

**Lab Worksheet**

ชื่อ-นามสกุล น.ส.สิริวัฒ แก้วคุ้ม รหัสนักศึกษา 653380155-3 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. ผู้เรียนสามารถนำซอฟต์แวร์ที่พัฒนาขึ้นให้สามารถรันบนสภาพแวดล้อมเดียวกันและทำงานร่วมกันกับสมาร์ทโฟนที่มี Docker hub ได้
5. ผู้เรียนสามารถเริ่มต้นใช้งาน Jenkins เพื่อสร้าง Pipeline ในการ Deploy งานได้

**Pre-requisite**

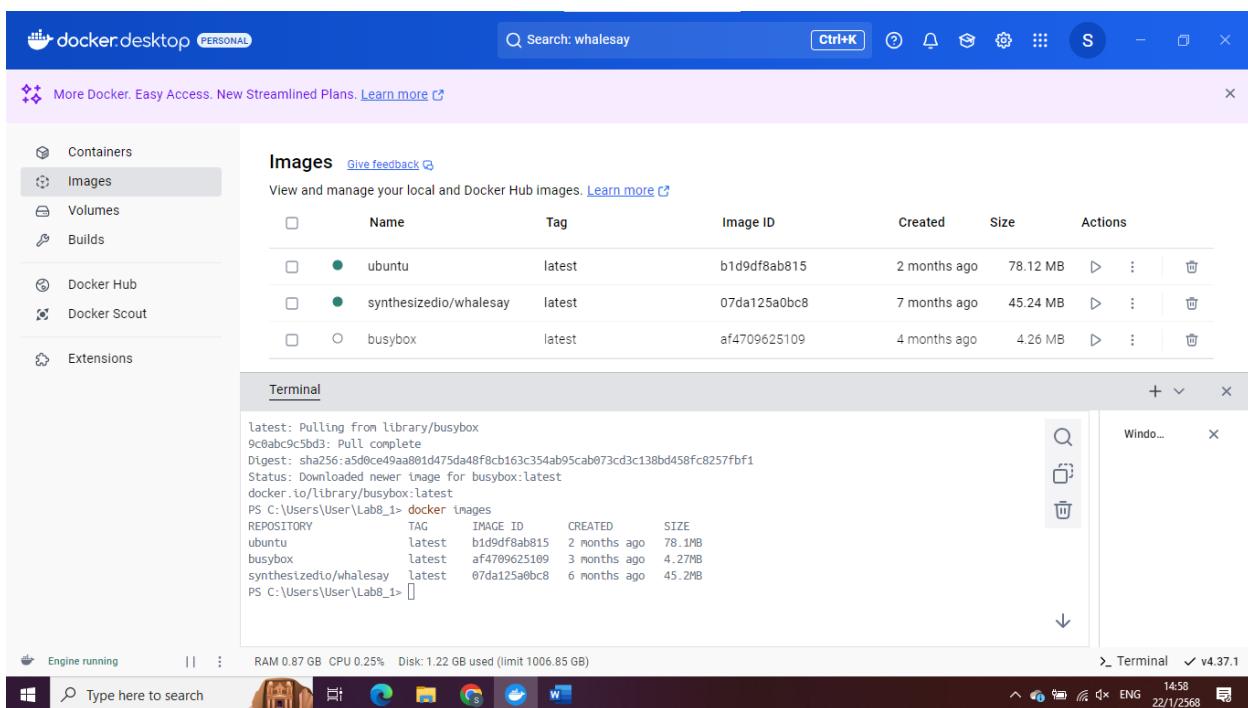
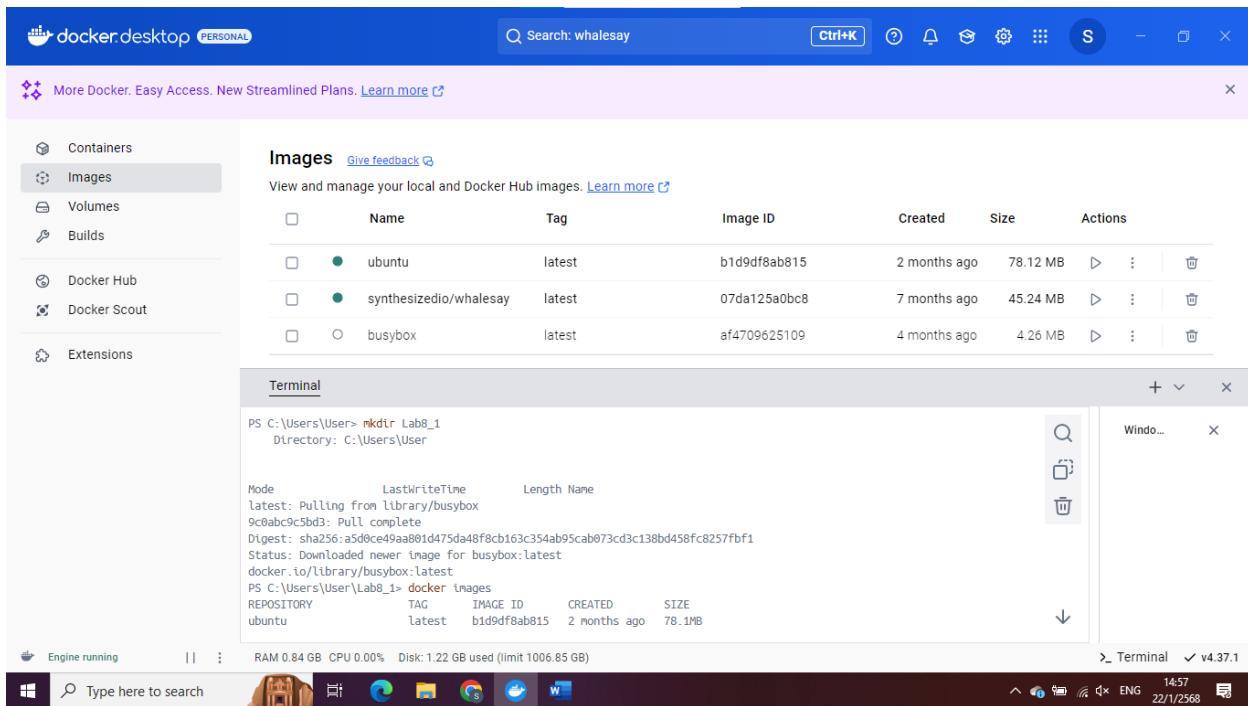
1. ติดตั้ง Docker desktop ลงบนเครื่องคอมพิวเตอร์ โดยดาวน์โหลดจาก <https://www.docker.com/get-started>
2. สร้าง Account บน Docker hub (<https://hub.docker.com/signup>)
3. กำหนดให้ \$ หมายถึง Command prompt และ <> หมายถึง ให้ป้อนค่าของพารามิเตอร์ที่กำหนด

**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) แสดงผลลัพธ์ที่ได้พร้อมกับตอบคำถามต่อไปนี้

## Lab Worksheet

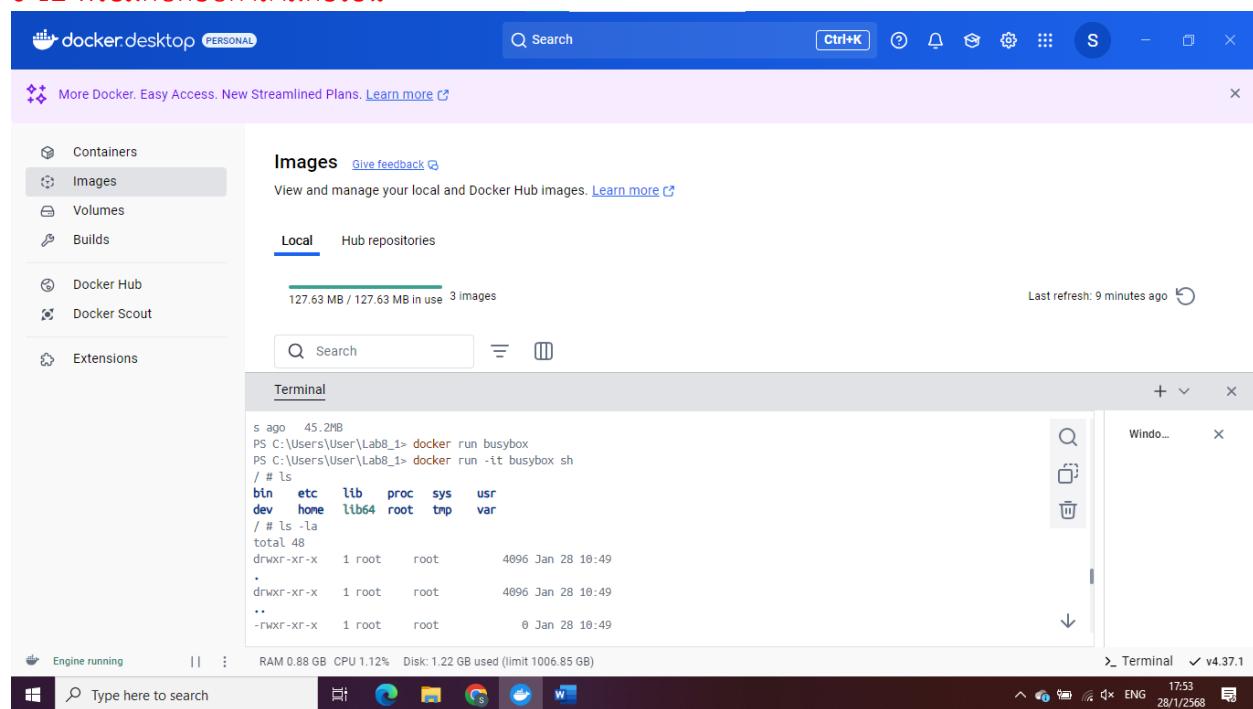


- (1) สิ่งที่อยู่ภายใต้คอลัมน์ Repository คืออะไร ชื่อ docker image
- (2) Tag ที่ใช้บ่งบอกถึงอะไร บอก version ของ docker image

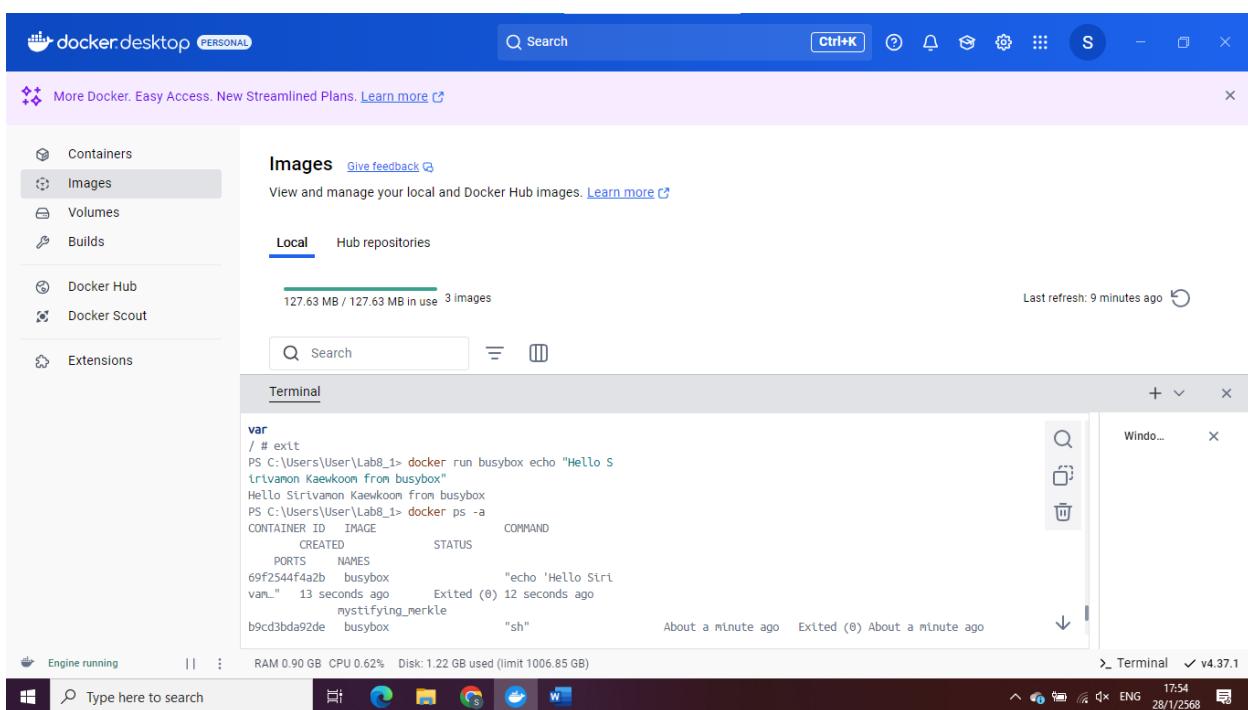
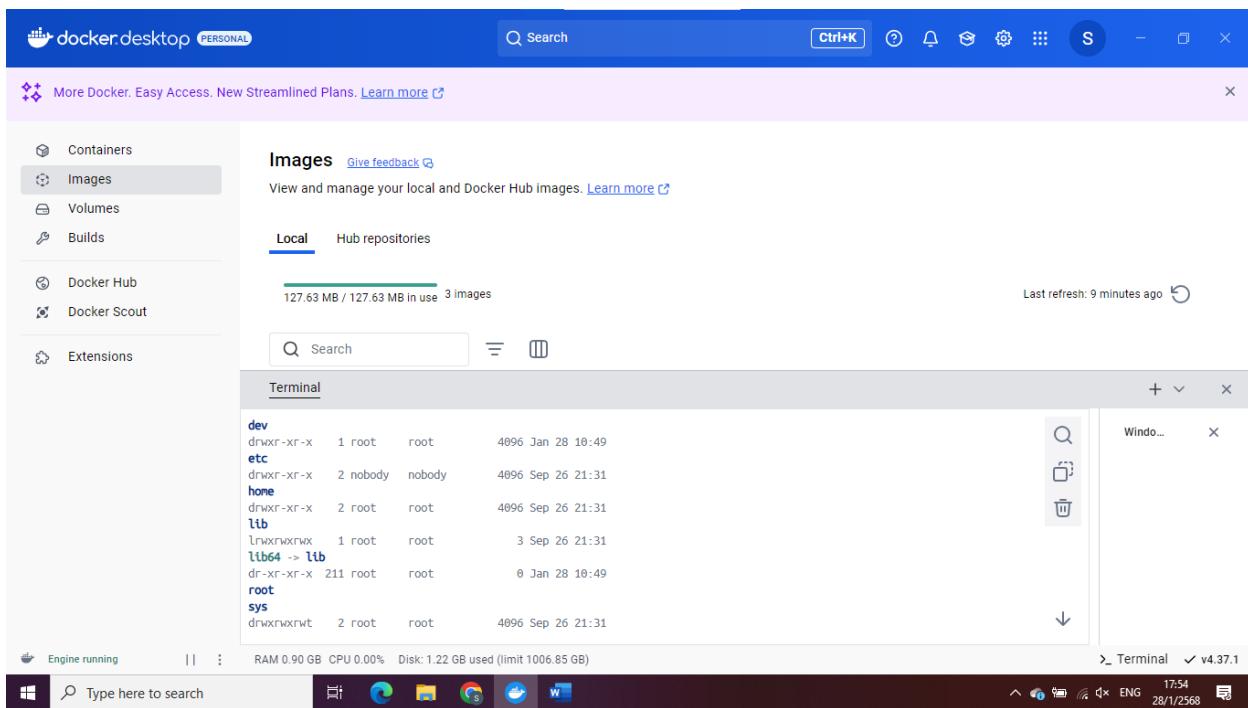
## Lab Worksheet

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

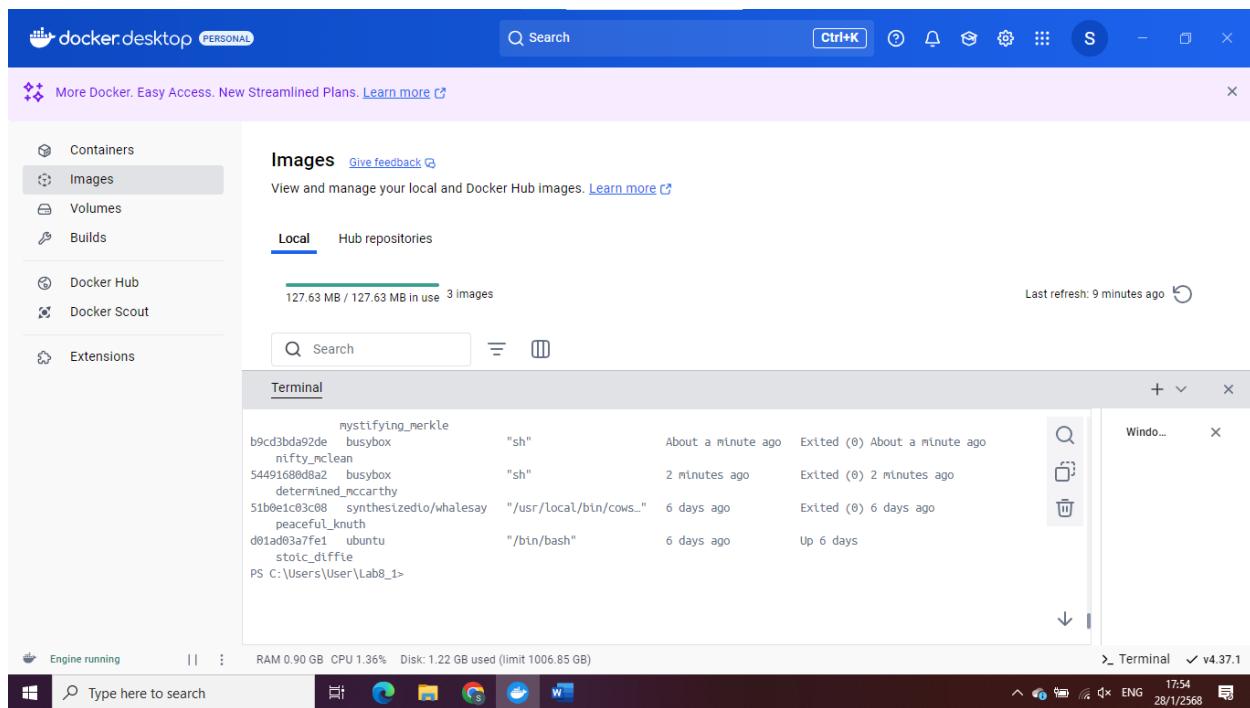
[Check point#2] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงผลลัพธ์ที่ได้ตั้งแต่ขั้นตอนที่ 6-12 พร้อมกับตอบคำถามต่อไปนี้



## Lab Worksheet



## Lab Worksheet



(1) เมื่อใช้ option -it ในคำสั่ง run สำหรับการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสั้นๆ เช่น

รัน container และเปิด shell ทำให้รับคำสั่งใน busybox

i คือ interactive input

t คือ แสดงผลแบบ terminal

(2) คลั่มน์ STATUS จากการรันคำสั่ง docker ps -a และถึงข้อมูลอะไร

แสดงสถานการณ์ทำงานของ container

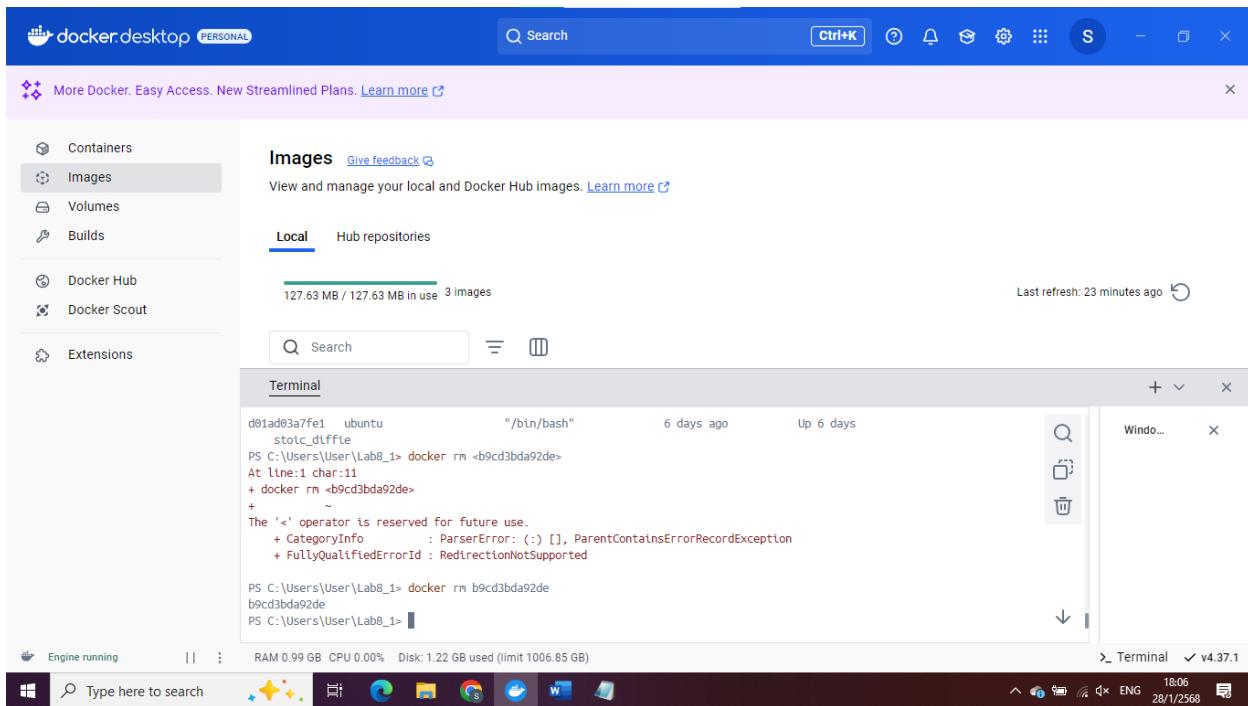
Exited คือ หยุดทำงานแล้ว

Up คือ กำลังทำงาน

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

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

## 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

สำหรับเครื่องที่ใช้ระบบปฏิบัติการวินโดว์ (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

## Lab Worksheet

\$ touch Dockerfile

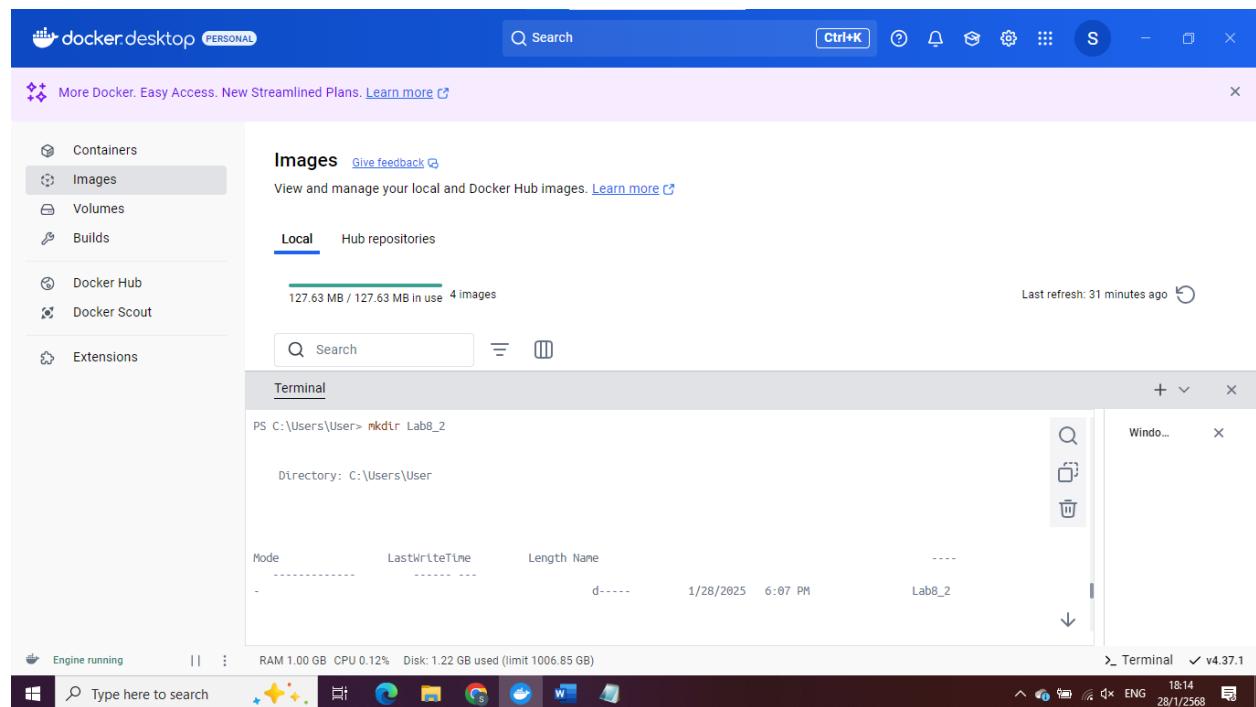
แล้วใช้ Text Editor ในการใส่เนื้อหาแทน

5. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้

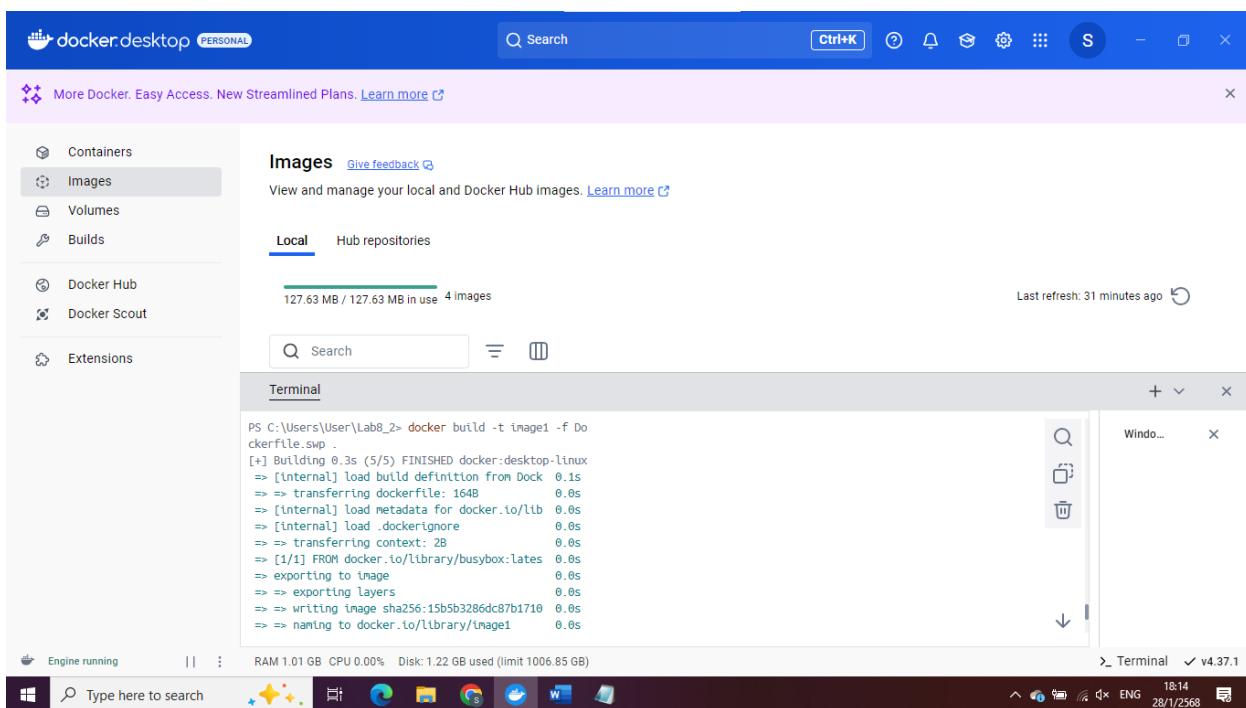
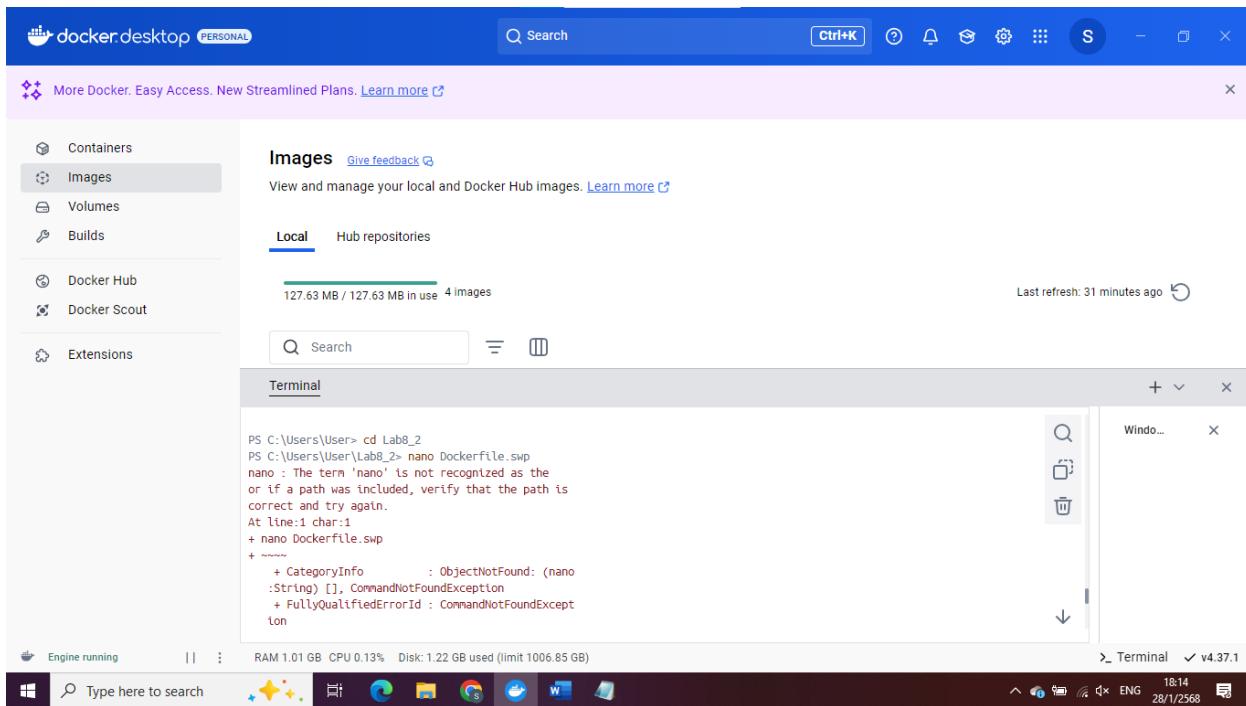
\$ docker build -t <ชื่อ Image> .

6. เมื่อ Build สำเร็จแล้ว ให้ทำการรัน Docker image ที่สร้างขึ้นในขั้นตอนที่ 5

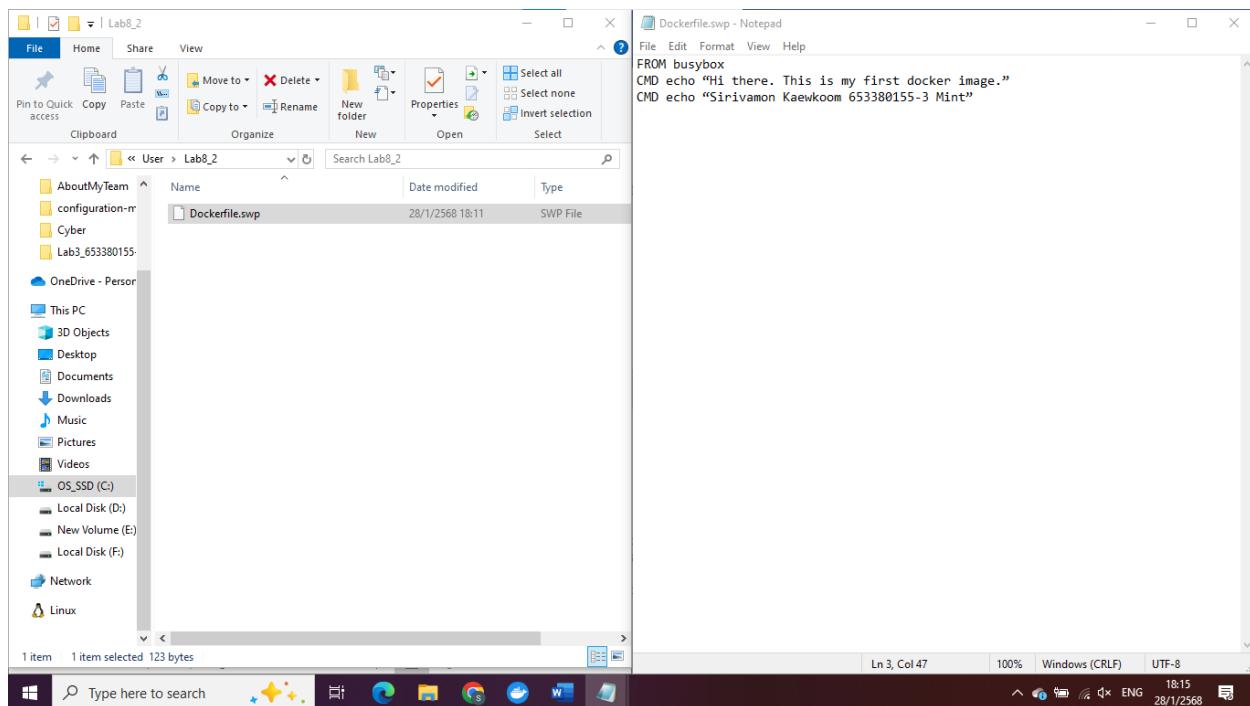
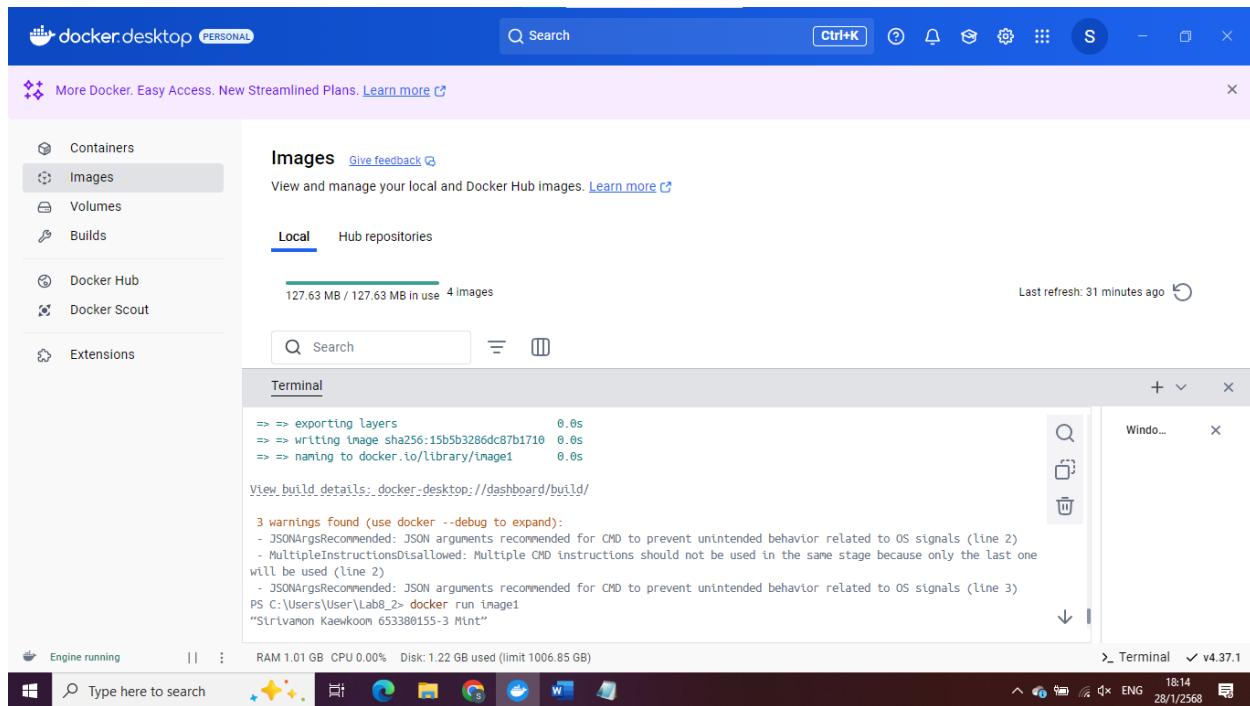
[Check point#4] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงผลลัพธ์ที่ได้ในขั้นตอนที่ 5 พร้อมกับตอบคำถามต่อไปนี้



## Lab Worksheet



## Lab Worksheet



- (1) คำสั่งที่ใช้ในการ run คือ docker run image1
- (2) Option -t ในคำสั่ง \$ docker build ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสั้นๆ

## Lab Worksheet

## 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
- สำหรับเครื่องที่ใช้ระบบปฏิบัติการwinдовส์ 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
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
```

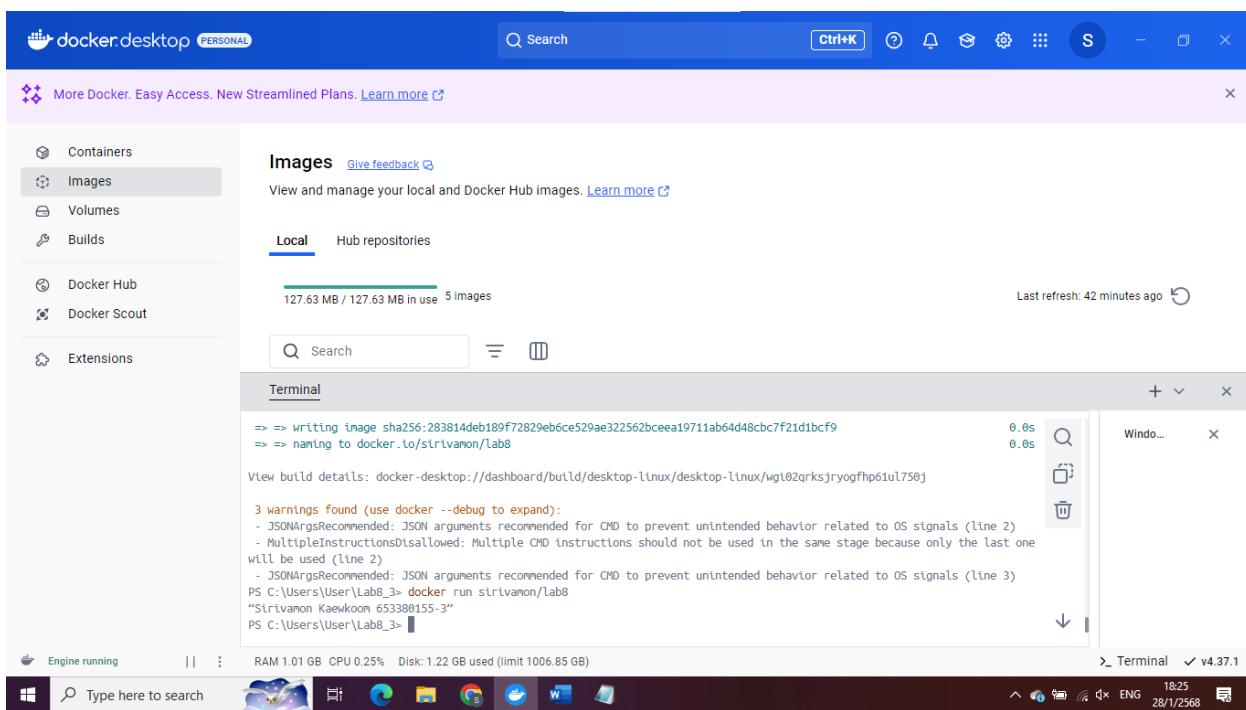
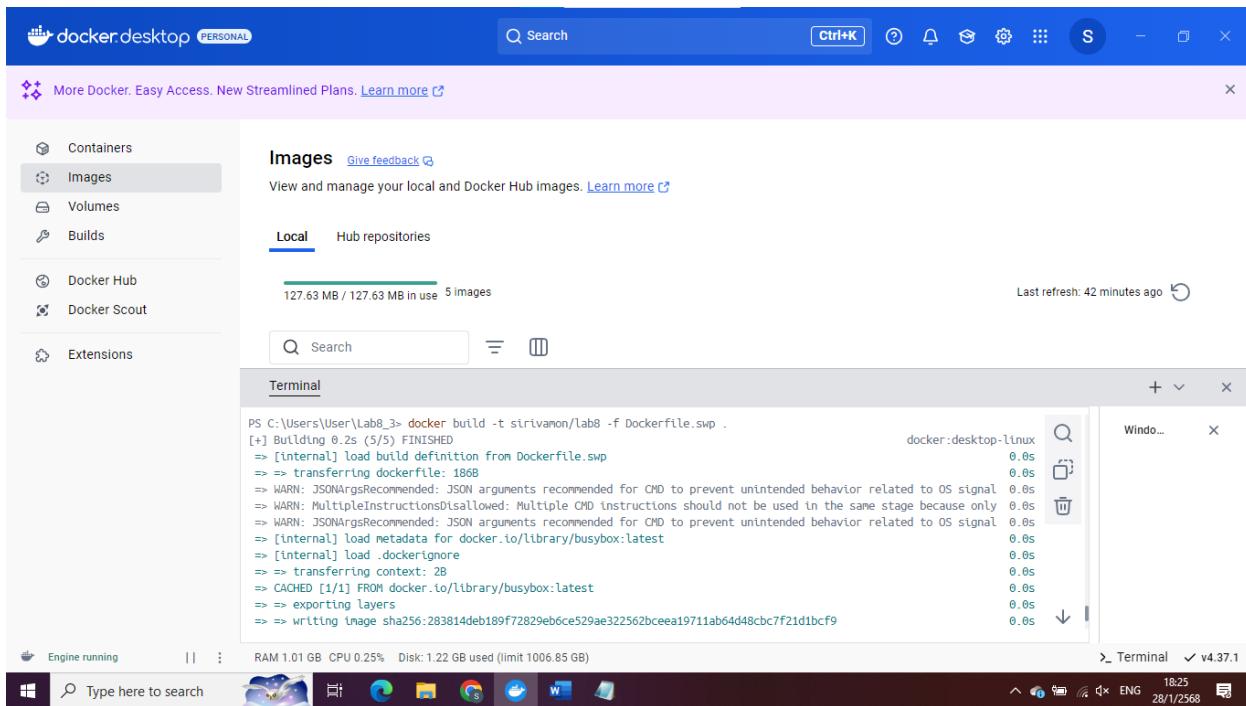
7. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้
 

```
$ docker build -t <username> ที่ลงทะเบียนกับ Docker Hub>/lab8
```
5. ทำการรัน Docker image บน Container ในเครื่องของตัวเองเพื่อทดสอบผลลัพธ์ ด้วยคำสั่ง
 

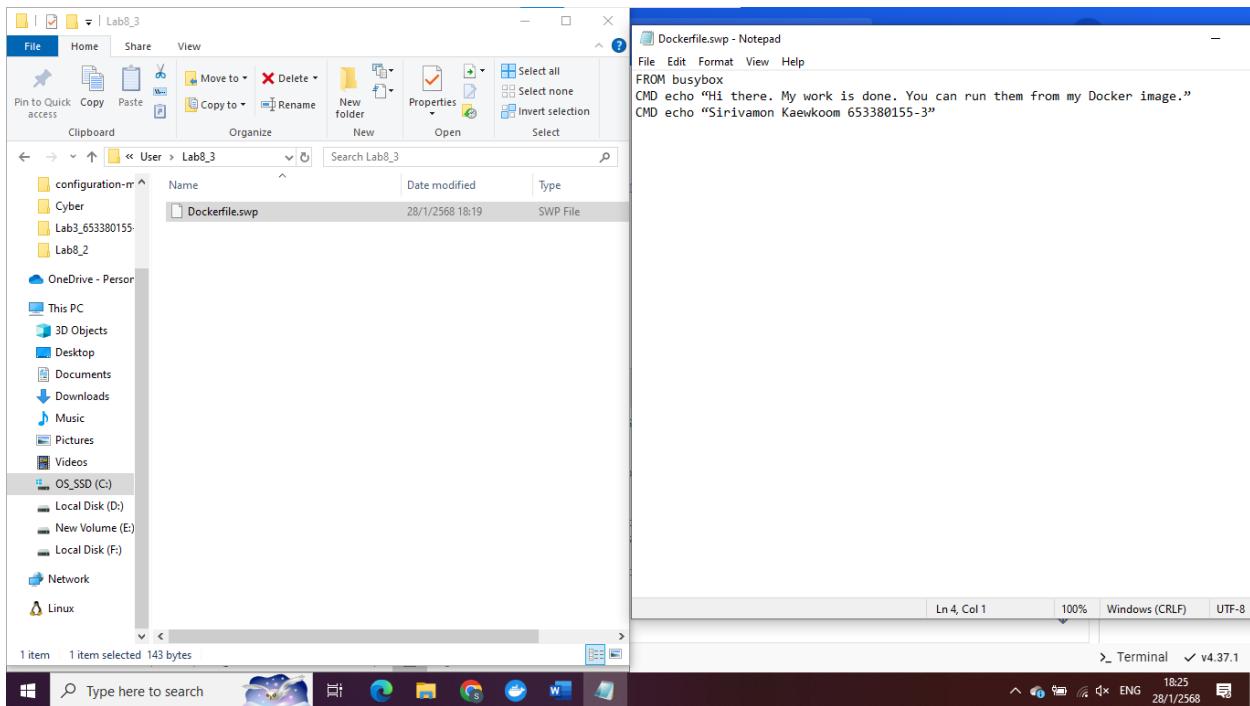
```
$ docker run <username> ที่ลงทะเบียนกับ Docker Hub>/lab8
```

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

## Lab Worksheet



## 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 ก็ได้

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

## Lab Worksheet

The screenshot shows the Docker Hub website at [hub.docker.com](https://hub.docker.com/). The user is logged in under the name 'sirivamon'. A repository named 'sirivamon/lab8' is listed, pushed 1 minute ago, and is a public image. To the right, there are links to 'Create an organization' and 'Create and manage users and grant access to your repositories.'

The screenshot shows the Docker Desktop application running on Windows. The 'Images' tab is selected in the sidebar. The terminal window shows the command to push the Docker image to the Docker Hub:

```

the last one will be used (line 2)
- JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 3)
PS C:\Users\User\Lab8_3> docker run sirivamon/lab8
"Sirivamon KaeWkood 653380155-3"
PS C:\Users\User\Lab8_3> docker push sirivamon/lab8
Using default tag: latest
The push refers to repository [docker.io/sirivamon/lab8]
5965ab79dad: Mounted from library/busybox
latest: digest: sha256:969b46aa84b4fd9da317159df0ebec8bf6033ca3539f88240b551fdb5255fdc4 size: 527
PS C:\Users\User\Lab8_3>

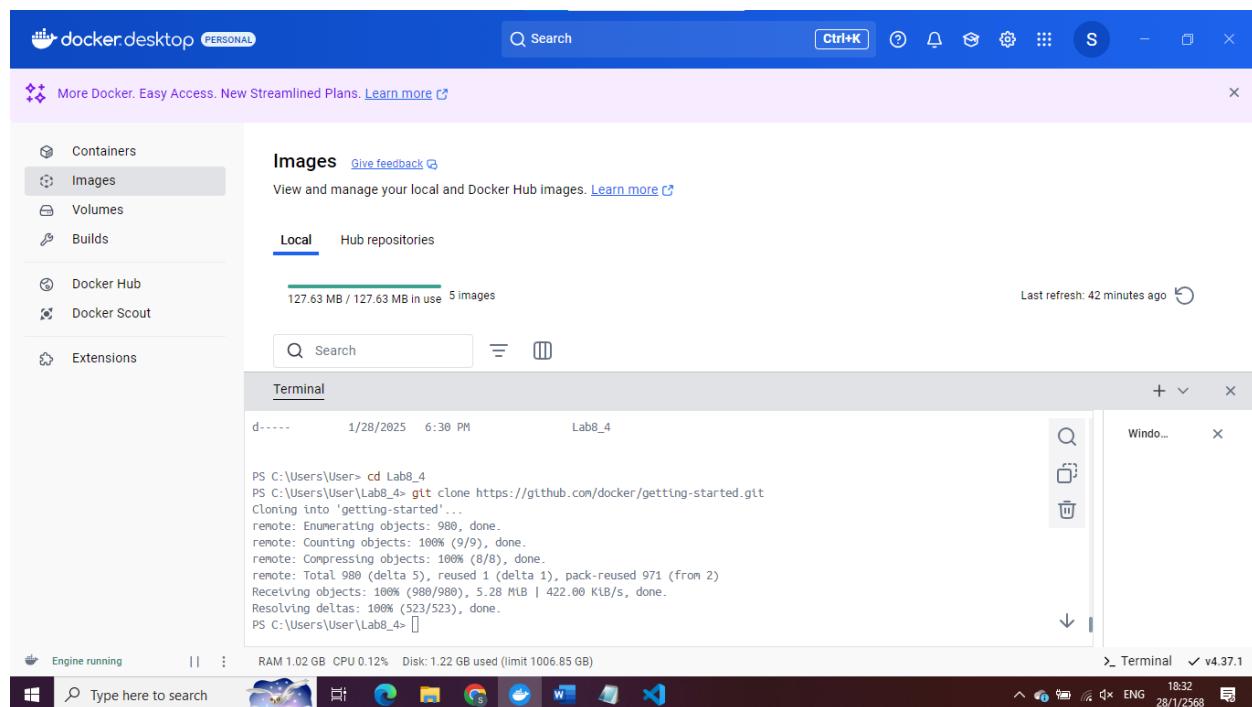
```

## Lab Worksheet

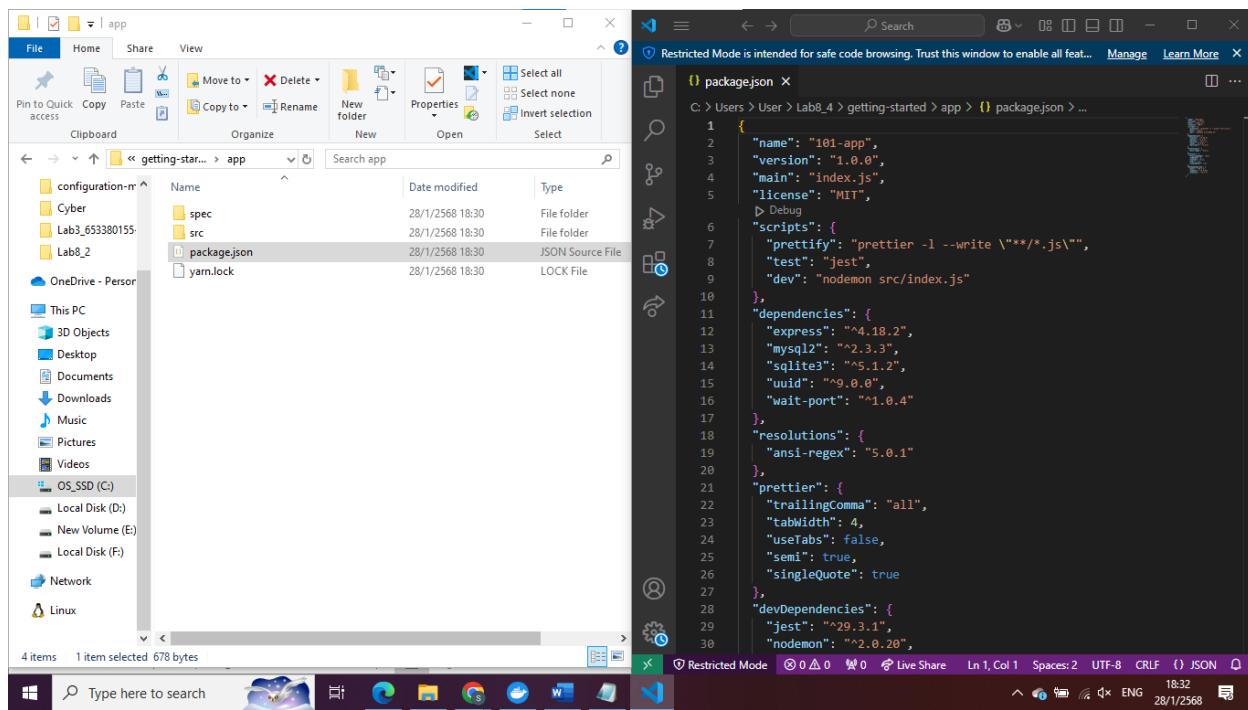
## แบบฝึกปฏิบัติที่ 8.4: การ Build แอปพลิเคชันจาก Container image และการ Update แอปพลิเคชัน

1. เปิด Command line หรือ Terminal จากนั้นสร้าง Directory ชื่อ Lab8\_4
2. ทำการ Clone ซอฟต์แวร์ของเว็บแอปพลิเคชันจาก 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



4. ภายใต้ getting-started/app ให้สร้าง Dockerfile พร้อมกับใส่เนื้อหาดังต่อไปนี้ลงไว้ในไฟล์
- ```

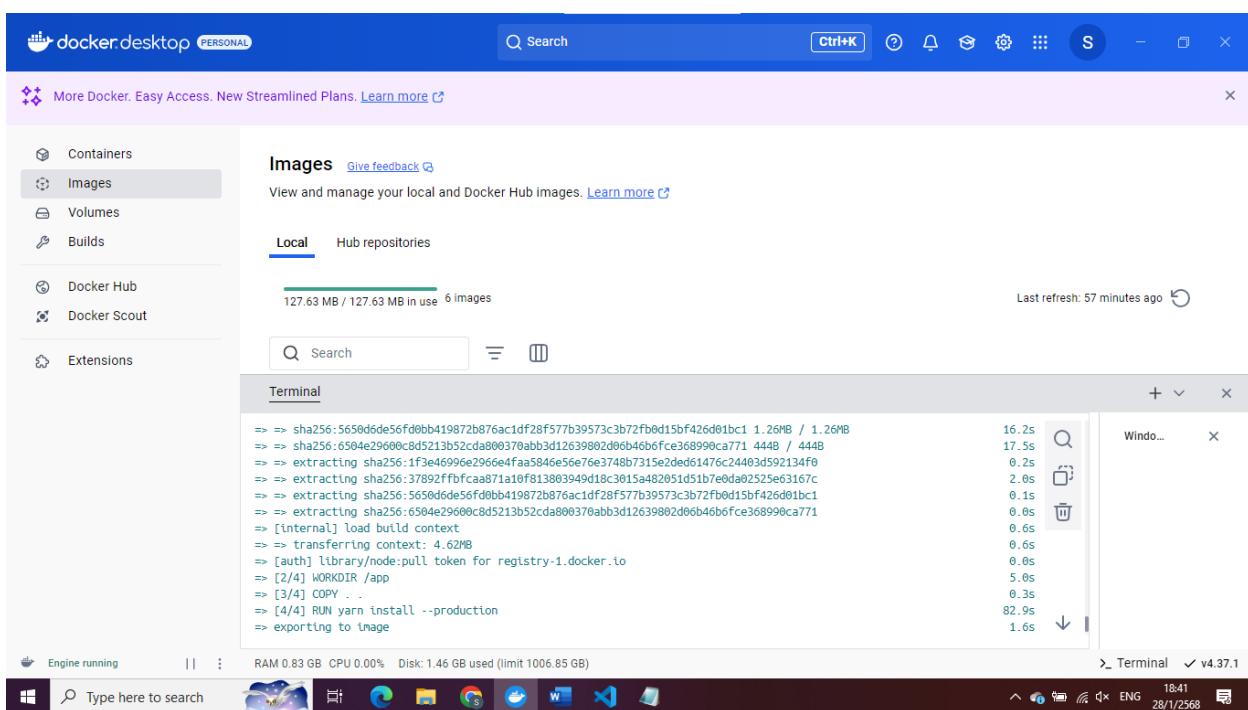
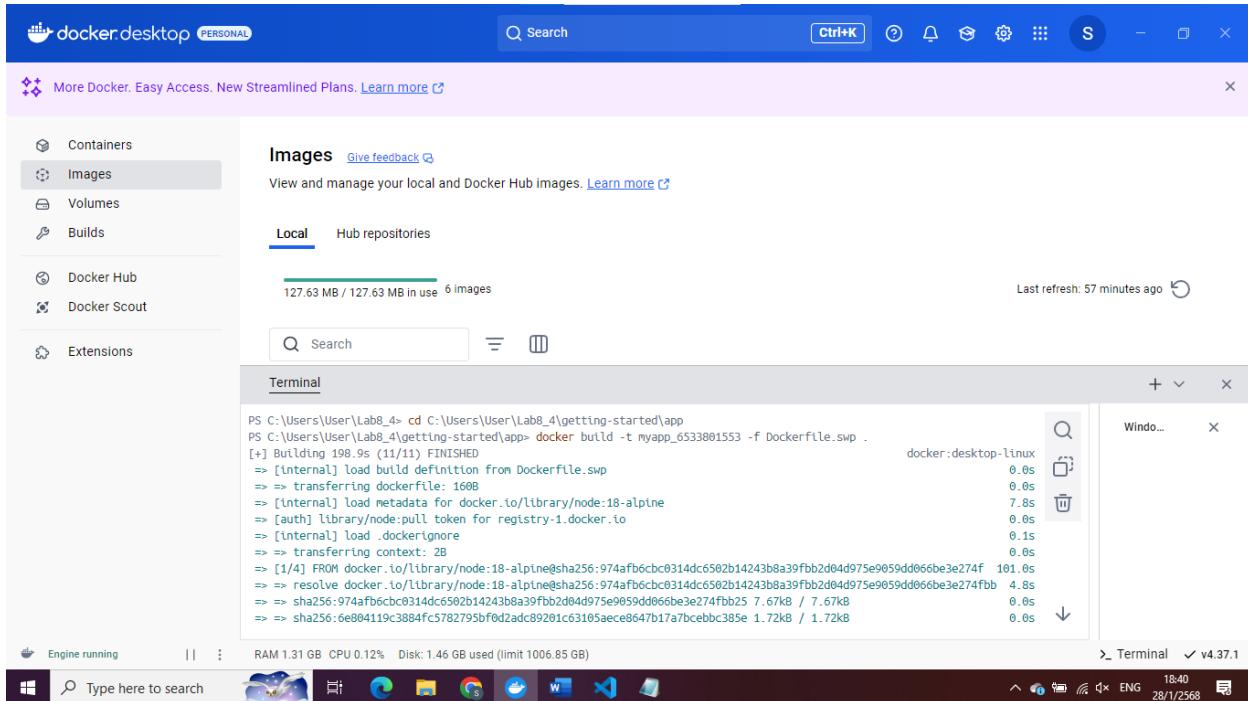
FROM node:18-alpine
WORKDIR /app
COPY .
RUN yarn install --production
CMD ["node", "src/index.js"]
EXPOSE 3000

```

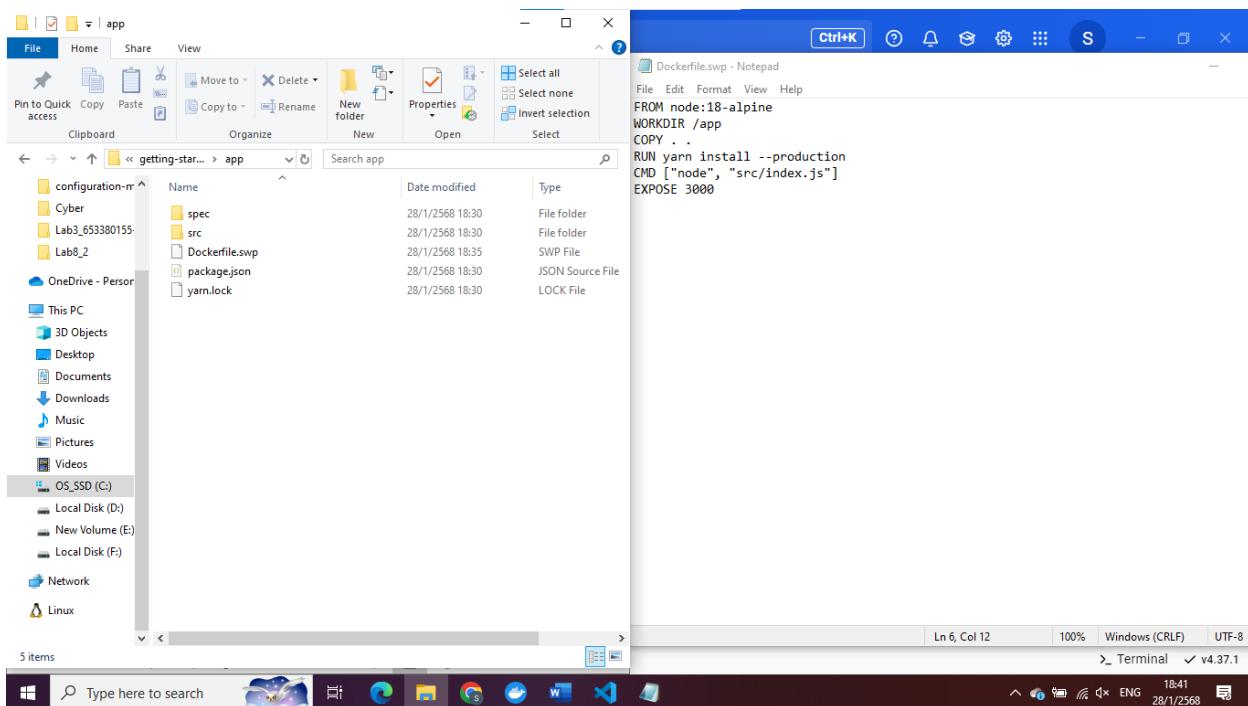
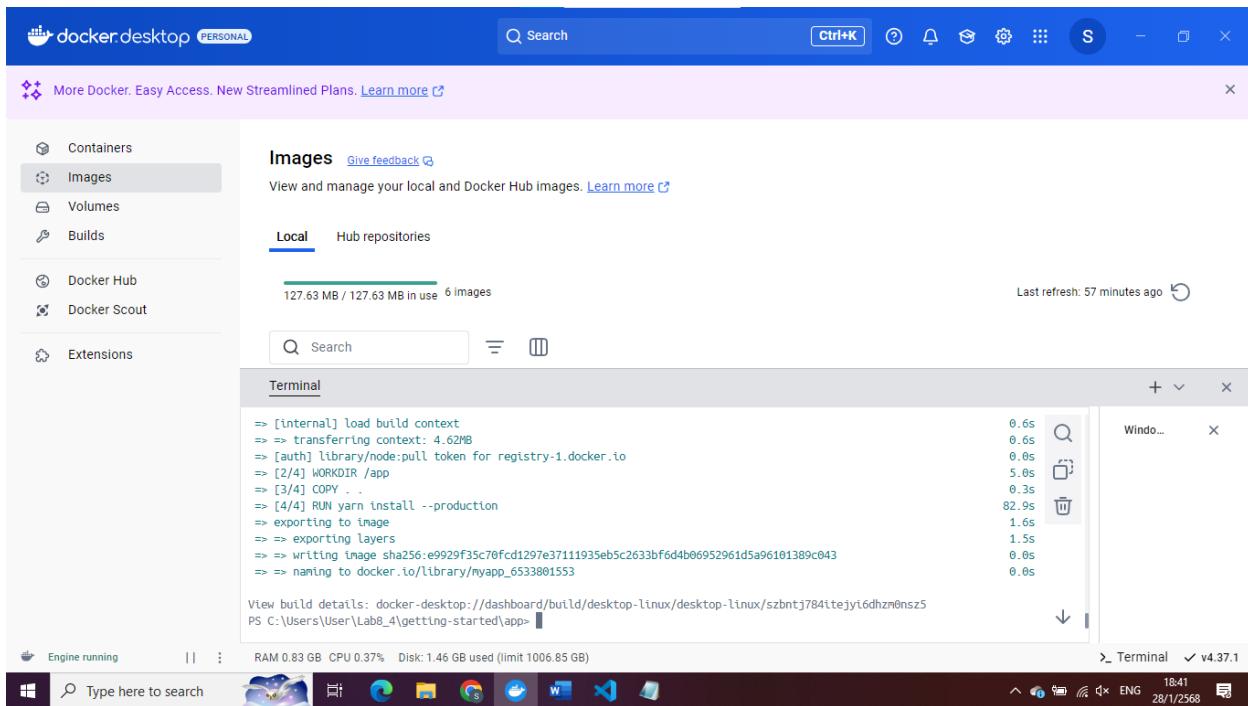
5. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้ โดยกำหนดใช้ชื่อ image เป็น myapp\_รหัสคน. ไม่มีจีด  
\$ docker build -t <myapp\_รหัสคน. ไม่มีจีด> .

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

## Lab Worksheet



## Lab Worksheet



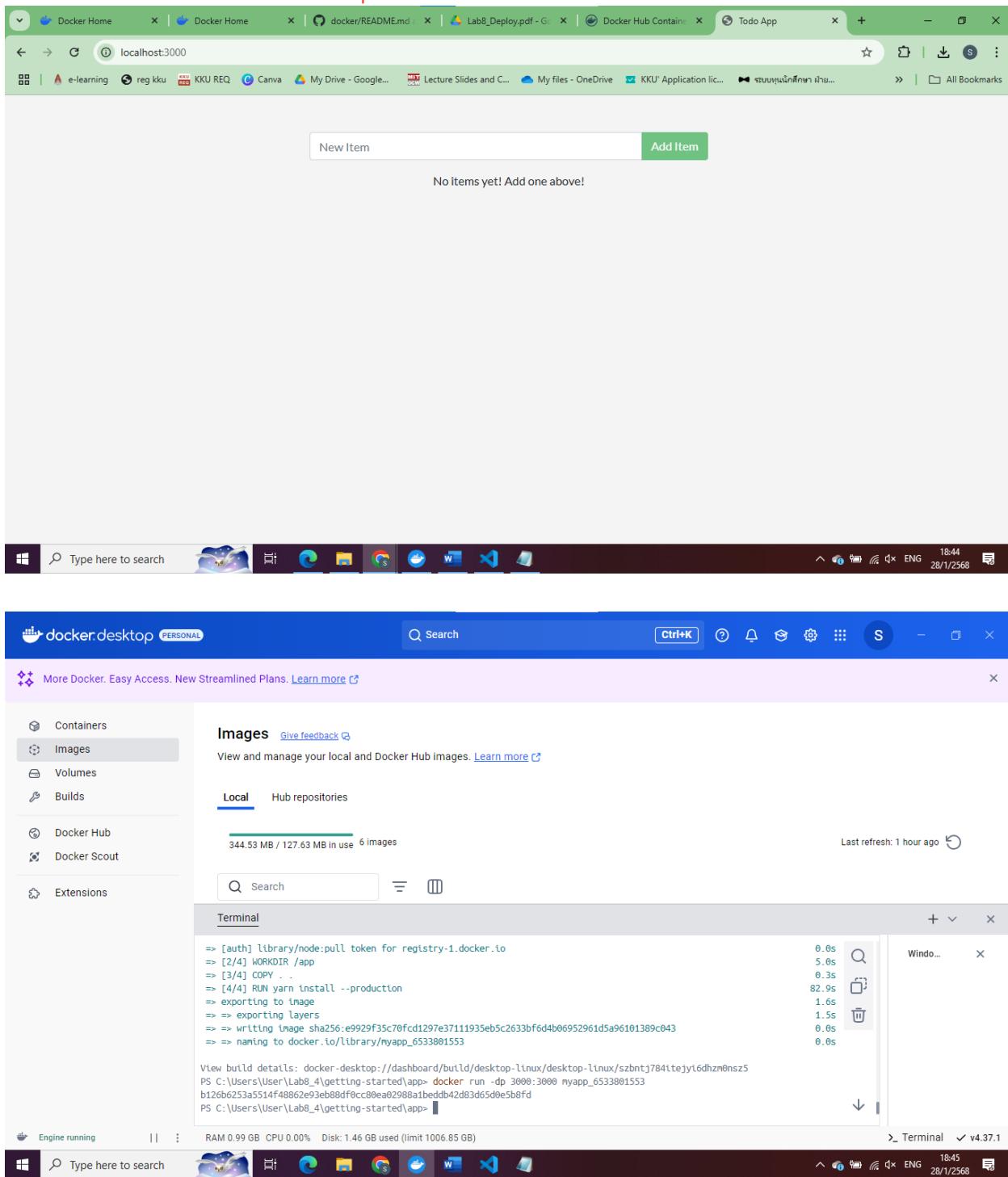
6. ทำการ Start ตัว Container ของแอปพลิเคชันที่สร้างขึ้น โดยใช้คำสั่ง

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

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

## Lab Worksheet

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



## Lab Worksheet

หมายเหตุ: นศ.สามารถทดลองเล่น 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

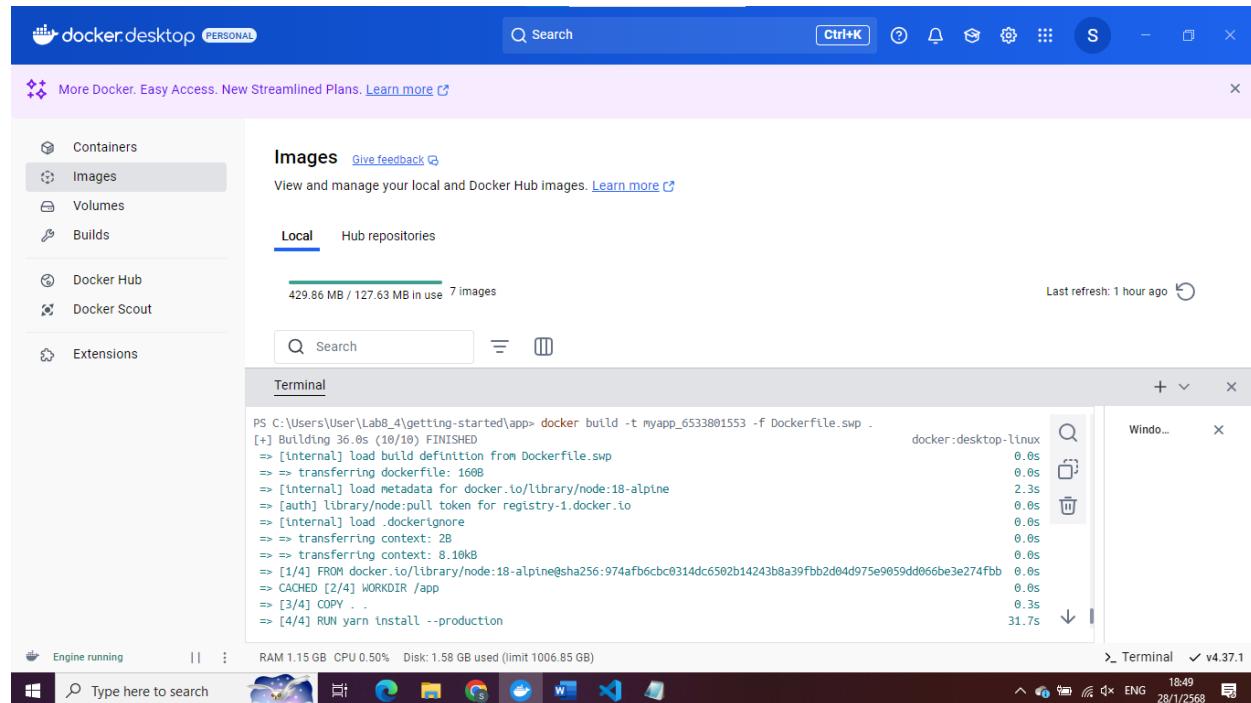
- b. Save ไฟล์ให้เรียบร้อย

9. ทำการ Build Docker image โดยใช้คำสั่งเดียวกันกับข้อ 5

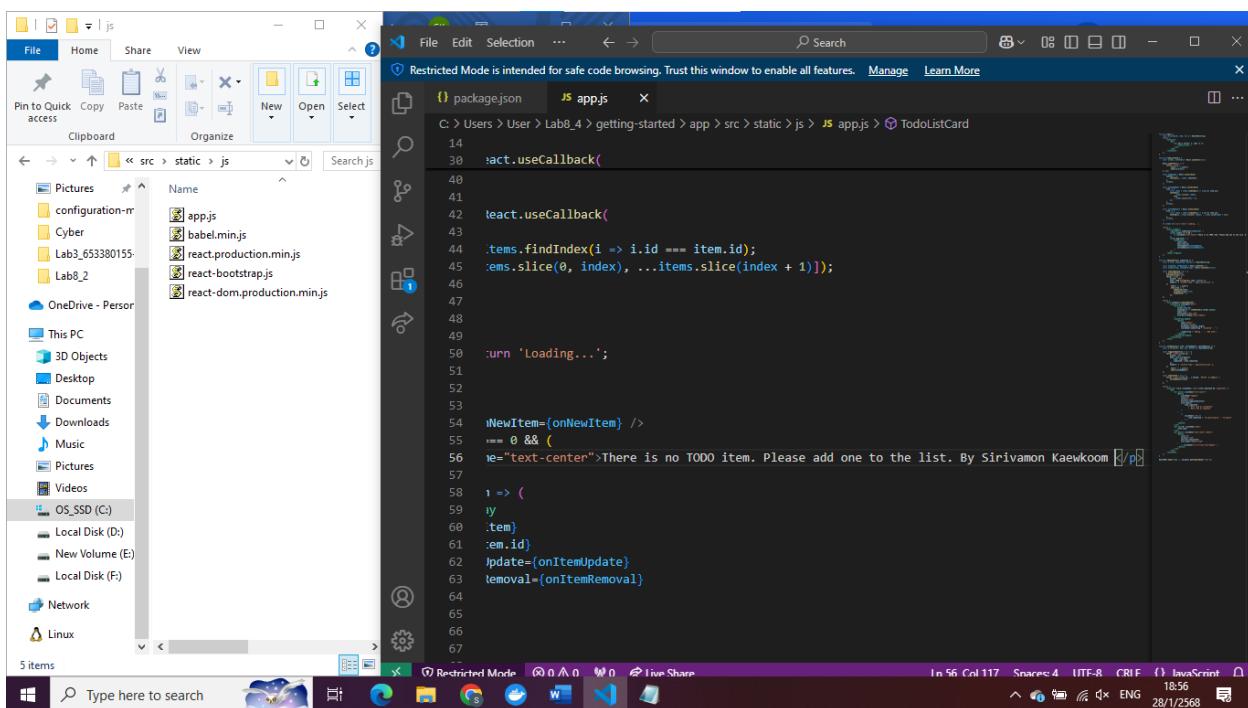
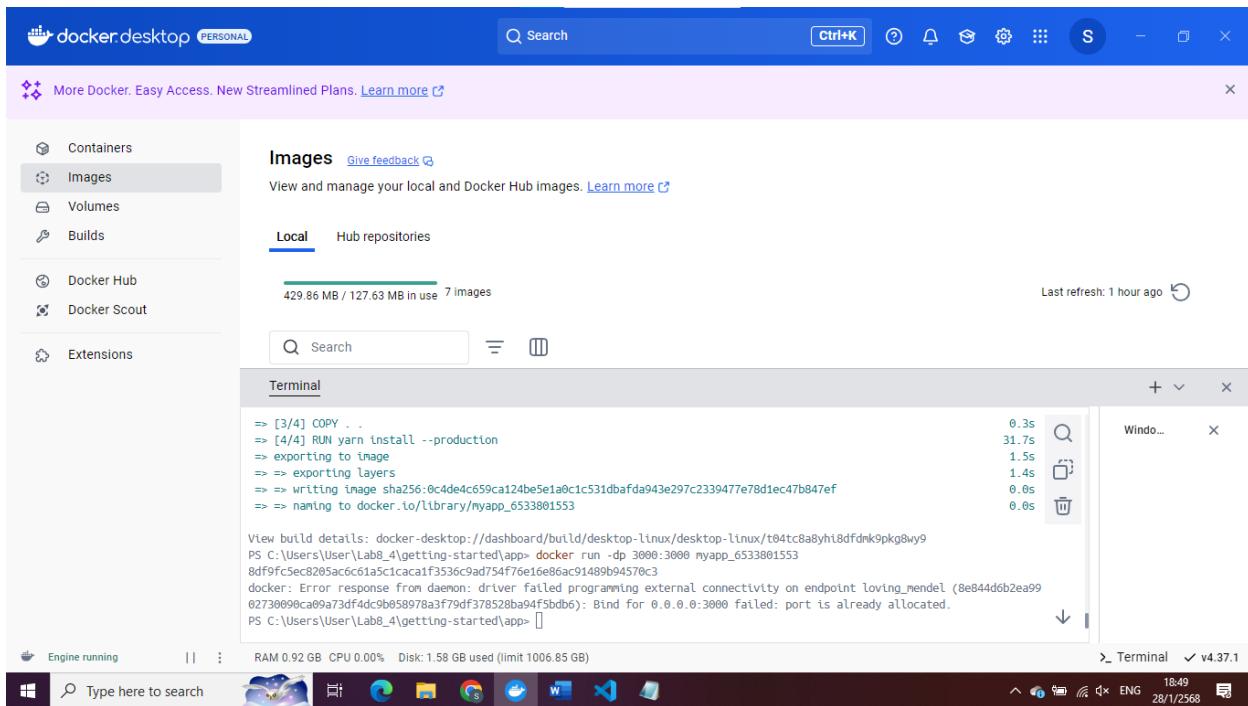
10. Start และรัน Container ตัวใหม่ โดยใช้คำสั่งเดียวกันกับข้อ 6

[Check point#10] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง)

แสดงคำสั่งและผลลัพธ์ที่ได้ทางหน้าจอ พร้อมกับตอบคำถามต่อไปนี้



## Lab Worksheet

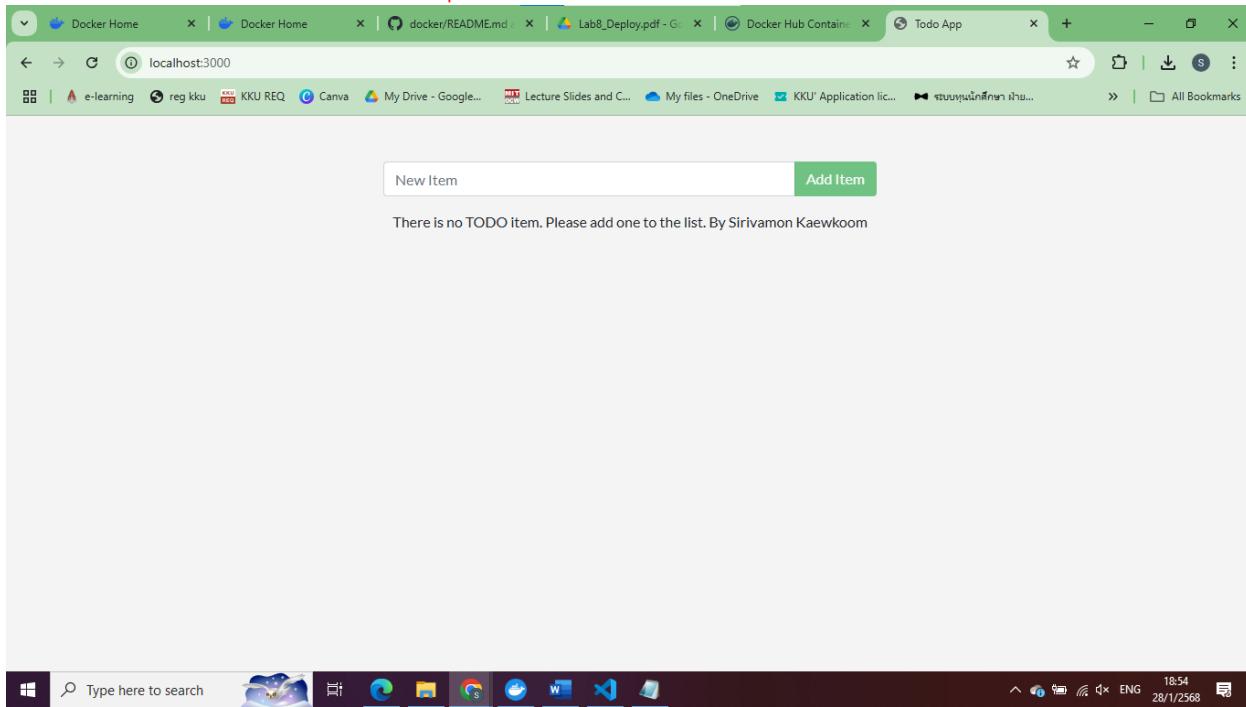


- (1) Error ที่เกิดขึ้นหมายความอย่างไร และเกิดขึ้นเพราะอะไร  
Port container ก่อนหน้านี้ยังคงทำงานอยู่
11. ลบ Container ของ Web application เวอร์ชันก่อนแก้ไขออกจากระบบ โดยใช้วิธีใดวิธีหนึ่งดังต่อไปนี้

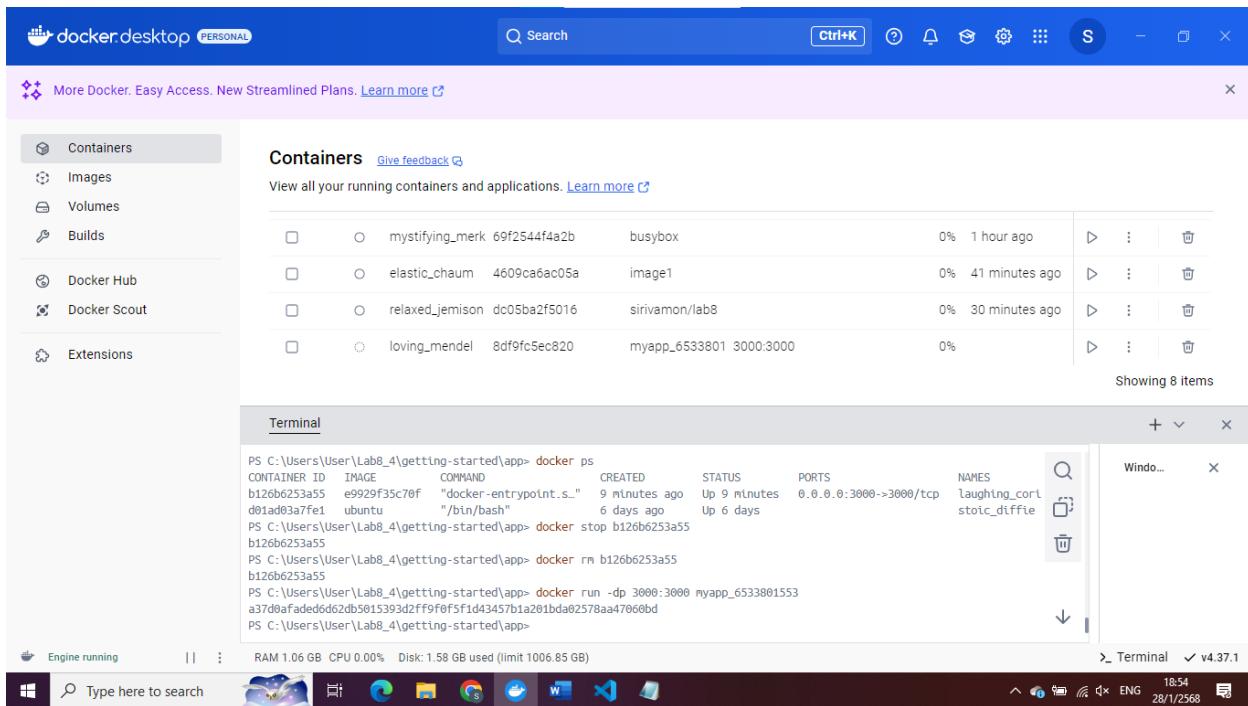
## Lab Worksheet

- 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
  - ผ่าน Docker desktop
    - ไปที่หน้าต่าง Containers
    - เลือกไอคอนถังขยะในແຄວของ Container ที่ต้องการจะลบ
    - ยืนยันโดยการกด Delete forever
12. Start และรัน Container ตัวใหม่อีกรัง โดยใช้คำสั่งเดียวกันกับข้อ 6
13. เปิด Browser ไปที่ URL = <http://localhost:3000>

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



## Lab Worksheet



### แบบฝึกปฏิบัติที่ 8.5: เริ่มต้นสร้าง Pipeline อย่างง่ายสำหรับการ Deploy ด้วย Jenkins

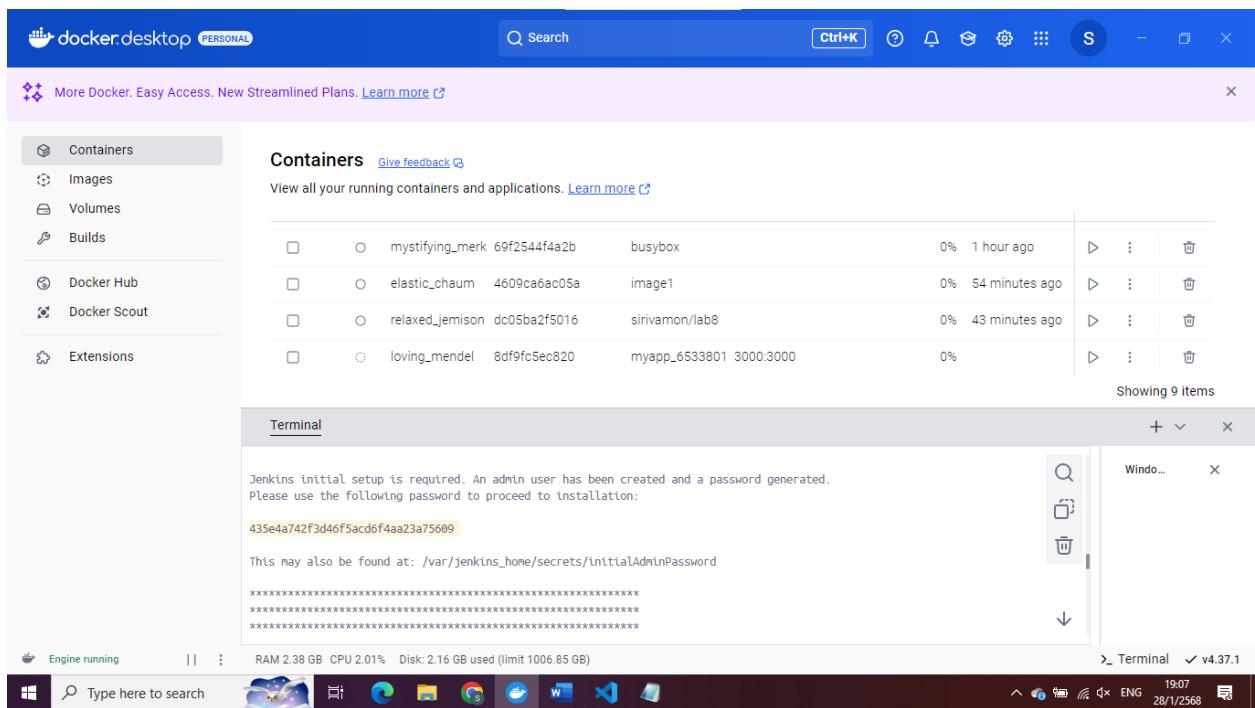
1. เปิด Command line หรือ Terminal บน Docker Desktop
2. ป้อนคำสั่งและทำการรัน container โดยผูกพอร์ต
 

```
$ 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**

## 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**

## Lab Worksheet

Getting Started

## Create First Admin User

Username

Password

Confirm password

Jenkins 2.479.3

Skip and continue as admin Save and Continue

Getting Started

Password

Confirm password

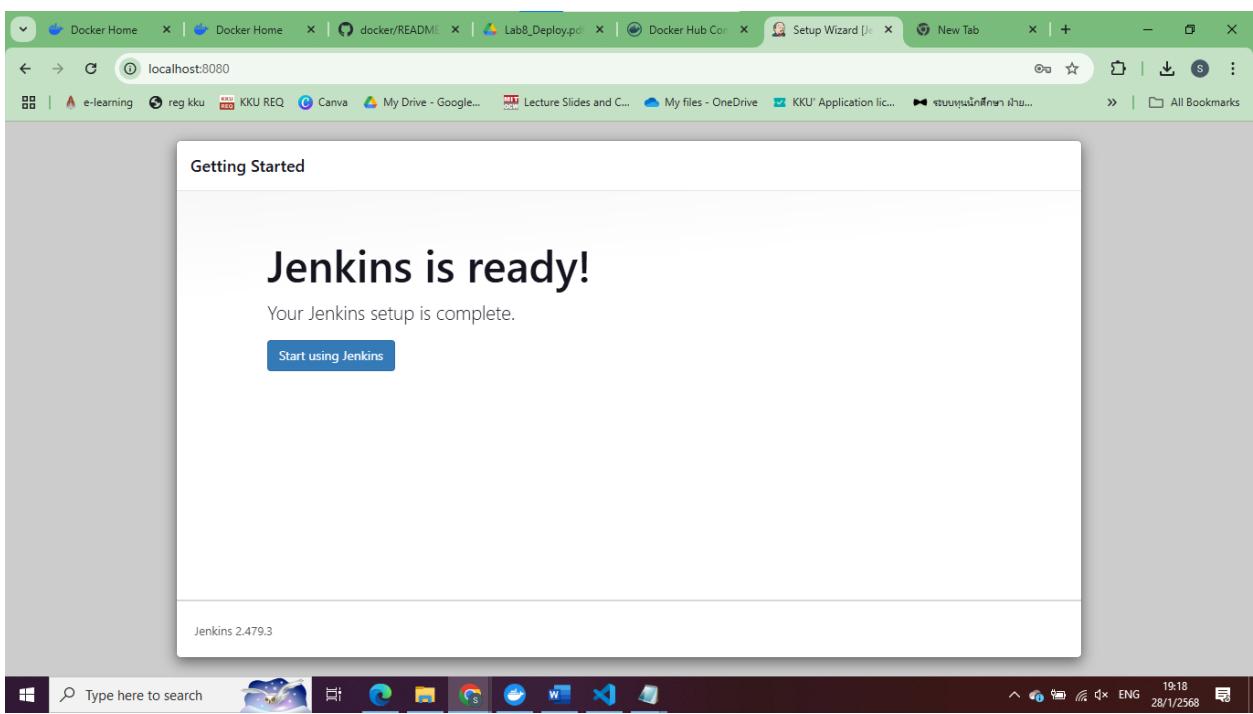
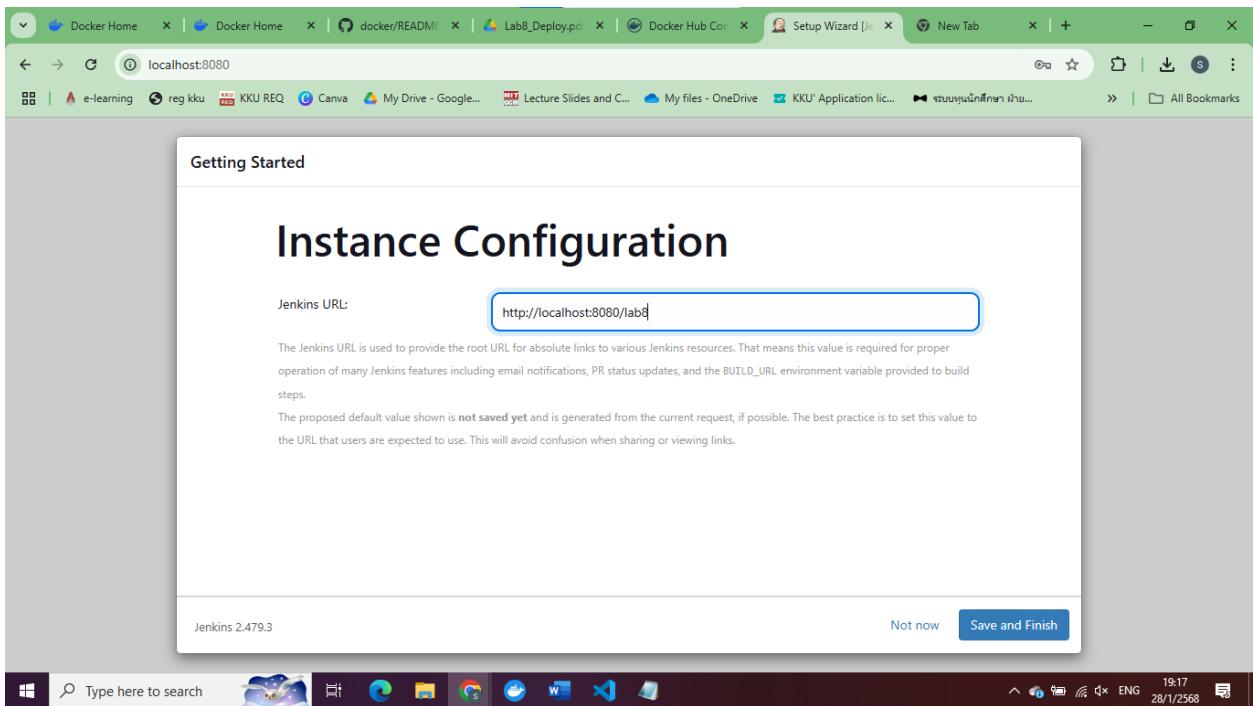
Full name

E-mail address

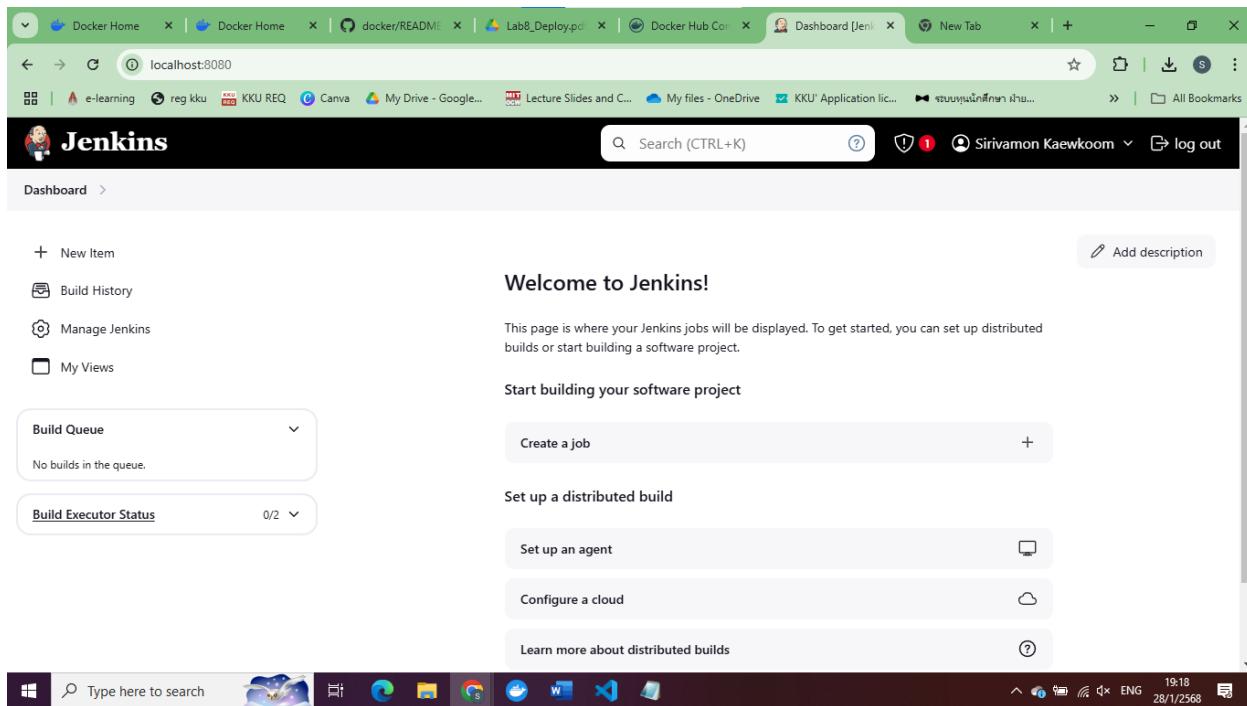
Jenkins 2.479.3

Skip and continue as admin Save and Continue

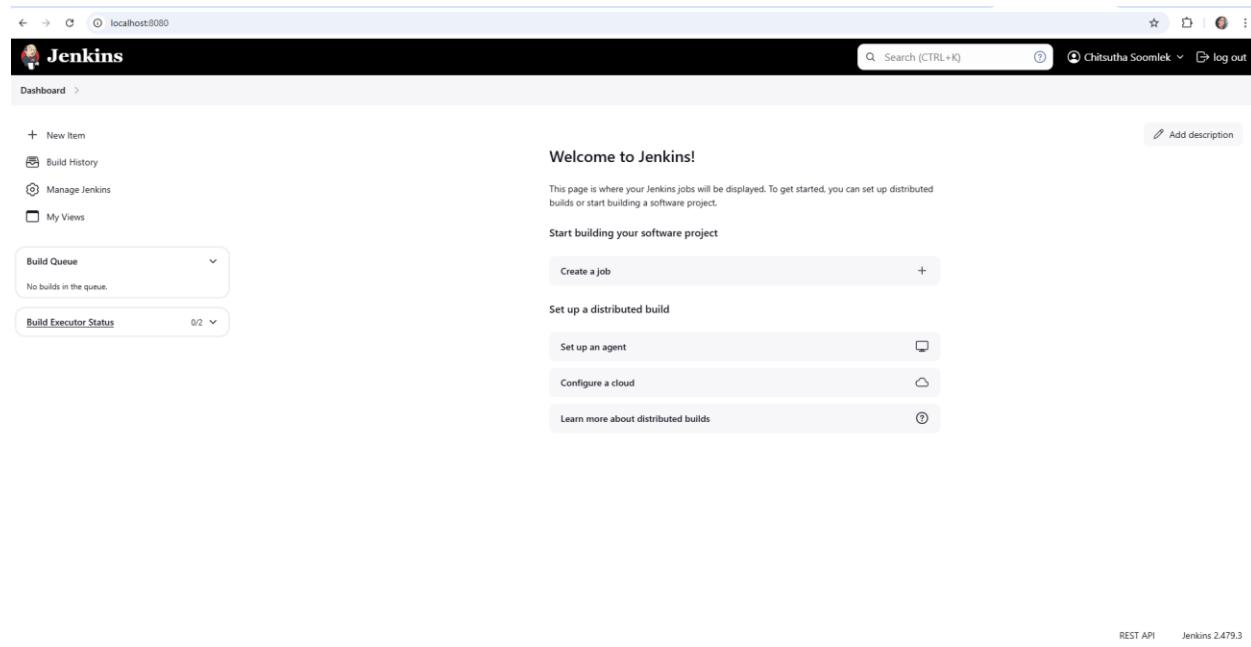
## Lab Worksheet



## Lab Worksheet



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



9. เลือก Manage Jenkins และไปที่เมนู Plugins

## Lab Worksheet

The screenshot shows the Jenkins Manage Jenkins interface. At the top, there's a red banner with the message "It appears that your reverse proxy set up is broken." Below this, there are several configuration sections:

- System Configuration:** Includes links for Build Queue, System, Tools, Plugins, Nodes, and Clouds.
- Security:** Includes links for Security, Credentials, Credential Providers, and Users.
- Status Information:** Includes links for System Information, System Log, Load Statistics, and About Jenkins.

10. ไปที่เมนู Available plugins แล้วเลือกติดตั้ง Robotframework เพิ่มเติม

The screenshot shows the Jenkins Plugin Manager. The left sidebar has links for Updates, Available plugins (which is selected), Installed plugins, Advanced settings, and Download progress. The main area shows a table of available plugins:

Install	Name	Released
	Robot Framework 5.0.0	2 mo 7 days ago
	Build Reports	

A note below the table states: "This publisher stores Robot Framework test reports for builds and shows summaries of them in project and build views along with trend graph."

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

## Lab Worksheet

New Item

Enter an item name

Select an item type

- Freestyle project**  
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.
- Pipeline**  
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
- Multi-configuration project**  
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.
- Folder**  
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.
- Multibranch Pipeline**  
Creates a set of Pipeline projects according to detected branches in one SCM repository.
- Organization Folder**  
Creates a set of multibranch project subfolders by scanning for repositories.

OK

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

Description: Lab 8.5

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

Build Trigger: เลือกแบบ Build periodically แล้วกำหนดให้ build ทุก 15 นาที

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

[Check point#14] Capture หน้าจอแสดงการตั้งค่า พร้อมกับตอบคำถามต่อไปนี้

## Lab Worksheet

The screenshot shows the Jenkins configuration interface for a job named "UAT". The "General" tab is selected. The "Description" field contains "Lab8.5". The "GitHub project" checkbox is checked, and the "Project url" field is set to "https://github.com/Sirivamon/Lab-7.git". The "Enabled" switch is turned on. The left sidebar lists "Source Code Management", "Build Triggers", "Build Environment", "Build Steps", and "Post-build Actions". The bottom navigation bar includes a search bar and system status indicators.

The screenshot shows the Jenkins configuration interface for a job named "UAT". The "Source Code Management" tab is selected. Under "Git", the "Repository URL" is set to "https://github.com/Sirivamon/Lab-7.git". The "Credentials" dropdown is currently empty. The "Save" and "Apply" buttons are visible at the bottom. The left sidebar lists "General", "Source Code Management", "Build Triggers", "Build Environment", "Build Steps", and "Post-build Actions". The bottom navigation bar includes a search bar and system status indicators.

## Lab Worksheet

The screenshot shows the Jenkins configuration interface for a job named "UAT Config". The left sidebar has "Source Code Management" selected. Under "Branches to build", the "Branch Specifier" field contains "\*/\*main". Below it, the "Repository browser" dropdown is set to "(Auto)". At the bottom are "Save" and "Apply" buttons.

The screenshot shows the Jenkins configuration interface for a job named "UAT Config". The left sidebar has "Build Triggers" selected. Under "Build Triggers", "Build periodically" is checked, with a schedule of "H/15 \* \* \* \*". A note below says "Would last have run at Tuesday, January 28, 2025 at 4:05:44 PM Coordinated Universal Time; would next run at Tuesday, January 28, 2025 at 4:20:44 PM Coordinated Universal Time." Other trigger options like "Trigger builds remotely" and "GitHub hook trigger for GITScm polling" are unchecked. At the bottom are "Save" and "Apply" buttons.

## Lab Worksheet

The screenshot shows the Jenkins UAT Configuration page under the 'Build Environment' tab. On the left sidebar, 'Build Environment' is selected. In the main area, there is a 'Build Steps' section containing a single 'Execute shell' step. The command entered is:

```
robot UAT-Lab7-001.robot UAT-Lab7-002.robot
```

Below the command input field are 'Save' and 'Apply' buttons.

The screenshot shows the Jenkins UAT Configuration page under the 'Post-build Actions' tab. On the left sidebar, 'Post-build Actions' is selected. In the main area, there is a 'Publish Robot Framework test results' step. The 'Directory of Robot output' field is set to 'Path to directory containing robot xml and html files (relative to build workspace)'. Below it, 'Advanced' settings are expanded, showing 'Thresholds for build result' with two entries: 20.0% and 80.0%. At the bottom are 'Save' and 'Apply' buttons.

- (1) คำสั่งที่ใช้ในการ Execute ไฟล์ .robot ใน Build Steps คือ  
robot UAT-Lab7-001.robot UAT-Lab7-002.robot

## Lab Worksheet

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

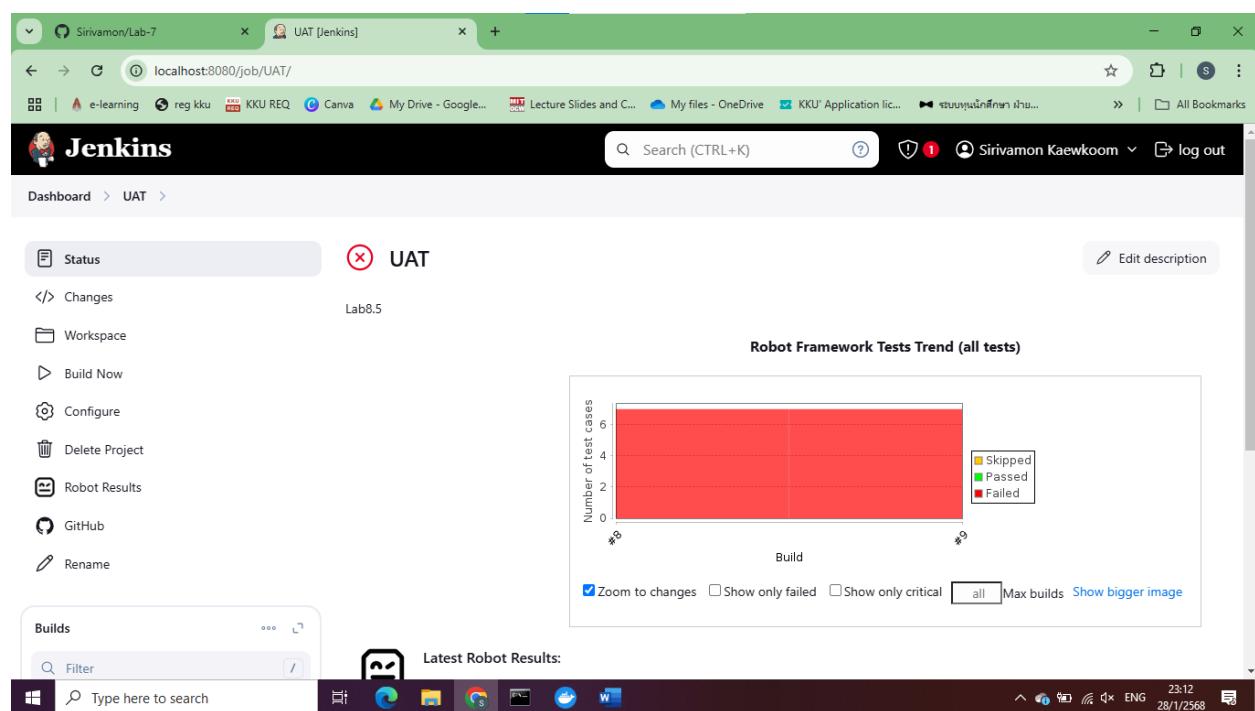
ระบุไดเร็คทอรีที่เก็บไฟล์ผลการทดสอบโดย Robot framework ในรูป xml และ html -> ตั้งค่า Threshold เป็น % ของการทดสอบที่ไม่ผ่านแล้วนับว่าซอฟต์แวร์มีปัญหา -> ตั้งค่า Threshold เป็น %

ของการทดสอบที่ผ่านแล้วนับว่าซอฟต์แวร์มีอยู่ในสถานะที่สามารถนำไปใช้งานได้ (เช่น 20, 80)

13. กด Apply และ Save

14. สร้าง Build Now

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



## Lab Worksheet

The screenshot shows the Jenkins UAT job dashboard. On the left, there's a sidebar titled "Builds" with a search bar and a list of builds from today, numbered #9 down to #1. Each build entry has a "Details" link. To the right, under "Build", there's a summary of "Latest Robot Results" showing 7 Failed, 0 Passed, 0 Skipped, and 0.0% Pass %. Below this are "Permalinks" for the last four builds.

**Build History:**

- #9 4:05 PM
- #8 4:04 PM
- #7 1:05 PM
- #6 12:55 PM
- #5 12:55 PM
- #4 12:55 PM
- #3 12:55 PM
- #2 12:55 PM
- #1 12:55 PM

**Latest Robot Results:**

Total	Failed	Passed	Skipped	Pass %
All tests	7	7	0	0.0

- [Browse results](#)
- [Open report.html](#)
- [Open log.html](#)

**Permalinks:**

- Last build (#9), 24 sec ago
- Last failed build (#9), 24 sec ago
- Last unsuccessful build (#9), 24 sec ago
- Last completed build (#9), 24 sec ago

REST API Jenkins 2.479.3

The screenshot shows the Jenkins UAT #9 Console output. On the left, there's a sidebar with links like Status, Changes, Console Output (which is selected), Edit Build Information, Delete build '#9', Timings, Git Build Data, Robot Results, and Previous Build. The main area is titled "Console Output" and shows the terminal logs for build #9. The logs include commands like git rev-parse, git config, and git fetch, along with commit details.

**Console Output:**

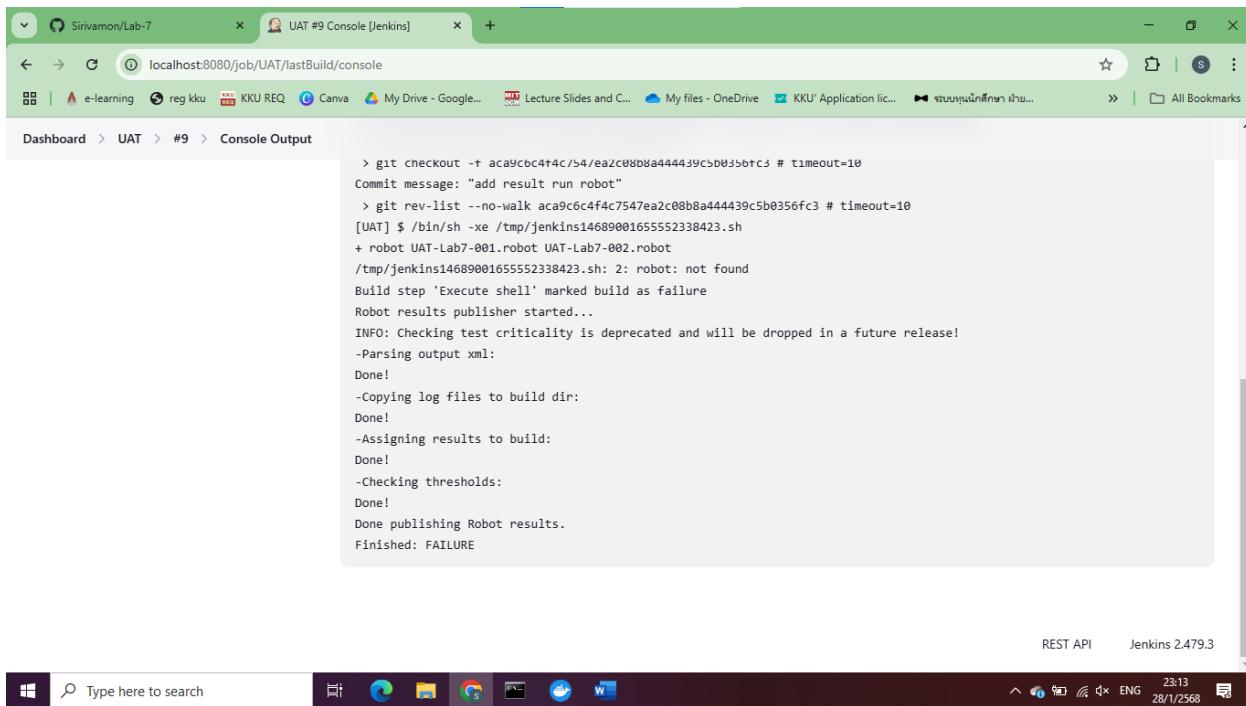
```

Started by timer
Running as SYSTEM
Building in workspace /var/jenkins_home/workspace/UAT
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /var/jenkins_home/workspace/UAT/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/Sirivamon/Lab-7.git # timeout=10
Fetching upstream changes from https://github.com/Sirivamon/Lab-7.git
> git --version # timeout=10
> git --version # 'git' version 2.39.5'
> git fetch --tags --force --progress -- https://github.com/Sirivamon/Lab-7.git +refs/heads/*:refs/remotes/origin/*
timeout=10
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision aca9c6c4f4c7547ea2c08b8a444439c5b0356fc3 (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f aca9c6c4f4c7547ea2c08b8a444439c5b0356fc3 # timeout=10

```

Search (CTRL+K) Sirivamon Kaewoom log out

## Lab Worksheet



The screenshot shows a Windows desktop environment. At the top, there is a browser window titled "UAT #9 Console [Jenkins]" displaying Jenkins console output for build #9. The output shows a series of git commands, a commit message, and a shell script execution. It includes messages about adding results, running robots, and publishing results. The Jenkins version is 2.479.3. At the bottom of the screen, the Windows taskbar is visible, featuring the Start button, a search bar, pinned application icons (File Explorer, Edge, File History, Task View, File Explorer, and a blue icon), and system status icons (Wi-Fi, battery, volume, and language). The system tray shows the date and time as 23:13, 28/1/2568.

```
> git checkout -t ac9c6c4f4c54/ea2c08b8a444439c5b0356fc3 # timeout=10
Commit message: "add result run robot"
> git rev-list --no-walk ac9c6c4f4c7547ea2c08b8a444439c5b0356fc3 # timeout=10
[UAT] $ /bin/sh -xe /tmp/jenkins14689001655552338423.sh
+ robot UAT-Lab7-001.robot UAT-Lab7-002.robot
/tmp/jenkins14689001655552338423.sh: 2: robot: 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:
Done!
-Copying log files to build dir:
Done!
-Assigning results to build:
Done!
-Checking thresholds:
Done!
Done publishing Robot results.
Finished: FAILURE
```