

**Lab Worksheet**

ชื่อ-นามสกุล นายศุภสันต์ มนิสสา รหัสนักศึกษา 653380152-9 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. ผู้เรียนสามารถนำซอฟต์แวร์ที่พัฒนาขึ้นให้สามารถรันบนสภาพแวดล้อมเดียวกันและทำงานร่วมกันกับบุคลากรในทีมพัฒนาซอฟต์แวร์ผ่าน 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 เครื่องเรา
1. เปิด Command line หรือ Terminal บน Docker Desktop จากนั้นสร้าง Directory ชื่อ Lab8\_1
2. ย้ายตำแหน่งปัจจุบันไปที่ Lab8\_1 เพื่อให้เป็น 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) แสดงผลลัพธ์ที่ได้พร้อมกับตอบคำถามต่อไปนี้

## Lab Worksheet

```
[suphasan@mac Lab8_1 % docker pull busybox
Using default tag: latest
latest: Pulling from library/busybox
559c60843878: Already exists
Digest: sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138bd458fc8257fbf1
Status: Downloaded newer image for busybox:latest
docker.io/library/busybox:latest
[suphasan@mac Lab8_1 % docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
jenkins-mo latest 98235bf1f56c 27 minutes ago 894MB
mysql 8.0 f6e7cf1cb86 3 months ago 608MB
busybox latest fc0179a204e2 3 months ago 4.04MB
[suphasan@mac Lab8_1 % docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
jenkins-mo latest 98235bf1f56c 28 minutes ago 894MB
mysql 8.0 f6e7cf1cb86 3 months ago 608MB
busybox latest fc0179a204e2 3 months ago 4.04MB
suphasan@mac Lab8_1 %
]
```

(1) ลิงก์ที่อยู่ภายในให้คอลัมน์ Repository คืออะไร

**ตอบ** Image name

(2) Tag ที่ใช้บ่งบอกถึงอะไร

**ตอบ** version

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

```

suphasan@mac Lab8_1 % docker pull busybox
Using default tag: latest
latest: Pulling from library/busybox
559c66843878: Already exists
Digest: sha256:a5d0ce49aa801d475da48f8cb163c954ab95cab073cd3c138bd458fc8257fbf1
Status: Downloaded newer image for busybox:latest
docker.io/library/busybox:latest
suphasan@mac Lab8_1 % docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
jenkins-mo latest 98235bf1f56c 27 minutes ago 894MB
mysql 8.0 f6e7cfdf1cb86 3 months ago 688MB
busybox latest fc0179a2b4ae2 3 months ago 4.04MB
suphasan@mac Lab8_1 % docker run busybox sh
suphasan@mac 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 08:11 .
drwxr-xr-x 1 root root 4096 Jan 22 08:11 ..
-rw-r--r-- 1 root root 0 Jan 22 08:11 .dockercfg
drwxr-xr-x 2 root root 12288 Sep 26 21:31 bin
drwxr-xr-x 5 root root 360 Jan 22 08:11 dev
drwxr-xr-x 1 root root 4096 Jan 22 08:11 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 222 root root 0 Jan 22 08:11 proc
drwx----- 1 root root 4096 Jan 22 08:11 root
dr-xr-xr-x 11 root root 0 Jan 22 08:11 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
suphasan@mac Lab8_1 % docker run busybox echo "Hello ภาษาไทย นะครับ นะครับ from busybox"
"Hello ภาษาไทย นะครับ นะครับ from busybox"
suphasan@mac Lab8_1 % docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
843e2a04ea8 busybox "echo \"Hello ภาษาไทย\""
6 seconds ago Exited (0) 5 seconds ago amazing_engelbart
bc27d277b15a busybox "sh"
49 seconds ago Exited (0) 35 seconds ago frosty_chaplygin
c915726ed7f0 busybox "sh"
55 seconds ago Exited (0) 53 seconds ago goofy_spence
3c93759c9265 jenkins-mo "/usr/bin/tini -- /u..."
30 minutes ago Up 30 minutes 0.0.0.0:8080->8080/tcp, 0.0.0.0:50000->50000/tcp jovial_haslett
suphasan@mac Lab8_1 %

```

- (1) เมื่อใช้ option -it ในคำสั่ง run แสดงผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสั้นๆ เช่น  
ตอบ จะสามารถใช้ command ใน container ได้
- (2) คอลัมน์ STATUS จากการรันคำสั่ง docker ps -a และแสดงถึงข้อมูลอะไร  
ตอบ แสดงว่า container นั้นกำลังทำงานอยู่หรือปั่นๆ

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

## Lab Worksheet

[Check point#3] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงผลลัพธ์ที่ได้ในขั้นตอนที่ 1

```

latest: Pulling from library/busybox
559c60843678: Already exists
Digest: sha256:a5d5cc49aa801d475da48f8cb163c354ab95cab073cd3c138bd458fc8257fbf1
Status: Downloaded newer image for busybox:latest
docker: /library/busybox:latest
suphasan@mac: Lab8_1 % docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
jenkins-mo latest 98235bf1f56c 27 minutes ago 894MB
mysql 8.0 f6e7cf1dcbb6 3 months ago 608MB
busybox latest fc0179a2b4e2 3 months ago 4.04MB
suphasan@mac: Lab8_1 % docker run busybox
suphasan@mac: Lab8_1 % docker run -it busybox sh
/ # ls -la
/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 08:11 .
drwxr-xr-x  1 root  root      4096 Jan 22 08:11 ..
-rwxr-xr-x  1 root  root          0 Jan 22 08:11 .dockerenv
drwxr-xr-x  2 root  root    12288 Sep 26 21:31 bin
drwxr-xr-x  5 root  root      368 Jan 22 08:11 dev
drwxr-xr-x  1 root  root      4096 Jan 22 08:11 etc
drwxr-xr-x  2 root  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  222 root  root      0 Jan 22 08:11 proc
drwxr-xr-x  1 root  root      4096 Jan 22 08:11 root
dr-xr-xr-x  11 root  root      0 Jan 22 08:11 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
suphasan@mac: Lab8_1 % docker run busybox echo "Hello ภาษาไทย นะนิส่า from busybox"
Hello ภาษาไทย นะนิส่า from busybox
suphasan@mac: Lab8_1 % docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
843e2a84ea8 busybox "echo \"Hello ภาษาไทย\""
49 seconds ago Exited (0) 5 seconds ago amazing_engelbert
bc27d277b15a busybox "sh"
49 seconds ago Exited (0) 35 seconds ago frosty_chaplygin
c915726ed7f0 busybox "sh"
55 seconds ago Exited (0) 53 seconds ago goofy_spence
3c93759c9265 jenkins-mo "/usr/bin/tini -- /u"
30 minutes ago Up 30 minutes 0.0.0.0:8080->8080/tcp, 0.0.0.0:50000->50000/tcp jovial_haslett
843e2a84ea8 suphasan@mac: Lab8_1 % docker rm 843e2a84ea8
843e2a84ea8 suphasan@mac: Lab8_1 %

```

### แบบฝึกปฏิบัติที่ 8.2: สร้าง Docker file และ Docker image

- เปิดใช้งาน Docker desktop และ Login ด้วย Username และ Password ที่ลงทะเบียนกับ Docker Hub เครื่อง
- เปิด Command line หรือ Terminal จากนั้นสร้าง Directory ชื่อ Lab8\_2
- ย้ายตำแหน่งปัจจุบันไปที่ Lab8\_2 เพื่อใช้เป็น Working directory
- สร้าง Dockerfile.swp ไว้ใน Working directory

สำหรับเครื่องที่ใช้ระบบปฏิบัติการวินโดว์ (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"

## Lab Worksheet

EOF

หรือใช้คำสั่ง

\$ touch Dockerfile

แล้วใช้ Text Editor ในการใส่เนื้อหาแทน

5. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้

\$ docker build -t &lt;ชื่อ Image&gt; .

6. เมื่อ Build สำเร็จแล้ว ให้ทำการรัน Docker image ที่สร้างขึ้นในขั้นตอนที่ 5

[Check point#4] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) และแสดงผลลัพธ์ที่ได้ในขั้นตอนที่ 5 พิจารณาและประเมินผล

```

suphasan@mac Lab8_2 % touch Dockerfile
suphasan@mac Lab8_2 % nvim Dockerfile
suphasan@mac Lab8_2 % docker build -t lab8_2 .
[+] Building 0.0s (5/5) FINISHED
=> [internal] load build definition from Dockerfile
=> [internal] load .dockerignore
=> [internal] transfer Dockerfile: 227B
=> [internal] transfer context: 2B
=> [1/1] FROM docker.io/library/busybox:latest
=> exporting to image
=> exporting layers
=> writing image sha256:3d43805d692e3195a52226a7effe065a6ea2391a063eea862b2cf69dd9b8ee
=> naming to docker.io/lab8_2

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/8f6uznpwbqrzb774bk023hega

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)

suphasan@mac Lab8_2 % docker run lab8_2
忙しく作業中 653380152-9 Dream
suphasan@mac Lab8_2 %

```

- (1) คำสั่งที่ใช้ในการ run คือ

**ตอบ** docker run lab8\_2

- (2) Option -t ในคำสั่ง \$ docker build ส่งผลต่อการทำ้งานของคำสั่งอย่างไรบ้าง อธิบายมาพอสั้นๆ เช่น  
**ตอบ** ทำให้สามารถตั้งชื่อและ tag ของ image ได้

### แบบฝึกปฏิบัติที่ 8.3: การแชร์ Docker image ผ่าน Docker Hub

- เปิดเว็บไซต์ Docker desktop และ Login ด้วย Username และ Password ที่ลงทะเบียนกับ Docker Hub เครื่อง

## Lab Worksheet

2. เปิด Command line หรือ Terminal จากนั้นสร้าง Directory ชื่อ Lab8\_3
3. ย้ายตำแหน่งปัจจุบันไปที่ Lab8\_3 เพื่อให้เป็น Working directory
4. สร้าง Dockerfile.swp ไว้ใน Working directory

สำหรับเครื่องที่ใช้ระบบปฏิบัติการwinโดว์ 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

```

suphasan@mac Lab8_3 % touch Dockerfile
suphasan@mac Lab8_3 % nvim Dockerfile
suphasan@mac Lab8_3 % docker build -t suphasan40/lab8 .
[+] Building 0.0s (5/5) FINISHED
   => [internal] load build definition from Dockerfile
   => transferring dockerfile: 246B
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 2)
=> WARN: MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the same stage because only the last one will be used (line 2)
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 3)
=> [internal] load metadata for docker.io/library/busybox:latest
=> [internal] load .dockerignore
=> => transferring context: 2B
=> CACHED [1/1] FROM docker.io/library/busybox:latest
=> exporting to image
=> => exporting layers
=> => writing image sha256:9a23fa984429b4ee24a8b58d0df59044b26c75f6410f5867fd555789758b4970
=> => naming to docker.io/suphasan40/lab8
suphasan@mac Lab8_3 % docker run suphasan40/lab8
婀ນດີເລືດ໌ ນະຊຸມ 653380152-9
suphasan@mac 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

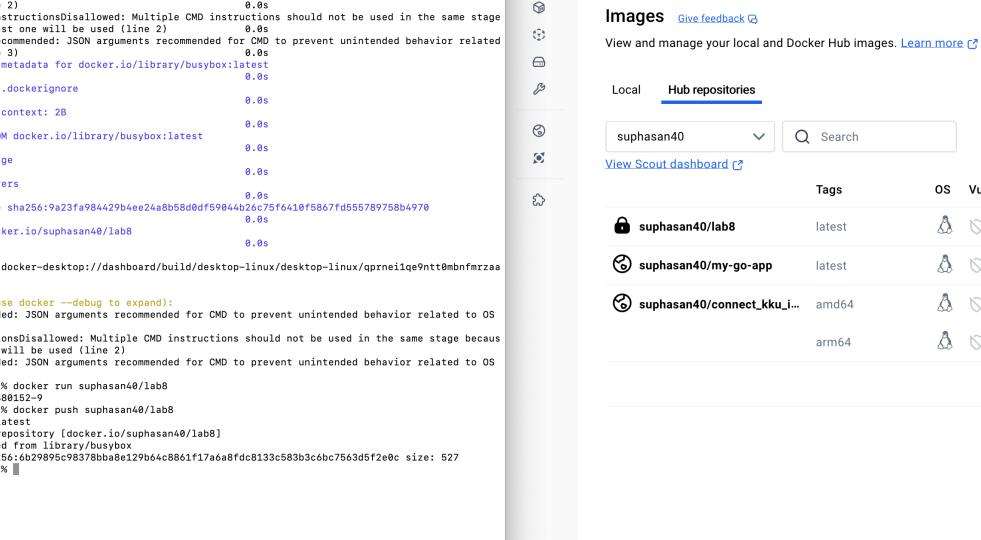
\$ docker login และป้อน Username และ Password ตามที่ระบุใน Command prompt หรือใช้คำสั่ง

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

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

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

## Lab Worksheet



The screenshot shows the Docker Desktop interface with the 'Images' tab selected. The left panel displays a terminal session with build logs for a Dockerfile named 'Lab8\_3'. The logs show the creation of a container image from 'busybox:latest', followed by exporting layers and writing the image to disk. The right panel lists three local repositories under 'Hub repositories': 'suphasan40/lab8' (latest tag), 'suphasan40/my-go-app' (latest tag), and 'suphasan40/connect\_kku\_i...' (amd64 tag). Each entry includes a 'Tags' column, an 'OS' column (showing 'linux' for the first two and 'arm64' for the third), and a 'Vulnerabilities' column with an 'Inactive' status.

```
Lab8_3 - zsh - 104x57
transferring dockerfile: 246B
transferring context: 2B
CACHED [1/1] FROM docker.io/library/busybox:latest
exporting to image
exporting layers
writing image sha256:9a23fa984429b4ee24a8b58d0df59044b26c75f6410f5867fd555789758b497e
naming to docker.io/suphasan40/lab8

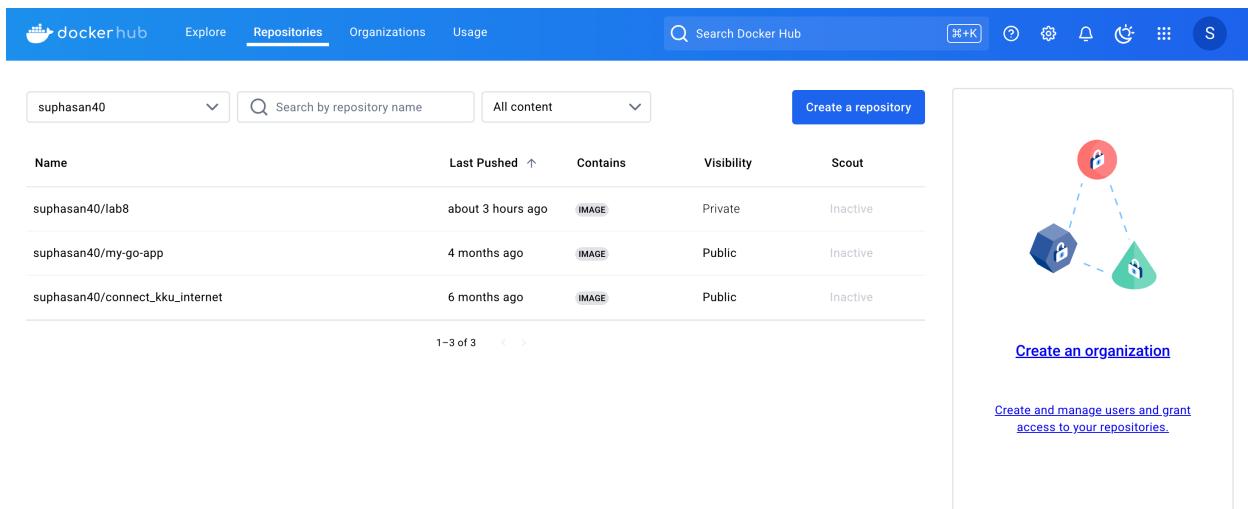
View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/qprne1qe9ntt@mbnfmrzas
s

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)

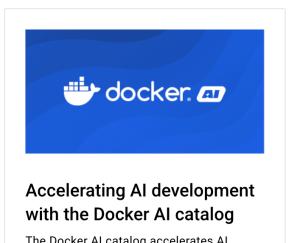
suphasan@mac Lab8_3 % docker run suphasan40/lab8
suphasan@mac Lab8_3 % docker push suphasan40/lab8
Using default tag: latest
The push refers to repository [docker.io/suphasan40/lab8]
c13ebfc506b9: Mounted from library/busybox
latest: digest: sha256:5d9895c98378ba8e129b64c8861f17a6a8fd8133c583b3c6bc7563d5f2e0c size: 527
suphasan@mac Lab8_3 %
```

Tags	OS	Vulnerabilities
suphasan40/lab8	latest	Inactive
suphasan40/my-go-app	latest	Inactive
suphasan40/connect_kku_i...	amd64	Inactive
	arm64	Inactive

## Lab Worksheet



The screenshot shows the Docker Hub interface. At the top, there are tabs for Explore, Repositories (which is selected), Organizations, and Usage. A search bar says "Search Docker Hub". Below the search bar, a dropdown shows "suphasan40". There is a "Create a repository" button. The main area displays three repositories owned by "suphasan40": "lab8" (pushed about 3 hours ago, private, inactive), "my-go-app" (pushed 4 months ago, public, inactive), and "connect\_kku\_internet" (pushed 6 months ago, public, inactive). To the right, there is a section titled "Create an organization" with icons for a lock, a key, and a padlock, and a link to "Create and manage users and grant access to your repositories".

The screenshot shows the Docker AI catalog page. It features the Docker logo and the text "Accelerating AI development with the Docker AI catalog". Below it, it says "The Docker AI catalog accelerates AI".

**แบบฝึกปฏิบัติที่ 8.4: การ Build และ Update แอปพลิเคชันจาก Container image**

1. เปิด Command line หรือ Terminal จากนั้นสร้าง Directory ชื่อ Lab8\_4
2. ทำการ Clone ซอฟต์แวร์ Docker มาจาก 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

```

{
  "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": "^2.0.28",
    "prettier": "^2.7.1"
  }
}

```

4. ภายใต้ getting-started/app ให้สร้าง Dockerfile พ้อมกับใส่เนื้อหาดังต่อไปนี้ลงไปในไฟล์

```

FROM node:18-alpine
WORKDIR /app
COPY . .
RUN yarn install --production
CMD ["node", "src/index.js"]
EXPOSE 3000

```

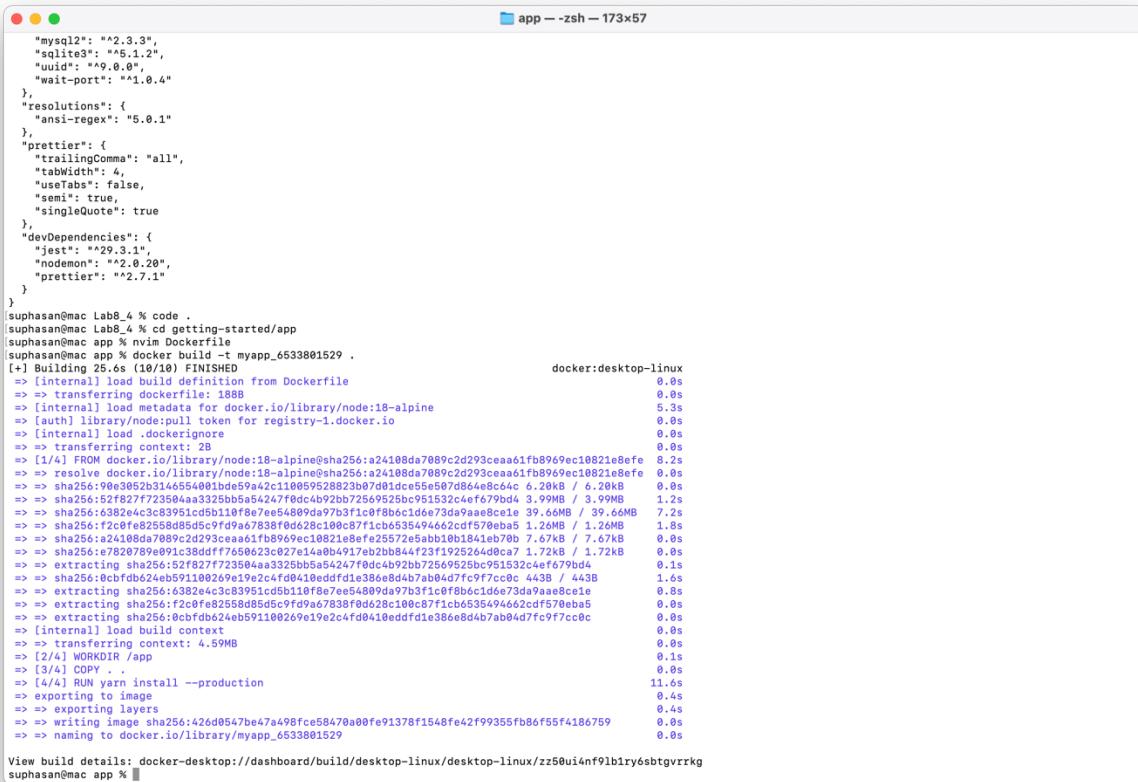
5. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้ โดยกำหนดให้ชื่อ image เป็น

myapp\_รหัสสนศ. ไม่มีขีด

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

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

## Lab Worksheet



```

app -- -zsh - 173x57

"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"
}
}
suphasan@mac Lab8_4 % code .
suphasan@mac Lab8_4 % cd getting-started/app
suphasan@mac app % nvim Dockerfile
suphasan@mac app % docker build -t myapp_6533801529 .
[+] Building 25.6s (10/10) FINISHED
=> [internal] load build definition from Dockerfile          docker:desktop-linux
=> => transferring dockerfile: 188B                         0.0s
=> [internal] load metadata for docker.io/library/node:18-alpine      0.0s
=> [internal] pull https://index.docker.io/v1/repo/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:a24108da7089c2d293ceaa61fb0969ec10821e8efe 8.2s
=> => resolve docker.io/library/node:18-alpine@sha256:a24108da7089c2d293ceaa61fb0969ec10821e8efe 0.0s
=> => sha256:9be3052b314655a0@01de5942c1108059528823b07d01dc5e55e507d864e8c64c 6.20KB / 6.20KB 0.0s
=> => sha256:52f827f723504a3325bb5a54247f0dc4b92bb72569525bc951532c4e679bd4 3.99MB / 3.99MB 1.2s
=> => sha256:16382e4c3c83951cd5b110f8e7ee54899da97b3f1c0f8b6c1d6673da9aae8c1e 39.66MB / 39.66MB 7.2s
=> => sha256:f2cfe82558d85d5c9fd9a67838f0d628c10c0e87f1cb6535494662cd576beba5 1.26MB / 1.26MB 1.8s
=> => sha256:a24108da7089c2d293ceaa61fb0969ec10821e8efe25572e5ab10b1841eb70b 7.67KB / 7.67KB 0.0s
=> => sha256:e7820789e091c38ddff7650623c027e14a0b917eb2bb844f231f925264d0ca7 1.72KB / 1.72KB 0.0s
=> => extracting sha256:52f827f723504a3325bb5a54247f0dc4b92bb72569525bc951532c4e679bd4 0.1s
=> => sha256:0cbfdb24e059110b269e19e2c4fd0410eddf1e38e8d4b7ab04d7fc9f7cc0c 443B / 443B 1.6s
=> => extracting sha256:16382e4c3c83951cd5b110f8e7ee54899da97b3f1c0f8b6c1d6673da9aae8c1e 0.0s
=> => sha256:52f827f723504a3325bb5a54247f0dc4b92bb72569525bc951532c4e679bd4 0.0s
=> => extracting sha256:0cbfdb24e059110b269e19e2c4fd0410eddf1e38e8d4b7ab04d7fc9f7cc0c 0.0s
=> [internal] load build context      0.0s
=> => transferring context: 4.59MB   0.0s
=> [2/4] WORKDIR /app               0.1s
=> [3/4] COPY .                   0.0s
=> [4/4] RUN yarn install --production    11.6s
=> => exporting to image           0.4s
=> => exporting layers            0.4s
=> => writing image sha256:426d0547be47a498fce58470a00fe91378f1548fe42f99355fb86f55f4186759 0.0s
=> => naming to docker.io/library/myapp_6533801529 0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/zz50ui4nf9lb1ry6sbtgvrrkg
suphasan@mac app %

```

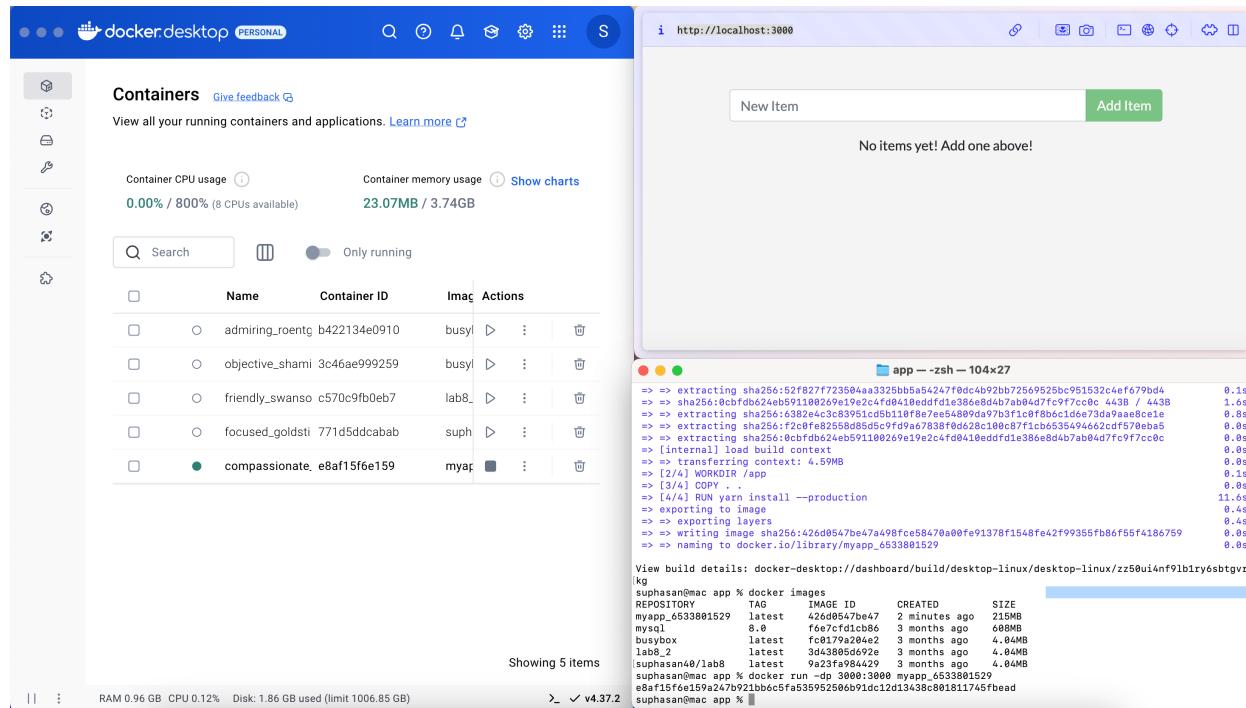
6. ทำการ Start ตัว Container ของแอปพลิเคชันที่สร้างขึ้น โดยใช้คำสั่ง

\$ docker run -dp 3000:3000 <myapp\_รหัสนศ. ไม่มีจีด>

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.สามารถทดลองตั้ง Web application ที่ทำงานอยู่ได้

### 8. ทำการแก้ไข Source code ของ Web application ดังนี้

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

<pre><p> className="text-center">No items yet! Add one above!</p></pre> เป็น

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

- Save ไฟล์ให้เรียบร้อย

### 9. ทำการ Build Docker image โดยใช้คำสั่งเดียวกันกับข้อ 5

### 10. Start และรัน Container ตัวใหม่ โดยใช้คำสั่งเดียวกันกับข้อ 6

[Check point#10] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง)

แสดงคำสั่งและผลลัพธ์ที่ได้ทางหน้าจอ พร้อมกับตอบคำถามต่อไปนี้

- Error ที่เกิดขึ้นหมายความอย่างไร และเกิดขึ้นเพราอะไร

**ตอบ** มีคนใช้ port 3000 อยู่แล้ว เกิดขึ้นเพราไม่ได้ stop หรือ remove container ของเก่าที่ run on port 3000

### 11. ลบ Container ของ Web application เวอร์ชันก่อนแก้ไขออกจากระบบ โดยใช้วิธีใดวิธีหนึ่งดังต่อไปนี้

- ผ่าน Command line interface

- ใช้คำสั่ง \$ docker ps เพื่อดู Container ID ที่ต้องการจะลบ
- Copy หรือบันทึก Container ID ไว้
- ใช้คำสั่ง \$ docker stop <Container ID> ที่ต้องการจะลบ เพื่อหยุดการทำงานของ Container ดังกล่าว

## Lab Worksheet

iv. ใช้คำสั่ง \$ docker rm <Container ID ที่ต้องการจะลบ> เพื่อทำการลบ

b. ผ่าน Docker desktop

i. เปิดหน้าต่าง Containers

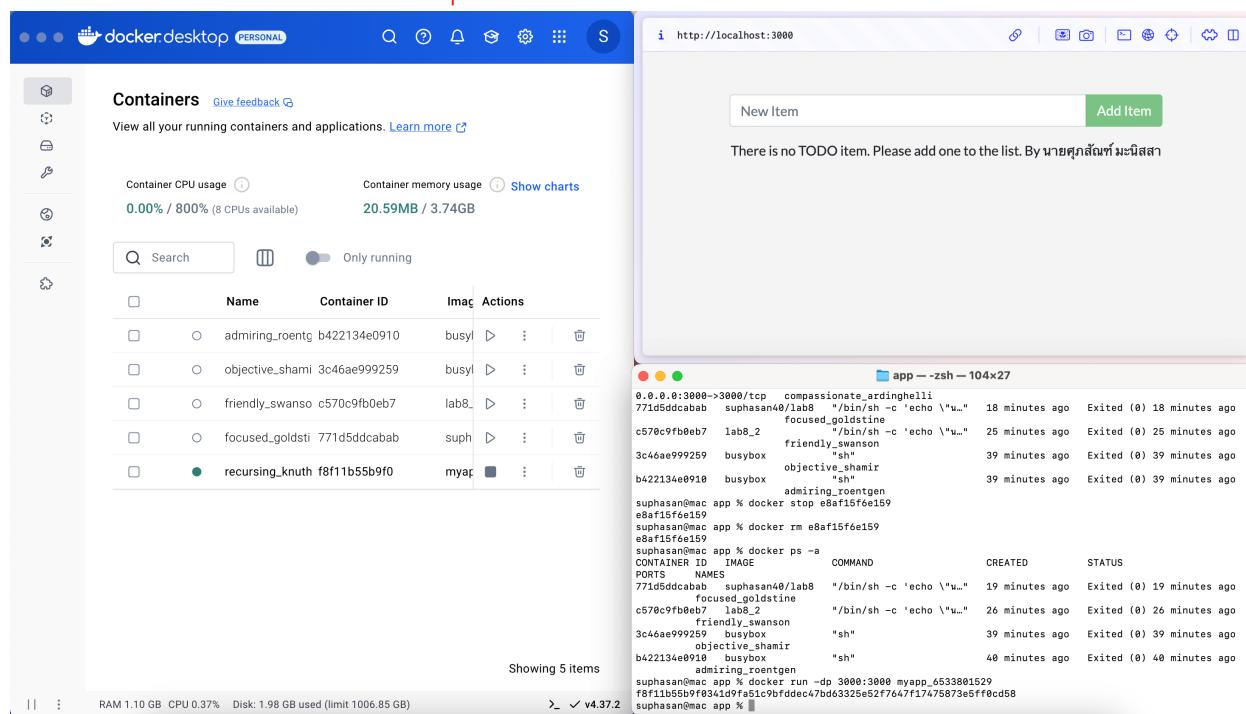
ii. เลือกไอคอนถังขยะในແລງຂອງ Container ที่ต้องการจะลบ

iii. ຢືນຢັນໂດຍກາຈົດ Delete forever

12. Start และรัน Container ตัวใหม่อีกครั้ง โดยใช้คำสั่งเดียวกันกับข้อ 6

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

[Check point#11] Capture หน้าจอ (ທັງໝົດທີ່ມີຈຳກັດ) ແລະ Dashboard ຂອງ Docker desktop



### ແບບຟິກປົງປົນທີ່ 8.5: ເຮັດຕັ້ງສ້າງ Pipeline ອີ່ງໆຍໍສໍາໜັກການ Deploy ດ້ວຍ Jenkins

1. เปิด Command line หรือ Terminal บน Docker Desktop

2. ປັບອຳນວຍຕັ້ງແລະທຳການຮັນ container ໂດຍຜູກພອർຕ

```
$ docker run -p 8080:8080 -p 50000:50000 --restart=on-failure jenkins/jenkins:lts-jdk17
หรື້ອ
```

```
$ docker run -p 8080:8080 -p 50000:50000 --restart=on-failure -v
```

```
jenkins_home:/var/jenkins_home jenkins/jenkins:lts-jdk17
```

3. ບັນທຶກຮັສຜ່ານຂອງ Admin user ໄວໆສໍາໜັກ log-in ໃນຄວັງແກ່

## Lab Worksheet

**[Check point#12] Capture หน้าจอที่แสดงผล Admin password**

● ● ● Lab8\_5 — docker run -p 8080:8080 -p 50000:50000 --restart=on-failure jenkins/jenkins:its-jdk17...

```
2025-01-22 07:04:23.022+0000 [id=39]    INFO    jenkins.InitReactorRunner$1#onAttained: Augmented all extensions
2025-01-22 07:04:23.095+0000 [id=39]    INFO    jenkins.InitReactorRunner$1#onAttained: System config loaded
2025-01-22 07:04:23.095+0000 [id=33]    INFO    jenkins.InitReactorRunner$1#onAttained: System config adapted
2025-01-22 07:04:23.096+0000 [id=33]    INFO    jenkins.InitReactorRunner$1#onAttained: Loaded all jobs
2025-01-22 07:04:23.097+0000 [id=33]    INFO    jenkins.InitReactorRunner$1#onAttained: Configuration for all jobs updated
2025-01-22 07:04:23.112+0000 [id=60]    INFO    hudson.util.Retrier#start: Attempt #1 to do the action check updates server
2025-01-22 07:04:23.349+0000 [id=42]    INFO    jenkins.install.SetupWizard#init:
*****
*****
```

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

`fc5d29dee43645929038be8a9edeeb55`

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

```
*****
```

```
2025-01-22 07:04:29.304+0000 [id=42]    INFO    jenkins.InitReactorRunner$1#onAttained: Completed initialization
2025-01-22 07:04:29.312+0000 [id=25]    INFO    hudson.lifecycle.Lifecycle#onReady: Jenkins is fully up and running
2025-01-22 07:04:31.564+0000 [id=60]    INFO    h.m.DownloadService$Downloadable#load: Obtained the updated data file for hudson.tasks.Maven.MavenInstaller
2025-01-22 07:04:31.565+0000 [id=60]    INFO    hudson.util.Retrier#start: Performed the action check updates server successfully at the attempt #1
```

4. เมื่อได้รับการยืนยันว่า Jenkins is fully up and running ให้เปิดбраузอร์ และป้อนที่อยู่เป็น `localhost:8080`
5. ทำการ Unlock Jenkins ด้วยรหัสผ่านที่ได้ในข้อที่ 3
6. สร้าง Admin User โดยใช้ username เป็นชื่อจริงของนักศึกษาพร้อมรหัสสีตัวท้าย เช่น somsri\_3062

**[Check point#13] Capture หน้าจอที่แสดงผลการตั้งค่า**

## Lab Worksheet

Getting Started

## Create First Admin User

Username

Password

Confirm password

Full name

E-mail address

Jenkins 2.479.3

Skip and continue as admin

**Save and Continue**

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

Welcome to Jenkins!

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.

Start building your software project

Create a job

Set up a distributed build

Set up an agent

Configure a cloud

Learn more about distributed builds

Build Queue

No builds in the queue.

Build Executor Status

0/2

REST API Jenkins 2.479.3

9. เลือก Manage Jenkins และไปที่เมนู Plugins

## Lab Worksheet

The screenshot shows the Jenkins Manage Jenkins page. In the top right corner, there is a message: "It appears that your reverse proxy set up is broken." Below this, the "System Configuration" section is visible, containing several tabs: Build Queue (No builds in the queue), System (Configure global settings and paths), Tools (Configure tools, their locations and automatic installers), Plugins (Add, remove, disable or enable plugins that can extend the functionality of Jenkins), Nodes (Add, remove, control and monitor the various nodes that Jenkins runs jobs on), Clouds (Add, remove, and configure cloud instances to provision agents on-demand), Appearance (Configure the look and feel of Jenkins), Security (Secure Jenkins; define who is allowed to access/use the system), Credentials (Configure credentials), Credential Providers (Configure the credential providers and types), Users (Create/delete/modify users that can log in to this Jenkins), System Information (Displays various environmental information to assist trouble-shooting), System Log (System log captures output from java.util.logging output related to Jenkins), Load Statistics (Check your resource utilization and see if you need more computers for your builds), and About Jenkins (See the version and license information).

10. ไปที่เมนู Available plugins และเลือกติดตั้ง Robotframework เพิ่มเติม

The screenshot shows the Jenkins Manage Jenkins > Plugins page. In the top right corner, there is a search bar with the text "robot". Below this, the "Available plugins" section is visible, listing the "Robot Framework 5.0.0" plugin. The plugin details are shown: Name: Robot Framework 5.0.0, Published by: Build Reports, Released: 2 mo 7 days ago. A description below states: "This publisher stores Robot Framework test reports for builds and shows summaries of them in project and build views along with trend graph." There is an "Install" button at the top right of the plugin card.

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

## Lab Worksheet

New Item

Enter an item name

Select an item type

- Freestyle project**  
Classic general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.
- Pipeline**  
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
- Multi-configuration project**  
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.
- Folder**  
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.
- Multibranch Pipeline**  
Creates a set of Pipeline projects according to detected branches in one SCM repository.
- Organization Folder**  
Creates a set of multibranch project subfolders by scanning for repositories.

OK

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

Description: Lab 8.5

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

Build Trigger: เลือกแบบ Build periodically แล้วกำหนดให้ build ทุก 15 นาที

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

[Check point#14] Capture หน้าจอแสดงการตั้งค่า พิริยมกับตอบคำถามต่อไปนี้

## Lab Worksheet

The screenshot shows the Jenkins 'Configure' screen for a job named 'UAT'. The 'Build Steps' section contains one step: 'Execute shell' with the command 'robot UAT-Lab7-001.robot'. The 'Post-build Actions' section is empty. At the bottom are 'Save' and 'Apply' buttons.

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

**ต่อไป** robot UAT-Lab7-001.robot

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

ระบบได้จัดทำไว้ที่เก็บไฟล์ผลการทดสอบโดย Robot framework ในรูป xml และ html -> ตั้งค่า Threshold เป็น % ของจำนวนทดสอบที่ไม่ผ่านแล้วนับว่าซอฟต์แวร์มีปัญหา -> ตั้งค่า Threshold เป็น %

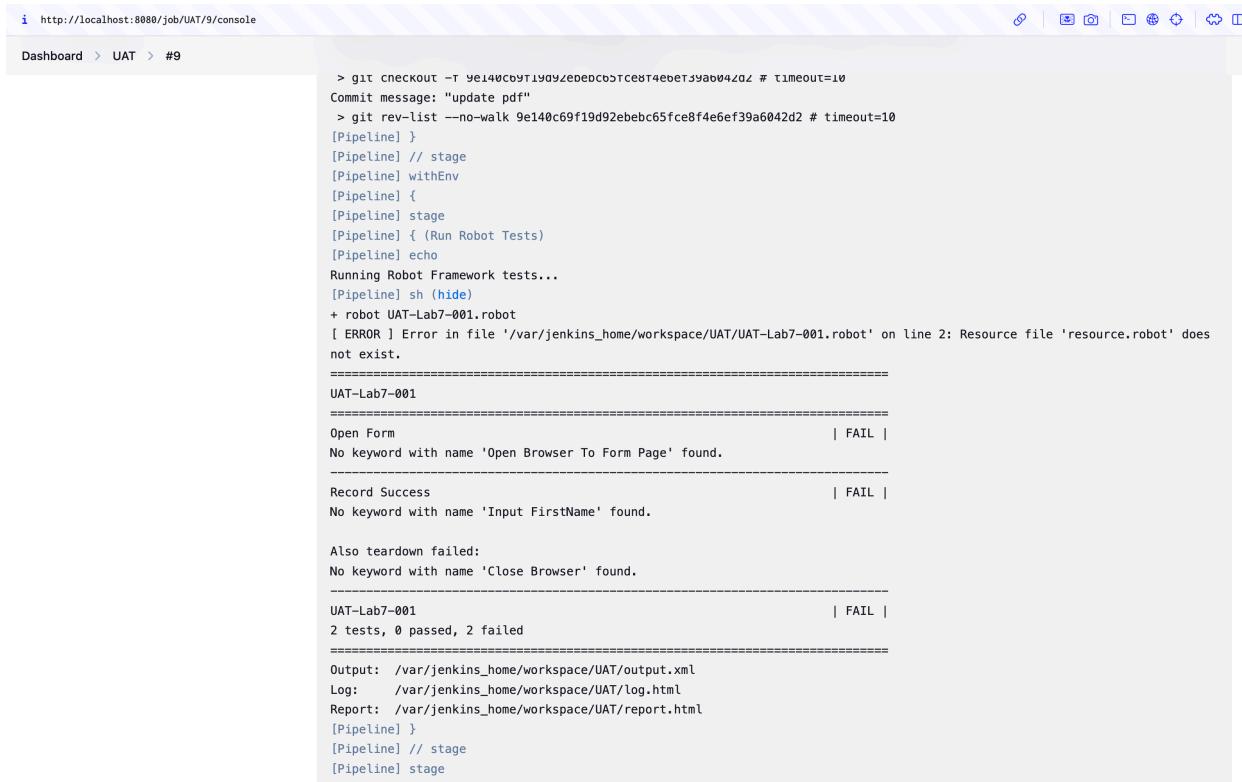
ของการทดสอบที่ผ่านแล้วนับว่าซอฟต์แวร์มีอยู่ในสถานะที่สามารถนำไปใช้งานได้ ( เช่น 20, 80 )

13. กด Apply และ Save

14. ล้าง Build Now

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

## Lab Worksheet



```

http://localhost:8080/job/UAT/9/console

Dashboard > UAT > #9

> git checkout -t 9e140c69f19d92ebcbc65fce8f4e6ef39a6042d2 # timeout=10
Commit message: "update pdf"
> git rev-list --no-walk 9e140c69f19d92ebcbc65fce8f4e6ef39a6042d2 # timeout=10
[Pipeline]
[Pipeline] // stage
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Run Robot Tests)
[Pipeline] echo
Running Robot Framework tests...
[Pipeline] sh (hide)
+ robot UAT-Lab7-001.robot
[ERROR] Error in file '/var/jenkins_home/workspace/UAT/UAT-Lab7-001.robot' on line 2: Resource file 'resource.robot' does not exist.
=====
UAT-Lab7-001
=====
Open Form | FAIL |
No keyword with name 'Open Browser To Form Page' found.
-----
Record Success | FAIL |
No keyword with name 'Input FirstName' found.
-----
Also teardown failed:
No keyword with name 'Close Browser' found.
-----
UAT-Lab7-001 | FAIL |
2 tests, 0 passed, 2 failed
=====
Output: /var/jenkins_home/workspace/UAT/output.xml
Log: /var/jenkins_home/workspace/UAT/log.html
Report: /var/jenkins_home/workspace/UAT/report.html
[Pipeline]
[Pipeline] // stage
[Pipeline] stage

```