

Lab Worksheet

ชื่อ-นามสกุล _____นางสาวปาริชาติ หงษ์ษา_____รหัสนักศึกษา _____653380206-2_____Section __3__

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 พร้อมกับตอบคำถามต่อไปนี้

The screenshot shows the Docker Desktop interface. The left sidebar has a menu with 'Containers', 'Images', 'Volumes', 'Builds', 'Docker Hub', 'Docker Scout', and 'Extensions'. The 'Images' tab is selected, displaying a table of local images:

Name	Tag	Image ID	Created	Size	Actions
synthesizedio/whalesay	latest	07da125a0bc8	7 months ago	45.24 MB	[Play] [More] [Trash]
busybox	latest	af4709625109	4 months ago	4.26 MB	[Play] [More] [Trash]

Below the table is a terminal window with the following output:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSHWindows

PS C:\Users\Lenovo> cd..
PS C:\Users> cd..
PS C:\> cd CS3_2
PS C:\CS3_2> cd SoftwareEn
PS C:\CS3_2\SoftwareEn> mkdir Lab8_1

Directory: C:\CS3_2\SoftwareEn

Mode                LastWriteTime         Length Name
----                -
d-----          1/23/2025   9:58 AM             Lab8_1
```

At the bottom, system resources are shown: RAM 1.18 GB, CPU 0.00%, Disk: 3.86 GB used (limit 1006.85 GB). The terminal window title is 'Terminal v4.37.1'.

The screenshot shows the Docker Desktop interface. The left sidebar has a menu with 'Containers', 'Images', 'Volumes', 'Builds', 'Docker Hub', 'Docker Scout', and 'Extensions'. The 'Images' tab is selected, displaying a table of local images:

Name	Tag	Image ID	Created	Size	Actions
synthesizedio/whalesay	latest	07da125a0bc8	7 months ago	45.24 MB	[Play] [More] [Trash]
busybox	latest	af4709625109	4 months ago	4.26 MB	[Play] [More] [Trash]

Below the table is a terminal window with the following output:

```
PS C:\CS3_2\SoftwareEn> cd Lab8_1
PS C:\CS3_2\SoftwareEn\Lab8_1> docker pull busybox
Using default tag: latest
latest: Pulling from library/busybox
9c0abc9c5bd3: Pull complete
Digest: sha256:e5d0ce49ea801d475da48f8cb163c354ab95cab073cd3c138bd458fc8257fbf1
Status: Downloaded newer image for busybox:latest
docker.io/library/busybox:latest

What's next:
View a summary of image vulnerabilities and recommendations --docker scout quickview busybox
PS C:\CS3_2\SoftwareEn\Lab8_1> docker images
REPOSITORY          TAG             IMAGE ID        CREATED        SIZE
busybox              latest          af4709625109   3 months ago   4.27MB
synthesizedio/whalesay latest          07da125a0bc8   6 months ago   45.2MB
PS C:\CS3_2\SoftwareEn\Lab8_1> ^C
PS C:\CS3_2\SoftwareEn\Lab8_1> ^C
PS C:\CS3_2\SoftwareEn\Lab8_1> []
```

At the bottom, system resources are shown: RAM 1.23 GB, CPU 0.00%, Disk: 3.86 GB used (limit 1006.85 GB). The terminal window title is 'Terminal v4.37.1'.

Lab Worksheet

- (1) สิ่งที่อยู่ภายใต้คอนเทนเนอร์ Repository คืออะไร [busybox](#)
- (2) Tag ที่ใช้บ่งบอกถึงอะไร [latest](#): เป็น Tag ที่ใช้ระบุ Image เวอร์ชันล่าสุดที่ Repository กำหนด ถ้าไม่ระบุ Tag ในคำสั่ง เช่น `docker pull busybox` ระบบจะใช้ latest โดยอัตโนมัติ
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

The screenshot shows the Docker Desktop interface. The left sidebar has a menu with 'Containers', 'Images', 'Volumes', 'Builds', 'Docker Hub', 'Docker Scout', and 'Extensions'. The 'Images' tab is selected, displaying a table of local and Docker Hub images:

Name	Tag	Image ID	Created	Size	Actions
synthesizedio/whalesay	latest	07da125a0bc8	7 months ago	45.24 MB	[Play] [More] [Trash]
busybox	latest	af4709625109	4 months ago	4.26 MB	[Play] [More] [Trash]

Below the images table is a terminal window with the following commands and output:

```
PS C:\CS3_2\SoftwareEn\Lab8_1> docker run busybox
PS C:\CS3_2\SoftwareEn\Lab8_1> docker run -it busybox sh
/ # ls
/ # ls -la
total 48
drwxr-xr-x 1 root root      4096 Jan 23 03:11 .
drwxr-xr-x 1 root root      4096 Jan 23 03:11 ..
-rwxr-xr-x 1 root root         0 Jan 23 03:11 .dockerenv
drwxr-xr-x 2 root root     12288 Sep 26 21:31 bin
drwxr-xr-x 5 root root      360 Jan 23 03:11 dev
drwxr-xr-x 1 root root      4096 Jan 23 03:11 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 305 root root         0 Jan 23 03:11 proc
drwx----- 1 root root      4096 Jan 23 03:11 root
dr-xr-xr-x 11 root root         0 Jan 23 03:11 sys
drwxrwxrwt 2 root root      4096 Sep 26 21:31 tmp
drwxr-xr-x 4 root root      4096 Sep 26 21:31 usr
```

The bottom status bar shows 'Engine running', 'RAM 1.30 GB', 'CPU 0.00%', and 'Disk: 3.86 GB used (limit 1006.85 GB)'.

The screenshot shows the Docker Desktop interface. The left sidebar has a menu with 'Containers', 'Images', 'Volumes', 'Builds', 'Docker Hub', 'Docker Scout', and 'Extensions'. The 'Images' tab is selected, displaying a table of local and Docker Hub images:

Name	Tag	Image ID	Created	Size	Actions
synthesizedio/whalesay	latest	07da125a0bc8	7 months ago	45.24 MB	[Play] [More] [Trash]
busybox	latest	af4709625109	4 months ago	4.26 MB	[Play] [More] [Trash]

Below the images table is a terminal window with the following commands and output:

```
PS C:\CS3_2\SoftwareEn\Lab8_1> docker run busybox echo "Hello Parichat Hongsa from busybox"
Hello Parichat Hongsa from busybox
PS C:\CS3_2\SoftwareEn\Lab8_1> docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
bcb712dd611	busybox	"echo 'Hello Paricha..."	8 seconds ago	Exited (0) 8 seconds ago		agitated_ard
844d781385ad	busybox	"sh"	About a minute ago	Exited (0) About a minute ago		sleepy_wiles
5fd90ed4e97	busybox	"sh"	2 minutes ago	Exited (0) 2 minutes ago		hungry_solom
777b31672adc	synthesizedio/whalesay	"/usr/local/bin/cows..."	59 minutes ago	Exited (0) 59 minutes ago		vigorous_mon

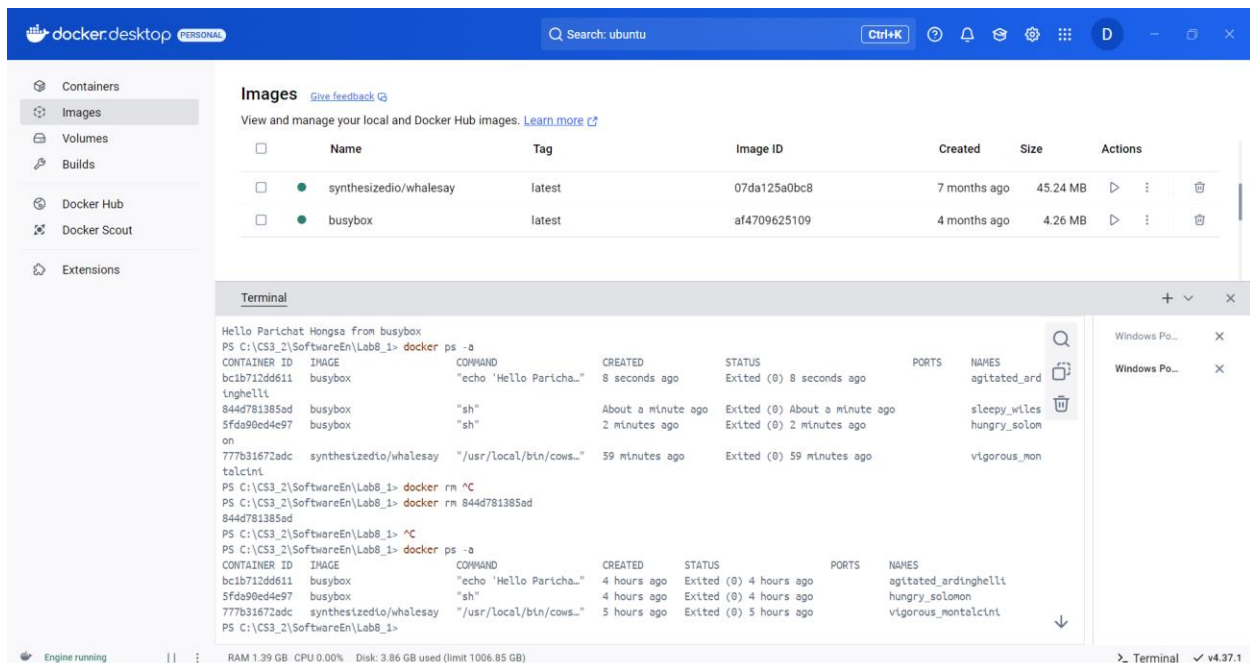
The bottom status bar shows 'Engine running', 'RAM 1.32 GB', 'CPU 0.00%', and 'Disk: 3.86 GB used (limit 1006.85 GB)'.

Lab Worksheet

- เมื่อใช้ option -it ในคำสั่ง run ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป
ทำให้สามารถโต้ตอบ (interactive) กับคอนเทนเนอร์ผ่าน terminal ได้ โดยเปิด shell หรือโปรแกรม
ในลักษณะที่ใช้งานง่ายและเหมาะกับการทำงานแบบเรียลไทม์ในคอนเทนเนอร์ เช่น การพิมพ์คำสั่งหรือ
รับผลลัพธ์โดยตรงใน terminal
- คอลัมน์ STATUS จากการรันคำสั่ง docker ps -a แสดงถึงข้อมูลอะไร
คอลัมน์ STATUS ใช้สำหรับตรวจสอบว่าคอนเทนเนอร์กำลังทำงานอยู่หรือไม่ และสถานะการทำงานใน
ปัจจุบัน พร้อมบอกช่วงเวลาเมื่อสถานะนั้นเกิดขึ้น เช่น คอนเทนเนอร์เพิ่งเริ่มหรือหยุดทำงานเมื่อใด

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

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



แบบฝึกปฏิบัติที่ 8.2: สร้าง Docker file และ Docker image

- เปิดใช้งาน Docker desktop และ Login ด้วย Username และ Password ที่ลงทะเบียนกับ Docker Hub เอาไว้
- เปิด Command line หรือ Terminal จากนั้นสร้าง Directory ชื่อ Lab8_2
- ย้ายตำแหน่งปัจจุบันไปที่ Lab8_2 เพื่อใช้เป็น Working directory
- สร้าง Dockerfile.swp ไว้ใน Working directory

Lab Worksheet

สำหรับเครื่องที่ใช้ระบบปฏิบัติการวินโดวส์ (Windows) บันทึกคำสั่งต่อไปนี้ลงในไฟล์ โดยใช้ Text Editor ที่มี

```
FROM busybox
```

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

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

สำหรับเครื่องที่ใช้ระบบปฏิบัติการ MacOS หรือ Linux บนหน้าต่าง Terminal 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

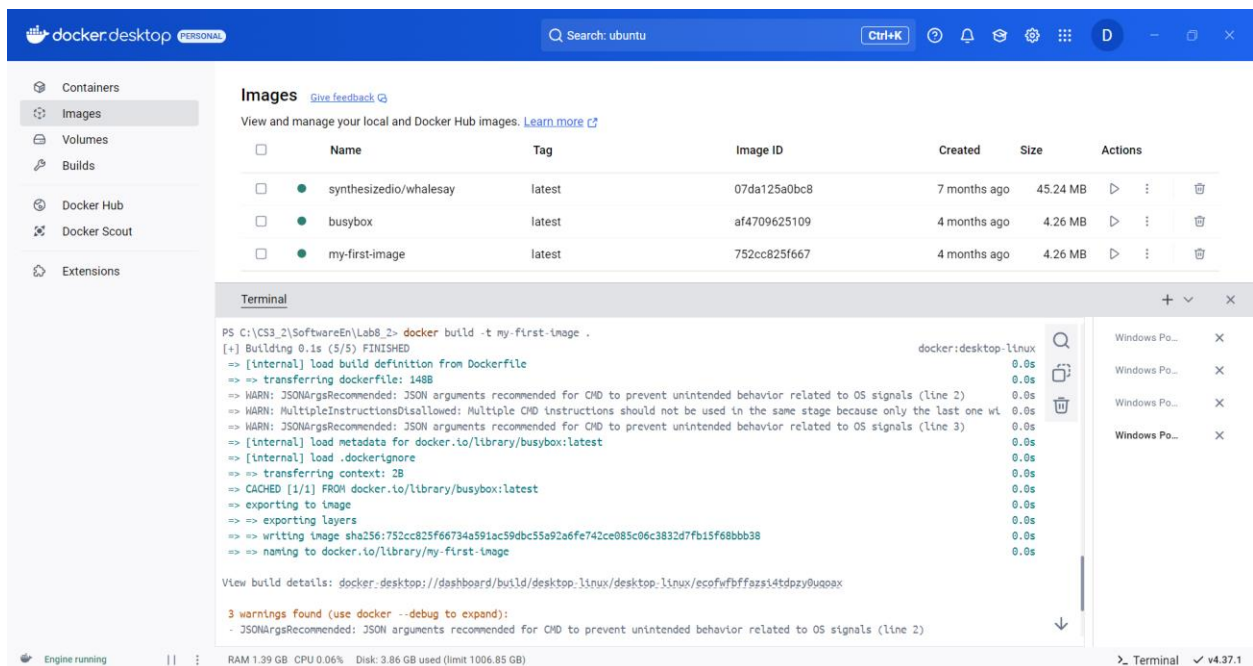
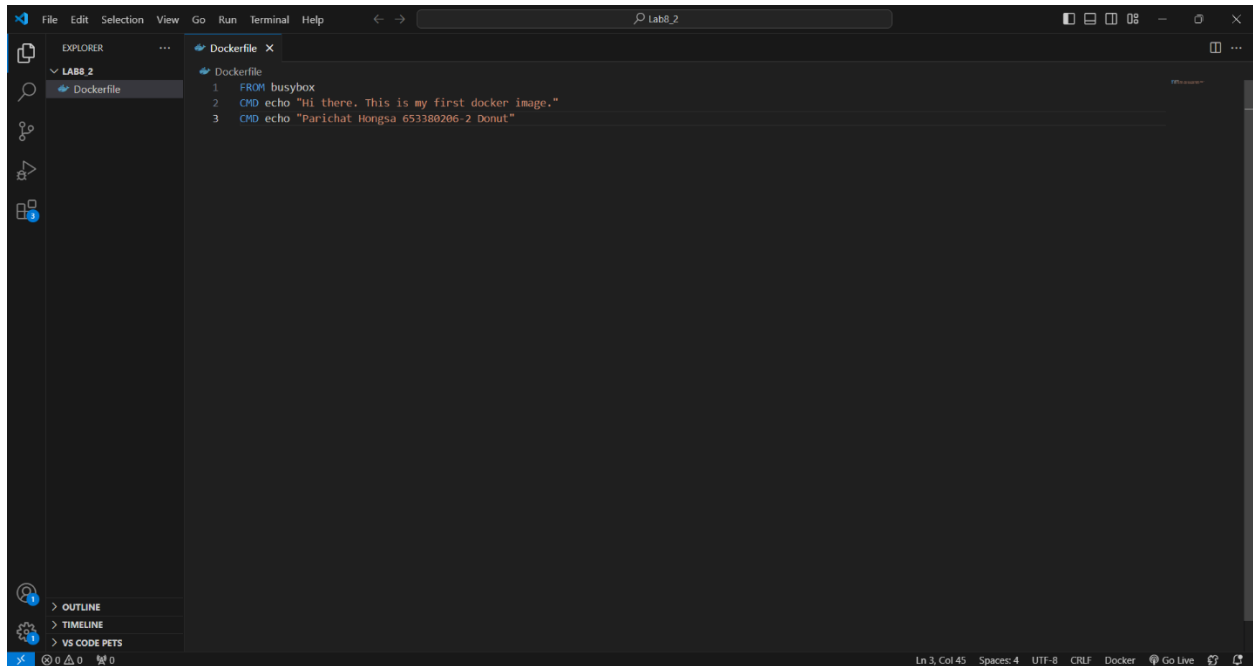
5. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้

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

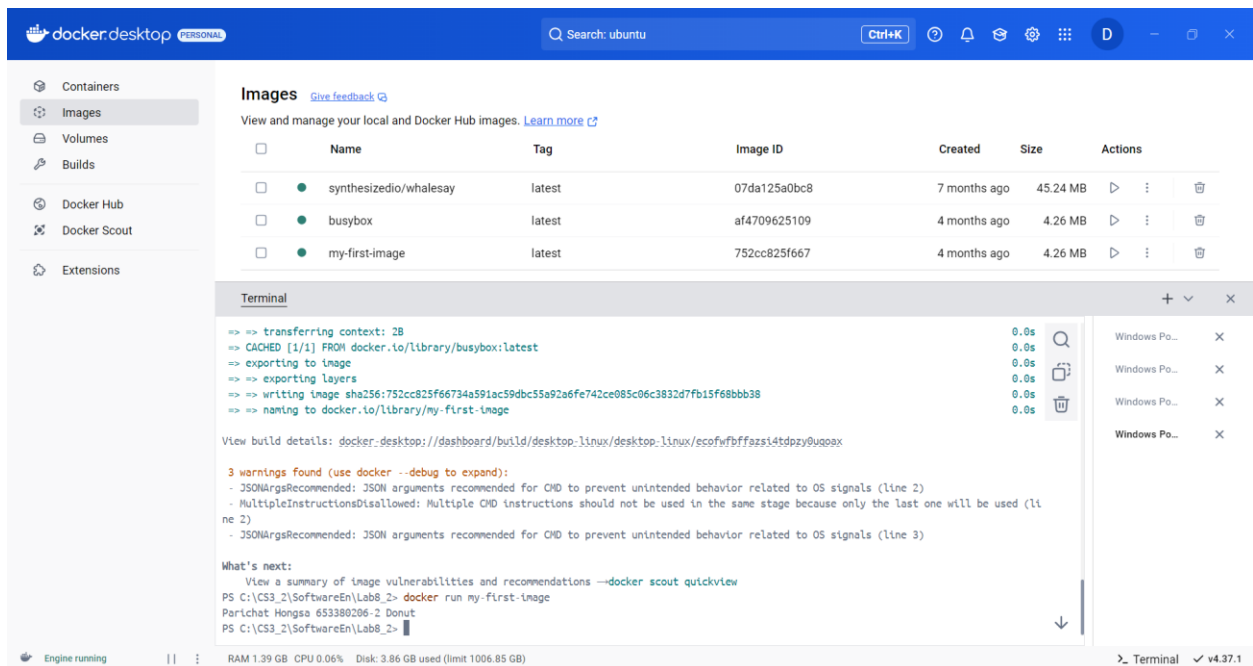
6. เมื่อ Build สำเร็จแล้ว ให้ทำการรัน Docker image ที่สร้างขึ้นในขั้นตอนที่ 5

Lab Worksheet

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



Lab Worksheet



- (1) คำสั่งที่ใช้ในการ run คือ `docker run my-first-image`
- (2) Option -t ในคำสั่ง `$ docker build` ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป
 Option -t ในคำสั่ง `$ docker build` ใช้สำหรับ กำหนดชื่อ (tag) ให้กับ Docker Image ที่สร้างขึ้น ทำให้จัดการและเรียกใช้งาน Image ได้ง่ายขึ้นในภายหลัง เช่น การรันหรือแชร์ไปยัง Docker Hub โดย `my-first-image` คือชื่อ (tag) ที่กำหนดให้ Image และทำให้ง่ายต่อการอ้างอิง Image ด้วยชื่อที่ตั้งไว้ แทนการใช้ Image ID ที่ยาวและจำยาก

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

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

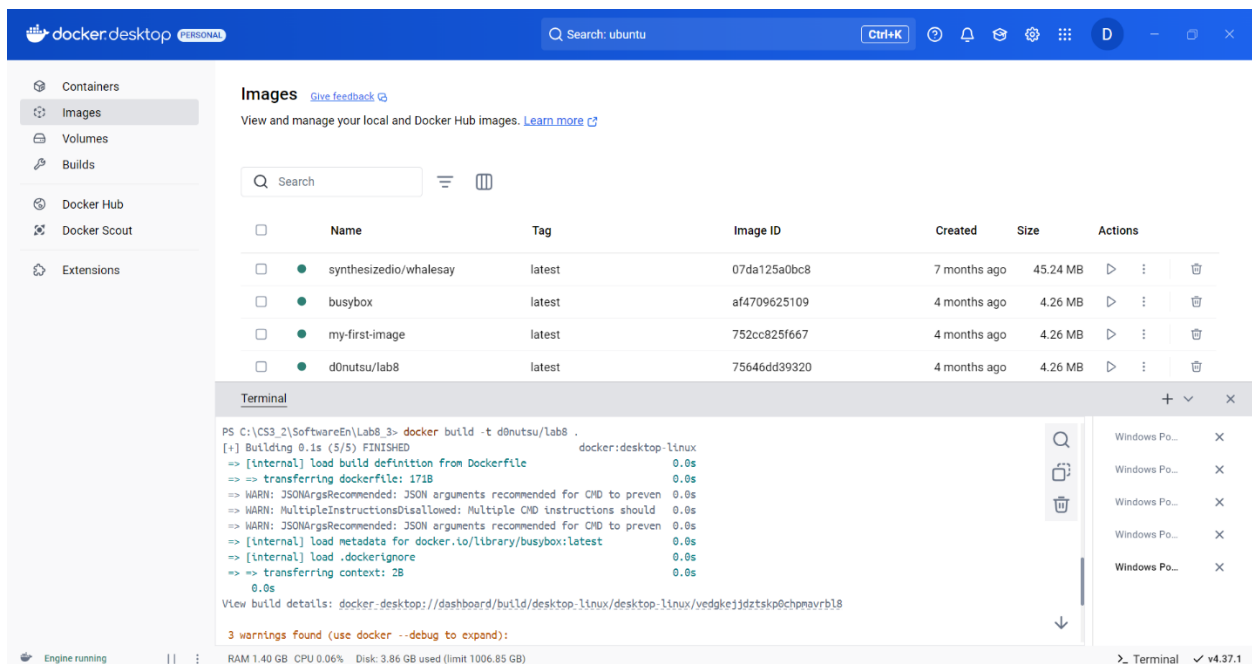
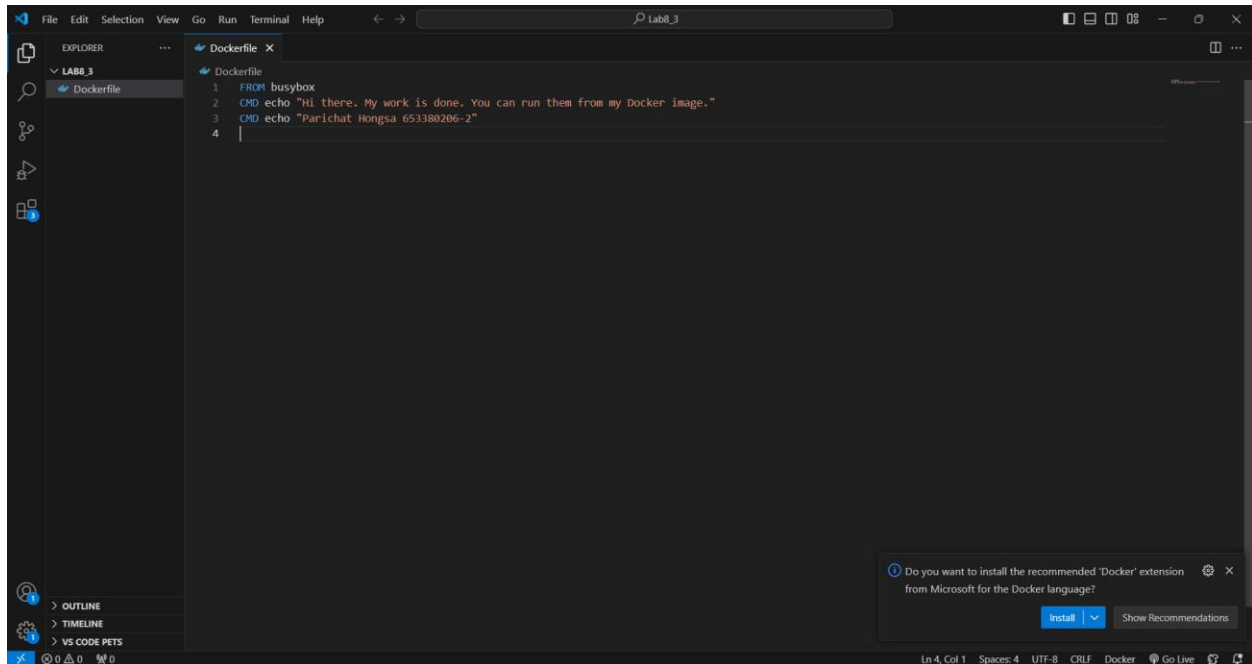
```
$ docker build -t <username ที่ลงทะเบียนกับ Docker Hub>/lab8
```

5. ทำการรัน Docker image บน Container ในเครื่องของตัวเองเพื่อทดสอบผลลัพธ์ ด้วยคำสั่ง

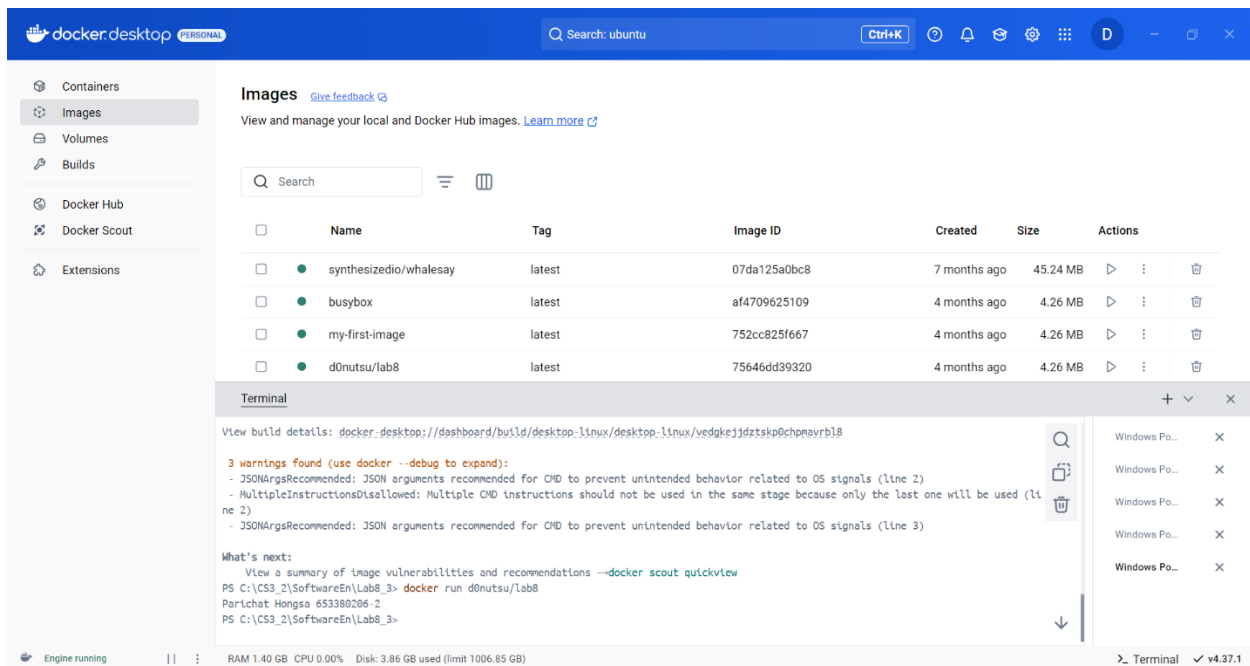
```
$ docker run <username ที่ลงทะเบียนกับ Docker Hub>/lab8
```

Lab Worksheet

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



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

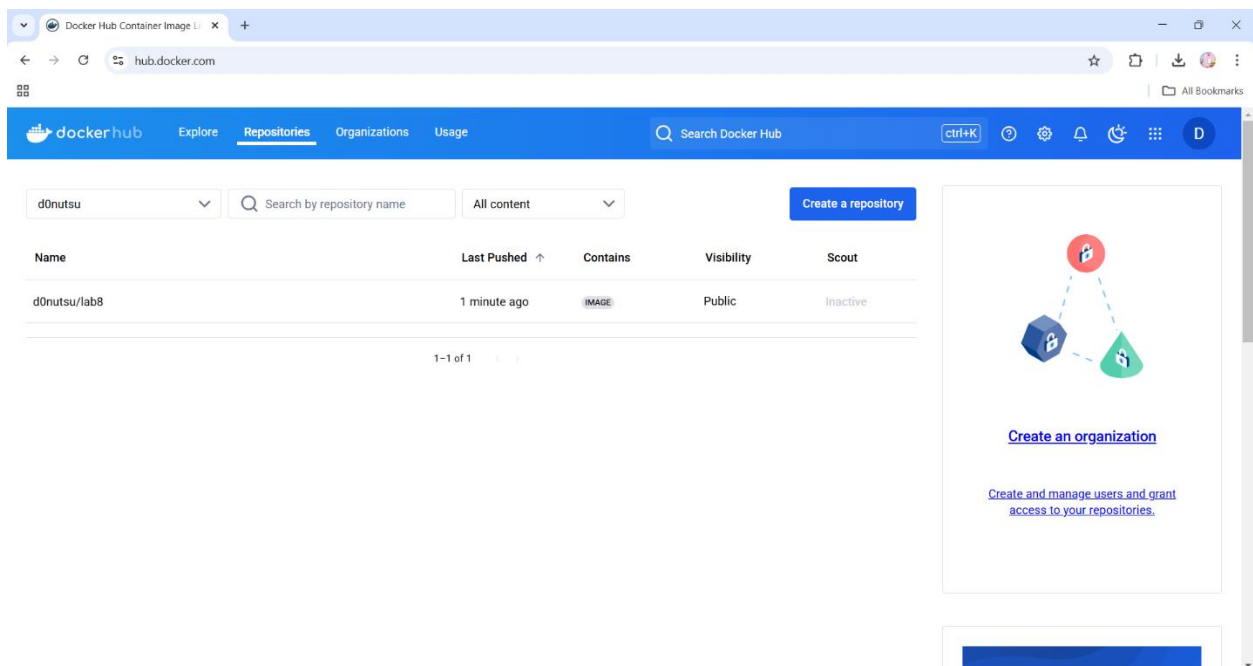
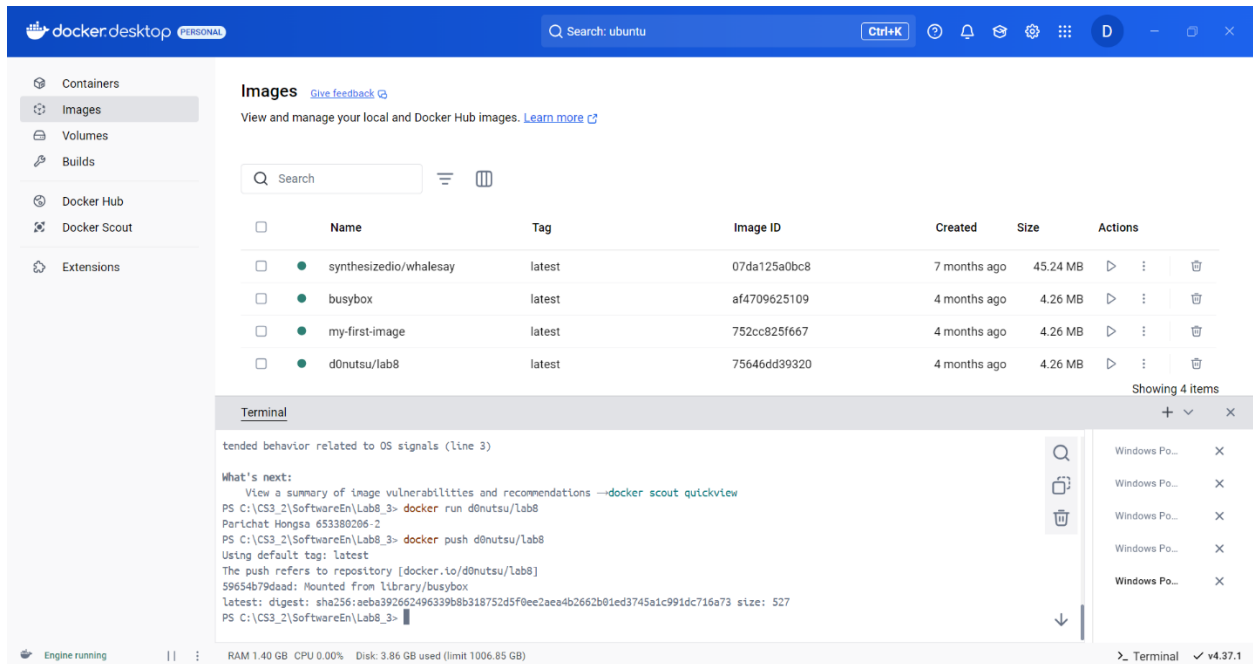
\$ docker login แล้วป้อน Username และ Password ตามที่ระบุใน Command prompt หรือใช้คำสั่ง

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

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

Lab Worksheet

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

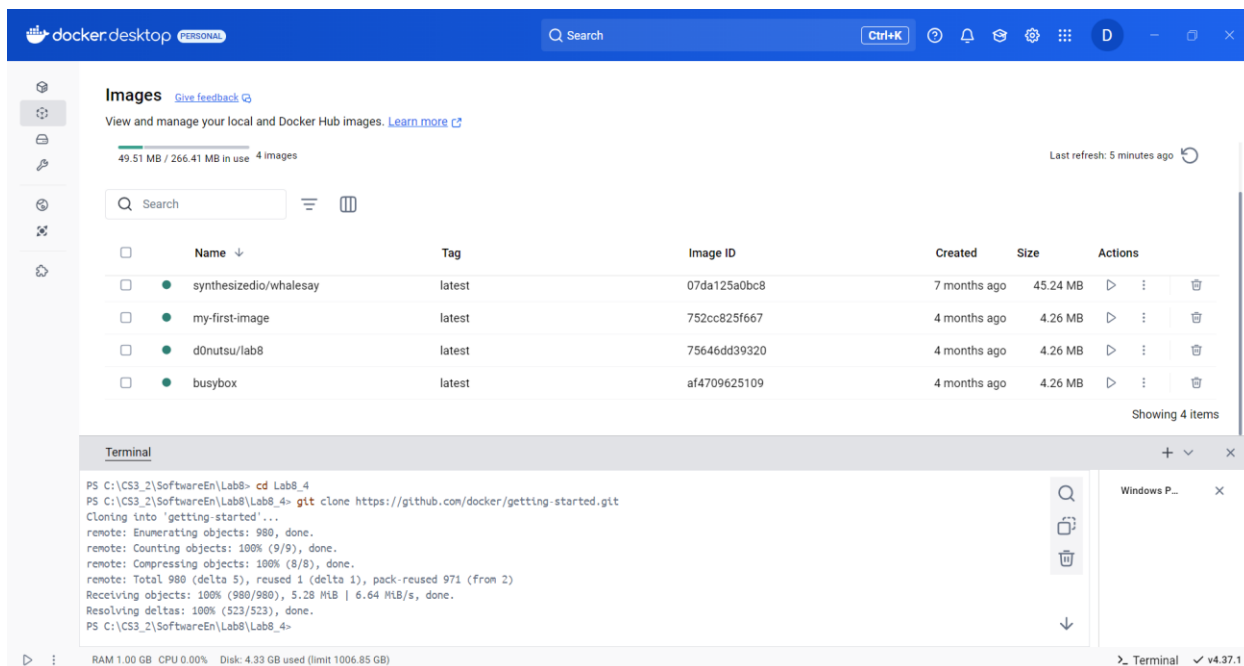


Lab Worksheet

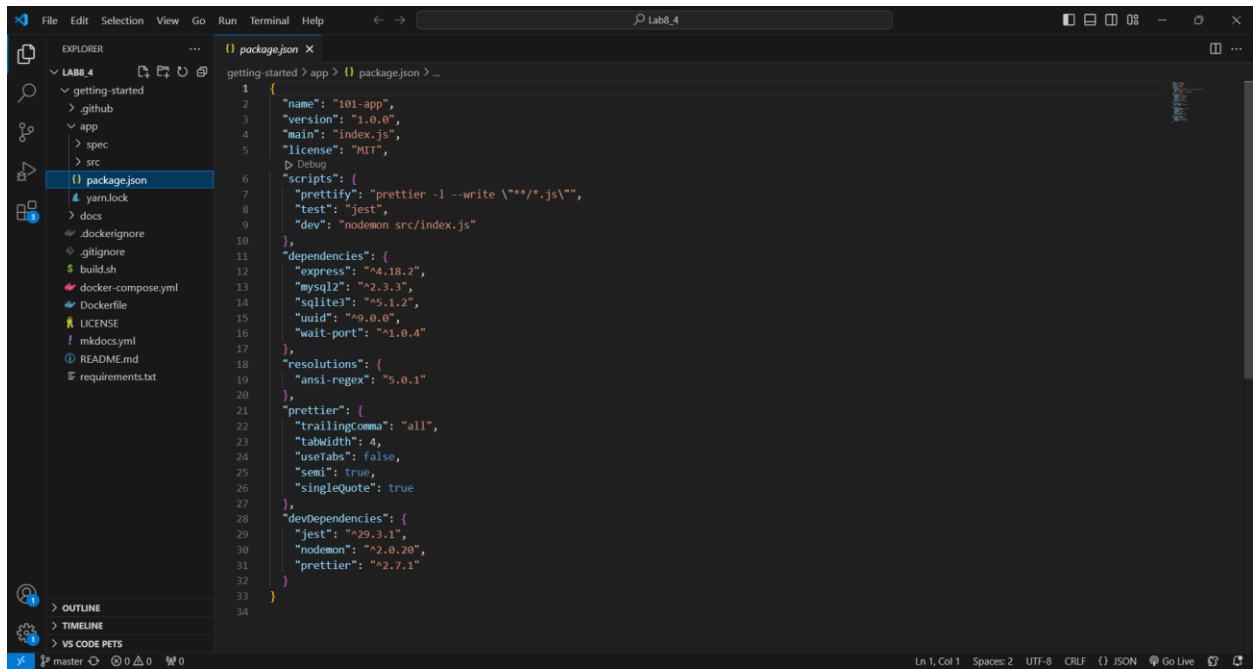
แบบฝึกปฏิบัติที่ 8.4: การ Build แอปพลิเคชันจาก Container image และการ Update แอปพลิเคชัน

1. เปิด Command line หรือ Terminal จากนั้นสร้าง Directory ชื่อ Lab8_4
2. ทำการ Clone ซอร์สโค้ดของเว็บแอปพลิเคชันจาก GitHub repository
<https://github.com/docker/getting-started.git> ลงใน Directory ที่สร้างขึ้น โดยใช้คำสั่ง
`$ git clone https://github.com/docker/getting-started.git`
3. เปิดดูองค์ประกอบภายใน getting-started/app เมื่อพบไฟล์ package.json ให้ใช้ Text editor ในการเปิดอ่าน

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



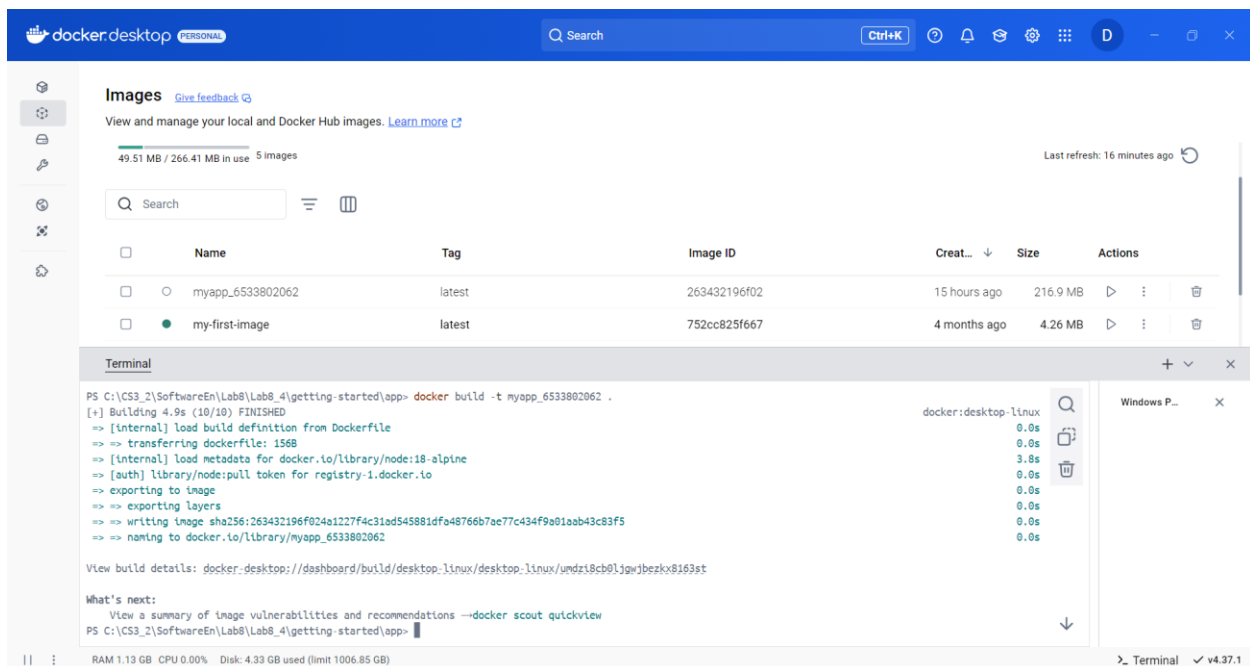
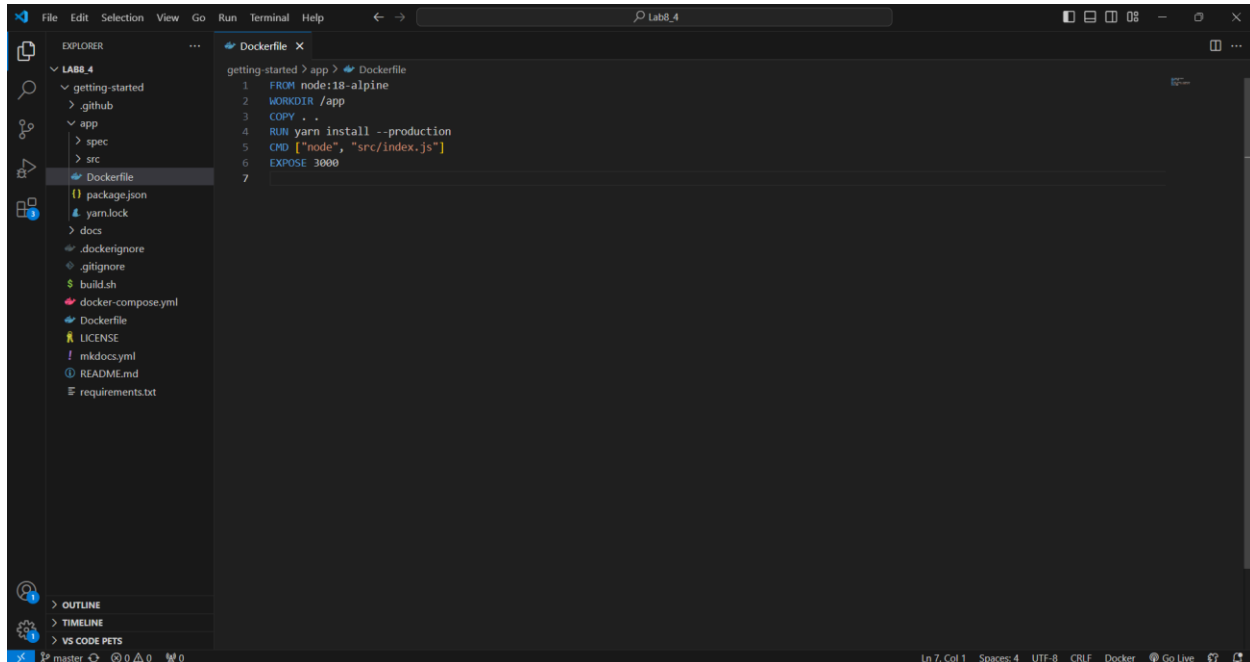
Lab Worksheet



4. ภายใต้ getting-started/app ให้สร้าง Dockerfile พร้อมกับใส่เนื้อหาดังต่อไปนี้ลงไปในไฟล์
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_รหัสสนศ. ไม่มีขีด
\$ docker build -t <myapp_รหัสสนศ. ไม่มีขีด> .

Lab Worksheet

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



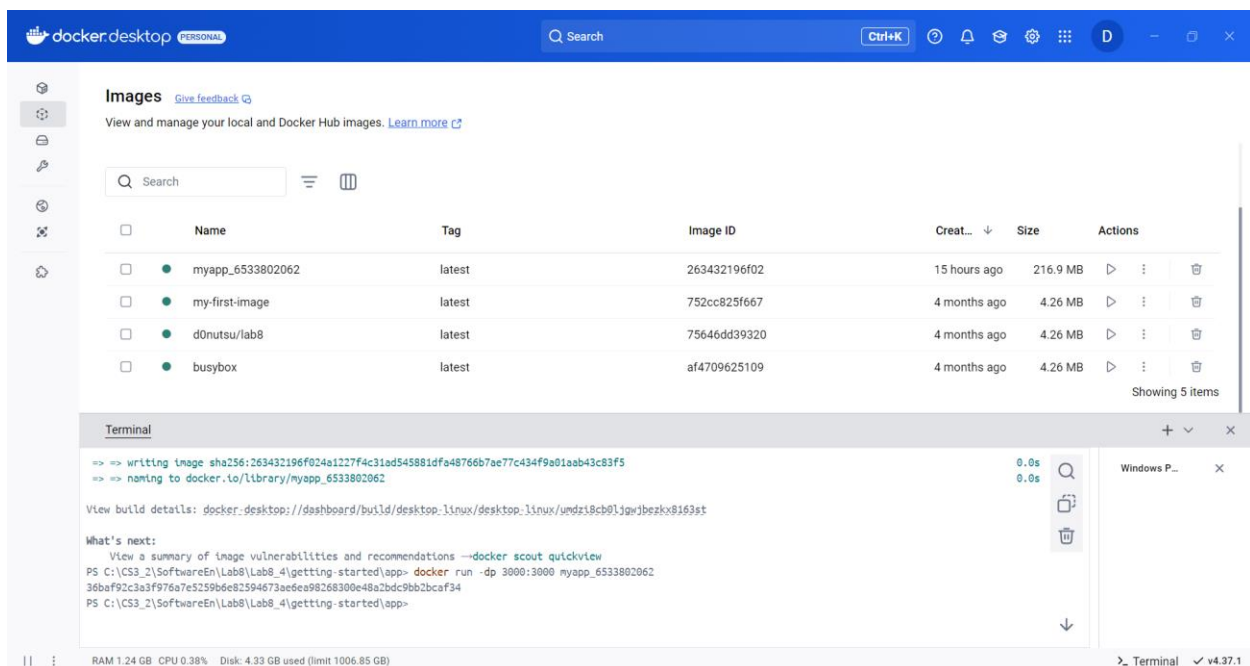
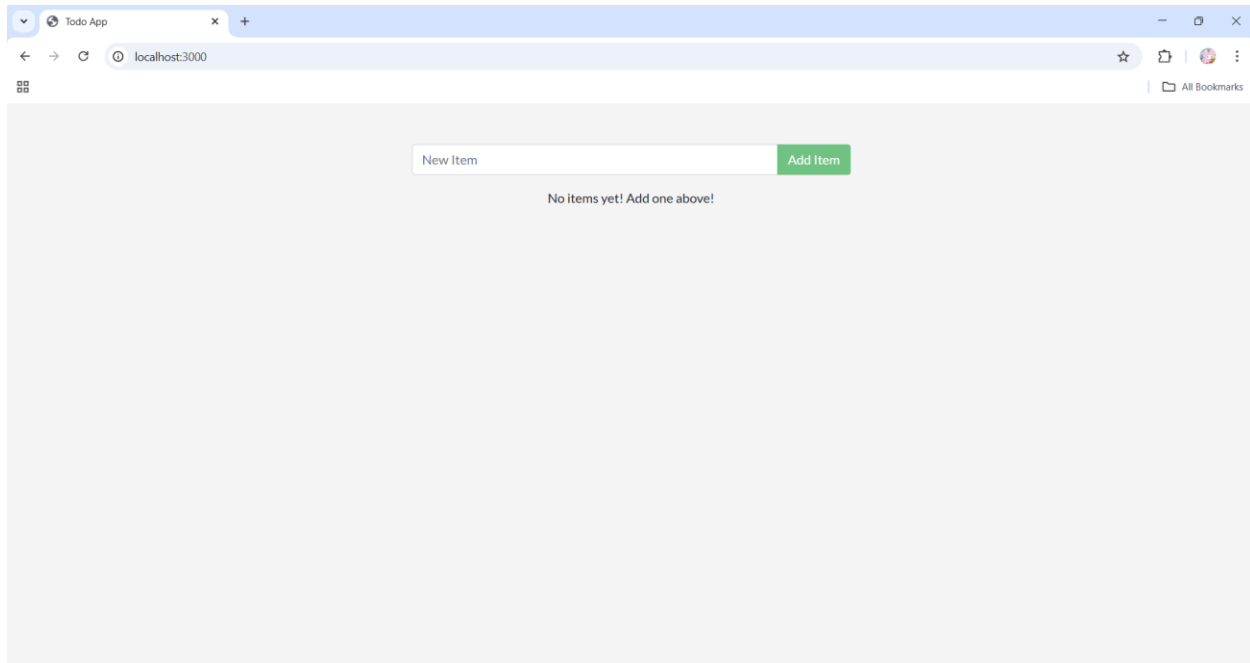
Lab Worksheet

6. ทำการ Start ตัว Container ของแอปพลิเคชันที่สร้างขึ้น โดยใช้คำสั่ง

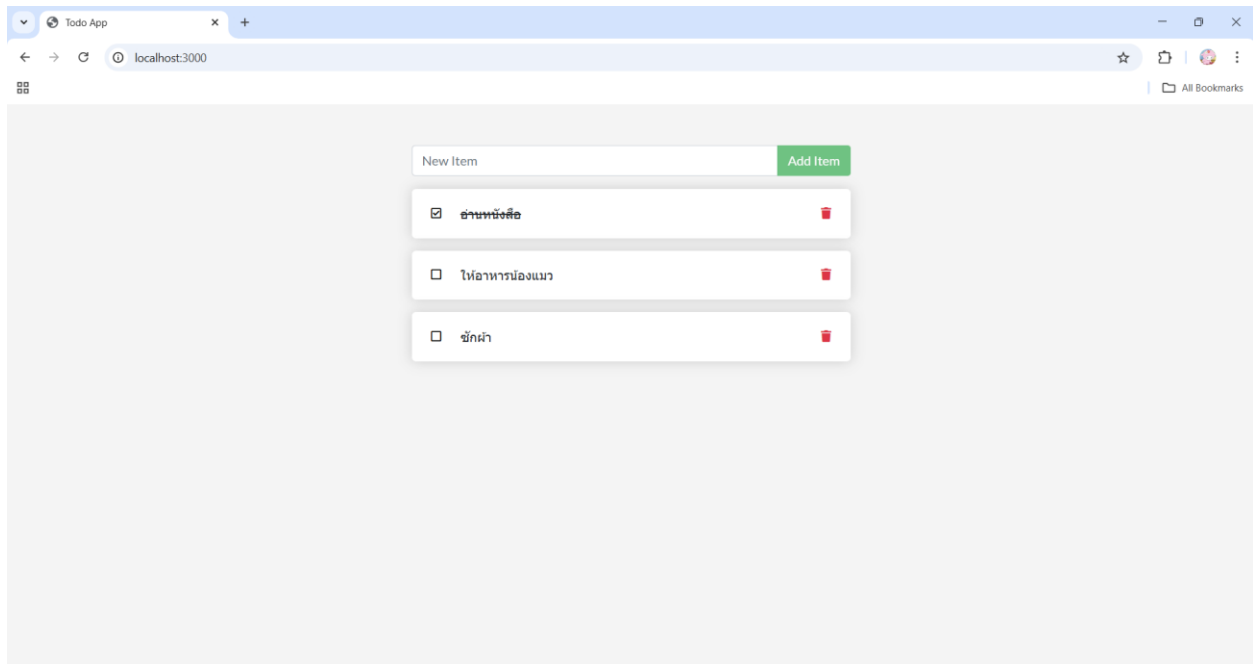
```
$ docker run -dp 3000:3000 <myapp_รหัสสนศ. ไม่มีขีด>
```

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

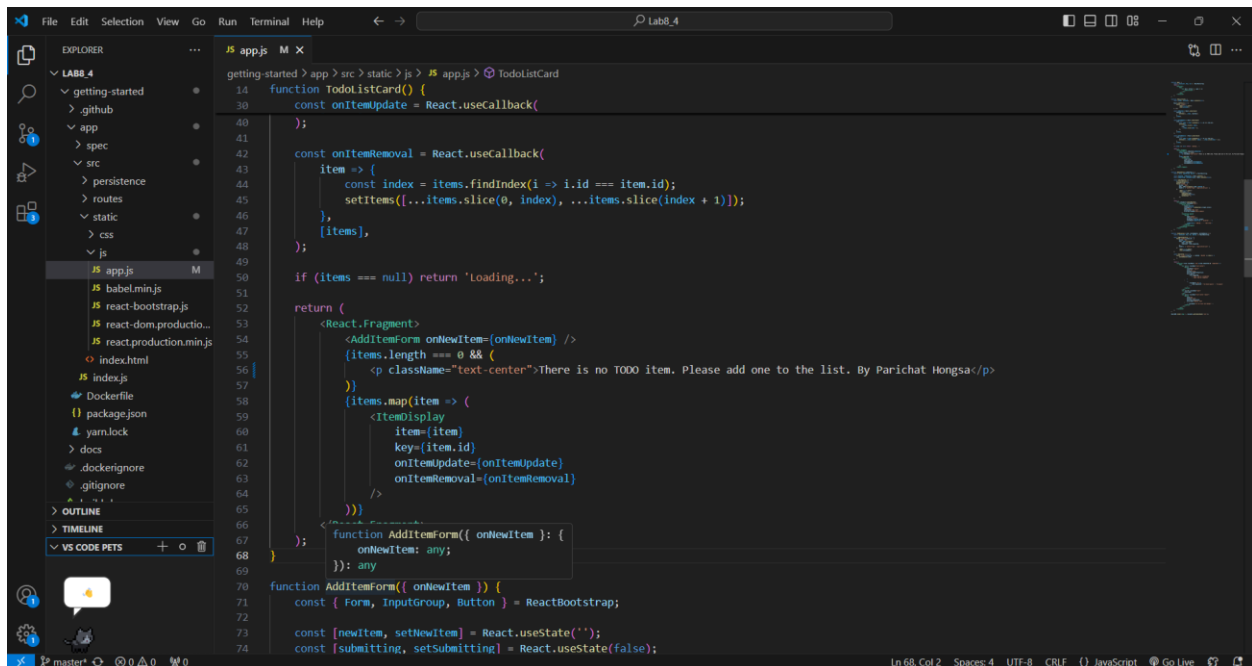
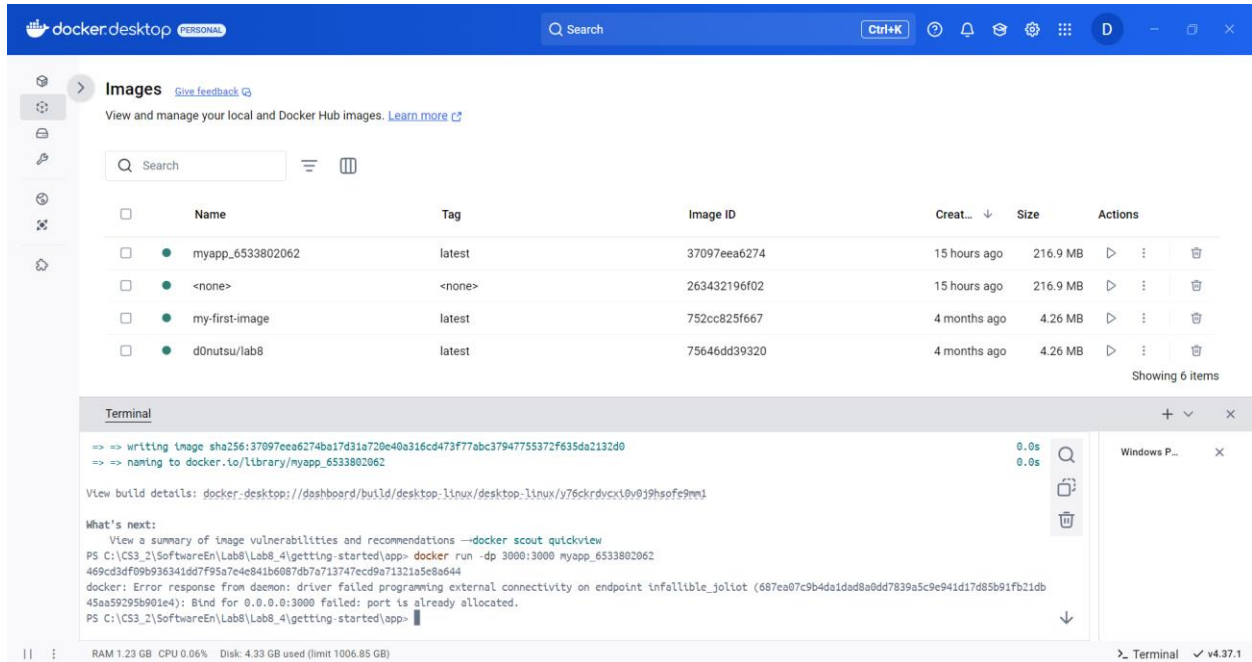
b. Save ไฟล์ให้เรียบร้อย

9. ทำการ Build Docker image โดยใช้คำสั่งเดียวกันกับข้อ 5

10. Start และรัน Container ตัวใหม่ โดยใช้คำสั่งเดียวกันกับข้อ 6

Lab Worksheet

[Check point#10] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงคำสั่งและผลลัพธ์ที่ได้ทางหน้าจอ พร้อมกับตอบคำถามต่อไปนี้



Lab Worksheet

(1) Error ที่เกิดขึ้นหมายความว่าอย่างไร และเกิดขึ้นเพราะอะไร

Docker ไม่สามารถเปิดใช้งาน container ที่พยายาม map port 3000 จาก host ไปยัง container ได้ เพราะ port 3000 บนเครื่อง host ถูกใช้งานอยู่แล้ว (port ถูกจองไว้) ปัญหานี้เกิดจากการจอง port ซ้ำ

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

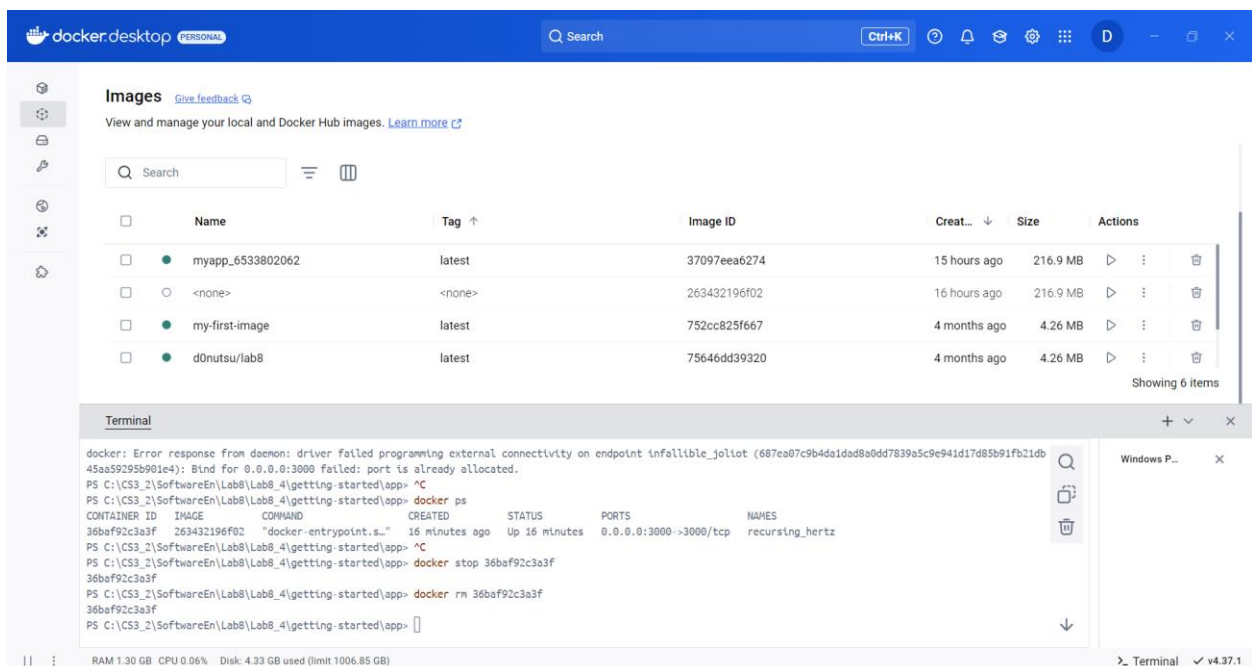
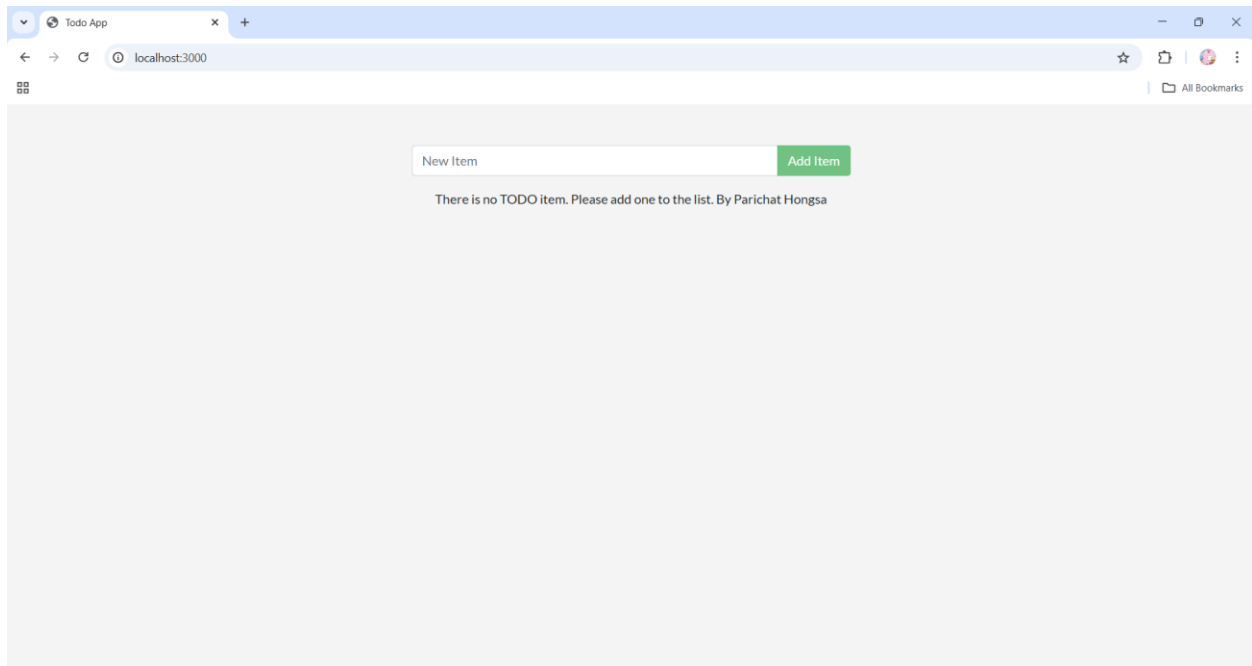
- i. ไปที่หน้าต่าง Containers
- ii. เลือกไอคอนถังขยะในแถวของ Container ที่ต้องการจะลบ
- iii. ยืนยันโดยการกด Delete forever

12. Start และรัน Container ตัวใหม่อีกครั้ง โดยใช้คำสั่งเดียวกันกับข้อ 6

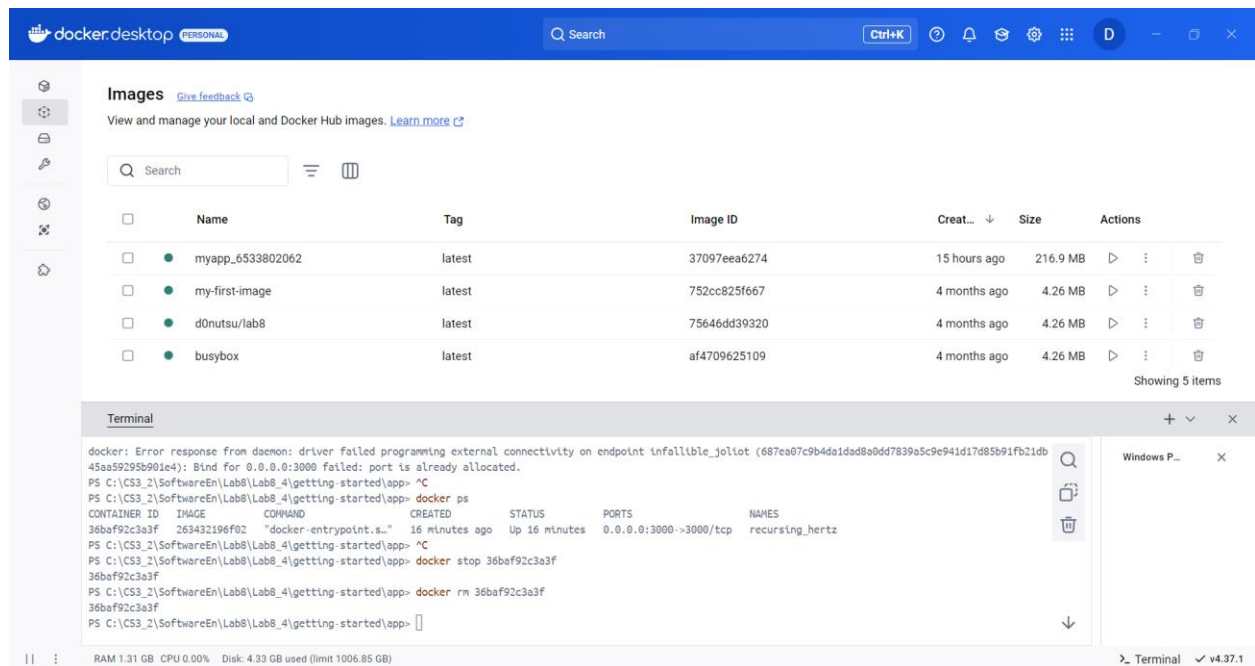
13. เปิด Browser ไปที่ URL = <http://localhost:3000>

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

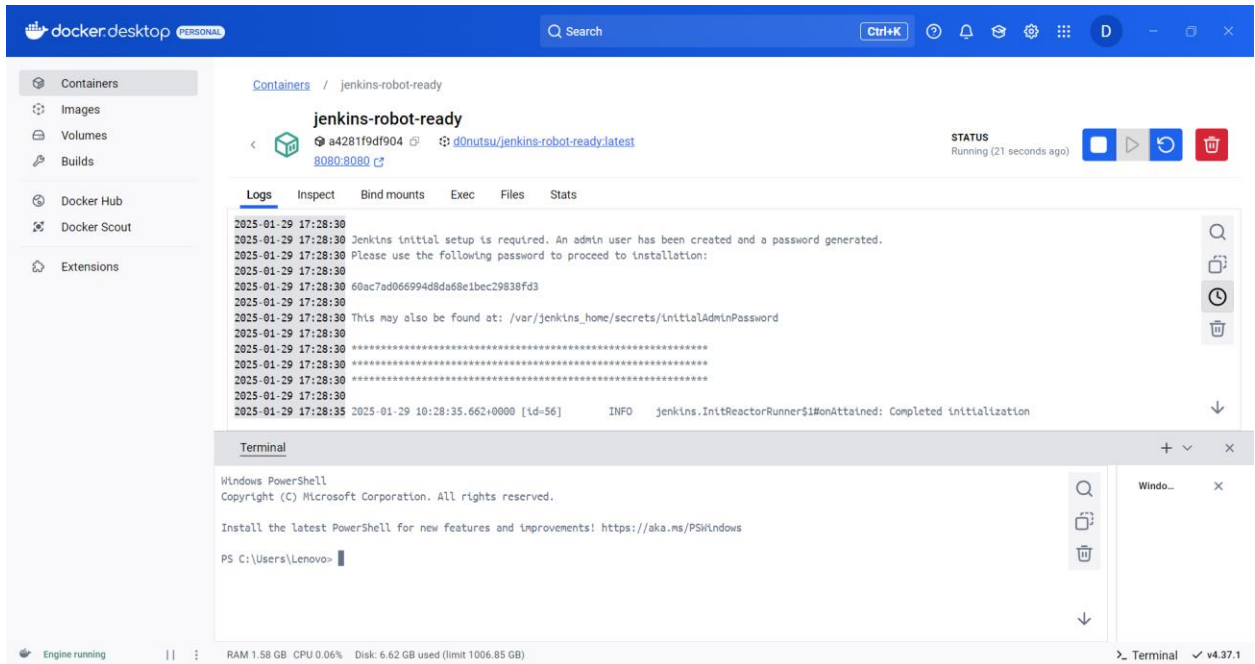
1. เปิด Command line หรือ Terminal บน Docker Desktop
2. ป้อนคำสั่งและทำการรัน container โดยผูกพอร์ต


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

 หรือ


```
$ docker run -p 8080:8080 -p 50000:50000 --restart=on-failure -v jenkins_home:/var/jenkins_home jenkins/jenkins:its-jdk17
```
3. บันทึกรหัสผ่านของ Admin user ไว้สำหรับ log-in ในครั้งแรก

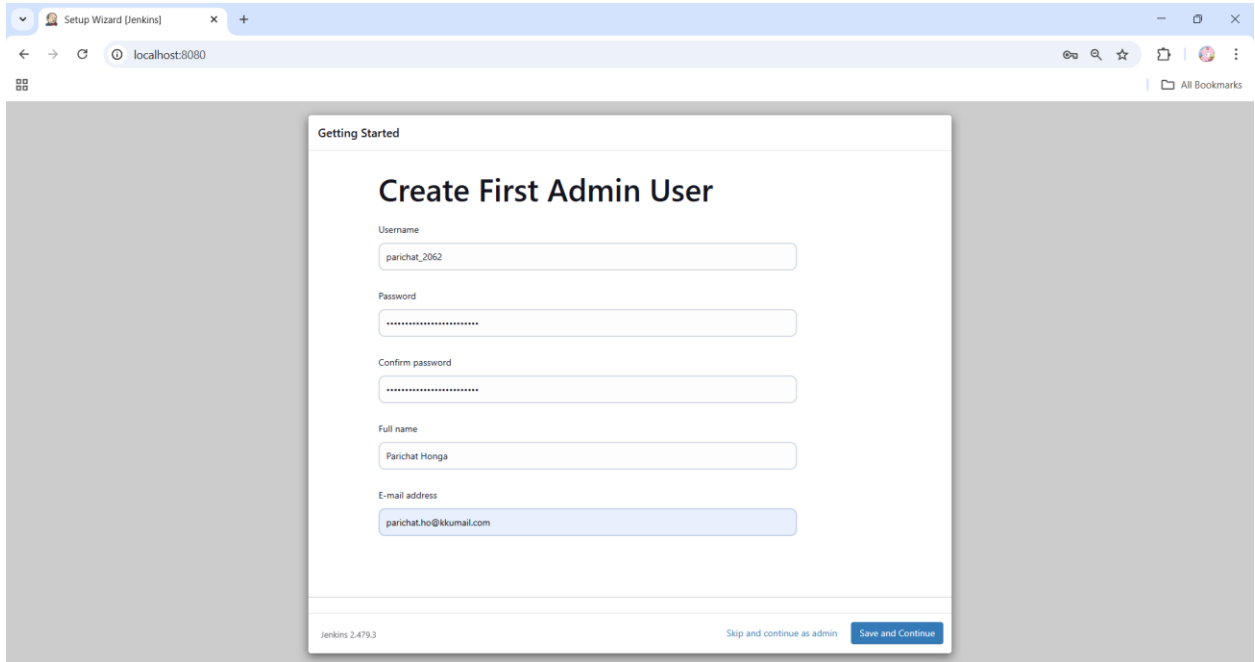
Lab Worksheet

[Check point#12] Capture หน้าจอที่แสดงผล Admin password

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

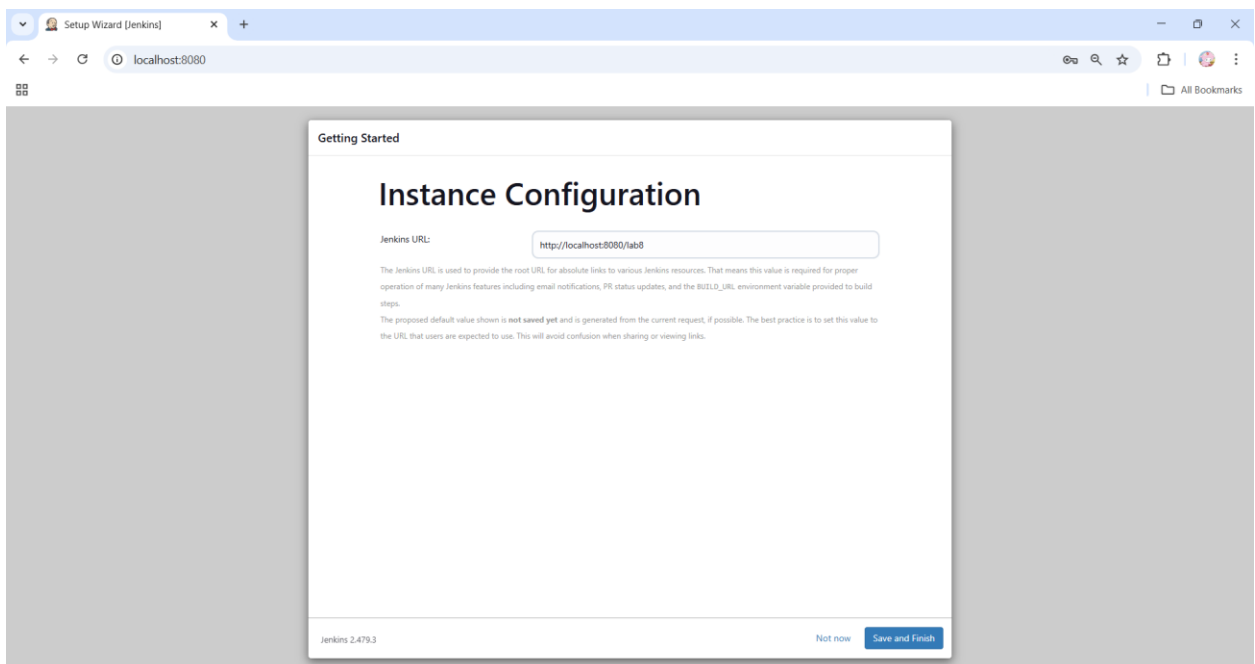
[Check point#13] Capture หน้าจอที่แสดงผลการตั้งค่า



The screenshot shows the Jenkins Setup Wizard interface in a web browser. The page title is "Getting Started" and the main heading is "Create First Admin User". The form contains the following fields:

- Username: parichat_2062
- Password: (masked with dots)
- Confirm password: (masked with dots)
- Full name: Parichat Honga
- E-mail address: parichat.ho@kkumail.com

At the bottom left, it says "Jenkins 2.479.3". At the bottom right, there are two buttons: "Skip and continue as admin" and "Save and Continue".



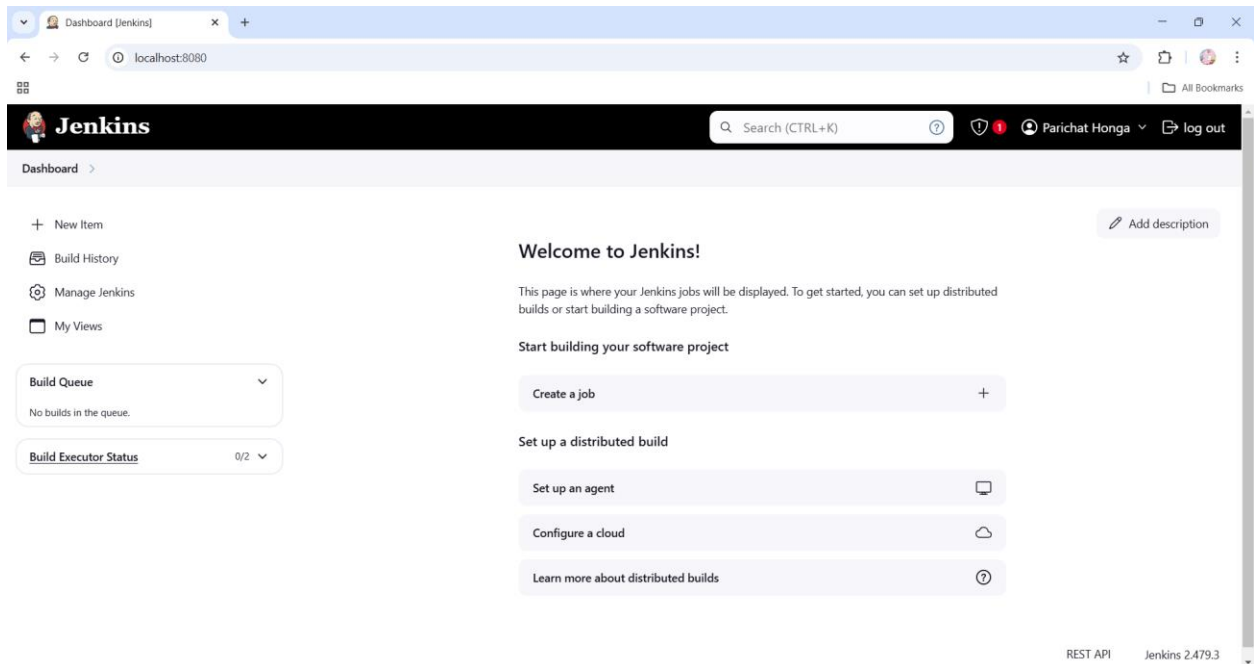
The screenshot shows the Jenkins Setup Wizard interface in a web browser. The page title is "Getting Started" and the main heading is "Instance Configuration". The form contains the following field:

- Jenkins URL: http://localhost:8080/lab8

Below the field, there is explanatory text: "The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD_URL environment variable provided to build steps. The proposed default value shown is not saved yet and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links."

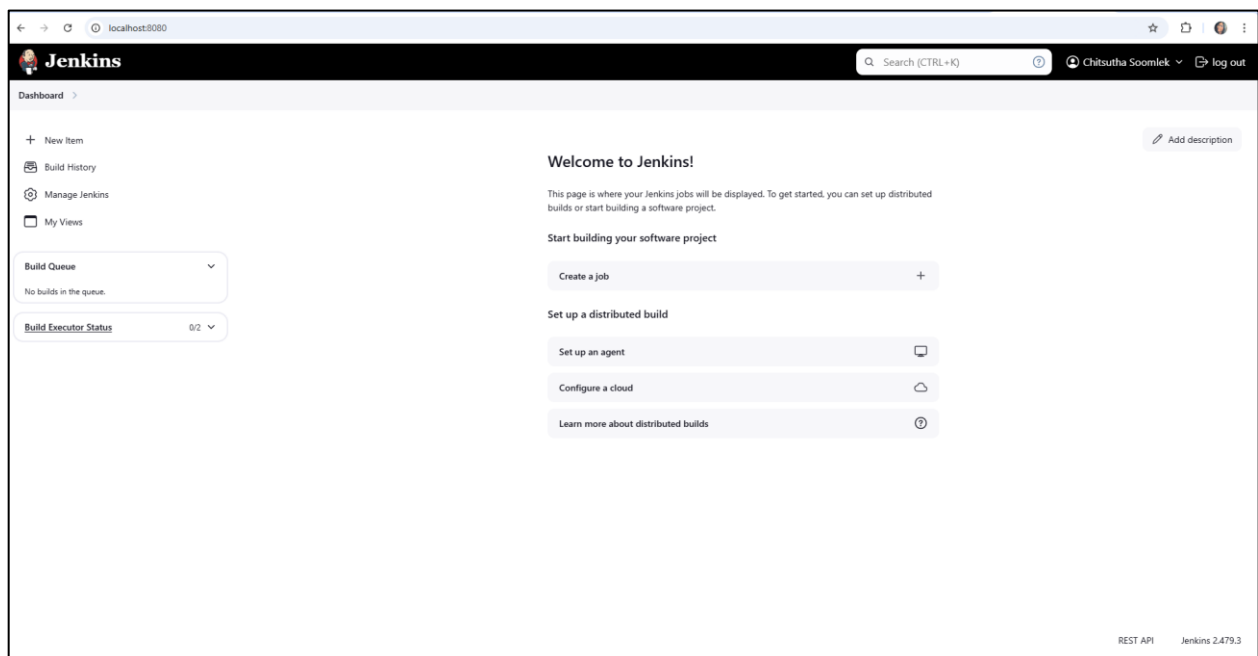
At the bottom left, it says "Jenkins 2.479.3". At the bottom right, there are two buttons: "Not now" and "Save and Finish".

Lab Worksheet



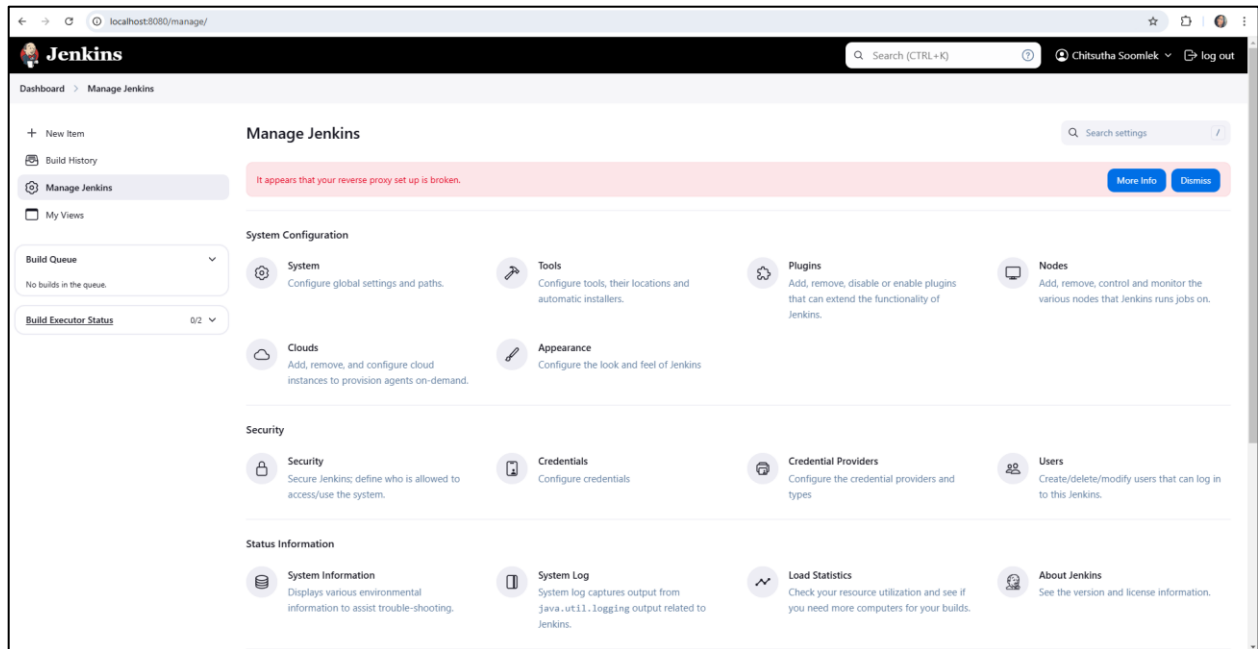
7. กำหนด Jenkins URL เป็น <http://localhost:8080/lab8>

8. เมื่อติดตั้งเรียบร้อยแล้วจะพบกันหน้า Dashboard ดังแสดงในภาพ

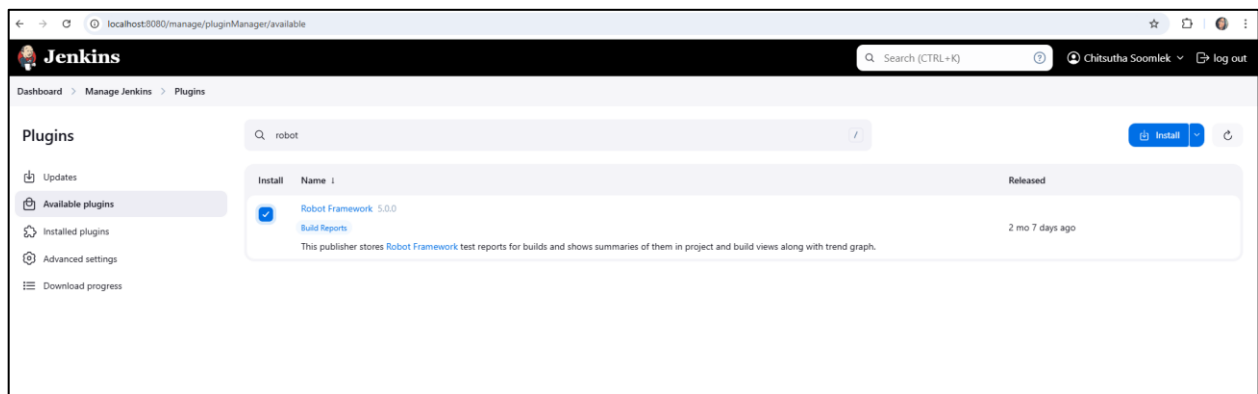


Lab Worksheet

9. เลือก Manage Jenkins แล้วไปที่เมนู Plugins

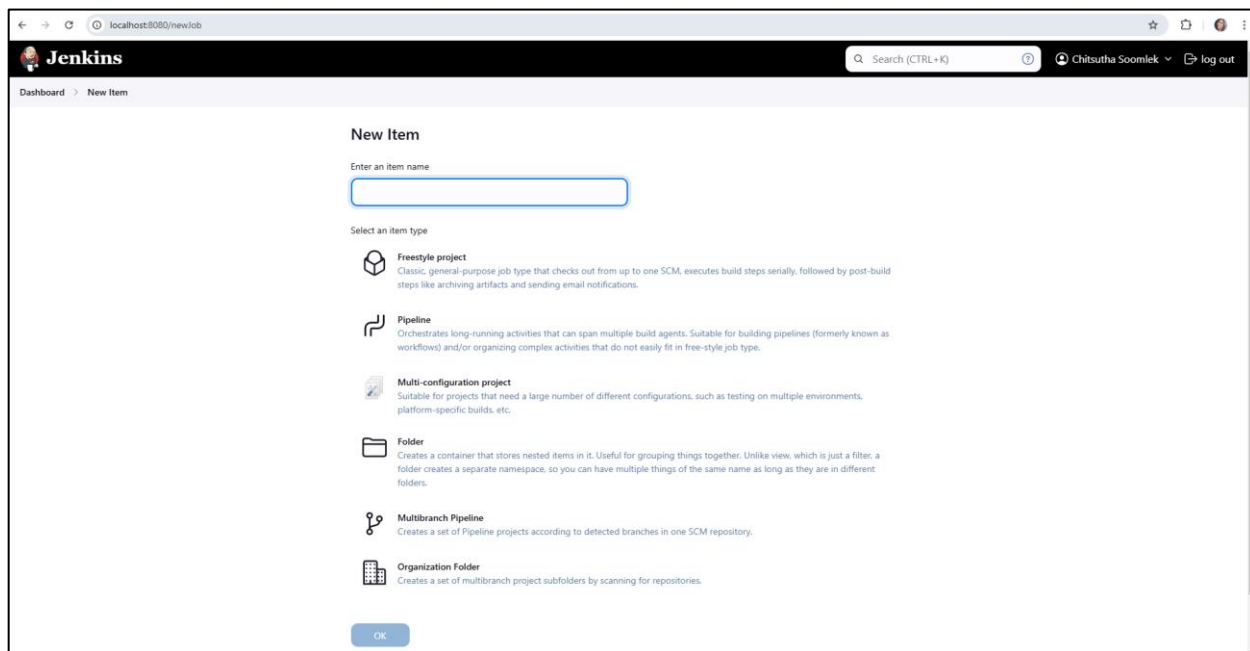


10. ไปที่เมนู Available plugins แล้วเลือกติดตั้ง Robotframework เพิ่มเติม



Lab Worksheet

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



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

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

robot complete.robot

Lab Worksheet

Post-build action: เพิ่ม Publish Robot Framework test results -> ระบุไดเรกทอรีที่เก็บไฟล์ผลการทดสอบโดย Robot framework ในรูป xml และ html -> ตั้งค่า Threshold เป็น % ของการทดสอบที่ไม่ผ่าน แล้วนับว่าซอฟต์แวร์มีปัญหา -> 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 shows the Jenkins Dashboard at localhost:8080. The main table lists the UAT pipeline with the following details:

S	W	Name	Last Success	Last Failure	Last Duration	Robot Results + Duration Trend
		UAT	N/A	1 min 26 sec	1.4 sec	

Additional information on the dashboard includes:

- Build Queue: No builds in the queue.
- Build Executor Status: 0/2
- REST API: Jenkins 2.479.3

The screenshot shows the Jenkins UAT pipeline details page. The latest robot results are as follows:

Latest Robot Results:					
Total	Failed	Passed	Skipped	Pass %	
All tests	2	2	0	0	0.0

Links provided for further details:

- Browse results
- Open report.html
- Open log.html

The "Robot Framework Tests Trend (all tests)" graph shows the number of test cases (0 to 2) over time, with a legend indicating Skipped (yellow), Passed (green), and Failed (red). The graph shows a single red bar for a failed test case.

Permalinks for the latest build (#2) are listed:

- Last build (#2), 1 min 16 sec ago
- Last failed build (#2), 1 min 16 sec ago
- Last unsuccessful build (#2), 1 min 16 sec ago
- Last completed build (#2), 1 min 16 sec ago

Lab Worksheet

Search (CTRL+K)

Parichat Honga
 log out

Dashboard > UAT > #2 > Console Output

Status

</> Changes

Console Output

Edit Build Information

Delete build '#2'

Timings

Git Build Data

Robot Results

<- Previous Build

Console Output

Download
 Copy
 View as plain text

```

Started by user Parichat Honga
Running as SYSTEM
Building in workspace /var/jenkins_home/workspace/UAT
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/ParichatHo/LabJenkins.git
> git init /var/jenkins_home/workspace/UAT # timeout=10
Fetching upstream changes from https://github.com/ParichatHo/LabJenkins.git
> git --version # timeout=10
> git --version # 'git version 2.39.5'
> git fetch --tags --force --progress -- https://github.com/ParichatHo/LabJenkins.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git config remote.origin.url https://github.com/ParichatHo/LabJenkins.git # timeout=10
> git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision 447371c43dccb454497db49d53b06b65ca7fe8bd (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f 447371c43dccb454497db49d53b06b65ca7fe8bd # timeout=10
Commit message: "Add complete.robot file to repo"
First time build. Skipping changelog.
[UAT] $ /bin/sh -xe /tmp/jenkins18063086560200954681.sh
+ robot complete.robot
[ ERROR ] Error in file '/var/jenkins_home/workspace/UAT/complete.robot' on line 2: Importing library 'SeleniumLibrary' failed:
ModuleNotFoundError: No module named 'SeleniumLibrary'
Traceback (most recent call last):
  None
PYTHONPATH:
  /usr/local/bin
  /usr/lib/python3.11.zip
  /usr/lib/python3.11
  /usr/lib/python3.11/lib-dynload
  /usr/local/lib/python3.11/dist-packages
  /usr/lib/python3/dist-packages
=====
Complete
=====
Open Browser To Form Page | FAIL |
Evaluating expression "sys.modules['selenium.webdriver'].ChromeOptions()" failed: KeyError: 'selenium.webdriver'
-----
REGISTER SUCCESS | FAIL |
No keyword with name 'Input Text' found.

Also teardown failed:
No keyword with name 'Close Browser' found.
-----
Complete | FAIL |
2 tests, 0 passed, 2 failed
=====
Output: /var/jenkins_home/workspace/UAT/output.xml
Log: /var/jenkins_home/workspace/UAT/log.html
Report: /var/jenkins_home/workspace/UAT/report.html
Build step 'Execute shell' marked build as failure
Robot results publisher started...
INFO: Checking test criticality is deprecated and will be dropped in a future release!
-Parsing output xml:
Done!
-Copying log files to build dir:
Done!
-Assigning results to build:
Done!
-Checking thresholds:
Done!
Done publishing Robot results.
Finished: FAILURE

```

REST API Jenkins 2.479.3

29

Lab Worksheet

The screenshot shows the Jenkins configuration page for a new job named "Lab 8.5". The page is divided into several sections: General, Source Code Management, Build Triggers, Build Environment, Build Steps, and Post-build Actions.

General

- Description: Lab 8.5
- Plan tool: [Produce](#)
- ☐ Discard old builds
- ☒ GitHub project
 - Project url: <https://github.com/PatchaiTubleringit/>
 - Advanced
- ☐ This project is parameterized
- ☐ Throttle builds
- ☐ Exclude concurrent builds if necessary
- Advanced

Source Code Management

- ☐ None
- ☒ Git
 - Repositories
 - Repository URL: <https://github.com/PatchaiTubleringit/>
 - Credentials: none
 - Add
 - Advanced
 - Add Repository
 - Branches to build
 - Branch Specifier (blank for 'any'): */main
 - Add Branch
 - Repository browser: (Auto)
 - Additional Behaviours: Add

Build Triggers

- ☐ Trigger builds remotely (e.g., from scripts)
- ☐ Build after other projects are built
- ☒ Build periodically
 - Schedule: H/15 * * * *
 - What last time run at: Wednesday, January 29, 2025 at 10:50:00 AM Coordinated Universal Time; would next run at: Wednesday, January 29, 2025 at 11:05:00 AM 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

- Execute shell**
 - Command: robot -c config.txt robot
 - Advanced
- Add build step

Post-build Actions

- Publish Robot Framework test results**
 - Directory of Robot output: Path to directory containing robot and html files (relative to build workspace)
 - Advanced
 - Follow
 - Thresholds for build result:
 - 0% (Yellow)
 - 200
 - 0% (Green)
 - 800
 - ☒ SUPPRESSED THIS FLAG DOES NOTHING - Use thresholds for critical tests only
 - ☐ Include skipped tests in total count for thresholds
- Add post-build action

Buttons: Save, Apply

Footer: REST API Jenkins 2.479.3