

Lab Worksheet

ชื่อ-นามสกุล นายบดินทร์ แสนสุข รหัสนักศึกษา 653380135-9 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

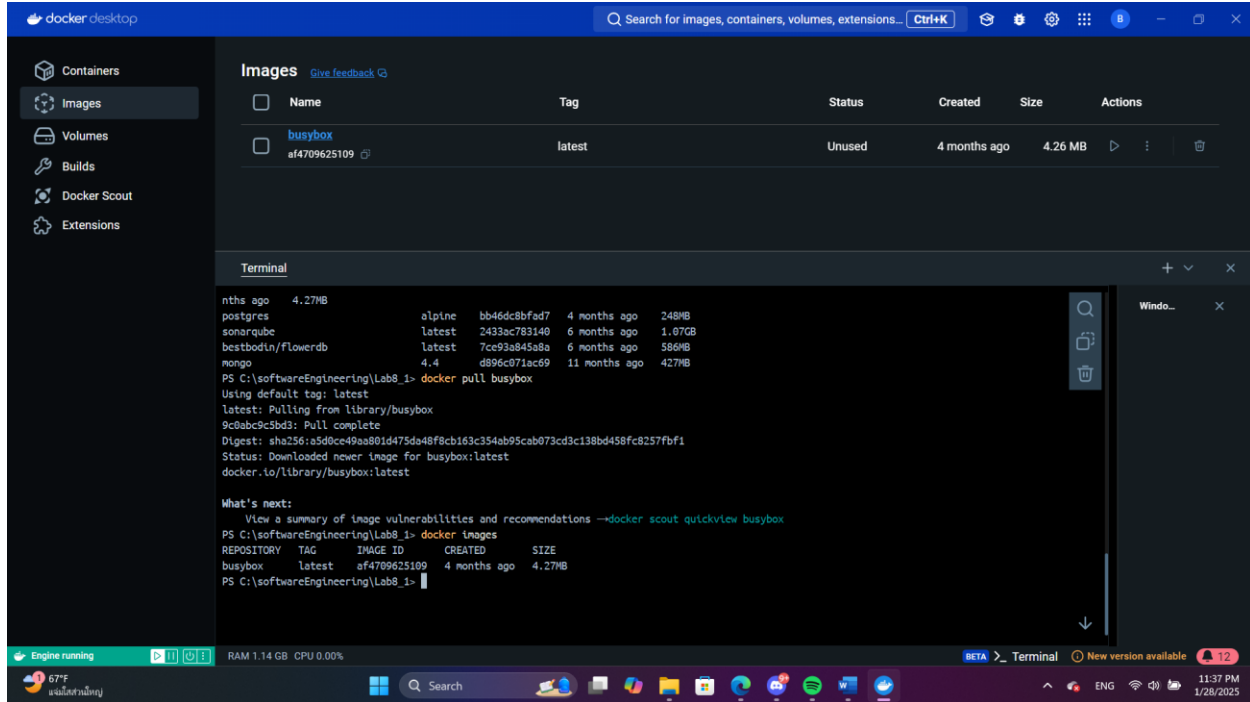
1. ติดตั้ง Docker desktop ลงบนเครื่องคอมพิวเตอร์ โดยดาวน์โหลดจาก <https://www.docker.com/get-started>
2. สร้าง Account บน Docker hub (<https://hub.docker.com/signup>)
3. กำหนดให้ \$ หมายถึง Command prompt และ <> หมายถึง ให้ป้อนค่าของพารามิเตอร์ที่กำหนด

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

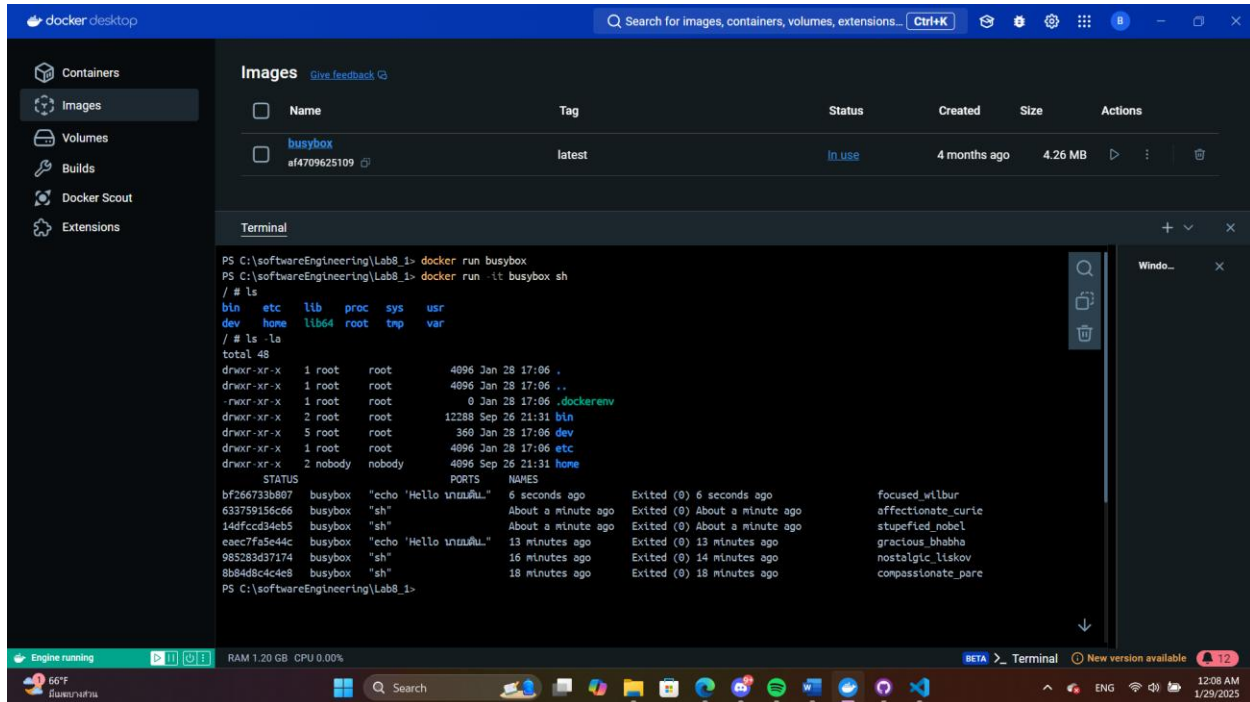


- (1) สิ่งที่อยู่ภายใต้คอลัมน์ Repository คืออะไร **ตอบ** images ที่มีชื่อว่า “busybox”
- (2) Tag ที่ใช้บอกถึงอะไร **ตอบ** TAG มีชื่อว่า latest บอกว่าเป็น tag เริ่มต้น ใช้เมื่อไม่ได้ระบุ tag โดยเฉพาะในการ Pull หรือ Run image จาก Repository

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

ตอบ สามารถ Run container แบบ Interactive และยังสามารถเปิด Shell ได้อีกด้วย

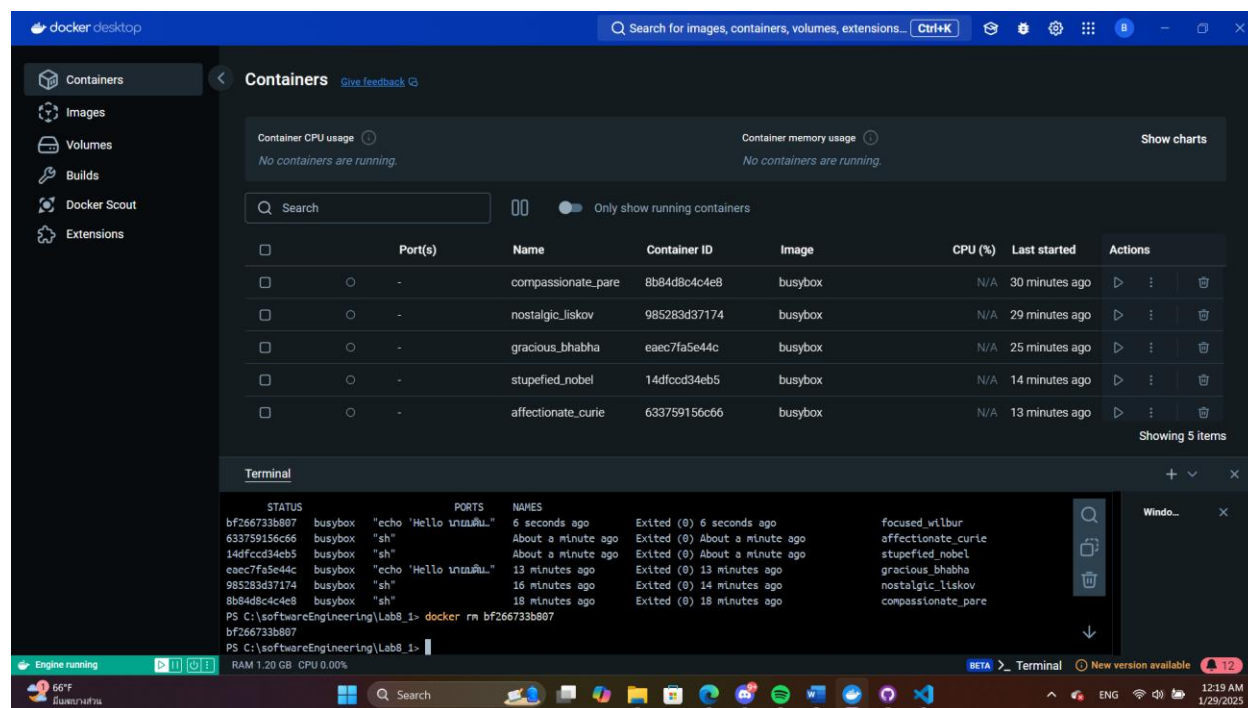
(2) คอลัมน์ STATUS จากการรันคำสั่ง docker ps -a แสดงถึงข้อมูลอะไร

ตอบ คอลัมน์ STATUS จากคำสั่ง docker ps -a แสดงสถานะของ container แต่ละตัว โดยบอกว่า container นั้นกำลังทำงานอยู่หรือหยุดทำงานแล้ว พร้อมด้วยเวลาที่เกี่ยวข้อง

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

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

Lab Worksheet



จะพบว่า ไม่มี container ID ของ bf266733b807 อีกแล้ว เนื่องจากได้ลบออกไปแล้ว

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

Lab Worksheet

FROM busybox

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

CMD echo "ชื่อ-นามสกุล รหัสนักศึกษา ชื่อเล่น"

EOF

หรือใช้คำสั่ง

\$ touch Dockerfile

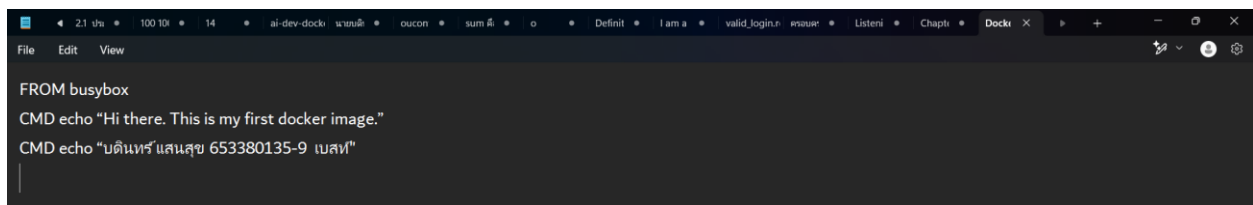
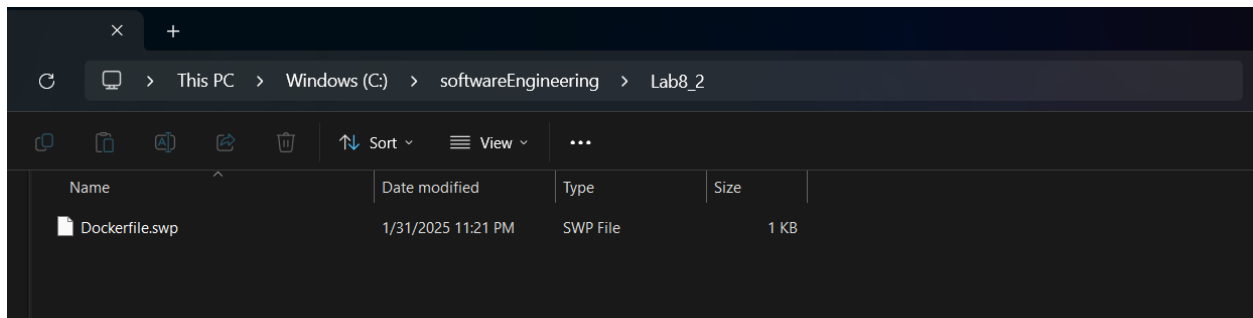
แล้วใช้ Text Editor ในการใส่เนื้อหาแทน

5. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้

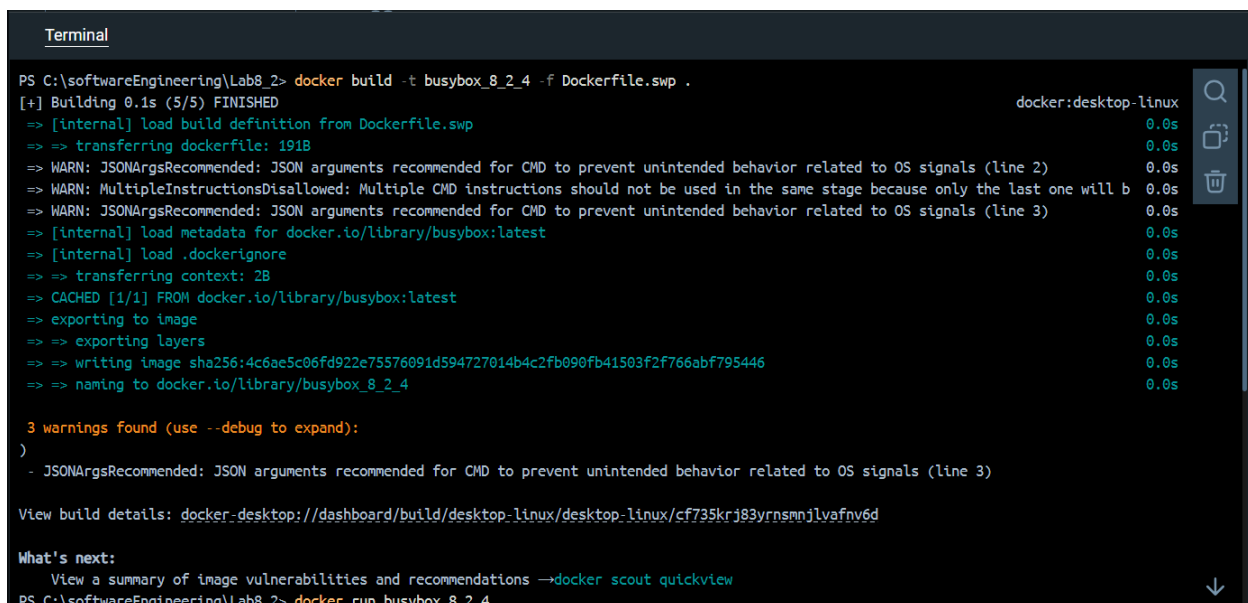
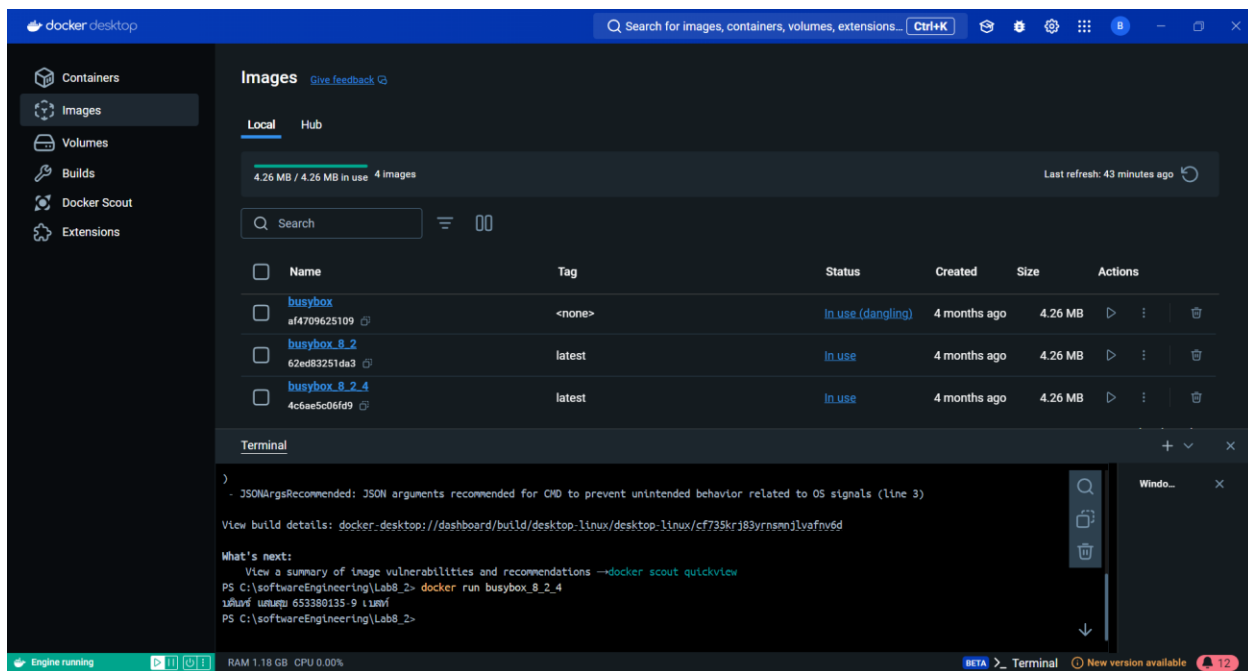
\$ docker build -t <ชื่อ Image> .

6. เมื่อ Build สำเร็จแล้ว ให้ทำการรัน Docker image ที่สร้างขึ้นในขั้นตอนที่ 5

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



Lab Worksheet



(1) คำสั่งที่ใช้ในการ run คือ

ตอบ docker run busybox_8_2_4

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

Lab Worksheet

ตอบ -t ใช้สำหรับกำหนดชื่อ (tag) ให้กับ Docker image เพื่อให้ง่ายต่อการอ้างอิงและจัดการ เพราะหากไม่ใช้จะต้องใช้ Docker ID ซึ่งจำยาก และยาต่อการเรียกใช้

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

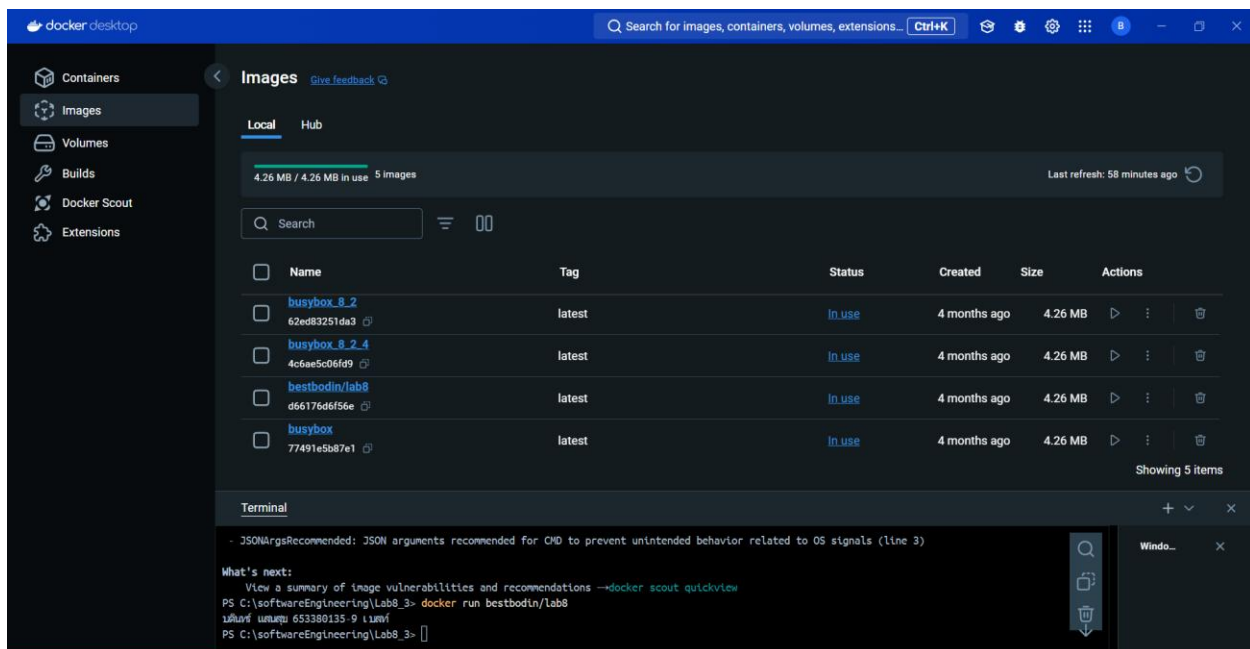
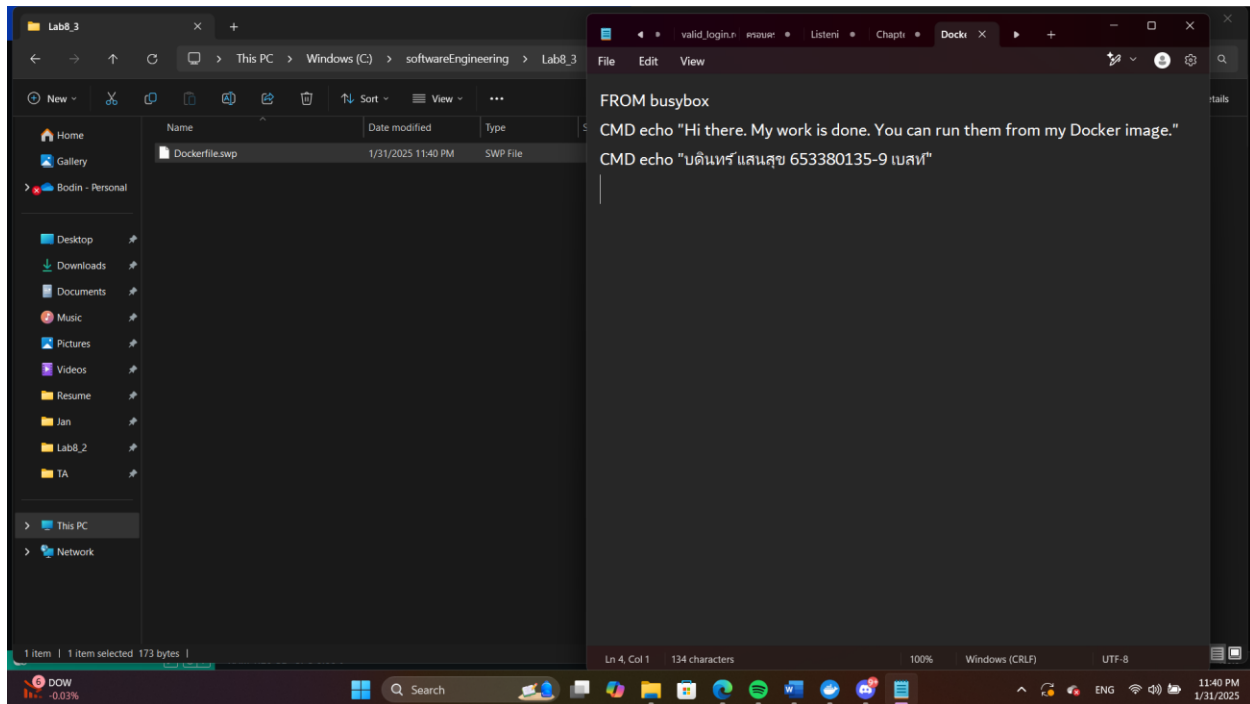
```
$ docker build -t <username ที่ลงทะเบียนกับ Docker Hub>/lab8
```

5. ทำการรัน Docker image บน Container ในเครื่องของตัวเองเพื่อทดสอบผลลัพธ์ ด้วยคำสั่ง

Lab Worksheet

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

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



Lab Worksheet

```

Terminal
PS C:\softwareEngineering> mkdir Lab8_3

Directory: C:\softwareEngineering

Mode                LastWriteTime         Length Name
----                -
d-----          1/31/2025  11:37 PM             Lab8_3

PS C:\softwareEngineering> cd Lab8_3
PS C:\softwareEngineering\Lab8_3> docker build -t bestbodin/lab8 -f Dockerfile.swp
ERROR: "docker buildx build" requires exactly 1 argument.
See 'docker buildx build --help'.

Usage: docker buildx build [OPTIONS] PATH | URL | -

Start a build
PS C:\softwareEngineering\Lab8_3> docker build -t bestbodin/lab8 -f Dockerfile.swp .
[+] Building 0.2s (5/5) FINISHED                                docker:desktop-linux
=> [internal] load build definition from Dockerfile.swp        0.0s
- JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 3)

What's next:

```

```

Terminal
- JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 3)

What's next:
View a summary of image vulnerabilities and recommendations →docker scout quickview
PS C:\softwareEngineering\Lab8_3> docker run bestbodin/lab8
binfmt_misc 653380135-9 เสร็จ
PS C:\softwareEngineering\Lab8_3>

```

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

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

Lab Worksheet

Terminal

```
PS C:\softwareEngineering\Lab8_3> docker push bestbodin/lab8
Using default tag: latest
The push refers to repository [docker.io/bestbodin/lab8]
59654b79daad: Mounted from library/busybox
latest: digest: sha256:ddf5f2e67dff4102c3532628072a2b290c957ab6d5628bb94ce0c41b2295f05b size: 528
PS C:\softwareEngineering\Lab8_3> docker login -u bestbodin -p bodin_1234
WARNING! Using --password via the CLI is insecure. Use --password-stdin.
Login Succeeded
PS C:\softwareEngineering\Lab8_3>
```

bestbodin/lab8

Last pushed 8 minutes ago

Add a description [✎](#)

Add a category [✎](#)

General **Tags** Builds Collaborators Webhooks Settings

Sort by **Newest** [Filter tags](#) [Delete](#)

TAG	Digest	OS/ARCH	Last pull	Compressed size
latest	ddf5f2e67dff	linux/amd64	---	2.06 MB

Docker commands

To push a new tag to this repository:

```
docker push bestbodin/lab8:tagname
```

[docker pull bestbodin/lab8:latest](#) [Copy](#)

<https://hub.docker.com/repository/docker/bestbodin/lab8/tags>

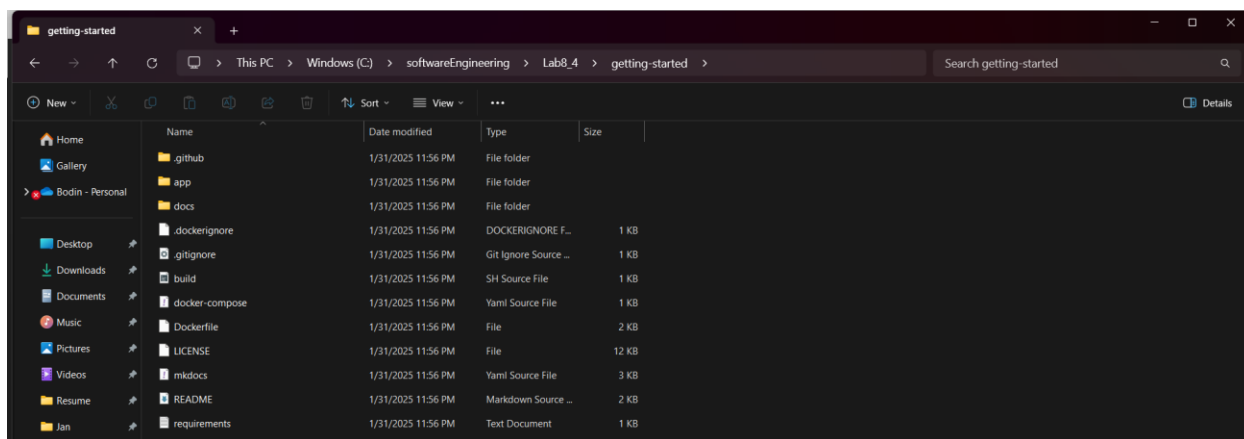
Lab Worksheet

แบบฝึกปฏิบัติที่ 8.4: การ Build แอปพลิเคชันจาก Container image และการ Update แอปพลิเคชัน

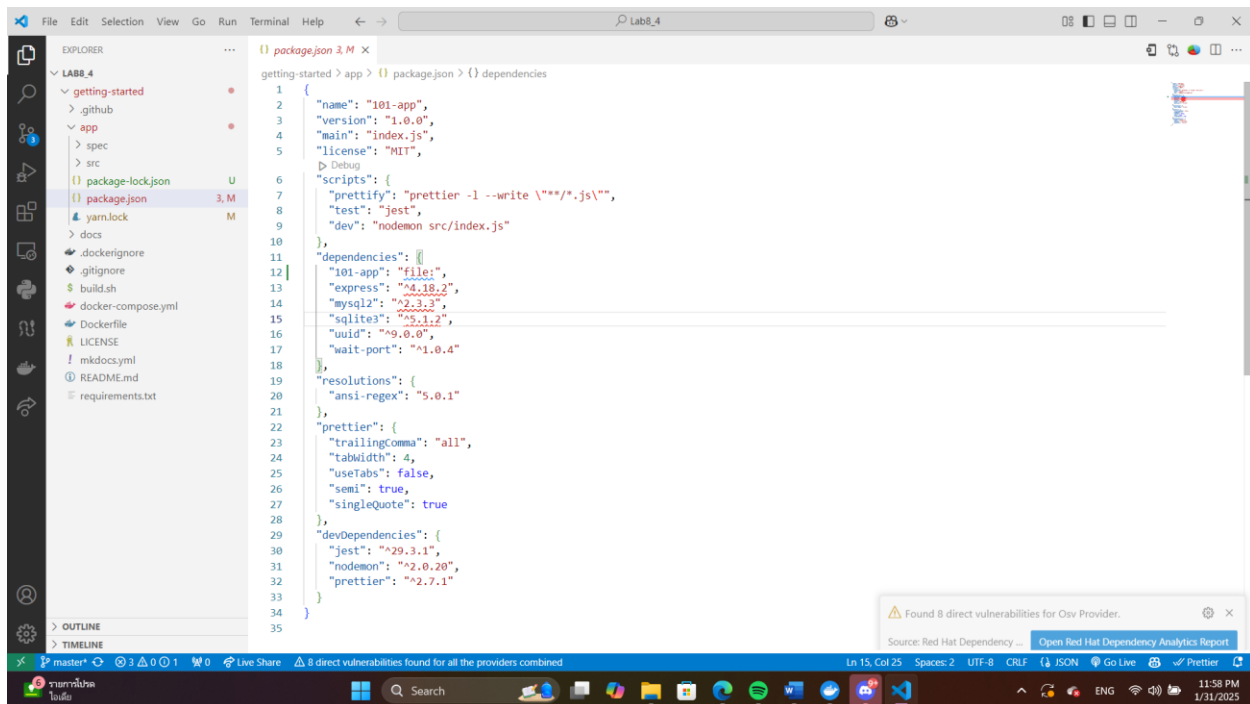
1. เปิด Command line หรือ Terminal จากนั้นสร้าง Directory ชื่อ Lab8_4
2. ทำการ Clone ซอร์สโค้ดของเว็บแอปพลิเคชันจาก GitHub repository
<https://github.com/docker/getting-started.git> ลงใน Directory ที่สร้างขึ้น โดยใช้คำสั่ง

```
$ 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



ท่อยุ่



4. ภายใต้ getting-started/app ให้สร้าง Dockerfile พร้อมกับใส่เนื้อหาดังต่อไปนี้ลงไปไฟล์
FROM node:18-alpine
WORKDIR /app
COPY . .

Lab Worksheet

RUN yarn install --production

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

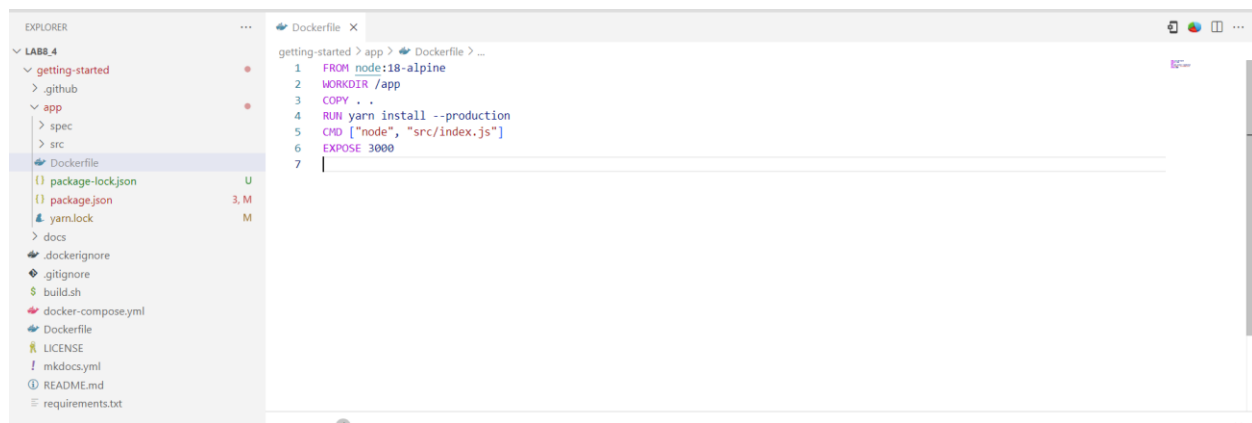
EXPOSE 3000

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

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

[Check point#8] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง)

แสดงคำสั่งและผลลัพธ์ที่ได้ทางหน้าจอ



Lab Worksheet

The screenshot shows a Dockerfile editor with the following content:

```

getting-started > app > Dockerfile > ...
1 FROM node:18-alpine
2 WORKDIR /app
3 COPY . .
4 RUN yarn install --production
5 CMD ["node", "src/index.js"]
6 EXPOSE 3000

```

Below the editor is a terminal window showing the output of the command `docker build -t myapp_6533801359 .`. The output shows the build process, including the resolution of the base image, the copying of the Dockerfile and source code, and the execution of the `yarn install --production` command. The build is successful, and the image is exported.

```

PS C:\softwareEngineering\Lab8_4\getting-started\app> docker build -t myapp_6533801359 .
[+] Building 21.5s (10/10) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 156B
=> [internal] load metadata for docker.io/library/node:18-alpine
=> [auth] library/node:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/4] FROM docker.io/library/node:18-alpine@sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d975e9059dd066be3e274fbb25
=> => resolve docker.io/library/node:18-alpine@sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d975e9059dd066be3e274fbb25
=> => sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e0da02525e63167c 40.01MB / 40.01MB
=> => sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15bf426d01bc1 1.26MB / 1.26MB
=> => sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d975e9059dd066be3e274fbb25 7.67kB / 7.67kB
=> => sha256:6e804119c3884fc5782795bf0d2adc89201c63105aace8647b17a7bcebbc385e 1.72kB / 1.72kB
=> => sha256:dcbf7b337595be6f4d214e4eed84f230eefe0e4ac03a50380d573e289b9e5e40 6.18kB / 6.18kB
=> => sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c24403d592134f0 3.64MB / 3.64MB
=> => sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce368990ca771 444B / 444B
=> => extracting sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c24403d592134f0
=> => extracting sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e0da02525e63167c
=> => extracting sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15bf426d01bc1
=> => extracting sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce368990ca771
=> [internal] load build context
=> => transferring context: 4.82MB
=> [2/4] WORKDIR /app
=> [3/4] COPY . .
=> [4/4] RUN yarn install --production
=> [3/4] COPY . .
=> [4/4] RUN yarn install --production
=> exporting to image
=> [3/4] COPY . .

```

At the bottom of the terminal window, there is a status bar that reads: "RHDA analysis has failed".

Lab Worksheet

The screenshot shows the Docker Desktop interface. The top pane displays the Dockerfile for 'getting-started > app' with the following content:

```

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

```

The bottom pane shows the 'TERMINAL' tab with the output of the build process:

```

=> [3/4] COPY . . 0.1s
=> [4/4] RUN yarn install --production 10.8s
=> exporting to image 0.8s
=> [3/4] COPY . . 0.1s
=> [4/4] RUN yarn install --production 10.8s
=> [3/4] COPY . . 0.1s
=> [4/4] RUN yarn install --production 10.8s
=> [3/4] COPY . . 0.1s
=> [4/4] RUN yarn install --production 10.8s
=> [3/4] COPY . . 0.1s
=> [4/4] RUN yarn install --production 10.8s
=> [3/4] COPY . . 0.1s
=> [4/4] RUN yarn install --production 10.8s
=> [3/4] COPY . . 0.1s
=> [4/4] RUN yarn install --production 10.8s
=> [3/4] COPY . . 0.1s
=> [4/4] RUN yarn install --production 10.8s
=> exporting to image 0.8s
=> exporting layers 0.8s
=> writing image sha256:b5cce30d1ec2e4afadedad6e9997db23ec721ca5c6050ed3183ec825a188ffa 0.0s
=> naming to docker.io/library/myapp_6533801359 0.0s

```

Below the terminal output, it says: 'View build details: [docker-desktop://dashboard/build/desktop-linux/desktop-linux/yfprs61z2yms38wkecr5h8dp6](#)'

What's next:
View a summary of image vulnerabilities and recommendations → `docker scout quickview`
PS C:\softwareEngineering\Lab8_4\getting-started\app>

6. ทำการ Start ตัว Container ของแอปพลิเคชันที่สร้างขึ้น โดยใช้คำสั่ง

\$ docker run -dp 3000:3000 <myapp_รหัสศ. ไม่มีขีด>

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

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

The screenshot shows the Docker Desktop interface. The top pane displays the Dockerfile for 'getting-started > app' with the following content:

```

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

```

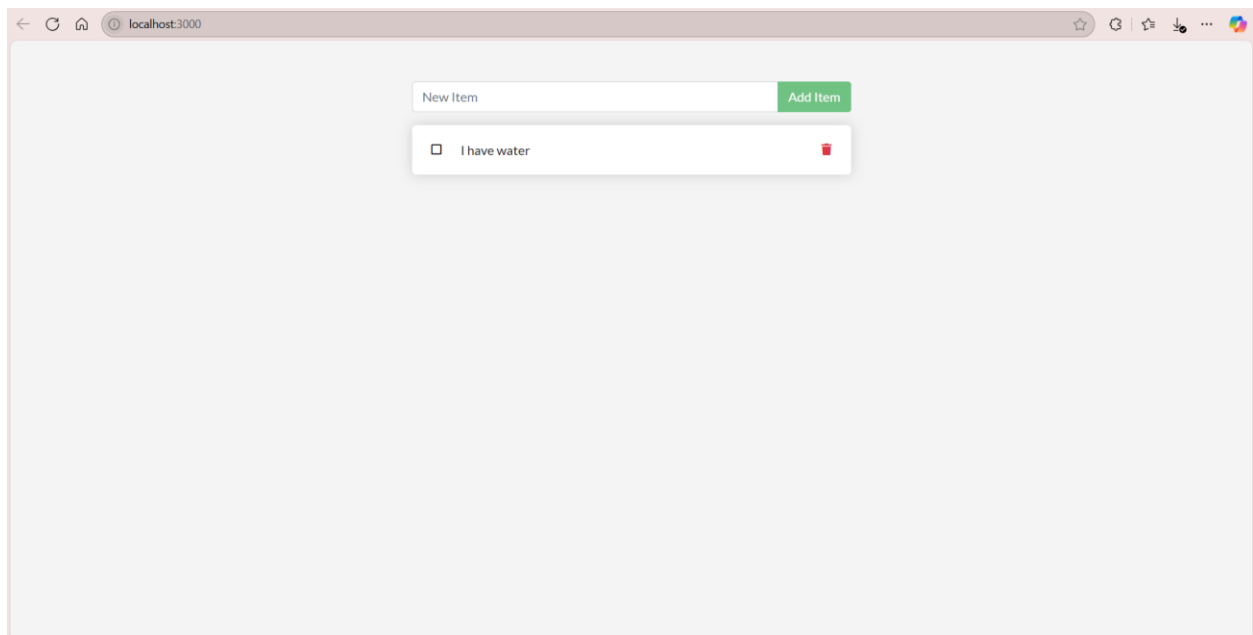
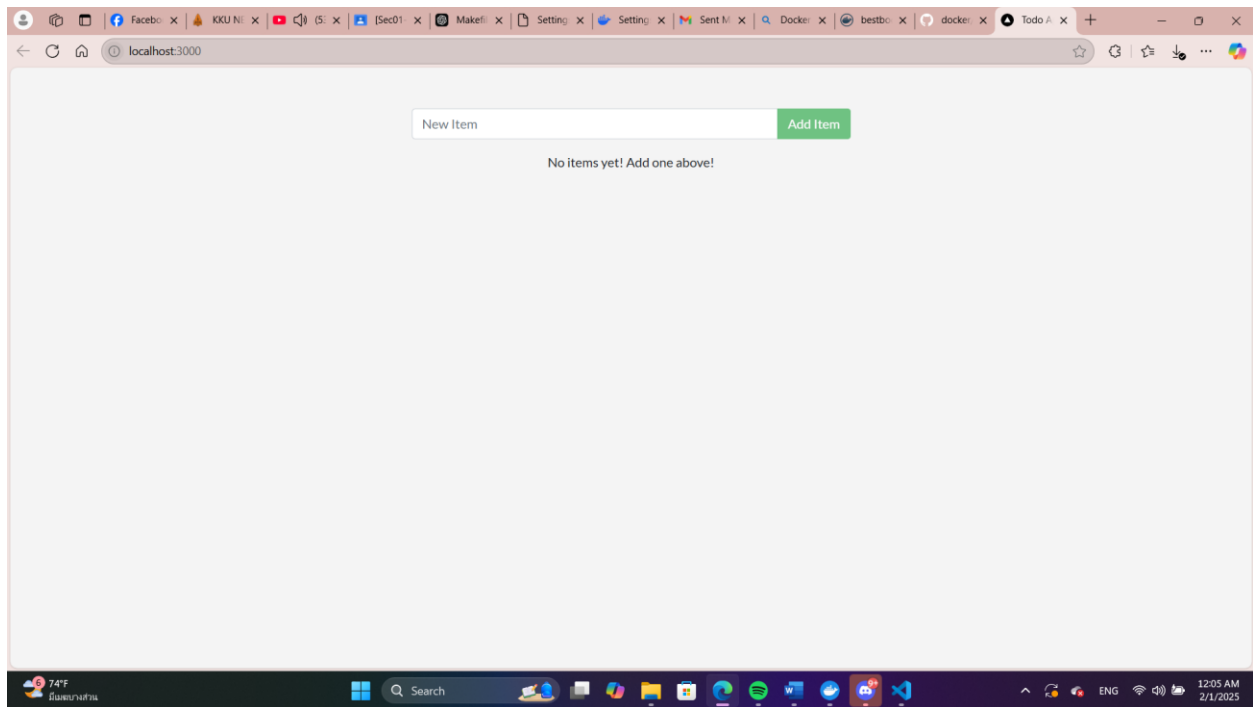
The bottom pane shows the 'TERMINAL' tab with the output of the `docker run` command:

```

PS C:\softwareEngineering\Lab8_4\getting-started\app> docker run -dp 3000:3000 myapp_6533801359
1971e973b5ecfa3c60e42b8ebc8e81d830cde29a87aab36702f93144e3e7e3f9
PS C:\softwareEngineering\Lab8_4\getting-started\app>

```

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

Lab Worksheet

<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 code editor with a file named 'app.js' and a terminal window. The code in 'app.js' is a React component 'TodoListCard' that displays a list of items. It includes a state variable 'items' and a function 'onNewItem' to add new items. The terminal shows the output of the 'docker build' and 'docker run' commands. The 'docker build' command successfully builds the image 'myapp_6533801359'. The 'docker run' command runs the container, and the output shows the container's IP address and the error message 'Error response from daemon: driver failed programming external connectivity on endpoint strange_dhawan (8bae62333468c4a848df9eb579cf3e21941de6f7072cced56f85226236ea836): Bind for 0.0.0.0:3000 failed: port is already allocated.'

```

getting-started > app > src > static > js > JS app.js > TodoListCard
14  function TodoListCard() {
42    const onItemRemoval = React.useCallback(
48    );
49
50    if (items === null) return 'Loading...';
51
52    return (
53      <React.Fragment>
54        <AddItemForm onNewItem={onNewItem} />
55        {items.length === 0 && (
56          <p className="text-center">There is no TODO item. Please add one to the list. By ชื่อและนามสกุลของนักศึกษา</p>
57        )}
58        {items.map(item => (

```

```

PS C:\softwareEngineering\Lab8_4\getting-started\app> docker build -t myapp_6533801359 .
[+] Building 15.3s (10/10) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 156B
=> [internal] load metadata for docker.io/library/node:18-alpine
=> [auth] library/node:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/4] FROM docker.io/library/node:18-alpine@sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d975e9059dd066be3e274fbb25
=> [internal] load build context
=> => transferring context: 8.19kB
=> CACHED [2/4] WORKDIR /app
=> [3/4] COPY . .
=> [4/4] RUN yarn install --production
=> exporting to image
=> => exporting layers
=> => writing image sha256:73ab83eb53a6f7d22c796a2cc3a436a92d67108b8122e0b510ac92a7857d6c3b
=> => naming to docker.io/library/myapp_6533801359

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/4kua2a0f8txk64xzqn9yf3qur

What's next:
  View a summary of image vulnerabilities and recommendations -> docker scout quickview
PS C:\softwareEngineering\Lab8_4\getting-started\app> docker run -dp 3000:3000 myapp_6533801359
0e3cf18699293f1fbd751d83fe48184b2fd0b349567a6c44e46f83d503d34354
docker: Error response from daemon: driver failed programming external connectivity on endpoint strange_dhawan (8bae62333468c4a848df9eb579cf3e21941de6f7072cced56f85226236ea836): Bind for 0.0.0.0:3000 failed: port is already allocated.
PS C:\softwareEngineering\Lab8_4\getting-started\app>

```

Lab Worksheet

(1) Error ที่เกิดขึ้นหมายความว่าอย่างไร และเกิดขึ้นเพราะอะไร

ตอบ

สิ่งที่เกิดขึ้น port is already allocated → หมายความว่า พอร์ต 3000 ถูกใช้งานอยู่แล้ว และไม่สามารถใช้ซ้ำได้

เกิดขึ้นเพราะ Docker พยายามเชื่อมต่อพอร์ต 3000 ของเครื่อง (host) กับพอร์ต 3000 ของคอนเทนเนอร์ แต่ไม่สำเร็จ เพราะมีบางอย่างกำลังใช้พอร์ตนี้อยู่

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

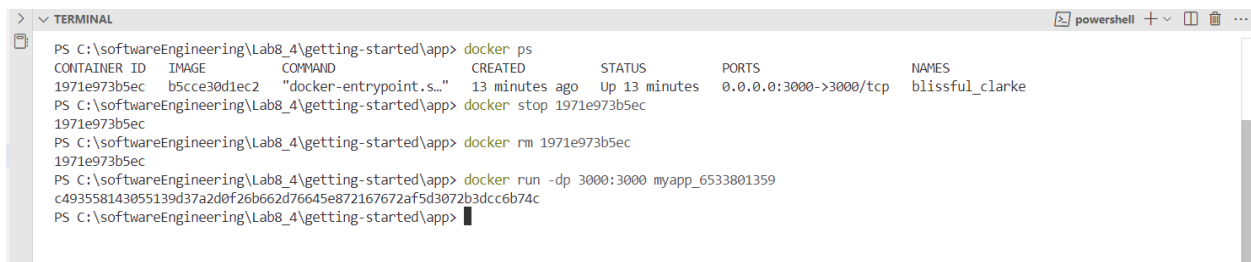
b. ผ่าน Docker desktop

- 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>

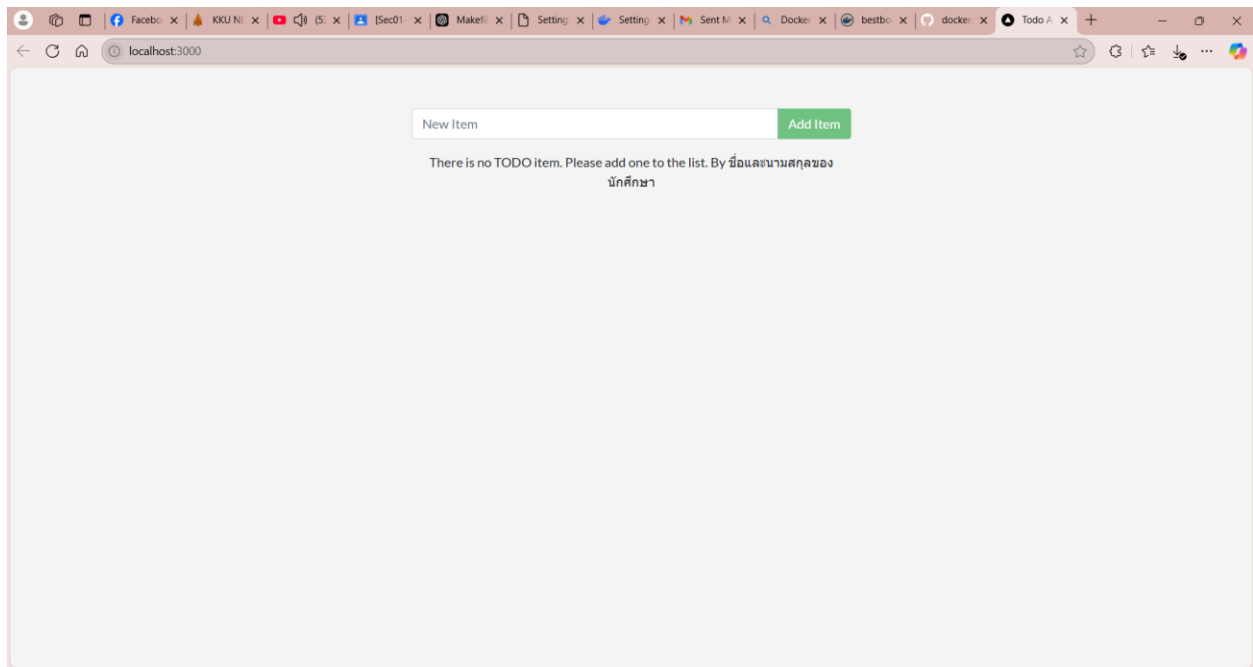
[Check point#11] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงผลลัพธ์ที่ได้นบน Browser และ Dashboard ของ Docker desktop



```

PS C:\softwareEngineering\Lab8_4\getting-started\app> docker ps
CONTAINER ID   IMAGE      COMMAND                  STATUS    PORTS                    NAMES
1971e973b5ec   b5cce30d1ec2  "docker-entrypoint.s..." 13 minutes ago  Up 13 minutes    0.0.0.0:3000->3000/tcp  blissful_clarke
PS C:\softwareEngineering\Lab8_4\getting-started\app> docker stop 1971e973b5ec
1971e973b5ec
PS C:\softwareEngineering\Lab8_4\getting-started\app> docker rm 1971e973b5ec
1971e973b5ec
PS C:\softwareEngineering\Lab8_4\getting-started\app> docker run -dp 3000:3000 myapp_6533801359
c493558143055139d37a2d0f26b662d76645e872167672af5d3072b3dccc6b74c
PS C:\softwareEngineering\Lab8_4\getting-started\app>
  
```

Lab Worksheet



แบบฝึกปฏิบัติที่ 8.5: เริ่มต้นสร้าง Pipeline อย่างง่ายสำหรับการ Deploy ด้วย Jenkins

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

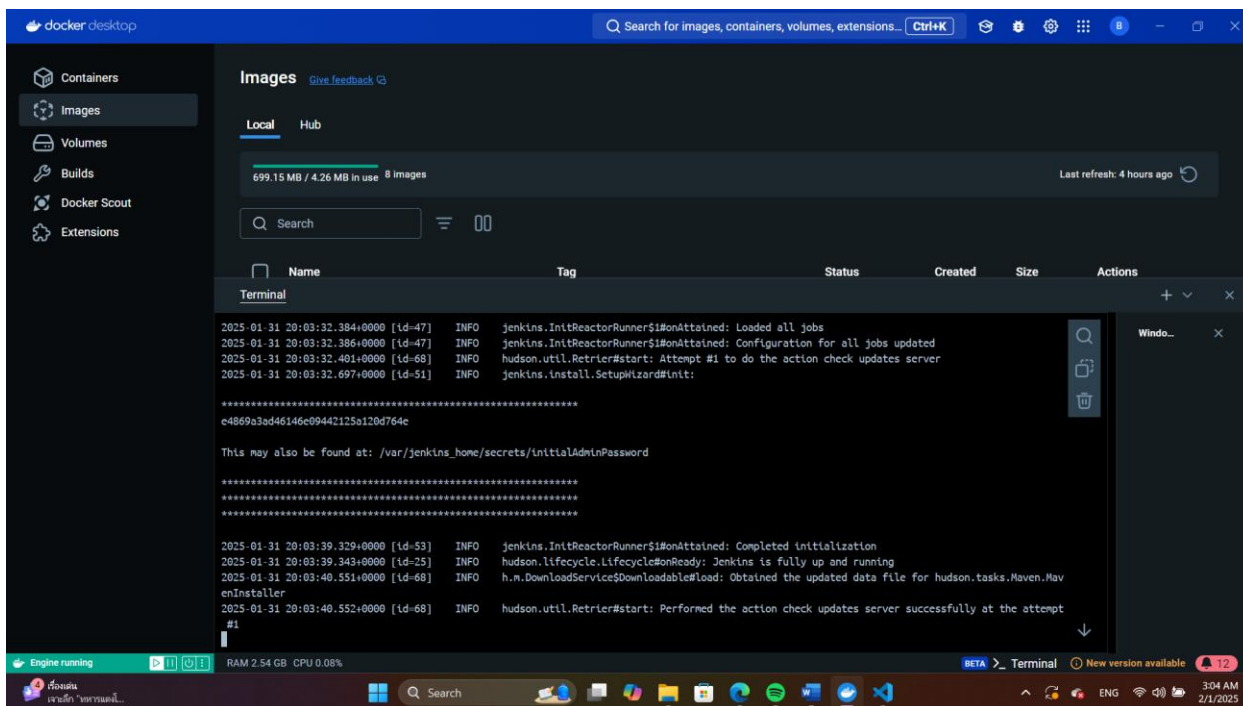
```
$ docker run -p 8080:8080 -p 50000:50000 --restart=on-failure jenkins/jenkins:its-jdk17
```

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

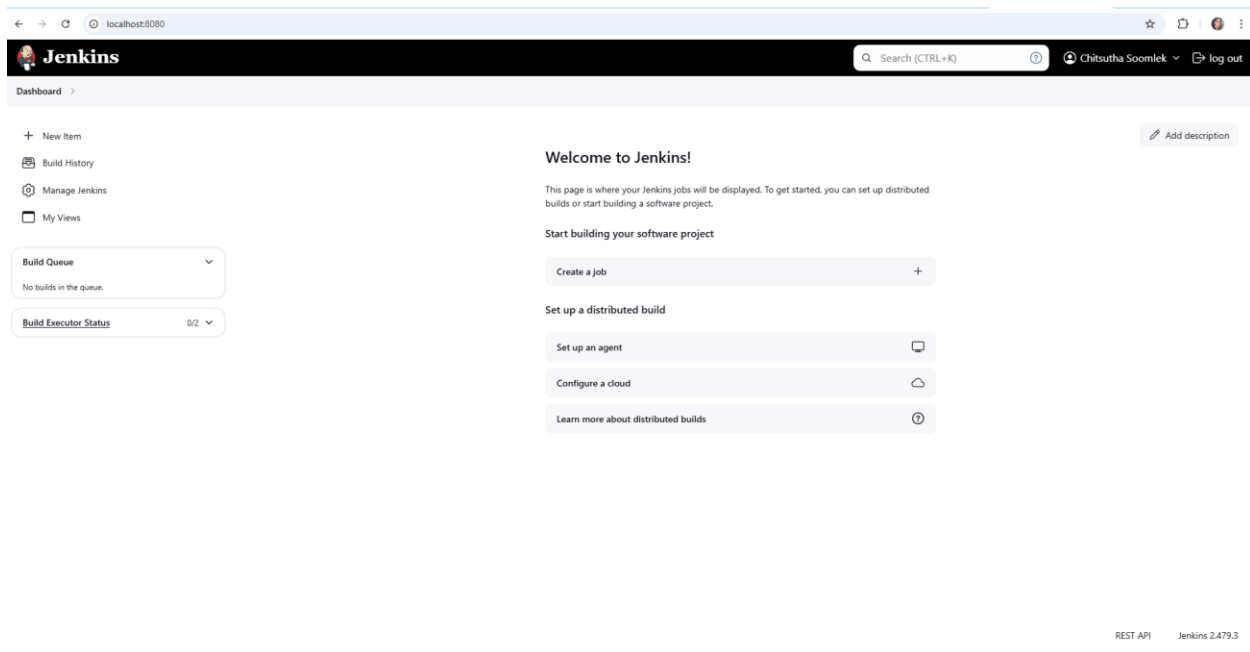
[Check point#12] Capture หน้าจอที่แสดงผล Admin password

Lab Worksheet



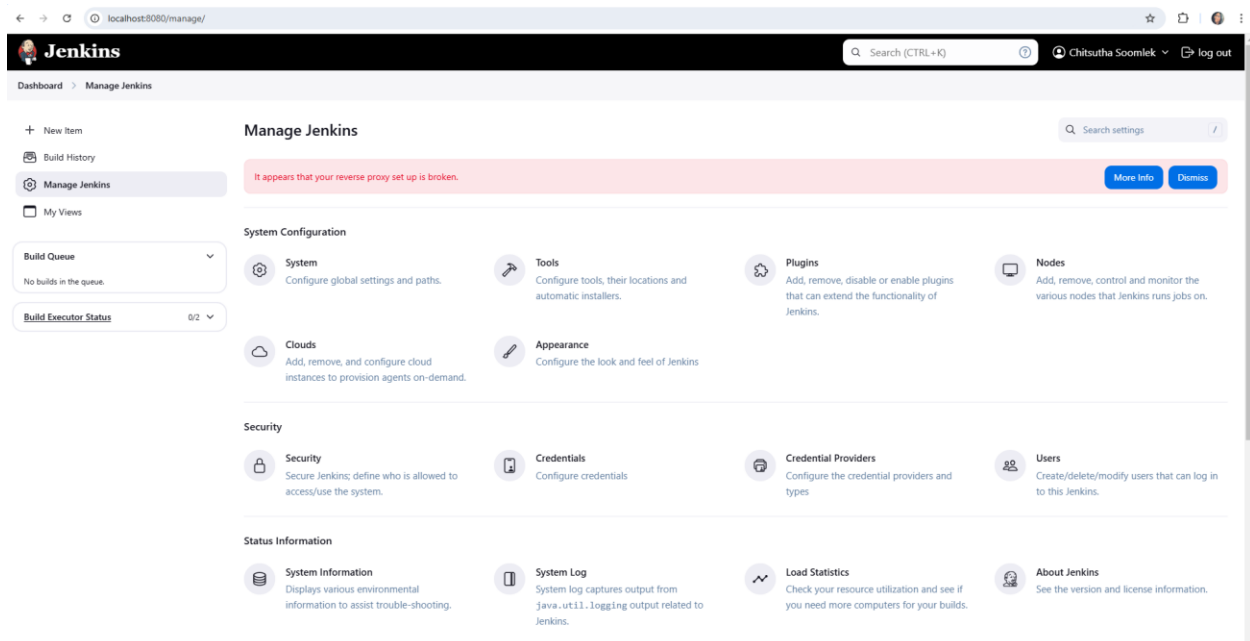
4. เมื่อได้รับการยืนยันว่า Jenkins is fully up and running ให้เปิดบราวเซอร์ และป้อนที่อยู่เป็น localhost:8080
5. ทำการ Unlock Jenkins ด้วยรหัสผ่านที่ได้ในข้อที่ 3
6. สร้าง Admin User โดยใช้ username เป็นชื่อจริงของนักศึกษาพร้อมรหัสสี่ตัวท้าย เช่น somsri_3062
- [Check point#13] Capture หน้าจอที่แสดงผลการตั้งค่า
7. กำหนด Jenkins URL เป็น <http://localhost:8080/lab8>
8. เมื่อติดตั้งเรียบร้อยแล้วจะพบหน้าจอ Dashboard ดังแสดงในภาพ

Lab Worksheet



The screenshot shows the Jenkins Dashboard at localhost:8080. The top navigation bar includes the Jenkins logo, a search bar, and user information (Chitsutha Soomlek) with a log out button. The left sidebar contains links for New Item, Build History, Manage Jenkins, and My Views. The main content area features a 'Welcome to Jenkins!' message, instructions on how to start building a software project, and buttons for 'Create a job', 'Set up a distributed build', 'Set up an agent', 'Configure a cloud', and 'Learn more about distributed builds'. A 'Build Queue' section shows 'No builds in the queue.' and a 'Build Executor Status' section shows '0/2'.

9. เลือก Manage Jenkins แล้วไปที่เมนู Plugins



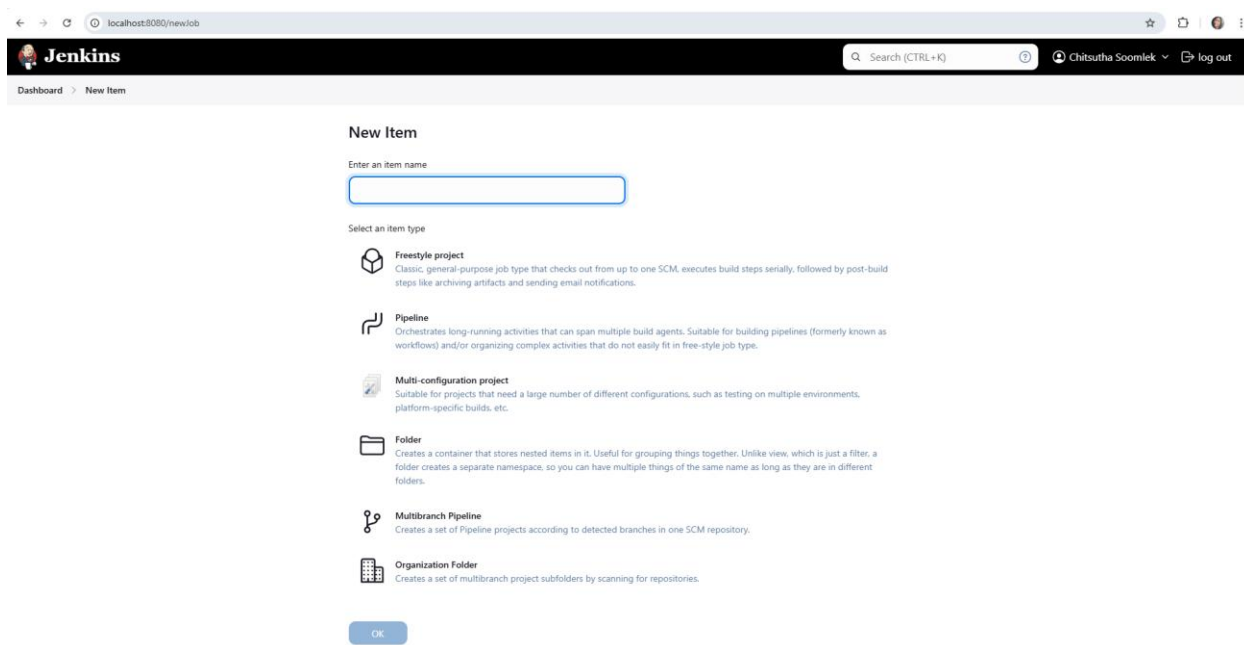
The screenshot shows the Jenkins 'Manage Jenkins' page at localhost:8080/manage/. The top navigation bar is the same as the dashboard. The left sidebar is also the same. The main content area has a 'Manage Jenkins' title and a search settings bar. A red banner at the top indicates 'It appears that your reverse proxy set up is broken.' with 'More Info' and 'Dismiss' buttons. Below this, the 'System Configuration' section includes links for System, Tools, Plugins, Nodes, Clouds, and Appearance. The 'Security' section includes links for Security, Credentials, Credential Providers, and Users. The 'Status Information' section includes links for System Information, System Log, Load Statistics, and About Jenkins.

Lab Worksheet

10. ไปที่เมนู Available plugins แล้วเลือกติดตั้ง Robotframework เพิ่มเติม



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



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

Description: Lab 8.5

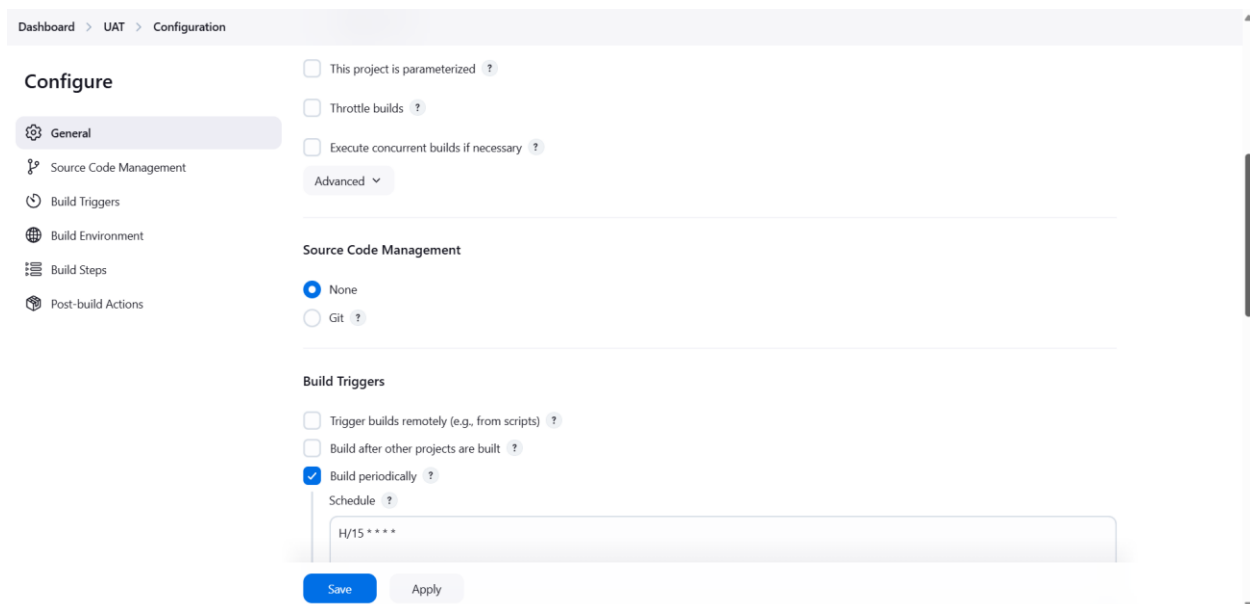
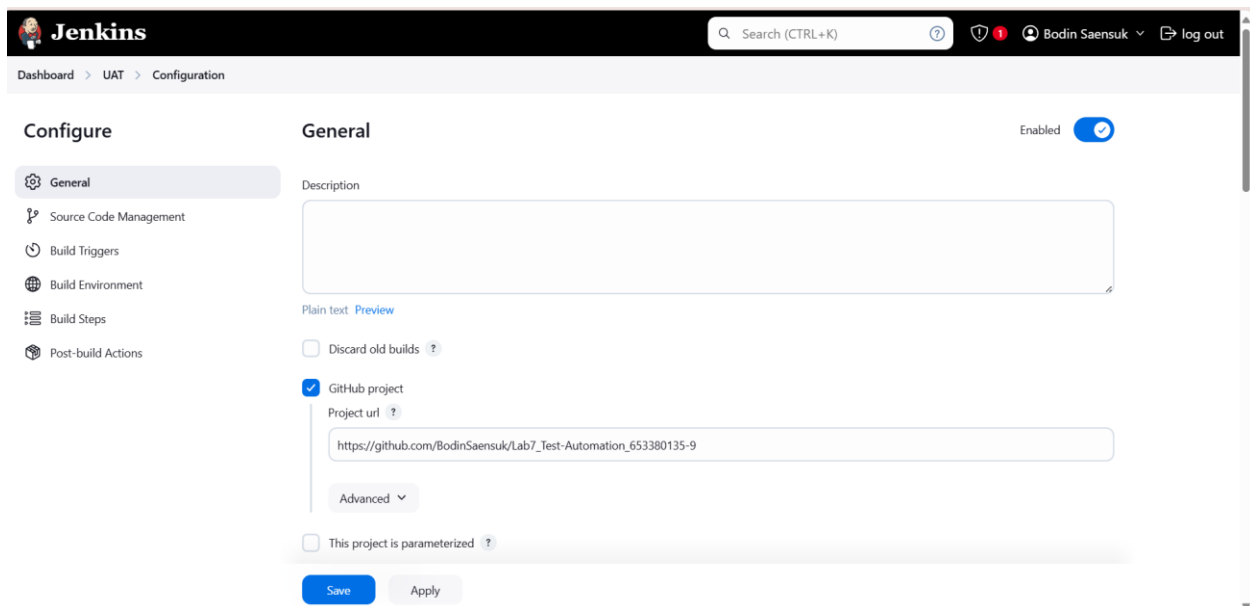
GitHub project: กดเลือก แล้วใส่ Project URL เป็น repository ที่เก็บโค้ด .robot (ดูขั้นตอนที่ 12)

Build Trigger: เลือกแบบ Build periodically แล้วกำหนดให้ build ทุก 15 นาที

Lab Worksheet

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

[Check point#14] Capture หน้าจอแสดงการตั้งค่า พร้อมกับตอบคำถามต่อไปนี้



Lab Worksheet

Dashboard > UAT > Configuration

Configure

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

Would last have run at Friday, January 31, 2025 at 10:50:30 PM Coordinated Universal Time; would next run at Friday, January 31, 2025 at 11:05:30 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

Execute shell ?

Save Apply

Dashboard > UAT > Configuration

Configure

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

Execute shell ?

Command

See the list of available environment variables

```
set -e
rm -rf WebDemo-master
git clone https://github.com/BodinSaensuk/Lab7_Test-Automation_653380135-9.git WebDemo-master
cd WebDemo-master
py demoapp/server.py &
sleep 2
robot login_tests/webdriver.robot
robot login_tests/invalid_login.robot
robot login_tests/valid_login.robot
kill $SERVER_PID || true
```

Save Apply

Lab Worksheet

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

ตอบ

```
set -e

rm -rf WebDemo-master

git clone https://github.com/BodinSaensuk/Lab7_Test-Automation_653380135-9.git

WebDemo-master

cd WebDemo-master

py demoapp/server.py &

sleep 2

robot login_tests/webdriver.robot

robot login_tests/invalid_login.robot

robot login_tests/valid_login.robot

kill $SERVER_PID || true
```

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

The image shows two screenshots of the Jenkins web interface. The top screenshot displays the 'Status' page for build #18, which is in a failed state. It shows the build was started by user Bodin Saensuk, took 12 seconds, and failed due to a 'No changes' error. The bottom screenshot shows the 'Console Output' for the same build, detailing the execution of a shell script that clones a repository, sets up a demo application, and attempts to publish results, ultimately failing with an exception.

Jenkins Dashboard - Build #18 Status

Dashboard > UAT > #18

Status (Failed) #18 (Feb 1, 2025, 4:52:18 PM)

Started by user [Bodin Saensuk](#) Started 18 sec ago
Took 12 sec

This run spent:

- 4 ms waiting;
- 12 sec build duration;
- 12 sec total from scheduled to completion.

No changes.

Jenkins Dashboard - Console Output

Dashboard > UAT > #18 > Console Output

Console Output

```

Started by user Bodin Saensuk
Running as SYSTEM
Building in workspace /var/jenkins_home/workspace/UAT
[UAT] $ /bin/sh -xe /tmp/jenkins78619375935638012.sh
+ set -e
+ rm -rf WebDemo-master
+ git clone https://github.com/BodinSaensuk/Lab8_653380135-9.git WebDemo-master
Cloning into 'WebDemo-master'...
+ cd WebDemo-master
+ sleep 2
+ py demoapp/server.py
/tmp/jenkins78619375935638012.sh: 6: py: not found
+ robot login_tests/webdriver.robot
/tmp/jenkins78619375935638012.sh: 8: 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:
ERROR: Build step failed with exception
/var/jenkins_home/workspace/UAT/WebDemo-master/report.html does not exist.
at org.apache.tools.ant.types.AbstractFileSet.getDirectoryScanner(AbstractFileSet.java:512)
at org.apache.tools.ant.types.AbstractFileSet.getDirectoryScanner(AbstractFileSet.java:489)
at PluginClassLoader for robot/hudson.plugins.robot.RobotParser$RobotParserCallable.invoke(RobotParser.java:76)
at PluginClassLoader for robot/hudson.plugins.robot.RobotParser$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)
  
```

Lab Worksheet

```
Dashboard > UAT > #18 > Console Output

+ sleep 2
+ py demoapp/server.py
/tmp/jenkins786192759356308012.sh: 6: py: not found
+ robot login_tests/webdriver.robot
/tmp/jenkins786192759356308012.sh: 8: 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:
ERROR: Build step failed with exception
/var/jenkins_home/workspace/UAT/WebDemo-master/report.html does not exist.
    at org.apache.tools.ant.types.AbstractFileSet.getDirectoryScanner(AbstractFileSet.java:512)
    at org.apache.tools.ant.types.AbstractFileSet.getDirectoryScanner(AbstractFileSet.java:489)
    at PluginClassLoader for robot/hudson.plugins.robot.RobotParser$RobotParserCallable.invoke(RobotParser.java:76)
    at PluginClassLoader for robot/hudson.plugins.robot.RobotParser$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:80)
    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)
Build step 'Publish Robot Framework test results' marked build as failure
Finished: FAILURE
```

REST API Jenkins 2.479.3