

**Lab Worksheet**

ชื่อ-นามสกุล \_\_\_\_\_ ปัจญາวุธ แสงแแดง \_\_\_\_\_ รหัสนักศึกษา \_\_\_\_\_ 653380138-3 \_\_\_\_\_ Section \_\_\_\_\_ 1 \_\_\_\_\_

**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**

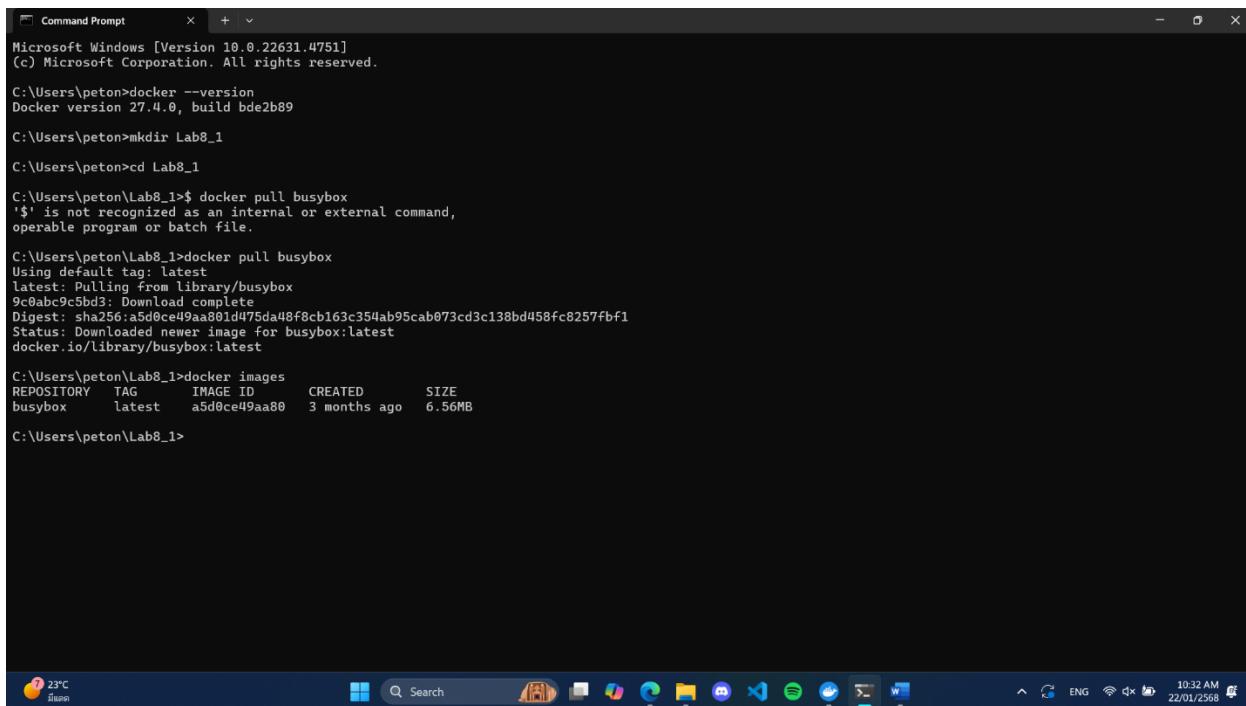
1. ติดตั้ง Docker desktop ลงบนเครื่องคอมพิวเตอร์ โดยดาวน์โหลดจาก <https://www.docker.com/get-started>
2. สร้าง Account บน Docker hub (<https://hub.docker.com/signup>)
3. กำหนดให้ \$ หมายถึง Command prompt และ <> หมายถึง ให้ป้อนค่าของพารามิเตอร์ที่กำหนด

**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

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

## Lab Worksheet



```

Command Prompt
Microsoft Windows [Version 10.0.22631.4751]
(c) Microsoft Corporation. All rights reserved.

C:\Users\peton>docker --version
Docker version 27.4.0, build bde2b89

C:\Users\peton>mkdir Lab8_1

C:\Users\peton>cd Lab8_1

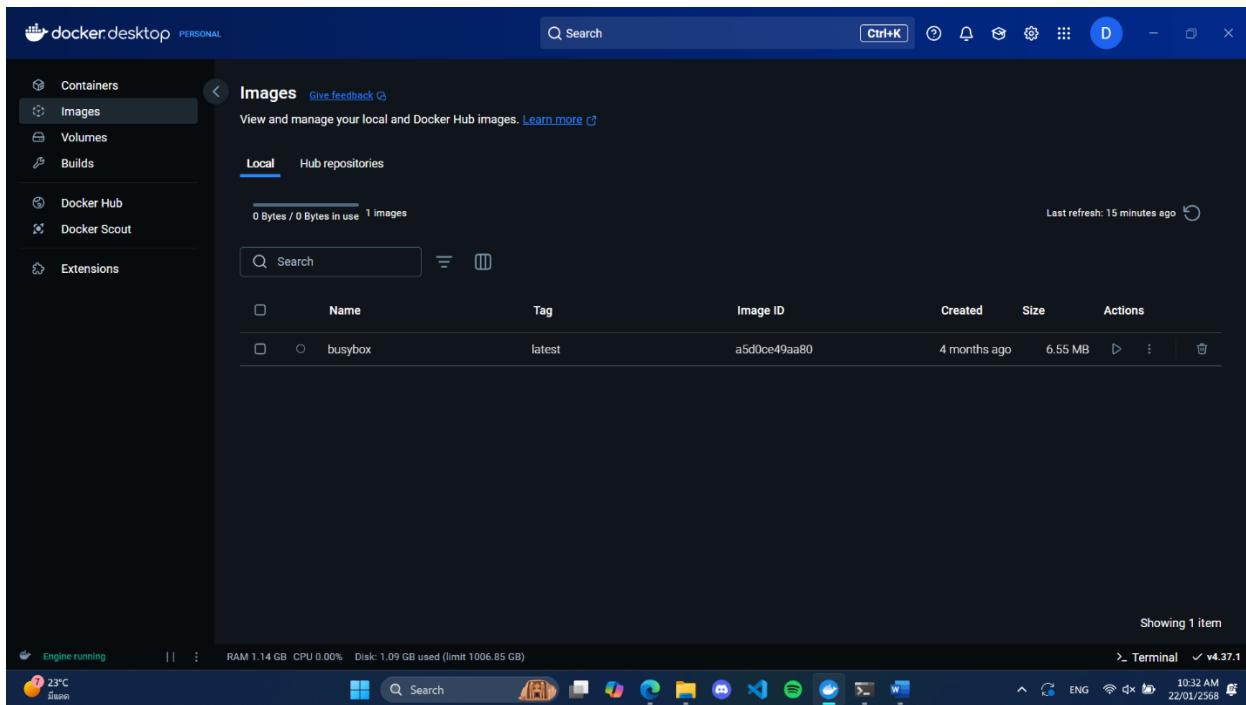
C:\Users\peton\Lab8_1>$ docker pull busybox
'$' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\peton\Lab8_1>docker pull busybox
Using default tag: latest
latest: Pulling from library/busybox
9c0abc9c5bd3: Download complete
Digest: sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138bd458fc8257fbf1
Status: Downloaded newer image for busybox:latest
docker.io/library/busybox:latest

C:\Users\peton\Lab8_1>docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
busybox latest a5d0ce49aa80 3 months ago 6.56MB

C:\Users\peton\Lab8_1>

```

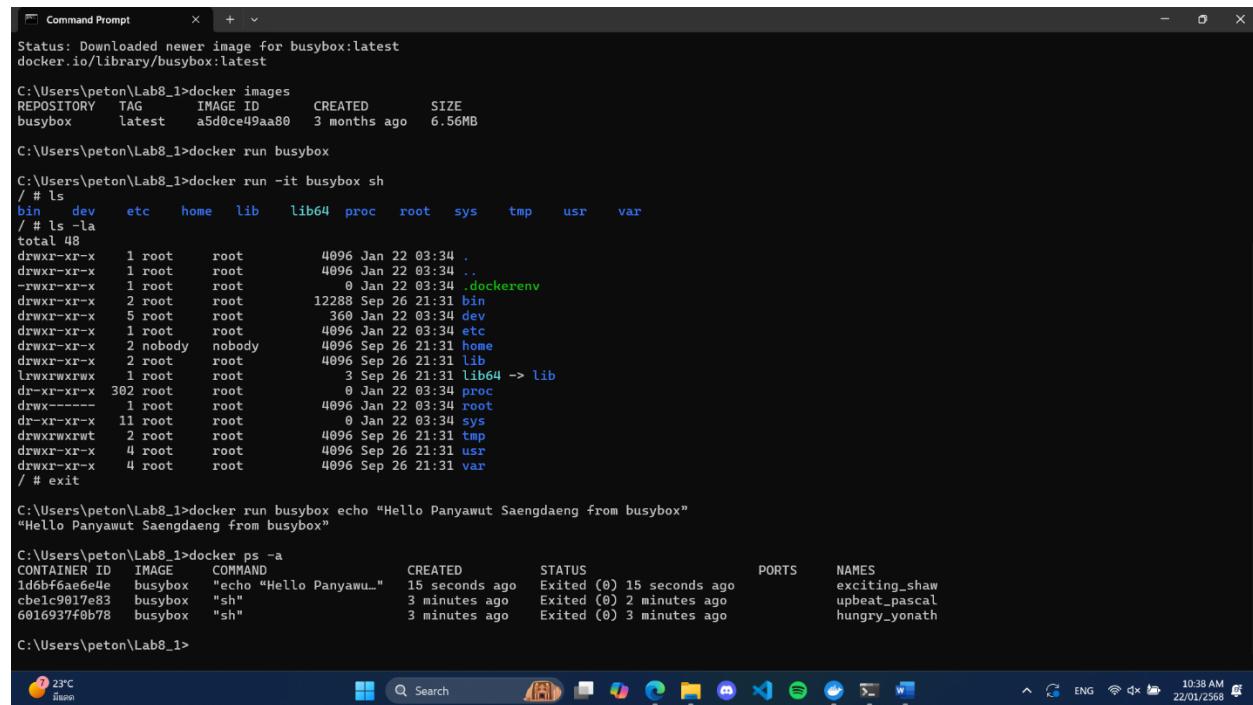


## Lab Worksheet

- (1) สิ่งที่อยู่ภายใต้คอลัมน์ Repository คืออะไร : ชื่อของ Docker image repository เช่น busybox  
 (2) Tag ที่ใช้บ่งบอกถึงอะไร : Tag บ่งบอกถึงเวอร์ชันของ Docker image นั้นๆ

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



```

C:\Users\peton\Lab8_1>docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
busybox         latest   a5d0ce49aa80  3 months ago  6.56MB

C:\Users\peton\Lab8_1>docker run busybox
/bin/sh: 0: exec: /bin/sh: not found
[1] 1000 stopped  bash

C:\Users\peton\Lab8_1>ls
total 48
drwxr-xr-x  1 root    root     4096 Jan 22 03:34 .
drwxr-xr-x  1 root    root     4096 Jan 22 03:34 ..
-rw-r--r--  1 root    root     0 Jan 22 03:34 dockerenv
drwxr-xr-x  2 root    root    12288 Sep 26 21:31 bin
drwxr-xr-x  5 root    root    368 Jan 22 03:34 dev
drwxr-xr-x  1 root    root    4096 Jan 22 03:34 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  302 root    root     0 Jan 22 03:34 proc
drwx-----  1 root    root    4096 Jan 22 03:34 root
dr-xr-xr-x  11 root   root     0 Jan 22 03:34 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

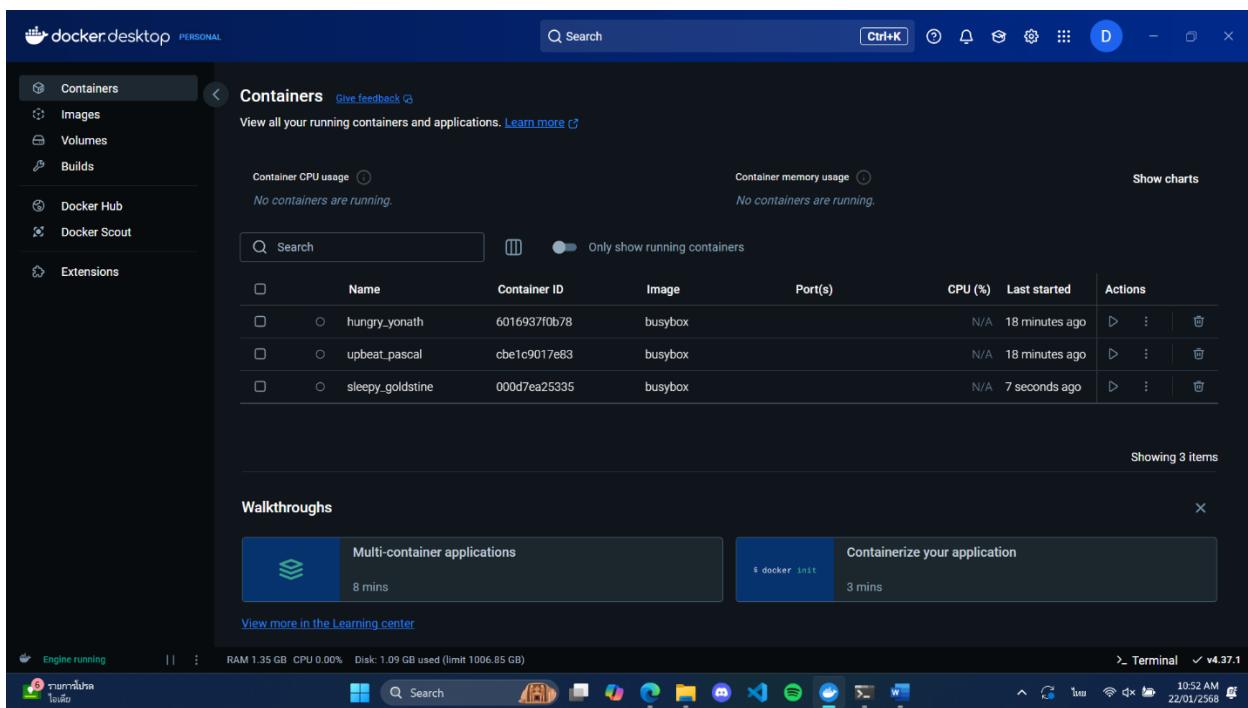
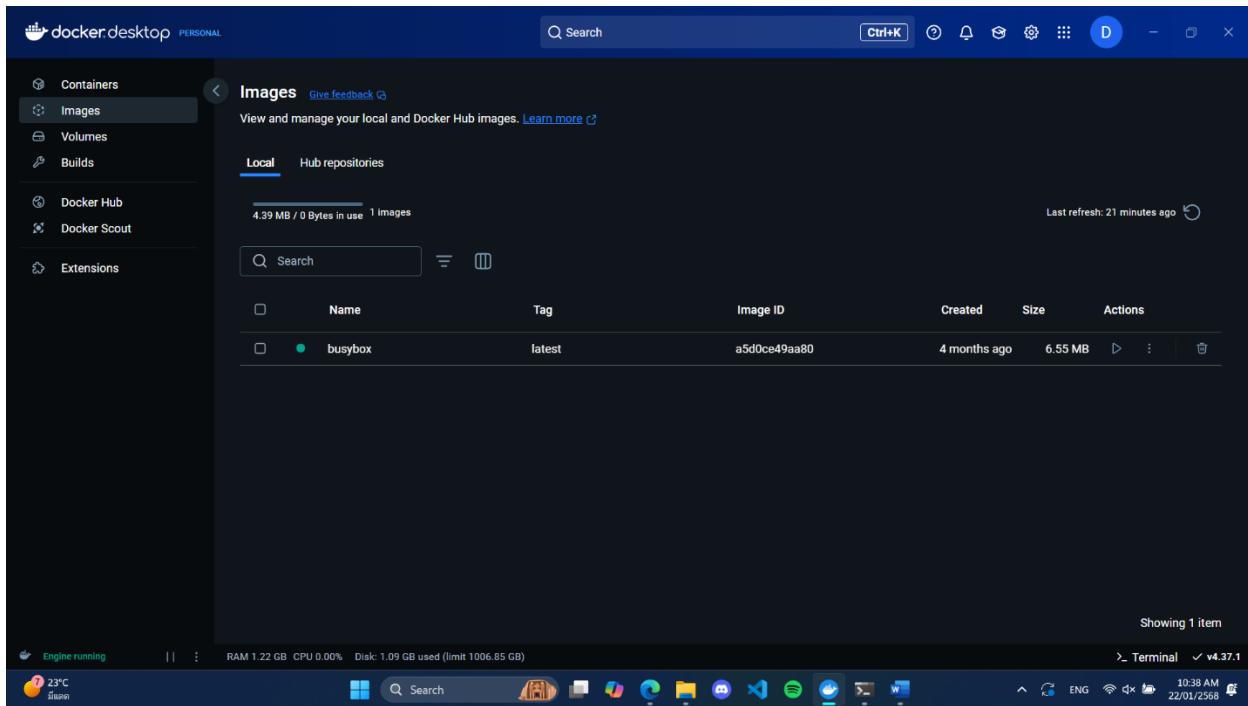
C:\Users\peton\Lab8_1>docker run busybox echo "Hello Panyawut Saengdaeng from busybox"
"Hello Panyawut Saengdaeng from busybox"

C:\Users\peton\Lab8_1>docker ps -a
CONTAINER ID        IMAGE           COMMAND       CREATED          STATUS          PORTS          NAMES
1d6bf6a6e6e4        busybox         "echo \"Hello Panyawu..." 15 seconds ago   Exited (0) 15 seconds ago   exciting_shaw
cbe1c9817e83        busybox         "sh"          3 minutes ago   Exited (0) 2 minutes ago   upbeat_pascal
6016937f0b78        busybox         "sh"          3 minutes ago   Exited (0) 3 minutes ago   hungry_yonath

C:\Users\peton\Lab8_1>

```

## Lab Worksheet



(1) เมื่อใช้ option -it ในคำสั่ง run ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสั้นๆ

## Lab Worksheet

- **-i (interactive):** เปิดใช้งานโหมด interactive ที่ทำให้ Container รับการป้อนข้อมูลจากผู้ใช้ผ่าน stdin
- **-t (tty):** จัดสรร pseudo-TTY (terminal) ให้กับ Container ทำให้ผู้ใช้สามารถโต้ตอบกับ shell ภายใน Container ได้เหมือนกับการทำงานใน terminal ปกติ
- การใช้ **-it** ทำให้ผู้ใช้สามารถเข้าใช้งาน shell ของ Container แบบ interactive ได้โดยตรง ซึ่งหมายความว่าการตรวจสอบหรือรันคำสั่งใน Container แบบโต้ตอบ

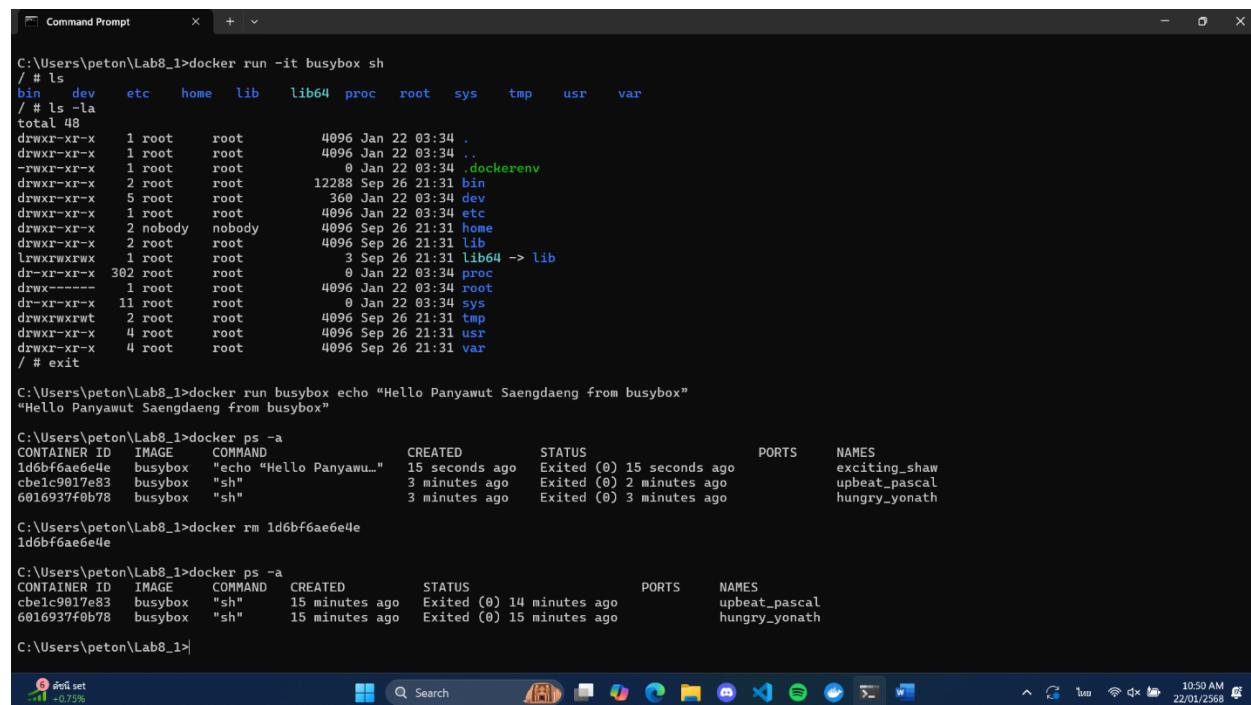
(2) คอลัมน์ STATUS จากการรันคำสั่ง docker ps -a แสดงถึงข้อมูลอะไร

คอลัมน์ STATUS แสดงสถานะปัจจุบันของแต่ละ Container เช่น:

- Up: Container กำลังรันอยู่
- Exited: Container ได้หยุดการทำงานแล้ว พร้อมกับระบุรหัส exit code
- Created: Container ถูกสร้างขึ้นแต่ยังไม่ได้รัน

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

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



```
C:\Users\peton\Lab8_1>docker run -it busybox sh
/ # ls
bin dev etc home lib lib64 proc root sys tmp usr var
/ # ls -la
total 48
drwxr-xr-x 1 root root 4096 Jan 22 03:34 .
drwxr-xr-x 1 root root 4096 Jan 22 03:34 ..
-rwxr-xr-x 1 root root 0 Jan 22 03:34 dockerenv
drwxr-xr-x 2 root root 12288 Sep 26 21:31 bin
drwxr-xr-x 5 root root 368 Jan 22 03:34 dev
drwxr-xr-x 1 root root 4096 Jan 22 03:34 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
drwxr-xr-x 382 root root 0 Jan 22 03:34 proc
drwx----- 1 root root 4096 Jan 22 03:34 root
drwxr-xr-x 11 root root 0 Jan 22 03:34 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

C:\Users\peton\Lab8_1>docker run busybox echo "Hello Panyawut Saengdaeng from busybox"
"Hello Panyawut Saengdaeng from busybox"

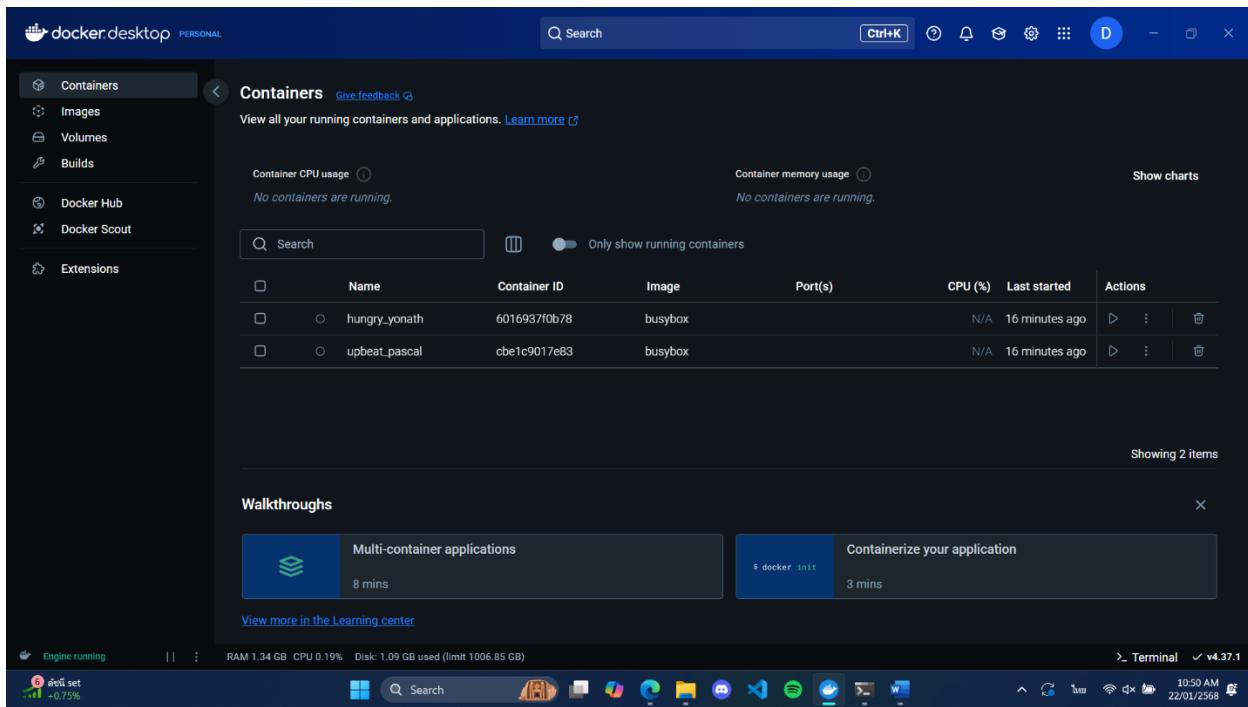
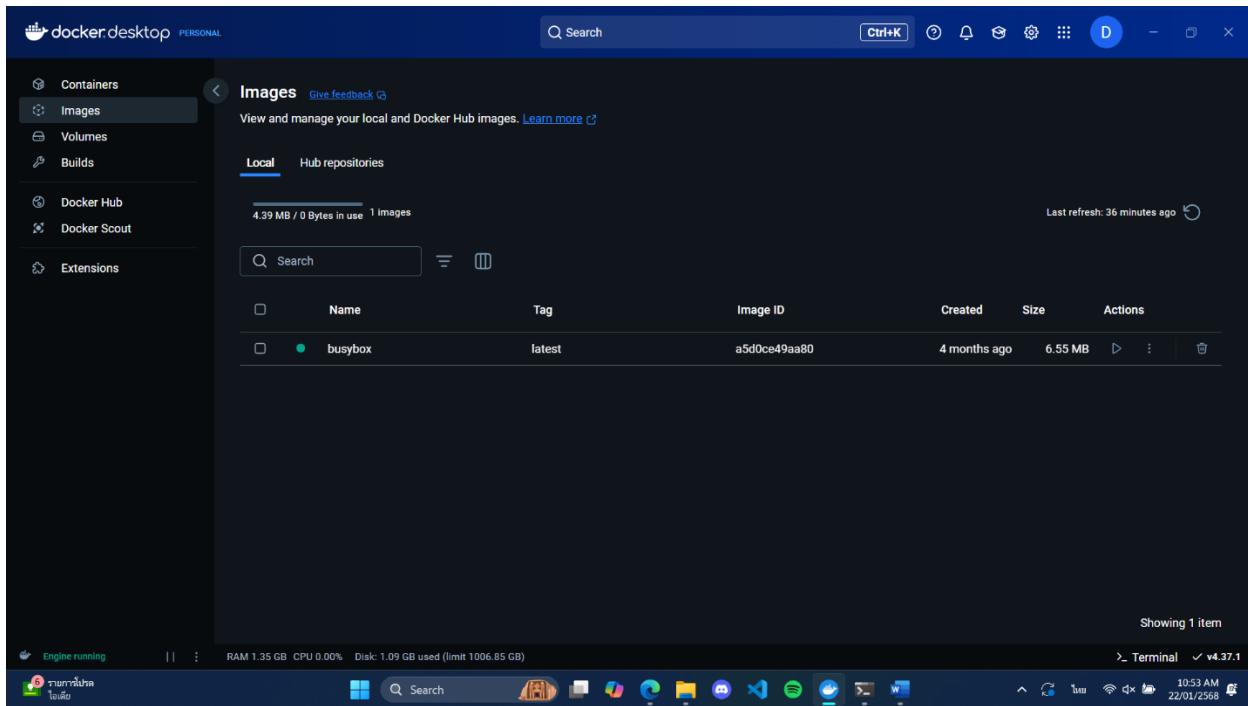
C:\Users\peton\Lab8_1>docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
1d6bf6ae6e4e busybox "echo \"Hello Panyawut Saengdaeng from busybox\""
3 minutes ago Exited (0) 15 seconds ago exciting_shaw
cbe1c9017e83 busybox "sh"
3 minutes ago Exited (0) 2 minutes ago upbeat_pascal
6016937f0b78 busybox "sh"
3 minutes ago Exited (0) 3 minutes ago hungry_yonath

C:\Users\peton\Lab8_1>docker rm 1d6bf6ae6e4e
1d6bf6ae6e4e

C:\Users\peton\Lab8_1>docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
cbe1c9017e83 busybox "sh"
15 minutes ago Exited (0) 14 minutes ago upbeat_pascal
6016937f0b78 busybox "sh"
15 minutes ago Exited (0) 15 minutes ago hungry_yonath

C:\Users\peton\Lab8_1|
```

## Lab Worksheet



**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

สำหรับเครื่องที่ใช้ระบบปฏิบัติการวินโดว์ส (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 และป้อนคำสั่งต่อไปนี้

```
$ 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

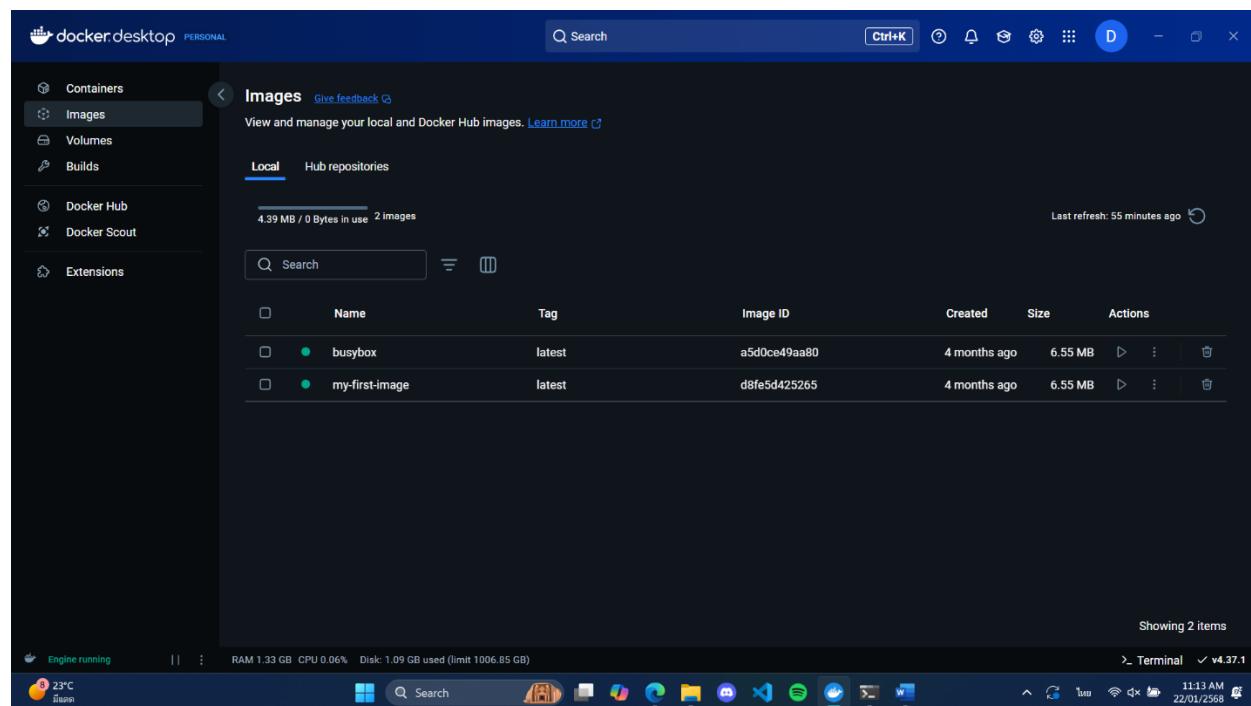
5. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้

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

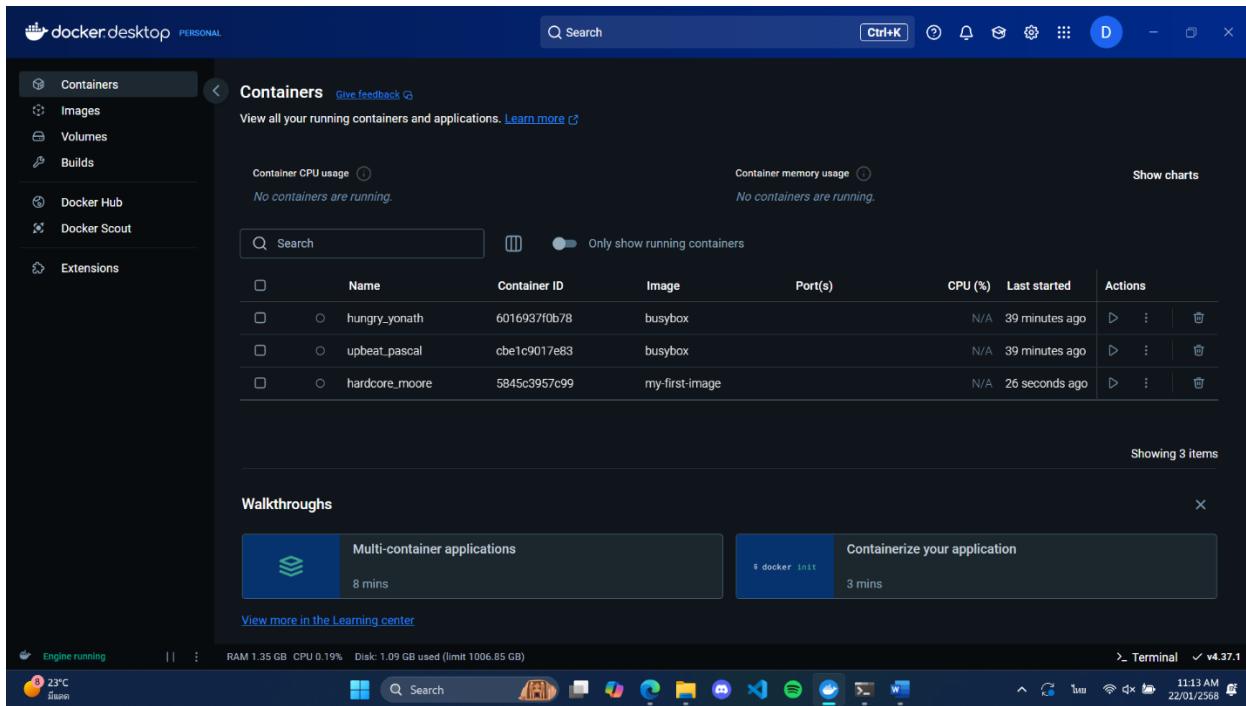
6. เมื่อ Build สำเร็จแล้ว ให้ทำการรัน Docker image ที่สร้างขึ้นในขั้นตอนที่ 5

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

## Lab Worksheet



## Lab Worksheet



(1) คำสั่งที่ใช้ในการ run คือ

`docker run my-first-image`

(2) Option -t ในคำสั่ง `$ docker build` ส่งผลต่อการทำางานของคำสั่งอย่างไรบ้าง อธิบายมาพอสักเช่น

-t ย่อมาจาก --tag ใช้สำหรับตั้งชื่อและแท็กให้กับ Docker image ที่สร้างขึ้น

โดยทั่วไปจะอยู่ในรูปแบบ ชื่อ:แท็ก (เช่น my-first-image:latest) แต่หากไม่ได้ระบุแท็ก จะใช้ค่าเริ่มต้นเป็น latest การตั้งชื่อและแท็กช่วยให้การอ้างอิง, จัดการ, และเผยแพร่ Docker image ง่ายขึ้น เช่น การ push ไปยัง Docker Hub หรือการรัน image ภายหลัง

### 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
- สำหรับเครื่องที่ใช้ระบบปฏิบัติการวินโดว์ บันทึกคำสั่งต่อไปนี้ลงในไฟล์ โดยใช้ Text Editor ที่มี

## 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"
```

สำหรับเครื่องที่ใช้ระบบปฏิบัติการ MacOS หรือ Linux บนหน้าต่าง Terminal และป้อนคำสั่งต่อไปนี้

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

```
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

```

Command Prompt >= unpacking to docker.io/doubleserz/lab8:latest
View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/dzvg3e2kbgun7pnc14urjwgab
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)

C:\Users\peton\Lab8_3>docker run doubleserz/lab8
Panyawut Saengdaeng 653380138-3

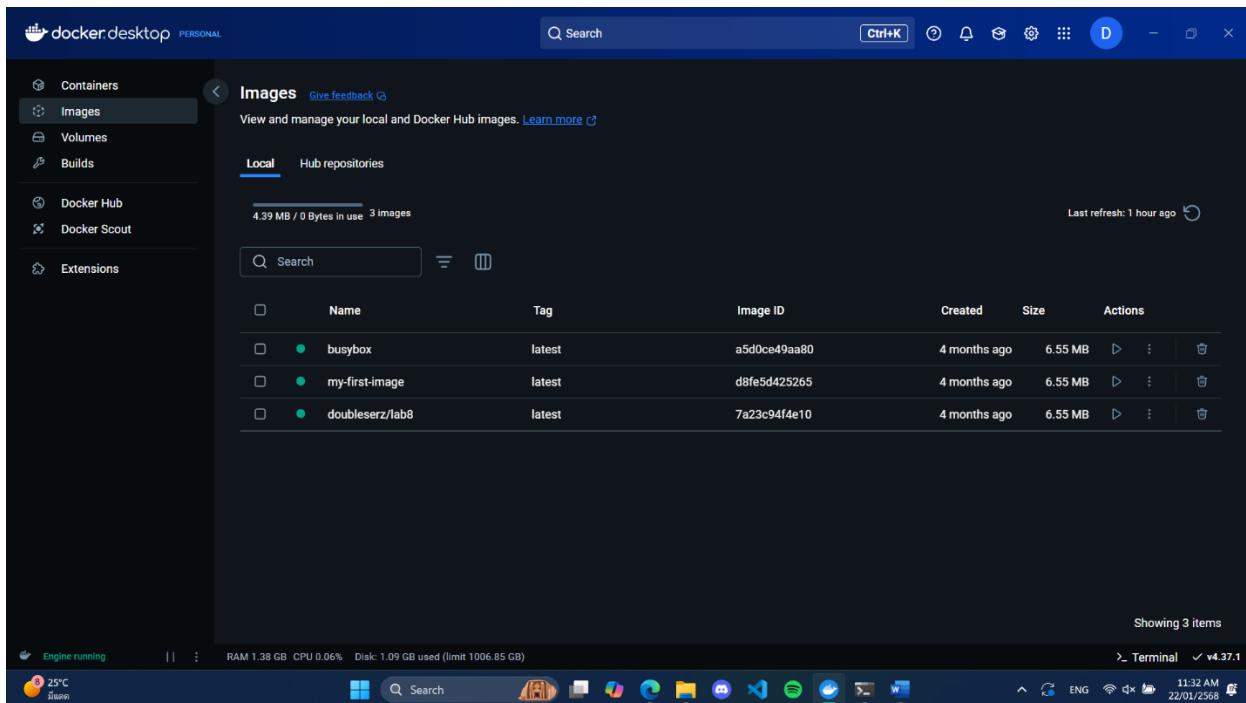
C:\Users\peton\Lab8_3>docker build -t doubleserz/lab8 .
[+] Building 0.3s (5/5) FINISHED
=> [internal] load build definition from Dockerfile
=> transferring dockerfile: 173B
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 2)
=> [internal] load metadata for docker.io/library/busybox:latest
=> [internal] load .dockerrcignore
=> transferring context: 2B
=> CACHED [1/1] FROM docker.io/library/busybox:latest@sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138bd458fc8257fbf1
=> resolve docker.io/library/busybox:latest@sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138bd458fc8257fbf1
=> exporting to image
=> exporting layers
=> exporting manifest sha256:ef93a7c4bac1e50b4e172238b0e2c9882624ad0b1f492992745d3f6ff67a4df7
=> exporting config sha256:a296279686634da249b27717cafab6228d7ff975b5874f728ec86a628c2196ac
=> exporting attestation manifest sha256:8b8dbbbfa4d1e279694ec433d0fbc12671261db63fc8b3857111a40945a17bd82
=> exporting manifest list sha256:7a23c94f0e10e3881b10eff09ad49640ae11c7567c730bb3e944b5b2ae187386
=> naming to docker.io/doubleserz/lab8:latest
=> unpacking to docker.io/doubleserz/lab8:latest

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/ui7bwl4okv645ky6gtwy5ebfd
1 warning found (use docker --debug to expand):
- JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 2)

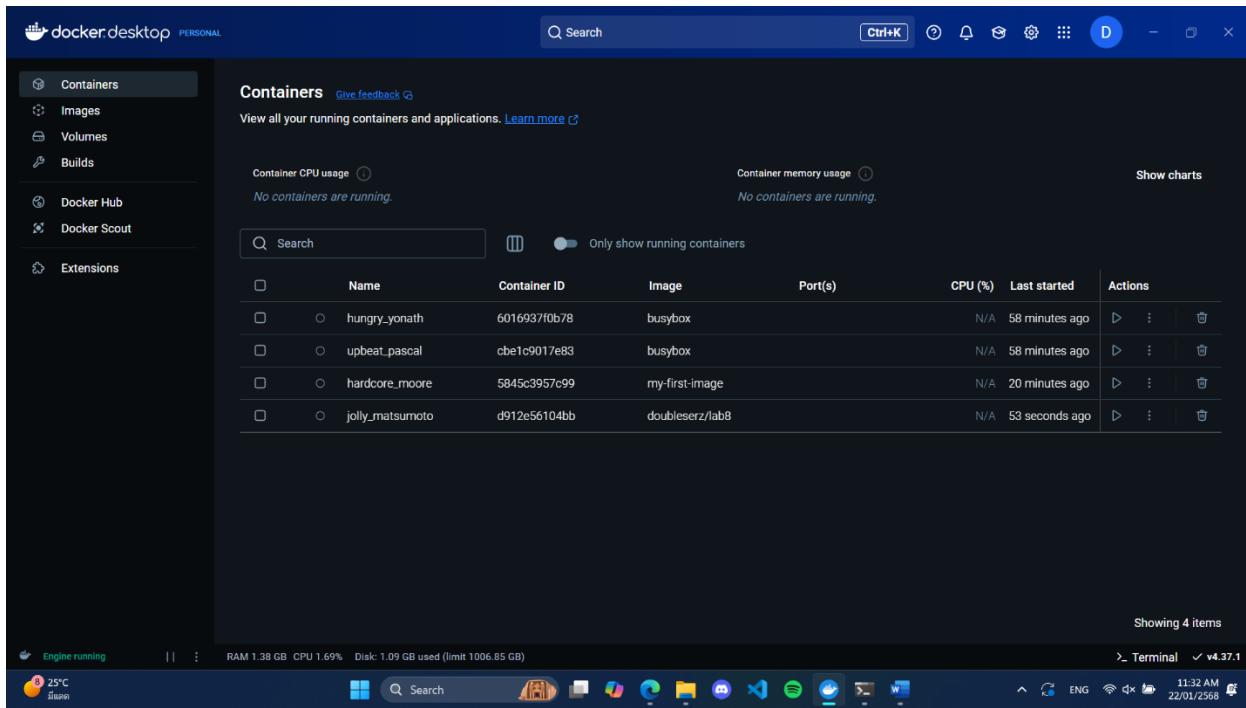
C:\Users\peton\Lab8_3>docker run doubleserz/lab8
Hi there. My work is done. You can run them from my Docker image.
Panyawut Saengdaeng 653380138-3

C:\Users\peton\Lab8_3>

```



## Lab Worksheet



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

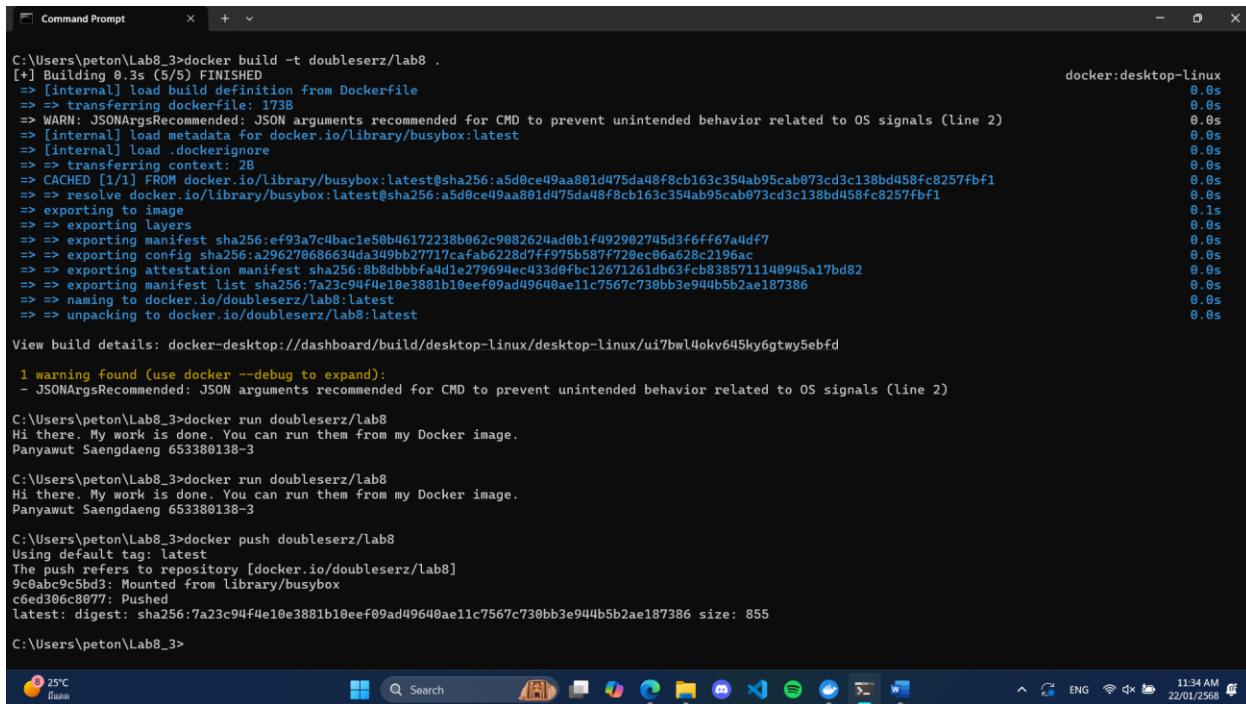
\$ docker login และป้อน Username และ Password ตามที่ระบุใน Command prompt หรือใช้คำสั่ง

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

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

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

## Lab Worksheet



```

C:\Users\peton\Lab8_3>docker build -t doubleserz/lab8 .
[+] Building 0.3s (5/5) FINISHED
=> [internal] load build definition from Dockerfile
=> transferring dockerfile: 173B
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 2)
=> [internal] load metadata for docker.io/library/busybox:latest
=> [internal] load .dockerrcignore
=> [internal] load context: 2B
=> CACHED [1/1] FROM docker.io/library/busybox:latest@sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138bd458fc8257fbf1
=> => resolve docker.io/library/busybox:latest@sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138bd458fc8257fbf1
=> => exporting image
=> => exporting layers
=> => exporting manifest sha256:ef93a7c4bac1e50b46172238b062c9082624ad0b1f492902745d3f6ff67a4df7
=> => exporting config sha256:a296270686634da49bb2771cafab6228d7ff975b587ff720ec86a628c2196ac
=> => exporting attestation manifest sha256:8b8dbbbfa4de279694ec433d0fb12671261db3fc8385711140945a17bd82
=> => exporting manifest list sha256:7a23c94f4e10e3881b10eef09ad49640ae11c7567c730bb3e944b5b2ae187386
=> => naming to docker.io/doubleserz/lab8:latest
=> => unpacking to docker.io/doubleserz/lab8:latest

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/ui7bwl4okv645ky6gtwy5ebfd

1 warning found (use docker --debug to expand):
- JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 2)

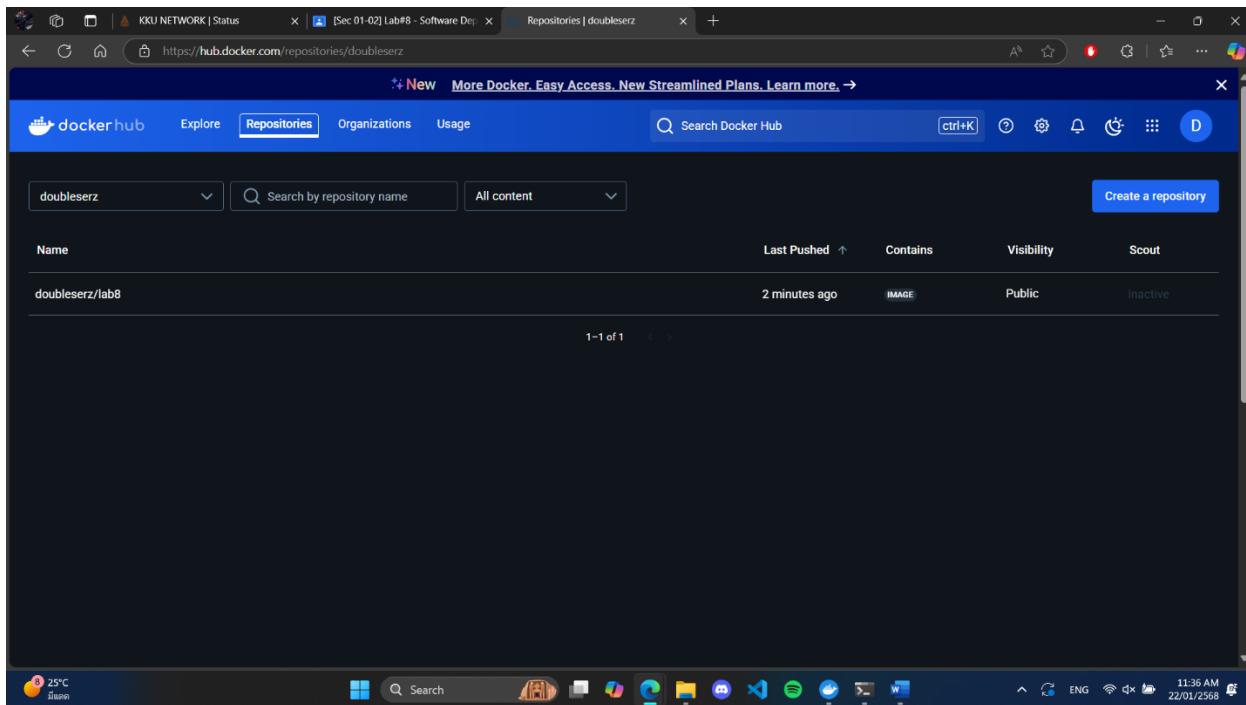
C:\Users\peton\Lab8_3>docker run doubleserz/lab8
Hi there. My work is done. You can run them from my Docker image.
Panyawut Saengdaeng 653380138-3

C:\Users\peton\Lab8_3>docker run doubleserz/lab8
Hi there. My work is done. You can run them from my Docker image.
Panyawut Saengdaeng 653380138-3

C:\Users\peton\Lab8_3>docker push doubleserz/lab8
Using default tag: latest
The push refers to repository [docker.io/doubleserz/lab8]
9c0abc9c5bd3: Mounted from library/busybox
c6ed306c8077: Pushed
latest: digest: sha256:7a23c94f4e10e3881b10eef09ad49640ae11c7567c730bb3e944b5b2ae187386 size: 855

C:\Users\peton\Lab8_3>

```



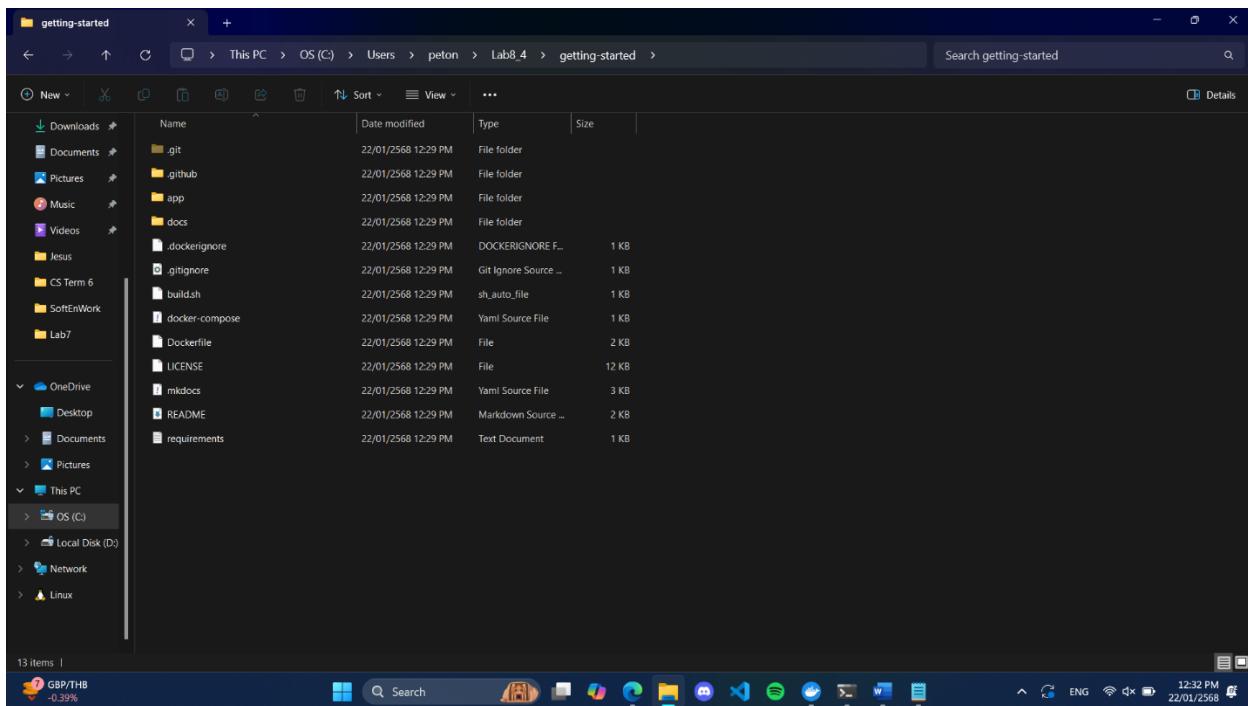
## Lab Worksheet

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

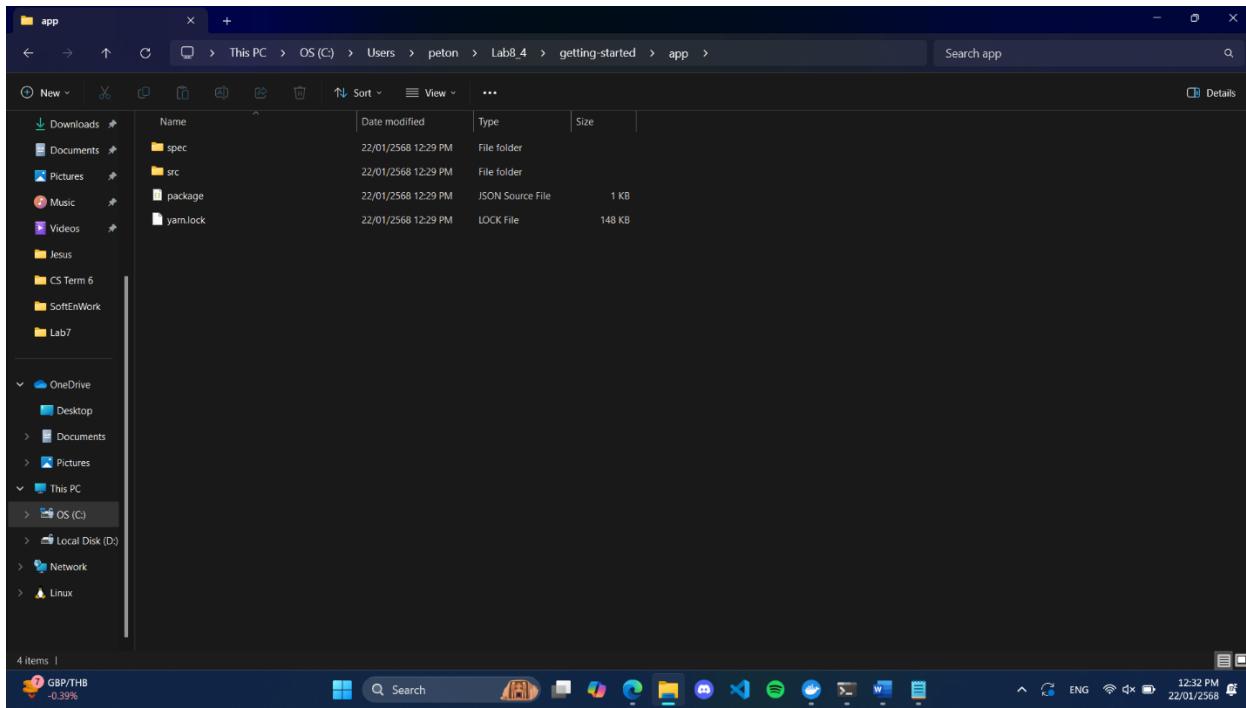
1. เปิด Command line หรือ Terminal จากนั้นสร้าง Directory ชื่อ Lab8\_4
2. ทำการ Clone ซอฟต์แวร์ Docker ที่อยู่ใน GitHub repository  
<https://github.com/docker/getting-started.git> ลงใน Directory ที่สร้างขึ้น โดยใช้คำสั่ง  

```
$ git clone https://github.com/docker/getting-started.git
```
3. เปิดดูองค์ประกอบภายใน getting-started/app เมื่อพับไฟล์ package.json ให้ใช้ Text editor ในการเปิดอ่าน

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



## Lab Worksheet



```

Key.txt      SCIENCE      message.txt      Stage 4 みたい      Untitled      สถานะ STATUS      echo FROM bu...      Dockerfile      package.json
File Edit View

{
  "name": "101-app",
  "version": "1.0.0",
  "main": "index.js",
  "license": "MIT",
  "scripts": {
    "prettify": "prettier -l --write \"**/*.js\"",
    "test": "jest"
  },
  "dev": "nodemon src/index.js"
},
"dependencies": {
  "express": "^4.19.2",
  "mysql2": "^0.2.3",
  "sqlite3": "^5.1.2",
  "uuid": "^9.0.0",
  "wait-port": "1.0.4"
},
"resolutions": {
  "ansi-regex": "5.0.1"
},
"prettier": {
  "trailingcomma": "all",
  "tabWidth": 4,
  "useTabs": false,
  "semi": true,
  "singleQuote": true
},
"devDependencies": {
  "jest": "29.3.1",
  "nodemon": "2.0.20",
  "prettier": "2.7.1"
}
}

Ln 1, Col 1  645 characters
100% Windows (CRLF)  UTF-8
EUR/THB -0.39%

```

## Lab Worksheet

4. ภายใต้ getting-started/app ให้สร้าง Dockerfile พร้อมกับใส่เนื้อหาดังต่อไปนี้ลงไว้ในไฟล์

```
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\_รหัสน

ศ. ไม่มีชีด

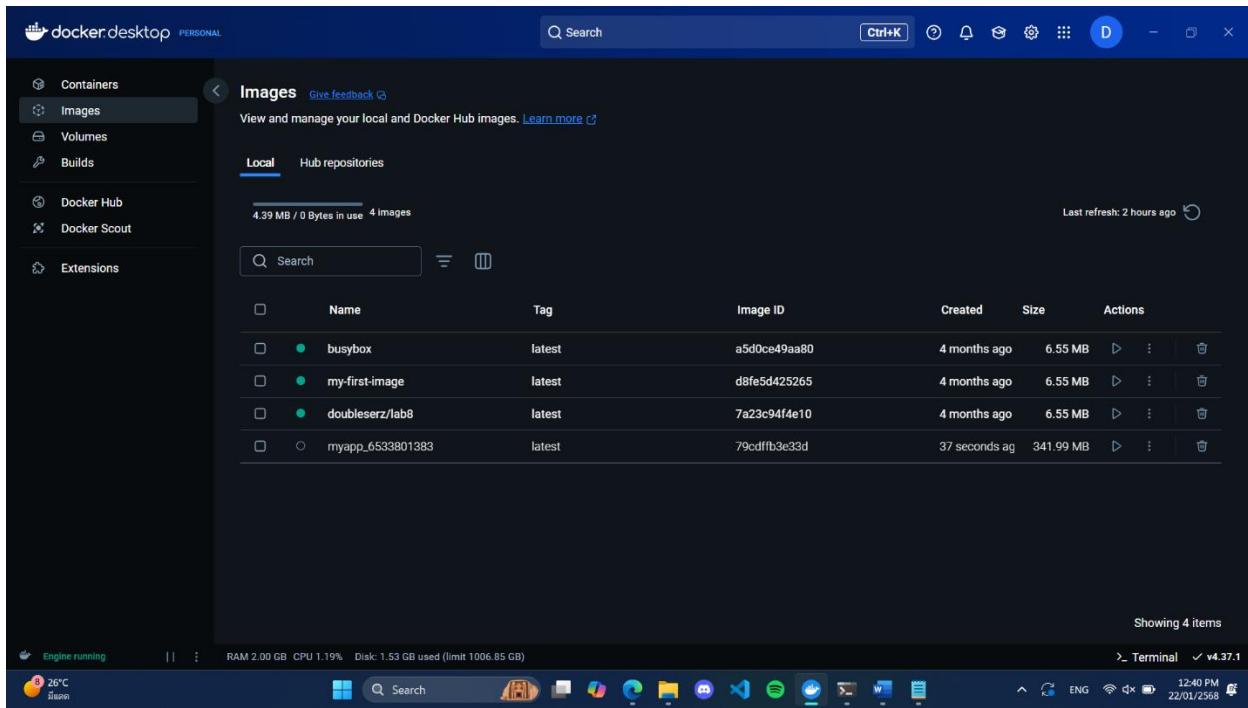
```
$ docker build -t <myapp_รหัสนศ. ไม่มีชีด> .
```

[Check point#8] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงคำสั่งและผลลัพธ์ที่ได้ทางหน้าจอ

```
Command Prompt
5 Dir(s) 31,743,184,896 bytes free
C:\Users\peton\Lab8_4\getting-started>cd app
C:\Users\peton\Lab8_4\getting-started\app>echo. > Dockerfile
C:\Users\peton\Lab8_4\getting-started>docker build -t myapp_6533801383 .
[+] Building 42.2s (10/10) FINISHED
--> [internal] load build definition from Dockerfile
--> transferring dockerfile: 158B
--> [internal] load metadata for docker.io/library/node:18-alpine
--> [auth] library/node:pull token for registry-1.docker.io
--> [internal] load .dockignore
--> transferring context: 2B
--> [1/4] FROM docker.io/library/node:18-alpine@sha256:a24188da7889c2d293ceaa61fb8969ec18821e8efe25572e5abb10b1841eb70b
--> resolve docker.io/library/node:18-alpine@sha256:a24188da7889c2d293ceaa61fb8969ec18821e8efe25572e5abb10b1841eb70b
--> sha256:56147f690fe66bbf54d1dbd27cad26d5b909e09c5f7d5dbe9684488f24b7 1.26MB / 1.26MB
--> sha256:be1198f7193c512f5ccb05ecbd736ff146c06df9edb4691aeeea6b87c8dbf0b 445B / 445B
--> sha256:4e8e5b9bf1c9499b5ef0f37c3dbeba9b0e0cdde6f5b6d289e093f5617ad4a3a5 40.00MB / 40.00MB
--> sha256:1f3e46996e2966e4fa5846e56e76e3748b7315e20ed61476c24493d592134f0 3.64MB / 3.64MB
--> extracting sha256:1f3e46996e2966e4fa5846e56e76e3748b7315e20ed61476c24493d592134f0
--> extracting sha256:4e8e5b9bf1c9499b5ef0f37c3dbeba9b0e0cdde6f5b6d289e093f5617ad4a3a5
--> extracting sha256:56147f690fe66bbf54d1dbd27cad26d5b909e09c5f7d5dbe9684488f24b7
--> extracting sha256:be1198f7193c512f5ccb05ecbd736ff146c06df9edb4691aeeea6b87c8dbf0b
--> [internal] load build context
--> transferring context: 4.62MB
--> [2/4] WORKDIR /app
--> [3/4] COPY . .
--> [4/4] RUN yarn install --production
--> exporting to image
--> exporting layers
--> exporting manifest sha256:f2d3aa8fb6221a9a1656ec9fcfa2933d6fa49ababb6039272f60f38719a790cbf
--> exporting config sha256:e99accadac1cef6a0e8438a34d10f2014ea14de8856da01b1e2939ea6b06
--> exporting attestation manifest sha256:79b5f50695ad7149114a0bfff44fa588bc2e053143c75ca70fe9729032ca48cc7
--> exporting manifest list sha256:79cdff3e33d2b99093aae0c272182eaa5b92b70297e2a7ed3b0e1c896b54bc3c
--> naming to docker.io/library/myapp_6533801383:latest
--> unpacking to docker.io/library/myapp_6533801383:latest

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/4ca4n4lx8l86ifk7ystjga17g
C:\Users\peton\Lab8_4\getting-started\app>
```

## Lab Worksheet



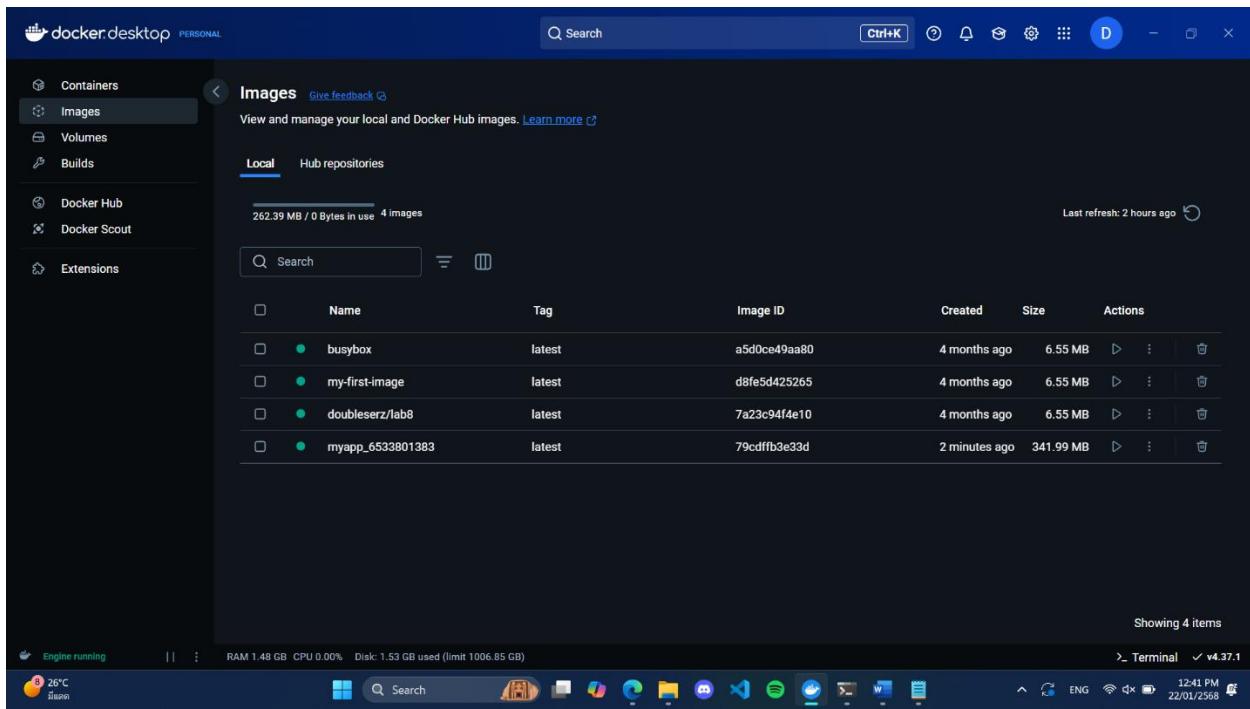
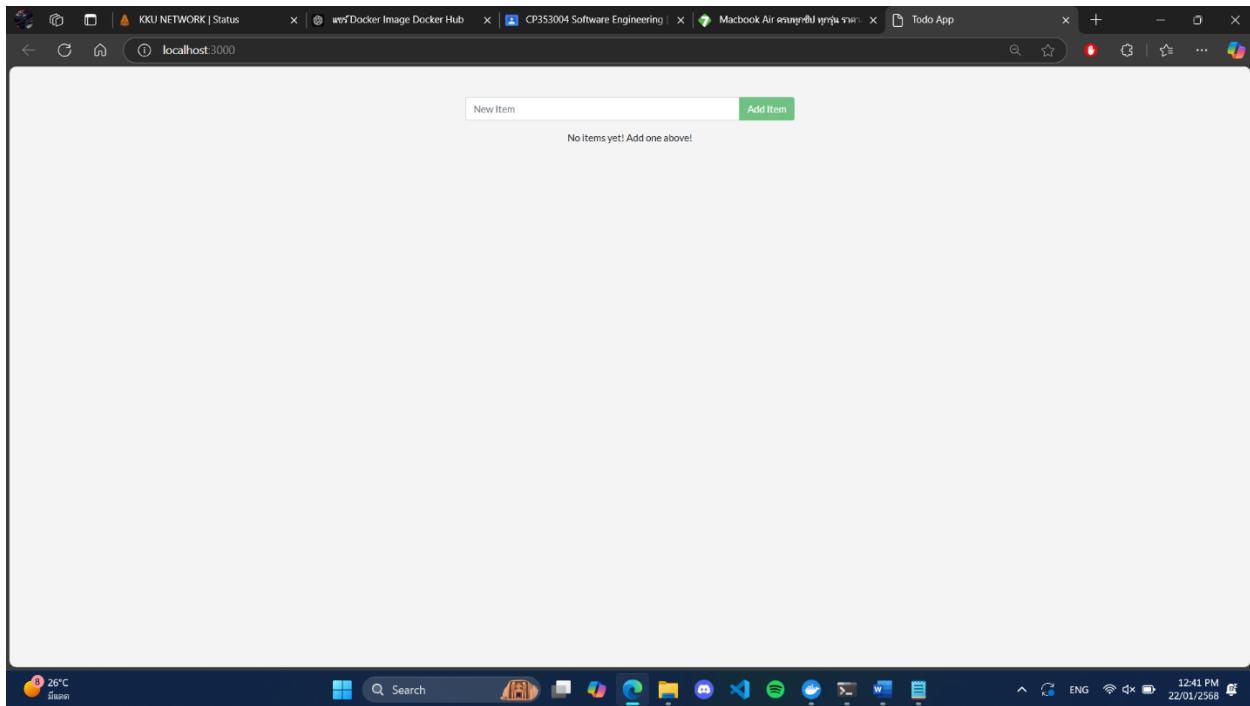
6. ทำการ Start ตัว Container ของแอปพลิเคชันที่สร้างขึ้น โดยใช้คำสั่ง

\$ docker run -dp 3000:3000 <myapp\_รหัสสค. ไม่มีขีด>

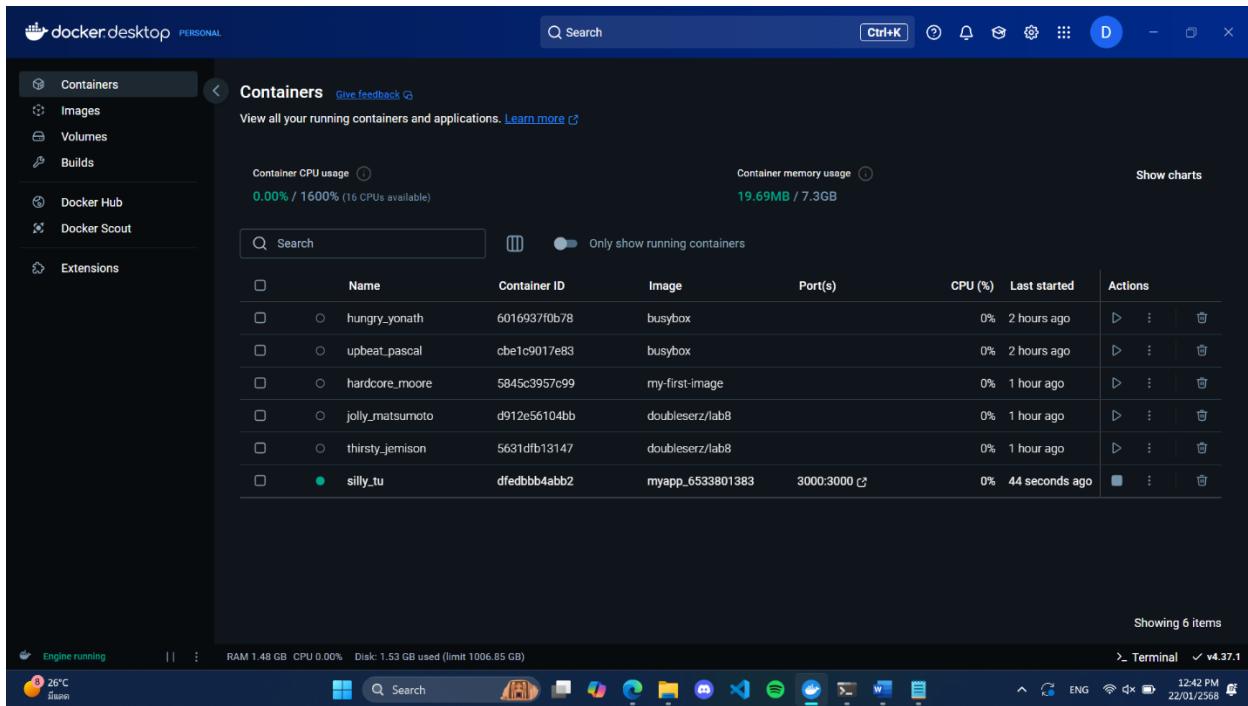
7. เปิด Browser ไปที่ URL = <http://localhost:3000>

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

## Lab Worksheet



## Lab Worksheet



หมายเหตุ: นศ.สามารถทดสอบลงเล่น Web application ที่ทำงานอยู่ได้

8. ทำการแก้ไข Source code ของ Web application ดังนี้

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

< p className="text-center">No items yet! Add one above! </p> เป็น

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

By ชื่อและนามสกุลของนักศึกษา

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

```
Command Prompt x + - 0.1s
=> => exporting manifest list sha256:79cdfffb3e33d2b99093aee8c272182eaa5b92b70297e2a7ed30e1c896b54bc3c
=> => naming to docker.io/library/myapp_6533801383:latest 0.0s
=> => unpacking to docker.io/library/myapp_6533801383:latest 3.7s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/4ca4n41x8l86ifk7ystjga17g

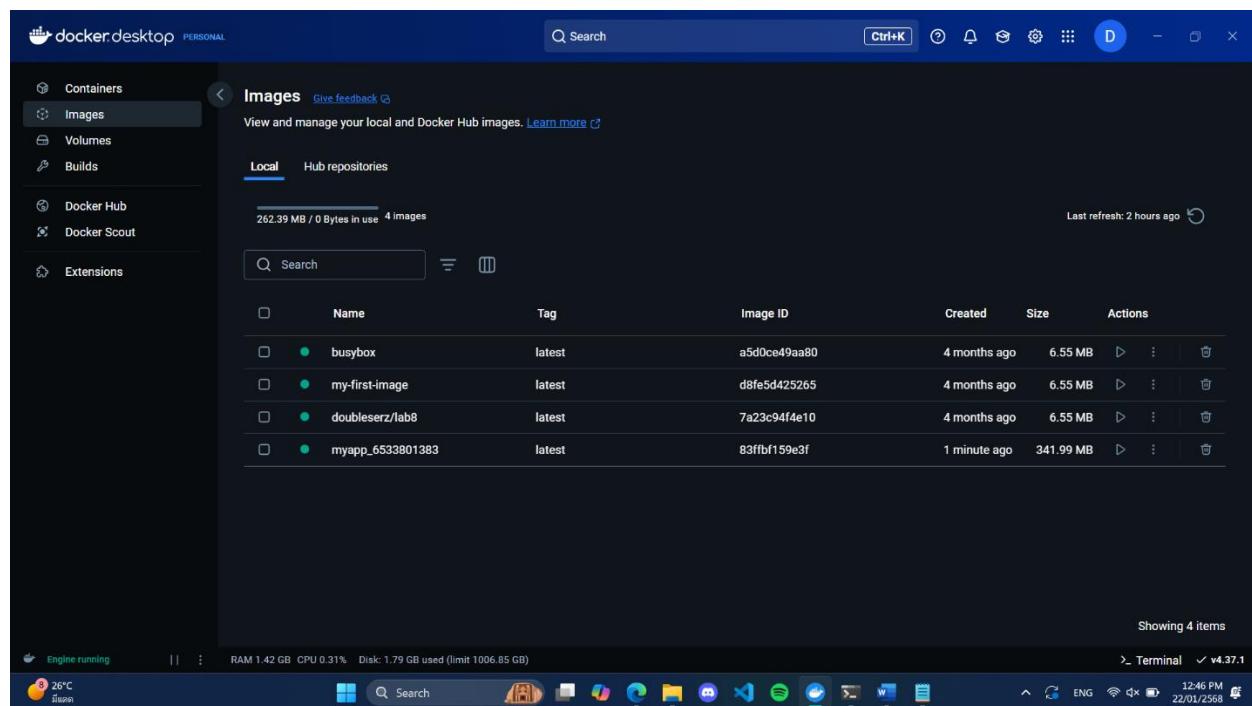
C:\Users\peton\Lab8_4\getting-started\app>docker run -dp 3000:3000 myapp_6533801383
dfedb84abb2cb1325029a6f2cdbcfc4e4e2145e5d57a4e8e0015d0058bd3da

C:\Users\peton\Lab8_4\getting-started\app>docker build -t myapp_6533801383 .
[+] Building 29.1s (10/10) FINISHED
=> [internal] load build definition from Dockerfile 0.0s
=> => transferring dockerfile: 158B 0.0s
=> [internal] load metadata for docker.io/library/node:18-alpine 2.9s
=> [auth] library/node:pull token for registry-1.docker.io 0.0s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [1/4] FROM docker.io/library/node:18-alpine@sha256:a24108da7089c2d293ceaa61fb8969ec10821e8efe25572e5abb10b1841eb70b 0.1s
=> => resolve docker.io/library/node:18-alpine@sha256:a24108da7089c2d293ceaa61fb8969ec10821e8efe25572e5abb10b1841eb70b 0.0s
=> [internal] load build context 0.1s
=> => transferring context: 8.10kB 0.1s
=> CACHED [2/4] WORKDIR /app 0.0s
=> [3/4] COPY . 0.0s
=> [4/4] RUN yarn install --production 18.1s
=> exporting to image 7.6s
=> => exporting layers 3.8s
=> => exporting manifest sha256:a4f72b304e8a8a3a7d166fe8f43304cb64542001d1dbf8c8958ecd62f5f3d1 0.0s
=> => exporting config sha256:b06fa2aac823d4101da618d89d0e03bd0e66e33263e0d07b2d80b3c8f9e1b617 0.0s
=> => exporting attestation manifest sha256:ed30372e02d7764c5d7fd29a3e791fb4288d2b7ed91143d5d42a4ee5dd1d80 0.0s
=> => exporting manifest list sha256:83f9f4283de673b6d8ce2d0dfe16e8863abe00a5168756977 0.0s
=> => naming to docker.io/library/myapp_6533801383:latest 0.0s
=> => unpacking to docker.io/library/myapp_6533801383:latest 3.6s

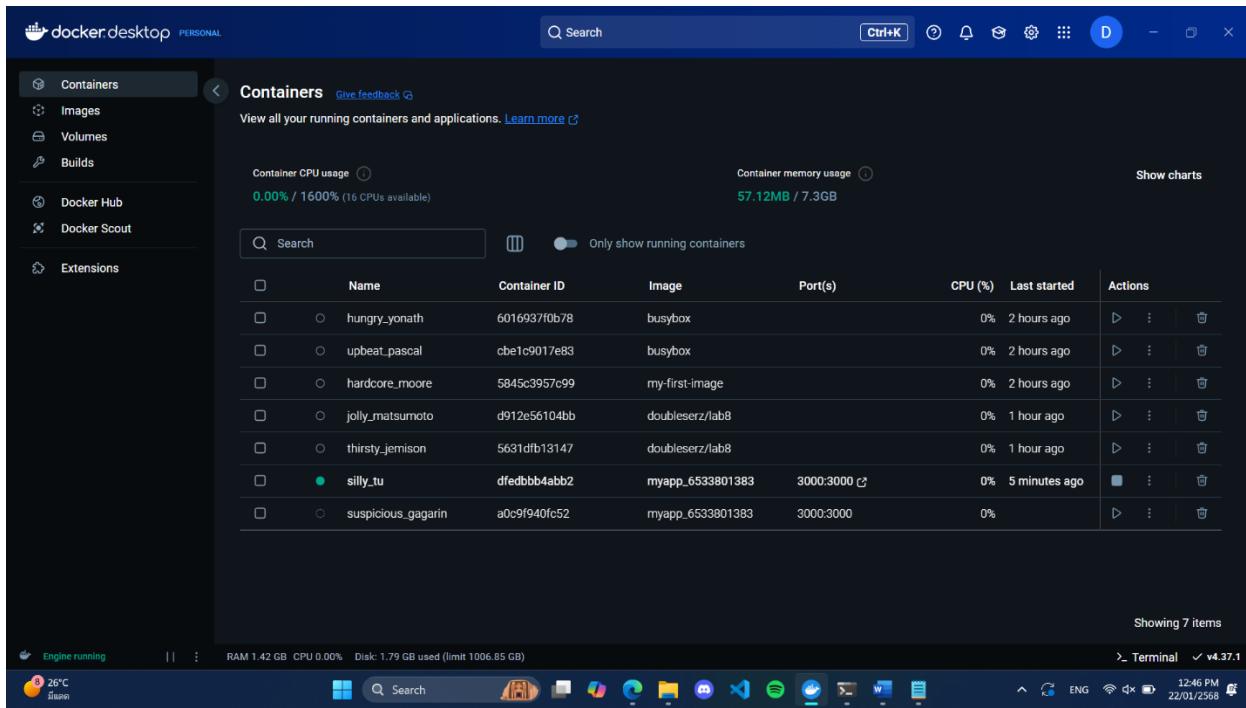
View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/gz7y1bnv7ux7l6rtpfgisc3n

C:\Users\peton\Lab8_4\getting-started\app>docker run -dp 3000:3000 myapp_6533801383
a0c9f940fc52e2745488bc7663345e5ee6dec05d6997ed59691b15e25fee19891
docker: Error response from daemon: driver failed programming external connectivity on endpoint suspicious_gagarin (78ac91624aa07b75d130105649c33761d40118c3
324517302ced4b69b2e358b5): Bind for 0.0.0.0:3000 failed: port is already allocated.

C:\Users\peton\Lab8_4\getting-started\app|
```



## Lab Worksheet



(1) Error ที่เกิดขึ้นหมายความอย่างไร และเกิดขึ้นเพราะอะไร

มีพอร์ต 3000 ถูกใช้งานอยู่โดย container ตัวเด่า ทำให้ไม่สามารถเริ่ม container ใหม่ได้

11. ลบ Container ของ Web application เวอร์ชันก่อนแก้ไขออกจากระบบ โดยใช้วิธีใดวิธีหนึ่งดังต่อไปนี้

a. ผ่าน Command line interface

- ใช้คำสั่ง \$ docker ps เพื่อดู Container ID ที่ต้องการจะลบ
- Copy หรือบันทึก Container ID ไว้
- ใช้คำสั่ง \$ docker stop <Container ID> ที่ต้องการจะลบ เพื่อหยุดการทำงานของ Container ดังกล่าว
- ใช้คำสั่ง \$ docker rm <Container ID> ที่ต้องการจะลบ เพื่อทำการลบ

b. ผ่าน Docker desktop

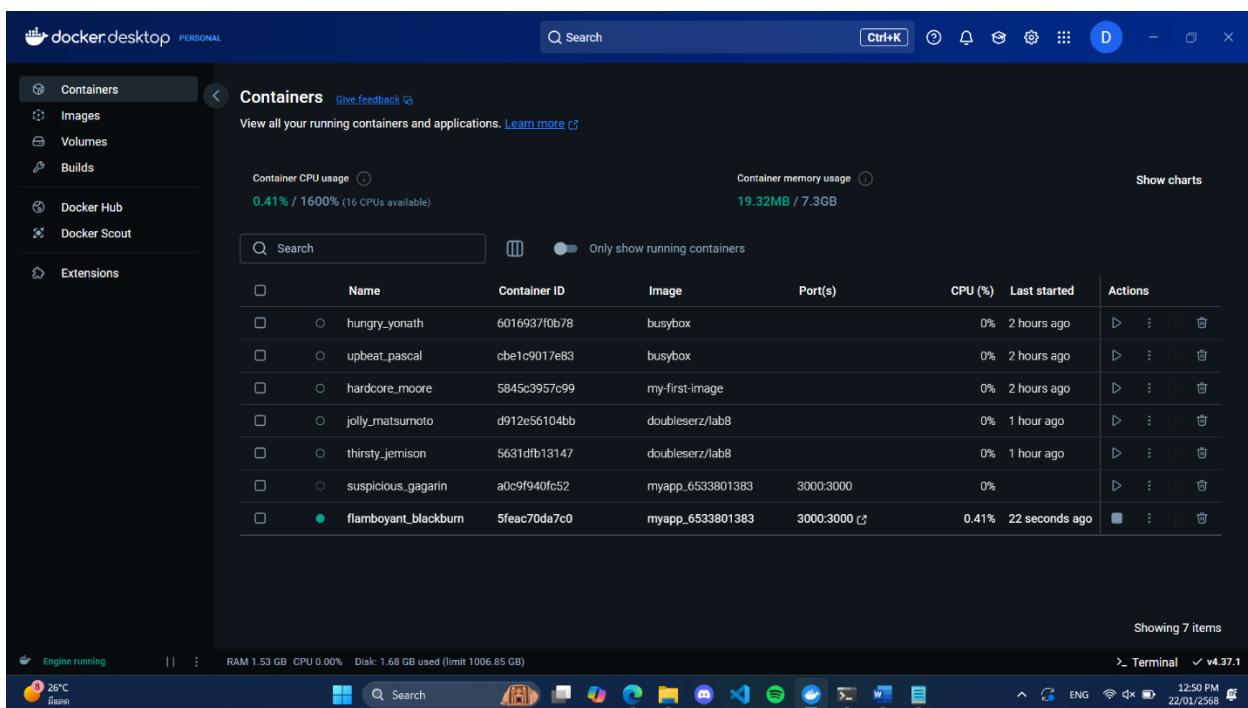
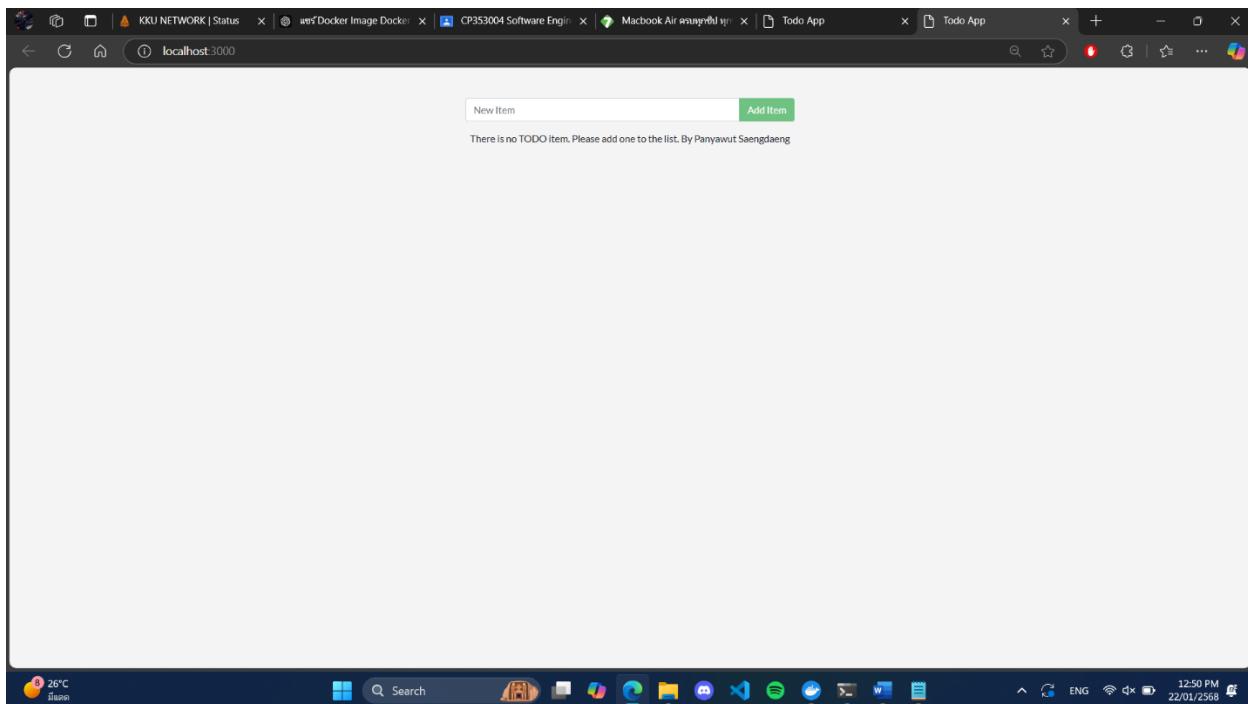
- ไปที่หน้าต่าง Containers
- เลือกไอคอนถังขยะในແຕວของ Container ที่ต้องการจะลบ
- ยืนยันโดยการกด Delete forever

12. Start และรัน Container ตัวใหม่อีกรัง โดยใช้คำสั่งเดียวกันกับข้อ 6

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

## Lab Worksheet

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



## Lab Worksheet

## แบบฝึกปฏิบัติที่ 8.5: เริ่มต้นสร้าง Pipeline อย่างง่ายสำหรับการ Deploy ด้วย Jenkins

1. เปิด Command line หรือ Terminal บน Docker Desktop

2. ป้อนคำสั่งและทำการรัน container โดยผ่านพอร์ต

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

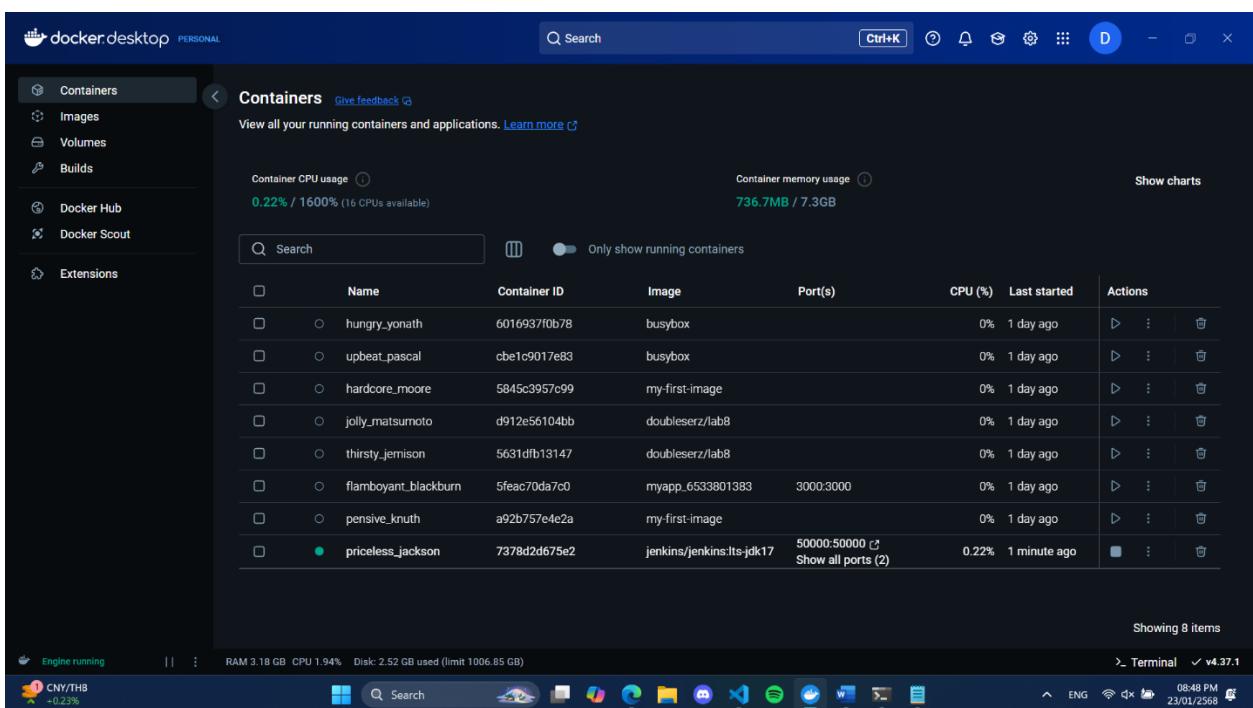
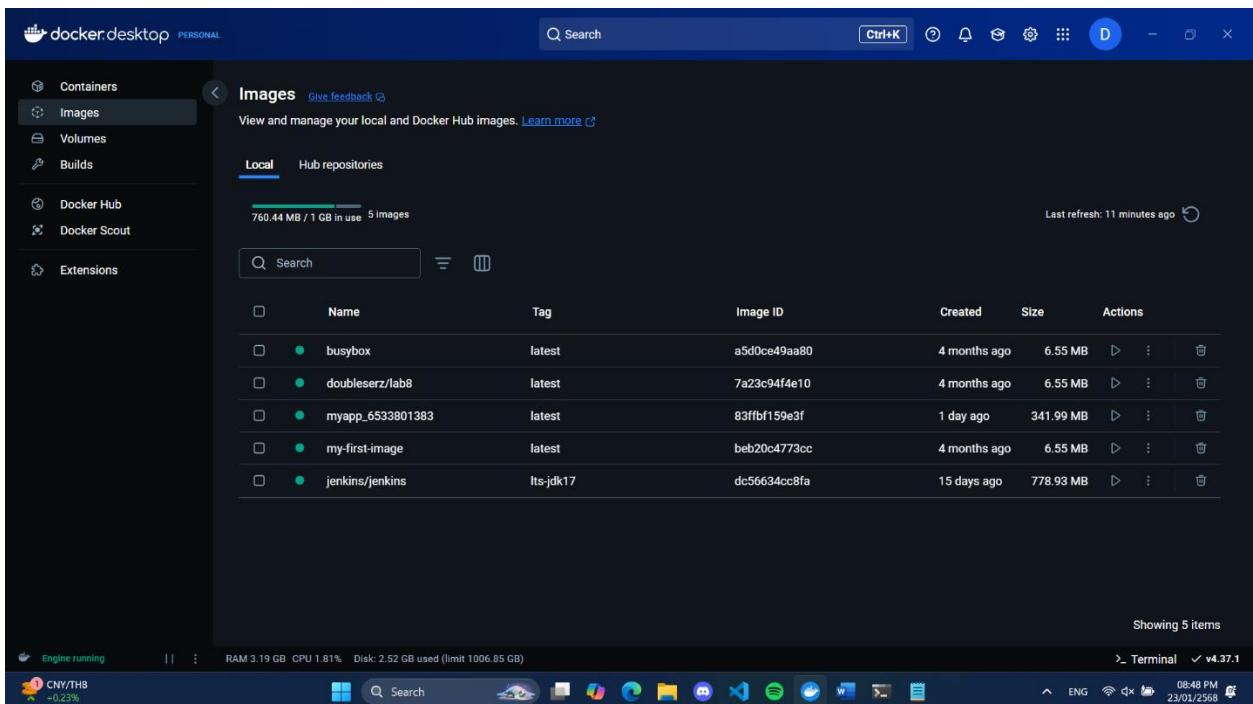
```
$ docker run -p 8080:8080 -p 50000:50000 --restart=on-failure -v
jenkins_home:/var/jenkins_home jenkins/jenkins:lts-jdk17
```

3. บันทึกรหัสผ่านของ Admin user ไว้สำหรับ log-in ในครั้งแรก

**[Check point#12] Capture หน้าจอที่แสดงผล Admin password**

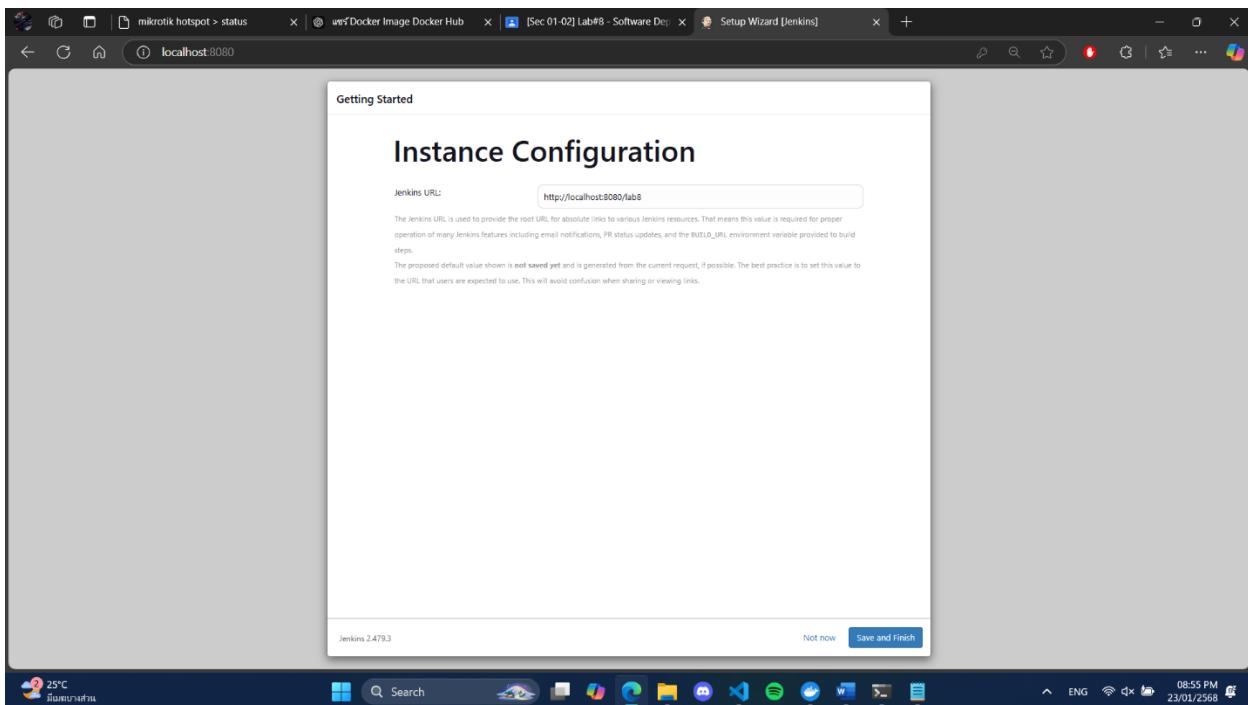
```
2025-01-23 13:47:29.388+0000 [id=58] INFO jenkins.InitReactorRunner$1#onAttained: System config adapted
2025-01-23 13:47:29.389+0000 [id=45] INFO jenkins.InitReactorRunner$1#onAttained: Loaded all jobs
2025-01-23 13:47:29.393+0000 [id=45] INFO jenkins.InitReactorRunner$1#onAttained: Configuration for all jobs updated
2025-01-23 13:47:29.428+0000 [id=85] INFO hudson.util.Retriger$Start: Attempt #1 to do the action check updates server
2025-01-23 13:47:29.877+0000 [id=61] INFO jenkins.install.SetupWizard#init:
*****
Jenkins initial setup is required. An admin user has been created and a password generated.
Please use the following password to proceed to installation:
c3cf3ccc11a74d929a77df2b22bb09c7
This may also be found at: /var/jenkins_home/secrets/initialAdminPassword
*****
2025-01-23 13:47:35.381+0000 [id=61] INFO jenkins.InitReactorRunner$1#onAttained: Completed initialization
2025-01-23 13:47:35.413+0000 [id=33] INFO hudson.lifecycle.Lifecycle$OnReady: Jenkins is fully up and running
2025-01-23 13:47:38.058+0000 [id=85] INFO h.m.DownloadService$Downloadable#load: Obtained the updated data file for hudson.tasks.Maven.MavenInstaller
2025-01-23 13:47:38.059+0000 [id=85] INFO hudson.util.Retriger$Start: Performed the action check updates server successfully at the attempt #1
```

## Lab Worksheet



## Lab Worksheet

4. เมื่อได้รับการยืนยันว่า Jenkins is fully up and running ให้เปิดбраузอร์ และป้อนที่อยู่เป็น localhost:8080
  5. ทำการ Unlock Jenkins ด้วยรหัสผ่านที่ได้ในข้อที่ 3
  6. สร้าง Admin User โดยใช้ username เป็นชื่อจริงของนักศึกษาพร้อมรหัสสี่ตัวท้าย เช่น somsri\_3062
- [Check point#13] Capture หน้าจอที่แสดงผลการตั้งค่า



7. กำหนด Jenkins URL เป็น <http://localhost:8080/lab8>
8. เมื่อติดตั้งเรียบร้อยแล้วจะพบกันหน้า Dashboard ดังแสดงในภาพ

## Lab Worksheet

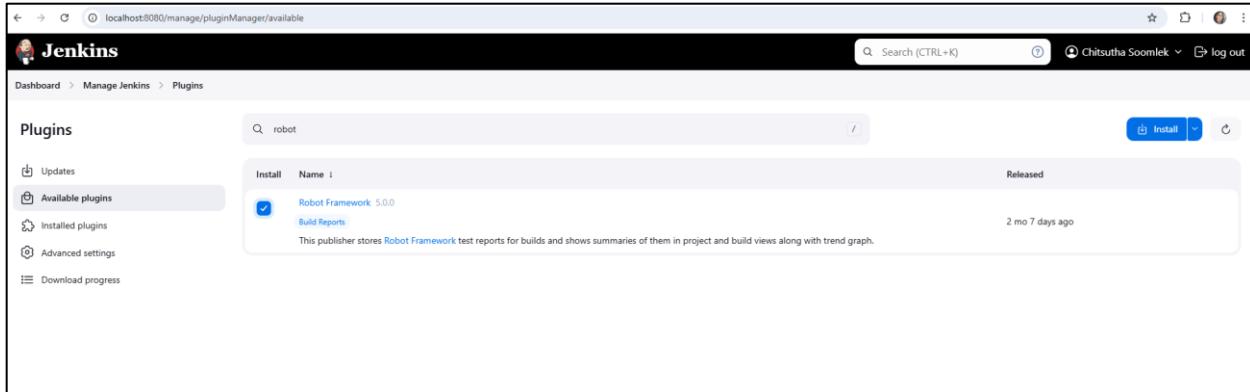
The screenshot shows the Jenkins dashboard at [localhost:8080](http://localhost:8080). The main header says "Jenkins". The left sidebar has links for "Dashboard", "+ New Item", "Build History", "Manage Jenkins", and "My Views". The main content area has a heading "Welcome to Jenkins!". Below it, a message says "This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project." There are three main sections: "Start building your software project" (with "Create a job" and a "+" button), "Set up a distributed build" (with "Set up an agent" and a computer icon, "Configure a cloud" and a cloud icon, and a link "Learn more about distributed builds"), and "Build Queue" (with a message "No builds in the queue." and a dropdown menu). At the bottom right, it says "REST API Jenkins 2.479.3".

### 9. เลือก Manage Jenkins แล้วไปที่เมนู Plugins

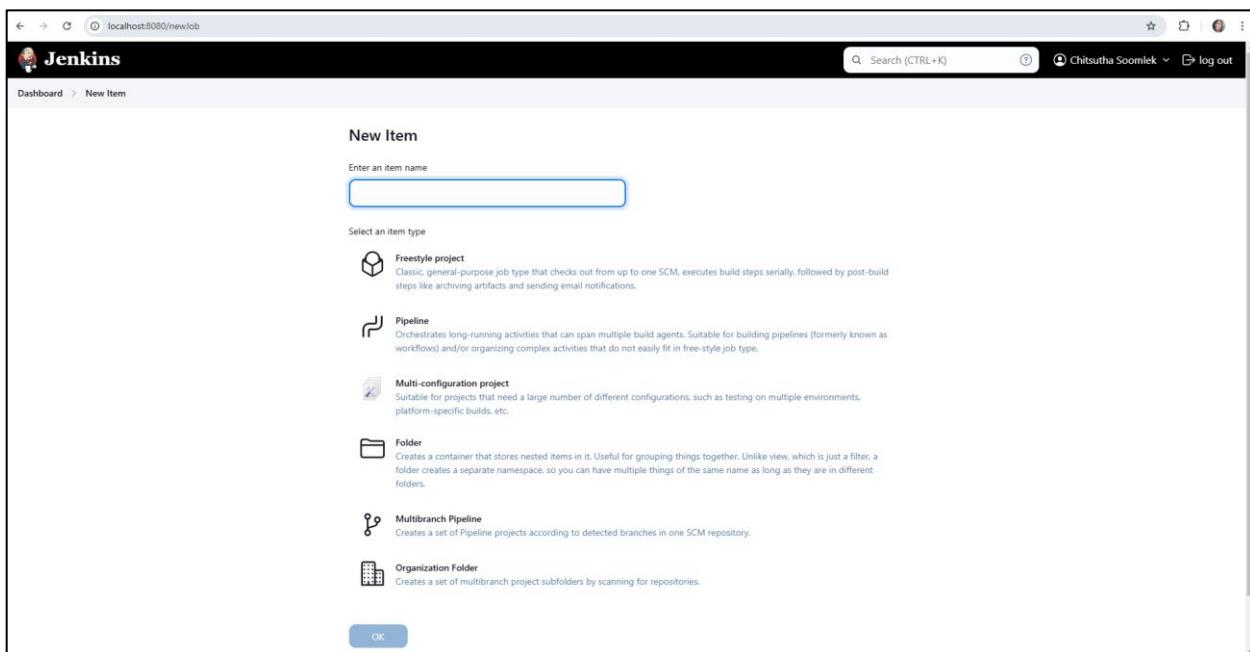
The screenshot shows the "Manage Jenkins" page at [localhost:8080/manage/](http://localhost:8080/manage/). The top navigation bar includes "Dashboard", "Manage Jenkins", "Search (CTRL+K)", and "Chitsutha Soomlek". The main content area has a heading "Manage Jenkins" with a message "It appears that your reverse proxy set up is broken." and buttons "More Info" and "Dismiss". It's divided into several sections: "System Configuration" (with "System", "Tools", "Clouds", "Appearance", and "Nodes" sub-sections), "Security" (with "Security", "Credentials", "Credential Providers", and "Users" sub-sections), "Status Information" (with "System Information", "System Log", "Load Statistics", and "About Jenkins" sub-sections). Each section contains a brief description and a link to its configuration page.

## Lab Worksheet

10. ไปที่เมนู Available plugins แล้วเลือกติดตั้ง Robotframework เพิ่มเติม



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



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

**Description:** Lab 8.5

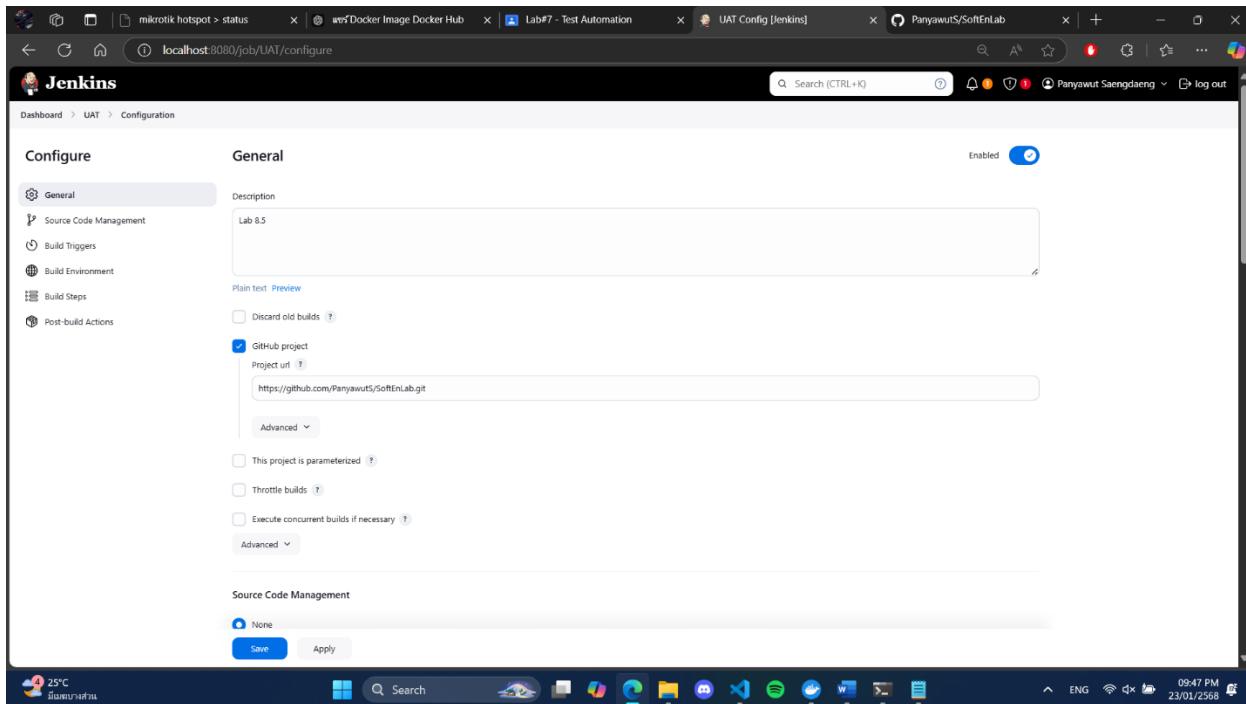
**GitHub project:** กดเลือก และใส่ Project URL เป็น repository ที่เก็บโค้ด .robot (ดูขั้นตอนที่ 12)

**Build Trigger:** เลือกแบบ Build periodically และกำหนดให้ build ทุก 15 นาที

## Lab Worksheet

**Build Steps:** เลือก Execute shell และใส่คำสั่งในการรันไฟล์ .robot (หากไฟล์ไม่ได้อยู่ในหน้าแรกของ repository ให้ใส่ Path ไปถึงไฟล์ให้เรียบร้อยด้วย)

[Check point#14] Capture หน้าจอแสดงการตั้งค่า พร้อมกับตอบคำถามต่อไปนี้



(1) คำสั่งที่ใช้ในการ Execute ไฟล์ .robot ใน Build Steps คือ

D:\File\CS Term 6\SoftEnLab> cd .\Lab7\

D:\File\CS Term 6\SoftEnLab\Lab7> cd .\Lab7\_test\

D:\File\CS Term 6\SoftEnLab\Lab7\Lab7\_test> robot valid\_register.robot

D:\File\CS Term 6\SoftEnLab\Lab7\Lab7\_test> robot invalid\_register.robot

**Post-build action:** เพิ่ม Publish Robot Framework test results -> ระบุไดเร็คทอรีที่เก็บไฟล์ผลการทดสอบโดย Robot framework ในรูป xml และ html -> ตั้งค่า Threshold เป็น % ของการทดสอบที่ไม่ผ่านแล้วนับว่าซอฟต์แวร์มีปัญหา -> ตั้งค่า Threshold เป็น % ของการทดสอบที่ผ่านแล้วนับว่าซอฟต์แวร์มีอยู่ในสถานะที่สามารถนำไปใช้งานได้ ( เช่น 20, 80 )

13. กด Apply และ Save

14. ตั้ง Build Now

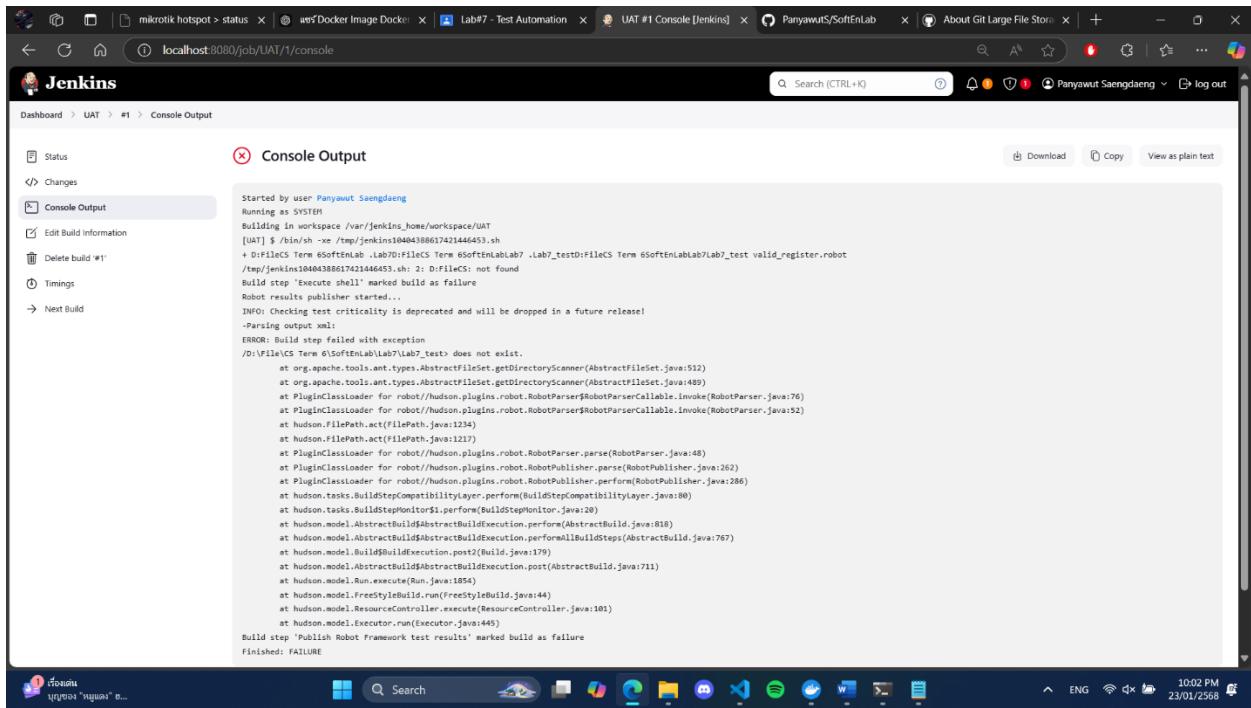
[Check point#15] Capture หน้าจอแสดงหน้าหลักของ Pipeline และ Console Output

## Lab Worksheet

The screenshot shows the Jenkins dashboard for the 'UAT' job. The left sidebar includes options like Status, Changes, Workspace, Build Now, Configure, Delete Project, Robot Results, GitHub, and Rename. The main area displays the latest robot results, which are currently empty. Below that is a section for Permalinks with a list of recent builds. A large table lists the build history from #1 to #7, each with a timestamp of '3:00 PM'. At the bottom right, there are links for REST API and Jenkins 2.479.3.

The screenshot shows the details of the first build (#1) of the 'UAT' job. The top bar indicates the build was started by 'Panyawut Saengdaeng' on 'Jan 23, 2025, 3:00:37 PM'. The build status is marked with a red circle and an 'X'. The build duration is listed as '0.14 sec'. The sidebar on the left provides links for Status, Changes, Console Output, Edit Build Information, Delete build #1, Timings, and Next Build. The main content area shows the build log with the message '</> No changes.'

## Lab Worksheet



```

Started by user Panyawut Saengdaeng
Running as SYSTEM
Building in workspace /var/jenkins_home/workspace/UAT
[UAT] $ /bin/sh -xe /tmp/jenkins1040438866742144653.sh
+ D:\FileCS Term 6SoftEnLab .Lab7\0D\FileCS Term 6SoftEnLabLab7 .Lab7_test0\FileCS Term 6SoftEnLabLab7_lab7_test valid_register.robot
/tmp/jenkins1040438866742144653.sh: 2: D:\FileCS: not found
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
ERROR: Build step failed with exception
/D:/FileCS Term 6SoftEnLabLab7_lab7_test> does not exist.
at org.apache.tools.ant.types.AbstractFileSet.getDirectoryScanner(AbstractFileSet.java:512)
at org.apache.tools.ant.types.AbstractFileSet.getDirectoryScanner(AbstractFileSet.java:489)
at PluginClassLoader.loadForRobot(/hudson/plugins/robot/RobotParser$RobotCallable.invoke(RobotParser.java:76)
at PluginClassLoader.loadForRobot(/hudson/plugins/robot/RobotParser$RobotCallable.invoke(RobotParser.java:52)
at hudson.FilePath.act(/filePath.java:1234)
at hudson.FilePath.act(/filePath.java:1217)
at PluginClassLoader.loadForRobot(/hudson/plugins/robot/RobotParser.parse(RobotParser.java:48)
at PluginClassLoader.loadForRobot(/hudson/plugins/robot/RobotPublisher.parse(RobotPublisher.java:262)
at PluginClassLoader.loadForRobot(/hudson/plugins/robot/RobotPublisher.perform(RobotPublisher.java:286)
at hudson.tasks.BuildStepCompatibilityLayer.perform(BuildStepCompatibilityLayer.java:80)
at hudson.tasks.BuildStepMonitor$1.perform(BuildStepMonitor.java:28)
at hudson.model.AbstractBuild$AbstractBuildExecution.perform(AbstractBuild.java:818)
at hudson.model.AbstractBuild$AbstractBuildExecution.performAllBuildSteps(AbstractBuild.java:767)
at hudson.model.BuildableBuildExecution.post(Build.java:179)
at hudson.model.AbstractBuild$AbstractBuildExecution.post(AbstractBuild.java:711)
at hudson.model.Run.execute(Run.java:1854)
at hudson.model.FreeStyleBuild.run(FreeStyleBuild.java:44)
at hudson.model.ResourceController.execute(ResourceController.java:101)
at hudson.model.Executor.run(Executor.java:445)
Build step 'Publish Robot Framework test results' marked build as failure
Finished: FAILURE

```