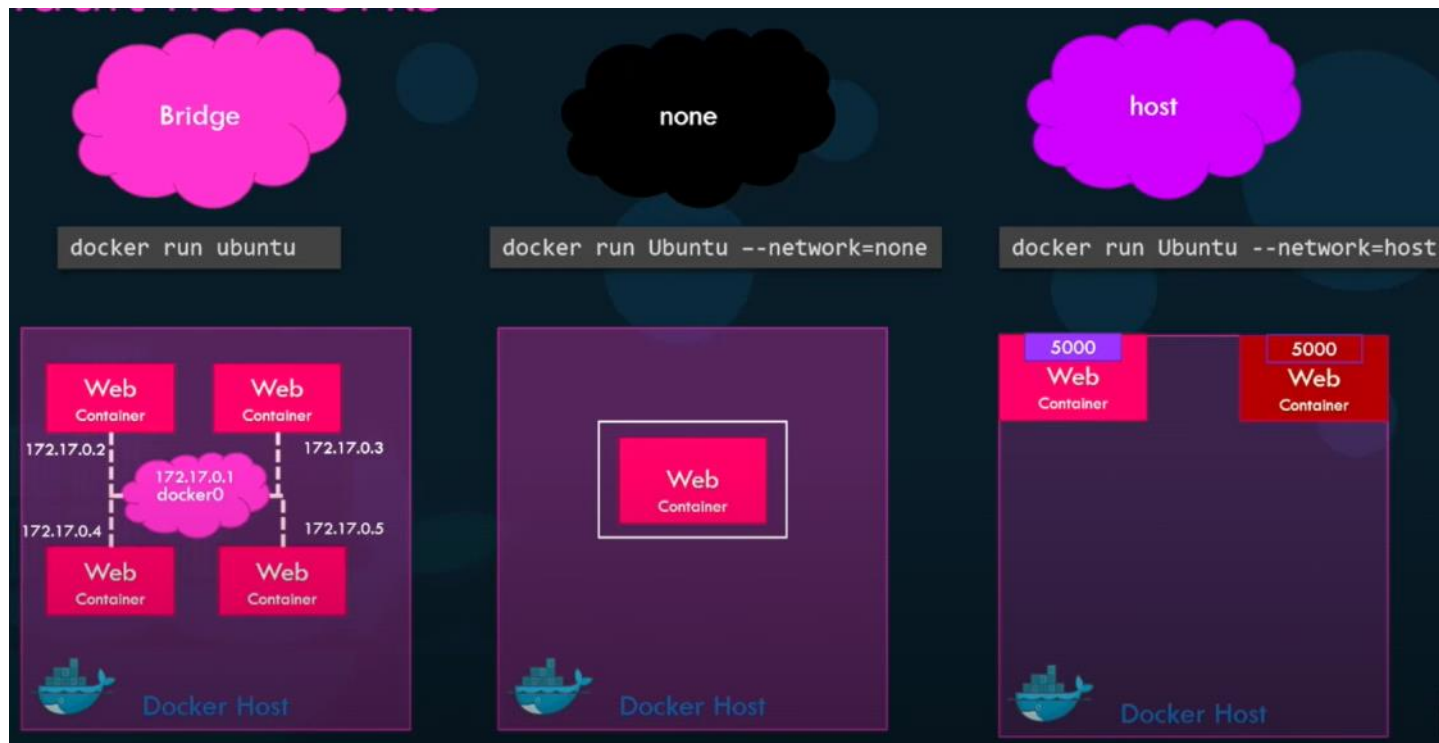


Networking with Docker

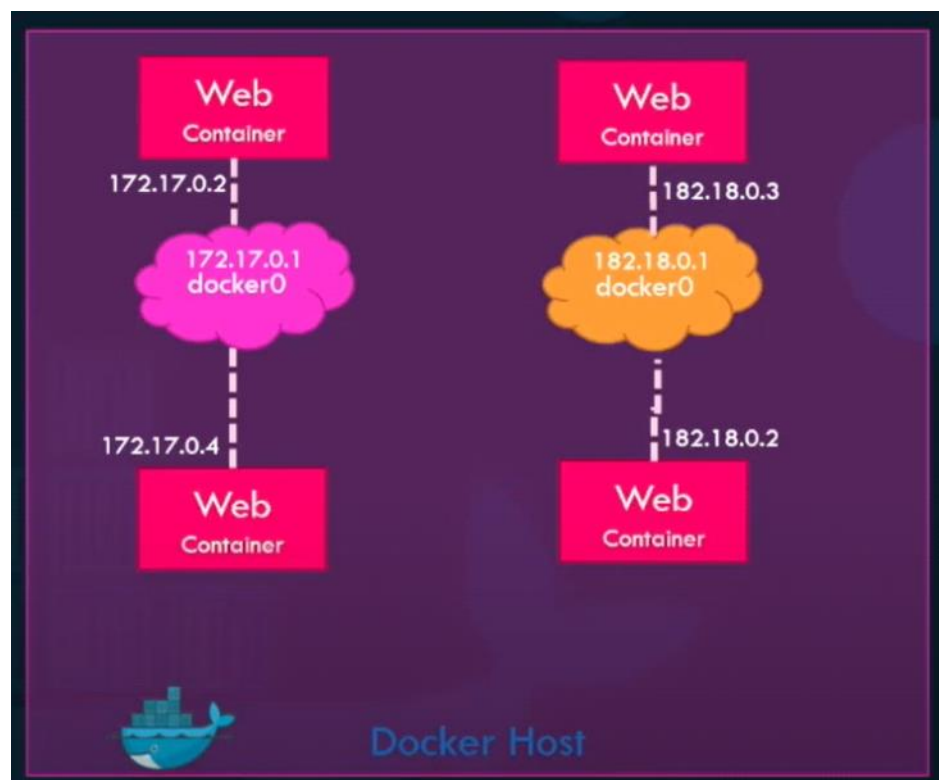
28 February 2022 08:44

Default networks



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

// List networks with
`docker network ls`

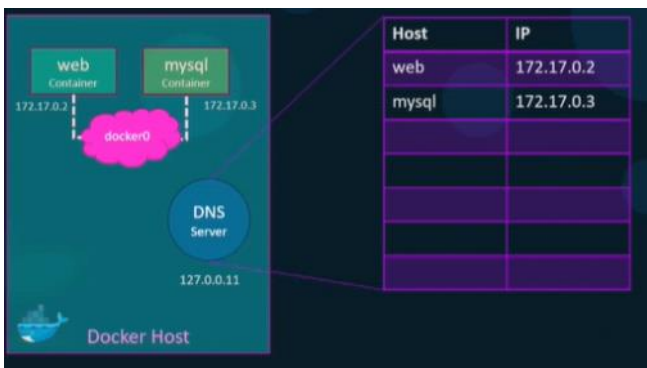


docker inspect blissful_hopper

```
[
  {
    "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

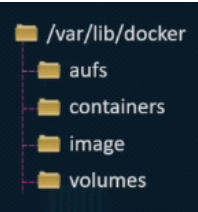
mysql.connect(mysql)



Docker Storage

28 February 2022 09:34

File System



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

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	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

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	

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

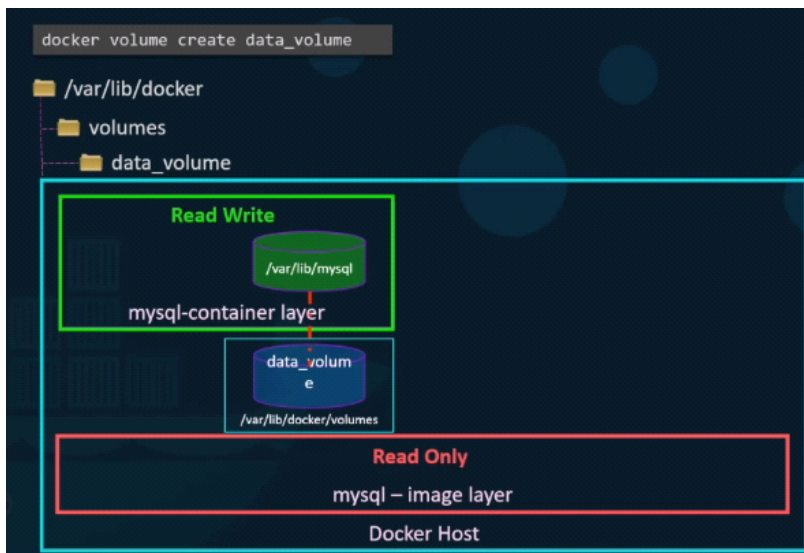
COPY-ON-WRITE



When container is stopped and removed the changed app.py and temp.txt gets removed as well. So what if we want the data to persist?

docker volume create data_volume

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



```
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
```

Docker Compose

28 February 2022 10:40

The slide is titled "docker run --links" in pink text. It lists five Docker commands in a terminal-like font:

```
docker run -d --name=redis redis
docker run -d --name=db postgres:9.4 --link db:db 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 --link db:db --link redis:redis worker
```

Below the commands is a code block with a Java snippet:

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

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

On the right side of the slide is a network diagram. It shows five services in colored boxes: "voting-app" (blue), "result-app" (green), "in-memory DB" (yellow), "db" (blue), and "worker" (blue). Arrows indicate dependencies: "voting-app" depends on "in-memory DB"; "result-app" depends on "db"; "worker" depends on both "in-memory DB" and "db". Above the services are three icons representing Redis (a red cylinder), PostgreSQL (a blue elephant), and a generic database icon.

After we have a list of these docker commands, creating a yaml file is easy
*****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:
    -redis
    -db
*****
```

//Now run
docker-compose up -d

If we haven't created an image for docker for our project we can instead use the following in docker compose:

```
*****docker-compose.yml*****
redis:
  image:redis
db:
  image:postgres:9.4
vote:
  build:./voting-app
  ports:
    -5000:80
  links:
    -redis
```


result:

build:./result

ports:

-5001:80

links:

-db

worker:

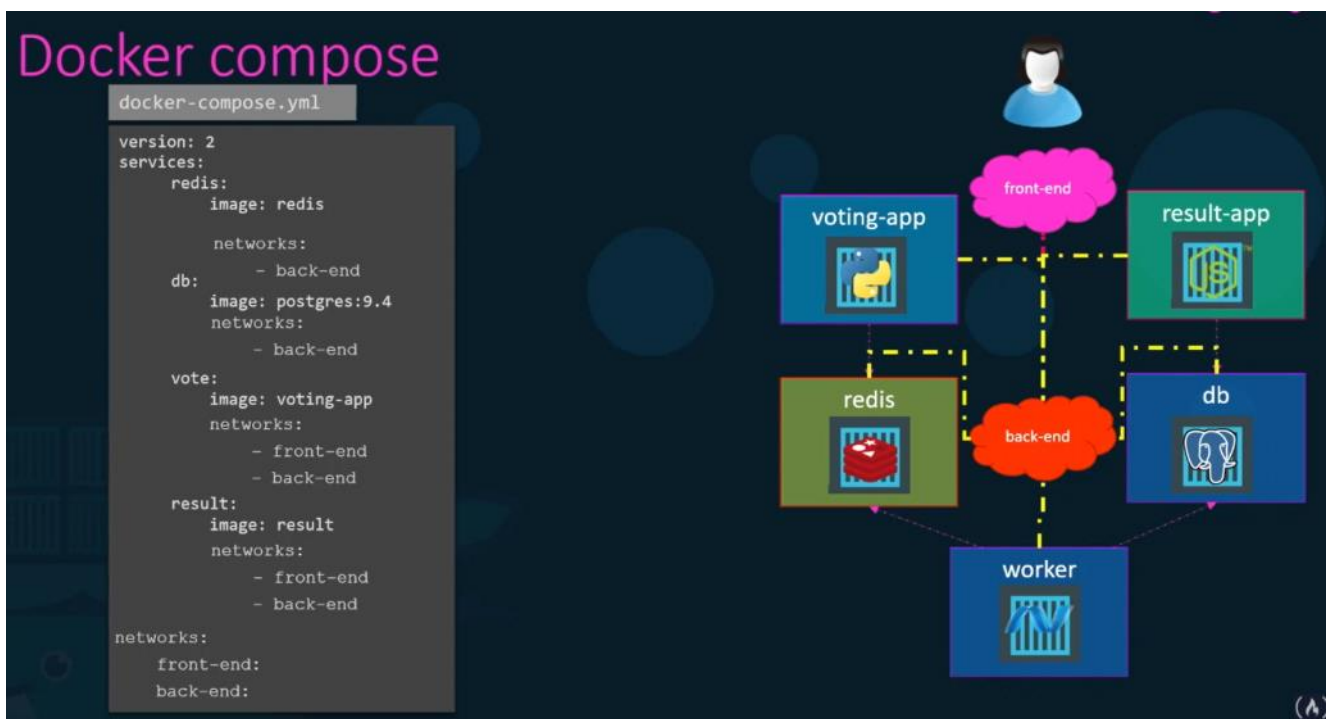
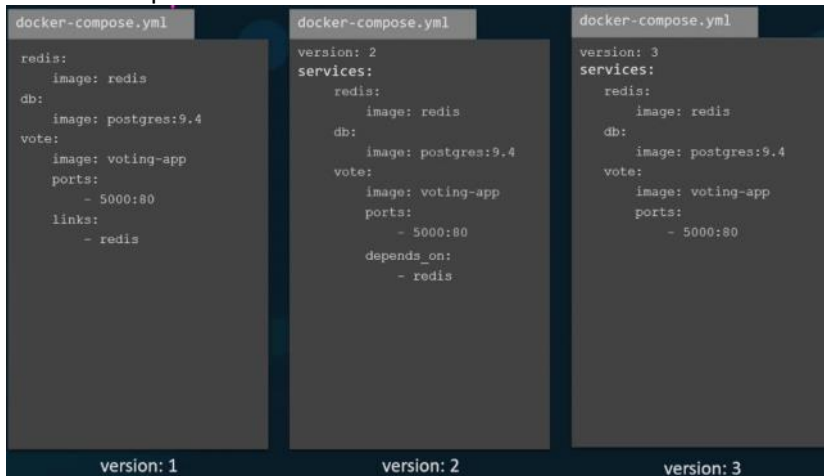
build:./worker

links:

-redis

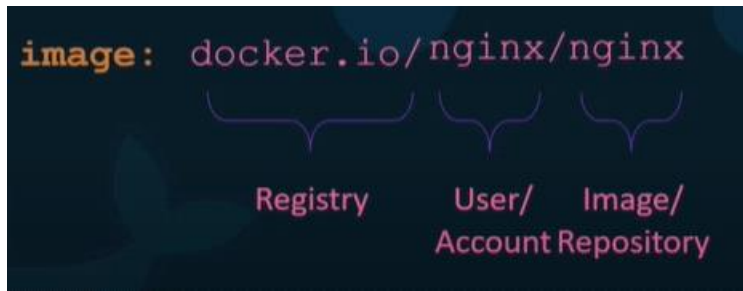
-db

Docker compose -versions



Registry

01 March 2022 08:45



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 on Premise

```
▶ 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 Engine

01 March 2022 08:48

