ชื่อ-นามสกุล นราธิป สังฆโสภณ รหัสนักศึกษา 653380329-6 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

```
C:\Users\KKU650001>docker login
Authenticating with existing credentials...
Login Succeeded

C:\Users\KKU650001>cd Lab8_1

C:\Users\KKU650001\Lab8_1>docker pull busybox
Using default tag: latest
latest: Pulling from library/busybox
Digest: sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138bd458fc8257fbf1
Status: Image is up to date for busybox:latest
docker.io/library/busybox:latest

C:\Users\KKU650001\Lab8_1>docker images
REPOSITORY              TAG       IMAGE ID       CREATED        SIZE
busybox                 latest    a5d0ce49aa80   3 months ago   6.56MB
synthesizedio/whalesay  latest    205c30ad19b1   6 months ago   63.9MB
```

(1) สิ่งที่อยู่ภายใต้คอลัมน์ Repository คืออะไร

```
C:\Users\KKU650001\Lab8_1>docker images
REPOSITORY              TAG       IMAGE ID       CREATED        SIZE
busybox                 latest    a5d0ce49aa80   3 months ago   6.56MB
```

REPOSITORY TAG IMAGE ID CREATED SIZE

busybox latest a5d0ce49aa80 3 months ago 6.56MB

(2) Tag ที่ใช้บ่งบอกถึงอะไร version ล่าสุด

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

```
C:\Users\KKU650001\Lab8_1>docker images
REPOSITORY               TAG       IMAGE ID       CREATED         SIZE
busybox                  latest    a5d0ce49aa80   3 months ago    6.56MB
synthesizedio/whalesay   latest    205c30ad19b1   6 months ago    63.9MB

C:\Users\KKU650001\Lab8_1>docker run busybox

C:\Users\KKU650001\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 23 02:35 .
drwxr-xr-x    1 root     root          4096 Jan 23 02:35 ..
-rwxr-xr-x    1 root     root             0 Jan 23 02:35 .dockerenv
drwxr-xr-x    2 root     root         12288 Sep 26 21:31 bin
drwxr-xr-x    5 root     root           360 Jan 23 02:35 dev
drwxr-xr-x    1 root     root          4096 Jan 23 02:35 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  237 root     root             0 Jan 23 02:35 proc
drwx------    1 root     root          4096 Jan 23 02:35 root
dr-xr-xr-x   11 root     root             0 Jan 23 02:35 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
```

(1) เมื่อใช้ option -it ในคำสั่ง run ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป
ทำให้ container เปิด interactive terminal ที่ทำให้สามารถ พิมพ์คำสั่งกับ container ได้โดยตรง

(2) คอลัมน์ STATUS จากการรันคำสั่ง docker ps -a แสดงถึงข้อมูลอะไร
แสดงสถานะปัจจุบันของ container

12. ป้อนคำสั่ง $ docker rm <container ID ที่ต้องการลบ>

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

```
C:\Users\KKU650001\Lab8_1>docker ps -a
CONTAINER ID   IMAGE                              COMMAND                  CREATED          STATUS
 PORTS     NAMES
cdce9bfa0894   busybox                            "echo 'Hello narathi…"   16 seconds ago   Exited (0) 16 se
          intelligent_antonelli
feda06012878   busybox                            "sh"                     6 minutes ago    Exited (0) About
          recursing_fermi
0cfd5d16a96a   busybox                            "sh"                     6 minutes ago    Exited (0) 6 mir
          peaceful_brattain
50d94b383a12   synthesizedio/whalesay:latest      "/usr/local/bin/cows…"   28 minutes ago   Exited (0) 28 mi
          modest_curran

C:\Users\KKU650001\Lab8_1>docker rm 0cfd5d16a96a
0cfd5d16a96a

C:\Users\KKU650001\Lab8_1>ps -a
'ps' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\KKU650001\Lab8_1>=docker ps -a
CONTAINER ID   IMAGE                              COMMAND                  CREATED          STATUS
          PORTS     NAMES
cdce9bfa0894   busybox                            "echo 'Hello narathi…"   8 minutes ago    Exited (0) 8 mir
      intelligent_antonelli
feda06012878   busybox                            "sh"                     13 minutes ago   Exited (0) 9 mir
      recursing_fermi
50d94b383a12   synthesizedio/whalesay:latest      "/usr/local/bin/cows…"   35 minutes ago   Exited (0) 35 mi
      modest_curran
```

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”

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

```
C:\Users\KKU650001\Lab8_2>docker build -t my_image .
[+] Building 3.6s (6/6) FINISHED                                                                    docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                                0.1s
 => => transferring dockerfile: 162B                                                                                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   3.2s
 => => 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                                                                                              0.1s
 => => exporting layers                                                                                             0.0s
 => => exporting manifest sha256:f265591b28481058b31ff5f6f01782ec12a6fd962eb22387bf95b31a8122c664                   0.0s
 => => exporting config sha256:9f750671d555930422403a5fe786a5325a49bd013f695d924b13d64aef0512aa                     0.0s
 => => exporting attestation manifest sha256:6123817078a05b2cf72e8b563079ea5a0f804363db5a766fd05c17ffb55ab2fe       0.0s
 => => exporting manifest list sha256:ba43bf650e1f7d115c58bd703fc37d665f6b88f068aa6749903dd73833dd8444              0.0s
 => => naming to docker.io/library/my_image:latest                                                                  0.0s
 => => unpacking to docker.io/library/my_image:latest                                                               0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/2la0e5coab9v6ctpe24ja7v7b

C:\Users\KKU650001\Lab8_2>docker images
REPOSITORY                TAG       IMAGE ID       CREATED         SIZE
my_image                  latest    ba43bf650e1f   3 months ago    6.56MB
busybox                   latest    a5d0ce49aa80   3 months ago    6.56MB
synthesizedio/whalesay    latest    205c30ad19b1   6 months ago    63.9MB
```

(1) คำสั่งที่ใช้ในการ run คือ

docker run my_image

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

-t ใช้สำหรับกำหนด ชื่อ ให้กับ Docker image ที่ถูกสร้างขึ้น

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”
```

```
$ 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

```
C:\Users\KKU650001\Lab8_3>docker build -t narathip345/lab8_3 .
[+] Building 0.5s (5/5) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 182B
 => [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:a5d0ce49aa801d475da48f8cb
 => => resolve docker.io/library/busybox:latest@sha256:a5d0ce49aa801d475da48f8cb163c354
 => exporting to image
 => => exporting layers
 => => exporting manifest sha256:ac122bb70a897a672269bb5e5b9a317ea8a550350a43b3e82406fb
 => => exporting config sha256:5aa1abce8578250ed2831eb7a31ab5632ab914e3ea5953ca542d0795
 => => exporting attestation manifest sha256:009fdda2810fc478019970fb2077b384325d61423c
 => => exporting manifest list sha256:0c8e790d2e4a0d4097ec5c6eee97a05ddc748049f7a8d5c43
 => => naming to docker.io/narathip345/lab8_3:latest
 => => unpacking to docker.io/narathip345/lab8_3:latest

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/4izg9r

 3 warnings found (use docker --debug to expand):
 - MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the
 used (line 2)
 - JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavi
 - JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavi

C:\Users\KKU650001\Lab8_3>docker run narathip345/lab8_3
“narathip sungkasopon 653380329-6”
```

6. ทำการ Push ตัว Docker image ไปไว้บน Docker Hub โดยการใช้คำสั่ง

   $ docker push <username ที่ลงทะเบียนกับ Docker Hub>/lab8

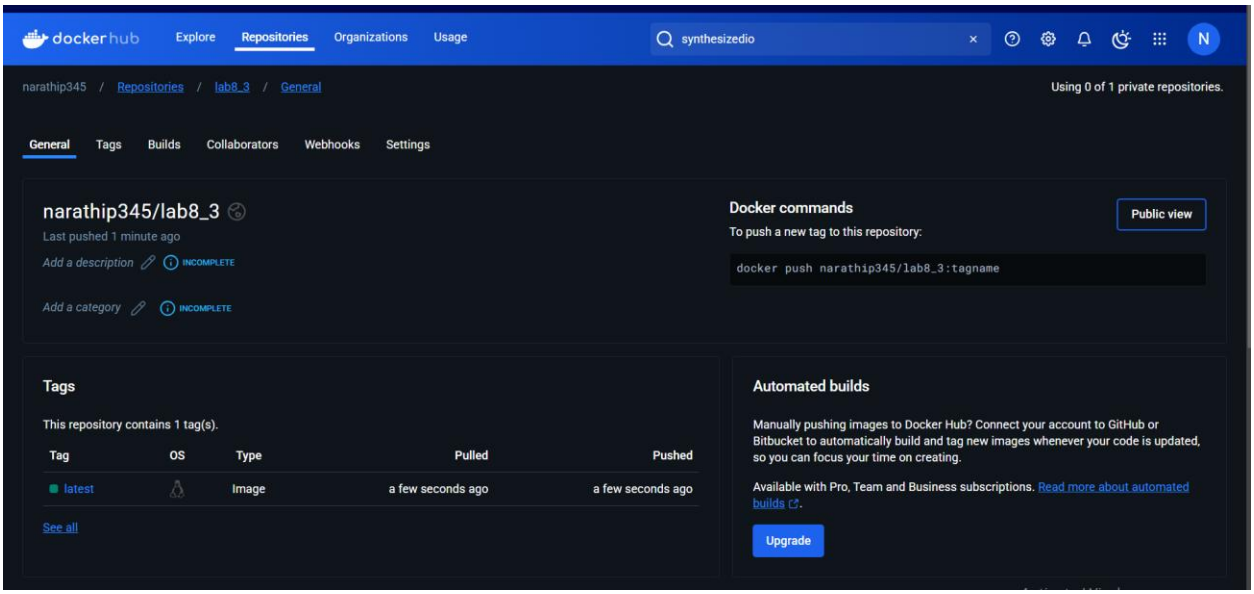
   ในกรณีที่ติดปัญหาไม่ได้ Login ไว้ก่อน ให้ใช้คำสั่งต่อไปนี้ เพื่อ Login ก่อนทำการ Push

   $ 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

เปิดดูองค์ประกอบภายใน getting-started/app เมื่อพบไฟล์ package.json ให้ใช้ Text editor ในการเปิดอ่าน

[Check point#7] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงที่อยู่ของ Source code ที่ Clone มาและเนื้อหาของไฟล์ p\ackage.jso

```
C:\Users\KKU650001\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 | 5.30 MiB/s, done.
Resolving deltas: 100% (523/523), done.

C:\Users\KKU650001\Lab8_4>
```

Users > KKU650001 > Lab8_4 > getting-started > app > {} package.json > ...

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

3. ภายใต้ getting-started/app ให้สร้าง Dockerfile พร้อมกับใส่เนื้อหาดังต่อไปนี้ลงไปในไฟล์

   FROM node:18-alpine

   WORKDIR /app

   COPY . .

   RUN yarn install --production

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

   EXPOSE 3000

4. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้ โดยกำหนดใช้ชื่อ image เป็น myapp_รหัสนศ. ไม่มีขีด

   $ docker build -t <myapp_รหัสนศ. ไม่มีขีด> .

```
C:\Users\KKU650001>mkdir Lab8_4

C:\Users\KKU650001>cd Lab8_4

C:\Users\KKU650001\Lab8_4>clone https://github.com/docker/getting-started.git
'clone' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\KKU650001\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 | 5.30 MiB/s, done.
Resolving deltas: 100% (523/523), done.
C:\Users\KKU650001\Lab8_4>cd getting-started

C:\Users\KKU650001\Lab8_4\getting-started>cd app

C:\Users\KKU650001\Lab8_4\getting-started\app>docker build -t myapp_6533803296 .
[+] Building 29.4s (10/10) FINISHED                                    docker:desktop-linux
 => [internal] load build definition from Dockerfile                                   0.0s
 => => transferring dockerfile: 154B                                                   0.0s
 => [internal] load metadata for docker.io/library/node:18-alpine                      4.7s
 => [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@sha256:77e3b76b47148e28acc84f2e903f34  5.4s
 => => resolve docker.io/library/node:18-alpine@sha256:77e3b76b47148e28acc84f2e903f34  0.0s
 => => sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15bf426d01bc1 1.26MB  0.5s
 => => sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e0da02525e63167c 40.01MB  3.9s
```

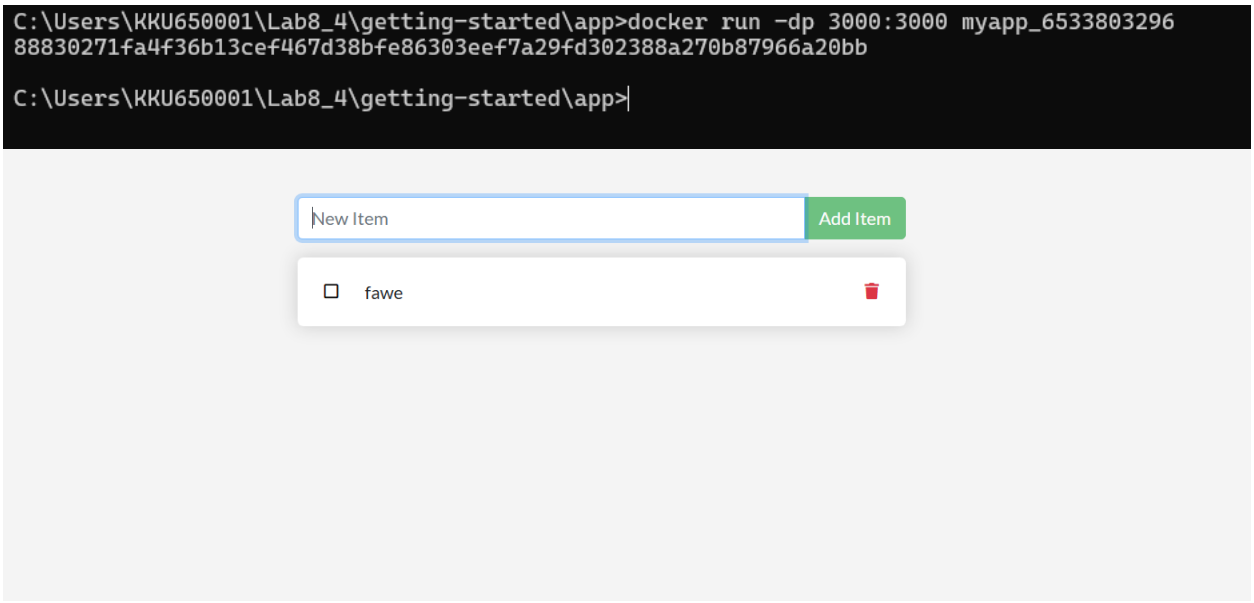
[Check point#8] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงคำสั่งและผลลัพธ์ที่ได้ทางหน้าจอ

5. ทำการ Start ตัว Container ของแอปพลิเคชันที่สร้างขึ้น โดยใช้คำสั่ง

   $ docker run -dp 3000:3000 <myapp_รหัสนศ. ไม่มีขีด>

6. 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

7. ทำการแก้ไข Source code ของ Web application ดังนี้
    a. เปิดไฟล์ src/static/js/app.js ด้วย Editor และแก้ไขบรรทัดที่ 56 จาก
    No items yet! Add one above! เป็น
    There is no TODO item. Please add one to the list. By ชื่อและนามสกุลของนักศึกษา
    b. Save ไฟล์ให้เรียบร้อย
8. ทำการ Build Docker image โดยใช้คำสั่งเดียวกันกับข้อ 5
9. Start และรัน Container ตัวใหม่ โดยใช้คำสั่งเดียวกันกับข้อ 6

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

```
C:\Users\KKU650001\Lab8_4\getting-started\app>docker build -t myapp_6533803296 .
[+] Building 28.1s (10/10) FINISHED                                                                                 docker:d
 => [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
 => [internal] load build context
 => => transferring context: 8.11kB
 => [1/4] FROM docker.io/library/node:18-alpine@sha256:77e3b76b47148e28acc84f2e903f34bedc21385b3644c9967aa25b324bb0e6b4
 => => resolve docker.io/library/node:18-alpine@sha256:77e3b76b47148e28acc84f2e903f34bedc21385b3644c9967aa25b324bb0e6b4
 => CACHED [2/4] WORKDIR /app
 => [3/4] COPY . .
 => [4/4] RUN yarn install --production
 => exporting to image
 => => exporting layers
 => => exporting manifest sha256:c70fa5be53ddaa821b86f85d2aebfaecd62dfdd0c50ec06244b203d321220234
 => => exporting config sha256:09bb93afb03d510b44ced9959fad1d75f8732df56b94c782e62e6a7e09e4e1d
 => => exporting attestation manifest sha256:a28c22ebbb014dff5eee583f2999fc420fc3651980806a18737d73efed4f39ab
 => => exporting manifest list sha256:98dc4e894998e04a43b863402f0cec6e30279c5148a66e9fb4a3a3a2976f9789
 => => naming to docker.io/library/myapp_6533803296:latest
 => => unpacking to docker.io/library/myapp_6533803296:latest

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/xlgf5h4rw4cvuqosxx3hiv2zx

C:\Users\KKU650001\Lab8_4\getting-started\app>docker run -dp 3000:3000 myapp_6533803296
ea3bc67aeef7487a93e78a987f5e42645d0d761858283cea6f5a8b36a26a2686
docker: Error response from daemon: driver failed programming external connectivity on endpoint wizardly_hawking (4fe4c3ead12d78a5b
06e98d160b768192cbe2b62cfaa0486fd5): Bind for 0.0.0.0:3000 failed: port is already allocated.
```

(1) Error ที่เกิดขึ้นหมายความอย่างไร และเกิดขึ้นเพราะอะไร

ans

พอร์ต 3000 ถูกใช้งานอยู่แล้วโดย Container หรือแอปพลิเคชันอื่นในระบบ

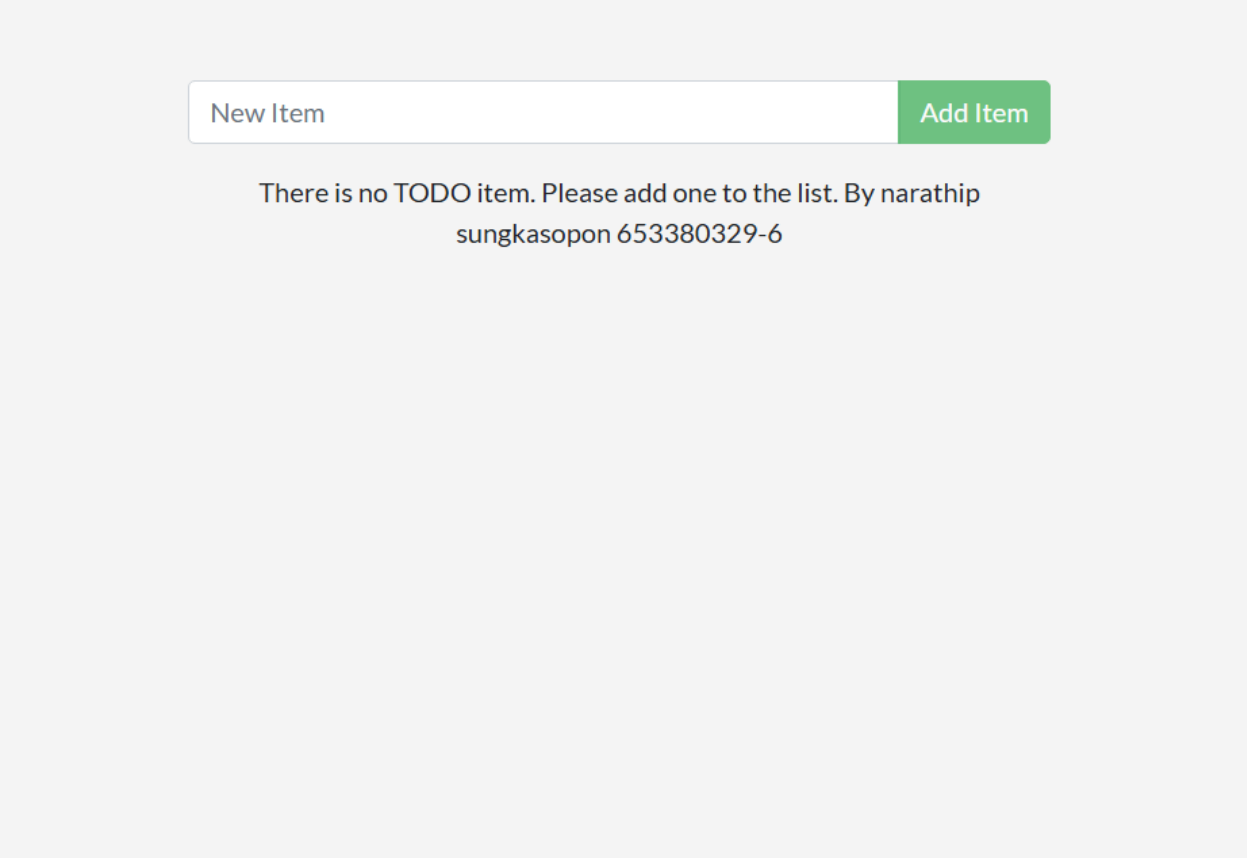
ทำให้ docker run -dp 3000:3000 ไม่สามารถเริ่มรัน Container ใหม่ได้

10. 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
11. Start และรัน Container ตัวใหม่อีกครั้ง โดยใช้คำสั่งเดียวกันกับข้อ 6

12. เปิด Browser ไปที่ URL = http://localhost:3000

[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

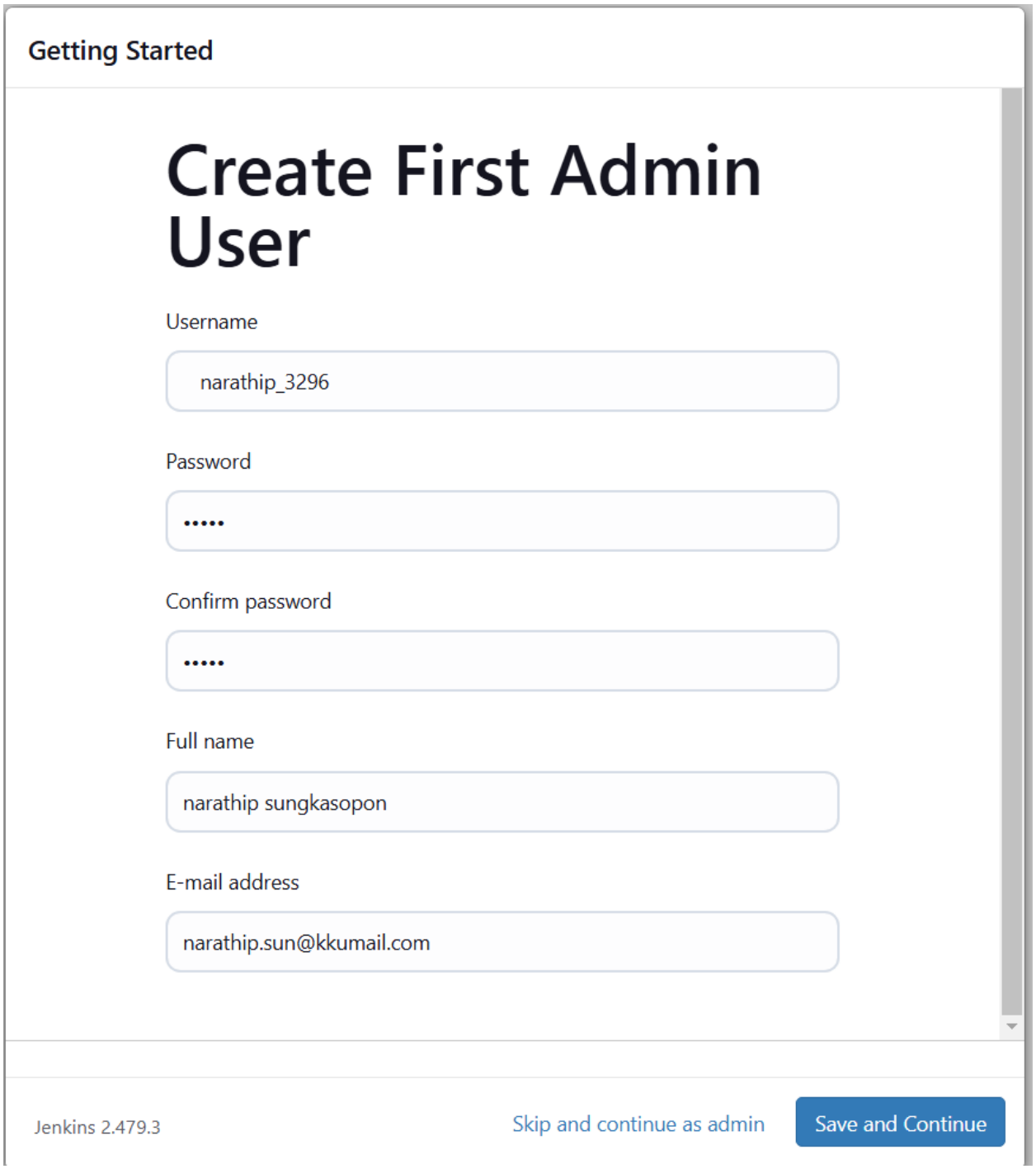
   jenkins_home:/var/jenkins_home jenkins/jenkins:lts-jdk17
3. บันทึกรหัสผ่านของ Admin user ไว้สำหรับ log-in ในครั้งแรก

   [Check point#12] Capture หน้าจอที่แสดงผล Admin password

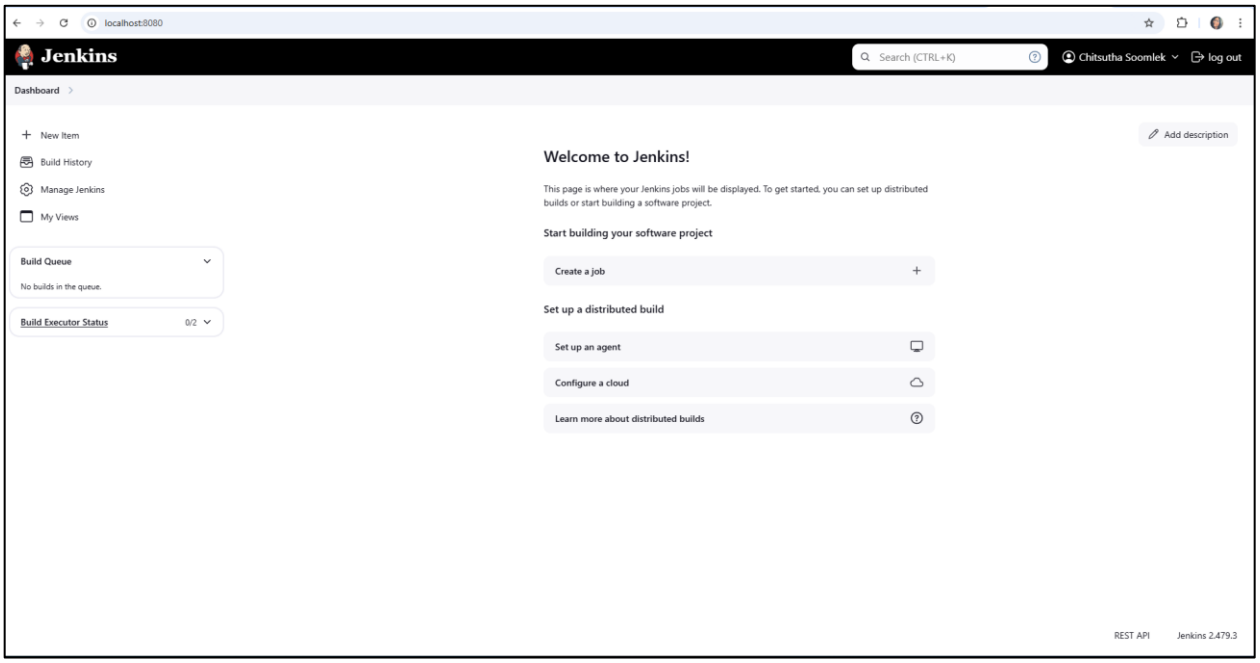
```
2025-01-29 13:17:32.888+0000 [id=44]    INFO    jenkins.InitReactorRunner$1#onAttained: Loaded all jobs
2025-01-29 13:17:32.891+0000 [id=34]    INFO    jenkins.InitReactorRunner$1#onAttained: Configuration for all j
d
2025-01-29 13:17:32.920+0000 [id=68]    INFO    hudson.util.Retrier#start: Attempt #1 to do the action check up
er
2025-01-29 13:17:33.366+0000 [id=52]    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:

13d085a5112b40e19b790e04f3dfcea3

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

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

2025-01-29 13:17:39.669+0000 [id=52]    INFO    jenkins.InitReactorRunner$1#onAttained: Completed initializatio
2025-01-29 13:17:39.693+0000 [id=25]    INFO    hudson.lifecycle.Lifecycle#onReady: Jenkins is fully up and run
2025-01-29 13:17:41.635+0000 [id=68]    INFO    h.m.DownloadService$Downloadable#load: Obtained the updated dat
 hudson.tasks.Maven.MavenInstaller
2025-01-29 13:17:41.636+0000 [id=68]    INFO    hudson.util.Retrier#start: Performed the action check updates s
essfully at the attempt #1
```

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

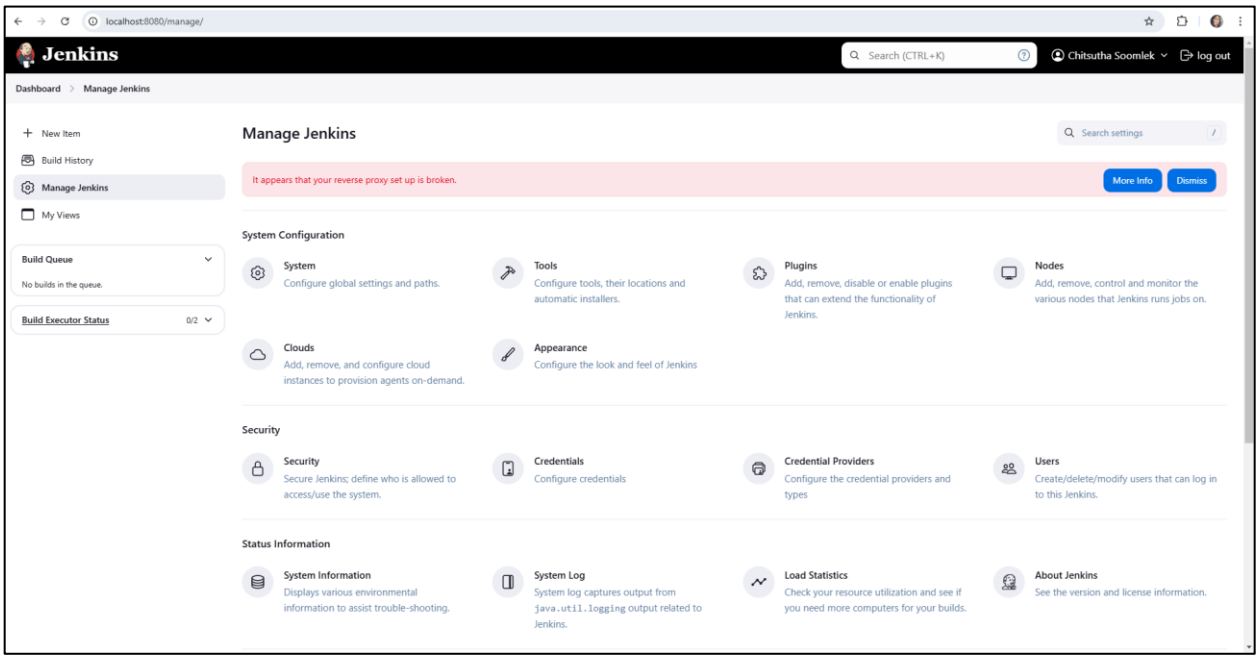
[Check point#13] Capture หน้าจอที่แสดงผลการตั้งค่า



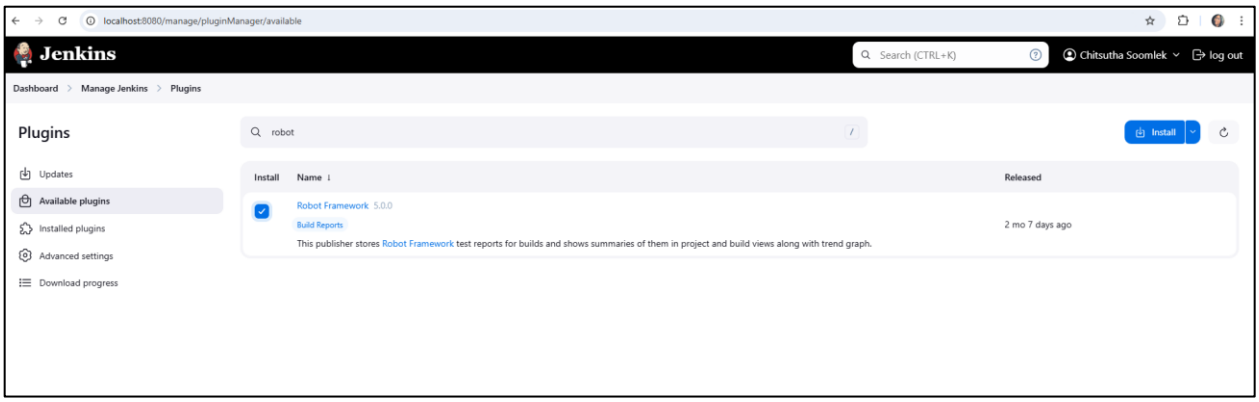
7. กำหนด Jenkins URL เป็น http://localhost:8080/lab8
8. เมื่อติดตั้งเรียบร้อยแล้วจะพบกันหน้า Dashboard ดังแสดงในภาพ



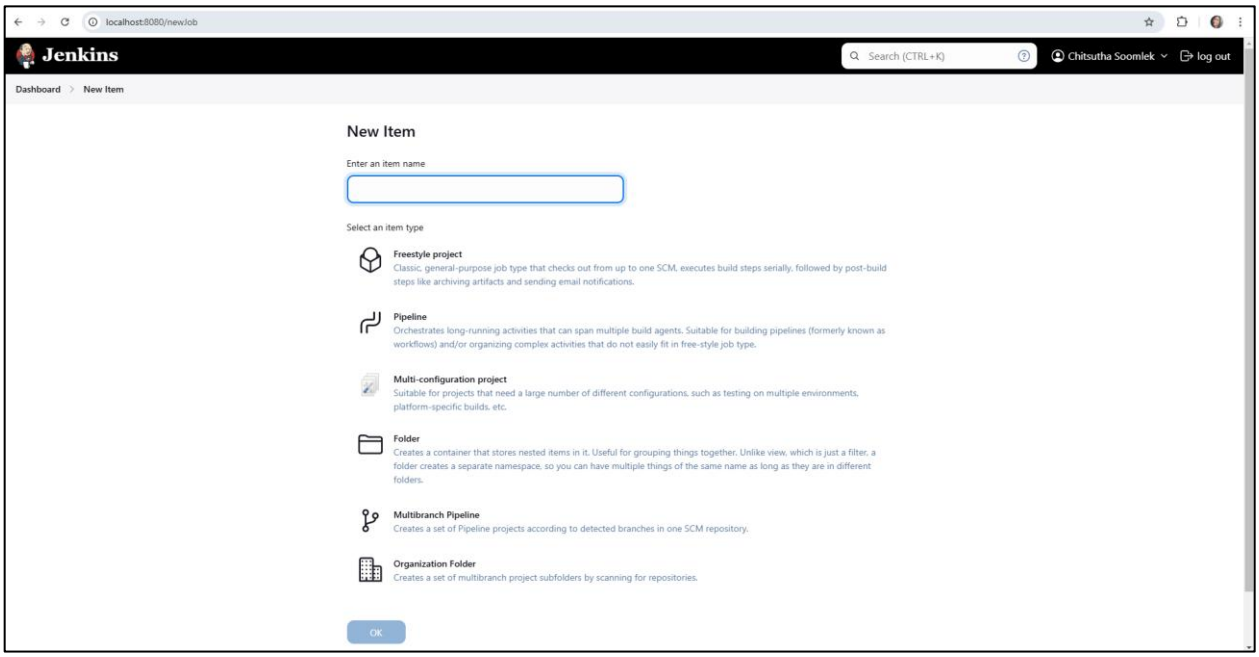
9. เลือก Manage Jenkins แล้วไปที่เมนู Plugins



10. ไปที่เมนู Available plugins แล้วเลือกติดตั้ง Robotframework เพิ่มเติม



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



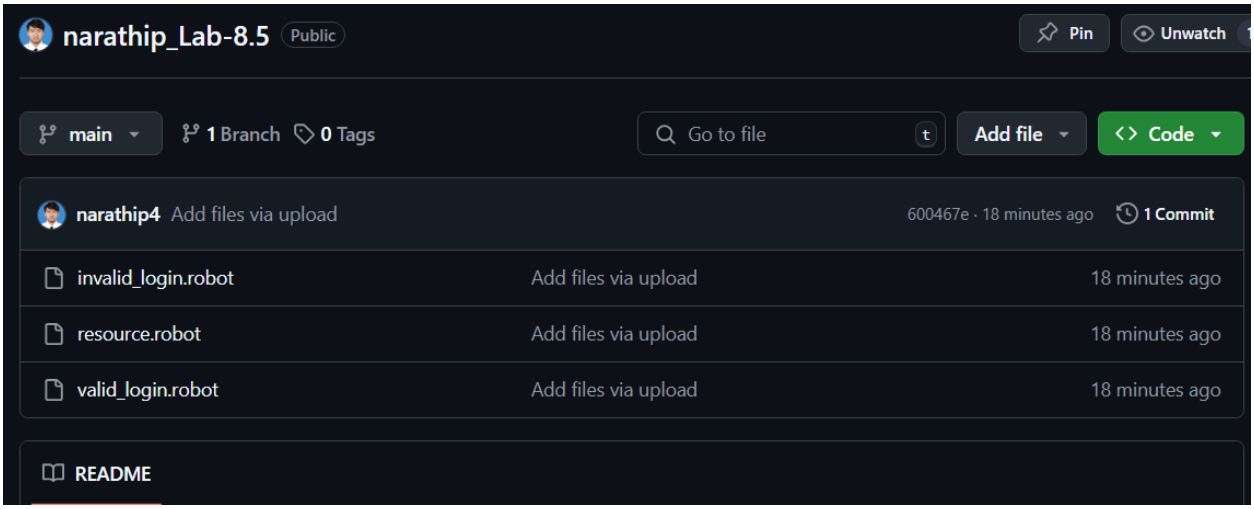
12. นำไฟล์ .robot ที่ทำให้แบบฝึกปฏิบัติที่ 7 (Lab#7) ไปไว้บน Repository ของนักศึกษา จากนั้นตั้งค่าที่จำเป็นในหน้านี้ทั้งหมด ดังนี้

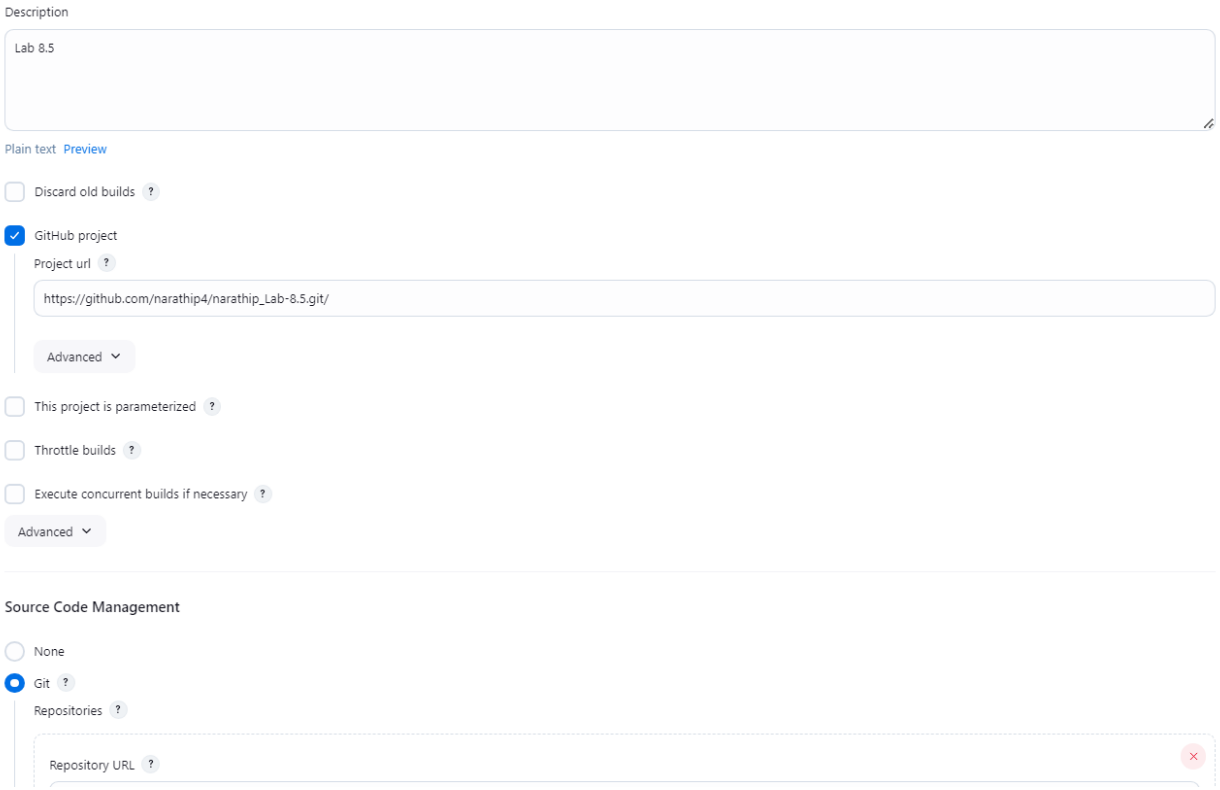
Description: Lab 8.5

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

Build Trigger: เลือกแบบ Build periodically แล้วกำหนดให้ build ทุก 15 นาที

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

ans

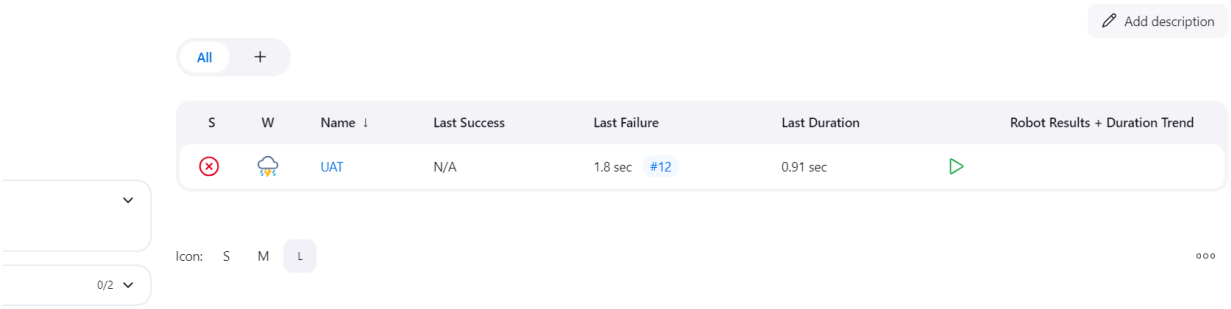
robot *.robot

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



```
Started by user narathip sungkasopon
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/narathip4/narathip_Lab-8.5.git # timeout=10
Fetching upstream changes from https://github.com/narathip4/narathip_Lab-8.5.git
 > git --version # timeout=10
 > git --version # 'git version 2.39.5'
 > git fetch --tags --force --progress -- https://github.com/narathip4/narathip_Lab-8.5.git +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision 926a1261f70bb97d386c7d6a56b789bcbf6c2293 (refs/remotes/origin/main)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f 926a1261f70bb97d386c7d6a56b789bcbf6c2293 # timeout=10
Commit message: "Add files via upload"
 > git rev-list --no-walk 926a1261f70bb97d386c7d6a56b789bcbf6c2293 # timeout=10
[UAT] $ /bin/sh -xe /tmp/jenkins4380077486263628767.sh
+ mkdir -p results
+ robot --outputdir results valid_login.robot
/tmp/jenkins4380077486263628767.sh: 4: 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:
Failed!
hudson.AbortException: No files found in path /var/jenkins_home/workspace/UAT/results with configured filemask: output.xml
                                        at PluginClassLoader for robot//hudson.plugins.robot.RobotParser$RobotParserCallable.invoke(RobotParser.java:81)
                                        at PluginClassLoader for robot//hudson.plugins.robot.RobotParser$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)
                                        at hudson.model.AbstractBuild$AbstractBuildExecution.performAllBuildSteps(AbstractBuild.java:767)
                                        at hudson.model.Build$BuildExecution.post2(Build.java:179)
                                        at hudson.model.AbstractBuild$AbstractBuildExecution.post(AbstractBuild.java:711)
                                        at hudson.model.Run.execute(Run.java:1854)
                                        at hudson.model.FreeStyleBuild.run(FreeStyleBuild.java:44)
                                        at hudson.model.ResourceController.execute(ResourceController.java:101)
                                        at hudson.model.Executor.run(Executor.java:445)
Finished: FAILURE
```