

## Lab Worksheet

ชื่อ-นามสกุล นายอนันดา มาตราช รหัสนักศึกษา 653380350-5 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

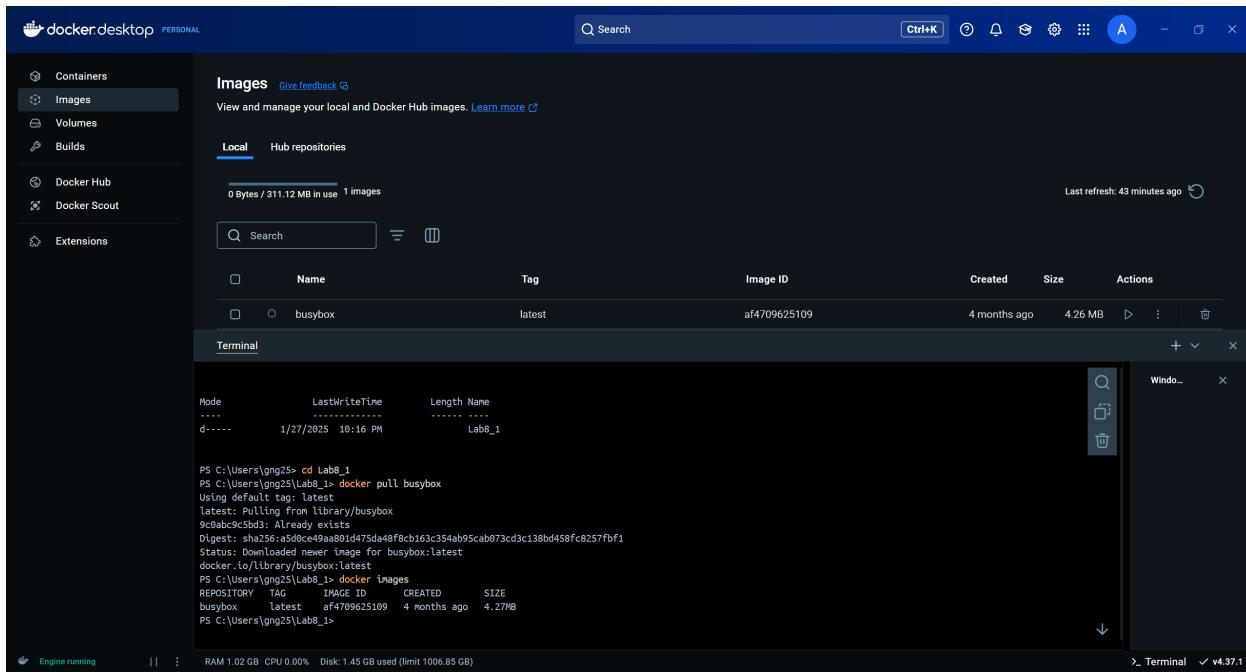
1. ติดตั้ง Docker desktop ลงบนเครื่องคอมพิวเตอร์ โดยดาวน์โหลดจาก <https://www.docker.com/get-started>
2. สร้าง Account บน Docker hub (<https://hub.docker.com/signup>)
3. กำหนดให้ \$ หมายถึง Command prompt และ <> หมายถึง ให้ป้อนค่าของพารามิเตอร์ที่กำหนด

#### 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

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

## Lab Worksheet



(1) สิ่งที่อยู่ภายใต้คอลัมน์ Repository คืออะไร ชื่อของ software suite คือ busybox ที่ถูก Pull ผ่านคำสั่ง docker pull busybox

(2) Tag ที่ใช้บ่งบอกถึงอะไร บ่งบอกถึงเวอร์ชันของ images ที่มีอยู่ ชื่อกายในภาพเป็น image ของ software suite อย่าง busybox ที่เป็นเวอร์ชันล่าสุด (latest)

5. ป้อนคำสั่ง \$ docker run busybox
6. ป้อนคำสั่ง \$ docker run -it busybox sh
7. ป้อนคำสั่ง ls
8. ป้อนคำสั่ง ls -la
9. ป้อนคำสั่ง exit
10. ป้อนคำสั่ง \$ docker run busybox echo "Hello ชื่อและนามสกุลของนักศึกษา from busybox"
11. ป้อนคำสั่ง \$ docker ps -a

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

## Lab Worksheet

```

PS C:\Users\gng25\Lab8_1> docker run busybox
PS C:\Users\gng25\Lab8_1> docker run -it busybox sh
/ # ls -la
bin dev etc home lib lib64 proc root sys tmp usr var
/ # ls -la
total 48
drwxr-xr-x 1 root root 4096 Jan 27 15:25 .
drwxr-xr-x 1 root root 4096 Jan 27 15:25 ..
-rwxr-xr-x 1 root root 0 Jan 27 15:25 .dockerenv
drwxr-xr-x 2 root root 12288 Sep 26 21:31 bin
drwxr-xr-x 5 root root 360 Jan 27 15:25 dev
drwxr-xr-x 1 root root 4096 Jan 27 15:25 etc
drwxr-xr-x 2 nobody nobody 4096 Sep 26 21:31 home
drwxr-xr-x 3 root root 4096 Sep 26 21:31 lib
drwxrwxrwx 1 root root 3 Sep 26 21:31 lib64 -> lib
dr-xr-xr-x 272 root root 0 Jan 27 15:25 proc
drwx----- 1 root root 4096 Jan 27 15:25 root
dr-xr-xr-x 11 root root 0 Jan 27 15:25 sys
drwxrwxrwt 2 root root 4096 Sep 26 21:31 tmp
drwxr-xr-x 4 root root 4096 Sep 26 21:31 usr
drwxr-xr-x 4 root root 4096 Sep 26 21:31 var
/ # exit
PS C:\Users\gng25\Lab8_1>

PS C:\Users\gng25\Lab8_1> docker run busybox echo "Hello Ananda Matarach from busybox"
Hello Ananda Matarach from busybox
PS C:\Users\gng25\Lab8_1> docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
05a3093bb0d7 busybox "echo 'Hello Ananda .." 10 seconds ago Exited (0) 9 seconds ago affectionate_turing
db0c7db429e8 busybox "sh" 41 seconds ago Exited (0) 27 seconds ago bold_bell
2969306ba15c busybox "sh" 46 seconds ago Exited (0) 46 seconds ago elastic_carver
PS C:\Users\gng25\Lab8_1>

```

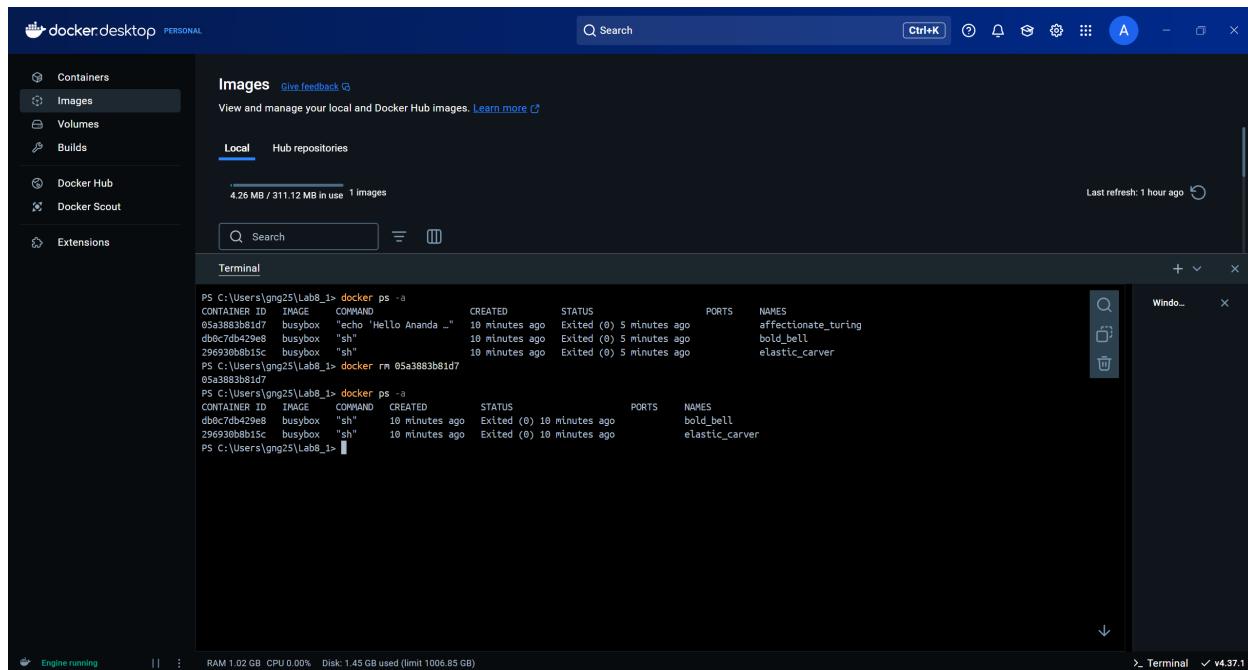
- (1) เมื่อใช้ option -it ในคำสั่ง run ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป  
option -it คือสองคำสั่งที่รวมกัน ใช้สำหรับเปิด Terminal ใน Container และสามารถพิมพ์คำสั่งได้แบบทันที จากคำสั่ง -i คือ เปิดโหมดอินเทอร์แอคทีฟ และ -t คือ สร้าง terminal จำลอง (pseudo-TTY)
- (2) คอลัมน์ STATUS จากการรันคำสั่ง docker ps -a แสดงถึงข้อมูลอะไร

## Lab Worksheet

แสดงข้อมูลของ Container ID (ไอเดียของ Container), Image (ชื่อของ Image), Command (คำสั่งที่ใช้), Created (เวลาที่สร้าง), Status (สถานการณ์ทำงาน), Ports (Ports ที่กำลังใช้งาน), Names (ชื่อของ Container)

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

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



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

สำหรับเครื่องที่ใช้ระบบปฏิบัติการวินโดวส์ (Windows) บันทึกคำสั่งต่อไปลงในไฟล์ โดยใช้ Text Editor ที่มี

FROM busybox

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

CMD echo "ชื่อ-นามสกุล รหัสนักศึกษา ชื่อเล่น"

## Lab Worksheet

สำหรับเครื่องที่ใช้ระบบปฏิบัติการ MacOS หรือ Linux บนหน้าต่าง Terminal และป้อนคำสั่งต่อไปนี้

```
$ cat > Dockerfile << EOF
FROM busybox
CMD echo "Hi there. This is my first docker image."
CMD echo "ชื่อ-นามสกุล รหัสนักศึกษา ชื่อเล่น"
EOF
```

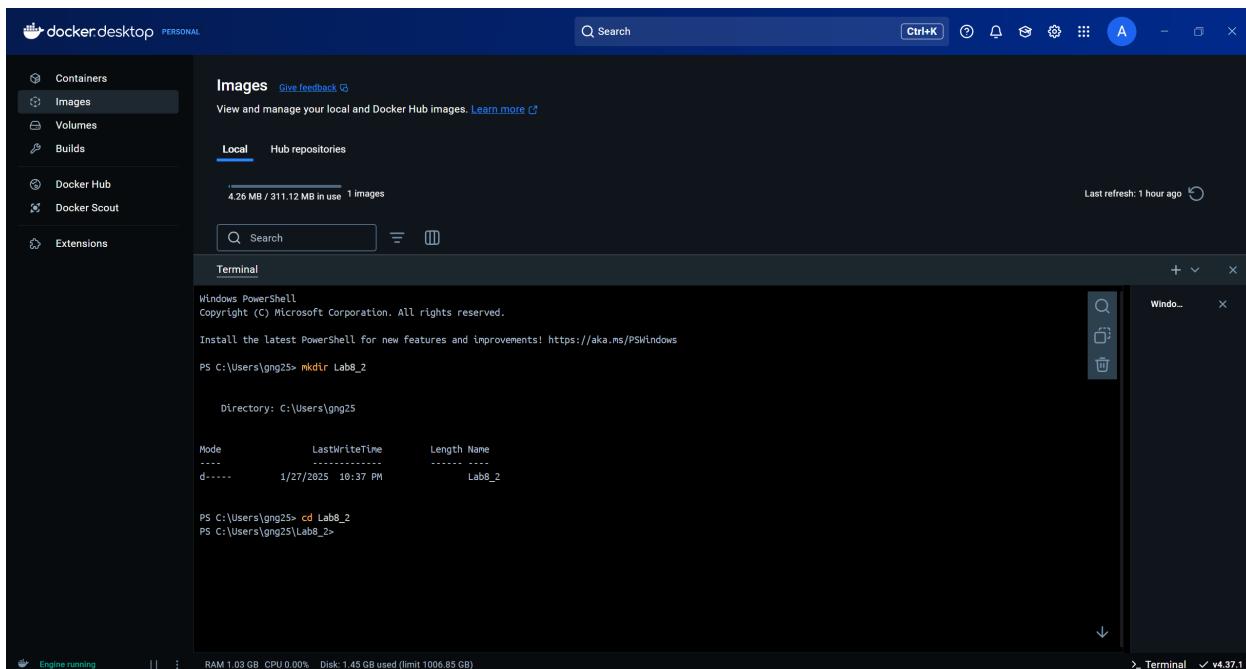
หรือใช้คำสั่ง

```
$ touch Dockerfile
```

แล้วใช้ Text Editor ในการใส่เนื้อหาแทน

5. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้
- \$ docker build -t <ชื่อ Image> .
6. เมื่อ Build สำเร็จแล้ว ให้ทำการรัน Docker image ที่สร้างขึ้นในขั้นตอนที่ 5

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



## Lab Worksheet

The screenshot shows the Docker Desktop application running on a Windows desktop. The top half of the screen displays a code editor with a Dockerfile named 'Dockerfile':

```
C: > Users > gng25 > Lab8_2 > Dockerfile > ...
1 FROM busybox
2 CMD echo "Hi there. This is my first docker image."
3 CMD echo "Ananda Matarach 653380350-5 Art"
```

The bottom half of the screen shows the Docker Desktop interface with the 'Images' tab selected. A terminal window is open at the bottom, showing the command:

```
docker build -t lab8_2build .
```

The terminal output shows the build process:

```
[+] Building 0.2s (5/5) FINISHED
--> [internal] load build definition from Dockerfile
--> => transferring dockerfile: 1kB
--> 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
--> => exporting layers
--> => writing image sha256:75165c97af8f054d2ad77adc07adcecd09988157535fe4f6ccffede298ab45cb
--> => naming to docker.io/lab8_2build
```

View build details: [docker://dashboard/build/desktop-linux/desktop-linux/nwreiee9pspnyw2kqvr579yn1](#)

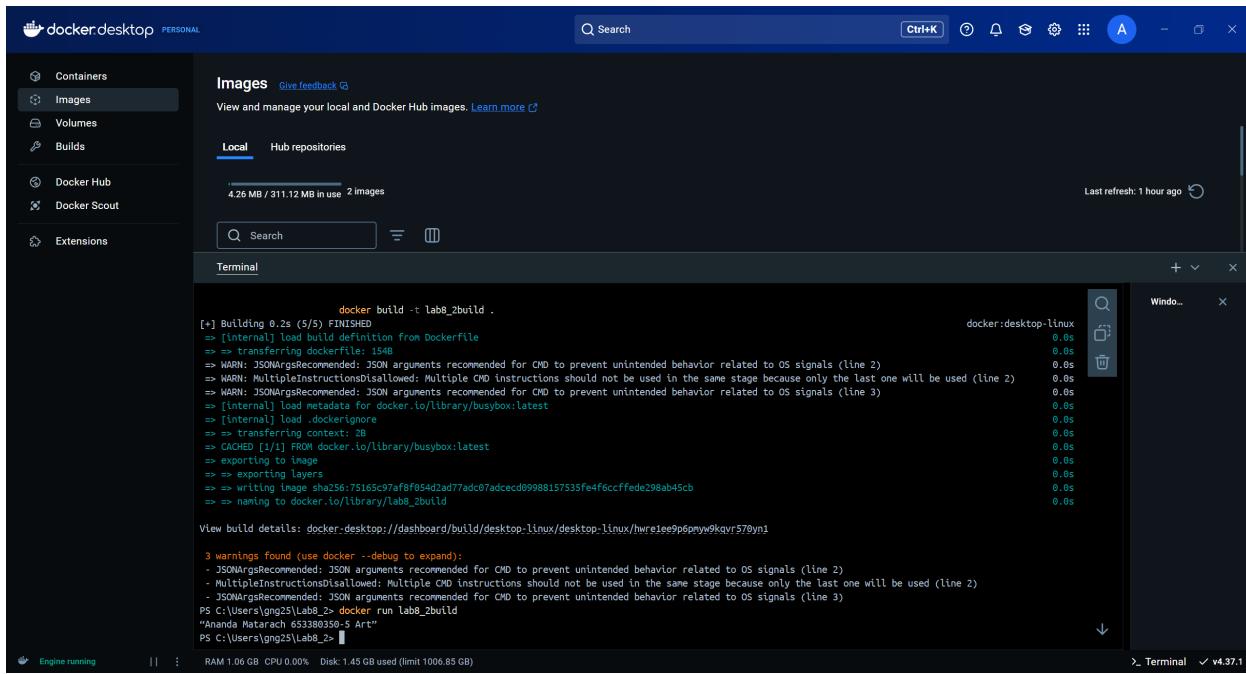
3 warnings found (use docker --debug to expand):

- 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)

PS C:\Users\gng25\Lab8\_2>

At the bottom of the interface, status information is displayed: Engine running, RAM 1.03 GB, CPU 0.00%, Disk 1.45 GB used (limit 1006.85 GB), Terminal v4.37.1.

## Lab Worksheet



(1) คำสั่งที่ใช้ในการ run คือ

`docker run lab8_2build`

(2) Option -t ในคำสั่ง \$ docker build ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสั้นๆ เช่น  
ใช้เพื่อกำหนด ชื่อ (Tag) ของ Docker Image ที่สร้างขึ้นมา

### แบบฝึกปฏิบัติที่ 8.3: การแชร์ Docker image ผ่าน Docker Hub

- เปิดใช้งาน Docker desktop และ Login ด้วย Username และ Password ที่ลงทะเบียนกับ Docker Hub เอ้าไว้
- เปิด Command line หรือ Terminal จากนั้นสร้าง Directory ชื่อ Lab8\_3
- ย้ายตัวแฟ้มง่ายๆไปที่ Lab8\_3 เพื่อใช้เป็น Working directory
- 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

## Lab Worksheet

```
$ 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
```

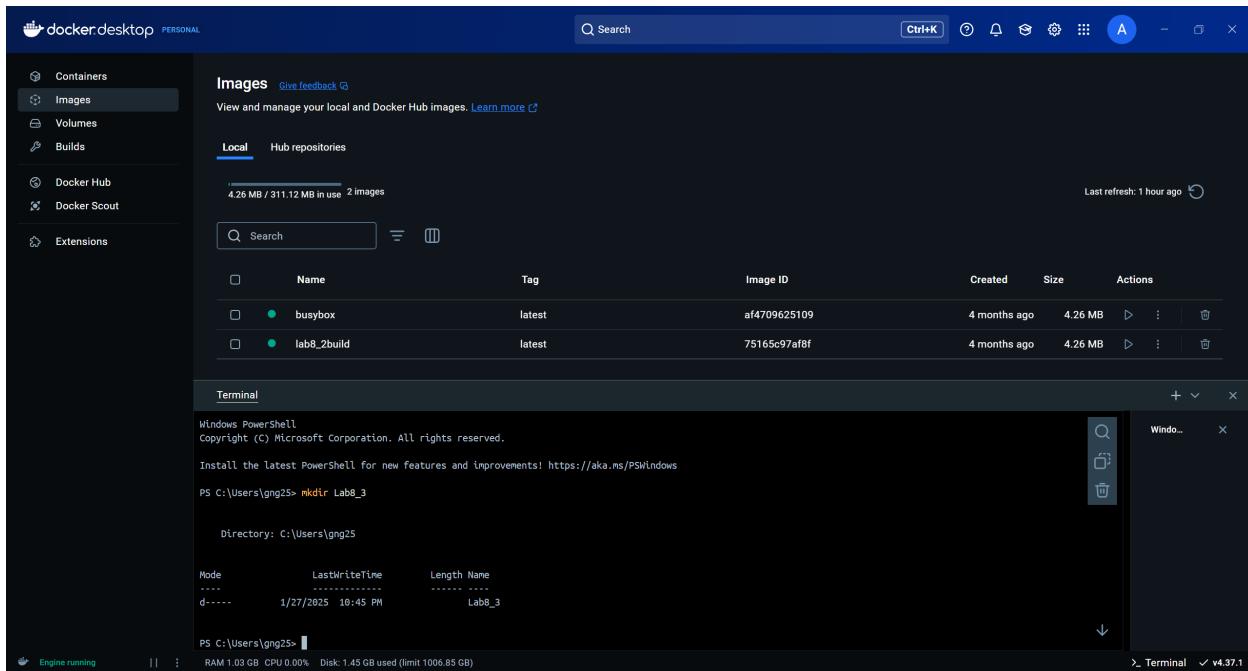
7. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้

```
$ 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

The screenshot shows the Docker Desktop application running on a Windows desktop. At the top, there's a dark-themed code editor window titled "Dockerfile 1" containing the following Dockerfile:

```
C: > Users > gng25 > Lab8_3 > Dockerfile > ...
1 FROM busybox
2 CMD echo "Hi there. My work is done. You can run them from my Docker image."
3 CMD echo "Ananda Matarach 653380350-5"
4
```

Below the code editor is a status bar showing "Ln 2, Col 50" and "UTF-8". To the right of the code editor are various icons for file operations like copy, paste, and search.

The main interface of Docker Desktop is visible, featuring a sidebar with options like Containers, Images, Volumes, Builds, Docker Hub, Docker Scout, and Extensions. The "Images" tab is selected, showing a list of local images:

Name	Tag	Image ID	Created	Size	Actions
busybox	latest	af4709625109	4 months ago	4.26 MB	<span>...</span>
lab8_2build	latest	75165c97af8f	4 months ago	4.26 MB	<span>...</span>
ar3an/lab8	latest	9a6f4e7d263a	4 months ago	4.26 MB	<span>...</span>

A terminal window is open at the bottom, showing the command-line history and some build logs. The logs include:

```
>> [internal] load .dockerignore
=> => transferring context: 2B
=> CACHED [1/1] FROM docker.io/library/busybox:latest
=> exporting to Image
=> exporting layers
=> => writing image sha256:9aef4e7d263adedf47d068894d4bd7c59af0ab8fe944dbae6769a8e4b944932f
=> => naming to docker.io/ar3an/lab8

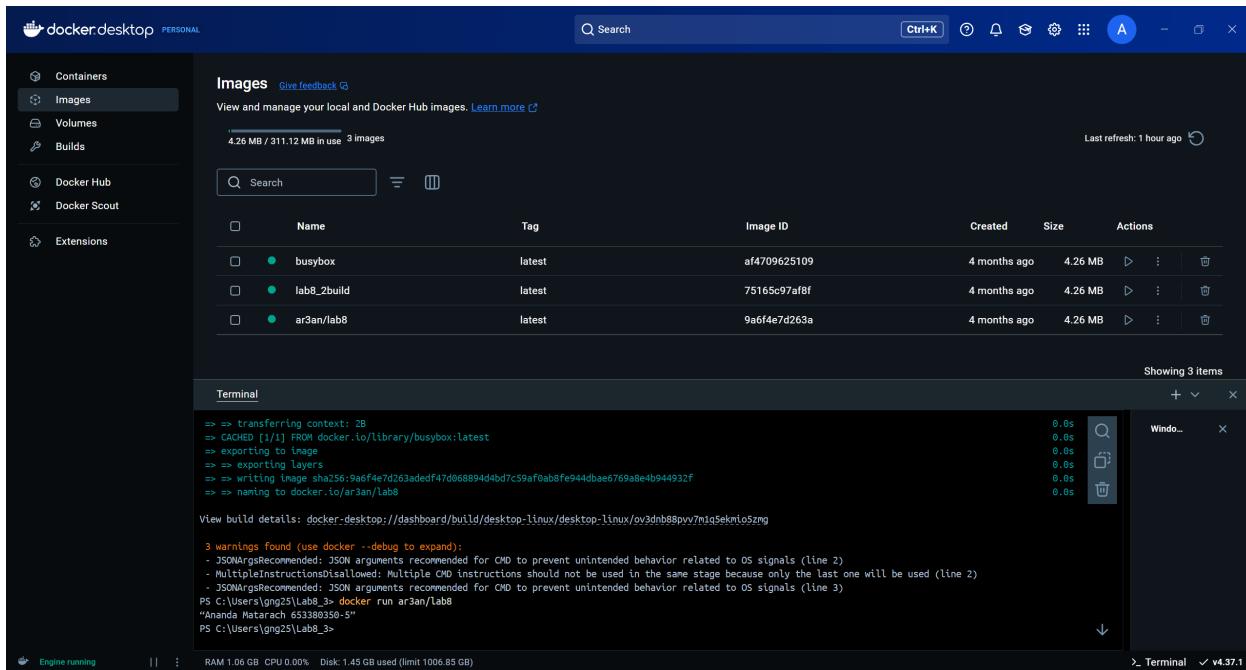
View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/ov3dhb88pvv7mq5eknioszmg

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)

PS C:\Users\gng25\Lab8_3>
```

The status bar at the bottom shows "Engine running", "RAM 1.05 GB CPU 0.75%", "Disk 1.45 GB used (limit 1006.85 GB)", and "Terminal v4.37.1".

## Lab Worksheet



6. ทำการ Push ตัว Docker image ไปเว็บ 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)

## Lab Worksheet

Docker Desktop interface showing search results for "ar3an" on Docker Hub. The results include four repositories: ar3an/lab8, ar3an/lab8, omar3anan/static\_website, and omar3anan/anan-website. A terminal window shows the command to push an image to Docker Hub. Below is a screenshot of a browser showing the Docker Hub repository page for ar3an/lab8, which has one tag named "latest".

## แบบฝึกปฏิบัติที่ 8.4: การ Build และ Update แอปพลิเคชันจาก Container image

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

## Lab Worksheet

```
$ git clone https://github.com/docker/getting-started.git
```

3. เปิดดูองค์ประกอบภายใน getting-started/app เมื่อพับไฟล์ package.json ให้ใช้ Text editor ในการเปิดอ่าน

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

```
PS C:\Users\gng25> cd Lab8_4
PS C:\Users\gng25\Lab8_4>
```

```
PS C:\Users\gng25> cd Lab8_4
PS C:\Users\gng25\Lab8_4> git clone https://github.com/docker/getting-started.git
Cloning into 'getting-started'...
remote: Enumerating objects: 100% (9/9), done.
remote: Counting objects: 100% (8/8), 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 | 17.73 MiB/s, done.
Resolving deltas: 100% (523/523), done.
PS C:\Users\gng25\Lab8_4>
```

## Lab Worksheet

```

1  {
2    "name": "101-app",
3    "version": "1.0.0",
4    "main": "index.js",
5    "license": "MIT",
6    "scripts": {
7      "prepretty": "prettier -l --write \"**/*.js\"",
8      "test": "jest",
9      "dev": "nodemon src/index.js"
10    },
11   "dependencies": {
12     "express": "^4.18.2",
13     "mysql2": "^2.3.3",
14     "sqlite3": "5.1.2",
15     "uuid": "9.0.0",
16     "wait-port": "^1.0.4"
17   },
18   "resolutions": {
19     "ansi-regex": "5.0.1"
20   },
21   "prettier": {
22     "trailingComma": "all",
23     "tabWidth": 4,
24     "useTabs": false,
25     "semi": true,
26     "singleQuote": true
27   },
28   "devDependencies": {
29     "jest": "^29.3.1",
30     "nodemon": "2.0.20",
31     "prettier": "2.7.1"
32   }
}

```

4. ภายใต้ getting-started/app ให้สร้าง Dockerfile พร้อมกับใส่เนื้อหาดังต่อไปนี้ลงในไฟล์

```

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.

ศ. ไม่มีชีด

\$ docker build -t <myapp\_รหัสนศ. ไม่มีชีด> .

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

## Lab Worksheet

The screenshot shows the Docker Desktop interface. At the top, a code editor displays a Dockerfile:

```

FROM node:18-alpine
WORKDIR /app
COPY .
RUN yarn install --production
CMD ["node", "src/index.js"]
EXPOSE 3000
    
```

Below the code editor is a sidebar with icons for Launchpad, Containers, Images, Volumes, Builds, Docker Hub, Docker Scout, and Extensions. The 'Images' tab is selected.

The main area shows the 'Images' section of the Docker Hub. It lists four local images with a total size of 4.26 MB. A terminal window is open at the bottom, showing the command-line output of a 'docker build' command:

```

PS C:\Users\gng25\Lab8_4\getting-started\app> docker build -t myapp_6533803505 .
[+] Building 3.6s (10/10) FINISHED
--> [internal] load build definition from Dockerfile
--> [internal] load metadata for docker.io/library/node:18-alpine
--> [auth] library/node:pull token for registry-1.docker.io
--> [internal] load .dockerignore
--> [internal] transfer context: 2B
--> [1/4] FROM docker.io/library/node:18-alpine@sha256:974afb6cbc0314dc6502b14243b8a39fb2d04d975e9059dd066be3e274fb25
--> [internal] load build context
--> [internal] transfer context: 4.62MB
--> [2/4] WORKDIR /app
--> [CACHED] [2/4] COPY .
--> [CACHED] [3/4] RUN yarn install --production
--> [CACHED] [4/4] EXPOSE 3000
--> exporting to image
--> exporting layers
--> writing image sha256:849345906b4a95364bb2e3540d49fb64f1dec20678524c5afc26f7113e972b9
--> naming to docker.io/library/myapp_6533803505

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/n4gcir9wqydznbeglnam86w
PS C:\Users\gng25\Lab8_4\getting-started\app>
    
```

The terminal also shows system resource usage: RAM 1.04 GB, CPU 0.1%, Disk 1.45 GB used (limit 1006.85 GB).

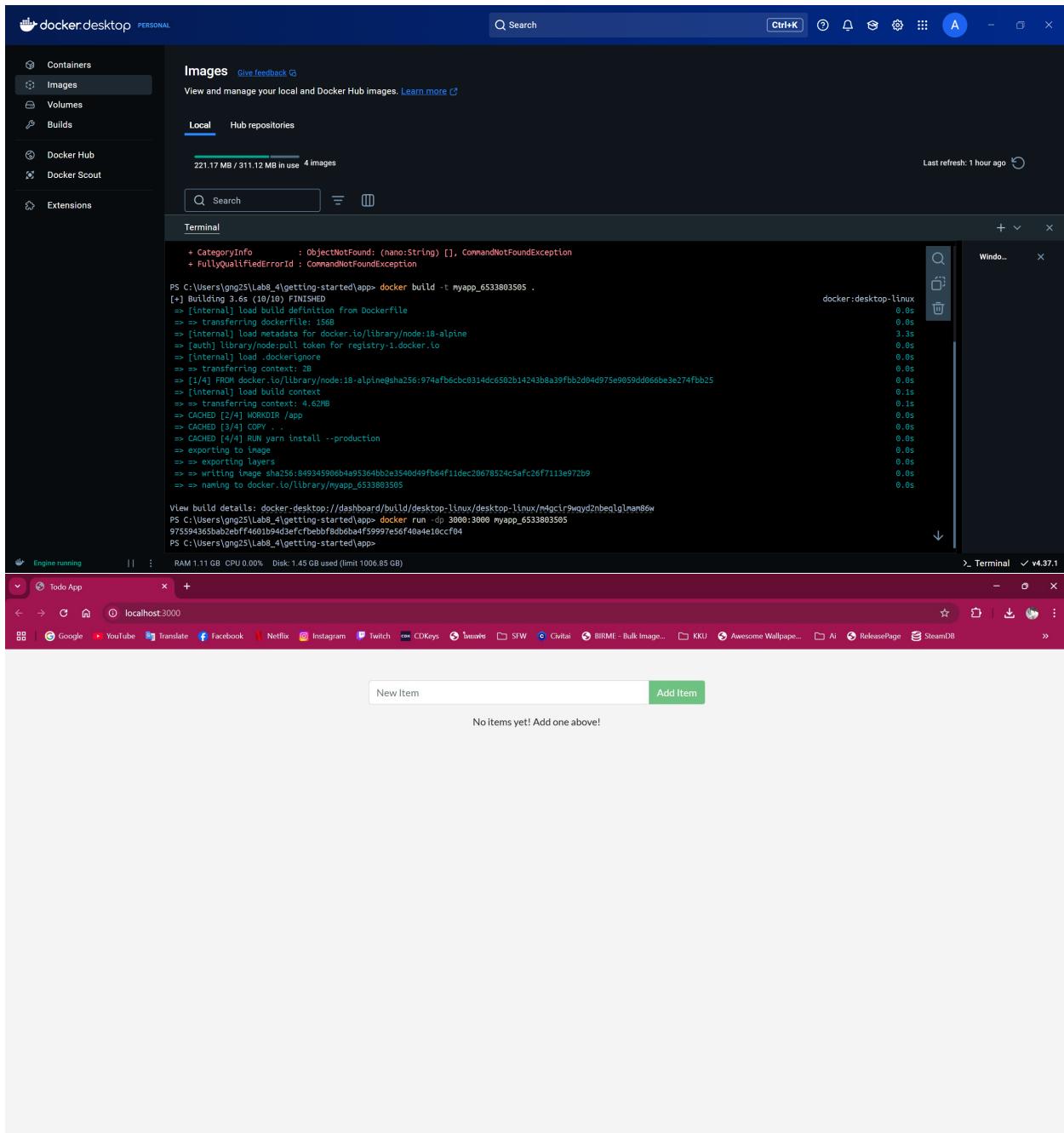
6. ทำการ Start ตัว Container ของแอปพลิเคชันที่สร้างขึ้น โดยใช้คำสั่ง

\$ docker run -dp 3000:3000 <myapp\_รหัสศ. ไม่มีขีด>

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

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

## Lab Worksheet



หมายเหตุ: นศ.สามารถทดลองเล่น Web application ที่ทำงานอยู่ได้

8. ทำการแก้ไข Source code ของ Web application ดังนี้

a. เปิดไฟล์ src/static/js/app.js ด้วย Editor และแก้ไขบรรทัดที่ 56 จาก

<p className="text-center">No items yet! Add one above!</p> เป็น

Lab Worksheet

<p className="text-center">There is no TODO item. Please add one to the list.  
By [ชื่อและนามสกุลของนักศึกษา](#)</p>

- b. Save ไฟล์ให้เรียบร้อย

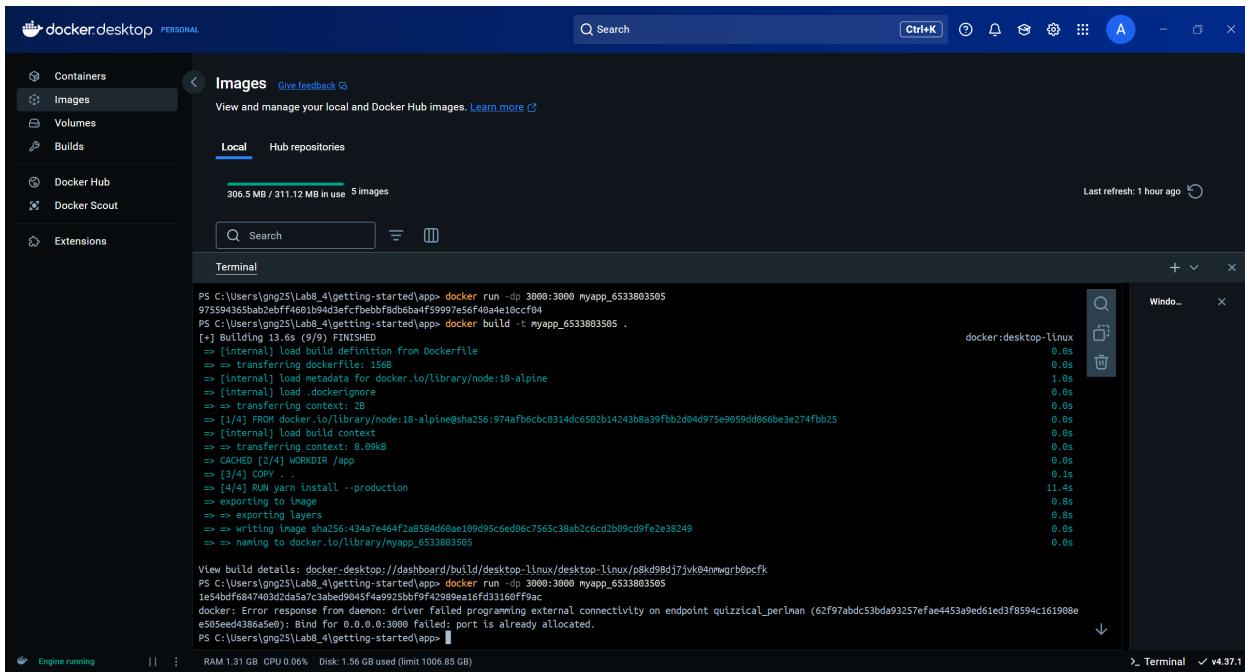
9. ทำการ Build Docker image โดยใช้คำสั่งเดียวกันกับข้อ 5

10. Start และรัน Container ตัวใหม่ โดยใช้คำสั่งเดียวกันกับข้อ 6

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

```
File Edit Selection View Go Run ... ← → Search
app.js ×
C:\Users\gng25>Lab8_4>getting-started>app>src>static>js> app.js > TodoListCard
14     function TodoListCard() {
15         ...
16
17         if (items === null) return 'Loading...';
18
19         return (
20             <React.Fragment>
21                 <AddItemForm onNewItem={onNewItem} />
22                 {items.length === 0 && (
23                     <p className="text-center">There is no TODO item. Please add one to the list. By Ananda Matarach</p>
24                     You, 1 second ago * Uncommitted change
25                 )}
26                 {items.map(item => (
27                     <ItemDisplay
28                         item={item}
29                         key={item.id}
30                         onItemUpdate={onItemUpdate}
31                         onItemRemoval={onItemRemoval}
32                     />
33                 )));
34             </React.Fragment>
35         );
36     }
37
38     function AddItemForm({ onNewItem }) {
39         const { Form, InputGroup, Button } = ReactBootstrap;
40
41         const [newItem, setNewItem] = React.useState('');
42         const [submitting, setSubmitting] = React.useState(false);
43
44         const submitNewItem = e => {
45             e.preventDefault();
46             setSubmitting(true);
47             fetch('/items', {
48                 method: 'POST',
49                 headers: {
50                     'Content-Type': 'application/json'
51                 },
52                 body: JSON.stringify({
53                     text: newItem
54                 })
55             })
56             .then(response => response.json())
57             .then(data => onNewItem(data))
58             .catch(error => console.error(error));
59         };
60
61         return (
62             <Form>
63                 <InputGroup>
64                     <Form.Control type="text" value={newItem} onChange={e => setNewItem(e.target.value)} />
65                     <Button onClick={submitNewItem}>Add</Button>
66                 </InputGroup>
67             </Form>
68         );
69     }
70
71     function ItemDisplay({ item, key, onItemUpdate, onItemRemoval }) {
72         const [isEditing, setIsEditing] = React.useState(false);
73
74         const startEdit = () => {
75             setIsEditing(true);
76         };
77
78         const cancelEdit = () => {
79             setIsEditing(false);
80             onItemUpdate(item);
81         };
82
83         const removeItem = () => {
84             onItemRemoval(item);
85         };
86
87         if (isEditing) {
88             return (
89                 <Form>
90                     <Form.Control type="text" value={item.text} onChange={e => onItemUpdate({ id: key, text: e.target.value })} />
91                     <Button onClick={cancelEdit}>Cancel</Button>
92                     <Button onClick={removeItem}>Delete</Button>
93                 </Form>
94             );
95         }
96
97         return (
98             <div>
99                 <div>{item.text}</div>
100                <div>Last updated: {item.updatedAt}</div>
101                <div>By {item.owner}</div>
102                <div><button onClick={startEdit}>Edit</button> <button onClick={removeItem}>Delete</button></div>
103            </div>
104        );
105    }
106
107    function App() {
108        const [items, setItems] = React.useState(null);
109
110        const onNewItem = item => {
111            setItems([item, ...items]);
112        };
113
114        const onItemUpdate = item => {
115            setItems(items.map(i => i.id === item.id ? item : i));
116        };
117
118        const onItemRemoval = id => {
119            setItems(items.filter(i => i.id !== id));
120        };
121
122        return (
123            <div>
124                <h1>My First Todo List</h1>
125                <h2>Todos</h2>
126                <TodoListCard onNewItem={onNewItem} onItemUpdate={onItemUpdate} onItemRemoval={onItemRemoval} />
127            </div>
128        );
129    }
130
131    ReactDOM.render(<App />, document.getElementById('root'));
132
```

## Lab Worksheet



(1) Error ที่เกิดขึ้นหมายความอย่างไร และเกิดขึ้นเพราะอะไร

Docker ไม่สามารถ bind พอร์ต 3000 บน 0.0.0.0 (ทุกอินเทอร์เฟซของเครื่อง) ได้ เกิดจาก พอร์ต 3000 ถูกใช้งานอยู่แล้ว โดยแอปพลิเคชันหรือ Container อื่น ซึ่งคือ build ก่อนหน้าที่รัน ทำให้ Docker ไม่สามารถจองพอร์ตนี้ได้

11. ลบ Container ของ Web application เวอร์ชันก่อนแก้ไขออกจากระบบ โดยใช้วิธีใดวิธีหนึ่งดังต่อไปนี้

a. ผ่าน Command line interface

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

b. ผ่าน Docker desktop

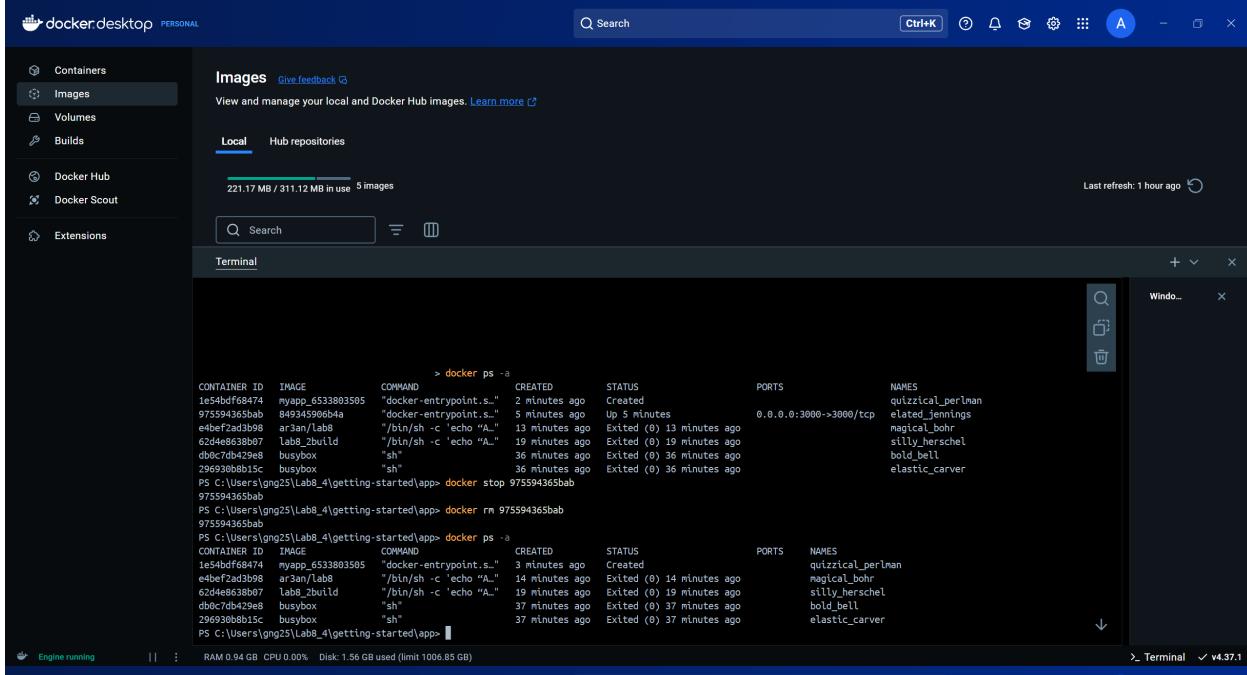
- ไปที่หน้าต่าง Containers
- เลือกไอคอนถังขยะในแท็บของ Container ที่ต้องการจะลบ
- ยืนยันโดยการกด Delete forever

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

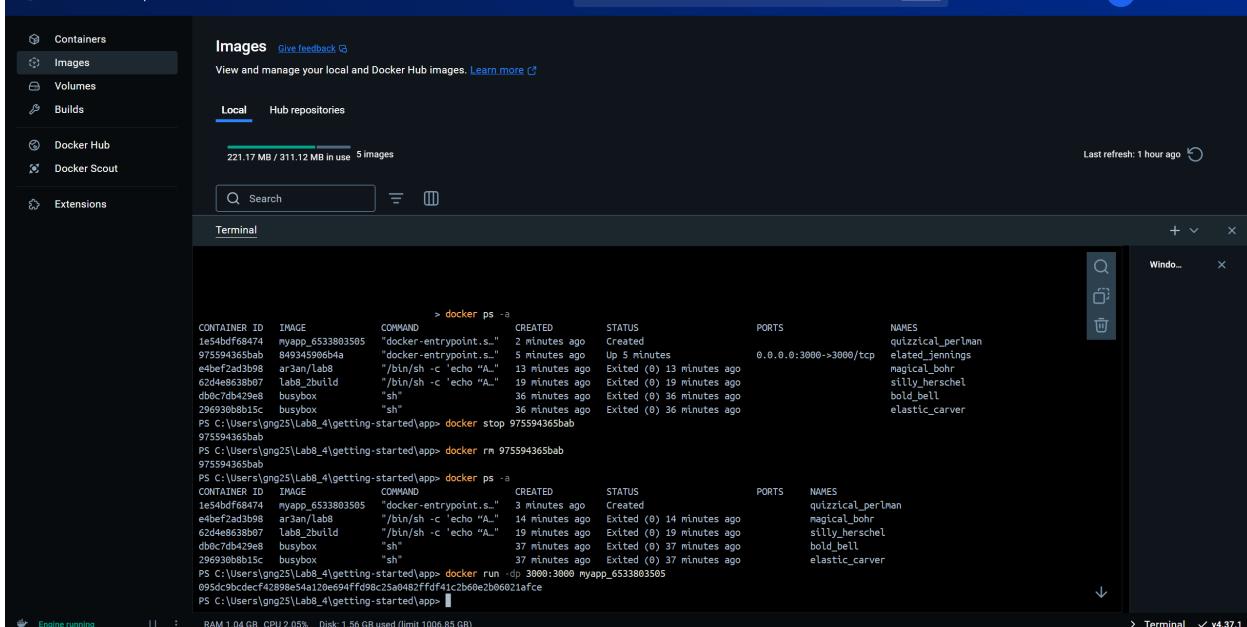


```

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
1e54bdf68474 myapp_6533803505 "docker-entrypoint.s..." 2 minutes ago Created
975594365bab 84934596b4a "docker-entrypoint.s..." 5 minutes ago Up 5 minutes 0.0.0.0:3000->3000/tcp elated_jennings
e4bef2ad3d98 arjan/lab8 "/bin/sh -c 'echo \"A...'" 13 minutes ago Exited (0) 13 minutes ago magical_bohr
62d4e6e38b07 lab8_2build "/bin/sh -c 'echo \"A...'" 19 minutes ago Exited (0) 19 minutes ago silly_herschel
db0c7db429eb busybox "sh" 36 minutes ago Exited (0) 36 minutes ago bold_bell
296930b8b15c busybox "sh" 36 minutes ago Exited (0) 36 minutes ago elastic_carver
PS C:\Users\ngn25\Lab8_4\getting-started> docker stop 975594365bab
975594365bab
PS C:\Users\ngn25\Lab8_4\getting-started> docker rm 975594365bab
975594365bab
PS C:\Users\ngn25\Lab8_4\getting-started> docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
1e54bdf68474 myapp_6533803505 "docker-entrypoint.s..." 3 minutes ago Created
e4bef2ad3d98 arjan/lab8 "/bin/sh -c 'echo \"A...'" 14 minutes ago Exited (0) 14 minutes ago magical_bohr
62d4e6e38b07 lab8_2build "/bin/sh -c 'echo \"A...'" 19 minutes ago Exited (0) 19 minutes ago silly_herschel
db0c7db429eb busybox "sh" 37 minutes ago Exited (0) 37 minutes ago bold_bell
296930b8b15c busybox "sh" 37 minutes ago Exited (0) 37 minutes ago elastic_carver
PS C:\Users\ngn25\Lab8_4\getting-started>

```

RAM 0.94 GB CPU 0.00% Disk 1.56 GB used (limit 1006.85 GB) > Terminal v 4.37.1

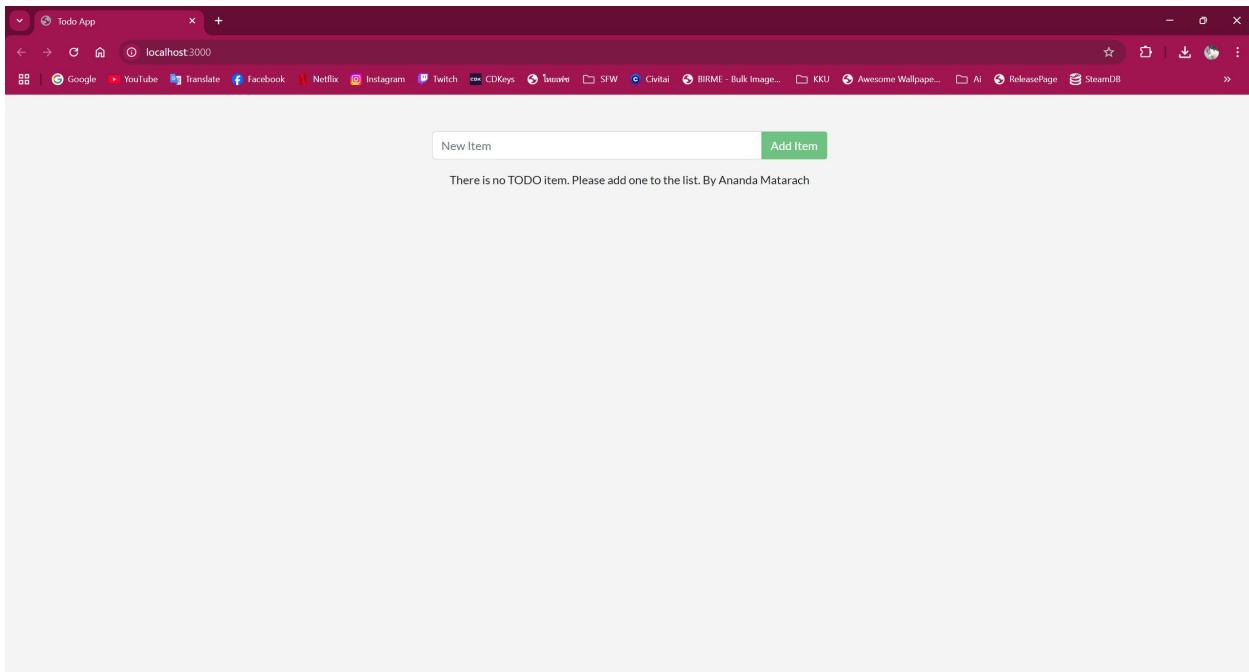
```

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
1e54bdf68474 myapp_6533803505 "docker-entrypoint.s..." 2 minutes ago Created
975594365bab 84934596b4a "docker-entrypoint.s..." 5 minutes ago Up 5 minutes 0.0.0.0:3000->3000/tcp elated_jennings
e4bef2ad3d98 arjan/lab8 "/bin/sh -c 'echo \"A...'" 13 minutes ago Exited (0) 13 minutes ago magical_bohr
62d4e6e38b07 lab8_2build "/bin/sh -c 'echo \"A...'" 19 minutes ago Exited (0) 19 minutes ago silly_herschel
db0c7db429eb busybox "sh" 36 minutes ago Exited (0) 36 minutes ago bold_bell
296930b8b15c busybox "sh" 36 minutes ago Exited (0) 36 minutes ago elastic_carver
PS C:\Users\ngn25\Lab8_4\getting-started> docker stop 975594365bab
975594365bab
PS C:\Users\ngn25\Lab8_4\getting-started> docker rm 975594365bab
975594365bab
PS C:\Users\ngn25\Lab8_4\getting-started> docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
1e54bdf68474 myapp_6533803505 "docker-entrypoint.s..." 3 minutes ago Created
e4bef2ad3d98 arjan/lab8 "/bin/sh -c 'echo \"A...'" 14 minutes ago Exited (0) 14 minutes ago magical_bohr
62d4e6e38b07 lab8_2build "/bin/sh -c 'echo \"A...'" 19 minutes ago Exited (0) 19 minutes ago silly_herschel
db0c7db429eb busybox "sh" 37 minutes ago Exited (0) 37 minutes ago bold_bell
296930b8b15c busybox "sh" 37 minutes ago Exited (0) 37 minutes ago elastic_carver
PS C:\Users\ngn25\Lab8_4\getting-started> docker run -dp 3000:3000 myapp_6533803505
095d9bdcdef424998a54a120e694ff498c25a0482fff4f1c2b60e2b06021afce
PS C:\Users\ngn25\Lab8_4\getting-started>

```

RAM 1.04 GB CPU 2.05% Disk 1.56 GB used (limit 1006.85 GB) > Terminal v 4.37.1

## Lab Worksheet



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

1. เปิด Command line หรือ Terminal บน Docker Desktop
2. ป้อนคำสั่งและทำการรัน container โดยผูกพอร์ต

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

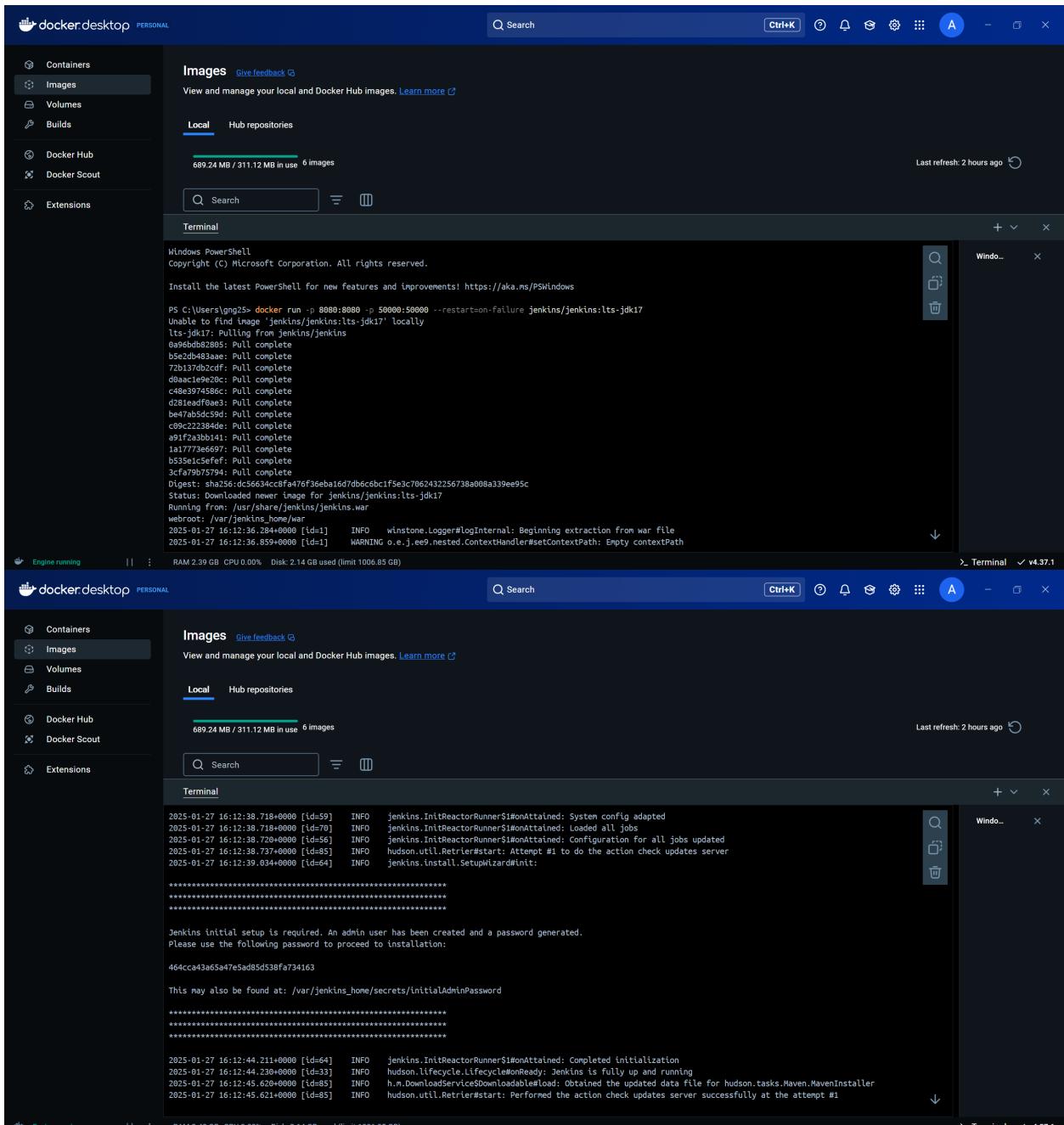
```
$ docker run -p 8080:8080 -p 50000:50000 --restart=on-failure -v  
jenkins_home:/var/jenkins_home jenkins/jenkins:lts-jdk17
```

3. บันทึกรหัสผ่านของ Admin user ไว้สำหรับ log-in ในครั้งแรก

4181b98b7ef34ee993b464eccb61d1c2

[Check point#12] Capture หน้าจอที่แสดงผล Admin password

## Lab Worksheet



The screenshot shows two instances of Docker Desktop running on Windows. Both instances have the terminal tab open, displaying log output from Jenkins containers.

**Terminal Log 1:**

```

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

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

PS C:\Users\rsomsri\pg2> docker run -p 8080:8080 -p 50000:50000 --restart=on-failure jenkins/jenkins:lts-jdk17
Unable to find image 'jenkins/jenkins:lts-jdk17' locally
lts-jdk17: Pulling from jenkins/jenkins
099d6db02085: Pull complete
b5e2d6483aae: Pull complete
72b127db2cdcf: Pull complete
d08ac1e9e20c: Pull complete
c48c3974586c: Pull complete
d281e4df9ae3: Pull complete
be47ab5dc59d: Pull complete
c09f222384de: Pull complete
a91f2a3b1d41: Pull complete
1a17773e6697: Pull complete
b535e1c5feef: Pull complete
3cf7a9b75794: Pull complete
Digest: sha256:dc5634cc0fa476f36eb16d7db6c6bc1fse3c7062432256738a008a339ee95c
Status: Downloaded newer image for jenkins/jenkins:lts-jdk17
Running from: /usr/share/jenkins/jenkins.war
webRoot: /var/jenkins_home/war
2025-01-27 16:12:36.284+0000 [id=1]   INFO  winstome.Logger#logInternal: Beginning extraction from war file
2025-01-27 16:12:36.859+0000 [id=1]  WARNING o.e.j.ee9.nested.ContextHandler#setContextPath: Empty contextPath

```

**Terminal Log 2:**

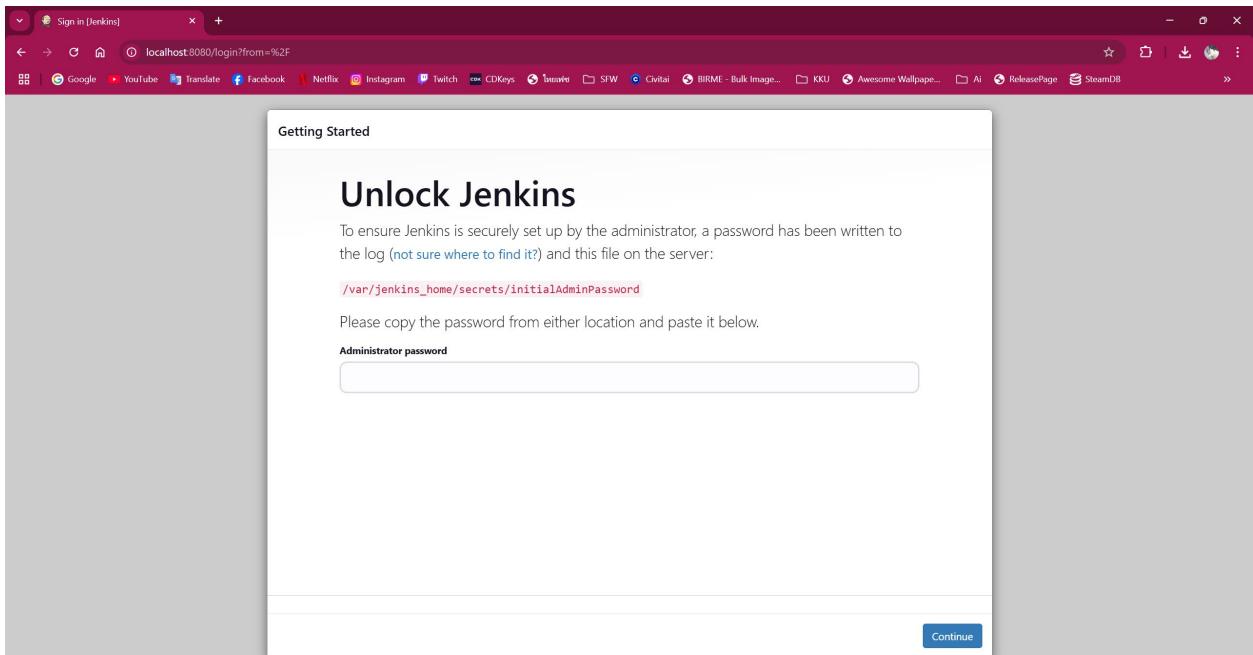
```

2025-01-27 16:12:38.718+0000 [id=59]   INFO  jenkins.InitReactorRunner$1#onAttained: System config adapted
2025-01-27 16:12:38.718+0000 [id=59]   INFO  jenkins.InitReactorRunner$1#onAttained: Loaded all jobs
2025-01-27 16:12:38.720+0000 [id=56]   INFO  jenkins.InitReactorRunner$1#onAttained: Configuration for all jobs updated
2025-01-27 16:12:38.737+0000 [id=85]   INFO  hudson.util.Retrier$#start: Attempt #1 to do the action check updates server
2025-01-27 16:12:39.034+0000 [id=64]   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:
464cca43a65947e5ad85d538fa734163
This may also be found at: /var/jenkins_home/secrets/initialAdminPassword
*****
*****
*****
2025-01-27 16:12:44.211+0000 [id=64]   INFO  jenkins.InitReactorRunner$1#onAttained: Completed initialization
2025-01-27 16:12:44.230+0000 [id=33]   INFO  hudson.lifecycle.Lifecycle$#onReady: Jenkins is fully up and running
2025-01-27 16:12:45.620+0000 [id=85]   INFO  hudson.downloadService.Downloadable$#load: Obtained the updated data file for hudson.tasks.Maven.MavenInstaller
2025-01-27 16:12:45.621+0000 [id=85]   INFO  hudson.util.Retrier$#start: Performed the action check updates server successfully at the attempt #1

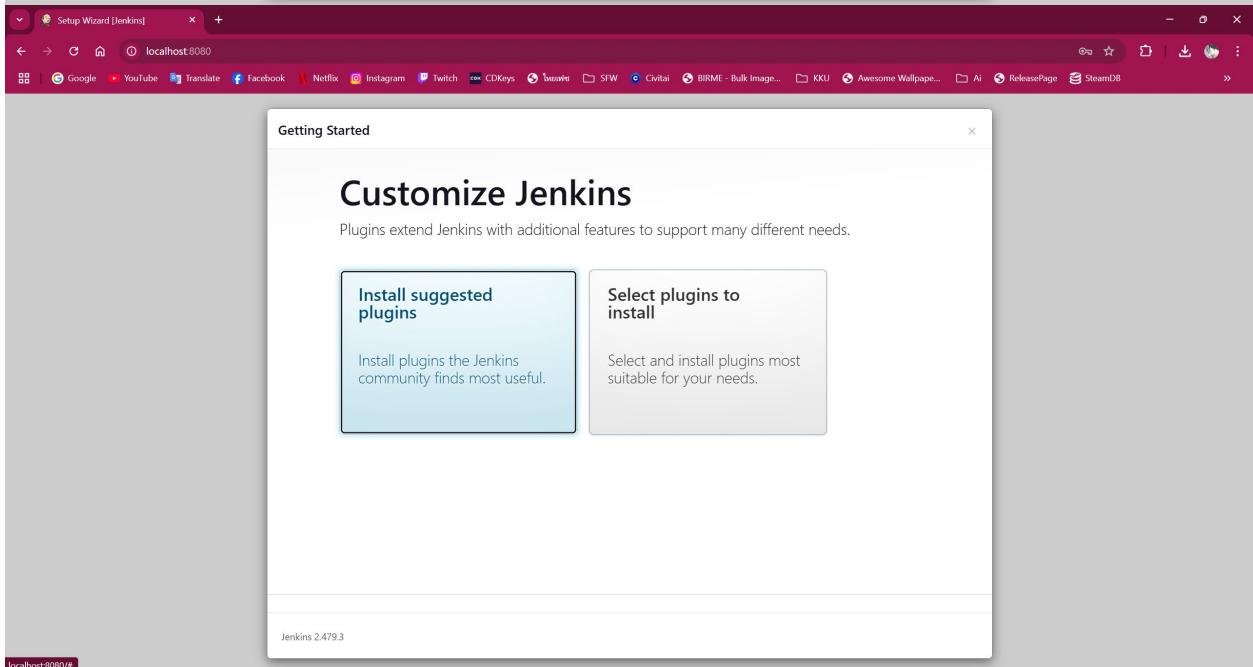
```

4. เมื่อได้รับการยืนยันว่า Jenkins is fully up and running ให้เปิดбраузอร์ 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



The screenshot shows the 'Unlock Jenkins' step of the Jenkins Setup Wizard. It instructs the user to find the initial admin password in the log file at `/var/jenkins_home/secrets/initialAdminPassword`. A text input field is provided for pasting the password, with a placeholder 'Administrator password'. A 'Continue' button is at the bottom right.

The screenshot shows the 'Customize Jenkins' step of the setup wizard. It explains that plugins extend Jenkins with additional features. Two options are presented: 'Install suggested plugins' (selected) and 'Select plugins to install'. Both options have descriptive text below them. At the bottom, it says 'Jenkins 2.479.3'.

## Lab Worksheet

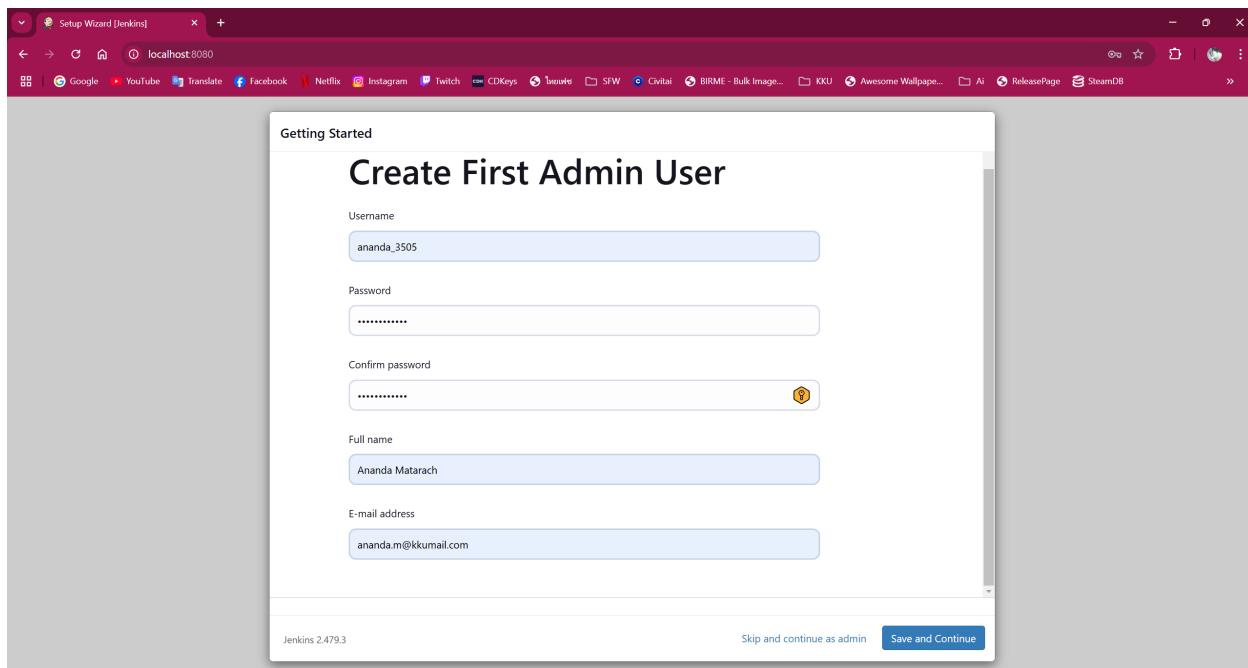
Jenkins 2.479.3

Jenkins 2.479.3

Skip and continue as admin    Save and Continue

## Lab Worksheet



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

9. เลือก Manage Jenkins และไปที่เมนู Plugins

## Lab Worksheet

The screenshot shows the Jenkins Manage Jenkins interface. The left sidebar has links for New Item, Build History, Manage Jenkins (which is selected), and My Views. The main area is titled 'Manage Jenkins' and contains several sections:

- System Configuration:** Includes links for System, Tools, Plugins, and Nodes.
- Security:** Includes links for Security, Credentials, Credential Providers, and Users.
- Status Information:** Includes links for System Information, System Log, Load Statistics, and About Jenkins.

A prominent message at the top center says: "It appears that your reverse proxy set up is broken." with buttons for "More Info" and "Dismiss".

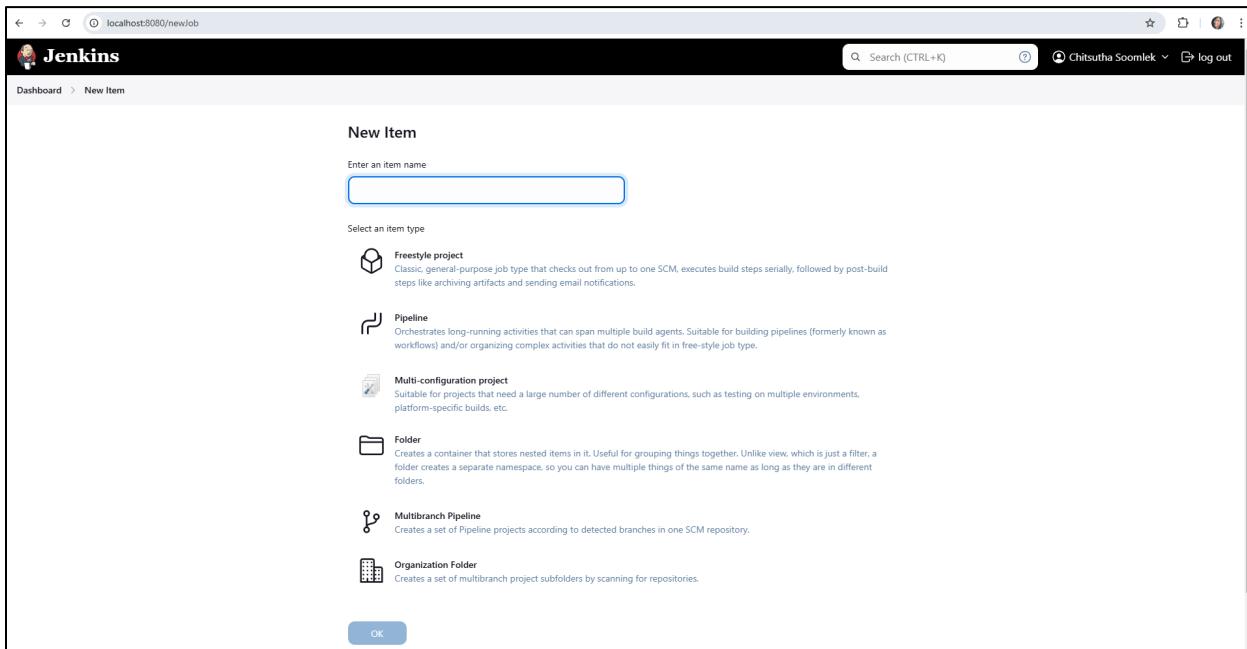
10. ไปที่เมนู Available plugins แล้วเลือกติดตั้ง Robotframework เพิ่มเติม

The screenshot shows the Jenkins Plugin Manager under the 'Available plugins' tab. A search bar at the top has the text 'robot'. Below it, a table lists a single plugin:

Install	Name	Released
<a href="#">Install</a>	Robot Framework 5.0.0	2 mo 7 days ago

The plugin details are shown: "This publisher stores Robot Framework test reports for builds and shows summaries of them in project and build views along with trend graph."

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 จนนั้นตั้งค่าที่จำเป็นในหน้านี้ทั้งหมด ดังนี้

**Description:** Lab 8.5

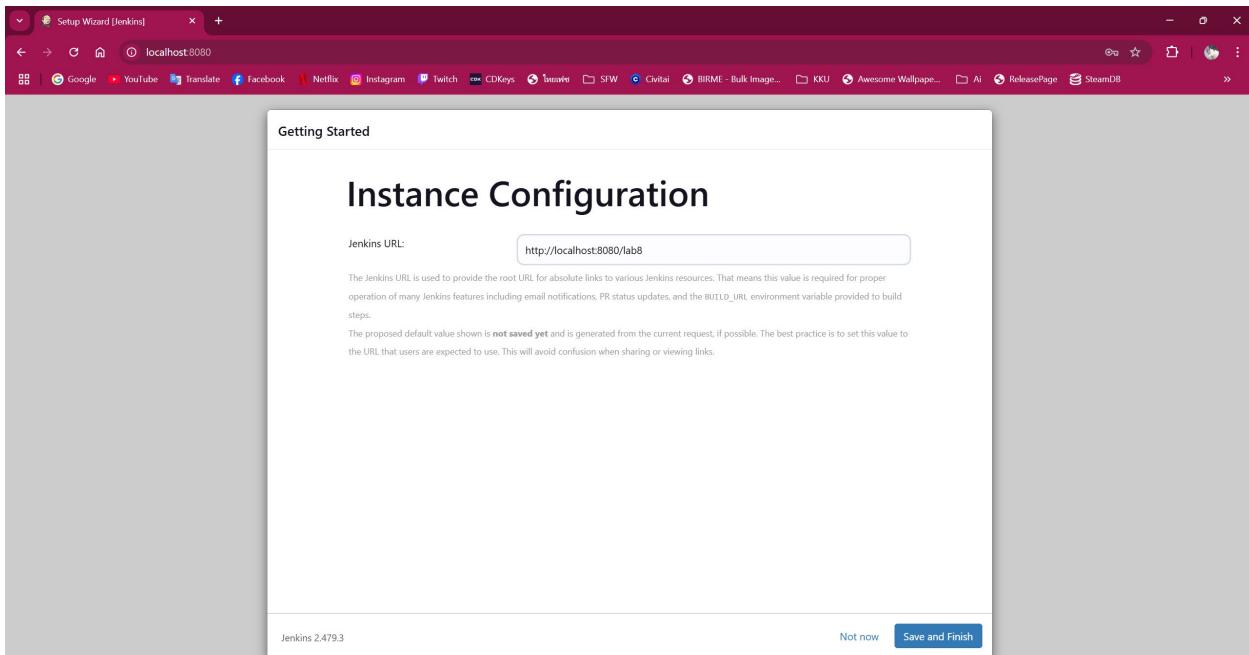
**GitHub project:** กดเลือก และใส่ Project URL เป็น repository ที่เก็บโค้ด .robot (ดูขั้นตอนที่ 12)

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

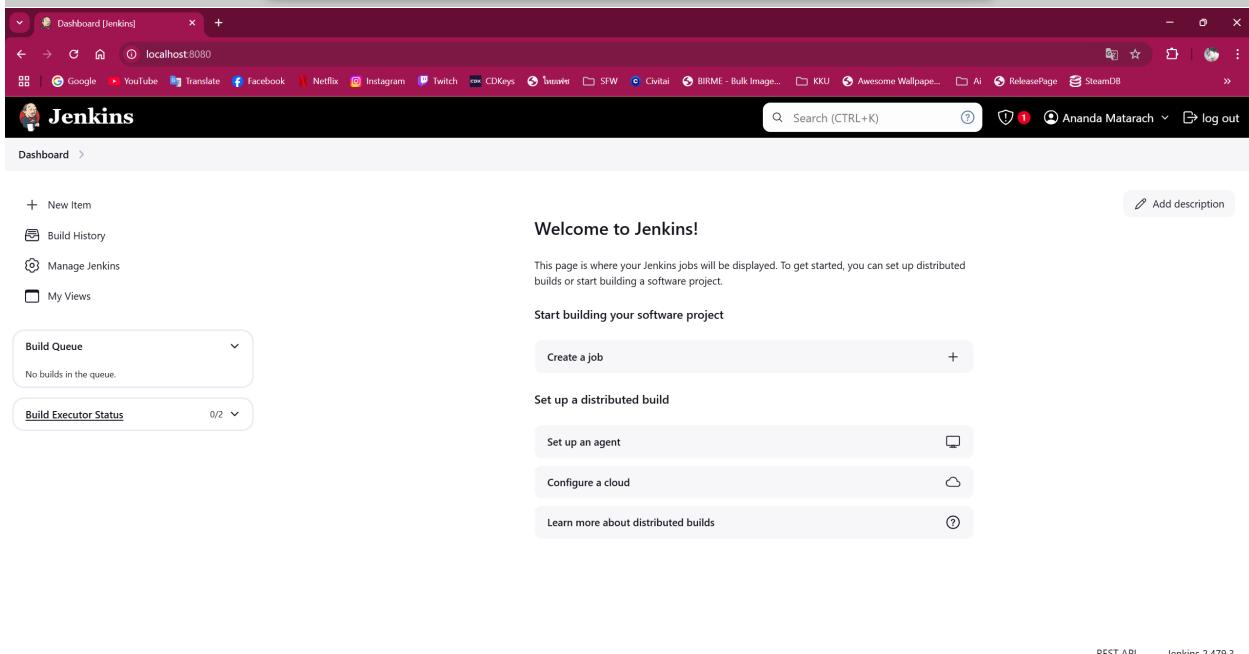
**Build Steps:** เลือก Execute shell และใส่คำสั่งในการรันไฟล์ .robot (หากไฟล์ไม่ได้อยู่ในหน้าแรกของ repository ให้ใส่ Path ไปถึงไฟล์ให้เรียบร้อยด้วย)

[Check point#14] Capture หน้าจอแสดงการตั้งค่า พร้อมกับตอบคำถามต่อไปนี้

## Lab Worksheet



The screenshot shows the Jenkins Setup Wizard at the 'Instance Configuration' step. The URL field contains `http://localhost:8080/lab8`. A note below the field states: "The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD\_URL environment variable provided to build steps. The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links." At the bottom right of the dialog are 'Not now' and 'Save and Finish' buttons.

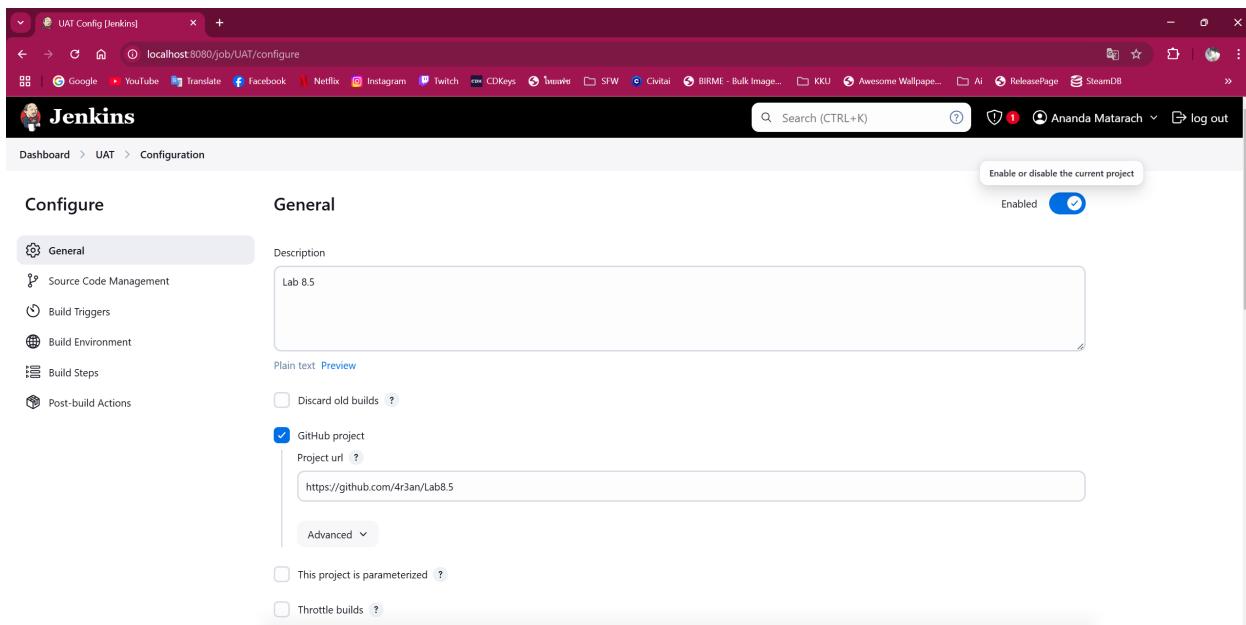
The screenshot shows the Jenkins Dashboard. On the left, there's a sidebar with links: '+ New Item', 'Build History', 'Manage Jenkins', and 'My Views'. The main area has a heading 'Welcome to Jenkins!'. It says: "This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project." Below this, there are sections for 'Start building your software project' and 'Set up a distributed build'. The 'Build Queue' section shows 'No builds in the queue.' The 'Build Executor Status' section shows '0/2'. At the bottom right, there are links for 'REST API' and 'Jenkins 2.479.3'.

## Lab Worksheet

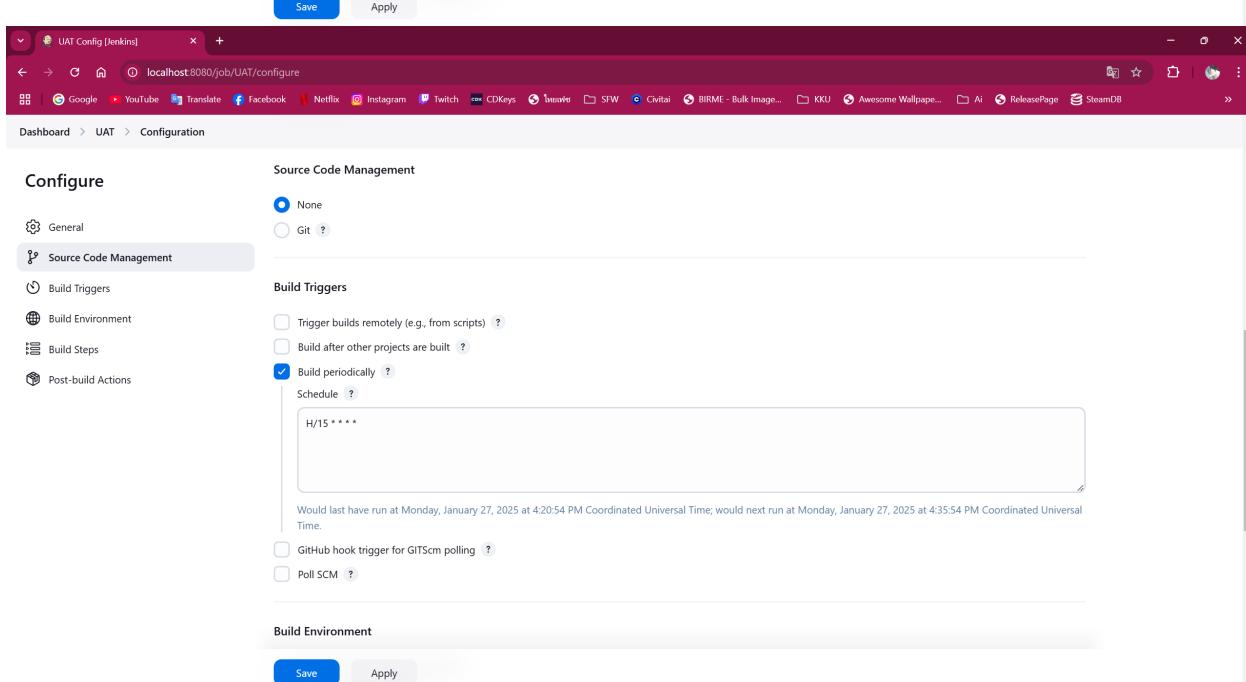
The screenshot shows the Jenkins Plugins management interface. On the left, there's a sidebar with links like 'Updates', 'Available plugins', 'Installed plugins', 'Advanced settings', and 'Download progress'. The main area is titled 'Plugins' and lists various Jenkins plugins with their status: Pipeline: GitHub Groovy Libraries (Success), Pipeline Graph Analysis (Success), Metrics (Success), Pipeline Graph View (Success), Git (Success), EDDSA API (Success), Trilead API (Success), SSH Build Agents (Success), Matrix Authorization Strategy (Success), PAM Authentication (Success), LDAP (Success), Email Extension (Success), Mailer (Success), Theme Manager (Success), Dark Theme (Success), Loading plugin extensions (Success), Robot Framework (Success), and Loading plugin extensions (Success). Below the list are two buttons: 'Go back to the top page' and 'Restart Jenkins when installation is complete and no jobs are running'.

The screenshot shows the Jenkins 'New Item' creation page. At the top, it says 'New Item [Jenkins]'. The main area is titled 'New Item' and has a search bar with 'Search (CTRL+K)'. It asks 'Enter an item name' with 'UAT' typed in. Below that, it asks 'Select an item type' and lists four options: 'Freestyle project' (selected, highlighted with a red border), 'Pipeline', 'Multi-configuration project', and 'Folder'. Each option has a description. At the bottom is an 'OK' button.

## Lab Worksheet



The screenshot shows the Jenkins configuration page for the 'UAT' job. Under the 'General' tab, the 'Description' field contains 'Lab 8.5'. The 'Enabled' switch is turned on. Other options like 'Discard old builds', 'GitHub project', and 'Project url' (set to https://github.com/4r3an/Lab8.5) are also visible.

The screenshot shows the Jenkins configuration page for the 'UAT' job. Under the 'Source Code Management' tab, 'None' is selected. Under 'Build Triggers', 'Build periodically' is selected with a schedule of 'H/15 \* \* \* \*'. Other trigger options like 'Trigger builds remotely' and 'Poll SCM' are available but not selected.

## Lab Worksheet

The screenshot shows the Jenkins configuration interface for a job named 'UAT'. In the 'Build Steps' section, an 'Execute shell' step is selected, containing the command 'robot UAT-Lab7-001.robot'. Below the build steps, there is a 'Post-build Actions' section with a 'Publish Robot Framework test results' option.

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

`ls -al`

`robot UAT-Lab7-001.robot`

โดย `ls -al` เพื่อดูไฟล์ที่มีอยู่และ `robot` เพื่อรันไฟล์ .robot

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

## Lab Worksheet

The screenshot shows the Jenkins configuration interface for a job named "Configuration".

**Source Code Management:**

- Selected provider: Git
- Repository URL: `https://github.com/43an/Lab8_5.git`
- Credentials: None
- Branches to build:

  - Branch Specifier (blank for 'any'): `*/main`

**Build Triggers:**

- Selected trigger: Build periodically
- Schedule: `H/15 * * * *`
- Other triggers available but disabled: Trigger builds remotely, Build after other projects are built, GitHub hook trigger for GITScm polling, Poll SCM.

**Build Environment:**

- Selected option: Delete workspace before build starts

**Buttons:** Save, Apply

## Lab Worksheet

The screenshot shows the Jenkins configuration interface for a job named "UAT".

**Build Steps:**

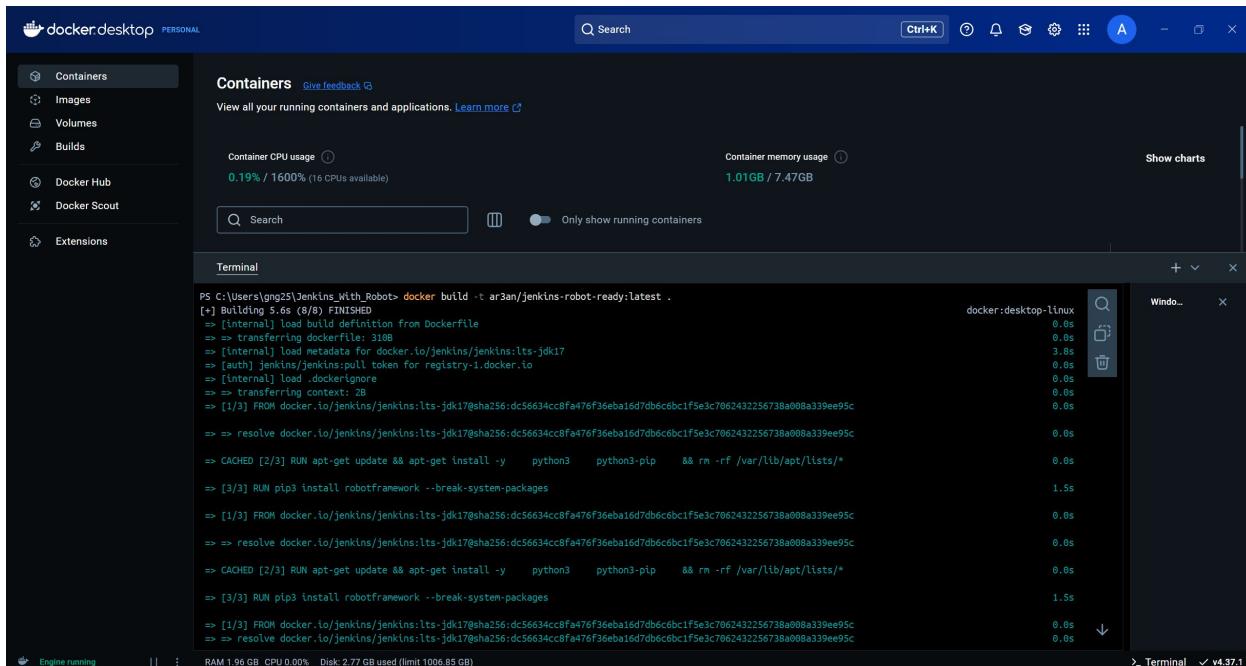
- Execute shell:** Command: `ls -al  
robot UAT-Lab7-001.robot`

**Post-build Actions:**

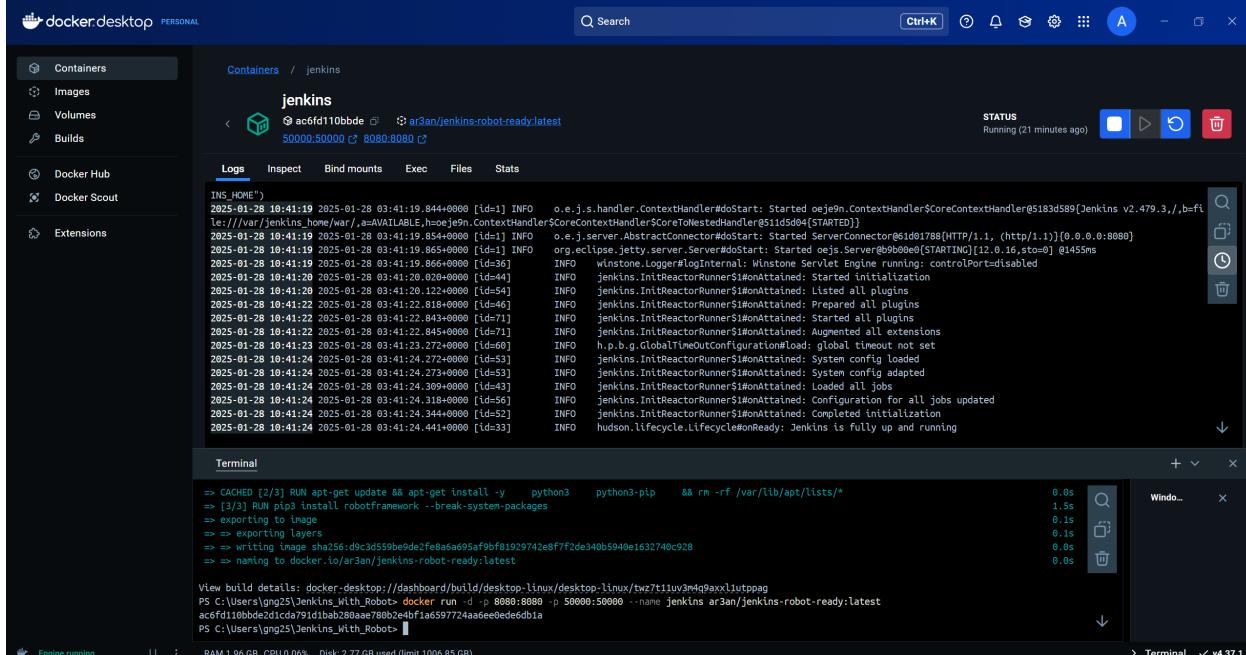
- Publish Robot Framework test results:**
  - Directory of Robot output: Path to directory containing robot xml and html files (relative to build workspace)
  - Advanced settings:
    - Thresholds for build result:
      - Yellow: 20.0
      - Blue: 80.0
    - DEPRECATED! THIS FLAG DOES NOTHING! - Use thresholds for critical tests only
    - Include skipped tests in total count for thresholds

Buttons at the bottom: Save, Apply.

## Lab Worksheet



The screenshot shows the Docker Desktop interface with the 'Containers' tab selected. A terminal window is open, displaying the Jenkins build logs for the 'jenkins-with-robot' container. The logs show the build process starting from a Dockerfile, pulling Jenkins images, installing dependencies, and finally running Jenkins. The terminal also shows the Jenkins UI accessible at <http://0.0.0.0:8080>.

This screenshot shows the Docker Desktop interface again, but this time the Jenkins container is selected in the sidebar. The terminal window displays Jenkins logs, showing the startup of the Jenkins server and the configuration of various components like the Hudson lifecycle and reactor runners. The Jenkins UI is still accessible at <http://0.0.0.0:8080>.

## Lab Worksheet

The screenshot shows the Docker Hub interface for a repository named `ar3an/jenkins-robot-ready`. The repository has one tag, `latest`, which was pushed 2 minutes ago. The Docker commands section shows the command to push a new tag: `docker push ar3an/jenkins-robot-ready:tagname`. The Automated builds section is available with Pro, Team and Business subscriptions.

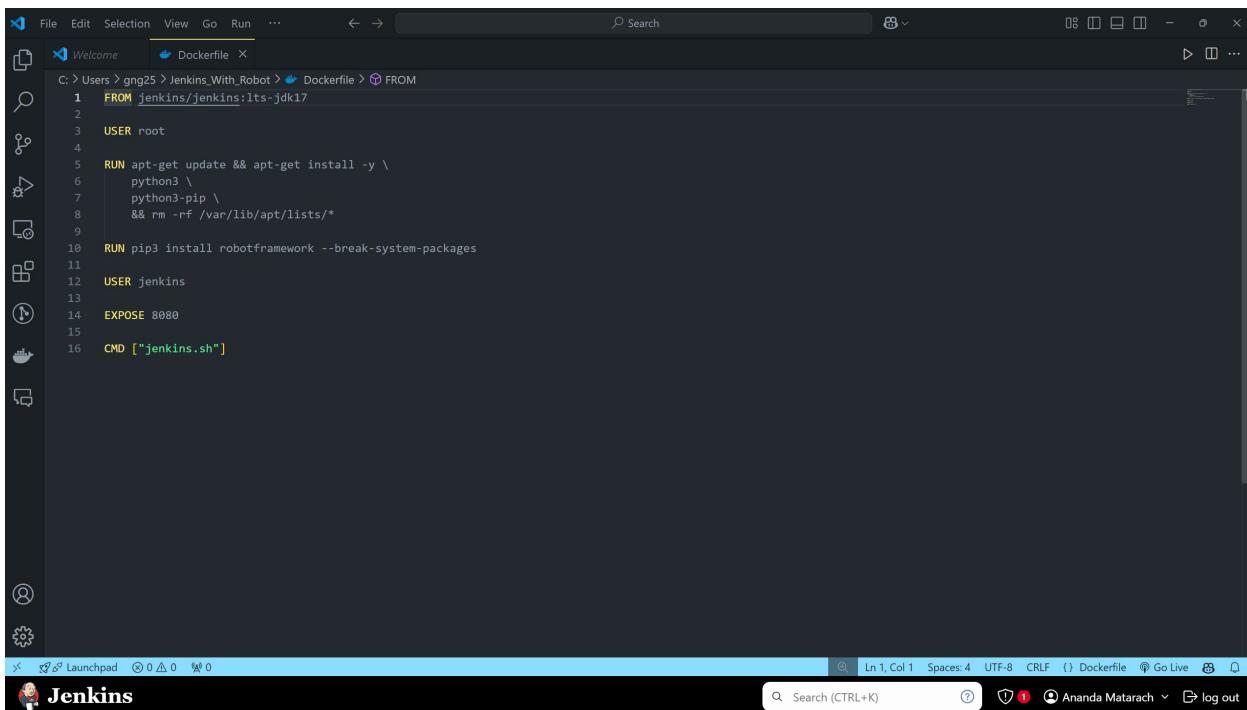
Tag	OS	Type	Pulled	Pushed
latest	Ubuntu	Image	2 minutes ago	2 minutes ago

**Tags**  
This repository contains 1 tag(s).

**Automated builds**  
Manually pushing images to Docker Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.  
Available with Pro, Team and Business subscriptions. [Read more about automated builds](#)

**Repository overview** (Incomplete)  
An overview describes what your image does and how to run it. It displays in the public view of your repository once you have pushed some content.

## Lab Worksheet

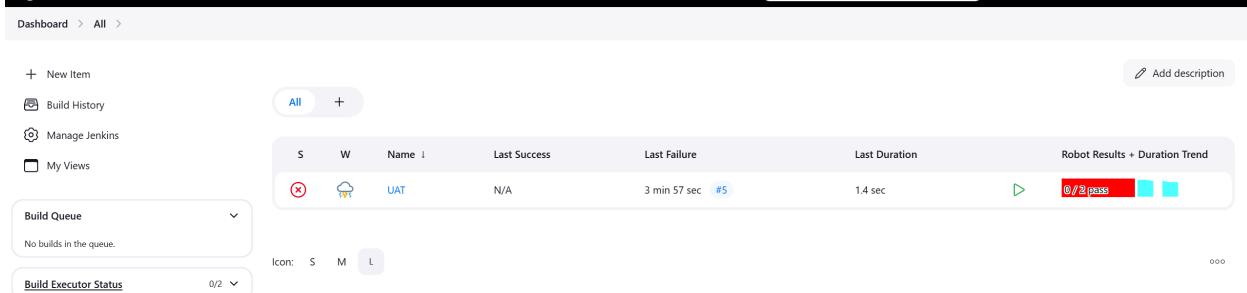


```

FROM jenkins/jenkins:lts-jdk17
USER root
RUN apt-get update && apt-get install -y \
    python3 \
    python3-pip \
    && rm -rf /var/lib/apt/lists/*
RUN pip3 install robotframework --break-system-packages
USER jenkins
EXPOSE 8080
CMD ["jenkins.sh"]

```

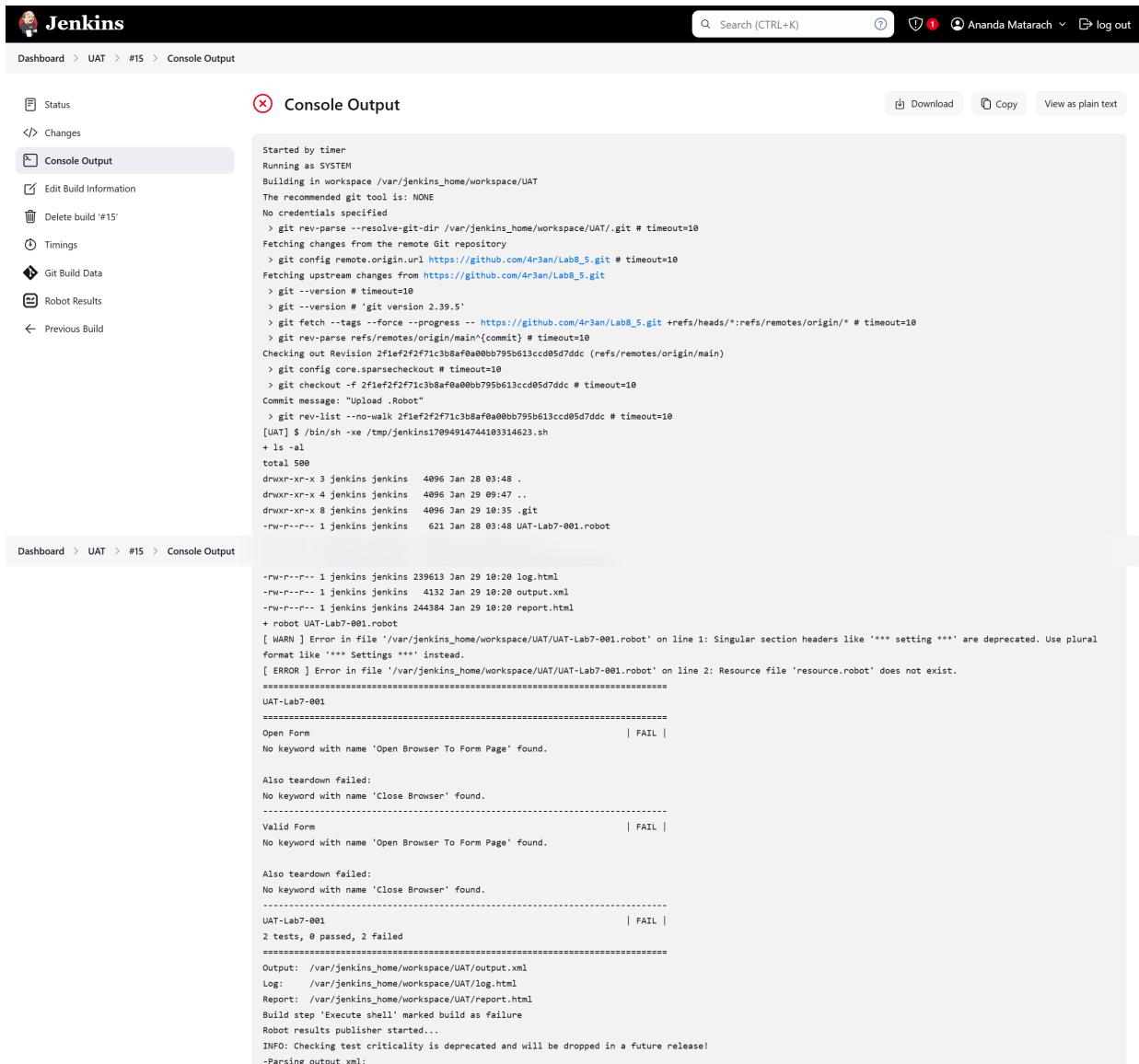
The terminal shows a Dockerfile named Dockerfile. The file defines a base image of jenkins/jenkins:lts-jdk17, runs as root, installs Java and Python via apt-get, removes apt lists, installs Robot Framework via pip3, runs as user jenkins, exposes port 8080, and finally runs the command jenkins.sh.

S	W	Name	Last Success	Last Failure	Last Duration	Robot Results + Duration Trend
		UAT	N/A	3 min 57 sec #5	1.4 sec	0/2 pass

The Jenkins dashboard displays a single build item named "UAT". The build status is marked with a red "X" and a cloud icon, indicating a failure. The last success was noted as "N/A". The last failure occurred 3 minutes and 57 seconds ago. The total duration of the last run was 1.4 seconds. A summary at the bottom indicates 0 out of 2 tests passed. There are also icons for "Add description", "Build History", "Manage Jenkins", and "My Views".

## Lab Worksheet



The screenshot shows the Jenkins interface with the following details:

- Project Path:** Dashboard > UAT > #15 > Console Output
- Job Information:** Status, Changes, Console Output (selected), Edit Build Information, Delete build #15, Timings, Git Build Data, Robot Results, Previous Build.
- Console Output Content:**

```

Started by timer
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/4r3an/Lab8_5.git # timeout=10
Fetching upstream changes from https://github.com/4r3an/Lab8_5.git
> git --version # timeout=10
> git --version # 'git version 2.39.5'
> git fetch --tags --force --progress -- https://github.com/4r3an/Lab8_5.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/main{commit} # timeout=10
Checking out Revision 2f1ef2f2f71c3b8af0a0bb795b613ccd05d7ddc (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f 2f1ef2f2f71c3b8af0a0bb795b613ccd05d7ddc # timeout=10
Commit message: "Upload .Robot"
> git rev-list --no-walk 2f1ef2f2f71c3b8af0a0bb795b613ccd05d7ddc # timeout=10
[UAT] $ /bin/sh -xe /tmp/jenkins17094914744103314623.sh
+ ls -al
total 500
drwxr-xr-x 3 jenkins jenkins 4096 Jan 28 03:48 .
drwxr-xr-x 4 jenkins jenkins 4096 Jan 29 09:47 ..
drwxr-xr-x 8 jenkins jenkins 4096 Jan 29 10:35 .git
-rw-r--r-- 1 jenkins jenkins 621 Jan 28 03:48 UAT-Lab7-001.robot

-rw-r--r-- 1 jenkins jenkins 239613 Jan 29 10:20 log.html
-rw-r--r-- 1 jenkins jenkins 4132 Jan 29 10:20 output.xml
-rw-r--r-- 1 jenkins jenkins 244384 Jan 29 10:20 report.html
+ robot UAT-Lab7-001.robot
[ WARN ] Error in file '/var/jenkins_home/workspace/UAT/UAT-Lab7-001.robot' on line 1: Singular section headers like '*** setting ***' are deprecated. Use plural format like '*** Settings ***' instead.
[ ERROR ] Error in file '/var/jenkins_home/workspace/UAT/UAT-Lab7-001.robot' on line 2: Resource file 'resource.robot' does not exist.

=====
UAT-Lab7-001
=====
Open Form | FAIL |
No keyword with name 'Open Browser To Form Page' found.

Also teardown failed:
No keyword with name 'Close Browser' found.

=====
Valid Form | FAIL |
No keyword with name 'Open Browser To Form Page' found.

Also teardown failed:
No keyword with name 'Close Browser' found.

=====
UAT-Lab7-001 | FAIL |
2 tests, 0 passed, 2 failed
=====
Output: /var/jenkins_home/workspace/UAT/output.xml
Log: /var/jenkins_home/workspace/UAT/log.html
Report: /var/jenkins_home/workspace/UAT/report.html
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:

```

## Lab Worksheet

```

Dashboard > UAT > #15 > Console Output
-----
No keyword with name 'Close Browser' found.
-----
Valid Form | FAIL |
No keyword with name 'Open Browser To Form Page' found.
-----
Also teardown failed:
No keyword with name 'Close Browser' found.
-----
UAT-Lab7-001 | FAIL |
2 tests, 0 passed, 2 failed
=====
=====
Output: /var/jenkins_home/workspace/UAT/output.xml
Log: /var/jenkins_home/workspace/UAT/log.html
Report: /var/jenkins_home/workspace/UAT/report.html
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

คำสั่งภายใน Dockerfile เพื่อติดตั้ง Python ภายใต้ Jenkins

```
FROM jenkins/jenkins:lts-jdk17
```

USER root

```
RUN apt-get update && apt-get install -y \
    python3 \
    python3-pip \
&& rm -rf /var/lib/apt/lists/*
```

```
RUN pip3 install robotframework --break-system-packages
```

USER jenkins

EXPOSE 8080

CMD ["jenkins.sh"]

คำสั่งที่ใช้ในการแก้ปัญหา Jenkins ใน Docker

1. docker build -t ar3an/jenkins-robot-ready:latest .

### Lab Worksheet

2. docker run -d -p 8080:8080 -p 50000:50000 --name jenkins ar3an/jenkins-robot-ready:latest
3. docker push ar3an/jenkins-robot-ready:latest