

Lab Worksheet

ชื่อ-นามสกุล _____ ภาควิชา _____ รหัสนักศึกษา _____ 653380278-7 _____ Section _____ 1 _____

Lab#8 – Software Deployment Using Docker

วัตถุประสงค์การเรียนรู้

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

Pre-requisite

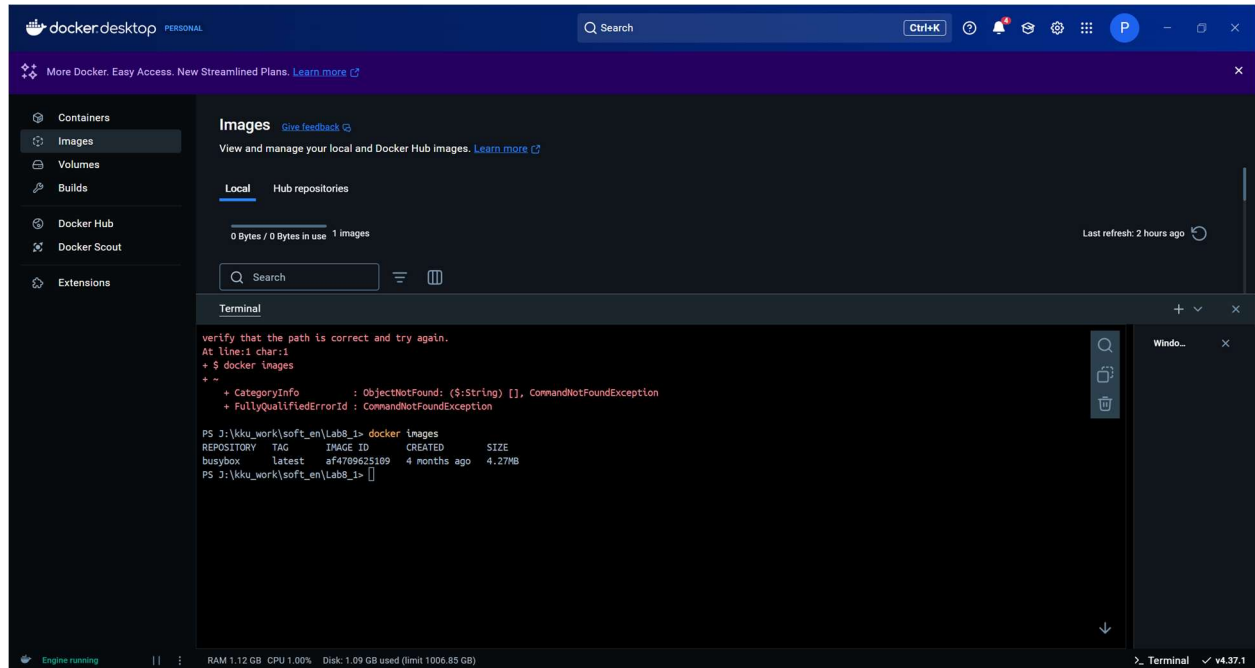
1. ติดตั้ง Docker desktop ลงบนเครื่องคอมพิวเตอร์ โดยดาวน์โหลดจาก <https://www.docker.com/get-started>
2. สร้าง Account บน Docker hub (<https://hub.docker.com/signup>)
3. กำหนดให้ \$ หมายถึง Command prompt และ <> หมายถึง ให้ป้อนค่าของพารามิเตอร์ที่กำหนด

แบบฝึกปฏิบัติที่ 8.1 Hello world - รัน Container จาก Docker image

1. เปิดใช้งาน Docker desktop และ Login ด้วย Username และ Password ที่ลงทะเบียนกับ Docker Hub เอาไว้
1. เปิด Command line หรือ Terminal บน Docker Desktop จากนั้นสร้าง Directory ชื่อ Lab8_1
2. ย้ายตำแหน่งปัจจุบันไปที่ Lab8_1 เพื่อใช้เป็น Working directory
3. ป้อนคำสั่ง \$ docker pull busybox หรือ \$ sudo docker pull busybox สำหรับกรณีที่ติดปัญหา Permission denied
(หมายเหตุ: BusyBox เป็น software suite ที่รองรับคำสั่งบางอย่างบน Unix - <https://busybox.net>)
4. ป้อนคำสั่ง \$ docker images

Lab Worksheet

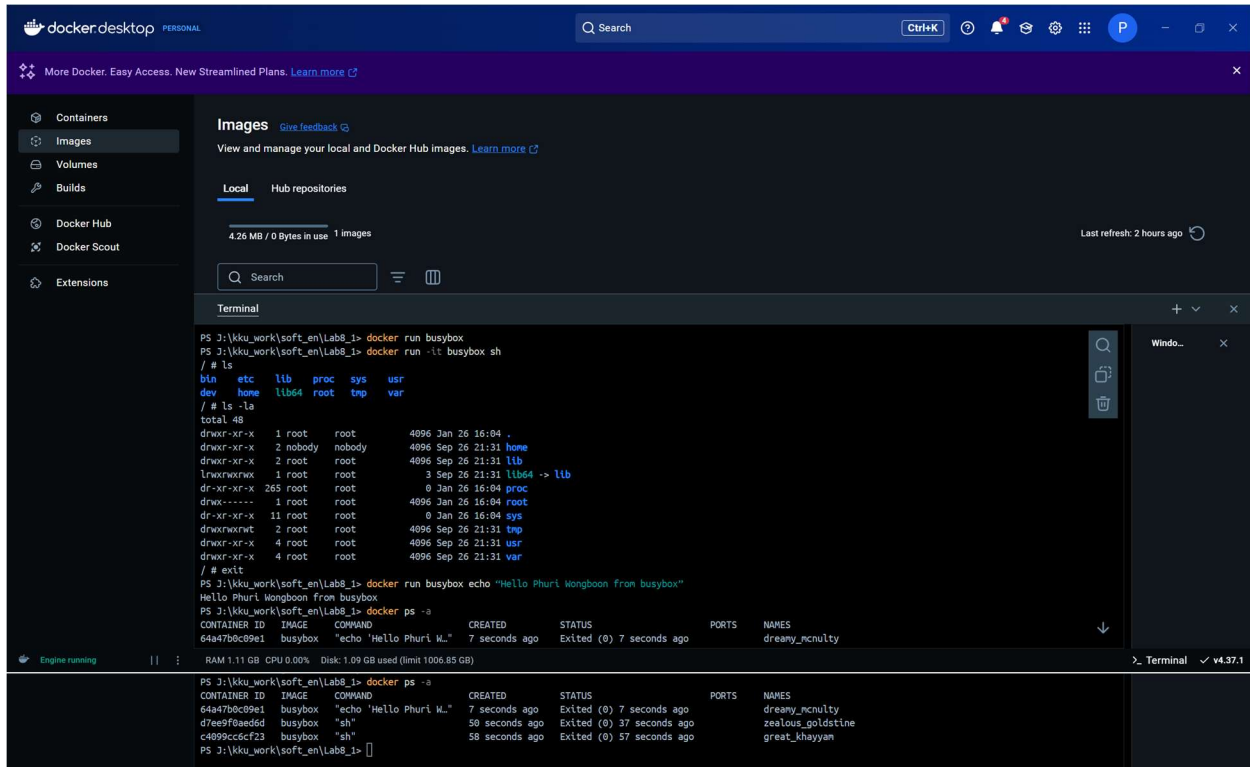
[Check point#1] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงผลลัพธ์ที่ได้ พร้อมคำตอบคำถามต่อไปนี้



- (1) สิ่งที่อยู่ภายใต้คอลัมน์ Repository คืออะไร___ busybox ___
- (2) Tag ที่ใช้บ่งบอกถึงอะไร___ latest ___
5. ป้อนคำสั่ง \$ docker run busybox
6. ป้อนคำสั่ง \$ docker run -it busybox sh
7. ป้อนคำสั่ง ls
8. ป้อนคำสั่ง ls -la
9. ป้อนคำสั่ง exit
10. ป้อนคำสั่ง \$ docker run busybox echo "Hello ชื่อและนามสกุลของนักศึกษา from busybox"
11. ป้อนคำสั่ง \$ docker ps -a

Lab Worksheet

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



(1) เมื่อใช้ option -it ในคำสั่ง run ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป
 ___ การให้ -it ใน docker run -it busybox sh จะทำให้สามารถพิมพ์คำสั่งลงไปเองได้หลายๆ คำสั่ง

(2) คอลัมน์ STATUS จากการรันคำสั่ง docker ps -a แสดงถึงข้อมูลอะไร
 ___ หมายถึงสถานะของคำสั่ง ว่าคำสั่งนั้นๆ ทำงานเสร็จแล้วไปกี่นาทีแล้ว

12. ป้อนคำสั่ง \$ docker rm <container ID ที่ต้องการลบ>
 ___ docker rm c4099cc6cf23d24d208bcfcba7e705ccf67b3da85494059fa368b4bb510018cd

Lab Worksheet

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

The screenshot displays the Docker Desktop interface with the 'Containers' tab selected. The interface shows a list of running containers and a terminal window with the following output:

```

dnx-r-xr-x 4 root root 4096 Sep 26 21:31 usr
dnx-r-xr-x 4 root root 4096 Sep 26 21:31 var
/ # exit
PS J:\kku_work\soft_en\Lab8_1> docker run busybox echo "Hello Phuri Wongboon from busybox"
Hello Phuri Wongboon from busybox
PS J:\kku_work\soft_en\Lab8_1> docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED    STATUS    PORTS    NAMES
64a47b0c09e1   busybox   "echo 'Hello Phuri W..." 7 seconds ago    Exited (0) 7 seconds ago    dreamy_mcnulty
d7ee9f0aed6d   busybox   "sh"                    50 seconds ago    Exited (0) 37 seconds ago    zealous_goldstine
c4099cc6cf23   busybox   "sh"                    58 seconds ago    Exited (0) 57 seconds ago    great_khayyam
PS J:\kku_work\soft_en\Lab8_1> docker rm c4099cc6cf23d242d208bcbfcb7e705ccf67b3da85494059fa368b4bb510018cd
PS J:\kku_work\soft_en\Lab8_1>
  
```

The terminal output shows the execution of a Docker command to run a container, followed by a list of containers and the removal of a specific container. The interface also displays system resources like RAM and CPU usage.

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

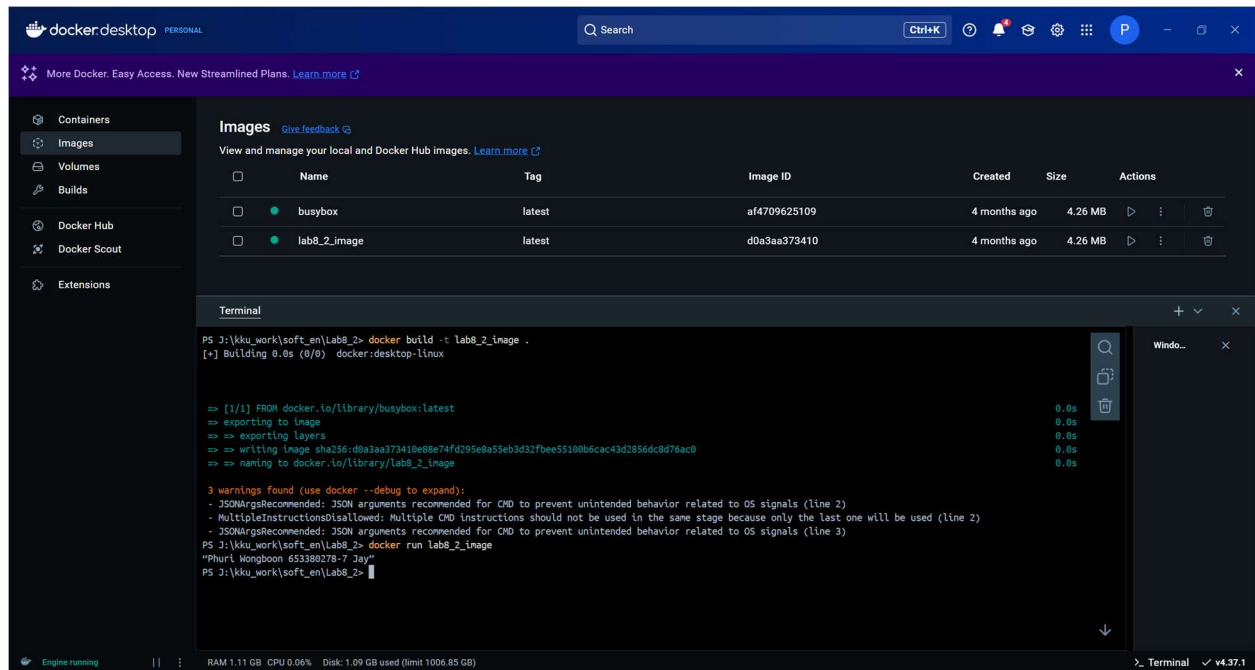
```
$ docker build -t <ชื่อ Image> .
```

```
docker build -t lab8_2_image .
```

6. เมื่อ Build สำเร็จแล้ว ให้ทำการรัน Docker image ที่สร้างขึ้นในขั้นตอนที่ 5

Lab Worksheet

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



(1) คำสั่งที่ใช้ในการ run คือ

___ ใช้คำสั่ง docker run lab8_2_image

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

___ ตัว -t จะทำหน้าที่ย้ายชื่อให้ image เพื่อความง่ายในการใช้งาน

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
```

```
docker build -t phuriwon/lab8 .
```

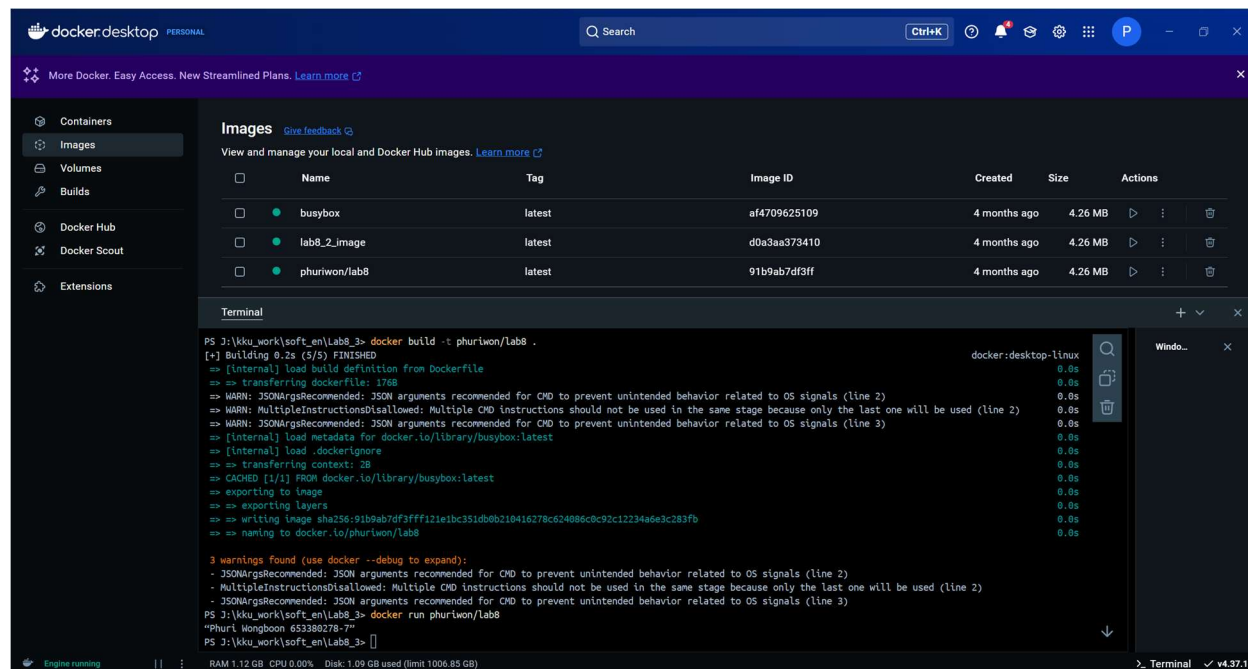
5. ทำการรัน Docker image บน Container ในเครื่องของตัวเองเพื่อทดสอบผลลัพธ์ ด้วยคำสั่ง

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

```
docker run phuriwon/lab8
```

Lab Worksheet

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



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 two windows. The top window is Docker Desktop, and the bottom window is the Docker Hub website.

Docker Desktop Window:

- Images Table:**

Name	Tag	Image ID	Created	Size	Actions
busybox	latest	af4709625109	4 months ago	4.26 MB	[Play] [More] [Delete]
lab8_2_image	latest	d0a3aa373410	4 months ago	4.26 MB	[Play] [More] [Delete]
phuriwon/lab8	latest	91b9ab7d3ff	4 months ago	4.26 MB	[Play] [More] [Delete]
- Terminal Output:**

```

=> WARN: MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the same stage because only the last one will be used (line 2)
=> 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: 28
=> CACHED [1/1] FROM docker.io/library/busybox:latest
=> exporting to image
=> exporting layers
=> writing image sha256:91b9ab7d3ff121e1bc351db0b210416278c624086c0c92c12234a6e3c283fb
=> naming to docker.io/phuriwon/lab8

3 warnings found (use docker --debug to expand):
- JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 2)
- MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the same stage because only the last one will be used (line 2)
- JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 3)
PS J:\kku_work\soft_en\Lab8_3> docker run phuriwon/lab8
"Phuri Wongboon 653380278-7"
PS J:\kku_work\soft_en\Lab8_3> docker push phuriwon/lab8
Using default tag: latest
The push refers to repository [docker.io/phuriwon/lab8]
59654b79daad: Mounted from library/busybox
latest: digest: sha256:0512126880a4a743c44adf733cf96d94af9a80b51bccc942ad3a8cfed91091f size: 527
PS J:\kku_work\soft_en\Lab8_3>

```

Docker Hub Window:

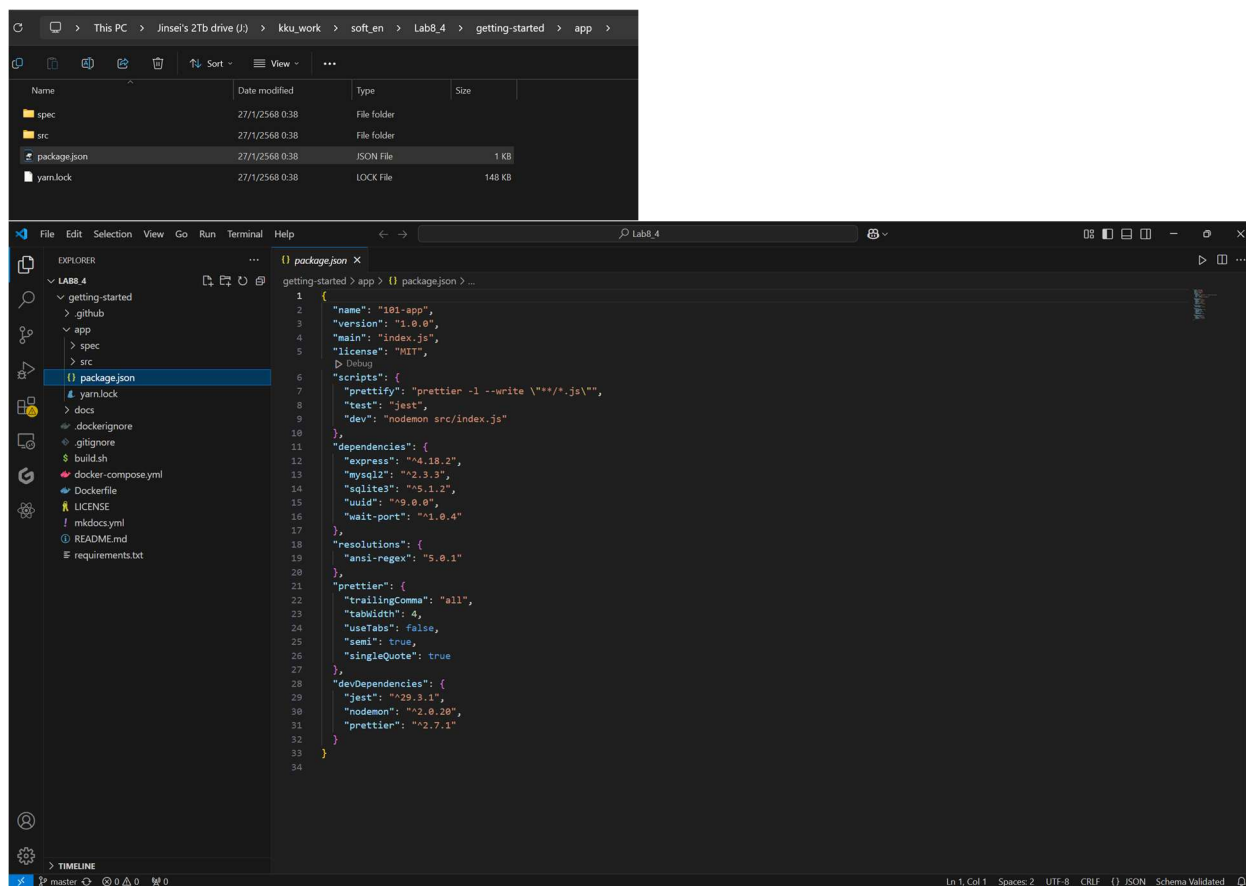
- User:** Phuri Wongboon (Community User)
- Repositories:**
 - phuriwon/lab8
 - By phuriwon · Updated a minute ago

Lab Worksheet

แบบฝึกปฏิบัติที่ 8.4: การ Build แอปพลิเคชันจาก Container image และการ Update แอปพลิเคชัน

1. เปิด Command line หรือ Terminal จากนั้นสร้าง Directory ชื่อ Lab8_4
2. ทำการ Clone ซอร์สโค้ดของเว็บแอปพลิเคชันจาก GitHub repository
<https://github.com/docker/getting-started.git> ลงใน Directory ที่สร้างขึ้น โดยใช้คำสั่ง
`$ git clone https://github.com/docker/getting-started.git`
3. เปิดดูองค์ประกอบภายใน getting-started/app เมื่อพบไฟล์ package.json ให้ใช้ Text editor ในการเปิดอ่าน

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



Lab Worksheet

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 โดยกำหนดให้ชื่อ image เป็น myapp_รหัสสนศ. ไม่มีขีด

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

docker build -t myapp_6533802787 .

[Check point#8] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง)

แสดงคำสั่งและผลลัพธ์ที่ได้ทางหน้าจอ

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

Image Name	Tag	Size	Created
lab8_2_image	latest	4.26 MB	4 months ago
phuriwon/lab8	latest	4.26 MB	4 months ago
myapp_6533802787	latest	216.9 MB	4 minutes ago

Below the images list, a terminal window is open, showing the command 'docker build -t myapp_6533802787 .' being executed. The output shows the build process, including pulling the node:18-alpine image and copying files to the container.

```
PS J:\kku_work\soft_en\Lab8_4> cd J:\kku_work\soft_en\Lab8_4\getting-started\app
PS J:\kku_work\soft_en\Lab8_4\getting-started\app> docker build -t myapp_6533802787 .
Building [42s] (20/20) 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@sha256:974af6dc8b314dc6502b14243b8a39fb2d94d975e9059d6d66be3e274fbb25
=> resolve docker.io/library/node:18-alpine@sha256:974af6dc8b314dc6502b14243b8a39fb2d94d975e9059d6d66be3e274fbb25
=> sha256:dc3b7b337595e6f4d214e4eed84f230ee4e4c03a50388d573e289b9e5e40 6.18kB / 6.18kB
=> sha256:1f3e46996e2966e4fa5846e56e76e3748b7315e2ded61476c24403d592134f0 3.64MB / 3.64MB
=> sha256:37892ffbca871a10f813803949d18c3015a482051d51b7e0da02525e63167c 40.01MB / 40.01MB
=> sha256:5650d6de56f0bb419872b076ac1df28f577b39573c3b72fb0d1507436091bc1 1.26MB / 1.26MB
=> sha256:974af6dc8b314dc6502b14243b8a39fb2d94d975e9059d6d66be3e274fbb25 7.67kB / 7.67kB
=> sha256:6e804119c3894fc3782793bf0d2adc89201c3185acc0647b17a7cc0bb385e 1.72kB / 1.72kB
=> sha256:6504e29608cd5213b52cda800370abb3d12639802d06b46b6fca368990ca771 444B / 444B
=> extracting sha256:1f3e46996e2966e4fa5846e56e76e3748b7315e2ded61476c24403d592134f0
=> extracting sha256:37892ffbca871a10f813803949d18c3015a482051d51b7e0da02525e63167c
=> extracting sha256:5650d6de56f0bb419872b076ac1df28f577b39573c3b72fb0d1507436091bc1
=> extracting sha256:6504e29608cd5213b52cda800370abb3d12639802d06b46b6fca368990ca771
=> [internal] load build context
=> transferring context: 4.62kB
=> [2/4] WORKDIR /app
=> [3/4] COPY . .
=> [4/4] RUN yarn install --production
=> exporting to image
=> exporting layers
=> writing image sha256:4f688a92c588f57a77598fca341ec2bc4ca2fde07b096fa18c577ae0478fdb
=> naming to docker.io/library/myapp_6533802787
PS J:\kku_work\soft_en\Lab8_4\getting-started\app>
```

Lab Worksheet

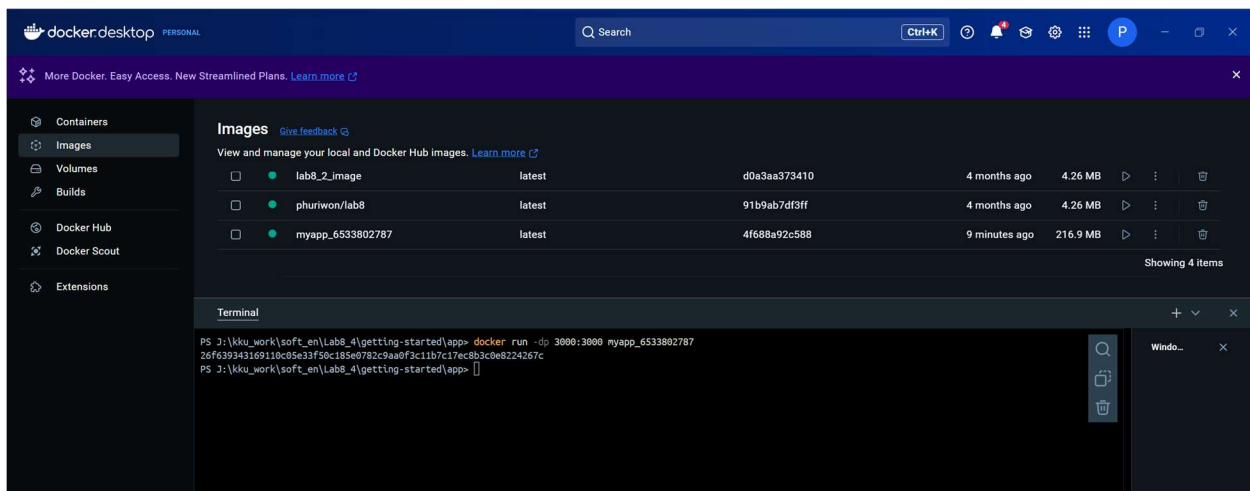
6. ทำการ Start ตัว Container ของแอปพลิเคชันที่สร้างขึ้น โดยใช้คำสั่ง

```
$ docker run -dp 3000:3000 <myapp_รหัสสนศ. ไม่มีขีด>
```

```
docker run -dp 3000:3000 myapp_6533802787
```

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

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



หมายเหตุ: นศ.สามารถทดลองเล่น Web application ที่ทำงานอยู่ได้

Lab Worksheet

8. ทำการแก้ไข Source code ของ Web application ดังนี้

a. เปิดไฟล์ src/static/js/app.js ด้วย Editor และแก้ไขบรรทัดที่ 56 จาก

`<p className="text-center">No items yet! Add one above!</p>` เป็น

`<p className="text-center">There is no TODO item. Please add one to the list. By ชื่อและนามสกุลของนักศึกษา</p>`

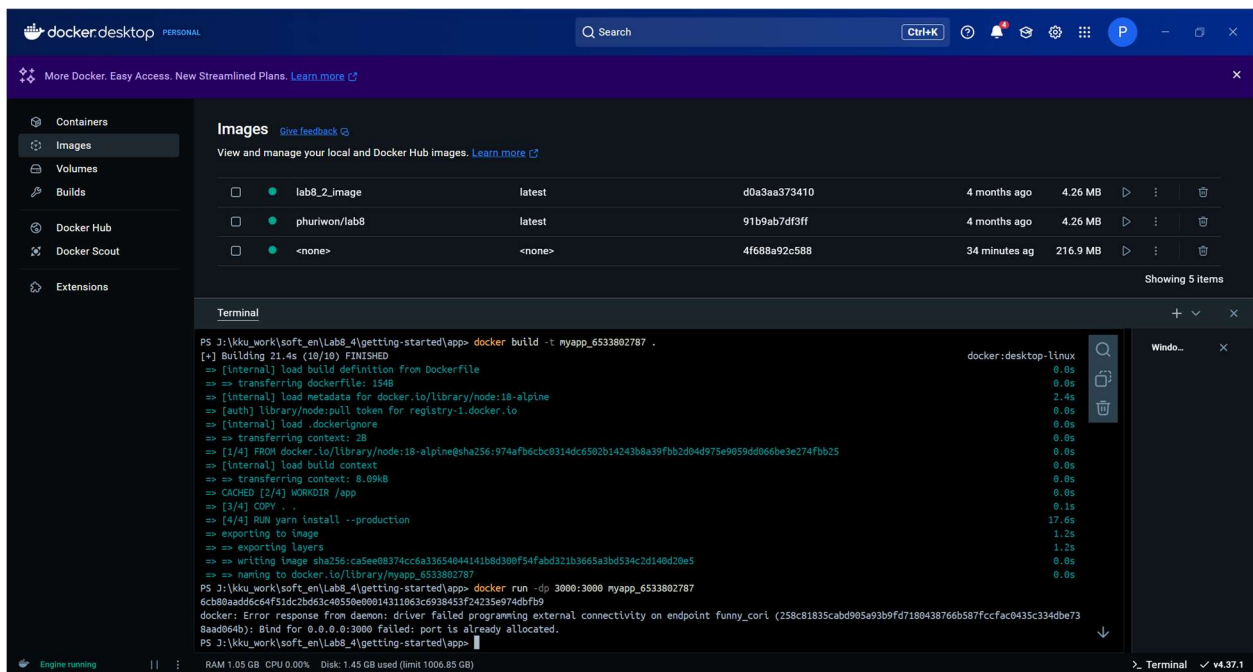
b. Save ไฟล์ให้เรียบร้อย

9. ทำการ Build Docker image โดยใช้คำสั่งเดียวกันกับข้อ 5

10. Start และรัน Container ตัวใหม่ โดยใช้คำสั่งเดียวกันกับข้อ 6

[Check point#10] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง)

แสดงคำสั่งและผลลัพธ์ที่ได้ทางหน้าจอ พร้อมกับตอบคำถามต่อไปนี้



Lab Worksheet

(1) Error ที่เกิดขึ้นหมายความว่าอย่างไร และเกิดขึ้นเพราะอะไร

___ Error หมายถึง port 3000 กำลังถูกใช้งานอยู่

11. ลบ Container ของ Web application เวอร์ชันก่อนแก้ไขออกจากระบบ โดยใช้วิธีใดวิธีหนึ่งดังต่อไปนี้

a. ผ่าน Command line interface

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

b. ผ่าน Docker desktop

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

12. Start และรัน Container ตัวใหม่อีกครั้ง โดยใช้คำสั่งเดียวกันกับข้อ 6

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

Lab Worksheet

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

The screenshot shows the Docker Desktop application window. The top bar is dark blue with the Docker logo and 'docker.desktop PERSONAL'. Below the bar is a purple banner with the text 'More Docker. Easy Access. New Streamlined Plans. Learn more'. The left sidebar contains navigation icons for Containers, Images, Volumes, Builds, Docker Hub, Docker Scout, and Extensions. The main area is titled 'Containers' and shows a table of running containers:

Container ID	Name	Image	Status	Uptime	Actions
64a47b0c09e1	dreamy_mcnulty	busybox	Running	2 hours ago	[Stop] [Refresh] [Delete]
9b96909540d5	keen_cannon	lab8_2_image	Running	2 hours ago	[Stop] [Refresh] [Delete]
41a36e811328	elated_edison	phuriwon/lab8	Running	1 hour ago	[Stop] [Refresh] [Delete]

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

```
PS J:\kku_work\soft_en\Lab8_4\getting-started\app> docker rm 26f639343169110c05e33f50c185e0782c9aa0f3c11b7c17ec8b3c0e8224267c
26f639343169110c05e33f50c185e0782c9aa0f3c11b7c17ec8b3c0e8224267c
PS J:\kku_work\soft_en\Lab8_4\getting-started\app> docker build -t myapp_6533802787 .
[+] Building 2.2s (18/18) 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@sha256:974af6c8c9314dc582b14243b8a39fbb2d04d975e9059dd066be3e274fbb25
=> [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:ca5ee08374cc6a33654044141b8d300f54fabd321b3665a3bd534c2d140d2be5
=> naming to docker.io/library/myapp_6533802787
PS J:\kku_work\soft_en\Lab8_4\getting-started\app> docker run -dp 3000:3000 myapp_6533802787
7fd8babad2d37c1cda0211bd77a3a0ef6d5084ahf87c0ca20c19da370cf140f
PS J:\kku_work\soft_en\Lab8_4\getting-started\app>
```

The bottom status bar shows 'Engine running', 'RAM 1.23 GB', 'CPU 0.06%', and 'Disk 1.45 GB used (limit 1006.85 GB)'. The bottom right corner shows 'Terminal' and 'v4.37.1'.

The screenshot shows a web application interface. At the top, there is a 'New Item' button and an 'Add Item' button. Below these buttons, a message reads: 'There is no TODO item. Please add one to the list. By Phuri Wongboon'. The background is a light gray color.

Lab Worksheet

แบบฝึกปฏิบัติที่ 8.5: เริ่มต้นสร้าง Pipeline อย่างง่ายสำหรับการ Deploy ด้วย Jenkins

1. เปิด Command line หรือ Terminal บน Docker Desktop

2. ป้อนคำสั่งและทำการรัน container โดยผูกพอร์ต

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

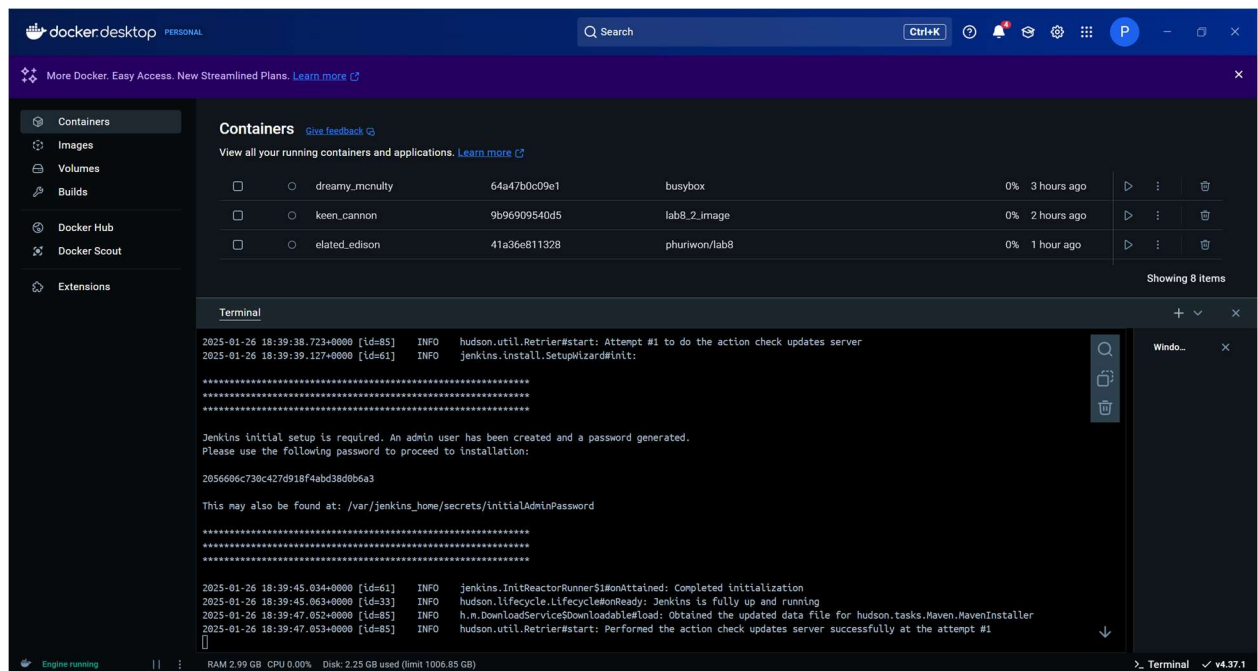
หรือ

```
$ 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

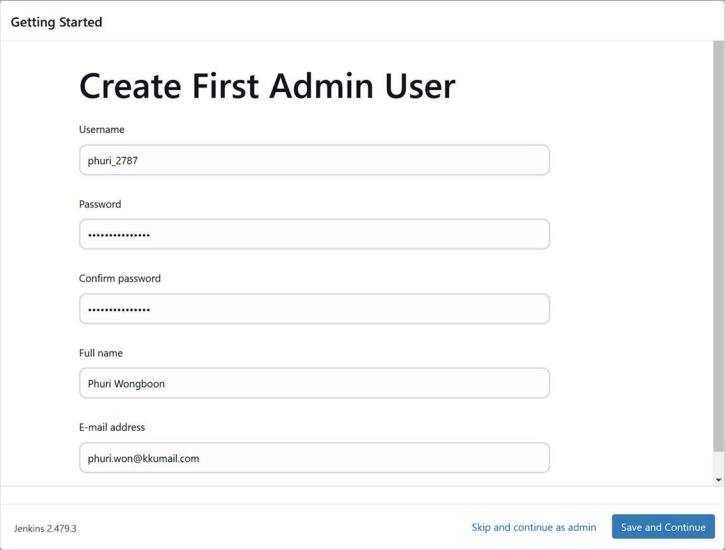
[Check point#12] Capture หน้าจอที่แสดงผล Admin password



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



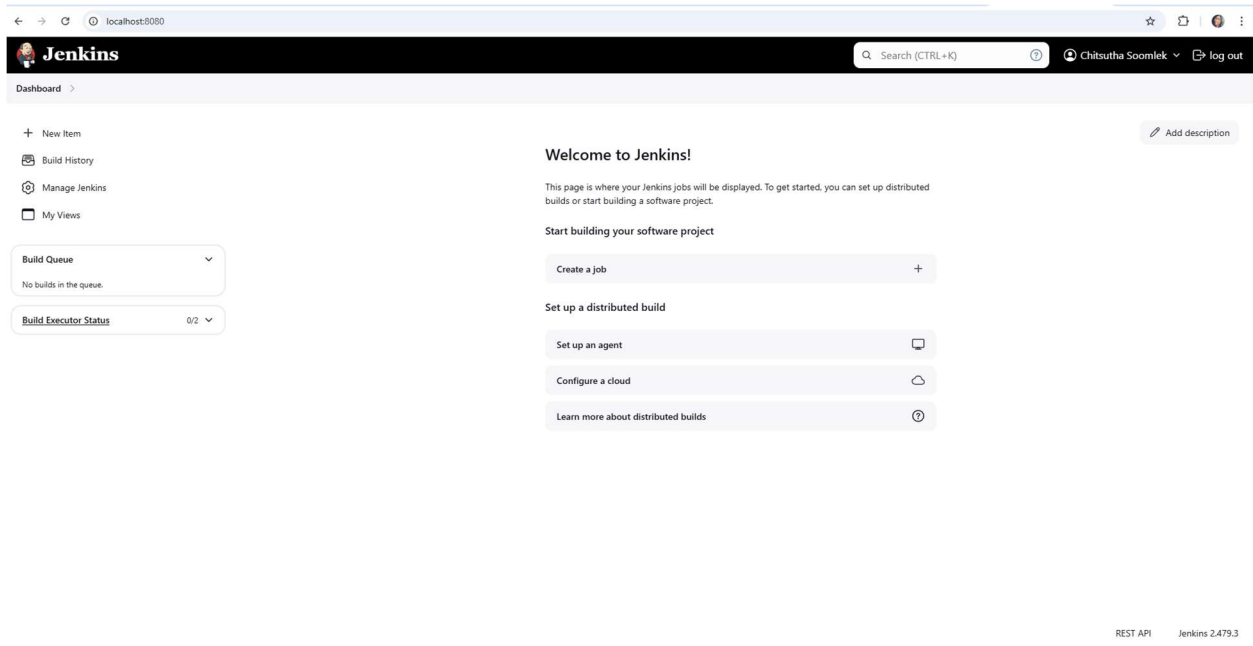
The screenshot shows the 'Getting Started' page of Jenkins 2.479.3. The main heading is 'Create First Admin User'. The form contains the following fields:

- Username:** phuri_2787
- Password:** (masked with dots)
- Confirm password:** (masked with dots)
- Full name:** Phuri Wongboon
- E-mail address:** phuri.won@kkumail.com

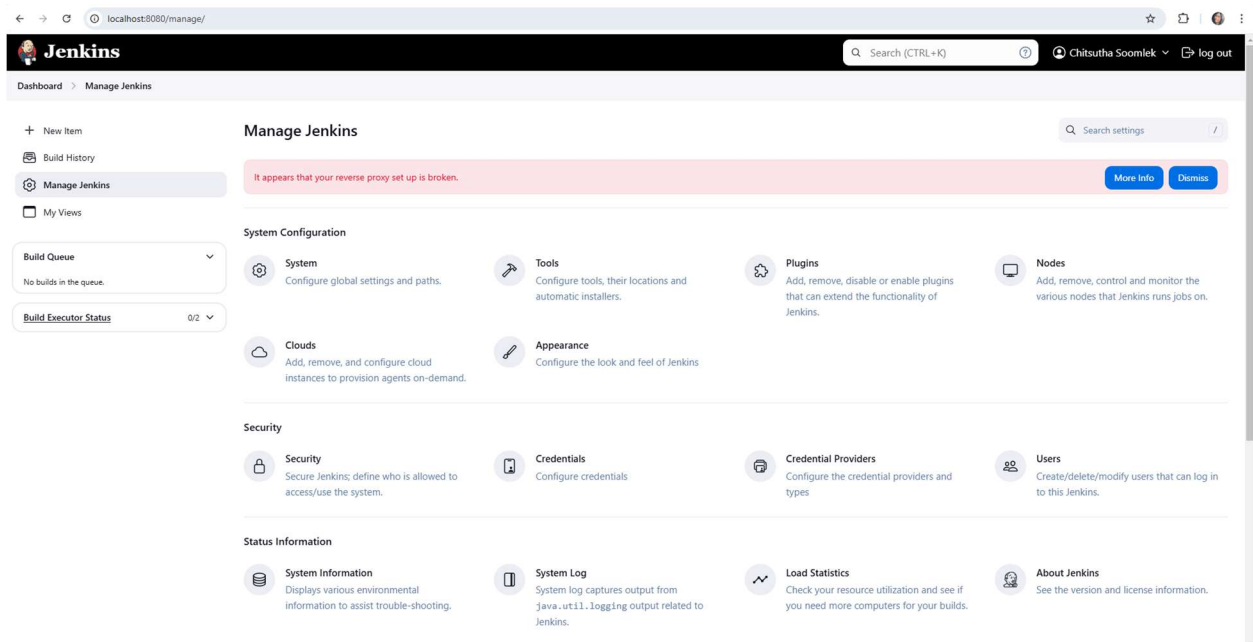
At the bottom, there are two buttons: 'Skip and continue as admin' and 'Save and Continue'.

Lab Worksheet

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



9. เลือก Manage Jenkins แล้วไปที่เมนู Plugins

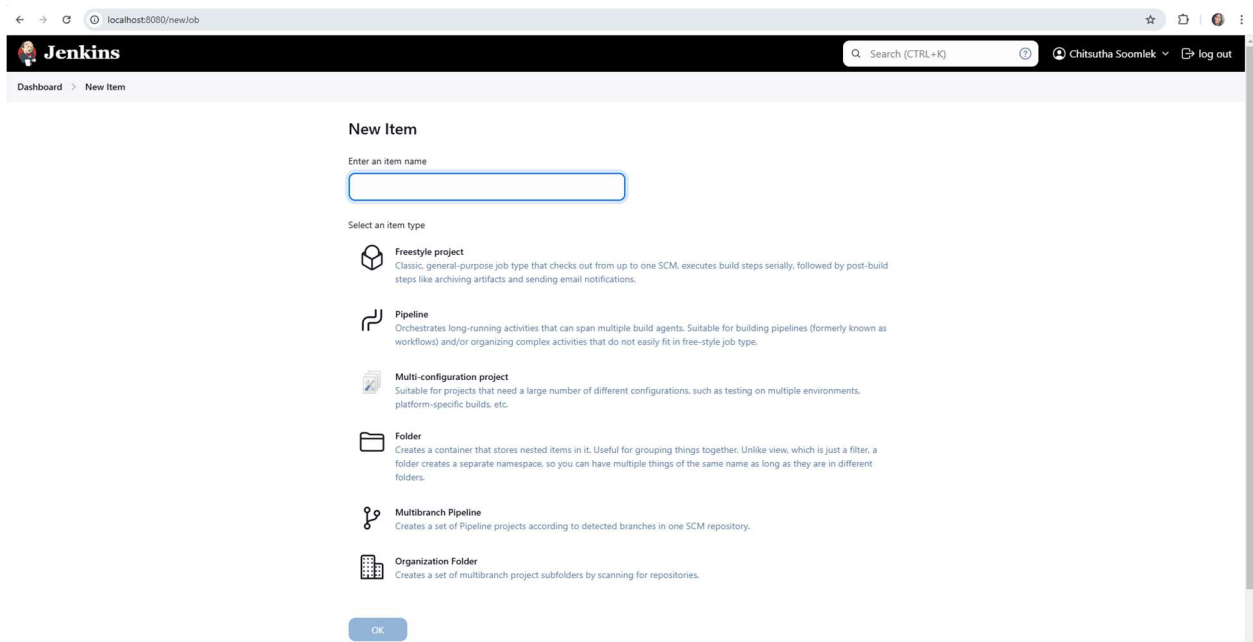


Lab Worksheet

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)

Lab Worksheet

[Check point#14] Capture หน้าจอแสดงการตั้งค่า พร้อมกับตอบคำถามต่อไปนี้

The screenshot shows the Jenkins Configuration page for a job named "Lab 8.5". The page is divided into three main sections: General, Build Triggers, and Build Steps.

- General:** The job is enabled. The description is "Lab 8.5". The "GitHub project" checkbox is checked, and the project URL is "https://github.com/Phuri-won/653380278-7-Lab7_TestScript.git".
- Build Triggers:** The "Build periodically" checkbox is checked. The schedule is "H/15 * * * *". The text below the schedule indicates the next run will be on Sunday, January 26, 2025, at 7:20:08 PM Coordinated Universal Time.
- Build Steps:** There is one build step named "Execute shell". The command is "robot WebDemo-master/lab7_2.robot".

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

___ robot WebDemo-master/lab7_2.robot

Lab Worksheet

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

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

