

Docker Fundamentals

Dr. Prashant Aparajeya

Docker

Docker is an open platform for developers and sysadmins to:

- build,
- ship, and
- run distributed applications,

whether on laptops, data center VMs, or the cloud.

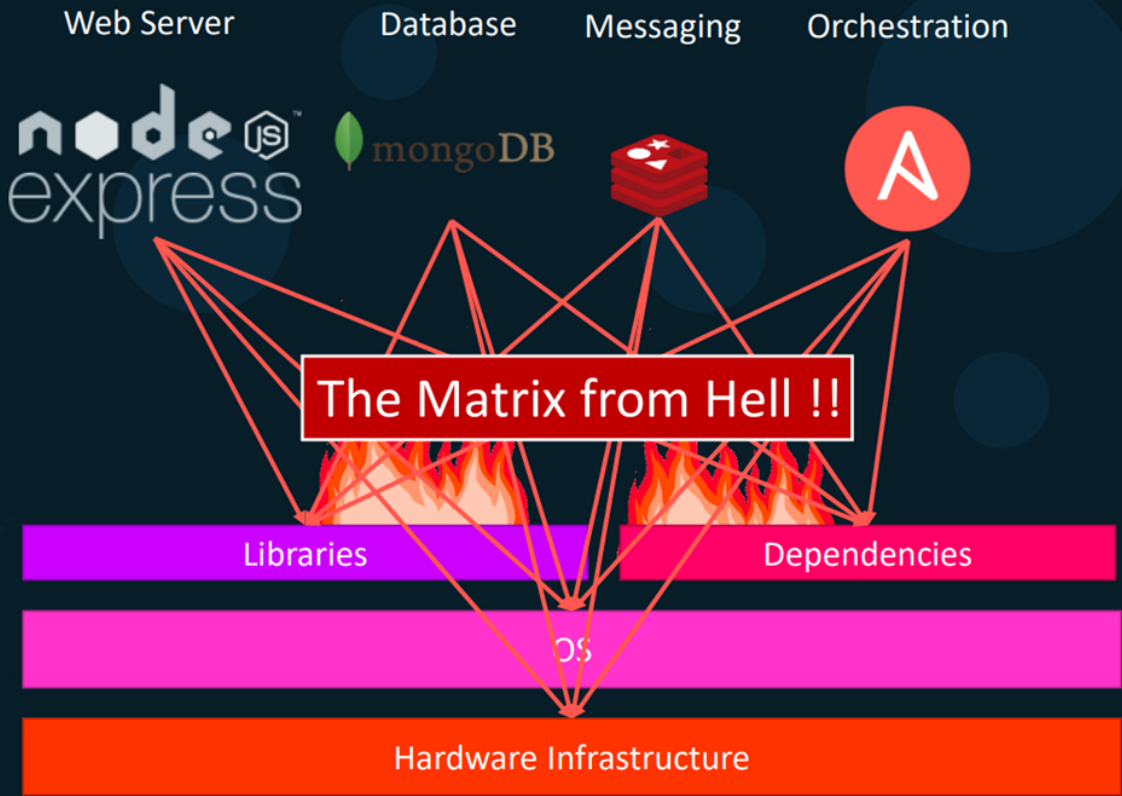


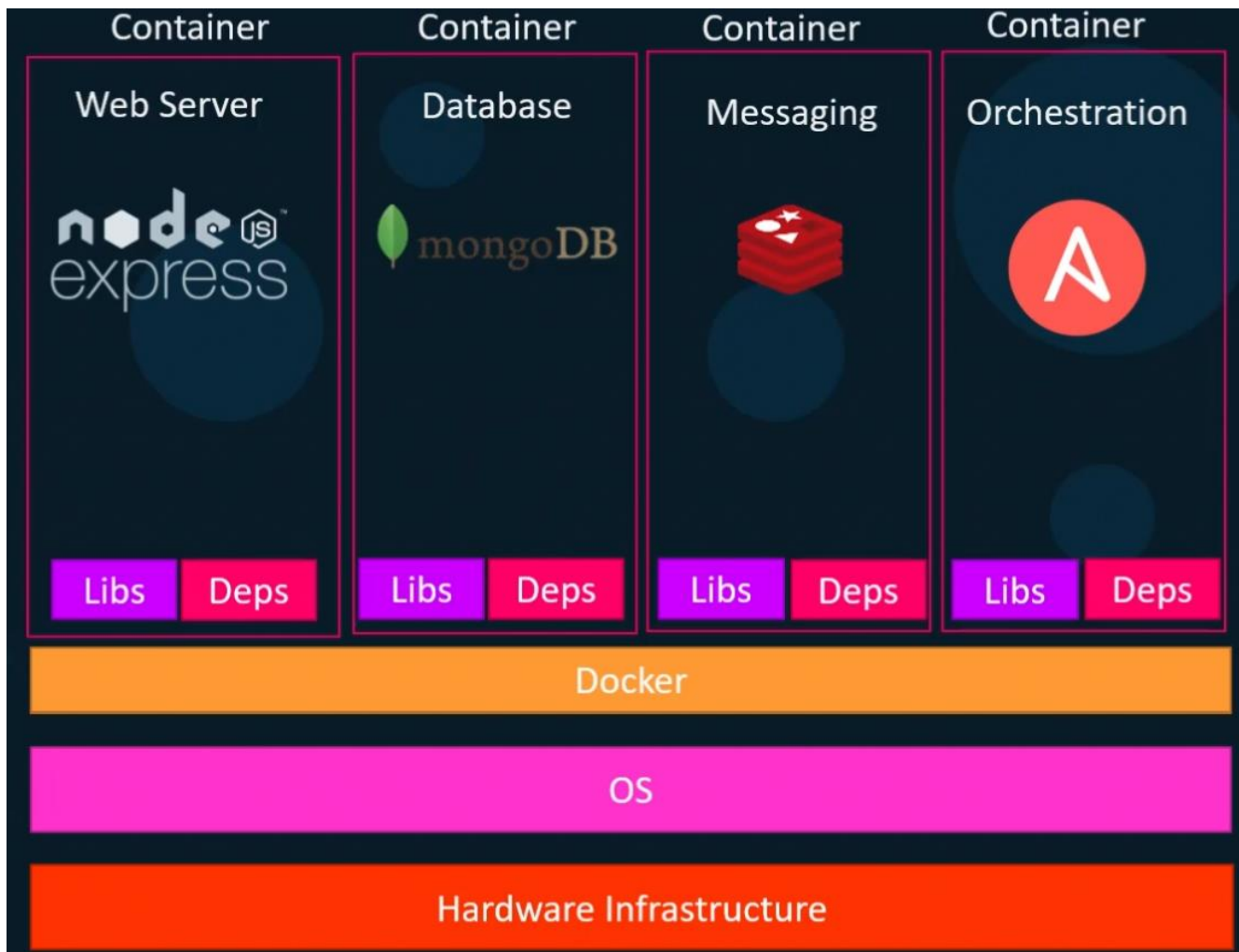
Objectives

1. What are Containers?
2. What is Docker?
3. Why do you need it?
4. What Can it do?
5. Run Docker Containers
6. Create a Docker Image

Why do you need docker?

- Compatibility/Dependency
- Long setup time
- Different Dev/Test/Prod environments





What can it do?

- Containerize the solution
- Run each service with its own dependencies in separate containers

Make sure the Docker is installed on the System

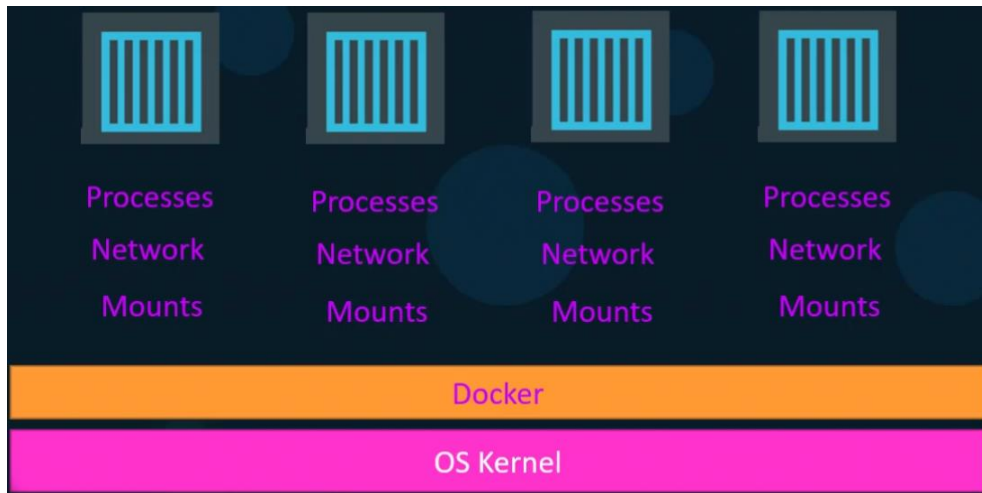
What are containers?

Containers are completely isolated environments as in they can have their own:

- Processes
- Network
- Mounts

NB: They all share the same OS kernel.

Docker is the host that runs containers.



Docker developed a **Linux container technology** – one that is portable, flexible and easy to deploy. Docker open sourced libcontainer and partnered with a worldwide community of contributors to further its development.

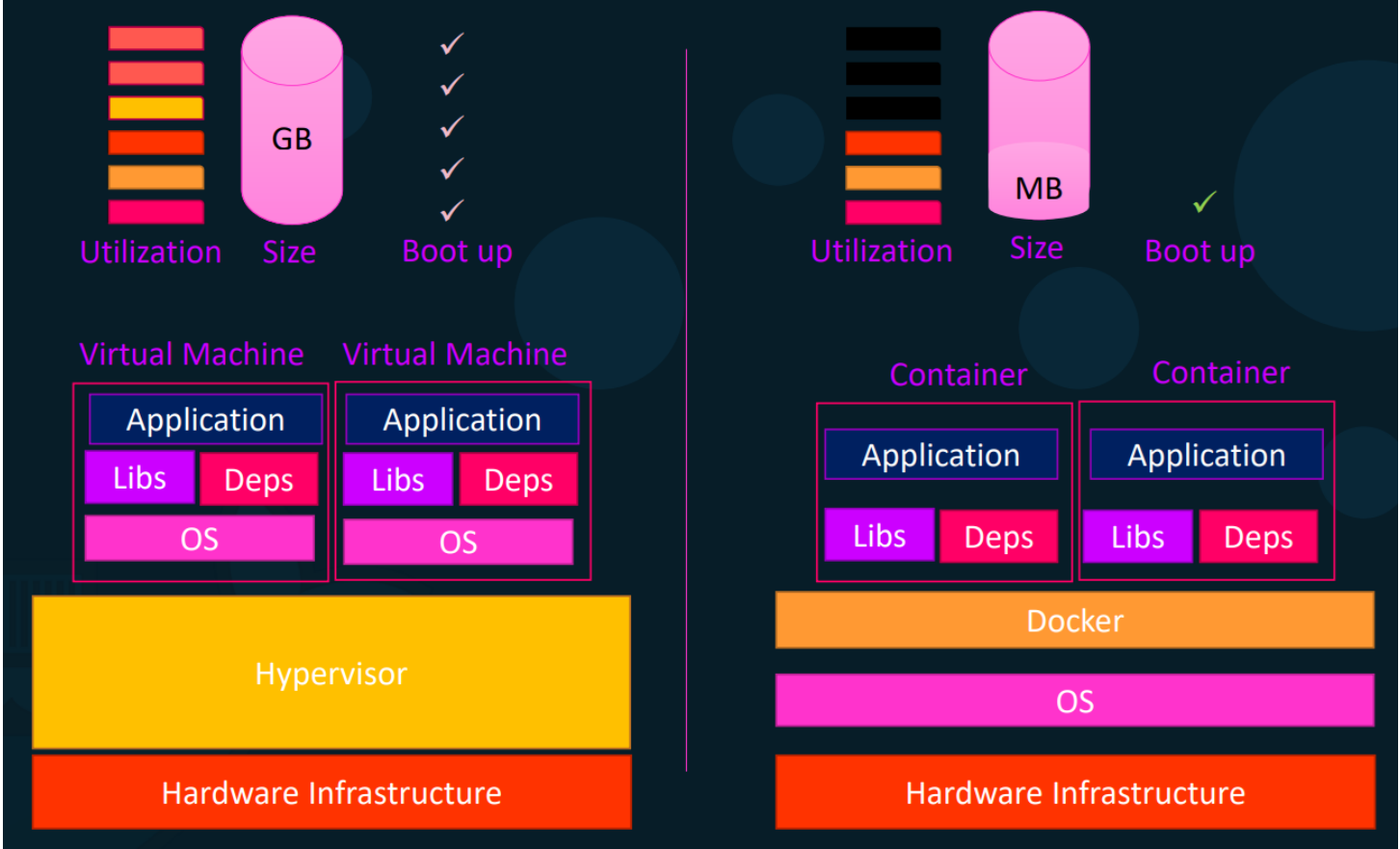
Tech Tips: Docker utilises LXC containers.

Purpose of Docker

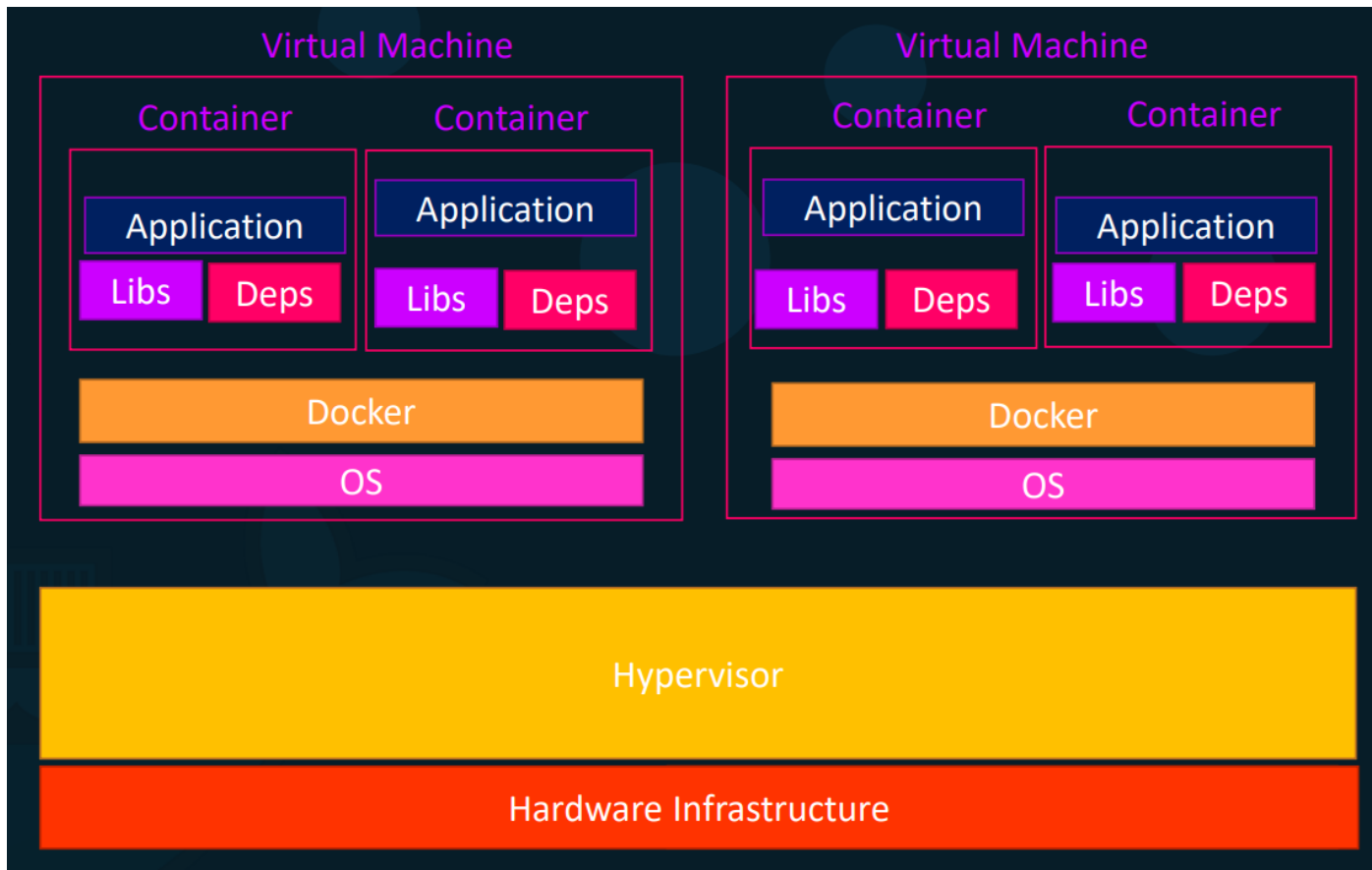
The main purpose of Docker is to:

- Package,
- Containerize Applications,
- Ship them
- Run them anywhere, anytime, as many times as you want.

Containers vs Virtual Machines



Containers & Virtual Machines



Installing and Running Docker

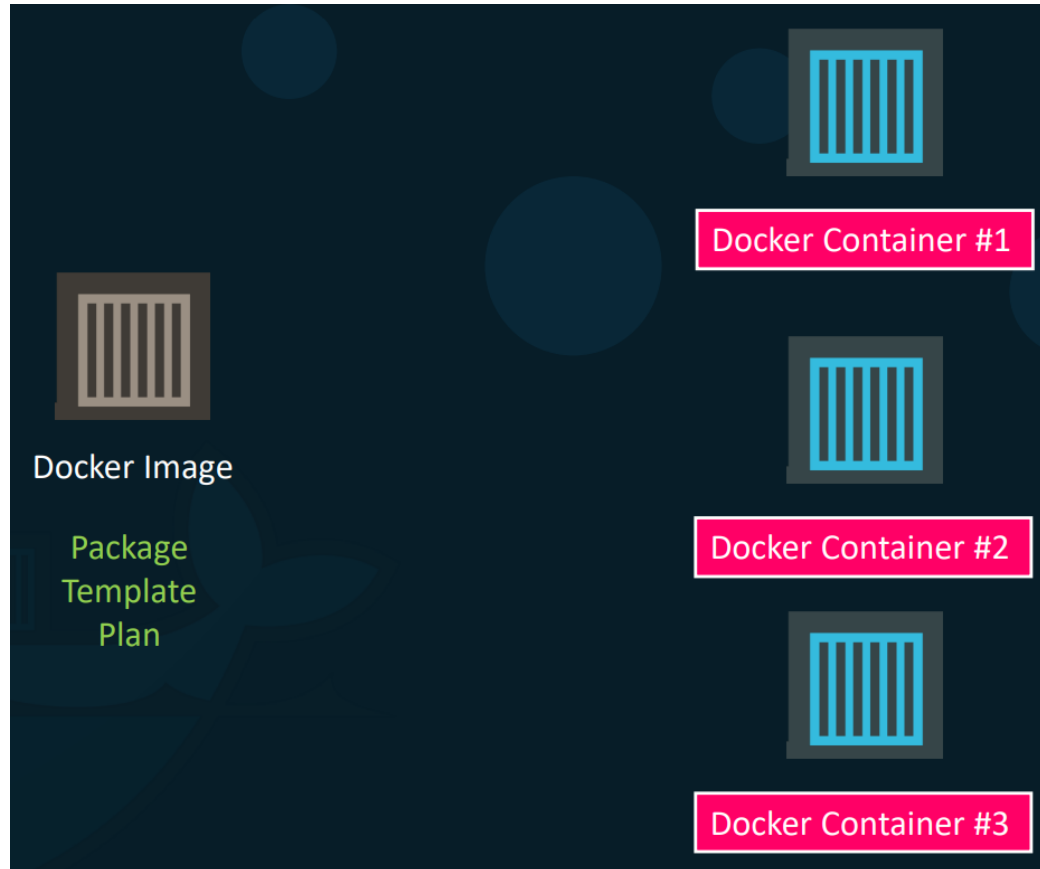
Check this URL: <https://docs.docker.com/engine/install/>

Check for the right OS and version on the left side panel and then follow the instructions.

```
$ sudo docker version
```

```
$ sudo run docker/whalesay cowsay Hello there!
```

Container vs Image



Basic Docker Commands

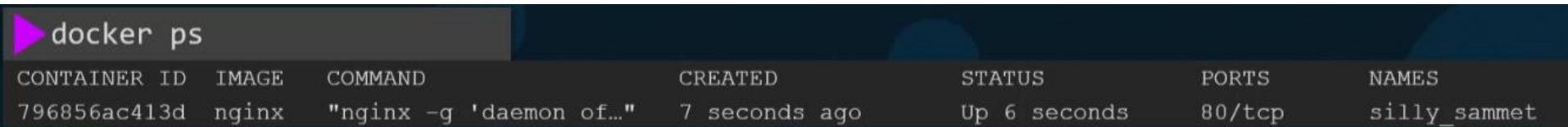
Run – start/run a container from an image

```
$ docker run nginx
```

Note: If the container image already exists on the local system, Docker runs it immediately. Otherwise, Docker pulls the image from Docker Hub or another container registry (such as Google Artifact Registry) before running it.

Basic Docker Commands

```
$ docker ps # list running containers
```



CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
796856ac413d	nginx	"nginx -g 'daemon of..."	7 seconds ago	Up 6 seconds	80/tcp	silly_sammet

```
$ docker ps -a # list all containers
```

```
$ docker stop <container_id> # Stop the container.
```

```
$ docker stop <container_name> # Stop the container.
```

```
$ docker rm <container_name/id> # Removes the container from memory
```

```
$ docker images # List images
```

```
$ docker rmi <image_name> # Delete the image. (NB: Make sure all dependent containers are stopped and removed before deleting the image)
```

Basic Docker Commands

\$ `docker pull nginx` # To only pull (or download) the image

\$ `docker run ubuntu` # It will download and run the ubuntu container and will exit immediately

This happens because containers are not designed to host an entire operating system but rather to execute specific tasks or processes.

```
▶ docker run ubuntu
▶ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED

```
▶ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED
45aacca36850	ubuntu	"/bin/bash"	43 seconds ago

Basic Docker Commands

`$ docker run ubuntu sleep 5` # It will now start the container and run a command sleep and will go to sleep for 5 secs. Then the container will stop.

`$ docker exec <container_name> cat /etc/hosts` # execute a command inside this container

Run – attach and detach

`$ docker run docker run -p 8080:80 nginxdemos/hello` # This is the attach mode and nothing else can be done on the terminal

`$ docker run docker run -d -p 8080:80 nginxdemos/hello` # This is the detached mode and the control on the terminal will be given back. Docker container will run in the background mode. Now you can run other commands on the prompt.

`$ docker attach <container_id/name>` # attach the background running container to the prompt.

Note: When specifying the container ID, you can provide just the first few characters instead of the full ID.

Docker Run Command

`$ docker run -it centos bash` # Interactive terminal mode with bash prompt.

To view the output, try running `cat /etc/*release*` in interactive terminal mode.

Type `exit` to exit the interactive terminal mode.

Run – tag

\$ docker run redis # When no tag specified, uses “latest” as tag

```
docker run redis
```

```
Using default tag: latest
```

\$ docker run redis:7.4 # specified version is pulled

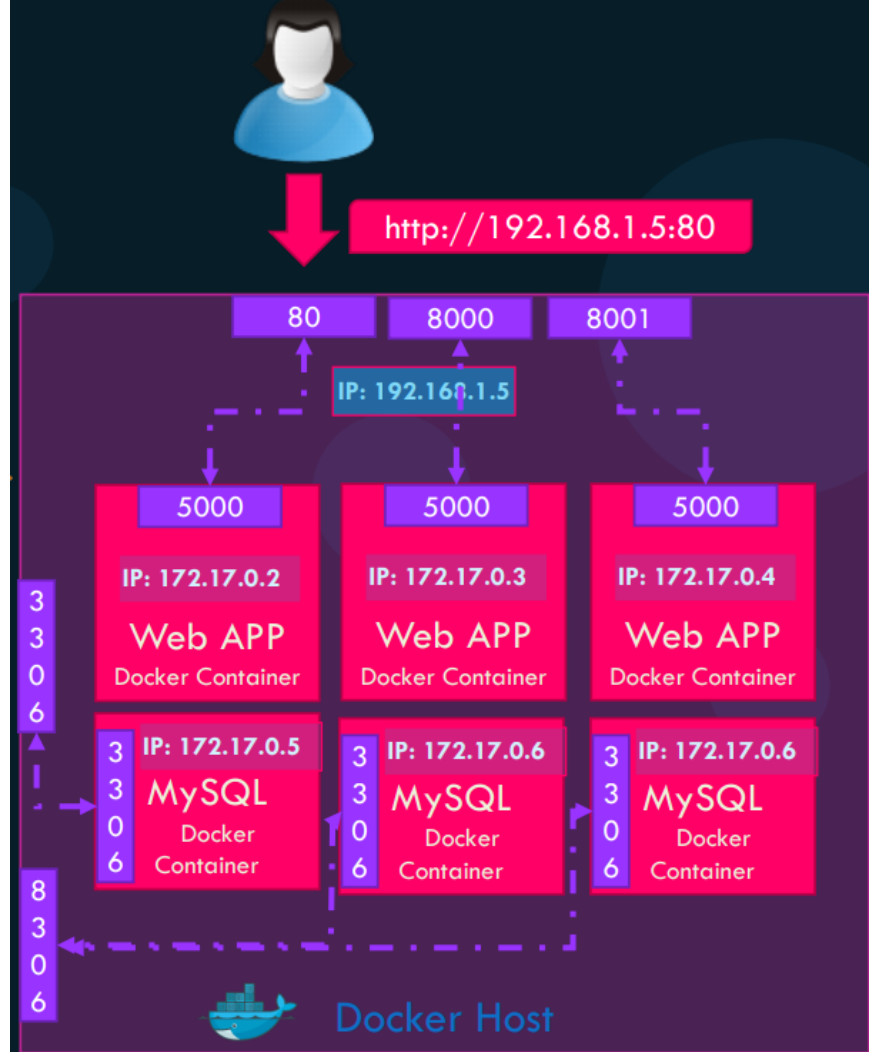
```
docker run redis:4.0
```

TAG

```
Unable to find image 'redis:4.0' locally
```

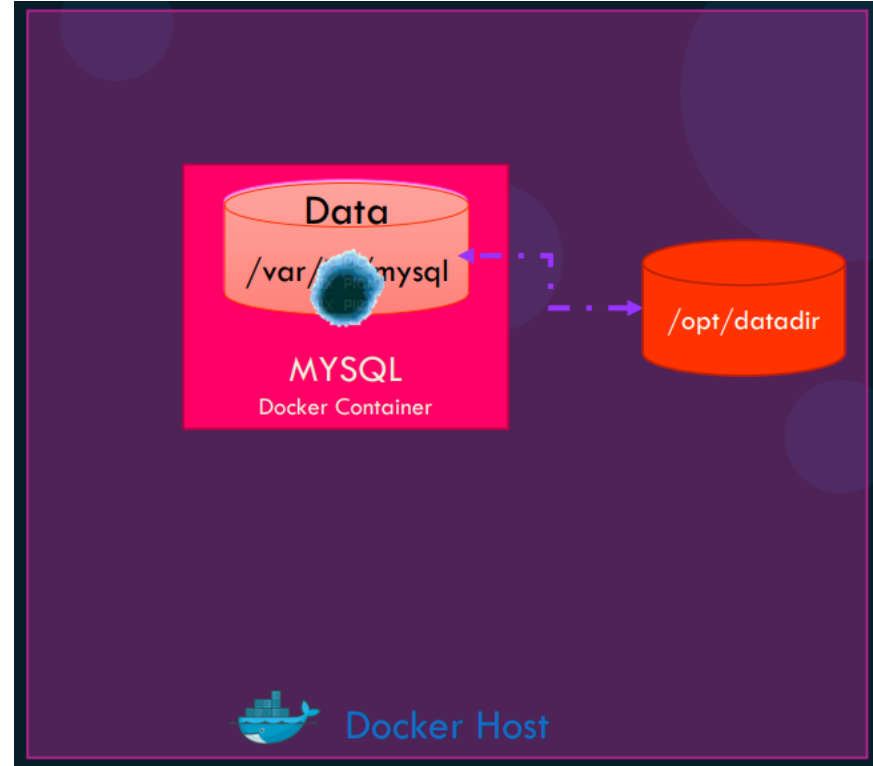
Run – PORT mapping

```
$ docker run docker run -p  
8080:80 nginxdemos/hello  
# Container's 80 port is mapped to  
port 8080 and now the host can  
access it via 8080 port.
```



Run – Volume mapping

```
$ docker run -v  
/opt/datadir:/var/lib/mysql  
mysql # Container's volume  
/var/lib/mysql is not mounted to  
/opt/datadir on the local (host) system. All  
data will be replicated/preserved.
```



Inspect Container

```
$ docker inspect <container_name> # Details about the container
```

Container Logs

```
$ docker logs <container_name> # Logs generated inside the container
```

ENV Variables in Docker

```
$ docker run -e APP_COLOR=blue simple-webapp-color
```

```
import os
from flask import Flask, render_template
```

```
app = Flask(__name__)
```

```
# Get the color from the environment variable APP_COLOR, default to
"red" if not set
color = os.environ.get("APP_COLOR", "red")
```

```
@app.route("/")
def main():
    print(color) # Log the color to the console
    return render_template('hello.html', color=color)
```

```
if __name__ == "__main__":
    app.run(host="0.0.0.0", port=8080)
```

```
$ export APP_COLOR=blue; python app.py
```

How to create my own image?

```
Dockerfile
FROM Ubuntu
```

```
RUN apt update
RUN apt install python
```

```
RUN pip install flask
RUN pip install flask-mysql
```

```
COPY . /opt/source-code
```

```
ENTRYPOINT FLASK_APP=/opt/source-code/app.py flask run
```

```
$ docker build Dockerfile -t <myusername>/my-custom-app
```

```
$ docker push <myusername>/my-custom-app
```

1. OS - Ubuntu

2. Update apt repo

3. Install dependencies using apt

4. Install Python dependencies using pip

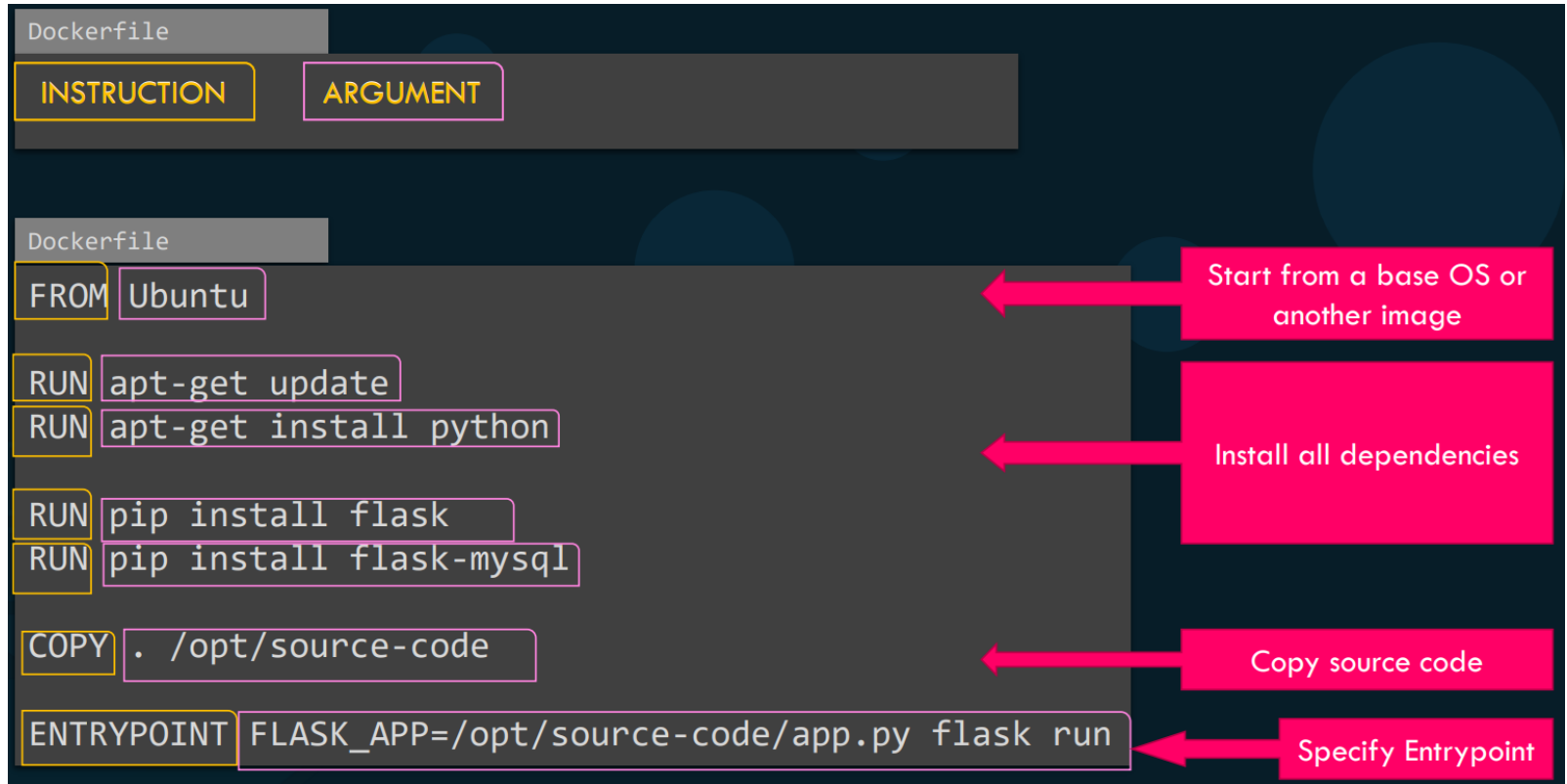
5. Copy source code to /opt folder

6. Run the web server using “flask” command



Docker
Registry

Dockerfile



Practice Coding – A LOT





Any Questions
