

Lab Worksheet

ชื่อ-นามสกุล นายภูมิพิพัฒน์ มะลิมาตร์

รหัสนักศึกษา 653380277-9

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

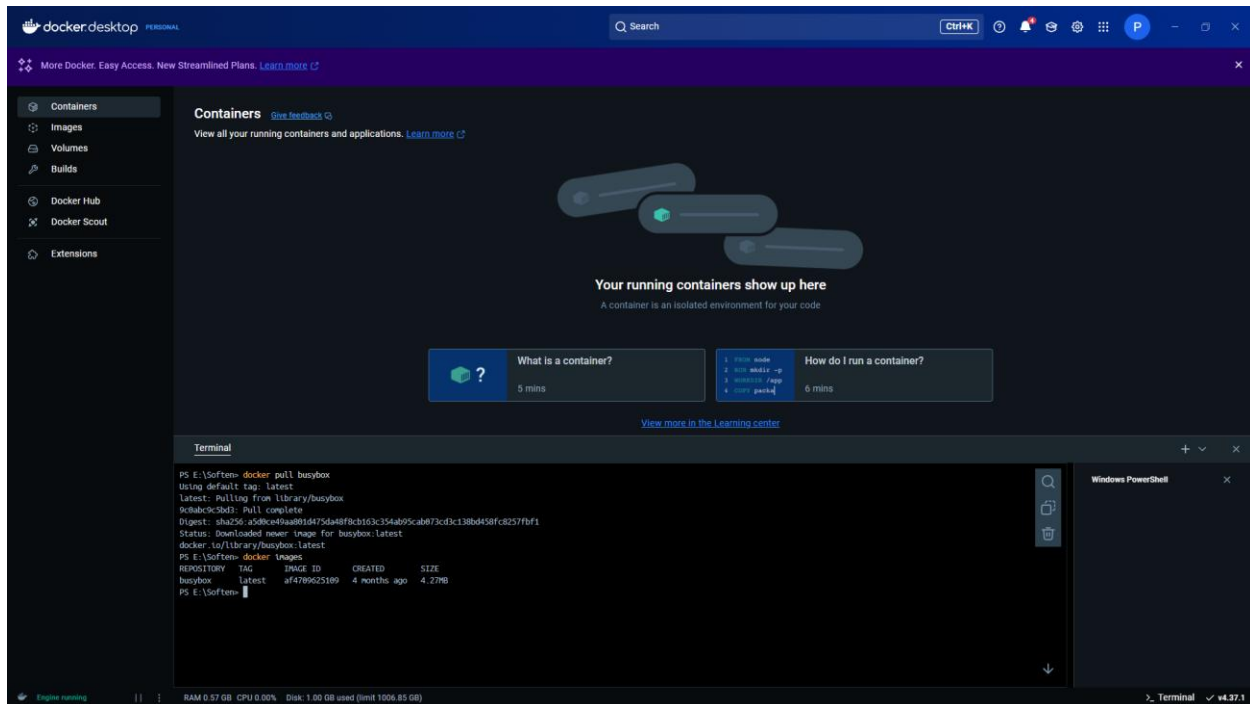
1. ติดตั้ง Docker desktop ลงบนเครื่องคอมพิวเตอร์ โดยดาวน์โหลดจาก <https://www.docker.com/get-started>
2. สร้าง Account บน Docker hub (<https://hub.docker.com/signup>)
3. กำหนดให้ \$ หมายถึง Command prompt และ <> หมายถึง ให้ป้อนค่าของพารามิเตอร์ที่กำหนด

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

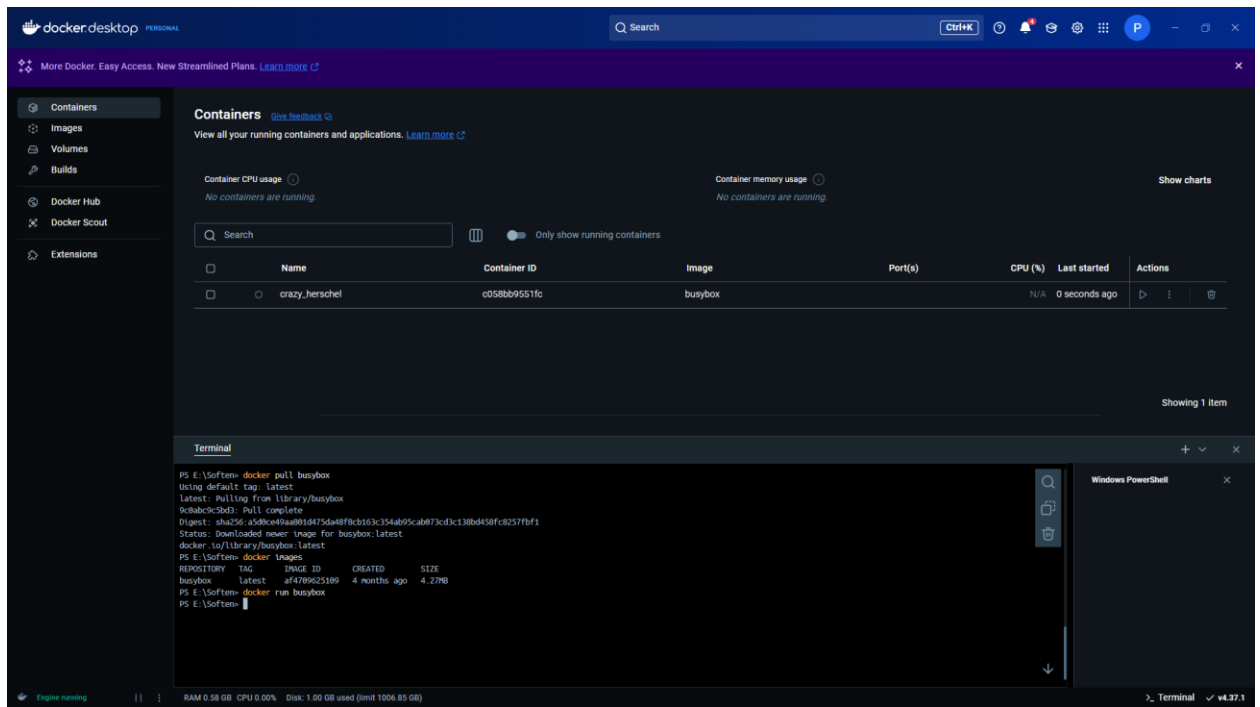
[Check point#1] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงผลลัพธ์ที่ได้ พร้อมกับตอบคำถามต่อไปนี้



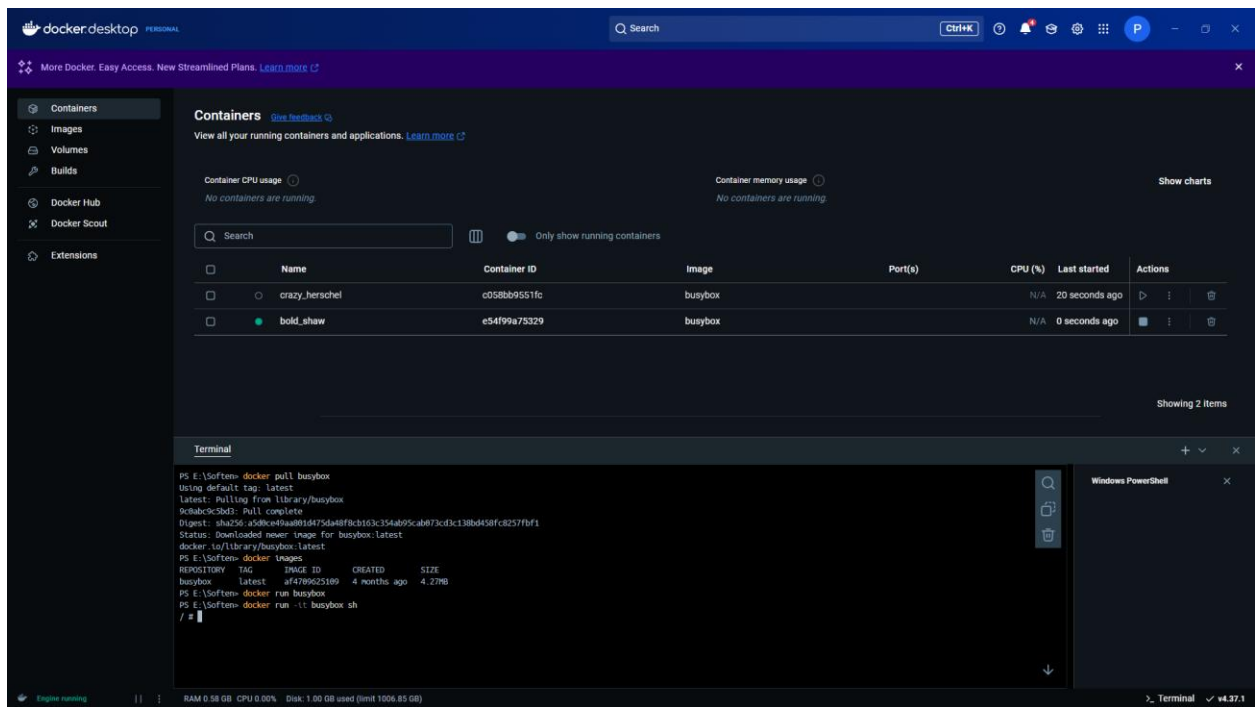
- (1) สิ่งที่อยู่ภายใต้คอนเทนเนอร์ Repository คืออะไร ไว้บ่งบอกถึงชื่อของ docker image จากภาพคือ repository มีชื่อว่า busybox
- (2) Tag ที่ใช้บ่งบอกถึงอะไร เป็นตัวบ่งบอกอัตลักษณ์และลักษณะของ docker image แต่ละตัวเพื่อจำแนกกรณีชื่อ image ได้ Repository ชื่อ จากภาพ tag คือ latest

5. ป้อนคำสั่ง \$ docker run busybox

Lab Worksheet

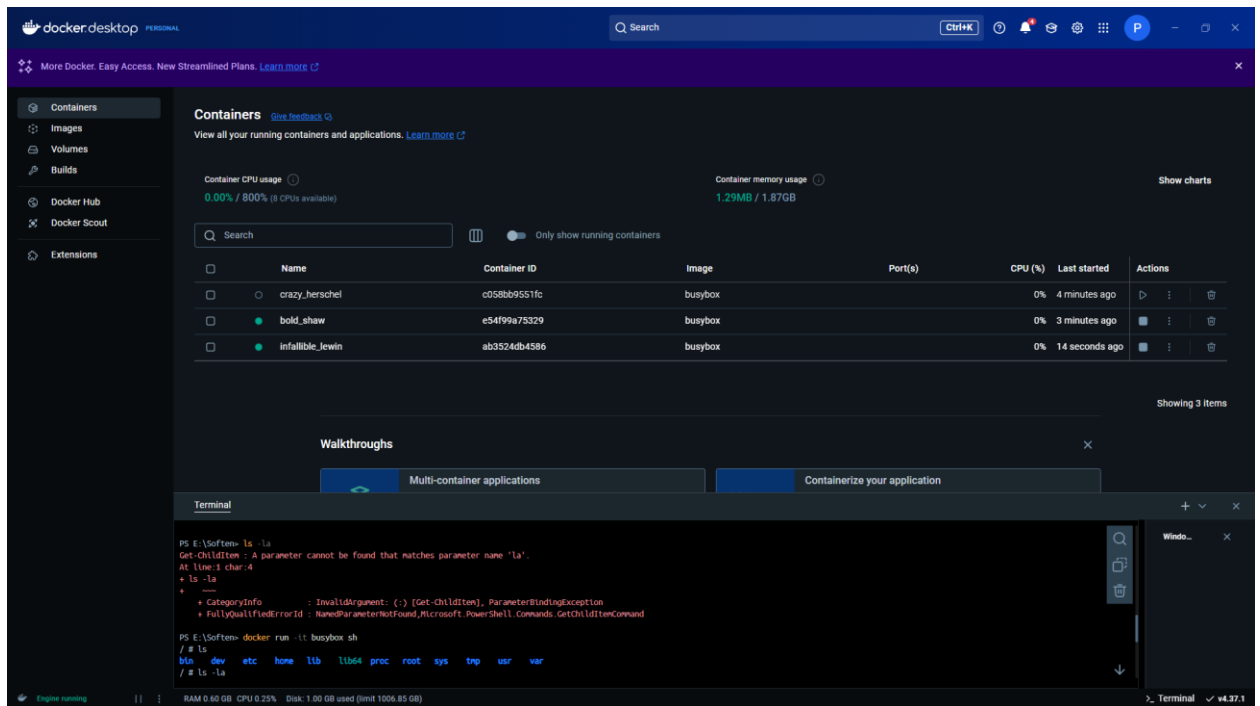


6. ป้อนคำสั่ง \$ docker run -it busybox sh

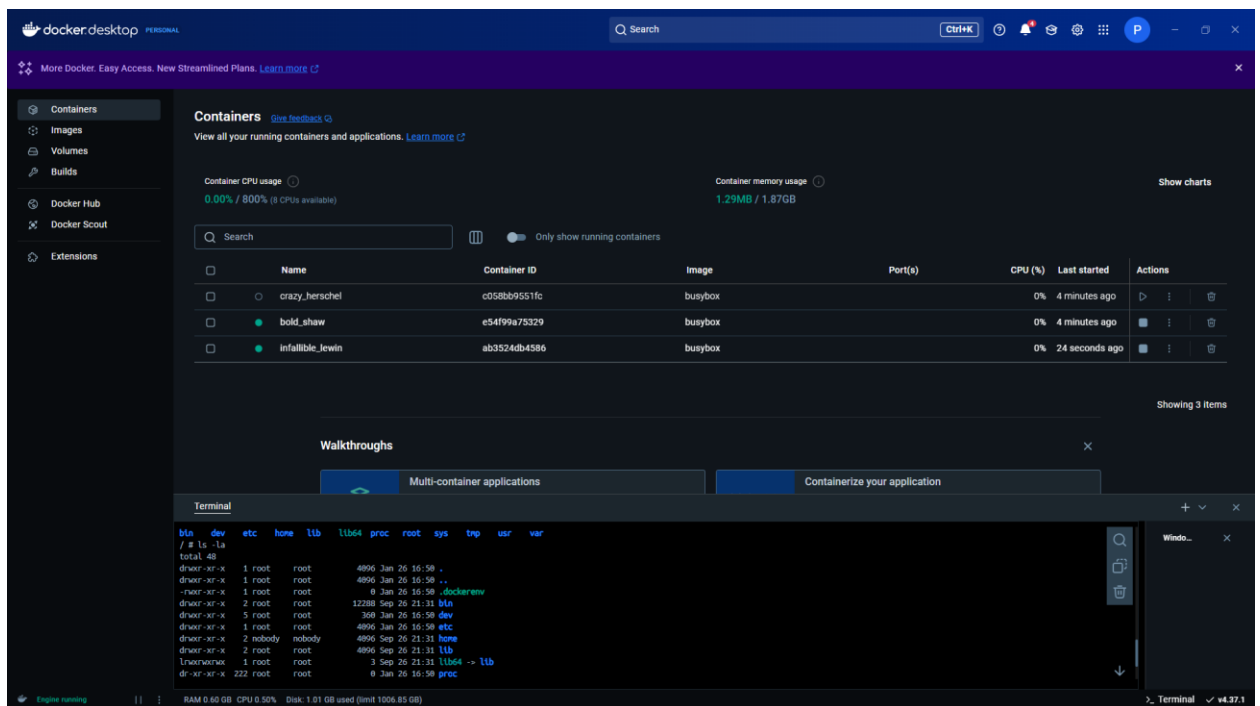


Lab Worksheet

7. ป้อนคำสั่ง ls

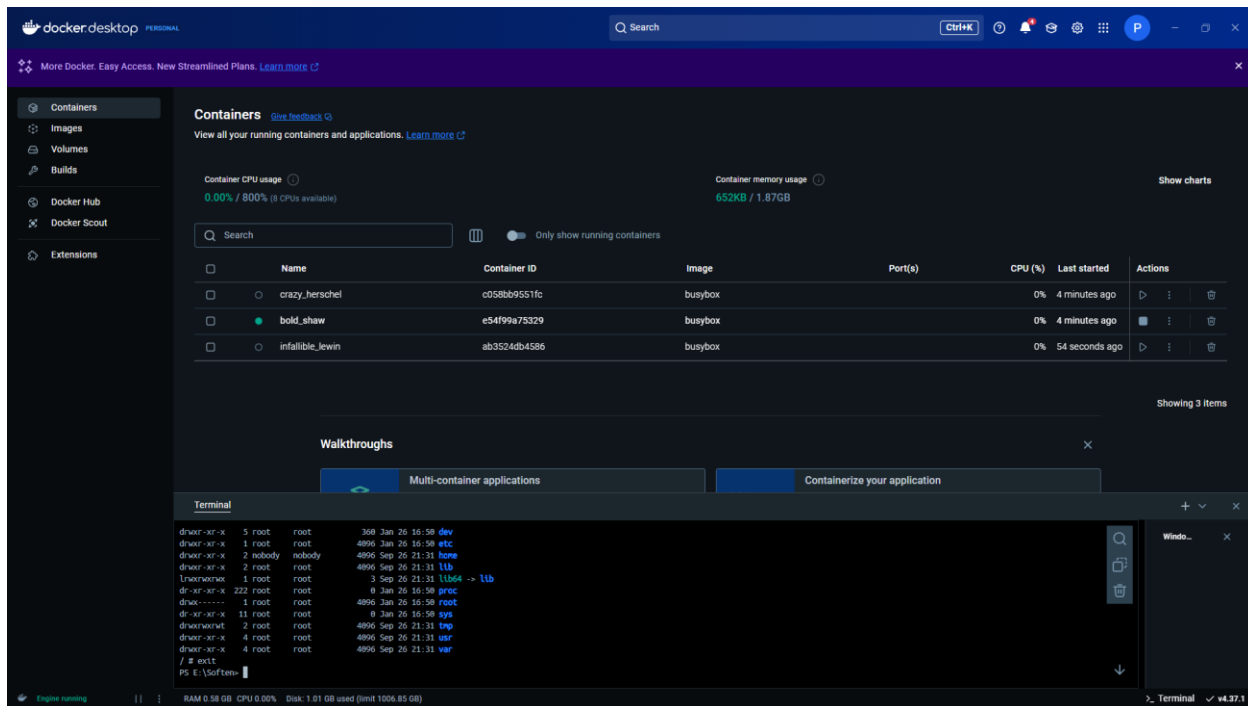
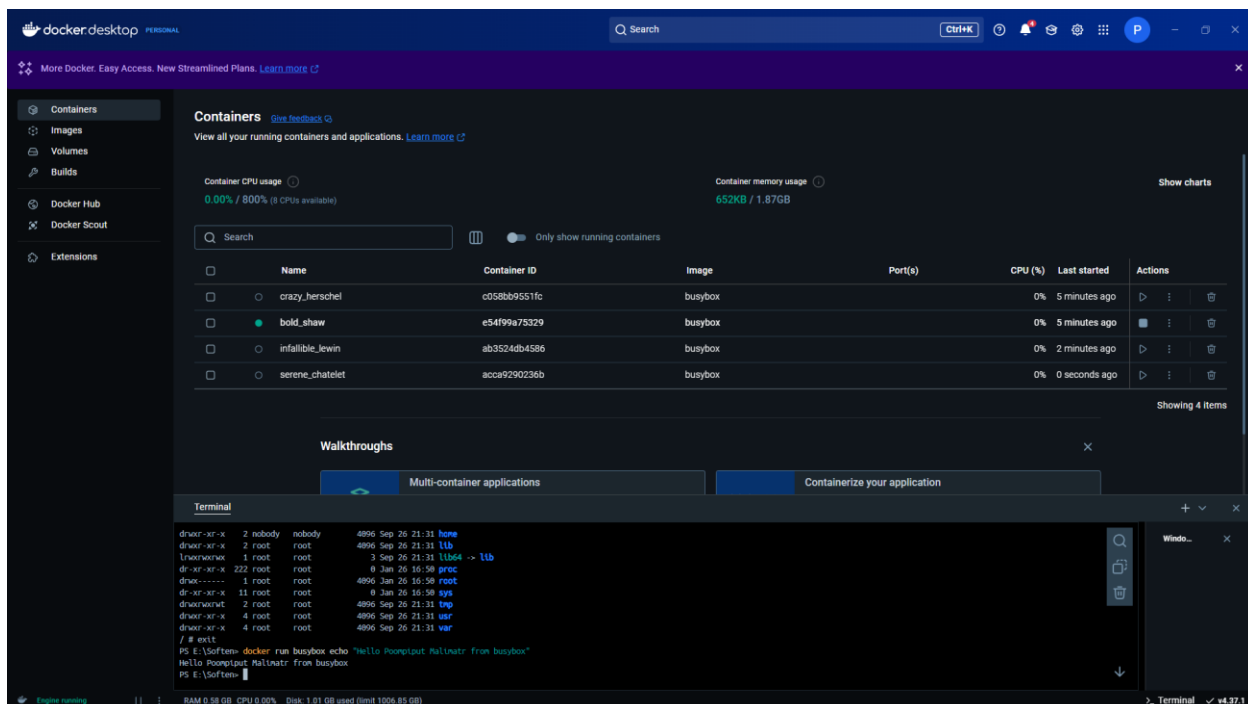


8. ป้อนคำสั่ง ls -la



Lab Worksheet

9. ป้อนคำสั่ง exit

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

Lab Worksheet

11. ป้อนคำสั่ง \$ docker ps -a

The screenshot shows the Docker Desktop interface. The 'Containers' tab is active, displaying a list of running containers. The CPU usage is 0.00% / 800% (8 CPUs available) and memory usage is 652KB / 1.87GB. A search bar and a toggle for 'Only show running containers' are present. Below the search bar is a table of containers.

	Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
<input type="checkbox"/>	crazy_herschel	c058bb9551fc	busybox		0%	6 minutes ago	🔍 🔧 🗑️
<input checked="" type="checkbox"/>	bold_shaw	e54f99a75329	busybox		0%	5 minutes ago	🔍 🔧 🗑️
<input type="checkbox"/>	infallible_lewin	ab352adb4586	busybox		0%	2 minutes ago	🔍 🔧 🗑️
<input type="checkbox"/>	serene_chatelet	acca9290236b	busybox		0%	28 seconds ago	🔍 🔧 🗑️

Showing 4 items

Below the container list, there are 'Walkthroughs' for 'Multi-container applications' and 'Containerize your application'.

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

```

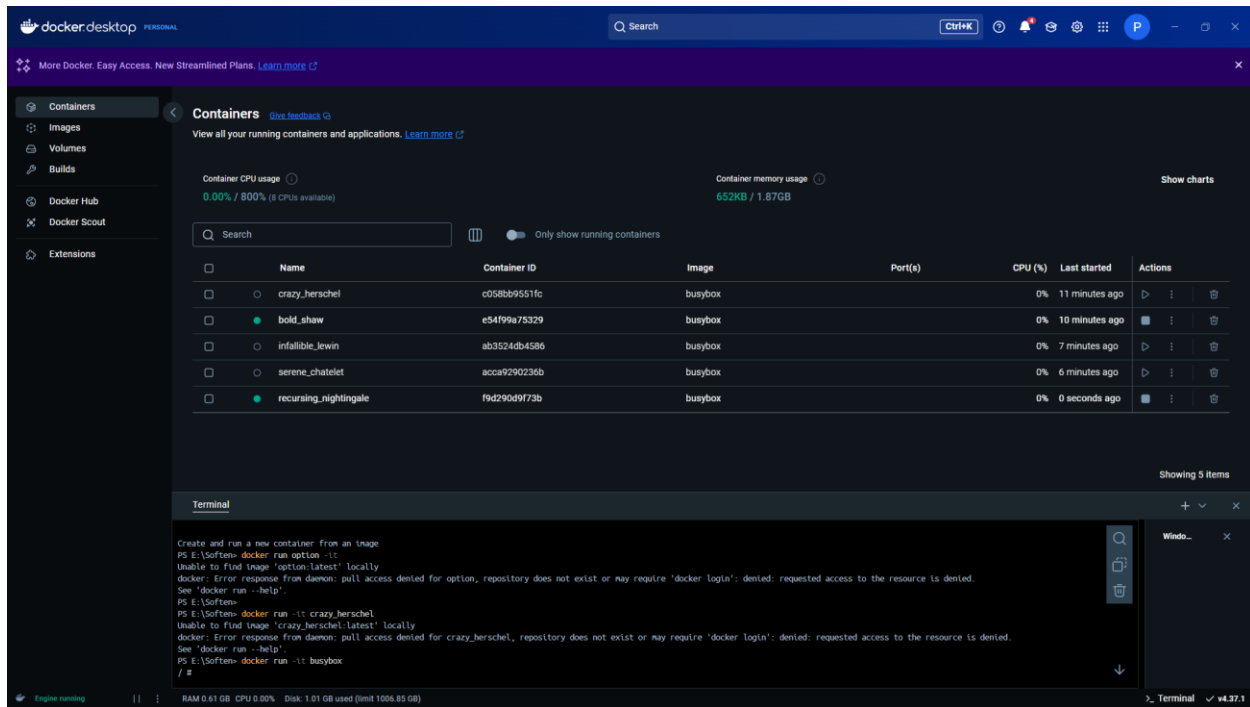
dnw@winut: ~$ docker run busybox echo "Hello Poorniput Mallinatr from busybox"
Hello Poorniput Mallinatr from busybox
PS E:\Softw> docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS   NAMES
acca9290236b   busybox   "echo 'Hello Poornip..." 27 seconds ago  Exited (0) 27 seconds ago           serene_chatelet
ab352adb4586   busybox   "sh"                    2 minutes ago  Exited (0) about a minute ago       infallible_lewin
e54f99a75329   busybox   "sh"                    5 minutes ago  Up 5 minutes                               bold_shaw
c058bb9551fc   busybox   "sh"                    5 minutes ago  Exited (0) 5 minutes ago              crazy_herschel
PS E:\Softw>

```

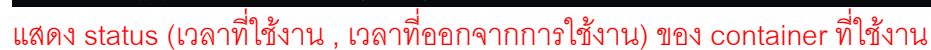
Lab Worksheet

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

(1) เมื่อใช้ option -it ในคำสั่ง run ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป



ทำการสร้างและรัน container ใหม่ จาก image ที่ได้ระบุขึ้นมาใหม่



The screenshot displays the Docker Desktop application window. The top navigation bar includes the Docker logo, user profile 'PERSONAL', a search bar, system tray icons (Ctrl+K, power, notifications, settings), and a window control button.

The main sidebar on the left contains navigation options: Containers (selected), Images, Volumes, Builds, Docker Hub, Docker Scout, and Extensions.

The central area is titled 'Containers' with a 'Give feedback' link. It states 'View all your running containers and applications.' Below this are two summary cards:

- Container CPU usage**: 0.00% / 800% (8 CPUs available)
- Container memory usage**: 652KB / 1.87GB

A 'Show charts' link is located to the right of the memory card.

Below the summary cards is a search bar and a toggle switch labeled 'Only show running containers'. A table lists four running containers:

	Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
<input checked="" type="checkbox"/>	bold_shaw	e54f99a75329	busybox		0%	15 minutes ago	[Stop] [Refresh] [Delete]
<input type="checkbox"/>	infallible_lewin	ab3524db4586	busybox		0%	12 minutes ago	[Play] [Refresh] [Delete]
<input type="checkbox"/>	serene_chatelet	ac0c9290236b	busybox		0%	10 minutes ago	[Play] [Refresh] [Delete]
<input type="checkbox"/>	recurring_nightingale	f9d290d9f73b	busybox		0%	5 minutes ago	[Play] [Refresh] [Delete]

'Showing 4 items' is displayed at the bottom right of the table.

At the bottom, there is a 'Terminal' pane showing command-line output from a shell inside the 'bold_shaw' container:

```
e54f99a75329 busybox "sh" 13 minutes ago Up 13 minutes bold_shaw
c858bb9551fc busybox "sh" 13 minutes ago Exit(0) 13 minutes ago crazy_herschel
PS E:\Softmon-docker> rm -rf c858bb9551fc-408be7694a664f598bc445c3dcda3b5c3ecff9edcaf664739e3
At line:1 char:11
+ ~~~~~
+ docker rm -rf c858bb9551fc-408be7694a664f598bc445c3dcda3b5c3ecff9edcaf664739e3
+ ~~~~~
The "+" operator is reserved for future use.
+ CategoryInfo          : Parameter ((:)) [], ParentContainsErrorRecordException
+ FullyQualifiedErrorId : RedirectNotSupported

PS E:\Softmon-docker> rm -rf c858bb9551fc-408be7694a664f598bc445c3dcda3b5c3ecff9edcaf664739e3
c858bb9551fc-408be7694a664f598bc445c3dcda3b5c3ecff9edcaf664739e3
PS E:\Softmon-
```

The status bar at the very bottom shows system metrics: RAM 6.60 GB, CPU 0.13%, Disk 1.01 GB used (limit 1006.85 GB).

Lab Worksheet

[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"
```

สำหรับเครื่องที่ใช้ระบบปฏิบัติการ MacOS หรือ Linux บนหน้าต่าง Terminal และบันทึกคำสั่งต่อไปนี้

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

```
FROM busybox
```

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

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

```
EOF
```

หรือใช้คำสั่ง

```
$ touch Dockerfile
```

แล้วใช้ Text Editor ในการใส่เนื้อหาแทน

5. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้

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

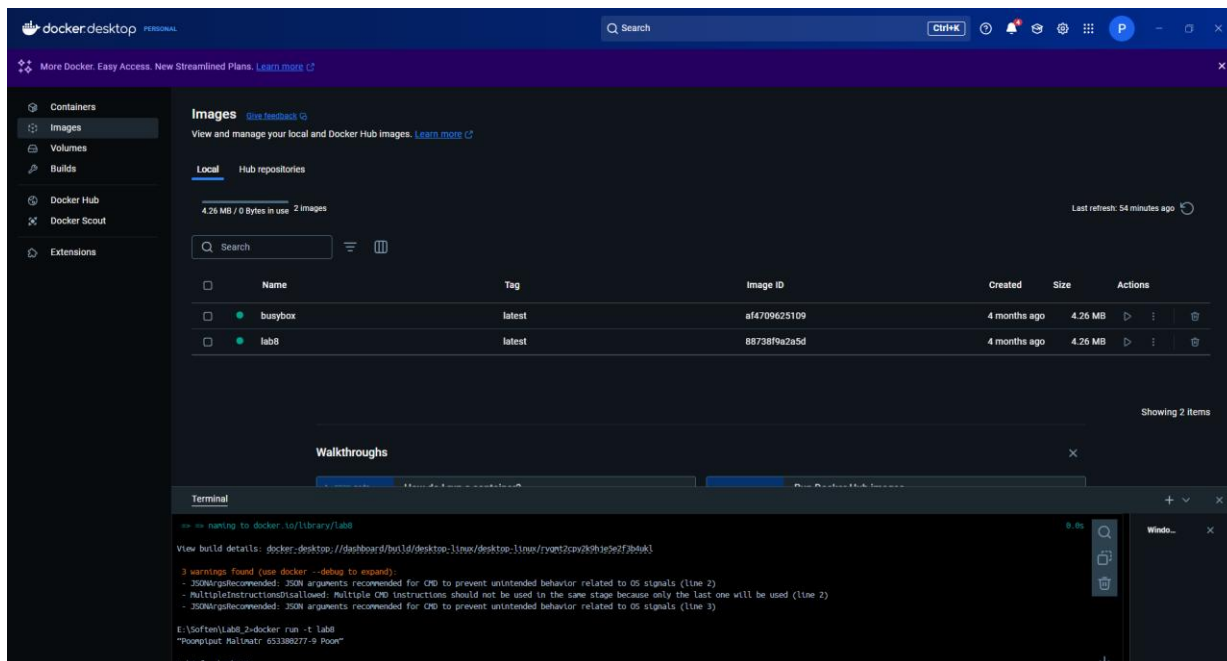
6. เมื่อ Build สำเร็จแล้ว ให้ทำการรัน Docker image ที่สร้างขึ้นในขั้นตอนที่ 5

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

พร้อมกับตอบคำถามต่อไปนี้

Lab Worksheet

(1) คำสั่งที่ใช้ในการ run คือ



Docker run -t lab8

(2) Option -t ในคำสั่ง \$ docker build ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป
เป็นตัวกำหนด tag หรือก็คือเป็นตัวกำหนดชื่อและ tag ของ build ที่สร้างขึ้น

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"

Lab Worksheet

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

The screenshot shows the Docker Desktop interface. On the left, the 'Containers' tab is selected, displaying a list of running containers. The main area shows a table of containers with columns for Name, Container ID, Image, Port(s), CPU (%), Last started, and Actions. Below the table, a 'Terminal' window is open, showing the output of the 'docker build' and 'docker run' commands. The terminal output indicates that the image was built successfully and is now running as a container named 'lab8'.

Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
bold_shaw	e54f99a75329	busybox		0%	51 minutes ago	[Stop] [Restart] [Refresh]
infallible_lewin	ab3224db4586	busybox		0%	48 minutes ago	[Stop] [Restart] [Refresh]
serene_chalet	acca9290236b	busybox		0%	46 minutes ago	[Stop] [Restart] [Refresh]
recurring_nightingale	f9d290d9f73b	busybox		0%	41 minutes ago	[Stop] [Restart] [Refresh]
crazy_galileo	9a1147d8807e	lab8		0%	13 minutes ago	[Stop] [Restart] [Refresh]
adoring_goodall	4f4a7be82441	poompiputn/lab8		0%	11 seconds ago	[Stop] [Restart] [Refresh]

Terminal Output:

```

=> exporting layers
=> writing image sha256:c3ae1ff999f28a5786f01624cf374c73b0b2456a3969964ac0b4a324bc
=> naming to docker.io/poompiputn/lab8

View build details: docker-desktop://dashboard/build/desktop.linux/desktop.linux/17f9a189999e9ec512115

2 warnings found (use docker --debug to expand):
- 350wArgsRecommended: 350w 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)
- 350wArgsRecommended: 350w arguments recommended for CMD to prevent unintended behavior related to OS signals (line 3)
PS E:\Soften\lab8_> docker run poompiputn/lab8
"pompiputn Kallnatr 653388277-9"
PS E:\Soften\lab8_>

```

Lab Worksheet

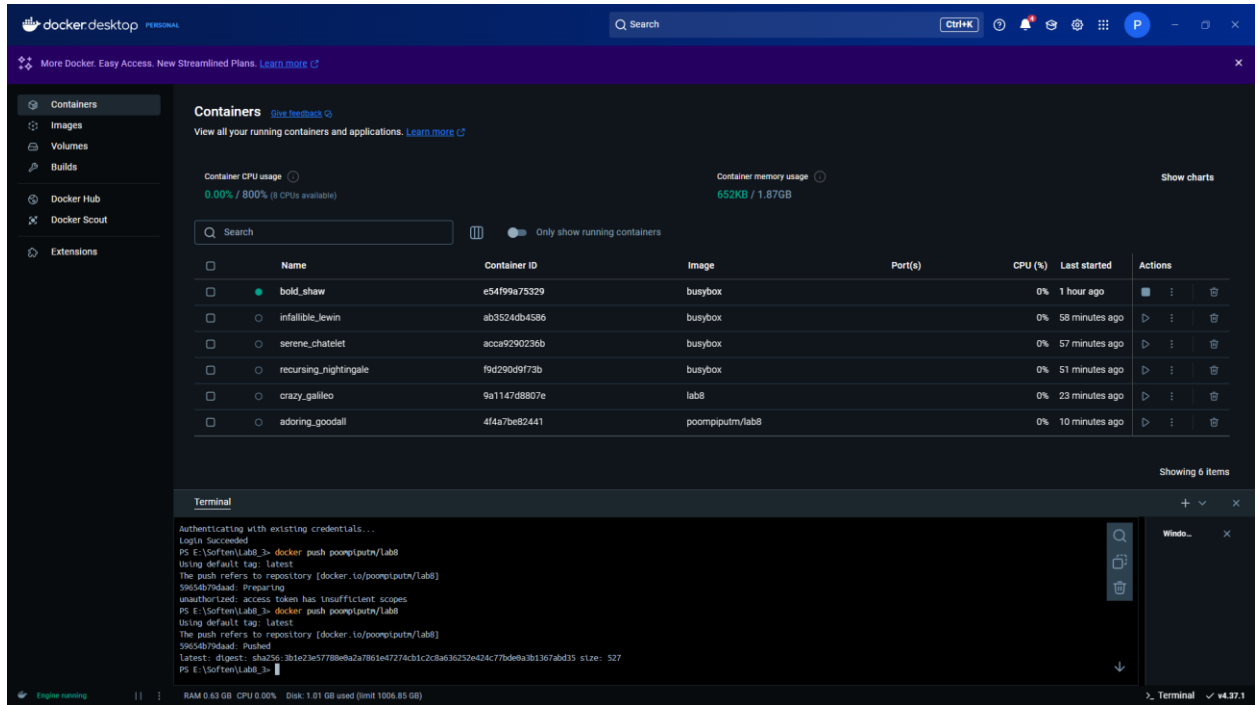
6. ทำการ Push ตัว Docker image ไปไว้บน Docker Hub โดยการใช้คำสั่ง

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

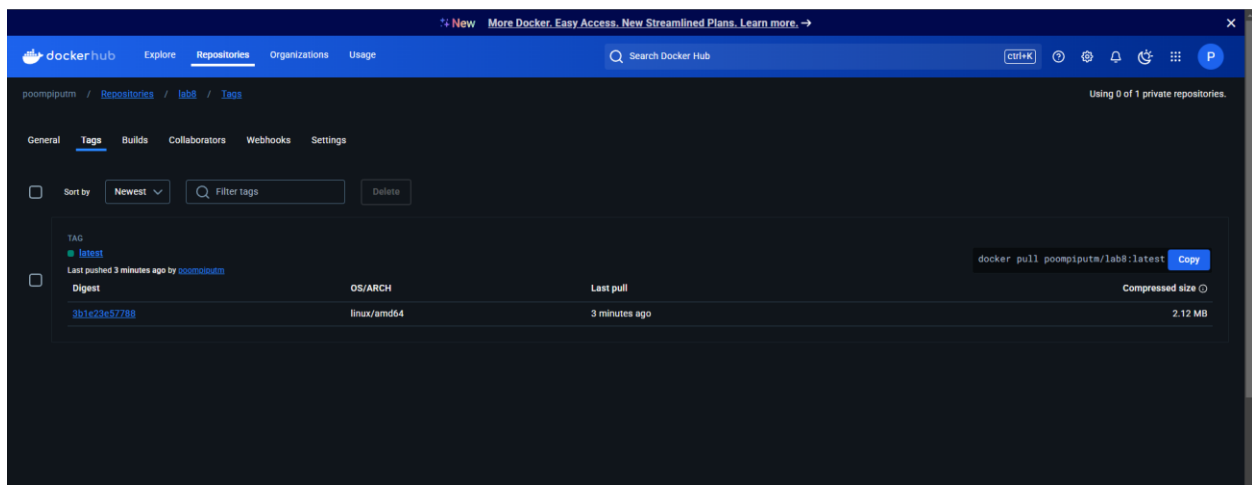
ในกรณีที่ติดปัญหาไม่ได้ Login ไว้ก่อน ให้ใช้คำสั่งต่อไปนี้ เพื่อ Login ก่อนทำการ Push

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

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

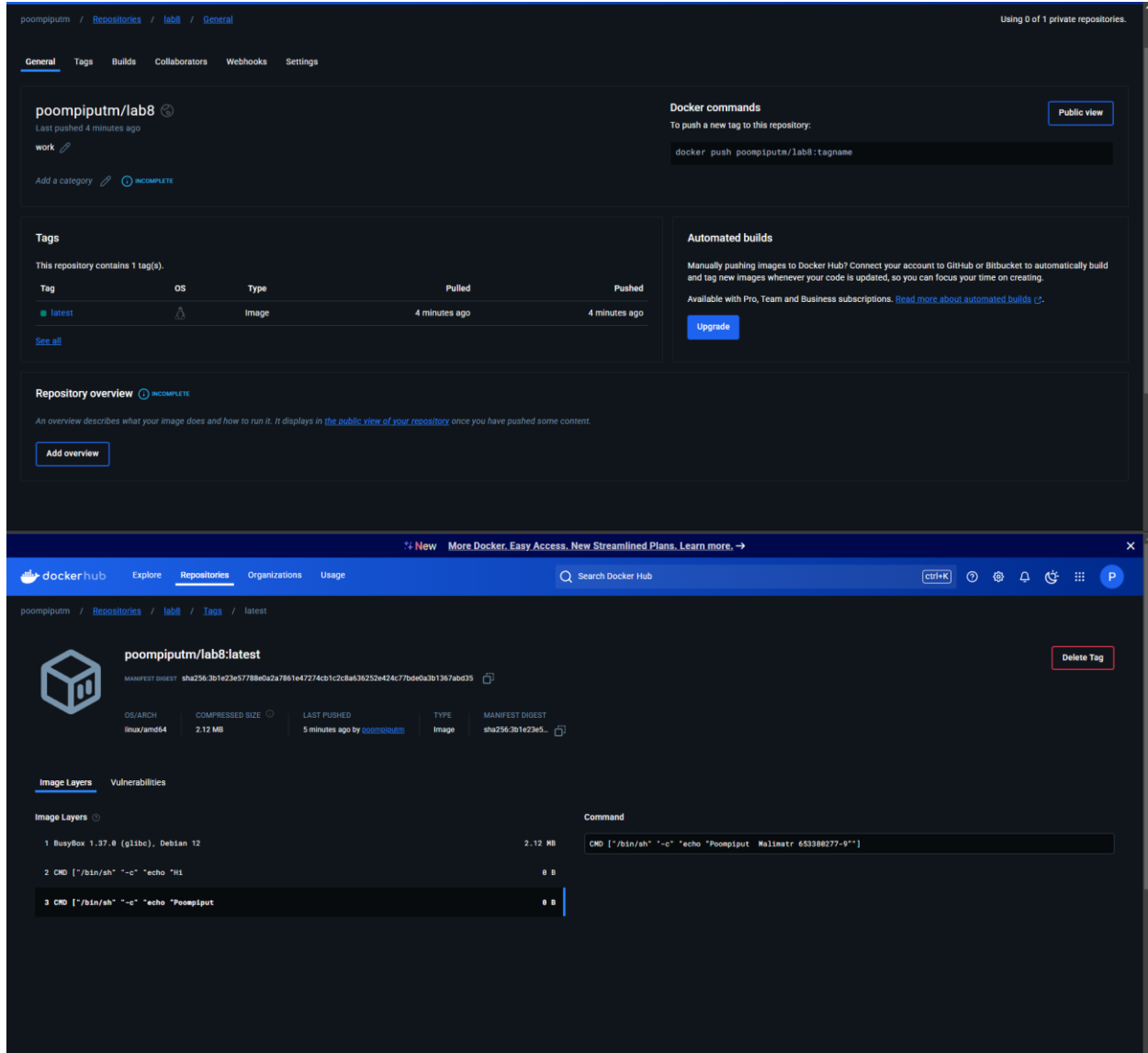


7. ไปที่ Docker Hub กด Tab ชื่อ Tags หรือไปที่ Repository ก็ได้



Lab Worksheet

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



The screenshot shows the Docker Hub interface for the repository `poomiputn/lab8`. The page is divided into several sections:

- General:** Shows the repository name `poomiputn/lab8`, a status icon, and the last push time "Last pushed 4 minutes ago". It includes a "work" link and an "Add a category" button.
- Tags:** A table showing the available tags. The table has columns for Tag, OS, Type, Pulled, and Pushed.

Tag	OS	Type	Pulled	Pushed
latest	linux/amd64	Image	4 minutes ago	4 minutes ago
- Docker commands:** A section for pushing new tags, with a "Public view" button and a command box containing `docker push poomiputn/lab8:tagname`.
- Automated builds:** A section explaining how to connect Docker Hub to GitHub or Bitbucket for automated builds, with an "Upgrade" button.
- Repository overview:** A section with an "Add overview" button.

Below the repository overview, there is a navigation bar with links for "New", "More Docker", "Easy Access", and "New Streamlined Plans".

The second part of the screenshot shows the `latest` tag details for `poomiputn/lab8:latest`. It includes a "Delete Tag" button and a table with the following information:

OS/ARCH	COMPRESSED SIZE	LAST PUSHED	TYPE	MANIFEST DIGEST
linux/amd64	2.12 MB	5 minutes ago by poomiputn	Image	sha256:3b1e23e5...

Below the table, there are tabs for "Image Layers" and "Vulnerabilities". The "Image Layers" tab is active, showing a list of layers with their sizes and commands. The layers are:

- BusyBox 1.37.0 (glibc), Debian 12: 2.12 MB
- CMD ["/bin/sh" "-c" "echo 'Hi'"]
- CMD ["/bin/sh" "-c" "echo 'Poomiput'"]

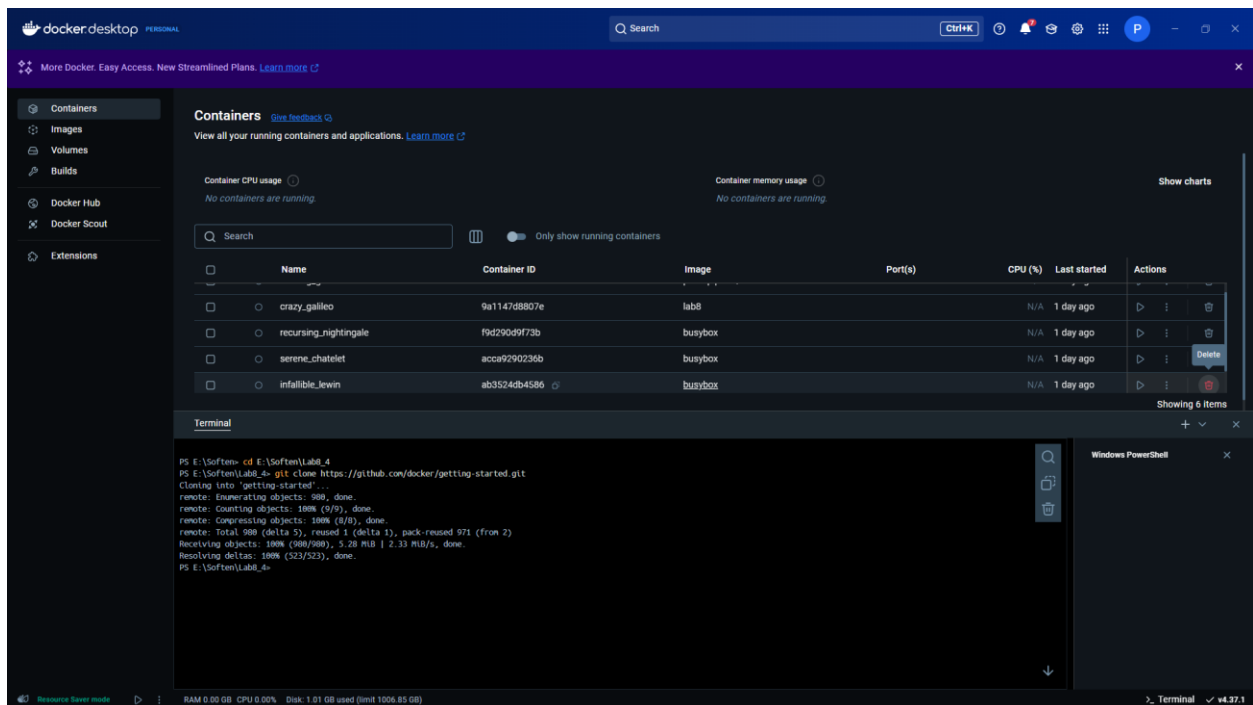
The "Command" tab is also visible, showing the command `CMD ["/bin/sh" "-c" "echo 'Poomiput Malimetr 653388277-9'"]`.

Lab Worksheet

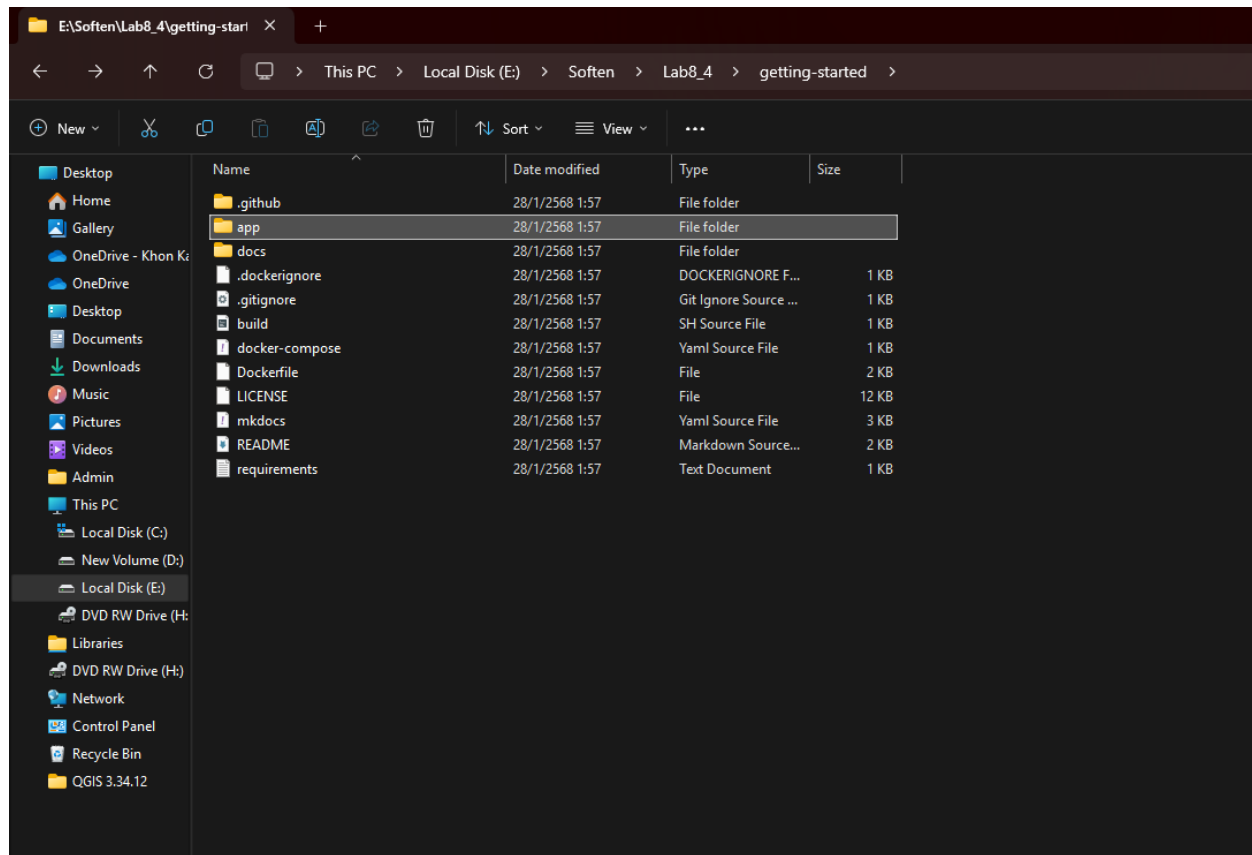
แบบฝึกปฏิบัติที่ 8.4: การ Build แอปพลิเคชันจาก Container image และการ Update แอปพลิเคชัน

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 พร้อมกับใส่เนื้อหาดังต่อไปนี้ลงไปไฟล์

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

```
WORKDIR /app
```

```
COPY . .
```

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

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

```
EXPOSE 3000
```

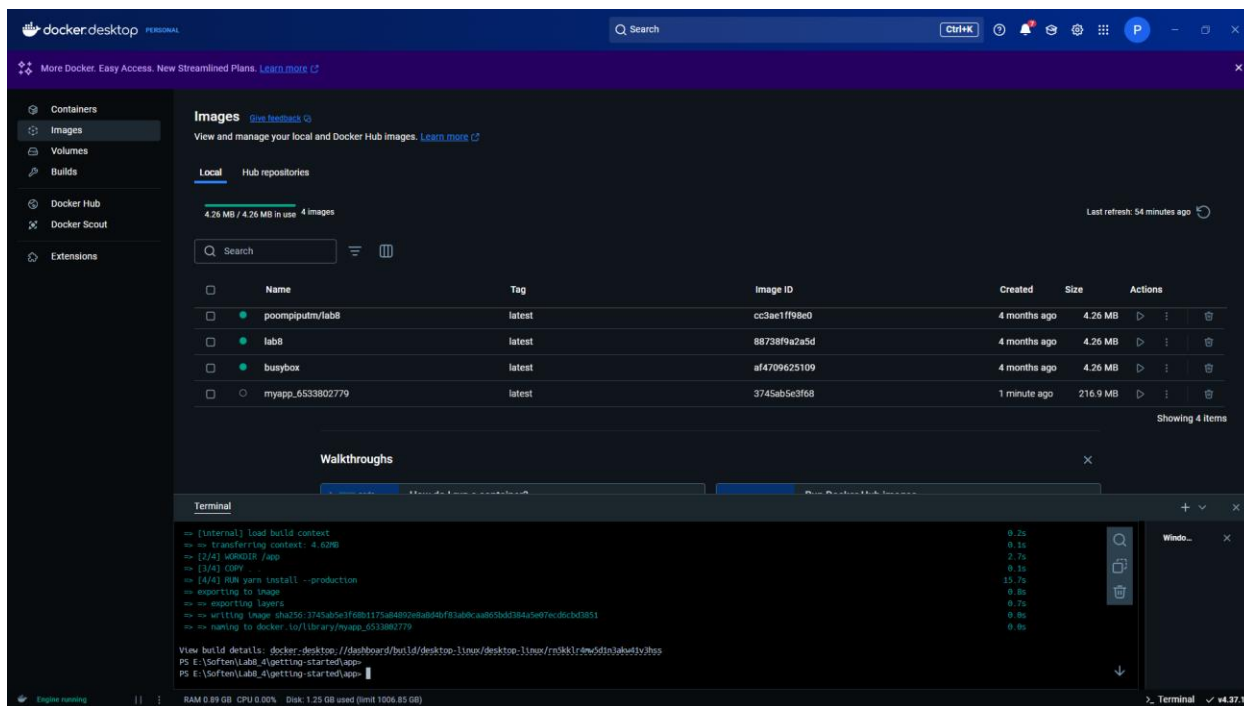
Lab Worksheet

5. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้ โดยกำหนดให้ชื่อ image เป็น myapp_รหัสสนศ. ไม่มีขีด

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

[Check point#8] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง)

แสดงคำสั่งและผลลัพธ์ที่ได้ทางหน้าจอ



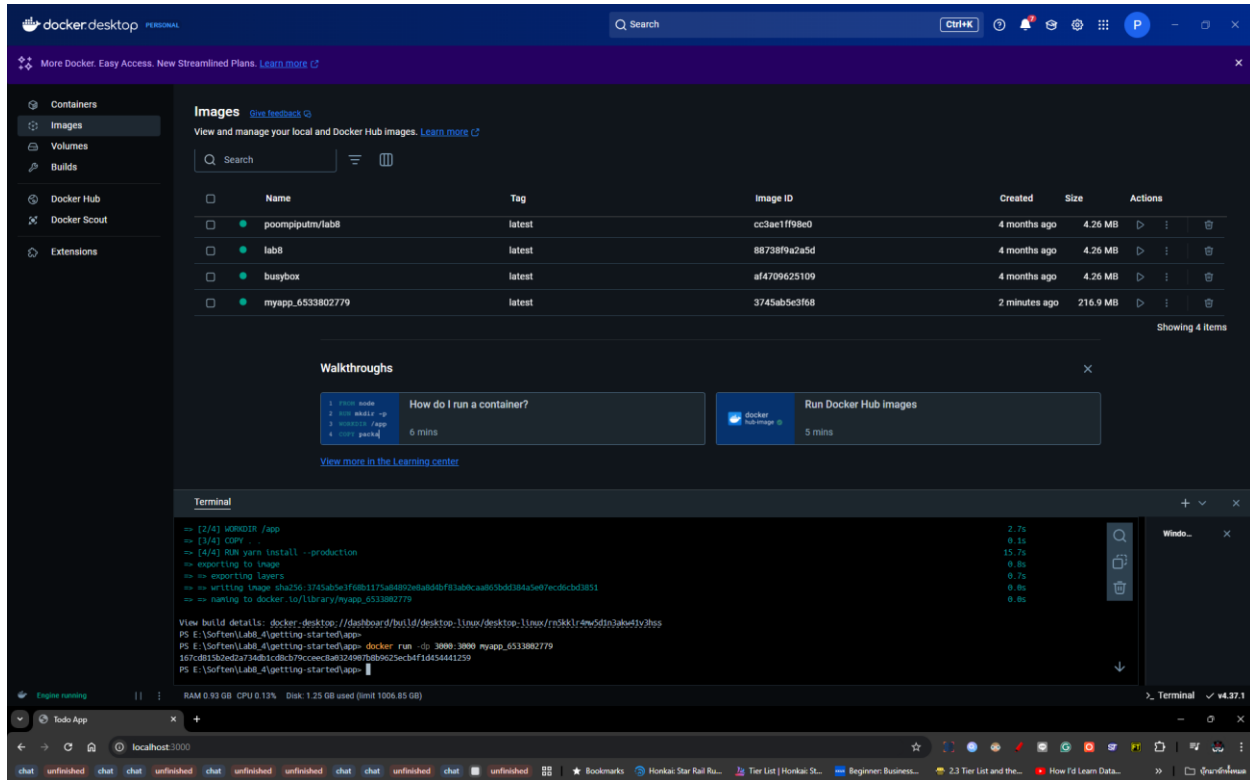
6. ทำการ Start ตัว Container ของแอปพลิเคชันที่สร้างขึ้น โดยใช้คำสั่ง

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

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

Lab Worksheet

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



New Item Add Item

No items yet! Add one above!

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>

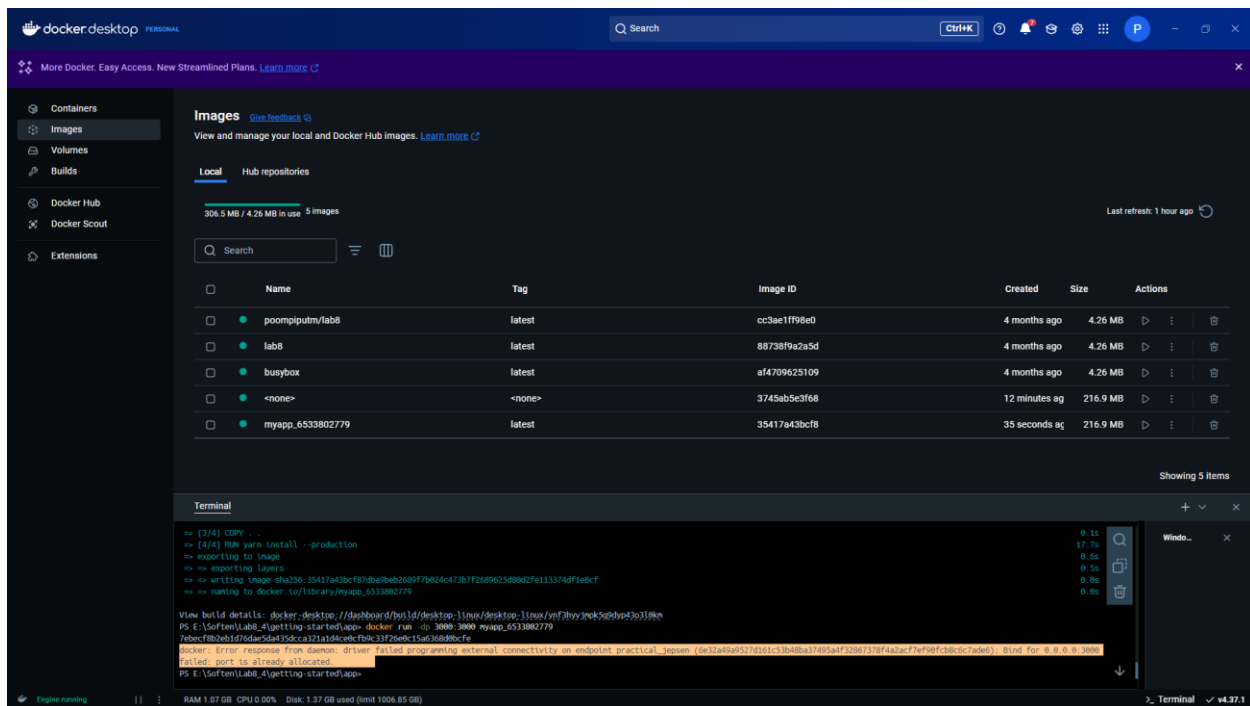
b. Save ไฟล์ให้เรียบร้อย

9. ทำการ Build Docker image โดยใช้คำสั่งเดียวกันกับข้อ 5

10. Start และรัน Container ตัวใหม่ โดยใช้คำสั่งเดียวกันกับข้อ 6

[Check point#10] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง)

แสดงคำสั่งและผลลัพธ์ที่ได้ทางหน้าจอ พร้อมกับตอบคำถามต่อไปนี้



(1) Error ที่เกิดขึ้นหมายความว่าอย่างไร และเกิดขึ้นเพราะอะไร

docker: Error response from daemon: driver failed programming external connectivity on endpoint practical_jepsen

(6e32a49a9527d161c53b48ba37495a4f32867378f4a2acf7ef90fcb8c6c7ade6): Bind for

Lab Worksheet

0.0.0.0:3000 failed: port is already allocated. หมายความว่าไม่สามารถใช้งานได้เนื่องจากคำสั่งนี้ใช้ port ที่กำลังใช้งานอยู่จึงใช้งานไม่ได้เพราะ port ทับซ้อนกัน

11. ลบ Container ของ Web application เวอร์ชันก่อนแก้ไขออกจากระบบ โดยใช้วิธีใดวิธีหนึ่งดังต่อไปนี้

a. ผ่าน Command line interface

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

b. ผ่าน Docker desktop

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

12. Start และรัน Container ตัวใหม่อีกครั้ง โดยใช้คำสั่งเดียวกันกับข้อ 6

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

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



Lab Worksheet

แบบฝึกปฏิบัติที่ 8.5: เริ่มต้นสร้าง Pipeline อย่างง่ายสำหรับการ Deploy ด้วย Jenkins

1. เปิด Command line หรือ Terminal บน Docker Desktop

2. บ้อนคำสั่งและทำการรัน container โดยผูกพอร์ต

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

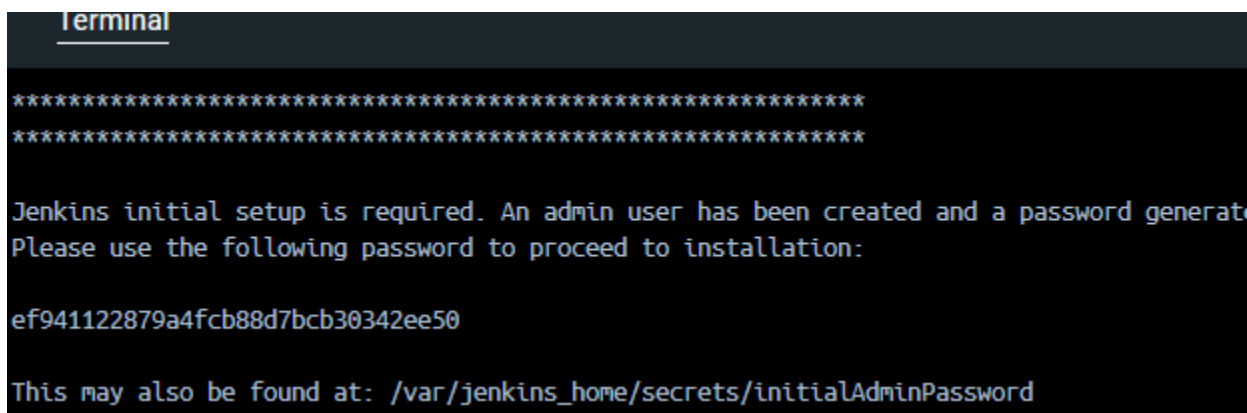
หรือ

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

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

3. บันทึกรหัสผ่านของ Admin user ไว้สำหรับ log-in ในครั้งแรก

[Check point#12] Capture หน้าจอที่แสดงผล Admin password



```

Terminal

*****
*****

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

ef941122879a4fcb88d7bcb30342ee50

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

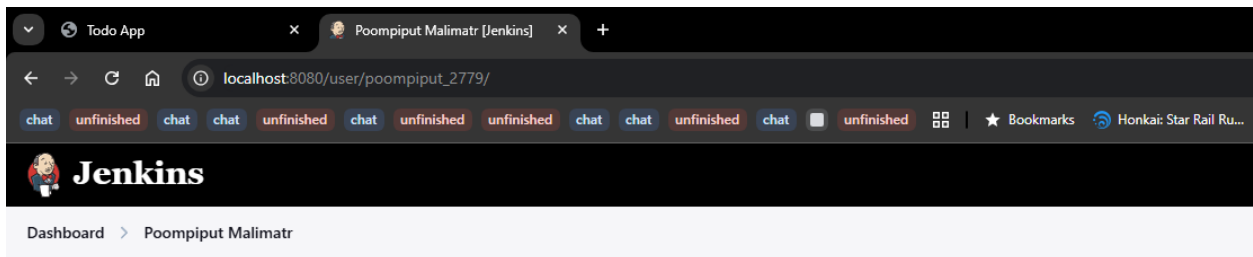
4. เมื่อได้รับการยืนยันว่า Jenkins is fully up and running ให้เปิดบราวเซอร์ และบ้อนที่อยู่เป็น
localhost:8080

5. ทำการ Unlock Jenkins ด้วยรหัสผ่านที่ได้ในข้อที่ 3

6. สร้าง Admin User โดยใช้ username เป็นชื่อจริงของนักศึกษาพร้อมรหัสสี่ตัวท้าย เช่น somsri_3062

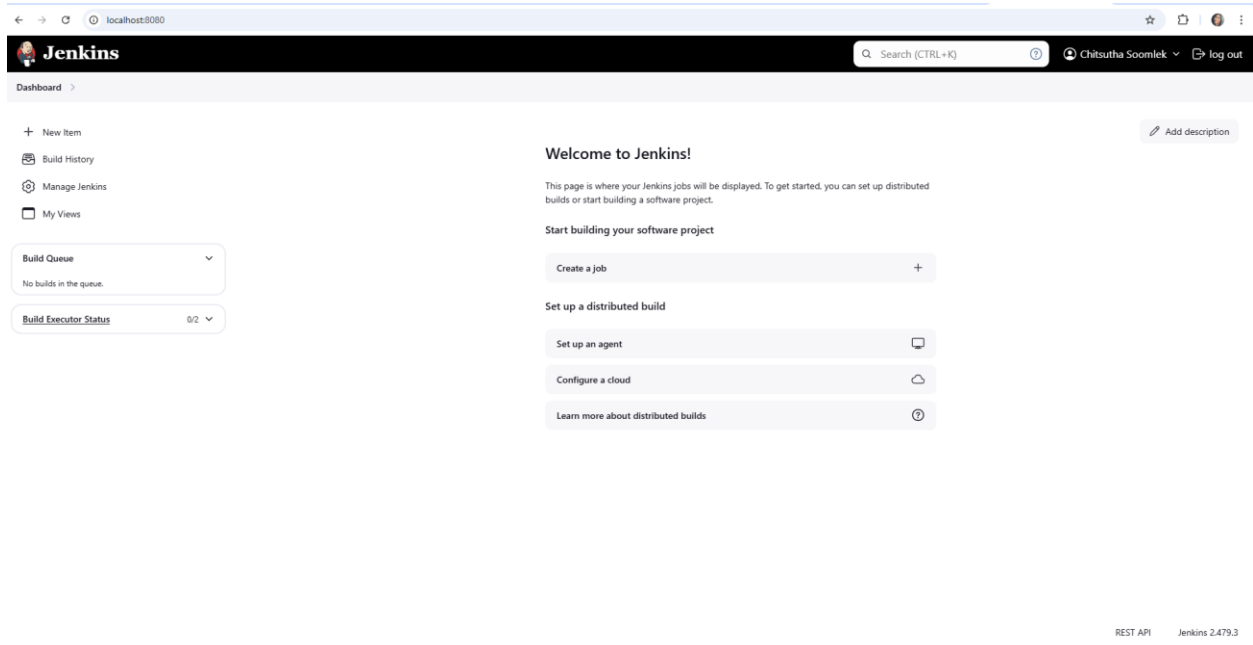
Lab Worksheet

[Check point#13] Capture หน้าจอที่แสดงผลการตั้งค่า



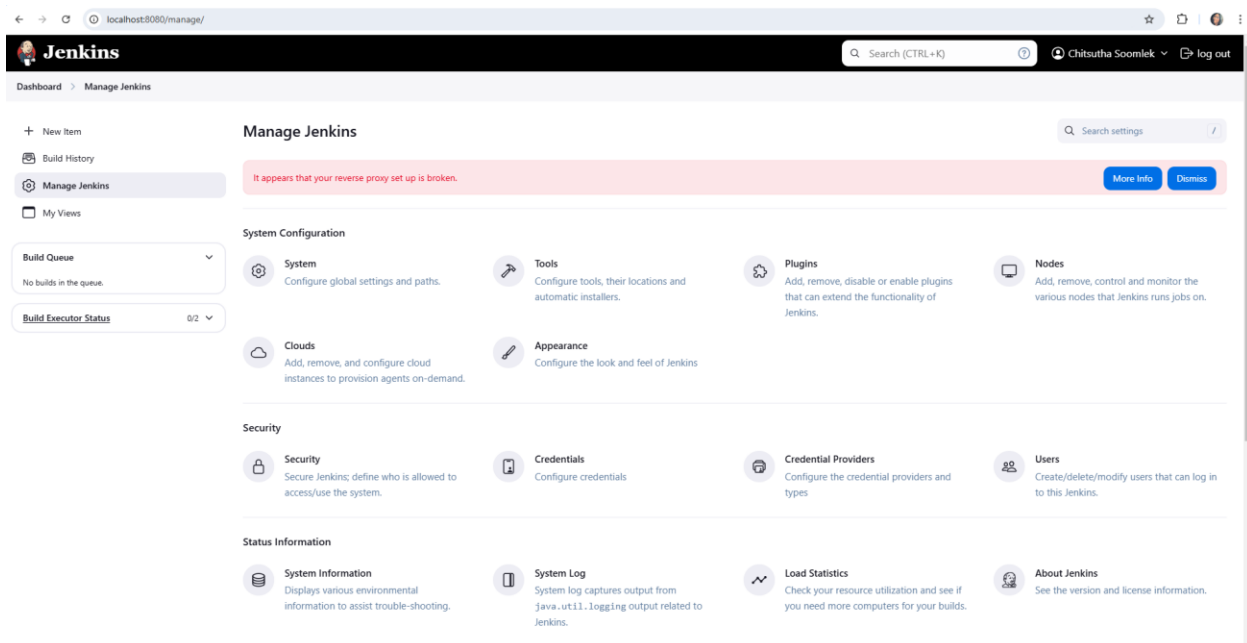
7. กำหนด Jenkins URL เป็น <http://localhost:8080/lab8>

8. เมื่อติดตั้งเรียบร้อยแล้วจะพบกันหน้า Dashboard ดังแสดงในภาพ



Lab Worksheet

9. เลือก Manage Jenkins แล้วไปที่เมนู Plugins

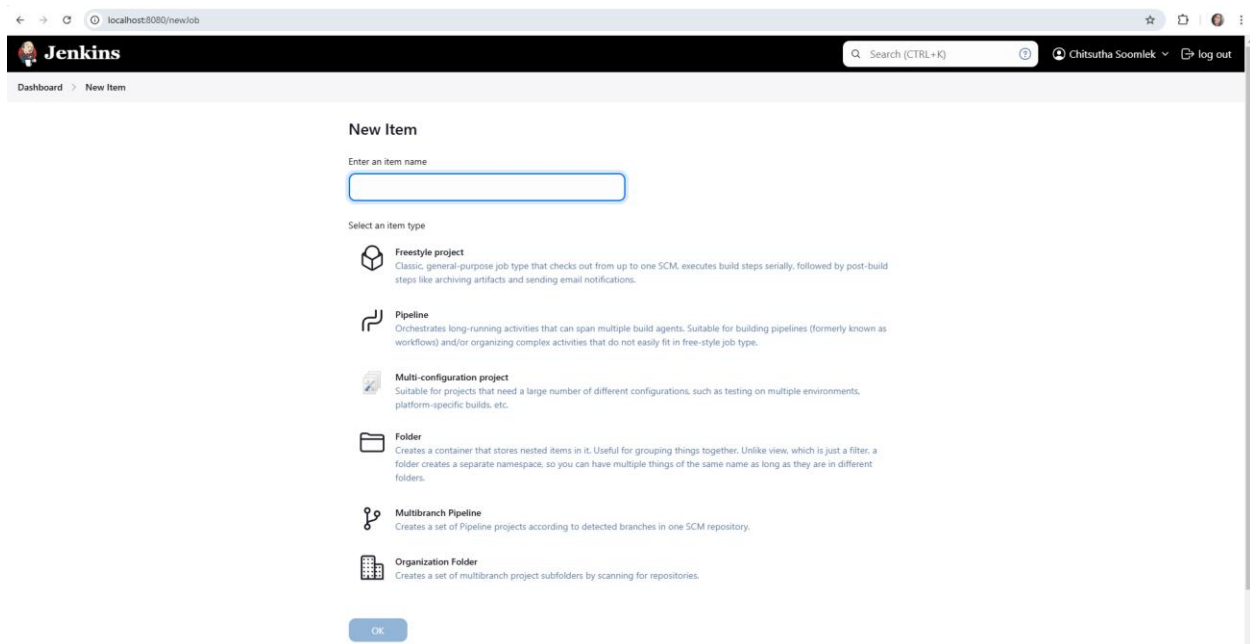


10. ไปที่เมนู Available plugins แล้วเลือกติดตั้ง Robotframework เพิ่มเติม



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

Lab Worksheet



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

Description: Lab 8.5

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

Build Trigger: เลือกแบบ Build periodically แล้วกำหนดให้ build ทุก 15 นาที

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

[Check point#14] Capture หน้าจอแสดงการตั้งค่า พร้อมกับตอบคำถามต่อไปนี้

Lab Worksheet

The screenshot shows the Jenkins configuration page for a job named 'Lab 8.5'. The page is divided into two main sections: 'General' and 'Build Triggers'. The 'General' section is currently active, showing the job's description, a GitHub project configuration, and various options like 'Discard old builds', 'This project is parameterized', 'Throttle builds', and 'Execute concurrent builds if necessary'. The 'Build Triggers' section is also visible, showing options for 'Trigger builds remotely', 'Build after other projects are built', and 'Build periodically'. The 'Build periodically' option is selected, and the schedule is set to 'H/15 * * * *'. The page also includes a search bar at the top and a user profile dropdown.

Jenkins Search (CTRL+K) Poompiput Malinrat log out

Dashboard > UAT > Configuration

Configure **General** Enabled

General

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

Description

Lab 8.5

Plain text [Preview](#)

☐ Discard old builds ?

☒ GitHub project

Project url ?

<https://github.com/kku-computer-science/configuration-management-Siegrimson.git/>

Advanced

☐ This project is parameterized ?

☐ Throttle builds ?

☐ Execute concurrent builds if necessary ?

Advanced

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 9:35:47 PM Coordinated Universal Time; would next run at Monday, January 27, 2025 at 9:50:47 PM Coordinated Universal Time.

☐ GitHub hook trigger for GITScm polling ?

☐ Poll SCM ?

Lab Worksheet

The screenshot shows the Jenkins configuration page for a job named 'UAT'. The left sidebar contains a 'Configure' menu with options: General, Source Code Management, Build Triggers, Build Environment, Build Steps, and Post-build Actions. The 'Build Steps' section is expanded, showing a step named 'Execute shell' with the following commands:

```
# Navigate to the mounted directory inside the container
cd /data

# Create a directory for test results
mkdir -p /data/results

# Run Robot Framework tests
robot --outputdir /data/results UAT-Lab7-001.robot UAT-Lab7-002.robot
```

The 'Post-build Actions' section is also expanded, showing a step named 'Publish Robot Framework test results'. The 'Directory of Robot output' is set to '/data/results'. The 'Thresholds for build result' are set to 20.0% (yellow) and 80.0% (blue). There is a checkbox for 'DEPRECATED! THIS FLAG DOES NOTHING! - Use thresholds for critical tests only' which is checked, and a checkbox for 'Include skipped tests in total count for thresholds' which is unchecked.

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

Navigate to the mounted directory inside the container

cd /data

Create a directory for test results

mkdir -p /data/results

Run Robot Framework tests

robot --outputdir /data/results UAT-Lab7-001.robot UAT-Lab7-002.robot

Lab Worksheet

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


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

13. กด Apply และ Save

14. สั่ง Build Now

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

Lab Worksheet


Jenkins

Dashboard > UAT > #11

Status

</> Changes
Console Output
Edit Build Information
Delete build '#11'
Timings
Previous Build

✖ #11 (Jan 27, 2025, 9:50:15 PM)

Started by user [Poompiput Malimat](#)

This run spent:

- 2 ms waiting;
- 12 ms build duration;
- 14 ms total from scheduled to completion.

</> No changes.

Dashboard > UAT > #11 > Console Output

</> Changes

Console Output

Edit Build Information
Delete build '#11'
Timings
Previous Build

```

Started by user Poompiput Malimat
Running as SYSTEM
Building in workspace /var/jenkins_home/workspace/UAT
[UAT] $ /bin/sh -xe /tmp/jenkins12523363679158248642.sh
+ cd /data
+ mkdir -p /data/results
+ robot --outputdir /data/results UAT-Lab7-001.robot UAT-Lab7-002.robot
/tmp/jenkins12523363679158248642.sh: 9: 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:
Failed!
hudson.AbortException: No files found in path /data/results with configured filemask: output.xml
    at PluginClassLoader for robot//hudson.plugins.robot.RobotParser$RobotParserCallable.invoke(RobotParser.java:81)
    at PluginClassLoader for robot//hudson.plugins.robot.RobotParser$RobotParserCallable.invoke(RobotParser.java:52)
    at hudson.FilePath.act(FilePath.java:1234)
    at hudson.FilePath.act(FilePath.java:1217)
    at PluginClassLoader for robot//hudson.plugins.robot.RobotParser.parse(RobotParser.java:48)
    at PluginClassLoader for robot//hudson.plugins.robot.RobotPublisher.parse(RobotPublisher.java:262)
    at PluginClassLoader for robot//hudson.plugins.robot.RobotPublisher.perform(RobotPublisher.java:286)
    at hudson.tasks.BuildStepCompatibilityLayer.perform(BuildStepCompatibilityLayer.java:88)
    at hudson.tasks.BuildStepMonitor$1.perform(BuildStepMonitor.java:20)
    at hudson.model.AbstractBuild$AbstractBuildExecution.perform(AbstractBuild.java:818)
    at hudson.model.AbstractBuild$AbstractBuildExecution.performAllBuildSteps(AbstractBuild.java:767)
    at hudson.model.Build$BuildExecution.post2(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)
Finished: FAILURE

```