

Lab Worksheet

ชื่อ-นามสกุล น.ส.พัฒนิตา เพ็ชรภักดี

รหัสนักศึกษา 653380274-5

Section 2

Lab#8 – Software Deployment Using Docker

วัตถุประสงค์การเรียนรู้

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

Pre-requisite

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

แบบฝึกปฏิบัติที่ 8.1 Hello world - รัน Container จาก Docker image

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

Lab Worksheet

[Check point#1] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงผลลัพธ์ที่ได้พร้อมกับตอบคำถามต่อไปนี้

```
D:\>cd Lab8_1
D:\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

D:\Lab8_1>docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
busybox              latest             a5d0ce49aa80       4 months ago       6.56MB
bde2020/spark-worker 3.0.0-hadoop3.2    fe69e7b8039a       4 years ago        770MB
bde2020/spark-master 3.0.0-hadoop3.2    64a883c48788       4 years ago        770MB
bde2020/hive-metastore-postgresql 2.3.0             9ab91699d151       4 years ago        412MB
bde2020/hadoop-nodemanager 2.0.0-hadoop3.2.1-java8 b5515b537a30       4 years ago        2.05GB
bde2020/hadoop-resourcemanager 2.0.0-hadoop3.2.1-java8 91ca2afe8016       4 years ago        2.05GB
bde2020/hadoop-namenode 2.0.0-hadoop3.2.1-java8 51ad9293ee52       4 years ago        2.05GB
bde2020/hadoop-historyserver 2.0.0-hadoop3.2.1-java8 ab2971d5f8c3       4 years ago        2.05GB
bde2020/hadoop-datanode 2.0.0-hadoop3.2.1-java8 ddf6e9ad55af       4 years ago        2.05GB
bde2020/hive         2.3.2-postgresql-metastore 620267768985       6 years ago        1.93GB
shawnzhu/prestodb    0.181             74237b72a6f0       7 years ago        4.72GB
```

(1) สิ่งที่อยู่ภายใต้คอลัมน์ Repository คืออะไร

Repository คือ ชื่อของ Docker Image ที่ดึงมาหรือสร้างขึ้น

(2) Tag ที่ใช้บ่งบอกถึงอะไร

Tag ใช้สำหรับระบุ เวอร์ชัน ของ Docker Image เพื่อแยกความแตกต่างระหว่าง Image ที่อยู่ใน Repository เดียวกัน

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

```
D:\Lab8_1>docker run busybox

D:\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 28 11:36 .
drwxr-xr-x 1 root root      4096 Jan 28 11:36 ..
-rwxr-xr-x 1 root root      0 Jan 28 11:36 .dockerenv
drwxr-xr-x 2 root root    12288 Sep 26 21:31 bin
drwxr-xr-x 5 root root     360 Jan 28 11:36 dev
drwxr-xr-x 1 root root     4096 Jan 28 11:36 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 283 root root      0 Jan 28 11:36 proc
drwx----- 1 root root     4096 Jan 28 11:36 root
dr-xr-xr-x 11 root root      0 Jan 28 11:36 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:\Lab8_1>docker run busybox echo "Hello พัดผัด พา เพ็ชร กัด from busybox"
Hello พัดผัด พา เพ็ชร กัด from busybox

D:\Lab8_1>docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS
f2f2d8e8ced0   busybox   sharp_torvalds          "echo 'Hello พัดผัด พา..." 10 seconds ago   Exited (0) 9 seconds ago
6a771c8ce73b   busybox   loving_lovelace         "sh"            About a minute ago   Exited (0) About a minute ago
b63bf9cf6726   busybox   objective_goodall       "sh"            About a minute ago   Exited (0) About a minute ago
313bec62e0f2   bde2020/spark-worker:3.0.0-hadoop3.2   spark-worker-1         "/bin/bash /worker.sh" 6 weeks ago       Exited (137) 6 weeks ago
be892dcc6c35   bde2020/hive:2.3.2-postgresql-metastore   hive-server            "entrypoint.sh /bin/..." 6 weeks ago       Exited (137) 6 weeks ago
fe033d434b43   bde2020/spark-master:3.0.0-hadoop3.2   spark-master          "/bin/bash /master.sh" 6 weeks ago       Exited (255) 6 minutes ago   0.0.0.
8177a0480620   bde2020/hadoop-historyserver:2.0.0-hadoop3.2.1-java8   historyserver         "/entrypoint.sh /run..." 6 weeks ago       Up 6 minutes (healthy)      8188/t
cp
28b3f17017b7   bde2020/hive-metastore-postgresql:2.3.0   hive-metastore-postgresql   "docker-entrypoint..." 6 weeks ago       Exited (0) 6 weeks ago
f45a21b6985d   bde2020/hadoop-resourcemanager:2.0.0-hadoop3.2.1-java8   resourcemanager       "/entrypoint.sh /run..." 6 weeks ago       Up 5 minutes (healthy)      8088/t
cp
```

- (1) เมื่อใช้ option -it ในคำสั่ง run ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป
ทำให้สามารถเข้าใช้งาน Container ในลักษณะอินเตอร์แอคทีฟ เช่น การพิมพ์คำสั่งใน Command Line ของ Container นั้นโดยตรง
- (2) คอลัมน์ STATUS จากการรันคำสั่ง docker ps -a แสดงถึงข้อมูลอะไร
สถานะของ Container ว่า Container อยู่ในสถานะใด ณ ปัจจุบัน หรือทำงานล่าสุดเมื่อใด

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

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

```
D:\Lab8_1>docker rm f2f2d8e8ced0
f2f2d8e8ced0
```

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

Lab Worksheet

3. ย้ายตำแหน่งปัจจุบันไปที่ Lab8_2 เพื่อใช้เป็น Working directory

4. สร้าง Dockerfile.swp ไว้ใน Working directory

สำหรับเครื่องที่ใช้ระบบปฏิบัติการวินโดวส์ (Windows) บันทึกคำสั่งต่อไปนี้ลงในไฟล์ โดยใช้ Text Editor ที่มี

```
FROM busybox
```

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

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

สำหรับเครื่องที่ใช้ระบบปฏิบัติการ MacOS หรือ Linux บนหน้าต่าง Terminal และบันทึกคำสั่งต่อไปนี้

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

```
FROM busybox
```

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

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

```
EOF
```

หรือใช้คำสั่ง

```
$ touch Dockerfile
```

แล้วใช้ Text Editor ในการใส่เนื้อหาแทน

5. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้

```
$ docker build -t <ชื่อ Image> .
```

6. เมื่อ Build สำเร็จแล้ว ให้ทำการรัน Docker image ที่สร้างขึ้นในขั้นตอนที่ 5

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

Lab Worksheet

```

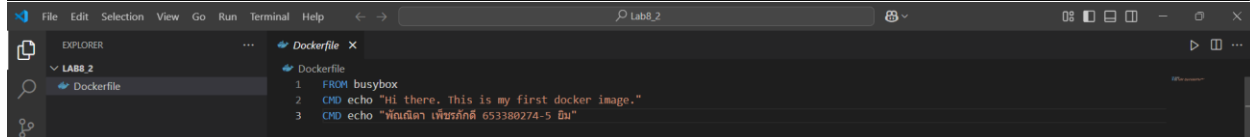
D:\Lab8_2>docker build -t punnita .
[+] Building 3.7s (6/6) FINISHED                                docker:desktop-linux
=> [internal] load build definition from Dockerfile              0.0s
=> => transferring dockerfile: 191B                             0.0s
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS si 0.0s
=> WARN: MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the same stage because 0.0s
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS si 0.0s
=> [internal] load metadata for docker.io/library/busybox:latest 0.0s
=> [internal] load .dockerignore                                 0.0s
=> => transferring context: 2B                                    0.0s
=> [1/1] FROM docker.io/library/busybox:latest@sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138bd458fc82 3.4s
=> => resolve docker.io/library/busybox:latest@sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138bd458fc82 3.4s
=> [auth] library/busybox:pull token for registry-1.docker.io 0.0s
=> => exporting to image                                          0.1s
=> => exporting layers                                           0.0s
=> => exporting manifest sha256:1880f05285b4a7ee8160df3ea29f4b8df28806e99b4b6b79269dea4709216291 0.0s
=> => exporting config sha256:c43d87ccf2997540c76f43d5e267aa6704a99ac20c62c80a650258cbda0f03fc 0.0s
=> => exporting attestation manifest sha256:72433eccf10239267a40fa3483422cd8dd8161142b8dd4e34d3af6385193ff64 0.0s
=> => exporting manifest list sha256:107dc696b1261677249478f58bde987618d35271e5db64166f067e2c43a084df 0.0s
=> => naming to docker.io/library/punnita:latest                0.0s
=> => unpacking to docker.io/library/punnita:latest              0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/ihevfhllvjxtfzaxzf8xjkinn

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)

D:\Lab8_2>docker run punnita
พณณณา เพ็ชรภักดิ์ 653380274-5 ปี ม

```



The screenshot shows the Docker Desktop interface. The top terminal window displays the output of the 'docker build' command, which successfully built the 'punnita' image from the Dockerfile. Below the terminal, the 'EXPLORER' pane shows the 'Dockerfile' with three commands: 1. FROM busybox, 2. CMD echo "Hi there. This is my first docker image.", and 3. CMD echo "พณณณา เพ็ชรภักดิ์ 653380274-5 ปี ม".

(1) คำสั่งที่ใช้ในการ run คือ

docker run punnita

(2) Option -t ในคำสั่ง \$ docker build ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป
-t ย่อมาจาก --tag ใช้เพื่อตั้งชื่อให้กับ Docker Image ที่สร้างขึ้น ทำให้เรียกใช้ Image ได้ง่ายขึ้นและ
ช่วยแยก Image ต่างๆ ได้สะดวก ถ้ามีหลาย Image ในระบบ

แบบฝึกปฏิบัติที่ 8.3: การแชร์ Docker image ผ่าน Docker Hub

1. เปิดใช้งาน Docker desktop และ Login ด้วย Username และ Password ที่ลงทะเบียนกับ Docker Hub เอาไว้
2. เปิด Command line หรือ Terminal จากนั้นสร้าง Directory ชื่อ Lab8_3
3. ย้ายตำแหน่งปัจจุบันไปที่ Lab8_3 เพื่อใช้เป็น Working directory
4. สร้าง Dockerfile.swp ไว้ใน Working directory

สำหรับเครื่องที่ใช้ระบบปฏิบัติการวินโดวส์ บันทึกคำสั่งต่อไปนี้ลงในไฟล์ โดยใช้ Text Editor ที่มี

FROM busybox

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

CMD echo "ชื่อ-นามสกุล รหัสนักศึกษา"

Lab Worksheet

สำหรับเครื่องที่ใช้ระบบปฏิบัติการ MacOS หรือ Linux บนหน้าต่าง Terminal และป้อนคำสั่งต่อไปนี้

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

```
FROM busybox
```

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

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

```
EOF
```

หรือใช้คำสั่ง

```
$ touch Dockerfile
```

แล้วใช้ Text Editor ในการใส่เนื้อหาแทน

7. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้

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

5. ทำการรัน Docker image บน Container ในเครื่องของตัวเองเพื่อทดสอบผลลัพธ์ ด้วยคำสั่ง

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

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

```
D:\Lab8_3>docker build -t punnitap/lab8 .
[+] Building 0.3s (5/5) FINISHED
=> [internal] load build definition from Dockerfile
=> transferring dockerfile: 206B
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS si
=> WARN: MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the same stage because
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS si
=> [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:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138bd
=> resolve docker.io/library/busybox:latest@sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138bd458fc82
=> exporting image
=> exporting layers
=> exporting manifest sha256:2de6f8d346f21206fb53f1e703d76652296ea76c4689d240d3354e9fd46659d3
=> exporting config sha256:c3963434a49d38f372702660aa150db80e610b863a782d1c9e7fa78c16c620b5
=> exporting attestation manifest sha256:c3a04e407392f45ddff67eab93e606f2ff8d5294bd11e7d3724f26eeal6eca3b
=> exporting manifest list sha256:904ca9daa3add29631106f46f67eab93e606f2ff8d5294bd11e7d3724f26eeal6eca3b
=> naming to docker.io/punnitap/lab8:latest
=> unpacking to docker.io/punnitap/lab8:latest

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/wtxeegijxyby90kthab34i74u

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)

D:\Lab8_3>docker run punnitap/lab8
พ นณิ ตา เ พ์ ชรภ กณั 653380274-5
```

Lab Worksheet

6. ทำการ Push ตัว Docker image ไปไว้บน Docker Hub โดยการใช้คำสั่ง

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

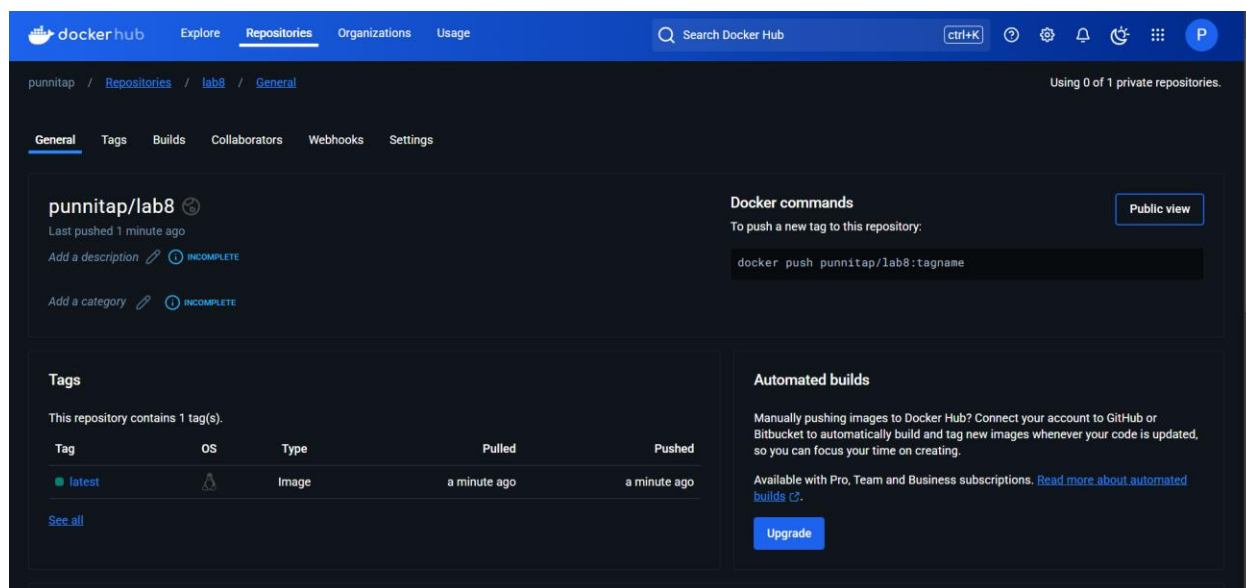
ในกรณีที่ติดปัญหาไม่ได้ Login ไว้ก่อน ให้ใช้คำสั่งต่อไปนี้ เพื่อ Login ก่อนทำการ Push

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

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

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

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



แบบฝึกปฏิบัติที่ 8.4: การ Build แอปพลิเคชันจาก Container image และการ Update แอปพลิเคชัน

1. เปิด Command line หรือ Terminal จากนั้นสร้าง Directory ชื่อ Lab8_4
2. ทำการ Clone ซอร์สโค้ดของเว็บแอปพลิเคชันจาก GitHub repository
<https://github.com/docker/getting-started.git> ลงใน Directory ที่สร้างขึ้น โดยใช้คำสั่ง

```
$ git clone https://github.com/docker/getting-started.git
```
3. เปิดดูองค์ประกอบภายใน getting-started/app เมื่อพบไฟล์ package.json ให้ใช้ Text editor ในการเปิดอ่าน

Lab Worksheet

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

The screenshot shows a terminal window at the top with the following output:

```
D:\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 | 3.92 MiB/s, done.
Resolving deltas: 100% (523/523), done.
```

Below the terminal is the VS Code editor interface. The Explorer sidebar on the left shows the file structure of the 'getting-started' repository, with 'package.json' selected. The main editor area displays the content of 'package.json' with line numbers 1 through 34:

```
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 }
34
```

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 โดยกำหนดชื่อ image เป็น

myapp_รหัสนศ. ไม่มีขีด

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

[Check point#8] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง)

แสดงคำสั่งและผลลัพธ์ที่ได้ทางหน้าจอ

Lab Worksheet

```

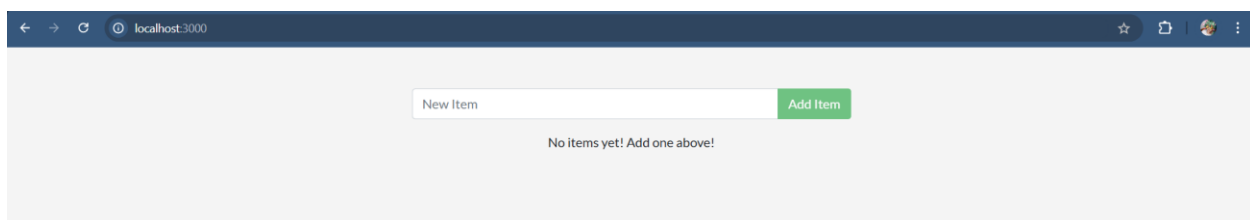
D:\Lab8_4>cd getting-started/app
D:\Lab8_4\getting-started\app>docker build -t myapp_6533802745 .
[+] Building 33.6s (10/10) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 154B
=> [internal] load metadata for docker.io/library/node:18-alpine
=> [auth] library/node:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/4] FROM docker.io/library/node:18-alpine@sha256:974afb6cbc8314dc6502b14243b8a39fbb2d04d975e9059dd066be3e274fbb25
=> resolve docker.io/library/node:18-alpine@sha256:974afb6cbc8314dc6502b14243b8a39fbb2d04d975e9059dd066be3e274fbb25
=> sha256:6594c29608c8d5213b52cda800370abb3d12639802d06b46b6fceb368990ca771 44MB / 44MB
=> sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15bf426d01bc1 1.26MB / 1.26MB
=> sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e0da02525e63167c 40.01MB / 40.01MB
=> sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c24403d592134f0 3.64MB / 3.64MB
=> extracting sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c24403d592134f0
=> extracting sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e0da02525e63167c
=> extracting sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15bf426d01bc1
=> extracting sha256:6504e29608c8d5213b52cda800370abb3d12639802d06b46b6fceb368990ca771
=> [internal] load build context
=> => transferring context: 4.62MB
=> [2/4] WORKDIR /app
=> [3/4] COPY . .
=> [4/4] RUN yarn install --production
=> exporting image
=> => exporting layers
=> => exporting manifest sha256:27c3e4c9f1414ee5f1d280820f5e8269ddb30f19cca9e8720e50283cdf45d491a
=> => exporting config sha256:1620fa1399f63e8604dbb6a140065a6d4473d235ca2533e576863989195f950
=> => exporting attestation manifest sha256:9d322a30384ab079cae2d90237fb55da6a396a66c4dc06f42f3c09d7688648e
=> => exporting manifest list sha256:ba903dc00dd4c2295e962b5c085326a6c152dcb4ec425e5c2067fbb645d23615
=> => naming to docker.io/library/myapp_6533802745:latest
=> => unpacking to docker.io/library/myapp_6533802745:latest
View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/x15vvob3pn6vpyuq06x7ev4br
  
```

6. ทำการ Start ตัว Container ของแอปพลิเคชันที่สร้างขึ้น โดยใช้คำสั่ง

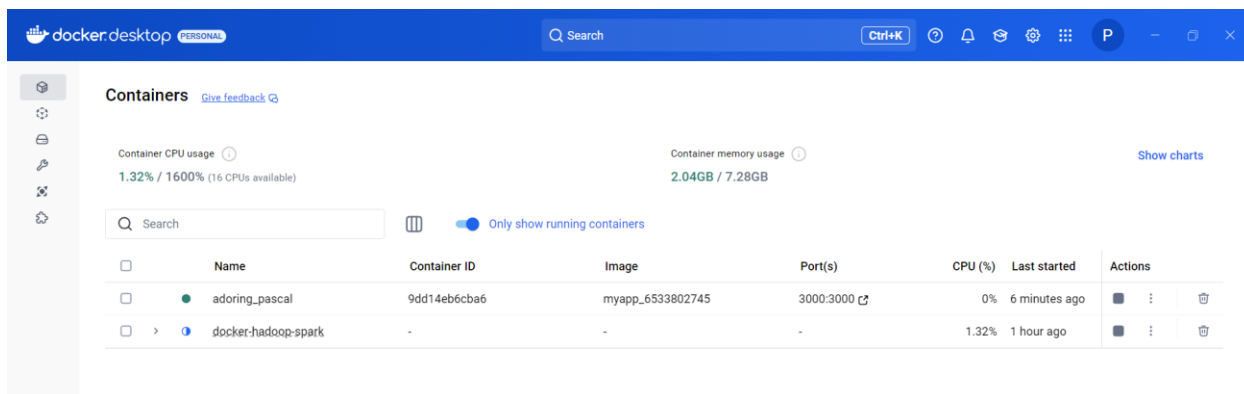
\$ docker run -dp 3000:3000 <myapp_รหัสศ. ไม่มีขีด>

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

[Check point#9] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) 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</p>`

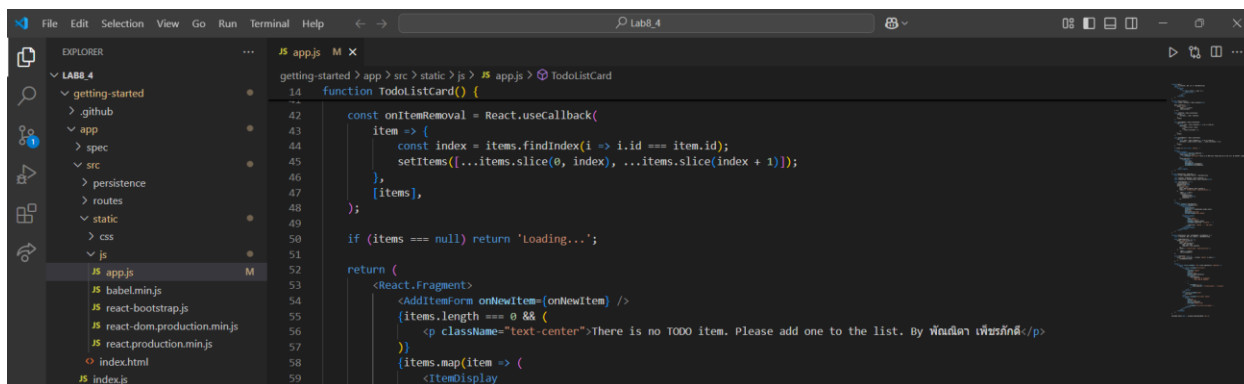
b. Save ไฟล์ให้เรียบร้อย

9. ทำการ Build Docker image โดยใช้คำสั่งเดียวกันกับข้อ 5

10. Start และรัน Container ตัวใหม่ โดยใช้คำสั่งเดียวกันกับข้อ 6

[Check point#10] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง)

แสดงคำสั่งและผลลัพธ์ที่ได้ทางหน้าจอ พร้อมกับตอบคำถามต่อไปนี้



Lab Worksheet

```

D:\Lab8_4\getting-started\app>docker build -t myapp_6533802745 .
[+] Building 2.3s (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@sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d975e9059dd066be3e27 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 image                                                            0.1s
=> => exporting layers                                                            0.0s
=> => exporting manifest sha256:dde99db259b3f5a89570fec229a72185f4b96edce61c01b9af8653e5bf81ed27 0.0s
=> => exporting config sha256:ad1f63beb8cfc0d8d81a4f99f2eaa90b0db81a806cea4caae319eb53271f2fe05 0.0s
=> => exporting attestation manifest sha256:ddc5ee2afa4dc6ad99e299a5358a0f5d8e949945e926ecf4ea0236f8d741637a 0.0s
=> => exporting manifest list sha256:e3cd78293fd2a7e333ca7e7d84920ec530a4d57ef8edaf0731ee0a07fbb212b2 0.0s
=> => naming to docker.io/library/myapp_6533802745:latest                      0.0s
=> => unpacking to docker.io/library/myapp_6533802745:latest                  0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/z5bm2a7fp2hgtyf9y1dhv21q0

D:\Lab8_4\getting-started\app>docker run -dp 3000:3000 myapp_6533802745
2648f12f6699159d933c1786a2156995d41b246a08fa04490a99a9ccc75d792a
docker: Error response from daemon: driver failed programming external connectivity on endpoint suspicious_sanderson (f008533f05545d59b64a154f94d7bb8199e86e881a590792d22ba82158bea42f): Bind for 0.0.0.0:3000 failed: port is already allocated.

```

(1) Error ที่เกิดขึ้นหมายความว่าอย่างไร และเกิดขึ้นเพราะอะไร

Docker ไม่สามารถเชื่อมต่อหรือใช้งานพอร์ตนี้ในการรัน container ได้ เพราะพอร์ต 3000 บนเครื่องถูกใช้งานแล้ว

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

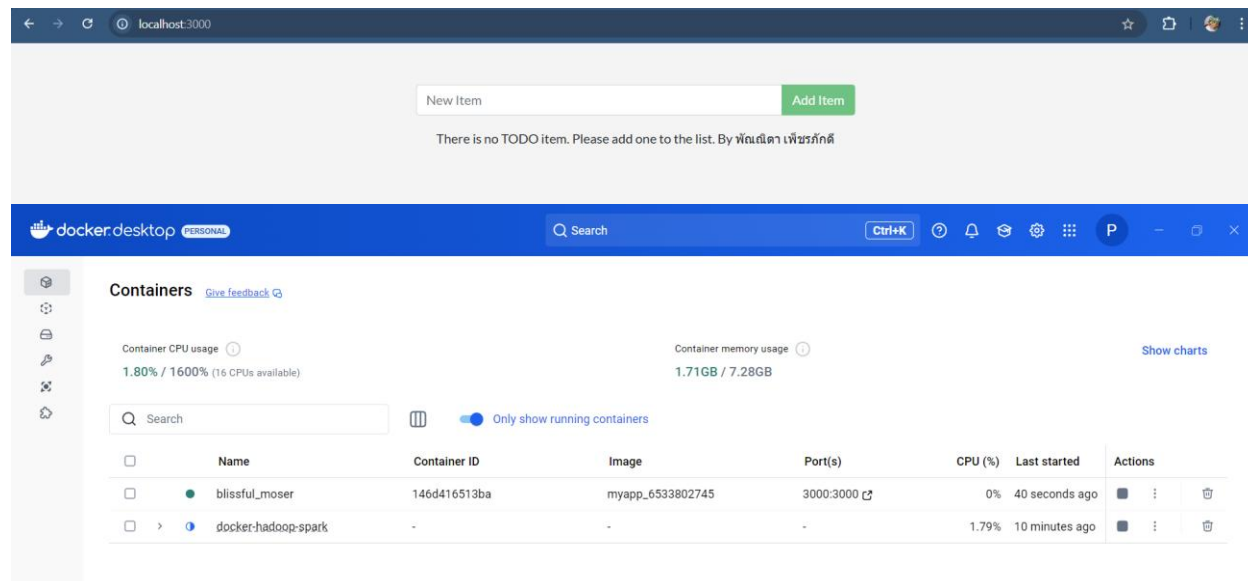
- i. ไปที่หน้าต่าง Containers
- ii. เลือกไอคอนถังขยะในแถวของ Container ที่ต้องการจะลบ
- iii. ยืนยันโดยการกด Delete forever

12. Start และรัน Container ตัวใหม่อีกครั้ง โดยใช้คำสั่งเดียวกันกับข้อ 6

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

Lab Worksheet

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



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
```

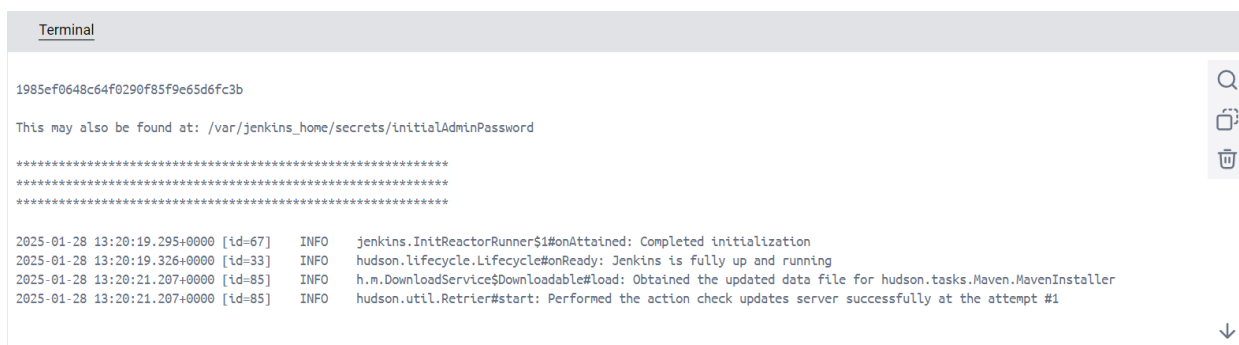
หรือ

```
$ 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

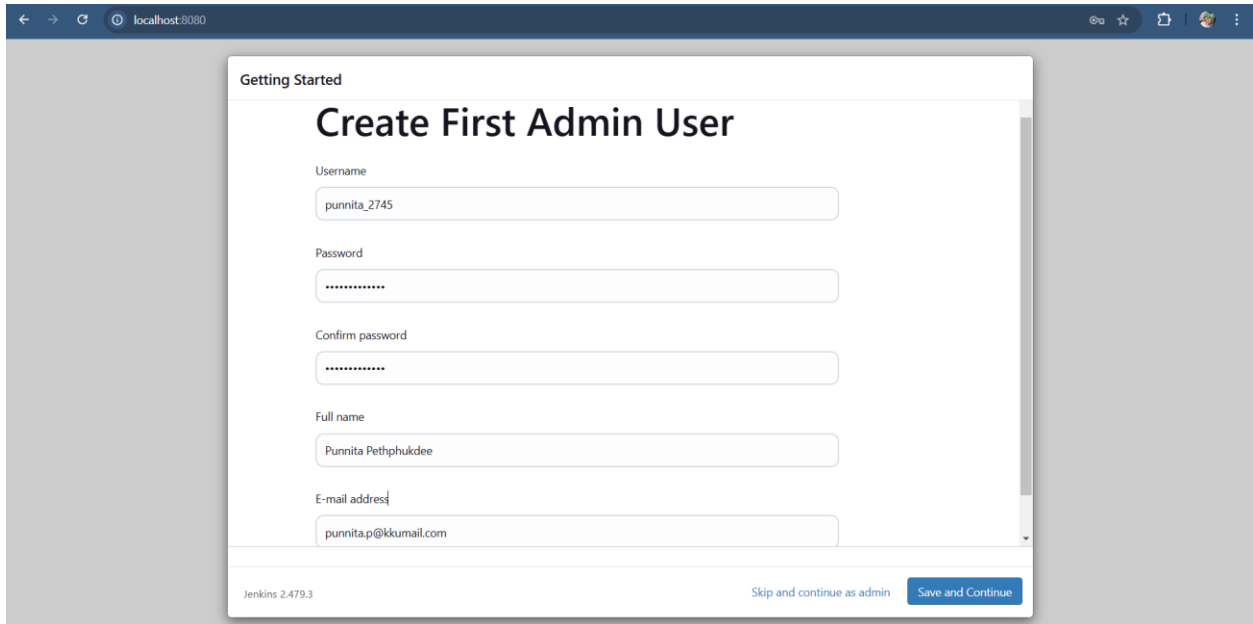
[Check point#12] Capture หน้าจอที่แสดงผล Admin password



Lab Worksheet

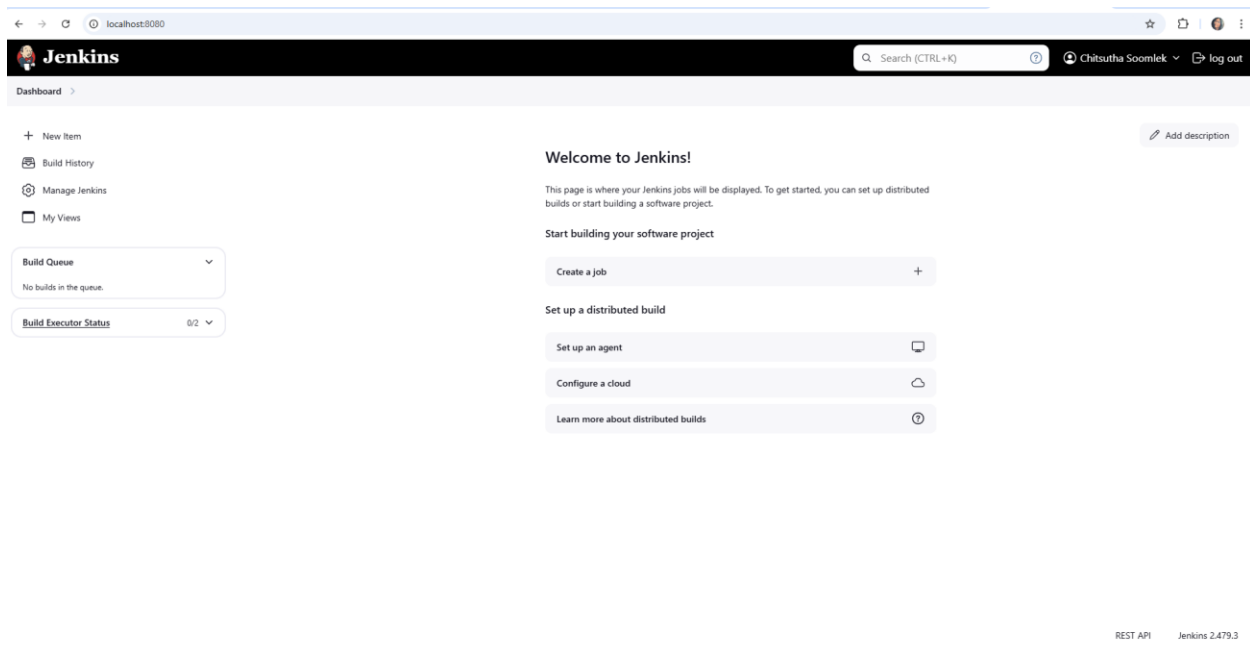
- เมื่อได้รับการยืนยันว่า Jenkins is fully up and running ให้เปิดบราวเซอร์ และป้อนที่อยู่เป็น localhost:8080
- ทำการ Unlock Jenkins ด้วยรหัสผ่านที่ได้ในข้อที่ 3
- สร้าง Admin User โดยใช้ username เป็นชื่อจริงของนักศึกษาพร้อมรหัสสี่ตัวท้าย เช่น somsri_3062

[Check point#13] Capture หน้าจอที่แสดงผลการตั้งค่า

A screenshot of a web browser window showing the Jenkins 'Getting Started' page. The main heading is 'Create First Admin User'. The form contains five input fields: 'Username' with the value 'punnita_2745', 'Password' with masked characters, 'Confirm password' with masked characters, 'Full name' with the value 'Punnita Pethphukdee', and 'E-mail address' with the value 'punnita.p@kkumail.com'. At the bottom left, it says 'Jenkins 2.479.3'. At the bottom right, there are two buttons: 'Skip and continue as admin' and 'Save and Continue'.

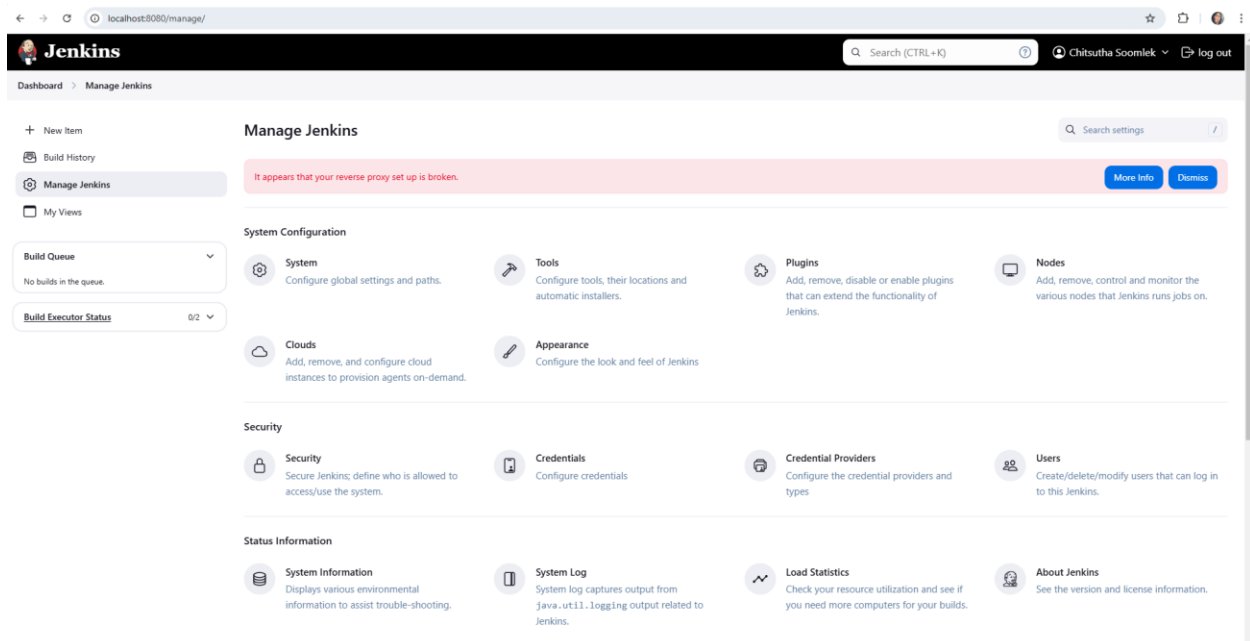
- กำหนด Jenkins URL เป็น <http://localhost:8080/lab8>
- เมื่อติดตั้งเรียบร้อยแล้วจะพบหน้าจอ Dashboard ดังแสดงในภาพ

Lab Worksheet



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

9. เลือก Manage Jenkins แล้วไปที่เมนู Plugins



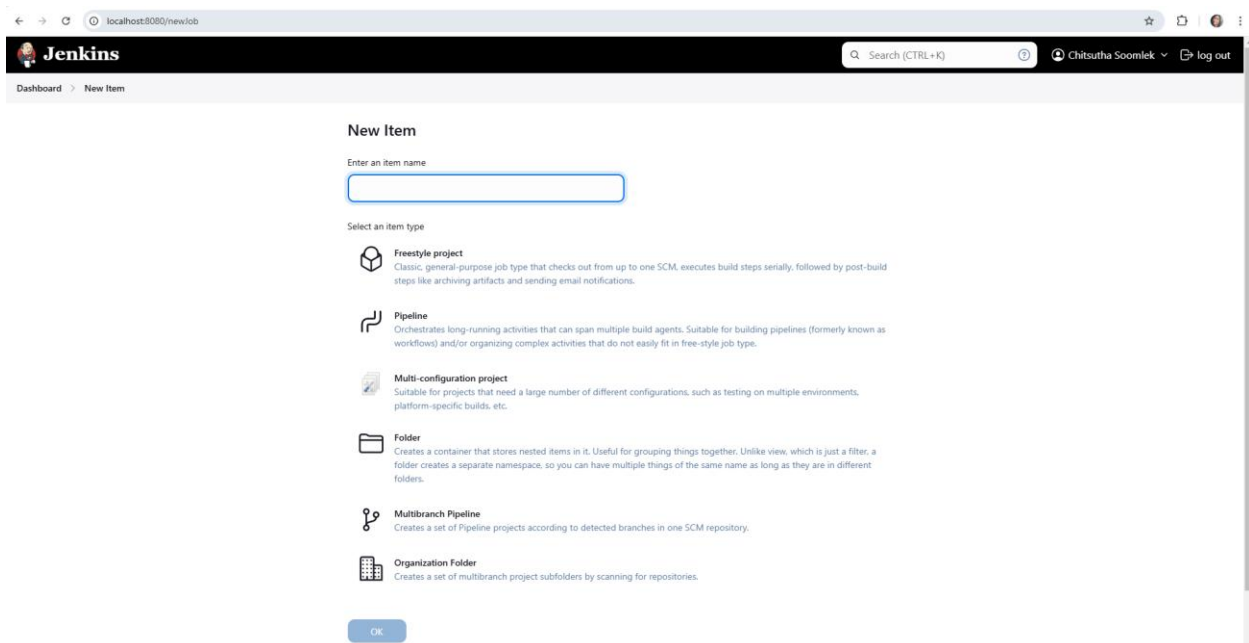
The screenshot shows the Jenkins 'Manage Jenkins' page at localhost:8080/manage/. The top navigation bar is the same as the dashboard. The left sidebar shows 'Manage Jenkins' selected. The main content area has a red warning banner stating 'It appears that your reverse proxy set up is broken.' Below this, the 'System Configuration' section includes links for System, Tools, Plugins, Nodes, Clouds, and Appearance. The 'Security' section includes links for Security, Credentials, Credential Providers, and Users. The 'Status Information' section includes links for System Information, System Log, Load Statistics, and About Jenkins.

Lab Worksheet

10. ไปที่เมนู Available plugins แล้วเลือกติดตั้ง Robotframework เพิ่มเติม



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



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

Description: Lab 8.5

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

Build Trigger: เลือกแบบ Build periodically แล้วกำหนดให้ build ทุก 15 นาที

Lab Worksheet

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

[Check point#14] Capture หน้าจอแสดงการตั้งค่า พร้อมกับตอบคำถามต่อไปนี้

The image displays two screenshots of the Jenkins configuration interface. The top screenshot shows the 'General' tab of the 'Configure' page for a job named 'UAT'. It includes fields for 'Description' (Lab 8.5), 'Project url' (https://github.com/zunnothere/Lab7.git), and checkboxes for 'Discard old builds' and 'GitHub project'. The bottom screenshot shows the 'Source Code Management' tab, where 'Git' is selected as the provider. It displays a 'Repository URL' (https://github.com/zunnothere/Lab7.git/) and a 'Credentials' dropdown set to '- none -'. Both screenshots show the Jenkins logo and navigation menu on the left, and a search bar and user profile on the right.

Lab Worksheet

Dashboard > UAT > Configuration

Configure

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

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 1:50:22 PM Coordinated Universal Time; would next run at Tuesday, January 28, 2025 at 2:05:22 PM Coordinated Universal Time.
- ☐ GitHub hook trigger for GITScm polling ?
- ☐ Poll SCM ?

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

```
robot Test/complete_input.robot
robot Test/incomplete_input.robot
```

Advanced ▾

Add build step ▾

Post-build Actions

Publish Robot Framework test results ?

Directory of Robot output

Path to directory containing robot.xml and html files (relative to build workspace)

Test/

Advanced ▾ Edited

Thresholds for build result ?

🟡%

20.0

🟢%

80.0

☒ DEPRECATED! THIS FLAG DOES NOTHING! - Use thresholds for critical tests only

☐ Include skipped tests in total count for thresholds

Dashboard > UAT > Configuration

Configure

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

Post-build Actions

Publish Robot Framework test results ?

Directory of Robot output

Path to directory containing robot.xml and html files (relative to build workspace)

Test/

Advanced ▾ Edited

Thresholds for build result ?

🟡%

20.0

🟢%

80.0

☒ DEPRECATED! THIS FLAG DOES NOTHING! - Use thresholds for critical tests only

☐ Include skipped tests in total count for thresholds

Save Apply

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

Lab Worksheet

robot Test/complete_input.robot

robot Test/incomplete_input.robot

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

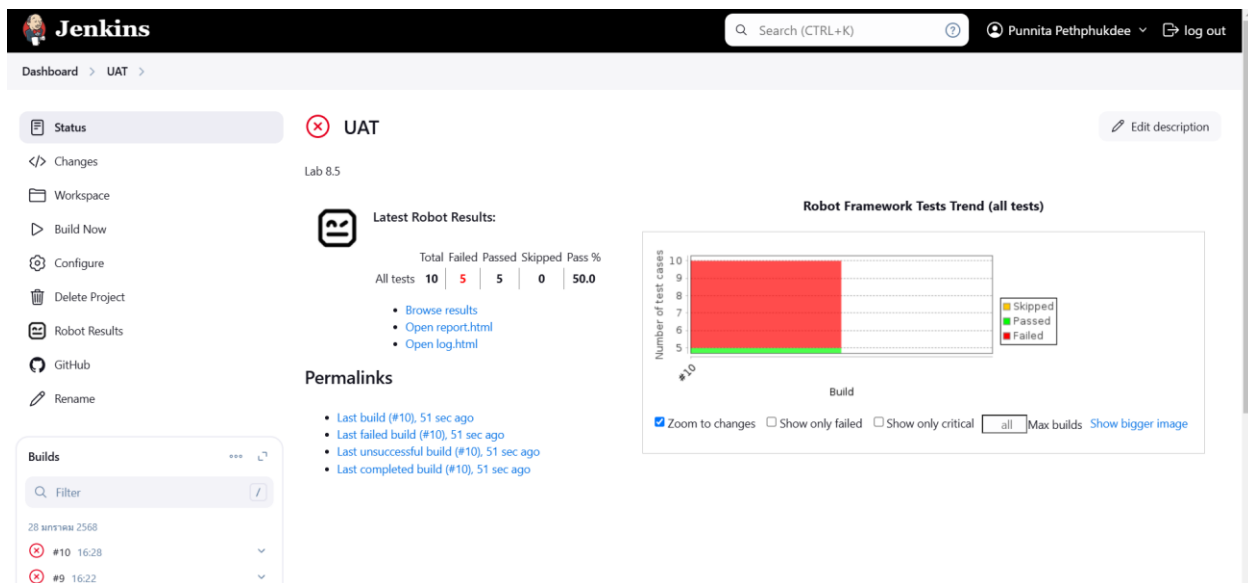
ระบุได้เร็คทอรีที่เก็บไฟล์ผลการทดสอบโดย Robot framework ในรูป xml และ html -> ตั้งค่า Threshold เป็น % ของการทดสอบที่ไม่ผ่านแล้วนับว่าซอฟต์แวร์มีปัญหา -> ตั้งค่า Threshold เป็น %

ของการทดสอบที่ผ่านแล้วนับว่าซอฟต์แวร์มีอยู่ในสถานะที่สามารถนำไปใช้งานได้ (เช่น 20, 80)

13. กด Apply และ Save

14. สั่ง Build Now

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



Lab Worksheet

The screenshot shows the Jenkins web interface. The top navigation bar includes the Jenkins logo, a search bar, and the user name 'Punnita Pethphukdee' with a 'log out' link. The breadcrumb trail is 'Dashboard > UAT > #10 > Console Output'. On the left sidebar, the 'Console Output' tab is selected. The main area displays the console output for build #10, which failed. The output shows the build process cloning a repository, checking out a revision, and attempting to execute a shell script. The script fails because a robot named 'Test/complete_input.robot' is not found. The build is marked as a failure, and the publisher starts, but the overall result is 'FAILURE'.

Console Output

```

Started by user Punnita Pethphukdee
Running as SYSTEM
Building in workspace /var/jenkins_home/workspace/UAT
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/zunnothere/Lab7.git/
> git init /var/jenkins_home/workspace/UAT # timeout=10
Fetching upstream changes from https://github.com/zunnothere/Lab7.git/
> git --version # timeout=10
> git --version # 'git version 2.39.5'
> git fetch --tags --force --progress -- https://github.com/zunnothere/Lab7.git/ +refs/heads/*:refs/remotes/origin/* # timeout=10
> git config remote.origin.url https://github.com/zunnothere/Lab7.git/ # timeout=10
> git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision 768962d9af1efedfa1a77eefdd052f298d013a7d (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f 768962d9af1efedfa1a77eefdd052f298d013a7d # timeout=10
Commit message: "Add files via upload"
First time build. Skipping changelog.
[UAT] $ /bin/sh -xe /tmp/jenkins4965190018709391692.sh
+ robot Test/complete_input.robot
/tmp/jenkins4965190018709391692.sh: 2: robot: not found
Build step 'Execute shell' marked build as failure
Robot results publisher started...
INFO: Checking test criticality is deprecated and will be dropped in a future release!
-Parsing output xml:
Done!
-Copying log files to build dir:
Done!
-Assigning results to build:
Done!
-Checking thresholds:
Done!
Done publishing Robot results.
Finished: FAILURE

```