

Lab Worksheet

ชื่อ-นามสกุล สร้อยฟ้า รักนุช รหัสนักศึกษา 653380347-4 Section 4

Lab#8 – Software Deployment Using Docker

วัตถุประสงค์การเรียนรู้

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

Pre-requisite

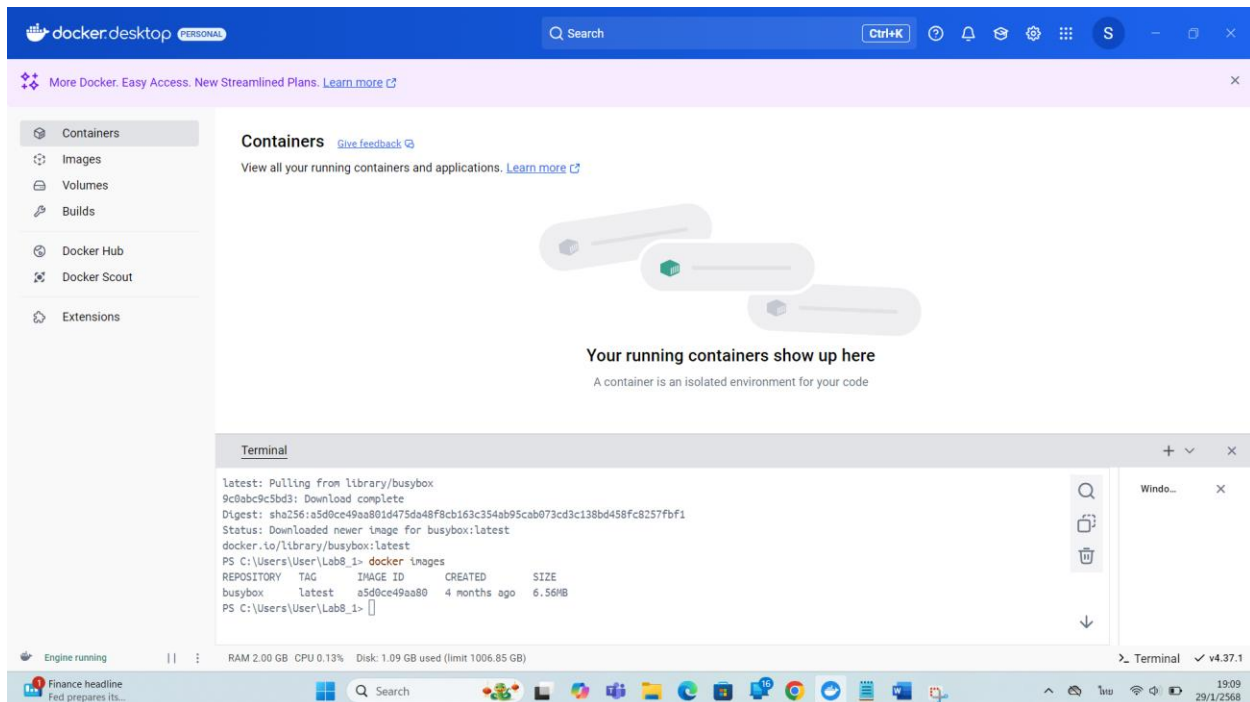
1. ติดตั้ง Docker desktop ลงบนเครื่องคอมพิวเตอร์ โดยดาวน์โหลดจาก <https://www.docker.com/get-started>
2. สร้าง Account บน Docker hub (<https://hub.docker.com/signup>)
3. กำหนดให้ \$ หมายถึง Command prompt และ <> หมายถึง ให้ป้อนค่าของพารามิเตอร์ที่กำหนด

แบบฝึกปฏิบัติที่ 8.1 Hello world - รัน Container จาก Docker image

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

Lab Worksheet

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



- (1) สิ่งที่อยู่ภายใต้คอลัมน์ Repository คืออะไร
ชื่อ docker image ที่ดาวน์โหลด
- (2) Tag ที่ใช้บ่งบอกถึงอะไร
เวอร์ชันของ docker image
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

The screenshot shows the Docker Desktop interface with three running containers: interesting_banach, elegant_merkle, and magical_chaplygin. The terminal window shows the execution of the following commands:

```
PS C:\Users\User\Lab8_1> docker run busybox
PS C:\Users\User\Lab8_1> docker run -it busybox sh
/ # ls
bin      dev      etc      home    lib      lib64    proc    root    sys      tmp      usr      var
/ # ls -ls
total 48
drwxr-xr-x 1 root root      4096 Jan 29 12:14 .
drwxr-xr-x 1 root root      4096 Jan 29 12:14 ..
-rwxr-xr-x 1 root root         0 Jan 29 12:14 .dockerenv
drwxr-xr-x 2 root root     12288 Sep 26 21:31 bin
drwxr-xr-x 4 root root      4096 Sep 26 21:31 var
/ # exit
PS C:\Users\User\Lab8_1> docker run busybox echo "Hello สวัสดี ใ้คุณ from busybox"
```

The second screenshot shows the same interface after running the command `docker ps -a`, displaying a table of all containers, including those that have exited.

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
97bd9c390522	busybox	"echo 'Hello สวัสดี'"	15 seconds ago	Exited (0) 14 seconds ago		magical_chaplygin
99c2c9db7ce4	busybox	"sh"	About a minute ago	Exited (0) About a minute ago		elegant_merkle
7695293b6a62	busybox	"sh"	2 minutes ago	Exited (0) 2 minutes ago		interesting_banach

- (1) เมื่อใช้ option `-it` ในคำสั่ง `run` ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป
- i (interactive mode) ทำให้ Docker รักษา input จากผู้ใช้ (เช่น การพิมพ์คำสั่ง)
 - t (pseudo-TTY) ให้ Docker จำลอง terminal เพื่อให้ใช้งาน shell ได้สะดวกขึ้น
 - it จะเปิด interactive shell ให้เราป้อนคำสั่งไปยัง container ได้โดยตรง

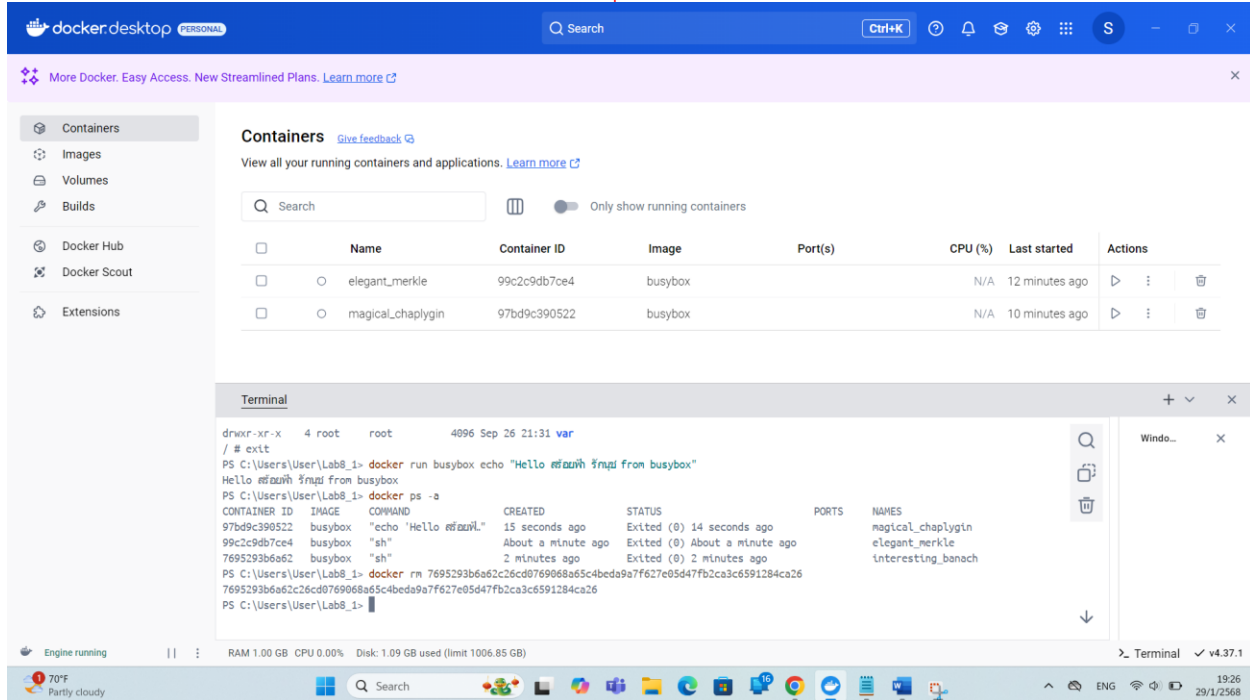
Lab Worksheet

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

คอลัมน์ STATUS แสดง สถานะของ container ว่า container กำลังทำงาน หรือ หยุดไปแล้ว พร้อมเวลาที่เกิดขึ้น

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

[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`

Lab Worksheet

FROM busybox

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

CMD echo "ชื่อ-นามสกุล รหัสนักศึกษา ชื่อเล่น"

EOF

หรือใช้คำสั่ง

\$ touch Dockerfile

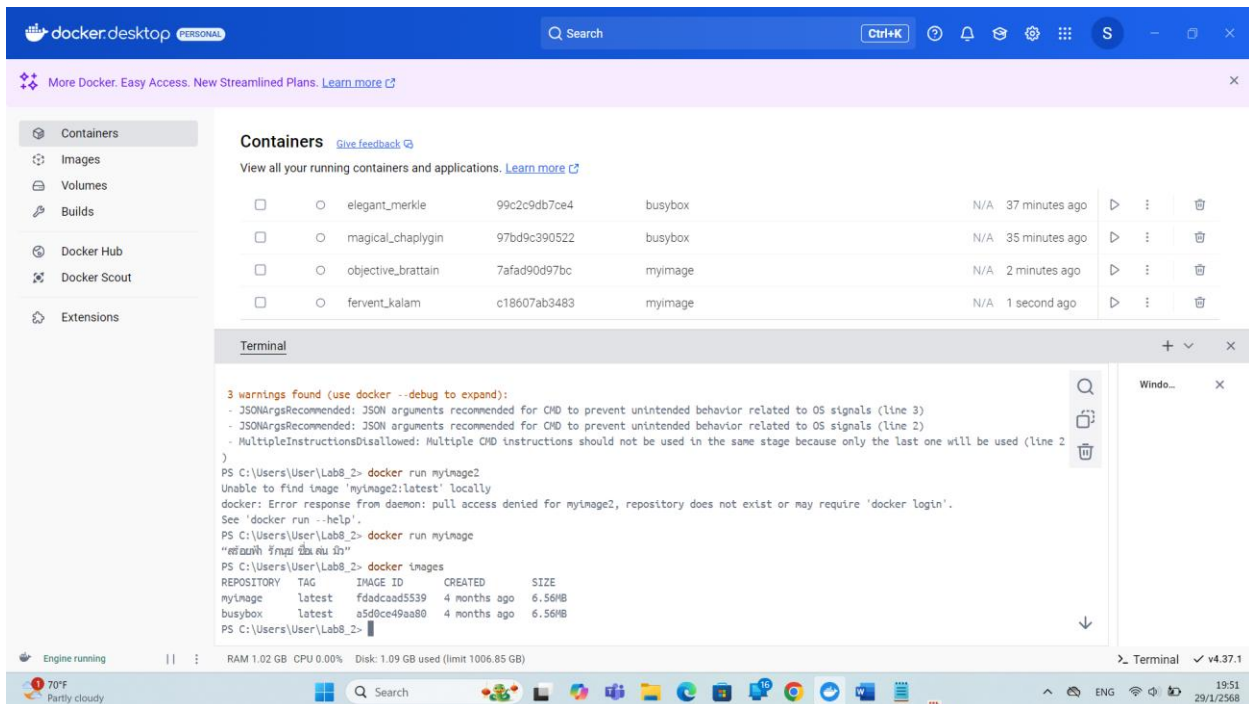
แล้วใช้ Text Editor ในการใส่เนื้อหาแทน

5. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้

\$ docker build -t <ชื่อ Image> .

6. เมื่อ Build สำเร็จแล้ว ให้ทำการรัน Docker image ที่สร้างขึ้นในขั้นตอนที่ 5

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



- (1) คำสั่งที่ใช้ในการ run คือ

`docker run myimage`

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

-t ย่อมาจาก --tag ใช้สำหรับ ตั้งชื่อ (Tag) ให้กับ Docker Image

ถ้า ไม่มี -t → Docker จะสร้าง Image แต่ไม่มีชื่อ ทำให้ใช้งานยาก

ถ้า ใช้ -t myimage → กำหนดชื่อของ Image เป็น myimage

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

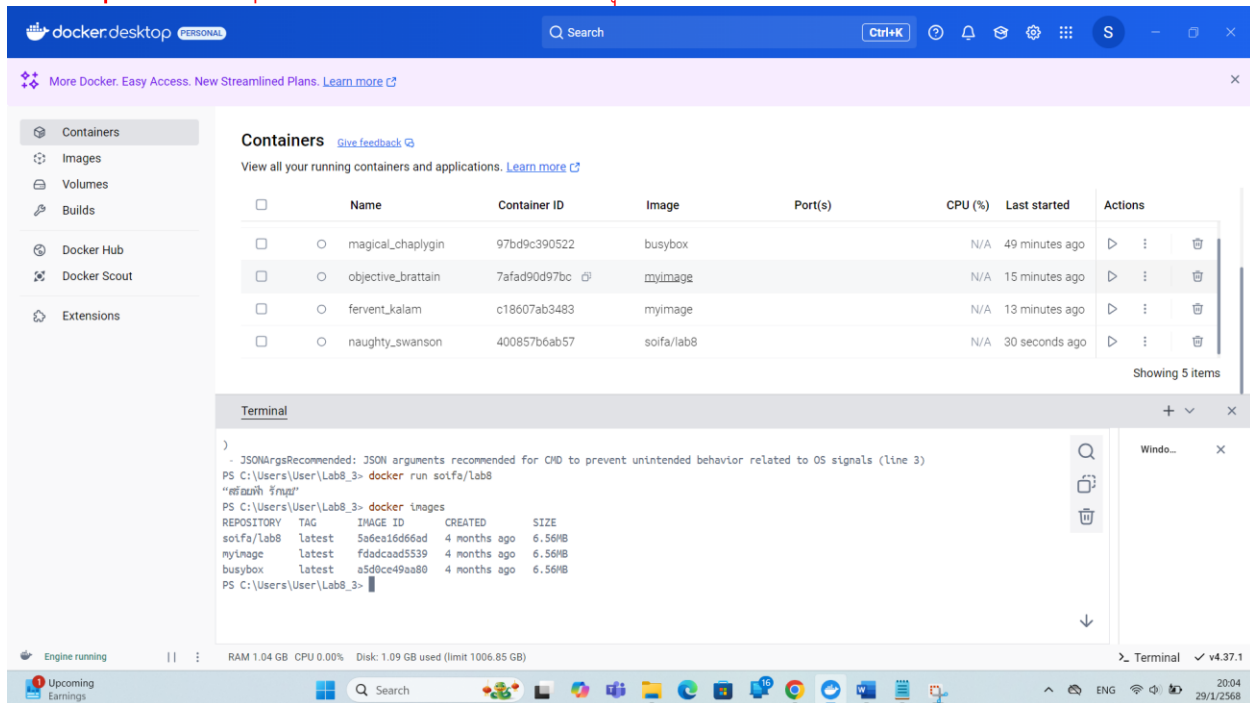
7. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้

```
$ 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



6. ทำการ Push ตัว Docker image ไปไว้บน Docker Hub โดยใช้คำสั่ง

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

ในกรณีที่ติดปัญหาไม่ได้ Login ไว้ก่อน ให้ใช้คำสั่งต่อไปนี้ เพื่อ Login ก่อนทำการ Push

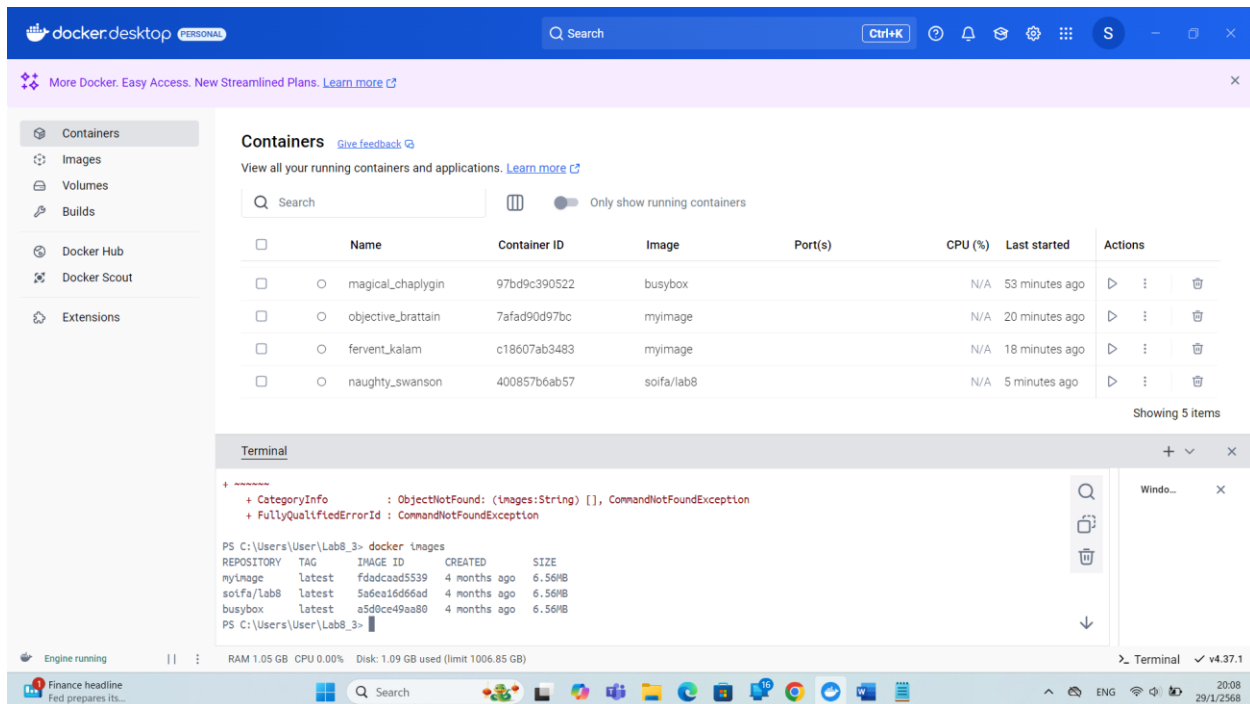
\$ docker login แล้วป้อน Username และ Password ตามที่ระบุใน Command prompt หรือใช้คำสั่ง

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

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

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

Lab Worksheet

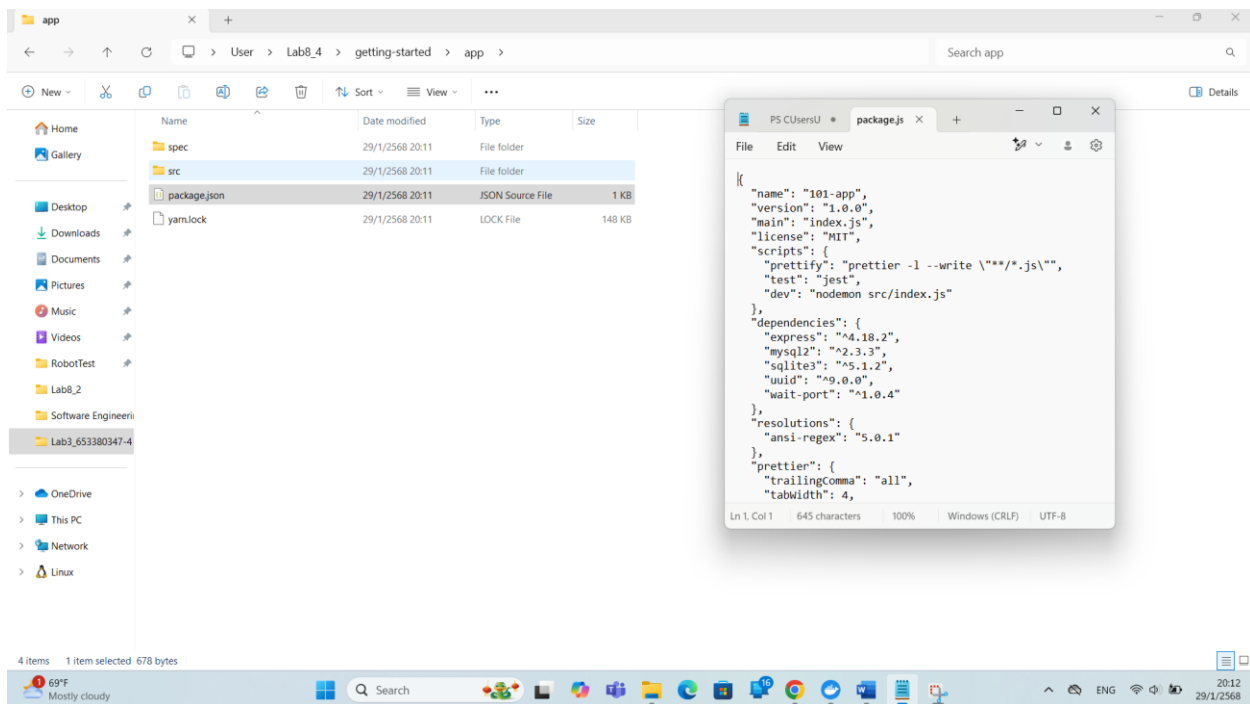


แบบฝึกปฏิบัติที่ 8.4: การ Build แอปพลิเคชันจาก Container image และการ Update แอปพลิเคชัน

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

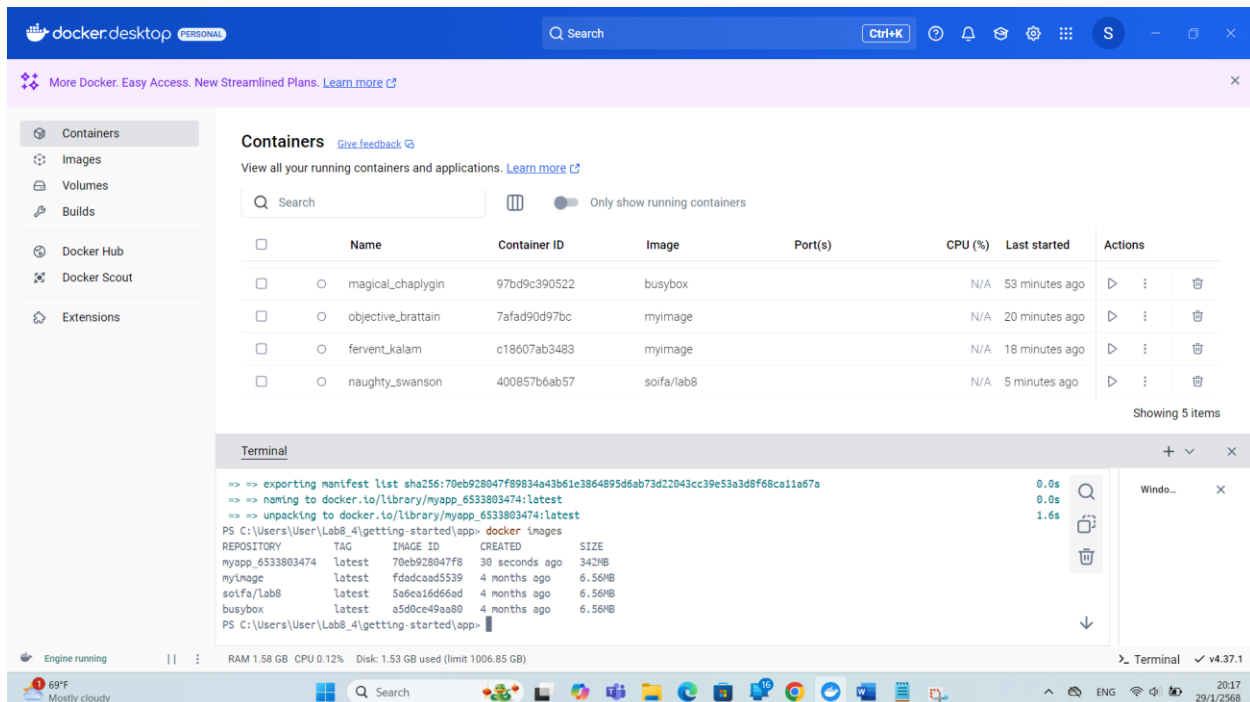
Lab Worksheet



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 โดยกำหนดใช้ชื่อ image เป็น myapp_รหัสสนศ. ไม่มีขีด
\$ docker build -t <myapp_รหัสสนศ. ไม่มีขีด> .

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

Lab Worksheet

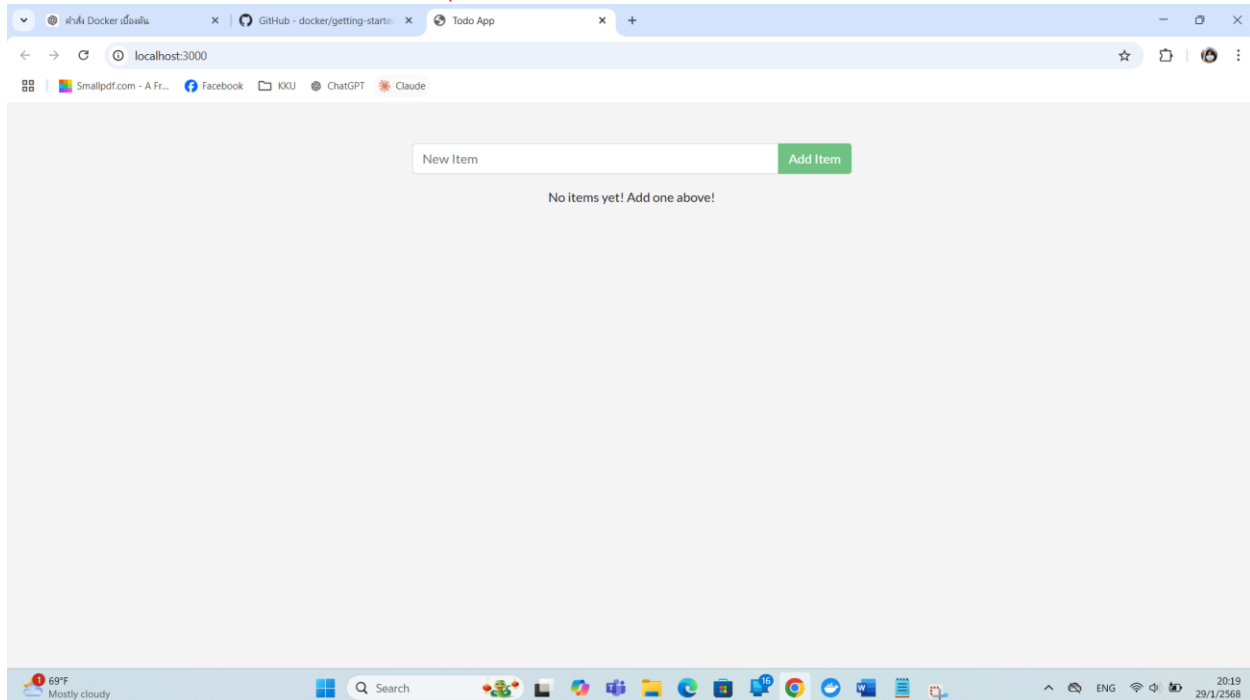


6. ทำการ Start ตัว Container ของแอปพลิเคชันที่สร้างขึ้น โดยใช้คำสั่ง

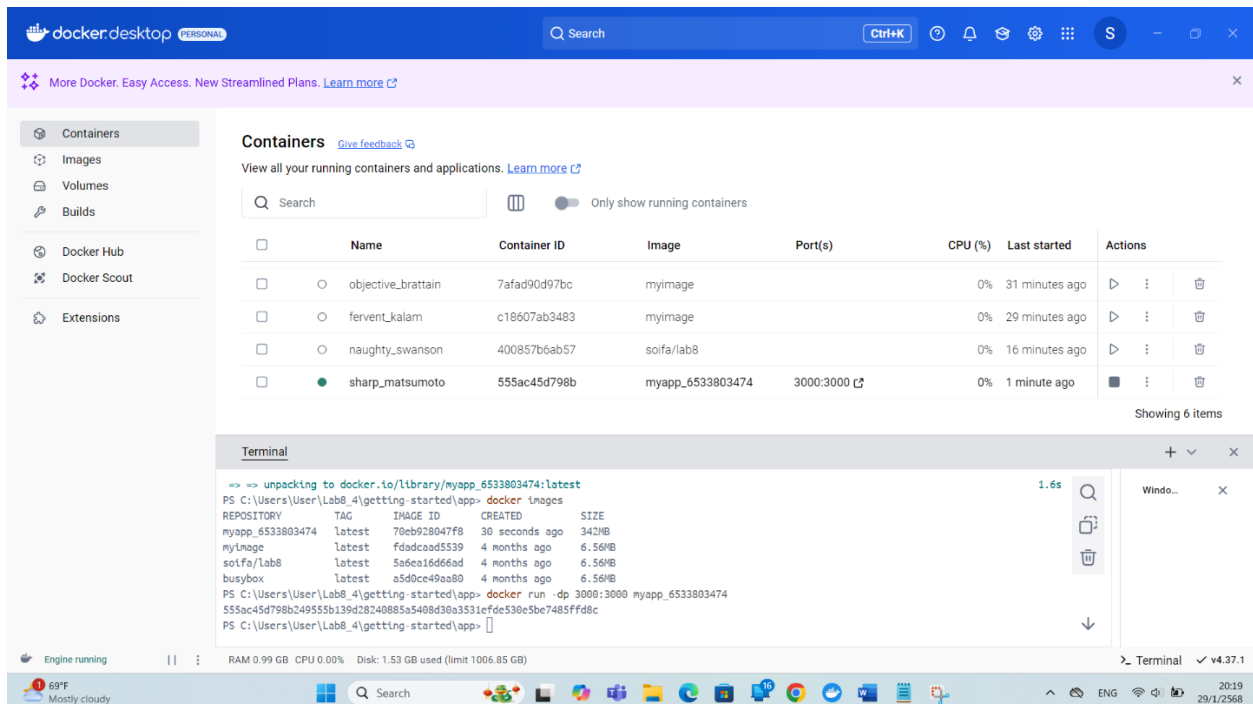
\$ docker run -dp 3000:3000 <myapp_รหัสศ. ไม่มีขีด>

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

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



Lab Worksheet



หมายเหตุ: นศ.สามารถทดลองเล่น Web application ที่ทำงานอยู่ได้

8. ทำการแก้ไข Source code ของ Web application ดังนี้

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

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

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

ชื่อและนามสกุลของนักศึกษา</p>

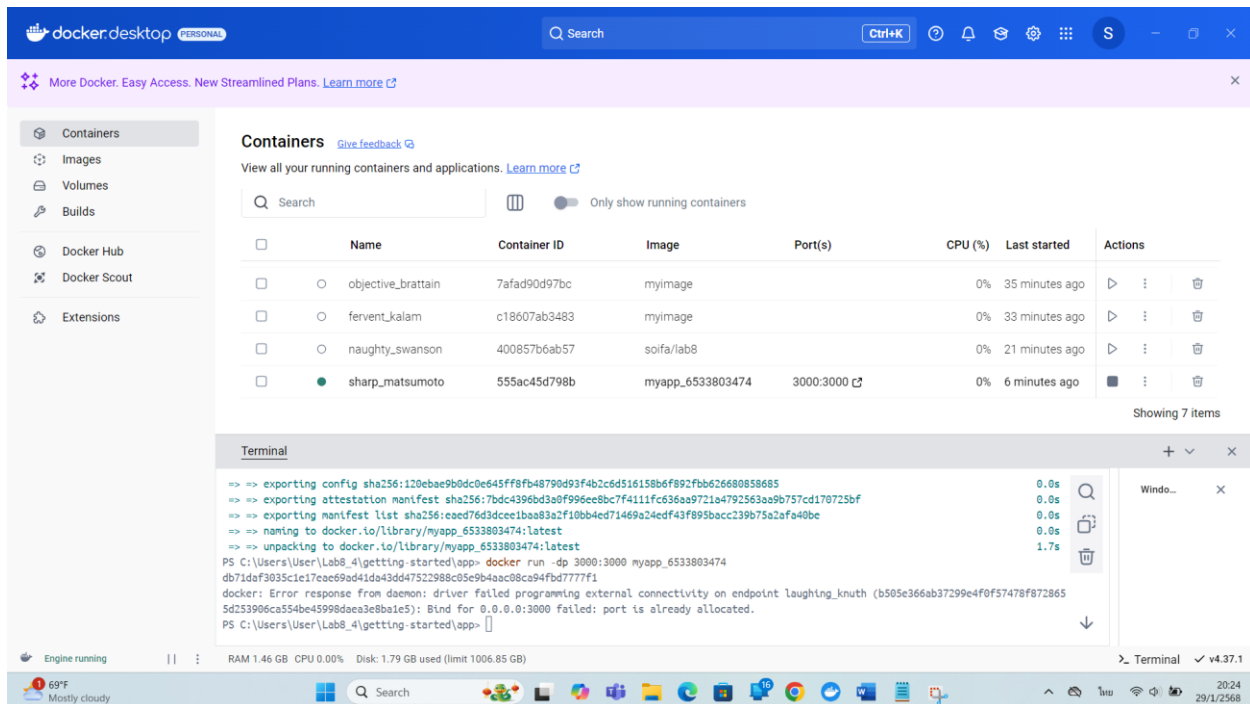
b. Save ไฟล์ให้เรียบร้อย

9. ทำการ Build Docker image โดยใช้คำสั่งเดียวกันกับข้อ 5

10. Start และรัน Container ตัวใหม่ โดยใช้คำสั่งเดียวกันกับข้อ 6

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

Lab Worksheet



(1) Error ที่เกิดขึ้นหมายความว่าอย่างไร และเกิดขึ้นเพราะอะไร

Docker พยายามผูก (bind) พอร์ต 3000 ของเครื่องโฮสต์ (0.0.0.0:3000) กับ Container แต่ล้มเหลว เพราะพอร์ต 3000 ถูกใช้งานอยู่แล้ว โดยโปรเซสอื่น

11. ลบ Container ของ Web application เวอร์ชันก่อนแก้ไขออกจากระบบ โดยใช้วิธีใดวิธีหนึ่งดังต่อไปนี้

a. ผ่าน Command line interface

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

b. ผ่าน Docker desktop

- ไปที่หน้าต่าง Containers
- เลือกไอคอนถังขยะในแถวของ Container ที่ต้องการจะลบ
- ยืนยันโดยการกด Delete forever

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 a web browser window with the address bar set to 'localhost:3000'. The page displays a 'Todo App' with a 'New Item' input field and an 'Add Item' button. Below the input, it says 'There is no TODO item. Please add one to the list. By สร้อยฟ้า รักบุญ'. The browser's address bar shows 'localhost:3000'. Below the browser, the Docker Desktop interface is visible, showing the 'Containers' tab with a list of running containers. The containers listed are 'objective_brattain', 'fervent_kalam', 'naughty_swanson', and 'laughing_knuth'. The 'laughing_knuth' container is highlighted, and its details are shown below, including the image 'myapp_6533803474' and the port '3000:3000'. The terminal window at the bottom shows the command 'docker images' and its output, listing the images and their sizes.

Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
objective_brattain	7afad90d97bc	myimage		0%	41 minutes ago	▶ ⋮ 🗑
fervent_kalam	c18607ab3483	myimage		0%	39 minutes ago	▶ ⋮ 🗑
naughty_swanson	400857b6ab57	soifa/lab8		0%	26 minutes ago	▶ ⋮ 🗑
laughing_knuth	db71daf3035c	myapp_6533803474	3000:3000	0%		▶ ⋮ 🗑

Showing 7 items

```

Terminal
555ac45d798b
PS C:\Users\User\Lab8_4\getting-started\app> docker images
REPOSITORY    TAG       IMAGE ID   CREATED   SIZE
myapp_6533803474 latest    eed76d3dcee 5 minutes ago 342MB
myimage        latest    fdadcaad5339 4 months ago 6.56MB
soifa/lab8     latest    5a6ea16d66ad 4 months ago 6.56MB
busybox        latest    a5d9ce49aa80 4 months ago 6.56MB
PS C:\Users\User\Lab8_4\getting-started\app> docker run -dp 3000:3000 myapp_6533803474
488e8e76d01a1a4c1e949255d4e239d597ed640e9000d583223d411f95f31df3
PS C:\Users\User\Lab8_4\getting-started\app>
  
```

Engine running | RAM 1.49 GB CPU 0.00% Disk: 1.68 GB used (limit 1006.85 GB) | Terminal v4.37.1

แบบฝึกปฏิบัติที่ 8.5: เริ่มต้นสร้าง Pipeline อย่างง่ายสำหรับการ Deploy ด้วย Jenkins

1. เปิด Command line หรือ Terminal บน Docker Desktop

Lab Worksheet

2. ป้อนคำสั่งและทำการรัน container โดยผูกพอร์ต

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

หรือ

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

3. บันทึกรหัสผ่านของ Admin user ไว้สำหรับ log-in ในครั้งแรก

[Check point#12] Capture หน้าจอที่แสดงผล Admin password

The screenshot shows the Docker Desktop interface. On the left, the 'Containers' tab is selected. The main area displays a table of running containers:

Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
naughty_swanson	400857b6ab57	soifa/lab8		0%	47 minutes ago	[Stop] [Refresh] [Delete]
laughing_knuth	db71daf3035c	myapp_6533803474	3000:3000	0%		[Stop] [Refresh] [Delete]
competent_moser	488e9e76d01a	myapp_6533803474	3000:3000	0%	22 minutes ago	[Stop] [Refresh] [Delete]
dreamy_boyd	78ed358159a9	jenkins/jenkins:lts-jdk17	50000:50000 Show all ports (2)	4.09%	17 minutes ago	[Stop] [Refresh] [Delete]

Below the table, a terminal window is open, showing the output of the Jenkins container startup. The logs indicate that Jenkins is fully up and running. The terminal prompt shows the user has entered the container and is ready to execute commands.

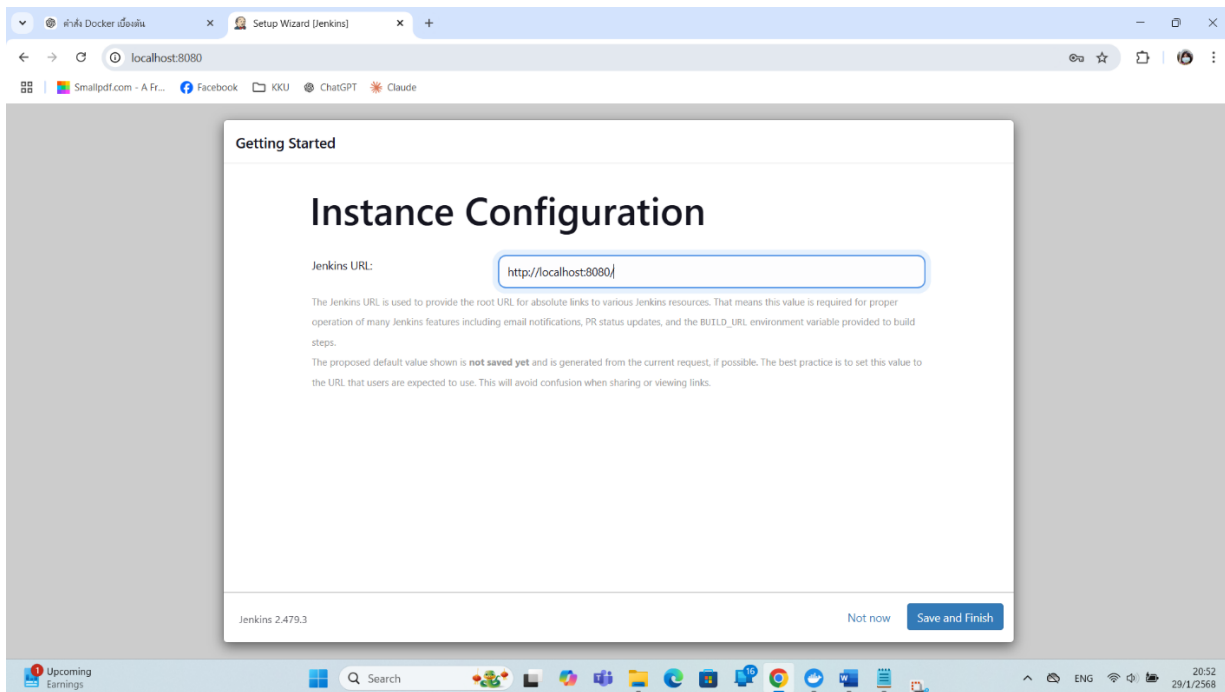
4. เมื่อได้รับการยืนยันว่า Jenkins is fully up and running ให้เปิดบราวเซอร์ และป้อนที่อยู่เป็น localhost:8080

5. ทำการ Unlock Jenkins ด้วยรหัสผ่านที่ได้ในข้อที่ 3

6. สร้าง Admin User โดยใช้ username เป็นชื่อจริงของนักศึกษาพร้อมรหัสสี่ตัวท้าย เช่น somsri_3062

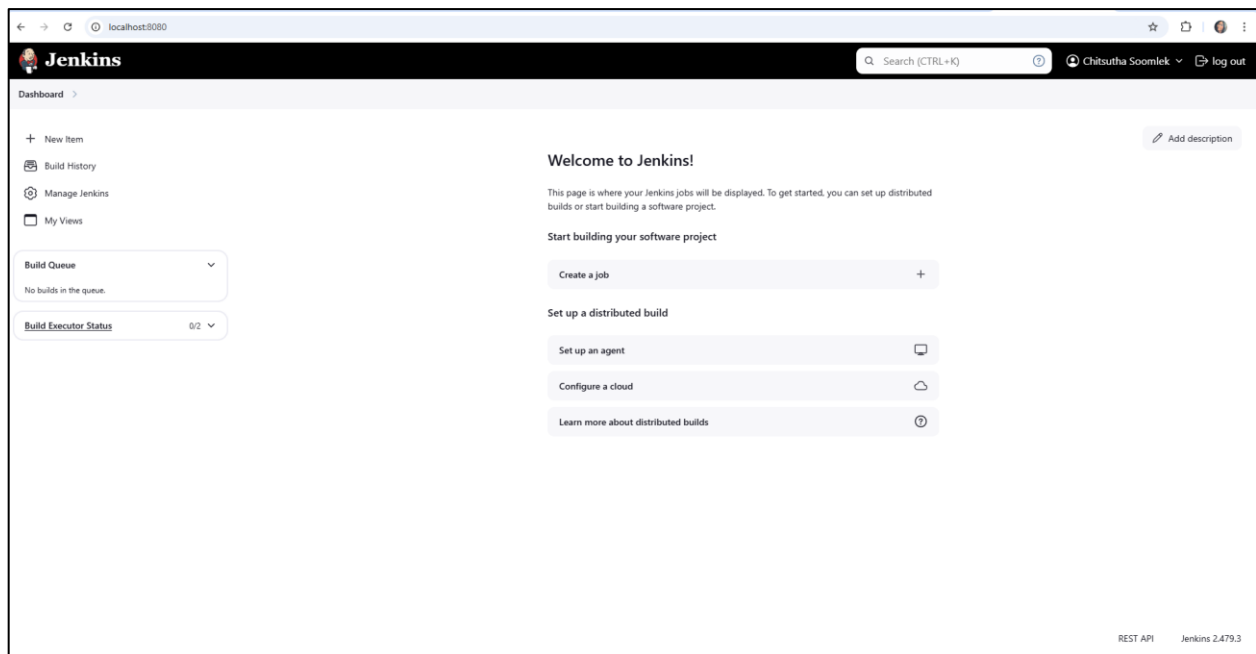
[Check point#13] Capture หน้าจอที่แสดงผลการตั้งค่า

Lab Worksheet



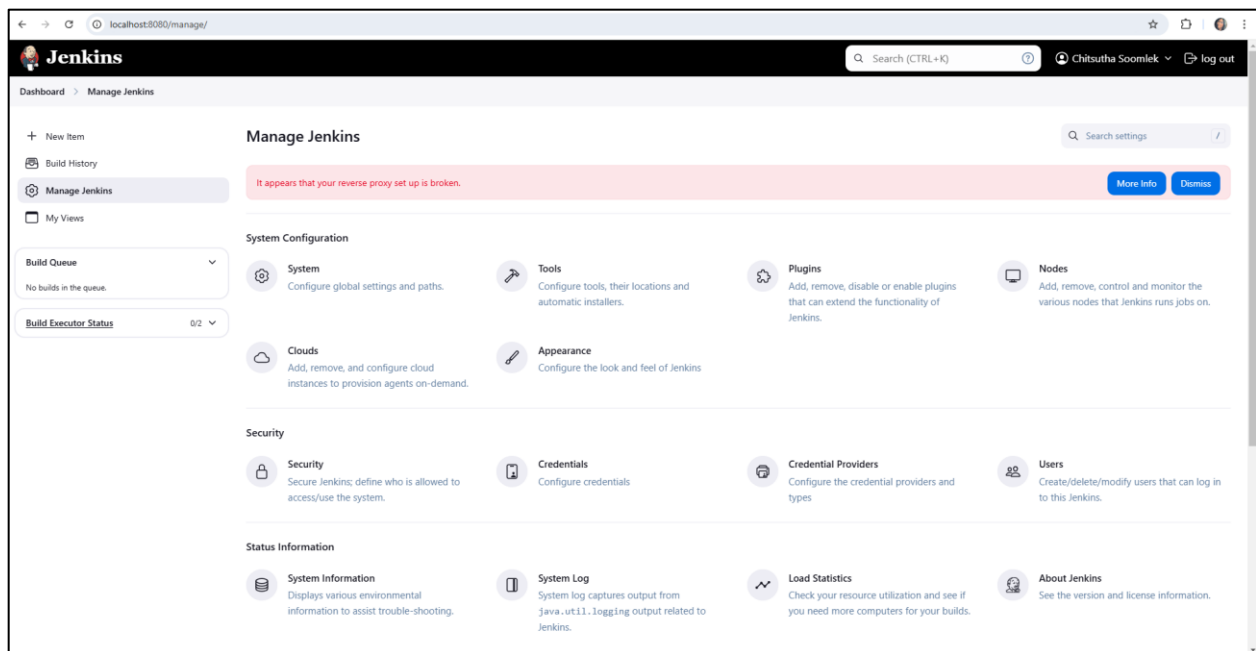
7. กำหนด Jenkins URL เป็น <http://localhost:8080/lab8>

8. เมื่อติดตั้งเรียบร้อยแล้วจะพบกันหน้า Dashboard ดังแสดงในภาพ

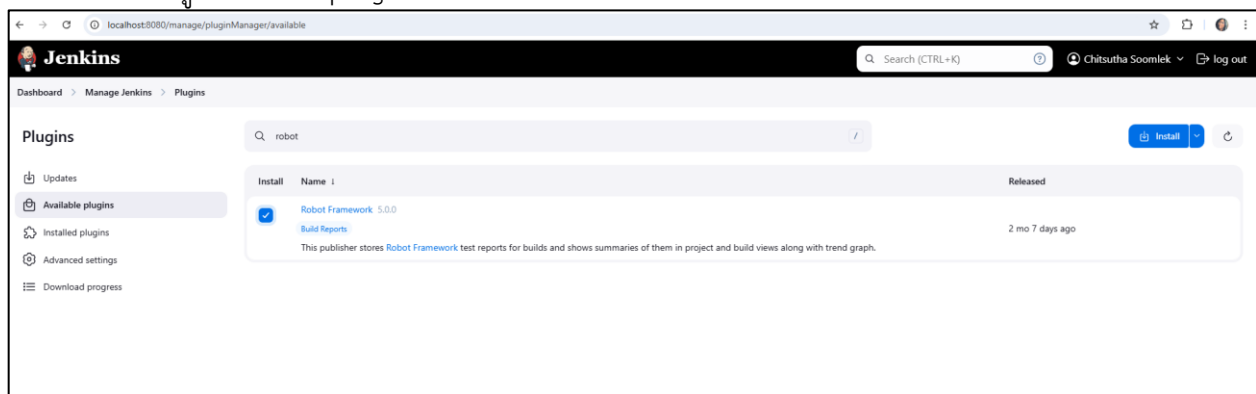


9. เลือก Manage Jenkins แล้วไปที่เมนู Plugins

Lab Worksheet

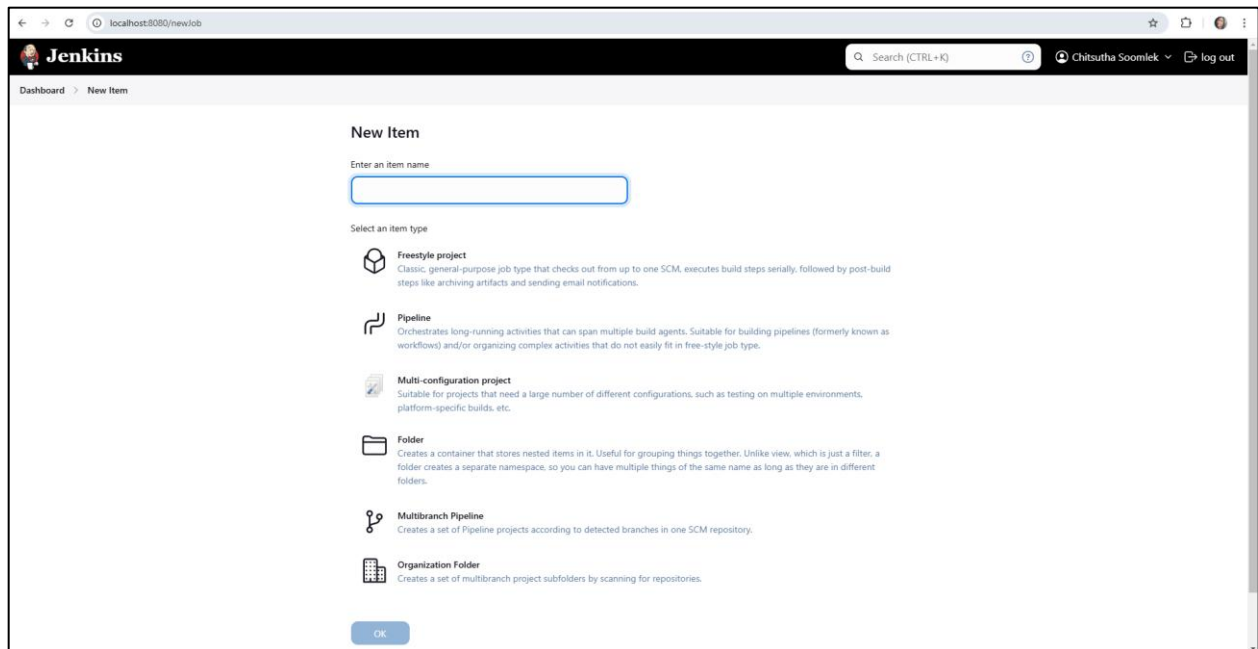


10. ไปที่เมนู Available plugins แล้วเลือกติดตั้ง Robotframework เพิ่มเติม



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

Lab Worksheet



12. นำไฟล์ .robot ที่ทำให้แบบฝึกปฏิบัติที่ 7 (Lab#7) ไปไว้บน Repository ของนักศึกษา จากนั้นตั้งค่าที่จำเป็นในหน้านี้อย่างหมด 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 shows two screenshots of the Jenkins web interface. The top screenshot displays the 'Configure' page for a job named 'UAT', specifically the 'General' tab. The 'Description' field contains 'Lab 8.5'. Under 'GitHub project', the 'Project url' is set to 'https://github.com/Soifa524/lab8.5'. The bottom screenshot shows the 'Source Code Management' tab, where 'Git' is selected. The 'Repository URL' is 'https://github.com/Soifa524/lab8.5.git', and the 'Branches to build' section is empty.

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

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

Lab Worksheet

แล้วนับว่าซอฟต์แวร์มีปัญหา -> ตั้งค่า Threshold เป็น % ของการทดสอบที่ผ่านแล้วนับว่าซอฟต์แวร์มีอยู่ในสถานะที่สามารถนำไปใช้งานได้ (เช่น 20, 80)

13. กด Apply และ Save

14. สั่ง Build Now

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

The image shows two screenshots from a Jenkins environment. The top screenshot is a browser view of the Jenkins console output for a build named 'UAT #32'. The output shows a test suite 'Valid Login' that failed due to a 'KeyError: selenium.webdriver'. The bottom screenshot is the Jenkins dashboard for the 'UAT' pipeline, showing the latest build (#33) as failed. The dashboard includes a 'Robot Framework Tests Trend' chart and a list of recent builds.

Console Output (UAT #32):

```
Valid Login :: A test suite with a single test for valid login.
=====
Valid Login                                     | FAIL |
Evaluating expression "sys.modules['selenium.webdriver'].ChromeOptions()" failed: KeyError: 'selenium.webdriver'

Also teardown failed:
No keyword with name 'Close Browser' found.
-----
Valid Login :: A test suite with a single test for valid login. | FAIL |
1 test, 0 passed, 1 failed
=====
Output: /var/jenkins_home/workspace/UAT/results/output.xml
Log: /var/jenkins_home/workspace/UAT/results/log.html
Report: /var/jenkins_home/workspace/UAT/results/report.html
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:
Done!
-Copying log files to build dir:
Done!
-Assigning results to build:
Done!
-Checking thresholds:
Done!
Done publishing Robot results.
Finished: FAILURE
```

Jenkins Dashboard (UAT):

Lab 8.5

Latest Robot Results:

Total	Failed	Passed	Skipped	Pass %
All tests	6	6	0	0.0

[Browse results](#)
[Open report.html](#)
[Open log.html](#)

Permalinks

- Last build (#33), 11 sec ago
- Last failed build (#33), 11 sec ago
- Last unsuccessful build (#33), 11 sec ago
- Last completed build (#33), 11 sec ago

Robot Framework Tests Trend (all tests)

Number of test cases: 6

Build

Zoom to changes ☐ Show only failed ☐ Show only critical ☐ all Max builds Show bigger image