

## Lab Worksheet

ชื่อ-นามสกุล นางสาว ไอลินทร์ เมษะสิทธิโรจน์ รหัสนักศึกษา 653380158-7 Section 2

## Lab#8 – Software Deployment Using Docker

## วัตถุประสงค์การเรียนรู้

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

## Pre-requisite

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

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

1. เปิดใช้งาน Docker desktop และ Login ด้วย Username และ Password ที่ลงทะเบียนกับ Docker Hub เอาไว้
1. เปิด Command line หรือ Terminal บน Docker Desktop จากนั้นสร้าง Directory ชื่อ Lab8\_1
2. ย้ายตำแหน่งปัจจุบันไปที่ Lab8\_1 เพื่อใช้เป็น Working directory
3. ป้อนคำสั่ง \$ docker pull busybox หรือ \$ sudo docker pull busybox สำหรับกรณีที่ติดปัญหา Permission denied  
(หมายเหตุ: BusyBox เป็น software suite ที่รองรับคำสั่งบางอย่างบน Unix - <https://busybox.net>)
4. ป้อนคำสั่ง \$ docker images

## Lab Worksheet

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

```

Terminal
PS C:\Users\bibibrown> cd Lab8_1
PS C:\Users\bibibrown\Lab8_1> docker pull busybox
Using default tag: latest
latest: Pulling from library/busybox
9c0abc9c5bd3: Pull complete
Digest: sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138bd458fc8257fbf1
Status: Downloaded newer image for busybox:latest
docker.io/library/busybox:latest
PS C:\Users\bibibrown\Lab8_1> docker images
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
busybox              latest          af4709625109    4 months ago   4.27MB
docker/welcome-to-docker latest          c1f619b6477e    14 months ago  18.6MB
PS C:\Users\bibibrown\Lab8_1>

```

RAM 1.12 GB CPU 0.00% Disk: 1.11 GB used (limit 1006.85 GB)

(1) สิ่งที่อยู่ภายใต้คอลัมน์ Repository คืออะไร

ชื่อ docker image

(2) Tag ที่ใช้บ่งบอกถึงอะไร

เวอร์ชันของ docker

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

The image shows two screenshots of a Docker terminal window. The top screenshot shows the execution of `docker run busybox` and `docker run -it busybox sh`, followed by `ls` and `ls -la` commands, displaying the directory structure of the busybox container. The bottom screenshot shows the execution of `docker run busybox echo "Hello Irin Maysasittiroj from busybox"` and `docker ps -a`, displaying the output of the echo command and a list of all containers.

```

Terminal
docker/welcome-to-docker latest c1f619b6477e 14 months ago 18.6MB
PS C:\Users\bibibrown\Lab8_1> docker run busybox
PS C:\Users\bibibrown\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 28 10:49 .
drwxr-xr-x 1 root root 4096 Jan 28 10:49 ..
-rwxr-xr-x 1 root root 0 Jan 28 10:49 .dockerenv
drwxr-xr-x 2 root root 12288 Sep 26 21:31 bin
drwxr-xr-x 5 root root 360 Jan 28 10:49 dev
drwxr-xr-x 1 root root 4096 Jan 28 10:49 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 264 root root 0 Jan 28 10:49 proc
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

RAM 1.19 GB CPU 0.12% Disk: 1.11 GB used (limit 1006.85 GB)

Terminal
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\bibibrown\Lab8_1> docker run busybox echo "Hello Irin Maysasittiroj from busybox"
Hello Irin Maysasittiroj from busybox
PS C:\Users\bibibrown\Lab8_1> docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED              STATUS              PORTS
8d94172c42ff        busybox            "echo 'Hello Irin Ma..." 8 seconds ago       Exited (0) 7 seconds ago
450b3f886989        busybox            "sh"                    About a minute ago   Exited (0) 45 seconds ago
126f86137788        busybox            "sh"                    About a minute ago   Exited (0) About a minute ago
f045aba3bbc3        docker/welcome-to-docker:latest "/docker-entrypoint..." 6 days ago          Exited (255) 29 hours ago
0.0.0.0:8088->80/tcp welcome-to-docker
PS C:\Users\bibibrown\Lab8_1>

RAM 1.21 GB CPU 0.12% Disk: 1.11 GB used (limit 1006.85 GB)

```

- (1) เมื่อใช้ option `-it` ในคำสั่ง `run` ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป
- รัน container และเปิด shell ทำให้รันคำสั่งใน busy box
- I คือ interactive input
  - T คือ แสดงผลแบบ terminal

## Lab Worksheet

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

แสดงสถานะการทำงานของ container

Exited คือ หยุดทำงานแล้ว

Up คือ กำลังทำงาน

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

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

The screenshot shows a terminal window with the following content:

```

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\bibibrown\Lab8_1> docker run busybox echo "Hello Irin Maysasittiroj from busybox"
Hello Irin Maysasittiroj from busybox
PS C:\Users\bibibrown\Lab8_1> docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS
PORTS         NAMES
8d94172c42ff   busybox        "echo 'Hello Irin Ma..." 8 seconds ago   Exited (0) 7 seconds ago
450b3f886989   busybox        "sh"                     About a minute ago   Exited (0) 45 seconds ago
126f86137788   busybox        "sh"                     About a minute ago   Exited (0) About a minute ago
f045aba3bbc3   docker/welcome-to-docker:latest "/docker-entrypoint..." 6 days ago       Exited (255) 29 hours ago
0.0.0.0:8088->80/tcp   welcome-to-docker
PS C:\Users\bibibrown\Lab8_1> ^C
PS C:\Users\bibibrown\Lab8_1> docker rm 8d94172c42ff
8d94172c42ff
PS C:\Users\bibibrown\Lab8_1>

```

At the bottom of the terminal, system resources are displayed: RAM 0.81 GB, CPU 0.00%, and Disk: 1.11 GB used (limit 1006.85 GB).

## แบบฝึกปฏิบัติที่ 8.2: สร้าง Docker file และ Docker image

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

สำหรับเครื่องที่ใช้ระบบปฏิบัติการวินโดวส์ (Windows) บันทึกคำสั่งต่อไปนี้ลงในไฟล์ โดยใช้ Text Editor ที่มี

## Lab Worksheet

FROM busybox

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

CMD echo "ชื่อ-นามสกุล รหัสนักศึกษา ชื่อเล่น"

สำหรับเครื่องที่ใช้ระบบปฏิบัติการ MacOS หรือ Linux บนหน้าต่าง Terminal และป้อนคำสั่งต่อไปนี้

\$ cat > Dockerfile << EOF

FROM busybox

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

CMD echo "ชื่อ-นามสกุล รหัสนักศึกษา ชื่อเล่น"

EOF

หรือใช้คำสั่ง

\$ touch Dockerfile

แล้วใช้ Text Editor ในการใส่เนื้อหาแทน

5. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้

\$ docker build -t <ชื่อ Image> .

6. เมื่อ Build สำเร็จแล้ว ให้ทำการรัน Docker image ที่สร้างขึ้นในขั้นตอนที่ 5

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

```

Terminal
8d94172c42ff
PS C:\Users\bibibrown\Lab8_1> cd ..
PS C:\Users\bibibrown> mkdir Lab8_2

Directory: C:\Users\bibibrown

Mode                LastWriteTime         Length Name
----                -
d-----          1/28/2025   5:58 PM             Lab8_2

PS C:\Users\bibibrown> cd Lab8_2
  
```

## Lab Worksheet

```

Terminal
PS C:\Users\bibibrown\Lab8_2> docker build -t firstdocker -f Dockerfile.swp .
[+] Building 0.1s (5/5) FINISHED
=> [internal] load build definition from Dockerfile.swp
=> => transferring dockerfile: 164B
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals
=> WARN: MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the same stage because only
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals
=> [internal] load metadata for docker.io/library/busybox:latest
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/1] FROM docker.io/library/busybox:latest
=> exporting to image
=> => exporting layers
=> => writing image sha256:26c11e14be8d7d3da760dd2115a441a197c7b1901f554d4c7a9e4cb27fb4d0a
=> => naming to docker.io/library/firstdocker

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/jod8vknqowymdr542v399p98i

RAM 1.00 GB CPU 0.00% Disk: 1.11 GB used (limit 1006.85 GB)

Terminal
View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/jod8vknqowymdr542v399p98i

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\bibibrown\Lab8_2> docker run
"docker run" requires at least 1 argument.
See 'docker run --help'.

Usage: docker run [OPTIONS] IMAGE [COMMAND] [ARG...]

Create and run a new container from an image
PS C:\Users\bibibrown\Lab8_2> docker run firstdocker
"Irin Maysasittiroj 653380158-7 View"
PS C:\Users\bibibrown\Lab8_2>

RAM 1.00 GB CPU 0.00% Disk: 1.11 GB used (limit 1006.85 GB)

```

(1) คำสั่งที่ใช้ในการ run คือ

`docker run firstdocker`

(2) Option -t ในคำสั่ง \$ docker build ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป  
ตั้งชื่อให้กับ docker image ในที่นี้ คือชื่อ firstdocker

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

1. เปิดใช้งาน Docker desktop และ Login ด้วย Username และ Password ที่ลงทะเบียนกับ Docker Hub เอาไว้

## Lab Worksheet

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 และบันทึกคำสั่งต่อไปนี้

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

```
FROM busybox
```

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

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

```
EOF
```

หรือใช้คำสั่ง

```
$ touch Dockerfile
```

แล้วใช้ Text Editor ในการใส่เนื้อหาแทน

7. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้
 

```
$ docker build -t <username ที่ลงทะเบียนกับ Docker Hub>/lab8
```
5. ทำการรัน Docker image บน Container ในเครื่องของตัวเองเพื่อทดสอบผลลัพธ์ ด้วยคำสั่ง
 

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

## Lab Worksheet

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

```

Terminal
PS C:\Users\bibibrown> mkdir Lab8_3

Directory: C:\Users\bibibrown

Mode                LastWriteTime         Length Name
----                -
d-----          1/28/2025   6:12 PM             Lab8_3

PS C:\Users\bibibrown> cd Lab8_3
PS C:\Users\bibibrown\Lab8_3> docker build -t rinnin
ERROR: "docker buildx build" requires exactly 1 argument.
See 'docker buildx build --help'.

Usage:  docker buildx build [OPTIONS] PATH | URL | -

Start a build
PS C:\Users\bibibrown\Lab8_3> docker build -t rinnin/lab8

RAM 0.99 GB  CPU 0.12%  Disk: 1.11 GB used (limit 1006.85 GB)

Terminal
PS C:\Users\bibibrown\Lab8_3> docker build -t rinnin/lab8 -f Dockerfile.swp .
[+] Building 0.1s (5/5) FINISHED                                docker:desktop-linux
=> [internal] load build definition from Dockerfile.swp         0.0s
=> => transferring dockerfile: 186B                             0.0s
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals 0.0s
=> WARN: MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the same stage because only 0.0s
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals 0.0s
=> [internal] load metadata for docker.io/library/busybox:latest 0.0s
=> [internal] load .dockerignore                                0.0s
=> => transferring context: 2B                                    0.0s
=> CACHED [1/1] FROM docker.io/library/busybox:latest          0.0s
=> exporting to image                                           0.0s
=> => exporting layers                                           0.0s
=> => writing image sha256:29346de5f23f8ecfc6a035f0724fa71223e0f6034283b8c86e5e9d5ceebbe64a 0.0s
=> => naming to docker.io/rinnin/lab8                          0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/qekza91ub3yv1czmxybxro1b

3 warnings found (use docker --debug to expand):
- JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 2)

RAM 1.01 GB  CPU 0.00%  Disk: 1.11 GB used (limit 1006.85 GB)

```



## Lab Worksheet

```

Terminal
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals 0.0s
=> [internal] load metadata for docker.io/library/busybox:latest 0.0s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> CACHED [1/1] FROM docker.io/library/busybox:latest 0.0s
=> exporting to image 0.0s
=> => exporting layers 0.0s
=> => writing image sha256:29346de5f23f8ecfc6a035f0724fa71223e0f6034283b8c86e5e9d5ceebbe64a 0.0s
=> => naming to docker.io/rinnin/lab8 0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/gekza91ub3yv1czmixybxr01b

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\bibibrown\Lab8_3> docker run rinnin/lab8
"Irin Maysasittiroj 653380158-7"
PS C:\Users\bibibrown\Lab8_3>

RAM 1.01 GB CPU 0.00% Disk: 1.11 GB used (limit 1006.85 GB)

```

6. ทำการ Push ตัว Docker image ไปไว้บน Docker Hub โดยการให้คำสั่ง

\$ 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

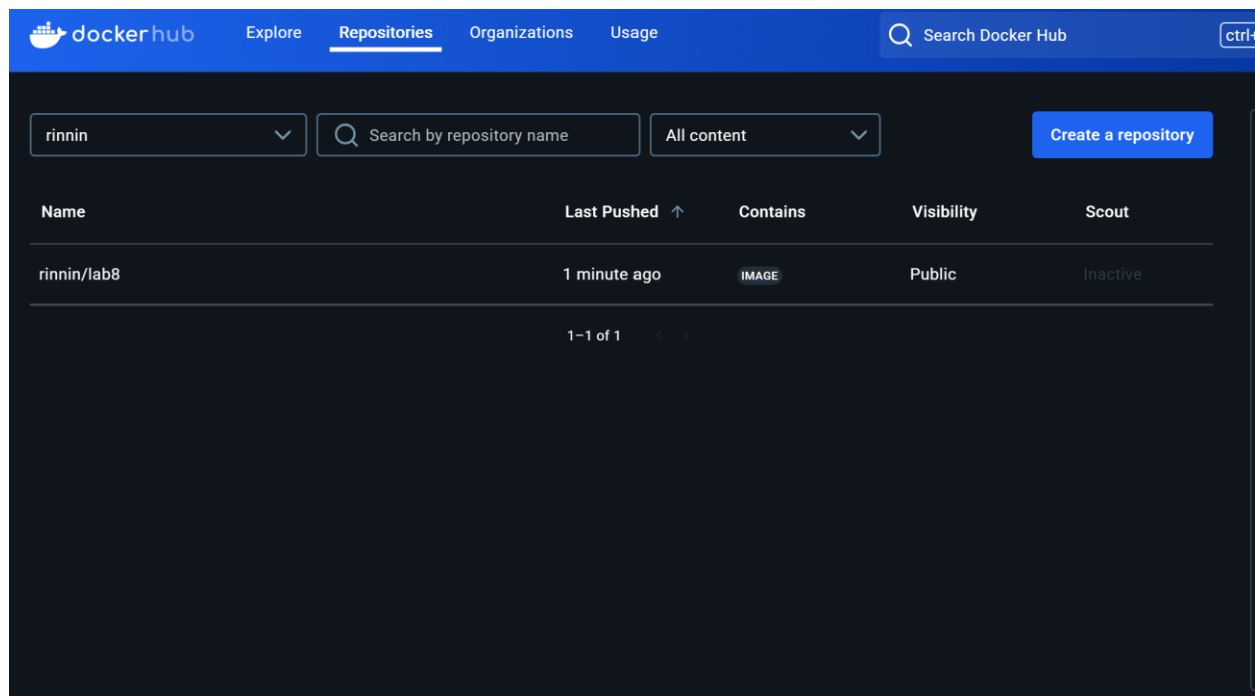
[Check point#6] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดง Repository ที่มี Docker image (<username>/lab8)

```

PS C:\Users\bibibrown\Lab8_3> docker push rinnin/lab8
Using default tag: latest
The push refers to repository [docker.io/rinnin/lab8]
59654b79daad: Pushed
latest: digest: sha256:6bbb91de4f5fb2e55f811ada5b4b39efa483e75b9faa158e4b5f878eb00ca382 size: 527
PS C:\Users\bibibrown\Lab8_3>

```

## 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  

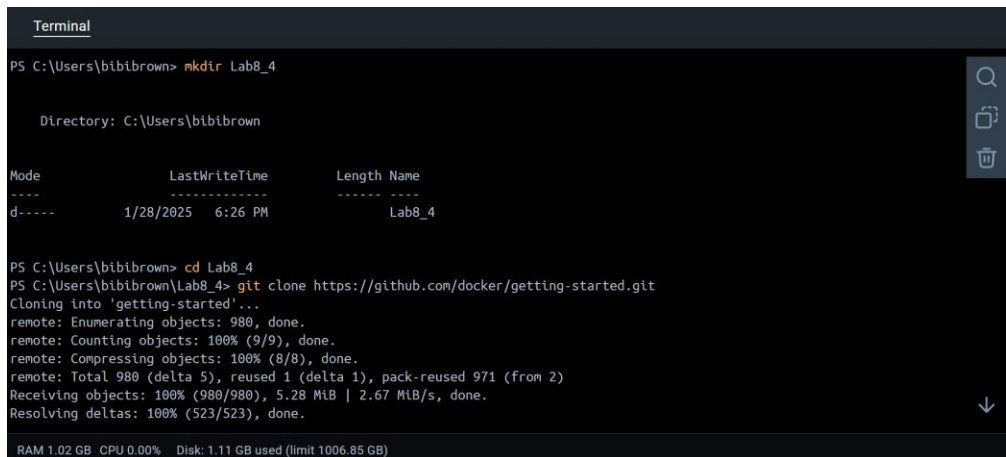
```
$ git clone https://github.com/docker/getting-started.git
```
3. เปิดดูองค์ประกอบภายใน getting-started/app เมื่อพบไฟล์ package.json ให้ใช้ Text editor ในการเปิดอ่าน

## Lab Worksheet

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

```
{} package.json X
C: > Users > bibibrown > Lab8_4 > getting-started > app > {} package.json > ...
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"
32   }
33 ]
34
```

## Lab Worksheet



```

Terminal
PS C:\Users\bibibrown> mkdir Lab8_4

Directory: C:\Users\bibibrown

Mode                LastWriteTime         Length Name
----                -
d-----          1/28/2025   6:26 PM             Lab8_4

PS C:\Users\bibibrown> cd Lab8_4
PS C:\Users\bibibrown\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 | 2.67 MiB/s, done.
Resolving deltas: 100% (523/523), done.

RAM 1.02 GB  CPU 0.00%  Disk: 1.11 GB used (limit 1006.85 GB)

```

4. ภายใต้ getting-started/app ให้สร้าง Dockerfile พร้อมกับใส่เนื้อหาดังต่อไปนี้ลงไปไฟล์

FROM node:18-alpine

WORKDIR /app

COPY . .

RUN yarn install --production

CMD ["node", "src/index.js"]

EXPOSE 3000

5. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้ โดยกำหนดใช้ชื่อ image เป็น

myapp\_รหัสนศ. ไม่มีขีด

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

## Lab Worksheet

[Check point#8] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง)

แสดงคำสั่งและผลลัพธ์ที่ได้ทางหน้าจอ

```
=> => transferring context: 2B 0.0s
=> [1/4] FROM docker.io/library/node:18-alpine@sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d975e9059dd066be3e274 0.0s
=> [internal] load build context 0.4s
=> => transferring context: 4.62MB 0.4s
=> CACHED [2/4] WORKDIR /app 0.0s
=> CACHED [3/4] COPY . . 0.0s
=> [4/4] RUN yarn install --production 47.6s
=> exporting to image 0.9s
=> => exporting layers 0.8s
=> => writing image sha256:eb02b4066648166a0e128fcef662d482211065c57c652e5c3dd41e3bd78d6218 0.0s
=> => naming to docker.io/library/myapp_6533801587 0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/gafph7icsfsednnu8o18sc5k7
PS C:\Users\bibibrown\Lab8_4\getting-started\app>
```

6. ทำการ Start ตัว Container ของแอปพลิเคชันที่สร้างขึ้น โดยใช้คำสั่ง

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

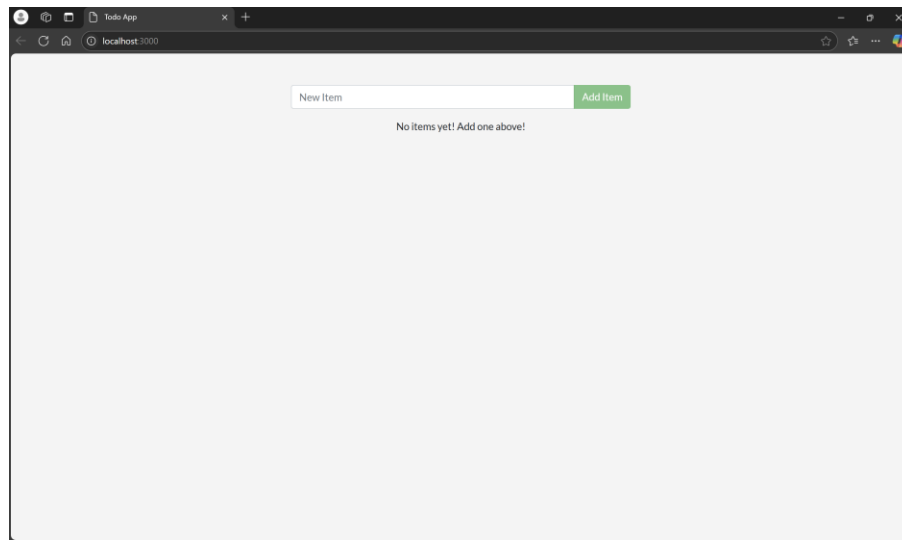
7. เปิด Browser ไปที่ URL = <http://localhost:3000>

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

และ Dashboard ของ Docker desktop

```
View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/gafph7icsfsednnu8o18sc5k7
PS C:\Users\bibibrown\Lab8_4\getting-started\app> docker run -dp 3000:3000 myapp_6533801587
32edc5e6b1a91a1e0463f0e1495cdaea61cd856e02ceec13f770367c69978a62
PS C:\Users\bibibrown\Lab8_4\getting-started\app>
```

## 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

Start และรัน Container ตัวใหม่ โดยใช้คำสั่ง

10. เดียวกันกับข้อ 6

## Lab Worksheet

[Check point#10] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง)

แสดงคำสั่งและผลลัพธ์ที่ได้ทางหน้าจอ พร้อมกับตอบคำถามต่อไปนี้

```
PS C:\Users\bibibrown\Lab8_4\getting-started\app> docker build -t myapp_6533801587 -f Dockerfile.swp .
[+] Building 31.0s (10/10) FINISHED                                docker:desktop-linux
=> [internal] load build definition from Dockerfile.swp          0.0s
=> => transferring dockerfile: 160B                             0.0s
=> [internal] load metadata for docker.io/library/node:18-alpine 2.7s
=> [auth] library/node:pull token for registry-1.docker.io      0.0s
=> [internal] load .dockerignore                                 0.0s
=> => transferring context: 2B                                    0.0s
=> [1/4] FROM docker.io/library/node:18-alpine@sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d975e9059dd066be3e274 0.0s
=> [internal] load build context                                0.0s
=> => transferring context: 8.10kB                               0.0s
=> CACHED [2/4] WORKDIR /app                                    0.0s
=> [3/4] COPY . .                                              0.1s
=> [4/4] RUN yarn install --production                         27.1s
=> exporting to image                                           1.0s
=> => exporting layers                                           1.0s
=> => writing image sha256:f3db7e7b0326f60875f56ec9adaeaececbabc090fc0ca3b9134b3783387a7f5 0.0s
=> => naming to docker.io/library/myapp_6533801587             0.0s
```

View build details: [docker-desktop://dashboard/build/desktop-linux/desktop-linux/nlp766dx9cealpf2t13500v6m](https://docker-desktop://dashboard/build/desktop-linux/desktop-linux/nlp766dx9cealpf2t13500v6m)

```
PS C:\Users\bibibrown\Lab8_4\getting-started\app>
```

```
JS app.js X
C: > Users > bibibrown > Lab8_4 > getting-started > app > src > static > js > JS app.js > TodoListCard > items.map() callback
14 function TodoListCard() {
42   const onItemRemoval = React.useCallback(
43     item => {
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 Irin Maysasittiroj</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>
67   );
68
69   function AddItemForm({ onNewItem }) {
70     const { Form, InputGroup, Button } = ReactBootstrap;
71
72     const [newItem, setNewItem] = React.useState('');
73     const [submitting, setSubmitting] = React.useState(false);
74
75     const submitNewItem = e => {
76       e.preventDefault();
77       setSubmitting(true);
78       fetch('/items', {
79         method: 'POST',
80         body: JSON.stringify({ name: newItem })
81       });
82     };
83   }
84 }
```

```
PS C:\Users\bibibrown\Lab8_4\getting-started\app\src\static\js> docker run -dp 3000:3000 myapp_6533801587
```

```
2844c77fe984e8fc458984fbf4bf2d603ccb4e4ccf4f550e9a368a6afd2b900d
docker: Error response from daemon: driver failed programming external connectivity on endpoint fervent_ride (863d0037e768ede52ed3382387a73e171bc9f3c45378e12760bae76c52e9920b): Bind for 0.0.0.0:3000 failed: port is already allocated.
```

```
PS C:\Users\bibibrown\Lab8_4\getting-started\app\src\static\js>
```

## Lab Worksheet

(1) Error ที่เกิดขึ้นหมายความว่าอย่างไร และเกิดขึ้นเพราะอะไร

Docker ต้องการตั้งค่าการเชื่อมต่อระหว่าง container และ host แต่ล้มเหลว และพยายามผูกพอร์ต 3000 บน host แต่พอร์ตนี้ถูกใช้งานอยู่แล้ว

11. ลบ Container ของ Web application เวอร์ชันก่อนแก้ไขออกจากระบบ โดยใช้วิธีใดวิธีหนึ่งดังต่อไปนี้

a. ผ่าน Command line interface

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

b. ผ่าน Docker desktop

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

12. Start และรัน Container ตัวใหม่อีกครั้ง โดยใช้คำสั่งเดียวกันกับข้อ 6

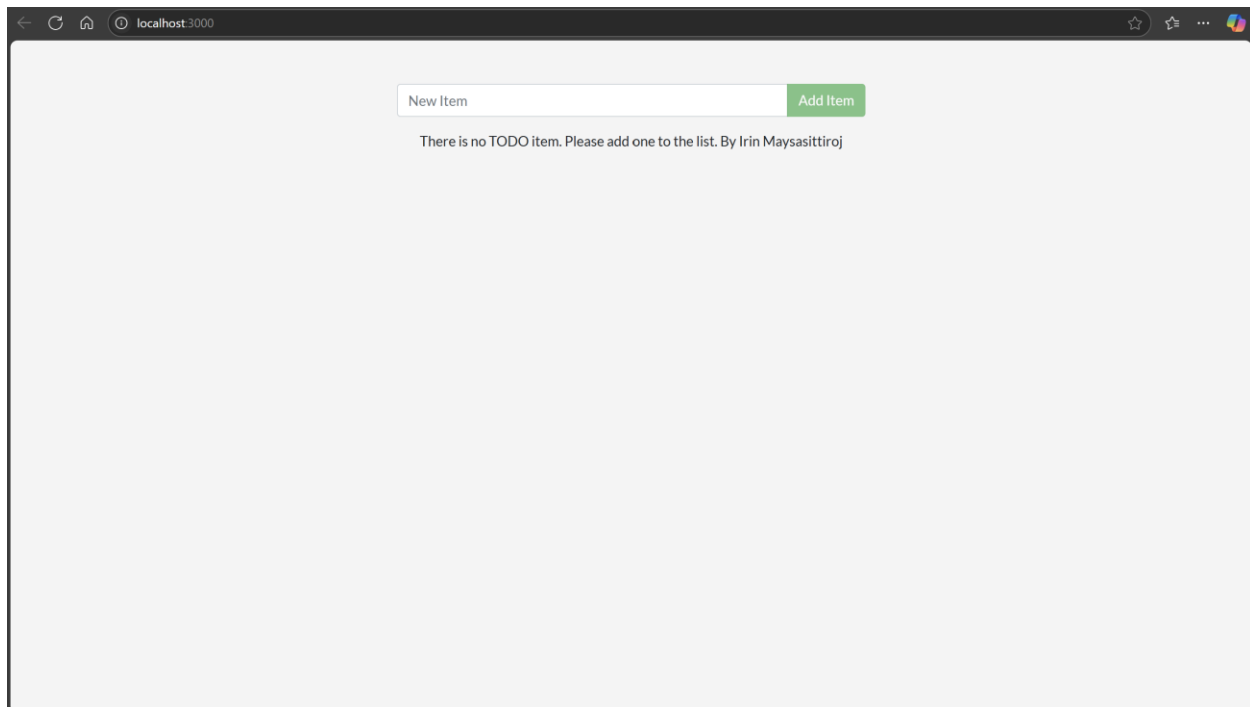
13. เปิด Browser ไปที่ URL = <http://localhost:3000>

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

```
PS C:\Users\bibibrown\Lab8_4\getting-started\app\src\static\js> docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                    NAMES
32edc5e6b1a9   eb02b4066648   "docker-entrypoint.s..." 8 minutes ago   Up 8 minutes   0.0.0.0:3000->3000/tcp   determined
_bass1
PS C:\Users\bibibrown\Lab8_4\getting-started\app\src\static\js> ^C
PS C:\Users\bibibrown\Lab8_4\getting-started\app\src\static\js> docker stop 32edc5e6b1a9
32edc5e6b1a9
PS C:\Users\bibibrown\Lab8_4\getting-started\app\src\static\js> docker rm 32edc5e6b1a9
32edc5e6b1a9
PS C:\Users\bibibrown\Lab8_4\getting-started\app\src\static\js> docker run -dp 3000:3000 myapp_6533801587
82c68fb67b00da68c8b0a46ece4dcedcd85e3707e9004bbcdfb2c73c8da0f14b
PS C:\Users\bibibrown\Lab8_4\getting-started\app\src\static\js>
```



## 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

```
Jenkins initial setup is required. An admin user has been created and a password generated.
Please use the following password to proceed to installation:

d9b6a4201bf54f26bb3d5ee73482071e

This may also be found at: /var/jenkins_home/secrets/initialAdminPassword

*****
*****
*****

2025-01-28 12:33:01.964+0000 [id=66] INFO jenkins.InitReactorRunner$1#onAttained: Completed initialization
```

## Lab Worksheet

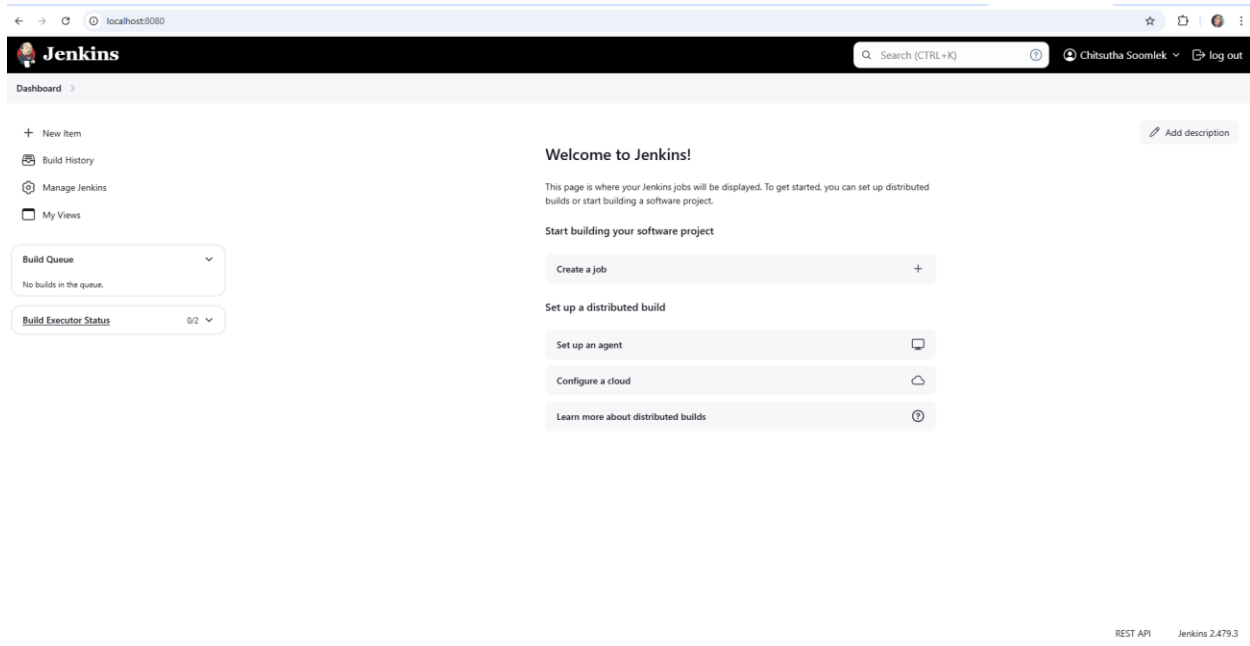
- เมื่อได้รับการยืนยันว่า Jenkins is fully up and running ให้เปิดบราวเซอร์ และป้อนที่อยู่เป็น localhost:8080
- ทำการ Unlock Jenkins ด้วยรหัสผ่านที่ได้ในข้อที่ 3
- สร้าง Admin User โดยใช้ username เป็นชื่อจริงของนักศึกษาพร้อมรหัสสี่ตัวท้าย เช่น somsri\_3062

[Check point#13] Capture หน้าจอที่แสดงผลการตั้งค่า

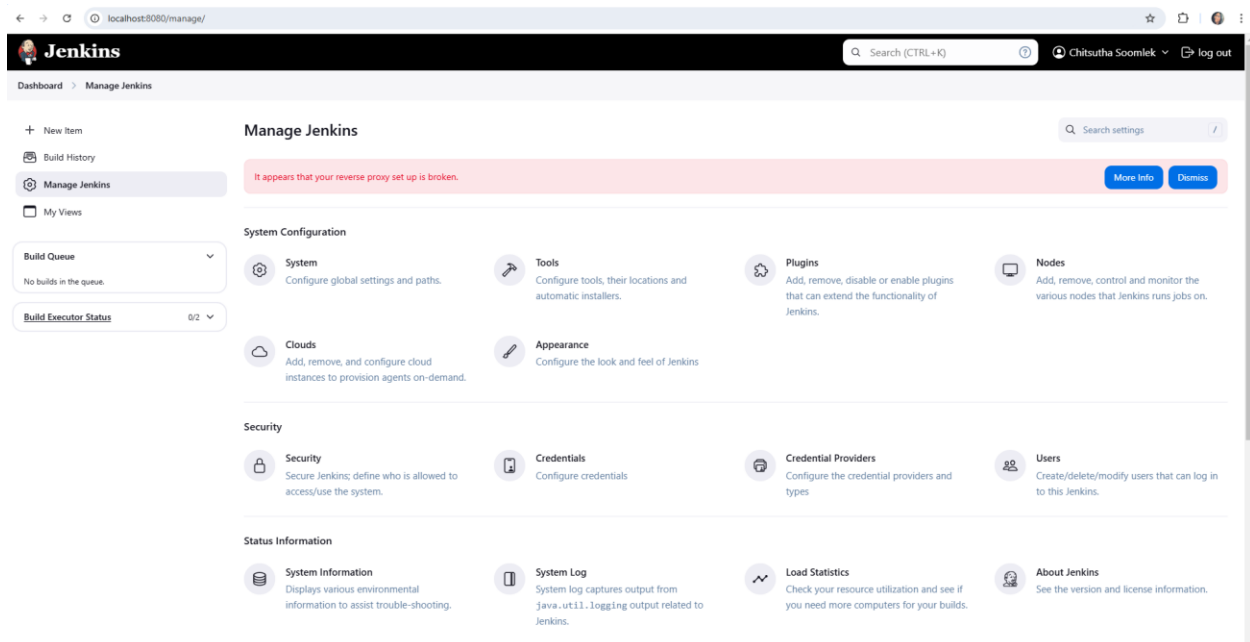
The image displays two screenshots of the Jenkins Setup Wizard interface. The top screenshot shows the 'Create First Admin User' form, which includes fields for Username (irin\_1587), Password, Confirm password, Full name (irin maysasittiroj), and E-mail address (irin.m@kkumail.com). The bottom screenshot shows the 'Instance Configuration' form, where the Jenkins URL is set to http://localhost:8080/lab8. Both screenshots are taken from a web browser window with the address bar showing localhost:8080.

## Lab Worksheet

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



9. เลือก Manage Jenkins แล้วไปที่เมนู Plugins

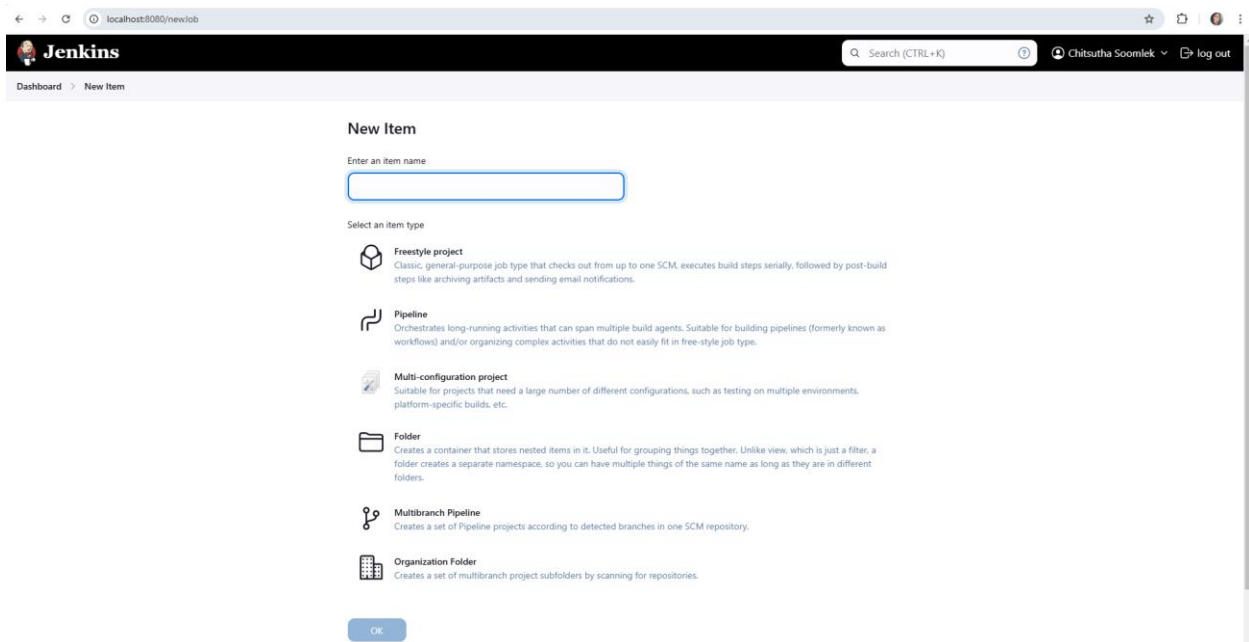


## Lab Worksheet

10. ไปที่เมนู Available plugins แล้วเลือกติดตั้ง Robotframework เพิ่มเติม



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



12. นำไฟล์ .robot ที่ทำให้แบบฝึกปฏิบัติที่ 7 (Lab#7) ไปไว้บน Repository ของนักศึกษา จากนั้นตั้งค่าที่จำเป็นในหน้านี้ทั้งหมด ดังนี้

Description: Lab 8.5

## Lab Worksheet

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

Build Trigger: เลือกแบบ Build periodically แล้วกำหนดให้ build ทุก 15 นาที

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

[Check point#14] Capture หน้าจอแสดงการตั้งค่า พร้อมกับตอบคำถามต่อไปนี้

The screenshot shows the 'General' tab of a GitHub Actions workflow configuration. The workflow is named 'Lab 8.5' and is currently 'Enabled'. The 'Description' field is empty. Under 'Plain text', there is a 'Preview' link. The 'Discard old builds' checkbox is unchecked. The 'GitHub project' checkbox is checked, and the 'Project url' is set to 'https://github.com/Bibibrown/Lab7\_SoftwareEngineer.git'. The 'Advanced' dropdown is visible. The 'This project is parameterized' and 'Throttle builds' checkboxes are unchecked. The 'Build Triggers' section shows 'Trigger builds remotely (e.g., from scripts)', 'Build after other projects are built', and 'Build periodically' (checked). The 'Schedule' is set to 'H/15 \* \* \* \*'. A note indicates the workflow would last have run on Tuesday, January 28, 2025 at 12:50:21 PM Coordinated Universal Time, and would next run at 1:05:21 PM Coordinated Universal Time. The 'GitHub hook trigger for GITScm polling' and 'Poll SCM' checkboxes are unchecked.

**General** Enabled ☒

Description

Lab 8.5

Plain text [Preview](#)

☐ Discard old builds ?

☒ GitHub project

Project url ?

[https://github.com/Bibibrown/Lab7\\_SoftwareEngineer.git](https://github.com/Bibibrown/Lab7_SoftwareEngineer.git)

Advanced ▾

☐ This project is parameterized ?

☐ Throttle builds ?

**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 Tuesday, January 28, 2025 at 12:50:21 PM Coordinated Universal Time; would next run at Tuesday, January 28, 2025 at 1:05:21 PM Coordinated Universal Time.

☐ GitHub hook trigger for GITScm polling ?

☐ Poll SCM ?

## Lab Worksheet

Build Steps

Execute shell ?

Command

See the list of available environment variables

robot UAT\_Lab7\_001.robot UAT\_Lab7\_002.robot

Advanced ▾

Add build step ▾

Publish Robot Framework test results ?

Directory of Robot output

Path to directory containing robot xml and html files (relative to build workspace)

Advanced ▾

Thresholds for build result ?

%

20

%

80

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

☐ Include skipped tests in total count for thresholds

Add post-build action ▾

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

robot UAT\_Lab7\_001.robot UAT\_Lab7\_002.robot

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

ระบุไดเรกทอรีที่เก็บไฟล์ผลการทดสอบโดย Robot framework ในรูป xml และ html -> ตั้งค่า Threshold เป็น % ของการทดสอบที่ไม่ผ่านแล้วนับว่าซอฟต์แวร์มีปัญหา -> 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 image shows two screenshots of the Jenkins dashboard for a pipeline named 'UAT' (Lab 8.5). The top screenshot shows the 'Robot Framework Tests Trend (all tests)' chart, which displays a bar chart with a red bar representing 'Failed' tests. The chart shows a trend of test cases over time. The bottom screenshot shows the 'Latest Robot Results' section, which displays a table of test results. The table has columns for 'Total', 'Failed', 'Passed', 'Skipped', and 'Pass %'. The data shows 7 failed tests, 0 passed tests, 0 skipped tests, and a 0.0% pass rate. The 'Permalinks' section lists links to the last build, last failed build, last unsuccessful build, and last completed build, all of which are 5.7 seconds ago.

**Jenkins Dashboard - UAT Pipeline**

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

Build	Number of test cases	Status
Build	6	Failed

**Latest Robot Results:**

Total	Failed	Passed	Skipped	Pass %
All tests	7	0	0	0.0

**Permalinks:**

- Last build (#1), 5.7 sec ago
- Last failed build (#1), 5.7 sec ago
- Last unsuccessful build (#1), 5.7 sec ago
- Last completed build (#1), 5.7 sec ago

## Lab Worksheet

