

## Lab Worksheet

ชื่อ-นามสกุล **เจษฎากร ดุจฉาย** รหัสนักศึกษา 653380192-7 Section 3

## 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

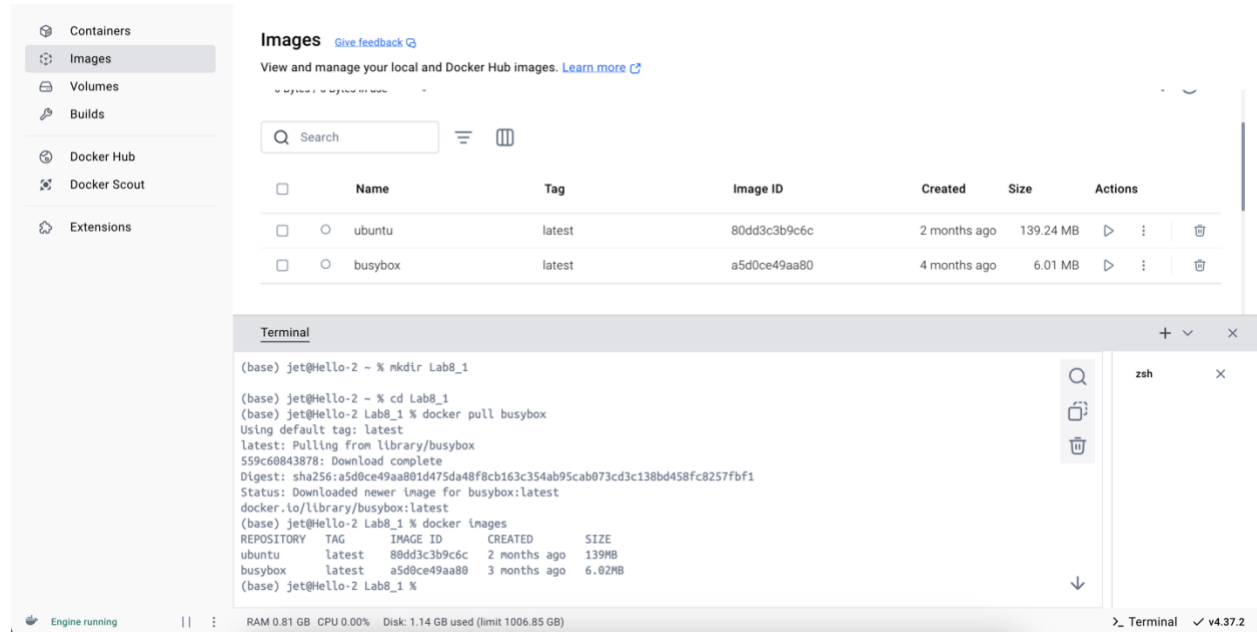
1. ติดตั้ง Docker desktop ลงบนเครื่องคอมพิวเตอร์ โดยดาวน์โหลดจาก <https://www.docker.com/get-started>
2. สร้าง Account บน Docker hub (<https://hub.docker.com/signup>)
3. กำหนดให้ \$ หมายถึง Command prompt และ <> หมายถึง ให้ป้อนค่าของพารามิเตอร์ที่กำหนด

## 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 ubuntu และ busybox ใช้บ่งบอกถึงสิ่งที่โหลดมา

(2) Tag ที่ใช้บ่งบอกถึงอะไร

ตอบ บ่งบอกเวอร์ชันของ Image

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 Jetsadakorn Dutphayap from busybox"
11. ป้อนคำสั่ง \$ docker ps -a

[Check point#2] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงผลลัพธ์ที่ได้ตั้งแต่นั้นตอนที 6-12 พร้อมกับตอบคำถามต่อไปนี้

## Lab Worksheet

The screenshot displays the Docker Desktop interface. On the left is a sidebar with navigation options: Containers, Images (selected), Volumes, Builds, Docker Hub, Docker Scout, and Extensions. The main area is titled 'Images' and includes a search bar and a table of local images. Below the table is a terminal window showing the following commands and output:

```
(base) jet@Hello-2 Lab8_1 % docker run busybox echo "Hello Jetsadakorn Dutphayap from busybox"
Hello Jetsadakorn Dutphayap from busybox
(base) jet@Hello-2 Lab8_1 % docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
dc165772df7b	busybox	"echo 'Hello Jetsada..'"	20 seconds ago	Exited (0) 19 seconds ago		vigorous_hod
gkin	busybox	"echo 'Hello ชีเมะ..'"	About a minute ago	Exited (0) About a minute ago		competent_no
e86f63682207	busybox	"sh"	3 minutes ago	Exited (0) 2 minutes ago		elegant_banz
31b89a2bd8ed	busybox	"sh"	3 minutes ago	Exited (0) 3 minutes ago		great_hofsta

At the bottom of the terminal, the output of 'docker ps -a' is shown, listing all containers with their IDs, images, commands, creation times, statuses, ports, and names.

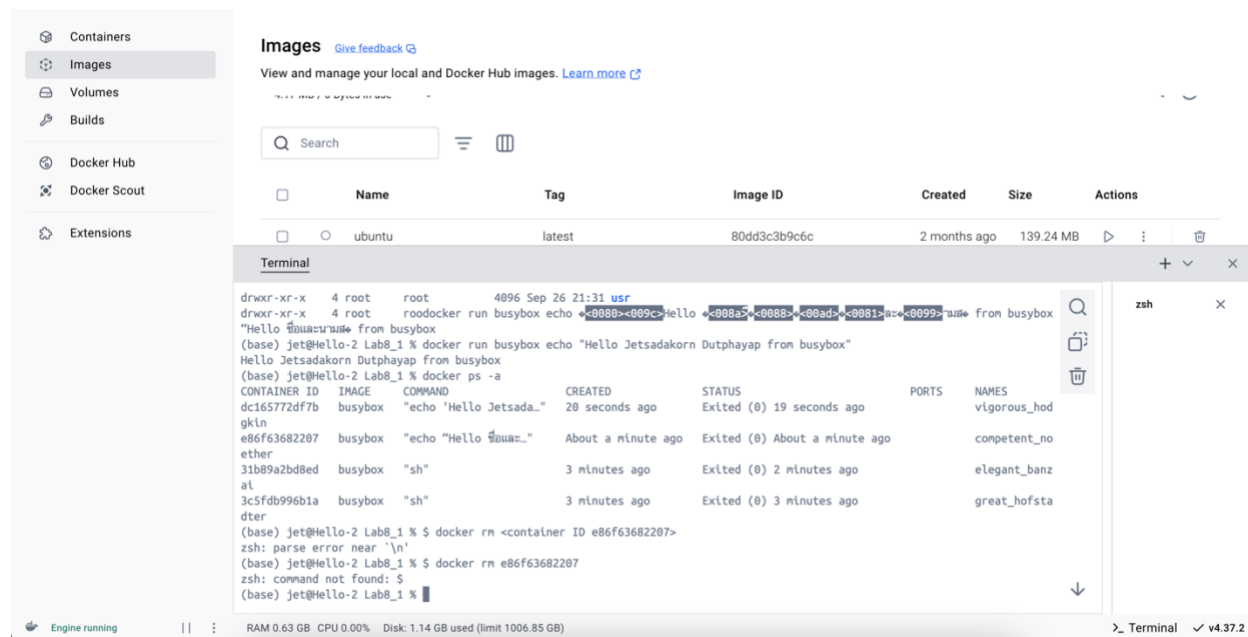
- (1) เมื่อใช้ option -it ในคำสั่ง run ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป
- ตอบ** ทำให้สามารถเข้าไปใน container และใช้งาน shell เพื่อ interact กับ container ได้ เปิดโหมด interactive ด้วย -i และเชื่อมต่อกับ terminal ด้วย -t
- (2) คอลัมน์ STATUS จากการรันคำสั่ง docker ps -a แสดงถึงข้อมูลอะไร

## Lab Worksheet

**ตอบ** สถานะของ Docker container ในปัจจุบัน โดยจะบอกว่า container นั้นๆ กำลังทำงานอยู่หรือถูกหยุด หรือมีสถานะอื่นๆ

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

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



## 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

## Lab Worksheet

```
$ 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

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

The screenshot shows the Docker Desktop interface. On the left, the 'Images' tab is selected. The main area displays a list of images, with 'my-docker-image:latest' being the only one. Below the list, a terminal window is open, showing the output of the 'docker build' command. The terminal output includes the following lines:

```
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 189B
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals
=> [internal] load metadata for docker.io/library/busybox:latest
=> [internal] load .dockerignore
=> => transferring context: 2B
=> CACHED [1/1] FROM docker.io/library/busybox:latest@sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138bd458fc8257fbf
=> => resolve docker.io/library/busybox:latest@sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138bd458fc8257fbf
=> => exporting to image
=> => exporting layers
=> => exporting manifest sha256:a8c1cc34ad8fa027fce0213537c29f855d87ac216d7fba7f3339f7dd1bf63808
=> => exporting config sha256:35cb61d4d9b62b2bbb04a66a17e6943600e6ba57aaf0e885c3a01222c4fcc6
=> => exporting attestation manifest sha256:7074eeefc7900a8e109969b998e6d8a53cf9ee3d1e508f23358c83bd4b3e1c62
=> => exporting manifest list sha256:e31d73e6912d7cdef6292f7fa9f098cdad7305bb94f23cccc078f980d52750e4
=> => naming to docker.io/library/my-docker-image:latest
=> => unpacking to docker.io/library/my-docker-image:latest
```

View build details: [docker-desktop://dashboard/build/desktop-linux/desktop-linux/p2vwln8q3hznigw4f5ofrc](#)

1 warning found (use docker --debug to expand):

```
- JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 2)
(base) jet@Hello-2 Lab8_2 % docker run my-docker-image
```

Hi there. This is my first docker image.

```
Jetsadakorn Dutphayap 653380192-7 Pooh
(base) jet@Hello-2 Lab8_2 %
```

(1) คำสั่งที่ใช้ในการ run คือ

## Lab Worksheet

**ตอบ** Docker run my-docker-image

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

**ตอบ** -t ใช้สำหรับตั้งชื่อ (tag) ให้กับ Docker image ที่สร้างขึ้นจาก Dockerfile ซึ่งช่วยให้จัดการและระบุ Image ได้ง่ายขึ้นในภายหลัง

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

1. เปิดใช้งาน Docker desktop และ Login ด้วย Username และ Password ที่ลงทะเบียนกับ Docker Hub เอาไว้
2. เปิด Command line หรือ Terminal จากนั้นสร้าง Directory ชื่อ Lab8\_3
3. ย้ายตำแหน่งปัจจุบันไปที่ Lab8\_3 เพื่อใช้เป็น Working directory
4. สร้าง Dockerfile.swp ไว้ใน Working directory

สำหรับเครื่องที่ใช้ระบบปฏิบัติการวินโดวส์ บันทึกคำสั่งต่อไปนี้ลงในไฟล์ โดยใช้ Text Editor ที่มี

```
FROM busybox
```

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

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

สำหรับเครื่องที่ใช้ระบบปฏิบัติการ MacOS หรือ Linux บนหน้าต่าง Terminal และป้อนคำสั่งต่อไปนี้

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

```
FROM busybox
```

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

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

```
EOF
```

หรือใช้คำสั่ง

```
$ touch Dockerfile
```

แล้วใช้ Text Editor ในการใส่เนื้อหาแทน

7. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้

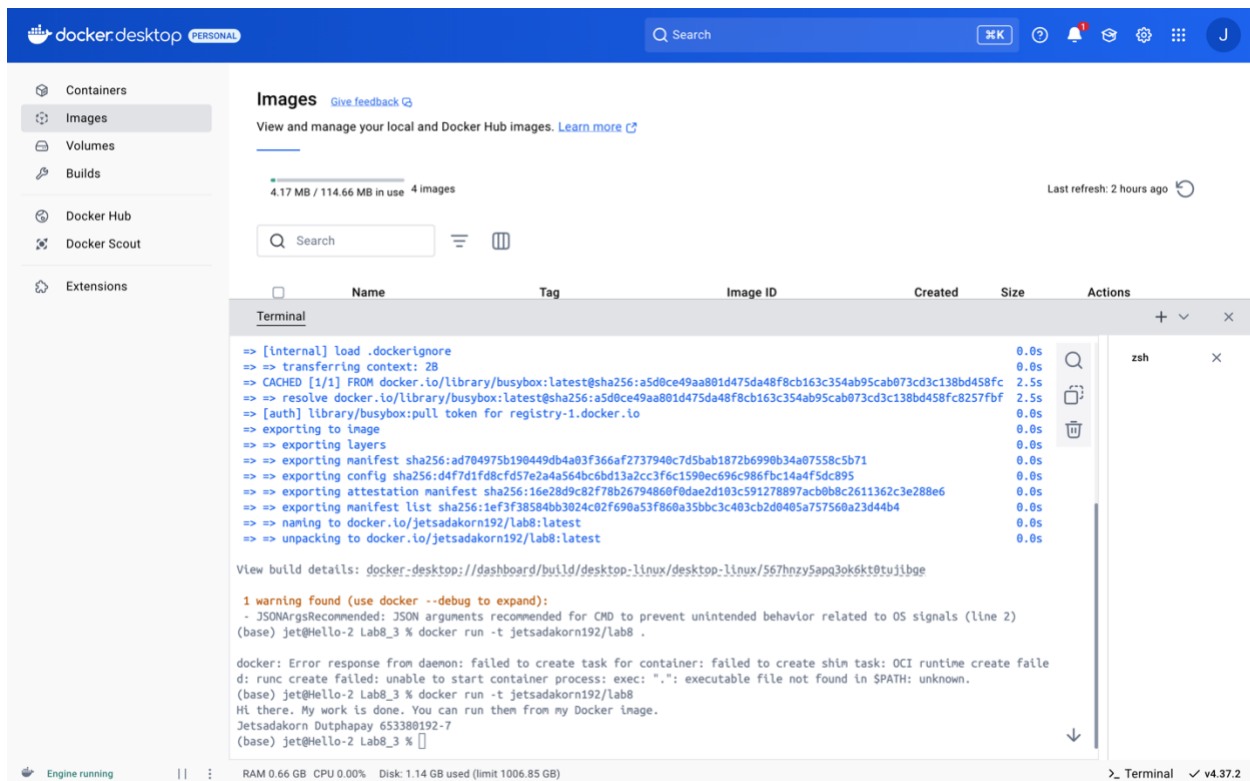
## Lab Worksheet

\$ docker build -t <username ที่ลงทะเบียนกับ Docker Hub>/lab8

5. ทำการรัน Docker image บน Container ในเครื่องของตัวเองเพื่อทดสอบผลลัพธ์ ด้วยคำสั่ง

\$ docker run <username ที่ลงทะเบียนกับ Docker Hub>/lab8

[Check point#5] Capture หน้าจอ (ทั้งหน้าตาและทุกหน้าต่างที่เกี่ยวข้อง) แสดงผลลัพธ์ที่ได้ในขั้นตอนที่ 5



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 (ทั้งหน้าตาและทุกหน้าต่างที่เกี่ยวข้อง) แสดง Repository ที่มี Docker image (<username>/lab8)

## Lab Worksheet

The screenshot displays the Docker Desktop application interface. The top navigation bar includes the Docker logo, a search bar, and various system icons. The left sidebar shows navigation options: Containers, Images (selected), Volumes, Builds, Docker Hub, Docker Scout, and Extensions.

The main area is titled "Images" and shows a progress bar indicating "4.17 MB / 114.66 MB in use" and "4 Images". Below this is a search bar and a table with columns: Name, Tag, Image ID, Created, Size, and Actions. The table is currently empty.

A terminal window is open, showing the following commands and output:

```

=> => exporting layers
=> => exporting manifest sha256:ad704975b190449db4a03f366af2737940c7d5bab1872b6990b34a07558c5b71 0.0s
=> => exporting config sha256:d4f7d1fd8cfd57e2a4a564bc6bd13a2cc3f6c1590ec696c986fbc14a4f5dc895 0.0s
=> => exporting attestation manifest sha256:16e28d9c82f778b26794860f0dae2d103c591278897acb0b8c2611362c3e288e6 0.0s
=> => exporting manifest list sha256:1ef3f38584bb3024c02f690a53f860a35bbc3c403cb2d0405a757560a23d44b4 0.0s
=> => naming to docker.io/jetsadakorn192/lab8:latest 0.0s
=> => unpacking to docker.io/jetsadakorn192/lab8:latest 0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/567hny5apq3ok6kt0tviibge

1 warning found (use docker --debug to expand):
- JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 2)
(base) jet@Hello-2 Lab8_3 % docker run -t jetsadakorn192/lab8 .

docker: Error response from daemon: failed to create task for container: failed to create shim task: OCI runtime create failed: runc create failed: unable to start container process: exec: ".": executable file not found in $PATH: unknown.
(base) jet@Hello-2 Lab8_3 % docker run -t jetsadakorn192/lab8
Hi there. My work is done. You can run them from my Docker image.
Jetsadakorn Dutphayap 653380192-7
(base) jet@Hello-2 Lab8_3 % docker push jetsadakorn192/lab8
Using default tag: latest
The push refers to repository [docker.io/jetsadakorn192/lab8]
94ff286a1149: Pushed
559c60843878: Mounted from library/busybox
latest: digest: sha256:1ef3f38584bb3024c02f690a53f860a35bbc3c403cb2d0405a757560a23d44b4 size: 855
(base) jet@Hello-2 Lab8_3 %

```

The bottom section of the interface shows the Docker Hub profile for "Jetsadakorn Dutphayap". It includes a search bar and a list of repositories. The repository "jetsadakorn192/lab8" is listed, created by "jetsadakorn192" and updated a few seconds ago.



## 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

## Lab Worksheet

The screenshot shows the Docker Desktop interface. On the left is a sidebar with navigation options: Containers, Images (selected), Volumes, Builds, Docker Hub, Docker Scout, and Extensions. The main area is titled 'Images' and shows a progress bar for '4.17 MB / 114.66 MB in use' and '4 images'. Below this is a search bar and a table with columns: Name, Tag, Image ID, Created, Size, and Actions. A terminal window is open, showing the following commands and output:

```
(base) jet@Hello-2 ~ % mkdir Lab8_4
(base) jet@Hello-2 ~ % 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 | 3.27 MiB/s, done.
Resolving deltas: 100% (523/523), done.
(base) jet@Hello-2 ~ % cd getting-started/app
(base) jet@Hello-2 app % ls
package.json  spec          src           yarn.lock
(base) jet@Hello-2 app % code package.json
zsh: command not found: code
(base) jet@Hello-2 app % nano package.json
(base) jet@Hello-2 app %
```

At the bottom, system statistics are displayed: RAM 0.00 GB, CPU 0.00%, Disk: 1.14 GB used (limit 1006.85 GB). The terminal window is titled 'Terminal' and shows 'v4.37.2'.

This screenshot shows the Docker Desktop interface with the 'Images' tab selected. The main area displays the same progress bar and search bar as the previous screenshot. A terminal window is open, showing the content of a file named 'package.json' in the 'Lab8\_4' directory. The file content is as follows:

```
{
  "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": {

```

The terminal window also shows a list of keyboard shortcuts at the bottom: Get Help, WriteOut, Read File, Prev Pg, Cut Text, Cur Pos, Exit, Justify, Where is, Next Pg, UnCut Text, and To Spell. System statistics at the bottom remain the same: RAM 0.00 GB, CPU 0.00%, Disk: 1.14 GB used (limit 1006.85 GB). The terminal window is titled 'Terminal' and shows 'v4.37.2'.

## 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

**Docker Desktop** PERSONAL

Search

Containers

**Images** Give feedback

View and manage your local and Docker Hub images. [Learn more](#)

4.17 MB / 114.66 MB in use 5 Images Last refresh: 3 hours ago

Search

Name	Tag	Image ID	Created	Size	Actions
<b>Terminal</b>					
<pre>(base) jet@hello-2 app % docker build -t myapp_6533801927:sec3 . [+] Building 35.6s (10/10) FINISHED =&gt; [Internal] load build definition from Dockerfile =&gt; =&gt; transferring dockerfile: 188B =&gt; [Internal] load metadata for docker.io/library/node:18-alpine =&gt; [auth] library/node:pull token for registry-1.docker.io =&gt; [Internal] load .dockerignore =&gt; transferring context: 2B =&gt; [1/4] FROM docker.io/library/node:18-alpine@sha256:974afb6c8314dc6502b14243b8a39fbb2d04d975e9059dd066be3e274fbb2 16.5s =&gt; resolve docker.io/library/node:18-alpine@sha256:974afb6c8314dc6502b14243b8a39fbb2d04d975e9059dd066be3e274fbb2 0.0s =&gt; sha256:fc4eb59aab89271acc5a8bd8e5e96fc17af489976964461471ea28fc8e0be459 1.26MB / 1.26MB 1.5s =&gt; sha256:e668eba0f82bcfec9fb0cd787bd7f3d013a1266567b092e90ff8b3d3be41807c 443B / 443B 1.4s =&gt; sha256:4fe16fa8f46966191d59cfcabfff137a623b3cdda747d387bd85dcfb0eff3dd 39.66MB / 39.66MB 15.8s =&gt; sha256:52f827f723504aa3325bb5a54247f0dc4b92bb72569525bc951532c4ef679bd4 4.2s =&gt; extracting sha256:52f827f723504aa3325bb5a54247f0dc4b92bb72569525bc951532c4ef679bd4 0.1s =&gt; extracting sha256:4fe16fa8f46966191d59cfcabfff137a623b3cdda747d387bd85dcfb0eff3dd 0.5s =&gt; extracting sha256:fc4eb59aab89271acc5a8bd8e5e96fc17af489976964461471ea28fc8e0be459 0.0s =&gt; extracting sha256:e668eba0f82bcfec9fb0cd787bd7f3d013a1266567b092e90ff8b3d3be41807c 0.0s =&gt; [Internal] load build context =&gt; transferring context: 4.60MB 0.1s =&gt; [2/4] WORKDIR /app 0.1s =&gt; [3/4] COPY . . 0.0s =&gt; [4/4] RUN yarn install --production 11.5s =&gt; exporting to image 3.1s =&gt; exporting layers 2.2s =&gt; exporting manifest sha256:446165186e080cb5be7bb22d406b4fecdd597c357ba5f33e3f29eb5c4265e196 0.0s =&gt; exporting manifest sha256:b508ab0d34ea6c98b82466edc3c67d07887b7b0871777feacf8decfb4b9b73fa 0.0s =&gt; exporting attestation manifest sha256:08b95f0838b1f1f329a256e352eb6129d20b08209652c5b194b8fa833b2ca61a 0.0s =&gt; exporting manifest list sha256:271b950f792f318d72ed9f6c5e2686a45902ad9091012b451acd0bccc81da3a3 0.0s =&gt; naming to docker.io/library/myapp_6533801927:sec3 0.0s =&gt; unpacking to docker.io/library/myapp_6533801927:sec3 0.9s</pre>					

Engine running RAM 1.20 GB CPU 0.00% Disk: 1.57 GB used (limit 1006.85 GB)

Terminal v4.37.2

## Lab Worksheet

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

The screenshot shows the Docker Desktop interface. On the left is a sidebar with navigation options: Containers, Images (selected), Volumes, Builds, Docker Hub, Docker Scout, and Extensions. The main area is titled 'Images' and shows a list of three images:

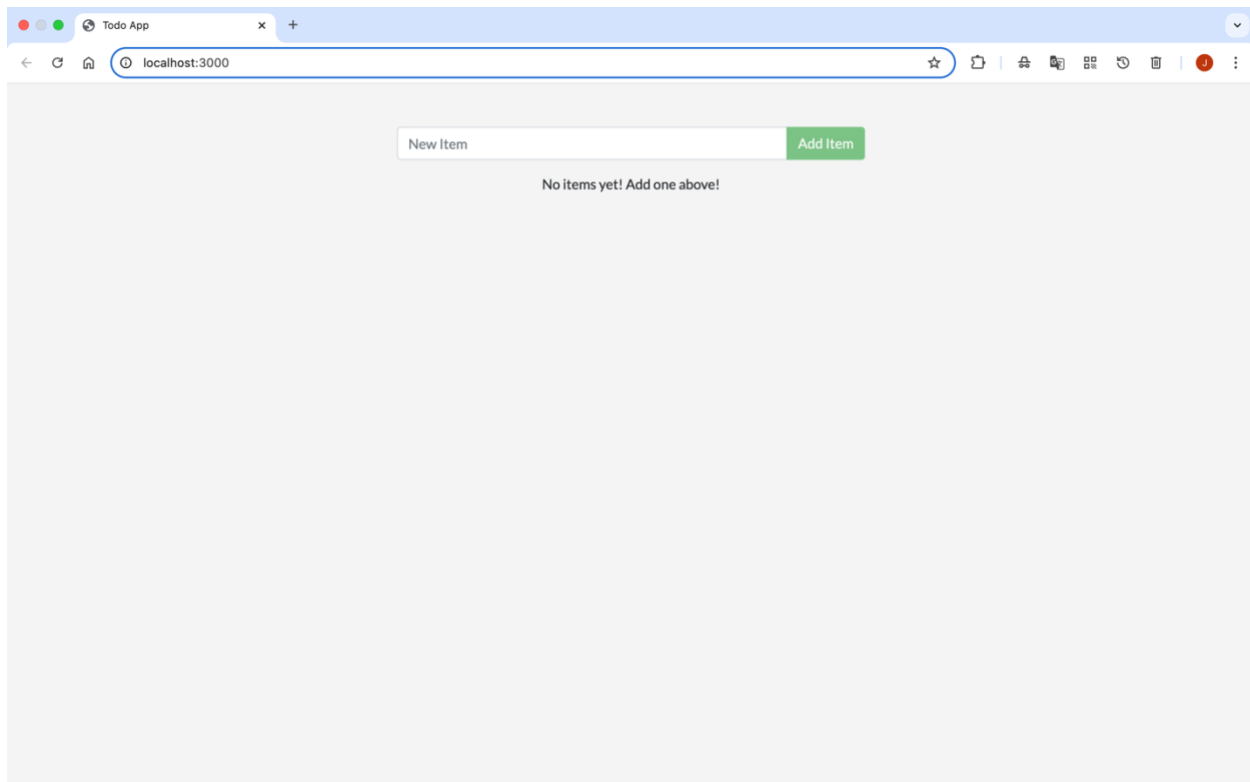
Image	Tag	Architecture	Created	Size
busybox	latest	a5d0ce49aa80	4 months ago	6.01 MB
my-docker-image	latest	e31d73e6912d	4 months ago	6.02 MB
jetsadakorn192/lab8	latest	1ef3f38584bb	4 months ago	6.02 MB

Below the images list is a terminal window showing the output of a Docker build command. The output includes steps like resolving the base image, extracting layers, and exporting the image. At the bottom, it shows the command used to run the container:

```
View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/wc4lf3k6rsox@guni9uka436
(base) jet@Hello-2 app % docker run -dp 3000:3000 myapp_6533801927:sec3
da410ca97cebe909e2c34aac21aa457e634c345348e880a95886ee79eb47ba7b
```

At the bottom of the Docker Desktop window, it shows 'Engine running' and system resources: RAM 1.22 GB, CPU 0.00%, Disk 1.58 GB used (limit 1006.85 GB).

## 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

The screenshot displays the Docker Desktop interface, which is divided into two main sections: 'Images' and 'Containers'.

**Images Section:** This section shows a list of Docker images. The table below represents the data shown in the interface:

Image	Tag	Architecture	Created	Size	Actions
busybox	latest	a5d0ce49aa80	4 months ago	6.01 MB	[Play] [More] [Delete]
my-docker-image	latest	e31d73e6912d	4 months ago	6.02 MB	[Play] [More] [Delete]
jetsadakorn192/lab8	latest	1ef3f38584bb	4 months ago	6.02 MB	[Play] [More] [Delete]

Below the images list is a terminal window showing the contents of a file named `src/static/js/app.js`. The code is a React component for a list of items, with a state `items` and a `setItems` function. It includes a form to add new items and a list of items to display.

**Containers Section:** This section shows the status of running containers. The top summary indicates:

- Container CPU usage: 0.00% / 800% (8 CPUs available)
- Container memory usage: 20.95MB / 3.74GB

Below this is a search bar and a toggle for 'Only show running containers'. The terminal window at the bottom shows the output of a `docker build` command, including the build context, cache status, and the final image name and tag.

## Lab Worksheet

(1) Error ที่เกิดขึ้นหมายความว่าอย่างไร และเกิดขึ้นเพราะอะไร

**ตอบ** Port 3000 บนเครื่อง ถูกใช้งานอยู่แล้ว และ Docker ไม่สามารถแมปพอร์ต 3000 ของ Container ไปยังเครื่องได้เนื่องจากพอร์ตนี้ถูกจับจองอยู่แล้วจากโปรแกรมอื่น ๆ หรือจาก Container อื่น

11. ลบ Container ของ Web application เวอร์ชันก่อนแก้ไขออกจากระบบ โดยใช้วิธีใดวิธีหนึ่งดังต่อไปนี้

a. ผ่าน Command line interface

- i. ใช้คำสั่ง `$ docker ps` เพื่อดู Container ID ที่ต้องการจะลบ
- ii. Copy หรือบันทึก Container ID ไว้
- iii. ใช้คำสั่ง `$ docker stop <Container ID ที่ต้องการจะลบ>` เพื่อหยุดการทำงานของ Container ดังกล่าว
- iv. ใช้คำสั่ง `$ docker rm <Container ID ที่ต้องการจะลบ>` เพื่อทำการลบ

b. ผ่าน Docker desktop

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

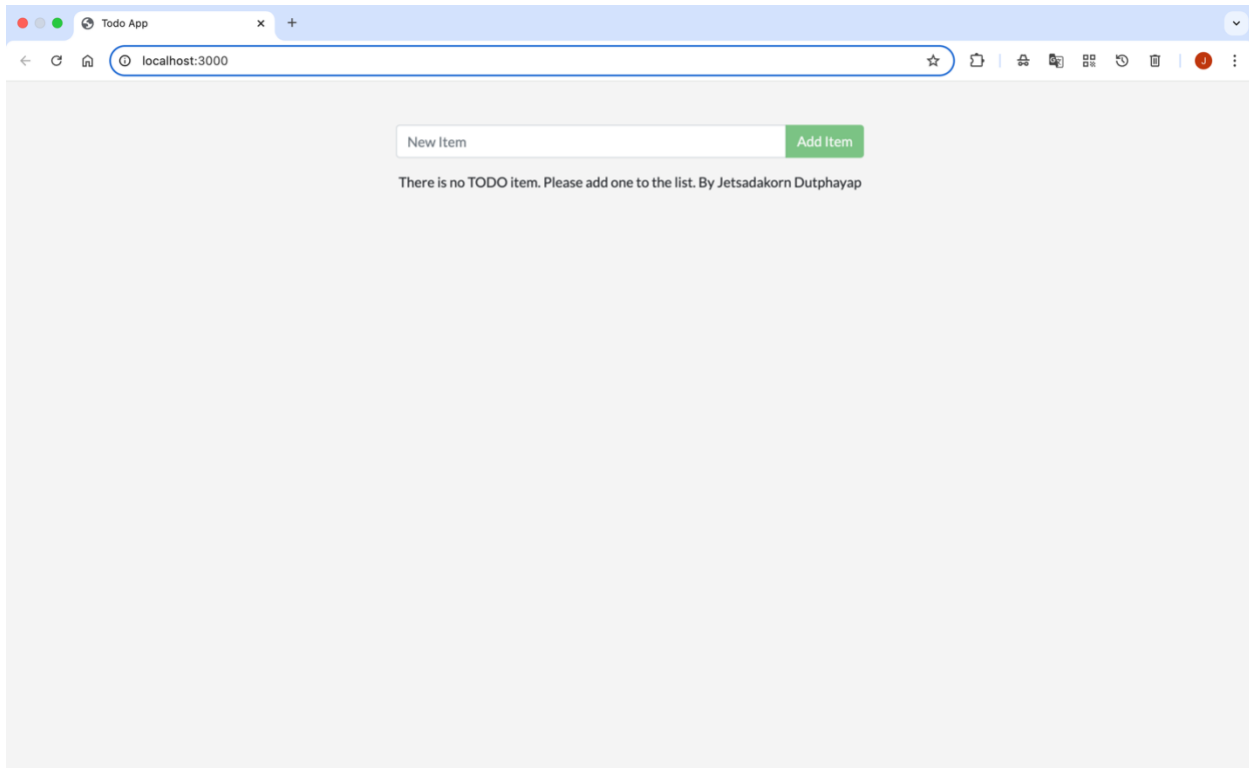
12. Start และรัน Container ตัวใหม่อีกครั้ง โดยใช้คำสั่งเดียวกันกับข้อ 6

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

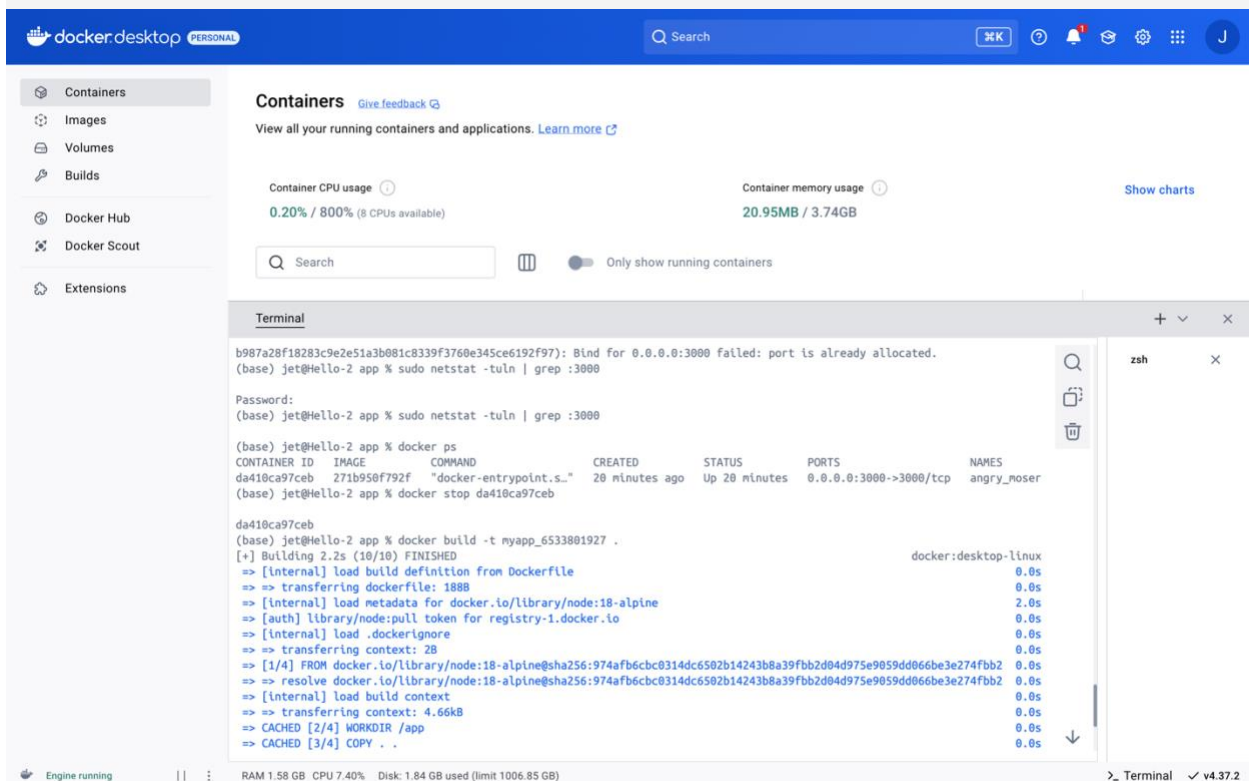
**[Check point#11]** Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงผลลัพธ์ที่ได้บน Browser และ Dashboard ของ Docker desktop



## Lab Worksheet



The screenshot shows a web browser window with the address bar set to `localhost:3000`. The page displays a simple 'Todo App' interface. At the top, there is a text input field labeled 'New Item' and a green 'Add Item' button. Below the input field, a message states: 'There is no TODO item. Please add one to the list. By Jetsadakorn Dutphayap'.

The screenshot shows the Docker Desktop interface. The 'Containers' tab is selected in the left sidebar. The main area displays 'Containers' with a search bar and a toggle for 'Only show running containers'. Below this, a table lists the running containers:

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
da410ca97ceb	271b950f792f	"docker-entrypoint.s..."	20 minutes ago	Up 20 minutes	0.0.0.0:3000->3000/tcp	angry_noser

Below the table, a terminal window is open, showing the following commands and their output:

```
(base) jet@Hello-2 app % sudo netstat -tuln | grep :3000
Password:
(base) jet@Hello-2 app % sudo netstat -tuln | grep :3000

(base) jet@Hello-2 app % docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS                               NAMES
da410ca97ceb   271b950f792f   "docker-entrypoint.s..." 20 minutes ago Up 20 minutes 0.0.0.0:3000->3000/tcp   angry_noser
(base) jet@Hello-2 app % docker stop da410ca97ceb

da410ca97ceb
(base) jet@Hello-2 app % docker build -t myapp_6533801927 .
[+] Building 2.2s (10/10) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 188B
=> [internal] load metadata for docker.io/library/node:18-alpine
=> [auth] library/node:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/4] FROM docker.io/library/node:18-alpine@sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d975e9059dd066be3e274fbb2
=> => resolve docker.io/library/node:18-alpine@sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d975e9059dd066be3e274fbb2
=> [internal] load build context
=> => transferring context: 4.66kB
=> CACHED [2/4] WORKDIR /app
=> CACHED [3/4] COPY . .
```

The bottom status bar indicates 'Engine running' and shows system resources: RAM 1.58 GB, CPU 7.40%, Disk: 1.84 GB used (limit 1006.85 GB). The terminal window title is 'Terminal' and the version is 'v4.37.2'.

## Lab Worksheet

The screenshot displays the Docker Desktop interface. The left sidebar contains navigation options: Containers, Images, Volumes, Builds, Docker Hub, Docker Scout, and Extensions. The main area is titled 'Containers' and shows 'View all your running containers and applications. Learn more'. It includes a search bar and a toggle for 'Only show running containers'. Below this, a 'Terminal' window shows the output of a Docker build command. The build process for 'myapp\_6533801927' is shown, including steps like 'Building 2.2s (10/10) FINISHED', 'load build definition from Dockerfile', 'transferring dockerfile: 188B', 'load metadata for docker.io/library/node:18-alpine', 'pull token for registry-1.docker.io', 'load .dockerignore', 'transferring context: 2B', 'FROM docker.io/library/node:18-alpine@sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d975e9059dd066be3e274fbb2', 'resolve docker.io/library/node:18-alpine@sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d975e9059dd066be3e274fbb2', 'load build context', 'transferring context: 4.66kB', 'WORKDIR /app', 'COPY . .', 'RUN yarn install --production', 'exporting to image', 'exporting layers', 'exporting manifest sha256:0f8409704d2ae601579193e8b5de913a5f84df3ce515b71b158522d3b146155e', 'exporting config sha256:0ab86580f3ed021b811bcd30d694f66b1740b4105b6b338a0a561c1e897886e7', 'exporting attestation manifest sha256:e5c4892ff7953eb1504435aed82f606342029b538c4ba44fb7f48285feb56137', 'exporting manifest list sha256:53be087554bb2101920765b68edc3061b5a9ee5e8cabe3c7ac84d1f007d1c9c8', 'naming to docker.io/library/myapp\_6533801927:latest', and 'unpacking to docker.io/library/myapp\_6533801927:latest'. The build details are also visible at the bottom of the terminal output.

Engine running | RAM 1.58 GB CPU 0.00% Disk: 1.84 GB used (limit 1006.85 GB) | Terminal v4.37.2

## Lab Worksheet

## แบบฝึกปฏิบัติที่ 8.5: เริ่มต้นสร้าง Pipeline อย่างง่ายสำหรับการ Deploy ด้วย Jenkins

1. เปิด Command line หรือ Terminal บน Docker Desktop

2. บ้อนคำสั่งและทำการรัน 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

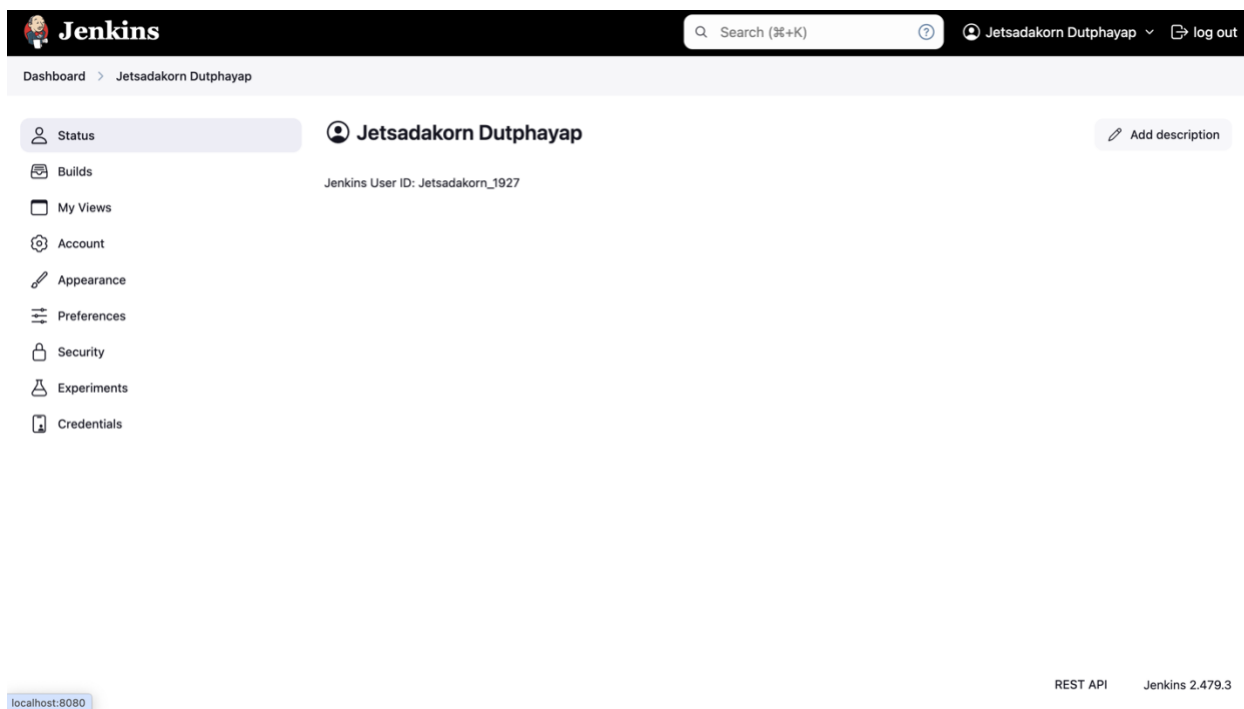
4. เมื่อได้รับการยืนยันว่า Jenkins is fully up and running ให้เปิดบราวเซอร์ และบ้อนที่อยู่เป็น

localhost:8080

5. ทำการ Unlock Jenkins ด้วยรหัสผ่านที่ได้ในข้อที่ 3

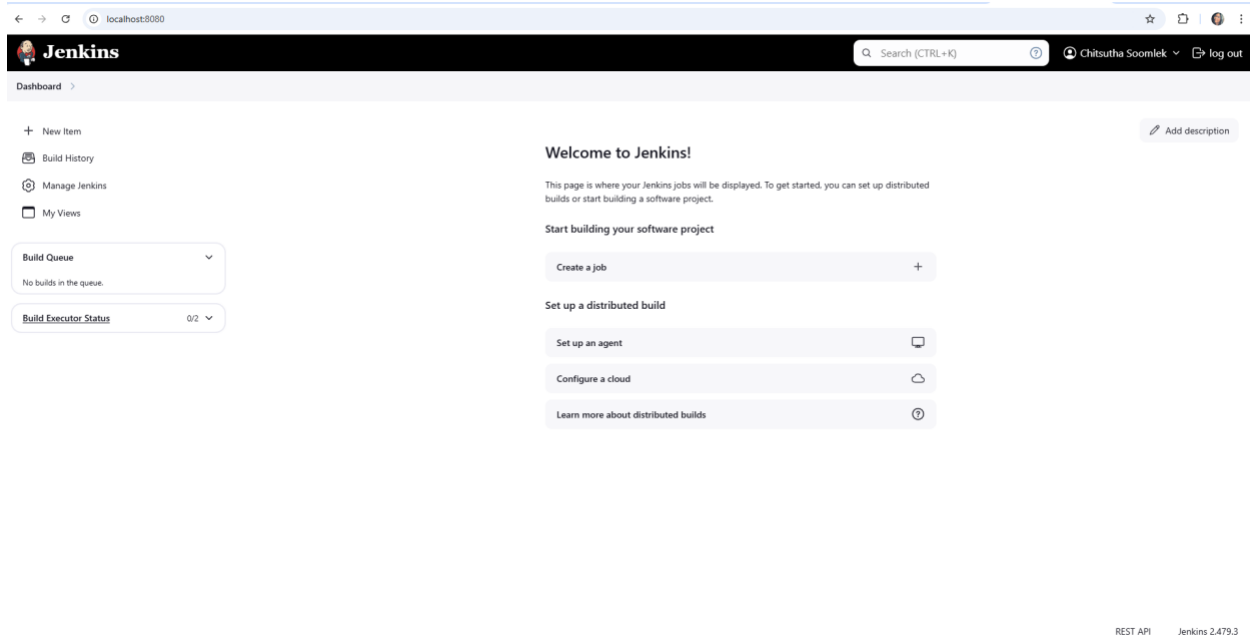
6. สร้าง Admin User โดยใช้ username เป็นชื่อจริงของนักศึกษาพร้อมรหัสสี่ตัวท้าย เช่น somsri\_3062

[Check point#13] Capture หน้าจอที่แสดงผลการตั้งค่า

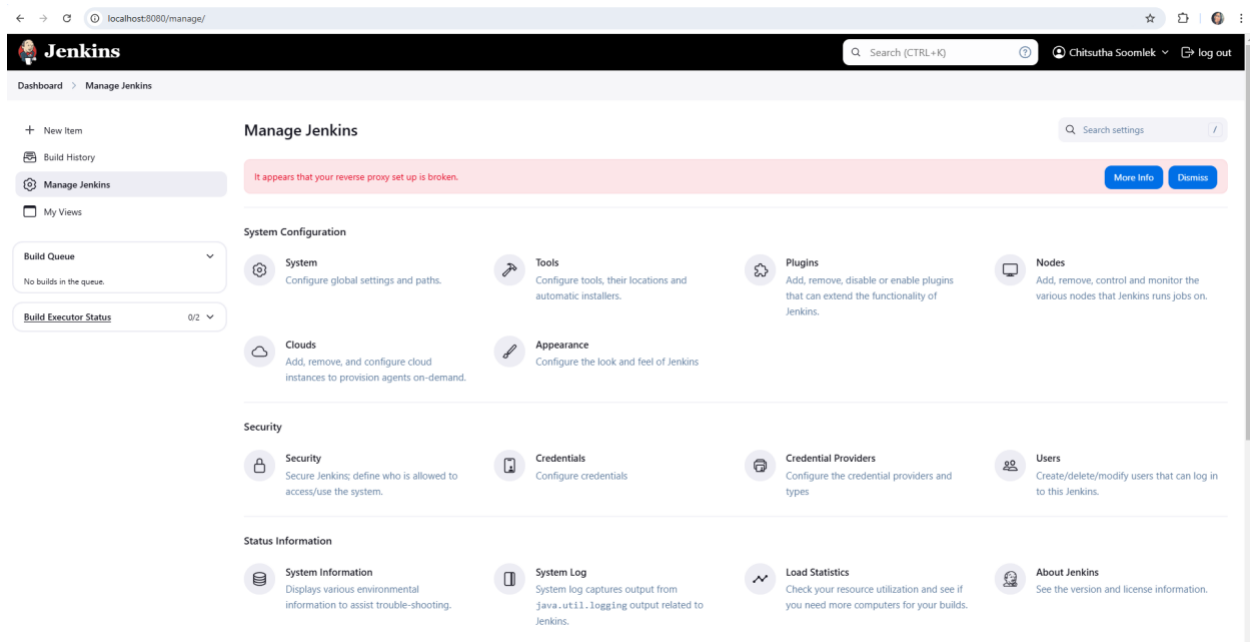


## Lab Worksheet

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



9. เลือก Manage Jenkins แล้วไปที่เมนู Plugins

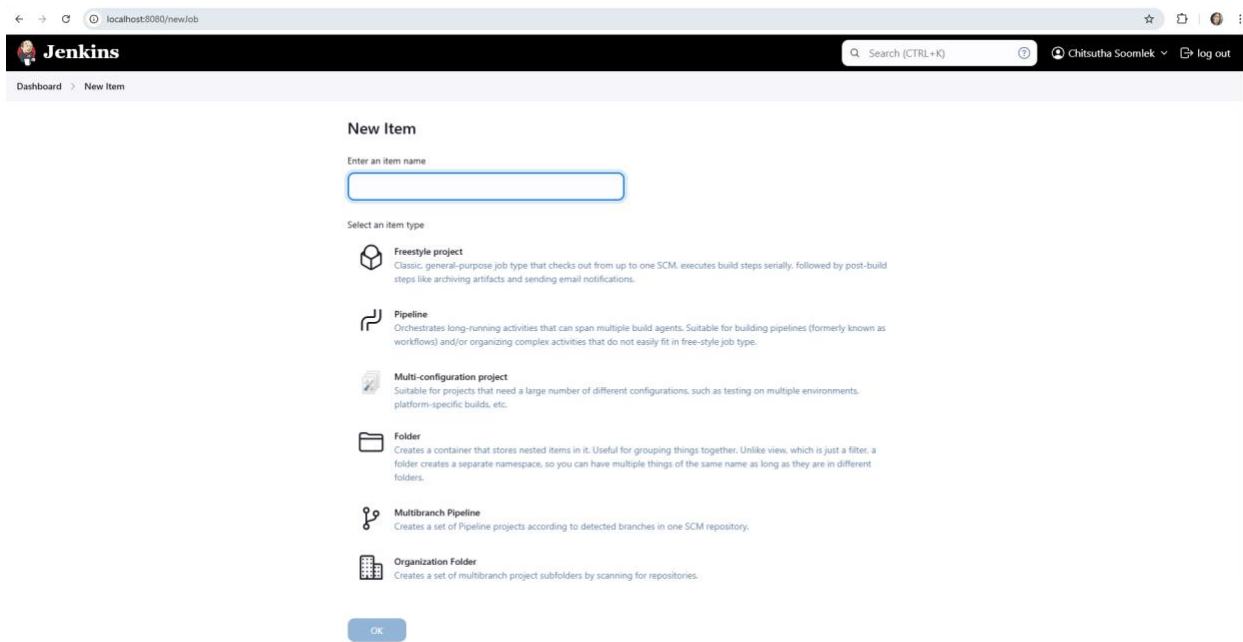


## Lab Worksheet

10. ไปที่เมนู Available plugins แล้วเลือกติดตั้ง Robotframework เพิ่มเติม



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



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

Description: Lab 8.5

## Lab Worksheet

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

Build Trigger: เลือกแบบ Build periodically แล้วกำหนดให้ build ทุก 15 นาที

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

[Check point#14] Capture หน้าจอแสดงการตั้งค่า พร้อมกับตอบคำถามต่อไปนี้

Dashboard > UAT > Configuration

### Configure

- General
- Source Code Management
- Build Triggers
- Build Environment
- Build Steps
- Post-build Actions

### General

Enabled

Description

Lab 8.5

Plain text [Preview](#)

☐ Discard old builds ?

☐ GitHub project

☐ This project is parameterized ?

☐ Throttle builds ?

☐ Execute concurrent builds if necessary ?

Advanced

---

Source Code Management

☐ None

☒ Git ?

[Save](#) [Apply](#)

## Lab Worksheet

Dashboard > UAT > Configuration

### Configure

- General
- Source Code Management**
- Build Triggers
- Build Environment
- Build Steps
- Post-build Actions

☐ None  
☒ **Git** ?

Repositories ?

Repository URL ?

Credentials ?

+ Add

Advanced ▾

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?

Add Branch

Save Apply

---

Dashboard > UAT > Configuration

### Configure

- General
- Source Code Management
- Build Triggers
- Build Environment
- Build Steps**
- Post-build Actions

#### Build Steps

Execute shell ?

Command
  
See the list of available environment variables
  

```

. /var/jenkins_home/robot_env/bin/activate
export PATH=$PATH:/usr/local/bin/chromedriver
mkdir -p results
robot --outputdir results test.robot

```

Advanced ▾

Add build step ▾

#### Post-build Actions

Add post-build action ▾

Save Apply

## Lab Worksheet

Dashboard > UAT > Configuration

### Configure

- General
- Source Code Management
- Build Triggers**
- Build Environment
- Build Steps
- Post-build Actions

☐ Trigger builds remotely (e.g., from scripts) ?
 ☐ Build after other projects are built ?
 ☒ Build periodically ?

Schedule ?

H/15 \* \* \* \*

Would last have run at Wednesday, January 29, 2025 at 3:05:44 PM Coordinated Universal Time; would next run at Wednesday, January 29, 2025 at 3:20:44 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 ?

Save Apply

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

ตอบ `./var/jenkins_home/robot_env/bin/activate`  
`export PATH=$PATH:/usr/local/bin/chromedriver`  
`mkdir -p results`  
`robot --outputdir results test.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



## Lab Worksheet

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

The screenshot shows the Jenkins web interface for Pipeline #8. The console output is visible, showing the execution of a shell script. The script sets up a virtual environment, installs dependencies, and runs a test. The test fails with a WebDriverException.

```

+ export PS1
+ [ -n ]
+ [ -z ]
+ _OLD_VIRTUAL_PS1=$
+ PS1=(robot_env) $
+ export PS1
+ VIRTUAL_ENV_PROMPT=(robot_env)
+ export VIRTUAL_ENV_PROMPT
+ [ -n -o -n ]
+ export
PATH=/var/jenkins_home/robot_env/bin:/opt/java/openjdk/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:
/usr/local/bin/chromedriver
+ mkdir -p results
+ robot --outputdir results test.robot
=====
Test
=====
Open KKU Website | FAIL |
WebDriverException: Message: Service /usr/local/bin/chromedriver unexpectedly exited. Status code was: -5

Test | FAIL |
1 test, 0 passed, 1 failed
=====
Output: /var/jenkins_home/workspace/UAT/results/output.xml
Log: /var/jenkins_home/workspace/UAT/results/log.html
Report: /var/jenkins_home/workspace/UAT/results/report.html
Build step 'Execute shell' marked build as failure
Finished: FAILURE

```

REST API Jenkins 2.479.3