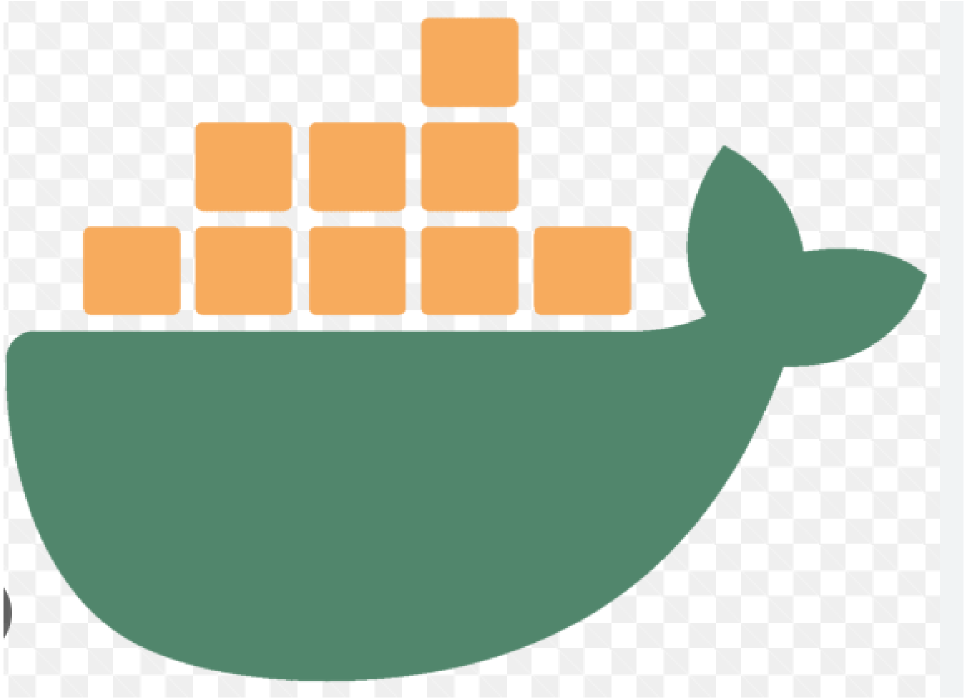
Day 16 Task: Docker for DevOps Engineers.

Docker

Docker is a software platform that allows you to build, test, and deploy applications quickly. Docker packages software into standardized units called containers that have everything the software needs to run including libraries, system tools, code, and runtime. Using Docker, you can quickly deploy and scale applications into any environment and know your code will run.



Docker is a tool that performs OS-level virtualization, called as Containerization. Using this container Docker run applications. It allows applications to use the same Linux.

When we decide to deploy a application, we need a dockerfile.

A Dockerfile is like a set of instructions for making a container. It tells Docker what base image to use, what commands to run, and what files to include.

Dockerfile uses some of commands to communicate with Docker and create a Docker image.

Brief description of Docker Commands:

1] FROM - This Command is used to specify the base image and version.

2]RUN - Execute a command in the image. This command is run during the building the image process.

3]COPY - This COPY command is used to copy the files from host machine to image.

4] ENV - This ENV command are used to set the environment variable in the image.

5] CMD - CMD execute the command same as shell command and they are not capable of run or execute a image

6]ENTRYPOINT - ENTRYPOINT execute the command same as CMD. This is latest version of CMD Command

7]EXPOSE - Specifies the ports that should be exposed on the container

8]ADD - This ADD command is same as COPY command used to copy the files from host machine to image.

The Only diff is, Using ADD Command you copy the data from tar archived files as well as you copy the data from using <URL>

Tasks

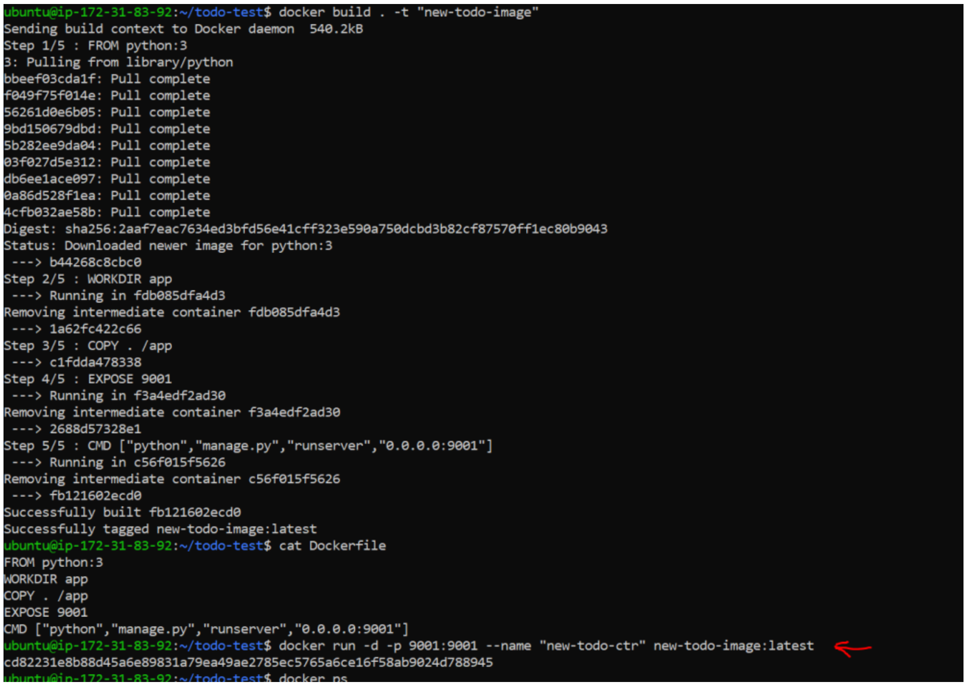
As you have already installed docker in previous days tasks, now is the time to run Docker commands.

- Use the docker run command to start a new container and interact with it through the command line. [Hint: docker run hello-world]

Syntax:-

```
docker run [OPTIONS] IMAGE [COMMAND] [ARG...]
```

Here we using Run Command. This command is used for Creating a Container



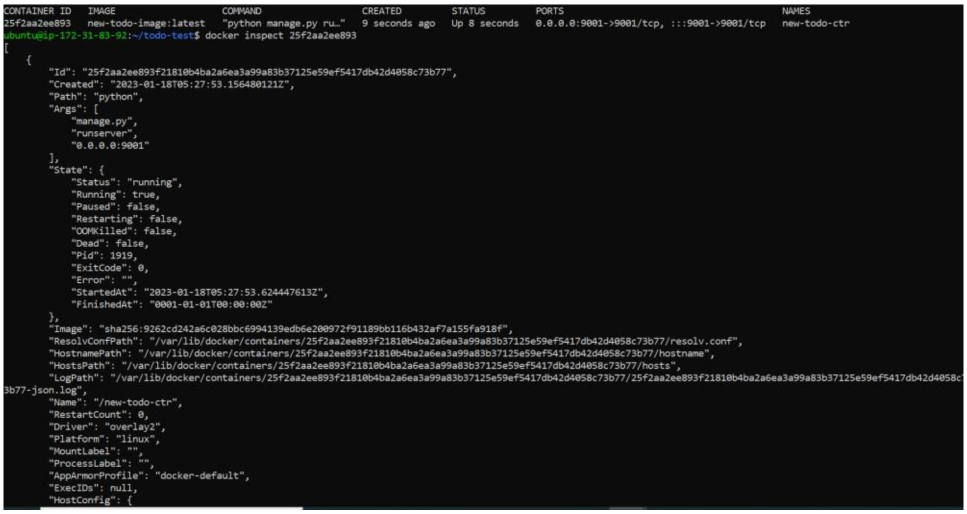
```
ubuntu@ip-172-31-83-92:~/todo-test$ docker build . -t "new-todo-image"
Sending build context to Docker daemon  540.2kB
Step 1/5 : FROM python:3
3: Pulling from library/python
bbeef03cda1f: Pull complete
f049f75f014e: Pull complete
56261d0e6b05: Pull complete
9bd150679dbd: Pull complete
5b282ee9da04: Pull complete
03f027d5e312: Pull complete
db6ee1ace097: Pull complete
0a86d528f1ea: Pull complete
4cfb032ae58b: Pull complete
Digest: sha256:2aaf7eac7634ed3bfd56e41cff323e590a750dcbd3b82cf87570ff1ec80b9043
Status: Downloaded newer image for python:3
 ---> b44268c8cbc0
Step 2/5 : WORKDIR app
 ---> Running in fdb085dfa4d3
Removing intermediate container fdb085dfa4d3
 ---> 1a62fc422c66
Step 3/5 : COPY . /app
 ---> c1fdda478338
Step 4/5 : EXPOSE 9001
 ---> Running in f3a4edf2ad30
Removing intermediate container f3a4edf2ad30
 ---> 2688d57328e1
Step 5/5 : CMD ["python","manage.py","runserver","0.0.0.0:9001"]
 ---> Running in c56f015f5626
Removing intermediate container c56f015f5626
 ---> fb121602ecd0
Successfully built fb121602ecd0
Successfully tagged new-todo-image:latest
ubuntu@ip-172-31-83-92:~/todo-test$ cat Dockerfile
FROM python:3
WORKDIR app
COPY . /app
EXPOSE 9001
CMD ["python","manage.py","runserver","0.0.0.0:9001"]
ubuntu@ip-172-31-83-92:~/todo-test$ docker run -d -p 9001:9001 --name "new-todo-ctr" new-todo-image:latest
cd82231e8b88d45a6e89831a79ea49ae2785ec5765a6ce16f58ab9024d788945
ubuntu@ip-172-31-83-92:~/todo-test$ docker ps
```

- Use the docker inspect command to view detailed information about a container or image.

Syntax:-

```
docker inspect [OPTIONS] NAME|ID [NAME|ID...]
```

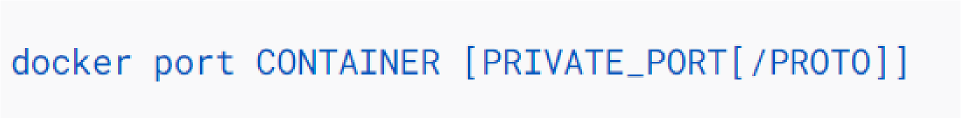
Docker inspect provides detailed information on constructs controlled by Docker.



```
CONTAINER ID   IMAGE                   COMMAND                  CREATED         STATUS         PORTS                                       NAMES
25f2aa2ee893   new-todo-image:latest   "python manage.py ru…"   9 seconds ago   Up 8 seconds   0.0.0.0:9001->9001/tcp, :::9001->9001/tcp   new-todo-ctr
ubuntu@ip-172-31-83-92:~/todo-test$ docker inspect 25f2aa2ee893
[
    {
        "Id": "25f2aa2ee893f21810b4ba2a6ea3a99a83b37125e59ef5417db42d4058c73b77",
        "Created": "2023-01-18T05:27:53.156480121Z",
        "Path": "python",
        "Args": [
            "manage.py",
            "runserver",
            "0.0.0.0:9001"
        ],
        "State": {
            "Status": "running",
            "Running": true,
            "Paused": false,
            "Restarting": false,
            "OOMKilled": false,
            "Dead": false,
            "Pid": 1919,
            "ExitCode": 0,
            "Error": "",
            "StartedAt": "2023-01-18T05:27:53.624447613Z",
            "FinishedAt": "0001-01-01T00:00:00Z"
        },
        "Image": "sha256:9262cd242a6c028bbc6994139edb6e200972f91189bb116b432af7a155fa918f",
        "ResolvConfPath": "/var/lib/docker/containers/25f2aa2ee893f21810b4ba2a6ea3a99a83b37125e59ef5417db42d4058c73b77/resolv.conf",
        "HostnamePath": "/var/lib/docker/containers/25f2aa2ee893f21810b4ba2a6ea3a99a83b37125e59ef5417db42d4058c73b77/hostname",
        "HostsPath": "/var/lib/docker/containers/25f2aa2ee893f21810b4ba2a6ea3a99a83b37125e59ef5417db42d4058c73b77/hosts",
        "LogPath": "/var/lib/docker/containers/25f2aa2ee893f21810b4ba2a6ea3a99a83b37125e59ef5417db42d4058c73b77/25f2aa2ee893f21810b4ba2a6ea3a99a83b37125e59ef5417db42d4058c
3b77-json.log",
        "Name": "/new-todo-ctr",
        "RestartCount": 0,
        "Driver": "overlay2",
        "Platform": "linux",
        "MountLabel": "",
        "ProcessLabel": "",
        "AppArmorProfile": "docker-default",
        "ExecIDs": null,
        "HostConfig": {
```

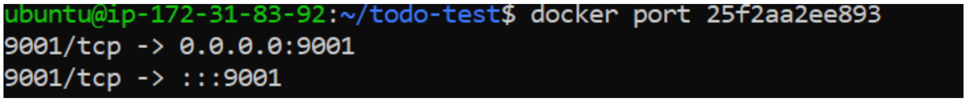
- Use the docker port command to list the port mappings for a container.

Syntax:-



```
docker port CONTAINER [PRIVATE_PORT[/PROTO]]
```

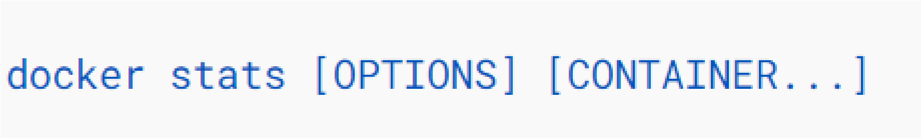
Suppose you want to find out all mapped port, then you will use this port command



```
ubuntu@ip-172-31-83-92:~/todo-test$ docker port 25f2aa2ee893
9001/tcp -> 0.0.0.0:9001
9001/tcp -> :::9001
```

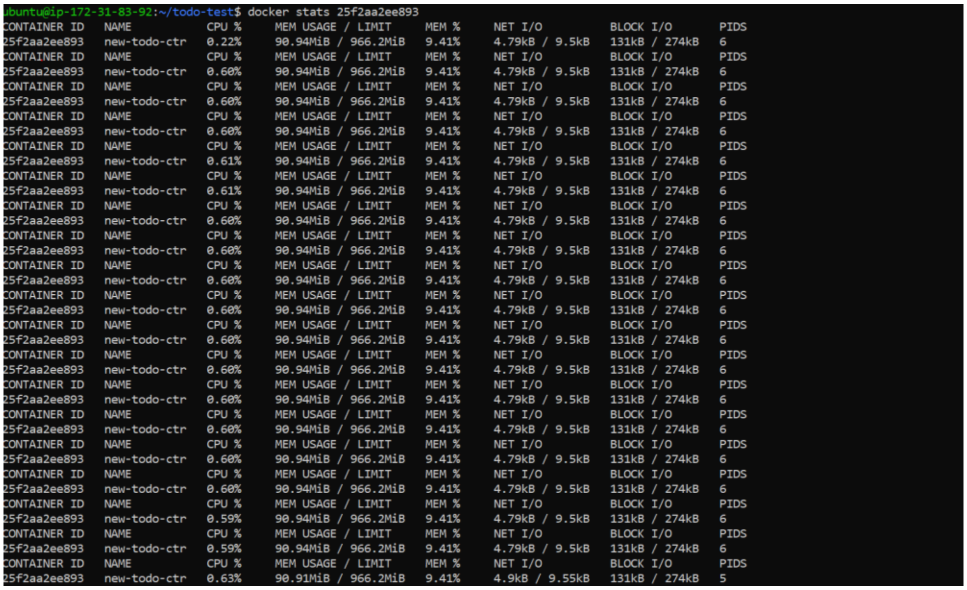
- Use the docker stats command to view resource usage statistics for one or more containers.

Syntax:-



```
docker stats [OPTIONS] [CONTAINER...]
```

The Docker stats returns a live data stream for running container.



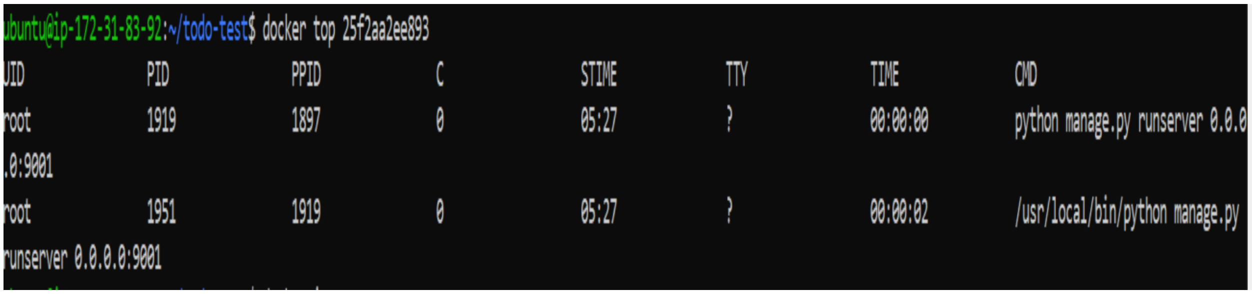
```
ubuntu@ip-172-31-83-92:~/todo-test$ docker stats 25f2aa2ee893
CONTAINER ID   NAME           CPU %     MEM USAGE / LIMIT     MEM %     NET I/O           BLOCK I/O         PIDS
25f2aa2ee893   new-todo-ctr   0.22%     90.94MiB / 966.2MiB   9.41%     4.79kB / 9.5kB    131kB / 274kB     6
CONTAINER ID   NAME           CPU %     MEM USAGE / LIMIT     MEM %     NET I/O           BLOCK I/O         PIDS
25f2aa2ee893   new-todo-ctr   0.60%     90.94MiB / 966.2MiB   9.41%     4.79kB / 9.5kB    131kB / 274kB     6
CONTAINER ID   NAME           CPU %     MEM USAGE / LIMIT     MEM %     NET I/O           BLOCK I/O         PIDS
25f2aa2ee893   new-todo-ctr   0.60%     90.94MiB / 966.2MiB   9.41%     4.79kB / 9.5kB    131kB / 274kB     6
CONTAINER ID   NAME           CPU %     MEM USAGE / LIMIT     MEM %     NET I/O           BLOCK I/O         PIDS
25f2aa2ee893   new-todo-ctr   0.60%     90.94MiB / 966.2MiB   9.41%     4.79kB / 9.5kB    131kB / 274kB     6
CONTAINER ID   NAME           CPU %     MEM USAGE / LIMIT     MEM %     NET I/O           BLOCK I/O         PIDS
25f2aa2ee893   new-todo-ctr   0.61%     90.94MiB / 966.2MiB   9.41%     4.79kB / 9.5kB    131kB / 274kB     6
CONTAINER ID   NAME           CPU %     MEM USAGE / LIMIT     MEM %     NET I/O           BLOCK I/O         PIDS
25f2aa2ee893   new-todo-ctr   0.61%     90.94MiB / 966.2MiB   9.41%     4.79kB / 9.5kB    131kB / 274kB     6
CONTAINER ID   NAME           CPU %     MEM USAGE / LIMIT     MEM %     NET I/O           BLOCK I/O         PIDS
25f2aa2ee893   new-todo-ctr   0.60%     90.94MiB / 966.2MiB   9.41%     4.79kB / 9.5kB    131kB / 274kB     6
CONTAINER ID   NAME           CPU %     MEM USAGE / LIMIT     MEM %     NET I/O           BLOCK I/O         PIDS
25f2aa2ee893   new-todo-ctr   0.60%     90.94MiB / 966.2MiB   9.41%     4.79kB / 9.5kB    131kB / 274kB     6
CONTAINER ID   NAME           CPU %     MEM USAGE / LIMIT     MEM %     NET I/O           BLOCK I/O         PIDS
25f2aa2ee893   new-todo-ctr   0.60%     90.94MiB / 966.2MiB   9.41%     4.79kB / 9.5kB    131kB / 274kB     6
CONTAINER ID   NAME           CPU %     MEM USAGE / LIMIT     MEM %     NET I/O           BLOCK I/O         PIDS
25f2aa2ee893   new-todo-ctr   0.60%     90.94MiB / 966.2MiB   9.41%     4.79kB / 9.5kB    131kB / 274kB     6
CONTAINER ID   NAME           CPU %     MEM USAGE / LIMIT     MEM %     NET I/O           BLOCK I/O         PIDS
25f2aa2ee893   new-todo-ctr   0.60%     90.94MiB / 966.2MiB   9.41%     4.79kB / 9.5kB    131kB / 274kB     6
CONTAINER ID   NAME           CPU %     MEM USAGE / LIMIT     MEM %     NET I/O           BLOCK I/O         PIDS
25f2aa2ee893   new-todo-ctr   0.60%     90.94MiB / 966.2MiB   9.41%     4.79kB / 9.5kB    131kB / 274kB     6
CONTAINER ID   NAME           CPU %     MEM USAGE / LIMIT     MEM %     NET I/O           BLOCK I/O         PIDS
25f2aa2ee893   new-todo-ctr   0.60%     90.94MiB / 966.2MiB   9.41%     4.79kB / 9.5kB    131kB / 274kB     6
CONTAINER ID   NAME           CPU %     MEM USAGE / LIMIT     MEM %     NET I/O           BLOCK I/O         PIDS
25f2aa2ee893   new-todo-ctr   0.60%     90.94MiB / 966.2MiB   9.41%     4.79kB / 9.5kB    131kB / 274kB     6
CONTAINER ID   NAME           CPU %     MEM USAGE / LIMIT     MEM %     NET I/O           BLOCK I/O         PIDS
25f2aa2ee893   new-todo-ctr   0.60%     90.94MiB / 966.2MiB   9.41%     4.79kB / 9.5kB    131kB / 274kB     6
CONTAINER ID   NAME           CPU %     MEM USAGE / LIMIT     MEM %     NET I/O           BLOCK I/O         PIDS
25f2aa2ee893   new-todo-ctr   0.60%     90.94MiB / 966.2MiB   9.41%     4.79kB / 9.5kB    131kB / 274kB     6
CONTAINER ID   NAME           CPU %     MEM USAGE / LIMIT     MEM %     NET I/O           BLOCK I/O         PIDS
25f2aa2ee893   new-todo-ctr   0.60%     90.94MiB / 966.2MiB   9.41%     4.79kB / 9.5kB    131kB / 274kB     6
CONTAINER ID   NAME           CPU %     MEM USAGE / LIMIT     MEM %     NET I/O           BLOCK I/O         PIDS
25f2aa2ee893   new-todo-ctr   0.60%     90.94MiB / 966.2MiB   9.41%     4.79kB / 9.5kB    131kB / 274kB     6
CONTAINER ID   NAME           CPU %     MEM USAGE / LIMIT     MEM %     NET I/O           BLOCK I/O         PIDS
25f2aa2ee893   new-todo-ctr   0.59%     90.94MiB / 966.2MiB   9.41%     4.79kB / 9.5kB    131kB / 274kB     6
CONTAINER ID   NAME           CPU %     MEM USAGE / LIMIT     MEM %     NET I/O           BLOCK I/O         PIDS
25f2aa2ee893   new-todo-ctr   0.59%     90.94MiB / 966.2MiB   9.41%     4.79kB / 9.5kB    131kB / 274kB     6
CONTAINER ID   NAME           CPU %     MEM USAGE / LIMIT     MEM %     NET I/O           BLOCK I/O         PIDS
25f2aa2ee893   new-todo-ctr   0.63%     90.91MiB / 966.2MiB   9.41%     4.9kB / 9.55kB    131kB / 274kB     5
```

- Use the docker top command to view the processes running inside a container.

Syntax:-

```
docker top CONTAINER [ps OPTIONS]
```

Mostly the Docker Top command is used to, Display the running process of a container.



```
ubuntu@ip-172-31-83-92:~/todo-test$ docker top 25f2aa2ee893
UID          PID          PPID         C            STIME        TTY          TIME         CMD
root         1919         1897         0            05:27        ?            00:00:00     python manage.py runserver 0.0.0
.0:9001
root         1951         1919         0            05:27        ?            00:00:02     /usr/local/bin/python manage.py
runserver 0.0.0.0:9001
```

- Use the docker save command to save an image to a tar archive.

Syntax:-

```
docker save [OPTIONS] IMAGE [IMAGE...]
```

Use Save command to save one or more images to a tar archive

```
ubuntu@ip-172-31-83-92:~/todo-test$ docker save new-todo-image:latest | gzip >new_todo_image.tar.gz
```

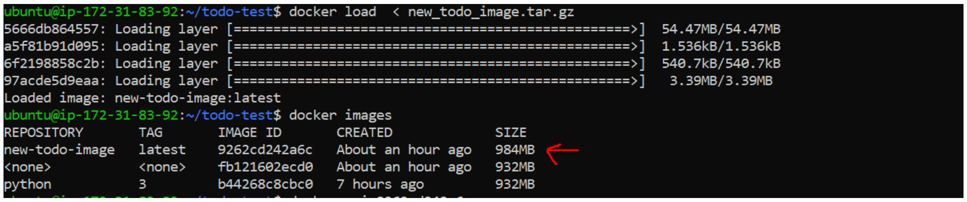
```
ubuntu@ip-172-31-83-92:~/todo-test$ ls
Dockerfile  LICENSE  README.md  db.sqlite3  manage.py  new_todo_image.tar.gz  staticfiles  todoApp  todos
```

- Use the docker load command to load an image from a tar archive.

Syntax:-

```
docker load [OPTIONS]
```

Load an image from a tar archive



```
ubuntu@ip-172-31-83-92:~/todo-test$ docker load  < new_todo_image.tar.gz
5666db864557: Loading layer [==================================================>]  54.47MB/54.47MB
a5f81b91d095: Loading layer [==================================================>]  1.536kB/1.536kB
6f2198858c2b: Loading layer [==================================================>]  540.7kB/540.7kB
97acde5d9eaa: Loading layer [==================================================>]   3.39MB/3.39MB
Loaded image: new-todo-image:latest
ubuntu@ip-172-31-83-92:~/todo-test$ docker images
REPOSITORY       TAG       IMAGE ID       CREATED             SIZE
new-todo-image   latest    9262cd242a6c   About an hour ago   984MB
<none>           <none>    fb121602ecd0   About an hour ago   932MB
python           3         b44268c8cbc0   7 hours ago         932MB
```

These tasks involve simple operations that can be used to manage images and containers.

---Happy Learning😋---