

## Lab Worksheet

ชื่อ-นามสกุล ภัทรพล โพธิ์มา รหัสนักศึกษา 653380145-6 Section 1

## 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

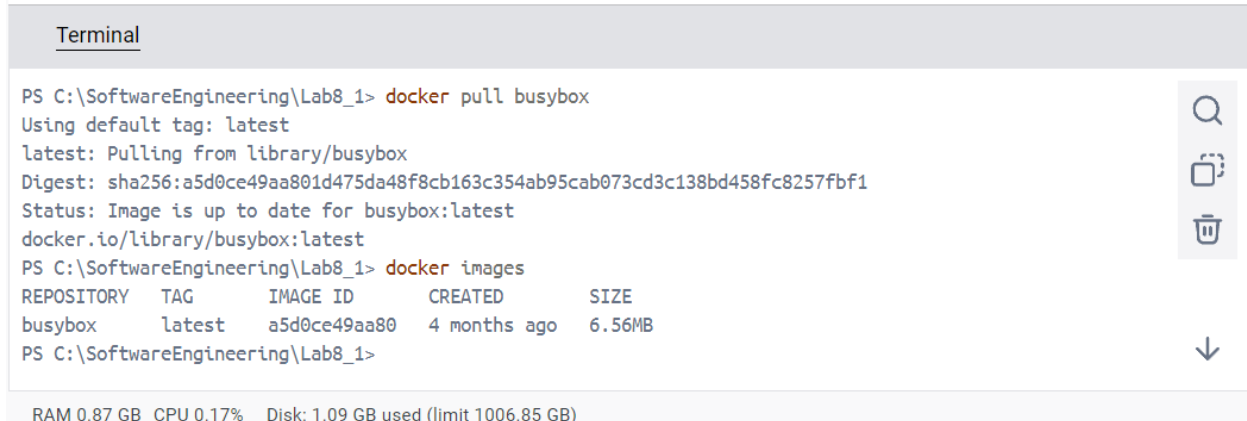
1. ติดตั้ง Docker desktop ลงบนเครื่องคอมพิวเตอร์ โดยดาวน์โหลดจาก <https://www.docker.com/get-started>
2. สร้าง Account บน Docker hub (<https://hub.docker.com/signup>)
3. กำหนดให้ \$ หมายถึง Command prompt และ <> หมายถึง ให้ป้อนค่าของพารามิเตอร์ที่กำหนด

## 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



```

Terminal
PS C:\SoftwareEngineering\Lab8_1> docker pull busybox
Using default tag: latest
latest: Pulling from library/busybox
Digest: sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138bd458fc8257fbf1
Status: Image is up to date for busybox:latest
docker.io/library/busybox:latest
PS C:\SoftwareEngineering\Lab8_1> docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
busybox       latest    a5d0ce49aa80   4 months ago   6.56MB
PS C:\SoftwareEngineering\Lab8_1>

RAM 0.87 GB  CPU 0.17%  Disk: 1.09 GB used (limit 1006.85 GB)

```

- (1) สิ่งที่อยู่ภายใต้คอลัมน์ Repository คืออะไร ชื่อของ Docker image ในภาพคือ busybox
- (2) Tag ที่ใช้บ่งบอกถึงอะไร สถานะเฉพาะของ image ในภาพคือ latest
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

```

PS C:\SoftwareEngineering\Lab8_1> docker run busybox
PS C:\SoftwareEngineering\Lab8_1> docker run -it busybox sh
/ # ls
bin    dev    etc    home   lib    lib64  proc   root   sys    tmp    usr    var
/ # ls -la
total 48

```

**Terminal**

```

bin    dev    etc    home   lib    lib64  proc   root   sys    tmp    usr    var
/ # ls -la
total 48
drwxr-xr-x 1 root    root      4096 Jan 28 10:42 .
drwxr-xr-x 1 root    root      4096 Jan 28 10:42 ..
-rwxr-xr-x 1 root    root         0 Jan 28 10:42 .dockerenv
drwxr-xr-x 2 root    root     12288 Sep 26 21:31 bin
drwxr-xr-x 5 root    root      360 Jan 28 10:42 dev
drwxr-xr-x 1 root    root      4096 Jan 28 10:42 etc
drwxr-xr-x 2 nobody nobody    4096 Sep 26 21:31 home

```

```

PS C:\SoftwareEngineering\Lab8_1> docker run busybox echo "Hello Phattharaphon Phoma from busybox"
Hello Phattharaphon Phoma from busybox
PS C:\SoftwareEngineering\Lab8_1>

```

**Terminal**

```

Hello Phattharaphon Phoma from busybox
PS C:\SoftwareEngineering\Lab8_1> docker ps -a

```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAME
MES						
2da3e35d3a8f	busybox	"echo 'Hello Phattha..."	43 seconds ago	Exited (0) 42 seconds ago		th
irsty_torvalds						
b4bdc539fd75	busybox	"echo"	2 minutes ago	Exited (0) 2 minutes ago		se
rene_dhawan						
548b0fc9a33b	busybox	"sh"	5 minutes ago	Exited (0) 3 minutes ago		pr
iceless_almeida						

- (1) เมื่อใช้ option -it ในคำสั่ง run ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป
- i (interactive): ทำให้สามารถโต้ตอบกับ container ได้ (รักษา connection ระหว่าง terminal และ container)
- t (TTY): จำลอง terminal (pseudo-terminal) ที่เหมือนกับการทำงานบน shell ปกติ

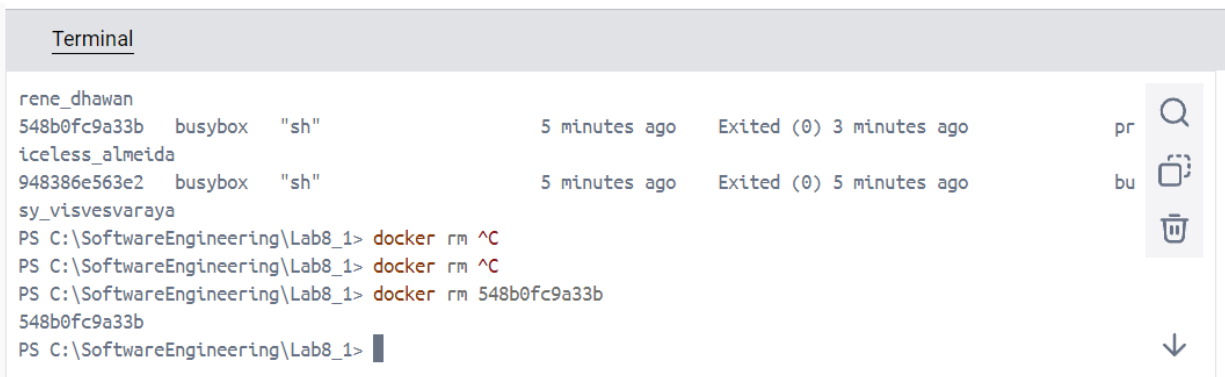
## Lab Worksheet

(2) คอลัมน์ STATUS จากการรันคำสั่ง `docker ps -a` แสดงถึงข้อมูลอะไร

แสดงถึงสถานะของการทำงานของ container และแสดงเวลาของสถานะนั้น

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

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



```

Terminal
rene_dhawan
548b0fc9a33b  busybox  "sh"          5 minutes ago  Exited (0) 3 minutes ago
iceless_almeida
948386e563e2  busybox  "sh"          5 minutes ago  Exited (0) 5 minutes ago
sy_visvesvaraya
PS C:\SoftwareEngineering\Lab8_1> docker rm ^C
PS C:\SoftwareEngineering\Lab8_1> docker rm ^C
PS C:\SoftwareEngineering\Lab8_1> docker rm 548b0fc9a33b
548b0fc9a33b
PS C:\SoftwareEngineering\Lab8_1>

```

## แบบฝึกปฏิบัติที่ 8.2: สร้าง Docker file และ Docker image

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

สำหรับเครื่องที่ใช้ระบบปฏิบัติการวินโดวส์ (Windows) บันทึกคำสั่งต่อไปนี้ลงในไฟล์ โดยใช้ Text Editor ที่มี

FROM busybox

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

CMD echo "ชื่อ-นามสกุล รหัสนักศึกษา ชื่อเล่น"

สำหรับเครื่องที่ใช้ระบบปฏิบัติการ MacOS หรือ Linux บนหน้าต่าง Terminal และป้อนคำสั่งต่อไปนี้

`$ cat > Dockerfile << EOF`

FROM busybox

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

## Lab Worksheet

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

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

The screenshot shows a Docker Desktop terminal window with the following output:

```

PS C:\SoftwareEngineering\Lab8_2> docker build -t myimage .
[+] Building 3.0s (6/6) FINISHED
=> [internal] load build definition from Dockerfile                                0.1s
=> => transferring dockerfile: 141B                                              0.0s
=> [internal] load metadata for docker.io/library/busybox:latest                 0.0s
=> [internal] load .dockerignore                                                 0.0s
=> => transferring context: 2B                                                  0.0s
=> [1/1] FROM docker.io/library/busybox:latest@sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3 2.7s
=> => resolve docker.io/library/busybox:latest@sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3 2.7s
=> [auth] library/busybox:pull token for registry-1.docker.io                  0.0s
=> exporting to image                                                            0.1s
=> => exporting layers                                                            0.0s
=> => exporting manifest sha256:d08f66fb557da1b7658fcf2c7b320b198327975836642596d39e86b7e790467d 0.0s
=> => exporting config sha256:6688eadb68b8effdf001a8cef64616190c1e9e434bef235f5354effcb95f6424 0.0s
=> => exporting attestation manifest sha256:798b953f0fb4231c05ee6c43375c2981144a58a1a1242d08d2e870c 0.0s
=> => exporting manifest list sha256:f96a31dbeb20ac065f516a5ad8710dfb7db4a622a70fb1c7002ff509000a97 0.0s
=> => naming to docker.io/library/myimage:latest                               0.0s
=> => unpacking to docker.io/library/myimage:latest                             0.0s
View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/7vt8ypebofqwpv588sh4z7mpz
  
```

## Lab Worksheet

```
View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/7vt8ypebofqwpv588sh4z7mpz

3 warnings found (use docker --debug to expand):
- 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)
- MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the same stage because only the last one will be used (line 2)
PS C:\SoftwareEngineering\Lab8_2>
my the last one will be used (line 2)
PS C:\SoftwareEngineering\Lab8_2> docker run myimage
"Phattharaphon Phoma"
PS C:\SoftwareEngineering\Lab8_2>
```

(1) คำสั่งที่ใช้ในการ run คือ

Docker run myimage

(2) Option -t ในคำสั่ง \$ docker build ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป  
-t ย่อมาจาก --tag ใช้สำหรับ "ตั้งชื่อ" หรือ "แท็ก" ให้กับ Docker 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"

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."

## Lab Worksheet

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 C:\SoftwareEngineering\Lab8_3> docker build -t phattharaphon01/lab8 .
[+] Building 3.0s (6/6) FINISHED                                docker:desktop-linux
=> [internal] load build definition from Dockerfile            0.0s
=> => transferring dockerfile: 168B                            0.0s
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior rel 0.0s
=> WARN: MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the same s 0.0s
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior rel 0.0s
=> [internal] load metadata for docker.io/library/busybox:latest 0.0s
=> [internal] load .dockerignore                               0.0s
=> => transferring context: 2B                                   0.0s

=> => transferring context: 2B                                   0.0s
=> CACHED [1/1] FROM docker.io/library/busybox:latest@sha256:a5d0ce49aa801d475da48f8cb163c354ab95ca 2.8s
=> => resolve docker.io/library/busybox:latest@sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3 2.7s
=> [auth] library/busybox:pull token for registry-1.docker.io 0.0s
=> exporting to image                                           0.1s
=> => exporting layers                                           0.0s
=> => exporting manifest sha256:c37439b3f369d2121a3ac37a9c74e986a7a5c6f19a4f1fe4fdf36b94a4ca0adc 0.0s
=> => exporting config sha256:9945ae4b9c0d064b64793d2c348a3a90dea4f03811c6b0adfdc918cc2f40b9a8 0.0s
=> => exporting attestation manifest sha256:7c4bb8a6bd4bf153f8222d58924d3475ef51e65fbd23ad26dd852a4 0.0s
=> => exporting manifest list sha256:b92c9dc26531dfff9d6c94f06317043ac85874d5c4469653c4e0ba4461fb0c 0.0s

=> => exporting attestation manifest sha256:7c4bb8a6bd4bf153f8222d58924d3475ef51e65fbd23ad26dd852a4 0.0s
=> => exporting manifest list sha256:b92c9dc26531dfff9d6c94f06317043ac85874d5c4469653c4e0ba4461fb0c 0.0s
=> => naming to docker.io/phattharaphon01/lab8:latest          0.0s
=> => unpacking to docker.io/phattharaphon01/lab8:latest       0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/ufaw6yy2joaa1mfxrocavndk8
```

## Lab Worksheet

View build details: [docker-desktop://dashboard/build/desktop-linux/desktop-linux/ufaw6yy2joaa1mfxrocavndk8](#)

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 C:\SoftwareEngineering\Lab8\_3>

nals (line 3)

PS C:\SoftwareEngineering\Lab8\_3> `docker run` phattharaphon01/lab8

"Phattharaphon Phoma"

PS C:\SoftwareEngineering\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` 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:\SoftwareEngineering\Lab8\_3> `docker push` phattharaphon01/lab8

Using default tag: latest

The push refers to repository [docker.io/phattharaphon01/lab8]

9c0abc9c5bd3: Mounted from library/busybox

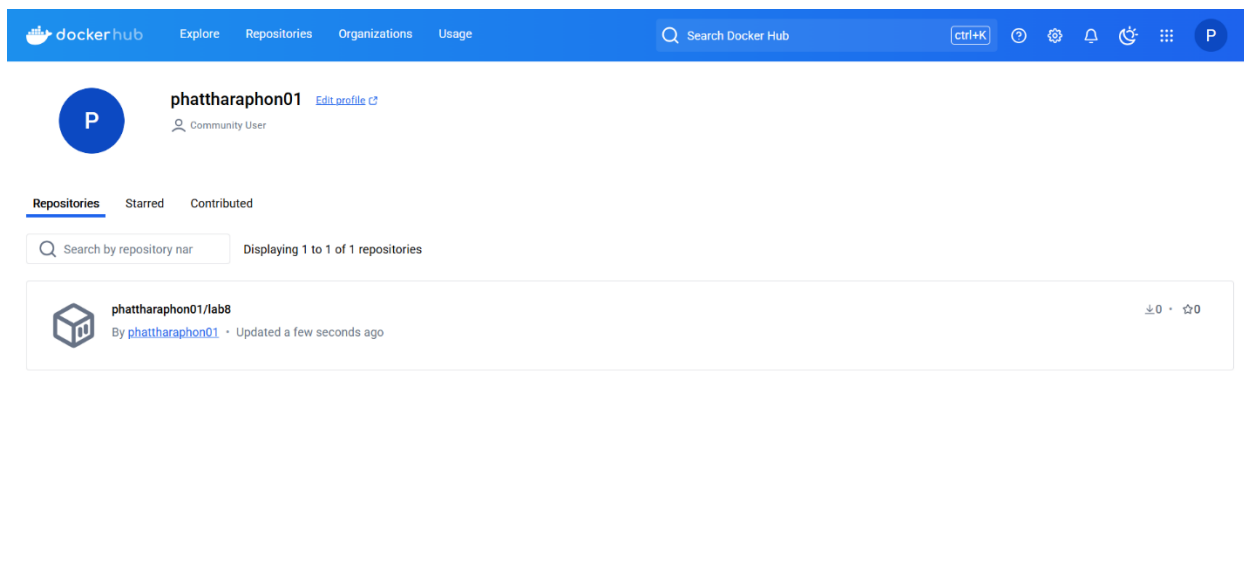
37e5d0191398: Pushed

latest: digest: sha256:b92c9dc26531dfff9d6c94f06317043ac85874d5c4469653c4e0ba4461fb0c8f size: 855

PS C:\SoftwareEngineering\Lab8\_3>



## Lab Worksheet



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

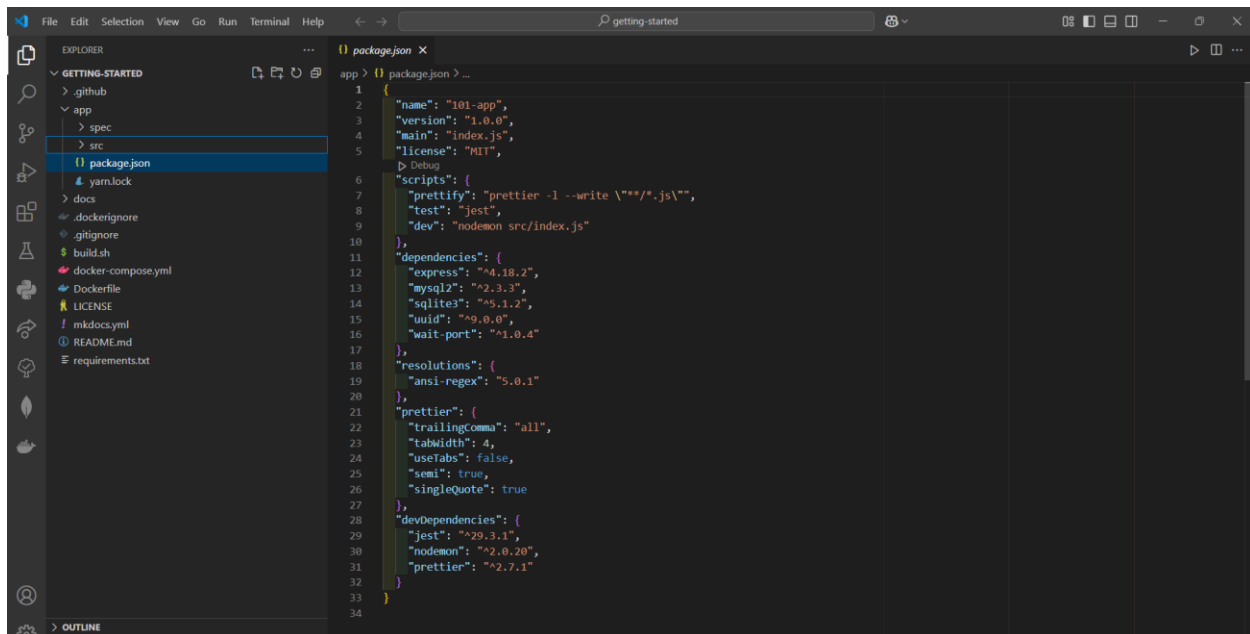
1. เปิด Command line หรือ Terminal จากนั้นสร้าง Directory ชื่อ Lab8\_4
2. ทำการ Clone ซอร์สโค้ดของเว็บแอปพลิเคชันจาก GitHub repository  
<https://github.com/docker/getting-started.git> ลงใน Directory ที่สร้างขึ้น โดยใช้คำสั่ง  
`$ git clone https://github.com/docker/getting-started.git`
3. เปิดดูองค์ประกอบภายใน 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:\SoftwareEngineering\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 | 7.14 MiB/s, done.
Resolving deltas: 100% (523/523), done.
PS C:\SoftwareEngineering\Lab8_4>
```



## Lab Worksheet



4. ภายใต้ getting-started/app ให้สร้าง Dockerfile พร้อมกับใส่เนื้อหาดังต่อไปนี้ลงไปไฟล์

FROM node:18-alpine

WORKDIR /app

COPY . .

RUN yarn install --production

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

EXPOSE 3000

5. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้ โดยกำหนดใช้ชื่อ image เป็น

myapp\_รหัสนศ. ไม่มีขีด

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

[Check point#8] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง)

แสดงคำสั่งและผลลัพธ์ที่ได้ทางหน้าจอ

## Lab Worksheet

```

PS C:\SoftwareEngineering\Lab8_4\getting-started\app> docker build -t myapp_6533801456 .
[+] Building 34.1s (10/10) FINISHED                                docker:desktop-linux
=> [internal] load build definition from Dockerfile                0.1s
=> => transferring dockerfile: 150B                               0.0s
=> [internal] load metadata for docker.io/library/node:18-alpine  3.5s
=> [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@sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d975e90 6.8s
=> => resolve docker.io/library/node:18-alpine@sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d975e90 0.0s

=> => sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15bf426d01bc1 1.26MB / 1.26MB 0.8s
=> => sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e0da02525e63167c 40.01MB / 40.01MB 5.3s
=> => sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c24403d592134f0 3.64MB / 3.64MB 1.5s
=> => extracting sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c24403d592134f0 0.2s
=> => extracting sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e0da02525e63167c 1.3s
=> => extracting sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15bf426d01bc1 0.1s
=> => extracting sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce368990ca771 0.0s
=> [internal] load build context                                   0.2s
=> => transferring context: 4.62MB                                  0.2s
=> [2/4] WORKDIR /app                                             0.4s

=> [3/4] COPY . .                                                0.1s
=> [4/4] RUN yarn install --production                          15.7s
=> exporting to image                                             7.5s
=> => exporting layers                                             4.7s
=> => exporting manifest sha256:a3c2e613e0d704549050df50d2de0b5e6fcd8f3fea5ae842f9a472f65fe3e99f 0.0s
=> => exporting config sha256:77a19ac73e720e407c45bbb0128def26a8561f9f9a2794097fd0b6aab89dcc9a 0.0s
=> => exporting attestation manifest sha256:8b8b14a4acac720af9628b6b492b23cd416da7c7b70b2a80e27e403 0.0s
=> => exporting manifest list sha256:820e536675ea35dde2e0fd7a0d439854d8014b56640a1da5cf9deb19141c3b 0.0s
=> => naming to docker.io/library/myapp_6533801456:latest        0.0s
=> => unpacking to docker.io/library/myapp_6533801456:latest     2.7s

```

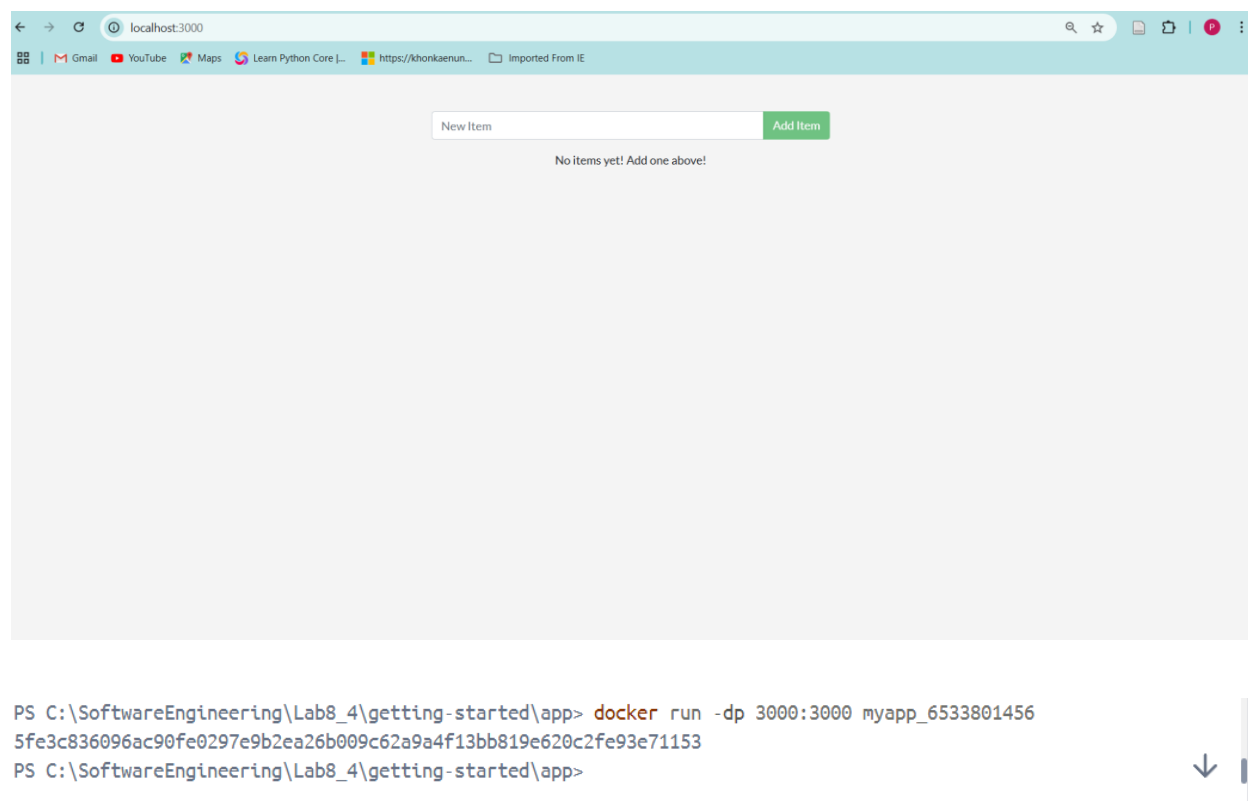
6. ทำการ Start ตัว Container ของแอปพลิเคชันที่สร้างขึ้น โดยใช้คำสั่ง

\$ 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

## 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

## Lab Worksheet

```
PS C:\SoftwareEngineering\Lab8_4\getting-started\app> docker build -t myapp_6533801456 .
[+] Building 25.6s (10/10) FINISHED                                docker:desktop-linux
=> [internal] load build definition from Dockerfile                0.0s
=> => transferring dockerfile: 150B                                0.0s
=> [internal] load metadata for docker.io/library/node:18-alpine  3.1s
=> [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@sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d975e90 0.0s
=> => resolve docker.io/library/node:18-alpine@sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d975e90 0.0s
=> [internal] load build context                                    0.0s
=> => transferring context: 8.10kB                                    0.0s
=> CACHED [2/4] WORKDIR /app                                       0.0s
=> [3/4] COPY . .                                                  0.0s
=> [4/4] RUN yarn install --production                             15.1s
```

```
=> => transferring context: 8.10kB                                0.0s
=> CACHED [2/4] WORKDIR /app                                       0.0s
=> [3/4] COPY . .                                                  0.0s
=> [4/4] RUN yarn install --production                             15.1s
=> exporting to image                                              7.1s
=> => exporting layers                                              4.3s
=> => exporting manifest sha256:81aef8b02ce2954e77f63c4406b72edba61724b47073f597de803b93da7d17b0 0.0s
=> => exporting config sha256:4acb5346aaebcc33f136a5204d7a85a147ee201a35a86200fe9c1d9d0ec6ca2b 0.0s
=> => exporting attestation manifest sha256:a24d1d32f7f192d75a568e9ae57a06afbbbf2b4033f5b425f0a5a8 0.0s
=> => exporting manifest list sha256:c1034d9521141381126cc41ed8a54f03d7c04ac441e4407813f3de95c329f1 0.0s
=> => naming to docker.io/library/myapp_6533801456:latest          0.0s
=> => unpacking to docker.io/library/myapp_6533801456:latest       2.7s
```

View build details: [docker-desktop://dashboard/build/desktop-linux/desktop-linux/sabpy7acricwtj65ey173da9l](https://docker-desktop://dashboard/build/desktop-linux/desktop-linux/sabpy7acricwtj65ey173da9l)

```
PS C:\SoftwareEngineering\Lab8_4\getting-started\app> docker run -dp 3000:3000 myapp_6533801456
ef022952c856e47efbddd5c2bb87e55cf71ad79100eea8182e575c20b75f5e5b
docker: Error response from daemon: driver failed programming external connectivity on endpoint dreamy_wing
(f35ffb8d00f97594ff0dd130613b21e0b51a2c9988a008502a8a8ced8b212e3d): Bind for 0.0.0.0:3000 failed: port is
already allocated.
PS C:\SoftwareEngineering\Lab8_4\getting-started\app>
```

(1) Error ที่เกิดขึ้นหมายความว่าอย่างไร และเกิดขึ้นเพราะอะไร

ใครเวอร์ล้มเหลวในการเขียนโปรแกรมการเชื่อมต่อภายนอกบนจุดสิ้นสุด ... การผูกสำหรับ  
0.0.0.0:3000 ล้มเหลว: พอร์ตได้รับการจัดสรรแล้ว เกิดขึ้นเพราะ พอร์ต 3000 ถูกใช้งานโดย  
Container หรือ Process อื่นบนเครื่อง

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

## Lab Worksheet

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

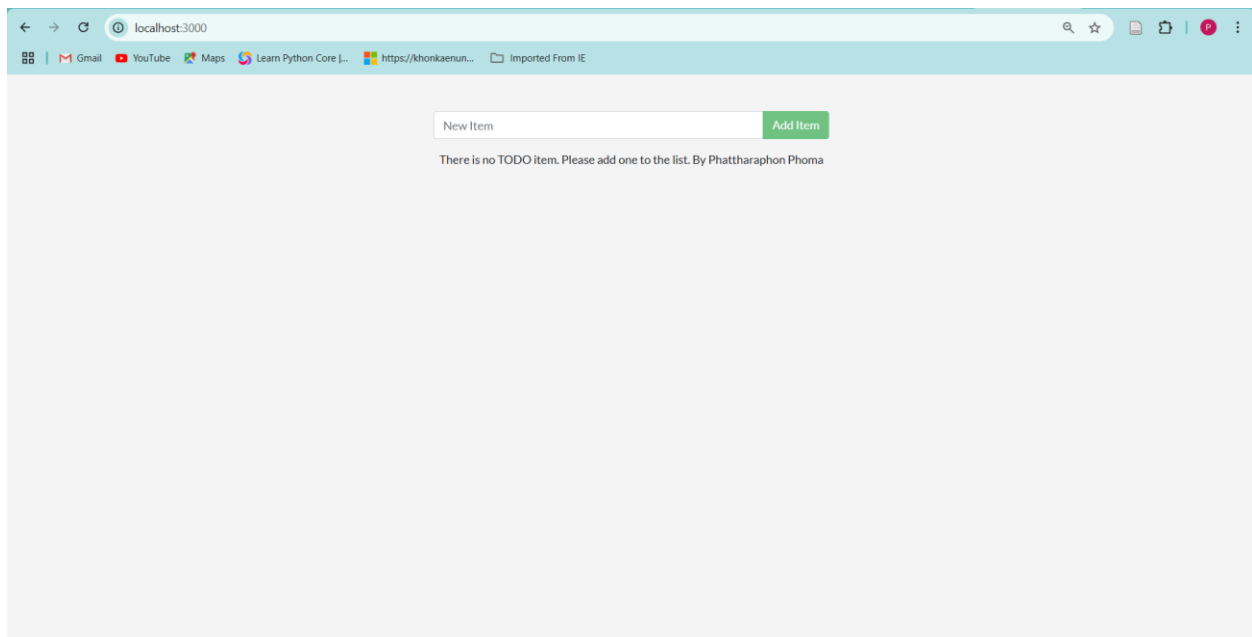
ii. เลือกไอคอนถังขยะในแถวของ Container ที่ต้องการจะลบ

iii. ยืนยันโดยการกด Delete forever

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



## Lab Worksheet

Containers [Give feedback](#)View all your running containers and applications. [Learn more](#)

<input type="checkbox"/>	Name	Container ID	Image	Port(s)	CPU (%)	Last sta	Actions
<input type="checkbox"/>	○ lucid_kepler	2e70c383eef7	myimage		0%	1 hour a	▶ ⋮ 🗑
<input type="checkbox"/>	○ friendly_carson	730f3763c404	phattharapl		0%	1 hour a	▶ ⋮ 🗑
<input type="checkbox"/>	○ elegant_volhard	2f2b26b118ef	phattharapl		0%	47 minut	▶ ⋮ 🗑
<input type="checkbox"/>	● dreamy_wing	ef022952c856	myapp_65	3000:3000 ↗	0%	1 minute	▶ ⋮ 🗑

## 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**

\*\*\*\*\*

```
3e615de9bc3d4531913c4ce8284d3979
```

```
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

## Lab Worksheet

6. สร้าง Admin User โดยใช้ username เป็นชื่อจริงของนักศึกษาพร้อมรหัสสี่ตัวท้าย เช่น somsri\_3062

[Check point#13] Capture หน้าจอที่แสดงผลการตั้งค่า

## Getting Started

## Create First Admin User

Username

Phattharaphon1456

Password

.....

Confirm password

.....

Full name

Phattharaphon Phoma

E-mail address

phattharaphon.phom@kkumail.com

Jenkins 2.479.3

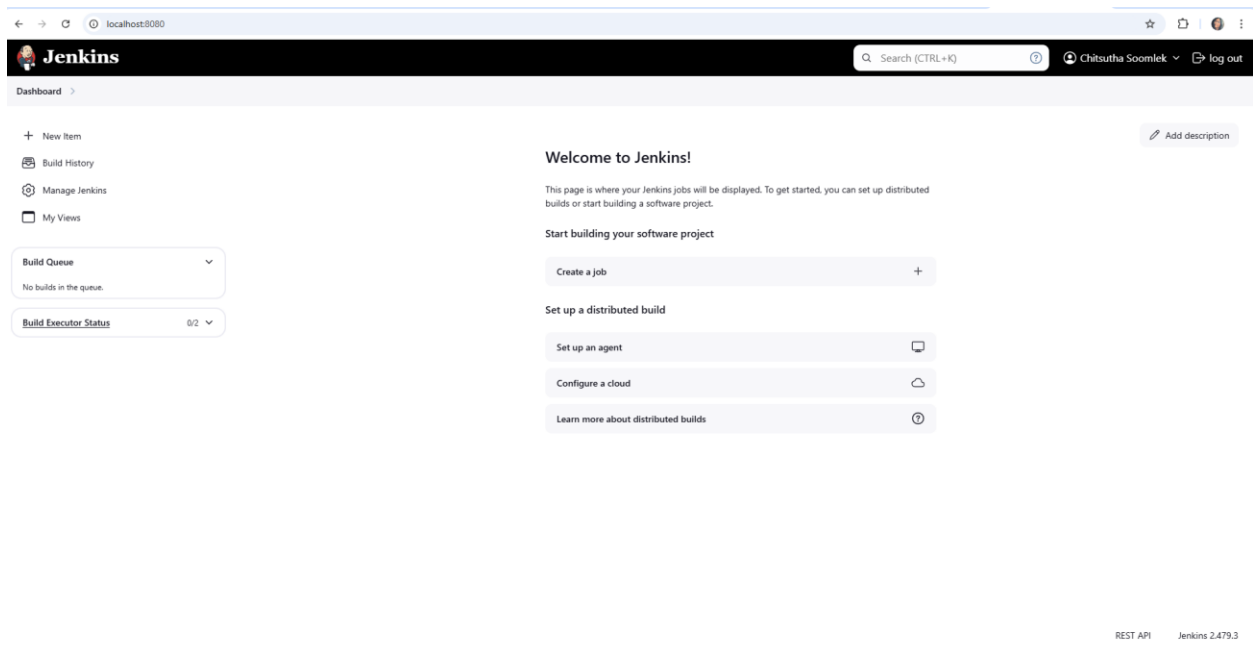
[Skip and continue as admin](#)[Save and Continue](#)

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

8. เมื่อติดตั้งเรียบร้อยแล้วจะพบหน้าจอ Dashboard ดังแสดงในภาพ

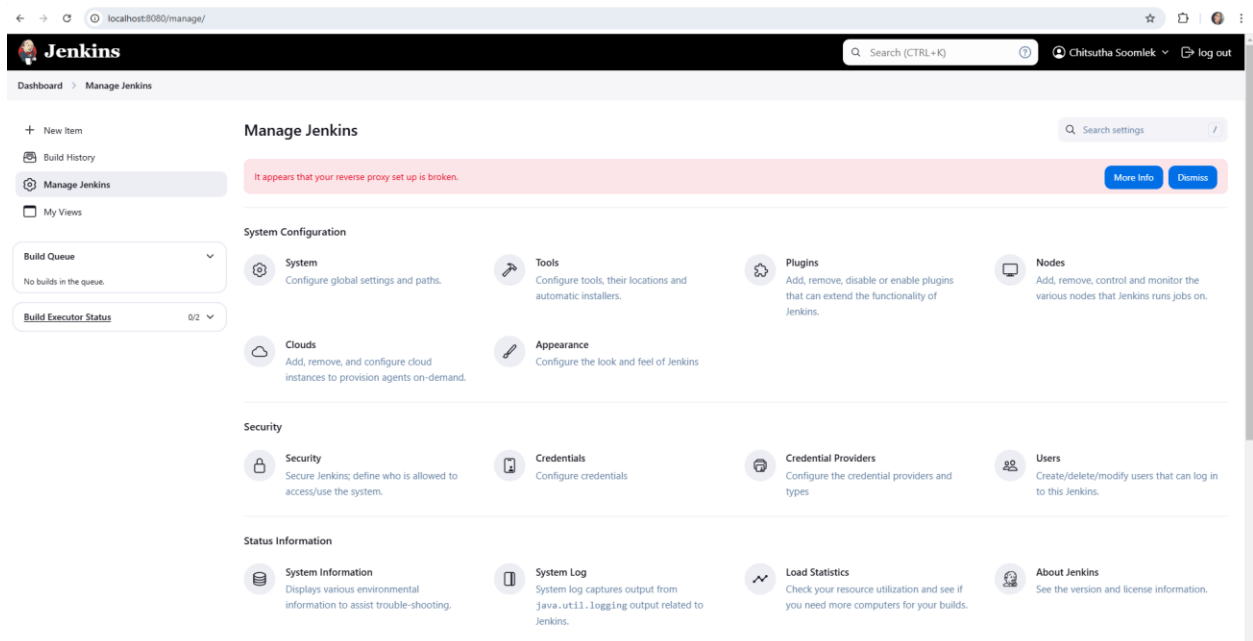


## Lab Worksheet



The screenshot shows the Jenkins Dashboard at localhost:8080. The top navigation bar includes the Jenkins logo, a search bar, and user information (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 displays a 'Welcome to Jenkins!' message and instructions on how to start building a software project. It includes buttons for 'Create a job', 'Set up a distributed build', 'Set up an agent', 'Configure a cloud', and 'Learn more about distributed builds'. The bottom right corner shows 'REST API' and 'Jenkins 2.479.3'.

## 9. เลือก Manage Jenkins แล้วไปที่เมนู Plugins



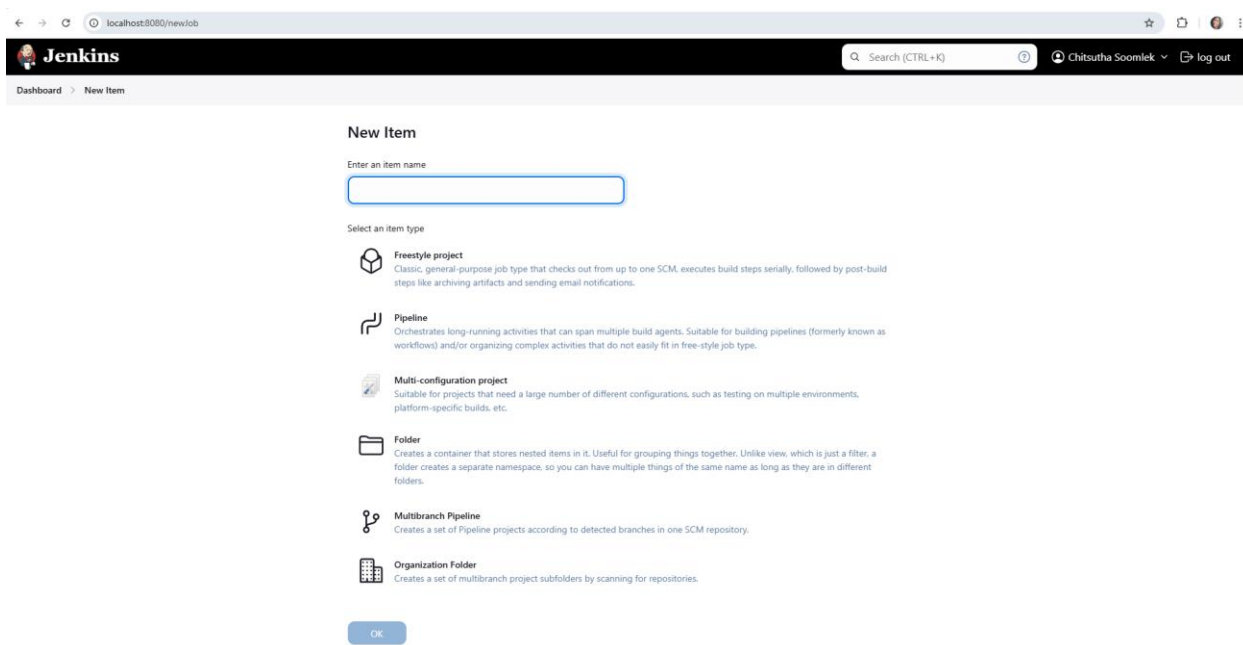
The screenshot shows the 'Manage Jenkins' page at localhost:8080/manage/. The top navigation bar is the same as the dashboard. The left sidebar shows 'Manage Jenkins' selected. The main content area has a 'Manage Jenkins' title and a search settings bar. A red banner at the top indicates 'It appears that your reverse proxy set up is broken.' Below this, the page is organized into sections: 'System Configuration' (System, Tools, Plugins, Nodes, Clouds, Appearance), 'Security' (Security, Credentials, Credential Providers, Users), and 'Status Information' (System Information, System Log, Load Statistics, About Jenkins). Each section contains a brief description of the configuration area.

## Lab Worksheet

10. ไปที่เมนู Available plugins แล้วเลือกติดตั้ง Robotframework เพิ่มเติม



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



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

Description: Lab 8.5

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

Build Trigger: เลือกแบบ Build periodically แล้วกำหนดให้ build ทุก 15 นาที

## Lab Worksheet

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

**[Check point#14] Capture** หน้าจอแสดงการตั้งค่า พร้อมกับตอบคำถามต่อไปนี้

Description

Lab 8.5

Plain text [Preview](#)

☐ Discard old builds ?

☒ GitHub project

Project url ?

Advanced ▾

☐ This project is parameterized ?

☒ Git ?

Repositories ?

Repository URL ?

Credentials ?  

- none - ▾

+ Add

Advanced ▾

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?

## Lab Worksheet

**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 Tuesday, January 28, 2025 at 12:50:37 PM Coordinated Universal Time; would next run at Tuesday, January 28, 2025 at 1:05:37 PM Coordinated Universal Time.

☐ GitHub hook trigger for GITScm polling ?

☐ Poll SCM ?

---

**Build Steps**

Execute shell ?

Command

See the list of available environment variables

robot invalid\_login.robot

Advanced ▾

Add build step ▾

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

robot invalid\_login.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

**UAT**

[Edit description](#)

Lab 8.5

**Latest Robot Results:**

	Total	Failed	Passed	Skipped	Pass %
All tests	1	0	1	0	100.0

- [Browse results](#)
- [Open report.html](#)
- [Open log.html](#)

**Permalinks**

- [Last build \(#7\), 46 sec ago](#)
- [Last failed build \(#7\), 46 sec ago](#)
- [Last unsuccessful build \(#7\), 46 sec ago](#)
- [Last completed build \(#7\), 46 sec ago](#)

**Robot Framework Tests Trend (all tests)**

Number of test cases

Build

Zoom to changes Show only failed Show only critical all Max builds [Show bigger image](#)

**Console Output**

[Download](#)
[Copy](#)
[View as plain text](#)

```

Started by user Phattharaphon Phoma
Running as SYSTEM
Building in workspace /var/jenkins_home/workspace/UAT
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /var/jenkins_home/workspace/UAT/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/Phattharaphon01/SoftwareLab8.git/ # timeout=10
Fetching upstream changes from https://github.com/Phattharaphon01/SoftwareLab8.git/
> git --version # timeout=10
> git --version # 'git version 2.39.5'
> git fetch --tags --force --progress -- https://github.com/Phattharaphon01/SoftwareLab8.git/ +refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision a7360652169f9ac699cc7c4b5e8bc856296988cc (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f a7360652169f9ac699cc7c4b5e8bc856296988cc # timeout=10
Commit message: "first commit"
> git rev-list --no-walk a7360652169f9ac699cc7c4b5e8bc856296988cc # timeout=10
[UAT] $ /bin/sh -xe /tmp/jenkins9372379520666699686.sh
+ robot invalid_login.robot
/tmp/jenkins9372379520666699686.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!

```

## Lab Worksheet

```
> git checkout -f a7360652169f9ac699cc7c4b5e8bc856296988cc # timeout=10
Commit message: "first commit"
> git rev-list --no-walk a7360652169f9ac699cc7c4b5e8bc856296988cc # timeout=10
[UAT] $ /bin/sh -xe /tmp/jenkins9372379520666699686.sh
+ robot invalid_login.robot
/tmp/jenkins9372379520666699686.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:
Done!
-Copying log files to build dir:
Done!
-Assigning results to build:
Done!
-Checking thresholds:
Done!
Done publishing Robot results.
Finished: FAILURE
```