

## Lab Worksheet

ชื่อ-นามสกุล นาย คัจฉพัฒน์ ศรีนวล

รหัสนักศึกษา 643021219-4

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

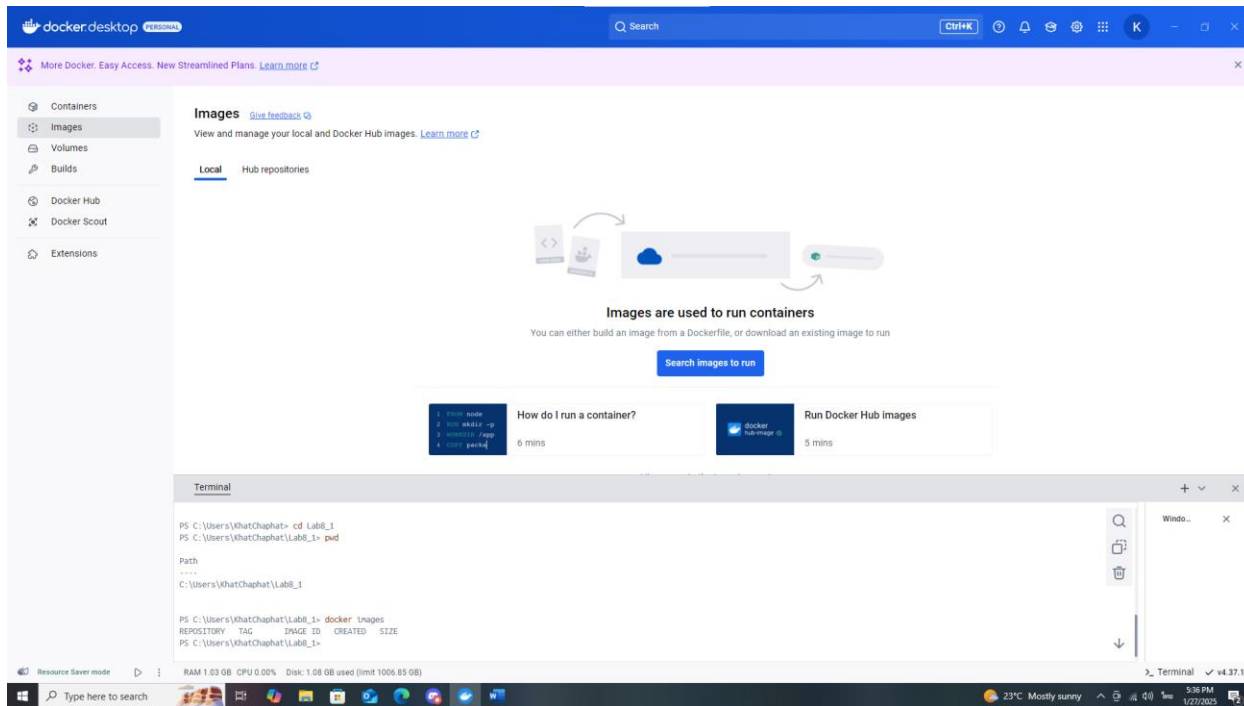
1. ติดตั้ง Docker desktop ลงบนเครื่องคอมพิวเตอร์ โดยดาวน์โหลดจาก <https://www.docker.com/get-started>
2. สร้าง Account บน Docker hub (<https://hub.docker.com/signup>)
3. กำหนดให้ \$ หมายถึง Command prompt และ <> หมายถึง ให้ป้อนค่าของพารามิเตอร์ที่กำหนด

## 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

## Lab Worksheet

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



(1) สิ่งที่อยู่ภายใต้คอลัมน์ Repository คืออะไร ชื่อของ Docker image repository

(2) Tag ที่ใช้บ่งบอกถึงอะไร เวอร์ชันหรือป้ายกำกับของ image (เช่น latest, stable)

5. ป้อนคำสั่ง \$ docker run busybox

6. ป้อนคำสั่ง \$ docker run -it busybox sh

7. ป้อนคำสั่ง ls

8. ป้อนคำสั่ง ls -la

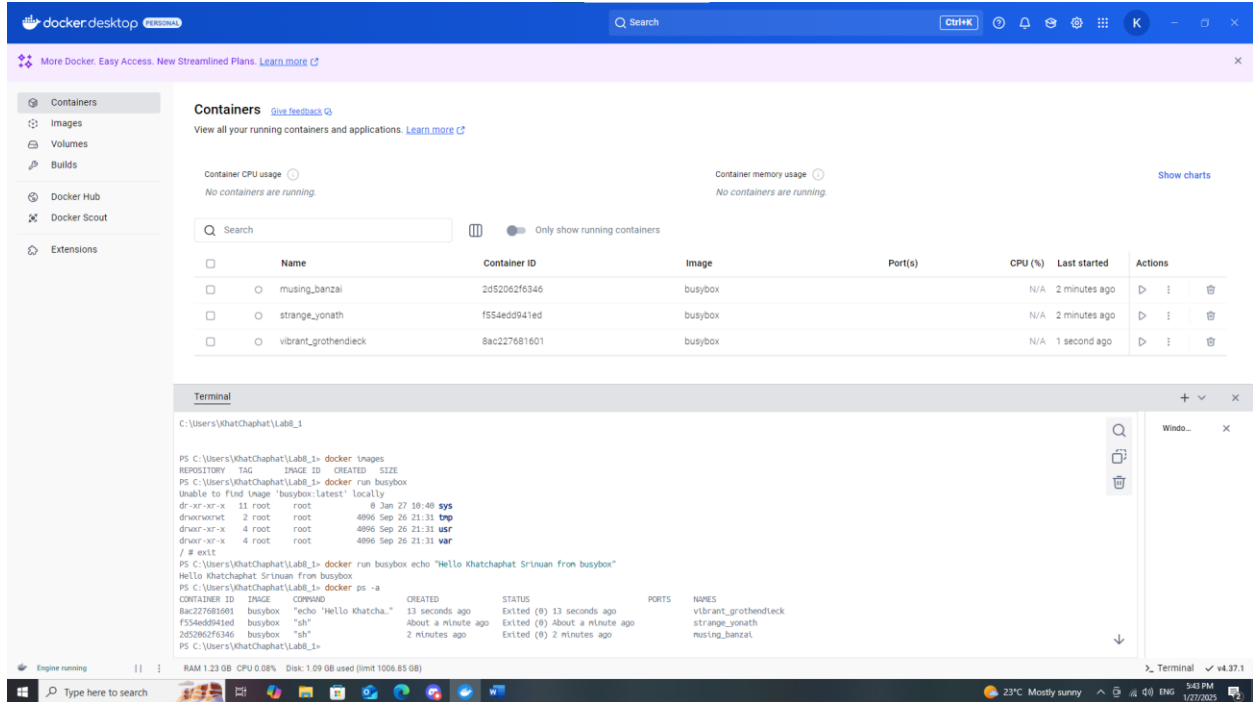
9. ป้อนคำสั่ง exit

10. ป้อนคำสั่ง \$ docker run busybox echo "Hello ชื่อและนามสกุลของนักศึกษา from busybox"

11. ป้อนคำสั่ง \$ docker ps -a

## Lab Worksheet

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



(1) เมื่อใช้ option -it ในคำสั่ง run ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป

Option -it มีความสำคัญเมื่อคุณต้องการทำงานกับ container แบบ interactive เช่น การเข้าไปใช้ shell เพื่อจัดการไฟล์หรือรันคำสั่งภายใน container โดยตรง

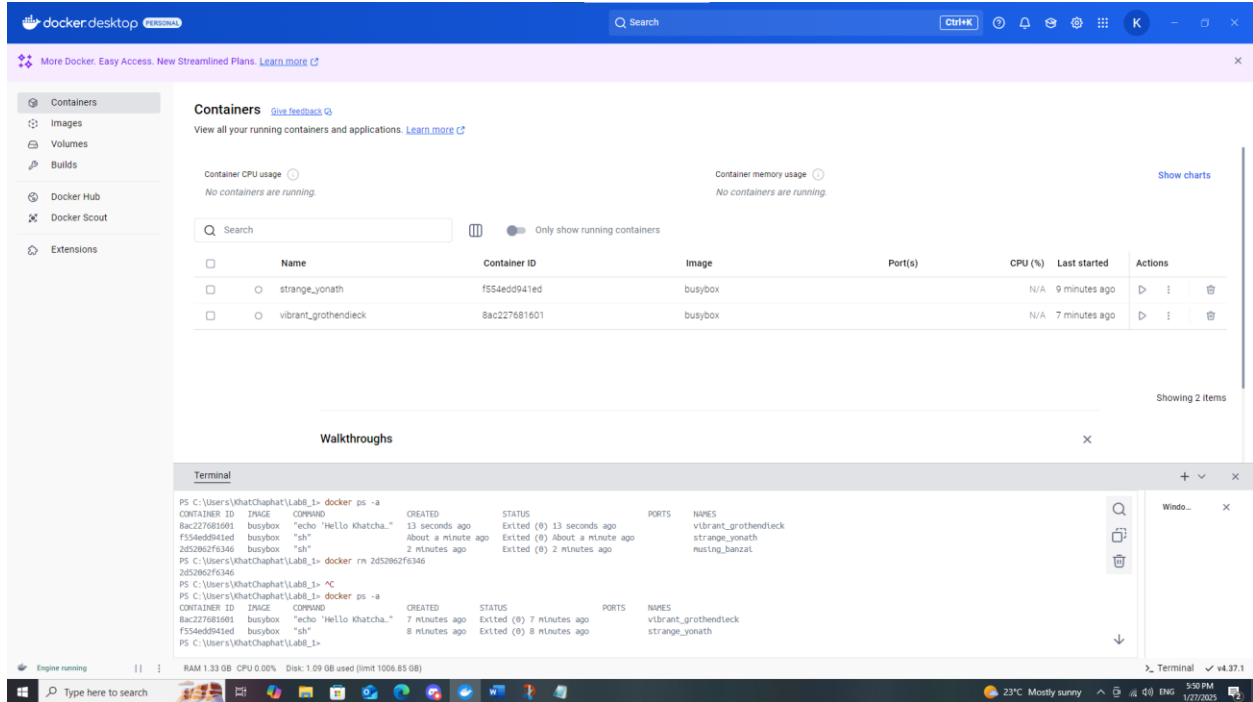
(2) คอลัมน์ STATUS จากการรันคำสั่ง docker ps -a แสดงถึงข้อมูลอะไร

แสดงข้อมูลเกี่ยวกับ สถานะของแต่ละ container ที่เคยถูกรัน

## Lab Worksheet

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

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



## 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"

## Lab Worksheet

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

The screenshot shows the Docker Desktop application. On the left, a sidebar contains navigation options: Containers, Images, Volumes, Builds, Docker Hub, Docker Scout, and Extensions. The main area is titled 'Containers' and shows 'No containers are running.' Below this, a terminal window is open, displaying the following commands and output:

```
File Edit Format View Help
FROM busybox
CMD echo "Hi there. This is my first docker image."
CMD echo "Khatchaphat Srinuan 6430212194 Kut."

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/e291pm8mnc5g7qgn7ayqf6
PS C:\Users\Khatchaphat\Lab8_2> docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
my-first-image latest 2c435d9d75c88c161b87e37285f4e1745abbee344882f9957429629dcf604 4 months ago 6.50MB
busybox latest a5d8ce48a88 4 months ago 6.50MB
PS C:\Users\Khatchaphat\Lab8_2> docker run my-first-image
Hi there. This is my first docker image.
Khatchaphat Srinuan 6430212194 Kut
PS C:\Users\Khatchaphat\Lab8_2>
```

On the right side of the Docker Desktop interface, there is a table showing the status of containers. The table has columns for Image, Port(s), CPU (%), Last started, and Actions. The table lists several instances of 'my-first-image' that have been started recently.

Image	Port(s)	CPU (%)	Last started	Actions
my-first-image		N/A	9 minutes ago	Stop, Restart, Delete
my-first-image		N/A	7 minutes ago	Stop, Restart, Delete
my-first-image		N/A	7 minutes ago	Stop, Restart, Delete
my-first-image		N/A	5 minutes ago	Stop, Restart, Delete
my-first-image		N/A	2 minutes ago	Stop, Restart, Delete
my-first-image		N/A	1 minute ago	Stop, Restart, Delete

At the bottom of the Docker Desktop window, a status bar shows system resources: RAM 1.23 GB, CPU 0.00%, Disk 1.09 GB used (limit 1006.85 GB). The bottom right corner shows the system clock: 6:22 PM, 1/27/2025.

## Lab Worksheet

(1) คำสั่งที่ใช้ในการ run คือ

`docker run my-first-image`

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

การใช้ -t ในคำสั่ง docker build จะช่วยให้สามารถตั้งชื่อและ tag ให้กับ Docker image ได้ ซึ่งช่วยให้การจัดการและอ้างอิง Docker image ง่ายขึ้นและเป็นระเบียบมากขึ้น

### 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`

`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

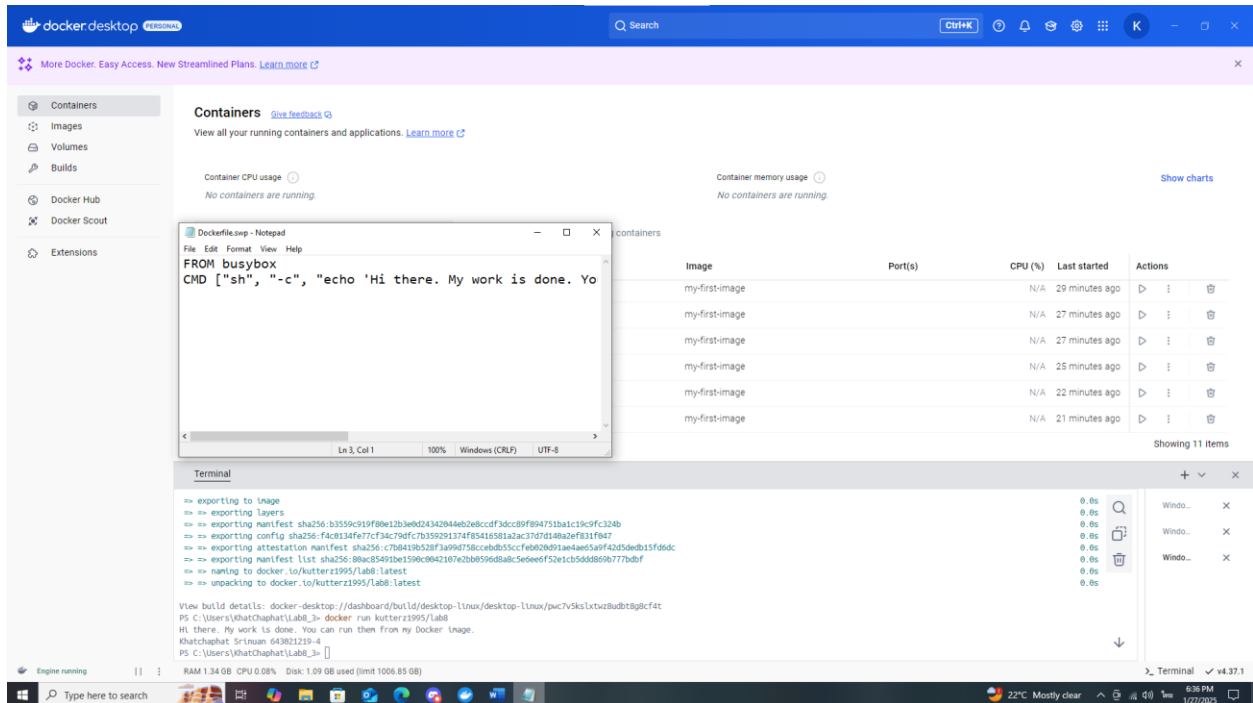
## Lab Worksheet

\$ 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



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

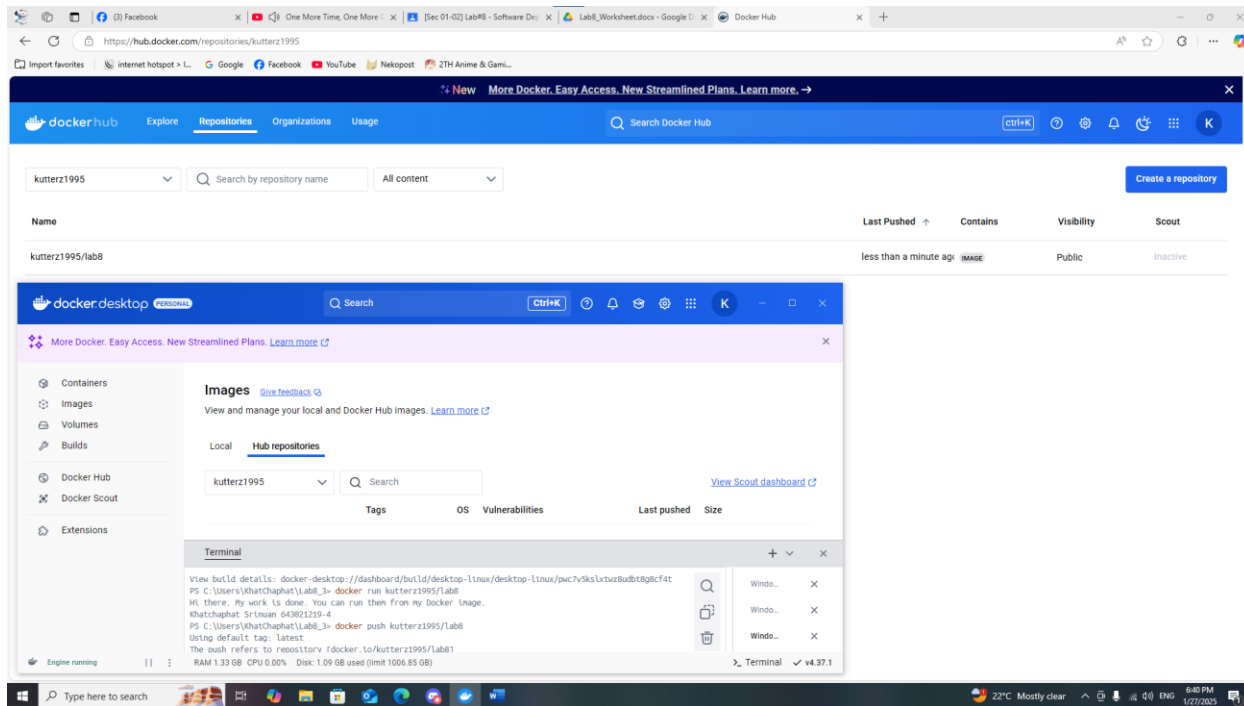
\$ docker login แล้วป้อน Username และ Password ตามที่ระบุใน Command prompt หรือใช้คำสั่ง

\$ 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)



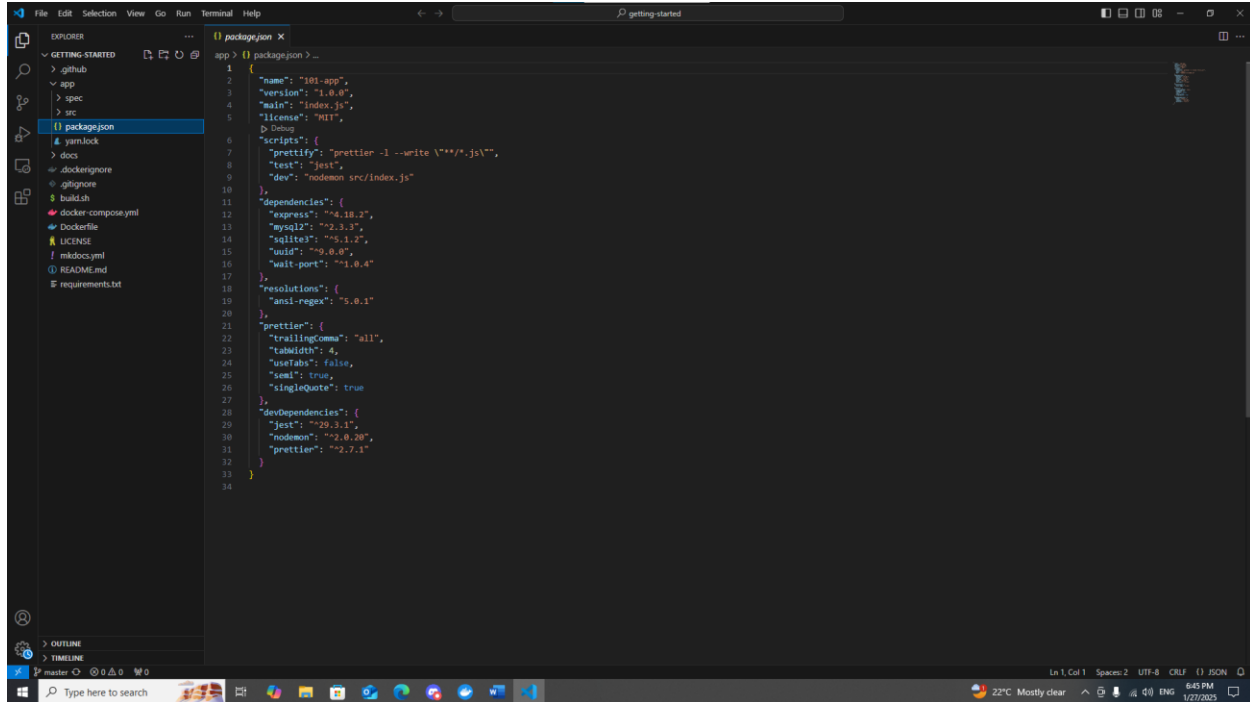
### แบบฝึกปฏิบัติที่ 8.4: การ Build แอปพลิเคชันจาก Container image และการ Update แอปพลิเคชัน

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



## Lab Worksheet

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



4. ภายใต้ getting-started/app ให้สร้าง Dockerfile พร้อมกับใส่เนื้อหาดังต่อไปนี้ลงไปไฟล์

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\_รหัสศ. ไม่มีขีด

\$ docker build -t <myapp\_รหัสศ. ไม่มีขีด> .

[Check point#8] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง)

แสดงคำสั่งและผลลัพธ์ที่ได้ทางหน้าจอ

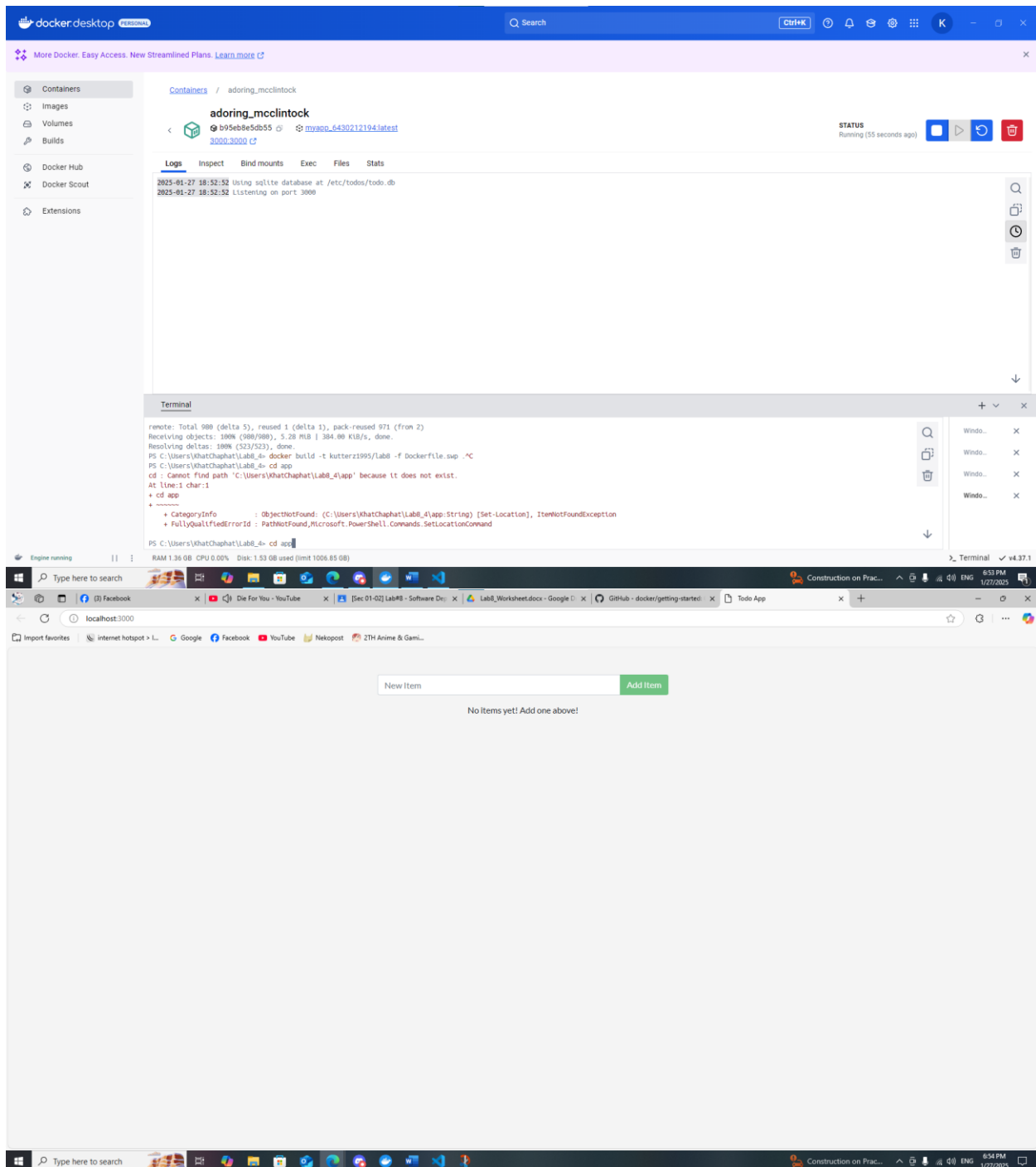
6. ทำการ Start ตัว Container ของแอปพลิเคชันที่สร้างขึ้น โดยใช้คำสั่ง

## Lab Worksheet

\$ docker run -dp 3000:3000 <myapp\_รหัสนศ. ไม่มีชื่อ>

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

[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</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

The screenshot shows a VS Code editor with a file explorer on the left displaying a project structure for a React application. The main editor window shows the file `src/static/js/app.js` with the following code:

```

42 function TodoListCard() {
43   const onItemRemoval = React.useCallback(
44     item => {
45       const index = items.findIndex(i => i.id === item.id);
46       setItems([...items.slice(0, index), ...items.slice(index + 1)]);
47     },
48     [items],
49   );
50   if (items === null) return 'loading...';
51   return (
52     <React.Fragment>
53       <AddItemForm onNewItem={onNewItem} />
54       {items.length === 0 && (
55         <p className="text-center">There is no TODO item. Please add one to the list. By Khatchaphat Srinuanic</p>
56       )}
57       {items.map(item => (
58         <ItemDisplay
59           item={item}
60           key={item.id}
61           onItemUpdate={onItemUpdate}
62           onItemRemoval={onItemRemoval}
63         />
64       ))}
65     </React.Fragment>
66   );
67 }
68
69 function AddItemForm({ onNewItem }) {
70   const [form, InputGroup, Button] = React.useState('');
71   const [newItem, setNewItem] = React.useState('');
72   const [submitting, setSubmitting] = React.useState(false);
73 }

```

The terminal window at the bottom shows the following commands and output:

```

PS C:\Users\Khatchaphat\Lab_4\getting-started> docker run -p 3000:3000 myapp_6430212194
10481101c624e342306c30862720551d4f8010d61a306c79f9f6b109
docker: Error response from daemon: driver failed programming external connectivity on endpoint sharp_chandrashukar (bf50e28701bcb139170929c40b5a6357fb81a3c6941480459881c124a0f7): Bind for 0.0.0.0:3000 failed: port is already allocated.
PS C:\Users\Khatchaphat\Lab_4\getting-started> docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED              STATUS      PORTS      NAMES
b99e8e5db55   faa0fe14fd  "docker-entrypoint.s..."  7 minutes ago       Up 7 minutes    0.0.0.0:3000->3000/tcp    adoring_mcclintock
PS C:\Users\Khatchaphat\Lab_4\getting-started> docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED              STATUS      PORTS      NAMES
c9bb05211599d6f295254c5747f4238ef7485433ea012ec70723c94254f  myapp_6430212194  7 minutes ago       Up 7 minutes    0.0.0.0:3000->3000/tcp    adoring_mcclintock
PS C:\Users\Khatchaphat\Lab_4\getting-started>

```

(1) Error ที่เกิดขึ้นหมายความว่าอย่างไร และเกิดขึ้นเพราะอะไร

Error ที่เกิดขึ้นหมายถึงไม่สามารถ Run Docker ใหม่ได้เนื่องจากของเก่าที่ Run อยู่ยังไม่หยุดทำงาน ต้องสั่งหยุดทำงาน Docker ของเก่าก่อนจึงถึงค่อยสั่ง Run ของใหม่

## Lab Worksheet

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

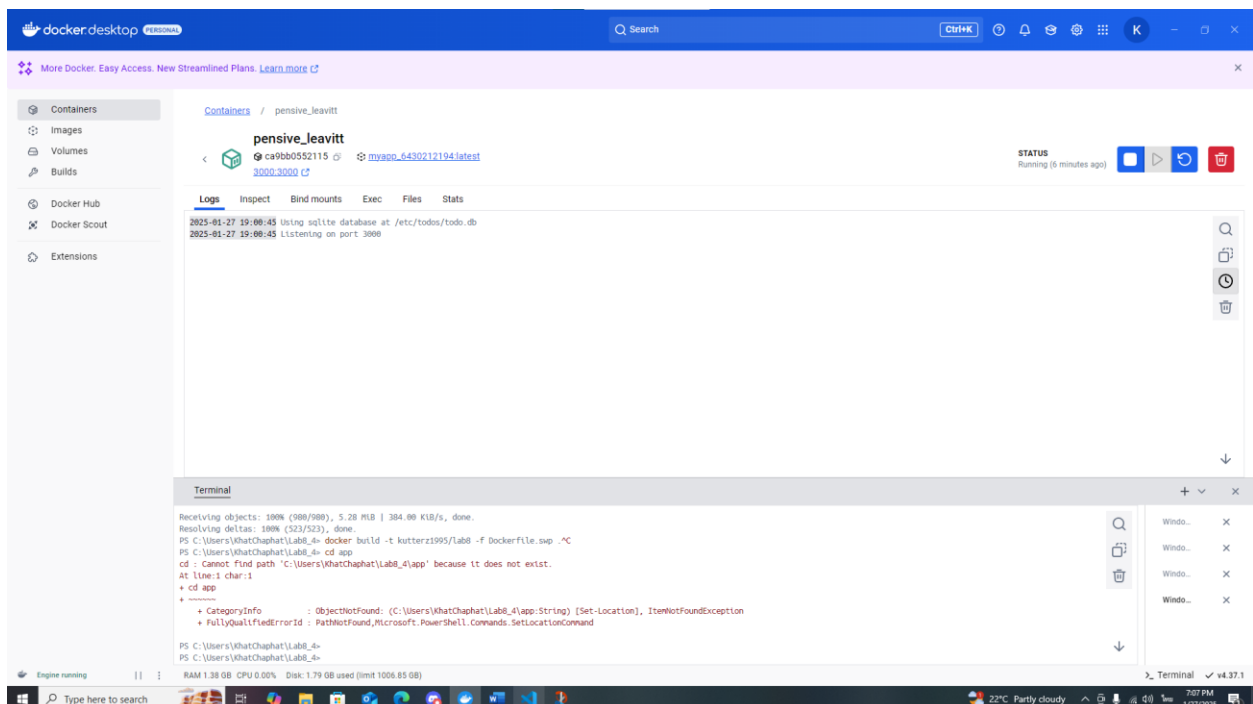
b. ผ่าน Docker desktop

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

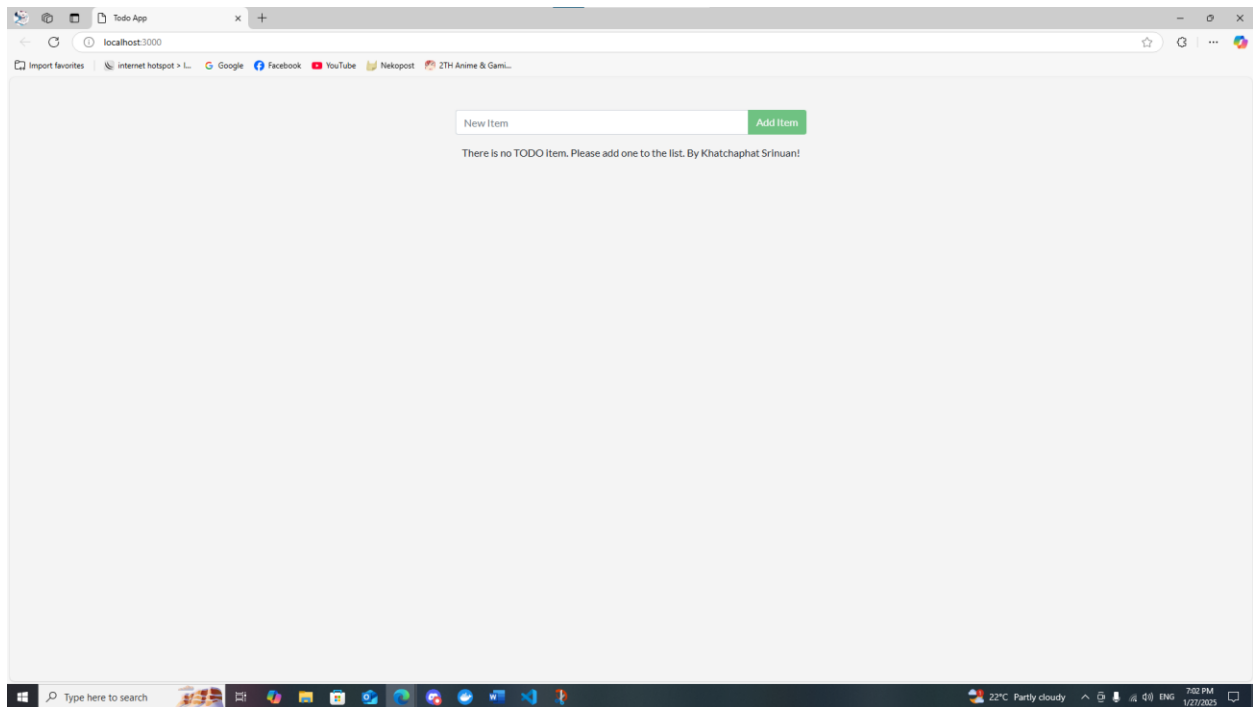
12. Start และรัน Container ตัวใหม่อีกครั้ง โดยใช้คำสั่งเดียวกันกับข้อ 6

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

[Check point#11] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงผลลัพธ์ที่ได้นบน 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

## Lab Worksheet

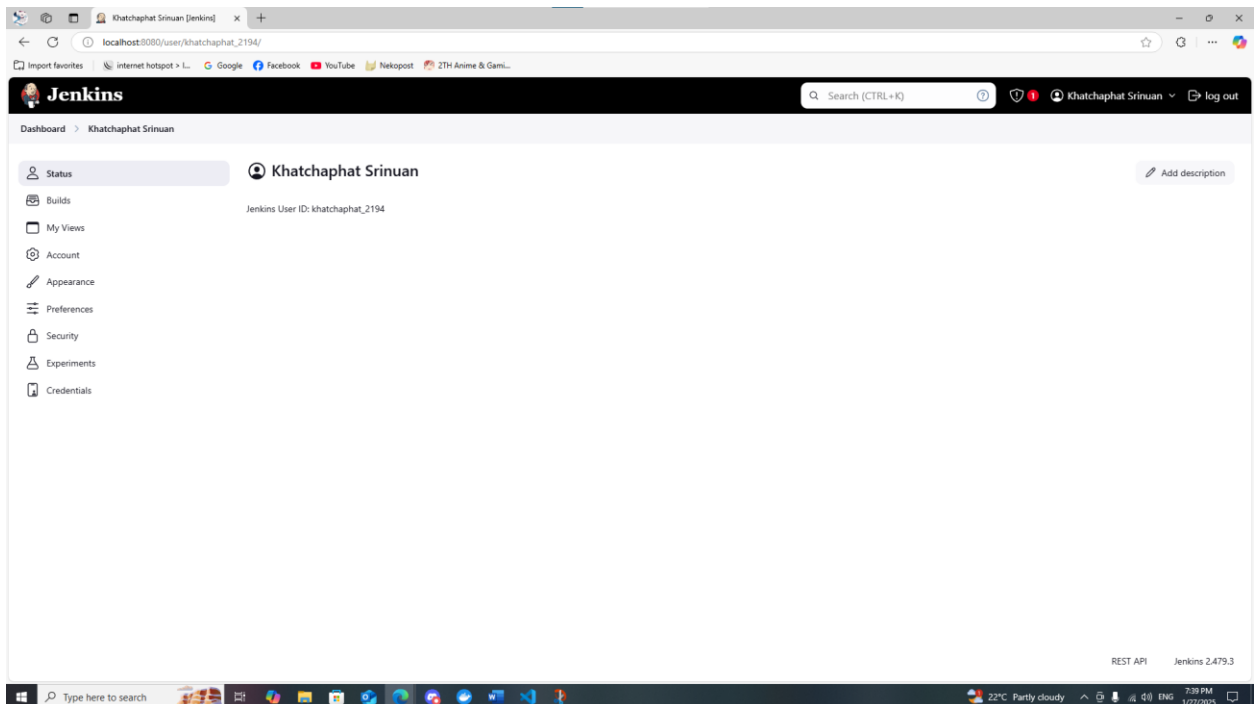
## [Check point#12] Capture หน้าจอที่แสดงผล Admin password

The screenshot shows the Docker Desktop interface for a container named 'magical\_hawking'. The 'Files' tab is active, displaying a list of files. The file 'initialAdminPassword' is selected, showing its content: '4923a588508f4f1c86d81a2cb0b6e6b'. The 'Terminal' tab is also visible, showing logs and the command 'cat /var/jenkins\_home/secrets/initialAdminPassword'.

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

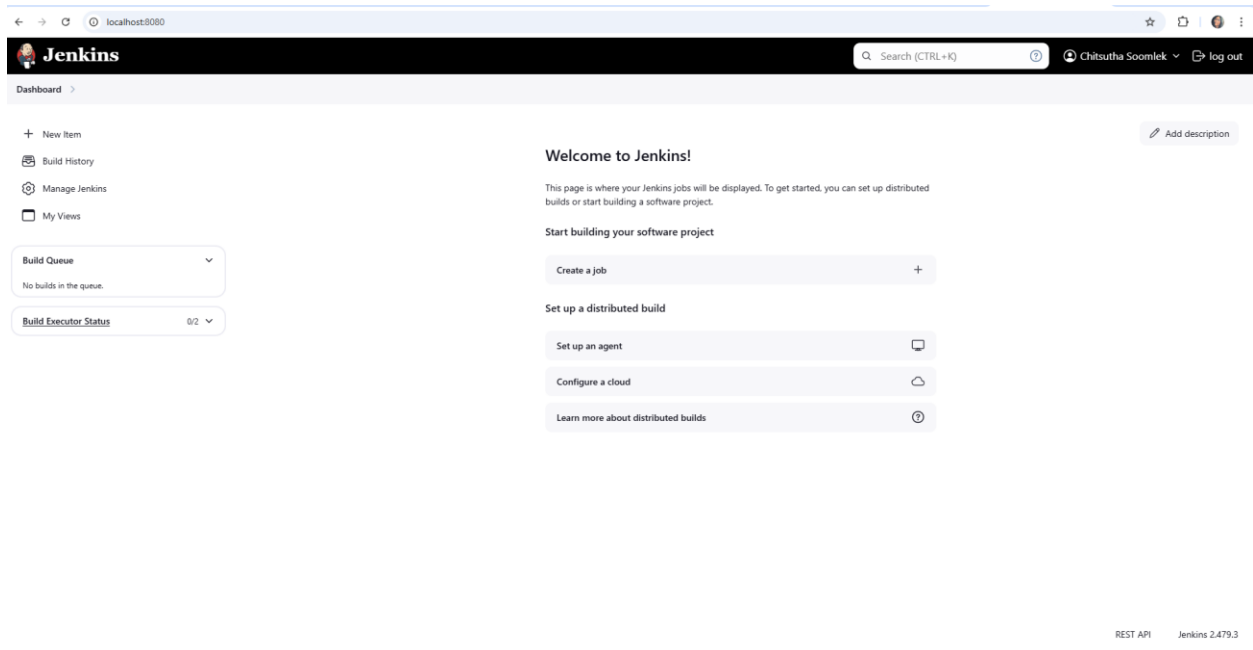
## Lab Worksheet

[Check point#13] Capture หน้าจอที่แสดงผลการตั้งค่า



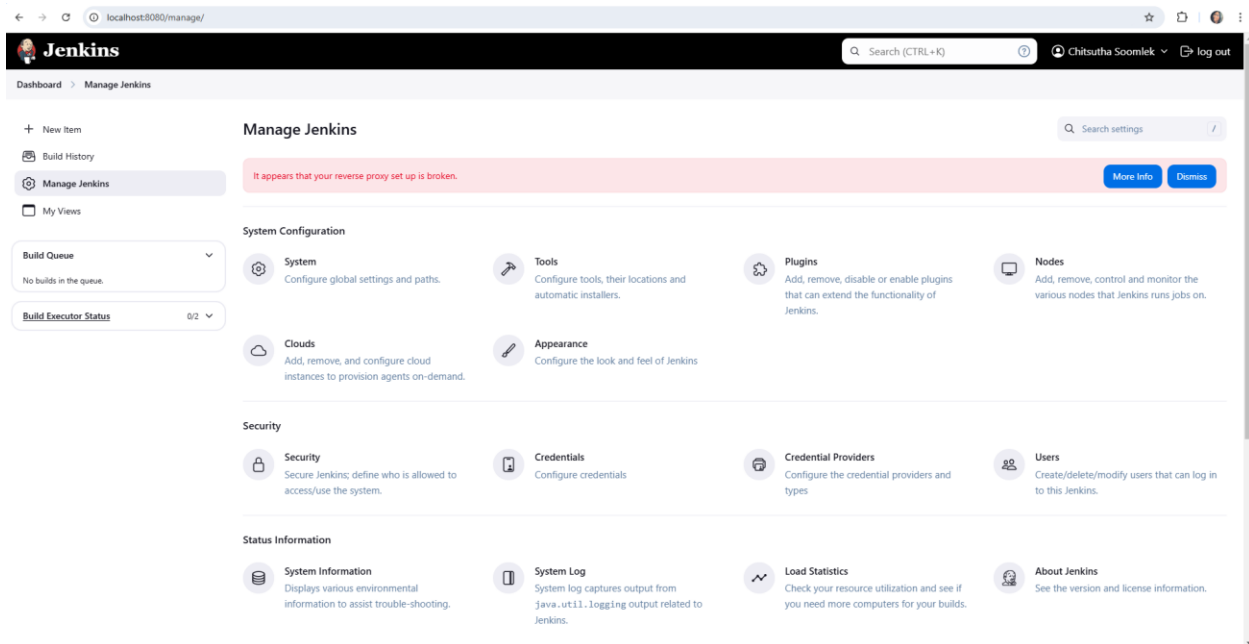
7. กำหนด Jenkins URL เป็น <http://localhost:8080/lab8>

8. เมื่อติดตั้งเรียบร้อยแล้วจะพบกันหน้า Dashboard ดังแสดงในภาพ



9. เลือก Manage Jenkins แล้วไปที่เมนู Plugins

## Lab Worksheet



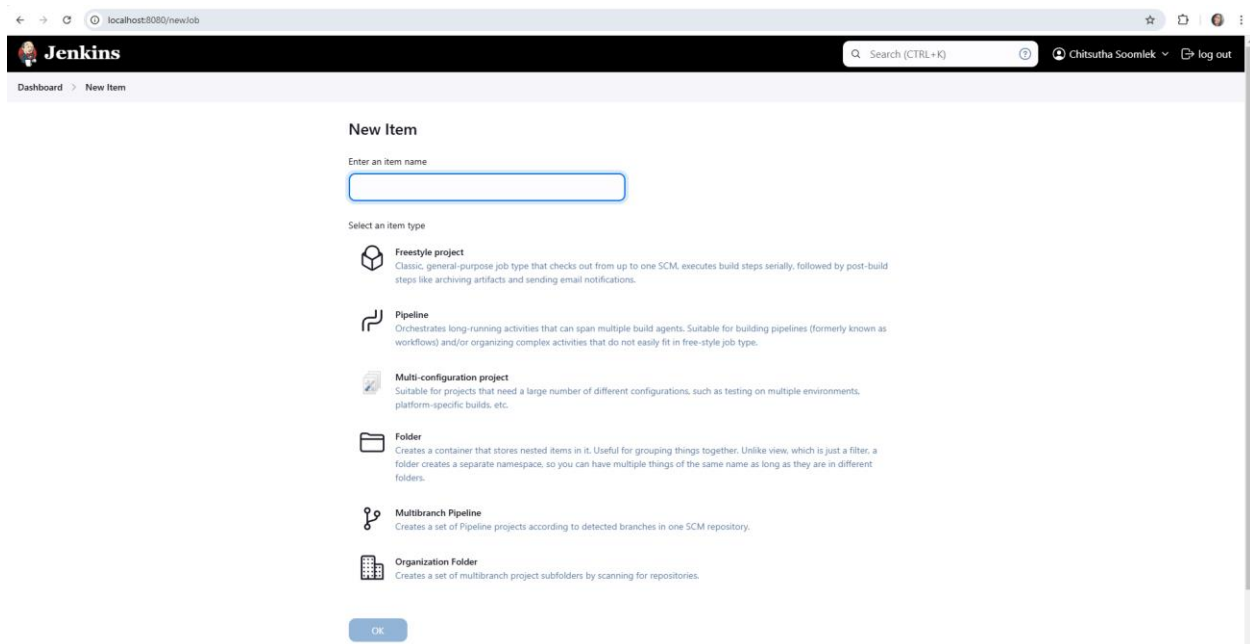
10. ไปที่เมนู Available plugins แล้วเลือกติดตั้ง Robotframework เพิ่มเติม



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)

[Check point#14] Capture หน้าจอแสดงการตั้งค่า พร้อมกับตอบคำถามต่อไปนี้

## Lab Worksheet

The screenshot shows the Jenkins web interface for configuring a job named 'UAT Config'. The 'General' tab is selected in the left sidebar. The main configuration area includes a 'Description' field with the text 'Lab 8.5'. Under the 'GitHub project' section, the 'Project url' is set to 'https://github.com/Khatchaphat/Lab-8.5.git/'. The 'Build Triggers' section shows 'Build periodically' selected with a schedule of 'H/15 \* \* \* \*'. The 'Build Environment' section has several options, with 'Delete workspace before build starts' and 'Use secret text(s) or file(s)' being checked. The 'Build Steps' section is currently empty. The interface is running on a Windows 10 desktop, as indicated by the taskbar and system tray.

Dashboard > UAT > Configuration

### Configure

- General
- Source Code Management
- Build Triggers
- Build Environment
- Build Steps
- Post-build Actions

#### General

Enabled

Description

Lab 8.5

Plain text [Preview](#)

☐ Discard old builds [?](#)

☒ GitHub project

Project url [?](#)

[https://github.com/Khatchaphat/Lab-8.5.git/](#)

[Advanced](#)

☐ This project is parameterized [?](#)

☐ Throttle builds [?](#)

☐ Execute concurrent builds if necessary [?](#)

[Advanced](#)

[Save](#) [Apply](#)

#### Build Triggers

☐ Trigger builds remotely (e.g., from scripts) [?](#)

☐ Build after other projects are built [?](#)

☒ Build periodically [?](#)

Schedule [?](#)

[H/15 \\* \\* \\* \\* \\*](#)

Would last have run at Monday, January 27, 2025 at 1:20:30 PM Coordinated Universal Time; would next run at Monday, January 27, 2025 at 1:35:30 PM Coordinated Universal Time.

☐ GitHub hook trigger for GITScm polling [?](#)

☐ Poll SCM [?](#)

#### Build Environment

☐ Delete workspace before build starts

☒ Use secret text(s) or file(s) [?](#)

☐ Add timestamps to the Console Output

☐ Inspect build log for published build scans

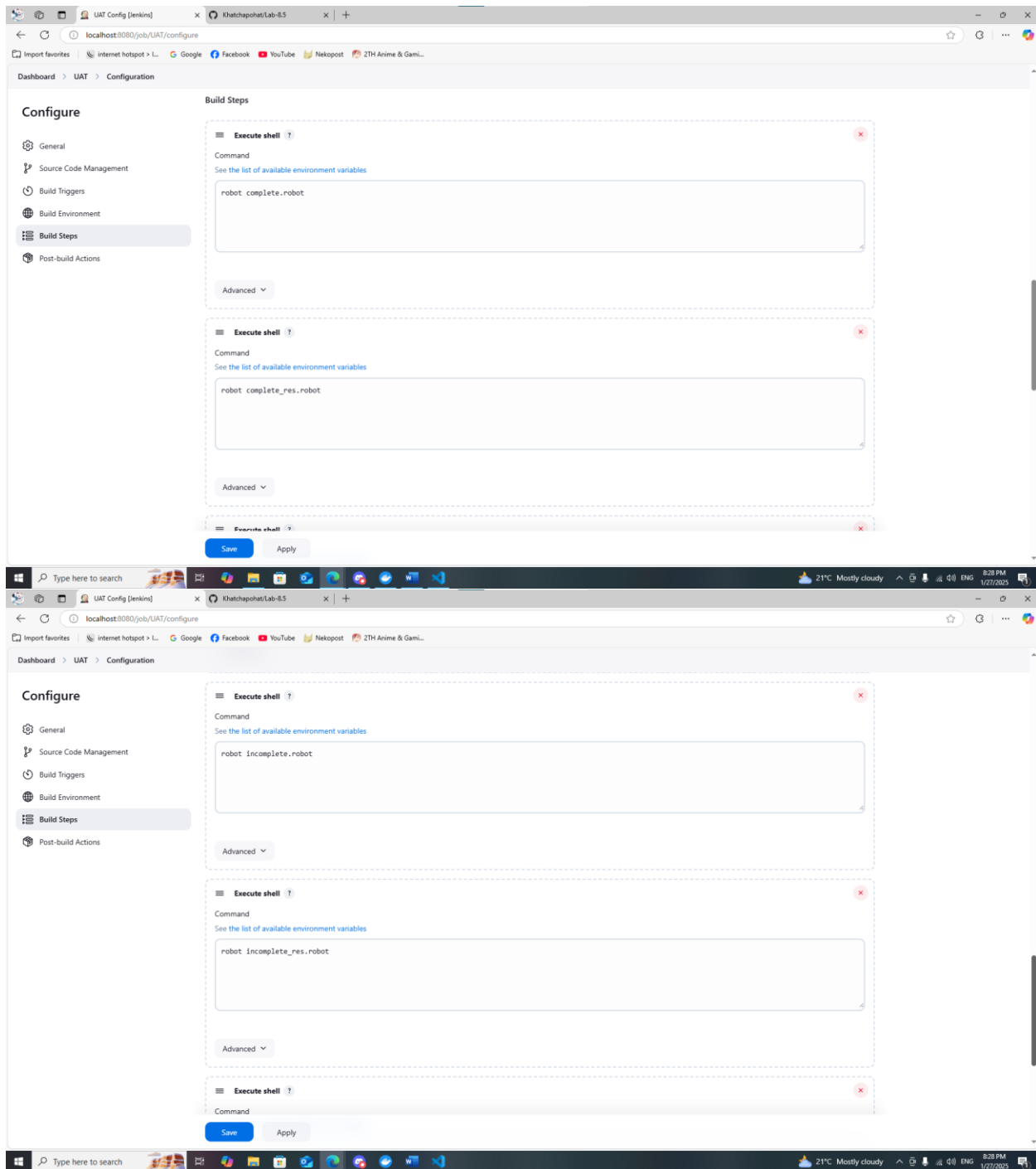
☐ Terminate a build if it's stuck

☐ With Ant [?](#)

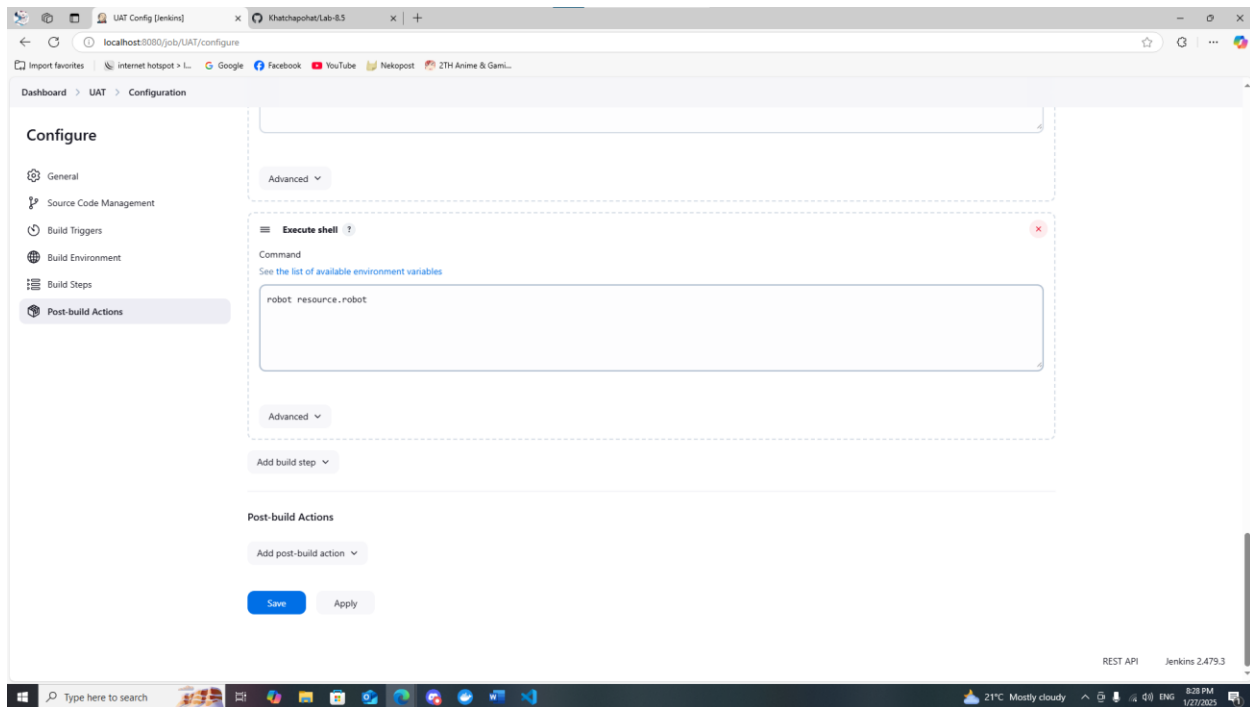
#### Build Steps

[Save](#) [Apply](#)

## Lab Worksheet



## Lab Worksheet



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

robot complete.robot  
 robot complete\_res.robot  
 robot incomplete.robot  
 robot incomplete\_res.robot  
 robot resource.robot

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

ระบุได้เร็คทอรีที่เก็บไฟล์ผลการทดสอบโดย 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

## Lab Worksheet

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

The screenshot displays the Jenkins Dashboard interface. On the left, there is a sidebar with navigation links: 'New Item', 'Build History', 'Manage Jenkins', and 'My Views'. Below these are two status boxes: 'Build Queue' (No builds in the queue) and 'Build Executor Status' (0/2). The main content area features a table of build history. The table has columns for 'S' (Status), 'W' (Workspace), 'Name', 'Last Success', 'Last Failure', 'Last Duration', and 'Robot Results + Duration Trend'. A single build is listed with the name 'UIAT', a status of 'Failed' (red circle with a white 'X'), and a duration of '4 ms'. The 'Last Failure' column shows '1.4 sec #15'. Below the table, there are filters for 'Icon', 'S', 'M', and 'L'. The bottom of the dashboard shows the 'REST API' and 'Jenkins 2.479.3' version. The browser's address bar shows 'localhost:8080' and the user is logged in as 'Khatchaphat Srinuan'.

S	W	Name	Last Success	Last Failure	Last Duration	Robot Results + Duration Trend
Failed		UIAT	N/A	1.4 sec #15	4 ms	