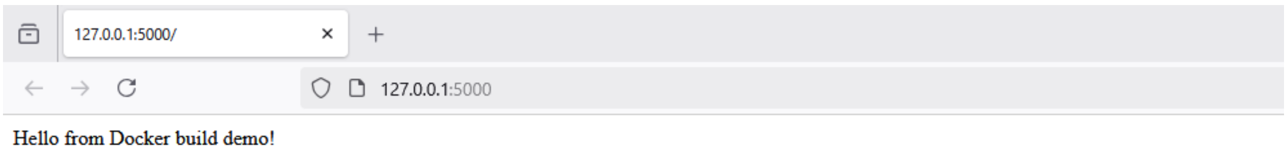
docker-depth

Our basic docker app is running fine

Dir structre is very simple:

1. app.py
2. requirements.txt
3. Dockerfile


Hello from Docker build demo!

Know we will see each

Dind

1. in order to create the DIND we first had to introduce a new file named dind-runner.sh. Its contect will be inside the repo.
2. we had o run the container using the command below:

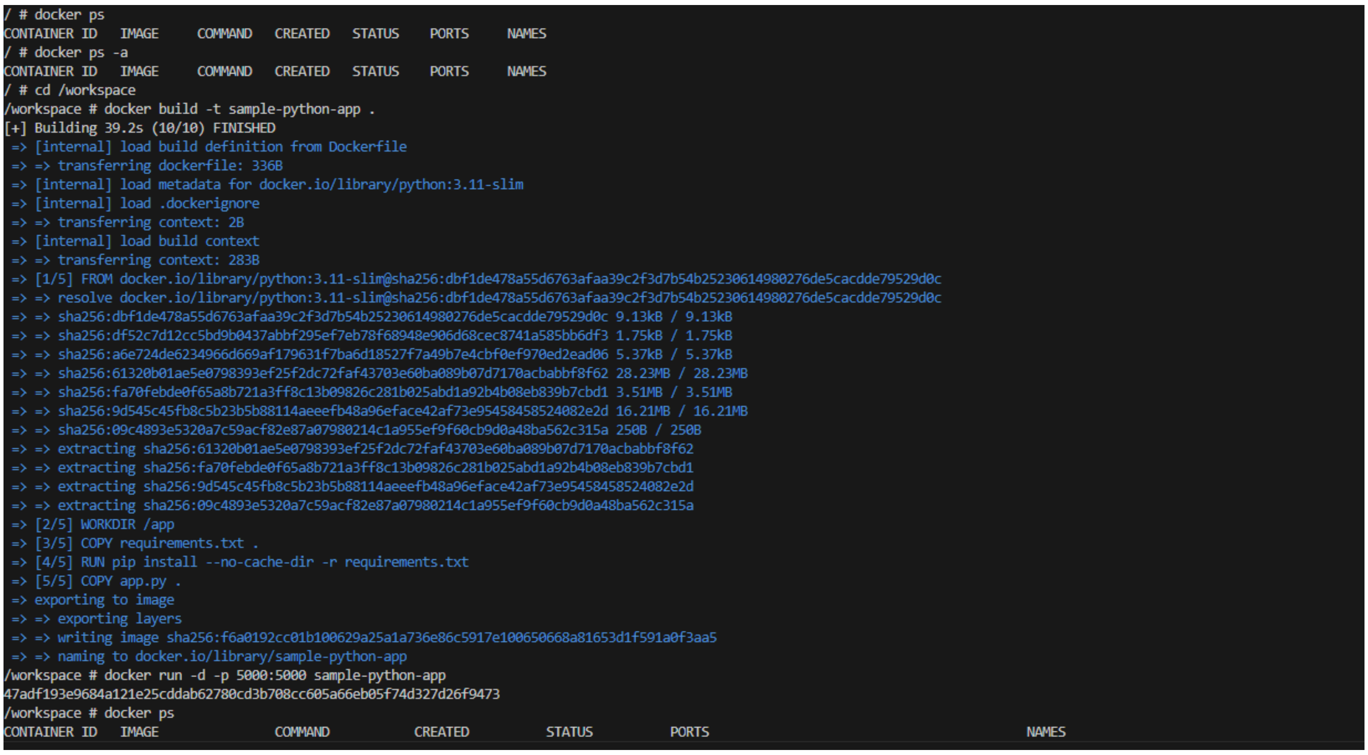
```
docker run --privileged --rm -it `
  -v ${PWD}:/workspace `
  docker:dind `
  /workspace/dind-runner.sh
```

3. Once the conainer is build we have to go inside the container using command and build our python app inside the workspace

```
docker exec -it 996d597251ff sh
```

```
 docker build -t sample-python-app .
```

The output for dind is below:



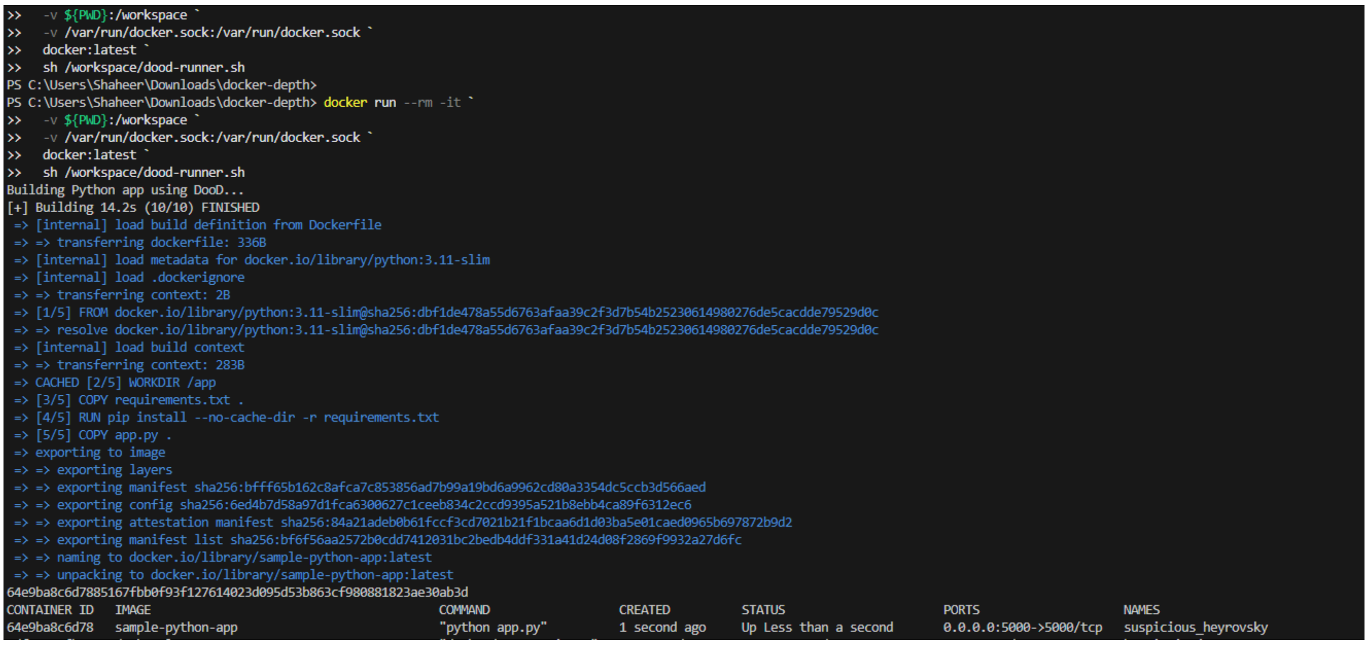
```
/ # docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
/ # docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
/ # cd /workspace
/workspace # docker build -t sample-python-app .
[+] Building 39.2s (10/10) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 336B
 => [internal] load metadata for docker.io/library/python:3.11-slim
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load build context
 => => transferring context: 283B
 => [1/5] FROM docker.io/library/python:3.11-slim@sha256:dbf1de478a55d6763afaa39c2f3d7b54b25230614980276de5cacdde79529d0c
 => => resolve docker.io/library/python:3.11-slim@sha256:dbf1de478a55d6763afaa39c2f3d7b54b25230614980276de5cacdde79529d0c
 => => sha256:dbf1de478a55d6763afaa39c2f3d7b54b25230614980276de5cacdde79529d0c 9.13kB / 9.13kB
 => => sha256:df52c7d12cc5bd9b0437abbf295ef7eb78f68948e906d68cec8741a585bb6df3 1.75kB / 1.75kB
 => => sha256:a6e724de6234966d669af179631f7ba6d18527f7a49b7e4cbf0ef970ed2ead06 5.37kB / 5.37kB
 => => sha256:61320b01ae5e0798393ef25f2dc72faf43703e60ba089b07d7170acbabbf8f62 28.23MB / 28.23MB
 => => sha256:fa70febde0f65a8b721a3ff8c13b09826c281b025abd1a92b4b08eb839b7cbd1 3.51MB / 3.51MB
 => => sha256:9d545c45fb8c5b23b5b88114aeeefb48a96eface42af73e95458458524082e2d 16.21MB / 16.21MB
 => => sha256:09c4893e5320a7c59acf82e87a07980214c1a955ef9f60cb9d0a48ba562c315a 250B / 250B
 => => extracting sha256:61320b01ae5e0798393ef25f2dc72faf43703e60ba089b07d7170acbabbf8f62
 => => extracting sha256:fa70febde0f65a8b721a3ff8c13b09826c281b025abd1a92b4b08eb839b7cbd1
 => => extracting sha256:9d545c45fb8c5b23b5b88114aeeefb48a96eface42af73e95458458524082e2d
 => => extracting sha256:09c4893e5320a7c59acf82e87a07980214c1a955ef9f60cb9d0a48ba562c315a
 => [2/5] WORKDIR /app
 => [3/5] COPY requirements.txt .
 => [4/5] RUN pip install --no-cache-dir -r requirements.txt
 => [5/5] COPY app.py .
 => exporting to image
 => => exporting layers
 => => writing image sha256:f6a0192cc01b100629a25a1a736e86c5917e100650668a81653d1f591a0f3aa5
 => => naming to docker.io/library/sample-python-app
/workspace # docker run -d -p 5000:5000 sample-python-app
47adf193e9684a121e25cddab62780cd3b708cc605a66eb05f74d327d26f9473
/workspace # docker ps
CONTAINER ID   IMAGE                COMMAND             CREATED            STATUS           PORTS                                            NAMES
```

Doond

1. We had to introduce a new script named dood-runner.sh in order to execute docker container in Doond.
2. The script is inside the repo which can be checked.
3. The command used to run the docker is below:

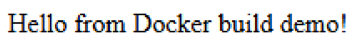
```
docker run --rm -it `
  -v ${PWD}:/workspace `
  -v /var/run/docker.sock:/var/run/docker.sock `
  docker:latest `
  sh /workspace/dood-runner.sh
```

the image below shows the container running after downloading the stuff



```
>>   -v ${PWD}:/workspace `
>>   -v /var/run/docker.sock:/var/run/docker.sock `
>>   docker:latest `
>>   sh /workspace/dood-runner.sh
PS C:\Users\Shaheer\Downloads\docker-depth>
PS C:\Users\Shaheer\Downloads\docker-depth> docker run --rm -it `
>>   -v ${PWD}:/workspace `
>>   -v /var/run/docker.sock:/var/run/docker.sock `
>>   docker:latest `
>>   sh /workspace/dood-runner.sh
Building Python app using DooD...
[+] Building 14.2s (10/10) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 336B
 => [internal] load metadata for docker.io/library/python:3.11-slim
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [1/5] FROM docker.io/library/python:3.11-slim@sha256:dbf1de478a55d6763afaa39c2f3d7b54b25230614980276de5cacdde79529d0c
 => => resolve docker.io/library/python:3.11-slim@sha256:dbf1de478a55d6763afaa39c2f3d7b54b25230614980276de5cacdde79529d0c
 => [internal] load build context
 => => transferring context: 283B
 => CACHED [2/5] WORKDIR /app
 => [3/5] COPY requirements.txt .
 => [4/5] RUN pip install --no-cache-dir -r requirements.txt
 => [5/5] COPY app.py .
 => exporting to image
 => => exporting layers
 => => exporting manifest sha256:bfff65b162c8afca7c853856ad7b99a19bd6a9962cd80a3354dc5ccb3d566aed
 => => exporting config sha256:6ed4b7d58a97d1fca6300627c1ceeb834c2ccd9395a521b8ebb4ca89f6312ec6
 => => exporting attestation manifest sha256:84a21adeb0b61fccf3cd7021b21f1bcaa6d1d03ba5e01caed0965b697872b9d2
 => => exporting manifest list sha256:bf6f56aa2572b0cdd7412031bc2bedb4ddf331a41d24d08f2869f9932a27d6fc
 => => naming to docker.io/library/sample-python-app:latest
 => => unpacking to docker.io/library/sample-python-app:latest
64e9ba8c6d7885167fbb0f93f127614023d095d53b863cf980881823ae30ab3d
CONTAINER ID   IMAGE                                        COMMAND                  CREATED          STATUS                      PORTS                    NAMES
64e9ba8c6d78   sample-python-app                            "python app.py"          1 second ago     Up Less than a second       0.0.0.0:5000->5000/tcp   suspicious_heyrovsky
```

the final result from the contaier shown on the web is:



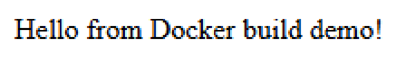
Hello from Docker build demo!

Kaniko

1. Serves best when we want to use in k8s or via ci/cd process. We push the image to docke hub and then we pull it in order to use. But for now we are just learning ang I am going to test in locally.
2. First we have to pull the kaniko image from the gcr.
3. The command used to work arond wiht Kaniko

```
docker run --rm -v ${PWD}:/workspace `
  gcr.io/kaniko-project/executor:latest `
  --dockerfile=Dockerfile `
  --context=dir:///workspace `
  --no-push `
  --tarPath=/workspace/sample-python-app.tar
```

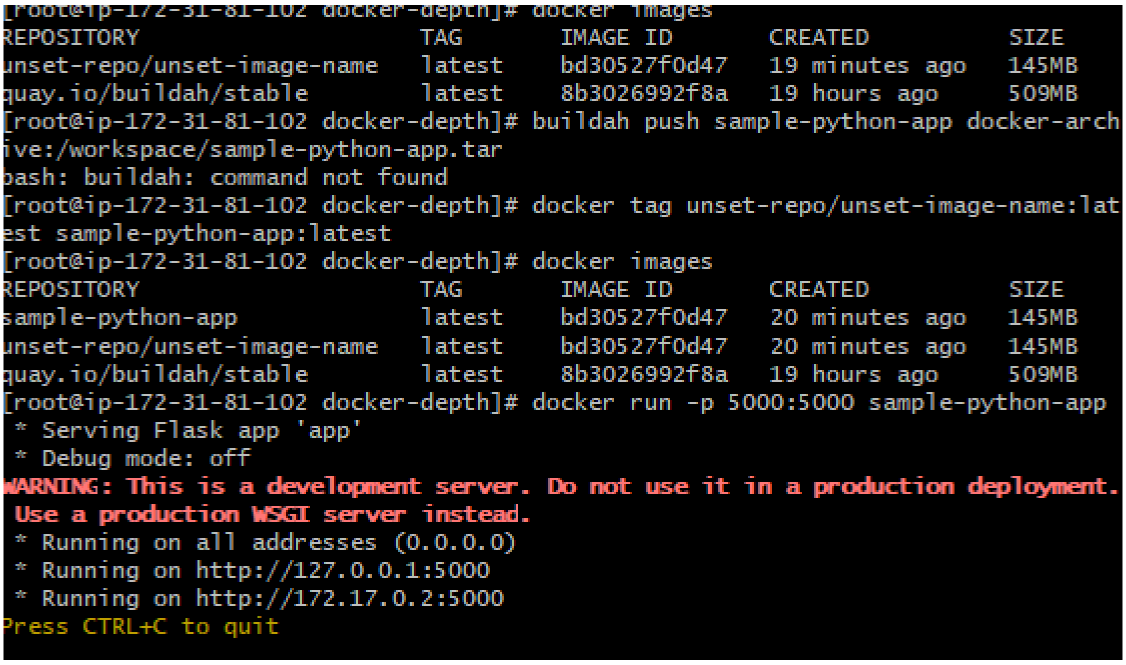
4. On port 5000 access the app now, maybe the other containers would bee running and already taken the port. But try to stop them or use another port.


Hello from Docker build demo!

Buildah

1. Buildah is a daemonless tool that allows us to build OCI-compliant container images securely and without Docker.
2. It is ideal for rootless and scriptable builds, often used in automated pipelines or Podman-based environments.
3. Since we are using Amazon Linux without native Buildah support, we ran it inside a privileged Docker container and exported the image as a .tar file.
4. The image was then loaded into Docker on the host for testing and run like a regular Docker image.
5. The command used to rerun inside buildah container:

```
buildah push sample-python-app docker-archive:/workspace/sample-python-app.tar:sample-python-app:latest
```


```
[root@ip-172-31-81-102 docker-depth]# docker images
REPOSITORY                     TAG       IMAGE ID       CREATED            SIZE
unset-repo/unset-image-name    latest    bd30527f0d47   19 minutes ago    145MB
quay.io/buildah/stable         latest    8b3026992f8a   19 hours ago      509MB
[root@ip-172-31-81-102 docker-depth]# buildah push sample-python-app docker-arch
ive:/workspace/sample-python-app.tar
bash: buildah: command not found
[root@ip-172-31-81-102 docker-depth]# docker tag unset-repo/unset-image-name:lat
est sample-python-app:latest
[root@ip-172-31-81-102 docker-depth]# docker images
REPOSITORY                     TAG       IMAGE ID       CREATED            SIZE
sample-python-app              latest    bd30527f0d47   20 minutes ago    145MB
unset-repo/unset-image-name    latest    bd30527f0d47   20 minutes ago    145MB
quay.io/buildah/stable         latest    8b3026992f8a   19 hours ago      509MB
[root@ip-172-31-81-102 docker-depth]# docker run -p 5000:5000 sample-python-app
 * Serving Flask app 'app'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment.
 Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://172.17.0.2:5000
Press CTRL+C to quit
```