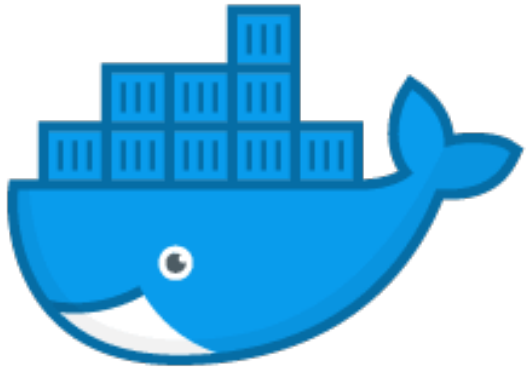


Docker

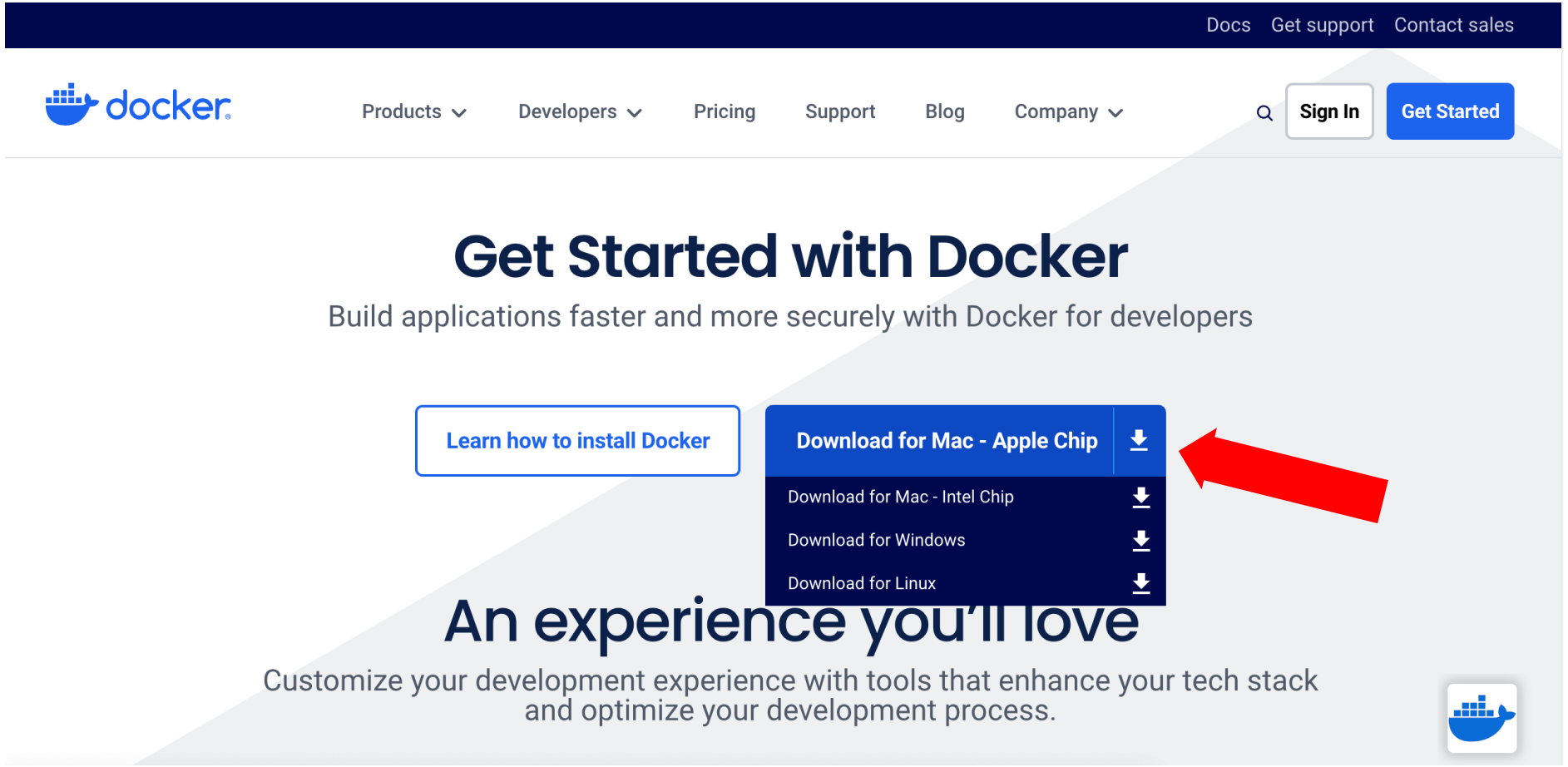
2110366 SE2 2566/2

https://github.com/2110366-2566-2/DockerPractices

Today objectives

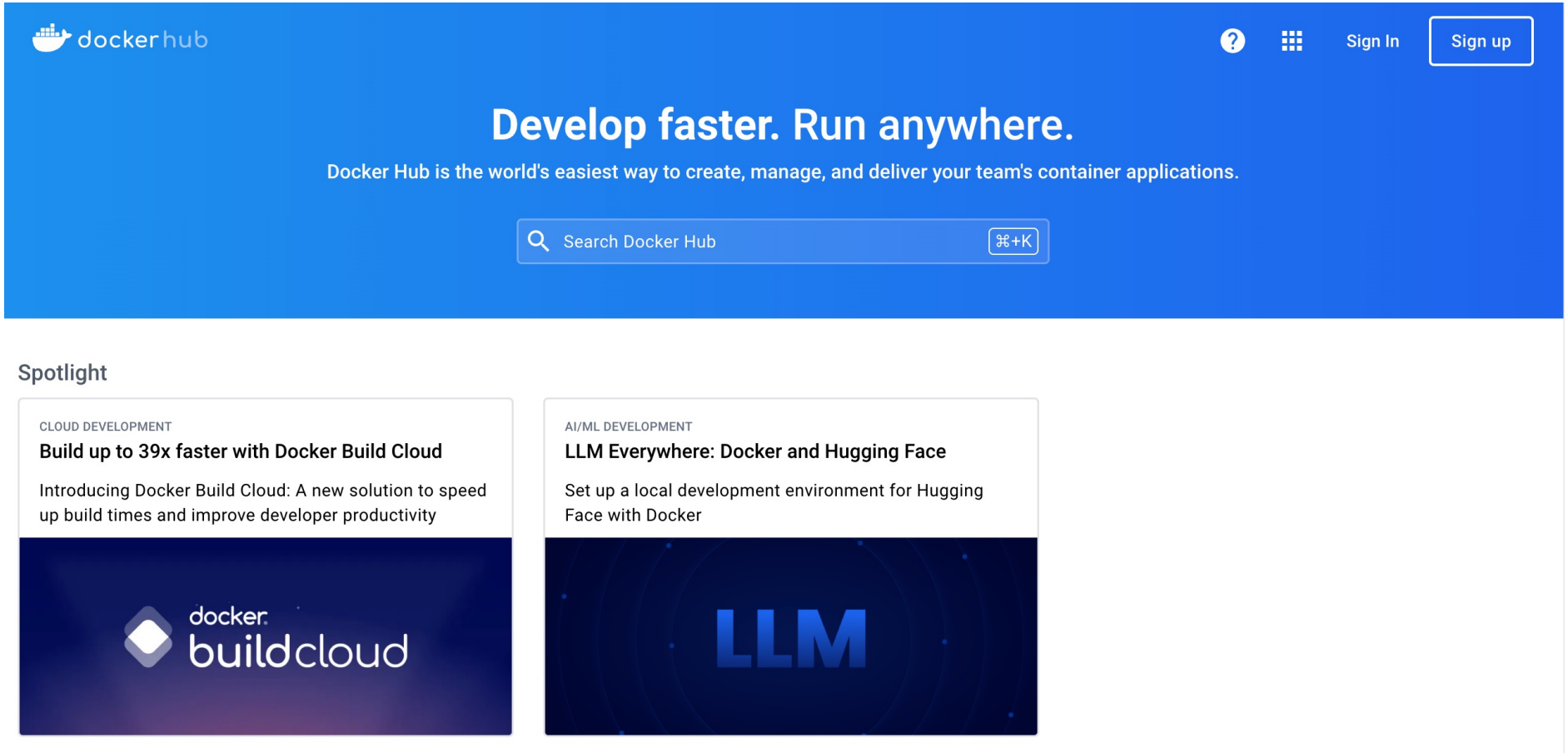
- ❑ Learn some basic concepts about containerization in Docker
- ❑ Be familiar with Docker's basic commands and be able to use them correctly.
- ❑ Can dockerize subsystems in your term project.

Install Docker engine and Docker Desktop



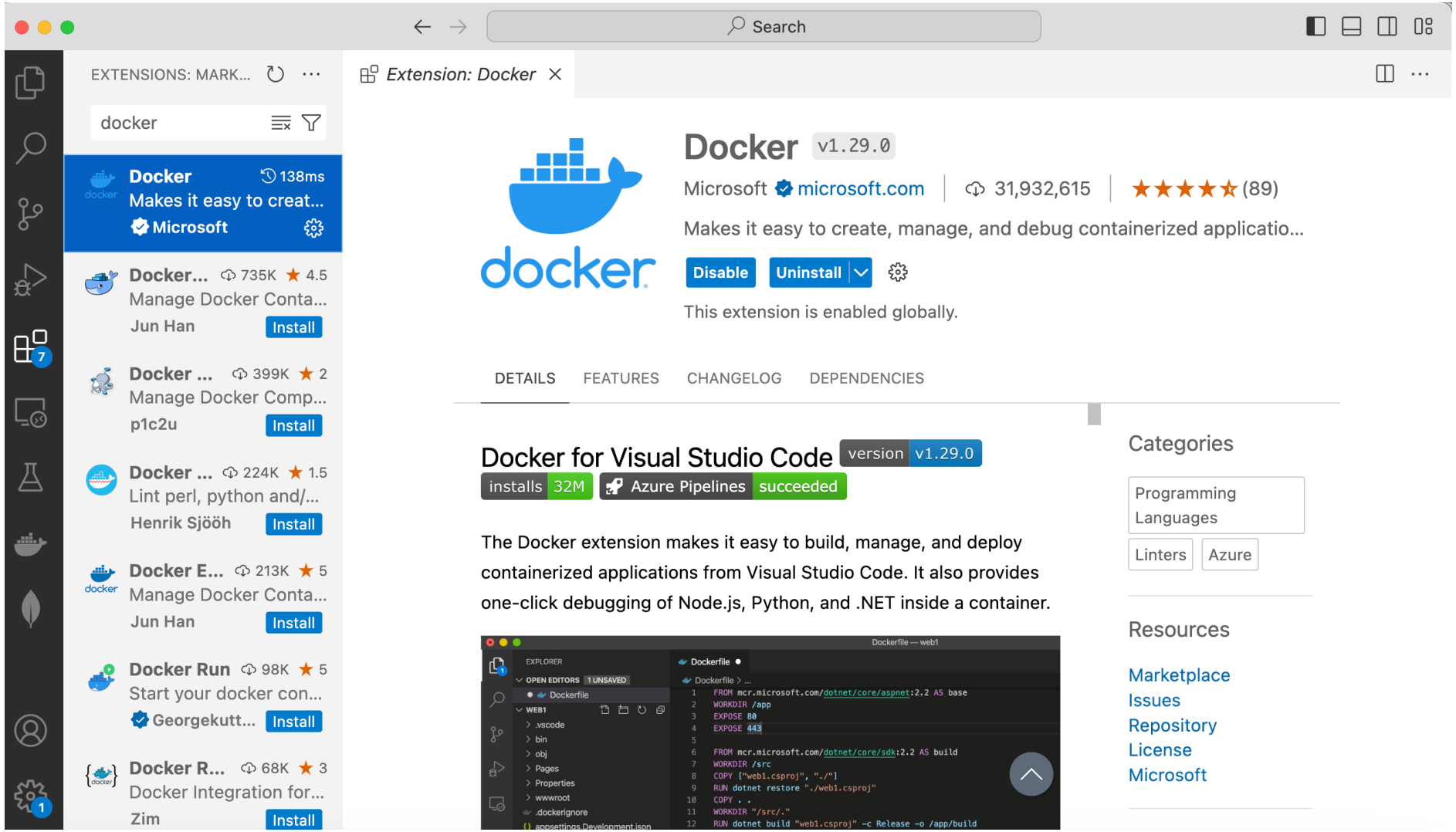
https://www.docker.com/get-started/

Create an account at Docker Hub



https://hub.docker.com/

Install Docker extension in VS code



Keywords

Unlike VM

Share files among containers

Dokerfile

Docker image

Port forwarding

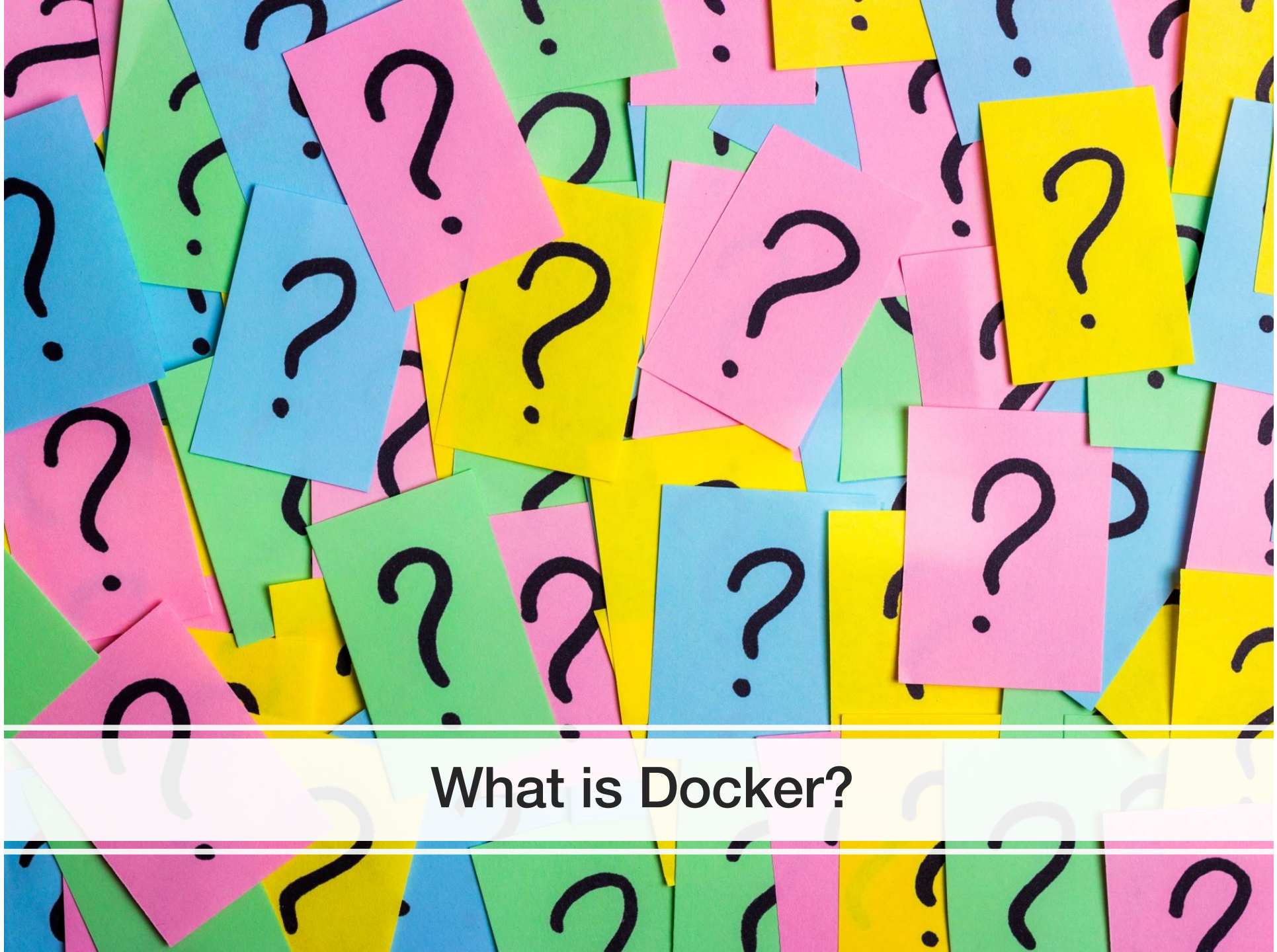
Docker hub

Container

Some docker commands

Docker compose

Volume



What is Docker?

Docker

Docker is a software platform that…

- ❑ Helps package a software to be runnable on any hardware

- ❑ Lets a developer create an *image*; package a *software,* from Dockerfile and then run instances of that image in a *container*.

- ❑ Enables a flexible and scalable deployment based on containerization

Docker main components

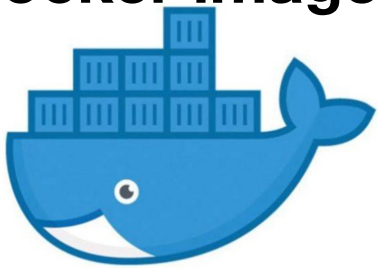
Dockerfile

```
Dockerfile > ...
1   FROM node:lts-alpine
2   ENV NODE_ENV=production
3   WORKDIR /usr/src/app
4   COPY ["package.json", "package-lock.json*", "npm-shrinkwrap.json*", "./"]
5   RUN npm install --production --silent && mv node_modules ../
6   COPY . .
7   EXPOSE 3000
8   RUN chown -R node /usr/src/app
9   USER node
10  CMD ["npm", "start"]
```

เพื่อ build environment
สำหรับการ deploy software

build

Docker image



deposit

Docker hub/
GitHub Packages

An online repository of docker container images

run/
deploy

Docker container

run/deploy on a virtual machine, e.g., EC instance or a Kubernetes cluster

Why containerization?

Before

Different of installation process on different OS

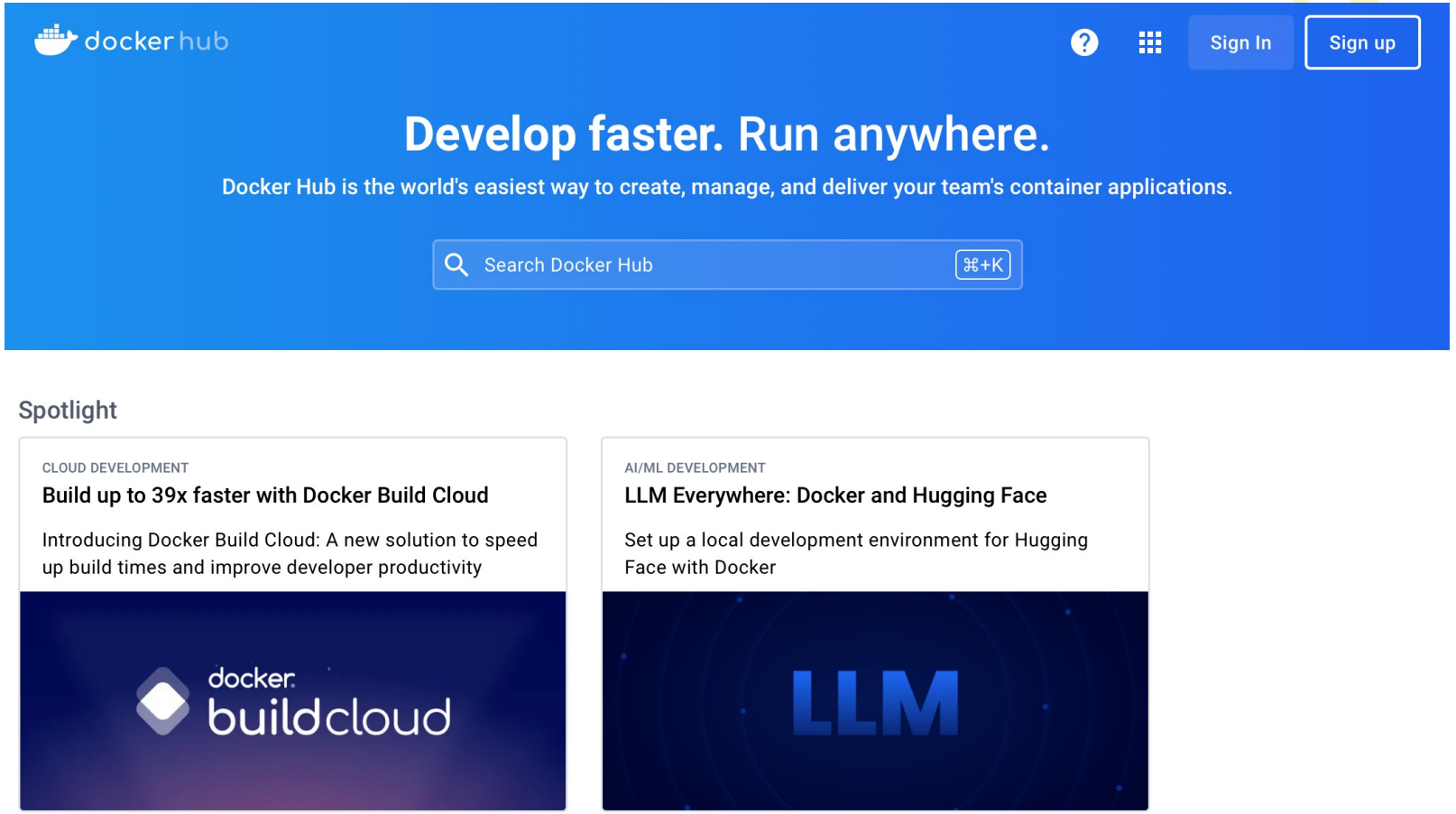
MySQL, mongodb, PostgreSQL, Redis, etc..

Python, node.js, Go, PHP, Java, R, etc.

After

1. **Own isolated environment**
2. **Packed with all needed configurations / dependencies**
3. **Easy installation**
4. **Can run the same app with two different versions**
5. **Easy and quick deployment**

Docker Hub



Docker hub (https://hub.docker.com) is a huge repository for docker images!!

docker

docker hub Explore Repositories Organizations ubuntu W

Filters

1 - 25 of 10,000 results for **ubuntu**.

Best Match

Products

- [ ] Images
- [ ] Extensions
- [ ] Plugins

Trusted Content

- [ ] Docker Official Image
- [ ] Verified Publisher
- [ ] Sponsored OSS

Operating Systems

- [ ] Linux
- [ ] Windows

ubuntu 1B+ • ☆10K+

Updated 18 days ago

Ubuntu is a Debian-based Linux operating system based on free software.

Linux 386 riscv64 x86-64 ARM ARM 64 PowerPC 64 LE IBM Z

Pulls: 29,607,640
Last week

Learn more

websphere-liberty 10M+ • ☆298

Updated 4 days ago

WebSphere Liberty multi-architecture images based on Ubuntu 18.04

Linux 386 x86-64 PowerPC 64 LE IBM Z ARM 64

Pulls: 5,082
Last week

Learn more

docker

docker hub Explore Repositories Organizations ubuntu W

Explore / Official Images / ubuntu

ubuntu

Docker Official Image • 1B+ • ☆10K+

Ubuntu is a Debian-based Linux operating system based on free software.

```
docker pull ubuntu
```
Copy

Overview Tags

Quick reference

- **Maintained by**:
  Canonical
- **Where to get help**:
  the Docker Community Slack, Server Fault, Unix & Linux, or Stack Overflow

Supported tags and respective `Dockerfile`

Recent Tags

rolling noble-20240225 noble
mantic-20240216 mantic latest
jammy-20240227 jammy focal-20240216
focal

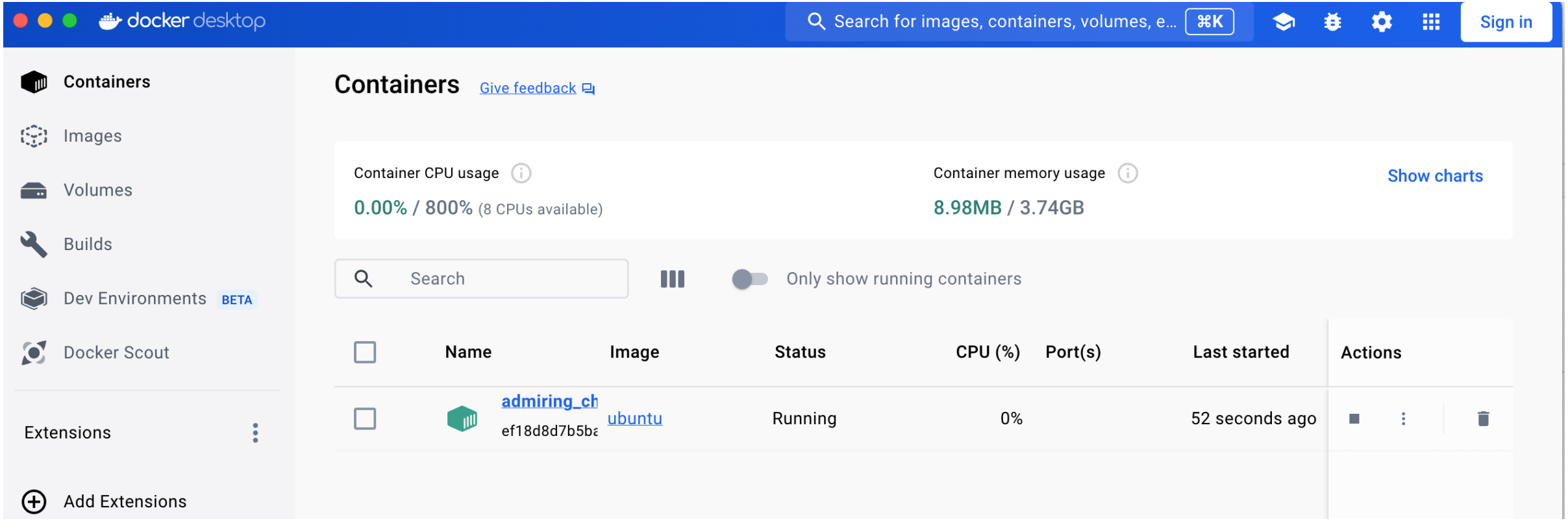
About Official Images

Try this

```
docker run -it ubuntu
```

```
docker ps
```

A list of running containers are also shown in the Docker Desktop



Try this at the VS code's terminal

```
docker run -it ubuntu
```

By default, a container's file system persists even after the container exits.

How about we also specify the specific version we need?

Hint: observe the printouts of the docker

```
docker run --rm it ubuntu
```

Automatically clean up the container and remove the file system when the container exits

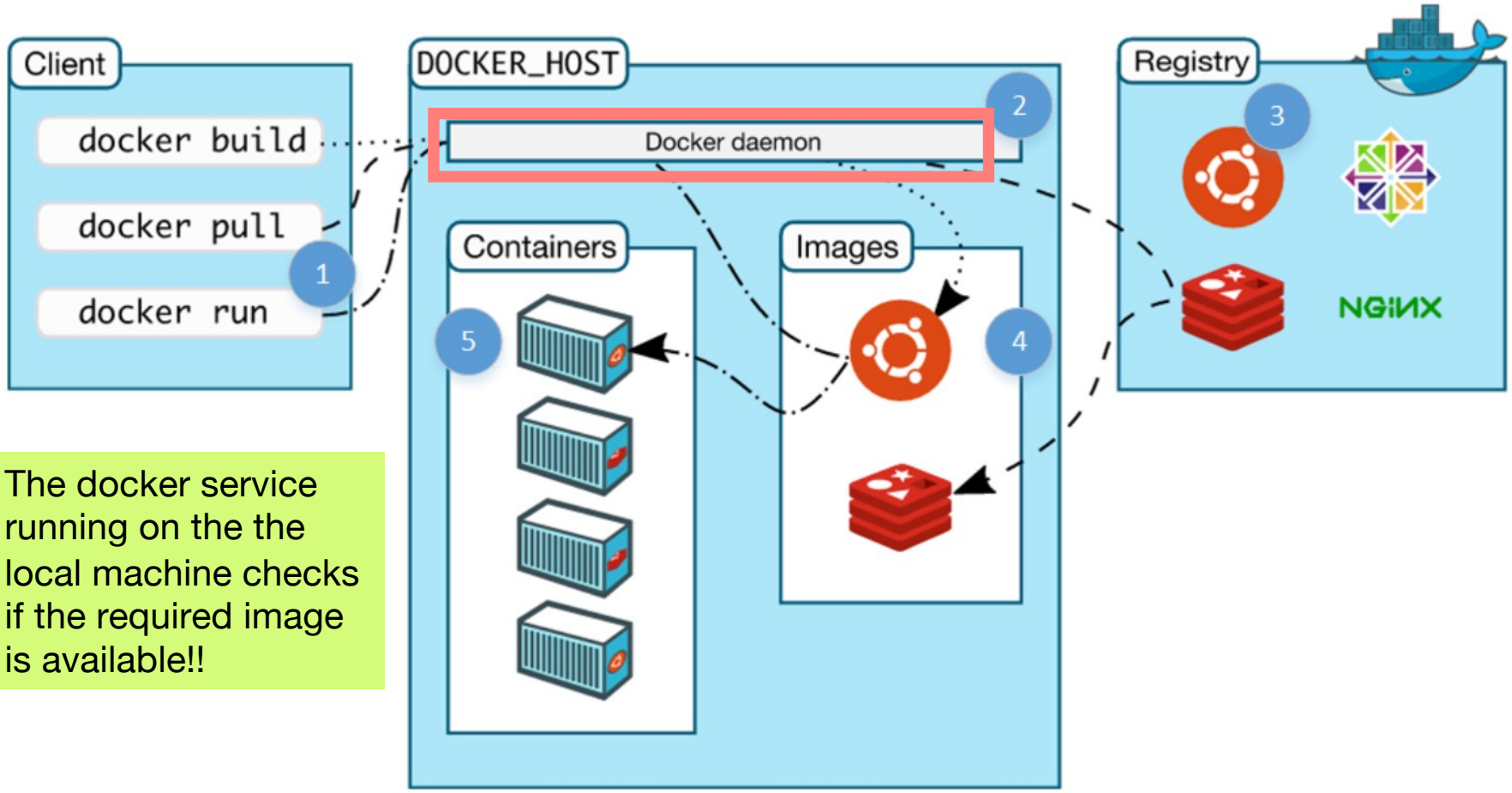
What you will get is as follows….

```
(base) Duangdaos-MacBook-Pro:~ wichadak$ docker run -it ubuntu
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
423ae2b273f4: Pull complete
de83a2304fa1: Pull complete
f9a83bce3af0: Pull complete
b6b53be908de: Pull complete
Digest: sha256:04d48df82c938587820d7b6006f5071dbbffceb7ca01d2814f81857c631d44df
Status: Downloaded newer image for ubuntu:latest
root@38a8c85f56e1:/#
root@38a8c85f56e1:/#
root@38a8c85f56e1:/#
root@38a8c85f56e1:/#
```

Exit from ubuntu

Try → docker ps / docker ps –a on your system's shell

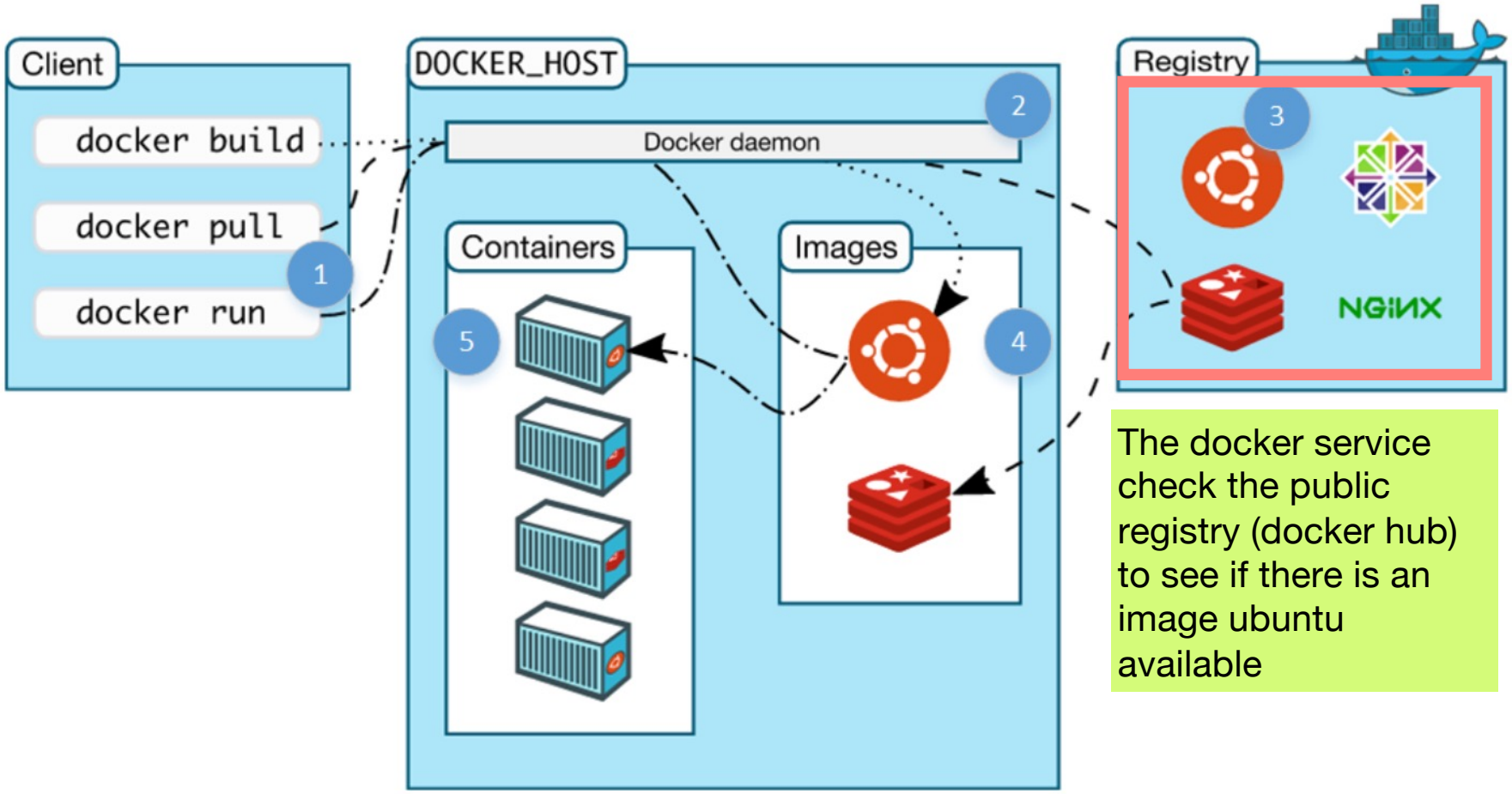
What has happened?



The docker service running on the the local machine checks if the required image is available!!

http://www.dwmkerr.com/learn-docker-by-building-a-microservice/

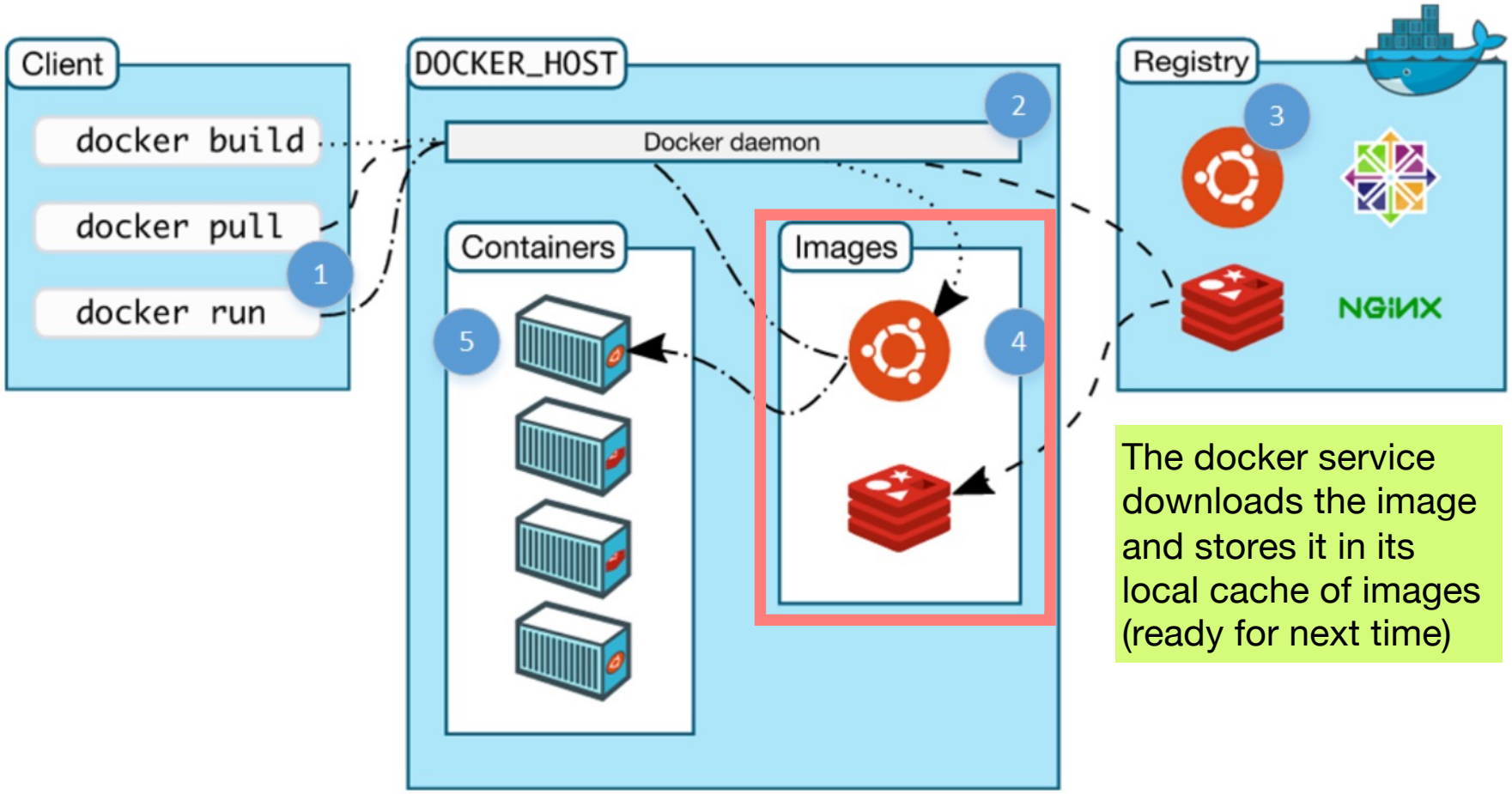
What has happened?



The docker service check the public registry (docker hub) to see if there is an image ubuntu available

http://www.dwmkerr.com/learn-docker-by-building-a-microservice/

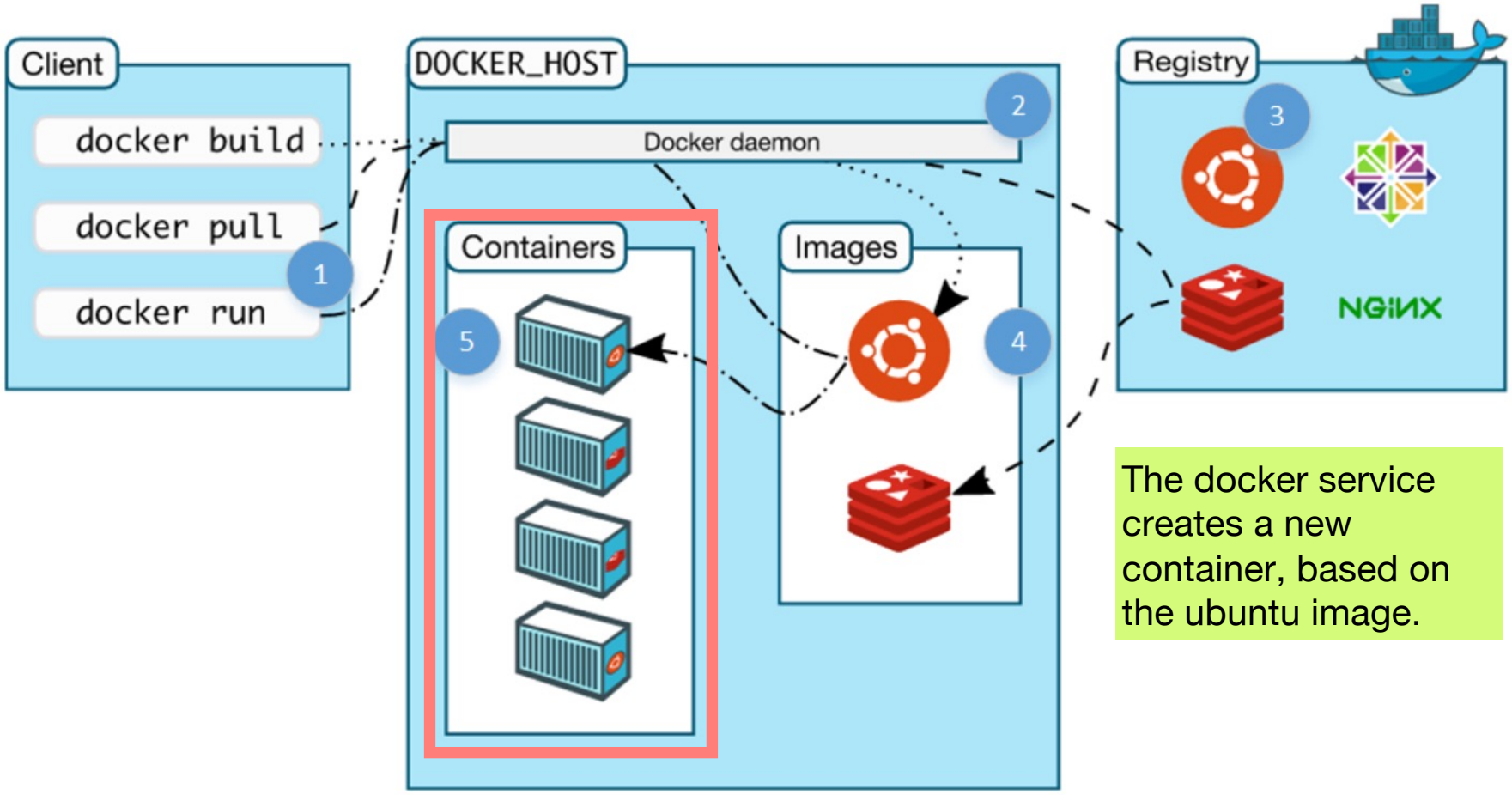
What has happened?



The docker service downloads the image and stores it in its local cache of images (ready for next time)

http://www.dwmkerr.com/learn-docker-by-building-a-microservice/

What has happened?

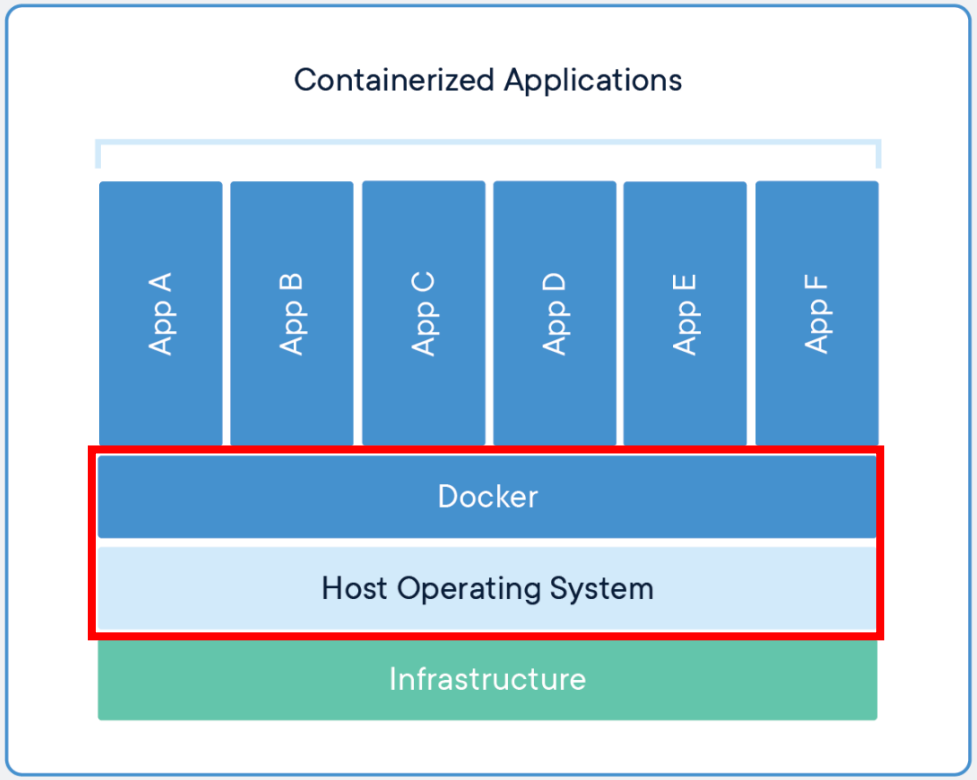


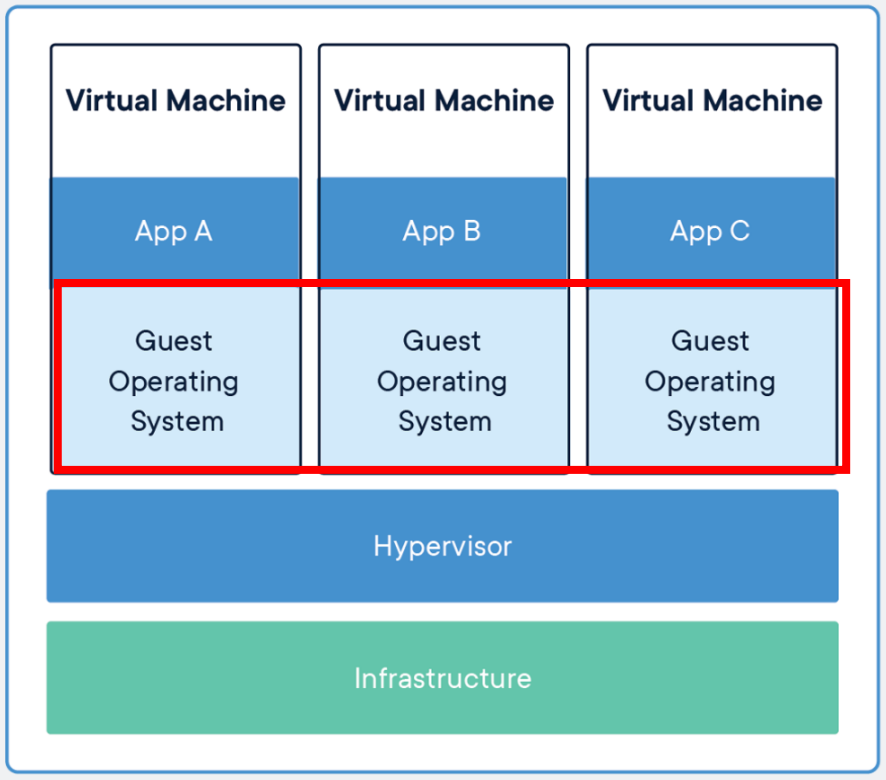
The docker service creates a new container, based on the ubuntu image.

Notice that different line patterns represent different command

http://www.dwmkerr.com/learn-docker-by-building-a-microservice/

Compare between containers and virtual machines (VMS)



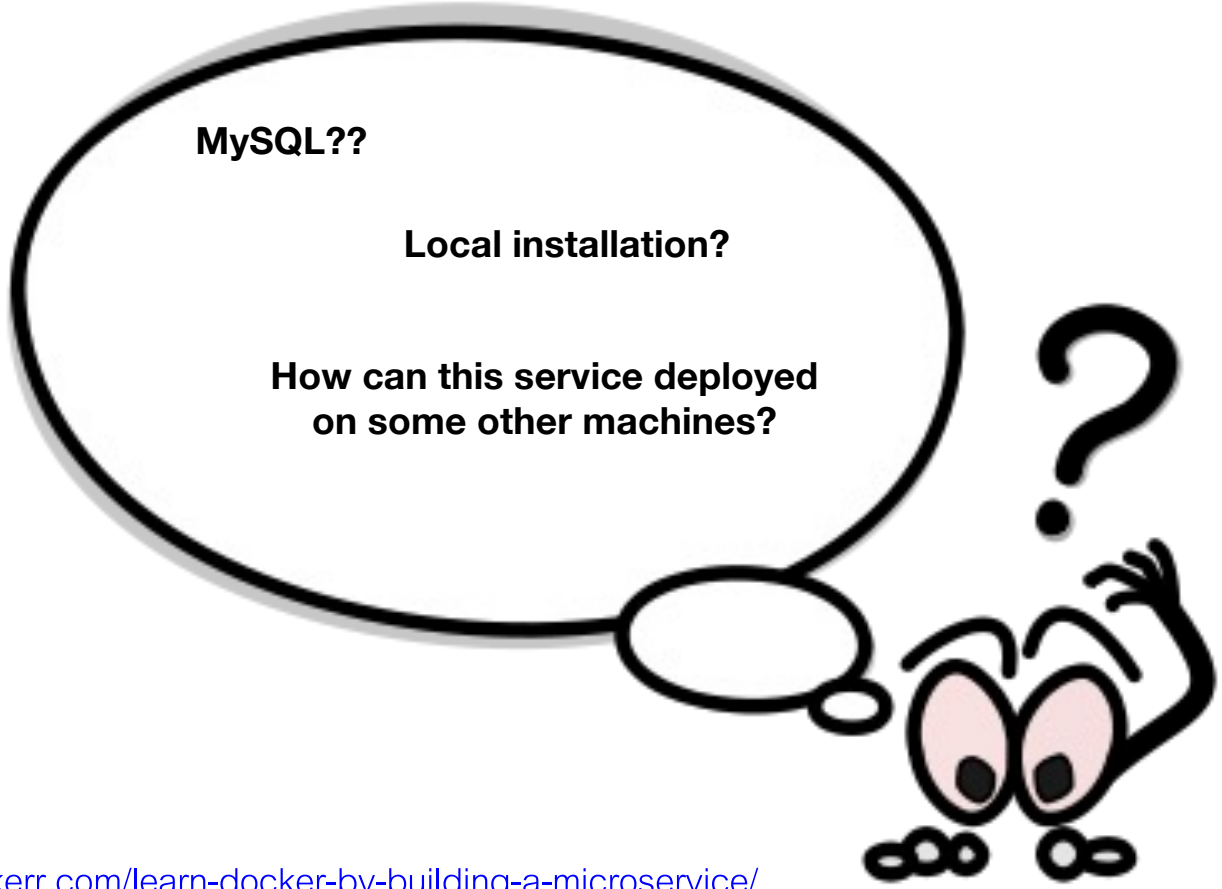


https://www.docker.com/resources/what-container

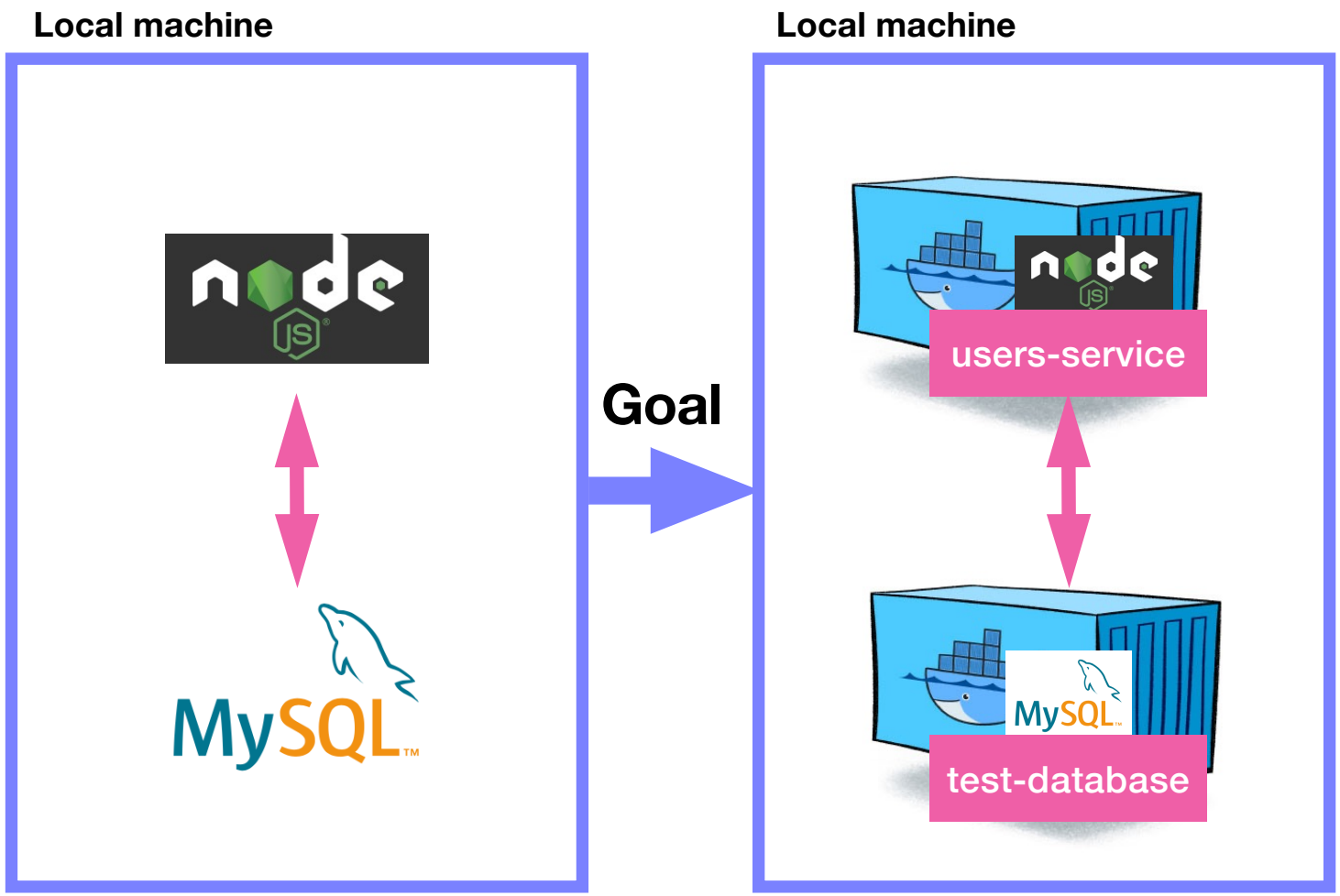
BREAK 2 MINUTES

Build microservices with Docker

If you are asked to write a service to do the directory mapping between email addresses and phone numbers, what will you do?



http://www.dwmkerr.com/learn-docker-by-building-a-microservice/



http://www.dwmkerr.com/learn-docker-by-building-a-microservice/

Go to docker hub to find mysql and show more details about its configuration and how to run it

https://hub.docker.com/_/mysql

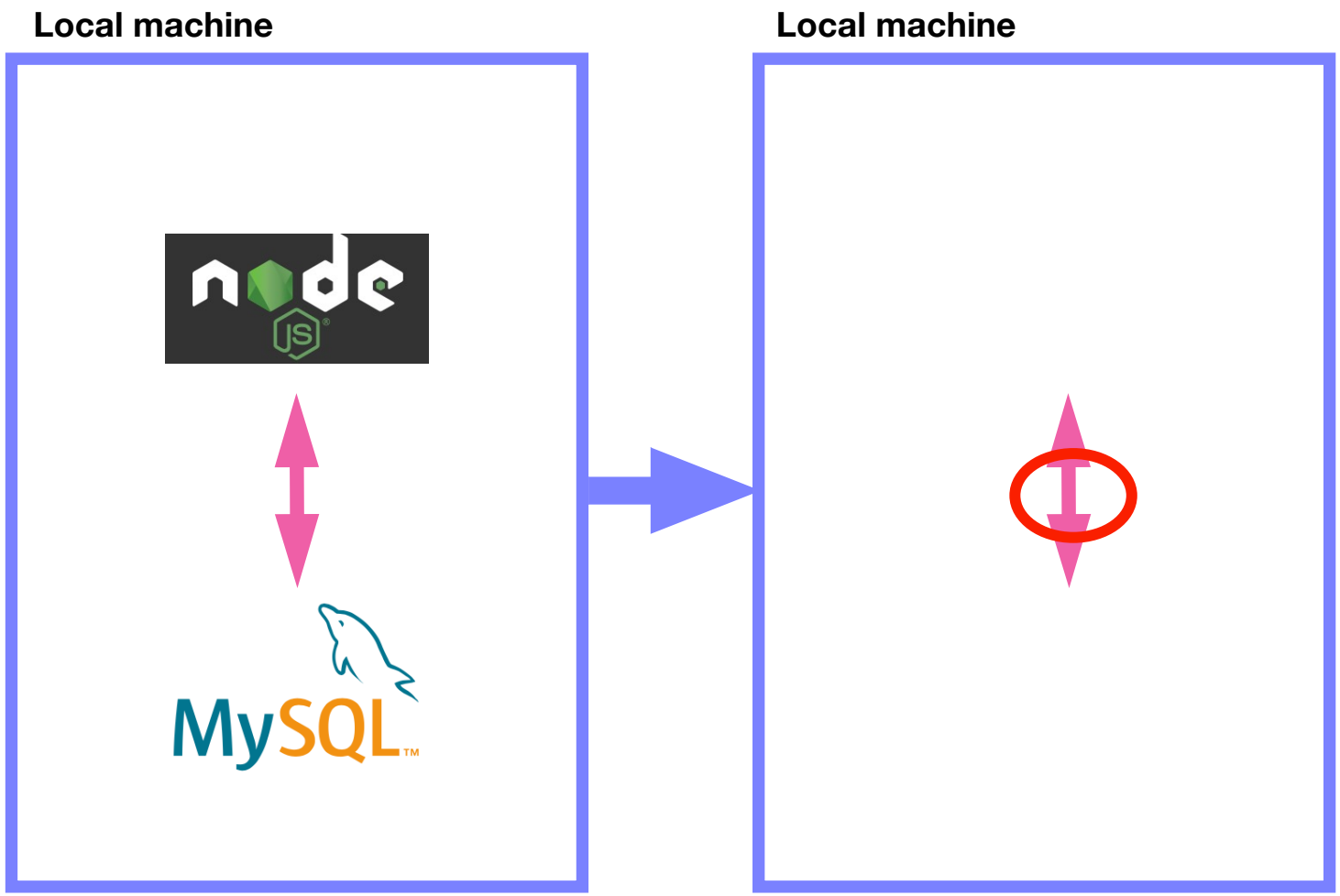
ลองทำอย่างแรก ใช้คำสั่งต่อไปนี้ เพื่อสร้างเครือข่ายเสมือน ขึ้นมาไว้สำหรับให้ containers คุยกัน

```
docker network create -d bridge my-net
```

```
docker network ls
```

```
docker network inspect my-net
```

https://docs.docker.com/engine/reference/commandline/network_create/



สร้างเครือข่ายเสมือนชื่อ my-net ให้ containers คุยกันเสร็จแล้ว

http://www.dwmkerr.com/learn-docker-by-building-a-microservice/

Build microservices with Docker

Step 1: Wrapping up the Test Database

- ✓ test-database\setup.sql
- ✓ test-database\Dockerfile
- ✓ test-database\docker-compose.yml
- ✓ test-database\stop.sh

DB server

docker

test-database/setup.sql

setup.sql

test-database > setup.sql

```
1  create table DIRECTORY (user_id INT NOT NULL AUTO_INCREMENT PRIMARY KEY, email TEXT, phon
2  insert into DIRECTORY (email, phone_number) values ('homer@thesimpsons.com', '+1 888 123
3  insert into DIRECTORY (email, phone_number) values ('marge@thesimpsons.com', '+1 888 123
4  insert into DIRECTORY (email, phone_number) values ('maggie@thesimpsons.com', '+1 888 123
5  insert into DIRECTORY (email, phone_number) values ('lisa@thesimpsons.com', '+1 888 123 1
6  insert into DIRECTORY (email, phone_number) values ('bart@thesimpsons.com', '+1 888 123 1
7
```

http://www.dwmkerr.com/learn-docker-by-building-a-microservice/

DB server

docker

test-database/Dockerfile

Dockerfile ●

test-database > Dockerfile > ...

```
1   FROM mysql:8
2
3   ENV MYSQL_ROOT_PASSWORD 123
4   ENV MYSQL_DATABASE users
5   ENV MYSQL_USER users_service
6   ENV MYSQL_PASSWORD 123
7
8   ADD setup.sql /docker-entrypoint-initdb.d
9
10  ENV TZ Asia/Bangkok
```

http://www.dwmkerr.com/learn-docker-by-building-a-microservice/

DB server

List docker images in our local machine

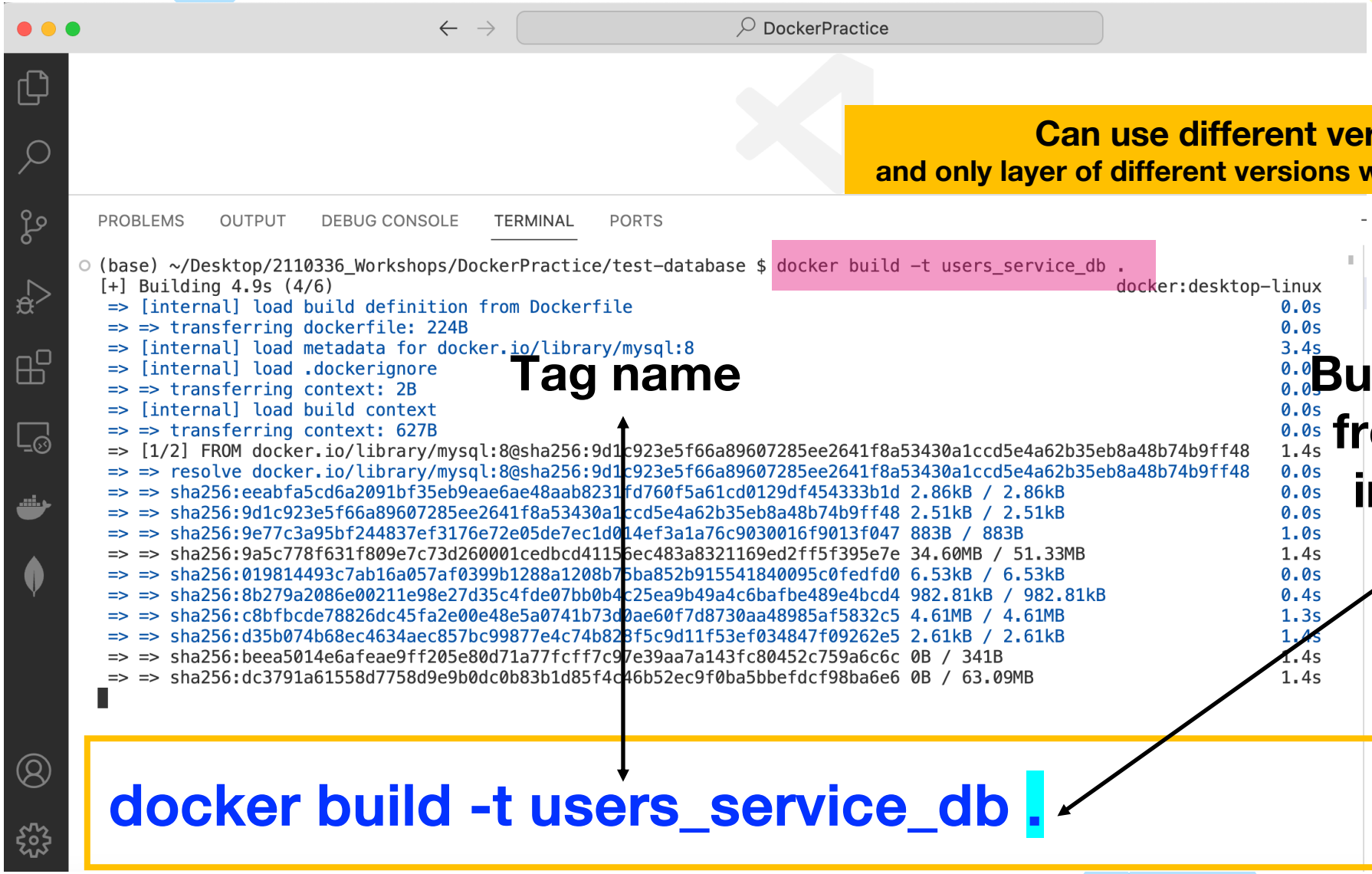
docker images

```
(base) ~/Desktop/2110336_Workshops/DockerPractice/test-database $ docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
```

docker system prune / docker system prune -a

DB server

Under the test-database folder call “docker build -t users_service_db .”



Can use different versions
and only layer of different versions will be downloaded

Tag name

Build docker image from a Dockerfile in this directory

```
docker build -t users_service_db .
```

DB server

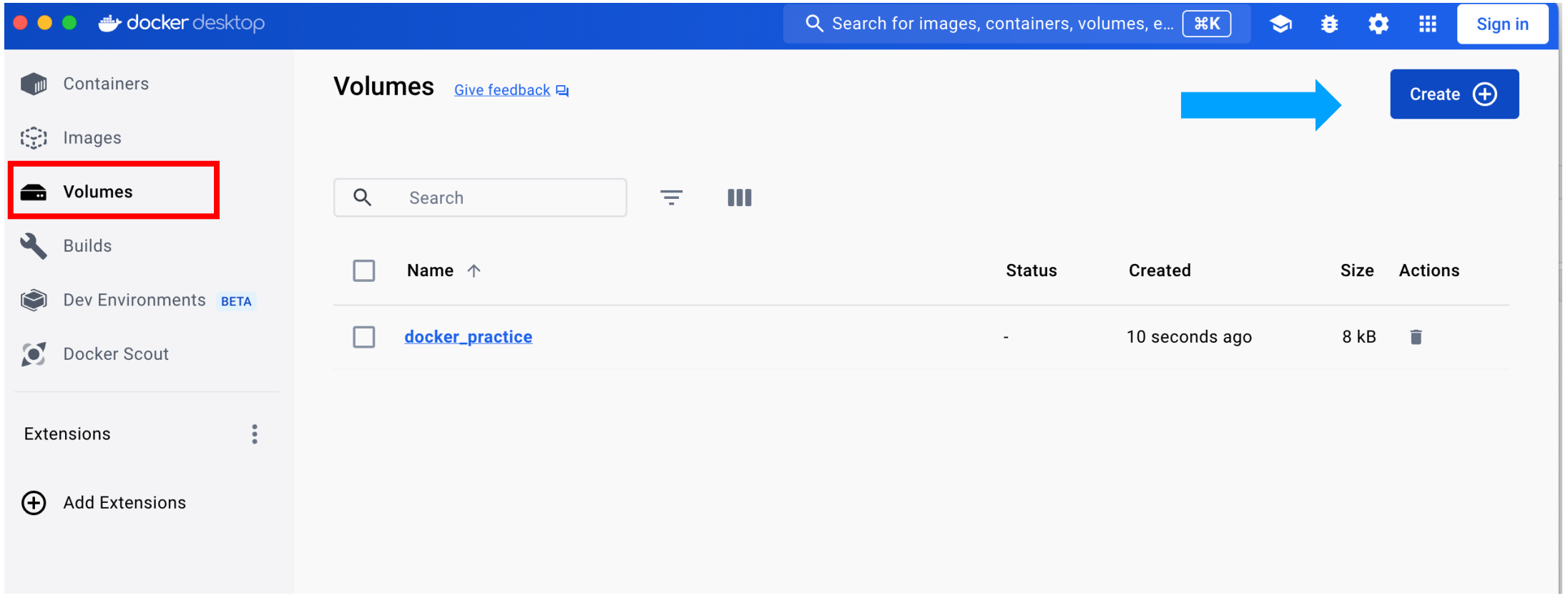
List docker images in our local machine

```
docker images
```

```
REPOSITORY          TAG       IMAGE ID       CREATED              SIZE
users_service_db    latest    4cd1649885e4   51 seconds ago       632MB
```

DB server

Create a volume via Docker Desktop



docker volume create [volume name]

DB server

Run the users_service_db via docker run

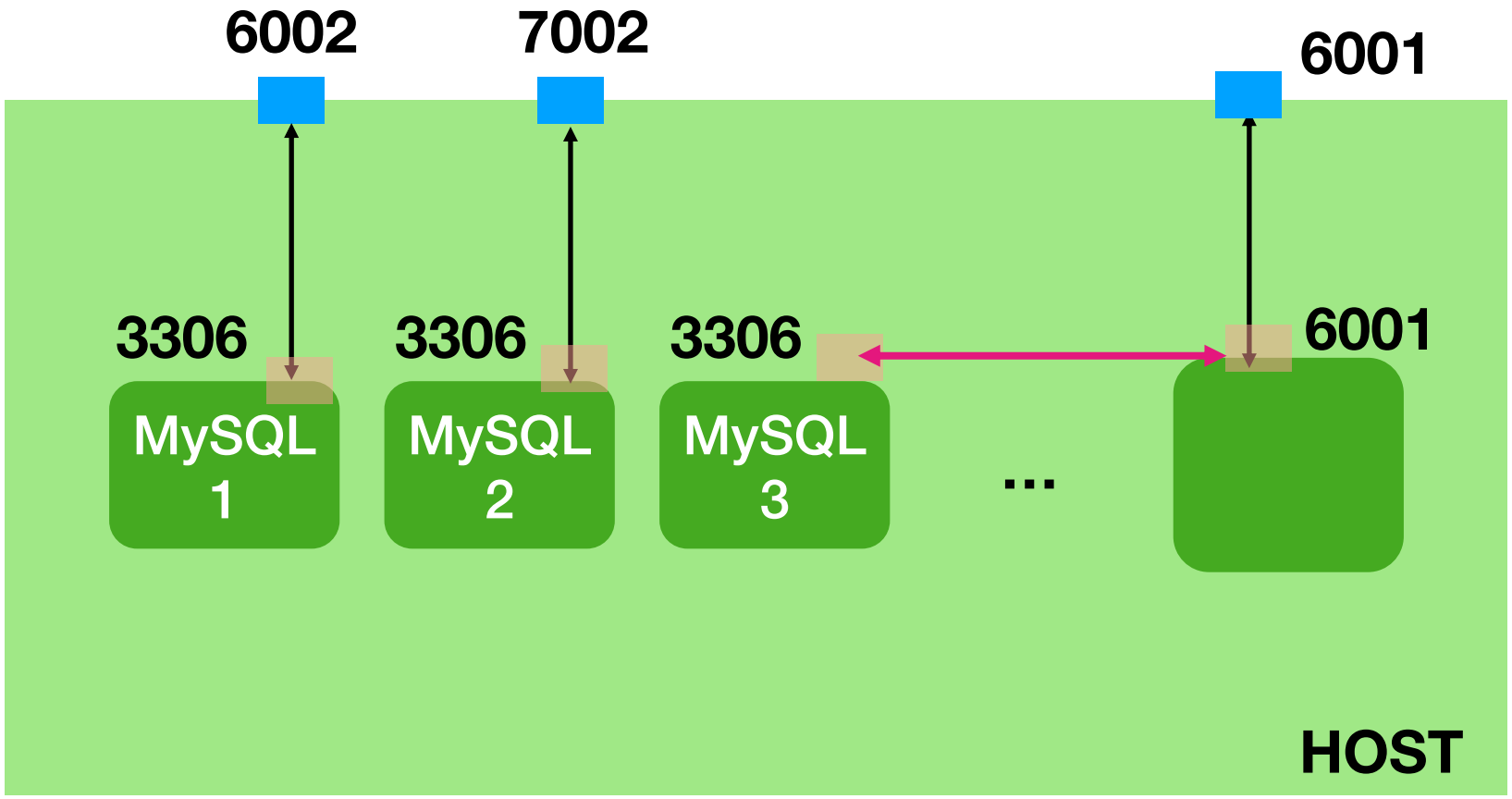
Named volume

- docker run -p 6002:3306 -v docker_practice:/var/lib/mysql --name db1 -it users_service_db

- docker run -p 7002:3306 -v share_data:/var/lib/mysql --name db2 -it users_service_db

We can run multiple instances of the same image with different port on the same machine

Port forwarding



Communicate between containers

DB server

Run the users_service_db via docker run

Named volume

- docker run -p 6002:3306 -v docker_practice:/var/lib/mysql --name db1 -it users_service_db

- docker run -p 6002:3306 -v Host volume /Users/duangdaowichadakul/Desktop/2110336_Workshops/DockerPractice/test-database/db1_data:var/lib/mysql --name db1 -it users_service_db

DB server

ลองใช้คำสั่งนี้ดูว่า db container ถูกสตาร์ทเรียบร้อยไหม

docker container ls /docker ps

```
CONTAINER ID   IMAGE              COMMAND                  CREATED         STATUS         PORTS
      NAMES
102fec4260fb   users_service_db   "docker-entrypoint.s…"   5 minutes ago   Up 5 minutes   33060/tcp, 0.0.0.0:6002->3306/
tcp   db1
```

http://www.dwmkerr.com/learn-docker-by-building-a-microservice/

DB server

docker

รันคำสั่งข้างล่างนี้เพื่อเชื่อมต่อเข้าไปที่ container db1 ที่สตาร์ทไว้

```
docker exec -it db1 /bin/bash
```

จากนั้น login เข้า MySQL โดยใช้ user: users_service และ
ใช้ password: 123 ตามที่กำหนดในไว้ Dockerfile ก่อนหน้า

```
(base) ~/Desktop/2110336_Workshops/DockerPractice $ docker exec -it db1 /bin/bash

bash-4.4# mysql -uusers_service -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.3.0 MySQL Community Server - GPL

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

mysql -u users_service -p

http://www.dwmkerr.com/learn-docker-by-building-a-microservice/

DB server

docker

```
mysql> use users
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+----------------+
| Tables_in_users |
+----------------+
| DIRECTORY      |
+----------------+
1 row in set (0.03 sec)

mysql> select * from directory;
ERROR 1146 (42S02): Table 'users.directory' doesn't exist
mysql> select * from DIRECTORY;
+---------+----------------------+-----------------+
| user_id | email                | phone_number    |
+---------+----------------------+-----------------+
|       1 | homer@thesimpsons.com  | +1 888 123 1111 |
|       2 | marge@thesimpsons.com  | +1 888 123 1112 |
|       3 | maggie@thesimpsons.com | +1 888 123 1113 |
|       4 | lisa@thesimpsons.com   | +1 888 123 1114 |
|       5 | bart@thesimpsons.com   | +1 888 123 1115 |
+---------+----------------------+-----------------+
5 rows in set (0.01 sec)
```

use users;

show tables;

SELECT * FROM DIRECTORY;

เราสามารถลอง connect ไปที่ mysql ใน container โดยใช้ port 6002 ก็ได้

mysql -h0.0.0.0 -P 6002 -u users_service -p123

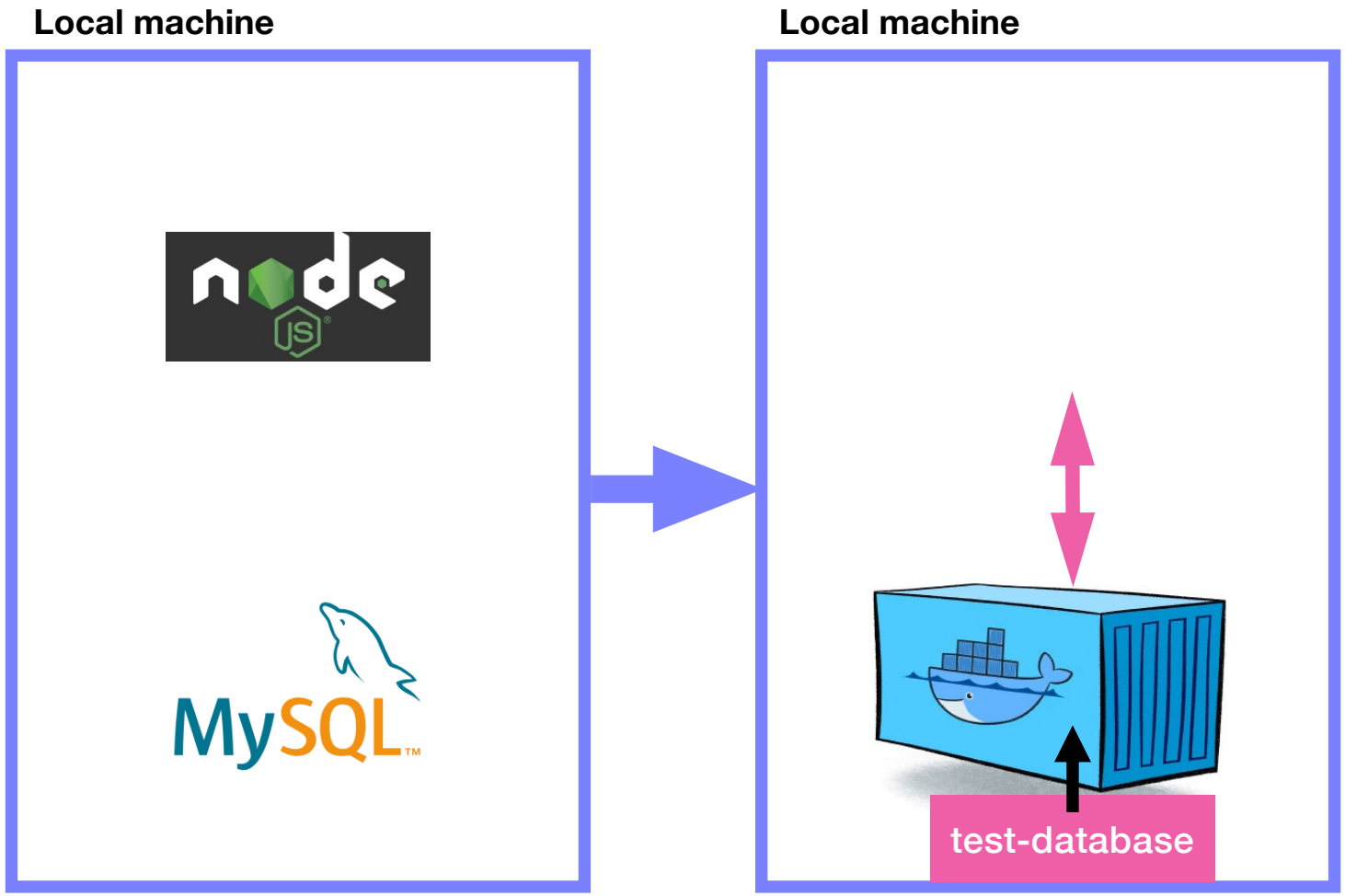
http://www.dwmkerr.com/learn-docker-by-building-a-microservice/

docker

test-database/stop.sh

```
1  #!/bin/sh
2
3  # Stop the db and remove the container.
4  docker stop db && docker rm db
```

http://www.dwmkerr.com/learn-docker-by-building-a-microservice/



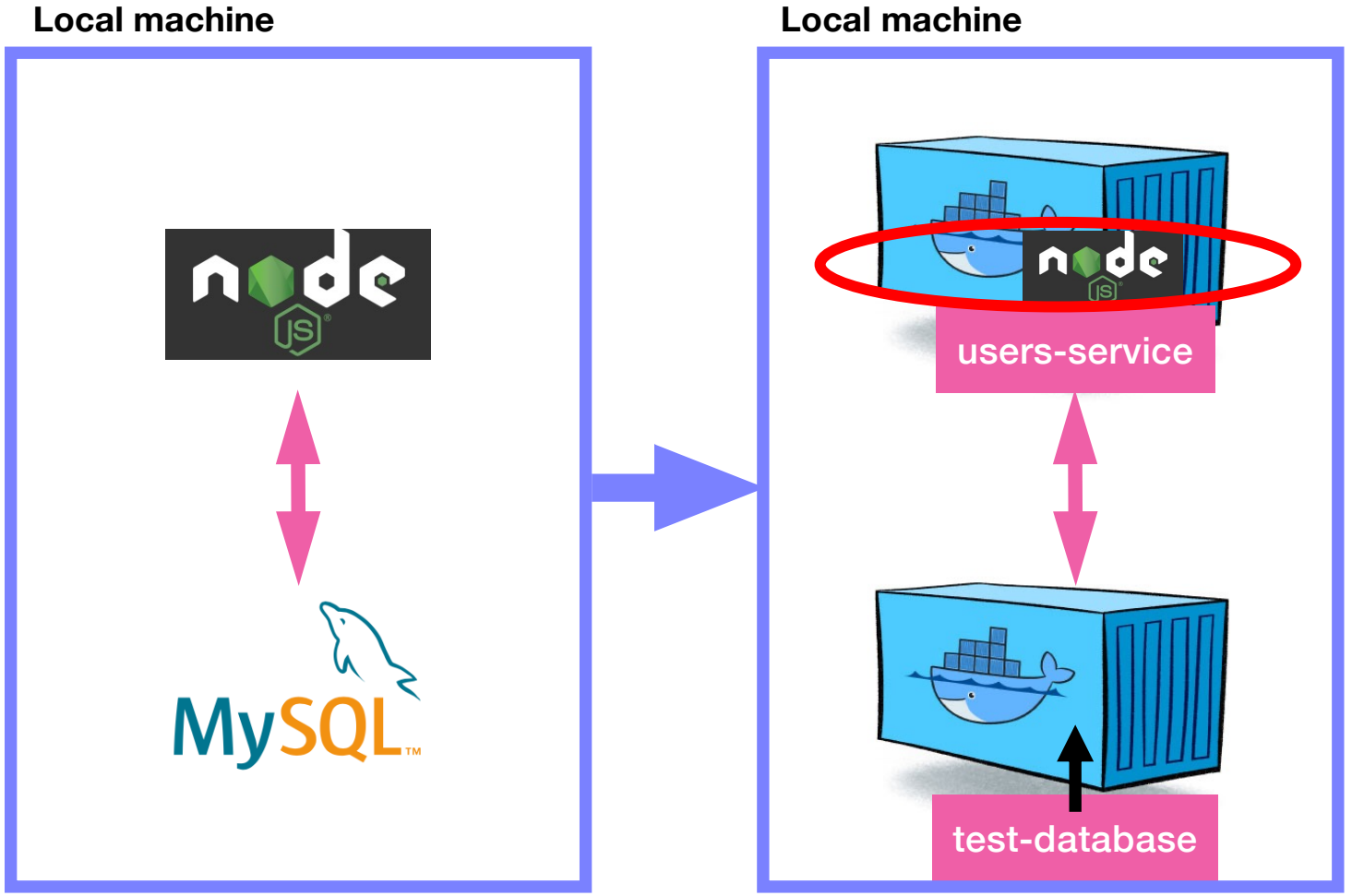
สร้าง container ชื่อ db โดยต่อกับเครือข่ายเสมือนชื่อ my-net เสร็จแล้ว

http://www.dwmkerr.com/learn-docker-by-building-a-microservice/

BREAK 2 MINUTES

Build microservices with Docker

Step 2: Creating a microservice in Node.js



http://www.dwmkerr.com/learn-docker-by-building-a-microservice/

Build microservices with Docker

Step 2: Creating a microservice in Node.js

```
test-database/          # contains the code seen in Step 1
users-service/          # root of our node.js microservice
- package.json          # dependencies, metadata
- index.js              # main entrypoint of the app
- api/                  # our apis and api tests
- config/               # config for the app
- repository/           # abstraction over our db
- server/               # server setup code
```

http://www.dwmkerr.com/learn-docker-by-building-a-microservice/

Build a microservice with Docker

Step 2: Creating a microservice in Node.js

```
test-database/          # contains the code seen in Step 1
users-service/          # root of our node.js microservice
- package.json          # dependencies, metadata
- index.js              # main entrypoint of the app
- api/                  # our apis and api tests
- config/               # config for the app
- repository/           # abstraction over our db
- server/               # server setup code
```

http://www.dwmkerr.com/learn-docker-by-building-a-microservice/

Microservice

docker

users-service/repository/repository.js

users-service > repository > JS repository.js > Repository

```
1   //  repository.js
2   //
3   //  Exposes a single function - 'connect', which returns
4   //  a connected repository. Call 'disconnect' on this object when you're done.
5   'use strict';
6
7   var mysql = require('mysql2');
8
9   //  Class which holds an open connection to a repository
10  //  and exposes some simple functions for accessing data.
11  class Repository {
12    constructor(connectionSettings) {
13      this.connectionSettings = connectionSettings;
14      this.connection = mysql.createConnection(this.connectionSettings);
15    }
```

Logic work/database access
w/ internal
getUsers()
getUserByEmail(email)

repository.js เก็บ methods ที่ใช้ในการ connect และ query ข้อมูลที่ต้องการ

http://www.dwmkerr.com/learn-docker-by-building-a-microservice/

Microservice

users-service/repository/repository.js

users-service > repository > JS repository.js > Repository > getUsers

```
11  class Repository {
17    getUsers() {
18      return new Promise((resolve, reject) => {
19
20        this.connection.query('SELECT email, phone_number FROM DIRECTORY', (err, results) => {
21          if(err) {
22            return reject(new Error('An error occured getting the users: ' + err));
23          }
24
25          resolve((results || []).map((user) => {
26            return {
27              email: user.email,
28              phone_number: user.phone_number
29            };
30          }));
31        });
32
33      });
34    }
```

getUsers()

Query to get all users…

http://www.dwmkerr.com/learn-docker-by-building-a-microservice/

Microservice

users-service/repository/repository.js

users-service > repository > JS repository.js > Repository

```
11  class Repository {
36    getUserByEmail(email) {
38      return new Promise((resolve, reject) => {
39
40        //  Fetch the customer.
41        this.connection.query('SELECT email, phone_number FROM DIRECTORY WHERE email = ?',
42        [email], (err, results) => {
43          if(err) {
44            return reject(new Error('An error occured getting the user: ' + err));
45          }
46          if(results.length === 0) {
47            resolve(undefined);
48          } else {
49            resolve({
50              email: results[0].email,
51              phone_number: results[0].phone_number
52            });
53          }
54        });
55      });
56    }
```

getUserByEmail(email)

http://www.dwmkerr.com/learn-docker-by-building-a-microservice/

Microservice

docker

users-service/repository/repository.js

users-service > repository > JS repository.js > Repository

```
64  module.exports.connect = (connectionSettings) => {
65    return new Promise((resolve, reject) => {
66      if(!connectionSettings.host) throw new Error("A host must be specified.");
67      if(!connectionSettings.user) throw new Error("A user must be specified.");
68      if(!connectionSettings.password) throw new Error("A password must be specified.");
69      if(!connectionSettings.port) throw new Error("A port must be specified.");
70
71      resolve(new Repository(connectionSettings));
72    });
73  };
```

Only one exported function

http://www.dwmkerr.com/learn-docker-by-building-a-microservice/

Build a microservice with Docker

Step 2: Creating a microservice in Node.js

```
test-database/          # contains the code seen in Step 1
users-service/          # root of our node.js microservice
- package.json          # dependencies, metadata
- index.js              # main entrypoint of the app
- api/                  # our apis and api tests
- config/               # config for the app
- repository/           # abstraction over our db
- server/               # server setup code
```

http://www.dwmkerr.com/learn-docker-by-building-a-microservice/

Microservice

docker

users-service/server/server.js

users-service > server > JS server.js > ...

```
6   module.exports.start = (options) => {
7
8     return new Promise((resolve, reject) => {
9
10      //  Make sure we have a repository and port provided.
11      if(!options.repository) throw new Error("A server must be started with a connected reposi
12      if(!options.port) throw new Error("A server must be started with a port.");
13
14      //  Create the app, add some logging.
15      var app = express();
16      app.use(morgan('dev'));
17
18      //  Add the APIs to the app.
19      require('../api/users')(app, options);
20
21      //  Start the app, creating a running server which we return.
22      var server = app.listen(options.port, () => {
23        resolve(server);
24      });
25
26    });
27  };
```

http://www.dwmkerr.com/learn-docker-by-building-a-microservice/

Build a microservice with Docker

Step 2: Creating a microservice in Node.js

```
test-database/          # contains the code seen in Step 1
users-service/          # root of our node.js microservice
- package.json          # dependencies, metadata
- index.js              # main entrypoint of the app
- api/                  # our apis and api tests
- config/               # config for the app
- repository/           # abstraction over our db
- server/               # server setup code
```

http://www.dwmkerr.com/learn-docker-by-building-a-microservice/

docker

users-service/api/user.js

```
10  app.get('/users', (req, res, next) => {
11    options.repository.getUsers().then((users) => {
12      res.status(200).send(users.map((user) => { return {
13          email: user.email,
14          phoneNumber: user.phone_number
15        };
16      }));
17    })
18    .catch(next);
19  });
```

Define the users api

Microservice

docker

users-service/api/user.js

```
21  app.get('/search', (req, res, next) => {
22
23    //  Get the email.
24    var email = req.query.email;
25    if (!email) {
26      throw new Error("When searching for a user, the email must be specified, e.g: '/search?
27    }
28
29    //  Get the user from the repo.
30    options.repository.getUserByEmail(email).then((user) => {
31
32      if(!user) {
33        res.status(404).send('User not found.');
34      } else {
35        res.status(200).send({
36          email: user.email,
37          phoneNumber: user.phone_number
38        });
39      }
40    })
41    .catch(next);
```

Define the users api

docker

Build a microservice with Docker

Step 2: Creating a microservice in Node.js

```
test-database/          # contains the code seen in Step 1
users-service/          # root of our node.js microservice
- package.json          # dependencies, metadata
- index.js              # main entrypoint of the app
- api/                  # our apis and api tests
- config/               # config for the app
- repository/           # abstraction over our db
- server/               # server setup code
```

http://www.dwmkerr.com/learn-docker-by-building-a-microservice/

users-service/index.js


```
5   var server = require('./server/server');
6   var repository = require('./repository/repository');
7   var config = require('./config/config');

21  repository.connect({
22    host: config.db.host,
23    database: config.db.database,
24    user: config.db.user,
25    password: config.db.password,
26    port: config.db.port
27  }).then((repo) => {
28    console.log("Connected. Starting server...");
29
30    return server.start({
31      port: config.port,
32      repository: repo
33    });
```

Entry point of the application

http://www.dwmkerr.com/learn-docker-by-building-a-microservice/

docker

Build a microservice with Docker

Step 2: Creating a microservice in Node.js

```
test-database/          # contains the code seen in Step 1
users-service/          # root of our node.js microservice
- package.json          # dependencies, metadata
- index.js              # main entrypoint of the app
- api/                  # our apis and api tests
- config/               # config for the app
- repository/           # abstraction over our db
- server/               # server setup code
```

http://www.dwmkerr.com/learn-docker-by-building-a-microservice/

Microservice

docker

users-service > config > JS config.js > ...

```
1   //  config.js
2   //
3   //  Simple application configuration. Extend as needed.
4   module.exports = {
5     port: process.env.PORT || 6006, // user service port
6     db: {
7       host: process.env.DATABASE_HOST || '127.0.0.1',
8       database: 'users',
9       user: 'users_service',
10      password: '123',
11      port: 6002   // local host port which is mapped to mysql container port
12    }
13  };
```

http://www.dwmkerr.com/learn-docker-by-building-a-microservice/

Microservice

Try running users_service which is not containerized

- **npm install**
- **npm start**

Then open the browser with http://localhost:6001/users

Try
http://localhost:6001/users

ผลที่ได้

```
// 20200313203351
// http://localhost:6001/users

[
  {
    "email": "homer@thesimpsons.com",
    "phoneNumber": "+1 888 123 1111"
  },
  {
    "email": "marge@thesimpsons.com",
    "phoneNumber": "+1 888 123 1112"
  },
  {
    "email": "maggie@thesimpsons.com",
    "phoneNumber": "+1 888 123 1113"
  },
  {
    "email": "lisa@thesimpsons.com",
    "phoneNumber": "+1 888 123 1114"
  },
  {
    "email": "bart@thesimpsons.com",
    "phoneNumber": "+1 888 123 1115"
  }
]
```

Try
http://localhost:6001/search?email=homer@thesimpsons.com

ผลที่ได้

```
// 20200313203245
// http://localhost:6001/search?email=homer@thesimpsons.com

{
  "email": "homer@thesimpsons.com",
  "phoneNumber": "+1 888 123 1111"
}
```

คราวนี้มาเริ่มทำให้ user service มาเป็นอีก container ด้วย

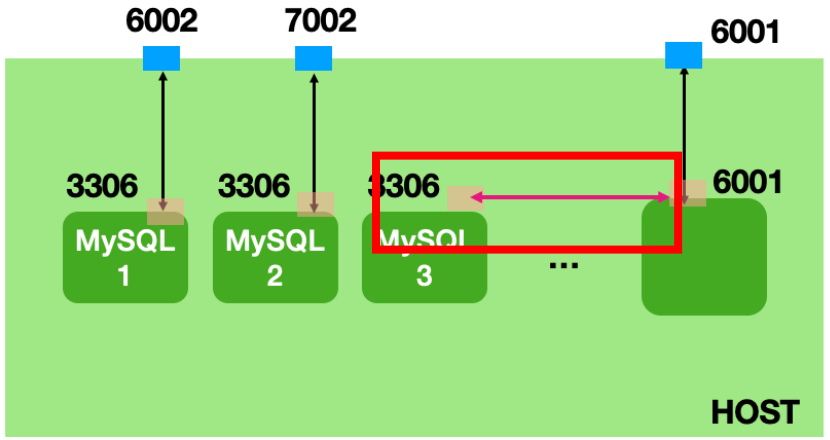
อย่างแรกให้แก่ config/config.js โดยเปลี่ยน port ของ db เป็นcontainer port 3306

Microservice

docker

users-service > config > JS config.js > ...

```
1   //  config.js
2   //
3   //  Simple application configuration. Extend as needed.
4   module.exports = {
5     port: process.env.PORT || 6001, // user service port
6     db: {
7       host: process.env.DATABASE_HOST || '127.0.0.1',
8       database: 'users',
9       user: 'users_service',
10      password: '123',
11      port: 3306   // local host port which is mapped to mysql container port
12    }
13  };
```



Communicate between containers

http://www.dwmkerr.com/learn-docker-by-building-a-microservice/

Create a Dockerfile

users-service/Dockerfile

users-service > Dockerfile > ...

```
1   # Use Node 12.16 LTS as the base image.
2   FROM node:16.14
3
4   # Add everything in the current directory to our image, in the 'app' folder.
5   COPY . /app
6
7   # Install dependencies
8   RUN cd /app; \
9       npm install --production
10
11  # Expose our server port.
12  EXPOSE 6001
13
14  # Run our app.
15  CMD ["node", "/app/index.js"]
```

Layer 1

Layer 2

Layer 3

...

```
docker build -t users-service .
```

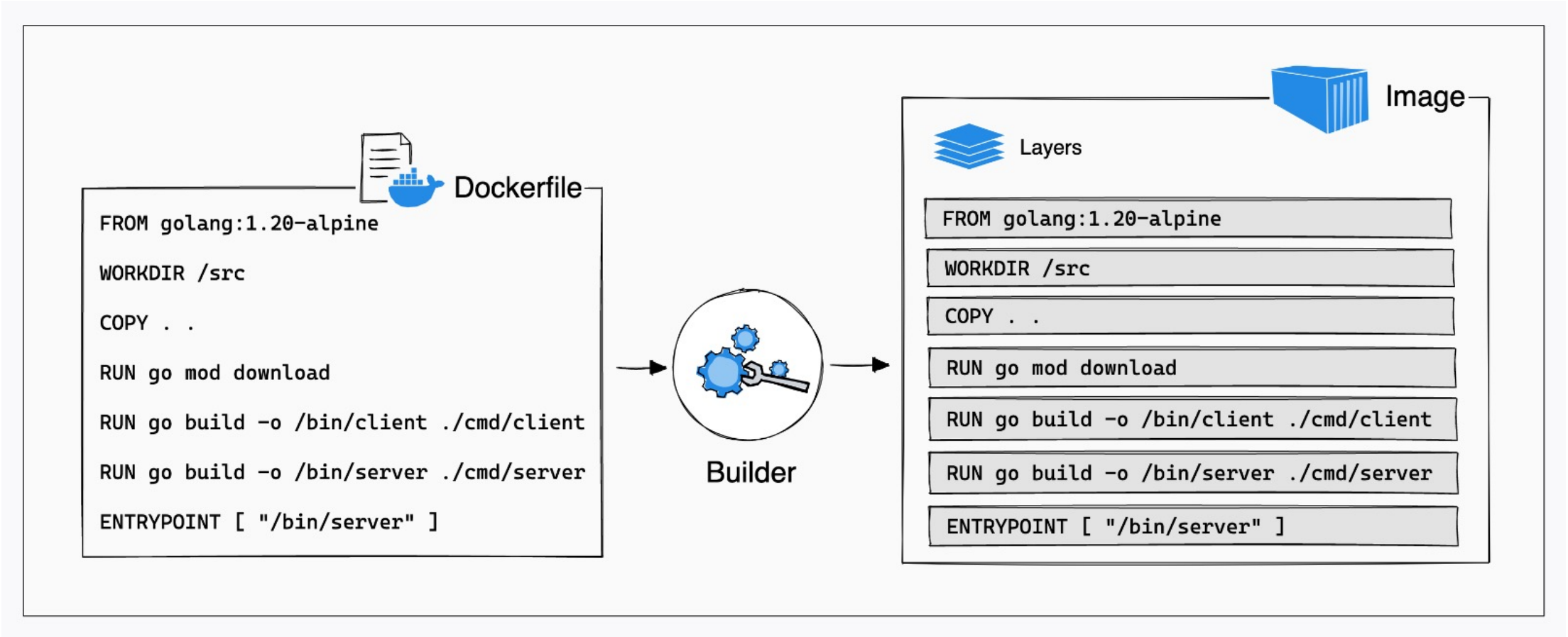
สร้าง docker image ของ user service

create a new image

find the Dockerfile from the current directory

http://www.dwmkerr.com/learn-docker-by-building-a-microservice/

Image layers generated from Dockerfile



https://docs.docker.com/build/guide/layers/

Cached layers

Any files changed in the project will invalidate since the COPY layer

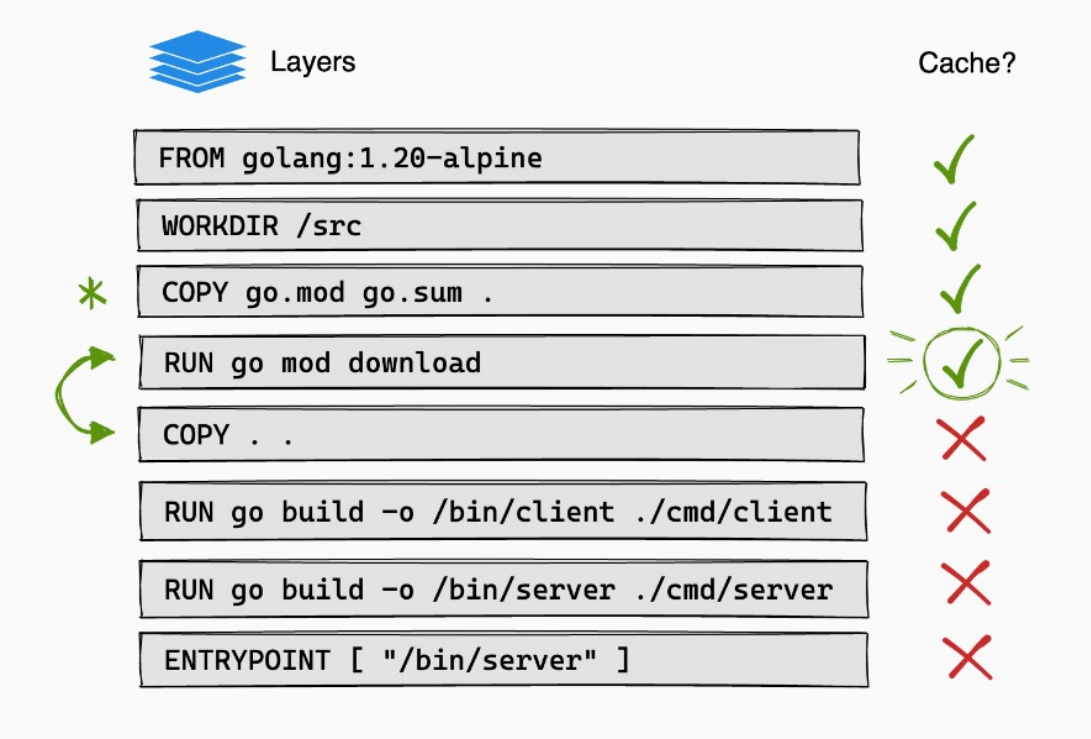
https://docs.docker.com/build/guide/layers/

How to improve the Dockerfile?

Reorder the statement in Dockerfile

```
  # syntax=docker/dockerfile:1
  FROM golang:1.21-alpine
  WORKDIR /src
- COPY . .
+ COPY go.mod go.sum .
  RUN go mod download
+ COPY . .
  RUN go build -o /bin/client ./cmd/client
  RUN go build -o /bin/server ./cmd/server
  ENTRYPOINT [ "/bin/server" ]
```

More layers have been cached and reused



https://docs.docker.com/build/guide/layers/

More about image layers

go to Docker desktop and see Docker scout

Build a docker image

docker build -t users-service .

```
(base) ~/Desktop/2110336_Workshops/DockerPractice/users-service $ docker build -
t users-service .

[+] Building 20.3s (8/8) FINISHED                                docker:desktop-linux
 => [internal] load build definition from Dockerfile                               0.0s
 => => transferring dockerfile: 338B                                               0.0s
 => [internal] load metadata for docker.io/library/node:16.14                      4.2s
 => [internal] load .dockerignore                                                  0.0s
 => => transferring context: 2B                                                    0.0s
 => [internal] load build context                                                  0.3s
 => => transferring context: 14.29MB                                               0.3s
 => [1/3] FROM docker.io/library/node:16.14@sha256:6e54786b2ad01667d4652          14.1s
 => => resolve docker.io/library/node:16.14@sha256:6e54786b2ad01667d46524          0.0s
 => => sha256:a76eec769c30e46d15a58a7d2f7f9253b9db0984e49 7.65kB / 7.65kB          0.0s
 => => sha256:85bed84afb9a834cf090b55d2e584abd55b4792d9 50.44MB / 50.44MB          2.2s
 => => sha256:6e54786b2ad01667d46524e82806298714f50d2be72 1.21kB / 1.21kB          0.0s
 => => sha256:5fdd409f4b2bf3771fac1cbc2df46d1cbb4b800b154 7.86MB / 7.86MB          1.7s
 => => sha256:fa3069e6cecf9d5e43ace0a925b53849b2fd189ef 10.00MB / 10.00MB          2.5s
 => => sha256:2c7096c53d46a266fa5b36de89a5070cd9e8791b799 2.21kB / 2.21kB          0.0s
 => => sha256:4ee16f45eff97cbe98b74c63ca2375b2c6b2592cd 51.84MB / 51.84MB          3.5s
 => => extracting sha256:85bed84afb9a834cf090b55d2e584abd55b4792d93b750db          1.7s
 => => sha256:2cee68386155951f51cf2088092ed4b624b2ba9 192.64MB / 192.64MB          9.1s
 => => sha256:0aa7befa4a1b0bc4aea77c1ccd87bd5b3c94ad4ff4f 4.20kB / 4.20kB          3.2s
 => => sha256:c71ce35a137fd1d07f18d5f318a78351d738cd2b1 33.89MB / 33.89MB          5.0s
```

Run the docker image into a container

```
docker images
```

รันคำสั่งข้างล่างนี้เพื่อสตาร์ท users-service container โดยให้เชื่อมต่อกับเครือข่าย my-net และเปิดพอร์ตสาธารณะเบอร์ 6001

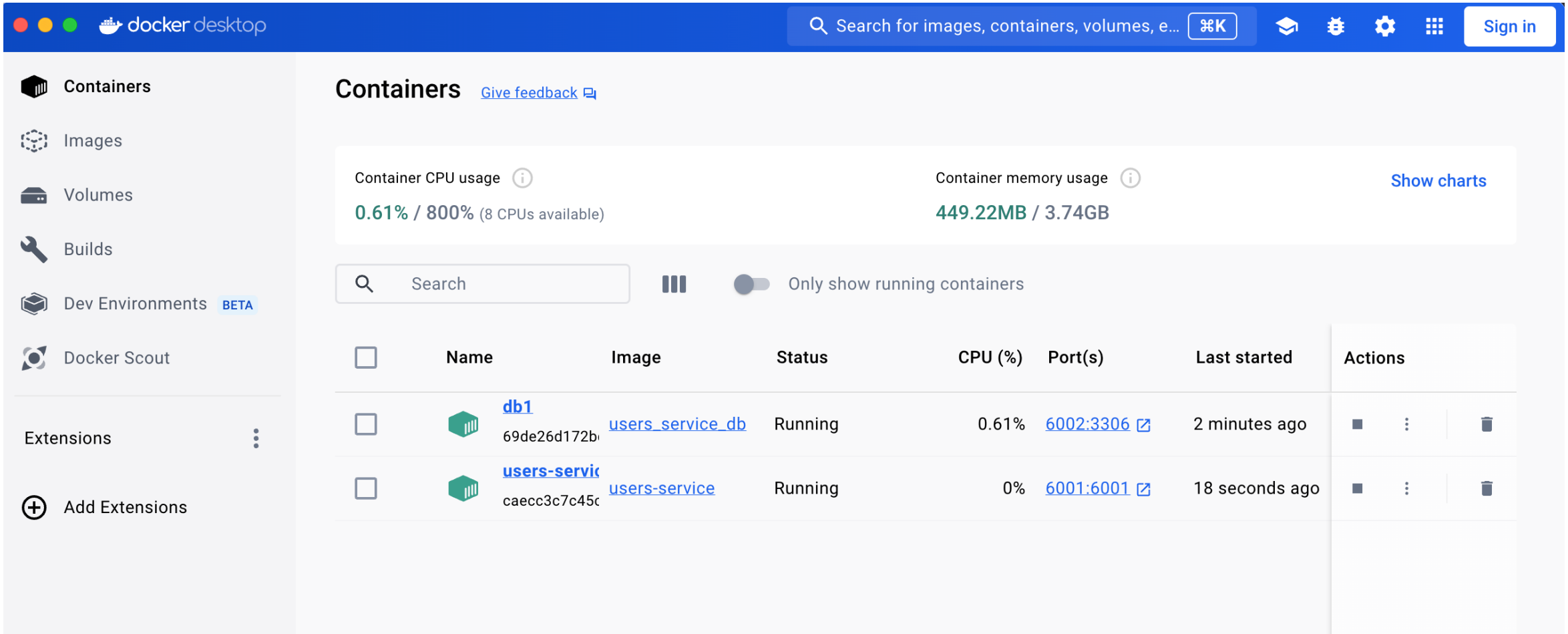
```
docker run -it --net=my-net -p 6001:6001
--name=users-service users-service
```

http://www.dwmkerr.com/learn-docker-by-building-a-microservice/

Run the docker image into a container

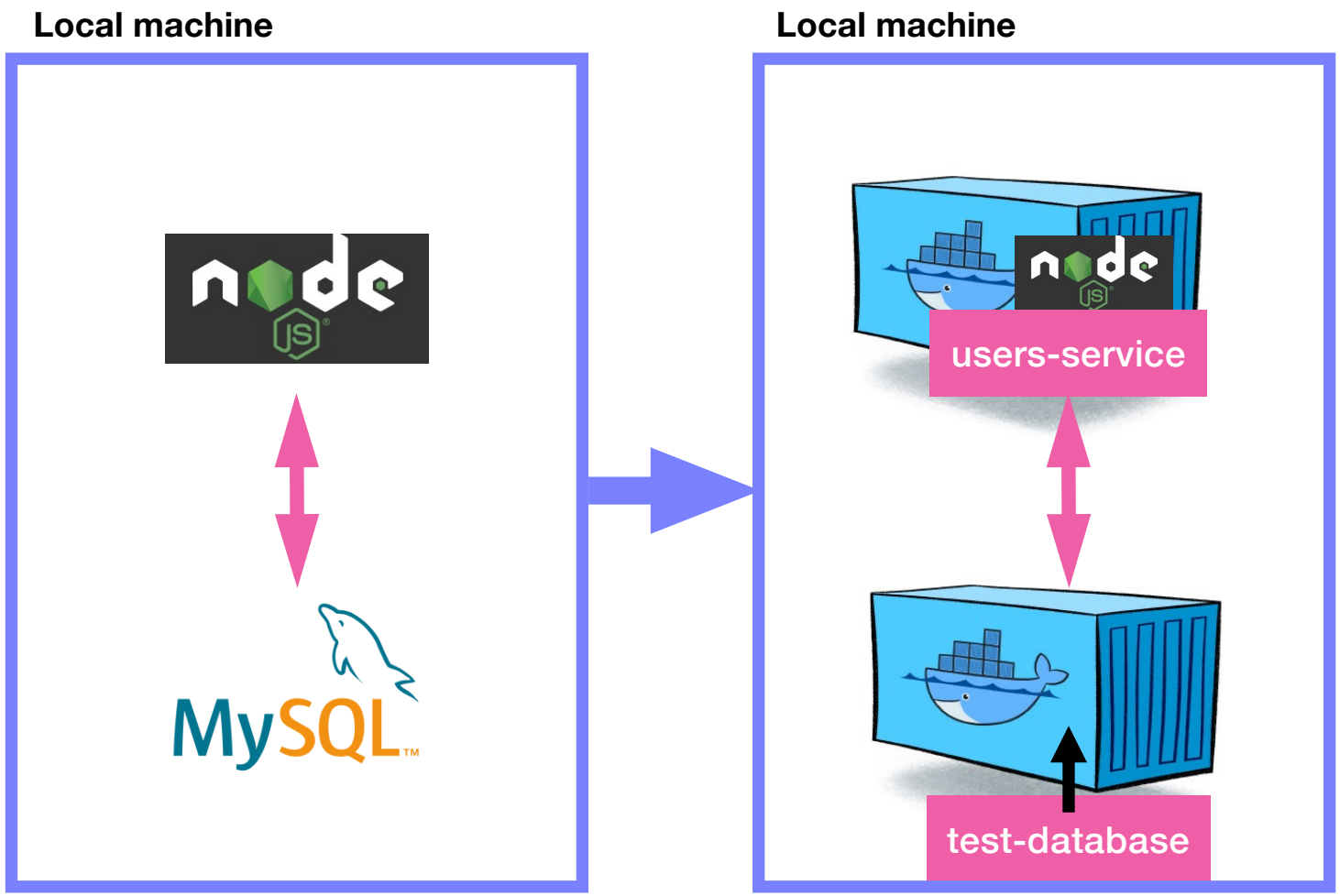
```
docker run -it --net=my-net -p 6001:6001
--name=users-service users-service
```

```
--- Customer Service---
Connecting to customer repository...
Connected. Starting server...
Server started successfully, running on port 6001.
Unhandled Exception Error: connect ECONNREFUSED 127.0.0.1:3306
    at TCPConnectWrap.afterConnect [as oncomplete] (node:net:1157:16) {
  errno: -111,
  code: 'ECONNREFUSED',
  syscall: 'connect',
  address: '127.0.0.1',
  port: 3306,
  fatal: true
}
Unhandled Exception Error: Connection lost: The server closed the connection.
    at Socket.<anonymous> (/app/node_modules/mysql2/lib/connection.js:117:31)
    at Socket.emit (node:events:526:28)
    at TCP.<anonymous> (node:net:687:12) {
  fatal: true,
  code: 'PROTOCOL_CONNECTION_LOST'
}
```



มี container ของ users_service ขึ้นแล้ว แต่เชื่อมต่อไปยัง db1 ไม่ได้

Dockerising

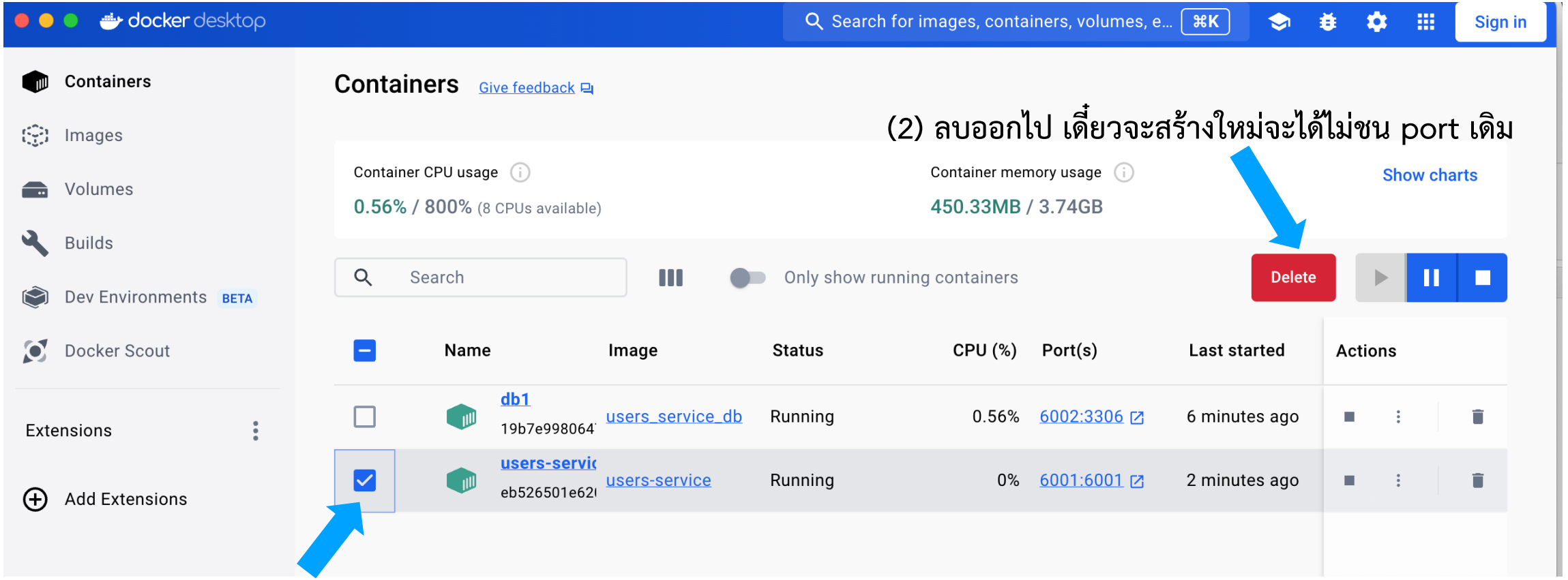


docker ps -a

```
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
                    NAMES
31c1be801bdd        users_service       "docker-entrypoint.s…"   31 seconds ago      Up 31 seconds       0.0.0.0:6001->
8123/tcp            users-service
ff64350a3440        db                  "docker-entrypoint.s…"   21 minutes ago      Up 21 minutes       33060/tcp, 0.0
.0.0:6002->3306/tcp   db
```

Both containers are there, so what is going on??

http://www.dwmkerr.com/learn-docker-by-building-a-microservice/



(1) เลือก users-service container

ลองดูด้วยว่า db1 container ต่อกับ my-net หรือยัง

docker network connect my-net db1

Link containers

#1 An “old fashioned” with “link” parameter

link the containers named db1 and refer to it as db1

```
docker run -it -p 6001:6001 --link db1:db1 -e DATABASE_HOST=db1 users-service
```

name of the image to run in our container

set the DATABASE_HOST environment variable to db1 … connecting to a host named db1…

http://www.dwmkerr.com/learn-docker-by-building-a-microservice/

```
docker run -it -p 6001:6001 --link db1:db1 -e DATABASE_HOST=db1
users-service
```

```
--- Customer Service---
Connecting to customer repository...
Connected. Starting server...
Server started successfully, running on port 6001.
```

http://www.dwmkerr.com/learn-docker-by-building-a-microservice/

Try
http://localhost:6001/users

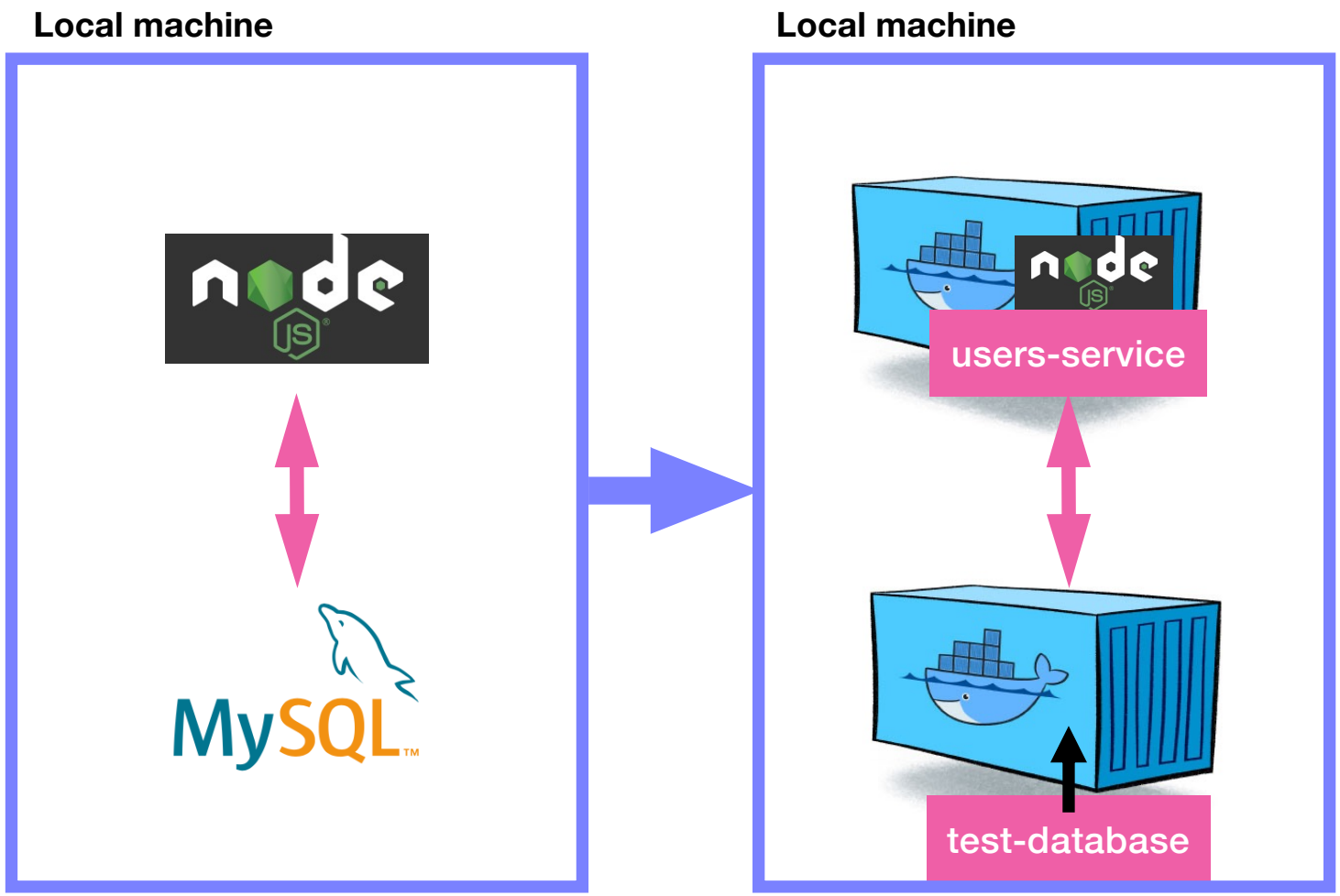
ผลที่ได้

```
// 20200313203351
// http://localhost:6001/users

[
  {
    "email": "homer@thesimpsons.com",
    "phoneNumber": "+1 888 123 1111"
  },
  {
    "email": "marge@thesimpsons.com",
    "phoneNumber": "+1 888 123 1112"
  },
  {
    "email": "maggie@thesimpsons.com",
    "phoneNumber": "+1 888 123 1113"
  },
  {
    "email": "lisa@thesimpsons.com",
    "phoneNumber": "+1 888 123 1114"
  },
  {
    "email": "bart@thesimpsons.com",
    "phoneNumber": "+1 888 123 1115"
  }
]
```

Try
http://localhost:6001/search?email=homer@thesimpsons.com

ผลที่ได้

```
// 20200313203245
// http://localhost:6001/search?email=homer@thesimpsons.com

{
  "email": "homer@thesimpsons.com",
  "phoneNumber": "+1 888 123 1111"
}
```


Local machine
Local machine
node
MySQL
node
users-service
test-database

Try not using --link

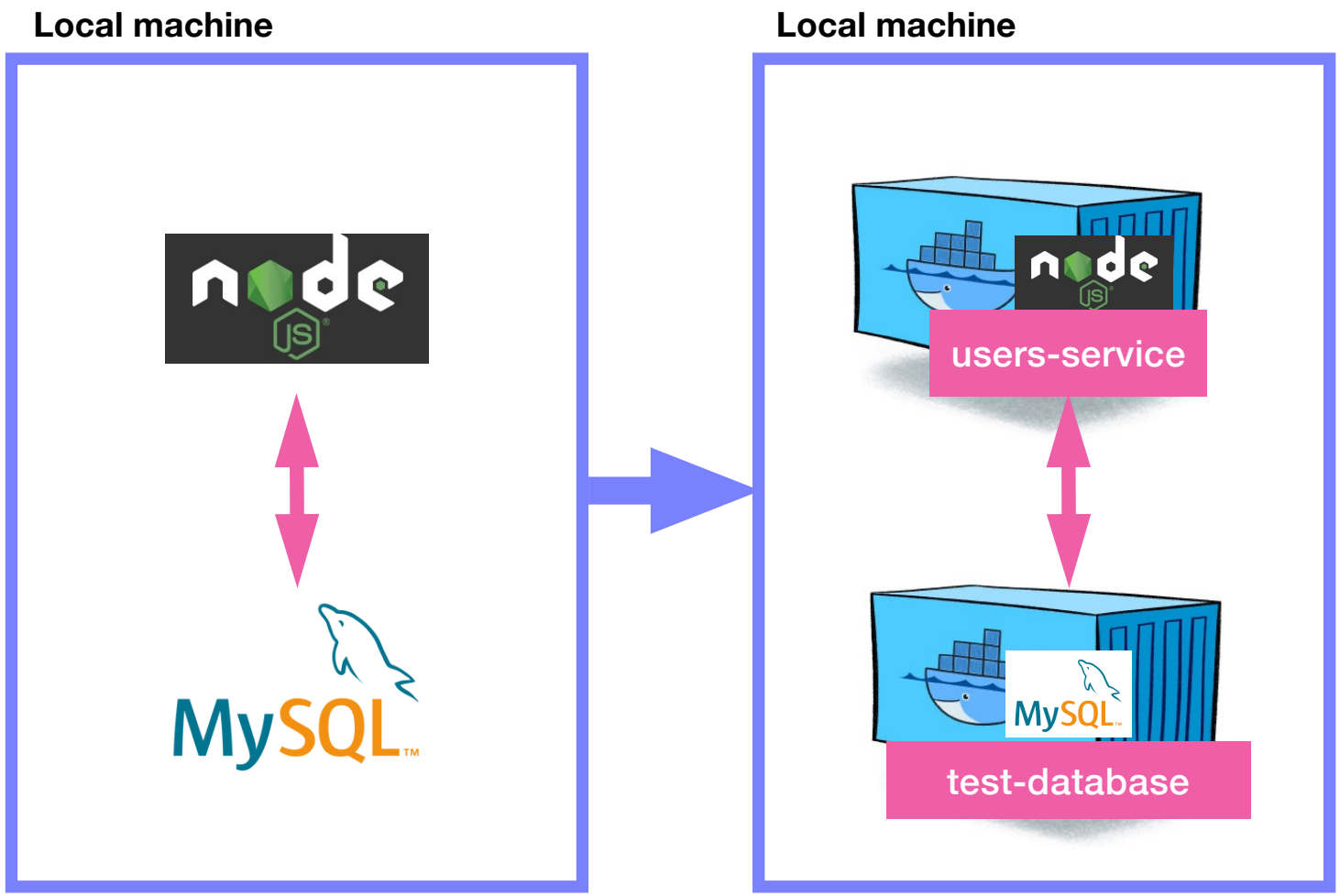
- Delete all db1 and users-service containers
- Start the new db1 containers using the following command:

```
docker run -p 6002:3306 -v docker_practice:/var/lib/mysql --name
db1 -h host.docker.internal -it users_service_db
```

- Start the users_service container using the following command:

```
docker run -it --net=my-net -p 6001:6001 -e
DATABASE_HOST=host.docker.internal --name=users-service users-
service
```

Then, it should work without --link.


Local machine
Local machine
node
MySQL
node
users-service
MySQL
test-database

Composing

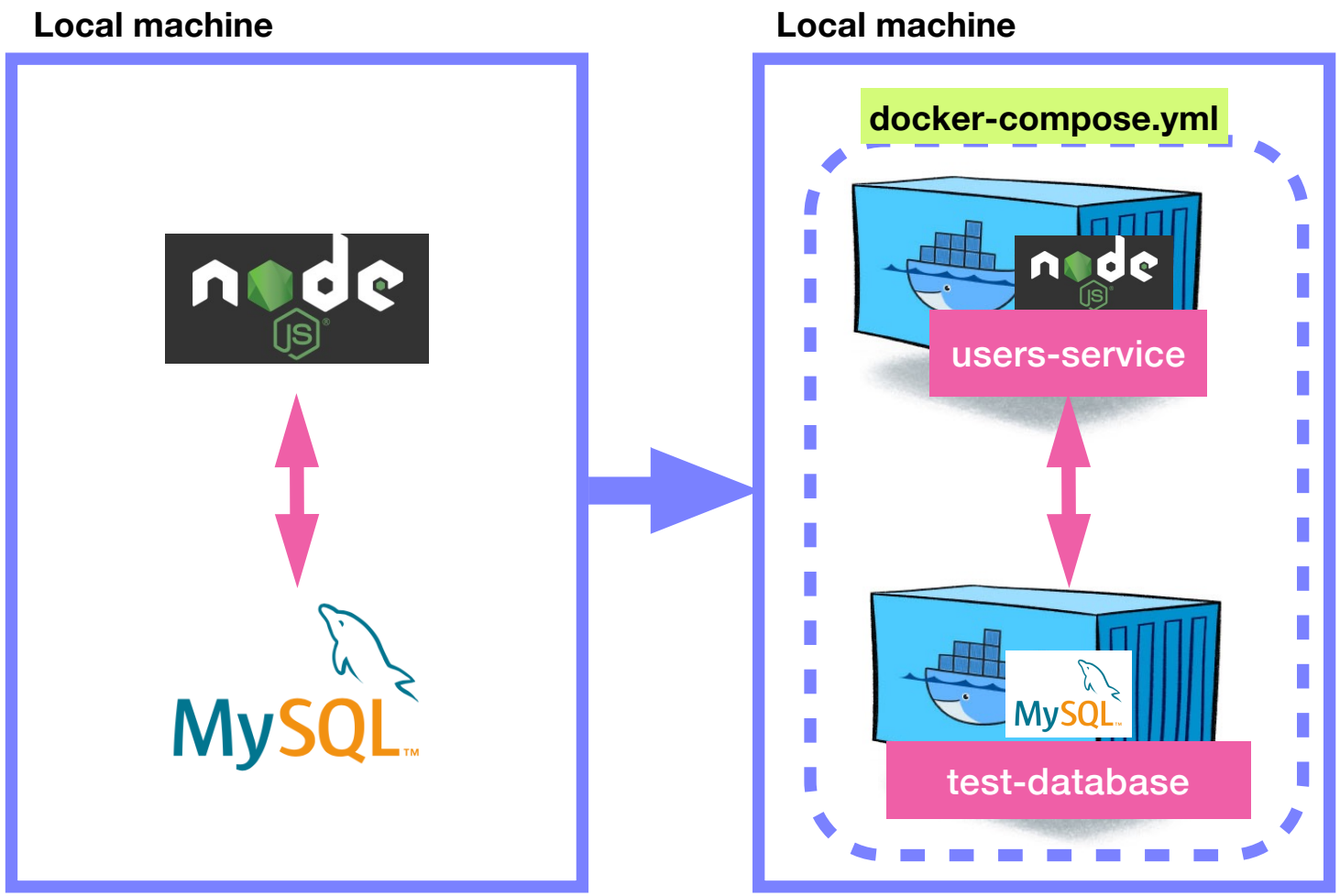
Building and running each container is still somewhat time consuming.

Docker Compose lets you create a file which defines each container in your system, the relationships between them, and build or run them all.

To do this, we need Docker Compose.

http://www.dwmkerr.com/learn-docker-by-building-a-microservice/

Composing


Local machine
Local machine
docker-compose.yml
users-service
test-database

Composing

DockerPractice/docker-compose.yml

```yaml
version: '2'
services:
  db:
    image: users_service_db
    build: ./test-database
    container_name: db1
    expose:
    - "3306"
    ports:
    - "6002:3306"
    healthcheck:
        test: ["CMD", "mysqladmin" ,"ping", "-h", "localhost"]
        timeout: 20s
        retries: 10
  users-service:
    restart: always
    depends_on:
      db:
        condition: service_healthy
    build: ./users-service
    image: users-service
    container_name: users-service
    ports:
     - "6001:6001"
    links:
    - db:db
    environment:
     - DATABASE_HOST=db
```

Where to find the Dockerfile

Where to find the Dockerfile

http://www.dwmkerr.com/learn-docker-by-building-a-microservice/

รันคำสั่งต่อไปนี้

```
docker-compose build
```

```
REPOSITORY                                TAG                 IMAGE ID            CREATED
SIZE
nodedockermicroservice_users-service      latest              d51a93f88911        17 seconds ago
689MB
nodedockermicroservice_db                 latest              23aa5375dbcf        2 minutes ago
408MB
mysql                                     5                   5fac85ee2c68        6 days ago
408MB
node                                      4                   ec7390a2bbe6        13 days ago
655MB
```

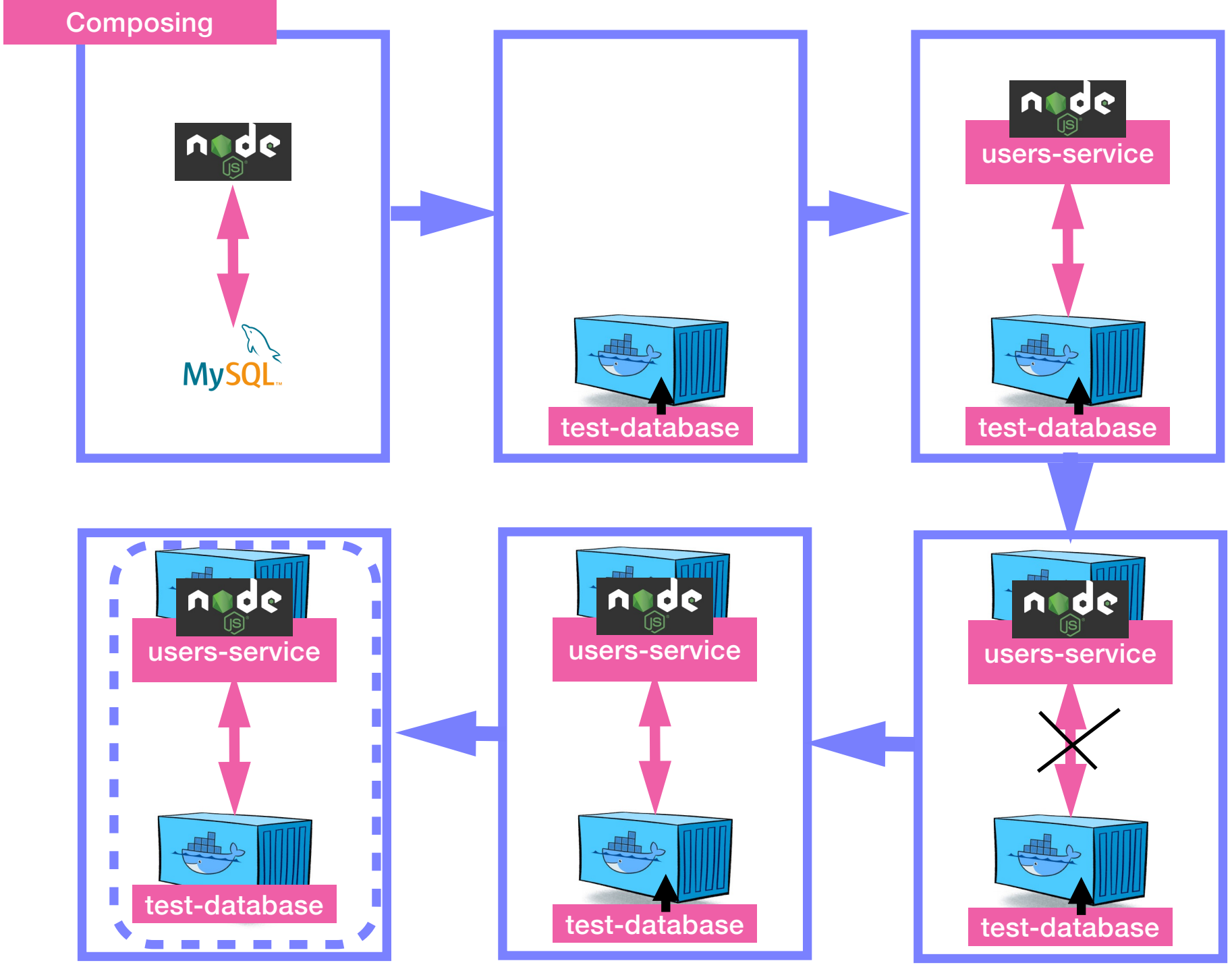
```
docker-compose up -d
docker network connect my-net db
docker network connect my-net users-service
```

http://www.dwmkerr.com/learn-docker-by-building-a-microservice/

Try

http://localhost:6001/users

http://localhost:6001/search?email=homer@thesimpsons.com


Composing
MySQL
test-database
users-service
test-database
users-service
test-database
users-service
test-database
users-service
test-database

BREAK 2 MINUTES

Docker logs

- **docker logs [container ID]**
- **docker logs [container name]**

Docker scout

- https://docs.docker.com/scout/quickstart/
- For analyzing image vulnerabilities
- For evaluating policy compliance
- For sharing the analysis of images to team members

CVSS score	Severity rating
0.1 – 3.9	Low (L)
4.0 – 6.9	Medium (M)
7.0 – 8.9	High (H)
9.0 – 10.0	Critical (C)

https://docs.docker.com/scout/image-analysis/

Docker best practices

- ✓ **Use official docker images as base image.** For example, use FROM node instead of FROM ubuntu and RUN apt-get for node

- ✓ **Use specific image version.** For example, use FROM node:17.0.1 instead of FROM node which will pull the latest one.

- ✓ **Use small-sized official images.** For example, use FROM node:17.0.1-alpine instead of FROM node:17.0.1.

- ✓ **Optimize caching image layers.** Order Dockerfile commands from least to more frequently changing.

Docker best practices

- ✓ **Use .dockerignore file** to explicitly exclude files and folders from a being built docker image.
- ✓ **Make use of "Multi-Stage Builds"** in Dockerfile
- ✓ **Use the least privileged user.** For example, node image has provided user node to run the application inside the container
- ✓ **Scan your images for vulnerabilities** using docker scan command or service on the Docker Hub.