

## Lab Worksheet

ชื่อ-นามสกุล วชิรัฐพล พันชนกกุล รหัสนักศึกษา 6533800230 Section 2

## Lab#8 – Software Deployment Using Docker

## วัตถุประสงค์การเรียนรู้

1. ผู้เรียนสามารถอธิบายเกี่ยวกับ Software deployment ได้
2. ผู้เรียนสามารถสร้างและรัน Container จาก Docker image ได้
3. ผู้เรียนสามารถสร้าง Docker files และ Docker images ได้
4. ผู้เรียนสามารถนำซอฟต์แวร์ที่พัฒนาขึ้นให้สามารถรันบนสภาพแวดล้อมเดียวกันและทำงานร่วมกันกับสมาชิกในทีมพัฒนาซอฟต์แวร์ผ่าน Docker hub ได้
5. ผู้เรียนสามารถเริ่มต้นใช้งาน Jenkins เพื่อสร้าง Pipeline ในการ Deploy งานได้

## Pre-requisite

1. ติดตั้ง Docker desktop ลงบนเครื่องคอมพิวเตอร์ โดยดาวน์โหลดจาก <https://www.docker.com/get-started>
2. สร้าง Account บน Docker hub (<https://hub.docker.com/signup>)
3. กำหนดให้ \$ หมายถึง Command prompt และ <> หมายถึง ให้ป้อนค่าของพารามิเตอร์ที่กำหนด

## แบบฝึกปฏิบัติที่ 8.1 Hello world - รัน Container จาก Docker image

1. เปิดใช้งาน Docker desktop และ Login ด้วย Username และ Password ที่ลงทะเบียนกับ Docker Hub เอาไว้
1. เปิด Command line หรือ Terminal บน Docker Desktop จากนั้นสร้าง Directory ชื่อ Lab8\_1
2. ย้ายตำแหน่งปัจจุบันไปที่ Lab8\_1 เพื่อใช้เป็น Working directory
3. ป้อนคำสั่ง \$ docker pull busybox หรือ \$ sudo docker pull busybox สำหรับกรณีที่ติดปัญหา Permission denied  
(หมายเหตุ: BusyBox เป็น software suite ที่รองรับคำสั่งบางอย่างบน Unix - <https://busybox.net>)
4. ป้อนคำสั่ง \$ docker images

## Lab Worksheet

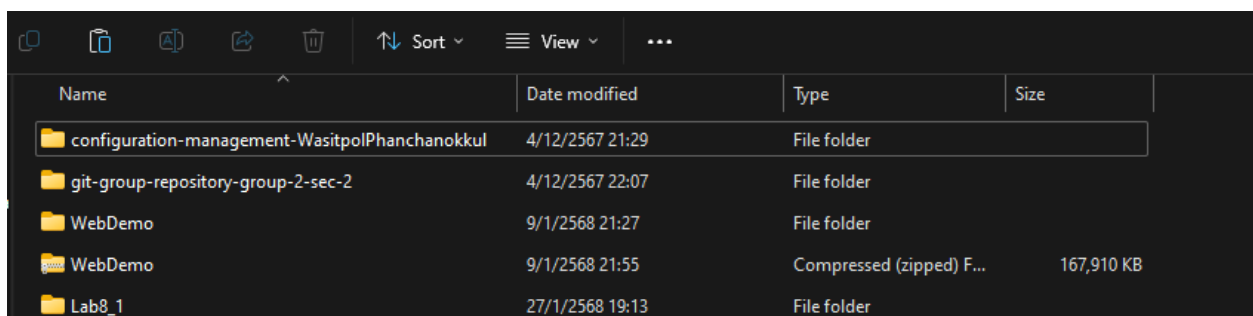
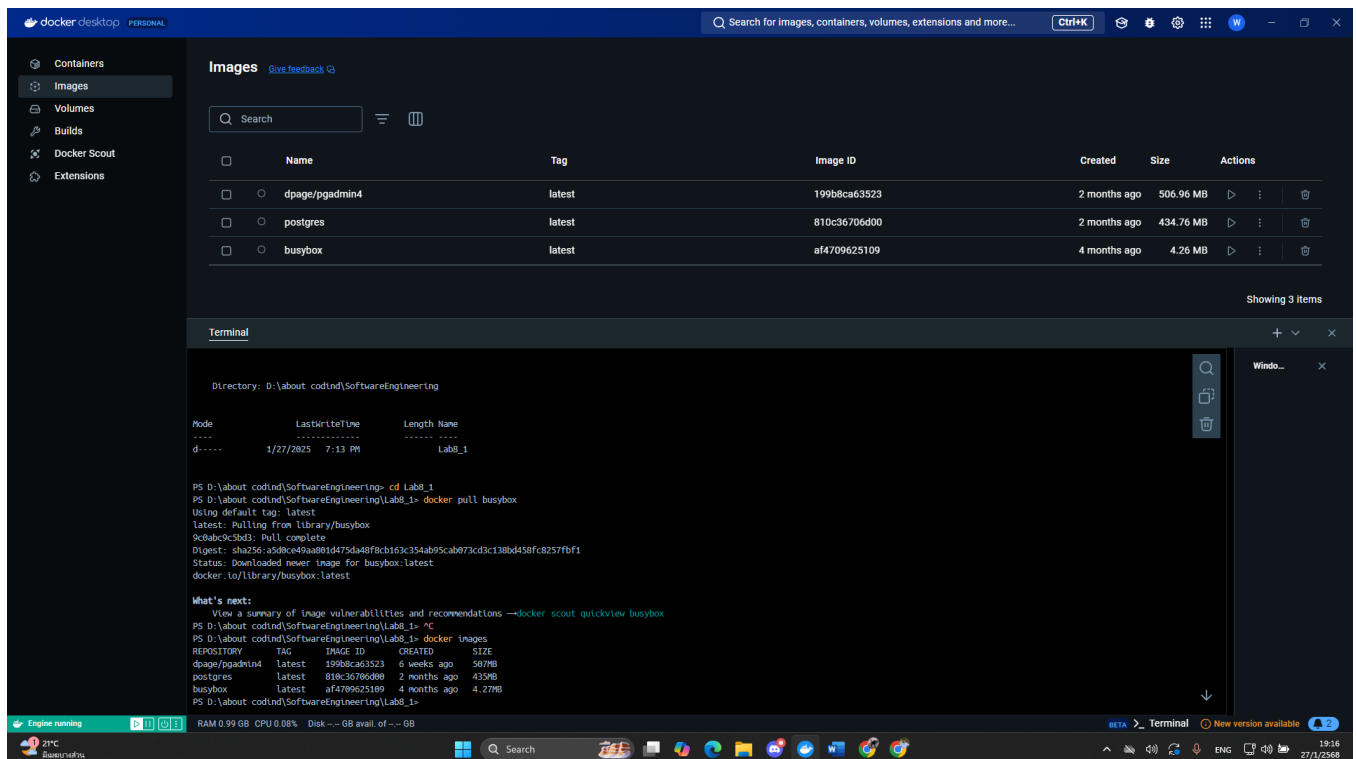
[Check point#1] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงผลลัพธ์ที่ได้ พร้อมคำตอบคำถามต่อไปนี้

(1) สิ่งที่อยู่ภายใต้คอลัมน์ Repository คืออะไร

ตอบ ชื่อของ Docker Image ในที่นี้คือ busybox

(2) Tag ที่ใช้นับบอกถึงอะไร

ตอบ เวอร์ชันเฉพาะของ Docker Image ในที่นี้คือ version ล่าสุด (latest)



<input type="checkbox"/>	Name	Tag	Image ID	Created	Size	Actions
<input type="checkbox"/>	dpape/pgadmin4	latest	199b8ca63523	2 months ago	506.96 MB	
<input type="checkbox"/>	postgres	latest	810c36706d00	2 months ago	434.76 MB	
<input type="checkbox"/>	busybox	latest	af4709625109	4 months ago	4.26 MB	

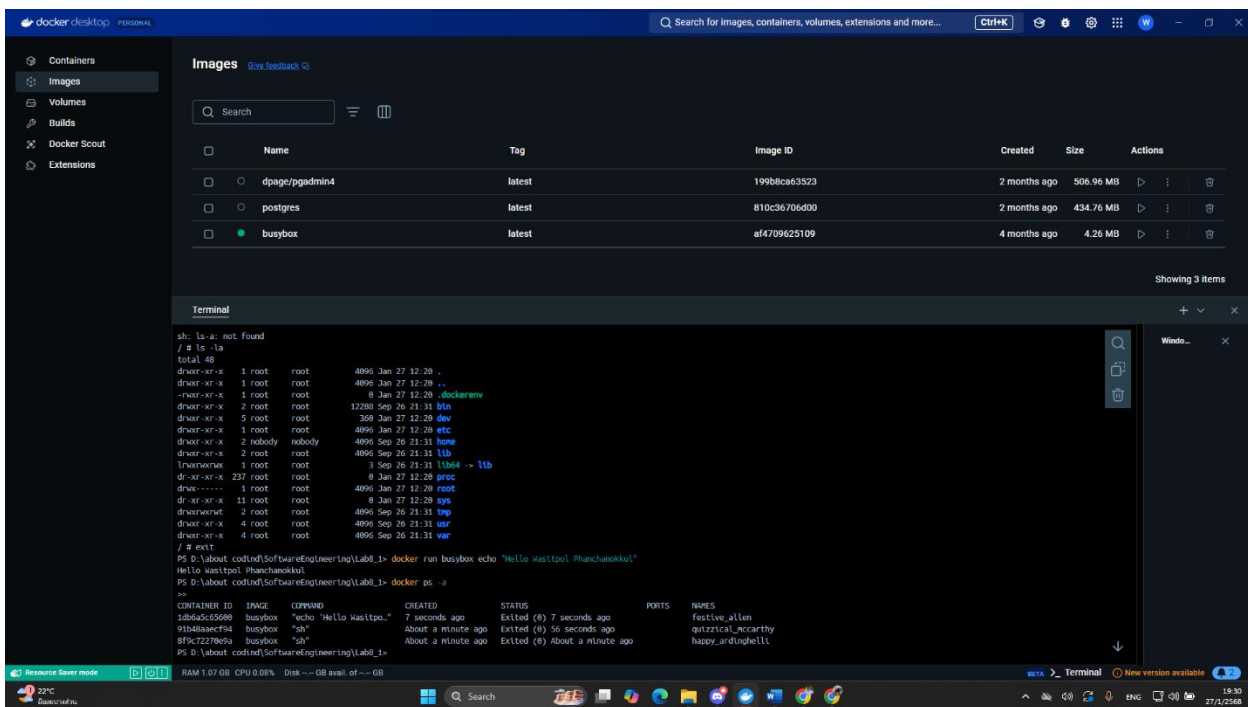
## Lab Worksheet

5. ป้อนคำสั่ง \$ docker run busybox
6. ป้อนคำสั่ง \$ docker run -it busybox sh
7. ป้อนคำสั่ง ls
8. ป้อนคำสั่ง ls -la
9. ป้อนคำสั่ง exit
10. ป้อนคำสั่ง \$ docker run busybox echo "Hello ชื่อและนามสกุลของนักศึกษา from busybox"
11. ป้อนคำสั่ง \$ docker ps -a

[Check point#2] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงผลลัพธ์ที่ได้ตั้งแต่ขั้นตอนที่ 6-12 พร้อมกับตอบคำถามต่อไปนี้

(1) เมื่อใช้ option -it ในคำสั่ง run ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป  
ตอบ ทำการเข้าไปใน shell เพื่อที่จะใช้ command ใน container ได้

(2) คอลัมน์ STATUS จากการรันคำสั่ง docker ps -a แสดงถึงข้อมูลอะไร  
ตอบ แสดงสถานะของแต่ละ container ว่าทำอะไรไปบ้าง



## Lab Worksheet

```

PS D:\about codind\SoftwareEngineering\Lab8_1> docker run busybox
PS D:\about codind\SoftwareEngineering\Lab8_1> docker run -it busybox sh
/ # ls
bin      dev      etc      home     lib      lib64    proc     root     sys      tmp      usr      var
/ # ls -a
sh: ls-a: not found
/ # ls -la
total 48
drwxr-xr-x  1 root    root      4096 Jan 27 12:20 .
drwxr-xr-x  1 root    root      4096 Jan 27 12:20 ..
-rwxr-xr-x  1 root    root        0 Jan 27 12:20 .dockerenv
drwxr-xr-x  2 root    root     12288 Sep 26 21:31 bin
drwxr-xr-x  5 root    root      360 Jan 27 12:20 dev
drwxr-xr-x  1 root    root      4096 Jan 27 12:20 etc
drwxr-xr-x  2 nobody  nobody    4096 Sep 26 21:31 home
drwxr-xr-x  2 root    root      4096 Sep 26 21:31 lib
lrwxrwxrwx  1 root    root        3 Sep 26 21:31 lib64 -> lib
dr-xr-xr-x 237 root    root        0 Jan 27 12:20 proc
drwx----- 1 root    root      4096 Jan 27 12:20 root
dr-xr-xr-x 11 root    root        0 Jan 27 12:20 sys
drwxrwxrwt  2 root    root      4096 Sep 26 21:31 tmp
drwxr-xr-x  4 root    root      4096 Sep 26 21:31 usr
drwxr-xr-x  4 root    root      4096 Sep 26 21:31 var
/ # exit
PS D:\about codind\SoftwareEngineering\Lab8_1>

```

RAM 1.07 GB CPU 0.08% Disk == GB avail of == GB

```

/ # exit
PS D:\about codind\SoftwareEngineering\Lab8_1> docker run busybox echo "Hello Wasitpol Phanchanokkul"
Hello Wasitpol Phanchanokkul
PS D:\about codind\SoftwareEngineering\Lab8_1> docker ps -a
>>

```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
1db6a5c65600	busybox	"echo 'Hello Wasitpo..."	7 seconds ago	Exited (0) 7 seconds ago		festive_allen
91b48aaecf94	busybox	"sh"	About a minute ago	Exited (0) 56 seconds ago		quizzical_mccarthy
8f9c72270e9a	busybox	"sh"	About a minute ago	Exited (0) About a minute ago		happy_ardinghelli

12. ป้อนคำสั่ง \$ docker rm <container ID ที่ต้องการลบ>

[Check point#3] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงผลลัพธ์ที่ได้ในขั้นตอนที่ 13

```

PS D:\about codind\SoftwareEngineering\Lab8_1> docker ps -a
>>

```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
1db6a5c65600	busybox	"echo 'Hello Wasitpo..."	7 seconds ago	Exited (0) 7 seconds ago		festive_allen
91b48aaecf94	busybox	"sh"	About a minute ago	Exited (0) 56 seconds ago		quizzical_mccarthy
8f9c72270e9a	busybox	"sh"	About a minute ago	Exited (0) About a minute ago		happy_ardinghelli

```

PS D:\about codind\SoftwareEngineering\Lab8_1> docker rm 9c0abc9c5bd3
Error response from daemon: No such container: 9c0abc9c5bd3
PS D:\about codind\SoftwareEngineering\Lab8_1> ^C
PS D:\about codind\SoftwareEngineering\Lab8_1> docker rm 1db6a5c65600
1db6a5c65600
PS D:\about codind\SoftwareEngineering\Lab8_1>

```

## Lab Worksheet

## แบบฝึกปฏิบัติที่ 8.2: สร้าง Docker file และ Docker image

1. เปิดใช้งาน Docker desktop และ Login ด้วย Username และ Password ที่ลงทะเบียนกับ Docker Hub เอาไว้
2. เปิด Command line หรือ Terminal จากนั้นสร้าง Directory ชื่อ Lab8\_2
3. ย้ายตำแหน่งปัจจุบันไปที่ Lab8\_2 เพื่อใช้เป็น Working directory
4. สร้าง Dockerfile.swp ไว้ใน Working directory

สำหรับเครื่องที่ใช้ระบบปฏิบัติการวินโดวส์ (Windows) บันทึกคำสั่งต่อไปนี้ลงในไฟล์ โดยใช้ Text Editor ที่มี

```
FROM busybox
```

```
CMD echo "Hi there. This is my first docker image."
```

```
CMD echo "ชื่อ-นามสกุล รหัสนักศึกษา ชื่อเล่น"
```

สำหรับเครื่องที่ใช้ระบบปฏิบัติการ MacOS หรือ Linux บนหน้าต่าง Terminal และบันทึกคำสั่งต่อไปนี้

```
$ cat > Dockerfile << EOF
```

```
FROM busybox
```

```
CMD echo "Hi there. This is my first docker image."
```

```
CMD echo "ชื่อ-นามสกุล รหัสนักศึกษา ชื่อเล่น"
```

```
EOF
```

หรือใช้คำสั่ง

```
$ touch Dockerfile
```

แล้วใช้ Text Editor ในการใส่เนื้อหาแทน

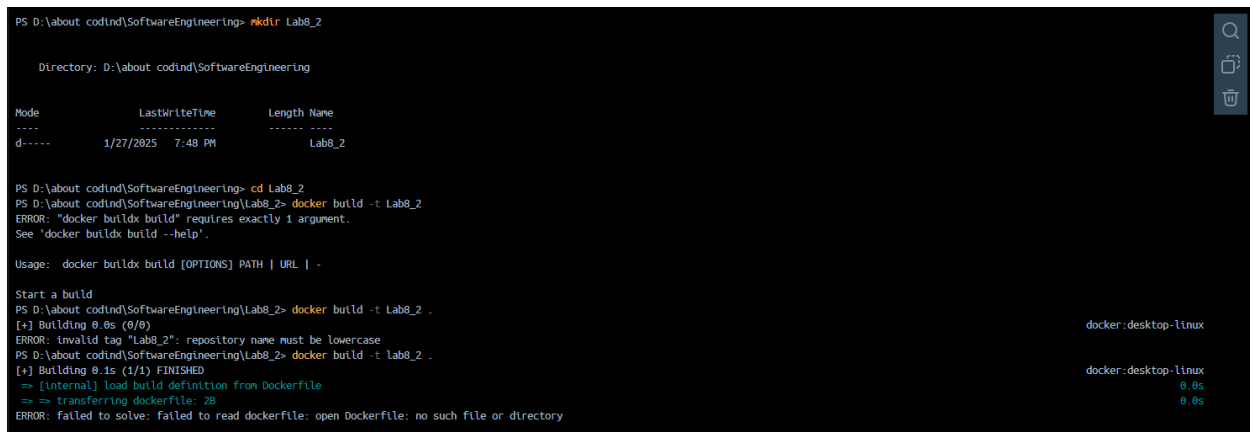
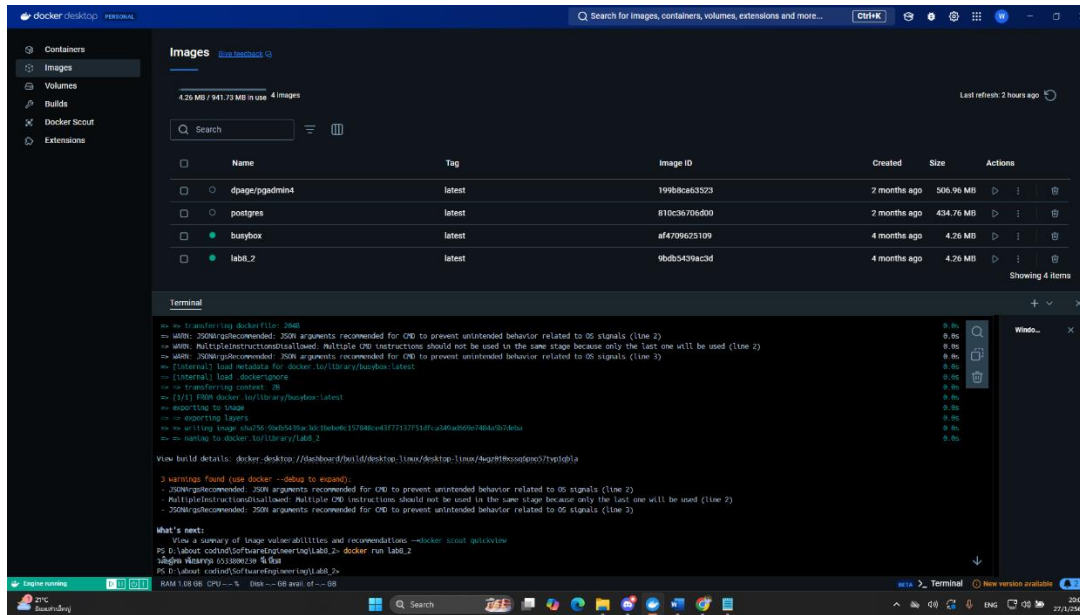
5. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้

```
$ docker build -t <ชื่อ Image> .
```

6. เมื่อ Build สำเร็จแล้ว ให้ทำการรัน Docker image ที่สร้างขึ้นในขั้นตอนที่ 5

## Lab Worksheet

[Check point#4] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงผลลัพธ์ที่ได้ในขั้นตอนที่ 5 พร้อมกับตอบคำถามต่อไปนี้



## Lab Worksheet

```

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/ghr1ux9s68a8j0kc4b3v2vnow
PS D:\about codind\SoftwareEngineering\Lab8_2> docker build -t lab8_2 .
>>
=> => transferring dockerfile: 284B                                0.0s
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 2) 0.0s
=> WARN: MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the same stage because only the last one will be used (line 2) 0.0s
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 3) 0.0s
=> [internal] load metadata for docker.io/library/busybox:latest      0.0s
=> [internal] load .dockerignore                                       0.0s
=> => transferring context: 2B                                         0.0s
=> [1/1] FROM docker.io/library/busybox:latest                       0.0s
=> => exporting to image                                              0.0s
=> => exporting layers                                              0.0s
=> => writing image sha256:9bdb5439ac3dc1bebe0c157848ce43f77137f51dfca349ad669e7484a5b7deba 0.0s
=> => naming to docker.io/library/lab8_2                             0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/4wqz818xssg6pno57vpj0b1a

3 warnings found (use docker --debug to expand):
- JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 2)
- MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the same stage because only the last one will be used (line 2)
- JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 3)

What's next:
View a summary of image vulnerabilities and recommendations →docker scout quickview
PS D:\about codind\SoftwareEngineering\Lab8_2> docker run lab8_2
วลิษฐ์ผด พันธ์มรกต 6533899238 ฝึกฝึก
PS D:\about codind\SoftwareEngineering\Lab8_2>

```

(1) คำสั่งที่ใช้ในการ run คือ

ตอบ docker run lab8\_2

(2) Option -t ในคำสั่ง \$ docker build ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป

ตอบ ใช้ในการกำหนดชื่อให้กับ docker image ที่สร้างมาและทำการอ้างถึงได้ง่ายเมื่อเวลาที่จะใช้

### แบบฝึกปฏิบัติที่ 8.3: การแชร์ Docker image ผ่าน Docker Hub

1. เปิดใช้งาน Docker desktop และ Login ด้วย Username และ Password ที่ลงทะเบียนกับ Docker Hub เอาไว้
2. เปิด Command line หรือ Terminal จากนั้นสร้าง Directory ชื่อ Lab8\_3
3. ย้ายตำแหน่งปัจจุบันไปที่ Lab8\_3 เพื่อใช้เป็น Working directory
4. สร้าง Dockerfile.swp ไว้ใน Working directory

สำหรับเครื่องที่ใช้ระบบปฏิบัติการวินโดวส์ บันทึกคำสั่งต่อไปนี้ลงในไฟล์ โดยใช้ Text Editor ที่มี

FROM busybox

CMD echo "Hi there. My work is done. You can run them from my Docker image."

CMD echo "ชื่อ-นามสกุล รหัสนักศึกษา"

สำหรับเครื่องที่ใช้ระบบปฏิบัติการ MacOS หรือ Linux บนหน้าต่าง Terminal และป้อนคำสั่งต่อไปนี้

\$ cat > Dockerfile << EOF

## Lab Worksheet

FROM busybox

CMD echo "Hi there. My work is done. You can run them from my Docker image."

CMD echo "ชื่อ-นามสกุล รหัสนักศึกษา"

EOF

หรือใช้คำสั่ง

\$ touch Dockerfile

แล้วใช้ Text Editor ในการใส่เนื้อหาแทน

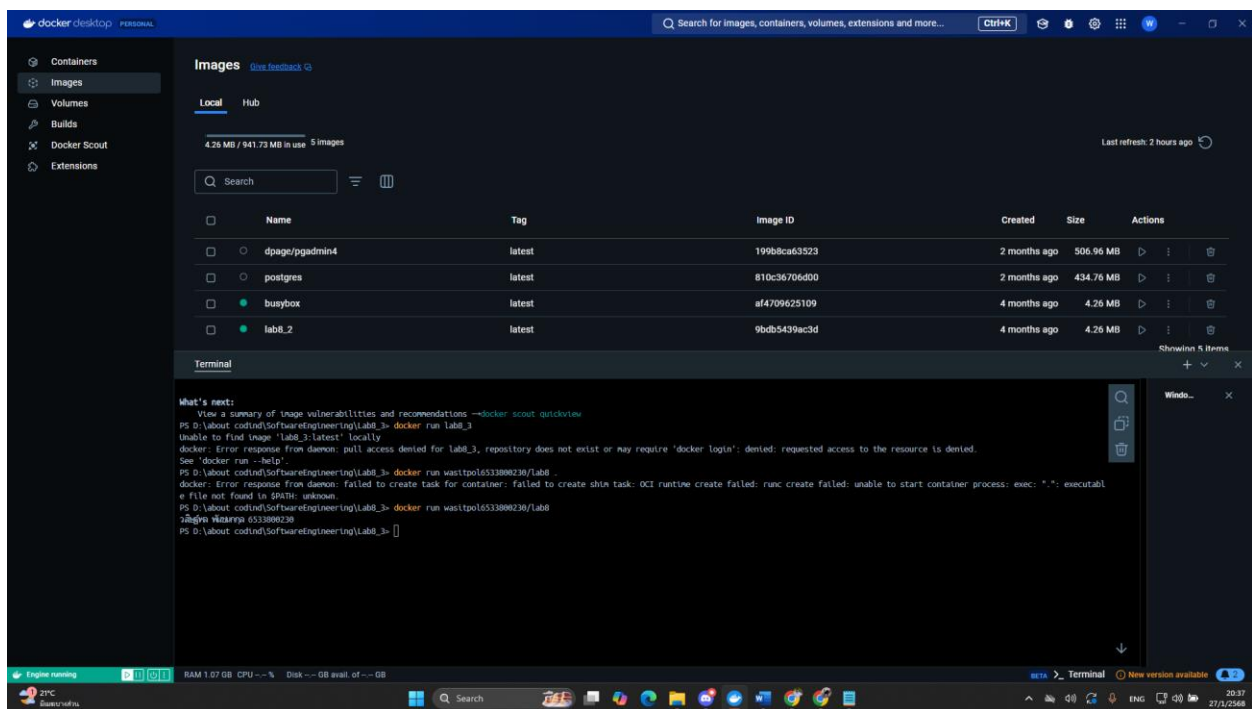
7. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้

\$ docker build -t <username ที่ลงทะเบียนกับ Docker Hub>/lab8

5. ทำการรัน Docker image บน Container ในเครื่องของตัวเองเพื่อทดสอบผลลัพธ์ ด้วยคำสั่ง

\$ docker run <username ที่ลงทะเบียนกับ Docker Hub>/lab8

[Check point#5] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงผลลัพธ์ที่ได้ในขั้นตอนที่ 5





## Lab Worksheet

```

ERROR: failed to solve: failed to read dockerfile: open Dockerfile: no such file or directory

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/v31mc186zgorzc892qfzn2k2k
PS D:\about codind\SoftwareEngineering\Lab8_3> docker build -t wasitpol6533800230/Lab8_3 .
>>
[+] Building 0.1s (1/1) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 2B
ERROR: failed to solve: failed to read dockerfile: open Dockerfile: no such file or directory

View build details: docker-desktop://dashboard/
                                ox4d1hnoo0w6z0Bd
PS D:\about codind\SoftwareEngineering\Lab8_3>
PS D:\about codind\SoftwareEngineering\Lab8_3>
PS D:\about codind\SoftwareEngineering\Lab8_3> docker build -t wasitpol6533800230/Lab8 .
[+] Building 0.1s (1/1) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 2B
ERROR: failed to solve: failed to read dockerfile: open Dockerfile: no such file or directory

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/gpp4r59en98thtqcdn8qlbalp

```

```

PS D:\about codind\SoftwareEngineering\Lab8_3> cd D:\about codind\SoftwareEngineering\Lab8_3
Set-Location : A positional parameter cannot be found that accepts argument 'codind\SoftwareEngineering\Lab8_3'.
At line:1 char:1
+ cd D:\about codind\SoftwareEngineering\Lab8_3
+ ~~~~~
+ CategoryInfo          : InvalidArgument: (:) [Set-Location], ParameterBindingException
+ FullyQualifiedErrorId : PositionalParameterNotFound,Microsoft.PowerShell.Commands.SetLocationCommand

PS D:\about codind\SoftwareEngineering\Lab8_3> ls

Directory: D:\about codind\SoftwareEngineering\Lab8_3

Mode                LastWriteTime         Length Name
----                -
-a----             1/27/2025   8:20 PM             168 Dockerfile.txt

PS D:\about codind\SoftwareEngineering\Lab8_3> docker build -t wasitpol6533800230/Lab8 .

What's next:
View a summary of image vulnerabilities and recommendations --> docker scout quickview
PS D:\about codind\SoftwareEngineering\Lab8_3> docker run Lab8_3
Unable to find image 'Lab8_3:latest' locally
docker: Error response from daemon: pull access denied for Lab8_3, repository does not exist or may require 'docker login': denied: requested access to the resource is denied.
See 'docker run --help'.

PS D:\about codind\SoftwareEngineering\Lab8_3> docker run wasitpol6533800230/Lab8 .
docker: Error response from daemon: failed to create task for container: failed to create shim task: OCI runtime create failed: runc create failed: unable to start container process: exec: ".": executable file not found in $PATH: unknown.
PS D:\about codind\SoftwareEngineering\Lab8_3> docker run wasitpol6533800230/Lab8
wasitpol6533800230

```

6. ทำการ Push ตัว Docker image ไปไว้บน Docker Hub โดยการใช้คำสั่ง

\$ docker push <username ที่ลงทะเบียนกับ Docker Hub>/lab8

ในกรณีที่ติดปัญหาไม่ได้ Login ไว้ก่อน ให้ใช้คำสั่งต่อไปนี้ เพื่อ Login ก่อนทำการ Push

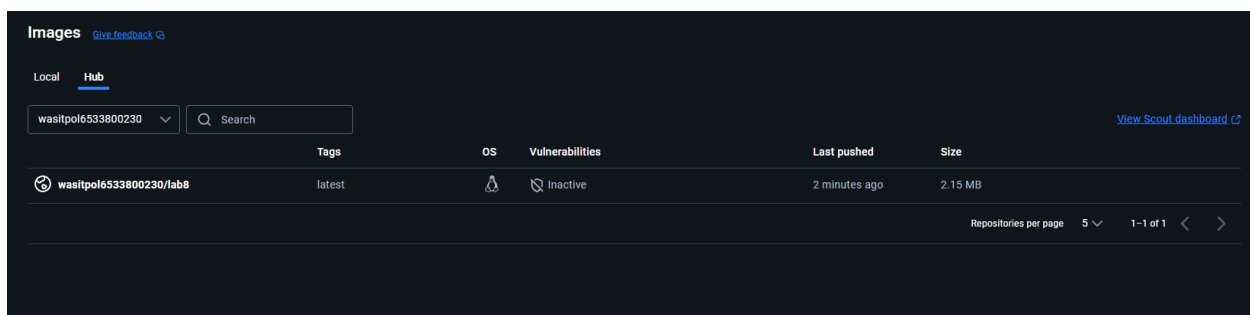
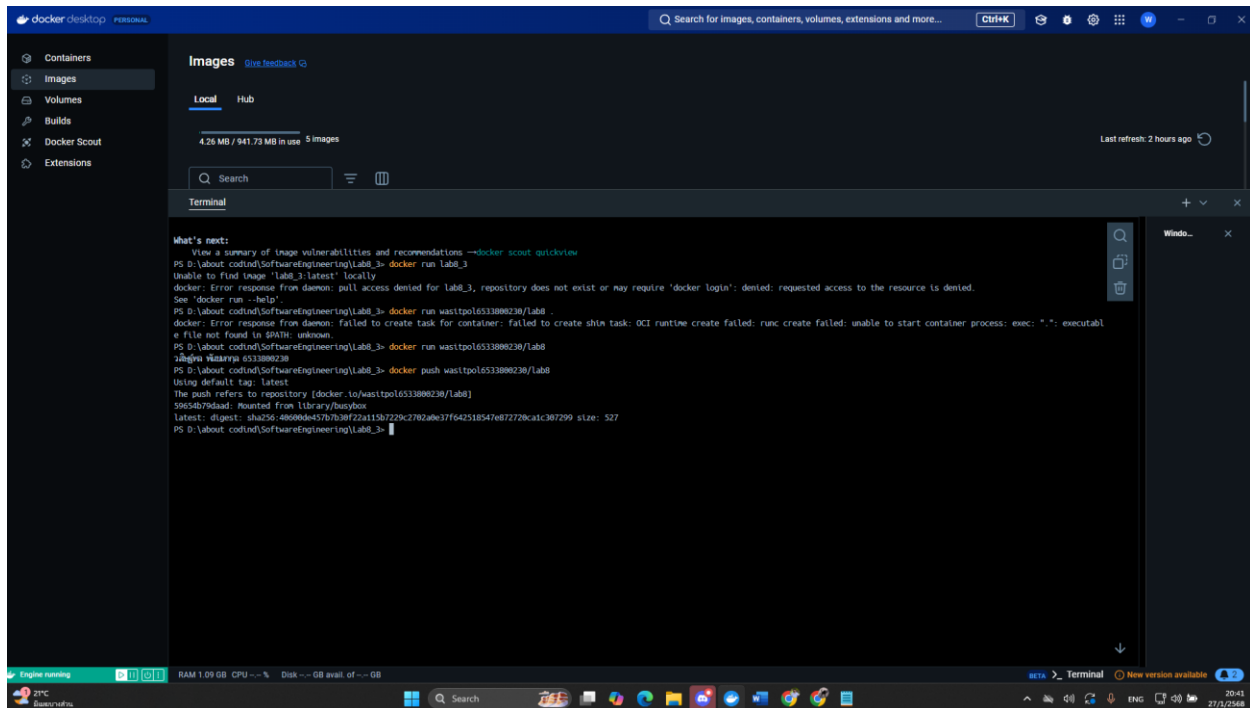
\$ docker login แล้วป้อน Username และ Password ตามที่ระบุใน Command prompt หรือใช้คำสั่ง

\$ docker login -u <username> -p <password>

7. ไปที่ Docker Hub กด Tab ชื่อ Tags หรือไปที่ Repository ก็ได้

## Lab Worksheet

[Check point#6] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดง Repository ที่มี Docker image (<username>/lab8)

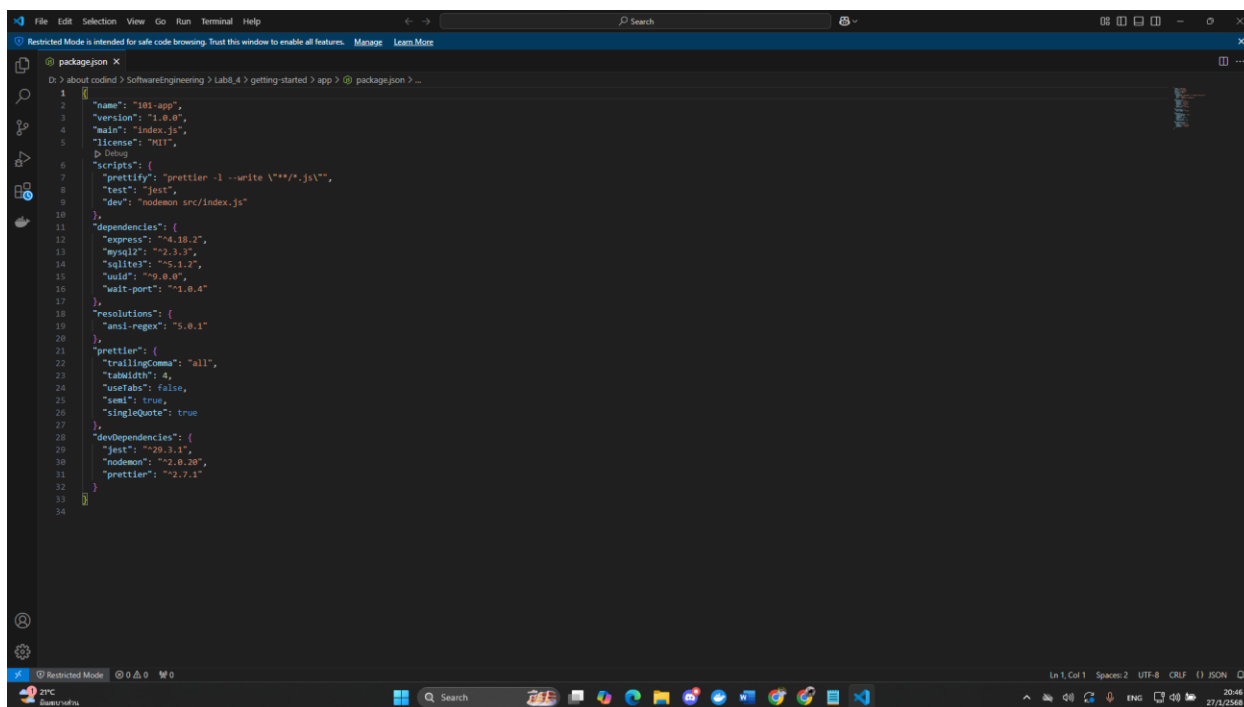


## Lab Worksheet

## แบบฝึกปฏิบัติที่ 8.4: การ Build แอปพลิเคชันจาก Container image และการ Update แอปพลิเคชัน

1. เปิด Command line หรือ Terminal จากนั้นสร้าง Directory ชื่อ Lab8\_4
2. ทำการ Clone ซอร์สโค้ดของเว็บแอปพลิเคชันจาก GitHub repository  
<https://github.com/docker/getting-started.git> ลงใน Directory ที่สร้างขึ้น โดยใช้คำสั่ง  
\$ git clone https://github.com/docker/getting-started.git
3. เปิดดูองค์ประกอบภายใน getting-started/app เมื่อพบไฟล์ package.json ให้ใช้ Text editor ในการเปิดอ่าน

[Check point#7] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงที่อยู่ของ Source code ที่ Clone มาและเนื้อหาของไฟล์ package.json



```
1 {
2   "name": "lab8-app",
3   "version": "1.0.0",
4   "main": "index.js",
5   "license": "MIT",
6   "scripts": {
7     "prettify": "prettier -l --write '**/*.js'",
8     "test": "jest",
9     "dev": "nodemon src/index.js"
10  },
11  "dependencies": {
12    "express": "^4.18.2",
13    "mysql2": "^2.3.3",
14    "sqlite3": "^5.1.2",
15    "uuid": "^9.0.0",
16    "wait-port": "^1.0.4"
17  },
18  "resolutions": {
19    "ansi-regex": "5.0.1"
20  },
21  "prettier": {
22    "trailingComma": "all",
23    "tabWidth": 4,
24    "useTabs": false,
25    "semi": true,
26    "singleQuote": true
27  },
28  "devDependencies": {
29    "jest": "^29.3.1",
30    "nodemon": "^2.0.20",
31    "prettier": "^2.7.1"
32  }
33 }
```

## Lab Worksheet

4. ภายใต้ getting-started/app ให้สร้าง Dockerfile พร้อมกับใส่เนื้อหาดังต่อไปนี้ลงในไฟล์

```
FROM node:18-alpine
```

```
WORKDIR /app
```

```
COPY . .
```

```
RUN yarn install --production
```

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

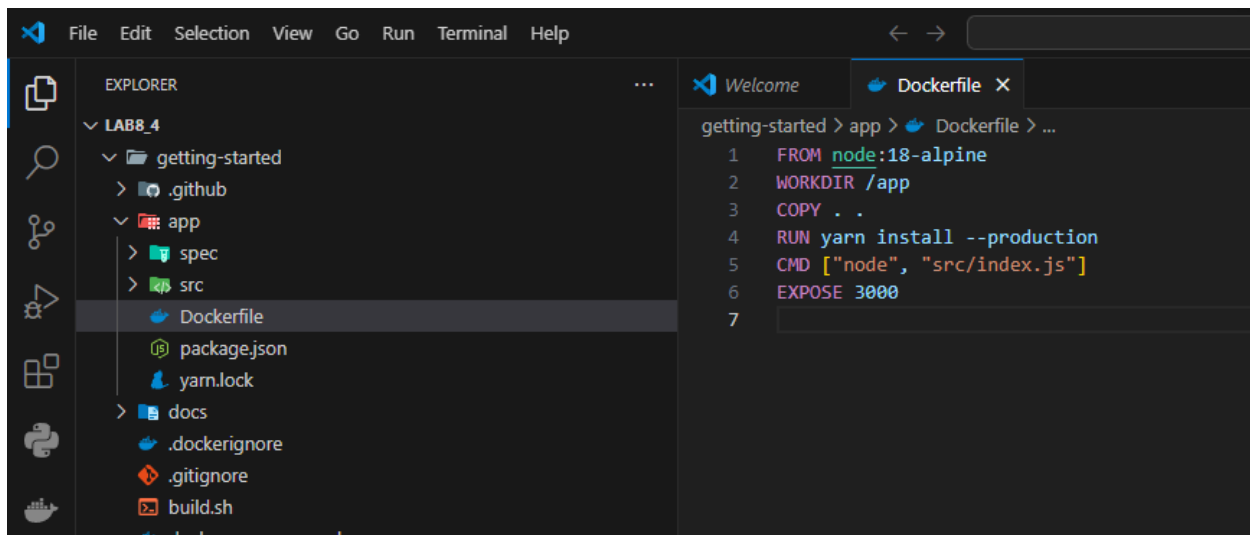
```
EXPOSE 3000
```

5. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้ โดยกำหนดชื่อ image เป็น myapp\_รหัสนศ. ไม่มีขีด

```
$ docker build -t <myapp_รหัสนศ. ไม่มีขีด> .
```

[Check point#8] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง)

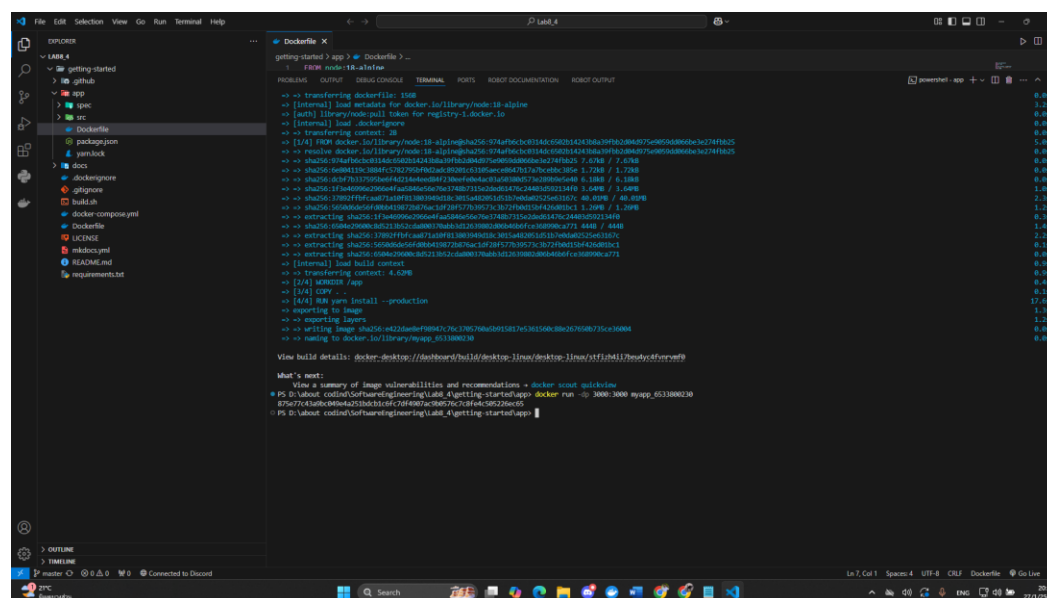
แสดงคำสั่งและผลลัพธ์ที่ได้ทางหน้าจอ



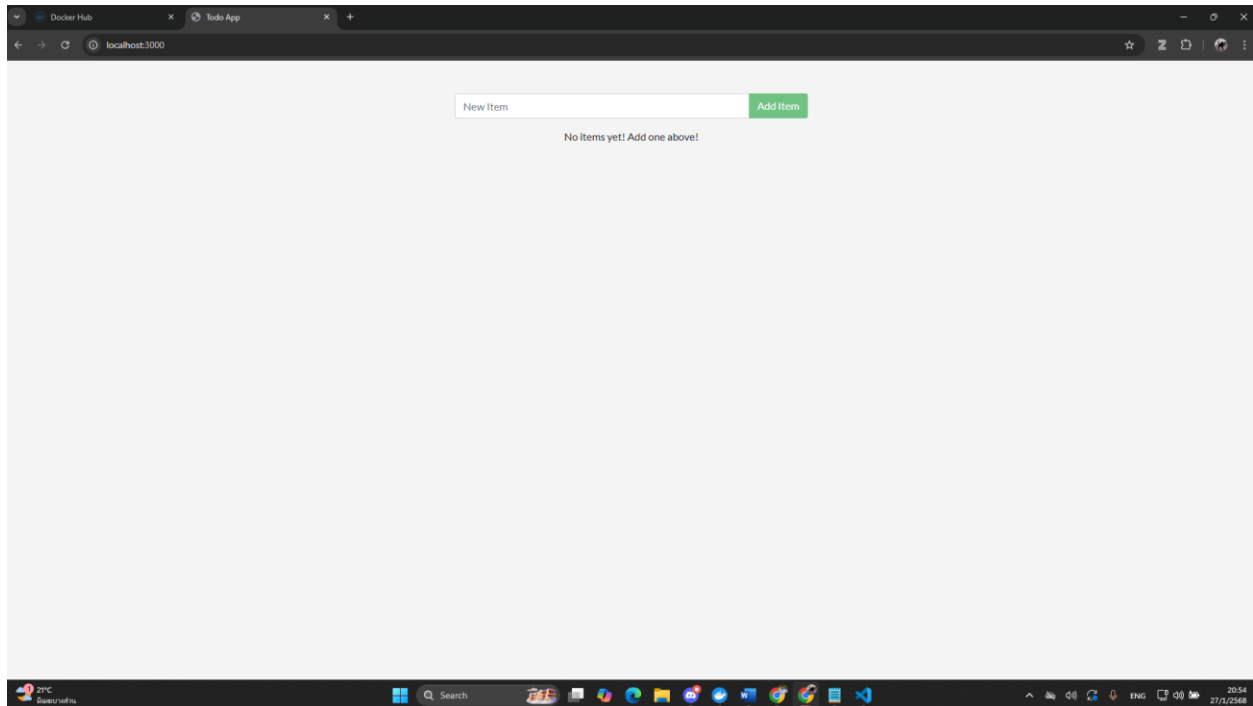
The screenshot shows the Visual Studio Code interface. On the left, the Explorer view displays the file structure of a project named 'LAB8\_4'. The 'getting-started' folder is expanded, showing subfolders '.github' and 'app'. The 'app' folder is further expanded, showing files 'spec', 'src', 'Dockerfile', 'package.json', 'yarn.lock', and a 'docs' folder. The 'Dockerfile' file is selected and its content is displayed in the main editor area. The Dockerfile content is as follows:

```
1 FROM node:18-alpine
2 WORKDIR /app
3 COPY . .
4 RUN yarn install --production
5 CMD ["node", "src/index.js"]
6 EXPOSE 3000
7
```

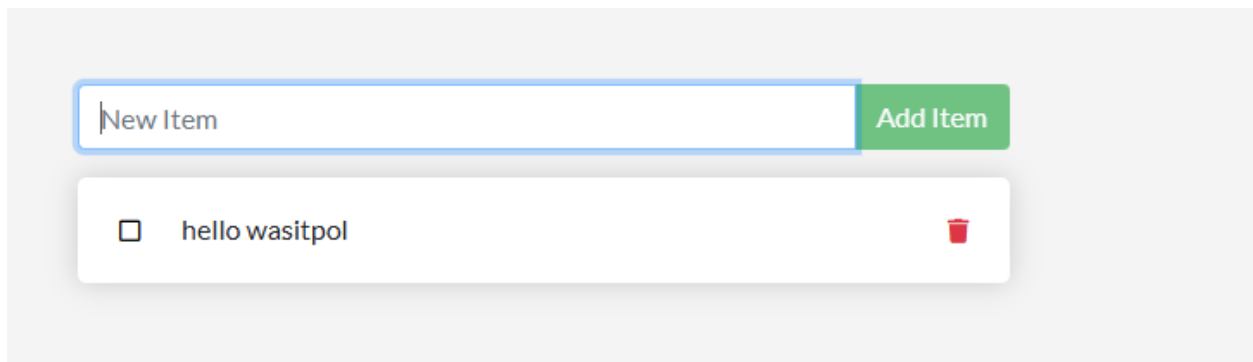
- [Check point#9] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงผลลัพธ์ที่ได้บน Browser และ Dashboard ของ Docker desktop



## Lab Worksheet



หมายเหตุ: นศ.สามารถทดลองเล่น Web application ที่ทำงานอยู่ได้



## Lab Worksheet

8. ทำการแก้ไข Source code ของ Web application ดังนี้

- a. เปิดไฟล์ src/static/js/app.js ด้วย Editor และแก้ไขบรรทัดที่ 56 จาก

No items yet! Add one above!

<p className="text-center">There is no TODO item. Please add one to the list. By  
นามสกุลของนักศึกษา</p>

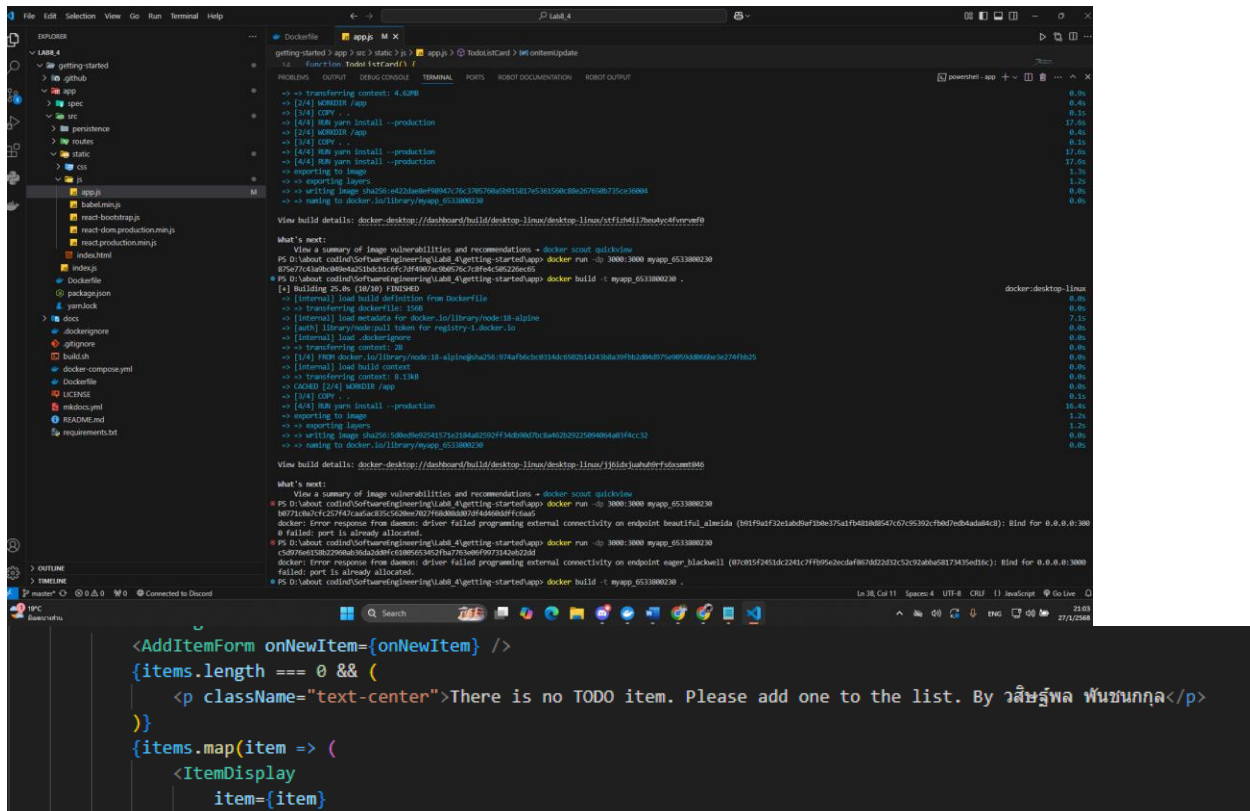
- b. Save ไฟล์ให้เรียบร้อย

9. ทำการ Build Docker image โดยใช้คำสั่งเดียวกันกับข้อ 5

10. Start และรัน Container ตัวใหม่ โดยใช้คำสั่งเดียวกันกับข้อ 6

[Check point#10] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง)

แสดงคำสั่งและผลลัพธ์ที่ได้ทางหน้าจอ พร้อมกับตอบคำถามต่อไปนี้



## Lab Worksheet

(1) Error ที่เกิดขึ้นหมายความว่าอย่างไร และเกิดขึ้นเพราะอะไร

ตอบ มีการใช้ port 3000 แล้วทำให้ทำงานไม่ได้ เพราะไม่ได้ทำการปิด container เก่าที่ใช้ port 3000 อยู่

11. ลบ Container ของ Web application เวอร์ชันก่อนแก้ไขออกจากระบบ โดยใช้วิธีใดวิธีหนึ่งดังต่อไปนี้

a. ผ่าน Command line interface

- i. ใช้คำสั่ง `$ docker ps` เพื่อดู Container ID ที่ต้องการจะลบ
- ii. Copy หรือบันทึก Container ID ไว้
- iii. ใช้คำสั่ง `$ docker stop <Container ID ที่ต้องการจะลบ>` เพื่อหยุดการทำงานของ Container ดังกล่าว
- iv. ใช้คำสั่ง `$ docker rm <Container ID ที่ต้องการจะลบ>` เพื่อทำการลบ

b. ผ่าน Docker desktop

- i. ไปที่หน้าต่าง Containers
- ii. เลือกไอคอนถังขยะในแถวของ Container ที่ต้องการจะลบ
- iii. ยืนยันโดยการกด Delete forever

12. Start และรัน Container ตัวใหม่อีกครั้ง โดยใช้คำสั่งเดียวกันกับข้อ 6

13. เปิด Browser ไปที่ URL = <http://localhost:3000>

[Check point#11] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงผลลัพธ์ที่ได้บน Browser และ Dashboard ของ Docker desktop

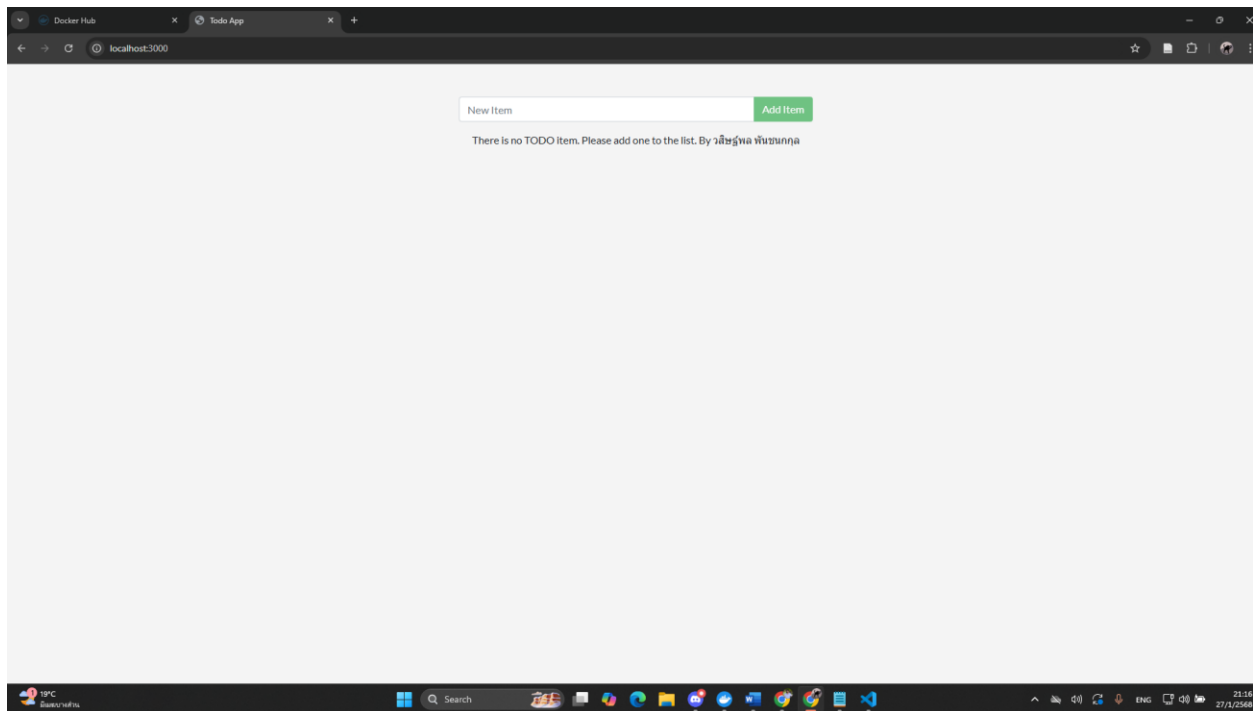
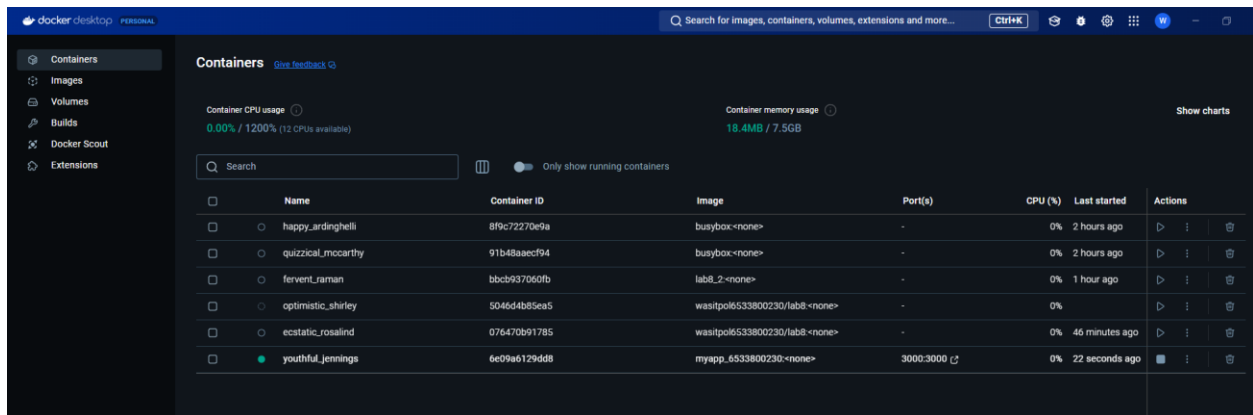
```

Terminal
PS D:\about codind\SoftwareEngineering\Lab8_4> docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                    NAMES
f66a2340ee40   myapp_6533800230  "docker-entrypoint.s..." 12 minutes ago Up 12 minutes  0.0.0.0:3000->3000/tcp   pensive_euler
PS D:\about codind\SoftwareEngineering\Lab8_4> ^C
PS D:\about codind\SoftwareEngineering\Lab8_4> docker stop f66a2340ee40
f66a2340ee40
PS D:\about codind\SoftwareEngineering\Lab8_4> docker rm f66a2340ee40
f66a2340ee40
PS D:\about codind\SoftwareEngineering\Lab8_4>

```



## Lab Worksheet



## Lab Worksheet

## แบบฝึกปฏิบัติที่ 8.5: เริ่มต้นสร้าง Pipeline อย่างง่ายสำหรับการ Deploy ด้วย Jenkins

1. เปิด Command line หรือ Terminal บน Docker Desktop

2. ป้อนคำสั่งและทำการรัน container โดยผูกพอร์ต

```
$ docker run -p 8080:8080 -p 50000:50000 --restart=on-failure jenkins/jenkins:lts-jdk17
```

หรือ

```
$ docker run -p 8080:8080 -p 50000:50000 --restart=on-failure -v
```

```
jenkins_home:/var/jenkins_home jenkins/jenkins:lts-jdk17
```

3. บันทึกรหัสผ่านของ Admin user ไว้สำหรับ log-in ในครั้งแรก

[Check point#12] Capture หน้าจอที่แสดงผล Admin password

```

Terminal
*****
*****
*****
Jenkins initial setup is required. An admin user has been created and a password generated.
Please use the following password to proceed to installation:

1ffa584073474e658466c15506bc6fc

This may also be found at: /var/jenkins_home/secrets/initialAdminPassword

*****
*****
*****
2025-01-27 14:26:33.819+0000 [id=51] INFO jenkins.InitReactorRunner$1#onAttained: Completed initialization
2025-01-27 14:26:33.836+0000 [id=25] INFO hudson.lifecycle.Lifecycle#onReady: Jenkins is fully up and running
2025-01-27 14:26:35.754+0000 [id=68] INFO h.m.DownloadService$Downloadable#load: Obtained the updated data file for hudson.tasks.Maven.MavenInstaller
2025-01-27 14:26:35.754+0000 [id=68] INFO hudson.util.Retrier#start: Performed the action check updates server successfully at the attempt #1
2025-01-27 14:27:18.501+0000 [id=19] INFO hudson.PluginManager#install: Starting installation of a batch of 20 plugins plus their dependencies
2025-01-27 14:27:18.505+0000 [id=19] INFO hudson.model.UpdateSite$Plugin#deploy: Adding dependent install of icons-api for plugin cloudbees-folder
2025-01-27 14:27:18.507+0000 [id=109] INFO h.model.UpdateCenter$DownloadJob#run: Starting the installation of icons-api on behalf of admin
2025-01-27 14:27:18.508+0000 [id=19] INFO hudson.model.UpdateSite$Plugin#deploy: Adding dependent install of json-path-api for plugin build-timestamp
2025-01-27 14:27:18.509+0000 [id=19] INFO hudson.model.UpdateSite$Plugin#deploy: Adding dependent install of asm-api for plugin json-path-api
2025-01-27 14:27:18.509+0000 [id=19] INFO hudson.model.UpdateSite$Plugin#deploy: Adding dependent install of token-macro for plugin build-timestamp
2025-01-27 14:27:18.511+0000 [id=19] INFO hudson.model.UpdateSite$Plugin#deploy: Adding dependent install of structs for plugin token-macro
2025-01-27 14:27:18.512+0000 [id=19] INFO hudson.model.UpdateSite$Plugin#deploy: Adding dependent install of workflow-step-api for plugin token-macro
2025-01-27 14:27:18.514+0000 [id=19] INFO hudson.model.UpdateSite$Plugin#deploy: Adding dependent install of plain-credentials for plugin credentials-binding

```

4. เมื่อได้รับการยืนยันว่า Jenkins is fully up and running ให้เปิดบราวเซอร์ และป้อนที่อยู่เป็น

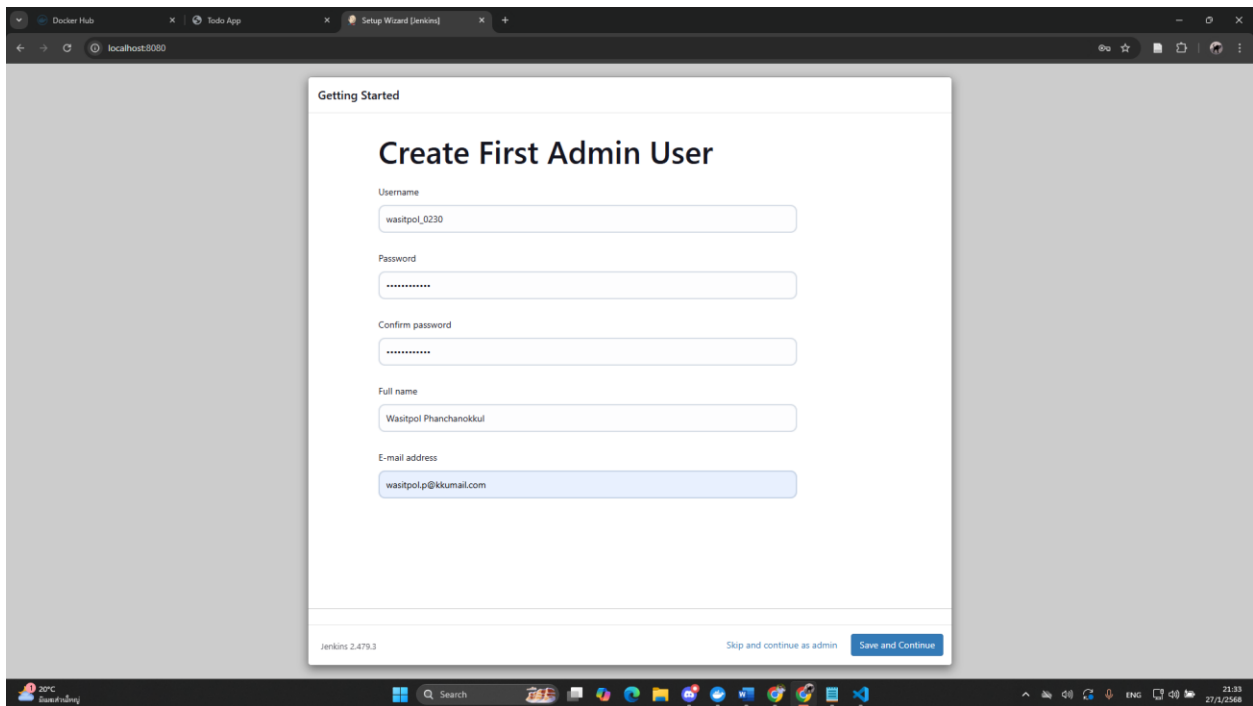
localhost:8080

5. ทำการ Unlock Jenkins ด้วยรหัสผ่านที่ได้ในข้อที่ 3

6. สร้าง Admin User โดยใช้ username เป็นชื่อจริงของนักศึกษาพร้อมรหัสสี่ตัวท้าย เช่น somsri\_3062

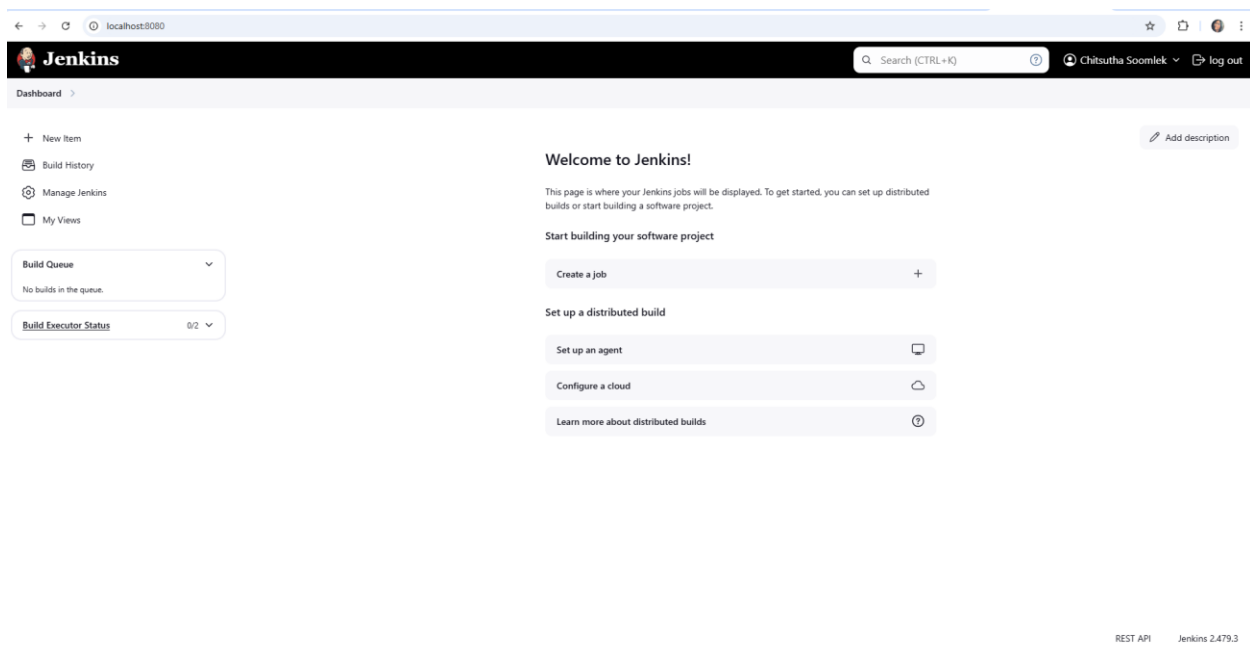
## Lab Worksheet

[Check point#13] Capture หน้าจอที่แสดงผลการตั้งค่า



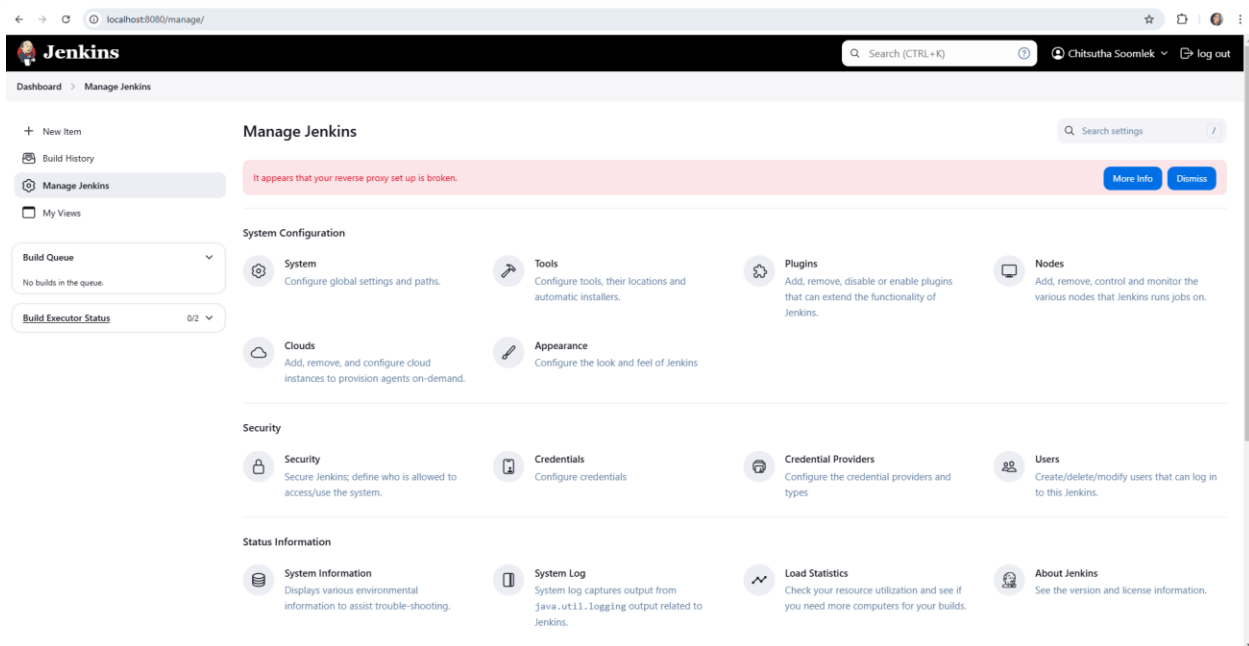
7. กำหนด Jenkins URL เป็น <http://localhost:8080/lab8>

8. เมื่อติดตั้งเรียบร้อยแล้วจะพบหน้าจอ Dashboard ดังแสดงในภาพ



## Lab Worksheet

## 9. เลือก Manage Jenkins แล้วไปที่เมนู Plugins

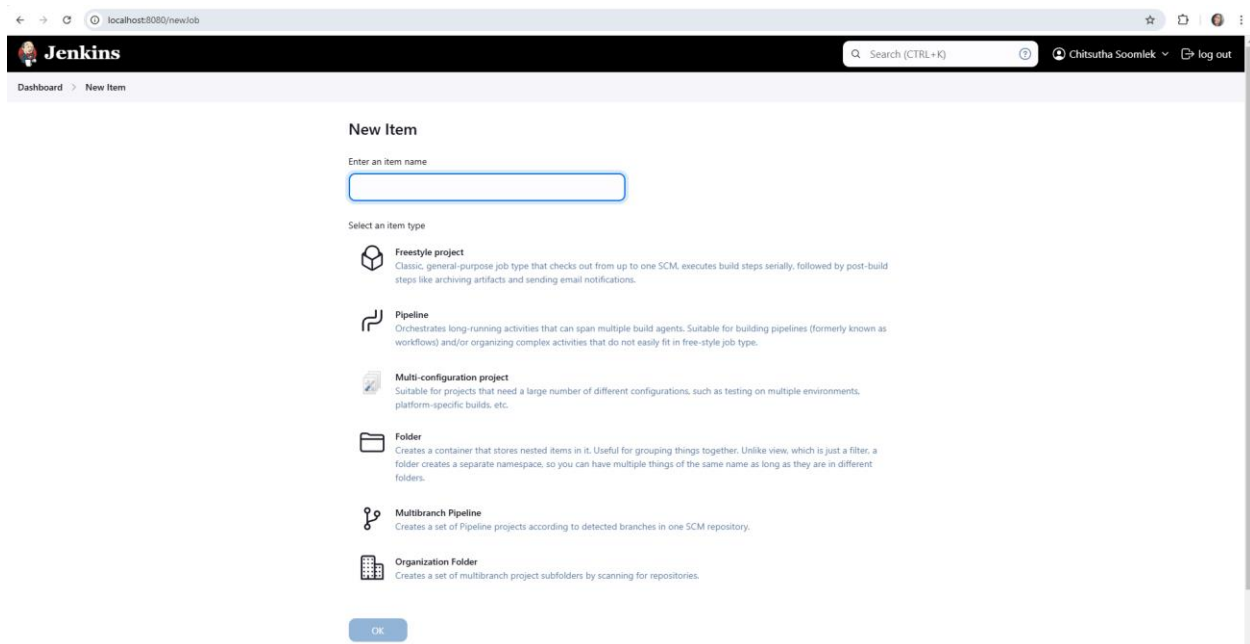


## 10. ไปที่เมนู Available plugins แล้วเลือกติดตั้ง Robotframework เพิ่มเติม



## 11. กลับไปที่หน้า Dashboard แล้วสร้าง Pipeline อย่างง่าย โดยกำหนด New item เป็น Freestyle project และตั้งชื่อเป็น UAT

## Lab Worksheet



12. นำไฟล์ .robot ที่ทำให้แบบฝึกปฏิบัติที่ 7 (Lab#7) ไปไว้บน Repository ของนักศึกษา จากนั้นตั้งค่าที่จำเป็นในหน้านี้ทั้งหมด ดังนี้

Description: Lab 8.5

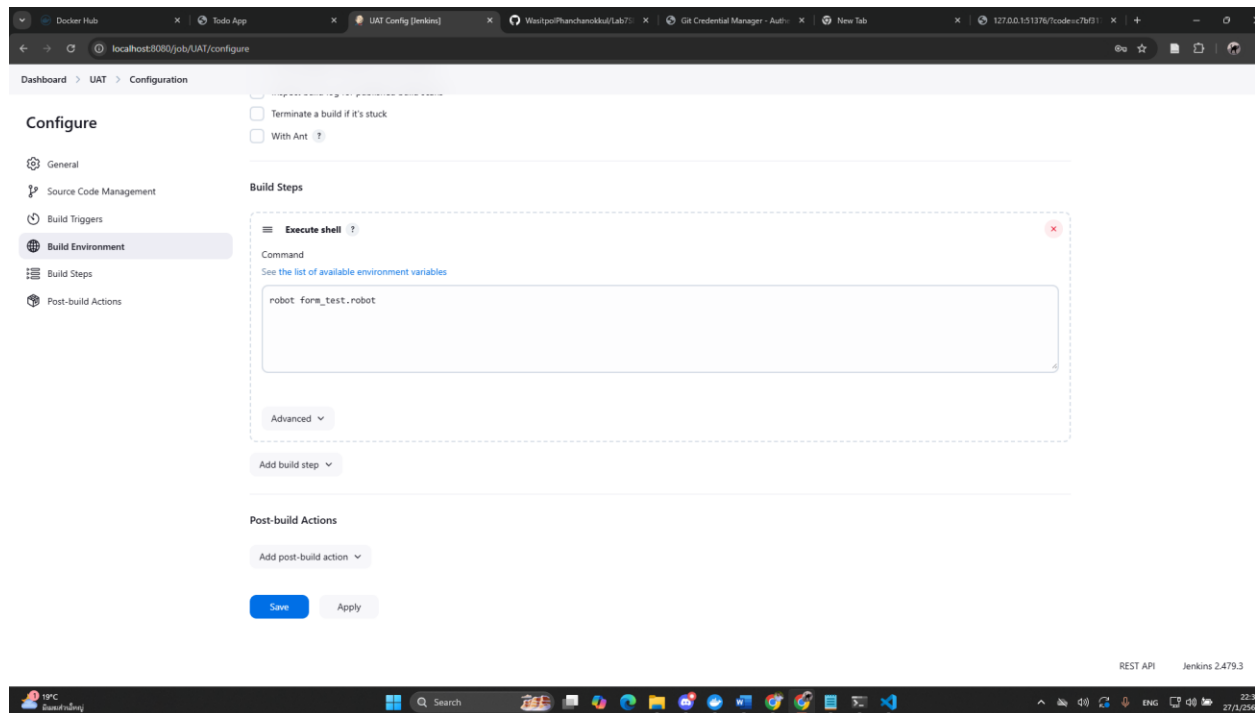
GitHub project: กดเลือก แล้วใส่ Project URL เป็น repository ที่เก็บโค้ด .robot (ดูขั้นตอนที่ 12)

Build Trigger: เลือกแบบ Build periodically แล้วกำหนดให้ build ทุก 15 นาที

Build Steps: เลือก Execute shell แล้วใส่คำสั่งในการรันไฟล์ .robot (หากไฟล์ไม่ได้อยู่ในหน้าแรกของ repository ให้ใส่ Path ไปถึงไฟล์ให้เรียบร้อยแล้ว)

## Lab Worksheet

[Check point#14] Capture หน้าจอแสดงการตั้งค่า พร้อมกับตอบคำถามต่อไปนี้



(1) คำสั่งที่ใช้ในการ Execute ไฟล์ .robot ใน Build Steps คือ  
ตอบ form\_test.robot

## Lab Worksheet

Post-build action: เพิ่ม Publish Robot Framework test results ->

ระบุไคเร็คทอรีที่เก็บไฟล์ผลการทดสอบโดย Robot framework ในรูป xml และ html -> ตั้งค่า Threshold เป็น % ของการทดสอบที่ไม่ผ่านแล้วนับว่าซอฟต์แวร์มีปัญหา -> ตั้งค่า Threshold เป็น % ของการทดสอบที่ผ่านแล้วนับว่าซอฟต์แวร์มีอยู่ในสถานะที่สามารถนำไปใช้งานได้ (เช่น 20, 80)

13. กด Apply และ Save

14. สั่ง Build Now

[Check point#15] Capture หน้าจอแสดงหน้าหลักของ Pipeline และ Console Output

Console Output

Download
Copy
View as plain text

```

Started by user Masitpol Phanchanukul
Running as SYSTEM
Building in workspace /var/jenkins_home/workspace/UAT
The recommended git tool is: NONE
No credentials specified
Cloning the remote git repository
Cloning repository https://github.com/MasitpolPhanchanukul/Lab75E.git
> git init /var/jenkins_home/workspace/UAT # timeout=10
Fetching upstream changes from https://github.com/MasitpolPhanchanukul/Lab75E.git
> git --version # timeout=10
> git --version # "git version 2.38.5"
> git fetch --tags --progress -- https://github.com/MasitpolPhanchanukul/Lab75E.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git config remote.origin.url https://github.com/MasitpolPhanchanukul/Lab75E.git # timeout=10
> git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision 68083e226eb7c495ceb32a49942e08429287 (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f 68083e226eb7c495ceb32a49942e08429287 # timeout=10
Commit message: "first commit"
First time build. Skipping changelog.
[MAT] $ mkdir -p /var/jenkins354258648756258329.sh
+ robot form_test.robot
[ ERROR ] Error in file "/var/jenkins_home/workspace/UAT/resource.robot" on line 7: Importing library "SeleniumLibrary" failed: ModuleNotFoundError: No module named "SeleniumLibrary"
Traceback (most recent call last):
  None
PYTHONPATH:
  /usr/local/bin
  /usr/lib/python3.11.zip
  /usr/lib/python3.11
  /usr/lib/python3.11/lib-dynload
  /usr/local/lib/python3.11/dist-packages
  /usr/lib/python3.11/dist-packages
=====
Form Test
=====
Open Form                                     | FAIL |
Evaluating expression "sys.modules['selenium.webdriver'].ChromeOptions()" failed: KeyError: 'selenium.webdriver'
-----
Record Success                               | FAIL |
No keyword with name "Go To" found.

Also teardown failed:
No keyword with name "Close Browser" found.
-----
[ WARN ] Multiple tests with name "Open Form" executed in suite "Form Test".
Open Form                                     | FAIL |
Evaluating expression "sys.modules['selenium.webdriver'].ChromeOptions()" failed: KeyError: 'selenium.webdriver'
-----
Empty Destination                             | FAIL |
No keyword with name "Go To" found.

Also teardown failed:
No keyword with name "Close Browser" found.
-----
[ WARN ] Multiple tests with name "Open Form" executed in suite "Form Test".
Open Form                                     | FAIL |
Evaluating expression "sys.modules['selenium.webdriver'].ChromeOptions()" failed: KeyError: 'selenium.webdriver'
-----
Empty Email                                   | FAIL |
No keyword with name "Go To" found.

Also teardown failed:
No keyword with name "Close Browser" found.
-----
[ WARN ] Multiple tests with name "Open Form" executed in suite "Form Test".
Open Form                                     | FAIL |
Evaluating expression "sys.modules['selenium.webdriver'].ChromeOptions()" failed: KeyError: 'selenium.webdriver'
-----
Empty Phone Number                             | FAIL |
No keyword with name "Go To" found.

Also teardown failed:
No keyword with name "Close Browser" found.
-----
[ WARN ] Multiple tests with name "Open Form" executed in suite "Form Test".
Open Form                                     | FAIL |
Evaluating expression "sys.modules['selenium.webdriver'].ChromeOptions()" failed: KeyError: 'selenium.webdriver'
-----
Invalid Phone Number                             | FAIL |
No keyword with name "Go To" found.

```

## Lab Worksheet

```
Also teardown failed:
No keyword with name "Close Browser" found.
.....
For: Test | FAIL |
12 tests, 0 passed, 12 failed
=====
Output: /var/jenkins_home/workspace/UMT/output.xml
Log: /var/jenkins_home/workspace/UMT/log.html
Report: /var/jenkins_home/workspace/UMT/report.html
Build step 'Execute shell' marked build as failure
Robot results publisher started...
INFO: Checking test criticality is deprecated and will be dropped in a future release!
-Parsing output xml:
Done!
-Copying log files to build dir:
Done!
-Assigning results to build:
Done!
-Checking thresholds:
Done!
Done publishing Robot results.
Finished: FAILURE
```