

ชื่อ-นามสกุล คณิศร ร้อยศรี รหัสนักศึกษา 653380121-0 Section 2

Lab#8 – Software Deployment Using Docker

วัตถุประสงค์การเรียนรู้

- ผู้เรียนสามารถอธิบายเกี่ยวกับ Software deployment ได้
- ผู้เรียนสามารถสร้างและรัน Container จาก Docker image ได้
- ผู้เรียนสามารถสร้าง Docker files และ Docker images ได้
- ผู้เรียนสามารถนำซอฟต์แวร์ที่พัฒนาขึ้นให้สามารถรันบนสภาพแวดล้อมเดียวกันและทำงานร่วมกันกับสมาชิกในทีมพัฒนาซอฟต์แวร์ผ่าน Docker hub ได้
- ผู้เรียนสามารถเริ่มต้นใช้งาน Jenkins เพื่อสร้าง Pipeline ในการ Deploy งานได้

Pre-requisite

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

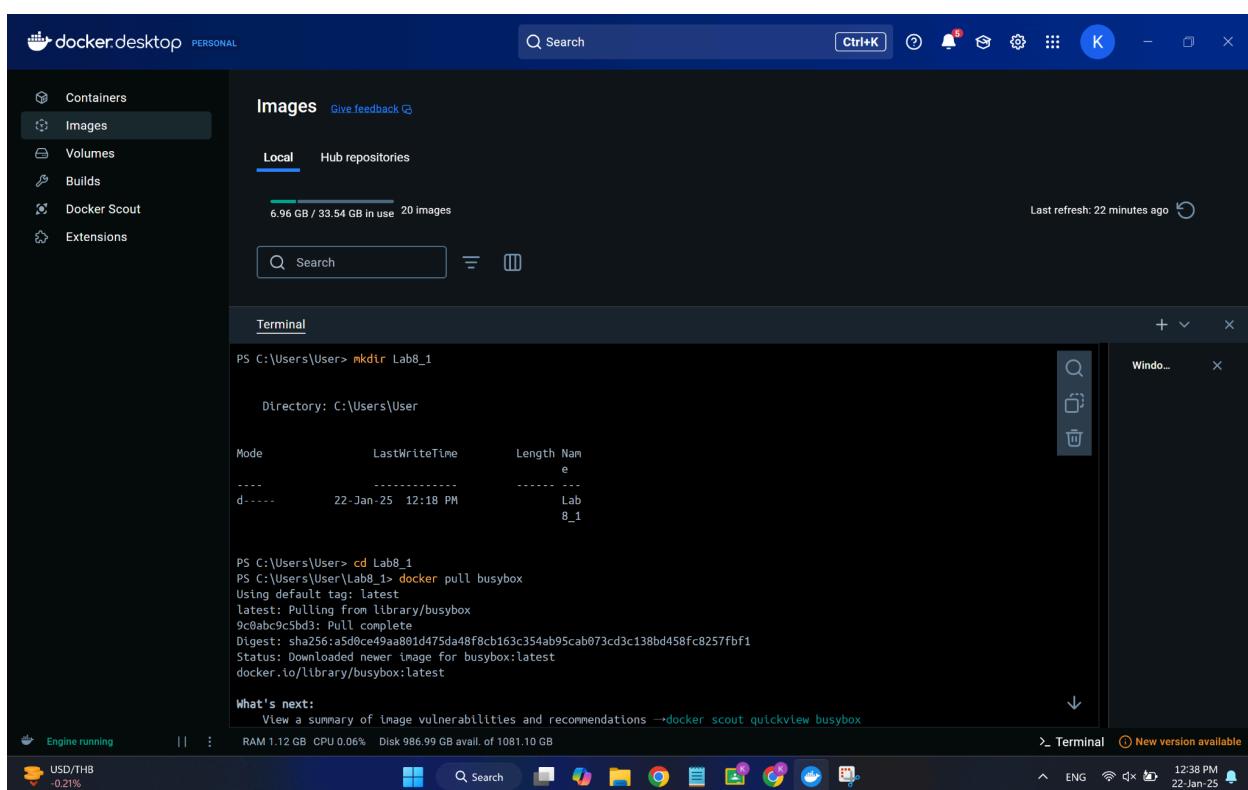
แบบฝึกปฏิบัติที่ 8.1 Hello world - รัน Container จาก Docker image

- เปิดใช้งาน Docker desktop และ Login ด้วย Username และ Password ที่ลงทะเบียนกับ Docker Hub เอาไว้
- เปิด Command line หรือ Terminal บน Docker Desktop จากนั้นสร้าง Directory ชื่อ Lab8_1

Lab Worksheet

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

```
PS C:\Users\User\Lab8_1> docker images
REPOSITORY          IMAGE ID      CREATED     SIZE
base-hadoop          7f5843bae89  7 weeks ago  2.34GB
<none>              edcc9d10c862  7 weeks ago  2.34GB
all_ai_lab_image     c500891262e8  3 months ago  7.65GB
<none>              bbb2a1beaa41  3 months ago  7.65GB
genai-web            d398d7e1f6b8  3 months ago  503MB
busybox              af4709625109  3 months ago  4.27MB
```

- (1) สิงทืออยู่ภายในได้คอลัมน์ Repository คืออะไร
ชื่อของ docker image ที่เก็บอยู่ในเครื่องที่ pull มาจาก docker registry
- (2) Tag ที่ใช้บ่งบอกถึงอะไร
บ่งบอกถึง version image ของ Repository นั้นๆ latest คือ version ล่าสุด

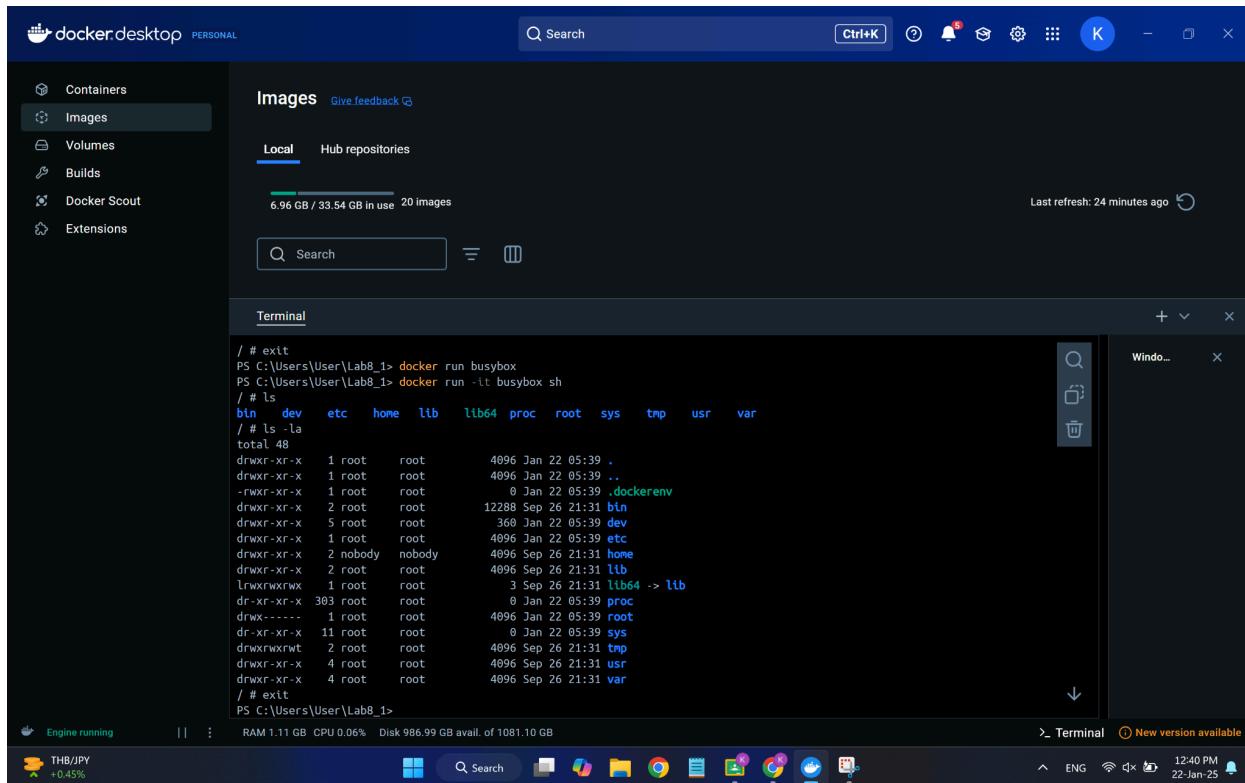
5. ป้อนคำสั่ง \$ docker run busybox
6. ป้อนคำสั่ง \$ docker run -it busybox sh
7. ป้อนคำสั่ง ls
8. ป้อนคำสั่ง ls -la
9. ป้อนคำสั่ง exit
10. ป้อนคำสั่ง \$ docker run busybox echo "Hello ชื่อและ
นามสกุลของนักศึกษา from busybox"
11. ป้อนคำสั่ง \$ docker ps -a

[Check point#2] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่
เกี่ยวข้อง) แสดงผลลัพธ์ที่ได้ตั้งแต่ขั้นตอนที่ 6-12 พร้อมกับตอบคำถาม
ต่อไปนี้

CP353004/SC313 004 Software Engineering (2/2567)

ผศ. ดร. ชิตสุชา สุ่มเล็ก

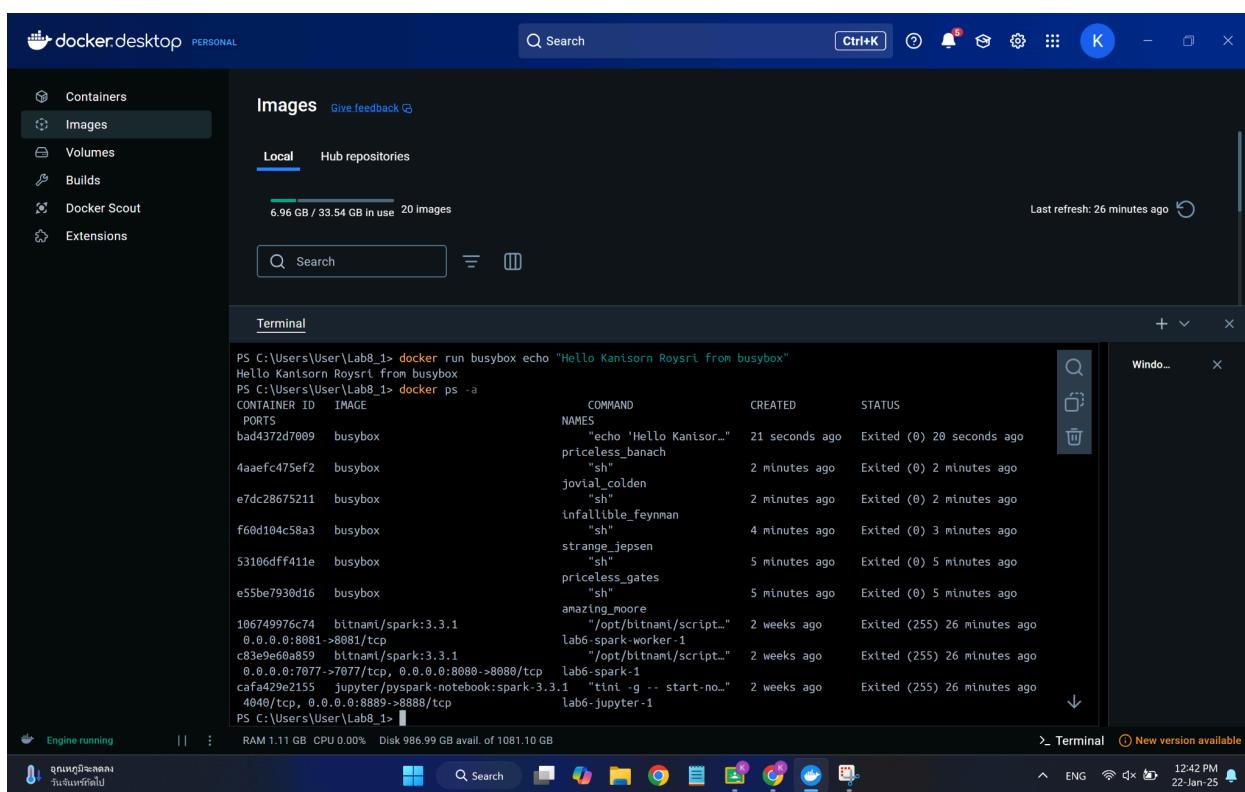
Lab Worksheet



Screenshot of Docker Desktop showing a terminal session with busybox. The terminal output is as follows:

```
/ # exit
PS C:\Users\User\Lab8_1> docker run busybox
PS C:\Users\User\Lab8_1> docker run -it busybox sh
/ # ls
bin dev etc home lib lib64 proc root sys tmp usr var
/ # ls -la
total 48
drwxr-xr-x 1 root root 4096 Jan 22 05:39 .
drwxr-xr-x 1 root root 4096 Jan 22 05:39 ..
-rwxr-xr-x 1 root root 6 Jan 22 05:39 .dockerenv
drwxr-xr-x 2 root root 12288 Sep 26 21:31 bin
drwxr-xr-x 5 root root 368 Jan 22 05:39 dev
drwxr-xr-x 1 root root 4096 Jan 22 05:39 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 303 root root 6 Jan 22 05:39 proc
drwx----- 1 root root 4096 Jan 22 05:39 root
dr-xr-xr-x 11 root root 6 Jan 22 05:39 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
PS C:\Users\User\Lab8_1>
```

The Docker Desktop interface shows the terminal window, the Images section with 20 images, and the status bar indicating the engine is running.



Screenshot of Docker Desktop showing a terminal session with multiple busybox containers. The terminal output is as follows:

```
PS C:\Users\User\Lab8_1> docker run busybox echo "Hello Kantsorn Roysri from busybox"
Hello Kantsorn Roysri from busybox
PS C:\Users\User\Lab8_1> docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS
NAMES
bad4372d7009 busybox "echo 'Hello Kanisor..." 21 seconds ago Exited (0) 20 seconds ago
priceless_banach
4aaefc475ef2 busybox "sh"
jovial_colden
e7dc28675211 busybox "sh"
infallible_feynman
f60d104c58a3 busybox "sh"
strange_jepsen
53106dff411e busybox "sh"
priceless_gates
e55be7930d16 busybox "sh"
amazing_moore
106749976c74 bitnami/spark:3.3.1 "/opt/bitnami/script..."
0.0.0.0:8081->8081/tcp lab6-spark-worker-1
c83e9e60a859 bitnami/spark:3.3.1 "/opt/bitnami/script..."
0.0.0.0:7077->7077/tcp, 0.0.0.0:8080->8080/tcp lab6-spark-1
caf3429e2155 jupyter/pyspark-notebook:spark-3.3.1 "tini -g -- start-no..."
4040/tcp, 0.0.0.0:8889->8888/tcp lab6-jupyter-1
PS C:\Users\User\Lab8_1>
```

The Docker Desktop interface shows the terminal window, the Images section with 20 images, and the status bar indicating the engine is running.

(1) เมื่อใช้ option -it ในคำสั่ง run ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสั้นๆ

-i เพื่อใช้งาน input(stdin) ให้สามารถส่งข้อมูลหรือคำสั่งไปยัง container ได้โดยตรง

-t เพื่อสร้าง terminal จำลอง

ดังนั้น -it เพื่อเปิดใช้งาน terminal ที่สามารถโต้ตอบกับ container

(2) คอลัมน์ STATUS จากการรันคำสั่ง docker ps -a แสดงถึงข้อมูลอะไร

แสดงถึงสถานะการทำงานของแต่ละ process ว่าจบการทำงานแล้วหรือยัง ถ้าจบแล้ว จะแสดงเวลาที่จบการทำงาน

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

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

CONTAINER ID	IMAGE	NAMES	COMMAND	CREATED	STATUS
4aaefc475ef2	busybox	jovial_colden	"sh"	2 hours ago	Exited (0) 2 hours ago
e7dc28675211	busybox	infallible_feynman	"sh"	2 hours ago	Exited (0) 2 hours ago
f60d104c58a3	busybox	strange_jepsen	"sh"	2 hours ago	Exited (0) 2 hours ago
53106dff411e	busybox	priceless_gates	"sh"	2 hours ago	Exited (0) 2 hours ago
e55be7930d16	busybox	amazing_moore	"sh"	2 hours ago	Exited (0) 2 hours ago

```
PS C:\Users\User\Lab8_1> docker ps -a
```

```
CONTAINER ID IMAGE NAMES COMMAND CREATED STATUS
```

```
4aaefc475ef2 busybox jovial_colden "sh" 2 hours ago Exited (0) 2 hours ago
```

```
e7dc28675211 busybox infallible_feynman "sh" 2 hours ago Exited (0) 2 hours ago
```

```
f60d104c58a3 busybox strange_jepsen "sh" 2 hours ago Exited (0) 2 hours ago
```

```
53106dff411e busybox priceless_gates "sh" 2 hours ago Exited (0) 2 hours ago
```

```
e55be7930d16 busybox amazing_moore "sh" 2 hours ago Exited (0) 2 hours ago
```

```
PS C:\Users\User\Lab8_1> docker rm 4aaefc475ef2
```

```
4aaefc475ef2
```

```
PS C:\Users\User\Lab8_1> 
```

CONTAINER ID	IMAGE	NAMES	COMMAND	CREATED	STATUS
e7dc28675211	busybox	infallible_feynman	"sh"	2 hours ago	Exited (0) 2 hours ago
f60d104c58a3	busybox	strange_jepsen	"sh"	2 hours ago	Exited (0) 2 hours ago
53106dff411e	busybox	priceless_gates	"sh"	2 hours ago	Exited (0) 2 hours ago
e55be7930d16	busybox	amazing_moore	"sh"	2 hours ago	Exited (0) 2 hours ago

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 บันหน้าต่าง 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
```

Lab Worksheet

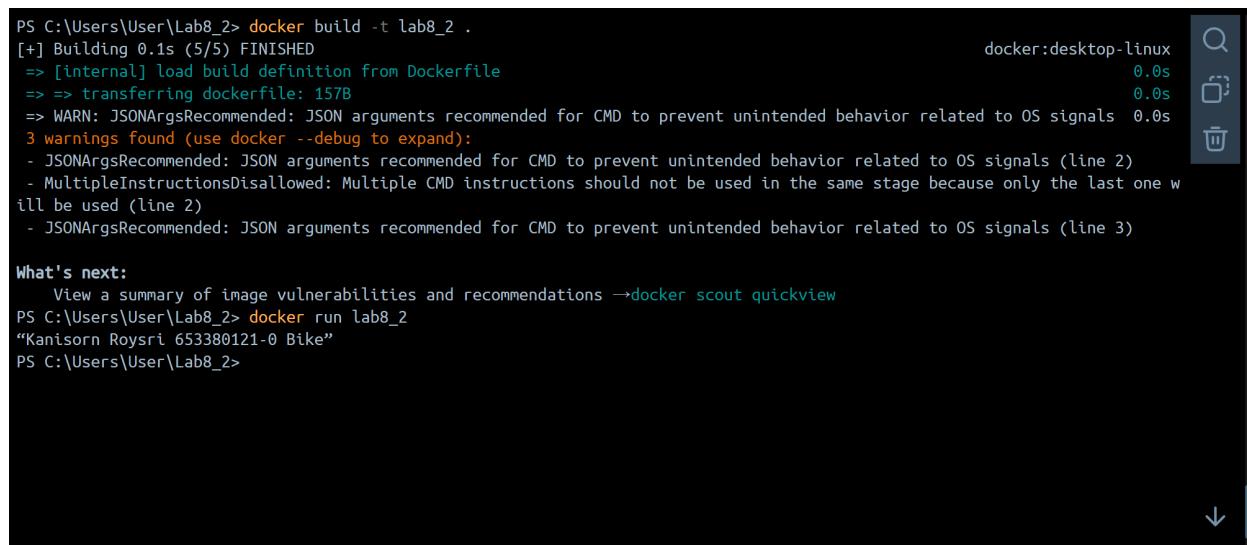
แล้วใช้ Text Editor ในการใส่เนื้อหาแทน

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



```
PS C:\Users\User\Lab8_2> docker build -t lab8_2 .
[+] Building 0.1s (5/5) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 157B
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals 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 2)
- 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)

What's next:
View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\Users\User\Lab8_2> docker run lab8_2
"Kanisorn Roysri 653380121-0 Bike"
PS C:\Users\User\Lab8_2>
```

(1) คำสั่งที่ใช้ในการ run คือ

`docker run lab8_2`

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

เพื่อใช้ในการกำหนดชื่อให้แก่ image ดังกล่าว รวมถึงการกำหนด tag หากไม่ใช่ -t docker จะสร้าง image โดยไม่ตั้งชื่อและ tag ที่ขัดเจน ทำให้ต้องอ้างอิงผ่าน image ID แทน

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. ย้ายตัวແහນໆປ່ຈຸບັນໄປທີ Lab8_3 ເພື່ອໃຫ້ເປັນ Working directory
4. ສ້າງ Dockerfile.swp ໄວໃນ Working directory

ສໍາຮັບເຄື່ອງທີ່ໃຫ້ຮັບປົງປົດການວິນໂດວັ້ນ ບັນທຶກຄໍາສັ່ງຕ່ອໄປນີ້ລັງໃນໄຟລ໌ ໂດຍໃຫ້ Text Editor ທີ່ມີ

FROM busybox

CMD echo “Hi there. My work is done. You can run them from my Docker image.”

CMD echo “ຊື່-ນາມສຸກລ ຮ້າສັນກີກຂາ”

ສໍາຮັບເຄື່ອງທີ່ໃຫ້ຮັບປົງປົດການ MacOS ອີ່ງ Linux ບັນໜ້າຕ່າງ Terminal ແລະປ່ອນຄໍາສັ່ງຕ່ອໄປນີ້

```
$ cat > Dockerfile << EOF
```

FROM busybox

CMD echo “Hi there. My work is done. You can run them from my Docker image.”

CMD echo “ຊື່-ນາມສຸກລ ຮ້າສັນກີກຂາ”

EOF

ຫຼືໃຫ້ຄໍາສັ່ງ

```
$ touch Dockerfile
```

ແລ້ວໃຫ້ Text Editor ໃນການໄສ່ເນື້ອຫາແນ

7. ทำการ Build Docker image ທີ່ສ້າງຂຶ້ນດ້ວຍຄໍາສັ່ງຕ່ອໄປນີ້

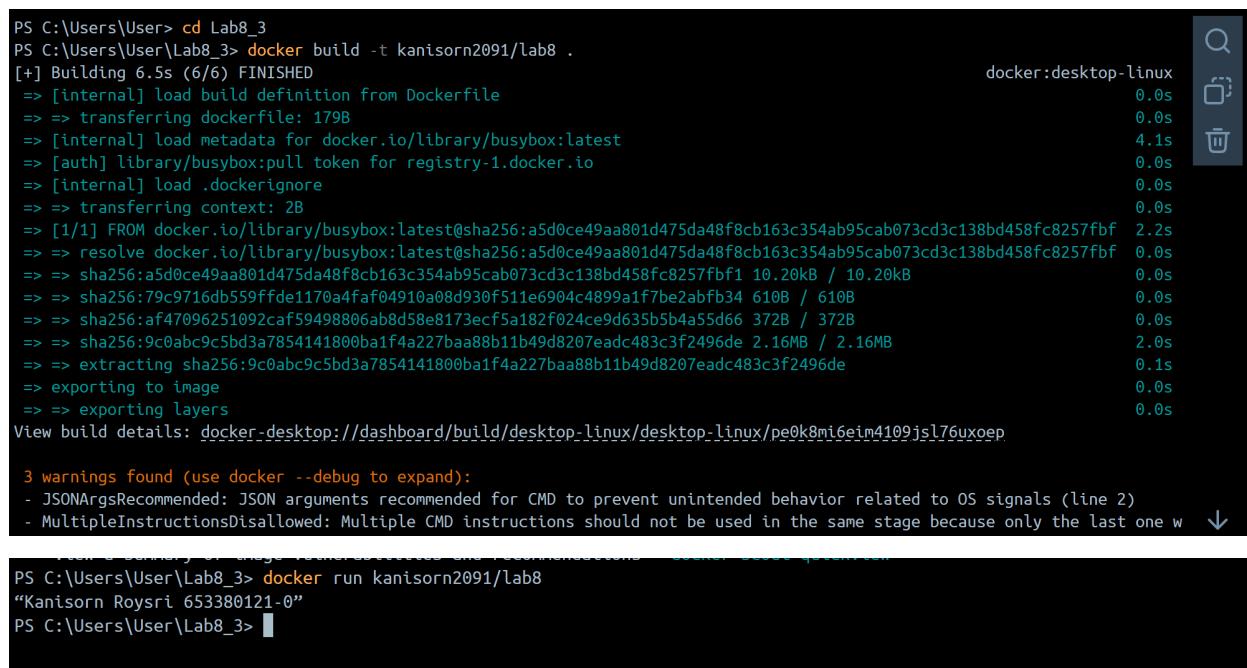
Lab Worksheet

\$ 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



```
PS C:\Users\User> cd Lab8_3
PS C:\Users\User\Lab8_3> docker build -t kanisorn2091/lab8 .
[+] Building 6.5s (6/6) FINISHED
=> [internal] load definition from Dockerfile
=> => transferring dockerfile: 179B
=> [internal] load metadata for docker.io/library/busybox:latest
=> [auth] library/busybox:pull token for registry-1.docker.io
=> [internal] load .dockerrcignore
=> => transferring context: 2B
=> [1/1] FROM docker.io/library/busybox:latest@sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138bd458fc8257fbf 2.2s
=> => resolve docker.io/library/busybox:latest@sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138bd458fc8257fbf 0.0s
=> => sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138bd458fc8257fbf1 10.20kB / 10.20kB 0.0s
=> => sha256:79c9716db559ffde1170a4faf04910a08d930f511e6904c4899a1f7be2abfb34 610B / 610B 0.0s
=> => sha256:af47096251092caf59498806ab8d58e8173ecf5a182f024ce9d635b5b4a55d66 372B / 372B 0.0s
=> => sha256:9c0abc9c5bd3a7854141800ba1f4a227baa88b11b49d8207eadc483c3f2496de 2.16MB / 2.16MB 2.0s
=> => extracting sha256:9c0abc9c5bd3a7854141800ba1f4a227baa88b11b49d8207eadc483c3f2496de 0.1s
=> exporting to image 0.0s
=> => exporting layers 0.0s
View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/pe0k8mi6eim4109jsl76uxoep

3 warnings found (use docker --debug to expand):
- JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 2)
- MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the same stage because only the last one w ↓
```

PS C:\Users\User\Lab8_3> docker run kanisorn2091/lab8
"Kanisorn Roysri 653380121-0"
PS C:\Users\User\Lab8_3>

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 และป้อน Username และ Password ตามที่ระบุใน Command prompt หรือใช้คำสั่ง

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

CP353004/SC313 004 Software Engineering (2/2567)

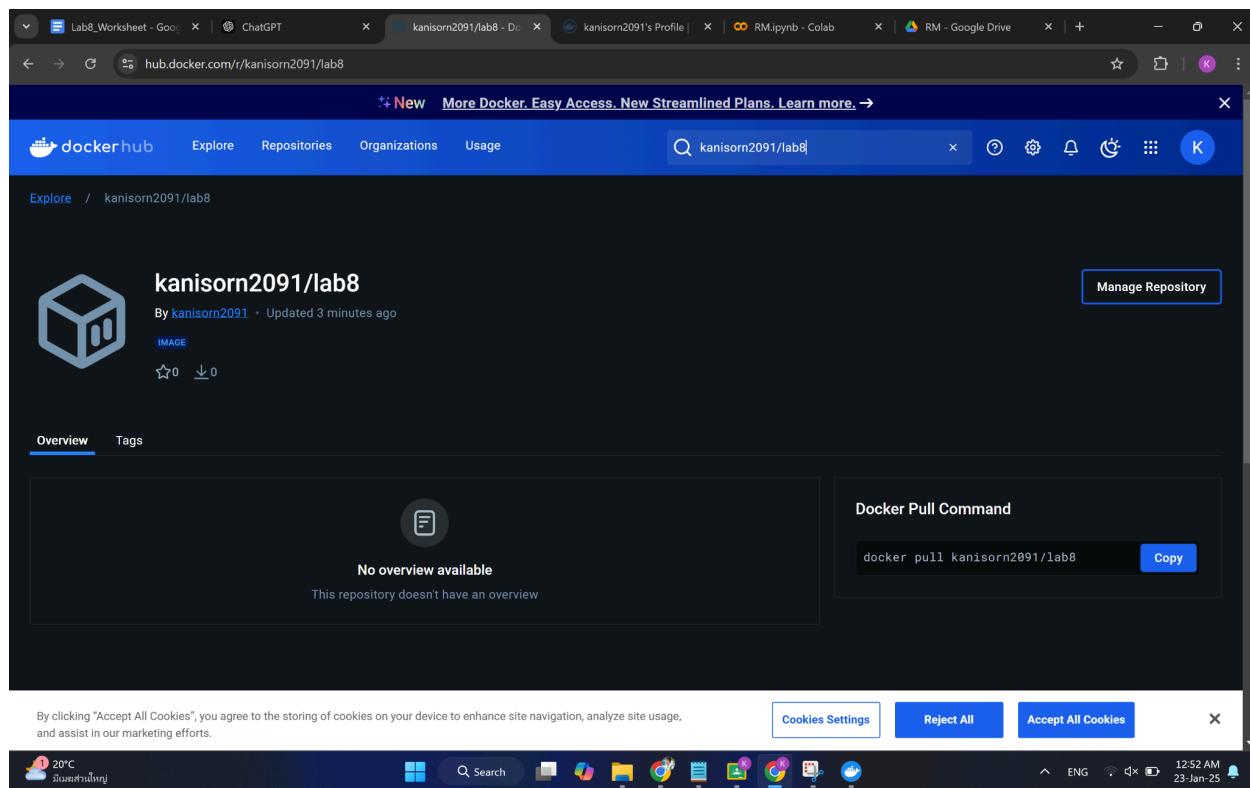
ผศ. ดร. ชิตสุชา สุ่มเล็ก

Lab Worksheet

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:\Users\User\Lab8_3> docker push kanisorn2091/lab8
Using default tag: latest
The push refers to repository [docker.io/kanisorn2091/lab8]
59654b79daad: Mounted from library/busybox
latest: digest: sha256:4575f5d6df7087b76c3ed2cec3bb696b3d7cf22ccb4739ed88dc9f9908987202 size: 527
PS C:\Users\User\Lab8_3>
```



แบบฝึกปฏิบัติที่ 8.4: การ Build และ Update แอปพลิเคชันจาก Container image และการ Update แอปพลิเคชัน

1. เปิด Command line หรือ Terminal จากนั้นสร้าง Directory ชื่อ Lab8_4

Lab Worksheet

- ทำการ Clone ซอฟต์แวร์โค้ดของเว็บแอปพลิเคชันจาก GitHub repository <https://github.com/docker/getting-started.git> ลงใน Directory ที่สร้างขึ้น โดยใช้คำสั่ง

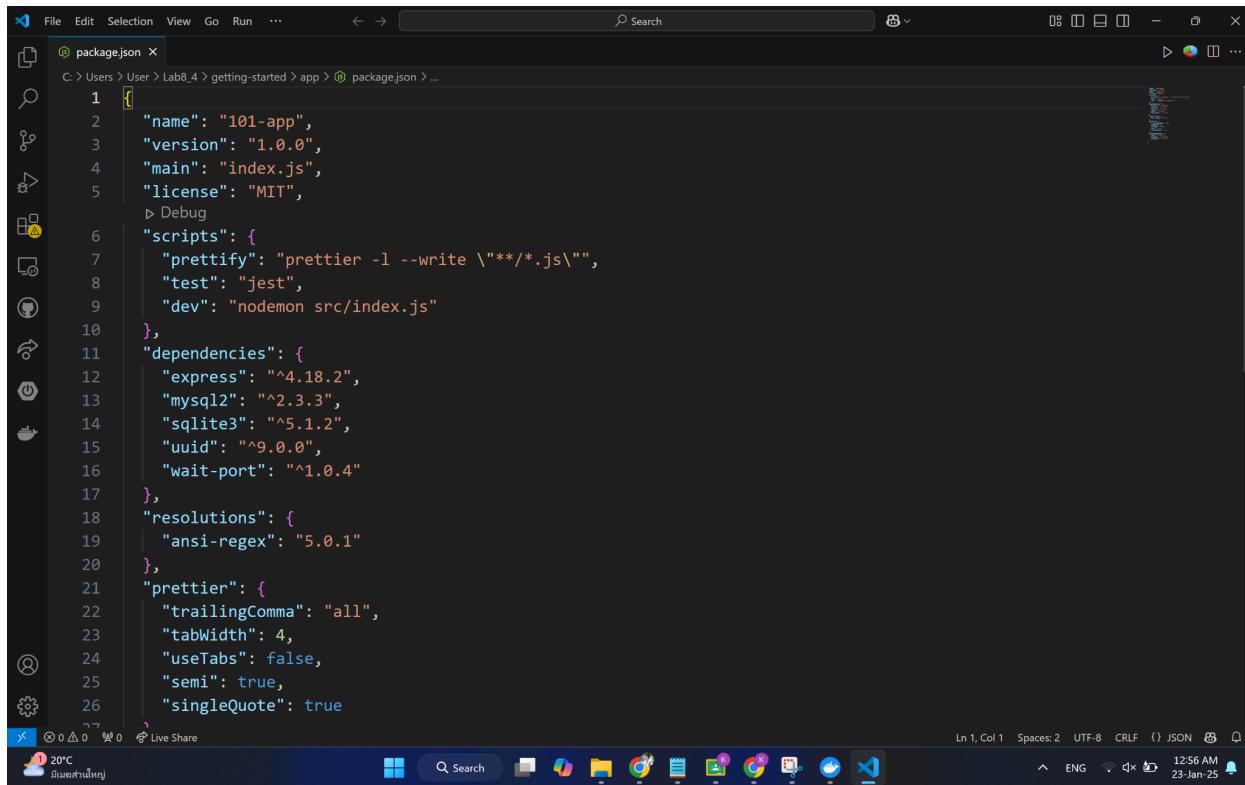
```
$ git clone https://github.com/docker/getting-started.git
```

- 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

```
PS C:\Users\User> cd Lab8_4
PS C:\Users\User\Lab8_4> git clone https://github.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 | 1001.00 KiB/s, done.
Resolving deltas: 100% (523/523), done.
PS C:\Users\User\Lab8_4> cd getting-started
PS C:\Users\User\Lab8_4\getting-started> pwd

Path
-----
C:\Users\User\Lab8_4\getting-started
```



```
1  {
2    "name": "101-app",
3    "version": "1.0.0",
4    "main": "index.js",
5    "license": "MIT",
6    "scripts": {
7      "prettify": "prettier -l --write \"**/*.js\"",
8      "test": "jest",
9      "dev": "nodemon src/index.js"
10    },
11    "dependencies": {
12      "express": "^4.18.2",
13      "mysql2": "^2.3.3",
14      "sqlite3": "^5.1.2",
15      "uuid": "^9.0.0",
16      "wait-port": "^1.0.4"
17    },
18    "resolutions": {
19      "ansi-regex": "5.0.1"
20    },
21    "prettier": {
22      "trailingComma": "all",
23      "tabWidth": 4,
24      "useTabs": false,
25      "semi": true,
26      "singleQuote": true
27    }
28  }
```

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

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

CP353004/SC313 004 Software Engineering (2/2567)

ຜ.ជ.ជ.ສູນ ສົ່ມເລັກ

Lab Worksheet

- ## 6. ทำการ Start ตัว Container ของแอปพลิเคชันที่สร้างขึ้น โดยใช้คำสั่ง

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

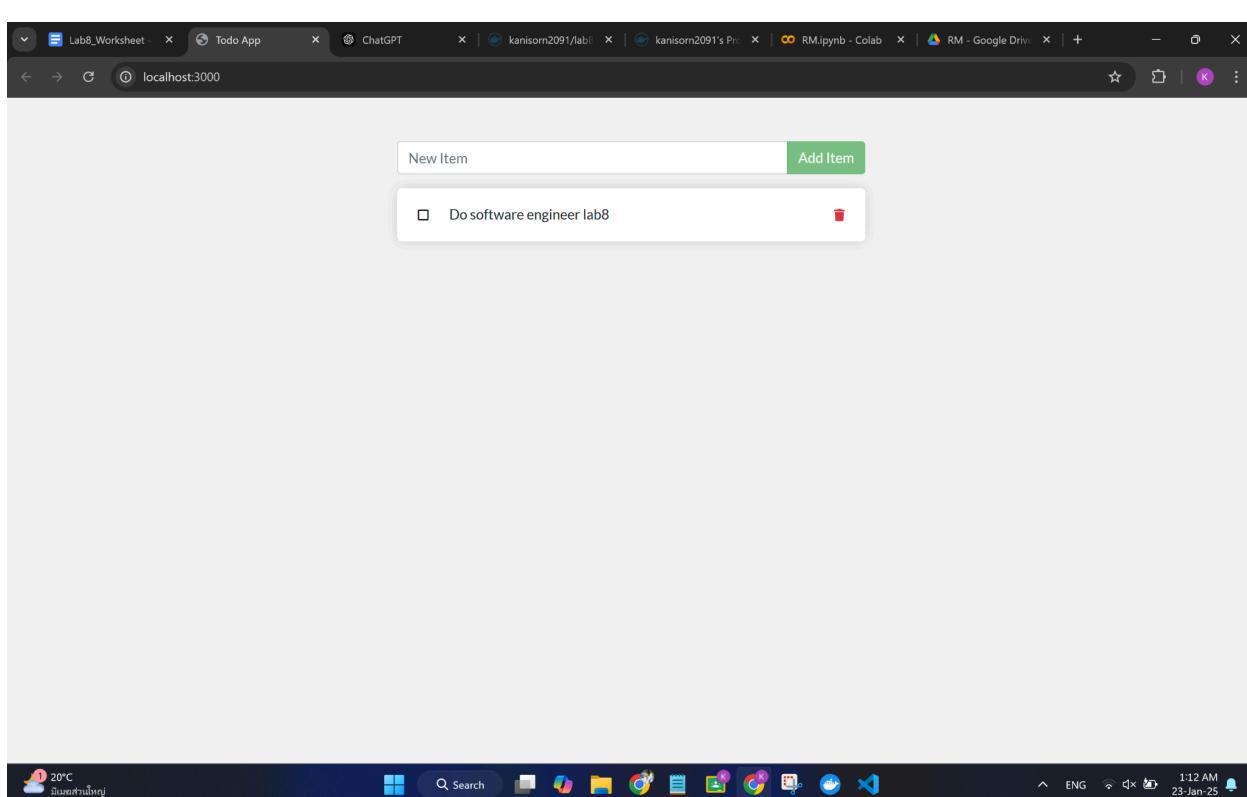
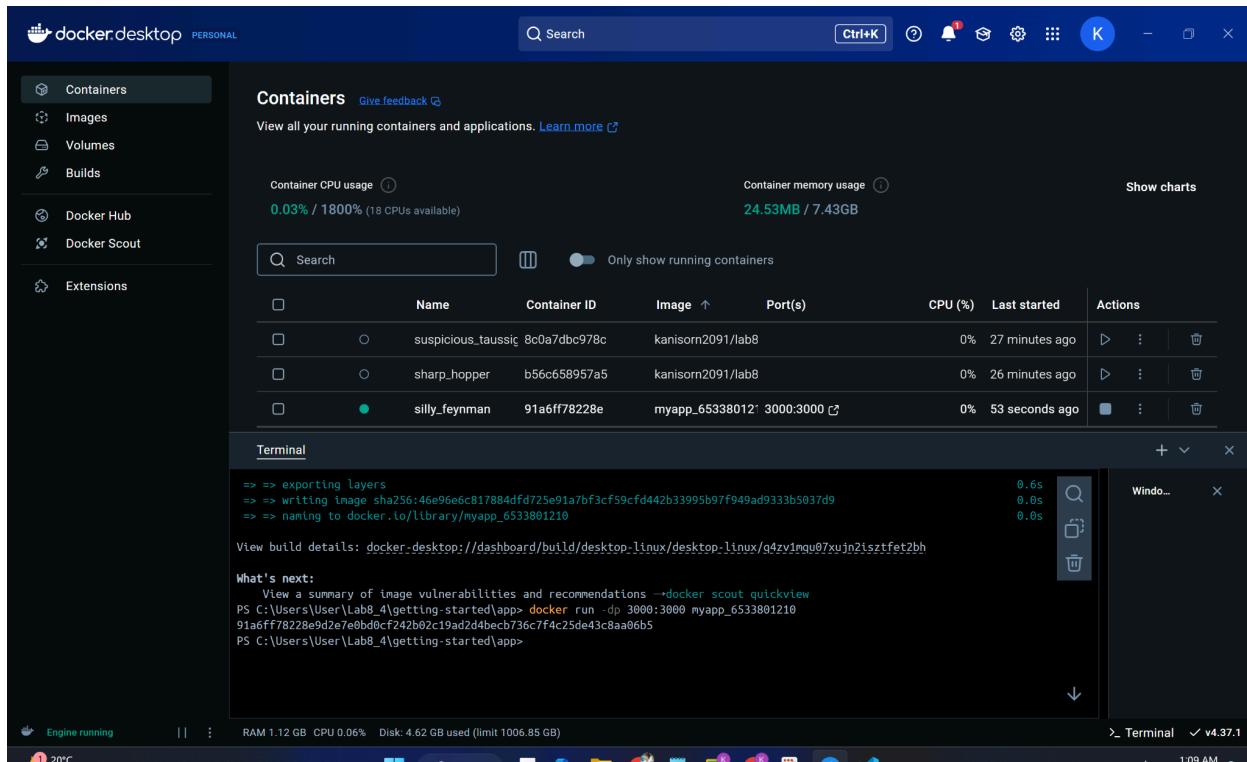
7. เปิด Browser ไปที่ URL = <http://localhost:3000>

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

CP353004/SC313 004 Software Engineering (2/2567)

ผศ. ดร. ชิตสุชา สุ่มเล็ก

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

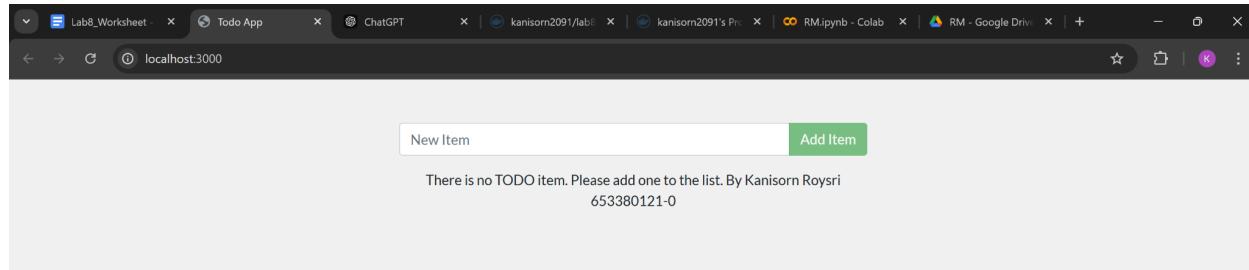
<p className="text-center">**There is no TODO item. Please add one to the list. By ชื่อและนามสกุลของนักศึกษา**</p>

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 ที่เกิดขึ้นหมายความอย่างไร และเกิดขึ้นเพราะอะไร

ไม่มี item ของ To Do list (ยังไม่มีการเพิ่ม item ใหม่ มีจำนวนเป็น 0) สามารถแก้ไขได้โดยเพิ่ม item ของ To Do list เข้าไปอย่างน้อย 1 รายการ

11. ลบ Container ของ Web application เวอร์ชันก่อนแก้ไขออกจากระบบ โดยใช้วิธีใดวิธีหนึ่งดังต่อไปนี้

Lab Worksheet

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. เลือกไอคอนถังขยะในແກ້ວຂອງ Container ที่ต้องการจะลบ
- iii. ยืนยันโดยการกด Delete forever

12. Start และรัน Container ตัวใหม่อีกครั้ง โดยใช้คำสั่งเดียวกันกับข้อ 6

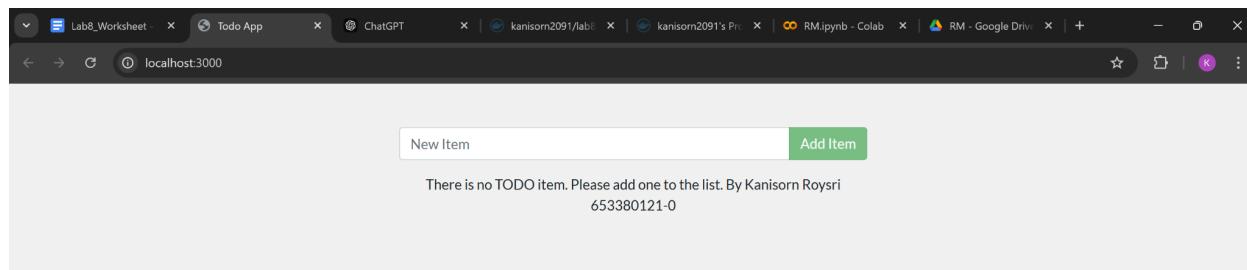
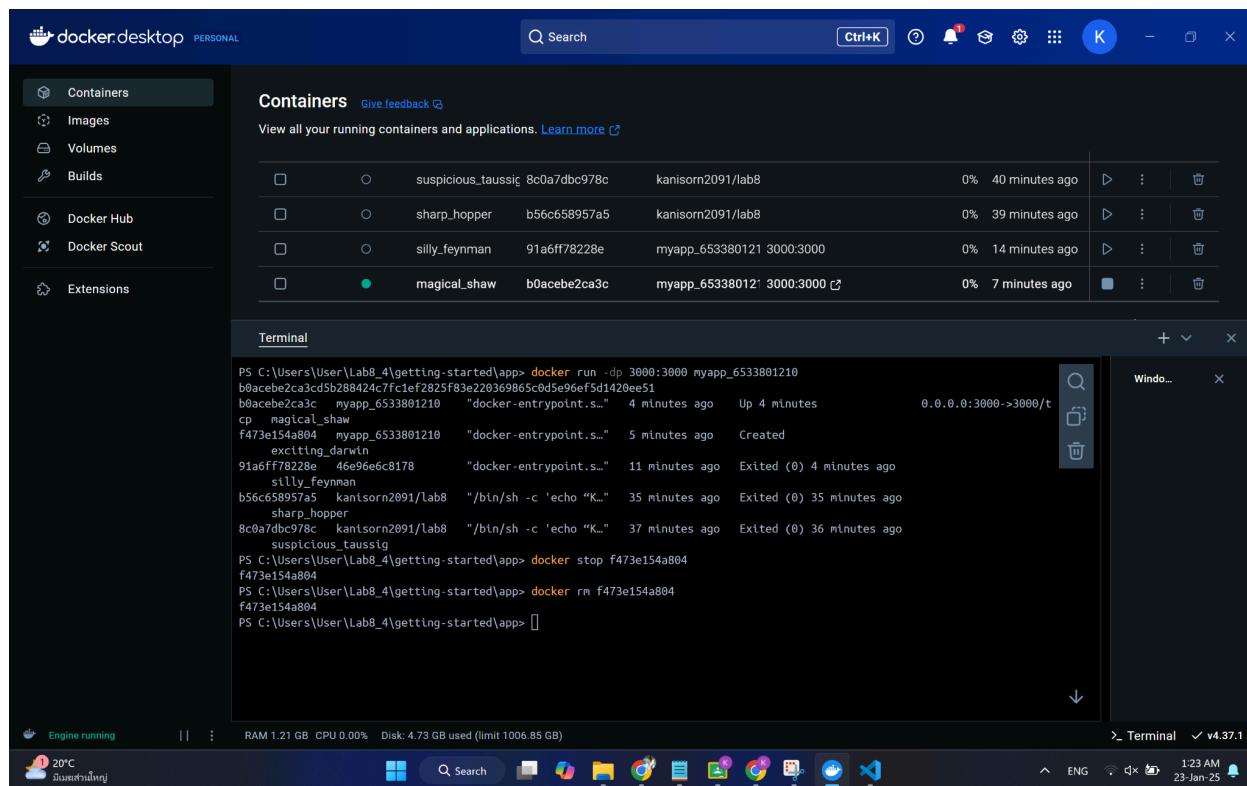
13. เปิด Browser ไปที่ URL = <http://localhost:3000>

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

CP353004/SC313 004 Software Engineering (2/2567)

ผศ. ดร. ชิตสุชา สุ่มเล็ก

Lab Worksheet



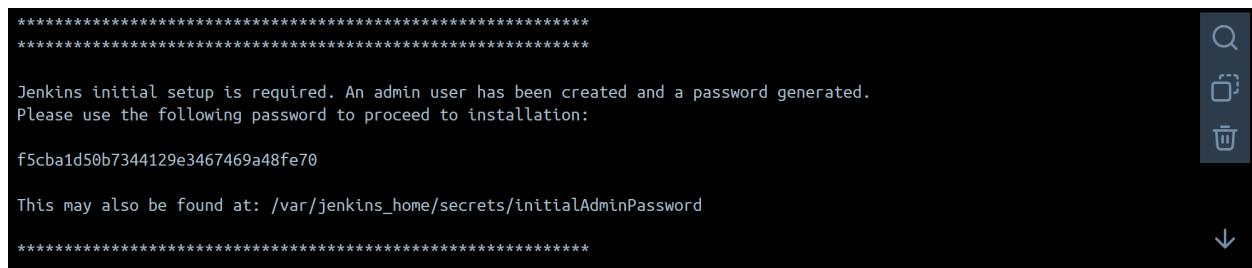
แบบฝึกปฏิบัติที่ 8.5: เริ่มต้นสร้าง Pipeline อย่างง่ายสำหรับการ Deploy ด้วย Jenkins

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:latest

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



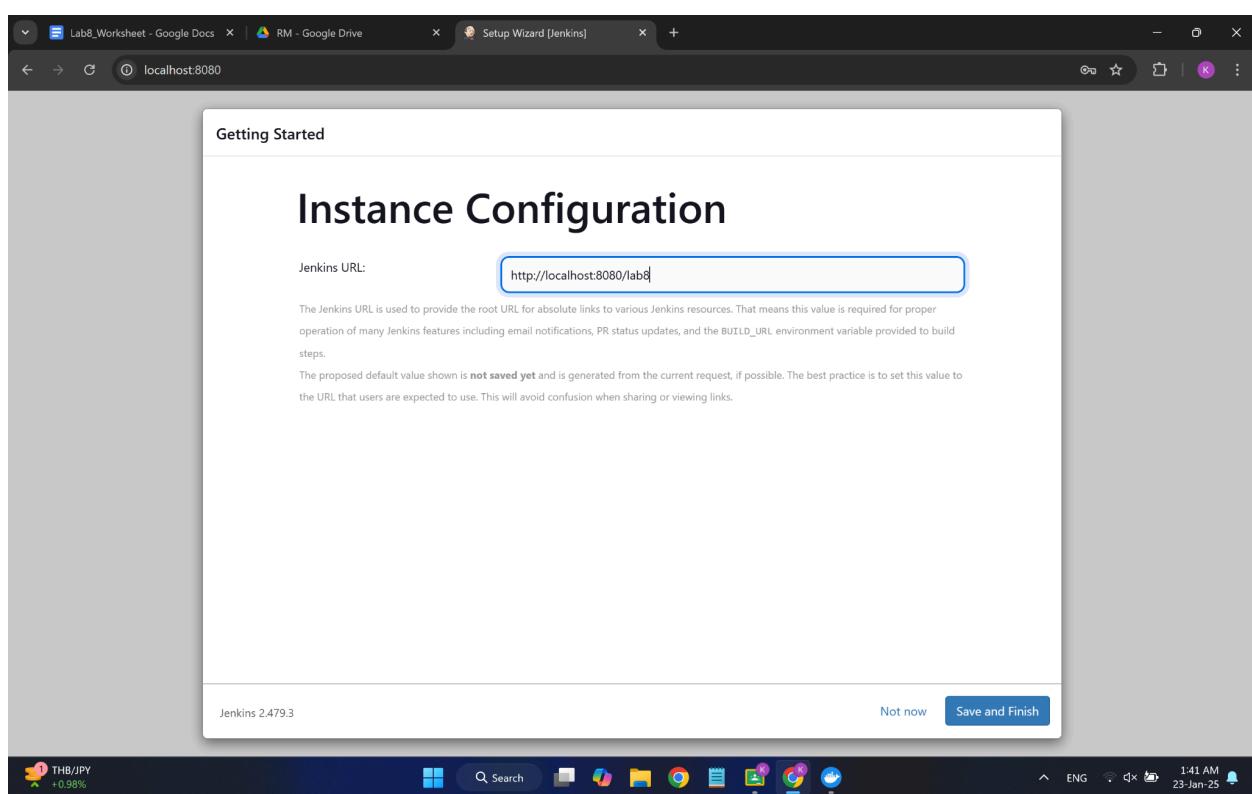
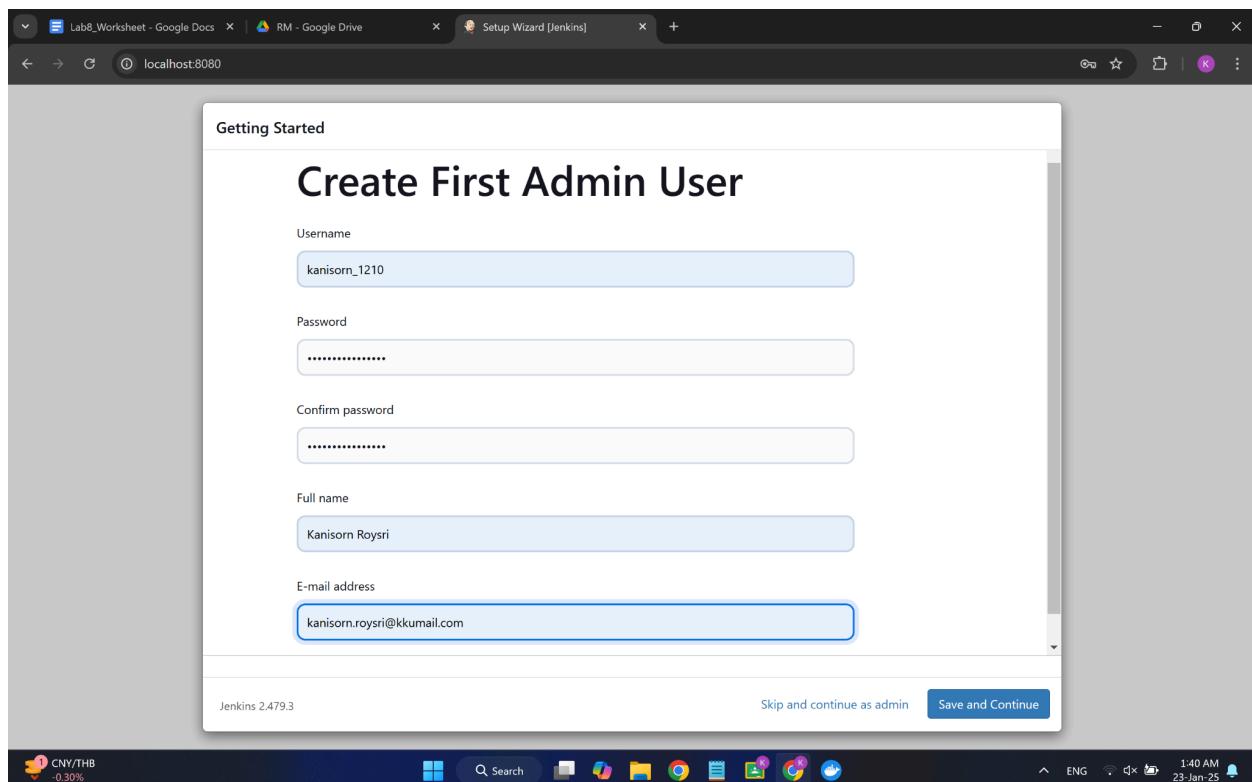
```
*****
Jenkins initial setup is required. An admin user has been created and a password generated.
Please use the following password to proceed to installation:
f5cba1d50b7344129e3467469a48fe70
This may also be found at: /var/jenkins_home/secrets/initialAdminPassword
*****
```

4. เมื่อได้รับการยืนยันว่า Jenkins is fully up and running ให้เปิดบราวเซอร์ และป้อนที่อยู่เป็น localhost:8080
5. ทำการ Unlock Jenkins ด้วยรหัสผ่านที่ได้ในข้อที่ 3
6. สร้าง Admin User โดยใช้ username เป็นชื่อจริงของนักศึกษา
พร้อมรหัสสี่ตัวท้าย เช่น somsri_3062
[Check point#13] Capture หน้าจอที่แสดงผลการตั้งค่า

CP353004/SC313 004 Software Engineering (2/2567)

ผศ. ดร. ชิตสุชา สุ่มเล็ก

Lab Worksheet



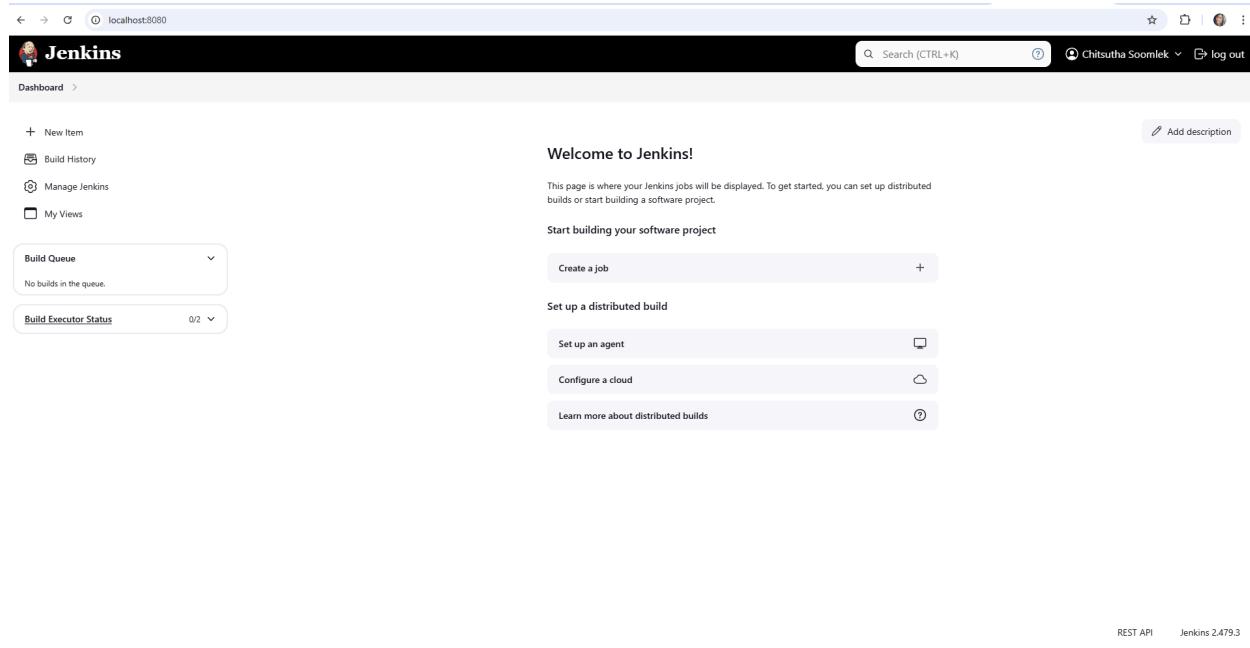
CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตสุชา สุ่มเล็ก

Lab Worksheet

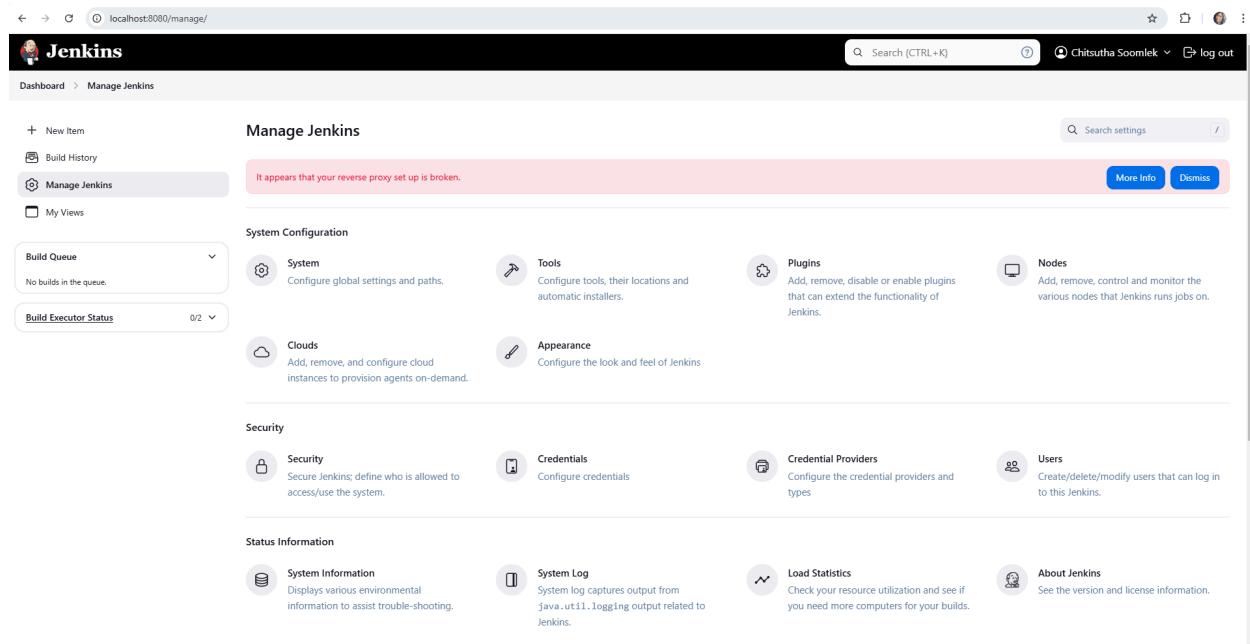
7. กำหนด Jenkins URL เป็น <http://localhost:8080/lab8>

8. เมื่อติดตั้งเรียบร้อยแล้วจะพบกันหน้า Dashboard ดังแสดงในภาพ



The screenshot shows the Jenkins dashboard at <http://localhost:8080>. The top navigation bar includes links for 'Dashboard', 'Search (CTRL+K)', 'Chitsutha Soomlek', and 'log out'. The main content area features a 'Welcome to Jenkins!' message with a sub-instruction: 'This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.' Below this, there are several sections: 'Start building your software project' (with 'Create a job' and a '+' button), 'Set up a distributed build' (with 'Set up an agent', 'Configure a cloud', and 'Learn more about distributed builds'), and 'Build Queue' (showing 'No builds in the queue'). The bottom right corner of the page displays 'REST API' and 'Jenkins 2.479.3'.

9. เลือก Manage Jenkins และไปที่เมนู Plugins



The screenshot shows the 'Manage Jenkins' page at <http://localhost:8080/manage/>. The top navigation bar includes links for 'Dashboard', 'Manage Jenkins', 'Search (CTRL+K)', 'Chitsutha Soomlek', and 'log out'. A prominent red message box states: 'It appears that your reverse proxy set up is broken.' Below this, the 'Manage Jenkins' interface is divided into several sections: 'System Configuration' (with 'System', 'Tools', 'Clouds', 'Appearance', and 'Plugins' sections), 'Security' (with 'Security', 'Credentials', 'Credential Providers', and 'Users' sections), and 'Status Information' (with 'System Information', 'System Log', 'Load Statistics', and 'About Jenkins' sections). The bottom right corner of the page displays 'More Info' and 'Dismiss' buttons.

10. ไปที่เมนู Available plugins แล้วเลือกติดตั้ง Robotframework เพิ่มเติม



localhost:8080/manage/pluginManager/available

Jenkins

Dashboard > Manage Jenkins > Plugins

Search (CTRL+K) Chitsutha Soomlek log out

Plugins

Updates Available plugins Installed plugins Advanced settings Download progress

Robot Framework 5.0.0

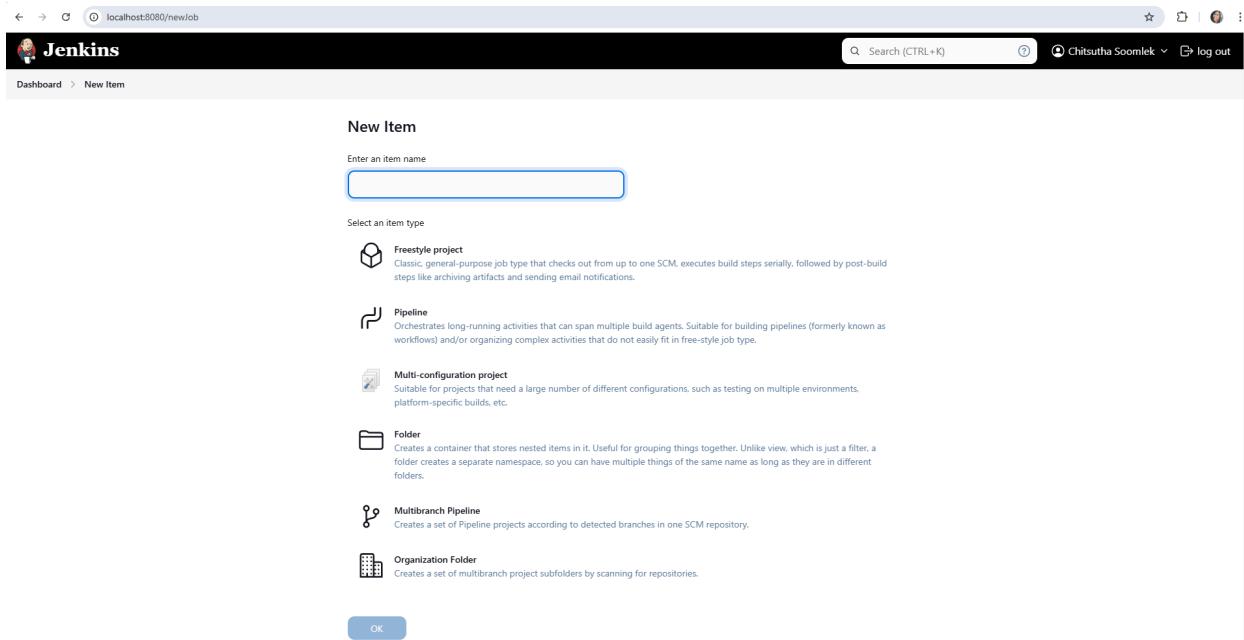
Build Reports

This publisher stores Robot Framework test reports for builds and shows summaries of them in project and build views along with trend graph.

Released 2 mo 7 days ago

Install

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



localhost:8080/newJob

Jenkins

Dashboard > New Item

Search (CTRL+K) Chitsutha Soomlek log out

New Item

Enter an item name

Select an item type

Freestyle project

Pipeline

Multi-configuration project

Folder

Multibranch Pipeline

Organization Folder

OK

Lab Worksheet

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

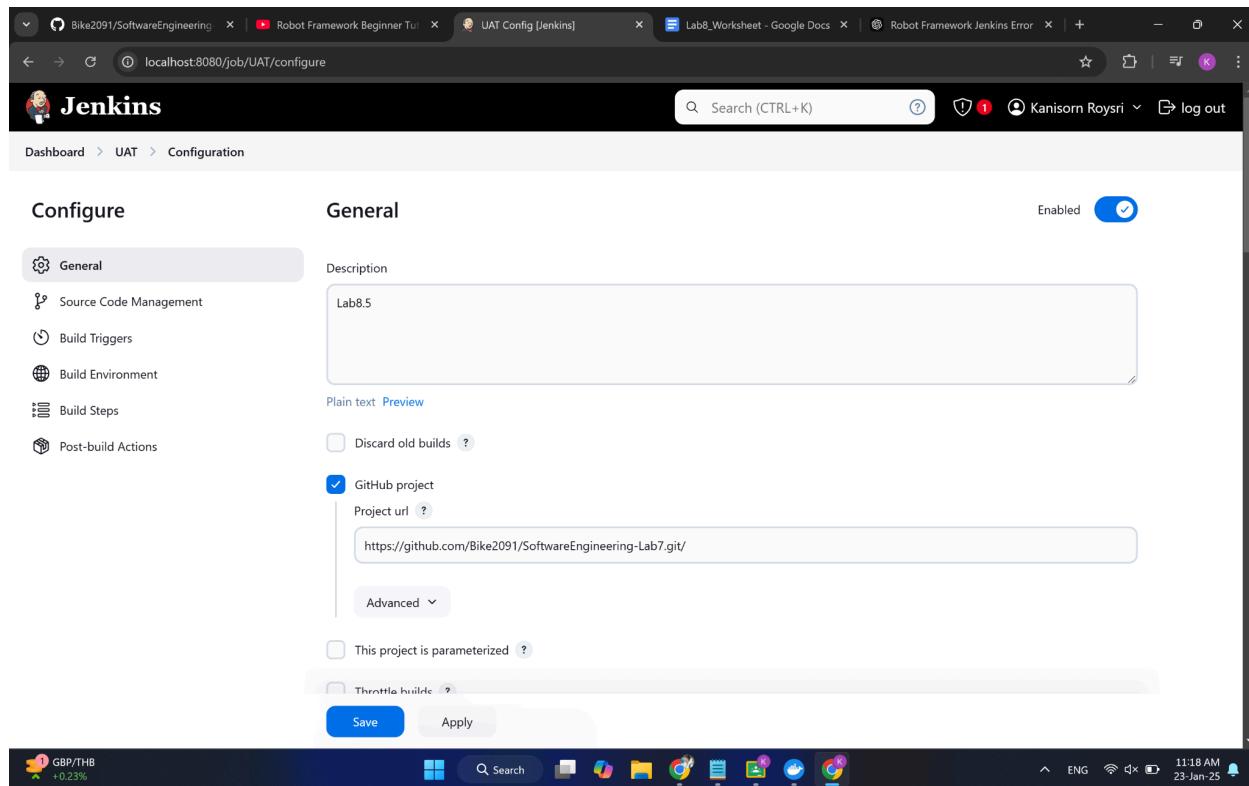
Description: Lab 8.5

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

Build Trigger: เลือกแบบ Build periodically และกำหนดให้ build ทุก 15 นาที

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

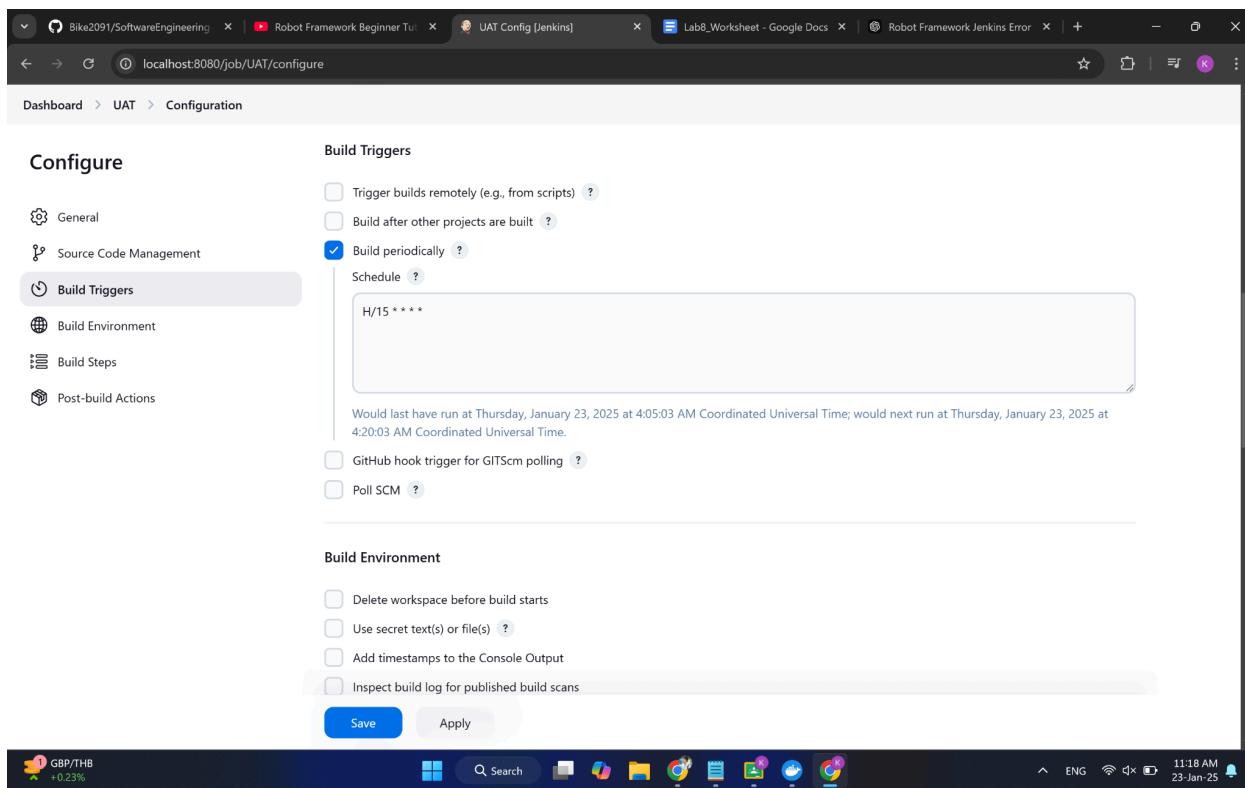
[Check point#14] Capture หน้าจอแสดงการตั้งค่า พร้อมกับต้องคำน้ำมันต่อไปนี้



CP353004/SC313 004 Software Engineering (2/2567)

ผศ. ดร. ชิตสุชา สุ่มเล็ก

Lab Worksheet



The screenshot shows the Jenkins UAT Configuration page. The 'Build Triggers' section is active, showing the 'Build periodically' option selected with a schedule of 'H/15 * * * *'. The 'Build Environment' section is also visible. At the bottom, there are 'Save' and 'Apply' buttons.

Build Triggers

- Trigger builds remotely (e.g., from scripts) ?
- Build after other projects are built ?
- Build periodically ?
Schedule ?
H/15 * * * *
Would last have run at Thursday, January 23, 2025 at 4:05:03 AM Coordinated Universal Time; would next run at Thursday, January 23, 2025 at 4:20:03 AM 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

Build Steps

Execute shell ?

Command

See the list of available environment variables

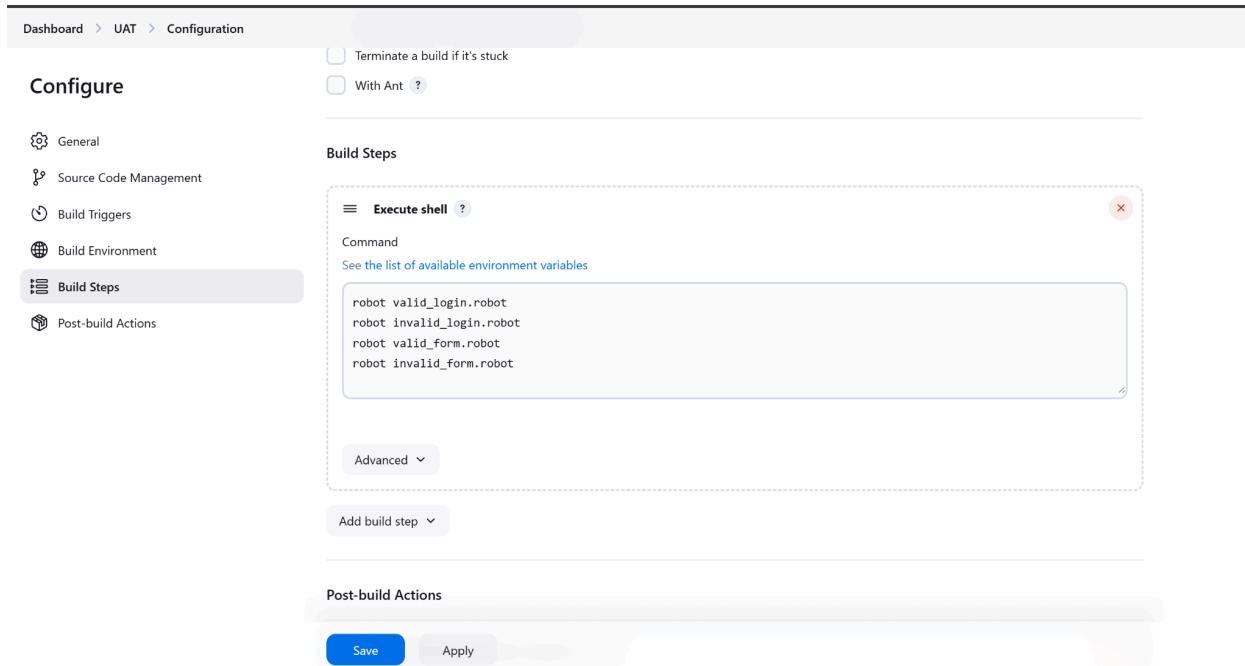
```
robot valid_login.robot
robot invalid_login.robot
robot valid_form.robot
robot invalid_form.robot
```

Advanced ▾

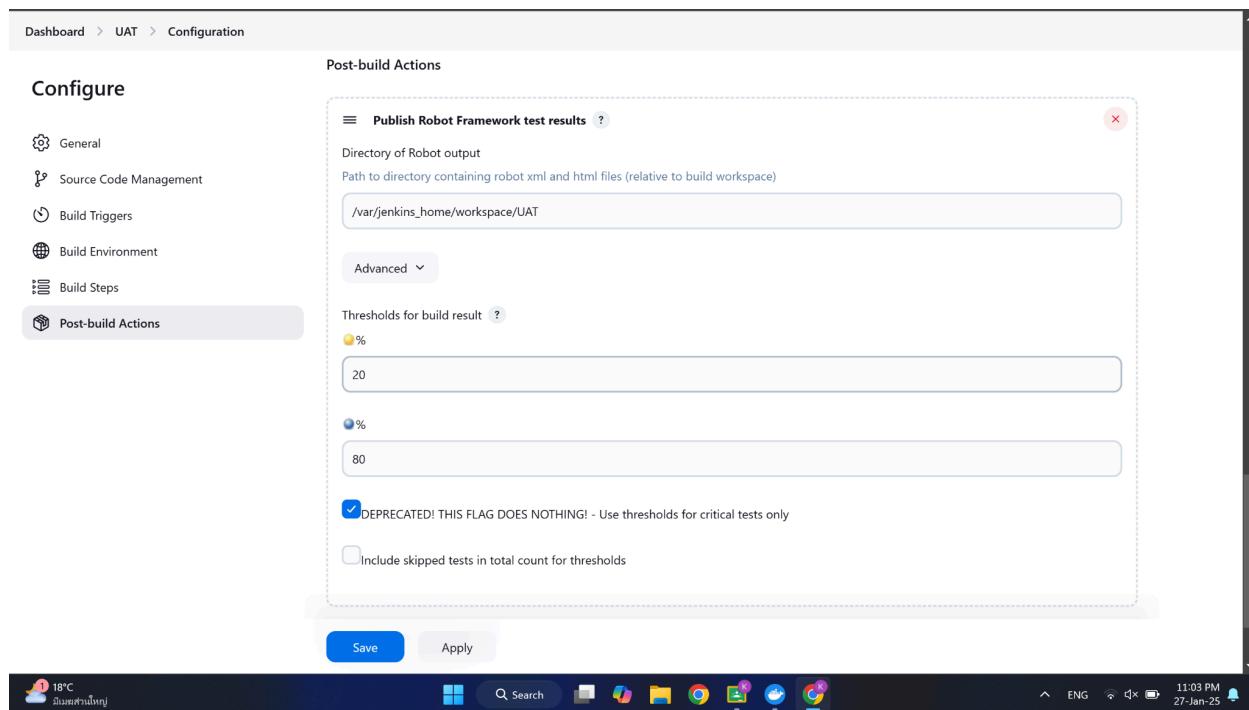
Add build step ▾

Post-build Actions

Save Apply



The screenshot shows the Jenkins UAT Configuration page. The 'Build Steps' section is active, displaying an 'Execute shell' step with the command: 'robot valid_login.robot', 'robot invalid_login.robot', 'robot valid_form.robot', and 'robot invalid_form.robot'. There is an 'Advanced' dropdown and a 'Add build step' button. The 'Post-build Actions' section is below it.



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

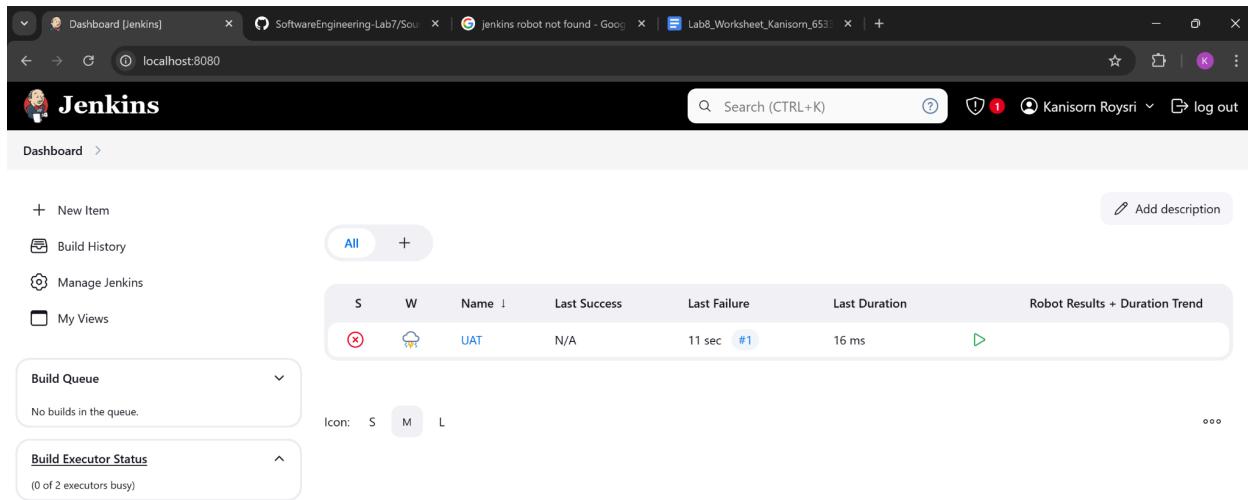
Post-build action: เพิ่ม Publish Robot Framework test results -> ระบุไดเร็คทอรีที่เก็บไฟล์ผลการทดสอบโดย Robot framework ในรูป xml และ html -> ตั้งค่า Threshold เป็น % ของการทดสอบที่ไม่ผ่านแล้ว นับว่าซอฟต์แวร์มีปัญหา -> ตั้งค่า Threshold เป็น % ของการทดสอบที่ผ่านแล้วนับว่าซอฟต์แวร์มีอยู่ในสถานะที่สามารถนำไปใช้งานได้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

13. กด Apply และ Save
14. สั่ง Build Now

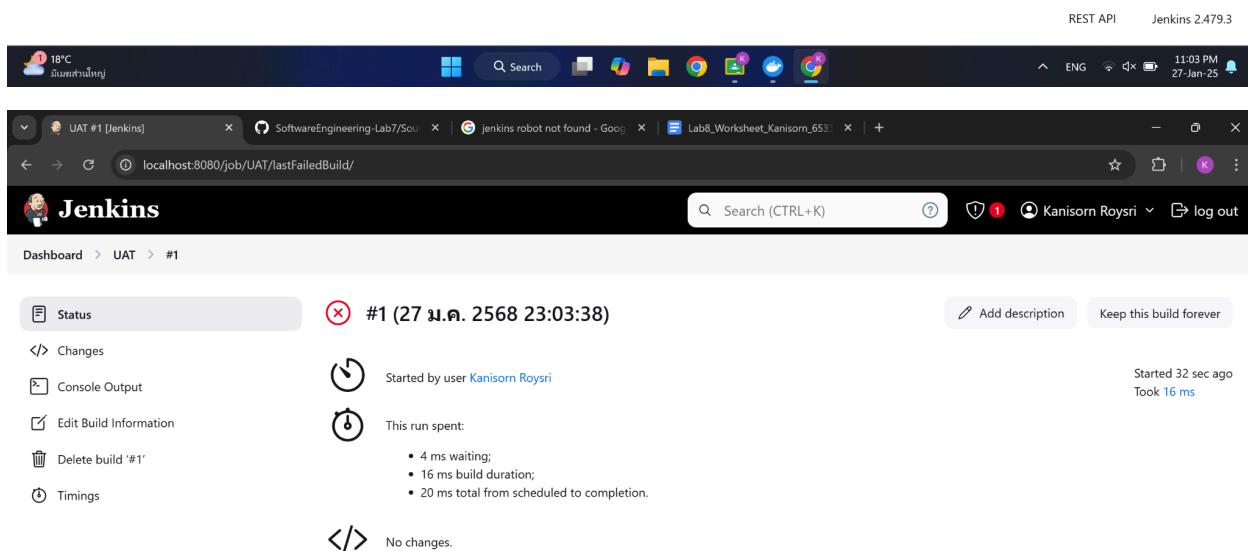
[Check point#15] Capture หน้าจอแสดงหน้าหลักของ Pipeline และ Console Output

CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตสุชา สุ่มเล็ก Lab Worksheet



The screenshot shows the Jenkins dashboard at localhost:8080. The main content area displays a table for the 'UAT' job. The table has columns for Status (S), Work (W), Name, Last Success, Last Failure, Last Duration, and Robot Results + Duration Trend. The 'UAT' job is currently failing (indicated by a red circle with a white 'X'). The last failure occurred 11 seconds ago, and the last duration was 16 ms. The 'Robot Results + Duration Trend' section shows a green arrow pointing right, indicating an upward trend. On the left, there are sections for 'Build Queue' (empty) and 'Build Executor Status' (0 of 2 executors busy). The top navigation bar includes links for 'Dashboard', 'Build History', 'Manage Jenkins', and 'My Views'. A search bar at the top right says 'Search (CTRL+K)'. The bottom right corner shows 'REST API' and 'Jenkins 2.479.3'.



The screenshot shows the Jenkins job details for 'UAT #1' at localhost:8080/job/UAT/lastFailedBuild/. The top navigation bar is identical to the dashboard. The main content area shows the details for build #1, which failed on January 27, 2023, at 23:03:38. The build was started by user 'Kanisorn Roysri'. The 'Status' section shows a red circle with a white 'X' and the text '#1 (27 ม.ค. 2568 23:03:38)'. The 'Changes' section is empty. The 'Console Output' section is collapsed. The 'Edit Build Information' section is collapsed. The 'Delete build #1' section is collapsed. The 'Timings' section is collapsed and shows the following details: Started 32 sec ago, Took 16 ms. The bottom right corner shows 'REST API' and 'Jenkins 2.479.3'.

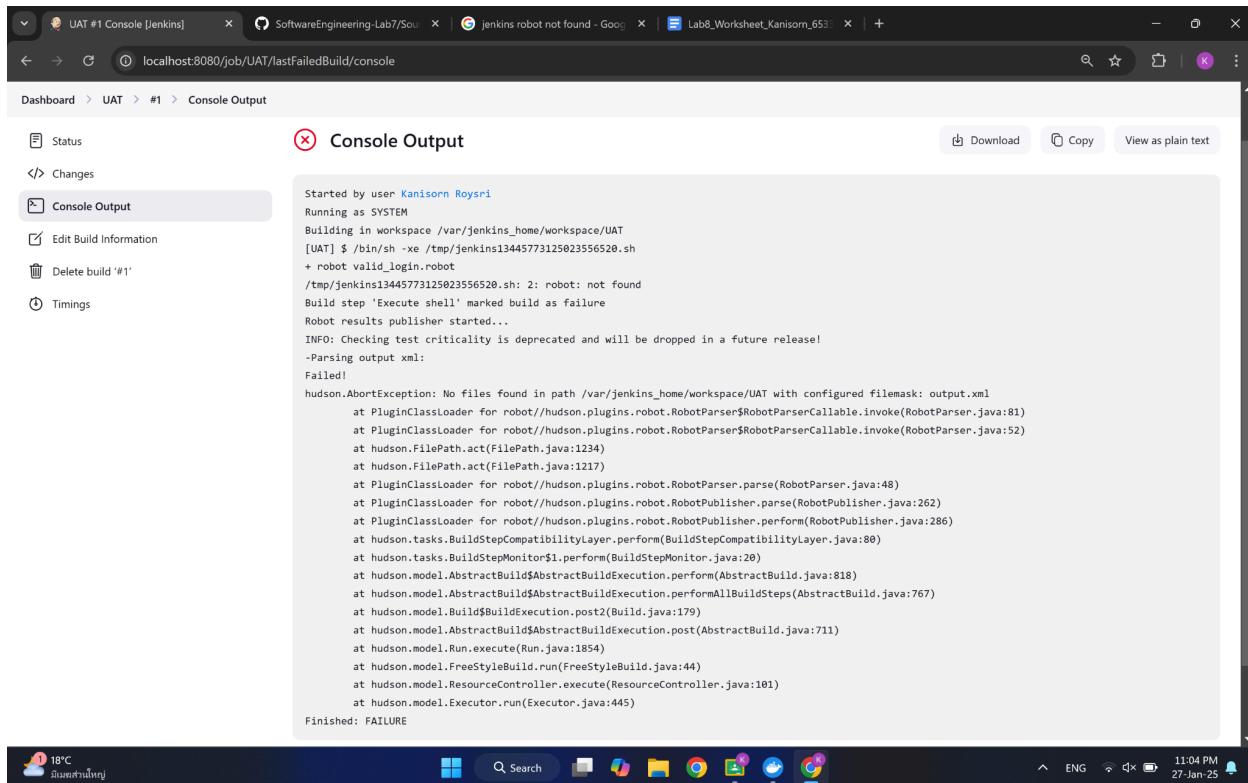


The screenshot shows the Windows taskbar at the bottom of the screen. It includes icons for the Start button, Search, File Explorer, Edge, and other system icons. A Jenkins icon is visible in the taskbar, indicating that the Jenkins application is running in the background.

CP353004/SC313 004 Software Engineering (2/2567)

ผศ. ดร. ชิตสุชา สุ่มเล็ก

Lab Worksheet



The screenshot shows a browser window with the URL localhost:8080/job/UAT/lastFailedBuild/console. The page title is "Console Output". The console output window is titled "Console Output" and contains the following text:

```
Started by user Kanisorn Roysri
Running as SYSTEM
Building in workspace /var/jenkins_home/workspace/UAT
[UAT] $ /bin/sh -xe /tmp/jenkins13445773125023556520.sh
+ robot valid_login.robot
/tmp/jenkins13445773125023556520.sh: 2: robot: not found
Build step 'Execute shell' marked build as failure
Robot results publisher started...
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 filenmask: 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
```

The browser taskbar at the bottom shows the date and time as 27-Jan-25, 11:04 PM.