

Programa DevOps



Working Time #3

≡ Docker

Es hora de que pongas en práctica todo lo aprendido. 😊

Este apartado tiene el objetivo de ayudarte a seguir potenciando tus habilidades, por lo que a continuación encontrarás diferentes **desafíos** que podrás resolver de forma independiente y a tu ritmo.

¡Manos a la obra!

1. Introducción:

Se creará e implementará una aplicación con Docker, luego se puede actualizar y compartir la aplicación contenerizada.

Los contenedores son entornos virtualizados compactos, como las máquinas virtuales, que proporcionan una plataforma para compilar y ejecutar aplicaciones. Los contenedores no requieren el tamaño y la sobrecarga de un sistema operativo completo. Docker es un proveedor de contenedores estándar del sector y un sistema de administración de contenedores de terceros.

Docker Desktop se ejecuta en el equipo y administra los contenedores locales. Las herramientas de desarrollo como Visual Studio y VS Code ofrecen extensiones que permiten trabajar con un servicio de Docker Desktop local. Puede crear aplicaciones contenerizadas, implementar aplicaciones en contenedores y depurar aplicaciones que se ejecutan en los contenedores.

En este tutorial, se aprenderá a:

- Crear un contenedor.
- Compilar una imagen de contenedor.
- Iniciar un contenedor de aplicaciones.
- Actualizar el código y reemplazar el contenedor.
- Compartir su imagen.
- Ejecutar la imagen en una nueva instancia.

2. Requisitos Previos

- Visual Studio Code.
- La extensión de VS Code de Docker.
- Docker Desktop.
- Una cuenta de Docker Hub. Puede crear una de forma gratuita.
- El tutorial funciona con Windows 10 y versiones posteriores y con Docker Desktop configurado para usar contenedores de Linux.

3. ¿Dónde se lleva a cabo?

Visual Studio Code

4. Tiempo de dedicación

1 hora a 1 hora y media

5. Recursos

<https://github.com/docker/getting-started>

[Docker Hub](#)

[Play with Docker](#)

6. Desafío

Crear un contenedor

Un contenedor es un proceso en el equipo. Está aislado de todos los demás procesos del equipo host. Este aislamiento usa espacios de nombres de kernel y grupos de control.

Un contenedor usa un sistema de archivos aislado. Este sistema de archivos personalizado se proporciona mediante una imagen de contenedor. La imagen contiene todo lo necesario para ejecutar una aplicación, como todas las dependencias, la configuración, los scripts y los archivos binarios. La imagen también contiene otra configuración para el contenedor, como las variables de entorno, un comando predeterminado para ejecutar y otros metadatos.

Después de instalar la extensión de Docker para VS Code, se puede trabajar con contenedores en VS Code. Además de los menús contextuales del panel de Docker, se puede seleccionar Terminal>Nuevo terminal para abrir una ventana de línea de comandos. También se puede ejecutar comandos en una ventana de Bash. A menos que se especifique, cualquier comando etiquetado como Bash puede ejecutarse en una ventana de Bash o en el terminal de VS Code.

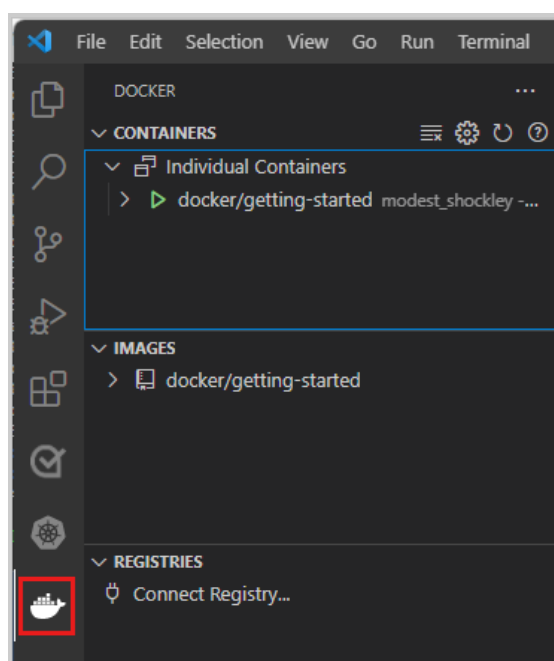
- A. Establecer Docker en el modo de contenedor de Linux. Para cambiar a contenedores de Linux, se deberá hacer clic con el botón derecho en el icono de Docker de la barra de tareas y seleccionar: Cambiar a contenedores de Linux.
- B. En VS Code, seleccionar Terminal>Nuevo terminal.
- C. En la ventana de terminal o en una ventana de Bash, ejecutar este comando.

```
docker run -d -p 80:80 docker/getting-started
```

Este comando contiene los siguientes parámetros:

- **-d**: el contenedor se ejecuta en modo desasociado en segundo plano.
- **-p 80:80**: el puerto 80 del host se asigna al puerto 80 del contenedor.
- **docker/getting-started**: especifica la imagen que se usará.

- D. En VS Code, seleccionar el icono de Docker situado a la izquierda para ver la extensión de Docker.



La extensión de VS Code de Docker muestra los contenedores que se ejecutan en el equipo. Se puede acceder a los registros de contenedor y administrar el ciclo de vida del contenedor, como las acciones de detener y quitar.

El nombre del contenedor (en este ejemplo, modest_schockly) se creó aleatoriamente por lo que en un nuevo intento tendrá un nombre diferente.

- E. Hacer clic con el botón derecho en **docker/getting-started** para abrir un menú contextual. Seleccionar **Abrir en el explorador**.

También se puede abrir un explorador y escribir <http://localhost/tutorial/>; se debe visualizar una página sobre DockerLabs hospedada localmente.

- F. Hacer clic con el botón derecho en docker/getting-started para abrir un menú contextual. Y luego seleccionar Quitar para quitar este contenedor.

Si se quiere quitar un contenedor mediante la línea de comandos, habrá que ejecutar este comando para obtener su identificador de contenedor:

```
docker ps
```

Después, detener y quitar el contenedor:

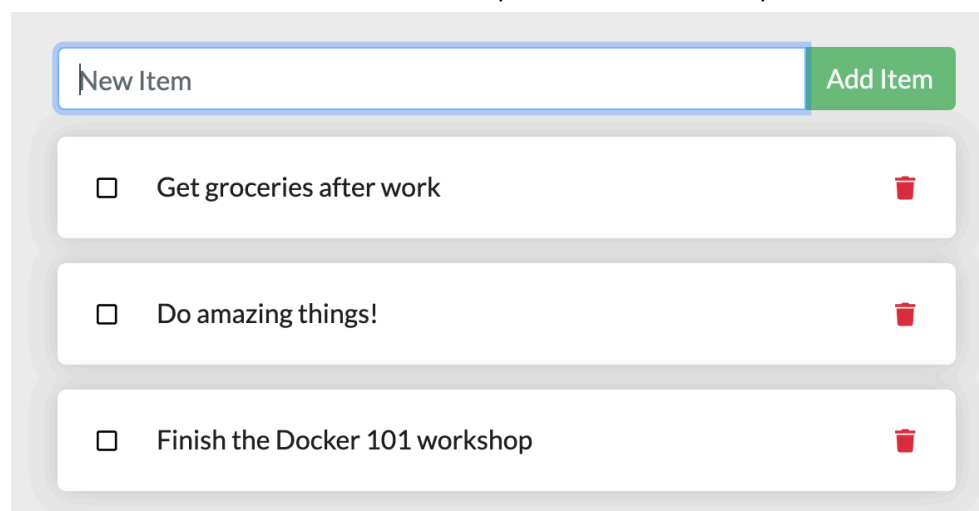
```
docker stop <container-id>
```

```
docker rm <container-id>
```

- G. Actualizar el explorador. La página de introducción que se visualizaba previamente ya habrá desaparecido.

Compilación de una imagen de contenedor para la aplicación

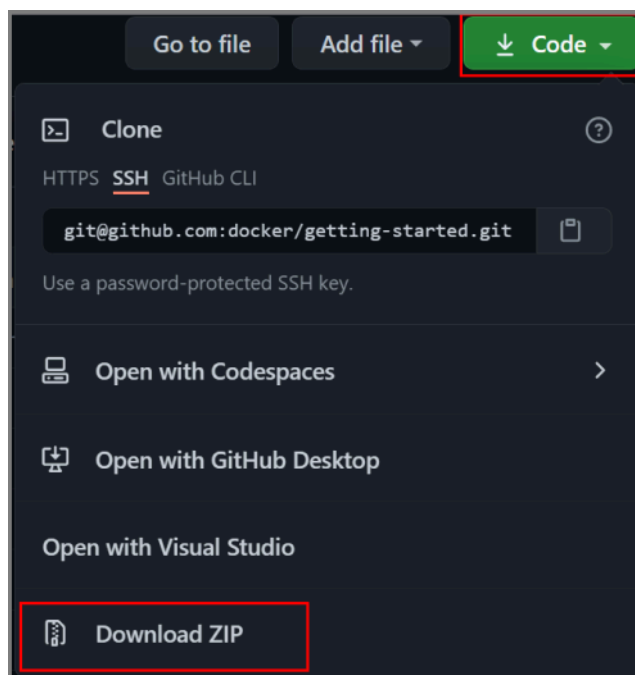
En este tutorial se usa una sencilla aplicación de tareas pendientes.



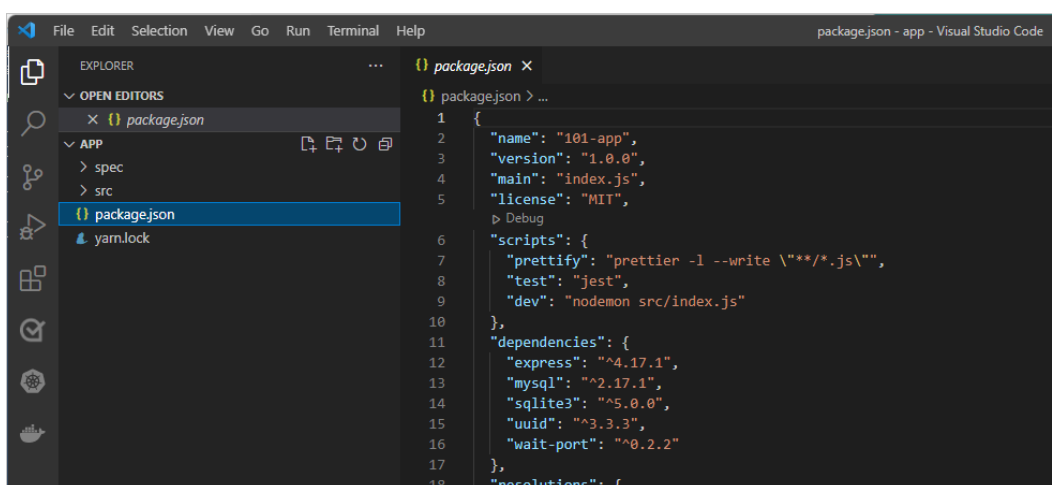
La aplicación permite crear elementos de trabajo y marcarlos como completados o eliminarlos.

Para compilar la aplicación, será necesario crear un **Dockerfile**. Un Dockerfile es un script de instrucciones basado en texto que se usa para crear una imagen de contenedor.

1. Dirigirse al repositorio del [tutorial de introducción de Docker](#) y seleccionar **Código>Descargar archivo ZIP**. Extraiga el contenido en una carpeta local.




2. En VS Code, seleccionar **Archivo>Abrir carpeta**. Ir a la carpeta app del proyecto extraído y abrirla. Debería aparecer un archivo denominado **package.json** y dos carpetas denominadas **src** y **spec**.



3. Crear un archivo denominado **Dockerfile** en la misma carpeta que el archivo **package.json** con el contenido siguiente.

```
FROM node:12-alpine
RUN apk add --no-cache python2 g++ make
WORKDIR /app
COPY . .
RUN yarn install --production
CMD ["node", "/app/src/index.js"]
```


 **Nota:** Asegúrese de que el archivo no tiene ninguna extensión de archivo, como, por ejemplo .txt.

4. En el explorador de archivos, en VS Code a la izquierda, hacer clic con el botón derecho en el Dockerfile y seleccionar Compilar imagen. Escribir getting-started como etiqueta para la imagen en el cuadro de entrada de texto.

La etiqueta es un nombre descriptivo para la imagen.

Para crear una imagen de contenedor desde la línea de comandos, hay que usar el siguiente comando.

```
docker build -t getting-started .
```

 **Nota:** En una ventana externa de Bash, vaya a la carpeta app que tiene el Dockerfile para ejecutar este comando.

En este punto se ha usado el DockerFile para compilar una nueva imagen de contenedor. Es posible observar que se han descargado múltiples "capas". El DockerFile se inicia a partir de la imagen node:12-alpine. A menos que ya esté disponible localmente en el computador, es necesario descargar esta imagen.

Una vez que se ha descargado la imagen, el DockerFile copia la aplicación y usa yarn para instalar las dependencias de la aplicación. El valor CMD del DockerFile especifica el comando predeterminado que se ejecutará al iniciar un contenedor a partir de esta imagen.

El carácter '.' al final del comando Docker build indica a Docker que debe buscar el DockerFile en el directorio actual.

Inicio del Contenedor de Aplicaciones

Ahora con una imagen ya se puede ejecutar la aplicación.

1. Para iniciar el contenedor, usar el comando siguiente.

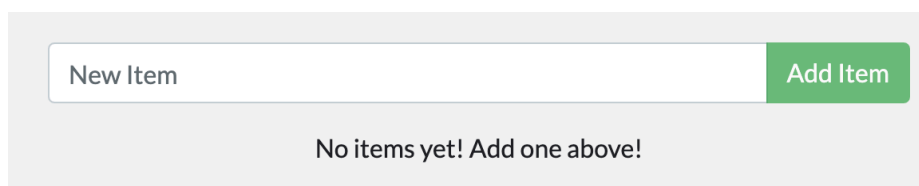
```
docker run -dp 3000:3000 getting-started
```

El parámetro `-d` indica que está ejecutando el contenedor en modo desasociado en segundo plano. El valor `-p` crea una asignación entre el puerto de host 3000 y el puerto de contenedor 3000. Sin la asignación de puertos, no se podría acceder a la aplicación.

2. Después de unos segundos, ir al área de Docker en VS Code y, en **CONTENEDORES**, hacer clic con el botón derecho en **getting-started** y seleccionar **Abrir en el explorador**.

También se puede abrir el explorador web en <http://localhost:3000>.

Debería ser visible la aplicación en ejecución.



3. Agregar uno o dos elementos, y comprobar si funcionan como se esperaba. Se pueden marcar los elementos como completos y quitar elementos. El frontend almacena correctamente los elementos en el backend.

Actualización del Código y reemplazo del Contenedor

En este momento, se tiene un administrador de lista de tareas pendientes en ejecución con algunos elementos. Ahora, se realizarán algunos cambios y se obtendrá información sobre cómo administrar los contenedores.

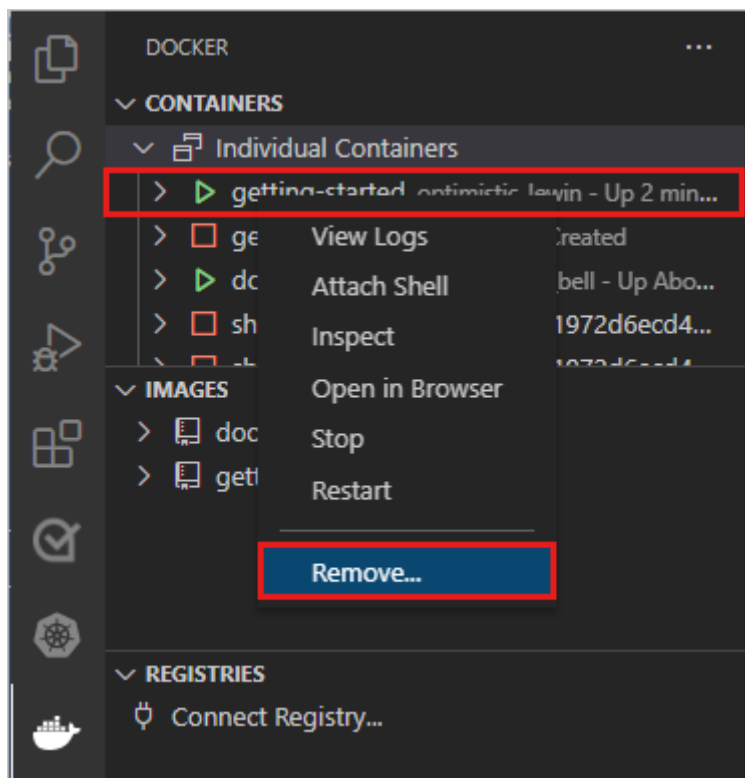
1. En el archivo `src/static/js/app.js`, habrá que actualizar la línea 56 para usar esta nueva etiqueta de texto:

```
- <p className="text-center">No items yet! Add one above!</p>  
+ <p className="text-center">You have no todo items yet! Add one above!</p>
```

Recuerda guardar el cambio.

2. Lo siguiente será detener y quitar la versión actual del contenedor. No se puede usar el mismo puerto para más de un contenedor.

Luego, hacer clic con el botón derecho en el contenedor getting-started y seleccionar Quitar.



Para obtener el identificador del contenedor, también puedes usar el comando siguiente desde la línea de comandos.

```
docker ps
```

Después, tendrás que detener y quitar el contenedor:

```
docker stop <container-id>
```

```
docker rm <container-id>
```

3. Compilar la versión actualizada de la imagen. En el explorador de archivos, haz clic con el botón derecho en Dockerfile y selecciona **Compilar imagen**.

Para compilar en la línea de comandos, también puedes usar el mismo comando de antes.

```
docker build -t getting-started
```

4. Iniciar un contenedor nuevo que use el código actualizado.

`docker run -dp 3000:3000 getting-started`

- Actualizar el explorador en `http://localhost:3000` para ver el texto de ayuda actualizado.

You have no todo items yet! Add one above!

Uso Compartido de la Imagen

Ahora que se ha creado una imagen, se puede **compartir**. Para compartir imágenes de Docker, usa un registro de Docker (Registry). El registro predeterminado es [Docker Hub](#), que es de donde proceden todas las imágenes que se han usado. Para insertar una imagen se debe:

- Primero crear un repositorio en Docker Hub.
- Después dirigirse a [Docker Hub](#) e iniciar sesión en su cuenta.
- Seleccionar Crear repositorio.
- Para el nombre del repositorio, escribir `getting-started`, y asegurarse de que la visibilidad sea pública.
- Finalmente seleccionar Crear.

En el lado derecho de la página, se ve una sección denominada **Docker commands** (Comandos de Docker). En esta sección se proporciona un comando de ejemplo que se puede ejecutar para insertar en este repositorio.

Docker commands

Public View

To push a new tag to this repository,

```
docker push docker/getting-started:tagname
```

6. En la vista de Docker de VS Code, en **IMÁGENES**, hacer clic con el botón derecho en la etiqueta de imagen y seleccionar **Insertar**. Seleccionar **Connect registry** (Conectar registro) y, luego, **Docker Hub**.

Deberá especificar su cuenta de Docker Hub, la contraseña y un espacio de nombres (namespace).

Para insertar en Docker Hub mediante la línea de comandos, sigue el siguiente procedimiento:

Inicia sesión en Docker Hub:

```
docker login -u <username>
```

Usa el comando siguiente para asignar a la imagen getting-started un nuevo nombre.

```
docker tag getting-started<username>/getting-started
```

Usa el comando siguiente para insertar el contenedor.

```
docker push <username>/getting-started
```

Ejecución de la imagen en una nueva instancia

Ahora que la imagen se ha compilado e insertado en un registro, prueba ejecutar la aplicación en una instancia nueva que nunca haya visto esta imagen de contenedor. Para ejecutar la aplicación, use Play with Docker.

1. Abre el explorador en [Play with Docker](#).
2. Inicia sesión con la cuenta de Docker Hub.
3. Selecciona **Inicio** y, luego, el vínculo + **AGREGAR NUEVA INSTANCIA** en la barra de la izquierda. Después de unos segundos, se abrirá una ventana de terminal en el explorador.

03:56:49

CLOSE SESSION

Instances

+ ADD NEW INSTANCE

192.168.0.48
node1

br0lcrti_br0ldviosm4g00b718h0

IP
192.168.0.48 OPEN PORT

Memory
0.82% (32.98MiB / 3.906GiB)

CPU
0.54%

SSH
ssh ip172-18-0-38-br0lcrtim9m000a8ass0@direct.labs.plk

DELETE EDITOR

```
#####
#                               #
#   WARNING!!!!                #
#   This is a sandbox environment. Using personal credentials        #
#   is HIGHLY! discouraged. Any consequences of doing so are         #
#   completely the user's responsiblites.                             #
#   The PWD team.                                                      #
#####
[node1] (local) root@192.168.0.48 ~
$
```

4. En el terminal, iniciar la aplicación.

```
docker run -dp 3000:3000 <username>/getting-started
```

Play with Docker extrae la imagen e inicia.

Seleccionar la notificación 3000, junto a ABRIR PUERTO. Se debe ver la aplicación con las modificaciones. Si no aparece la notificación 3000, seleccionar ABRIR PUERTO y escribir 3000.

