Lab Worksheet

ชื่อ-นามสกุล_____พัชริดา เฟื่องอารมย์_____รหัสนักศึกษา_____663380608-3_____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

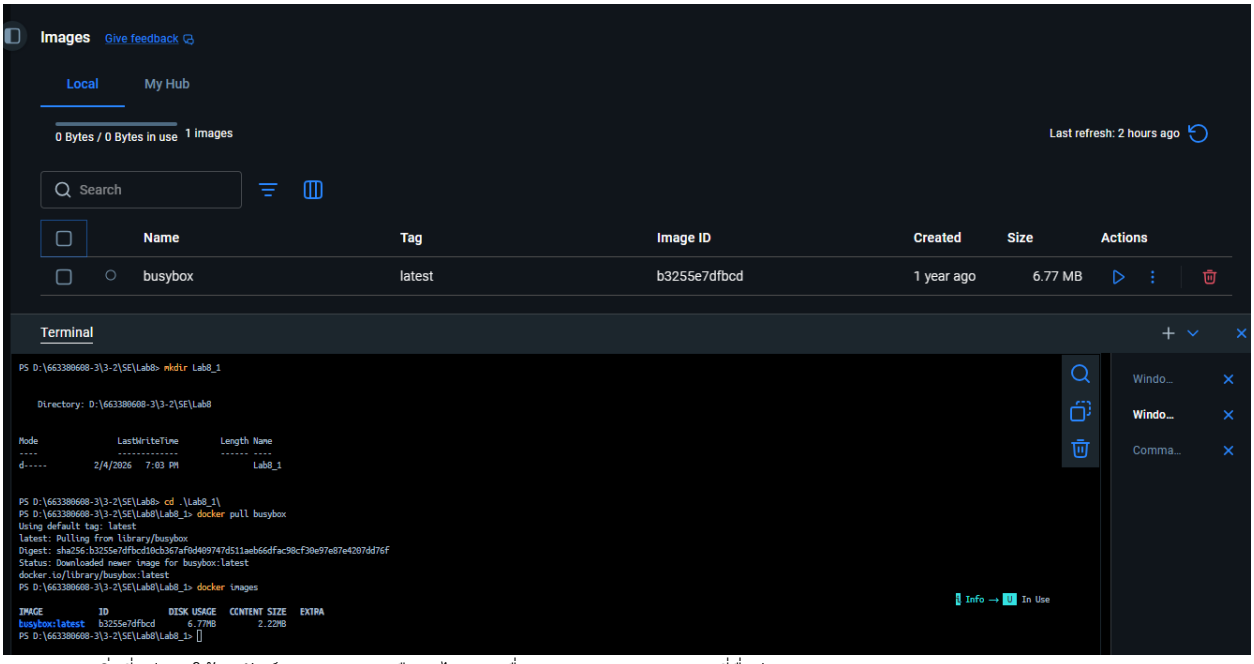
Pre-requisite

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
2. เปิด Command line หรือ Terminal บน Docker Desktop จากนั้นสร้าง Directory ชื่อ Lab8_1
3. ย้ายตำแหน่งปัจจุบันไปที่ Lab8_1 เพื่อใช้เป็น Working directory
4. ป้อนคำสั่ง $ docker pull busybox หรือ $ sudo docker pull busybox สำหรับกรณีที่ติดปัญหา Permission denied (หมายเหตุ: BusyBox เป็น software suite ที่รองรับคำสั่งบางอย่างบน Unix - https://busybox.net)
5. ป้อนคำสั่ง $ docker images

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

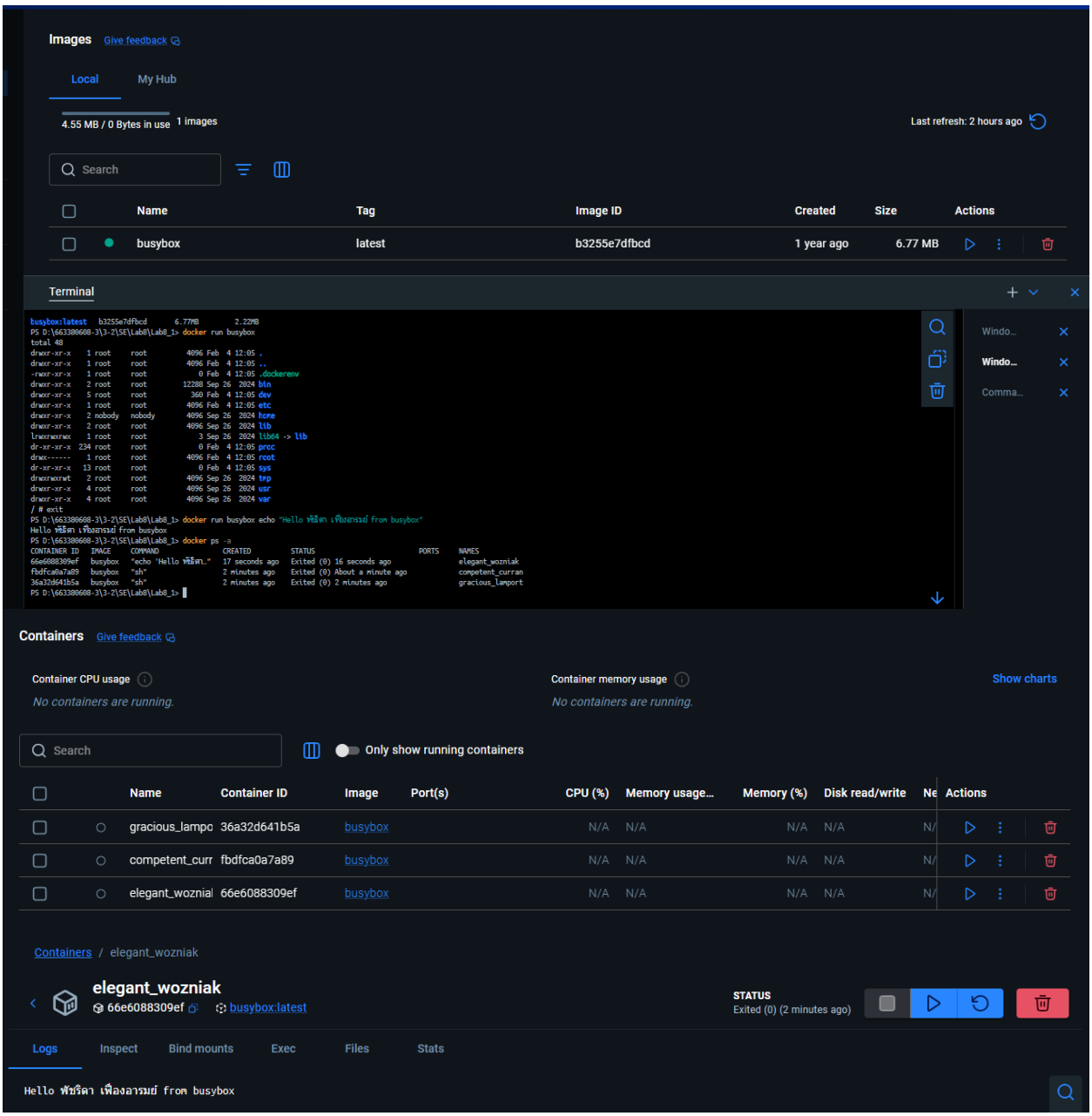


(1) สิ่งที่อยู่ภายใต้คอลัมน์ Repository คืออะไร____ชื่อของ Docker images ที่ชื่อว่า busybox_____

(2) Tag ที่ใช้บ่งบอกถึงอะไร___บอกถึง Version ของ Docker image ซึ่งจากผลลัพธ์ busybox:latest ในที่นี้คือ lastest = Version ล่าสุด_

6. ป้อนคำสั่ง $ docker run busybox
7. ป้อนคำสั่ง $ docker run -it busybox sh
8. ป้อนคำสั่ง ls
9. ป้อนคำสั่ง ls -la
10. ป้อนคำสั่ง exit
11. ป้อนคำสั่ง $ docker run busybox echo “Hello ชื่อและนามสกุลของนักศึกษา from busybox”
12. ป้อนคำสั่ง $ docker ps -a

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



(1) เมื่อใช้ option -it ในคำสั่ง run ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป

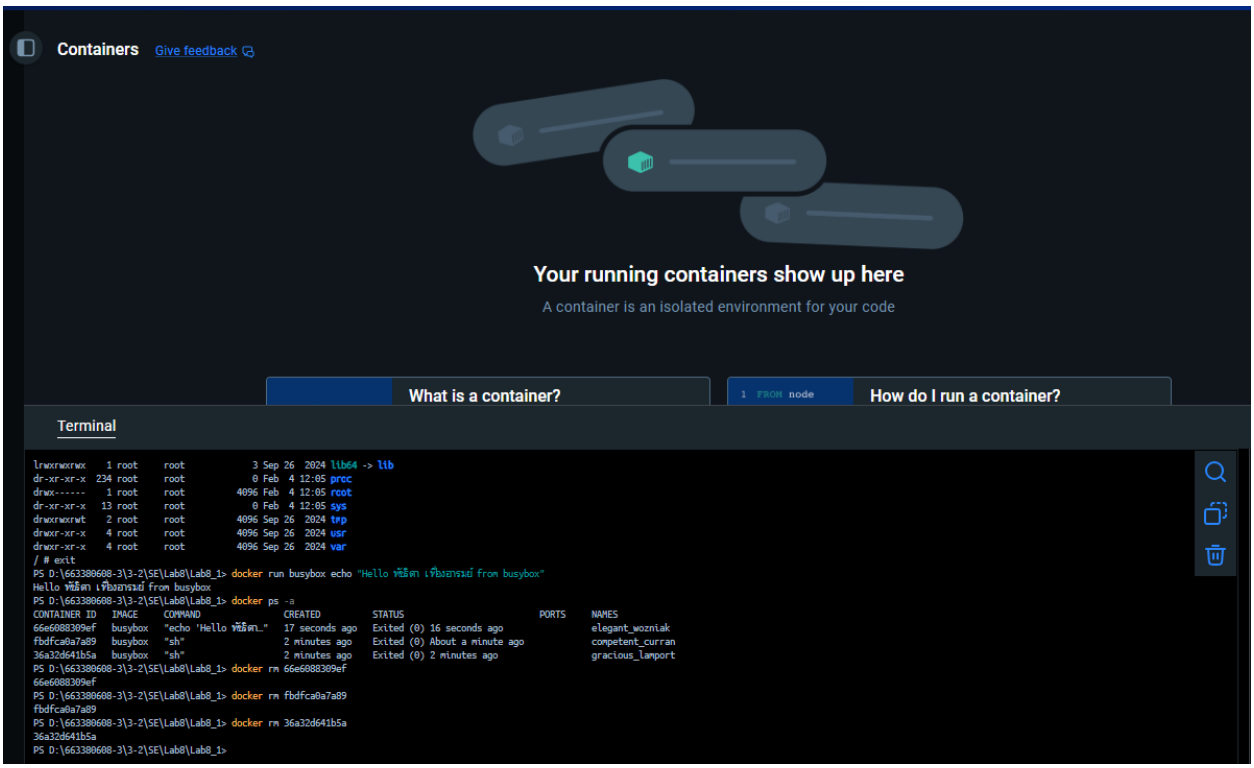
-it จะทำให้สามารถพิมพ์คำสั่งและตอบโต้กันได้แบบ real time มาจาก -i (interactive) ที่ทำให้ contrainer เปิดตลอดเวลา ทำให้สามารถส่ง input เข้าไปได้ และ -t (–tty) สร้าง terminal จำลองมาเชื่อมต่อกับ input/output ของcontrainer กับ terminal สามารถพิมคำสั่งและเห็นผลลัพธ์ได้ทันที

(2) คอลัมน์ STATUS จากการรันคำสั่ง docker ps -a แสดงถึงข้อมูลอะไร

_____สถานะการทำงานของ contrainer ในระบบ โดยผลลัพธ์คอลัมน์ STATUS ที่ได้แสดงว่า Exited __

13. 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
FROM busybox
CMD echo “Hi there. This is my first docker image.”
CMD echo “ชื่อ-นามสกุล รหัสนักศึกษา ชื่อเล่น”
EOF
```

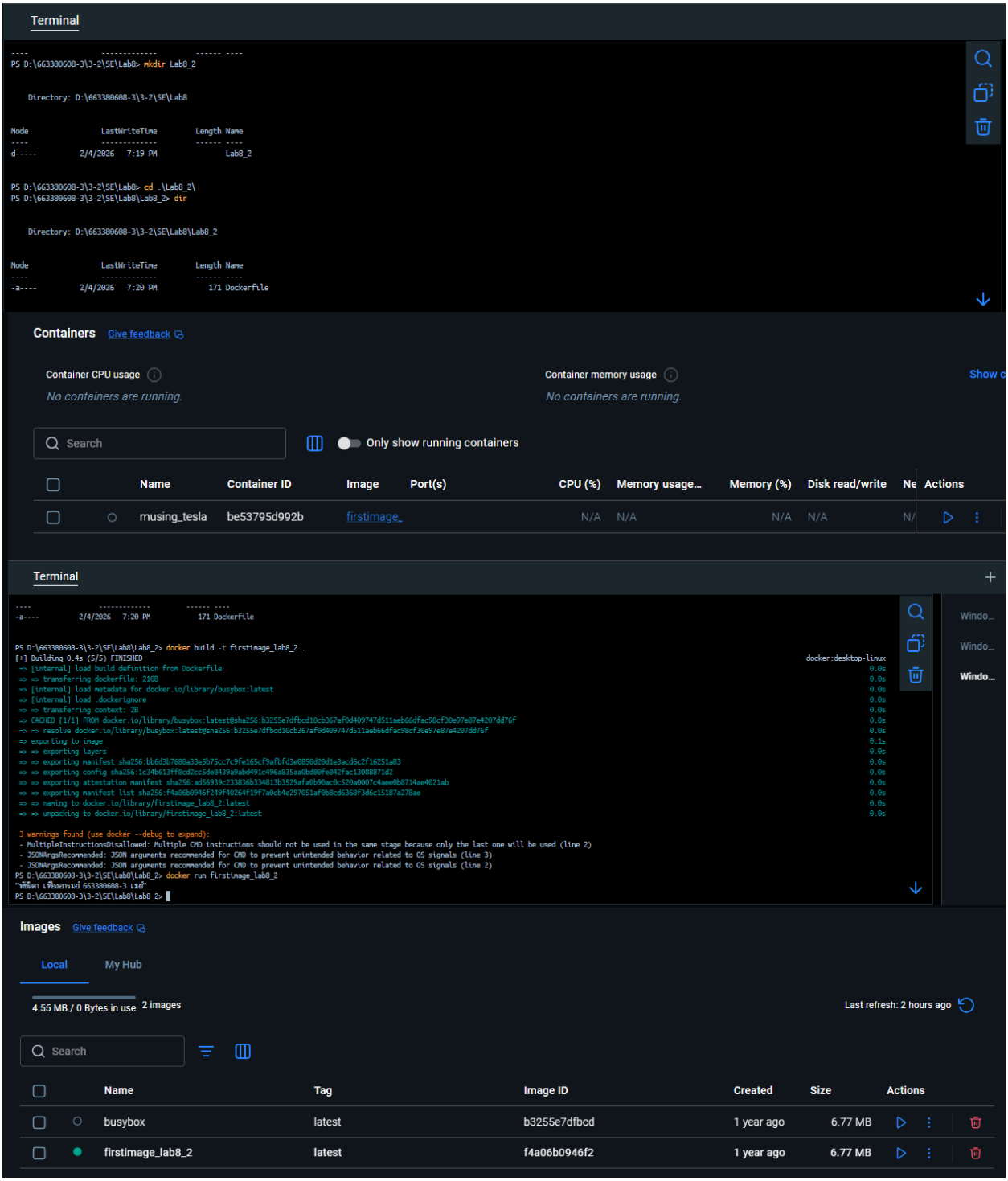
หรือใช้คำสั่ง

```
$ 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 <ชื่อ image>____ ในที่นี้ใช้คำสั่ง docker run firstimage_lab8_2_____

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

_____-t มาจากคำว่า Tag ใช้กำหนดชื่อหรือเวอร์ชั่นให้กับ Docker ที่เราสร้างขึ้น_____

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
```

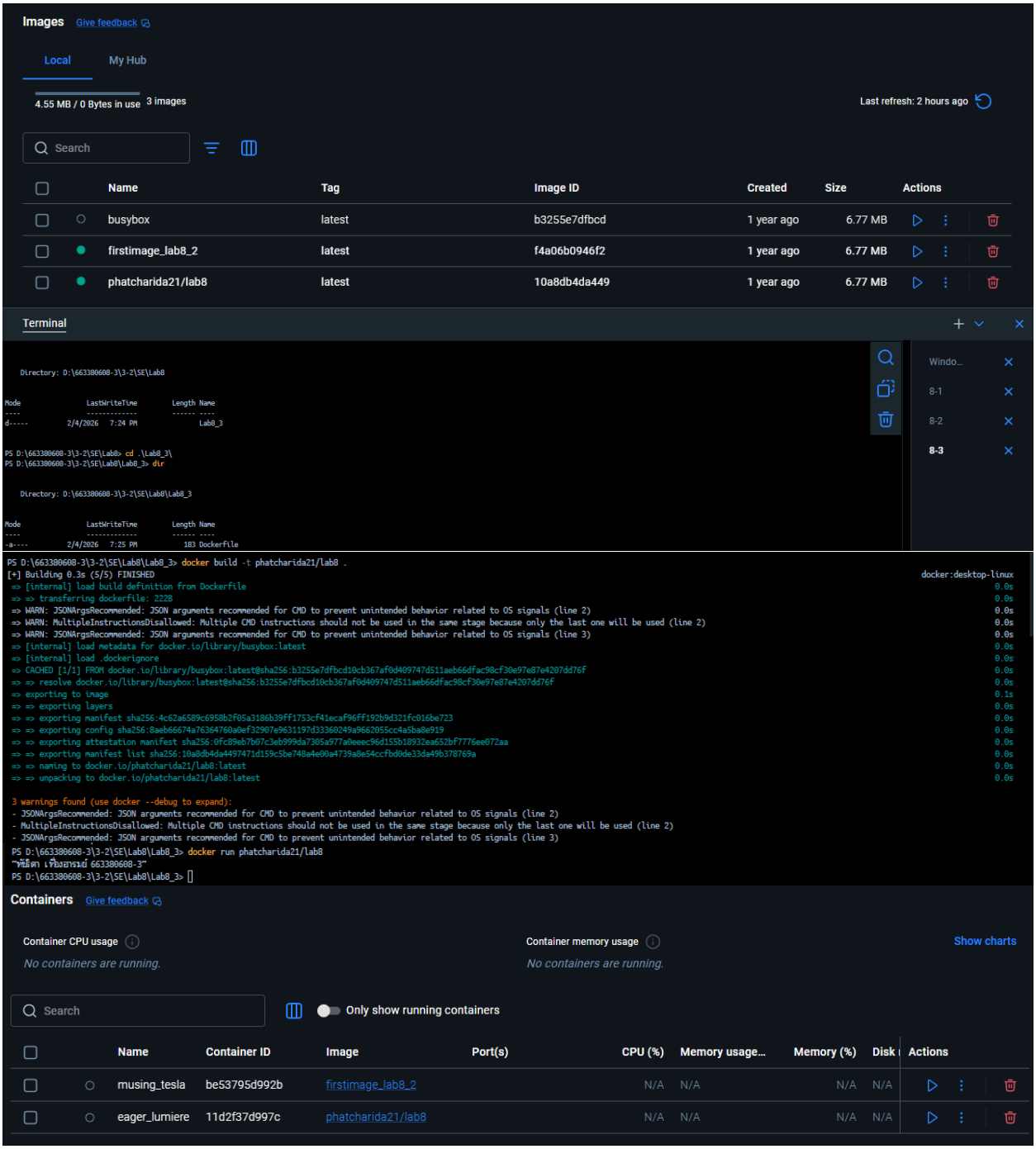
แล้วใช้ Text Editor ในการใส่เนื้อหาแทน

5. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้

   $ docker build -t <username ที่ลงทะเบียนกับ Docker Hub>/lab8 .
6. ทำการรัน Docker image บน Container ในเครื่องของตัวเองเพื่อทดสอบผลลัพธ์ ด้วยคำสั่ง

   $ docker run <username ที่ลงทะเบียนกับ Docker Hub>/lab8

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



7. ทำการ Push ตัว Docker image ไปไว้บน Docker Hub โดยการใช้คำสั่ง

   $ docker push <username ที่ลงทะเบียนกับ Docker Hub>/lab8

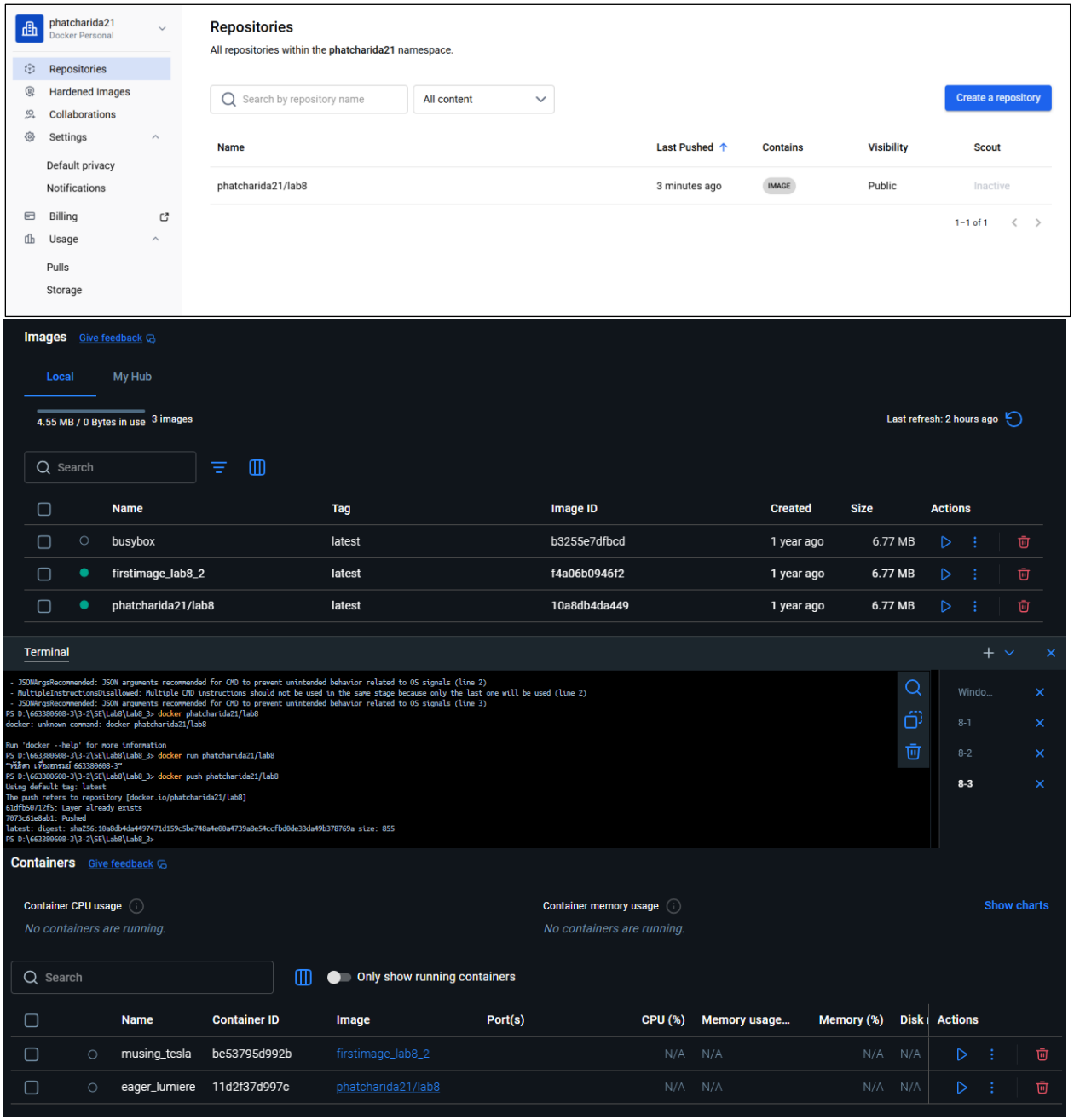
   ในกรณีที่ติดปัญหาไม่ได้ Login ไว้ก่อน ให้ใช้คำสั่งต่อไปนี้ เพื่อ Login ก่อนทำการ Push

   $ docker login แล้วป้อน Username และ Password ตามที่ระบุใน Command prompt หรือใช้คำสั่ง

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

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

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



แบบฝึกปฏิบัติที่ 8.4: การ Build แอปพลิเคชันจาก Container image และการ Update แอปพลิเคชัน

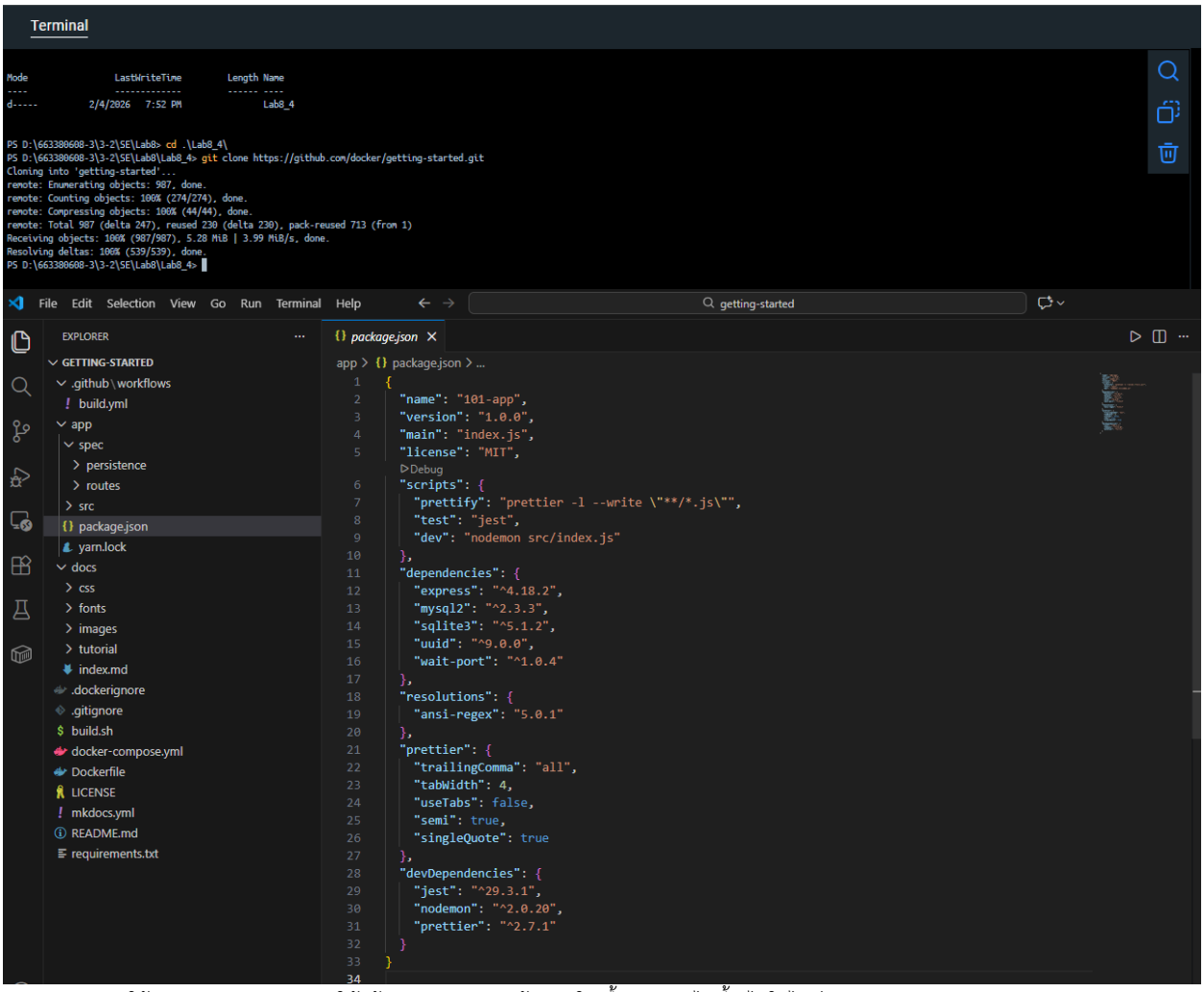
1. เปิด Command line หรือ Terminal จากนั้นสร้าง Directory ชื่อ Lab8_4

2. ทำการ Clone ซอร์สโค้ดของเว็บแอปพลิเคชันจาก GitHub repository https://github.com/docker/getting-started.git ลงใน Directory ที่สร้างขึ้น โดยใช้คำสั่ง

   $ 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



4. ภายใต้ getting-started/app ให้สร้าง Dockerfile พร้อมกับใส่เนื้อหาดังต่อไปนี้ลงไปในไฟล์

   FROM node:18-alpine

   WORKDIR /app

   COPY . .

   RUN yarn install --production

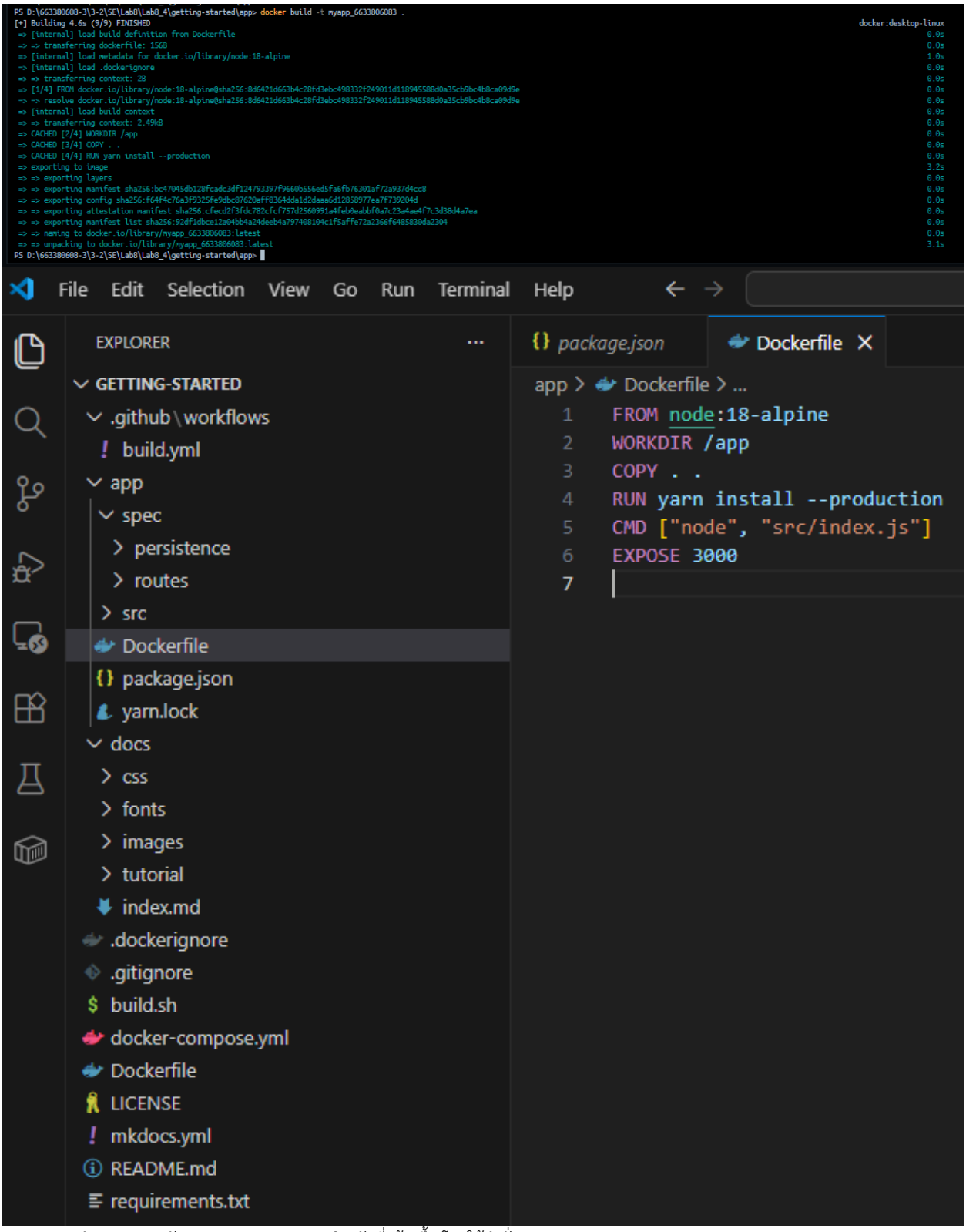
   CMD ["node", "src/index.js"]

   EXPOSE 3000

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

   $ docker build -t <myapp_รหัสนศ. ไม่มีขีด> .

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

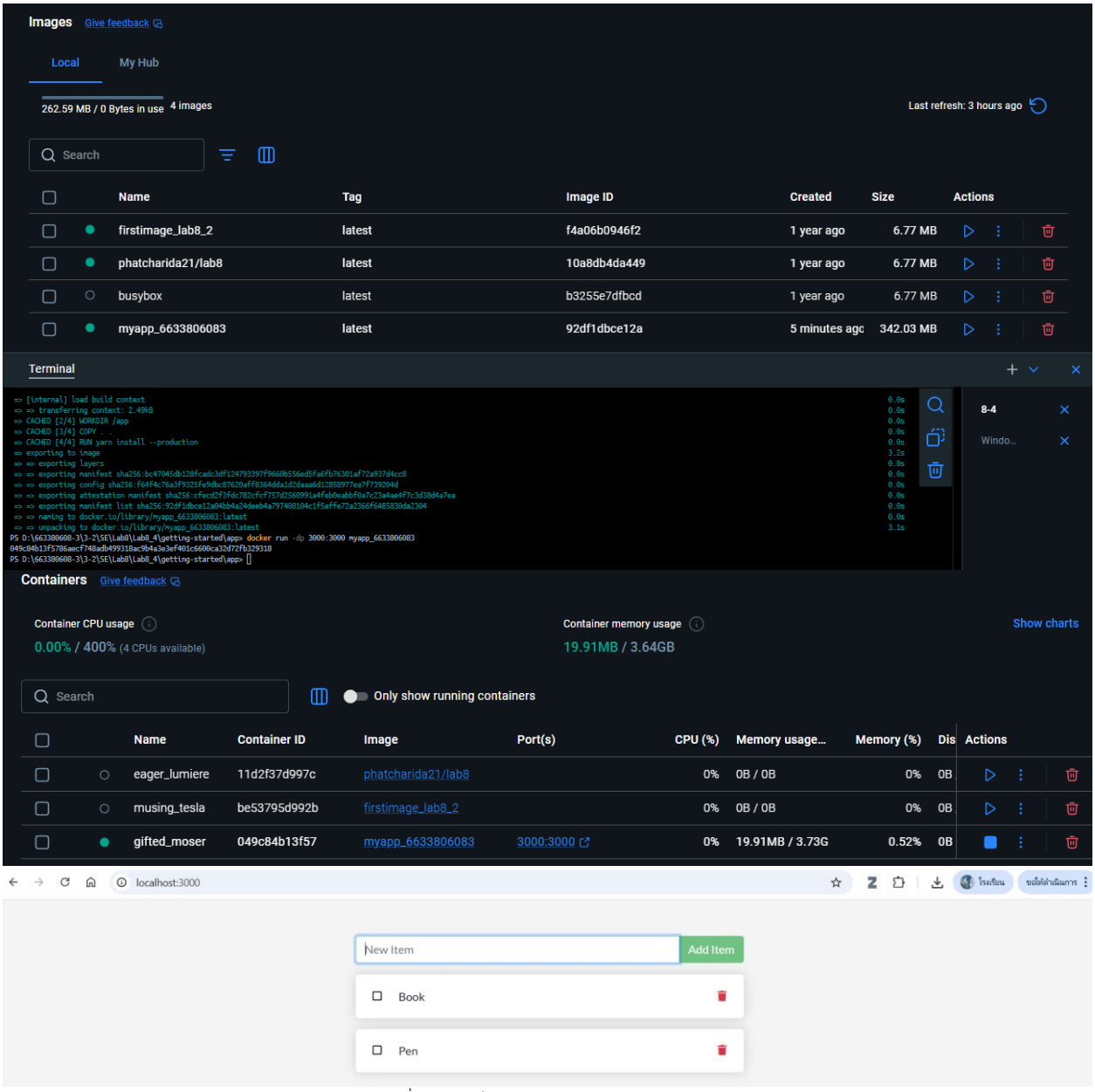


6. ทำการ Start ตัว Container ของแอปพลิเคชันที่สร้างขึ้น โดยใช้คำสั่ง

$ docker run -dp 3000:3000 <myapp_รหัสนศ. ไม่มีขีด>

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

8. ทำการแก้ไข Source code ของ Web application ดังนี้

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

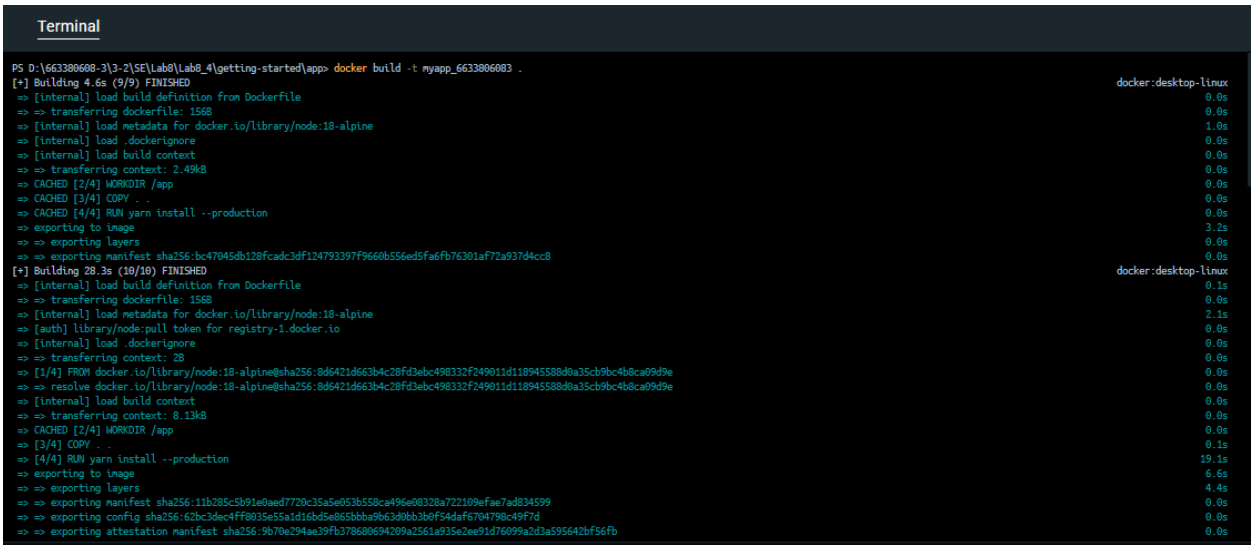
   No items yet! Add one above! เป็น

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

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



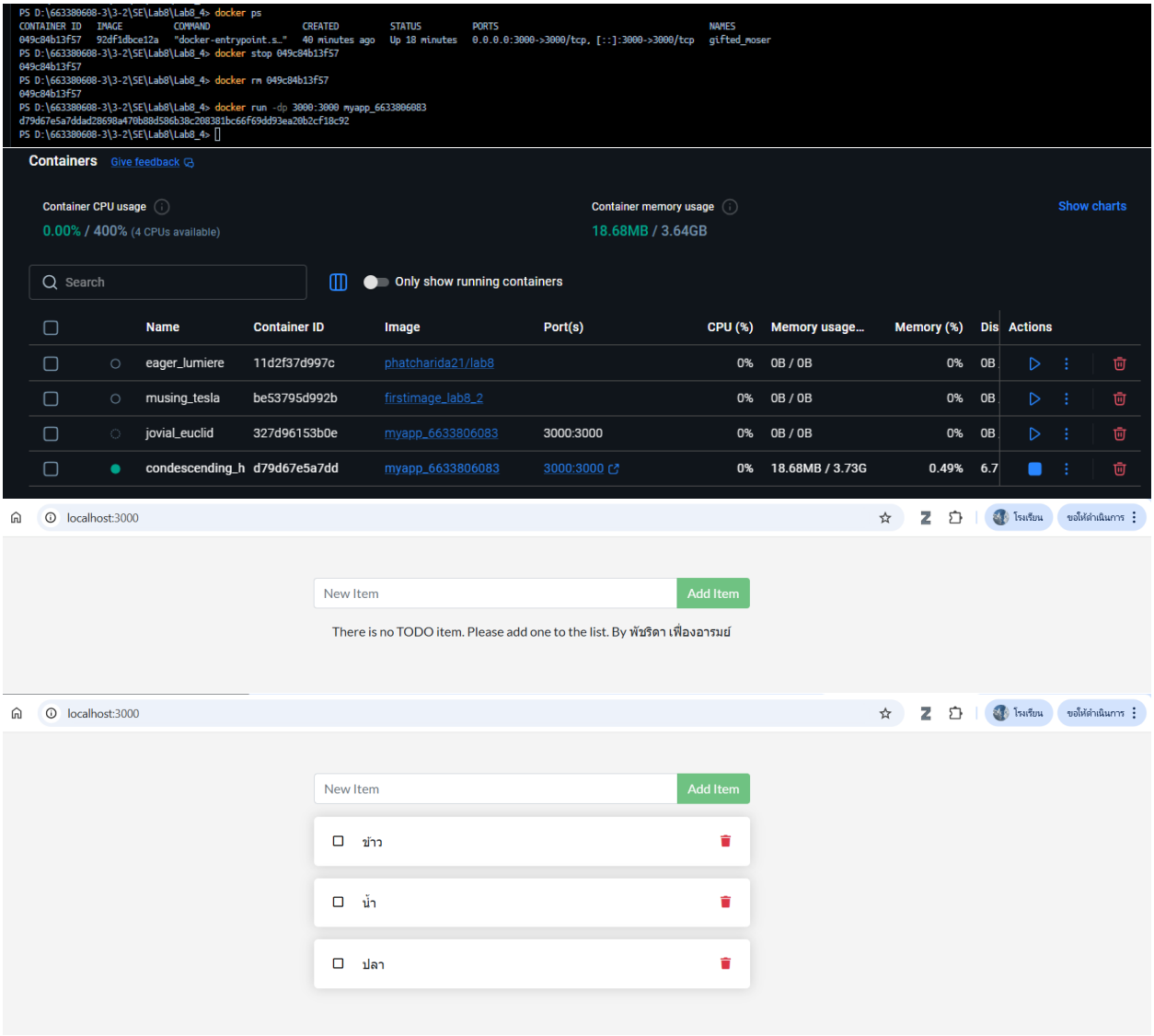
(1) Error ที่เกิดขึ้นหมายความอย่างไร และเกิดขึ้นเพราะอะไร

__พอร์ต 3000 กำลังถูกใช้งานอยู่ เกิดจากการใช้ พอร์ต 3000 โดยยังไม่ได้หยุดการทำงานโปรแกรมที่เคยใช้ก่อนหน้า _____

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

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



แบบฝึกปฏิบัติที่ 8.5: เริ่มต้นสร้าง Pipeline อย่างง่ายสำหรับการ Deploy ด้วย Jenkins

1. สร้าง Dockerfile เพื่อสร้าง Jenkins และ Environment ที่เหมาะสมกับการรัน Robot framework ใน Container

[Check point#12] ส่ง Dockerfile ที่ใส่คำสั่งที่เกี่ยวข้องไว้

```
FROM jenkins/jenkins:lts
USER root
RUN apt-get update && apt-get install -y python3 python3-pip python3-venv
RUN pip3 install robotframework --break-system-packages
USER jenkins
```

2. เปิด Command line หรือ Terminal บน Docker Desktop
3. Build image ที่สร้างขึ้นในข้อที่ 1 พร้อมกับตั้งชื่อของ image เป็น jenkins-robot-local

   $ docker build -t jenkins-robot-local .
4. รัน container โดยผูกพอร์ตให้เรียบร้อย เช่น

   $ docker run -d \

   --name jenkins-robot \

   -p 8080:8080 -p 50000:50000 \

   -v jenkins_home:/var/jenkins_home \

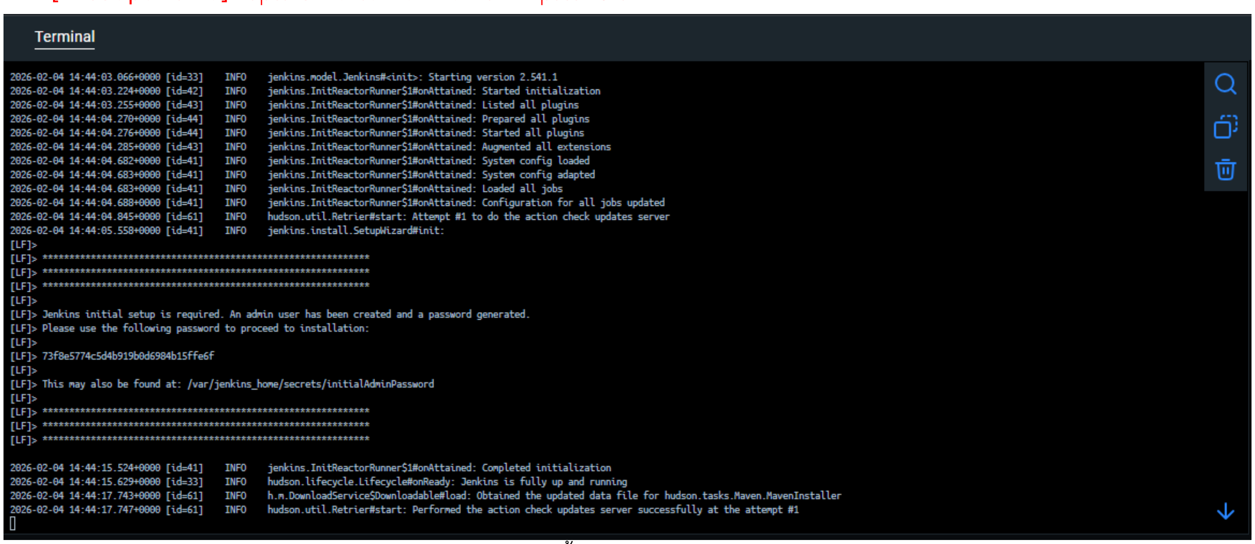
   -v /var/run/docker.sock:/var/run/docker.sock \

   jenkins-robot-local

   หรือ

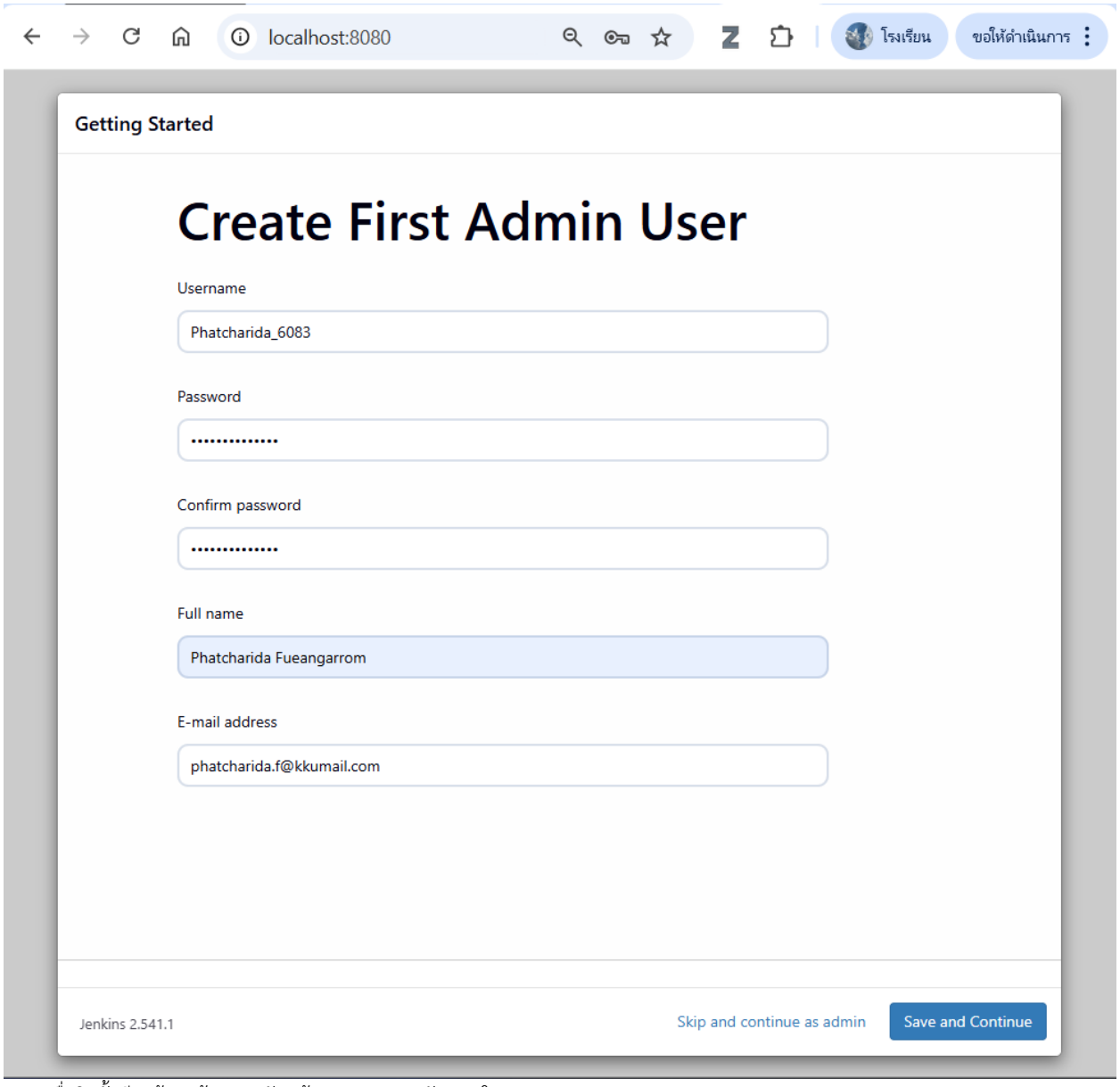
   $ docker run -it --name jenkins-debug -p 8080:8080 -v /var/run/docker.sock:/var/run/docker.sock jenkins-robot-local

[Check point#13] Capture หน้าจอที่แสดงผล Admin password

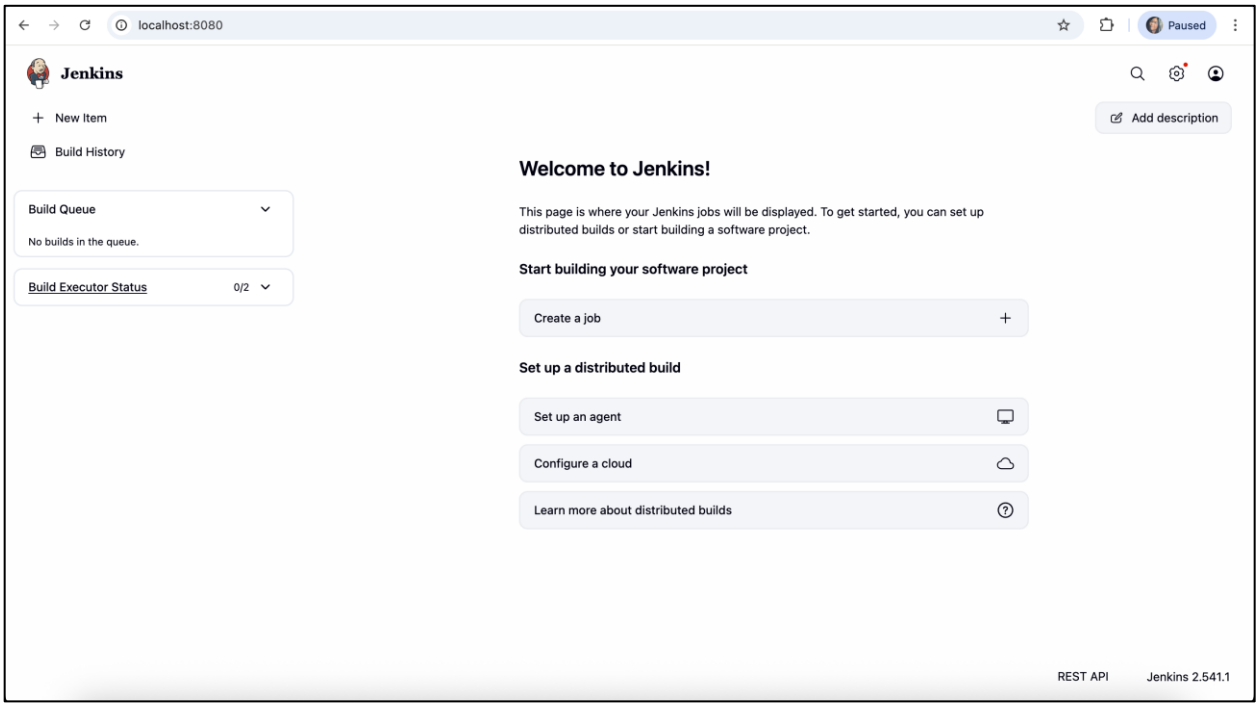


5. บันทึกรหัสผ่านของ Admin user ไว้สำหรับ log-in ในครั้งแรก
6. เมื่อได้รับการยืนยันว่า Jenkins is fully up and running ให้เปิดบราวเซอร์ และป้อนที่อยู่เป็น http://localhost:8080
7. ทำการ Unlock Jenkins ด้วยรหัสผ่านที่ได้ในข้อที่ 3
8. สร้าง Admin User โดยใช้ username เป็นชื่อจริงของนักศึกษาพร้อมรหัสสี่ตัวท้าย เช่น somsri_3062

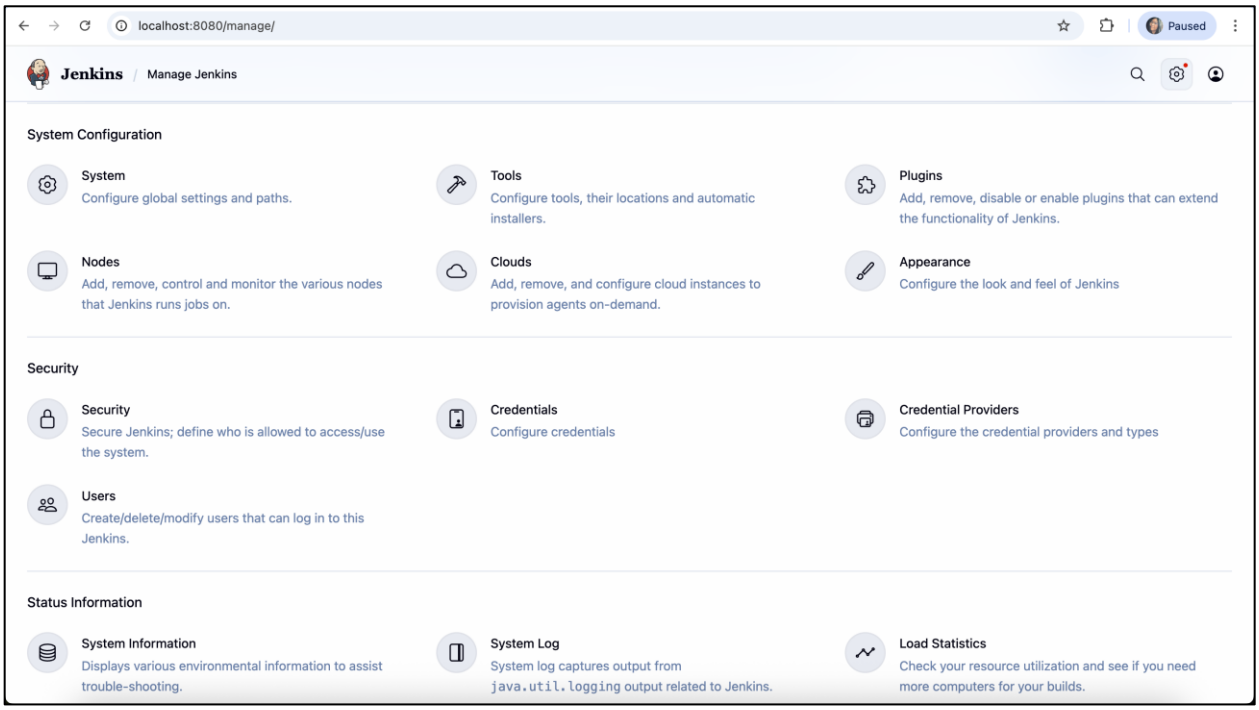
[Check point#14] Capture หน้าจอที่แสดงผลการตั้งค่า



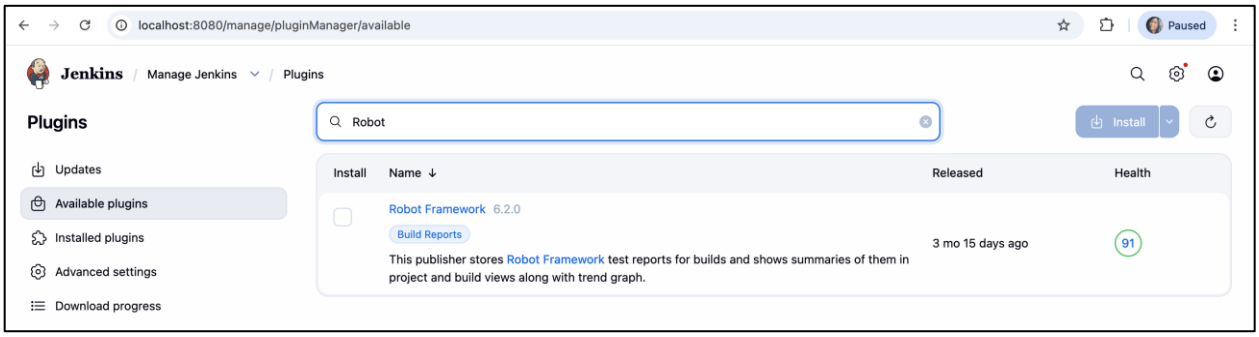
9. เมื่อติดตั้งเรียบร้อยแล้วจะพบกันหน้า Dashboard ดังแสดงในภาพ



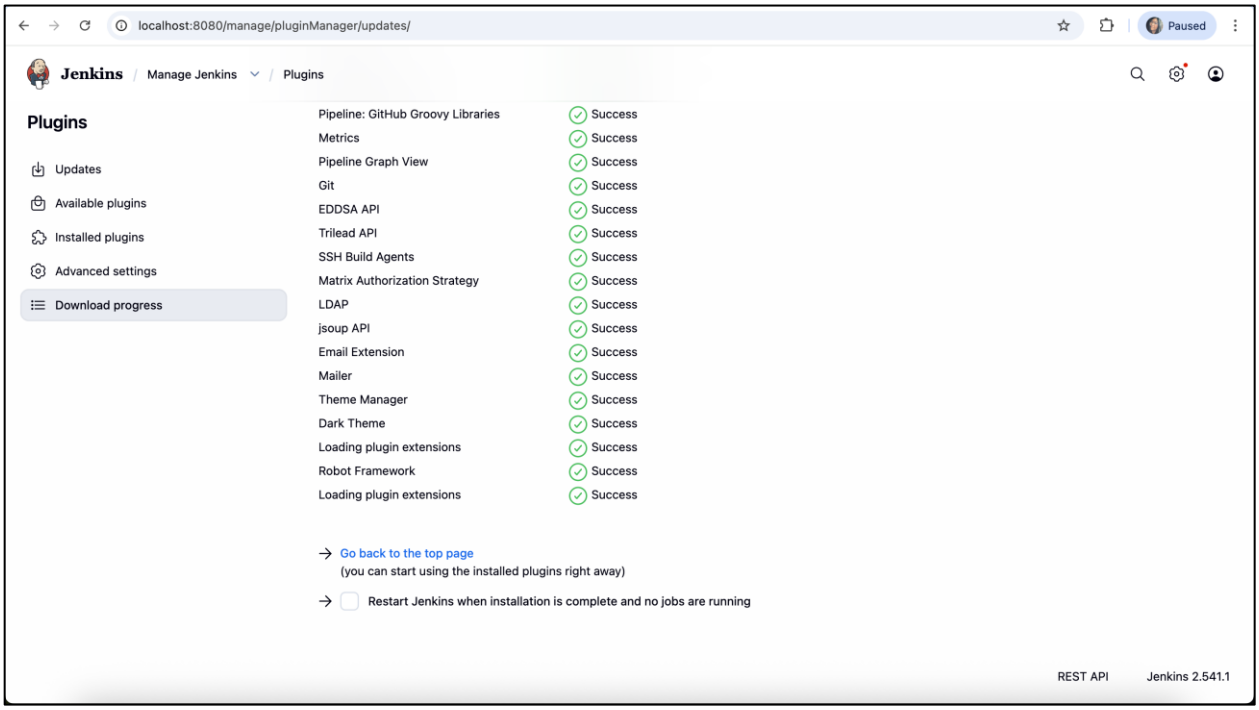
10. เลือก Manage Jenkins แล้วไปที่เมนู Plugins



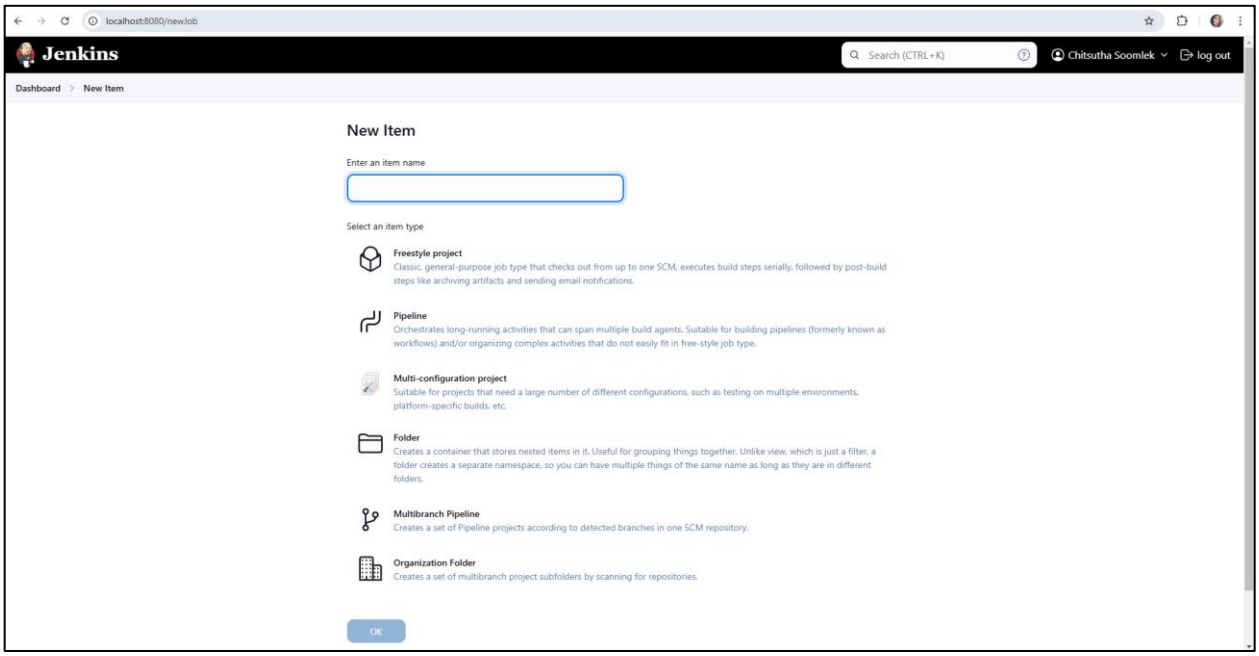
11. ไปที่เมนู Plugins > Available plugins แล้วเลือกติดตั้ง Robot Framework เพิ่มเติม



12. เมื่อติดตั้งสำเร็จจะพบกับรายการ Plugins ทั้งหมดที่ถูกติดตั้ง ถ้าติดตั้งสำเร็จให้เลือก “Restart Jenkins...” แล้วกด Go back to the top page



13. สร้างไฟล์ Jenkinsfile ไม่มีนามสกุล เพื่อ execute คำสั่งต่าง ๆ กับ built-in agent แล้วเอาไฟล์ดังกล่าว เก็บไว้ที่ root ของ GitHub Repository ของนักศึกษา
14. สร้าง folder ชื่อ tests/ บน GitHub Repository ของนักศึกษา และสร้างไฟล์ Lab8.robot แล้วนำไฟล์ไปไว้ folder ที่สร้าง
15. กลับไปที่หน้า Dashboard ของ Jenkins แล้วสร้าง Pipeline อย่างง่าย โดยกำหนด New item เป็น Freestyle project และตั้งชื่อเป็น UAT



16. ตั้งค่าที่จำเป็นในหน้านี้ทั้งหมด ดังนี้

Description: Lab 8.5

GitHub project: กดเลือก แล้วใส่ Project URL เป็น repository ที่เก็บโค้ด .robot (ดูขั้นตอนที่ 14)

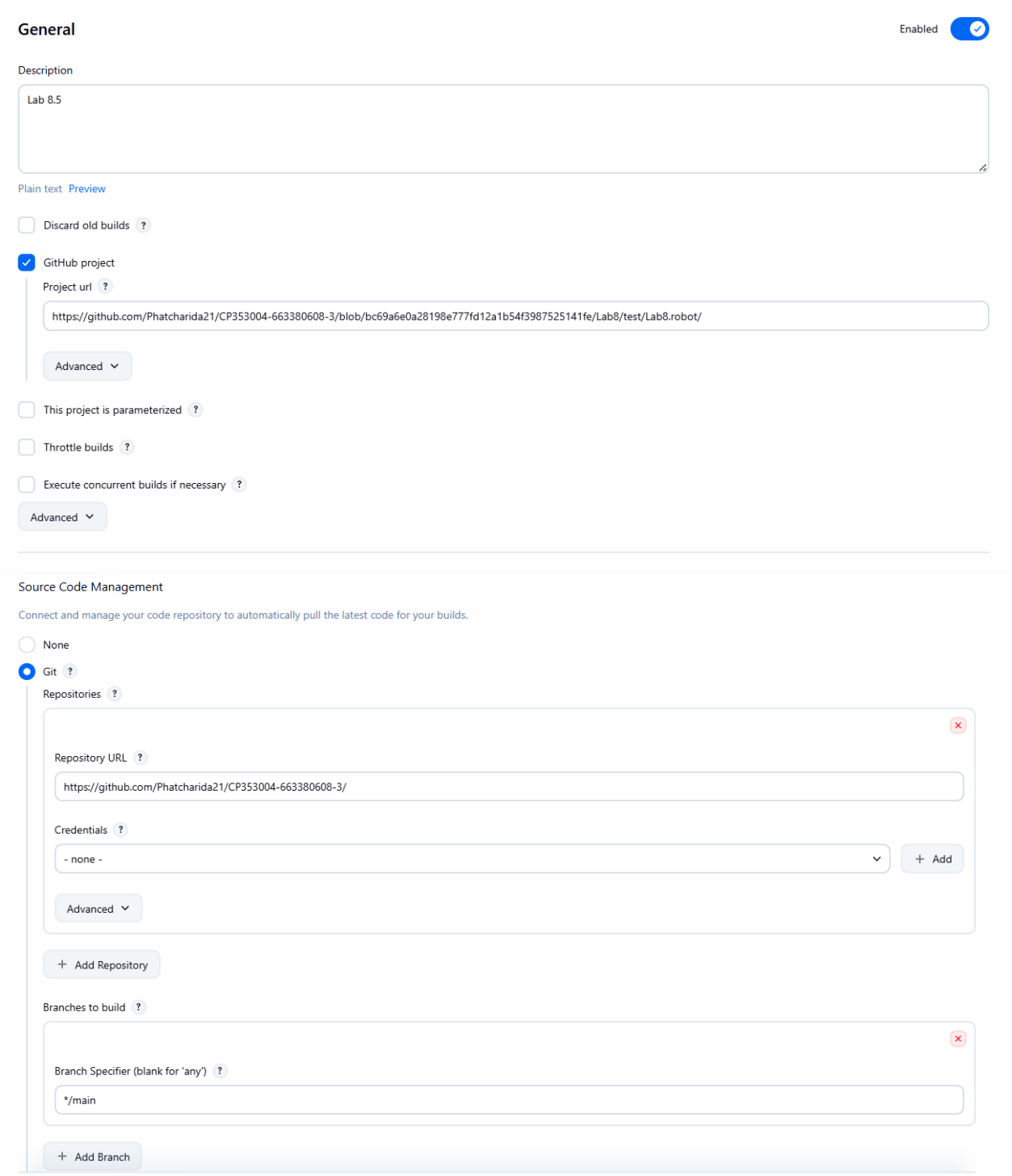
Build Trigger: เลือกแบบ Build periodically แล้วกำหนดให้ build ทุก 15 นาที

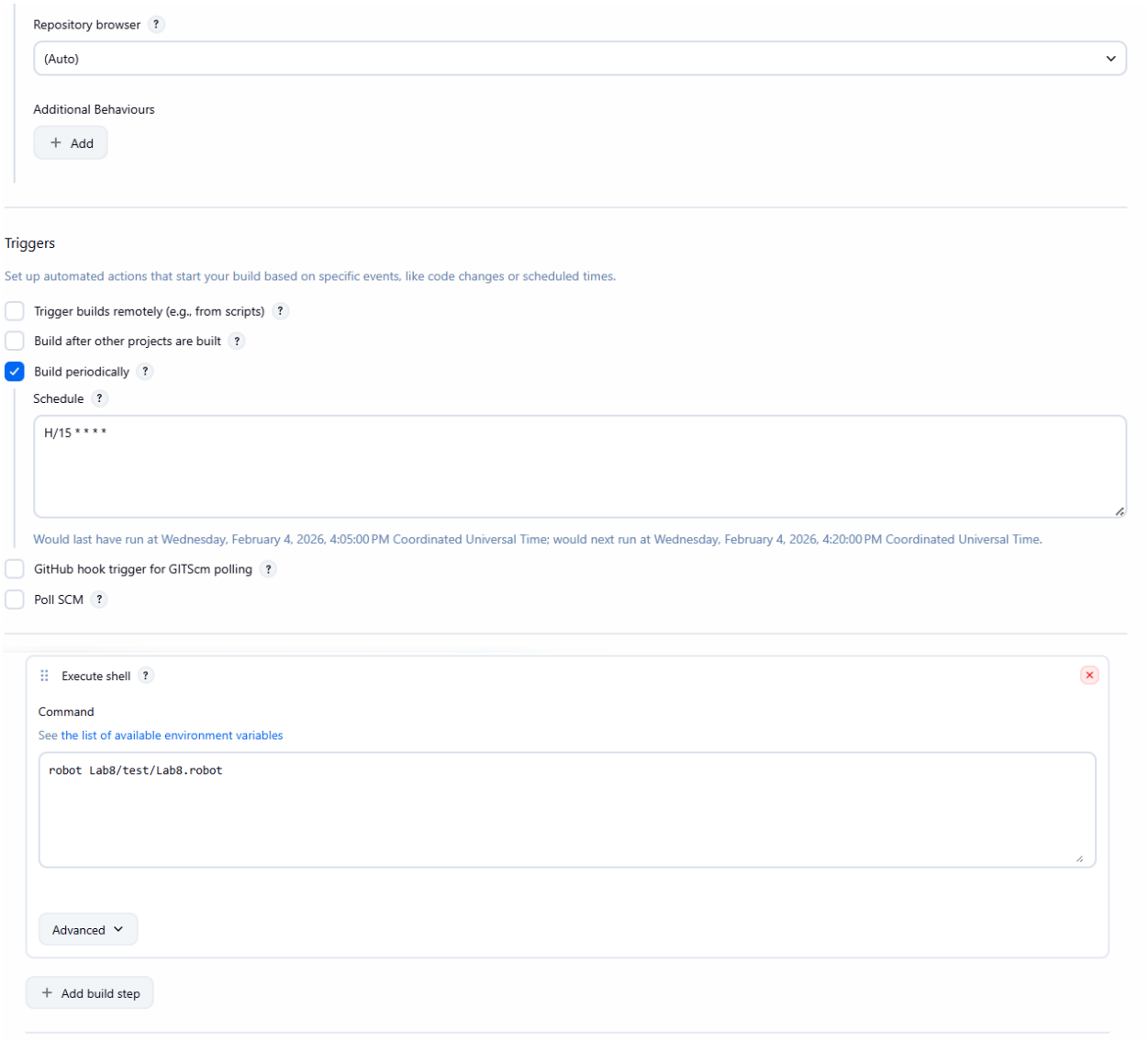
Build Steps: เลือก Execute shell แล้วใส่คำสั่งในการรันไฟล์ .robot (หากไฟล์ไม่ได้อยู่ในหน้าแรกของ repository ให้ใส่ Path ไปถึงไฟล์ให้เรียบร้อยด้วย)

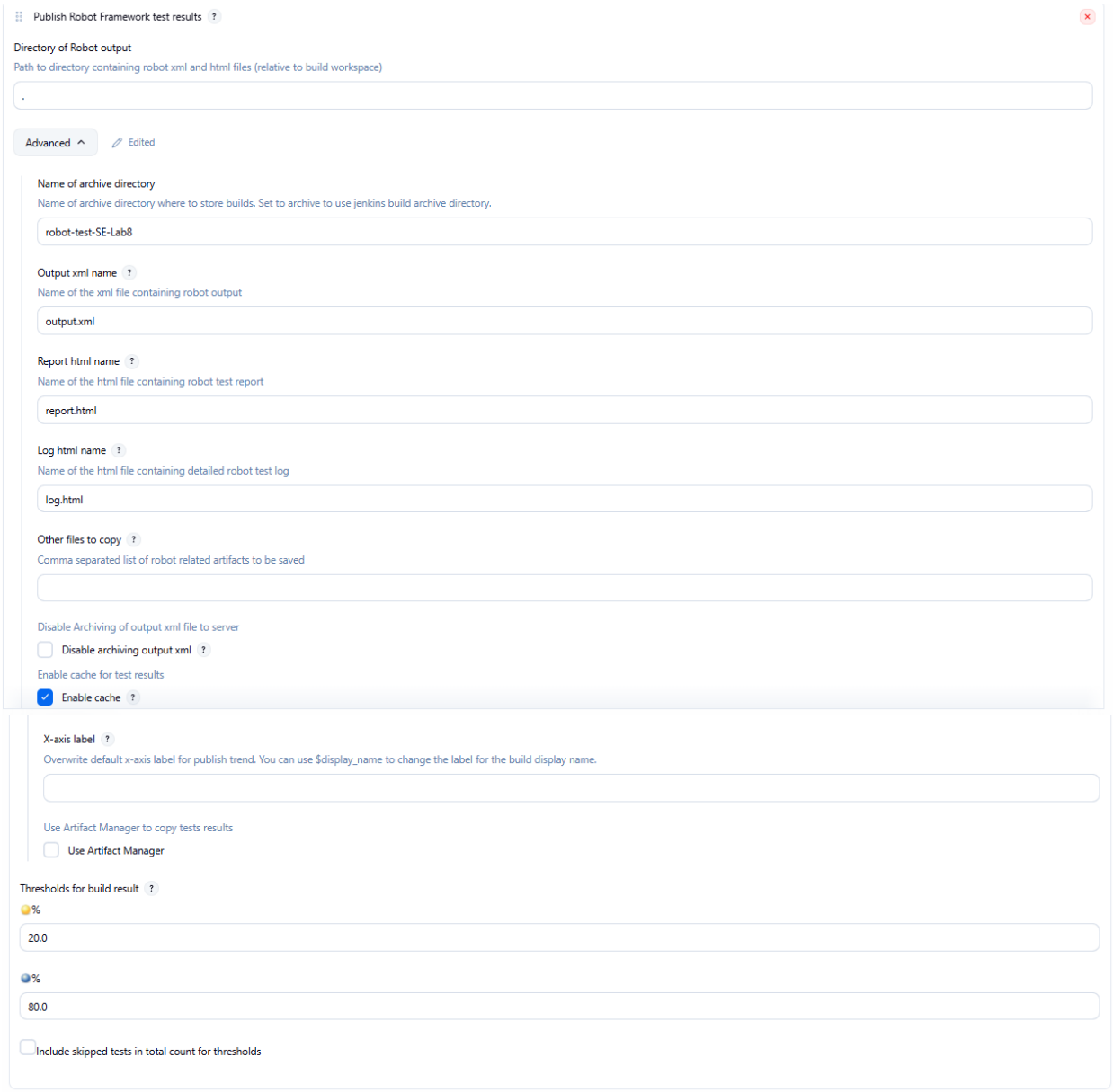
[Check point#15] Capture หน้าจอแสดงการตั้งค่า พร้อมกับตอบคำถามต่อไปนี้

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

_____robot Lab8/test/Lab8.robot_____


General
Enabled
Description
Lab 8.5
Plain text Preview
Discard old builds
GitHub project
Project url
https://github.com/Phatcharida21/CP353004-663380608-3/blob/bc69a6e0a28198e777fd12a1b54f3987525141fe/Lab8/test/Lab8.robot/
Advanced
This project is parameterized
Throttle builds
Execute concurrent builds if necessary
Advanced
Source Code Management
Connect and manage your code repository to automatically pull the latest code for your builds.
None
Git
Repositories
Repository URL
https://github.com/Phatcharida21/CP353004-663380608-3/
Credentials
- none -
Add
Advanced
Add Repository
Branches to build
Branch Specifier (blank for 'any')
*/main
Add Branch

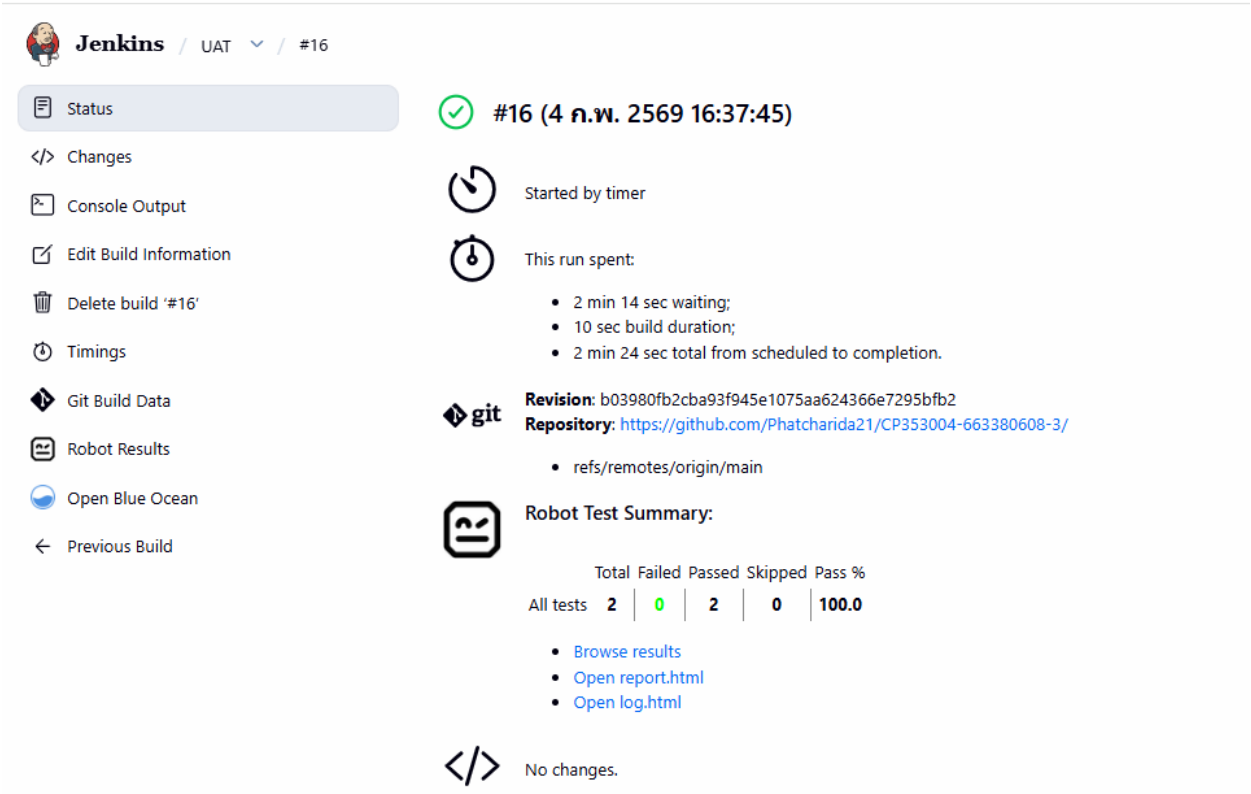

Repository browser
(Auto)
Additional Behaviours
Add
Triggers
Set up automated actions that start your build based on specific events, like code changes or scheduled times.
Trigger builds remotely (e.g., from scripts)
Build after other projects are built
Build periodically
Schedule
H/15 * * * *
Would last have run at Wednesday, February 4, 2026, 4:05:00 PM Coordinated Universal Time; would next run at Wednesday, February 4, 2026, 4:20:00 PM Coordinated Universal Time.
GitHub hook trigger for GITScm polling
Poll SCM
Execute shell
Command
See the list of available environment variables
robot Lab8/test/Lab8.robot
Advanced
Add build step



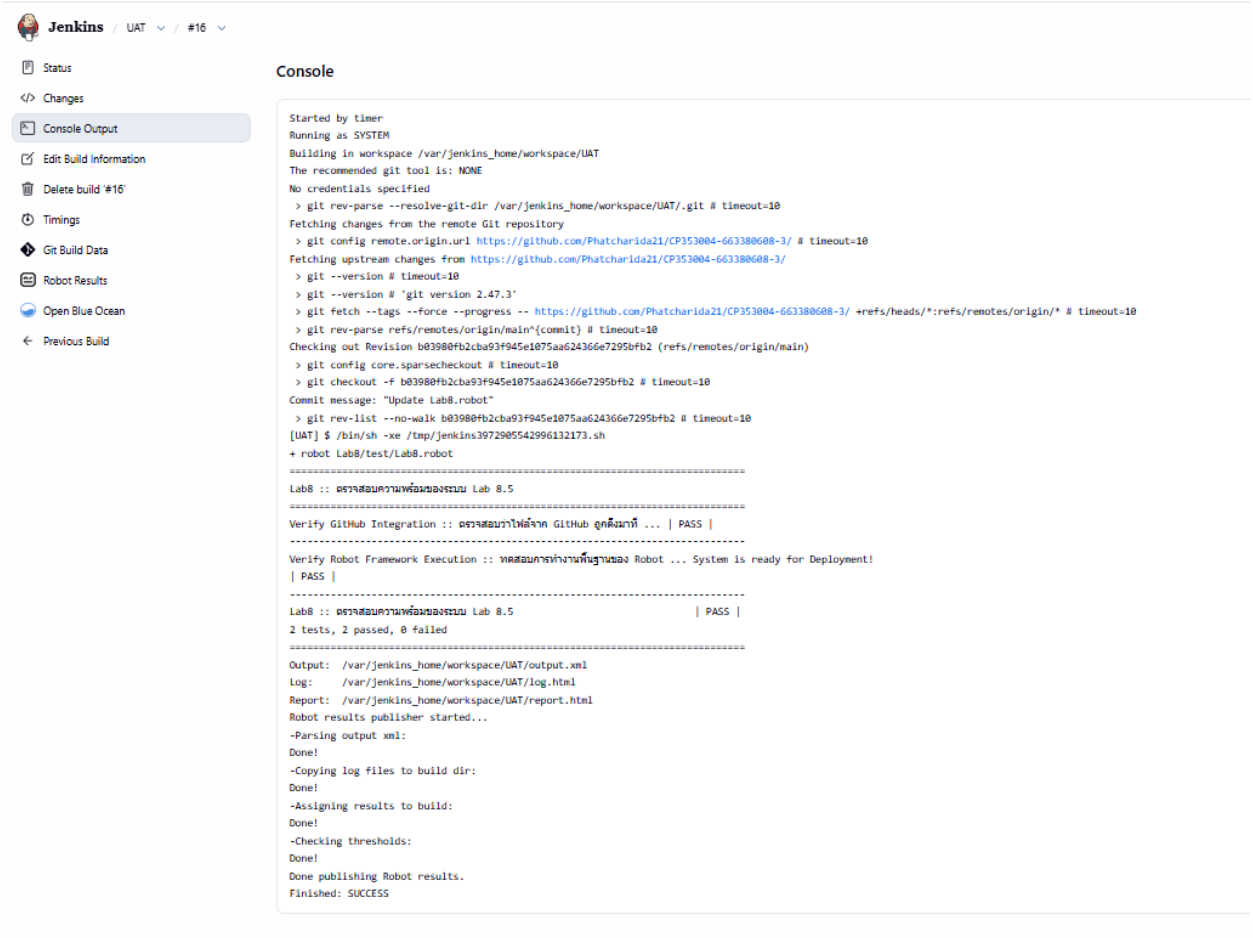
Post-build action: เพิ่ม Publish Robot Framework test results -> ระบุไดเร็คทอรีที่เก็บไฟล์ผลการทดสอบโดย Robot framework ในรูป xml และ html -> ตั้งค่า Threshold เป็น % ของการทดสอบที่ไม่ผ่านแล้วนับว่าซอฟต์แวร์มีปัญหา -> ตั้งค่า Threshold เป็น % ของการทดสอบที่ผ่านแล้วนับว่าซอฟต์แวร์มีอยู่ในสถานะที่สามารถนำไปใช้งานได้ (เช่น 20, 80)

17. กด Apply และ Save
18. สั่ง Build Now

[Check point#16] Capture หน้าจอแสดงหน้าหลักของ Pipeline และ Console Output


Jenkins / UAT / #16
Status
Changes
Console Output
Edit Build Information
Delete build '#16'
Timings
Git Build Data
Robot Results
Open Blue Ocean
Previous Build
#16 (4 ก.พ. 2569 16:37:45)
Started by timer
This run spent:
2 min 14 sec waiting;
10 sec build duration;
2 min 24 sec total from scheduled to completion.
Revision: b03980fb2cba93f945e1075aa624366e7295bfb2
Repository: https://github.com/Phatcharida21/CP353004-663380608-3/
refs/remotes/origin/main
Robot Test Summary:
Total Failed Passed Skipped Pass %
All tests 2 0 2 0 100.0
Browse results
Open report.html
Open log.html
No changes.


Jenkins / UAT / #16 / Robot result
Robot Framework Test Results
Executed: 2026-02-04T16:37:50.661349
Duration: 0:00:00.225 (+0:00:00.175)
Status: 2 test total (±0), 2 passed, 0 failed, 0 skipped
Results: report.html
log.html
Original result files
Test Result Trend
Duration Trend
Test Suites
Name Failed tests (diff) Total tests (diff) Duration (diff)
Lab8 0 (±0) 2 (±0) 0:00:00.225 (+0:00:00.175)


Jenkins / UAT / #16
Status
Changes
Console Output
Edit Build Information
Delete build '#16'
Timings
Git Build Data
Robot Results
Open Blue Ocean
Previous Build
Console
Started by timer
Running as SYSTEM
Building in workspace /var/jenkins_home/workspace/UAT
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /var/jenkins_home/workspace/UAT/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/Phatcharida21/CP353004-663380608-3/ # timeout=10
Fetching upstream changes from https://github.com/Phatcharida21/CP353004-663380608-3/
> git --version # timeout=10
> git --version # 'git version 2.47.3'
> git fetch --tags --force --progress -- https://github.com/Phatcharida21/CP353004-663380608-3/ +refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision b03980fb2cba93f945e1075aa624366e7295bfb2 (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f b03980fb2cba93f945e1075aa624366e7295bfb2 # timeout=10
Commit message: "Update Lab8.robot"
> git rev-list --no-walk b03980fb2cba93f945e1075aa624366e7295bfb2 # timeout=10
[UAT] $ /bin/sh -xe /tmp/jenkins3972905542996132173.sh
+ robot Lab8/test/Lab8.robot
==
Lab8 :: ตรวจสอบความพร้อมของระบบ Lab 8.5
==
Verify GitHub Integration :: ตรวจสอบว่าไฟล์จาก GitHub ถูกดึงมาที่ ... | PASS |

Verify Robot Framework Execution :: ทดสอบการทำงานพื้นฐานของ Robot ... System is ready for Deployment!
| PASS |

Lab8 :: ตรวจสอบความพร้อมของระบบ Lab 8.5 | PASS |
2 tests, 2 passed, 0 failed
==
Output: /var/jenkins_home/workspace/UAT/output.xml
Log: /var/jenkins_home/workspace/UAT/log.html
Report: /var/jenkins_home/workspace/UAT/report.html
Robot results publisher started...
-Parsing output xml:
Done!
-Copying log files to build dir:
Done!
-Assigning results to build:
Done!
-Checking thresholds:
Done!
Done publishing Robot results.
Finished: SUCCESS