Pull Ubuntu Image with latest version.

docker knows how to manage containers as well images. User only need to instruct docker through different commands.

To pull images, instruct docker with below command.

docker pull ubuntu:latest

What above command says in human language:

docker please download or install Ubuntu image with latest version.

What's happening behind the scene after running above command:

1. docker checks from the list of available images(using "docker images" command) that whether the Ubuntu:latest image is present locally or not.
2. If present locally then fine.
3. Else, docker will download image from [DockerHub](https://hub.docker.com/" \t "_blank). DockerHub is public registory for Images as GitHub is public repository for codes.

Note:

Like git is a cli tool to interact with GitHub. Similarly, docker is also a cli tool to interact with DockerHub.

Relate with git

If you are familiar with git, then you must know to clone or download repo from github. To clone git repo from GitHub, command is:

git clone repository\_name

Similarly, to pull image from DockerHub command is:

docker pull ubuntu:latest

git ---> docker

clone ---> pull

repository\_name ---> image name

To see list of all available images instruct docker with below command.

docker images

Conclusion

docker pull ubuntu:latest

## Run Container from Ubuntu Image.

#### To run container from Ubuntu image, instruct docker with command:

docker run ubuntu:latest

#### What does above command says in human language:

docker run new container or new light weight OS on top of host OS using Image Ubuntu:latest.

Do you know ?

The actual image used in your host OS is of size approx 8-16 GiB, whereas the image used by Containers are of sizes 10 MiB to 1000 MiB.

Now check whether container is running or not?

docker ps

You will find that no container is Running. Why??

You will find that container is in stop mode.

docker ps -a

Yes, the container is in stop mode. Why so?

After knowing the answer you will be like "WOW :)".

#### Understand this:

Let’s understand with real life examples. As you all know that you can use your phone to make phone calls or use the internet as long as you have talktime balance or data balance respectively. As it ends, you can’t make phone calls or use the Internet. Similarly, the same logic applied on Containers. As long as some program is running inside your container, the container will continue to run. If programs stops running, then your container will also stop. That's why your container stops automatically after running "docker run ubuntu:latest" .So, to make your container keep running you need to run some program i.e. a terminal or bash shell program using -t option. And along with -t option you need to include -i option also. Why so? To make your terminal continuously run. Have you ever noticed that your terminal cursor continously blinks? It's a signal to the user that it’s waiting for your input.

NOTE: Terminal is a black color interface which takes your input and prints the corresponding output. Terminal is also a program.

Windows has Poweshell as Terminal.

Ubuntu, RedHat, or any other Linux distros has bash shell as Terminal.

#### Are you Curious ??

Why on starting new host operating system you see graphical interface, beautiful icons, etc, whereas on starting new light weight OS(i.e. container) you see a black screen known as Terminal. Why so ? Click here to know.

Now, let's get back to the topic and run container.

docker run -t -i --name=tutorial ubuntu:latest

#### What above command says in human language:

Hey, docker please run container(new light-weight OS) on top of host OS using Ubuntu Image, with name of container as tutorial and along with that start program called Terminal in Interactive or input mode, So, that my container keeps running.

#### What's happening behind the scene after running above command:

At the time of running command you was in your host Operating System. But now you are inside Container. In technical terms, now you are inside different namespaces. Initially you was in host namespace or inside host boundary and now you are inside container namespace or inside container boundary. We will discuss later about the namespaces.

#### How to get confirmation that you are inside Container.

1. Whatever files(that you created) present inside your host operating system, you won't see inside container and vice-versa.
2. See to the left of terminal, from

Switched from "ubuntu $" to "root@ba94a237248f:/#" ubuntu $ ---> you are in host OS. root@ba94a237248f:/# ---> You are inside container.

### **Conclusion**

docker run -t -i --name=tutorial ubuntu:latest

# **Install dependencies**

The final motive is to run golang "hello world" program inside containers, which require golang programming languge.

Before installing golang you need to download some commands which are not present in Ubuntu Image. The reason is that Images are light-weight and it only contains only necessary programs.

First of update packages and liabraries of Image:

apt-get update

Now install commands like wget, vim

apt-get install wget -y

apt-get install vim -y

Now, download or install golang programming langiuage.

wget https://go.dev/dl/go1.18.3.linux-amd64.tar.gz

Since, the file is in tar mode, similar to zip. Need to extract files from it. Run the below command to extract files.

tar xvf go1.18.3.linux-amd64.tar.gz

x ---> Extract files

v ---> show me all output while extracting files

f ---> denotes file

The file will be extracted in your current directory, inside go/ directory.

Move this go/ directory inside /usr/local/

mv go/ /usr/local/

Now, check whether go is installed or not.

go version

There is no such command. Why so? After installing go, then also it says no go command is found. The go command is present inside /usr/local/go/bin folder. So, add this folder in your path using export keyword.

export PATH=$PATH:/usr/local/go/bin

or

echo "export PATH=$PATH:/usr/local/go/bin" >> ~/.bashrc

Now, again check whether go is installed or not.

go version

Yes, now it is installed. Finally golang is installed inside you new light weight operating system.

### **Conclusion**

apt-get update

apt-get install wget -y

apt-get install vim -y

wget https://go.dev/dl/go1.18.3.linux-amd64.tar.gz

mv go/ /usr/local/

go version

export PATH=$PATH:/usr/local/go/bin

BACKNEXT

## Write hello world program

#### Crete new directory

mkdir hello

#### Create file with name main.go

vim main.go

#### paste the below code,

package main

import "fmt"

func main() {

fmt.Println("Hello, World!")

}

#### Execute the program

go run main.go

### **Conclusion**

mkdir hello

vim main.go //Copy the code here

go run main.go

## Create Manually Image

Now let's create imgae from stopped container not running container.

To stop container, write below command inside your container.

exit

#### What's happening behind the scene after running above command:

Remember that point that container continue to run till any program running inside it. And that program was Terminal or bash shell. On executing command "exit" on terminal, the terminal gets exited. In other words terminal program stops running. As a result now no program is running inside container. Therefore container will also stop.

#### Do you know?

Is there any way to come out of container without stopping it ? Yes,

press "ctrl p+q" at the same time

Is there any way to go inside any running container? Yes,

docker attach tutorial

The above command takes you inside tutorial container.

Lets get back to the topic. After stopping container. Check it using below command

docker ps -a

Now, to create Image from it. Run the below command:

docker commit tutorial tutorial-image

#### What above command says in human language:

Hey docker please save all the changes that I made inside container tutorial. And create new image of name tutorial-image.

#### Relate with git

"docker commit" is similar to "git commit" command. In "git commit" also it save all the changes till that point of time you made.

Now, check whether "tutorial-image" is present or not:

docker images

#### What is the advantage of creating image:

You don't need to install dependencies again and again if you are using "tutorial-image". You can use the same image everytime. And also share with anyone. The environment will be same for everyone whosoever using this image.

### **Conclusion**

exit

docker ps -a

docker commit tutorial tutorial-image





