

Docker

2110366 SE2 2566/2

docker

<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

The screenshot shows the Docker website's 'Get Started with Docker' page. At the top, there is a dark blue header bar with links for 'Docs', 'Get support', and 'Contact sales'. Below the header is a navigation bar with links for 'Products', 'Developers', 'Pricing', 'Support', 'Blog', and 'Company'. On the right side of the navigation bar are a search icon, a 'Sign In' button, and a prominent blue 'Get Started' button. The main content area features a large title 'Get Started with Docker' and a subtitle 'Build applications faster and more securely with Docker for developers'. Below this, there is a call-to-action button labeled 'Learn how to install Docker' and a dropdown menu for download options. The dropdown menu includes 'Download for Mac - Apple Chip' (which is highlighted with a red arrow), 'Download for Mac - Intel Chip', 'Download for Windows', and 'Download for Linux'. At the bottom of the page, there is a section titled 'An experience you'll love' with a subtext 'Customize your development experience with tools that enhance your tech stack and optimize your development process.' and the Docker logo.

Docs Get support Contact sales

Products Developers Pricing Support Blog Company

Sign In Get Started

Get Started with Docker

Build applications faster and more securely with Docker for developers

Learn how to install Docker

Download for Mac - Apple Chip

Download for Mac - Intel Chip

Download for Windows

Download for Linux

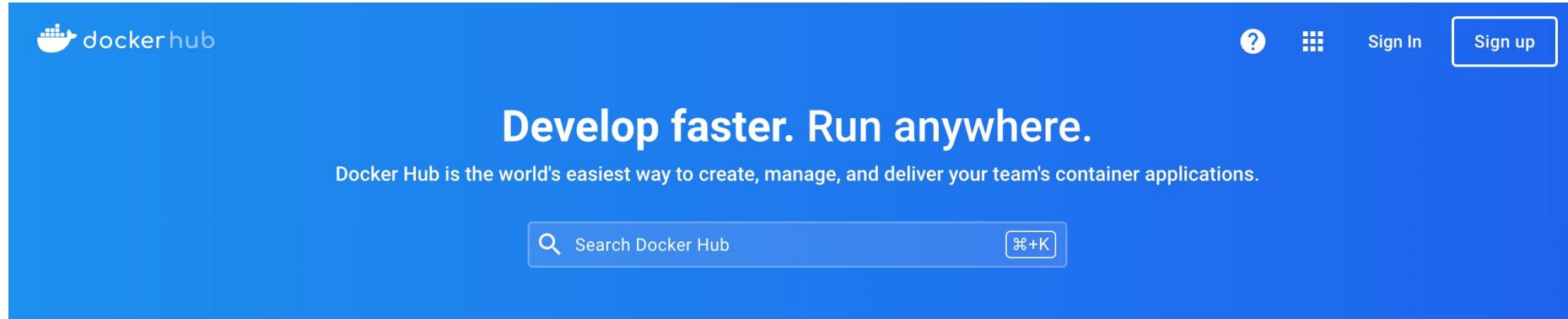
An experience you'll love

Customize your development experience with tools that enhance your tech stack and optimize your development process.



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

Create an account at Docker Hub



Spotlight

CLOUD DEVELOPMENT

Build up to 39x faster with Docker Build Cloud

Introducing Docker Build Cloud: A new solution to speed up build times and improve developer productivity



The Docker Build Cloud section features a dark blue gradient background with the 'docker buildcloud' logo. The logo consists of a stylized white diamond icon followed by the text 'docker' in a lowercase sans-serif font and 'buildcloud' in a larger, bold, lowercase sans-serif font.

AI/ML DEVELOPMENT

LLM Everywhere: Docker and Hugging Face

Set up a local development environment for Hugging Face with Docker



The AI/ML Development section features a dark blue background with the 'LLM' logo in large, bold, light blue letters. The background has a subtle circular pattern of small white dots radiating from the center.

<https://hub.docker.com/>

Install Docker extension in VS code

The screenshot shows the Visual Studio Code Marketplace interface. On the left, there's a sidebar with various icons for extensions like Git, GitHub, and others. The main area has a search bar at the top with the placeholder "Search". Below it, a list of extensions is shown, with "docker" selected. The results include:

- Docker** by Microsoft (v1.29.0): Makes it easy to create, manage, and debug containerized applications. Enabled globally.
- Docker...** by Jun Han (v4.5): Manage Docker Conta...
- Docker ...** by p1c2u (v2): Manage Docker Comp...
- Docker ...** by Henrik Sjööh (v1.5): Lint perl, python and...
- Docker E...** by Jun Han (v5): Manage Docker Conta...
- Docker Run** by Georgekutt... (v5): Start your docker con...
- Docker R...** by Zim (v3): Docker Integration for...

On the right, the details page for the **Docker** extension is displayed. It shows the extension icon (a blue Docker logo), the name **Docker**, version **v1.29.0**, developer **Microsoft**, and the URL **microsoft.com**. It has **31,932,615** installs and a rating of **★★★★★ (89)**. A brief description says: "Makes it easy to create, manage, and debug containerized applicatio...". Below this, there are buttons for **Disable** and **Uninstall**. A note states: "This extension is enabled globally." Below the extension details, there's a section for **Docker for Visual Studio Code** (version v1.29.0), which includes stats like "installs 32M" and "Azure Pipelines succeeded". A description explains: "The Docker extension makes it easy to build, manage, and deploy containerized applications from Visual Studio Code. It also provides one-click debugging of Node.js, Python, and .NET inside a container." At the bottom, there's a screenshot of the VS Code interface showing the Docker extension in action.

Categories

- Programming Languages
- Linters
- Azure

Resources

- Marketplace
- Issues
- Repository
- License
- Microsoft

Keywords

Unlike VM

Share files among containers

Dokerfile

Docker image

Port forwarding

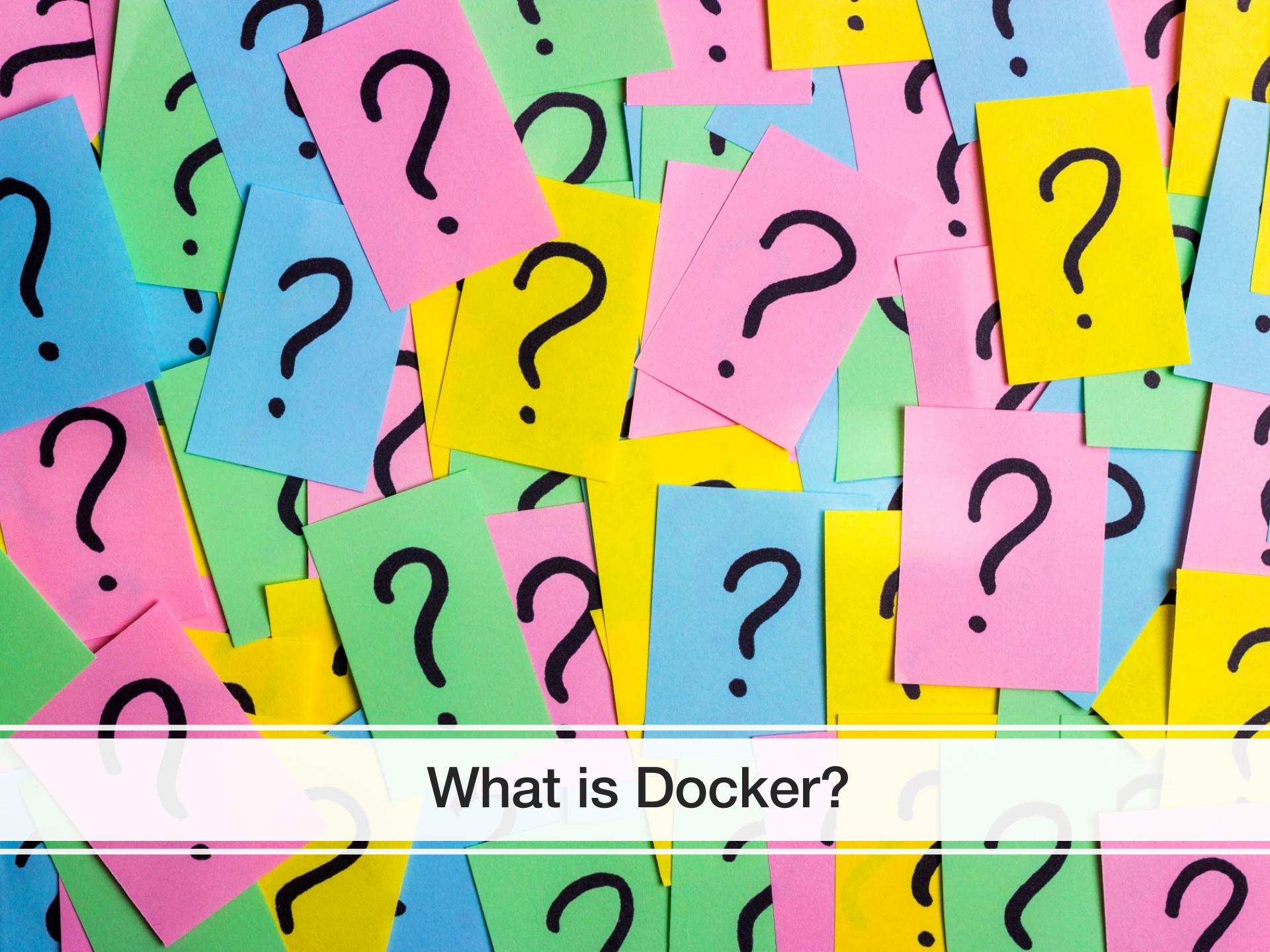
Docker hub

Container

Some docker commands

Docker compose

Volume

A large pile of colorful sticky notes, mostly yellow, pink, and blue, with black question marks drawn on them. They are stacked and overlapping, creating a textured, layered effect.

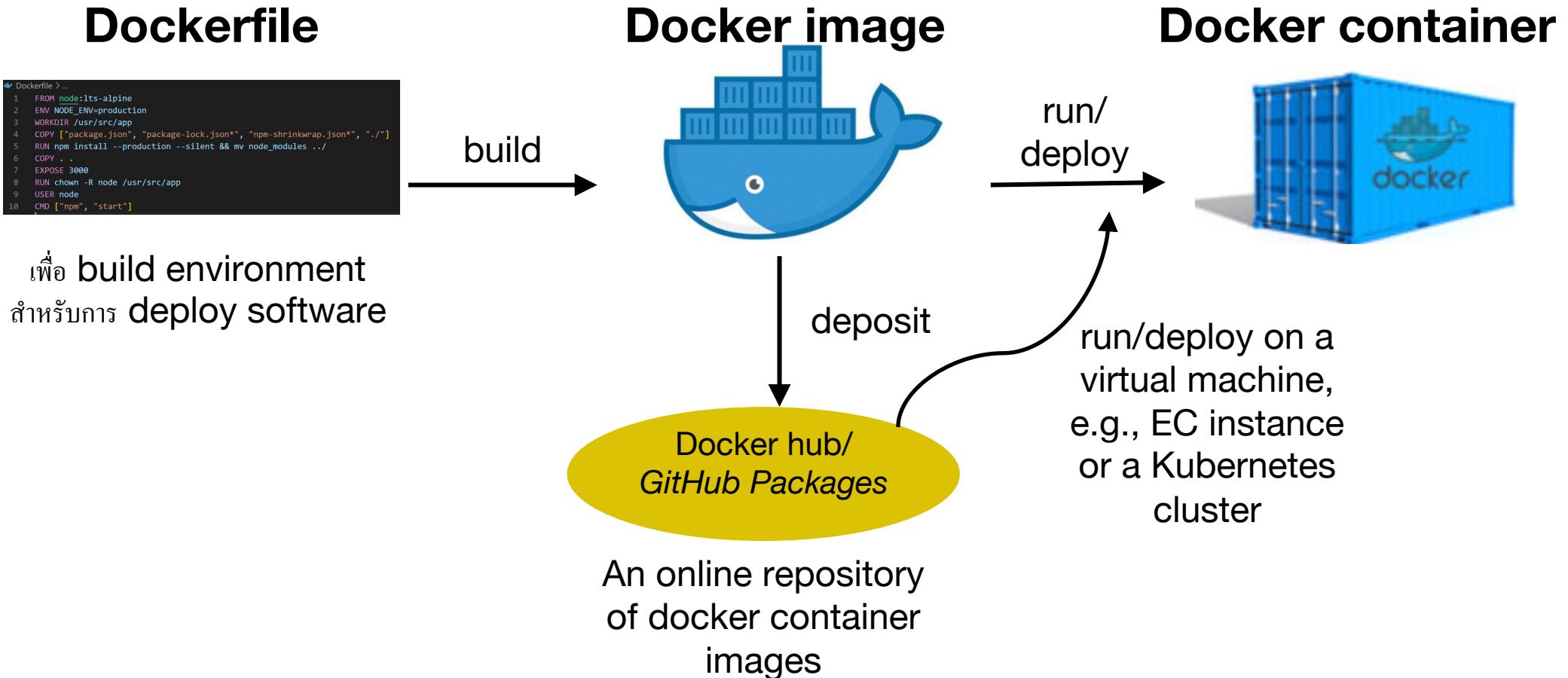
What is Docker?

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



Why containerization?

Before

Different of installation process on different OS

MySQL, mongodb, PostgreSQL, Redis, etc..

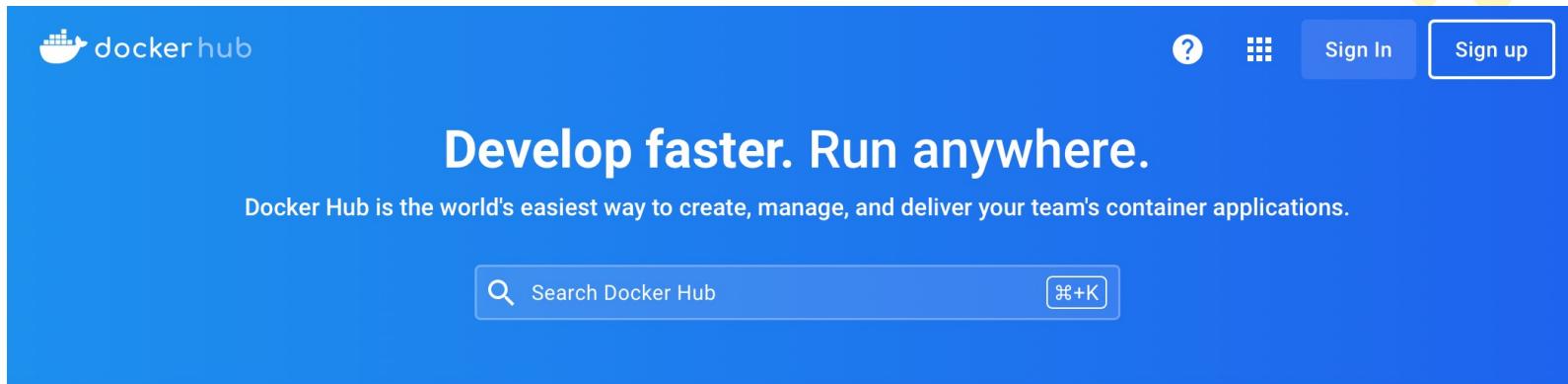
Python, node.js, Go, PHP, Java, R, etc.

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



The screenshot shows the Docker Hub homepage with a blue header. On the left is the Docker Hub logo. On the right are navigation links for 'Sign In' and 'Sign up'. Below the header is a large text area with the slogan 'Develop faster. Run anywhere.' followed by a subtext: 'Docker Hub is the world's easiest way to create, manage, and deliver your team's container applications.' A search bar with a magnifying glass icon and the placeholder 'Search Docker Hub' is centered below the slogan. To the right of the search bar is a keyboard shortcut '⌘+K'.

Spotlight

CLOUD DEVELOPMENT

Build up to 39x faster with Docker Build Cloud

Introducing Docker Build Cloud: A new solution to speed up build times and improve developer productivity



AI/ML DEVELOPMENT

LLM Everywhere: Docker and Hugging Face

Set up a local development environment for Hugging Face with Docker



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

 docker hub Explore Repositories Organizations x ?  

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 

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 

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 hub](#) [Explore](#) [Repositories](#) [Organizations](#)

x

[?](#) [grid](#) [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

The screenshot shows the Docker Desktop application window. The left sidebar has icons for Containers, Images, Volumes, Builds, Dev Environments (BETA), Docker Scout, Extensions, and Add Extensions. The main area is titled "Containers" with a search bar and a "Give feedback" link. It displays resource usage statistics: Container CPU usage (0.00% / 800%) and Container memory usage (8.98MB / 3.74GB). A "Show charts" link is also present. Below these stats is a table with columns: Name, Image, Status, CPU (%), Port(s), Last started, and Actions. One row is listed: Name is "admiring_ch", Image is "ubuntu ef18d8d7b5b2", Status is "Running", CPU (%) is "0%", Last started is "52 seconds ago", and Actions show a copy icon, a more options icon, and a delete icon.

	Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
	admiring_ch	ubuntu ef18d8d7b5b2	Running	0%		52 seconds ago	

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
```

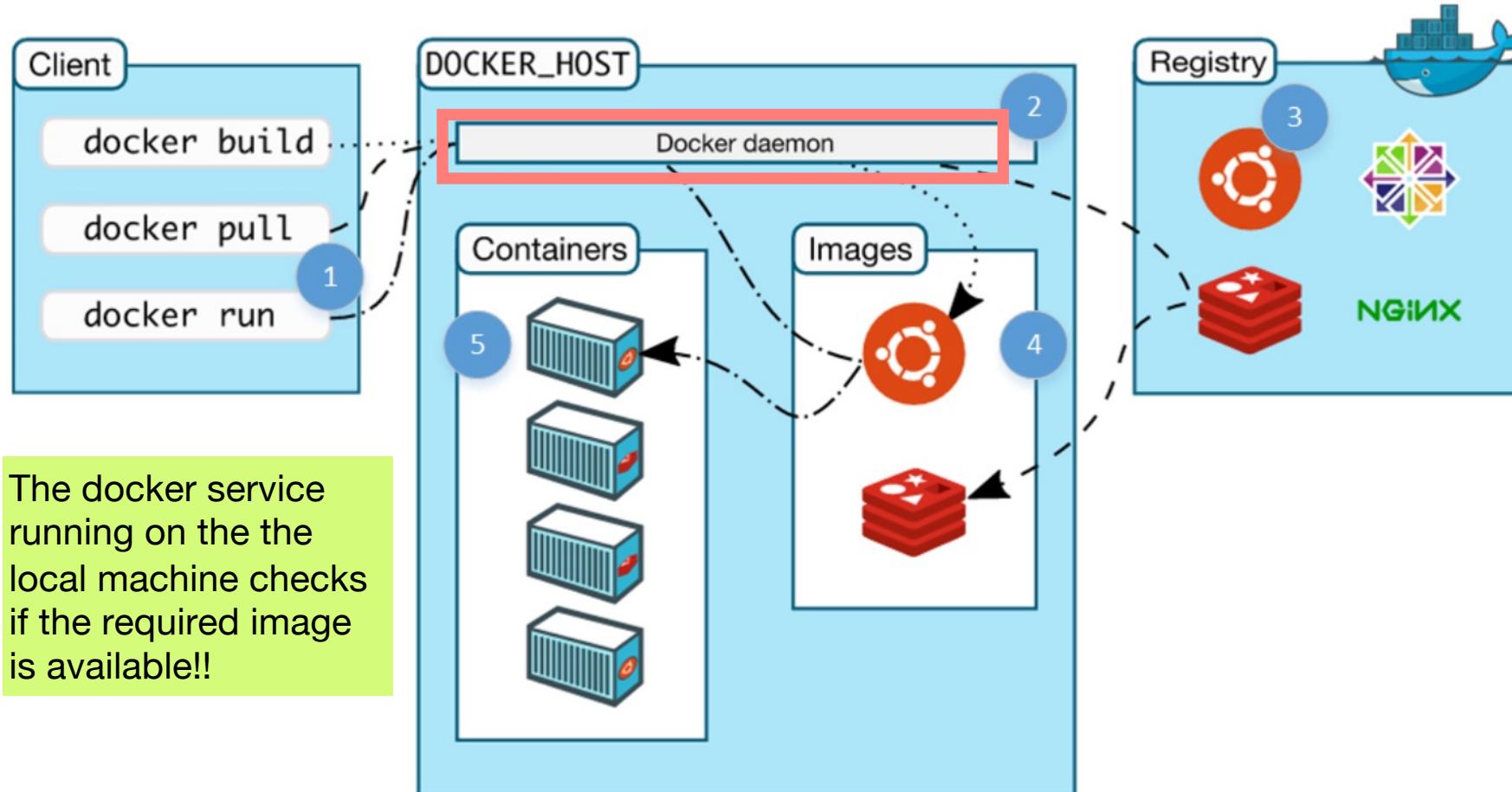
Automatically clean up the container and remove the file system when the container exits

What you will get is as follows....

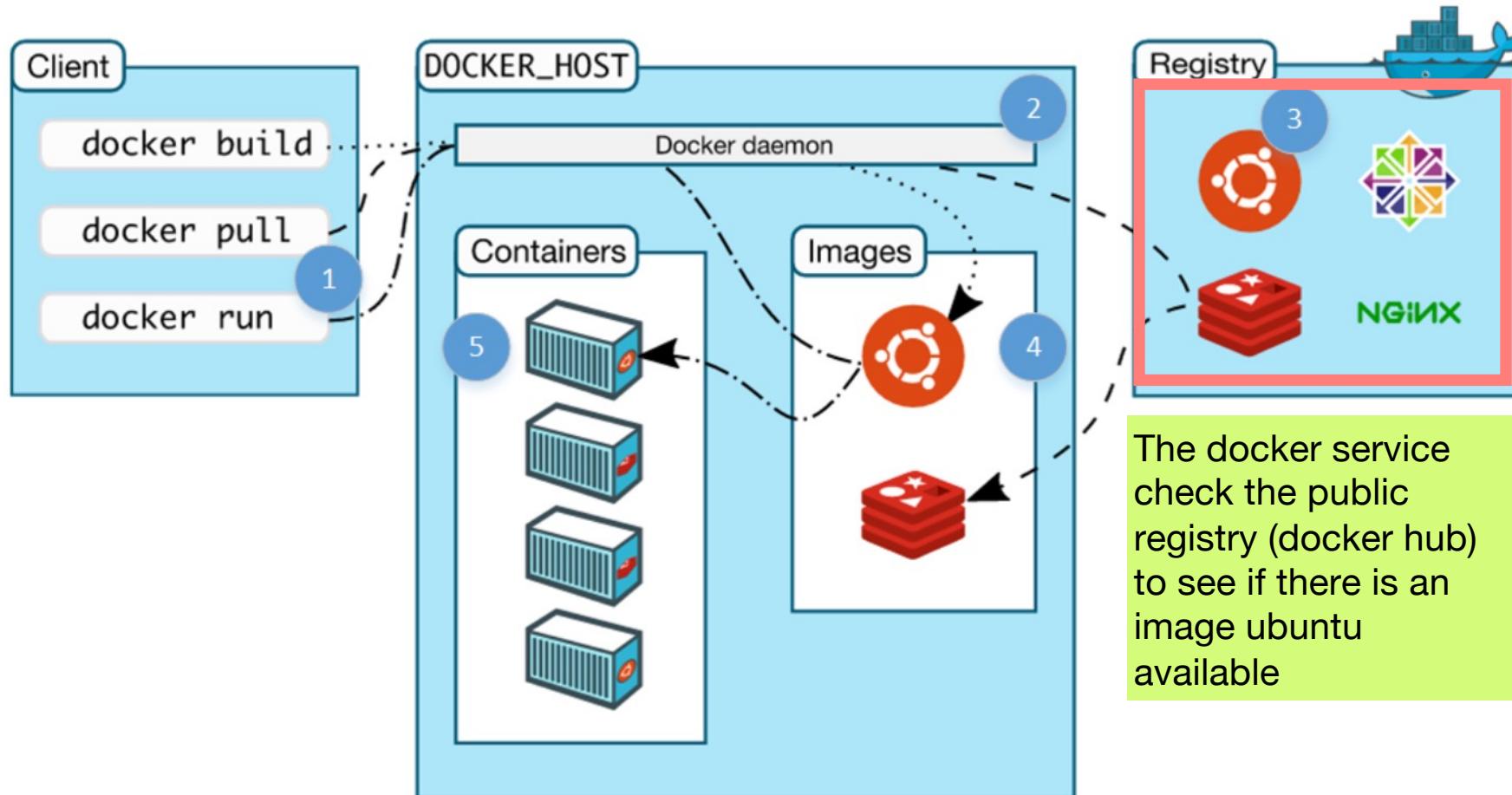
```
(base) Duangdaos-MacBook-Pro:~ wichadak$ docker run -it ubuntu
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
423ae2b273f4: Pull complete
de83a2304fa1: Pull complete
f9a83bce3af0: Pull complete
b6b53be908de: Pull complete
Digest: sha256:04d48df82c938587820d7b6006f5071dbbfceb7ca01d2814f81857c631d44df
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

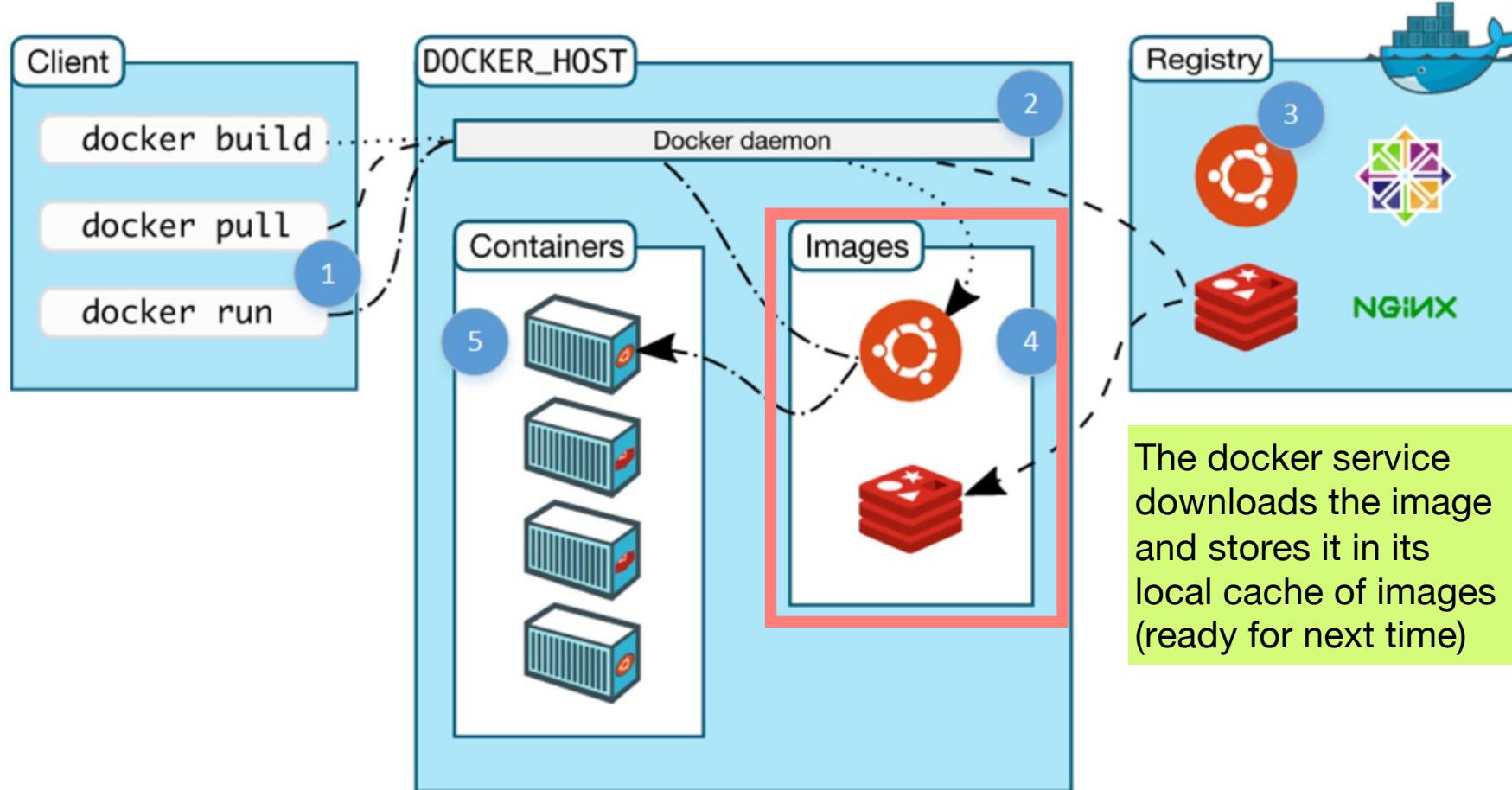
What has happened?



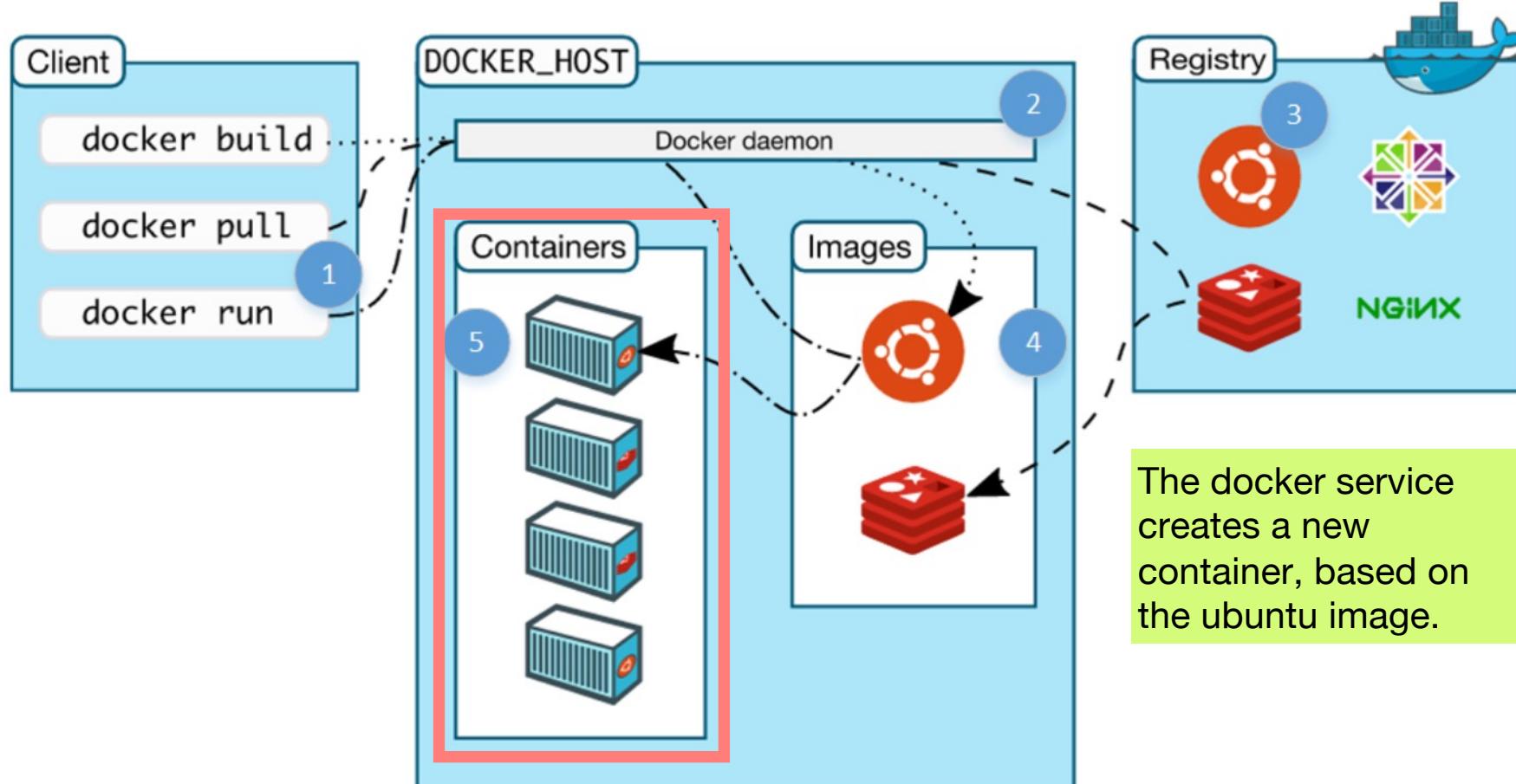
What has happened?



What has happened?



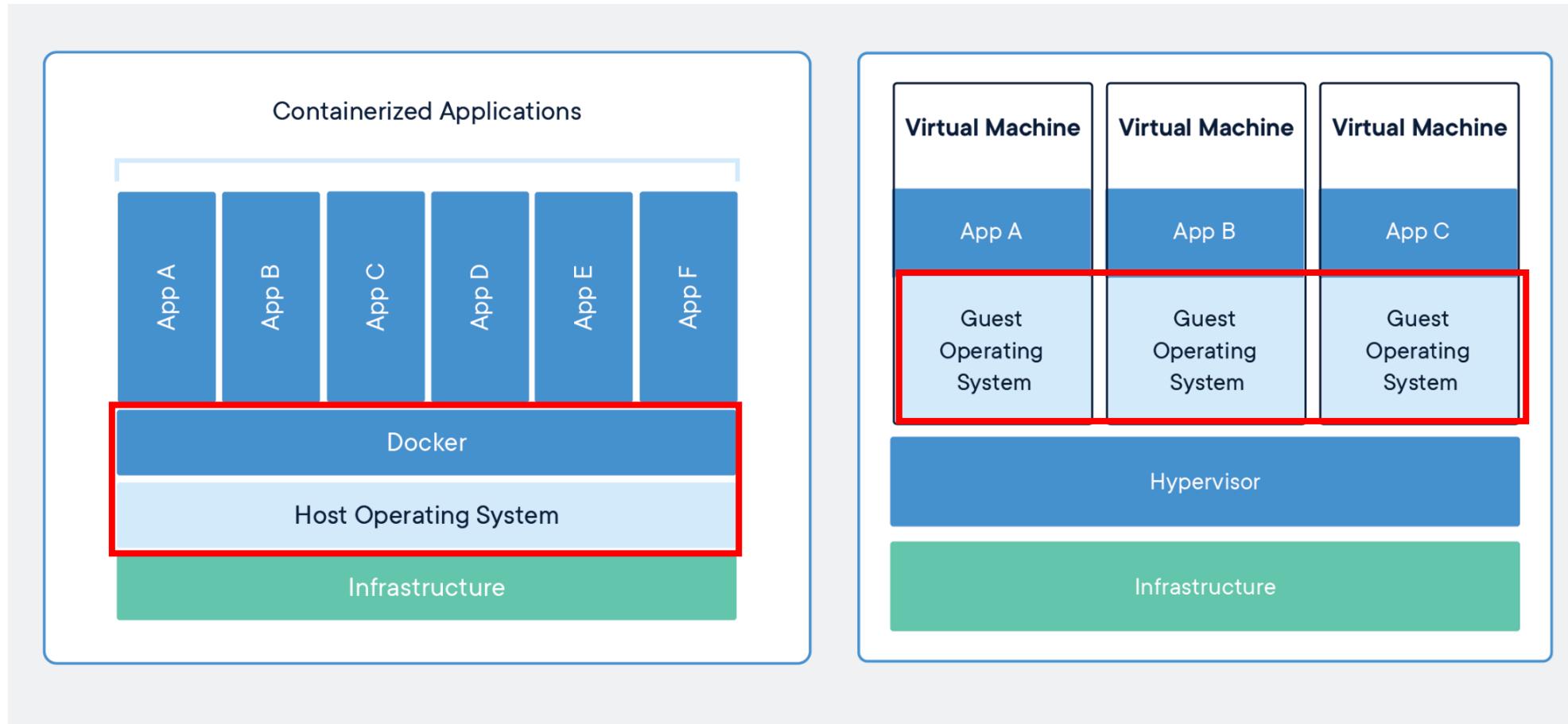
What has happened?



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

Notice that different line patterns represent different command

Compare between containers and virtual machines (VMS)



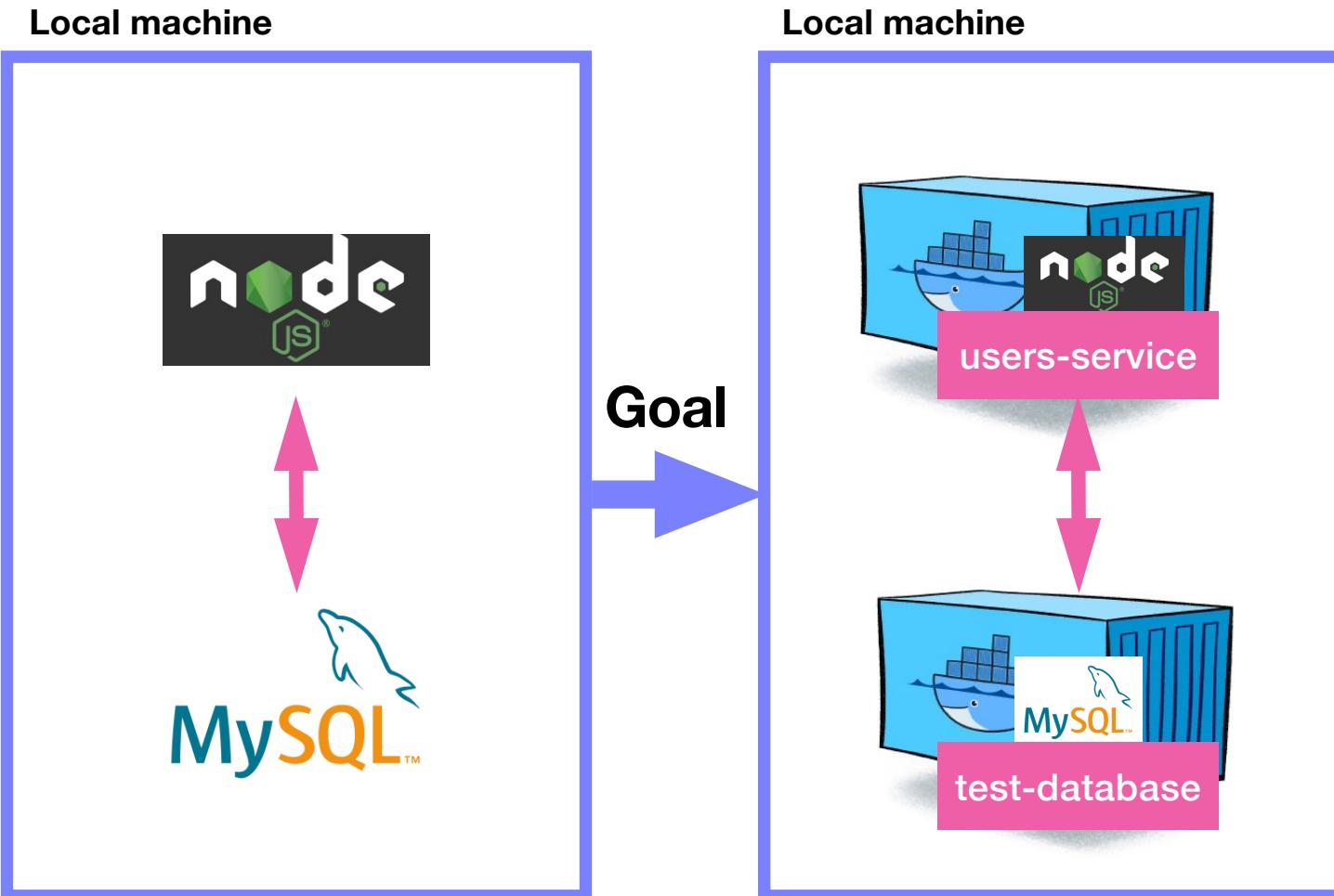
BREAK 2 MINUTES



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?







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

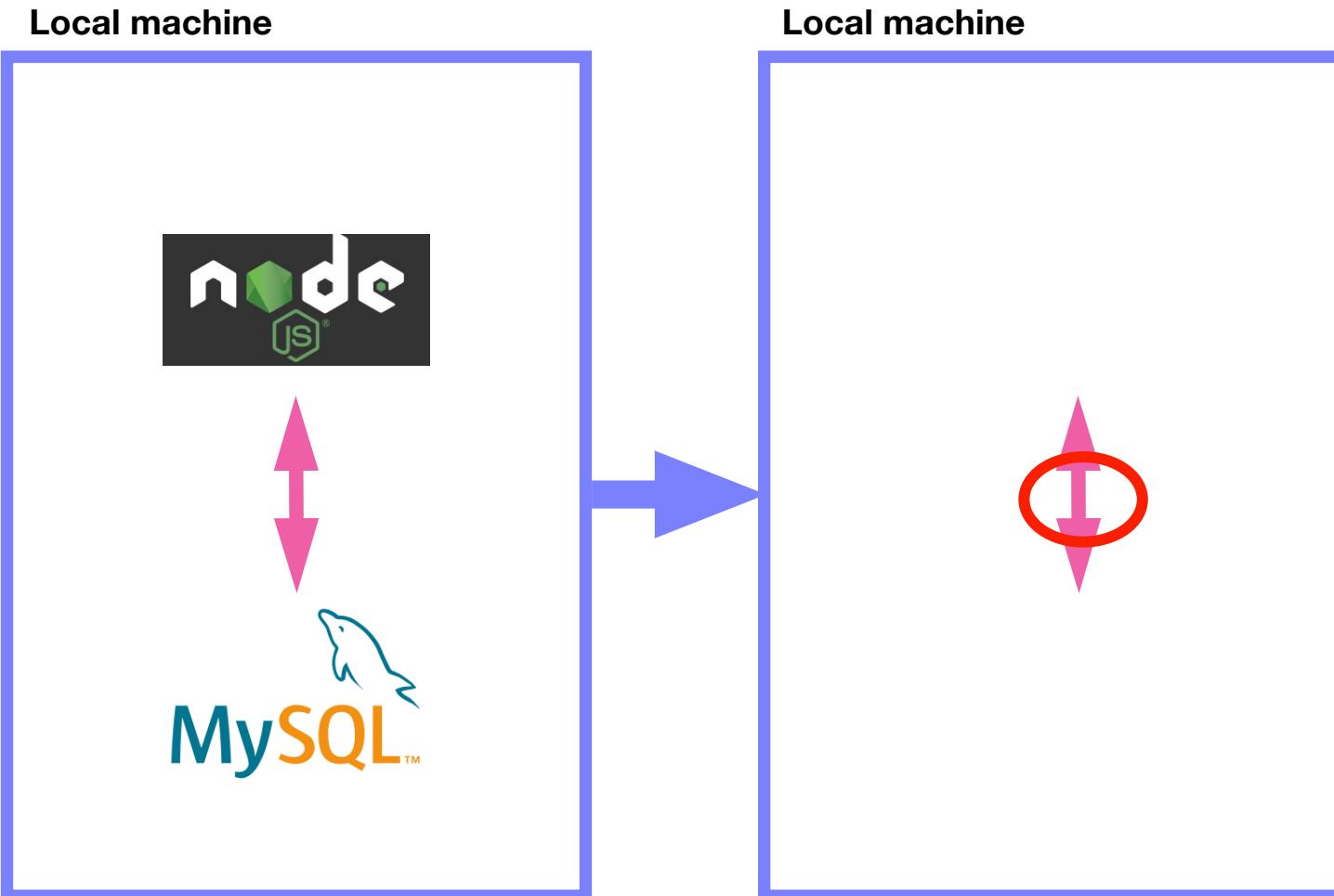
https://hub.docker.com/_/mysql

ลองทำอย่างแรก ใช้คำสั่งต่อไปนี้ เพื่อสร้างเครือข่ายเสมือน
ขึ้นมาไว้สำหรับให้ containers คุยกัน

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

```
docker network ls
```

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



สร้างเครือข่ายเสมือนชื่อ my-net ให้ containers คุยกันเสร็จแล้ว

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

test-database/setup.sql

setup.sql X

test-database > setup.sql

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

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
```

List docker images in our local machine

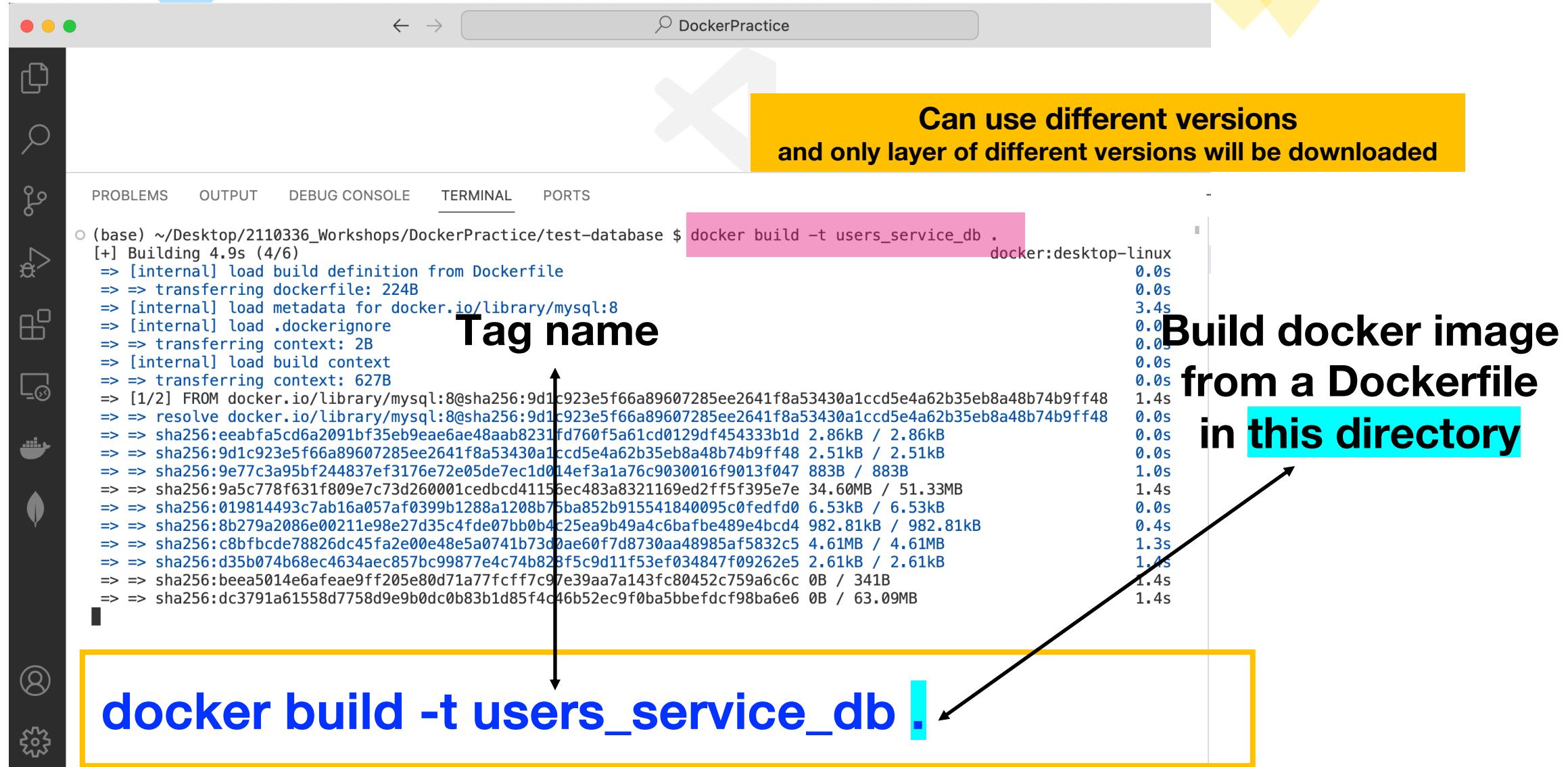
docker images

```
● (base) ~/Desktop/2110336_Workshops/DockerPractice/test-database $ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
------------	-----	----------	---------	------

docker system prune / docker system prune -a

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



The screenshot shows a terminal window in VS Code with the title bar "DockerPractice". The terminal tab is selected, showing the command:

```
(base) ~/Desktop/2110336_Workshops/DockerPractice/test-database $ docker build -t users_service_db .
```

A yellow callout box contains the text:

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

An arrow points from the text "Tag name" to the tag "-t users_service_db" in the command line.

A large blue callout box contains the text:

Build docker image
from a Dockerfile
in this directory

An arrow points from the bottom of the callout box to the command line.

Below the command line, another yellow callout box contains the text:

docker build -t users_service_db .

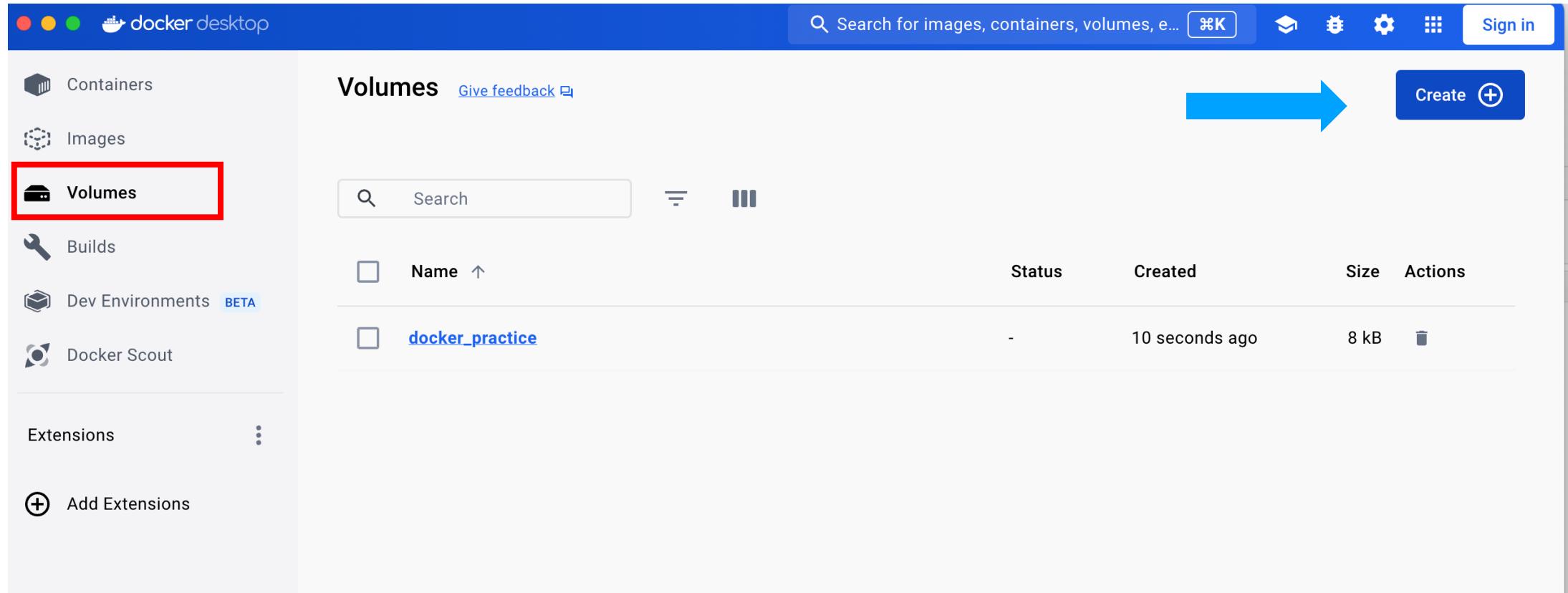
An arrow points from the bottom of this callout box to the command line.

List docker images in our local machine

docker images

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

Create a volume via Docker Desktop



The screenshot shows the Docker Desktop interface. The left sidebar has icons for Containers, Images, Volumes (which is highlighted with a red box), Builds, Dev Environments (BETA), and Docker Scout. The main area is titled "Volumes" and contains a table with columns: Name, Status, Created, Size, and Actions. One row is shown: "docker_practice" (Status: -, Created: 10 seconds ago, Size: 8 kB). A blue arrow points from the "Volumes" tab towards the "Create" button in the top right corner of the main area.

Name	Status	Created	Size	Actions
docker_practice	-	10 seconds ago	8 kB	

docker volume create [volume name]

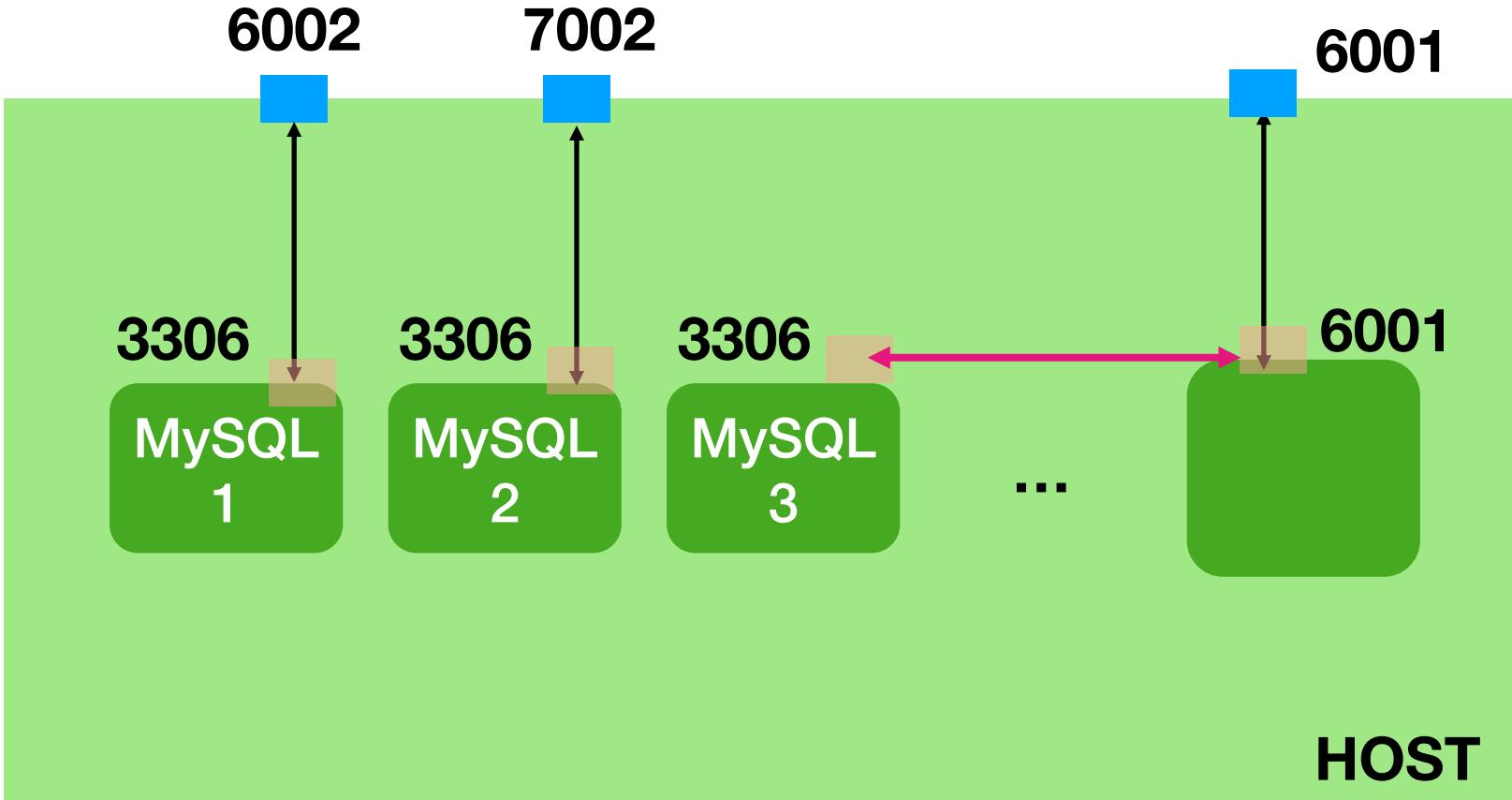
Run the `users_service_db` via `docker run`

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

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`

ลองใช้คำสั่งนี้ดูว่า db container ถูกสร้างเรียบร้อย
ไหม

docker container ls /docker ps

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

รันคำสั่งข้างล่างนี้เพื่อเชื่อมต่อเข้าไปที่ container db1 ที่สร้างไว้

docker exec -it db1 /bin/bash

จากนั้น login เข้า MySQL โดยใช้ user: users_service และ
ใช้ password: 123 ตามที่กำหนดไว้ 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> use users      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;    show tables;
+-----+
| Tables_in_users |
+-----+
| DIRECTORY       |
+-----+
1 row in set (0.03 sec)
```

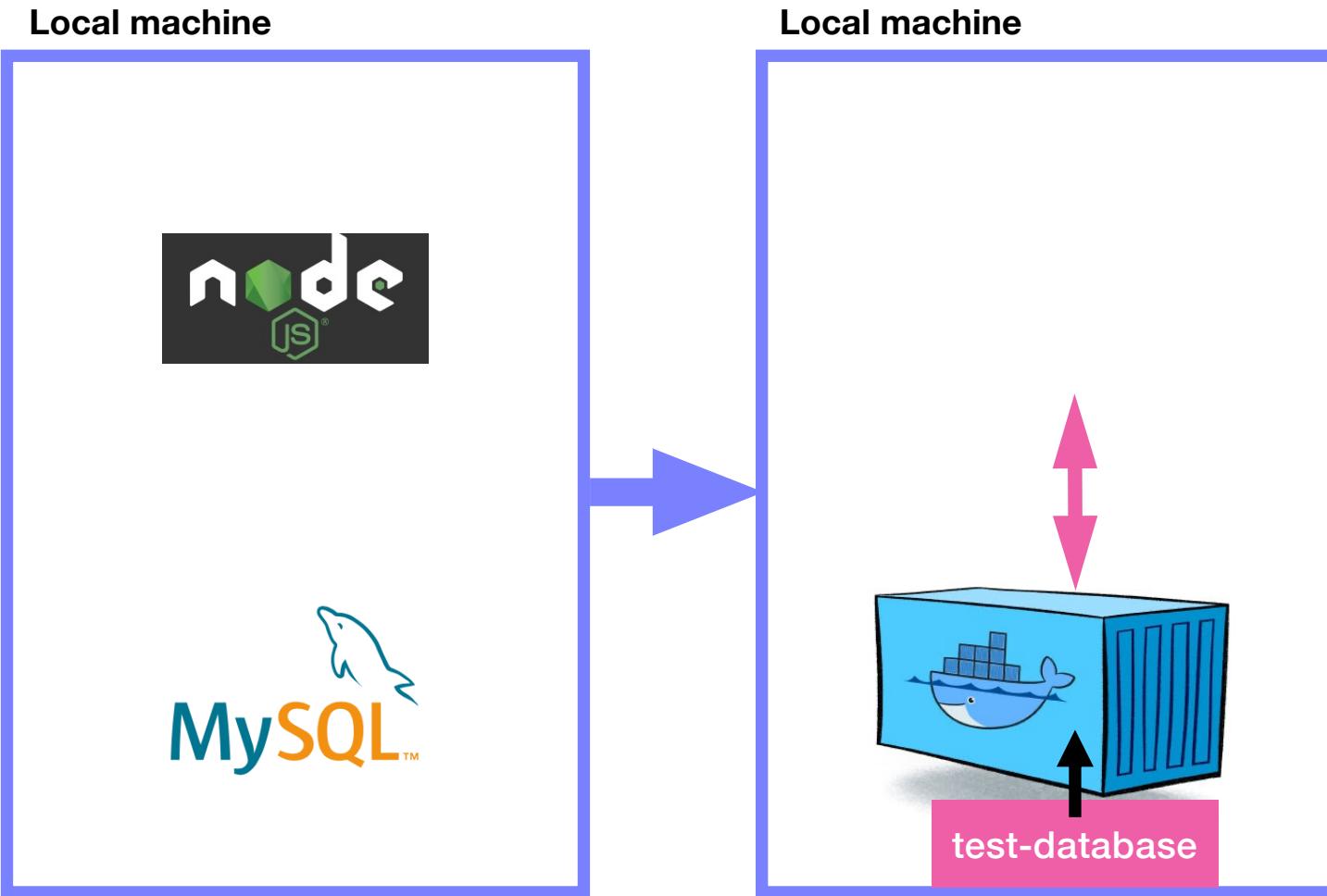
```
mysql> select * from directory;
ERROR 1146 (42S02): Table 'users.directory' doesn't exist
mysql> select * from DIRECTORY;  SELECT * FROM DIRECTORY;
+-----+-----+-----+
| user_id | email           | phone_number |
+-----+-----+-----+
|      1  | homer@thesimpsons.com | +1 888 123 1111 |
|      2  | 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)
```

เราสามารถลอง connect ไปที่ mysql ใน container โดยใช้ port 6002 ก็ได้

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

test-database/stop.sh

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



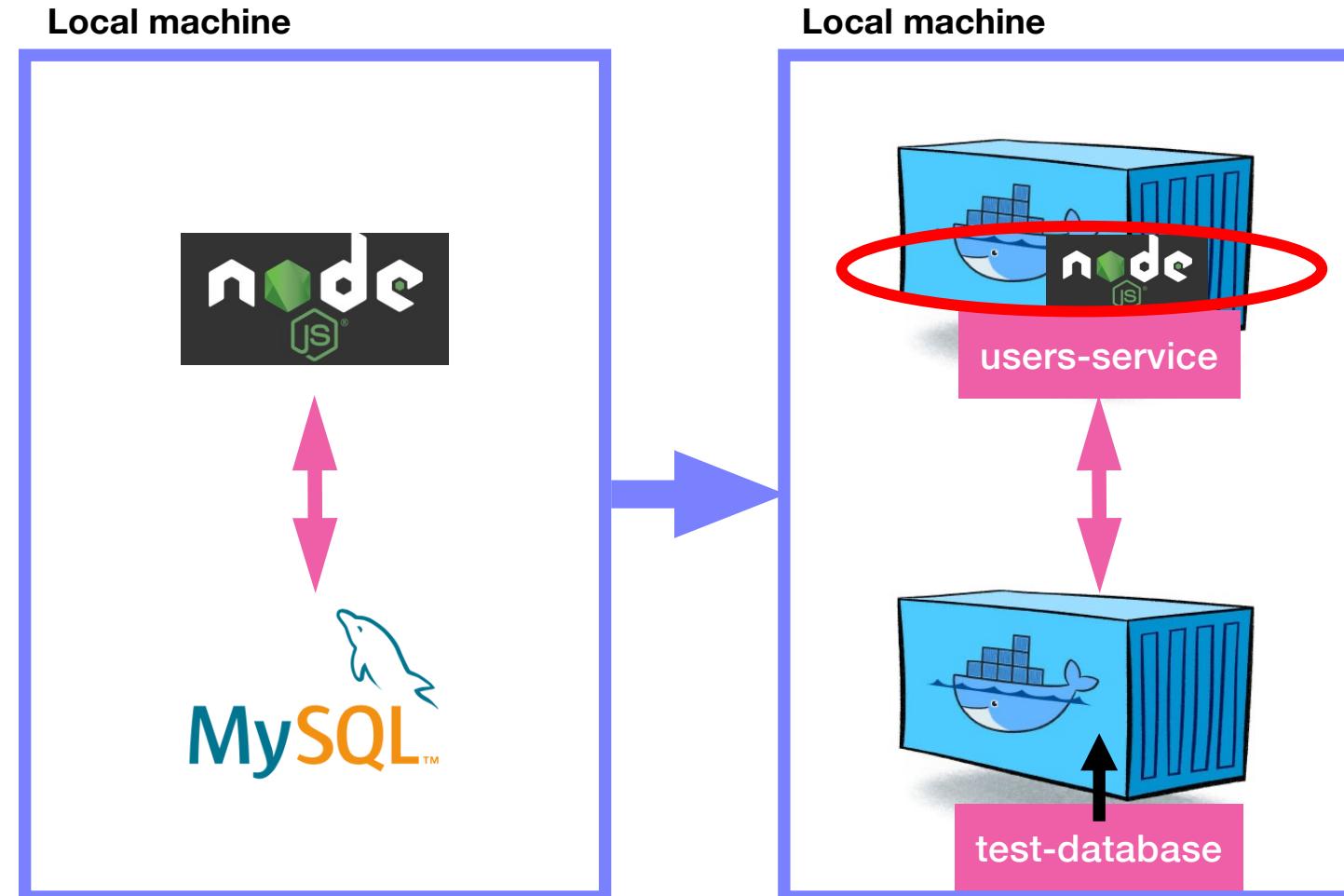
สร้าง container ชื่อ db โดยต่อกับเครือข่ายเสมือนชื่อ my-net เสร็จแล้ว

BREAK 2 MINUTES



Build microservices with Docker

Step 2: Creating a microservice in Node.js



Build microservices with Docker

Step 2: Creating a microservice in Node.js

```
test-database/          # contains the code seen in Step 1
users-service/
- 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
```

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
```

users-service/repository/repository.js

users-service > repository >  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 ข้อมูล
ที่ต้องการ

users-service/repository/repository.js

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

```
11  class Repository {  
12      // ...  
13  
14      getUsers() {  
15          // ...  
16          return new Promise((resolve, reject) => {  
17              // ...  
18              this.connection.query('SELECT email, phone_number FROM DIRECTORY', (err, results) => {  
19                  // ...  
20                  if(err) {  
21                      // ...  
22                      return reject(new Error('An error occurred getting the users: ' + err));  
23                  }  
24  
25                  resolve((results || []).map(user) => {  
26                      // ...  
27                      return {  
28                          email: user.email,  
29                          phone_number: user.phone_number  
30                      };  
31                  });  
32              }  
33          );  
34      }  
35  }
```

getUsers()

Query to get all users...

users-service/repository/repository.js

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

```
11  class Repository {
12
13    getUserByEmail(email) {
14
15      return new Promise((resolve, reject) => {
16
17        // Fetch the customer.
18        this.connection.query('SELECT email, phone_number FROM DIRECTORY WHERE email = ?',
19        [email], (err, results) => {
20          if(err) {
21            return reject(new Error('An error occurred getting the user: ' + err));
22          }
23
24          if(results.length === 0) {
25            resolve(undefined);
26          } else {
27            resolve({
28              email: results[0].email,
29              phone_number: results[0].phone_number
30            });
31          }
32        });
33      });
34    });
35  };
36}
```

getUserByEmail(email)

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

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
```

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 repository.");
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  };
```

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
```

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

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?e=...'" );  
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);  
42 }
```

Define the users api

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
```

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

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
```

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

```
1 // config.js
2 //
3 // Simple application configuration. Extend as needed.
4 module.exports = {
5   ...
6   port: process.env.PORT || 6006, // user service port
7   db: {
8     host: process.env.DATABASE_HOST || '127.0.0.1',
9     database: 'users',
10    user: 'users_service',
11    password: '123'.
12    port: 6002 // local host port which is mapped to mysql container port
13  };
14};
```

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**

```
1  // 20200313203351
2  // http://localhost:6001/users
3
4  [
5  {
6      "email": "homer@thesimpsons.com",
7      "phoneNumber": "+1 888 123 1111"
8  },
9  {
10     "email": "marge@thesimpsons.com",
11     "phoneNumber": "+1 888 123 1112"
12 },
13 {
14     "email": "maggie@thesimpsons.com",
15     "phoneNumber": "+1 888 123 1113"
16 },
17 {
18     "email": "lisa@thesimpsons.com",
19     "phoneNumber": "+1 888 123 1114"
20 },
21 {
22     "email": "bart@thesimpsons.com",
23     "phoneNumber": "+1 888 123 1115"
24 }
25 ]
```

Try<http://localhost:6001/search?email=homer@thesimpsons.com>**ผลที่ได้**

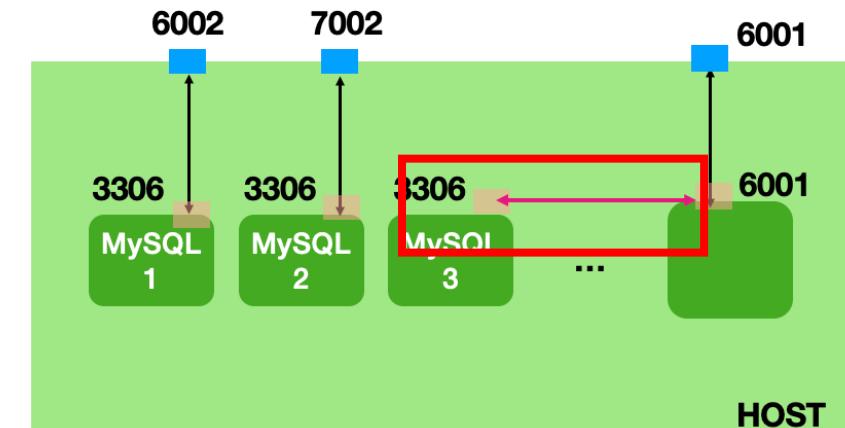
```
1 // 20200313203245
2 // http://localhost:6001/search?email=homer@thesimpsons.com
3
4 {
5     "email": "homer@thesimpsons.com",
6     "phoneNumber": "+1 888 123 1111"
7 }
```

คราวน์มาเริ่มทำให้ user service มาเป็นอีก container ด้วย

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

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};
```



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

...

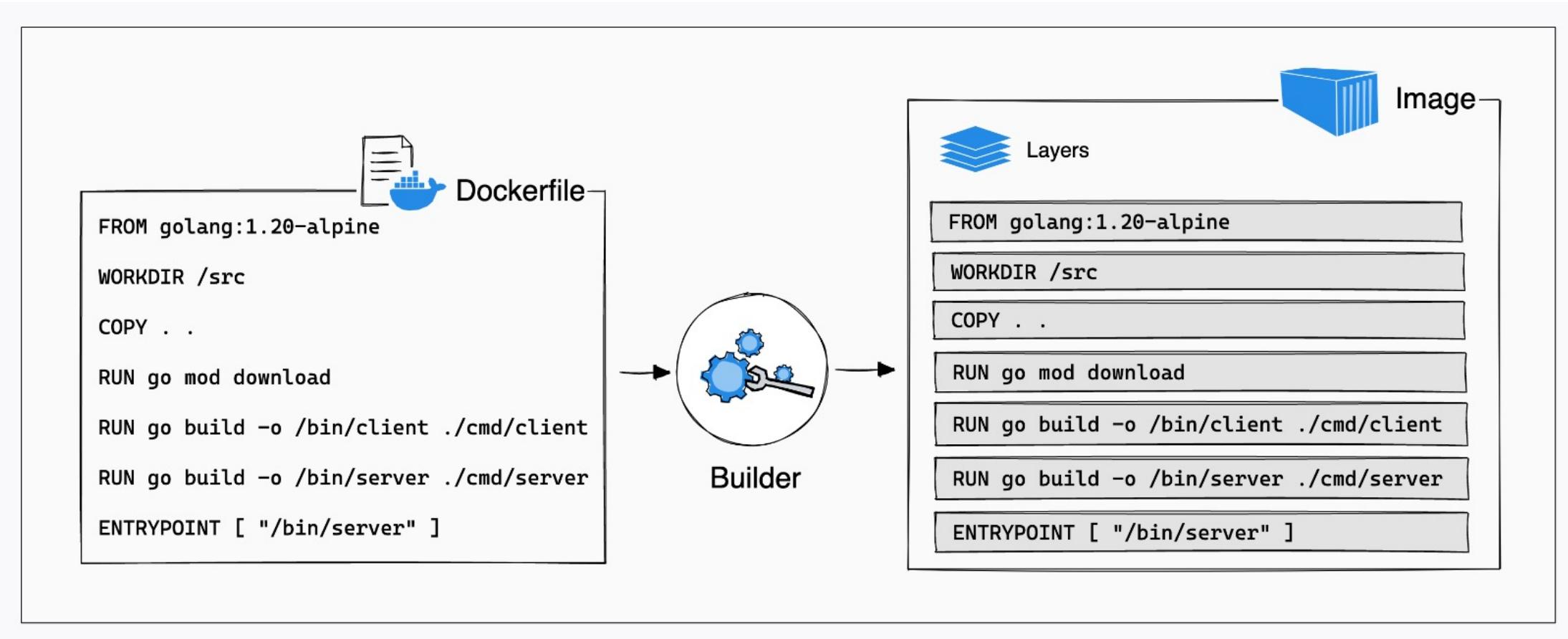
docker build -t users-service .

create
a new image

find the Dockerfile from
the current directory

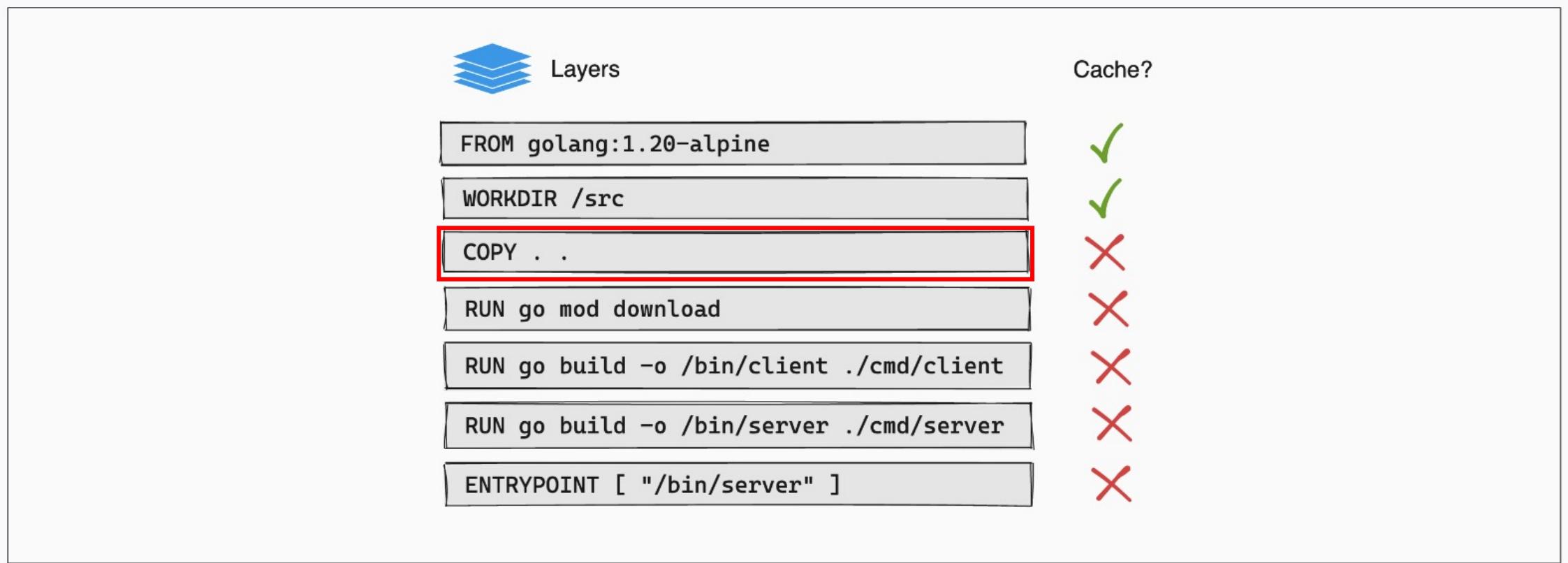
สร้าง docker image ของ
user service

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

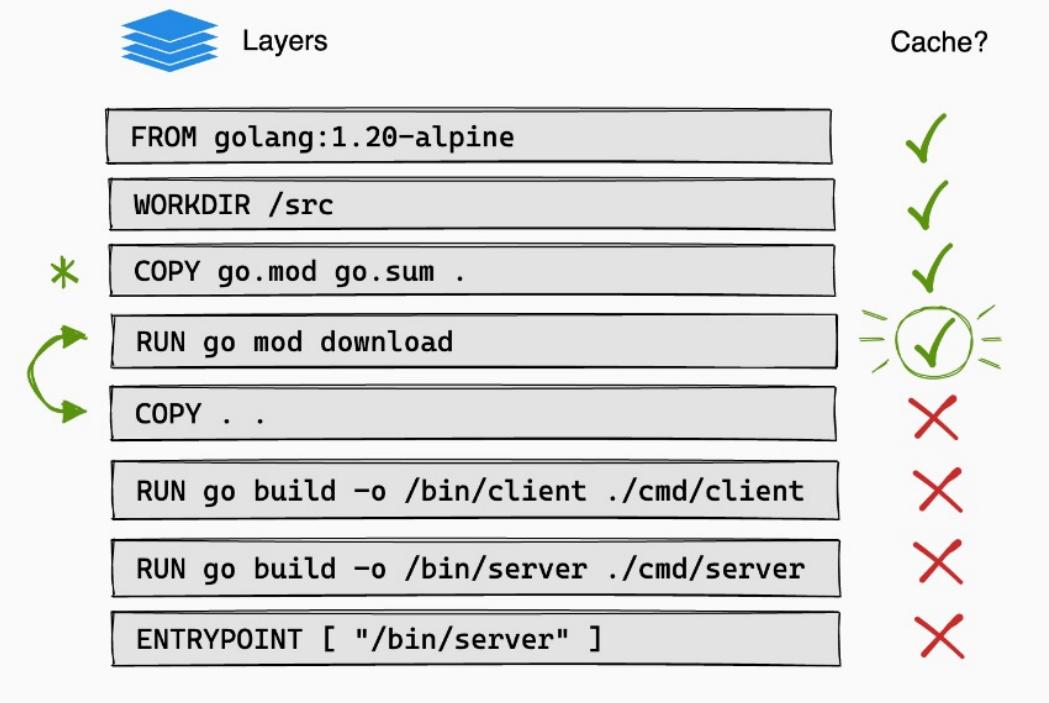
<https://docs.docker.com/build/guide/layers/>

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
```

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'  
}
```

docker desktop

Search for images, containers, volumes, e...

Containers [Give feedback](#)

Images

Volumes

Builds

Dev Environments BETA

Docker Scout

Extensions

Add Extensions

Container CPU usage **0.61% / 800%** (8 CPUs available)

Container memory usage **449.22MB / 3.74GB**

Show charts

Search Only show running containers

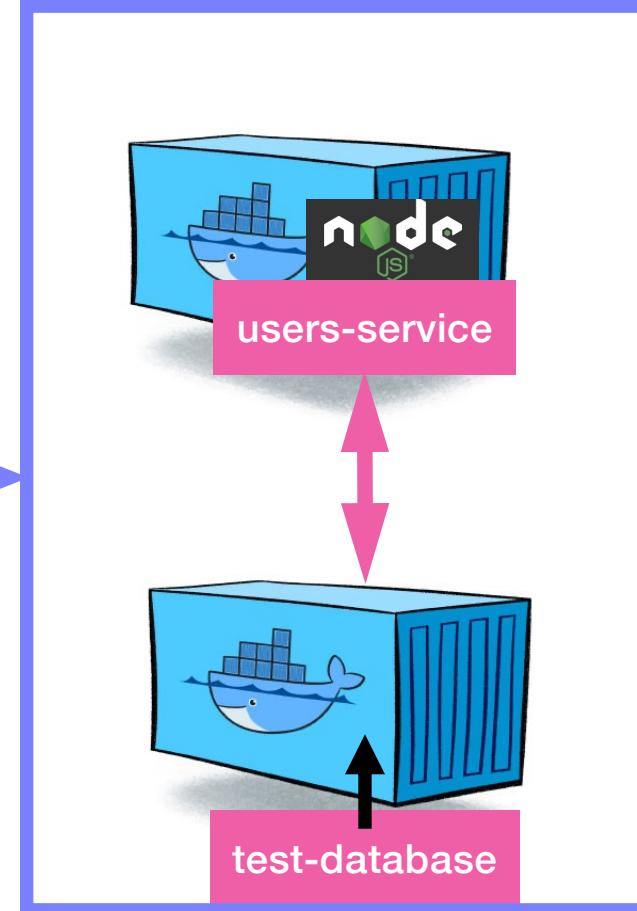
	Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
	db1 69de26d172b...	users_service_db	Running	0.61%	6002:3306	2 minutes ago	
	users-service caecc3c7c45c...	users-service	Running	0%	6001:6001	18 seconds ago	

มี container ของ users_service ขึ้นแล้ว แต่เชื่อมต่อไปยัง db1 ไม่ได้

Local machine



Local machine



docker ps -a

CONTAINER ID	IMAGE NAMES	COMMAND	CREATED	STATUS	PORTS
31c1be801bdd 8123/tcp ff64350a3440 .0 0.6002_~3306/tcp	users_service users-service db db	"docker-entrypoint.s..." "docker-entrypoint.s..."	31 seconds ago 21 minutes ago	Up 31 seconds Up 21 minutes	0.0.0.0:6001-> 33060/tcp, 0.0

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

(1) เลือก users-service container

(2) ลบออกไป เดียวจะสร้างใหม่จะได้ไม่ใช่ชื่อ port เดิม

	Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
<input type="checkbox"/>	db1 19b7e998064	users_service_db	Running	0.56%	6002:3306	6 minutes ago	<input type="button"/> <input type="button"/> <input type="button"/>
<input checked="" type="checkbox"/>	users-service eb526501e621	users-service	Running	0%	6001:6001	2 minutes ago	<input type="button"/> <input type="button"/> <input type="button"/>

ลองดูด้วยว่า db1 container ต่อ กับ 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...

```
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.
```

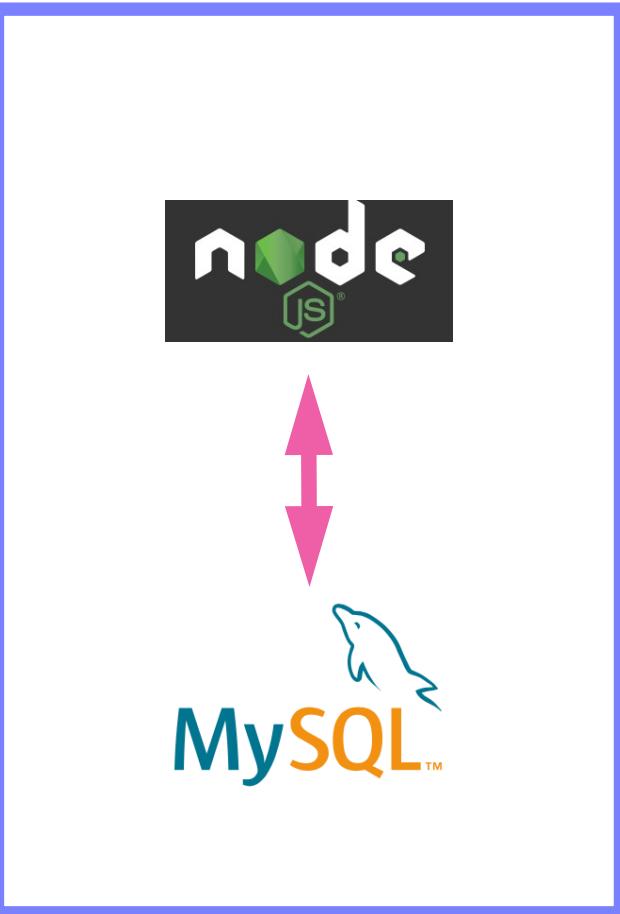
Try<http://localhost:6001/users>**ผลที่ได้**

```
1 // 20200313203351
2 // http://localhost:6001/users
3
4 [
5   {
6     "email": "homer@thesimpsons.com",
7     "phoneNumber": "+1 888 123 1111"
8   },
9   {
10    "email": "marge@thesimpsons.com",
11    "phoneNumber": "+1 888 123 1112"
12  },
13  {
14    "email": "maggie@thesimpsons.com",
15    "phoneNumber": "+1 888 123 1113"
16  },
17  {
18    "email": "lisa@thesimpsons.com",
19    "phoneNumber": "+1 888 123 1114"
20  },
21  {
22    "email": "bart@thesimpsons.com",
23    "phoneNumber": "+1 888 123 1115"
24  }
25 ]
```

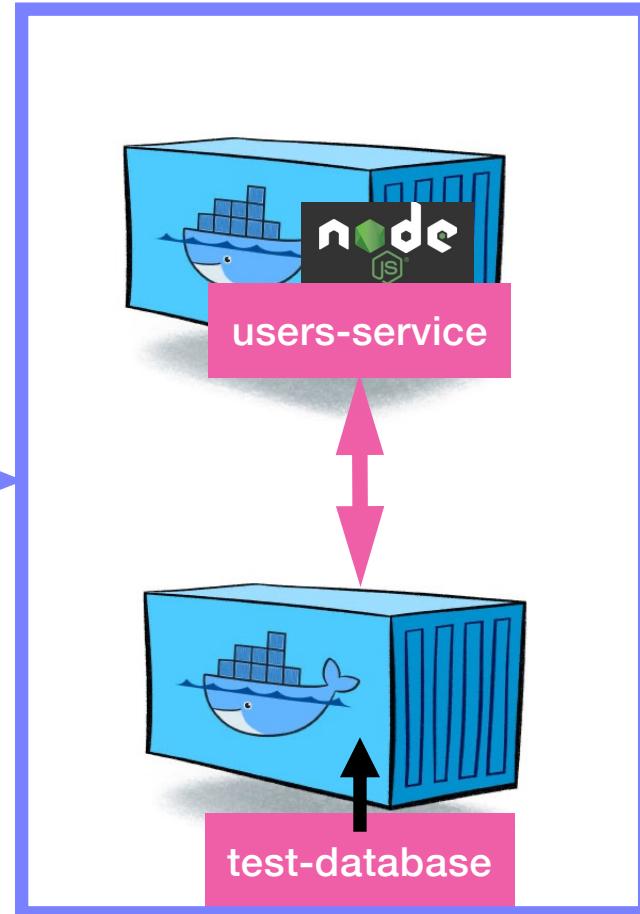
Try<http://localhost:6001/search?email=homer@thesimpsons.com>**ผลที่ได้**

```
1 // 20200313203245
2 // http://localhost:6001/search?email=homer@thesimpsons.com
3
4 {
5     "email": "homer@thesimpsons.com",
6     "phoneNumber": "+1 888 123 1111"
7 }
```

Local machine



Local machine



Try not using --link

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

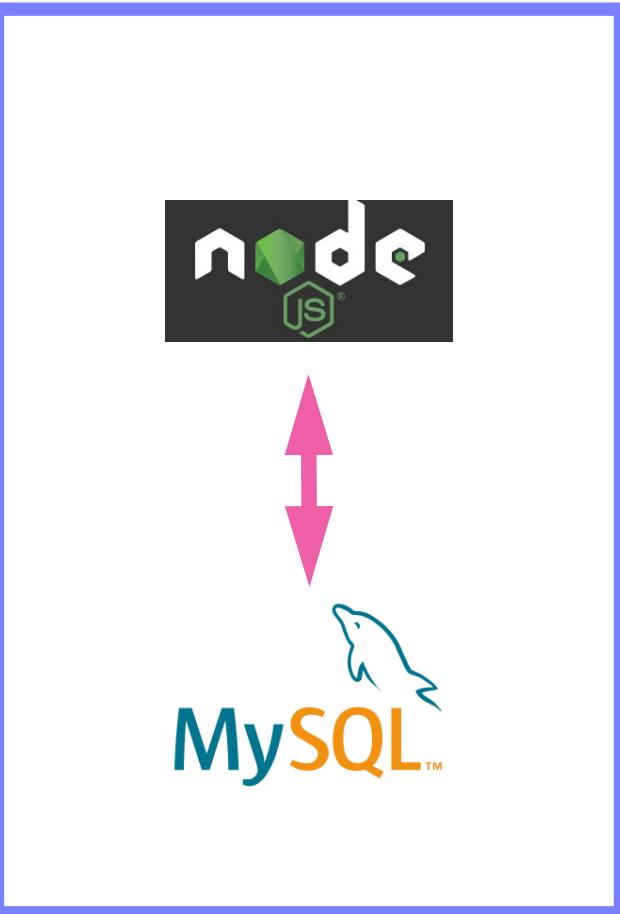
```
docker run -p 6002:3306 -v docker_practice:/var/lib/mysql --name db1 -h host.docker.internal -it users_service_db
```

- Start the users_service container using the following command:

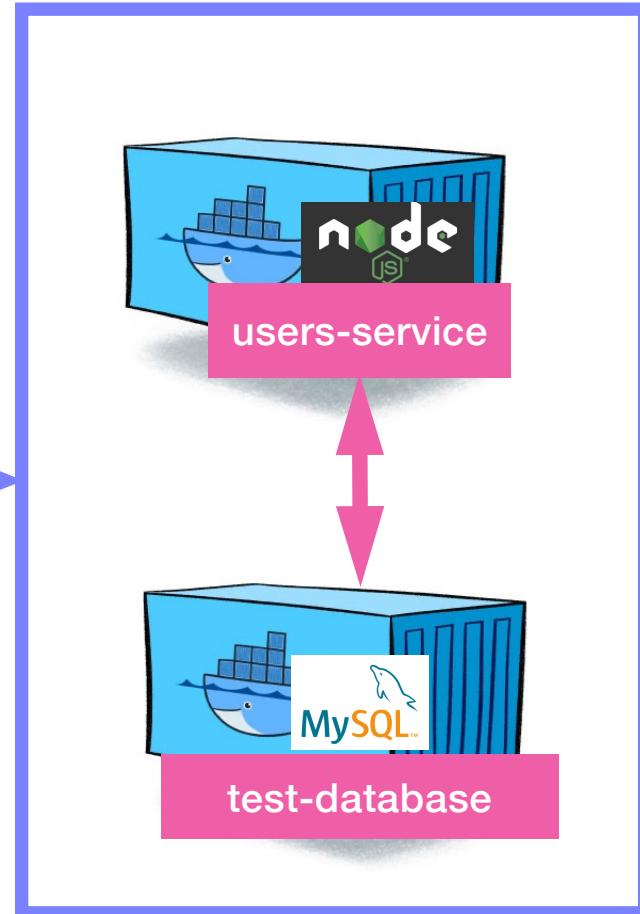
```
docker run -it --net=my-net -p 6001:6001 -e DATABASE_HOST=host.docker.internal --name=users-service users-service
```

Then, it should work without --link.

Local machine



Local machine



Composing

Building and running each container is still somewhat time consuming.

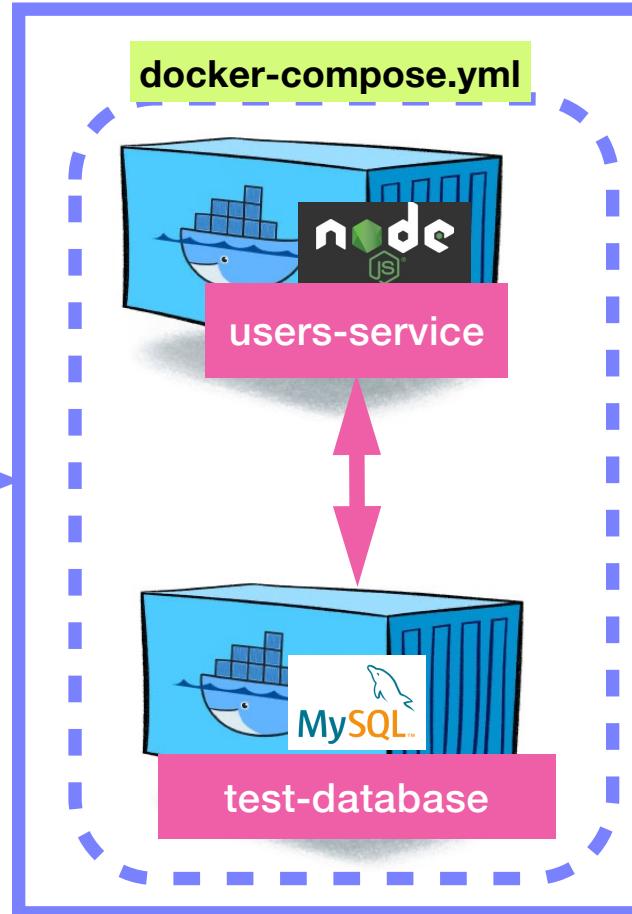
Docker Compose lets you create a file which defines each container in your system, the relationships between them, and build or run them all.

To do this, we need Docker Compose.

Local machine



Local machine



```
1 version: '2'  
2 services:  
3   db:  
4     image: users_service_db  
5     build: ./test-database  
6     container_name: db1  
7     expose:  
8       - "3306"  
9     ports:  
10      - "6002:3306"  
11     healthcheck:  
12       test: ["CMD", "mysqladmin" , "ping", "-h", "localhost"]  
13       timeout: 20s  
14       retries: 10  
15   users-service:  
16     restart: always  
17     depends_on:  
18       db:  
19         condition: service_healthy  
20     build: ./users-service  
21     image: users-service  
22     container_name: users-service  
23     ports:  
24       - "6001:6001"  
25     links:  
26       - db:db  
27     environment:  
28       - DATABASE HOST=db
```

Where to find the Dockerfile

Where to find the Dockerfile

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

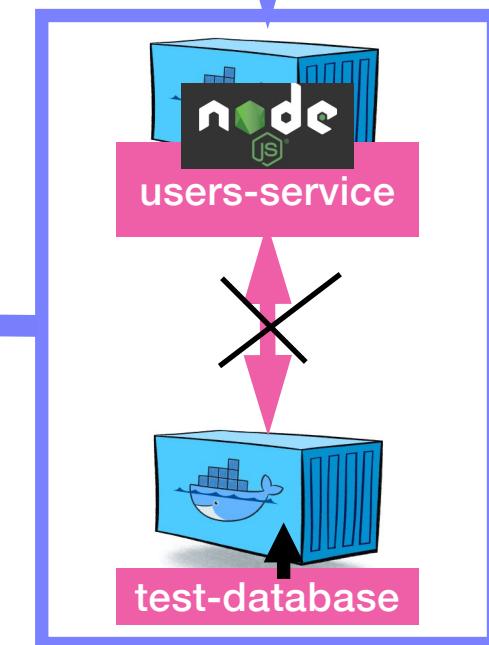
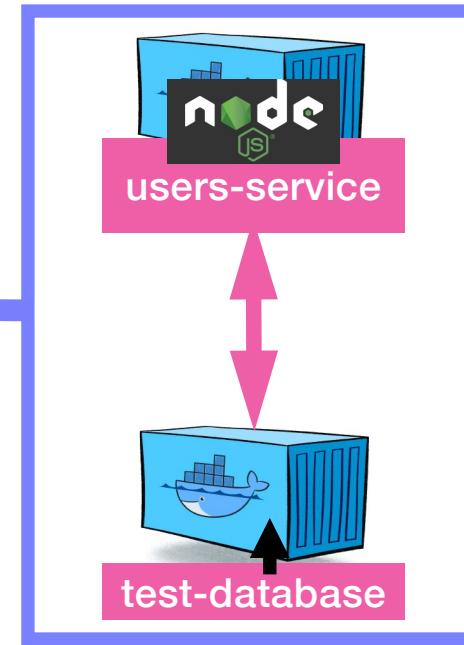
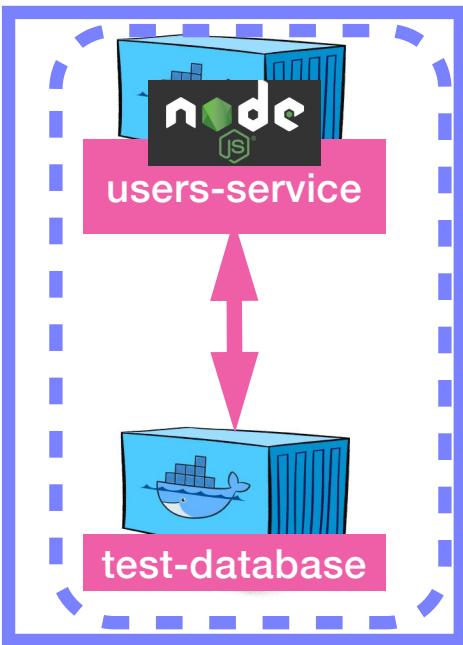
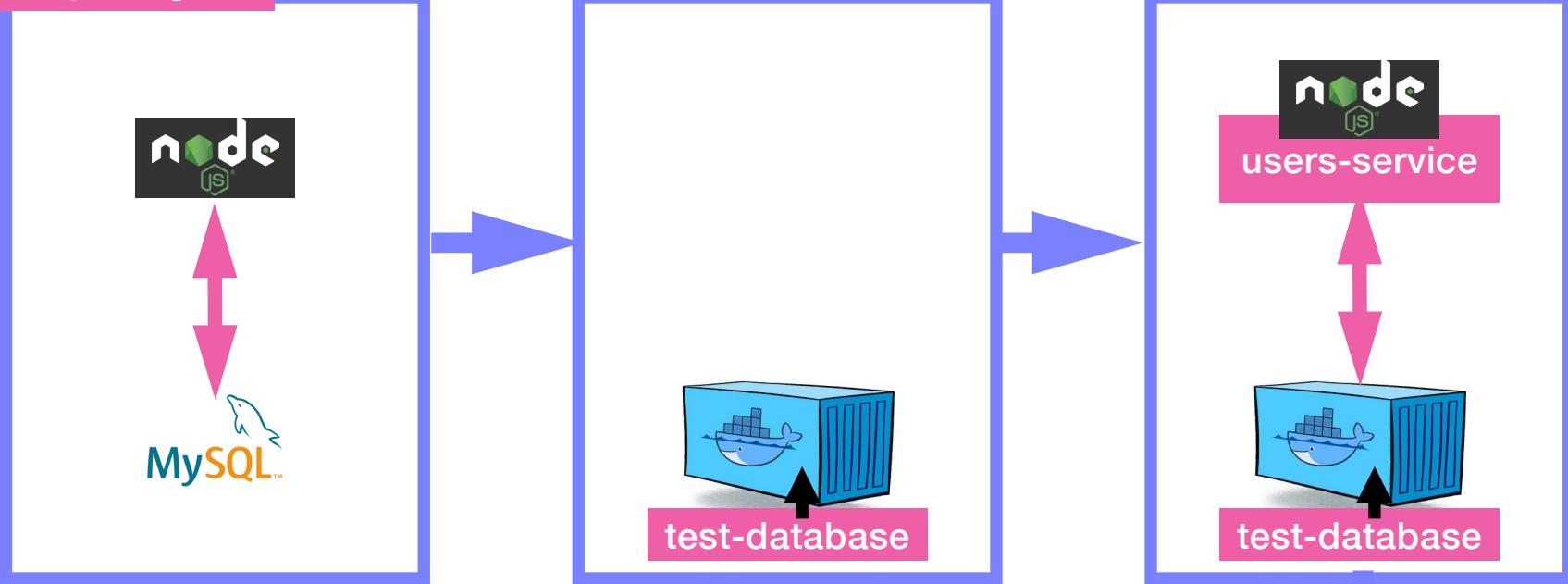
docker network connect my-net users-service

Try

<http://localhost:6001/users>

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

Composing



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.