

## Lab Worksheet

ชื่อ-นามสกุล นายอัฐวัฒน์ คำมาศ รหัสนักศึกษา 653380352-1 Section 4

## Lab#8 – Software Deployment Using Docker

## วัตถุประสงค์การเรียนรู้

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

## Pre-requisite

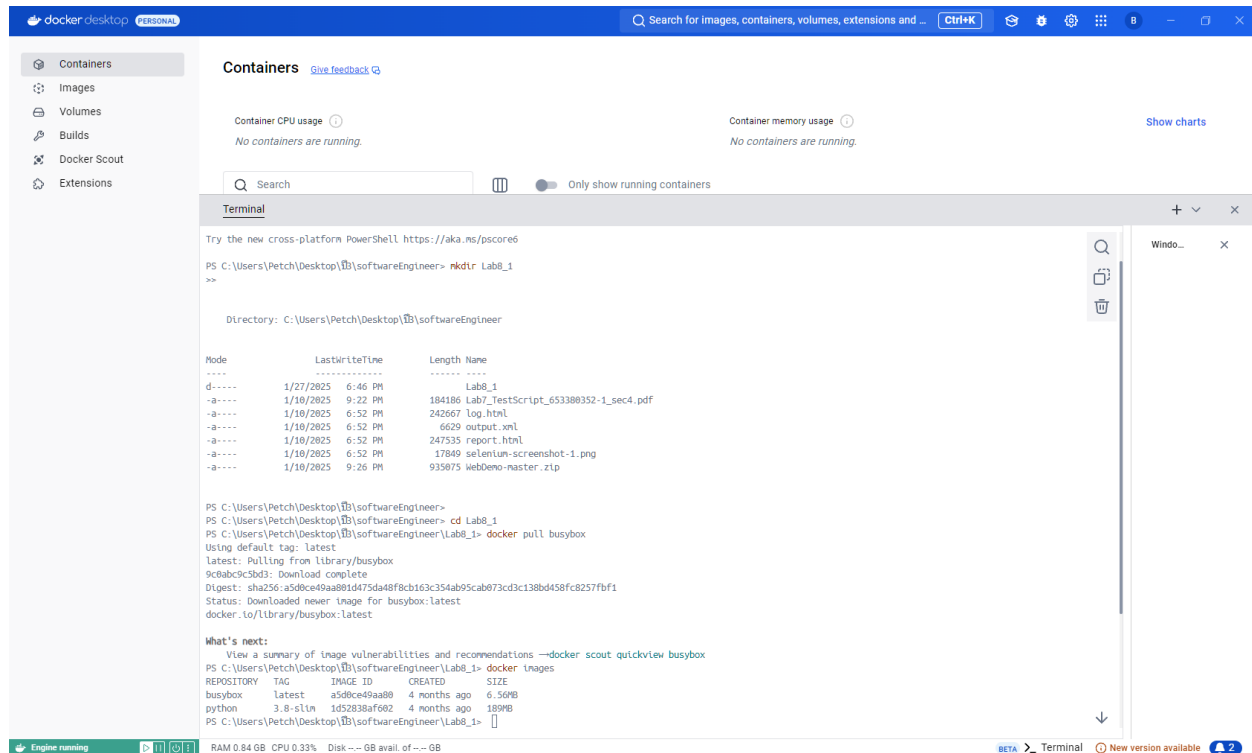
1. ติดตั้ง Docker desktop ลงบนเครื่องคอมพิวเตอร์ โดยดาวน์โหลดจาก <https://www.docker.com/get-started>
2. สร้าง Account บน Docker hub (<https://hub.docker.com/signup>)
3. กำหนดให้ \$ หมายถึง Command prompt และ <> หมายถึง ให้ป้อนค่าของพารามิเตอร์ที่กำหนด

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

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

## Lab Worksheet

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



(1) สิ่งที่อยู่ภายใต้คอลัมน์ Repository คืออะไร busybox กับ python

(2) Tag ที่ใช้บ่งบอกถึงอะไร บ่งบอกถึง 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

## Lab Worksheet

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

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

```

/ # ls
bin dev etc home lib lib64 proc root sys tmp usr var
/ #
/ # ex
sh: ex: not found
/ # exit
PS C:\Users\Petch\Desktop\lab\SoftwareEngineer\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 27 12:03 .
drwxr-xr-x 1 root root      4096 Jan 27 12:03 ..
-rwxr-xr-x 1 root root         0 Jan 27 12:03 .dockerenv
drwxr-xr-x 2 root root     12288 Sep 26 21:31 bin
drwxr-xr-x 5 root root      3680 Jan 27 12:03 dev
drwxr-xr-x 1 root root      4096 Jan 27 12:03 etc
drwxr-xr-x 2 nobody nobody    4096 Sep 26 21:31 home
drwxr-xr-x 2 root root      4096 Sep 26 21:31 lib
lrwxrwxr-x 1 root root         3 Sep 26 21:31 lib64 -> lib
dr-xr-xr-x 196 root root         0 Jan 27 12:03 proc
drwx----- 1 root root      4096 Jan 27 12:03 root
dr-xr-xr-x 11 root root         0 Jan 27 12:03 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\Petch\Desktop\lab\SoftwareEngineer\Lab8_1> docker run busybox echo "Hello Attawat Kamas from busybox"
Hello Attawat Kamas from busybox
PS C:\Users\Petch\Desktop\lab\SoftwareEngineer\Lab8_1> docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS   NAMES
a7195897bcfb   busybox   "echo 'Hello Attawat..." 8 seconds ago  Exited (0) 7 seconds ago           vigilant_dubinsky
bb7559a987f1   busybox   "sh"                    2 minutes ago  Exited (0) About a minute ago       lucid_poincare
4287f0513a29   busybox   "sh"                    4 minutes ago  Exited (127) 2 minutes ago         sharp_aryabhata
7883a47a31c6   busybox   "sh"                    8 minutes ago  Exited (0) 8 minutes ago         friendly_booth
23c594b4cddb   python:3.8-slim "python3"               3 months ago  Exited (137) 20 minutes ago        my_python_container
  
```

The bottom status bar shows 'Engine running', 'RAM 1.02 GB', 'CPU --%', and 'Disk -- GB avail. of -- GB'. There is also a 'New version available' notification.

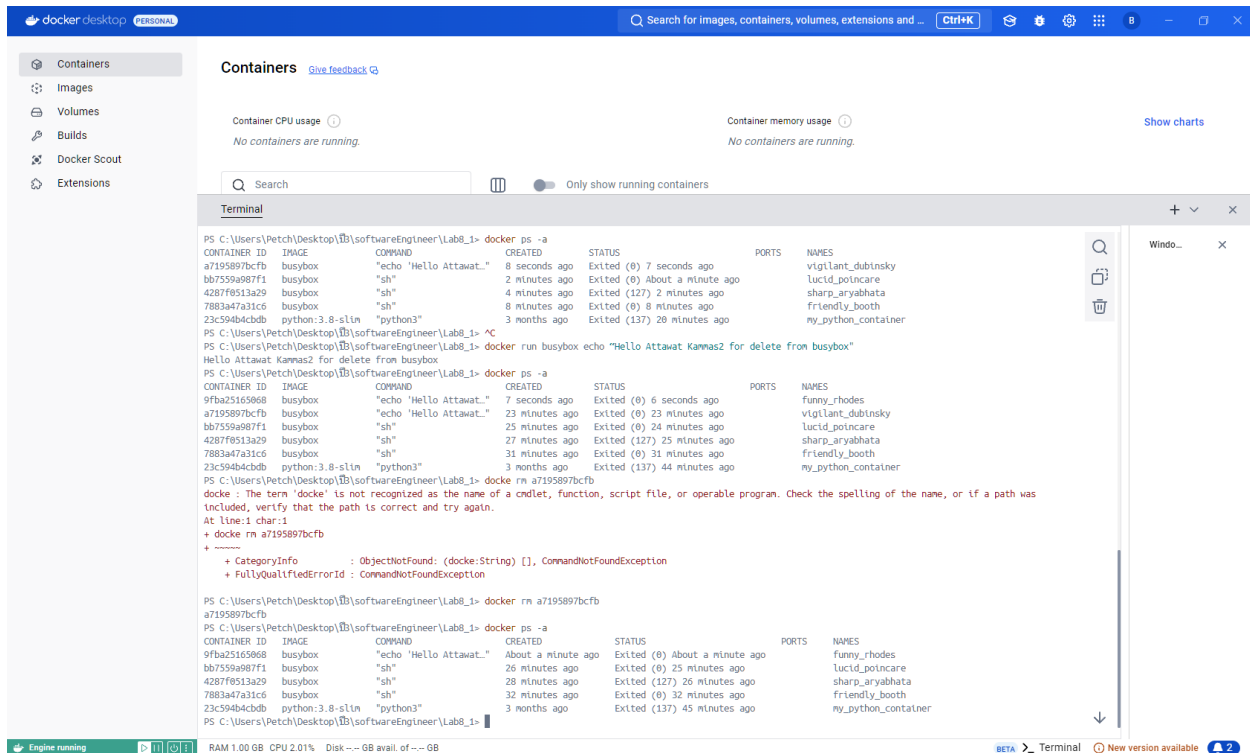
(1) เมื่อใช้ option -it ในคำสั่ง run ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป  
Option -it ใช้เพื่อเปิดโหมดโต้ตอบ (Interactive Shell) และจำลอง Terminal สำหรับ Container ทำให้คุณสามารถสื่อสารกับ Container ได้โดยตรงผ่าน Command Line และใช้งาน Shell หรือโปรแกรมที่ต้องการการโต้ตอบจากผู้ใช้

(2) คอลัมน์ STATUS จากการรันคำสั่ง docker ps -a แสดงถึงข้อมูลอะไร  
คอลัมน์ STATUS จากผลลัพธ์ของคำสั่ง docker ps -a แสดง สถานะของแต่ละ Container ที่ถูกสร้างขึ้นในระบบ ซึ่งระบุถึงว่า Container กำลังทำงานอยู่หรือหยุดทำงานแล้ว และแสดงระยะเวลาที่เกิดสถานะนั้น

## 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 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 interface with the 'Containers' tab selected. The 'Terminal' window displays the following output:

```
=> [Internal] load build definition from Dockerfile
=> => transferring dockerfile: 196B
[Internal] load metadata for docker.io/library/busybox:latest
=> [Internal] load .dockerignore
=> transferring context: 2B
[1/1] FROM docker.io/library/busybox:latest@sha256:a5d9ce49aa881d475da48f8cb163c354ab95cab873cd3c138bd458fc8257fbf1
=> resolve docker.io/library/busybox:latest@sha256:a5d9ce49aa881d475da48f8cb163c354ab95cab873cd3c138bd458fc8257fbf1
[Auth] library/busybox:pull token for registry-1.docker.io
=> exporting to image
=> exporting layers
=> exporting manifest sha256:4c62dfdb815eb3df41bf43475b3583ca3c9b54e88b1cc3aabe51b56ee664b23
=> exporting config sha256:722bb6873336aa674bed3a8ab7bace9838f24c48fae1e843813ce7786283315
=> exporting attestation manifest sha256:ecc3ba93238461c1cb37ef2ce83242a1db1293e138a68c3814662f2ef8455626
=> exporting manifest list sha256:76b602884abfc32d58583ee2eff90e45e78999f2d595e199cd7a68cdcdd72148
=> naming to docker.io/library/my_first_image:latest
=> unpacking to docker.io/library/my_first_image:latest

3 warnings found (use docker --debug to expand):
- JSNWargsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 3)
- JSNWargsRecommended: 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)

What's next:
View a summary of image vulnerabilities and recommendations --docker scout quickview

C:\Users\Petch\Desktop\lab\softwareEngineer\Lab8_2>

C:\Users\Petch\Desktop\lab\softwareEngineer\Lab8_2> docker images
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
my_first_image      latest          76b602884abf   4 months ago   6.50MB
busybox              latest         a5d9ce49aa88   4 months ago   6.50MB
python              3.8-slim      1d52838af602   4 months ago   189MB

C:\Users\Petch\Desktop\lab\softwareEngineer\Lab8_2> docker run my_first_image
Hi there. This is my first docker image.
ชื่อ-นามสกุล รหัสนักศึกษา ชื่อเล่น
```

## Lab Worksheet

(1) คำสั่งที่ใช้ในการ run คือ

`docker run my_first_image`

(2) Option -t ในคำสั่ง `$ docker build` ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป  
ใช้สำหรับการกำหนดชื่อและแท็กให้กับ Docker image ที่สร้างขึ้น ซึ่งทำให้การใช้งานและจัดการ Docker image เป็นระเบียบและสะดวกยิ่งขึ้น

### 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

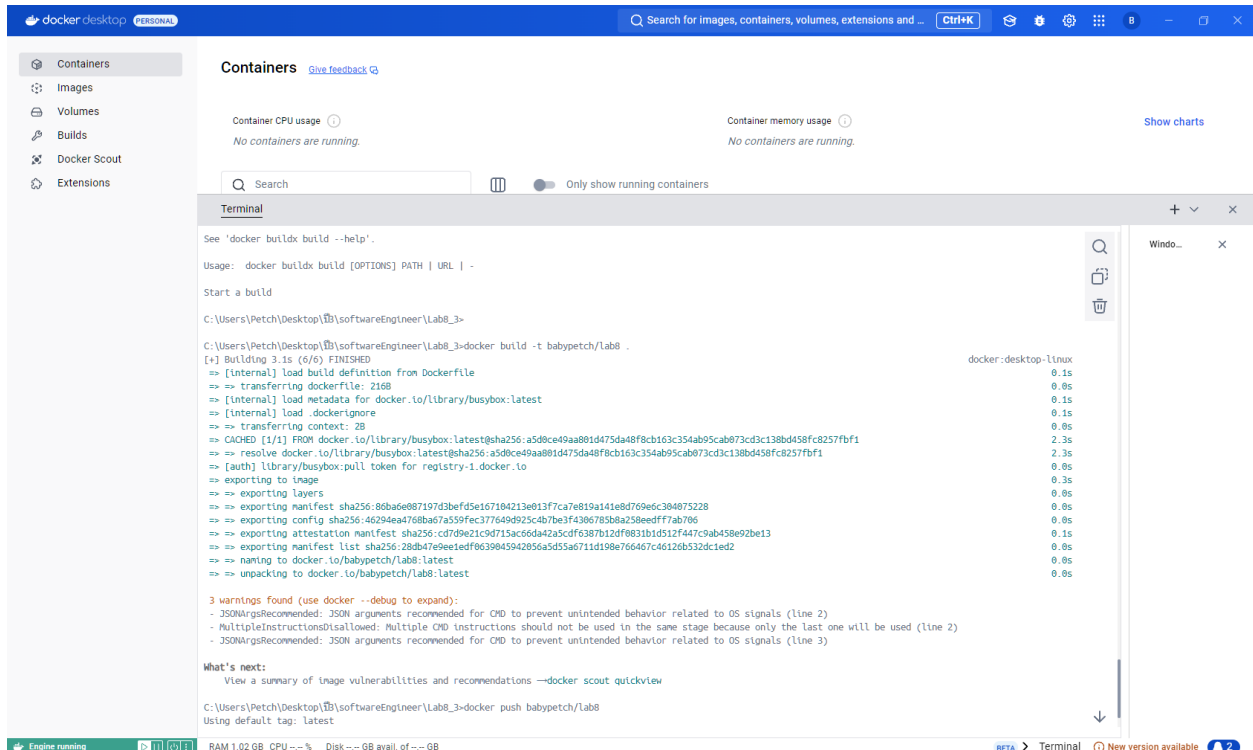
## Lab Worksheet

\$ 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



6. ทำการ Push ตัว Docker image ไปไว้บน Docker Hub โดยใช้คำสั่ง

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

ในกรณีที่ติดปัญหาไม่ได้ Login ไว้ก่อน ให้ใช้คำสั่งต่อไปนี้ เพื่อ Login ก่อนทำการ Push

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

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

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

## Lab Worksheet

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

The screenshot shows the Docker Desktop interface with the Terminal window open. The terminal output displays the commands and progress for building and pushing a Docker image to Docker Hub.

```

C:\Users\Petch\Desktop\lab8> docker build -t babypetch/lab8 .
[+] Building 3.1s (6/6) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 216B
=> [internal] load metadata for docker.io/library/busybox:latest
=> [internal] load dockerignore
=> => transferring context: 2B
=> CACHED [1/1] FROM docker.io/library/busybox:latest@sha256:a5d8ce49aa801d475da48f8cb163c354ab95cab873cd3c138bd458fc8257fbf1
=> => resolve docker.io/library/busybox:latest@sha256:a5d8ce49aa801d475da48f8cb163c354ab95cab873cd3c138bd458fc8257fbf1
=> [auth] library/busybox:pull token for registry-1.docker.io
=> exporting to image
=> => exporting layers
=> => exporting manifest sha256:86ba6e087197d3befd5e167194213e813f7ca7e819a141e8d769e6c304073228
=> => exporting config sha256:46294ea4768ba67a559fec377649d925c4b7be3f4306785b8a25eedff7ab706
=> => exporting attestation manifest sha256:cd7d9e21c9d715acc66da42a5cdf6387b12df6831bd1d512f447c9ab458e92be13
=> => exporting manifest list sha256:28db47e9ee1edf6639845942856a5d55a6711d198e766467c46126b532dc1ed2
=> => naming to docker.io/babypetch/lab8:latest
=> => unpacking to docker.io/babypetch/lab8: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

C:\Users\Petch\Desktop\lab8> docker push babypetch/lab8
Using default tag: latest
The push refers to repository [docker.io/babypetch/lab8]
9c8abc9c5bd3: Mounted from library/busybox
d912a91b44e3: Pushed
latest: digest: sha256:28db47e9ee1edf6639845942856a5d55a6711d198e766467c46126b532dc1ed2 size: 855

C:\Users\Petch\Desktop\lab8>
  
```

The Docker Hub interface shows the repository 'babypetch/lab8' with a table of repository details:

Name	Last Pushed	Contains	Visibility	Scout
babypetch/lab8	2 minutes ago	IMAGE	Public	Inactive

The interface also includes a sidebar with options to 'Create a repository', 'Create an organization', and 'Create and manage users and grant access to your repositories'.

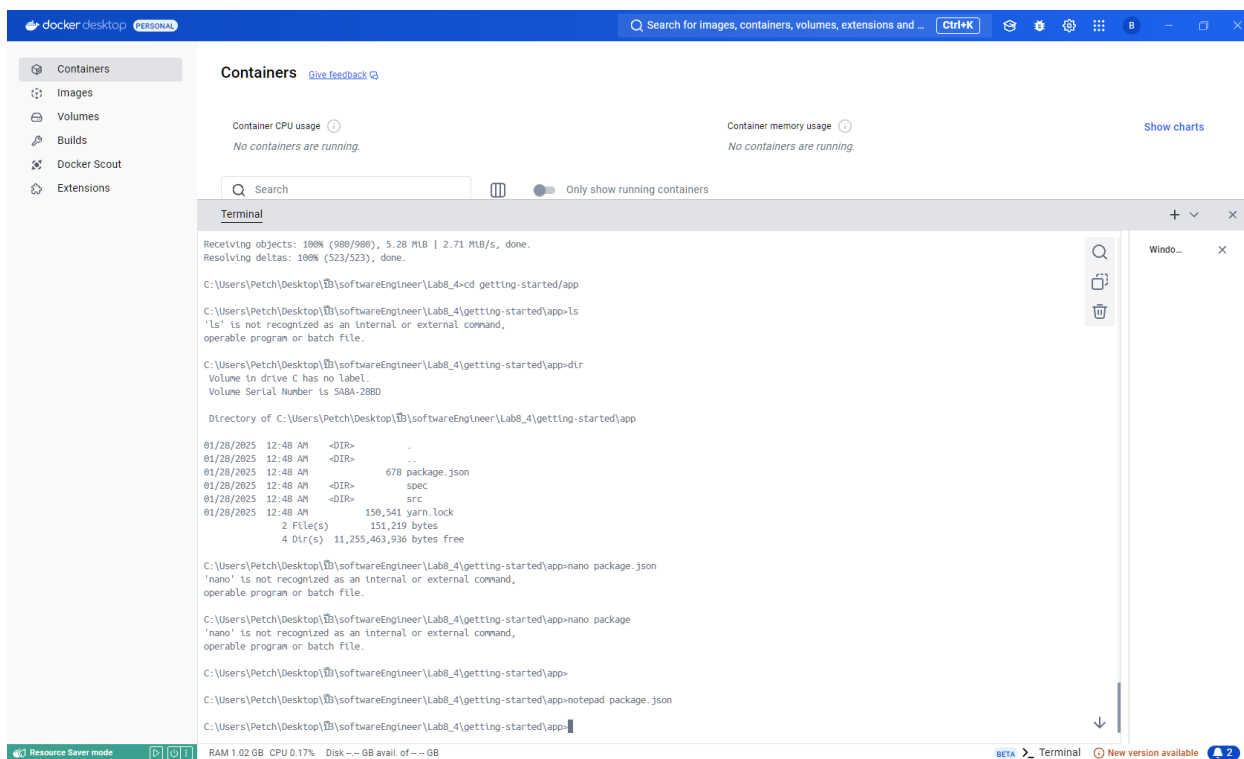


## Lab Worksheet

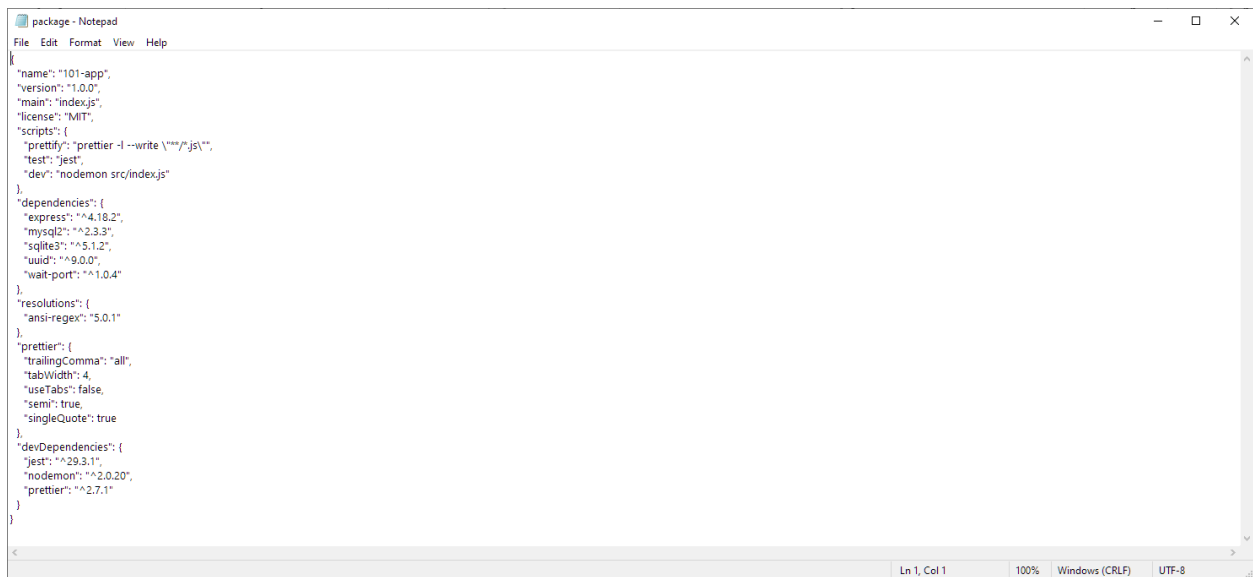
## แบบฝึกปฏิบัติที่ 8.4: การ Build แอปพลิเคชันจาก Container image และการ Update แอปพลิเคชัน

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



```

package - Notepad
File Edit Format View Help
{
  "name": "101-app",
  "version": "1.0.0",
  "main": "index.js",
  "license": "MIT",
  "scripts": {
    "prettify": "prettier -i --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"
  }
}
Ln 1, Col 1 100% Windows (CRLF) UTF-8

```

4. ภายใต้ getting-started/app ให้สร้าง Dockerfile พร้อมกับใส่เนื้อหาดังต่อไปนี้ลงไปไฟล์

FROM node:18-alpine

WORKDIR /app

COPY . .

RUN yarn install --production

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

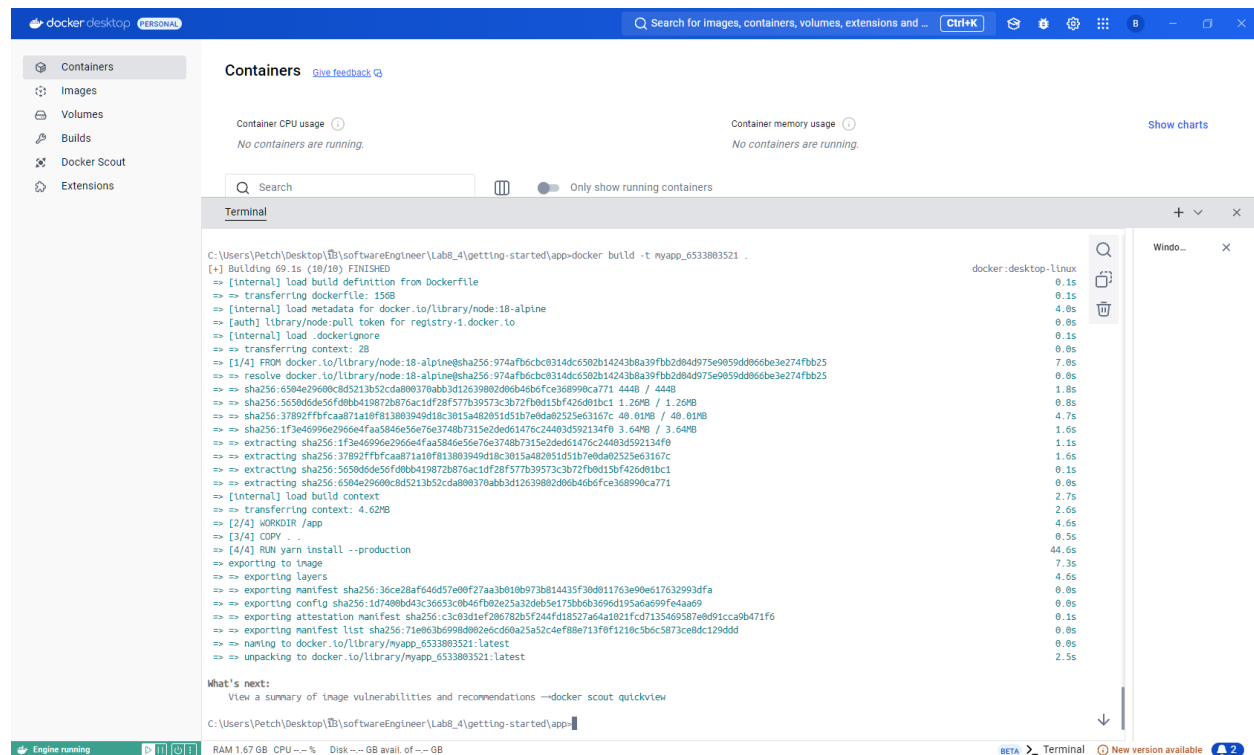
EXPOSE 3000

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



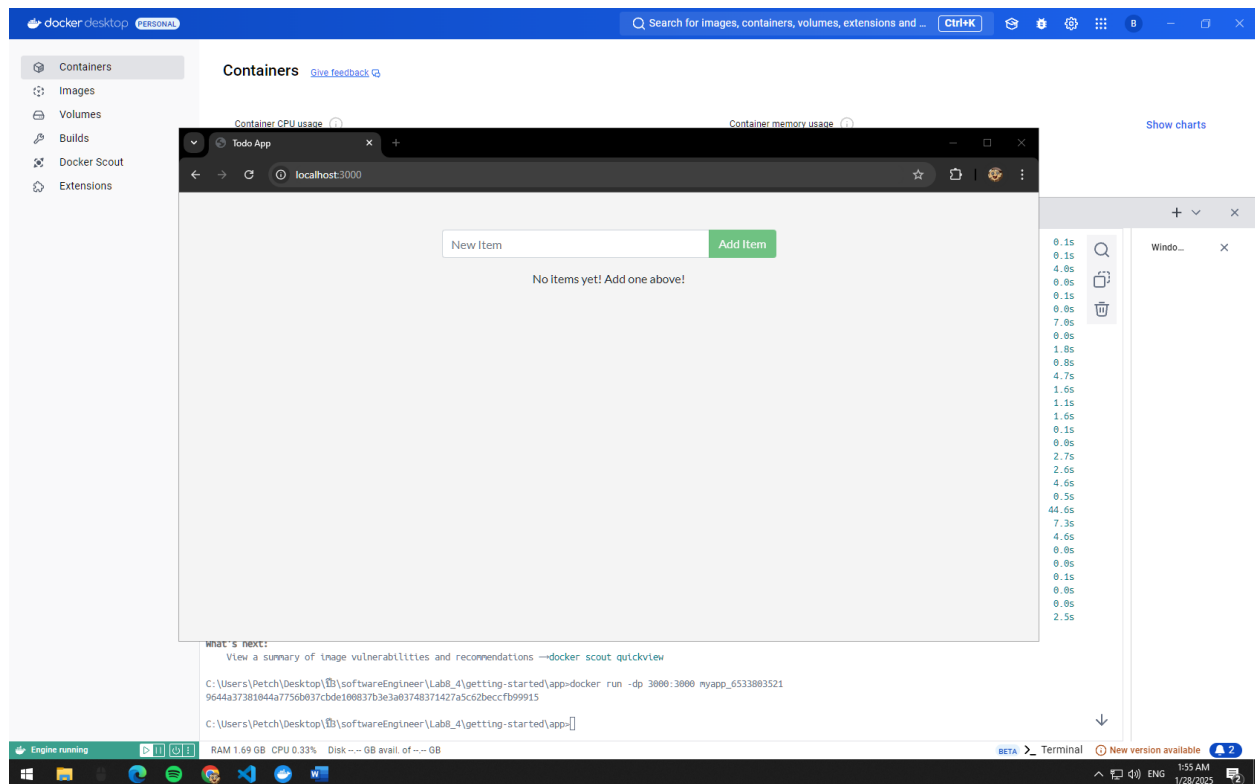
6. ทำการ Start ตัว Container ของแอปพลิเคชันที่สร้างขึ้น โดยใช้คำสั่ง

\$ 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

## 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>
```

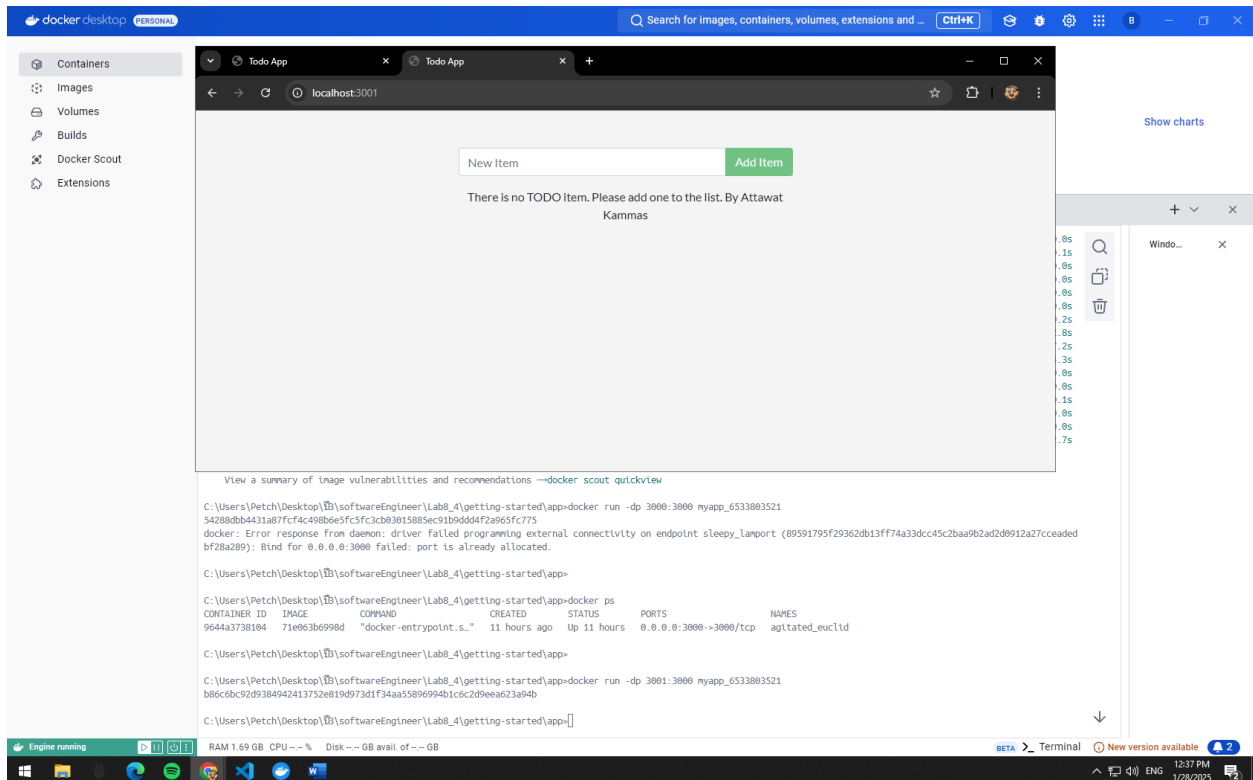
 เป็น
 

```
<p className="text-center">There is no TODO item. Please add one to the list.
```

 By ชื่อและนามสกุลของนักศึกษา
  - 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



(1) Error ที่เกิดขึ้นหมายความว่าอย่างไร และเกิดขึ้นเพราะอะไร

docker: Error response from daemon: driver failed programming external connectivity on endpoint sleepy\_lampport

(89591795f29362db13ff74a33dcc45c2baa9b2ad2d0912a27cceedad bf28a289): Bind for 0.0.0.0:3000 failed: port is already allocated.

หมายความว่าพอร์ต 3000 ถูกใช้งานโดยแอปพลิเคชันหรือ Container อื่นแล้ว

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

## Lab Worksheet

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

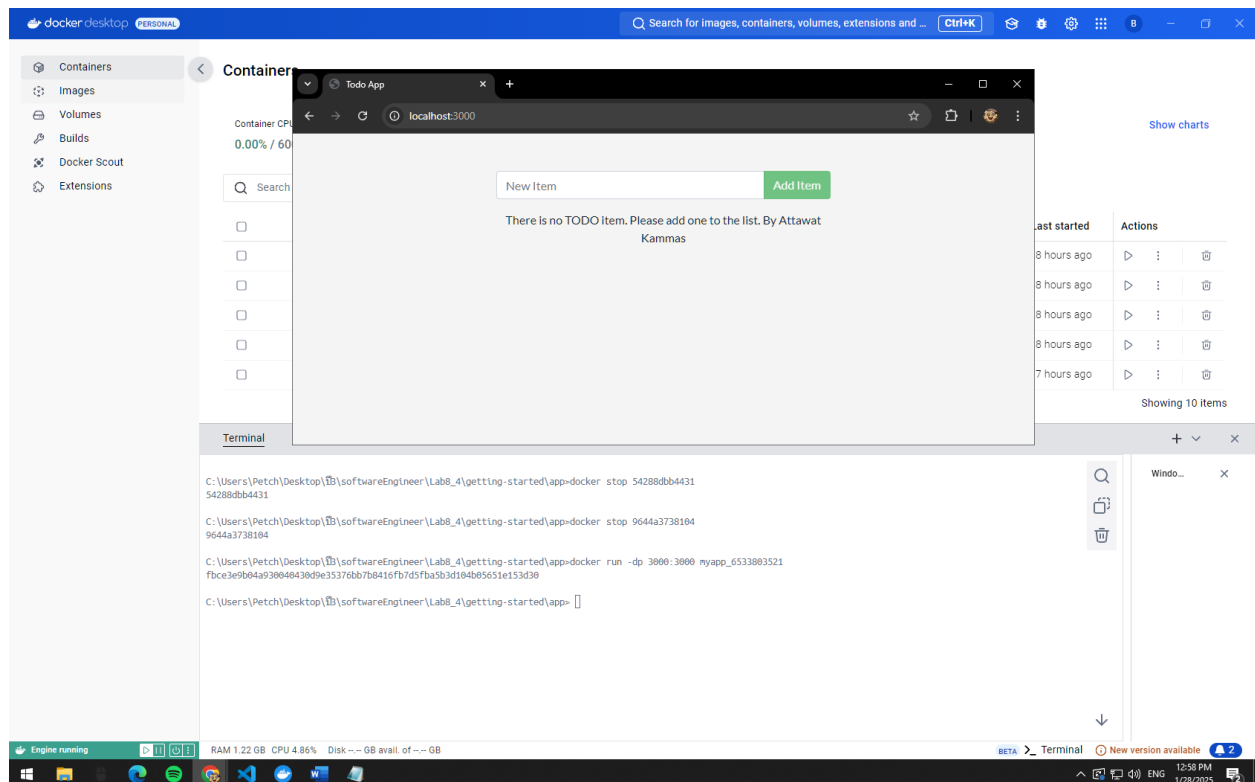
ii. เลือกไอคอนถังขยะในแถวของ Container ที่ต้องการจะลบ

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



## 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. บันทึกที่รหัสผ่านของ Admin user ไว้สำหรับ log-in ในครั้งแรก

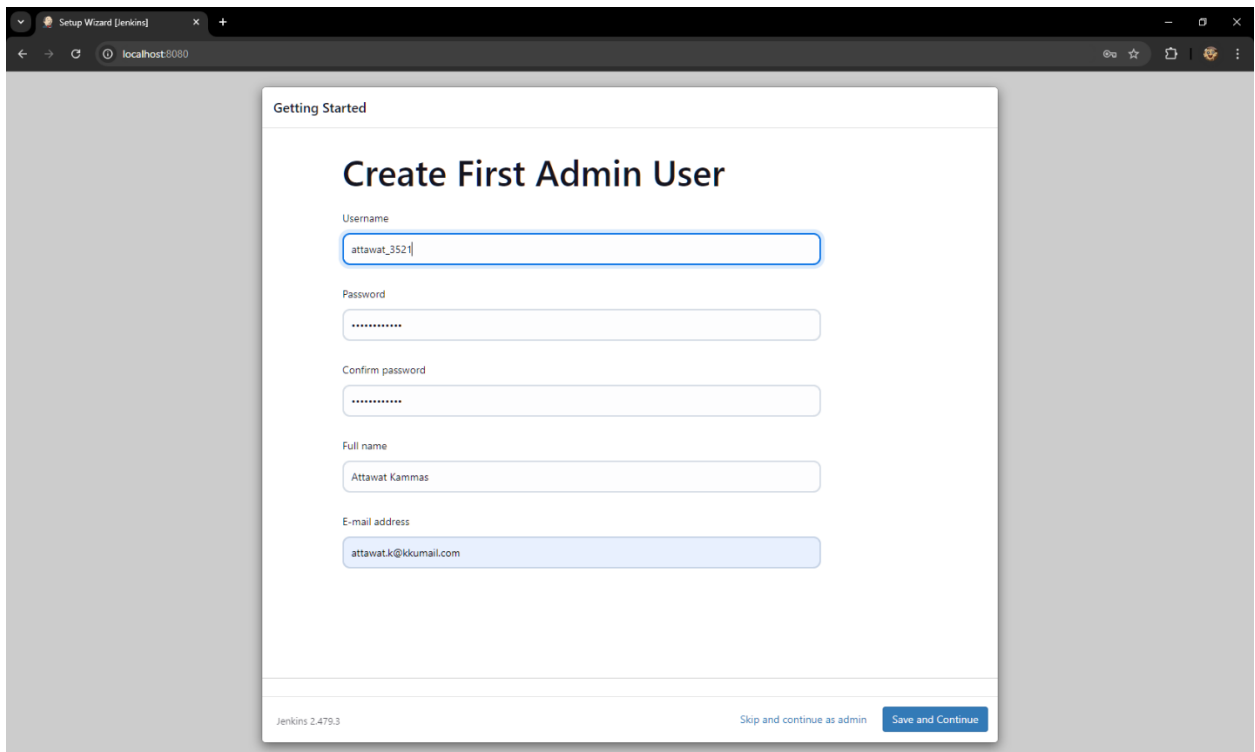
[Check point#12] Capture หน้าจอที่แสดงผล Admin password

The screenshot displays the Docker Desktop interface. The top section shows the 'Containers' tab with a table of running containers. Below this, the 'Terminal' tab shows the logs of the Jenkins container. The logs indicate that Jenkins is starting up, including the initialization of the Jenkins home directory and the loading of plugins. A message in the logs states: 'This may also be found at: /var/jenkins\_home/secrets/initialAdminPassword'. Below the logs, the Jenkins web interface is shown, displaying the 'Unlock Jenkins' screen. The screen prompts the user to enter the administrator password, which was previously captured from the logs.

## Lab Worksheet

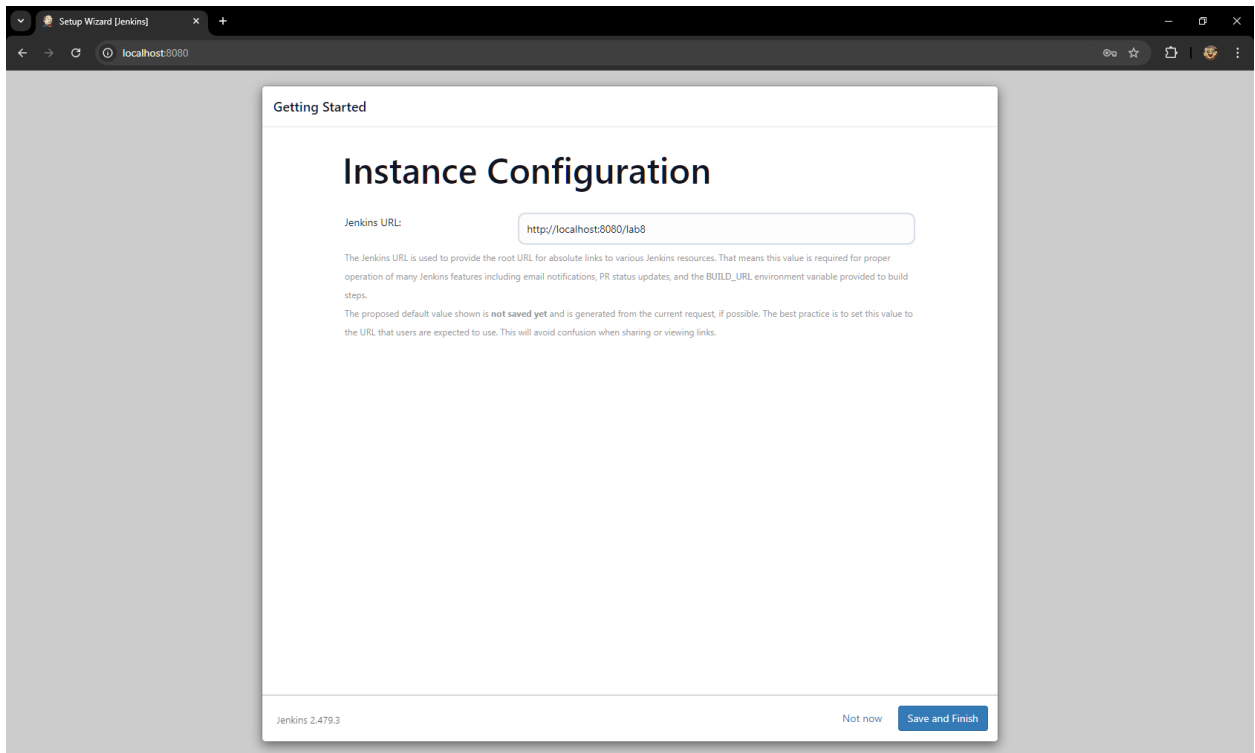
- เมื่อได้รับการยืนยันว่า Jenkins is fully up and running ให้เปิดบราวเซอร์ และป้อนที่อยู่เป็น localhost:8080
- ทำการ Unlock Jenkins ด้วยรหัสผ่านที่ได้ในข้อที่ 3
- สร้าง Admin User โดยใช้ username เป็นชื่อจริงของนักศึกษาพร้อมรหัสสี่ตัวท้าย เช่น somsri\_3062

[Check point#13] Capture หน้าจอที่แสดงผลการตั้งค่า

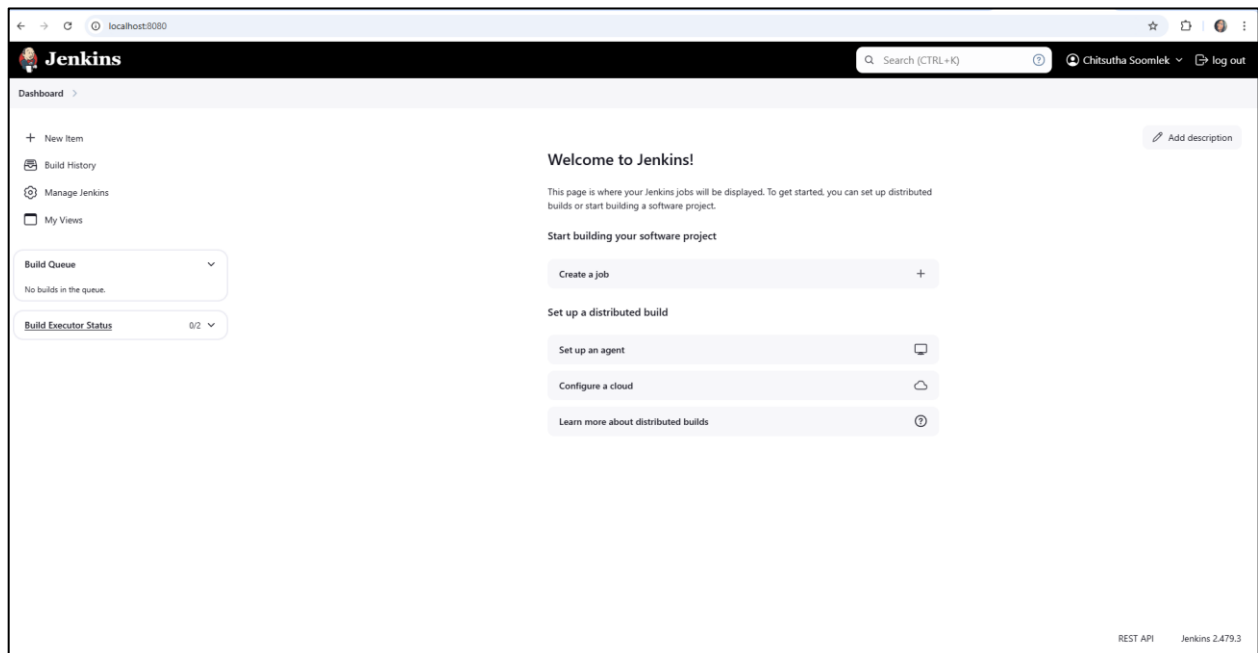




## Lab Worksheet

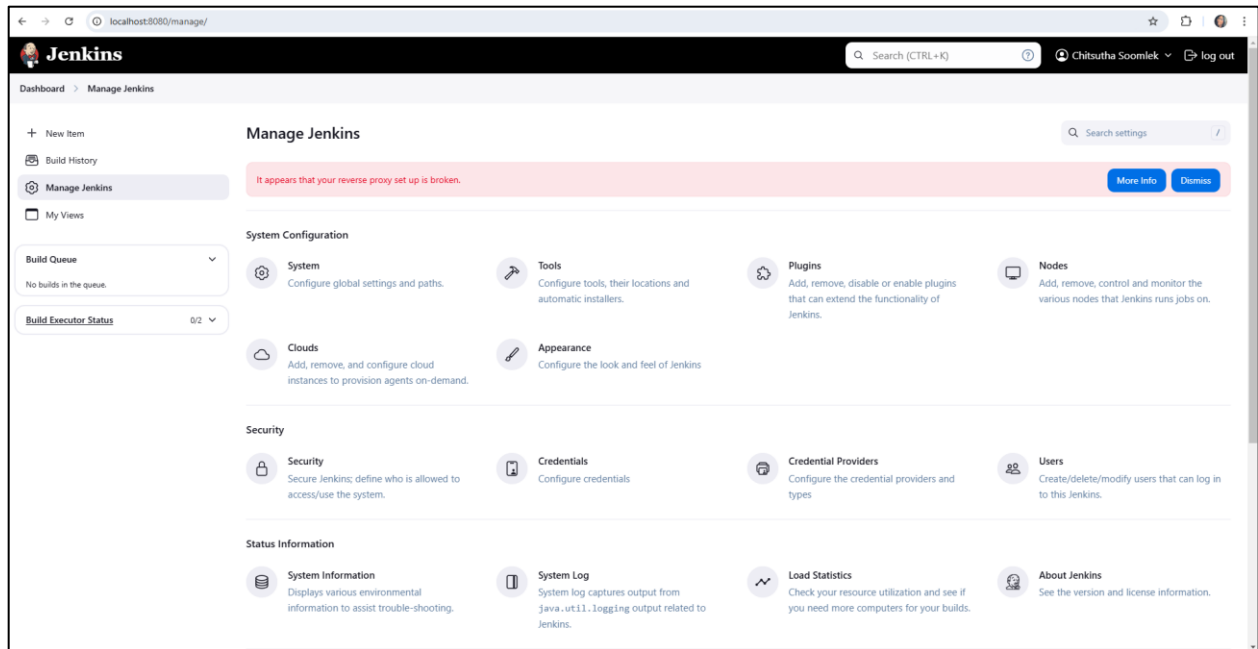


7. กำหนด Jenkins URL เป็น <http://localhost:8080/lab8>
8. เมื่อติดตั้งเรียบร้อยแล้วจะพบกันหน้า Dashboard ดังแสดงในภาพ

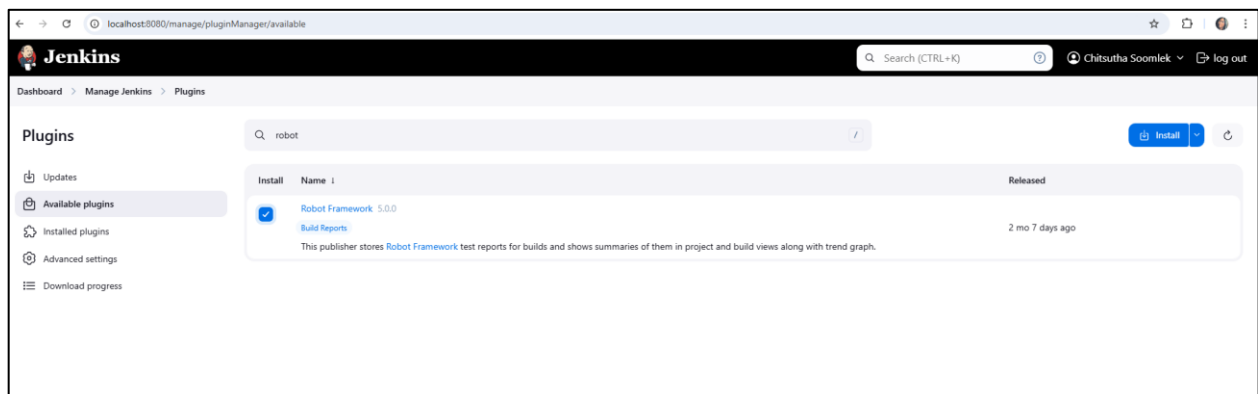


## Lab Worksheet

## 9. เลือก Manage Jenkins แล้วไปที่เมนู Plugins

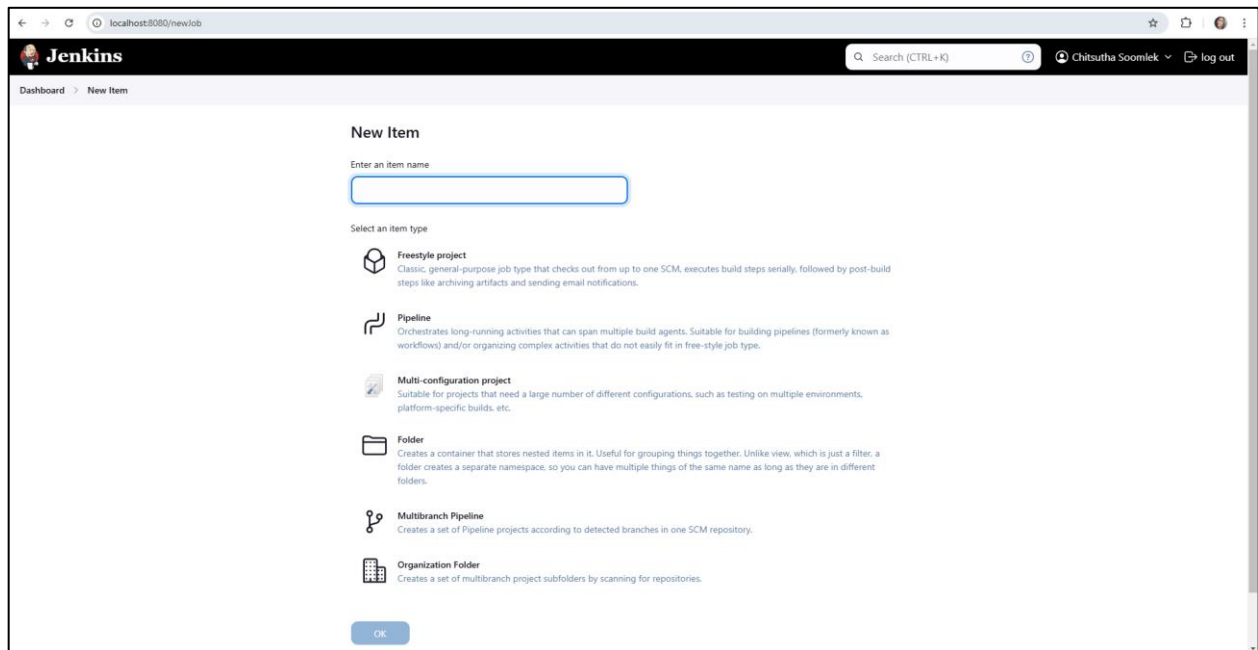


## 10. ไปที่เมนู Available plugins แล้วเลือกติดตั้ง Robotframework เพิ่มเติม



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

## Lab Worksheet



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

**Description:** Lab 8.5

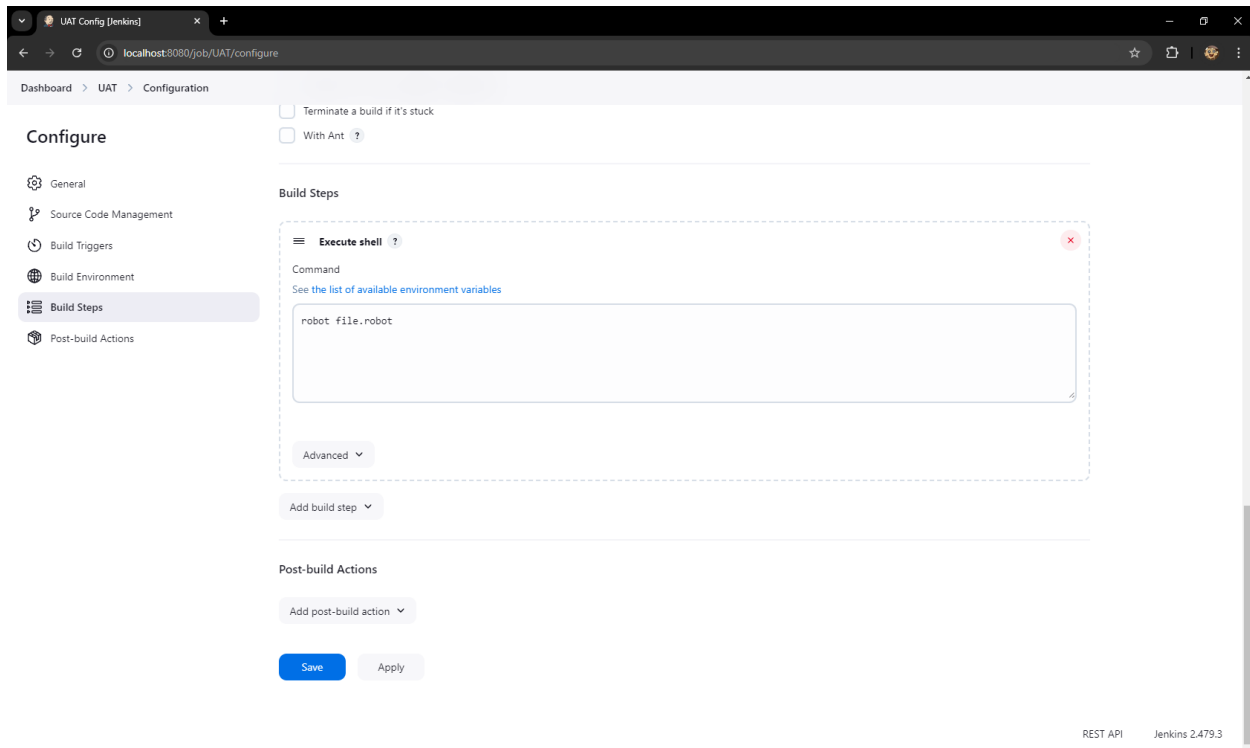
**GitHub project:** กดเลือก แล้วใส่ Project URL เป็น repository ที่เก็บโค้ด .robot (ดูขั้นตอนที่ 12)

**Build Trigger:** เลือกแบบ Build periodically แล้วกำหนดให้ build ทุก 15 นาที

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

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

## Lab Worksheet



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

`robot file.robot`

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

The screenshot shows the Jenkins Dashboard interface. The top navigation bar includes the Jenkins logo, a search bar, and user information (Attawat Kammass). The left sidebar contains links for 'New Item', 'Build History', 'Manage Jenkins', and 'My Views'. The main content area displays a table of pipeline builds. The table has columns for 'S' (Status), 'W' (Webhook), 'Name', 'Last Success', 'Last Failure', 'Last Duration', and 'Robot Results + Duration Trend'. A single build is listed with the name 'UAT', a last failure of '7 min 2 sec', and a last duration of '8 ms'. The status 'S' is marked with a red 'X' icon, and the 'W' column shows a cloud icon. Below the table, there are filters for 'Icon: S M L' and a 'Build Queue' section indicating 'No builds in the queue'. The 'Build Executor Status' section shows '0/2'.

S	W	Name	Last Success	Last Failure	Last Duration	Robot Results + Duration Trend
		UAT	N/A	7 min 2 sec	8 ms	

Icon: S M L

Build Queue: No builds in the queue.

Build Executor Status: 0/2

REST API Jenkins 2.479.3