

Lab Worksheet

ชื่อ-นามสกุล น.ส.สุวรินทร์ โพธิ์ตาก รหัสนักศึกษา 653380028-0 Section 2

Lab#8 – Software Deployment Using Docker

วัตถุประสงค์การเรียนรู้

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

Pre-requisite

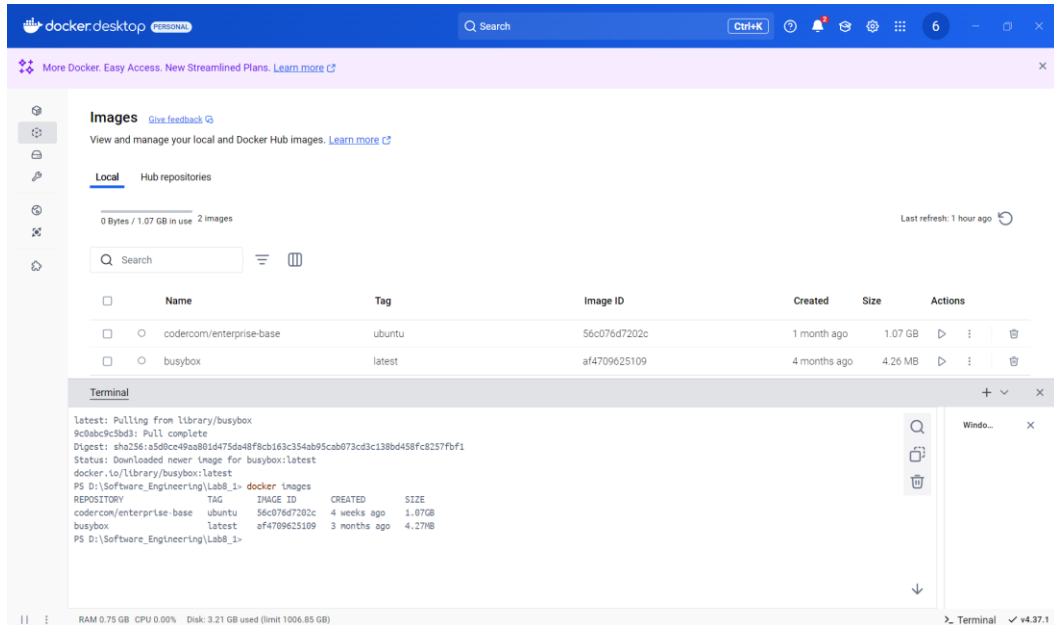
1. ติดตั้ง Docker desktop ลงบนเครื่องคอมพิวเตอร์ โดยดาวน์โหลดจาก <https://www.docker.com/get-started>
2. สร้าง Account บน Docker hub (<https://hub.docker.com/signup>)
3. กำหนดให้ \$ หมายถึง Command prompt และ <> หมายถึง ให้ป้อนค่าของพารามิเตอร์ที่กำหนด

แบบฝึกปฏิบัติที่ 8.1 Hello world - รัน Container จาก Docker image

1. เปิดใช้งาน Docker desktop และ Login ด้วย Username และ Password ที่ลงทะเบียนกับ Docker Hub เอาไว้
1. เปิด Command line หรือ Terminal บน Docker Desktop จากนั้นสร้าง Directory ชื่อ Lab8_1
2. ย้ายตำแหน่งปัจจุบันไปที่ Lab8_1 เพื่อใช้เป็น Working directory
3. ป้อนคำสั่ง \$ docker pull busybox หรือ \$ sudo docker pull busybox สำหรับกรณีที่ติดปัญหา Permission denied
(หมายเหตุ: BusyBox เป็น software suite ที่รองรับคำสั่งบางอย่างบน Unix - <https://busybox.net>)
4. ป้อนคำสั่ง \$ docker images

Lab Worksheet

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



(1) สิ่งที่อยู่ภายใต้คอลัมน์ Repository คืออะไร

ตอบ ชื่อcontainers codercom/enterprise-base และ busybox

(2) Tag ที่ใช้บ่งบอกถึงอะไร

ตอบ latest ระบุเวอร์ชัน ซึ่งถึงว่าเป็นเวอร์ชันที่ใช้

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

```

PS C:\Users\acer> cd D:\Software_Engineering\Lab8_1
PS D:\Software_Engineering\Lab8_1> docker run busybox
PS D:\Software_Engineering\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 27 15:27 .
drwxr-xr-x 1 root root      4096 Jan 27 15:27 ..
drwxr-xr-x 2 root root      4096 Sep 26 21:31 lib
drwxr-xr-x 2 root root      4096 Sep 26 21:31 lib
drwxr-xr-x 2 root root      4096 Sep 26 21:31 lib
x 2 root root      4096 Sep 26 21:31 lib drwxr-xr-x 2 root root      4096 Sep 26 21:31 lib drwxr-xr-x 2 root r
oot 4096 Sep 26 21:31 lib drwxr-xr-x 2 root root      4096 Sep 26 21:31 lib drwxr-xr-x 2 root r
21:31 lib drwxr-xr-x 2 root root      4096 Sep 26 21:31 lib lrwxrwxrwx 1 root root      3 Sep 26 21:31 lib64 -> lib
drwxr-xr-x 2 root root      4096 Sep 26 21:31 lib lrwxrwxrwx 1 root root      4096 Sep 26 21:31 lib
lrwxrwxrwx 1 root root      3 Sep 26 21:31 lib64 -> lib
dr-xr-xr-x 244 root root      0 Jan 27 15:27 proc drwxr-xr-x 2 root root      4096 Sep 26 21:31 lib lrwxrwxrwx 1 root
/ # exit
PS D:\Software_Engineering\Lab8_1> docker run busybox echo "Hello Suwarin Photak 653380028-0 from busybox"
Hello Suwarin Photak 653380028-0 from busybox
PS D:\Software_Engineering\Lab8_1> docker ps -a

```

```

PS D:\Software_Engineering\Lab8_1> docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED              STATUS              PORTS          NAMES
c56ede5ff193   busybox   "echo 'Hello Suwarin..." 15 seconds ago      Exited (0) 14 seconds ago                modest_diffie44fcd21c7e13 busybox "sh"
About a minute ago   Exited (0) About a minute ago                elated_bohr
4514c4bb5db4   busybox   "sh"                    2 minutes ago       Exited (0) 2 minutes ago                intelligent_wing
PS D:\Software_Engineering\Lab8_1>

```

(1) เมื่อใช้ option -it ในคำสั่ง run ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป

ตอบ เปิดคอนเทนเนอร์ที่ผู้ใช้งานสามารถปฏิสัมพันธ์กับ terminal

(2) คอลัมน์ STATUS จากการรันคำสั่ง docker ps -a แสดงถึงข้อมูลอะไร

ตอบ แสดงสถานะปัจจุบันของคอนเทนเนอร์ Docker คือ CONTAINER ID, IMAGE, COMMAND, CREATED, STATUS, PORTS, NAMES

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

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

```

PS D:\Software_Engineering\Lab8_1> docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED              STATUS              PORTS          NAMES
c56ede5ff193   busybox   "echo 'Hello Suwarin..." 15 seconds ago      Exited (0) 14 seconds ago                modest_diffie44fcd21c7e13 busybox "sh"
3 busybox "sh"                    About a minute ago   Exited (0) About a minute ago                elated_bohr
4514c4bb5db4   busybox   "sh"                    2 minutes ago       Exited (0) 2 minutes ago                intelligent_wing
PS D:\Software_Engineering\Lab8_1> ^C
PS D:\Software_Engineering\Lab8_1> ^C
PS D:\Software_Engineering\Lab8_1> docker rm c56ede5ff193
c56ede5ff193
PS D:\Software_Engineering\Lab8_1> docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED              STATUS              PORTS          NAMES
44fcd21c7e13   busybox   "sh"                    10 minutes ago      Exited (0) 9 minutes ago                elated_bohr
4514c4bb5db4   busybox   "sh"                    10 minutes ago      Exited (0) 10 minutes ago                intelligent_wing
PS D:\Software_Engineering\Lab8_1>

```

Lab Worksheet

แบบฝึกปฏิบัติที่ 8.2: สร้าง Docker file และ Docker image

1. เปิดใช้งาน Docker desktop และ Login ด้วย Username และ Password ที่ลงทะเบียนกับ Docker Hub เอาไว้
2. เปิด Command line หรือ Terminal จากนั้นสร้าง Directory ชื่อ Lab8_2
3. ย้ายตำแหน่งปัจจุบันไปที่ Lab8_2 เพื่อใช้เป็น Working directory
4. สร้าง Dockerfile.swp ไว้ใน Working directory

สำหรับเครื่องที่ใช้ระบบปฏิบัติการวินโดวส์ (Windows) บันทึกคำสั่งต่อไปนี้ลงในไฟล์ โดยใช้ Text Editor ที่มี

```
FROM busybox
```

```
CMD echo "Hi there. This is my first docker image."
```

```
CMD echo "ชื่อ-นามสกุล รหัสนักศึกษา ชื่อเล่น"
```

สำหรับเครื่องที่ใช้ระบบปฏิบัติการ MacOS หรือ Linux บนหน้าต่าง Terminal และบันทึกคำสั่งต่อไปนี้

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

```
FROM busybox
```

```
CMD echo "Hi there. This is my first docker image."
```

```
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

```

Terminal
[+] Building 0.6s (5/5) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 294B
=> [internal] load metadata for docker.io/library/busybox:latest
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/1] FROM docker.io/library/busybox:latest
=> exporting to image
=> => exporting layers
=> => writing image sha256:9b134567f8f653716dafba90469b08c531720c80ee27e1f7e0b96bd307525674
=> => naming to docker.io/library/lab8_2

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/svo0cwtbvvoae10kfoujt3kqd

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)
PS D:\Software_Engineering\Lab8_2> docker run lab8_2
“ชื่อ นางสาวสุรินทร์ โพธิ์ตาก รหัสนักศึกษา 653380028-0 ชื่อเล่น เม็ก”
PS D:\Software_Engineering\Lab8_2>
  
```

(1) คำสั่งที่ใช้ในการ run คือ

ตอบ docker run lab8_2

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

ตอบ ใช้สำหรับตั้งชื่อ image ที่สร้างขึ้น

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”

Lab Worksheet

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

```
PS D:\Software_Engineering\Lab8_3> docker build -t 6533800280suwarinphotak/lab8_3 .
```

```
[+] Building 0.7s (5/5) FINISHED
```

```
=> [internal] load build definition from Dockerfile
```

```
=> => transferring dockerfile: 234B
```

```
=> [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:d94c507a4c2b5f7c73cee80d63fc6267b2d676c2c3d2ca4fcfc82321a8e0b53f
```

```
=> => naming to docker.io/6533800280suwarinphotak/lab8_3
```

```
docker:desktop-linux
```

```
0.1s
```

```
0.0s
```

```
0.0s
```

```
0.1s
```

```
0.0s
```

```
0.0s
```

```
0.1s
```

```
0.0s
```

```
0.0s
```

```
0.0s
```

```
View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/8w8weys1xyr2agg42dwpharc6
```

```
3 warnings found (use docker --debug to expand):
```

```
- MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the same stage because only the last one will be used (line 2)
```

```
- JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 3)
```

```
- JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 2)
```

```
PS D:\Software_Engineering\Lab8_3> docker run 6533800280suwarinphotak lab8_3
```

```
docker: Error response from daemon: failed to create task for container: failed to create shim task: OCI runtime create failed: runc crea
```

Lab Worksheet

6. ทำการ Push ตัว Docker image ไปไว้บน Docker Hub โดยการใช้คำสั่ง

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

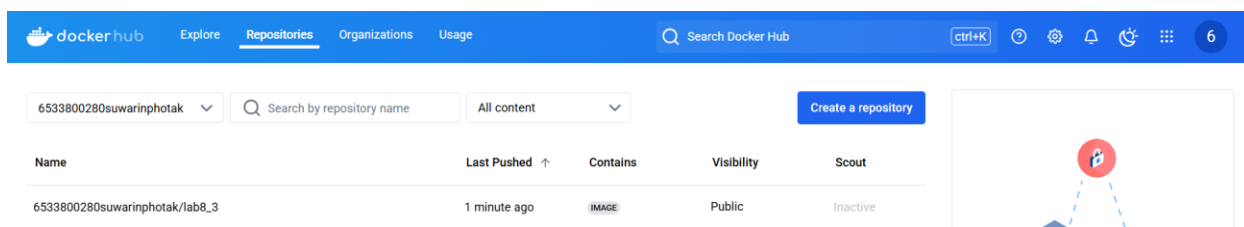
ในกรณีที่ติดปัญหาไม่ได้ Login ไว้ก่อน ให้ใช้คำสั่งต่อไปนี้ เพื่อ Login ก่อนทำการ Push

```
$ docker login แล้วป้อน Username และ Password ตามที่ระบุใน Command prompt หรือใช้คำสั่ง
```

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

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

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



แบบฝึกปฏิบัติที่ 8.4: การ Build แอปพลิเคชันจาก Container image และการ Update แอปพลิเคชัน

1. เปิด Command line หรือ Terminal จากนั้นสร้าง Directory ชื่อ Lab8_4

2. ทำการ Clone ซอร์สโค้ดของเว็บแอปพลิเคชันจาก GitHub repository

<https://github.com/docker/getting-started.git> ลงใน Directory ที่สร้างขึ้น โดยใช้คำสั่ง

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

3. เปิดดูองค์ประกอบภายใน getting-started/app เมื่อพบไฟล์ package.json ให้ใช้ Text editor ในการเปิดอ่าน

[Check point#7] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงที่อยู่ของ Source code ที่ Clone มาและเนื้อหาของไฟล์ package.json

Lab Worksheet

The screenshot shows the Docker Desktop interface. On the left is a sidebar with navigation options: Containers, Images, Volumes, Builds, Docker Hub, Docker Scout, and Extensions. The 'Images' tab is selected, displaying a table of local Docker images:

Name	Tag	Image ID
codercor/enterprise-base	ubuntu	56c076d72
busybox	latest	af4709625
lab8_2	latest	9b134567f

Below the table is a 'Terminal' section showing the command prompt in the directory D:\Software_Engineering. The user has run the following commands:

```
PS D:\Software_Engineering> cd D:\Software_Engineering\Lab8_4
PS D:\Software_Engineering\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 | 10.44 MiB/s, done.
Resolving deltas: 100% (523/523), done.
PS D:\Software_Engineering\Lab8_4> cd getting-started/app
PS D:\Software_Engineering\Lab8_4\getting-started\app> notepad package.json
PS D:\Software_Engineering\Lab8_4\getting-started\app> 
```

On the right, a 'package.json' file is open in a text editor, showing the following content:

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

4. ภายใต้ getting-started/app ให้สร้าง Dockerfile พร้อมกับใส่เนื้อหาดังต่อไปนี้ลงไปไฟล์

FROM node:18-alpine

WORKDIR /app

COPY . .

RUN yarn install --production

CMD ["node", "src/index.js"]

EXPOSE 3000

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

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

[Check point#8] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง)

แสดงคำสั่งและผลลัพธ์ที่ได้ทางหน้าจอ

Lab Worksheet

```

Terminal
=> => sha256:6e804119c3884fc5782795bf0d2adc89201c63105aace8647b17a7bcebb385e 1.72kB / 1.72kB 0.0s
=> => sha256:dcfb77b337595b6f4d214e4eed84f230eeffe0e4ac03a50380d573e289b9e5e40 6.18kB / 6.18kB 0.0s
=> => sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e0da02525e63167c 40.01MB / 40.01MB 2.7s
=> => sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15bf426d01bc1 1.26MB / 1.26MB 1.1s
=> => sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c24403d592134f0 3.64MB / 3.64MB 1.5s
=> => sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6f368990ca771 444B / 444B 1.6s
=> => extracting sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c24403d592134f0 0.4s
=> => extracting sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e0da02525e63167c 1.8s
=> => extracting sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15bf426d01bc1 0.1s
=> => extracting sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6f368990ca771 0.0s
=> [internal] load build context 1.3s
=> => transferring context: 4.62MB 1.1s
=> [2/4] WORKDIR /app 0.9s
=> [3/4] COPY . . 0.1s
=> [4/4] RUN yarn install --production 18.7s
=> exporting to image 1.4s
=> exporting layers 1.3s
=> writing image sha256:b638b392d5af03ee4f54a6042b6278ef2d12675fe1ab640ebce564c1e266ae69 0.0s
=> naming to docker.io/library/myapp_6533800280 0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/9mkvjo8q2qwl01c8ns7z2pfvj
PS D:\Software_Engineering\Lab8_4\getting-started\app>

```

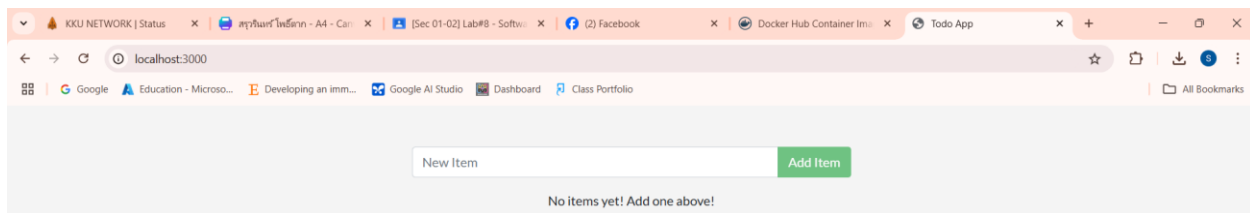
6. ทำการ Start ตัว Container ของแอปพลิเคชันที่สร้างขึ้น โดยใช้คำสั่ง

\$ docker run -dp 3000:3000 <myapp_รหัสศ. ไม่มีขีด>

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

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

View build details: <docker-desktop://dashboard/build/desktop-linux/desktop-linux/9mkvjo8q2qwl01c8ns7z2pfvj>
 PS D:\Software_Engineering\Lab8_4\getting-started\app> docker run -dp 3000:3000 myapp_6533800280
 52d56bebaa7ad686fd4eaae004e2b89e0b1fb762e612c1b64d49db7da8df80d7
 PS D:\Software_Engineering\Lab8_4\getting-started\app> █



หมายเหตุ: ศ.สามารถทดลองเล่น Web application ที่ทำงานอยู่ได้

8. ทำการแก้ไข Source code ของ Web application ดังนี้

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

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

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

ชื่อและนามสกุลของนักศึกษา</p>

b. Save ไฟล์ให้เรียบร้อย

Lab Worksheet

9. ทำการ Build Docker image โดยใช้คำสั่งเดียวกันกับข้อ 5

10. Start และรัน Container ตัวใหม่ โดยใช้คำสั่งเดียวกันกับข้อ 6

[Check point#10] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง)

แสดงคำสั่งและผลลัพธ์ที่ได้ทางหน้าจอ พร้อมกับตอบคำถามต่อไปนี้

```
=> => transferring dockerfile: 156B 0.0s
=> [internal] load metadata for docker.io/library/node:18-alpine 2.5s
=> [auth] library/node:pull token for registry-1.docker.io 0.0s
=> [internal] load .dockerignore 0.1s
=> => transferring context: 2B 0.0s
=> [1/4] FROM docker.io/library/node:18-alpine@sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d975e9059dd066be3e274fbb25 0.0s
=> [internal] load build context 0.1s
=> => transferring context: 8.16kB 0.0s
=> CACHED [2/4] WORKDIR /app 0.0s
=> [3/4] COPY . . 0.3s
=> [4/4] RUN yarn install --production 17.3s
=> exporting to image 1.2s
=> => exporting layers 1.2s
=> => writing image sha256:7564fab53a52d3aa5987bb1f83fd4ebe6e43fff8e7da7f0b1cf087cb495ced84 0.0s
=> => naming to docker.io/library/myapp_6533800280 0.0s
```

View build details: docker_desktop://dashboard/build/desktop-linux/desktop-linux/vjab5di5gbd2vpcynfiha6mf

PS D:\Software_Engineering\Lab8_4\getting-started\app> docker run -dp 3000:3000 myapp_6533800280

1b705cb4b7f02f3eccdb230d46b038459e68cf6483a2cb925c35fc97f5a85909

docker: Error response from daemon: driver failed programming external connectivity on endpoint focused_goldstine (03be8e0c42a7b97e342f1bee0f3d8996268dc1d4727a62698ff6cb6477a8c9af): Bind for 0.0.0.0:3000 failed: port is already allocated.

PS D:\Software_Engineering\Lab8_4\getting-started\app> █

(1) Error ที่เกิดขึ้นหมายความว่าอย่างไร และเกิดขึ้นเพราะอะไร

ตอบ เพราะใช้port 3000 อยู่ จึงใช้ไม่ได้

11. ลบ Container ของ Web application เวอร์ชันก่อนแก้ไขออกจากระบบ โดยใช้วิธีใดวิธีหนึ่งดังต่อไปนี้

a. 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
- ใช้คำสั่ง `$ docker rm <Container ID ที่ต้องการจะลบ>` เพื่อทำการลบ

b. ผ่าน Docker desktop

- ไปที่หน้าต่าง Containers
- เลือกไอคอนถังขยะในแถวของ Container ที่ต้องการจะลบ
- 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

```

=> [internal] load build context                                0.1s
=> => transferring context: 8.16kB                             0.0s
=> CACHED [2/4] WORKDIR /app                                  0.0s
=> [3/4] COPY . .                                             0.3s
=> [4/4] RUN yarn install --production                       17.3s
=> exporting to image                                         1.2s
=> => exporting layers                                         1.2s
=> => writing image sha256:7564fab53a52d3aa5987bb1f83fd4ebe6e43fff8e7da7f0b1cf087cb495ced84 0.0s
=> => naming to docker.io/library/myapp_6533800280           0.0s

See 'docker rm --help'.

Usage:  docker rm [OPTIONS] CONTAINER [CONTAINER...]

Remove one or more containers
PS D:\Software_Engineering\Lab8_4\getting-started\app> docker rm 52d56ebee7a
52d56ebee7a
PS D:\Software_Engineering\Lab8_4\getting-started\app> docker run -dp 3000:3000 myapp_6533800280
8441cdacd72ca233a85470b3e5510a22a88343df71170cc58f195ecb601a7744
PS D:\Software_Engineering\Lab8_4\getting-started\app>
  
```

The browser screenshot shows a web application with a "New Item" button and a message: "There is no TODO item. Please add one to the list. By นางสาวสุรินทร์ โพธิ์ตาก 653380028-0".

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

```
99b692fb2dce401fa63047702eafbde8
```

[Check point#12] Capture หน้าจอที่แสดงผล Admin password

Lab Worksheet

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

99b692fb2dce401fa63047702eafbde8

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


```
2025-01-27 16:34:18.075+0000 [id=54] INFO jenkins.InitReactorRunner$1#onAttained: Completed initialization
2025-01-27 16:34:18.115+0000 [id=25] INFO hudson.lifecycle.Lifecycle#onReady: Jenkins is fully up and running
2025-01-27 16:34:20.031+0000 [id=68] INFO h.m.DownloadService$Downloadable#load: Obtained the updated data file for hudson.tasks.Maven.MavenInstaller
2025-01-27 16:34:20.032+0000 [id=68] INFO hudson.util.Retrier#start: Performed the action check updates server successfully at the attempt
```

Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

`/var/jenkins_home/secrets/initialAdminPassword`

Please copy the password from either location and paste it below.

Administrator password

.....

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

[Check point#13] Capture หน้าจอที่แสดงผลการตั้งค่า

Lab Worksheet

localhost:8080

Education - Microso... Developing an imm... Google AI Studio Dashboard Class Portfolio

Getting Started

Create First Admin User

Username

Password

Confirm password

Full name

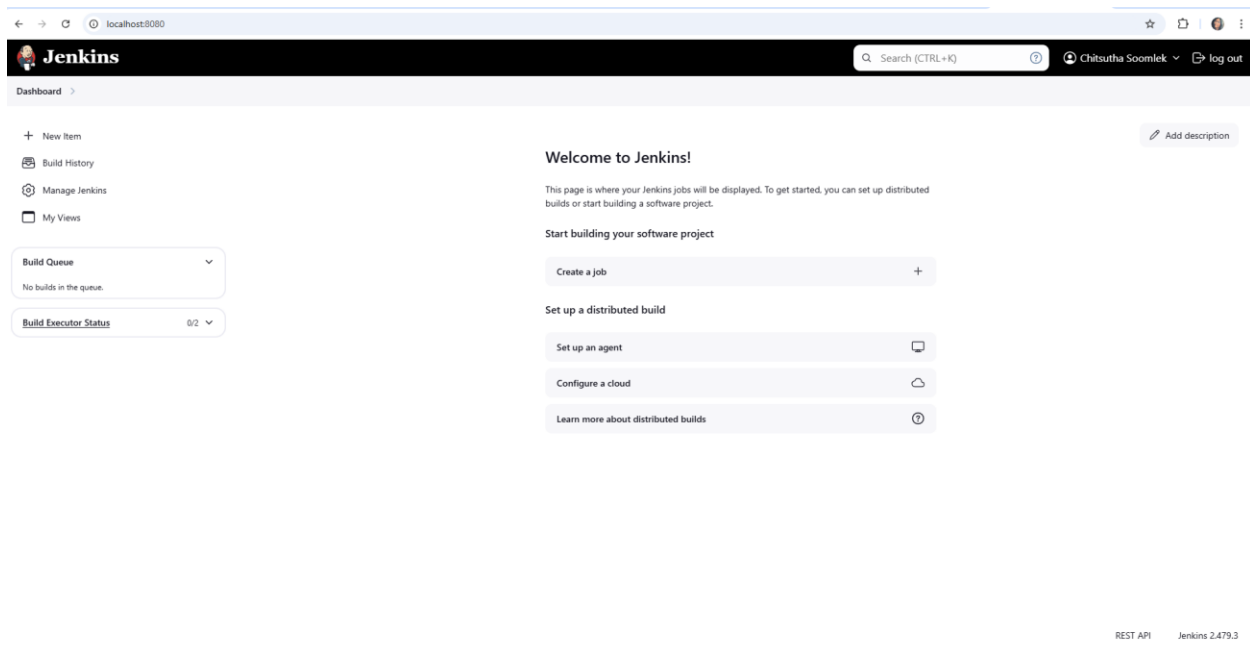
E-mail address

Jenkins 2.479.3

[Skip and continue as admin](#) [Save and Continue](#)

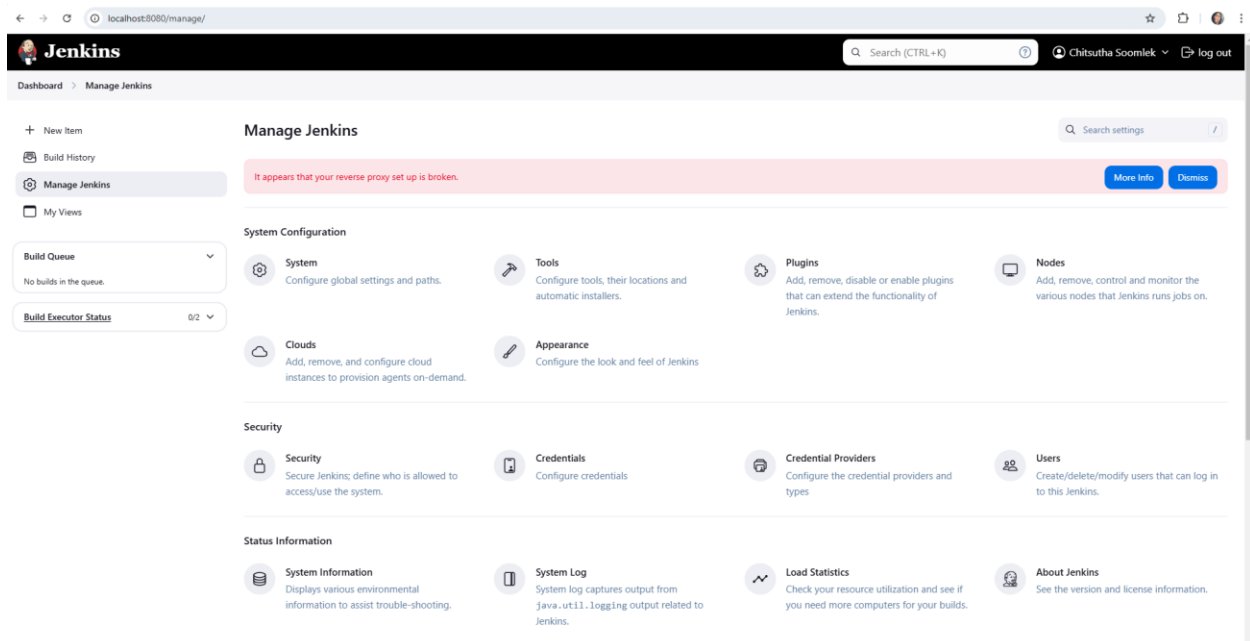
- กำหนด Jenkins URL เป็น <http://localhost:8080/lab8>
- เมื่อติดตั้งเรียบร้อยแล้วจะพบหน้าจอ Dashboard ดังแสดงในภาพ

Lab Worksheet



The screenshot shows the Jenkins Dashboard at localhost:8080. The top navigation bar includes the Jenkins logo, a search bar, and a user profile for Chitsutha Soomlek with a log out button. The left sidebar contains links for New Item, Build History, Manage Jenkins, and My Views. The main content area features a 'Welcome to Jenkins!' message, a 'Start building your software project' section with a 'Create a job' button, and a 'Set up a distributed build' section with buttons for 'Set up an agent', 'Configure a cloud', and 'Learn more about distributed builds'. The bottom right corner displays 'REST API' and 'Jenkins 2.479.3'.

9. เลือก Manage Jenkins แล้วไปที่เมนู Plugins



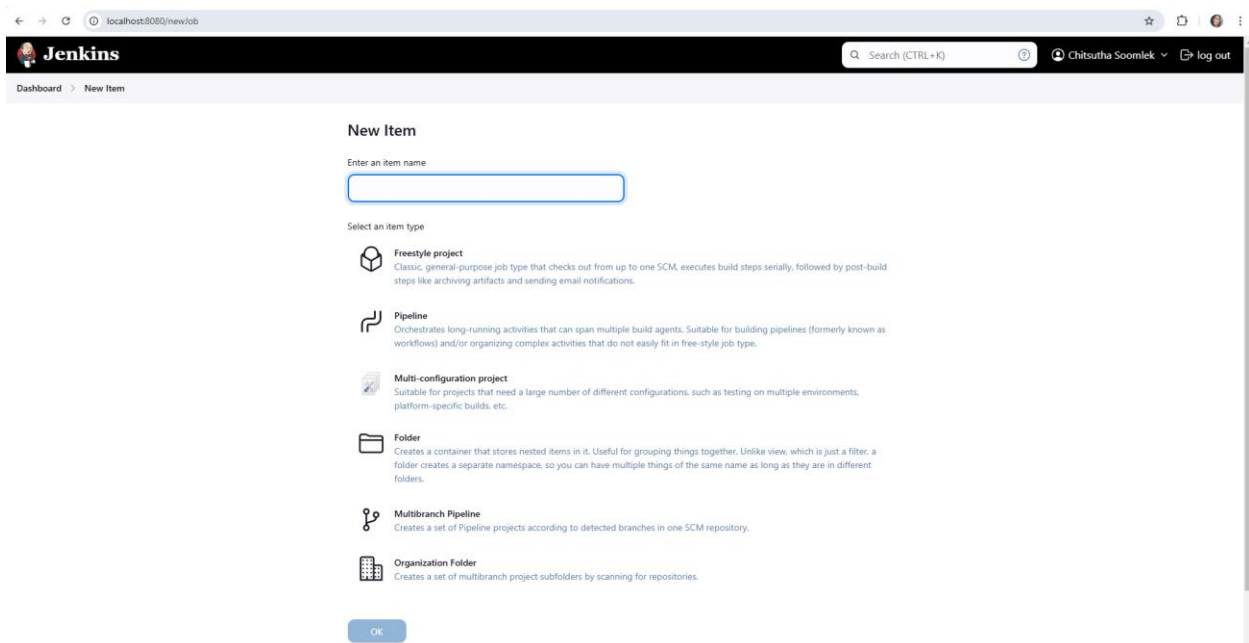
The screenshot shows the 'Manage Jenkins' page at localhost:8080/manage/. The top navigation bar is similar to the dashboard. The left sidebar highlights 'Manage Jenkins'. The main content area has a 'Manage Jenkins' title and a search settings bar. A red warning banner states 'It appears that your reverse proxy set up is broken.' with 'More Info' and 'Dismiss' buttons. Below this, the 'System Configuration' section includes links for System, Tools, Plugins, Nodes, Clouds, and Appearance. The 'Security' section includes links for Security, Credentials, Credential Providers, and Users. The 'Status Information' section includes links for System Information, System Log, Load Statistics, and About Jenkins.

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

Lab Worksheet

The screenshot shows the Jenkins configuration page for a job named 'Lab 8.5'. The page is divided into several sections: General, Source Code Management, Build Triggers, Build Environment, Build Steps, and Post-build Actions.

General

- Description: Lab 8.5
- Plain text: [Preview](#)
- ☐ Discard old builds
- ☒ GitHub project
 - Project url: https://github.com/MGXEE/Lab8_5.git/
 - [Advanced](#)
- ☐ This project is parameterized
- ☐ Throttle builds
- ☐ Execute concurrent builds if necessary
- [Advanced](#)

Source Code Management

- ☒ None
- ☐ Git

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 Monday, January 27, 2025 at 5:05:35 PM Coordinated Universal Time; would next run at Monday, January 27, 2025 at 5:20:35 PM 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
- ☐ Terminate a build if it's stuck
- ☐ With Ant

Build Steps

- Execute shell**
 - Command: `robot Test1_1-robot`
 - [See the list of available environment variables](#)
 - [Advanced](#)
 - [Add build step](#)

Post-build Actions

- Publish Robot Framework test results**
 - Directory of Robot output: `.`
 - Path to directory containing robot xml and html files (relative to build workspace): `.`
 - [Advanced](#) [Edited](#)
 - Thresholds for build result
 - ☒ `%`: `20.0`
 - ☒ `%`: `80.0`
 - ☒ DEPRECATED! THIS FLAG DOES NOTHING! - Use thresholds for critical tests only
 - ☐ Include skipped tests in total count for thresholds
 - [Add post-build action](#)

Lab Worksheet

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

ตอบ robot Test1_1.robot

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

ระบุได้เร็คทอรีที่เก็บไฟล์ผลการทดสอบโดย 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

The screenshot shows the Jenkins web interface. The top navigation bar includes the Jenkins logo, a search bar, and user information (Suvarin Photak). The main content area displays the 'Console Output' for a build named 'UAT'. The output text indicates a failure in the 'Execute shell' step due to a missing file. The error message is: 'hudson.AbortException: No files found in path /var/jenkins_home/workspace/UAT with configured filemask: output.xml'. The build status is 'FAILURE'.

```

Started by timer
Running as SYSTEM
Building in workspace /var/jenkins_home/workspace/UAT
[UAT] $ /bin/sh -xe /tmp/jenkins18260125884916678559.sh
+ robot Test1_1.robot
/tmp/jenkins18260125884916678559.sh: 2: robot: not found
Build step 'Execute shell' marked build as failure
Robot results publisher started...
INFO: Checking test criticality is deprecated and will be dropped in a future release!
-Parsing output xml:
failed!
hudson.AbortException: No files found in path /var/jenkins_home/workspace/UAT with configured filemask: output.xml
    at PluginClassLoader for robot/hudson.plugins.robot.RobotParser$RobotParserCallable.invoke(RobotParser.java:81)
    at PluginClassLoader for robot/hudson.plugins.robot.RobotParser$RobotParserCallable.invoke(RobotParser.java:52)
    at hudson.FilePath.act(FilePath.java:1234)
    at hudson.FilePath.act(FilePath.java:1217)
    at PluginClassLoader for robot/hudson.plugins.robot.RobotParser.parse(RobotParser.java:48)
    at PluginClassLoader for robot/hudson.plugins.robot.RobotPublisher.parse(RobotPublisher.java:262)
    at PluginClassLoader for robot/hudson.plugins.robot.RobotPublisher.perform(RobotPublisher.java:286)
    at hudson.tasks.BuildStepCompatibilityLayer.perform(BuildStepCompatibilityLayer.java:88)
    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:181)
    at hudson.model.Executor.run(Executor.java:445)
Finished: FAILURE
  
```