# Cómo instalar y usar Docker en Ubuntu 20.04

Docker es una aplicación que simplifica el proceso de administración de procesos de aplicación en contenedores. Los contenedores le permiten ejecutar sus aplicaciones en procesos con aislamiento de recursos. Son similares a las máquinas virtuales, pero los contenedores son más portátiles, más flexibles con los recursos y más dependientes del sistema operativo host.

En este tutorial, instalará y usará Docker Community Edition (CE) en Ubuntu 20.04. Instalará Docker, trabajará con contenedores e imágenes e introducirá una imagen en un repositorio de Docker.

# Paso 1: Instalar Docker

Es posible que la versión del paquete de instalación de Docker disponible en el repositorio oficial de Ubuntu no sea la más reciente. Para asegurarnos de contar con la versión más reciente, instalaremos Docker desde el repositorio oficial de Docker. Para hacerlo, agregaremos una nueva fuente de paquetes y la clave GPG de Docker para garantizar que las descargas sean válidas, y luego instalaremos el paquete.

Primero, actualice su lista de paquetes existente:

***$sudo apt update***

A continuación, instale algunos paquetes de requisitos previos que permitan a apt usar paquetes a través de HTTPS:

***$sudo apt install apt-transport-https ca-certificates curl software-properties-common***

Luego, añada la clave de GPG para el repositorio oficial de Docker en su sistema:

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

Agregue el repositorio de Docker a las fuentes de APT:

***$sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable"***

A continuación, actualice el paquete de base de datos con los paquetes de Docker del repositorio recién agregado:

***$sudo apt update***

Asegúrese de estar a punto de realizar la instalación desde el repositorio de Docker en lugar del repositorio predeterminado de Ubuntu:

***$apt-cache policy docker-ce***

Si bien el número de versión de Docker puede ser distinto, verá un resultado como el siguiente:

Output of apt-cache policy docker-ce

docker-ce:

Installed: (none)

Candidate: 5:19.03.9~3-0~ubuntu-focal

Version table:

5:19.03.9~3-0~ubuntu-focal 500

500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages

Observe que docker-ce no está instalado, pero la opción más viable para la instalación es del repositorio de Docker para Ubuntu 20.04 (focal).

**Por último, instale Docker:**

***$sudo apt install docker-ce***

Con esto, Docker quedará instalado, el demonio se iniciará y el proceso se habilitará para ejecutarse en el inicio. Compruebe que funcione:

***$sudo systemctl status docker***

El resultado debe ser similar al siguiente, y mostrar que el servicio está activo y en ejecución:

Output

● docker.service - Docker Application Container Engine

Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)

Active: active (running) since Tue 2020-05-19 17:00:41 UTC; 17s ago

TriggeredBy: ● docker.socket

Docs: https://docs.docker.com

Main PID: 24321 (dockerd)

Tasks: 8

Memory: 46.4M

CGroup: /system.slice/docker.service

└─24321 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

La instalación de Docker ahora le proporcionará no solo el servicio de Docker (demonio) sino también la utilidad de línea de comandos docker o el cliente de Docker. Más adelante, exploraremos la forma de usar el comando docker en este tutorial.

# Paso 2 : Ejecutar el comando Docker sin sudo (opcional)

Por defecto, el comando docker solo puede ser ejecutado por el usuario root o un usuario del grupo docker, que se crea automáticamente durante el proceso de instalación de Docker. Si intenta ejecutar el comando docker sin sudo como prefijo o sin formar parte del grupo docker, obtendrá un resultado como este:

Output

docker: Cannot connect to the Docker daemon. Is the docker daemon running on this host?.

See 'docker run --help'.

Si desea evitar escribir sudo al ejecutar el comando docker, agregue su nombre de usuario (alumno) al grupo docker:

$sudo usermod -aG docker alumno

Para aplicar la nueva membresía de grupo, cierre la sesión del servidor y vuelva a iniciarla o escriba lo siguiente:

$su - alumno

Para continuar, se le solicitará ingresar la contraseña de su usuario.

Confirme que ahora su usuario se agregó al grupo docker escribiendo lo siguiente:

$groups alumno

Output

alumno sudo docker

Si debe agregar al grupo docker un usuario con el que no inició sesión, declare dicho nombre de usuario de forma explícita usando lo siguiente:

sudo usermod -aG docker username

Para el resto de este artículo, se supone que ejecutará el comando docker como usuario del grupo docker. Si elige no hacerlo, incluya sudo al principio de los comandos.

A continuación, exploremos el comando docker.

# Paso 3: Usar el comando docker

El uso de docker consiste en pasar a este una cadena de opciones y comandos seguida de argumentos. La sintaxis adopta esta forma:

docker [option] [command] [arguments]

Para ver todos los subcomandos disponibles, escriba lo siguiente:

docker

A partir de Docker 19, en la lista completa de subcomandos disponibles se incluye lo siguiente:

Output

|  |
| --- |
| attach Attach local standard input, output, and error streams to a running container |
| build Build an image from a Dockerfile |
| commit Create a new image from a container's changes |
| cp Copy files/folders between a container and the local filesystem |
| create Create a new container |
| diff Inspect changes to files or directories on a container's filesystem |
| events Get real time events from the server |
| exec Run a command in a running container |
| export Export a container's filesystem as a tar archive |
| history Show the history of an image |
| images List images |
| import Import the contents from a tarball to create a filesystem image |
| info Display system-wide information |
| inspect Return low-level information on Docker objects |
| kill Kill one or more running containers |
| load Load an image from a tar archive or STDIN |
| login Log in to a Docker registry |
| logout Log out from a Docker registry |
| logs Fetch the logs of a container |
| pause Pause all processes within one or more containers |
| port List port mappings or a specific mapping for the container |
| ps List containers |
| pull Pull an image or a repository from a registry |
| push Push an image or a repository to a registry |
| rename Rename a container |
| restart Restart one or more containers |
| rm Remove one or more containers |
| rmi Remove one or more images |
| run Run a command in a new container |
| save Save one or more images to a tar archive (streamed to STDOUT by default) |
| search Search the Docker Hub for images |
| start Start one or more stopped containers |
| stats Display a live stream of container(s) resource usage statistics |
| stop Stop one or more running containers |
| tag Create a tag TARGET\_IMAGE that refers to SOURCE\_IMAGE |
| top Display the running processes of a container |
| unpause Unpause all processes within one or more containers |
| update Update configuration of one or more containers |
| version Show the Docker version information |
| wait Block until one or more containers stop, then print their exit codes |

Si desea ver las opciones disponibles para un comando específico, escriba lo siguiente:

docker docker-subcommand --help

Para ver información sobre Docker relacionada con todo el sistema, utilice lo siguiente:

docker info

Exploremos algunos de estos comandos. Comenzaremos trabajando con imágenes.

# Paso 4: Trabajar con imágenes de Docker

Los contenedores de Docker se construyen con imágenes de Docker. Por defecto, Docker obtiene estas imágenes de Docker Hub, un registro de Docker gestionado por Docker, la empresa responsable del proyecto Docker. Cualquiera puede alojar sus imágenes en Docker Hub, de modo que la mayoría de las aplicaciones y las distribuciones de Linux que necesitará tendrán imágenes alojadas allí.

Para verificar si puede acceder a imágenes y descargarlas de Docker Hub, escriba lo siguiente:

***$docker run hello-world***

El resultado indicará que Docker funciona de forma correcta:

Output

Unable to find image 'hello-world:latest' locally

latest: Pulling from library/hello-world

0e03bdcc26d7: Pull complete

Digest: sha256:6a65f928fb91fcfbc963f7aa6d57c8eeb426ad9a20c7ee045538ef34847f44f1

Status: Downloaded newer image for hello-world:latest

Hello from Docker!

This message shows that your installation appears to be working correctly.

...

Inicialmente, Docker no pudo encontrar la imagen de hello-world a nivel local. Por ello la descargó de Docker Hub, el repositorio predeterminado. Una vez que se descargó la imagen, Docker creó un contenedor a partir de ella y de la aplicación dentro del contenedor ejecutado, y mostró el mensaje.

Puede buscar imágenes disponibles en Docker Hub usando el comando docker con el subcomando search. Por ejemplo, para buscar la imagen de Ubuntu, escriba lo siguiente:

docker search ubuntu

La secuencia de comandos rastreará Docker Hub y mostrará una lista de todas las imágenes cuyo nombre coincida con la cadena de búsqueda. En este caso, el resultado será similar a lo siguiente:

Output

NAME DESCRIPTION STARS OFFICIAL AUTOMATED

ubuntu Ubuntu is a Debian-based Linux operating sys… 10908 [OK]

dorowu/ubuntu-desktop-lxde-vnc Docker image to provide HTML5 VNC interface … 428 [OK]

rastasheep/ubuntu-sshd Dockerized SSH service, built on top of offi… 244 [OK]

consol/ubuntu-xfce-vnc Ubuntu container with "headless" VNC session… 218 [OK]

ubuntu-upstart Upstart is an event-based replacement for th… 108 [OK]

ansible/ubuntu14.04-ansible Ubuntu 14.04 LTS with

...

En la columna de OFFICIAL, OK indica una imagen creada y avalada por la empresa responsable del proyecto. Una vez que identifique la imagen que desearía usar, puede descargarla a su computadora usando el subcomando pull.

Ejecute el siguiente comando para descargar la imagen oficial de ubuntu a su ordenador:

docker pull ubuntu

Verá el siguiente resultado:

Output

Using default tag: latest

latest: Pulling from library/ubuntu

d51af753c3d3: Pull complete

fc878cd0a91c: Pull complete

6154df8ff988: Pull complete

fee5db0ff82f: Pull complete

Digest: sha256:747d2dbbaaee995098c9792d99bd333c6783ce56150d1b11e333bbceed5c54d7

Status: Downloaded newer image for ubuntu:latest

docker.io/library/ubuntu:latest

Una vez descargada una imagen, puede ejecutar un contenedor usando la imagen descargada con el subcomando run. Como pudo ver en el ejemplo de hello-world, si no se descargó una imagen al ejecutarse docker con el subcomando run, el cliente de Docker descargará primero la imagen y luego ejecutará un contenedor utilizándola.

Para ver las imágenes descargadas a su computadora, escriba lo siguiente:

***$docker images***

El resultado debe tener un aspecto similar al siguiente:

Output

REPOSITORY TAG IMAGE ID CREATED SIZE

ubuntu latest 1d622ef86b13 3 weeks ago 73.9MB

hello-world latest bf756fb1ae65 4 months ago 13.3kB

Como verá más adelante en este tutorial, las imágenes que utilice para ejecutar contenedores pueden modificarse y utilizarse para generar nuevas imágenes que se pueden cargar posteriormente (“introducir” es el término técnico) a Docker Hub o a otros registros de Docker.

Veamos en mayor profundidad la forma de ejecutar los contenedores.

# Paso 5: Ejecutar un contenedor de Docker

El contenedor hello-world que ejecutó en el paso anterior es un ejemplo de un contenedor que se ejecuta y se cierra tras emitir un mensaje de prueba. Los contenedores pueden ofrecer una utilidad mucho mayor y ser interactivos. Después de todo, son similares a las máquinas virtuales, aunque más flexibles con los recursos.

Como ejemplo, ejecutemos un contenedor usando la imagen más reciente de Ubuntu. La combinación de los conmutadores -i y -t le proporcionan un acceso interactivo del shell al contenedor:

***$docker run -it ubuntu***

Su símbolo del sistema debe cambiar para reflejar el hecho de que ahora trabaja dentro del contenedor y debe adoptar esta forma:

Output

root@d9b100f2f636:/#

Tenga en cuenta el ID del contenedor en el símbolo del sistema. En este ejemplo, es d9b100f2f636. Más adelante, necesitará ese ID de contenedor para identificar el contenedor cuando desee eliminarlo.

Ahora puede ejecutar cualquier comando dentro del contenedor. Por ejemplo, actualicemos la base de datos del paquete dentro del contenedor. No es necesario prefijar ningún comando con sudo, ya que realiza operaciones dentro del contenedor como el usuario root:

***$Apt update***

Luego, instale cualquier aplicación en él. Probemos con Node.js:

***$apt install nodejs***

Con esto, se instala Node.js en el contenedor desde el repositorio oficial de Ubuntu. Cuando finalice la instalación, verifique que Node.js esté instalado:

node -v

Verá el número de versión en su terminal:

Output

v10.19.0

Cualquier cambio que realice dentro del contenedor solo se aplica a este.

Para cerrar el contenedor, escriba exit a línea de comandos.

A continuación, veremos la forma de administrar los contenedores en nuestro sistema.

# Paso 6: Administrar contenedores de Docker

Después de usar Docker durante un tiempo, tendrá muchos contenedores activos (en ejecución) e inactivos en su computadora. Para ver los activos, utilice lo siguiente:

docker ps

Verá una salida similar a la siguiente:

Output

CONTAINER ID IMAGE COMMAND CREATED

A través de este tutorial, inició dos contenedores: uno de la imagen hello-world y otro de la imagen ubuntu. Ambos contenedores ya no están en ejecución, pero aún existen en su sistema.

Para ver todos los contenedores, activos e inactivos, ejecute docker ps con el conmutador -a:

docker ps -a

Visualizará un resultado similar a esto:

1c08a7a0d0e4 ubuntu "/bin/bash" 2 minutes ago Exited (0) 8 seconds ago quizzical\_mcnulty

a707221a5f6c hello-world "/hello" 6 minutes ago Exited (0) 6 minutes ago youthful\_curie

Para ver el último contenedor que creó, páselo al conmutador -l:

docker ps -l

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

1c08a7a0d0e4 ubuntu "/bin/bash" 2 minutes ago Exited (0) 40 seconds ago quizzical\_mcnulty

Para iniciar un contenedor detenido, utilice docker start, seguido del o el nombre ID del contenedor. Vamos a iniciar el contenedor basado en Ubuntu con el ID 1c08a7a0d0e4:

***$docker start 1c08a7a0d0e4***

El contenedor se iniciará y podrá usar docker ps para ver su estado:

Output

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

1c08a7a0d0e4 ubuntu "/bin/bash" 3 minutes ago Up 5 seconds quizzical\_mcnulty

Para detener un contenedor en funcionamiento, utilice **docker stop**, seguido del ID o nombre del contenedor. Esta vez usaremos el nombre que Docker asignó al contenedor, que es quizzical\_mcnulty:

***$docker stop quizzical\_mcnulty***

Una vez que decidida que ya no necesita un contenedor, elimínelo con el comando docker rm y use nuevamente el ID o el nombre del contenedor. Utilice el comando docker ps -a para encontrar el ID o nombre del contenedor asociado con la imagen hello-world y elimínelo.

***$docker rm youthful\_curie***

Puede iniciar un nuevo contenedor y darle un nombre usando el conmutador --name. También podrá usar el conmutador de --rm para crear un contenedor que se elimine de forma automática cuando se detenga. Consulte el comando docker run help para obtener más información sobre estas y otras opciones.

Los contenedores pueden convertirse en imágenes que podrá usar para crear contenedores nuevos. Veamos cómo funciona esto.

# Paso 7: Confirmar cambios aplicados a una imagen de Docker en un contenedor

Cuando inicie una imagen de Docker, podrá crear, modificar y eliminar archivos del mismo modo que con una máquina virtual. Los cambios que realice solo se aplicarán al contenedor en cuestión. Podrá iniciarlo y detenerlo, pero una vez que lo destruya con el comando docker rm, los cambios se perderán por completo.

En esta sección verá la forma de guardar el estado de un contenedor como una nueva imagen de Docker.

Después de instalar Node.js dentro del contenedor de Ubuntu, dispondrá de un contenedor que se ejecuta a partir de una imagen, pero este es diferente de la imagen que utilizó para crearlo. Sin embargo, quizá desee reutilizar este contenedor Node.js como base de nuevas imágenes posteriormente.

Luego, confirme los cambios en una nueva instancia de imagen de Docker utilizando el siguiente comando:

***$docker commit -m "What you did to the image" -a "Author Name" container\_id repository/new\_image\_name***

El conmutador -m es para el mensaje de confirmación que le permite a usted y a otros a saber qué cambios realizaron, mientras que -a se utiliza para especificar el autor. El container\_id es el que observó anteriormente en el tutorial cuando inició la sesión interactiva de Docker. A menos que haya creado repositorios adicionales en Docker Hub, repository generalmente corresponde a su nombre de usuario de Docker Hub.

Por ejemplo, para el usuario sammy, con el ID de contenedor d9b100f26f636, el comando sería el siguiente:

***$docker commit -m "added Node.js" -a "sammy" d9b100f2f636 sammy/ubuntu-nodejs***

Cuando confirme una imagen, la nueva imagen se guardará a nivel local en su computadora. Más adelante, en este tutorial, aprenderá a introducir una imagen en un registro de Docker como Docker Hub para que otros puedan acceder a ella.

Listar las imágenes de Docker de nuevo mostrará la nueva imagen, así como la anterior de la que se derivó:

***$docker images***

Verá resultados como este:

Output

REPOSITORY TAG IMAGE ID CREATED SIZE

sammy/ubuntu-nodejs latest 7c1f35226ca6 7 seconds ago 179MB

...

En este ejemplo ubuntu-nodejs es la nueva imagen, derivada de la imagen ubuntu existente de Docker Hub. La diferencia de tamaño refleja los cambios realizados. En este ejemplo, el cambio fue la instalación de NodeJS. Por ello, la próxima vez que deba ejecutar un contenedor usando Ubuntu con NodeJS preinstalado, podrá usar simplemente la nueva imagen.

También podrá crear imágenes de un Dockerfile, lo cual le permitirá automatizar la instalación de software en una nueva imagen. Sin embargo, eso queda fuera del alcance de este tutorial.

Ahora, compartiremos la nueva imagen con terceros para que puedan crear contenedores a partir de ella.

# Paso 8: Introducir imágenes de Docker en un repositorio de Docker

El siguiente paso lógico después de crear una nueva imagen a partir de una imagen existente es compartirla con algunos de sus amigos, con todo el mundo en Docker Hub, o en otro registro de Docker al que tenga acceso. Para introducir una imagen a Docker Hub o a cualquier otro registro de Docker, deberá tener una cuenta en el sistema.

En esta sección verá cómo introducir una imagen de Docker en Docker Hub. Para aprender a crear su propio registro privado de Docker, consulte Cómo configurar un registro de Docker privado en Ubuntu 14.04.

Para introducir su imagen, primero inicie sesión en Docker Hub.

***$docker login -u docker-registry-username***

Se le solicitará autenticarse usando su contraseña de Docker Hub. Si especificó la contraseña correcta, la autenticación tendrá éxito.

Nota: Si su nombre de usuario de registro de Docker es diferente del nombre de usuario local que usó para crear la imagen, deberá etiquetar su imagen con su nombre de usuario de registro. Para el ejemplo que se muestra en el último paso, deberá escribir lo siguiente:

***$docker tag sammy/ubuntu-nodejs docker-registry-username/ubuntu-nodejs***

Luego podrá introducir su propia imagen usando lo siguiente:

***$docker push docker-registry-username/docker-image-name***

Para introducir la imagen ubuntu-nodejs en el repositorio de sammy, el comando sería el siguiente:

***$docker push sammy/ubuntu-nodejs***

El proceso puede tardar un tiempo en completarse cuando se suben las imágenes, pero una vez que finalice el resultado será el siguiente:

Output

The push refers to a repository [docker.io/sammy/ubuntu-nodejs]

e3fbbfb44187: Pushed

5f70bf18a086: Pushed

a3b5c80a4eba: Pushed

7f18b442972b: Pushed

3ce512daaf78: Pushed

7aae4540b42d: Pushed

...

# ANEXO HERRAMIENTAS ONLINE PARA LA GESTIÓN DE CONTENEDORES:

## Portainer

Portainer es una solución de administración simple para Docker, consiste en una interfaz de usuario web que le permite administrar fácilmente sus contenedores, imágenes, redes y volúmenes de Docker.

Portainer está destinado a ser tan simple de implementar como de usar. Consiste en un único contenedor que puede ejecutarse en cualquier motor de Docker (se puede implementar como contenedor de Linux o como contenedor nativo de Windows).

Portainer le permite administrar sus contenedores Docker, imágenes, volúmenes, redes y más. Es compatible con el motor Docker independiente y con el modo Docker Swarm.

Implementar Portainer es tan simple como:

$ docker run -d -p 9000:9000 --restart always -v /var/run/docker.sock:/var/run/docker.sock -v /opt/portainer:/data portainer/portainer

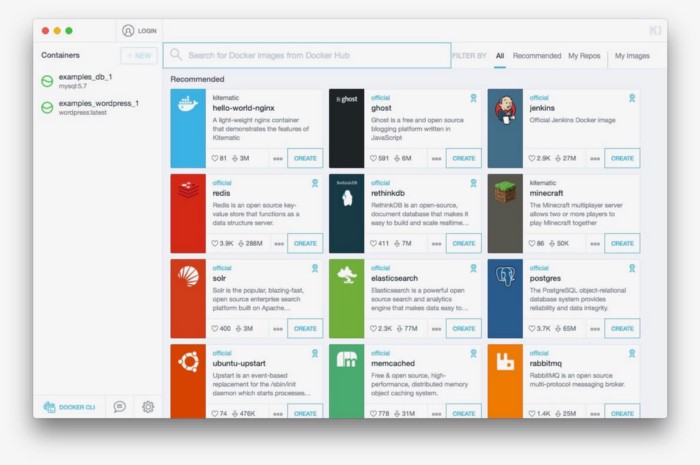
Se accede desde local o remoto en un navegador usando la siguiente dirección:

<http://DOCKER_HOST:9000>

Portainer hace uso de la API de Docker para manejar sus interacciones y monitoreo, pero para la automatización, también expone su propia API HTTP y formato de archivo de plantilla. Portainer cubre la mayoría de las áreas principales de Docker con las que le gustaría interactuar, ofreciendo creación, edición, administración, monitoreo y eliminación de contenedores, así como la capacidad de agregar, eliminar y ver imágenes, redes y volúmenes, pero no editarlos. .

## KITEMATIC

Kitematic es una aplicación simple para administrar contenedores Docker en Mac, Linux y Windows. Es un proyecto de código abierto creado para simplificar y agilizar el uso de Docker en una PC Mac o Windows. Kitematic automatiza el proceso de instalación y configuración de Docker y proporciona una interfaz gráfica de usuario (GUI) intuitiva para ejecutar contenedores de Docker. Kitematic se integra con Docker Machine para aprovisionar una máquina virtual VirtualBox e instalar Docker Engine localmente en su máquina.



Descargar:

Puede descargar e instalar Kitematic de una de las siguientes maneras:

* Elija Kitematic en el menú Docker para Mac o Docker para Windows para comenzar con la instalación de Kitematic.
* Instale Docker Toolbox (en sistemas más antiguos que no cumplan con los requisitos de Docker para Mac o Docker para Windows).
* Descargue Kitematic directamente desde la página de lanzamientos de Kitematic.

Documentación :

La documentación de Kitematic se puede encontrar en http://kitematic.com/docs.

CREAR UN CONTENEDOR CON KITEMATIC:

Una vez que tengas instalado Kitematic, levantar un contenedor es tan sencillo como elegir la imagen que quieres, o necesitas, y pulsar el botón Create. Si, tan sencillo, como lo que acabas de leer.

Así, mientras estoy escribiendo este artículo, he levantado, sin pestañear una web implementada con ghost. Tan solo he tenido que elegir la imagen, pulsar el correspondiente botón… y eso si, averiguar en que puerto se estaba publicando. Nada complicado tan solo Settings > Hostname/Ports.

Una vez instalado Kitematic, encontrarás en la ventana inicial una selección de imágenes que puedes ejecutar de inmediato. Cierto es que si la imagen no la tienes ya en tu equipo tardará unos instantes en descargarse, dependiendo de la velocidad de tu conexión. Pero una vez descargada, levantar el contenedor es cuestión de segundos, y desde ese momento podrás empezar a disfrutar.

INSTALACIÓN Y PRIMEROS PASOS

Antes de nada, instalar la aplicación. Kitematic, no se encuentra en los repositorios oficiales de Ubuntu, pero, la puedes instalar fácilmente descargando el archivo comprimido que puedes encontrar en GitHub.

En el momento de escribir este artículo, la última versión disponible era la 0.17.10. De esta forma, desde terminal, es tan sencillo, como ejecutar esta secuencia de instrucciones,

Solucionamos dependencias:

$sudo apt install gconf2 gconf-service python

descargar de:

$wget <https://github.com/docker/kitematic/releases/download/v0.17.13/Kitematic-0.17.13-Ubuntu.zip>

descomprimimos:

unzip Kitematic-0.17.13-Ubuntu.zip

instalamos:

sudo dpkg -i Kitematic-0.17.10\_amd64.deb

El siguiente paso, recomendado, es que te registres en Docker Hub, para poder utilizar tu cuenta, y así poder recuperar las imágenes que tengas subidas, y poder instalarlas en tu equipo.

USO Y FUNCIONAMIENTO

Una vez realizados los pasos anteriores, te encontrarás una ventana principal como la que ves en la siguiente captura de pantalla.

La primera pantalla de Kitematic, , un interfaz gráfico para Docker

Donde como ves, en la parte izquierda de la ventana, te encuentras un listado de los contenedores que tienes en tu equipo, así como el estado en el que se encuentran. Por otro lado, en la parte derecha, tienes una selección de imágenes, que tienes a tu disposición para descargar y poner en marcha.

### LEVANTANDO TU PRIMER CONTENEDOR

Mi recomendación es que pruebes con la primera de las imágenes, que se encuentra en la parte superior izquierda hello-world-nginx. Es el típico hello-wold, pero con un poco más de chicha, en tanto en cuanto, lo que hace es levantar un servidor web.

Una vez levantado ese contendor, haz clic en el contenedor, en la barra lateral sobre hello-worl-nginx. Verás que te aparece, en la pestaña Home, el log del contenedor que acabas de levantar, así como los volúmenes que le tocan a ese contenedor website\_files.

Al hacer clic sobre el volumen, te va a preguntar si lo quieres habilitar, y en caso que respondas afirmativamente, podrás acceder a su contenido, y en su caso modificarlo. Y digo en su caso, porque vas a necesitar derechos de administrador, root, para poder hacerlo.

Pero ¿donde puedo ver ese contenedor? Al lado de la pestaña Home, encuentras la pestaña Settings. Haz clic sobre ella, y a continuación, clic en Hostname/Ports. Al hacerlo, verás los puertos configurados, y la correspondencia entre el puerto del contenedor, y el puerto de tu equipo. Por ejemplo, en mi caso el puerto era el 32774, pero en tu caso puede ser cualquier otro. Así, simplemente haz clic en localhost, y se abrirá una nueva pestaña en tu navegador llevará a esa página en cuestión.

Ahora ¿Quieres jugar a 2048? Esto es algo que te propone directamente los desarrolladores de Kitematic. Para hacer esto es realmente sencillo, simplemente, tienes que seguir las indicaciones que te he comentado anteriormente, para editar el volumen. Una vez hecho esto, sigue los pasos que te indico a continuación,

cd ~/Documents/Kitematic/hello-world-nginx/website\_files/

sudo su

wget https://github.com/gabrielecirulli/2048/archive/master.zip -O 2048.zip

unzip 2048.zip

cd 2048-master/

mv \* ../

A partir de aquí, recarga la página, y a jugar un rato… pero no te distraigas mucho, que este juego tiene mucho vicio.

Jugando al 2048 utilizando Kitematic, , un interfaz gráfico para Docker

OPERANDO CON LOS CONTENEDORES

Además de poder modificar todos los parámetros que consideres de la configuración, también es posible realizar otras operaciones como, parar o reiniciar un contenedor o entrar en el contenedor para trabajar desde su interior.

En este último caso, previo, tienes que configurar el terminal que quieres utilizar para estos menesteres. Para ello, en la parte inferiror izquierda encontrarás una rueda dentada. Haciendo clic sobre ella irás a la configuración de la aplicación, donde puedes elegir varias opciones. Entre ellas, se encuentra la ruta del terminal. En mi caso estaba definido a /usr/bin/xterm. Sin embargo, ese terminal, no lo tengo instalado. En su defecto, he configurado /usr/bin/terminator.

Por otro lado, también puedes elegir el shell que quieres utilizar. Sin embargo, ten en cuenta, que este shell va a depender del shell que esté disponible dentro del contenedor, no es el que tu utilizas por defecto en tu equipo. Yo te recomendaría que dejarás el que viene por defecto sh.