

SECURE SYSTEMS ENGINEERING

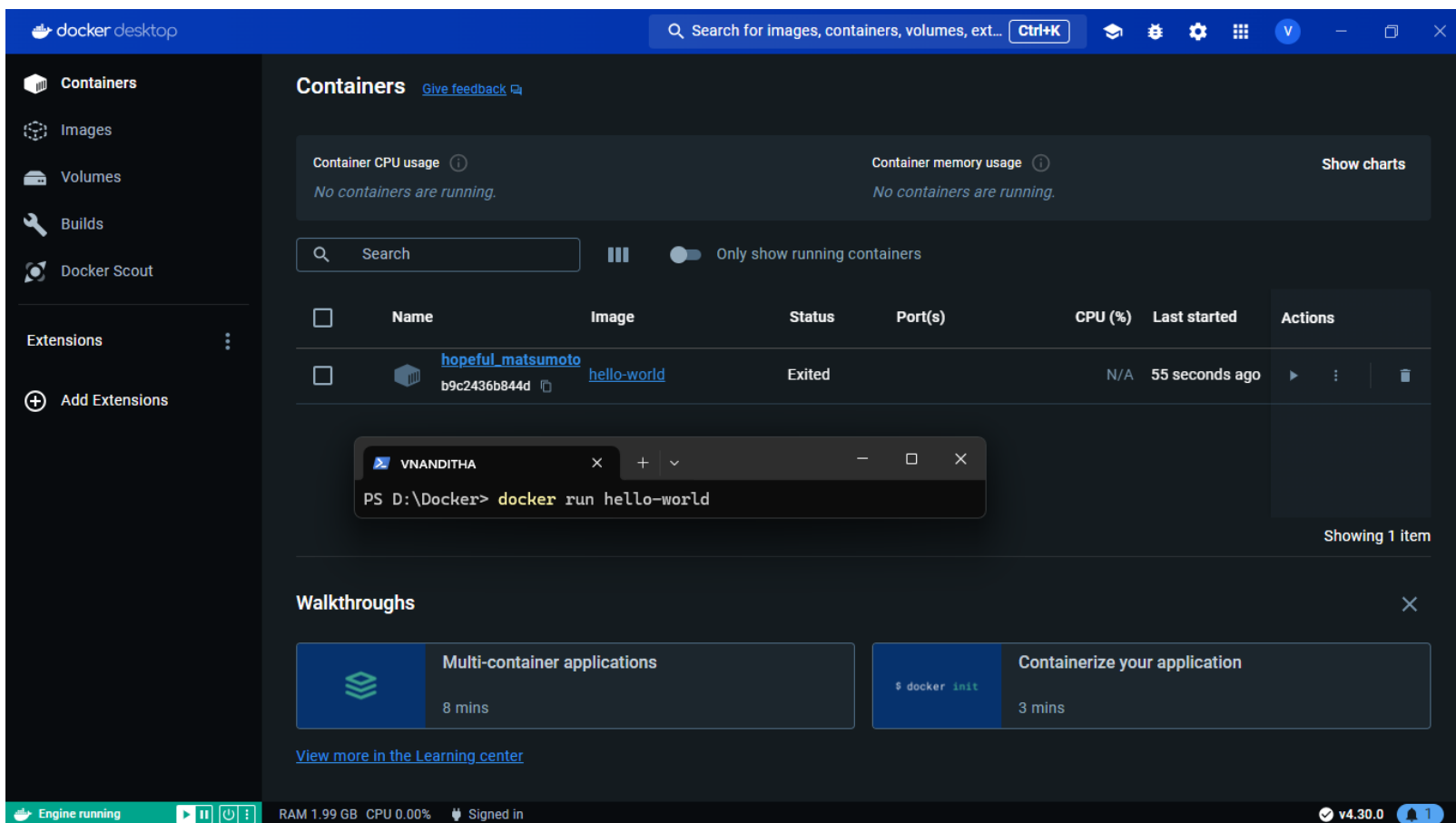
Assignment – 3 (DOCKER & KUBERNETES)

V NANDITHA
CB.SC.P2CYS23018

(2nd Year / 3rd Sem- MTech, CYBERSECURITY)

1) Create and use a Docker container interactively

- Open Docker Desktop, Terminal where the Folder is Stored.
- RUN : `docker run hello-world`



```
PS D:\Docker> docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:266b191e926f65542fa8daaec01a192c4d292bff79426f4730a046e1bc576fd
Status: Downloaded newer image for hello-world:latest

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.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/

PS D:\Docker>
```

Output When Typed: `docker run hello-world`

SUCCESSFULLY CREATED A DOCKER CONTAINER

- RUN : `docker run -it ubuntu:xenial /bin/bash`

The screenshot shows the Docker Desktop application. The left sidebar contains navigation options: Containers, Images, Volumes, Builds, Docker Scout, Extensions, and Add Extensions. The main area is titled 'Containers' and shows two containers: 'hopeful_matsumoto' (image: hello-world) and 'adoring_shtern' (image: ubuntu:xenial). Both are in an 'Exited' state. A terminal window titled 'VNANDITHA' is open, showing the command `docker run -it ubuntu:xenial /bin/bash` and its output, which includes pulling the 'ubuntu:xenial' image from the library. The bottom status bar indicates the engine is running, with RAM at 2.21 GB and CPU at 0.50%.

Name	Image	Status	Port(s)	CPU (%)	Last started	Actions
hopeful_matsumoto b9c2436b844d	hello-world	Exited		N/A	14 minutes ago	[Restart] [Logs] [Delete]
adoring_shtern 7da120d5f659	ubuntu:xenial	Exited		N/A	14 seconds ago	[Restart] [Logs] [Delete]

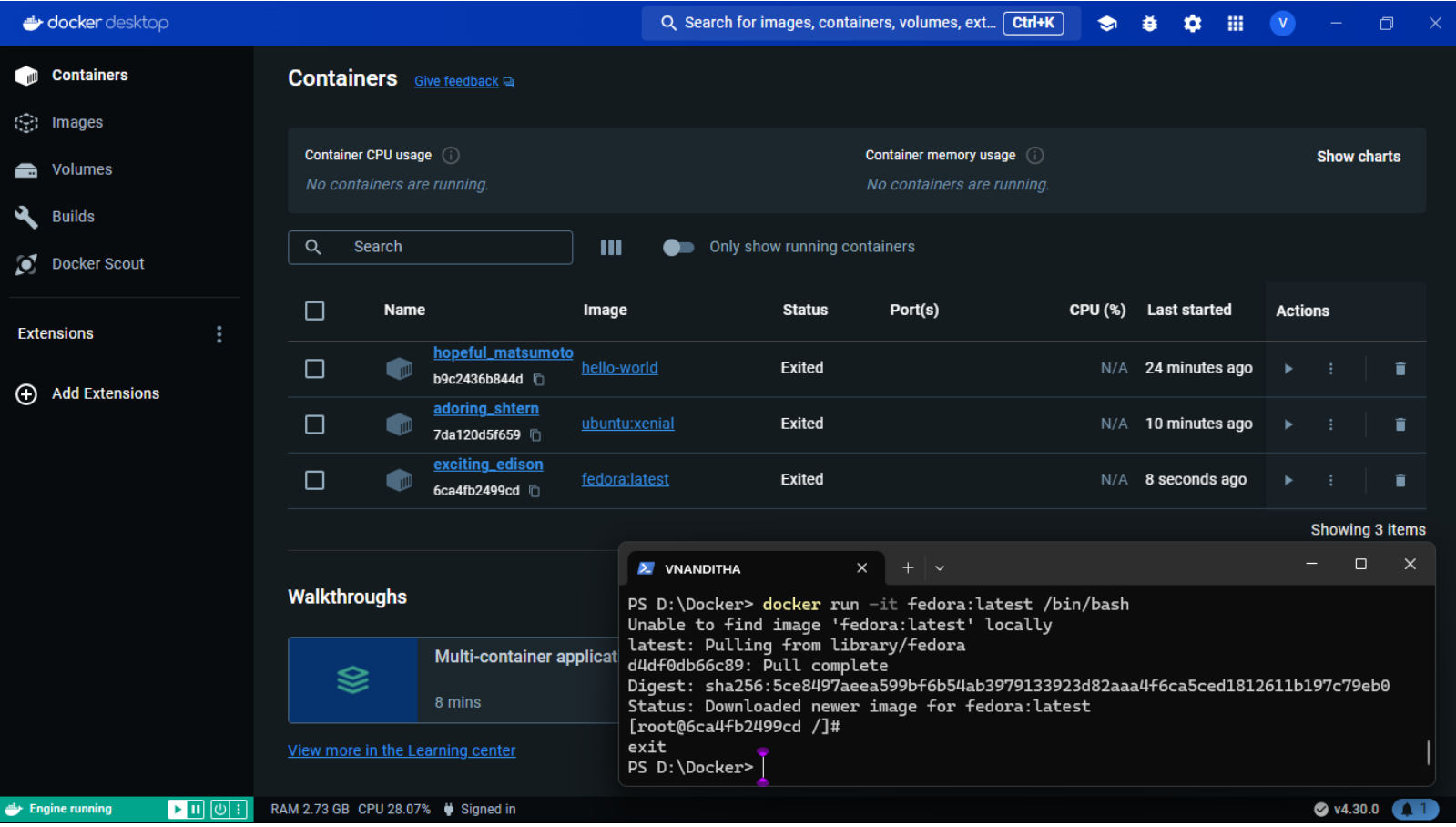
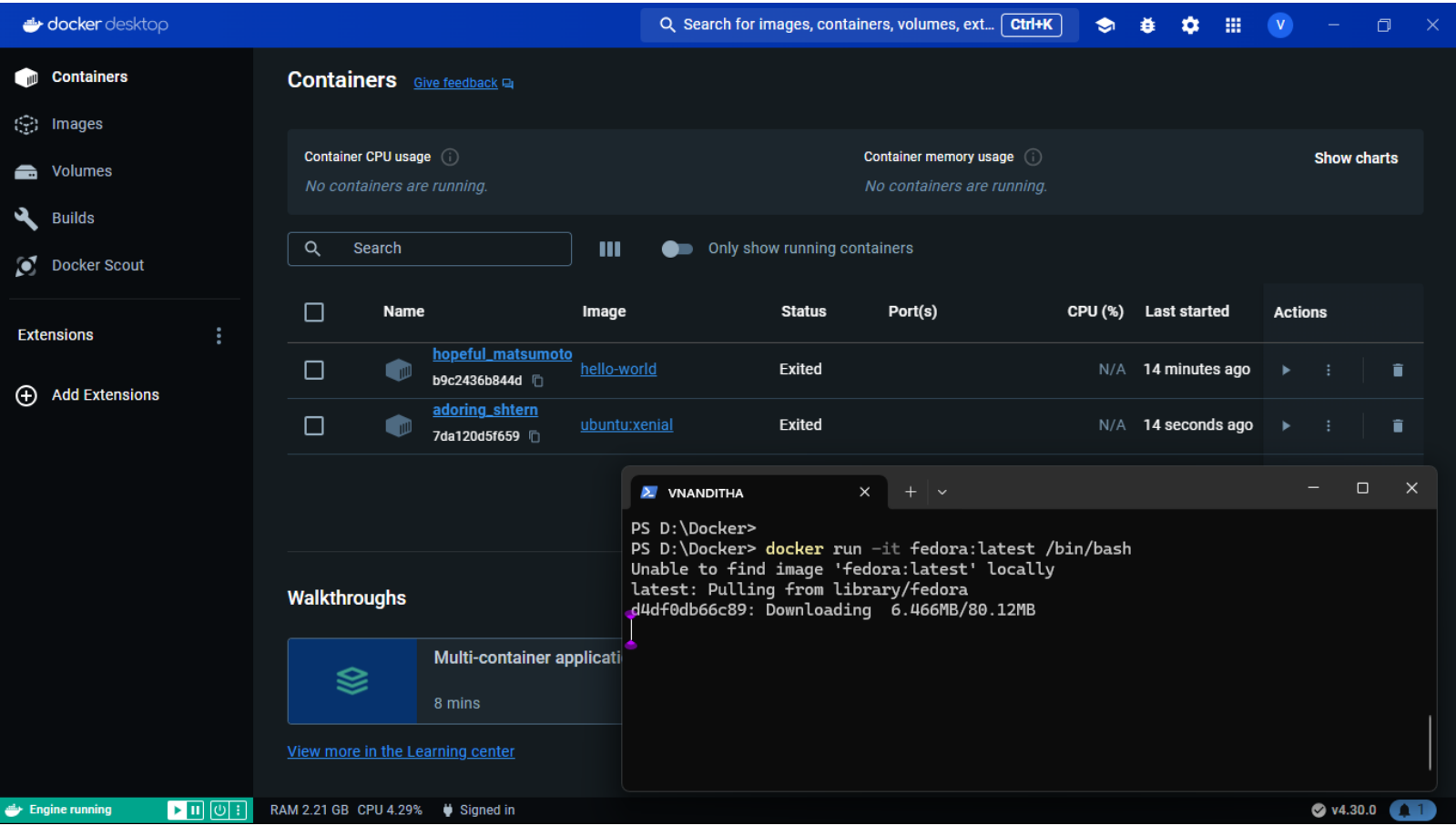
```
PS D:\Docker> docker run -it ubuntu:xenial /bin/bash
Unable to find image 'ubuntu:xenial' locally
xenial: Pulling from library/ubuntu
58690f9b18fc: Pull complete
b51569e7c507: Pull complete
da8ef40b9eca: Pull complete
fb15d46c38dc: Pull complete
Digest: sha256:1f1a2d56de1d604801a9671f301190704c25d604a416f59e03c04f5c6ffee0d6
Status: Downloaded newer image for ubuntu:xenial
root@7da120d5f659:/#
root@7da120d5f659:/# exit
PS D:\Docker>
```

The `-i` flag tells docker to keep STDIN open to your container, and the `-t` flag allocates a pseudo-TTY for you. Basically, you need both for you to have a way to enter text and have this display properly. At the end of the command, `/bin/bash` is just the command you want to run once the container starts up.

Notice how even though your VM is running Debian, you were able to run the Xenial version of Ubuntu in a container.

This all works because Linux distributions all share the Linux kernel. For that same reason, you won't be able to run MacOS or Windows in a container.

We can Observe that the Containers are being Interactive.



SUCCESSFULLY RAN AN INTERACTIVE DOCKER CONTAINER

2) Create a Docker file, which allows you to declaratively define your containers

- To see the running containers on a system : `docker ps` `docker ps -a`

```
PS D:\Docker> docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS        NAMES
6ca4fb2499cd   fedora:latest  "/bin/bash"             8 minutes ago  Exited (0)    6 minutes ago  exciting_edison
7da120d5f659   ubuntu:xenial  "/bin/bash"             18 minutes ago Exited (0)    16 minutes ago  adoring_shtern
b9c2436b844d   hello-world    "/hello"                 32 minutes ago Exited (0)    30 minutes ago  hopeful_matsumoto
```

- To see the logs : `docker logs <name or id>`

```
PS D:\Docker> docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS        NAMES
6ca4fb2499cd   fedora:latest  "/bin/bash"             8 minutes ago  Exited (0)    6 minutes ago  exciting_edison
7da120d5f659   ubuntu:xenial  "/bin/bash"             18 minutes ago Exited (0)    16 minutes ago  adoring_shtern
b9c2436b844d   hello-world    "/hello"                 32 minutes ago Exited (0)    30 minutes ago  hopeful_matsumoto

PS D:\Docker> docker logs 6ca4fb2499cd
[root@6ca4fb2499cd /]#
exit

PS D:\Docker> docker logs adoring_shtern
root@7da120d5f659:/#
root@7da120d5f659:/# exit
PS D:\Docker>
```

- To remove the Container : `docker rm <name or id>`

```
PS D:\Docker> docker rm b9c2436b844d
b9c2436b844d

PS D:\Docker> docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS        NAMES
6ca4fb2499cd   fedora:latest  "/bin/bash"             3 hours ago    Exited (0)    3 hours ago  exciting_edison
7da120d5f659   ubuntu:xenial  "/bin/bash"             3 hours ago    Exited (0)    3 hours ago  adoring_shtern
```

- It even gets removed from the Docker Desktop Application






- To get Docker Images : `docker images`

```
PS D:\Docker>
PS D:\Docker> docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
fedora         latest    5e22da79803c   5 weeks ago    222MB
hello-world    latest    d2c94e258dcb   13 months ago  13.3kB
ubuntu        xenial    b6f507652425   2 years ago    135MB
PS D:\Docker>
```

- To remove Docker Image : `docker rmi <image_id>`

```
PS D:\Docker> docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
fedora         latest    5e22da79803c   5 weeks ago    222MB
hello-world    latest    d2c94e258dcb   13 months ago  13.3kB
ubuntu        xenial    b6f507652425   2 years ago    135MB
PS D:\Docker>
PS D:\Docker> docker rmi d2c94e258dcb
Untagged: hello-world:latest
Untagged: hello-world@sha256:266b191e926f65542fa8daaec01a192c4d292bfff79426f47300a046e1bc576fd
Deleted: sha256:d2c94e258dcb3c5ac2798d32e1249e42ef01cba4841c2234249495f87264ac5a
Deleted: sha256:ac28809ec8bb38d5c35b49d45a6ac4777544941199075dff8c4eb63e093aa81e
PS D:\Docker>
PS D:\Docker> docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
fedora         latest    5e22da79803c   5 weeks ago    222MB
ubuntu        xenial    b6f507652425   2 years ago    135MB
PS D:\Docker>
```

- Initializing Docker

	.dockerignore	03-06-2024 22:07	DOCKERIGNORE F...	1 KB
	compose	03-06-2024 22:07	Yaml Source File	2 KB
	Docker Desktop Installer	03-06-2024 14:42	Application	4,87,594 KB
	Dockerfile	03-06-2024 22:07	File	2 KB
	README.Docker	03-06-2024 22:07	Markdown Source...	1 KB

- These files get created in the Dockers Directory

```

V NANDITHA
PS D:\Docker> docker init

Welcome to the Docker Init CLI!

This utility will walk you through creating the following files with sensible defaults for your project:
- .dockerignore
- Dockerfile
- compose.yaml
- README.Docker.md

Let's get started!

? What application platform does your project use? Python
? What version of Python do you want to use? (3.11.9) 3.11.9
? What version of Python do you want to use? 3.11.9
? What port do you want your app to listen on? (8000) 8000
? What port do you want your app to listen on? 8000
? What is the command you use to run your app (e.g., gunicorn 'myapp.example:app' --bind=0.0.0.0:8000)?

X Sorry, your reply was invalid: Value is required
? What is the command you use to run your app (e.g., gunicorn 'myapp.example:app' --bind=0.0.0.0:8000)? gunicorn 'myapp.example:app' --bind=0.0.0.0:8000

✓ Created → .dockerignore
✓ Created → Dockerfile
✓ Created → compose.yaml
✓ Created → README.Docker.md

+ Your Docker files are ready!
Review your Docker files and tailor them to your application.
Consult README.Docker.md for information about using the generated files.

! Warning → No requirements.txt file found. Create one with the dependencies for your application, including an entry for the gunicorn package, before running it.

What's next?
Start your application by running → docker compose up --build
Your application will be available at http://localhost:8000
PS D:\Docker>
```

- Building the file : `docker build -t mypython:latest .`

```

V NANDITHA
PS D:\Docker> python -m venv venv
PS D:\Docker> .\venv\Scripts\activate
(venv) PS D:\Docker> pip freeze > requirements.txt
(venv) PS D:\Docker>
(venv) PS D:\Docker>
(venv) PS D:\Docker>
(venv) PS D:\Docker>
(venv) PS D:\Docker> ^C
(venv) PS D:\Docker>
(venv) PS D:\Docker> deactivate
PS D:\Docker> docker build -t mypython:latest .
[+] Building 30.6s (14/14) FINISHED
=> [internal] load build definition from Dockerfile                                docker:default
=> => transferring dockerfile: 1.69kB                                           0.2s
=> resolve image config for docker-image://docker.io/docker/dockerfile:1       0.1s
=> [auth] docker/dockerfile:pull token for registry-1.docker.io                2.4s
=> CACHED docker-image://docker.io/docker/dockerfile:1@sha256:a57df69d0ea827fb7266491f2813635de6f17269be881f696fbfdf2d83dda33e 0.0s
=> [internal] load metadata for docker.io/library/python:3.11.9-slim           0.0s
=> [auth] library/python:pull token for registry-1.docker.io                   2.1s
=> [internal] load .dockerignore                                                 0.0s
=> => transferring context: 671B                                                0.0s
=> [base 1/5] FROM docker.io/library/python:3.11.9-slim@sha256:fc39d2e68b554c3f0a5c8b8a776280c0b3d73b4c04b83dbade835e2a171ca27ef 0.0s
=> [internal] load build context                                                8.8s
=> => transferring context: 12.03MB                                           8.8s
=> CACHED [base 2/5] WORKDIR /app                                              0.0s
=> CACHED [base 3/5] RUN adduser --disabled-password --gecos "" --home "/nonexistent" --shell "/sbin/nologin" --no-creat 0.0s
=> [base 4/5] RUN --mount=type=cache,target=/root/.cache/pip --mount=type=bind,source=requirements.txt,target=requirements.txt pytho 6.5s
=> [base 5/5] COPY . .                                                         5.0s
=> exporting to image                                                           5.0s
=> => exporting layers                                                         4.9s
=> => writing image sha256:07431ad0aa7d7d3765916336682546297c4f889ff4335c83d89cb51423a0488e 0.0s
=> => naming to docker.io/library/mypython:latest                             0.0s

What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview
PS D:\Docker>
```

```

VNANDITHA
PS D:\Docker> docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
mypythont     latest    07431ad0aa7d   4 minutes ago  642MB
mongo         latest    2f732130b5c3   12 days ago    797MB
fedora        latest    5e22da79803c   6 weeks ago    222MB
httpd         latest    356125da0595   2 months ago   147MB
ubuntu        xenial    b6f507652425   2 years ago    135MB
PS D:\Docker>

```

Interactively run the Docker Image and execute Python commands.

```

VNANDITHA
PS D:\Docker> .\venv\Scripts\activate
(venv) PS D:\Docker> pip install gunicorn
Collecting gunicorn
  Downloading gunicorn-22.0.0-py3-none-any.whl.metadata (4.4 kB)
Collecting packaging (from gunicorn)
  Using cached packaging-24.0-py3-none-any.whl.metadata (3.2 kB)
Downloading gunicorn-22.0.0-py3-none-any.whl (84 kB)
 84.4/84.4 kB 215.7 kB/s eta 0:00:00
Using cached packaging-24.0-py3-none-any.whl (53 kB)
Installing collected packages: packaging, gunicorn
Successfully installed gunicorn-22.0.0 packaging-24.0
(venv) PS D:\Docker> deactivate
PS D:\Docker>

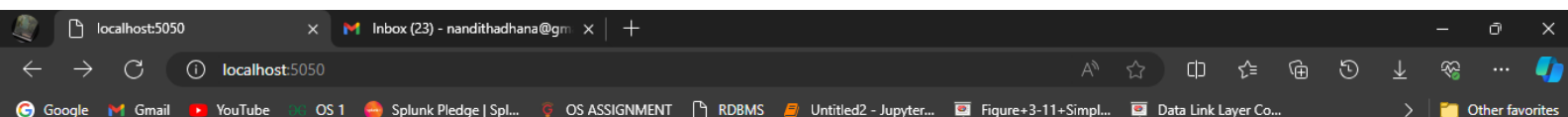
```


3) Run detached containers and understand port forwarding

- Docker creates a separate virtual network for containers, so you will need to do forward your host port to your container's port (this is called Port Forwarding, or port mapping). The container is listening on port 80, so let's try to forward our host machine's port 5050 to the container's port 80 when we run the container : `docker run -d -p=5050:80 httpd`

```
PS D:\Docker> docker run -d -p=5050:80 httpd
Unable to find image 'httpd:latest' locally
latest: Pulling from library/httpd
09f376ebb190: Pull complete
dab55b4abfc3: Pull complete
4f4fb700ef54: Pull complete
1a6d0283f224: Pull complete
1abf9110528c: Pull complete
7bacb8f85f3a: Pull complete
Digest: sha256:43c7661a3243c04b0955c81ac994ea13a1d8a1e53c15023a7b3cd5e8bb25de3c
Status: Downloaded newer image for httpd:latest
c341191a1f801063a28cce12752e9868fdee8cb3fd26f94108ffc06f4cdf9eea
PS D:\Docker>
```

```
PS D:\Docker>
PS D:\Docker> docker run -d -p=5050:80 httpd
fc21666f9cea806c75442d3268d2c397cab67a746aa382f30d66c17900575d8e
PS D:\Docker>
```

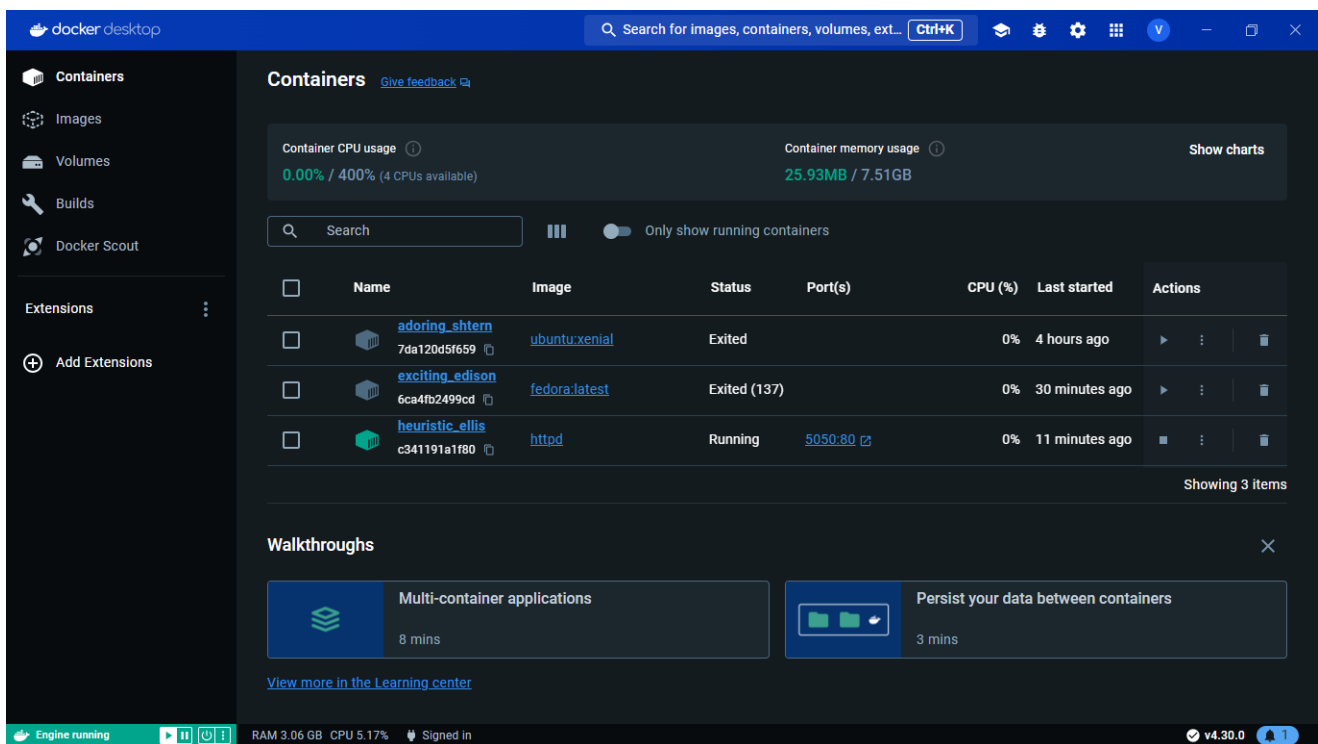


It works!

- To Stop a Running Process of Container : `docker stop <id>`

```
PS D:\Docker> docker stop 6ca4fb2499cd
6ca4fb2499cd
PS D:\Docker>
```

- For Restarting : `docker restart <container_id_or_name>`



4) Use docker-compose to run a multi-container web application

- Run MongoDB within a container in detached mode :

```
docker run -d mongo:latest
```

```
PS D:\Docker> docker run -d mongo:latest
Unable to find image 'mongo:latest' locally
latest: Pulling from library/mongo
a8b1c5f80c2d: Pulling fs layer
4e1f192fe085: Pulling fs layer
a8b1c5f80c2d: Pull complete
4e1f192fe085: Pull complete
329c0ae46358: Pull complete
a41d483e5ca3: Pull complete
e9ce4a7bbe77: Pull complete
57884d3aea66: Pull complete
5117a1be0e20: Pull complete
4a739f30e1d2: Pull complete
Digest: sha256:8f9f843d383e358d9be2f172ba9d2455e8736f3c59b00330da1f1b44273ce267
Status: Downloaded newer image for mongo:latest
135cb54fd6daf0ccf83def918070ac256972749030d9c3ad200a9f1e0f149f0b
PS D:\Docker> |
```

```
PS D:\Docker> docker-compose version
Docker Compose version v2.27.0-desktop.2
PS D:\Docker> |
```

The NodeJS project is downloaded from the provided location by typing the below commands : `git clone https://github.com/0xcf/decal-sp18-a10.git`

```
PS D:\Docker> git clone https://github.com/0xcf/decal-sp18-a10.git
Cloning into 'decal-sp18-a10'...
remote: Enumerating objects: 479, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 479 (delta 0), reused 2 (delta 0), pack-reused 476
Receiving objects: 100% (479/479), 64.96 KiB | 251.00 KiB/s, done.
Resolving deltas: 100% (168/168), done.
PS D:\Docker> |
```

```
File Edit Selection View Go Run ... Search
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

compose.yaml docker-compose.yaml Dockerfile

D: > Docker > docker-compose.yaml
1 version: '3'
2 services:
3   web:
4     build:
5       context: decal-sp18-a10 # Path to your Node.js app Dockerfile
6       dockerfile: Dockerfile
7     ports:
8       - "80:8080" # Port mapping for Node.js app
9     depends_on:
10      - database # Ensure MongoDB container is started first
11   database:
12     image: mongo:latest # MongoDB official Docker image
13     environment:
14       MONGO_INITDB_ROOT_USERNAME: root
15       MONGO_INITDB_ROOT_PASSWORD: example
16     ports:
17       - "27017:27017" # Port mapping for MongoDB
18     volumes:
19       - mongo_data:/data/db # Volume for MongoDB data persistence
20
21 volumes:
22   mongo_data:
```

```
File Edit Selection View Go Run ... Search
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

compose.yaml docker-compose.yaml Dockerfile X

D: > Docker > Dockerfile
1 syntax=docker/dockerfile:1
2
3 # Comments are provided throughout this file to help you get started.
4 # If you need more help, visit the Dockerfile reference guide at
5 # https://docs.docker.com/go/dockerfile-reference/
6
7 # Want to help us make this template better? Share your feedback here: https://forms.gle/ybq9KrtBj1BL31ck?
8
9 ARG PYTHON_VERSION=3.11.9
10 FROM python:${PYTHON_VERSION}-slim as base
11
12 # Prevents Python from writing pyc files.
13 ENV PYTHONUNBUFFERED=1
14
15 # Keeps Python from buffering stdout and stderr to avoid situations where
16 # the application crashes without emitting any logs due to buffering.
17 ENV PYTHONUNBUFFERED=1
18
19 WORKDIR /app
20
21 # Create a non-privileged user that the app will run under.
22 # See https://docs.docker.com/go/dockerfile-user-best-practices/
23 ARG UID=10001
24 RUN adduser \
25   --disabled-password \
26   --gecos "" \
27   --home "/nonexistent" \
28   --shell "/sbin/nologin" \
29   --no-create-home \
30   --uid "${UID}" \
31   appuser
32
33 # Download dependencies as a separate step to take advantage of Docker's caching.
```

```
File Edit Selection View Go Run ... Search
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

compose.yaml docker-compose.yaml Dockerfile X

D: > Docker > Dockerfile
24 RUN adduser \
25   --disabled-password \
26   --gecos "" \
27   --home "/nonexistent" \
28   --shell "/sbin/nologin" \
29   --no-create-home \
30   --uid "${UID}" \
31   appuser
32
33 # Download dependencies as a separate step to take advantage of Docker's caching.
34 # Leverage a cache mount to /root/.cache/pip to speed up subsequent builds.
35 # Leverage a bind mount to requirements.txt to avoid having to copy them into
36 # into this layer.
37 RUN --mount=type=cache,target=/root/.cache/pip \
38     --mount=type=bind,source=requirements.txt,target=requirements.txt \
39     python -m pip install -r requirements.txt
40
41 # Switch to the non-privileged user to run the application.
42 USER appuser
43
44 # Copy the source code into the container.
45 COPY . .
46
47 # Copy the rest of your application files into the container
48 COPY . /app
49
50
51 # Expose the port that the application listens on.
52 EXPOSE 8080
53
54 # Run the application.
55 CMD gunicorn 'myapp.example:app' --bind=0.0.0.0:8080
56
```

```
PS D:\Docker> docker-compose up
time="2024-06-05T14:44:18+05:30" level=warning msg="Found multiple config files with supported names: D:\\Docker\\compose.yaml, D:\\Docker\\docker-c
ompose.yaml"
time="2024-06-05T14:44:18+05:30" level=warning msg="Using D:\\Docker\\compose.yaml"
time="2024-06-05T14:44:18+05:30" level=warning msg="Found multiple config files with supported names: D:\\Docker\\compose.yaml, D:\\Docker\\docker-c
ompose.yaml"
time="2024-06-05T14:44:18+05:30" level=warning msg="Using D:\\Docker\\compose.yaml"
[+] Building 32.9s (15/15) FINISHED
=> [server internal] load build definition from Dockerfile
=> => transferring dockerfile: 1.76kB
=> [server] resolve image config for docker-image://docker.io/docker/dockerfile:1
=> [server auth] docker/dockerfile:pull token for registry-1.docker.io
=> CACHED [server] docker-image://docker.io/docker/dockerfile:1@sha256:a57df69d0ea827fb7266491f2813635de6f17269be881f696fbfdf2d83dda33e
=> [server internal] load metadata for docker.io/library/python:3.11.9-slim
=> [server auth] library/python:pull token for registry-1.docker.io
=> [server internal] load .dockerignore
=> => transferring context: 671B
=> [server base 1/6] FROM docker.io/library/python:3.11.9-slim@sha256:fc39d2e68b554c3f0a5cb8a776280c0b3d73b4c04b83dbade835e2a171ca27ef
=> [server internal] load build context
=> => transferring context: 84.16kB
=> CACHED [server base 2/6] WORKDIR /app
=> CACHED [server base 3/6] RUN adduser --disabled-password --gecos "" --home "/nonexistent" --shell "/sbin/nologin" --n
=> CACHED [server base 4/6] RUN --mount=type=cache,target=/root/.cache/pip --mount=type=bind,source=requirements.txt,target=requirements
=> [server base 5/6] COPY . .
=> [server base 6/6] COPY . /app
=> [server] exporting to image
=> => exporting layers
=> => writing image sha256:0fae46e907212cc010fb3a2edca9b0c09dcdb0078ab733ddb8e901afacd23cf
=> => naming to docker.io/library/docker-server
[+] Running 2/2
✔Network docker_default Created
✔Container docker-server-1 Created
Attaching to server-1
server-1 | [2024-06-05 09:14:58 +0000] [7] [INFO] Starting gunicorn 22.0.0
server-1 | [2024-06-05 09:14:58 +0000] [7] [INFO] Listening at: http://0.0.0.0:8000 (7)
server-1 | [2024-06-05 09:14:58 +0000] [7] [INFO] Using worker: sync
server-1 | [2024-06-05 09:14:58 +0000] [8] [INFO] Booting worker with pid: 8
```

I'm a Todo-aholic

- ☐
- ☐
- ☐
- ☐
- ☐
- ☐

I want to buy a puppy that will love me forever

Add

A demo by [Scotch](#).

Read the [tutorial](#).

5) Task 1: Run some simple Docker containers

My ROOT-ID : 192.168.0.18

Time: Approximately 30 minutes

Tasks:

- Task 0: Prerequisites
- Task 1: Run some simple Docker containers
- Task 2: Package and run a custom app using Docker
- Task 3: Modify a Running Website

Task 0: Prerequisites

You will need all of the following to complete this lab:

- A clone of the lab's GitHub repo.
- A DockerID.

Clone the Lab's GitHub Repo

Use the following command to clone the lab's repo from GitHub (you can click the command or manually type it). This will make a copy of the lab's repo in a new sub-directory called `linux_tweet_app`.

```
git clone https://github.com/dockerexamples/linux_tweet_app
```

Make sure you have a DockerID

If you do not have a DockerID (a free login used to access Docker Hub), please visit [Docker Hub](#) and register for one. You will need this for later steps.

If the commandline doesn't appear in the terminal, make sure popups are enabled or try resizing the browser window.

```
#####  
#                               #  
# WARNING!!!!                   #  
# This is a sandbox environment. Using personal credentials #  
# is HIGHLY! discouraged. Any consequences of doing so are #  
# completely the user's responsibilities.                   #  
# The PWD team.                                               #  
#####  
[node1] (local) root@192.168.0.18 ~  
$ git clone https://github.com/dockerexamples/linux_tweet_app  
Cloning into 'linux_tweet_app'...  
remote: Enumerating objects: 14, done.  
remote: Counting objects: 100% (8/8), done.  
remote: Compressing objects: 100% (4/4), done.  
remote: Total 14 (delta 4), reused 4 (delta 4), pack-reused 6  
Receiving objects: 100% (14/14), 10.79 KiB | 5.39 MiB/s, done.  
Resolving deltas: 100% (5/5), done.  
[node1] (local) root@192.168.0.18 ~  
$
```

Task 1: Run some simple Docker containers

There are different ways to use containers. These include:

1. **To run a single task:** This could be a shell script or a custom app.
2. **Interactively:** This connects you to the container similar to the way you SSH into a remote server.
3. **In the background:** For long-running services like websites and databases.

In this section you'll try each of those options and see how Docker manages the workload.

Run a single task in an Alpine Linux container

In this step we're going to start a new container and tell it to run the `hostname` command. The container will start, execute the `hostname` command, then exit.

1. Run the following command in your Linux console.

```
docker container run alpine hostname
```

The output below shows that the `alpine:latest` image could not be found locally. When this happens, Docker automatically *pulls* it from Docker Hub.

After the image is pulled, the container's hostname is displayed (`888e89a3b36b` in the example below).

```
Unable to find image 'alpine:latest' locally  
latest: Pulling from library/alpine  
88286f41530e: Pull complete  
Digest: sha256:f006ecbb824d87947d0b51ab8488634bf69fe4094959d935c0c103f4820a417d  
Status: Downloaded newer image for alpine:latest  
888e89a3b36b
```

If the commandline doesn't appear in the terminal, make sure popups are enabled or try resizing the browser window.

```
$  
[node1] (local) root@192.168.0.18 ~  
$  
[node1] (local) root@192.168.0.18 ~  
$  
[node1] (local) root@192.168.0.18 ~  
$  
[node1] (local) root@192.168.0.18 ~  
$  
[node1] (local) root@192.168.0.18 ~  
$  
[node1] (local) root@192.168.0.18 ~  
$  
[node1] (local) root@192.168.0.18 ~  
$  
[node1] (local) root@192.168.0.18 ~  
$  
[node1] (local) root@192.168.0.18 ~  
$ docker container run alpine hostname  
Unable to find image 'alpine:latest' locally  
docker container run alpine hostname  
latest: Pulling from library/alpine  
d25f557d7f31: Pull complete  
Digest: sha256:77726ef6b57ddf65bb551896826ec38bc3e53f75cdde31354fbffb4f25238ebd92ebf3e67ec2  
Status: Downloaded newer image for alpine:latest  
[node1] (local) root@192.168.0.18 ~  
$ docker container run alpine hostname  
9bf316d9fd45  
[node1] (local) root@192.168.0.18 ~  
$
```

2. Docker keeps a container running as long as the process it started inside the container is still running. In this case the `hostname` process exits as soon as the output is written. This means the container stops. However, Docker doesn't delete resources by default, so the container still exists in the `Exited` state.

List all containers.

```
docker container ls --all
```

Notice that your Alpine Linux container is in the `Exited` state.

CONTAINER ID	IMAGE	PORTS	COMMAND	NAMES	CREATE
D	STATUS				
888e89a3b36b	alpine		"hostname"		50 sec
onds ago	Exited (0) 49 seconds ago				aweso
me_elion					

Note: The container ID is the hostname that the container displayed. In the example above it's `888e89a3b36b`.

Containers which do one task and then exit can be very useful. You could build a Docker image that executes a script to configure something. Anyone can execute that task just by running the container - they don't need the actual scripts or configuration information.

Run an interactive Ubuntu container

You can run a container based on a different version of Linux than is running on your Docker host.

In the next example, we are going to run an Ubuntu Linux container on top of an Alpine Linux Docker host (Play With Docker uses Alpine Linux for its nodes).

If the commandline doesn't appear in the terminal, make sure popups are enabled or try resizing the browser window.

```
$
[node1] (local) root@192.168.0.18 ~
$
[node1] (local) root@192.168.0.18 ~
$
[node1] (local) root@192.168.0.18 ~
$
[node1] (local) root@192.168.0.18 ~
$ docker container run alpine hostname
Unable to find image 'alpine:latest' locally
docker container run alpine hostname
latest: Pulling from library/alpine
d25f557d7f31: Pull complete
Digest: sha256:77726ef6b57ddf65bb551896826ec38bc3e53f75cdde31354fbfffb4f25238ebd
Status: Downloaded newer image for alpine:latest
92ebf3e67ec2
[node1] (local) root@192.168.0.18 ~
$ docker container run alpine hostname
9bf316d9fd45
[node1] (local) root@192.168.0.18 ~
$ docker container ls --all
CONTAINER ID   IMAGE     COMMAND                  CREATED              STATUS              PORTS          NAMES
9bf316d9fd45   alpine    "hostname"              About a minute ago   Exited (0)          About a m
inute ago      eloquent_gagarin
92ebf3e67ec2   alpine    "hostname"              About a minute ago   Exited (0)          About a m
inute ago      reverent_cray
[node1] (local) root@192.168.0.18 ~
$
```

1. Run a Docker container and access its shell.

```
docker container run --interactive --tty --rm ubuntu bash
```

In this example, we're giving Docker three parameters:

- `--interactive` says you want an interactive session.
- `--tty` allocates a pseudo-tty.
- `--rm` tells Docker to go ahead and remove the container when it's done executing.

The first two parameters allow you to interact with the Docker container.

We're also telling the container to run `bash` as its main process (PID 1).

When the container starts you'll drop into the bash shell with the default prompt `root@container id>:/#`. Docker has attached to the shell in the container, relaying input and output between your local session and the shell session in the container.

2. Run the following commands in the container.

`ls /` will list the contents of the root directory in the container, `ps aux` will show running processes in the container, `cat /etc/issue` will show which Linux distro the container is running, in this case Ubuntu 20.04.3 LTS.

```
ls /
```

```
ps aux
```

```
cat /etc/issue
```

3. Type `exit` to leave the shell session. This will terminate the `bash` process, causing the container to exit.

If the commandline doesn't appear in the terminal, make sure popups are enabled or try resizing the browser window.

```
92ebf3e67ec2
[node1] (local) root@192.168.0.18 ~
$ docker container run alpine hostname
9bf316d9fd45
[node1] (local) root@192.168.0.18 ~
$ docker container ls --all
CONTAINER ID   IMAGE     COMMAND                  CREATED              STATUS              PORTS          NAMES
9bf316d9fd45   alpine    "hostname"              About a minute ago   Exited (0)          About a m
inute ago      eloquent_gagarin
92ebf3e67ec2   alpine    "hostname"              About a minute ago   Exited (0)          About a m
inute ago      reverent_cray
[node1] (local) root@192.168.0.18 ~
$ docker container run --interactive --tty --rm ubuntu bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
00d679a470c4: Pull complete
Digest: sha256:e3f92abc0967a6c19d0dfa2d55838833e947b9d74edbc0113e48535ad4be12a
Status: Downloaded newer image for ubuntu:latest
root@c9339599d435:/# ls/
bash: ls/: No such file or directory
root@c9339599d435:/# ps aux
USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.4  0.0   4580  4128 pts/0    Ss   09:46   0:00 bash
root         9  0.0  0.0   7880  4116 pts/0    R+   09:47   0:00 ps aux
root@c9339599d435:/# cat /etc/issue
Ubuntu 20.04 LTS \n \l
root@c9339599d435:/#
```


3. Type `exit` to leave the shell session. This will terminate the `bash` process, causing the container to exit.

```
exit
```

Note: As we used the `--rm` flag when we started the container, Docker removed the container when it stopped. This means if you run another `docker container ls --all` you won't see the Ubuntu container.

4. For fun, let's check the version of our host VM.

```
cat /etc/issue
```

You should see:

```
Welcome to Alpine Linux 3.8
Kernel \r on an \m (\l)
```

Notice that our host VM is running Alpine Linux, yet we were able to run an Ubuntu container. As previously mentioned, the distribution of Linux inside the container does not need to match the distribution of Linux running on the Docker host.

However, Linux containers require the Docker host to be running a Linux kernel. For example, Linux containers cannot run directly on Windows Docker hosts. The same is true of Windows containers - they need to run on a Docker host with a Windows kernel.

Interactive containers are useful when you are putting together your own image. You can run a container and verify all the steps you need to deploy your app, and capture them in a Dockerfile.

You can commit a container to make an image from it, but you

If the commandline doesn't appear in the terminal, make sure popups are enabled or try resizing the browser window.

```
9bf316d9fd45  alpine  "hostname"  About a minute ago  Exited (0)  About a m
inute ago      eloquent_gagarin
92ebf3e67ec2  alpine  "hostname"  About a minute ago  Exited (0)  About a m
inute ago      reverent_cray
[node1] (local) root@192.168.0.18 ~
$ docker container run --interactive --tty --rm ubuntu bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
00d679a470c4: Pull complete
Digest: sha256:e3f92abc0967a6c19d0dfa2d55838833e947b9d74edbc0113e48535ad4be12a
Status: Downloaded newer image for ubuntu:latest
root@c9339599d435:/# ls/
bash: ls/: No such file or directory
root@c9339599d435:/# ps aux
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.4  0.0   4580  4128 pts/0    Ss   09:46   0:00 bash
root         9  0.0  0.0   7880  4116 pts/0    R+   09:47   0:00 ps aux
root@c9339599d435:/# cat /etc/issue
Ubuntu 24.04 LTS \n \l
```

```
root@c9339599d435:/# exit
exit
[node1] (local) root@192.168.0.18 ~
$ cat /etc/issue
Welcome to Alpine Linux 3.18
Kernel \r on an \m (\l)
[node1] (local) root@192.168.0.18 ~
$
```

Run a background MySQL container

Background containers are how you'll run most applications. Here's a simple example using MySQL.

1. Run a new MySQL container with the following command.

```
docker container run \
--detach \
--name mydb \
-e MYSQL_ROOT_PASSWORD=my-secret-pw \
mysql:latest
```

- `--detach` will run the container in the background.
- `--name` will name it `mydb`.
- `-e` will use an environment variable to specify the root password (NOTE: This should never be done in production).

As the MySQL image was not available locally, Docker automatically pulled it from Docker Hub.

```
Unable to find image 'mysql:latest' locallylatest: Pulling from li
brary/mysql
aa18ad1a0d33: Pull complete
fdb8d83dece3: Pull complete
75b6ce7b50d3: Pull complete
ed1d0a3a64e4: Pull complete
8eb36a82c85b: Pull complete
41be6f1a1c40: Pull complete
0e1b414eac71: Pull complete
914c28654a91: Pull complete
587693eb988c: Pull complete
b183c3585729: Pull complete
315e21657aa4: Pull complete
Digest: sha256:0dc3dacb751ef46a6647234abdec2d4740f0dfbe77ab490b02
```

If the commandline doesn't appear in the terminal, make sure popups are enabled or try resizing the browser window.

```
[node1] (local) root@192.168.0.18 ~
$ docker container run \
> --detach \
> --name mydb \
> -e MYSQL_ROOT_PASSWORD=my-secret-pw \
> docker container run \
> --detach \
> --name mydb \
> -e MYSQL_ROOT_PASSWORD=my-secret-pw \
> mysql:latest
Unable to find image 'docker:latest' locally
latest: Pulling from library/docker
d25f557d7f31: Already exists
709fdef6a978: Pull complete
4f4fb700ef54: Pull complete
aee0e04186d5: Pull complete
5040cb473dd8: Pull complete
033d787e83ee: Pull complete
ca0e4ad1d5fb: Pull complete
add3d3fa802ed: Pull complete
fa82f68d1319: Pull complete
945935a84494: Pull complete
eb849552362f: Pull complete
0b8dd1c36edf: Pull complete
0c2644864cc4: Pull complete
934d8d2105be: Pull complete
51f324d3ecf3: Pull complete
a4382da9867c: Pull complete
```

```
a4382da9867c: Pull complete
Digest: sha256:76ba10a4aed708c7b2db09d45740d711edf707f7368f6808bd32a53eae33404
Status: Downloaded newer image for docker:latest
6a65da86cc54e8d371845f02a05111ea72d2d61f67daf45e2fd7e190a7943d6f
[node1] (local) root@192.168.0.18 ~
$
```


6) Task 2 : Package & Run a custom app using Docker

Task 2: Package and run a custom app using Docker

In this step you'll learn how to package your own apps as Docker images using a [Dockerfile](#).

The Dockerfile syntax is straightforward. In this task, we're going to create a simple NGINX website from a Dockerfile.

Build a simple website image

Let's have a look at the Dockerfile we'll be using, which builds a simple website that allows you to send a tweet.

1. Make sure you're in the `linux_tweet_app` directory.

```
cd ~/linux_tweet_app
```

2. Display the contents of the Dockerfile.

```
cat Dockerfile
```

```
FROM nginx:latest

COPY index.html /usr/share/nginx/html
COPY linux.png /usr/share/nginx/html

EXPOSE 80 443

CMD ["nginx", "-g", "daemon off;"]
```

Let's see what each of these lines in the Dockerfile do.

- [FROM](#) specifies the base image to use as the starting point for this new image you're

If the commandline doesn't appear in the terminal, make sure popups are enabled or try resizing the browser window.

```
$ cd ~/linux_tweet_app
[node1] (local) root@192.168.0.18 ~/linux_tweet_app
$ cat Dockerfile
FROM nginx:latest

COPY index.html /usr/share/nginx/html
COPY linux.png /usr/share/nginx/html

EXPOSE 80 443

CMD ["nginx", "-g", "daemon off;"]
[node1] (local) root@192.168.0.18 ~/linux_tweet_app
$
[node1] (local) root@192.168.0.18 ~/linux_tweet_app
$
[node1] (local) root@192.168.0.18 ~/linux_tweet_app
$
[node1] (local) root@192.168.0.18 ~/linux_tweet_app
$
[node1] (local) root@192.168.0.18 ~/linux_tweet_app
$
[node1] (local) root@192.168.0.18 ~/linux_tweet_app
$
[node1] (local) root@192.168.0.18 ~/linux_tweet_app
$
[node1] (local) root@192.168.0.18 ~/linux_tweet_app
$
[node1] (local) root@192.168.0.18 ~/linux_tweet_app
```

3. In order to make the following commands more copy/paste friendly, export an environment variable containing your DockerID (if you don't have a DockerID you can get one for free via [Docker Hub](#)).

You will have to manually type this command as it requires your unique DockerID.

```
export DOCKERID=<your docker id>
```

4. Echo the value of the variable back to the terminal to ensure it was stored correctly.

```
echo $DOCKERID
```

5. Use the `docker image build` command to create a new Docker image using the instructions in the Dockerfile.

- `--tag` allows us to give the image a custom name. In this case it's comprised of our DockerID, the application name, and a version. Having the Docker ID attached to the name will allow us to store it on Docker Hub in a later step
- `.` tells Docker to use the current directory as the build context

Be sure to include period (`.`) at the end of the command.

```
docker image build --tag $DOCKERID/linux_tweet_app:1.0 .
```

The output below shows the Docker daemon executing each line in the Dockerfile

```
Sending build context to Docker daemon 32.77kB
Step 1/5 : FROM nginx:latest
latest: Pulling from library/nginx
afeb2bfd31c0: Pull complete
7ff5d10493db: Pull complete
d2562f1ae1d0: Pull complete
Digest: sha256:af32e714a9cc3157157374e68c818b05ebe9e0737aac06b55a09da374209a8f9
Status: Downloaded newer image for nginx:latest
--> da5939581ac8
Step 2/5 : COPY index.html /usr/share/nginx/html
--> sha2a6c2baa9
```

If the commandline doesn't appear in the terminal, make sure popups are enabled or try resizing the browser window.

```
$
[node1] (local) root@192.168.0.18 ~/linux_tweet_app
$ echo $DOCKERID

[node1] (local) root@192.168.0.18 ~/linux_tweet_app
$ docker image build --tag $DOCKERID/linux_tweet_app:1.0 .
[+] Building 0.0s (0/0) docker:default
ERROR: invalid tag "/linux_tweet_app:1.0": invalid reference format
[node1] (local) root@192.168.0.18 ~/linux_tweet_app
$ echo $DOCKERID

[node1] (local) root@192.168.0.18 ~/linux_tweet_app
$ docker image build --tag $DOCKERID/linux_tweet_app:1.0 .
[+] Building 0.0s (0/0) docker:default
ERROR: invalid tag "/linux_tweet_app:1.0": invalid reference format
[node1] (local) root@192.168.0.18 ~/linux_tweet_app
$ docker container run \
> --detach \
> --publish 80:80 \
> --name linux_tweet_app \
> $DOCKERID/linux_tweet_app:1.0
docker: invalid reference format.
See 'docker run --help'.
[node1] (local) root@192.168.0.18 ~/linux_tweet_app
$ docker container rm --force linux_tweet_app
Error response from daemon: No such container: linux_tweet_app
[node1] (local) root@192.168.0.18 ~/linux_tweet_app
$
```

6. Use the `docker container run` command to start a new container from the image you created.

As this container will be running an NGINX web server, we'll use the `--publish` flag to publish port 80 inside the container onto port 80 on the host. This will allow traffic coming in to the Docker host on port 80 to be directed to port 80 in the container. The format of the `--publish` flag is `host_port:container_port`.

```
docker container run \
--detach \
--publish 80:80 \
--name linux_tweet_app \
$DOCKERID/linux_tweet_app:1.0
```

Any external traffic coming into the server on port 80 will now be directed into the container on port 80.

In a later step you will see how to map traffic from two different ports - this is necessary when two containers use the same port to communicate since you can only expose the port once on the host.

7. Click [here](#) to load the website which should be running.

8. Once you've accessed your website, shut it down and remove it.

```
docker container rm --force linux_tweet_app
```

Note: We used the `--force` parameter to remove the running container without shutting it down. This will ungracefully shutdown the container and permanently remove it from the Docker host.

In a production environment you may want to use `docker container stop` to gracefully stop the container and leave it on the host. You can then use `docker container rm` to permanently remove it.

If the commandline doesn't appear in the terminal, make sure popups are enabled or try resizing the browser window.

```
$
[node1] (local) root@192.168.0.18 ~/linux_tweet_app
$ echo $DOCKERID

[node1] (local) root@192.168.0.18 ~/linux_tweet_app
$ docker image build --tag $DOCKERID/linux_tweet_app:1.0 .
[+] Building 0.0s (0/0) docker:default
ERROR: invalid tag "/linux_tweet_app:1.0": invalid reference format
[node1] (local) root@192.168.0.18 ~/linux_tweet_app
$ echo $DOCKERID

[node1] (local) root@192.168.0.18 ~/linux_tweet_app
$ docker image build --tag $DOCKERID/linux_tweet_app:1.0 .
[+] Building 0.0s (0/0) docker:default
ERROR: invalid tag "/linux_tweet_app:1.0": invalid reference format
[node1] (local) root@192.168.0.18 ~/linux_tweet_app
$ docker container run \
> --detach \
> --publish 80:80 \
> --name linux_tweet_app \
> $DOCKERID/linux_tweet_app:1.0
docker: invalid reference format.
See 'docker run --help'.
[node1] (local) root@192.168.0.18 ~/linux_tweet_app
$ docker container rm --force linux_tweet_app
Error response from daemon: No such container: linux_tweet_app
[node1] (local) root@192.168.0.18 ~/linux_tweet_app
$
```

Linux Tweet App!



You've successfully deployed the Linux tweet app! Why not let the world know?

[Tweet](#)

7) Task 3: Modify the Running Website

Modify the running website

Bind mounts mean that any changes made to the local file system are immediately reflected in the running container.

1. Copy a new `index.html` into the container.

The Git repo that you pulled earlier contains several different versions of an `index.html` file. You can manually run an `ls` command from within the `~/linux_tweet_app` directory to see a list of them. In this step we'll replace `index.html` with `index-new.html`.

```
cp index-new.html index.html
```

2. Go to the running website and refresh the page. Notice that the site has changed.

If you are comfortable with `vi` you can use it to load the local `index.html` file and make additional changes. Those too would be reflected when you reload the webpage. If you are really adventurous, why not try using `exec` to access the running container and modify the files stored there.

Even though we've modified the `index.html` local filesystem and seen it reflected in the running container, we've not actually changed the Docker image that the container was started from.

To show this, stop the current container and re-run the `1.0` image without a bind mount.

1. Stop and remove the currently running container.

```
docker rm --force linux_tweet_app
```

2. Rerun the current version without a bind mount.

```
docker container run \
```

If the commandline doesn't appear in the terminal, make sure popups are enabled or try resizing the browser window.

Username: vnanditha

Password:

Error response from daemon: Get "https://registry-1.docker.io/v2/": unauthorized: incorrect username or password

```
[node1] (local) root@192.168.0.18 ~/linux_tweet_app
```

```
$ docker login
```

Log in with your Docker ID or email address to push and pull images from Docker Hub. If you don't have a Docker ID, head over to <https://hub.docker.com/> to create one.

You can log in with your password or a Personal Access Token (PAT). Using a limited-scope PAT grants better security and is required for organizations using SSO. Learn more at <https://docs.docker.com/go/access-tokens/>

Username: cyberops17

Password:

Error response from daemon: Get "https://registry-1.docker.io/v2/": unauthorized: incorrect username or password

```
[node1] (local) root@192.168.0.18 ~/linux_tweet_app
```

```
$ docker image push $DOCKERID/linux_tweet_app:1.0
invalid reference format
```

```
[node1] (local) root@192.168.0.18 ~/linux_tweet_app
```

```
cp index-new.html index.html
```

```
[node1] (local) root@192.168.0.18 ~/linux_tweet_app
```

```
$ docker rm --force linux_tweet_app
```

Error response from daemon: No such container: linux_tweet_app

```
[node1] (local) root@192.168.0.18 ~/linux_tweet_app
```

```
$
```

Test the new version

1. Run a new container from the new version of the image.

```
docker container run \
--detach \
--publish 80:80 \
--name linux_tweet_app \
$DOCKERID/linux_tweet_app:2.0
```

2. Check the new version of the website (You may need to refresh your browser to get the new version to load).

The web page will have an orange background.

We can run both versions side by side. The only thing we need to be aware of is that we cannot have two containers using port 80 on the same host.

As we're already using port 80 for the container running from the `2.0` version of the image, we will start a new container and publish it on port 8080. Additionally, we need to give our container a unique name (`old_linux_tweet_app`)

3. Run another new container, this time from the old version of the image.

Notice that this command maps the new container to port 8080 on the host. This is because two containers cannot map to the same port on a single Docker host.

```
docker container run \
--detach \
--publish 8080:80 \
--name old_linux_tweet_app \
$DOCKERID/linux_tweet_app:1.0
```

4. View the old version of the website.

```
>> [2/3] COPY index.html /usr/share/nginx/html 0.1s
>> [3/3] COPY linux.png /usr/share/nginx/html 0.1s
>> exporting to image 0.0s
>> exporting layers 0.0s
>> writing image sha256:38816e0c9671297a899553a01ba2f2d057d00f8862fc400edc1535789426081 0.0s
>> naming to docker.io/shrinadhav/linux_tweet_app:2.0 0.0s
[node1] (local) root@192.168.0.8 ~/linux_tweet_app
$ docker image ls
REPOSITORY          TAG          IMAGE ID       CREATED        SIZE
shrinadhav/linux_tweet_app  2.0         38816e0c9671   6 seconds ago  188MB
shrinadhav/linux_tweet_app  1.0         ac471bf29ba8   10 minutes ago 188MB
[node1] (local) root@192.168.0.8 ~/linux_tweet_app
$ docker container run \
> --detach \
> --publish 80:80 \
> --name linux_tweet_app \
> $DOCKERID/linux_tweet_app:2.0
d935545b6cb54a3bc06f33630e0aaa287f83efa45767852586986dc666883fd9
[node1] (local) root@192.168.0.8 ~/linux_tweet_app
$ docker container run \
> --detach \
> --publish 8080:80 \
> --name old_linux_tweet_app \
> $DOCKERID/linux_tweet_app:1.0
96b48a1ccf7d673f9bc4e8ecaa3e4c5137a613369bda29fd37a799993064b2c9
[node1] (local) root@192.168.0.8 ~/linux_tweet_app
$
```

Docker Hub is the default public registry for Docker images.

2. Before you can push your images, you will need to log into Docker Hub.

```
docker login
```

You will need to supply your Docker ID credentials when prompted.

```
Username: <your docker id>
Password: <your docker id password>
Login Succeeded
```

3. Push version 1.0 of your web app using `docker image push`.

```
docker image push $DOCKERID/linux_tweet_app:1.0
```

```
ub. If you don't have a Docker ID, head over to https://hub.docker.com/ to create
one.
You can log in with your password or a Personal Access Token (PAT). Using a limit
ed-scope PAT grants better security and is required for organizations using SSO.
Learn more at https://docs.docker.com/go/access-tokens/

Username: cyberops17
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
[node1] (local) root@192.168.0.8 ~/linux_tweet_app
$
```



Sort by

Newest ▾

Filter Tags

Q

Delete



TAG

1.0

docker pull cyberops17/linux_tweet_app:1.0 Copy

Last pushed a few seconds ago by cyberops17

Digest	OS/ARCH	Last pull	Compressed Size ⓘ
612063e0747f	linux/amd64	a few seconds ago	67.71 MB



TAG

2.0

docker pull cyberops17/linux_tweet_app:2.0 Copy

Last pushed a few seconds ago by cyberops17

Digest	OS/ARCH	Last pull	Compressed Size ⓘ
612063e0747f	linux/amd64	a few seconds ago	67.71 MB

KUBERNETES

INSTALLATION : `curl.exe -LO "https://dl.k8s.io/release/v1.30.0/bin/windows/amd64/kubect.exe"`

```
PS D:\Kubernetes> curl.exe -LO "https://dl.k8s.io/release/v1.30.0/bin/windows/amd64/kubect.exe"
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total   Spent    Left     Speed
100 138    100    138     0     0    102      0  0:00:01  0:00:01 --:--:--  102
100 50.3M  100 50.3M     0     0   511k      0  0:01:40  0:01:40 --:--:-- 447k
PS D:\Kubernetes>
```

CertUtil -hashfile kubect.exe SHA256

```
PS D:\Kubernetes>
PS D:\Kubernetes> CertUtil -hashfile kubect.exe SHA256
SHA256 hash of kubect.exe:
e0e72bf37bf563fdea4a6070b07e2fbaa818aa02ed38c5d10d9ce146106cab70
CertUtil: -hashfile command completed successfully.
PS D:\Kubernetes>
```

kubect version --client

```
PS D:\Kubernetes> kubect version --client
Client Version: v1.29.2
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
PS D:\Kubernetes>
PS D:\Kubernetes>
```

Open the downloaded directory and start minikube.exe

minikube.exe start

```
PS D:\Kubernetes\Minikube> minikube.exe start
W0605 15:50:50.024213 15548 main.go:291] Unable to resolve the current Docker CLI context "default": context "default"
: context not found: open C:\Users\sri\.docker\contexts\meta\37a8eec1ce19687d132fe29051dca629d164e2c4958ba141d5f4133a33f
0688f\meta.json: The system cannot find the path specified.
🐹 minikube v1.33.1 on Microsoft Windows 11 Pro 10.0.22631.3593 Build 22631.3593
🌟 Automatically selected the docker driver. Other choices: virtualbox, ssh
🚀 Using Docker Desktop driver with root privileges
👉 Starting "minikube" primary control-plane node in "minikube" cluster
📡 Pulling base image v0.0.44 ...
📦 Downloading Kubernetes v1.30.0 preload ...
> preloaded-images-k8s-v18-v1...: 22.31 MiB / 342.90 MiB 6.51% 663.38 KiB
```

Docker Container > Settings > Kubernetes > Enable Kubernetes > Apply Restart

docker desktop

Search for images, containers, volumes, ext... Ctrl+K

🎓

🐙

⚙️

🗃️

👤

—

🗑️

✖️

Settings

Give feedback

✖️

General

Resources

Docker Engine

Builders

Kubernetes

Software updates

Extensions

Features in development

Notifications

Kubernetes

v1.29.2

☒ Enable Kubernetes

Start a Kubernetes single-node cluster when starting Docker Desktop.

☐ Show system containers (advanced)

Show Kubernetes internal containers when using Docker commands.

Reset Kubernetes Cluster

All stacks and Kubernetes resources will be deleted.

Cancel

Apply & restart

Engine running

▶️

⏸️

🔌

⋮

RAM 4.46 GB CPU 0.00% 🐙 Signed in

✓ v4.30.0

🔔 2

docker desktop

Search for images, containers, volumes, ext... Ctrl+K

🎓

🐙

⚙️

🗃️

👤

—

🗑️

✖️

Settings

Give feedback

✖️

General

Resources

Docker Engine

Builders

Kubernetes

Software updates

Extensions

Features in development

Notifications

Kubernetes

v1.29.2

☒ Enable Kubernetes

Starting ... pulling images

☐ Show system containers (advanced)

Show Kubernetes internal containers when using Docker commands.

Reset Kubernetes Cluster

All stacks and Kubernetes resources will be deleted.

Cancel

Apply & restart

Engine running

▶️

⏸️

🔌

⋮

RAM 4.46 GB CPU 1.75% 🐙 Signed in

✓ v4.30.0

🔔 2

```
PS D:\> kubectl version
Client Version: v1.29.2
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
Server Version: v1.30.0
```

```
PS D:\> kubectl get pods --namespace=kube-system
```

NAME	READY	STATUS	RESTARTS	AGE
coredns-7db6d8ff4d-6vjvp	1/1	Running	0	94m
etcd-minikube	1/1	Running	0	94m
kube-apiserver-minikube	1/1	Running	1 (5m29s ago)	94m
kube-controller-manager-minikube	1/1	Running	0	94m
kube-proxy-bzkhg	1/1	Running	0	94m
kube-scheduler-minikube	1/1	Running	0	94m
storage-provisioner	1/1	Running	2 (5m18s ago)	94m

```
PS D:\> kubectl apply -f https://github.com/kubernetes/dashboard/raw/v2.7.0/aio/deploy/recommended.yaml
namespace/kubernetes-dashboard created
serviceaccount/kubernetes-dashboard created
service/kubernetes-dashboard created
secret/kubernetes-dashboard-certs created
secret/kubernetes-dashboard-csrf created
secret/kubernetes-dashboard-key-holder created
configmap/kubernetes-dashboard-settings created
role.rbac.authorization.k8s.io/kubernetes-dashboard created
clusterrole.rbac.authorization.k8s.io/kubernetes-dashboard created
rolebinding.rbac.authorization.k8s.io/kubernetes-dashboard created
clusterrolebinding.rbac.authorization.k8s.io/kubernetes-dashboard created
deployment.apps/kubernetes-dashboard created
service/dashboard-metrics-scraper created
deployment.apps/dashboard-metrics-scraper created
```

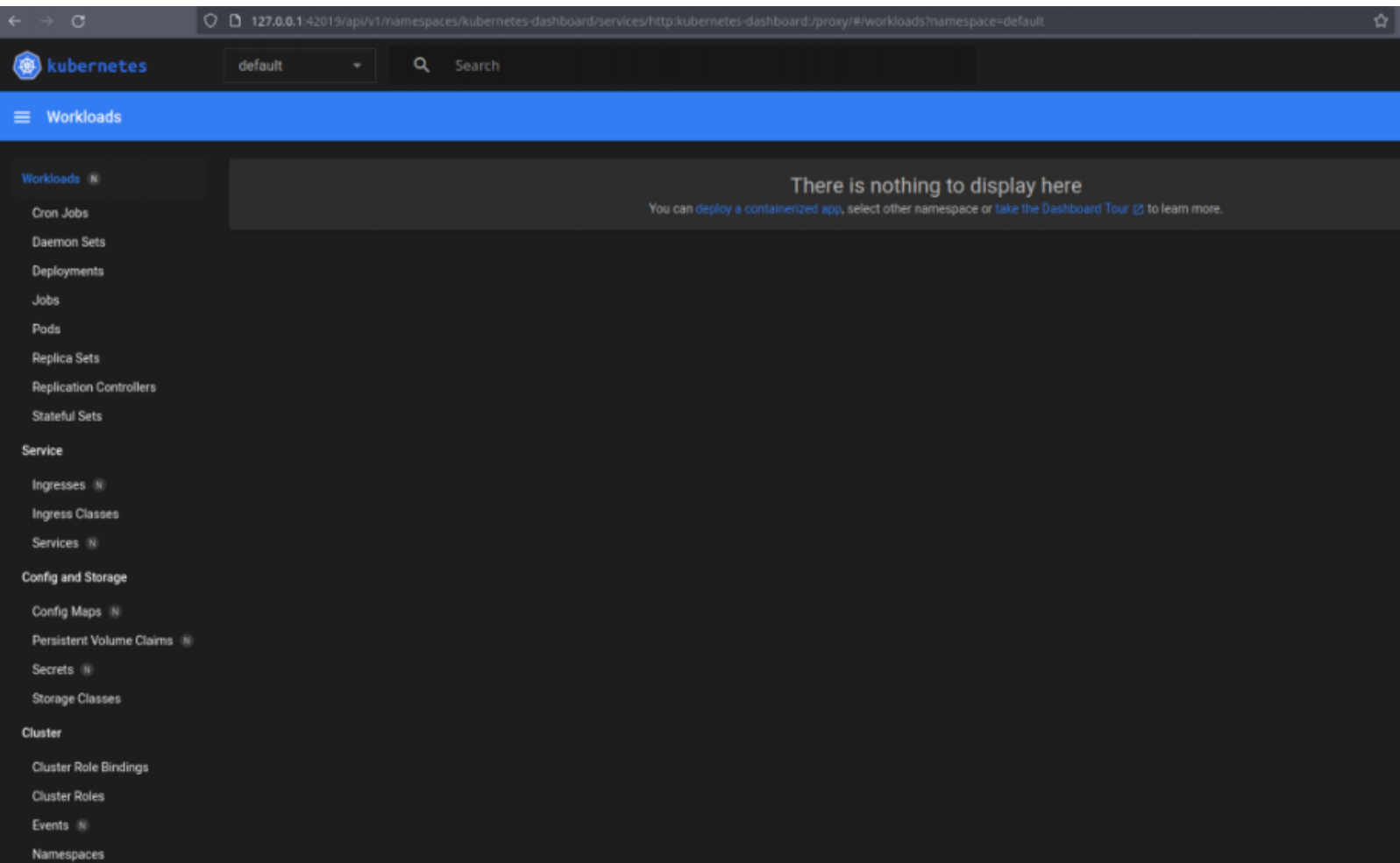
Kubernetes Dashboard

☒ Token
Every Service Account has a Secret with valid Bearer Token that can be used to log in to Dashboard. To find out more about how to configure and use Bearer Tokens, please refer to the [Authentication](#) section.

☐ Kubeconfig
Please select the kubeconfig file that you have created to configure access to the cluster. To find out more about how to configure and use kubeconfig file, please refer to the [Configure Access to Multiple Clusters](#) section.

Enter token *

Sign in



V NANDITHA
[CB.SC.P2CYS23018]