

## Lab Worksheet

ชื่อ-นามสกุล นางสาวทัศนันท์ แก้วกลม รหัสนักศึกษา 653380130-9 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

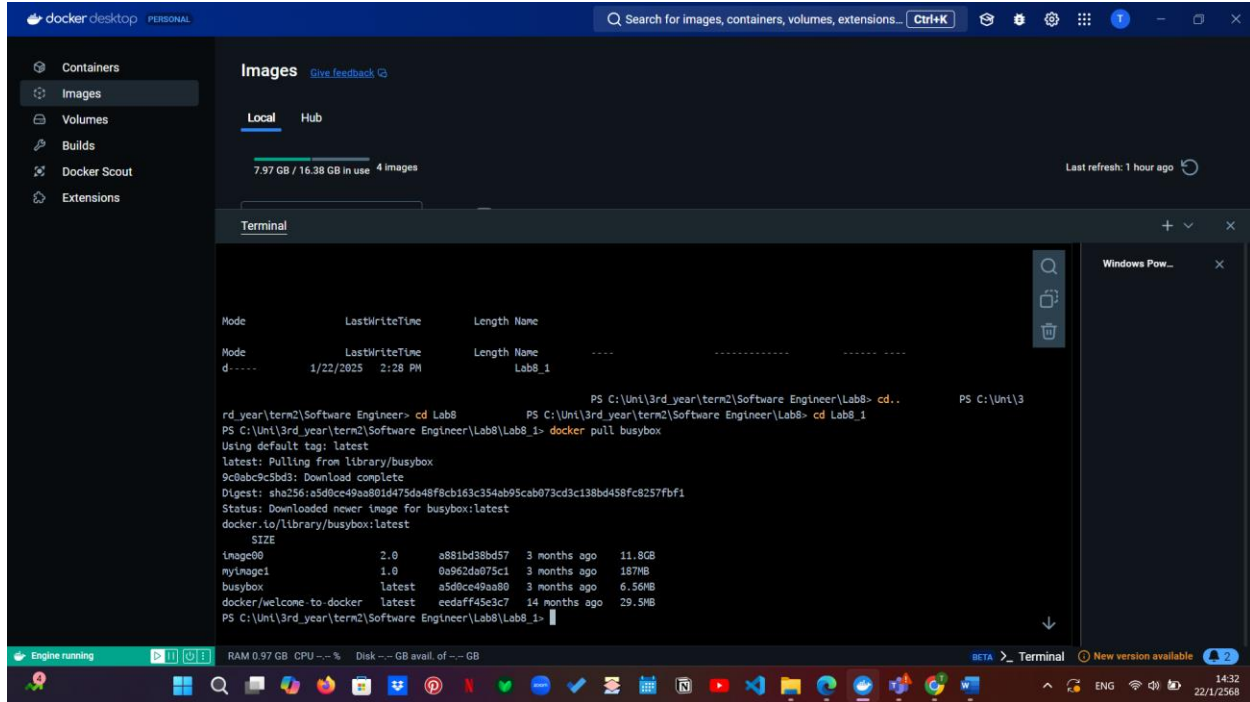
1. ติดตั้ง Docker desktop ลงบนเครื่องคอมพิวเตอร์ โดยดาวน์โหลดจาก <https://www.docker.com/get-started>
2. สร้าง Account บน Docker hub (<https://hub.docker.com/signup>)
3. กำหนดให้ \$ หมายถึง Command prompt และ <> หมายถึง ให้ป้อนค่าของพารามิเตอร์ที่กำหนด

## 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 พร้อมคำตอบคำถามต่อไปนี้



- (1) สิ่งที่อยู่ภายใต้คอลัมน์ Repository คืออะไร ชื่อของ docker image
- (2) Tag ที่ใช้บ่งบอกถึงอะไร version ของแต่ละ docker 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

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

## Lab Worksheet

```

Terminal
PS C:\Uni\3rd_year\term2\Software Engineer\Lab8\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 22 07:38 .
drwxr-xr-x 1 root root      4096 Jan 22 07:38 ..
-rwxr-xr-x 1 root root          0 Jan 22 07:38 .dockerenv
drwxr-xr-x 2 root root     12288 Sep 26 21:31 bin
drwxr-xr-x 5 root root      3600 Jan 22 07:38 dev
drwxr-xr-x 1 root root      4096 Jan 22 07:38 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 210 root root          0 Jan 22 07:38 proc
drwx----- 1 root root      4096 Jan 22 07:38 root
dr-xr-xr-x 11 root root          0 Jan 22 07:38 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:\Uni\3rd_year\term2\Software Engineer\Lab8\Lab8_1> docker run busybox echo "Hello Taksanant Keawklom from busybox"
Hello Taksanant Keawklom from busybox
PS C:\Uni\3rd_year\term2\Software Engineer\Lab8\Lab8_1> docker ps -a

Terminal
PS C:\Uni\3rd_year\term2\Software Engineer\Lab8\Lab8_1> docker ps -a
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS              PORTS          NAMES
05c089552069   busybox   "echo 'Hello Taksana..." 8 seconds ago   Exited (0) 7 seconds ago           ed_robinson
0c1fee4bdf4e   busybox   "sh"                      About a minute ago   Exited (0) About a minute ago           0c1fee4bdf4e
0275356e1d22   busybox   "sh"                      3 minutes ago     Exited (0) 3 minutes ago           0275356e1d22
0903ea39210e   image00:2.0 "jupyter lab --ip=0..." 3 months ago     Exited (255) About an hour ago   8000/tcp   graci
ous_payne
1843562bfd92   image00:2.0 "jupyter lab --ip=0..." 3 months ago     Exited (0) 3 months ago           adori
ng_euclid
c6a5c74df783   image00:2.0 "jupyter lab --ip=0..." 3 months ago     Exited (255) About an hour ago   8000/tcp   compa
ssionate_meninsky
fc5f0761c0f2   image00:2.0 "jupyter lab --ip=0..." 3 months ago     Exited (255) About an hour ago   8000/tcp   deter
mined_margulis
6918ce76de7b   image00:2.0 "jupyter lab --ip=0..." 3 months ago     Exited (255) About an hour ago   8000/tcp   keen_
009d77e8566c   image00:2.0 "jupyter lab --ip=0..." 3 months ago     Exited (0) 3 months ago           frien
dly_sanderson
ed35c69cbd11   image00:2.0 "jupyter lab --ip=0..." 3 months ago     Exited (0) 3 months ago           aweso
me_gates
91df42383b6b   docker/welcome-to-docker:latest "/docker-entrypoint..." 3 months ago     Exited (0) 3 months ago           welco
me-to-docker
PS C:\Uni\3rd_year\term2\Software Engineer\Lab8\Lab8_1>

```

(1) เมื่อใช้ option -it ในคำสั่ง run ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป  
เป็นการรวมสอง flag:

-i (interactive): ทำให้สามารถโต้ตอบกับคอนเทนเนอร์ได้ผ่าน stdin

-t (tty): สร้าง terminal จำลอง เพื่อให้แสดงผลพอร์ทัลในลักษณะเหมือน terminal ปกติ

## Lab Worksheet

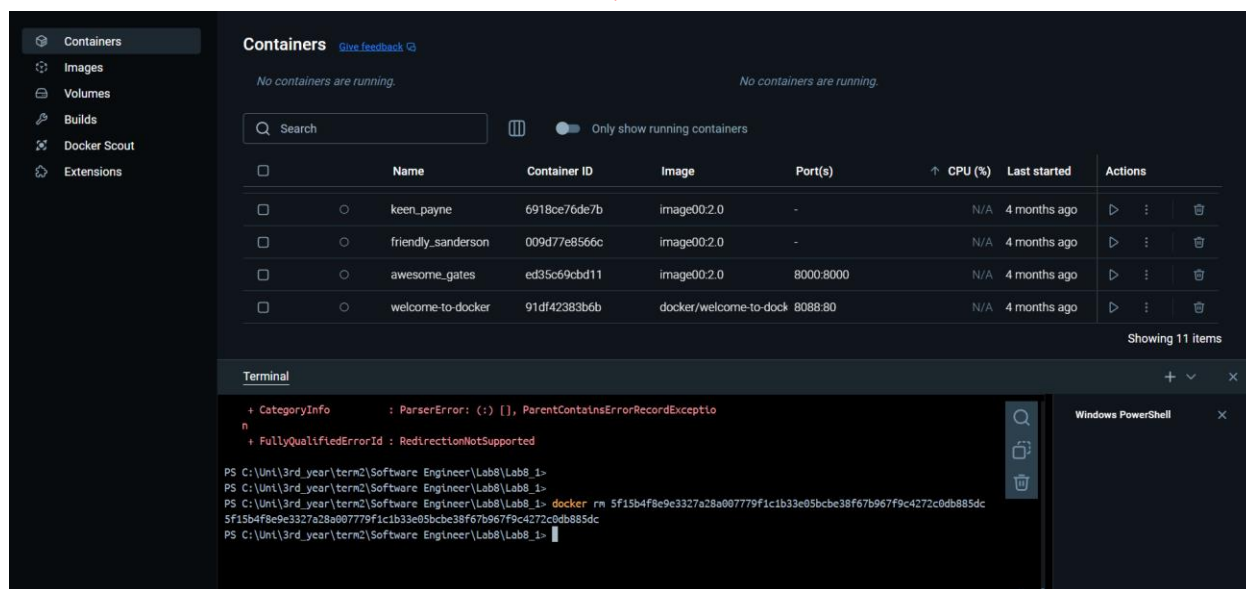
stdin (Standard Input) คือช่องทางมาตรฐานที่ระบบปฏิบัติการใช้สำหรับรับข้อมูลจากผู้ใช้หรือแหล่งข้อมูลภายนอก โดยปกติ stdin มักจะหมายถึง การรับข้อมูลจากคีย์บอร์ด ในบริบทของโปรแกรมหรือเชลล์คำสั่ง (command-line interface)

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

บอกสถานะของคอนเทนเนอร์โดยจะแสดงข้อมูลเกี่ยวกับ สถานะการทำงาน และ ระยะเวลาที่คอนเทนเนอร์อยู่ในสถานะนั้น เช่น คอนเทนเนอร์กำลังทำงาน (running) หยุดการทำงาน (exited) หรือหยุดชั่วคราว (paused)

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

## Lab Worksheet

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

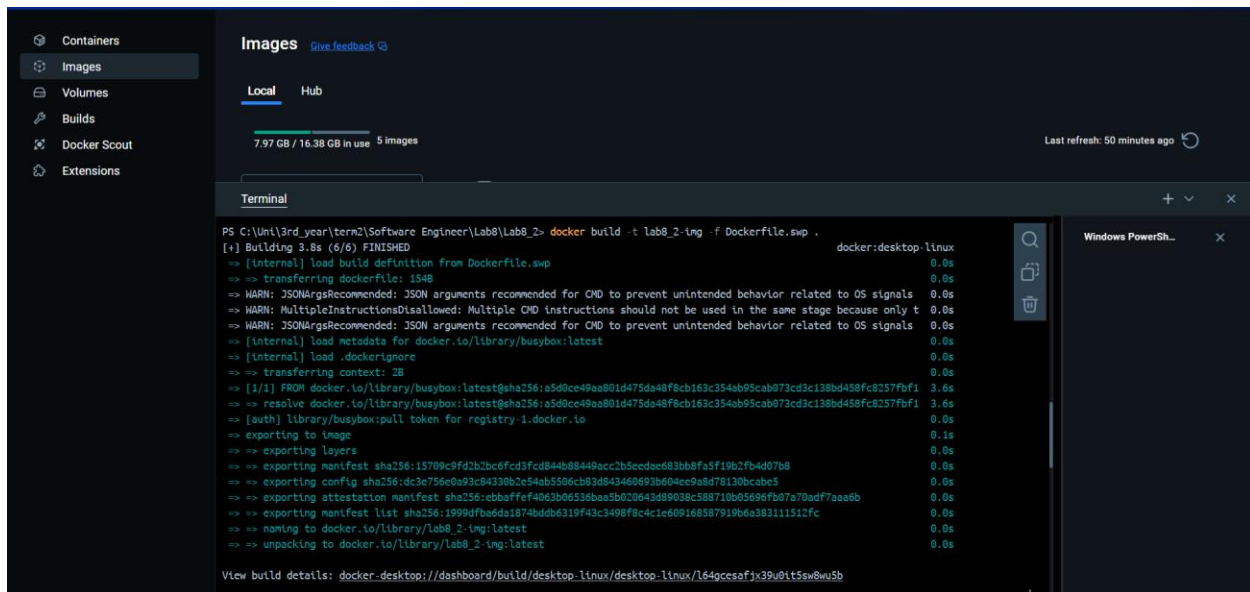
5. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้

\$ docker build -t <ชื่อ Image> .

6. เมื่อ Build สำเร็จแล้ว ให้ทำการรัน Docker image ที่สร้างขึ้นในขั้นตอนที่ 5

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

## Lab Worksheet



(1) คำสั่งที่ใช้ในการ run คือ

`run -d --name lab8_2-container lab8_2-img`

(2) Option -t ในคำสั่ง `$ docker build` ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป  
-t: ใช้สำหรับการตั้งชื่อให้กับ Docker image ที่จะถูกสร้างขึ้น

**ผลการทำงานของ -t :**

- เมื่อใช้ -t คำสั่งจะสร้าง Docker image โดยใช้ชื่อที่กำหนดหลัง -t เช่น lab8\_2-img
- ชื่อนี้จะถูกใช้ในภายหลังเมื่อคุณต้องการรันหรือดึง Docker image

### 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

## Lab Worksheet

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

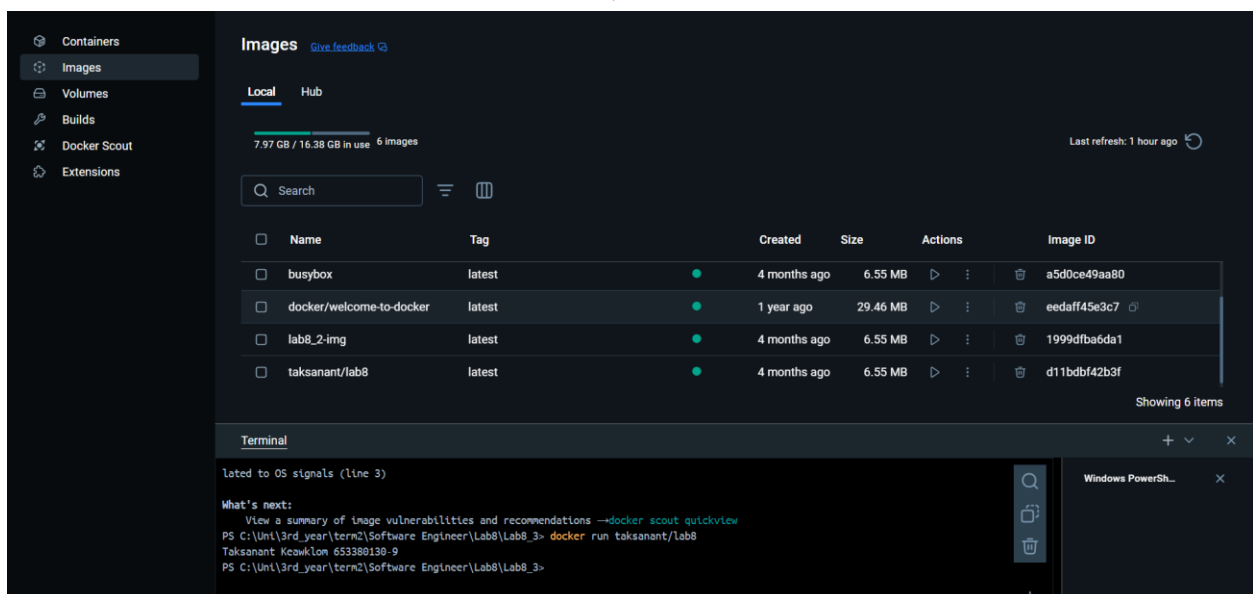
7. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้

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

5. ทำการรัน Docker image บน Container ในเครื่องของตัวเองเพื่อทดสอบผลลัพธ์ ด้วยคำสั่ง

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

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



## Lab Worksheet

6. ทำการ Push ตัว Docker image ไปไว้บน Docker Hub โดยการใช้คำสั่ง

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

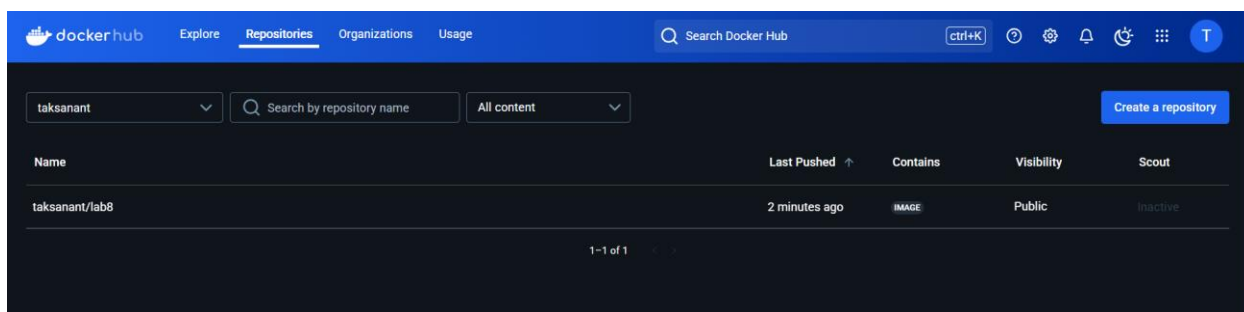
ในกรณีที่ติดปัญหาไม่ได้ Login ไว้ก่อน ให้ใช้คำสั่งต่อไปนี้ เพื่อ Login ก่อนทำการ Push

```
$ 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)



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

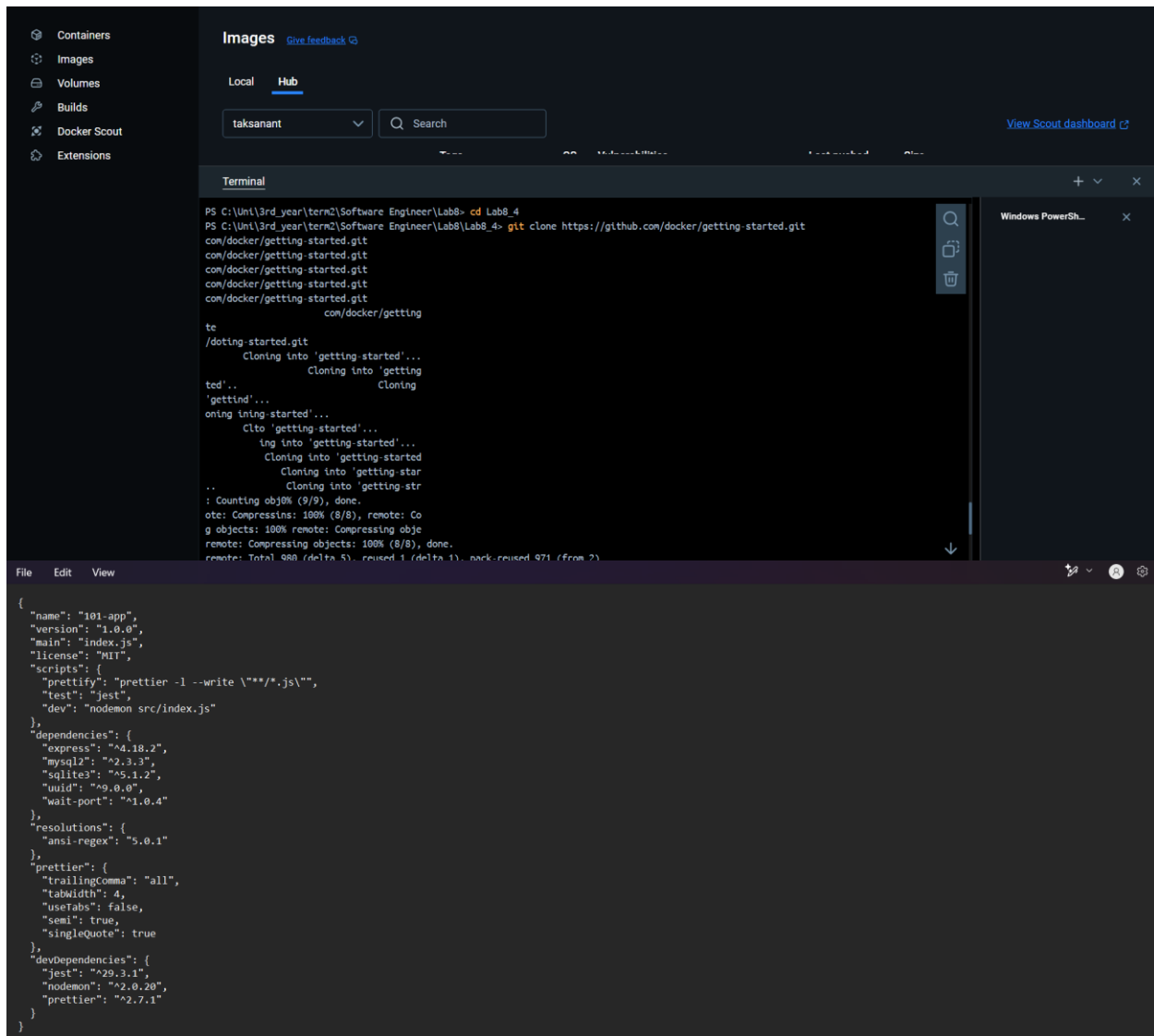
1. เปิด Command line หรือ Terminal จากนั้นสร้าง Directory ชื่อ Lab8\_4
2. ทำการ Clone ซอร์สโค้ดของเว็บแอปพลิเคชันจาก GitHub repository  
<https://github.com/docker/getting-started.git> ลงใน Directory ที่สร้างขึ้น โดยใช้คำสั่ง  

```
$ 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



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\_รหัสนศ. ไม่มีขีด

## Lab Worksheet

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

[Check point#8] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง)

แสดงคำสั่งและผลลัพธ์ที่ได้ทางหน้าจอ

The screenshot shows the Docker Desktop interface. The 'Images' tab is selected, displaying a list of local images. Below the list, a terminal window is open, showing the output of the 'docker build' command.

Name	Tag	Created	Size	Actions	Image ID
busybox	latest	4 months ago	6.55 MB		a5d0ce49aa80
docker/welcome-to-docker	latest	1 year ago	29.46 MB		eedaff45e3c7
lab8_2-img	latest	4 months ago	6.55 MB		1999dfba6da1
taksanant/lab8	latest	4 months ago	6.55 MB		d11bdf42b3f
myapp_6533801309	latest	24 seconds ago	342 MB		a18b5082570d

Showing 7 items

Terminal

```
View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/yky4pwtz2u35pyt6ky3c7g6c
What's next:
View a summary of image vulnerabilities and recommendations --docker scout quickview
PS C:\Uni\3rd_year\term2\Software Engineer\Lab8\Lab8_4\getting-started\app> docker build -t myapp_6533801309 -f Dockerfile.swp .
[+] Building 25.2s (5/9)                                docker:desktop-linux
=> [internal] load build definition from Dockerfile.swp                                0.0s
=> => transferring dockerfile: 160B                                                    0.0s
=> [internal] load metadata for docker.io/library/node:18-alpine                      5.1s
=> [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@sha256:974afb6cbc0314dc6502b14243b8 20.1s
=> => resolve docker.io/library/node:18-alpine@sha256:974afb6cbc0314dc6502b14243b8a 0.0s
=> => sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce36899 444B / 444B 0.6s
=> => sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15b 1.26MB / 1.26MB 1.6s
=> => sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e0d 38.80MB / 40.01MB 20.0s
=> => sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c2440 3.64MB / 3.64MB 4.7s
=> => extracting sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c24403d592 0.1s
=> [internal] load build context                                                       0.9s
=> => transferring context: 4.62MB                                                    0.8s
[+] Building 25.4s (5/9)                                docker:desktop-linux
=> [internal] load build definition from Dockerfile.swp                                0.0s
=> => transferring dockerfile: 160B                                                    0.0s
=> [internal] load metadata for docker.io/library/node:18-alpine                      5.1s
=> [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@sha256:974afb6cbc0314dc6502b14243b8 20.2s
```

## Lab Worksheet

```

Terminal
=> => resolve docker.io/library/node:18-alpine@sha256:974afb6cbc0314dc6502b14243b8a 0.0s
[+] Building 27.1s (8/9)                                docker:desktop-linux
=> [1/4] FROM docker.io/library/node:18-alpine@sha256:974afb6cbc0314dc6502b14243b8a 21.3s
=> => resolve docker.io/library/node:18-alpine@sha256:974afb6cbc0314dc6502b14243b8a 0.0s
=> => sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce36899 444B / 444B 0.6s
=> => sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15b 1.26MB / 1.26MB 1.6s
=> => sha256:37892ffbfc8a871a10f813803949d18c3015a482051d51b7e0d 40.01MB / 40.01MB 20.2s
=> => sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c2440 3.64MB / 3.64MB 4.7s
=> => extracting sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c24403d592 0.1s
=> => extracting sha256:37892ffbfc8a871a10f813803949d18c3015a482051d51b7e0da02525e6 0.9s
=> => extracting sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15b4f26d 0.1s
[+] Building 27.4s (8/9)                                docker:desktop-linux
=> => sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce36899 444B / 444B 0.6s
=> => sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15b 1.26MB / 1.26MB 1.6s
=> => sha256:37892ffbfc8a871a10f813803949d18c3015a482051d51b7e0d 40.01MB / 40.01MB 20.2s
=> => sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c2440 3.64MB / 3.64MB 4.7s
=> => extracting sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c24403d592 0.1s
=> => extracting sha256:37892ffbfc8a871a10f813803949d18c3015a482051d51b7e0da02525e6 0.9s
=> => extracting sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15b4f26d 0.1s
=> => extracting sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15b4f26d 0.1s
=> => extracting sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce36899 0.0s
=> [internal] load build context                                0.9s
[+] Building 27.5s (8/9)                                docker:desktop-linux
=> => sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce36899 444B / 444B 0.6s
=> => sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15b 1.26MB / 1.26MB 1.6s
=> => sha256:37892ffbfc8a871a10f813803949d18c3015a482051d51b7e0d 40.01MB / 40.01MB 20.2s

Terminal
=> => sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c2440 3.64MB / 3.64MB 4.7s
=> => extracting sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c24403d592 0.1s
=> => extracting sha256:37892ffbfc8a871a10f813803949d18c3015a482051d51b7e0da02525e6 0.1s
=> => extracting sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15b4f26d 0.9s
=> => extracting sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce36899 0.0s
=> [internal] load build context                                0.9s
[+] Building 27.7s (8/9)                                docker:desktop-linux
=> => sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce36899 444B / 444B 0.6s
=> => sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15b 1.26MB / 1.26MB 1.6s
=> => sha256:37892ffbfc8a871a10f813803949d18c3015a482051d51b7e0d 40.01MB / 40.01MB 20.2s
=> => sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c2440 3.64MB / 3.64MB 4.7s
=> => extracting sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c24403d592 0.1s
=> => extracting sha256:37892ffbfc8a871a10f813803949d18c3015a482051d51b7e0da02525e6 0.9s
=> => extracting sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15b4f26d 0.1s
=> => extracting sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce36899 0.0s
=> [internal] load build context                                0.9s
[+] Building 27.8s (8/9)                                docker:desktop-linux
=> => sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce36899 444B / 444B 0.6s
=> => sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15b 1.26MB / 1.26MB 1.6s
=> => sha256:37892ffbfc8a871a10f813803949d18c3015a482051d51b7e0d 40.01MB / 40.01MB 20.2s
=> => sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c2440 3.64MB / 3.64MB 4.7s
=> => extracting sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c24403d592 0.1s
=> => extracting sha256:37892ffbfc8a871a10f813803949d18c3015a482051d51b7e0da02525e6 0.9s
=> => extracting sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15b4f26d 0.1s
=> => extracting sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce36899 0.0s

```

## Lab Worksheet

```

=> [internal] load build context                                0.9s
[+] Building 28.0s (8/9)                                       docker:desktop-linux
=> => sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce36899 444B / 444B 0.6s
=> => sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15b 1.26MB / 1.26MB 1.6s
=> => sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e0d 40.01MB / 40.01MB 20.2s
=> => sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c2440 3.64MB / 3.64MB 4.7s
=> => extracting sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c24403d592 0.1s
=> => extracting sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e0da02525e6 0.9s
=> => extracting sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15bf426d 0.1s
=> => extracting sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce36899 0.0s
=> [internal] load build context                                0.9s
[+] Building 28.1s (8/9)                                       docker:desktop-linux
=> => sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce36899 444B / 444B 0.6s
=> => sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15b 1.26MB / 1.26MB 1.6s
=> => sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e0d 40.01MB / 40.01MB 20.2s
=> => sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c2440 3.64MB / 3.64MB 4.7s
=> => extracting sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c24403d592 0.1s
=> => extracting sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e0da02525e6 0.9s
=> => extracting sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15bf426d 0.1s
=> => extracting sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce36899 0.0s
=> [internal] load build context                                0.9s
[+] Building 28.3s (8/9)                                       docker:desktop-linux
=> => sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce36899 444B / 444B 0.6s
=> => sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15b 1.26MB / 1.26MB 1.6s
=> => sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e0d 40.01MB / 40.01MB 20.2s

Terminal
=> => sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c2440 3.64MB / 3.64MB 4.7s
=> => extracting sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c24403d592 0.1s
=> => extracting sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e0da02525e6 0.9s
=> => extracting sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15bf426d 0.1s
=> => extracting sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce36899 0.0s
=> [internal] load build context                                0.9s
[+] Building 28.4s (8/9)                                       docker:desktop-linux
=> => sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce36899 444B / 444B 0.6s
=> => sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15b 1.26MB / 1.26MB 1.6s
=> => sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e0d 40.01MB / 40.01MB 20.2s
=> => sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c2440 3.64MB / 3.64MB 4.7s
=> => extracting sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c24403d592 0.1s
=> => extracting sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e0da02525e6 0.9s
=> => extracting sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15bf426d 0.1s
=> => extracting sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce36899 0.0s
=> [internal] load build context                                0.9s
[+] Building 28.6s (8/9)                                       docker:desktop-linux
=> => sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce36899 444B / 444B 0.6s
=> => sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15b 1.26MB / 1.26MB 1.6s
=> => sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e0d 40.01MB / 40.01MB 20.2s
=> => sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c2440 3.64MB / 3.64MB 4.7s
=> => extracting sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c24403d592 0.1s
=> => extracting sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e0da02525e6 0.9s
=> => extracting sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15bf426d 0.1s
=> => extracting sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce36899 0.0s

```

## Lab Worksheet

```

Terminal
=> [internal] load build context                                0.9s
[+] Building 28.7s (8/9)                                       docker:desktop-linux
=> => sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce36899 444B / 444B 0.6s
=> => sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15b 1.26MB / 1.26MB 1.6s
=> => sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e0d 40.01MB / 40.01MB 20.2s
=> => sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c2440 3.64MB / 3.64MB 4.7s
=> => extracting sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c24403d592 0.1s
=> => extracting sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e0da02525e6 0.9s
=> => extracting sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15bf426d 0.1s
=> => extracting sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce368990 0.0s
=> [internal] load build context                                0.9s
[+] Building 28.9s (8/9)                                       docker:desktop-linux
=> => sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce36899 444B / 444B 0.6s
=> => sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15b 1.26MB / 1.26MB 1.6s
=> => sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e0d 40.01MB / 40.01MB 20.2s
=> => sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c2440 3.64MB / 3.64MB 4.7s
=> => extracting sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c24403d592 0.1s
=> => extracting sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e0da02525e6 0.9s
=> => extracting sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15bf426d 0.1s
=> => extracting sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce368990 0.0s
=> [internal] load build context                                0.9s
[+] Building 29.0s (8/9)                                       docker:desktop-linux
=> => sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce36899 444B / 444B 0.6s
=> => sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15b 1.26MB / 1.26MB 1.6s
=> => sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e0d 40.01MB / 40.01MB 20.2s

Terminal
=> => sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c2440 3.64MB / 3.64MB 4.7s
=> => extracting sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c24403d592 0.1s
=> => extracting sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e0da02525e6 0.9s
=> => extracting sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15bf426d 0.1s
=> => extracting sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce368990 0.0s
=> [internal] load build context                                0.9s
[+] Building 29.2s (8/9)                                       docker:desktop-linux
=> => sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce36899 444B / 444B 0.6s
=> => sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15b 1.26MB / 1.26MB 1.6s
=> => sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e0d 40.01MB / 40.01MB 20.2s
=> => sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c2440 3.64MB / 3.64MB 4.7s
=> => extracting sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c24403d592 0.1s
=> => extracting sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e0da02525e6 0.9s
=> => extracting sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15bf426d 0.1s
=> => extracting sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce368990 0.0s
=> [internal] load build context                                0.9s
[+] Building 29.3s (8/9)                                       docker:desktop-linux
=> => sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce36899 444B / 444B 0.6s
=> => sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15b 1.26MB / 1.26MB 1.6s
=> => sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e0d 40.01MB / 40.01MB 20.2s
=> => sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c2440 3.64MB / 3.64MB 4.7s
=> => extracting sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c24403d592 0.1s
=> => extracting sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e0da02525e6 0.9s
=> => extracting sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15bf426d 0.1s
=> => extracting sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce368990 0.0s

```

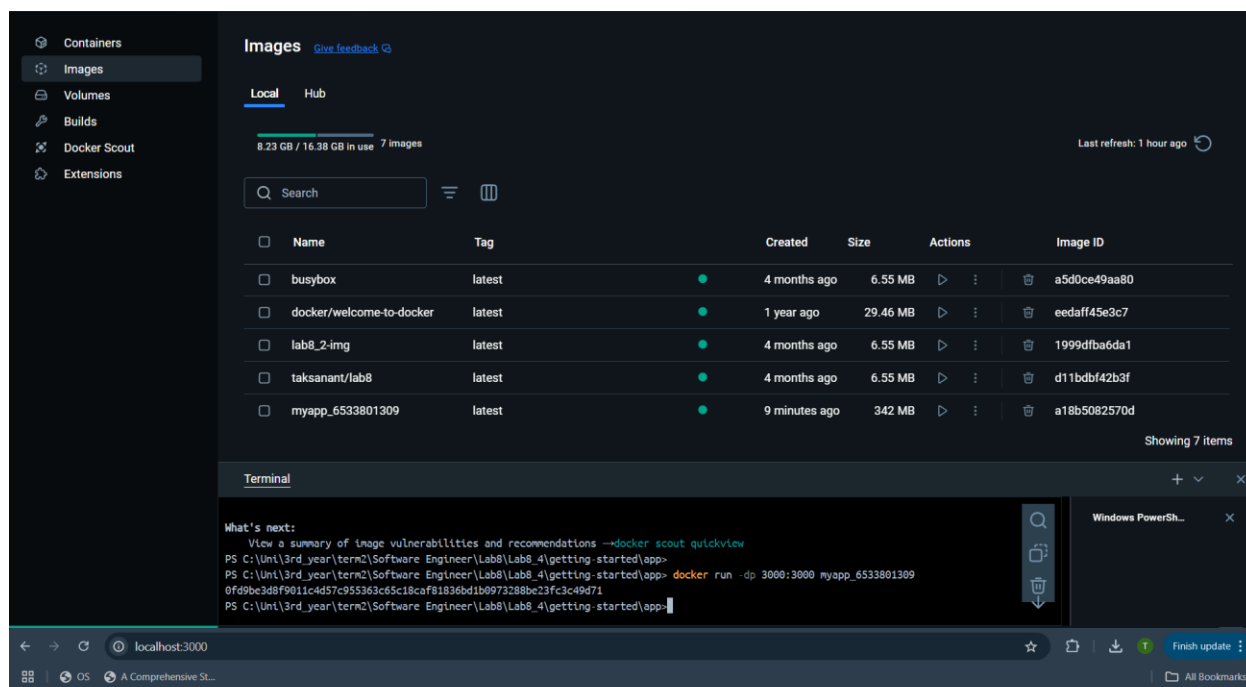
6. ทำการ Start ตัว Container ของแอปพลิเคชันที่สร้างขึ้น โดยใช้คำสั่ง

\$ 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

## 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

## Lab Worksheet

9. ทำการ Build Docker image โดยใช้คำสั่งเดียวกันกับข้อ 5

10. Start และรัน Container ตัวใหม่ โดยใช้คำสั่งเดียวกันกับข้อ 6

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

The screenshot displays the Docker Desktop interface. On the left, the sidebar shows 'Containers', 'Images', 'Volumes', 'Builds', 'Docker Scout', and 'Extensions'. The main area is divided into two sections. The top section, titled 'Images', shows a list of local images with columns for Name, Tag, Created, Size, Actions, and Image ID. The bottom section, titled 'Terminal', shows the execution of Docker commands. The first command is 'docker build -t lab8\_4-img -f Dockerfile.swp .', which successfully builds the image. The second command is 'docker run -dp 3000:3000 lab8\_4-img', which fails with an error message: 'docker: Error response from daemon: driver failed programmatic external connectivity on endpoint wonderful\_shamir (5f2dab71c3de782d5bedc306ea10d0b3bb74c34ae6c089398da0cf290aed777): Bind for 0.0.0.0:3000 failed: port is already allocated.'

Name	Tag	Created	Size	Actions	Image ID
lab8_2-img	latest	4 months ago	6.55 MB		1999dfba6da1
taksanant/lab8	latest	4 months ago	6.55 MB		d11bdbf42b3f
myapp_6533801309	latest	22 minutes ag	342 MB		a18b5082570d
lab8_4-img	latest	3 minutes ago	342 MB		42944483dfb7

```

\3rd_year\term2\Software Engineer\Lab8\Lab8_4\getting-started\app> docker build -t lab8_4-img -f Dockerfile.swp .
0s (10/10) FINISHED
docker:desktop-linux
=> [internal] load build definition from Dockerfile.swp 0.0s
=> transferring Dockerfile: 160B 0.0s
=> [internal] load metadata for docker.io/library/node: 3.0s
=> [auth] library/node:pull token for registry-1.docker 0.0s
=> [internal] load .dockerignore 0.0s
=> transferring context: 2B 0.0s
=> [1/4] FROM docker.io/library/node:18-alpine@sha256:9 0.0s
=> resolve docker.io/library/node:18-alpine@sha256:9 0.0s
=> [internal] load build context 0.0s

```

```

PS C:\Unit3rd_year\term2\Software Engineer\Lab8\Lab8_4\getting-started\app> docker run -dp 3000:3000 lab8_4-img
3ac10803cd9723b73c7111a8e74c3bb23cfcfb49a780a05d3e7bac5080b2f425
docker: Error response from daemon: driver failed programmatic external connectivity on endpoint wonderful_shamir (5f2dab71c3de782d5bedc306ea10d0b3bb74c34ae6c089398da0cf290aed777): Bind for 0.0.0.0:3000 failed: port is already allocated.
PS C:\Unit3rd_year\term2\Software Engineer\Lab8\Lab8_4\getting-started\app>

```

(1) Error ที่เกิดขึ้นหมายความว่าอย่างไร และเกิดขึ้นเพราะอะไร

ข้อผิดพลาดนี้เกิดจากการที่พอร์ต 3000 บนเครื่องของคุณถูกใช้งานแล้ว ซึ่งอาจจะมีแอปพลิเคชันหรือ container อื่นๆ ที่กำลังใช้พอร์ตนี้อยู่



## Lab Worksheet

11. ลบ Container ของ Web application เวอร์ชันก่อนแก้ไขออกจากระบบ โดยใช้วิธีใดวิธีหนึ่งดังต่อไปนี้

a. ผ่าน Command line interface

- i. ใช้คำสั่ง `$ docker ps` เพื่อดู Container ID ที่ต้องการจะลบ
- ii. Copy หรือบันทึก Container ID ไว้
- iii. ใช้คำสั่ง `$ docker stop <Container ID ที่ต้องการจะลบ>` เพื่อหยุดการทำงานของ Container ดังกล่าว
- iv. ใช้คำสั่ง `$ docker rm <Container ID ที่ต้องการจะลบ>` เพื่อทำการลบ

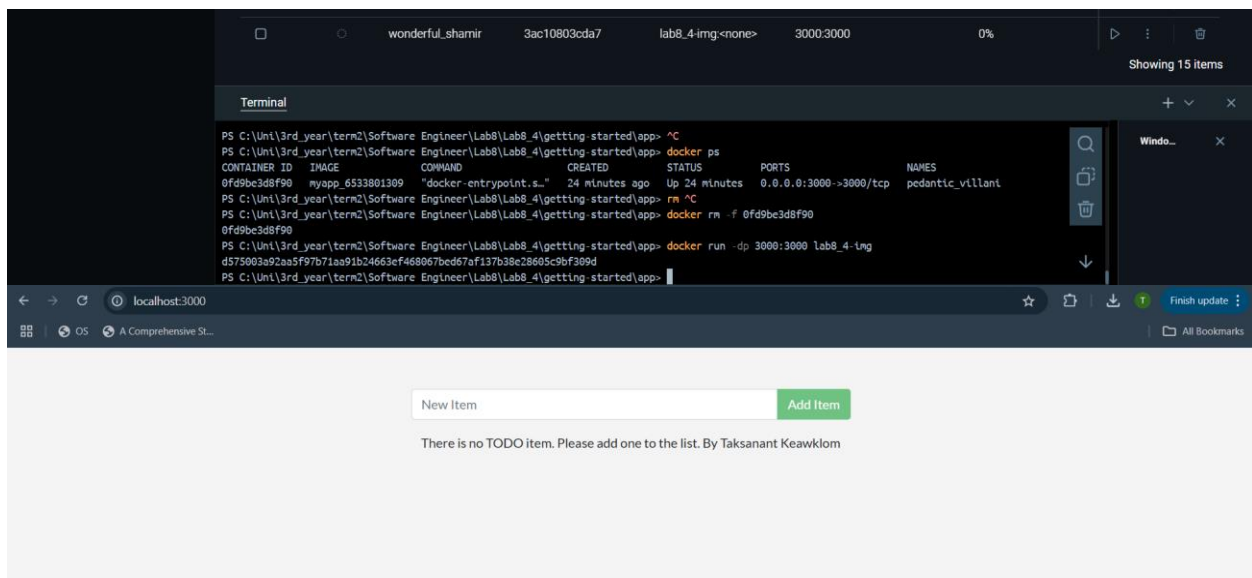
b. ผ่าน Docker desktop

- i. ไปที่หน้าต่าง Containers
- ii. เลือกไอคอนถังขยะในแถวของ Container ที่ต้องการจะลบ
- iii. ยืนยันโดยการกด Delete forever

12. Start และรัน Container ตัวใหม่อีกครั้ง โดยใช้คำสั่งเดียวกันกับข้อ 6

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

[Check point#11] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงผลลัพธ์ที่ได้นับ 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. บ้อนคำสั่งและทำการรัน 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. บันทึกกรหัสผ่านของ Admin user ไว้สำหรับ log-in ในครั้งแรก

[Check point#12] Capture หน้าจอที่แสดงผล Admin password

The screenshot shows the Docker Desktop interface. On the left, the 'Containers' tab is selected. The main area displays a table of containers with one container named 'awesome\_gates' running. Below the table, a terminal window is open, showing the output of the Jenkins installation. The terminal output includes the following text:

```
2025-02-01 17:33:06.280+0000 [id=43] INFO jenkins.install.SetupWizard#init:
*****
*****
*****
Jenkins initial setup is required. An admin user has been created and a password generated.
Please use the following password to proceed to installation:
6a45751acf4949bd8596e8ca5f8154ef
This may also be found at: /var/jenkins_home/secrets/initialAdminPassword
*****
ctorRunner$!onAttained: Completed initialization
2025-02-01 17:33:13.699+0000 [id=25] INFO hudson.lifecycle.Lifecycle$onReady: Jenkins is fully up and running
2025-02-01 17:33:16.206+0000 [id=60] INFO h.n.DownloadService$Downloadable$load: Obtained the updated data file for hudson.tasks.Maven.MavenInstaller
2025-02-01 17:33:16.206+0000 [id=60] INFO hudson.util.Retrier$start: Performed the action check updates server successfully at the attempt #1
```

4. เมื่อได้รับการยืนยันว่า Jenkins is fully up and running ให้เปิดบราวเซอร์ และบ้อนที่อยู่เป็น

localhost:8080

5. ทำการ Unlock Jenkins ด้วยรหัสผ่านที่ได้ในข้อที่ 3

6. สร้าง Admin User โดยใช้ username เป็นชื่อจริงของนักศึกษาพร้อมรหัสสี่ตัวท้าย เช่น somsri\_3062

[Check point#13] Capture หน้าจอที่แสดงผลการตั้งค่า

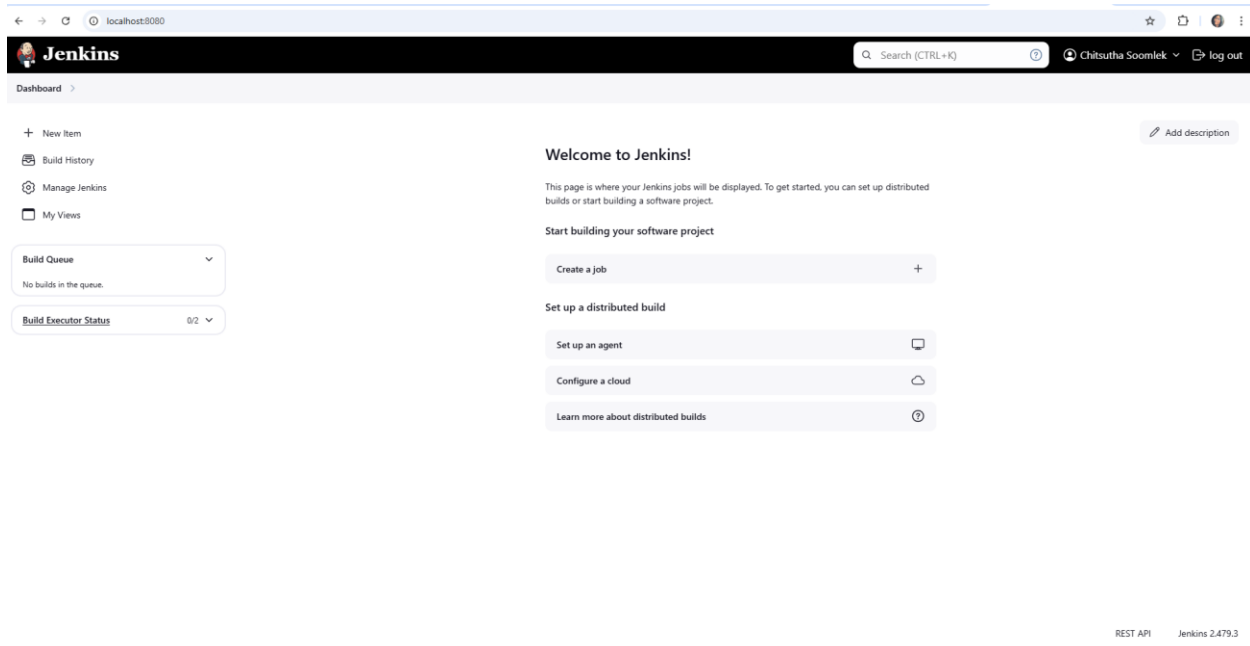
Taksanant

Jenkins User ID: taksanant\_1309

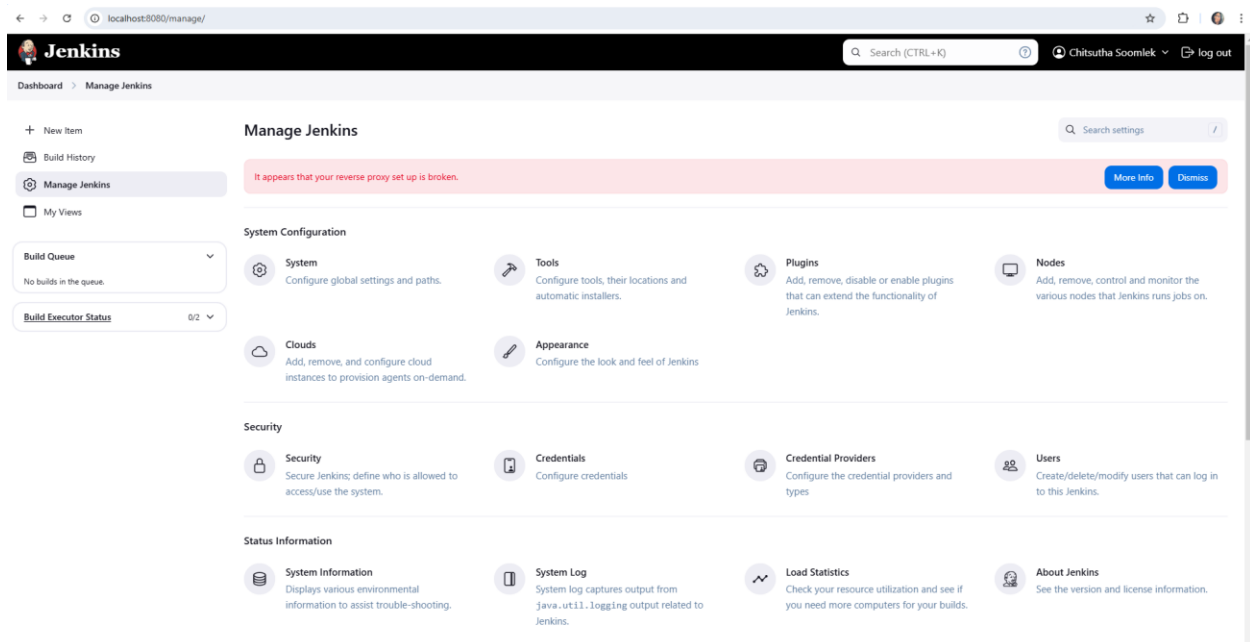
Add description

## Lab Worksheet

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



9. เลือก Manage Jenkins แล้วไปที่เมนู Plugins

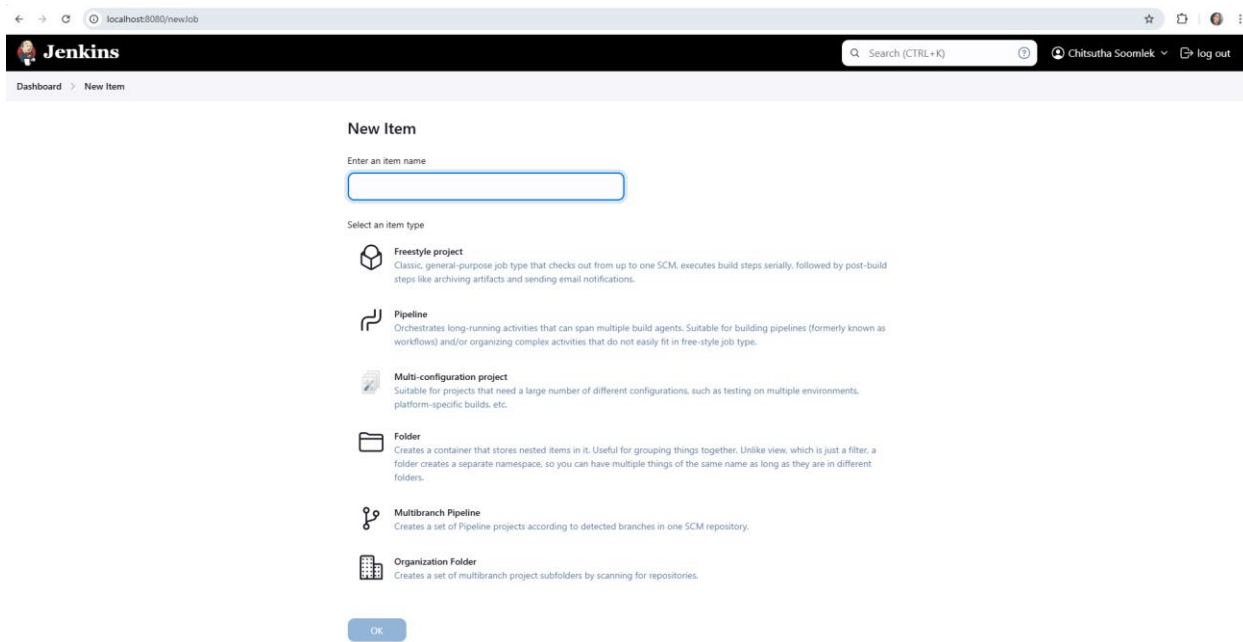


## Lab Worksheet

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

## Lab Worksheet

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

The screenshot displays the Jira Build Configuration interface. The top navigation bar shows 'Dashboard > UAT > Configuration'. The left sidebar, titled 'Configure', lists various settings: General (selected), Source Code Management, Build Triggers, Build Environment, Build Steps, and Post-build Actions. The main content area is divided into two tabs: 'General' and 'Source Code Management'.

In the 'General' tab, the 'Description' field is empty. Below it, the 'Plain text' preview is also empty. There are checkboxes for 'Discard old builds' (unchecked), 'GitHub project' (checked), 'This project is parameterized' (unchecked), and 'Throttle builds' (unchecked). The 'Project url' field is populated with 'https://github.com/Taksanant/Lab7-RobotTesting/'. There are 'Save' and 'Apply' buttons at the bottom.

The 'Source Code Management' tab is also visible, showing the 'Repositories' section. The 'Repository URL' field is populated with 'https://github.com/Taksanant/Lab7-RobotTesting'. The 'Credentials' dropdown menu is set to '- none -'. There is an '+ Add' button and an 'Advanced' dropdown menu. 'Save' and 'Apply' buttons are also present at the bottom of this section.

## Lab Worksheet

Dashboard > UAT > Configuration

### Configure

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

Branches to build ?

Branch Specifier (blank for 'any') ?

Add Branch

Repository browser ?

(Auto)

Additional Behaviours

Add

#### Build Triggers

☐ Trigger builds remotely (e.g., from scripts) ?

☐ Build after other projects are built ?

☒ Build periodically ?

Schedule ?

Would last have run at Sunday, February 2, 2025 at 12:20:40 PM Coordinated Universal Time; would next run at Sunday, February 2, 2025 at 12:35:40 PM Coordinated Universal Time.

☐ GitHub hook trigger for GITScm polling ?

☐ Poll SCM ?

#### Build Environment

☐ Delete workspace before build starts

### Configure

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

#### Build Steps

**Execute shell** ?

Command

See the list of available environment variables

Advanced

Add build step

#### Post-build Actions

## Lab Worksheet

Dashboard > UAT > Configuration

### Configure

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

#### 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 ?

🟡 %

🟢 %

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

Save Apply

- (1) คำสั่งที่ใช้ในการ Execute ไฟล์ .robot ใน Build Steps คือ
- robot <ชื่อไฟล์.robot>
- robot webdriver

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

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

13. กด Apply และ Save

14. สั่ง Build Now

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

## Lab Worksheet

Dashboard > UAT > #13 > Robot result

Status

</> Changes

Console Output

Edit Build Information

Timings

Git Build Data

**Robot Results**

← Previous Build

### Robot Framework Test Results

**Executed:** 2025-01-09T21:00:44.745846

**Duration:** 0:00:04.336 (±0)

**Status:** 1 critical test, 1 passed, 0 failed, 0 skipped  
1 test total (±0), 1 passed, 0 failed, 0 skipped

**Results:** [report.html](#)  
[log.html](#)  
[Original result files](#)

#### Test Result Trend

Zoom to changes ☐ Show only failed ☐ Show only critical ☐ all Max builds Show bigger image

---

**Jenkins** Search (CTRL+K) Taksanant log out

Dashboard > UAT > #13 > Console Output

Status

</> Changes

**Console Output** Download Copy View as plain text

Delete build '#13'

Timings

Git Build Data

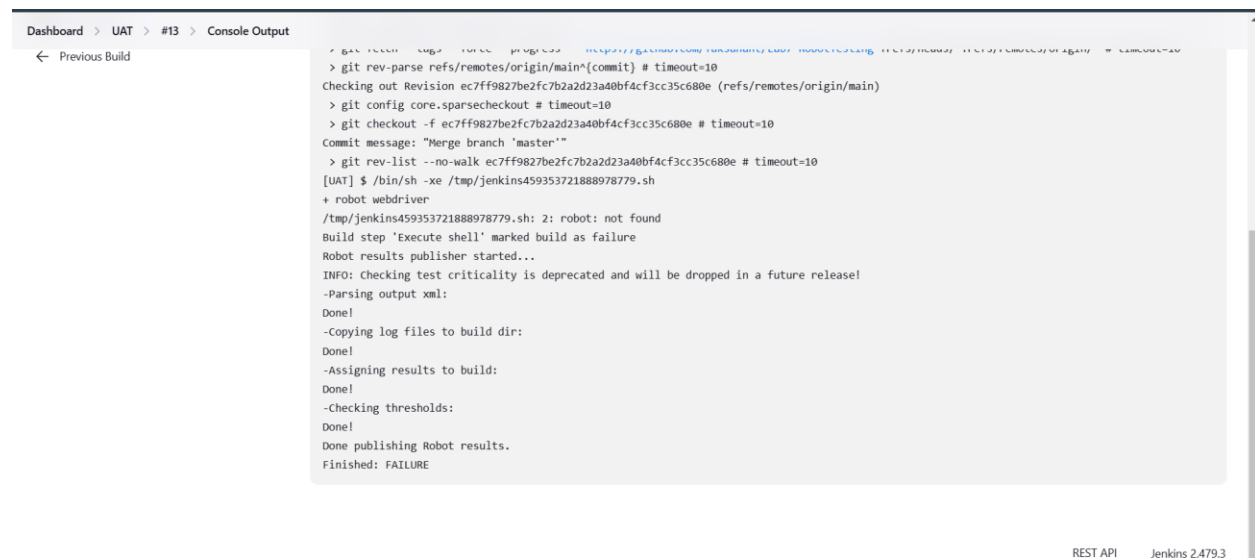
Robot Results

← Previous Build

```

Started by user Taksanant
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/Taksanant/Lab7-RobotTesting # timeout=10
Fetching upstream changes from https://github.com/Taksanant/Lab7-RobotTesting
> git --version # timeout=10
> git --version # 'git version 2.39.5'
> git fetch --tags --force --progress -- https://github.com/Taksanant/Lab7-RobotTesting +refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision ec7ff9827be2fc7b2a2d23a40bf4cf3cc35c680e (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f ec7ff9827be2fc7b2a2d23a40bf4cf3cc35c680e # timeout=10
Commit message: "Merge branch 'master'"
> git rev-list --no-walk ec7ff9827be2fc7b2a2d23a40bf4cf3cc35c680e # timeout=10
[UAT] $ /bin/sh -xe /tmp/jenkins459353721888978779.sh
+ robot webdriver
/tmp/jenkins459353721888978779.sh: 2: robot: not found
  
```

## Lab Worksheet



The screenshot displays the Jenkins web interface for a build named 'UAT' with ID '#13'. The 'Console Output' tab is selected, showing the execution of a shell script. The script performs a git checkout, configures sparse checkout, and attempts to run a robot test. The build fails at the robot test step because the 'robot' executable is not found. The console output is as follows:

```
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision ec7ff9827be2fc7b2a2d23a40bf4cf3cc35c680e (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f ec7ff9827be2fc7b2a2d23a40bf4cf3cc35c680e # timeout=10
Commit message: "Merge branch 'master'"
> git rev-list --no-walk ec7ff9827be2fc7b2a2d23a40bf4cf3cc35c680e # timeout=10
[UAT] $ /bin/sh -xe /tmp/jenkins459353721888978779.sh
+ robot webdriver
/tmp/jenkins459353721888978779.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
```

REST API Jenkins 2.479.3