

## Lab Worksheet

ชื่อ-นามสกุล อติยา บุษากุล รหัสนักศึกษา 653380218-5 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

---

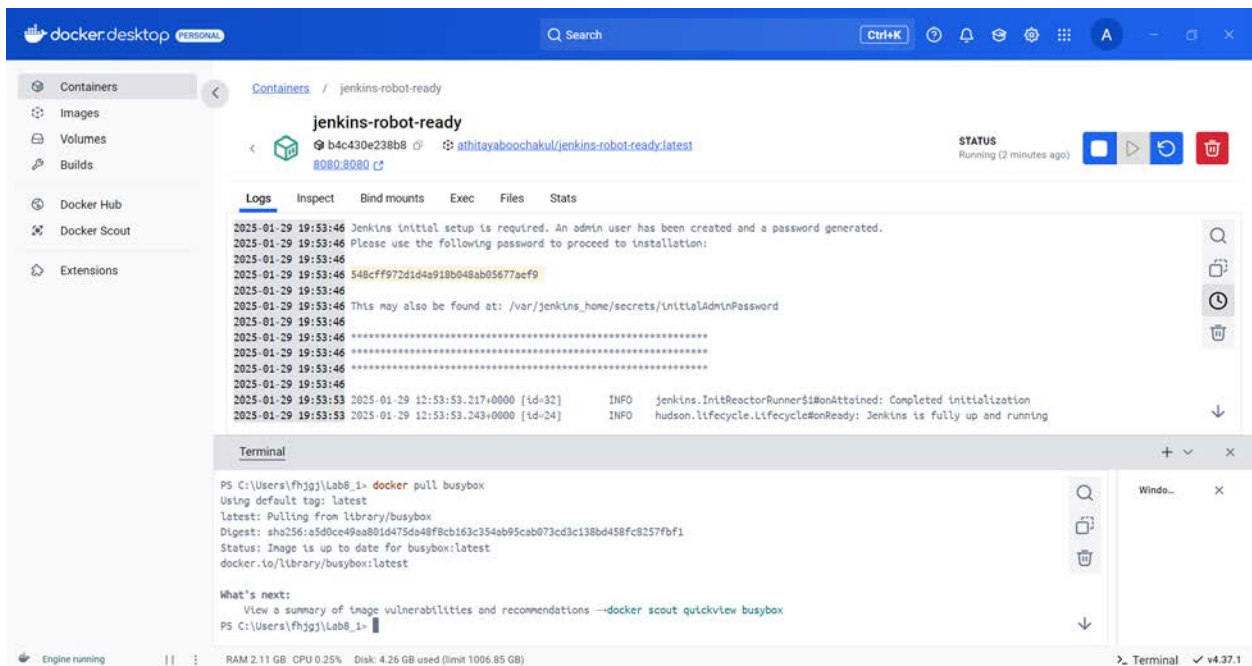
1. ติดตั้ง Docker desktop ลงบนเครื่องคอมพิวเตอร์ โดยดาวน์โหลดจาก <https://www.docker.com/get-started>
2. สร้าง Account บน Docker hub (<https://hub.docker.com/signup>)
3. กำหนดให้ \$ หมายถึง Command prompt และ <> หมายถึง ให้ป้อนค่าของพารามิเตอร์ที่กำหนด

## Lab Worksheet

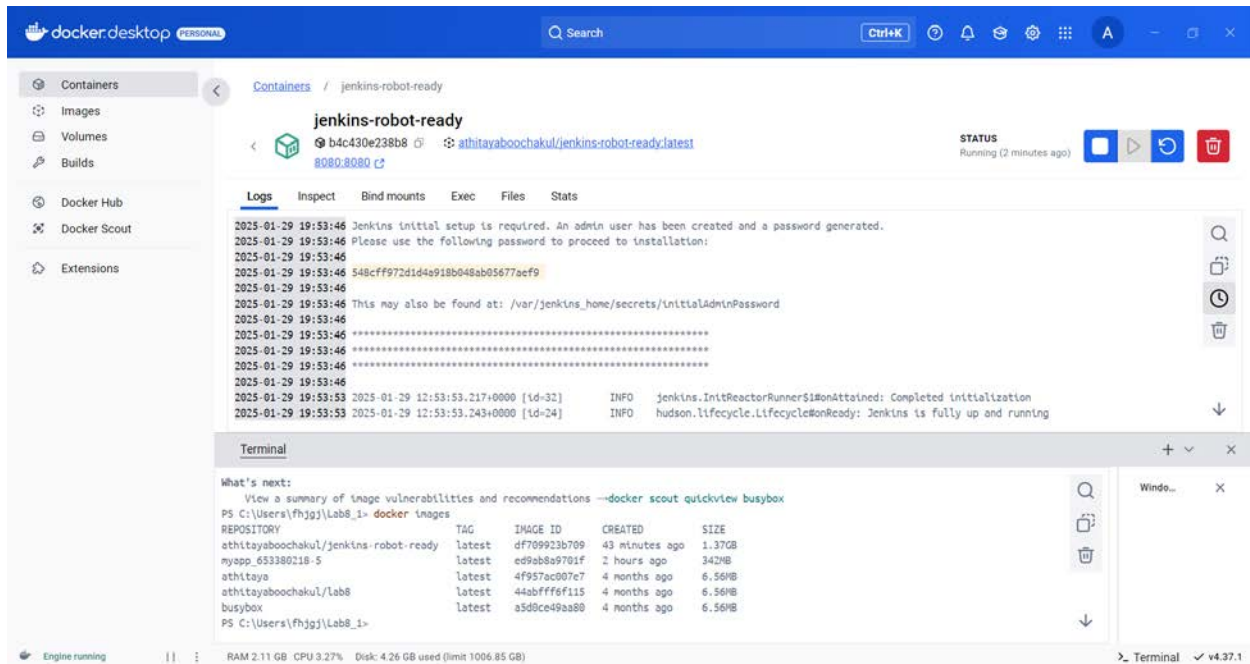
## แบบฝึกปฏิบัติที่ 8.1 Hello world - รัน Container จาก Docker image

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` สำหรับกรณีที่เกิดปัญหา 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

ชื่อของอิมเมจที่อยู่ในระบบ Docker

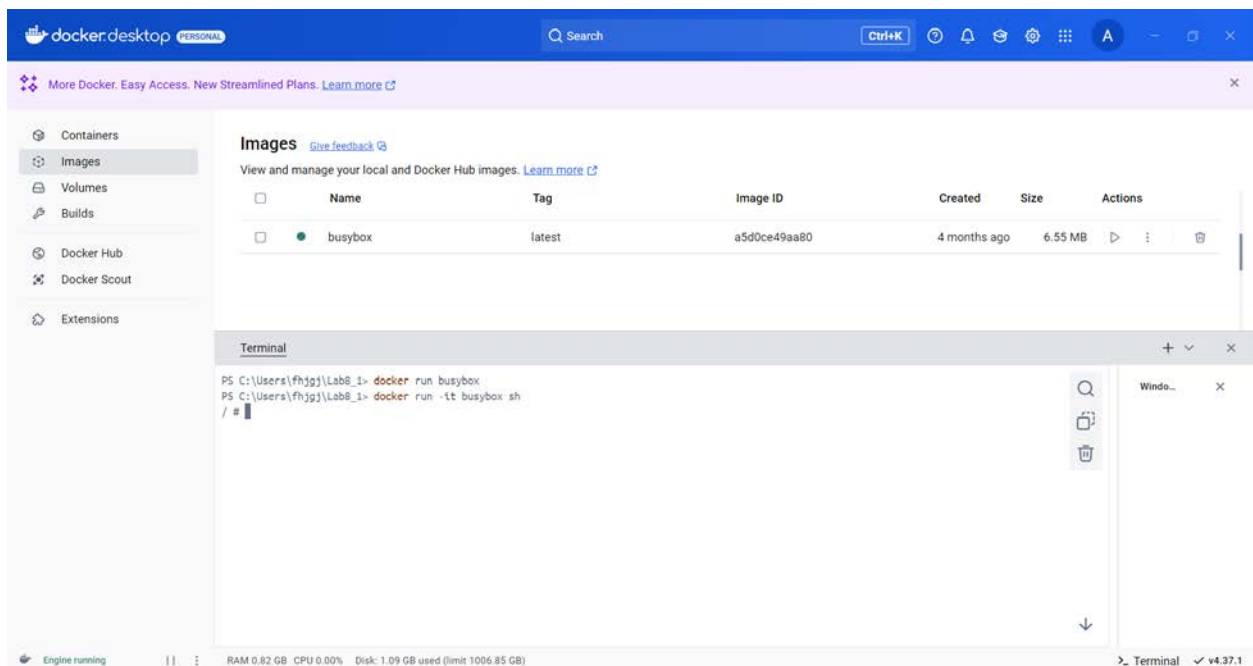
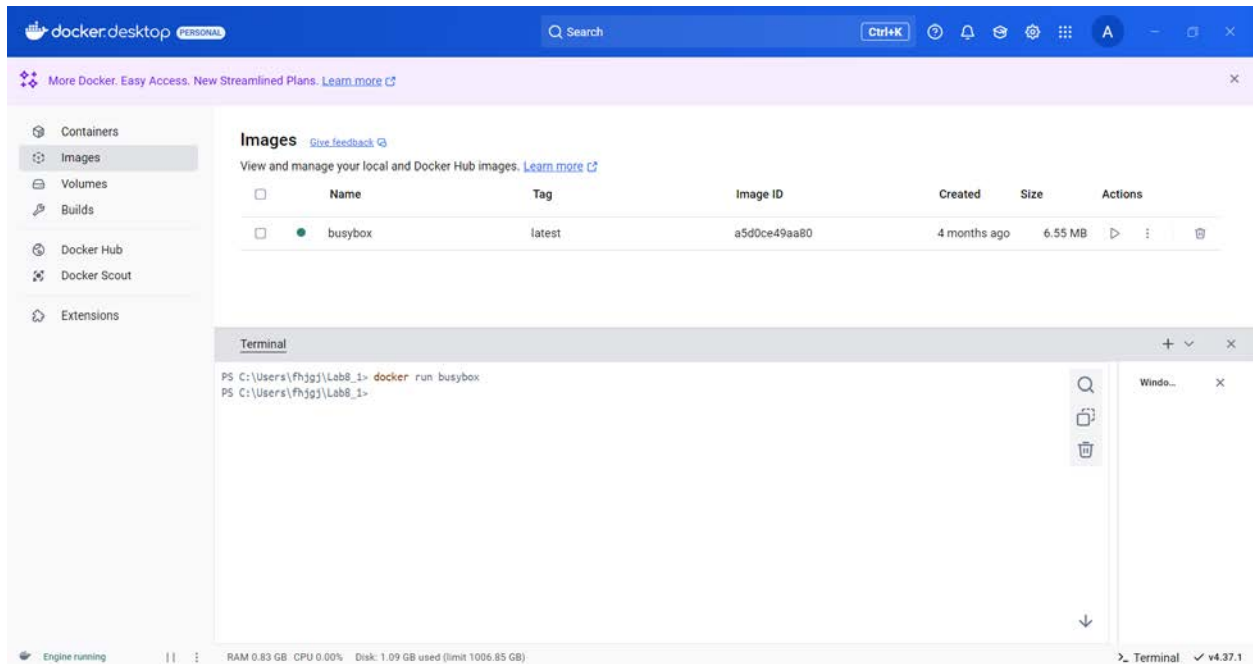
(2) Tag ที่ใช้บ่งบอกถึงอะไร

Tag ใช้บ่งบอกถึงเวอร์ชันของอิมเมจ นั้น ๆ

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

## Lab Worksheet

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



## Lab Worksheet

The screenshot shows the Docker Desktop interface. The left sidebar contains navigation options: Containers, Images (selected), Volumes, Builds, Docker Hub, Docker Scout, and Extensions. The main area displays the 'Images' tab, showing a table of local images. The table has columns for Name, Tag, Image ID, Created, Size, and Actions. One image is listed: 'busybox' with tag 'latest' and Image ID 'a5d0ce49aa80', created 4 months ago and 6.55 MB in size. Below the table is a terminal window showing the execution of 'docker run busybox' and 'docker run -it busybox sh'. The terminal output shows the directory listing of the busybox container: 'ls' results in 'bin dev etc home lib lib64 proc root sys tmp usr var'. The bottom status bar indicates 'Engine running', 'RAM 0.78 GB', 'CPU 0.00%', and 'Disk: 1.09 GB used (limit 1006.85 GB)'. The terminal window title is 'Terminal v4.37.1'.

| Name    | Tag    | Image ID     | Created      | Size    | Actions                |
|---------|--------|--------------|--------------|---------|------------------------|
| busybox | latest | a5d0ce49aa80 | 4 months ago | 6.55 MB | [Play] [Info] [Delete] |

```

PS C:\Users\fhjgj\Lab8_1> docker run busybox
PS C:\Users\fhjgj\Lab8_1> docker run -it busybox sh
/ # ls
bin  dev  etc  home  lib  lib64  proc  root  sys  tmp  usr  var
/ #
  
```

The image displays two screenshots of the Docker Desktop interface, illustrating the process of running a container from a local image.

**Top Screenshot:**

- The **Images** tab is selected, showing a table of local images:

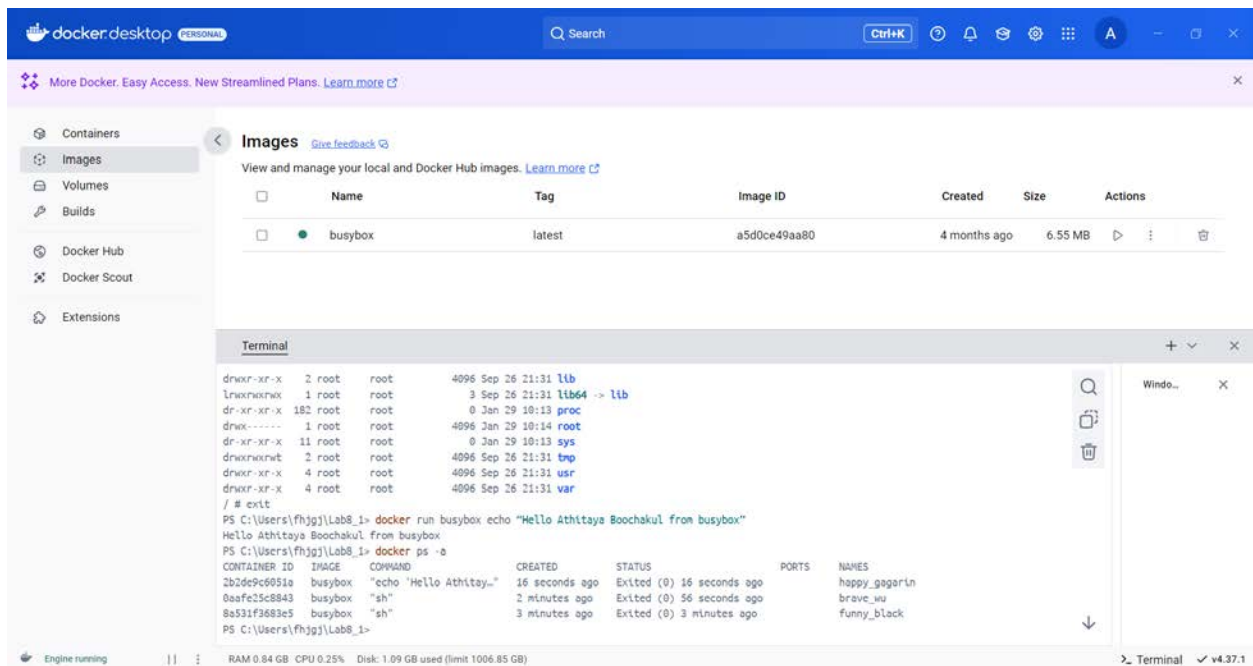
| Name    | Tag    | Image ID     | Created      | Size    | Actions               |
|---------|--------|--------------|--------------|---------|-----------------------|
| busybox | latest | a5d0ce49aa80 | 4 months ago | 6.55 MB | [Play] [List] [Trash] |

- Below the table, the **Terminal** window shows the output of the command `docker ps -a`, listing various containers and their details.

**Bottom Screenshot:**

- The **Images** tab is still selected, showing the same table of local images.
- The **Terminal** window now shows the command `docker run busybox echo "Hello Athitaya Boochakul from busybox"` being executed, with the output `Hello Athitaya Boochakul from busybox` displayed.

## Lab Worksheet



(1) เมื่อใช้ option `-it` ในคำสั่ง `run` ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป

การใช้ `docker run -it busybox sh` ช่วยให้สามารถ เข้าไปโต้ตอบกับคอนเทนเนอร์ busybox ได้แบบอินเทอร์แอคทีฟ โดยเปิด sh (shell) และสามารถพิมพ์คำสั่งต่าง ๆ ได้ เหมือนใช้ Linux

Terminal

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

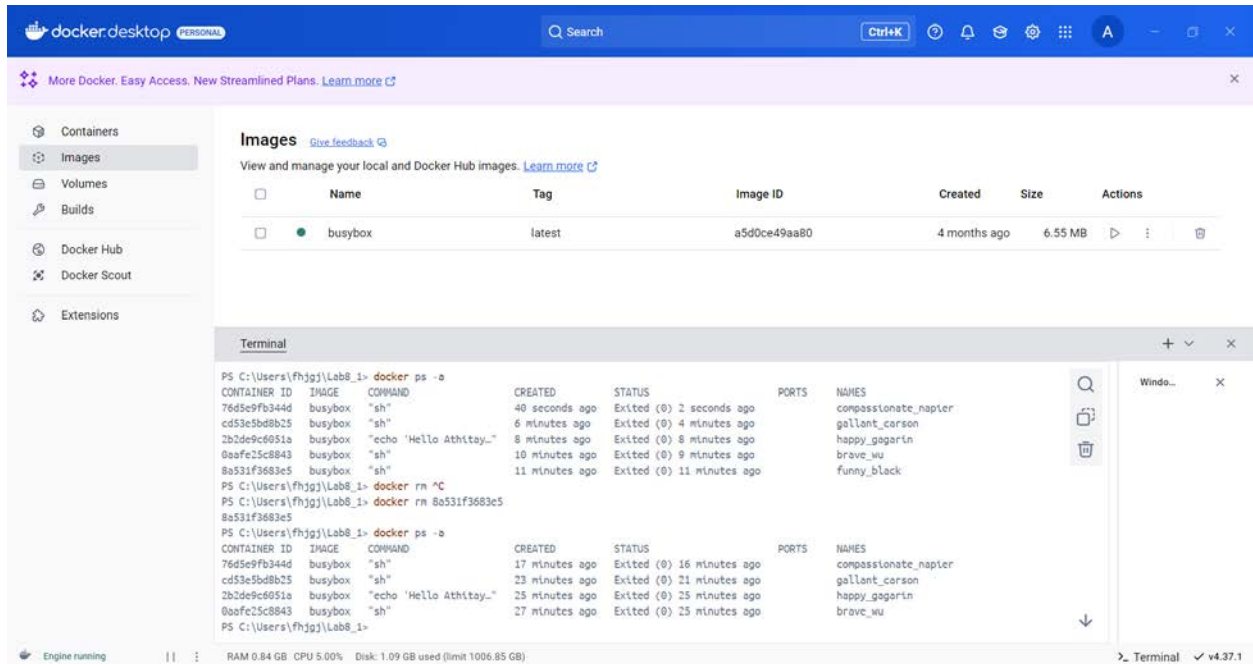
ระบบจะแสดงรายการ คอนเทนเนอร์ทั้งหมด ที่เคยถูกสร้างขึ้น (รวมถึงที่กำลังรันและที่หยุดทำงานแล้ว) โดยคอลัมน์ STATUS ใช้แสดง สถานะของแต่ละคอนเทนเนอร์



## Lab Worksheet

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

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



## แบบฝึกปฏิบัติที่ 8.2: สร้าง Docker file และ Docker image

1. เปิดใช้งาน Docker desktop และ Login ด้วย Username และ Password ที่ลงทะเบียนกับ Docker Hub เอาไว้
2. เปิด Command line หรือ Terminal จากนั้นสร้าง Directory ชื่อ Lab8\_2
3. ย้ายตำแหน่งปัจจุบันไปที่ Lab8\_2 เพื่อใช้เป็น Working directory
4. สร้าง Dockerfile.swp ไว้ใน Working directory

สำหรับเครื่องที่ใช้ระบบปฏิบัติการวินโดวส์ (Windows) บันทึกคำสั่งต่อไปนี้ลงในไฟล์ โดยใช้ Text Editor ที่มี

FROM busybox

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

CMD echo "ชื่อ-นามสกุล รหัสนักศึกษา ชื่อเล่น"

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

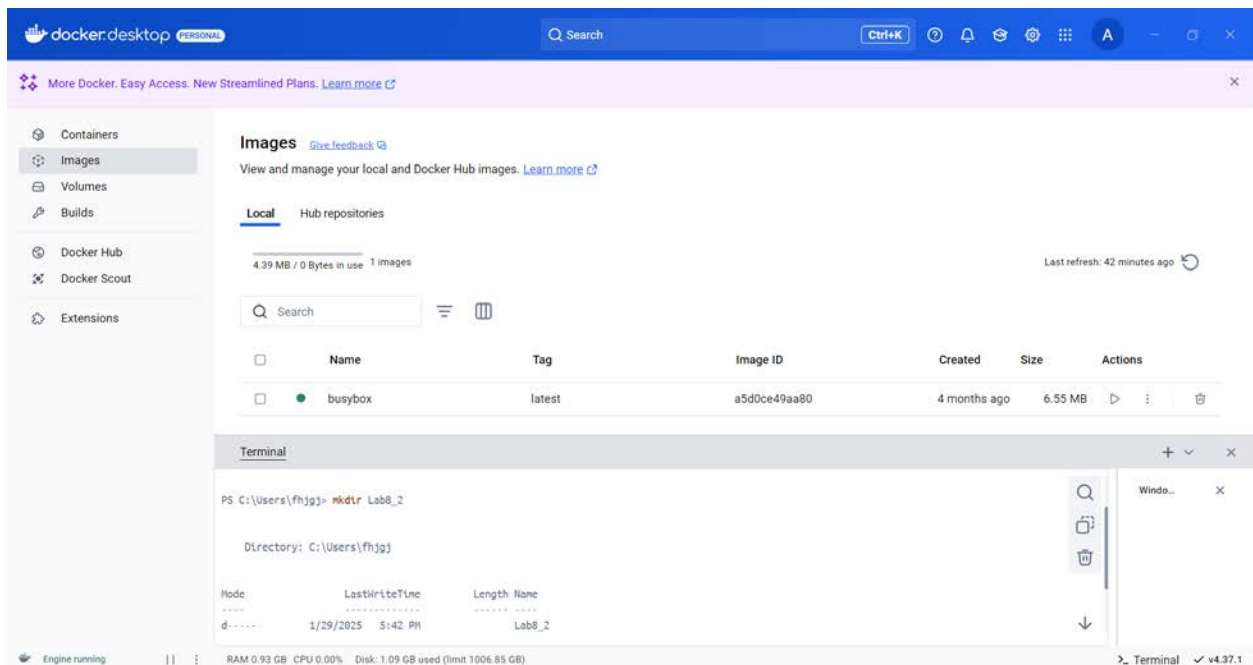
แล้วใช้ Text Editor ในการใส่เนื้อหาแทน

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) แสดงผลลัพธ์ที่ได้ในขั้นตอนที่ 5 พร้อมกับตอบคำถามต่อไปนี้



## Lab Worksheet

The screenshot displays the Docker Desktop application interface. The top navigation bar includes the Docker logo, a search bar, and a 'Ctrl+K' shortcut. Below the navigation bar, a sidebar on the left lists various components: Containers, Images (selected), Volumes, Builds, Docker Hub, Docker Scout, and Extensions. The main content area is divided into two sections: 'Images' and 'Walkthroughs'. The 'Images' section shows a table of local Docker images, with one image listed: 'busybox' with the tag 'latest', image ID 'a5d0ce49aa80', created '4 months ago', and size '6.55 MB'. Below the table, it indicates 'Showing 1 item'. The 'Walkthroughs' section is currently empty. At the bottom, a terminal window is open, showing the command prompt 'PS C:\Users\fhjgg> cd Lab8\_2' and the output 'notepad Dockerfile.swp'. The terminal also shows the command 'PS C:\Users\fhjgg> notepad Dockerfile.swp' and the output 'notepad Dockerfile.swp'. The bottom status bar indicates 'Engine running' and system resources: 'RAM 0.94 GB CPU 0.00% Disk: 1.09 GB used (limit 1006.85 GB)'. The version 'v4.37.1' is also visible.

**Images** [Give feedback](#)

View and manage your local and Docker Hub images. [Learn more](#)

| Name    | Tag    | Image ID     | Created      | Size    | Actions  |
|---------|--------|--------------|--------------|---------|--|
| busybox | latest | a5d0ce49aa80 | 4 months ago | 6.55 MB | <a href="#">Play</a> <a href="#">Info</a> <a href="#">Delete</a> |

Showing 1 item

**Walkthroughs**

**Terminal**

```

Mode                LastWriteTime         Length Name
----                -
d-----          1/29/2025   5:42 PM           Lab8_2

PS C:\Users\fhjgg> cd Lab8_2
PS C:\Users\fhjgg\Lab8_2> notepad Dockerfile.swp
PS C:\Users\fhjgg\Lab8_2>
  
```

Engine running | RAM 0.94 GB CPU 0.00% Disk: 1.09 GB used (limit 1006.85 GB) | Terminal v4.37.1

## Lab Worksheet

The screenshot shows the Docker Desktop interface. The 'Images' tab is active, displaying a list of local and Docker Hub images. The table below shows the details of the images:

| Name     | Tag    | Image ID     | Created      | Size    | Actions                   |
|----------|--------|--------------|--------------|---------|---------------------------|
| busybox  | latest | a5d0ce49aa80 | 4 months ago | 6.55 MB | [Play] [Refresh] [Delete] |
| athitaya | latest | 4f957ac007e7 | 4 months ago | 6.55 MB | [Play] [Refresh] [Delete] |

Below the table, the 'Walkthroughs' section is visible. The 'Terminal' window shows the command 'docker build -t athitaya .' and its output, including warnings about JSON arguments and instructions. The bottom status bar indicates 'Engine running' with RAM 0.96 GB, CPU 0.00%, and Disk 1.09 GB used.

(1) คำสั่งที่ใช้ในการ run คือ

Docker run athitaya

(2) Option -t ในคำสั่ง \$ docker build ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป

Option -t ช่วยให้เราสามารถตั้งชื่อและแท็ก (เวอร์ชัน) ของ Docker image ที่ถูกสร้างขึ้น ทำให้ง่ายในการจัดการและเรียกใช้งานอิมเมจในภายหลัง

## Lab Worksheet

## แบบฝึกปฏิบัติที่ 8.3: การแชร์ Docker image ผ่าน Docker Hub

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

สำหรับเครื่องที่ใช้ระบบปฏิบัติการวินโดวส์ บันทึกคำสั่งต่อไปนี้ลงในไฟล์ โดยใช้ Text Editor ที่มี

```
FROM busybox
```

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

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

สำหรับเครื่องที่ใช้ระบบปฏิบัติการ MacOS หรือ Linux บนหน้าต่าง Terminal และป้อนคำสั่งต่อไปนี้

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

```
FROM busybox
```

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

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

```
EOF
```

หรือใช้คำสั่ง

```
$ touch Dockerfile
```

แล้วใช้ Text Editor ในการใส่เนื้อหาแทน

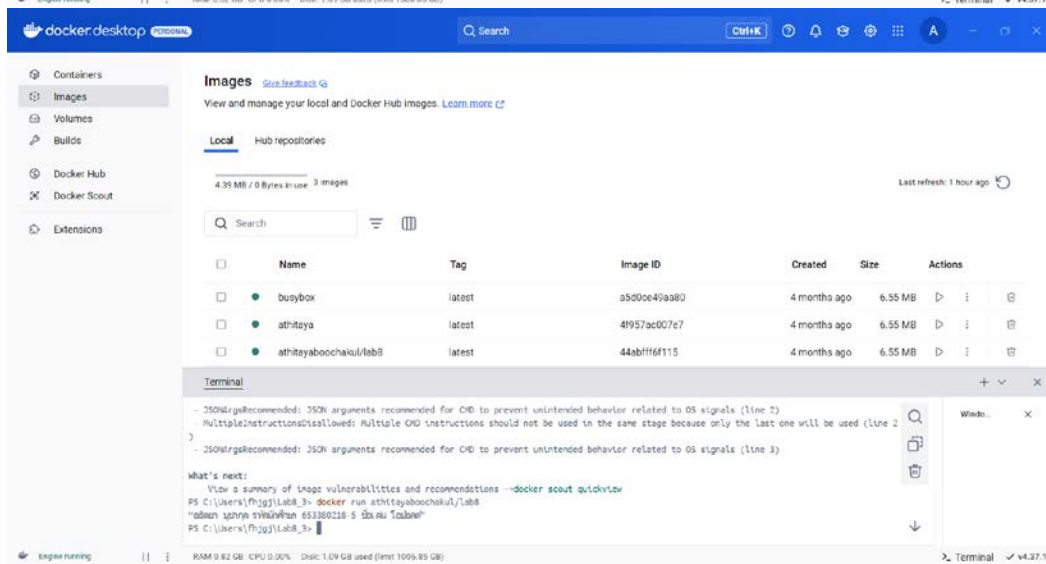
6. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้

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

5. ทำการรัน Docker image บน Container ในเครื่องของตัวเองเพื่อทดสอบผลลัพธ์ ด้วยคำสั่ง

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

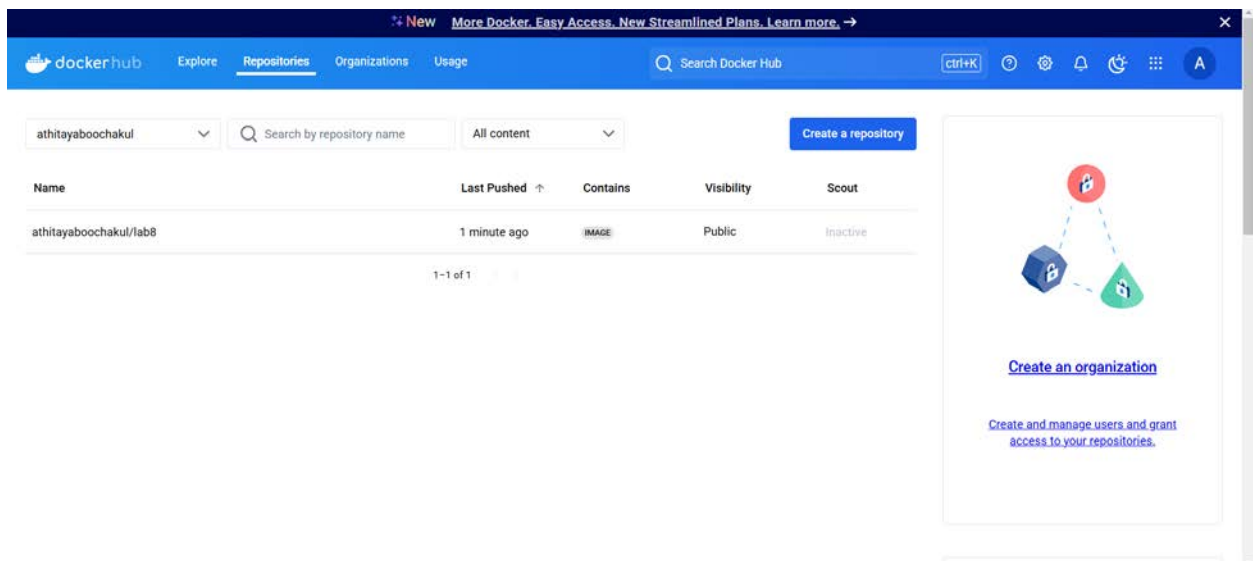
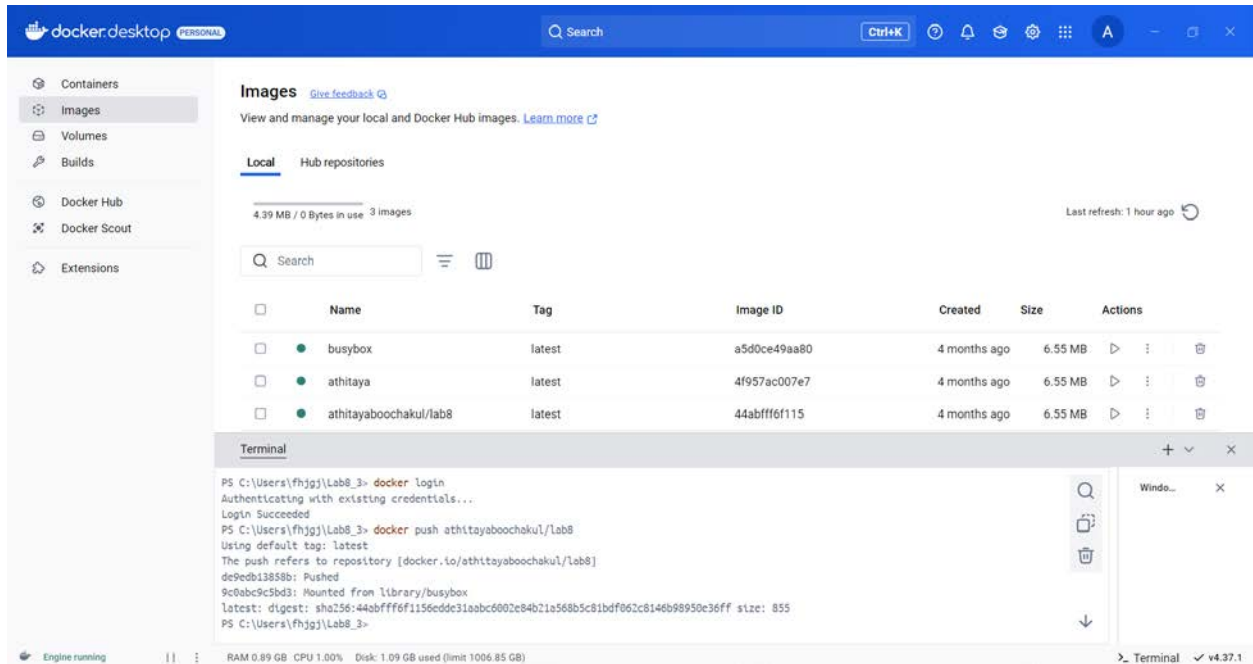
[Check point#5] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงผลลัพธ์ที่ได้ในขั้นตอนที่ 5



- 13

## Lab Worksheet

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

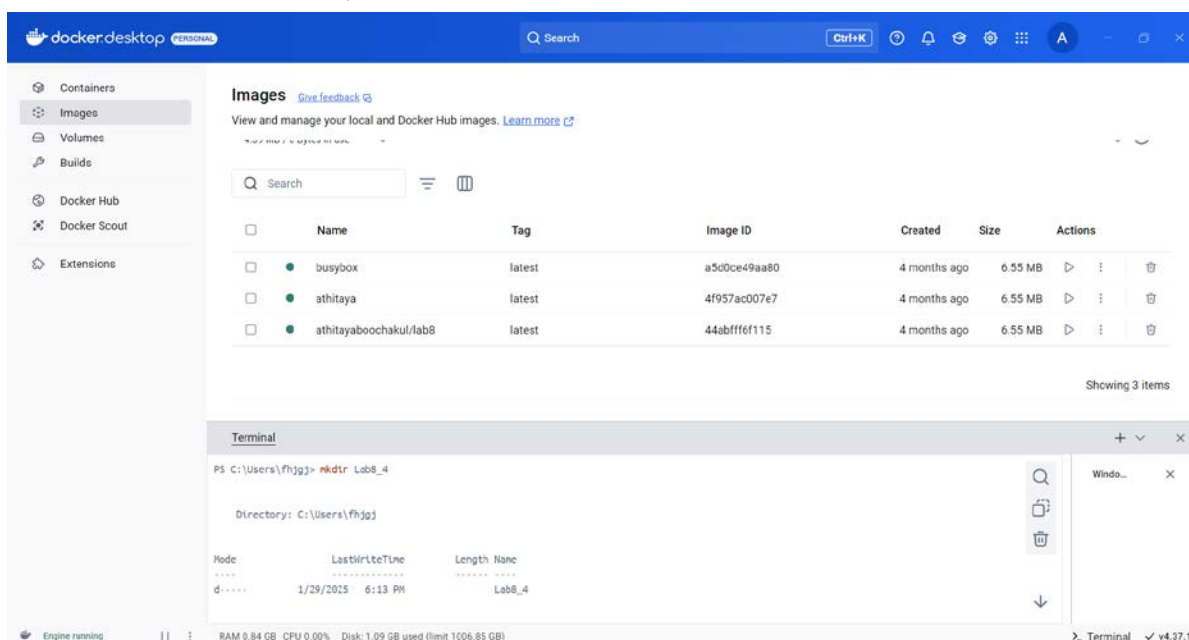


## Lab Worksheet

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

1. เปิด Command line หรือ Terminal จากนั้นสร้าง Directory ชื่อ Lab8\_4
2. ทำการ Clone ซอร์สโค้ดของเว็บแอปพลิเคชันจาก GitHub repository  
<https://github.com/docker/getting-started.git> ลงใน Directory ที่สร้างขึ้น โดยใช้คำสั่ง  
`$ git clone https://github.com/docker/getting-started.git`
3. เปิดดูองค์ประกอบภายใน getting-started/app เมื่อพบไฟล์ package.json ให้ใช้ Text editor ในการเปิดอ่าน

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





## Lab Worksheet

**Screenshot 1: Docker Desktop - Images**

View and manage your local and Docker Hub images. [Learn more](#)

Search

| <input type="checkbox"/> | Name                   | Tag    | Image ID     | Created      | Size    | Actions |
|--------------------------|------------------------|--------|--------------|--------------|---------|---------|
| <input type="checkbox"/> | busybox                | latest | a5d0ce49aa80 | 4 months ago | 6.55 MB |         |
| <input type="checkbox"/> | athitaya               | latest | 4f957ac007e7 | 4 months ago | 6.55 MB |         |
| <input type="checkbox"/> | athitayaboochakul/lab8 | latest | 44abfff6f115 | 4 months ago | 6.55 MB |         |

Showing 3 items

**Terminal**

```
PS C:\Users\fhjgg\Lab8_4> git clone https://github.com/docker/getting-started.git
Cloning into 'getting-started'...
remote: Enumerating objects: 980, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 980 (delta 5), reused 1 (delta 1), pack-reused 971 (from 2)
Receiving objects: 100% (980/980), 5.28 MiB | 2.04 MiB/s, done.
Resolving deltas: 100% (523/523), done.
PS C:\Users\fhjgg\Lab8_4>
```

Resource Saver mode | RAM 0.84 GB | CPU 0.00% | Disk: 1.09 GB used (limit 1006.85 GB) | Terminal v4.37.1

**Screenshot 2: Docker Desktop - Images**

View and manage your local and Docker Hub images. [Learn more](#)

Search

| <input type="checkbox"/> | Name                   | Tag    | Image ID     | Created      | Size    | Actions |
|--------------------------|------------------------|--------|--------------|--------------|---------|---------|
| <input type="checkbox"/> | busybox                | latest | a5d0ce49aa80 | 4 months ago | 6.55 MB |         |
| <input type="checkbox"/> | athitaya               | latest | 4f957ac007e7 | 4 months ago | 6.55 MB |         |
| <input type="checkbox"/> | athitayaboochakul/lab8 | latest | 44abfff6f115 | 4 months ago | 6.55 MB |         |

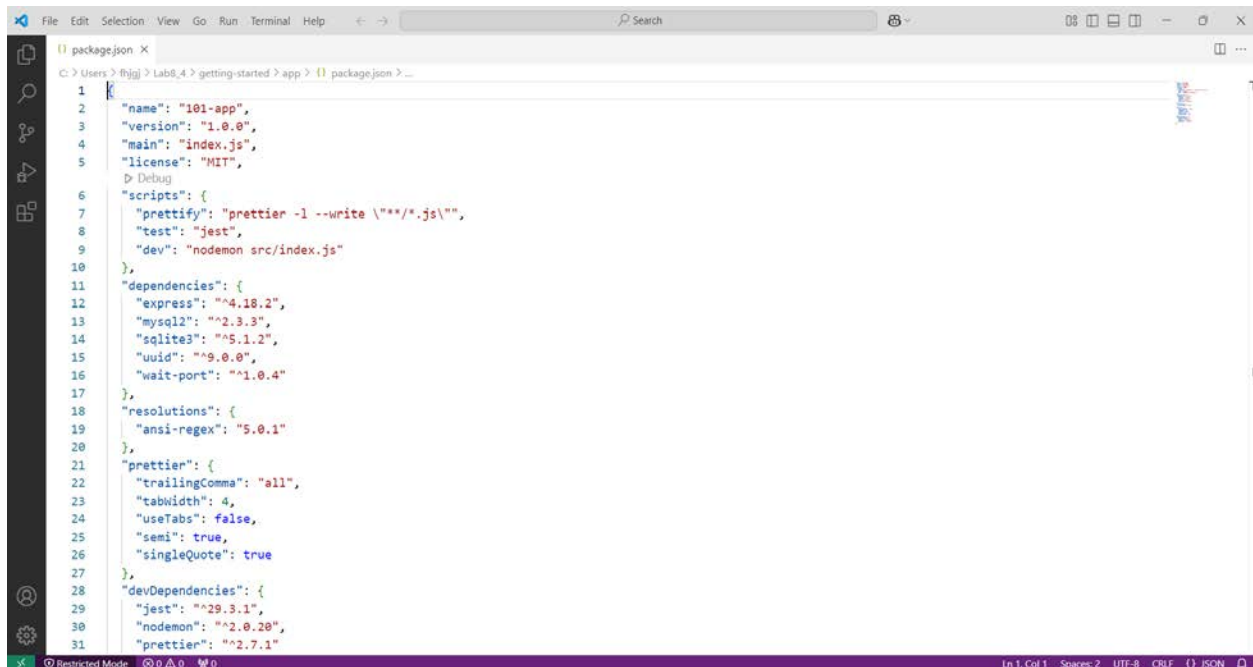
Showing 3 items

**Terminal**

```
Cloning into 'getting-started'...
remote: Enumerating objects: 980, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 980 (delta 5), reused 1 (delta 1), pack-reused 971 (from 2)
Receiving objects: 100% (980/980), 5.28 MiB | 2.04 MiB/s, done.
Resolving deltas: 100% (523/523), done.
PS C:\Users\fhjgg\Lab8_4> cd getting-started/app
PS C:\Users\fhjgg\Lab8_4\getting-started\app> code package.json
PS C:\Users\fhjgg\Lab8_4\getting-started\app>
```

Resource Saver mode | RAM 0.84 GB | CPU 0.50% | Disk: 1.09 GB used (limit 1006.85 GB) | Terminal v4.37.1

## Lab Worksheet



```

1  {
2    "name": "101-app",
3    "version": "1.0.0",
4    "main": "index.js",
5    "license": "MIT",
6    "scripts": {
7      "prettify": "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"
  
```

4. ภายใต้ getting-started/app ให้สร้าง Dockerfile พร้อมกับใส่เนื้อหาดังต่อไปนี้ลงไปไฟล์
 

```

FROM node:18-alpine
WORKDIR /app
COPY . .
RUN yarn install --production
CMD ["node", "src/index.js"]
EXPOSE 3000
      
```
5. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้ โดยกำหนดใช้ชื่อ image เป็น myapp\_รหัสสนศ. ไม่มีขีด
 

```
$ docker build -t <myapp_รหัสสนศ. ไม่มีขีด> .
```

## Lab Worksheet

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

The screenshot shows the Docker Desktop interface. At the top, there's a text editor window titled 'Dockerfile' with the following content:

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

Below the editor is the 'Images' section, which displays a list of local Docker images:

| Name                   | Tag    | Image ID     | Created       | Size    | Actions |
|------------------------|--------|--------------|---------------|---------|---------|
| busybox                | latest | a5d0ce49aa80 | 4 months ago  | 6.55 MB | [Icons] |
| athitaya               | latest | 4f957ac007e7 | 4 months ago  | 6.55 MB | [Icons] |
| athitayaboochakul/lab8 | latest | 44abfff6f115 | 4 months ago  | 6.55 MB | [Icons] |
| myapp_653380218-5      | latest | 9500ac3d182c | 6 seconds ago | 342 MB  | [Icons] |

Below the images list is a terminal window showing the output of the 'docker build' command:

```
>> exporting manifest sha256:608ba469d97a7131e9d753f93c35ebbae4de5ba318dbcc87eab4f2e4766a2666
>> exporting config sha256:36c1553f9c2b9cf9d8e1b0d40be7468d81c3d94a6d256768d390f034722a9781
>> exporting attestation manifest sha256:35514aa6f46d86c20022c228ae459df5d4a3c0b03cd692f371f785ed7dda48b
>> exporting manifest list sha256:9500ac3d182cdec6201e0d975639701f56c5d0d1cf20d5076a9480d74c72cb58
>> naming to docker.io/library/myapp_653380218-5:latest
>> unpacking to docker.io/library/myapp_653380218-5:latest
```

The terminal also shows the command 'docker build -t myapp\_653380218-5 .' and the resulting image ID '9500ac3d182c'.

6. ทำการ Start ตัว Container ของแอปพลิเคชันที่สร้างขึ้น โดยใช้คำสั่ง

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

## Lab Worksheet

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

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

The screenshot shows the Docker Desktop interface. The left sidebar contains navigation options: Containers, Images (selected), Volumes, Builds, Docker Hub, Docker Scout, and Extensions. The main area displays the 'Images' tab with a table of local images:

| Name                   | Tag    | Image ID     | Created       | Size    | Actions                |
|------------------------|--------|--------------|---------------|---------|------------------------|
| busybox                | latest | a5d0ce49aa80 | 4 months ago  | 6.55 MB | [Play] [More] [Delete] |
| athitaya               | latest | 4f957ac007e7 | 4 months ago  | 6.55 MB | [Play] [More] [Delete] |
| athitayaboochakul/lab8 | latest | 44abff6f115  | 4 months ago  | 6.55 MB | [Play] [More] [Delete] |
| myapp_653380218-5      | latest | 9500ac3d182c | 5 minutes ago | 342 MB  | [Play] [More] [Delete] |

Below the table, there are 'Walkthroughs' for 'How do I run a container?' and 'Run Docker Hub images'. A terminal window is open at the bottom, showing the command to run the container:

```
PS C:\Users\fhjgj\Lab8_4\getting-started\app> docker run -dp 3000:3000 myapp_653380218-5
09a854ace9ca9edfc48bdc6ba28f62b68c07f03a926b0f68f436df67743d4068
PS C:\Users\fhjgj\Lab8_4\getting-started\app>
```

The bottom of the image shows a web browser window at [localhost:3000](http://localhost:3000) displaying a 'New Item' form with an 'Add Item' button. The message 'No items yet! Add one above!' is shown below the form.

หมายเหตุ: นศ.สามารถทดลองเล่น Web application ที่ทำงานอยู่ได้

## Lab Worksheet

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

```

14 function TodoListCard() {
30   const onItemUpdate = React.useCallback(
31     item => {
38     },
39     [items],
40   );
41
42   const onItemRemoval = React.useCallback(
43     item => {
44       const index = items.findIndex(i => i.id === item.id);
45       setItems([...items.slice(0, index), ...items.slice(index + 1)]);
46     },
47     [items],
48   );
49
50   if (items === null) return 'Loading...';
51
52   return (
53     <React.Fragment>
54       <AddItemForm onNewItem={onNewItem} />
55       {items.length === 0 && (
56         <p className="text-center">There is no TODO item. Please add one to the list. By อธิชา นานะ</p>
57       )}
58       {items.map(item => (
59         <ItemDisplay
60           item={item}
61           key={item.id}
62           onItemUpdate={onItemUpdate}
63           onItemRemoval={onItemRemoval}
64         />
65       ))}
66     </React.Fragment>

```

## Lab Worksheet

The screenshot shows the Docker Desktop interface. The 'Images' tab is active, displaying a list of images:

| Name                   | Tag    | Image ID     | Created       | Size    | Actions |
|------------------------|--------|--------------|---------------|---------|---------|
| busybox                | latest | a5d0ce49aa80 | 4 months ago  | 6.55 MB | [Icons] |
| athitaya               | latest | 4f957ac007e7 | 4 months ago  | 6.55 MB | [Icons] |
| athitayaboochakul/lab8 | latest | 44abffff115  | 4 months ago  | 6.55 MB | [Icons] |
| myapp_653380218-5      | latest | ed9ab8a9701f | 6 seconds ago | 342 MB  | [Icons] |

Below the images list, there are 'Walkthroughs' and a 'Terminal' window. The terminal shows the following commands and output:

```

PS C:\Users\fhjgj\Lab8_4\getting-started\app> docker run -dp 3000:3000 myapp_653380218-5
8bbd3f314c2eb831737182fc7f10152774b7e781a9e01c69f2c25406bb962c
docker: Error response from daemon: driver failed programming external connectivity on endpoint trusting_culer (67e49da1fa3922200abec77b099494b29e9bc3402cc7681d9f9a92178c1fa9b5): Bind for 0.0.0.0:3000 failed: port is already allocated.
PS C:\Users\fhjgj\Lab8_4\getting-started\app>
  
```

The error message indicates that port 3000 is already allocated, preventing Docker from starting the container.

(1) Error ที่เกิดขึ้นหมายความว่าอย่างไร และเกิดขึ้นเพราะอะไร

Docker ไม่สามารถทำการเชื่อมต่อพอร์ต 3000 ที่เครื่องคุณกับคอนเทนเนอร์ได้ เนื่องจากพอร์ต 3000 ถูกใช้งานแล้วจากโปรแกรมหรือคอนเทนเนอร์อื่นๆ ที่รันอยู่

## Lab Worksheet

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

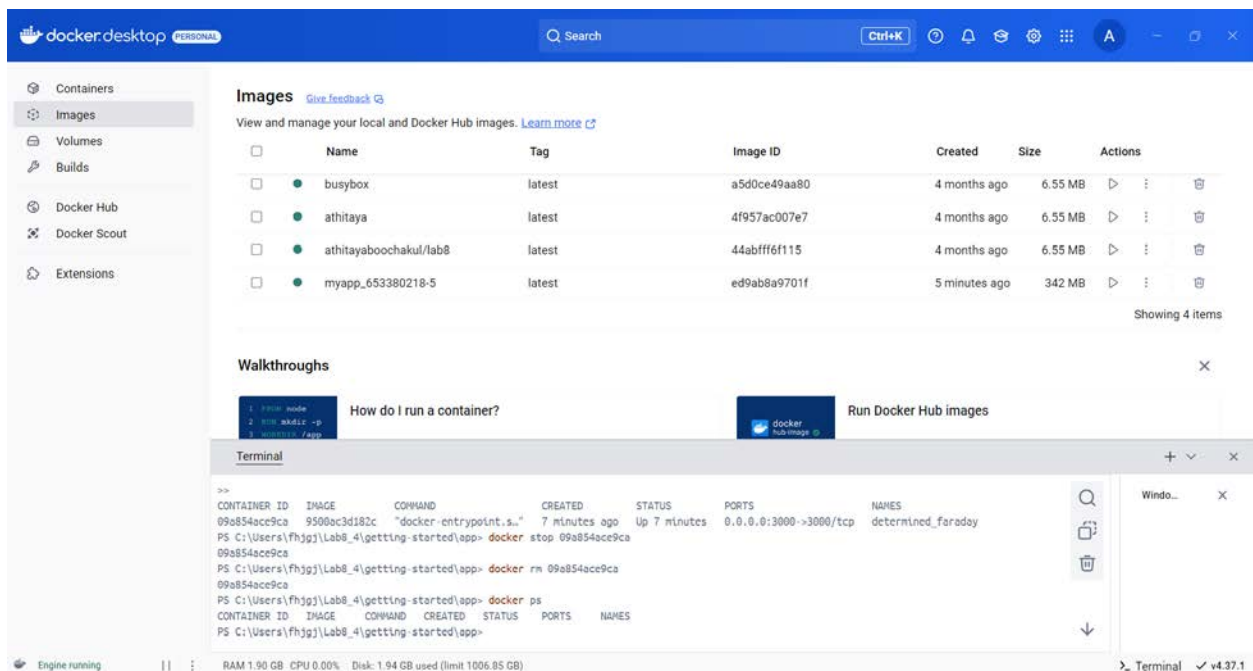
b. ผ่าน Docker desktop

- ไปที่หน้าต่าง Containers
- เลือกไอคอนถังขยะในแถวของ Container ที่ต้องการจะลบ
- ยืนยันโดยการกด Delete forever

12. Start และรัน Container ตัวใหม่อีกครั้ง โดยใช้คำสั่งเดียวกันกับข้อ 6

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

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





## Lab Worksheet

The screenshot displays the Docker Desktop application interface. The left sidebar contains navigation options: Containers, Images (selected), Volumes, Builds, Docker Hub, Docker Scout, and Extensions. The main area is divided into two sections: 'Images' and 'Walkthroughs'.

**Images Section:** A table lists local and Docker Hub images. The table has columns for Name, Tag, Image ID, Created, Size, and Actions.

| Name                   | Tag    | Image ID     | Created       | Size    | Actions                   |
|------------------------|--------|--------------|---------------|---------|---------------------------|
| busybox                | latest | a5d0ce49aa80 | 4 months ago  | 6.55 MB | [Play] [Refresh] [Delete] |
| athitaya               | latest | 4f957ac007e7 | 4 months ago  | 6.55 MB | [Play] [Refresh] [Delete] |
| athitayaboochakul/lab8 | latest | 44abfff6f115 | 4 months ago  | 6.55 MB | [Play] [Refresh] [Delete] |
| myapp_653380218-5      | latest | ed9ab8a9701f | 5 minutes ago | 342 MB  | [Play] [Refresh] [Delete] |

Showing 4 items

**Walkthroughs Section:** Two walkthroughs are listed: 'How do I run a container?' and 'Run Docker Hub images'.

**Terminal Section:** A terminal window shows the following commands and output:

```

09a854ace9ca 9506ac3d182c "docker-entrypoint.s..." 7 minutes ago Up 7 minutes 0.0.0.0:3000->3000/tcp determined_faraday
PS C:\Users\fhjgj\Lab8_4\getting-started\app> docker stop 09a854ace9ca
09a854ace9ca
PS C:\Users\fhjgj\Lab8_4\getting-started\app> docker rm 09a854ace9ca
09a854ace9ca
PS C:\Users\fhjgj\Lab8_4\getting-started\app> docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
PS C:\Users\fhjgj\Lab8_4\getting-started\app> docker run -dp 3000:3000 myapp_653380218-5
e3158da2e8f7a69b7c3d0dc264fae6cdecbe8cc14d56c98673343f6812c74c61
PS C:\Users\fhjgj\Lab8_4\getting-started\app>
  
```

The bottom status bar shows 'Engine running', 'RAM 1.94 GB', 'CPU 0.00%', and 'Disk: 1.83 GB used (limit 1006.85 GB)'. The bottom-most section shows a browser window with a 'Todo App' interface, displaying a 'New Item' input field, an 'Add Item' button, and a message: 'There is no TODO item. Please add one to the list. By ชิตสุธา บุญฤทธิ์'.

## 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
```

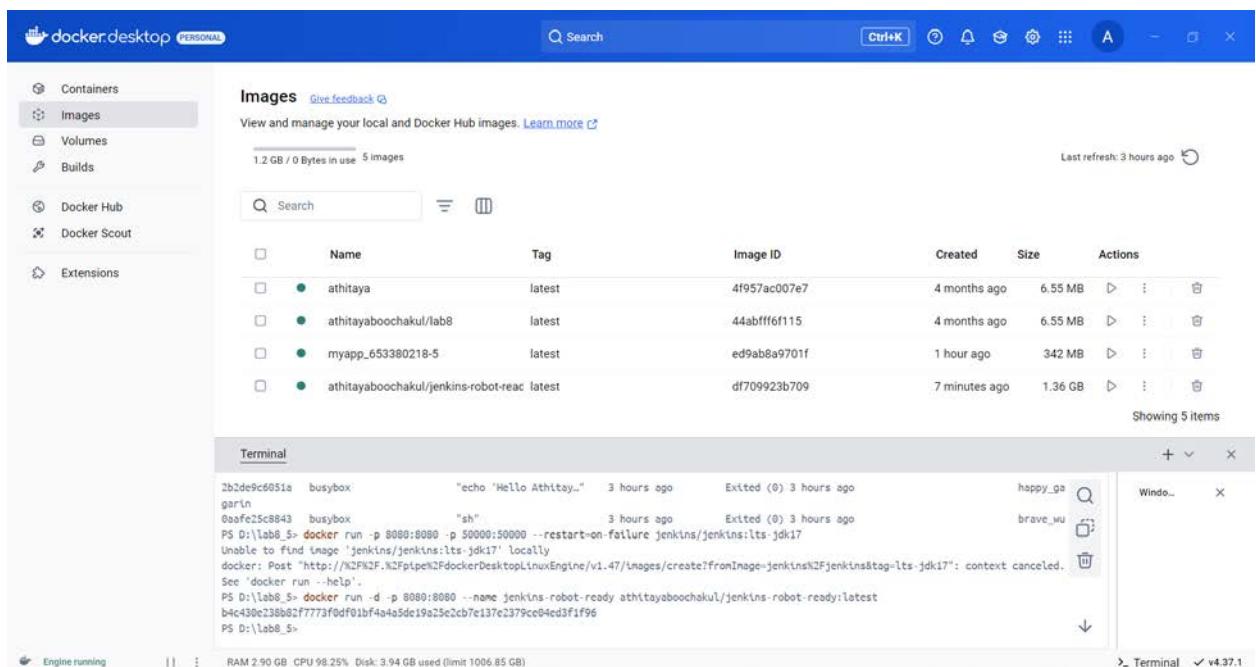
หรือ

```
$ 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

[Check point#12] Capture หน้าจอที่แสดงผล Admin password



b4c430e238b82f7773f0df01bf4a4a5de19a25e2cb7e137e2379ce04ed3f1f96

4. เมื่อได้รับการยืนยันว่า Jenkins is fully up and running ให้เปิดบราวเซอร์ และป้อนที่อยู่เป็น localhost:8080

5. ทำการ Unlock Jenkins ด้วยรหัสผ่านที่ได้ในข้อที่ 3

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

## Lab Worksheet

[Check point#13] Capture หน้าจอที่แสดงผลการตั้งค่า

The image displays two screenshots of the Jenkins installation process. The top screenshot shows the 'Create First Admin User' form, which includes fields for Username, Password, Confirm password, Full name, and E-mail address. The bottom screenshot shows the 'Instance Configuration' form, which includes a field for Jenkins URL.

**Getting Started**

### Create First Admin User

Username: athitaya\_2185

Password: [REDACTED]

Confirm password: [REDACTED]

Full name: athitaya boochakul

E-mail address: athitaya.bo@kkumail.com

Jenkins 2.479.3

Skip and continue as admin | **Save and Continue**

**Getting Started**

### Instance Configuration

Jenkins URL: http://localhost:8080/lab8

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.

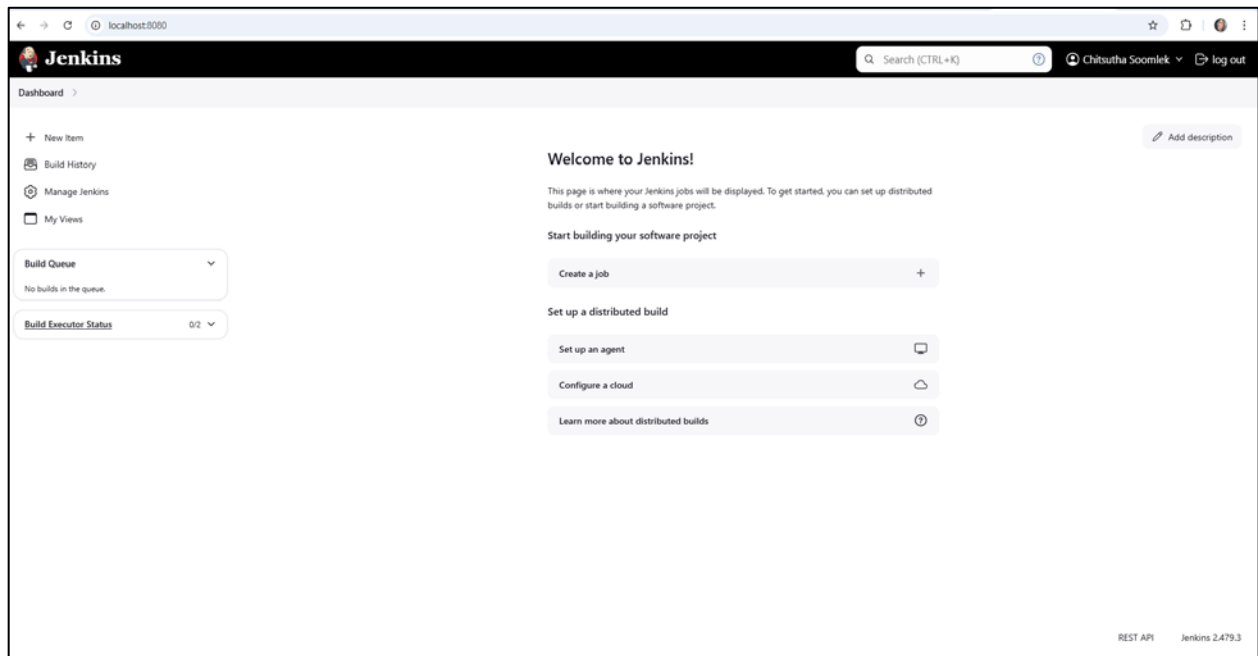
The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Jenkins 2.479.3

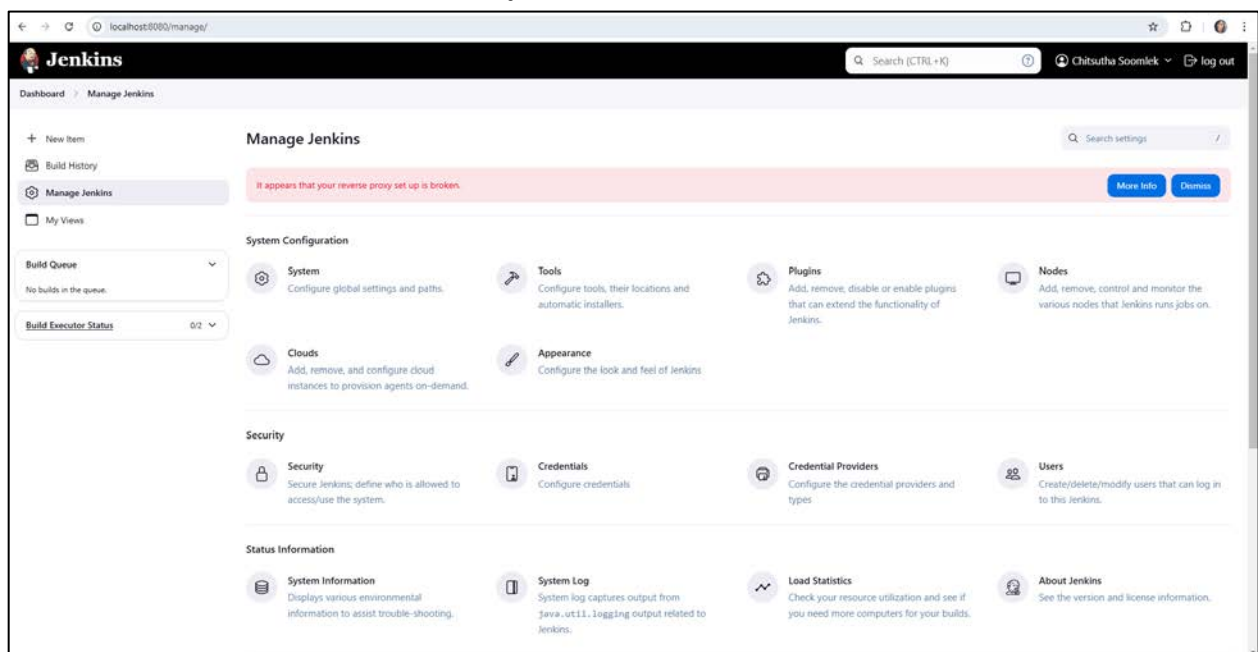
Not now | **Save and Finish**

- กำหนด Jenkins URL เป็น <http://localhost:8080/lab8>
- เมื่อติดตั้งเรียบร้อยแล้วจะพบหน้าจอ Dashboard ดังแสดงในภาพ

## Lab Worksheet

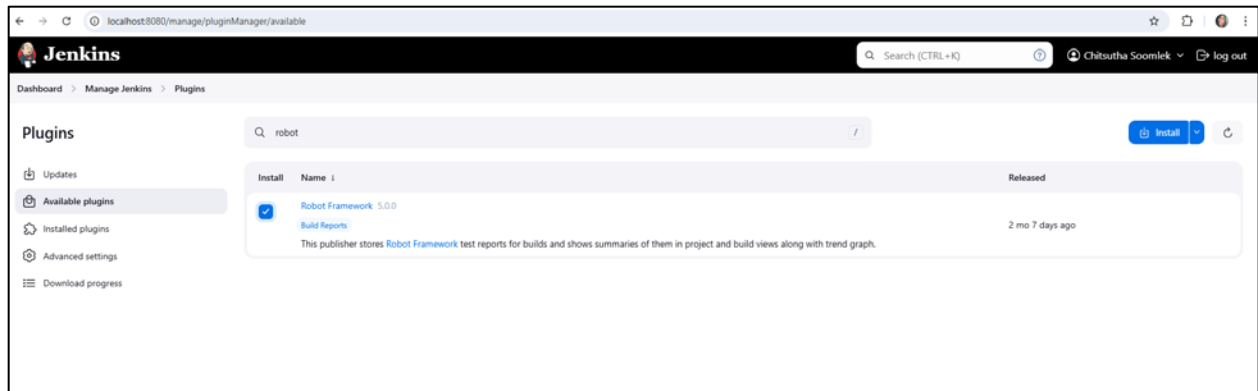


## 9. เลือก Manage Jenkins แล้วไปที่เมนู Plugins

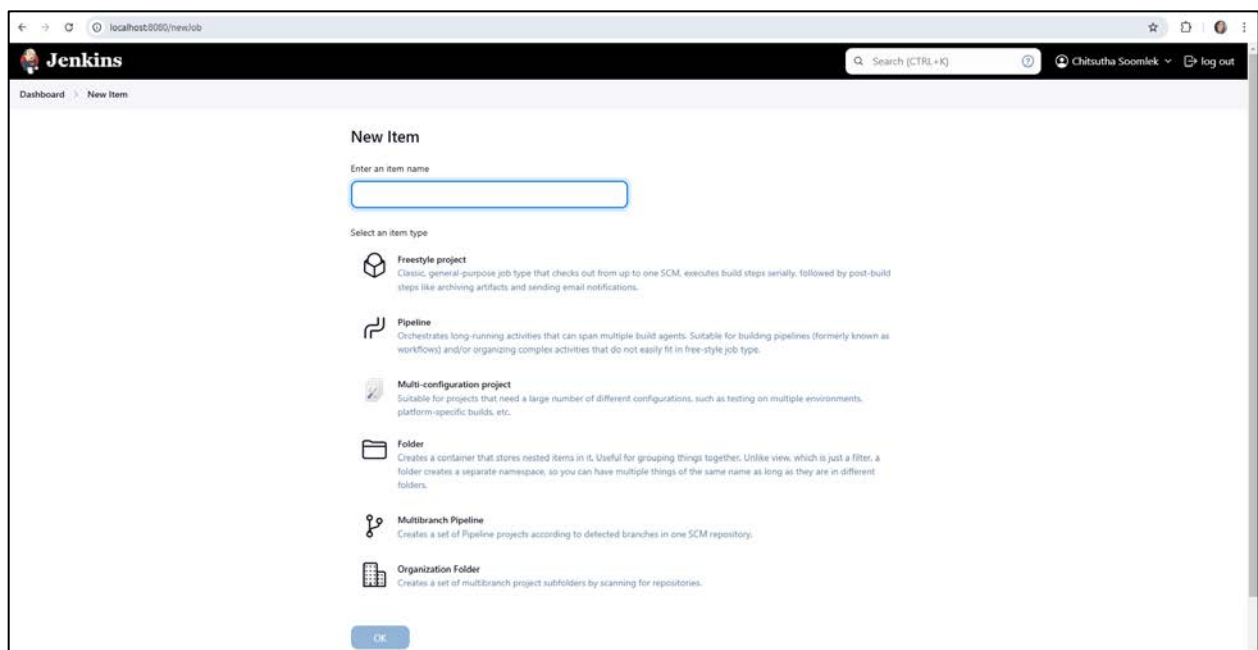


## Lab Worksheet

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

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

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

## Lab Worksheet

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

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

The image displays two screenshots of the Jenkins configuration interface. The top screenshot shows the 'General' tab, where the 'Description' is 'Lab 8.5'. Under 'Source Code Management', 'GitHub project' is selected. The 'Project url' is set to 'https://github.com/RubyOpal/jenkins/'. The bottom screenshot shows the 'Source Code Management' tab, where 'Git' is selected. The 'Repository URL' is 'https://github.com/RubyOpal/jenkins', and the 'Credentials' are set to 'none'. The 'Branches to build' section shows a 'Branch Specifier (blank for \'any\')' set to '\*/main'.

## Lab Worksheet

Dashboard > UAT > Configuration

### Configure

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

Build Triggers

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

Build after other projects are built ☐

☒ Build periodically ☐

Schedule

Would last have run at Wednesday, January 29, 2025 at 1:05:45 PM Coordinated Universal Time; would next run at Wednesday, January 29, 2025 at 1:20:45 PM Coordinated Universal Time.

GitHub hook trigger for GITScm polling ☐

Poll SCM ☐

Build Environment

Delete workspace before build starts ☐

Use secret text(s) or file(s) ☐

Add timestamps to the Console Output ☐

Inspect build log for published build scans ☐

Save Apply



## Lab Worksheet

The image shows two screenshots of the Jenkins configuration page for a job named 'UAT'. The top screenshot shows the 'Build Steps' section with a single step 'Execute shell' containing the command 'robot valid\_login.robot'. The bottom screenshot shows the 'Post-build Actions' section with a single action 'Publish Robot Framework test results' where the 'Directory of Robot output' is empty and the 'Thresholds for build result' are set to 20.0 for failed tests and 80.0 for passed tests. A checkbox for 'Include skipped tests in total count for thresholds' is checked.

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

robot login\_valid.robot

**Post-build action:** เพิ่ม Publish Robot Framework test results -> ระบุไดเรกทอรีที่เก็บไฟล์ผลการทดสอบโดย Robot framework ในรูป xml และ html -> ตั้งค่า Threshold เป็น % ของการทดสอบที่ไม่ผ่าน แล้วนับว่าซอฟต์แวร์มีปัญหา -> ตั้งค่า Threshold เป็น % ของการทดสอบที่ผ่านแล้วนับว่าซอฟต์แวร์มีอยู่ในสถานะที่สามารถนำไปใช้งานได้ (เช่น 20, 80)

13. กด Apply และ Save

## Lab Worksheet

## 14. สั่ง Build Now

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

The screenshot shows the Jenkins dashboard at localhost:8080. The main view displays the 'UAT' pipeline with a table of build history. The table has columns for Status (S), Name (I), Last Success, Last Failure, Last Duration, and Robot Results + Duration Trend. The first build is marked as failed (red X) and has a duration of 1.2 sec. Below the table, there are filters for 'Build Queue' (No builds in the queue) and 'Build Executor Status' (0/2).

The second screenshot shows the detailed view of build #2 (29 ม.ค. 2568 13:09:23). The left sidebar contains a menu with options: Status, Changes, Console Output, Edit Build Information, Delete build '#2', Timings, Git Build Data, Robot Results, and Next Build. The main content area shows the build details, including the user 'athitaya boochakul', the repository 'https://github.com/RubyOpal/jenkins', and the commit '0ef248be727e562827953ddd9064753a98d9eda'. The 'Robot Test Summary' table shows 1 failed test, 0 passed tests, and 0 skipped tests, resulting in a 0.0 pass percentage. The console output shows 'No changes.'

## Lab Worksheet

```
Dashboard > UAT > #2 > Console Output

First time build. Skipping changelog.
[UAT] $ /bin/sh -xe /tmp/jenkins17892819258229538432.sh
+ robot valid_login.robot
[ ERROR ] Error in file '/var/jenkins_home/workspace/UAT/valid_login.robot' on line 6: Resource file 'resource.robot' does not exist.
=====
Valid Login :: A test suite with a single test for valid login.
=====
Valid Login                                     | FAIL |
No keyword with name 'Open Browser To Login Page' found.

Also teardown failed:
No keyword with name 'Close Browser' found.
-----
Valid Login :: A test suite with a single test for valid login.      | FAIL |
1 test, 0 passed, 1 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
```