

Lab Worksheet

ชื่อ-นามสกุล นางสาวณภสร สุปงกช รหัสนักศึกษา 653380133-3 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

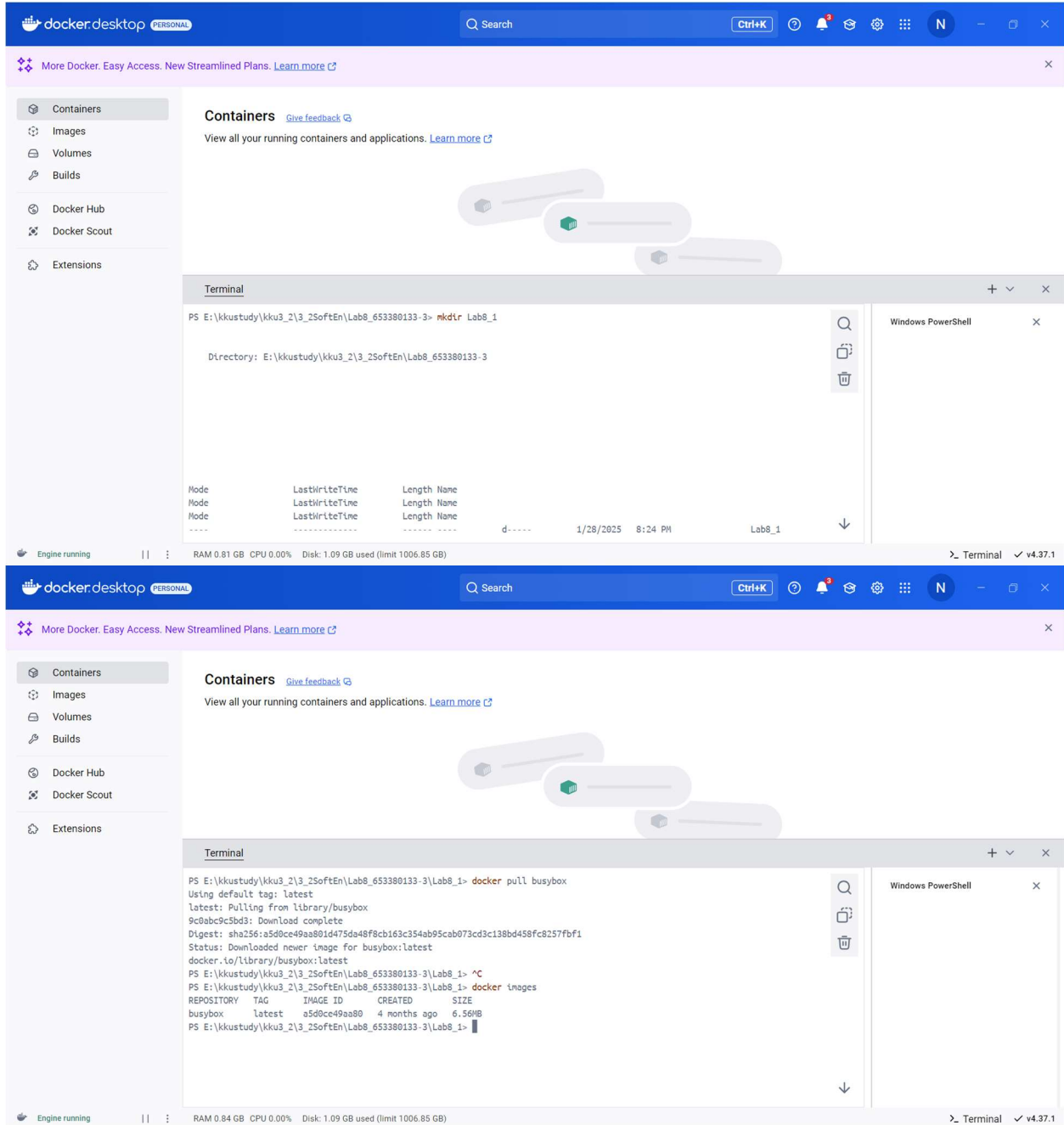
1. ติดตั้ง Docker desktop ลงบนเครื่องคอมพิวเตอร์ โดยดาวน์โหลดจาก <https://www.docker.com/get-started>
2. สร้าง Account บน Docker hub (<https://hub.docker.com/signup>)
3. กำหนดให้ \$ หมายถึง Command prompt และ <> หมายถึง ให้ป้อนค่าของพารามิเตอร์ที่กำหนด

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) แสดงผลลัพธ์ที่ได้ พร้อมกับตอบคำถามต่อไปนี้



(1) สิ่งที่อยู่ภายใต้คอลัมน์ Repository คืออะไร

ชื่อของ Docker image (ในกรณีนี้คือ busybox)

Lab Worksheet

(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

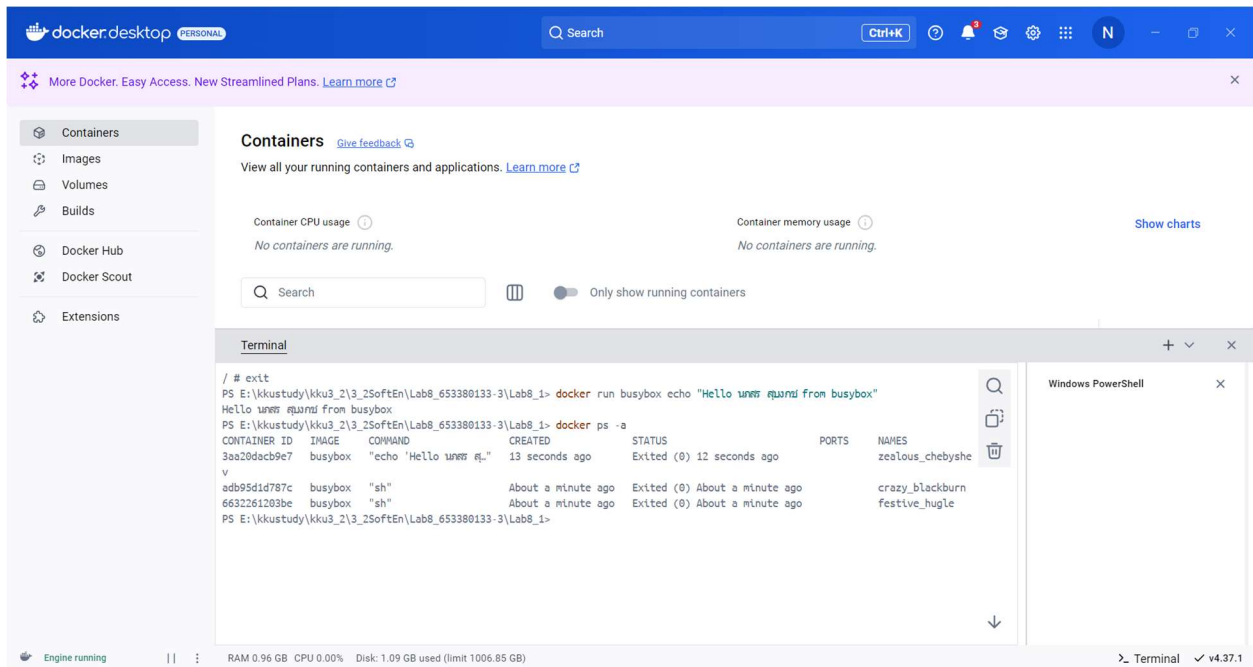
[Check point#2] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงผลลัพธ์ที่ได้ตั้งแต่ขั้นตอนที่ 6-12 พร้อมกับตอบคำถามต่อไปนี้

The screenshot shows the Docker Desktop application window. The left sidebar contains navigation options: Containers, Images, Volumes, Builds, Docker Hub, Docker Scout, and Extensions. The main area displays the 'Containers' tab, which shows 'No containers are running.' Below this, there is a search bar and a toggle switch for 'Only show running containers'. A terminal window is open at the bottom, showing the following commands and output:

```
PS E:\kkustudy\kku3_2\3_2SoftEn\Lab8_653388133-3\Lab8_1> docker run busybox
PS E:\kkustudy\kku3_2\3_2SoftEn\Lab8_653388133-3\Lab8_1> docker run -it busybox sh
/ # ls
bin  etc  lib  proc  sys  usr
dev  home lib64 root tmp var
/ # ls -la
total 48
drwxr-xr-x 1 root root      4096 Jan 28 13:55 .
drwxr-xr-x 1 root root      4096 Jan 28 13:55 ..
-rwxr-xr-x 1 root root         0 Jan 28 13:55 .dockerenv
drwxr-xr-x 2 root root     12288 Sep 26 21:31 bin
drwxr-xr-x 5 root root       360 Jan 28 13:55 dev
drwxr-xr-x 1 root root      4096 Jan 28 13:55 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
/ # exit
```

The bottom status bar indicates 'Engine running', 'RAM 0.96 GB', 'CPU 0.12%', and 'Disk: 1.09 GB used (limit 1006.85 GB)'. The terminal window title is 'Terminal' and the version is 'v4.37.1'.

Lab Worksheet



(1) เมื่อใช้ option -it ในคำสั่ง run ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป

สามารถเข้าไปใช้งาน shell ของ container (sh) และพิมพ์คำสั่งโต้ตอบใน runtime ได้

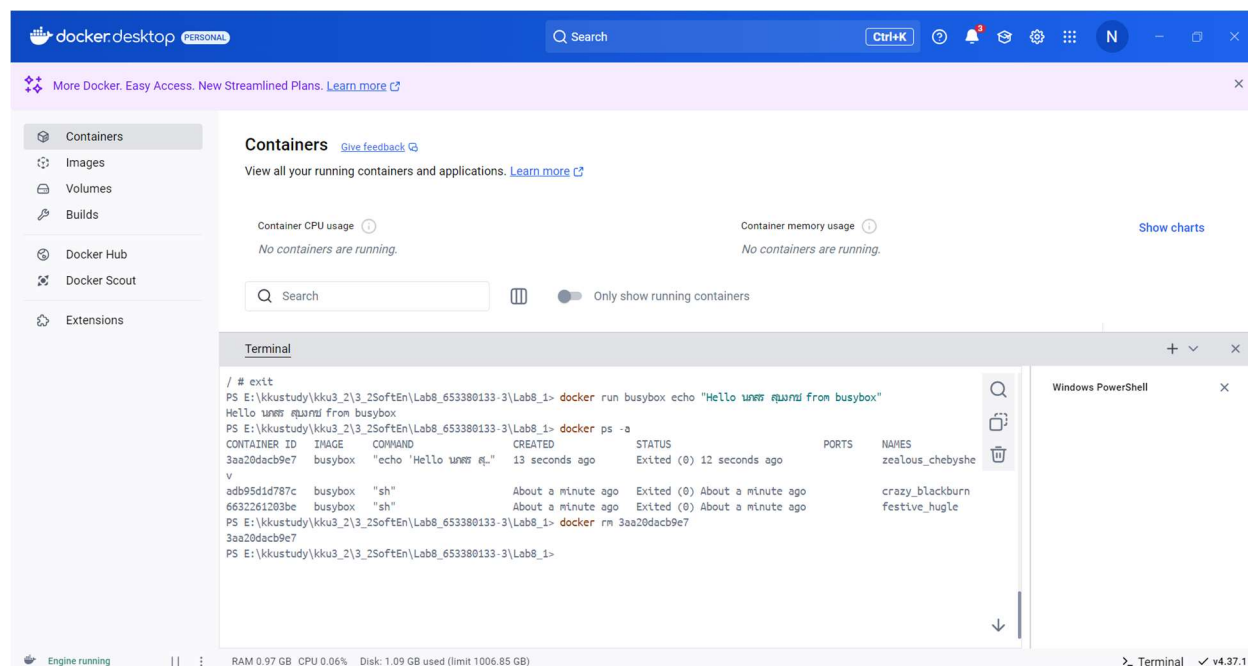
(2) คอลัมน์ STATUS จากการรันคำสั่ง docker ps -a แสดงถึงข้อมูลอะไร

แสดงสถานะของ container ว่าทำงานหรือหยุดทำงาน

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

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

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."

Lab Worksheet

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

Lab Worksheet

The first screenshot shows the Docker Desktop interface with the 'Containers' tab selected. The terminal window displays the output of the command `docker build -t dockerfile .`. The build process includes downloading the Dockerfile, resolving the base image `busybox:latest`, and exporting the resulting image. The terminal output shows the image ID `sha256:f470833a1d1c3beee68518520df9d28e0cb5ad9d271d7c3dbc9d83b2b8c048f`.

The second screenshot shows the Docker Desktop interface after the container has been built. The terminal window displays the output of the command `docker run dockerfile`. The container is successfully created and runs, showing the `3 warnings found` message. The terminal output shows the container ID `sha256:da7de08944803184e3a4a12bcf92230652db28f3fbbf033ee2a72e19d8dc51`.

(1) คำสั่งที่ใช้ในการ run คือ

`docker run dockerfile`

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

Lab Worksheet

_____ สำหรับกำหนดชื่อ (Tag) ให้กับ Docker Image ที่สร้างขึ้น ทำให้ง่ายต่อการอ้างอิงและจัดการ Image โดยใช้ชื่อที่กำหนด _____

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

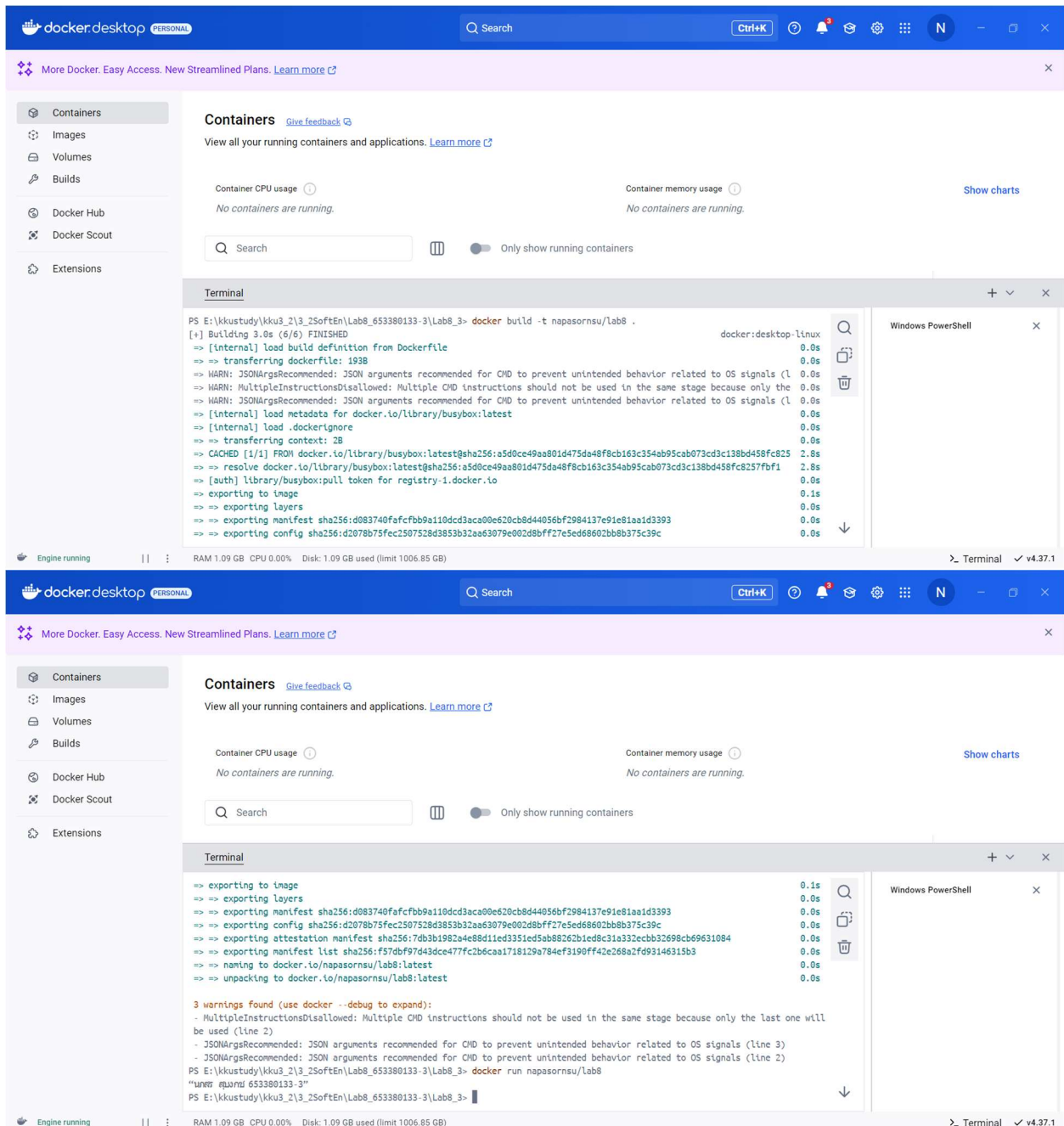
7. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้

```
$ 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 โดยการใช้คำสั่ง

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

ในกรณีที่ติดปัญหาไม่ได้ Login ไว้ก่อน ให้ใช้คำสั่งต่อไปนี้ เพื่อ Login ก่อนทำการ Push

Lab Worksheet

\$ 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)

The screenshot shows two windows. The top window is Docker Desktop, displaying the 'Containers' tab with a search bar and a toggle for 'Only show running containers'. Below this is a terminal window showing the following commands and output:

```

=> => naming to docker.io/napasornsu/lab8:latest
=> => unpacking to docker.io/napasornsu/lab8:latest

3 warnings found (use docker --debug to expand):
- 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)
- JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 2)
PS E:\kkustudy\kku3_2\3_2SoftEn\Lab8_653380133-3\Lab8_3> docker run napasornsu/lab8
"warning: 653380133-3"
PS E:\kkustudy\kku3_2\3_2SoftEn\Lab8_653380133-3\Lab8_3> docker push napasornsu/lab8
Using default tag: latest
The push refers to repository [docker.io/napasornsu/lab8]
9c0abc9c5bd3: Mounted from library/busybox
c2505abbab9b: Pushed
latest: digest: sha256:f57dbf97d43dce477fc2b6caa1718129a784ef3190ff42e268a2fd93146315b3 size: 855
PS E:\kkustudy\kku3_2\3_2SoftEn\Lab8_653380133-3\Lab8_3>
  
```

The bottom window is a web browser showing the Docker Hub repository page for 'napasornsu/lab8'. The page includes a search bar, a 'Create a repository' button, and a table of repository details:

Name	Last Pushed	Contains	Visibility	Scout
napasornsu/lab8	3 minutes ago	IMAGE	Public	Inactive

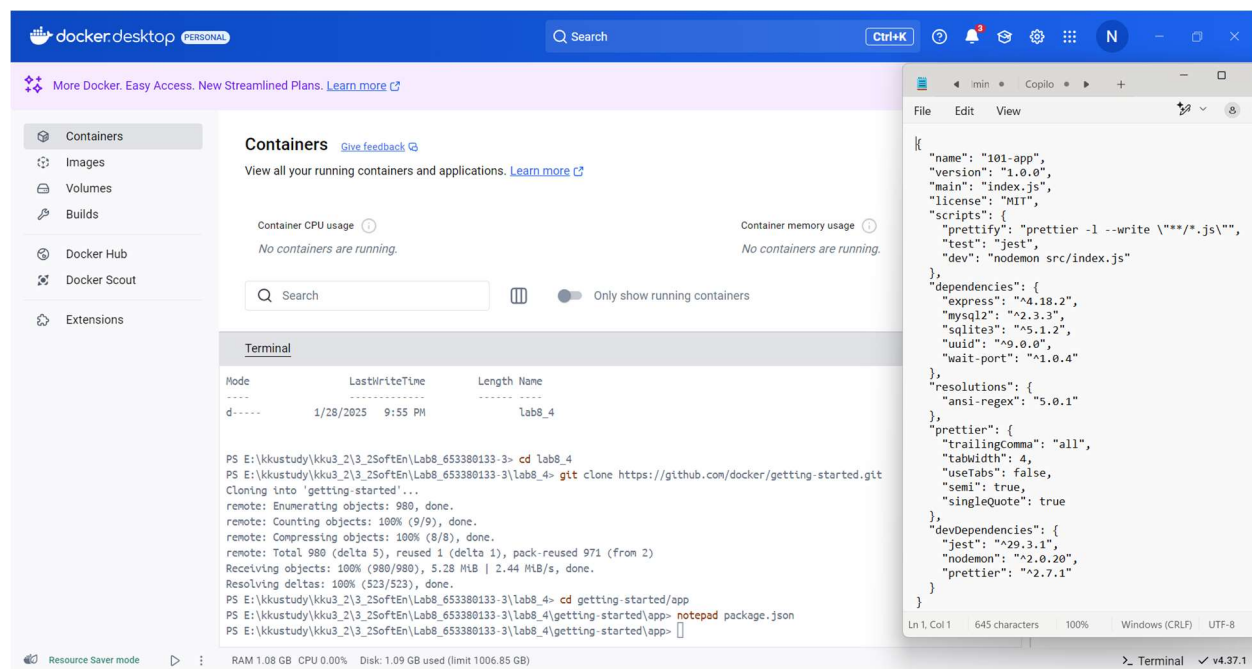
Below the table, it shows '1-1 of 1' results. On the right side of the page, there is a section titled 'Create an organization' with a link to 'Create and manage users and grant access to your repositories'.

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 (ทั้งหน้าตาและทุกหน้าต่างที่เกี่ยวข้อง) แสดงที่อยู่ของ Source code ที่ Clone มาและเนื้อหาของไฟล์ package.json



4. ภายใต้ getting-started/app ให้สร้าง Dockerfile พร้อมกับใส่เนื้อหาดังต่อไปนี้ลงในไฟล์
FROM node:18-alpine
WORKDIR /app
COPY . .
RUN yarn install --production
CMD ["node", "src/index.js"]

Lab Worksheet

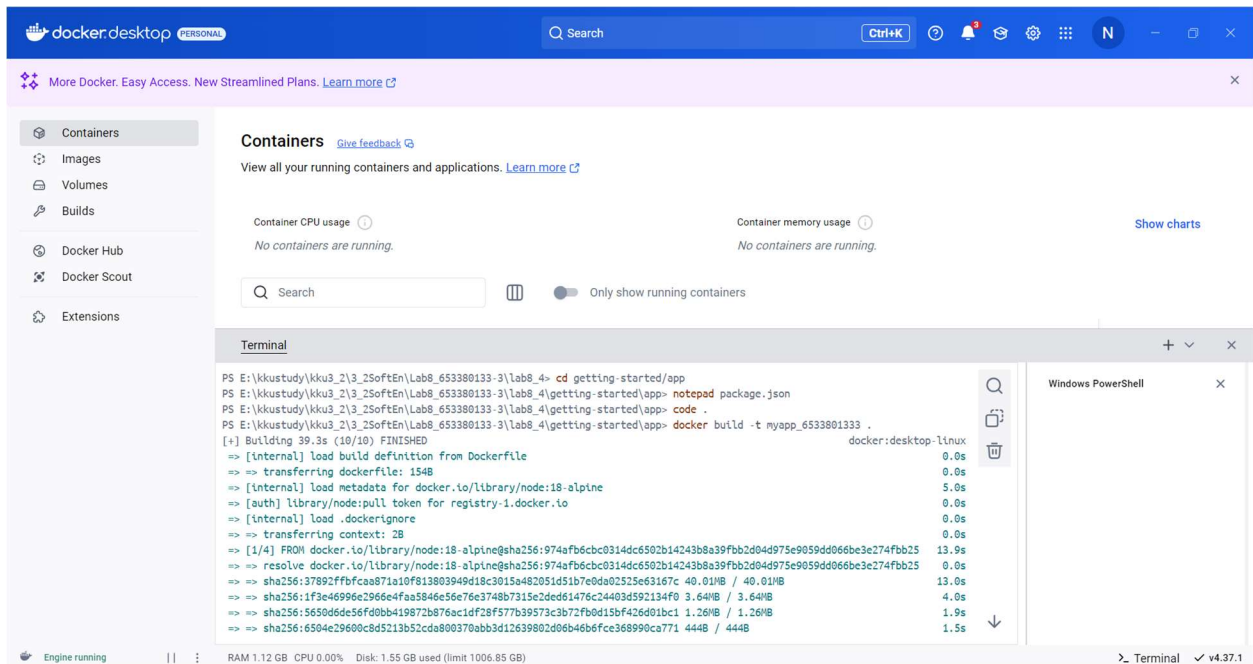
EXPOSE 3000

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

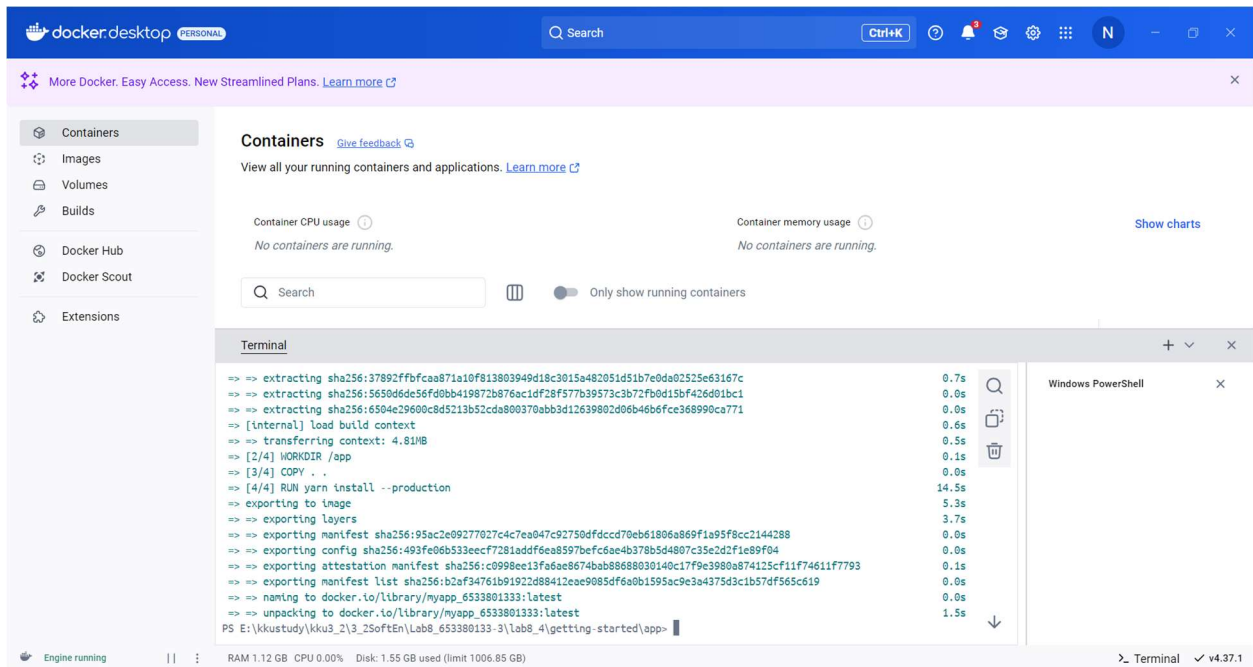
\$ docker build -t <myapp_รหัสนศ. ไม่มีขีด> .

[Check point#8] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง)

แสดงคำสั่งและผลลัพธ์ที่ได้ทางหน้าจอ



Lab Worksheet

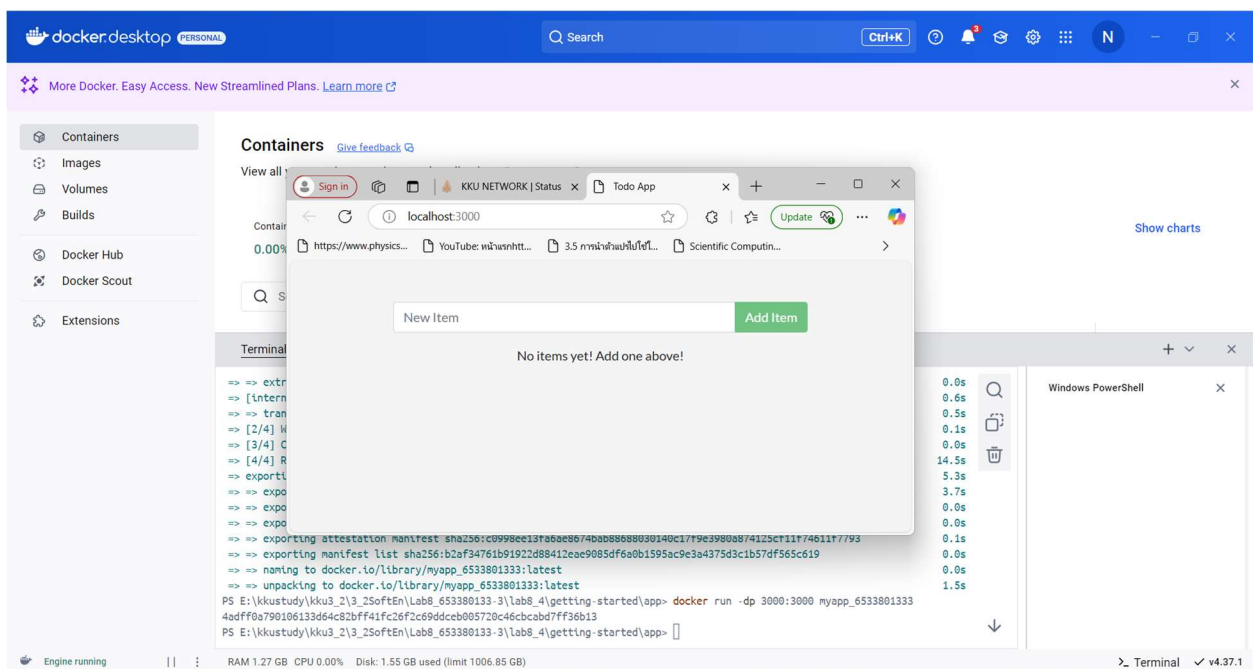


6. ทำการ Start ตัว Container ของแอปพลิเคชันที่สร้างขึ้น โดยใช้คำสั่ง

\$ 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



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

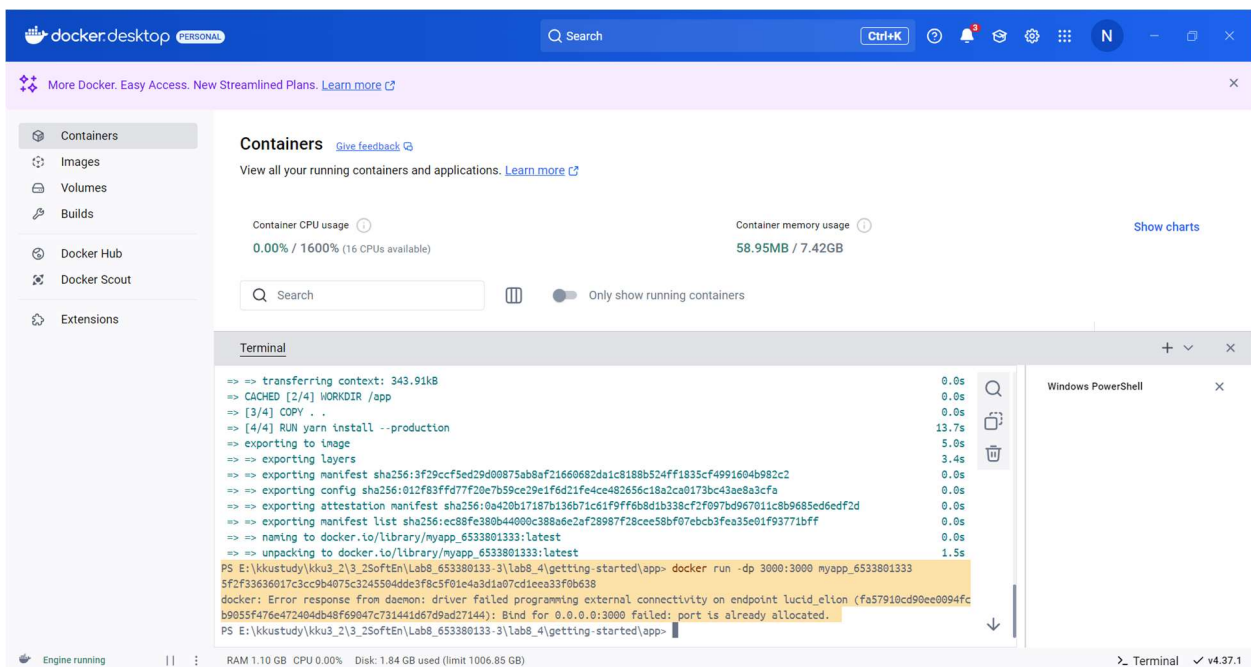
`<p className="text-center">There is no TODO item. Please add one to the list. By ชื่อและนามสกุลของนักศึกษา</p>`

- 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



(1) Error ที่เกิดขึ้นหมายความว่าอย่างไร และเกิดขึ้นเพราะอะไร

เกิดจาก Port 3000 บนเครื่องของคุณถูกใช้งานอยู่แล้ว

Lab Worksheet

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

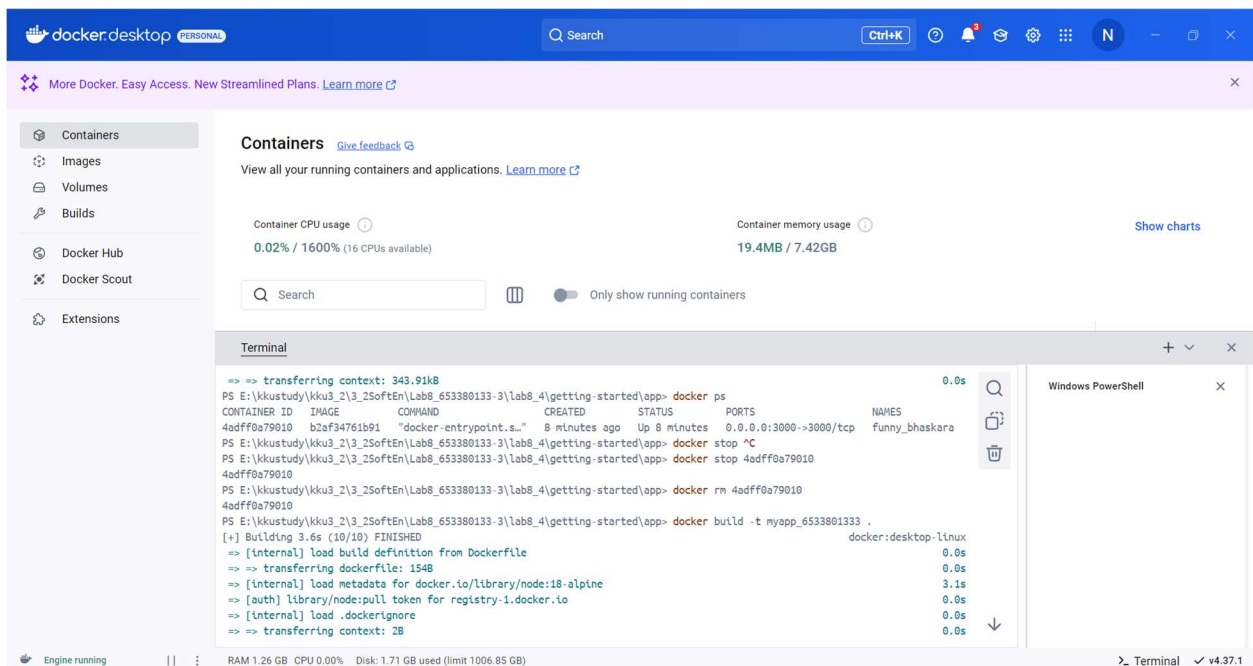
b. ผ่าน Docker desktop

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

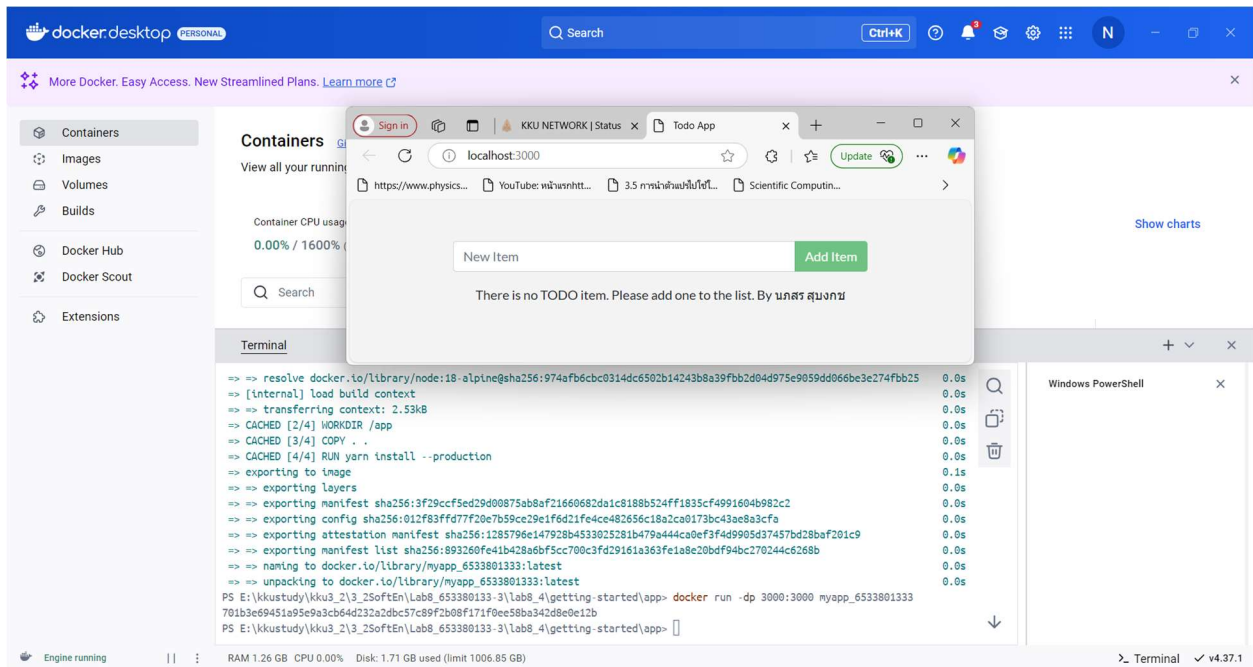
12. Start และรัน Container ตัวใหม่อีกครั้ง โดยใช้คำสั่งเดียวกันกับข้อ 6

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

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



Lab Worksheet

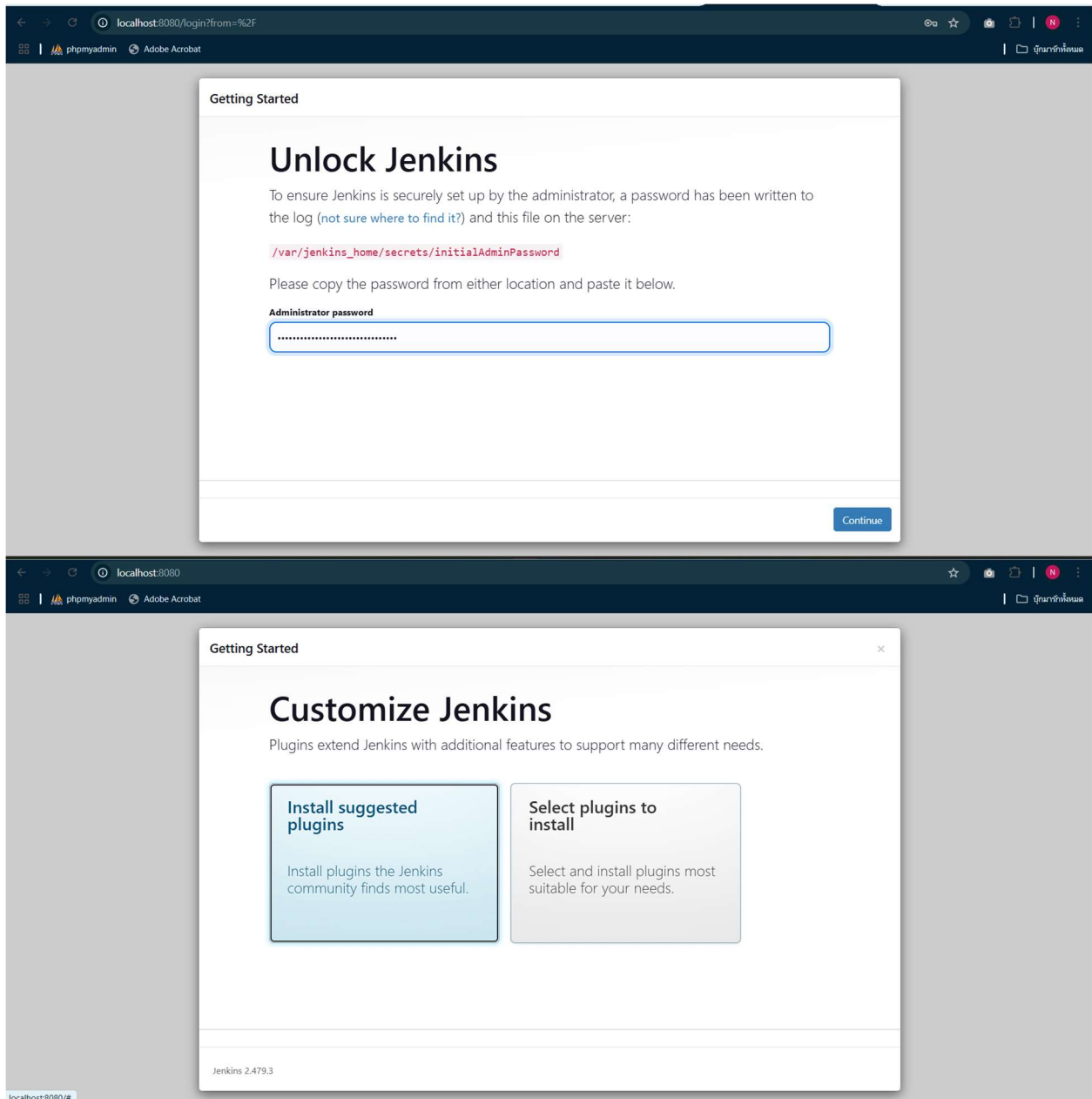


แบบฝึกปฏิบัติที่ 8.5: เริ่มต้นสร้าง Pipeline อย่างง่ายสำหรับการ Deploy ด้วย Jenkins

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:its-jdk17`
3. บันทึกรหัสผ่านของ Admin user ไว้สำหรับ log-in ในครั้งแรก
 1564e8f2ae9a4ab4a811e8118ab43042

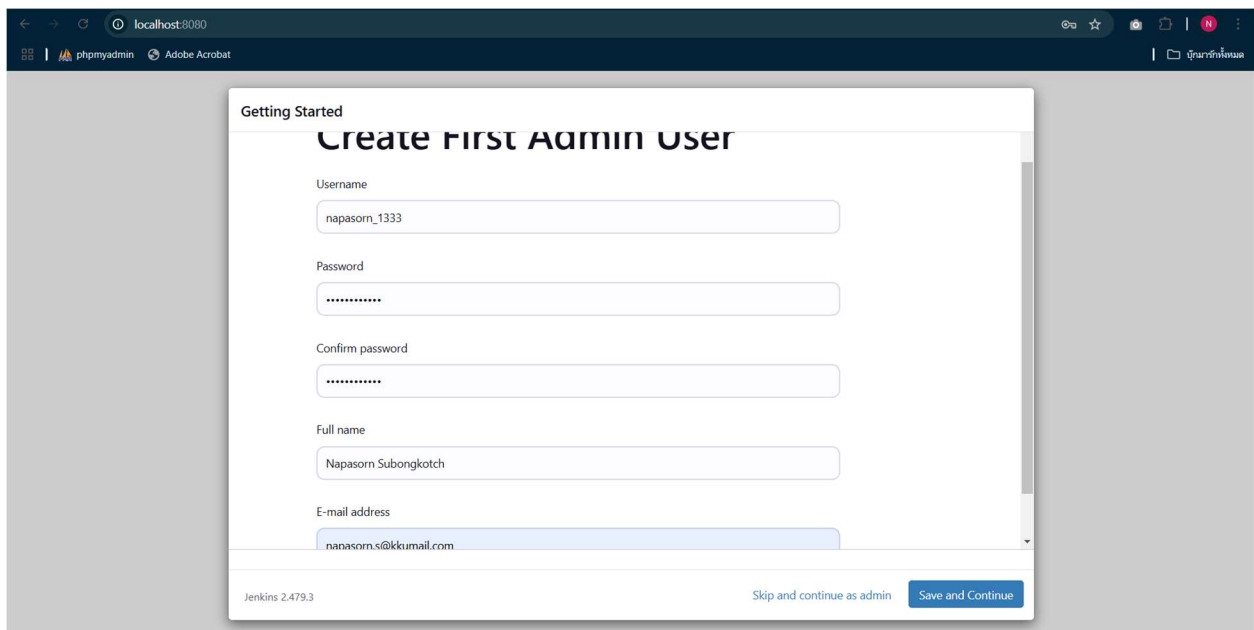
[Check point#12] Capture หน้าจอที่แสดงผล Admin password

Lab Worksheet



- เมื่อได้รับการยืนยันว่า Jenkins is fully up and running ให้เปิดบราวเซอร์ และป้อนที่อยู่เป็น `localhost:8080`
 - ทำการ Unlock Jenkins ด้วยรหัสผ่านที่ได้ในข้อที่ 3
 - สร้าง Admin User โดยใช้ username เป็นชื่อจริงของนักศึกษาพร้อมรหัสสี่ตัวท้าย เช่น `somsri_3062`
- [Check point#13] Capture หน้าจอที่แสดงผลการตั้งค่า

Lab Worksheet

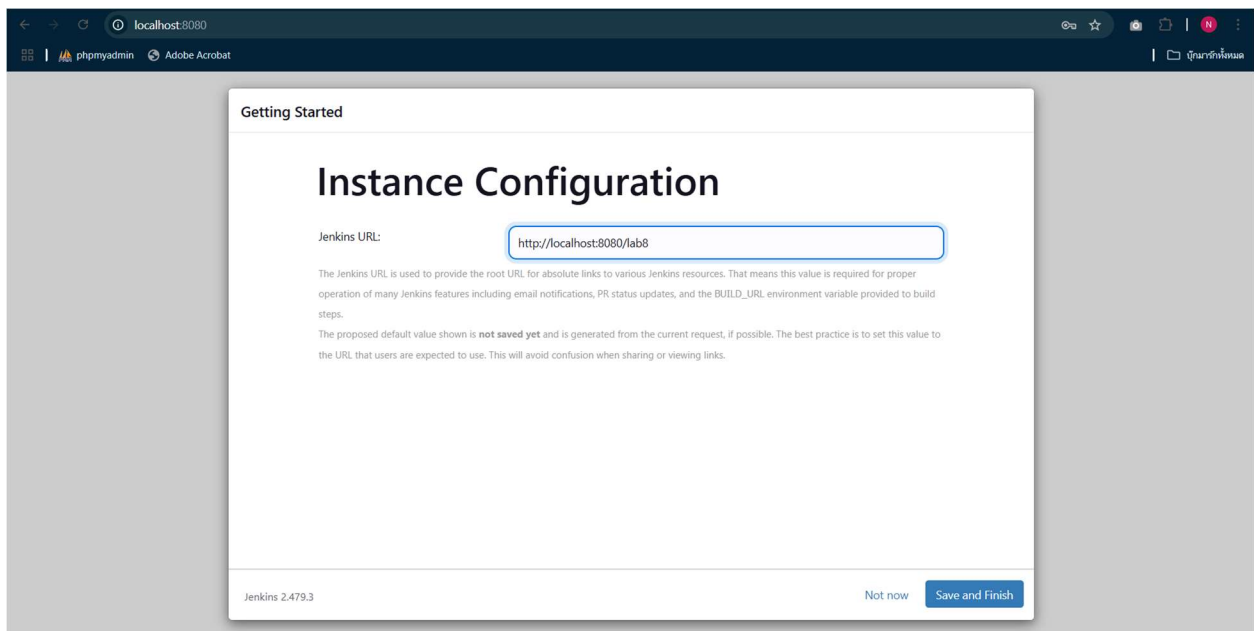


The screenshot shows the Jenkins 'Getting Started' page with the 'Create First Admin User' form. The form fields are filled with the following information:

- Username: napasorn_1333
- Password: (masked with dots)
- Confirm password: (masked with dots)
- Full name: Napasorn Subongkotch
- E-mail address: napasorn.s@kkumail.com

At the bottom of the form, there are two buttons: 'Skip and continue as admin' and 'Save and Continue'. The Jenkins version 'Jenkins 2.479.3' is displayed in the bottom left corner.

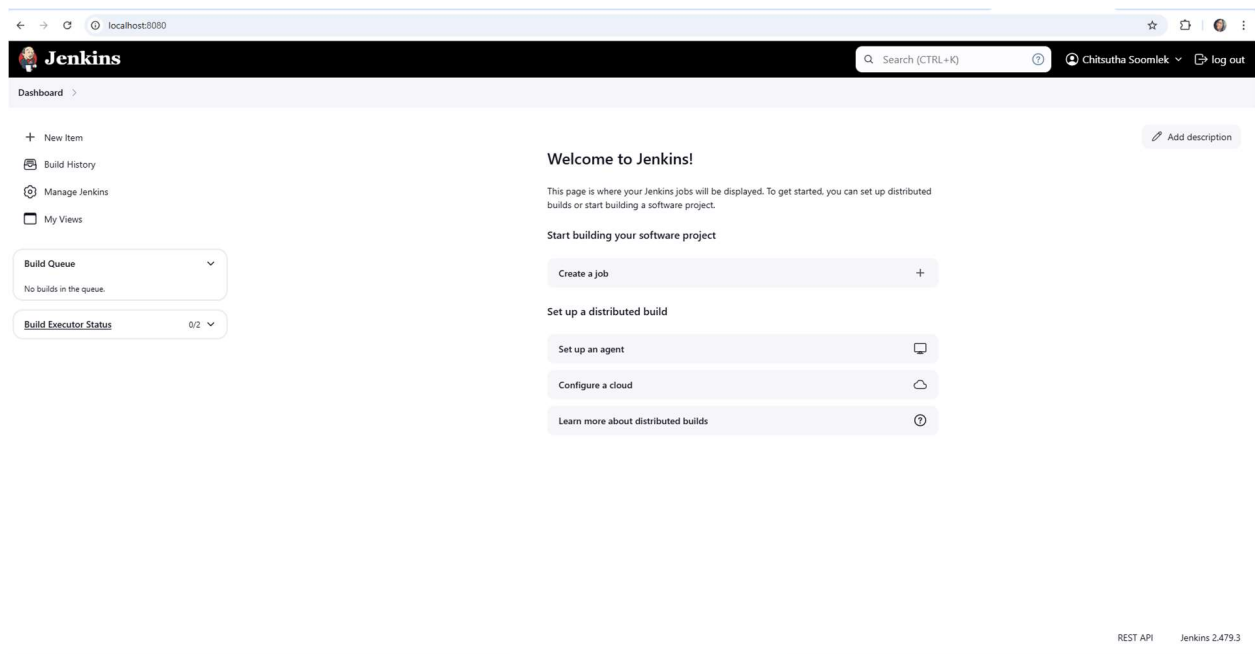
7. กำหนด Jenkins URL เป็น <http://localhost:8080/lab8>



The screenshot shows the Jenkins 'Getting Started' page with the 'Instance Configuration' form. The 'Jenkins URL' field is filled with the value 'http://localhost:8080/lab8'. Below the field, there is explanatory text about the Jenkins URL and its importance for various features. At the bottom of the form, there are two buttons: 'Not now' and 'Save and Finish'. The Jenkins version 'Jenkins 2.479.3' is displayed in the bottom left corner.

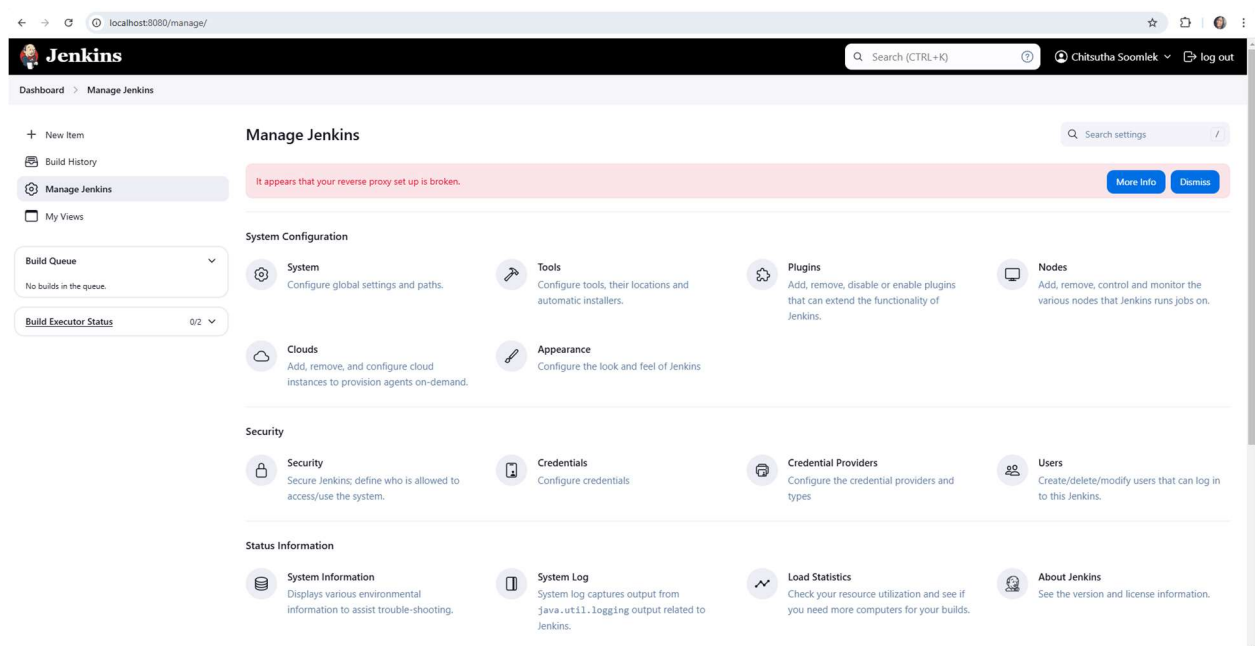
8. เมื่อติดตั้งเรียบร้อยแล้วจะพบกันหน้า Dashboard ดังแสดงในภาพ

Lab Worksheet



The screenshot shows the Jenkins Dashboard at localhost:8080. The top navigation bar includes the Jenkins logo, a search bar, and a user profile for Chitsutha Soomlek with a log out button. The left sidebar contains links for New Item, Build History, Manage Jenkins, and My Views. The main content area displays a 'Welcome to Jenkins!' message and a 'Start building your software project' section with buttons for 'Create a job', 'Set up a distributed build', 'Set up an agent', 'Configure a cloud', and 'Learn more about distributed builds'. A 'Build Queue' widget shows 'No builds in the queue.' and a 'Build Executor Status' widget shows '0/2'.

9. เลือก Manage Jenkins แล้วไปที่เมนู Plugins



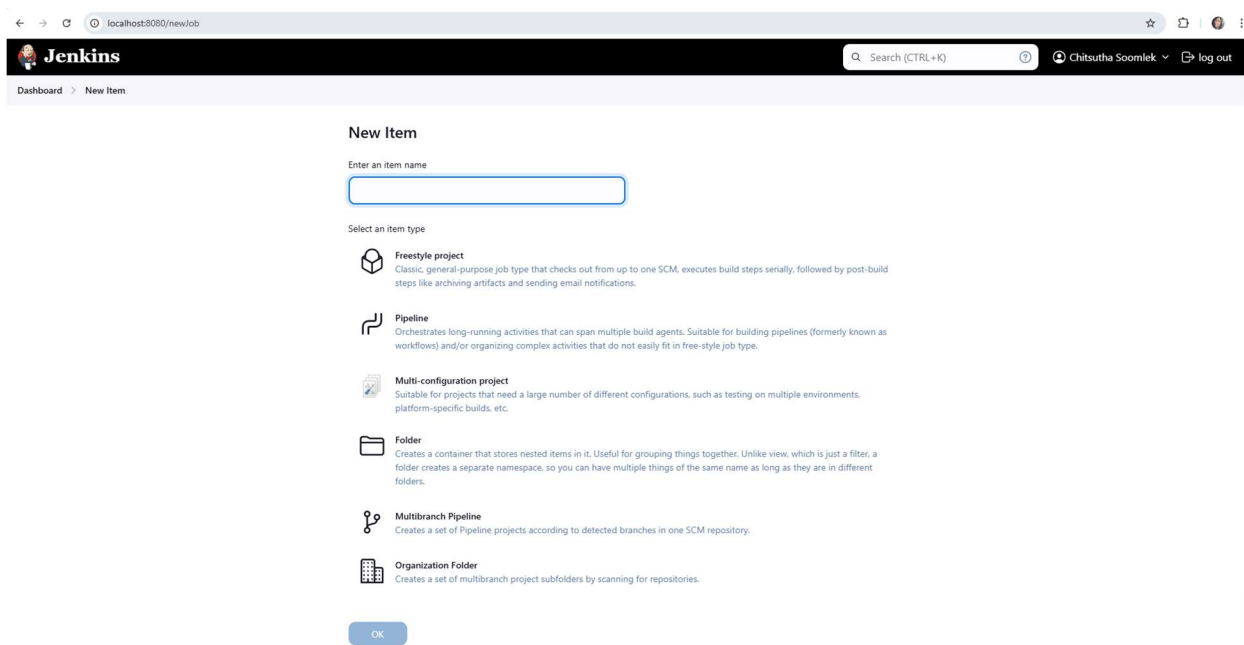
The screenshot shows the Jenkins 'Manage Jenkins' page at localhost:8080/manage/. A red banner at the top indicates 'It appears that your reverse proxy set up is broken.' with 'More Info' and 'Dismiss' buttons. The page is organized into several sections: 'System Configuration' (System, Tools, Plugins, Nodes, Clouds, Appearance), 'Security' (Security, Credentials, Credential Providers, Users), and 'Status Information' (System Information, System Log, Load Statistics, About Jenkins). Each section contains a brief description of its function.

Lab Worksheet

10. ไปที่เมนู Available plugins แล้วเลือกติดตั้ง Robotframework เพิ่มเติม



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



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

Lab Worksheet

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

[Check point#14] Capture หน้าจอแสดงการตั้งค่า พร้อมกับตอบคำถามต่อไปนี้

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

Top Screenshot: General Tab

- Configuration:** General (Selected), Source Code Management, Build Triggers, Build Environment, Build Steps, Post-build Actions.
- General:** Enabled (Toggle).
- Description:** Lab 8.5
- Plain text:** Preview
- Discard old builds:** ☐
- GitHub project:** ☒
 - Project url:** https://github.com/napasornsu/Lab8_5_653380133-3.git/
 - Advanced:** ☐
- This project is parameterized:** ☐
- Buttons:** Save, Apply

Bottom Screenshot: Source Code Management Tab

- Configuration:** General, Source Code Management (Selected), Build Triggers, Build Environment, Build Steps, Post-build Actions.
- Source Code Management:**
 - None:** ☒
 - Git:** ☐
- Build Triggers:**
 - Trigger builds remotely (e.g., from scripts):** ☐
 - Build after other projects are built:** ☐
 - Build periodically:** ☒
 - Schedule:**
 - Would last have run at Tuesday, January 28, 2025 at 4:05:39 PM Coordinated Universal Time; would next run at Tuesday, January 28, 2025 at 4:20:39 PM Coordinated Universal Time.
 - GitHub hook trigger for GITScm polling:** ☐
 - Poll SCM:** ☐
- Buttons:** Save, Apply

Lab Worksheet

The image shows two screenshots of the Jenkins configuration page for a UAT job. The top screenshot displays the 'Build Steps' section, where a single step named 'Execute shell' is configured with the command 'robot testttt.robot'. The bottom screenshot displays the 'Post-build Actions' section, showing the 'Directory of Robot output' set to '.', and 'Thresholds for build result' set to 20.0 and 80.0. A checkbox labeled 'DEPRECATED! THIS FLAG DOES NOTHING! - Use thresholds for critical tests only' is checked.

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

robot testttt.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

The top screenshot shows the Jenkins 'Console Output' for a build named 'UAT #1'. The output text is as follows:

```
Started by user Napasorn Subongkotch
Running as SYSTEM
Building in workspace /var/jenkins_home/workspace/UAT
[UAT] $ /bin/sh -xe /tmp/jenkins8940082619000507533.sh
+ robot testtt.robot
/tmp/jenkins8940082619000507533.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)
```

The bottom screenshot shows the same console output but with more detailed error information, including the full stack trace and the final status 'Finished: FAILURE'.

```
[UAT] $ /bin/sh -xe /tmp/jenkins8940082619000507533.sh
+ robot testtt.robot
/tmp/jenkins8940082619000507533.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
```