Getting Started with Docker Containers

Objective:

This tutorial is designed to guide you through the basics of building and running a simple Docker container. It's tailored for beginners and assumes no prior knowledge of Docker. By the end of this tutorial, you will have a foundational understanding of Docker containers and how to use them to run a Python script in an isolated environment.

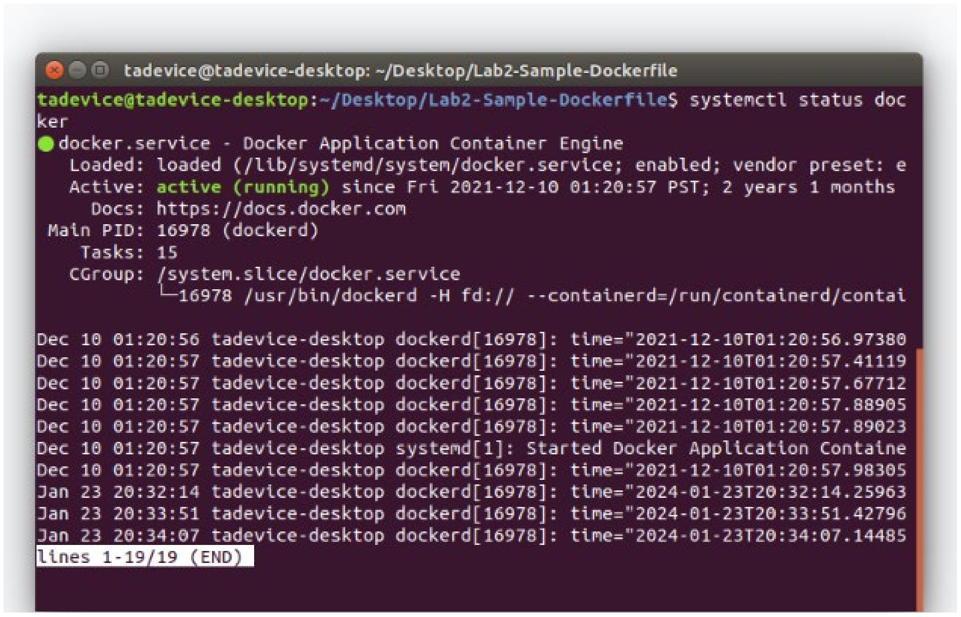
Prerequisites:

- Download the sample Dockerfile and Python Script

Installation

- **On a Jetson device**: Docker is preconfigured.
    - To check if Docker is running, open a terminal and enter

```
$ systemctl status docker
```



```
tadevice@tadevice-desktop: ~/Desktop/Lab2-Sample-Dockerfile
tadevice@tadevice-desktop:~/Desktop/Lab2-Sample-Dockerfile$ systemctl status doc
ker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: e
   Active: active (running) since Fri 2021-12-10 01:20:57 PST; 2 years 1 months
     Docs: https://docs.docker.com
 Main PID: 16978 (dockerd)
    Tasks: 15
   CGroup: /system.slice/docker.service
           └─16978 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/contai

Dec 10 01:20:56 tadevice-desktop dockerd[16978]: time="2021-12-10T01:20:56.97380
Dec 10 01:20:57 tadevice-desktop dockerd[16978]: time="2021-12-10T01:20:57.41119
Dec 10 01:20:57 tadevice-desktop dockerd[16978]: time="2021-12-10T01:20:57.67712
Dec 10 01:20:57 tadevice-desktop dockerd[16978]: time="2021-12-10T01:20:57.88905
Dec 10 01:20:57 tadevice-desktop dockerd[16978]: time="2021-12-10T01:20:57.89023
Dec 10 01:20:57 tadevice-desktop systemd[1]: Started Docker Application Containe
Dec 10 01:20:57 tadevice-desktop dockerd[16978]: time="2021-12-10T01:20:57.98305
Jan 23 20:32:14 tadevice-desktop dockerd[16978]: time="2024-01-23T20:32:14.25963
Jan 23 20:33:51 tadevice-desktop dockerd[16978]: time="2024-01-23T20:33:51.42796
Jan 23 20:34:07 tadevice-desktop dockerd[16978]: time="2024-01-23T20:34:07.14485
lines 1-19/19 (END)
```

- **On a Laptop/Desktop**: If you don't have Docker installed, follow the installation guide at Docker Engine Installation.

Understanding Dockerfile

- A Dockerfile is a text file containing a set of instructions to build a Docker image.
- Our Dockerfile uses Ubuntu as a base image, sets up a working directory, copies a Python script into the container, installs Python dependencies, and specifies the command to run the script.

```
# Use an official Ubuntu base image
FROM ubuntu:latest
# Set the working directory in the container
WORKDIR /srv/
# Copy the Python script into the container at /srv/
COPY mycode.py .
# Install Dependencies
RUN apt-get -y update && apt-get -y install python3 && apt-get -y install
python3-pip
# Run the Python script when the container launches
CMD ["python3", "mycode.py"]
```

Note for Jetson Devices:

For Jetson devices, run Docker as a root user. So before executing any of the following commands switch to root user.

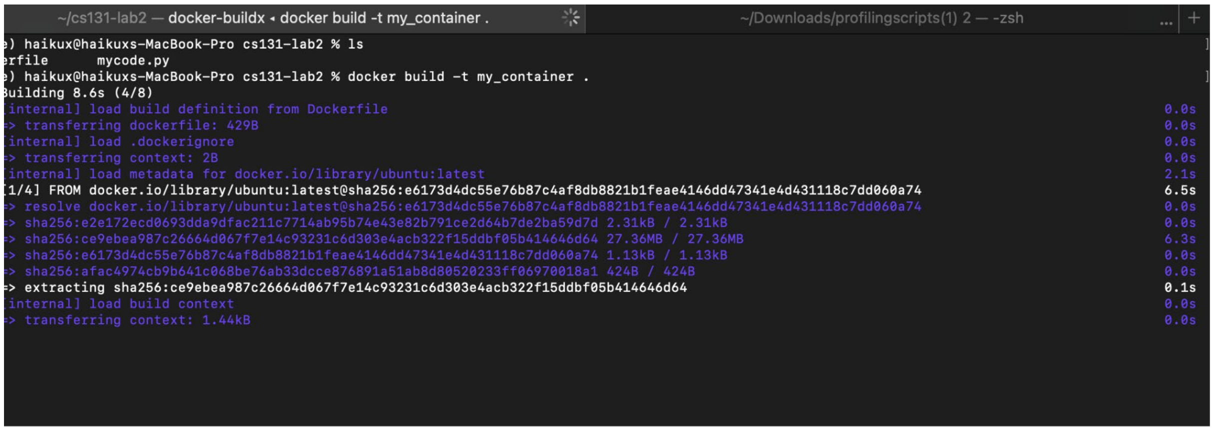
```
$ sudo su
```

Building and Running Your Docker Container

- **Build the Docker Image**:
  - Open a terminal in the directory containing your Dockerfile and `mycode.py`.
  - Build your image with the command:

```
$ docker build -t my_container .
```

    - Replace `my_container` with your container name, this creates a Docker image named `my_container` based on the instructions in your Dockerfile.

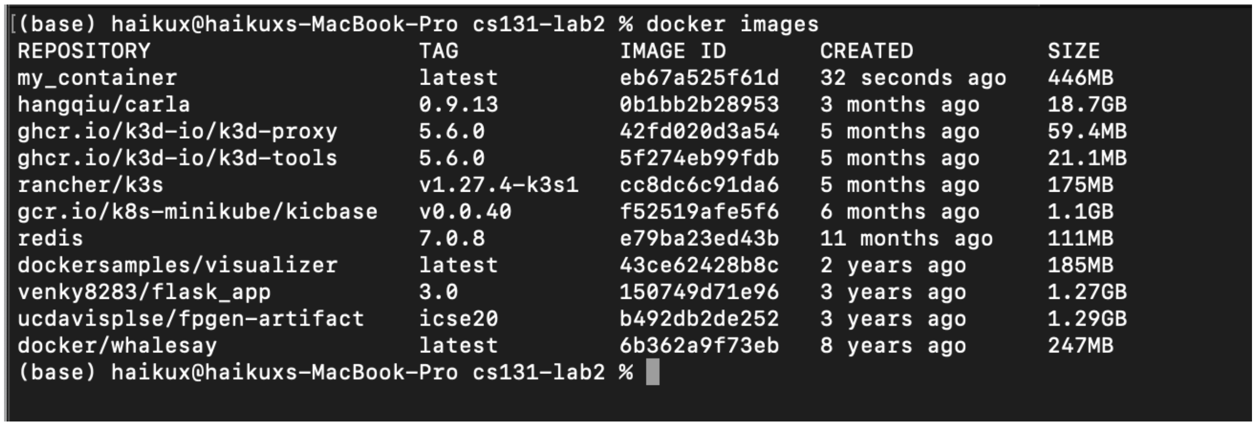


~/cs131-lab2 — docker-buildx ◂ docker build -t my_container .
~/Downloads/profilingscripts(1) 2 — -zsh

```
e) haikux@haikuxs-MacBook-Pro cs131-lab2 % ls
erfile      mycode.py
e) haikux@haikuxs-MacBook-Pro cs131-lab2 % docker build -t my_container .
Building 8.6s (4/8)
[internal] load build definition from Dockerfile                                                              0.0s
=> transferring dockerfile: 429B                                                                               0.0s
[internal] load .dockerignore                                                                                  0.0s
=> transferring context: 2B                                                                                    0.0s
[internal] load metadata for docker.io/library/ubuntu:latest                                                  2.1s
[1/4] FROM docker.io/library/ubuntu:latest@sha256:e6173d4dc55e76b87c4af8db8821b1feae4146dd47341e4d431118c7dd060a74   6.5s
=> resolve docker.io/library/ubuntu:latest@sha256:e6173d4dc55e76b87c4af8db8821b1feae4146dd47341e4d431118c7dd060a74   0.0s
=> sha256:e2e172ecd0693dda9dfac211c7714ab95b74e43e82b791ce2d64b7de2ba59d7d 2.31kB / 2.31kB                     0.0s
=> sha256:ce9ebea987c26664d067f7e14c93231c6d303e4acb322f15ddbf05b414646d64 27.36MB / 27.36MB                   6.3s
=> sha256:e6173d4dc55e76b87c4af8db8821b1feae4146dd47341e4d431118c7dd060a74 1.13kB / 1.13kB                     0.0s
=> sha256:afac4974cb9b641c068be76ab33dcce876891a51ab8d80520233ff06970018a1 424B / 424B                         0.0s
=> extracting sha256:ce9ebea987c26664d067f7e14c93231c6d303e4acb322f15ddbf05b414646d64                          0.1s
[internal] load build context                                                                                  0.0s
=> transferring context: 1.44kB                                                                                0.0s
```

- **Viewing Available Docker Images**:
    - To see a list of all Docker images available locally, use:

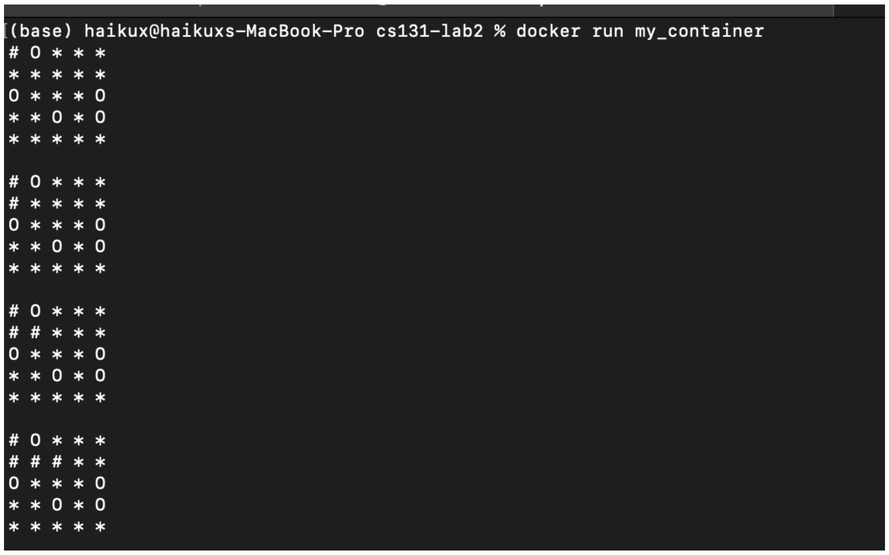
```
$ docker images
```



```
(base) haikux@haikuxs-MacBook-Pro cs131-lab2 % docker images
REPOSITORY                    TAG            IMAGE ID       CREATED          SIZE
my_container                  latest         eb67a525f61d   32 seconds ago   446MB
hangqiu/carla                 0.9.13         0b1bb2b28953   3 months ago     18.7GB
ghcr.io/k3d-io/k3d-proxy      5.6.0          42fd020d3a54   5 months ago     59.4MB
ghcr.io/k3d-io/k3d-tools      5.6.0          5f274eb99fdb   5 months ago     21.1MB
rancher/k3s                   v1.27.4-k3s1   cc8dc6c91da6   5 months ago     175MB
gcr.io/k8s-minikube/kicbase   v0.0.40        f52519afe5f6   6 months ago     1.1GB
redis                         7.0.8          e79ba23ed43b   11 months ago    111MB
dockersamples/visualizer      latest         43ce62428b8c   2 years ago      185MB
venky8283/flask_app           3.0            150749d71e96   3 years ago      1.27GB
ucdavisplse/fpgen-artifact    icse20         b492db2de252   3 years ago      1.29GB
docker/whalesay               latest         6b362a9f73eb   8 years ago      247MB
(base) haikux@haikuxs-MacBook-Pro cs131-lab2 %
```

- **Running Your Docker Container**:
    - **Standard Mode**:
        - To run your container in standard mode (which will execute `mycode.py`), use:

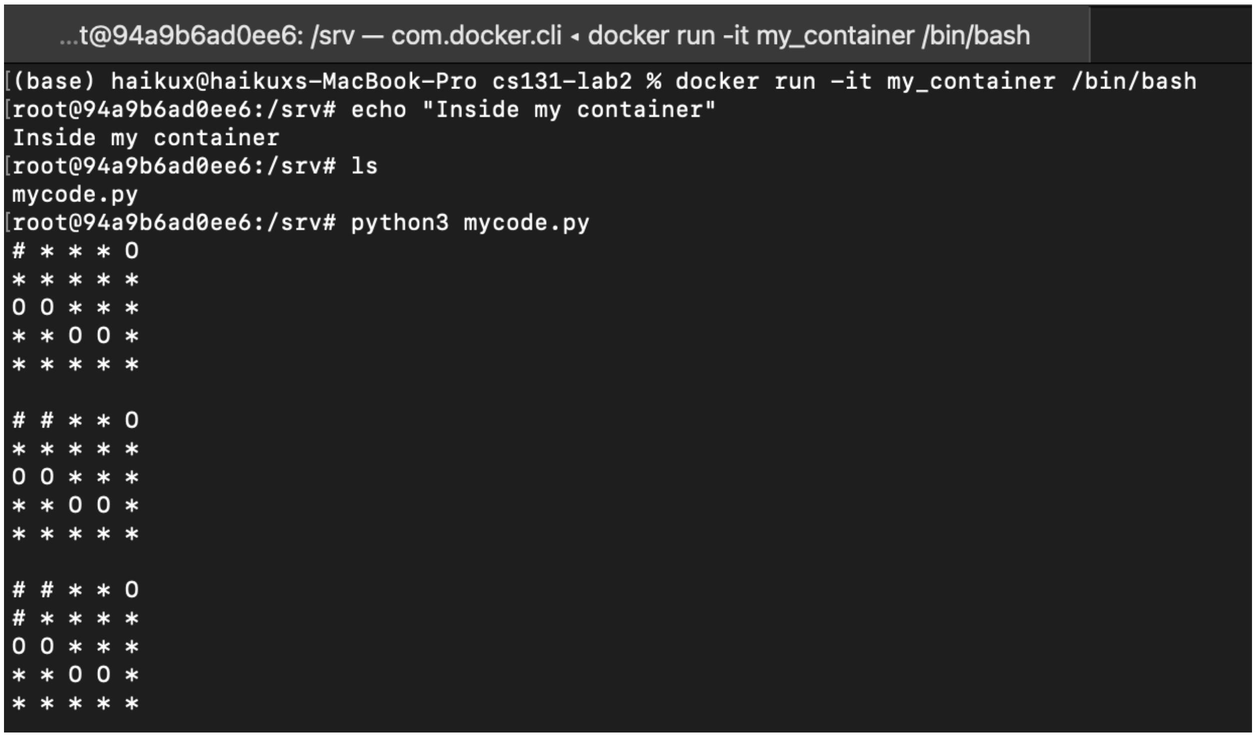
```
$ docker run my_container
```



```
(base) haikux@haikuxs-MacBook-Pro cs131-lab2 % docker run my_container
# O * * *
* * * * *
O * * * O
* * O * O
* * * * *

# O * * *
# * * * *
O * * * O
* * O * O
* * * * *

# O * * *
# # * * *
O * * * O
* * O * O
* * * * *

# O * * *
# # # * *
O * * * O
* * O * O
* * * * *
```

- **Interactive Mode**:
    - For an interactive session (which allows you to interact with the container via Bash), use:

```
$ docker run -it my_container /bin/bash
```


...t@94a9b6ad0ee6: /srv — com.docker.cli ◂ docker run -it my_container /bin/bash

```
[(base) haikux@haikuxs-MacBook-Pro cs131-lab2 % docker run -it my_container /bin/bash
[root@94a9b6ad0ee6:/srv# echo "Inside my container"
Inside my container
[root@94a9b6ad0ee6:/srv# ls
mycode.py
[root@94a9b6ad0ee6:/srv# python3 mycode.py
# * * * O
* * * * *
O O * * *
* * O O *
* * * * *

# # * * O
* * * * *
O O * * *
* * O O *
* * * * *

# # * * O
# * * * *
O O * * *
* * O O *
* * * * *
```

- **Background Mode**:
  - To run your container in the background (detached mode), use the `-d` flag:

```
docker run -d my_container
```

- This starts the container in the background and frees up your terminal. The container runs like a service, executing the script specified in the CMD instruction of the Dockerfile.
- To check the status of your background containers, use `docker ps`.

References:

  - Docker Documentation
  - Docker CLI Reference
  - Dockerfile Reference
  - Docker Hub
  - Play with Docker Interactive Classroom