Docker File

Next step to use Docker

Dockerfile

- Docker สร้าง image ขึ้นมาเพื่อห่อสิ่งที่จำเป็นสำหรับการ run application ไว้ภายใน ไม่ว่าจะเป็น Linux Kernel, network layer รวมไปถึง dependency ต่างๆ จากนั้น docker engine จะนำ image ไปสร้างเป็น container ตอน runtime

- Docker image ถูกสร้างได้จาก command ผ่าน docker engine หรือผ่านทาง Dockerfile

- ใน Dockerfile จะมี script, ชุดของ command ต่างๆ ของ docker และบน linux รวมไปถึง argument และ parameter ต่างๆ เพื่อสร้าง docker image อัตโนมัติโดยที่ developer หรือ operation ไม่จำเป็นต้องสร้าง image ทีละคำสั่ง

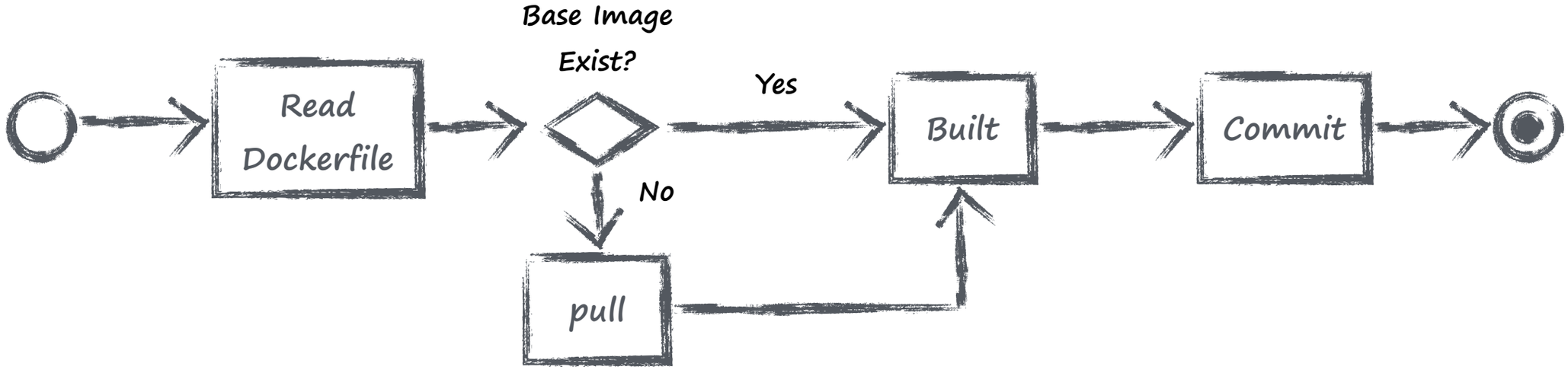
- Dockerfile เป็น text file ธรรมดา โดยชื่อไฟล์จะต้องเป็นคำว่า “Dockerfile” (ไม่มีนามสกุล) การนำไปสร้างเป็น image จะใช้คำสั่ง docker built ใน folder ที่เก็บ Dockerfile ไว้ เช่น

```
$ docker build –t name .
```

note1: -t เป็นการกำหนดชื่อของ image ที่จะถูกสร้างขึ้นมา
note2: มีจุด (.) ต่อท้าย name ด้วย

Usage Dockerfile

$ docker built -t my_Image .



Dockerfile Example

```
FROM ubuntu:14.04

MAINTAINER super admin

RUN apt-get update && apt-get install -y
nodejs npm

ADD ./src/ /src

RUN cd /src; npm install

CMD ["node", "/src/index.js"]
```

Dockerfile Syntax

- Syntax ของ Dockerfile มี 2 แบบ คือ instruction และ instruction + arguments เช่น

```
FROM ubuntu:latest
RUN apt-get update
```

- Instruction จะเป็นตัวใหญ่ทั้งหมด (Capital Letter)
- Comment

```
# comment message
```

Docker Commands (1/11)

- ADD

  เป็นคำสั่งที่ใช้เพื่อเพิ่มไฟล์เข้าไปใน container มี argument 2 ตัว คือ source และ destination ถ้า source เป็น URL ไฟล์ที่อยู่ที่ server (ตาม url) จะถูก download และ save ไปยัง destination

```
# Usage: ADD [source directory or URL] [destination directory]
```

เช่น

```
ADD /home/admin/my_app_folder /my_app_folder
```

Docker Commands (2/11)

- CMD

  เป็นคำสั่งที่สั่ง execute คำสั่งของ Linux ใน container (ขึ้นอยู่กับว่าเป็น Linux อะไร เช่น Fedora หรือ Debien) แต่จะไม่ได้ทำงานตอน built แต่จะถูกเรียกใช้ตอน container เริ่มทำงาน

  ```
  Usage 1: CMD application "argument", "argument", ..
  ```

  เช่น

  ```
  CMD "echo" "Hello docker!"
  ```

Docker Commands (3/11)

- ENTRYPOINT

  เป็นการกำหนด default application ให้กับ container โดยทุกครั้งที่ container เริ่มทำงาน ก็ให้ไปเรียก app นี้มาทำงานเสมอ เหมาะกับการสร้าง container ที่ทำงานเฉพาะอย่าง เช่น AppServer หรือ Database เป็นต้น

```
# Usage: ENTRYPOINT application "argument", "argument", ..
```

เช่น

```
ENTRYPOINT node
```

การทำงานของ ENTRYPOINT จะคล้ายกับ CMD เราสามารถใช้ร่วมกับ CMD ได้ คือ กำหนด app ที่ ENTRYPOINT แต่กำหนด argument ที่ CMD เช่น

```
CMD “server.js”
ENTRYPOINT node
```

Docker Commands (4/11)

- ENV

  เป็นการกำหนด environment variable เพื่อเอาไปใช้ภายใน Dockerfile การกำหนด variable จะเป็นแบบ key = value แต่ไม่ต้องใส่เครื่องหมาย “=”

```
# Usage: ENV key value
```

เช่น

```
ENV BUILD_SCRIPT http://myserver.com/script.sh
RUN wget BUILD_SCRIPT
```

Docker Commands (5/11)

- EXPOSE

  ใช้เพื่อกำหนด port เพื่อให้ app ภายนอก container สามารถติดต่อกับ app ที่อยู่ภายใน container ได้

```
# Usage: EXPOSE [port]
```

เช่น

```
EXPOSE 3000
```

Docker Commands (6/11)

- FROM

  เป็นคำสั่งสำคัญที่จะต้องกำหนดไว้ใน Dockerfile ตั้งแต่ต้น เพื่อระบุว่าเราจะใช้ base image ตัวไหนมา built image ใหม่ อาจจะเป็น image ที่มีอยู่แล้วที่ local, image ที่เรา build เองก่อนหน้านี้ หรือบน docker hub ก็ได้

```
# Usage: FROM [image name] [:tag]
```

เช่น

```
FROM ubuntu:14.04
```

Docker Commands (7/11)

- MAINTAINER

  เป็นคำสั่งที่ไม่ได้ถูก execute วางไว้จุดไหนของไฟล์ก็ได้
  ใช้เพื่อบอกว่า ใครเป็นเจ้าของไฟล์นี้

```
# Usage: MAINTAINER [name]
```

เช่น

```
MAINTAINER Steve Jobs
```

Docker Commands (8/11)

- RUN

  เป็นคำสั่งที่ใช้สั่งให้ container ทำงานในระหว่าง built ซึ่งจะเป็นคำสั่งต่างๆ บน Linux หรือการเรียก app ทำงาน คำสั่ง RUN จะต่างกับ CMD คือ RUN จะทำงานระหว่าง built เลย แต่ CMD จะทำงานตอน container เริ่มทำงาน

```
# Usage: RUN [command]
```

เช่น

```
RUN apt-get update && apt-get upgrade -y
```

Docker Commands (9/11)

- USER

  กำหนด user ให้กับ process ที่ทำงานใน container แทนที่จะเป็น root

```
# Usage: USER [UID or NAME]
```

เช่น

```
USER 751
USER jenkins
```

Docker Commands (10/11)

- VOLUME

  เป็นคำสั่งที่กำหนด share folder จาก host OS ไปยัง container

```
# Usage: VOLUME ["/dir_1", "/dir_2" ..]
```

เช่น

```
VOLUME ["/my_files"]
```

Docker Commands (11/11)

- WORKDIR

  เป็นคำสั่งที่ใช้กำหนด folder ที่เราต้องการให้ container ทำงานระหว่าง built

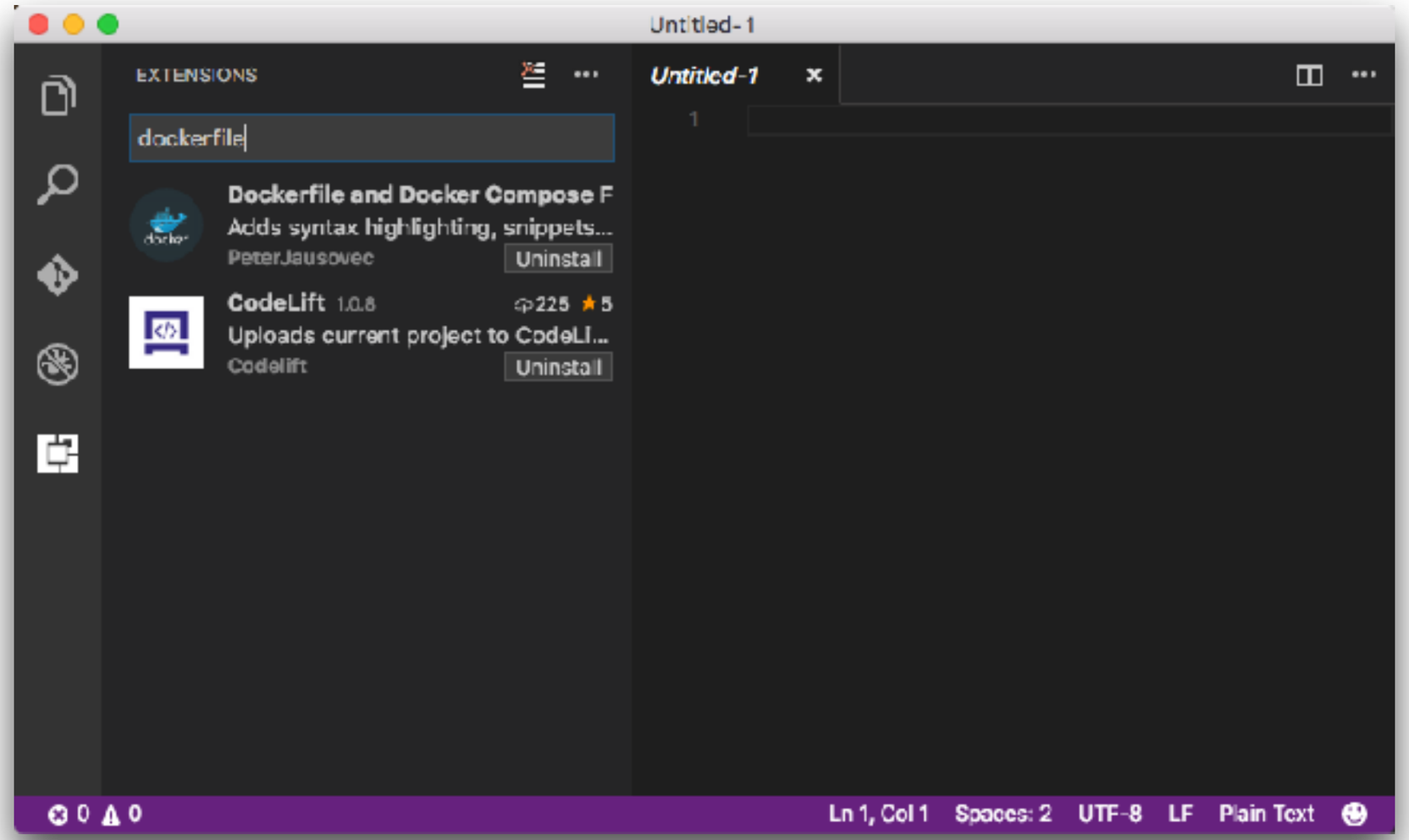
```
# Usage: WORKDIR /path
```

เช่น

```
WORKDIR /home/app
```

VS Code (optional)

- ติดตั้ง Visual Studio Code
- ติดตั้ง Dockerfile Syntax Highlighting



Lab: Dockerfile (1/3)

1. สร้าง folder ชื่อ src แล้ว
2. สร้างไฟล์ชื่อ package.json ใน folder src แล้วเพิ่มรายละเอียดดังนี้

```
{
  "name": "docker",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "express": "^4.14.0"
  }
}
```

Lab: Dockerfile (2/3)

3. สร้างไฟล์ชื่อ index.js ใน folder src แล้วเพิ่ม code ดังนี้

```
var express = require('express');

var PORT = 3000;

var app = express();
app.get('/', function (req, res) {
  res.send('Hello world\n');
});

app.listen(PORT);
console.log('Running on http://localhost:' + PORT);
```

4. สร้างไฟล์ชื่อ Dockerfile (ไม่มีนามสกุล) แล้วเพิ่ม code ดังนี้

```
FROM node
MAINTAINER super admin

ADD . /src
WORKDIR /src
RUN npm install
EXPOSE 3000

CMD ["node", "index.js"]
```

Lab: Dockerfile (3/3)

5. run คำสั่งเพื่อสร้าง docker image (มีจุดด้วย)

   ```
   docker build -t mynode .
   ```

6. สั่ง docker ให้สร้าง container จาก image

   ```
   docker run -dt --name mynode  -p 3000:3000 mynode
   ```

7. browser ไปที่ server ผ่าน browser

   ```
   http://localhost:3000
   ```

Exercise

1. สร้าง Dockerfile เพื่อติดตั้ง “MySql Server” จาก Ubuntu ชื่อ mySqlServer

2. Run Container โดยตั้งชื่อ container ว่า mySqlServer โดย mount volume ของ database มาไว้ที่ host

3. สร้าง Dockerfile เพื่อติดตั้ง “MySql Client” จาก Ubuntu ชื่อ mySqlClient

4. Run Container โดยตั้งชื่อ container ว่า mySqlClient

5. exec จาก container mySqlClient เพื่อ access mySqlServer

MySQL Note

MySQL จะไม่เปิดให้ remote access โดย default วิธีแก้ คือ

1. ใช้ text editor เปิดไฟล์ `/etc/mysql/mysql.conf.d/mysqld.cnf`

2. แก้ค่า bind-address เป็น 0.0.0.0

3. login จาก local

```
$ mysql --user=root --password=<password_จากตอน install>
> GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' IDENTIFIED BY 'password' WITH GRANT OPTION;
> FLUSH PRIVILEGES;
```

4. Restart MySQL Service

```
$ service mysql restart
```

5. connect จาก mysql client ด้วยคำสั่ง

```
$ mysql --host=<mySqlServer_IP> --user=root --password=<password>
```