

## Lab Worksheet

ชื่อ-นามสกุล \_\_\_\_\_ นาย วัชรินทร์ จุลเพชร \_\_\_\_\_ รหัสนักศึกษา \_\_\_\_\_ 653380213-5 \_\_\_\_\_ 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

## Lab Worksheet

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

```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS D:\Software En\Lab8_1> docker pull busybox
Using default tag: latest
latest: Pulling from library/busybox
9c0abc9c5bd3: Download complete
Digest: sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138bd458fc8257fbf1
Status: Downloaded newer image for busybox:latest
docker.io/library/busybox:latest

What's next:
View a summary of image vulnerabilities and recommendations → docker scout quickview busybox
PS D:\Software En\Lab8_1> docker images
REPOSITORY    TAG       IMAGE ID      CREATED       SIZE
busybox       latest    a5d0ce49aa80  3 months ago  6.56MB
PS D:\Software En\Lab8_1>

```

- (1) สิ่งที่อยู่ภายใต้คอลัมน์ Repository คืออะไร ชื่อ image ในที่นี้จะแสดง busybox
- (2) Tag ที่ใช้บ่งบอกถึงอะไร version ของ image
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

```

Windows PowerShell
busybox latest a5d0ce49aa80 3 months ago 6.56MB
PS D:\Software En\Lab8_1> docker run busybox
PS D:\Software En\Lab8_1> docker run -it busybox sh
/ # ls
bin    etc    lib    proc   sys    usr
dev    home  lib64  root   tmp    var
/ # ls -la
total 48
drwxr-xr-x 1 root root      4096 Jan 23 02:50 .
drwxr-xr-x 1 root root      4096 Jan 23 02:50 ..
-rwxr-xr-x 1 root root         0 Jan 23 02:50 .dockerenv
drwxr-xr-x 2 root root     12288 Sep 26 21:31 bin
drwxr-xr-x 5 root root       360 Jan 23 02:50 dev
drwxr-xr-x 1 root root      4096 Jan 23 02:50 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
lrwxrwxrwx 1 root root         3 Sep 26 21:31 lib64 -> lib
dr-xr-xr-x 239 root root         0 Jan 23 02:50 proc
drwx----- 1 root root      4096 Jan 23 02:50 root
dr-xr-xr-x 11 root root         0 Jan 23 02:50 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 D:\Software En\Lab8_1> docker run busybox echo "Hello Wangphai Jullapech form busy box"
Hello Wangphai Jullapech form busy box
PS D:\Software En\Lab8_1> docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS          PORTS          NAMES
388ba4c6b31c   busybox       "echo 'Hello Wangpha..." 10 seconds ago Exited (0) 9 seconds ago           upbeat_mestorf
angry_roentgen
8efb7b76cd01   busybox       "sh"                    2 minutes ago Exited (0) About a minute ago           upbeat_mestorf
e25270f8c6b5   busybox       "sh"                    2 minutes ago Exited (0) 2 minutes ago           eloquent_turing
PS D:\Software En\Lab8_1>

```

(1) เมื่อใช้ option -it ในคำสั่ง run ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป

เมื่อใช้ option -it จะเป็นการเข้าไปใน image เข้าไปใช้งาน service ต่างๆ ใน image หาก run แบบที่ไม่ใช้ option -it จะเป็นการรันแบบไม่ตอบโต้กับผู้ใช้

(2) คอลัมน์ STATUS จากการรันคำสั่ง docker ps -a แสดงถึงข้อมูลอะไร

STATUS จะแสดงสถานะการทำงานใน container ว่าทำงานอยู่หรือหยุดทำงานแล้ว

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

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

```

PS D:\Software En\Lab8_1> docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS          PORTS          NAMES
16fd371aebff   busybox       "sh"                    3 minutes ago Exited (0) 3 minutes ago           goofy_beaver
3d5bb303936e   busybox       "sh"                    3 minutes ago Exited (0) 3 minutes ago           busy_kare
388ba4c6b31c   busybox       "echo 'Hello Wangpha..." 11 minutes ago Exited (0) 11 minutes ago           angry_roentgen
8efb7b76cd01   busybox       "sh"                    13 minutes ago Exited (0) 12 minutes ago           upbeat_mestorf
e25270f8c6b5   busybox       "sh"                    14 minutes ago Exited (0) 14 minutes ago           eloquent_turing
PS D:\Software En\Lab8_1>

```

## 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

## Lab Worksheet

- ย้ายตำแหน่งปัจจุบันไปที่ Lab8\_2 เพื่อใช้เป็น Working directory
- สร้าง Dockerfile.swp ไว้ใน Working directory

สำหรับเครื่องที่ใช้ระบบปฏิบัติการวินโดวส์ (Windows) บันทึกคำสั่งต่อไปนี้ลงในไฟล์ โดยใช้ Text Editor ที่มี

```
FROM busybox
```

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

```
CMD echo "ชื่อ-นามสกุล รหัสนักศึกษา ชื่อเล่น"
```

สำหรับเครื่องที่ใช้ระบบปฏิบัติการ MacOS หรือ Linux บนหน้าต่าง Terminal 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

- ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้

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

- เมื่อ Build สำเร็จแล้ว ให้ทำการรัน Docker image ที่สร้างขึ้นในขั้นตอนที่ 5

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

```
PS D:\Software En\Lab8_2> docker build -t first-docker .
[+] Building 0.2s (5/5) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 153B
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 2)
=> WARN: MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the same stage because only the last one will be used (line 2)
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 3)
=> [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:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138bd458fc8257fbf1
=> => resolve docker.io/library/busybox:latest@sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138bd458fc8257fbf1
=> exporting to image
=> => exporting layers
=> => exporting manifest sha256:781b5581f07830b5bde51d084834f26320ff803f3f86582a1a88cb903552d806
=> => exporting config sha256:14e6cd657bec4a3ae3bfbd4a9a35e902c0cf5be7f12b984fc535f67cce62c0a
=> => exporting attestation manifest sha256:990c8fc873515952fa6f8fc20fa00cd426265058f0669d5cc0368b449cf0575
=> => exporting manifest list sha256:e8e649a851dae32f0eb758f3e6173d5d3d3adf7bd4dc1844f175cc869dae59e
=> => naming to docker.io/library/first-docker:latest
=> => unpacking to docker.io/library/first-docker:latest

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/t921ghpnemg1cl64nehkwxe4v

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:\Software En\Lab8_2> docker run first-docker
Wangphai Jullaech 653380213-5 Boss
PS D:\Software En\Lab8_2>
```

## Lab Worksheet

- (1) คำสั่งที่ใช้ในการ run คือ  
รัน container จาก image first-image
- (2) Option -t ในคำสั่ง \$ docker build ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป  
option-t มีไว้สำหรับ ตั้งชื่อและแท็ก (tag) ให้กับ Docker image ที่สร้างขึ้น

## แบบฝึกปฏิบัติที่ 8.3: การแชร์ Docker image ผ่าน Docker Hub

1. เปิดใช้งาน Docker desktop และ Login ด้วย Username และ Password ที่ลงทะเบียนกับ Docker Hub เอาไว้
2. เปิด Command line หรือ Terminal จากนั้นสร้าง Directory ชื่อ Lab8\_3
3. ย้ายตำแหน่งปัจจุบันไปที่ Lab8\_3 เพื่อใช้เป็น Working directory
4. สร้าง Dockerfile.swp ไว้ใน Working directory

สำหรับเครื่องที่ใช้ระบบปฏิบัติการวินโดวส์ บันทึกคำสั่งต่อไปนี้ลงในไฟล์ โดยใช้ Text Editor ที่มี

```
FROM busybox
```

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

```
CMD echo "ชื่อ-นามสกุล รหัสนักศึกษา"
```

สำหรับเครื่องที่ใช้ระบบปฏิบัติการ MacOS หรือ Linux บนหน้าต่าง Terminal และป้อนคำสั่งต่อไปนี้

```
$ cat > Dockerfile << EOF
```

```
FROM busybox
```

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

```
CMD echo "ชื่อ-นามสกุล รหัสนักศึกษา"
```

```
EOF
```

หรือใช้คำสั่ง

```
$ touch Dockerfile
```

แล้วใช้ Text Editor ในการใส่เนื้อหาแทน

7. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้  
\$ docker build -t <username ที่ลงทะเบียนกับ Docker Hub>/lab8
5. ทำการรัน Docker image บน Container ในเครื่องของตัวเองเพื่อทดสอบผลลัพธ์ ด้วยคำสั่ง  
\$ docker run <username ที่ลงทะเบียนกับ Docker Hub>/lab8

## Lab Worksheet

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

```
PS D:\Software En\Lab8_3> docker build -t wangphai1446/lab8 .
[+] Building 4.4s (6/6) FINISHED                                docker:desktop-linux
=> [internal] load build definition from Dockerfile              0.1s
=> CACHED [1/1] FROM docker.io/library/busybox:latest@sha256:a5d0ce49aa801d475da48f8cb1 3.9s
=> => resolve docker.io/library/busybox:latest@sha256:a5d0ce49aa801d475da48f8cb163c354a 3.9s
=> [auth] library/busybox:pull token for registry-1.docker.io 0.0s
=> exporting to image                                           0.1s
=> => exporting layers                                          0.0s
=> => exporting manifest sha256:03107c22845902134848e3aae859e34c403279376ed139eb336fb6e 0.0s
=> => exporting config sha256:9cd3c99098d09a203757a59d9fbda1a6db4a71874a6b332fd06e18e5b 0.0s
=> => exporting attestation manifest sha256:04d385756c7b6e3ca294cd721df4737170ae9a4400a 0.0s
=> => exporting manifest list sha256:297f3258b47d7e6f2c058135d4ebc43dea06bbddd92d88325b 0.0s
=> => naming to docker.io/wangphai1446/lab8:latest            0.0s
=> => unpacking to docker.io/wangphai1446/lab8:latest         0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/m9qlhni70uvd0tw8lmg0sg1h

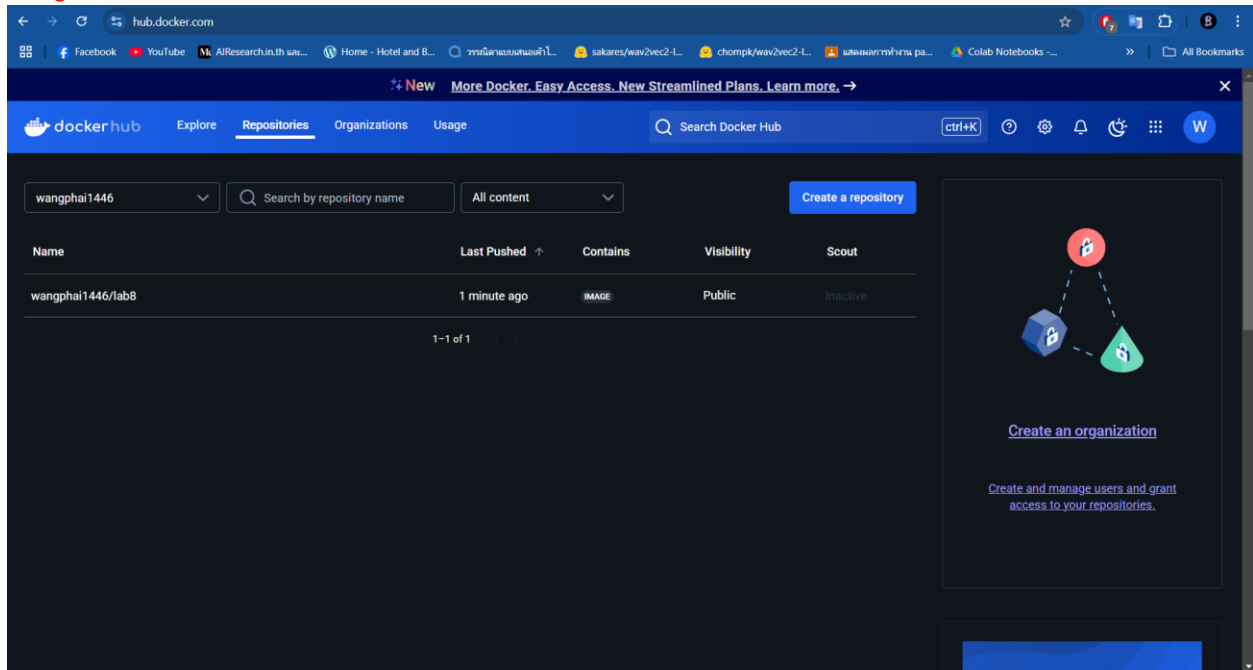
3 warnings found (use docker --debug to expand):
- JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 2)
- MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the same stage because only the last one will be used (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:\Software En\Lab8_3> docker run wangphai1446/lab8
Wangphai Jullapech 653380213-5
PS D:\Software En\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

## Lab Worksheet

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



#### 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

## Lab Worksheet

[Check point#7] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงที่อยู่ของ Source code ที่ Clone มาและเนื้อหาของไฟล์ package.json

The screenshot shows a VS Code editor with the Explorer sidebar on the left displaying a file tree for 'LAB8\_4'. The file 'package.json' is selected and its content is shown in the main editor. The package.json file contains the following JSON:

```
{
  "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
  }
}
```

Below the editor, the TERMINAL panel shows the output of a git clone command:

```
PS D:\Software En\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 | 3.52 MiB/s, done.
Resolving deltas: 100% (523/523), done.
PS D:\Software En\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
5. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้ โดยกำหนดใช้ชื่อ image เป็น myapp\_รหัสสนศ. ไม่มีขีด  
 \$ docker build -t <myapp\_รหัสสนศ. ไม่มีขีด> .



## Lab Worksheet

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

The screenshot shows a VS Code editor with a file explorer on the left containing files like package.json, yarn.lock, Dockerfile, and others. The main editor displays a Dockerfile with the following content:

```

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

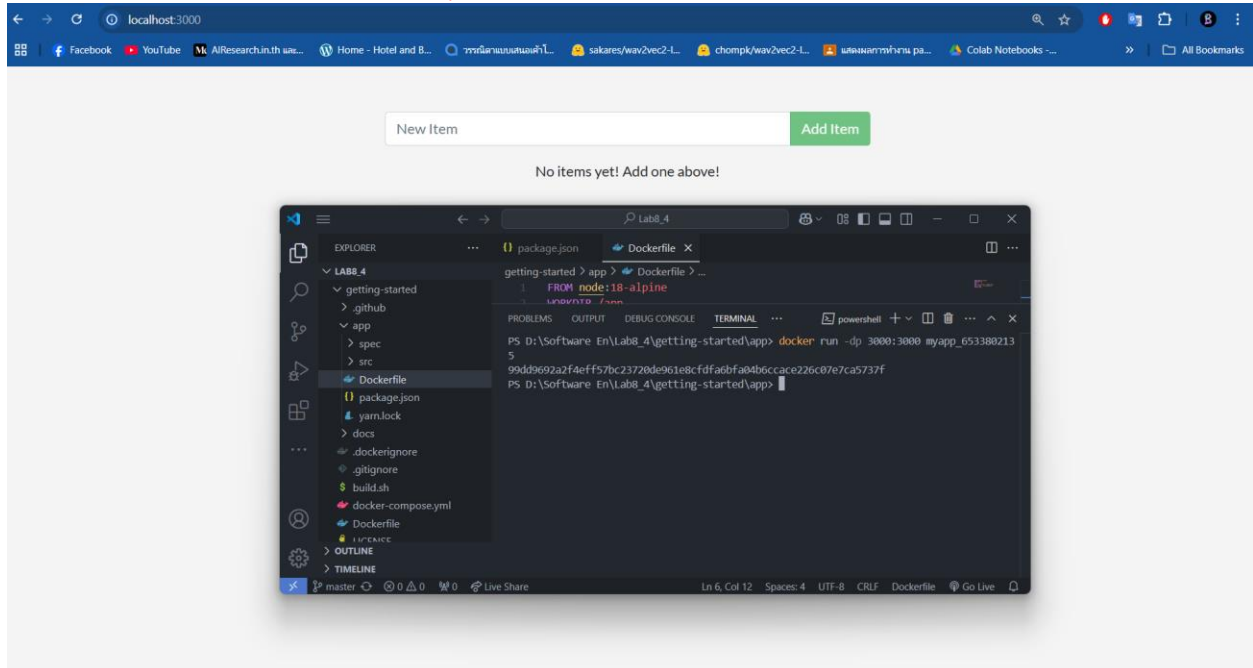
```

The terminal at the bottom shows the output of the command `docker build -t myapp_6533802135 .`. The output includes building the image, transferring the Dockerfile, loading metadata, and finally exporting the manifest. The status bar at the bottom indicates the file is a Dockerfile and the encoding is UTF-8.

6. ทำการ Start ตัว Container ของแอปพลิเคชันที่สร้างขึ้น โดยใช้คำสั่ง  
`$ docker run -dp 3000:3000 <myapp_รหัสสนศ. ไม่มีขีด>`
7. เปิด Browser ไปที่ URL = <http://localhost:3000>

## Lab Worksheet

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



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>

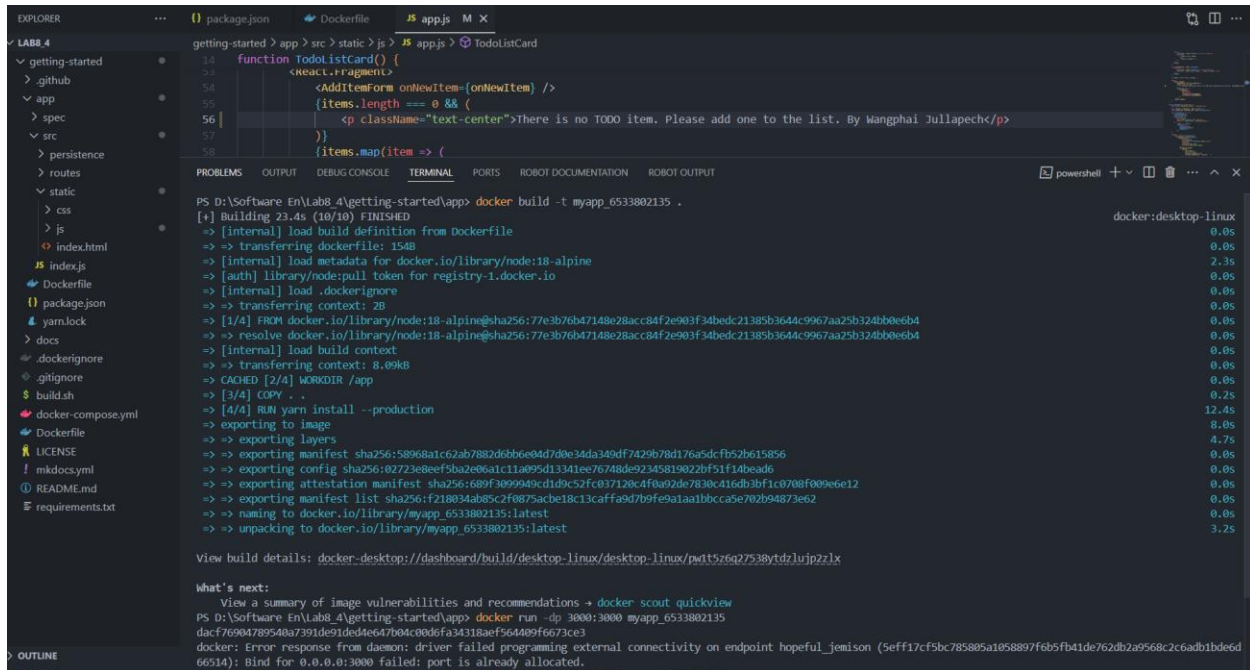
b. Save ไฟล์ให้เรียบร้อย

9. ทำการ Build Docker image โดยใช้คำสั่งเดียวกันกับข้อ 5

10. Start และรัน Container ตัวใหม่ โดยใช้คำสั่งเดียวกันกับข้อ 6

## Lab Worksheet

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



```

package.json
Dockerfile
app.js
M X
LAB8.4
getting-started > app > src > static > js > JS app.js > TodolistCard
14 function TodolistCard() {
15   <React.Fragment>
16     <AddItemForm onNewItem={onNewItem} />
17     {items.length === 0 && (
18       <p className="text-center">There is no TODO item. Please add one to the list. By Wangphai Jullapech</p>
19     )}
20     {items.map(item => (
21       <ItemCard key={item.id} item={item} />
22     ))}
23   </React.Fragment>
24 )}
25
26 |
27
28 items.map(item => {
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS ROBOT DOCUMENTATION ROBOT OUTPUT
PS D:\Software\En\Lab8.4\getting-started\app> docker build -t myapp.6533802135 .
[+] Building 23.4s (10/10) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 154B
=> [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@sha256:77e3b76b47148e28ac84f2e903f34bedc21385b3644c9967aa25b324bb0e6b4
=> resolve docker.io/library/node:18-alpine@sha256:77e3b76b47148e28ac84f2e903f34bedc21385b3644c9967aa25b324bb0e6b4
=> [internal] load build context
=> => transferring context: 8.09kB
=> CACHED [2/4] WORKDIR /app
=> [3/4] COPY . .
=> [4/4] RUN yarn install --production
=> exporting image
=> exporting layers
=> exporting manifest sha256:58968a1c62ab7882d6bb6e0dd7d0e34da349df7429b78d176a5dcfb52b615856
=> exporting config sha256:02723e8eef5ba2e00a1c11a095d13341ee76748de92345819022b5f1f4beade
=> exporting attestation manifest sha256:689f3a9999acdd10c52fc037126c4f0a02d6783bc416db3bfc0708f00e0e6e12
=> exporting manifest list sha256:f210034ab05caf0875cde18c13caffa9d70fe9a1aa1bbcca5e702b94073ee2
=> naming to docker.io/library/myapp.6533802135:latest
=> unpacking to docker.io/library/myapp.6533802135:latest
View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/pa01576q275389td2jup27lx
what's next:
View a summary of image vulnerabilities and recommendations -> docker scout quickview
PS D:\Software\En\Lab8.4\getting-started\app> docker run -dp 3000:3000 myapp.6533802135
daf76904789540a7391de91dd4e647b04c08d0fa34318aef564409f6673ce3
docker: Error response from daemon: driver failed programming external connectivity on endpoint hopeful_jemison (5eff17cf5bc785805a1058897f6b5fb41de762db2a9568c2c6adb1bde6d66514): Bind for 0.0.0.0:3000 failed: port is already allocated.

```

(1) Error ที่เกิดขึ้นหมายความว่าอย่างไร และเกิดขึ้นเพราะอะไร

Port 3000:3000 จะใช้งานสำหรับ container นั้น ถูกใช้งานอยู่แล้ว

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>` เพื่ทำการลบ

b. ผ่าน Docker desktop

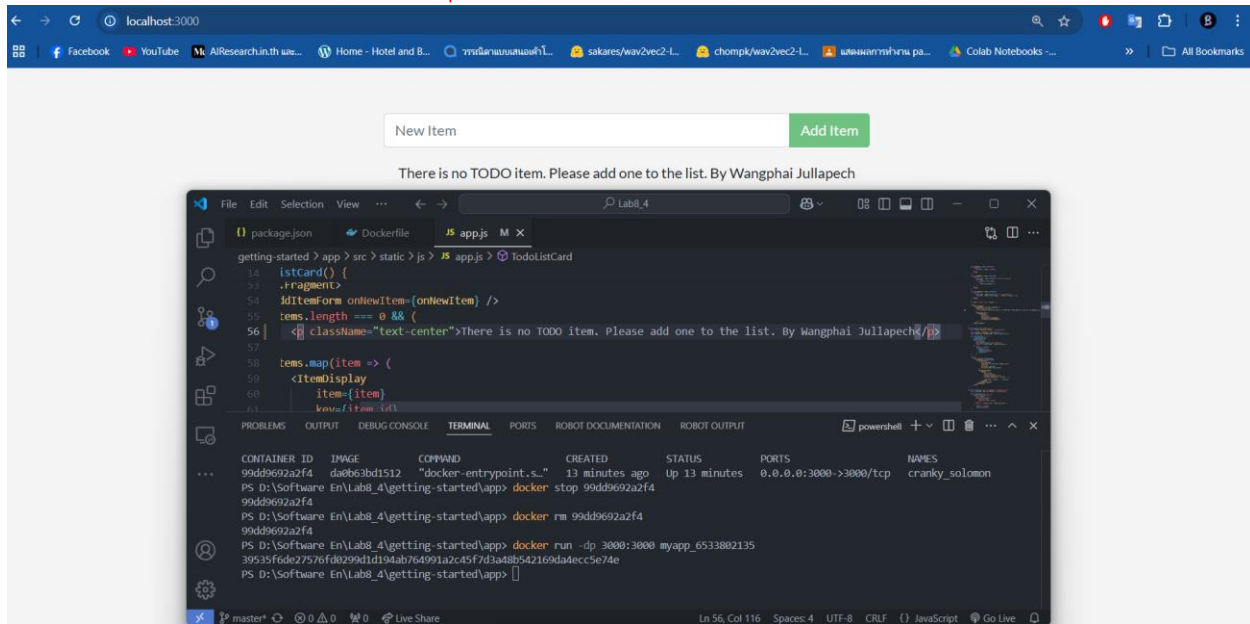
- ไปที่หน้าต่าง Containers
- เลือกไอคอนถังขยะในแถวของ Container ที่ต้องการจะลบ
- ยืนยันโดยการกด Delete forever

12. Start และรัน Container ตัวใหม่อีกครั้ง โดยใช้คำสั่งเดียวกันกับข้อ 6

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

## Lab Worksheet

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



### แบบฝึกปฏิบัติที่ 8.5: เริ่มต้นสร้าง Pipeline อย่างง่ายสำหรับการ Deploy ด้วย Jenkins

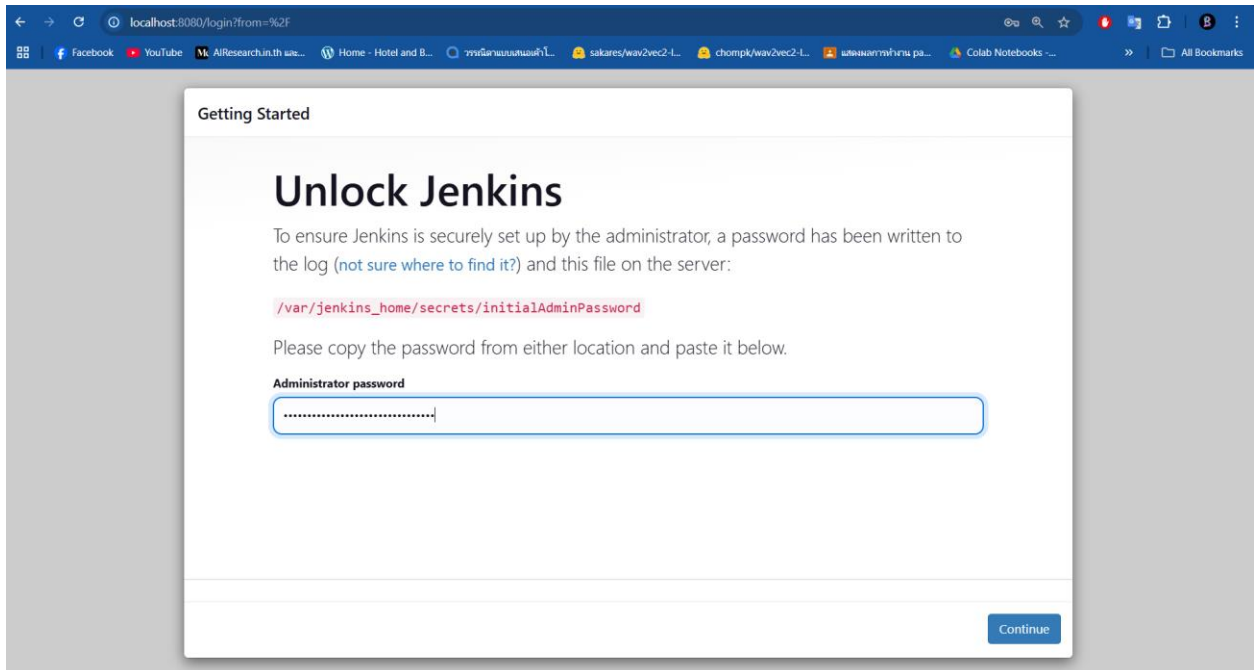
1. เปิด Command line หรือ Terminal บน Docker Desktop
2. ป้อนคำสั่งและทำการรัน container โดยผูกพอร์ต  
`$ docker run -p 8080:8080 -p 50000:50000 --restart=on-failure jenkins/jenkins:its-jdk17`  
 หรือ  
`$ docker run -p 8080:8080 -p 50000:50000 --restart=on-failure -v jenkins_home:/var/jenkins_home jenkins/jenkins:its-jdk17`
3. บันทึกรหัสผ่านของ Admin user ไว้สำหรับ log-in ในครั้งแรก

## Lab Worksheet

**[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:

bcea41c533ee4908ae6176538dcdd871



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

## Lab Worksheet

[Check point#13] Capture หน้าจอที่แสดงผลการตั้งค่า

Getting Started

## Create First Admin User

Username  
wangphai\_2135

Password  
\*\*\*\*\*

Confirm password  
\*\*\*\*\*

Full name  
Wangphai Jullapech

Jenkins 2.479.3

[Skip and continue as admin](#) [Save and Continue](#)

7. กำหนด Jenkins URL เป็น <http://localhost:8080/lab8>

8. เมื่อติดตั้งเรียบร้อยแล้วจะพบหน้าจอ Dashboard ดังแสดงในภาพ

Jenkins

Dashboard

+ New Item

Build History

Manage Jenkins

My Views

Build Queue  
No builds in the queue.

Build Executor Status  
0/2

Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

Create a job

Set up a distributed build

Set up an agent

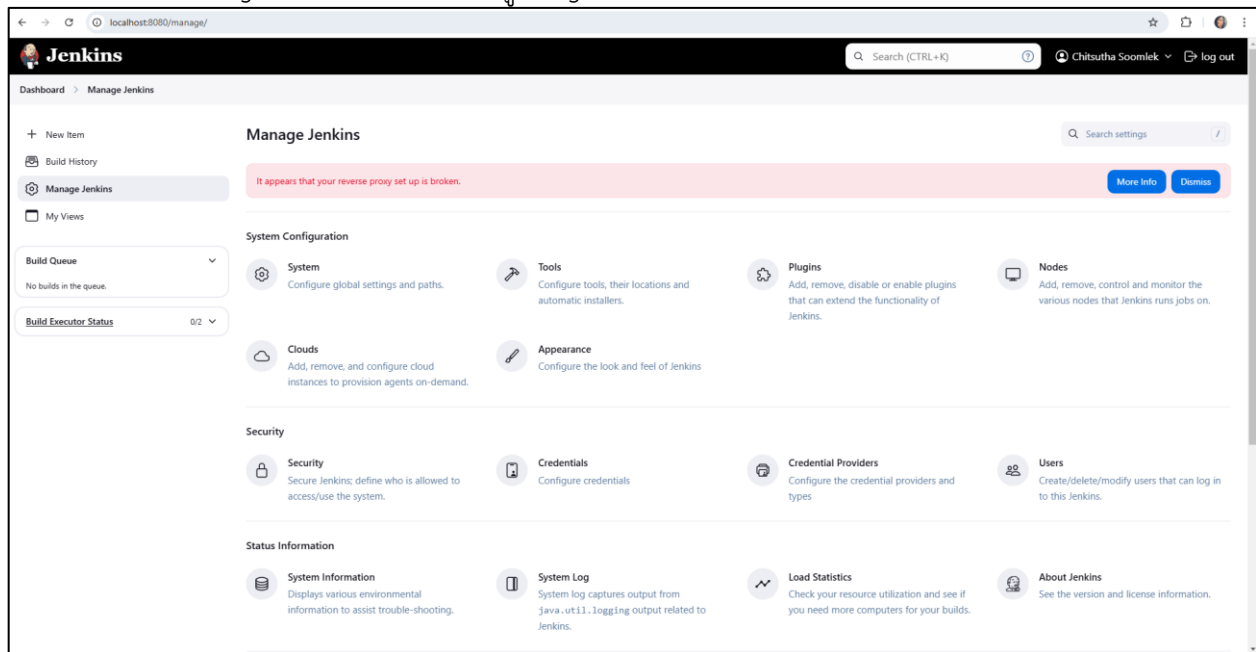
Configure a cloud

Learn more about distributed builds

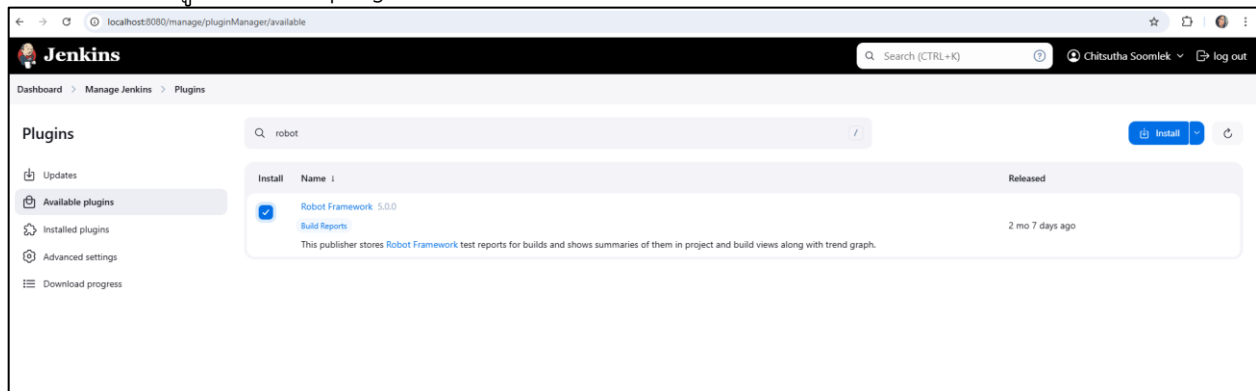
REST API Jenkins 2.479.3

## Lab Worksheet

## 9. เลือก Manage Jenkins แล้วไปที่เมนู Plugins

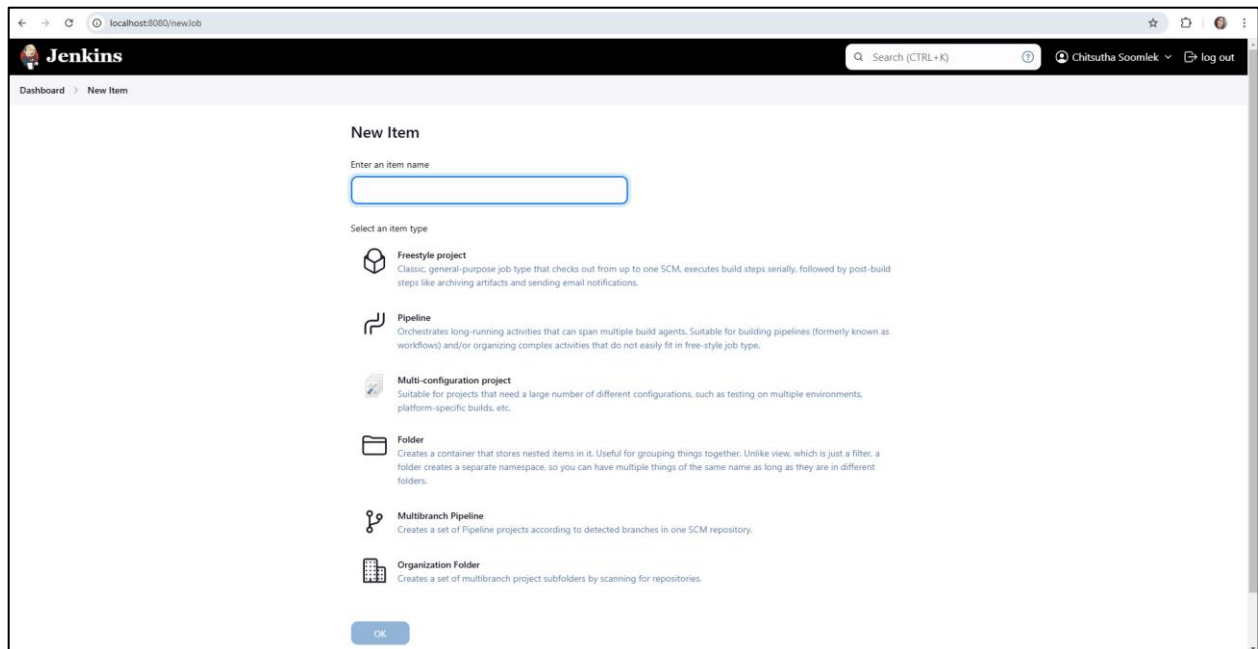


## 10. ไปที่เมนู Available plugins แล้วเลือกติดตั้ง Robotframework เพิ่มเติม



## 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)

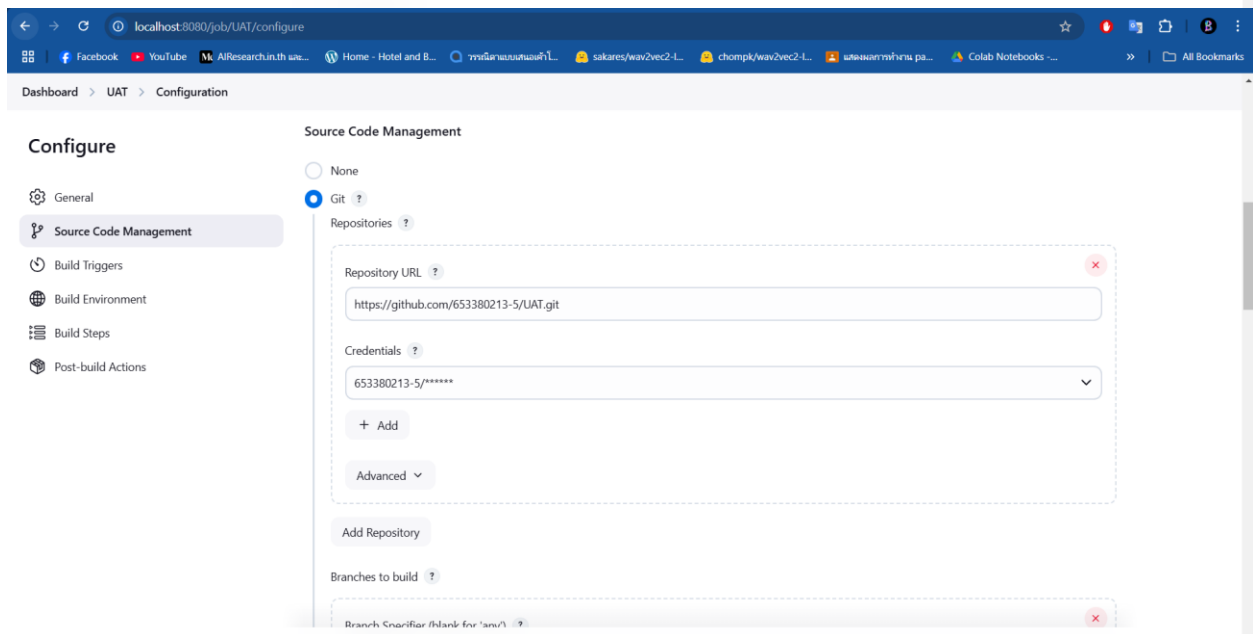
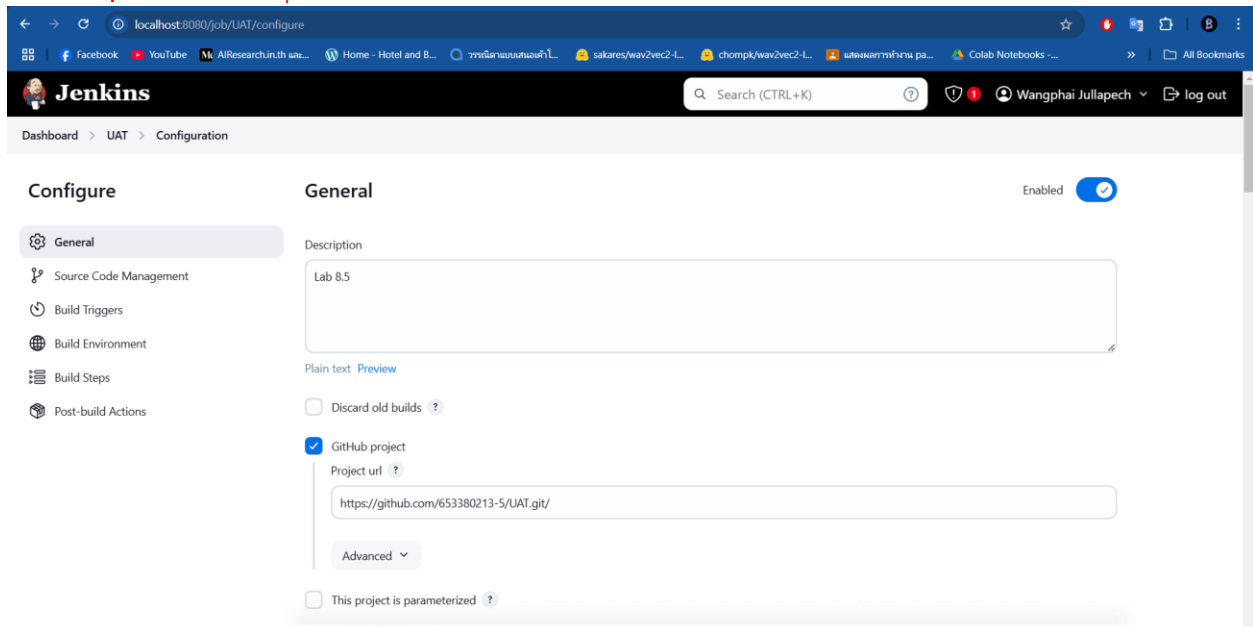
**Build Trigger:** เลือกแบบ Build periodically แล้วกำหนดให้ build ทุก 15 นาที

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



## Lab Worksheet

[Check point#14] Capture หน้าจอแสดงการตั้งค่า พร้อมกับตอบคำถามต่อไปนี้



## Lab Worksheet

The image displays two screenshots of the UAT Configuration interface, showing the configuration for a build job.

**Top Screenshot: Build Triggers**

- Build Triggers:**
  - ☐ Trigger builds remotely (e.g., from scripts) ?
  - ☐ Build after other projects are built ?
  - ☒ Build periodically ?
- Schedule:** H/15 \* \* \* \*
- Next Run:** Would last have run at Wednesday, January 29, 2025 at 8:05:15 AM Coordinated Universal Time; would next run at Wednesday, January 29, 2025 at 8:20:15 AM Coordinated Universal Time.
- ☐ GitHub hook trigger for GITScm polling ?
- ☐ Poll SCM ?

**Build Environment:**

- ☐ Delete workspace before build starts
- ☐ Use secret text(s) or file(s) ?
- ☐ Add timestamps to the Console Output

**Bottom Screenshot: Build Steps**

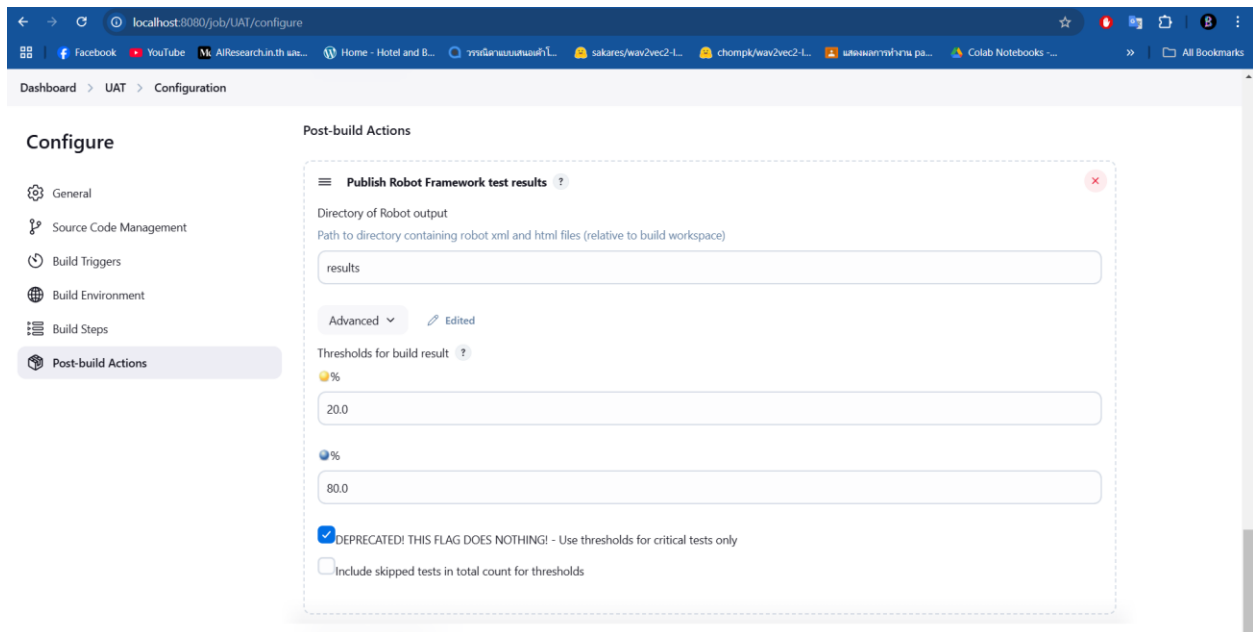
- Build Steps:**
  - Execute shell ?**
    - Command:**

```

./myenv/bin/activate
export PATH=$PATH:/usr/bin
mkdir -p results
robot --outputdir results test01.robot

```
    - Advanced:** (Dropdown menu)
    - Add build step:** (Dropdown menu)
- Post-build Actions:** (Empty section)

## Lab Worksheet



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

`./myenv/bin/activate`

`export PATH=$PATH:/usr/bin`

`mkdir -p results`

`robot --outputdir results test01.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

**Robot Framework Tests Trend (all tests)**

Build	Skipped	Passed	Failed
#32	0	1	0
#33	0	1	0

**Latest Robot Results:**

Total	Failed	Passed	Skipped	Pass %
All tests	1	0	1	100.0

```

Started by timer
Running as SYSTEM
Building in workspace /var/jenkins_home/workspace/UAT
The recommended git tool is: NONE
using credential fdfe0b3d-a602-4ca4-affb-556b70e86269
> 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/653380213-5/UAT.git # timeout=10
Fetching upstream changes from https://github.com/653380213-5/UAT.git
> git --version # timeout=10
> git --version # 'git version 2.39.5'
using GIT_ASKPASS to set credentials
> git fetch --tags --force --progress -- https://github.com/653380213-5/UAT.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision 22637a66e6ac528bb3bade46c53822b7d06876a (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f 22637a66e6ac528bb3bade46c53822b7d06876a # timeout=10
Commit message: "fixed path chrome in file test01.robot"
> git rev-list --no-walk 22637a66e6ac528bb3bade46c53822b7d06876a # timeout=10
[UAT] $ /bin/sh -xe /tmp/jenkins10044693035992657867.sh
+ . /myenv/bin/activate
+ deactivate nondestructive
+ [ -n ]
  
```