

## Lab Worksheet

ชื่อ-นามสกุล นายธนธรณ์ จุฬหลาย รหัสนักศึกษา 653380131-7 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 สำหรับกรณีที่เกิดปัญหา 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 พร้อมคำตอบคำถามต่อไปนี้

The screenshot shows a PowerShell terminal window and the Docker Desktop application interface. In the terminal, a directory named 'Lab8\_1' is created, and the 'busybox' Docker image is pulled. The Docker Desktop interface shows a list of local images, including 'learning\_style\_project-frontend', 'learning\_style\_project-backend', 'examai', 'busybox', and 'mongo'.

```
PowerShell 7.4.6
PS C:\Users\petch\Desktop\Software_Engineering\Lab8> mkdir Lab8_1

Directory: C:\Users\petch\Desktop\Software_Engineering\Lab8

Mode                LastWriteTime         Length Name
----                -
d-----          27/1/2568      21:13             Lab8_1

PS C:\Users\petch\Desktop\Software_Engineering\Lab8> cd Lab8_1
PS C:\Users\petch\Desktop\Software_Engineering\Lab8\Lab8_1> docker pull busybox
Using default tag: latest
latest: Pulling from library/busybox
Digest: sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138bd458fc8257fbf1
Status: Image is up to date for busybox:latest
docker.io/library/busybox:latest

What's next:
  View a summary of image vulnerabilities and recommendations → docker scout quickview busybox
PS C:\Users\petch\Desktop\Software_Engineering\Lab8\Lab8_1> docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
learning_style_project-frontend	latest	d265ed59dae6	3 months ago	1.68GB
learning_style_project-backend	latest	17c3falabcb4	3 months ago	147MB
examai	latest	2aa417ee7482	3 months ago	7.64GB
busybox	latest	af4709625109	4 months ago	4.27MB
mongo	4.4	d896c071ac69	11 months ago	427MB

PS C:\Users\petch\Desktop\Software\_Engineering\Lab8\Lab8\_1> |

The Docker Desktop interface shows the following details:

- Images:** Local tab selected. 5 images listed.
- Storage:** 2.25 GB / 9.89 GB in use.
- Last refresh:** 7 minutes ago.
- Image List:**

Name	Tag	Created	Size	Actions
learning_style_project-frontend	latest	4 months ago	1.67 GB	▶ ⋮ 🗑
learning_style_project-backend	latest	4 months ago	146.93 MB	▶ ⋮ 🗑
examai	latest	4 months ago	7.63 GB	▶ ⋮ 🗑
busybox	latest	4 months ago	4.26 MB	▶ ⋮ 🗑
mongo	4.4	11 months ago	427.29 MB	▶ ⋮ 🗑

## Lab Worksheet

(1) สิ่งที่อยู่ภายใต้คอนเทนเนอร์ Repository คืออะไร

ตอบ สิ่งที่อยู่ภายใต้คอนเทนเนอร์ Repository คือชื่อของอิมเมจ (Image) บน Docker ซึ่งใช้ระบุโปรเจกต์หรือแอปพลิเคชันที่อิมเมจนั้นถูกสร้างขึ้นมา เช่น learning\_style\_project-frontend, mongo หรือ busybox เป็นต้น

(2) Tag ที่ใช้บ่งบอกถึงอะไร

ตอบ Tag ใช้บ่งบอกถึงเวอร์ชันของอิมเมจ หรือสถานะของอิมเมจนั้น เช่น latest หมายถึงอิมเมจล่าสุด หรือหมายเลขเวอร์ชัน เช่น 4.4 เป็นต้น ซึ่งช่วยให้สามารถจัดการและเลือกใช้อิมเมจเวอร์ชันที่ต้องการได้ง่ายขึ้นใน Docker

5. ป้อนคำสั่ง \$ docker run busybox

6. ป้อนคำสั่ง \$ docker run -it busybox sh

7. ป้อนคำสั่ง ls

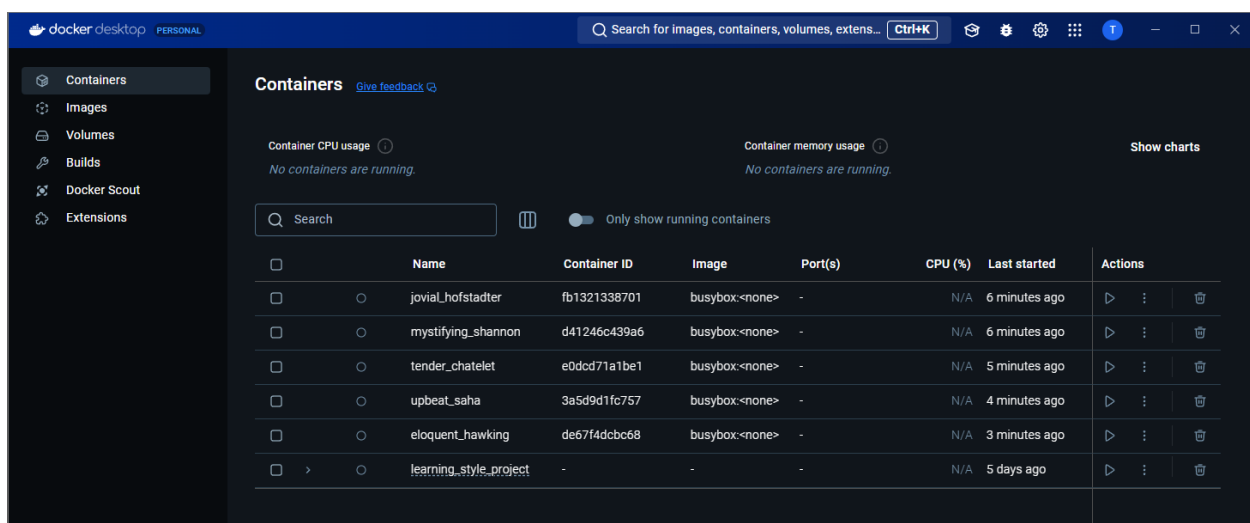
8. ป้อนคำสั่ง ls -la

9. ป้อนคำสั่ง exit

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

11. ป้อนคำสั่ง \$ docker ps -a

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



## Lab Worksheet

```

PS C:\Users\petch\Desktop\Software_Engineering\Lab8\Lab8_1> docker run busybox
PS C:\Users\petch\Desktop\Software_Engineering\Lab8\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 27 14:27 .
drwxr-xr-x 1 root root      4096 Jan 27 14:27 ..
-rwxr-xr-x 1 root root        0 Jan 27 14:27 .dockerenv
drwxr-xr-x 2 root root     12288 Sep 26 21:31 bin
drwxr-xr-x 5 root root      360 Jan 27 14:27 dev
drwxr-xr-x 1 root root      4096 Jan 27 14:27 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 27 14:27 proc
drwx----- 1 root root      4096 Jan 27 14:27 root
dr-xr-xr-x 11 root root        0 Jan 27 14:27 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\petch\Desktop\Software_Engineering\Lab8\Lab8_1> docker run busybox echo "Hello Thanathorn Chulay from busybox"
Hello Thanathorn Chulay from busybox
PS C:\Users\petch\Desktop\Software_Engineering\Lab8\Lab8_1> docker ps -a
CONTAINER ID   IMAGE          PORTS          NAMES                COMMAND                CREATED        STATUS
de67f4dc68    busybox       eloquent_hawking  "echo 'Hello Thanath..." 19 seconds ago  Exited (0)
18 seconds ago
3a5d9d1fc757  busybox       upbeat_saha      "sh"                        2 minutes ago  Exited (0)
About a minute ago
e0dcd71a1be1  busybox       tender_chatelet  "sh"                        2 minutes ago  Exited (0)
2 minutes ago
d41246c439a6  busybox       mystifying_shannon "sh"                        3 minutes ago  Exited (0)
2 minutes ago
fb1321338701  busybox       jovial_hofstadter "sh"                        4 minutes ago  Exited (0)
3 minutes ago
4c7343e0d128  learning_style_project-frontend "docker-entrypoint.s..." 3 months ago  Exited (1)
5 days ago
9346ebdf76cc  learning_style_project-backend  "uvicorn app.main:ap..." 3 months ago  Exited (0)
5 days ago
41f183a78ca8  mongo:4.4    "docker-entrypoint.s..." 3 months ago  Exited (0)
5 days ago
PS C:\Users\petch\Desktop\Software_Engineering\Lab8\Lab8_1> |

```

(1) เมื่อใช้ option -it ในคำสั่ง run ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป

ตอบ การใช้ -it ในคำสั่ง docker run ทำให้คอนเทนเนอร์ที่ถูกสร้างขึ้นสามารถโต้ตอบกับผู้ใช้ได้ใน

ลักษณะ Interactive และเปิด Terminal ภายในคอนเทนเนอร์ (TTY) เพื่อให้สามารถพิมพ์คำสั่งต่าง ๆ

ได้เหมือนกับการใช้งานในระบบปฏิบัติการจริง ตัวอย่างเช่น การเรียกใช้งาน sh หรือ bash เพื่อสำรวจ

โครงสร้างไฟล์ในคอนเทนเนอร์หรือทดสอบการทำงานบางอย่างในสภาพแวดล้อมนั้น

## Lab Worksheet

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

ตอบ แสดงสถานะของคอนเทนเนอร์แต่ละตัว โดยข้อมูลในคอลัมน์นี้บอกว่าคอนเทนเนอร์กำลังทำงานอยู่ (Up) หรือหยุดทำงานแล้ว (Exited) พร้อมด้วยระยะเวลาที่คอนเทนเนอร์หยุดหรือเริ่มทำงาน เช่น Exited (0) หมายถึงคอนเทนเนอร์หยุดทำงานเรียบร้อยแล้วโดยไม่มีข้อผิดพลาด (exit code = 0) และบอกเวลาที่เกิดการหยุดนั้น

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

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

The screenshot shows a PowerShell terminal window with the following commands and output:

```
PS C:\Users\petch\Desktop\Software_Engineering\Lab8\Lab8_1> docker rm fb1321338701f36f780b6a833e21849
97f88a3306c46c1fad4d26f30638de404
fb1321338701f36f780b6a833e2184997f88a3306c46c1fad4d26f30638de404
PS C:\Users\petch\Desktop\Software_Engineering\Lab8\Lab8_1> docker rm d41246c439a6707cec78f29608e728f
354fc3527f696b071489b15801c80ec2b
d41246c439a6707cec78f29608e728f354fc3527f696b071489b15801c80ec2b
PS C:\Users\petch\Desktop\Software_Engineering\Lab8\Lab8_1> docker rm e0dcd71a1be151ee9a5506e2f1dd3cb
2cfbc89a43aa089d8e13259dd9d14d7e7
e0dcd71a1be151ee9a5506e2f1dd3cb2cfbc89a43aa089d8e13259dd9d14d7e7
PS C:\Users\petch\Desktop\Software_Engineering\Lab8\Lab8_1> docker rm 3a5d9d1fc757064df3c201690433719
6f9c9b4de38299676c3aacf1375bca457
3a5d9d1fc757064df3c2016904337196f9c9b4de38299676c3aacf1375bca457
PS C:\Users\petch\Desktop\Software_Engineering\Lab8\Lab8_1> docker rm de67f4dcb687a0742e9fe247295f41
9119982e8da2104c1fb9f8c2846b558fe
de67f4dcb687a0742e9fe247295f419119982e8da2104c1fb9f8c2846b558fe
PS C:\Users\petch\Desktop\Software_Engineering\Lab8\Lab8_1> |
```

Below the terminal, the Docker Desktop interface is shown. The 'Containers' tab is active, displaying a table with one container:

Name	Container ID	Image	Port(s)	Actions
learning_style_project	-	-	-	Play, Stop, Delete

## 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 และบันทึกคำสั่งต่อไปนี้

```
$ 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

## Lab Worksheet

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

```

PowerShell
PS C:\Users\petch\Desktop\Software_Engineering\Lab8> cd Lab8_2
PS C:\Users\petch\Desktop\Software_Engineering\Lab8\Lab8_2> docker build -t lab8_2 .
[+] Building 0.1s (5/5) FINISHED                                docker:desktop-linux
x
=> [internal] load build definition from Dockerfile                                0.0
s
=> => transferring dockerfile: 264B                                              0.0
s
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior relat 0.0
s
=> WARN: MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the same sta 0.0
s
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior relat 0.0
s
=> [internal] load metadata for docker.io/library/busybox:latest                  0.0
s
=> [internal] load .dockerignore                                                  0.0
s
=> => transferring context: 2B                                                    0.0
s
=> [1/1] FROM docker.io/library/busybox:latest                                  0.0
s
=> exporting to image                                                            0.0
s
=> => exporting layers                                                            0.0
s
=> => writing image sha256:e2bfff7f38aeadd78a85eb202bc4a3d8df6c89478b368d6baa5c44753ea4c0ce35 0.0
s
=> => naming to docker.io/library/lab8_2                                         0.0
s

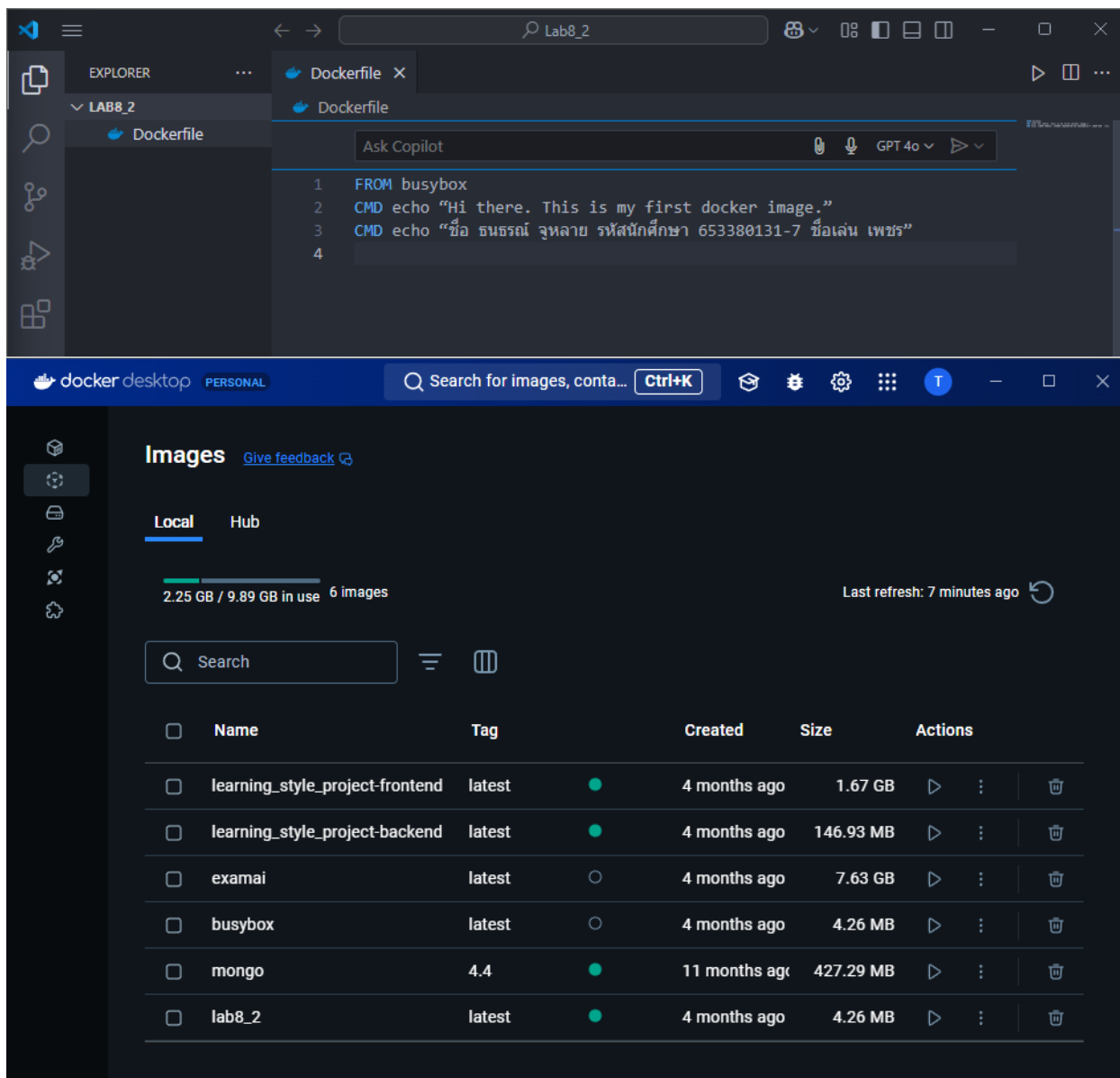
View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/s210kvcj1bmy9w0h3tdn9pxr5

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

What's next:
View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\Users\petch\Desktop\Software_Engineering\Lab8\Lab8_2> docker run lab8_2
" ชื่อ นรณณ์ จุหลาย รหัสนักศึกษา 653380131-7 ชื่อเล่น เพชร"
PS C:\Users\petch\Desktop\Software_Engineering\Lab8\Lab8_2> |

```

## Lab Worksheet



(1) คำสั่งที่ใช้ในการ run คือ

ตอบ `docker run lab8_2`

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

ตอบ Option -t (หรือ --tag) ในคำสั่ง `docker build` ใช้สำหรับตั้งชื่อ (Name) และแท็ก (Tag) ให้กับ Docker Image ที่สร้างขึ้น เพื่อให้ง่ายต่อการระบุและเรียกใช้งานในภายหลัง ตัวอย่างเช่น การใช้ `-t lab8_2` จะตั้งชื่อ Image ว่า lab8\_2 หากไม่ใช้ -t ระบบจะสร้าง Image แต่ไม่ได้ตั้งชื่อ ซึ่งจะทำให้ระบุ Image ได้ยาก (เพราะต้องใช้ Image ID ที่สร้างแบบสุ่ม)



## Lab Worksheet

## แบบฝึกปฏิบัติที่ 8.3: การแชร์ Docker image ผ่าน Docker Hub

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

สำหรับเครื่องที่ใช้ระบบปฏิบัติการวินโดวส์ บันทึกคำสั่งต่อไปนี้ลงในไฟล์ โดยใช้ Text Editor ที่มี

```
FROM busybox
```

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

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

สำหรับเครื่องที่ใช้ระบบปฏิบัติการ MacOS หรือ Linux บนหน้าต่าง Terminal และป้อนคำสั่งต่อไปนี้

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

```
FROM busybox
```

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

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

```
EOF
```

หรือใช้คำสั่ง

```
$ touch Dockerfile
```

แล้วใช้ Text Editor ในการใส่เนื้อหาแทน

7. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้

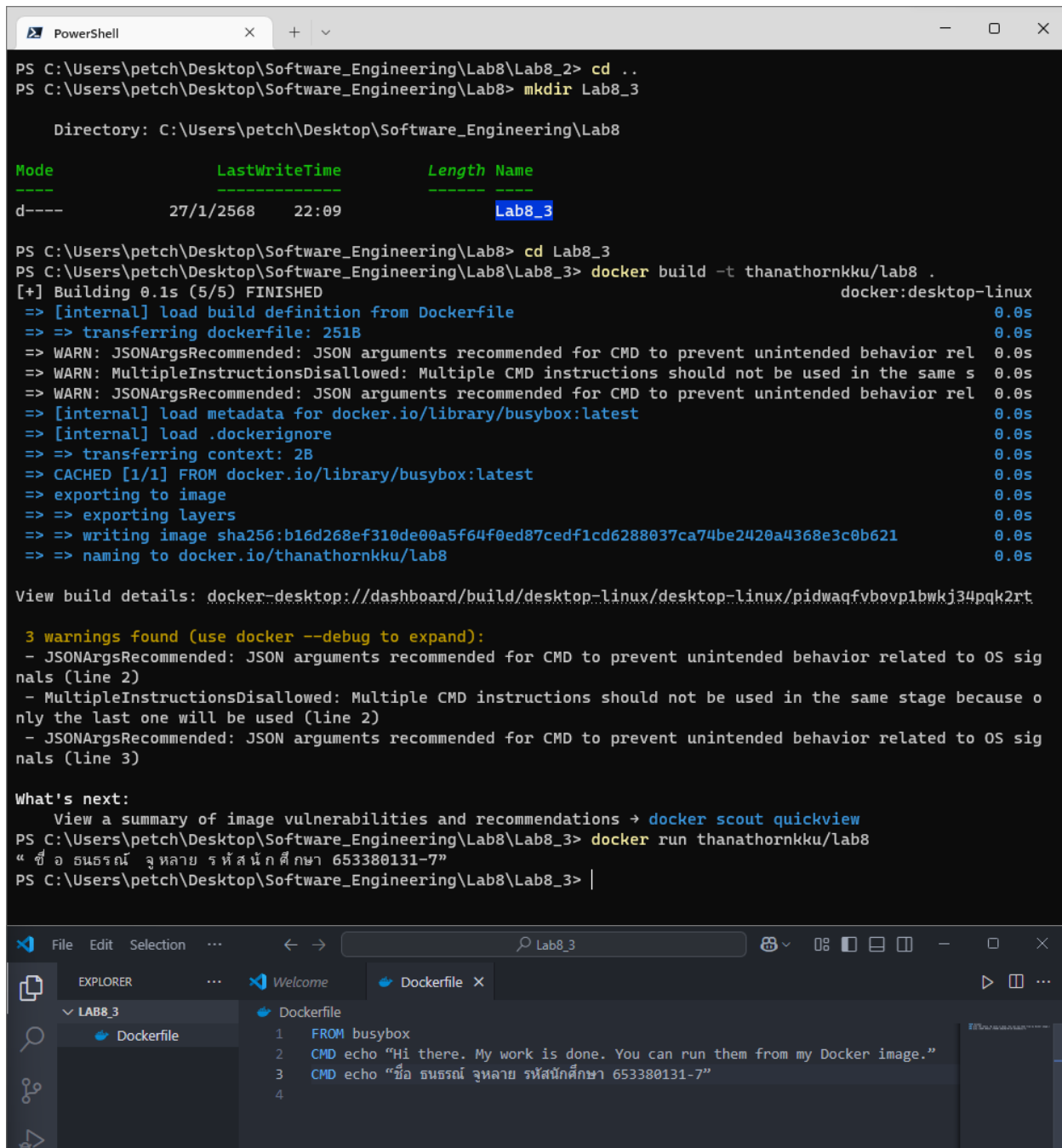
```
$ docker build -t <username ที่ลงทะเบียนกับ Docker Hub>/lab8
```

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



```

PowerShell
PS C:\Users\petch\Desktop\Software_Engineering\Lab8\Lab8_2> cd ..
PS C:\Users\petch\Desktop\Software_Engineering\Lab8> mkdir Lab8_3

Directory: C:\Users\petch\Desktop\Software_Engineering\Lab8

Mode                LastWriteTime         Length Name
----                -
d-----          27/1/2568    22:09             Lab8_3

PS C:\Users\petch\Desktop\Software_Engineering\Lab8> cd Lab8_3
PS C:\Users\petch\Desktop\Software_Engineering\Lab8\Lab8_3> docker build -t thanathornkku/lab8 .
[+] Building 0.1s (5/5) FINISHED                                docker:desktop-linux
=> [internal] load build definition from Dockerfile             0.0s
=> => transferring dockerfile: 251B                             0.0s
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior rel 0.0s
=> WARN: MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the same s 0.0s
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior rel 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:b16d268ef310de00a5f64f0ed87cedf1cd6288037ca74be2420a4368e3c0b621 0.0s
=> => naming to docker.io/thanathornkku/lab8                     0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/pidwaqfvybovp1bwkj34pqk2rt

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

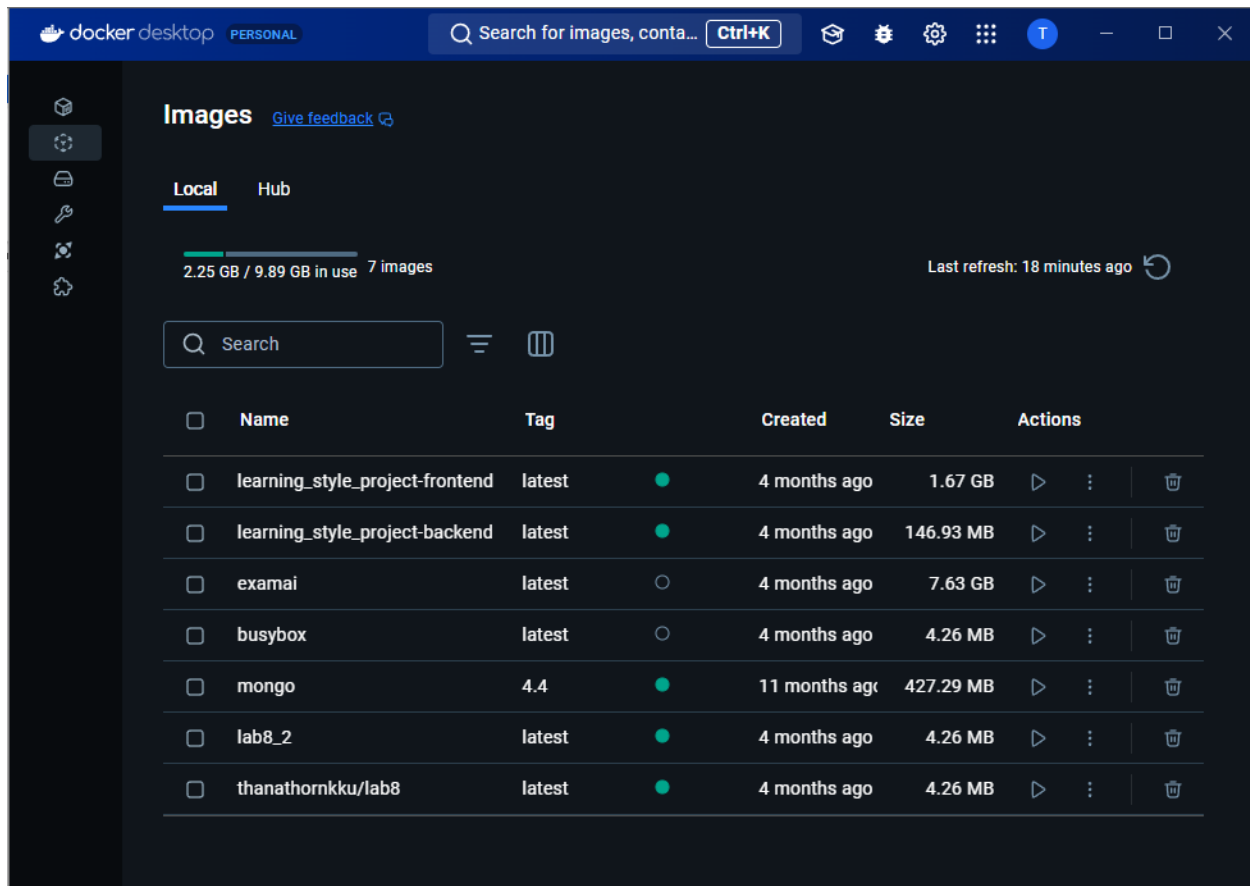
What's next:
View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\Users\petch\Desktop\Software_Engineering\Lab8\Lab8_3> docker run thanathornkku/lab8
" ชื่อ ธนธรณ์ จุฬาลาย รหัสนักศึกษ 653380131-7"
PS C:\Users\petch\Desktop\Software_Engineering\Lab8\Lab8_3> |

EXPLORER
LAB8_3
  Dockerfile

Dockerfile
1 FROM busybox
2 CMD echo "Hi there. My work is done. You can run them from my Docker image."
3 CMD echo "ชื่อ ธนธรณ์ จุฬาลาย รหัสนักศึกษ 653380131-7"
4

```

## Lab Worksheet



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

## Lab Worksheet

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

The screenshot shows a PowerShell terminal window and a web browser displaying the Docker Hub repository page for 'thanathornkku/lab8'.

**PowerShell Terminal Output:**

```
PS C:\Users\petch\Desktop\Software_Engineering\Lab8\Lab8_3> docker push thanathornkku/lab8
Using default tag: latest
The push refers to repository [docker.io/thanathornkku/lab8]
59654b79daad: Mounted from library/busybox
latest: digest: sha256:262ef9a19f66becdf8250efc8d7462f719420f65d268486c1c8657e9fbfe0f21 size: 527
PS C:\Users\petch\Desktop\Software_Engineering\Lab8\Lab8_3> |
```

**Docker Hub Repository Page:**

The page shows the repository 'thanathornkku/lab8' with the following details:

Name	Last Pushed	Contains	Visibility	Scout
thanathornkku/lab8	6 minutes ago	IMAGE	Public	Inactive

**Docker Desktop Interface:**

The Docker Desktop interface shows the 'Images' section with the 'Hub' tab selected. The repository 'thanathornkku/lab8' is listed with the following details:

Tags	OS	Vulnerabilities	Last pushed	Size
latest		Inactive	8 minutes ago	2.15 MB

## Lab Worksheet

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

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

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

The screenshot shows a Windows File Explorer window titled 'Lab8\_4' with the address bar showing the path 'Desktop > Software\_Engineering > Lab8 > Lab8\_4'. The main pane shows a folder named 'getting-started' with a last modified time of '27/1/2568 22:31'. Below the File Explorer is a PowerShell terminal window with the following commands and output:

```
PS C:\Users\petch\Desktop\Software_Engineering\Lab8> mkdir Lab8_4

Directory: C:\Users\petch\Desktop\Software_Engineering\Lab8

Mode                LastWriteTime         Length Name
----                -
d-----          27/1/2568     22:30             Lab8_4

PS C:\Users\petch\Desktop\Software_Engineering\Lab8> cd Lab8_4
PS C:\Users\petch\Desktop\Software_Engineering\Lab8\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 | 2.82 MiB/s, done.
Resolving deltas: 100% (523/523), done.
PS C:\Users\petch\Desktop\Software_Engineering\Lab8\Lab8_4> |
```

## Lab Worksheet

The screenshot shows a VS Code editor with the Explorer sidebar on the left displaying the file structure of a project named 'LAB8\_4'. The file structure includes a 'getting-started' directory with sub-directories '.github', 'app', 'spec', and 'src'. The 'app' directory contains 'package.json', 'yarn.lock', 'docs', '.dockerignore', '.gitignore', 'build.sh', 'docker-compose.yml', 'Dockerfile', 'LICENSE', 'mkdocs.yml', 'README.md', and 'requirements.txt'. The main editor window shows the 'package.json' file for the 'app' directory. The file content is as follows:

```

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

```

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

## Lab Worksheet

[Check point#8] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง)

แสดงคำสั่งและผลลัพธ์ที่ได้ทางหน้าจอ

The screenshot shows a PowerShell terminal window and a VS Code editor window. The PowerShell window displays the output of a Docker build command, showing the progress of building the image 'myapp\_6533801317'. The output includes details about the build definition, metadata, and the final image export. The VS Code editor shows the file explorer with the 'Lab8\_4' directory selected, containing files like 'getting-started', 'app', 'spec', 'src', 'Dockerfile', 'package.json', 'yarn.lock', 'docs', '.dockerignore', '.gitignore', 'build.sh', 'docker-compose.yml', 'LICENSE', 'mkdocs.yml', 'README.md', and 'requirements.txt'.

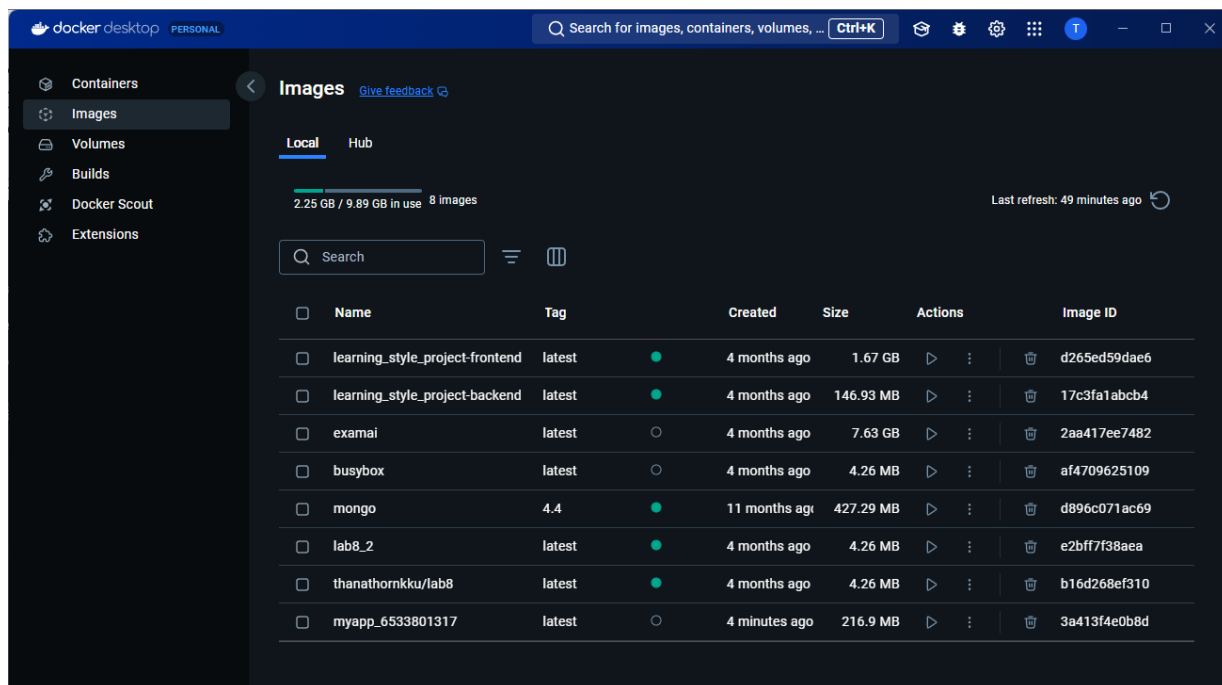
```

PS C:\Users\petch\Desktop\Software_Engineering\Lab8\Lab8_4> cd getting-started/app
PS C:\Users\petch\Desktop\Software_Engineering\Lab8\Lab8_4\getting-started\app> docker build -t myapp_6533801317 .
[+] Building 44.0s (10/10) FINISHED                                docker:desktop-linux
=> [internal] load build definition from Dockerfile                0.0s
=> => transferring dockerfile: 156B                                0.0s
=> [internal] load metadata for docker.io/library/node:18-alpine  4.2s
=> [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@sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d975e9059dd066b 21.8s
=> => resolve docker.io/library/node:18-alpine@sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d975e9059dd066b 0.0s
=> => sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d975e9059dd066b3e274fbb25 7.67kB / 7.67kB 0.0s
=> => sha256:6e804119c3884fc5782795bf0d2adc89201c63105aece8647b17a7bcebbbc385e 1.72kB / 1.72kB 0.0s
=> => sha256:dcbf7b337595be6f4d214e4eed84f230eeffe0e4ac03a50380d573e289b9e5e40 6.18kB / 6.18kB 0.0s
=> => sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c24403d592134f0 3.64MB / 3.64MB 2.0s
=> => sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e0da02525e63167c 40.01MB / 40.01MB 20.2s
=> => sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15bf426d01bc1 1.26MB / 1.26MB 2.2s
=> => extracting sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c24403d592134f0 0.1s
=> => sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce368990ca771 444B / 444B 2.6s
=> => extracting sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e0da02525e63167c 1.3s
=> => 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 . .                                               0.0s
=> [4/4] RUN yarn install --production                          16.6s
=> exporting to image                                           0.9s
=> => exporting layers                                             0.9s
=> => writing image sha256:3a413f4e0b8dc6a98a29beb1c6cb8f4eb06b8de9c554e731aaa121386a0b0760 0.0s
=> => naming to docker.io/library/myapp_6533801317              0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/yfm24i3qbun9f4n235byyqzj1

What's next:
View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\Users\petch\Desktop\Software_Engineering\Lab8\Lab8_4\getting-started\app>
  
```

## Lab Worksheet

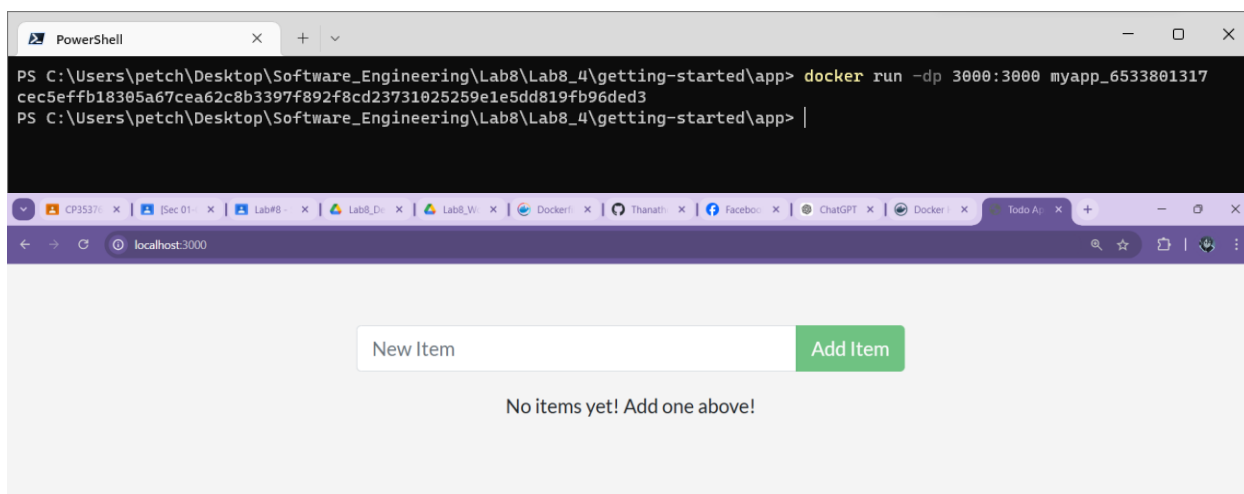


6. ทำการ Start ตัว Container ของแอปพลิเคชันที่สร้างขึ้น โดยใช้คำสั่ง

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

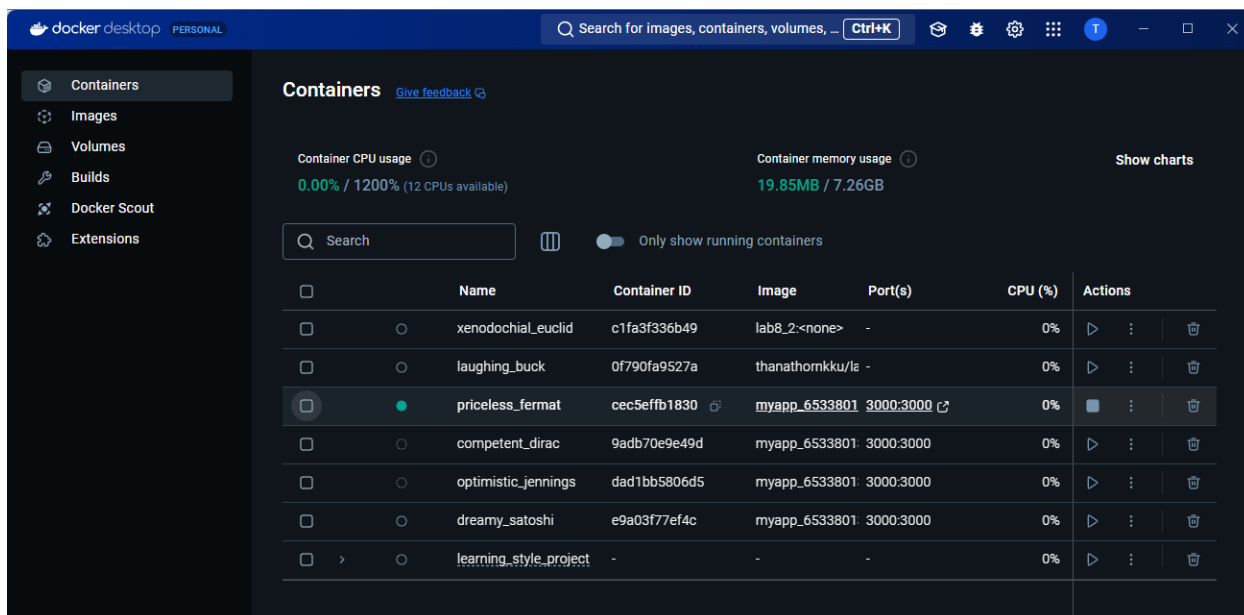
7. เปิด Browser ไปที่ URL = <http://localhost:3000>

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





## Lab Worksheet



\*\*\*เนื่องจากก่อนหน้านี้อัปเดตบันทึกภาพใน Docker Desktop จึงติด Container เวอร์ชันที่แก้ไขแล้วมาด้วย\*\*\*

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

The image shows a VS Code editor with a file explorer on the left, a code editor in the center, and a PowerShell terminal at the bottom. The file explorer shows a project structure for 'Lab8\_4' with folders like 'getting-started', 'github', 'app', 'spec', 'src', 'persistence', 'routes', 'static', 'css', and 'js'. The code editor shows a file named 'app.js' with a React component 'TodoListCard' and a function 'AddItemForm'. The PowerShell terminal shows the execution of Docker commands to build and run a container.

```

function TodoListCard() {
  const onItemUpdate = React.useCallback(
    // ...
  );

  const onItemRemoval = React.useCallback(
    // ...
  );

  if (items === null) return 'Loading...';

  return (
    <React.Fragment>
      <AddItemForm onNewItem={onNewItem} />
      {items.length === 0 && (
        // <p className="text-center">No items yet! Add one above!</p>
        <p className="text-center">There is no TODO item. Please add one to the list. By ชิตสุภา สุ่มเล็ก</p>
      )}
      {items.map(item => (
        <ItemDisplay
          item={item}
          key={item.id}
          onItemUpdate={onItemUpdate}
          onItemRemoval={onItemRemoval}
        />
      ))}
    </React.Fragment>
  );
}

function AddItemForm({ onNewItem }) {
  const { Form, InputGroup, Button } = ReactBootstrap;
  const [newItem, setNewItem] = React.useState('');
}

```

```

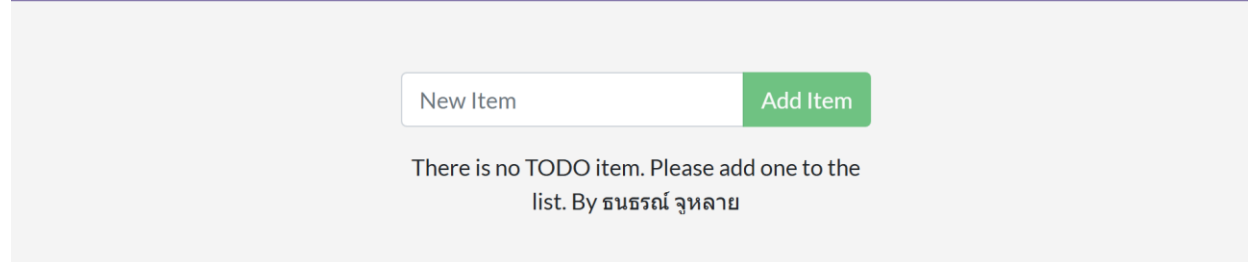
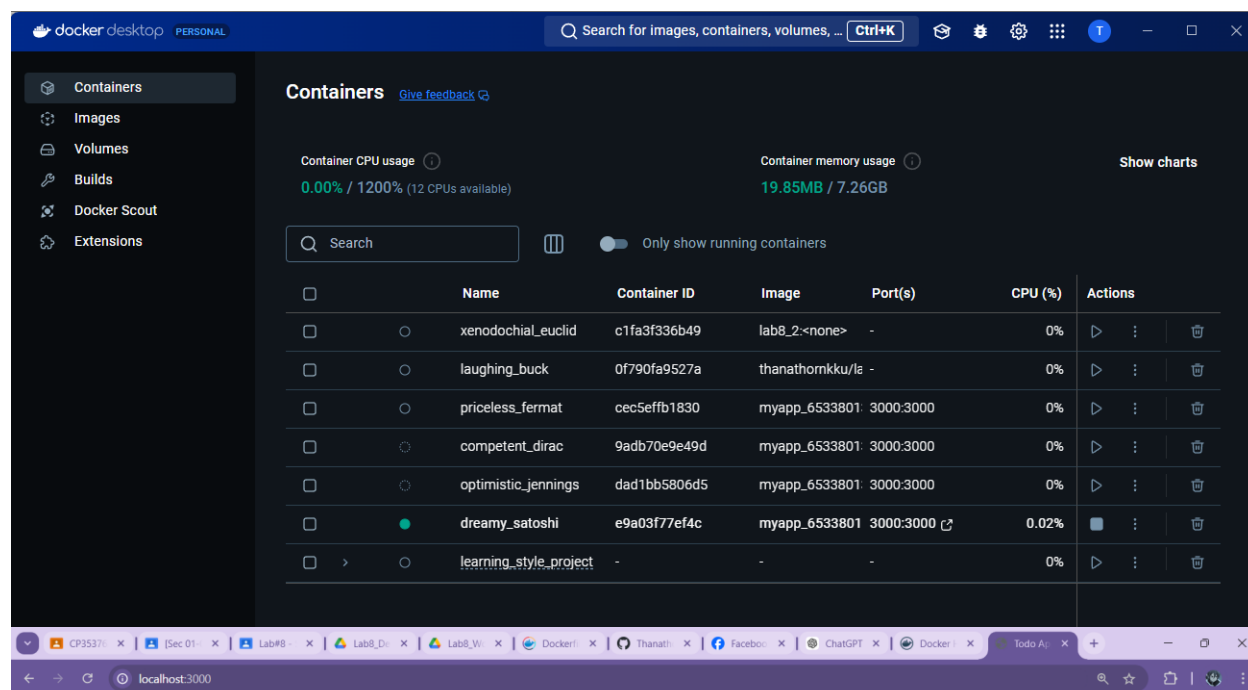
PS C:\Users\petch\Desktop\Software_Engineering\Lab8\Lab8_4\getting-started\app> docker run -dp 3000:3000 myapp_6533801317
cec5effb18305a67cea62c8b3397f892f8cd23731025259e1e5dd819fb96ded3
PS C:\Users\petch\Desktop\Software_Engineering\Lab8\Lab8_4\getting-started\app> docker build -t myapp_6533801317 .
[+] Building 23.4s (10/10) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 156B
=> [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@sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d975e9059dd066be3e274fb
=> [internal] load build context
=> => transferring context: 8.19kB
=> CACHED [2/4] WORKDIR /app
=> [3/4] COPY . .
=> [4/4] RUN yarn install --production
=> exporting to image
=> => exporting layers
=> writing image sha256:3a97ad809589441167af87eebe7d9d4085c3d6b7db502c71ac852e30b1ededc6
=> naming to docker.io/library/myapp_6533801317

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/ijzonkivn64zp3zdrwf7b5ehg

What's next:
View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\Users\petch\Desktop\Software_Engineering\Lab8\Lab8_4\getting-started\app> docker run -dp 3000:3000 myapp_6533801317
9adb70e9e49d1af99e3d610bdfaf3cbaaaf9a196ec9412f659ac9d78a8cbdded
docker: Error response from daemon: driver failed programming external connectivity on endpoint competent_dirac (0747ald32d1e4baaf9c6fbd3a76dbe4615b12b7a47772915d28b937d43cee4eb): Bind for 0.0.0.0:3000 failed: port is already allocated.
PS C:\Users\petch\Desktop\Software_Engineering\Lab8\Lab8_4\getting-started\app> docker run -dp 3000:3000 myapp_6533801317
dad1bb5806d5d4b85f10c6b0526c3dd4ed8fc5e30f0fa7d3316a632fbc3895ce
docker: Error response from daemon: driver failed programming external connectivity on endpoint optimistic_jennings (b257767dd6dc09dddf5d647a09e0d7748148c6add098abc5723db30675ada35cd): Bind for 0.0.0.0:3000 failed: port is already allocated.
PS C:\Users\petch\Desktop\Software_Engineering\Lab8\Lab8_4\getting-started\app> docker run -dp 3000:3000 myapp_6533801317
e9a03f77ef4c24b6c9e8b5641b13b3456cfa335596138748a33a8049ed78c508
PS C:\Users\petch\Desktop\Software_Engineering\Lab8\Lab8_4\getting-started\app>

```

## Lab Worksheet



(1) Error ที่เกิดขึ้นหมายความว่าอย่างไร และเกิดขึ้นเพราะอะไร

ตอบ หมายความว่า พอร์ต 3000 บนเครื่อง Host (0.0.0.0:3000) ถูกใช้งานอยู่แล้วโดย Container อื่น หรือแอปพลิเคชันอื่น ทำให้ Docker ไม่สามารถผูก (bind) พอร์ตนี้กับ Container ใหม่ได้ และเกิดขึ้นเพราะ Container ชื่อ priceless\_fermat กำลังทำงานอยู่และใช้พอร์ต 3000:3000 ดังนั้นจำเป็นต้องหยุดการทำงานของ Container ชื่อ priceless\_fermat ทำให้หลังจากนั้นสามารถใช้คำสั่ง run และแสดง Web application ที่แก้ไขแล้วได้

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

## Lab Worksheet

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) แสดงผลลัพธ์ที่ได้นบน Browser และ Dashboard ของ Docker desktop

The image shows a PowerShell terminal window and the Docker Desktop application interface.

**PowerShell Terminal:**

```
PS C:\Users\petch\Desktop\Software_Engineering\Lab8\Lab8_4\getting-started\app> docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS        NAMES
e9a03f77ef4c   myapp_6533801317 "docker-entrypoint.s..." 24 minutes ago Exited (0) 14 seconds ago           dreamy_satoshi
dad1bb5806d5   myapp_6533801317 "docker-entrypoint.s..." 25 minutes ago Created                               optimistic_jennings
9adb70e9e49d   myapp_6533801317 "docker-entrypoint.s..." 25 minutes ago Created                               competent_dirac
cec5effb1830   3a413f4e0b8d    "/bin/sh -c 'echo "e..." 33 minutes ago Exited (0) 8 minutes ago           priceless_fermat
0f790fa9527a   thanathornkku/lab8_2 "/bin/sh -c 'echo "e..." About an hour ago Exited (0) About an hour ago           laughing_buck
c1fa3f336b49   lab8_2          "/bin/sh -c 'echo "e..." About an hour ago Exited (0) About an hour ago           xenodochial_euclid
4c7343e0d128   learning_style_project-frontend "docker-entrypoint.s..." 3 months ago   Exited (1) 5 days ago           learning_style_project-frontend-1
9346ebdf76cc   learning_style_project-backend "uvicorn app.main:ap..." 3 months ago   Exited (0) 5 days ago           learning_style_project-backend-1
41f183a78ca8   mongo:4.4       "docker-entrypoint.s..." 3 months ago   Exited (0) 5 days ago           learning_style_project-mongo-1

PS C:\Users\petch\Desktop\Software_Engineering\Lab8\Lab8_4\getting-started\app> docker stop cec5effb18305a67cea62c8b3397f892f8cd23731025259e1e5dd819fb96ded3
PS C:\Users\petch\Desktop\Software_Engineering\Lab8\Lab8_4\getting-started\app> docker rm cec5effb18305a67cea62c8b3397f892f8cd23731025259e1e5dd819fb96ded3
PS C:\Users\petch\Desktop\Software_Engineering\Lab8\Lab8_4\getting-started\app> docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS        NAMES
e9a03f77ef4c   myapp_6533801317 "docker-entrypoint.s..." 27 minutes ago Exited (0) 2 minutes ago           dreamy_satoshi
dad1bb5806d5   myapp_6533801317 "docker-entrypoint.s..." 27 minutes ago Created                               optimistic_jennings
9adb70e9e49d   myapp_6533801317 "docker-entrypoint.s..." 28 minutes ago Created                               competent_dirac
0f790fa9527a   thanathornkku/lab8_2 "/bin/sh -c 'echo "e..." About an hour ago Exited (0) About an hour ago           laughing_buck
c1fa3f336b49   lab8_2          "/bin/sh -c 'echo "e..." About an hour ago Exited (0) About an hour ago           xenodochial_euclid
4c7343e0d128   learning_style_project-frontend "docker-entrypoint.s..." 3 months ago   Exited (1) 5 days ago           learning_style_project-frontend-1
9346ebdf76cc   learning_style_project-backend "uvicorn app.main:ap..." 3 months ago   Exited (0) 5 days ago           learning_style_project-backend-1
41f183a78ca8   mongo:4.4       "docker-entrypoint.s..." 3 months ago   Exited (0) 5 days ago           learning_style_project-mongo-1

PS C:\Users\petch\Desktop\Software_Engineering\Lab8\Lab8_4\getting-started\app>
```

**Docker Desktop Interface:**

The Docker Desktop interface shows the "Containers" tab. It displays a list of containers with columns for Name, Container ID, Image, Port(s), CPU (%), and Actions. The containers listed are:

Name	Container ID	Image	Port(s)	CPU (%)	Actions
xenodochial_euclid	c1fa3f336b49	lab8_2-<none>	-	N/A	Stop, Restart, Delete
laughing_buck	0f790fa9527a	thanathornkku/le	-	N/A	Stop, Restart, Delete
competent_dirac	9adb70e9e49d	myapp_6533801	3000:3000	N/A	Stop, Restart, Delete
optimistic_jennings	dad1bb5806d5	myapp_6533801	3000:3000	N/A	Stop, Restart, Delete
dreamy_satoshi	e9a03f77ef4c	myapp_6533801	3000:3000	N/A	Stop, Restart, Delete
learning_style_project	-	-	-	N/A	Stop, Restart, Delete

## Lab Worksheet

PowerShell

```
PS C:\Users\petch\Desktop\Software_Engineering\Lab8\Lab8_4\getting-started\app> docker run -dp 3000:3000 myapp_653380131744540ef6eba36104a10c74f2f6f208f5846952ea26e00e9040238853bf14ea9d
PS C:\Users\petch\Desktop\Software_Engineering\Lab8\Lab8_4\getting-started\app> |
```

docker desktop PERSONAL

Search for Images, containers, volumes, ... **Ctrl+K**

Containers [Give feedback](#)

Container CPU usage ⓘ **0.00% / 1200%** (12 CPUs available)

Container memory usage ⓘ **19.03MB / 7.26GB**

Show charts

Search

Only show running containers

	Name	Container ID	Image	Port(s)	CPU (%)	Actions
<input type="checkbox"/>	xenodochial_euclid	c1fa3f336b49	lab8_2:<none>	-	0%	
<input type="checkbox"/>	laughing_buck	0f790fa9527a	thanathornkku/le	-	0%	
<input type="checkbox"/>	competent_dirac	9adb70e9e49d	myapp_6533801	3000:3000	0%	
<input type="checkbox"/>	optimistic_jennings	dad1bb5806d5	myapp_6533801	3000:3000	0%	
<input type="checkbox"/>	dreamy_satoshi	e9a03f77ef4c	myapp_6533801	3000:3000	0%	
<input type="checkbox"/>	gracious_easley	44540ef6eba3	myapp_6533801	3000:3000	0%	
<input type="checkbox"/>	learning_style_project	-	-	-	0%	

CP353 x [Sec 0] x Lab#8 x Lab8\_1 x Lab8\_2 x Docker x Thana x Facebook x ChatG x Docker x Todo x Todo x

localhost:3000

New Item

Add Item

There is no TODO item. Please add one to the list. By ธนธรณ์ จุฬลายุ

## Lab Worksheet

## แบบฝึกปฏิบัติที่ 8.5: เริ่มต้นสร้าง Pipeline อย่างง่ายสำหรับการ Deploy ด้วย Jenkins

1. เปิด Command line หรือ Terminal บน Docker Desktop

2. บ้อนคำสั่งและทำการรัน container โดยผูกพอร์ต

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

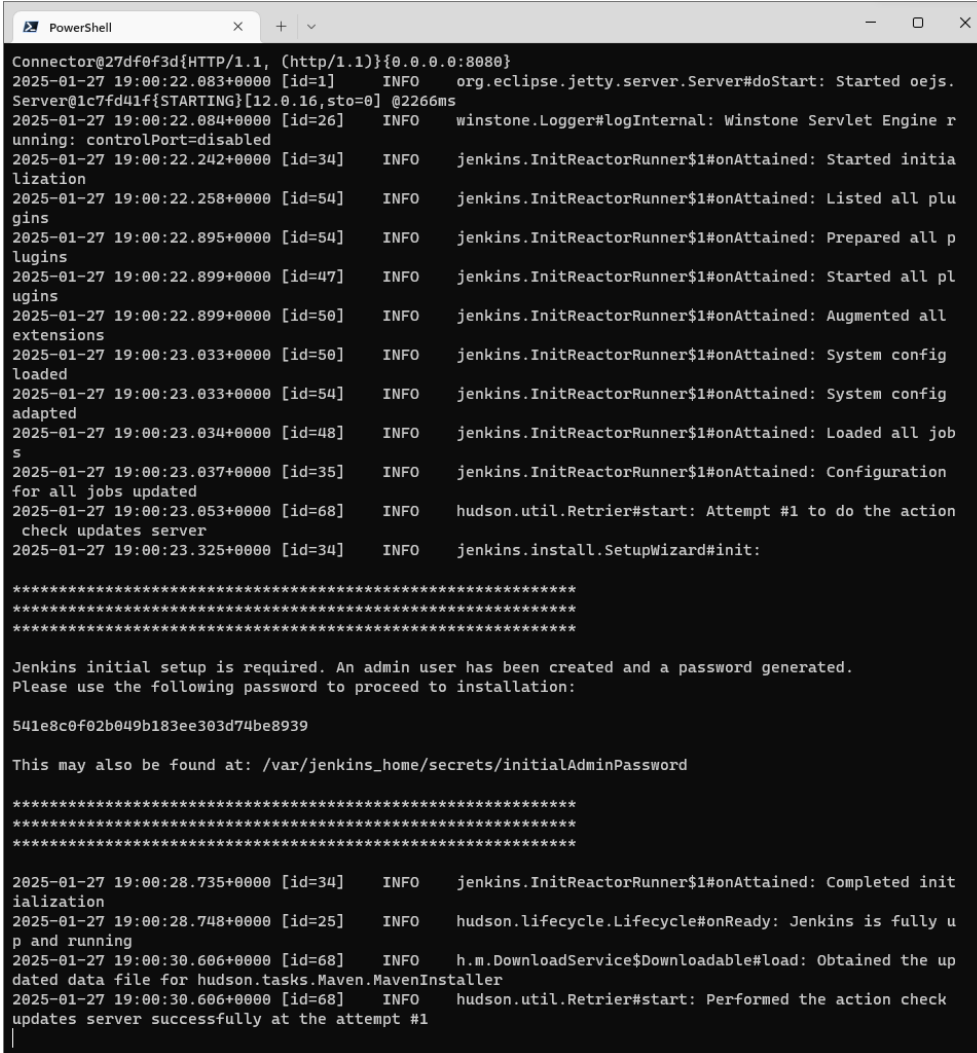
หรือ

```
$ 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



```
PowerShell
Connector@27df0f3d{HTTP/1.1, (http/1.1)}{0.0.0.0:8080}
2025-01-27 19:00:22.083+0000 [id=1] INFO org.eclipse.jetty.server.Server#doStart: Started oejs.
Server@1c7fd41f{STARTING}[12.0.16,sto=0] @2266ms
2025-01-27 19:00:22.084+0000 [id=26] INFO winstone.Logger#logInternal: Winstone Servlet Engine r
unning: controlPort=disabled
2025-01-27 19:00:22.242+0000 [id=34] INFO jenkins.InitReactorRunner$1#onAttained: Started initia
lization
2025-01-27 19:00:22.258+0000 [id=54] INFO jenkins.InitReactorRunner$1#onAttained: Listed all plu
gins
2025-01-27 19:00:22.895+0000 [id=54] INFO jenkins.InitReactorRunner$1#onAttained: Prepared all p
lugins
2025-01-27 19:00:22.899+0000 [id=47] INFO jenkins.InitReactorRunner$1#onAttained: Started all pl
ugins
2025-01-27 19:00:22.899+0000 [id=50] INFO jenkins.InitReactorRunner$1#onAttained: Augmented all
extensions
2025-01-27 19:00:23.033+0000 [id=50] INFO jenkins.InitReactorRunner$1#onAttained: System config
loaded
2025-01-27 19:00:23.033+0000 [id=54] INFO jenkins.InitReactorRunner$1#onAttained: System config
adapted
2025-01-27 19:00:23.034+0000 [id=48] INFO jenkins.InitReactorRunner$1#onAttained: Loaded all job
s
2025-01-27 19:00:23.037+0000 [id=35] INFO jenkins.InitReactorRunner$1#onAttained: Configuration
for all jobs updated
2025-01-27 19:00:23.053+0000 [id=68] INFO hudson.util.Retrier#start: Attempt #1 to do the action
check updates server
2025-01-27 19:00:23.325+0000 [id=34] INFO jenkins.install.SetupWizard#init:

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

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

541e8c0f02b049b183ee303d74be8939

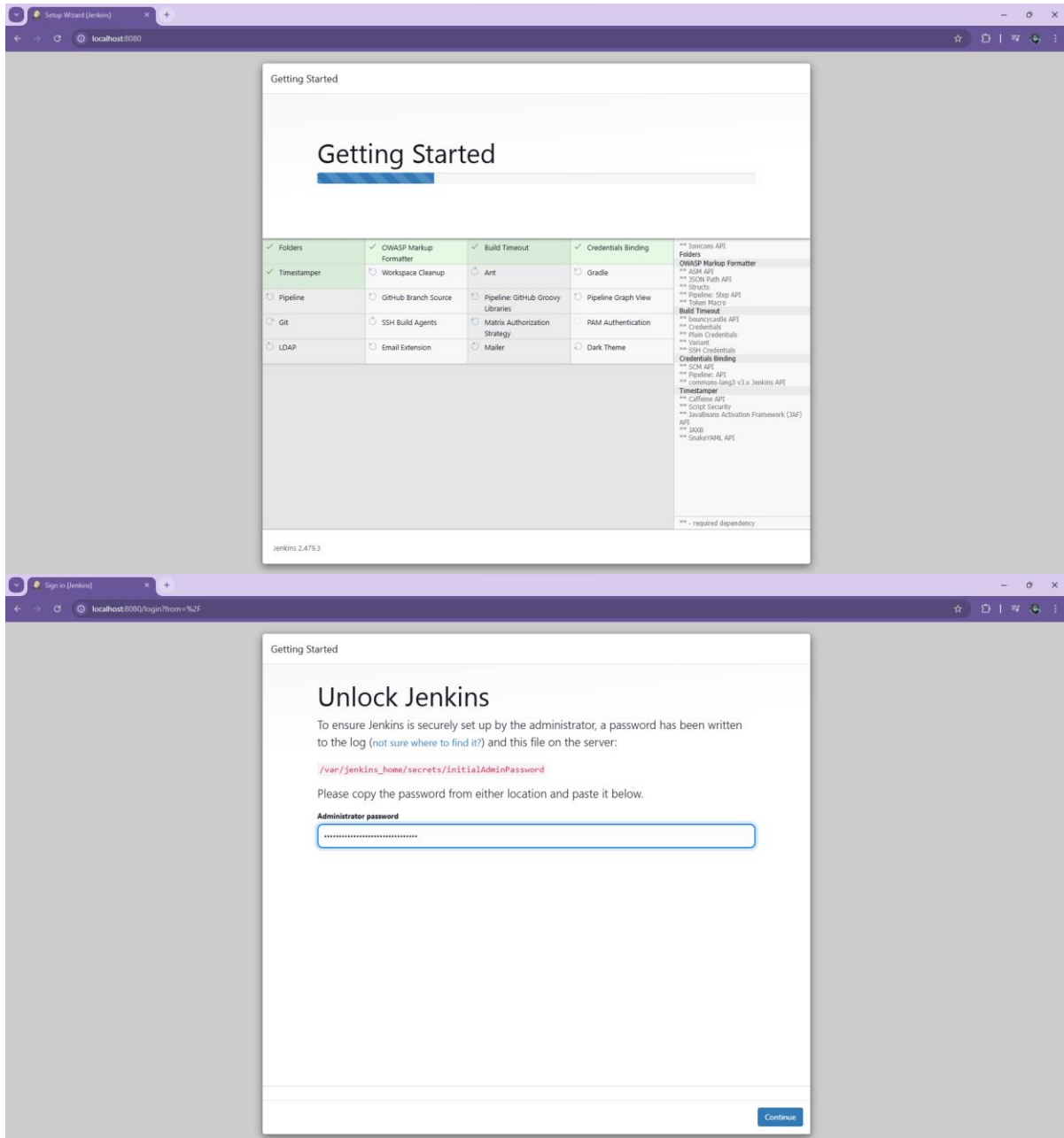
This may also be found at: /var/jenkins_home/secrets/initialAdminPassword

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

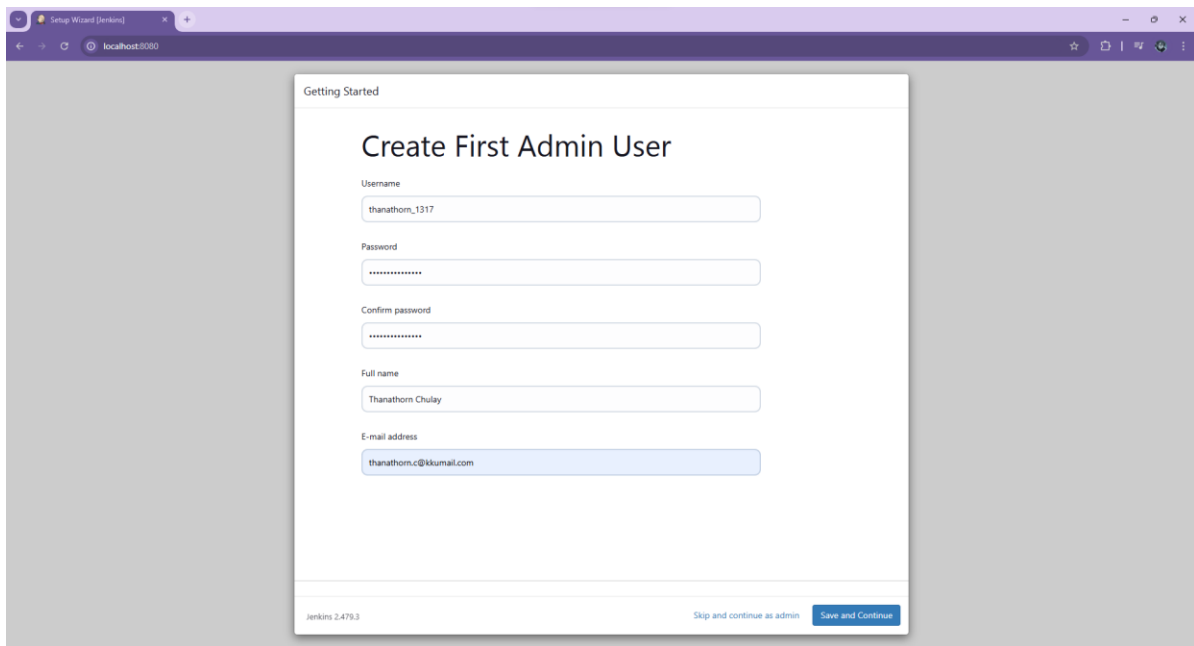
2025-01-27 19:00:28.735+0000 [id=34] INFO jenkins.InitReactorRunner$1#onAttained: Completed init
ialization
2025-01-27 19:00:28.748+0000 [id=25] INFO hudson.lifecycle.Lifecycle#onReady: Jenkins is fully u
p and running
2025-01-27 19:00:30.606+0000 [id=68] INFO h.m.DownloadService$Downloadable#load: Obtained the up
dated data file for hudson.tasks.Maven.MavenInstaller
2025-01-27 19:00:30.606+0000 [id=68] INFO hudson.util.Retrier#start: Performed the action check
updates server successfully at the attempt #1
```

## Lab Worksheet

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

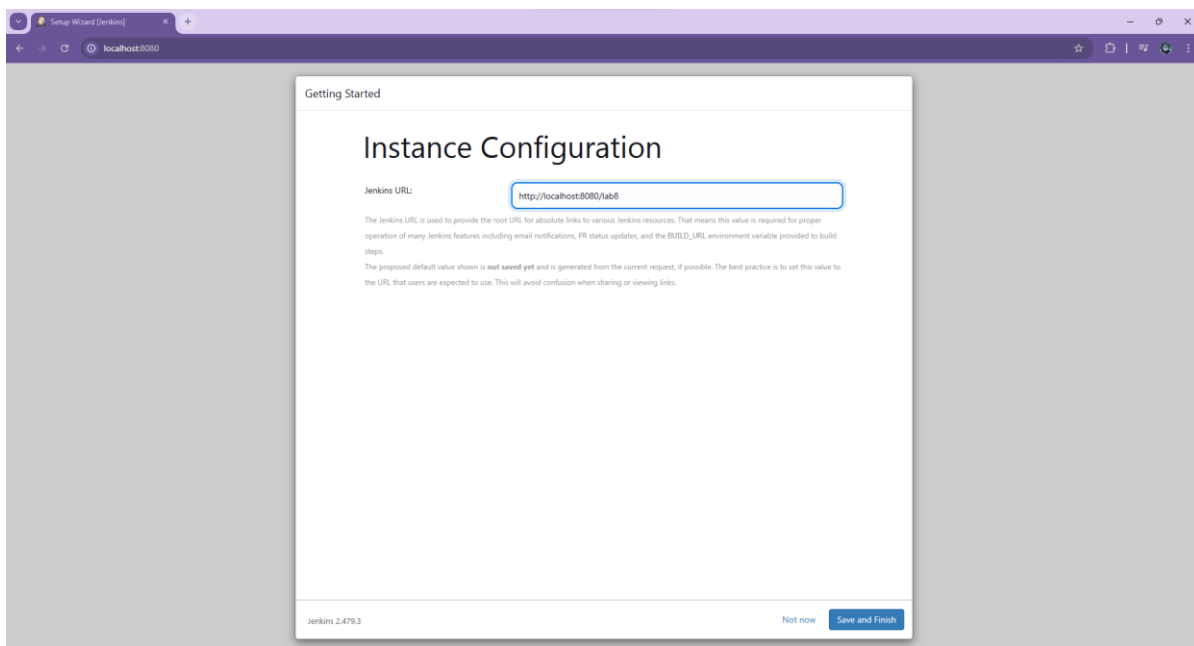


The screenshot shows the 'Getting Started' page of the Jenkins Setup Wizard. The main heading is 'Create First Admin User'. The form contains the following fields:

- Username:
- Password:
- Confirm password:
- Full name:
- E-mail address:

At the bottom, there is a 'Skip and continue as admin' link and a 'Save and Continue' button. The Jenkins version 2.479.3 is displayed in the footer.

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



The screenshot shows the 'Getting Started' page of the Jenkins Setup Wizard. The main heading is 'Instance Configuration'. The form contains the following field:

- Jenkins URL:

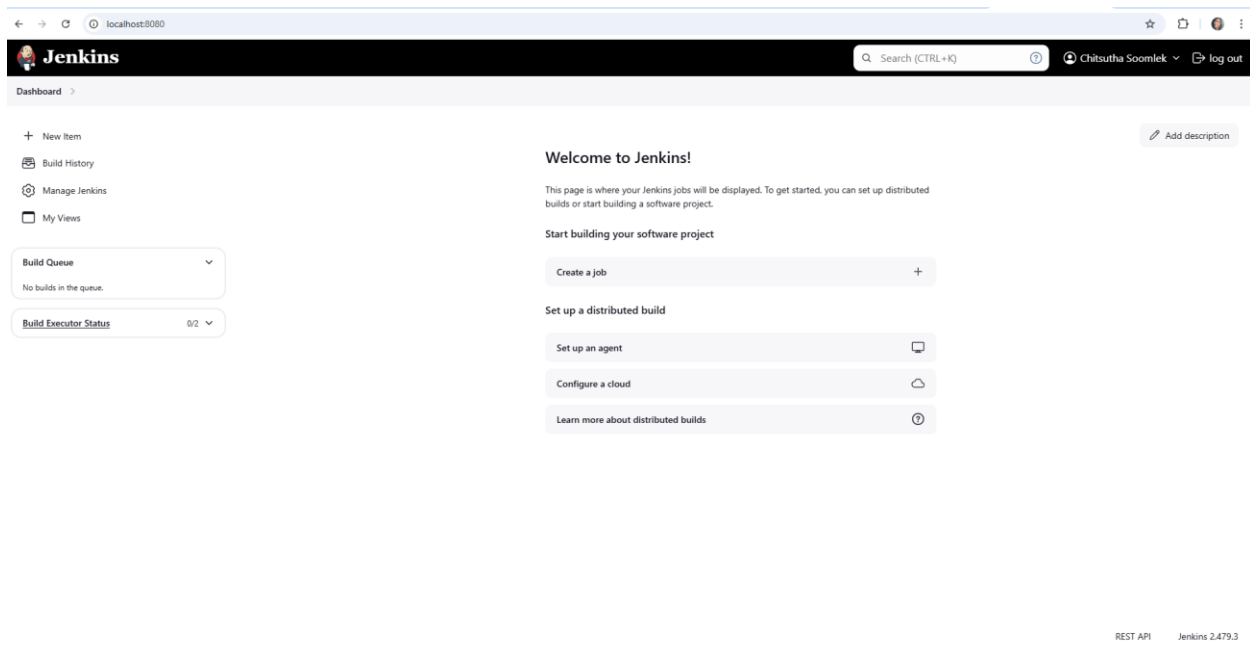
Below the field, there is a note: "The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD\_URL environment variable provided to build steps. The proposed default value shown is not saved yet and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links."

At the bottom, there is a 'Not now' link and a 'Save and Finish' button. The Jenkins version 2.479.3 is displayed in the footer.

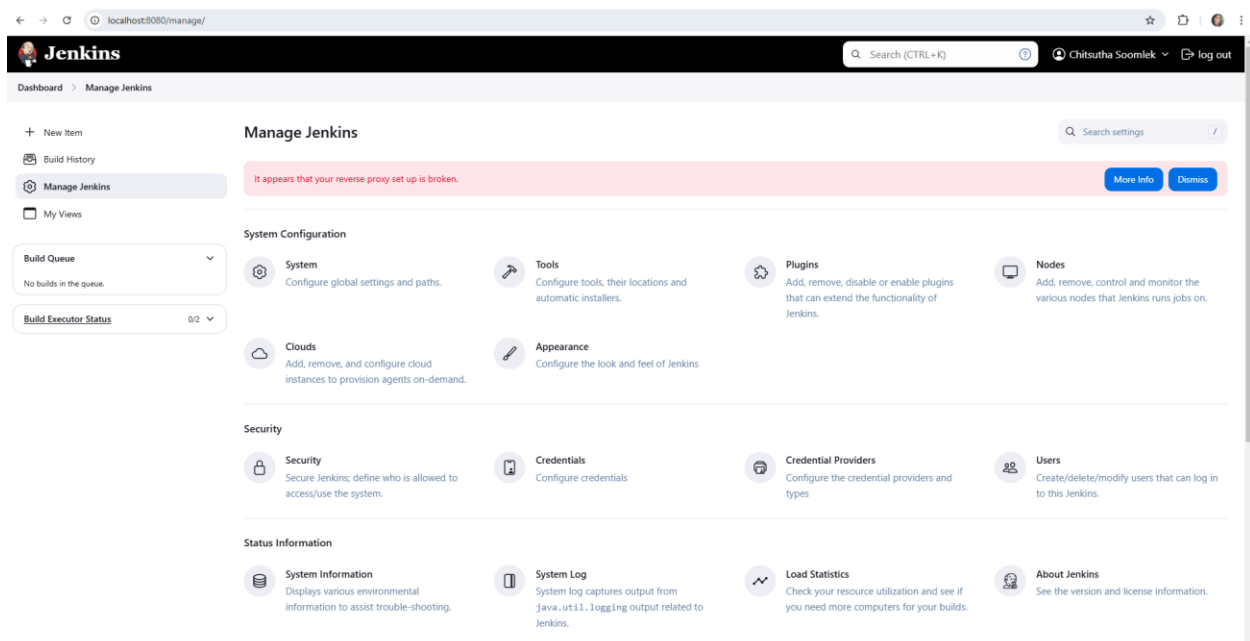


## Lab Worksheet

## 8. เมื่อติดตั้งเรียบร้อยแล้วจะพบกันหน้า Dashboard ดังแสดงในภาพ



## 9. เลือก Manage Jenkins แล้วไปที่เมนู Plugins

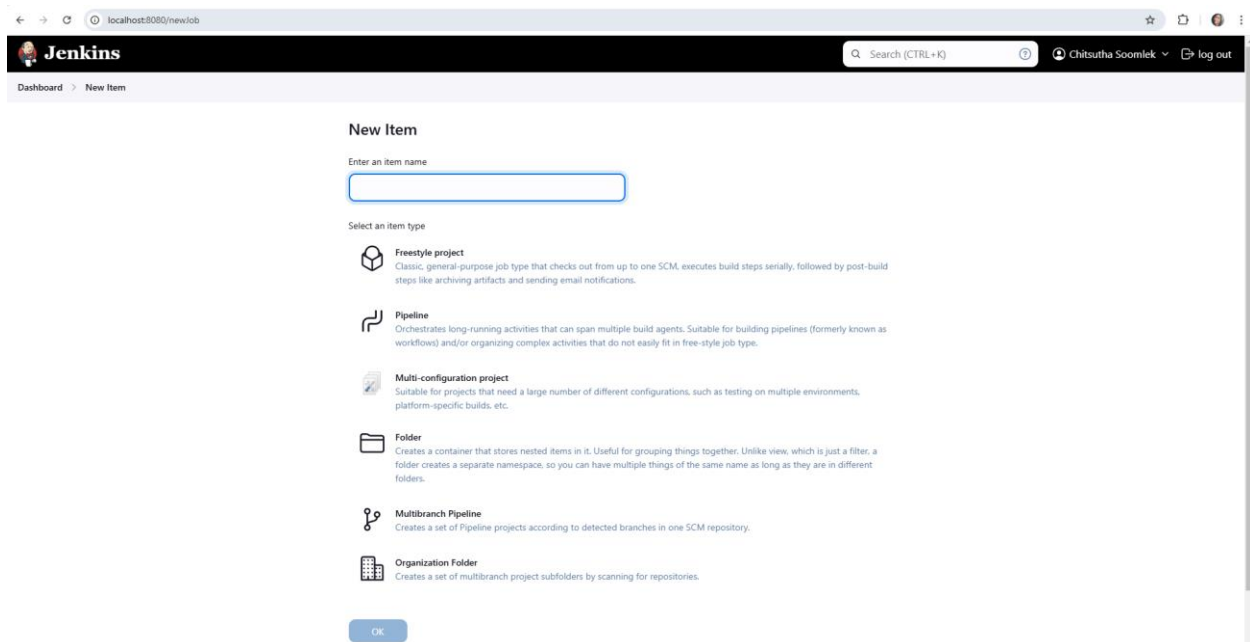


## Lab Worksheet

10. ไปที่เมนู Available plugins แล้วเลือกติดตั้ง Robotframework เพิ่มเติม



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



## Lab Worksheet

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

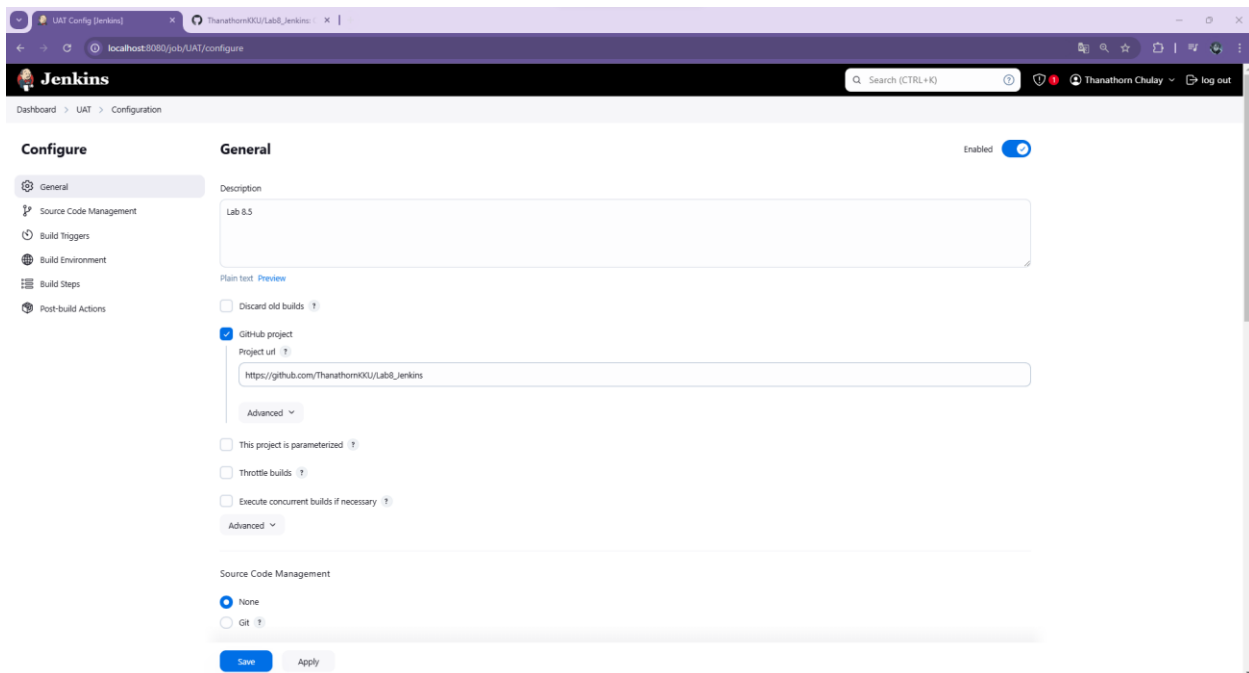
Description: Lab 8.5

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 ไปถึงไฟล์ให้เรียบร้อยแล้ว)

[Check point#14] Capture หน้าจอแสดงการตั้งค่า พร้อมกับตอบคำถามต่อไปนี้



## Lab Worksheet

The image displays two screenshots of the Jenkins configuration page for a job named 'UAT Config'.

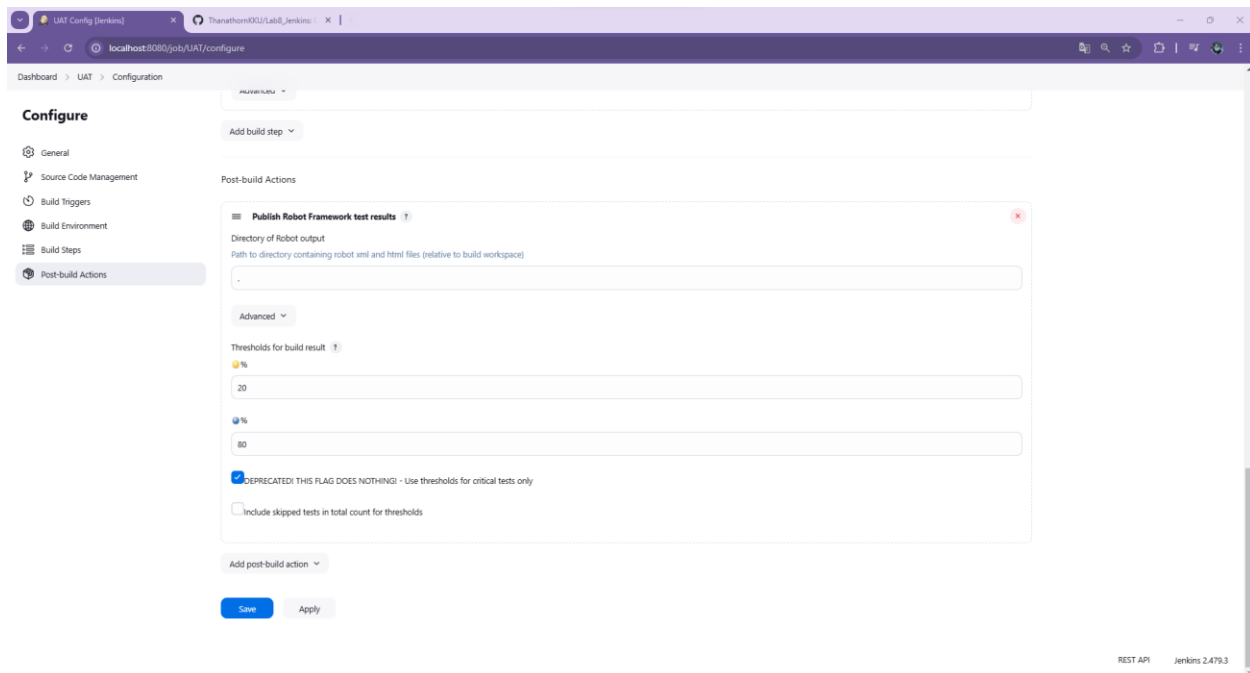
**Top Screenshot:**

- Configuration:** General, Source Code Management, Build Triggers, Build Environment, Build Steps, Post-build Actions.
- Build Triggers:**
  - ☐ Trigger builds remotely (e.g., from scripts)
  - ☐ Build after other projects are built
  - ☒ Build periodically
    - Schedule: `H/15 * * * * *`
    - Would last have run at Monday, January 27, 2025 at 7:20:47 PM Coordinated Universal Time; would next run at Monday, January 27, 2025 at 7:35:47 PM Coordinated Universal Time.
  - ☐ GitHub hook trigger for GITScm polling
  - ☐ Poll SCM
- Build Environment:**
  - ☐ Delete workspace before build starts
  - ☐ Use secret text(s) or file(s)
  - ☐ Add timestamps to the Console Output
  - ☐ Inspect build log for published build scans
  - ☐ Terminate a build if it's stuck
  - ☐ With Ant
- Build Steps:**
  - Execute shell**
    - Command: `See the list of available environment variables`
- Buttons:** Save, Apply

**Bottom Screenshot:**

- Configuration:** General, Source Code Management, Build Triggers, Build Environment, Build Steps, Post-build Actions.
- Build Triggers:** (Same as top screenshot)
- Build Environment:** (Same as top screenshot)
- Build Steps:**
  - Execute shell**
    - Command: `robot: test_001_1_robot`
- Buttons:** Save, Apply
- Post-build Actions:**
  - Add post-build action
- Footer:** REST API, Jenkins 2.479.3

## Lab Worksheet



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

ตอบ `robot test_001_1.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

14. สั่ง Build Now

## Lab Worksheet

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

The screenshot displays the Jenkins web interface. The top navigation bar shows the Jenkins logo, a search bar, and user information (Thanathorn Chulay). The main content area is divided into two sections: 'UAT' and 'Console Output'.

**UAT Section:**

- Status:** UAT (Lab 8.5)
- Latest Robot Results:** No results available yet.
- Permalinks:**
  - Last build (#1), 3 min 30 sec ago
  - Last failed build (#1), 3 min 30 sec ago
  - Last unsuccessful build (#1), 3 min 30 sec ago
  - Last completed build (#1), 3 min 30 sec ago
- Buils:** A list of builds with a filter bar. The first build is #1, 19:42.

**Console Output Section:**

- Status:** Console Output
- Download, Copy, View as plain text:** Buttons for interacting with the console output.
- Log:** A detailed log of the build process, including the user (Thanathorn Chulay), the system (SYSTEM), the workspace (/var/jenkins\_home/workspace/UAT), and the build steps. The log shows a failure due to a missing robot file.

The console output log includes the following text:

```

Started by user Thanathorn Chulay
Running as SYSTEM
Building in workspace /var/jenkins_home/workspace/UAT
[UAT] $ /bin/sh -xe /tmp/jenkins3401415354682665530.sh
+ robot_test_001_1.robot
/tmp/jenkins3401415354682665530.sh: 2: 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 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:52)
    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
  
```