Docker Introduction

Getting Started With Docker

Topics

What is Docker?

Docker Engine

Concepts

Testing Docker

Docker and Python

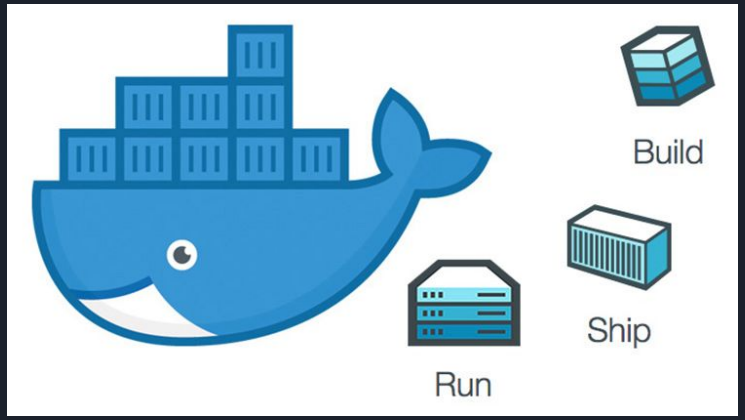
What Is Docker?

What is Docker

Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications. By taking advantage of Docker's methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production.

What is Docker

Basically, Docker is a tool that allows developers, sys-admins etc. to easily deploy their applications in a sandbox (called containers) to run on the host operating system i.e. Linux. Docker is a platform for developers and sysadmins to build, run, and share applications with containers. The use of containers to deploy applications is called containerization. Containers are not new, but their use for easily deploying applications is.



What is Docker

Containerization is increasingly popular because containers are:

- **Flexible**: Even the most complex applications can be containerized.
- **Lightweight**: Containers leverage and share the host kernel, making them much more efficient in terms of system resources than virtual machines.
- **Portable**: You can build locally, deploy to the cloud, and run anywhere.
- **Loosely coupled**: Containers are highly self sufficient and encapsulated, allowing you to replace or upgrade one without disrupting others.
- **Scalable**: You can increase and automatically distribute container replicas across a datacenter.
- **Secure**: Containers apply aggressive constraints and isolations to processes without any configuration required on the part of the user.

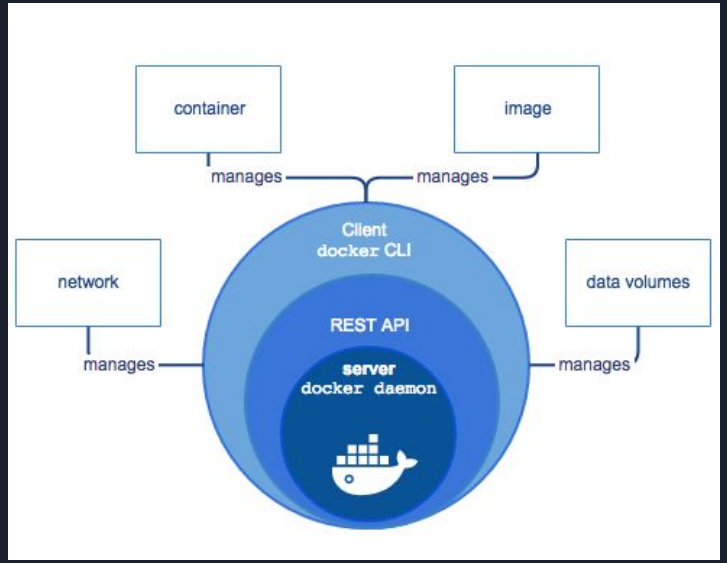
Docker Engine and Architecture

Docker Engine

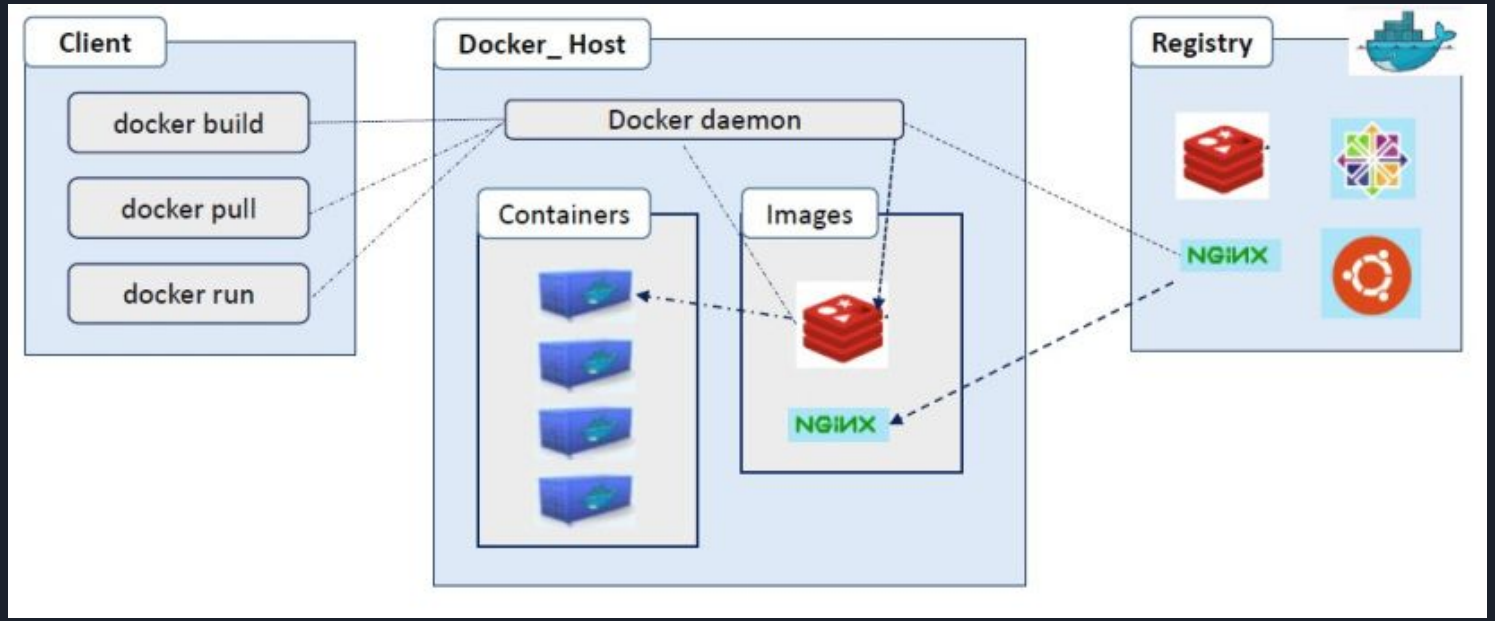
Docker Engine is a client-server application with these major components:

- A server which is a type of long-running program called a daemon process (the dockerd command).
- A REST API which specifies interfaces that programs can use to talk to the daemon and instruct it what to do.
- A command line interface (CLI) client (the docker command).

Docker Engine



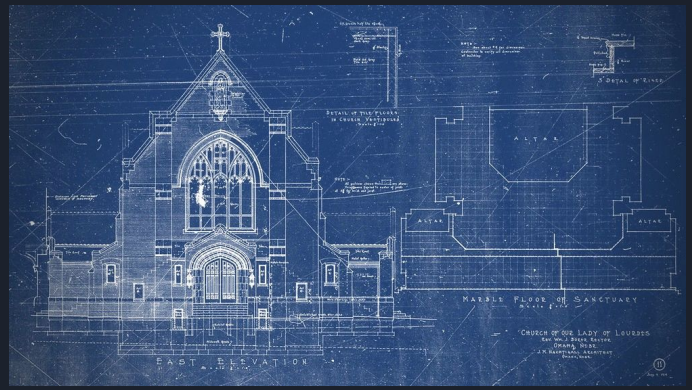
Docker Architecture



Docker Concepts

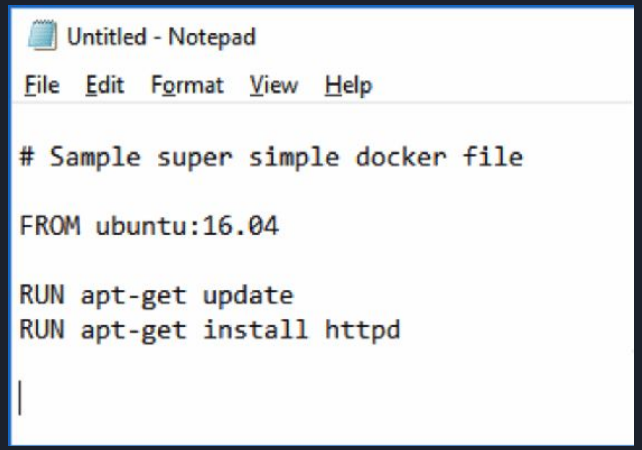
Docker Image

An image is a static representation of the app or service and its configuration and dependencies. A Docker image is built up from a series of layers, allowing a minimal amount of data to be sent when transferring images over network.



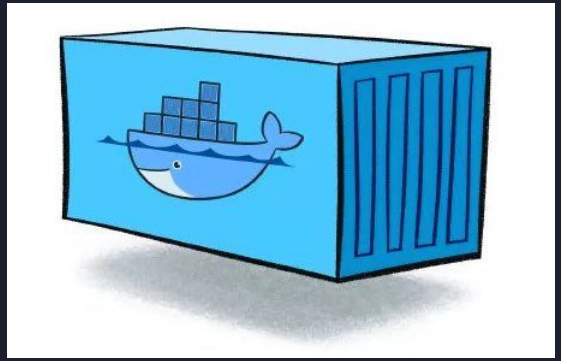
Dockerfile

A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image.

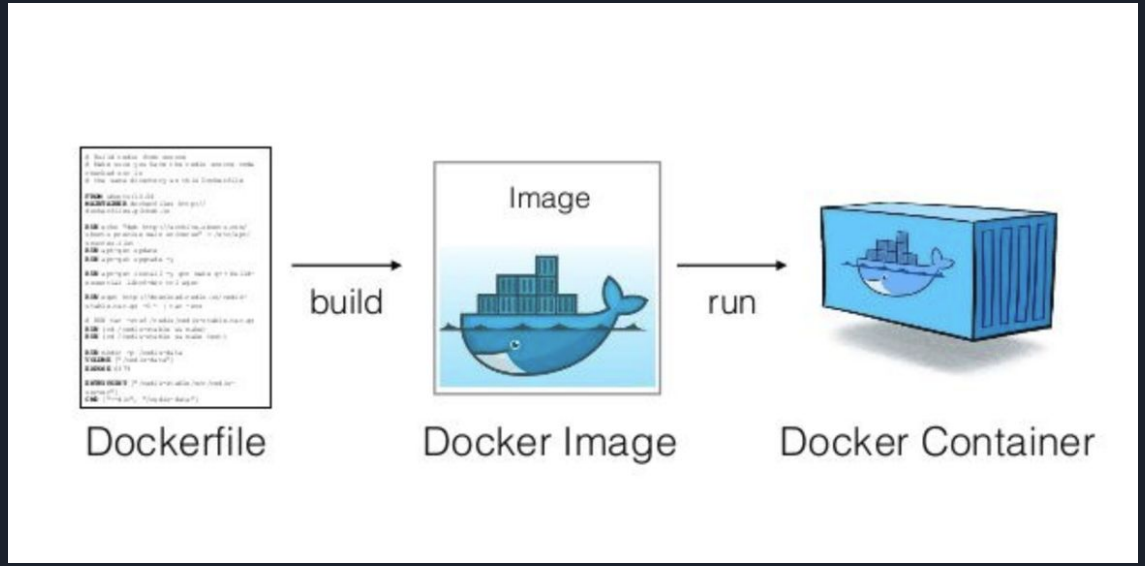


Docker Container

Container is runtime object or representation of an image. Containers are lightweight & portable encapsulation of an environment where applications are run.



Docker Pipeline



Testing Docker

Installation

Windows 10 Professional:

- Instalacion: https://www.youtube.com/watch?v=BK-C2RofmTE
- documentacion: https://docs.docker.com/docker-for-windows/

Windows 10 Home Edition:

- Instalacion: https://www.youtube.com/watch?v=YH3sutAsxEM
- Documentacion: https://docs.docker.com/docker-for-windows/install-windows-home/

MacOs:

- Instalacion: https://www.youtube.com/watch?v=mbSsh40_8WM
- Documentacion: https://docs.docker.com/docker-for-mac/install/

Your First Container

```
$ docker run hello-world

Hello from Docker.
This message shows that your installation appears to be working correctly.
...
```

Your Second Container

```
$ docker pull busybox
```

Checking Images

```
$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             VIRTUAL
busybox             latest              c51f86c28340        4 weeks ago         1.109 ME
```

Running Your Second Container

```
$ docker run busybox
$
```

Running With Extra Parameters

```
$ docker run busybox echo "hello from busybox"
hello from busybox
```

Check Running Containers

```
$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
```

Check stopped containers

```
$ docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
305297d7a235        busybox             "uptime"            11 minutes ago      Exited (0)
ff0a5c3750b9        busybox             "sh"                12 minutes ago      Exited (0)
14e5bd11d164        hello-world         "/hello"            2 minutes ago       Exited (0)
```

Run and get inside the container

```
$ docker run -it busybox sh
/ # ls
bin   dev   etc   home  proc  root  sys   tmp   usr   var
/ # uptime
 05:45:21 up  5:58,  0 users,  load average: 0.00, 0.01, 0.04
```

Delete Containers

```
$ docker rm 305297d7a235 ff0a5c3750b9
305297d7a235
ff0a5c3750b9
```

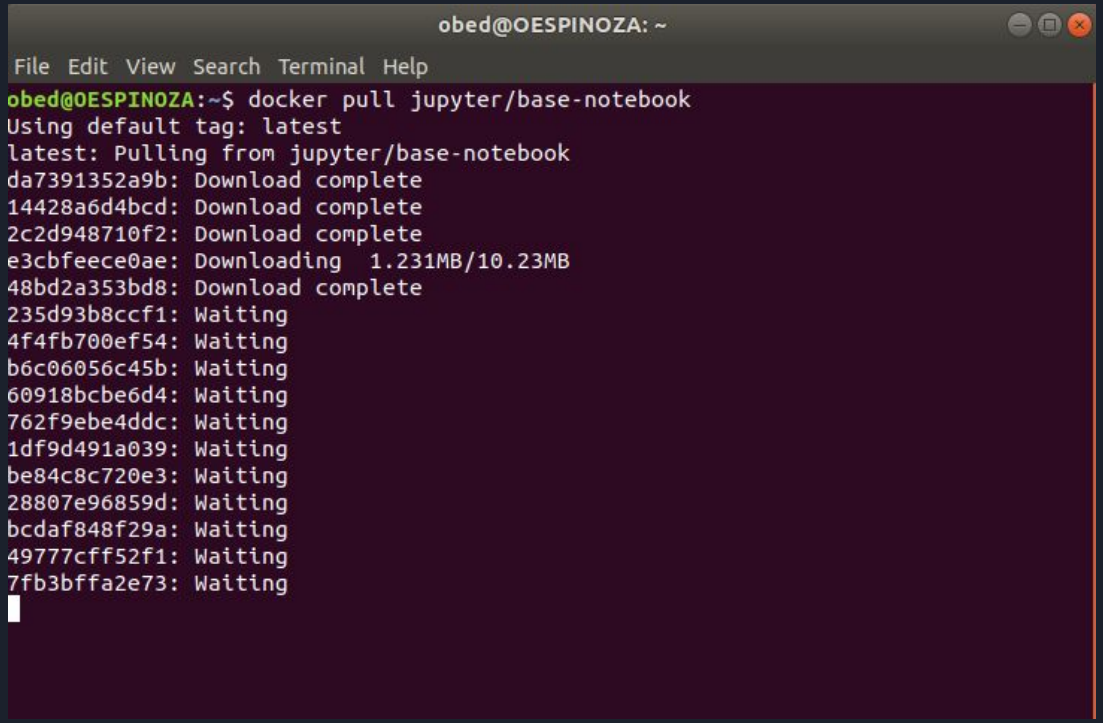
Delete All

```
$ docker container prune
WARNING! This will remove all stopped containers.
Are you sure you want to continue? [y/N] y
Deleted Containers:
4a7f7eebae0f63178aff7eb0aa39f0627a203ab2df258c1a00b456cf20063
f98f9c2aa1eaf727e4ec9c0283bcaa4762fbdba7f26191f26c97f64090360

Total reclaimed space: 212 B
```

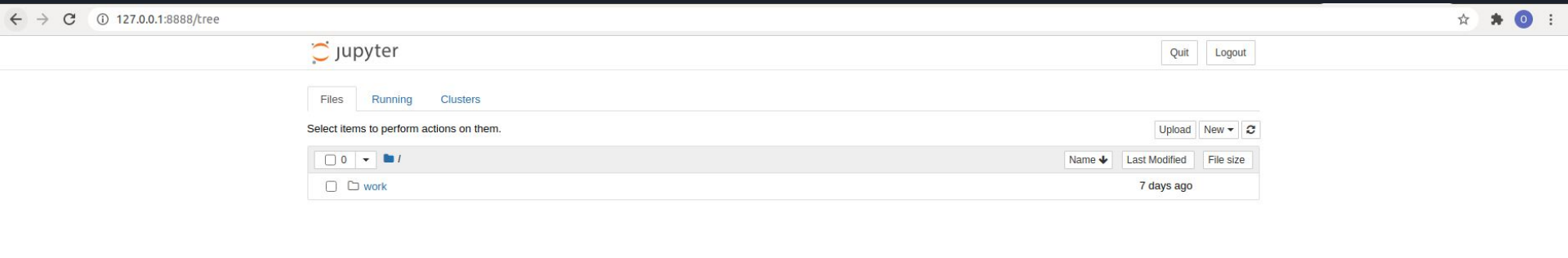
Docker and Python

Install Jupyter



Run Jupyter

- docker run -p 8888:8888 jupyter/base-notebook



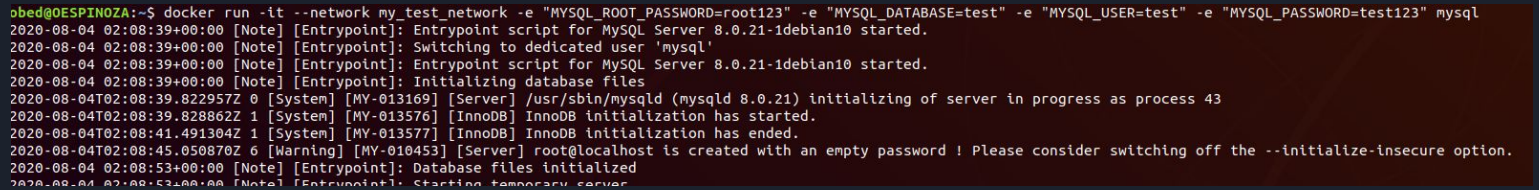
Create a Network

- docker network create --driver bridge my_test_network

```
obed@OESPINOZA:~$ docker network create --driver bridge my_test_network
f4afb7fa41d1f41737f516b55b4e9e62908df1513e4974ec639f7593cd176703
obed@OESPINOZA:~$
```

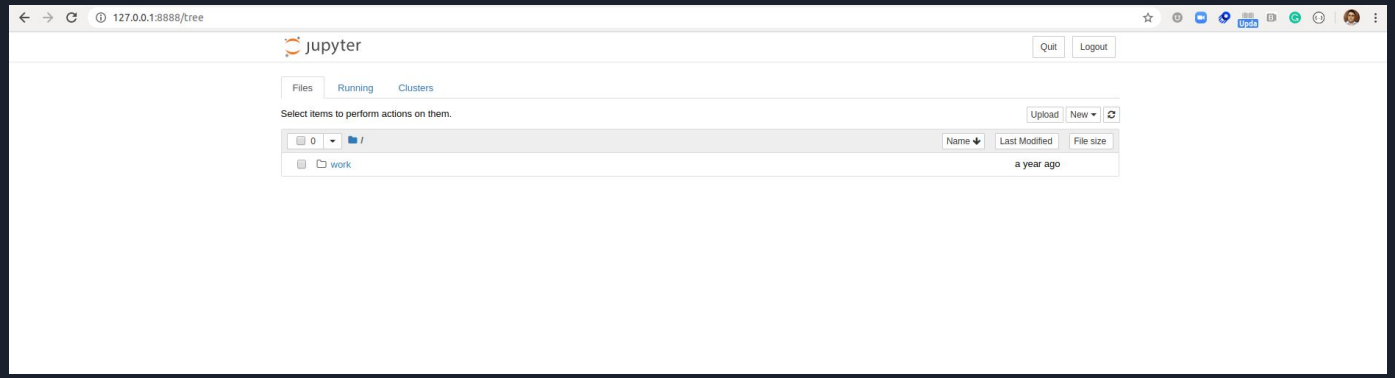
Run Mysql Container

- docker run -it --network my_test_network -e "MYSQL_ROOT_PASSWORD=root123" -e "MYSQL_DATABASE=test" -e "MYSQL_USER=test" -e "MYSQL_PASSWORD=test123" mysql:5.7.35



Run Jupyter With Network

- docker run --network my_test_network -p 8888:8888 jupyter/base-notebook



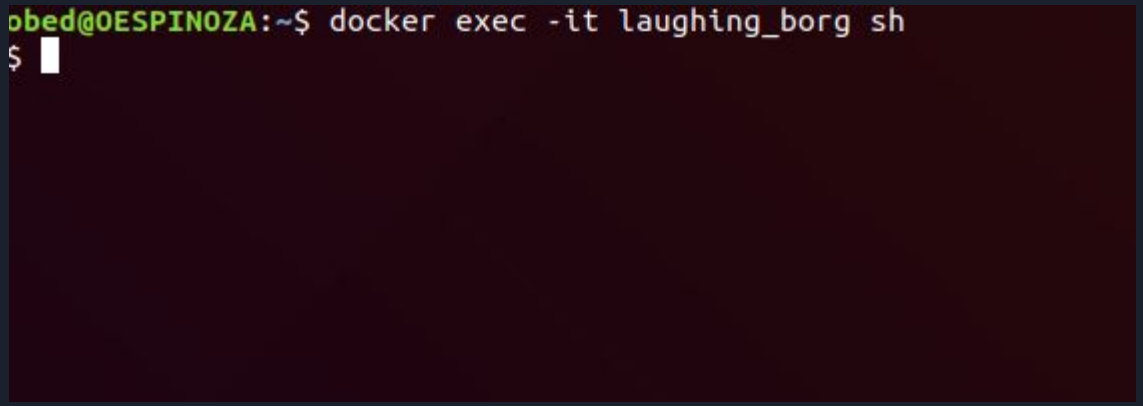
Check Running Containers

- Docker ps

```
obed@OESPINOZA:~$ docker ps
CONTAINER ID        IMAGE                                  COMMAND                  CREATED             STATUS              PORTS                              NAMES
bac8401dab67        jupyter/scipy-notebook:17aba6048f44    "tini -g -- start-no…"   28 seconds ago      Up 26 seconds       0.0.0.0:8888->8888/tcp             laughing_borg
68acdb620f0c        mysql                                  "docker-entrypoint.s…"   9 minutes ago       Up 9 minutes        3306/tcp, 33060/tcp                affectionate_johnson
```

Accessing Jupyter Container

- docker exec -it laughing_borg sh



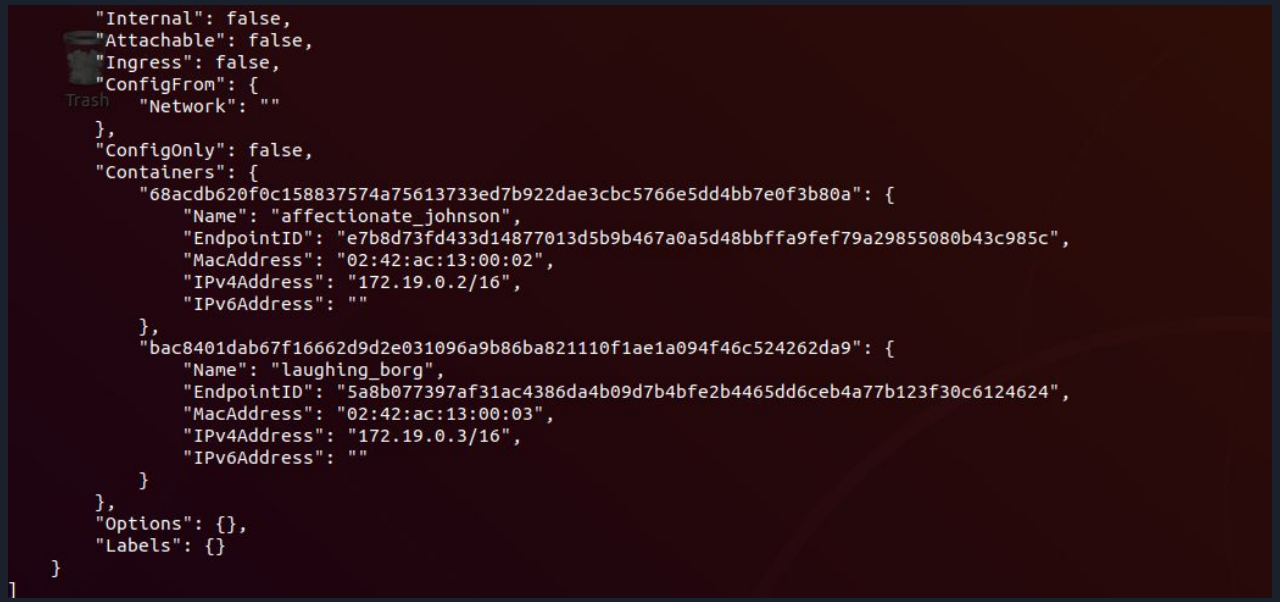
Install Mysql Connector

- pip install mysql-connector-python

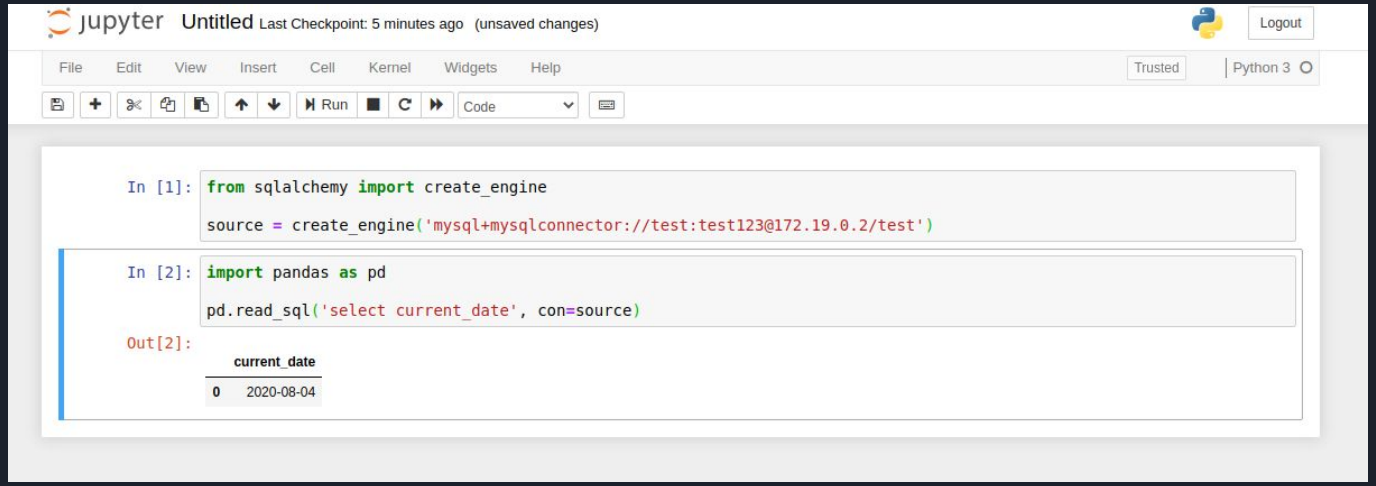
```
$ pip install mysql-connector-python
Collecting mysql-connector-python
  Downloading https://files.pythonhosted.org/packages/ce/1d/378f92f45fb98d7e033f6a94b0600b8ae496e67ba411b992f536784a3d0d/mysql_connector_python-8.0.21-cp36-cp36m-manylinux1_x86_64.whl (15.8MB)
    100% |████████████████████████████████| 15.8MB 2.4MB/s
Requirement already satisfied: protobuf>=3.0.0 in /opt/conda/lib/python3.6/site-packages (from mysql-connector-python) (3.6.1)
Requirement already satisfied: six>=1.9 in /opt/conda/lib/python3.6/site-packages (from protobuf>=3.0.0->mysql-connector-python) (1.12.0)
Requirement already satisfied: setuptools in /opt/conda/lib/python3.6/site-packages (from protobuf>=3.0.0->mysql-connector-python) (40.6.3)
Installing collected packages: mysql-connector-python
Successfully installed mysql-connector-python-8.0.21
$
```

Check Network

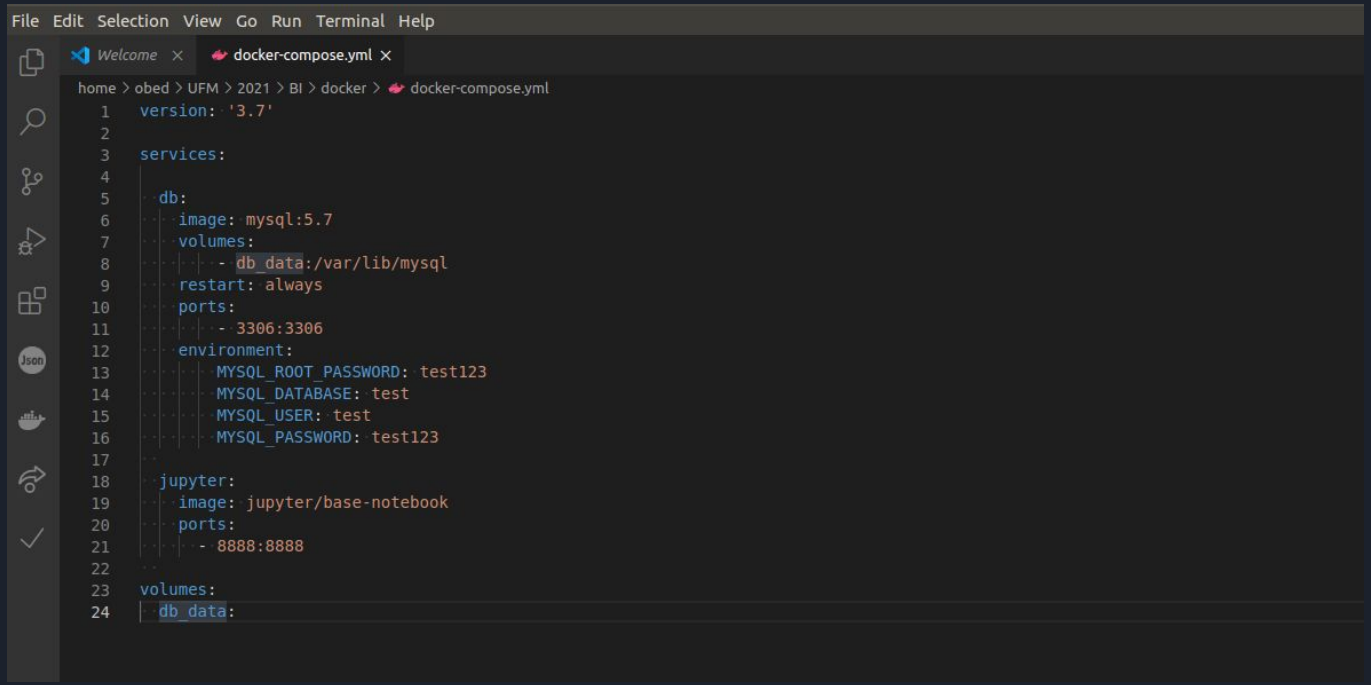
- docker network inspect my_test_network



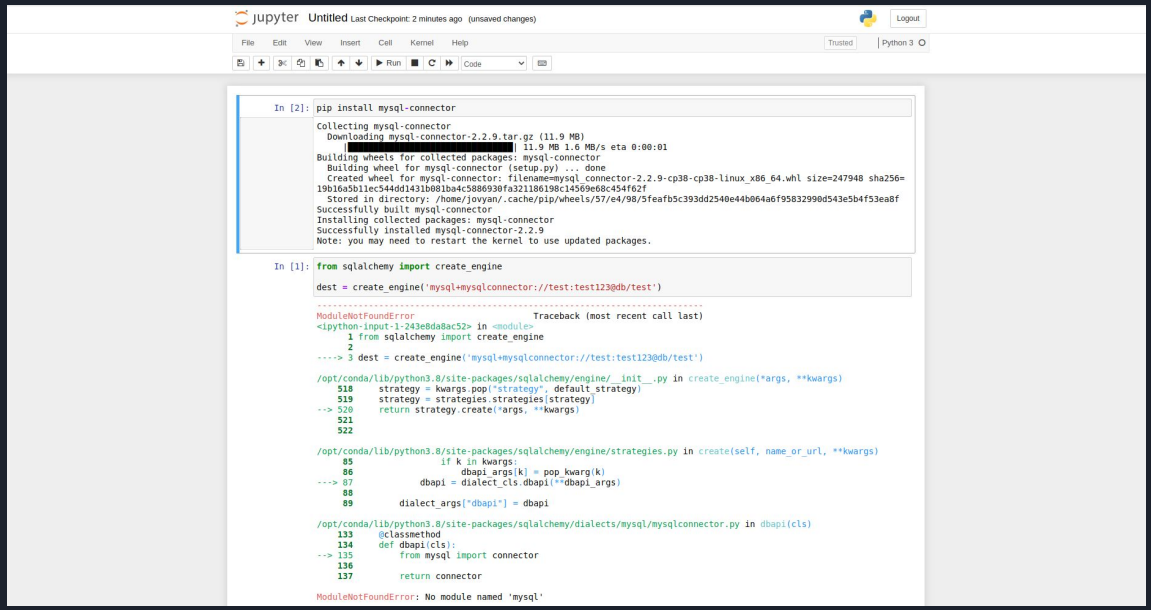
Test our connection



Introducing Docker Compose



Install Mysql Client



Install Pandas