

Docker [day 1]

This doc all about 1. Docker installation 2. How to pull official docker image. 3. How to access that application

- ✓ Docker is a tool or a platform. which is used to applications dependencies. These dependencies can be libraries can be a set of binaries or any tools, so whatever that is required for your application to work independently, all the requirement will be bundled up with the help of docker. So we use Docker to do that.
- ✓ Docker which will isolate your services. So when you're using Docker, you don't have to worry about the infrastructure.
- ✓ So before Docker, what we have to do if you are creating and docker. One is for packaging your application so that your application becomes platform independent.
- ✓ Also, when you're using docker, say, for example, I'm developing an application. I, in my application. I have a front-end part. I have a back end, and I have a database part.
- ✓ So previously before docker or before virtual machines come up, what they were doing. They were maintaining separate servers, independent servers for doing all these tasks. They maintain the separate server for front end development, we maintain a separate server for back-end development. We maintain a separate server. So, you have to maintain different servers so that each process does not conflict with one another.
- ✓ Conflict with your front-end process. Your front-end process does not affect your back-end process.
- ✓ So, the process to isolate the services to isolate your front-end service, to isolate a back-end service to isolate your database service we have. We were creating. We were developing all the application in a different independent machine. This work, for doing your dB work or for doing your Frame work. Okay? So you used multiple virtual machines.
- ✓ but even when you're using virtual machine. So if you want to, if you want to launch multiple services, say, for example, I'm via my application requires multiple services. In that case you had to launch. If I if my application requires 5 services, I had to launch 5 virtual machines.
- ✓ okay, so every virtual machine was huge. Virtual machine, said Bulkier. It was very hard to handle your virtual machine to booting up your virtual machine took time, though the virtual machine achieved isolation because of its size. And everything becomes difficult to handle your virtual machine. You are not able to move your virtual machine, so virtual machines are not portable. So end of the day. You are also the services that you're running

in the virtual machine was very small. Your virtual machine is huge, every operating system because you're bringing in a complete operating system the resources you are not utilizing it properly.

- ✓ You're not utilizing the virtual machines resources properly. So what happened? End of the day? Lot of resources are being left unused. You're not utilizing the resource to the maximum. So, you're wasting lot of resources and end of the day. You have, high capital expenditure and operational expenditure. Capital expenditure means to install the virtual machine and to launch a virtual machine. Everything took time.
- ✓ So, in order to avoid this, so we had to achieve isolation among the services.
- ✓ But you cannot afford to create the virtual machine. So, what we to do this docker was to avoid this, to sort this issue. Docker was introduced.
- ✓ Using Docker, you were able to achieve isolation, but without the help of virtual mission.
- ✓ So, your docker will be running on your post operating system.
- ✓ Only you know where the how your virtual machine will be running. Your virtual machine will take the help of your hardware directly.
- ✓ Your virtual machine will directly talk to your hardware. It will not talk to your mother operating system. Say, suppose in your, in your machine is a vendor's machine and your running multiple virtual machine. Your virtual machine will not talk to your windows operating system.
- ✓ So when you're using Docker, I will just show you this. I always have pictures. Yeah.



We Isolate Services



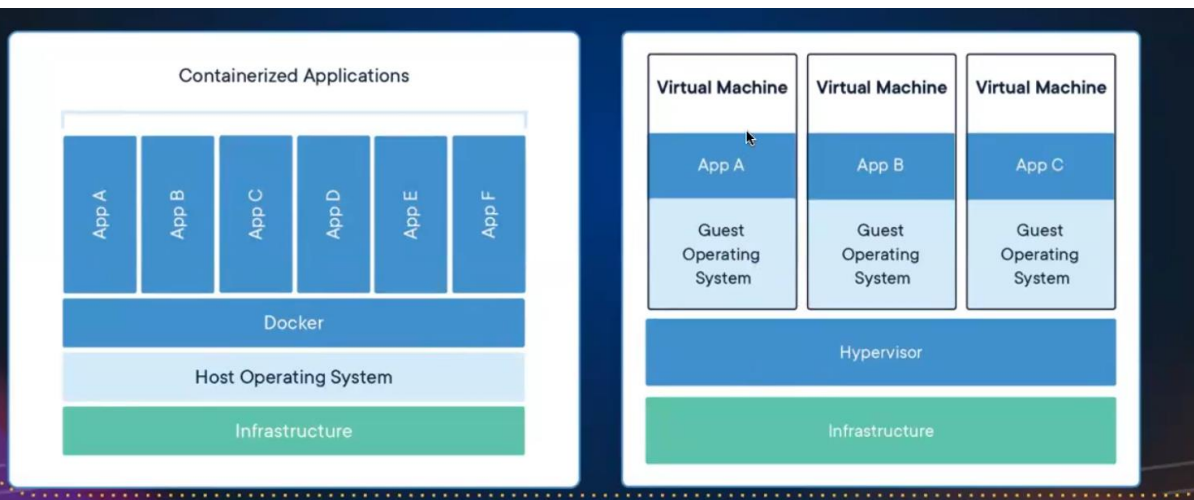
- To host our apps we need Infrastructure.
- We Use VM's/Cloud Computing to setup Infra
- We Isolate our service in OS of VM
- Because of Isolation we end up setting up multiple VM's/Instances.
- VM's/Instances will be overprovisioned.
- Results in High CapEx and OpEx

VM's are expensive

- Every VM has OS
- OS needs nurturing
- OS Needs Licensing
- OS takes time to boot
- VM's are Portable but Bulky.
- VM needs Resources for its OS
- All this to Isolate services

Point to be Noted.

- Isolating services are IMP (Need OS)
- High availability achieved by multiple instances/vm's
- Portability Matters or Eases the Deployment.
- All this raises CapEx and OpEx



Isolation without OS?

Imagine Multiple Services running in same OS but isolated.

one of the advantages of your docker. So, without help of docker, you can run multiple services or multiple application in the same machine **by achieving isolation, so that your applications can function independently without depending on other application.**

✓ Docker documentation link

[zen-class-devops-documentation/005 - Docker/docker.md at main · zen-class/zen-class-devops-documentation · GitHub](#)

docker installation:

sudo apt update

sudo apt install [docker.io](https://docs.docker.com/engine/install/ubuntu/)

```
ubuntu@ip-172-31-44-222:~$ sudo apt update
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Hit:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease
Get:4 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [1709 kB]
Get:5 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [316 kB]
Get:6 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [1966 kB]
Get:7 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Get:8 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted Translation-en [335 kB]
Get:9 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1085 kB]
Get:10 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 Packages [43.0 kB]
Fetched 5710 kB in 2s (3299 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
20 packages can be upgraded. Run 'apt list --upgradable' to see them.
ubuntu@ip-172-31-44-222:~$ sudo apt install docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap docker-doc rinse zfs-fuse | zfsutils
The following NEW packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base docker.io pigz runc ubuntu-fan
0 upgraded, 8 newly installed, 0 to remove and 20 not upgraded.
Need to get 69.8 MB of archives.
After this operation, 267 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

Client version check:

docker version

```
ubuntu@ip-172-31-44-222:~$ docker version
Client:
 Version:           24.0.5
 API version:       1.43
 Go version:        go1.20.3
 Git commit:        24.0.5-0ubuntu1-22.04.1
 Built:             Mon Aug 21 19:50:14 2023
 OS/Arch:           linux/amd64
 Context:           default
permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get
n": dial unix /var/run/docker.sock: connect: permission denied
```

Server version check:

you know Docker follows a client server architecture. So you get the client side and the server side.commands, whatever we give that get executor in the docker client.

where the commands will be processed who will receive our comment. Client will receive our comment, and the processing happens in your docker server. Okay?

So now I told you we will be running all the commands. So you will be running all the commands in your docker client.


```
ubuntu@ip-172-31-44-222:~$ sudo docker version
```

```
Client:
```

```
Version:           24.0.5
API version:       1.43
Go version:        go1.20.3
Git commit:        24.0.5-0ubuntu1~22.04.1
Built:            Mon Aug 21 19:50:14 2023
OS/Arch:          linux/amd64
Context:          default
```

```
Server:
```

```
Engine:
```

```
Version:           24.0.5
API version:       1.43 (minimum version 1.12)
Go version:        go1.20.3
Git commit:        24.0.5-0ubuntu1~22.04.1
Built:            Mon Aug 21 19:50:14 2023
OS/Arch:          linux/amd64
Experimental:     false
```

```
containerd:
```

```
Version:           1.7.2
GitCommit:
```

```
runc:
```

```
Version:           1.1.7-0ubuntu1~22.04.2
GitCommit:
```

```
docker-init:
```

```
Version:           0.19.0
GitCommit:
```

```
ubuntu@ip-172-31-44-222:~$
```

<https://docs.docker.com/engine/install/ubuntu/>

So now I have installed docker. You can say a docker group will be created and a docker user will be created. So how do you check that file. [cat /etc/group]

```
ubuntu@ip-172-31-44-222:~$ cat /etc/group
```

```
sasl:x:45:  
plugdev:x:46:ubuntu  
staff:x:50:  
games:x:60:  
users:x:100:  
nogroup:x:65534:  
systemd-journal:x:101:  
systemd-network:x:102:  
systemd-resolve:x:103:  
crontab:x:104:  
messagebus:x:105:  
systemd-timesync:x:106:  
input:x:107:  
sgx:x:108:  
kvm:x:109:  
render:x:110:  
syslog:x:111:  
tss:x:112:  
uidd:x:113:  
tcpdump:x:114:  
_ssh:x:115:  
landscape:x:116:  
fwupd-refresh:x:117:  
admin:x:118:  
netdev:x:119:ubuntu  
lxd:x:120:ubuntu  
_chrony:x:121:  
ubuntu:x:1000:  
ssl-cert:x:122:postgres  
postgres:x:123:  
postfix:x:124:  
postdrop:x:125:  
docker:x:126:
```

This is docker group.

How to check Docker user: [cat /etc/passwd]

```
ubuntu@ip-172-31-44-222:~$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:102:105:/:nonexistent:/usr/sbin/nologin
systemd-timesync:x:103:106:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
syslog:x:104:111:/:home/syslog:/usr/sbin/nologin
_apt:x:105:65534:/:nonexistent:/usr/sbin/nologin
tss:x:106:112:TPM software stack,,,:/var/lib/tpm:/bin/false
uidd:x:107:113:/:run/uidd:/usr/sbin/nologin
tcpdump:x:108:114:/:nonexistent:/usr/sbin/nologin
sshd:x:109:65534:/:run/sshd:/usr/sbin/nologin
pollinate:x:110:1:/:var/cache/pollinate:/bin/false
landscape:x:111:116:/:var/lib/landscape:/usr/sbin/nologin
```

Docker will not allow me to run the commands as a normal user. So to run your docker command, either you should be a pseudo user or you should be a root user.

if you want to avoid it. I can put my ubuntu user. I will add my ubuntu user to the Docker group you saw a docker group got created. So I want to add my ubuntu user to the Docker group. So how do you modify the user? I have every time by running my docker command, I have to give Sudo privilege.

Okay? So in order to avoid that, I add my ubuntu user to the Docker group.

So you give user mode, command.

-aG means add group as ubuntu

sudo usermod -aG docker ubuntu

```
ubuntu@ip-172-31-44-222:~$ sudo usermod -aG docker ubuntu
ubuntu@ip-172-31-44-222:~$
```

After give above command restart the machine.

```
*** System restart required ***
You have mail.
Last login: Thu Jun  6 11:11:25 2024 from 13.233.177.4
```



```
ubuntu@ip-172-31-44-222:~$ cat /etc/group
```

```
sasl:x:45:
plugdev:x:46:ubuntu
staff:x:50:
games:x:60:
users:x:100:
nogroup:x:65534:
systemd-journal:x:101:
systemd-network:x:102:
systemd-resolve:x:103:
crontab:x:104:
messagebus:x:105:
systemd-timesync:x:106:
input:x:107:
sgx:x:108:
kvm:x:109:
render:x:110:
syslog:x:111:
tss:x:112:
uidd:x:113:
tcpdump:x:114:
_ssh:x:115:
landscape:x:116:
fwupd-refresh:x:117:
admin:x:118:
netdev:x:119:ubuntu
lxd:x:120:ubuntu
_chrony:x:121:
ubuntu:x:1000:
ssl-cert:x:122:postgres
postgres:x:123:
postfix:x:124:
postdrop:x:125:
docker:x:126:ubuntu
```

Ubuntu is added as docker group

Docker images:





you have 2 different types of docker images.

1. Official docker image:





One is default docker image or official docker image or custom docker image. So official docker image means the docker image that is being given. It is already available in the docker hub library. So, you can just use that image. You are not creating your own image. The images are already available.

It's already available. You just use this docker image in order to use this application. Okay. Now, if you want to access this docker image, if you want to run your nginx application what you do, you use this image, you convert this image into a container, and then you access the application.





Trending this week ↗

 homeassistant/am... ☆83 ± 5M+	 paketobuildpacks/b... ☆35 ± 50M+	 vitess/lite A slimmed down version of Vitess containers, with just the Vitess... ☆31 ± 10M+	 friendica Welcome to the free social web. ☆86 ± 1M+
---	--	--	--

Most pulled images [View all](#)

 postgres The PostgreSQL object-relational database system provides... ☆10K+ ± 1B+	 nginx Official build of Nginx. ☆10K+ ± 1B+	 memcached Free & open source, high-performance, distributed memor... ☆2.2K ± 1B+	 busybox Busybox base image. ☆3.3K ± 1B+
--	---	---	--

Databases & Storage [View all](#)

 postgres The PostgreSQL object-relational database system provides... ☆10K+ ± 1B+	 mysql MySQL is a widely used, open-source relational database... ☆10K+ ± 1B+	 neo4j Neo4j is a highly scalable, robust native graph database. ☆1.2K ± 100M+	 mongo MongoDB document databases provide high availability and eas... ☆10K+ ± 1B+
--	---	--	--

you can see the size of the docker image. It is very portable, and you can see the size is also very small.

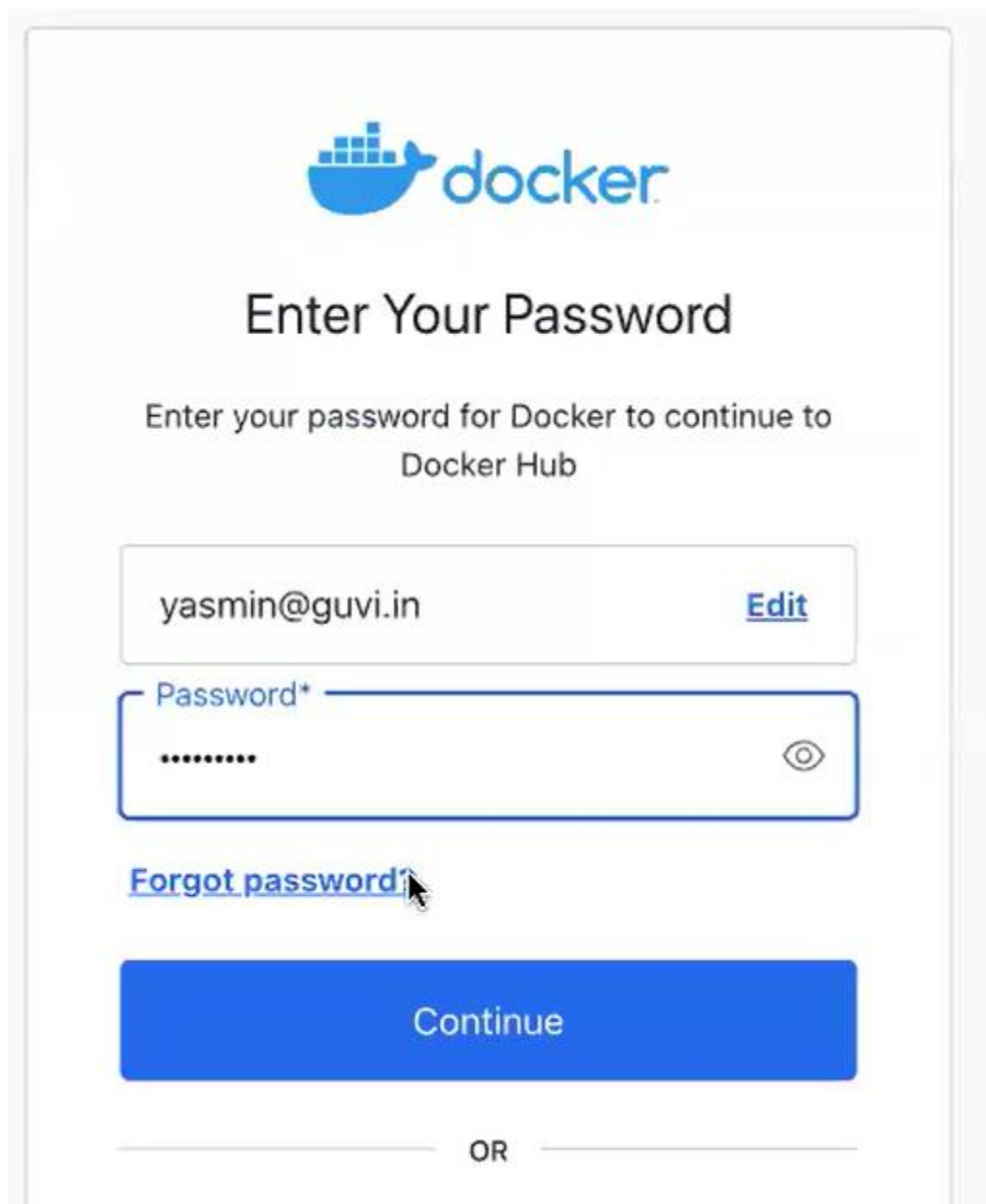
Okay, so if you're using Mysql, you don't have to. If you want to run or practice Mysql, you don't have to go and download and install mysql in your machine. So, it is a huge application. So, you can use Docker image of your Mysql application, and you can access.

2. custom Docker image:

we will be creating our own docker images for our application. Okay, this is what it happens in the real time. So developer gives you an application in the Github Repository. You have to take the application, and you have to convert that application into a docker image. So we that is called as custom Docker image.

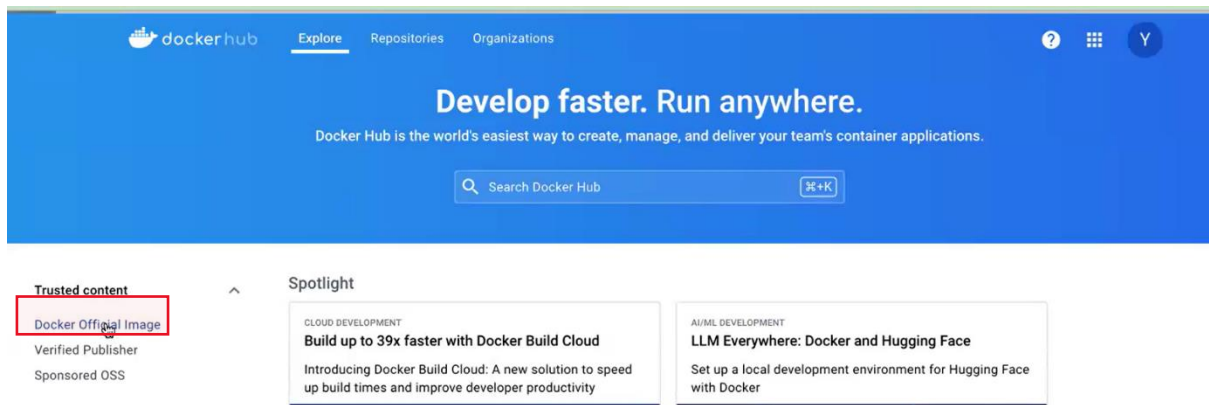
How to pull official docker image in dockerhub registry:

1. create an account in dockerhub

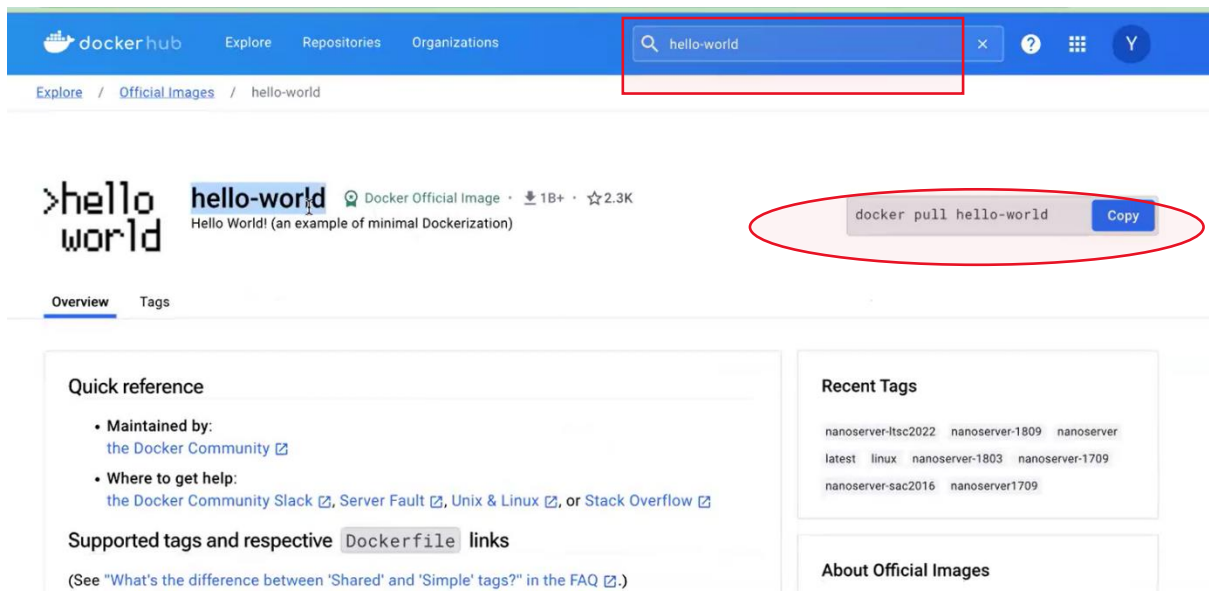


The screenshot shows the Docker Hub login interface. At the top is the Docker logo, which consists of a blue whale icon and the word "docker" in a sans-serif font. Below the logo is the heading "Enter Your Password" in a large, bold, black font. Underneath the heading is the instruction "Enter your password for Docker to continue to Docker Hub" in a smaller, regular black font. There are two input fields: the first is for the email address, containing "yasmin@guvi.in", with an "Edit" link to its right; the second is for the password, labeled "Password*" and filled with dots, with an eye icon to its right for toggling visibility. Below the password field is a link that says "Forgot password?". At the bottom of the form is a large blue button with the text "Continue" in white. Below the button is a horizontal line with the word "OR" in the center, indicating an alternative login method.

2. let's sign in



Search **hello-world image**



Copy the above command paste it in machine.

```
ubuntu@ip-172-31-44-222:~$ docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:266b191e926f65542fa8daaec01a192c4d292bfff79426f47300a046e1bc576fd
Status: Downloaded newer image for hello-world:latest
docker.io/library/hello-world:latest
```

docker images

list all images

```
ubuntu@ip-172-31-44-222:~$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
hello-world    latest    d2c94e258dcb   13 months ago  13.3kB
ubuntu@ip-172-31-44-222:~$
```

images are portable. so you can transport your image from one machine to another machine. Okay. Now, if I want to use my Hello world application. I cannot use it. If it is in the form of docker image I have to convert this docker container.

how do I convert it as container using a command called as docker run:

this is hello world application .

```
ubuntu@ip-172-31-44-222:~$ 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/
```

How to list all containers:


docker ps [show only running container]

docker ps -a [show all running and exited container]

```
ubuntu@ip-172-31-44-222:~$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS          NAMES
a22b651738e8   hello-world    "/hello"                 33 seconds ago Exited (0)    32 seconds ago          zen_driscoll
```


exited state to check the container in the exited state. Application is not running now to stop. Okay, so to check. If the container is to check the stop, the container, or the exited container




pull nginx image:

 Explore Repositories Organizations

hello-w

Explore / Official Images / hello-world



 Explore Repositories Organizations

Filters (1) [Clear All](#)

1 - 25 of 176 available results.



Suggested

 postgres 

Updated 14 days ago

The PostgreSQL object-relational database system provides reliability and data integrity.

DATABASES & STORAGE

 nginx 

Updated 6 days ago

Official build of Nginx.

WEB SERVERS



Products



☐ Images



☐ Extensions

☐ Plugins

Trusted Content

☒  Docker Official Image 

☐  Verified Publisher 

☐  Sponsored OSS 

Categories

Explore / Official Images / nginx

 nginx  Docker Official Image ·  1B+ ·  10K+

Official build of Nginx.

WEB SERVERS

docker pull nginx Copied!

Overview Tags

Quick reference

- Maintained by:
[the NGINX Docker Maintainers](#) 
- Where to get help:
[the Docker Community Slack](#) , [Server Fault](#) , [Unix & Linux](#) , or [Stack Overflow](#) 

Supported tags and respective `Dockerfile` links

Recent Tags

stable-alpine3.19-perl stable-alpine3.19-otel

stable-alpine-perl stable-alpine-otel

mainline-alpine3.19-perl mainline-alpine3.19-otel

mainline-alpine-perl mainline-alpine-otel

alpine3.19-perl alpine3.19-otel

```
ubuntu@ip-172-31-44-222:~$ docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
09f376ebb190: Pull complete
5529e0792248: Pull complete
9b3add3eb3d: Pull complete
57910a8c4316: Pull complete
7b5f78f21449: Pull complete
b7923aa4e8a6: Pull complete
785625911f12: Pull complete
Digest: sha256:0f04e4f646a3f14bf31d8bc8d885b6c951fdcf42589d06845f64d18aec6a3c4d
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
```

```
ubuntu@ip-172-31-44-222:~$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
nginx	latest	4f67c83422ec	7 days ago	188MB
hello-world	latest	d2c94e258dcb	13 months ago	13.3kB

```
ubuntu@ip-172-31-44-222:~$ docker run --name containernginx -d -p 80:80 nginx
a85210b8fd13c6215daf5ac3dbbba60b7ad881cf37f73e38a0ec0f312b6277808
```

```
ubuntu@ip-172-31-44-222:~$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
a85210b8fd13	nginx	"/docker-entrypoint..."	11 seconds ago	Up 10 seconds	0.0.0.0:80->80/tcp, :::80->80/tcp	containernginx

80/3000/4000/8080 : 80 [container or application port no:]



Ec2 port no: [[port no: its be anything]

Open port number in ec2 instance. That method called as port binding.

Copy ec2 ip :80 .access that application via web browser.



Pull sql image:

Its not web base application. Only for web base application we use port binding. Goto docker hub search MySQL. read this for understanding. For container we use environmental variable instead port number.



How to use this image

Start a `mysql` server instance

Starting a MySQL instance is simple:

```
$ docker run --name some-mysql -e MYSQL_ROOT_PASSWORD=my-secret-pw -d mysql:tag
```

... where `some-mysql` is the name you want to assign to your container, `my-secret-pw` is the password to be set for the MySQL root user and `tag` is the tag specifying the MySQL version you want. See the list above for relevant tags.

Connect to MySQL from the MySQL command line client

The following command starts another `mysql` container instance and runs the `mysql` command line client against your original `mysql` container, allowing you to execute SQL statements against your database instance:

```
$ docker run -it --network some-network --rm mysql mysql -hsome-mysql -uusername -p
```

... where `some-mysql` is the name of your original `mysql` container (connected to the `some-network` Docker network).

This image can also be used as a client for non-Docker or remote instances:

```
$ docker run -it --rm mysql mysql -hsome.mysql.host -usome-mysql-user -p
```

More information about the MySQL command line client can be found in the [MySQL](#)

```
ubuntu@ip-172-31-34-193:~$ docker pull mysql
Using default tag: latest
latest: Pulling from library/mysql
07bc88e18c4a: Pull complete
1a9c1668bf49: Pull complete
1021dda8eecf: Pull complete
fb61b56acac1: Pull complete
0bca83908a5b: Pull complete
165e8b3d37ca: Pull complete
3e1b086f1295: Pull complete
dba651668484: Pull complete
ed90f5355e12: Pull complete
0412f59ab2b5: Pull complete
Digest: sha256:aa021e164da6aacbefc59ed0b933427e4835636be380f3b6523f4a6c9564e1f0
Status: Downloaded newer image for mysql:latest
docker.io/library/mysql:latest
```

After mysql we did not mention version.so take it as latest image. That is mentioned as tag

```
ubuntu@ip-172-31-34-193:~$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
mysql	latest	fcd86ff8ce8c	5 weeks ago	578MB

```
ubuntu@ip-172-31-34-193:~$ docker run --name mysqlcontainer -d -e MYSQL_ROOT_PASSWORD=admin mysql
c9f307c17add4a9e7eede5882c0be502e01c887f947442f4153857ab158c34de
```

-d detach mode

-e Environmental variable. Environmental variable taken in mysql image description.

Admin password set by Yasmin

Mysql already pulled docker image

Environment Variables

When you start the `mysql` image, you can adjust the configuration of the MySQL instance by passing one or more environment variables on the `docker run` command line. Do note that none of the variables below will have any effect if you start the container with a data directory that already contains a database: any pre-existing database will always be left untouched on container startup.

See also <https://dev.mysql.com/doc/refman/5.7/en/environment-variables.html> for documentation of environment variables which MySQL itself respects (especially variables like `MYSQL_HOST`, which is known to cause issues when used with this image).

`MYSQL_ROOT_PASSWORD`

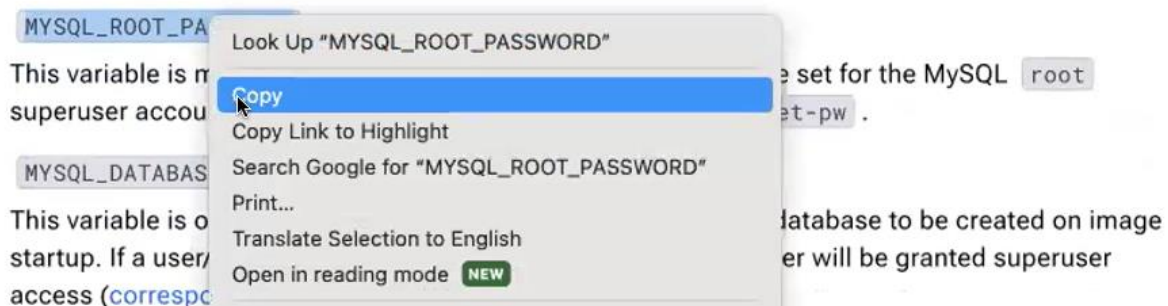
This variable is mandatory and specifies the password that will be set for the MySQL `root` superuser account. In the above example, it was set to `my-secret-pw`.

Password anything we give. she gave password as admin.

Environment Variables

When you start the `mysql` image, you can adjust the configuration of the MySQL instance by passing one or more environment variables on the `docker run` command line. Do note that none of the variables below will have any effect if you start the container with a data directory that already contains a database: any pre-existing database will always be left untouched on container startup.

See also <https://dev.mysql.com/doc/refman/5.7/en/environment-variables.html> for documentation of environment variables which MySQL itself respects (especially variables like `MYSQL_HOST`, which is known to cause issues when used with this image).



```
ubuntu@ip-172-31-34-193:~$ docker run --name mysqlcontainer -d -e MYSQL_ROOT_PASSWORD=admin mysql  
c9f307c17add4a9e7eede5882c0be502e01c887f947442f4153857ab158c34de
```

```
ubuntu@ip-172-31-34-193:~$ docker ps  
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS                               NAMES  
c9f307c17add   mysql    "docker-entrypoint.s..." 13 seconds ago Up 11 seconds 3306/tcp, 33068/tcp               mysqlcontainer
```

my container is running. Even if you don't mention the port your Mysql default will be running the port number double 3306. This is the container port number. So in the container your Mysql application will be running the port number 3306.

We don't have to manage your Mysql via Browser. So that is the reason I am not attached or I have not binded my container port with a ec2 machine port. So, I will just access my Mysql application from the Cli itself.

How to get into the container:


```
ubuntu@ip-172-31-34-193:~$ docker exec -it mysqlcontainer bash
bash-5.1#
```

MySQLcontainer – container name that we created.

Bash – that application run in bash

Environment Variables

When you start the `mysql` image, you can adjust the configuration of the MySQL instance by passing one or more environment variables on the `docker run` command line. Do note that none of the variables below will have any effect if you start the container with a data directory that already contains a database: any pre-existing database will always be left untouched on container startup.

See also <https://dev.mysql.com/doc/refman/5.7/en/environment-variables.html> for documentation of environment variables which MySQL itself respects (especially variables like `MYSQL_HOST`, which is known to cause issues when used with this image).

`MYSQL_ROOT_PASSWORD`

This variable is mandatory and specifies the password that will be set for the MySQL `root` superuser account. In the above example, it was set to `my-secret-pw`.

`-u` account name

`-p` password we give

```
ubuntu@ip-172-31-34-193:~$ docker exec -it mysqlcontainer bash
bash-5.1# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 8.4.0 MySQL Community Server - GPL

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql>
```

Now inside the mysql application.

we can create database inside

```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sys |
+-----+
4 rows in set (0.02 sec)

mysql> create database mydb;
Query OK, 1 row affected (0.01 sec)

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mydb |
| mysql |
| performance_schema |
| sys |
+-----+
5 rows in set (0.00 sec)

mysql>
```

```
mysql> use mydb;
Database changed
mysql> exit;
Bye
bash-5.1# exit
exit
ubuntu@ip-172-31-34-193:~$
```

This doc all about 1. Docker installation 2. How to pull official docker image. 3. How to access that application