

Desarrollo ágil de Software

Actividad: setup (estimado 60 min)

Objetivo

- Dejar instalado y configurado el ambiente que vamos a utilizar durante el curso
 - Visual Studio Code y extensiones
 - Docker
 - Imagen docker del curso
 - Prueba de colaboración

Pasos a seguir

Prerrequisitos:

- Equipo con Windows 10 con al menos 8GB de RAM
- MacOS/Linux también pueden usarse pero los pasos de instalación pueden variar
- Buena conexión a Internet

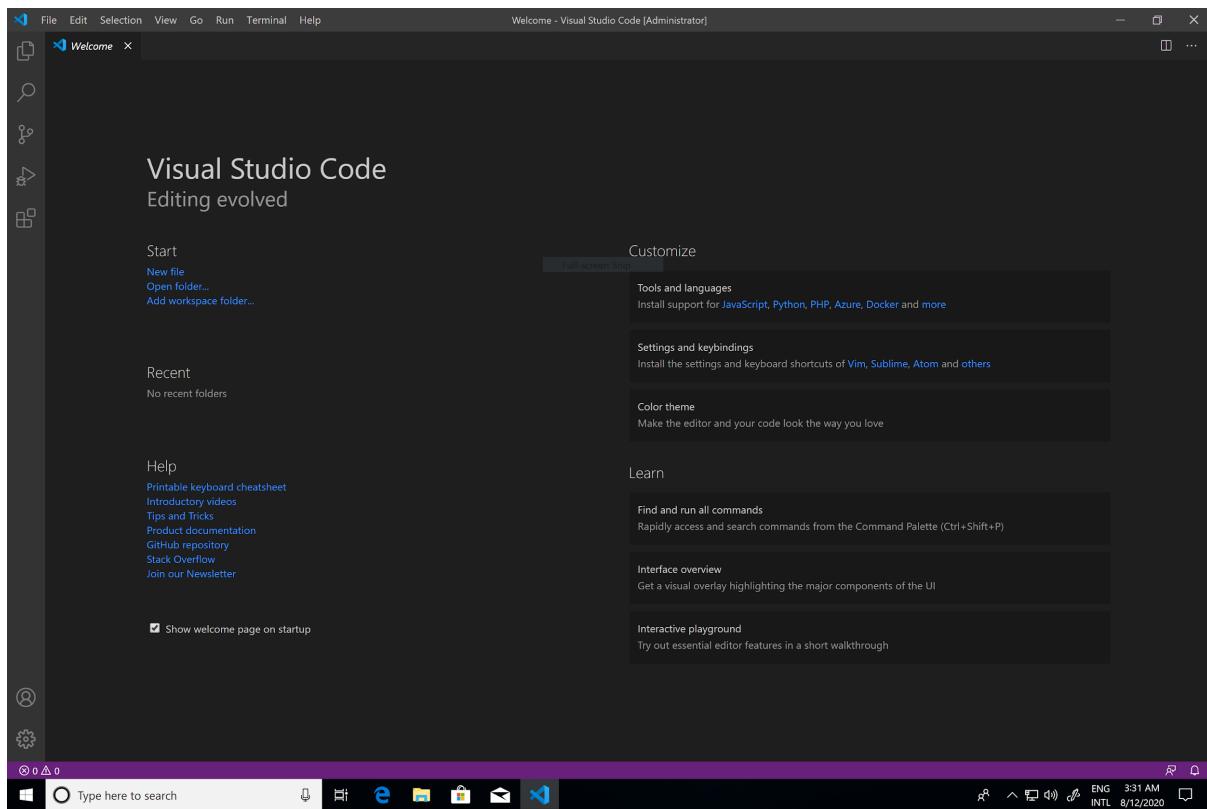
Instalación de herramientas utilizadas

Las herramientas que vamos a estar instalando son tres: **Visual Studio Code**, **Docker** y **GIt**

1. Descargar e instalar **Visual Studio Code** desde [la siguiente dirección](#)

La instalación default funciona sin problemas, no es necesario hacer ninguna configuración en particular.

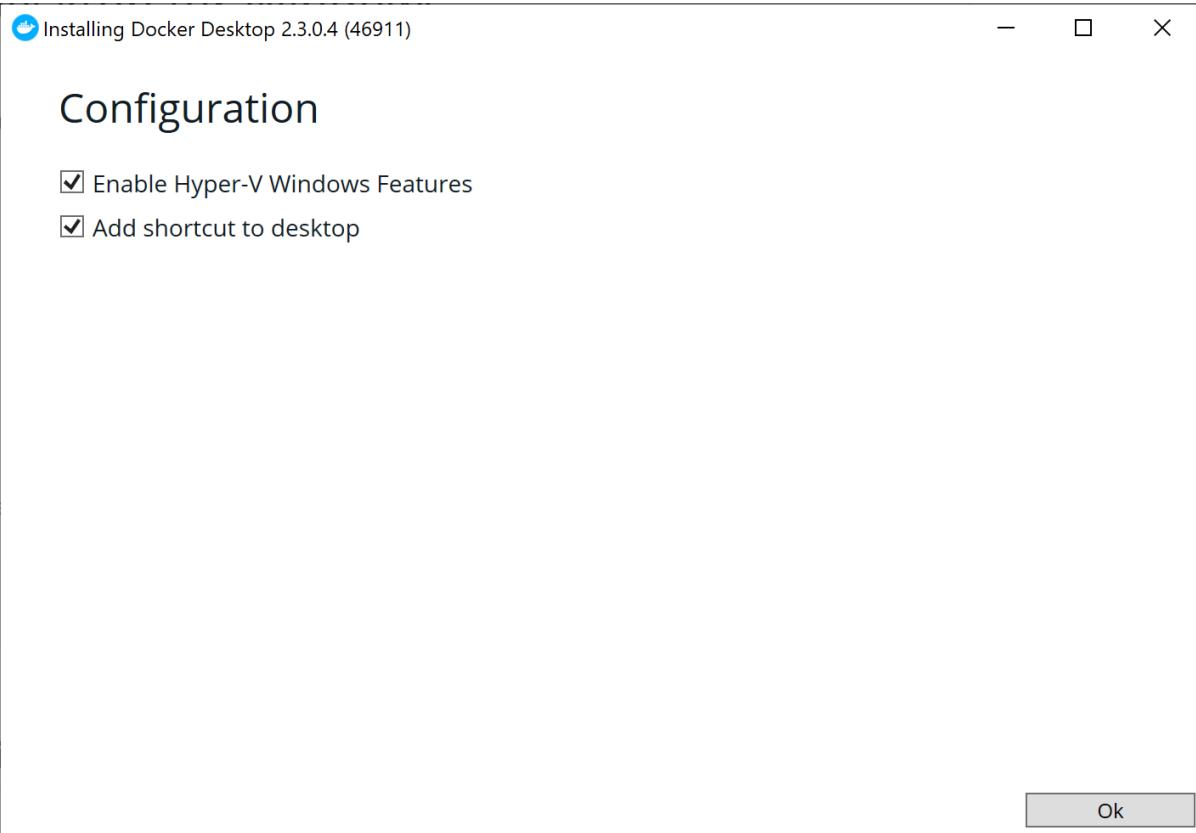
Para verificar la instalación alcanza con probar que Visual Studio Code se pueda abrir sin errores.



2. Descargar e instalar **Docker** desde [la siguiente dirección](#)

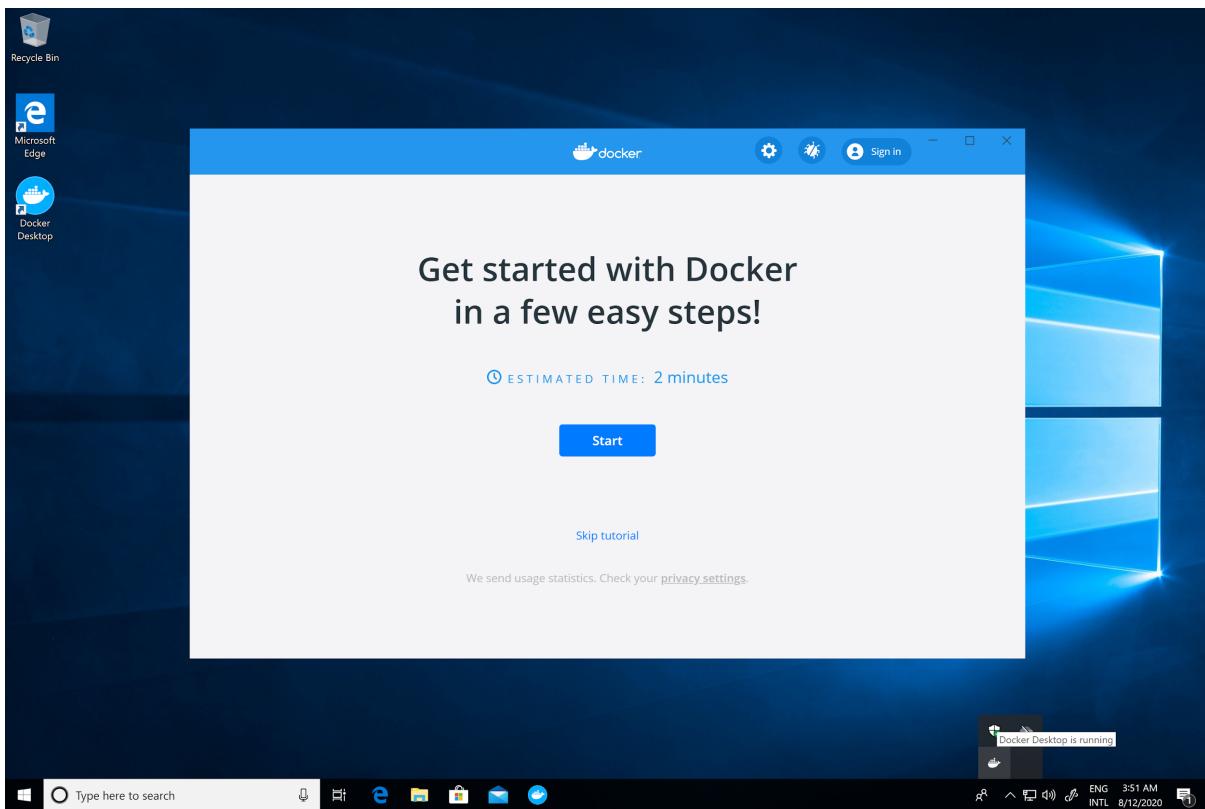
Elegir la última versión estable (2.3.0.4 al momento de escribir esta línea)

La instalación default funciona sin problemas incluyendo dejar seleccionado “Enable Hyper-V Windows Features” durante la instalación.



Una vez finalizada la instalación Docker nos va a pedir que reiniciemos Windows

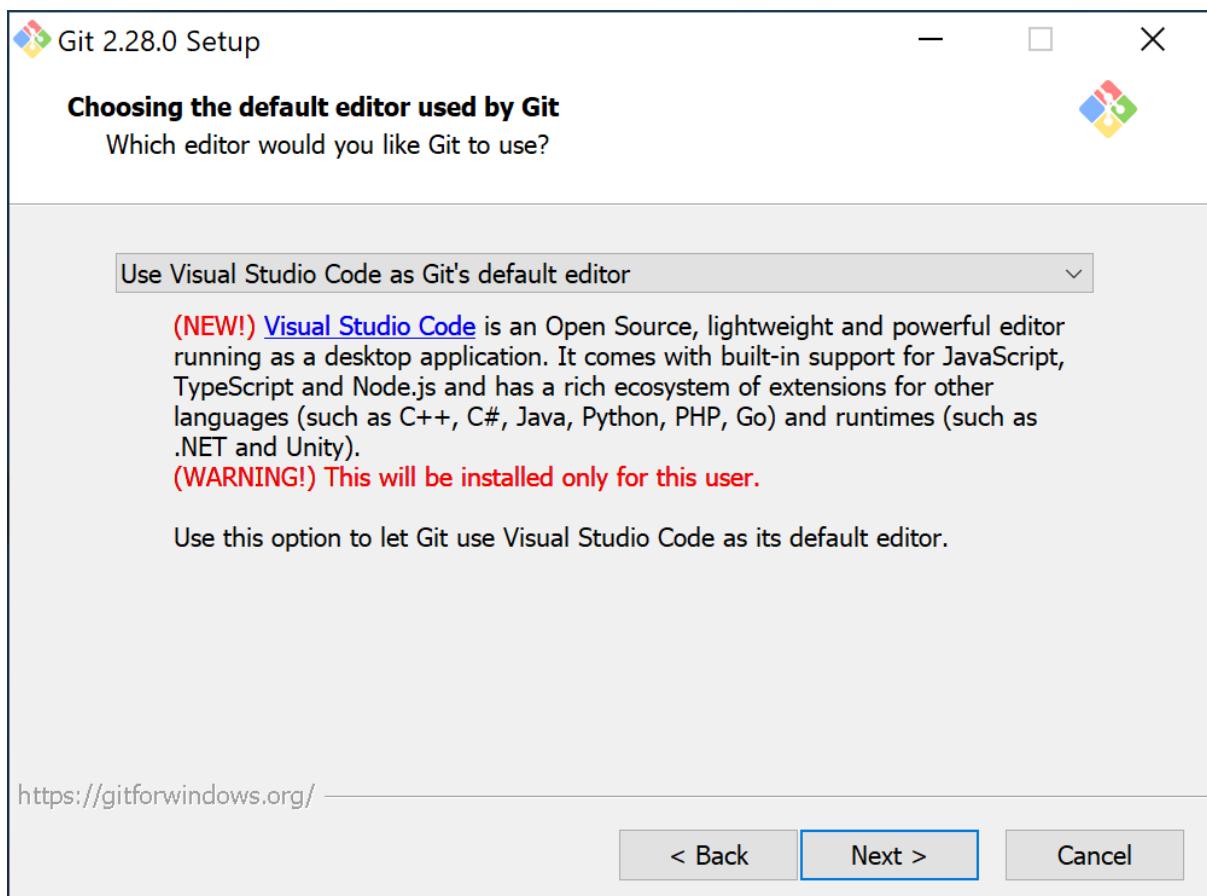
Para verificar la instalación luego reiniciar Windows verificar que el ícono de Docker aparezca en el área de notificaciones e indique que Docker está funcionando (a veces demora en iniciar)



Adicionalmente a esto Docker nos ofrece hacer un tutorial que por el momento recomendamos no realizar.

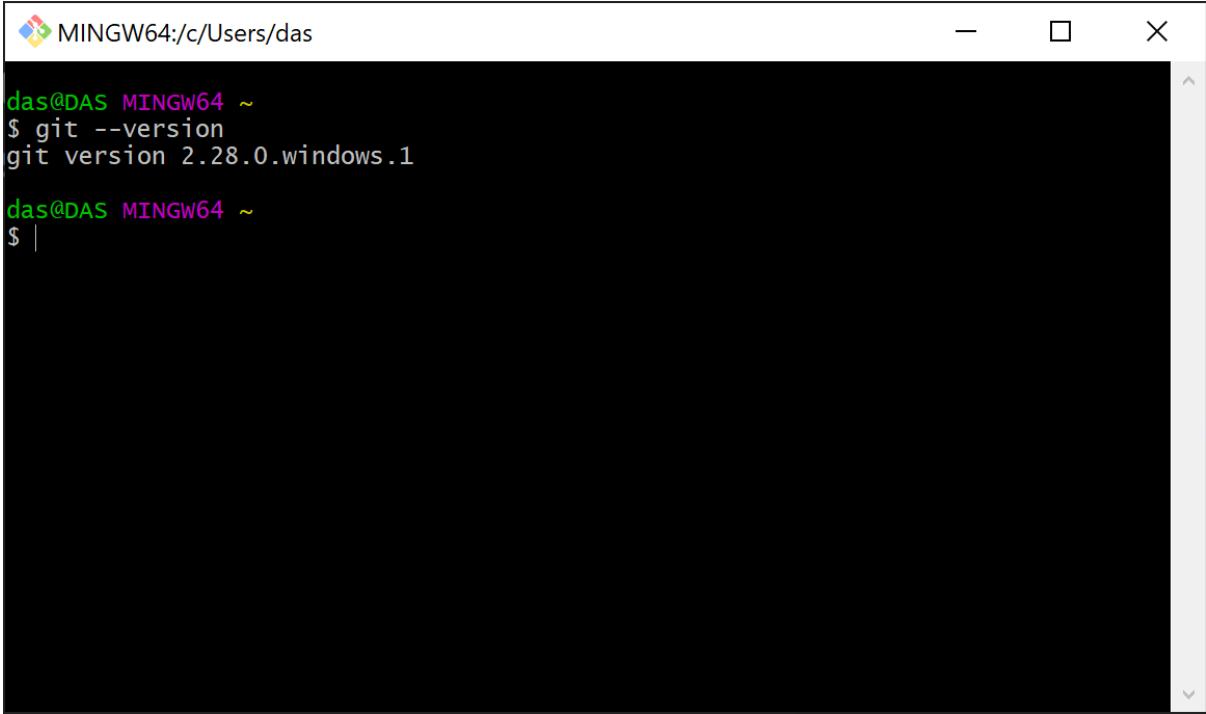
3. Descargar e instalar **Git** desde [la siguiente dirección](#)

La única opción a modificar durante la instalación es la que nos pide seleccionar el editor por default de Git, en esa opción tenemos que seleccionar Visual Studio Code. El resto de la instalación puede hacerse con las opciones por default.



Para verificar la instalación de Git podemos buscar ejecutar la App Git Bash desde el menú de inicio de Windows y escribir:

```
git --version
```



```
das@DAS MINGW64 ~
$ git --version
git version 2.28.0.windows.1

das@DAS MINGW64 ~
$ |
```

Listo! Si llegaste hasta acá ya tenes instalado tres de los productos que más vamos a utilizar durante el curso!

Setup del ambiente de desarrollo

1. Instalación de las Extensiones de Visual Studio Code

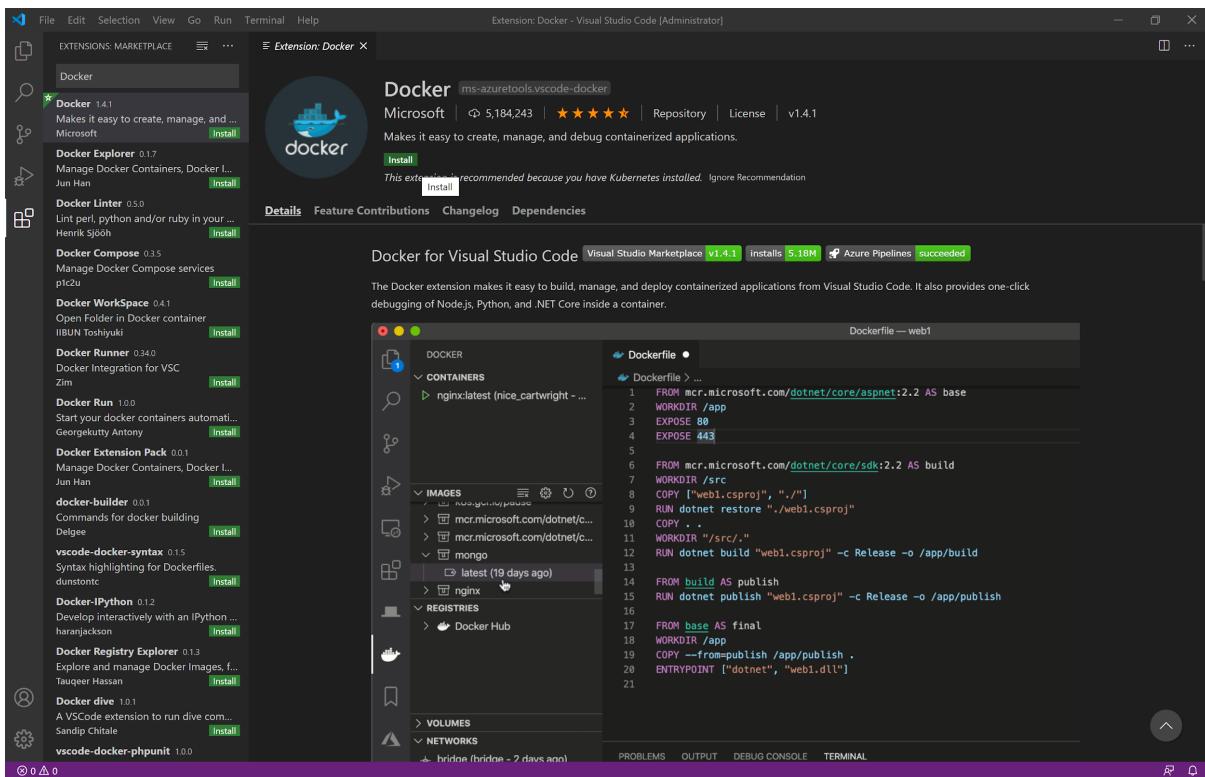
Visual Studio Code cuenta con un gran ecosistema de Extensiones para múltiples propósitos, nosotros en particular vamos a estar usando las siguientes:

Docker

Live Share

Ruby

Para instalar una Extensión alcanza con buscarla dentro de Visual Studio Code en el menú de la izquierda y seleccionarla para instalar. Abajo pueden ver un ejemplo de la instalación de la Extensión de Docker, el resto de las Extensiones se instalan de manera similar.

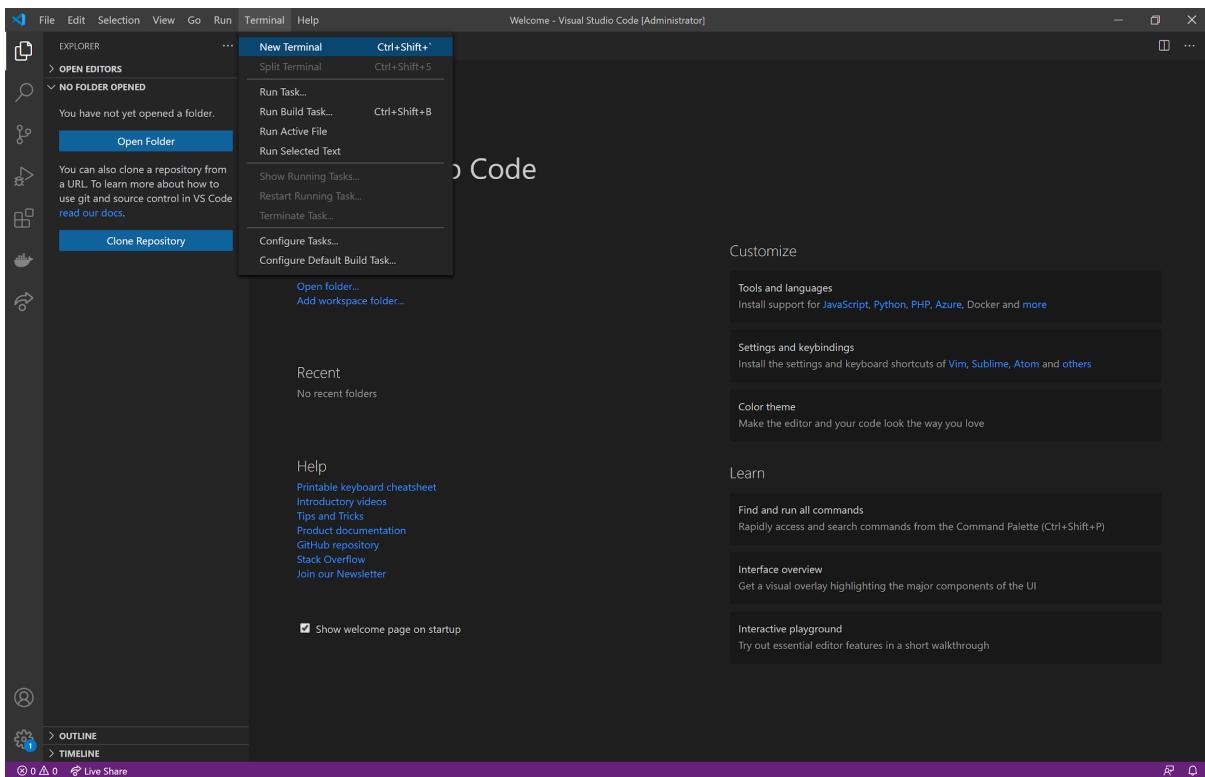


2. Clonar desde Github el proyecto que vamos a estar usando de base para el curso

Si bien Visual Studio Code no es necesario para ejecutar este paso, ya que podemos usar Git desde afuera de Visual Studio Code, vamos a ejecutar Git desde dentro de Visual Studio Code porque es la forma en la cual lo vamos a estar usando durante el curso.

Abrir Visual Studio Code

Abrir un Terminal desde el menú Terminal para que Visual Studio Code nos habilite el uso de una ventana de Powershell (por default lo hace en la parte inferior de la pantalla)

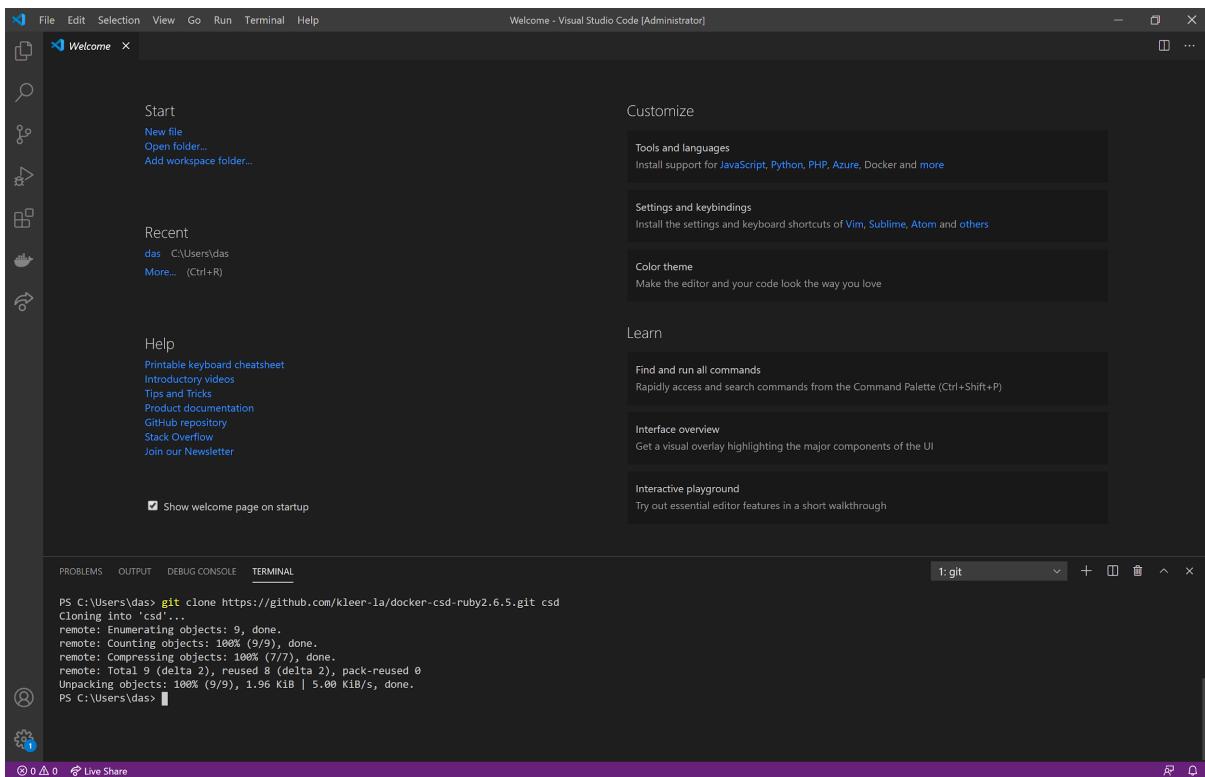


Ejecutar el siguiente comando de Git para clonar el proyecto

```
git clone https://github.com/kleer-la/docker-csd-ruby2.6.5 csd
```

Este comando clona el proyecto dentro de una carpeta llamada **csd** en la carpeta desde el cual lo estas ejecutando.

Lo más probable es que esa carpeta sea **c:\users\nombreusuario** por lo cual el proyecto te va a quedar en **c:\users\nombreusuario\csd**

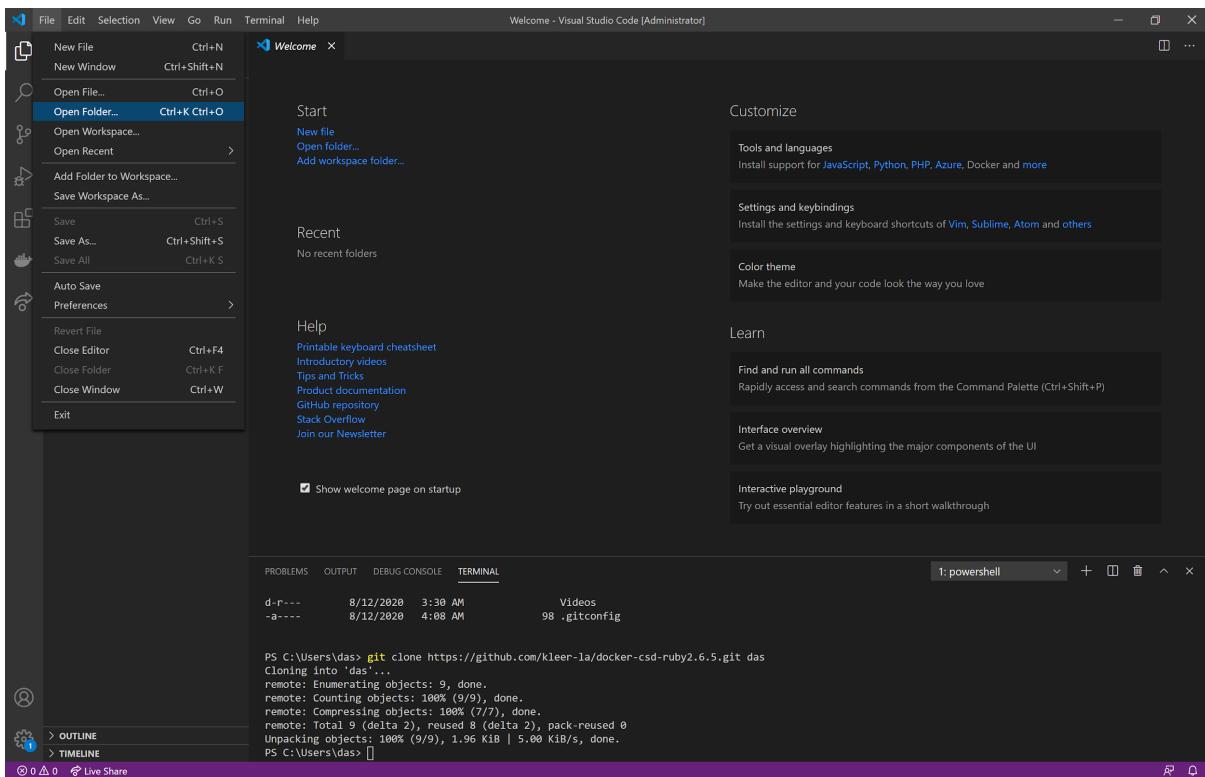


En el ejemplo de la imagen el proyecto queda en la carpeta **csd** del usuario **das**

Sino conoces **Git** ni como funciona el comando **clone** no te preocupes, lo vamos a ver y explicar durante el curso. Lo mismo aplica para **Docker**.

Por el momento solo tenemos que hacer algunos últimos pasos para verificar que todo funciona sin problemas.

Abrir la carpeta donde descargaste el proyecto desde Visual Studio Code



Desde la terminal ejecutar

```
docker-compose run --service-ports csd bash
```

Este comando puede demorar unos minutos, ya que descarga la imagen de docker que vamos a estar usando para todo el curso.

Es muy probable que Windows te pida algunas autorizaciones (File Sharing y Firewall) cuando ejecutas este comando, **por favor, aceptalas, las necesitamos.**

Si todo funcionó bien tu terminal debería estar ejecutando bash en el Container de Docker y mostrarte algo como **bash-5.0#**

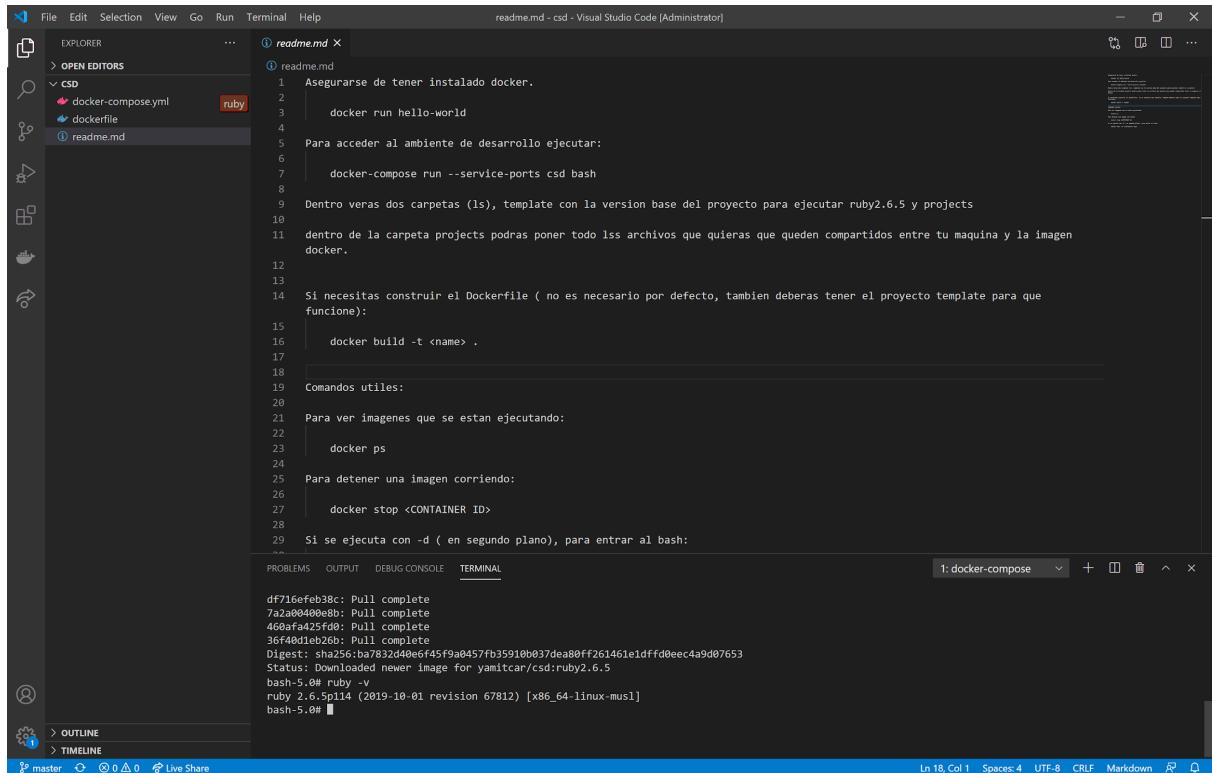
Desde esa terminal ejecuta:

```
bash-5.0# ruby -v
```

y deberías obtener:



```
ruby 2.6.5p114 (2019-10-01 revision 67812) [x86_64-linux-musl]
```



The screenshot shows a Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Terminal:** readme.md - csd - Visual Studio Code [Administrator].
- Explorer:** OPEN EDITORS, CSD, docker-compose.yml, ruby, dockerfile, readme.md.
- Content:** A README.md file containing instructions for running Ruby code in a Docker container. The text includes commands like `docker run hello-world`, `docker-compose run --service-ports csd bash`, and `docker build -t <name>`. It also provides information about Dockerfile template usage and useful commands like `docker ps` and `docker stop`.
- Bottom Status Bar:** PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, 1: docker-compose, Line 18, Col 1, Spaces: 4, UTF-8, CRLF, Markdown, Live Share.

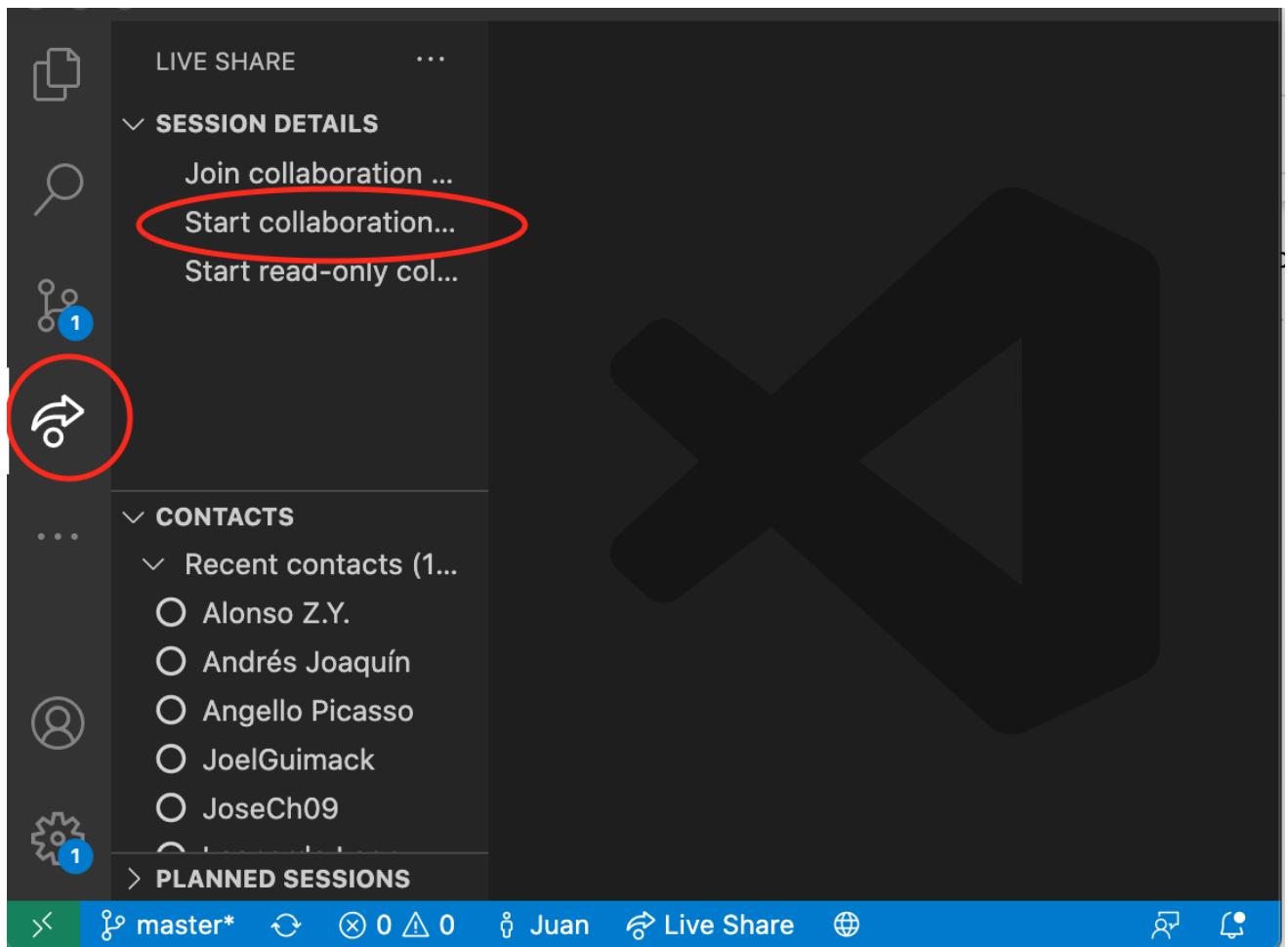
Excelente! Ese fue el último paso que necesitabas para tener todo instalado y configurado para poder empezar a jugar con **Ruby!** 🎉

3. BONUS. Iniciar una sesión de colaboración con Live Share

Para trabajar colaborativamente, usaremos Live Share, que permite a varias personas interactuar en el desarrollo en forma distribuida editando, ejecutando comandos en la terminal y usando la aplicación.

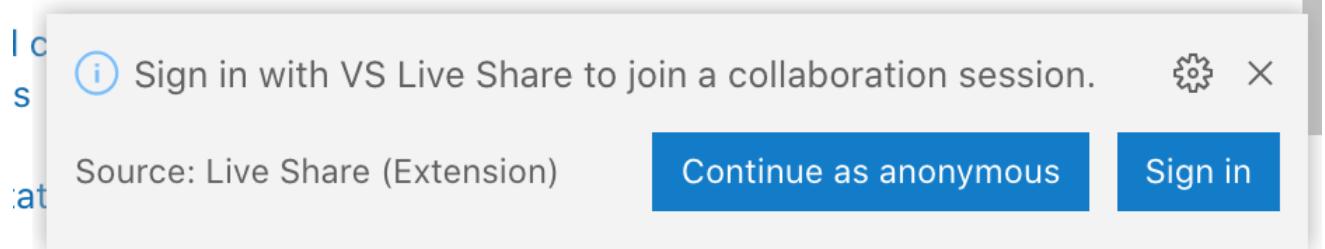
En una sesión de colaboración de Live Share una persona inicia la colaboración y genera un URL, que le pasa a los participantes. Estos pueden sumarse a la sesión autenticados (por GitHub o Microsoft) o anónimos.

Para probar mínimamente Live Share, te pedimos que inicies una sesión de colaboración. Haz click en el ícono de Live share en la barra de la izquierda y luego en la opción de iniciar una colaboración (*Start collaboration...*). Te pediría identificarte, puedes usar tu usuario de GitHub.



Iniciarás una sesión y dejará el URL en tu portapapeles. En un caso normal le pasarías esta URL a tus compañeros de equipo.

Ahora, para probar, pegas esa URL en un campo de dirección de tu navegador. Te aparecerá un mensaje:

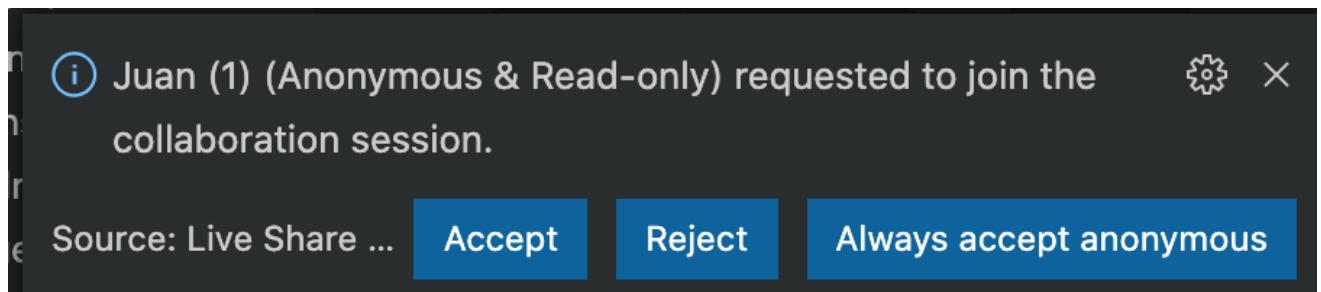


Continúa como anónimo, elige un nombre para tu conexión anónima.

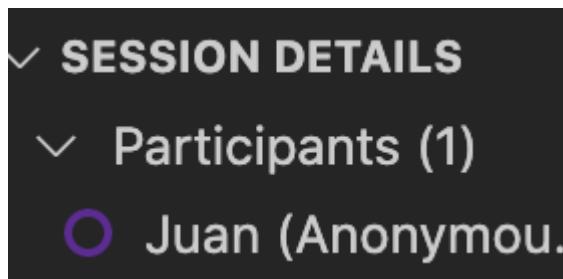
Juan

Enter the name to display to other participants (Press 'Enter' to confirm or 'Escape' to cancel)

En tu VS Code te avisará que alguien está intentando sumarse,



Aceptate :D y aparecerás en la lista de participantes



Finalizamos la prueba de colaboración, ya puedes cerrar la sesión

