

## Lab Worksheet

ชื่อ-นามสกุล น.ส. พลอยชมพู วงศ์ปรีดิกุล รหัสนักศึกษา 653380141-4 Section 2

## Lab#8 – Software Deployment Using Docker

## วัตถุประสงค์การเรียนรู้

1. ผู้เรียนสามารถอธิบายเกี่ยวกับ Software deployment ได้
2. ผู้เรียนสามารถสร้างและรัน Container จาก Docker image ได้
3. ผู้เรียนสามารถสร้าง Docker files และ Docker images ได้
4. ผู้เรียนสามารถนำซอฟต์แวร์ที่พัฒนาขึ้นให้สามารถรันบนสภาพแวดล้อมเดียวกันและทำงานร่วมกันกับสมาชิกในทีมพัฒนาซอฟต์แวร์ผ่าน Docker hub ได้
5. ผู้เรียนสามารถเริ่มต้นใช้งาน Jenkins เพื่อสร้าง Pipeline ในการ Deploy งานได้

## Pre-requisite

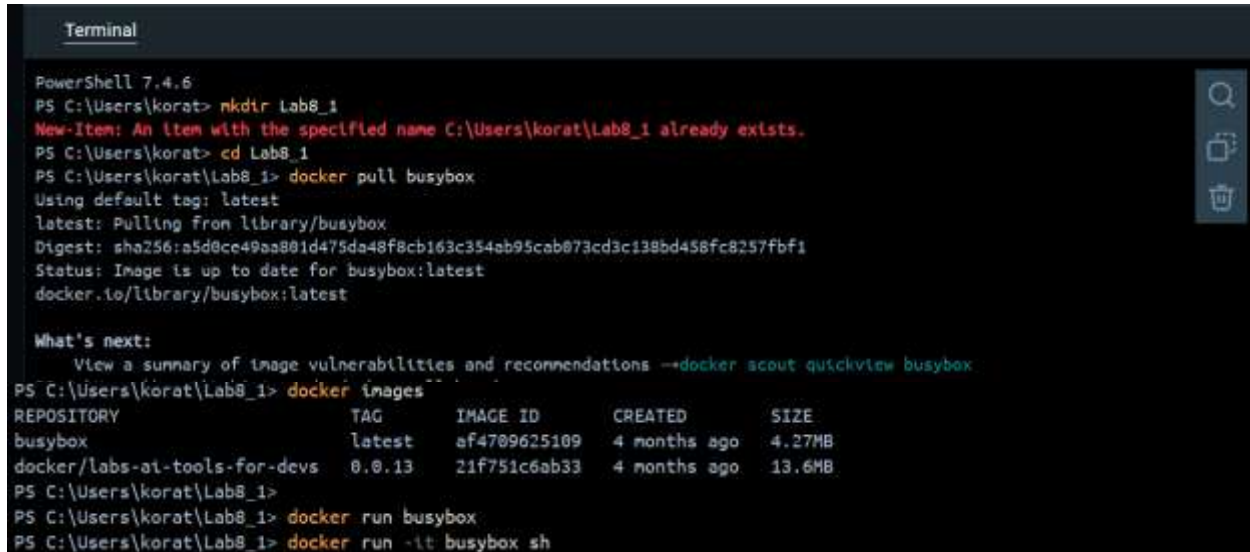
1. ติดตั้ง Docker desktop ลงบนเครื่องคอมพิวเตอร์ โดยดาวน์โหลดจาก <https://www.docker.com/get-started>
2. สร้าง Account บน Docker hub (<https://hub.docker.com/signup>)
3. กำหนดให้ \$ หมายถึง Command prompt และ <> หมายถึง ให้ป้อนค่าของพารามิเตอร์ที่กำหนด

## แบบฝึกปฏิบัติที่ 8.1 Hello world - รัน Container จาก Docker image

1. เปิดใช้งาน Docker desktop และ Login ด้วย Username และ Password ที่ลงทะเบียนกับ Docker Hub เอาไว้
1. เปิด Command line หรือ Terminal บน Docker Desktop จากนั้นสร้าง Directory ชื่อ Lab8\_1
2. ย้ายตำแหน่งปัจจุบันไปที่ Lab8\_1 เพื่อใช้เป็น Working directory
3. ป้อนคำสั่ง \$ docker pull busybox หรือ \$ sudo docker pull busybox สำหรับกรณีที่ติดปัญหา Permission denied  
(หมายเหตุ: BusyBox เป็น software suite ที่รองรับคำสั่งบางอย่างบน Unix - <https://busybox.net>)
4. ป้อนคำสั่ง \$ docker images

## Lab Worksheet

[Check point#1] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงผลลัพธ์ที่ได้พร้อมกับตอบคำถามต่อไปนี้



```

Terminal
PowerShell 7.4.6
PS C:\Users\korat> mkdir Lab8_1
New-Item: An item with the specified name C:\Users\korat\Lab8_1 already exists.
PS C:\Users\korat> cd Lab8_1
PS C:\Users\korat\Lab8_1> docker pull busybox
Using default tag: latest
latest: Pulling from library/busybox
Digest: sha256:a5d0cc49aa801d475da48f8cb163c354ab95cab073cd3c138bd458fc8257fbf1
Status: Image is up to date for busybox:latest
docker.io/library/busybox:latest

What's next:
  View a summary of image vulnerabilities and recommendations -->docker scout quickview busybox
PS C:\Users\korat\Lab8_1> docker images
REPOSITORY          TAG         IMAGE ID      CREATED       SIZE
busybox              latest      af4709625109  4 months ago  4.27MB
docker/labs-ai-tools-for-devs  0.0.13     21f751c6ab33  4 months ago  13.6MB
PS C:\Users\korat\Lab8_1>
PS C:\Users\korat\Lab8_1> docker run busybox
PS C:\Users\korat\Lab8_1> docker run -it busybox sh
  
```

- (1) สิ่งที่อยู่ภายใต้คอลัมน์ Repository คืออะไร  
ชื่อของ Docker image ที่ดึงมาจาก Docker Hub เช่น busybox
- (2) Tag ที่ใช้บ่งบอกถึงอะไร  
ใช้บ่งบอกเวอร์ชันหรือสถานะของ Image เช่น latest หมายถึง Image เวอร์ชันล่าสุด
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

## Lab Worksheet

```

/ # 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 07:09 .
drwxr-xr-x 1 root   root   4096 Jan 28 07:09 ..
-rwxr-xr-x 1 root   root     0 Jan 28 07:09 .dockerenv
drwxr-xr-x 2 root   root  12288 Sep 26 21:31 bin
drwxr-xr-x 5 root   root   3680 Jan 28 07:09 dev
drwxr-xr-x 1 root   root   4096 Jan 28 07:09 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 240 root   root     0 Jan 28 07:09 proc
drwx----- 1 root   root   4096 Jan 28 07:10 root
dr-xr-xr-x 11 root   root     0 Jan 28 07:09 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\korat\Lab8_1> docker run busybox echo "Hello Ploychomp Wongkeeratikul from busybox"
Hello Ploychomp Wongkeeratikul from busybox
PS C:\Users\korat\Lab8_1> docker ps -a
CONTAINER ID   IMAGE      COMMAND                  CREATED         STATUS
PORTS         NAMES
071dbf5029fc   busybox   "echo 'Hello Ploycho..." 12 seconds ago   Exited (0) 11 seco
nds ago       nifty_carson
afbed358ef2f   busybox   "sh"                     2 minutes ago   Exited (0) 2 minut
es ago       pedantic_ronan
1d086fbabebd   busybox   "echo 'Hello Ploycho..." 5 days ago       Exited (0) 5 days
ago         silly_margulits
64223976aa9d   busybox   "sh"                     5 days ago       Exited (0) 5 days
ago         wizardly_euler
0870d1c22c49   busybox   "sh"                     5 days ago       Exited (0) 5 days
ago         sad_wu
3edca70d109b   docker/labs-at-tools-for-devs:0.0.13 "/bin/sh -c '/servic..." 3 months ago     Exited (1) 3 month
s ago       cranky_bardeen

```

(1) เมื่อใช้ option -it ในคำสั่ง run ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป

Option -it เปิดโหมด interactive ซึ่งทำให้สามารถใช้งาน Command Line ภายใน Container ได้ เช่น การพิมพ์คำสั่ง sh

(2) คอลัมน์ STATUS จากการรันคำสั่ง docker ps -a แสดงถึงข้อมูลอะไร

แสดงสถานะของ Container:

Up หมายถึง Container กำลังทำงาน

Exited หมายถึง Container หยุดทำงานแล้ว

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

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

## Lab Worksheet

```

C:\Users\korat\Lab8_1> docker rm id086fbabebd
id086fbabebd
C:\Users\korat\Lab8_1> docker ps -a

```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
71dbf5820fc	busybox	"echo 'Hello Playcha..."	17 minutes ago	Exited (0) 17 minutes ago		mifty_carson
82078771d66	busybox	"sh"	18 minutes ago	Exited (0) 17 minutes ago		awesome_lumiere
fbed358ef2f	busybox	"sh"	19 minutes ago	Exited (0) 19 minutes ago		pedantic_raman
4223976aa9d	busybox	"sh"	6 days ago	Exited (0) 6 days ago		wizardly_euler
070d1c22c49	busybox	"sh"	6 days ago	Exited (0) 6 days ago		sad_wu
edca78d189b	docker/labs-at-tools-for-devs:0.0.13	"/bin/sh -c '/service..."	3 months ago	Exited (1) 3 months ago		cranky_bardeen

## 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."

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

## Lab Worksheet

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

```
PS C:\Users\korat> mkdir Lab8_2

Directory: C:\Users\korat

Mode                LastWriteTime         Length Name
----                -
d-----          28/1/2568         14:33      Lab8_2

C:\Users\korat\Lab8_2>docker build -t myfirstimage .
[+] Building 0.2s (5/5) FINISHED                                docker:desktop-linux
=> [internal] load build definition from Dockerfile              0.0s
=> == transferring dockerfile: 267B                             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           0.0s
=> exporting to image                                           0.0s

=> == exporting layers                                           0.0s
=> == writing image sha256:b81587ba08c7924cd85467fc0b7d70daeed993ebc5907a50765bb88e5c497f29 0.0s
=> == naming to docker.io/library/myfirstimage                  0.0s

New build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/y4ymBq5quyt5Bkmgbfk00jklid

C:\Users\korat\Lab8_2>docker run myfirstimage
พอลอยชมพู วงศ์วิจิตร วรโสนิกศึกษา 653380141-4 ชื่อเล่น พอลอย
```

```
FROM busybox
CMD echo "Hi there. This is my first docker image."
CMD echo "พอลอยชมพู วงศ์วิจิตร วรโสนิกศึกษา 653380141-4 ชื่อเล่น พอลอย"
```

- (1) คำสั่งที่ใช้ในการ run คือ  
`docker run myfirstimage`
- (2) Option -t ในคำสั่ง \$ `docker build` ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป  
 -t ใช้กำหนดชื่อและ Tag ให้กับ Image ที่สร้าง เช่น `myfirstimage:latest` โดยค่าเริ่มต้นคือ `latest`

## 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:\Users\korat> mkdir Lab8_3

Directory: C:\Users\korat

Mode                LastWriteTime         Length Name
----                -
d-----          28/1/2568      15:46             Lab8_3

PS C:\Users\korat\Lab8_3> docker build -t ploychonpu/lab8 .
[+] Building 0.3s (5/5) FINISHED                                docker:desktop-linux
=> [internal] load build definition from dockerfile             0.1s
=> -> transferring dockerfile: 254B                             0.0s
=> [internal] load metadata for docker.io/library/busybox:latest 0.0s
=> [internal] load .dockerignore                               0.1s
=> -> transferring context: 2B                                    0.0s
=> CACHED [1/1] FROM docker.io/library/busybox:latest          0.0s
=> exporting to image                                           0.1s
=> -- exporting layers                                          0.0s
View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/dioz01cm4pge9k84b3m1uy1y

PS C:\Users\korat\Lab8_3> docker run ploychonpu/lab8
พจนนพ วังศิริกุล รวณิกศาน 653380141-4

```

6. ทำการ Push ตัว Docker image ไปไว้บน Docker Hub โดยการใช้นำคำสั่ง

\$ 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

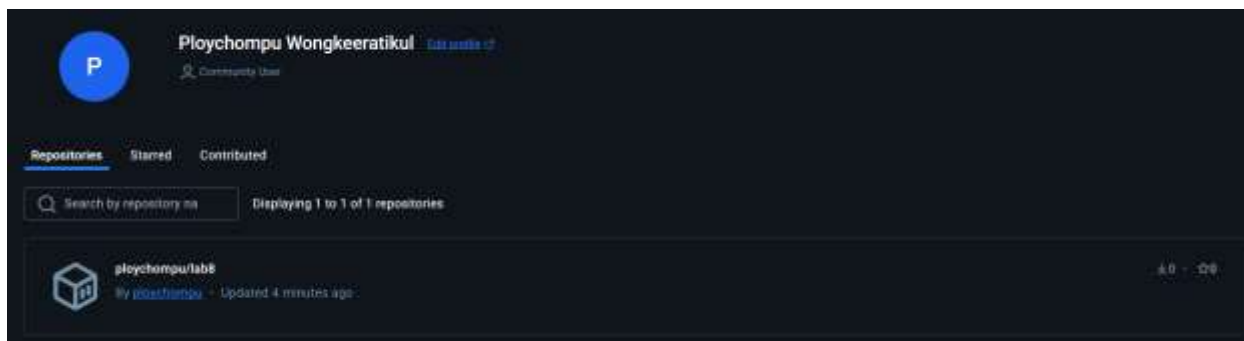
[Check point#6] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดง Repository ที่มี Docker image (<username>/lab8)

```

PS C:\Users\korat\Lab8_3> docker push ploychonpu/lab8
Using default tag: latest
The push refers to repository [docker.io/ploychonpu/lab8]
59654b79daad: Mounted from library/busybox
latest: digest: sha256:352c77ddc3fffb5988d9bd7e9fa5fa87043a8d940fc96cda4cec54124ed2a9ed size: 528

```

## 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 (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงที่อยู่ของ Source code ที่ Clone มาและเนื้อหาของไฟล์ package.json

```

PS C:\Users\korat> cd Lab8_4
PS C:\Users\korat\Lab8_4> git clone https://github.com/docker/getting-started.git
Cloning into 'getting-started'...
remote: Enumerating objects: 988, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 988 (delta 5), reused 1 (delta 1), pack-reused 971 (from 2)
Receiving objects: 100% (988/988), 5.28 MiB | 4.52 MiB/s, done.
Resolving deltas: 100% (523/523), done.
PS C:\Users\korat\Lab8_4> cd getting-started/app
PS C:\Users\korat\Lab8_4\getting-started\app> cat package.json
{
  "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",
  }
}

```



## Lab Worksheet

```

    "uid": "^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"
  }
}

```

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 โดยกำหนดชื่อ image เป็น

myapp\_รหัสนศ. ไม่มีขีด

\$ docker build -t <myapp\_รหัสนศ. ไม่มีขีด> .

[Check point#8] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง)

แสดงคำสั่งและผลลัพธ์ที่ได้ทางหน้าจอ

```

PS C:\Users\korat\Lab8_4\getting-started\app> docker build -t myapp_6533801414 .
[+] Building 24.6s (10/10) FINISHED
=> [internal] load build definition from Dockerfile 0.0s
=> -- transferring dockerfile: 156B 0.0s
=> [internal] load metadata for docker.io/library/node:18-alpine 2.7s
=> [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@sha256:974af6c6bc8314dc6502b14243b8a39fbb2d84d975e9059dd066be3c274fbb23 0.0s
=> [internal] load build context 0.3s
=> -- transferring context: 4.62MB 0.3s
=> CACHED [2/4] WORKDIR /app 0.0s
=> [3/4] COPY . . 0.1s
=> [4/4] RUN yarn install --production 18.9s

```

6. ทำการ Start ตัว Container ของแอปพลิเคชันที่สร้างขึ้น โดยใช้คำสั่ง

## Lab Worksheet

\$ docker run -dp 3000:3000 <myapp\_รหัสสนศ. ไม่มีชี้ด>

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>

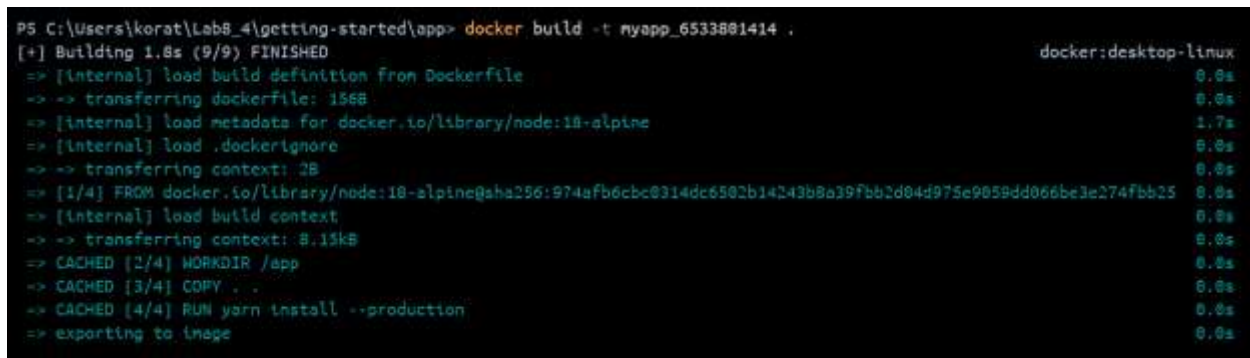
- b. Save ไฟล์ให้เรียบร้อย

9. ทำการ Build Docker image โดยใช้คำสั่งเดียวกันกับข้อ 5

10. Start และรัน Container ตัวใหม่ โดยใช้คำสั่งเดียวกันกับข้อ 6

[Check point#10] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง)

แสดงคำสั่งและผลลัพธ์ที่ได้ทางหน้าจอ พร้อมกับตอบคำถามต่อไปนี้




## Lab Worksheet

```

-> exporting to image
-> => exporting layers
-> => writing image sha256:484cc9186e6ef4e7a5f6fed65d01b0c83d6599acfe46f29f56d96b8e8dd27aee
-> => naming to docker.io/library/myapp_6533881414
View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/06vnjnkxylxul81awvqmh8bfu

What's next:
View a summary of image vulnerabilities and recommendations --> docker scout quickview
PS C:\Users\korat\Lab8_4\getting-started\app> docker run -dp 3000:3000 myapp_6533881414
60611f8ebb61e7888f278f0b32418eb94112c6fd4f041714a151f3932c34f200
docker: Error response from daemon: driver failed programming external connectivity on endpoint bold-beaver (c3ca06803079129e0
cb038c3c3f14860bc5c08d745452afa247aac98de37071): Bind for 0.0.0.0:3000 failed: port is already allocated.

```



(1) Error ที่เกิดขึ้นหมายความว่าอย่างไร และเกิดขึ้นเพราะอะไร

พอร์ต 3000 ที่พยายามใช้สำหรับ Container ถูกใช้งานอยู่แล้วโดยแอปพลิเคชันอื่นหรือ Container อื่นที่กำลังรันอยู่ ทำให้ Docker ไม่สามารถจับคู่ พอร์ต 3000 บนเครื่องกับพอร์ต 3000 บน Container ได้

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

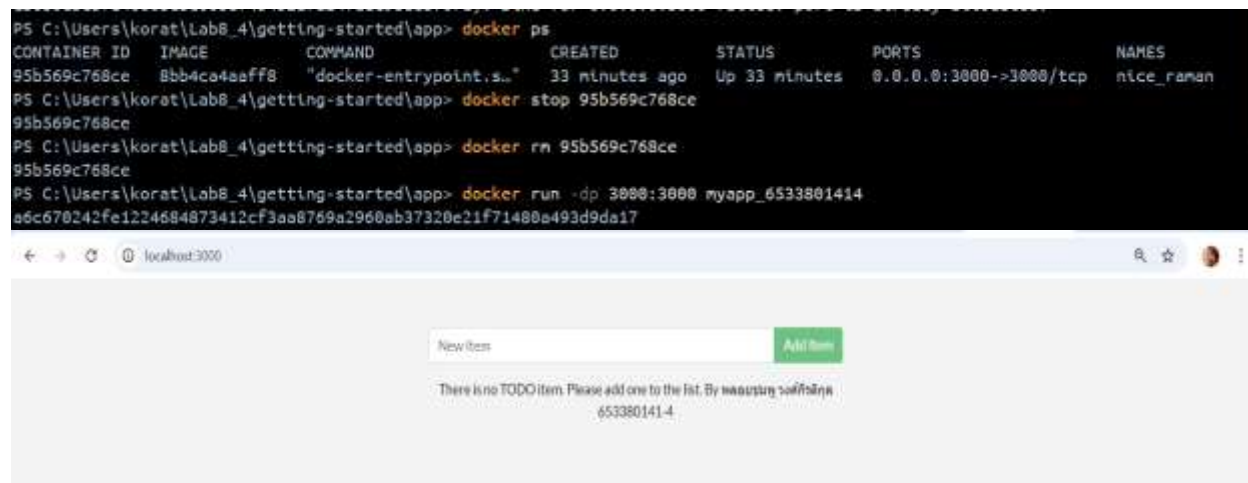
- i. ไปที่หน้าต่าง Containers
- ii. เลือกไอคอนถังขยะในแถวของ Container ที่ต้องการจะลบ
- iii. ยืนยันโดยการกด Delete forever

## Lab Worksheet

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

```
Jenkins initial setup is required. An admin user has been created and a password generated.
Please use the following password to proceed to installation:

b73d54a9eefc40e7bb651991f888083d

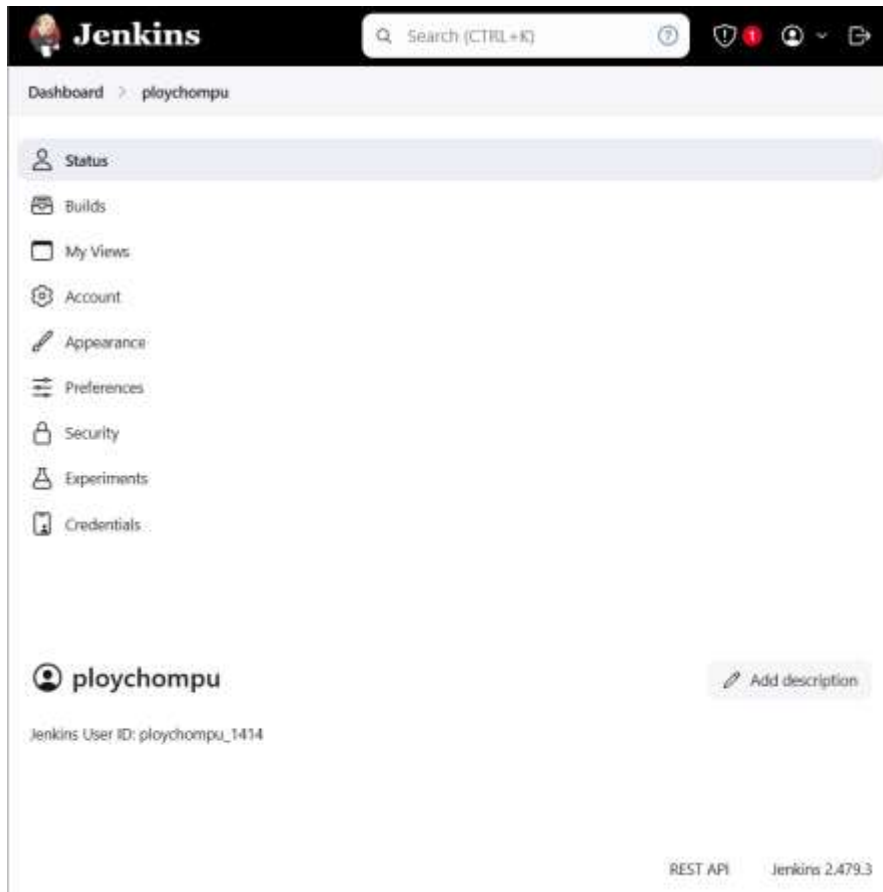
This may also be found at: /var/jenkins_home/secrets/initialAdminPassword
```

4. เมื่อได้รับการยืนยันว่า Jenkins is fully up and running ให้เปิดบราวเซอร์ และป้อนที่อยู่เป็น localhost:8080

## Lab Worksheet

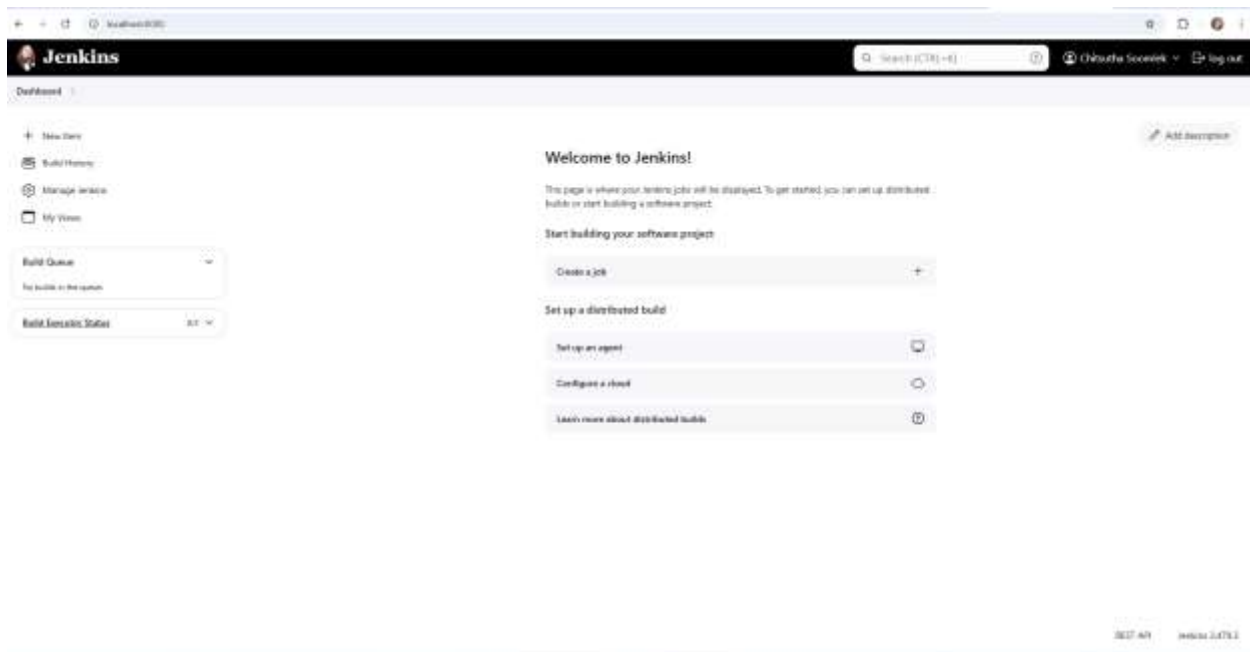
5. ทำการ Unlock Jenkins ด้วยรหัสผ่านที่ได้ในข้อที่ 3
6. สร้าง Admin User โดยใช้ username เป็นชื่อจริงของนักศึกษาพร้อมรหัสสี่ตัวท้าย เช่น somsri\_3062

[Check point#13] Capture หน้าจอที่แสดงผลการตั้งค่า

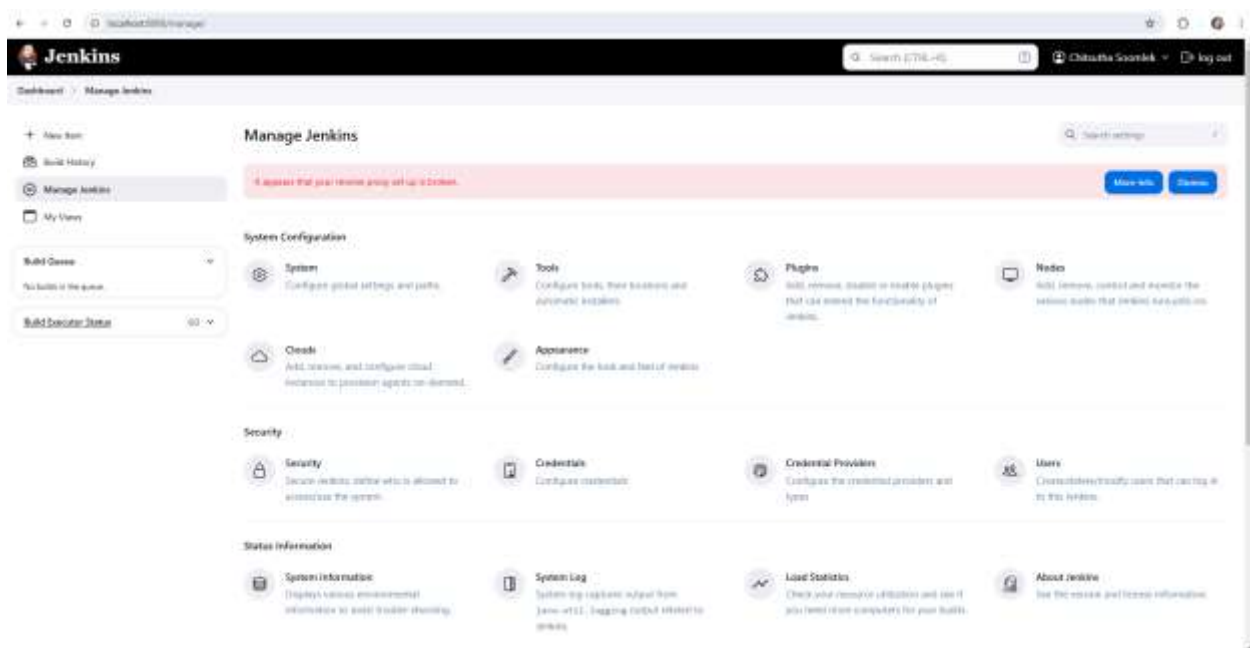


7. กำหนด Jenkins URL เป็น <http://localhost:8080/lab8>
8. เมื่อติดตั้งเรียบร้อยแล้วจะพบหน้าจอ Dashboard ดังแสดงในภาพ

## Lab Worksheet



## 9. เลือก Manage Jenkins แล้วไปที่เมนู Plugins



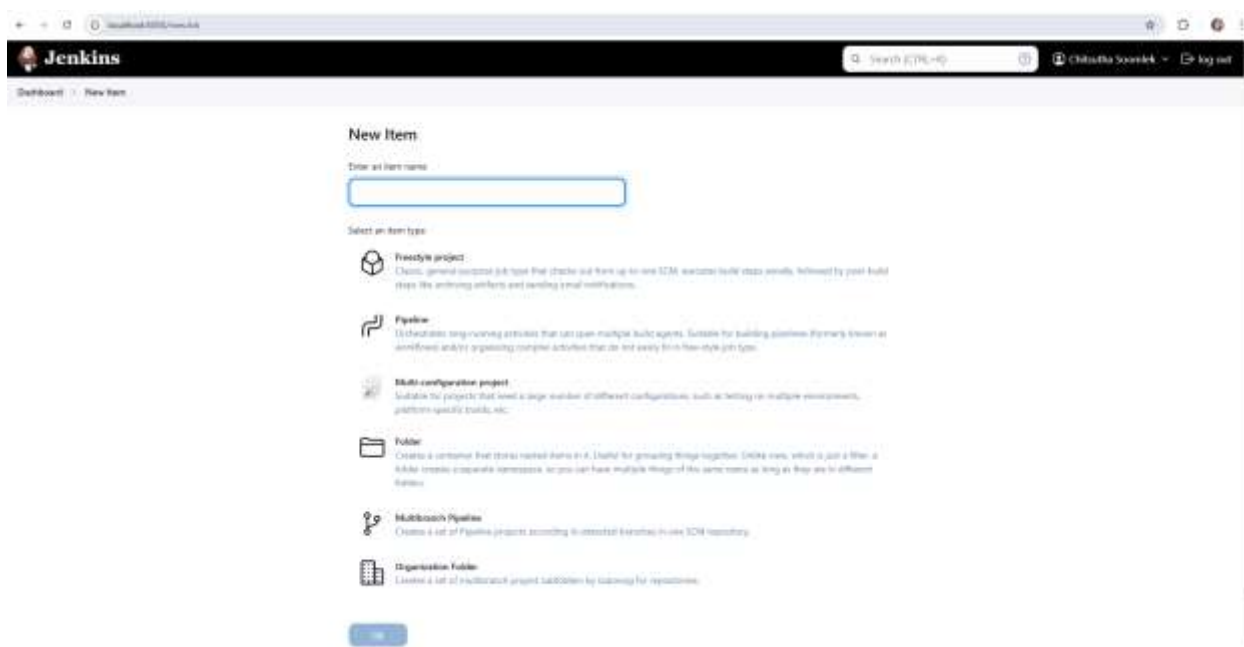


## Lab Worksheet

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

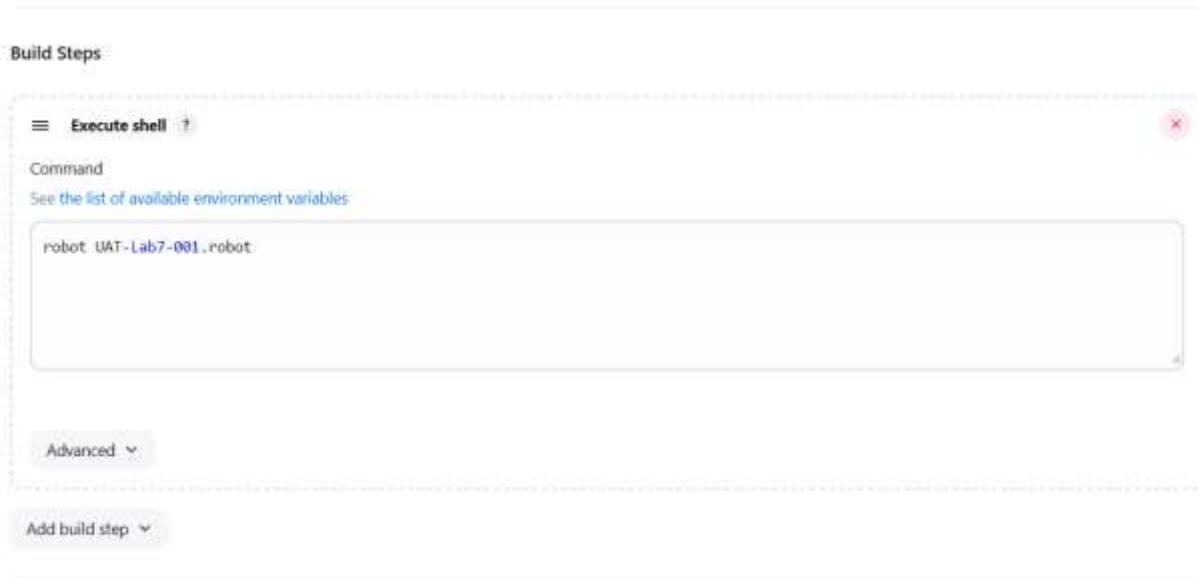
GitHub project: กดเลือก แล้วใส่ Project URL เป็น repository ที่เก็บโค้ด .robot (ดูขั้นตอนที่ 12)

Build Trigger: เลือกแบบ Build periodically แล้วกำหนดให้ build ทุก 15 นาที

## Lab Worksheet

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

[Check point#14] Capture หน้าจอแสดงการตั้งค่า พร้อมกับตอบคำถามต่อไปนี้



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

robot path/to/tests.robot

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

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

% ของการทดสอบที่ไม่ผ่านแล้วนับว่าซอฟต์แวร์มีปัญหา -> ตั้งค่า Threshold เป็น %

ของการทดสอบที่ผ่านแล้วนับว่าซอฟต์แวร์มีอยู่ในสถานะที่สามารถนำไปใช้งานได้ (เช่น 20, 80)

13. กด Apply และ Save

14. สั่ง Build Now

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

## Lab Worksheet

Dashboard > UAT > #1

Status

</> Changes

Console Output

Full Build Information

Delete build #1

Timings

✓ #1 (28 ม.ค. 2568 15:54:43) [Add description](#)

Started by user [phychomp](#)

This run spent:

- 63 ms waiting;
- 54 ms build duration;
- 0.11 sec total from scheduled to completion.

</> No changes.

17