Entorno de desarrollo con Docker y Visual Studio Code

Índice

Este documento es un manual breve para poder desplegar el entorno de desarrollo utilizando Visual Studio Code, Docker y la extensión Dev - Containers.

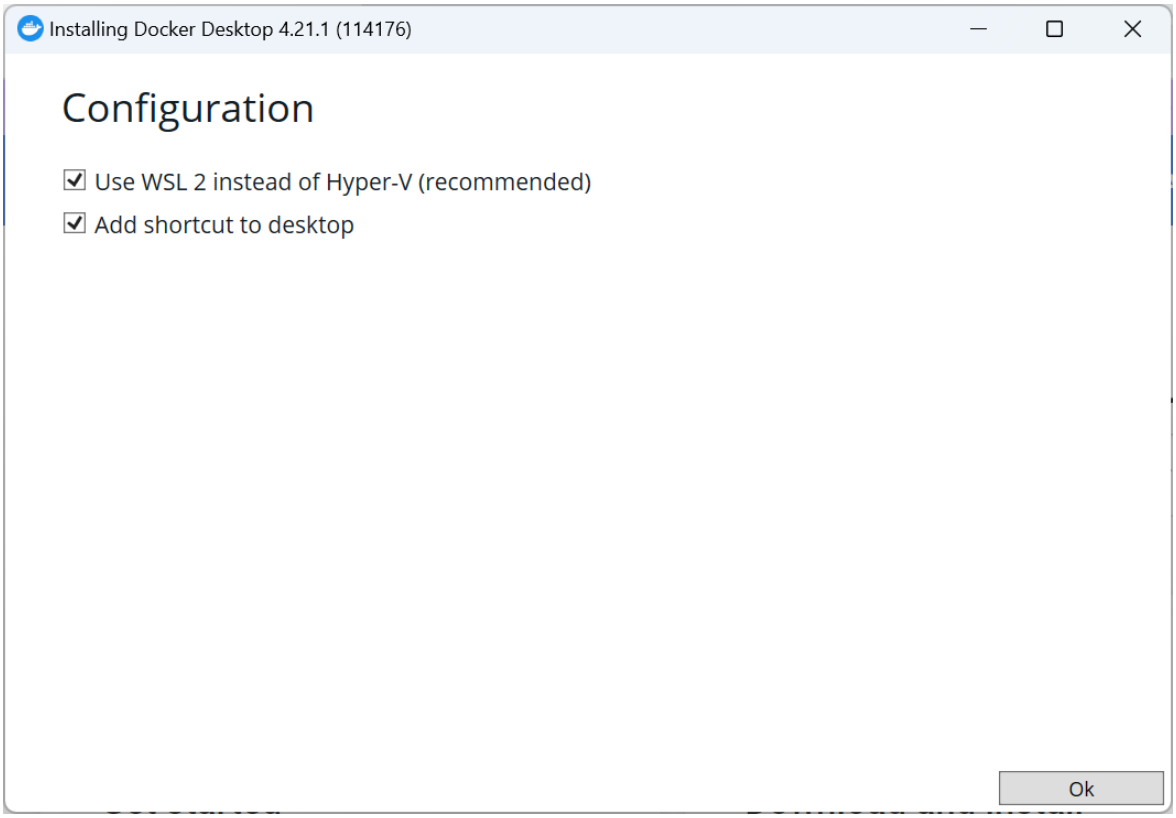
1 Instalación de Docker Desktop

- Descarga e instala Docker Desktop desde el sitio web oficial de Docker para Windows o Mac OS, según el sistema operativo que utilices.
- Asegúrate de que Docker Desktop esté ejecutándose y funcione correctamente.

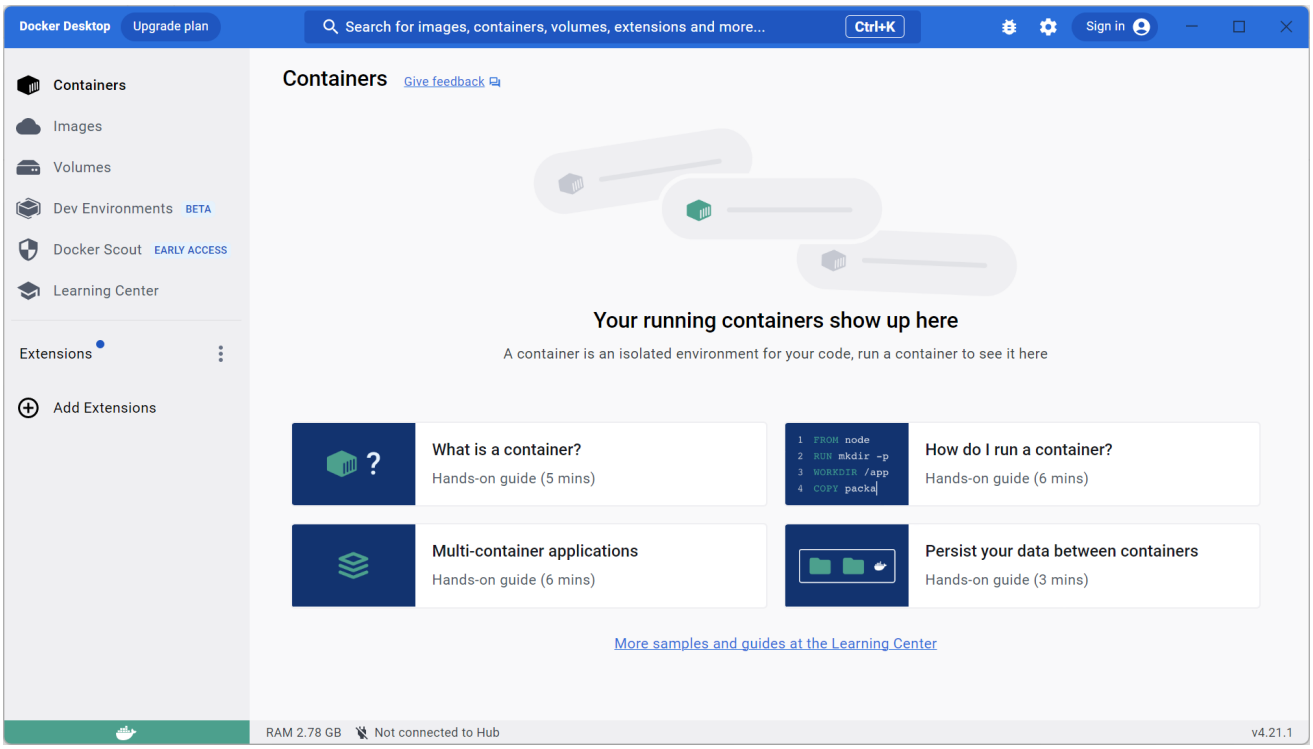
Nota para usuarios de MS Windows: Al arrancar Docker puede que te dé avisos importantes.

- Seguramente tengas que habilitar la virtualización en tu equipo desde la BIOS.
- También te puede pedir que actualices el Windows Subsystem for Linux (WSL): para ello abre una Power Shell y ejecuta los comandos: `wsl --install` (por si no está instalado) y `wsl --update`.

Aquí puedes ver una captura con Docker instalándose:



Y otra captura con Docker en ejecución:



2 Instalación de Visual Studio Code

- Descarga e instala Visual Studio Code desde el sitio web oficial.
- Asegúrate de tener instalada la extensión `Dev Containers` en Visual Studio Code. Puedes instalarla desde la sección de Extensiones de Visual Studio Code buscando "Dev Containers" y haciendo clic en "Install".

3 Estructura de una carpeta de proyecto

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

- `proyecto`: carpeta de proyecto, con el nombre que deseemos.
  - `.devcontainer`: subcarpeta de configuración del contenedor y del entorno (**fíjate en el punto inicial**).
    - `Dockerfile`: fichero que configura la imagen del contenedor
    - `devcontainer.json`: fichero que configura la apariencia y plugins iniciales del VSCode que se ejecuta en el contenedor
  - `...` y dentro de `proyecto`, el código fuente con la estructura que queramos.

A continuación se proporciona el contenido de los ficheros `Dockerfile` y `devcontainer.json`.

El fichero `Dockerfile` debe contener lo siguiente:

```
FROM debian:latest

RUN apt-get clean
RUN apt-get update --fix-missing
RUN apt-get update
RUN DEBIAN_FRONTEND=noninteractive apt-get install -y locales sudo nano gcc make gdb glibc-doc git man ba

# LOCALE
RUN sed -i '/es_ES.UTF-8/s/^# //g' /etc/locale.gen && locale-gen
ENV LANG es_ES.UTF-8
ENV LANGUAGE es_ES:es
ENV LC_ALL es_ES.UTF-8

# USUARIO
RUN useradd -m usuario -p sa5u2O0Xjsufg
RUN usermod -aG sudo usuario

USER usuario
WORKDIR /home/usuario
```

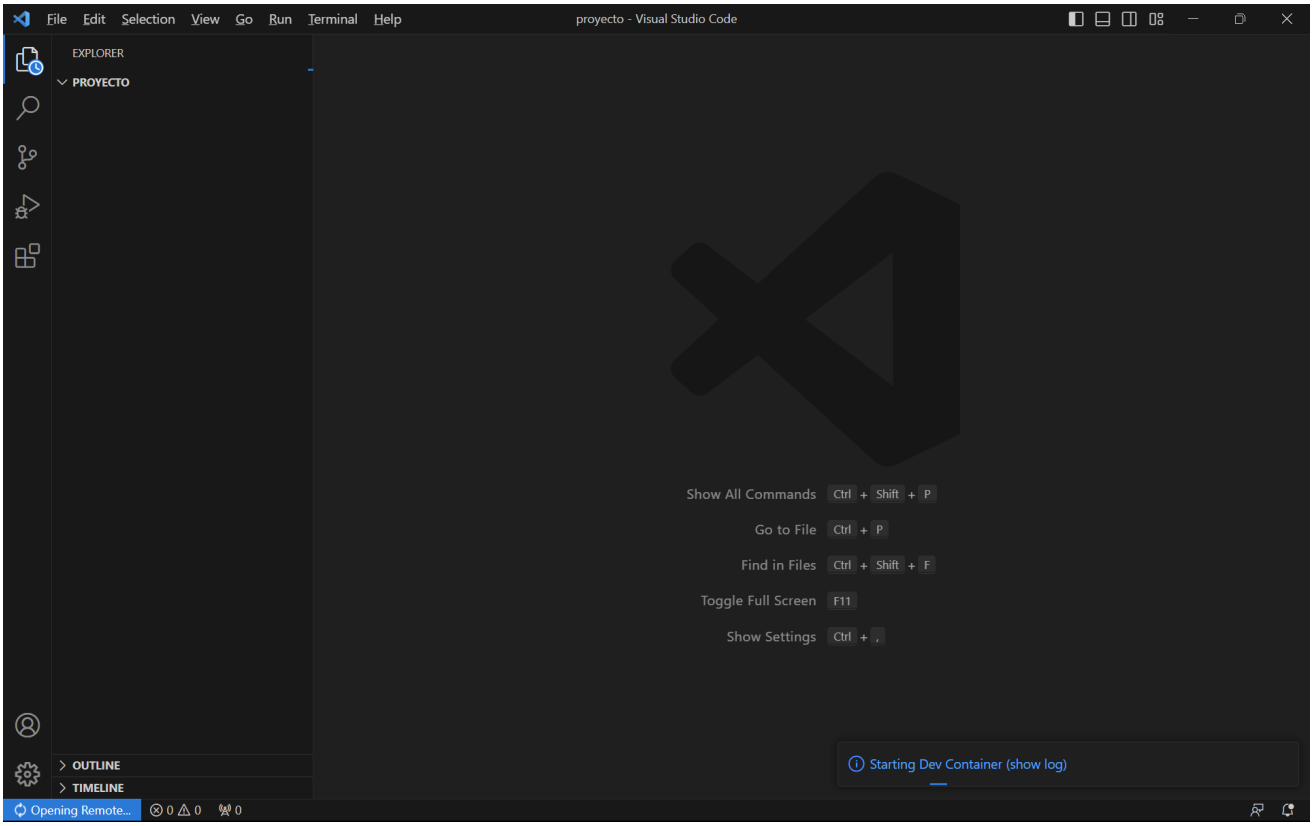
Mientras que el fichero `devcontainer.json` debe contener lo siguiente:

```
{
    "name": "container-so",
    "dockerFile": "Dockerfile",
    "settings": {
        "terminal.integrated.shell.linux": "/bin/bash"
    },
    "extensions": [
                "redhat.fabric8-analytics",
                "esbenp.prettier-vscode",
                "ms-vscode.cpptools",
                "ms-vscode.cpptools-extension-pack",
        ]
}
```

4 Conexión al contenedor

- Una vez que tengas la estructura completa de la carpeta `proyecto`, con su carpeta `.devcontainer`, abre la carpeta de trabajo con Visual Studio Code.
- Te preguntará si abrirlo con Dev Containers. Responde que sí.
- Visual Studio Code comenzará a construir y configurar el contenedor según las especificaciones del Dockerfile y devcontainer.json. Esto puede llevar unos minutos la primera vez que se ejecute.
- Una vez que se complete el proceso, la barra de estado de Visual Studio Code mostrará "Dev Container" junto al nombre de la carpeta de trabajo. Esto indica que estás trabajando dentro del contenedor.
- Los cambios realizados en la carpeta de trabajo se reflejarán dentro del contenedor y viceversa.

Esta es una captura de pantalla con VSCode creando el contenedor



5 Proceso de desarrollo, compilación y ejecución

- Crea un nuevo archivo dentro del proyecto llamado `hello.c`
- Introduce el siguiente código:

```c
#include <stdio.h>

int main(int argc, char** argv) {
    printf("Hello World!\n");
    return 0;
}
```

- Haz click en `Terminal`, `New Terminal`.
- Introduce las siguientes órdenes:

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

- Deberías ver en pantalla `Hello World!`.
- Prueba a modificar `hello.c`. Los cambios se deberían efectuar también en tu equipo local.

Aquí se puede ver una captura de pantalla con VSCode ejecutando el programa `hello`: