

## Lab Worksheet

ชื่อ-นามสกุล ธีรชาติ วิชัย รหัสนักศึกษา 653380019-1 Section 2

## 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

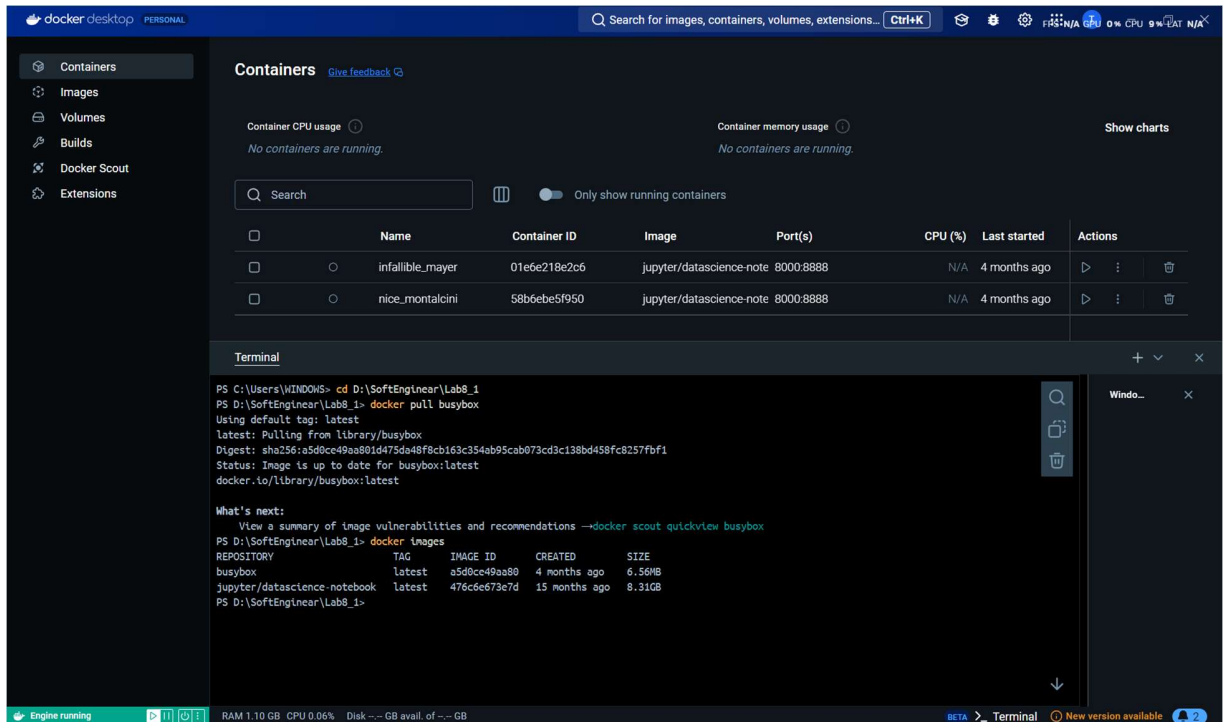
## แบบฝึกปฏิบัติที่ 8.1 Hello world - รัน Container จาก Docker image

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 Image name



(2) Tag ที่ใช้บ่งบอกถึงอะไร version

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

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

## Lab Worksheet

The screenshot shows the Docker Desktop interface. On the left is a sidebar with navigation options: Containers, Images, Volumes, Builds, Docker Scout, and Extensions. The main area is titled 'Containers' and shows 'No containers are running.' Below this is a search bar and a toggle for 'Only show running containers'. A terminal window is open, displaying the following commands and output:

```

PS D:\SoftEngineer\Lab8_1> docker pull busybox
Using default tag: latest
latest: Pulling from library/busybox
Digest: sha256:a5d0ce49aa801d475d448f8cb163c354ab95cab073cd3c138bd458fc8257fbf1
Status: Image is up to date for busybox:latest
docker.io/library/busybox:latest

What's next:
View a summary of image vulnerabilities and recommendations --docker scout quickview busybox
sh: ls -la: not found
/ # exit
PS D:\SoftEngineer\Lab8_1> docker run busybox echo "Hello ชื่นชมงานสฤพของนักศึกษ from busybox"
Hello ชื่นชมงานสฤพของนักศึกษ from busybox
PS D:\SoftEngineer\Lab8_1> docker run busybox echo "Hello ชาตล รวิษ from busybox"
Hello ชาตล รวิษ from busybox
PS D:\SoftEngineer\Lab8_1> docker ps -a

```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
488283bdfbea	busybox	"echo 'Hello ชาตล ...'"	11 seconds ago	Exited (0) 10 seconds ago		nostalgi
c_borg	busybox	"echo 'Hello ชื่นชม...'"	50 seconds ago	Exited (0) 49 seconds ago		trusting
1a9683cd2923	busybox	"sh"	About a minute ago	Exited (127) About a minute ago		eloquent
77a52031840b	busybox	"sh"	About a minute ago	Exited (127) About a minute ago		eloquent
bc9141317b46	busybox	"sh"	2 minutes ago	Exited (0) 2 minutes ago		trusting
01e6e218e2c6	jupyter/datascience-notebook	"tini -g -- start-no..."	3 months ago	Exited (0) 3 months ago		infallib
1e_mayer	jupyter/datascience-notebook	"tini -g -- start-no..."	3 months ago	Exited (0) 3 months ago		nice_mon
58b6ebc5f950	jupyter/datascience-notebook	"tini -g -- start-no..."	3 months ago	Exited (0) 3 months ago		nice_mon
talcini	jupyter/datascience-notebook	"tini -g -- start-no..."	3 months ago	Exited (0) 3 months ago		nice_mon

At the bottom of the terminal, it shows 'PS D:\SoftEngineer\Lab8\_1>'.

(1) เมื่อใช้ option -it ในคำสั่ง run ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป

จะใช้ command ใน container ได้โดยตรง

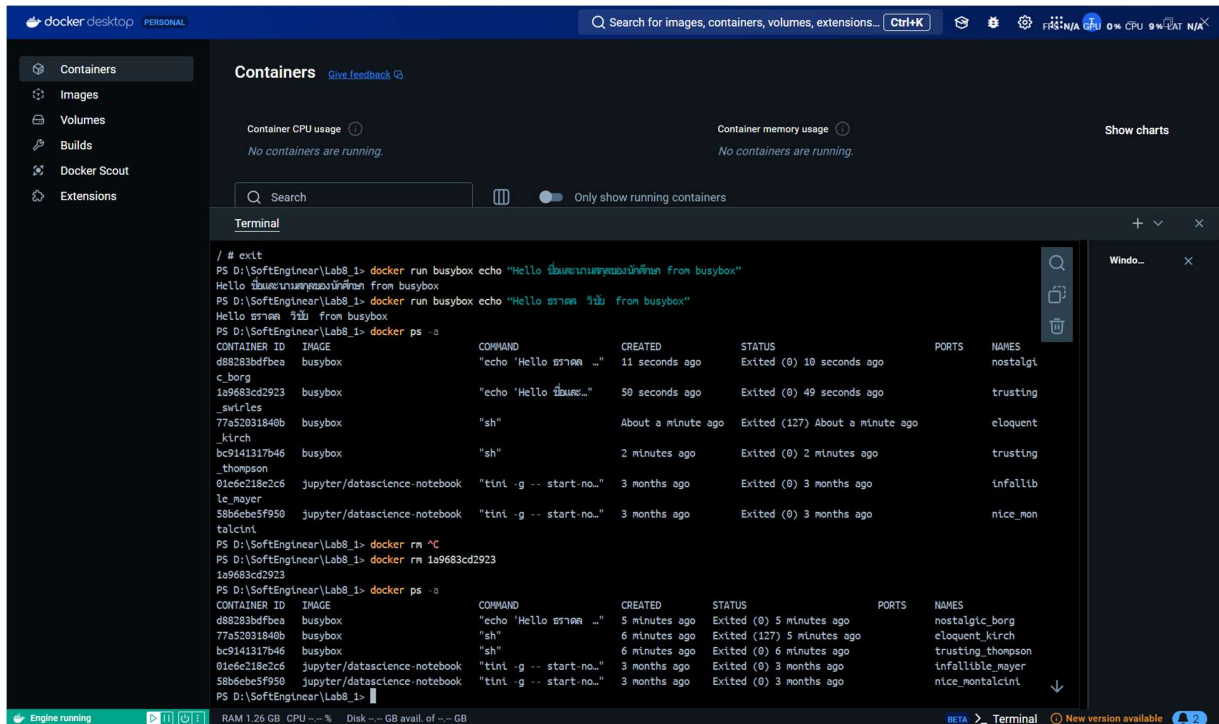
(2) คอลัมน์ STATUS จากการรันคำสั่ง docker ps -a แสดงถึงข้อมูลอะไร

แสดงสถานะของทุก container และจะบอกสถานะของการทำงานของ container นั้นๆ

## Lab Worksheet

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 เพื่อให้เป็น 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
```

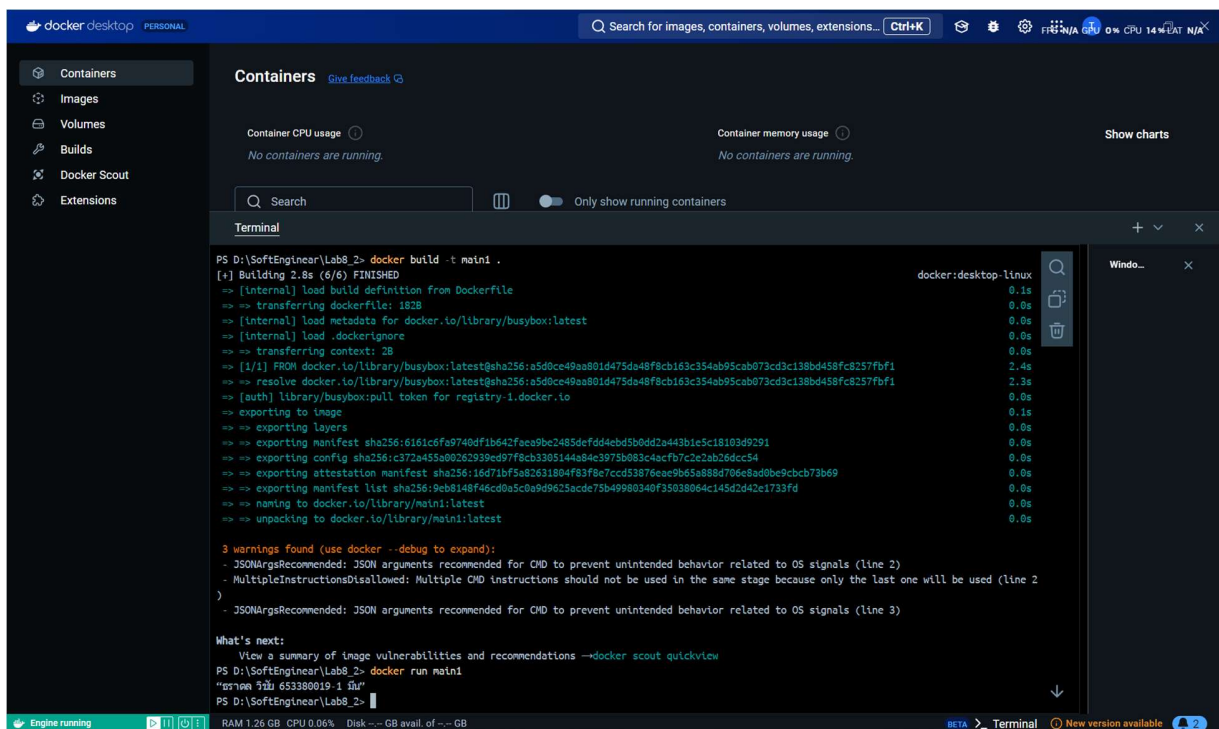
แล้วใช้ Text Editor ในการใส่เนื้อหาแทน

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



```
PS D:\SoftEng\Lab8_2> docker build -t main1 .
[+] Building 2.8s (6/6) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 182B
=> [internal] load metadata for docker.io/library/busybox:latest
=> [internal] load .dockerignore
=> transferring context: 2B
=> [1/1] FROM docker.io/library/busybox:latest@sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138b0458fc6257fbf1
=> resolve docker.io/library/busybox:latest@sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138b0458fc6257fbf1
=> [auth] library/busybox:pull token for registry-1.docker.io
=> exporting to image
=> exporting layers
=> exporting manifest sha256:6161c9fa9740df1b642f0ea9be2485defdd4ebd5b0dd2a443b1e5c18103d9291
=> exporting config sha256:c372a455a00263939ed97f8cb3305144a84c3975b083c4acfb7c2e2ab26d4cc54
=> exporting attestation manifest sha256:16d71bf5a82631804f83f8e7cc43376eae9b65a888d706e8ad0be9c9cb73b69
=> exporting manifest list sha256:9eb8148f46cd0b5c0a0949625acde75b49980340f35038094c145d2d42e1733fd
=> naming to docker.io/library/main1:latest
=> unpacking to docker.io/library/main1:latest

3 warnings found (use docker --debug to expand):
- JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 2)
- MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the same stage because only the last one will be used (line 2)
- 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 D:\SoftEng\Lab8_2> docker run main1
"สมชาย ภูมิ 653380019-1 ภูมิ"
PS D:\SoftEng\Lab8_2>
```

## Lab Worksheet

- (1) คำสั่งที่ใช้ในการ run คือ

`docker run main1`

- (2) Option -t ในคำสั่ง `$ docker build` ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป  
ตั้งชื่อ และติด tag ให้ docker image ที่สร้างขึ้นมา

### 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



## Lab Worksheet

\$ 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

```
Terminal
PS D:\SoftEnginar\Lab8_3> notepad .\Dockerfile
PS D:\SoftEnginar\Lab8_3> docker build -t tharadolwchai6533800191/lab8 .
[+] Building 4.0s (6/6) FINISHED
=> [internal] load build definition from Dockerfile 0.0s
=> => transferring dockerfile: 197B 0.0s
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 2) 0.0s
=> WARN: MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the same stage because only the last one will be used (line 2) 0.0s
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 3) 0.0s
=> [internal] load metadata for docker.io/library/busybox:latest 0.0s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> CACHED [1/1] FROM docker.io/library/busybox:latest@sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138bd458fc8257fbf1 3.7s
=> => resolve docker.io/library/busybox:latest@sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138bd458fc8257fbf1 3.7s
=> [auth] library/busybox:pull token for registry-1.docker.io 0.0s
=> exporting to image 0.1s
=> => exporting layers 0.0s
=> => exporting manifest sha256:46e8b0b38866a9807ae1477cfb2b0a9901097eeb4788912f1772832a104a7698 0.0s
=> => exporting config sha256:d82eaf9cbd4672be4d17715c72bee2775f4a8385439d712f1d3418e7e3a10d9 0.0s
=> => exporting attestation manifest sha256:6fb764d0f5e2492291ea4ec242f3bea4b68f1df1decfb0229bb189c4eda2da2c 0.0s
=> => exporting manifest list sha256:18091778b84fc4ea9269df07ded7084bcf858e03b891216d4907ab29427ddf16 0.0s
=> => naming to docker.io/tharadolwchai6533800191/lab8:latest 0.0s
=> => unpacking to docker.io/tharadolwchai6533800191/lab8:latest 0.0s

3 warnings found (use docker --debug to expand):
- JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 2)
- MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the same stage because only the last one will be used (line 2)
- JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 3)
```

```
Terminal
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 3) 0.0s
=> [internal] load metadata for docker.io/library/busybox:latest 0.0s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> CACHED [1/1] FROM docker.io/library/busybox:latest@sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138bd458fc8257fbf1 3.7s
=> => resolve docker.io/library/busybox:latest@sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138bd458fc8257fbf1 3.7s
=> [auth] library/busybox:pull token for registry-1.docker.io 0.0s
=> exporting to image 0.1s
=> => exporting layers 0.0s
=> => exporting manifest sha256:46e8b0b38866a9807ae1477cfb2b0a9901097eeb4788912f1772832a104a7698 0.0s
=> => exporting config sha256:d82eaf9cbd4672be4d17715c72bee2775f4a8385439d712f1d3418e7e3a10d9 0.0s
=> => exporting attestation manifest sha256:6fb764d0f5e2492291ea4ec242f3bea4b68f1df1decfb0229bb189c4eda2da2c 0.0s
=> => exporting manifest list sha256:18091778b84fc4ea9269df07ded7084bcf858e03b891216d4907ab29427ddf16 0.0s
=> => naming to docker.io/tharadolwchai6533800191/lab8:latest 0.0s
=> => unpacking to docker.io/tharadolwchai6533800191/lab8:latest 0.0s

3 warnings found (use docker --debug to expand):
- JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 2)
- MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the same stage because only the last one will be used (line 2)
- 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 D:\SoftEnginar\Lab8_3> docker run tharadolwchai6533800191/lab8 .
docker: Error response from daemon: failed to create shim task: OCI runtime create failed: runc create failed: unable to start container process: exec: ".": executable file not found in $PATH: unknown.
PS D:\SoftEnginar\Lab8_3> docker run tharadolwchai6533800191/lab8
"สราดล รันนั 653380019-1"
PS D:\SoftEnginar\Lab8_3>
```

## Lab Worksheet

6. ทำการ Push ตัว Docker image ไปไว้บน Docker Hub โดยการใช้คำสั่ง

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

ในกรณีที่ติดปัญหาไม่ได้ Login ไว้ก่อน ให้ใช้คำสั่งต่อไปนี้ เพื่อ Login ก่อนทำการ Push

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

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

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

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

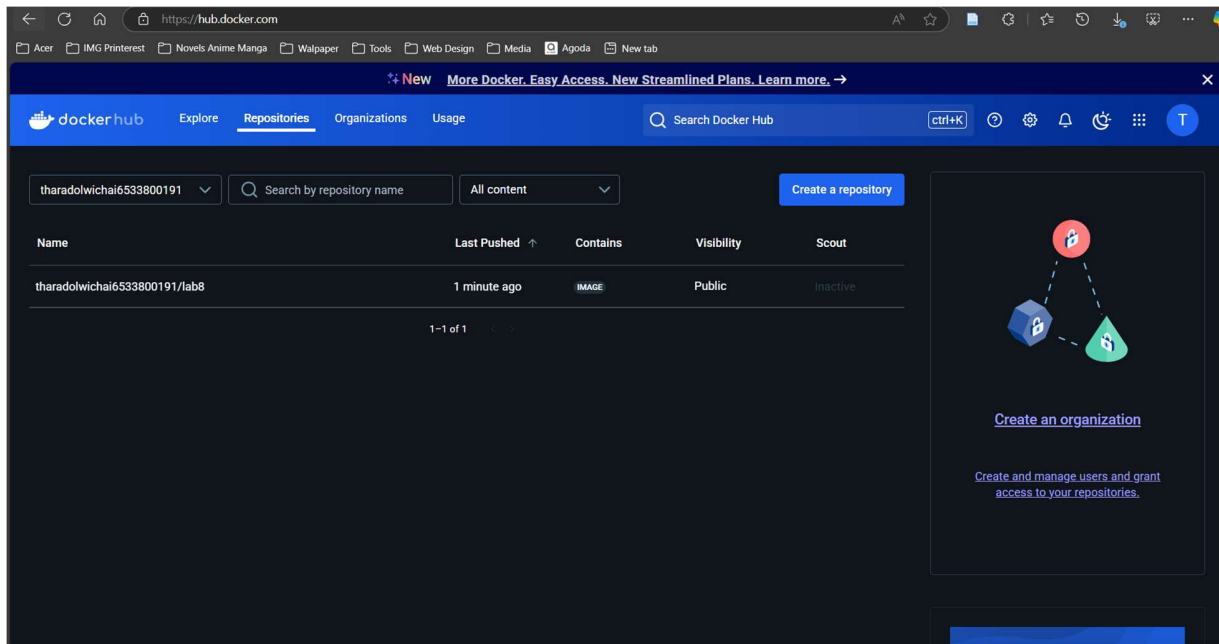
```
Terminal
=> [auth] library/busybox:pull token for registry-1.docker.io 0.0s
=> exporting to image 0.1s
=> => exporting layers 0.0s
=> => exporting manifest sha256:46e8b0b38866a9807ae1477cfb2b0a9901097eeb4788912f1772832a104a7698 0.0s
=> => exporting config sha256:d827eaf9cbd4672be4d17715c72bee2775f4a8385439d712fd3418e7e3a10d9 0.0s
=> => exporting attestation manifest sha256:6fb764d0f5e2492291ea4ec242f3bea4b68f1df1decfb0229bb189c4eda2da2c 0.0s
=> => exporting manifest list sha256:18091778b84fc4ea9269df07ded7084bcf858e03b891216d4907ab29427ddf16 0.0s
=> => naming to docker.io/tharadolwichai6533800191/lab8:latest 0.0s
=> => unpacking to docker.io/tharadolwichai6533800191/lab8:latest 0.0s

3 warnings found (use docker --debug to expand):
- JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 2)
- MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the same stage because only the last one will be used (line 2)
- 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 D:\SoftEngineer\Lab8_3> docker run tharadolwichai6533800191/lab8 .
docker: Error response from daemon: failed to create shim task: OCI runtime create failed: runc create failed: unable to start container process: exec: ".": executable file not found in $PATH: unknown.
PS D:\SoftEngineer\Lab8_3> docker run tharadolwichai6533800191/lab8
"ตราดล วิชัย 653380019-1"
PS D:\SoftEngineer\Lab8_3> docker push tharadolwichai6533800191/lab8
Using default tag: latest
The push refers to repository [docker.io/tharadolwichai6533800191/lab8]
5781bc26ef28: Pushed
9c0abc9c5bd3: Mounted from library/busybox
latest: digest: sha256:18091778b84fc4ea9269df07ded7084bcf858e03b891216d4907ab29427ddf16 size: 855
PS D:\SoftEngineer\Lab8_3>
```



## 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

```

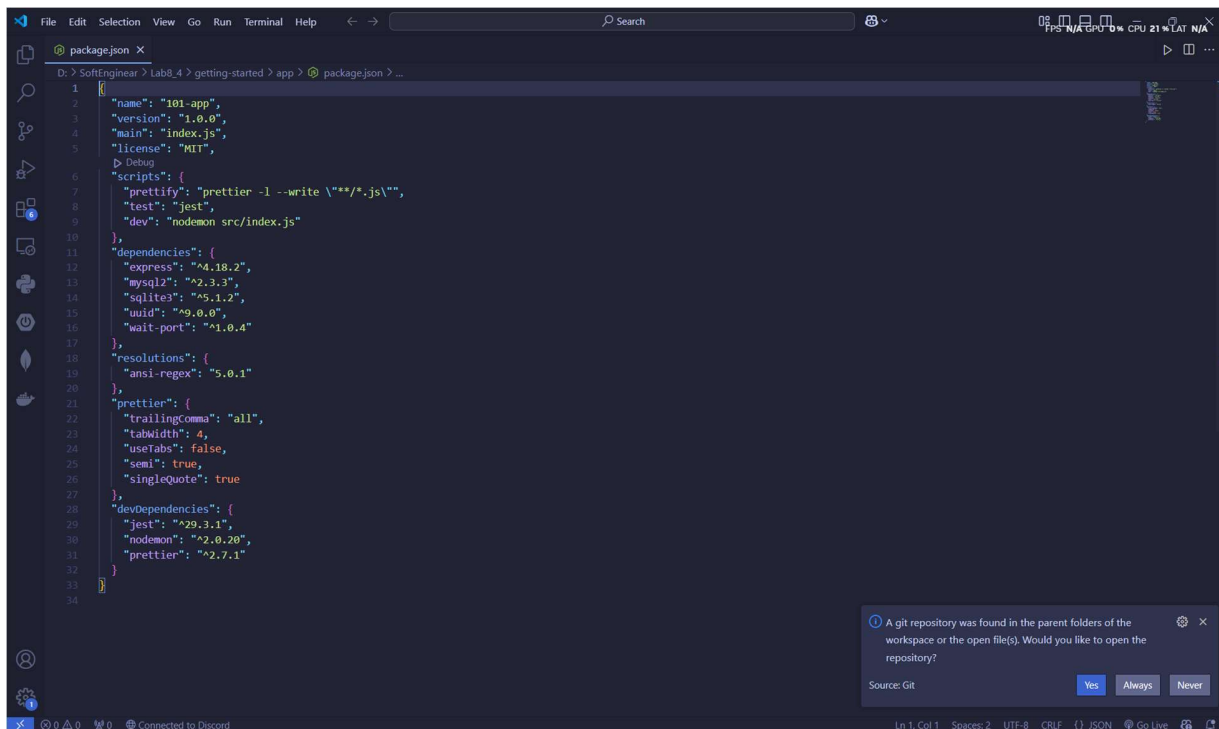
PS D:\SoftEngineer\Lab8_3> cd ..
PS D:\SoftEngineer> mkdir Lab8_4

Directory: D:\SoftEngineer

Mode                LastWriteTime         Length Name
----                -
d-----          1/28/2025   2:04 PM             Lab8_4

PS D:\SoftEngineer> cd Lab8_4
PS D:\SoftEngineer\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 | 3.15 MiB/s, done.
Resolving deltas: 100% (523/523), done.
PS D:\SoftEngineer\Lab8_4>

```



```

{
  "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": "^9.0.0",
    "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 พร้อมกับใส่เนื้อหาดังต่อไปนี้ลงไปไฟล์

FROM node:18-alpine

WORKDIR /app

COPY . .

RUN yarn install --production

## Lab Worksheet

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

The screenshot shows a Windows Terminal window with the following content:

```

Terminal
PS D:\SoftEnginear\Lab8_4> cd .\getting-started\app\
PS D:\SoftEnginear\Lab8_4\getting-started\app> notepad Dockerfile
PS D:\SoftEnginear\Lab8_4\getting-started\app> ls

Directory: D:\SoftEnginear\Lab8_4\getting-started\app

Mode                LastWriteTime         Length Name
----                -
d-----          1/28/2025   2:05 PM             spec
d-----          1/28/2025   2:05 PM             src
-a-----          1/28/2025   2:10 PM           119 Dockerfile.txt
-a-----          1/28/2025   2:05 PM           678 package.json
-a-----          1/28/2025   2:05 PM        150541 yarn.lock

PS D:\SoftEnginear\Lab8_4\getting-started\app> mv .\Dockerfile.txt .\Dockerfile
PS D:\SoftEnginear\Lab8_4\getting-started\app> ls

Directory: D:\SoftEnginear\Lab8_4\getting-started\app

Mode                LastWriteTime         Length Name
----                -
d-----          1/28/2025   2:05 PM             spec
d-----          1/28/2025   2:05 PM             src
-a-----          1/28/2025   2:10 PM           119 Dockerfile
-a-----          1/28/2025   2:05 PM           678 package.json
  
```

## Lab Worksheet

```

Terminal

Mode                LastWriteTime         Length Name
-----
d-----          1/28/2025    2:05 PM             spec
d-----          1/28/2025    2:05 PM             src
-a-----          1/28/2025    2:10 PM             119 Dockerfile
-a-----          1/28/2025    2:05 PM             678 package.json
-a-----          1/28/2025    2:05 PM            150541 yarn.lock

PS D:\SoftEnginear\Lab8_4\getting-started\app> cat .\Dockerfile
FROM node:18-alpine
WORKDIR /app
COPY . .
RUN yarn install --production
CMD ["node", "src/index.js"]
EXPOSE 3000
PS D:\SoftEnginear\Lab8_4\getting-started\app> docker build -t myapp 6533800191
[+] Building 0.0s (0/0)                                     docker:desktop-linux
ERROR: unable to prepare context: path "6533800191" not found
PS D:\SoftEnginear\Lab8_4\getting-started\app> docker build -t myapp_6533800191 .
[+] Building 33.6s (10/10) FINISHED                         docker:desktop-linux
=> [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

```

```

Terminal

=> [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
=> sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15bf426d01bc1 1.26MB / 1.26MB
=> sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce368990ca771 444B / 444B
=> sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e0da02525e63167c 40.01MB / 40.01MB
=> sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c24403d592134f0 3.64MB / 3.64MB
=> => extracting sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e0da02525e63167c
=> => extracting sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15bf426d01bc1
=> => extracting sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce368990ca771
=> [internal] load build context
=> => transferring context: 4.62MB
=> [2/4] WORKDIR /app
=> [3/4] COPY . .
=> [4/4] RUN yarn install --production
=> exporting to image
=> => exporting layers
=> => exporting manifest sha256:15265792ddcc2df417c8fc05fc6705d199334e7a3156ee5ecf55e320a3629b6
=> => exporting config sha256:a7e2f39ffee148333caf5fbf1fcaabbf8cac7b2a503e241ab519901450a82f24
=> => exporting attestation manifest sha256:364a54c854d57d42a52d37ad3dccc03c87525518177fe630fcdc273adb65de80a
=> => exporting manifest list sha256:04639ba1752b662f6e4f78bd76bde06eced36263357d8025e073a117506aa5d9
=> => naming to docker.io/library/myapp_6533800191:latest
=> => unpacking to docker.io/library/myapp_6533800191:latest

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

```

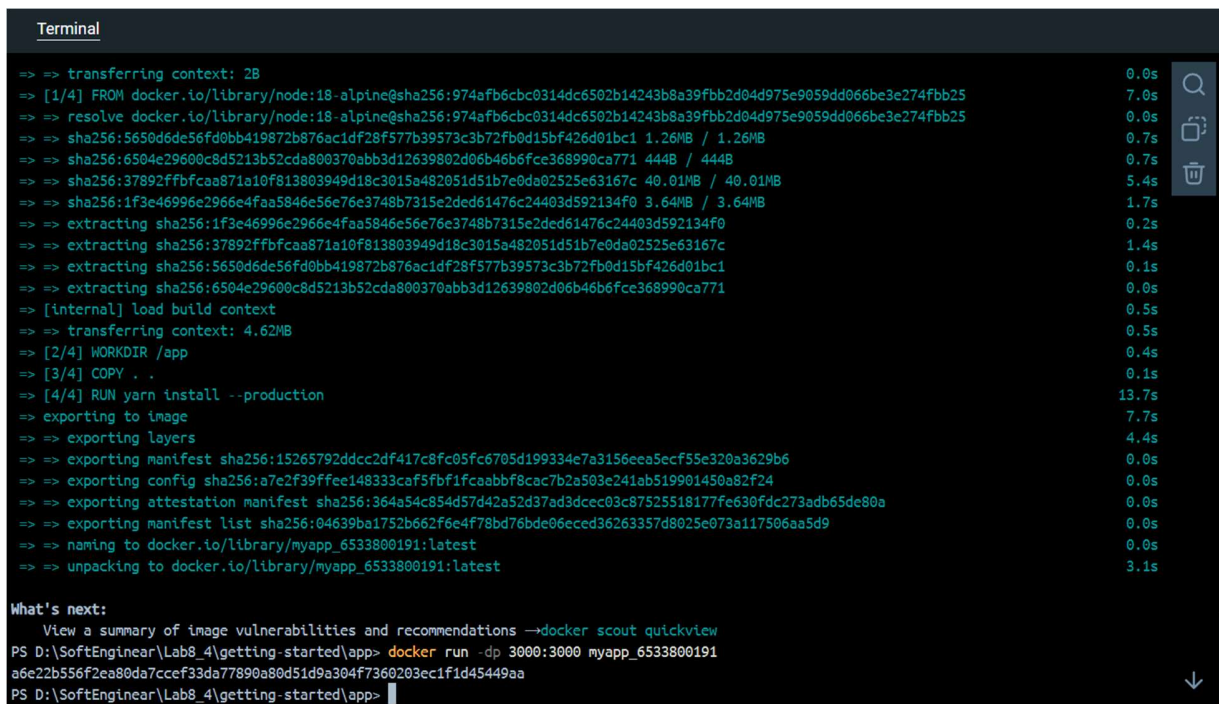
## Lab Worksheet

6. ทำการ Start ตัว Container ของแอปพลิเคชันที่สร้างขึ้น โดยใช้คำสั่ง

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

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

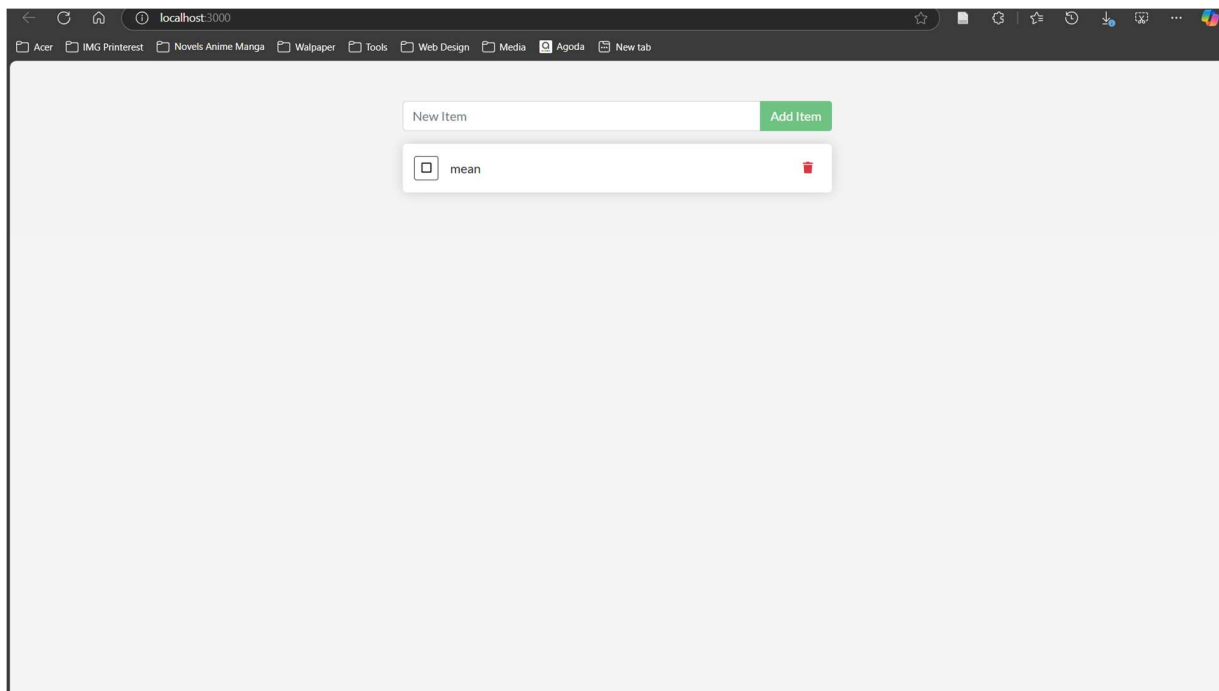
[Check point#9] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงผลลัพธ์ที่ได้บน Browser และ Dashboard ของ Docker desktop



```
Terminal
=> => transferring context: 2B 0.0s
=> [1/4] FROM docker.io/library/node:18-alpine@sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d975e9059dd0666e3e274fbb25 7.0s
=> => resolve docker.io/library/node:18-alpine@sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d975e9059dd0666e3e274fbb25 0.0s
=> => sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15bf426d01bc1 1.26MB / 1.26MB 0.7s
=> => sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce368990ca771 444B / 444B 0.7s
=> => sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e0da02525e63167c 40.01MB / 40.01MB 5.4s
=> => sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c24403d592134f0 3.64MB / 3.64MB 1.7s
=> => extracting sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c24403d592134f0 0.2s
=> => extracting sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e0da02525e63167c 1.4s
=> => extracting sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15bf426d01bc1 0.1s
=> => extracting sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce368990ca771 0.0s
=> [internal] load build context 0.5s
=> => transferring context: 4.62MB 0.5s
=> [2/4] WORKDIR /app 0.4s
=> [3/4] COPY . . 0.1s
=> [4/4] RUN yarn install --production 13.7s
=> exporting to image 7.7s
=> => exporting layers 4.4s
=> => exporting manifest sha256:15265792ddcc2df417c8fc05fc6705d199334e7a3156eea5ecf55e320a3629b6 0.0s
=> => exporting config sha256:a7e2f39ffef148333caf5fbf1fcaabbbf8cac7b2a503e241ab519901450a82f24 0.0s
=> => exporting attestation manifest sha256:364a54c854d57d42a52d37ad3dcec03c87525518177fe630fcd273adb65de80a 0.0s
=> => exporting manifest list sha256:04639ba1752b662f6e4f78bd76bde06eced36263357d8025e073a117506aa5d9 0.0s
=> => naming to docker.io/library/myapp_6533800191:latest 0.0s
=> => unpacking to docker.io/library/myapp_6533800191:latest 3.1s

What's next:
View a summary of image vulnerabilities and recommendations → docker scout quickview
PS D:\SoftEnginear\Lab8_4\getting-started\app> docker run -dp 3000:3000 myapp_6533800191
a6e22b556f2ea80da7ccef33da77890a80d51d9a304f7360203ec1f1d45449aa
PS D:\SoftEnginear\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

`<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



## Lab Worksheet

```

Terminal
PS D:\SoftEngineer\Lab8_4\getting-started\app> docker build -t myapp_6533800191 .
[+] Building 25.1s (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
=> [internal] load build context
=> => transferring context: 8.11kB
=> CACHED [2/4] WORKDIR /app
=> [3/4] COPY . .
=> [4/4] RUN yarn install --production
=> exporting to image
=> => exporting layers
=> => exporting manifest sha256:1d0964e51d958112eab9ddee4596230249abe2dc92c8430cc5cdf9aa0e0dd4f1
=> => exporting config sha256:89cac45de4dab97c54065810eee5aa17e5b899391ec0e6dcbcd8e63051896731
=> => exporting attestation manifest sha256:e1c0230ff5ec018aa77e4ba53e0e65f0e2c3ac834d632b4667ccf7181baaf30b
=> => exporting manifest list sha256:94208937f9c03ed038684e3b6e27c5015a617720a16da35d7c5fcdf7ecd38d88
=> => naming to docker.io/library/myapp_6533800191:latest
=> => unpacking to docker.io/library/myapp_6533800191:latest
What's next:
View a summary of image vulnerabilities and recommendations →docker scout quickview
PS D:\SoftEngineer\Lab8_4\getting-started\app> docker run -dp 3000:3000 myapp_6533800191
5c876e27db090dae75d1dfe54568e7b9e1b03101ad36ad9c9acf90a21f71773
docker: Error response from daemon: driver failed programming external connectivity on endpoint pensive_lumiere (f325015d892959f2acb7f6bf12011ac3d15e86d5a75637f85869add33d1690c4): Bind for 0.0.0.0:3000 failed: port is already allocated.

```

(1) Error ที่เกิดขึ้นหมายความว่าอย่างไร และเกิดขึ้นเพราะอะไร

Port 3000 ถูก run อยู่ ยังไม่ได้ stop run เลยเกิด error ขึ้น

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

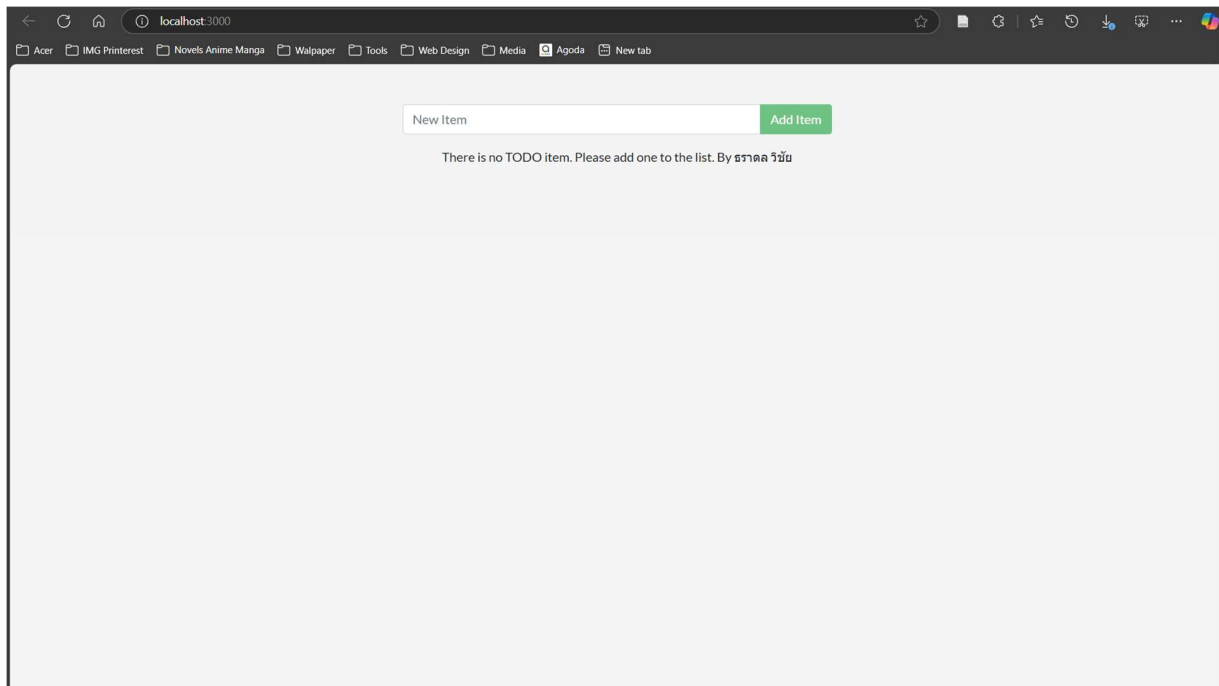
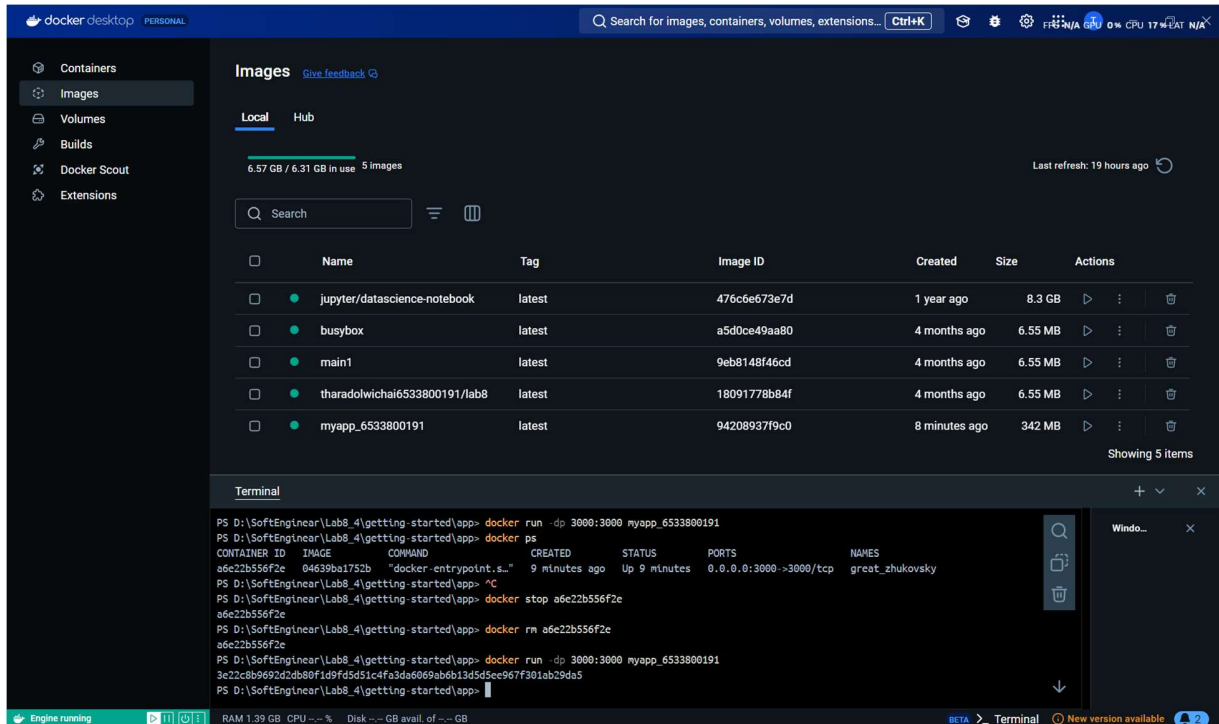
- ไปที่หน้าต่าง Containers
- เลือกไอคอนถังขยะในแถวของ Container ที่ต้องการจะลบ
- ยืนยันโดยการกด Delete forever

12. Start และรัน Container ตัวใหม่อีกครั้ง โดยใช้คำสั่งเดียวกันกับข้อ 6

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

## Lab Worksheet

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



## Lab Worksheet

## แบบฝึกปฏิบัติที่ 8.5: เริ่มต้นสร้าง Pipeline อย่างง่ายสำหรับการ Deploy ด้วย Jenkins

1. เปิด Command line หรือ Terminal บน Docker Desktop

2. ป้อนคำสั่งและทำการรัน container โดยผูกพอร์ต

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

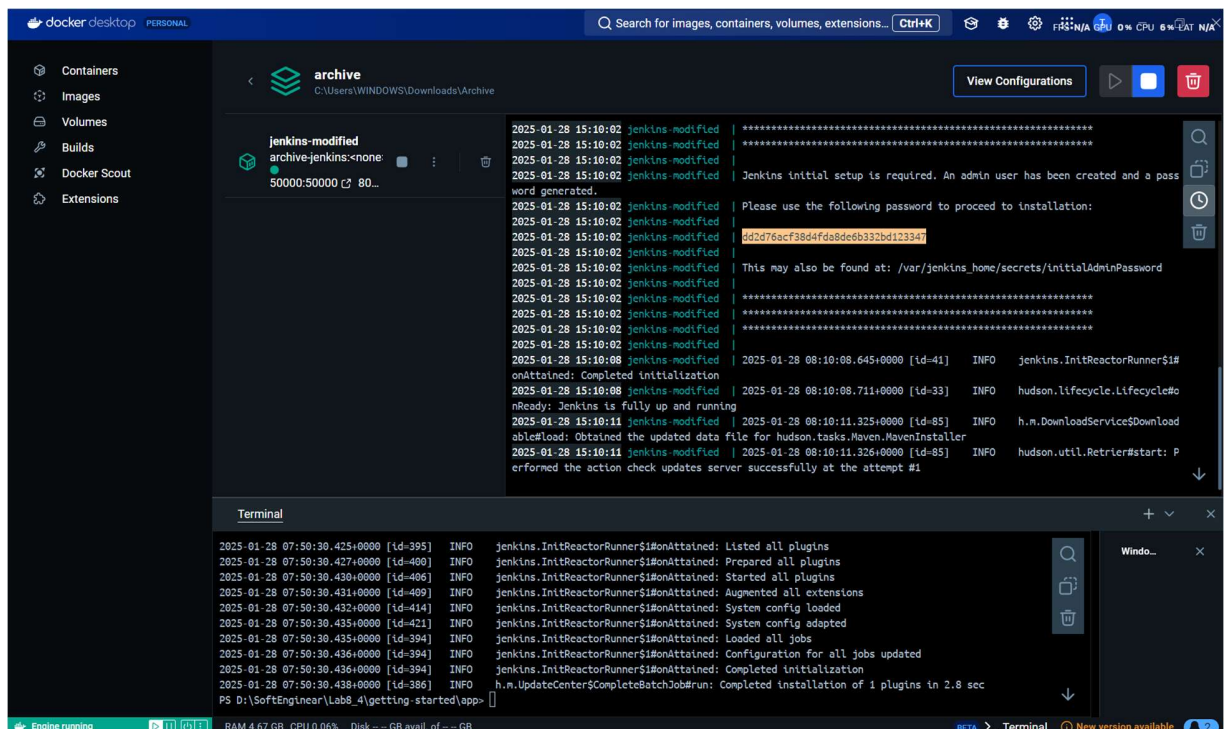
หรือ

```
$ 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

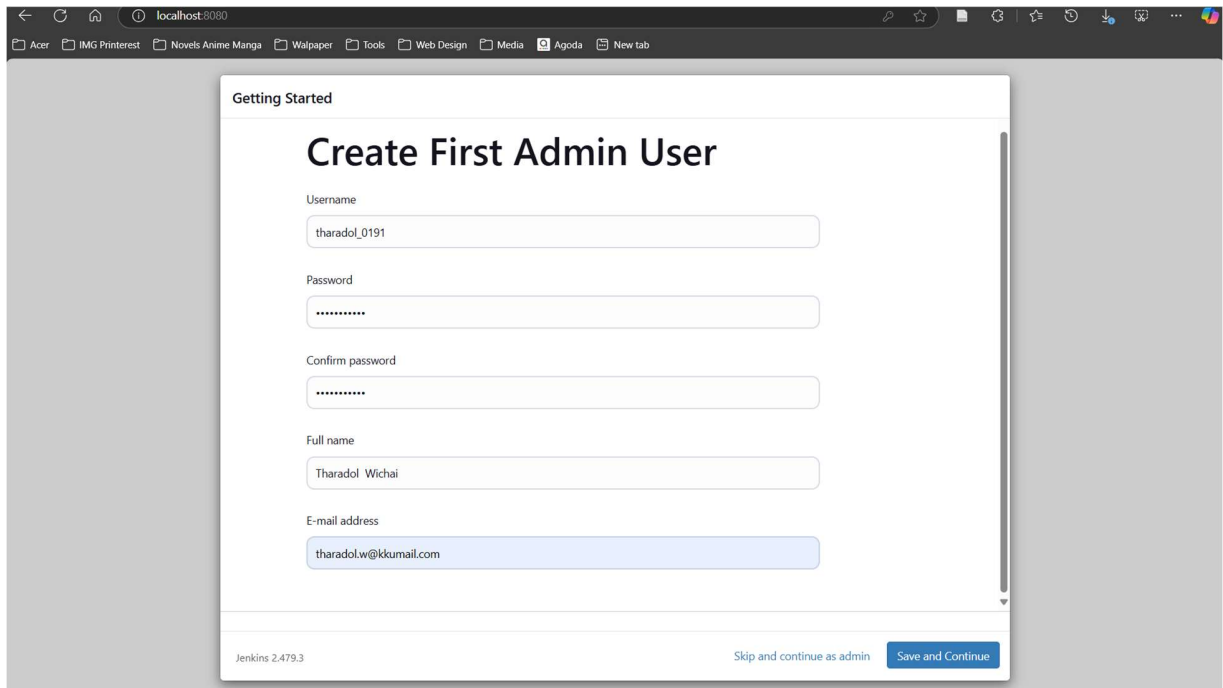
[Check point#12] Capture หน้าจอที่แสดงผล Admin password



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

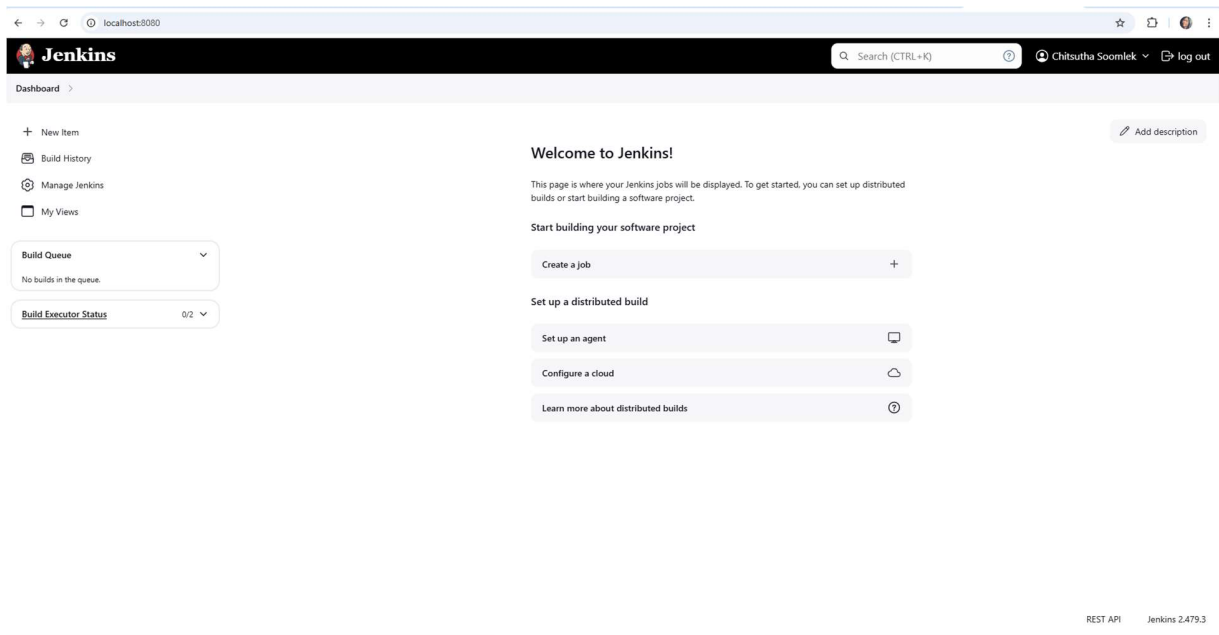
## Lab Worksheet

5. ทำการ Unlock Jenkins ด้วยรหัสผ่านที่ได้ในข้อที่ 3
  6. สร้าง Admin User โดยใช้ username เป็นชื่อจริงของนักศึกษาพร้อมรหัสสี่ตัวท้าย เช่น somsri\_3062
- [Check point#13] Capture หน้าจอที่แสดงผลการตั้งค่า

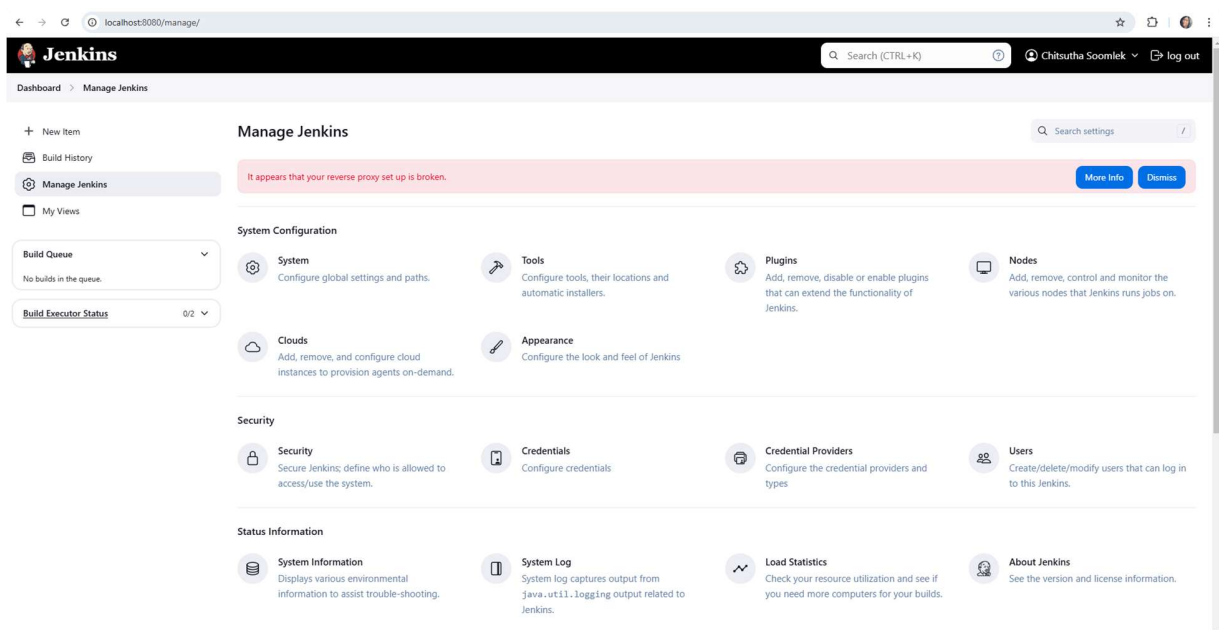


7. กำหนด Jenkins URL เป็น <http://localhost:8080/lab8>
8. เมื่อติดตั้งเรียบร้อยแล้วจะพบหน้าจอ Dashboard ดังแสดงในภาพ

## Lab Worksheet



## 9. เลือก Manage Jenkins แล้วไปที่เมนู Plugins

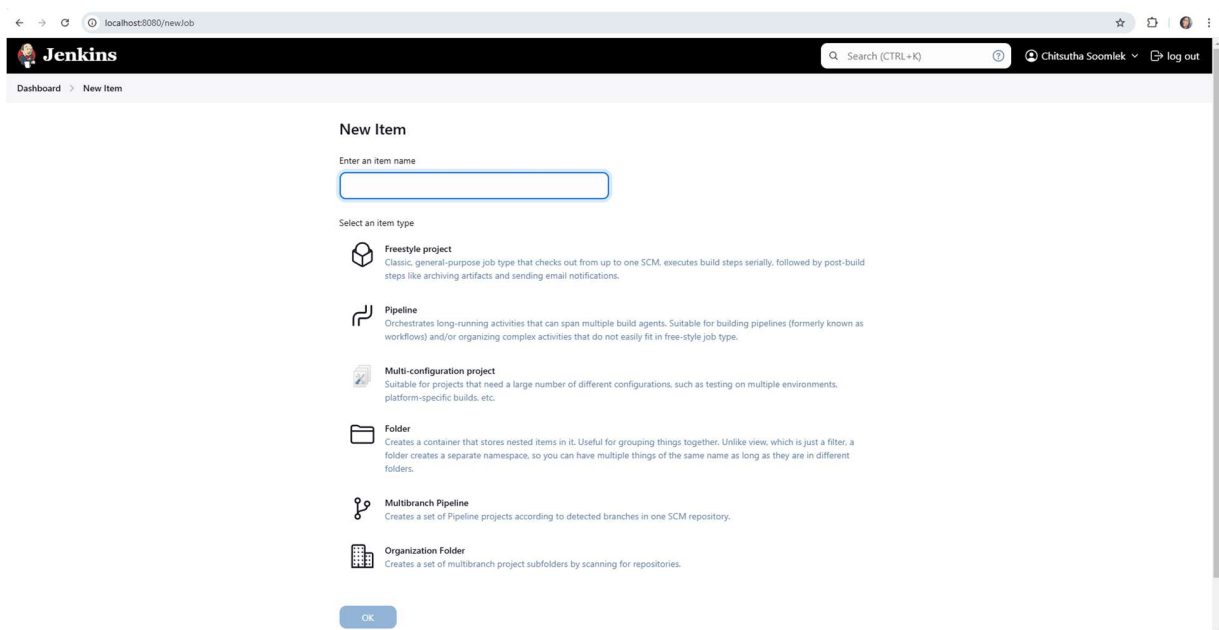


## Lab Worksheet

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)

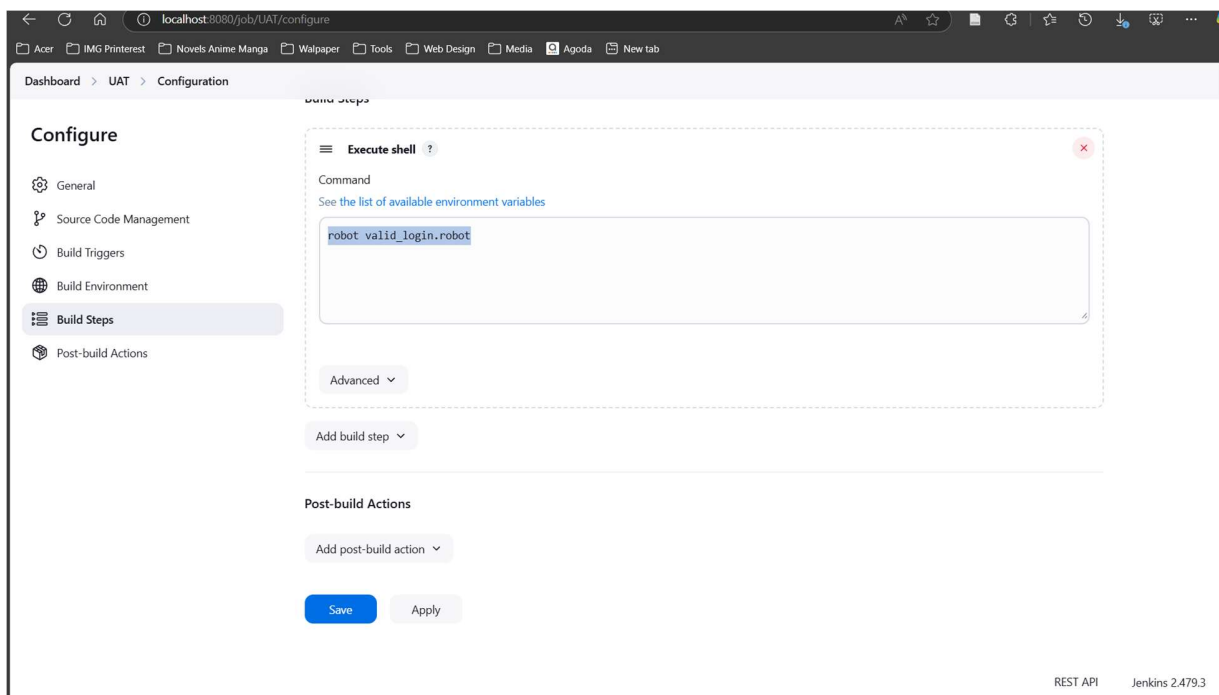
Build Trigger: เลือกแบบ Build periodically แล้วกำหนดให้ build ทุก 15 นาที



## Lab Worksheet

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

[Check point#14] Capture หน้าจอแสดงการตั้งค่า พร้อมกับตอบคำถามต่อไปนี้



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

**robot valid\_login.robot**

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

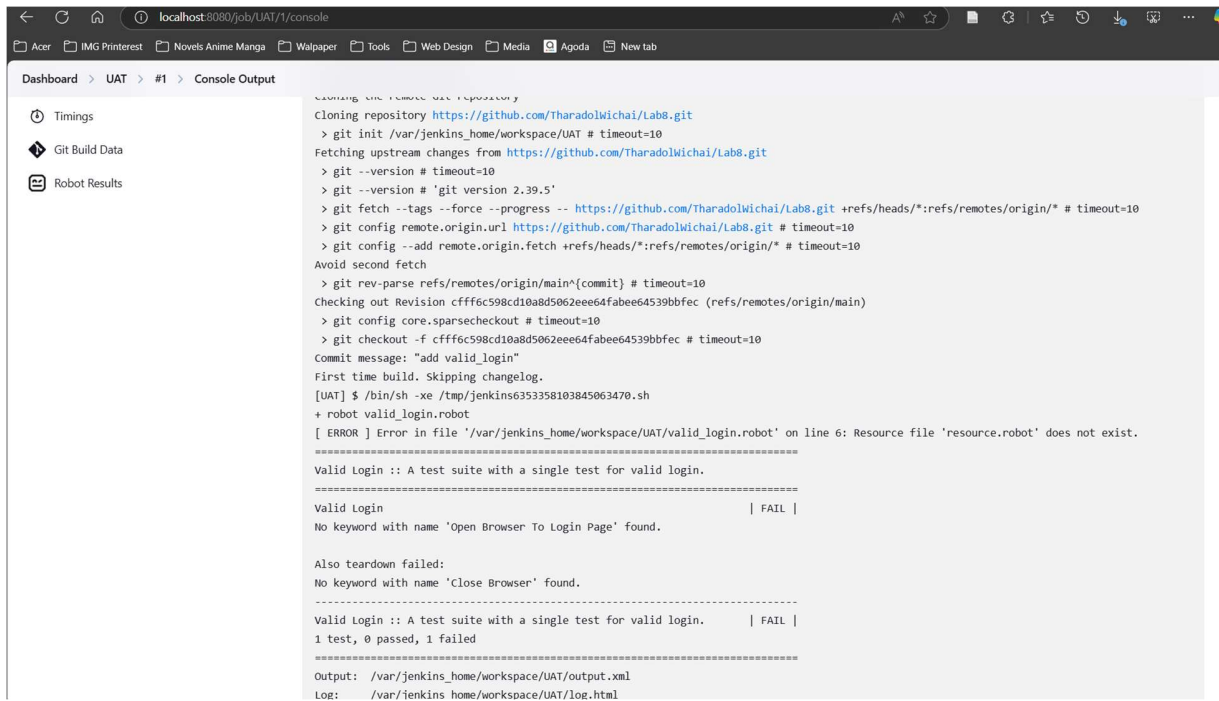
ระบุไดเรกทอรีที่เก็บไฟล์ผลการทดสอบโดย Robot framework ในรูป xml และ html -> ตั้งค่า Threshold เป็น % ของการทดสอบที่ไม่ผ่านแล้วนับว่าซอฟต์แวร์มีปัญหา -> ตั้งค่า Threshold เป็น % ของการทดสอบที่ผ่านแล้วนับว่าซอฟต์แวร์มีอยู่ในสถานะที่สามารถนำไปใช้งานได้ (เช่น 20, 80)

13. กด Apply และ Save

14. สั่ง Build Now

## Lab Worksheet

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



```
Cloning the remote Git repository
Cloning repository https://github.com/Tharadolwchai/Lab8.git
> git init /var/jenkins_home/workspace/UAT # timeout=10
Fetching upstream changes from https://github.com/Tharadolwchai/Lab8.git
> git --version # timeout=10
> git --version # 'git version 2.39.5'
> git fetch --tags --force --progress -- https://github.com/Tharadolwchai/Lab8.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git config remote.origin.url https://github.com/Tharadolwchai/Lab8.git # timeout=10
> git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision cfff6c598cd10a8d5062eee64fabee64539bbfec (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f cfff6c598cd10a8d5062eee64fabee64539bbfec # timeout=10
Commit message: "add valid_login"
First time build. Skipping changelog.
[UAT] $ /bin/sh -xe /tmp/jenkins6353358103845063470.sh
+ robot valid_login.robot
[ ERROR ] Error in file '/var/jenkins_home/workspace/UAT/valid_login.robot' on line 6: Resource file 'resource.robot' does not exist.

Valid Login :: A test suite with a single test for valid login.
=====
Valid Login                                     | FAIL |
No keyword with name 'Open Browser To Login Page' found.

Also teardown failed:
No keyword with name 'Close Browser' found.
-----
Valid Login :: A test suite with a single test for valid login.      | FAIL |
1 test, 0 passed, 1 failed
=====
Output: /var/jenkins_home/workspace/UAT/output.xml
Log: /var/jenkins_home/workspace/UAT/log.html
```