

## **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

## CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตสุธา สุ่มเล็ก

### Lab Worksheet

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**

- (1) สิ่งที่อยู่ภายใต้คอลัมน์ Repository คืออะไร

คอลัมน์ REPOSITORY จะแสดงชื่อของ Docker image ที่ถูกดึง (pull) ลงมาจาก Docker Hub หรือจาก registry อื่นๆ

- (2) Tag ที่ใช้บ่งบอกถึงอะไร

คอลัมน์ TAG ใช้เพื่อระบุเวอร์ชันหรือการอ้างอิงเฉพาะของ Docker image นั้นๆ โดยทั่วไปจะใช้ **latest** เพื่อบ่งชี้ว่าเป็นเวอร์ชันล่าสุด หากไม่ระบุ tag จะถูกอ้างอิงเป็น **latest** โดยอัตโนมัติ

# CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตส์ฐา สุ่มเล็ก

## Lab Worksheet

### 5. ป้อนคำสั่ง \$ docker run busybox

The screenshot shows the Docker Desktop application. The top bar includes the Docker logo, 'docker desktop PERSONAL', a search bar, and a 'Ctrl+K' button. The left sidebar contains icons for Containers, Images, Clusters, Settings, and Extensions. The main area is titled 'Containers' and shows 'No containers are running.' for both CPU and memory usage. Below this is a table of containers:

Name	Container ID	Image	Port(s)	Actions
elated_chaum	415a9157385c	busybox	-	[Play] [More] [Delete]
docker	-	-	-	[Play] [More] [Delete]

Below the containers list is a 'Terminal' window showing the following commands and output:

```
PS C:\Users\fernk\Lab8_1> docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
busybox       latest   af4709625109   4 months ago   4.27MB
bestbodini/flowerdb latest    7ce93a845a8a   6 months ago   586MB
PS C:\Users\fernk\Lab8_1> docker run busybox
PS C:\Users\fernk\Lab8_1>
PS C:\Users\fernk\Lab8_1>
```

The bottom status bar shows system information: RAM 1.37 GB, CPU 0.17%, and Disk usage.

# CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตสุธา สุ่มเล็ก

## Lab Worksheet

### 6. ป้อนคำสั่ง \$ docker run -it busybox sh

The screenshot shows the Docker Desktop application window. The top bar is blue with the Docker logo, 'docker desktop PERSONAL', a search bar, and system icons. The left sidebar contains icons for Containers, Images, Clusters, Settings, and Extensions. The main area is titled 'Containers' and shows 'Container CPU usage' and 'Container memory usage' as 'No containers are running.' Below this is a search bar and a toggle for 'Only running'. A table lists three containers: 'elated\_chaum', 'vigorous\_yalow', and 'docker'. The 'docker' container is highlighted. At the bottom, a terminal window is open, showing the command 'docker run -it busybox sh' and the resulting shell prompt '#'. The terminal also shows the output of 'docker images' and 'docker run busybox'.

Name	Container ID	Image	Port(s)	Actions
elated_chaum	415a9157385c	busybox	-	[Play] [Vertical Dots] [Trash]
vigorous_yalow	f708526eae9c	busybox	-	[Stop] [Vertical Dots] [Trash]
docker	-	-	-	[Play] [Vertical Dots] [Trash]

```
n
+ FullyQualifiedErrorId : CommandNotFoundException

PS C:\Users\fernk\Lab8_1> docker images
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
busybox              latest          af4709625109   4 months ago   4.27MB
bestbodini/flowerdb latest          7ce93a845a8a   6 months ago   586MB
PS C:\Users\fernk\Lab8_1> docker run busybox
PS C:\Users\fernk\Lab8_1> docker run -it busybox sh
>> C:\Users\fernk\Lab8_1>
/ #
```

# CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตสุธา สุ่มเล็ก

## Lab Worksheet

### 7. ป้อนคำสั่ง ls

The screenshot shows the Docker Desktop interface. The top bar includes the Docker logo, a search bar, and various icons. The left sidebar contains navigation icons. The main area is titled 'Containers' and shows system usage: 'Container CPU usage 0.00% / 1200% (12 CPUs available)' and 'Container memory usage 452KB / 7.46GB'. Below this is a table of containers.

	Name	Container ID	Image	Port(s)	Actions
<input type="checkbox"/>	elated_chaum	415a9157385c	busybox	-	
<input type="checkbox"/>	vigorous_yalow	f708526eae9c	busybox	-	<input type="checkbox"/>
<input type="checkbox"/>	docker	-	-	-	

Below the containers table is a 'Terminal' window titled 'Showing 3 items'. It contains the following commands and output:

```
PS C:\Users\fernk\Lab8_1> docker images
REPOSITORY          TAG         IMAGE ID      CREATED        SIZE
busybox              latest      af4709625109 4 months ago   4.27MB
bestbodn/flowerdb   latest      7ce93a845a8a 6 months ago   586MB
PS C:\Users\fernk\Lab8_1> docker run busybox
PS C:\Users\fernk\Lab8_1> docker run -it busybox sh
>> C:\Users\fernk\Lab8_1>
/ # ls
bin    etc    lib    proc   sys    usr
dev    home  lib64  root   tmp    var
/ #
```

At the bottom of the Docker Desktop window, system statistics are shown: 'RAM 0.91 GB CPU 0.17% Disk -- GB avail. of -- GB'. A notification badge shows '4'.

# CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตสุธา สุ่มเล็ก

## Lab Worksheet

### 8. ป้อนคำสั่ง ls -la

The screenshot shows the Docker Desktop interface. The top bar includes the Docker logo, a search bar, and a 'PERSONAL' label. The main area is titled 'Containers' and shows system usage: 'Container CPU usage 0.00% / 1200% (12 CPUs available)' and 'Container memory usage 2.66MB / 7.46GB'. Below this is a table of containers:

	Name	Container ID	Image	Port(s)	Actions
<input type="checkbox"/>	elated_chaum	415a9157385c	busybox	-	
<input type="checkbox"/>	vigorous_yalow	f708526eae9c	busybox	-	<input type="checkbox"/>

Below the containers table is a 'Terminal' window. The terminal shows the following commands and output:

```
PS C:\Users\fernk\Lab8_1> docker run -it busybox sh
>> C:\Users\fernk\Lab8_1>
/ # ls
bin    etc    lib    proc   sys    usr
dev    home   lib64  root   tmp    var
drwxr-xr-x  5 root   root   360 Jan 28 13:30 dev
drwxr-xr-x  1 root   root   4096 Jan 28 13:30 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 242 root   root    0 Jan 28 13:30 proc
drwx----- 1 root   root   4096 Jan 28 13:30 root
dr-xr-xr-x 11 root   root    0 Jan 28 13:30 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
/ #
```

The bottom status bar shows 'RAM 0.97 GB CPU -- % Disk -- GB avail. of -- GB' and a notification icon with the number 4.

# CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตส์ฐา สุ่มเล็ก

## Lab Worksheet

### 9. ป้อนคำสั่ง exit

The screenshot shows the Docker Desktop interface. The top bar includes the Docker logo, a search bar, and window controls. The left sidebar contains icons for containers, images, volumes, settings, and a plus sign. The main area is titled 'Containers' and shows that no containers are running. Below this, there is a search bar and a toggle for 'Only running'. A table lists two containers: 'elated\_chaum' and 'vigorous\_yalow', both using the 'busybox' image. At the bottom, a terminal window is open, displaying the output of the 'ls' command, showing the directory structure of the container's root filesystem.

Name	Container ID	Image	Port(s)	Actions
elated_chaum	415a9157385c	busybox	-	[Play] [More] [Trash]
vigorous_yalow	f708526eae9c	busybox	-	[Play] [More] [Trash]

Permissions	User	Group	Path	Size	Time	File Type
drwxr-xr-x	5	root	root	360	Jan 28 13:30	dev
drwxr-xr-x	1	root	root	4096	Jan 28 13:30	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	242	root	root	0	Jan 28 13:30	proc
drwx-----	1	root	root	4096	Jan 28 13:30	root
dr-xr-xr-x	11	root	root	0	Jan 28 13:30	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

### 10. ป้อนคำสั่ง \$ docker run busybox echo "Hello ชื่อและนามสกุลของนักศึกษา from busybox"

```
PS C:\Users\fernk\Lab8_1> docker run busybox echo "Hello Kamonlak Wongsanga from busybox"
Hello Kamonlak Wongsanga from busybox
```

# CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตส์ฐา สุ่มเล็ก

## Lab Worksheet

### 11. ป้อนคำสั่ง \$ docker ps -a

The screenshot shows the Docker Desktop application. The top bar includes the Docker logo, 'docker desktop PERSONAL', a search bar, and a 'Ctrl+K' button. The left sidebar contains navigation icons. The main area is titled 'Containers' and shows 'No containers are running.' for both CPU and memory usage. Below this is a search bar and a toggle for 'Only running'. A table lists containers, including 'elated\_chaum' and 'vigorous\_yalow'. The 'Terminal' pane at the bottom shows the command 'docker ps -a' and its output, which lists several containers including 'tender\_bell', 'vigorous\_yalow', 'elated\_chaum', and 'mysql\_container'. The bottom status bar shows system resources like RAM and CPU usage.

Name	Container ID	Image	Port(s)	Actions
elated_chaum	415a9157385c	busybox	-	[Play] [More] [Trash]
vigorous_yalow	f708526eae9c	busybox	-	[Play] [More] [Trash]

```
drwxr-xr-x  4 root    root      4096 Sep 26 21:31 var
/ # exit
PS C:\Users\fern\Lab8_1> docker run busybox echo "Hello Kamonlak Wongsanga from busybox"
Hello Kamonlak Wongsanga from busybox
PS C:\Users\fern\Lab8_1> docker ps -a
CONTAINER ID   IMAGE          PORTS          COMMAND                  CREATED
STATUS        NAMES
3694abe6af57   busybox                "echo 'Hello Kamonla..." 15 seconds ago
Exited (0) 14 seconds ago      tender_bell
f708526eae9c   busybox                "sh"                      3 minutes ago
Exited (0) About a minute ago  vigorous_yalow
415a9157385c   busybox                "sh"                      4 minutes ago
Exited (0) 4 minutes ago      elated_chaum
e827b3f9e927   bestbodin/flowerdb:latest "docker-entrypoint.s..." 4 months ago
Restarting (3) 36 seconds ago  mysql_container
PS C:\Users\fern\Lab8_1>
```



## CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตสุธา สุ่มเล็ก

### Lab Worksheet

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

- (1) เมื่อใช้ option -it ในคำสั่ง run ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป  
เข้าสู่ shell หรือใช้คำสั่งต่างๆ ภายในคอนเทนเนอร์ได้เหมือนกับว่าเป็นการใช้งานเครื่องคอมพิวเตอร์ปกติ
- (2) คอลัมน์ STATUS จากการรันคำสั่ง docker ps -a แสดงถึงข้อมูลอะไร  
ข้อมูลเกี่ยวกับสถานะปัจจุบันของคอนเทนเนอร์ เช่น กำลังทำงานอยู่ หรือถูกหยุดชั่วคราว

# CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตส์ฐา สุ่มเล็ก

## Lab Worksheet

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

The screenshot shows the Docker Desktop interface. The top bar includes the Docker logo, 'docker desktop PERSONAL', a search bar, and system icons. The left sidebar contains navigation icons. The main area is titled 'Containers' and features a search bar, a toggle for 'Only running', and a table of containers.

	Name	Container ID	Image	Port(s)	Actions
<input type="checkbox"/>	elated_chaum	415a9157385c	busybox	-	
<input type="checkbox"/>	vigorous_yalow	f708526eae9c	busybox	-	
<input type="checkbox"/>	docker	-	-	-	

Below the containers table is a 'Terminal' window showing the following commands and output:

```
Hello Kamonlak Wongsanga from busybox
PS C:\Users\fernk\Lab8_1> docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED
STATUS        PORTS    NAMES
3694abe6af57   busybox   "echo 'Hello Kamonla..." 15 seconds ago
Exited (0) 14 seconds ago    tender_bell
f708526eae9c   busybox   "sh"                      3 minutes ago
Exited (0) About a minute ago    vigorous_yalow
415a9157385c   busybox   "sh"                      4 minutes ago
Exited (0) 4 minutes ago    elated_chaum
e827b3fbe927   bestbodn/flowerdb:latest "docker-entrypoint.s..." 4 months ago
Restarting (3) 36 seconds ago    mysql_container
PS C:\Users\fernk\Lab8_1> ls
PS C:\Users\fernk\Lab8_1> docker rm ^C
PS C:\Users\fernk\Lab8_1> docker rm 3694abe6af57
3694abe6af57
PS C:\Users\fernk\Lab8_1>
```

The bottom status bar shows system resources: RAM 1.01 GB, CPU 0.25%, and Disk usage.

**[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."
```

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

```
EOF
```

หรือใช้คำสั่ง

```
$ touch Dockerfile
```

แล้วใช้ Text Editor ในการใส่เนื้อหาแทน

# CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตสุธา สุ่มเล็ก

## Lab Worksheet

5. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้  
\$ docker build -t <ชื่อ Image> .

The screenshot shows the Docker Desktop application window. The top bar is blue with the Docker logo, 'docker desktop PERSONAL', a search bar, and a 'Ctrl+K' button. The left sidebar contains icons for Containers, Images, Volumes, Settings, and a plus sign. The main area is titled 'Containers' and includes a search bar, a toggle for 'Only running', and a table of containers.

	Name	Container ID	Image	Port(s)	Actions
<input type="checkbox"/>	elated_chaum	415a9157385c	busybox	-	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
<input type="checkbox"/>	vigorous_yalow	f708526eae9c	busybox	-	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
<input type="checkbox"/>	docker	-	-	-	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>

Showing 3 items

The bottom panel is titled 'Terminal' and shows the output of a Docker build command. The command is `docker build -t my_first_image .`. The output shows three warnings related to JSON arguments and multiple instructions in the same stage. The build process is finished, and the terminal shows the prompt `C:\Users\fernk\Lab8_2>`.

# CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตสุธา สุ่มเล็ก

## Lab Worksheet

6. เมื่อ Build สำเร็จแล้ว ให้ทำการรัน Docker image ที่สร้างขึ้นในขั้นตอนที่ 5

The screenshot shows the Docker Desktop interface. The top bar includes the Docker logo, 'docker desktop PERSONAL', a search bar, and window controls. The left sidebar contains icons for Containers, Images, Clusters, Settings, and Extensions. The main area is titled 'Containers' and shows a table of running containers. Below the table is a 'Terminal' window showing the command history and output.

	Name	Container ID	Image	Port(s)	Actions
<input type="checkbox"/>	elated_chaum	415a9157385c	busybox	-	
<input type="checkbox"/>	vigorous_yalow	f708526eae9c	busybox	-	
<input type="checkbox"/>	quizzical_almeic	d16cbc76ab7e	my_first_img	-	
<input type="checkbox"/> >	docker	-	-	-	

Showing 4 items

```
PS C:\Users\fernk\Lab8_2>
PS C:\Users\fernk\Lab8_2>
PS C:\Users\fernk\Lab8_2>
PS C:\Users\fernk\Lab8_2>
PS C:\Users\fernk\Lab8_2>
PS C:\Users\fernk\Lab8_2>
PS C:\Users\fernk\Lab8_2> docker images
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
busybox              latest          af4709625109   4 months ago   4.27MB
my_first_image       latest          bba3512982d6   4 months ago   4.27MB
bestbodin/flowerdb   latest          7ce93a845a8a   6 months ago   586MB
PS C:\Users\fernk\Lab8_2> docker run my_first_image
"ผมสละผมไว้สง่า 653380261-4 เพ็น"
PS C:\Users\fernk\Lab8_2>
PS C:\Users\fernk\Lab8_2>
```

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

- (1) คำสั่งที่ใช้ในการ run คือ  
`docker run my_first_image`
- (2) Option -t ในคำสั่ง `$ docker build` ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป  
จะทำให้การเรียกใช้อิมเมจในภายหลังสะดวกขึ้น โดยสามารถใช้ชื่อแท็กที่กำหนดนี้ในการดึง (pull) หรือรับ image จาก Docker ได้ง่าย

### 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 บันทึกคำสั่งต่อไปนี้ลงในไฟล์โดยใช้ 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
```

## CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตสุธา สุ่มเล็ก

### Lab Worksheet

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

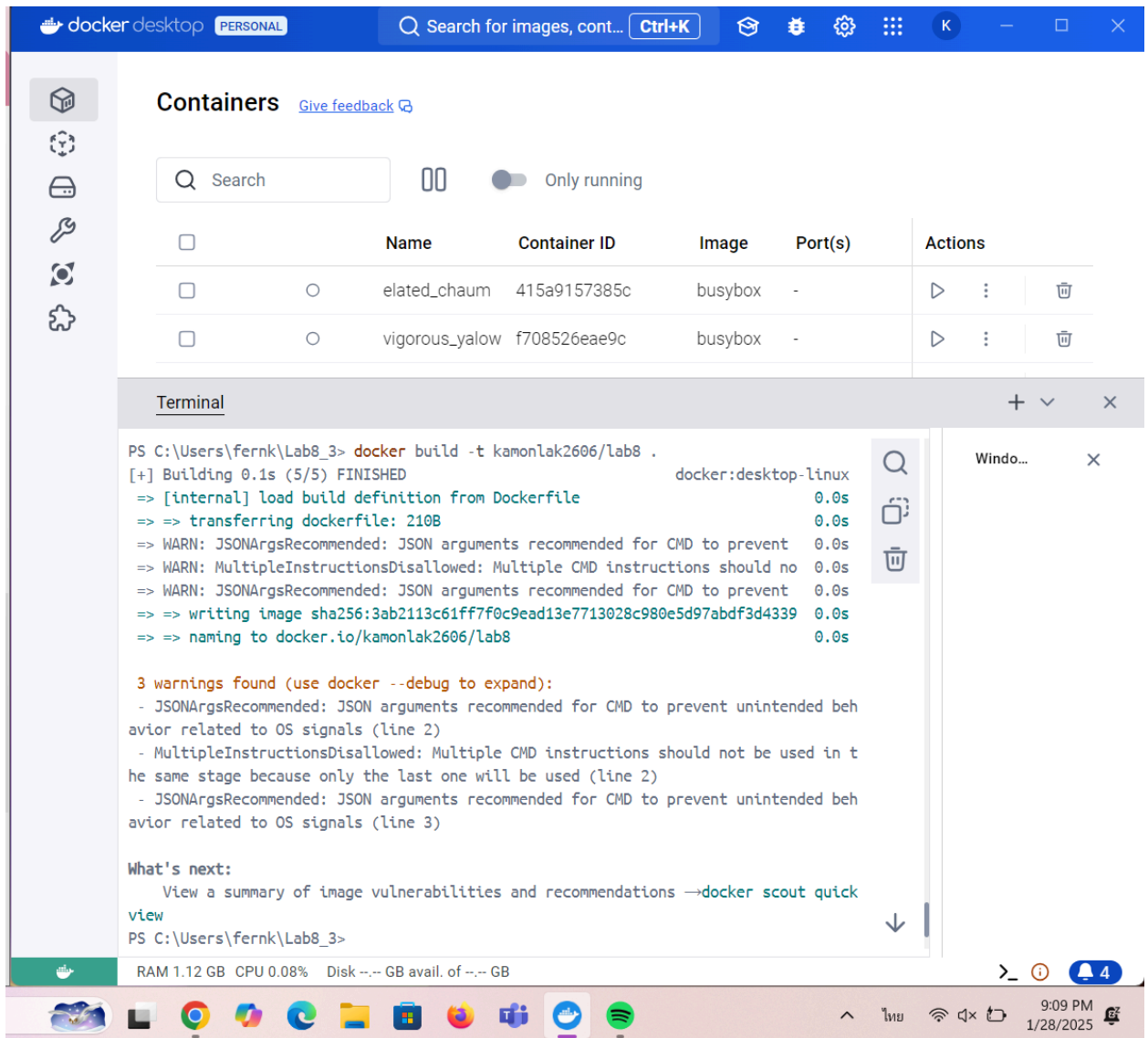
แล้วใช้ Text Editor ในการใส่เนื้อหาแทน

# CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตสุธา สุ่มเล็ก

## Lab Worksheet

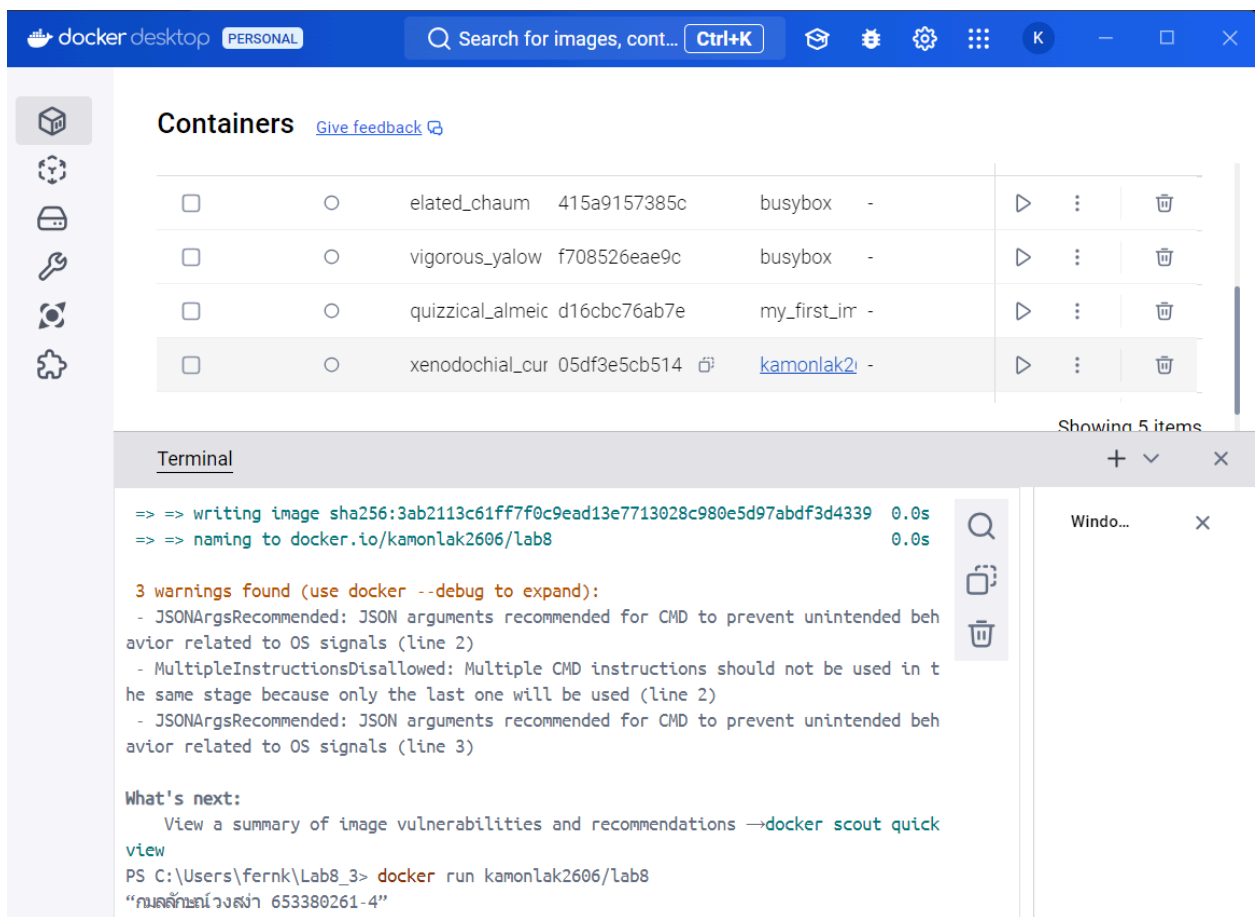
7. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้  
\$ docker build -t <username ที่ลงทะเบียนกับ Docker Hub>/lab8





5. ทำการรัน Docker image บน Container ในเครื่องของตัวเองเพื่อทดสอบผลลัพธ์ ด้วยคำสั่ง

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



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

6. ทำการ Push ตัว Docker image ไปไว้บน Docker Hub โดยการใช้คำสั่ง

## CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตสุธา สุ่มเล็ก

### Lab Worksheet

\$ 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>

```
PS C:\Users\fernk\Lab8_3> docker push kamonlak2606/lab8
```

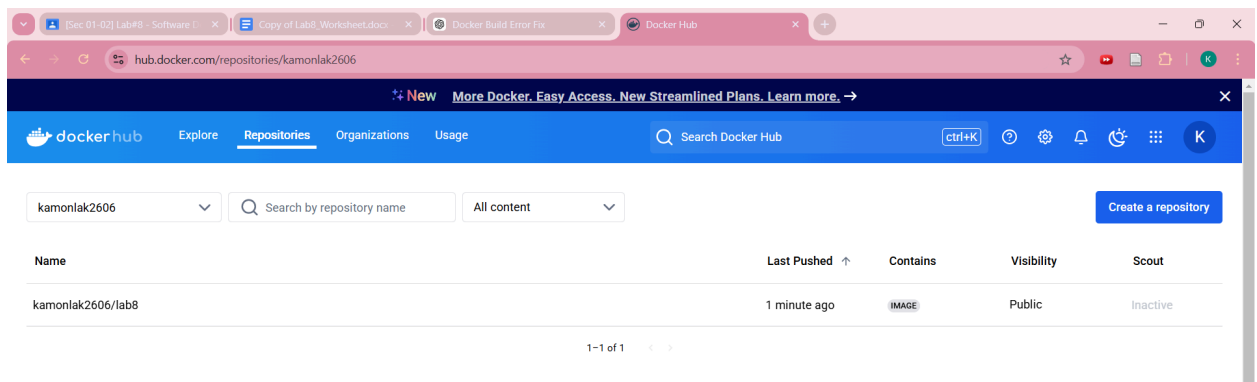
```
Using default tag: latest
```

```
The push refers to repository [docker.io/kamonlak2606/lab8]
```

```
59654b79daad: Layer already exists
```

```
latest: digest: sha256:b2fdc82270f4959429f02a9acc0bce63454a2bb83b4853adddf7ac24b7715d75 size: 527
```

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



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

### 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

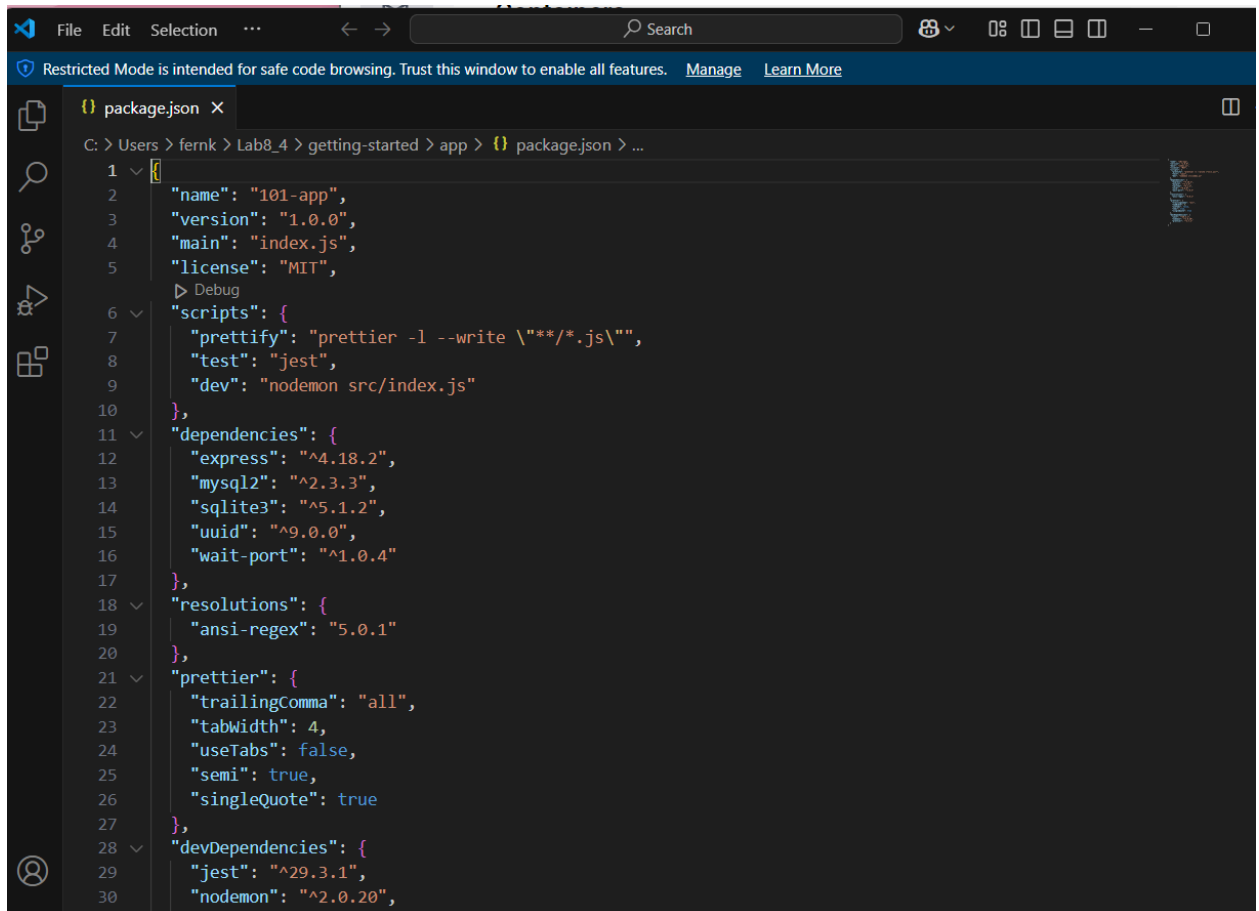
# CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตสุธา สุ่มเล็ก

## Lab Worksheet

\$ git clone <https://github.com/docker/getting-started.git>

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



The screenshot shows a VS Code editor window with the file explorer on the left and the package.json file open in the editor. The file path is C:\Users\fernk\Lab8\_4\getting-started\app\package.json. The package.json file contains the following JSON structure:

```
{
  "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": "^2.8.1"
  }
}
```

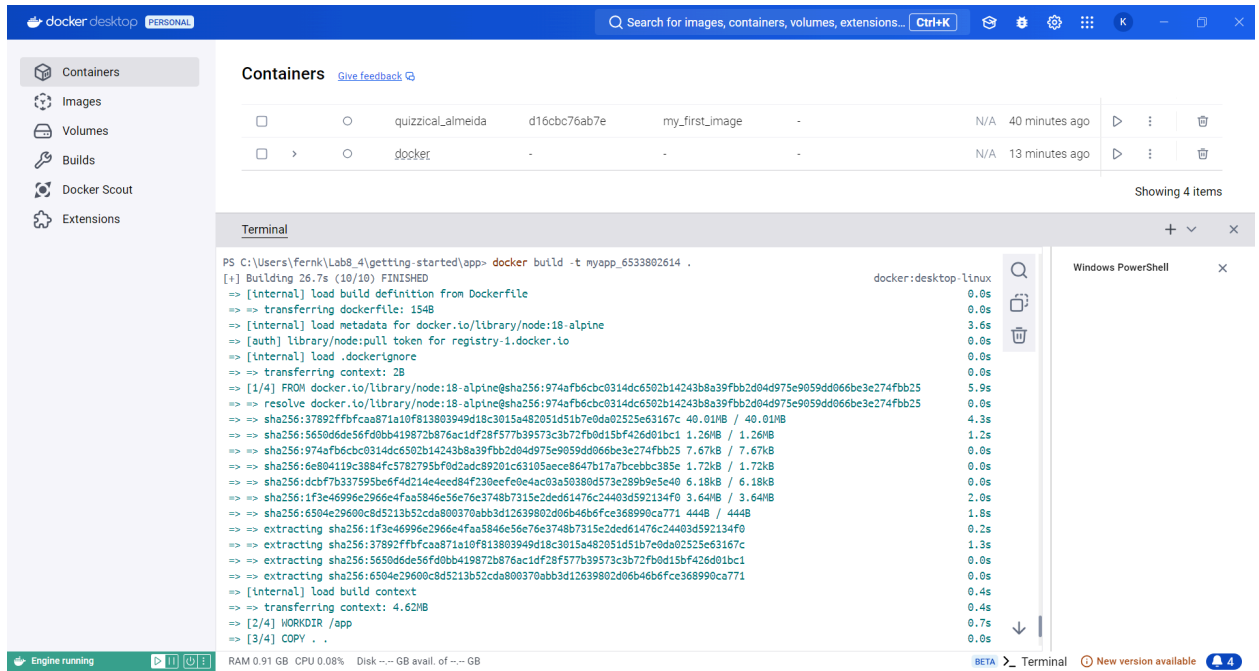
**[Check point#7]** Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงที่อยู่ของ Source code ที่ Clone มาและเนื้อหาของไฟล์ package.json

4. ภายใต้ getting-started/app ให้สร้าง Dockerfile พร้อมกับใส่เนื้อหาดังต่อไปนี้ลงในไฟล์  
FROM node:18-alpine  
WORKDIR /app  
COPY . .  
RUN yarn install --production  
CMD ["node", "src/index.js"]  
EXPOSE 3000
5. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้ โดยกำหนดชื่อ image เป็น myapp\_รหัสสนศ. ไม่มีขีด  
\$ docker build -t <myapp\_รหัสสนศ. ไม่มีขีด> .

# CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตสุธา สุ่มเล็ก

## Lab Worksheet



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

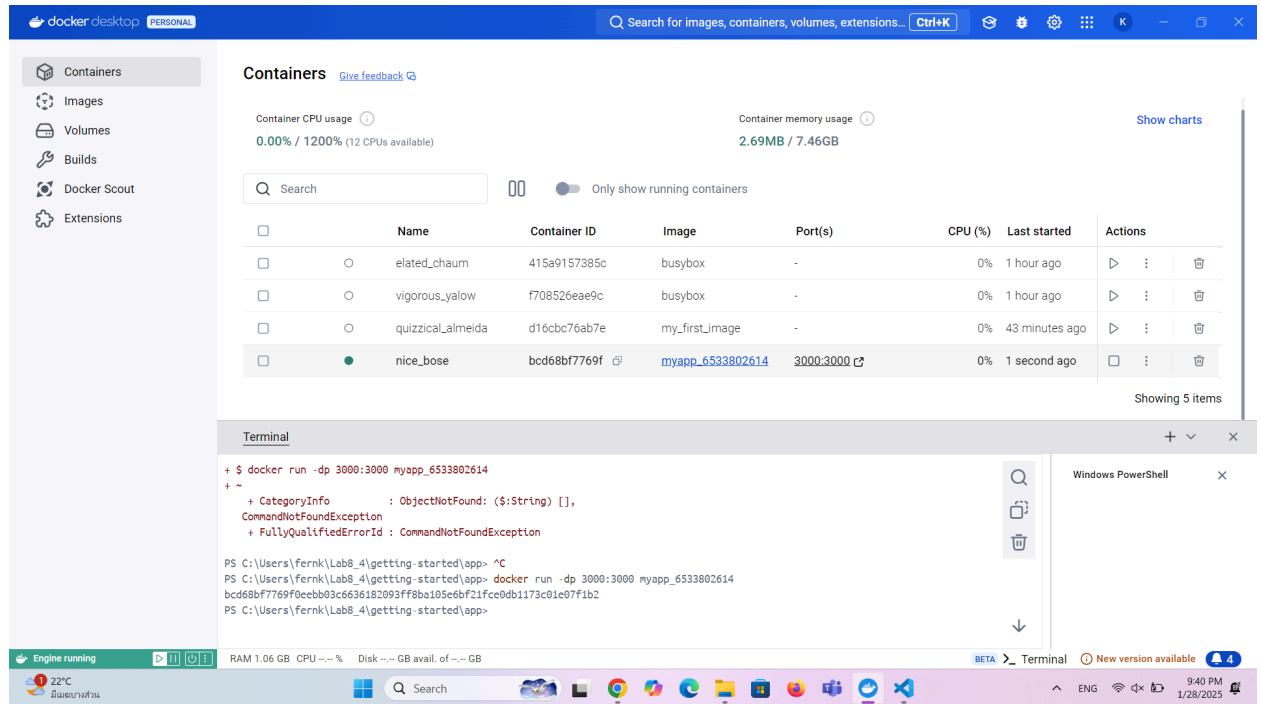
6. ทำการ Start ตัว Container ของแอปพลิเคชันที่สร้างขึ้น โดยใช้คำสั่ง

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

# CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตส์ฐา สุ่มเล็ก

## Lab Worksheet

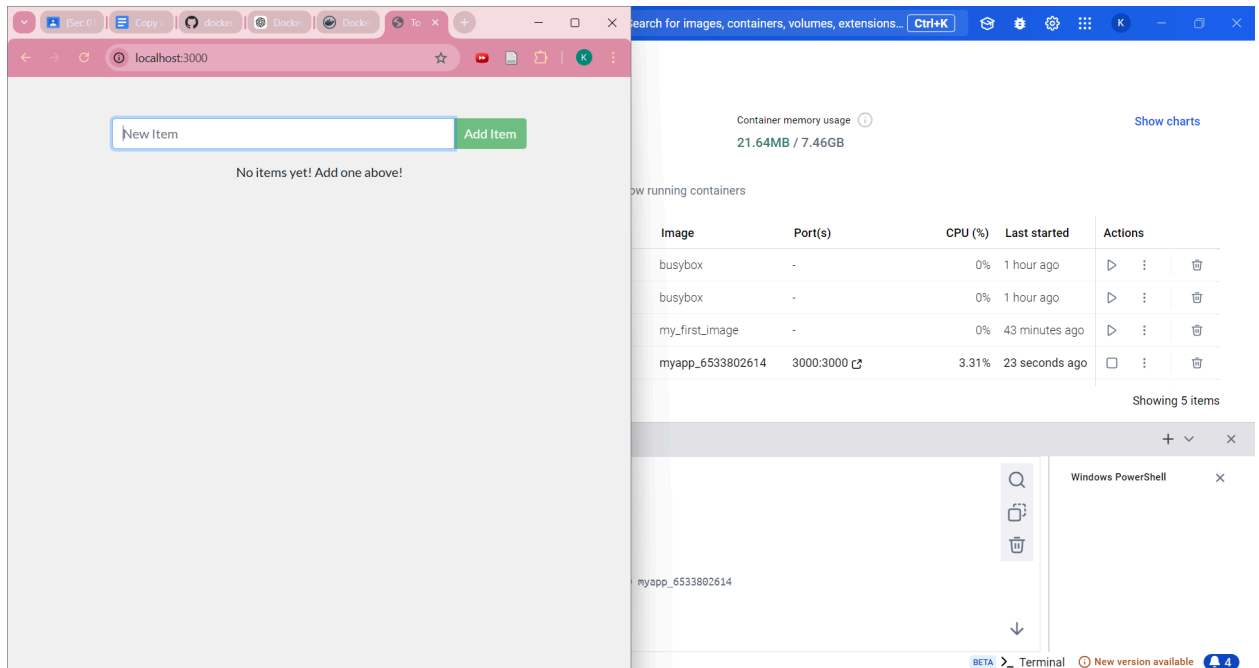


7. เปิด Browser ไปที่ URL = <http://localhost:3000>

# CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตสุธา สุ่มเล็ก

## Lab Worksheet



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

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

```
PS C:\Users\fernk\Lab8_4\getting-started\app> docker build -t myapp_6533802614 .
[+] Building 2.1s (10/10) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 154B
=> [internal] load metadata for docker.io/library/node:18-alpine
=> [auth] library/node:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/4] FROM docker.io/library/node:18-alpine@sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d975e9059dd066be3e274fbb25
=> [internal] load build context
=> => transferring context: 2.49kB
=> CACHED [2/4] WORKDIR /app
=> CACHED [3/4] COPY . .
=> CACHED [4/4] RUN yarn install --production
=> exporting to image
=> => exporting layers
=> => writing image sha256:8ee466feb9e26aaa28383f494094b1afd50f48f4ab73e212e21589acc323e16
=> => naming to docker.io/library/myapp_6533802614
```

What's next:

View a summary of image vulnerabilities and recommendations → `docker scout quickview`

### 10. Start และรัน Container ตัวใหม่ โดยใช้คำสั่งเดียวกันกับข้อ 6

```
PS C:\Users\fernk\Lab8_4\getting-started\app> docker run -dp 3000:3000 myapp_6533802614
c9b0c4d4eb0faad394c10d8abd05a36d89efa40d19ab2a10c1a5dfacbfd32b17
docker: Error response from daemon: driver failed programming external connectivity on endpoint silly_lalande (fa453060b42d89f4c8e03aad0ed9c0f6f2d00af02f9afe0f4342676609639bf7): Bind for 0.0.0.0:3000 failed: port is already allocated.
```

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

(1) Error ที่เกิดขึ้นหมายความว่าอย่างไร และเกิดขึ้นเพราะอะไร **port 3000** ที่พยายามใช้งานนั้นถูกใช้งานอยู่

### 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



# CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตสุธา สุ่มเล็ก

## Lab Worksheet

- iv. ใช้คำสั่ง `$ docker rm <Container ID ที่ต้องการจะลบ>` เพื่อทำการลบ

```
nutes 0.0.0.0:3000->3000/tcp nice_bose PS C:\Users\fern\Lab8_4\getting-started\app> Docker stop bcd68bf7769f bcd68bf7769f
```

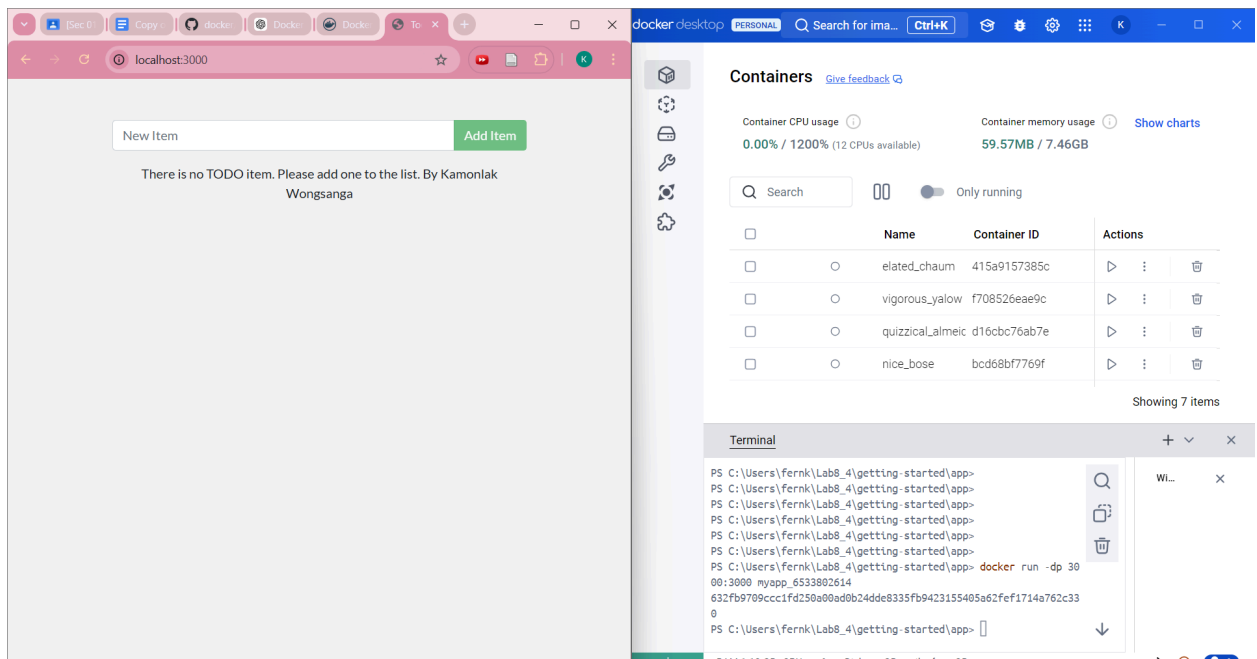
b. ผ่าน Docker desktop

i. ไปที่หน้าต่าง Containers

ii. เลือกไอคอนถังขยะในแถวของ Container ที่ต้องการจะลบ

iii. ยืนยันโดยการกด Delete forever

12. Start และรัน Container ตัวใหม่อีกครั้ง โดยใช้คำสั่งเดียวกันกับข้อ 6

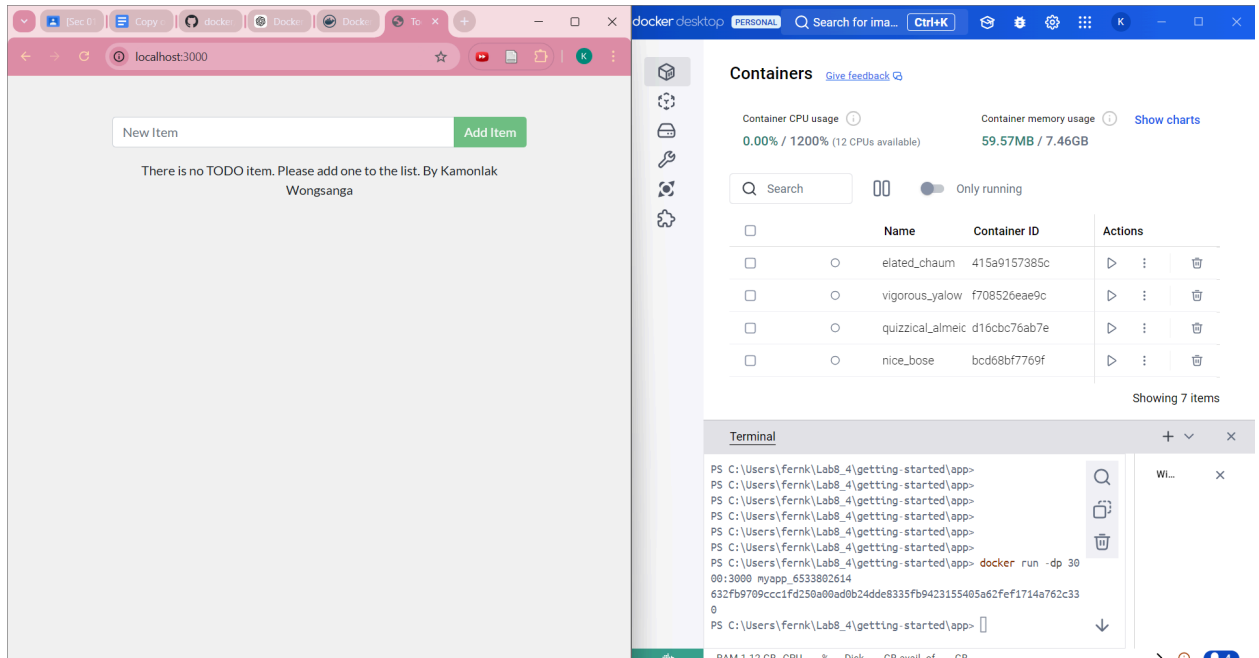


13. เปิด Browser ไปที่ URL = <http://localhost:3000>

# CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตสุธา สุ่มเล็ก

## Lab Worksheet



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

## แบบฝึกปฏิบัติที่ 8.5: เริ่มต้นสร้าง Pipeline อย่างง่ายสำหรับการ Deploy ด้วย Jenkins

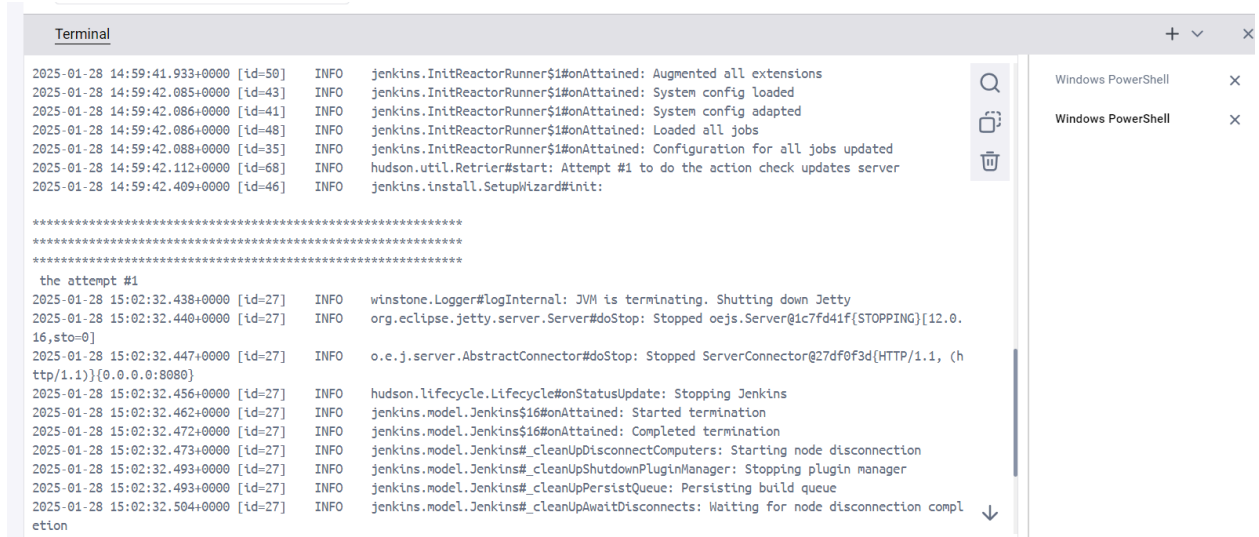
1. เปิด Command line หรือ Terminal บน Docker Desktop
2. ป้อนคำสั่งและทำการรัน container โดยผูกพอร์ต  
`$ docker run -p 8080:8080 -p 50000:50000 --restart=on-failure jenkins/jenkins:lts-jdk17`  
หรือ  
`$ docker run -p 8080:8080 -p 50000:50000 --restart=on-failure -v jenkins_home:/var/jenkins_home`

# CP353004/SC313 004 Software Engineering (2/2567)

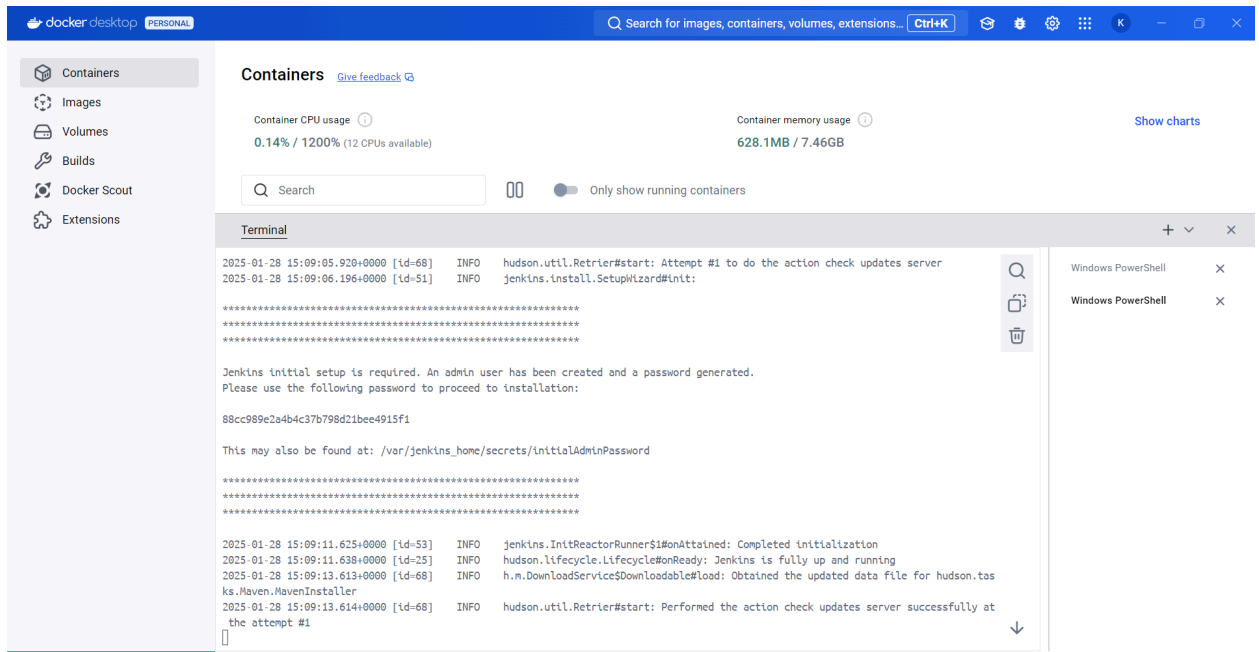
ผศ.ดร.ชิตสุธา สุ่มเล็ก

## Lab Worksheet

jenkins/jenkins:lts-jdk17



### 3. บันทึกรหัสผ่านของ Admin user ไว้สำหรับ log-in ในครั้งแรก



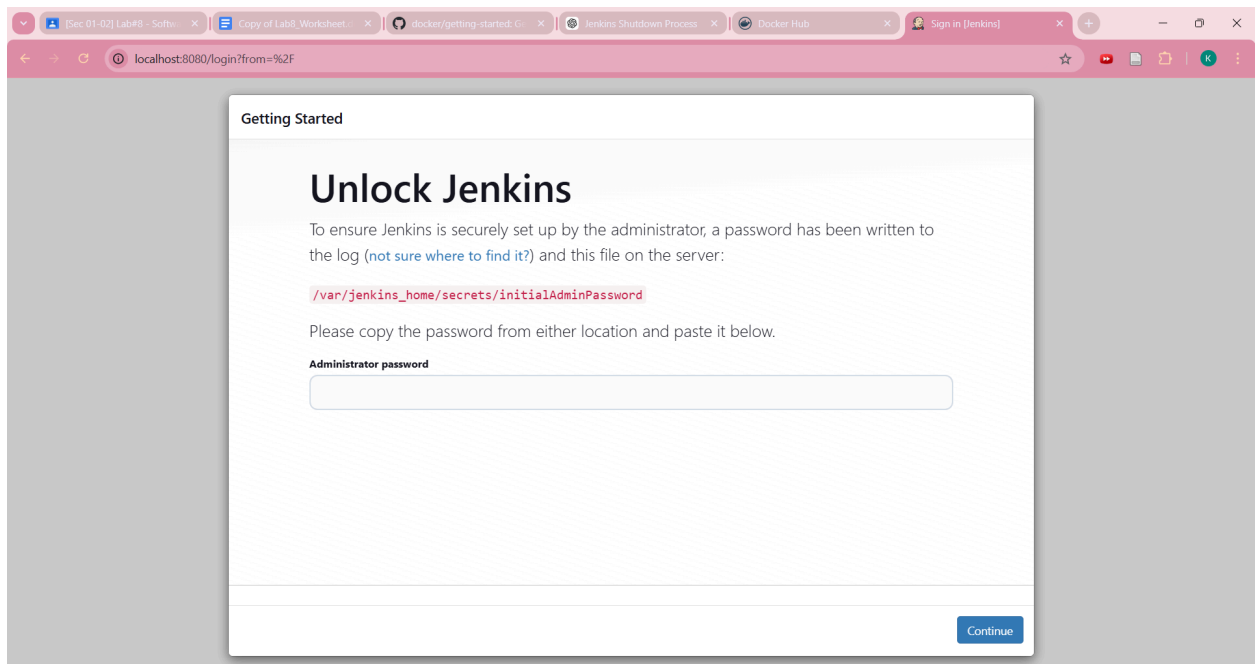
### [Check point#12] Capture หน้าจอที่แสดงผล Admin password

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

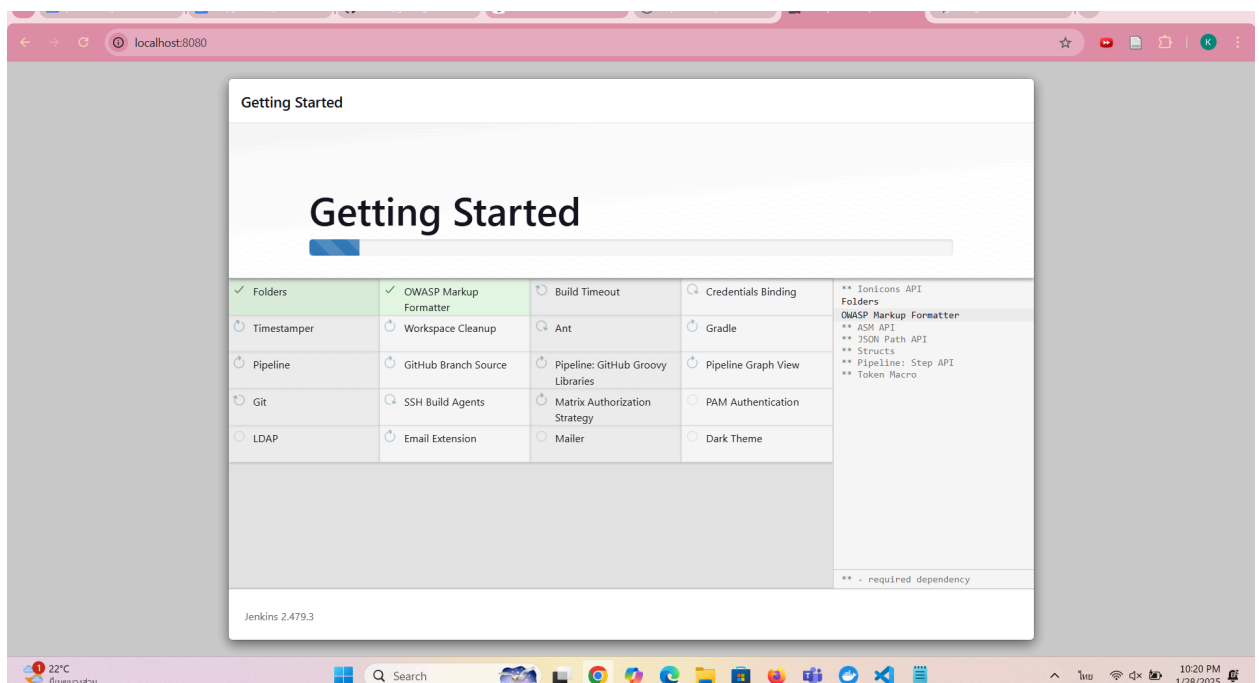
# CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตสุธา สุ่มเล็ก

## Lab Worksheet



### 5. ทำการ Unlock Jenkins ด้วยรหัสผ่านที่ได้ในข้อที่ 3



### 6. สร้าง Admin User โดยใช้ username เป็นชื่อจริงของนักศึกษา พร้อมรหัสสี่ตัวท้าย เช่น somsri\_3062

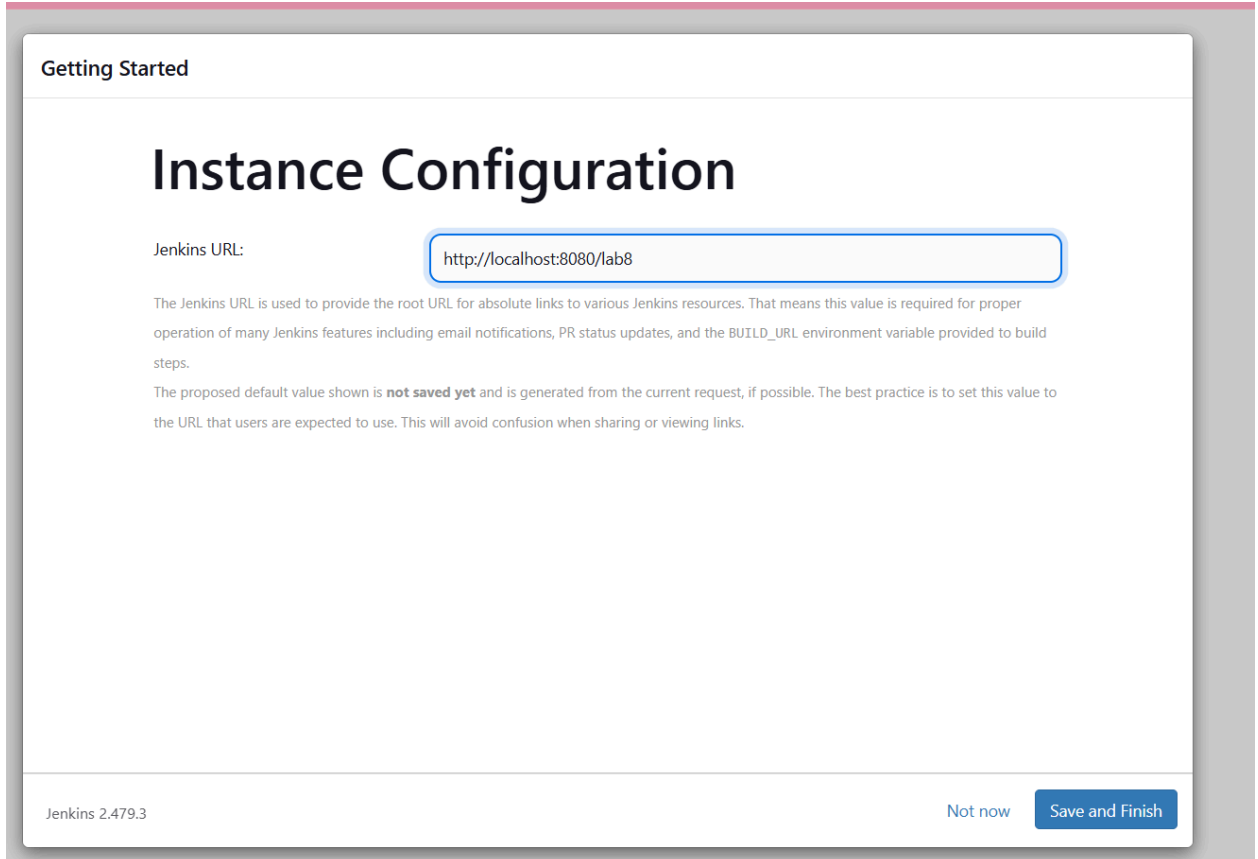
**[Check point#13] Capture หน้าจอที่แสดงผลการตั้งค่า**

## CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตสุธา สุ่มเล็ก

### Lab Worksheet

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



The screenshot shows the 'Getting Started' section of the Jenkins Instance Configuration page. The title 'Instance Configuration' is prominently displayed. Below it, the 'Jenkins URL' field is pre-filled with 'http://localhost:8080/lab8'. A detailed explanation of the Jenkins URL's purpose is provided, along with a note about the default value. At the bottom, there are two buttons: 'Not now' and 'Save and Finish'.

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.

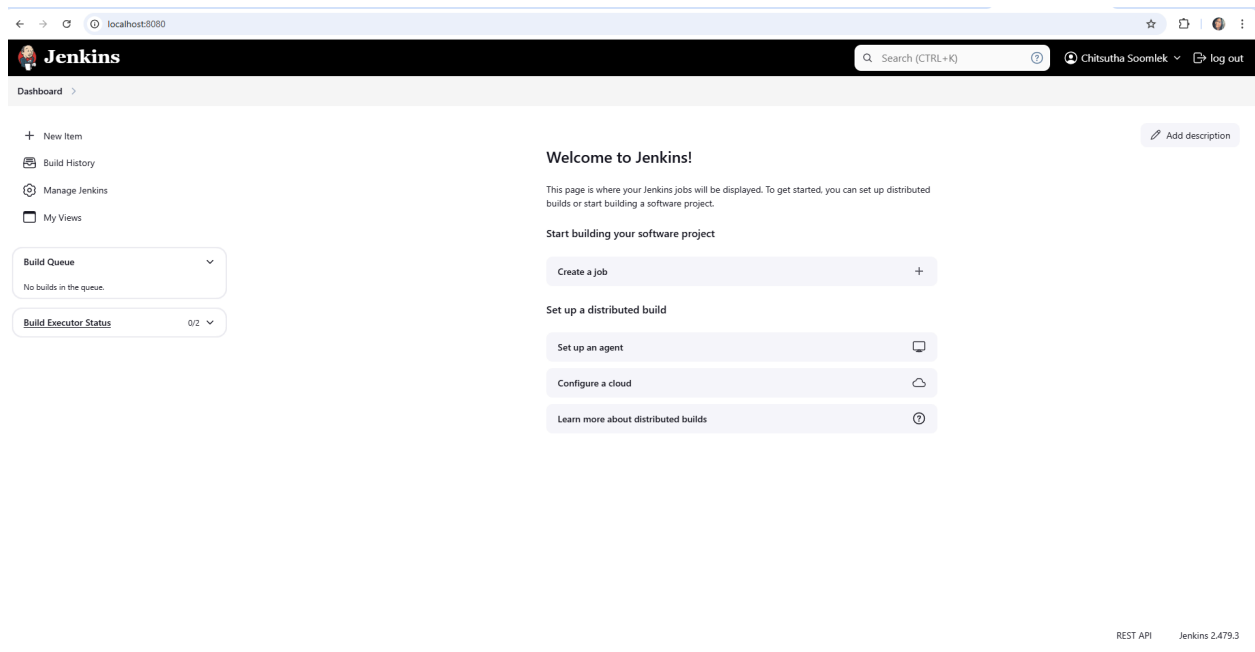
Jenkins 2.479.3 Not now Save and Finish

8. เมื่อติดตั้งเรียบร้อยแล้วจะพบกันหน้า Dashboard ดังแสดงในภาพ

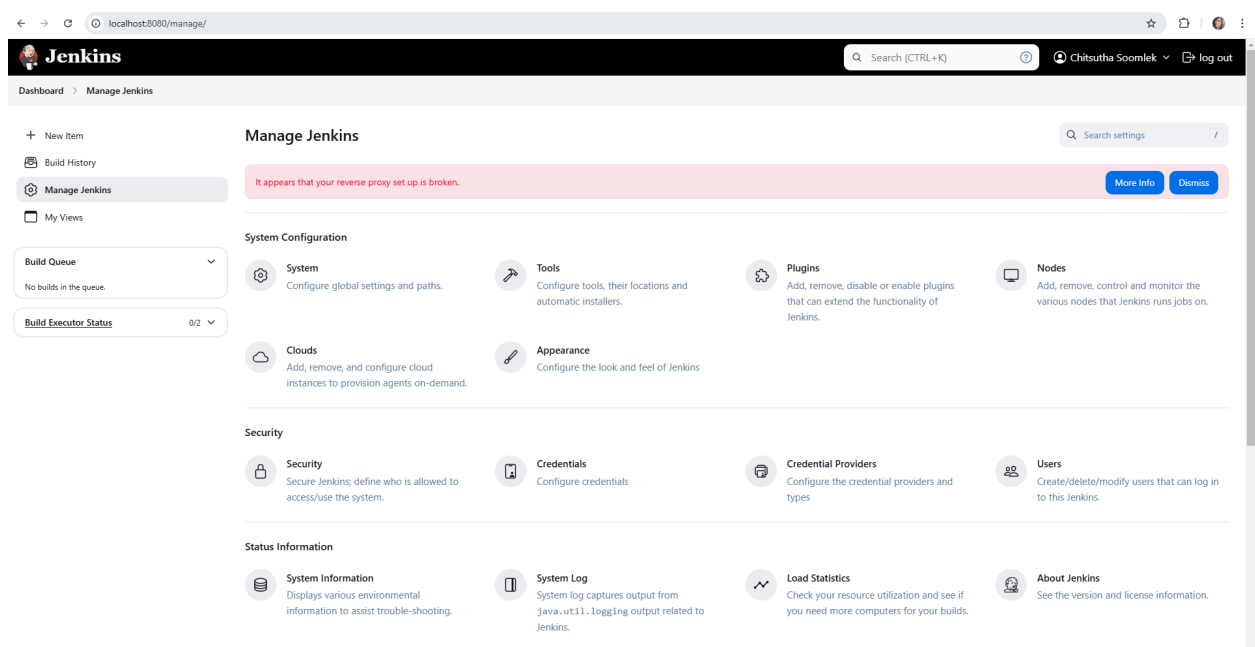
# CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตสุธา สุ่มเล็ก

## Lab Worksheet



## 9. เลือก Manage Jenkins แล้วไปที่เมนู Plugins



# CP353004/SC313 004 Software Engineering (2/2567)

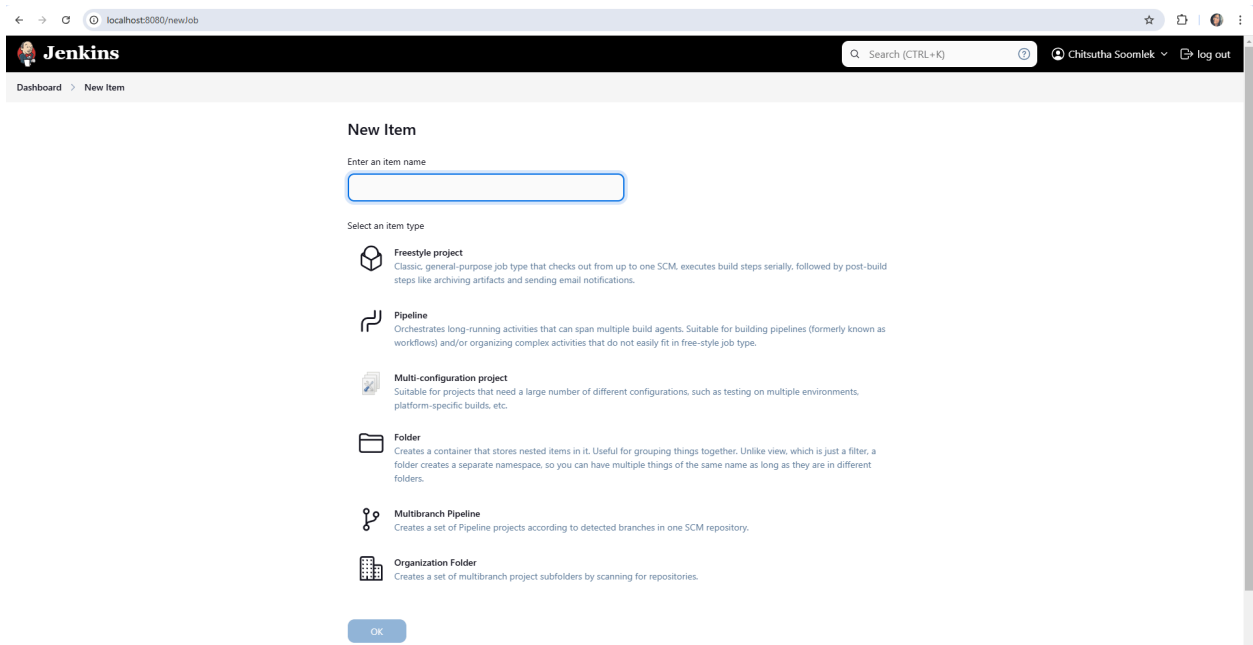
ผศ.ดร.ชิตสุธา สุ่มเล็ก

## 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

## CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตสุธา สุ่มเล็ก

### Lab Worksheet

**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 ไปถึงไฟล์ให้เรียบร้อยแล้ว)

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

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

---

---

---

**Post-build action:** เพิ่ม Publish Robot Framework test results -> ระบุไดเรกทอรีที่เก็บไฟล์ผลการทดสอบโดย Robot framework ในรูป xml และ html -> ตั้งค่า Threshold เป็น % ของการทดสอบที่ไม่ผ่านแล้ว นับว่าซอฟต์แวร์มีปัญหา -> 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