

CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตสุธา สุ่มเล็ก

Lab Worksheet

ชื่อ-นามสกุล ปรานภา วิบูลย์อัฐพล รหัสนักศึกษา 653380021-4 Section 2

Lab#8 – Software Deployment Using Docker

วัตถุประสงค์การเรียนรู้

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

Pre-requisite

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

แบบฝึกปฏิบัติที่ 8.1 Hello world - รัน Container จาก Docker image

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

CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตสุธา สุ่มเล็ก

Lab Worksheet

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

```
C:\Windows\System32\cmd.exe

--tlscert string      Path to TLS certificate file (default
                    "C:\Users\computer\docker\cert.pem")
--tlskey string       Path to TLS key file (default
                    "C:\Users\computer\docker\key.pem")
--tlsverify           Use TLS and verify the remote
-v, --version         Print version information and quit

Run 'docker COMMAND --help' for more information on a command.

For more help on how to use Docker, head to https://docs.docker.com/go/guides/

D:\653380021-4\lab8_1>docker pull busybox
Using default tag: latest
latest: Pulling from library/busybox
9c0abc9c5bd3: Pull complete
Digest: sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138bd458fc8257fbf1
Status: Downloaded newer image for busybox:latest
docker.io/library/busybox:latest

What's next:
  View a summary of image vulnerabilities and recommendations → docker scout quickview busybox

D:\653380021-4\lab8_1>docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
busybox latest af4709625109 3 months ago 4.27MB
```

- (1) สิ่งที่อยู่ภายใต้คอลัมน์ Repository คืออะไร ชื่อของภาพหรือ registry ที่เก็บ image
- (2) Tag ที่ใช้บ่งบอกถึงอะไร ระบุเวอร์ชันหรือความแตกต่างของภาพใน repository
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

CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตสุธา สุ่มเล็ก

Lab Worksheet

```
C:\Windows\System32\cmd.exe
Usage:  docker run [OPTIONS] IMAGE [COMMAND] [ARG...]

Create and run a new container from an image

D:\653380021-4\lab8_1>docker run busybox

D:\653380021-4\lab8_1>docker run -it busybox sh
/ # ls
bin      dev      etc      home     lib      lib64    proc     root     sys      tmp      usr      var
/ # ls -la
total 48
drwxr-xr-x  1 root    root      4096 Jan 22 07:27 .
drwxr-xr-x  1 root    root      4096 Jan 22 07:27 ..
-rwxr-xr-x  1 root    root        0 Jan 22 07:27 .dockerenv
drwxr-xr-x  2 root    root     12288 Sep 26 21:31 bin
drwxr-xr-x  5 root    root      360 Jan 22 07:27 dev
drwxr-xr-x  1 root    root      4096 Jan 22 07:27 etc
drwxr-xr-x  2 nobody  nobody    4096 Sep 26 21:31 home
drwxr-xr-x  2 root    root      4096 Sep 26 21:31 lib
lrwxrwxrwx  1 root    root        3 Sep 26 21:31 lib64 -> lib
dr-xr-xr-x 250 root    root        0 Jan 22 07:27 proc
drwx----- 1 root    root      4096 Jan 22 07:27 root
dr-xr-xr-x 11 root    root        0 Jan 22 07:27 sys
drwxrwxrwt  2 root    root      4096 Sep 26 21:31 tmp
drwxr-xr-x  4 root    root      4096 Sep 26 21:31 usr
drwxr-xr-x  4 root    root      4096 Sep 26 21:31 var
/ # exit

D:\653380021-4\lab8_1>docker run busybox echo "Hello Prangnapha Wibunatthaphon from busybox"
Hello Prangnapha Wibunatthaphon from busybox

D:\653380021-4\lab8_1>docker ps -a
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS              PORTS          NAMES
8be1265062c5   busybox    "echo 'Hello Prangna..." 9 seconds ago   Exited (0) 9 seconds ago           condescending_meninsky
6f46f50978f8   busybox    "sh"                    About a minute ago   Exited (0) About a minute ago           zen_banach
c3336a536ce4   busybox    "sh"                    2 minutes ago     Exited (0) 2 minutes ago           priceless_bhaskara

D:\653380021-4\lab8_1>
```

(1) เมื่อใช้ option -it ในคำสั่ง run ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป แบ่งได้เป็น -i ช่วยเปิดการรับ input จากผู้ใช้.
-t จำลอง terminal ให้ใช้งาน shell ได้สมบูรณ์.
ดังนั้นใช้ -it เพื่อรัน container แบบ interactive และได้ตอบกับ terminal ภายใน container ได้โดยตรง.

(2) คอลัมน์ STATUS จากการรันคำสั่ง docker ps -a แสดงถึงข้อมูลอะไร
สถานะปัจจุบัน ของ container แต่ละตัวในระบบ โดยให้ข้อมูลเกี่ยวกับการทำงานของ container อย่างเช่น running ,exited, paused

CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตสุธา สุ่มเล็ก

Lab Worksheet

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

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

```
D:\653380021-4\lab8_1>docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS        NAMES
8be1265062c5   busybox   "echo 'Hello Prangna... " 9 seconds ago  Exited (0)   9 seconds ago  zen_banach
6f46f50978f8   busybox   "sh"                    About a minute ago  Exited (0)   About a minute ago  zen_banach
c3336a536ce4   busybox   "sh"                    2 minutes ago  Exited (0)   2 minutes ago  priceless_bhaskara

D:\653380021-4\lab8_1>docker rm 8be1265062c5
8be1265062c5

D:\653380021-4\lab8_1>docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS        NAMES
6f46f50978f8   busybox   "sh"                    6 minutes ago  Exited (0)   5 minutes ago  zen_banach
c3336a536ce4   busybox   "sh"                    6 minutes ago  Exited (0)   6 minutes ago  priceless_bhaskara

D:\653380021-4\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."
CMD echo "ชื่อ-นามสกุล รหัสนักศึกษา ชื่อเล่น"
EOF
```

หรือใช้คำสั่ง

CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตสุธา สุ่มเล็ก

Lab Worksheet

\$ 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

```
PS D:\online_class\docker\lab8_2> mv Dockerfile.swp Dockerfile
PS D:\online_class\docker\lab8_2> docker build -t dockerfile .
[+] Building 7.1s (6/6) FINISHED
=> [internal] load build definition from Dockerfile                                docker:desktop-linux 0.1s
=> => transferring dockerfile: 156B                                              0.0s
=> [internal] load metadata for docker.io/library/busybox:latest                  5.3s
=> [auth] library/busybox:pull token for registry-1.docker.io                   0.0s
=> [internal] load .dockerignore                                                 0.1s
=> => transferring context: 2B                                                  0.0s
=> [1/1] FROM docker.io/library/busybox:latest@sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138bd458fc82 0.9s
=> => resolve docker.io/library/busybox:latest@sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138bd458fc82 0.0s
=> => sha256:9c0abc9c5bd3a7854141800ba1f4a227baa88b11b49d8207eadc483c3f2496de 2.16MB / 2.16MB 0.8s
=> => exporting to image                                                         1.2s
=> => exporting layers                                                           0.0s
=> => exporting manifest sha256:e84ca0fd9a06cf8573513c992d2d8ec973838e18aeefa7db905335eb305e04af 0.0s
=> => exporting config sha256:b6bdcf3dc82229070200048998165383a5f3f3cbe14f6a835a4e814dbb678ee1 0.0s
=> => exporting attestation manifest sha256:d74e63c490c3e85bce3563d520f4d738cb9610205c3069bdd16780f59d3a0cb0 0.0s
=> => exporting manifest list sha256:dd36379c29fe9a0e109bd8ddcd9cf3ec0865b659f9929951d39834a6528623aa 0.0s
=> => naming to docker.io/library/dockerfile:latest                           0.0s
=> => unpacking to docker.io/library/dockerfile:latest                         1.0s

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:\online_class\docker\lab8_2> Docker image
```

Images [Give feedback](#)

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

Local Hub repositories

4.39 MB / 0 Bytes in use 1 Images

Last refresh: 23 minutes ago

Search

<input type="checkbox"/>	Name	Tag	Image ID	Created	Size	Actions
<input type="checkbox"/>	dockerfile	latest	dd36379c29fe	4 months ago	6.55 MB	

CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตสุธา สุ่มเล็ก

Lab Worksheet

```
Run 'docker image COMMAND --help' for more information on a command.
PS D:\online_class\docker\lab8_2> Docker images
REPOSITORY    TAG        IMAGE ID      CREATED       SIZE
dockerfile    latest     dd36379c29fe  4 months ago  6.56MB
PS D:\online_class\docker\lab8_2> docker run dockerfile
Prangnapha Wibunatthaphon 653380021-4 Fah
PS D:\online_class\docker\lab8_2> |
```

(1) คำสั่งที่ใช้ในการ run คือ docker run dockerfile

(2) Option -t ในคำสั่ง \$ docker build ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป
-t มีไว้เพื่อกำหนด **ชื่อและ tag** ให้กับ **image** ที่สร้าง ซึ่งช่วยให้จัดการ image ได้ง่ายและสะดวกต่อการใช้งาน
หรือ push ไปยัง Docker registry

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
```

CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตสุธา สุ่มเล็ก

Lab Worksheet

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:\online_class\docker\lab8_3> ls

Directory: D:\online_class\docker\lab8_3

Mode                LastWriteTime         Length Name
----                -
-a-----         1/27/2025   8:58 PM             140 Dockerfile

PS D:\online_class\docker\lab8_3> docker build -t prangnapha/lab8_3 .
[+] Building 4.6s (6/6) FINISHED                                docker:desktop-linux
=> [internal] load build definition from Dockerfile              0.0s
=> => transferring dockerfile: 179B                             0.0s
=> [internal] load metadata for docker.io/library/busybox:latest 4.3s
=> [auth] library/busybox:pull token for registry-1.docker.io   0.0s
=> [internal] load .dockerignore                                0.0s
=> => transferring context: 2B                                    0.0s
=> CACHED [1/1] FROM docker.io/library/busybox:latest@sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138bd 0.0s
=> => resolve docker.io/library/busybox:latest@sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138bd458fc82 0.0s
=> => exporting to image                                         0.1s
=> => exporting layers                                           0.0s
=> => exporting manifest sha256:71262ff0d52fbc8947b7d7dc911bf08d5cc83dd8df9beb114455f74ed9d69d4b 0.0s
=> => exporting config sha256:5134459bcb3b026edfac9d220bf1d7c4298a5f9335543ffeale1fc0a0da326 0.0s
=> => exporting attestation manifest sha256:bbb0a294916fd75025ff835e47d01e0542f306db68ec5b9babc86f2dbbe5d6eb 0.0s
=> => exporting manifest list sha256:2750d26782b9eafe0ce6d1bf84e085ae0095685af4a5405aa9529864b03ab74 0.0s
=> => naming to docker.io/prangnapha/lab8_3:latest              0.0s
=> => unpacking to docker.io/prangnapha/lab8_3:latest           0.0s
```

<input type="checkbox"/>	Name	Tag	Image ID	Created	Size	Actions
<input type="checkbox"/>	 dockerfile	latest	dd36379c29fe	4 months ago	6.55 MB	  
<input type="checkbox"/>	 prangnapha/lab8_3	latest	2750d26782b9	4 months ago	6.55 MB	  

CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตสุธา สุ่มเล็ก

Lab Worksheet

```
3 warnings found (use docker --debug to expand):
- JSONArgsRecommended: JSON arguments recommended for CMD to prevent u
- MultipleInstructionsDisallowed: Multiple CMD instructions should not
one will be used (line 2)
- JSONArgsRecommended: JSON arguments recommended for CMD to prevent u
PS D:\online_class\docker\lab8_3> docker run -t prangnapha/lab8_3
Prangnapha Wibunatthaphon 653380021-4
PS D:\online_class\docker\lab8_3> |
```

6. ทำการ Push ตัว Docker image ไปไว้บน Docker Hub โดยใช้คำสั่ง

\$ 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 D:\online_class\docker\lab8_3> docker push prangnapha/lab8_3
Using default tag: latest
The push refers to repository [docker.io/prangnapha/lab8_3]
93354d94ddc7: Pushed
9c0abc9c5bd3: Pushed
latest: digest: sha256:2750d26782b9eafef0ce6d1bf84e085ae0095685af4a5405aa9529864b03ab74 size: 855
PS D:\online_class\docker\lab8_3> |
```


CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตสุธา สุ่มเล็ก

Lab Worksheet

The screenshot shows the GitHub repository page for `prangnapha/lab8_3`. The `General` tab is selected, showing the repository name, last pushed time (2 minutes ago), and options to add a description and category (both marked as incomplete). Below this, the `Tags` section is visible, indicating 1 tag(s). A table lists the tags:

Tag	OS	Type	Pulled	Pushed
<code>latest</code>		Image	2 minutes ago	2 minutes ago

Below the table, there is a `See all` link. The bottom part of the screenshot shows the `Tags` tab selected, displaying a list of tags with details like digest, OS/ARCH, last pull time, and compressed size. A `docker pull` command is shown with a `Copy` button.

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

CP353004/SC313 004 Software Engineering (2/2567)

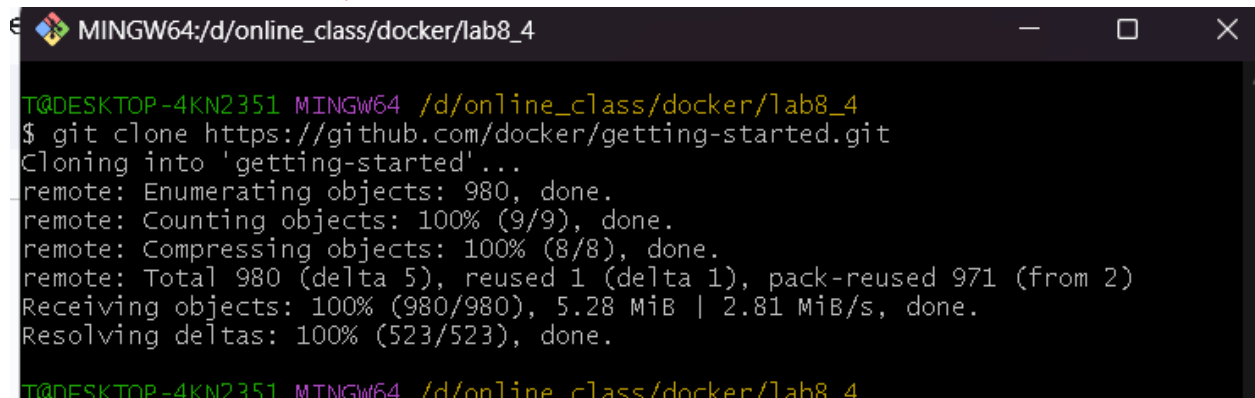
ผศ.ดร.ชิตสุธา สุ่มเล็ก

Lab Worksheet

```
$ 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

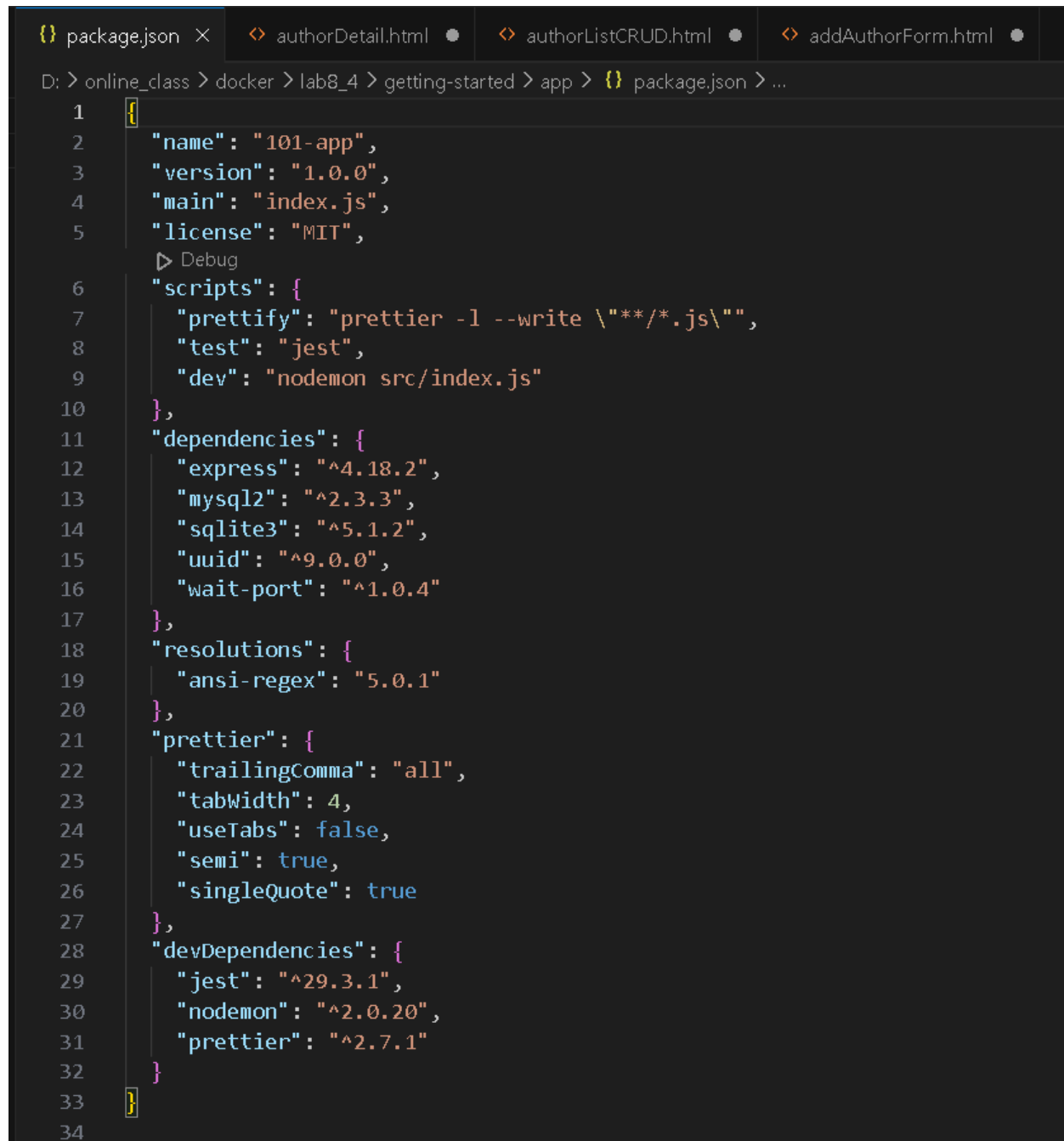


```
MINGW64:/d/online_class/docker/lab8_4
T@DESKTOP-4KN2351 MINGW64 /d/online_class/docker/lab8_4
$ git clone https://github.com/docker/getting-started.git
Cloning into 'getting-started'...
remote: Enumerating objects: 980, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 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 | 2.81 MiB/s, done.
Resolving deltas: 100% (523/523), done.
T@DESKTOP-4KN2351 MINGW64 /d/online_class/docker/lab8_4
```

CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตส์ฐา สุ่มเล็ก

Lab Worksheet



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

4. ภายใต้ getting-started/app ให้สร้าง Dockerfile พร้อมกับใส่เนื้อหาดังต่อไปนี้ลงไปไฟล์

FROM node:18-alpine

WORKDIR /app

CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตสุธา สุ่มเล็ก

Lab Worksheet

COPY . .

RUN yarn install --production

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

EXPOSE 3000

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

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

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

```
PS D:\online_class\docker\lab8_4\getting-started\app> docker build -t myapp_6533800214 .
[+] Building 37.3s (10/10) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 154B
=> [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@sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d975e9059dd066be3e27
=> => resolve docker.io/library/node:18-alpine@sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d975e9059dd066be3e27
=> => sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce368990ca771 444B / 444B
=> => sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15bf426d01bc1 1.26MB / 1.26MB
=> => sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e0da02525e63167c 40.01MB / 40.01MB
=> => sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c24403d592134f0 3.64MB / 3.64MB
=> => extracting sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c24403d592134f0
```

Q Search						
<input type="checkbox"/>	Name	Tag	Image ID	Created	Size	Actions
<input type="checkbox"/>	prangnapha/lab8_3	latest	2750d26782b9	4 months ago	6.55 MB	
<input type="checkbox"/>	dockerfile	latest	dd36379c29fe	4 months ago	6.55 MB	
<input type="checkbox"/>	myapp_6533800214	latest	5858e23a0fe0	1 minute ago	341.97 MB	

6. ทำการ Start ตัว Container ของแอปพลิเคชันที่สร้างขึ้น โดยใช้คำสั่ง

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

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

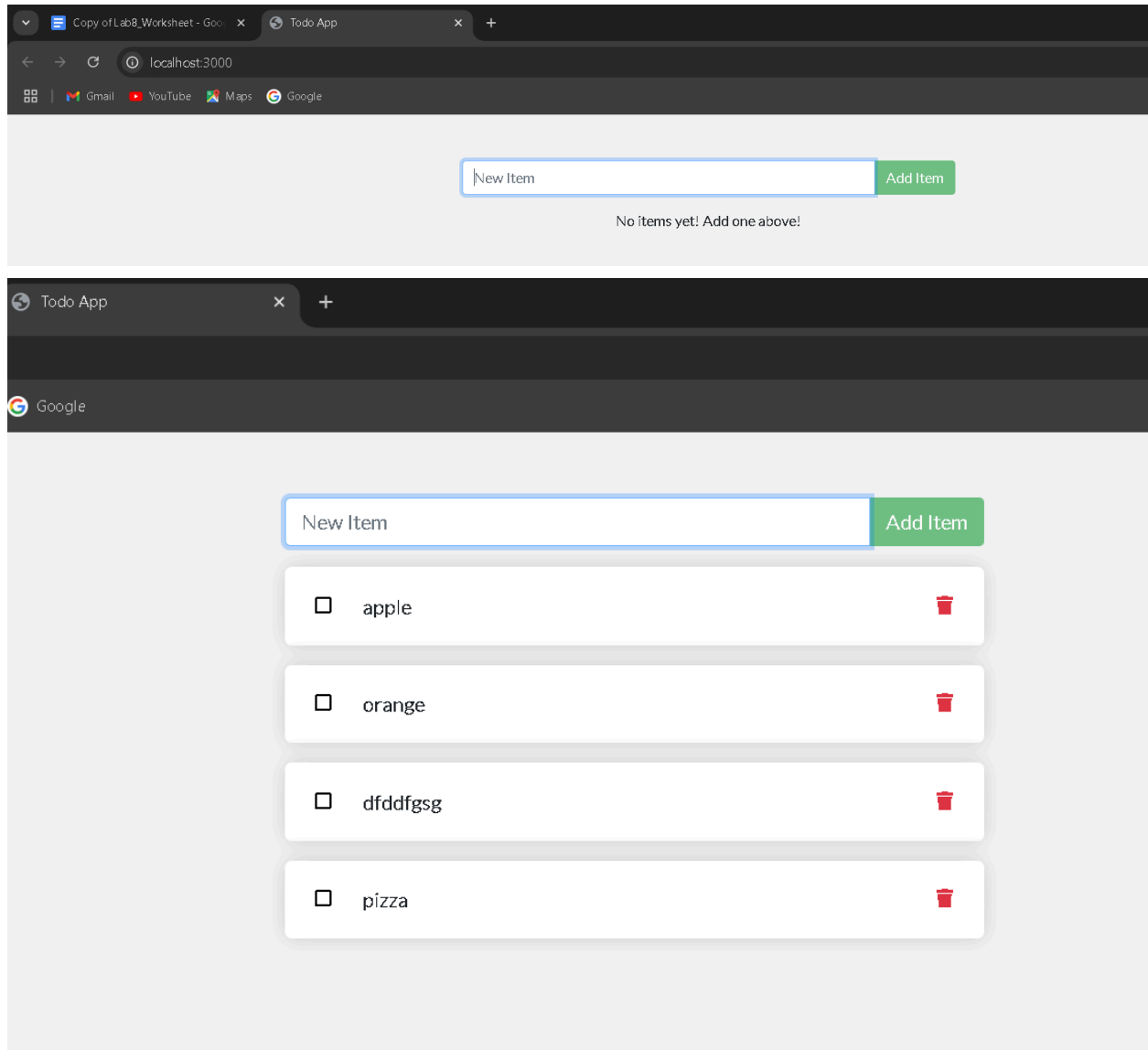
[Check point#9] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงผลลัพธ์ที่ได้บน Browser และ Dashboard ของ Docker desktop

```
D:\online_class\docker\lab8_4>docker run -dp 3000:3000 myapp_6533800214
c7cd2952d0050dc2848b7510e87ad85dfe7b2f8d69c6d553371ad6d08038ec35
```

CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตสุธา สุ่มเล็ก

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

CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตสุธา สุ่มเล็ก

Lab Worksheet

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

```
=> => unpacking to docker.io/library/myapp_6533800214:latest 2.4s
PS D:\online_class\docker\lab8_4\getting-started\app> docker build -t myapp_6533800214 .
[+] Building 1.4s (9/9) FINISHED                                docker:desktop-linux
=> [internal] load build definition from Dockerfile              0.0s
=> => transferring dockerfile: 154B                             0.0s
=> [internal] load metadata for docker.io/library/node:18-alpine 1.0s
=> [internal] load .dockerignore                                0.0s
=> => transferring context: 2B                                    0.0s
=> [1/4] FROM docker.io/library/node:18-alpine@sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d975e9059dd066be3e27 0.0s
=> => resolve docker.io/library/node:18-alpine@sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d975e9059dd066be3e27 0.0s
=> [internal] load build context                                0.0s
=> => transferring context: 2.49kB                                0.0s
=> CACHED [2/4] WORKDIR /app                                    0.0s
=> CACHED [3/4] COPY . .                                        0.0s
=> CACHED [4/4] RUN yarn install --production                  0.0s
=> exporting to image                                           0.1s
=> => exporting layers                                           0.0s
=> => exporting manifest sha256:8e4e68589f4f8af1865c1747b86cb81707cd665f273e34b17855d9ec52a82f06 0.0s
=> => exporting config sha256:147bb95ea8e2705009c777172b987c67a7d09e144f313f78ccc4750acc76dfa7 0.0s
=> => exporting attestation manifest sha256:5afa6f0469d91aaee2b89e63638fae6afd3a4f774c00fb26c5fc561d6b076f8d 0.0s
=> => exporting manifest list sha256:f7c0a54b7992f3307ee6fc91640e7ecec5d58e9ce158b07ce0ac1e52ec53d4f3 0.0s
=> => naming to docker.io/library/myapp_6533800214:latest      0.0s
=> => unpacking to docker.io/library/myapp_6533800214:latest 0.0s
PS D:\online_class\docker\lab8_4\getting-started\app> |

D:\online_class\docker\lab8_4>docker run -dp 3000:3000 myapp_6533800214
950951c90d14488fc3435b86f4c88ed610af663bd048efe2ac9ff8ad8009ba9b
docker: Error response from daemon: driver failed programming external connectivity on endpoint stupefied_bohr (ffdb756175daeb91ad2c3f0669e1b8313bdd8120b968570e8b76691feaa660b3): Bind for 0.0.0.0:3000 failed: port is already allocated.
D:\online_class\docker\lab8_4>|
```

(1) Error ที่เกิดขึ้นหมายความว่าอย่างไร และเกิดขึ้นเพราะอะไร

Docker พยายามสร้างcontainerและผูกพอร์ต 3000 บนเครื่องhost กับคอนเทนเนอร์ แต่พบว่าพอร์ตนี้ถูกใช้งานอยู่แล้วโดยโปรเซสอื่นหรือคอนเทนเนอร์อื่นในระบบของคุณ ซึ่งเกิดจาก มีโปรแกรมหรือคอนเทนเนอร์ที่ใช้งานพอร์ต 3000 บนเครื่องโฮสต์อยู่แล้ว

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

CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตสุธา สุ่มเล็ก

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

```
=> => unpacking to docker.io/library/myapp_6533800214:latest 0.0s
PS D:\online_class\docker\lab8_4\getting-started\app> docker build -t myapp_6533800214 .
[+] Building 6.1s (10/10) FINISHED docker:desktop-linux
=> [internal] load build definition from Dockerfile 0.0s
=> => transferring dockerfile: 154B 0.0s
=> [internal] load metadata for docker.io/library/node:18-alpine 2.9s
=> [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@sha256:974afb6c0314dc6502b14243b8a39fbb2d04d975e9059dd066be3e27 0.0s
=> => resolve docker.io/library/node:18-alpine@sha256:974afb6c0314dc6502b14243b8a39fbb2d04d975e9059dd066be3e27 0.0s
=> [internal] load build context 0.4s
=> => transferring context: 2.49kB 0.4s
=> CACHED [2/4] WORKDIR /app 0.0s
=> CACHED [3/4] COPY . . 0.0s
=> CACHED [4/4] RUN yarn install --production 0.0s
=> exporting to image 2.4s
=> => exporting layers 0.0s
=> => exporting manifest sha256:8e4e68589f4f8af1865c1747b86cb81707cd665f273e34b17855d9ec52a82f06 0.0s
=> => exporting config sha256:147bb95ea8e2705009c777172b987c67a7d09e144f313f78ccc4750acc76dfa7 0.0s
=> => exporting attestation manifest sha256:a7926a7bbf9954156023ed37d7141983bee4bad554b636f2e552f765ac41b428 0.0s
=> => exporting manifest list sha256:bb08c8e950ee5fd0c24e39e3b14ed618d1bc961d0ad25faa281865c6fa461e09 0.0s
=> => naming to docker.io/library/myapp_6533800214:latest 0.0s
=> => unpacking to docker.io/library/myapp_6533800214:latest 2.3s
```

```
D:\online_class\docker\lab8_4>docker run -dp 3000:3000 myapp_6533800214
950951c90d14488fc3435b86f4c88ed610af663bd048efe2ac9ff8ad8009ba9b
docker: Error response from daemon: driver failed programming external connectivity on endpoint stupefied_bohr (ffdb756175daeb91ad2c3f0669e1b8313bdd8120b968570e8b76691feaa660b3): Bind for 0.0.0.0:3000 failed: port is already allocated.

D:\online_class\docker\lab8_4>docker run -dp 3000:3000 myapp_6533800214
84ade647103b8a94d4bdad432933117b692b75cb5062eaa46490cd30232f51af

D:\online_class\docker\lab8_4>
```

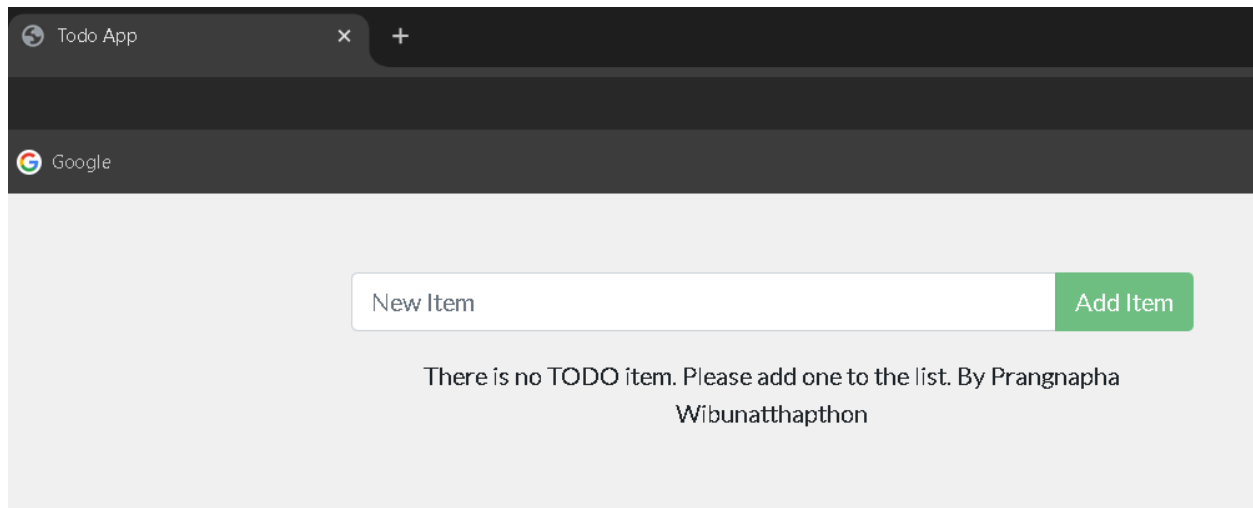
<input type="checkbox"/>	Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
<input type="checkbox"/>	<input type="radio"/> epic_heisenberg	84ab11227312	prangnapha/lab8_3		0%	2 hours ago	<input type="play"/> <input type="info"/> <input type="trash"/>
<input type="checkbox"/>	<input type="radio"/> kind_yalow	f9e1648a5b1c	dockerfile:latest		0%	2 hours ago	<input type="play"/> <input type="info"/> <input type="trash"/>
<input type="checkbox"/>	<input type="radio"/> busy_kepler	9464bd8091c4	dockerfile		0%	2 hours ago	<input type="play"/> <input type="info"/> <input type="trash"/>
<input type="checkbox"/>	<input checked="" type="radio"/> peaceful_jepsen	84ade647103b	myapp_6533800214	3000:3000 <input type="checkbox"/>	0.02%	1 minute ago	<input type="stop"/> <input type="info"/> <input type="trash"/>

CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตสุธา สุ่มเล็ก

Lab Worksheet

<input type="checkbox"/>	Name	Tag	Image ID	Created	Size	Actions
<input type="checkbox"/>	dockerfile	latest	dd36379c29fe	4 months ago	6.55 MB	
<input type="checkbox"/>	prangnapha/lab8_3	latest	2750d26782b9	4 months ago	6.55 MB	
<input type="checkbox"/>	myapp_6533800214	latest	bb08c8e950ee	14 minutes ag	341.97 MB	



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

CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตสุธา สุ่มเล็ก

Lab Worksheet

```
2025-01-28 10:32:56.312+0000 [id=45] INFO jenkins.InitReactorRunner$1#onAttained: System config adapted
2025-01-28 10:32:56.313+0000 [id=42] INFO jenkins.InitReactorRunner$1#onAttained: Loaded all jobs
2025-01-28 10:32:56.316+0000 [id=46] INFO jenkins.InitReactorRunner$1#onAttained: Configuration for all jobs updated
2025-01-28 10:32:56.337+0000 [id=60] INFO hudson.util.Retrier#start: Attempt #1 to do the action check updates server
2025-01-28 10:32:57.147+0000 [id=45] INFO jenkins.install.SetupWizard#init:

*****
*****
*****

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

3c25745171464e138f618baf4e27b543

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

*****
*****
*****

2025-01-28 10:33:05.527+0000 [id=42] INFO jenkins.InitReactorRunner$1#onAttained: Completed initialization
2025-01-28 10:33:05.566+0000 [id=25] INFO hudson.lifecycle.Lifecycle#onReady: Jenkins is fully up and running
2025-01-28 10:33:06.992+0000 [id=60] INFO h.m.DownloadService$Downloadable#load: Obtained the updated data file for hudson.tasks.Maven.MavenInstaller
2025-01-28 10:33:06.993+0000 [id=60] INFO hudson.util.Retrier#start: Performed the action check updates server successfully at the attempt #1
```

- 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

 Prangnapha Wibunatthaphon

Jenkins User ID: Prangnapha_0214

Getting Started

Instance Configuration

Jenkins URL:

<http://localhost:8080/>

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD_URL environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Jenkins 2.479.3

Not now

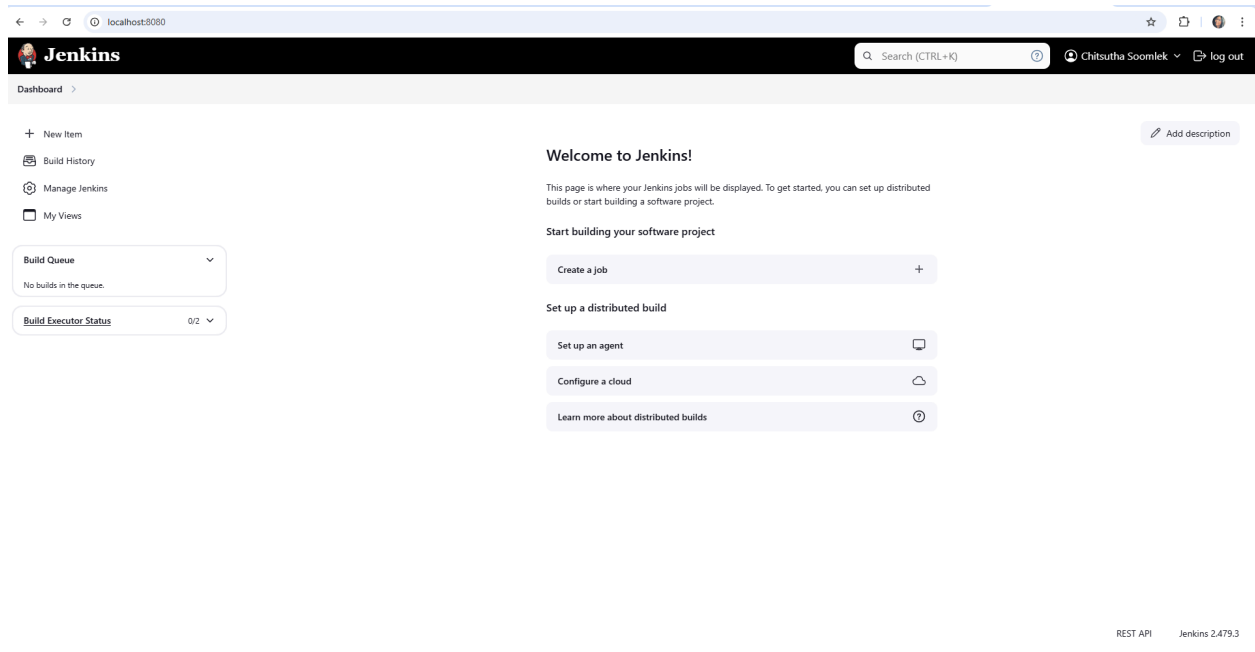
Save and Finish

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

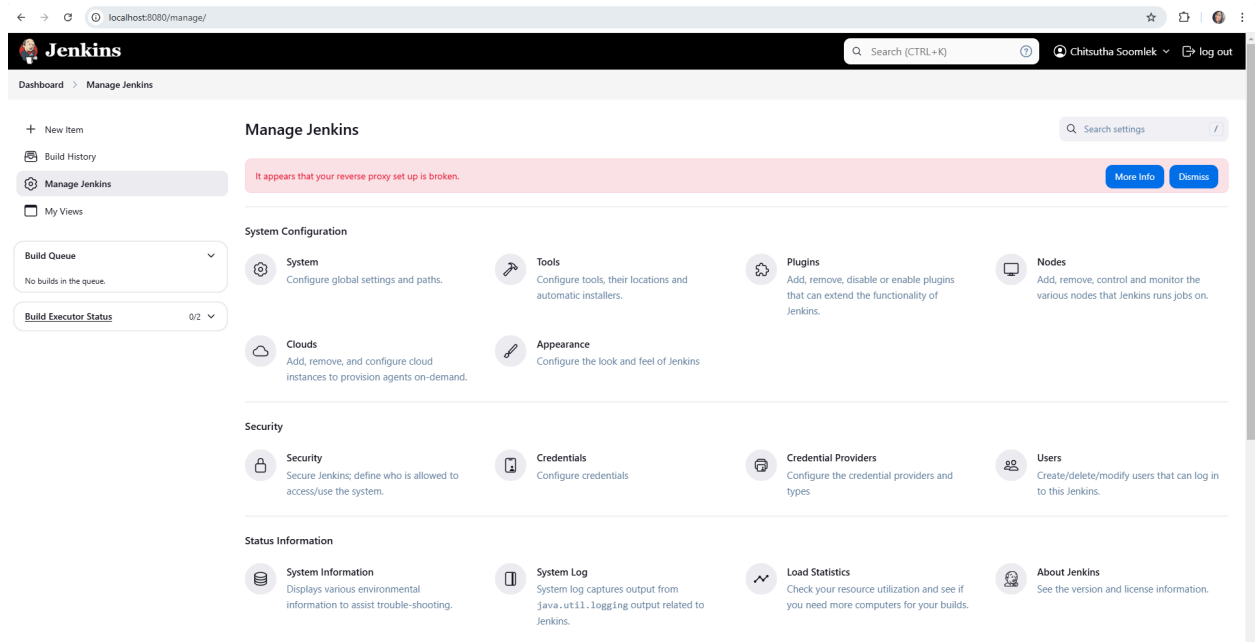
CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตสุธา สุ่มเล็ก

Lab Worksheet



9. เลือก Manage Jenkins แล้วไปที่เมนู Plugins



CP353004/SC313 004 Software Engineering (2/2567)

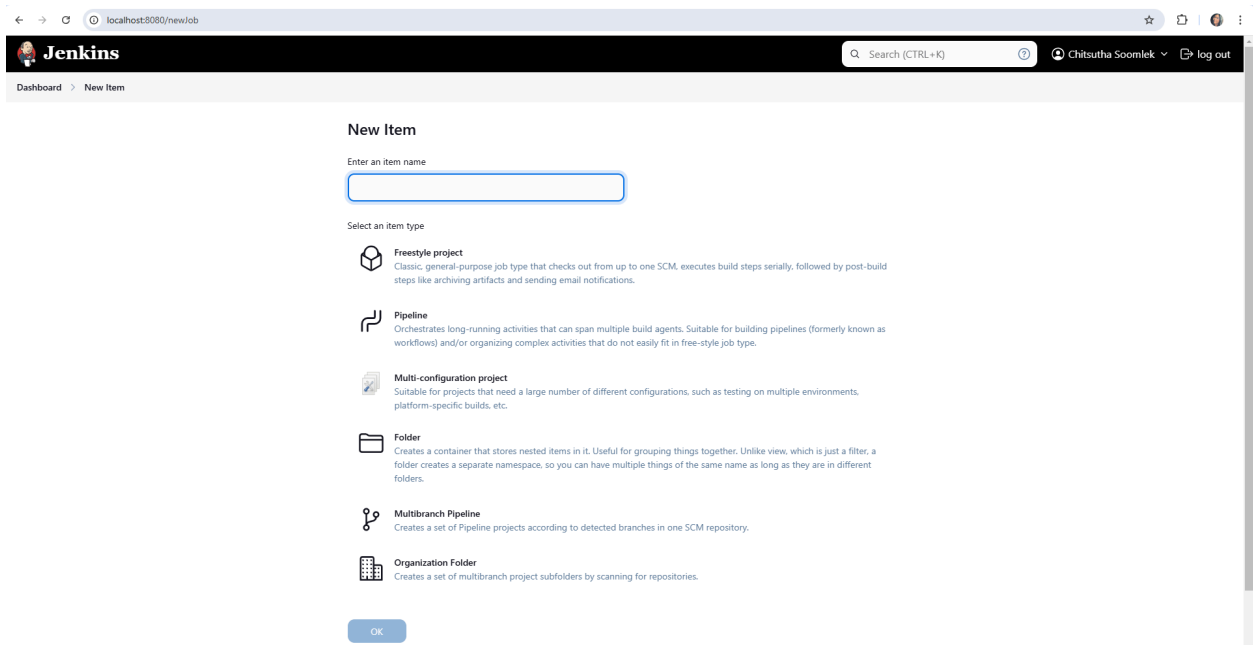
ผศ.ดร.ชิตสุธา สุ่มเล็ก

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 จากนั้นตั้งค่าที่จำเป็นในหน้านี้อย่างครบถ้วน ดังนี้

Description: Lab 8.5

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

CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตสุธา สุ่มเล็ก

Lab Worksheet

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

General

Description

Description: Lab 8.5

Plain text [Preview](#)

☐ Discard old builds [?](#)

☒ GitHub project

Project url [?](#)

https://github.com/Prangfa/Robot_653380021-4.git/

Advanced [?](#)

☐ This project is parameterized [?](#)

☐ Throttle builds [?](#)

CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตสุธา สุ่มเล็ก

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 12:00 PM Coordinated Universal Time.

☐ GitHub hook trigger for GITScm polling ?

☐ Poll SCM ?

CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตสุธา สุ่มเล็ก

Lab Worksheet

```
pipeline {
  agent any
  stages {
    stage('Setup') {
      steps {
        echo 'Setting up environment'
        sh 'pip install -r requirements.txt'
      }
    }
    stage('Start Server') {
      steps {
        echo 'Starting server.py'
        sh 'nohup python server.py &'
      }
    }
    stage('Run Robot Tests') {
      steps {
        echo 'Running Robot Framework tests'
        sh 'robot --outputdir results test/login_test'
      }
    }
    stage('Cleanup') {
      steps {
        echo 'Cleaning up background processes'
        sh 'pkill -f server.py'
      }
    }
  }
}
```

Save

Apply

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

```
pipeline {
  agent any
  stages {
    stage('Setup') {
      steps {
        echo 'Setting up environment'
        sh 'pip install -r requirements.txt'
      }
    }
  }
}
```

CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตสุธา สุ่มเล็ก

Lab Worksheet

```
stage('Start Server') {
  steps {
    echo 'Starting server.py'
    sh 'nohup python server.py &'
  }
}

stage('Run Robot Tests') {
  steps {
    echo 'Running Robot Framework tests'
    sh 'robot --outputdir results test/login_test'
  }
}

stage('Cleanup') {
  steps {
    echo 'Cleaning up background processes'
    sh 'pkill -f server.py'
  }
}
}
```

Post-build action: เพิ่ม Publish Robot Framework test results -> ระบุได้เร็คทอรีที่เก็บไฟล์ผลการทดสอบโดย Robot framework ในรูป xml และ html -> ตั้งค่า Threshold เป็น % ของการทดสอบที่ไม่ผ่าน แล้วนับว่าซอฟต์แวร์มีปัญหา -> 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

25