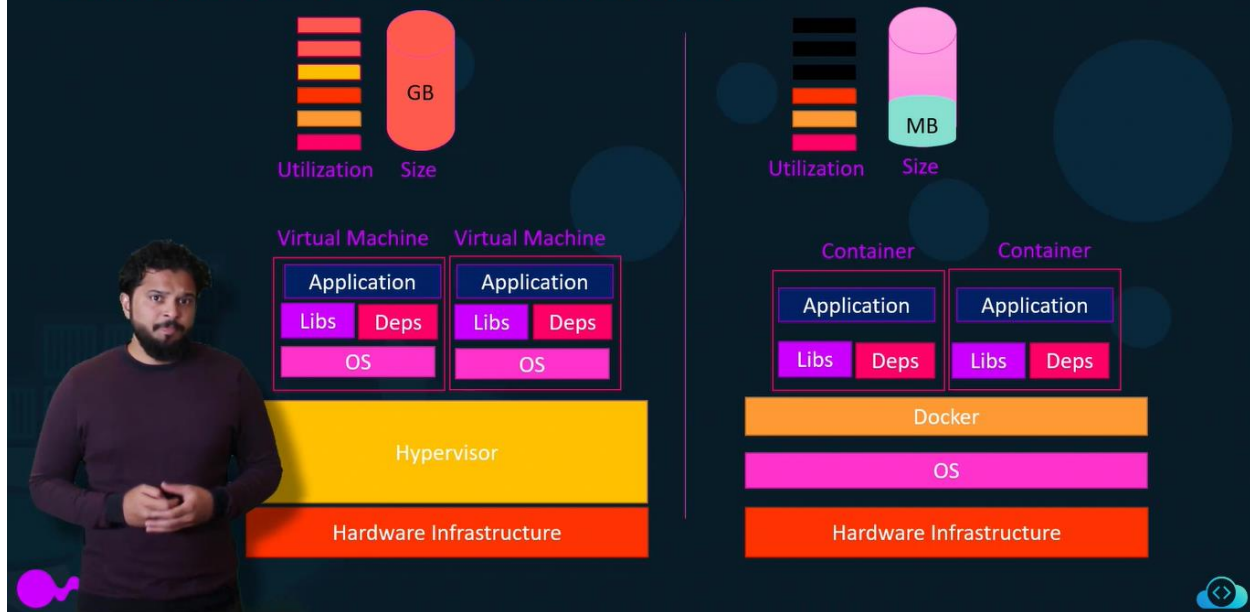


Containers vs Virtual Machines

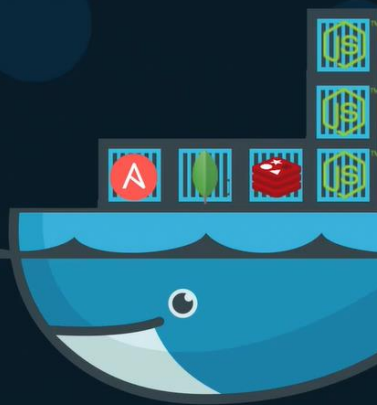


How is it done?

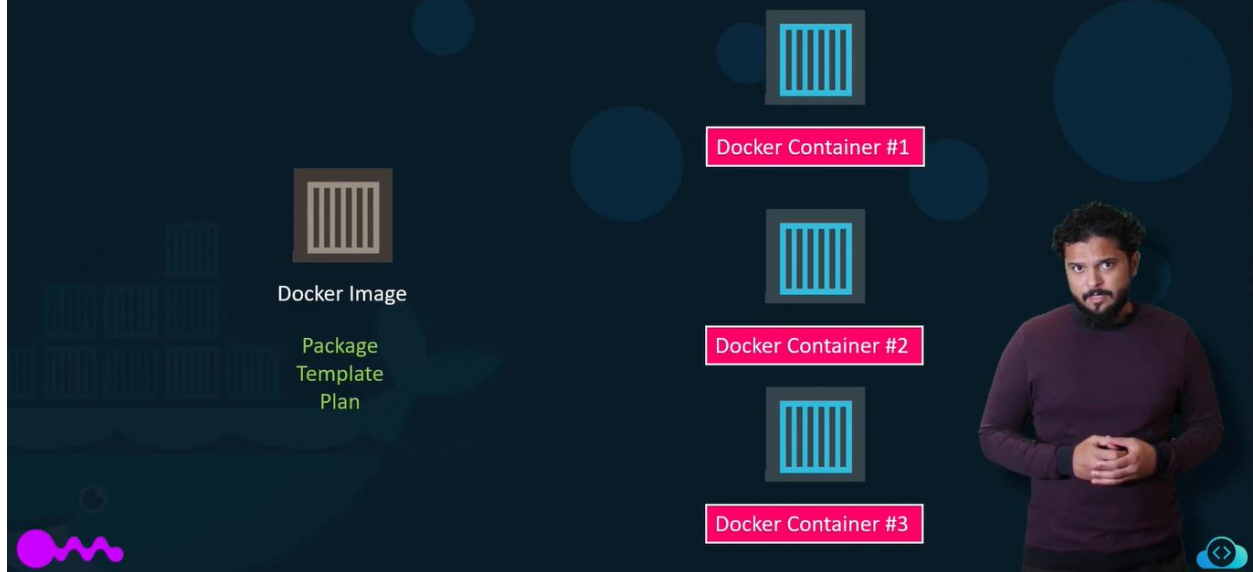
```
docker run ansible
docker run mongodb
docker run redis
docker run nodejs
docker run nodejs
docker run nodejs
```



Public Docker registry - dockerhub



Container vs image



Install docker engine in linux :

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

BASIC DOCKER COMMANDS :

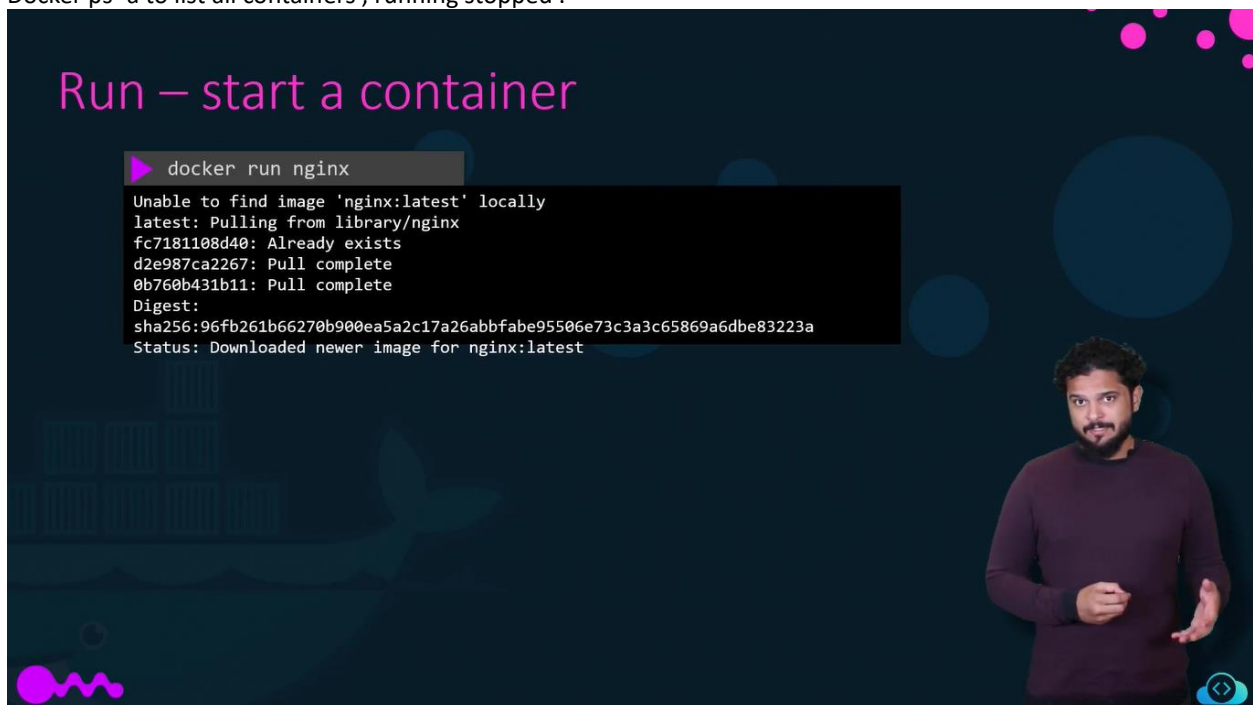
Docker run command is used to run container from a image .

Docker ps command list all running containers and some basic information about them such as container id , name of image used to run container , current status , name of container .

Docker ps -a to list all containers , running stopped .

Run – start a container

```
▶ docker run nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
fc7181108d40: Already exists
d2e987ca2267: Pull complete
0b760b431b11: Pull complete
Digest:
sha256:96fb261b66270b900ea5a2c17a26abfbabe95506e73c3a3c65869a6dbe83223a
Status: Downloaded newer image for nginx:latest
```



ps – list containers

```
▶ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
796856ac413d	nginx	"nginx -g 'daemon of..."	7 seconds ago	Up 6 seconds	80/tcp	silly_sammet

```
▶ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	NAMES
796856ac413d	nginx	"nginx -g 'daemon of..."	7 seconds ago	Up 6 seconds	silly_sammet
cff8ac918a2f	redis	"docker-entrypoint.s..."	6 seconds ago	Exited (0) 3 seconds ago	

Docker stop container id / name – to stop container .

STOP – stop a container

```
▶ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
796856ac413d	nginx	"nginx -g 'daemon of..."	7 seconds ago	Up 6 seconds	80/tcp	silly_sammet

```
▶ docker stop silly_sammet
```

```
silly_sammet
```

```
▶ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	NAME
796856ac413d	nginx	"nginx -g 'daemon of..."	7 seconds ago	Exited (0) 3 seconds ago	silly_sammet
cff8ac918a2f	redis	"docker-entrypoint.s..."	6 seconds ago	Exited (0) 3 seconds ago	

Docker rm to remove container

Rm – Remove a container

```
▶ docker rm silly_sammet
```

```
silly_sammet
```

```
▶ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	NAMES
cff8ac918a2f	redis	"docker-entrypoint.s..."	6 seconds ago	Exited (0) 3 seconds ago	relaxing_aryabhata

Docker image to get list of all images .

images – List images

```
▶ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
nginx	latest	f68d6e55e065	4 days ago	109MB
redis	latest	4760dc956b2d	15 months ago	107MB
ubuntu	latest	f975c5035748	16 months ago	112MB

Docker rmi to remove all images .

rmi – Remove images

```
▶ docker rmi nginx
```

```
Untagged: nginx:latest
Untagged: nginx@sha256:96fb261b66270b900ea5a2c17a26abbfabe95506e73c3a3c65869a6dbe83223a
Deleted: sha256:f68d6e55e06520f152403e6d96d0de5c9790a89b4cfc99f4626f68146faldbdc
Deleted: sha256:1b0c768769e2bb66e74a205317ba531473781a78b77feef8ea6fd7be7f4044e1
Deleted: sha256:34138fb60020a180e512485fb96fd42e286fb0d86cf1fa2506b11ff6b945b03f
Deleted: sha256:cf5b3c6798f77b1f78bf4e297b27cfa5b6caa982f04caeb5de7d13c255fd7a1e
```

! Delete all dependent containers to remove image

Docker pull to only pull image .

Pull – download an image

```
▶ docker run nginx
```

```
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
fc7181108d40: Already exists
d2e987ca2267: Pull complete
0b760b431b11: Pull complete
Digest:
sha256:96fb261b66270b900ea5a2c17a26abbfabe95506e73c3a3c65869a6dbe83223a
Status: Downloaded newer image for nginx:latest
```

```
▶ docker pull nginx
```

```
Using default tag: latest
latest: Pulling from library/nginx
fc7181108d40: Pull complete
d2e987ca2267: Pull complete
0b760b431b11: Pull complete
Digest:
sha256:96fb261b66270b900ea5a2c17a26abbfabe95506e73c3a3c65869a6dbe83223a
Status: Downloaded newer image for nginx:latest
```

Docker exec command to execute a command on docker container .


```
$ docker --version
Docker version 19.03.15, build 99e3ed8919
$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
PORTS	NAMES			

```
$
```

How many containers are running on this host? 0

```
$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
mysql	latest	4f06b49211c0	5 weeks ago	530MB
nginx	alpine	2bc7edbc3cf2	7 weeks ago	40.7MB
postgres	latest	680aba37fd0f	7 weeks ago	379MB
alpine	latest	b2aa39c304c2	7 weeks ago	7.05MB
redis	latest	2f66aad5324a	7 weeks ago	117MB
nginx	latest	3f8a00f137a0	7 weeks ago	142MB
ubuntu	latest	58db3edaf2be	2 months ago	77.8MB
kodekloud/simple-webapp-mysql	latest	129dd9f67367	4 years ago	96.6MB
kodekloud/simple-webapp	latest	c6e3cd9aae36	4 years ago	84.8MB

```
$
```

How many images are available on this host? 9

Run a container using the redis image

```
$ docker run redis
1:C 04 Apr 2023 08:46:27.432 # 000000000000 Redis is starting 000000000000
1:C 04 Apr 2023 08:46:27.432 # Redis version=7.0.8, bits=64, commit=00000000, modified=0, pid=1, just started
1:C 04 Apr 2023 08:46:27.432 # Warning: no config file specified, using the default config. In order to specify
1:M 04 Apr 2023 08:46:27.433 * monotonic clock: POSIX clock_gettime
1:M 04 Apr 2023 08:46:27.434 * Running mode=standalone, port=6379.
1:M 04 Apr 2023 08:46:27.435 # Server initialized
1:M 04 Apr 2023 08:46:27.435 # WARNING Memory overcommit must be enabled! Without it, a background save or repli
led, it can can also cause failures without low memory condition, see https://github.com/jemalloc/jemalloc/issue
' to /etc/sysctl.conf and then reboot or run the command 'sysctl vm.overcommit_memory=1' for this to take effect
1:M 04 Apr 2023 08:46:27.436 * Ready to accept connections
```

Stop the container you just created

```
$ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
f5303ef7a538	redis	"docker-entrypoint.s..."	About a minute ago	Exited (0) 9 seconds ago		vigilant_vaughan

```
$ docker stop f5303ef7a538
f5303ef7a538
$
```

How many containers are RUNNING on this host now? 0

```
$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
--------------	-------	---------	---------	--------	-------	-------

```
$
```

How many containers are PRESENT on the host now? 6

```
$ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
fedd4b3360a9	alpine	"/bin/sh"	4 minutes ago	Exited (0) 4 minutes ago		sweet_diffie
4fade38dbe64	alpine	"sleep 1000"	4 minutes ago	Up 4 minutes		thirsty_engelbart
d25ea8a47d2f	nginx:alpine	"/docker-entrypoint.s..."	4 minutes ago	Up 4 minutes	80/tcp	nginx-2
6f3f5b831cc0	nginx:alpine	"/docker-entrypoint.s..."	4 minutes ago	Up 4 minutes	80/tcp	nginx-1
9955c5b20708	ubuntu	"sleep 1000"	4 minutes ago	Up 4 minutes		awesome_northcut
f5303ef7a538	redis	"docker-entrypoint.s..."	7 minutes ago	Exited (0) 6 minutes ago		vigilant_vaughan

```
$
```

What is the image used to run the nginx-1 container? Nginx:alpine

What is the name of the container created using the ubuntu image? Awesome_northcut

What is the ID of the container that uses the alpine image and is not running? Fed

What is the state of the stopped alpine container? Exited

Delete all containers from the Docker Host.

Both Running and Not Running ones. Remember you may have to stop containers before deleting them.

```
$ docker ps -aq
fedd4b3360a9
4fade38dbe64
d25ea8a47d2f
6f3f5b831cc0
9955c5b20708
f5303ef7a538
$ docker stop $(docker ps -aq)
fedd4b3360a9
4fade38dbe64
d25ea8a47d2f
6f3f5b831cc0
9955c5b20708
f5303ef7a538
$ docker rm $(docker ps -aq)
fedd4b3360a9
4fade38dbe64
d25ea8a47d2f
6f3f5b831cc0
9955c5b20708
f5303ef7a538
$
```

Run the command `docker rmi ubuntu`

```
$ docker images
REPOSITORY          TAG          IMAGE ID          CREATED          SIZE
mysql               latest       4f06b49211c0     5 weeks ago     530MB
nginx               alpine      2bc7edbc3cf2     7 weeks ago     40.7MB
postgres            latest       680aba37fd0f     7 weeks ago     379MB
alpine              latest       b2aa39c304c2     7 weeks ago     7.05MB
redis               latest       2f66aad5324a     7 weeks ago     117MB
nginx               latest       3f8a00f137a0     7 weeks ago     142MB
ubuntu              latest       58db3edaf2be     2 months ago    77.8MB
kodekloud/simple-webapp-mysql latest       129dd9f67367     4 years ago     96.6MB
kodekloud/simple-webapp latest       c6e3cd9aae36     4 years ago     84.8MB
$ docker rmi ubuntu
Untagged: ubuntu:latest
Untagged: ubuntu@sha256:9a0bdde4188b896a372804be2384015e90e3f84906b750c1a53539b585fbbe7f
Deleted: sha256:58db3edaf2be6e80f628796355b1bdeaf8bea1692b402f48b7e7b8d1ff100b02
Deleted: sha256:c5ff2d88f67954bdcf1cfdd46fe3d683858d69c2cadd6660812edfc83726c654
$
```

You are required to pull a docker image which will be used to run a container later. Pull the image nginx:1.14-alpine
Only pull the image, do not create a container.

```
$ docker pull nginx:1.14-alpine
1.14-alpine: Pulling from library/nginx
bdf0201b3a05: Pull complete
3d0a573c81ed: Pull complete
8129faeb2eb6: Pull complete
3dc99f571daf: Pull complete
Digest: sha256:485b610fefec7ff6c463ced9623314a04ed67e3945b9c08d7e53a47f6d108dc7
Status: Downloaded newer image for nginx:1.14-alpine
docker.io/library/nginx:1.14-alpine
```

Run a container with the nginx:1.14-alpine image and name it webapp

```
$ docker run -d --name webapp nginx:1.14-alpine
e110f3c826a12c8df214d8edeada4ba039d78aad80dc67bda57ea80259f5a563
```

Cleanup: Delete all images on the host

Remove containers as necessary

```
$ docker rmi $(docker images -aq)
Untagged: mysql:latest
Untagged: mysql@sha256:d8dc78532e9eb3759344bf89e6e7236a34132ab79150607eb08cc746989
Deleted: sha256:4f06b49211c09ddf194684fbc6c02be56773227927b1e937b489ec414a04b3f7
Deleted: sha256:c37feeb65db60f2c4f050d31bbb15b6c484157dee1adb5c9606ac0365a23fb2f
Deleted: sha256:a2827d51846f5a47e17efcfff29fd631a2919e3518986b6aca9595f41ffb91e08
Deleted: sha256:bdcbffd6fa64f16bdc359a95adc01e29a08c988d00c07d1e0886381691c5fbbb
Deleted: sha256:abab2cb7f5c06e876f20709316506250d22ecf1a9a80a7335b5856440ec62dea
Deleted: sha256:273a05d7983e9456fa42bb421abf0a0d6efc6ccfcef48222c046d76c22a18e39
Deleted: sha256:3b99ea3f5cbfaa1575a5e2509ec8597def22d4d79b390d75014917b6a5f642d4
Deleted: sha256:8563b361cdd8efc210b7467e47c263613a88c621a4b38970269f90406aa58131
Deleted: sha256:2a1b2d738042755133d039a7daf1748ccfa2720d268d5eba40c81b7916a9179c
Deleted: sha256:899199e05f011449cf6d6b7650b9eaff7633d9d0a7d879214ac68209c419617e
Deleted: sha256:039d7eb7d1f64c9901924f551cefdd49602294d22343035f1a86e1b7207a0ce7
Deleted: sha256:4286bd3f1f4a6020052b600032b18bd63add505dd2d98dfdf9857d8f19fd8afa
Untagged: nginx:alpine
```

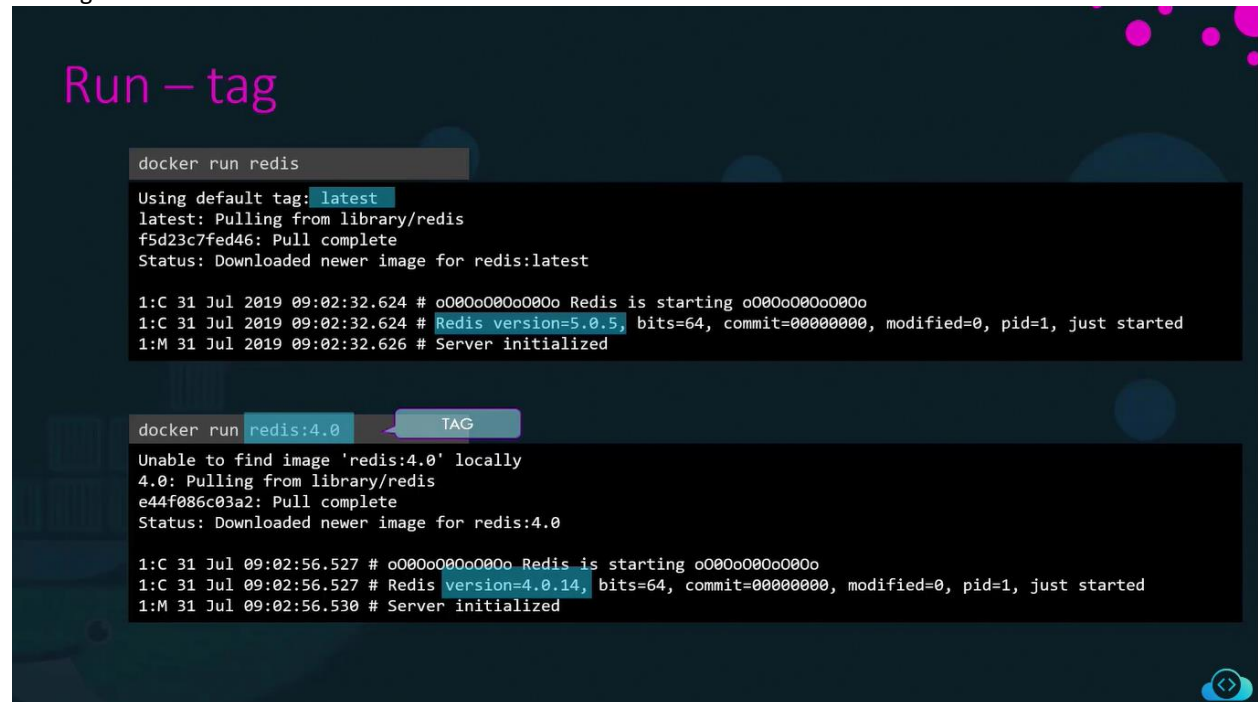
docker images -aq : it gives image id .

```
$ docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
e110f3c826a1        nginx:1.14-alpine   "nginx -g 'daemon of..." 3 minutes ago       Exited (0) 55 seconds ago              webapp
$ docker rmi 8a2fb25a19f5
Error response from daemon: conflict: unable to delete 8a2fb25a19f5 (must be forced) - image is being used by stopped container e110f3c826a1
$ docker rm e110f3c826a1
e110f3c826a1
$ docker rmi 8a2fb25a19f5
Untagged: nginx:1.14-alpine
Untagged: nginx@sha256:485b610fefec7ff6c463ced9623314a04ed67e3945b9c08d7e53a47f6d108dc7
Deleted: sha256:8a2fb25a19f5dc1528b7a3fabe8b3145ff57fe10e4f1edac6c718a3cf4aa4b73
Deleted: sha256:f68a8bcb9dbd06e0d2750eabf63c45f51734a72831ed650d2349775865d5fc20
Deleted: sha256:cbf2c7789332fe231e8defa490527a7b2c3ae8589997ceee00895f3263f0a8cf
Deleted: sha256:894f3fad7e6ecd7f24e88340a44b7b73663a85c0eb7740e7ade169e9d8491a4c
Deleted: sha256:a464c54f93a9e88fc1d33df1e0e39cca427d60145a360962e8f19a1dbf900da9
$
```


Docker run command :

Docker run redis – to run a container running redis service .

If we want to use any other version of image we can use it in tag . if we don't provide tag docker will use latest tag of image to run container .



The screenshot shows a terminal window with a dark background and pink decorative circles in the top right corner. The title of the window is "Run – tag". The terminal displays two Docker commands and their outputs. The first command is `docker run redis`, which pulls the latest Redis image and starts a container. The output shows the container ID, timestamp, and Redis startup logs, including "Redis version=5.0.5". The second command is `docker run redis:4.0`, which pulls the Redis 4.0 image and starts a container. The output shows the container ID, timestamp, and Redis startup logs, including "Redis version=4.0.14". A small Docker logo is visible in the bottom right corner of the terminal window.

```
docker run redis

Using default tag: latest
latest: Pulling from library/redis
f5d23c7fed46: Pull complete
Status: Downloaded newer image for redis:latest

1:C 31 Jul 2019 09:02:32.624 # o000o000o000o Redis is starting o000o000o000o
1:C 31 Jul 2019 09:02:32.624 # Redis version=5.0.5, bits=64, commit=00000000, modified=0, pid=1, just started
1:M 31 Jul 2019 09:02:32.626 # Server initialized

docker run redis:4.0 TAG

Unable to find image 'redis:4.0' locally
4.0: Pulling from library/redis
e44f086c03a2: Pull complete
Status: Downloaded newer image for redis:4.0

1:C 31 Jul 09:02:56.527 # o000o000o000o Redis is starting o000o000o000o
1:C 31 Jul 09:02:56.527 # Redis version=4.0.14, bits=64, commit=00000000, modified=0, pid=1, just started
1:M 31 Jul 09:02:56.530 # Server initialized
```

Inputs :

By default docker container does not listens to standard input .

It runs in non intercative mode .

To use input – we must map the standard input of our host to the docker container using -l parameter .

-i is for intercative mode and when we input it gives expected output .

-t stands for pseudo terminal .

RUN - STDIN

```
~/prompt-application$ ./app.sh
Welcome! Please enter your name: Mumshad

Hello and Welcome Mumshad!
```

```
docker run kodekloud/simple-prompt-docker

Hello and Welcome !
```

```
docker run -i kodekloud/simple-prompt-docker

Mumshad

Hello and Welcome Mumshad!
```

```
docker run -it kodekloud/simple-prompt-docker

Welcome! Please enter your name: Mumshad

Hello and Welcome Mumshad!
```

Port mapping / port publishing on containers .

Underlying host where docker is installed is known as docker host or docker engine .

When we run a containerized application it runs and we are able to see that server is running .

Now how does user application ?

Run – PORT mapping

```
docker run kodekloud/webapp
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
```

```
docker run -p 80:5000 kodekloud/webapp
```

```
docker run -p 8000:5000 kodekloud/webapp
```

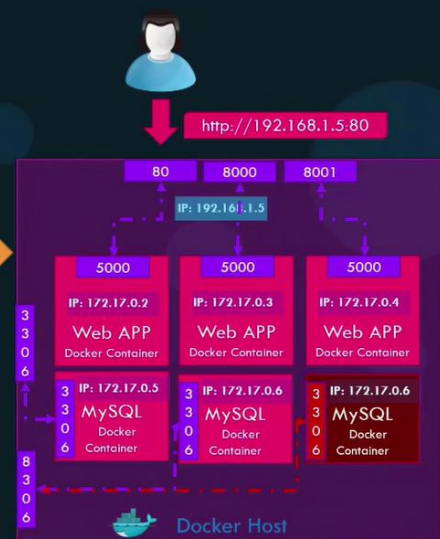
```
docker run -p 8001:5000 kodekloud/webapp
```

```
docker run -p 3306:3306 mysql
```

```
docker run -p 8306:3306 mysql
```

```
docker run -p 8306:3306 mysql
```

```
root@osboxes:/root # docker run -p 8306:3306 -e MYSQL_ROOT_PASSWORD=pass mysql
docker: Error response from daemon: driver failed programming external connectivity on endpoint boring_bhahba (5079d342b7e8e11c71d46): Bind for 0.0.0.0:8306 failed: port is already allocated.
```



Our application is listening on port 5000 so I could access my application using port 5000 .

What ip to use to access our application from web browser ?

There are two options available : use ip of docker container (every docker container gets ip assigned by default),

If we open browser from within docker host we can access application using internal ip . But since this is an internal ip user outside of docker host cant access it using this ip for this we could use the ip of the docker host .

But for that to work we must have mapped the port inside the docker container to a free port on docker host .

Eg : if we want our users to access my application from port 80 on my docker host we could map port 80 of docker host to port 5000 on the docker container . using -p in run command .

So user can access application by going to docker host ip and 80 port . and all traffic on port 80 on docker host will be routed to 5000 inside docker container .

This way we can run multiple instances of our application and map them to different ports on docker host .

We are running an instance of mysql that runs a database and listens on default port . We cannot map to same port more than once on docker host .

DATA PERSISTENCY :

How data is persisted in docker container ?

Lets say we are running mysql container , when db and tables are created the data files are stored in location /var/lib/mysql inside docker container .

Docker container has its own isolated file system any changes to any files happen within the container lets assume you dump a lot of data in databases .


What happens if we were to delete the mysql container and remove it ?

RUN – Volume mapping

```
docker run mysql
```

```
docker stop mysql  
docker rm mysql
```

```
docker run -v /opt/datadir:/var/lib/mysql mysql
```



As soon as we do that container along with all of its data gets deleted .

If you would like to persist the data you would want to map a directory outside the container on docker host to directory inside container .

We have created a directory called /opt/datadir and map that to /var/lib/mysql inside docker container using -v option and specifying the directory on docker .

This way when docker container run it will implicitly mount external directory to folder inside docker container .

Thus data will persist .

Docker inspect container name/id – to get all details in json format for containers .

Inspect Container

```
docker inspect blissful_hopper

[
  {
    "Id": "35505f7810d17291261a43391d4b6c0846594d415ce4f4d0a6ffbf9cc5109048",
    "Name": "/blissful_hopper",
    "Path": "python",
    "Args": [
      "app.py"
    ],
    "State": {
      "Status": "running",
      "Running": true,
    },
    "Mounts": [],
    "Config": {
      "Entrypoint": [
        "python",
        "app.py"
      ],
    },
    "NetworkSettings": {...}
  }
]
```

To view logs of container we ran in background .
Docker logs container id / name

Container Logs

```
docker logs blissful_hopper
```

This is a sample web application that displays a colored background.
A color can be specified in two ways.

1. As a command line argument with `--color` as the argument. Accepts one of red,green,blue,blue2,pink,darkblue
 2. As an Environment variable `APP_COLOR`. Accepts one of red,green,blue,blue2,pink,darkblue
 3. If none of the above then a random color is picked from the above list.
- Note: Command line argument precedes over environment variable.

No command line argument or environment variable. Picking a Random Color =blue

- * Serving Flask app "app" (lazy loading)
- * Environment: production
- WARNING: Do not use the development server in a production environment.
Use a production WSGI server instead.
- * Debug mode: off
- * Running on http://0.0.0.0:8080/ (Press CTRL+C to quit)

```
$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
351176629d5a	nginx:alpine	"/docker-entrypoint..."	26 seconds ago	Up 23 seconds	0.0.0.0:3456->3456/tcp, 0.0.0.0:38080->80/tcp	gallant_wu

```
$
```

Let us first inspect the environment. How many containers are running on this host? 1

What is the image used by the container?

Nginx:alpine

How many ports are published on this container? 2

Which of the below ports are the exposed on the CONTAINER? 3456 & 80

Which of the below ports are published on Host? 38080 & 3456

Run an instance of kodecloud/simple-webapp with a tag blue and map port 8080 on the container to 38282 on the host.

```
$ docker run -p 38282:8080 kodecloud/simple-webapp:blue
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: Do not use the development server in a production environment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:8080/ (Press CTRL+C to quit)
```