

Lab Worksheet

ชื่อ-นามสกุล ณธพ จันทร์หอม รหัสนักศึกษา 653380195-1 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

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

Lab Worksheet

```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

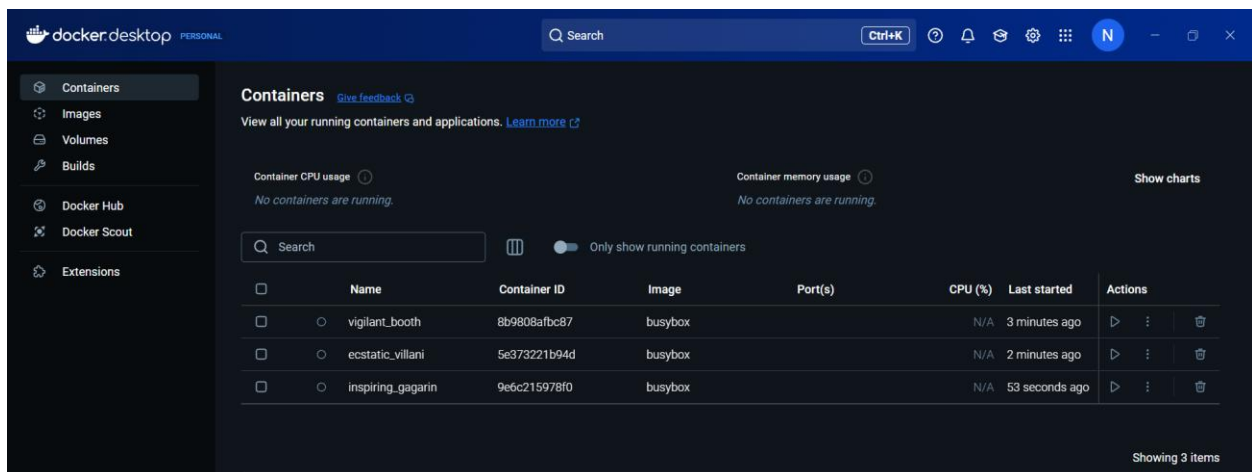
PS D:\Work\Software Engineering\LAB8_1> docker pull busybox
Using default tag: latest
latest: Pulling from library/busybox
9c0abc9c5bd3: Download complete
Digest: sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138bd458fc8257fbf1
Status: Downloaded newer image for busybox:latest
docker.io/library/busybox:latest
PS D:\Work\Software Engineering\LAB8_1> docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
busybox latest a5d0ce49aa80 3 months ago 6.56MB
PS D:\Work\Software Engineering\LAB8_1> |

```

- (1) สิ่งที่อยู่ภายใต้คอลัมน์ Repository คืออะไร คือ ชื่อของ images ที่ดึงมาชื่อว่า busybox
- (2) Tag ที่ใช้บอกถึงอะไร แท็กที่ระบุ ว่าเป็นเวอร์ชันอะไร ในที่นี้ คือเวอร์ชันล่าสุด

5. ป้อนคำสั่ง \$ docker run busybox
6. ป้อนคำสั่ง \$ docker run -it busybox sh
7. ป้อนคำสั่ง ls
8. ป้อนคำสั่ง ls -la
9. ป้อนคำสั่ง exit
10. ป้อนคำสั่ง \$ docker run busybox echo "Hello ชื่อและนามสกุลของนักศึกษา from busybox"
11. ป้อนคำสั่ง \$ docker ps -a

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



Lab Worksheet

```

PS D:\Work\Software Engineering\LAB8_1> docker run busybox
PS D:\Work\Software Engineering\LAB8_1> docker run -it busybox sh
/ # ls
bin      dev      etc      home    lib      lib64    proc    root    sys      tmp      usr      var
/ # ls -la
total 48
drwxr-xr-x  1 root    root      4096 Jan 23 02:43 .
drwxr-xr-x  1 root    root      4096 Jan 23 02:43 ..
-rwxr-xr-x  1 root    root        0 Jan 23 02:43 .dockerenv
drwxr-xr-x  2 root    root     12288 Sep 26 21:31 bin
drwxr-xr-x  5 root    root      360 Jan 23 02:43 dev
drwxr-xr-x  1 root    root      4096 Jan 23 02:43 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 268 root    root        0 Jan 23 02:43 proc
drwx----- 1 root    root      4096 Jan 23 02:43 root
dr-xr-xr-x 11 root    root        0 Jan 23 02:43 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
PS D:\Work\Software Engineering\LAB8_1> docker run busybox echo "Hello สวัสดี สวัสดี from busybox"
Hello สวัสดี สวัสดี from busybox
PS D:\Work\Software Engineering\LAB8_1> docker ps -a

```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
9e6c215978f0	busybox	"echo 'Hello สวัสดี สวัสดี ...'"	7 seconds ago	Exited (0) 6 seconds ago		inspiring_gagarin
5e373221b94d	busybox	"sh"	About a minute ago	Exited (0) About a minute ago		ecstatic_villani
8b9808afbc87	busybox	"sh"	About a minute ago	Exited (0) About a minute ago		vigilant_booth

(1) เมื่อใช้ option -it ในคำสั่ง run ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป

-l จะทำให้ป้อนข้อมูลไปยังคอนเทนเนอร์ได้

-t จะเปิดเทอร์มินัลในคอนเทนเนอร์เพื่อรันคำสั่ง

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

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

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

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
9e6c215978f0	busybox	"echo 'Hello สวัสดี สวัสดี ...'"	7 seconds ago	Exited (0) 6 seconds ago		inspiring_gagarin
5e373221b94d	busybox	"sh"	About a minute ago	Exited (0) About a minute ago		ecstatic_villani
8b9808afbc87	busybox	"sh"	About a minute ago	Exited (0) About a minute ago		vigilant_booth

```

PS D:\Work\Software Engineering\LAB8_1> docker rm 9e6c215978f0
9e6c215978f0

```

Lab Worksheet

แบบฝึกปฏิบัติที่ 8.2: สร้าง Docker file และ Docker image

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

สำหรับเครื่องที่ใช้ระบบปฏิบัติการวินโดวส์ (Windows) บันทึกคำสั่งต่อไปนี้ลงในไฟล์ โดยใช้ Text Editor ที่มี

```
FROM busybox
```

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

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

สำหรับเครื่องที่ใช้ระบบปฏิบัติการ MacOS หรือ Linux บนหน้าต่าง Terminal และป้อนคำสั่งต่อไปนี้

```
$ 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

Lab Worksheet

```

PS D:\Work\Software Engineering\Lab8\Lab8_2> docker build -t dockerfile .
[+] Building 3.9s (6/6) FINISHED                                docker:desktop-linux
=> [internal] load build definition from dockerfile              0.0s
=> => transferring dockerfile: 257B                             0.0s
=> [internal] load metadata for docker.io/library/busybox:latest 0.0s
=> [internal] load .dockerignore                                0.1s
=> => transferring context: 2B                                    0.0s
=> [1/1] FROM docker.io/library/busybox:latest@sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138bd458fc82 3.3s
=> => resolve docker.io/library/busybox:latest@sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138bd458fc82 3.2s
=> [auth] library/busybox:pull token for registry-1.docker.io  0.0s
=> exporting to image                                           0.2s
=> => exporting layers                                           0.0s
=> => exporting manifest sha256:80937832278187d66e55ce9098e1fcl4fa3bedf96ee746d4b2c7a509d789bd2 0.0s
=> => exporting config sha256:37c3d43a6a7c8eee649e87ba9becd83b2beeb7f5e24623e5de4376bc2d81f017 0.0s
=> => exporting attestation manifest sha256:51277a85bdf74729fc33beac5c579d19381114ea11ae14141743e8f10d69323 0.1s
=> => exporting manifest list sha256:7d2732d2c5a0648cbf67436e09f95d20491300251e8644f379047920f4d5ed8b 0.0s
=> => naming to docker.io/library/dockerfile:latest            0.0s
=> => unpacking to docker.io/library/dockerfile:latest         0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/7b3xabevfdduft5s0kww5jn74

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)

```

(1) คำสั่งที่ใช้ในการ run คือ

Docker build -t dockerfile . เป็นคำสั่งสำหรับสร้าง Docker Image จาก Dockerfile

(2) Option -t ในคำสั่ง \$ docker build ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป
ทำให้ Image ที่สร้างขึ้นมีชื่อระบุ หรือ ใช้แท็กเพื่อแยกเวอร์ชันของ Image ช่วยให้่ายต่อการอ้างอิงและจัดการ Image

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"

Lab Worksheet

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

The screenshot shows the Docker Desktop interface. On the left, the 'Containers' tab is selected. The main area displays a table of running containers:

Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
vigilant_booth	8b9808a1bc87	busybox		N/A	1 hour ago	[Stop] [Refresh] [Delete]
ecstatic_villani	5e373221b94d	busybox		N/A	1 hour ago	[Stop] [Refresh] [Delete]
affectionate_sutherland	84e9b9900f19	natonchanhom/lab8_3		N/A	2 minutes ago	[Stop] [Refresh] [Delete]

Below the table, the terminal output for the build and run commands is shown:

```
PS D:\Work\Software Engineering\Lab8\Lab8_3> docker build -t natonchanhom/lab8_3 .
[+] Building 1.4s (5/5) FINISHED
=> [internal] load build definition from dockerfile 0.2s
=> => transferring dockerfile: 167B 0.0s
=> [internal] load metadata for docker.io/library/busybox:latest 0.1s
=> [internal] load .dockerignore 0.1s
=> => transferring context: 2B 0.0s
=> CACHED [1/1] FROM docker.io/library/busybox:latest@sha256:a5d0ce4 0.1s
=> => resolve docker.io/library/busybox:latest@sha256:a5d0ce49aa801d 0.1s
=> => exporting image 0.4s
=> => exporting layers 0.0s
=> => exporting manifest sha256:9a651db12c5fcd0f4f1315b04d3cd1b681ca 0.1s
=> => exporting config sha256:8377072a61a2b9b0f1d6cf001a68bc01109195 0.1s
=> => exporting attestation manifest sha256:3bc31f4519423cbbaa02ae96 0.1s
=> => exporting manifest list sha256:1b5986f9d91f0ad6e661b12d2cc580f 0.1s
=> => naming to docker.io/natonchanhom/lab8_3:latest 0.0s
=> => unpacking to docker.io/natonchanhom/lab8_3:latest 0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/71hu230flmb66a17xcinh8v8f

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:\Work\Software Engineering\Lab8\Lab8_3> docker run natonchanhom/lab8_3
Naton Chanhom 653380195-1
```

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

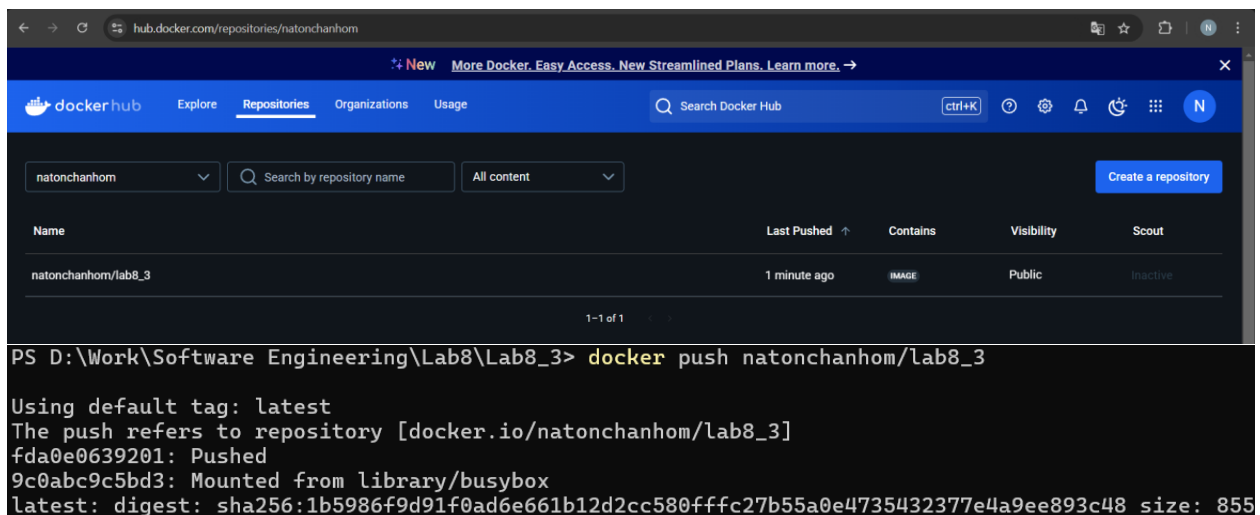
Lab Worksheet

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

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

- ไปที่ Docker Hub กด Tab ชื่อ Tags หรือไปที่ Repository ก็ได้

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

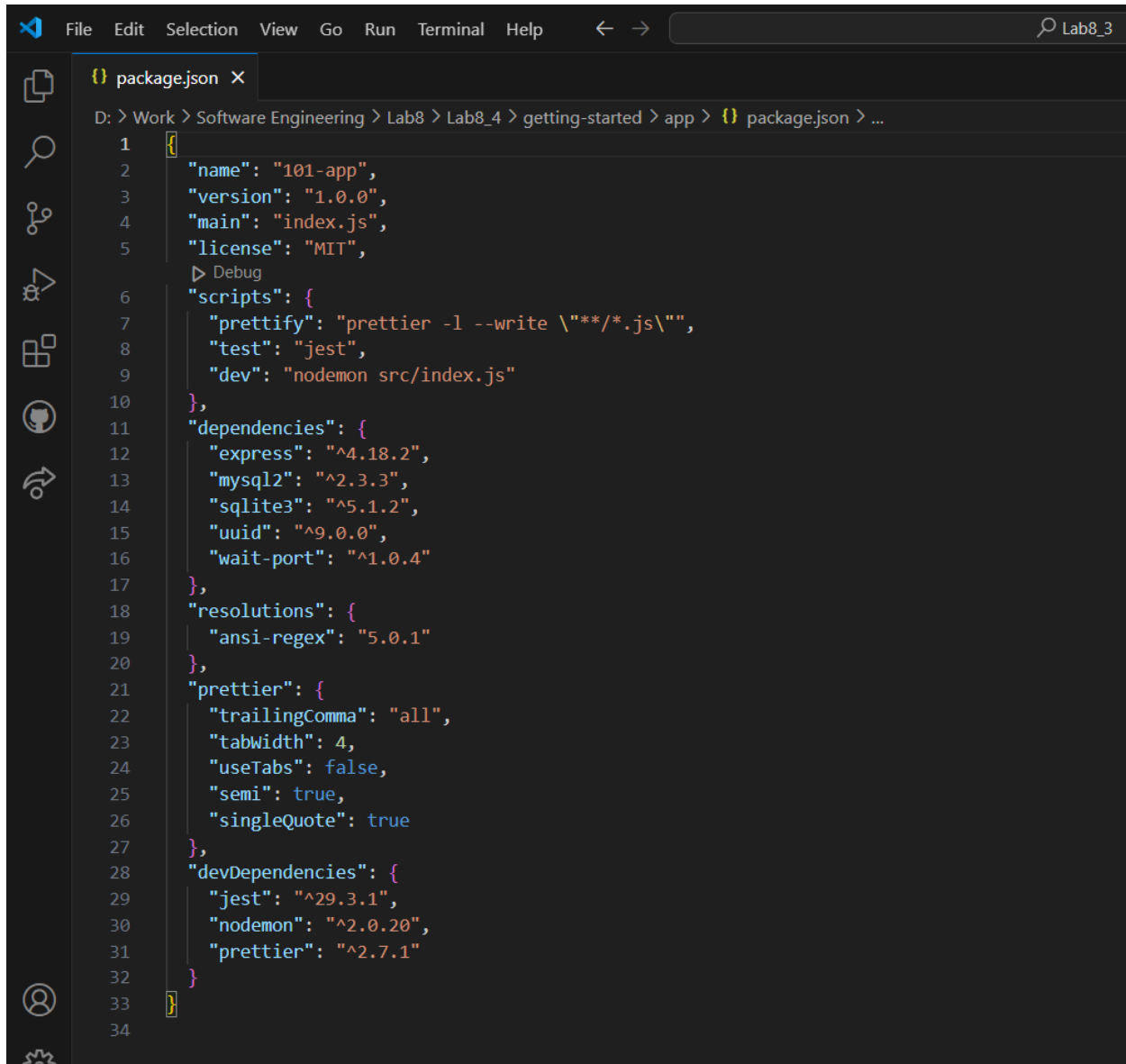


แบบฝึกปฏิบัติที่ 8.4: การ Build แอปพลิเคชันจาก Container image และการ Update แอปพลิเคชัน

- เปิด Command line หรือ Terminal จากนั้นสร้าง Directory ชื่อ Lab8_4
- 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
- เปิดดูองค์ประกอบภายใน getting-started/app เมื่อพบไฟล์ package.json ให้ใช้ Text editor ในการเปิดอ่าน

Lab Worksheet

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



```
1 {
2   "name": "101-app",
3   "version": "1.0.0",
4   "main": "index.js",
5   "license": "MIT",
6   "scripts": {
7     "prettify": "prettier -l --write \"**/*.js\"",
8     "test": "jest",
9     "dev": "nodemon src/index.js"
10  },
11  "dependencies": {
12    "express": "^4.18.2",
13    "mysql2": "^2.3.3",
14    "sqlite3": "^5.1.2",
15    "uuid": "^9.0.0",
16    "wait-port": "^1.0.4"
17  },
18  "resolutions": {
19    "ansi-regex": "5.0.1"
20  },
21  "prettier": {
22    "trailingComma": "all",
23    "tabWidth": 4,
24    "useTabs": false,
25    "semi": true,
26    "singleQuote": true
27  },
28  "devDependencies": {
29    "jest": "^29.3.1",
30    "nodemon": "^2.0.20",
31    "prettier": "^2.7.1"
32  }
33 }
```

4. ภายใต้ getting-started/app ให้สร้าง Dockerfile พร้อมกับใส่เนื้อหาดังต่อไปนี้ลงในไฟล์

FROM node:18-alpine

WORKDIR /app

COPY . .

RUN yarn install --production

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

Lab Worksheet

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:\Work\Software Engineering\Lab8\Lab8_4\getting-started\app> docker build -t myapp_6533801951 .
[+] Building 30.4s (10/10) FINISHED                                docker:desktop-linux
=> [internal] load build definition from Dockerfile                0.1s
=> => transferring dockerfile: 154B                               0.0s
=> [internal] load metadata for docker.io/library/node:18-alpine  3.3s
=> [auth] library/node:pull token for registry-1.docker.io        0.0s
=> [internal] load .dockerignore                                   0.1s
=> => transferring context: 2B                                      0.0s
=> [1/4] FROM docker.io/library/node:18-alpine@sha256:77e3b76b47148e 5.9s
=> => resolve docker.io/library/node:18-alpine@sha256:77e3b76b47148e 0.1s
=> => sha256:6504e29600c8d5213b52cda800370abb3d12639802d 444B / 444B 0.3s
=> => sha256:5650d6de56fd0bb419872b876ac1df28f577b39 1.26MB / 1.26MB 0.7s
=> => sha256:37892ffbfc8a871a10f813803949d18c3015a 40.01MB / 40.01MB 4.2s
=> => sha256:1f3e46996e2966e4faa5846e56e76e3748b7315 3.64MB / 3.64MB 1.4s
=> => extracting sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded 0.2s
=> => extracting sha256:37892ffbfc8a871a10f813803949d18c3015a482051d 1.1s
=> => extracting sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3 0.1s
=> => extracting sha256:6504e29600c8d5213b52cda800370abb3d12639802d0 0.1s
=> [internal] load build context                                  0.6s
=> => transferring context: 4.62MB                                  0.5s
=> [2/4] WORKDIR /app                                           0.4s
=> [3/4] COPY . .                                               0.2s
=> [4/4] RUN yarn install --production                          12.6s
=> exporting to image                                           7.2s
=> => exporting layers                                           4.2s
=> => exporting manifest sha256:1ea235fe34010f6d787d56d83b82890f2b4e 0.1s
=> => exporting config sha256:0179601a9fa56a0c73d7bf56f351a6c80733c0 0.1s
=> => exporting attestation manifest sha256:1b4a2ba9aea348cf2e4c2167 0.1s
=> => exporting manifest list sha256:c29e8b76a15fee9b907b6cf2d484ff 0.1s
=> => naming to docker.io/library/myapp_6533801951:latest        0.0s
=> => unpacking to docker.io/library/myapp_6533801951:latest     2.6s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/lyrxyb6qz2wiosd5ah088bwcp
```

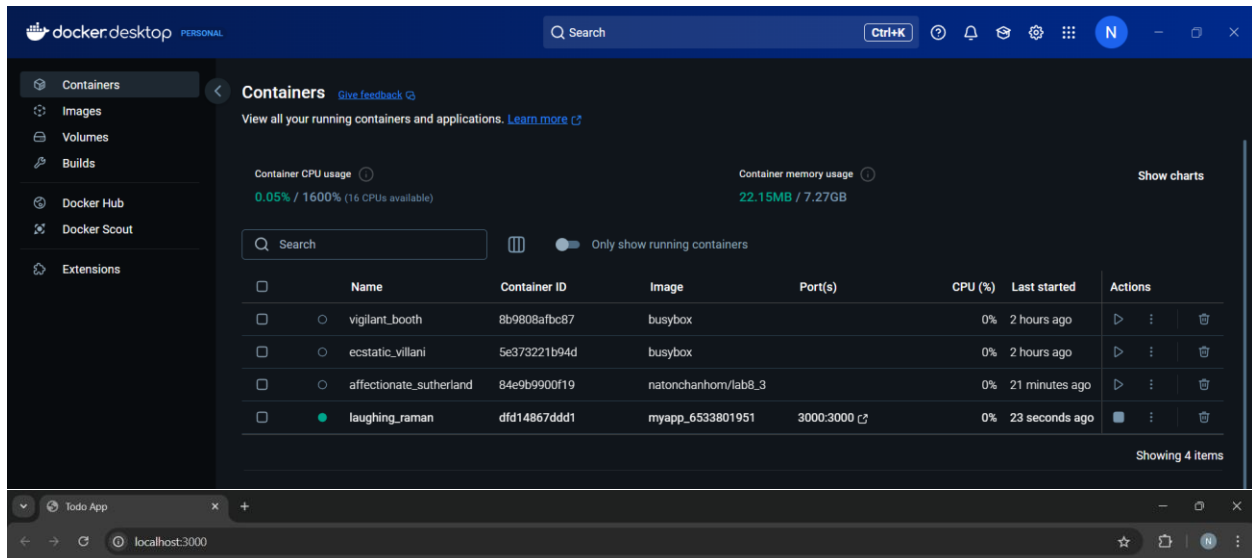
6. ทำการ Start ตัว Container ของแอปพลิเคชันที่สร้างขึ้น โดยใช้คำสั่ง

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

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

[Check point#9] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงผลลัพธ์ที่ได้นบน 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
 - b. Save ไฟล์ให้เรียบร้อย
9. ทำการ Build Docker image โดยใช้คำสั่งเดียวกันกับข้อ 5
10. Start และรัน Container ตัวใหม่ โดยใช้คำสั่งเดียวกันกับข้อ 6

Lab Worksheet

[Check point#10] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงคำสั่งและผลลัพธ์ที่ได้ทางหน้าจอ พร้อมกับตอบคำถามต่อไปนี้

```
PS D:\Work\Software Engineering\Lab8\Lab8_4\getting-started\app> docker build -t myapp_6533801951 .
[+] Building 22.2s (10/10) FINISHED
=> [internal] load build definition from Dockerfile 0.0s
=> => transferring dockerfile: 154B 0.0s
=> [internal] load metadata for docker.io/library/node:18-alpine 2.0s
=> [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@sha256:77e3b76b47148e 0.1s
=> => resolve docker.io/library/node:18-alpine@sha256:77e3b76b47148e 0.1s
=> [internal] load build context 0.1s
=> => transferring context: 8.09kB 0.0s
=> CACHED [2/4] WORKDIR /app 0.0s
=> [3/4] COPY . . 0.2s
=> [4/4] RUN yarn install --production 12.0s
=> exporting to image 7.2s
=> => exporting layers 4.1s
=> => exporting manifest sha256:531a6d384aa74133377c7bbb2202549fd00b 0.1s
=> => exporting config sha256:ff2b4ee795357f9346367b1872058d4779ac01 0.1s
=> => exporting attestation manifest sha256:cla349c-fbcef566b00f6d63a 0.1s
=> => exporting manifest list sha256:06b58b9c1d3655143029f5bb7aa8c7d 0.1s
=> => naming to docker.io/library/myapp_6533801951:latest 0.0s
=> => unpacking to docker.io/library/myapp_6533801951:latest 2.8s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/wkmji3bos9mh2rkkodiowmz
PS D:\Work\Software Engineering\Lab8\Lab8_4\getting-started\app> docker run -dp 3000:3000 myapp_6533801951
56351dc9acblacaba6d1e326efdd8e355136613ff4c8ea151533b44a75026d8a
docker: Error response from daemon: driver failed programming external connectivity on endpoint sharp_dubinsky (9eace0c60f88a00f653b6732c43ef12b057d86c64b3b224798acb246cd354f26): Bind for 0.0.0.0:3000 failed: port is already allocated.
```

(1) Error ที่เกิดขึ้นหมายความว่าอย่างไร และเกิดขึ้นเพราะอะไร

Error นี้ เกิดขึ้นเมื่อ Docker พยายามเชื่อมโยง Port 3000 บนเครื่องคอมพิวเตอร์ที่ใช้งาน ให้กับคอนเทนเนอร์ แต่ Port ดังกล่าว ถูกใช้งานอยู่แล้ว โดยคอนเทนเนอร์อื่น

11. ลบ Container ของ Web application เวอร์ชันก่อนแก้ไขออกจากระบบ โดยใช้วิธีใดวิธีหนึ่งดังต่อไปนี้

a. ผ่าน Command line interface

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

b. ผ่าน Docker desktop

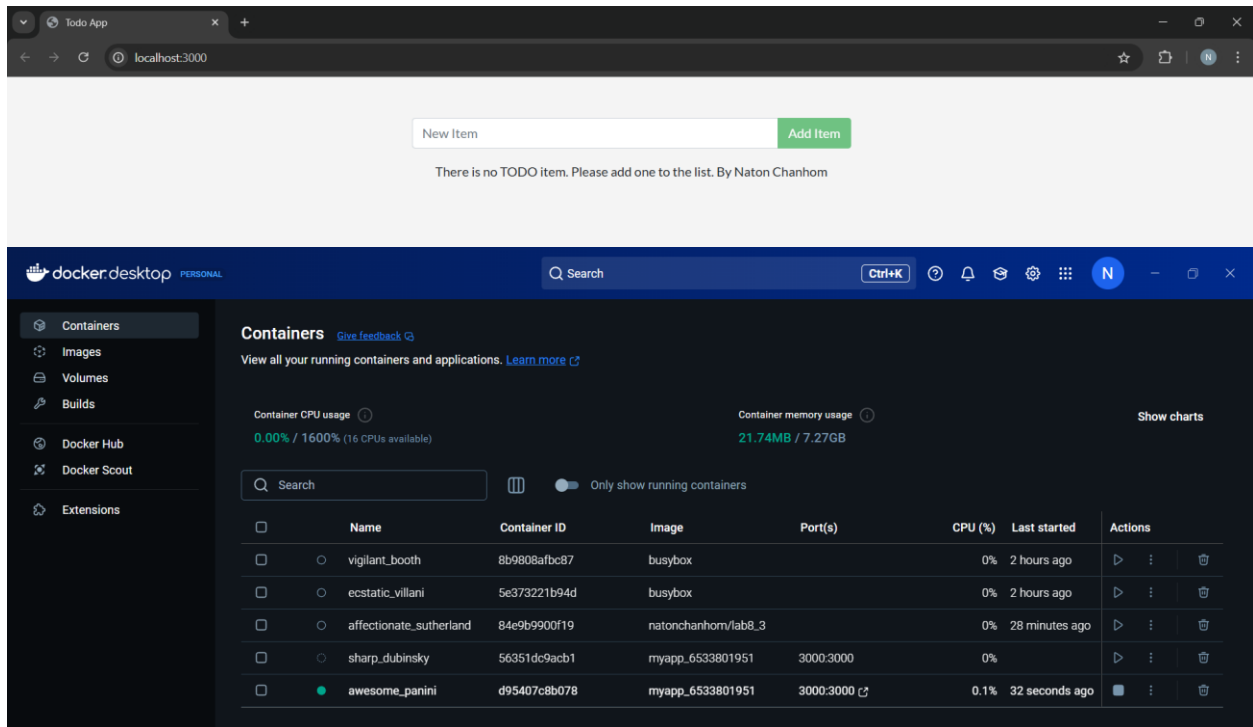
- ไปที่หน้าต่าง Containers
- เลือกไอคอนถังขยะในแถวของ Container ที่ต้องการจะลบ
- ยืนยันโดยการกด Delete forever

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



แบบฝึกปฏิบัติที่ 8.5: เริ่มต้นสร้าง Pipeline อย่างง่ายสำหรับการ Deploy ด้วย Jenkins

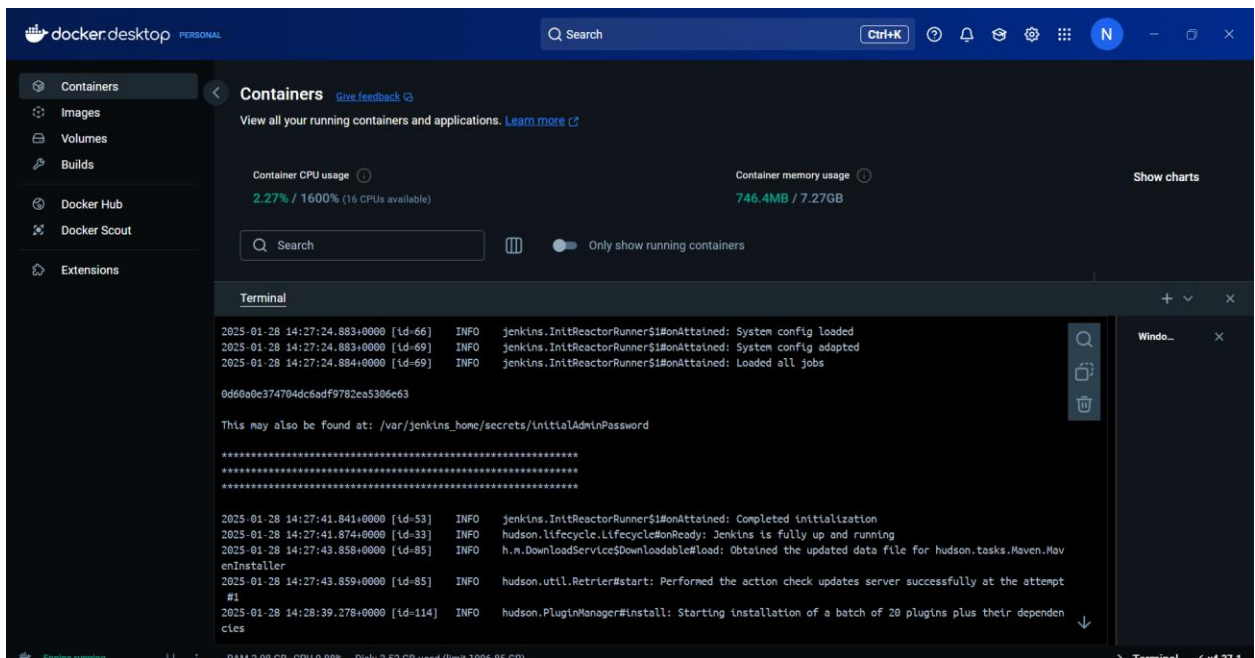
1. เปิด Command line หรือ Terminal บน Docker Desktop
2. ป้อนคำสั่งและทำการรัน container โดยผูกพอร์ต

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

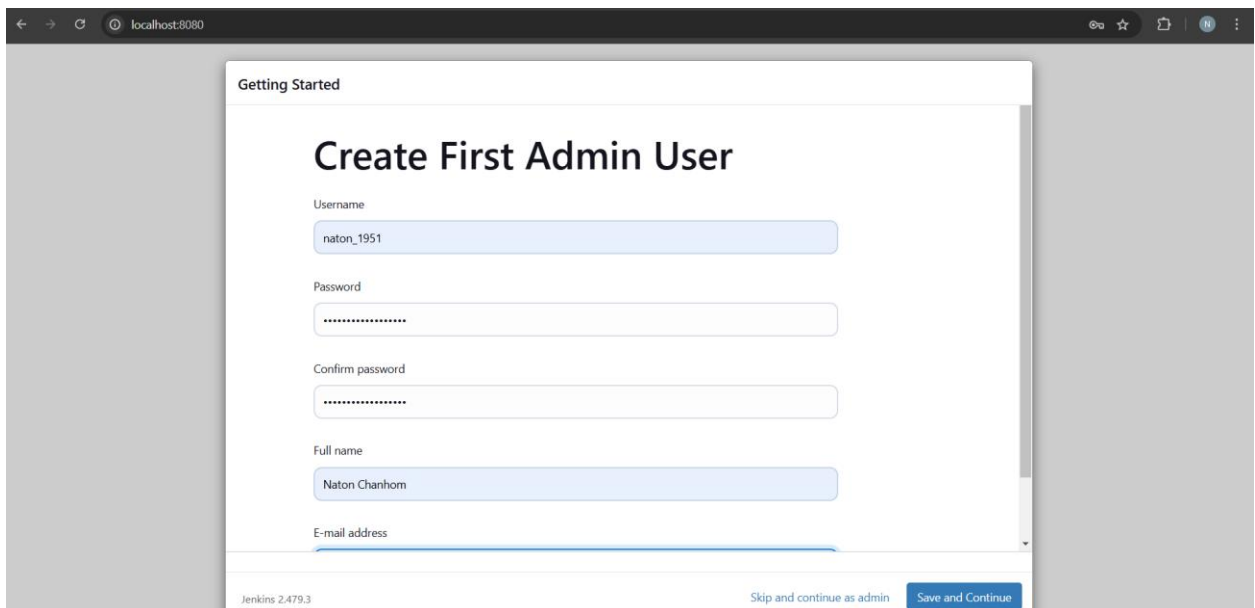
หรือ

```
$ 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

Lab Worksheet

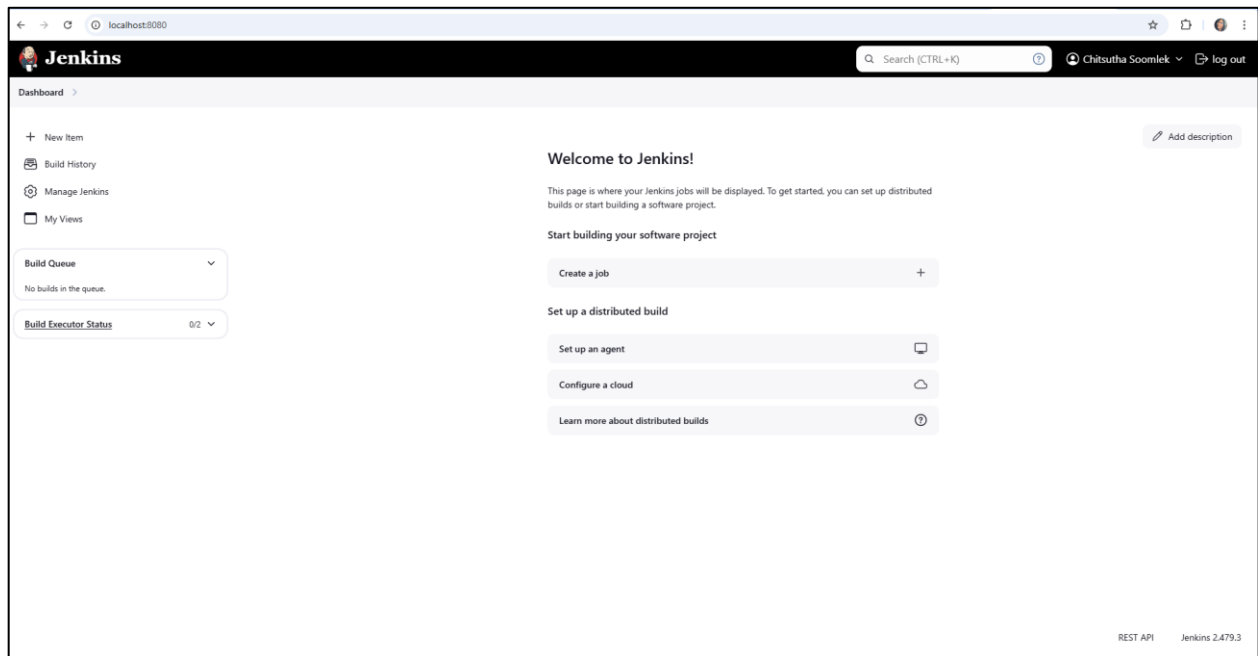


4. เมื่อได้รับการยืนยันว่า Jenkins is fully up and running ให้เปิดบราวเซอร์ และป้อนที่อยู่เป็น localhost:8080
 5. ทำการ Unlock Jenkins ด้วยรหัสผ่านที่ได้ในข้อที่ 3
 6. สร้าง Admin User โดยใช้ username เป็นชื่อจริงของนักศึกษาพร้อมรหัสสี่ตัวท้าย เช่น somsri_3062
- [Check point#13] Capture หน้าจอที่แสดงผลการตั้งค่า

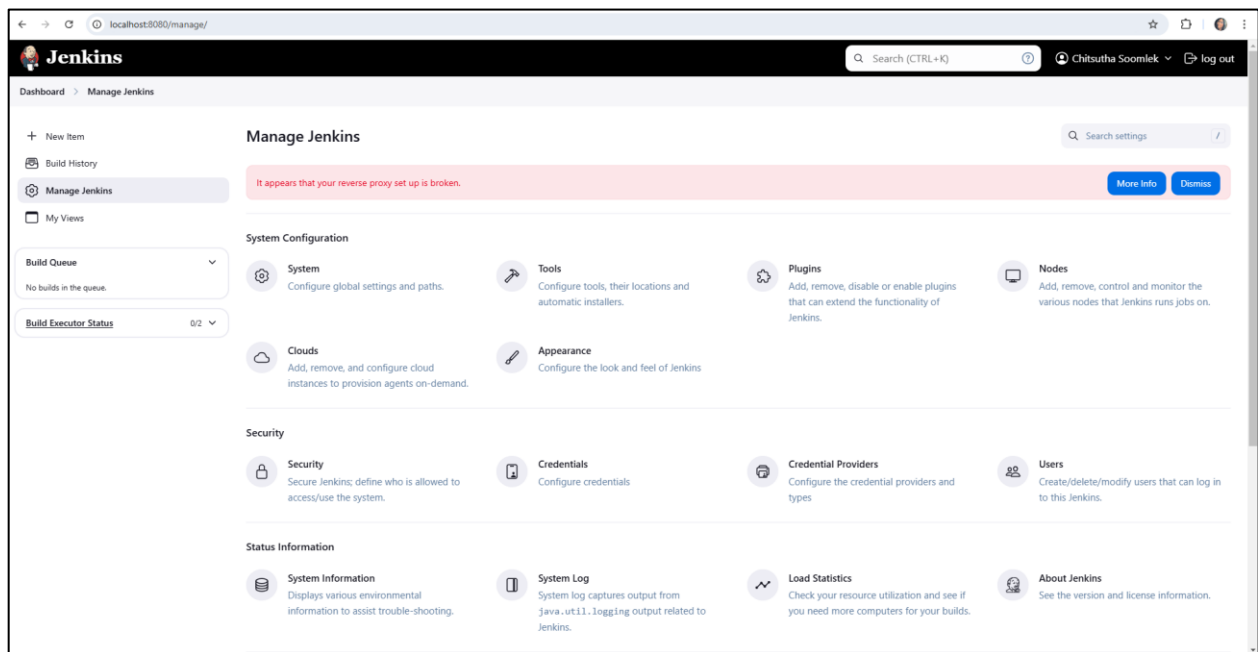


7. กำหนด Jenkins URL เป็น <http://localhost:8080/lab8>
8. เมื่อติดตั้งเรียบร้อยแล้วจะพบหน้าจอ Dashboard ดังแสดงในภาพ

Lab Worksheet

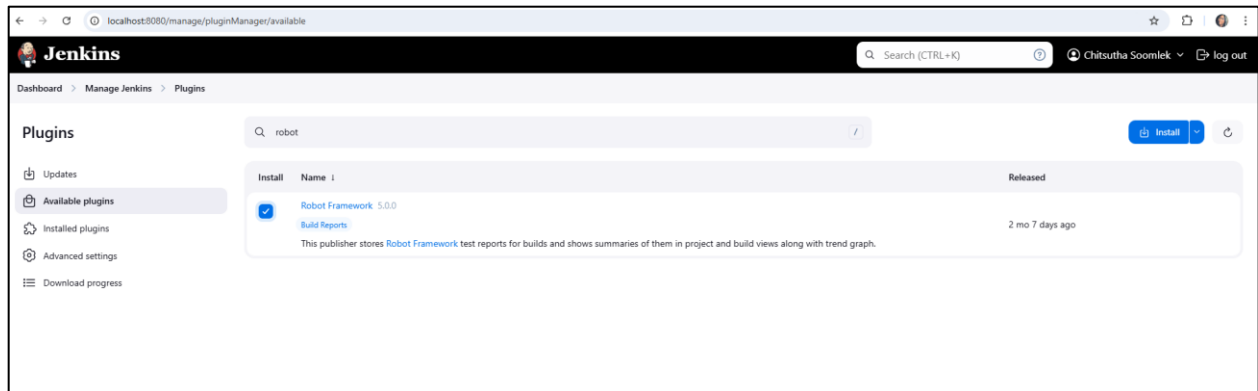


9. เลือก Manage Jenkins แล้วไปที่เมนู Plugins

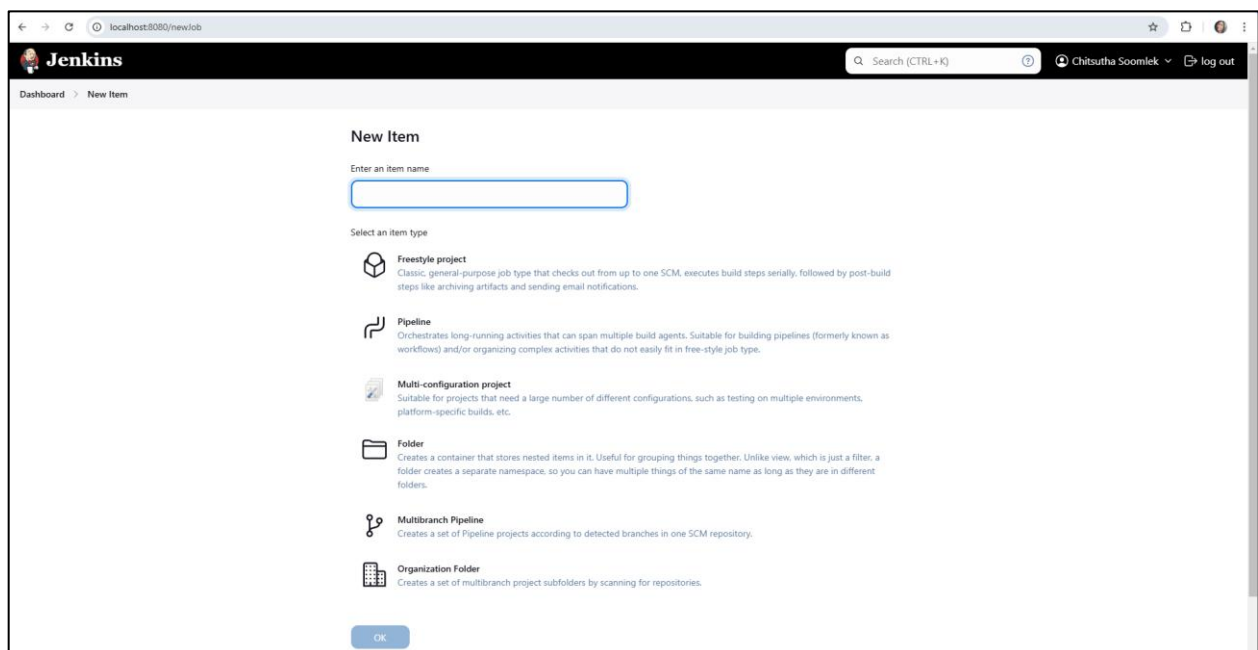


Lab Worksheet

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

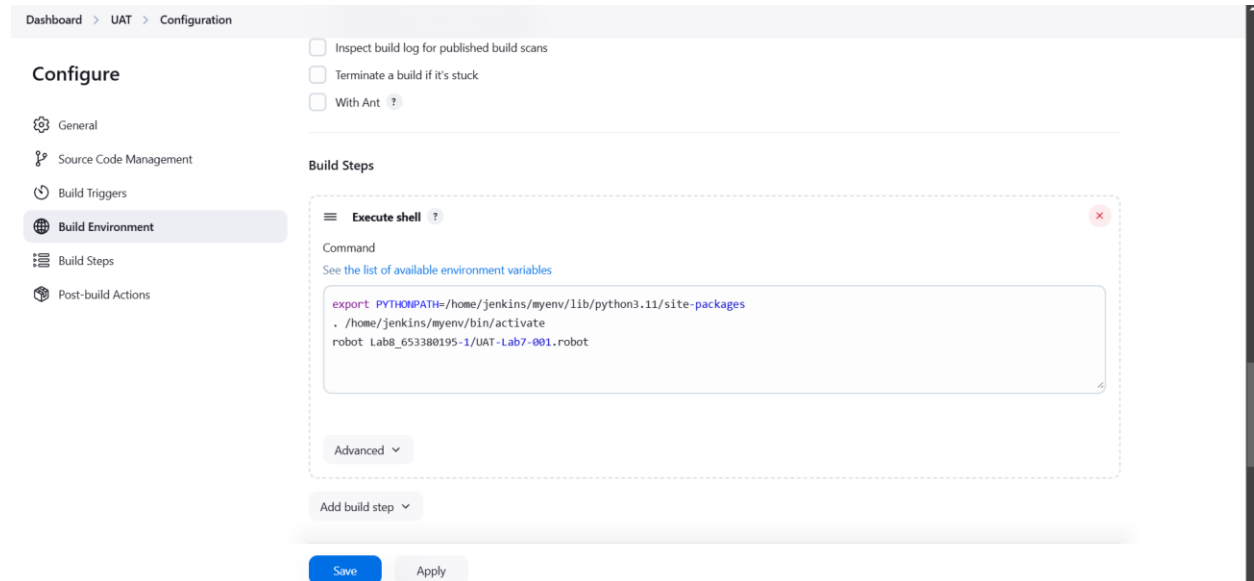
GitHub project: กดเลือก แล้วใส่ Project URL เป็น repository ที่เก็บโค้ด .robot (ดูขั้นตอนที่ 12)

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



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

export PYTHONPATH=/home/jenkins/myenv/lib/python3.11/site-packages

./home/jenkins/myenv/bin/activate

robot Lab8_653380195-1/UAT-Lab7-001.robot

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

The screenshot displays the Jenkins UAT dashboard. The top navigation bar includes the Jenkins logo, a search bar, and user information (Naton Chanhom). The main content area is divided into two sections: 'UAT' and 'Console Output'.

UAT Section:

- Status:** UAT (green checkmark)
- 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 (#68), 50 sec ago
 - Last stable build (#68), 50 sec ago
 - Last successful build (#68), 50 sec ago
 - Last failed build (#64), 48 min ago
 - Last unsuccessful build (#64), 48 min ago
 - Last completed build (#68), 50 sec ago
- Robot Framework Tests Trend (all tests):** A bar chart showing the number of test cases (Skipped, Passed, Failed) across builds. The chart shows a high number of failed tests in previous builds, which have since been resolved.

Console Output Section:

- Status:** Console Output (green checkmark)
- Builds:** #68 (13:37), #67 (13:35)
- Console Output:**

```

Started by user Naton Chanhom
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/kku-computer-science/configuration-management-6533801951.git # timeout=10
Fetching upstream changes from https://github.com/kku-computer-science/configuration-management-6533801951.git
> git --version # timeout=10
> git --version # 'git version 2.39.5'
> git fetch --tags --force --progress -- https://github.com/kku-computer-science/configuration-management-6533801951.git
+refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision 4f7baf6f4cc301bc02cf39529e0688bc77af9d41 (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f 4f7baf6f4cc301bc02cf39529e0688bc77af9d41 # timeout=10
Commit message: "Add Lab8"
First time build. skipping changelog.
[UAT] $ /bin/sh -xe /tmp/jenkins5385260111128355883.sh
+ export PYTHONPATH=/home/jenkins/myenv/lib/python3.11/site-packages

```