Department of Computer Engineering
Academic Term: Jan-May 2022
Class: BE COMPUTERS

Subject Name: CLOUD COMPUTING LAB
Subject Code: CSL803

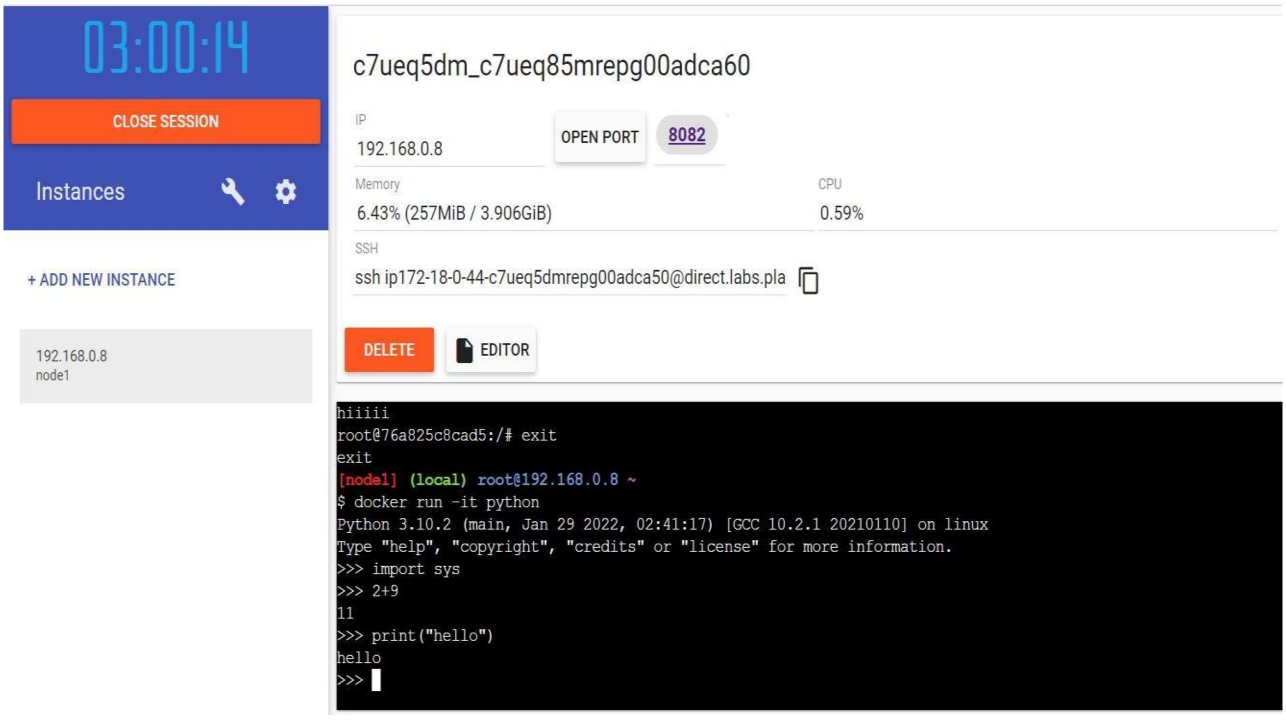
Practical No:	03
Title:	Containerization: Docker
Date of Performance:	28/01/22
Date of Submission:	09/02/2022
Roll No:	8626
Name of the Student:	Divita Phadakale

Evaluation:

Sr. No	Rubric	Grade
1	On time submission(2)	
2	Preparedness(2)	
3	Output(2)	
4	Post Lab Questions (4)	
	TOTAL	

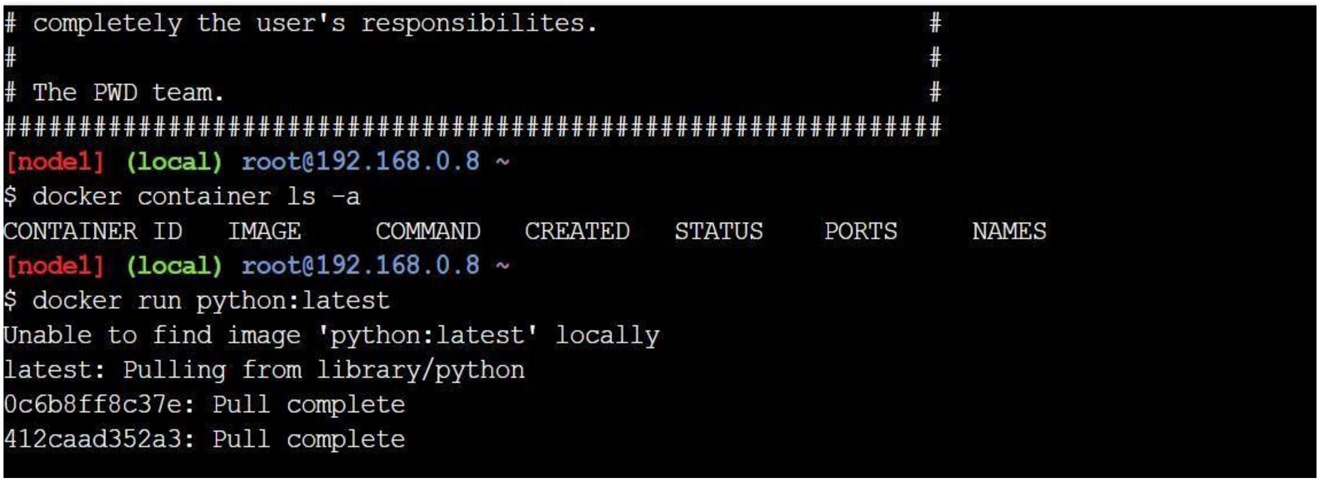
Signature of the Teacher:

To study and Implement Containerization using Docker

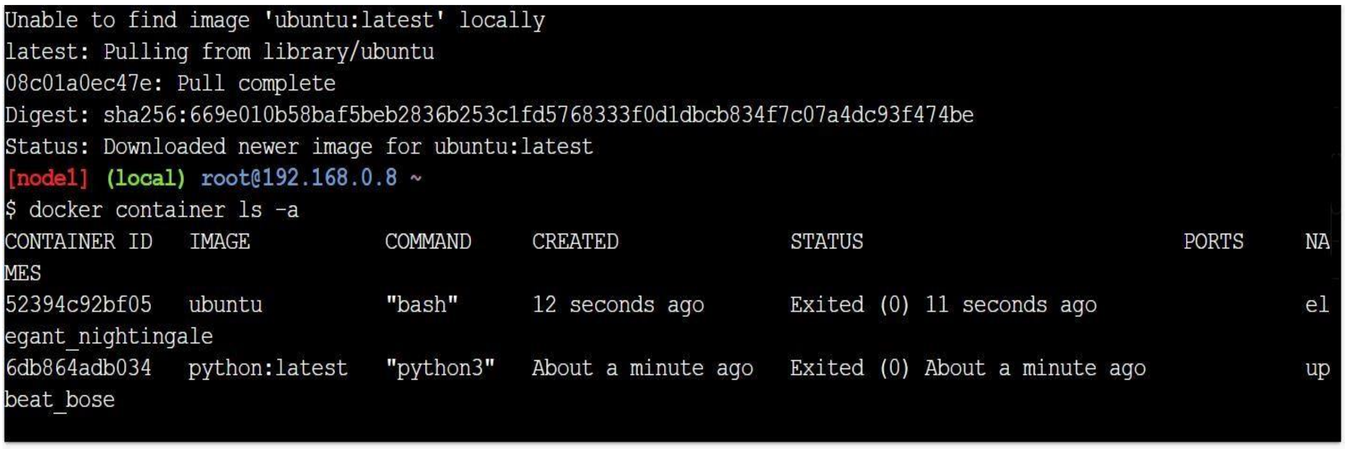


```
hiiiii
root@76a825c8cad5:/# exit
exit
[node1] (local) root@192.168.0.8 ~
$ docker run -it python
Python 3.10.2 (main, Jan 29 2022, 02:41:17) [GCC 10.2.1 20210110] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import sys
>>> 2+9
11
>>> print("hello")
hello
>>>
```

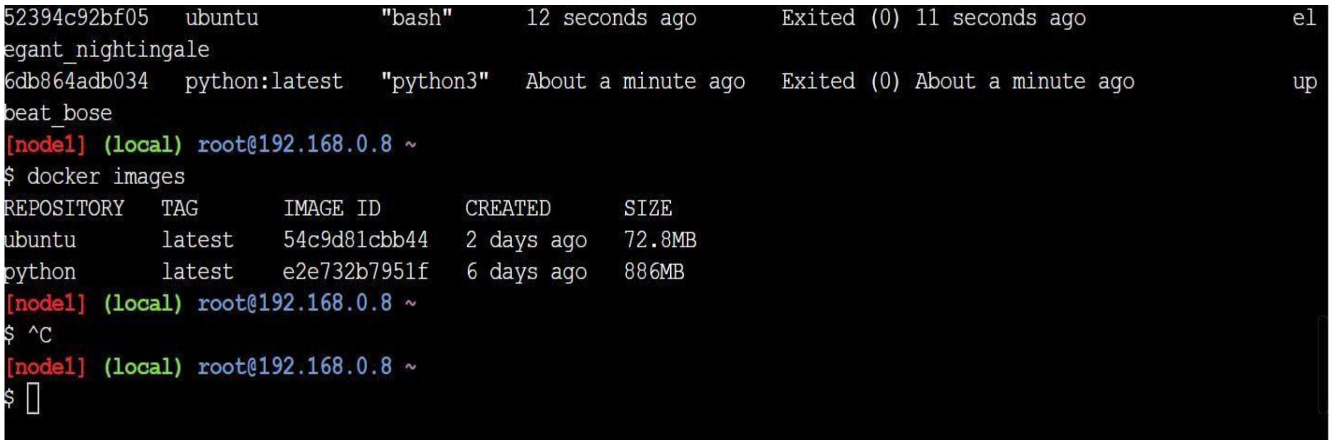
docker container ls -a



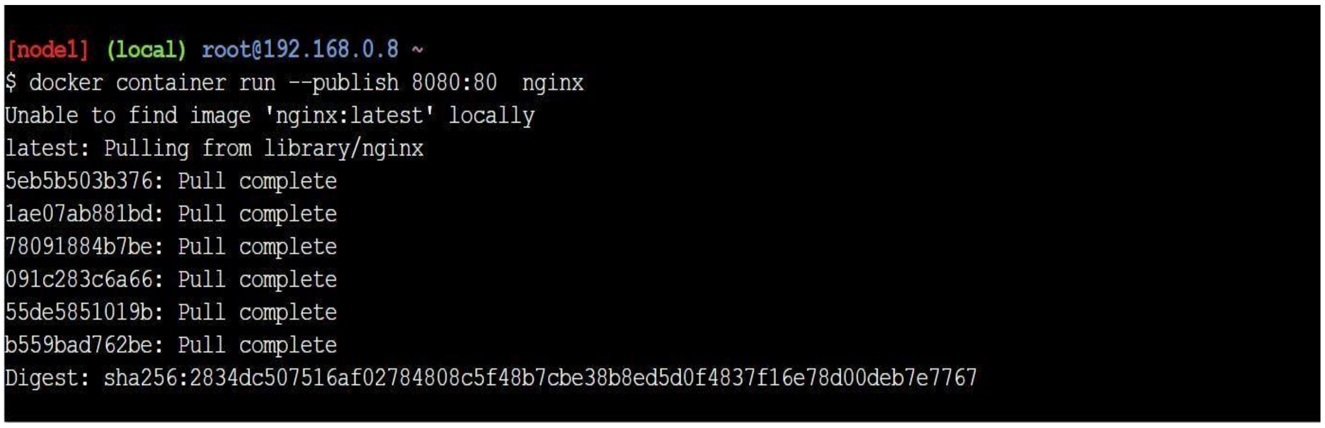
```
# completely the user's responsibilites.                          #
#                                                                 #
# The PWD team.                                                   #
###################################################################
[node1] (local) root@192.168.0.8 ~
$ docker container ls -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
[node1] (local) root@192.168.0.8 ~
$ docker run python:latest
Unable to find image 'python:latest' locally
latest: Pulling from library/python
0c6b8ff8c37e: Pull complete
412caad352a3: Pull complete
```

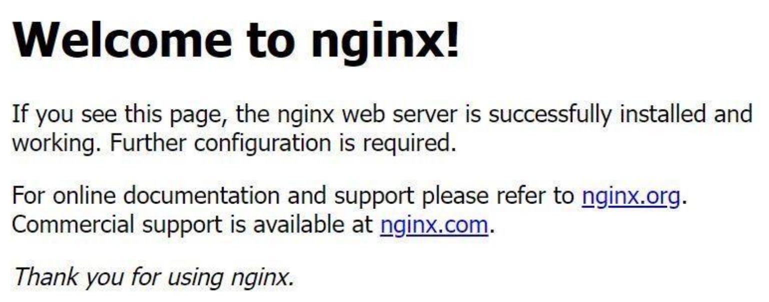

```
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
08c01a0ec47e: Pull complete
Digest: sha256:669e010b58baf5beb2836b253c1fd5768333f0d1dbcb834f7c07a4dc93f474be
Status: Downloaded newer image for ubuntu:latest
[node1] (local) root@192.168.0.8 ~
$ docker container ls -a
CONTAINER ID   IMAGE           COMMAND     CREATED              STATUS                          PORTS     NA
MES
52394c92bf05   ubuntu          "bash"      12 seconds ago       Exited (0) 11 seconds ago                 el
egant_nightingale
6db864adb034   python:latest   "python3"   About a minute ago   Exited (0) About a minute ago             up
beat_bose
```

Docker images


```
52394c92bf05   ubuntu          "bash"      12 seconds ago       Exited (0) 11 seconds ago                 el
egant_nightingale
6db864adb034   python:latest   "python3"   About a minute ago   Exited (0) About a minute ago             up
beat_bose
[node1] (local) root@192.168.0.8 ~
$ docker images
REPOSITORY   TAG       IMAGE ID       CREATED      SIZE
ubuntu       latest    54c9d81cbb44   2 days ago   72.8MB
python       latest    e2e732b7951f   6 days ago   886MB
[node1] (local) root@192.168.0.8 ~
$ ^C
[node1] (local) root@192.168.0.8 ~
$ 
```

docker container run --publish 8080:80 nginx


```
[node1] (local) root@192.168.0.8 ~
$ docker container run --publish 8080:80  nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
5eb5b503b376: Pull complete
1ae07ab881bd: Pull complete
78091884b7be: Pull complete
091c283c6a66: Pull complete
55de5851019b: Pull complete
b559bad762be: Pull complete
Digest: sha256:2834dc507516af02784808c5f48b7cbe38b8ed5d0f4837f16e78d00deb7e7767
```

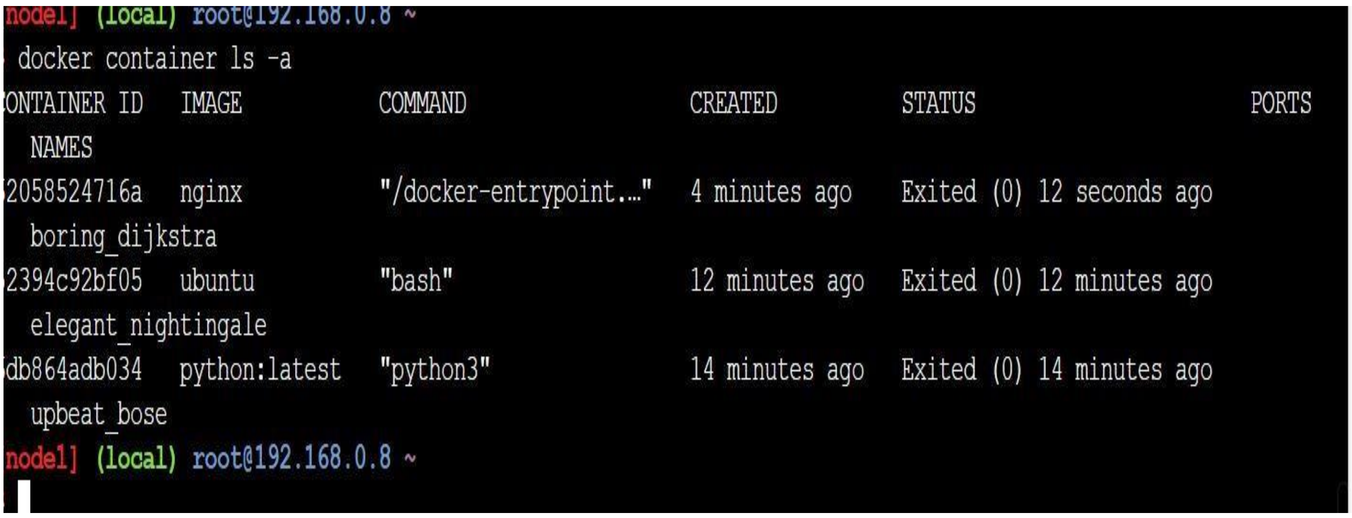


Welcome to nginx!

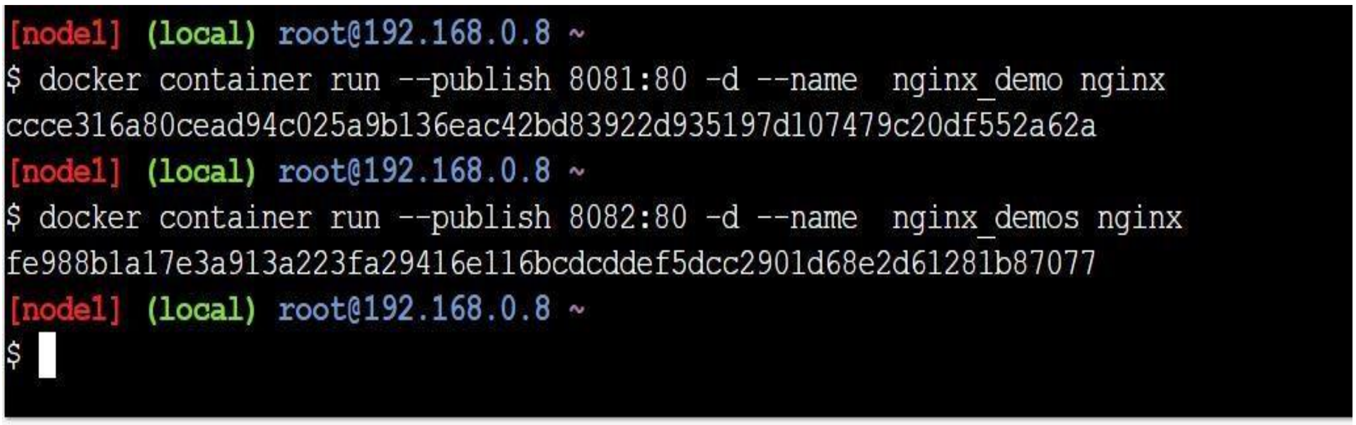
If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

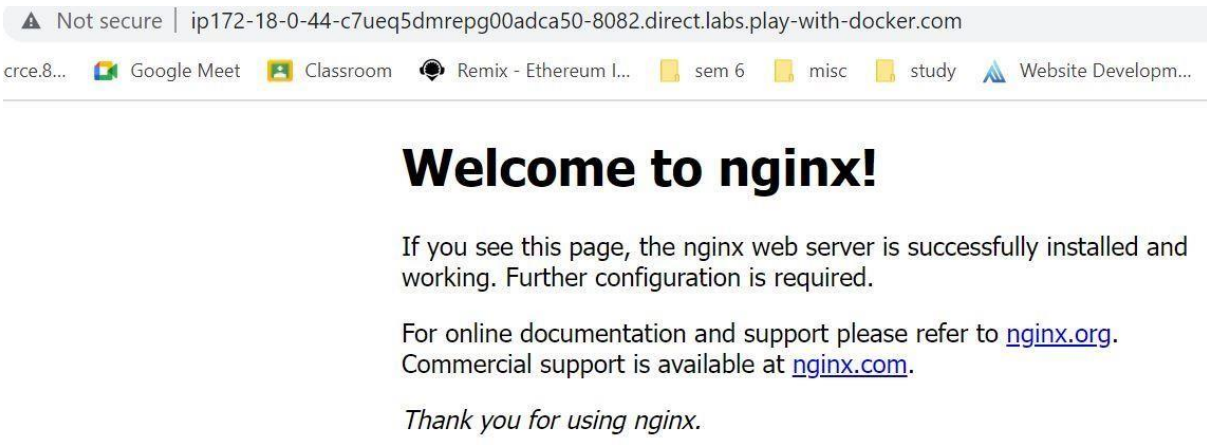
Thank you for using nginx.



```
node1] (local) root@192.168.0.8 ~
 docker container ls -a
ONTAINER ID   IMAGE           COMMAND                  CREATED          STATUS                      PORTS
  NAMES
2058524716a   nginx           "/docker-entrypoint.…"   4 minutes ago    Exited (0) 12 seconds ago
  boring_dijkstra
2394c92bf05   ubuntu          "bash"                   12 minutes ago   Exited (0) 12 minutes ago
  elegant_nightingale
db864adb034   python:latest   "python3"                14 minutes ago   Exited (0) 14 minutes ago
  upbeat_bose
node1] (local) root@192.168.0.8 ~
```



```
[node1] (local) root@192.168.0.8 ~
$ docker container run --publish 8081:80 -d --name  nginx_demo nginx
ccce316a80cead94c025a9b136eac42bd83922d935197d107479c20df552a62a
[node1] (local) root@192.168.0.8 ~
$ docker container run --publish 8082:80 -d --name  nginx_demos nginx
fe988b1a17e3a913a223fa29416e116bcdcddef5dcc2901d68e2d61281b87077
[node1] (local) root@192.168.0.8 ~
$
```

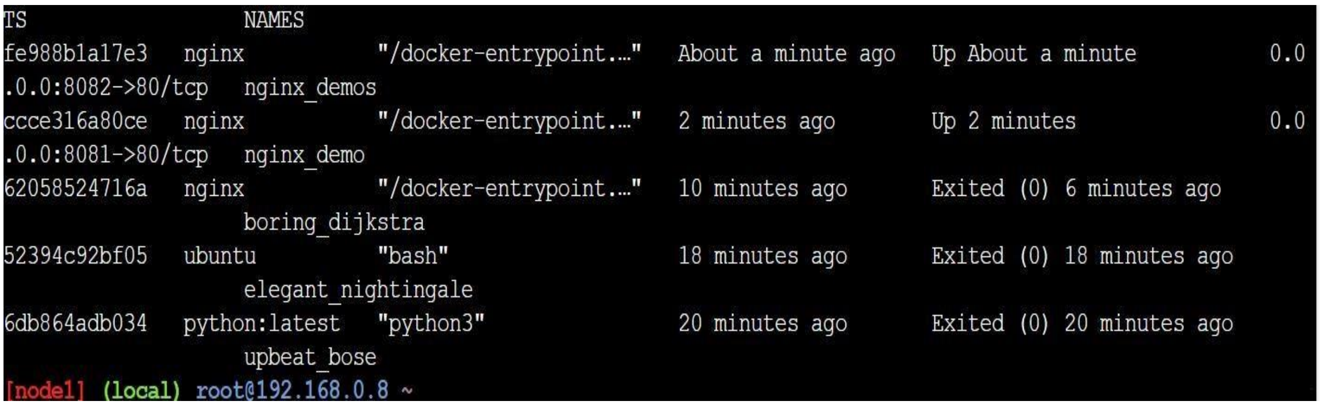

Not secure | ip172-18-0-44-c7ueq5dmrepg00adca50-8082.direct.labs.play-with-docker.com
crce.8...
Google Meet
Classroom
Remix - Ethereum I...
sem 6
misc
study
Website Developm...

Welcome to nginx!

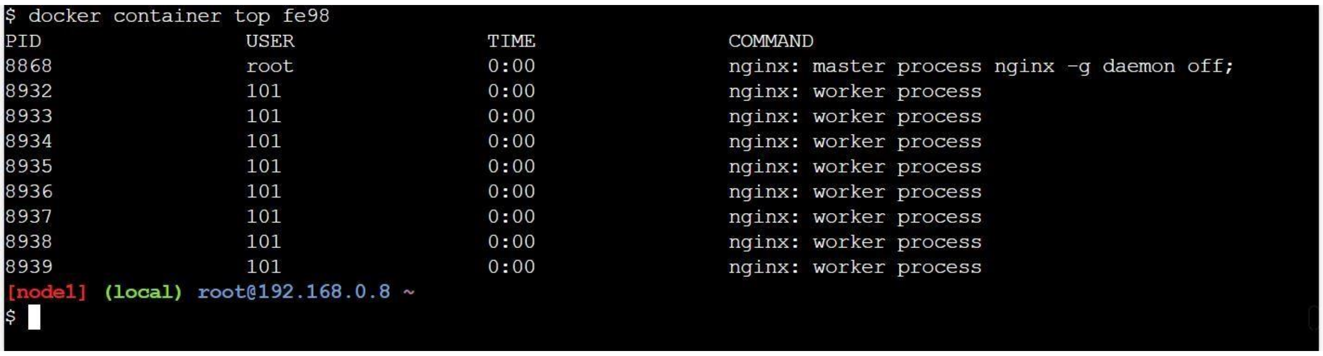
If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

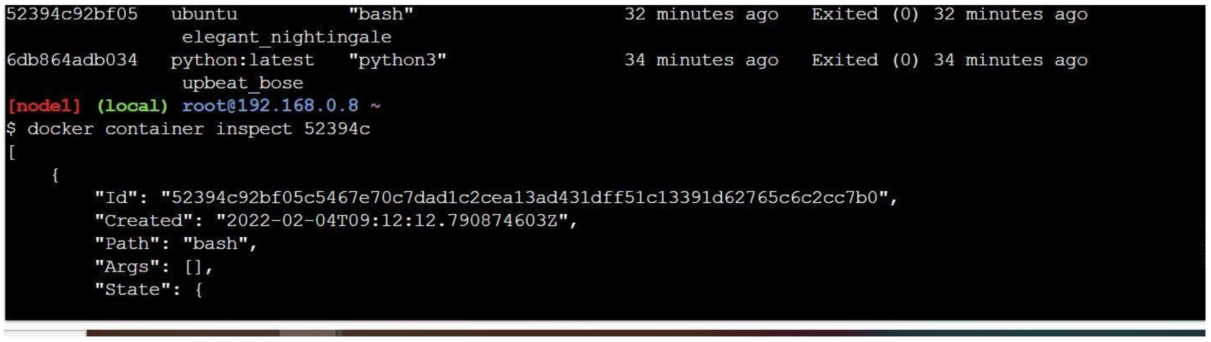
Thank you for using nginx.



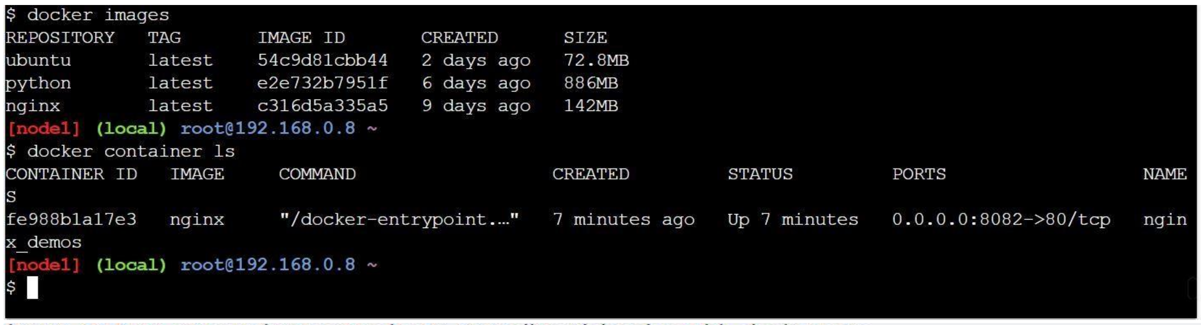
```
TS                  NAMES
fe988b1a17e3   nginx          "/docker-entrypoint.…"   About a minute ago   Up About a minute          0.0
.0.0:8082->80/tcp   nginx_demos
ccce316a80ce   nginx          "/docker-entrypoint.…"   2 minutes ago        Up 2 minutes               0.0
.0.0:8081->80/tcp   nginx_demo
62058524716a   nginx          "/docker-entrypoint.…"   10 minutes ago       Exited (0) 6 minutes ago
                    boring_dijkstra
52394c92bf05   ubuntu         "bash"                   18 minutes ago       Exited (0) 18 minutes ago
                    elegant_nightingale
6db864adb034   python:latest  "python3"                20 minutes ago       Exited (0) 20 minutes ago
                    upbeat_bose
[node1] (local) root@192.168.0.8 ~
```



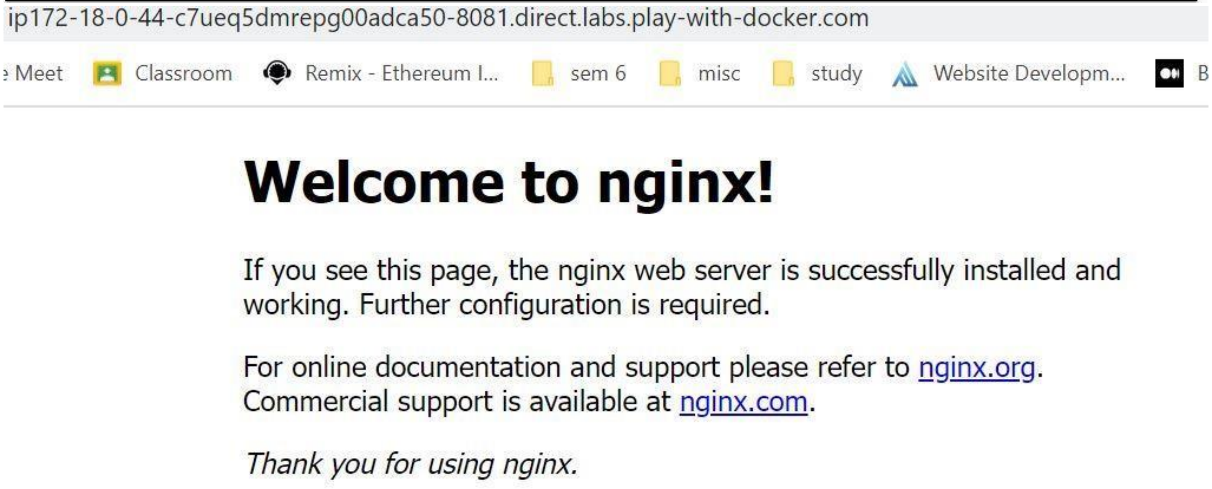
```
$ docker container top fe98
PID                 USER                TIME                COMMAND
8868                root                0:00                nginx: master process nginx -g daemon off;
8932                101                 0:00                nginx: worker process
8933                101                 0:00                nginx: worker process
8934                101                 0:00                nginx: worker process
8935                101                 0:00                nginx: worker process
8936                101                 0:00                nginx: worker process
8937                101                 0:00                nginx: worker process
8938                101                 0:00                nginx: worker process
8939                101                 0:00                nginx: worker process
[node1] (local) root@192.168.0.8 ~
$ 
```



```
52394c92bf05   ubuntu          "bash"                   32 minutes ago   Exited (0) 32 minutes ago
                elegant_nightingale
6db864adb034   python:latest   "python3"                34 minutes ago   Exited (0) 34 minutes ago
                upbeat_bose
[node1] (local) root@192.168.0.8 ~
$ docker container inspect 52394c
[
    {
        "Id": "52394c92bf05c5467e70c7dad1c2cea13ad431dff51c13391d62765c6c2cc7b0",
        "Created": "2022-02-04T09:12:12.790874603Z",
        "Path": "bash",
        "Args": [],
        "State": {
```



```
$ docker images
REPOSITORY   TAG       IMAGE ID       CREATED      SIZE
ubuntu       latest    54c9d81cbb44   2 days ago   72.8MB
python       latest    e2e732b7951f   6 days ago   886MB
nginx        latest    c316d5a335a5   9 days ago   142MB
[node1] (local) root@192.168.0.8 ~
$ docker container ls
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS                  NAME
S
fe988b1a17e3   nginx     "/docker-entrypoint.…"   7 minutes ago   Up 7 minutes   0.0.0.0:8082->80/tcp   ngin
x_demos
[node1] (local) root@192.168.0.8 ~
$ 
```



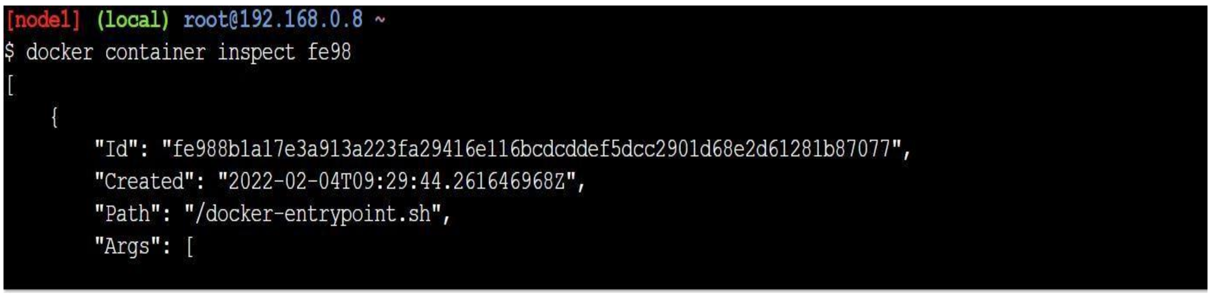
ip172-18-0-44-c7ueq5dmrepg00adca50-8081.direct.labs.play-with-docker.com

Welcome to nginx!

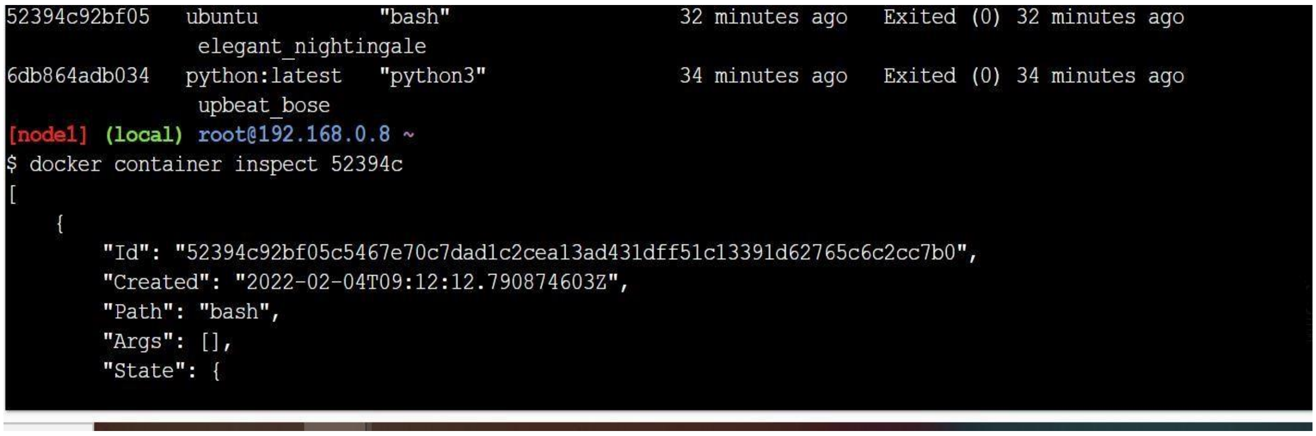
If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

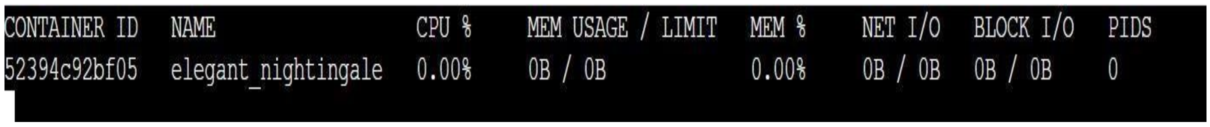
Thank you for using nginx.



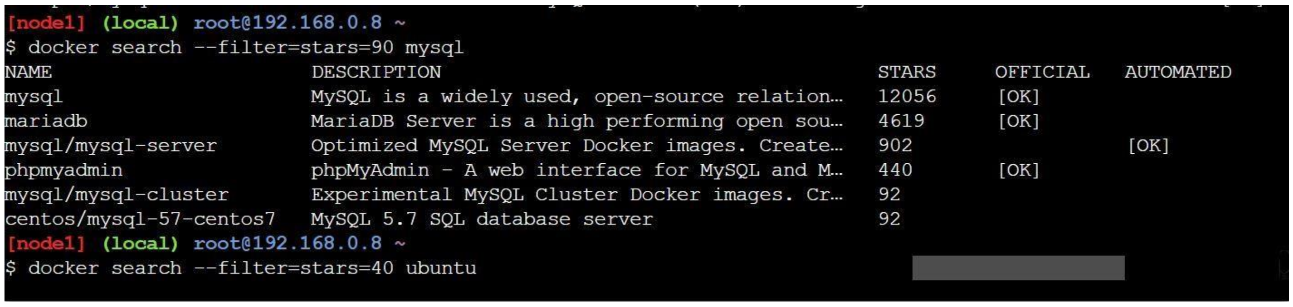
```
[node1] (local) root@192.168.0.8 ~
$ docker container inspect fe98
[
    {
        "Id": "fe988b1a17e3a913a223fa29416e116bcdcddef5dcc2901d68e2d61281b87077",
        "Created": "2022-02-04T09:29:44.261646968Z",
        "Path": "/docker-entrypoint.sh",
        "Args": [
```



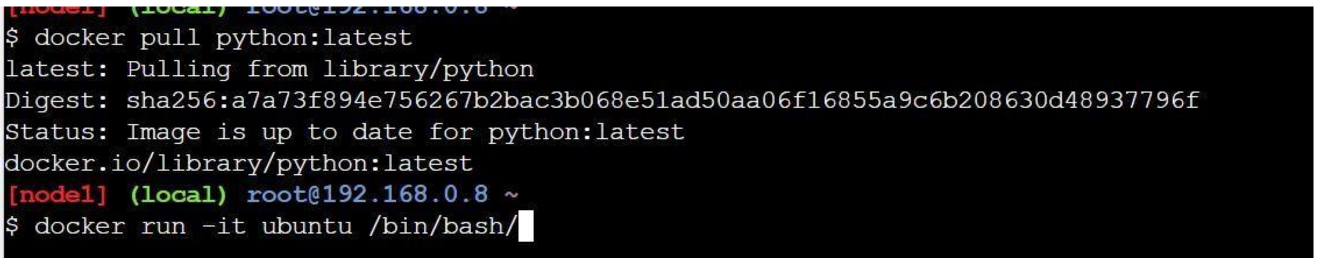
```
52394c92bf05   ubuntu          "bash"                    32 minutes ago   Exited (0) 32 minutes ago
                elegant_nightingale
6db864adb034   python:latest   "python3"                 34 minutes ago   Exited (0) 34 minutes ago
                upbeat_bose
[node1] (local) root@192.168.0.8 ~
$ docker container inspect 52394c
[
    {
        "Id": "52394c92bf05c5467e70c7dad1c2cea13ad431dff51c13391d62765c6c2cc7b0",
        "Created": "2022-02-04T09:12:12.790874603Z",
        "Path": "bash",
        "Args": [],
        "State": {
```



```
CONTAINER ID   NAME                 CPU %    MEM USAGE / LIMIT   MEM %    NET I/O   BLOCK I/O   PIDS
52394c92bf05   elegant_nightingale  0.00%    0B / 0B             0.00%    0B / 0B   0B / 0B     0
```

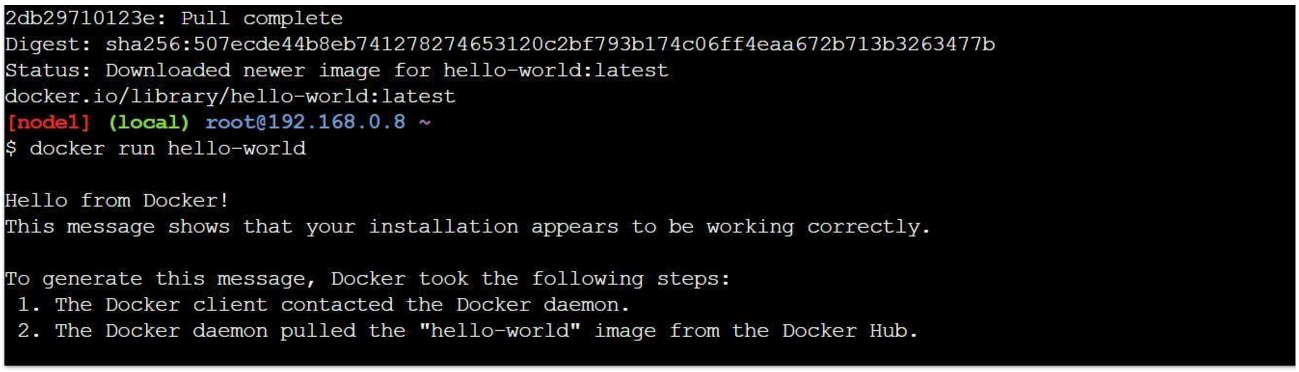


```
[node1] (local) root@192.168.0.8 ~
$ docker search --filter=stars=90 mysql
NAME                       DESCRIPTION                                     STARS     OFFICIAL   AUTOMATED
mysql                      MySQL is a widely used, open-source relation…   12056     [OK]
mariadb                    MariaDB Server is a high performing open sou…   4619      [OK]
mysql/mysql-server         Optimized MySQL Server Docker images. Create…   902                  [OK]
phpmyadmin                 phpMyAdmin - A web interface for MySQL and M…   440       [OK]
mysql/mysql-cluster        Experimental MySQL Cluster Docker images. Cr…   92
centos/mysql-57-centos7    MySQL 5.7 SQL database server                   92
[node1] (local) root@192.168.0.8 ~
$ docker search --filter=stars=40 ubuntu
```

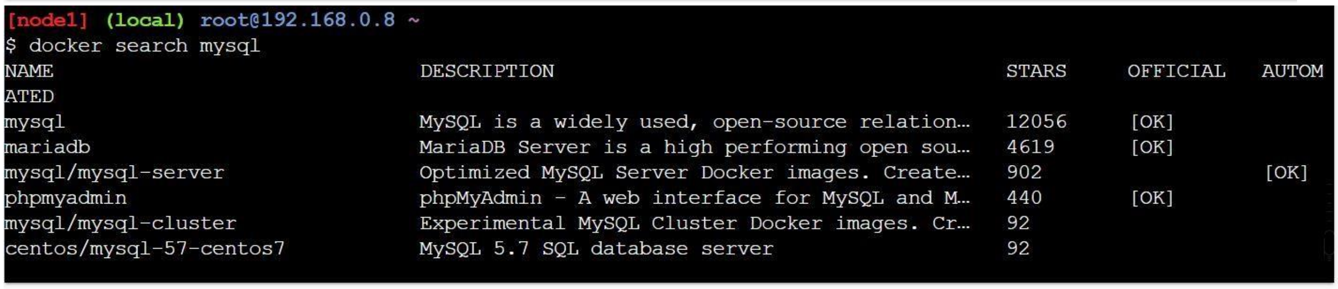


```
$ docker pull python:latest
latest: Pulling from library/python
Digest: sha256:a7a73f894e756267b2bac3b068e51ad50aa06f16855a9c6b208630d48937796f
Status: Image is up to date for python:latest
docker.io/library/python:latest
[node1] (local) root@192.168.0.8 ~
$ docker run -it ubuntu /bin/bash/
```

Docker run hello-world

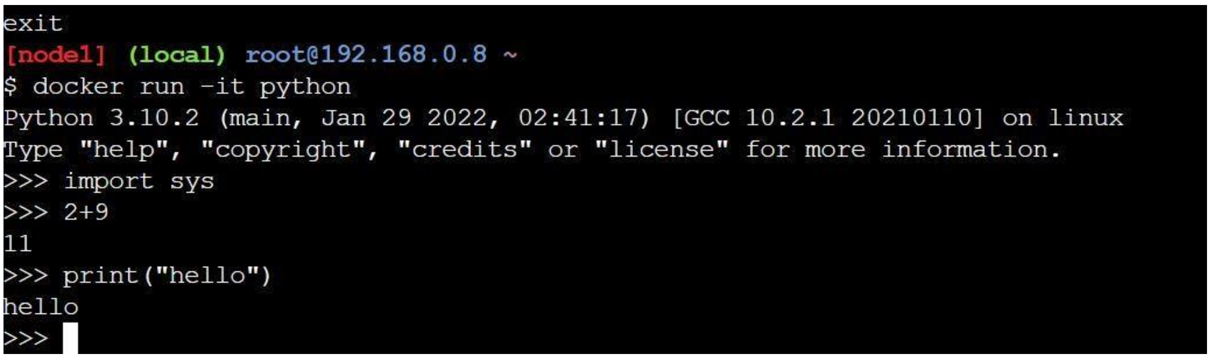


```
2db29710123e: Pull complete
Digest: sha256:507ecde44b8eb741278274653120c2bf793b174c06ff4eaa672b713b3263477b
Status: Downloaded newer image for hello-world:latest
docker.io/library/hello-world:latest
[node1] (local) root@192.168.0.8 ~
$ docker run hello-world

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.
```



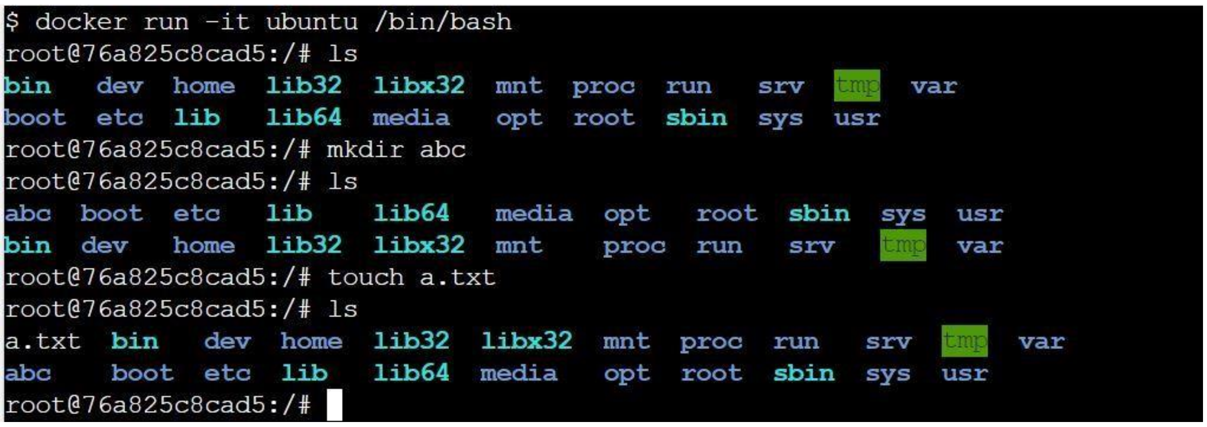
```
[node1] (local) root@192.168.0.8 ~
$ docker search mysql
NAME                                DESCRIPTION                                     STARS     OFFICIAL   AUTOM
ATED
mysql                               MySQL is a widely used, open-source relation…   12056     [OK]
mariadb                             MariaDB Server is a high performing open sou…   4619      [OK]
mysql/mysql-server                  Optimized MySQL Server Docker images. Create…   902                  [OK]
phpmyadmin                          phpMyAdmin - A web interface for MySQL and M…   440       [OK]
mysql/mysql-cluster                 Experimental MySQL Cluster Docker images. Cr…   92
centos/mysql-57-centos7             MySQL 5.7 SQL database server                   92
```

Docker run -it python

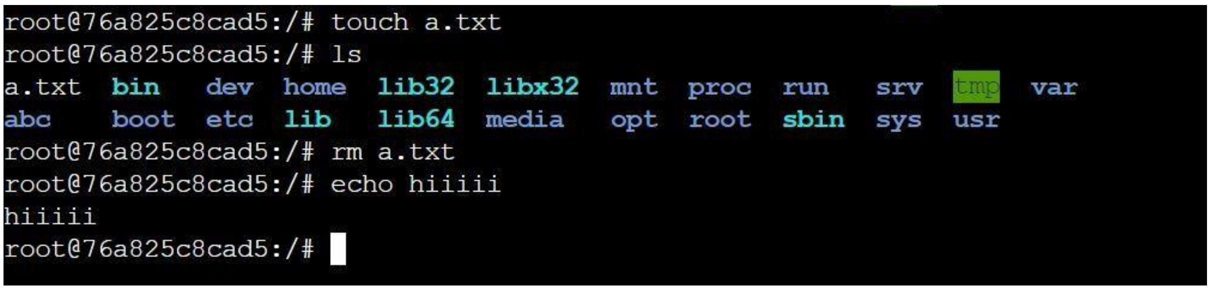


```
exit
[node1] (local) root@192.168.0.8 ~
$ docker run -it python
Python 3.10.2 (main, Jan 29 2022, 02:41:17) [GCC 10.2.1 20210110] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import sys
>>> 2+9
11
>>> print("hello")
hello
>>>
```

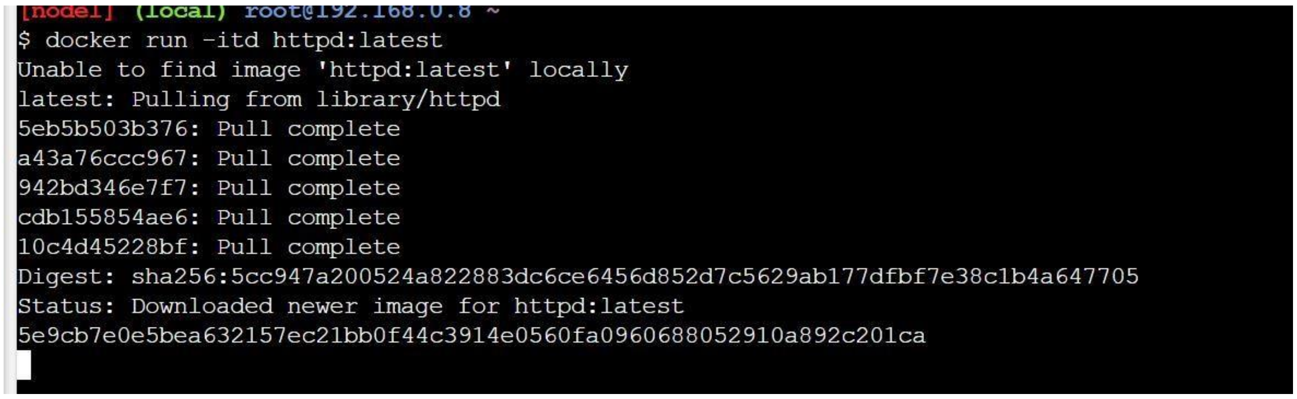
Docker run -it ubuntu /bin/bash



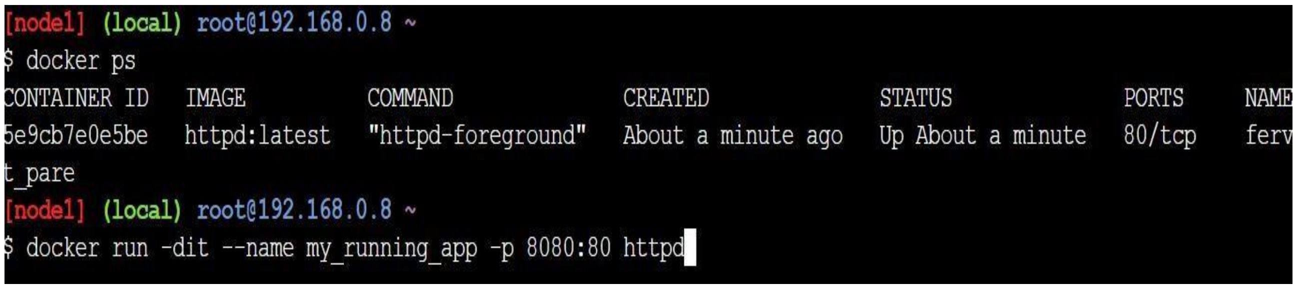
```
$ docker run -it ubuntu /bin/bash
root@76a825c8cad5:/# ls
bin   dev  home  lib32  libx32  mnt  proc  run   srv  tmp  var
boot  etc  lib   lib64  media   opt  root  sbin  sys  usr
root@76a825c8cad5:/# mkdir abc
root@76a825c8cad5:/# ls
abc  boot  etc   lib    lib64   media  opt   root  sbin  sys  usr
bin  dev   home  lib32  libx32  mnt    proc  run   srv   tmp  var
root@76a825c8cad5:/# touch a.txt
root@76a825c8cad5:/# ls
a.txt  bin   dev  home  lib32  libx32  mnt  proc  run   srv  tmp  var
abc    boot  etc  lib   lib64  media   opt  root  sbin  sys  usr
root@76a825c8cad5:/#
```



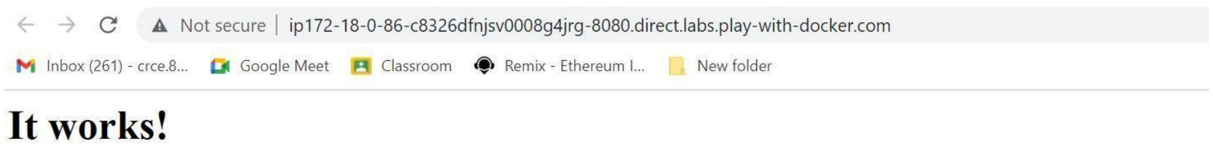
```
root@76a825c8cad5:/# touch a.txt
root@76a825c8cad5:/# ls
a.txt  bin   dev  home  lib32  libx32  mnt  proc  run   srv  tmp  var
abc    boot  etc  lib   lib64  media   opt  root  sbin  sys  usr
root@76a825c8cad5:/# rm a.txt
root@76a825c8cad5:/# echo hiiiii
hiiiii
root@76a825c8cad5:/#
```



```
[node1] (local) root@192.168.0.8 ~
$ docker run -itd httpd:latest
Unable to find image 'httpd:latest' locally
latest: Pulling from library/httpd
5eb5b503b376: Pull complete
a43a76ccc967: Pull complete
942bd346e7f7: Pull complete
cdb155854ae6: Pull complete
10c4d45228bf: Pull complete
Digest: sha256:5cc947a200524a822883dc6ce6456d852d7c5629ab177dfbf7e38c1b4a647705
Status: Downloaded newer image for httpd:latest
5e9cb7e0e5bea632157ec21bb0f44c3914e0560fa0960688052910a892c201ca
```



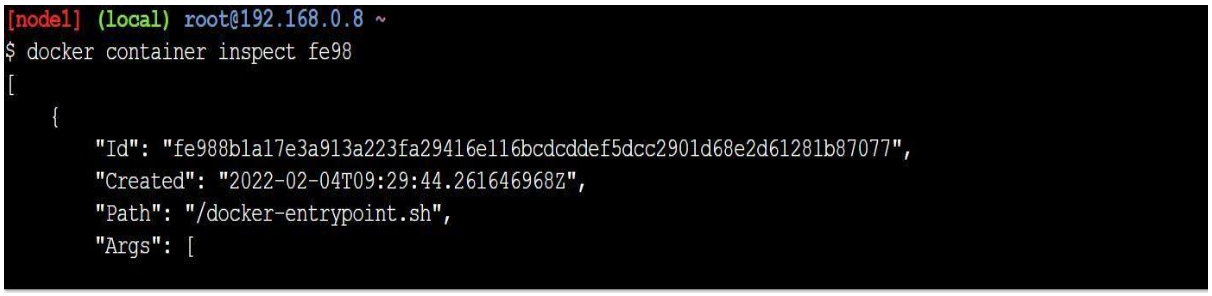
```
[node1] (local) root@192.168.0.8 ~
$ docker ps
CONTAINER ID   IMAGE          COMMAND              CREATED              STATUS              PORTS     NAME
5e9cb7e0e5be   httpd:latest   "httpd-foreground"   About a minute ago   Up About a minute   80/tcp    ferv
t_pare
[node1] (local) root@192.168.0.8 ~
$ docker run -dit --name my_running_app -p 8080:80 httpd
```



Not secure | ip172-18-0-86-c8326dfnjsv0008g4jrg-8080.direct.labs.play-with-docker.com

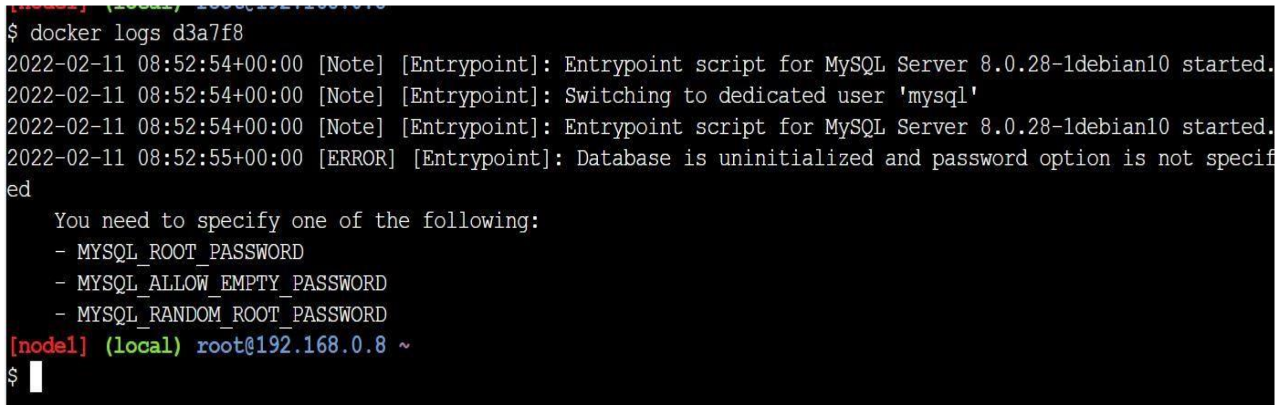
Inbox (261) - crce.8... Google Meet Classroom Remix - Ethereum I... New folder

It works!



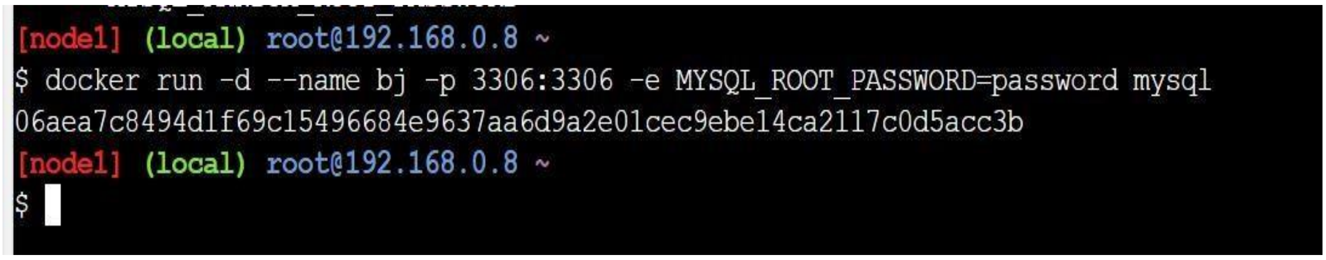
```
[node1] (local) root@192.168.0.8 ~
$ docker container inspect fe98
[
    {
        "Id": "fe988b1a17e3a913a223fa29416e116bcdcddef5dcc2901d68e2d61281b87077",
        "Created": "2022-02-04T09:29:44.261646968Z",
        "Path": "/docker-entrypoint.sh",
        "Args": [
```

Docker logs

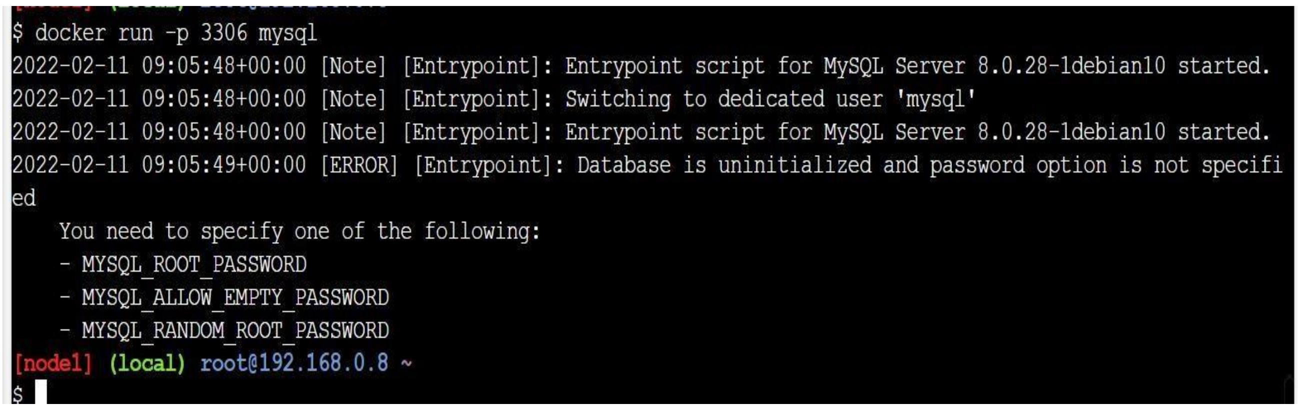


```
$ docker logs d3a7f8
2022-02-11 08:52:54+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.28-1debian10 started.
2022-02-11 08:52:54+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
2022-02-11 08:52:54+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.28-1debian10 started.
2022-02-11 08:52:55+00:00 [ERROR] [Entrypoint]: Database is uninitialized and password option is not specif
ed
    You need to specify one of the following:
    - MYSQL_ROOT_PASSWORD
    - MYSQL_ALLOW_EMPTY_PASSWORD
    - MYSQL_RANDOM_ROOT_PASSWORD
[node1] (local) root@192.168.0.8 ~
$
```

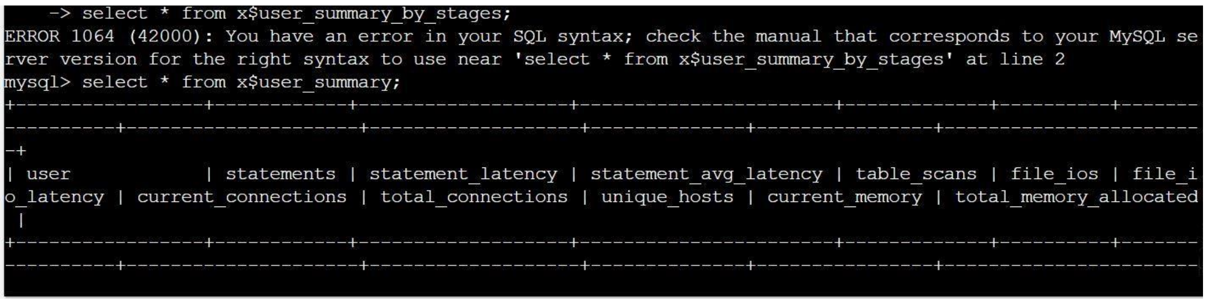
Mysql password



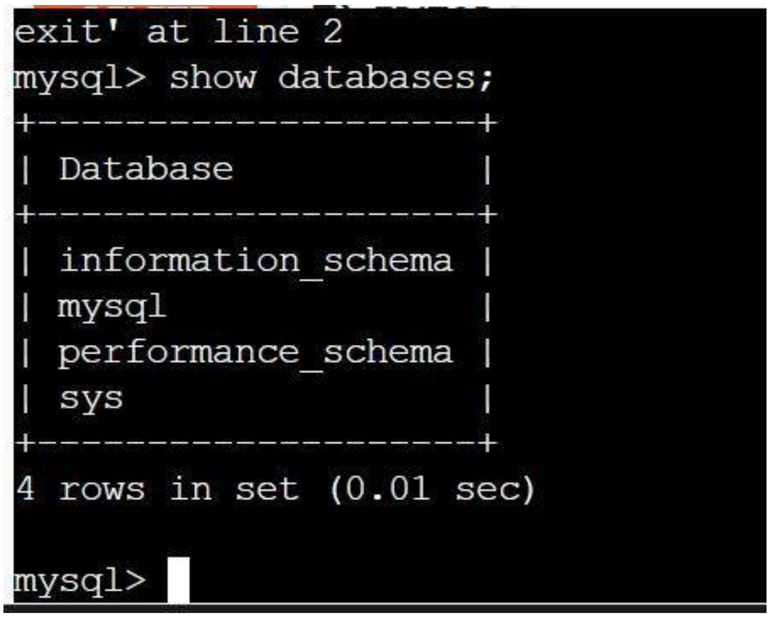
```
[node1] (local) root@192.168.0.8 ~
$ docker run -d --name bj -p 3306:3306 -e MYSQL_ROOT_PASSWORD=password mysql
06aea7c8494d1f69c15496684e9637aa6d9a2e01cec9ebe14ca2117c0d5acc3b
[node1] (local) root@192.168.0.8 ~
$
```



```
$ docker run -p 3306 mysql
2022-02-11 09:05:48+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.28-1debian10 started.
2022-02-11 09:05:48+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
2022-02-11 09:05:48+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.28-1debian10 started.
2022-02-11 09:05:49+00:00 [ERROR] [Entrypoint]: Database is uninitialized and password option is not specifi
ed
    You need to specify one of the following:
    - MYSQL_ROOT_PASSWORD
    - MYSQL_ALLOW_EMPTY_PASSWORD
    - MYSQL_RANDOM_ROOT_PASSWORD
[node1] (local) root@192.168.0.8 ~
$
```



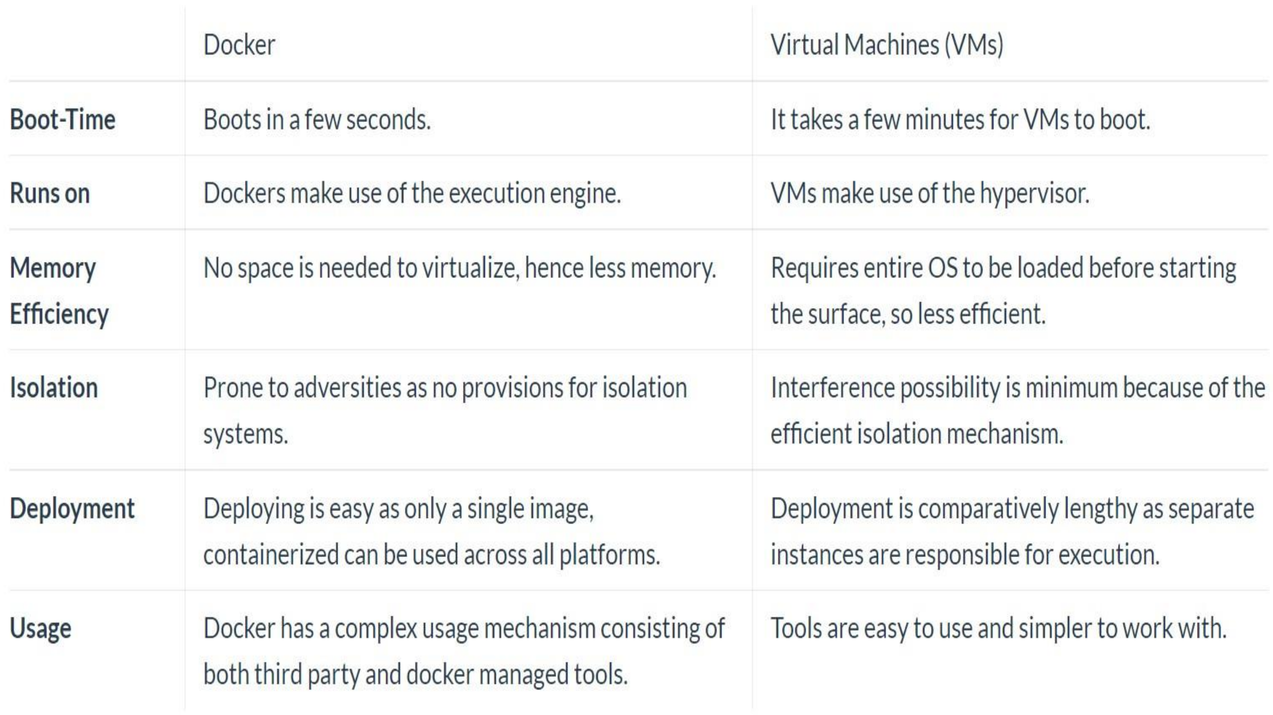
```
    -> select * from x$user_summary_by_stages;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL se
rver version for the right syntax to use near 'select * from x$user_summary_by_stages' at line 2
mysql> select * from x$user_summary;
+------------------+------------+-------------------+----------------------+--------------+----------+-------
----------+--------------------+-------------------+--------------+----------------+----------------------
-+
| user             | statements | statement_latency | statement_avg_latency | table_scans | file_ios | file_i
o_latency | current_connections | total_connections | unique_hosts | current_memory | total_memory_allocated
 |
+------------------+------------+-------------------+----------------------+--------------+----------+-------
----------+--------------------+-------------------+--------------+----------------+----------------------
```



```
exit' at line 2
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.01 sec)

mysql>
```

POSTLAB QUESTIONS:

1) Docker containerization and Virtualization?



	Docker	Virtual Machines (VMs)
Boot-Time	Boots in a few seconds.	It takes a few minutes for VMs to boot.
Runs on	Dockers make use of the execution engine.	VMs make use of the hypervisor.
Memory Efficiency	No space is needed to virtualize, hence less memory.	Requires entire OS to be loaded before starting the surface, so less efficient.
Isolation	Prone to adversities as no provisions for isolation systems.	Interference possibility is minimum because of the efficient isolation mechanism.
Deployment	Deploying is easy as only a single image, containerized can be used across all platforms.	Deployment is comparatively lengthy as separate instances are responsible for execution.
Usage	Docker has a complex usage mechanism consisting of both third party and docker managed tools.	Tools are easy to use and simpler to work with.

2) Image vs container.

Docker images and containers work together to let you unleash the full potential of the innovative Docker technology. However, they have subtle differences that may be difficult to notice, especially for a beginner.

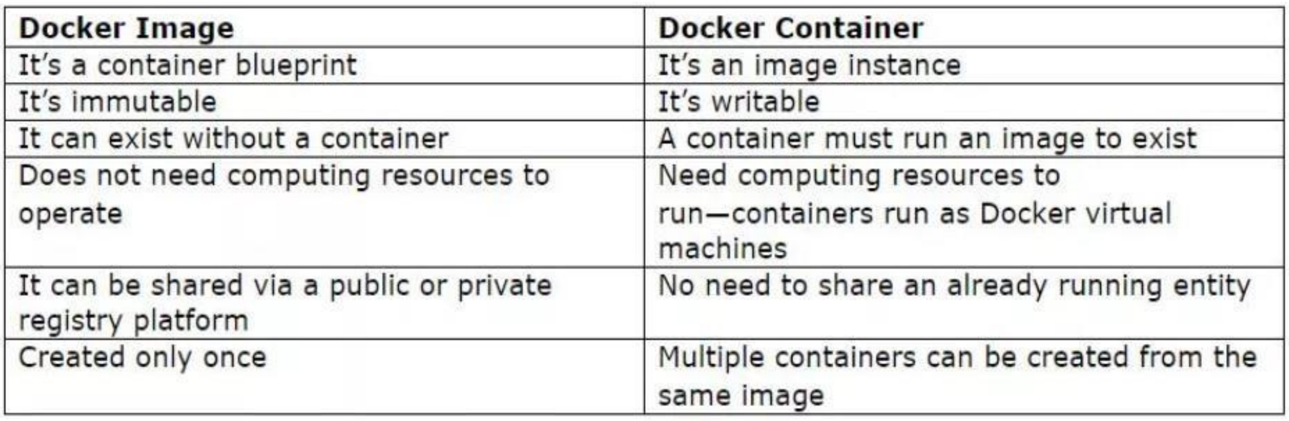
A simple analogy that compares their differences is to think of a Docker image as a recipe and a container as the cake prepared from that recipe.

The recipe sets out the instructions for baking the cake. You cannot enjoy eating the cake if you do not put the instructions into action.

You need to follow the recipe to prepare the cake and eat it. Similarly, you should follow the instructions in the Docker image to create and start a container, and enjoy the benefits of Docker.

You can bake as many cakes as possible from a single recipe—just like an image can create multiple containers. However, if you change the recipe, the taste of your existing cakes will not change.

Only newly baked cakes will use the modified recipe. Likewise, if you make changes to a container image, you'll not affect the already running containers.



Docker Image	Docker Container
It's a container blueprint	It's an image instance
It's immutable	It's writable
It can exist without a container	A container must run an image to exist
Does not need computing resources to operate	Need computing resources to run—containers run as Docker virtual machines
It can be shared via a public or private registry platform	No need to share an already running entity
Created only once	Multiple containers can be created from the same image

3) What is Dev server, Test Server and Production Server?

A **development server** is a type of server that is designed to facilitate the development and testing of programs, websites, software or applications for software programmers. It provides a run-time environment, as well as all hardware/software utilities that are essential to program debugging and development

The **Test Server** is a place where new updates, features, and mechanics are tested before being released to the main servers. Sometimes, these servers are in a closed-testing mode, meaning that only Developers and Testers can access them. Often, however, a large player base is required to test a new feature, and then the Test Server is open publicly for anyone to join.

A **production server** is a server used to host website content and applications for deployment to a live environment. It is the main server on which websites and Web applications are accessed by end users and is also referred to as a live server. A production server may be a dedicated machine, virtual server, basic PC or multiple machines dispersed geographically.

4) Docker hub vs Docker compose vs Docker file?

Docker Hub is a service provided by Docker for finding and sharing container images with your team. It is the world's largest repository of container images with an array of content sources including container community developers, open source projects and independent software vendors (ISV) building and distributing their code in containers.

Docker file is a simple text file that contains the commands a user could call to assemble an image whereas Docker Compose is a tool for defining and running multi-container Docker applications.

Docker Compose define the services that make up your app in docker-compose.yml so they can be run together in an isolated environment. It gets an app running in one command by just running Docker-compose up. Docker compose uses the Docker file if you add the build command to your project's docker-compose.yml. Your Docker workflow should be to build a suitable Docker file for each image you wish to create, then use compose to assemble the images using the build command.

5) Container port vs host port?

Container Port: A *container port* specifies a port within a container. This is only necessary as BRIDGE or USER part of a *port mapping* when using mode networking with a Docker container.

Host Port: A *host port* specifies a port on the host to bind to. When used with BRIDGE or USER mode networking, you specify a *port mapping* from a *host port* to a *container port*. In HOST networking, requested ports are *host ports* by default. Note that only *host ports* are made available to a task through environment variables.

6) Illustrate Docker advantages, disadvantages and applications?

Benefits of Docker

a. Return on Investment and Cost Savings

Docker's first advantage is ROI. Especially for large, established companies, which need to generate steady revenue over the long term, the solution is only better if it can drive down costs while raising profits.

b. Rapid Deployment

It can decrease deployment to seconds. It is because of the fact that it can create a container for every process and even does not boot an OS. So, even without worrying about the cost to bring it up again, it would be higher than what is affordable, Data can be created as well as destroyed.

c. Security

Docker makes sure that applications that are running on containers are completely segregated and isolated from each other, from a security point of view, by granting us complete control over traffic flow and management.

d. Simplicity and Faster Configurations

The way Docker simplifies the matters is one of the key benefits of it. It gives flexibility to users to take their own configuration, put that into the code, and further deploy it without any problems. However, the requirements of the infrastructure are no longer linked with the environment of the application, as Docker can be used in a wide variety of environments. e. CI Efficiency

With the help of a Docker, we can build a container image and can further use that same image over every step of the deployment process. The advantage of it is the ability to separate non-dependent steps and also run them in parallel. In addition, the duration of time it takes from build to production may speed up notably.

f. Continuous Integration

While it comes to Continuous Integration, Docker works well as part of its pipelines along with tools such as Travis, Jenkins, and Wercker. These tools can save the new version as a Docker image, every time our source code is updated, just tag it with a version number and push to Docker Hub, then deploy it to production.

Limitations of Docker

a. Missing features

There are a ton of feature requests are under progress, like container self-registration, and self-inspects, copying files from the host to the container, and many more.

b. Data in the container

There are times when a container goes down, so after that, it needs a backup and recovery strategy, although we have several solutions for that they are not automated or not very scalable yet.

c. Run applications as fast as a bare-metal serve

In comparison with the virtual machines, Docker containers have less overhead but not zero overhead. If we run, an application directly on a bare-metal server we get true bare-metal speed even without using containers or virtual machines. However, Containers don’t run at bare-metal speeds.

d. Provide cross-platform compatibility

The one major issue is if an application designed to run in a Docker container on Windows, then it can’t run on Linux or vice versa. However, Virtual machines are not subject to this limitation. So, this limitation makes Docker less attractive in some highly heterogeneous environments which are composed of both Windows and Linux servers.

e. Run applications with graphical interfaces

In general, Docker is designed for hosting applications which run on the command line. Though we have a few ways (like X11 forwarding) by which we can make it possible to run a graphical interface inside a Docker container, however, this is clunky. Hence we can say, for applications that require rich interfaces, Docker is not a good solution.

f. Solve all your security problems

In simple words, we need to evaluate the Docker-specific security risks and make sure we can handle them before moving workloads to Docker. The reason behind it is that Docker creates new security challenges like the difficulty of monitoring multiple moving pieces within a large-scale, dynamic Docker environment.