

Lab Worksheet

ชื่อ-นามสกุล สุภูมิ มณีราชกิจ รหัสนักศึกษา 6533802830-4. 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

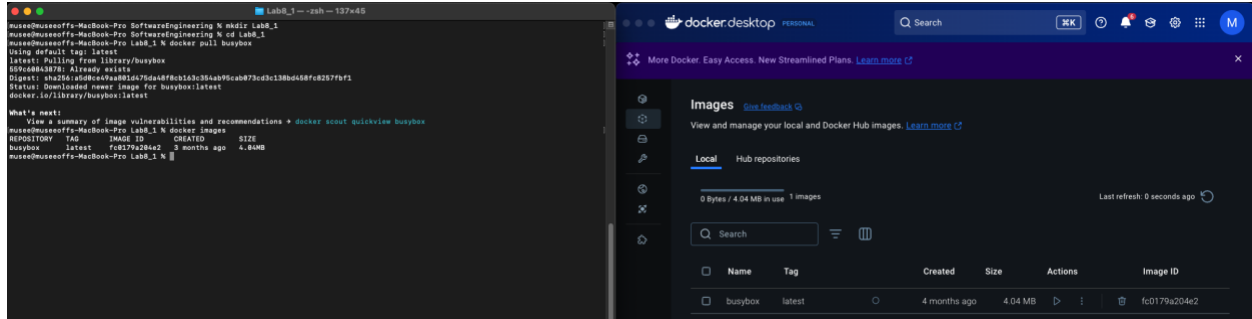
1. ติดตั้ง Docker desktop ลงบนเครื่องคอมพิวเตอร์ โดยดาวน์โหลดจาก <https://www.docker.com/get-started>
2. สร้าง Account บน Docker hub (<https://hub.docker.com/signup>)
3. กำหนดให้ \$ หมายถึง Command prompt และ <> หมายถึง ให้ป้อนค่าของพารามิเตอร์ที่กำหนด

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 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) แสดงผลลัพธ์ที่ได้พร้อมกับตอบคำถามต่อไปนี้

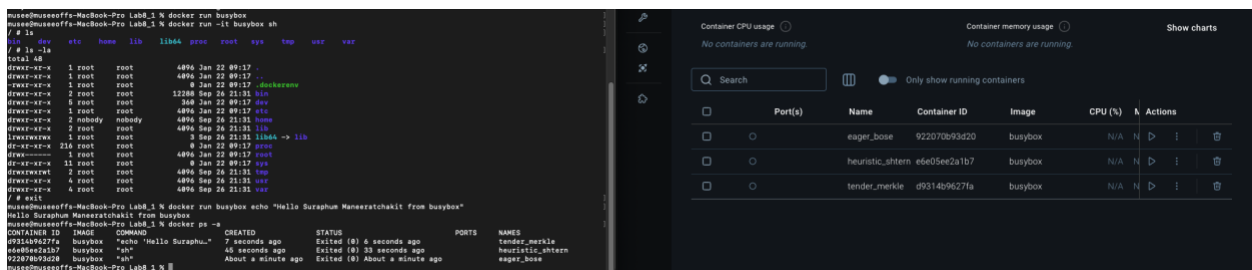


(1) สิ่งที่อยู่ภายใต้คอลัมน์ Repository คืออะไร image

(2) Tag ที่ใช้บ่งบอกถึงอะไร version

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



(1) เมื่อใช้ option -it ในคำสั่ง run ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป

Lab Worksheet

มีผลต่อการทำงานของคอนเทนเนอร์โดยเปิดใช้งาน Interactive Mode และ Pseudo-TTY เพื่อให้ผู้ใช้งานสามารถโต้ตอบกับคอนเทนเนอร์ผ่าน Terminal ได้โดยตรง

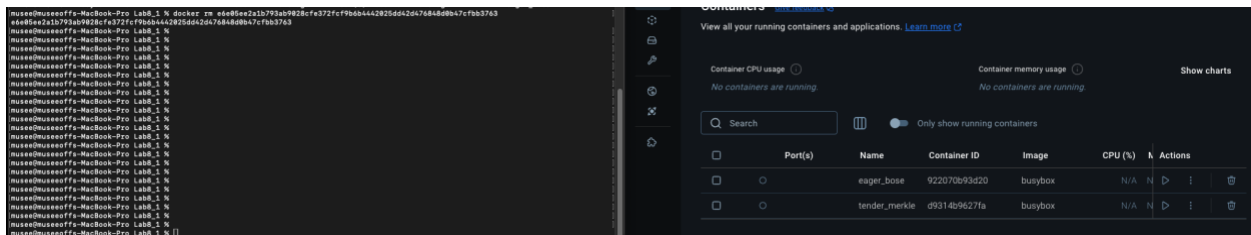
(2) คอลัมน์ STATUS จากการรันคำสั่ง `docker ps -a` แสดงถึงข้อมูลอะไร

แสดงสถานะการทำงานของแต่ละคอนเทนเนอร์ในระบบ โดยข้อมูลในคอลัมน์นี้ระบุถึง:

- สถานะปัจจุบันของคอนเทนเนอร์:
 - บอกว่าคอนเทนเนอร์กำลังรัน, หยุดทำงาน, หรือหยุดชั่วคราว.
- ระยะเวลาที่คอนเทนเนอร์อยู่ในสถานะนั้น:
 - ระบุช่วงเวลาที่คอนเทนเนอร์เข้าสู่สถานะปัจจุบัน เช่น “Up 5 minutes” หรือ “Exited (0) 2 hours ago”

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

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



แบบฝึกปฏิบัติที่ 8.2: สร้าง Docker file และ Docker image

- เปิดใช้งาน Docker desktop และ Login ด้วย Username และ Password ที่ลงทะเบียนกับ Docker Hub เอาไว้
- เปิด Command line หรือ Terminal จากนั้นสร้าง Directory ชื่อ Lab8_2
- ย้ายตำแหน่งปัจจุบันไปที่ Lab8_2 เพื่อใช้เป็น Working directory
- 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”

Lab Worksheet

`docker run firstimage`

- (2) Option -t ในคำสั่ง `$ docker build` ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป
สามารถกำหนด Tag ของ Image ได้ โดย `<repository_name>:<tag>` เช่น `myapp:latest`.

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

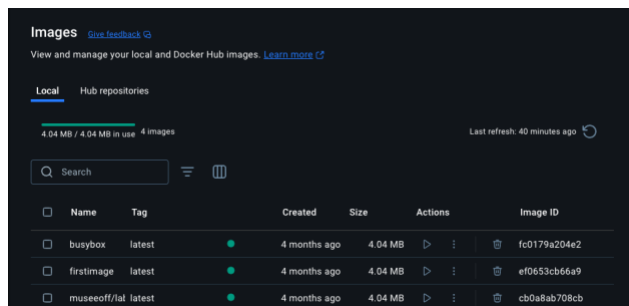
7. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้

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

5. ทำการรัน Docker image บน Container ในเครื่องของตัวเองเพื่อทดสอบผลลัพธ์ ด้วยคำสั่ง

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

[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)

```
lab8_3 - ssh - 137a65
heredoc CMD echo "Hi there. My work is done. You can run the Docker Image."
heredoc CMD echo "Suraphim Maneerachai@ 65188283-4"
heredoc EOF

museumuseoffs-MacBook-Pro Lab8_3 N docker build -t museuffff/lab8 .
=> Building RHEL (v3) FPMImage                                docker/desktop-linux
=> [internal] load build definition from Dockerfile                                0.0s
=> => transferring Dockerfile 374B                                                0.0s
=> WARN: JNAGN:JNAGN:JNAGN:JNAGN arguments recommended for CMD to prevent unintended behavior related to OS signals (line 2) 0.0s
=> WARN: Multiple instructions allowed: Multiple CMD instructions should not be used in the same stage because only the last one will be used (line 2) 0.0s
=> WARN: JNAGN:JNAGN:JNAGN:JNAGN arguments recommended for CMD to prevent unintended behavior related to OS signals (line 3) 0.0s
=> [internal] load definitions from Dockerfile                                  0.0s
=> => transferring Dockerfile 374B                                                0.0s
=> CACHED [1/1] FROM docker.io/library/busybox:latest                          0.0s
=> exporting image                                                       0.0s
=> == exporting layers                                                      0.0s
=> == writing image sha256:b48c7f945dabce9e077f9da251f5f7eb796af45ad52        0.0s
=> == sending to docker.io/museuffff/lab8                                     0.0s

View build details: docker-desktop://build/desktop-linux/desktop-linux/docker-image/c277a19cc39cccba8ee1235

# warnings found (see docker --help to expand):
# JNAGN:JNAGN:JNAGN:JNAGN arguments recommended for CMD to prevent unintended behavior related to OS signals (line 2)
# Multiple instructions allowed: Multiple CMD instructions should not be used in the same stage because only the last one will be used (line 2)
# JNAGN:JNAGN:JNAGN:JNAGN arguments recommended for CMD to prevent unintended behavior related to OS signals (line 3)

What's next:
View a summary of image vulnerabilities and recommendations > docker scout quickview
museumuseoffs-MacBook-Pro Lab8_3 N docker run museuffff/lab8
Suraphim Maneerachai@ 65188283-4
museumuseoffs-MacBook-Pro Lab8_3 N docker push museuffff/lab8
Using default tag: latest
The push refers to repository (docker.io/museuffff/lab8)
c13efcf8a0b1 Mounted from library/busybox
manifest blob unknown blob unknown to registry
museumuseoffs-MacBook-Pro Lab8_3 N docker login -u museuffff -p Suraphim1113
WARNING! Using --password via the CLI is insecure. Use --password-stdin.
Login Succeeded
museumuseoffs-MacBook-Pro Lab8_3 N docker push museuffff/lab8
Using default tag: latest
The push refers to repository (docker.io/museuffff/lab8)
c13efcf8a0b1 Layer already exists
latest digest: sha256:f7945dabce9e077f9da251f5f7eb796af45ad52 size: 527
museumuseoffs-MacBook-Pro Lab8_3 N
```

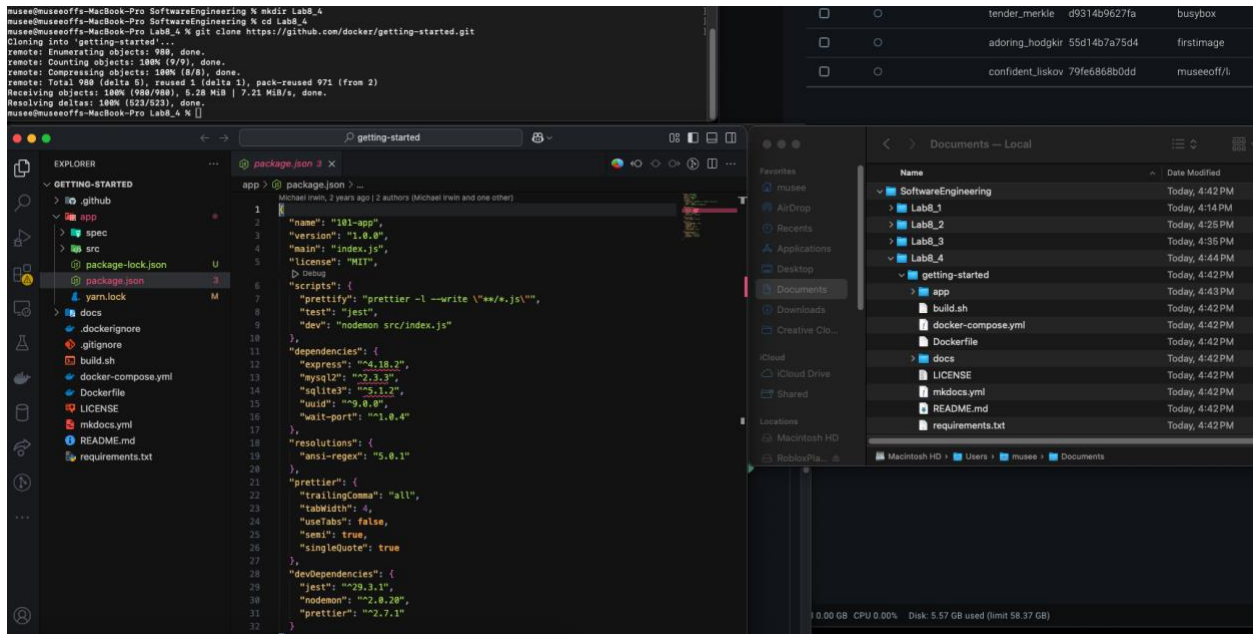
แบบฝึกปฏิบัติที่ 8.4: การ Build แอปพลิเคชันจาก Container image และการ Update แอปพลิเคชัน

1. เปิด Command line หรือ Terminal จากนั้นสร้าง Directory ชื่อ Lab8_4
2. ทำการ Clone ซอร์สโค้ดของเว็บแอปพลิเคชันจาก GitHub repository
<https://github.com/docker/getting-started.git> ลงใน Directory ที่สร้างขึ้น โดยใช้คำสั่ง

```
$ git clone https://github.com/docker/getting-started.git
```
3. เปิดดูองค์ประกอบภายใน getting-started/app เมื่อพบไฟล์ package.json ให้ใช้ Text editor ในการเปิดอ่าน

Lab Worksheet

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



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

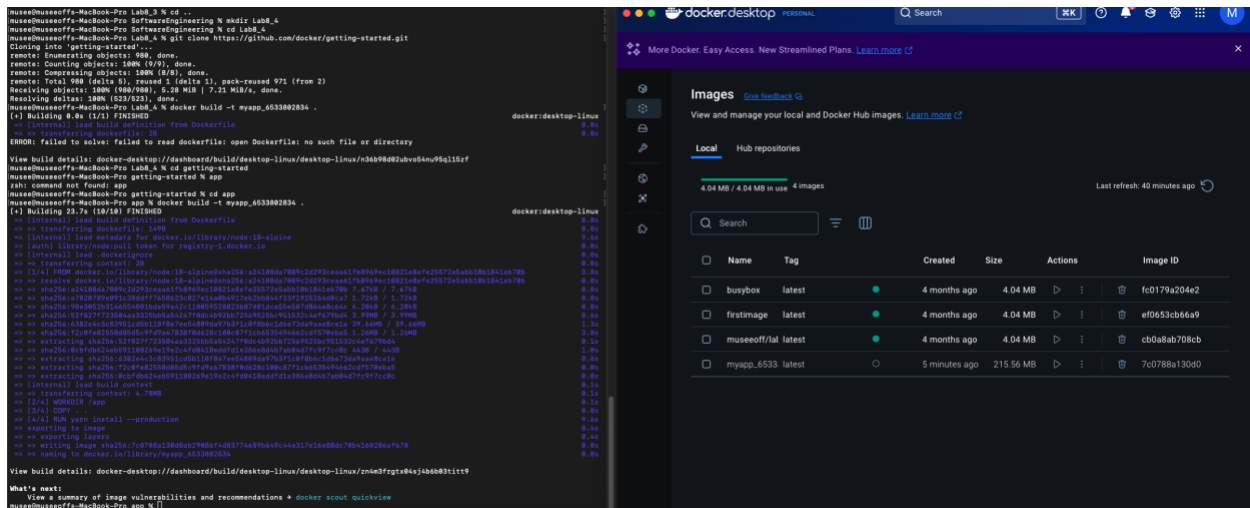
5. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้ โดยกำหนดให้ชื่อ image เป็น myapp_รหัสสนศ. ไม่มีขีด

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

Lab Worksheet

[Check point#8] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง)

แสดงคำสั่งและผลลัพธ์ที่ได้ทางหน้าจอ



6. ทำการ Start ตัว Container ของแอปพลิเคชันที่สร้างขึ้น โดยใช้คำสั่ง

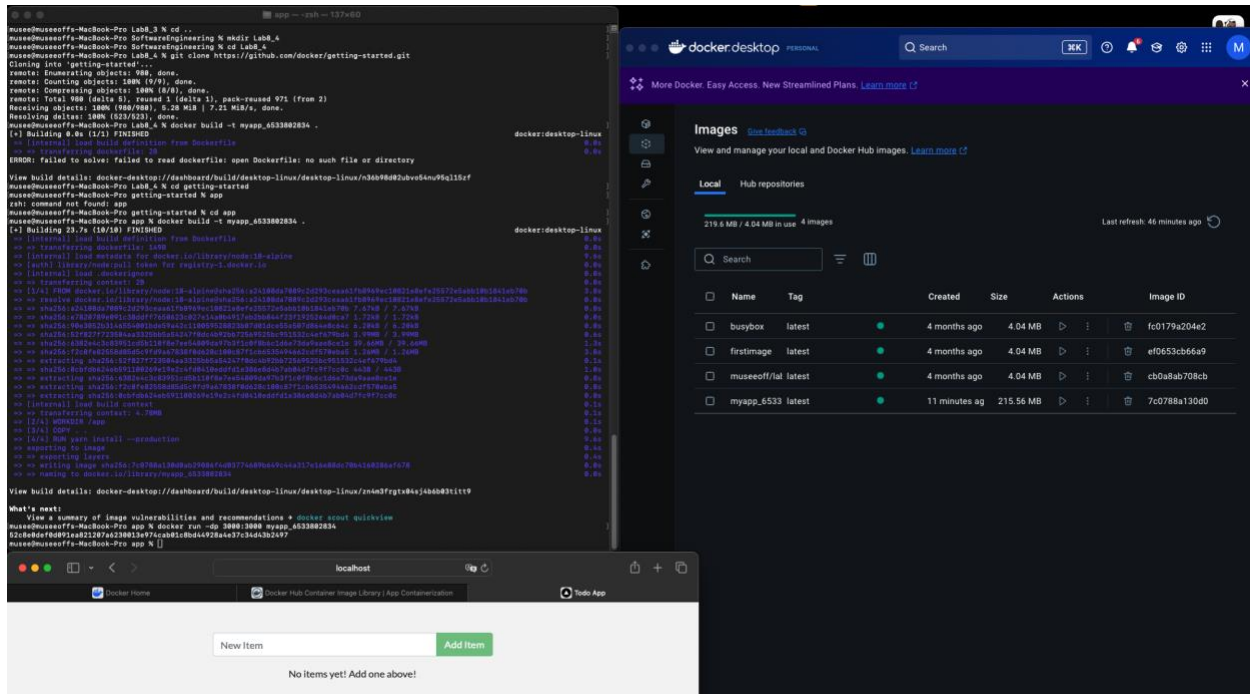
\$ docker run -dp 3000:3000 <myapp_รหัสศ. ไม่มีขีด>

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

The screenshot shows a terminal window at the top with the following error messages:

```

musee@museeoffs-MacBook-Pro app % docker run -dp 3000:3000 myapp_6533802834
96002e61c55b5cfe5490a2fee4c6c4b7119726aea076726f6308c2697f11afb
docker: Error response from daemon: driver failed programming external connectivity on endpoint silly_goldstine (1f58b8865db9efce3a3bd390
d77134149fc0ba27be752d9204b95ad3e8656b86): Bind for 0.0.0.0:3000 failed: port is already allocated.
musee@museeoffs-MacBook-Pro app % docker run -dp 3000:3000 myapp_6533802834
5296c64984a5c44f34b211b2cfff0d2ea568db9cfbfb2e9f2054a37e2e03f4fe2
docker: Error response from daemon: driver failed programming external connectivity on endpoint objective_saha (69c3ee2ddf29cc866bb949ac0
0d561d104f7469349519dff5473ee7080b5179e): Bind for 0.0.0.0:3000 failed: port is already allocated.

```

Below the terminal is the VS Code editor interface. The Explorer sidebar on the left shows a file tree for a project named 'app'. The file 'app.js' is selected. The main editor area shows the code for 'app.js', which is a React component named 'TodoListCard'. The code includes state management for 'items' and functions for adding and removing items. The UI renders a list of items or a message if there are no items.

(1) Error ที่เกิดขึ้นหมายความว่าอย่างไร และเกิดขึ้นเพราะอะไร

Port 3000 ถูกใช้อยู่

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

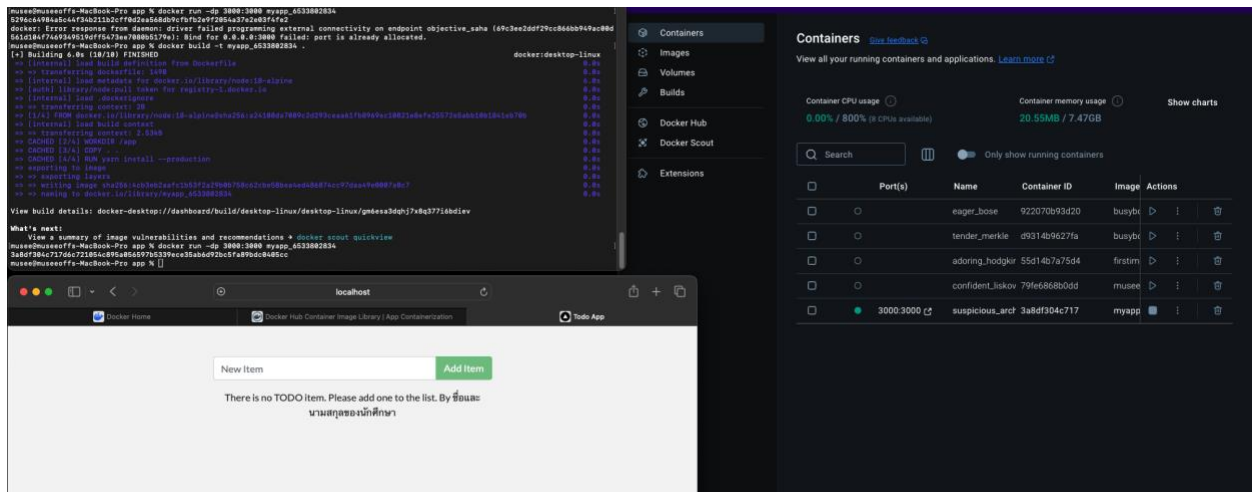
ii. เลือกไอคอนถังขยะในแถวของ Container ที่ต้องการจะลบ

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) แสดงผลลัพธ์ที่ได้บน Browser และ Dashboard ของ Docker desktop



แบบฝึกปฏิบัติที่ 8.5: เริ่มต้นสร้าง Pipeline อย่างง่ายสำหรับการ Deploy ด้วย Jenkins

1. เปิด Command line หรือ Terminal บน Docker Desktop

2. ป้อนคำสั่งและทำการรัน container โดยผูกพอร์ต

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

หรือ

\$ docker run -p 8080:8080 -p 50000:50000 --restart=on-failure -v

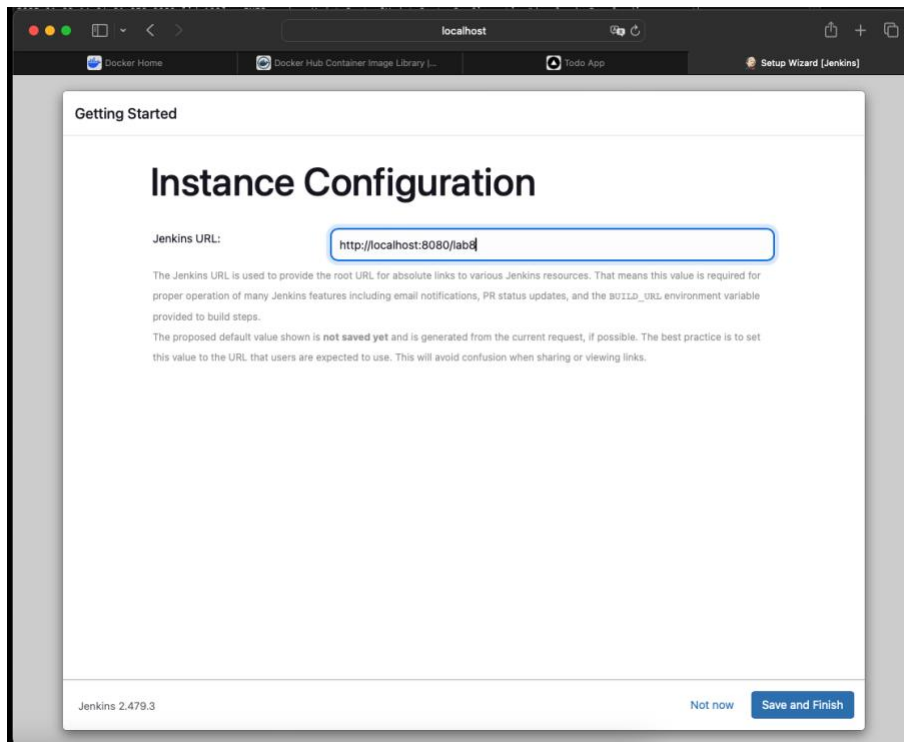
jenkins_home:/var/jenkins_home jenkins/jenkins:lts-jdk17

3. บันทึกรหัสผ่านของ Admin user ไว้สำหรับ log-in ในครั้งแรก

[illegible]

- 13

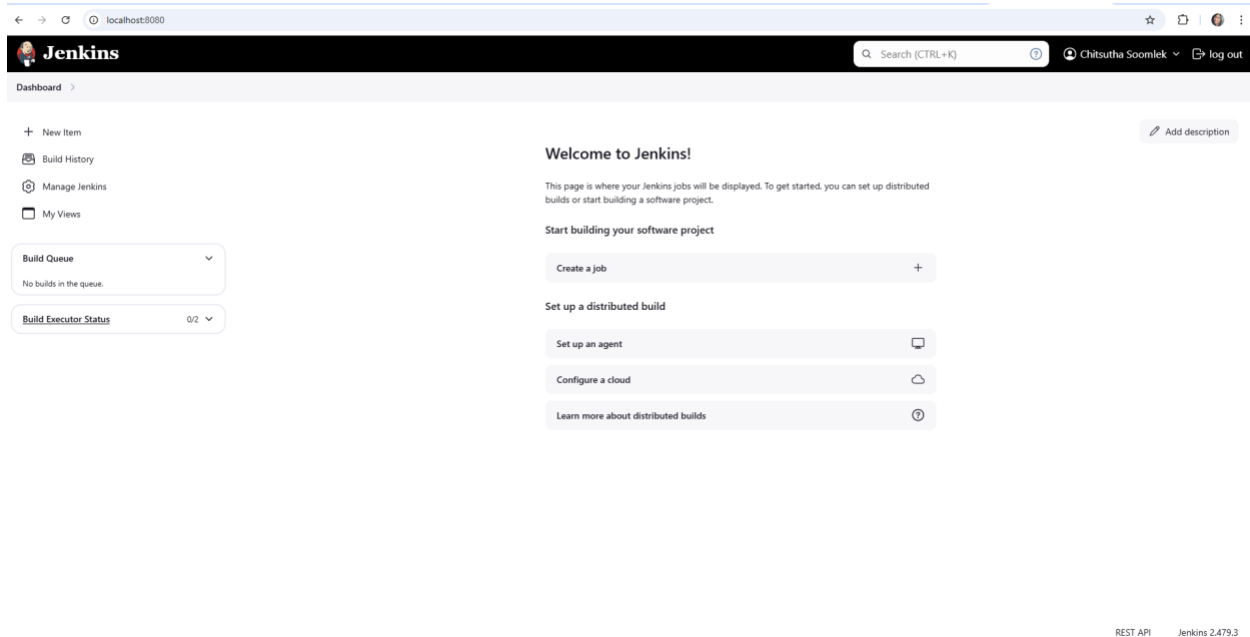
Lab Worksheet



(capture ไม่ทันครับ กด Save & Finish มาก่อน TT)

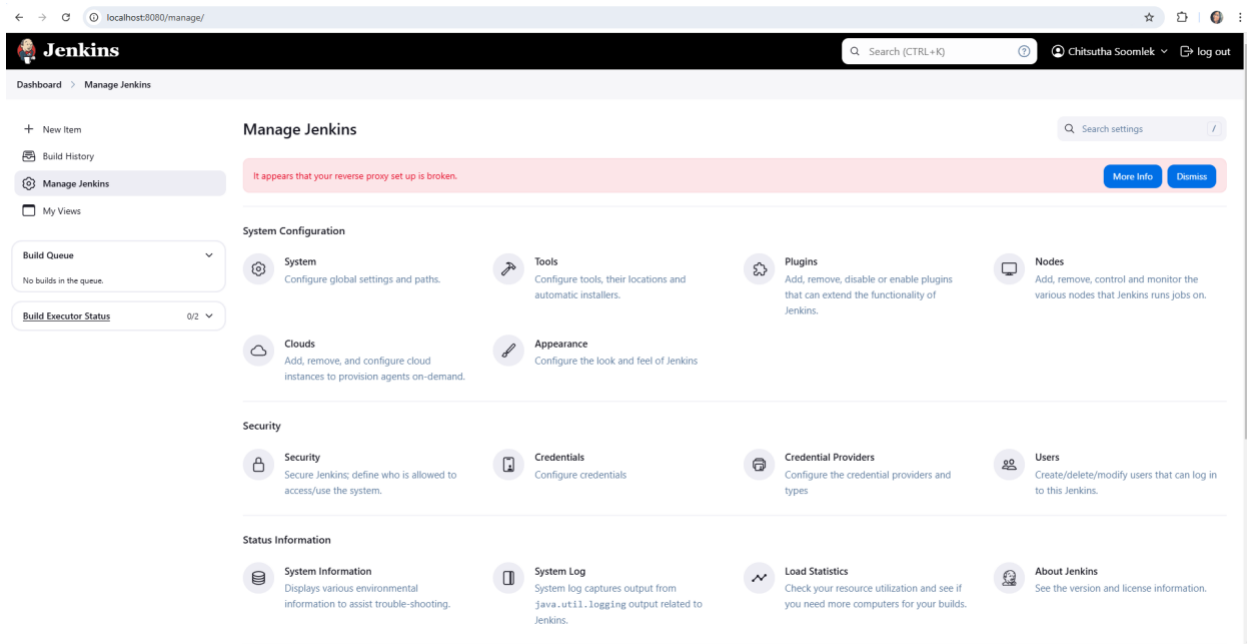
7. กำหนด Jenkins URL เป็น <http://localhost:8080/lab8>

8. เมื่อติดตั้งเรียบร้อยแล้วจะพบกันหน้า Dashboard ดังแสดงในภาพ



Lab Worksheet

9. เลือก Manage Jenkins แล้วไปที่เมนู Plugins

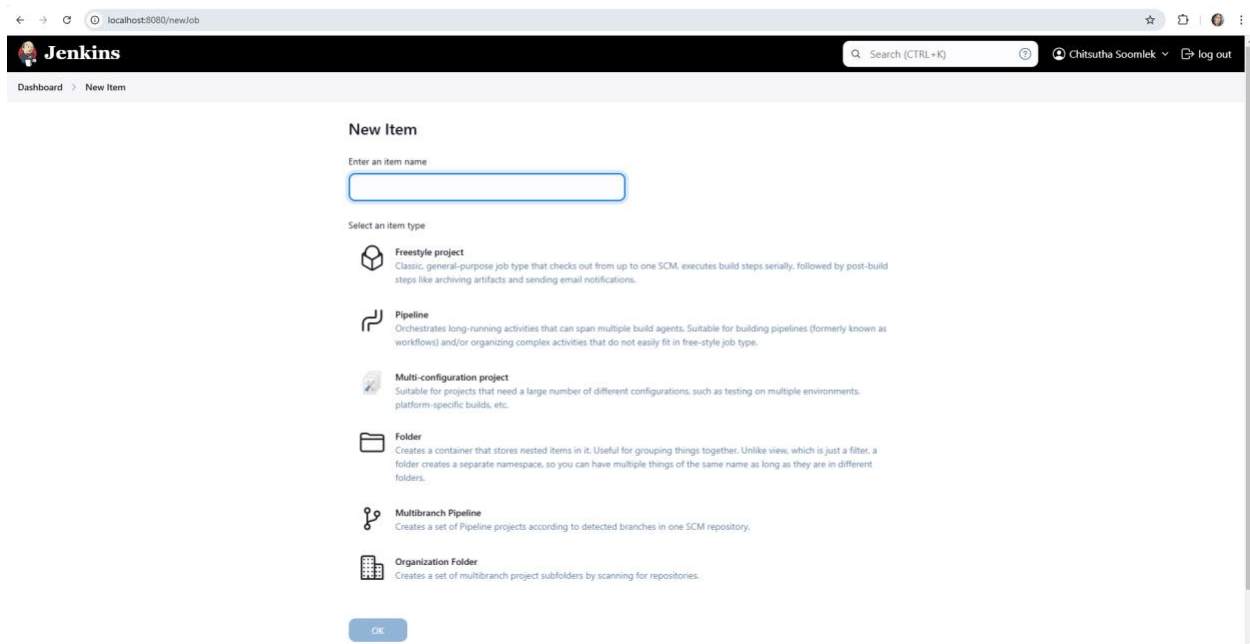


10. ไปที่เมนู Available plugins แล้วเลือกติดตั้ง Robotframework เพิ่มเติม



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

Lab Worksheet



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

Description: Lab 8.5


GitHub project: กดเลือก แล้วใส่ Project URL เป็น repository ที่เก็บโค้ด .robot (ดูขั้นตอนที่ 12)

Build Trigger: เลือกแบบ Build periodically แล้วกำหนดให้ build ทุก 15 นาที

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

[Check point#14] Capture หน้าจอแสดงการตั้งค่า พร้อมกับตอบคำถามต่อไปนี้

Lab Worksheet

General Enabled 

Description


Lab 8.5

Plain text [Preview](#)

☐ Discard old builds [?](#)

☒ GitHub project
Project url [?](#)


https://github.com/Museey/Lab7-Test-Automation

Advanced 

☐ This project is parameterised [?](#)

☐ Throttle builds [?](#)

☐ Execute concurrent builds if necessary [?](#)

Advanced 

Source Code Management

☒ None

☐ Git [?](#)

Build Triggers

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

☐ Build after other projects are built [?](#)

☒ Build periodically [?](#)
Schedule [?](#)

H/15 * * * *

Lab Worksheet

☒ Build periodically ?

Schedule ?

H/15 * * * *

Would last have run at Wednesday, January 22, 2025 at 2:20:20 PM Coordinated Universal Time; would next run at Wednesday, January 22, 2025 at 2:35:20 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](#)

robot Lab7-002.robot

Advanced ▾

Add build step ▾

Save Apply

(1) คำสั่งที่ใช้ในการ Execute ไฟล์ .robot ใน Build Steps คือ robot Lab7-002.robot

Post-build action: เพิ่ม Publish Robot Framework test results ->

ระบุไดเรกทอรีที่เก็บไฟล์ผลการทดสอบโดย Robot framework ในรูป xml และ html -> ตั้งค่า Threshold เป็น

Lab Worksheet

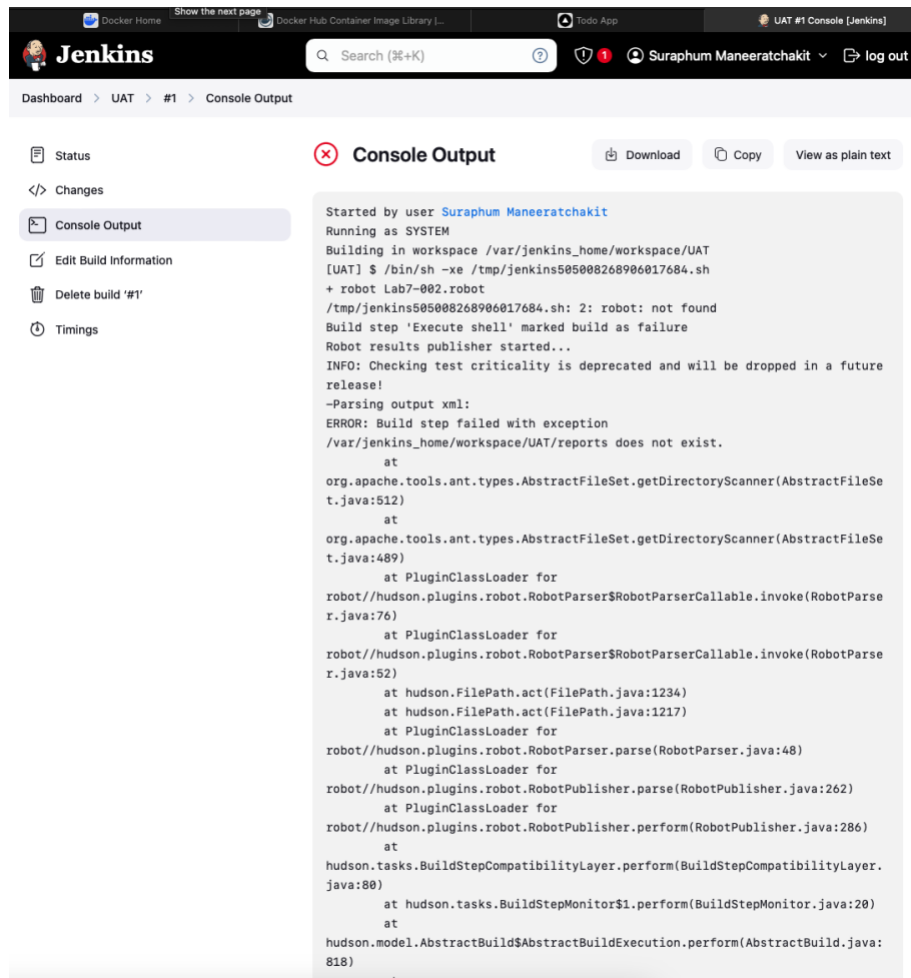
% ของการทดสอบที่ไม่ผ่านแล้วนับว่าซอฟต์แวร์มีปัญหา -> ตั้งค่า Threshold เป็น %

ของการทดสอบที่ผ่านแล้วนับว่าซอฟต์แวร์มีอยู่ในสถานะที่สามารถนำไปใช้งานได้ (เช่น 20, 80)

13. กด Apply และ Save

14. สั่ง Build Now

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



Lab Worksheet

The screenshot displays the Jenkins Dashboard for user Suraphum Maneeratchakit. The interface includes a top navigation bar with links to Docker Home, Docker Hub Container Image Library, and Todo App. The main content area shows the 'Build Queue' and 'Build Executor Status' sections. The 'Build Queue' section indicates 'No builds in the queue.' The 'Build Executor Status' section shows 0/2 executors. A table of build history is visible, with one build named 'UAT' in progress, showing a last success of 'N/A', a last failure of '52 sec #1', and a last duration of '11 ms'. The table also includes columns for 'S', 'W', 'Name', 'Last Success', 'Last Failure', 'Last Duration', and 'Robot Results + Duration'.

S	W	Name	Last Success	Last Failure	Last Duration	Robot Results + Duration
		UAT	N/A	52 sec #1	11 ms	