# QUÉ ES DOCKER

Docker empaqueta el software en unidades estandarizadas llamadas containers.

Los containers tienen todo lo que el software necesita para ser ejecutado (librerías, herramientas de sistema, código, etc…)

Desplegar y escalar aplicaciones se hace muy rápido a cualquier entorno.

Un contenedor es una unidad autónoma que contiene todas las dependencias que el software necesita y que permite desplegar de un servidor a otro (laptop, proveedor de nube, etc…) y así el contenedor será ejecutado de la misma manera.

El container es un proceso aislado que incluso tiene su propio sistema de archivos.

Una VM virtualiza el hardware del servidor

Un container virtualiza el sistema operativo del servidor

|  |  |
| --- | --- |
| VIRTUAL MACHINE | CONTAINER |
| Es una copia completa del sistema operativo | Comparte del kernel del host con otros contenedores |
| Puede mantener aplicaciones separadas que se ejecutan sobre el mismo hardware | Aíslan los ambientes de ejecución junto con todas sus dependencias y comparten el proceso de ejecución de los ambientes |
| Cada VM requiere su propio sistema operativo, esto implica que se más grande y pesada y difícil de mantener y actualizar | Comparten el sistema operativo del host |
|  | Con construidos a partir de imágenes (templates) de solo lectura |
|  | Un contenedor es la instancia de una imagen |
|  | Una imagen se convierte en un contenedor una vez es ejecutada |
|  | Cada contenedor es independiente del host, tiene su propia configuración de red, almacenamiento y procesamiento de espacio |
|  | Una imagen se crea utilizando un Dockerfile (que es un archivo con todas las instrucciones necesarias para crear la imagen) |

A partir de un “Dockerfile” se crea una “Imagen” y a partir de una imagen se crea un contenedor

Dockerfile -> Image -> Container

# INSTALACION DOCKER EN WINDOWS

1. Descargar Docker <https://www.docker.com>
   1. Developers
   2. Getting started
   3. Download for Windows
2. Buscar en Google “Download for Linux kernel update package”
   1. En el paso 4 descargar e instalar “WSL2 Linux kernel update package for x64 machines”
   2. Abrir powershell y ejecutar “wsl --set-default-version 2” (paso 5)
   3. En el Microsoft store instalar y abrir “Ubuntu”
   4. Ejecutar Docker desktop
   5. En Ubuntu escribir “docker run -d -p 80:80 docker/getting-started”
   6. En Ubuntu escribir “docker pull hello-world”
3. En Docker desktop -> Images > Hello-world -> Ejecutar

# DESPLEGAR APLICACIÓN EN DOCKER

# Publicar localmente

1. Seleccionar el proyecto web (MVC) clic derecho y “open in terminal”
2. Escribir en la terminal “dotnet restore”. Esto asegura a todos los paquetes nuget que están especificados en el archivo del proyecto
3. Escribir en la terminal “dotnet publish -c Release -o out”. Esto publica toda la aplicación en la carpeta out
   1. Verificar que se haya creado la carpeta out con la publicación en el file explorer
4. Escribir en la terminal “dotnet out/DockerDotNetApp.dll”. DockerDotNetApp es el nombre de la aplicación web (MVC) y tiene el nombre de la dll del proyecto web (MVC)
5. Abrir el navegador y escribir <http://localhost:5024>. Este puerto (5024) fue el que se configuró en la aplicación web (MVC) en el cuerpo del método “CreateHostBuilder” de la clase “Program.cs” así:

public static IHostBuilder CreateHostBuilder(string[] args) =>

Host.CreateDefaultBuilder(args)

.ConfigureWebHostDefaults(webBuilder =>

{

webBuilder.UseStartup<Startup>();

//TODO: No importa el DNS, solo importa el puerto aleatorio que se eligió (5024).

//Para evitar que docker tenga problemas con el puerto

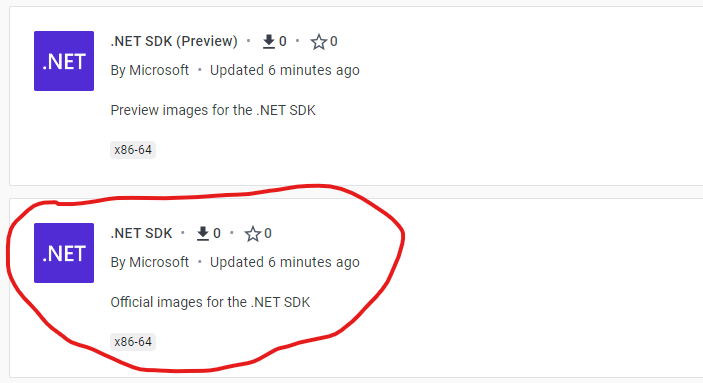
webBuilder.UseUrls("http://\*:5024");

});

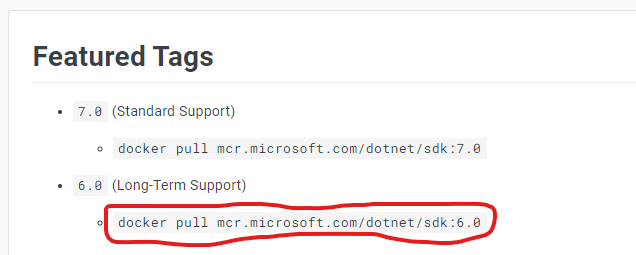
1. En la terminal dar ctrl + c. Esto detiene la publicación.

# Publicar en Docker

1. Crear un archivo con el nombre “Dockerfile” (sin extensión) al mismo nivel donde se encuentra el archivo del proyecto “DockerDotNetApp.csproj”. DockerDotNetApp es el nombre de la aplicación web (MVC).
2. Agregar este archivo a la solución desde visual studio. Agregar archivo existente/Add existing item.
3. Ingresar a la página <https://hub.docker.com>.
   1. Escribir en el campo de búsqueda “.net SDK”
   2. Seleccionar el SDK establecido (Nunca el preview)



* 1. En “Featured Tags” copiar el texto(según el framework con el cual se construyó la aplicación):



1. Editar el archivo “Dockerfile” así:

#TODO: Solicita a docker que obtenga la siguiente imagen.

#TODO: Docker va a buscar la imagen oficial del sdk de .net 6.0 y se renombra a 'build'

FROM mcr.microsoft.com/dotnet/sdk:6.0 AS build

#TODO: Define un nombre para la aplicación (en este caso 'webapp')

#TODO: Este es un nombre para el contenedor

WORKDIR webapp

#TODO: Expone los puertos 80 y 5024 (Este último fue el que se definio en el método 'CreateHostBuilder' de la clase 'Program.cs')

EXPOSE 80

EXPOSE 5024

#TODO: Copia los archivos de proyecto en el mismo directorio

#TODO: Archivos que estén en el directorio en el cual está ubicado (este archivo ''DockerFile'. Por esta razón se creó el DockerFile a nivel del archivo de proyecto 'DockerDotNetApp.csproj')

COPY ./\*.csproj ./

#TODO: Verifica que el archivo de proyecto tenga todas las dependencias que necesita.

RUN dotnet restore

#TODO: Copia el resto de las cosas.

#TODO: Copia el resto al mismo contenedor.

#TODO: Estamos moviendo archivos desde el build de esta aplicacion hacia adentro del contenedor de docker

COPY . .

#TODO: Obtiene los artefactos que docker va a necesitar para ejecutar la aplicación (tomar las dll's y ejecutarlas)

RUN dotnet publish -c Release -o out

#TODO: Construir la imagen (hacer un build de la imagen) que luego se va a agregar a un contenedor.

FROM mcr.microsoft.com/dotnet/sdk:6.0

WORKDIR /webapp

COPY --from=build /webapp/out .

ENTRYPOINT ["dotnet", "DockerDotNetApp.dll"]

1. Seleccionar el proyecto web (MVC) clic derecho y “open in terminal”
   1. Escribir en la terminal “docker build -t dotnetcoreapp”. “dotnetcoreapp” va a ser el nombre de la imagen
   2. Escribir en la terminal “docker images”. Esto muestra la imagen creada
   3. Escribir en la terminal “docker run -d -p 5024:5024 –name webapp dotnetcoreapp”. “webapp” es el nombre del contenedor y “dotnetcoreapp” es el nombre de la imagen. Se escribe “5024:5024” para hacer un link entre el puerto del equipo local (host) y el puerto del contenedor de Docker.