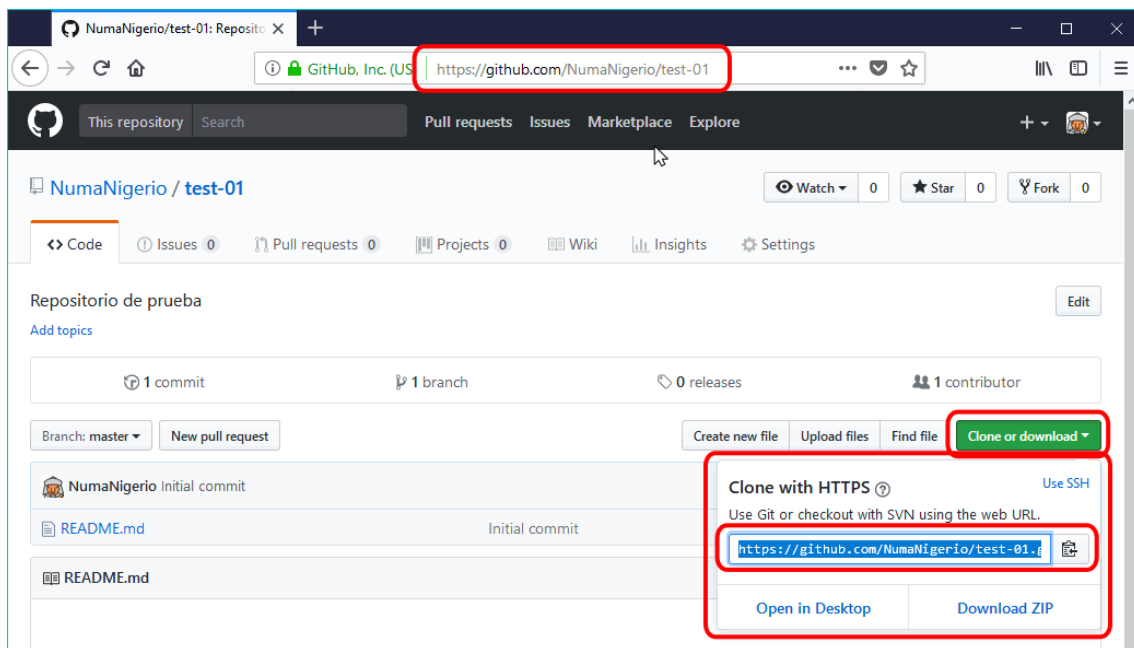


3. Para clonar el repositorio necesitará dos rutas (el origen y el destino):

- La URL de repositorio, que se puede copiar desde el propio repositorio haciendo clic en el botón "Clone or download".

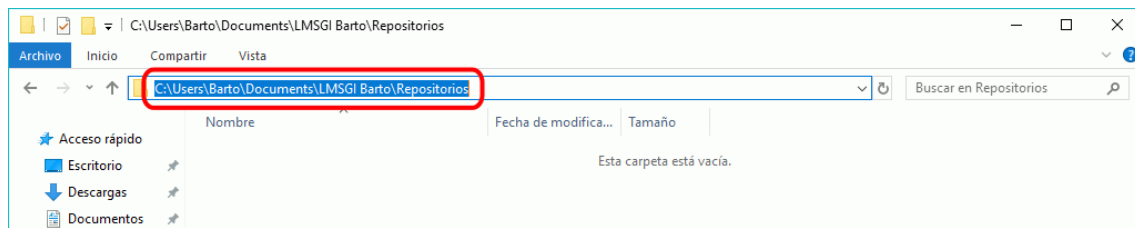
Esta URL es la misma URL que aparece en el navegador pero terminada en .git.

En la captura de ejemplo, la URL de origen es <https://github.com/NumaNigerio/test-01.git>



- La ruta del directorio de destino, es decir, la ruta de la carpeta en la que se va a crear la carpeta que contendrá la copia del repositorio. Esta ruta se puede copiar del Explorador de Windows.

En la captura de ejemplo, el directorio de destino es `C:\Users\Barto\Documents\LMSGI\Barto\Repositorios`

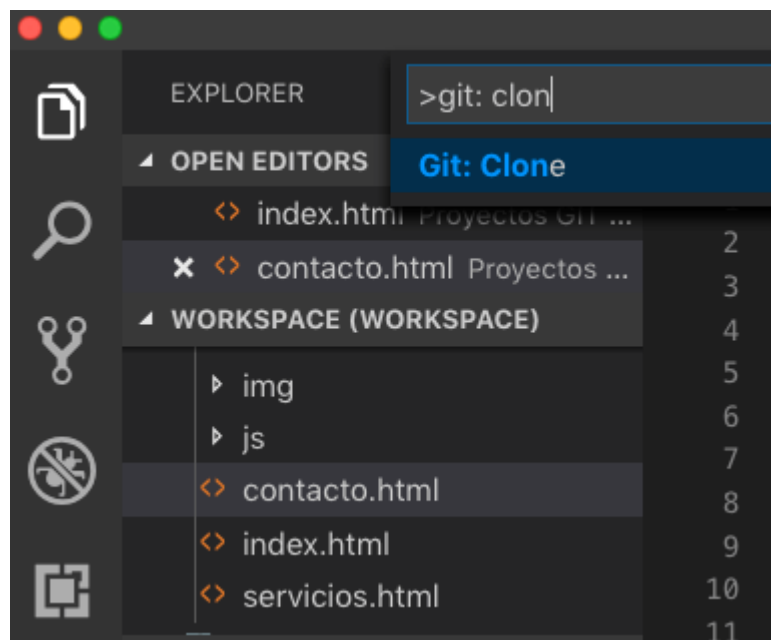


Dentro de Git debemos definir nuestro usuario e email de la siguiente manera:

```
$ git config --global user.name "eniun"
```

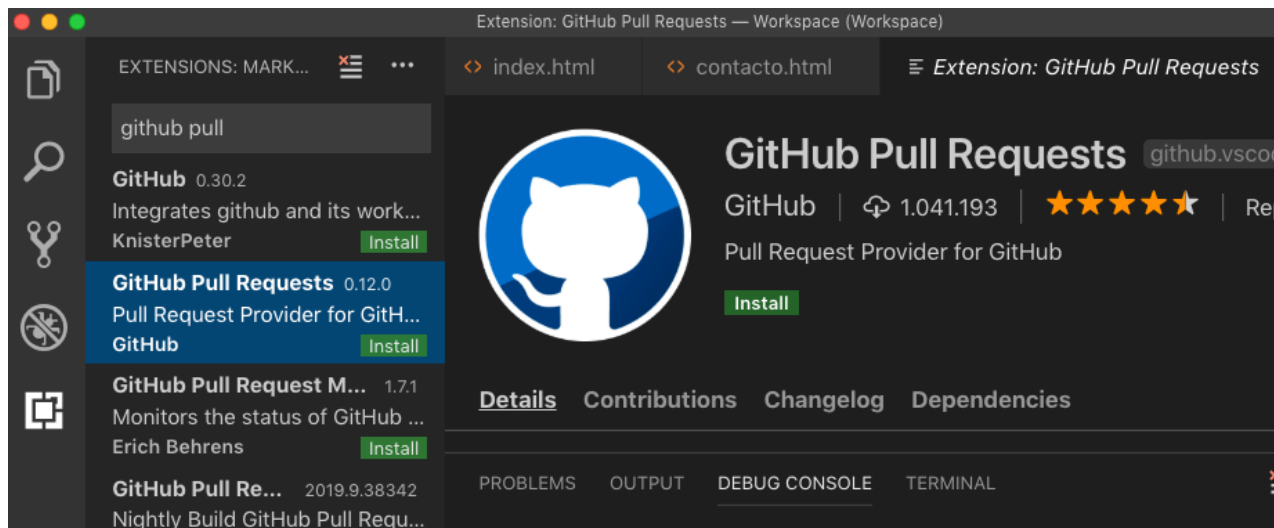
```
$ git config --global user.email info@eniun.com
```

Ahora vamos a clonar el proyecto creado en Github desde Visual Studio Code. Para ello, nos vamos al menú “view” seleccionamos “command palette”. En ese punto buscamos “git: Clone”. La herramienta nos pedirá la ruta y ahí es donde tenemos que pegar la URL del repositorio que hemos creado en el punto anterior en Github.

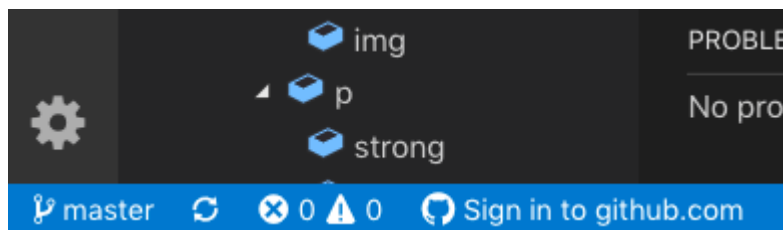


De esta forma ya se ha creado nuestro repositorio y la herramienta nos va marcando los ficheros que vamos modificando.

Antes de hacer un commit debemos instalar la extensión “GitHub pull request” desde el menú view/terminal.



Además debemos autorizar nuestra cuenta Github en Visual Studio Code. Para ello, vete a la barra inferior de Visual Studio y presiona sobre el icono de Git e introduce tu cuenta de Github.



Desde la pestaña de Git podemos hacer el “commit” de los ficheros. En primer lugar, incluiremos los ficheros a los que queremos hacer stage pulsando en “+” o “todos”. Además debemos escribir el mensaje del Commit e Intro para terminar.

Para hacer push de los cambios o pull para copiar los datos de servidor podemos hacerlo de dos maneras:

- Podemos pulsar en la parte derecha de la barra de git (icono tres puntos) y elegir la opción *push* o *pull*.
- Podemos pulsar en la parte inferior sobre sincronizar (en este caso haríamos *push* y *pull*).

Ver los cambios en Github

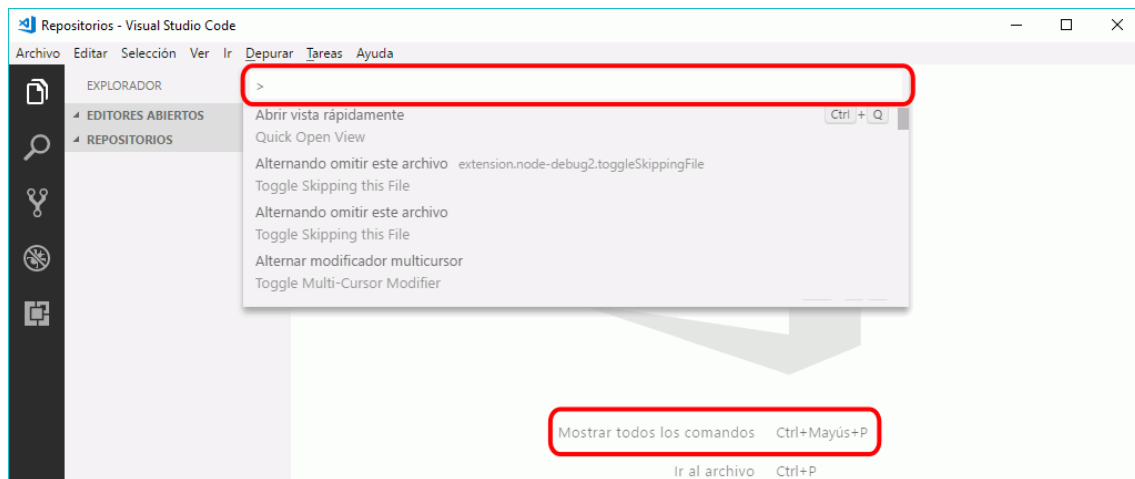
En Github dale a “Settings / Source y selecciona “**Master Branch**”.

Visualiza tu web. La URL de visualización tendrá el siguiente

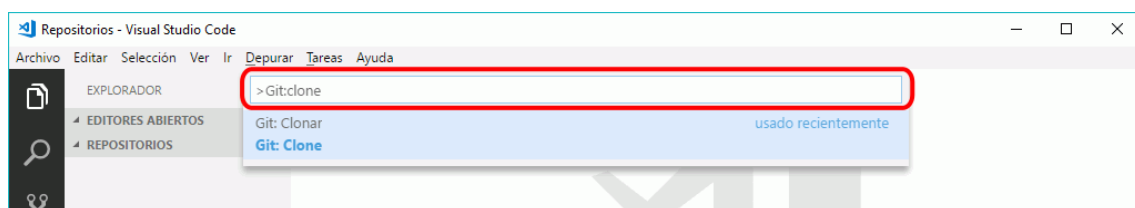
aspecto: <https://nombredetucuenta/github.io/nombredeturepositorio/> (La página mostrada es la página index.html). En mi caso es la siguiente: <https://eniun.github.io/diw/>

Si quieres seguir aprendiendo funcionalidades de Github puedes comenzar haciendo el siguiente tutorial: <https://guides.github.com/activities/hello-world/>.

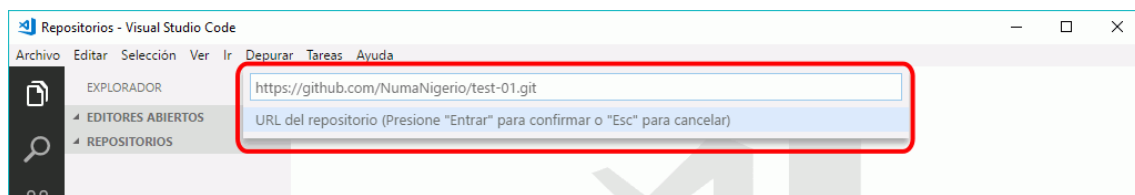
4. En Visual Studio abra la ventana de comandos con **Ctrl+May+p**



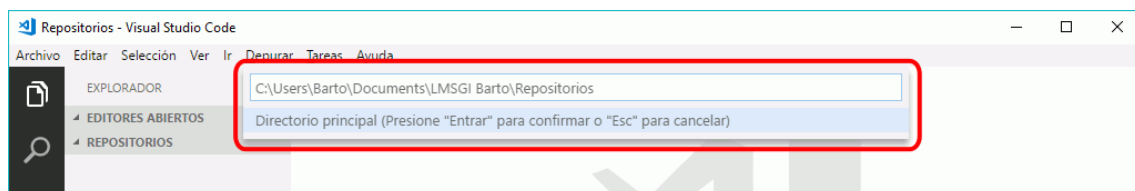
5. Escriba el comando **git:clone** y pulse Intro:



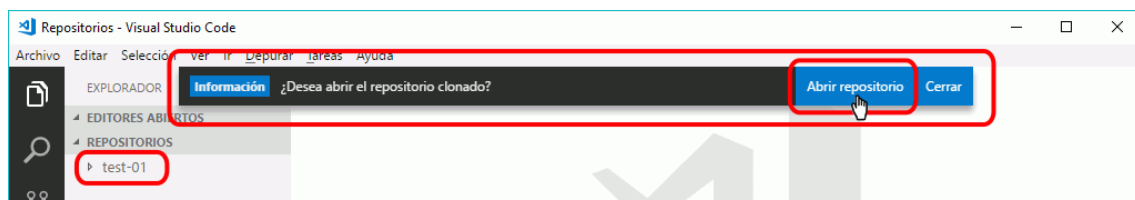
6. Escriba la URL del repositorio de origen y pulse Intro:



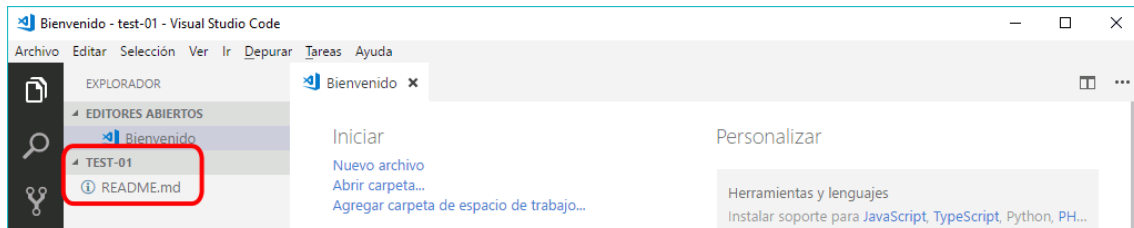
7. Escriba la ruta del directorio que contendrá la copia y pulse Intro:



8. En unos segundos se mostrará la carpeta del repositorio clonado en la barra del explorador y se ofrecerá la posibilidad de abrir la carpeta del repositorio clonado. Haga clic en "Abrir repositorio".



9. Visual Studio Code abrirá la carpeta del repositorio clonado, mostrando su contenido (en el ejemplo, el repositorio contiene únicamente el fichero README.md).



## Realizar commits en el repositorio remoto

En git, un commit es un conjunto de cambios que se realizan en los ficheros del repositorio.

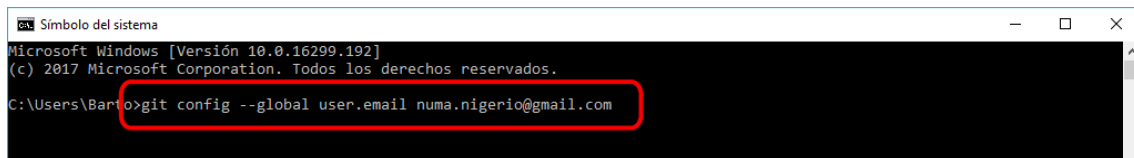
En Visual Studio Code podemos realizar commits, que se guardan en el repositorio local. Para sincronizar nuestros cambios en el repositorio, debemos identificarnos como usuario del repositorio.

Esa identificación se guarda como credencial de Windows, por lo que si estamos utilizando un ordenador al que tienen acceso otras personas, es muy importante que eliminemos la credencial antes de abandonar el ordenador o nos arriesgamos a que otro usuario nos suplante y acceda a los repositorios remotos.

El proceso en Visual Studio Code sería el siguiente:

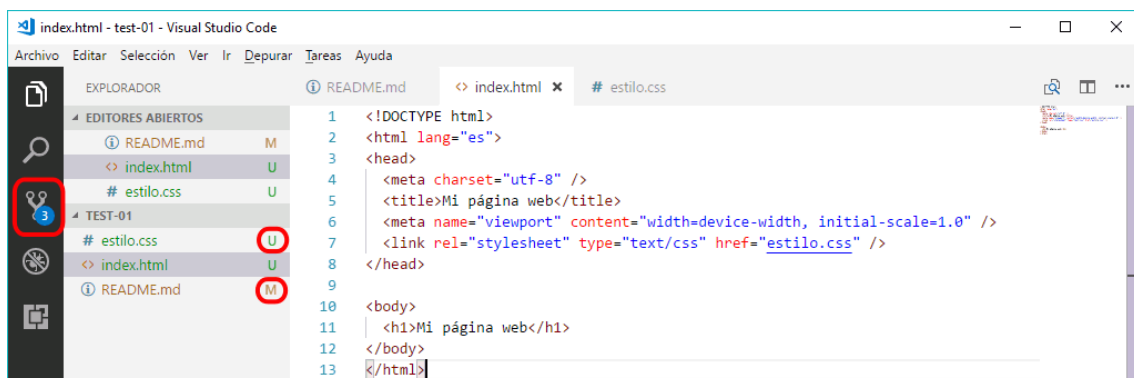
- Abra una ventana de Terminal e indique a git la dirección de correo del usuario (la dirección de correo vinculada a su cuenta en GitHub):

`git config --global user.email SU-CORREO-ELECTRONICO`



- Modifique los archivos del repositorio o añada nuevos archivos

En la captura del ejemplo, en el icono Control de código fuente se nos indica el número de ficheros con cambios (3). En la barra del Explorador se indica qué ficheros son nuevos (con la letra U) o se han modificado (con la letra M).



- Abra la barra lateral de Control de código fuente haciendo clic en el icono correspondiente. Se mostrará la lista de ficheros con cambios:



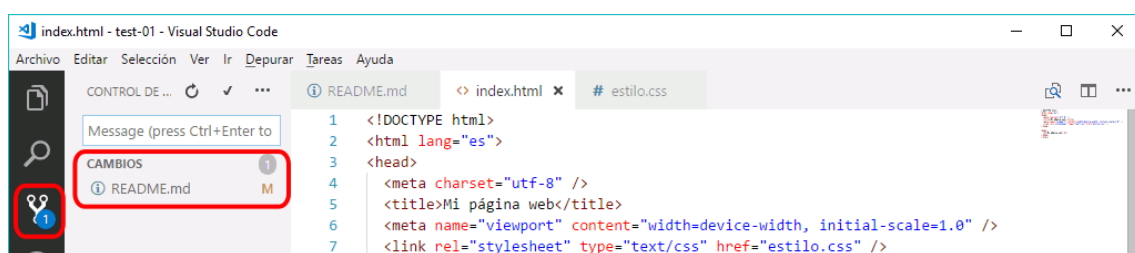
- Seleccione los ficheros cuyos cambios formarán parte del commit (puede elegir todos los ficheros modificados o sólo una parte) haciendo clic en el icono +:



- Los ficheros seleccionados se muestran en un apartado "Cambios almacenados provisionalmente". Escriba el mensaje descriptivo del commit en la caja superior y pulse **Ctrl+Intro**



- Los ficheros incluidos en el commit dejan de mostrarse en la parte superior y en el contador del icono Control de código fuente:

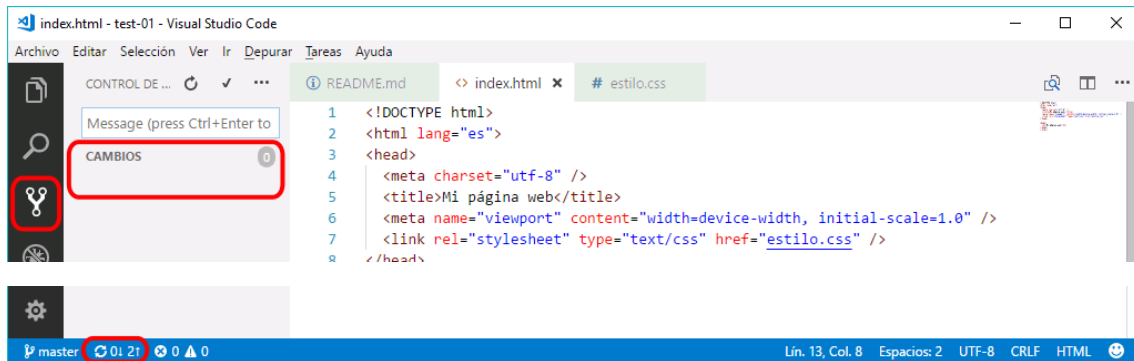


- En la barra inferior, Visual Studio Code nos avisa que el repositorio local ya no está sincronizado con el repositorio remoto y nos indica el número de commits de diferencia:



- Podemos hacer nuevos commits hasta que no queden cambios pendientes.

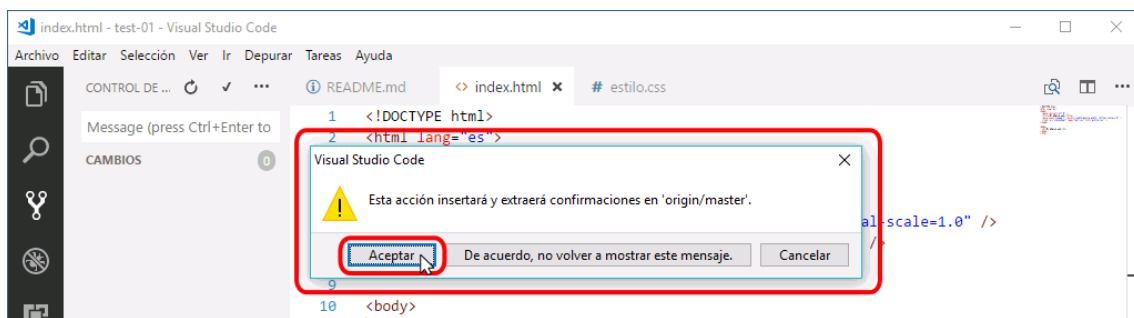




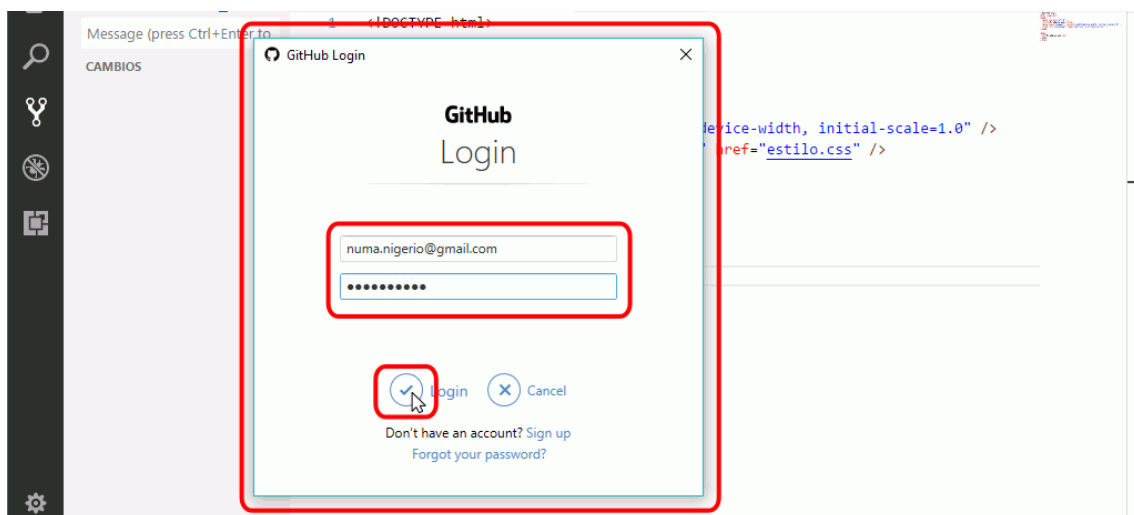
- Haga clic en el icono de la barra lateral inferior para sincronizar el repositorio remoto con el local



- Visual Studio Code mostrará un aviso indicando que al sincronizar se producirán cambios en el repositorio remoto. Para no volver a ver este aviso, haga clic en "De acuerdo, no volver a ver este mensaje":



- La primera vez que sincronicemos, deberemos introducir el usuario y contraseña de GitHub.



- Una vez realizada la sincronización, el contador deja de mostrarse.



- Puede comprobar visitando el repositorio remoto en el navegador que se han realizado los cambios.

