

Components of FL architecture

Control Plane

Minikube to manage the cluster

Aggregator

FL Protocol: To define the set of steps and periodically run and monitor the FL steps

Secure Aggregator: To perform secure aggregation of gradients

Privacy Manager: To ensure privacy of received model updates

Coordinator: To receive and manage the model updates (gradients, global model, etc)

Trainer

MONAI: To run the complete model development workflows

Substra: To run decentralized model development nodes

Reporter: To report the periodic model updates and manage the local models

Data Manager: To query and fetch the data from data providers and manage local copies for current pipelines

Data Provider

FLIP: To provide medical data streams

Other: To provide custom, synthetic, and 3rd party datasets

Docker Commands Cheet sheet

- **Starting a container**

`docker run image-name`

- **Listing all containers**

`docker ps`

`docker ps -a`

- **Stop all containers**

`docker stop container-name or container ID`

- **To remove the container**

`docker rm image name or container id`

- **To see all images**

`docker images`

- **To remove images**

`docker rmi image-name`

- **Download the images**

`docker pull image-name`

- **Docker attach/detach**

`docker run container-name`

`docker run -d container-name` (detached mode)

`docker run -it container-name` (interactive terminal)

`docker run -p 80:5000 container-name` (port mapping)

`docker run -v /opt/datadir:/var/lib/mysql container-name`

the above document attaches persistent storage volume by mapping external directory with a directory inside container

- **Inspect containers**

`docker inspect container-name`

- **Container logs**

`docker logs container-name`

- **Environment variables**

`docker run -e (variable with value) container-name`

- **Create image**

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
```

- **Environment variables**

`docker run -e (variable with value) container-name`

`docker run image-name command`

FROM Ubuntu

CMD sleep 5

CMD command param1

CMD sleep 5

CMD ["command", "param1"]

CMD ["sleep", "5"]

CMD ["sleep 5"]



```
▶ docker build -t ubuntu-sleeper .
```

```
▶ docker run ubuntu-sleeper
```

FROM Ubuntu

ENTRYPOINT ["sleep"]

CMD ["5"]

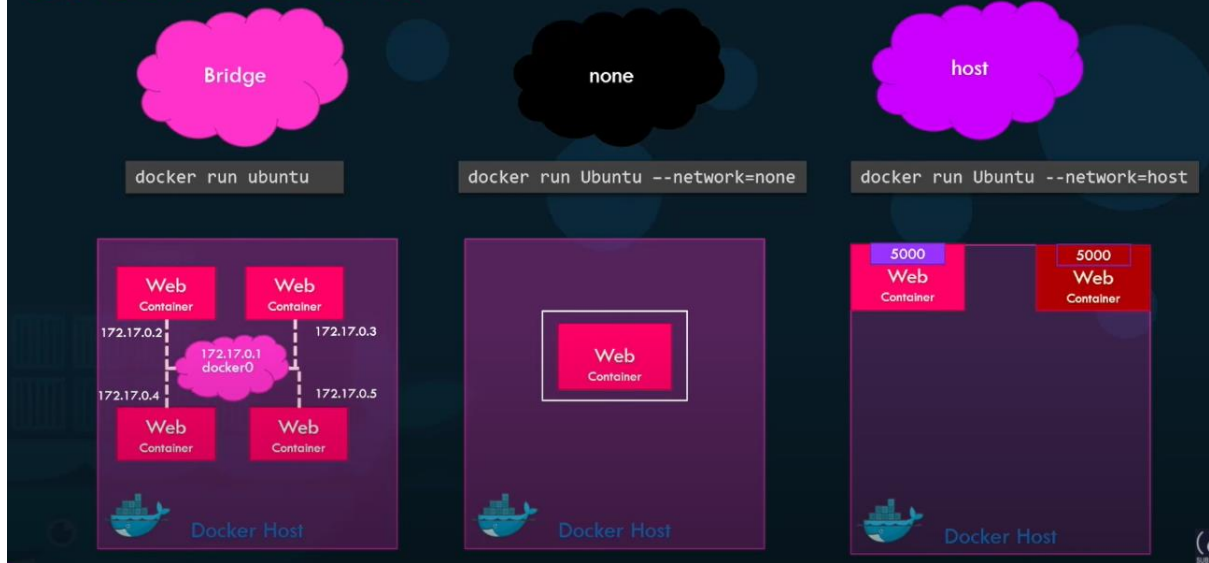
Command at Startup: sleep 5

```
▶ docker run ubuntu-sleeper
```

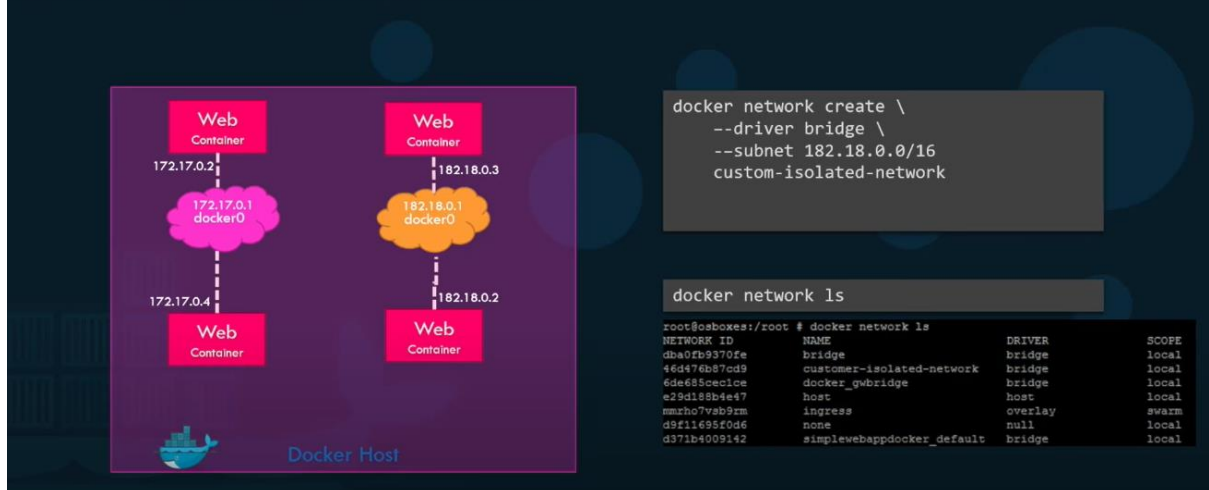
```
sleep: missing operand  
Try 'sleep --help' for more information.
```

```
▶ docker run ubuntu-sleeper 10
```

Default networks



User-defined networks



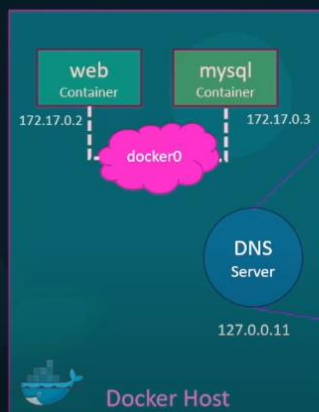
Inspect Network

```
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

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	0 B

Dockerfile2

```
FROM Ubuntu

RUN apt-get update && apt-get -y install python

RUN pip install flask flask-mysql

COPY app2.py /opt/source-code

ENTRYPOINT FLASK_APP=/opt/source-code/app2.py flask
run
```

```
docker build Dockerfile2 -t mmumshad/my-custom-app-2
```

Layer 1. Base Ubuntu Layer	0 MB
Layer 2. Changes in apt packages	0 MB
Layer 3. Changes in pip packages	0 MB
Layer 4. Source code	229 B
Layer 5. Update Entrypoint	0 B

Layered architecture

Container Layer

Read Write

Layer 6. Container Layer

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

Image Layers

Read Only

Layer 5. Update Entrypoint with "flask" command

Layer 4. Source code

Layer 3. Changes in pip packages

Layer 2. Changes in apt packages

Layer 1. Base Ubuntu Layer

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

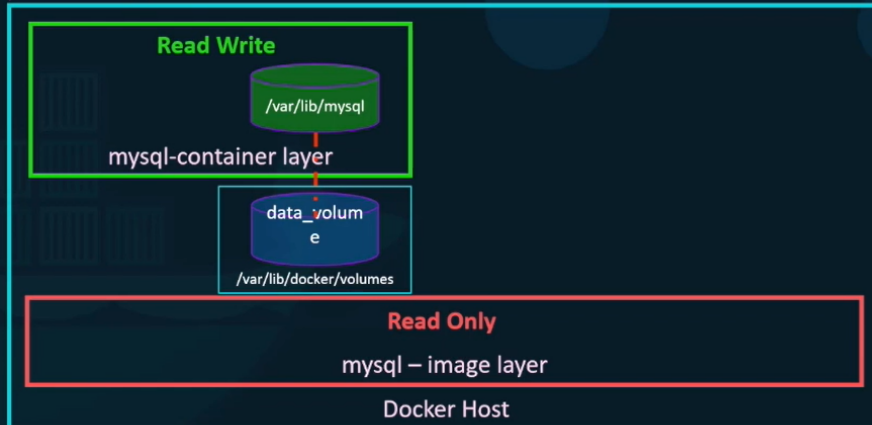
volumes

```
docker volume create data_volume
```

```
docker run -v data_volume:/var/lib/mysql mysql
```

```
docker run -v data_volume2:/var/lib/mysql mysql
```

/var/lib/docker
├── volumes
└── data_volume



volumes

```
docker volume create data_volume
```

```
docker run -v data_volume:/var/lib/mysql mysql
```

```
docker run -v data_volume2:/var/lib/mysql mysql
```

```
docker run -v /data/mysql:/var/lib/mysql mysql
```

```
docker run \  
--mount type=bind,source=/data/mysql,target=/var/lib/mysql mysql
```

/var/lib/docker
├── volumes
└── data_volume



SUBSCRIBE

Docker compose

```
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
```

Public Docker registry - dockerhub

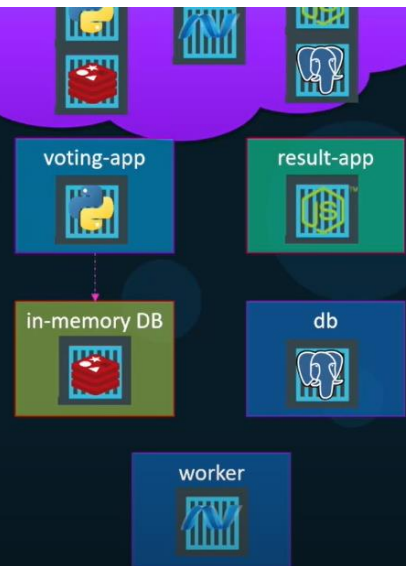


docker run --links

```
docker run -d --name=redis redis
docker run -d --name=db postgres:9.4 result-app
docker run -d --name=vote -p 5000:80 --link redis:redis voting-app
docker run -d --name=result -p 5001:80
docker run -d --name=worker worker
```

```
def get_redis():
    if not hasattr(g, 'redis'):
        g.redis = Redis(host="redis", db=0, socket_timeout=5)
    return g.redis
```

```
/app # cat /etc/hosts
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 890d9eb563da
172.17.0.3 ebcac9eb46bf
```



Docker compose

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

```
docker run -d --name=db postgres:9.4
```

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

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

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

db:db = db

```
docker-compose up
```

```
docker-compose.yml
```

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

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

Code Issues Pull requests

Branch: master 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

version: 1

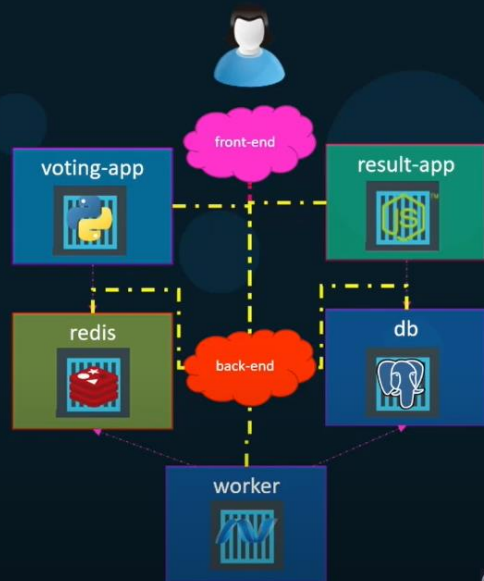
version: 2

```
docker-compose.yml
```

version: 3

Docker compose

```
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:
```



Private Registry

```
▶ docker login private-registry.io
```

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 run -d -p 5000:5000 --name registry registry:2
```

```
▶ docker image tag my-image localhost:5000/my-image
```

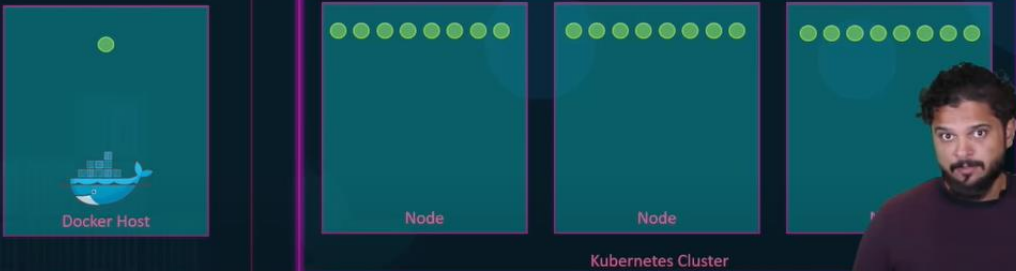
```
▶ docker push localhost:5000/my-image
```

```
▶ docker pull localhost:5000/my-image
```

```
▶ docker pull 192.168.56.100:5000/my-image
```

`docker run my-web-server`

`kubectl run --replicas=1000 my-web-server`



The diagram illustrates the difference in scaling between Docker and Kubernetes. On the left, a single 'Docker Host' is shown as a teal square containing a Docker logo and a single green dot representing a pod. On the right, a 'Kubernetes Cluster' is shown as a larger teal square containing three smaller teal squares, each labeled 'Node'. Each node contains a row of ten green dots, representing 1000 replicas of a pod. A man is standing in front of the Kubernetes Cluster diagram.

Docker Host

Node

Node

Kubernetes Cluster

kubectl

```
kubectl run hello-minikube
```

```
kubectl cluster-info
```

```
kubectl get nodes
```

