

## Lab Worksheet

ชื่อ-นามสกุล วาโรณี อนุสุเรนทร์

รหัสนักศึกษา 653380341-6

Section 3

## Lab#8 – Software Deployment Using Docker

## วัตถุประสงค์การเรียนรู้

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

## Pre-requisite

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

## แบบฝึกปฏิบัติที่ 8.1 Hello world - รัน Container จาก Docker image

1. เปิดใช้งาน Docker desktop และ Login ด้วย Username และ Password ที่ลงทะเบียนกับ Docker Hub เอาไว้
1. เปิด Command line หรือ Terminal บน Docker Desktop จากนั้นสร้าง Directory ชื่อ Lab8\_1
2. ย้ายตำแหน่งปัจจุบันไปที่ Lab8\_1 เพื่อใช้เป็น Working directory
3. ป้อนคำสั่ง \$ docker pull busybox หรือ \$ sudo docker pull busybox สำหรับกรณีที่ติดปัญหา Permission denied  
(หมายเหตุ: BusyBox เป็น software suite ที่รองรับคำสั่งบางอย่างบน Unix - <https://busybox.net>)
4. ป้อนคำสั่ง \$ docker images

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

## Lab Worksheet

```
PS C:\Users\ACER> mkdir Lab8_1
```

```
Directory: C:\Users\ACER
```

Mode	LastWriteTime	Length	Name
d-----	1/23/2025 9:25 AM		Lab8_1

```
PS C:\Users\ACER> cd Lab8_1
```

## Terminal

```
PS C:\Users\ACER\Lab8_1> docker pull busybox
Using default tag: latest
latest: Pulling from library/busybox
Digest: sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138bd458fc8257fbf1
```

## What's next:

View a summary of image vulnerabilities and recommendations → `docker scout quickview busybox`

```
PS C:\Users\ACER\Lab8_1> docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
jenkins/jenkins	latest	fe12c0bbe82f	35 hours ago	466MB
ubuntu	latest	b1d9df8ab815	2 months ago	78.1MB
busybox	latest	af4709625109	3 months ago	4.27MB

- (1) สิ่งที่อยู่ภายใต้คอลัมน์ Repository คืออะไร คือชื่อของ Docker image ที่ถูกสร้างหรือที่เราดึงมา
- (2) Tag ที่ใช้บ่งบอกถึงอะไร ระบุเวอร์ชันของ image นั้นๆ

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

Terminal

```

PS C:\Users\ACER\lab8_1> docker run busybox
PS C:\Users\ACER\lab8_1> docker run -it busybox sh
/ #
/ # ls
bin      dev      etc      home     lib      lib64    proc     root     sys      tmp      usr      var
/ # ls -la
total 48
drwxr-xr-x  1 root    root      4096 Jan 23 02:51 .
drwxr-xr-x  1 root    root      4096 Jan 23 02:51 ..
-rwxr-xr-x  1 root    root        0 Jan 23 02:51 .dockerenv

```

Terminal

```

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 265 root    root        0 Jan 23 02:51 proc
drwx----- 1 root    root      4096 Jan 23 02:52 root
dr-xr-xr-x 11 root    root        0 Jan 23 02:51 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
/ # exit
PS C:\Users\ACER\lab8_1>

```

Terminal

```

/ # exit
PS C:\Users\ACER\lab8_1> docker run busybox echo "Hello วาริณีอนุสรณ์ from busybox"
Hello วาริณีอนุสรณ์ from busybox
PS C:\Users\ACER\lab8_1> docker ps -a

```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
88203b2d79c2	busybox	"echo 'Hello วาริณีอนุสรณ์'"	9 seconds ago	Exited (0)
8 seconds ago			crazy_pasteur	
0727d6dacf55	busybox	"sh"	4 minutes ago	Exited (0) Ab
out a minute ago			happy_hermann	

Terminal

```

8 seconds ago
0727d6dacf55 busybox "sh" 4 minutes ago Exited (0) Ab
out a minute ago happy_hermann
de448fd4ee3f busybox "sh" 4 minutes ago Exited (0) 4
minutes ago blissful_albattani
03fid7440a89 jenkins/jenkins:lts-jdk17 "/usr/bin/tini -- /u..." 19 minutes ago Up 19 minutes
0.0.0.0:8080->8080/tcp, 0.0.0.0:50000->50000/tcp upbeat_blackburn
dc6084bbdeb3 docker/welcome-to-docker:latest "/docker-entrypoint...." About an hour ago Exited (0) 55
minutes ago welcome-to-docker
PS C:\Users\ACER\lab8_1>

```

(1) เมื่อใช้ option -it ในคำสั่ง run ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป

- สามารถรันคำสั่งและโต้ตอบกับ container ได้ตามปกติในขณะที่มันทำงาน

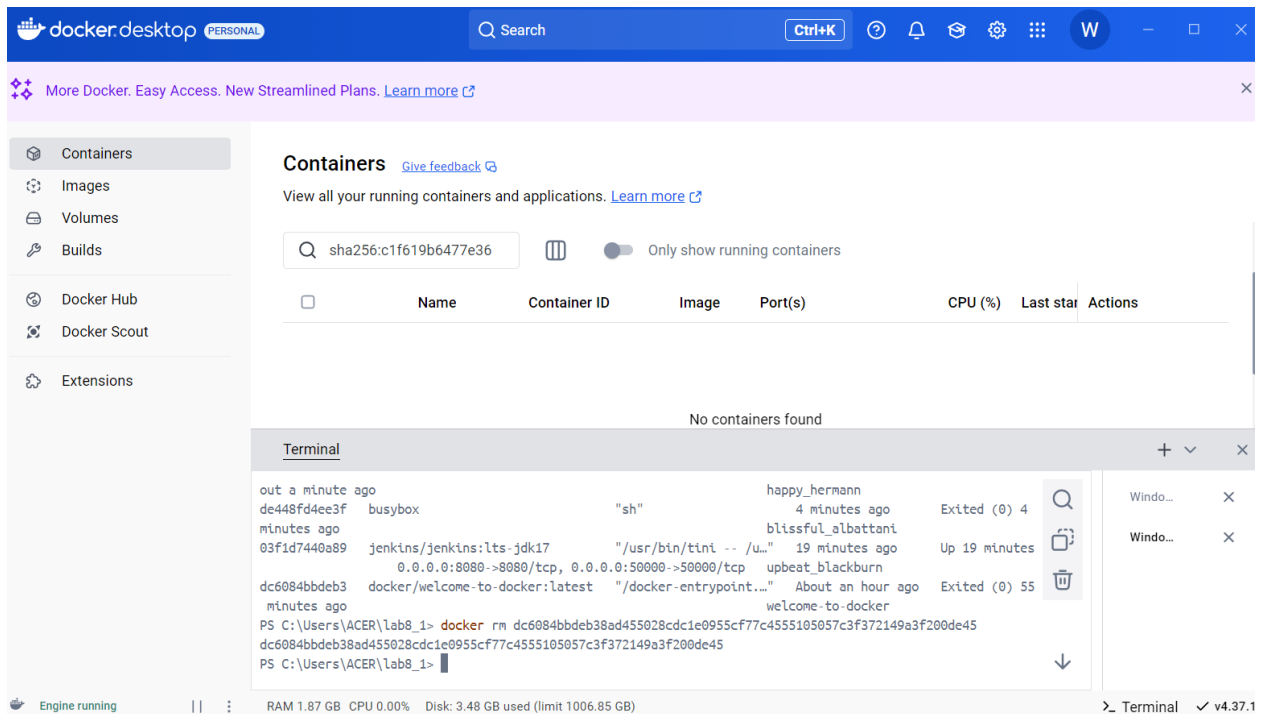
## Lab Worksheet

(2) คอลัมน์ STATUS จากการรันคำสั่ง `docker ps -a` แสดงถึงข้อมูลอะไร

สถานะปัจจุบันของแต่ละ images เช่น `up 19 minutes ago` คือทำงานมาแล้ว 19 นาที `exited` คือเลิกทำงาน `Created` คือ ได้ถูกสร้างขึ้น แต่ยังไม่เริ่มทำงาน+

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

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



## แบบฝึกปฏิบัติที่ 8.2: สร้าง Docker file และ Docker image

1. เปิดใช้งาน Docker desktop และ Login ด้วย Username และ Password ที่ลงทะเบียนกับ Docker Hub เอาไว้
2. เปิด Command line หรือ Terminal จากนั้นสร้าง Directory ชื่อ Lab8\_2
3. ย้ายตำแหน่งปัจจุบันไปที่ Lab8\_2 เพื่อใช้เป็น Working directory
4. สร้าง Dockerfile.swp ไว้ใน Working directory

สำหรับเครื่องที่ใช้ระบบปฏิบัติการวินโดวส์ (Windows) บันทึกคำสั่งต่อไปนี้ลงในไฟล์ โดยใช้ Text Editor ที่มี

FROM busybox

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

CMD echo "ชื่อ-นามสกุล รหัสนักศึกษา ชื่อเล่น"

## Lab Worksheet

สำหรับเครื่องที่ใช้ระบบปฏิบัติการ MacOS หรือ Linux บนหน้าต่าง Terminal และป้อนคำสั่งต่อไปนี้

```
$ cat > Dockerfile << EOF
FROM busybox
CMD echo "Hi there. This is my first docker image."
CMD echo "ชื่อ-นามสกุล รหัสนักศึกษา ชื่อเล่น"
EOF
```

หรือใช้คำสั่ง

```
$ touch Dockerfile
```

แล้วใช้ Text Editor ในการใส่เนื้อหาแทน

5. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้

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

6. เมื่อ Build สำเร็จแล้ว ให้ทำการรัน Docker image ที่สร้างขึ้นในขั้นตอนที่ 5

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

The screenshot shows the Docker Desktop application. The left sidebar contains navigation options: Containers, Images (selected), Volumes, Builds, Docker Hub, Docker Scout, and Extensions. The main area displays the 'Images' tab with a table of local and Docker Hub images.

Name	Tag	Image ID	Created	Size	Ac
docker/welcome-to-doc	latest	c1f619b6477e	1 year ago	18.55 MB	[ ]
ubuntu	latest	b1d9df8ab815	2 months ago	78.12 MB	[ ]
jenkins/jenkins	latest	fe12c0bbe82f	2 days ago	465.97 MB	[ ]
busybox	latest	af4709625109	4 months ago	4.26 MB	[ ]

Below the images table is a terminal window titled 'Terminal'. It shows the command 'mkdir Lab8\_2' being executed in a PowerShell prompt. The output shows the directory 'C:\Users\ACER' and a file listing for 'Lab8\_2'.

```
PS C:\Users\ACER> mkdir Lab8_2

Directory: C:\Users\ACER

Mode                LastWriteTime         Length Name
----                -
d-----          1/23/2025  10:00 AM             Lab8_2
```

## Lab Worksheet

```
PS C:\Users\ACER\lab8_2> New-Item -Path . -Name "Dockerfile" -ItemType "File"
```

Directory: C:\Users\ACER\lab8\_2

Mode	LastWriteTime	Length	Name
-a----	1/23/2025 10:24 AM	0	Dockerfile

More Docker. Easy Access. New Streamlined Plans. [Learn more](#)

**Images** [Give feedback](#)

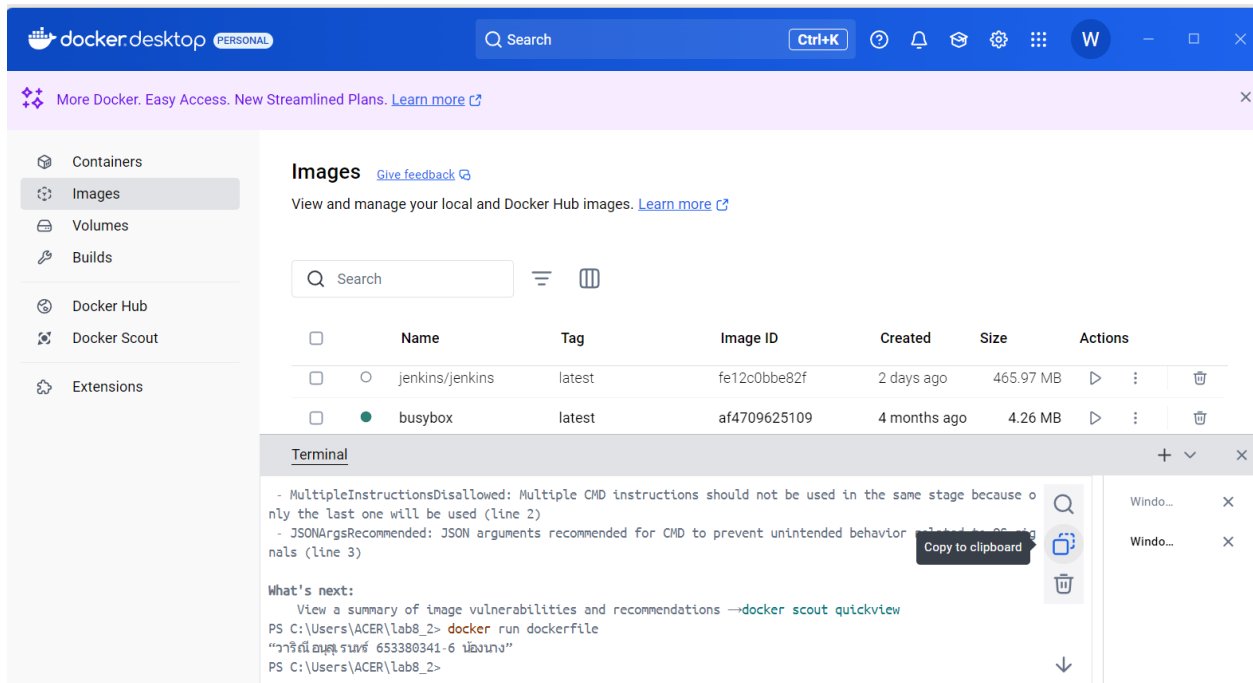
View and manage your local and Docker Hub images. [Learn more](#)

	Name	Tag	Image ID	Created	Size	Actions
<input type="checkbox"/>	ubuntu	latest	b1d9df8ab815	2 months ago	78.12 MB	
<input type="checkbox"/>	jenkins/jenkins	latest	fe12c0bbe82f	2 days ago	465.97 MB	

**Terminal**

```
PS C:\Users\ACER\lab8_2> docker build -t dockerfile .
[+] Building 0.1s (5/5) FINISHED
=> [internal] load build definition from Dockerfile 0.0s
=> => transferring dockerfile: 213B 0.0s
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior rel 0.0s
=> WARN: MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the same s 0.0s
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior rel 0.0s
=> [internal] load metadata for docker.io/library/busybox:latest 0.0s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
```

## Lab Worksheet



(1) คำสั่งที่ใช้ในการ run คือ

**docker run dockerfile**

(2) Option -t ในคำสั่ง \$ docker build ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป  
ช่วยให้เราตั้งชื่อ ให้กับ image ที่สร้างขึ้นใหม่

### 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

```
PS C:\Users\ACER> cd Lab8_3
PS C:\Users\ACER\Lab8_3> New-Item -Path . -Name "Dockerfile" -ItemType "File"
```

Directory: C:\Users\ACER\Lab8\_3

Mode	LastWriteTime	Length	Name
-a----	1/29/2025 5:33 PM	0	Dockerfile





## Lab Worksheet

```

FROM busybox
CMD echo "Hi there. My work is done. You can run them from my Docker image."
CMD echo "วารีนท์ อนุสรณ์ 653380341-6"

Start a build
PS C:\Users\ACER\Lab8_3> docker build -t warinee3416/lab8 .
[+] Building 0.2s (5/5) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 216B
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior rel
=> WARN: MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the same s
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior rel
=> [internal] load metadata for docker.io/library/busybox:latest
=> [internal] load .dockerignore
PS C:\Users\ACER\Lab8_3> docker run warinee3416/lab8
"วารีนท์ อนุสรณ์ 653380341-6"
PS C:\Users\ACER\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

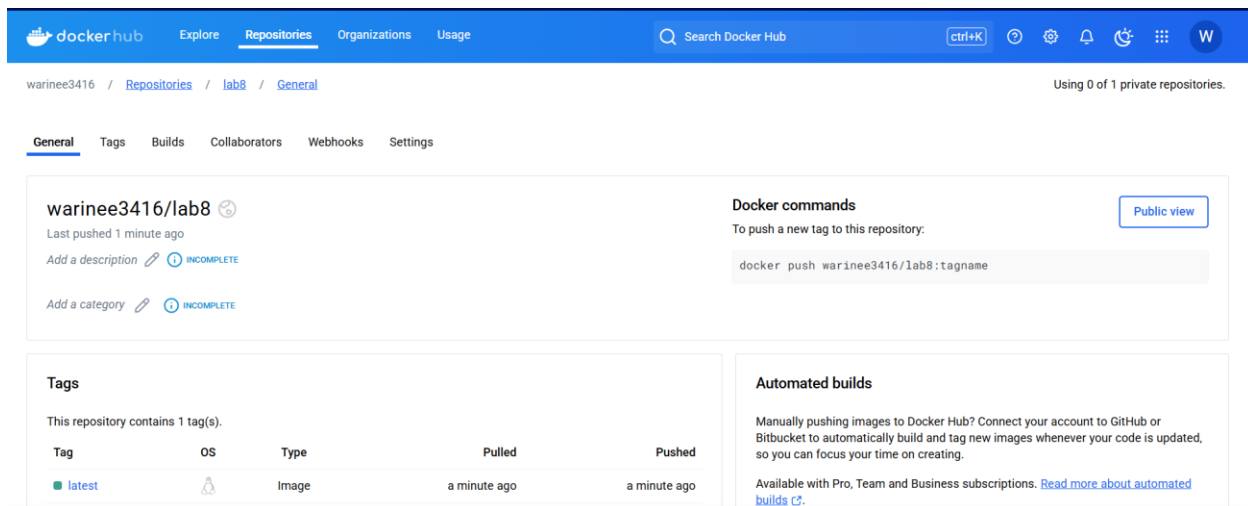
\$ docker login -u <username> -p <password>

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

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

Name	Last Pushed	Contains	Visibility	Scout
warinee3416	3 minutes ago	IMAGE	Public	Inactive

## 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

```
PS C:\Users\ACER> mkdir Lab8_4
```

Directory: C:\Users\ACER

Mode	LastWriteTime	Length	Name
d----	1/29/2025 5:58 PM		Lab8_4

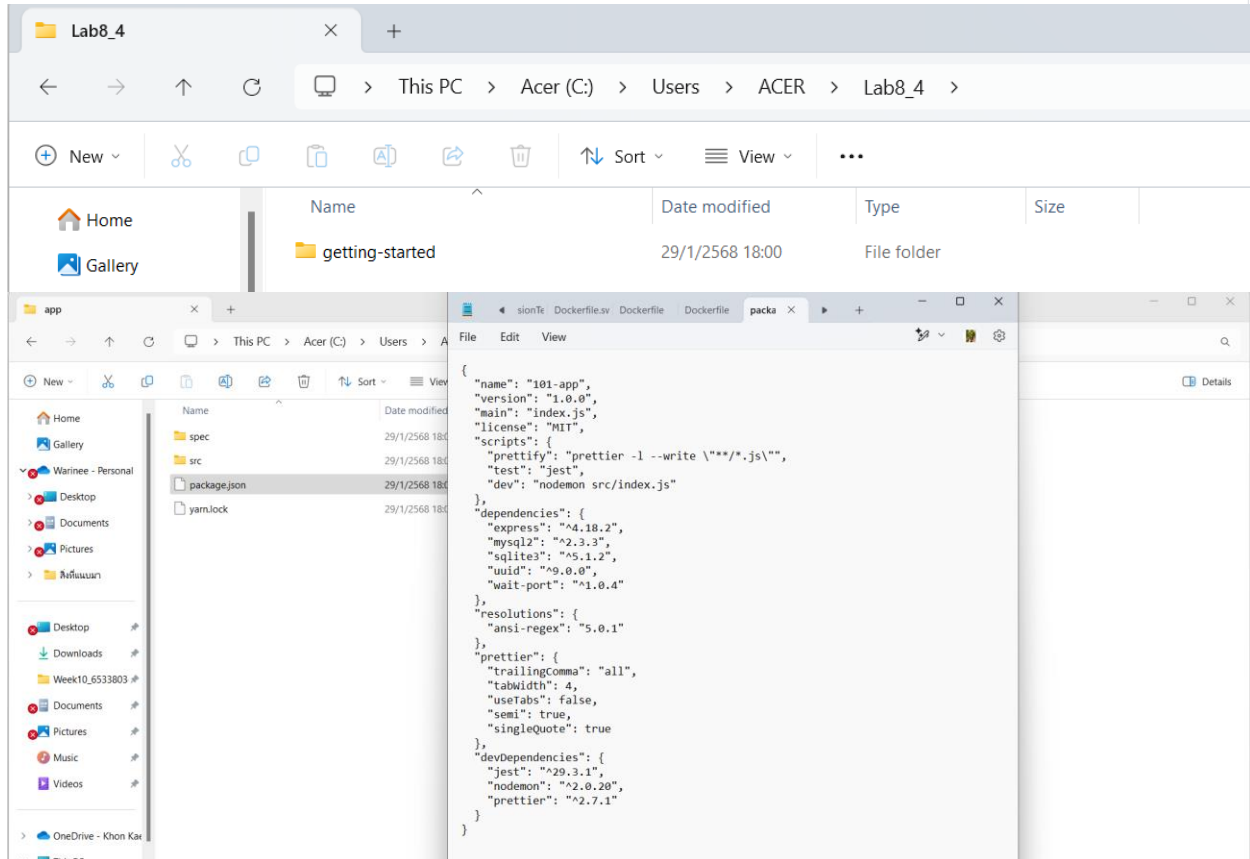


## Lab Worksheet

```

PS C:\Users\ACER> cd Lab8_4
PS C:\Users\ACER\Lab8_4> git clone https://github.com/docker/getting-started.git
Cloning into 'getting-started'...
remote: Enumerating objects: 980, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 980 (delta 5), reused 1 (delta 1), pack-reused 971 (from 2)
Receiving objects: 100% (980/980), 5.28 MiB | 305.00 KiB/s, done.
Resolving deltas: 100% (523/523), done.
PS C:\Users\ACER\Lab8_4>

```



4. ภายใต้ getting-started/app ให้สร้าง Dockerfile พร้อมกับใส่เนื้อหาดังต่อไปนี้ลงไปไฟล์

FROM node:18-alpine

WORKDIR /app

COPY . .

RUN yarn install --production

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

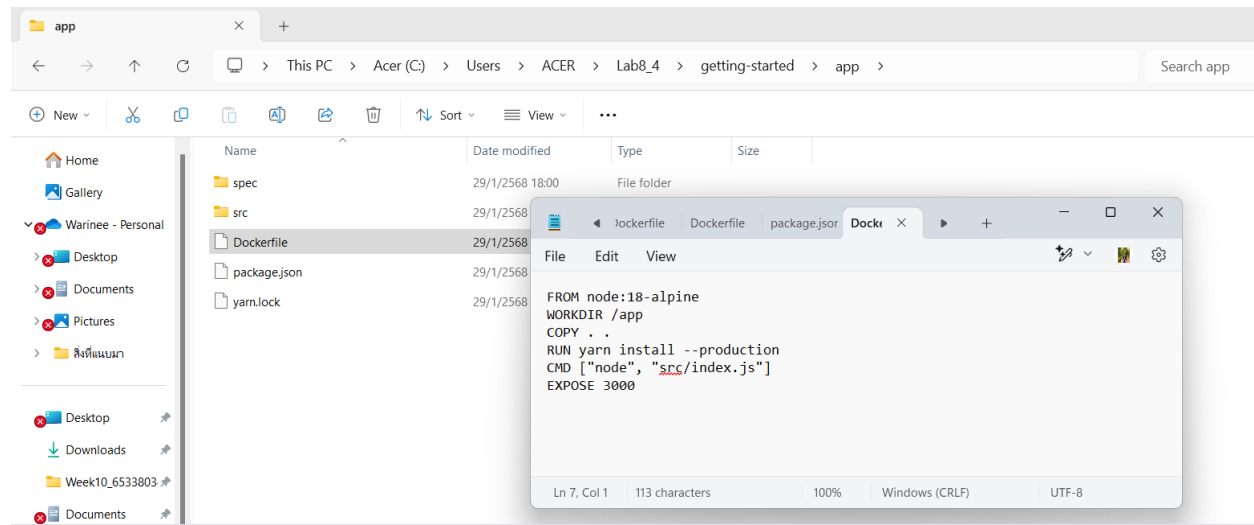
EXPOSE 3000

## Lab Worksheet

5. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้ โดยกำหนดใช้ชื่อ image เป็น myapp\_รหัสสนศ. ไม่มีขีด

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

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



The screenshot shows a Windows File Explorer window titled 'app' with the following files and folders:

Name	Date modified	Type	Size
spec	29/1/2568 18:00	File folder	
src	29/1/2568	File folder	
Dockerfile	29/1/2568	File	
package.json	29/1/2568	File	
yarn.lock	29/1/2568	File	

The Dockerfile editor window shows the following content:

```
FROM node:18-alpine
WORKDIR /app
COPY . .
RUN yarn install --production
CMD ["node", "src/index.js"]
EXPOSE 3000
```

The Terminal window shows the following commands and output:

```
PS C:\Users\ACER\Lab8_4> cd C:\Users\ACER\Lab8_4\getting-started\app
PS C:\Users\ACER\Lab8_4\getting-started\app> docker build -t myapp_6533803416 .
[+] Building 69.1s (10/10) FINISHED
=> [internal] load build definition from Dockerfile 0.0s
=> => transferring dockerfile: 156B 0.0s
=> [internal] load metadata for docker.io/library/node:18-alpine 5.3s
=> [auth] library/node:pull token for registry-1.docker.io 0.0s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [1/4] FROM docker.io/library/node:18-alpine@sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d975e9 40.9s
```

The second Terminal window shows the following commands and output:

```
=> [internal] load build context 0.5s
=> => transferring context: 4.59MB 0.5s
=> [2/4] WORKDIR /app 0.2s
=> [3/4] COPY . . 0.1s
=> [4/4] RUN yarn install --production 20.8s
=> exporting to image 1.7s
=> => exporting layers 1.7s
=> => writing image sha256:8a647c93510288e8967418239b5b5e6b9837c498dca3296f63a28258d5f9691c 0.0s
=> => naming to docker.io/library/myapp_6533803416 0.0s
```

## Lab Worksheet

6. ทำการ Start ตัว Container ของแอปพลิเคชันที่สร้างขึ้น โดยใช้คำสั่ง

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

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

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

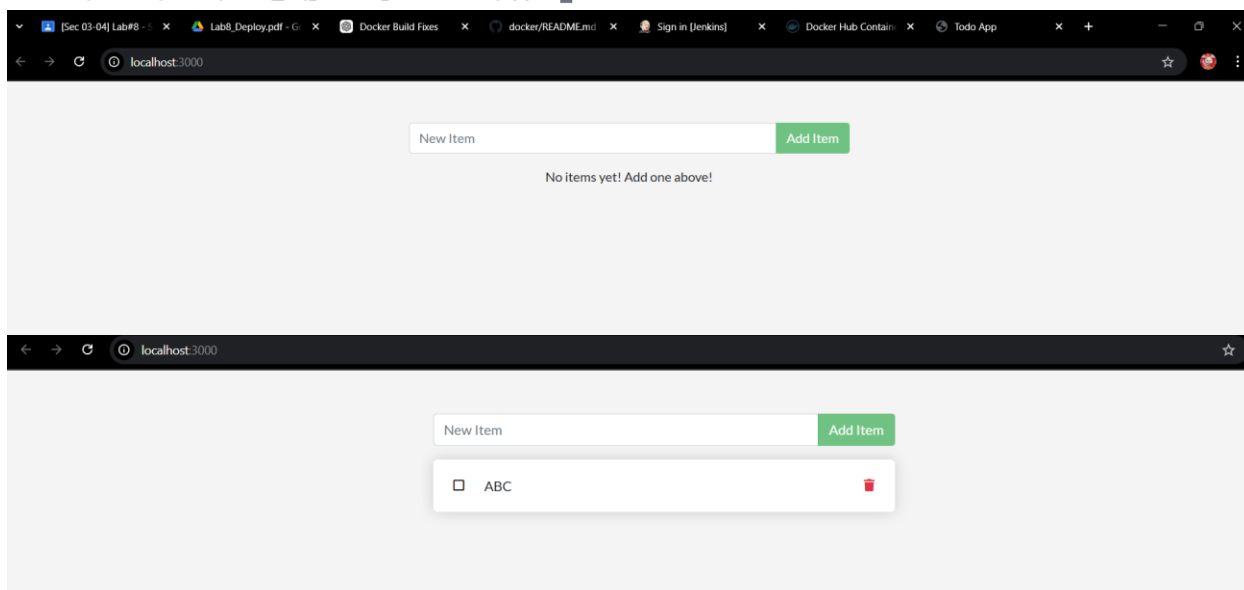
## What's next:

View a summary of image vulnerabilities and recommendations → [docker scout quickview](#)

```
PS C:\Users\ACER\Lab8_4\getting-started\app> docker run -dp 3000:3000 myapp_6533803416
```

```
076414d251f3daf346d15b893b7f761ab37618ef61c5f0d0045934e0969d7b47
```

```
PS C:\Users\ACER\Lab8_4\getting-started\app>
```



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>

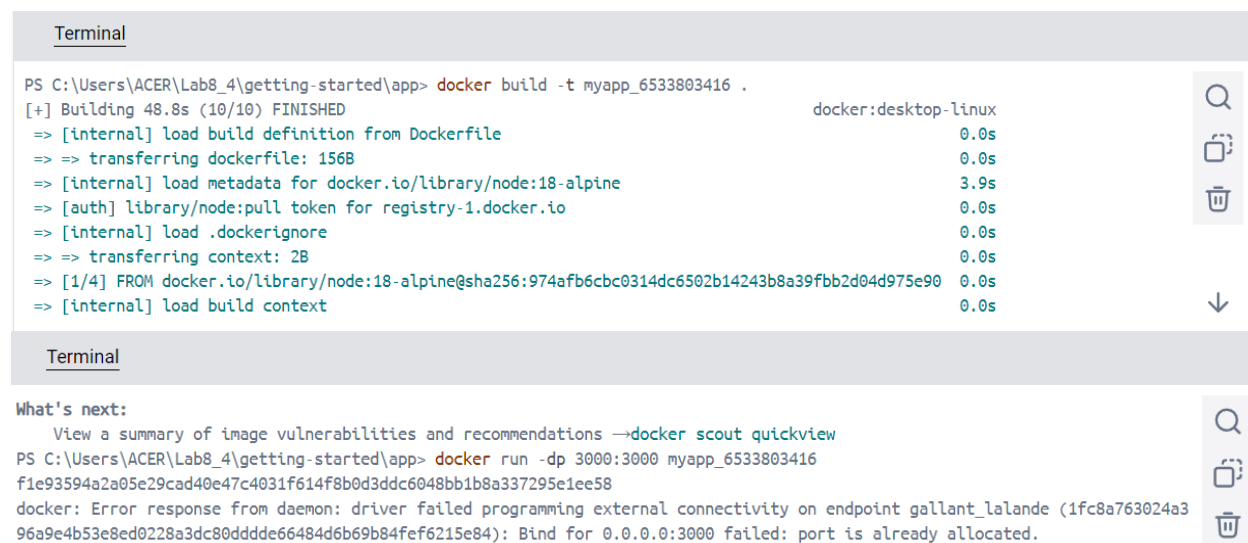
- b. Save ไฟล์ให้เรียบร้อย

9. ทำการ Build Docker image โดยใช้คำสั่งเดียวกันกับข้อ 5

10. Start และรัน Container ตัวใหม่ โดยใช้คำสั่งเดียวกันกับข้อ 6

## Lab Worksheet

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



```

Terminal
PS C:\Users\ACER\Lab8_4\getting-started\app> docker build -t myapp_6533803416 .
[+] Building 48.8s (10/10) FINISHED                                docker:desktop-linux
=> [internal] load build definition from Dockerfile                0.0s
=> => transferring dockerfile: 156B                               0.0s
=> [internal] load metadata for docker.io/library/node:18-alpine  3.9s
=> [auth] library/node:pull token for registry-1.docker.io        0.0s
=> [internal] load .dockerignore                                  0.0s
=> => transferring context: 2B                                     0.0s
=> [1/4] FROM docker.io/library/node:18-alpine@sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d975e90 0.0s
=> [internal] load build context                                  0.0s

Terminal
What's next:
View a summary of image vulnerabilities and recommendations →docker scout quickview
PS C:\Users\ACER\Lab8_4\getting-started\app> docker run -dp 3000:3000 myapp_6533803416
f1e93594a2a05e29cad40e47c4031f614f8b0d3ddc6048bb1b8a337295e1ee58
docker: Error response from daemon: driver failed programming external connectivity on endpoint gallant_lalande (1fc8a763024a396a9e4b53e8ed0228a3dc80ddde66484d6b69b84fef6215e84): Bind for 0.0.0.0:3000 failed: port is already allocated.

```

(1) Error ที่เกิดขึ้นหมายความว่าอย่างไร และเกิดขึ้นเพราะอะไร

- หมายความว่า พอร์ต 3000 ถูกใช้งานแล้ว เกิดจากมีคอนเทนเนอร์ Docker ที่ใช้พอร์ต 3000 อยู่แล้ว

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

## Lab Worksheet

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

The screenshot shows the Docker Desktop interface. The top terminal window displays the command `docker build -t myapp_6533803416 .` and its output, including building the image and transferring the context. The bottom terminal window shows the command `docker run -dp 3000:3000 myapp_6533803416` and its output, including the container ID and the URL to view build details. Below the terminal windows, a browser window is open at `localhost:3000`, showing a simple web page with a 'New Item' input field and an 'Add Item' button. The page also displays a message: 'There is no TODO item. Please add one to the list. By Warinee Anusurean'.

### 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`

## Lab Worksheet

3. บันทึกรหัสผ่านของ Admin user ไว้สำหรับ log-in ในครั้งแรก

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

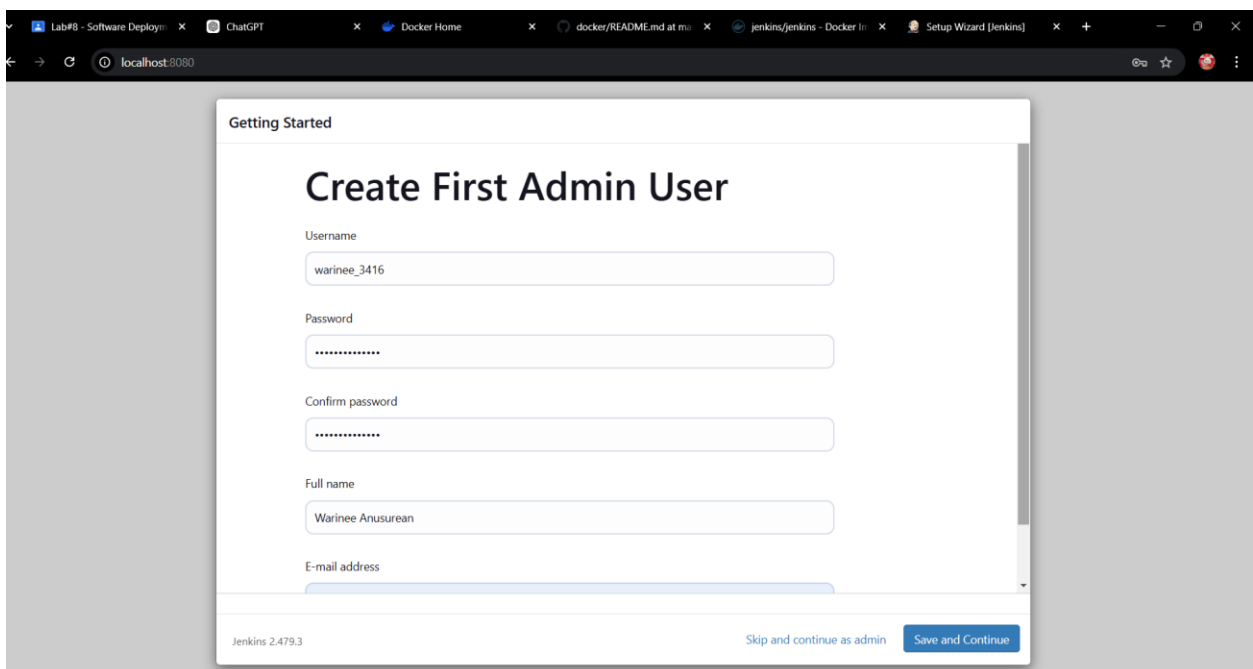
Jenkins initial setup is required. An admin user has been created and a password generated.  
Please use the following password to proceed to installation:

4e14fbdf3b2e49e7b26e711a71260f36

This may also be found at: /var/jenkins\_home/secrets/initialAdminPassword

4. เมื่อได้รับการยืนยันว่า Jenkins is fully up and running ให้เปิดบราวเซอร์ และป้อนที่อยู่เป็น localhost:8080
5. ทำการ Unlock Jenkins ด้วยรหัสผ่านที่ได้ในข้อที่ 3
6. สร้าง Admin User โดยใช้ username เป็นชื่อจริงของนักศึกษาพร้อมรหัสสี่ตัวท้าย เช่น somsri\_3062

**[Check point#13]** Capture หน้าจอที่แสดงผลการตั้งค่า



The screenshot shows the Jenkins 'Getting Started' page with the 'Create First Admin User' form. The form fields are as follows:

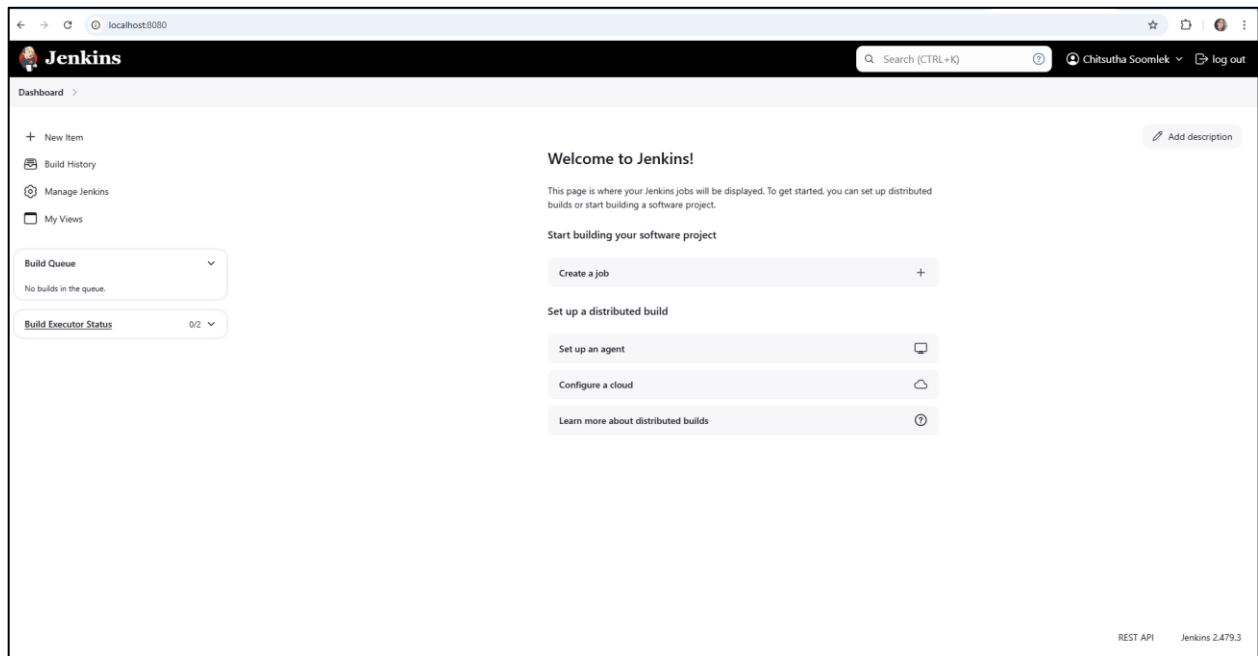
- Username: warinee\_3416
- Password: [masked]
- Confirm password: [masked]
- Full name: Warinee Anusurean
- E-mail address: [empty]

At the bottom of the form, there are two buttons: 'Skip and continue as admin' and 'Save and Continue'.

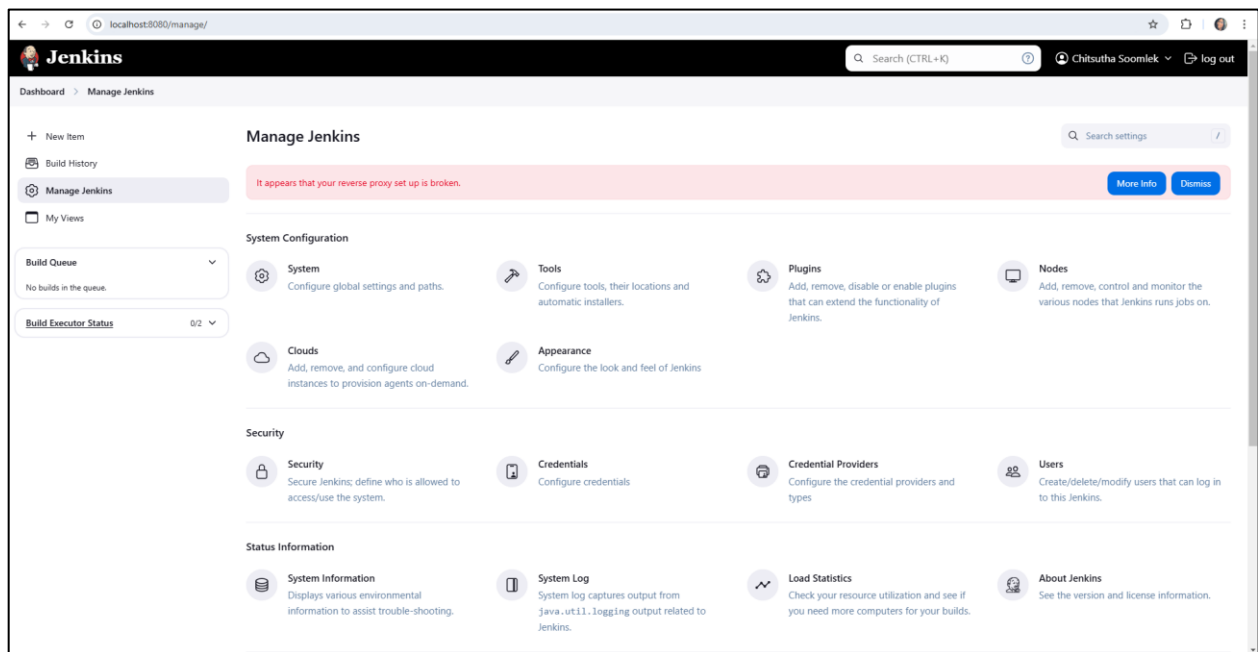
7. กำหนด Jenkins URL เป็น <http://localhost:8080/lab8>
8. เมื่อติดตั้งเรียบร้อยแล้วจะพบกันหน้า Dashboard ดังแสดงในภาพ



## Lab Worksheet

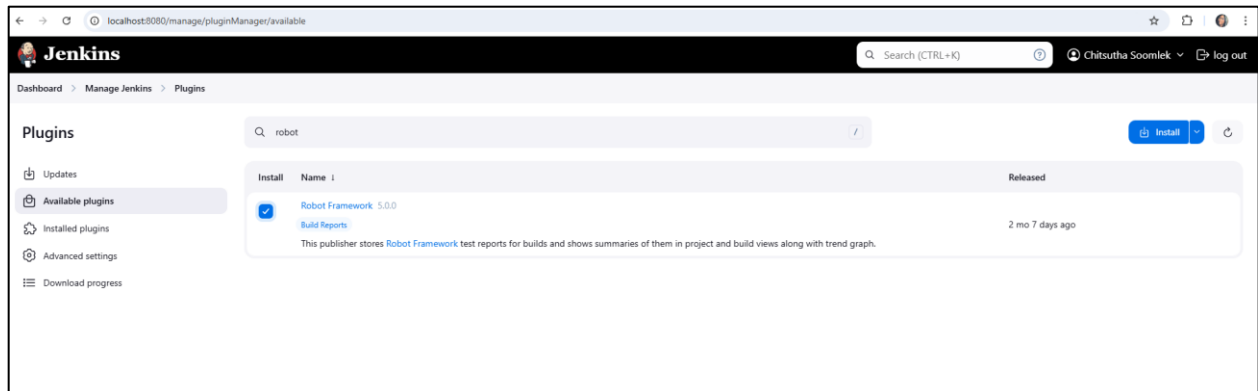


## 9. เลือก Manage Jenkins แล้วไปที่เมนู Plugins

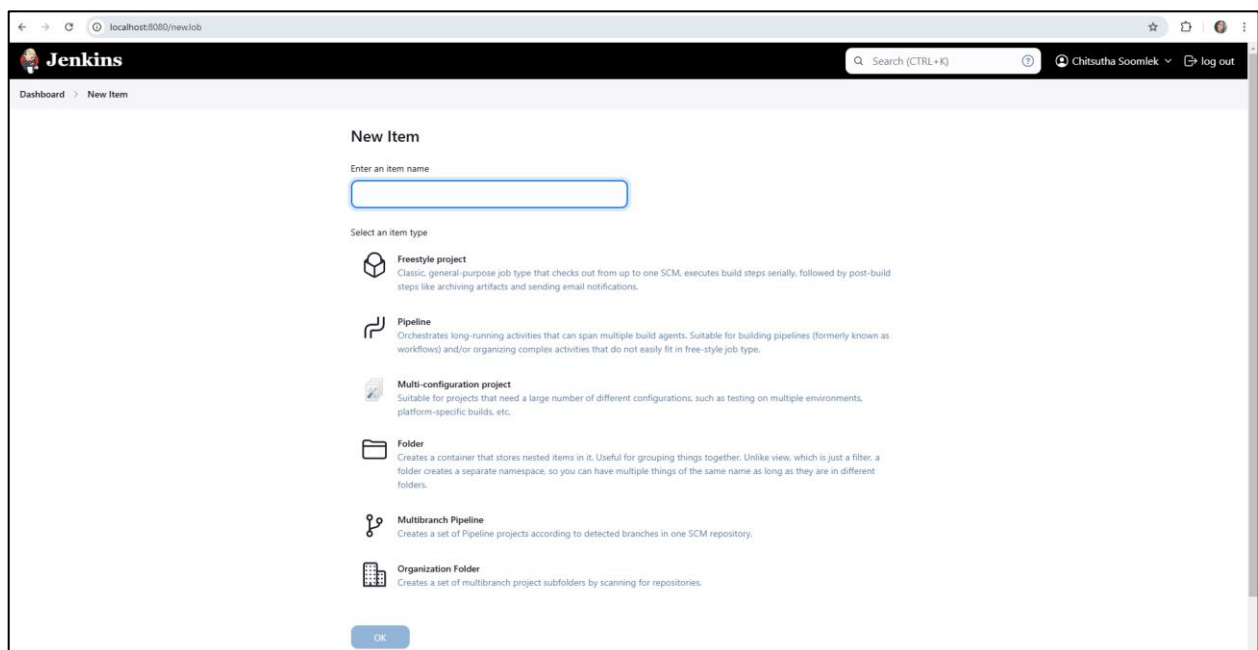


## Lab Worksheet

10. ไปที่เมนู Available plugins แล้วเลือกติดตั้ง Robotframework เพิ่มเติม



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



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 ไปถึงไฟล์ให้เรียบร้อยด้วย)

**[Check point#14]** Capture หน้าจอแสดงการตั้งค่า พร้อมกับตอบคำถามต่อไปนี้

## Getting Started

## Instance Configuration

Jenkins 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.

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.

The screenshot shows the Jenkins web interface for configuring a new instance. At the top, it says 'Jenkins 2.479.3' and has a 'Save and Finish' button. The main header includes the Jenkins logo, a search bar, and user information 'Warinee Anusuran'. The breadcrumb trail is 'Dashboard > UAT > Configuration'. On the left, the 'Configure' sidebar shows options: General (selected), Source Code Management, Build Triggers, Build Environment, Build Steps, and Post-build Actions. The 'General' tab is active, showing a 'Description' field with 'Lab 8.5'. Below this, there are checkboxes for 'Discard old builds' (unchecked) and 'GitHub project' (checked). The 'GitHub project' section shows a 'Project url' field with the value 'https://github.com/warinee341-6/Lab8'. At the bottom, there is an 'Advanced' dropdown and a checkbox for 'This project is parameterized' (unchecked).

## Lab Worksheet

## 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 Wednesday, January 29, 2025 at 1:05:29 PM Coordinated Universal Time; would next run at Wednesday, January 29, 2025 at 1:20:29 PM Coordinated Universal Time.

☐ GitHub hook trigger for GITScm polling ?☐ Poll SCM ?

## Source Code Management

☐ None☒ Git ?

Repositories ?

Repository URL ?

<https://github.com/warinee341-6/Lab8.git>

Credentials ?

- none -

+ Add

Advanced ▾

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?

\*/main

Save

Apply

## Lab Worksheet

## Build Steps

**Execute shell** ?

Command

[See the list of available environment variables](#)

robot valid\_login.robot

Advanced ▾

Add build step ▾

## Post-build Actions

**Publish Robot Framework test results** ?

Directory of Robot output

Path to directory containing robot xml and html files (relative to build workspace)

Advanced ▾ Edited

Thresholds for build result ?

🟡 %

20.0

🟢 %

80.0

☒ DEPRECATED! THIS FLAG DOES NOTHING! - Use thresholds for critical tests only

☐ Include skipped tests in total count for thresholds

Add post-build action ▾

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

robot <ชื่อไฟล์>.robot เช่น robot resource.robot

**Post-build action:** เพิ่ม Publish Robot Framework test results -> ระบุไดเรกทอรีที่เก็บไฟล์ผลการทดสอบโดย Robot framework ในรูป xml และ html -> ตั้งค่า Threshold เป็น % ของการทดสอบที่ไม่ผ่าน

## Lab Worksheet

แล้วนับว่าซอฟต์แวร์มีปัญหา -> 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

The screenshot shows the Jenkins dashboard for a pipeline named 'UAT'. The top navigation bar includes the Jenkins logo, a search bar, and user information. The left sidebar contains links for 'New Item', 'Build History', 'Manage Jenkins', and 'My Views'. The main content area displays a table of builds with columns: S (Status), W (Webhook), Name, Last Success, Last Failure, Last Duration, and Robot Results + Duration Trend. Build #11 is shown with a red status icon, a duration of 1.4 sec, and a green bar indicating a successful test result.

Below the dashboard, a detailed view of build #11 is shown. The build is titled '#11 (29 ม.ค. 2025 14:01:40)' and is marked as failed. The build was started by user 'Warinee Anusurean'. The console output shows the build duration and the total time from scheduled to completion. The build is linked to a GitHub repository: 'https://github.com/warinee341-6/Lab8.git'. The Robot Test Summary shows a total of 6 tests, all passed, with a pass percentage of 100.0. The build is also linked to a report and log.

## Lab Worksheet

The screenshot displays the Docker Desktop application. The top bar shows the Docker logo, a search bar, and system icons. The left sidebar contains navigation options: Containers, Images, Volumes, Builds, Docker Hub, Docker Scout, and Extensions. The main area is titled 'Containers' and shows a table of running containers.

Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
dreamy_yonath	101fc8c16c7	myapp_6533803416	3000:3000	0%		[Stop] [Refresh] [Delete]
quizzical_brattain	a2b41411e086	myapp_6533803416	3000:3000	0%		[Stop] [Refresh] [Delete]
thirsty_jennings	75a52fb537f2	myapp_6533803416	3000:3000	0%	3 hours ago	[Stop] [Refresh] [Delete]
fervent_ferri	f0a6c47e6bf4	jenkins/jenkins:its-jdk17	50000:50000 Show all ports (2)	0.35%	2 hours ago	[Stop] [Refresh] [Delete]

Below the containers table is a 'Terminal' window showing logs for the 'fervent\_ferri' container. The logs indicate the installation of Jenkins plugins and the completion of the installation process.

```

2025-01-29 11:59:17.338+0000 [id=382] INFO h.model.UpdateCenter$DownloadJob#run: Starting the installation of robot on behalf of warinee_3416
2025-01-29 11:59:17.339+0000 [id=382] INFO h.n.UpdateCenter$UpdateCenterConfiguration#download: Downloading robot
2025-01-29 11:59:20.445+0000 [id=382] INFO h.model.UpdateCenter$DownloadJob#run: Installation successful: robot
2025-01-29 11:59:20.712+0000 [id=415] INFO jenkins.InitReactorRunner$1#onAttained: System config adapted
2025-01-29 11:59:20.713+0000 [id=418] INFO jenkins.InitReactorRunner$1#onAttained: Loaded all jobs
2025-01-29 11:59:20.714+0000 [id=392] INFO jenkins.InitReactorRunner$1#onAttained: Configuration for all jobs updated
2025-01-29 11:59:20.714+0000 [id=397] INFO jenkins.InitReactorRunner$1#onAttained: Completed initialization
2025-01-29 11:59:20.719+0000 [id=382] INFO h.n.UpdateCenter$CompleteBatchJob#run: Completed installation of 1 plugins in 3.3 sec
  
```

The bottom part of the screenshot shows a 'Console Output' window for the 'UAT' job. The output shows the build process, including fetching changes from a remote repository, checking out the code, and running the build script. The build failed due to a 'robot: not found' error.

```

Started by user Warinee Anusurean
Running as SYSTEM
Building in workspace /var/jenkins_home/workspace/UAT
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /var/jenkins_home/workspace/UAT/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/warinee341-6/Lab8.git # timeout=10
Fetching upstream changes from https://github.com/warinee341-6/Lab8.git
> git --version # timeout=10
> git --version # 'git version 2.39.5'
> git fetch --tags --force --progress -- https://github.com/warinee341-6/Lab8.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/main (commit) # timeout=10
Checking out Revision d762faa42d04e3b276f1f57ee3cd6efc9acf2fe (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f d762faa42d04e3b276f1f57ee3cd6efc9acf2fe # timeout=10
Commit message: 'Added Lab7 -robot File'
> git rev-list --no-walk d762faa42d04e3b276f1f57ee3cd6efc9acf2fe # timeout=10
[UAT] $ /bin/sh -xe /tmp/jenkins14903694498044501968.sh
+ robot valid_login_robot
/tmp/jenkins14903694498044501968.sh: 2: 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:
Done!
-Copying log files to build dir:
Done!
-Assigning results to build:
Done!
-Checking thresholds:
Done!
Done publishing Robot results.
Finished: FAILURE
  
```