

Lab Worksheet

ชื่อ-นามสกุล: นายณัฐกร หาญกล้า รหัสนักศึกษา: 653380324-6

Section: 4

Lab#8 – Software Deployment Using Docker**วัตถุประสงค์การเรียนรู้**

1. ผู้เรียนสามารถอธิบายเกี่ยวกับ Software deployment ได้
2. ผู้เรียนสามารถสร้างและรัน Container จาก Docker image ได้
3. ผู้เรียนสามารถสร้าง Docker files และ Docker images ได้
4. ผู้เรียนสามารถนำซอฟต์แวร์ที่พัฒนาขึ้นให้สามารถรันบนสภาพแวดล้อมเดียวกันและทำงานร่วมกันกับสมาชิกในทีมพัฒนาซอฟต์แวร์ผ่าน Docker hub ได้
5. ผู้เรียนสามารถเริ่มต้นใช้งาน Jenkins เพื่อสร้าง Pipeline ในการ Deploy งานได้

Pre-requisite

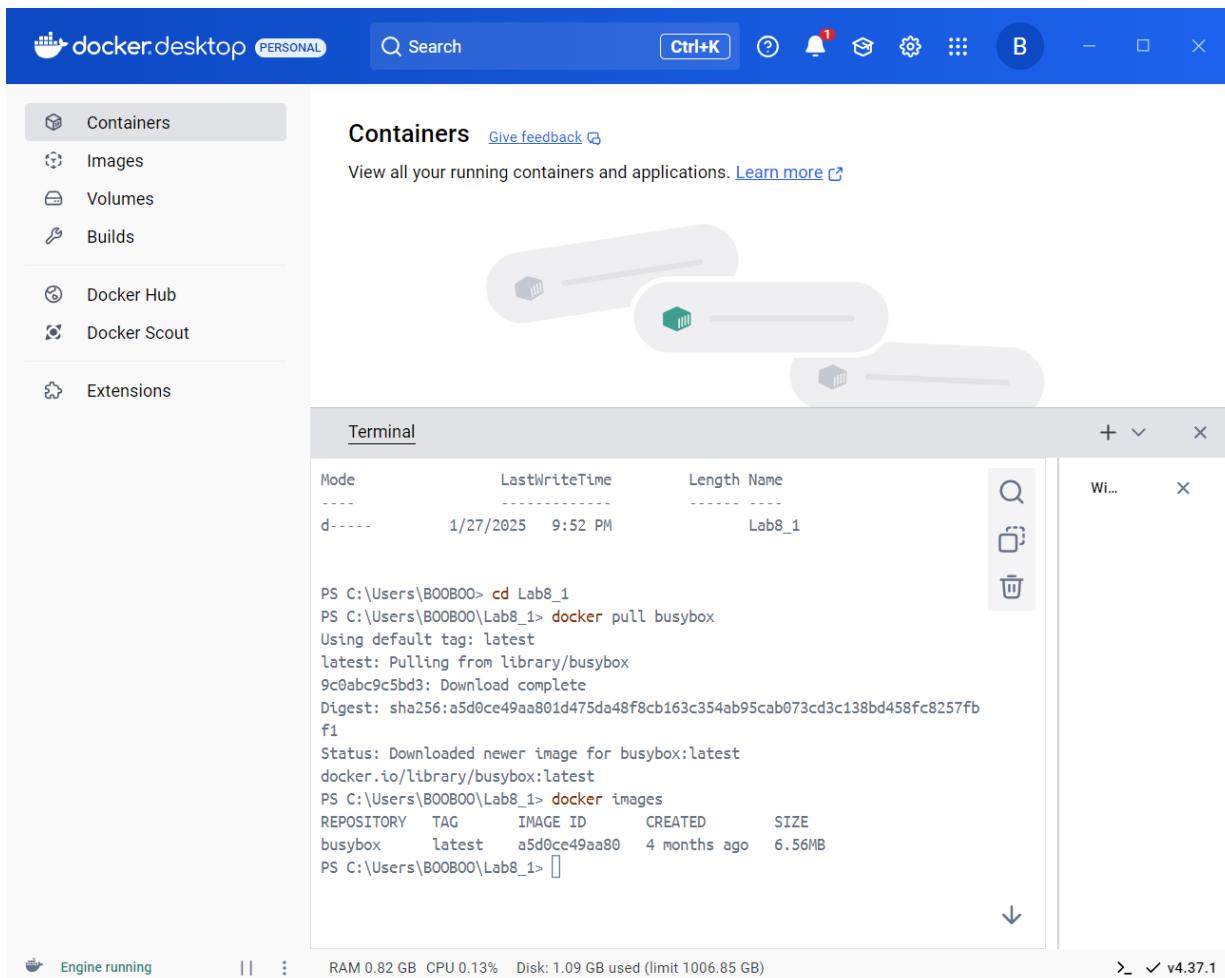
1. ติดตั้ง Docker desktop ลงบนเครื่องคอมพิวเตอร์ โดยดาวน์โหลดจาก <https://www.docker.com/get-started>
2. สร้าง Account บน Docker hub (<https://hub.docker.com/signup>)
3. กำหนดให้ \$ หมายถึง Command prompt และ <> หมายถึง ให้ป้อนค่าของพารามิเตอร์ที่กำหนด

แบบฝึกปฏิบัติที่ 8.1 Hello world - รัน Container จาก Docker image

1. เปิดใช้งาน Docker desktop และ Login ด้วย Username และ Password ที่ลงทะเบียนกับ Docker Hub เอาไว้
 1. เปิด Command line หรือ Terminal บน Docker Desktop จากนั้นสร้าง Directory ชื่อ Lab8_1
 2. ย้ายตำแหน่งปัจจุบันไปที่ Lab8_1 เพื่อใช้เป็น Working directory
 3. ป้อนคำสั่ง \$ docker pull busybox หรือ \$ sudo docker pull busybox สำหรับกรณีที่ติดปัญหา Permission denied
(หมายเหตุ: BusyBox เป็น software suite ที่รองรับคำสั่งบางอย่างบน Unix - <https://busybox.net>)
 4. ป้อนคำสั่ง \$ docker images

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

Lab Worksheet



(1) สิ่งที่อยู่ภายใต้คอลัมน์ Repository คืออะไร =>

ตอบ ชื่อของ Docker Image : busybox ที่ถูก Pull มา

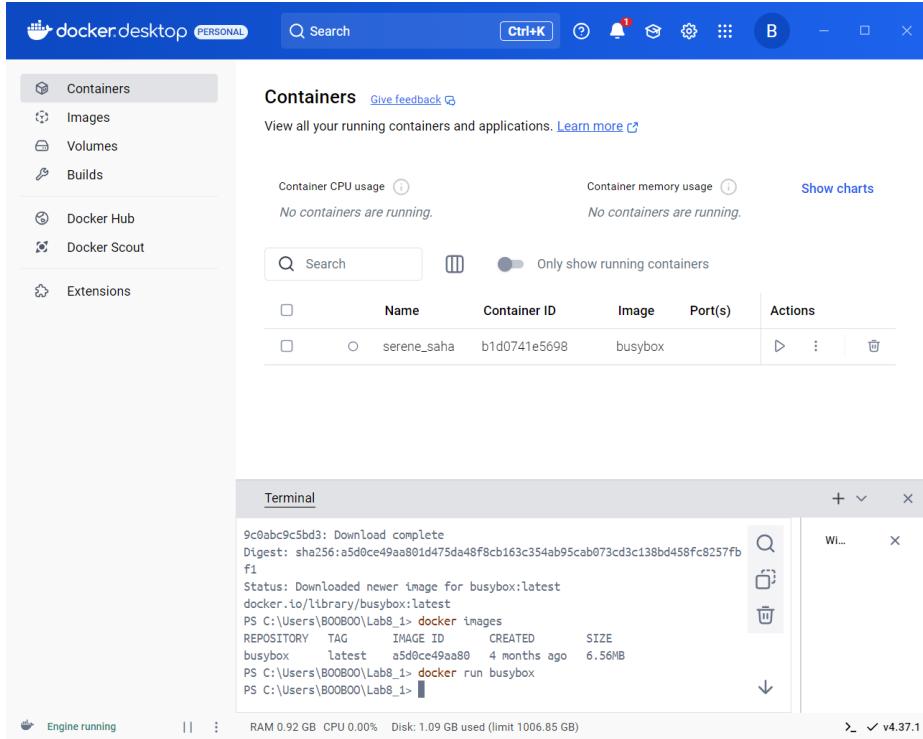
(2) Tag ที่ใช้บ่งบอกถึงอะไร

ตอบ Version ปัจจุบันของ Image นั้นๆ ในภาพคือ Latest คือล่าสุด

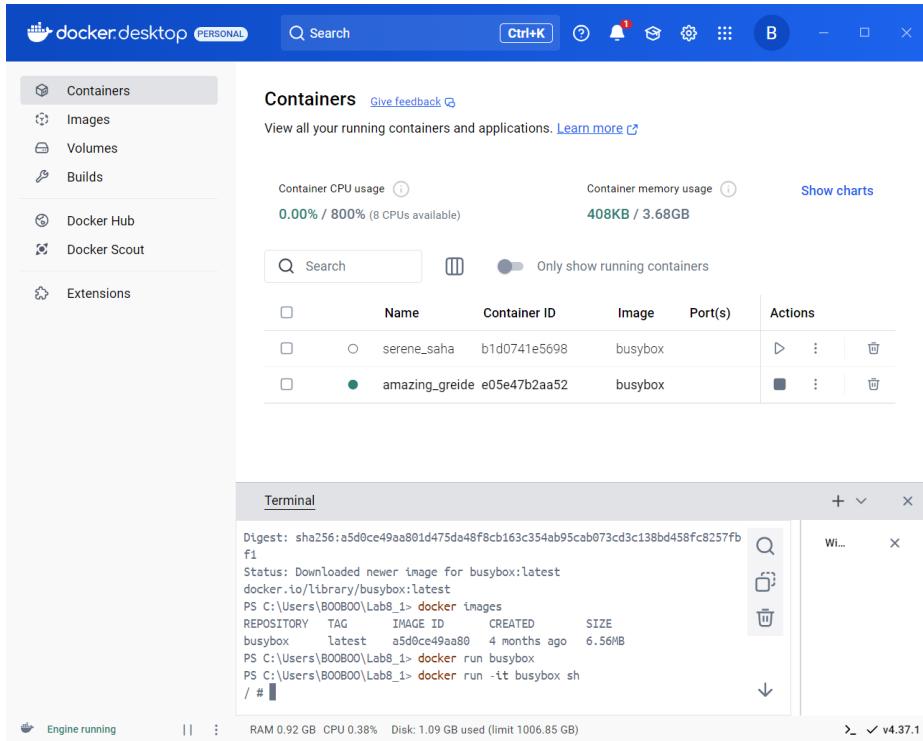
[Check point#2] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) และแสดงผลลัพธ์ที่ได้ตั้งแต่ขั้นตอนที่ 6-12 พร้อมกับตอบคำถามต่อไปนี้

Lab Worksheet

5. ป้อนคำสั่ง \$ docker run busybox



6. ป้อนคำสั่ง \$ docker run -it busybox sh



Lab Worksheet

7. ป้อนคำสั่ง ls

Docker Desktop interface showing the 'Containers' tab. Two containers are listed:

Name	Container ID	Image	Port(s)	Actions
serene_saha	b1d0741e5698	busybox		
amazing_greide	e05e47b2aa52	busybox		

The terminal window shows the output of the 'ls' command in a busybox container:

```
PS C:\Users\BOOBOO\Lab8_1> docker run busybox
/ # ls
bin etc lib proc sys usr
dev home lib64 root tmp var
/ #
```

8. ป้อนคำสั่ง ls -la

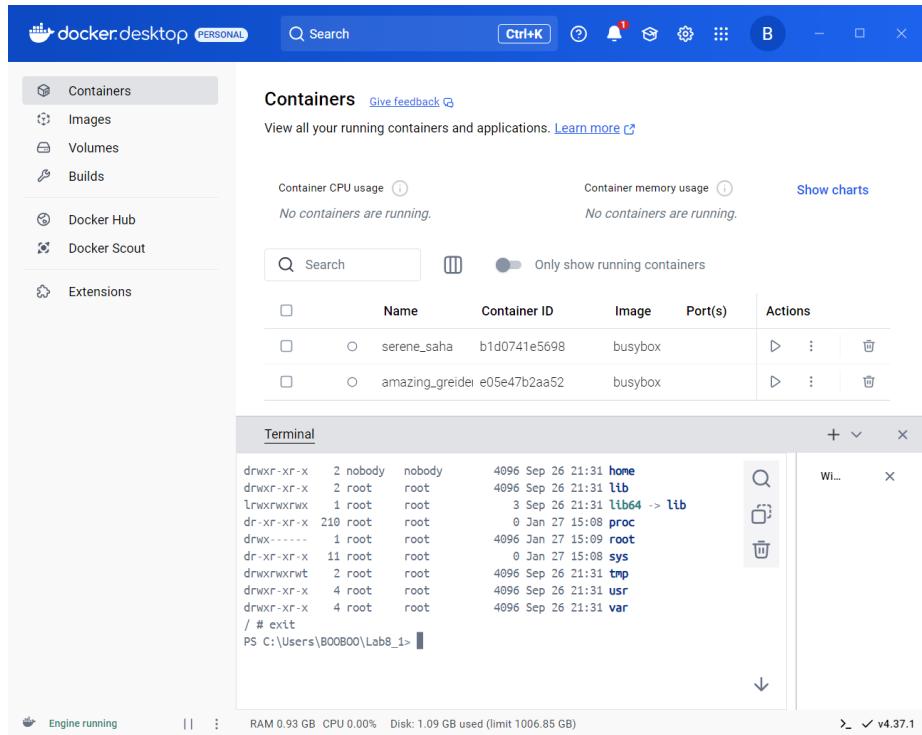
Docker Desktop interface showing the 'Containers' tab. The same two containers are listed.

The terminal window shows the output of the 'ls -la' command in a busybox container:

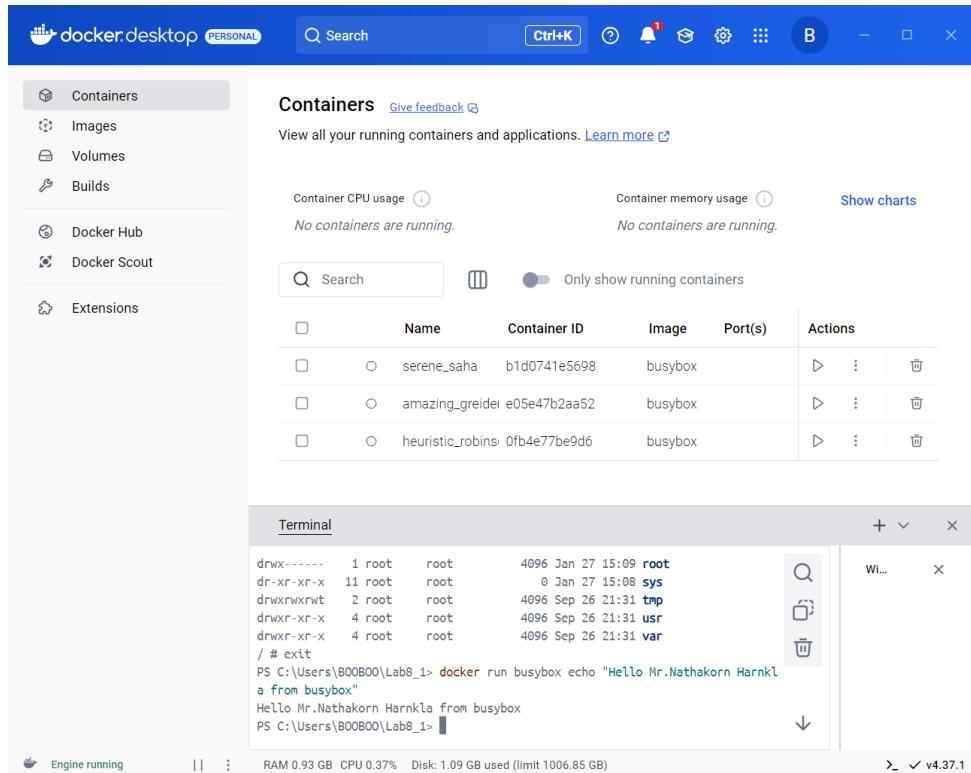
```
PS C:\Users\BOOBOO\Lab8_1> docker run -it busybox sh
/ # ls -la
drwxr-xr-x  2 nobody  nobody   4096 Sep 26 21:31 home
drwxr-xr-x  2 root    root     4096 Sep 26 21:31 lib
lrwxrwxrwx  1 root    root     3 Sep 26 21:31 lib64 -> lib
dr-xr-xr-x  210 root   root    0 Jan 27 15:08 proc
drwx----- 1 root    root    4096 Jan 27 15:09 root
dr-xr-xr-x  11 root   root    0 Jan 27 15:08 sys
drwxrwxrwt  2 root   root    4096 Sep 26 21:31 tmp
drwxr-xr-x  4 root   root    4096 Sep 26 21:31 usr
drwxr-xr-x  4 root   root    4096 Sep 26 21:31 var
/ #
```

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. On the left, a sidebar has 'Containers' selected. The main area displays a table of running containers:

	Name	Container ID	Image	Port(s)	Actions
<input type="checkbox"/>	serene_saha	b1d0741e5698	busybox		▷ ⋮ 🗑
<input type="checkbox"/>	amazing_greider	e05e47b2aa52	busybox		▷ ⋮ 🗑
<input type="checkbox"/>	heuristic_robins	0fb4e77be9d6	busybox		▷ ⋮ 🗑

Below this is a 'Terminal' window showing the command output:

```
PS C:\Users\BOOBOO\Lab8_1> docker run busybox echo "Hello Mr.Nathakorn Harnkl  
a from busybox"  
PS C:\Users\BOOBOO\Lab8_1> docker ps -a  
CONTAINER ID IMAGE COMMAND CREATED STATUS  
PORTS NAMES  
0fb4e77be9d6 busybox "echo 'Hello Mr.Nathakorn Harnkl  
a from busybox'" About a minute ago Exited  
(0) About a minute ago heuristic_robinson  
e05e47b2aa52 busybox "sh"  
(0) 3 minutes ago amazing_greider  
b1d0741e5698 busybox "sh"  
(0) 9 minutes ago serene_saha
```

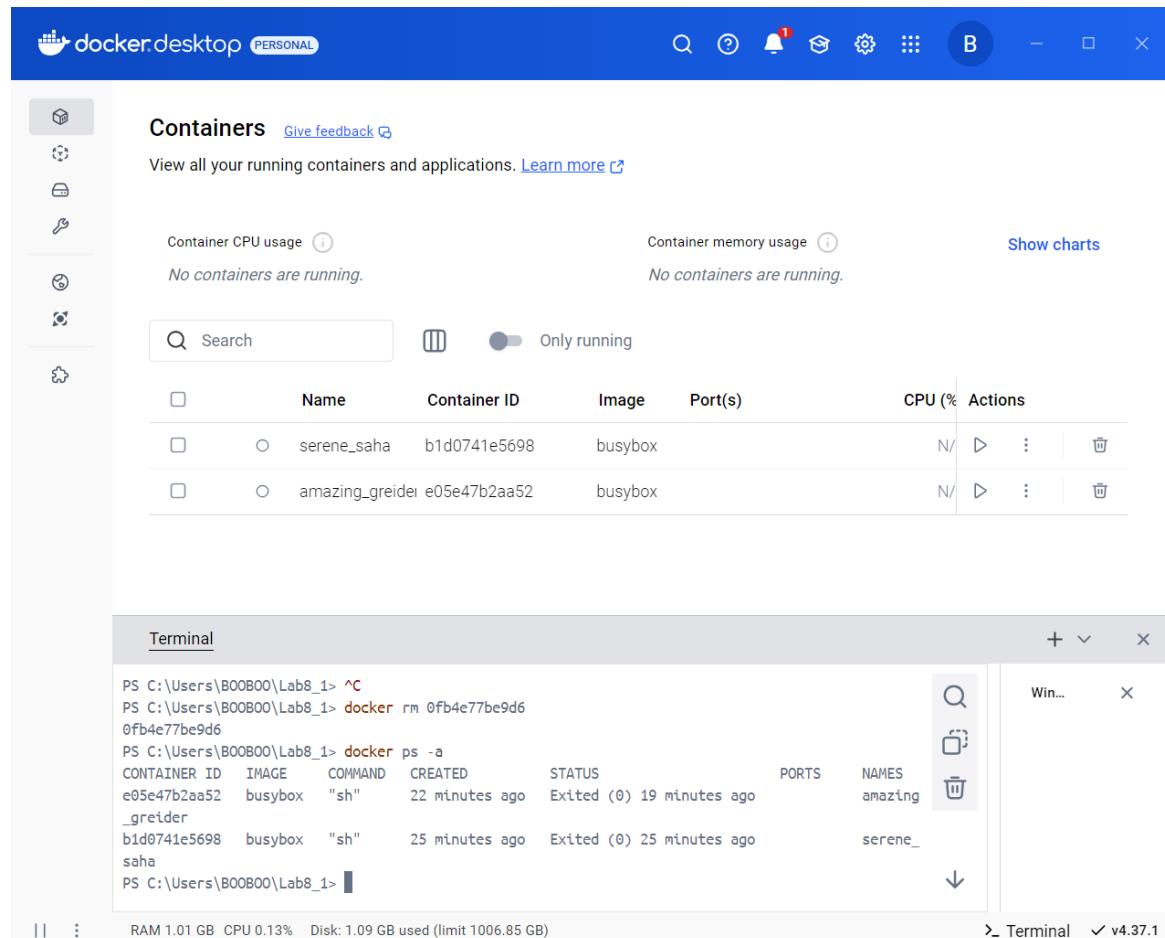
At the bottom, status information is shown: Engine running, RAM 0.92 GB, CPU 0.00%, Disk: 1.09 GB used (limit 1006.85 GB), and version v4.37.1.

- (1) เมื่อใช้ option -it ในคำสั่ง run ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสั้นๆ
ตอบ เมื่อใช้ -it จะช่วยให้สามารถเข้าสู่ Shell ภายใน Container และทำงานได้ต่อไปได้ เช่น การป้อนคำสั่ง ls, pwd, exit สามารถโต้ตอบกับ Container ผ่าน Terminal ได้โดยตรง
- (2) คอลัมน์ STATUS จากการรันคำสั่ง docker ps -a แสดงถึงข้อมูลอะไร
ตอบ แสดงผลลัพธ์ของ docker ps -a คือ สถานะของแต่ละ Container ในภาพแสดง Exited หมายถึง Container หยุดทำงานแล้ว และระบุเวลาที่หยุดทำงาน เช่น Exited (0) 3 minutes ago

Lab Worksheet

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

[Check point#3] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงผลลัพธ์ที่ได้ในขั้นตอนที่ 12



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

สำหรับเครื่องที่ใช้ระบบปฏิบัติการวินโดว์ (Windows) บันทึกคำสั่งต่อไปนี้ลงในไฟล์ โดยใช้ Text Editor ที่มี FROM busybox

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

Lab Worksheet

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

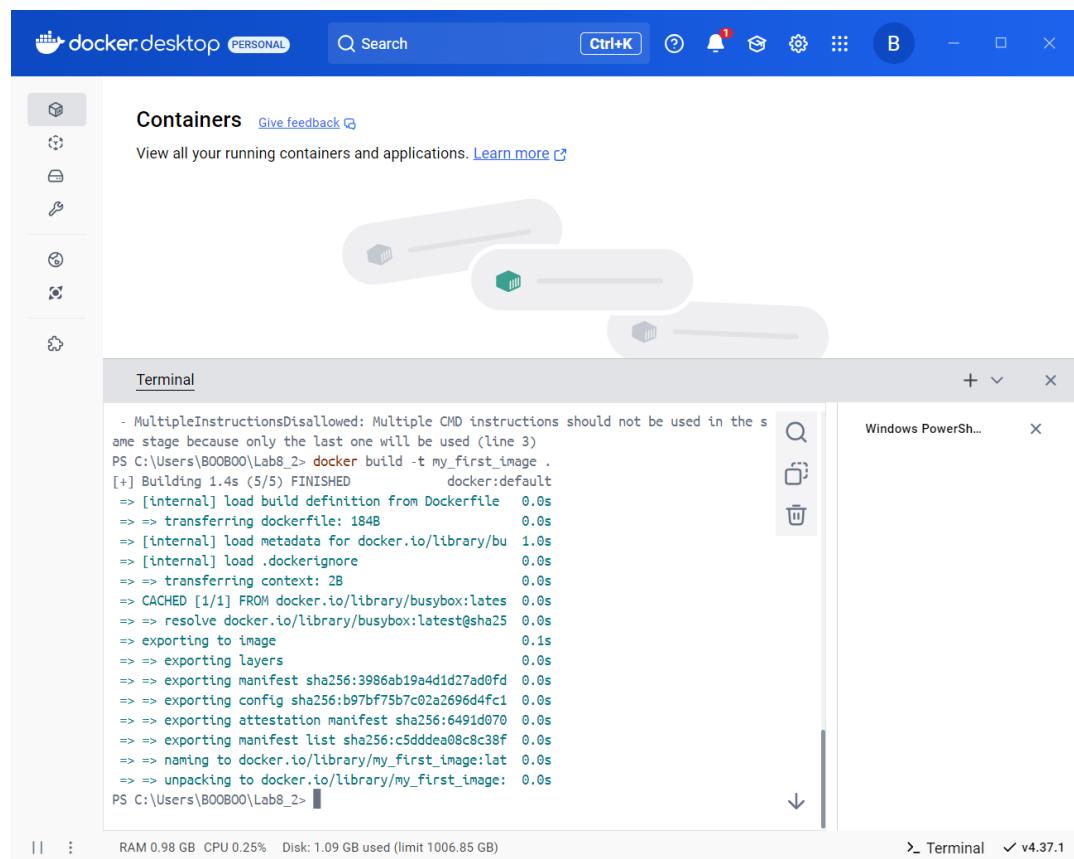
แล้วใช้ Text Editor ในการใส่เนื้อหาแทน



```
Dockerfile - Notepad
File Edit Format View Help
FROM busybox
CMD ["sh", "-c", "echo 'Hi there. This is my first docker image.' && echo 'Mr.Nathakorn Harnkla id:653380324-6 nickname:Ball'"]
```

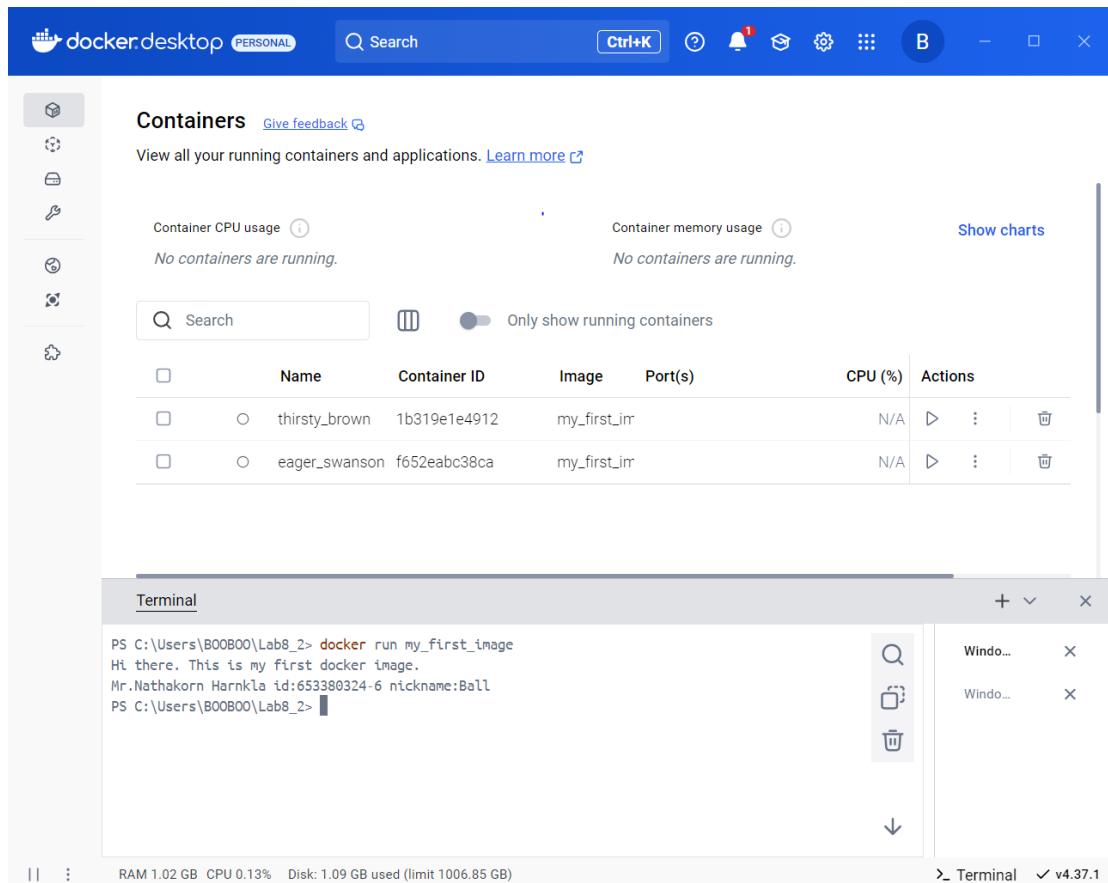
5. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้

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



Lab Worksheet

6. เมื่อ Build สำเร็จแล้ว ให้ทำการรัน Docker image ที่สร้างขึ้นในขั้นตอนที่ 5



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

(1) คำสั่งที่ใช้ในการ run คือ

ตอบ \$ docker build -t my_first_image . สำหรับสร้าง image ขึ้นมาใหม่

\$ docker run my_first_image สำหรับแสดงผลลัพธ์

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

ตอบ การใช้ -t เป็นการตั้งชื่อและแท็กที่ชัดเจน ทำให้จัดการ image ได้ง่ายขึ้น

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

สำหรับเครื่องที่ใช้ระบบปฏิบัติการ Mac OS หรือ Linux บนหน้าต่าง Terminal และป้อนคำสั่งต่อไปนี้

FROM busybox

CMD echo "Hi there. My work is done. You can run them from my Docker image."

CMD echo "ชื่อ-นามสกุล รหัสนักศึกษา"

สำหรับเครื่องที่ใช้ระบบปฏิบัติการ Mac OS 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

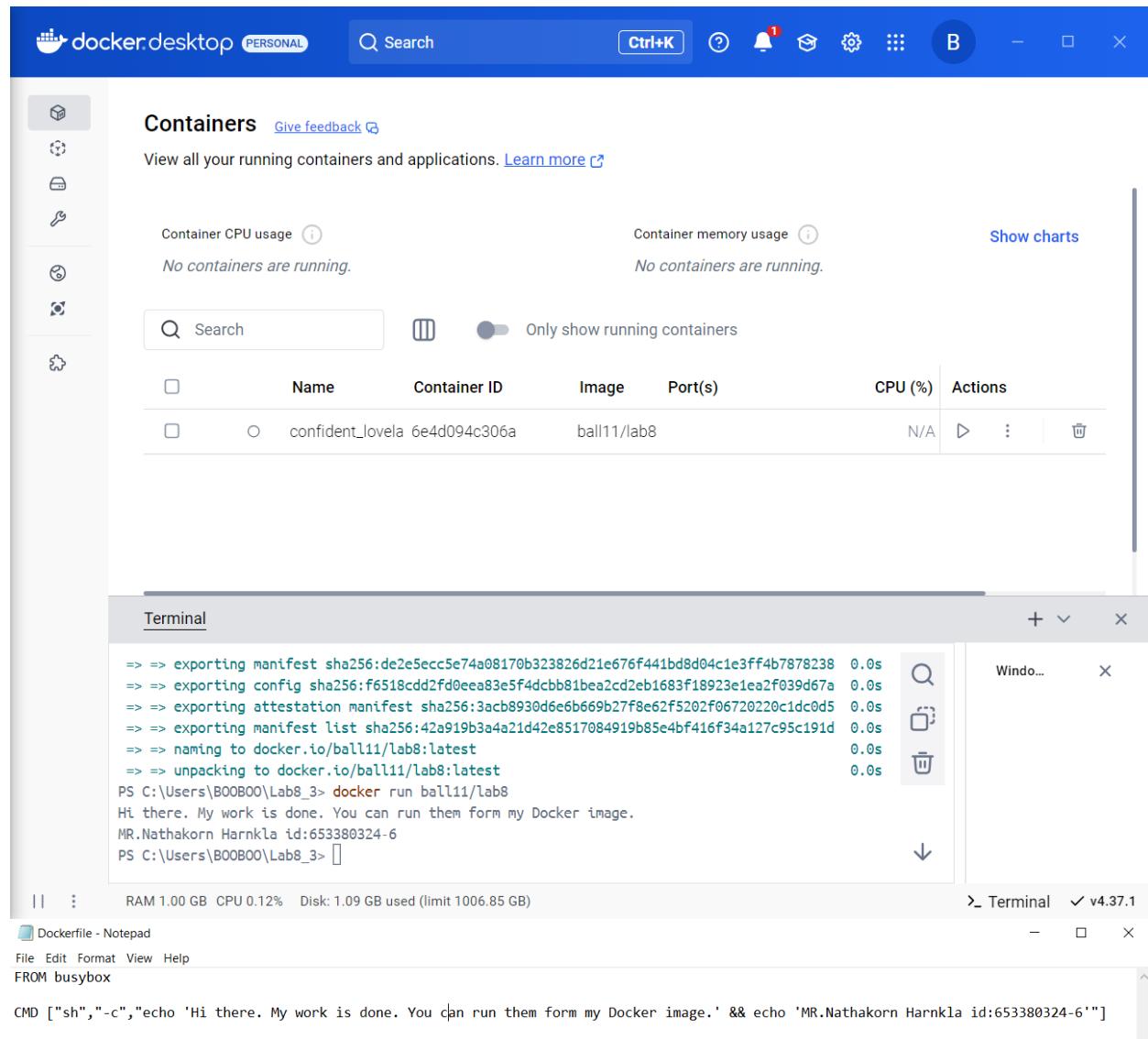
\$ docker build -t <username ที่ลงทะเบียนกับ Docker Hub>/lab8

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



6. ทำการ Push ตัว Docker image ไปไว้บน Docker Hub โดยการใช้คำสั่ง

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

ในการนี้ที่ติดปัญหาไม่ได้ Login ไว้ก่อน ให้ใช้คำสั่งต่อไปนี้ เพื่อ Login ก่อนทำการ Push

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

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

Lab Worksheet

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

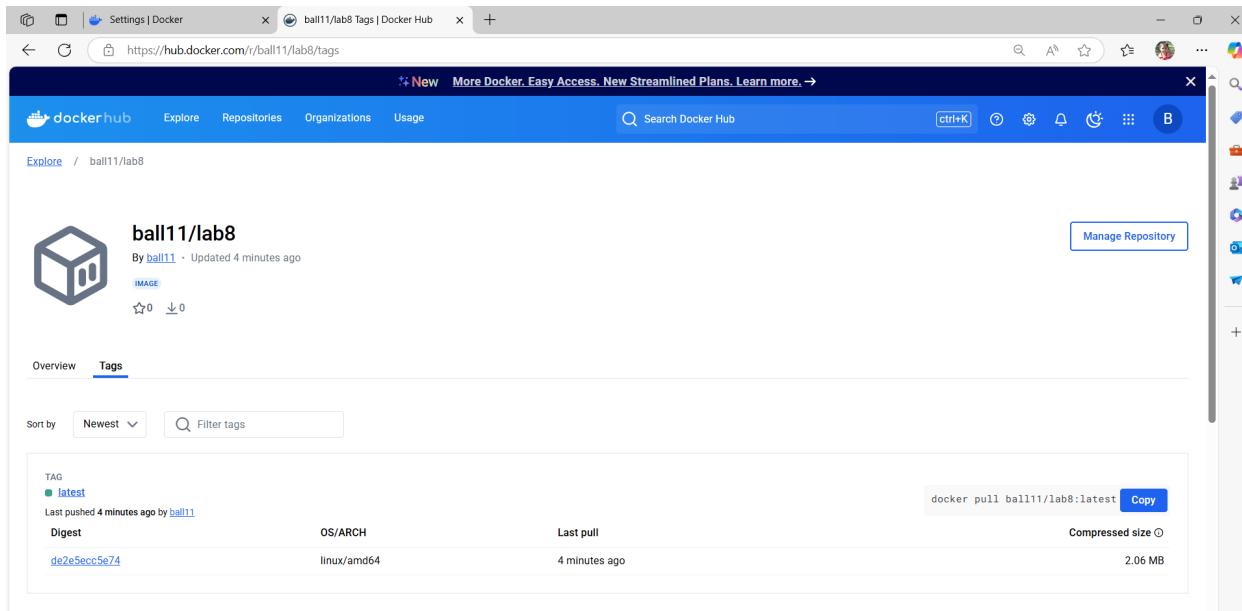
Docker Desktop interface showing a single running container named "confident_lovela" with ID 6e4d094c306a, running the image "ball11/lab8". The CPU usage is N/A.

```
PS C:\Users\B00B00\Lab8_3> docker run ball11/lab8
Hello there. My work is done. You can run them from my Docker image.
MR.Nathakorn Harnkla id:653380324-6
PS C:\Users\B00B00\Lab8_3> docker push ball11/lab8
Using default tag: latest
The push refers to repository [docker.io/ball11/lab8]
b84ea9a666c9: Pushed
9c0abc9c5bd3: Pushed
latest: digest: sha256:42a919b3a4a21d42e851708491b85e4bf416f34a127c95c191de44ade3f509a size: 855
PS C:\Users\B00B00\Lab8_3>
```

RAM 1.03 GB CPU 0.00% Disk: 1.09 GB used (limit 1006.85 GB) Terminal v4.37.1

Docker Hub profile page for user "ball11". The profile shows 1 repository: "ball11/lab8" (updated 3 minutes ago).

Lab Worksheet



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

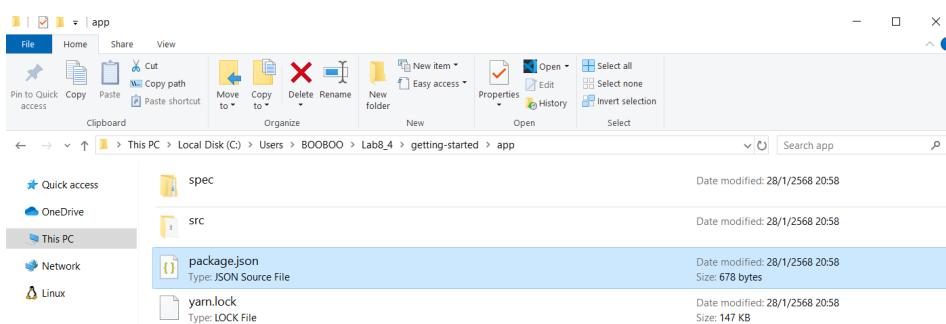
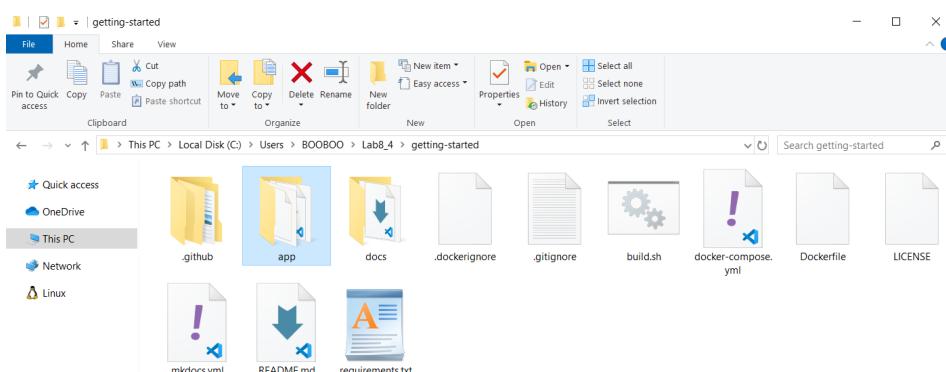
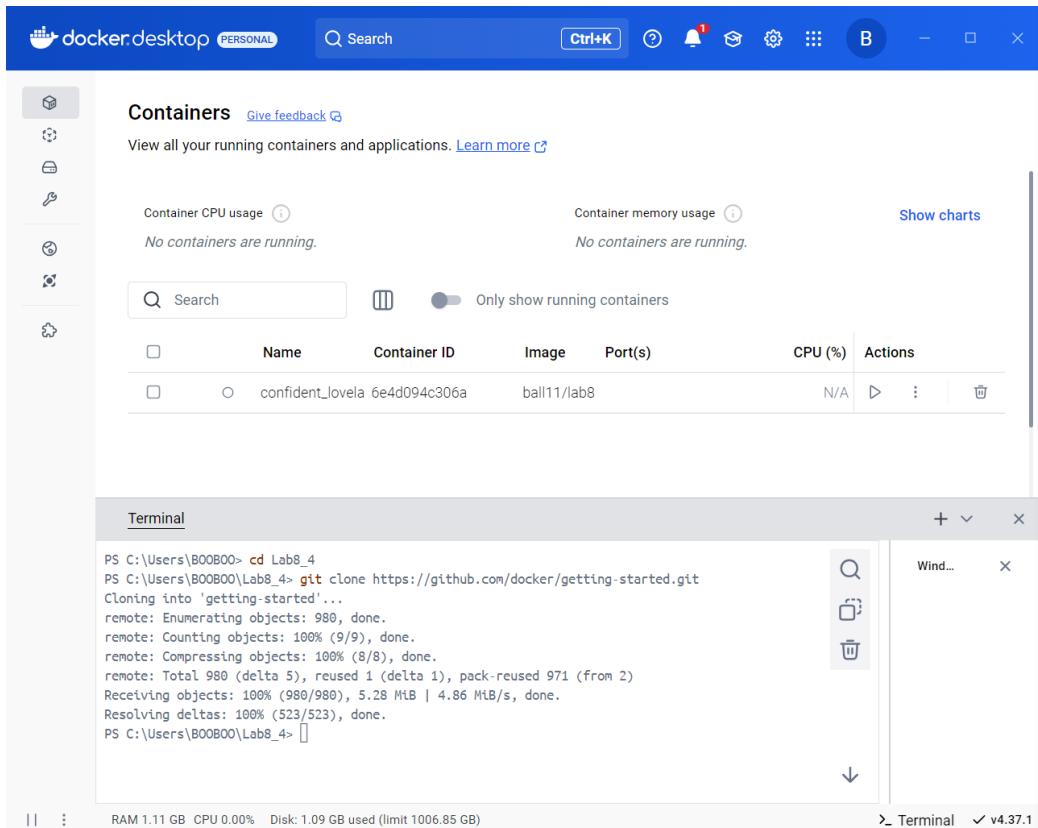
แบบฝึกปฏิบัติที่ 8.4: การ Build และ Update Container image และการ Update แอปพลิเคชัน

1. เปิด Command line หรือ Terminal จากนั้นสร้าง Directory ชื่อ Lab8_4
2. ทำการ Clone ซอฟต์แวร์ Docker Getting Started จาก 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



Lab Worksheet

```

{
  "name": "101-app",
  "version": "1.0.0",
  "main": "index.js",
  "license": "MIT",
  "scripts": {
    "prettify": "prettier -l --write '**/*.js**",
    "test": "jest",
    "dev": "nodemon src/index.js"
  },
  "dependencies": {
    "express": "^4.18.2",
    "mysql2": "^2.3.3",
    "sqlite3": "^5.1.2",
    "uuid": "^3.0.0",
    "wait-port": "^1.0.4"
  },
  "resolutions": {
    "ansi-regex": "5.0.1"
  },
  "prettier": {
    "trailingComma": "all",
    "tabWidth": 4,
    "useTabs": false,
    "semi": true,
    "singleQuote": true
  },
  "devDependencies": {
    "jest": "^29.3.1",
    "nodemon": "^2.0.20",
    "prettier": "^2.7.1"
  }
}

```

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

```

```

Dockerfile - Notepad
File Edit Format View Help
FROM node:18-alpine

WORKDIR /app

COPY .

RUN yarn install --production

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

EXPOSE 3000

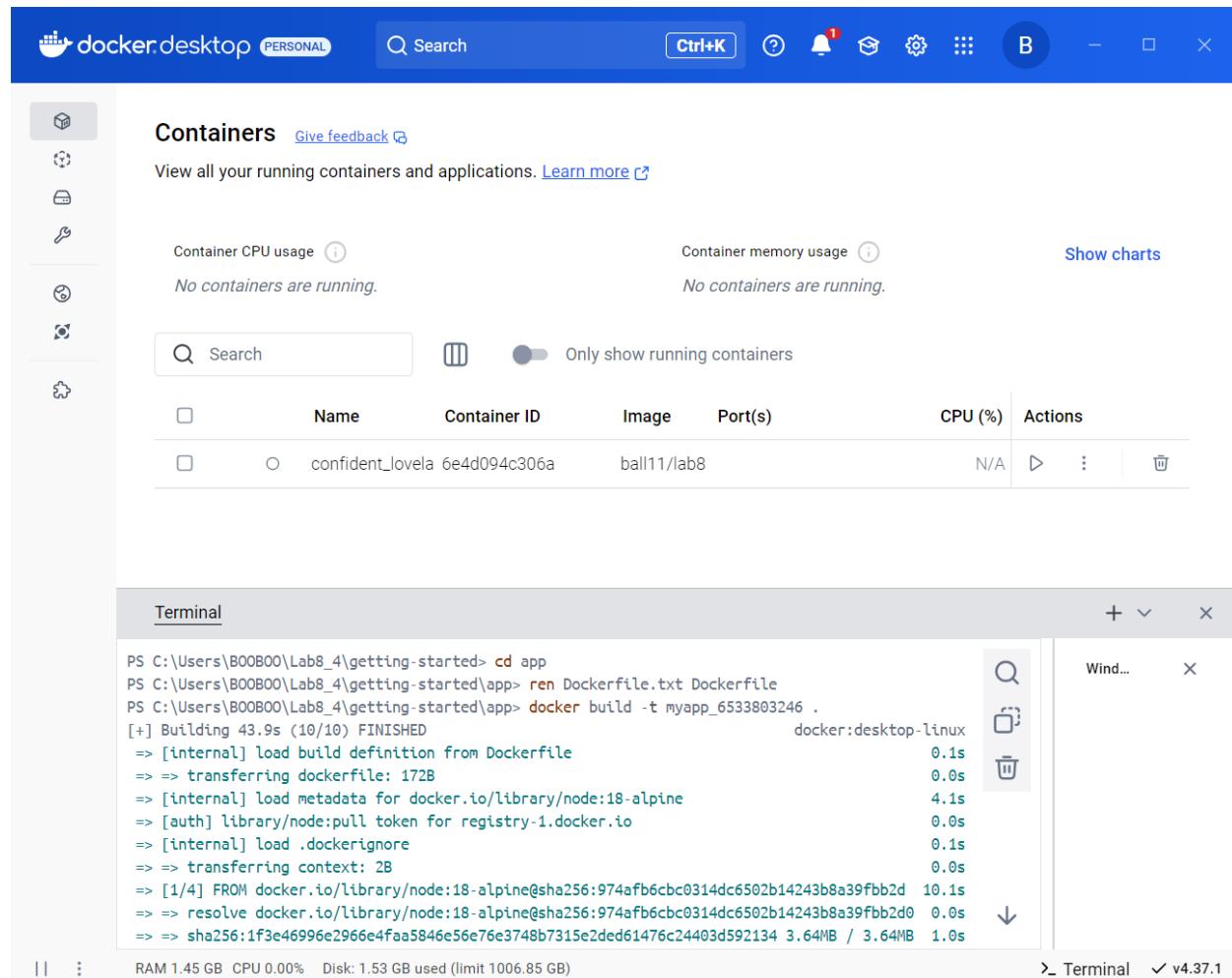
```

Lab Worksheet

5. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้ โดยกำหนดให้ชื่อ image เป็น myapp_รหัส
ศ. ไม่มีจีด

\$ docker build -t <myapp_รหัสศ. ไม่มีจีด> .

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



The screenshot shows the Docker Desktop application interface. The top navigation bar includes 'docker desktop PERSONAL', a search bar, and various icons for settings and notifications. The main area is titled 'Containers' with a sub-instruction 'Give feedback'. It displays CPU and memory usage charts for running containers, both of which show 'No containers are running'. Below this, there is a search bar and a filter option 'Only show running containers'. A table lists one container entry:

	Name	Container ID	Image	Port(s)	CPU (%)	Actions
<input type="checkbox"/>	confident_lovela	6e4d094c306a	ball11/lab8		N/A	▶ ⋮ 🗑

At the bottom of the interface is a 'Terminal' window. The terminal output shows the command to build the Docker image:

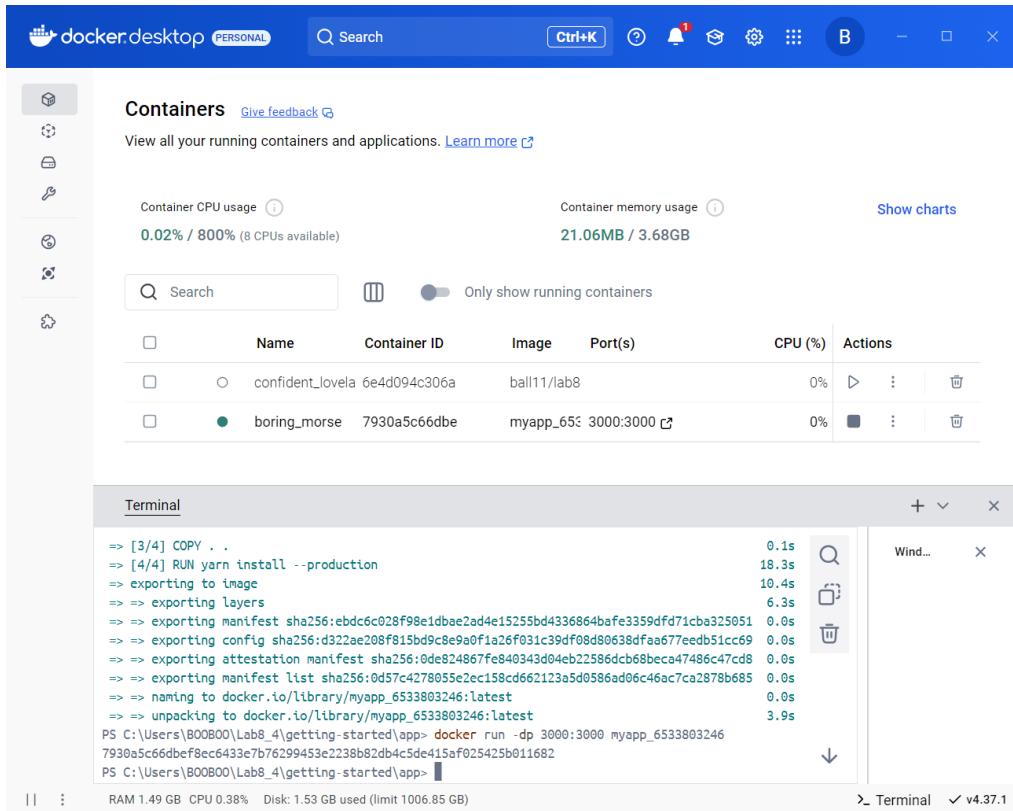
```
PS C:\Users\B00B00\Lab8_4\getting-started> cd app
PS C:\Users\B00B00\Lab8_4\getting-started\app> ren Dockerfile.txt Dockerfile
PS C:\Users\B00B00\Lab8_4\getting-started\app> docker build -t myapp_6533803246 .
[+] Building 43.9s (10/10) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 172B
=> [internal] load metadata for docker.io/library/node:18-alpine
=> [auth] library/node:pull token for registry-1.docker.io
=> [internal] load .dockerrcignore
=> => transferring context: 2B
=> [1/4] FROM docker.io/library/node:18-alpine@sha256:974afb6cbc0314dc6502b14243b8a39fbb2d
=> => resolve docker.io/library/node:18-alpine@sha256:974afb6cbc0314dc6502b14243b8a39fbb2d0
=> => sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c24403d592134 3.64MB / 3.64MB 1.0s
```

The terminal also displays system status at the bottom: RAM 1.45 GB, CPU 0.00%, Disk: 1.53 GB used (limit 1006.85 GB).

Lab Worksheet

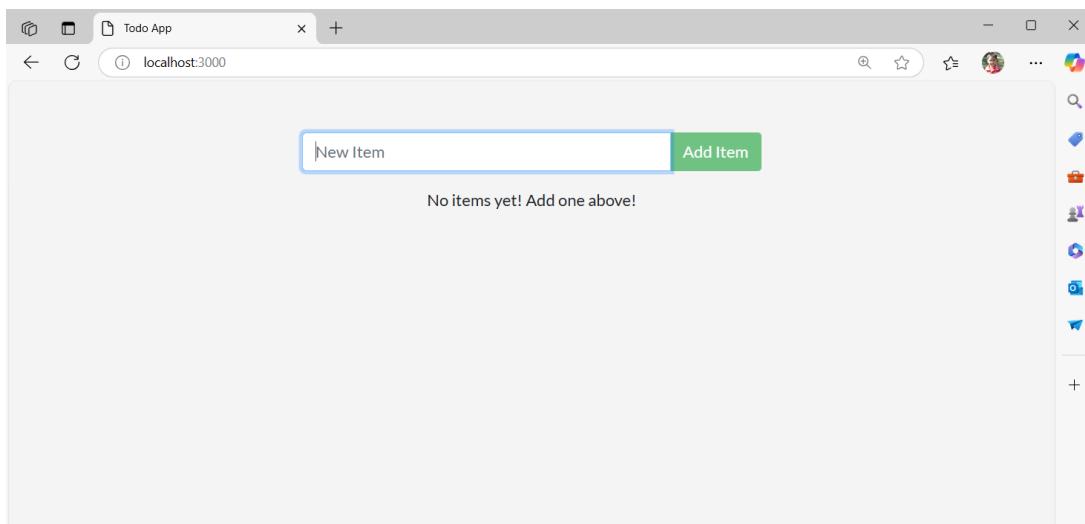
6. ทำการ Start ตัว Container ของแอปพลิเคชันที่สร้างขึ้น โดยใช้คำสั่ง

```
$ docker run -dp 3000:3000 <myapp_รหัสศ.ไม่มีชีด>
```



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

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



Lab Worksheet

หมายเหตุ: นศ.สามารถทดลองเล่น Web application ที่ทำงานอยู่ได้

8. ทำการแก้ไข Source code ของ Web application ดังนี้

- 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.**

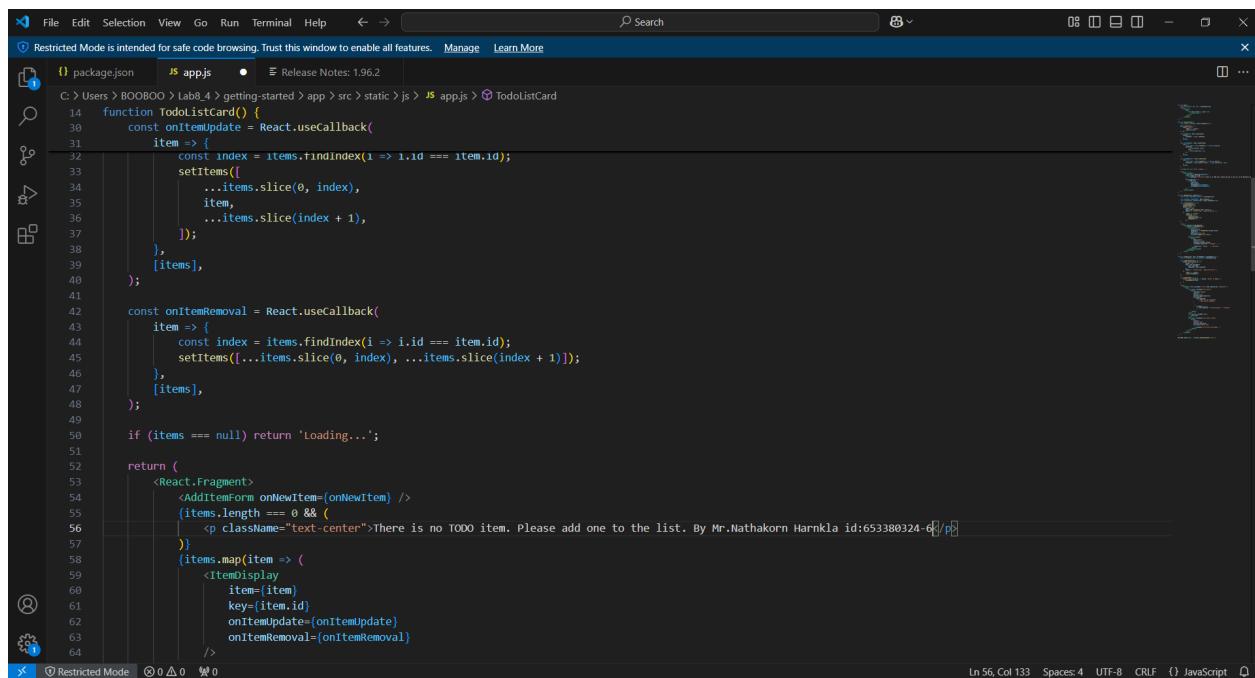
By ชื่อและนามสกุลของนักศึกษา

- Save ไฟล์ให้เรียบร้อย

9. ทำการ Build Docker image โดยใช้คำสั่งเดียวกันกับข้อ 5

10. Start และรัน Container ตัวใหม่ โดยใช้คำสั่งเดียวกันกับข้อ 6

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



```

File Edit Selection View Go Run Terminal Help ⏎ → Search
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
package.json app.js Release Notes: 1.9.6.2
C: > Users > BOOBOO > Lab8_4 > getting-started > app > src > static > js > app.js > TodoListCard
14 function TodoListCard() {
30   const onItemUpdate = React.useCallback(
31     item => {
32       const index = items.findIndex(i => i.id === item.id);
33       setItems([
34         ...items.slice(0, index),
35         item,
36         ...items.slice(index + 1),
37       ]);
38     },
39     [items],
40   );
41
42   const onItemRemoval = React.useCallback(
43     item => {
44       const index = items.findIndex(i => i.id === item.id);
45       setItems([
46         ...items.slice(0, index),
47         ...items.slice(index + 1),
48       ]);
49     },
50   );
51
52   if (items === null) return 'Loading...';
53
54   return (
55     <React.Fragment>
56       <AddItemForm onNewItem={onNewItem} />
57       {items.length === 0 && (
58         <p className="text-center">There is no TODO item. Please add one to the list. By Mr.Nathakorn Harnkla id:653380324-6</p>
59       )}
60       {items.map(item => (
61         <ItemDisplay
62           item={item}
63           key={item.id}
64           onItemUpdate={onItemUpdate}
65           onItemRemoval={onItemRemoval}
66         />
67       )}
68     </React.Fragment>
69   );
70 }
71
72 export default TodoListCard;

```

Ln 56, Col 133 Spaces: 4 UTF-8 CRLF {} JavaScript

Lab Worksheet

The screenshot shows the Docker Desktop interface. On the left, there's a sidebar with icons for creating a new container, portainer, secrets, volumes, and environment variables. The main area is titled "Containers" with a "Search" bar and a toggle switch for "Only show running containers". A table lists four containers:

	Name	Container ID	Image	Port(s)	CPU (%)	Start	Logs
<input type="checkbox"/>	boring_morse	7930a5c66dbe	myapp_653	3000:3000	0%		
<input type="checkbox"/>	goofy_bhabha	74b4f18b0cc4	myapp_653	3000:3000	0%		
<input type="checkbox"/>	reverent_kapitsa	db838181dc34	myapp_653	3000:3000	0%		
<input type="checkbox"/>	competent_dris	75c2ee53eacea7f2b32dd7afcecc5e216eaf6b122425765407878403db0201f6	myapp_653	3000:3000	0%		

Below the table, it says "Showing 5 items". At the bottom, there's a terminal window with the following command history:

```

PS C:\Users\B00B00\Lab8_4\getting-started\app> docker run -dp 3000:3000 myapp_6533803246
74b4f18b0cc4b7c25104f0c21ac0e3d5c61dbb76e7300e662a9512303b2623ba
docker: Error response from daemon: driver failed programming external connectivity on endpoint goof_y_bhabha (d0b052b4870d52042b8a8c1860be0873fdd47b8c82b20c379b915908a252e76b): Bind for 0.0.0.0:3000
failed: port is already allocated.
PS C:\Users\B00B00\Lab8_4\getting-started\app> docker run -dp 3000:3000 myapp_6533803246
db838181dc34e99cbc04a056cd1f331bf7639be1fb7c170a4279ec27dd7b96f
docker: Error response from daemon: driver failed programming external connectivity on endpoint reverent_kapitsa (0893f56e8f946861d8737d8caa822dd8213a09db1d41c08524bbfdb3b9c1d7c7): Bind for 0.0.0.0:
3000 failed: port is already allocated.
PS C:\Users\B00B00\Lab8_4\getting-started\app> docker run -dp 3000:3000 myapp_6533803246
75c2ee53eacea7f2b32dd7afcecc5e216eaf6b122425765407878403db0201f6
PS C:\Users\B00B00\Lab8_4\getting-started\app>

```

The terminal also shows system information: RAM 1.82 GB, CPU 0.12%, Disk: 1.79 GB used (limit 1006.85 GB). The status bar at the bottom right indicates "Terminal ✓ v4.37.1".

The screenshot shows a browser window titled "Todo App" with the URL "localhost:3000". The page displays a simple form with a text input field labeled "New Item" and a green "Add Item" button. Below the form, a message reads: "There is no TODO item. Please add one to the list. By Mr.Nathakorn Harnkla id:653380324-6". To the right of the browser window, there's a vertical sidebar with various icons for file operations like copy, paste, and delete, as well as other application-related functions.

Lab Worksheet

(1) Error ที่เกิดขึ้นหมายความอย่างไร และเกิดขึ้นเพราะอะไร

ตอบ เกิดจากการที่ Container ของ browser ก่อนหน้านี้ทำงานอยู่

11. ลบ Container ของ Web application เวอร์ชันก่อนแก้ไขออกจากระบบ โดยใช้วิธีใดวิธีหนึ่งดังต่อไปนี้

a. ผ่าน Command line interface

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

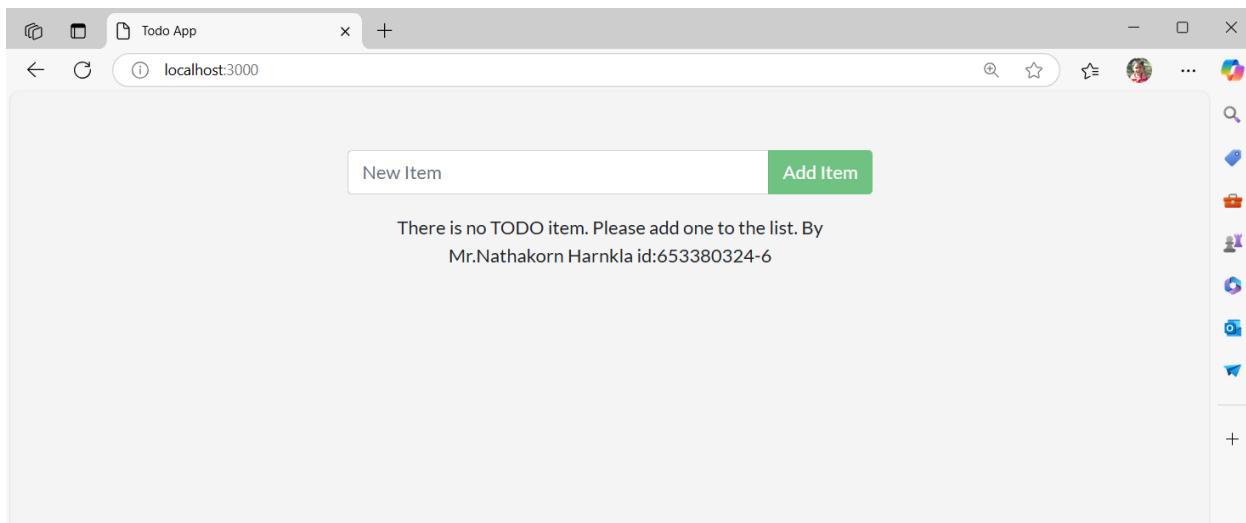
b. ผ่าน Docker desktop

- ไปที่หน้าต่าง Containers
- เลือกไอคอนถังขยะในແຄວของ Container ที่ต้องการจะลบ
- ยืนยันโดยการกด Delete forever

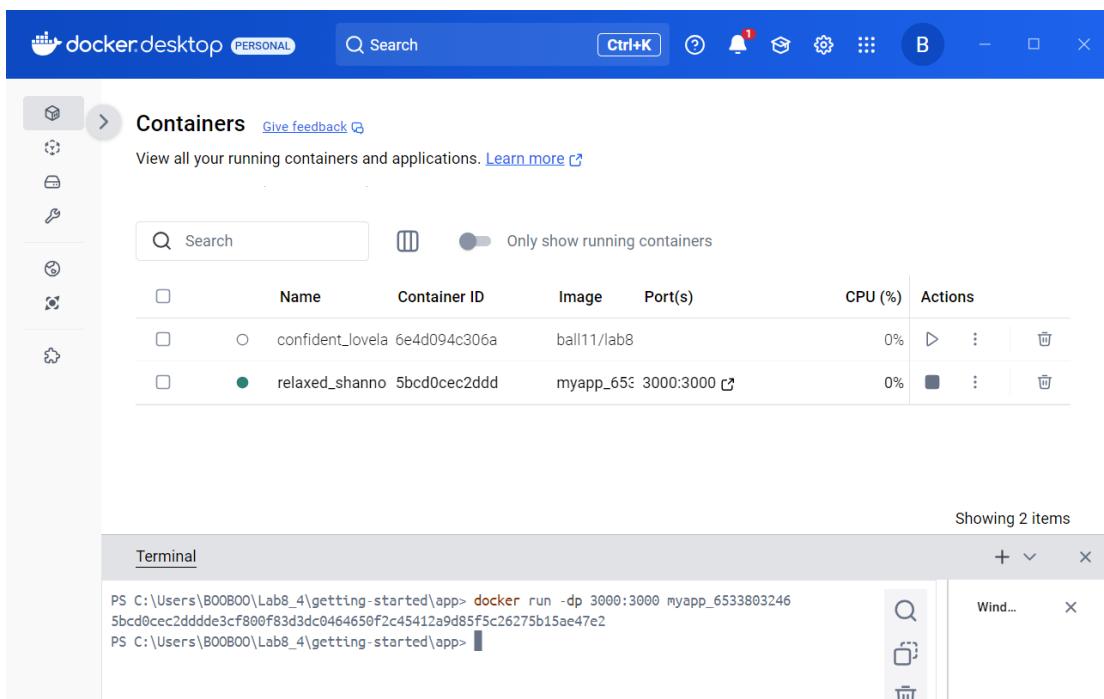
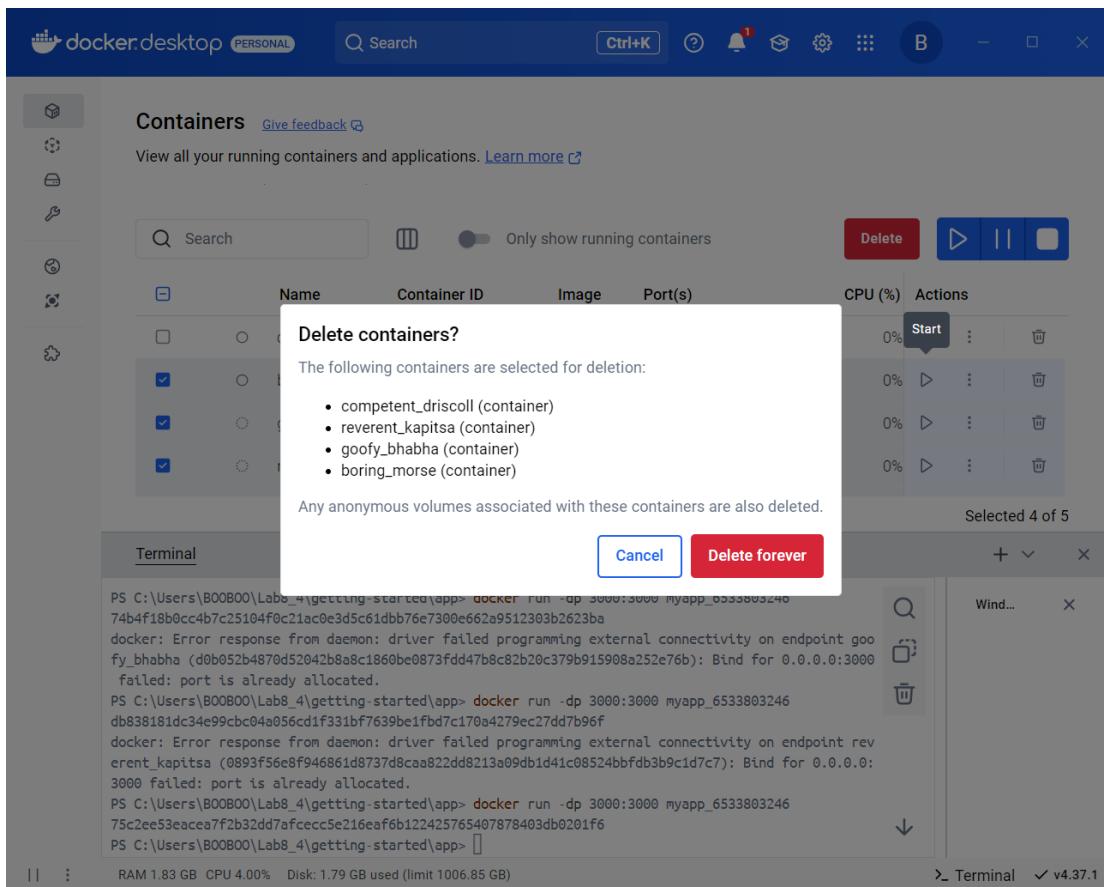
12. Start และรัน Container ตัวใหม่อีกรอบ โดยใช้คำสั่งเดียวกับข้อ 6

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

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



Lab Worksheet



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

The screenshot shows the Docker Desktop interface. The top bar includes the Docker logo, a search field, and various icons. The main area is titled 'Containers' with a 'Give feedback' link. It displays three running containers:

Name	Container ID	Image	Port(s)	CPU (%)	Actions
confident_lovela	6e4d094c306a	ball11/lab8		0%	▶ ⋮ trash
relaxed_shanno	5bcd0cec2ddd	myapp_653	3000:3000	0%	■ ⋮ trash
youthful_ramanl	c92ae73a8bcd	jenkins/jen	50000:50000	0.23%	■ ⋮ Show all ports (2) trash

Below this is a 'Terminal' window titled 'Terminal'. It contains the following text:

```
*****
Jenkins initial setup is required. An admin user has been created and a password generated.
Please use the following password to proceed to installation:
4bd621b8bdd24d528d2f6c05619924df
This may also be found at: /var/jenkins_home/secrets/initialAdminPassword
*****
```

The bottom status bar shows system information: RAM 2.72 GB, CPU 0.00%, Disk: 2.52 GB used (limit 1006.85 GB), and the terminal version v4.37.1.

4. เมื่อได้รับการยืนยันว่า Jenkins is fully up and running ให้เปิดเบราว์เซอร์ และป้อนที่อยู่เป็น localhost:8080

Lab Worksheet

5. ทำการ Unlock Jenkins ด้วยรหัสผ่านที่ได้ในข้อที่ 3
 6. สร้าง Admin User โดยใช้ username เป็นชื่อจริงของนักศึกษาพร้อมรหัสสี่ตัวท้าย เช่น somsri_3062
- [Check point#13] Capture หน้าจอที่แสดงผลการตั้งค่า**

The image contains two screenshots of the Jenkins Setup Wizard interface. Both screenshots are titled 'Getting Started'.

Screenshot 1: Create First Admin User

- Username: Nathakorn_324
- Password: (Redacted)
- Confirm password: (Redacted)
- Full name: Nathakorn Harnka
- Buttons at the bottom: 'Skip and continue as admin' and 'Save and Continue'

Screenshot 2: Instance Configuration

- Jenkins URL: http://localhost:8080/lab8
- Description below URL: 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.
- Note below description: 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.
- Buttons at the bottom: 'Not now' and 'Save and Finish'

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

Lab Worksheet

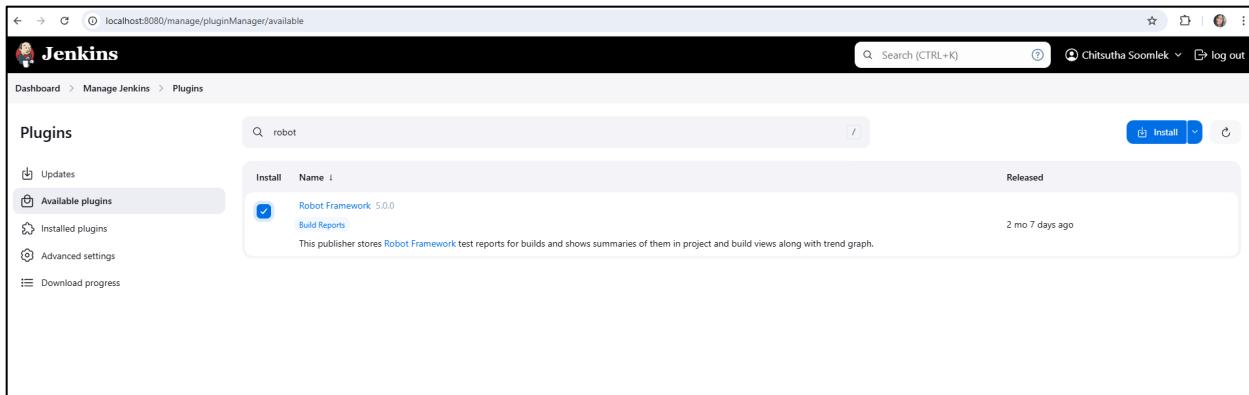
The screenshot shows the Jenkins dashboard at localhost:8080. The main header says "Jenkins". The left sidebar has links for "New Item", "Build History", "Manage Jenkins", and "My Views". A "Build Queue" section shows "No builds in the queue." Below it is a "Build Executor Status" section with "0/2". The central area has a "Welcome to Jenkins!" message: "This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project." It includes buttons for "Create a job" (with a plus sign), "Set up an agent" (with a monitor icon), "Configure a cloud" (with a cloud icon), and "Learn more about distributed builds" (with a question mark icon). At the bottom right, it says "REST API Jenkins 2.479.3".

9. เลือก Manage Jenkins แล้วไปที่เมนู Plugins

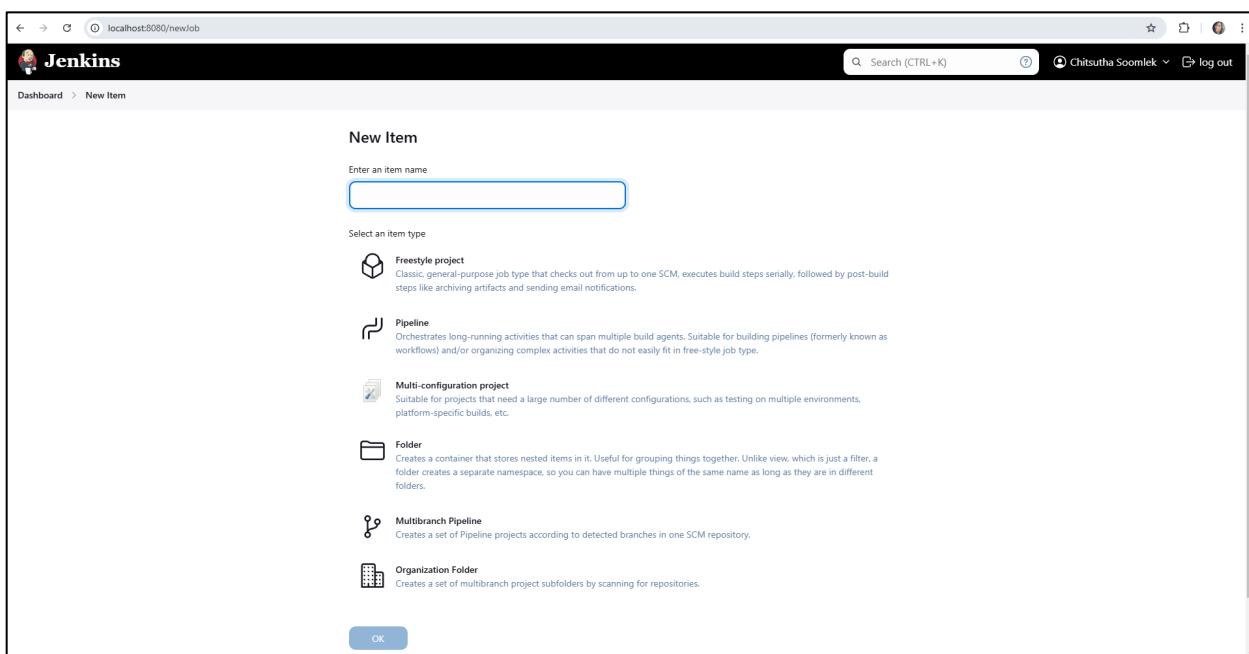
The screenshot shows the Jenkins "Manage Jenkins" page at localhost:8080/manage/. The top navigation bar includes "Dashboard", "Manage Jenkins", "Search (CTRL+K)", and "Chitsutha Soomlek". The left sidebar has links for "New Item", "Build History", "Manage Jenkins" (which is selected and highlighted in grey), and "My Views". A "Build Queue" section shows "No builds in the queue." Below it is a "Build Executor Status" section with "0/2". The main content area has several sections: "System Configuration" (with icons for System, Tools, Plugins, Nodes, Clouds, Appearance), "Security" (with icons for Security, Credentials, Credential Providers, Users), "Status Information" (with icons for System Information, System Log, Load Statistics, and About Jenkins). A red banner at the top right says "It appears that your reverse proxy set up is broken." with "More Info" and "Dismiss" buttons.

Lab Worksheet

10. ไปที่เมนู Available plugins แล้วเลือกติดตั้ง Robotframework เพิ่มเติม



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



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

Description: Lab 8.5

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

Build Trigger: เลือกแบบ Build periodically และกำหนดให้ build ทุก 15 นาที

Lab Worksheet

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

[Check point#14] Capture หน้าจอแสดงการตั้งค่า พร้อมกับตอบคำถามต่อไปนี้

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

ตอบ robot valid_login.robot, robot invalid_login.robot

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

13. กด Apply และ Save

14. สั่ง Build Now

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

Lab Worksheet

Jenkins

Dashboard > UAT >

UAT

Lab 8.5

Latest Robot Results:
No results available yet.

Permalinks

- Last build (#2), 36 sec ago
- Last failed build (#2), 36 sec ago
- Last unsuccessful build (#2), 36 sec ago
- Last completed build (#2), 36 sec ago

Builds

S	Build	Time Since	Duration
①	#5	2.7 sec	16 ms
②	#4	4.4 sec	22 ms
③	#3	18 sec	25 ms
④	#2	2 min 51 sec	19 ms
⑤	#1	3 min 11 sec	0.12 sec

Jenkins

Dashboard > UAT > Build Time Trend

Build Time Trend

S	Build	Time Since	Duration
①	#5	2.7 sec	16 ms
②	#4	4.4 sec	22 ms
③	#3	18 sec	25 ms
④	#2	2 min 51 sec	19 ms
⑤	#1	3 min 11 sec	0.12 sec

Icon: S M L

Builds

S	Build	Time Since	Duration
①	#5	2.7 sec	16 ms
②	#4	4.4 sec	22 ms
③	#3	18 sec	25 ms
④	#2	2 min 51 sec	19 ms
⑤	#1	3 min 11 sec	0.12 sec