Arquitectura de Computadoras
Configuración Docker para TPE

Introducción **1**
Descargar e instalar Docker **1**
Mac 1
Windows 2
Linux 2
Uso de Docker **3**
Imagen 3
No requerir sudo para comandos docker 4
Útil 4
Compilando el proyecto 4
Instalar QEMU **5**
Desde Ubuntu 5
Desde Mac 5
Extras **5**
Script compilación 5
Script bashrc para dentro del contenedor 6
Agregar permisos a imagen 6
Script compilación: nuevo usuario e imagen con permisos 6

Introducción

Para el desarrollo del TPE, se utiliza Docker en la compilación del proyecto. Este es un tutorial para comenzar a utilizarlo y tener un entorno de trabajo Docker.

Descargar e instalar Docker

El primer paso es instalar Docker. Vamos a ver los distintos casos.

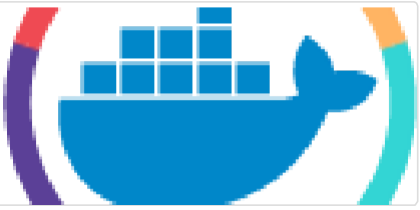
Mac

Para utilizarlo en Mac, se puede descargar desde el siguiente link:

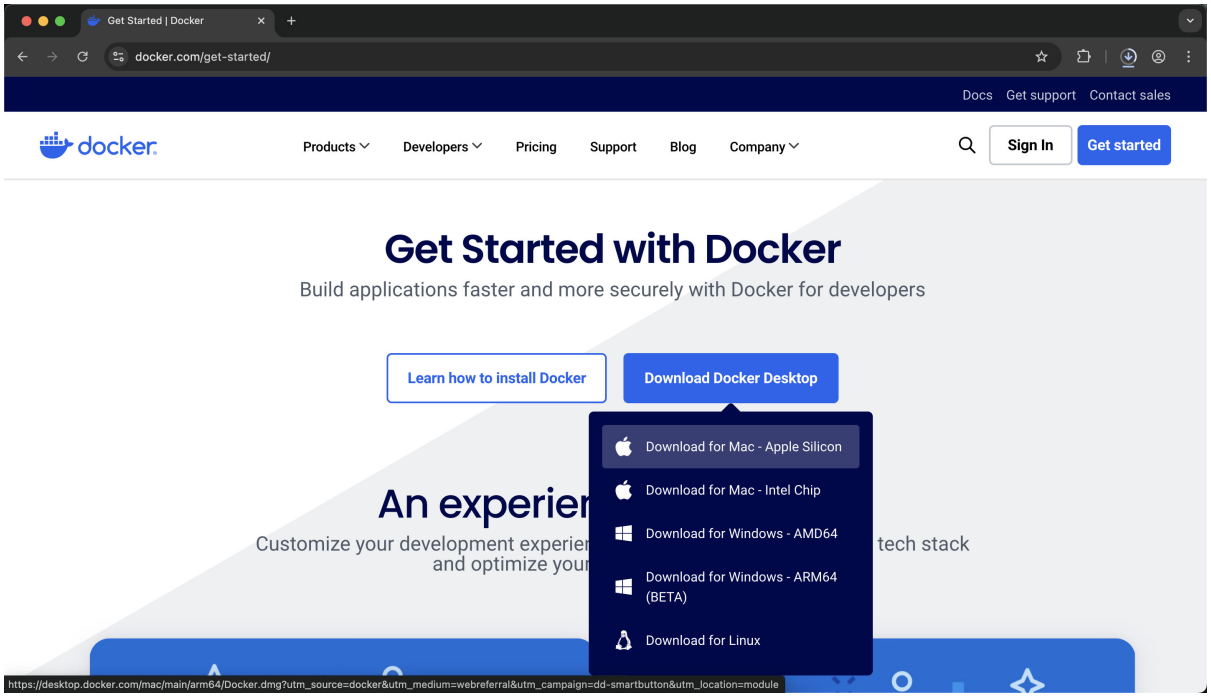
Install Docker Desktop on Mac

Estimated reading time: 7 minutes Docker Desktop terms Commercial use of Docker Desktop in larger enterprises (more than 250 employees OR more than $10 million USD in annual revenue) requires a paid subscription. Welcome to Docker Desktop for

https://docs.docker.com/desktop/install/mac-install/



- Por favor, para una Mac con chip M1/M2, usar la opción de “Apple Silicon”.



Windows

Antes de seguir, verificar que tiene la versión 2 de WSL. Para verificarlo, correr desde PowerShell

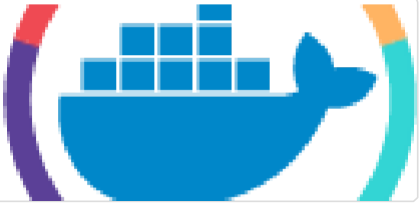
```
wsl -l -v
```

Luego, ir al siguiente link para descargarlo

Install Docker Desktop on Windows

Estimated reading time: 7 minutes Docker Desktop terms Commercial use of Docker Desktop in larger enterprises (more than 250 employees OR more than $10 million USD in annual revenue) requires a paid subscription. Welcome to Docker Desktop for

https://docs.docker.com/desktop/install/windows-install/



Linux

Por lo general se puede instalar directamente desde el manejador de paquetes de la distribución que tengan.

Desde Ubuntu pueden hacer:

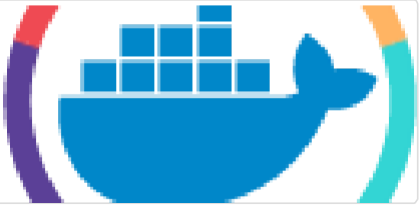
```
sudo apt install docker.io
```

La instalación en Linux puede variar según la distribución. Les dejo un link para consultar, a la izquierda tienen instrucciones particulares para algunas (Debian, Fedora, Ubuntu, Arch).

Install Docker Desktop on Linux

Docker Desktop terms Commercial use of Docker Desktop in larger enterprises (more than 250 employees OR more than $10 million USD in annual revenue) requires a paid subscription. This page contains information about system requirements, download

https://docs.docker.com/desktop/install/linux-install/



Uso de Docker

Imagen

1. Descargar la imagen que vamos a utilizar. Para hacerlo, ir a la terminal y correr:

```
docker pull agodio/itba-so:2.0
```

Esto va a descargar la imagen que se utilizará para los contenedores.

2. Ir al directorio con el TPE. Ese va a ser el directorio asociado con /root en el contenedor, por lo que todos los cambios en ese directorio se van a reflejar en el contenedor y viceversa.
3. Ejecutar el siguiente comando:

```
docker run -d -v ${PWD}:/root --security-opt seccomp:unconfined -it --name <NOMBRE> agodio/itba-so:2.0
```

<NOMBRE> es el nombre que le vamos a dar al contenedor. La idea es reutilizarlo en el desarrollo. Sin embargo, si se desea crear un contenedor cada vez que se quiere utilizar, es necesario correr:

```
docker run --rm -v ${PWD}:/root --security-opt seccomp:unconfined -it agodio/itba-so:2.0
```

La version anterior elimina al contenedor cuando salimos de él.

- El flag **--security-opt seccomp:unconfined** quita la restricción a syscalls utilizadas por gdb, strace, ltrace, PVS-studio, etc.
- El flag **-v ${PWD}:/root** hace un bind-mount de la carpeta actual ($PWD) del host en la carpeta /root del guest, esto permite compartir archivos entre el host y el guest.
- **Cuidado!** Los archivos **no se copian**, sino que se **comparten**, es decir que cualquier cambio tanto desde el host como desde el guest se podrá ver en el host y el guest.

Si al intentar ejecutar "docker run" se obtiene el siguiente error:

```
docker: Error response from daemon: failed to start shim: exec: "docker-containerd-shim": executable file not found in $PATH: unknown.
```

Se puede resolver reiniciando el servicio de Docker:

```
service docker restart
```

Para más explicación de las opciones utilizadas, visitar el link:

https://docs.docker.com/engine/reference/run/

No requerir sudo para comandos docker

El siguiente seteo (opcional) permite evitar tener que correr como sudo los comandos a Docker. Crear el grupo docker (en algunas distribuciones Linux se crea con la instalación, en tal caso, simplemente les notificará de ello):

```
sudo groupadd docker
```

Agregar mi usuario al grupo de docker:

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

Posiblemente requieran reiniciar la computadora para que el cambio se haga efectivo.

Mas información en: https://docs.docker.com/engine/install/linux-postinstall/

Útil

Un comando que puede resultar útil, es el de listar todos los contenedores en su computadora (contenedores activos e inactivos):

```
docker container list –a
```

Información básica acerca de la configuración de Docker:

```
docker info
```

Compilando el proyecto

Luego, debemos entrar en el entorno del contenedor.

Aunque no es necesario hacerlo justo luego de crearlo, por lo general primero tenemos que empezar al contenedor (si le dimos un nombre) corriendo:

```
docker start <NOMBRE>
```

Luego, para entrar a la terminal del contenedor, debemos correr:

```
docker exec -it <NOMBRE> bash
```

Esto debería mostrar un prompt similar a:

```
root@c3285f863835:/#
```

Luego, ir al proyecto:

```
cd /root
```

Finalmente, ahí podemos correr lo necesario para la compilación (la imagen de Docker ya viene con todo lo necesario para compilar el proyecto).

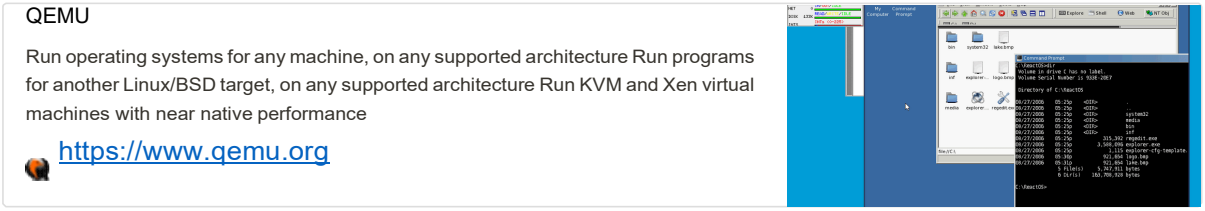
```
cd Toolchain
make all
cd ..
make all
```

La idea es usar Docker **para compilar**, y luego ejecutar el proyecto desde su entorno.

No se debe ejecutar run.sh desde el contenedor. Se debe ejecutar desde el entorno local habiendo instalado antes a QEMU (desde el contenedor van a tener el error de que no encuentra un dispositivo de video).

Instalar QEMU

Para poder ejecutar el proyecto, deben tener instalado a QEMU localmente. Dejo un link a la página:



Desde Ubuntu

Pueden instalar QEMU con:

```
sudo apt install qemu-system-x86 qemu-utils
```

Desde Mac

Pueden instalar QEMU con:

```
brew install qemu
```

Finalmente, ya teniendo QEMU instalado localmente y habiendo compilado el proyecto, deben ubicarse en el directorio local y correr:

```
./run.sh
```

Esto abre una ventana donde van a poder interactuar con el proyecto.

Extras

Script compilación

Se puede hacer un script para que haga la compilación sin que tengamos que entrar manualmente al contenedor en cada intento, por ejemplo:

```
docker start <NOMBRE>
docker exec -it <NOMBRE> make clean -C /root/Toolchain
docker exec -it <NOMBRE> make clean -C /root/
docker exec -it <NOMBRE> make -C /root/Toolchain
docker exec -it <NOMBRE> make -C /root/
docker stop <NOMBRE>
```

Script bashrc para dentro del contenedor

Si queremos evitar hacer cd /root cada vez que entramos o agregarle colores a la salida de la terminal de Docker, podemos crear un archivo .bashrc en el directorio /root del contenedor o en el directorio donde se creó el mismo, y agregar:

```
export GCC_COLORS="error=01;31:warning=01;35:note=01;36:caret=01;32:locus=
01:quote=01"
export LS_COLORS="rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:
bd=40;33;01:cd=40;33;01:or=40;31;01"
alias ls="ls --color=auto"
cd /root
```

Agregar permisos a imagen

Sin tener los permisos en la imagen, no se va a poder hacer **./run.sh**

Revisar que en la pc tengamos el $UID = 1000. Muestro también el valor de <MI_USER>.

```
echo $UID
echo $USER
```

Primero ingresar al contenedor:

```
docker start <NOMBRE>
docker exec -it <NOMBRE> bash
```

Los anteriores comandos hicieron que cambie el prompt, porque se ingresó al bash del contenedor <NOMBRE>.

Ahora, dentro del contenedor:

```
groupadd -g 1000 -o <MI_USER>
useradd -m -u 1000 -g 1000 -o -s /bin/bash <MI_USER>
```

Script compilación: nuevo usuario e imagen con permisos

Ahora, vamos a usar el contenedor **no** con el usuario **root** (utilizado por defecto), sino que con el usuario que acabamos de crear:

```
docker exec -u <MI_USER> -it <NOMBRE> /bin/bash
```

Hacer los 2 clean y 2 make

Si hacemos un script

```
docker start <NOMBRE>
docker exec –u <MI_USER> -it <NOMBRE> make clean –C /root/Toolchain
docker exec –u <MI_USER> -it <NOMBRE> make clean –C /root/
docker exec –u <NUESTRO_USER> -it <NOMBRE> make –C /root/Toolchain
docker exec –u <MI_USER> -it <NOMBRE> make –C /root/
docker stop <NOMBRE>
```