

Lab Worksheet

ชื่อ-นามสกุล จุฬาลักษณ์ จันทศรี รหัสนักศึกษา 653380124-4 Section 1

Lab#8 – Software Deployment Using Docker

วัตถุประสงค์การเรียนรู้

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

Pre-requisite

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

แบบฝึกปฏิบัติที่ 8.1 Hello world - รัน Container จาก Docker image

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

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

Lab Worksheet

```

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

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

PS C:\Users\USER> cd D:\Lab8_1
PS D:\Lab8_1> docker pull busybox
Using default tag: latest
latest: Pulling from library/busybox
9c0abc9c5bd3: Pull complete
Digest: sha256:a5d0cc49aa801d475da48f8cb163c354ab95cab073cd3c138bd458fc8257fbf1
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
PS D:\Lab8_1> docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
base-hadoop          1.0                ac9bdcdec00d7      7 weeks ago        2.34GB
<none>               <none>             e81c487d8079       7 weeks ago        2.34GB
bde2020/spark-master 3.0.0-hadoop3.2    f0a4c57d8b16       4 years ago        433MB
bde2020/hive-metastore-postgresql 2.3.0             7ab9e8f93813       4 years ago        275MB
bde2020/hadoop-nodemanager 2.0.0-hadoop3.2.1-java8 4c47dabd148f       4 years ago        1.37GB
bde2020/hadoop-resource-manager 2.0.0-hadoop3.2.1-java8 3deba4a1885f       4 years ago        1.37GB
bde2020/hadoop-namenode 2.0.0-hadoop3.2.1-java8 839ec11d95f8       4 years ago        1.37GB
bde2020/hadoop-historyserver 2.0.0-hadoop3.2.1-java8 173c52d1f624       4 years ago        1.37GB
bde2020/hadoop-datanode 2.0.0-hadoop3.2.1-java8 df288ee0a7f9       4 years ago        1.37GB
bde2020/hive         2.3.2-postgresql-metastore 87f5c9f4e2df       6 years ago        1.17GB
shawnzhu/prestodb   0.181             7cc5e6c14cc8       7 years ago        3.46GB
PS D:\Lab8_1>

```

(1) สิ่งที่อยู่ภายใต้คอลัมน์ Repository คืออะไร

คือชื่อของ image ที่เก็บไว้บน Docker Hub

(2) Tag ที่ใช้บ่งบอกถึงอะไร

ระบุ image version ภายใน 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

Lab Worksheet

```

Terminal
PS D:\Lab8_1> docker run busybox
PS D:\Lab8_1> docker run -it busybox sh
/ # ls
bin      etc      lib      proc     sys      usr
dev      home    lib64    root     tmp      var
/ # ls -la
total 48
drwxr-xr-x 1 root root      4096 Jan 22 04:20 .
drwxr-xr-x 1 root root      4096 Jan 22 04:20 ..
-rwxr-xr-x 1 root root      0 Jan 22 04:20 .dockerenv
drwxr-xr-x 2 root root     12288 Sep 26 21:31 bin
drwxr-xr-x 5 root root      360 Jan 22 04:20 dev
drwxr-xr-x 1 root root      4096 Jan 22 04:20 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 239 root root      0 Jan 22 04:20 proc
drwxr-xr-x 1 root root      4096 Jan 22 04:20 root
dr-xr-xr-x 11 root root      0 Jan 22 04:20 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:\Lab8_1> docker run busybox echo "Hello chulaluck chansri from busybox"
Hello chulaluck chansri from busybox
PS D:\Lab8_1> docker ps -a
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS              PORTS   NAMES
23ff6765de27   busybox    "echo 'Hello chulalu..." 11 seconds ago Exited (0) 11 seconds ago          vi
gorous_sutherland
5a3ad9a6d23a   busybox    "sh"                     About a minute ago Exited (0) About a minute ago          lo
4e8a167cf173   bitnami/spark:3.3.1 "/opt/bitnami/script..." 2 weeks ago Exited (137) 2 weeks ago          la
b05-spark-worker-1
a22a153a090b   jupyter/pyspark-notebook:spark-3.3.1 "tiny -g -- start-no..." 2 weeks ago Exited (0) 2 weeks ago          la
b05-jupyter-1
77c2bf3c6b13   bitnami/spark:3.3.1 "/opt/bitnami/script..." 2 weeks ago Exited (137) 2 weeks ago          la
b05-spark-1
b05-spark-1
PS D:\Lab8_1>

```

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

(1) เมื่อใช้ option -it ในคำสั่ง run ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป

ทำให้สามารถเข้าสู่ shell ภายใน container และสามารถโต้ตอบกลับได้ เช่นคำสั่ง ls

(2) คอลัมน์ STATUS จากการรันคำสั่ง docker ps -a แสดงถึงข้อมูลอะไร

ใช้เพื่อแสดงสถานะปัจจุบันของ container แต่ละตัว และการดูปัญหาจาก exit code รวมไปถึงระยะเวลาที่ container กำลังรันอยู่หรือหยุดทำงานไปแล้ว ซึ่ง busybox status Exited (0) 11 seconds ago บ่งบอกว่า หยุดทำงานสำเร็จด้วย exit code 0 เมื่อ 10 วินาทีที่แล้ว

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

```

PS D:\Lab8_1> docker rm 23ff6765de27
23ff6765de27
PS D:\Lab8_1>

```

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

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

Lab Worksheet

The screenshot shows a Windows terminal window with the following output from a Docker build command:

```

PS D:\Lab8 2> docker build -t ththth .
[+] Building 0.1s (5/5) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 156B
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 2)
=> WARN: MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the same stage because only the last one will b
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 3)
=> [internal] load metadata for docker.io/library/busybox:latest
=> [internal] load .dockerignore
- 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)
- JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 2)

What's next:
  View a summary of image vulnerabilities and recommendations ->docker scout quickview
PS D:\Lab8 2> docker run ththth
"chulaluck chansri 653380124-4 oat"
  
```

Below the terminal, a Dockerfile editor shows the following content:

```

1 FROM busybox
2 CMD echo "Hi there. This is my first docker image."
3 CMD echo "chulaluck chansri 653380124-4 oat"
  
```

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

(1) คำสั่งที่ใช้ในการ run คือ

`docker build -t ththth .`

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

ช่วยตั้งชื่อและ tag ให้กับ image เพื่อความสะดวกในการอ้างอิงและใช้งานในอนาคต หากไม่ใช่ ก็จะใช้ Image ID แทน

Lab Worksheet

แบบฝึกปฏิบัติที่ 8.3: การแชร์ Docker image ผ่าน Docker Hub

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

สำหรับเครื่องที่ใช้ระบบปฏิบัติการวินโดวส์ บันทึกคำสั่งต่อไปนี้ลงในไฟล์ โดยใช้ Text Editor ที่มี

```
FROM busybox
```

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

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

สำหรับเครื่องที่ใช้ระบบปฏิบัติการ MacOS หรือ Linux บนหน้าต่าง Terminal และป้อนคำสั่งต่อไปนี้

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

```
FROM busybox
```

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

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

```
EOF
```

หรือใช้คำสั่ง

```
$ touch Dockerfile
```

แล้วใช้ Text Editor ในการใส่เนื้อหาแทน

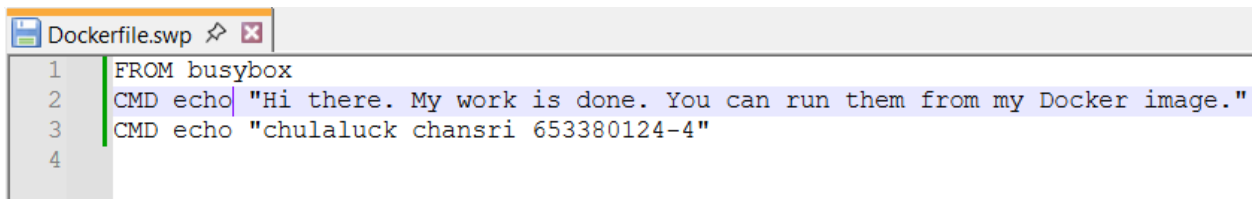
7. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้

```
$ 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



```
1 FROM busybox
2 CMD echo "Hi there. My work is done. You can run them from my Docker image."
3 CMD echo "chulaluck chansri 653380124-4"
4
```

Lab Worksheet

```

Terminal
PS D:\Lab8_3> docker build -t chulaluck/lab8 .
[+] Building 0.1s (5/5) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 181B
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 2)
=> WARN: MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the same stage because only the last one will b
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 3)
=> [internal] load metadata for docker.io/library/busybox:latest
=> [internal] load .dockerignore
=> => transferring context: 2B
=> CACHED [1/1] FROM docker.io/library/busybox:latest
=> exporting to image
=> => exporting layers
=> => writing image sha256:dd11c479d6a1ac8b271c09c8121c20b50f366f562eef46398e58fc7215fb0d73
=> => naming to docker.io/chulaluck/lab8
View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/wc50b5h102xjz2flowqzi9at6

```

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

```

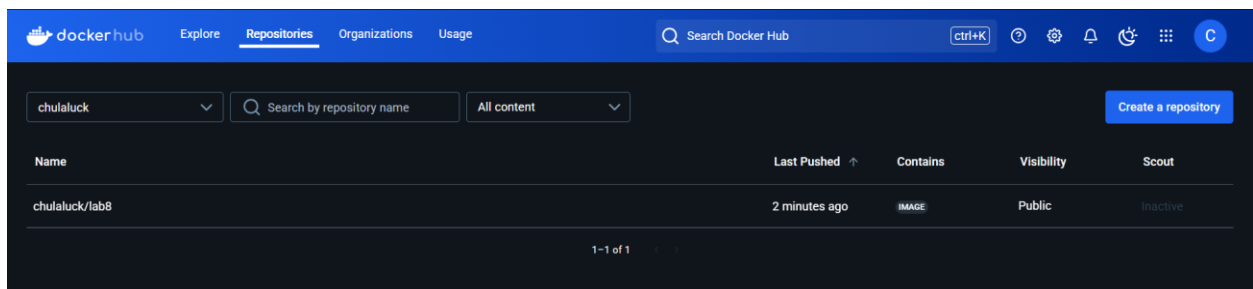
PS D:\Lab8_3> docker run chulaluck/lab8
"chulaluck chansri 653380124-4"
PS D:\Lab8_3> docker push chulaluck/lab8
Using default tag: latest
The push refers to repository [docker.io/chulaluck/lab8]
59654b79daad: Mounted from library/busybox
latest: digest: sha256:0ce35574d487b8816a0b433ecc672ad372fd41b4f152628731b6ca525551280f size: 527
PS D:\Lab8_3>

```

\$ 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)

Lab Worksheet

แบบฝึกปฏิบัติที่ 8.4: การ Build แอปพลิเคชันจาก Container image และการ Update แอปพลิเคชัน

1. เปิด Command line หรือ Terminal จากนั้นสร้าง Directory ชื่อ Lab8_4
2. ทำการ Clone ซอร์สโค้ดของเว็บแอปพลิเคชันจาก GitHub repository
<https://github.com/docker/getting-started.git> ลงใน Directory ที่สร้างขึ้น โดยใช้คำสั่ง
\$ git clone https://github.com/docker/getting-started.git
3. เปิดดูองค์ประกอบภายใน getting-started/app เมื่อพบไฟล์ package.json ให้ใช้ Text editor ในการเปิดอ่าน

```
PS D:\Lab8_3> cd D:\Lab8_4
PS 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 | 7.85 MiB/s, done.
Resolving deltas: 100% (523/523), done.
PS D:\Lab8_4>
```

```
package.json X
getting-started > app > 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 }
```


Lab Worksheet

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

- 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
- ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้ โดยกำหนดใช้ชื่อ image เป็น myapp_รหัสสนศ. ไม่มีขีด
 \$ docker build -t <myapp_รหัสสนศ. ไม่มีขีด> .

```

Terminal
PS D:\Lab8_4\getting-started\app> docker build -t my_app6533801244 .
[+] Building 25.9s (10/10) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 156B
=> [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@sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d975e9059dd066be3e274fbb25
=> => resolve docker.io/library/node:18-alpine@sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d975e9059dd066be3e274fbb25
=> => sha256:dcfb7b337595be6f4d214e4eed84f230ee0e0e4ac03a50380d573e289b9e5e40 6.18kB / 6.18kB
=> => sha256:1f3e46996e2966e4faa5846e56e3748b7315e2ded61476c24403d592134f0 3.64MB / 3.64MB
=> => sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e0da02525e63167c 40.01MB / 40.01MB
=> => sha256:5659d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15bf426d01bc1 1.26MB / 1.26MB
=> => sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d975e9059dd066be3e274fbb25 7.67kB / 7.67kB
=> => sha256:6e804119c3884fc5782795bf0d2adc89201c63105aece8647b17a7bcebbbc385e 1.72kB / 1.72kB
=> => extracting sha256:1f3e46996e2966e4faa5846e56e3748b7315e2ded61476c24403d592134f0
=> => sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce368990ca771 444B / 444B
=> => extracting sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e0da02525e63167c
=> => extracting sha256:5659d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15bf426d01bc1
=> => extracting sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce368990ca771
=> [internal] load build context
=> => transferring context: 4.82MB
=> [2/4] WORKDIR /app
=> [3/4] COPY . .
=> [4/4] RUN yarn install --production
=> exporting to image
=> => exporting layers
=> => writing image sha256:f6aa178b757b2b0b1f63fe4a58d226e0611ecc081bfde109bc1be88af972081
=> => naming to docker.io/library/my_app6533801244
View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/qorhbqddvys3mn5tao9ka61f0
What's next:
  View a summary of image vulnerabilities and recommendations ->docker scout quickview
PS D:\Lab8_4\getting-started\app>

```

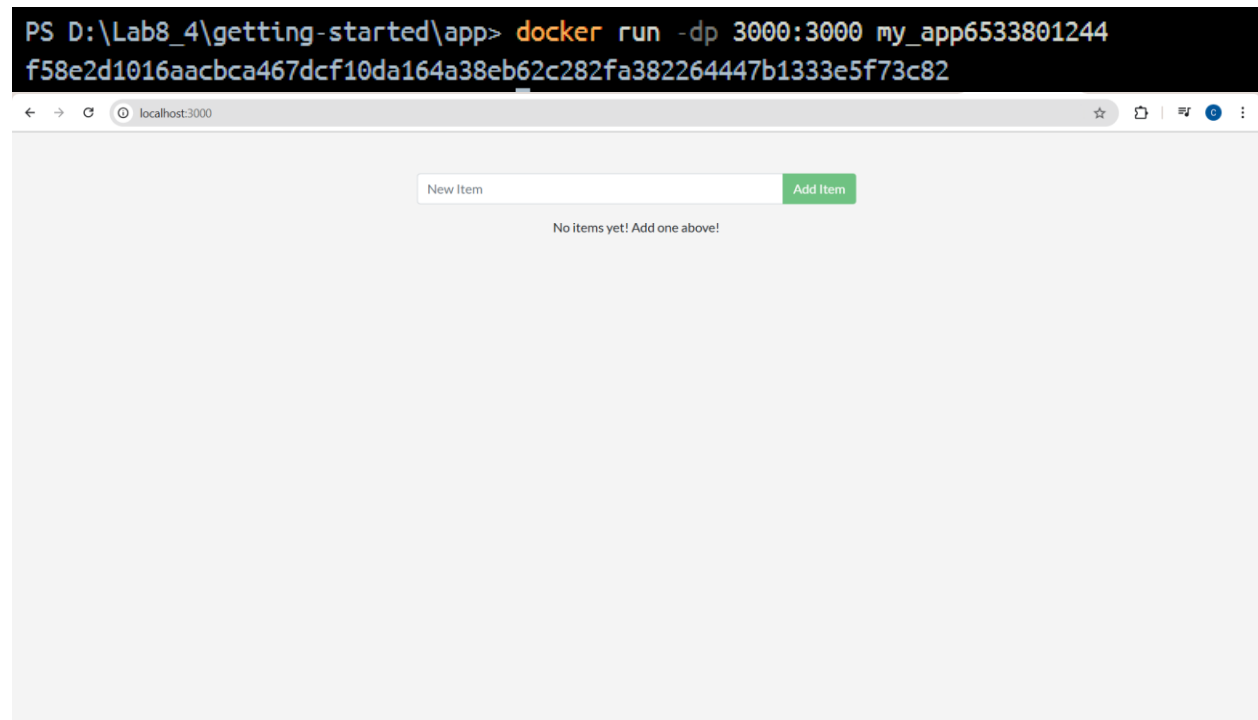
Lab Worksheet

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

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

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

Lab Worksheet

```

JS app.js x
D:\Lab8_4> getting-started > app > src > static > js > .js app.js > TodoListCard > items.map() callback
14 function TodoListCard() {
30   const onItemUpdate = React.useCallback(
40   );
41
42   const onItemRemoval = React.useCallback(
43     item => {
44       const index = items.findIndex(i => i.id === item.id);
45       setItems([...items.slice(0, index), ...items.slice(index + 1)]);
46     },
47     [items],
48   );
49
50   if (items === null) return 'Loading...';
51
52   return (
53     <React.Fragment>
54       <AddItemForm onNewItem={onNewItem} />
55       {items.length === 0 && (
56         <p className="text-center">There is no TODO item. Please add one to the list. By chulaluck chansri</p>
57       )}
58       {items.map(item => (
59         <ItemDisplay
60           item={item}
61           key={item.id}
62           onItemUpdate={onItemUpdate}
63           onItemRemoval={onItemRemoval}
64         />
65       ))}
66     </React.Fragment>
67   );
68 }

```

9. ทำการ Build Docker image โดยใช้คำสั่งเดียวกันกับข้อ 5

```

PS D:\Lab8_4\getting-started\app> docker build -t my_app6533801244 .
[+] Building 17.9s (10/10) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 156B
=> [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@sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d975e9059dd066be3e274fbb25
=> [internal] load build context
=> => transferring context: 8.13kB
=> CACHED [2/4] WORKDIR /app
=> [3/4] COPY . .
=> [4/4] RUN yarn install --production
=> exporting to image
=> => exporting layers
=> => writing image sha256:f086557534da6f4c2c03f7e9727c7932c1cbb0e0752d93ee4629f8b4b8d83847
=> => naming to docker.io/library/my_app6533801244
View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/3t8wg4s5fhkxfvknkgtdln8jzq

```

10. Start และรัน Container ตัวใหม่ โดยใช้คำสั่งเดียวกันกับข้อ 6

```

PS D:\Lab8_4\getting-started\app> docker run -dp 3000:3000 my_app6533801244
b6923094d0ecac6b76efad6bf7d99f4f51982efb7bce8b4cc99b795e539e1628
docker: Error response from daemon: driver failed programming external connectivity on endpoint bold_rubin (a58d66fe09c32ba7b15f11a4d0fa98e474942a16f776c48572878d30efad01e7): Bind for 0.0.0.0:3000 failed: port is already allocated.
PS D:\Lab8_4\getting-started\app>

```

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

(1) Error ที่เกิดขึ้นหมายความว่าอย่างไร และเกิดขึ้นเพราะอะไร

port 3000 บนเครื่องถูกใช้งานอยู่แล้ว ทำให้ Docker ไม่สามารถ bind พอร์ตดังกล่าวให้กับ container ใหม่ได้ -dp 3000:3000 Docker จะพยายามแมปพอร์ต บนเครื่องโฮสต์ (host machine) ไปยัง พอร์ต

Lab Worksheet

ภายใน container ถ้าพอร์ตถูกใช้ไปแล้ว Docker จะไม่สามารถทำการ mapping ได้และจะเกิด error ข้างต้น

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

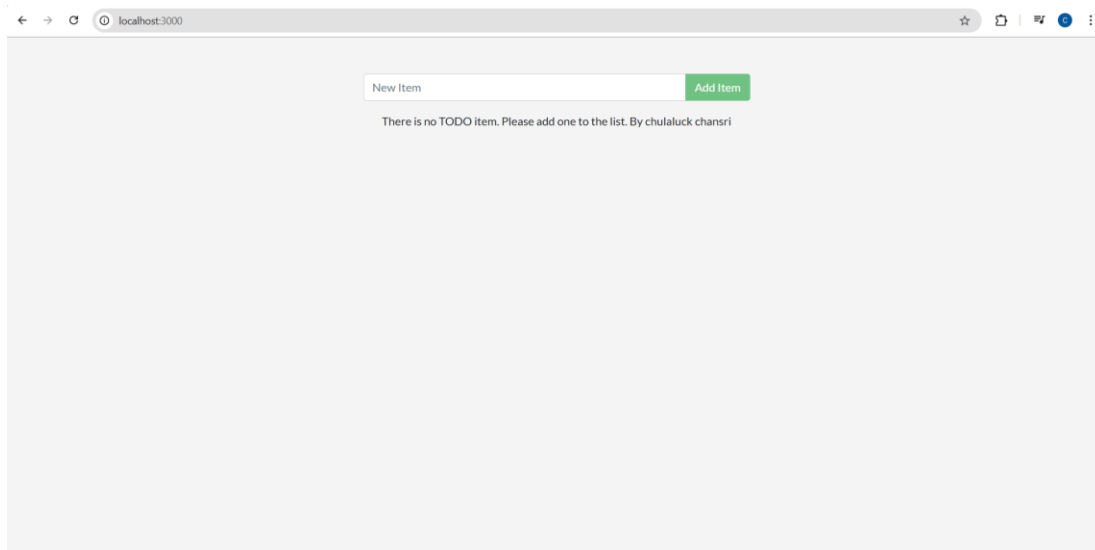
b. ผ่าน Docker desktop

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

12. Start และรัน Container ตัวใหม่อีกครั้ง โดยใช้คำสั่งเดียวกันกับข้อ 6

```
PS D:\Lab8_4\getting-started\app> docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                   NAMES
f58e2d1016aa   f6aa178b757b   "docker-entrypoint.s..." 9 minutes ago  Up 8 minutes  0.0.0.0:3000->3000/tcp   crazy_shaw
PS D:\Lab8_4\getting-started\app> docker stop f58e2d1016aa
f58e2d1016aa
PS D:\Lab8_4\getting-started\app> docker rm f58e2d1016aa
f58e2d1016aa
PS D:\Lab8_4\getting-started\app> docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                   NAMES
PS D:\Lab8_4\getting-started\app> docker run -dp 3000:3000 my_app6533801244
90fb942773d50752f7bb4eb0c9e9dec03085655ef906eff2a2f2e4aa4d30d474
PS D:\Lab8_4\getting-started\app>
```

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

```

2025-01-27 18:01:48.006+0000 [id=52] INFO jenkins.InitReactorRunner$1#onAttained: Configuration for all jobs updated
2025-01-27 18:01:48.022+0000 [id=68] INFO hudson.util.Retrier#start: Attempt #1 to do the action check updates server
2025-01-27 18:01:48.352+0000 [id=43] 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:

a772a8636a7d42fc888f4cdca7e5305d

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

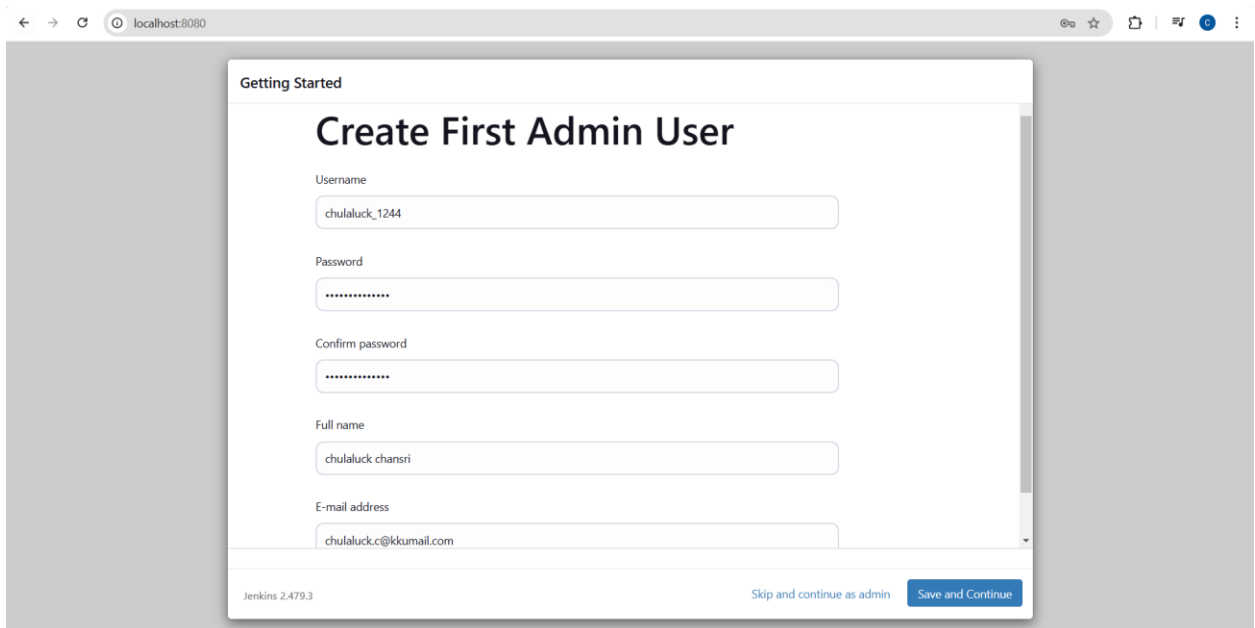
*****
*****

```

[Check point#12] Capture หน้าจอที่แสดงผล Admin password

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

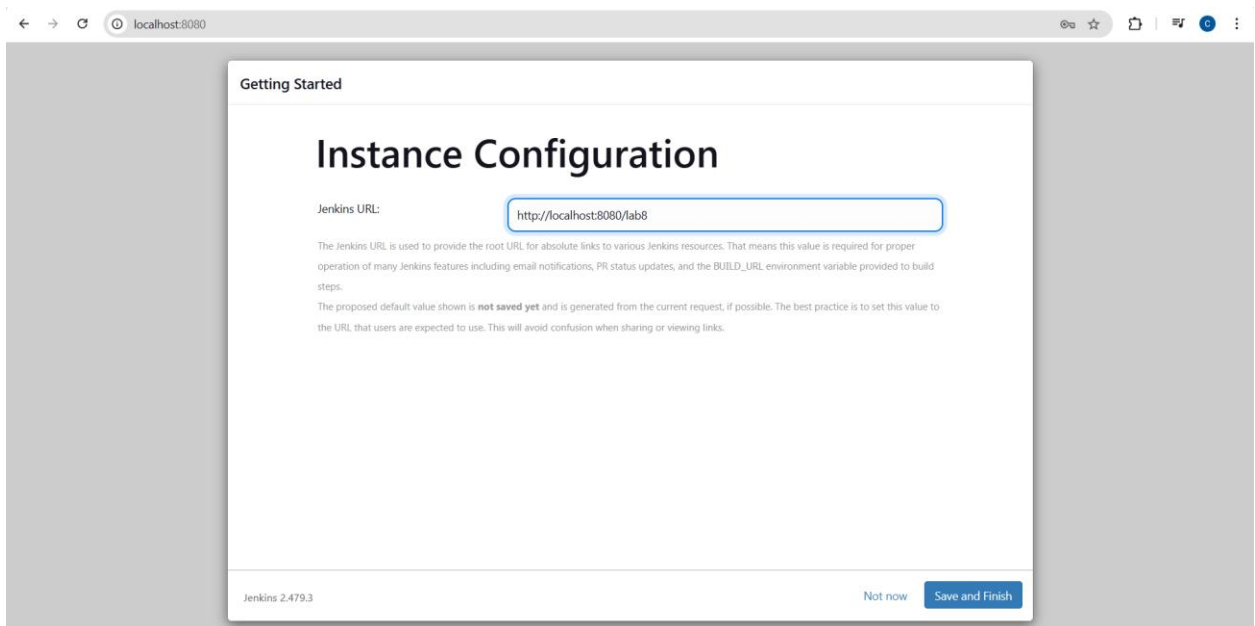
Lab Worksheet



The screenshot shows the 'Getting Started' page of Jenkins 2.479.3. The main heading is 'Create First Admin User'. The form contains the following fields: Username (chulaluck_1244), Password (masked with dots), Confirm password (masked with dots), Full name (chulaluck chansri), and E-mail address (chulaluck.c@kkumail.com). At the bottom, there are two buttons: 'Skip and continue as admin' and 'Save and Continue'.

[Check point#13] Capture หน้าจอที่แสดงผลการตั้งค่า

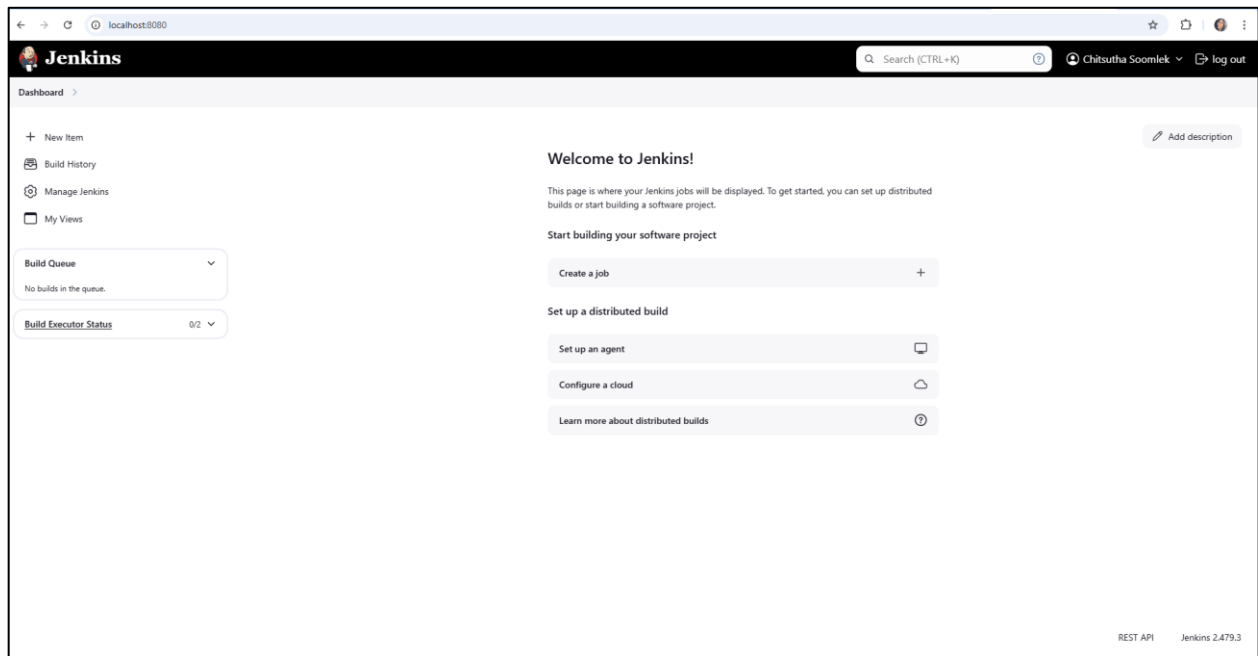
7. กำหนด Jenkins URL เป็น <http://localhost:8080/lab8>



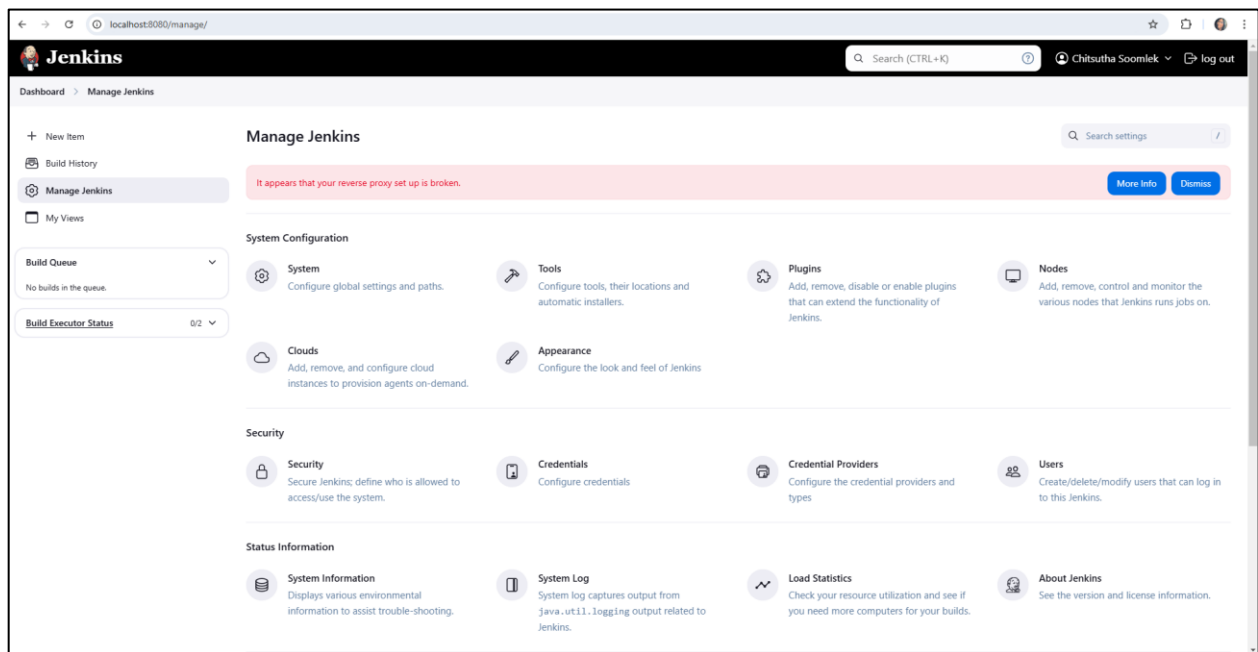
The screenshot shows the 'Getting Started' page of Jenkins 2.479.3. The main heading is 'Instance Configuration'. The form contains the following fields: Jenkins URL (http://localhost:8080/lab8). Below the field, there is a note: 'The Jenkins URL is used to provide the root URL for absolute links to 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.' At the bottom, there are two buttons: 'Not now' and 'Save and Finish'.

8. เมื่อติดตั้งเรียบร้อยแล้วจะพบกันหน้า Dashboard ดังแสดงในภาพ

Lab Worksheet

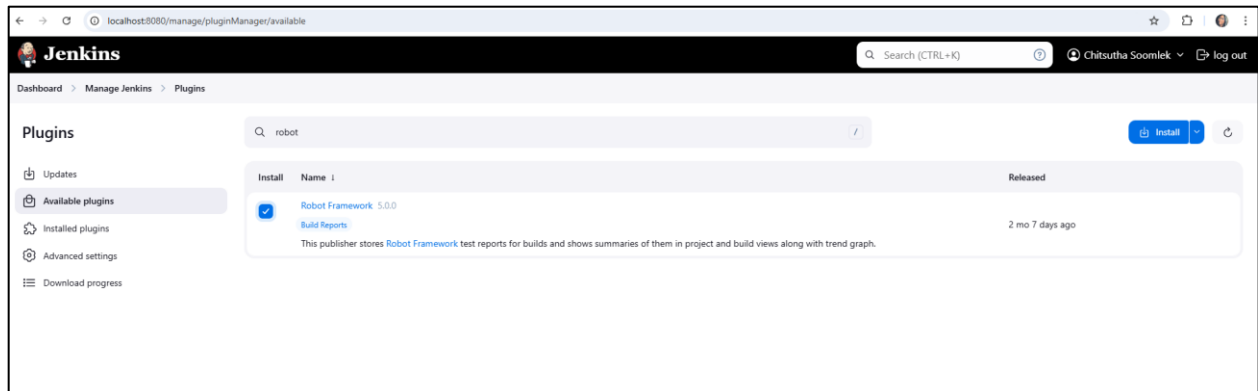


9. เลือก Manage Jenkins แล้วไปที่เมนู Plugins

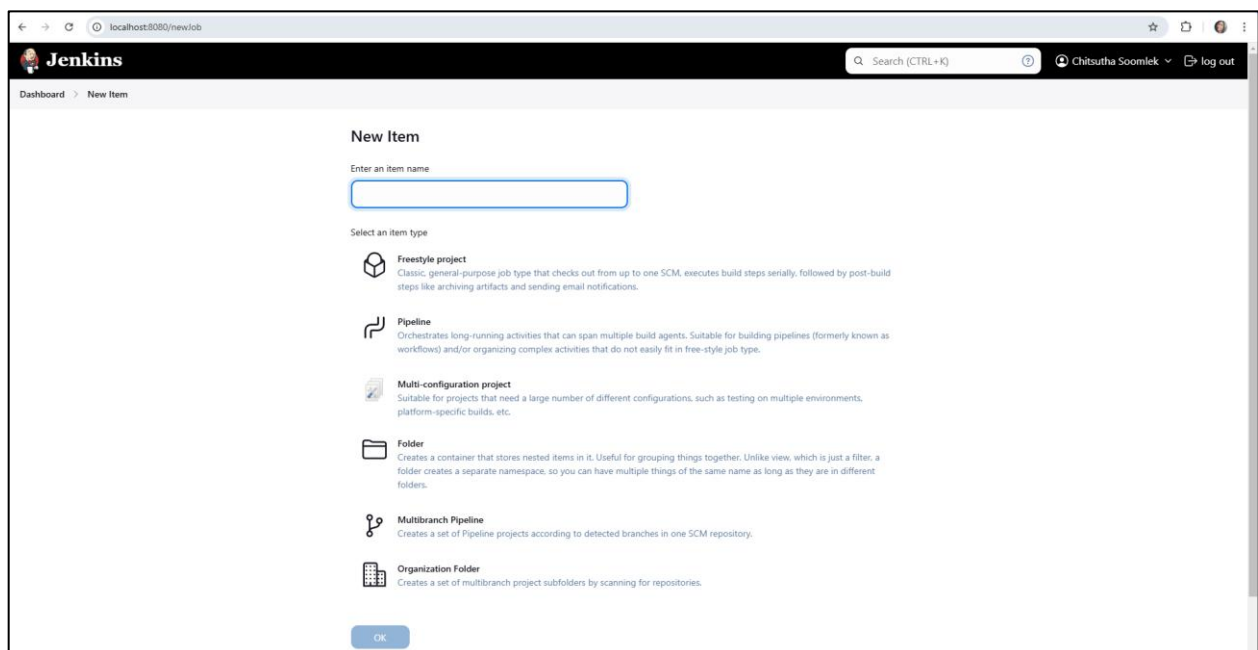


Lab Worksheet

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 ไปถึงไฟล์ให้เรียบร้อยแล้ว)

The screenshot shows the Jenkins Configuration page for a job named 'UAT'. The 'General' tab is selected, and the 'Enabled' toggle is turned on. The 'Description' field contains 'Lab 8.5'. The 'GitHub project' checkbox is checked, and the 'Project url' is set to 'https://github.com/chulaluckkkk/lab8.git'. The 'Save' button is highlighted. Below this, the 'Source Code Management' tab is selected, showing 'None' as the selected option. The 'Build Triggers' section has 'Build periodically' checked, with a schedule of 'H/15 * * * *'. The 'Save' button is also highlighted here.

Jenkins Configuration - General

Dashboard > UAT > Configuration

Configure

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

General Enabled

Description: Lab 8.5

Plain text [Preview](#)

☐ Discard old builds ?

☒ GitHub project

Project url ?
<https://github.com/chulaluckkkk/lab8.git>

Advanced ▾

☐ This project is parameterized ?

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

Jenkins Configuration - Source Code Management

Dashboard > UAT > Configuration

Configure

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

Source Code Management

☒ None

☐ Git ?

Build Triggers

☐ Trigger builds remotely (e.g., from scripts) ?

☐ Build after other projects are built ?

☒ Build periodically ?

Provides a [cron](#) -like feature to periodically execute this project. This feature is primarily for using Jenkins as a cron replacement, and it is **not ideal for continuously building software projects**. When people first start continuous integration, they are often so used to the idea of regularly scheduled builds like nightly/weekly that they use this feature. However, the point of continuous integration is to start a build as soon as a change is made, to provide a quick feedback to the change. To do that you need to [hook up SCM change notification to Jenkins](#).

So, before using this feature, stop and ask yourself if this is really what you want.

Schedule ?
H/15 * * * *

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

Lab Worksheet

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 resource.robot

Advanced ▾

Add build step ▾

Post-build Actions

Save Apply

[Check point#14] Capture หน้าจอแสดงการตั้งค่า พร้อมกับตอบคำถามต่อไปนี้

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

robot resource.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

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)

output.xml

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

Lab Worksheet

Dashboard > UAT > #2

Status

</> Changes

Console Output

Edit Build Information

Delete build '#2'

Timings

Previous Build

#2 (Jan 27, 2025, 6:43:01 PM)

Started by user [chulaluck chansri](#)

This run spent:

- 9 ms waiting;
- 18 ms build duration;
- 27 ms total from scheduled to completion.

No changes.

Add description

Keep this build forever

Started 7.9 sec ago
Took 18 ms

REST API Jenkins 2.479.3

Status

</> Changes

Console Output

Edit Build Information

Delete build '#4'

Timings

Previous Build

Console Output

Started by timer

Running as SYSTEM

Building in workspace /var/jenkins_home/workspace/UAT

[UAT] \$ /bin/sh -xe /tmp/jenkins14642686133550903216.sh

+ robot resource.robot

/tmp/jenkins14642686133550903216.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:

ERROR: Build step failed with exception

/var/jenkins_home/workspace/UAT/output.xml does not exist.

```

at org.apache.tools.ant.types.AbstractFileSet.getDirectoryScanner(AbstractFileSet.java:512)
at org.apache.tools.ant.types.AbstractFileSet.getDirectoryScanner(AbstractFileSet.java:489)
at PluginClassLoader for robot//hudson.plugins.robot.RobotParser$RobotParserCallable.invoke(RobotParser.java:76)
at PluginClassLoader for robot//hudson.plugins.robot.RobotParser$RobotParserCallable.invoke(RobotParser.java:52)
at hudson.FilePath.act(FilePath.java:1234)
at hudson.FilePath.act(FilePath.java:1217)
at PluginClassLoader for robot//hudson.plugins.robot.RobotParser.parse(RobotParser.java:48)
at PluginClassLoader for robot//hudson.plugins.robot.RobotPublisher.parse(RobotPublisher.java:262)
at PluginClassLoader for robot//hudson.plugins.robot.RobotPublisher.perform(RobotPublisher.java:286)
at hudson.tasks.BuildStepCompatibilityLayer.perform(BuildStepCompatibilityLayer.java:80)
at hudson.tasks.BuildStepMonitor$1.perform(BuildStepMonitor.java:20)
at hudson.model.AbstractBuild$AbstractBuildExecution.perform(AbstractBuild.java:818)

```

Download

Copy

View as plain text

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