Para usuarios de Windows y Mac OS: entorno de desarrollo con Docker y Visual Studio Code

Índice

Este documento es un manual breve destinado a los usuarios de los sistemas operativos Windows y Mac OS, que describe cómo configurar el entorno de desarrollo utilizando Visual Studio Code, Docker y la extensión Dev-Containers. Los usuarios de GNU/Linux no necesitan usar contenedores en esta asignatura.

1 Instalación de Docker Desktop

Para comenzar, necesitamos instalar Docker Desktop en nuestro sistema:

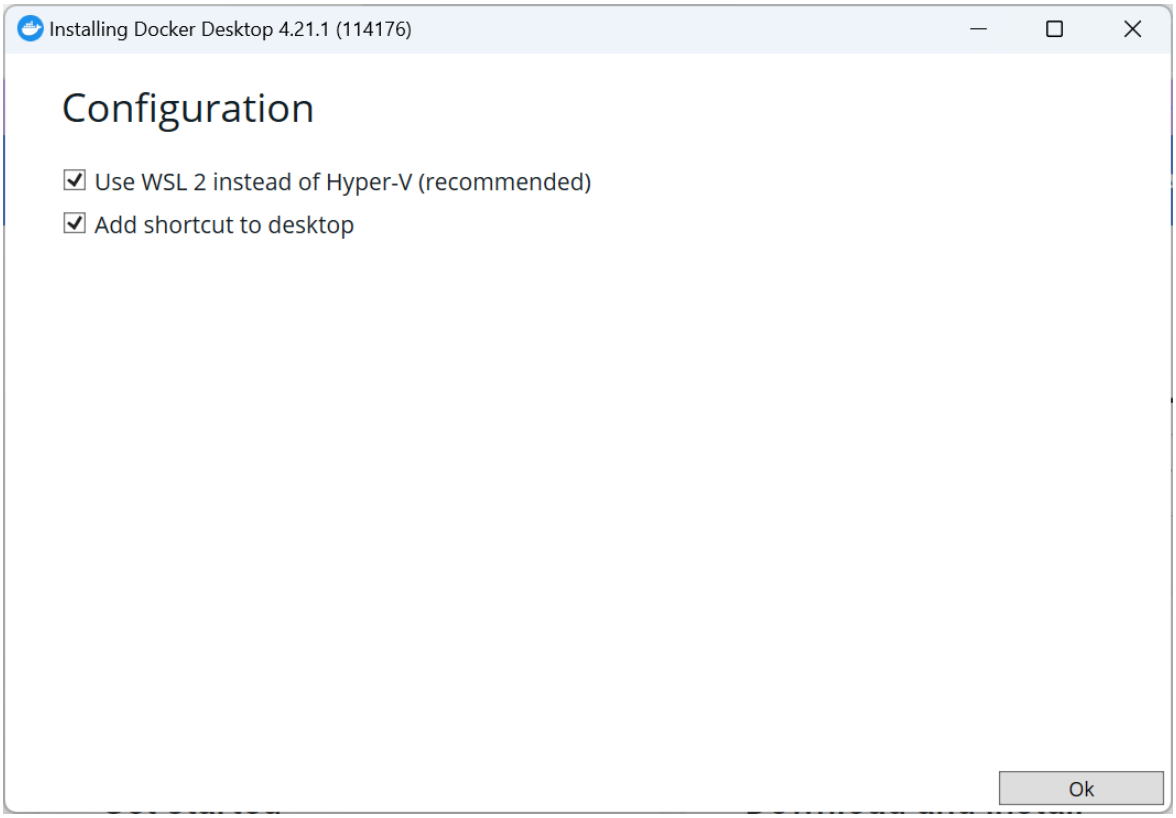
1. Descargamos e instalamos Docker Desktop desde el sitio web oficial de Docker según el sistema operativo que estemos utilizando (Windows o Mac OS).
2. Después, conviene asegurarse de que Docker Desktop se esté ejecutando correctamente.

Nota para usuarios de MS Windows: Al iniciar Docker Desktop, es posible que se muestren avisos importantes. Para solucionarlos:

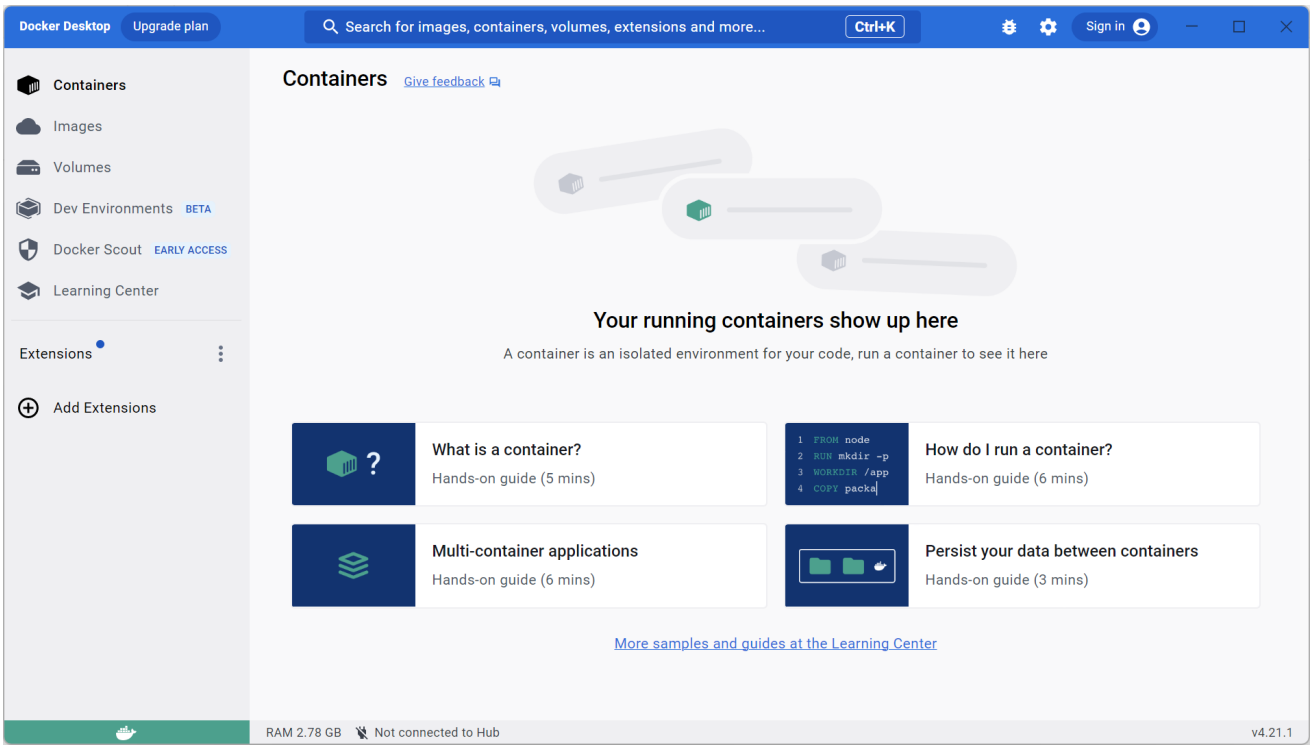
- Verificamos si la virtualización está habilitada en la BIOS de nuestro equipo.
- Comprobamos si es necesario actualizar el Windows Subsystem for Linux (WSL). Para hacerlo, abrimos una Power Shell y ejecutamos los siguientes comandos:

```
wsl --install
wsl --update
```

A continuación, se muestra una captura de pantalla de Docker instalándose:



Y otra captura con Docker en funcionamiento:



2 Instalación de Visual Studio Code

Continuamos instalando Visual Studio Code:

1. Descargamos e instalamos Visual Studio Code desde el sitio web oficial.
2. Hay que asegurarse de que el pack de extensiones "Remote Development" esté instalado en Visual Studio Code. Podemos instalarlo desde la sección de Extensiones de Visual Studio Code buscando "Remote Development" y haciendo clic en "Instalar".

3 Estructura de una carpeta de proyecto

Para que VSCode funcione contra un contenedor debemos tener una carpeta de proyecto con la siguiente estructura:

- `PARProject`: esta es la carpeta principal del proyecto. Podemos alterar su nombre si lo deseamos.
  - `.devcontainer`: una subcarpeta que contiene la configuración del contenedor y el entorno. Es importante notar que comienza con un punto (.) para hacerla oculta.
    - `Dockerfile`: un archivo que configura la imagen del contenedor
    - `devcontainer.json`: un archivo que configura la apariencia y los complementos iniciales de Visual Studio Code que se ejecutarán en el contenedor.
  - Además, dentro de la carpeta `proyecto`, podemos organizar el código fuente como deseemos.

La estructura de directorio mencionada anteriormente, junto con los archivos Dockerfile y devcontainer.json, se proporcionan en el archivo comprimido entorno-vscode-docker.zip.

3.1 Nota para los usuarios de GNU/Linux

Si abrimos un proyecto con una carpeta `.devcontainer` y el complemento `Dev Containers` está instalado (viene con `Remote Development`), pueden ocurrir varias situaciones:

1. Si tenemos Docker instalado: se puede crear el contenedor si lo permitimos. Sin embargo, esto es totalmente **innecesario** en GNU/Linux, ya que podemos compilar de forma nativa.
2. Si no tenemos Docker instalado o no tenemos permisos para crear contenedores: Visual Studio Code puede mostrar algunas advertencias o mensajes de error. No obstante, podemos trabajar con normalidad.

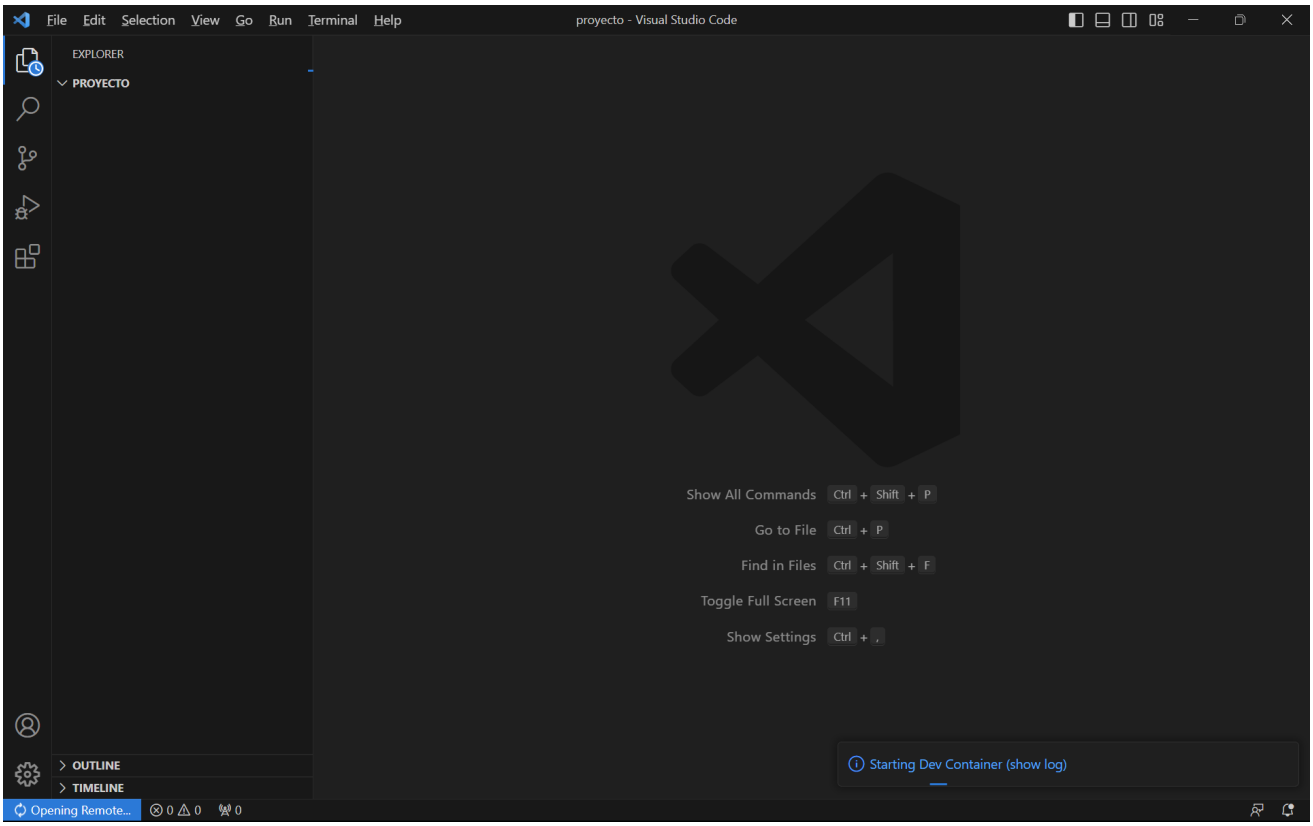
La recomendación, si trabajamos en un entorno mixto (desarrollando en Windows/Mac y Linux por ejemplo), es no copiar la carpeta `.devcontainer` al pasar los archivos de Windows/Mac OS a GNU/Linux.

4 Conexión al contenedor

Una vez que tengamos la estructura completa de la carpeta de trabajo proyecto, junto con su carpeta `.devcontainer`, procedemos de la siguiente manera:

1. Abrimos la carpeta de trabajo con Visual Studio Code.
2. Cuando se nos pregunte si deseamos abrirla con Dev Containers, respondemos afirmativamente.
3. Visual Studio Code comenzará a construir y configurar el contenedor según las especificaciones del Dockerfile y devcontainer.json. La primera vez que se ejecute, esto puede llevar unos minutos.
4. Una vez completado el proceso, la barra de estado de Visual Studio Code mostrará "Dev Container" junto al nombre de la carpeta de trabajo (ver esquina inferior izquierda de ventana de VSCode.). Esto indica que estamos trabajando dentro del contenedor.
5. Los cambios realizados en la carpeta de trabajo se reflejarán automáticamente dentro del contenedor y viceversa. Esto se debe a que el contenedor sincroniza el directorio `/workspaces/parssdd` visible dentro del contenedor con el directorio de trabajo equivalente en el host.

A continuación, se muestra una captura de pantalla de Visual Studio Code creando el contenedor (como nombre del proyecto debería mostrarse PARProject):



5 Proceso de Desarrollo, Compilación y Ejecución

Ahora que estamos configurados y conectados al contenedor, podemos comenzar a desarrollar, compilar y ejecutar nuestros programas:

1. En el directorio de trabajo, crearemos un fichero llamado `hello.c`, y almacenaremos en él un programa simple en C de prueba:

```c
#include <stdio.h>

int main(void){
  printf("Hello World!\n");
    return 0;
}
```

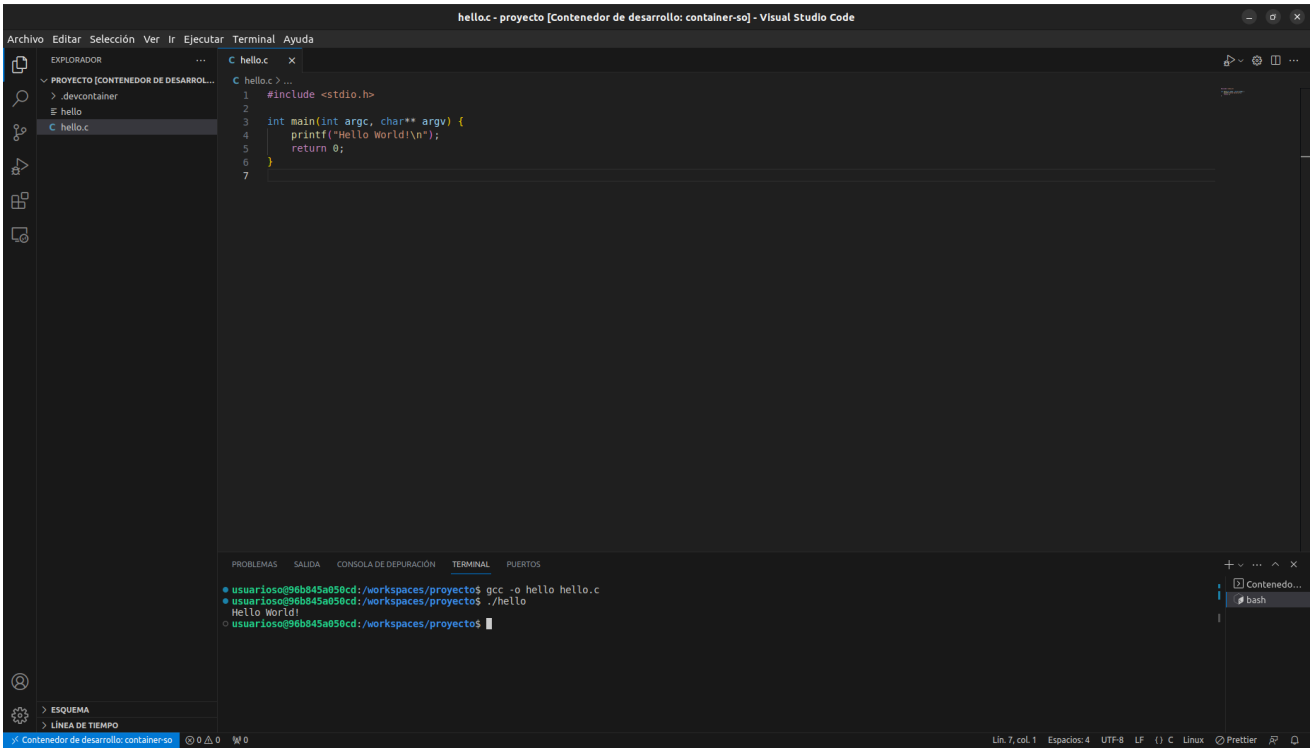
2. Para compilar y ejecutar el programa, abrimos una nueva terminal en Visual Studio Code haciendo clic en `Terminal` y luego en `New Terminal`.

3. Introducimos las siguientes órdenes en la terminal:

```
gcc -g -Wall -o hello hello.c
./hello
```

- Deberíamos ver en pantalla el mensaje `Hello World!`.
- Si modificamos el archivo `hello.c`, los cambios se reflejarán automáticamente en el contenedor y en nuestro equipo local.

A continuación, se muestra una captura de pantalla de Visual Studio Code ejecutando el programa `hello` desde la terminal (en nuestro caso el directorio actual será `/workspaces/parssdd`):



6 Enlaces de interés

La documentación completa para cada sistema operativo se puede encontrar en las siguientes fuentes:

1. Developing inside a Container: Página de Microsoft que proporciona instrucciones detalladas para configurar todo el kit de desarrollo, con variantes para Windows y MacOS. Consultar la sección "Installation" y seguir las instrucciones según el sistema operativo que estemos utilizando.
2. Install Docker Desktop on Windows: página de Docker con instrucciones para instalar Docker Desktop en Windows.
3. Install Docker Desktop on Mac: página de Docker con instrucciones para instalar Docker Desktop en MacOS.

7 Algunos comandos de utilidad para Docker

En algunas ocasiones, puede resultar más práctico trabajar exclusivamente dentro del contenedor. Para hacerlo, se proporcionan a continuación algunos comandos que pueden ser de utilidad. Para ejecutar estos comandos, es necesario contar con privilegios de administrador. En sistemas GNU/Linux, esto implica agregar sudo al principio de cada comando o asegurarse de que el usuario pertenezca al grupo docker:

```
sudo usermod -aG docker $USER
```

A continuación, se presentan los comandos más comunes:

- Crear una imagen desde un Dockerfile: `docker build -t nombre_de_la_imagen - < Dockerfile`
- Crear un contenedor: `docker run [--name nombre_del_contenedor] -it nombre_de_la_imagen`
- Crear un contenedor con mapeo de volúmenes entre host y contenedor: `docker run -v /ruta/del/host:/ruta/e [--name nombre_del_contenedor] -it nombre_de_la_imagen`
- Iniciar un contenedor ya creado pero detenido: `docker start -ai nombre_del_contenedor`

- Listar contenedores: `docker ps -a`
- Listar imágenes: `docker images -a`
- Eliminar un contenedor: `docker rm nombre_del_contenedor`
- Eliminar una imagen: `docker rmi nombre_de_la_imagen`
- Copiar archivos entre el host y contenedor: `docker cp ruta_del_archivo_local nombre_del_contenedor`