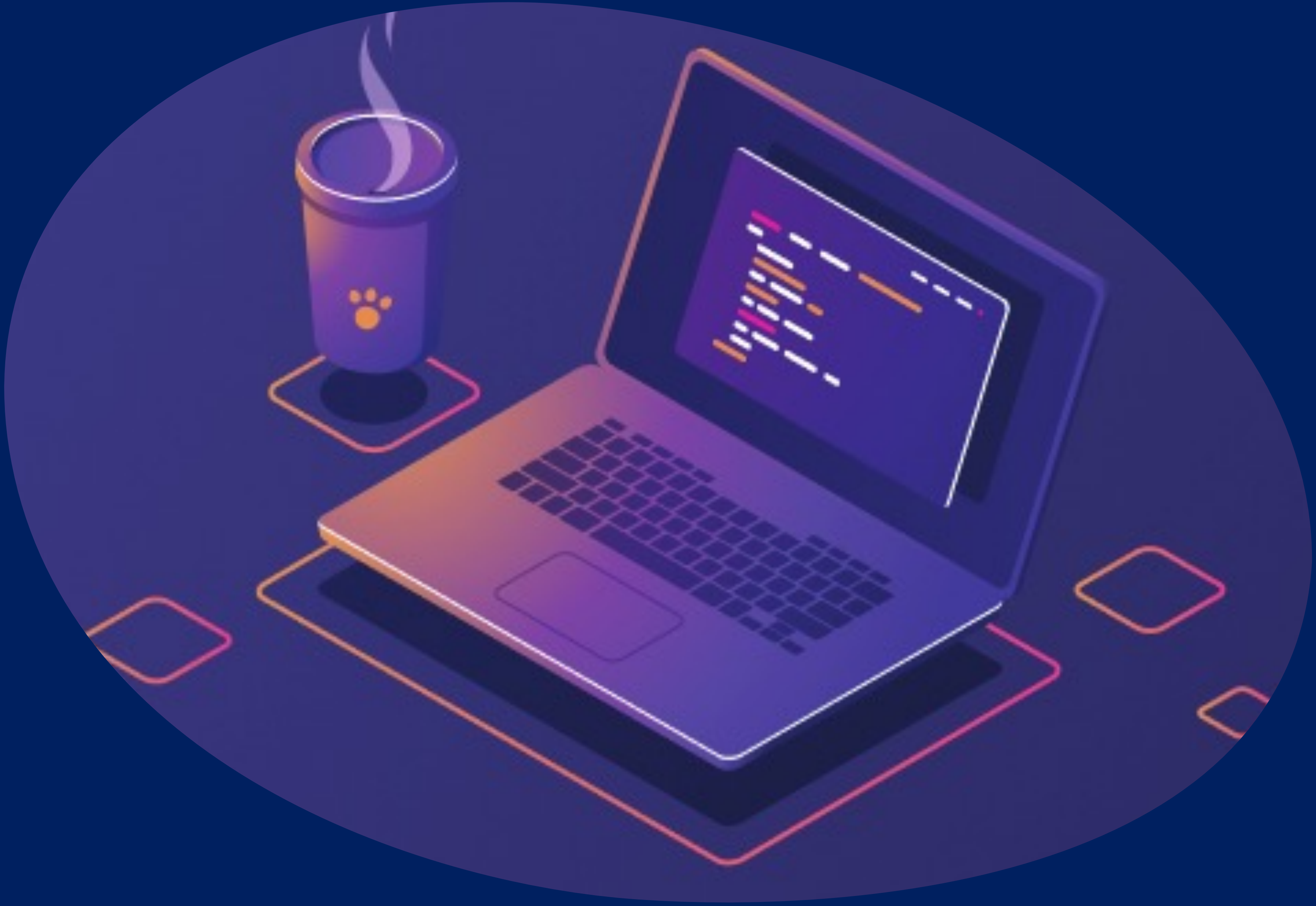
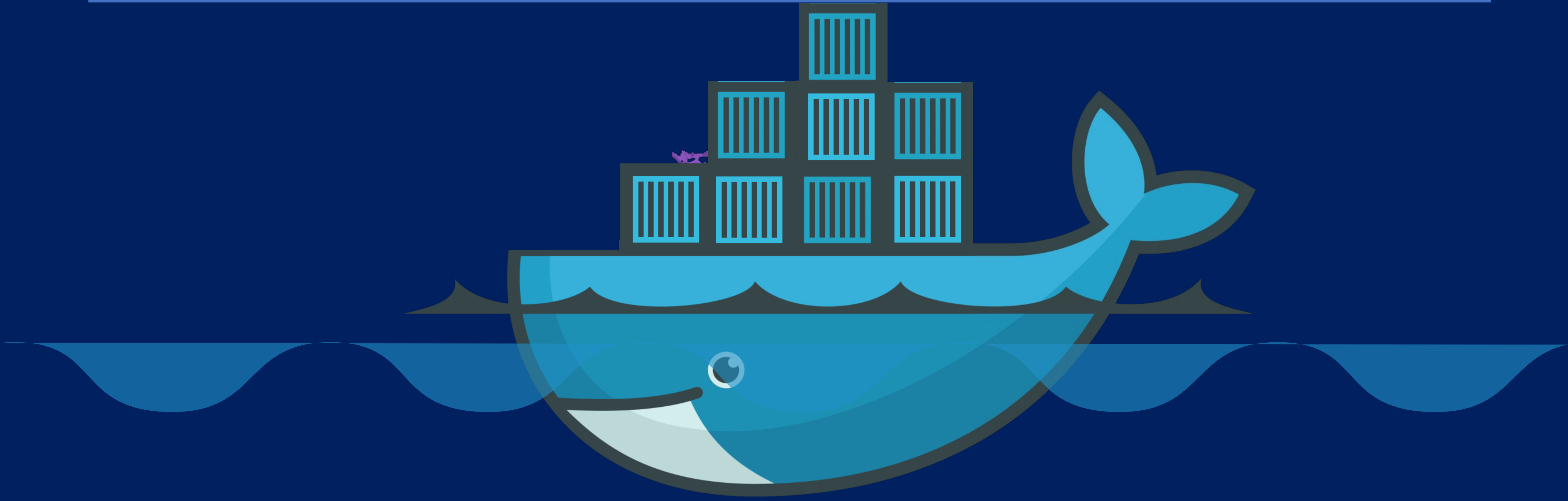
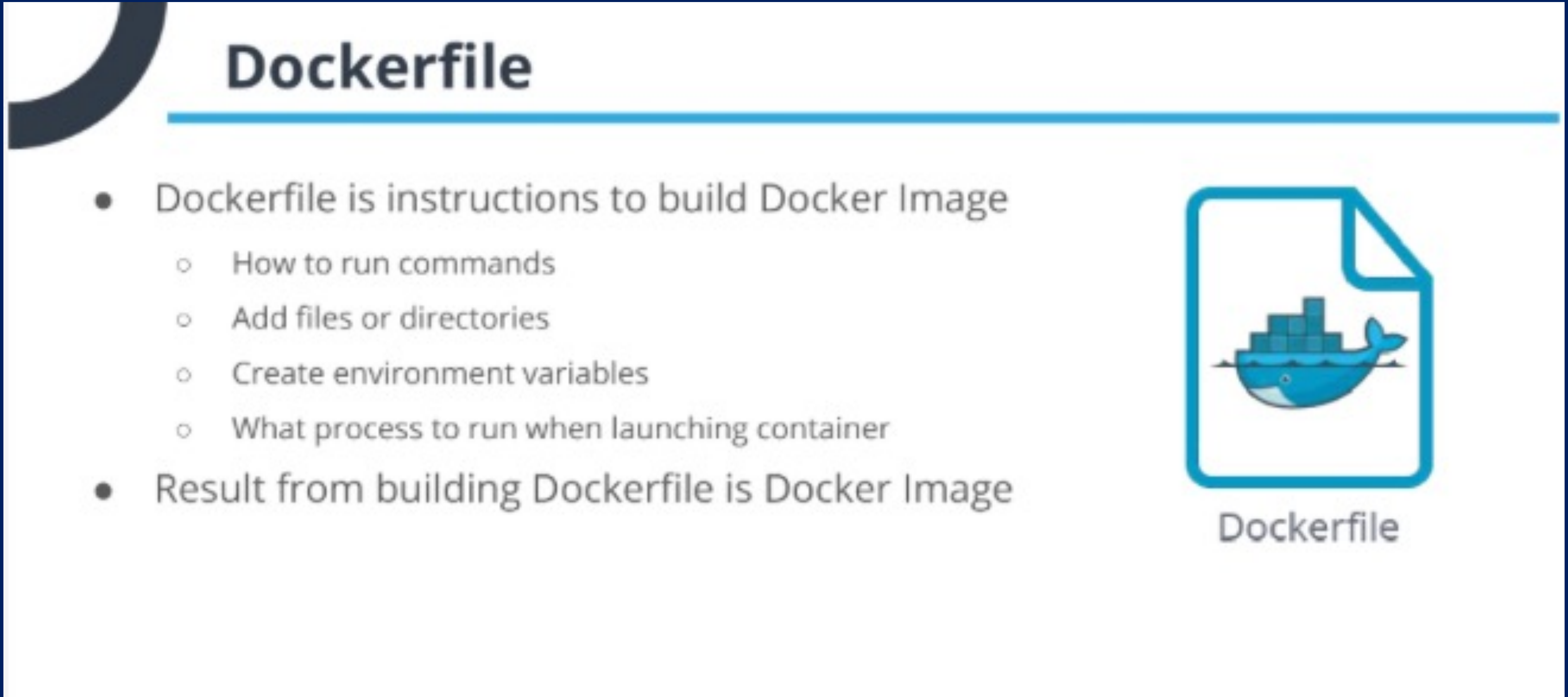


Week 10 : SOFTWARE DEVELOPMENT TOOLS AND ENVIRONMENTS

Dockerfile and Docker Image

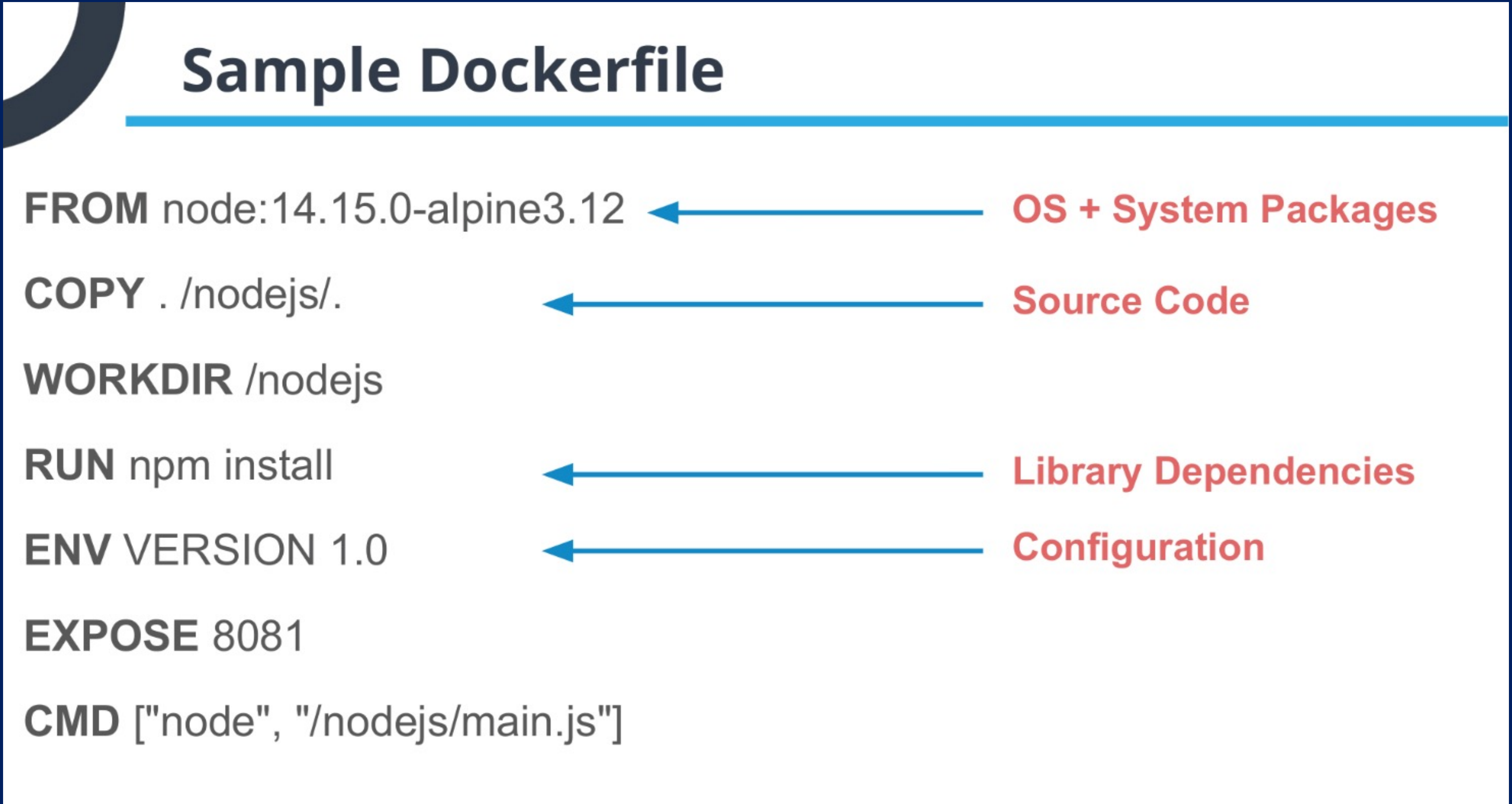




Dockerfile

- Dockerfile is instructions to build Docker Image
  - How to run commands
  - Add files or directories
  - Create environment variables
  - What process to run when launching container
- Result from building Dockerfile is Docker Image

Dockerfile



Sample Dockerfile

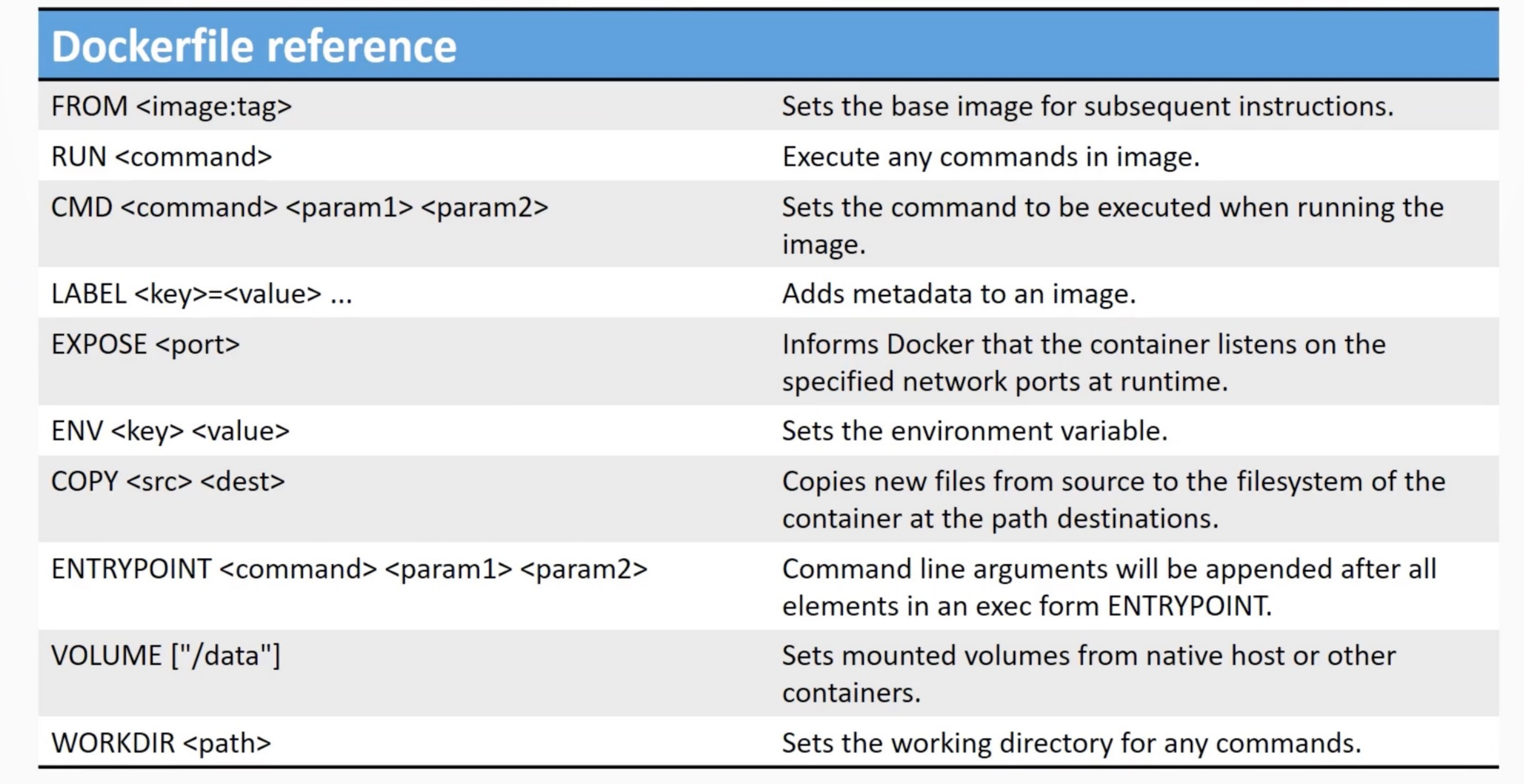
```
FROM node:14.15.0-alpine3.12
COPY . /nodejs/.
WORKDIR /nodejs
RUN npm install
ENV VERSION 1.0
EXPOSE 8081
CMD ["node", "/nodejs/main.js"]
```

OS + System Packages

Source Code

Library Dependencies

Configuration



Dockerfile reference

FROM <image:tag>	Sets the base image for subsequent instructions.
RUN <command>	Execute any commands in image.
CMD <command> <param1> <param2>	Sets the command to be executed when running the image.
LABEL <key>=<value> ...	Adds metadata to an image.
EXPOSE <port>	Informs Docker that the container listens on the specified network ports at runtime.
ENV <key> <value>	Sets the environment variable.
COPY <src> <dest>	Copies new files from source to the filesystem of the container at the path destinations.
ENTRYPOINT <command> <param1> <param2>	Command line arguments will be appended after all elements in an exec form ENTRYPOINT.
VOLUME ["/data"]	Sets mounted volumes from native host or other containers.
WORKDIR <path>	Sets the working directory for any commands.

How to create my own image?

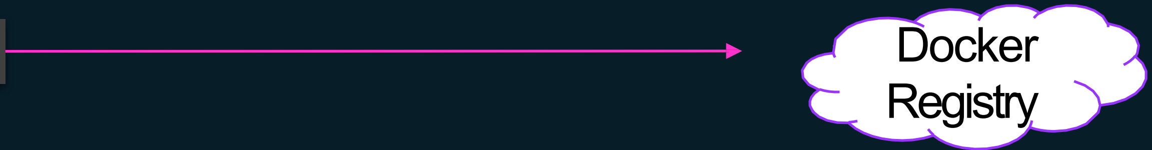
Dockerfile

```
FROM Ubuntu

RUN apt-get update
RUN apt-get 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
```

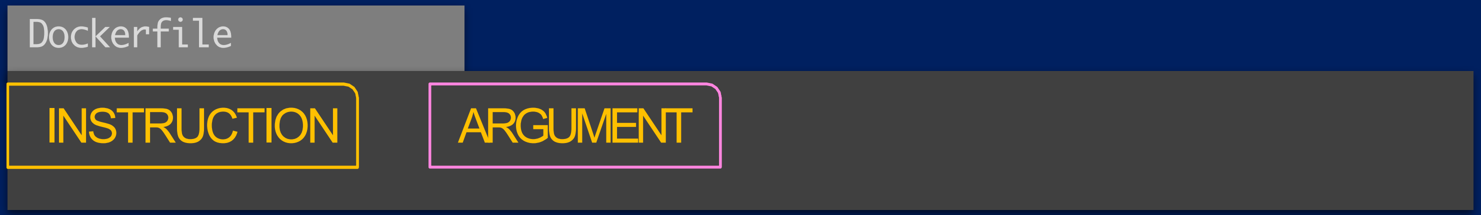
1. OS - Ubuntu
2. Update apt repo
3. Install dependencies using apt
4. Install Python dependencies using pip
5. Copy source code to /opt folder
6. Run the web server using “flask” command

```
docker build Dockerfile –t mmumshad/my-custom-app
```

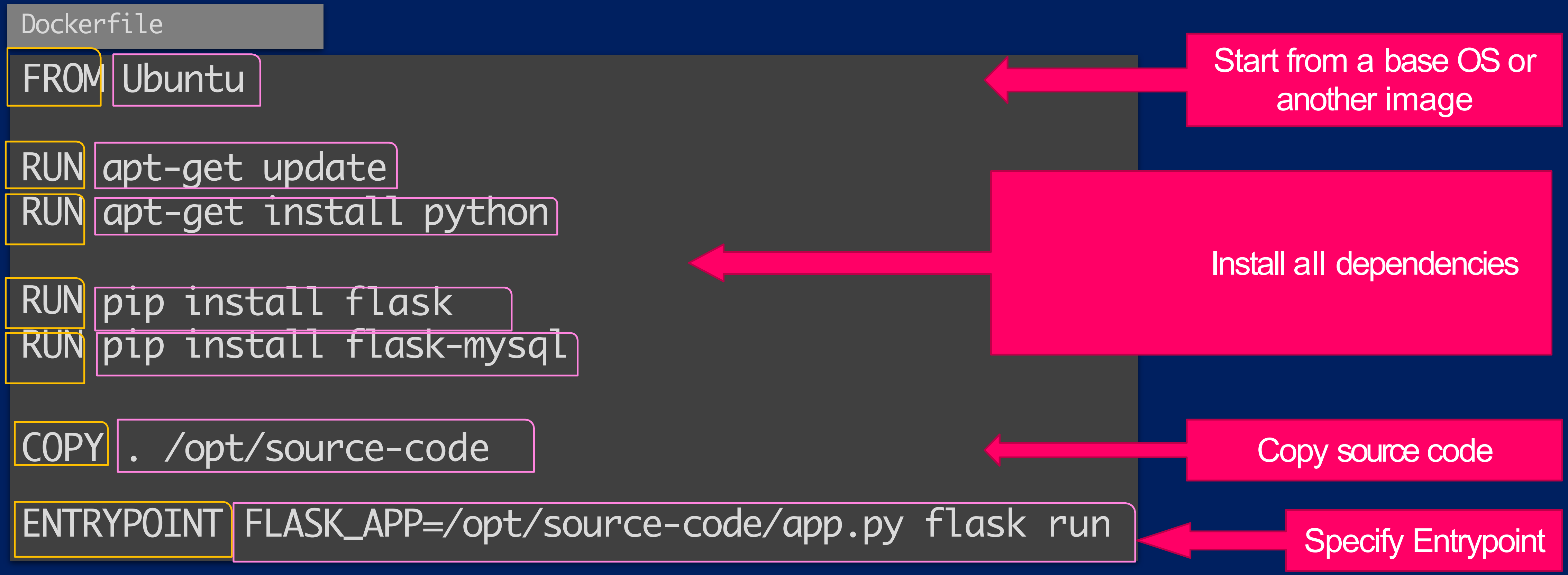
```
docker push mmumshad/my-custom-app
```



Dockerfile



```
Dockerfile
INSTRUCTION   ARGUMENT
```



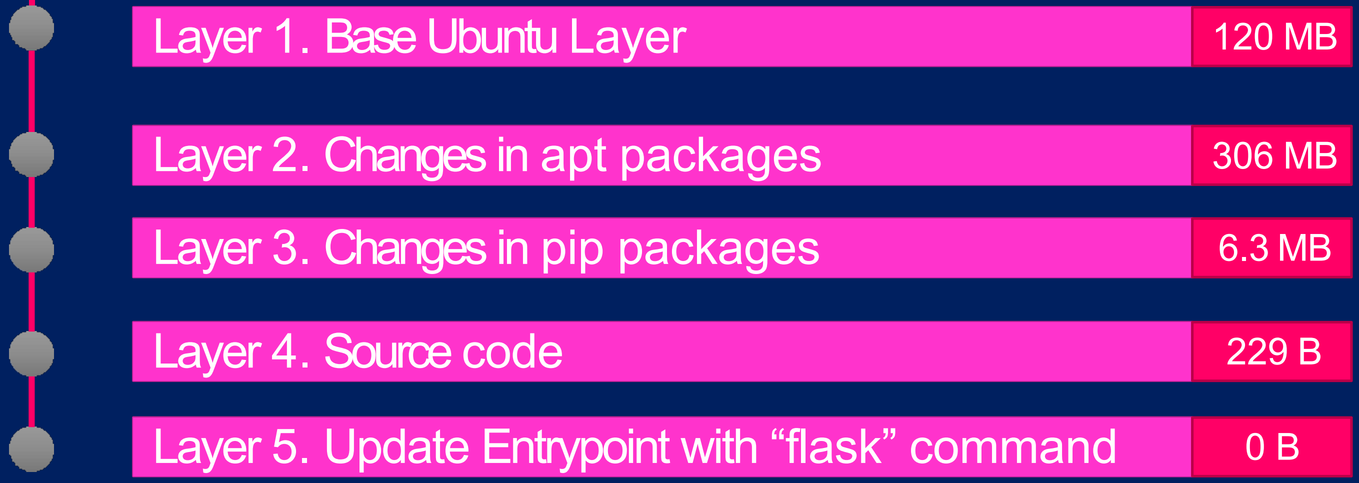
```
Dockerfile
FROM Ubuntu

RUN apt-get update
RUN apt-get 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
```

Layered architecture

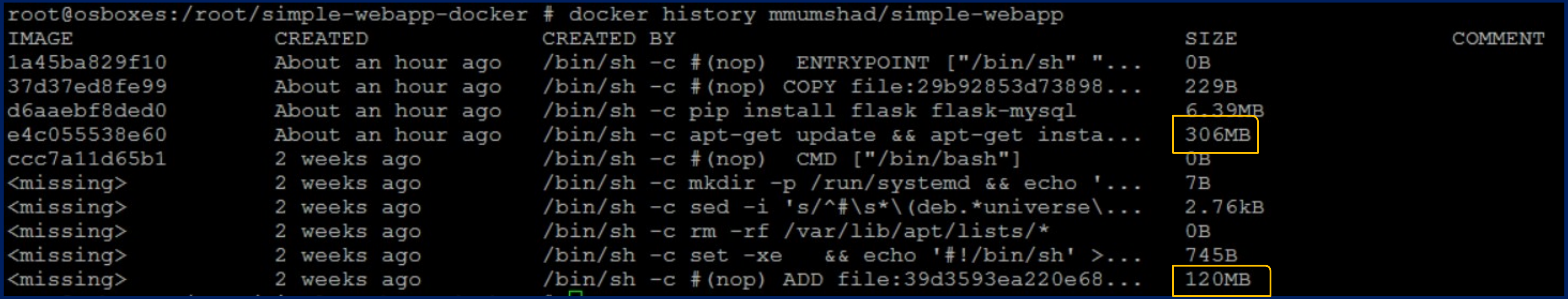
Dockerfile

```
FROM Ubuntu

RUN apt-get update && apt-get -y install
python
RUN pip install flask flask-
mysql
COPY . /opt/source-
code
ENTRYPOINT FLASK_APP=/opt/source-code/app.py flask run
```

```
docker build Dockerfile –t mmumshad/my-custom-app
```

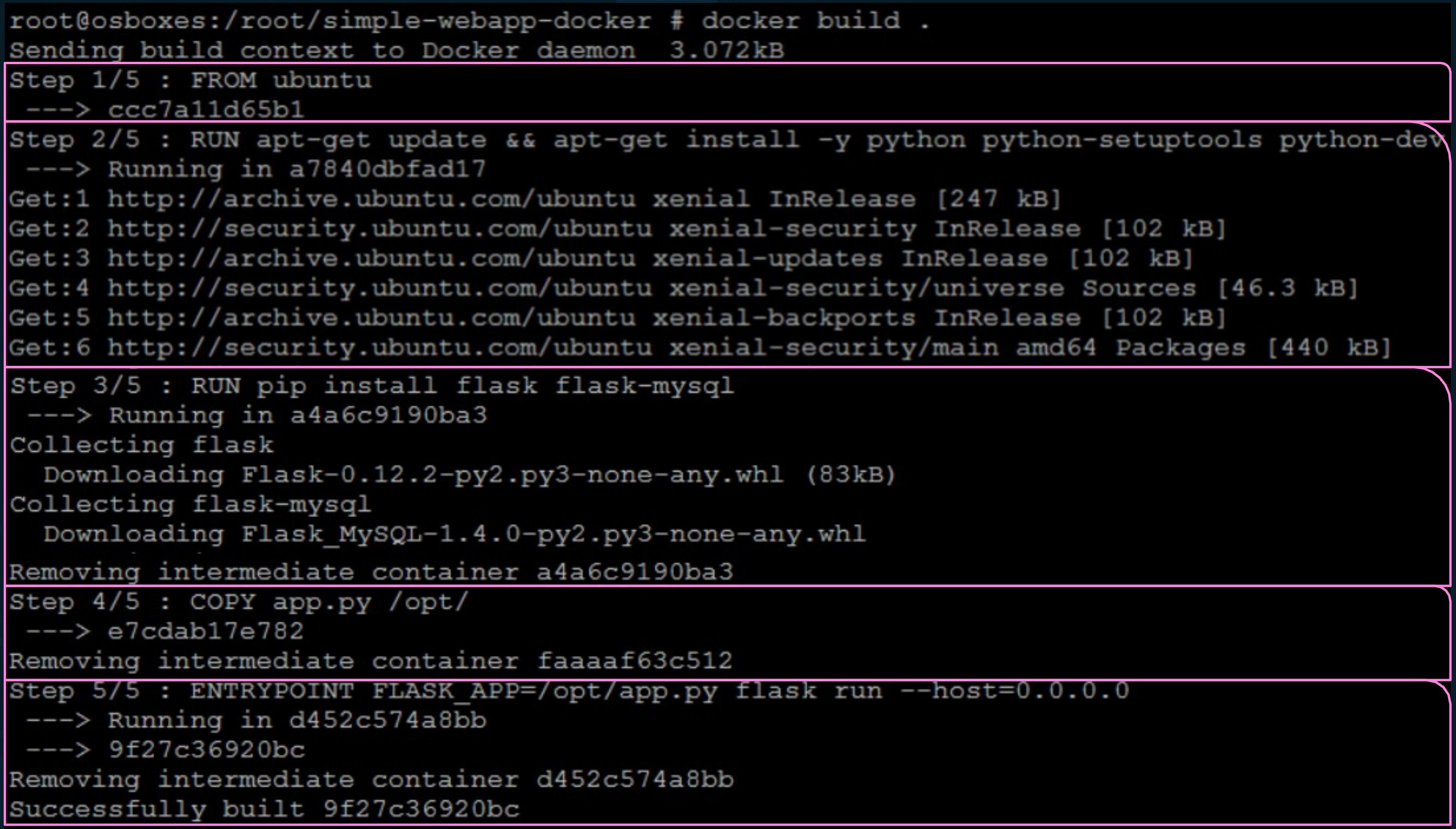


Layer 1. Base Ubuntu Layer	120 MB
Layer 2. Changes in apt packages	306 MB
Layer 3. Changes in pip packages	6.3 MB
Layer 4. Source code	229 B
Layer 5. Update Entrypoint with “flask” command	0 B



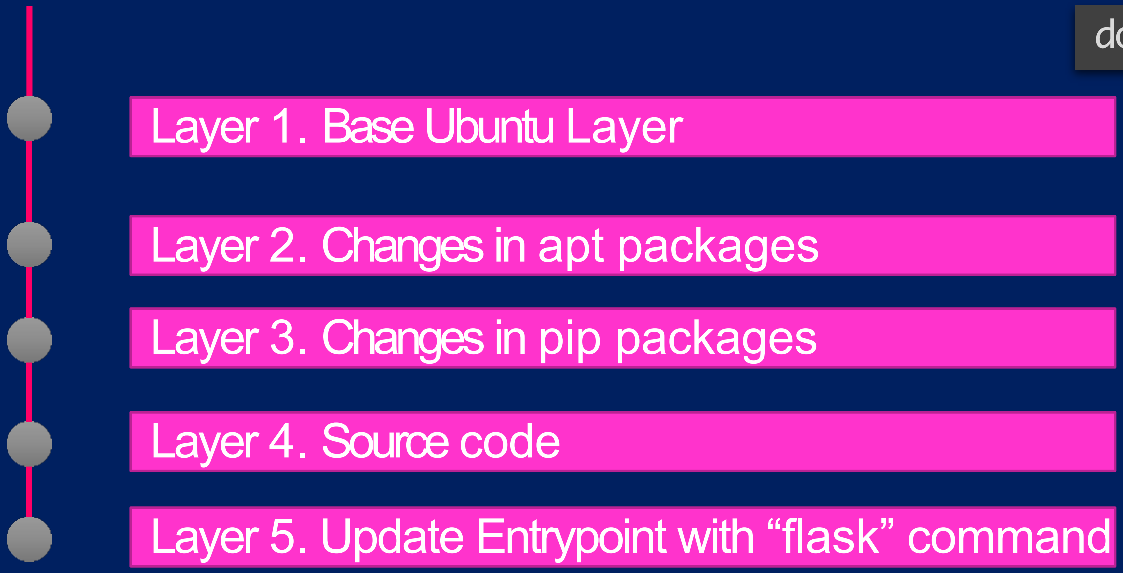
```
root@osboxes:/root/simple-webapp-docker # docker history mmumshad/simple-webapp
IMAGE               CREATED             CREATED BY                                      SIZE                COMMENT
1a45ba829f10        About an hour ago   /bin/sh -c #(nop)  ENTRYPOINT ["/bin/sh" "...   0B
37d37ed8fe99        About an hour ago   /bin/sh -c #(nop) COPY file:29b92853d73898...   229B
d6aaebf8ded0        About an hour ago   /bin/sh -c pip install flask flask-mysql        6.39MB
e4c055538e60        About an hour ago   /bin/sh -c apt-get update && apt-get insta...   306MB
ccc7a11d65b1        2 weeks ago         /bin/sh -c #(nop)  CMD ["/bin/bash"]            0B
<missing>           2 weeks ago         /bin/sh -c mkdir -p /run/systemd && echo '...   7B
<missing>           2 weeks ago         /bin/sh -c sed -i 's/^#\s*\(deb.*universe\...   2.76kB
<missing>           2 weeks ago         /bin/sh -c rm -rf /var/lib/apt/lists/*          0B
<missing>           2 weeks ago         /bin/sh -c set -xe   && echo '#!/bin/sh' >...   745B
<missing>           2 weeks ago         /bin/sh -c #(nop) ADD file:39d3593ea220e68...   120MB
```

Docker build output



```
root@osboxes:/root/simple-webapp-docker # docker build .
Sending build context to Docker daemon  3.072kB
Step 1/5 : FROM ubuntu
 ---> ccc7a11d65b1
Step 2/5 : RUN apt-get update && apt-get install -y python python-setuptools python-dev
 ---> Running in a7840dbfad17
Get:1 http://archive.ubuntu.com/ubuntu xenial InRelease [247 kB]
Get:2 http://security.ubuntu.com/ubuntu xenial-security InRelease [102 kB]
Get:3 http://archive.ubuntu.com/ubuntu xenial-updates InRelease [102 kB]
Get:4 http://security.ubuntu.com/ubuntu xenial-security/universe Sources [46.3 kB]
Get:5 http://archive.ubuntu.com/ubuntu xenial-backports InRelease [102 kB]
Get:6 http://security.ubuntu.com/ubuntu xenial-security/main amd64 Packages [440 kB]
Step 3/5 : RUN pip install flask flask-mysql
 ---> Running in a4a6c9190ba3
Collecting flask
  Downloading Flask-0.12.2-py2.py3-none-any.whl (83kB)
Collecting flask-mysql
  Downloading Flask_MySQL-1.4.0-py2.py3-none-any.whl
Removing intermediate container a4a6c9190ba3
Step 4/5 : COPY app.py /opt/
 ---> e7cdab17e782
Removing intermediate container faaaaf63c512
Step 5/5 : ENTRYPOINT FLASK_APP=/opt/app.py flask run --host=0.0.0.0
 ---> Running in d452c574a8bb
 ---> 9f27c36920bc
Removing intermediate container d452c574a8bb
Successfully built 9f27c36920bc
```

failure



Layer 1. Base Ubuntu Layer

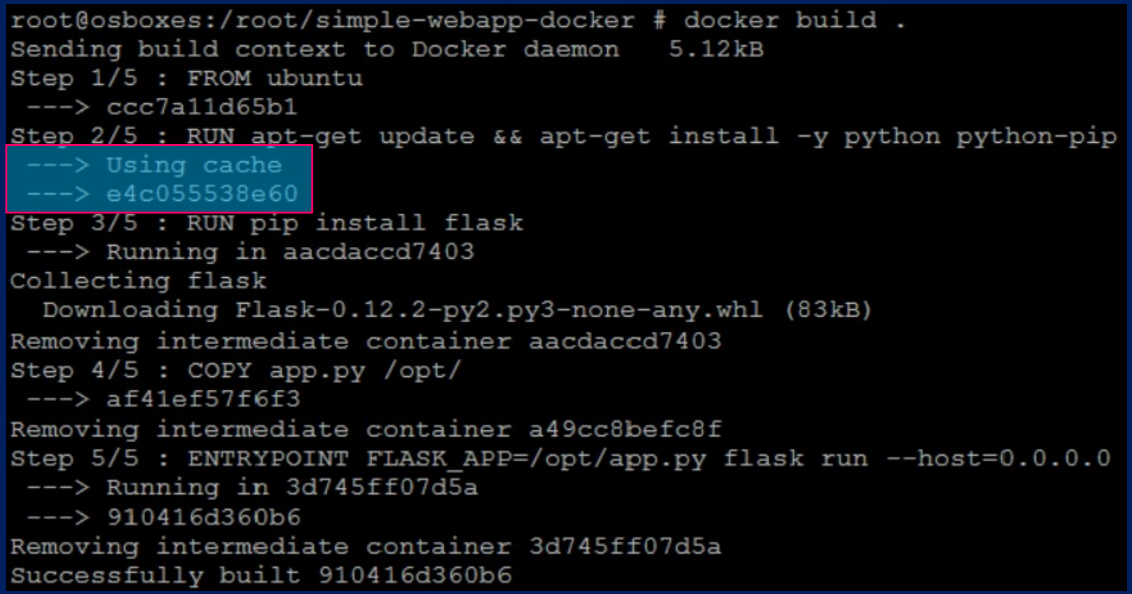
Layer 2. Changes in apt packages

Layer 3. Changes in pip packages

Layer 4. Source code

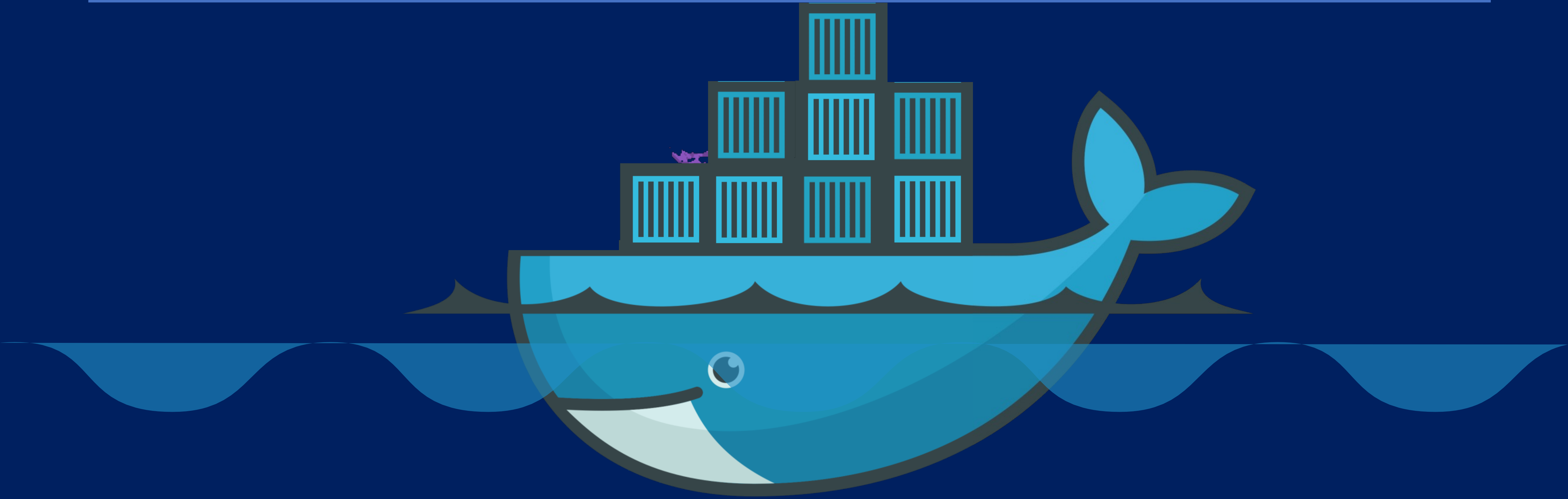
Layer 5. Update Entrypoint with “flask” command

```
docker build Dockerfile –t mmumshad/my-custom-app
```



```
root@osboxes:/root/simple-webapp-docker # docker build .
Sending build context to Docker daemon   5.12kB
Step 1/5 : FROM ubuntu
 ---> ccc7a11d65b1
Step 2/5 : RUN apt-get update && apt-get install -y python python-pip
 ---> Using cache
 ---> e4c055538e60
Step 3/5 : RUN pip install flask
 ---> Running in aacdaccd7403
Collecting flask
  Downloading Flask-0.12.2-py2.py3-none-any.whl (83kB)
Removing intermediate container aacdaccd7403
Step 4/5 : COPY app.py /opt/
 ---> af41ef57f6f3
Removing intermediate container a49cc8befc8f
Step 5/5 : ENTRYPOINT FLASK_APP=/opt/app.py flask run --host=0.0.0.0
 ---> Running in 3d745ff07d5a
 ---> 910416d360b6
Removing intermediate container 3d745ff07d5a
Successfully built 910416d360b6
```

Docker registry



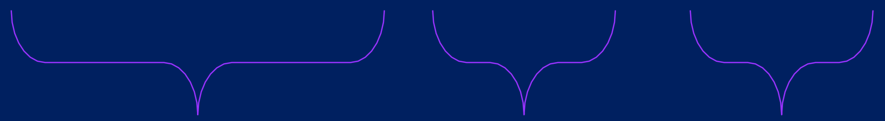
Image

```
docker run nginx
```

Image

docker.io
Docker Hub

```
image:  docker.io/nginx/nginx
```



Registry

User/
Account

Image/
Repository

```
gcr.io/ kubernetes-e2e-test-images/dnsutils
```

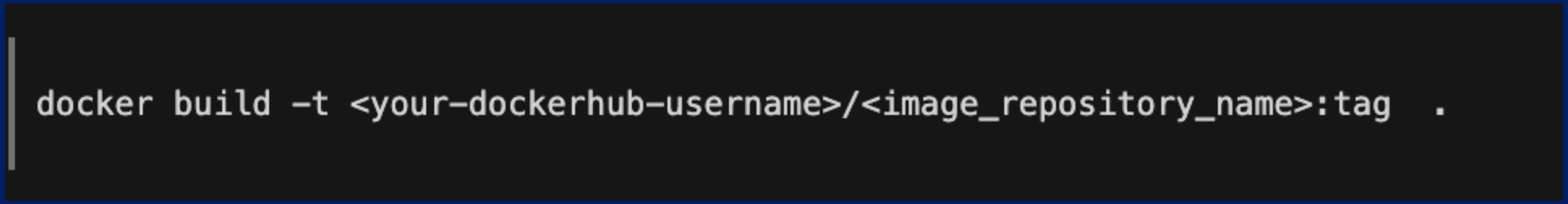
Private Registry

```
▶ docker login hub.docker.com
```

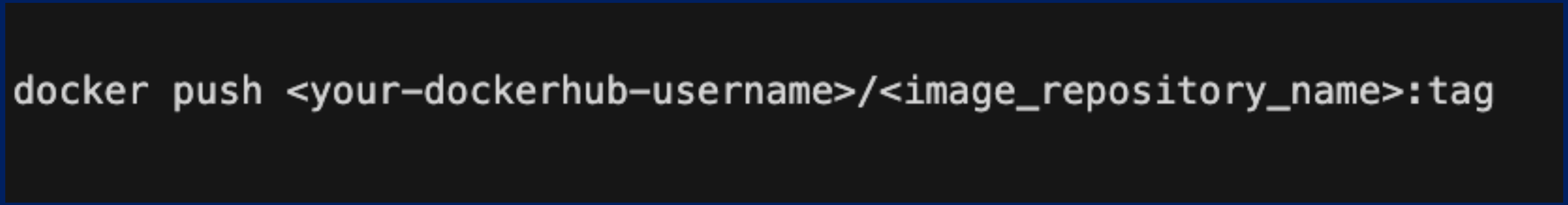
```
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to
https://hub.docker.com to create one.
Username: registry-user
Password:
WARNING! Your password will be stored unencrypted in /home/vagrant/.docker/config.json.

Login Succeeded
```

```
▶ docker run private-registry.io/apps/internal-app
```

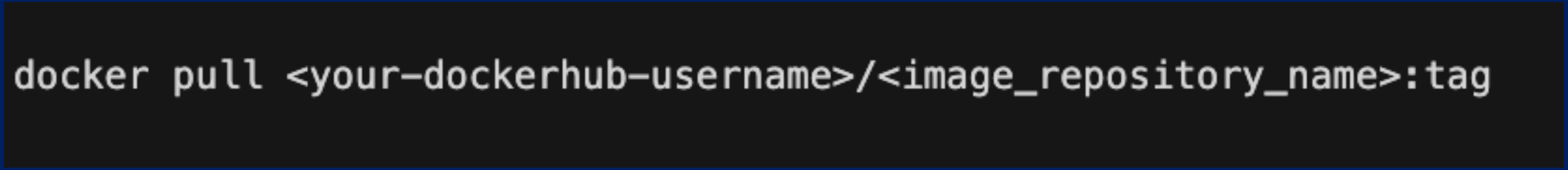
Deploy Private Registry



```
docker build -t <your-dockerhub-username>/<image_repository_name>:tag  .
```

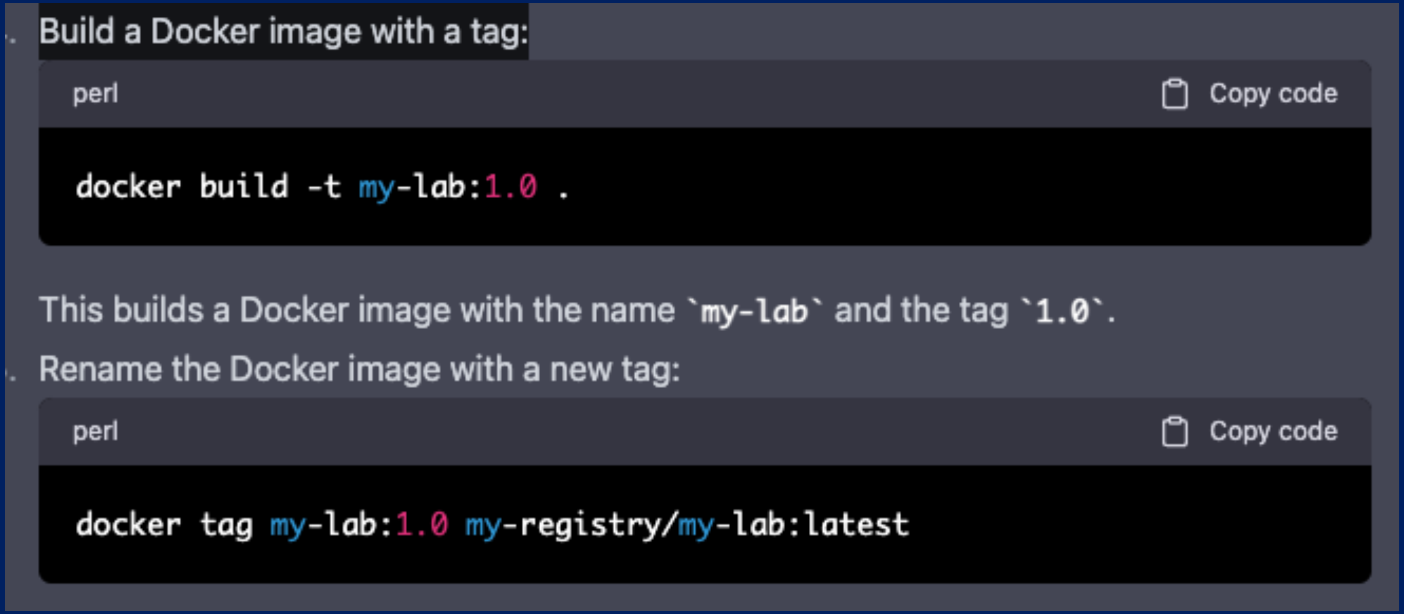


```
docker push <your-dockerhub-username>/<image_repository_name>:tag
```



```
docker pull <your-dockerhub-username>/<image_repository_name>:tag
```

How to rename image with Docker tag



Build a Docker image with a tag:

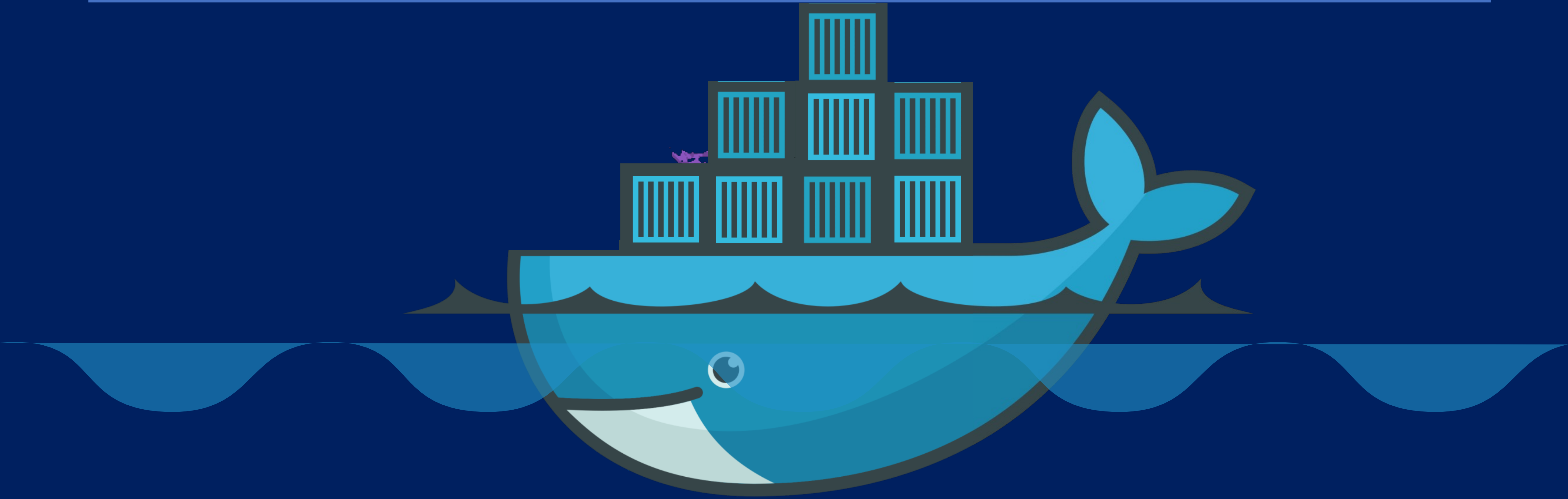
```perl
docker build -t my-lab:1.0 .
```

This builds a Docker image with the name `my-lab` and the tag `1.0`.

Rename the Docker image with a new tag:

```perl
docker tag my-lab:1.0 my-registry/my-lab:latest
```

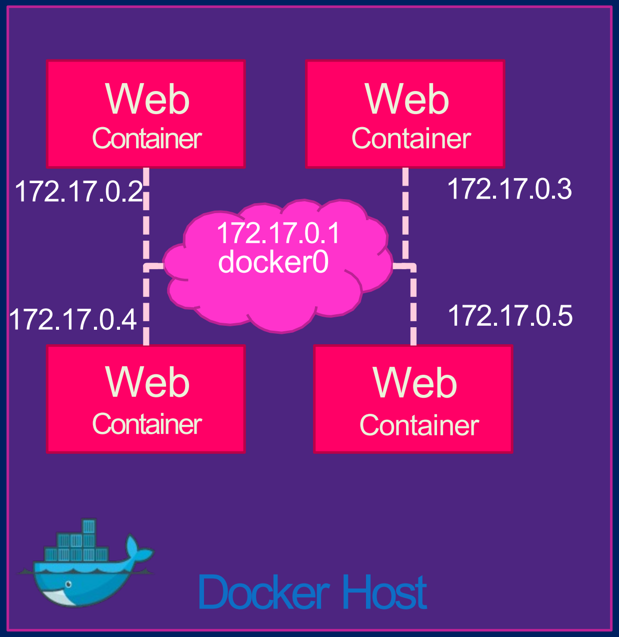
Docker Network



Default networks

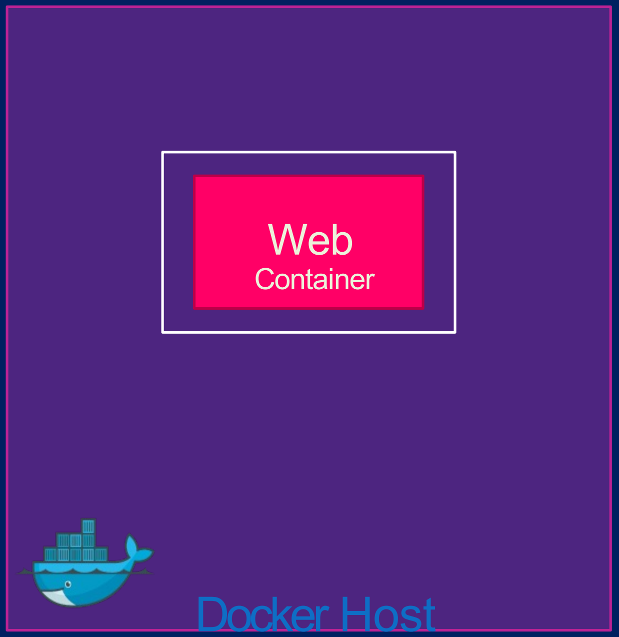
Bridge

```
docker run ubuntu
```



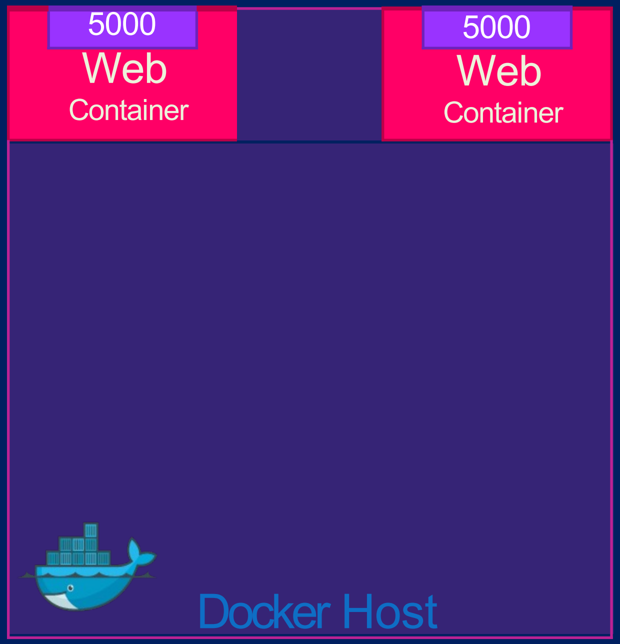
none

```
docker run Ubuntu --network=none
```

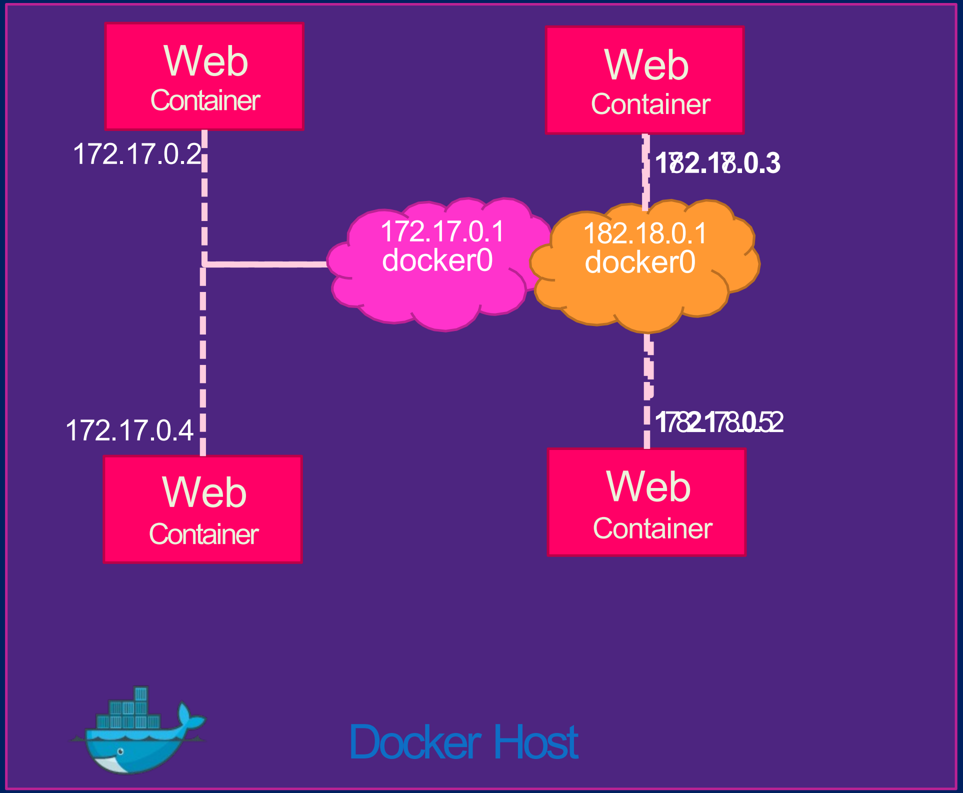


host

```
docker run Ubuntu --network=host
```

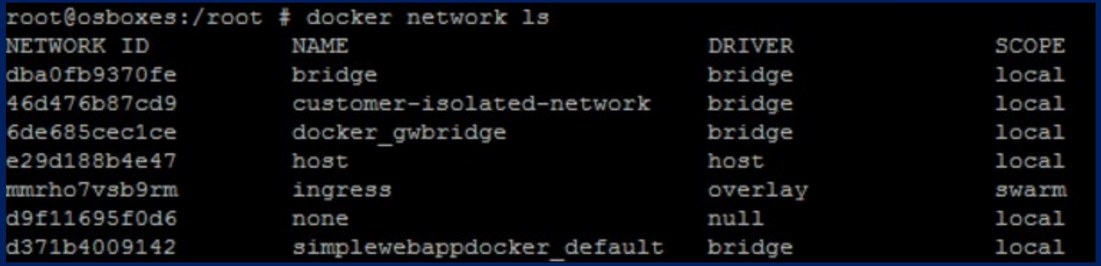


User-defined networks



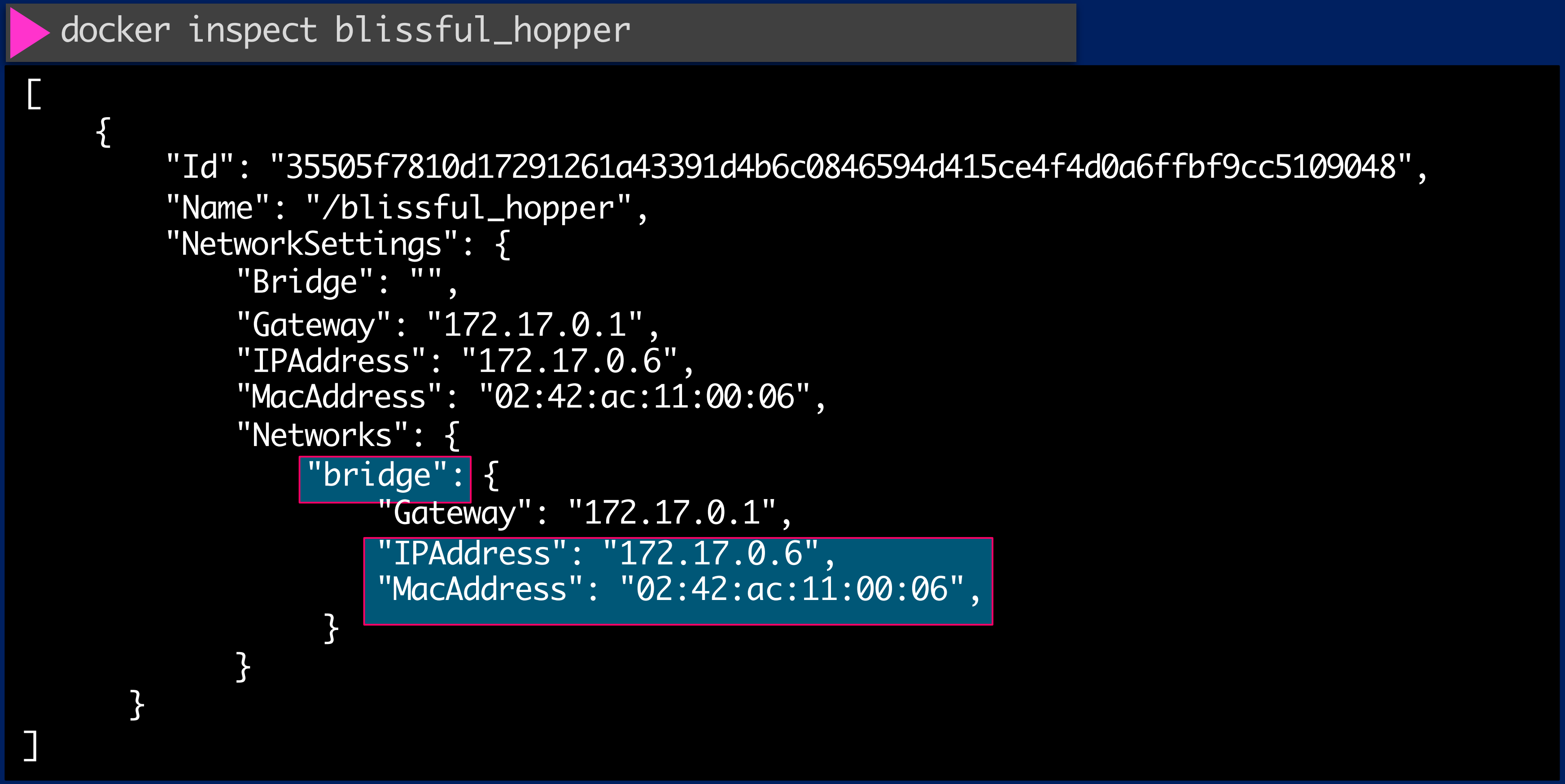
```
docker network create \
    --driver bridge \
    --subnet 182.18.0.0/16
    custom-isolated-network
```

```
docker network ls
```



```
root@osboxes:/root # docker network ls
NETWORK ID          NAME                          DRIVER              SCOPE
dba0fb9370fe        bridge                        bridge              local
46d476b87cd9        customer-isolated-network     bridge              local
6de685cec1ce        docker_gwbridge               bridge              local
e29d188b4e47        host                          host                local
mmrho7vsb9rm        ingress                       overlay             swarm
d9f11695f0d6        none                          null                local
d371b4009142        simplewebappdocker_default    bridge              local
```

Inspect Network

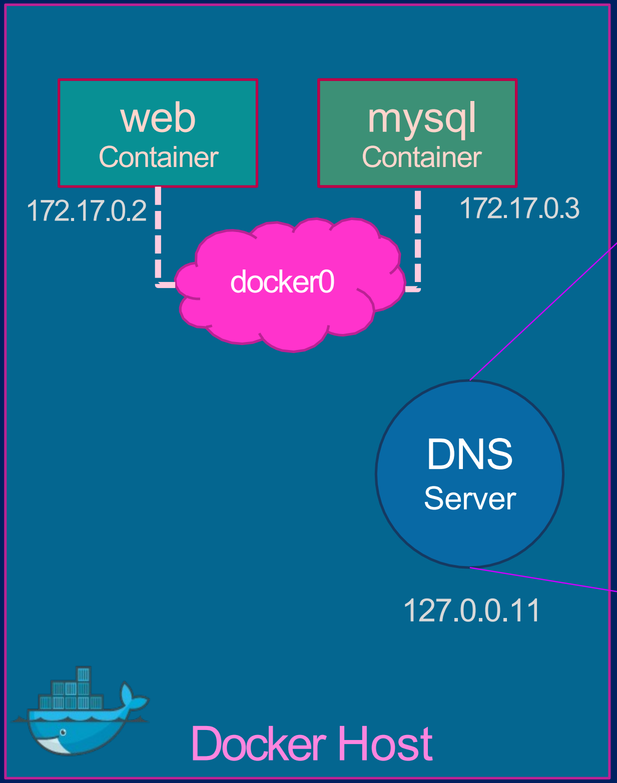


```
docker inspect blissful_hopper
```

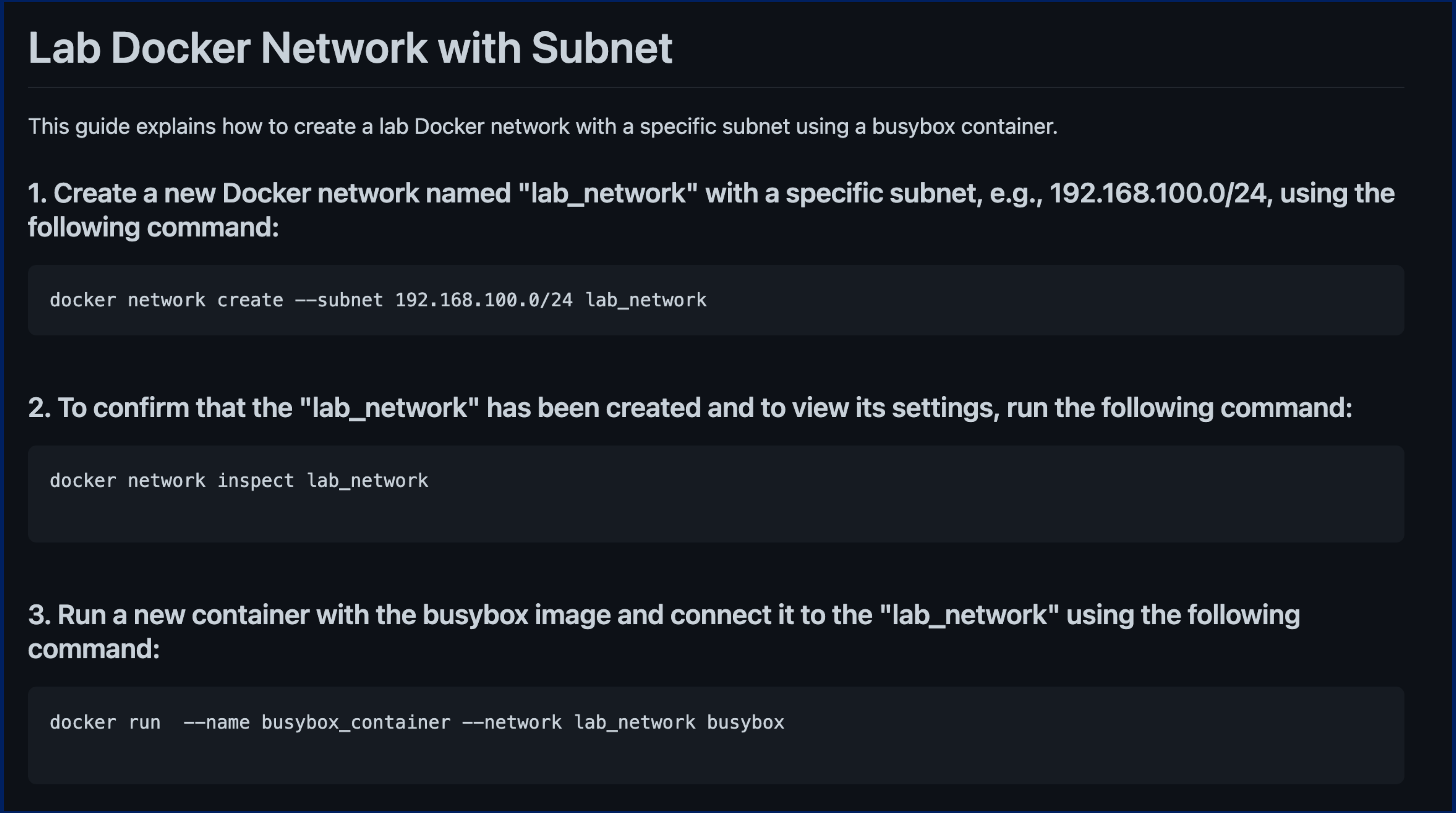
```
[
    {
        "Id": "35505f7810d17291261a43391d4b6c0846594d415ce4f4d0a6ffbf9cc5109048",
        "Name": "/blissful_hopper",
        "NetworkSettings": {
            "Bridge": "",
            "Gateway": "172.17.0.1",
            "IPAddress": "172.17.0.6",
            "MacAddress": "02:42:ac:11:00:06",
            "Networks": {
                "bridge": {
                    "Gateway": "172.17.0.1",
                    "IPAddress": "172.17.0.6",
                    "MacAddress": "02:42:ac:11:00:06",
                 }
            }
      }
]
```

Embedded DNS

```
mysql.connect( mysql      )
```



Host	**IP**
web	172.17.0.2
mysql	172.17.0.3



Lab Docker Network with Subnet

This guide explains how to create a lab Docker network with a specific subnet using a busybox container.

1. Create a new Docker network named "lab_network" with a specific subnet, e.g., 192.168.100.0/24, using the following command:

```
docker network create --subnet 192.168.100.0/24 lab_network
```

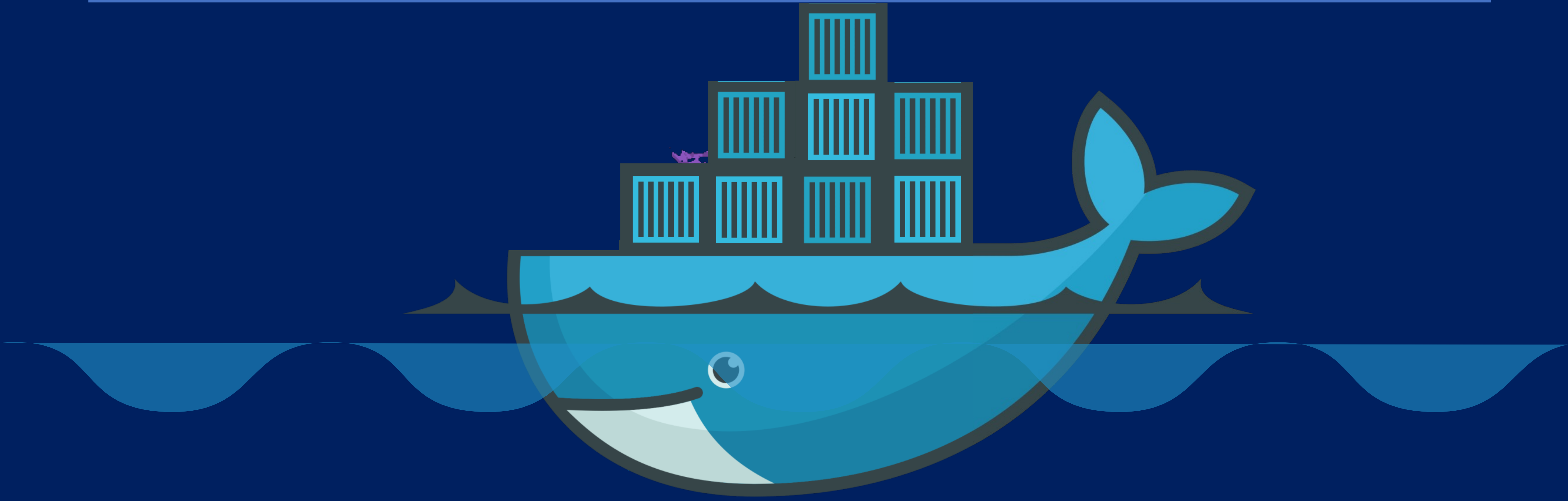
2. To confirm that the "lab_network" has been created and to view its settings, run the following command:

```
docker network inspect lab_network
```

3. Run a new container with the busybox image and connect it to the "lab_network" using the following command:

```
docker run  --name busybox_container --network lab_network busybox
```

Docker compose



Docker compose

Public Docker registry - dockerhub

```
docker run mmumshad/simple-webapp
```

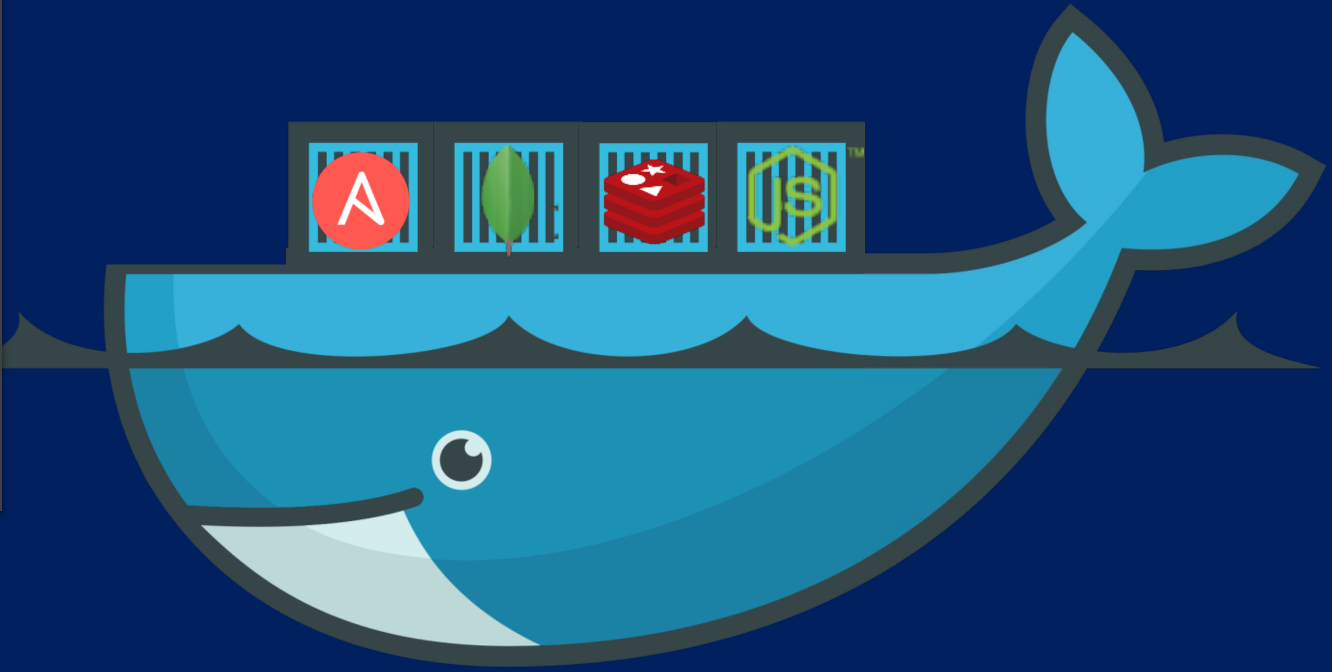
```
docker run mongodb
```

```
docker run redis:alpine
```

```
docker run ansible
```

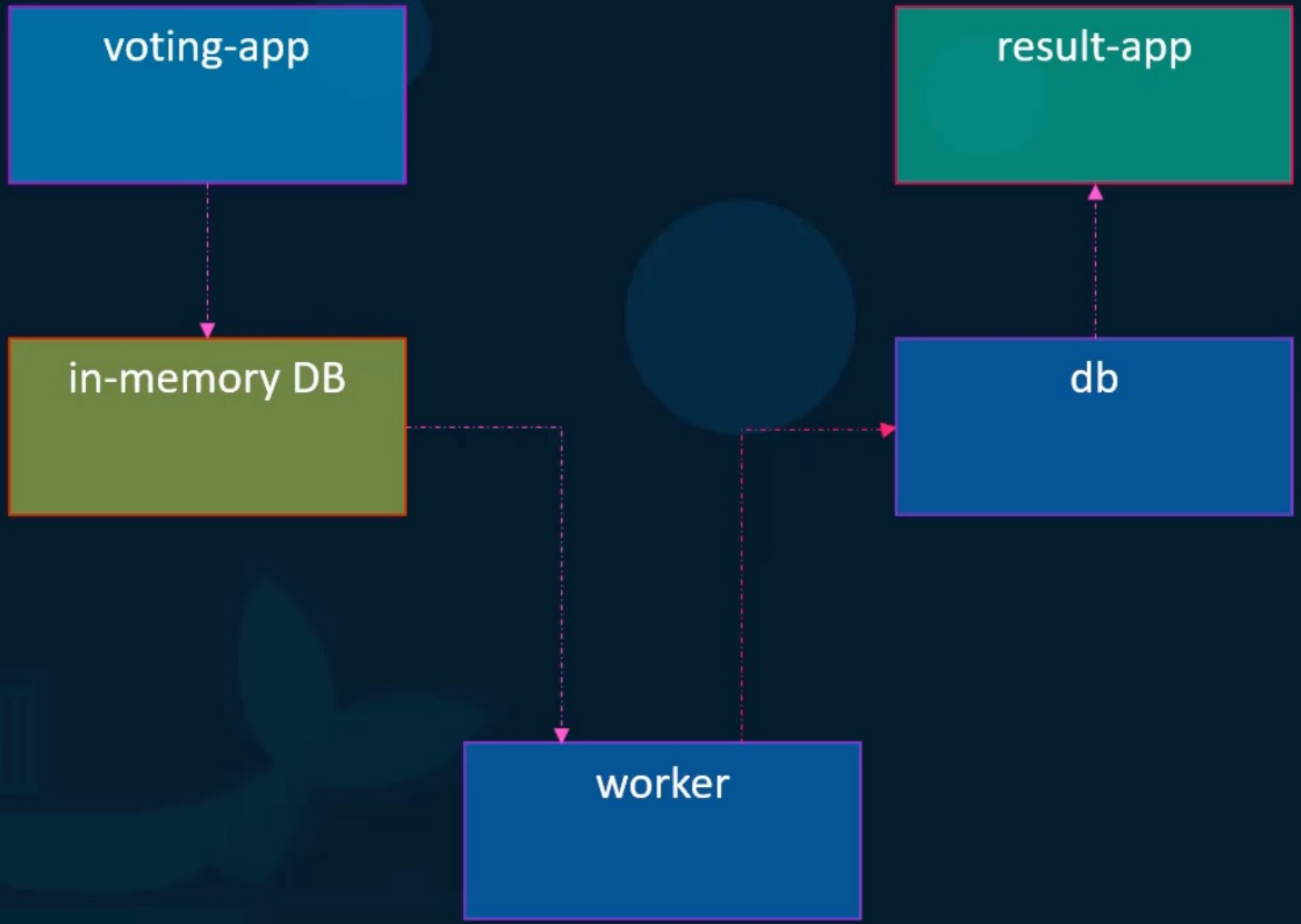
docker-compose.yml

```
services:
    web:
        image: “mmumshad/simple-webapp"
    database:
        image: “mongodb“
    messaging:
        image: "redis:alpine“
    orchestration:
        image:  “ansible“
```



```
docker-compose up
```

Sample application – voting application



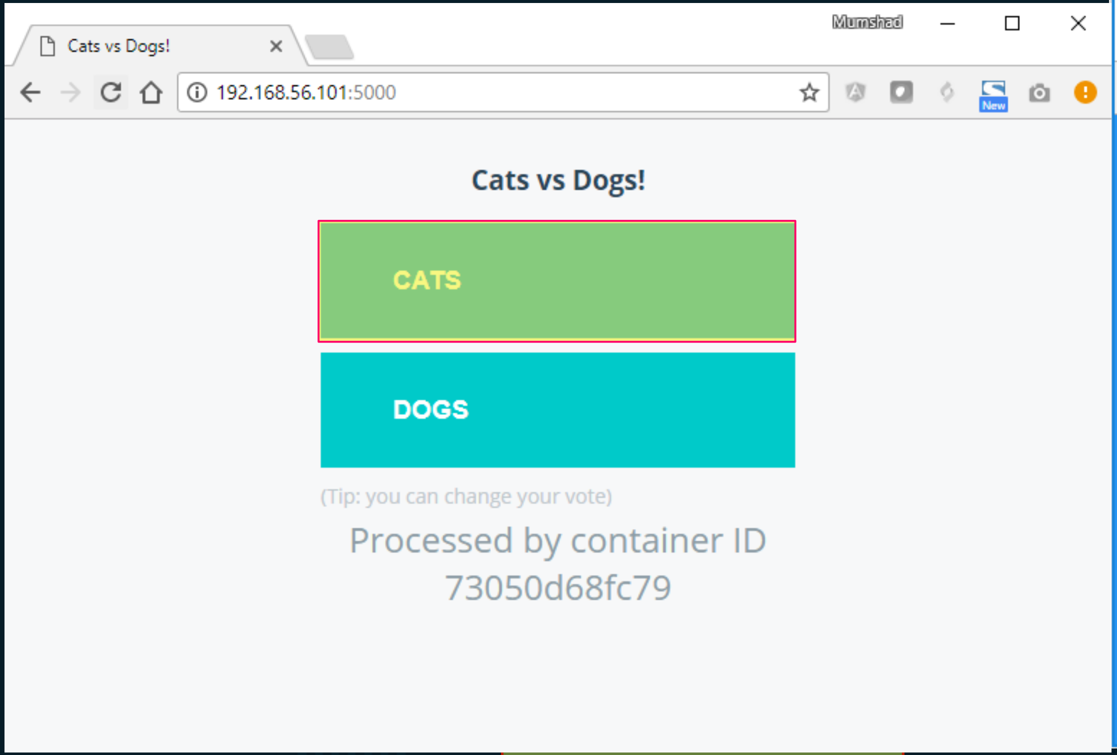
voting-app

in-memory DB

worker

db

result-app



Cats vs Dogs!

192.168.56.101:5000

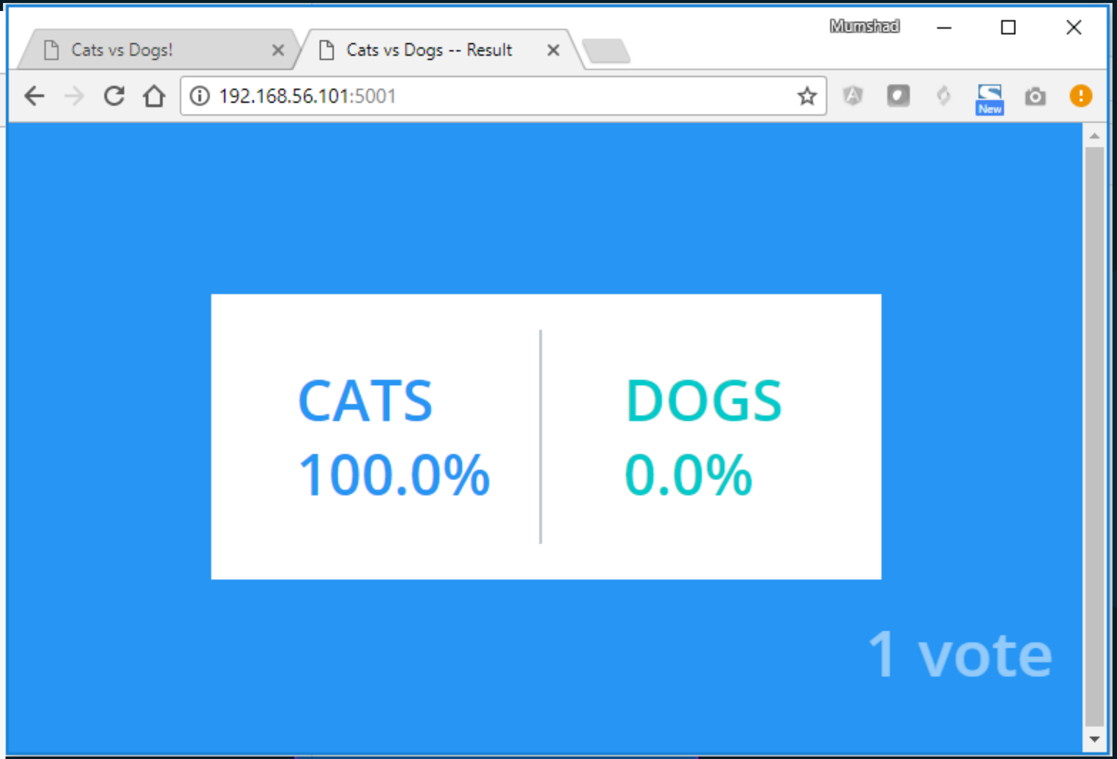
Cats vs Dogs!

CATS

DOGS

(Tip: you can change your vote)

Processed by container ID 73050d68fc79



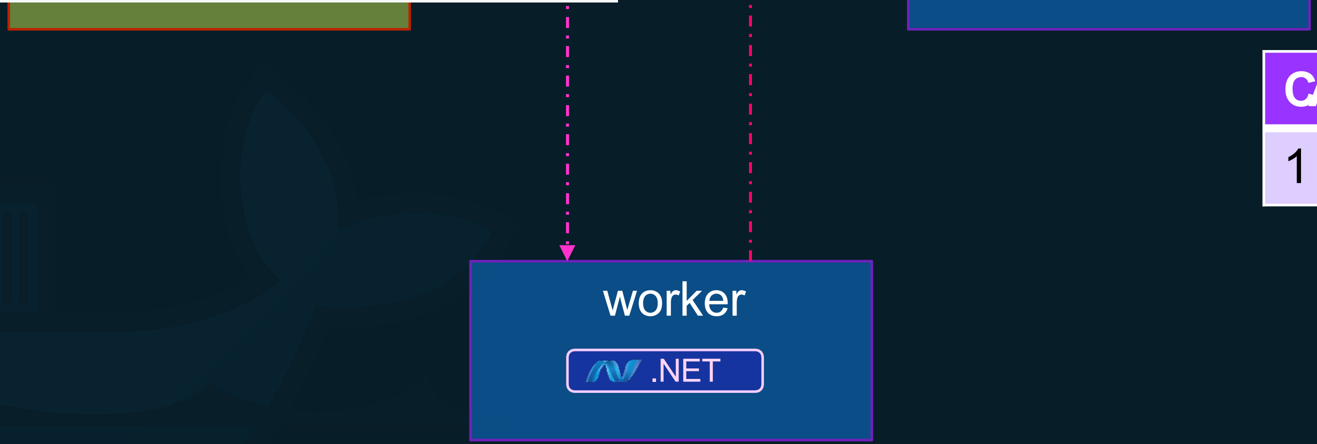
Cats vs Dogs! | Cats vs Dogs -- Result

192.168.56.101:5001

CATS 100.0%

DOGS 0.0%

1 vote



CATS	DOGS
1	0

docker run --links

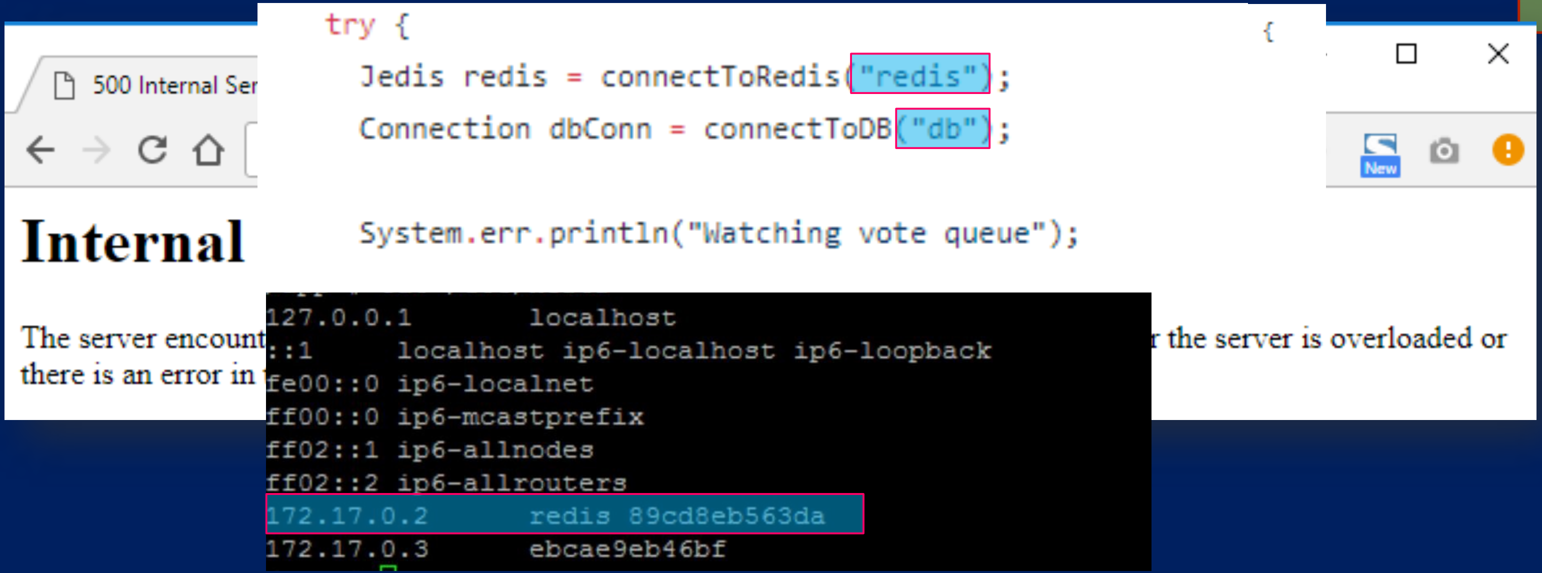
```
docker run -d --name=redis redis
```

```
docker run -d --name=db
```

```
docker run -d --name=vote –p 5000:80 --link redis:redis
```

```
docker run -d --name=result –p 5001:80 --link db:db
```

```
docker run -d --name=worker --link db:db --link redis:redis
```



500 Internal Ser

Internal

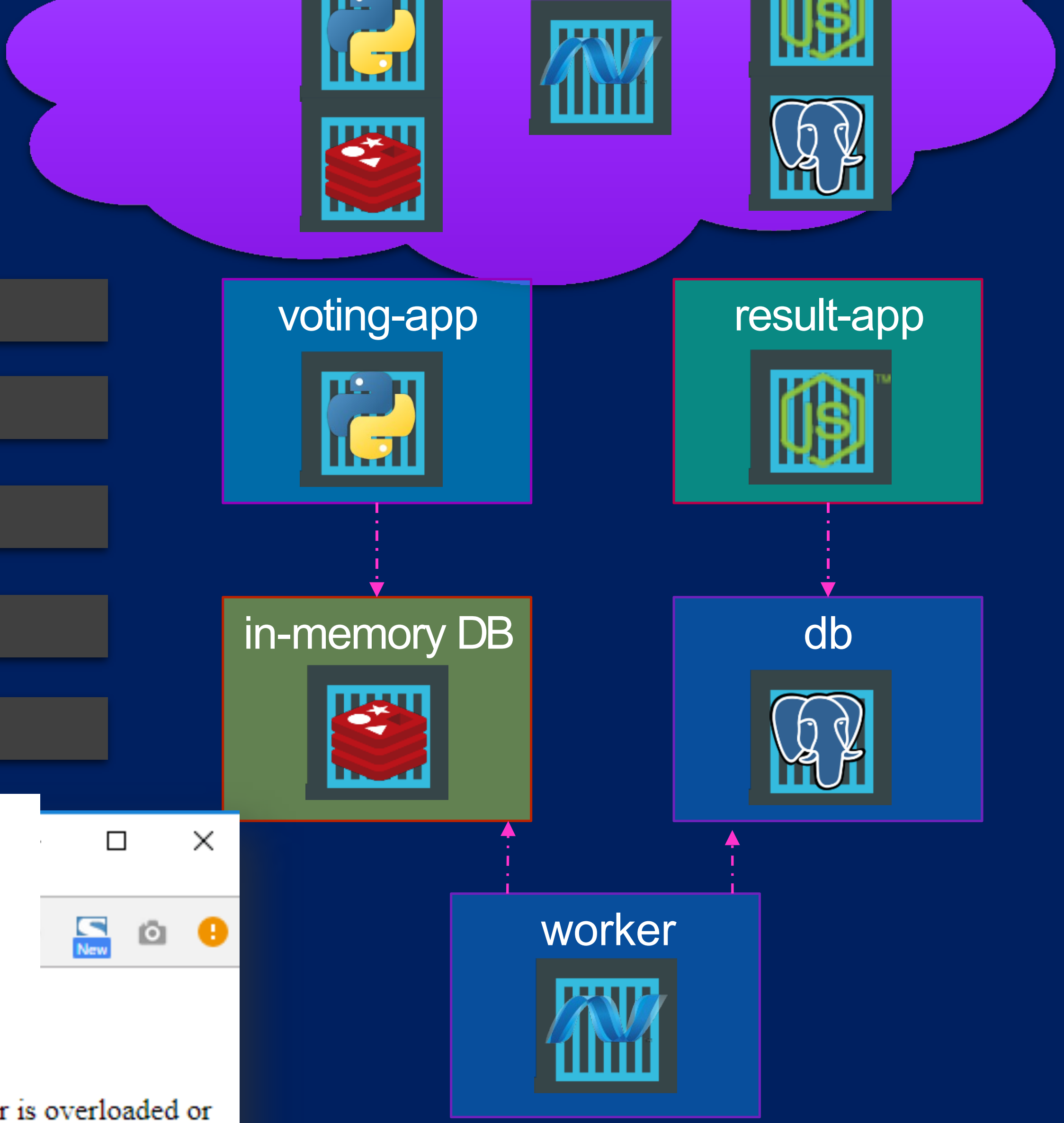
The server encount

r the server is overloaded or

there is an error in

```
try {
    Jedis redis = connectToRedis("redis");
    Connection dbConn = connectToDB("db");

    System.err.println("Watching vote queue");
```

```
127.0.0.1       localhost
::1     localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
172.17.0.2      redis 89cd8eb563da
172.17.0.3      ebcae9eb46bf
```



! Deprecation Warning

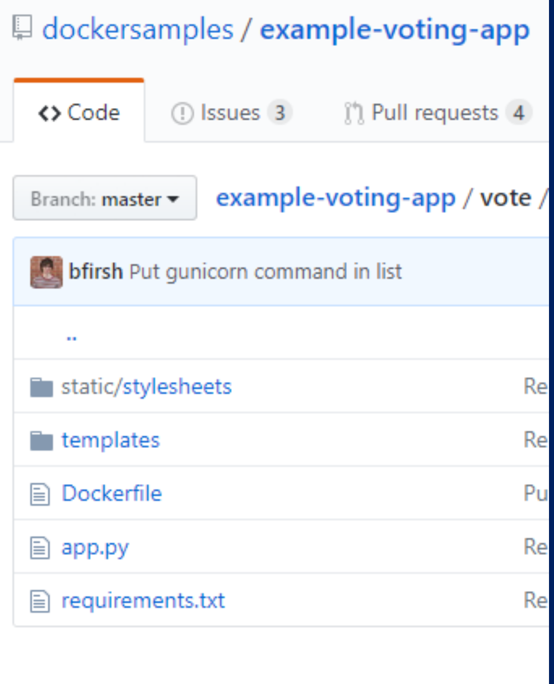
Docker compose - build

docker-compose.yml

```
redis:
    image: redis
db:
    image: postgres:9.4
vote:
    image: voting-app
    ports:
        - 5000:80
    links:
        - redis
result:
    image: result
    ports:
        - 5001:80
    links:
        - db
worker:
    image: worker
    links:
        - db
        - redis
```

docker-compose.yml

```
redis:
    image: redis
db:
    image: postgres:9.4
vote:
    build: ./vote
    ports:
        - 5000:80
    links:
        - redis
result:
    build: ./result
    ports:
        - 5001:80
    links:
        - db
worker:
    build: ./worker
    links:
        - db
        - redis
```



dockersamples / example-voting-app

example-voting-app / vote /

bfirsh Put gunicorn command in list

..

static/stylesheets Re

templates Re

Dockerfile Pu

app.py Re

requirements.txt Re

Docker compose - versions

docker-compose.yml

```
redis:
    image: redis
db:
    image: postgres:9.4
vote:
    image: voting-app
    ports:
        - 5000:80
    links:
        - redis
```

version: 1

docker-compose.yml

```
version: 2
services:
   redis:
       image: redis
   db:
       image: postgres:9.4
   vote:
       image: voting-app
       ports:
           - 5000:80
         depends_on:
             - redis
```

version: 2

docker-compose.yml

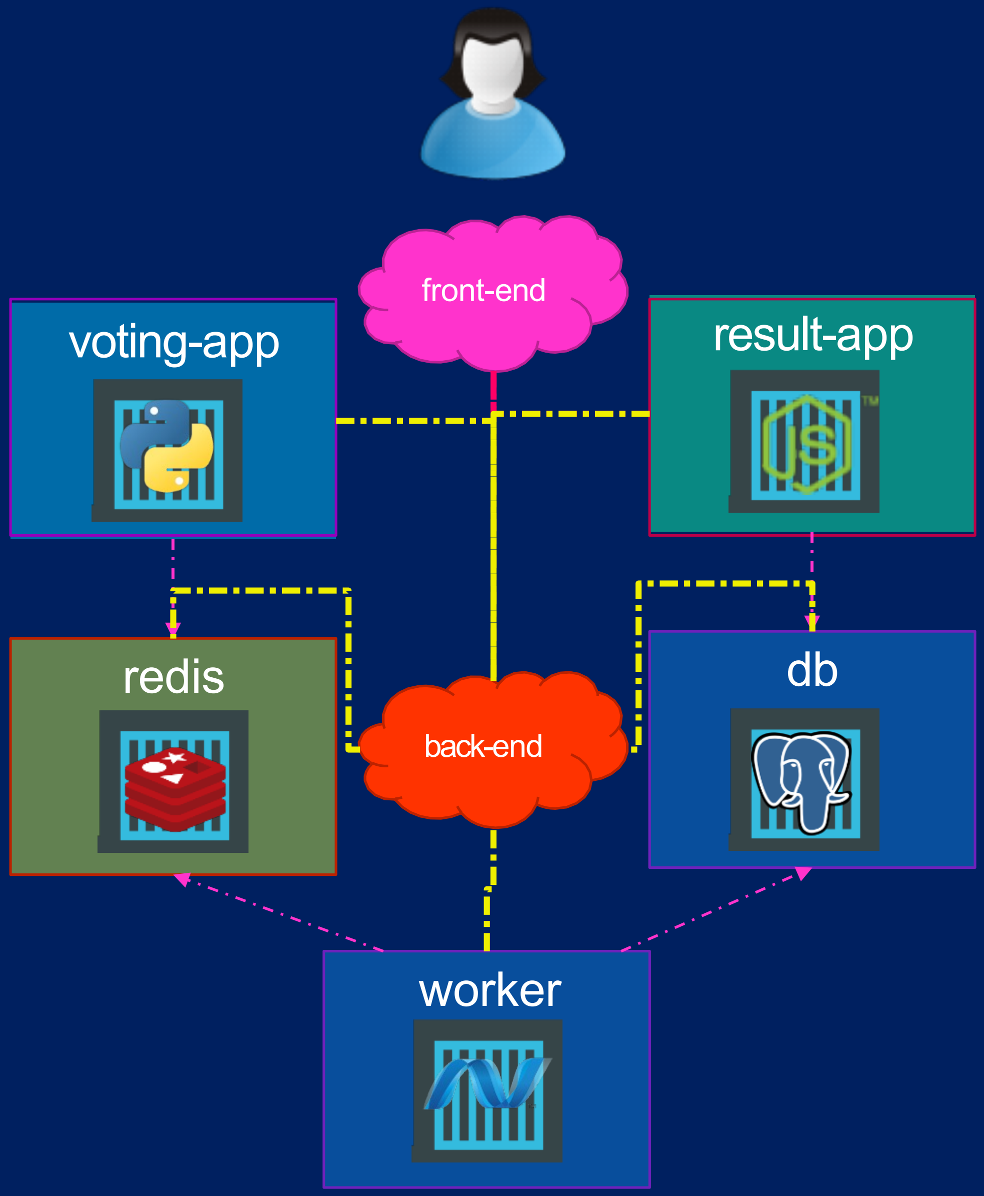
```
version: 3
services:
```

version: 3

Docker compose

docker-compose.yml

```
version: 2
services:
    redis:
        image: redis

        networks:

            - back-end
    db:
        image: postgres:9.4
        networks:
            - back-end

    vote:
        image: voting-app
        networks:
            - front-end
            - back-end
    result:
        image: result
        networks:
            - front-end
            - back-end

networks:
    front-end:
    back-end:
```


front-end
voting-app
result-app
redis
back-end
db
worker