

## Lab Worksheet

ชื่อ-นามสกุล นาย เศรษฐ์ เสริฐกวี รหัสนักศึกษา 653380345-8 Section 4

## 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

```

PS C:\Users\ASUS> mkdir Lab8_1

Directory: C:\Users\ASUS

Mode                LastWriteTime         Length Name
----                -
d-----          1/29/2025   8:11 PM             Lab8_1

latest: Pulling from library/busybox
9c0abc9c5bd3: Download complete
Digest: sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138bd458fc8257fbf1
Status: Downloaded newer image for busybox:latest
docker.io/library/busybox:latest

latest: Pulling from library/busybox
9c0abc9c5bd3: Download complete
Digest: sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138bd458fc8257fbf1
Status: Downloaded newer image for busybox:latest
docker.io/library/busybox:latest
PS C:\Users\ASUS\Lab8_1> docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
busybox       latest   a5d0ce49aa80   4 months ago   6.56MB

```

(1) สิ่งที่อยู่ภายใต้คอลัมน์ Repository คืออะไร

ตอบ คือ ชื่อของ Docker image ที่ถูกดึงเข้ามาหรือถูกสร้างขึ้นภายในระบบ เช่น busybox, ubuntu, หรือ my\_custom\_image เป็นต้น

(2) Tag ที่ใช้บ่งบอกถึงอะไร

ตอบ Tag ใช้บ่งบอกถึงเวอร์ชันของ Docker image ซึ่งช่วยให้สามารถดึงหรือใช้งานเวอร์ชันที่ต้องการได้ ตัวอย่างเช่น

- busybox:latest หมายถึงเวอร์ชันล่าสุดของ busybox
- ubuntu:20.04 หมายถึง Ubuntu เวอร์ชัน 20.04

(3) busybox:latest หมายถึงเวอร์ชันล่าสุดของ busybox

(4) ubuntu:20.04 หมายถึง Ubuntu เวอร์ชัน 20.04

## Lab Worksheet

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

```
PS C:\Users\ASUS\Lab8_1> docker run busybox
PS C:\Users\ASUS\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 29 13:20 .
drwxr-xr-x  1 root    root      4096 Jan 29 13:20 ..
-rwxr-xr-x  1 root    root        0 Jan 29 13:20 .dockerenv
drwxr-xr-x  2 root    root     12288 Sep 26 21:31 bin
drwxr-xr-x  5 root    root      360 Jan 29 13:20 dev
drwxr-xr-x  1 root    root      4096 Jan 29 13:20 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 244 root    root        0 Jan 29 13:20 proc
drwx----- 1 root    root     4096 Jan 29 13:20 root
dr-xr-xr-x 11 root    root        0 Jan 29 13:20 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\ASUS\Lab8_1> docker run busybox echo "Hello Seth Sertkawe from busybox"
Hello Seth Sertkawe from busybox
```

## Lab Worksheet

```
PS C:\Users\ASUS\Lab8_1> docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
NAMES					
4615a3565dd4	busybox	"echo 'Hello Seth Se..."	9 seconds ago	Exited (0) 8 seconds ago	
practical_matsumoto					
55e135ddf73a	busybox	"sh"	About a minute ago	Exited (0) About a minute ago	
boring_jepsen					
72dff597c821	busybox	"sh"	About a minute ago	Exited (0) About a minute ago	
objective_swartz					
52798601cd5a	busybox	"sh"	About a minute ago	Exited (0) About a minute ago	
nostalgic_roentgen					
8dc09b714a6c	busybox	"sh"	2 minutes ago	Exited (0) About a minute ago	
festive_blackburn					
10ed23213a3b	busybox	"sh"	2 minutes ago	Exited (0) 2 minutes ago	
hungry_chandrasekhar					

(1) เมื่อใช้ option -it ในคำสั่ง run ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป

ตอบ Option -it ประกอบด้วย:

-i (interactive) → ทำให้ Container ยังคงรอรับอินพุตจากผู้ใช้ แม้ว่าจะไม่มีการเชื่อมต่อกับ stdin

-t (tty - terminal) → สร้าง terminal จำลอง เพื่อให้สามารถโต้ตอบกับ Container ได้

ผลลัพธ์ที่ได้:

- สามารถพิมพ์คำสั่งภายใน Container ได้โดยตรง เช่น ls, cd, echo
- ใช้ Shell (sh, bash) ภายใน Container ได้

(2) คอลัมน์ STATUS จากการรันคำสั่ง docker ps -a แสดงถึงข้อมูลอะไร

ตอบ คอลัมน์ STATUS แสดง สถานะของ Container ว่าอยู่ในสถานะใด เช่น:

- Up X seconds/minutes → Container กำลังทำงานอยู่
- Exited (0) X seconds ago → Container รันเสร็จแล้วและหยุดทำงานโดยไม่มีข้อผิดพลาด (Exit Code 0)
- Exited (1) X seconds ago → Container หยุดทำงานเพราะเกิดข้อผิดพลาด (Exit Code 1 ขึ้นไป)

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

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

```
PS C:\Users\ASUS\Lab8_1> docker rm 10ed23213a3b
10ed23213a3b
```

## Lab Worksheet

## 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."
```

```
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

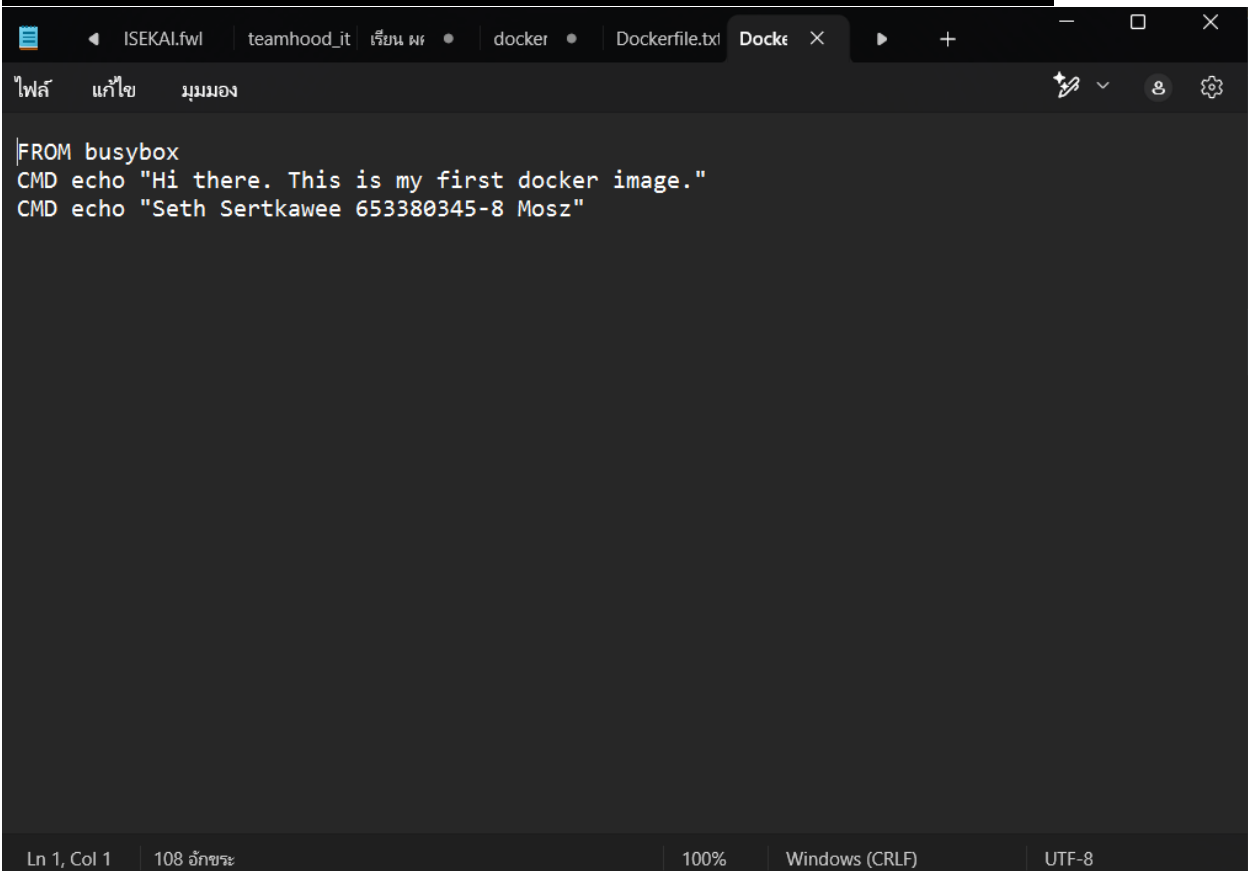
## Lab Worksheet

```
PS C:\Users\ASUS\Lab8_1> cd ..  
PS C:\Users\ASUS> mkdir Lab8_2
```

Directory: C:\Users\ASUS

Mode	LastWriteTime	Length	Name
d-----	1/29/2025 8:32 PM		Lab8_2

```
PS C:\Users\ASUS> cd Lab8_2  
PS C:\Users\ASUS\Lab8_2> notepad Dockerfile
```



The screenshot shows a Notepad application window with the title bar "Dockerfile.txt". The window contains the following text:

```
FROM busybox  
CMD echo "Hi there. This is my first docker image."  
CMD echo "Seth Sertkawee 653380345-8 Mosz"
```

The status bar at the bottom of the Notepad window displays "Ln 1, Col 1", "108 อักขระ", "100%", "Windows (CRLF)", and "UTF-8".

## Lab Worksheet

```

PS C:\Users\ASUS\Lab8_2> docker build -t my_first_image .
[+] Building 3.1s (6/6) FINISHED
    docker:desktop-linux
=> [internal] load build definition from Dockerfile
    0.0s
=> => transferring dockerfile: 148B
    0.0s
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 2)
    0.0s
=> WARN: MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the same stage because only the last one will be used (line 2)
    0.0s
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 3)
    0.0s
=> [internal] load metadata for docker.io/library/busybox:latest
    0.0s
=> [internal] load .dockerignore
    0.0s
=> => transferring context: 2B
    0.0s
=> [1/1] FROM docker.io/library/busybox:latest@sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138bd458fc8257fbf1
    2.8s
=> => resolve docker.io/library/busybox:latest@sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138bd458fc8257fbf1
    2.8s
=> [auth] library/busybox:pull token for registry-1.docker.io
    0.0s
=> exporting to image
    0.1s
=> => exporting layers
    0.0s
=> => exporting manifest sha256:0b44419e9c29d35eb1bee33cc92f7682f47a823064a64ee3678bf82f04ce63
    0.0s

View build details: docker_desktop://dashboard/build/desktop-linux/desktop-linux/0ulrdh5iv02e08h18nv4nyu5v

3 warnings found (use docker --debug to expand):
- MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the same stage because only the last one will be used (line 2)
)
3 warnings found (use docker --debug to expand):
- 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)
- JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 2)
- -----

PS C:\Users\ASUS\Lab8_2> docker run my_first_image
Seth Sertkawe 653380345-8 Mosz

```

(1) คำสั่งที่ใช้ในการ run คือ

ตอบ docker run my\_first\_image คำสั่งนี้ใช้เพื่อรัน Docker Container ที่สร้างจาก Image ชื่อ my\_first\_image

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

ตอบ Option -t ในคำสั่ง \$ docker build ใช้สำหรับกำหนดชื่อและแท็กให้กับ Docker image ที่สร้างขึ้น ทำให้สามารถอ้างอิงและเรียกใช้ได้ง่ายขึ้น หากไม่ใช้ -t Docker จะกำหนดค่าเริ่มต้นเป็น <none> ซึ่งอาจทำให้จัดการได้ยาก

## 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 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
```

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



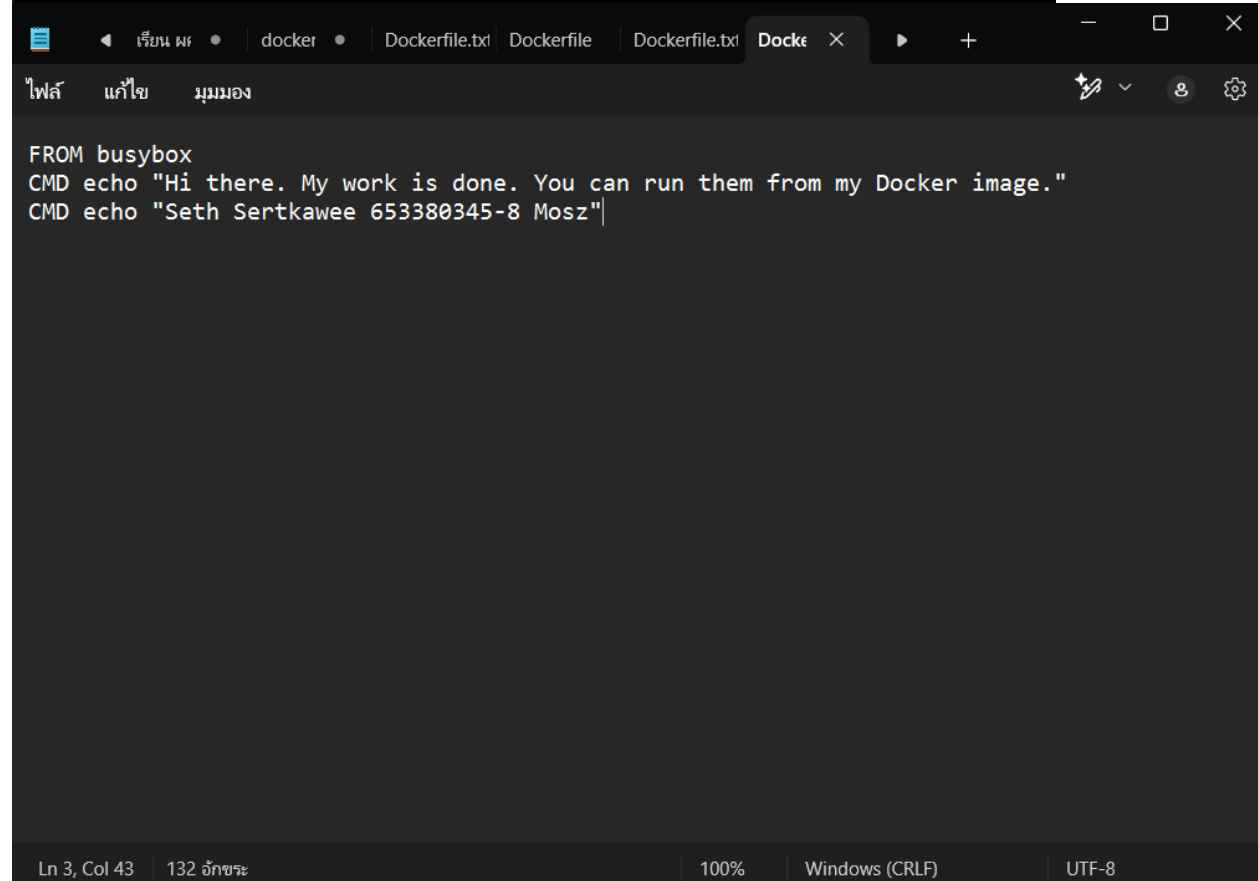
## Lab Worksheet

```
PS C:\Users\ASUS\Lab8_2> cd ..
PS C:\Users\ASUS> mkdir Lab8_3

Directory: C:\Users\ASUS

Mode                LastWriteTime         Length Name
----                -
d-----          1/29/2025   8:52 PM             Lab8_3

PS C:\Users\ASUS> cd Lab8_3
PS C:\Users\ASUS\Lab8_3> notepad Dockerfile
```



```
FROM busybox
CMD echo "Hi there. My work is done. You can run them from my Docker image."
CMD echo "Seth Sertkawe 653380345-8 Mosz"
```

Ln 3, Col 43 | 132 อักขระ | 100% | Windows (CRLF) | UTF-8

## Lab Worksheet

```

PS C:\Users\ASUS\Lab8_3> docker build -t seth2003/lab8 .
[+] Building 2.9s (6/6) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 173B
=> 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 b
=> 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@sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138bd458fc8257fbf1
=> => resolve docker.io/library/busybox:latest@sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138bd458fc8257fbf1
=> [auth] library/busybox:pull token for registry-1.docker.io
=> exporting to image
=> => exporting config sha256:559737b63bc2755e183fbae5b3821b501df971d90a9ed0ac8a00978e54719
=> => exporting attestation manifest sha256:1564f9cc5d375377a53b9e54d55ccc81f58e893138a4aa8810b4f7140e110b1f
=> => exporting manifest list sha256:8ef885c7b9a022aaed59509b846b62b6b4dd2b8abf9695f30d78f667fcaa2fde
=> => naming to docker.io/seth2003/lab8:latest
=> => unpacking to docker.io/seth2003/lab8:latest

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

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

PS C:\Users\ASUS\Lab8_3> docker run seth2003/lab8
Seth Sertkawee 653380345-8 Mosz

```

6. ทำการ Push ตัว Docker image ไปไว้บน Docker Hub โดยการใช้คำสั่ง

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

ในกรณีที่ติดปัญหาไม่ได้ Login ไว้ก่อน ให้ใช้คำสั่งต่อไปนี้ เพื่อ Login ก่อนทำการ Push

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

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

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

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

```

PS C:\Users\ASUS\Lab8_3> docker push seth2003/lab8
Using default tag: latest
The push refers to repository [docker.io/seth2003/lab8]
09a04ae90ab9: Pushed
9c0abc9c5bd3: Mounted from library/busybox
latest: digest: sha256:8ef885c7b9a022aaed59509b846b62b6b4dd2b8abf9695f30d78f667fcaa2fde size: 855

```

## Lab Worksheet

The screenshot shows the Docker Desktop interface. On the left is a sidebar with navigation options: Containers, Images, Volumes, Builds, Docker Hub, Docker Scout, and Extensions. The main area is divided into two sections. The top section, titled 'Search results for "seth2003/lab8\_3"', shows a search bar with the text 'seth2003/lab8\_3' and a single result card for 'seth2003/lab8'. The bottom section, titled 'Images', shows a list of local Docker images. The list has columns for Name, Tag, Image ID, Created, Size, and Actions. The images listed are 'busybox', 'my\_first\_image', and 'seth2003/lab8'. Below the list, there is a section for 'seth2003/lab8' showing 'No repositories found' and 'Recent Tags' with a 'LATEST' tag.

## 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

```
PS C:\Users\ASUS> mkdir Lab8_4
```

```
Directory: C:\Users\ASUS
```

Mode	LastWriteTime	Length	Name
d-----	1/29/2025 9:13 PM		Lab8_4

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

```
PS C:\Users\ASUS\Lab8_4>
```

```
PS C:\Users\ASUS\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 | 2.49 MiB/s, done.
```

```
Resolving deltas: 100% (523/523), done.
```

```
PS C:\Users\ASUS\Lab8_4>
```

```
PS C:\Users\ASUS\Lab8_4> cd getting-started/app
```

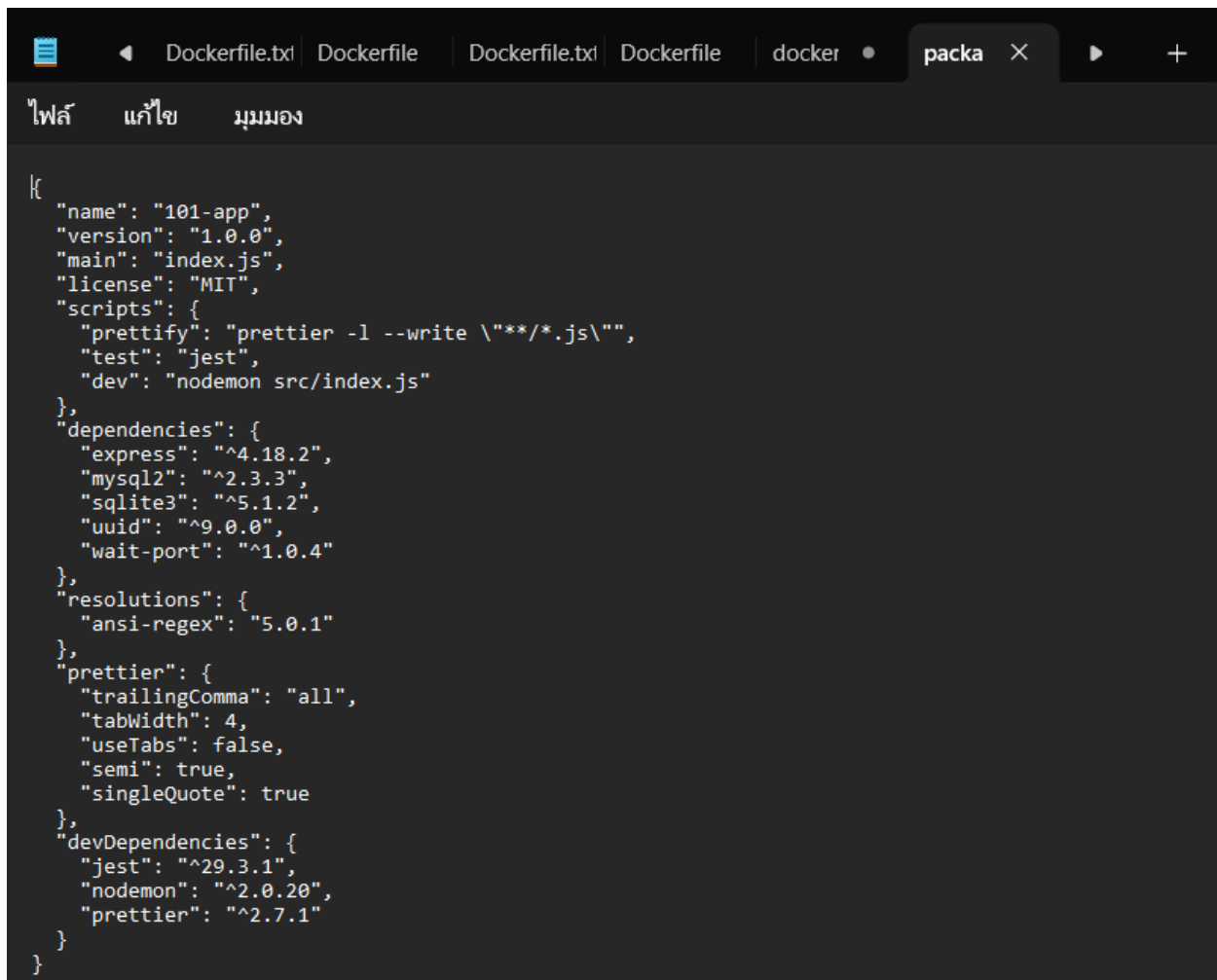
```
PS C:\Users\ASUS\Lab8_4\getting-started\app> ls
```

```
Directory: C:\Users\ASUS\Lab8_4\getting-started\app
```

Mode	LastWriteTime	Length	Name
d-----	1/29/2025 9:14 PM		spec
d-----	1/29/2025 9:14 PM		src
-a----	1/29/2025 9:14 PM	678	package.json
-a----	1/29/2025 9:14 PM	150541	yarn.lock

```
PS C:\Users\ASUS\Lab8_4\getting-started\app> notepad package.json
```

## Lab Worksheet



```

{
  "name": "101-app",
  "version": "1.0.0",
  "main": "index.js",
  "license": "MIT",
  "scripts": {
    "prettify": "prettier -l --write \"**/*.js\"",
    "test": "jest",
    "dev": "nodemon src/index.js"
  },
  "dependencies": {
    "express": "^4.18.2",
    "mysql2": "^2.3.3",
    "sqlite3": "^5.1.2",
    "uuid": "^9.0.0",
    "wait-port": "^1.0.4"
  },
  "resolutions": {
    "ansi-regex": "5.0.1"
  },
  "prettier": {
    "trailingComma": "all",
    "tabWidth": 4,
    "useTabs": false,
    "semi": true,
    "singleQuote": true
  },
  "devDependencies": {
    "jest": "^29.3.1",
    "nodemon": "^2.0.20",
    "prettier": "^2.7.1"
  }
}

```

4. ภายใต้ getting-started/app ให้สร้าง Dockerfile พร้อมกับใส่เนื้อหาดังต่อไปนี้ลงไปไฟล์
 

```

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

```
5. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้ โดยกำหนดใช้ชื่อ image เป็น myapp\_รหัสนศ. ไม่มีขีด

## Lab Worksheet

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

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

```

PS C:\Users\ASUS\Lab8_4\getting-started\app> notepad Dockerfile
FROM node:18-alpine
WORKDIR /app
COPY . .
RUN yarn install --production
CMD ["node", "src/index.js"]
EXPOSE 3000

PS C:\Users\ASUS\Lab8_4\getting-started\app> docker build -t myapp_6533803458 .
[+] Building 82.8s (10/10) FINISHED                                docker:desktop-linux
=> [internal] load build definition from Dockerfile                0.0s
=> => transferring dockerfile: 154B                               0.0s
=> [internal] load metadata for docker.io/library/node:18-alpine  4.3s
=> [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@sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d975e9059 19.8s
=> => resolve docker.io/library/node:18-alpine@sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d975e9059d 0.0s
=> => sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15bf426d01bc1 1.26MB / 1.26MB 0.9s
=> => sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce368990ca771 444B / 444B 0.9s
=> => sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e0da02525e63167c 40.01MB / 40.01MB 18.8s
=> => sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c24403d592134f0 3.64MB / 3.64MB 3.8s
=> => extracting sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c24403d592134f0 0.1s
=> => extracting sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c24403d592134f0 0.1s
=> => extracting sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e0da02525e63167c 0.8s
=> => extracting sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15bf426d01bc1 0.0s
=> => extracting sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce368990ca771 0.0s
=> [internal] load build context                                  0.4s
=> => transferring context: 4.62MB                                  0.4s
=> [2/4] WORKDIR /app                                           0.4s
=> [3/4] COPY . .                                              0.1s
=> [4/4] RUN yarn install --production                          53.4s
=> exporting to image                                           4.7s
=> => exporting layers                                           3.1s
=> => exporting manifest sha256:41de31a485e88763857bbb23b54a44d99f30bb58b44b6e42d35de359a86ccdbe 0.0s
=> => exporting config sha256:27e46888d5c5c84cc2c76e0b344f78cfb96282e92680e4c837758c4b692f136b 0.0s
=> => exporting attestation manifest sha256:61016078b718ec29018994dc4c8818261fda1adbfcff5ee70d3b8d96bc 0.0s
=> => exporting manifest list sha256:b89f0c0d089e9a29b09f787efb6e4b843a7ffa43eede00647359ffc498833522 0.0s
  
```

## Lab Worksheet

```
=> => exporting layers 3.1s
=> => exporting manifest sha256:41de31a485e88763857bbb23b54a44d99f30bb58b44b6e42d35de359a86ccdbe 0.0s
=> => exporting config sha256:27e46888d5c5c84cc2c76e0b344f78cfb96282e92680e4c837758c4b692f136b 0.0s
=> => exporting attestation manifest sha256:61016078b718ec29018994dc4c8818261fda1adbfccff5ee70d3b8d96bc 0.0s
=> => exporting manifest list sha256:b89f0c0d089e9a29b09f787efb6e4b843a7ffa43eede00647359ffc498833522 0.0s
=> => naming to docker.io/library/myapp_6533803458:latest 0.0s
=> => unpacking to docker.io/library/myapp_6533803458:latest 1.5s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/jo3a8wo1wplgdfs7dts7u03v4
PS C:\Users\ASUS\Lab8_4\getting-started\app>
```

6. ทำการ Start ตัว Container ของแอปพลิเคชันที่สร้างขึ้น โดยใช้คำสั่ง

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

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

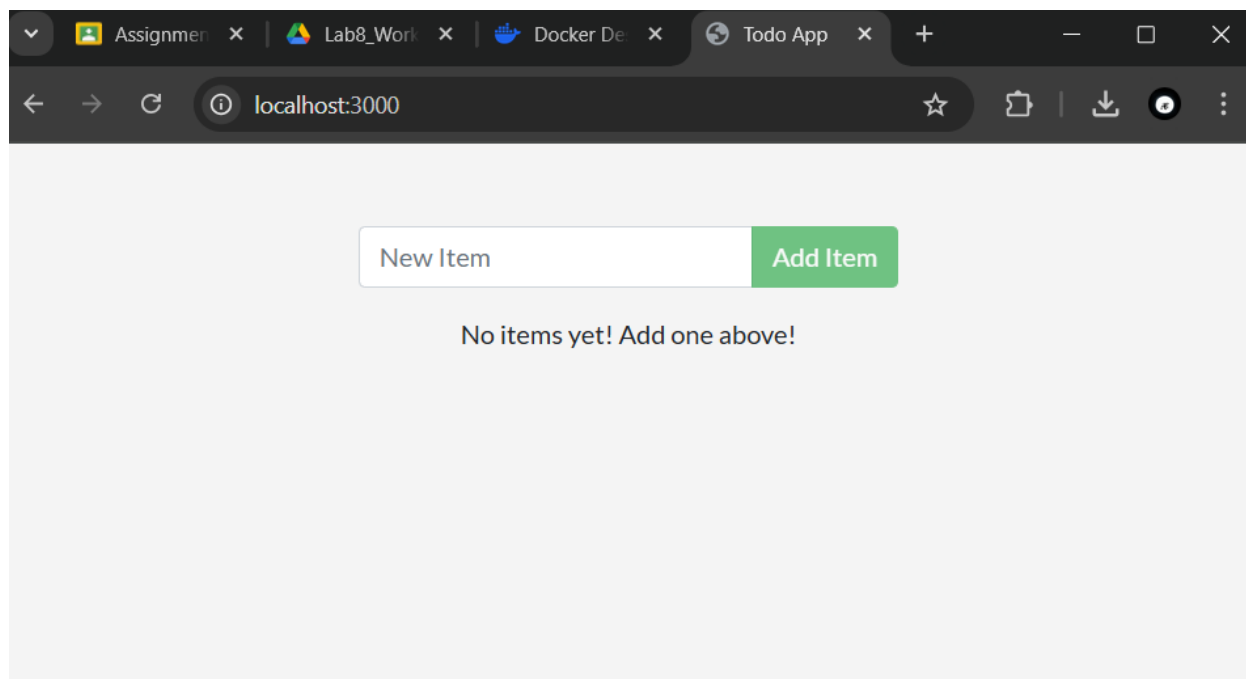
[Check point#9] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงผลลัพธ์ที่ได้บน Browser และ Dashboard ของ Docker desktop

```
PS C:\Users\ASUS\Lab8_4\getting-started\app> docker run -dp 3000:3000 myapp_6533803458
f3e0429c7cc92e8e6239f7d8841b1dfff01d58d8ca20c295ca5975804006981af
PS C:\Users\ASUS\Lab8_4\getting-started\app> docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
f3e0429c7cc9	myapp_6533803458	"docker-entrypoint.s..."	7 seconds ago	Up 6 seconds	0.0.0.0:3000->3000/tcp

```
cp cool_chaplygin
PS C:\Users\ASUS\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.`

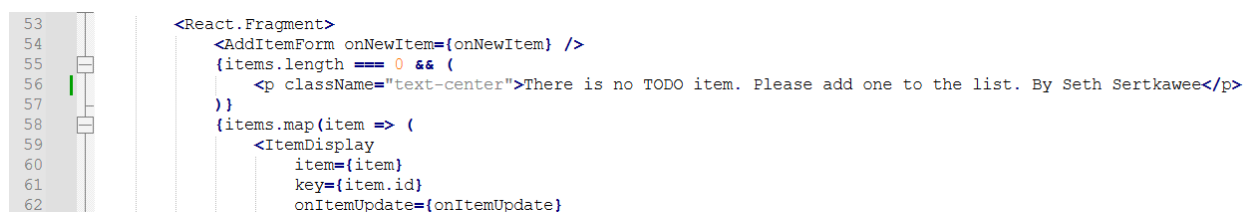
`By ชื่อและนามสกุลของนักศึกษา</p>`

b. Save ไฟล์ให้เรียบร้อย

9. ทำการ Build Docker image โดยใช้คำสั่งเดียวกันกับข้อ 5

10. Start และรัน Container ตัวใหม่ โดยใช้คำสั่งเดียวกันกับข้อ 6

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





## Lab Worksheet

```
PS C:\Users\ASUS\Lab8_4\getting-started\app> docker build -t myapp_6533803458 .
[+] Building 25.0s (10/10) FINISHED                                docker:desktop-linux
=> [internal] load build definition from Dockerfile                0.0s
=> => transferring dockerfile: 154B                                0.0s
=> [internal] load metadata for docker.io/library/node:18-alpine  2.6s
=> [auth] library/node:pull token for registry-1.docker.io        0.0s
=> => exporting layers                                              4.3s
=> => exporting manifest sha256:e6b580ed966da076b41c603b153278bad9d82cf1c507fc2c701d751be7ff4d4d 0.0s
=> => exporting config sha256:0abb2494e82cf4c10364c5d9825e69d9f1e062ec0884f73f725d4770709d9395 0.1s
=> => exporting attestation manifest sha256:8832d65ec609861dedfc05d646da46062f3a3393cbef6417753502327f7f2a26 0.1s
=> => exporting manifest list sha256:144fd3d1c6c6efaa271eef1f6da0561d8b9a29c08628626935add038e6e6d0b3 0.0s
=> => naming to docker.io/library/myapp_6533803458:latest         0.0s
=> => unpacking to docker.io/library/myapp_6533803458:latest      2.3s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/9d537bc1a24cg9qksbpg3d2a

PS C:\Users\ASUS\Lab8_4\getting-started\app> docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                    NAMES
9700ca8fd5f0   myapp_6533803458 "docker-entrypoint.s..." 7 minutes ago  Up 7 minutes  0.0.0.0:3000->3000/tcp    focused_wright

PS C:\Users\ASUS\Lab8_4\getting-started\app>
PS C:\Users\ASUS\Lab8_4\getting-started\app> docker run -dp 3000:3000 myapp_6533803458
374f551c7a633209092aaffcf589df389aed082cc572135255c1e4d0bd8bb19c
docker: Error response from daemon: driver failed programming external connectivity on endpoint recursing_williams (15381d1d9413d042d94590e86152d2437a1b4309faaeb30f37135eca6973b592c): Bind for 0.0.0.0:3000 failed: port is already allocated.
PS C:\Users\ASUS\Lab8_4\getting-started\app> █
```

(1) Error ที่เกิดขึ้นหมายความว่าอย่างไร และเกิดขึ้นเพราะอะไร

ตอบ Docker ไม่สามารถรัน Container ใหม่ได้ เพราะ Port 3000 ถูกใช้งานอยู่แล้ว โดย Container อื่น สาเหตุคือ มี Container อื่นที่ใช้ Port 3000 กำลังรันอยู่ ทำให้ Container ใหม่ไม่สามารถ Bind ไปยัง Port 3000:3000 ได้

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

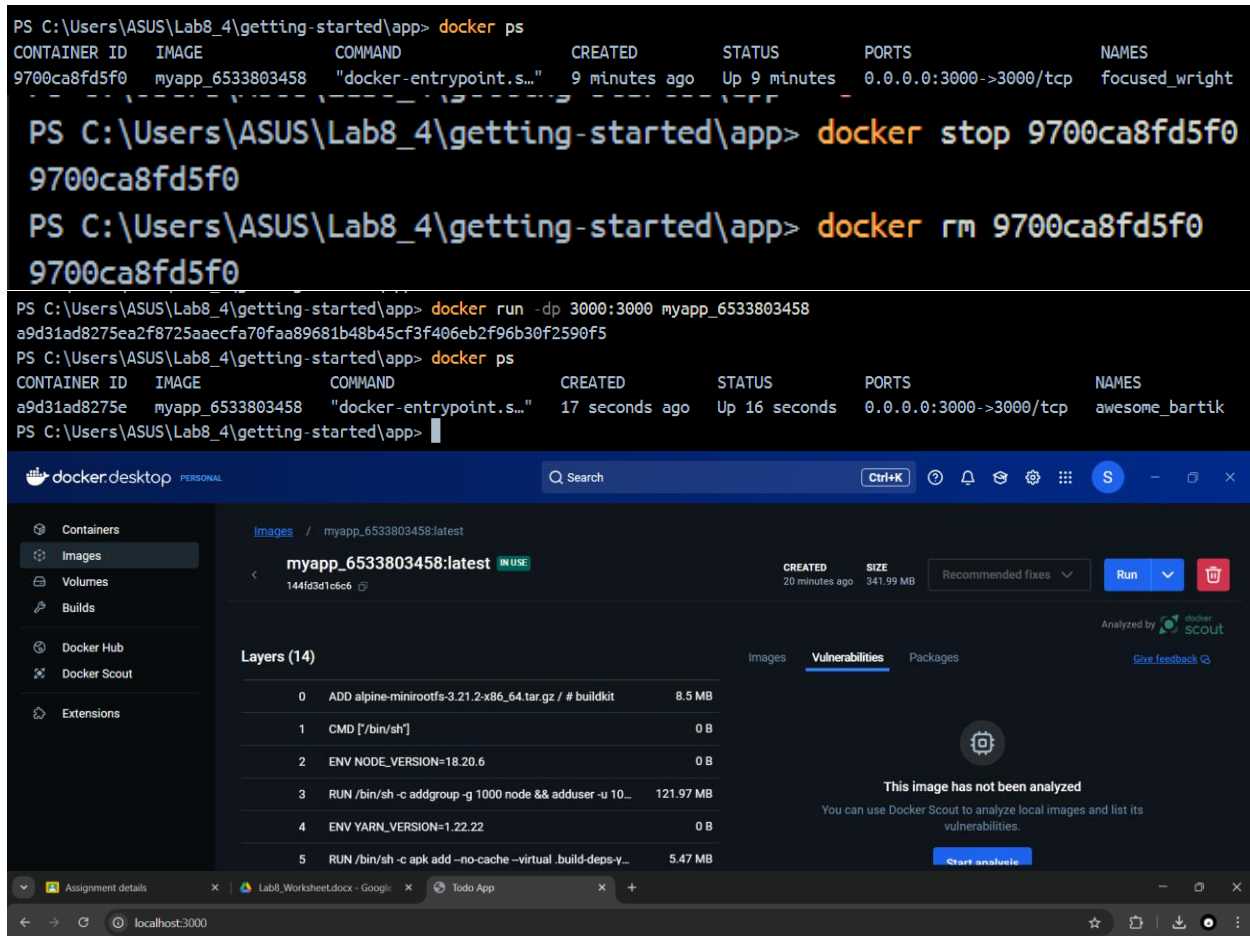
- ไปที่หน้าต่าง Containers
- เลือกไอคอนถังขยะในแถวของ Container ที่ต้องการจะลบ
- ยืนยันโดยการกด Delete forever

## Lab Worksheet

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



## 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

## Lab Worksheet

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

3. บันทึกการรหัสผ่านของ Admin user ไว้สำหรับ log-in ในครั้งแรก

[Check point#12] Capture หน้าจอที่แสดงผล Admin password

```
PS C:\Users\ASUS> mkdir Lab8_5

Directory: C:\Users\ASUS

Mode                LastWriteTime         Length Name
----                -
d-----          1/29/2025   9:50 PM             Lab8_5

PS C:\Users\ASUS> cd Lab8_5
PS C:\Users\ASUS\Lab8_5> docker run -p 8080:8080 -p 50000:50000 --restart=on-failure jenkins/jenkins:lts-jdk17
Running from: /usr/share/jenkins/jenkins.war
webroot: /var/jenkins_home/war
2025-01-29 15:01:01.064+0000 [id=1] INFO winstone.Logger#logInternal: Beginning extraction from war file
2025-01-29 15:01:01.674+0000 [id=1] WARNING o.e.j.ee9.nested.ContextHandler#setContextPath: Empty contextPath
2025-01-29 15:01:01.718+0000 [id=1] INFO org.eclipse.jetty.server.Server#doStart: jetty-12.0.16; built: 2024-12-09T21:02:54.535Z; git: c3f88bafb4e393f23204dc14dc57b042e84debc7; jvm 17.0.13+11
2025-01-29 15:01:02.003+0000 [id=1] INFO o.e.j.e.w.StandardDescriptorProcessor#visitServlet: NO JSP Support for /, did not find org.eclip
se.jetty.ee9.jsp.JettyJspServlet
2025-01-29 15:01:02.046+0000 [id=1] INFO o.e.j.s.DefaultSessionIdManager#doStart: Session workerName=node0
2025-01-29 15:01:02.404+0000 [id=1] INFO hudson.WebAppMain#contextInitialized: Jenkins home directory: /var/jenkins_home found at: EnvVar
s.masterEnvVars.get("JENKINS_HOME")
2025-01-29 15:01:02.519+0000 [id=1] INFO o.e.j.s.handler.ContextHandler#doStart: Started oeje9n.ContextHandler$CoreContextHandler@772861a
a{Jenkins v2.479.3/,b=file:///var/jenkins_home/war/,a=AVAILABLE,h=oeje9n.ContextHandler$CoreContextHandler$CoreToNestedHandler@6631cb64{STARTED
}}
2025-01-29 15:01:02.530+0000 [id=1] INFO o.e.j.server.AbstractConnector#doStart: Started ServerConnector@27df0f3d{HTTP/1.1, (http/1.1)}{0
.0.0.0:8080}
2025-01-29 15:01:02.545+0000 [id=1] INFO org.eclipse.jetty.server.Server#doStart: Started oejs.Server@1c7fd41f{STARTING}[12.0.16,sto=0] @
```

## Lab Worksheet

```
*****
*****
*****

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

7b8b73a63f284f149260b1ebee7457dd

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

*****
*****
*****
```

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

[Check point#13] Capture หน้าจอที่แสดงผลการตั้งค่า

## Lab Worksheet

### Getting Started

Username

seth\_3458

Password

.....

Confirm password

.....

Full name

Seth sertkawee

E-mail address

seth.s@kkumail.com

Jenkins 2.479.3

[Skip and continue as admin](#) [Save and Continue](#)

## Lab Worksheet

## Getting Started

# Instance Configuration

Jenkins URL:

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](#)

## Lab Worksheet

## Getting Started

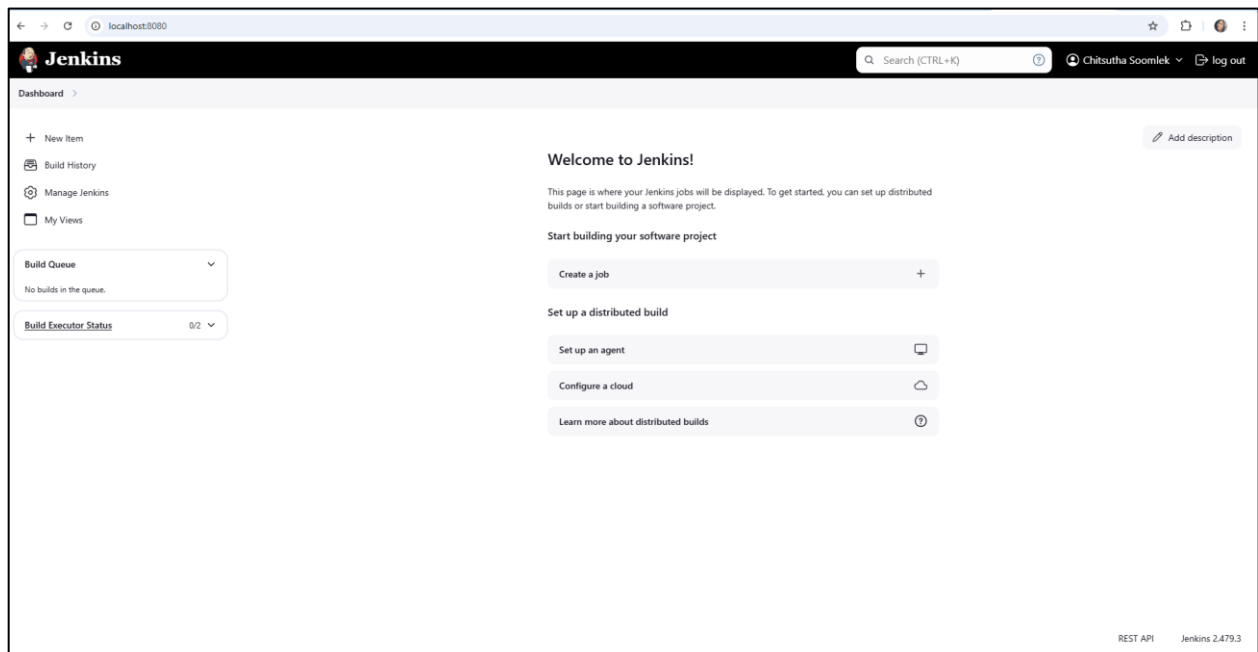
# Jenkins is ready!

Your Jenkins setup is complete.

Start using Jenkins

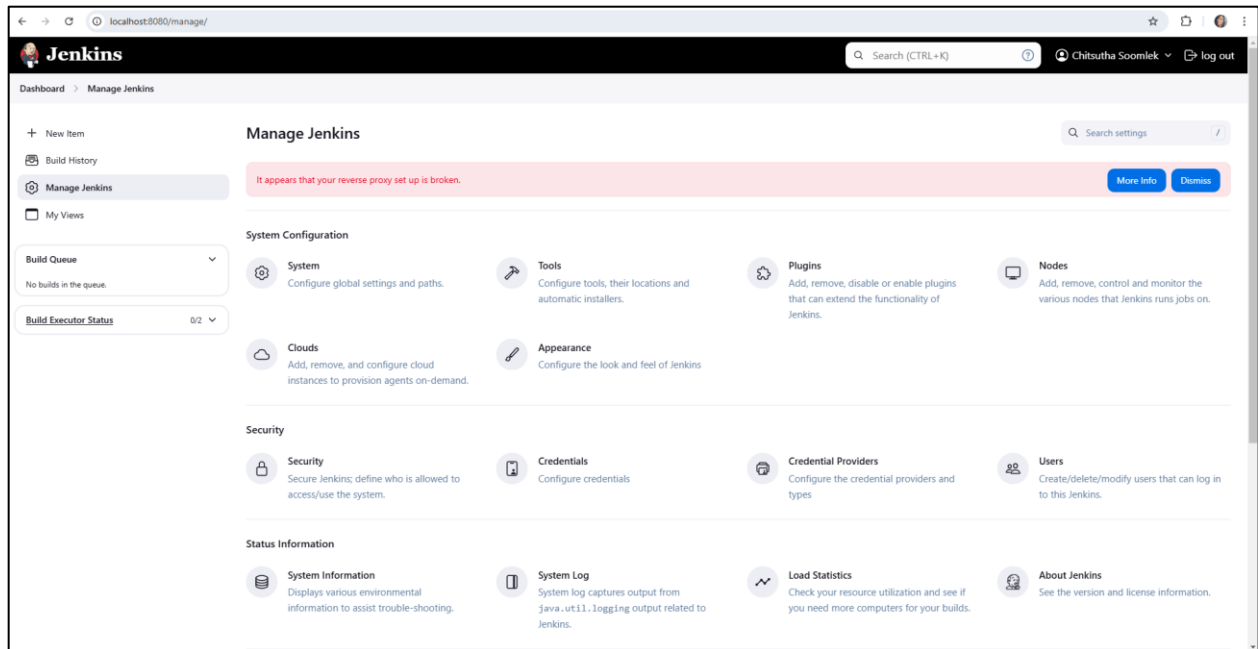
7. กำหนด Jenkins URL เป็น <http://localhost:8080/lab8>

8. เมื่อติดตั้งเรียบร้อยแล้วจะพบกันหน้า Dashboard ดังแสดงในภาพ

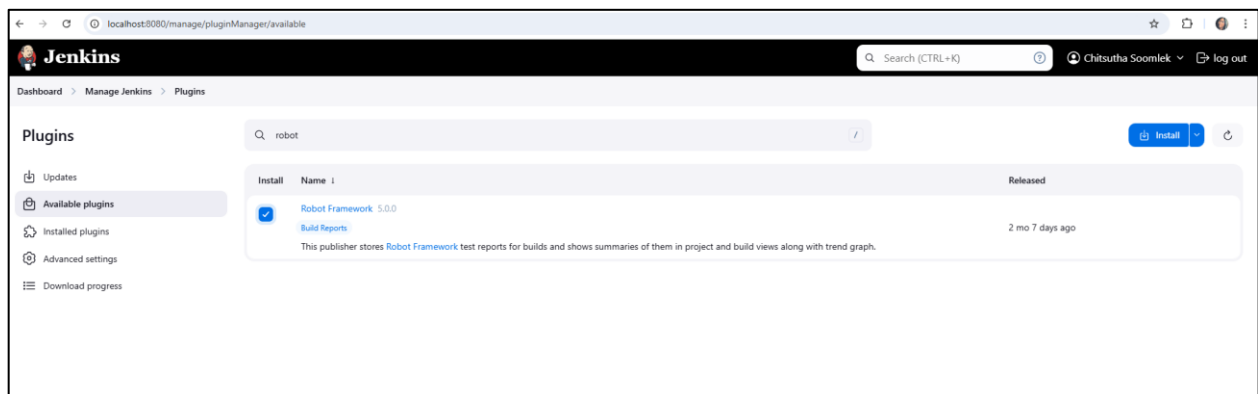


## Lab Worksheet

## 9. เลือก Manage Jenkins แล้วไปที่เมนู Plugins



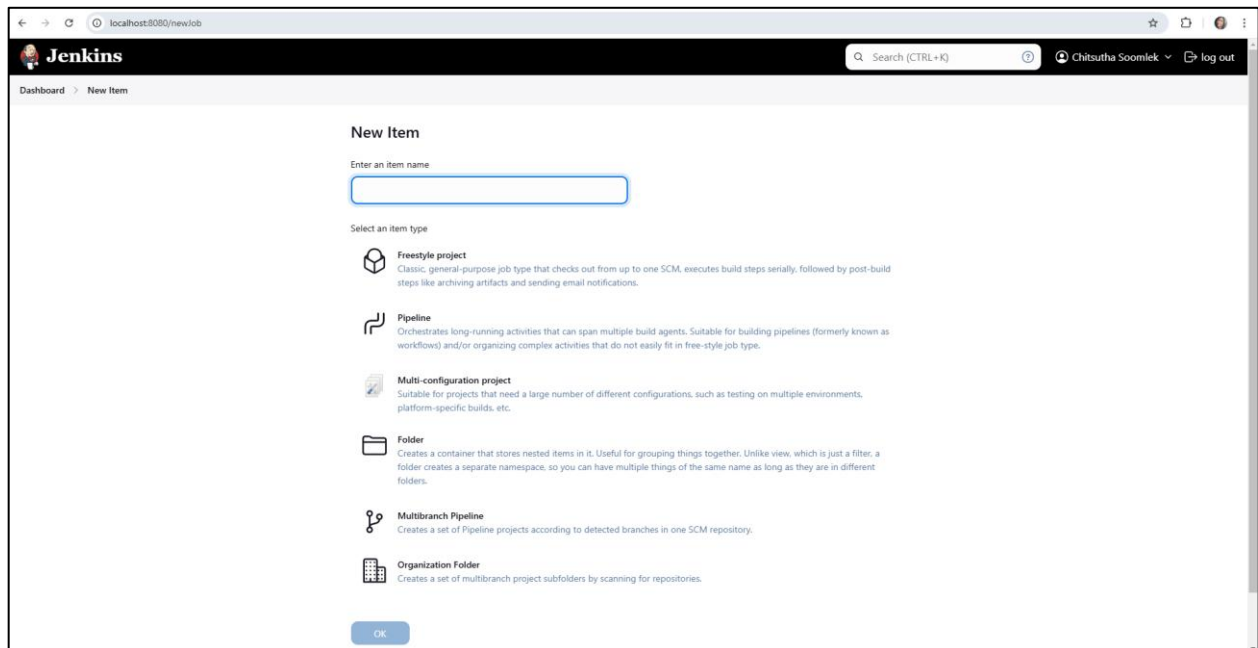
## 10. ไปที่เมนู Available plugins แล้วเลือกติดตั้ง Robotframework เพิ่มเติม



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



## Lab Worksheet



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

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

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

## Lab Worksheet

Dashboard > UAT > Configuration

### Configure

General

Source Code Management

Build Triggers

Build Environment

Build Steps

Post-build Actions

### General

Enabled

Description

Lab 8.5

Plain text [Preview](#)

☐ Discard old builds ?

☒ GitHub project

Project url ?

<https://github.com/kku-computer-science/configuration-management-seth5400.git/>

Advanced ▾

☐ This project is parameterized ?

☐ Throttle builds ?

☐ Execute concurrent builds if necessary ?

Advanced ▾

[Save](#) [Apply](#)

### Build Triggers

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

☐ Build after other projects are built ?

☒ Build periodically ?

Schedule ?

H/15 \* \* \* \*

Would last have run at Wednesday, January 29, 2025 at 3:35:07 PM Coordinated Universal Time; would next run at Wednesday, January 29, 2025 at 3:50:07 PM Coordinated Universal Time.

☐ GitHub hook trigger for GITSCM polling ?

### Build Steps

Execute shell ?

Command

[See the list of available environment variables](#)

```
cd Lab8_653380345-8
mkdir -p results
robot --outputdir results *.robot
```

Advanced ▾

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

ตอบ    cd Lab8\_653380345-8  
           mkdir -p results  
           robot --outputdir results \*.robot

## Lab Worksheet

## อธิบาย

cd Lab8\_653380345-8 → เข้าไปในโฟลเดอร์ที่เก็บไฟล์ .robot

mkdir -p results → สร้างโฟลเดอร์ results

robot --outputdir results \*.robot → รันทุกไฟล์ .robot ในโฟลเดอร์ที่เก็บไฟล์ .robot และเอาผลลัพธ์เก็บไว้ใน results

**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

14. สั่ง Build Now

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

The screenshot shows the Jenkins web interface. At the top, the Jenkins logo and a search bar are visible. The main content area is for a pipeline named 'UAT'. The 'Status' tab is selected, showing a red 'X' icon and the text 'Lab 8.5'. Below this, there is a section for 'Latest Robot Results' which states 'No results available yet.' and a 'Permalinks' section with a list of build links. On the left sidebar, there is a 'Builds' list showing a series of builds from #25 to #35, with status icons (red for failed, green for successful) and timestamps.

## Lab Worksheet

## Build Steps

Execute shell ?

Command

See the list of available environment variables

```
cd Lab8_653380345-8
mkdir -p results
robot --outputdir results *.robot
```

Advanced ▾

## Post-build Actions

Publish Robot Framework test results ?

Directory of Robot output

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

results

Advanced ▾ Edited

Thresholds for build result ?

%

20.0

%

80.0

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

☐ Include skipped tests in total count for thresholds

Add post-build action ▾

Save

Apply

Search (CTRL+K)
Seth sertkaweew
log out

Dashboard > UAT > #35 > Console Output

Status
Changes
Console Output
Edit Build Information
Delete build #35
Timings
Previous Build

Console Output
Download
Copy
View as plain text

Started by user Seth sertkaweew
Running as SYSTEM
Building in workspace /var/jenkins\_home/workspace/UAT
[UAT] \$ /bin/sh -xe /tmp/jenkins102474841539210599.sh
+ cd Lab8\_653380345-8
+ mkdir -p results
+ robot --outputdir results Lab7Test-1.robot Lab7Test-2.robot Lab7Test.robot gherkin\_login.robot invalid\_login.robot resource.robot valid\_login.robot
/tmp/jenkins102474841539210599.sh: 4: robot: not found
Build step 'Execute shell' marked build as failure
Robot results publisher started...
INFO: Checking test criticality is deprecated and will be dropped in a future release!
-Parsing output xml:
Failed!
hudson.AbortException: No files found in path /var/jenkins\_home/workspace/UAT/results with configured filemask: output.xml
at PluginClassLoader for robot/hudson.plugins.robot.RobotParser\$RobotParserCallable.invoke(RobotParser.java:81)
at PluginClassLoader for robot/hudson.plugins.robot.RobotParser\$RobotParserCallable.invoke(RobotParser.java:152)
at hudson.FilePath.act(FilePath.java:1234)
at hudson.FilePath.act(FilePath.java:1217)
at PluginClassLoader for robot/hudson.plugins.robot.RobotParser.parse(RobotParser.java:48)
at PluginClassLoader for robot/hudson.plugins.robot.RobotPublisher.parse(RobotPublisher.java:262)
at PluginClassLoader for robot/hudson.plugins.robot.RobotPublisher.perform(RobotPublisher.java:286)
at hudson.tasks.BuildStepCompatibilityLayer.perform(BuildStepCompatibilityLayer.java:80)
at hudson.tasks.BuildStepMonitor\$1.perform(BuildStepMonitor.java:20)
at hudson.model.AbstractBuild\$AbstractBuildExecution.perform(AbstractBuild.java:818)
at hudson.model.AbstractBuild\$AbstractBuildExecution.performAllBuildSteps(AbstractBuild.java:767)
at hudson.model.Build\$BuildExecution.post2(Build.java:179)
at hudson.model.AbstractBuild\$AbstractBuildExecution.post(AbstractBuild.java:711)
at hudson.model.Run.execute(Run.java:1854)
at hudson.model.FreeStyleBuild.run(FreeStyleBuild.java:44)
at hudson.model.ResourceController.execute(ResourceController.java:101)
at hudson.model.Executor.run(Executor.java:445)
Finished: FAILURE