**Refer the link: -https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-22-04**

**Step 4 — Working with Docker Images**

To check whether you can access and download images from Docker Hub, type:

1. docker run hello-world

The output will indicate that Docker in working correctly:

Output

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

latest: Pulling from library/hello-world

2db29710123e: Pull complete

Digest: sha256:bfea6278a0a267fad2634554f4f0c6f31981eea41c553fdf5a83e95a41d40c38

Status: Downloaded newer image for hello-world:latest

Hello from Docker!

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

...

You can search for images available on Docker Hub by using the docker command with the search subcommand. For example, to search for the Ubuntu image, type:

1. docker search ubuntu

Copy

The script will crawl Docker Hub and return a listing of all images whose name matches the search string. In this case, the output will be similar to this:

Output

NAME DESCRIPTION STARS OFFICIAL AUTOMATED

ubuntu Ubuntu is a Debian-based Linux operating sys… 14048 [OK]

websphere-liberty WebSphere Liberty multi-architecture images … 283 [OK]

ubuntu-upstart DEPRECATED, as is Upstart (find other proces… 112 [OK]

neurodebian NeuroDebian provides neuroscience research s… 88 [OK]

open-liberty Open Liberty multi-architecture images based… 51 [OK]

...

In the **OFFICIAL** column, **OK** indicates an image built and supported by the company behind the project. Once you’ve identified the image that you would like to use, you can download it to your computer using the pull subcommand.

Execute the following command to download the official ubuntu image to your computer:

1. docker pull ubuntu

Copy

You’ll see the following output:

Output

Using default tag: latest

latest: Pulling from library/ubuntu

e0b25ef51634: Pull complete

Digest: sha256:9101220a875cee98b016668342c489ff0674f247f6ca20dfc91b91c0f28581ae

Status: Downloaded newer image for ubuntu:latest

docker.io/library/ubuntu:latest

After an image has been downloaded, you can then run a container using the downloaded image with the run subcommand. As you saw with the hello-world example, if an image has not been downloaded when docker is executed with the run subcommand, the Docker client will first download the image, then run a container using it.

To see the images that have been downloaded to your computer, type:

1. docker images

Copy

The output will look similar to the following:

Output

REPOSITORY TAG IMAGE ID CREATED SIZE

ubuntu latest 1d622ef86b13 3 weeks ago 73.9MB

hello-world latest bf756fb1ae65 4 months ago 13.3kB

As you’ll see later in this tutorial, images that you use to run containers can be modified and used to generate new images, which may then be uploaded (*pushed* is the technical term) to Docker Hub or other Docker registries.

Let’s look at how to run containers in more detail.

**Step 5 — Running a Docker Container**

The hello-world container you ran in the previous step is an example of a container that runs and exits after emitting a test message. Containers can be much more useful than that, and they can be interactive. After all, they are similar to virtual machines, only more resource-friendly.

As an example, let’s run a container using the latest image of Ubuntu. The combination of the **-i** and **-t** switches gives you interactive shell access into the container:

1. docker run -it ubuntu

Copy

Your command prompt should change to reflect the fact that you’re now working inside the container and should take this form:

Output

root@d9b100f2f636:/#

Note the container id in the command prompt. In this example, it is d9b100f2f636. You’ll need that container ID later to identify the container when you want to remove it.

Now you can run any command inside the container. For example, let’s update the package database inside the container. You don’t need to prefix any command with sudo, because you’re operating inside the container as the **root** user:

1. apt update

Copy

Then install any application in it. Let’s install Node.js:

1. apt install nodejs

Copy

This installs Node.js in the container from the official Ubuntu repository. When the installation finishes, verify that Node.js is installed:

1. node -v

Copy

You’ll see the version number displayed in your terminal:

Output

v12.22.9

Any changes you make inside the container only apply to that container.

To exit the container, type exit at the prompt.

Let’s look at managing the containers on our system next.

**Step 6 — Managing Docker Containers**

After using Docker for a while, you’ll have many active (running) and inactive containers on your computer. To view the **active ones**, use:

1. docker ps

Copy

You will see output similar to the following:

Output

CONTAINER ID IMAGE COMMAND CREATED

In this tutorial, you started two containers; one from the hello-world image and another from the ubuntu image. Both containers are no longer running, but they still exist on your system.

To view all containers — active and inactive, run docker ps with the -a switch:

1. docker ps -a

Copy

You’ll see output similar to this:

Output

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

1c08a7a0d0e4 ubuntu "bash" About a minute ago Exited (0) 7 seconds ago dazzling\_taussig

587000e49d53 hello-world "/hello" 5 minutes ago Exited (0) 5 minutes ago adoring\_kowalevski

To view the latest container you created, pass it the -l switch:

1. docker ps -l

Copy

Output

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

1c08a7a0d0e4 ubuntu "bash" 3 minutes ago Exited (0) 2 minutes ago dazzling\_taussig

To start a stopped container, use docker start, followed by the container ID or the container’s name. Let’s start the Ubuntu-based container with the ID of 1c08a7a0d0e4:

1. docker start 1c08a7a0d0e4

Copy

The container will start, and you can use docker ps to see its status:

Output

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

1c08a7a0d0e4 ubuntu "bash" 6 minutes ago Up 8 seconds dazzling\_taussig

To stop a running container, use docker stop, followed by the container ID or name. This time, we’ll use the name that Docker assigned the container, which is dazzling\_taussig:

1. docker stop dazzling\_taussig

Copy

Once you’ve decided you no longer need a container anymore, remove it with the docker rm command, again using either the container ID or the name. Use the docker ps -a command to find the container ID or name for the container associated with the hello-world image and remove it.

1. docker rm adoring\_kowalevski

Copy

You can start a new container and give it a name using the --name switch. You can also use the --rm switch to create a container that removes itself when it’s stopped. See the docker run help command for more information on these options and others.

Containers can be turned into images which you can use to build new containers. Let’s look at how that works.

**Step 7 — Committing Changes in a Container to a Docker Image**

When you start up a Docker image, you can create, modify, and delete files just like you can with a virtual machine. The changes that you make will only apply to that container. You can start and stop it, but once you destroy it with the docker rm command, the changes will be lost for good.

This section shows you how to save the state of a container as a new Docker image.

After installing Node.js inside the Ubuntu container, you now have a container running off an image, but the container is different from the image you used to create it. But you might want to reuse this Node.js container as the basis for new images later.

Then commit the changes to a new Docker image instance using the following command.

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

Copy

The **-m** switch is for the commit message that helps you and others know what changes you made, while **-a** is used to specify the author. The container\_id is the one you noted earlier in the tutorial when you started the interactive Docker session. Unless you created additional repositories on Docker Hub, the repository is usually your Docker Hub username.

For example, for the user **sammy**, with the container ID of d9b100f2f636, the command would be:

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

Copy

When you *commit* an image, the new image is saved locally on your computer. Later in this tutorial, you’ll learn how to push an image to a Docker registry like Docker Hub so others can access it.

Listing the Docker images again will show the new image, as well as the old one that it was derived from:

1. docker images

Copy

You’ll see output like this:

Output

REPOSITORY TAG IMAGE ID CREATED SIZE

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

...

In this example, ubuntu-nodejs is the new image, which was derived from the existing ubuntu image from Docker Hub. The size difference reflects the changes that were made. And in this example, the change was that NodeJS was installed. So next time you need to run a container using Ubuntu with NodeJS pre-installed, you can just use the new image.

You can also build Images from a Dockerfile, which lets you automate the installation of software in a new image. However, that’s outside the scope of this tutorial.

Now let’s share the new image with others so they can create containers from it.

**Step 8 — Pushing Docker Images to a Docker Repository**

The next logical step after creating a new image from an existing image is to share it with a select few of your friends, the whole world on Docker Hub, or other Docker registry that you have access to. To push an image to Docker Hub or any other Docker registry, you must have an account there.

To push your image, first log into Docker Hub.

1. docker login -u docker-registry-username

Copy

You’ll be prompted to authenticate using your Docker Hub password. If you specified the correct password, authentication should succeed.

**Note:** If your Docker registry username is different from the local username you used to create the image, you will have to tag your image with your registry username. For the example given in the last step, you would type:

1. docker tag sammy/ubuntu-nodejs docker-registry-username/ubuntu-nodejs

Copy

Then you may push your own image using:

1. docker push docker-registry-username/docker-image-name

Copy

To push the ubuntu-nodejs image to the **sammy** repository, the command would be:

1. docker push sammy/ubuntu-nodejs

Copy

The process may take some time to complete as it uploads the images, but when completed, the output will look like this:

Output

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

e3fbbfb44187: Pushed

5f70bf18a086: Pushed

a3b5c80a4eba: Pushed

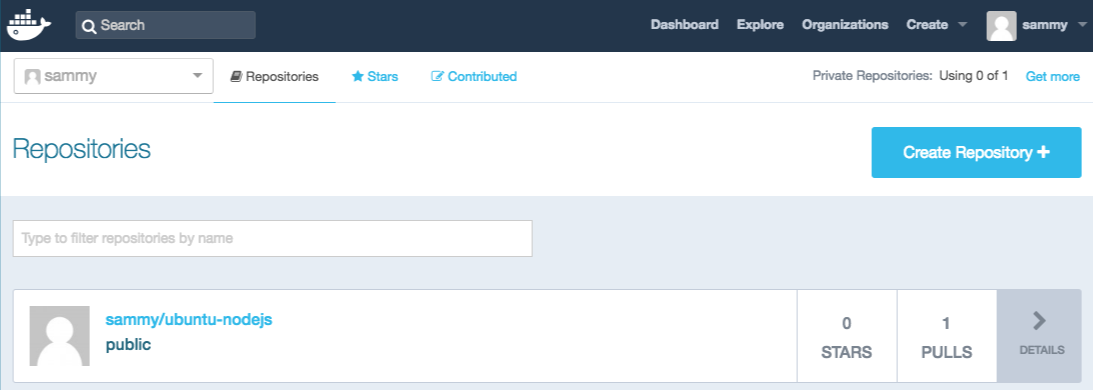
7f18b442972b: Pushed

3ce512daaf78: Pushed

7aae4540b42d: Pushed

...

After pushing an image to a registry, it should be listed on your account’s dashboard, like that show in the image below.



If a push attempt results in an error of this sort, then you likely did not log in:

Output

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

e3fbbfb44187: Preparing

5f70bf18a086: Preparing

a3b5c80a4eba: Preparing

7f18b442972b: Preparing

3ce512daaf78: Preparing

7aae4540b42d: Waiting

unauthorized: authentication required

Log in with docker login and repeat the push attempt. Then verify that it exists on your Docker Hub repository page.

You can now use docker pull sammy/ubuntu-nodejs to pull the image to a new machine and use it to run a new container.