

Lab Worksheet

ชื่อ-นามสกุล ณัฐภัทร ตรังวัฒนาวุฒิ รหัสนักศึกษา 653380197-7 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


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

Lab Worksheet

```
C:\CP353004(SoftwareEngineering)\Lab8_1>docker pull busybox
Using default tag: latest
latest: Pulling from library/busybox
9c0abc9c5bd3: Download complete
Digest: sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138bd458fc8257f
bf1
Status: Downloaded newer image for busybox:latest
docker.io/library/busybox:latest

C:\CP353004(SoftwareEngineering)\Lab8_1>docker images
REPOSITORY      TAG        IMAGE ID      CREATED        SIZE
busybox          latest     a5d0ce49aa80  4 months ago  6.56MB

C:\CP353004(SoftwareEngineering)\Lab8_1>|
```



- (1) สิ่งที่อยู่ภายใต้คอลัมน์ Repository คืออะไร Repository หมายถึงชื่อของ Image ที่ถูกดึงมาจาก Docker Hub หรือ Registry อื่น เช่น busybox ใน
- (2) Tag ที่ใช้บ่งบอกถึงอะไร Tag ใช้บ่งบอกถึง เวอร์ชัน หรือ สถานะ ของ Image ตัวนั้น เช่น latest คือ Tag ที่ใช้ระบุเวอร์ชันล่าสุด
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

```
C:\CP353004(SoftwareEngineering)\Lab8_1>docker run busybox
C:\CP353004(SoftwareEngineering)\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 27 09:24 .
drwxr-xr-x 1 root    root          4096 Jan 27 09:24 ..
-rwxr-xr-x 1 root    root           0 Jan 27 09:24 .dockerenv
drwxr-xr-x 2 root    root        12288 Sep 26 21:31 bin
drwxr-xr-x 5 root    root         360 Jan 27 09:24 dev
drwxr-xr-x 1 root    root         4096 Jan 27 09:24 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
lrwxrwxrwx 1 root    root           3 Sep 26 21:31 lib64 -> lib
dr-xr-xr-x 236 root   root           0 Jan 27 09:24 proc
drwx----- 1 root    root         4096 Jan 27 09:25 root
dr-xr-xr-x 11 root    root           0 Jan 27 09:24 sys
drwxrwxrwt 2 root    root         4096 Sep 26 21:31 tmp
drwxr-xr-x 4 root    root         4096 Sep 26 21:31 usr
drwxr-xr-x 4 root    root         4096 Sep 26 21:31 var
/ # exit

C:\CP353004(SoftwareEngineering)\Lab8_1>docker run busybox echo "Hello ณัฐภ"
หรือ ตรงๆ ง่ายๆ ฝึ from busybox
Hello ณัฐภ หรือ ตรงๆ ง่ายๆ ฝึ from busybox

C:\CP353004(SoftwareEngineering)\Lab8_1>docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS              PORTS          NAMES
cf81b3cea957   busybox   "echo 'Hello ณัฐภ หรือ'" 8 seconds ago  Exited (0) 7 seconds ago           hungry_ishizaka
100bb7eb3c2f   busybox   "sh"                    2 minutes ago  Exited (0) 36 seconds ago           hopeful_hodgkin
3299381d95c6   busybox   "sh"                    2 minutes ago  Exited (0) 2 minutes ago           lucid_visvesvaraya
```

(1) เมื่อใช้ option -it ในคำสั่ง run ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป

Option -it คือการรวม -i (interactive mode) และ -t (allocate a pseudo-TTY) ซึ่งทำให้สามารถโต้ตอบกับ container ได้แบบ interactive ผ่าน terminal (เช่น ใช้คำสั่ง sh หรือ bash ได้).

- -i ทำให้ container ยังคงเปิดอยู่และรับ input จากผู้ใช้.
- -t สร้าง terminal จำลองสำหรับการพิมพ์คำสั่งและแสดงผลลัพธ์.

(2) คอลัมน์ STATUS จากการรันคำสั่ง docker ps -a แสดงถึงข้อมูลอะไร

คอลัมน์ STATUS ใช้แสดงสถานะของ container ว่ากำลังทำงานหรือหยุดทำงานแล้ว โดยประกอบด้วย:

- "Up X seconds/minutes": แสดงว่า container กำลังรันอยู่ในเวลาที่ระบุ.
- "Exited (Code) X seconds/minutes ago": แสดงว่า container ได้หยุดทำงานแล้ว พร้อมรหัสสถานะการหยุด (Code).
- "Created": แสดงว่า container ถูกสร้างขึ้น แต่ยังไม่เริ่มทำงาน

Lab Worksheet

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

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

```
C:\CP353004(SoftwareEngineering)\Lab8_1>docker rm cf81b3cea957
cf81b3cea957

C:\CP353004(SoftwareEngineering)\Lab8_1>docker ps -a
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS              PORTS          NAMES
100bb7eb3c2f   busybox    "sh"                    10 minutes ago Exited (0) 9 minutes ago           hopeful_hodgkin
3299381d95c6   busybox    "sh"                    10 minutes ago Exited (0) 10 minutes ago           lucid_visvesvaraya

C:\CP353004(SoftwareEngineering)\Lab8_1>
```

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

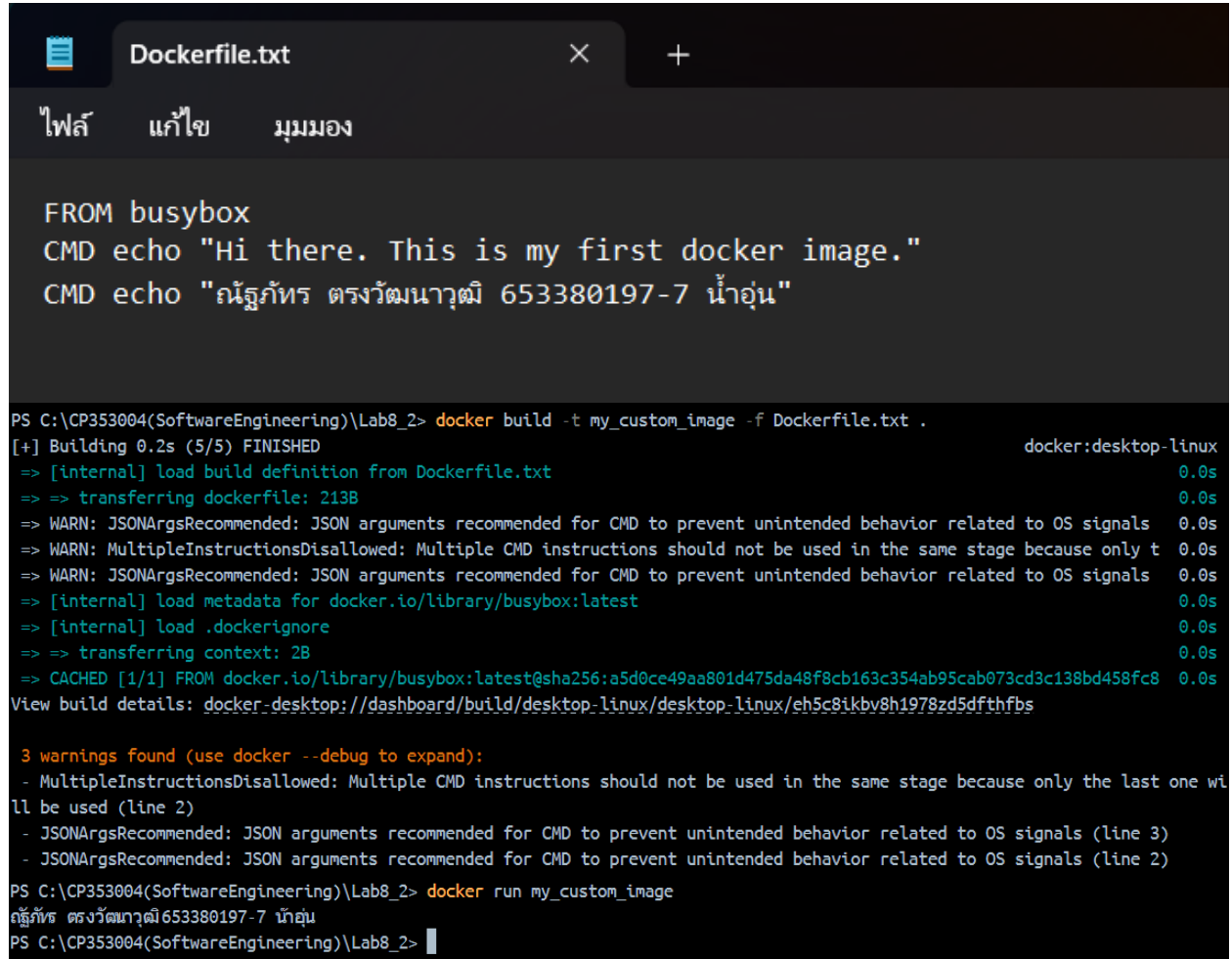
5. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้

\$ docker build -t <ชื่อ Image> .

Lab Worksheet

6. เมื่อ Build สำเร็จแล้ว ให้ทำการรัน Docker image ที่สร้างขึ้นในขั้นตอนที่ 5

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



```

Dockerfile.txt
ไฟล์ แก้ไข มุมมอง

FROM busybox
CMD echo "Hi there. This is my first docker image."
CMD echo "ณัฐภัทร ตรงวัฒนาวุฒิ 653380197-7 น้าอุ่น"

PS C:\CP353004(SoftwareEngineering)\Lab8_2> docker build -t my_custom_image -f Dockerfile.txt .
[+] Building 0.2s (5/5) FINISHED                                docker:desktop-linux
=> [internal] load build definition from Dockerfile.txt        0.0s
=> => transferring dockerfile: 213B                             0.0s
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals 0.0s
=> WARN: MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the same stage because only t 0.0s
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals 0.0s
=> [internal] load metadata for docker.io/library/busybox:latest 0.0s
=> [internal] load .dockerignore                                0.0s
=> => transferring context: 2B                                    0.0s
=> CACHED [1/1] FROM docker.io/library/busybox:latest@sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138bd458fc8 0.0s
View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/eh5c8ikbv8h1978zd5dfthfbs

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 C:\CP353004(SoftwareEngineering)\Lab8_2> docker run my_custom_image
ณัฐภัทร ตรงวัฒนาวุฒิ 653380197-7 น้าอุ่น
PS C:\CP353004(SoftwareEngineering)\Lab8_2>

```

- (1) คำสั่งที่ใช้ในการ run คือ

`docker run <ชื่อ Image>`

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

Option -t ใช้สำหรับกำหนดชื่อ (tag) ให้กับ Docker Image ที่กำลังถูกสร้าง.

- ช่วยให้สามารถอ้างอิง Image ได้สะดวกโดยใช้ชื่อที่กำหนด (แทนการอ้างอิงด้วย Image ID).
- ชื่อ Image ที่สร้างจะอยู่ในรูปแบบ <Repository>:<Tag> เช่น myfirstimage:latest (ถ้าไม่ได้กำหนด Tag เฉพาะ จะใช้ latest เป็นค่าเริ่มต้น)

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
```

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

Lab Worksheet

```

PS C:\CP353004(SoftwareEngineering)\Lab8_3> docker build -t nutthapatrongwattanawut/lab8 -f Dockerfile.txt .
[+] Building 4.0s (6/6) FINISHED      docker:desktop-linux
=> [internal] load build definition from Dockerfile.txt 0.0s
=> => transferring dockerfile: 226B 0.0s
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 3) 0.0s
=> WARN: MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the same stage because only the last one will be used (line 2) 0.0s
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 2) 0.0s
=> [internal] load metadata for docker.io/library/busybox 0.0s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> CACHED [1/1] FROM docker.io/library/busybox:latest@sha256:a3bd380197-7 3.7s
=> => resolve docker.io/library/busybox:latest@sha256:a3bd380197-7 3.7s
=> [auth] library/busybox:pull token for registry-1.docker.io 0.0s
=> exporting to image 0.1s
=> => exporting layers 0.0s
=> => exporting manifest sha256:366996e861f00786eb54e84 0.0s
=> => exporting config sha256:456af911720465720ce7c8058 0.0s
=> => exporting attestation manifest sha256:591ca003576 0.0s
=> => exporting manifest list sha256:11e7355cb9df57bde9 0.0s
=> => naming to docker.io/nutthapatrongwattanawut/lab8 0.0s

3 warnings found (use docker --debug to expand):
- JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 3)
- 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 2)

PS C:\CP353004(SoftwareEngineering)\Lab8_3> docker run nutthapatrongwattanawut/lab8
"อิฐก้อนนี้ ตรงวัดนาญาติ 653380197-7"
PS C:\CP353004(SoftwareEngineering)\Lab8_3>

```

Dockerfile.txt

ไฟล์

แก้ไข

มุมมอง

```

FROM busybox
CMD echo "Hi there. My work is done. You can run them from my Docker image."
CMD echo "อิฐก้อนนี้ ตรงวัดนาญาติ 653380197-7"

```

Lab Worksheet

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

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

```
PS C:\CP353004(SoftwareEngineering)\Lab8_3> docker push nutthapatrongwattanawut/lab8
Using default tag: latest
The push refers to repository [docker.io/nutthapatrongwattanawut/lab8]
d7f67fbd5d9e: Pushed
9c0abc9c5bd3: Mounted from library/busybox
latest: digest: sha256:11e7355cb9df57bde9c3c794c0306ff710e2f5c6399b150851d992f8b796ca5e size: 855
PS C:\CP353004(SoftwareEngineering)\Lab8_3>
```

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

```
PS C:\CP353004(SoftwareEngineering)> mkdir Lab8_4
```

Directory: C:\CP353004(SoftwareEngineering)

Mode	LastWriteTime	Length	Name
d----	27/1/2568 20:03		Lab8_4

```

ithub.com/docker/getting-started.git
Cloning into 'getting-started'...
remote: Enumerating objects: 980, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 980 (delta 5), reused 1 (delta 1), pack-reused 971 (from 2)
Receiving objects: 100% (980/980), 5.28 MiB | 5.14 MiB/s, done. Resolving deltas: 100% (523/523), done.
PS C:\CP353004(SoftwareEngineering)\Lab8_4>

```

The screenshot shows the Windows File Explorer interface. The address bar displays the path C:\CP353004(SoftwareEngineering). The main pane shows a list of folders: Lab8_1, Lab8_2, Lab8_3, and Lab8_4. The 'Lab8_4' folder is selected. The 'getting-started' folder is visible inside Lab8_4.

Name	Date modified	Type
Lab8_1	27/1/2568 15:55	File folder
Lab8_2	27/1/2568 17:10	File folder
Lab8_3	27/1/2568 19:48	File folder
Lab8_4	27/1/2568 20:03	File folder

The screenshot also shows the 'getting-started' folder inside the 'Lab8_4' directory.

Name	Date modified	Type
getting-started	27/1/2568 20:04	File folder

Lab Worksheet

The image shows two screenshots of a Windows File Explorer window. The top screenshot displays the 'getting-started' directory, and the bottom screenshot displays the 'app' subdirectory.

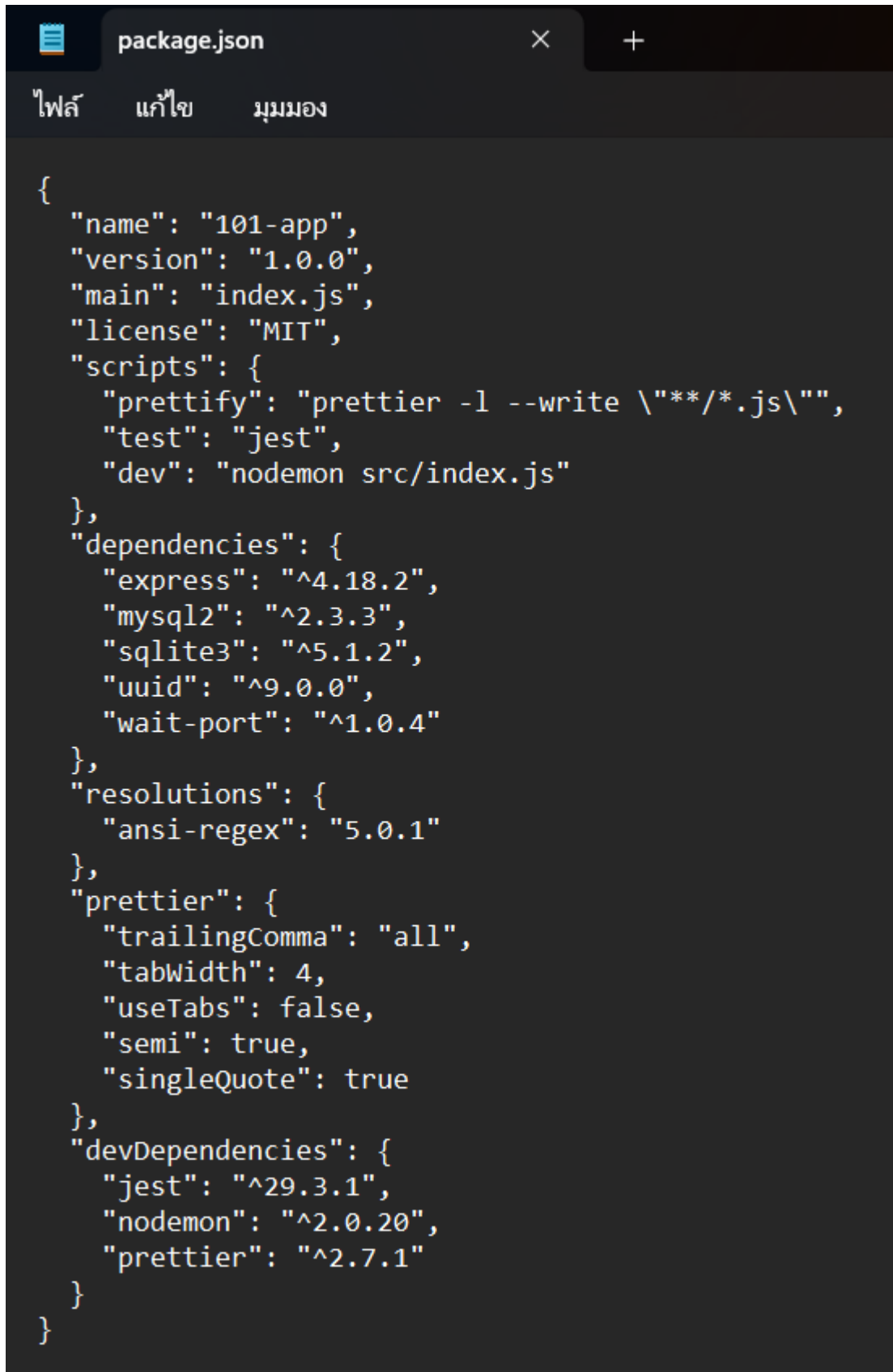
Top Screenshot: 'getting-started' directory

Name	Date modified	Type	Size
.github	27/1/2568 20:04	File folder	
app	27/1/2568 20:04	File folder	
docs	27/1/2568 20:04	File folder	
.dockerignore	27/1/2568 20:04	DOCKERIGNORE F...	1
.gitignore	27/1/2568 20:04	Git Ignore Source ...	1
build.sh	27/1/2568 20:04	sh_auto_file	1
docker-compose	27/1/2568 20:04	Yaml Source File	1
Dockerfile	27/1/2568 20:04	File	2
LICENSE	27/1/2568 20:04	File	12
mkdocs	27/1/2568 20:04	Yaml Source File	3
README	27/1/2568 20:04	Markdown Source ...	2
requirements	27/1/2568 20:04	เอกสารข้อความ	1

Bottom Screenshot: 'app' subdirectory

Name	Date modified	Type	Size
spec	27/1/2568 20:04	File folder	
src	27/1/2568 20:04	File folder	
package	27/1/2568 20:04	JSON Source File	1
yarn.lock	27/1/2568 20:04	LOCK File	148

Lab Worksheet



```
{
  "name": "101-app",
  "version": "1.0.0",
  "main": "index.js",
  "license": "MIT",
  "scripts": {
    "prettify": "prettier -l --write \"**/*.js\"",
    "test": "jest",
    "dev": "nodemon src/index.js"
  },
  "dependencies": {
    "express": "^4.18.2",
    "mysql2": "^2.3.3",
    "sqlite3": "^5.1.2",
    "uuid": "^9.0.0",
    "wait-port": "^1.0.4"
  },
  "resolutions": {
    "ansi-regex": "5.0.1"
  },
  "prettier": {
    "trailingComma": "all",
    "tabWidth": 4,
    "useTabs": false,
    "semi": true,
    "singleQuote": true
  },
  "devDependencies": {
    "jest": "^29.3.1",
    "nodemon": "^2.0.20",
    "prettier": "^2.7.1"
  }
}
```

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

```
PS C:\CP353004(SoftwareEngineering)\Lab8_4> cd getting-started
PS C:\CP353004(SoftwareEngineering)\Lab8_4\getting-started>cd app
PS C:\CP353004(SoftwareEngineering)\Lab8_4\getting-started\app> docker build -t myapp_6533801977 -f Dockerfile.txt .
[+] Building 29.4s (10/10) FINISHED   docker:desktop-linux
=> [internal] load build definition from Dockerfile. 0.0s
=> => transferring dockerfile: 160B 0.0s
=> [internal] load metadata for docker.io/library/node 4.0s
=> [auth] library/node:pull token for registry-1.docker.io 0.0s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [1/4] FROM docker.io/library/node:18-alpine@sha256:6504e29600c8d5213b52cda8003444b 6.4s
=> => resolve docker.io/library/node:18-alpine@sha256:6504e29600c8d5213b52cda8003444b 0.0s
=> => sha256:6504e29600c8d5213b52cda8003444b / 444B 0.3s
=> => sha256:37892ffbfcaa871a10f8140.01MB / 40.01MB 5.1s
=> => sha256:5650d6de56fd0bb419872b8 1.26MB / 1.26MB 1.6s
=> => sha256:1f3e46996e2966e4faa5846 3.64MB / 3.64MB 5.2s
=> => extracting sha256:1f3e46996e2966e4faa584656e7 0.1s
=> => extracting sha256:37892ffbfcaa871a10f813803949 0.9s
=> => extracting sha256:5650d6de56fd0bb419872b876ac1 0.0s
```

Lab Worksheet

```

=> => extracting sha256:6504e29600c8d5213b52cda80037 0.0s
=> [internal] load build context 0.4s
=> => transferring context: 4.62MB 0.4s
=> [2/4] WORKDIR /app 0.3s
=> [3/4] COPY . . 0.1s
=> [4/4] RUN yarn install --production 12.6s
=> exporting to image 5.8s
=> => exporting layers 3.5s
=> => exporting config sha256:63a9ed34a06f05d34ea86c 0.0s
=> => exporting attestation manifest sha256:73c79f7b 0.0s
=> => exporting manifest list sha256:b9fa9a91b6ce972 0.0s
=> => naming to docker.io/library/myapp_6533801977:l 0.0s
=> => unpacking to docker.io/library/myapp_653380197 2.2s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/xe0ubuc17i9u5dgabzieo8ujf
PS C:\CP353004(SoftwareEngineering)\Lab8_4\getting-started\app>

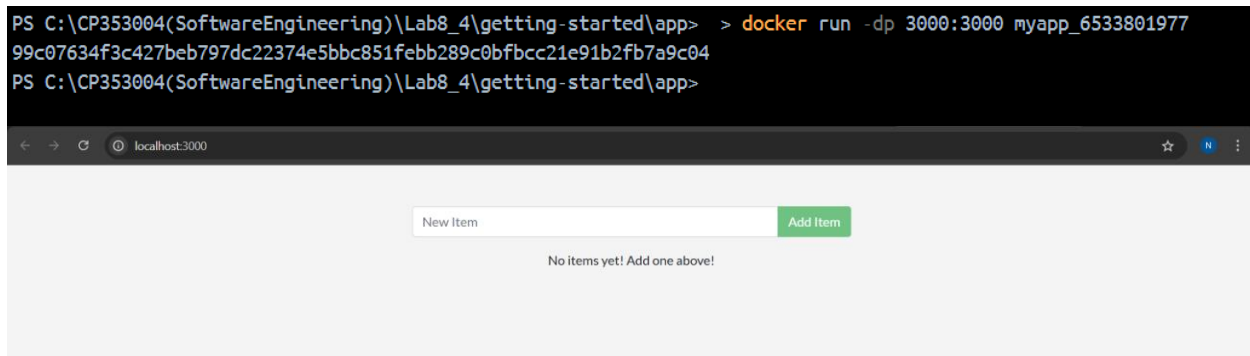
```

6. ทำการ Start ตัว Container ของแอปพลิเคชันที่สร้างขึ้น โดยใช้คำสั่ง

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

```
<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

Lab Worksheet

9. ทำการ Build Docker image โดยใช้คำสั่งเดียวกันกับข้อ 5
10. Start และรัน Container ตัวใหม่ โดยใช้คำสั่งเดียวกันกับข้อ 6

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

```
return (
  <React.Fragment>
    <AddItemForm onNewItem={onNewItem} />
    {items.length === 0 && (
      <p className="text-center">There is no TODO item. Please add one to the list. By ณัฐภัทร ดร.วัฒนาวุฒ</p>
    )}
    {items.map(item => (
      <ItemDisplay
        item={item}
        key={item.id}
        onItemUpdate={onItemUpdate}
        onItemRemoval={onItemRemoval}
      />
    ))}
  </React.Fragment>
);
}
```

View build details: [docker-desktop://dashboard/build/desktop-linux/desktop-linux/z1tz3r6ncxziv3hc5h9vxukjf](#)
 PS C:\CP353004(SoftwareEngineering)\Lab8_4\getting-started\app> docker run -dp 3000:3000 myapp_653380197711e9eaab11d883779a1): Bind for 0.0.0.0:3000 failed: port is already allocated.

(1) Error ที่เกิดขึ้นหมายความว่าอย่างไร และเกิดขึ้นเพราะอะไร

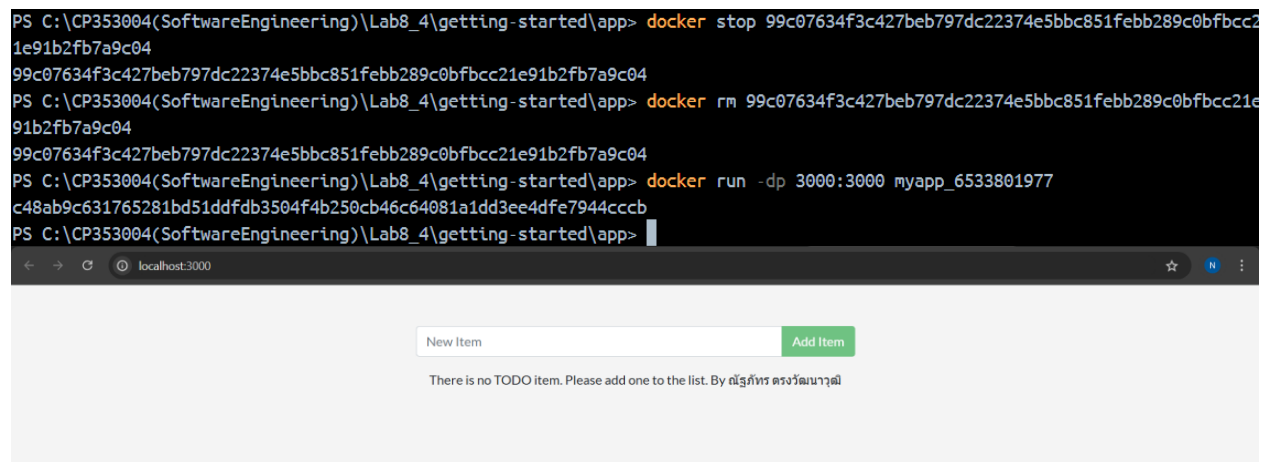
ข้อผิดพลาด "Bind for 0.0.0.0:3000 failed: port is already allocated" หมายความว่า Port 3000 ที่พยายามใช้งานใน Docker ถูกใช้งานอยู่แล้วจากโปรแกรมหรือ container อื่นบนเครื่องทำให้ Docker ไม่สามารถใช้ port นี้ได้

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

Lab Worksheet

- i. ไปที่หน้าต่าง Containers
 - ii. เลือกไอคอนถึงขยะในแถวของ 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

[Check point#12] Capture หน้าจอที่แสดงผล Admin password

Lab Worksheet

```
051df97b5113491a8987bfa379200383
```

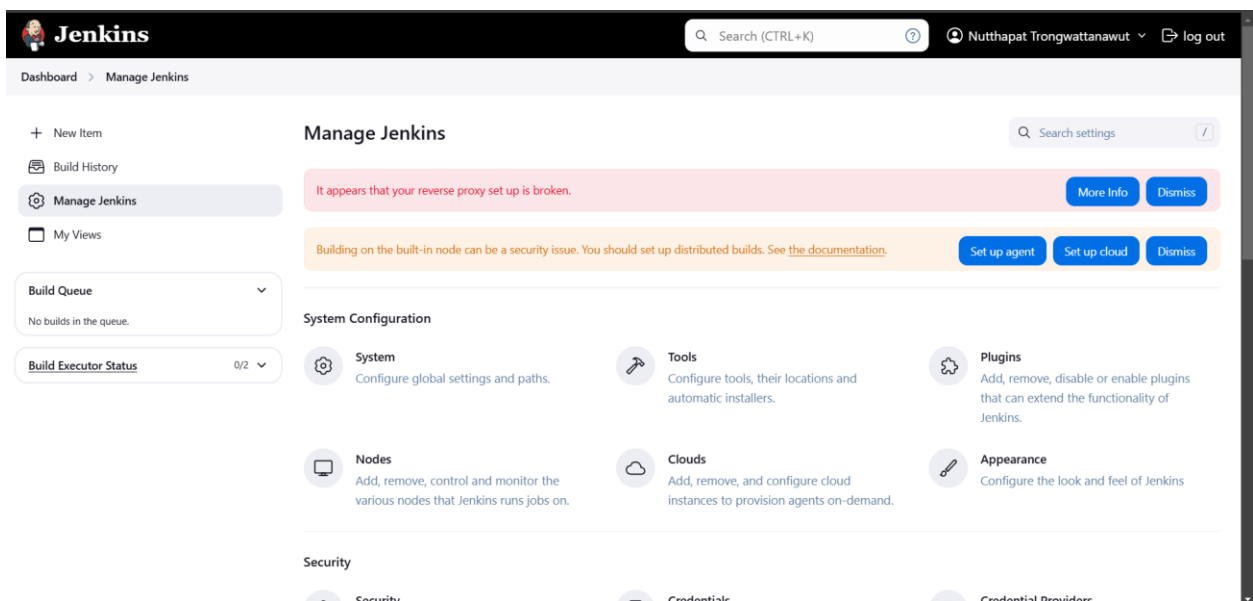
```
This may also be found at: /var/jenkins_home/secrets/initialAdminPassword
```

```
*****
*****
*****
```

```
2025-01-27 15:36:43.868+0000 [id=53] INFO jenkins.InitReactorRunner$1#onAttained: Completed initialization
2025-01-27 15:36:43.935+0000 [id=25] INFO hudson.lifecycle.Lifecycle#onReady: Jenkins is fully up and running
```

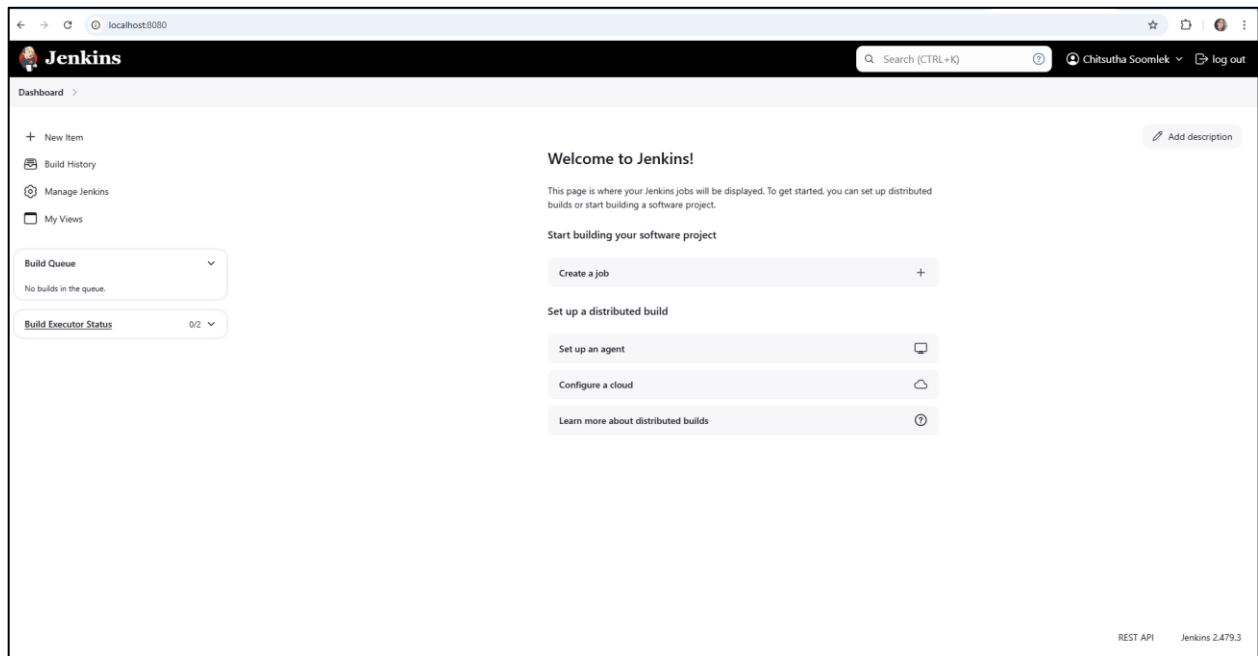
4. เมื่อได้รับการยืนยันว่า Jenkins is fully up and running ให้เปิดบราวเซอร์ และป้อนที่อยู่เป็น localhost:8080
5. ทำการ Unlock Jenkins ด้วยรหัสผ่านที่ได้ในข้อที่ 3
6. สร้าง Admin User โดยใช้ username เป็นชื่อจริงของนักศึกษาพร้อมรหัสสี่ตัวท้าย เช่น somsri_3062

[Check point#13] Capture หน้าจอที่แสดงผลการตั้งค่า

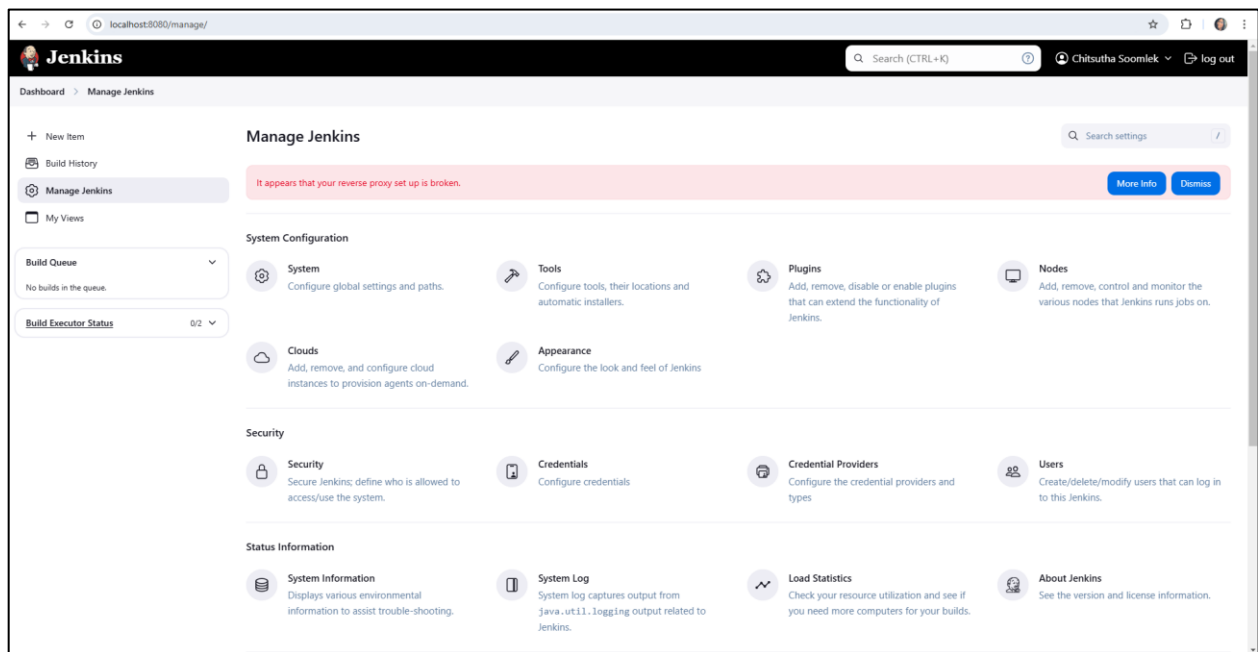


7. กำหนด Jenkins URL เป็น <http://localhost:8080/lab8>
8. เมื่อติดตั้งเรียบร้อยแล้วจะพบหน้าจอ Dashboard ดังแสดงในภาพ

Lab Worksheet

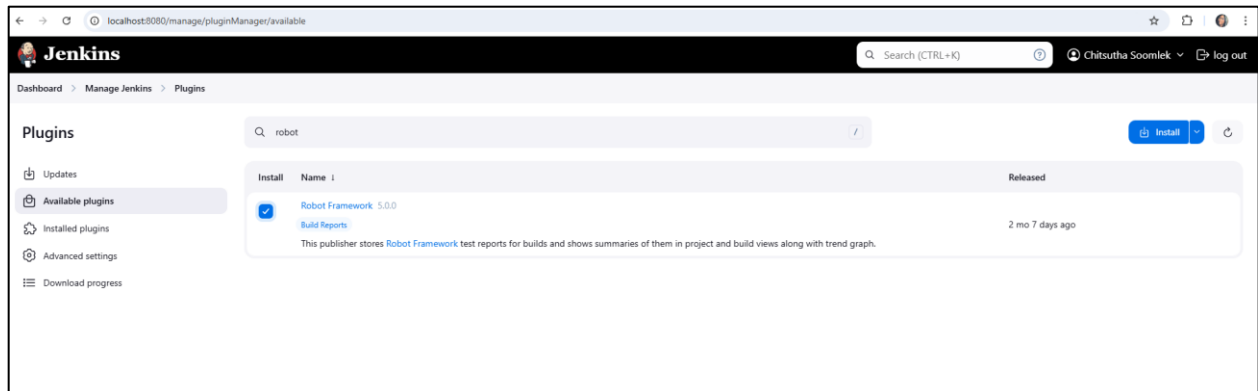


9. เลือก Manage Jenkins แล้วไปที่เมนู Plugins

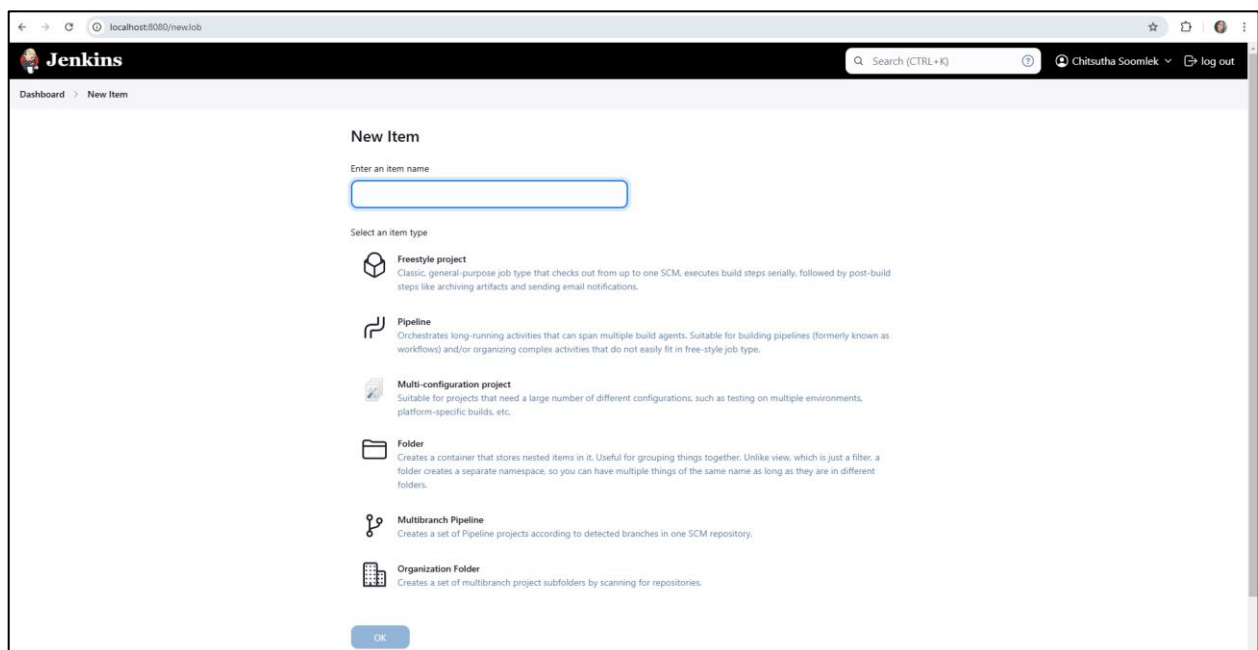


Lab Worksheet

10. ไปที่เมนู Available plugins แล้วเลือกติดตั้ง Robotframework เพิ่มเติม



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



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

Plain text [Preview](#)

☐ Discard old builds ?

☒ GitHub project

Project url ?

Advanced ▾

☐ This project is parameterized ?

☐ Throttle builds ?

☐ Execute concurrent builds if necessary ?

Advanced ▾

localhost:8080/job/UAT/configure

Dashboard > UAT > Configuration

Configure

General

Source Code Management

Build Triggers

Build Environment

Build Steps

Post-build Actions

Git ?

Build Triggers

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

☐ Build after other projects are built ?

☒ Build periodically ?

Schedule ?

Would last have run at Monday, January 27, 2025 at 4:05:06 PM Coordinated Universal Time; would next run at Monday, January 27, 2025 at 4:20:06 PM Coordinated Universal Time.

☐ GitHub hook trigger for GITScm polling ?

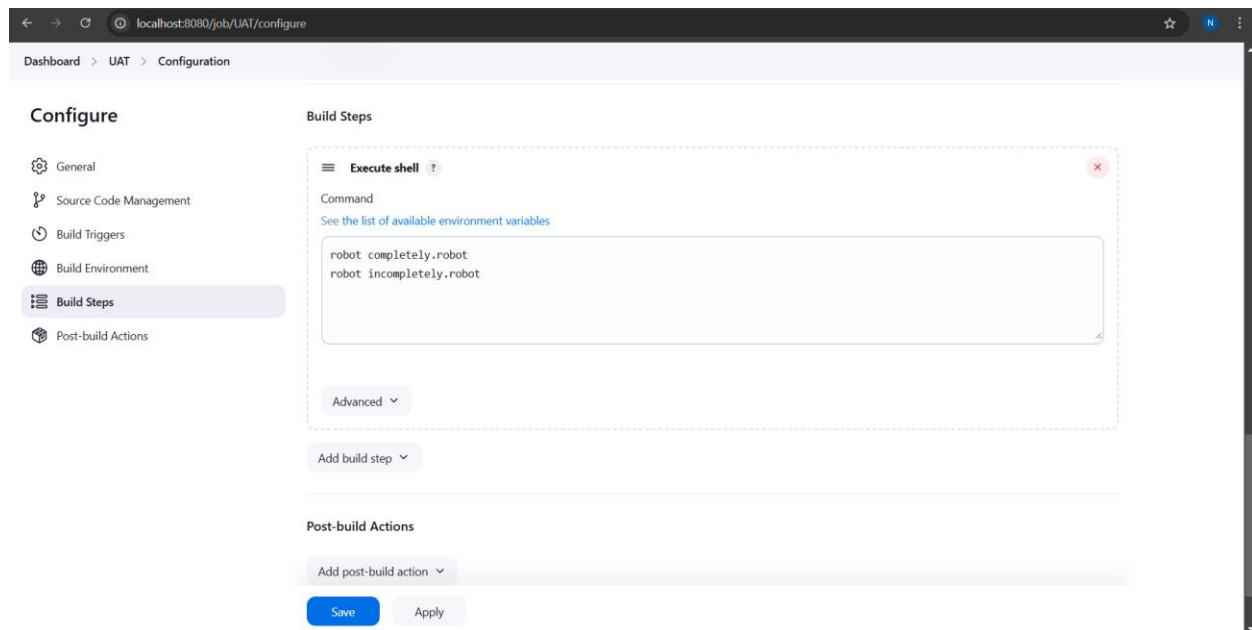
☐ Poll SCM ?

Build Environment

☐ Delete workspace before build starts

Save Apply

Lab Worksheet



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

`robot completely.robot`

`robot incompletely.robot`

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

Lab Worksheet

✖

Console Output

Download

Copy

View as plain text

```

Started by user Nutthapat Trongwattanawut
Running as SYSTEM
Building in workspace /var/jenkins_home/workspace/UAT
[UAT] $ /bin/sh -xe /tmp/jenkins6460213603666264359.sh
+ robot completely.robot
/tmp/jenkins6460213603666264359.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

```

Build Time Trend

S	Build ↑	Time Since	Duration	
✖	#147	1.9 sec	5 ms	🔍
✖	#146	2.8 sec	10 ms	🔍
✖	#145	7 min 3 sec	6 ms	🔍
✖	#144	7 min 11 sec	8 ms	🔍
✖	#143	22 min	5 ms	🔍
✖	#142	22 min	6 ms	🔍
✖	#141	22 min	5 ms	🔍
✖	#140	22 min	5 ms	🔍
✖	#139	22 min	6 ms	🔍
✖	#138	22 min	6 ms	🔍
✖	#137	22 min	5 ms	🔍
✖	#136	22 min	6 ms	🔍
✖	#135	22 min	8 ms	🔍

