Lab Worksheet

ณัฐภัทร โพธิจันทร์ รหัสนักศึกษา 653380129-4 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

(1) สิ่งที่อยู่ภายใต้คอลัมน์ Repository คืออะไร ชื่อของ Image หรือที่เก็บ Image บน Docker Hub

(2) Tag ที่ใช้บ่งบอกถึงอะไร เวอร์ชัน ของ Image

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 6-11 พร้อมกับตอบคำถามต่อไปนี้

```
C:\Users\WINDOWS 11\Lab8_1>docker run busybox

C:\Users\WINDOWS 11\Lab8_1>docker run -it busybox sh
/ # ls
bin    dev    etc    home   lib    lib64  proc   root   sys    tmp    usr    var
/ # ls-la
sh: ls-la: not found
/ # ls -la
total 48
drwxr-xr-x    1 root     root          4096 Jan 22 03:24 .
drwxr-xr-x    1 root     root          4096 Jan 22 03:24 ..
-rwxr-xr-x    1 root     root             0 Jan 22 03:24 .dockerenv
drwxr-xr-x    2 root     root         12288 Sep 26 21:31 bin
drwxr-xr-x    5 root     root           360 Jan 22 03:24 dev
drwxr-xr-x    1 root     root          4096 Jan 22 03:24 etc
drwxr-xr-x    2 nobody   nobody        4096 Sep 26 21:31 home
drwxr-xr-x    2 root     root          4096 Sep 26 21:31 lib
lrwxrwxrwx    1 root     root             3 Sep 26 21:31 lib64 -> lib
dr-xr-xr-x  268 root     root             0 Jan 22 03:24 proc
drwx------    1 root     root          4096 Jan 22 03:25 root
dr-xr-xr-x   11 root     root             0 Jan 22 03:24 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
```

```
C:\Users\WINDOWS 11\Lab8_1>docker run busybox echo "Hello ณัฐภัทร โพธิจันทร์  from busybox"
Hello ณัฐภัทร โพธิจันทร์  from busybox

C:\Users\WINDOWS 11\Lab8_1>docker ps -a
CONTAINER ID   IMAGE              COMMAND                  CREATED          STATUS                          PORTS      NAMES
55bdae6d1f40   busybox            "echo 'Hello ณัฐภัทร…"    50 seconds ago   Exited (0) 49 seconds ago
stoic_merkle
80a7f15563d6   busybox            "sh"                     4 minutes ago    Exited (0) 2 minutes ago                   affectionate_ramanujan
4d487d742bbe   busybox            "sh"                     4 minutes ago    Exited (0) 4 minutes ago                   flamboyant_nash
245a7ed44b96   demo_image:latest  "jupyter lab --ip=0.…"   3 months ago     Exited (255) 22 minutes ago   8000/tcp     epic_galileo
31fb4226440b   demo_image         "jupyter lab --ip=0.…"   3 months ago     Exited (0) 3 months ago                    eager_sanderson
```

1) เมื่อใช้ option -it ในคำสั่ง run ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป
ทำให้สามารถรัน container ในแบบ interactive และใช้ shell ภายใน container ได้

2) คอลัมน์ STATUS จากการรันคำสั่ง docker ps -a แสดงถึงข้อมูลอะไร
แสดงข้อมูลสถานะ ของcontainer เช่น Exited (0) 2 minutes ago หมายถึง container หยุดทำงานสำเร็จ (รหัส 0) เมื่อ 2 นาทีที่แล้ว

12. ป้อนคำสั่ง $ docker rm <container ID ที่ต้องการลบ>

[Check point#3] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงผลลัพธ์ที่ได้ในขั้นตอนที่ 12

```
C:\Users\WINDOWS 11\Lab8_1>docker ps -a
CONTAINER ID   IMAGE               COMMAND                  CREATED          STATUS                       PORTS      NAMES
55bdae6d1f40   busybox             "echo 'Hello กัฐภ์ ทร…"    50 seconds ago   Exited (0) 49 seconds ago
stoic_merkle
80a7f15563d6   busybox             "sh"                     4 minutes ago    Exited (0) 2 minutes ago                affectionate_ramanujan
4d487d742bbe   busybox             "sh"                     4 minutes ago    Exited (0) 4 minutes ago                flamboyant_nash
245a7ed44b96   demo_image:latest   "jupyter lab --ip=0.…"   3 months ago     Exited (255) 22 minutes ago   8000/tcp   epic_galileo
31fb4226440b   demo_image          "jupyter lab --ip=0.…"   3 months ago     Exited (0) 3 months ago                 eager_sanderson

C:\Users\WINDOWS 11\Lab8_1>docker rm 245a7ed44b96
245a7ed44b96

C:\Users\WINDOWS 11\Lab8_1>docker ps -a
CONTAINER ID   IMAGE        COMMAND                  CREATED          STATUS                      PORTS     NAMES
55bdae6d1f40   busybox      "echo 'Hello กัฐภ์ ทร…"    11 minutes ago   Exited (0) 11 minutes ago             stoic_merkle
80a7f15563d6   busybox      "sh"                     14 minutes ago   Exited (0) 12 minutes ago             affectionate_ramanujan
4d487d742bbe   busybox      "sh"                     15 minutes ago   Exited (0) 15 minutes ago             flamboyant_nash
31fb4226440b   demo_image   "jupyter lab --ip=0.…"   3 months ago     Exited (0) 3 months ago               eager_sanderson
```

แบบฝึกปฏิบัติที่ 8.2: สร้าง Docker file และ Docker image

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

สำหรับเครื่องที่ใช้ระบบปฏิบัติการวินโดวส์ (Windows) บันทึกคำสั่งต่อไปนี้ลงในไฟล์ โดยใช้ Text Editor ที่มี

```
FROM busybox
CMD echo “Hi there. This is my first docker image.”
CMD echo “ชื่อ-นามสกุล รหัสนักศึกษา ชื่อเล่น”
```

สำหรับเครื่องที่ใช้ระบบปฏิบัติการ MacOS หรือ Linux บนหน้าต่าง Terminal และป้อนคำสั่งต่อไปนี้

```
$ cat > Dockerfile << EOF
FROM busybox
CMD echo “Hi there. This is my first docker image.”
CMD echo “ชื่อ-นามสกุล รหัสนักศึกษา ชื่อเล่น”
EOF
```

หรือใช้คำสั่ง

```
$ touch Dockerfile
```

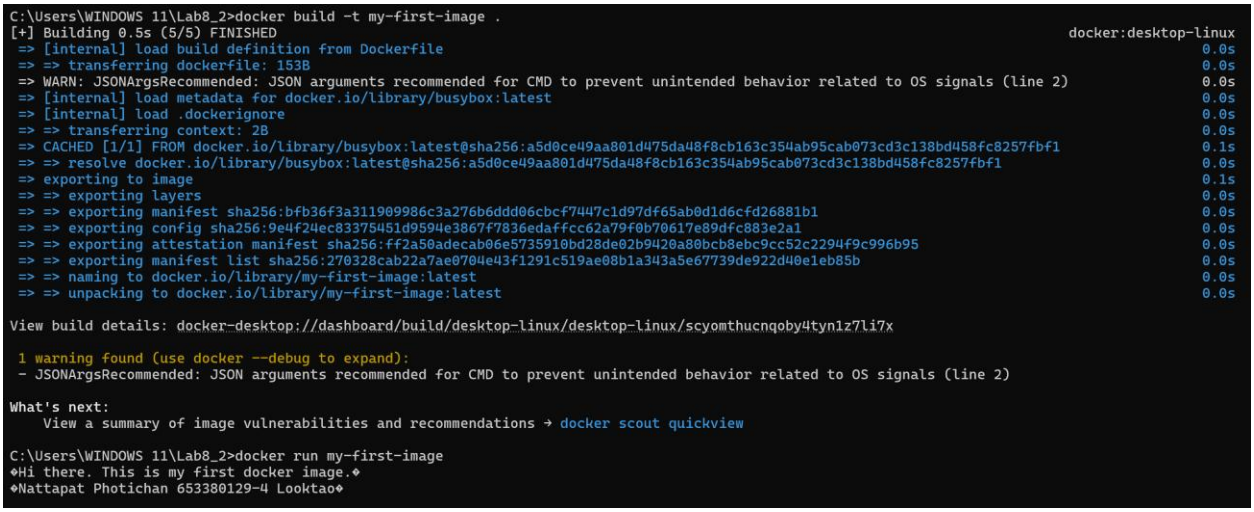
แล้วใช้ Text Editor ในการใส่เนื้อหาแทน

5. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้

   $ docker build -t <ชื่อ Image> .

6. เมื่อ Build สำเร็จแล้ว ให้ทำการรัน Docker image ที่สร้างขึ้นในขั้นตอนที่ 5

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



(1) คำสั่งที่ใช้ในการ run คือ

docker run my-first-image

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

-t จะช่วยตั้งชื่อไฟล์ให้และ ระบุtag ให้ docker image ช่วยให้จัดการ Image ง่ายขึ้น

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

5. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้

   $ docker build -t <username ที่ลงทะเบียนกับ Docker Hub>/lab8
6. ทำการรัน Docker image บน Container ในเครื่องของตัวเองเพื่อทดสอบผลลัพธ์ ด้วยคำสั่ง

   $ docker run <username ที่ลงทะเบียนกับ Docker Hub>/lab8

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

```
C:\Users\WINDOWS 11\Lab8_3>docker build -t nattapatph/lab8 .
[+] Building 3.8s (6/6) FINISHED                                                docker:desktop-linux
 => [internal] load build definition from Dockerfile                                           0.1s
 => => transferring dockerfile: 174B                                                           0.0s
 => [internal] load metadata for docker.io/library/busybox:latest                              0.0s
 => [internal] load .dockerignore                                                              0.0s
 => => transferring context: 2B                                                                0.0s
 => CACHED [1/1] FROM docker.io/library/busybox:latest@sha256:a5d0ce49aa801d475da48f8cb163c354  3.2s
 => => resolve docker.io/library/busybox:latest@sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab  3.2s
 => [auth] library/busybox:pull token for registry-1.docker.io                                 0.0s
 => exporting to image                                                                         0.2s
 => => exporting layers                                                                        0.0s
 => => exporting manifest sha256:a913ab03dbdab428e17ffaa3bb765d90cfee8f85cd330c422418a6af25470  0.0s
 => => exporting config sha256:4e7701bf708e27c9471961cbd35a76107610b7b032104d417f891b4d8b01665  0.0s
 => => exporting attestation manifest sha256:5d92f2e97a9149e5847e6844be86183bf61d298fab6d2421c  0.0s
 => => exporting manifest list sha256:cce862d9190a128306534e5dedc2119bb672552e616fce5fdabf491  0.0s
 => => naming to docker.io/nattapatph/lab8:latest                                              0.0s
 => => unpacking to docker.io/nattapatph/lab8:latest                                           0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/q828xgb9tgfhtecshmhxuxpct

 1 warning found (use docker --debug to expand):
 - 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

C:\Users\WINDOWS 11\Lab8_3>docker run nattapatph/lab8
Hi there. My work is done. You can run them from my Docker image.
Nattapat Photichan 653380129-4
```

7. ทำการ Push ตัว Docker image ไปไว้บน Docker Hub โดยการใช้คำสั่ง

   $ docker push <username ที่ลงทะเบียนกับ Docker Hub>/lab8

   ในกรณีที่ติดปัญหาไม่ได้ Login ไว้ก่อน ให้ใช้คำสั่งต่อไปนี้ เพื่อ Login ก่อนทำการ Push

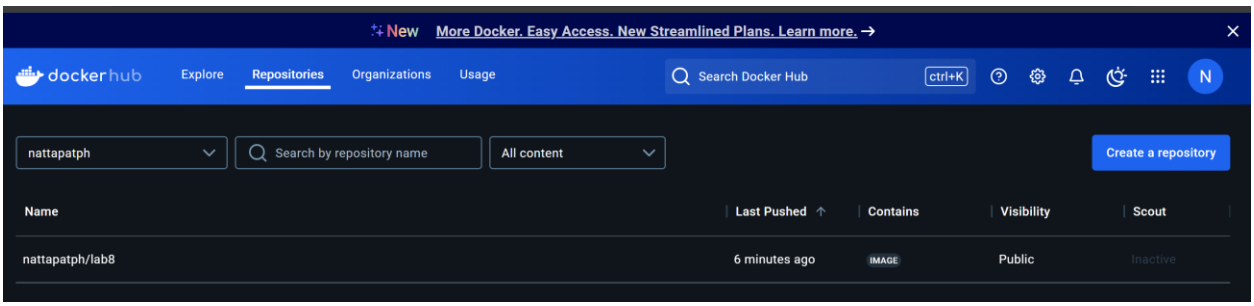
   $ docker login แล้วป้อน Username และ Password ตามที่ระบุใน Command prompt หรือใช้คำสั่ง

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

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

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

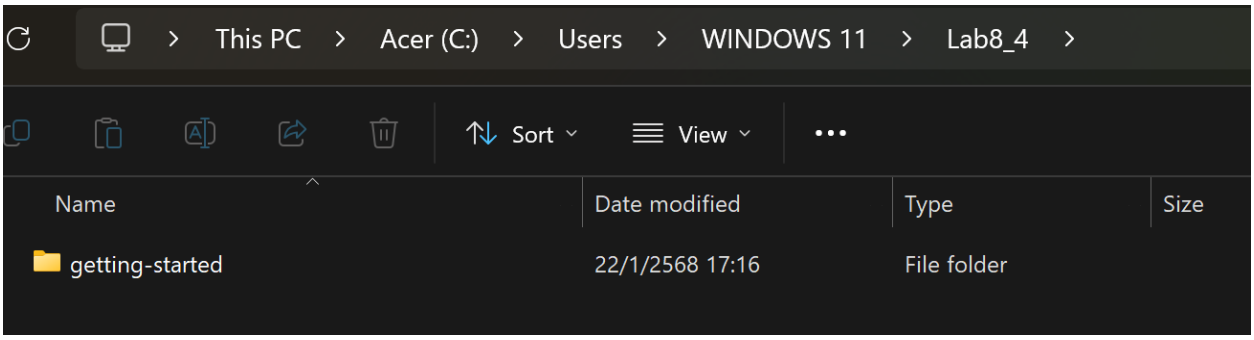
```
C:\Users\WINDOWS 11\Lab8_3>docker push nattapatph/lab8
Using default tag: latest
The push refers to repository [docker.io/nattapatph/lab8]
9c0abc9c5bd3: Mounted from library/busybox
76a3f82bbb94: Pushed
latest: digest: sha256:cce862d9190a128306534e5dedc2119bb672552e616fce5fdabf491b2ae9d1bd size: 855
```



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

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



change.log index.html package.json

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

```

4. ภายใต้ getting-started/app ให้สร้าง Dockerfile พร้อมกับใส่เนื้อหาดังต่อไปนี้ลงไปในไฟล์

   FROM node:18-alpine

   WORKDIR /app

   COPY . .

   RUN yarn install --production

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

   EXPOSE 3000

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

   $ docker build -t <myapp_รหัสนศ. ไม่มีขีด> .

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

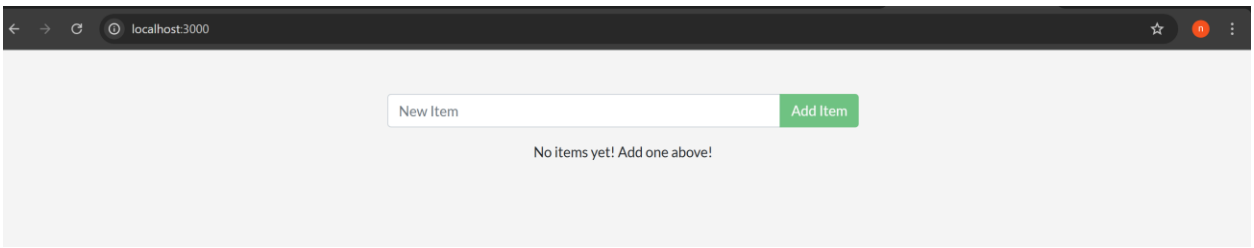
```
C:\Users\WINDOWS 11\Lab8_4\getting-started\app>docker build -t myapp_6533801294 .
[+] Building 25.5s (10/10) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 156B
 => [internal] load metadata for docker.io/library/node:18-alpine
 => [auth] library/node:pull token for registry-1.docker.io
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [1/4] FROM docker.io/library/node:18-alpine@sha256:a24108da7089c2d293ceaa61fb8969ec10821e8efe25572e5abb10b1841eb70b
 => => resolve docker.io/library/node:18-alpine@sha256:a24108da7089c2d293ceaa61fb8969ec10821e8efe25572e5abb10b1841eb70b
 => => sha256:be1495f7193c512f5cbb05ecbdb736ff146c06df9edb4691eaeea6b87c8dbf0b 445B / 445B
 => => sha256:56147f690fe66b9f54d41dbd27cad26d9d5be909e49c5f7d5deb9684488f24b7 1.26MB / 1.26MB
 => => sha256:4e8e5b9bf1c9499b5ef0f37c3dbeb5ab9e0cdde6f5b6d289e093f5617ad4a3a5 40.00MB / 40.00MB
 => => sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c24403d592134f0 3.64MB / 3.64MB
 => => extracting sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c24403d592134f0
 => => extracting sha256:4e8e5b9bf1c9499b5ef0f37c3dbeb5ab9e0cdde6f5b6d289e093f5617ad4a3a5
 => => extracting sha256:56147f690fe66b9f54d41dbd27cad26d9d5be909e49c5f7d5deb9684488f24b7
 => => extracting sha256:be1495f7193c512f5cbb05ecbdb736ff146c06df9edb4691eaeea6b87c8dbf0b
 => [internal] load build context
 => => transferring context: 4.62MB
 => [2/4] WORKDIR /app
 => [3/4] COPY . .
 => [4/4] RUN yarn install --production
 => exporting to image
 => => exporting layers
 => => exporting manifest sha256:54b471dccbe3d8024f350685ee4540f76cba761e7820c4ab89664d18f7d86cd3
 => => exporting config sha256:15df3cb0ee56b9f1e9c0fded8d41d18594928398ca98d18cf5f162fe6e0b50bf
 => => exporting attestation manifest sha256:af2d4470f30f4248371c9bb924ceecacc8397f3a9e088ce9055653de1f4c0934
 => => exporting manifest list sha256:591f933f20d7466cd2dc78a50345e8964a754261728d6e7704a0b011395d9d3e
 => => naming to docker.io/library/myapp_6533801294:latest
 => => unpacking to docker.io/library/myapp_6533801294:latest

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/qnk199roizj92n3v4amp56fnb

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

C:\Users\WINDOWS 11\Lab8_4\getting-started\app>docker run -dp 3000:3000 myapp_6533801294
f9d0280a837e5c07fef1dbcfa7c1027831c5d8dae25f975d571d4bb671cf6c16
```

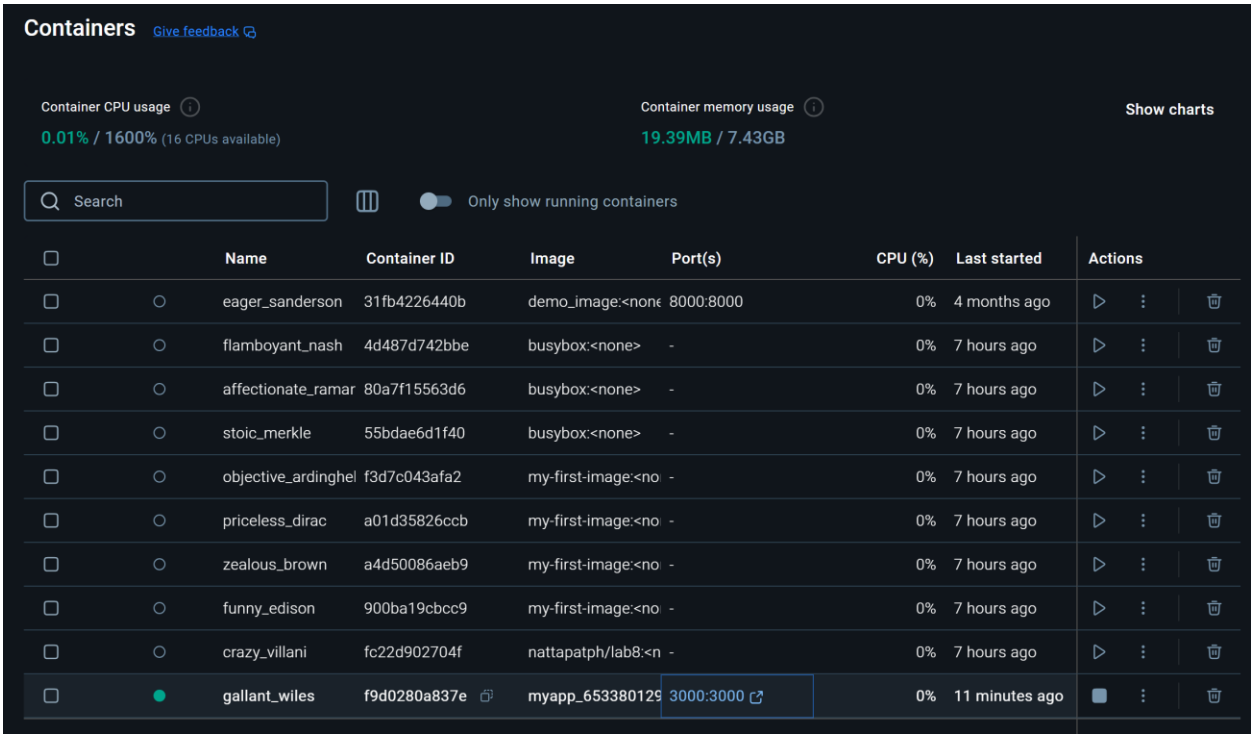
6. ทำการ Start ตัว Container ของแอปพลิเคชันที่สร้างขึ้น โดยใช้คำสั่ง

   $ docker run -dp 3000:3000 <myapp_รหัสนศ. 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
    No items yet! Add one above! เป็น
    There is no TODO item. Please add one to the list. By ชื่อและนามสกุลของนักศึกษา
    b. Save ไฟล์ให้เรียบร้อย
9. ทำการ Build Docker image โดยใช้คำสั่งเดียวกันกับข้อ 5
10. Start และรัน Container ตัวใหม่ โดยใช้คำสั่งเดียวกันกับข้อ 6

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

```
C:\Users\WINDOWS 11\Lab8_4\getting-started\app>docker build -t myapp_6533801294 .
[+] Building 21.1s (10/10) FINISHED                                                                                  docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                                                 0.0s
 => => transferring dockerfile: 156B                                                                                                 0.0s
 => [internal] load metadata for docker.io/library/node:18-alpine                                                                    3.2s
 => [auth] library/node:pull token for registry-1.docker.io                                                                          0.0s
 => [internal] load .dockerignore                                                                                                    0.0s
 => => transferring context: 2B                                                                                                      0.0s
 => [internal] load build context                                                                                                    0.0s
 => => transferring context: 8.11kB                                                                                                  0.0s
 => [1/4] FROM docker.io/library/node:18-alpine@sha256:a24108da7089c2d293ceaa61fb8969ec10821e8efe25572e5abb10b1841eb70b              0.1s
 => => resolve docker.io/library/node:18-alpine@sha256:a24108da7089c2d293ceaa61fb8969ec10821e8efe25572e5abb10b1841eb70b              0.1s
 => CACHED [2/4] WORKDIR /app                                                                                                        0.0s
 => [3/4] COPY . .                                                                                                                   0.1s
 => [4/4] RUN yarn install --production                                                                                             12.2s
 => exporting to image                                                                                                               5.3s
 => => exporting layers                                                                                                              3.5s
 => => exporting manifest sha256:136612ca5e05005911982958 3aac7026ed7351b4d33b9f80acdf5ea36c9c4ffe                                   0.0s
 => => exporting config sha256:79a2a1a7fe24d051506ccb58291cf7cb078ce501c7f2a159114c55c6a923f30c                                      0.0s
 => => exporting attestation manifest sha256:4499af07a6eb6d7ae6358a79f627dab8b978b9b93d646e21787d9cfab3f4876e                        0.0s
 => => exporting manifest list sha256:4db39023f66c067fe29ae483a45d4a633ace5f7aa028fae69c077b185307b677                               0.0s
 => => naming to docker.io/library/myapp_6533801294:latest                                                                           0.0s
 => => unpacking to docker.io/library/myapp_6533801294:latest                                                                        1.7s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/31286p81ju31au1oibqiy5lv5

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

C:\Users\WINDOWS 11\Lab8_4\getting-started\app>docker run -dp 3000:3000 myapp_6533801294
9eb61d0f802d6e5cb9196a9a48ff1b527ad22d5001ccd53edb0c914554624446f
docker: Error response from daemon: driver failed programming external connectivity on endpoint goofy_bell (a13b5c6ab89fad29b19ca3087f83446004a1fb0e
59fc3f55b1ff17d03c70ef42): Bind for 0.0.0.0:3000 failed: port is already allocated.
```

(1) Error ที่เกิดขึ้นหมายความอย่างไร และเกิดขึ้นเพราะอะไร

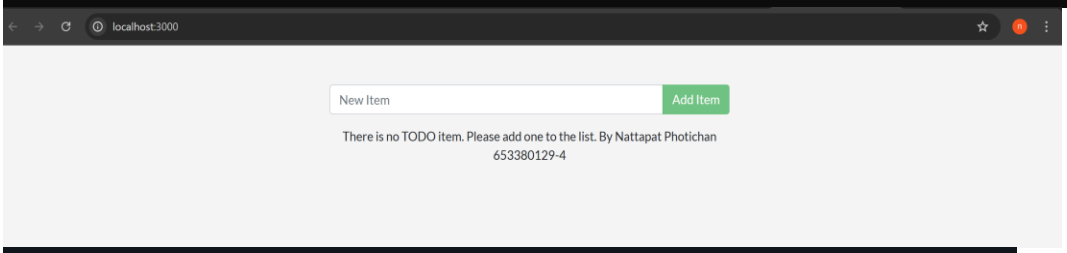
เกิดจาก พอร์ต 3000 ถูกใช้งานโดยcontainerอื่นอยู่

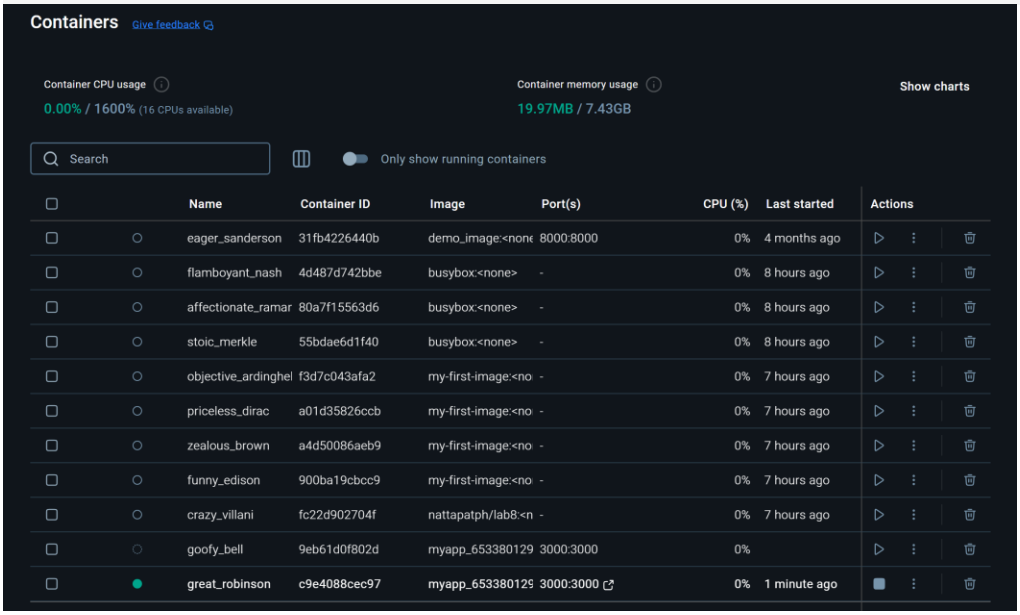
11. ลบ Container ของ Web application เวอร์ชันก่อนแก้ไขออกจากระบบ โดยใช้วิธีใดวิธีหนึ่งดังต่อไปนี้
    a. ผ่าน Command line interface
        i. ใช้คำสั่ง $ docker ps เพื่อดู Container ID ที่ต้องการจะลบ
        ii. Copy หรือบันทึก Container ID ไว้
        iii. ใช้คำสั่ง $ docker stop <Container ID ที่ต้องการจะลบ> 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

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

```
C:\Users\WINDOWS 11\Lab8_4\getting-started\app>docker build -t myapp_6533801294 .
[+] Building 19.0s (10/10) FINISHED                                                                                    docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                                                0.0s
 => => transferring dockerfile: 156B                                                                                                0.0s
 => [internal] load metadata for docker.io/library/node:18-alpine                                                                   2.4s
 => [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:a24108da7089c2d293ceaa61fb8969ec10821e8efe25572e5abb10b1841eb70b             0.0s
 => => resolve docker.io/library/node:18-alpine@sha256:a24108da7089c2d293ceaa61fb8969ec10821e8efe25572e5abb10b1841eb70b             0.0s
 => [internal] load build context                                                                                                   0.0s
 => => transferring context: 8.11kB                                                                                                 0.0s
 => CACHED [2/4] WORKDIR /app                                                                                                       0.0s
 => [3/4] COPY . .                                                                                                                  0.1s
 => [4/4] RUN yarn install --production                                                                                            11.1s
 => exporting to image                                                                                                              5.1s
 => => exporting layers                                                                                                             3.6s
 => => exporting manifest sha256:6168bc7e4ace15fbd699baa23511e632ed5b4e0908b9ab6643f7885042688c23                                   0.0s
 => => exporting config sha256:828bb5aa874e2b9e82775f674a3d8f4868ce78cdf57def8706b1e22d9684c971                                     0.0s
 => => exporting attestation manifest sha256:d1d39c91b8c3e53d651869290e6a002db3b2abf2da44114069c5397f5cbf0dfb                       0.0s
 => => exporting manifest list sha256:c2ff310f301c3522134bf56cb4e71b6e66ba2ad987480fdf75fa4baa9b085d81                              0.0s
 => => naming to docker.io/library/myapp_6533801294:latest                                                                          0.0s
 => => unpacking to docker.io/library/myapp_6533801294:latest                                                                       1.5s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/qlp8hoq6s4lwyni5jko5syum9

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

C:\Users\WINDOWS 11\Lab8_4\getting-started\app>docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED              STATUS              PORTS                    NAMES
97cf72705df4   4db39023f66c   "docker-entrypoint.s…"   About a minute ago   Up About a minute   0.0.0.0:3000->3000/tcp   happy_driscoll

C:\Users\WINDOWS 11\Lab8_4\getting-started\app>docker stop 97cf72705df4a297078095f658a8a74d7f832a1611ddb3966d457f3ac9f9c737
97cf72705df4a297078095f658a8a74d7f832a1611ddb3966d457f3ac9f9c737

C:\Users\WINDOWS 11\Lab8_4\getting-started\app>docker rm 97cf72705df4a297078095f658a8a74d7f832a1611ddb3966d457f3ac9f9c737
97cf72705df4a297078095f658a8a74d7f832a1611ddb3966d457f3ac9f9c737

C:\Users\WINDOWS 11\Lab8_4\getting-started\app>docker run -dp 3000:3000 myapp_6533801294
c9e4088cec9765e4f8956315f88a213d0b33c8391afe796bd26fe4ab4c439e8a
```





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

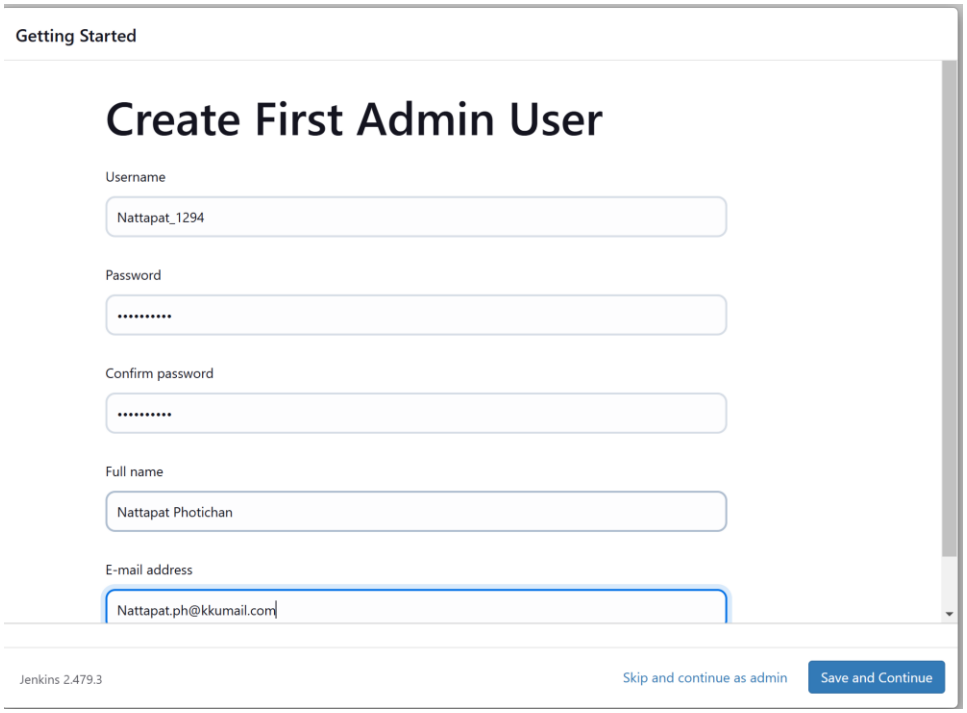
   $ docker run -p 8080:8080 -p 50000:50000 --restart=on-failure -v jenkins_home:/var/jenkins_home jenkins/jenkins:lts-jdk17
3. บันทึกรหัสผ่านของ Admin user ไว้สำหรับ log-in ในครั้งแรก

[Check point#12] Capture หน้าจอที่แสดงผล Admin password

```
C:\Users\WINDOWS 11>docker exec 8bc961b1fa3b cat /var/jenkins_home/secrets/initialAdminPassword
825c8b3e2ca54dd0baa784e8f60c0e7c
```

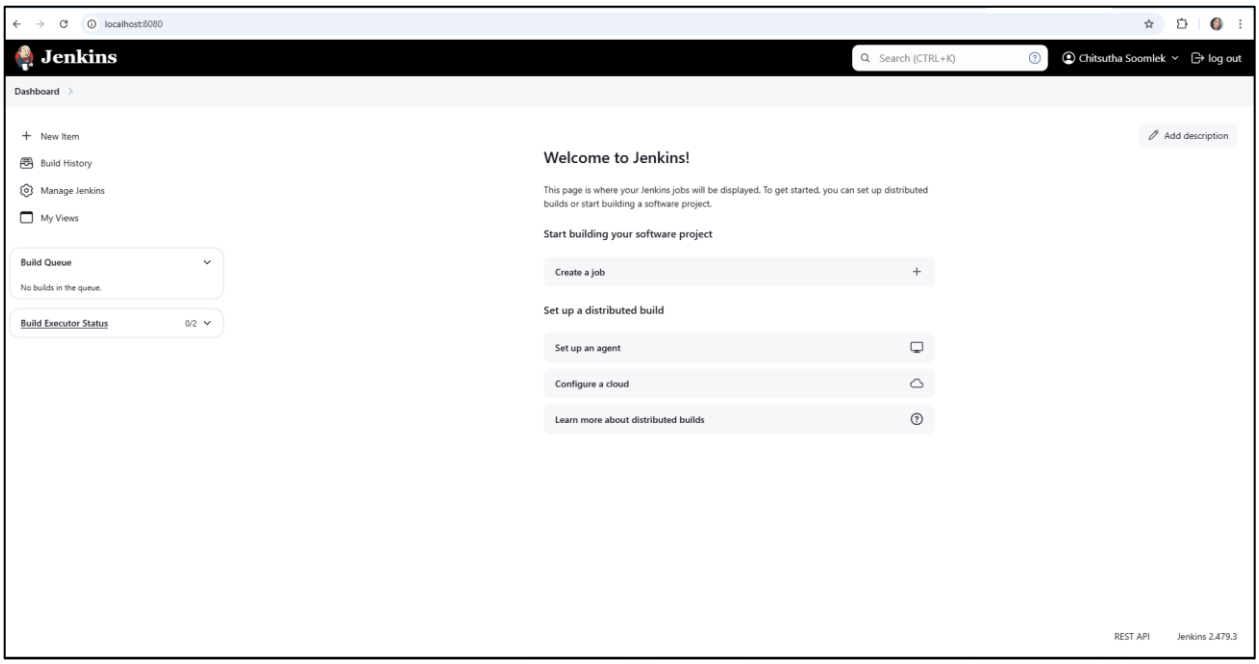
4. เมื่อได้รับการยืนยันว่า Jenkins is fully up and running ให้เปิดบราวเซอร์ และป้อนที่อยู่เป็น localhost:8080
5. ทำการ Unlock Jenkins ด้วยรหัสผ่านที่ได้ในข้อที่ 3
6. สร้าง Admin User โดยใช้ username เป็นชื่อจริงของนักศึกษาพร้อมรหัสสี่ตัวท้าย เช่น somsri_3062

[Check point#13] Capture หน้าจอที่แสดงผลการตั้งค่า

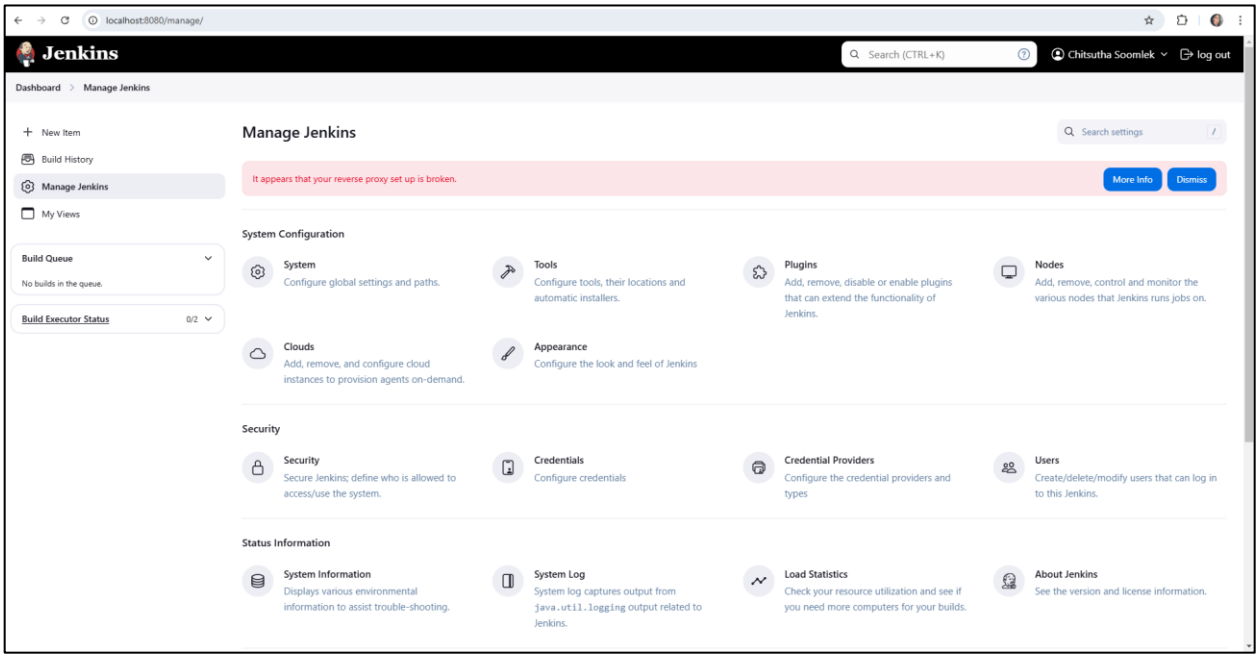


7. กำหนด Jenkins URL เป็น http://localhost:8080/lab8

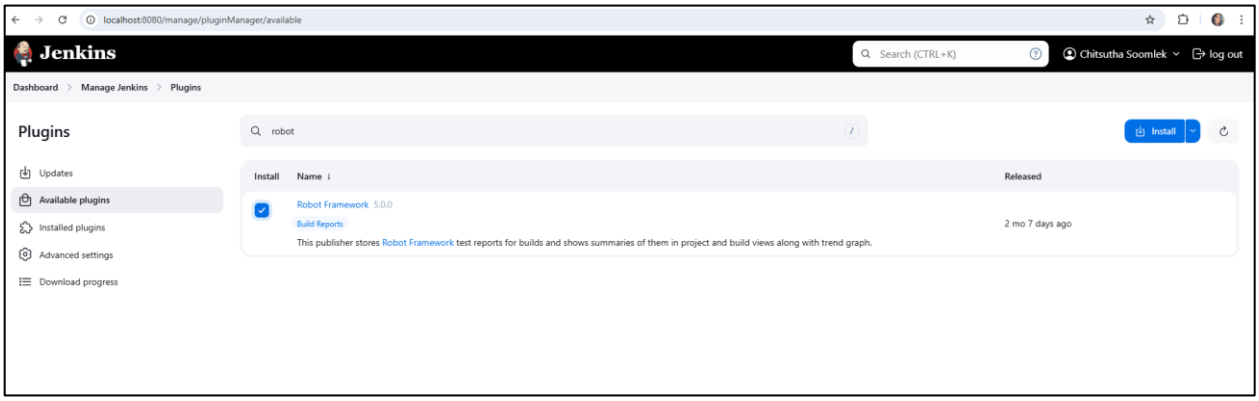
8. เมื่อติดตั้งเรียบร้อยแล้วจะพบกันหน้า Dashboard ดังแสดงในภาพ



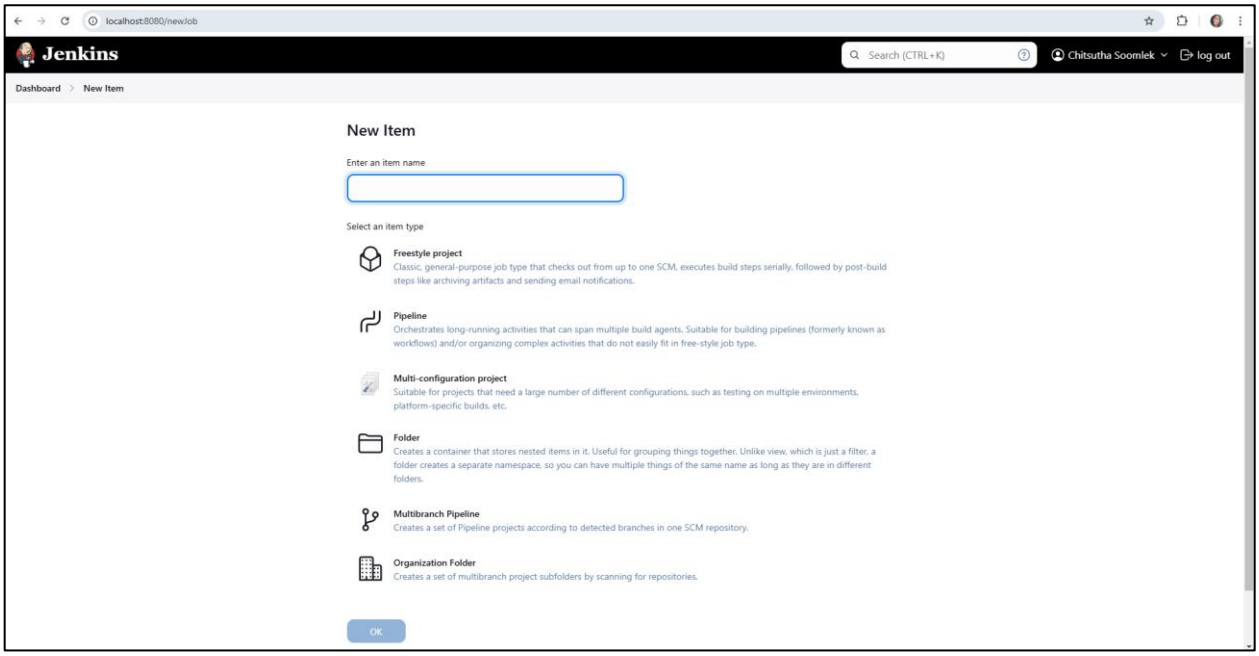
9. เลือก Manage Jenkins แล้วไปที่เมนู Plugins



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

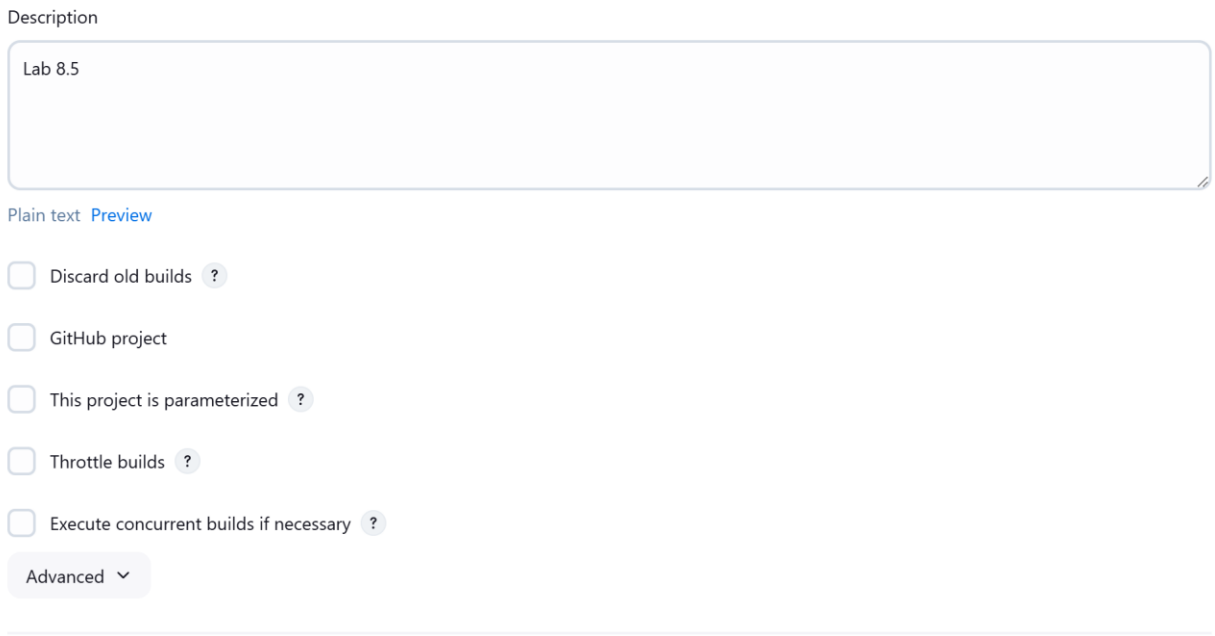
Description: Lab 8.5

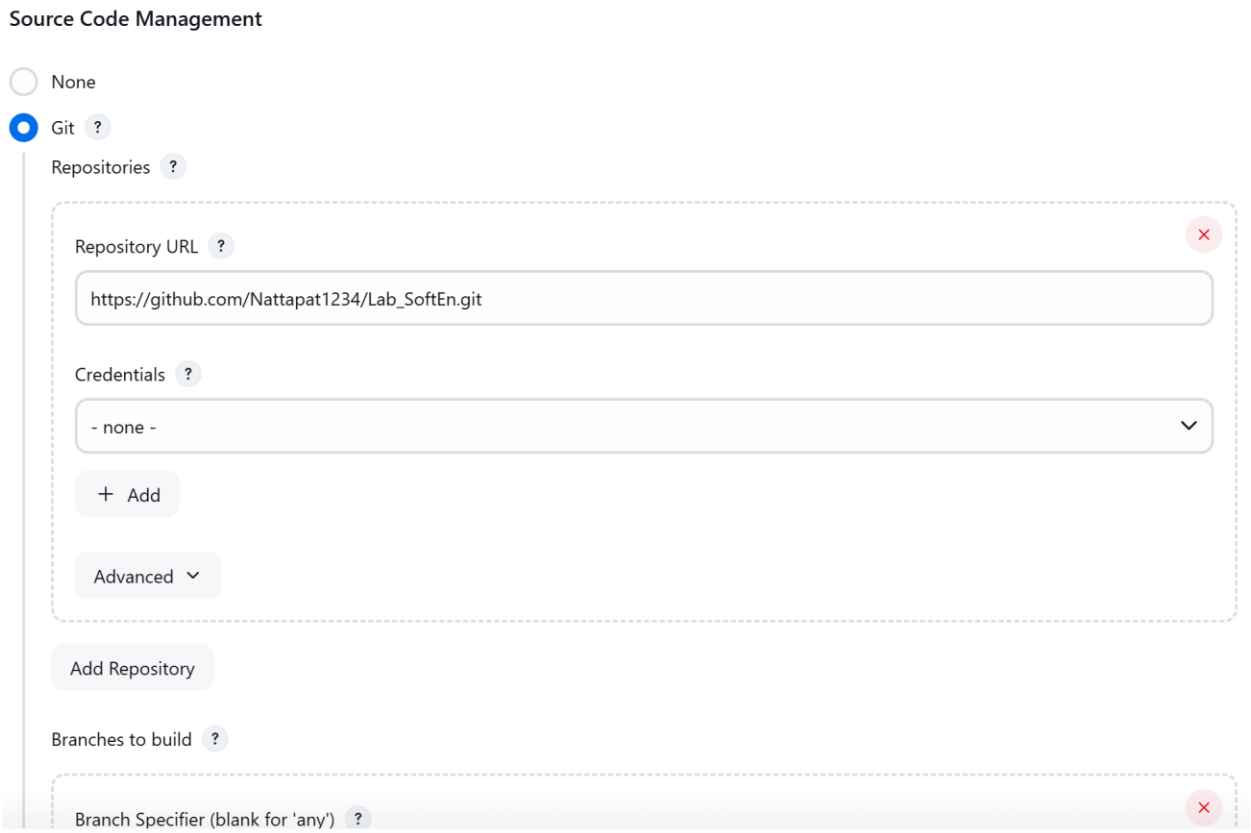
GitHub project: กดเลือก แล้วใส่ Project URL เป็น repository ที่เก็บโค้ด .robot (ดูขั้นตอนที่ 12)

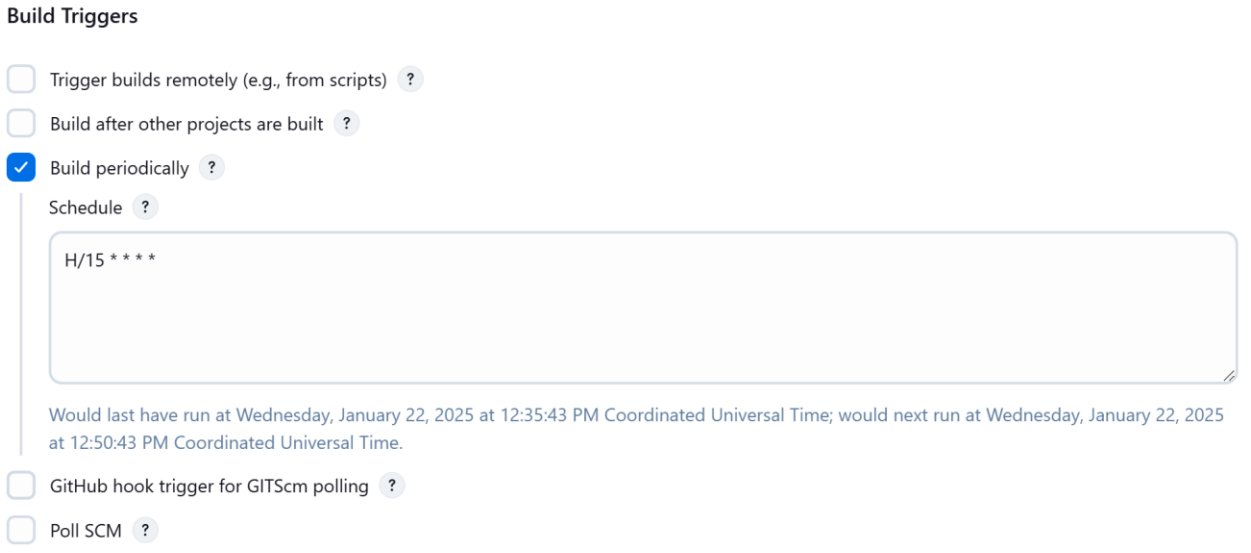
Build Trigger: เลือกแบบ Build periodically แล้วกำหนดให้ build ทุก 15 นาที

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

[Check point#14] Capture หน้าจอแสดงการตั้งค่า พร้อมกับตอบคำถามต่อไปนี้




Source Code Management
None
Git
Repositories
Repository URL
https://github.com/Nattapat1234/Lab_SoftEn.git
Credentials
- none -
Add
Advanced
Add Repository
Branches to build
Branch Specifier (blank for 'any')


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 Wednesday, January 22, 2025 at 12:35:43 PM Coordinated Universal Time; would next run at Wednesday, January 22, 2025 at 12:50:43 PM Coordinated Universal Time.
GitHub hook trigger for GITScm polling
Poll SCM

Build Steps

Execute shell

Command

See the list of available environment variables

```
robot LAB_7.2/test_Empty_Destination.robot
robot LAB_7.2/test_Empty_Email.robot
robot LAB_7.2/test_Empty_PhoneNumber.robot
robot LAB_7.2/test_Invalid_Email.robot
robot LAB_7.2/test_Invalid_PhoneNumber.robot
robot LAB_7.2/test_Record_Success.robot
```

Advanced

Publish JUnit test result report

Test report XMLs

Fileset 'includes' setting that specifies the generated raw XML report files, such as 'myproject/target/test-reports/*.xml'. Basedir of the fileset is the workspace root.

LAB_7.2/output.xml

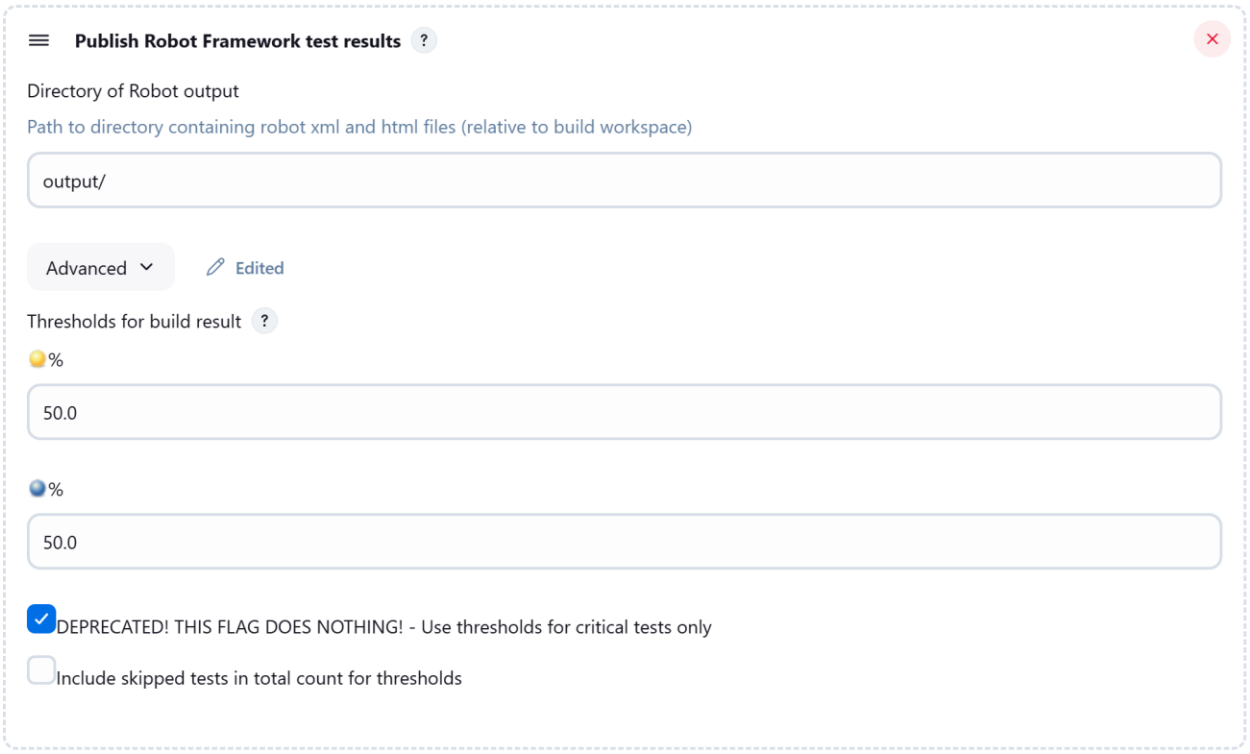
Test output retention

None

keep all the properties

Keep original test names, even when reporting from multiple stages

Health report amplification factor



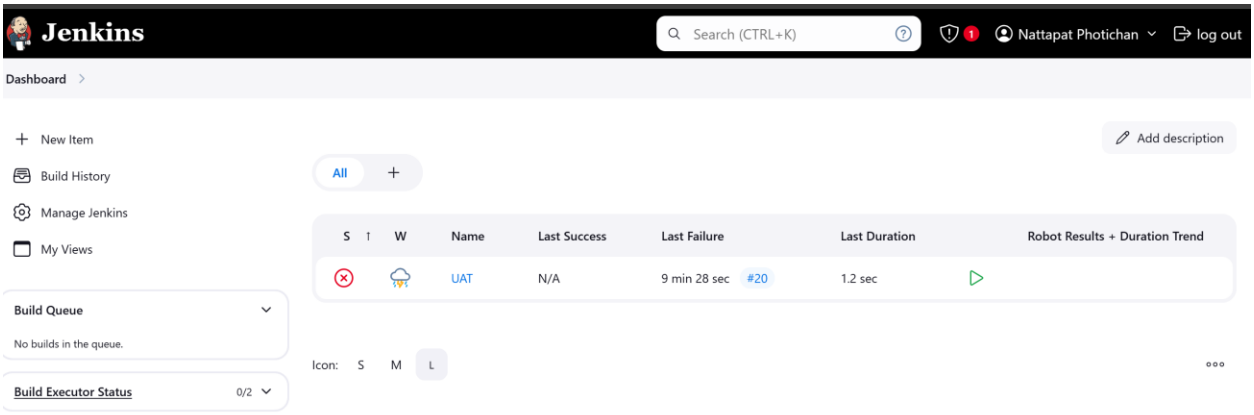
(1) คำสั่งที่ใช้ในการ Execute ไฟล์ .robot ใน Build Steps คือ

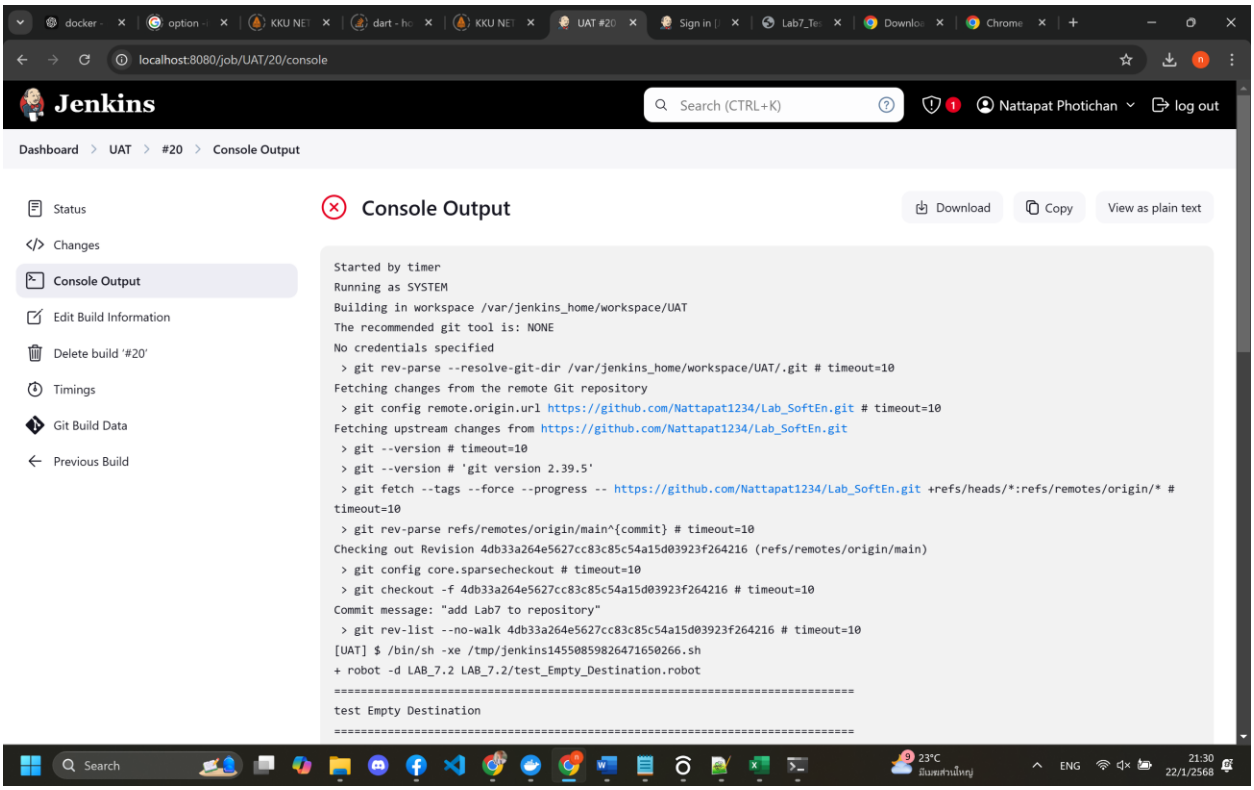
robot <pathของRobot file >

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


Jenkins
Search (CTRL+K)
Nattapat Photichan
log out
Dashboard
New Item
Build History
Manage Jenkins
My Views
Add description
All
S
W
Name
Last Success
Last Failure
Last Duration
Robot Results + Duration Trend
UAT
N/A
9 min 28 sec
#20
1.2 sec
Build Queue
No builds in the queue.
Build Executor Status
0/2
Icon: S M L


localhost:8080/job/UAT/20/console
Jenkins
Search (CTRL+K)
Nattapat Photichan
log out
Dashboard > UAT > #20 > Console Output
Status
Changes
Console Output
Edit Build Information
Delete build '#20'
Timings
Git Build Data
Previous Build
Console Output
Download
Copy
View as plain text
Started by timer
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/Nattapat1234/Lab_SoftEn.git # timeout=10
Fetching upstream changes from https://github.com/Nattapat1234/Lab_SoftEn.git
> git --version # timeout=10
> git --version # 'git version 2.39.5'
> git fetch --tags --force --progress -- https://github.com/Nattapat1234/Lab_SoftEn.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision 4db33a264e5627cc83c85c54a15d03923f264216 (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f 4db33a264e5627cc83c85c54a15d03923f264216 # timeout=10
Commit message: "add Lab7 to repository"
> git rev-list --no-walk 4db33a264e5627cc83c85c54a15d03923f264216 # timeout=10
[UAT] $ /bin/sh -xe /tmp/jenkins14550859826471650266.sh
+ robot -d LAB_7.2 LAB_7.2/test_Empty_Destination.robot
test Empty Destination