

Lab Worksheet

ชื่อ-นามสกุล นายอัสนี อินทรประเสริฐ อรหานักศึกษา 653380354-7 Section 3

Lab#8 – Software Deployment Using Docker**วัตถุประสงค์การเรียนรู้**

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

Pre-requisite

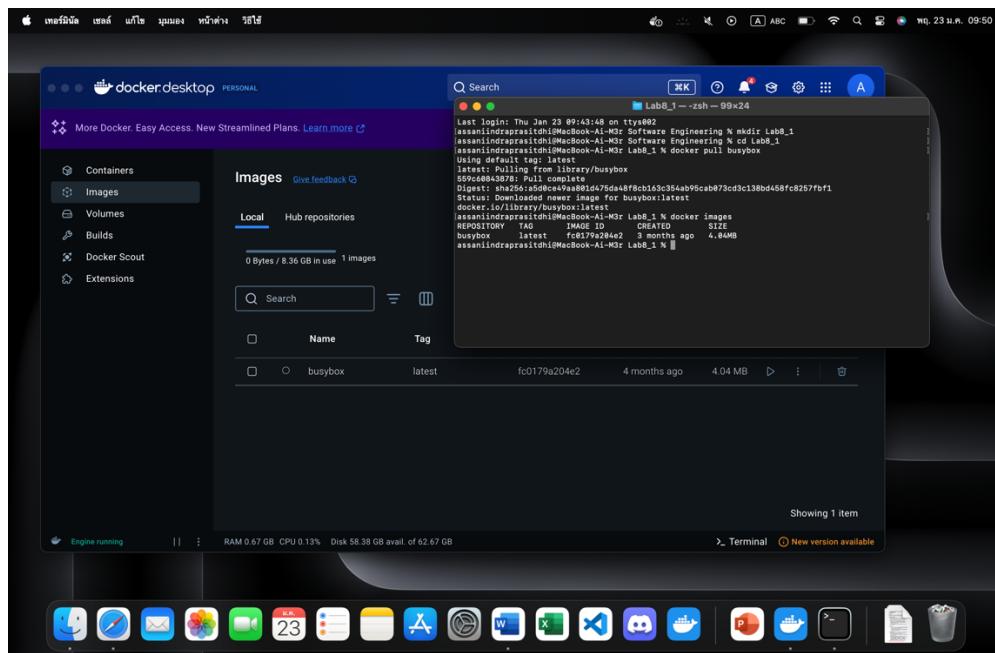
1. ติดตั้ง Docker desktop ลงบนเครื่องคอมพิวเตอร์ โดยดาวน์โหลดจาก <https://www.docker.com/get-started>
2. สร้าง Account บน Docker hub (<https://hub.docker.com/signup>)
3. กำหนดให้ \$ หมายถึง Command prompt และ <> หมายถึง ให้ป้อนค่าของพารามิเตอร์ที่กำหนด

แบบฝึกปฏิบัติที่ 8.1 Hello world - รัน Container จาก Docker image

1. เปิดใช้งาน Docker desktop และ Login ด้วย Username และ Password ที่ลงทะเบียนกับ Docker Hub เอาไว้
1. เปิด Command line หรือ Terminal บน Docker Desktop จากนั้นสร้าง Directory ชื่อ Lab8_1
2. ย้ายตำแหน่งปัจจุบันไปที่ Lab8_1 เพื่อใช้เป็น Working directory
3. ป้อนคำสั่ง \$ docker pull busybox หรือ \$ sudo docker pull busybox สำหรับกรณีที่ติดปัญหา Permission denied
(หมายเหตุ: BusyBox เป็น software suite ที่รองรับคำสั่งบางอย่างบน Unix - <https://busybox.net>)
4. ป้อนคำสั่ง \$ docker images

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

Lab Worksheet



- (1) สิ่งที่อยู่ภายใต้คอลัมน์ Repository คืออะไร
ตอบ Image ที่มีชื่อว่า busybox
- (2) Tag ที่ใช้บ่งบอกถึงอะไร
ตอบ Version ของตัว Image ในที่นี่คือ Version ล่าสุด

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) และแสดงผลลัพธ์ที่ได้ตั้งแต่ขั้นตอนที่ 6-12 พร้อมกับตอบคำถามต่อไปนี้

Lab Worksheet

```
Last login: Thu Jan 23 09:43:48 on ttys002
assaninindrapsitdhi@MacBook-A1-M3r: ~ Software Engineering % mkdir Lab8_1
assaninindrapsitdhi@MacBook-A1-M3r: ~ Software Engineering % cd Lab8_1
assaninindrapsitdhi@MacBook-A1-M3r: ~ Lab8_1 % docker pull busybox
Using default tag: latest
latest: Pulling from library/busybox
559c69843878: Pull complete
Digest: sha256:a5d9c0e49a0891a475da48f8cb163c354ab95cab073cd3c138bd458fc8257fbf1
Status: Downloaded newer image for busybox:latest
docker.io/library/busybox:latest
assaninindrapsitdhi@MacBook-A1-M3r: ~ Lab8_1 % docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
busybox latest fc0179a28ae2 3 months ago 4.04MB
assaninindrapsitdhi@MacBook-A1-M3r: ~ Lab8_1 % docker run busybox
assaninindrapsitdhi@MacBook-A1-M3r: ~ Lab8_1 % docker run -it busybox sh
/ # ls
bin dev etc home lib lib64 proc root sys tmp usr var
/ # ls -la
total 48
drwxr-xr-x 1 root root 4096 Jan 23 02:52 .
drwxr-xr-x 1 root root 4096 Jan 23 02:52 ..
-rwxr-xr-x 1 root root 0 Jan 23 02:52 .dockercfg
drwxr-xr-x 2 root root 12288 Sep 26 21:31 bin
drwxr-xr-x 5 root root 368 Jan 23 02:52 dev
drwxr-xr-x 1 root root 4096 Jan 23 02:52 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 215 root root 0 Jan 23 02:52 proc
drwxr-xr-x 1 root root 4096 Jan 23 02:52 root
dr-xr-xr-x 11 root root 0 Jan 23 02:52 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
assaninindrapsitdhi@MacBook-A1-M3r: ~ Lab8_1 % docker run busybox echo "Hello Assani Indraprasitdhi from busybox"
Hello Assani Indraprasitdhi from busybox
assaninindrapsitdhi@MacBook-A1-M3r: ~ Lab8_1 % docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
cdefa7e0b761 busybox "echo 'Hello Assani ...'" 11 seconds ago Exited (0) 10 seconds ago mystifying_maxwell
0ce52ba2399c busybox "sh" About a minute ago Exited (0) 49 seconds ago loving_matsumoto
64f1acfdic15 busybox "sh" 2 minutes ago Exited (0) 2 minutes ago inspiring_hermann
assaninindrapsitdhi@MacBook-A1-M3r: ~ Lab8_1 %
```

- (1) เมื่อใช้ option -it ในคำสั่ง run ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสั้นๆ เช่น
ตอบ -i ทำให้สามารถพิมพ์คำสั่งใน container ได้ ส่วน -t ทำให้หน้าจอเมื่อൺการใช้งาน terminal โดยตรง
- (2) คอลัมน์ STATUS จากการรันคำสั่ง docker ps -a แสดงถึงข้อมูลอะไร
ตอบ และแสดงสถานะปัจจุบันของแต่ละ Docker container และข้อมูลเกี่ยวกับเวลาที่สถานะนั้นเกิดขึ้นหรือถูกอัปเดต ทำให้รู้ว่า container นั้นกำลังทำงานหรือหยุดทำงานเมื่อใด

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

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

Lab Worksheet

```

Last login: Thu Jan 23 09:43:48 on ttys002
assaninindrprasitdhi@MacBook-Ai-M3r: Software Engineering % mkdir Lab8_1
assaninindrprasitdhi@MacBook-Ai-M3r: Software Engineering % cd Lab8_1
assaninindrprasitdhi@MacBook-Ai-M3r: Lab8_1 % docker pull busybox
Using default tag: latest
latest: Pulling from library/busybox
559c60843878: Pull complete
Digest: sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138bd458fc8257fbff
Status: Downloaded newer image for busybox:latest
docker.io/library/busybox:latest
assaninindrprasitdhi@MacBook-Ai-M3r: Lab8_1 % docker run busybox
assaninindrprasitdhi@MacBook-Ai-M3r: 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 4896 Jan 23 02:52 .
drwxr-xr-x 1 root root 4896 Jan 23 02:52 ..
-rwxr-xr-x 1 root root 0 Jan 23 02:52 .dockerenv
drwxr-xr-x 2 root root 12288 Sep 26 21:31 bin
drwxr-xr-x 5 root root 369 Jan 23 02:52 dev
drwxr-xr-x 1 root root 4896 Jan 23 02:52 etc
drwxr-xr-x 2 nobody nobody 4896 Sep 26 21:31 home
drwxr-xr-x 2 root root 4896 Sep 26 21:31 lib
lrwxrwxrwx 1 root root 3 Sep 26 21:31 lib64 -> lib
dr-xr-xr-x 215 root root 0 Jan 23 02:52 proc
drwxr-xr-x 1 root root 4896 Jan 23 02:52 root
dr-xr-xr-x 11 root root 0 Jan 23 02:52 sys
drwxrwxwt 2 root root 4896 Sep 26 21:31 tmp
drwxr-xr-x 4 root root 4896 Sep 26 21:31 usr
drwxr-xr-x 4 root root 4896 Sep 26 21:31 var
assaninindrprasitdhi@MacBook-Ai-M3r: Lab8_1 % docker run busybox echo "Hello Assani Indraprasitdhi from busybox"
Hello Assani Indraprasitdhi from busybox
assaninindrprasitdhi@MacBook-Ai-M3r: Lab8_1 % docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
cdfea7e0b761 busybox "echo 'Hello Assani ...'" 11 seconds ago Exited (0) 10 seconds ago mystifying_maxwell
0ce52ba2399c busybox "sh" About a minute ago Exited (0) 49 seconds ago loving_matsumoto
64f1acfd1c15 busybox "sh" 2 minutes ago Exited (0) 2 minutes ago inspiring_hermann
assaninindrprasitdhi@MacBook-Ai-M3r: Lab8_1 % docker rm cdfea7e0b761
cdfea7e0b761
assaninindrprasitdhi@MacBook-Ai-M3r: Lab8_1 % docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
0ce52ba2399c busybox "sh" 10 minutes ago Exited (0) 9 minutes ago loving_matsumoto
64f1acfd1c15 busybox "sh" 11 minutes ago Exited (0) 11 minutes ago inspiring_hermann
assaninindrprasitdhi@MacBook-Ai-M3r: Lab8_1 %

```

แบบฝึกปฏิบัติที่ 8.2: สร้าง Docker file และ Docker image

- เปิดใช้งาน Docker desktop และ Login ด้วย Username และ Password ที่ลงทะเบียนกับ Docker Hub เอาไว้
- เปิด Command line หรือ Terminal จากนั้นสร้าง Directory ชื่อ Lab8_2
- ย้ายตำแหน่งปัจจุบันไปที่ Lab8_2 เพื่อใช้เป็น Working directory
- สร้าง Dockerfile.swp ไว้ใน Working directory

สำหรับเครื่องที่ใช้ระบบปฏิบัติการวินโดว์ส (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.”

Lab Worksheet

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) และแสดงผลลัพธ์ที่ได้ในขันตอนที่ 5

พร้อมกัน เตือนความต่อไปนี้

The screenshot shows a Mac desktop with several open windows. In the center, the Docker Desktop application is running, displaying a sidebar with options like Containers, Images, Volumes, Builds, Docker Scout, and Extensions. The 'Images' tab is selected, showing a list of available images: 'busybox' (latest version, sha256:fc0179a204e2) and 'dockerfile' (latest version, sha256:80edb90bed59). Below the list, a message indicates 3 warnings found when using the --debug option. On the right side of the Docker window, there's a preview of a Dockerfile for a 'busybox' image. At the bottom of the Docker window, it says 'Showing 2 items'. To the right of the Docker window, a terminal window titled 'Lab8_2 -- zsh - 145x35' is open, showing a session where a Dockerfile for a 'busybox' image is being built. The terminal output includes several warning messages related to JSON arguments and multiple CMD instructions. The status bar at the bottom of the screen shows system information like RAM usage (0.52 GB), CPU usage (0.00%), and disk availability (58.37 GB of 62.67 GB). A notification in the top right corner indicates a 'New version available' for Docker Desktop.

- (1) คำสั่งที่ใช้ในการ run คือ docker run dockerfile
 - (2) Option -t ในคำสั่ง \$ docker build ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสั้นๆ

Lab Worksheet

ตอบ กำหนดชื่อ (tag) ให้กับ Docker image ที่ถูกสร้างขึ้นมา ทำให้การจัดการและอ้างอิง image ง่าย
ขึ้นในภายหลัง

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

สำหรับเครื่องที่ใช้ระบบปฏิบัติการwinโดว์ 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

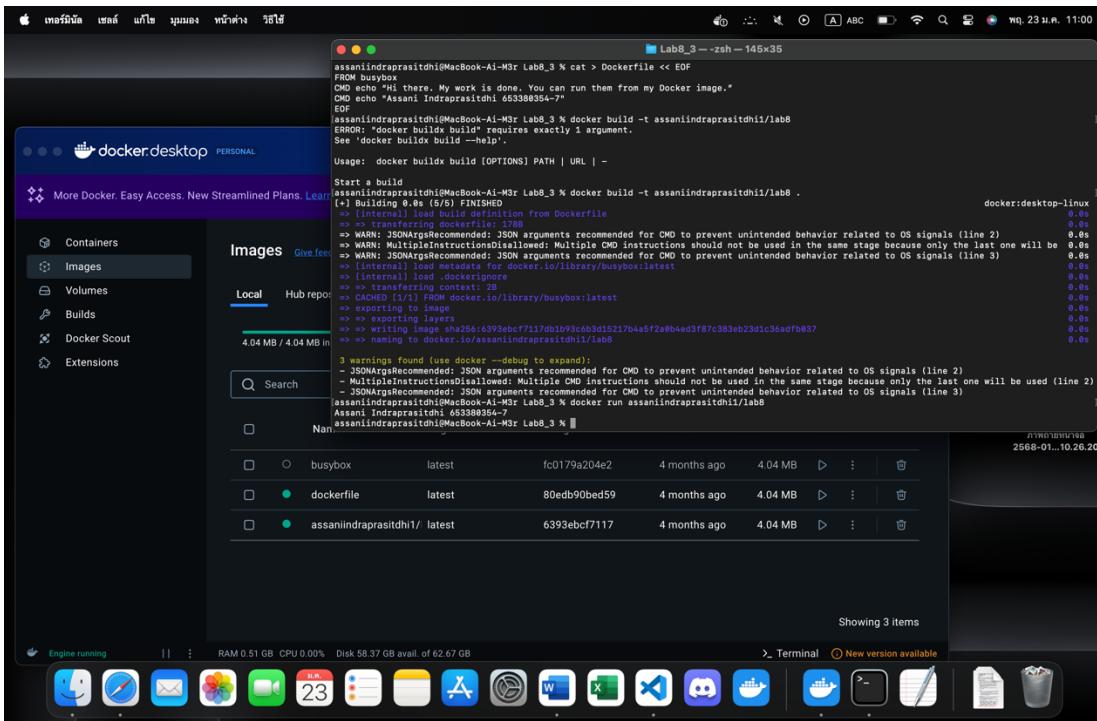
```
$ docker build -t <username ที่ลงทะเบียนกับ Docker Hub>/lab8
```

5. ทำการรัน Docker image บน Container ในเครื่องของตัวเองเพื่อทดสอบผลลัพธ์ ด้วยคำสั่ง

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

[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

ในการนี้ที่ติดปัญหาไม่ได้ Login ไว้ก่อน ให้ใช้คำสั่งต่อไปนี้ เพื่อ Login ก่อนทำการ Push

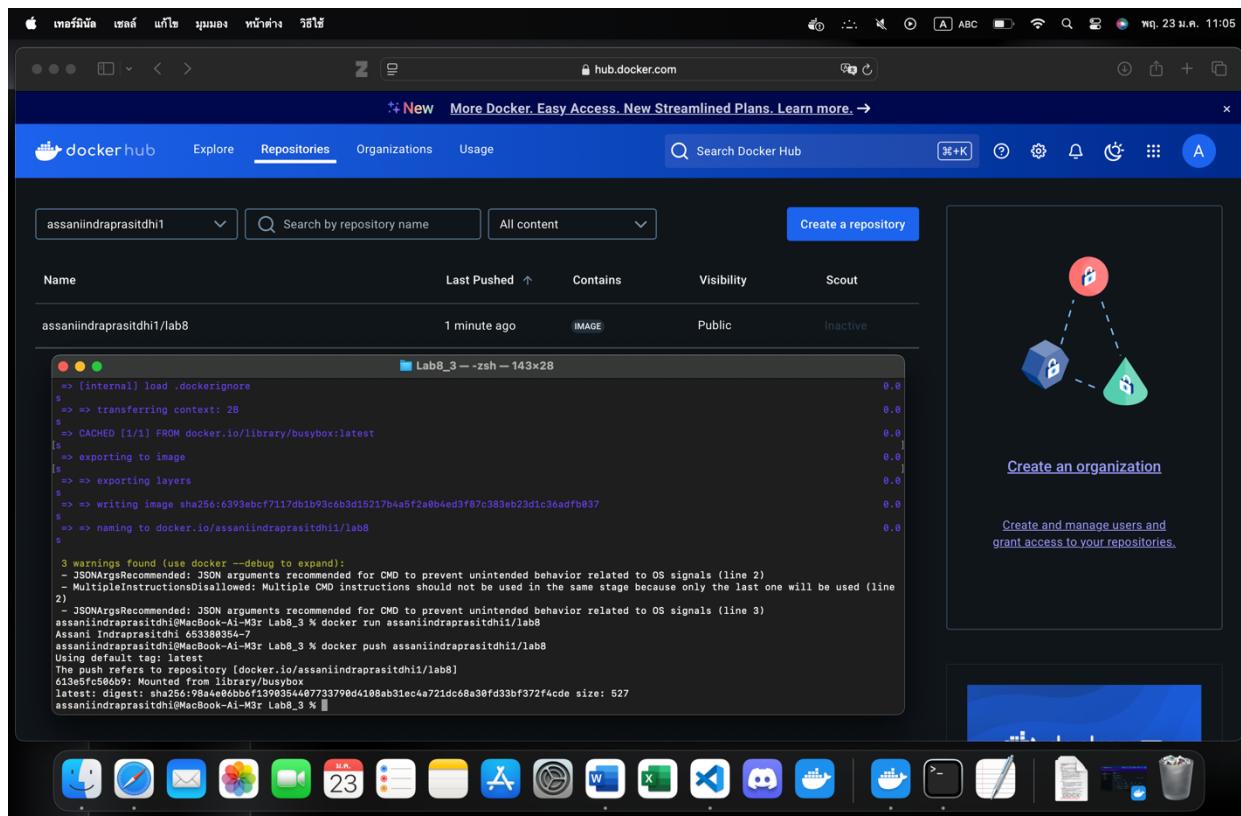
\$ docker login และป้อน Username และ Password ตามที่ระบุใน Command prompt หรือใช้คำสั่ง

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

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

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

Lab Worksheet



แบบฝึกปฏิบัติที่ 8.4: การ Build และ Update แอปพลิเคชันจาก Container image

1. เปิด Command line หรือ Terminal จากนั้นสร้าง Directory ชื่อ Lab8_4
2. ทำการ Clone ซอฟต์แวร์ docker/getting-started ลงใน Directory ที่สร้างขึ้น โดยใช้คำสั่ง

<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

```

{
  "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": "^2.3.0",
    "sqlite3": "^5.1.2",
    "uuid": "9.0.0",
    "wait-port": "^1.0.4"
  },
  "resolutions": {
    "ansi-regex": "5.0.1"
  },
  "prettier": {
    "trailingComma": "all",
    "tabWidth": 4,
    "useTabs": false,
    "semi": true,
    "singleQuote": true
  },
  "devDependencies": {
    "jest": "^29.3.1",
    "nodemon": "^2.0.20",
    "prettier": "^2.7.1"
  }
}

```

4. ภายใต้ getting-started/app ให้สร้าง Dockerfile พร้อมกับใส่เนื้อหาดังต่อไปนี้ลงไว้ในไฟล์

```

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_รหัส
ศ. ไม่มีขีด

\$ docker build -t <myapp_รหัสศ. ไม่มีขีด> .

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

Lab Worksheet

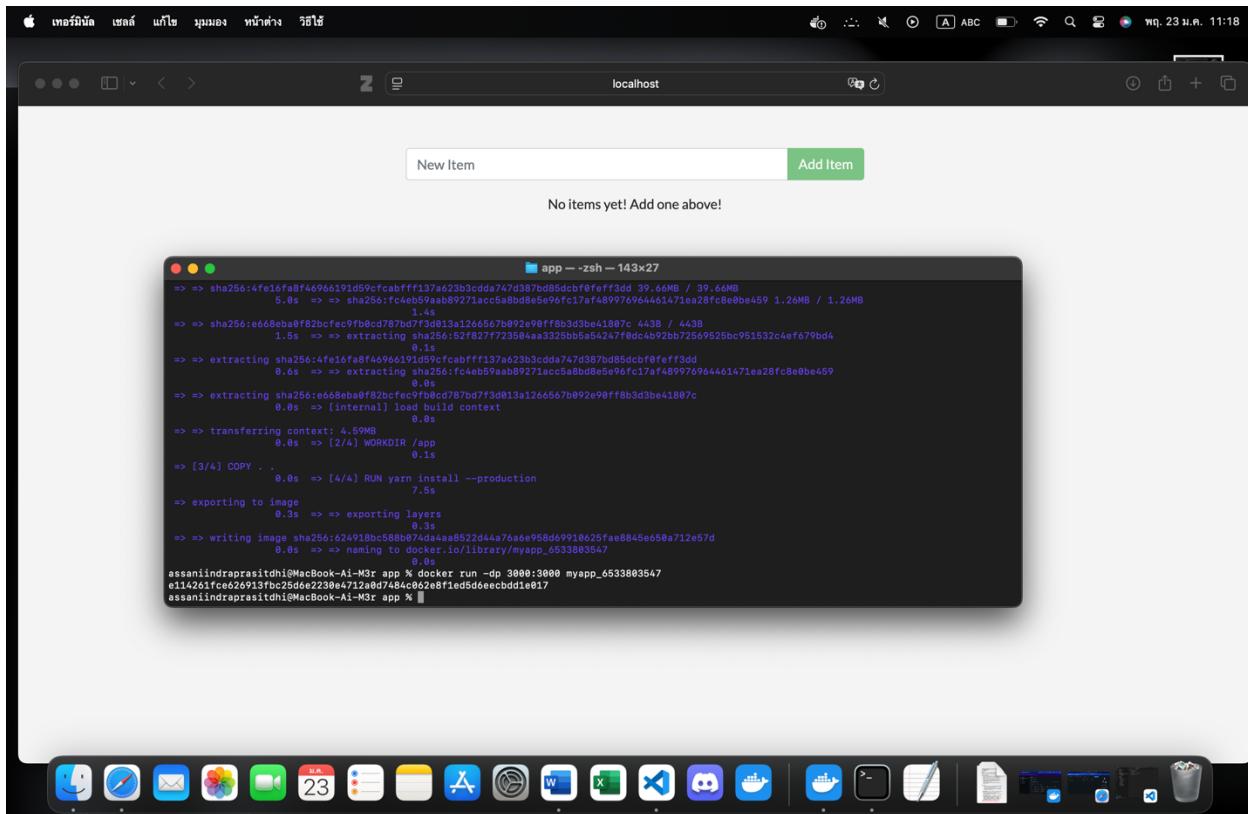
- ## 6. ทำการ Start ตัว Container ของแอปพลิเคชันที่สร้างขึ้น โดยใช้คำสั่ง

```
§ docker run -dp 3000:3000 <myapp> รหัสนศ. ไม่มีปิด>
```

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

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

9. ทำการ Build Docker image โดยใช้คำสั่งเดียวกันกับข้อ 5

10. Start และรัน Container ตัวใหม่ โดยใช้คำสั่งเดียวกันกับข้อ 6

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

Lab Worksheet

```

assaniindrapsasitdh@MacBook-A1-M3r:~/Desktop/getting-started$ docker build -t myapp_6533803547 .
[+] Building 9.9s (1/10) FINISHED
--> [internal] load build definition from dockerfile
--> [internal] load metadata for docker.io/library/node:18-alpine
--> [internal] load token for registry-1.docker.io
--> [internal] load context: 8.01kB
--> CACHED [2/4] WORKDIR /app
--> [3/4] COPY .
--> [4/4] RUN yarn install --production
--> exporting image
--> exporting layers
--> writing image sha256:d8966cc3b8c343a2ba844d78077ad3c3fb555b0cbe80ee47b2d1d04844f8b2
--> naming to docker.io/library/myapp_6533803547
assaniindrapsasitdh@MacBook-A1-M3r:~/Desktop/getting-started$ docker run myapp_6533803547
Using sqlite database at /etc/todos/todo.db
Listening on port 3000

```

(1) Error ที่เกิดขึ้นหมายความอย่างไร และเกิดขึ้นเพราะอะไร

ตอบ ข้อผิดพลาดนี้เกิดจาก port 3000 เพราะว่า port 3000 ถูกใช้งานใน Container ก่อนหน้านี้แล้ว

11. ลบ Container ของ Web application เวอร์ชันก่อนแก้ไขออกจากระบบ โดยใช้รูปไดร์ฟที่แนบมา

a. ผ่าน Command line interface

- ใช้คำสั่ง `$ docker ps` เพื่อดู Container ID ที่ต้องการจะลบ
- Copy หรือบันทึก Container ID ไว้
- ใช้คำสั่ง `$ docker stop <Container ID>` ที่ต้องการจะลบ เพื่อยุดการทำงานของ Container ดังกล่าว
- ใช้คำสั่ง `$ docker rm <Container ID>` ที่ต้องการจะลบ เพื่อทำการลบ

b. ผ่าน Docker desktop

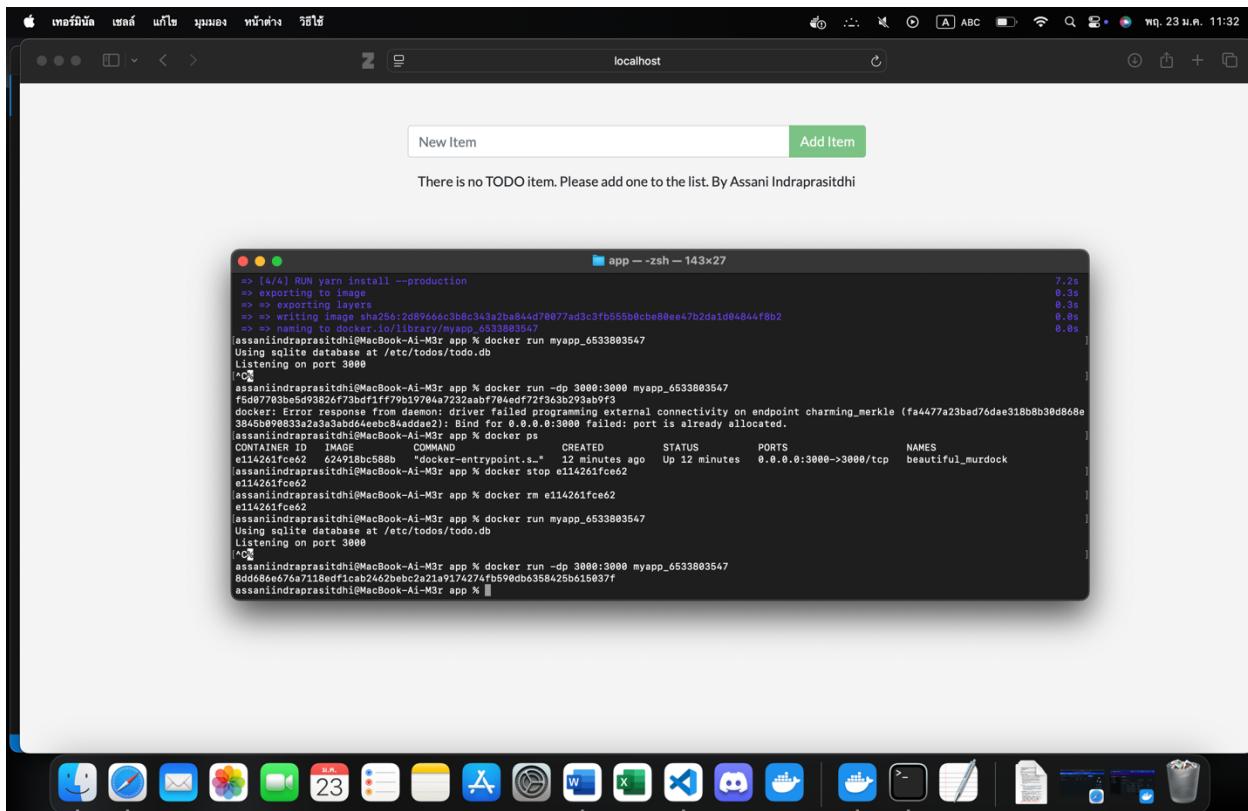
- ไปที่หน้าต่าง Containers
- เลือกไอคอนถังขยะในແກ່ງของ Container ที่ต้องการจะลบ
- ยืนยันโดยการกด Delete forever

12. Start และรัน Container ตัวใหม่อีกรัง โดยใช้คำสั่งเดียวกันกับข้อ 6

Lab Worksheet

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

[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:lts-jdk17
```

หรือ

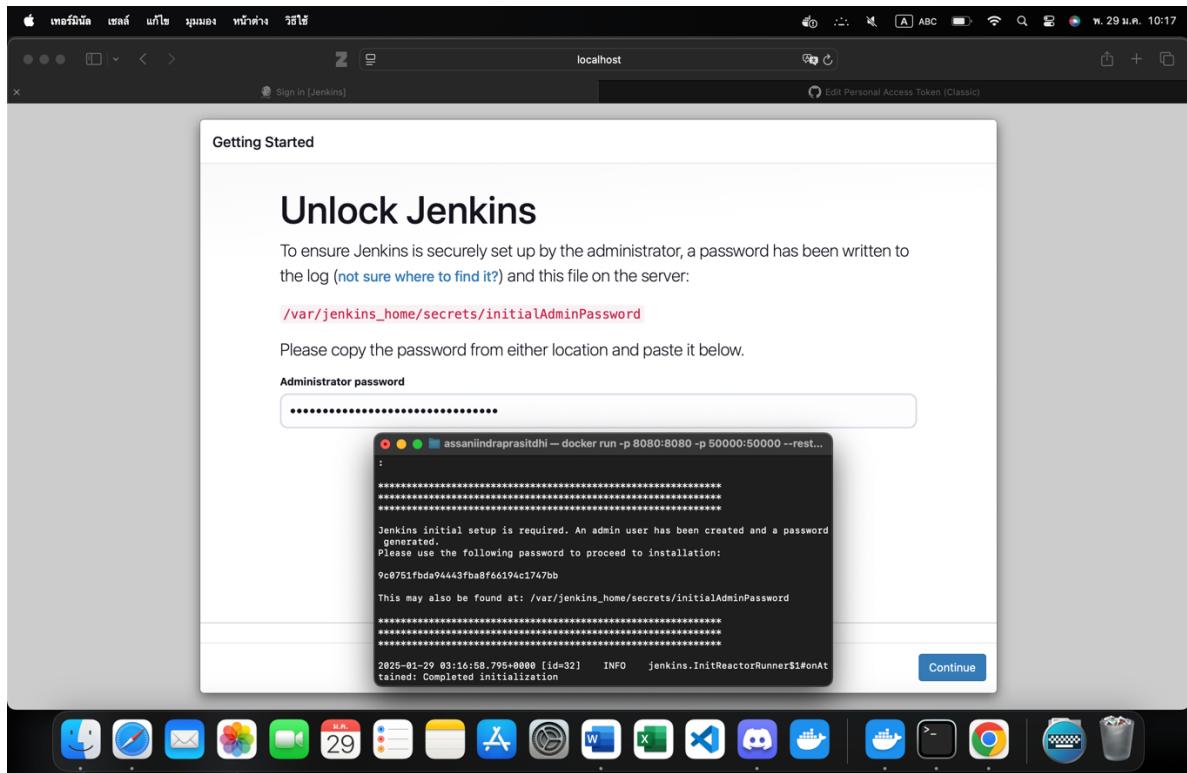
```
$ docker run -p 8080:8080 -p 50000:50000 --restart=on-failure -v
```

```
jenkins_home:/var/jenkins_home jenkins/jenkins:lts-jdk17
```

3. บันทึกรหัสผ่านของ Admin user ไว้สำหรับ log-in ในครั้งแรก

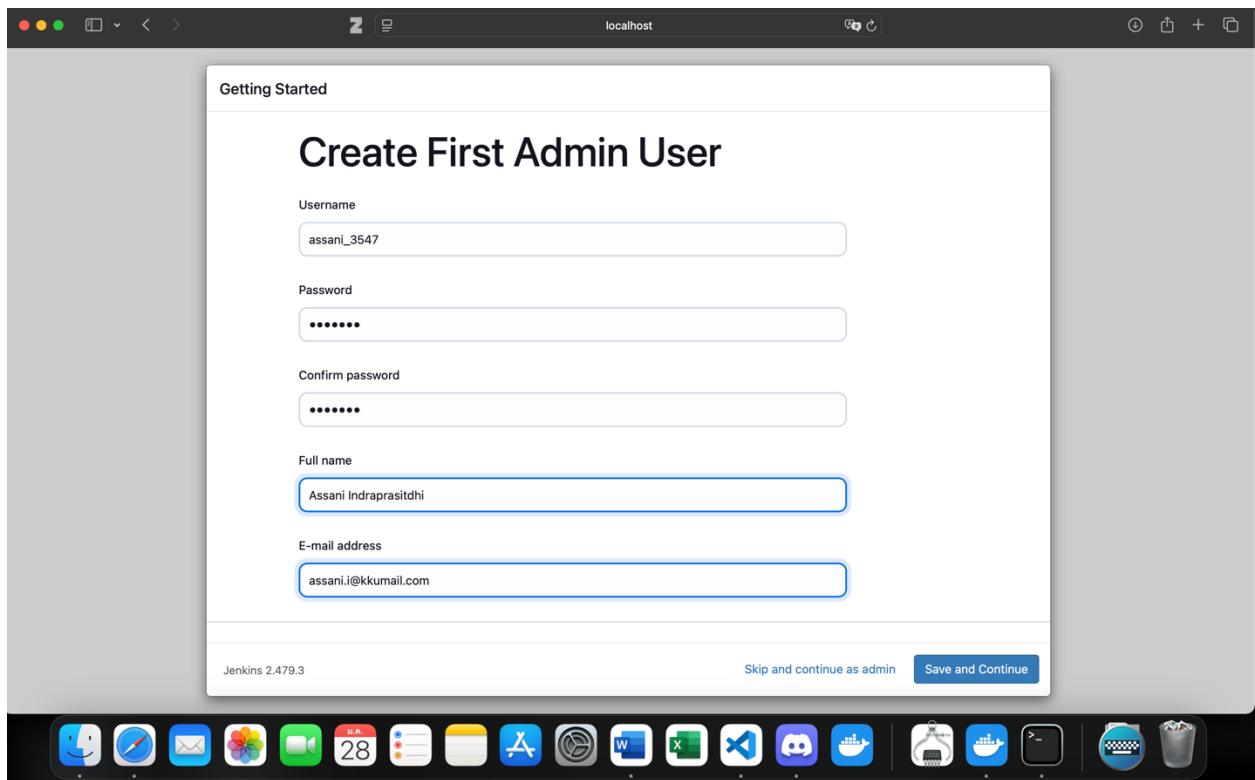
[Check point#12] Capture หน้าจอที่แสดงผล Admin password

Lab Worksheet



4. เมื่อได้รับการยืนยันว่า Jenkins is fully up and running ให้เปิดบราวเซอร์ และป้อนที่อยู่เป็น localhost:8080
5. ทำการ Unlock Jenkins ด้วยรหัสผ่านที่ได้ในข้อที่ 3
6. สร้าง Admin User โดยใช้ username เป็นชื่อจริงของนักศึกษาพร้อมรหัสสี่ตัวท้าย เช่น somsri_3062
[Check point#13] Capture หน้าจอที่แสดงผลการตั้งค่า

Lab Worksheet



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

Lab Worksheet**9. เลือก Manage Jenkins แล้วไปที่เมนู Plugins**

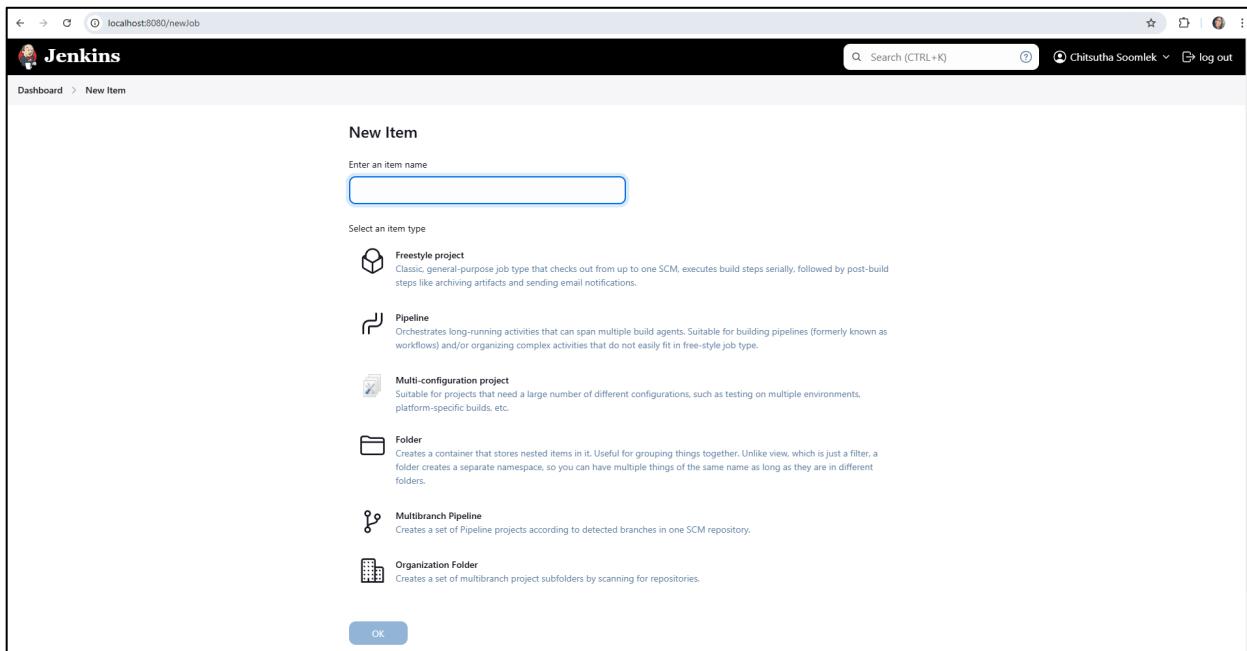
The screenshot shows the Jenkins Manage Jenkins interface. The left sidebar has 'Manage Jenkins' selected. The main area is titled 'Manage Jenkins' and contains several sections: 'System Configuration' (Build Queue, System, Tools, Clouds, Appearance, Plugins, Nodes), 'Security' (Security, Credentials, Credential Providers, Users), and 'Status Information' (System Information, System Log, Load Statistics, About Jenkins). A message at the top right says 'It appears that your reverse proxy set up is broken.' A search bar at the top right says 'Search (CTRL+K)'.

10. ไปที่เมนู Available plugins แล้วเลือกติดตั้ง Robotframework เพิ่มเติม

The screenshot shows the Jenkins Plugin Manager under the 'Available plugins' tab. A search bar at the top right shows 'robot'. A single result is listed: 'Robot Framework 5.0.0' by 'Build Reports'. The result card includes a description: 'This publisher stores Robot Framework test reports for builds and shows summaries of them in project and build views along with trend graph.', a release date of '2 mo 7 days ago', and an 'Install' button.

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

Lab Worksheet



12. นำไฟล์ .robot ที่ทำให้แบบฝึกปฏิบัติที่ 7 (Lab#7) ไปวิ่งบน Repository ของนักศึกษา จนนั่นตั้งค่าที่
จำเป็นในหน้านี้ทั้งหมด ดังนี้

Description: Lab 8.5

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

Build Trigger: เลือกแบบ Build periodically และกำหนดให้ build ทุก 15 นาที

Build Steps: เลือก Execute shell และใส่คำสั่งในการรันไฟล์ .robot (หากไฟล์ไม่ได้อยู่ในหน้าแรกของ repository ให้ใส่ Path ไปถึงไฟล์ให้เรียบร้อยด้วย)

Lab Worksheet

[Check point#14] Capture หน้าจอแสดงการตั้งค่า พร้อมกับตอบคำถามต่อไปนี้

General Enabled

Description
Lab 8.5

Plain text [Preview](#)

Discard old builds ?

GitHub project

This project is parameterized ?

Throttle builds ?

Execute concurrent builds if necessary ?

Advanced ▼

None

Git ?

Repositories ?

Repository URL ?
https://ghp_8uiDFzuzdvV9rMTjmx5l0hOrqqf3dy3K4gJ2@github.com/653380354-7/UAT.git

Credentials ?
653380354-7/*********

+ Add

Advanced ▼

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?
*/main

Add Branch

Save **Apply**

Lab Worksheet

Build Triggers

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

Build after other projects are built ?

Build periodically ?

Schedule ?

H/15 * * * *

Would last have run at Wednesday, January 29, 2025 at 7:05:13 AM Coordinated Universal Time; would next run at Wednesday, January 29, 2025 at 7:20:13 AM Coordinated Universal Time.

GitHub hook trigger for GITScm polling ?

Poll SCM ?

Build Steps

Execute shell ?

Command

See the list of available environment variables

```
. /var/jenkins_home/robot_env/bin/activate
export PATH=$PATH:/usr/bin/chromedriver
mkdir -p results
robot --outputdir results test.robot
```

Advanced ▾

Add build step ▾

Lab Worksheet

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)
results

Advanced Edited

Thresholds for build result ?

🟡 %
20.0

🔵 %
80.0

DEPRECATED! THIS FLAG DOES NOTHING! - Use thresholds for critical tests only
 Include skipped tests in total count for thresholds

Add post-build action ▾

Save **Apply**

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

```
-> ./var/jenkins_home/robot_env/bin/activate
export PATH=$PATH:/usr/bin/chromedriver
mkdir -p results
robot --outputdir results test.robot
```

Post-build action: เพิ่ม Publish Robot Framework test results -> ระบุไดเร็คทอรีที่เก็บไฟล์ผลการทดสอบโดย Robot framework ในรูป xml และ html -> ตั้งค่า Threshold เป็น % ของการทดสอบที่ไม่ผ่านแล้วนับว่าซอฟต์แวร์มีปัญหา -> ตั้งค่า Threshold เป็น % ของการทดสอบที่ผ่านแล้วนับว่าซอฟต์แวร์มีอุปปัญหในสถานะที่สามารถนำไปใช้งานได้ (เช่น 20, 80)

13. กด Apply และ Save
14. สร้าง Build Now

Lab Worksheet

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

The screenshot shows the Jenkins UAT pipeline dashboard. On the left, there's a sidebar with options like Status, Changes, Workspace, Build Now, Configure, Delete Project, Robot Results, and Rename. The main area has tabs for Status (selected), Changes, and UAT. Under Status, it says "UAT 8.5". Below that is a chart titled "Robot Framework Tests Trend (all tests)" showing 1 test case: 0 Failed, 1 Passed, and 0 Skipped. To the right of the chart is a "Builds" section listing recent builds from 29 มกราคม 2568, including #19, #18, #17, #16, #15, #14, and #13. Each build has a green checkmark indicating success. At the bottom of the dashboard, there's a "Latest Robot Results" summary with a total of 1 Failed test case, 0 Passed, 1 Skipped, and 100.0% Pass %.

The screenshot shows the Jenkins UAT console output for build #19. The sidebar on the left includes Status, Changes, Console Output (selected), Edit Build Information, Delete build '#19', Timings, Git Build Data, Robot Results, and Previous Build. The main area is titled "Console Output" and displays the terminal logs for the build. The logs show the Jenkins environment starting by timer, cloning the repository from GitHub, fetching upstream changes, and executing the test script. The output ends with the command "sh /tmp/jenkins1296098163421866581.sh" which activates the robot environment.

```

Started by timer
Running as SYSTEM
Building in workspace /var/jenkins_home/workspace/UAT
The recommended git tool is: NONE
using credential 435b6ed-672c-44af-a9d7-22d9d18c33f7
> 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://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@github.com/653380354-7/UAT.git #
timeout=10
Fetching upstream changes from https://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@github.com/653380354-7/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://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@github.com/653380354-7/UAT.git +refs/heads/* refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision 241d017ba5c908e67cb955f09620216e11608339 (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f 241d017ba5c908e67cb955f09620216e11608339 # timeout=10
Commit message: "Add test.robot"
> git rev-list --no-walk 241d017ba5c908e67cb955f09620216e11608339 # timeout=10
[UAT] $ /bin/sh -xe /tmp/jenkins1296098163421866581.sh
+ . /var/jenkins_home/robot_env/bin/activate
+ deactivate nondestructive
+ [ -n ]
+ [ -n ]
+ [ -n -o -n ]
+ [ -n ]

```