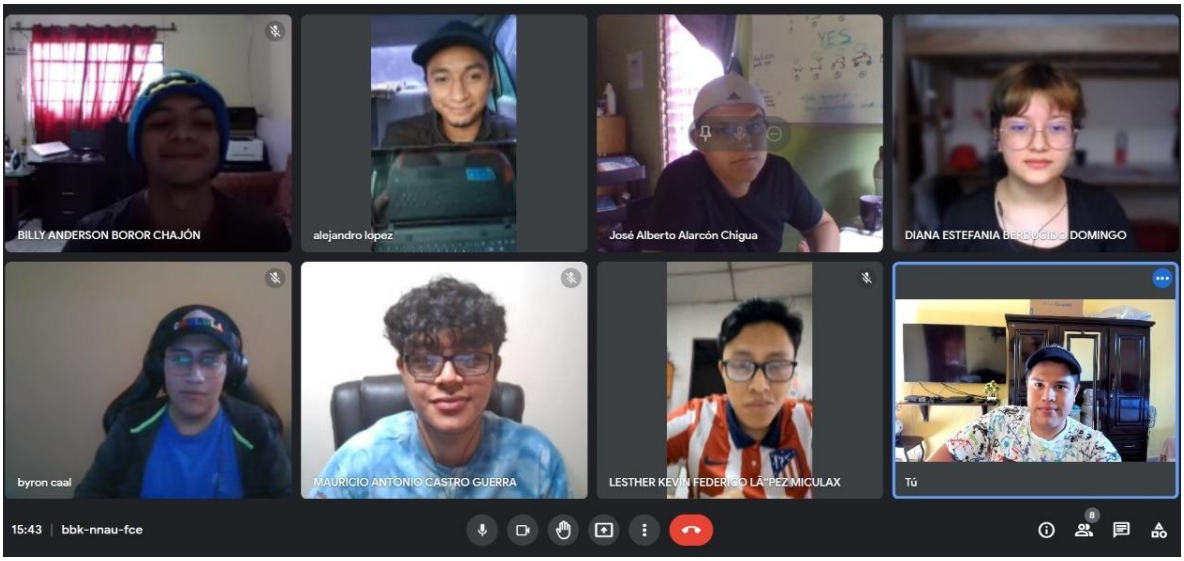
Universidad de San Carlos de Guatemala
Facultad de Ingeniería.
Escuela de Ciencias y Sistemas
Ing. Herman Veliz
Curso de Prácticas Iniciales, Sección F-
Tutores: Byron Caal, José Alarcón

Como instalar Docker, crear Dockerfile y Dockerhub

Diana Estefanía Berducido Domingo 202000277
Mauricio Antonio Castro Guerra 202100299
Kevin Orlando Cámbara García 202001851
Omar Alejandro Lopez Cifuentes 201709469
Lesther Kevin Federico López Miculax 202110897
Billy Anderson Boror Chajon 201901385

Foto con todos los integrantes del grupo y los tutores



Cómo instalar Docker en Ubuntu 18.04

Docker no forma parte de los repositorios oficiales de Ubuntu 18.04. Sin embargo, el proceso de instalación no se ve afectado por esto. Empecemos.

1. Accede a tu VPS

Primero, tienes que conectarte al servidor usando SSH.

2. Actualiza tu sistema

Luego, el sistema debe actualizarse para que sea más seguro y confiable para la instalación de Docker. Ejecuta los siguientes dos comandos:

```
sudo apt update
sudo apt upgrade
```

3. Instala el paquete de requisitos previos

Una vez que hayas actualizado el sistema, necesitarás instalar algunos paquetes necesarios antes de instalar Docker. Puedes hacerlo con la ayuda de un solo comando:

```
sudo apt-get install  curl apt-transport-https ca-certificates software-properties-common
```

Para comprender mejor el comando anterior, aquí hay una breve descripción de lo que significa:

- **apt-transport-https:** permite que el administrador de paquetes transfiera datos a través de https
- **ca-certificates:** permite que el navegador web y el sistema verifiquen los certificados de seguridad
- **curl:** transfiere datos
- **software-properties-common:** agrega scripts para administrar el software

4. Agrega los repositorios de Docker

Ahora tienes que agregar los repositorios de Docker. Esto facilitará mucho el proceso de instalación y al mismo tiempo podrás utilizar el método de instalación oficialmente compatible.
Primero, agrega la clave GPG, ingresando el siguiente comando en la línea de comando:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

A continuación, agrega el repositorio:

```
sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
```

Después de eso, actualiza la información del repositorio:

```
sudo apt update
```

Puedes asegurarte de estar instalando desde el repositorio de Docker en lugar del repositorio predeterminado de Ubuntu con este comando:

```
apt-cache policy docker-ce
```

Una salida correcta se verá como la siguiente con diferentes números de versión:

```
docker-ce:
  Installed: (none)
  Candidate: 16.04.1~ce~4-0~ubuntu
  Version table:
     16.04.1~ce~4-0~ubuntu 500
        500 https://download.docker.com/linux/ubuntubionic/stableamd64packages
```

Como puedes ver, **docker-ce** no está instalado, por lo que podemos pasar al siguiente paso.

5. Instala Docker en Ubuntu 18.04

Ya casi terminas. Usa el comando apt para instalar Docker:
sudo apt install docker-ce

6. Comprueba el estado de Docker

Una vez que se completa la instalación, es buena idea verificar el estado del servicio:

sudo systemctl status docker

Cómo comenzar a usar Docker en Ubuntu 18.04

Una vez que Docker está instalado, todo lo que tienes que hacer es usar la imagen de prueba para verificar que todo funcione como debería. Puedes hacer esto con el siguiente comando:
sudo docker run hello-world

Ahora, si quieres buscar imágenes disponibles, solo tienes que usar el siguiente comando:
sudo docker search [search_query]

Simplemente reemplaza su consulta con el texto entre corchetes.

Por ejemplo, si quieres buscar una imagen relacionada con Debian, el comando y la salida se verán así:
sudo docker search debian

Luego, para descargar la imagen a tu computadora, utiliza el nombre de la

```
$ docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/


$ docker images hello-world
REPOSITORY   TAG      IMAGE ID      SIZE
hello-world  latest   bf756fb1ae65  13336
```

imagen junto con el siguiente comando:
sudo docker pull [nombre_imagen]

En la práctica, el comando se vería así:
sudo docker pull debian

Normalmente, los usuarios tienen varias imágenes en su sistema. Puedes listarlas con el comando:
sudo docker images

La lista se parecerá mucho a la que se muestra cuando ingresas una consulta de búsqueda.

Después de eso, puedes ejecutar una imagen usando el comando de extracción y la ID de la imagen.
sudo docker run -i -t [imagen]

Hay opciones que amplían la funcionalidad del comando en sí. Por ejemplo, la opción **–i** hace que la ejecución de la imagen sea interactiva. O la opción **–d**, que ejecuta Docker en segundo plano.

Una vez que ejecutes una imagen, puedes finalizar la ejecución utilizando la combinación de teclas **CTRL+D**.

Finalmente, si quieres usar Docker sin privilegios de root, debes ejecutar el siguiente comando:
sudo usermod -aG docker $(whoami)

Después de esto, reinicia el sistema y se aplicarán los cambios.

Usando el comando Docker

Para ver todos los subcomandos disponibles, utiliza el siguiente comando:
docker

Para ver las opciones disponibles con un comando:
docker docker-subcommand --help

Aquí están los subcomandos disponibles de Docker 18′:
docker attach - Adjunta flujos de entrada, salida y error estándar locales a un contenedor en ejecución
docker build - Crea una imagen a partir de un Dockerfile
docker builder - Administra compilaciones
docker checkpoint - Administra puntos de control
docker commit - Crea una nueva imagen a partir de los cambios de un contenedor
docker config - Administra configuraciones de Docker
docker container - Administra contenedores
docker context - Administra contextos
docker cp - Copia archivos/carpetas entre un contenedor y el sistema de archivos local
docker create - Crea un nuevo contenedor
docker diff - Inspecciona cambios en archivos o directorios en el sistema de archivos de un contenedor
docker events - Obtener eventos en tiempo real del servidor
docker exec - Ejecuta un comando en un contenedor en ejecución
docker export - Exporta el sistema de archivos de un contenedor como un archivo tar
docker history - Muestra el historial de una imagen
docker image- Administra imágenes
docker images - Lista imágenes
docker import - Importa el contenido de un tarball para crear una imagen del sistema de archivos
docker info - Muestra información de todo el sistema
docker inspect - Devuelve información de bajo nivel sobre objetos Docker

docker kill - Elimina uno o más contenedores en ejecución
docker load - Carga una imagen desde un archivo tar o STDIN
docker login - Inicia sesión en un registro de Docker
docker logout - Cierra la sesión de un registro de Docker
docker logs - Recupera los registros de un contenedor.
docker manifest - Administra manifiestos de imágenes de Docker y listas de manifiestos
docker network - Administra redes
docker node - Administra nodos Swarm
docker pause - Pausa todos los procesos dentro de uno o más contenedores
docker plugin - Administra complementos
docker port - Lista las asignaciones de puertos o una asignación específica para el contenedor
docker ps - Lista los contenedores
docker pull - Extrae una imagen o un repositorio de un registro
docker push - Envía una imagen o un repositorio a un registro
docker rename - Cambia el nombre de un contenedor
docker restart - Reinicia uno o más contenedores
docker rm - Elimina uno o más contenedores
docker rmi - Elimina una o más imágenes
docker run - Ejecuta un comando en un nuevo contenedor
docker save - Guarda una o más imágenes en un archivo tar (transmitido a STDOUT de forma predeterminada)
docker search - Busca imágenes en Docker Hub
docker secret - Administra los secretos de Docker
docker service - Administra servicios
docker stack - Administra stacks
docker start - Inicia uno o más contenedores detenidos
docker stats - Muestra una transmisión en vivo de las estadísticas de uso de recursos de los contenedores
docker stop - Detiene uno o más contenedores en ejecución
docker swarm - Administra swarm
docker system - Administra Docker
docker tag - Crea una etiqueta TARGET_IMAGE que haga referencia a SOURCE_IMAGE
docker top - Muestra los procesos en ejecución de un contenedor
docker trust - Administra la confianza en las imágenes de Docker
docker unpause - Reanuda todos los procesos dentro de uno o más contenedores
docker update - Actualiza la configuración de uno o más contenedores
docker version - Muestra la información de la versión de Docker
docker volume - Administra volúmenes
docker wait - Bloquea hasta que uno o más contenedores se detengan, luego imprima sus códigos de salida

Imágenes a medida con Dockerfile

Docker puede construir imágenes automáticamente, leyendo las instrucciones indicadas en un fichero Dockerfile. Se trata de un documento de texto que contiene todas las órdenes a las que un usuario dado puede llamar, desde la línea de comandos, para crear una imagen.

Los pasos principales para crear una imagen a partir de un fichero Dockerfile son:

- Crear un nuevo directorio que contenga el fichero, con el guión y otros ficheros que fuesen necesarios para crear la imagen.
- Crear el contenido.
- Construir la imagen mediante el comando docker build.

La sintaxis para el comando es:

```
docker build [opciones] RUTA | URL | -
```

Las opciones más comunes son:

- **-t, nombre [:etiqueta].** Crea una imagen con el nombre y la etiqueta especificada a partir de las instrucciones indicadas en el fichero. Es una opción muy recomendable.
- **–no-cache.** Por defecto, Docker guarda en memoria caché las acciones realizadas recientemente. Si se diese el caso de que ejecutamos un docker build varias veces, Docker comprobará si el fichero contiene las mismas instrucciones y, en caso afirmativo, no generará una nueva imagen. Para generar una nueva imagen omitiendo la memoria caché utilizaremos siempre esta opción.
- **–pull.** También por defecto. Docker solo descargará la imagen especificada en la expresión FROM si no existe. Para forzar que descargue la nueva versión de la imagen utilizaremos esta opción.
- **–quiet.** Por defecto, se muestra todo el proceso de creación, los comandos ejecutados y su salida. Utilizando esta opción solo mostrará el identificador de la imagen creada.

Laboratorio de pruebas
La mejor forma de aprender algo es dándole caña, así que vamos al lío. Seguiremos los pasos expuestos en los párrafos anteriores.

```
mkdir ubuntutest

cd ubuntutest

vi Dockerfile
```

Indicamos en el guión qué imagen base vamos a utilizar mediante **FROM**, después con **RUN** apuntamos los comandos a ejecutar y con **CMD** decimos el comando por defecto:

```
FROM ubuntu:latest



RUN apt-get -y update; \

    apt-get -y upgrade; \

    apt-get -y install apt-utils \

    vim \

    htop;

RUN apt-get -y install dstat



CMD ["bash"]
```

Una vez creado el fichero y editado, lo guardamos. Ahora realizamos la construcción de la imagen:

```
docker build  -t "pruebas9:dockerfile" .
```

Con el resultado:

```
[....]

Step 6/6 : CMD bash
```

```
 ---> Running in 75324aa704ba

 ---> b2dcc8d2d69d

Removing intermediate container 75324aa704ba

Successfully built b2dcc8d2d69d

Successfully tagged pruebas9:dockerfile
```

Lo expuesto anteriormente sólo es la parte final de los registros de creación de la imagen. Ahora ya la podemos ver en la lista de imágenes disponibles:

```
[root@servdocker /]# docker images

REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE

pruebas9            dockerfile          b2dcc8d2d69d        7 minutes ago       254MB
```

Pero esto no para aquí. Vamos a crear un contenedor a partir de la imagen:

```
[root@servdocker /]# docker run -dti --name pruebacontainer b2dcc8d2d69d

cd1fd50239d9c17f91ab1aa739e9230eaf9a17dd273305c5fda9a8a48df194f3

[root@servdocker /]# docker ps

CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS               NAMES

cd1fd50239d9        b2dcc8d2d69d        "bash"              33 seconds ago      Up 32
seconds                                 pruebacontainer
```

Y accedemos a él para comprobar que, efectivamente, los programas instalados están disponibles:

```
[root@servdocker /]# docker exec -i -t pruebacontainer /bin/bash

root@cd1fd50239d9:/# dstat

You did not select any stats, using -cdngy by default.

----total-cpu-usage---- -dsk/total- -net/total- ---paging-- ---system--
```

```
usr sys idl wai hiq siq| read  writ| recv  send|  in   out | int   csw
 2   1  94   3   0   0|  56k  338k|   0     0 |  68B  159B| 152   165
 0   0 100   0   0   0|   0     0 |   0     0 |   0     0 |  38    70
 0   0 100   0   0   0|   0     0 |   0     0 |   0     0 |  79    94
```

Ya podéis ver que el comando '***dstat***' se ha instalado sin problema y funciona correctamente; lo mismo podemos decir con el resto de programas indicados en el guión del Dockerfile.

¿Cómo crear un repositorio en DockerHub?

- En el menú principal dirígete a Repositories > Create Repository +
- Escribe el nombre de tu repositorio y la descripción.
- Elige Public para hacer tu repositorio público.
- Da clic en el botón Create.

DockerHub te permite crear una serie de repositorios públicos sin ningún costo. También te ofrece la posibilidad de tener hasta 1 repositorio privado. Si se desean crear mas repositorios privados esto tiene un costo, por lo que hay que contratar un plan acorde al número de repositorios privados que se vayan a manejar.