Lab Worksheet

ชื่อ-นามสกุล ประจักษ์ สายแถม รหัสนักศึกษา 653380270-3 Section 2

Lab#8 – Software Deployment Using Docker

วัตถุประสงค์การเรียนรู้

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

Pre-requisite

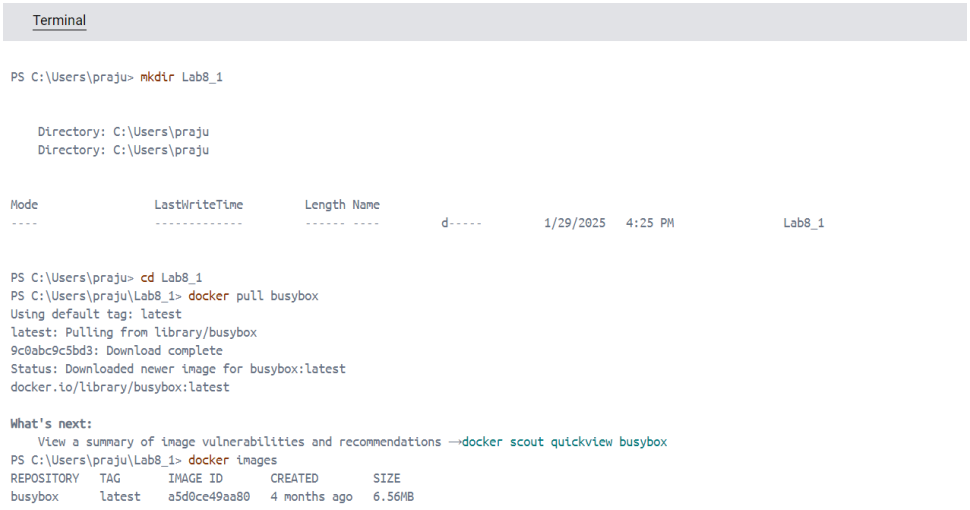
1. ติดตั้ง Docker desktop ลงบนเครื่องคอมพิวเตอร์ โดยดาวน์โหลดจาก https://www.docker.com/get-started
2. สร้าง Account บน Docker hub (https://hub.docker.com/signup)
3. กำหนดให้ $ หมายถึง Command prompt และ <> หมายถึง ให้ป้อนค่าของพารามิเตอร์ที่กำหนด

แบบฝึกปฏิบัติที่ 8.1 Hello world - รัน Container จาก Docker image

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

Lab Worksheet

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



(1) สิ่งที่อยู่ภายใต้คอลัมน์ Repository คืออะไร

ชื่อ docker image

(2) Tag ที่ใช้บ่งบอกถึงอะไร

Version docker

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

Terminal

```
.56MB
PS C:\Users\praju\Lab8_1> docker run busybox
PS C:\Users\praju\Lab8_1> docker run -it busybox sh
/ # ls
bin    etc    lib    proc   sys    usr
dev    home   lib64  root   tmp    var
/ # ls -la
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 29 09:30 proc
drwx------    1 root     root          4096 Jan 29 09:31 root
dr-xr-xr-x   11 root     root             0 Jan 29 09:30 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 C:\Users\praju\Lab8_1> docker run busybox echo "Hello Prachak Saithaem from busybox"
Hello Prachak Saithaem from busybox
PS C:\Users\praju\Lab8_1> docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS                          PORTS     NAMES
87c890b51494   busybox   "echo 'Hello Prachak…"   14 seconds ago   Exited (0) 12 seconds ago                 elastic_sanderson712fce0bf66a   bus
ybox   "sh"                    2 minutes ago    Exited (0) About a minute ago             sweet_merkle
11f440b92c8e   busybox   "sh"                     2 minutes ago    Exited (0) 2 minutes ago                  interesting_matsumoto
PS C:\Users\praju\Lab8_1>
```

(1) เมื่อใช้ option -it ในคำสั่ง run ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป

รัน container และเปิด shell เพื่อรันคำสั่งใน busy box โดย

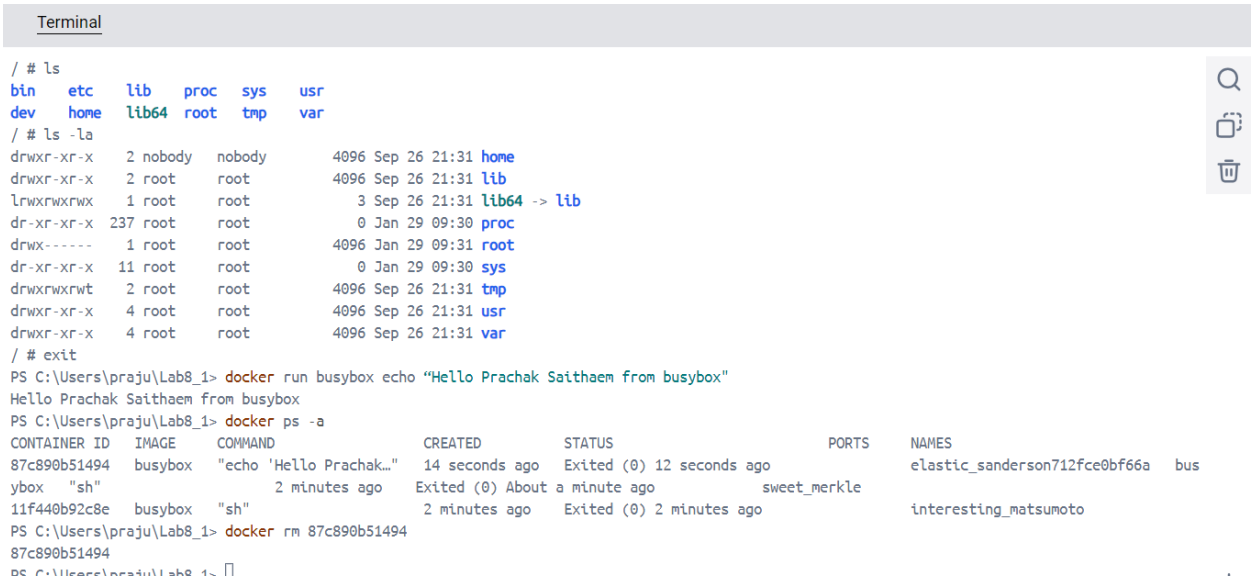
I คือ inactive input และ T คือ แสดงผลแบบ terminal

(2) คอลัมน์ STATUS จากการรันคำสั่ง docker ps -a แสดงถึงข้อมูลอะไร

แสดงการทำงานของ container โดย Exited คือ หยุดทำงานแล้ว และ Up คือ กำลังทำงาน

12. ป้อนคำสั่ง $ docker rm <container ID ที่ต้องการลบ>

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



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) บันทึกคำสั่งต่อไปนี้ลงในไฟล์โดยใช้ 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

```
PS C:\Users\praju\Lab8_2> docker build -t docker_1 -f Dockerfile.swp .
[+] Building 4.2s (6/6) FINISHED                                                                                    docker:desktop-linux
 => [internal] load build definition from Dockerfile.swp                                                                          0.0s
 => => transferring dockerfile: 202B                                                                                              0.0s
 => WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 2)  0.0s
 => WARN: MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the same stage because only the last one will b  0.0s
 => WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 3)  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:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138bd458fc8257fbf1           3.8s
 => => resolve docker.io/library/busybox:latest@sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138bd458fc8257fbf1           3.7s
 => [auth] library/busybox:pull token for registry-1.docker.io                                                                    0.0s
 => exporting to image                                                                                                            0.2s
 => => exporting layers                                                                                                           0.0s
 => => exporting manifest sha256:b1fbe970000bb498960b1e58fb0b4e75522e7a4b234a743df71fbde4ab86d68d                                 0.0s
 => => exporting config sha256:0e32263c479da0f428653b582b967caae4e3626b0bc0bf1ffebae7fb23e58ef4                                   0.0s
 => => exporting attestation manifest sha256:3f87a68365d31f57f8d28774ab12fd1abdfcf7f9ba876c2d381ee6cbf37eca1f                     0.1s
 => => exporting manifest list sha256:6df89bb360b9fb5b7646fa64897ff42f835dddbe23bb7666fb6678bd0030532e                            0.0s
 => => naming to docker.io/library/docker_1:latest                                                                                0.0s
 => => unpacking to docker.io/library/docker_1:latest                                                                             0.0s

3 warnings found (use docker --debug to expand):
 - JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 2)
 - MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the same stage because only the last one will be used (line 2
```

(1) คำสั่งที่ใช้ในการ run คือ

Docker run docker_1

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

เป็นการตั้งชื่อให้กับ docker image

แบบฝึกปฏิบัติที่ 8.3: การแชร์ Docker image ผ่าน Docker Hub

1. เปิดใช้งาน Docker desktop และ Login ด้วย Username และ Password ที่ลงทะเบียนกับ Docker Hub เอาไว้

2. เปิด Command line หรือ Terminal จากนั้นสร้าง Directory ชื่อ Lab8_3
3. ย้ายตำแหน่งปัจจุบันไปที่ Lab8_3 เพื่อใช้เป็น Working directory
4. สร้าง Dockerfile.swp ไว้ใน Working directory

สำหรับเครื่องที่ใช้ระบบปฏิบัติการวินโดวส์ บันทึกคำสั่งต่อไปนี้ลงในไฟล์ โดยใช้ Text Editor ที่มี

```
FROM busybox
CMD echo “Hi there. My work is done. You can run them from my Docker image.”
CMD echo “ชื่อ-นามสกุล รหัสนักศึกษา”
```

สำหรับเครื่องที่ใช้ระบบปฏิบัติการ MacOS หรือ Linux บนหน้าต่าง Terminal และป้อนคำสั่งต่อไปนี้

```
$ cat > Dockerfile << EOF
FROM busybox
CMD echo “Hi there. My work is done. You can run them from my Docker image.”
CMD echo “ชื่อ-นามสกุล รหัสนักศึกษา”
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

```
 => [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:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138bd458fc8257fbf1  0.0s
 => => resolve docker.io/library/busybox:latest@sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138bd458fc8257fbf1  0.0s
 => exporting to image                                                                                               0.1s
 => => exporting layers                                                                                              0.0s
 => => exporting manifest sha256:f7958e0815ae5e1414d83079c5b2e72db95dd01c53adbbbf0a13626c34cafca1                    0.0s
 => => exporting config sha256:a03b9a11f6c3d2ded68c4402871f7df40589721befa71d80d18fe0892487d3b4                      0.0s
 => => exporting attestation manifest sha256:88d5440ee84355b932c880c1c199a2211fc133993630672c3c1eac9f4b3e3d02       0.0s
 => => exporting manifest list sha256:4cfac5359d5163648a6f4adf30885b112b392cafb05236f439a0476bff823dee               0.0s
 => => naming to docker.io/jjug/lab8:latest                                                                          0.0s
 => => unpacking to docker.io/jjug/lab8:latest                                                                       0.0s

 3 warnings found (use docker --debug to expand):
 - 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)
 - MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the same stage because only the last one will be used (line 2
)

What's next:
    View a summary of image vulnerabilities and recommendations →docker scout quickview
PS C:\Users\praju\Lab8_3> docker run jjug/lab8
“ประจักษ์ สายแถม 653380270-3”
PS C:\Users\praju\Lab8_3>
```

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

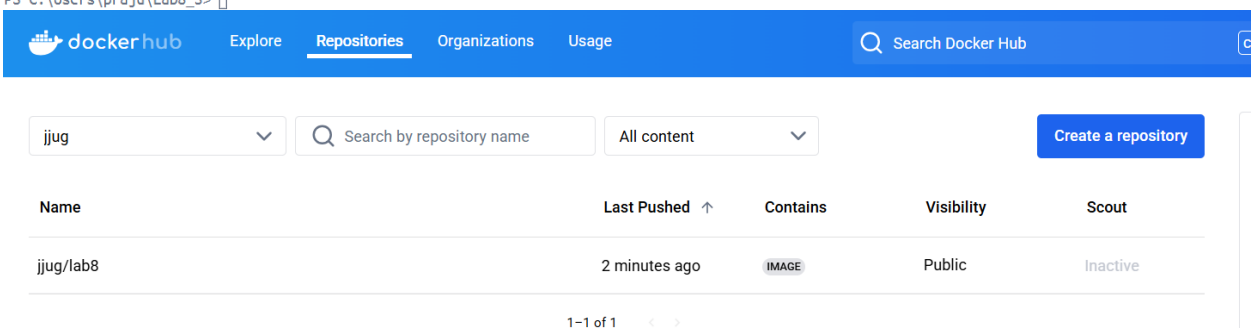
   $ docker login แล้วป้อน Username และ Password ตามที่ระบุใน Command prompt หรือใช้คำสั่ง

   $ 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)

```
    View a summary of image vulnerabilities and recommendations →docker scout quickview
PS C:\Users\praju\Lab8_3> docker run jjug/lab8
“ประจักษ์ สายแถม 653380270-3”
PS C:\Users\praju\Lab8_3> docker push jjug/lab8
Using default tag: latest
The push refers to repository [docker.io/jjug/lab8]
a2e7313cb979: Pushed
9c0abc9c5bd3: Mounted from library/busybox
latest: digest: sha256:4cfac5359d5163648a6f4adf30885b112b392cafb05236f439a0476bff823dee size: 855
PS C:\Users\praju\Lab8_3>
```



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

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

```
PS C:\Users\praju> cd Lab8_4
PS C:\Users\praju\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 | 6.22 MiB/s, done.
Resolving deltas: 100% (523/523), done.
```

```
[+] Building 43.4s (10/10) FINISHED
 => [internal] load build definition from Dockerfile.swp
 => => transferring dockerfile: 160B
 => [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
 => => sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce368990ca771 444B / 444B
 => => sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15bf426d01bc1 1.26MB / 1.26MB
 => => sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e0da02525e63167c 40.01MB / 40.01MB
 => => sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c24403d592134f0 3.64MB / 3.64MB
 => => extracting sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c24403d592134f0
 => => extracting sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e0da02525e63167c
 => => extracting sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15bf426d01bc1
 => => extracting sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce368990ca771
 => [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
```

RAM 1.69 GB CPU 0.17% Disk -- -- GB avail. of -- -- GB

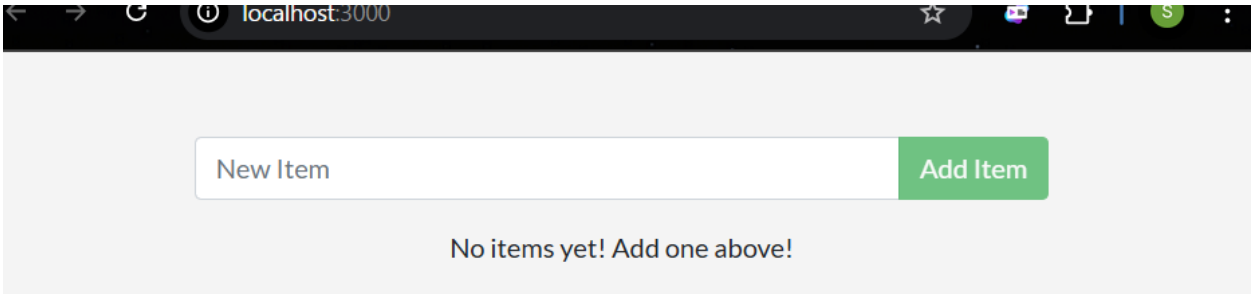
6. ทำการ Start ตัว Container ของแอปพลิเคชันที่สร้างขึ้น โดยใช้คำสั่ง

   $ 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

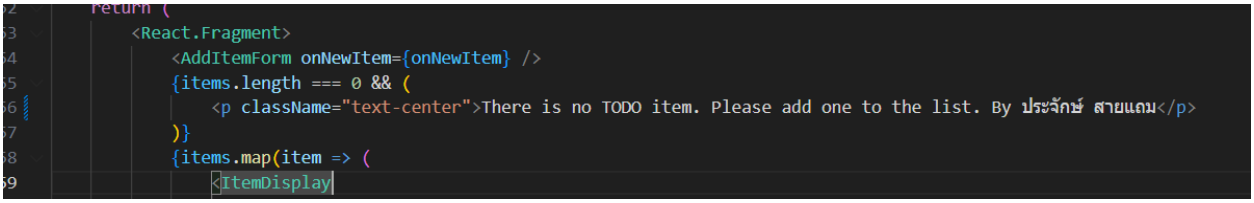
```
PS C:\Users\praju\Lab8_4\getting-started\app> docker run -dp 3000:3000 myapp_6533802703
0e00ff5e69a5978b98a6ccd7b27f0ba42b9458ed5ac1df0c535ae40e0d86593e
PS C:\Users\praju\Lab8_4\getting-started\app>
```



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



```
PS C:\Users\praju\Lab8_4\getting-started\app> docker build -t myapp_6533802703 -f Dockerfile.swp .
[+] Building 44.6s (10/10) FINISHED                                                                                    doc
 => [internal] load build definition from Dockerfile.swp
 => => transferring dockerfile: 160B
 => [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
 => [internal] load build context
 => => transferring context: 8.12kB
 => CACHED [2/4] WORKDIR /app
 => [3/4] COPY . .
 => [4/4] RUN yarn install --production
 => exporting to image
 => => exporting layers
 => => exporting manifest sha256:7283b2e34a00f2de101ebdfe0aed08016592bf1d1d02f9e0e1a3ea86e4a1889e
 => => exporting config sha256:ab35811889f00572c7ecf3de7310dffe23948e1c5da2ccc26dce461a01243eb1
 => => exporting attestation manifest sha256:f889f824eba490801482e4682ccd1ac1f780c2fe9f265b552712a4fe08ff100a
 => => exporting manifest list sha256:9a87dcecc4df15557f8680cc9401717cac13a95b2e12a581ef68b0565478a28f

PS C:\Users\praju\Lab8_4\getting-started\app> docker run -dp 3000:3000 myapp_6533802703
e0a02b3a15d1b177d06a514e5768b226f4313c756892eae3b65160732b8e2d40
docker: Error response from daemon: driver failed programming external connectivity on endpoint eloquent_keller (b6030dff20ab6dd2317ccdcfbf34a84
9791840fb84e4f0e55cfc010500cbf1ec): Bind for 0.0.0.0:3000 failed: port is already allocated.
PS C:\Users\praju\Lab8_4\getting-started\app>
```

(1) Error ที่เกิดขึ้นหมายความอย่างไร และเกิดขึ้นเพราะอะไร

เกิดจากไม่สามารถเชื่อมต่อกับ Port 3000 ได้ เพราะ ถูกใช้งานไปแล้ว

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

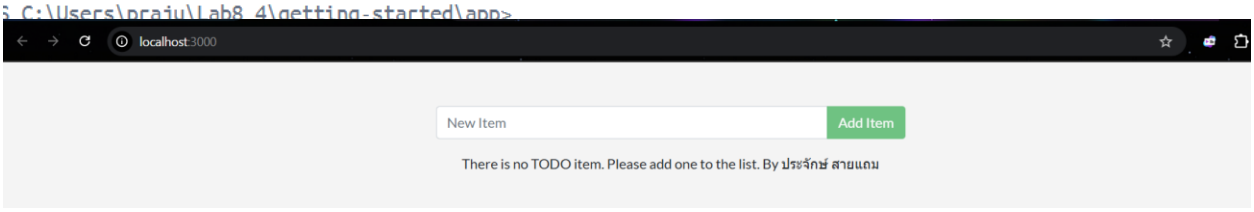
ii. เลือกไอคอนถังขยะในแถวของ Container ที่ต้องการจะลบ

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

```
PS C:\Users\praju\Lab8_4\getting-started\app> docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS         PORTS                    NAMES
0e00ff5e69a5   0412bd92dc11   "docker-entrypoint.s…"   9 minutes ago   Up 9 minutes   0.0.0.0:3000->3000/tcp   great_archimedes
PS C:\Users\praju\Lab8_4\getting-started\app> ^C
PS C:\Users\praju\Lab8_4\getting-started\app> docker stop 0e00ff5e69a5
0e00ff5e69a5
PS C:\Users\praju\Lab8_4\getting-started\app> docker rm 0e00ff5e69a5
0e00ff5e69a5
PS C:\Users\praju\Lab8_4\getting-started\app> docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
   View a summary of image vulnerabilities and recommendations →docker scout quickview
 C:\Users\praju\Lab8_4\getting-started\app> docker run -dp 3000:3000 myapp_6533802703
03d6bf2b200329c27622daa790f5d36669c1410b72b410a3a04ccf8f18826ae
```



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

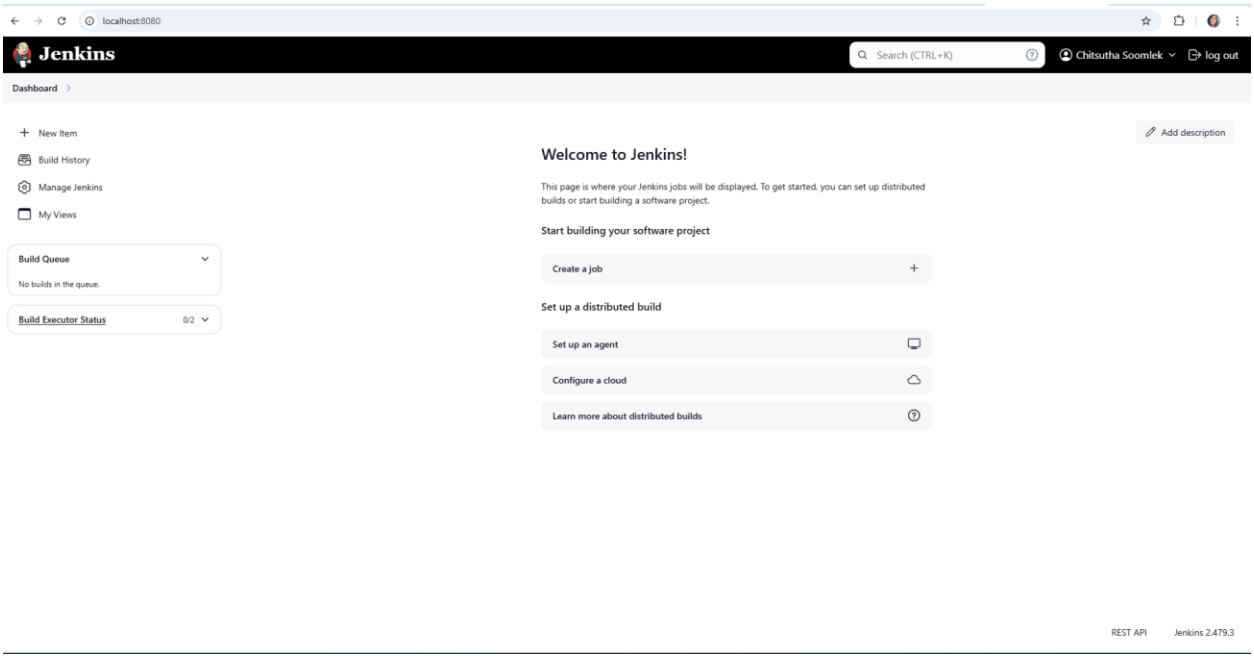
[Check point#12] Capture หน้าจอที่แสดงผล Admin password

```
2025-01-29 10:23:45.591+0000 [id=68]    INFO    hudson.util.Retrier#start: Attempt #1 to do the action check updates server
2025-01-29 10:23:46.085+0000 [id=47]    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:

f9b33a2930d94baaa916aa61a695db5d

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

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

2025-01-29 10:23:53.015+0000 [id=51]    INFO    jenkins.InitReactorRunner$1#onAttained: Completed initialization
```

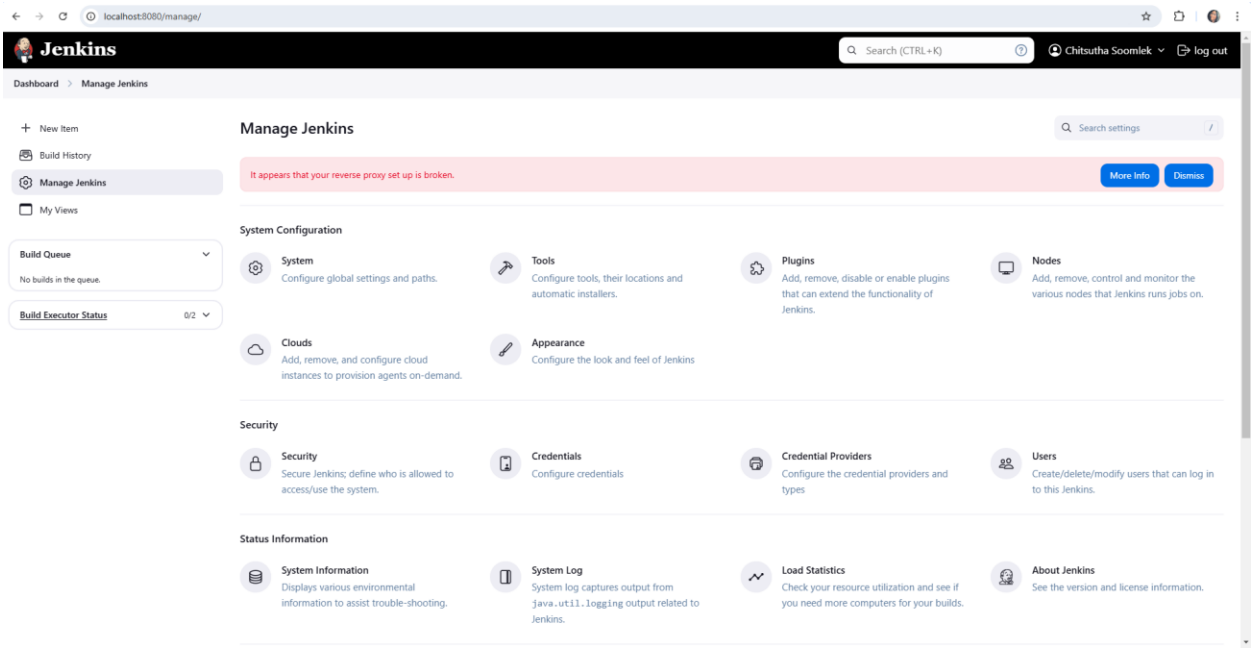
4. เมื่อได้รับการยืนยันว่า Jenkins is fully up and running ให้เปิดบราวเซอร์ และป้อนที่อยู่เป็น localhost:8080
5. ทำการ Unlock Jenkins ด้วยรหัสผ่านที่ได้ในข้อที่ 3
6. สร้าง Admin User โดยใช้ username เป็นชื่อจริงของนักศึกษาพร้อมรหัสสี่ตัวท้าย เช่น somsri_3062

[Check point#13] Capture หน้าจอที่แสดงผลการตั้งค่า

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

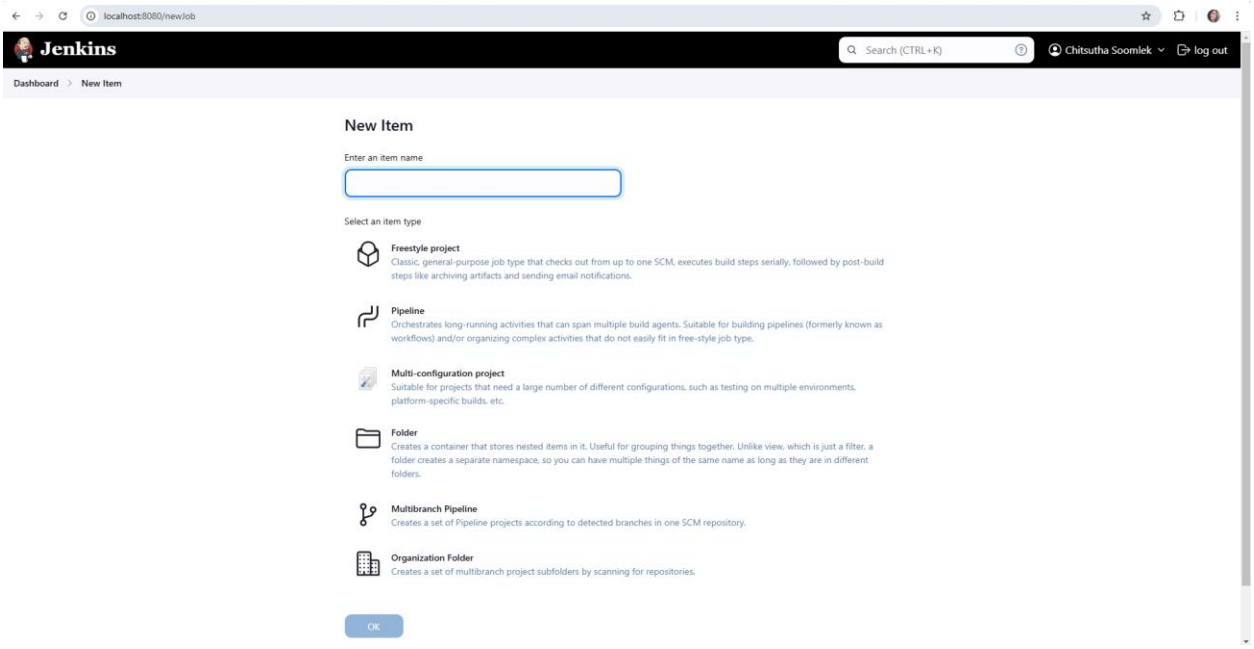


9. เลือก Manage Jenkins แล้วไปที่เมนู Plugins



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

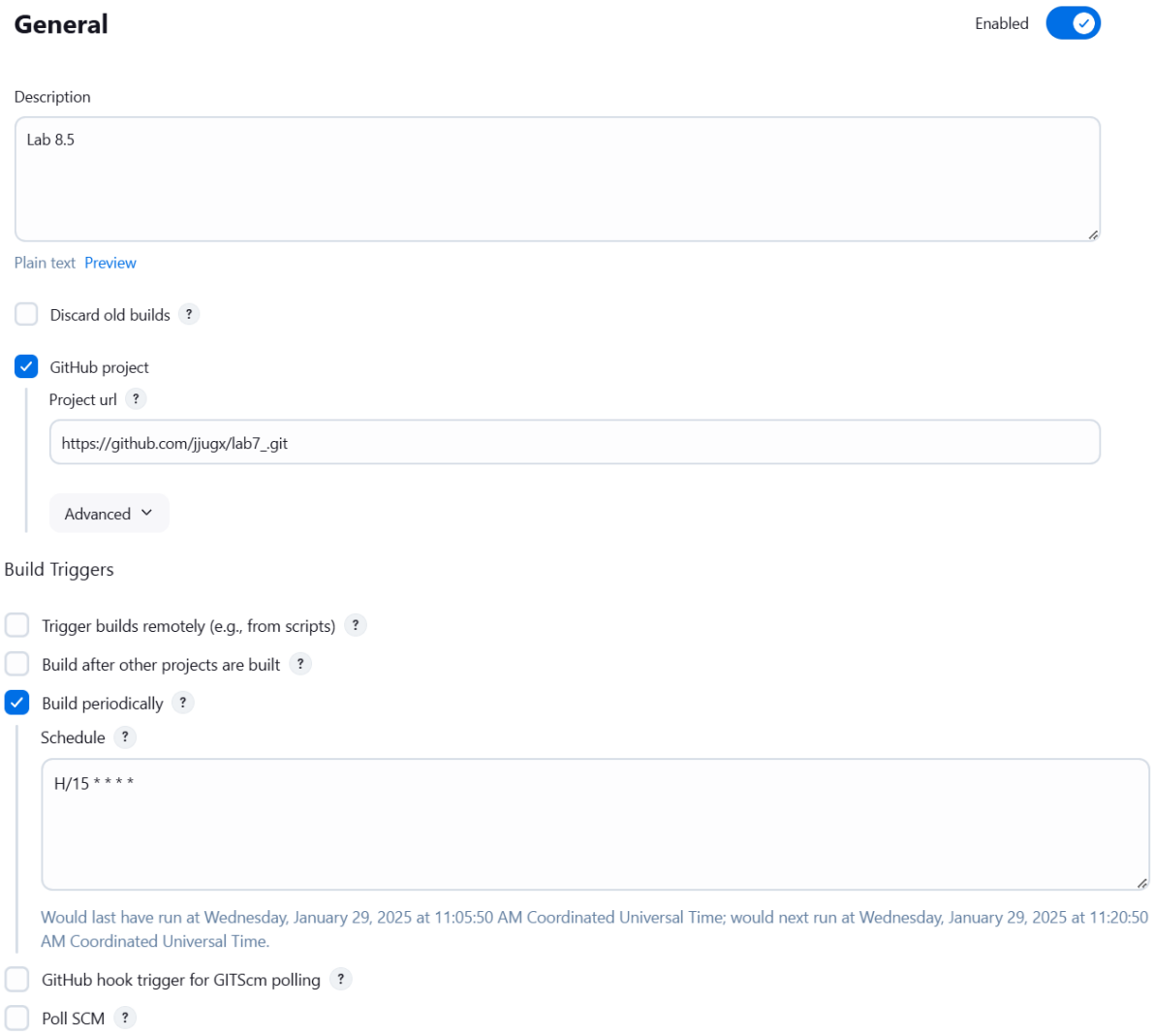
Description: Lab 8.5

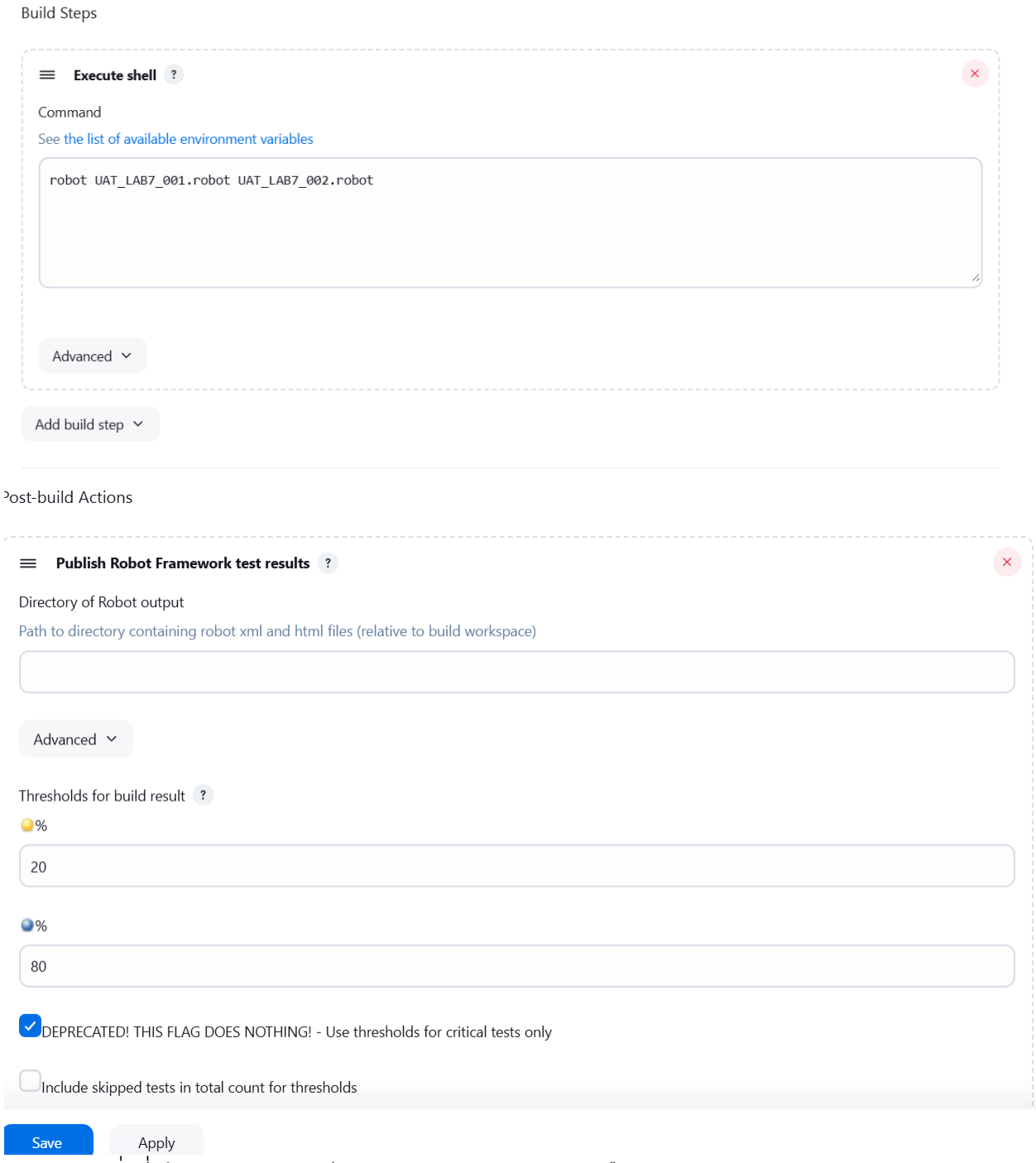
GitHub project: กดเลือก แล้วใส่ Project URL เป็น repository ที่เก็บโค้ด .robot (ดูขั้นตอนที่ 12)

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

robot UAT_LAB7_001.robot UAT_LAB7_002.robot

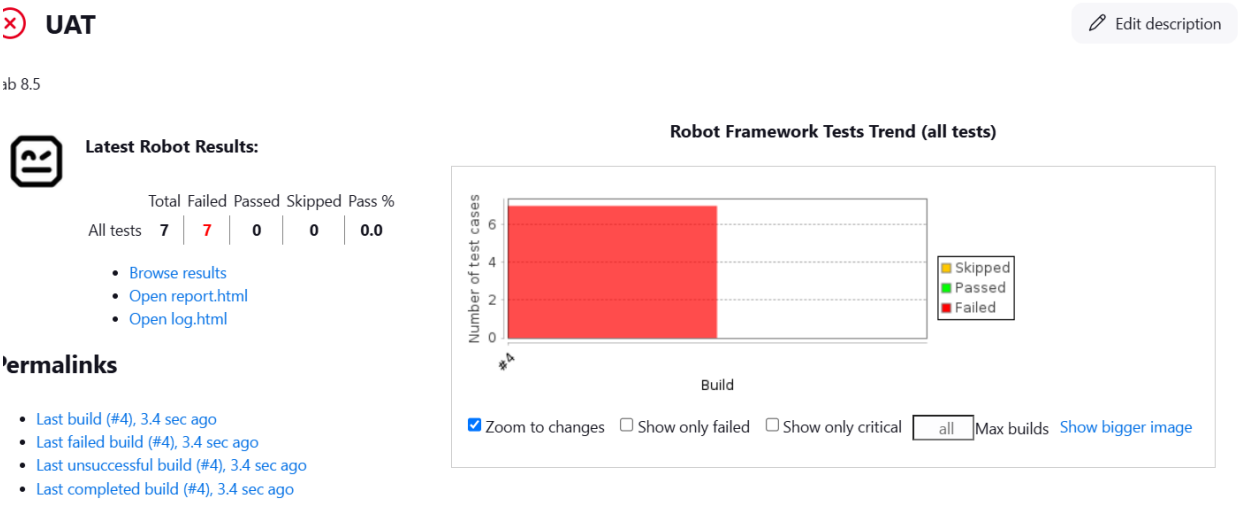
Post-build action: เพิ่ม Publish Robot Framework test results ->

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



Console Output

```
Started by user Prachak Saithaem
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/jjugx/lab7_.git # timeout=10
Fetching upstream changes from https://github.com/jjugx/lab7_.git
 > git --version # timeout=10
 > git --version # 'git version 2.39.5'
 > git fetch --tags --force --progress -- https://github.com/jjugx/lab7_.git +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision 75b684205cf2c04293e90d892f965c86ddb4ab78 (refs/remotes/origin/main)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f 75b684205cf2c04293e90d892f965c86ddb4ab78 # timeout=10
Commit message: "add report"
 > git rev-list --no-walk 270d9b59aee38ad71fe20fe1760323c1f77bb409 # timeout=10
[UAT] $ /bin/sh -xe /tmp/jenkins16154576977628151820.sh
+ robot UAT_LAB7_001.robot UAT_LAB7_002.robot
/tmp/jenkins16154576977628151820.sh: 2: robot: not found
Build step 'Execute shell' marked build as failure
Robot results publisher started...
INFO: Checking test criticality is deprecated and will be dropped in a future release!
-Parsing output xml:
Done!
-Copying log files to build dir:
Done!
-Assigning results to build:
Done!
-Checking thresholds:
Done!
Done publishing Robot results.
Finished: FAILURE
```