Lab Worksheet

ชื่อ-นามสกุล_นายพีระพงษ์ พลชา_รหัสนักศึกษา__653380143-0__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

```
llo.2.o8abzd7d0bxzv2txqv6u0jf4w
99aef0b12147   peeraphong/pphtml:v0.1   "/docker-entrypoint.…"   46 seconds ago   Up 40 s
llo.1.d1pxlhi6oc6lv3uvgpwn8rxja
PS D:\SoftwareEN\Lab8_1> docker images
REPOSITORY              TAG       IMAGE ID       CREATED         SIZE
dockerfilltestforv0.2   latest    5caac0e9eff4   7 weeks ago     363MB
peeraphong/pphtml       v0.1      8987949924fc   2 months ago    279MB
busybox                 latest    a5d0ce49aa80   4 months ago    6.56MB
peeraphong/pphtml       v0.2      0c454cfe6ed8   5 months ago    74MB
PS D:\SoftwareEN\Lab8_1>
```

(1) สิ่งที่อยู่ภายใต้คอลัมน์ Repository คืออะไร ชื่อของ Images

(2) Tag ที่ใช้บ่งบอกถึงอะไร บ่งบอกถึง version ของ images หรือเอาไว้ระบุ tag

5. ป้อนคำสั่ง $ docker run busybox

```
PS D:\SoftwareEN\Lab8_1> 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

6.

```
PS D:\SoftwareEN\Lab8_1> docker run -it busybox sh
/ #
```

7.

```
/ # ls
bin    dev    etc    home   lib    lib64  proc   root   sys    tmp    usr    var
```

8.

```
/ # ls -la
total 48
drwxr-xr-x    1 root     root          4096 Jan 28 08:40 .
drwxr-xr-x    1 root     root          4096 Jan 28 08:40 ..
-rwxr-xr-x    1 root     root             0 Jan 28 08:40 .dockerenv
drwxr-xr-x    2 root     root         12288 Sep 26 21:31 bin
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  317 root     root             0 Jan 28 08:40 proc
drwx------    1 root     root          4096 Jan 28 08:40 root
dr-xr-xr-x   11 root     root             0 Jan 28 08:40 sys
```

9.

```
/ # exit
PS D:\SoftwareEN\Lab8_1>
```

10.

```
PS D:\SoftwareEN\Lab8_1> docker run busybox echo "Hello Peeraphong Poncha from busybox"
Hello Peeraphong Poncha from busybox
```

11.

```
PS D:\SoftwareEN\Lab8_1> docker ps -a
CONTAINER ID   IMAGE                          COMMAND                  CREATED          STATUS
        PORTS     NAMES
5aab1dea3e5f   peeraphong/pphtml:v0.2         "/docker-entrypoint.…"   3 seconds ago    Created
           wonderful_bardeen.1.wl2mqb1pds4gy7ihv5ofwkmib
5273fe8adedf   dockerfilltestforv0.2:latest   "docker-entrypoint.s…"   5 seconds ago    Exited (127) 1 second
ago            adoring_montalcini.1.kdrz3pnor32k6bnezirv7i5eq
194ee5aad3c3   peeraphong/pphtml:v0.2         "/docker-entrypoint.…"   9 seconds ago    Exited (127) 3 seconds
 ago           wonderful_bardeen.1.rd57o9ztoj38vcumh0ua0vh10
3f695417ef8c   dockerfilltestforv0.2:latest   "docker-entrypoint.s…"   10 seconds ago   Exited (127) 6 seconds
 ago           adoring_montalcini.1.8an6vipinmaxkz96zwrb4e8ub
e01444a9ca68   peeraphong/pphtml:v0.2         "/docker-entrypoint.…"   15 seconds ago   Exited (127) 9 seconds
```

(1) เมื่อใช้ option -it ในคำสั่ง run ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป

เข้าถึง container แบบ interactive mode สามารถพิมพ์คำสั่งหรือโต้ตอบกับโปรแกรมใน container ได้

(2) คอลัมน์ STATUS จากการรันคำสั่ง docker ps -a แสดงถึงข้อมูลอะไร

แสดงถึงสถานการณ์ทำงานของ Container ใน Docker

12. ป้อนคำสั่ง $ docker rm <container ID ที่ต้องการลบ>

```
PS D:\SoftwareEN\Lab8_1> docker rm c40
c40
```

[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) 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

File Edit View

```
FROM busybox
CMD echo "Hi there. This is my first docker image."
CMD echo "Peeraphong Poncha 653380143-0 PP"
```

Dockerfile.swp

```
PS D:\SoftwareEN\Lab8_2> docker build -t lab8 -f Dockerfile.swp .
[+] Building 2.8s (6/6) FINISHED                                    docker:desktop-linux
 => [internal] load build definition from Dockerfile.swp                            0.0s
 => => transferring dockerfile: 153B                                                0.0s
 => WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior relate  0.0s
 => WARN: MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the same stag  0.0s
 => WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior relate  0.0s
 => [internal] load metadata for docker.io/library/busybox:latest                   0.0s
 => [internal] load .dockerignore                                                   0.0s
 => => transferring context: 2B                                                     0.0s

PS D:\SoftwareEN\Lab8_2> docker run lab8
Peeraphong Poncha 653380143-0 PP
```

(1) คำสั่งที่ใช้ในการ run คือ

docker run lab8

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

ใช้สำหรับกำหนดชื่อ และ tag ให้กับ Docker image ที่สร้างขึ้น

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 บันทึกคำสั่งต่อไปนี้ลงในไฟล์โดยใช้ 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

```
PS D:\SoftwareEN\Lab8_3> docker build -t peeraphong/lab8 -f Dockerfile.swp .
[+] Building 3.5s (6/6) FINISHED
 => [internal] load build definition from Dockerfile.swp
 => => transferring dockerfile: 177B
 => [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:a5d0ce49aa801c
 => => resolve docker.io/library/busybox:latest@sha256:a5d0ce49aa801d475da48
 => [auth] library/busybox:pull token for registry-1.docker.io
 => exporting to image
 => => exporting layers
```

```
PS D:\SoftwareEN\Lab8_3> docker run peeraphong/lab8
Peeraphong Poncha 653380143-0
```

```
FROM busybox
CMD echo "Hi there. My work is done. You can run them from my Docker image."
CMD echo "Peeraphong Poncha 653380143-0"
```

6. ทำการ Push ตัว Docker image ไปไว้บน Docker Hub โดยการใช้คำสั่ง

   $ docker push <username ที่ลงทะเบียนกับ Docker Hub>/lab8

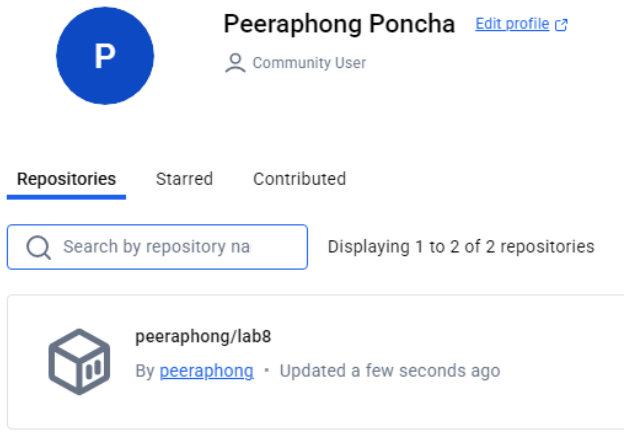
   ในกรณีที่ติดปัญหาไม่ได้ Login ไว้ก่อน ให้ใช้คำสั่งต่อไปนี้ เพื่อ Login ก่อนทำการ Push

   $ 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)

```
PS D:\SoftwareEN\Lab8_3> docker push peeraphong/lab8
Using default tag: latest
The push refers to repository [docker.io/peeraphong/lab8]
9c0abc9c5bd3: Mounted from library/busybox
7b9c790c5f33: Pushed
latest: digest: sha256:b09aa964550dd38e5b65fc84aad85854da671749effc85bbf6134b05b1d08e1c 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

```
PS D:\softwareen\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 | 8.28 MiB/s, done.
Resolving deltas: 100% (523/523), done.
```

```
{
  "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

```
FROM node:18-alpine
WORKDIR /app
COPY . .
RUN yarn install --production
CMD ["node", "src/index.js"]
EXPOSE 3000
```

```
PS D:\softwareen\Lab8_4\getting-started\app> docker build -t myapp_6533801430 .
[+] Building 38.5s (10/10) FINISHED                                    docker:desktop-linux
 => [internal] load build definition from Dockerfile                                  0.1s
 => => transferring dockerfile: 156B                                                  0.0s
 => [internal] load metadata for docker.io/library/node:18-alpine                     3.4s
 => [auth] library/node:pull token for registry-1.docker.io                           0.0s
 => [internal] load .dockerignore                                                     0.0s
 => => transferring context: 2B                                                       0.0s
```

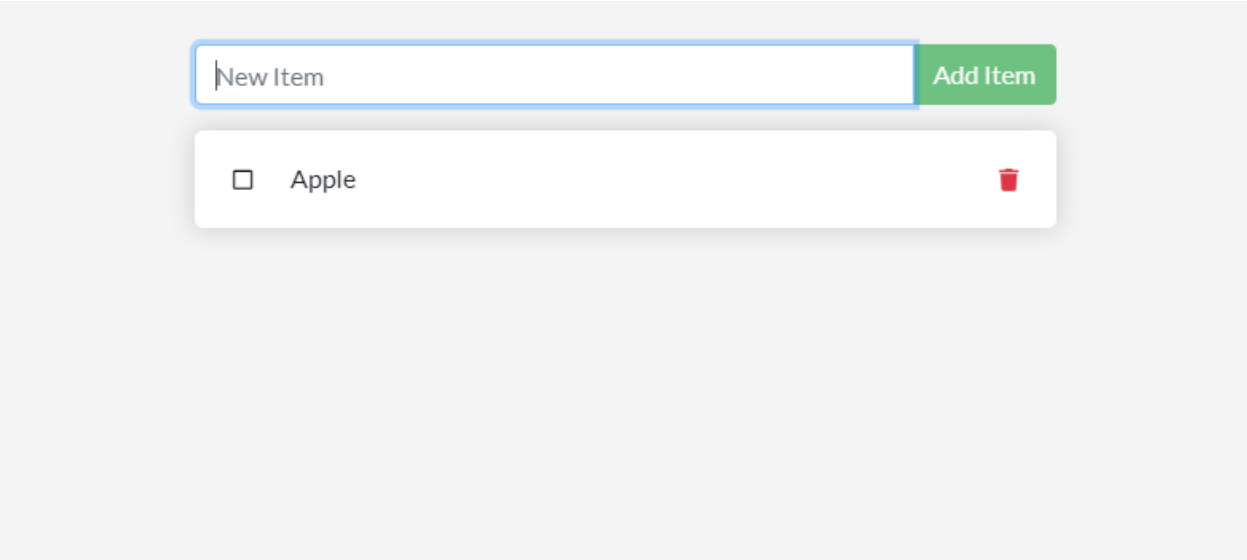
6. ทำการ Start ตัว Container ของแอปพลิเคชันที่สร้างขึ้น โดยใช้คำสั่ง

   $ 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 D:\softwareen\Lab8_4\getting-started\app> docker run -dp 3000:3000 myapp_6533801430
45a78ac56fde9b5eee0eadd039e08b6bd93d2f3df47ba07ba2997eb98074a6d7
PS D:\softwareen\Lab8_4\getting-started\app> docker ps
CONTAINER ID   IMAGE                  COMMAND                  CREATED              STATUS              PORTS
               NAMES
45a78ac56fde   myapp_6533801430       "docker-entrypoint.s…"   13 seconds ago       Up 12 seconds       0.0.0.
```



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

```
            {items.length === 0 && (
                <p className="text-center">There is no TODO item. Please add one to the
list. By Peeraphong Poncha</p>
            )}
```

```
PS D:\softwareen\Lab8_4\getting-started\app> docker build -t myapp_6533801430 .
[+] Building 27.3s (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                      2.0s
 => [auth] library/node:pull token for registry-1.docker.io                            0.0s
 => [internal] load .dockerignore                                                      0.0s
 => => transferring context: 2B                                                        0.0s
 => [1/4] FROM docker.io/library/node:18-alpine@sha256:974afb6cbc0314dc6502b14243b8a39  0.0s
 => => resolve docker.io/library/node:18-alpine@sha256:974afb6cbc0314dc6502b14243b8a39  0.0s
 => [internal] load build context                                                      0.0s
 => => transferring context: 8.09kB                                                    0.0s
```

```
PS D:\softwareen\Lab8_4\getting-started\app> docker run -dp 3000:3000 myapp_6533801430
505c4aed1f8126876bca9df3dbb59ff23e7f4da786b08c7d82e4193049639035
docker: Error response from daemon: driver failed programming external connectivity on endpoint xenodochial_ba
rtik (d7f64256b0ad88e6097b1b0c772ba374cfaa640ae1a6ad667acf0034b7a34864): Bind for 0.0.0.0:3000 failed: port is
 already allocated.
```

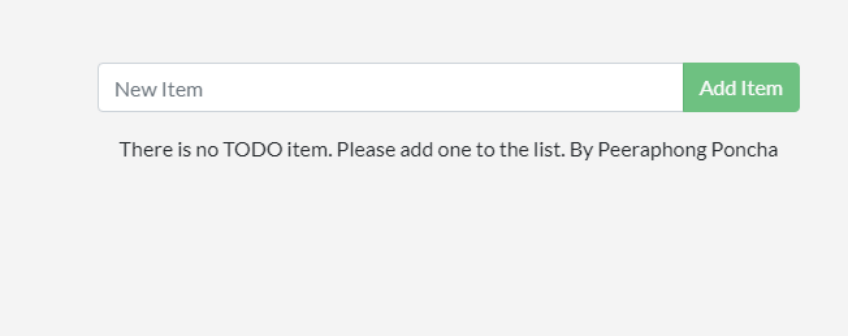
(1) Error ที่เกิดขึ้นหมายความอย่างไร และเกิดขึ้นเพราะอะไร

หมายความว่า Docker ไม่สามารถ map port 3000 บนเครื่องได้ เนื่องจากถูกใช้ไปแล้ว เกิดขึ้นเพราะ run บน port เดียวกันที่เคยใช้ไปแล้ว ใช้ port ซ้ำ

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 D:\softwareen\Lab8_4\getting-started\app> docker ps
CONTAINER ID   IMAGE                  COMMAND                  CREATED         STATUS         PORTS
            NAMES
45a78ac56fde   8b5d2db6df26           "docker-entrypoint.s…"   19 minutes ago  Up 19 minutes  0.0.0.0:3006
->3000/tcp   sharp_mestorf
0aeb401c5601   peeraphong/pphtml:v0.1   "/docker-entrypoint.…"   2 hours ago     Up 2 hours     80/tcp
            hello.2.o8abzd7d0bxzv2txqv6u0jf4w
99aef0b12147   peeraphong/pphtml:v0.1   "/docker-entrypoint.…"   2 hours ago     Up 2 hours     80/tcp
            hello.1.d1pxlhi6oc6lv3uvgpwn8rxja
PS D:\softwareen\Lab8_4\getting-started\app> docker stop 45a
45a
PS D:\softwareen\Lab8_4\getting-started\app> docker rm 45a
45a
PS D:\softwareen\Lab8_4\getting-started\app> docker run -dp 3000:3000 myapp_6533801430
7fece6bfe634e380cfe0bcd015bd08e401c541acd7c08f50f292cb4a76b54283
```

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

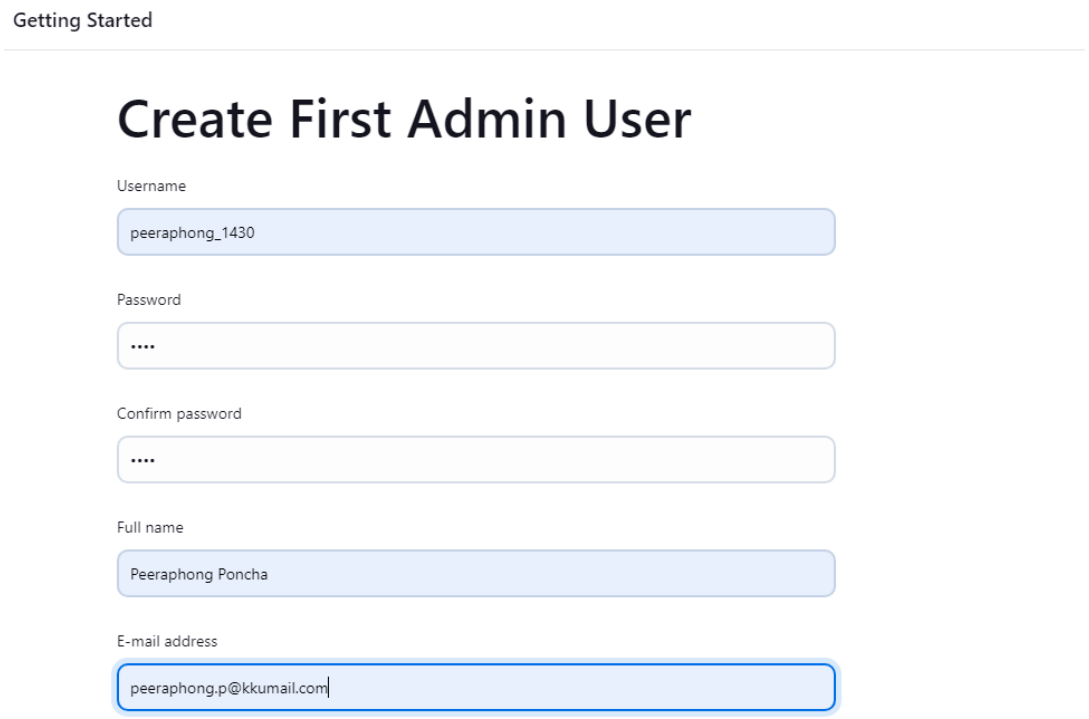
[Check point#12] Capture หน้าจอที่แสดงผล Admin password

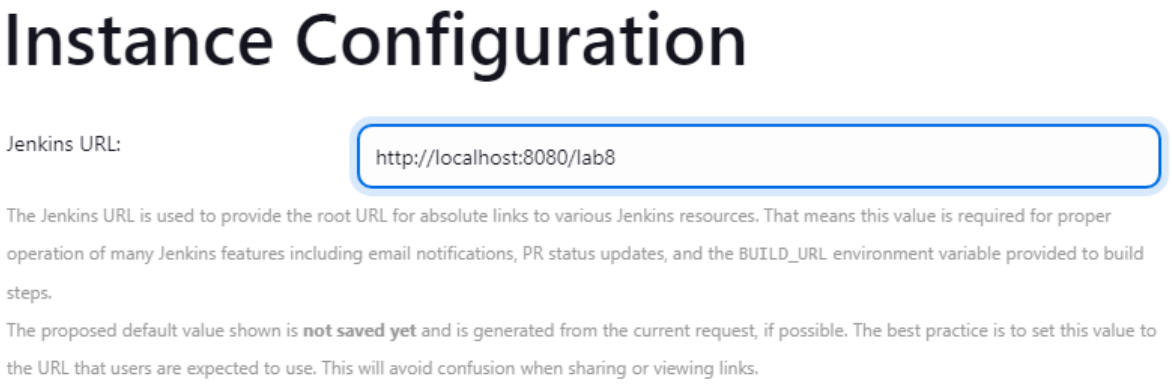
```
Jenkins initial setup is required. An admin user has been created and a password generated.
Please use the following password to proceed to installation:

988e0caa3219464aa4bffde9dbfd3ddd

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

4. เมื่อได้รับการยืนยันว่า Jenkins is fully up and running ให้เปิดบราวเซอร์ และป้อนที่อยู่เป็น

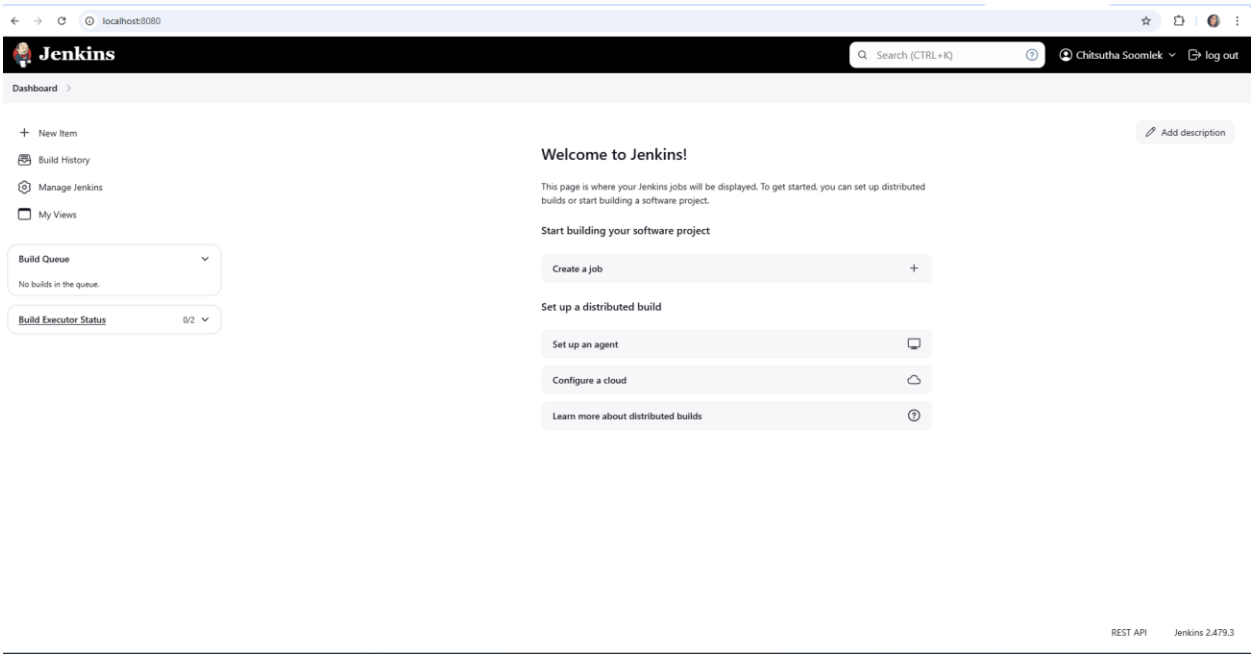
   localhost:8080
5. ทำการ Unlock Jenkins ด้วยรหัสผ่านที่ได้ในข้อที่ 3
6. สร้าง Admin User โดยใช้ username เป็นชื่อจริงของนักศึกษาพร้อมรหัสสี่ตัวท้าย เช่น somsri_3062

[Check point#13] Capture หน้าจอที่แสดงผลการตั้งค่า

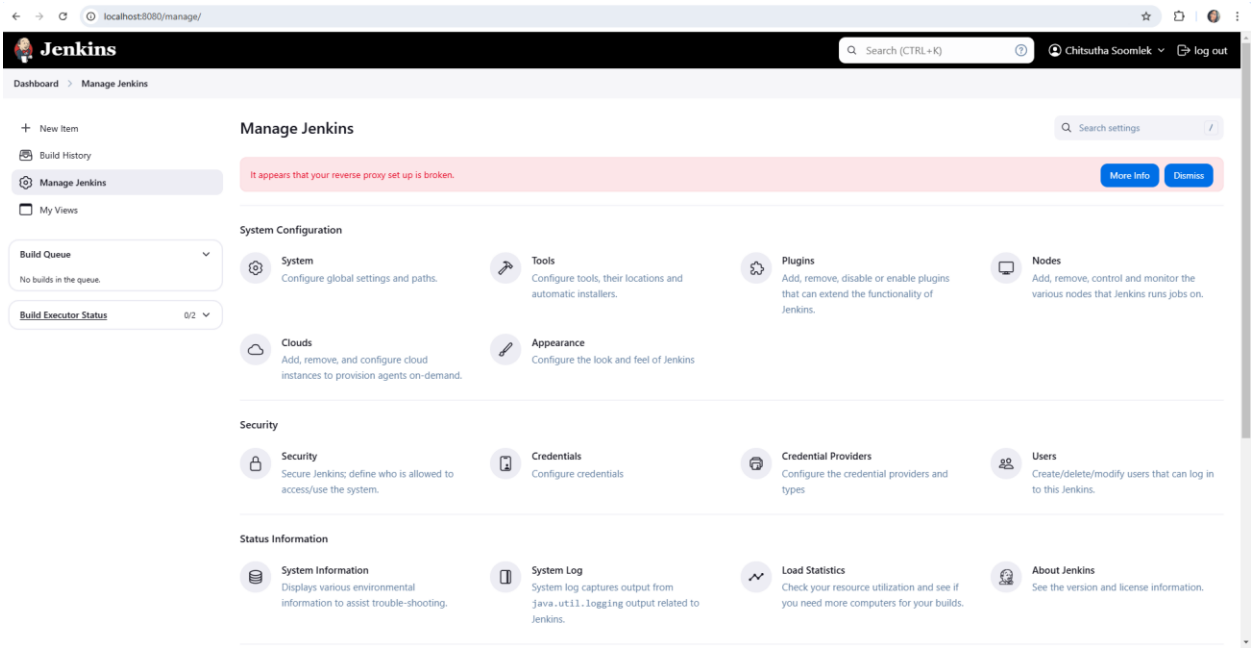




7. กำหนด Jenkins URL เป็น http://localhost:8080/lab8
8. เมื่อติดตั้งเรียบร้อยแล้วจะพบกันหน้า Dashboard ดังแสดงในภาพ

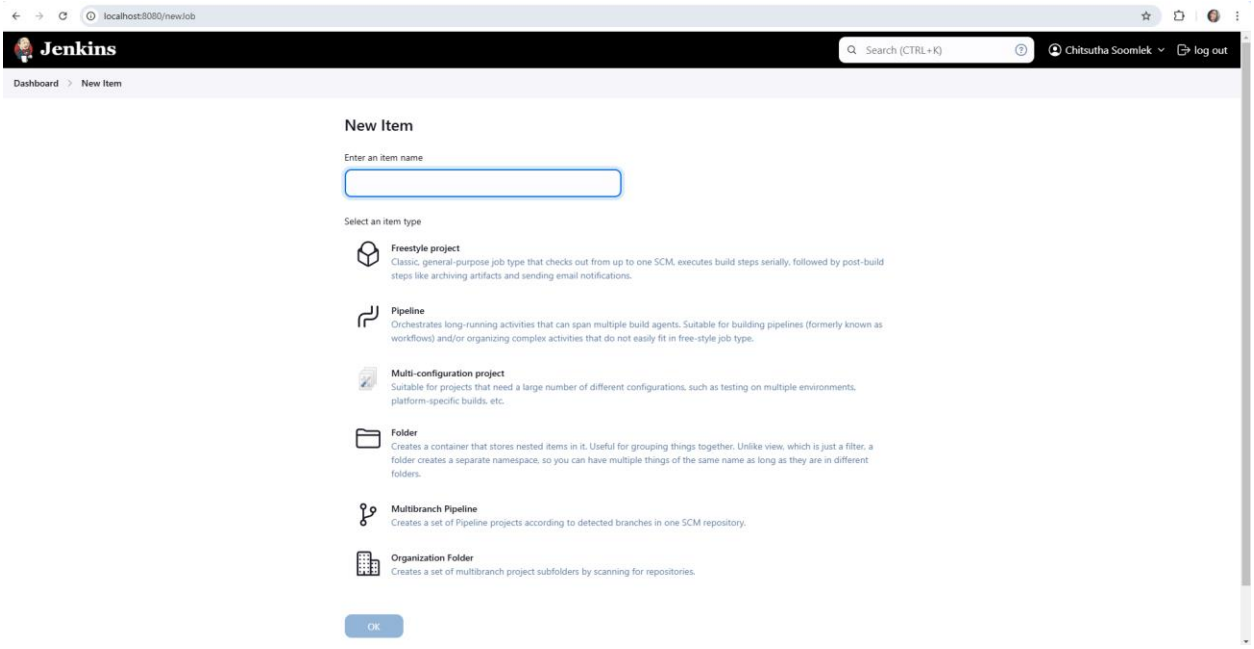


9. เลือก Manage Jenkins แล้วไปที่เมนู Plugins



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

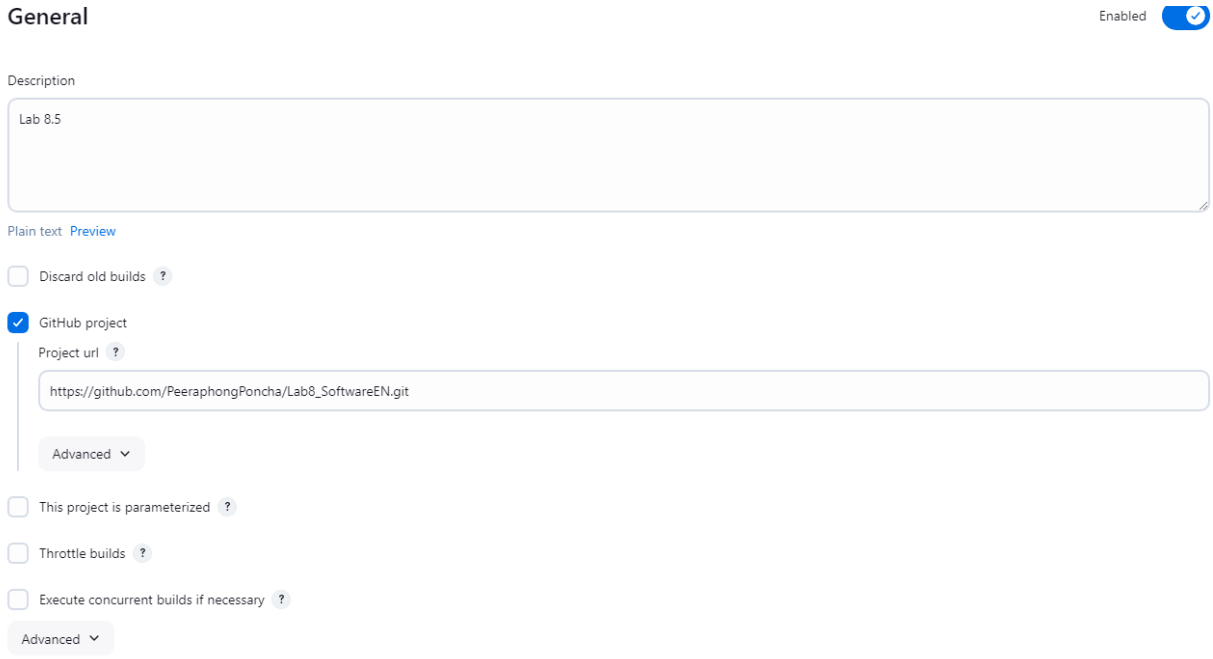
Description: Lab 8.5

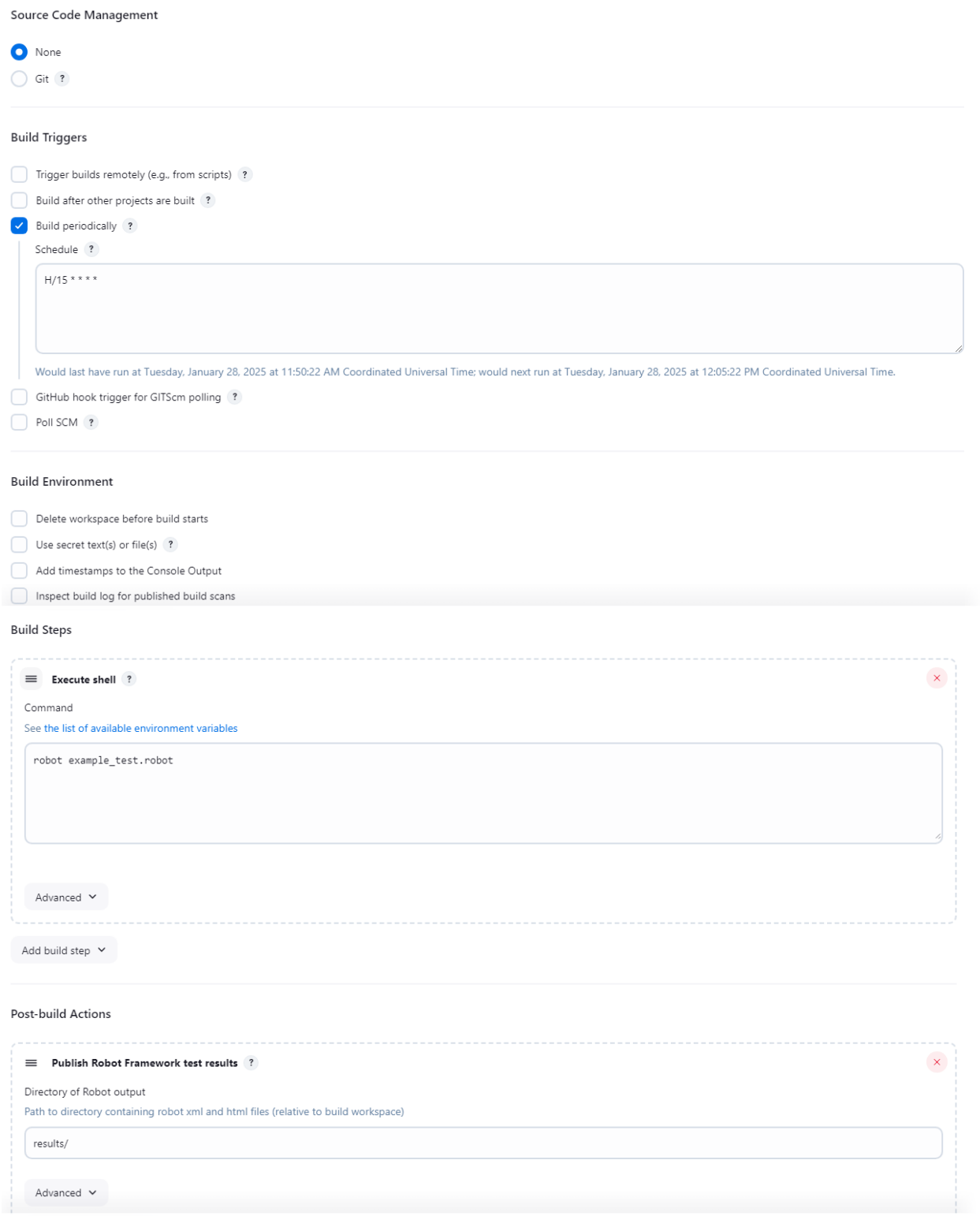
GitHub project: กดเลือก แล้วใส่ Project URL เป็น repository ที่เก็บโค้ด .robot (ดูขั้นตอนที่ 12)

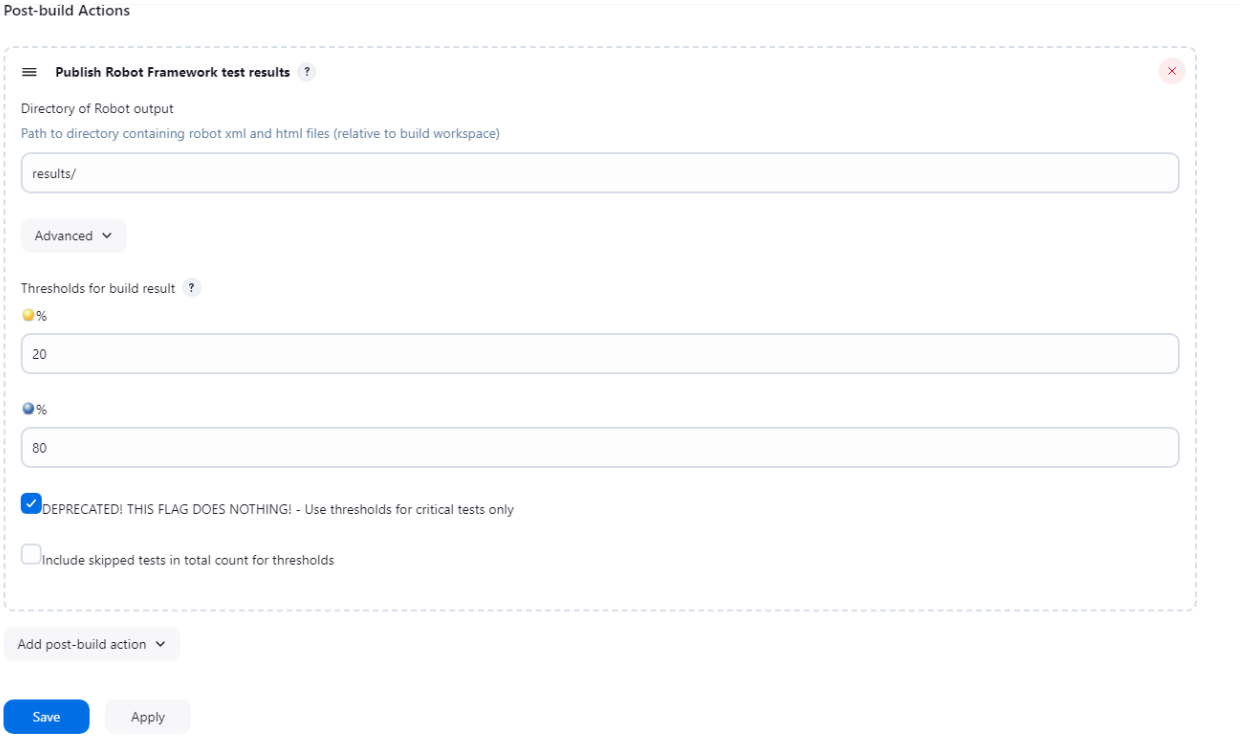
Build Trigger: เลือกแบบ Build periodically แล้วกำหนดให้ build ทุก 15 นาที

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

[Check point#14] Capture หน้าจอแสดงการตั้งค่า พร้อมกับตอบคำถามต่อไปนี้




Source Code Management
None
Git
Build Triggers
Trigger builds remotely (e.g., from scripts)
Build after other projects are built
Build periodically
Schedule
H/15 * * * *
Would last have run at Tuesday, January 28, 2025 at 11:50:22 AM Coordinated Universal Time; would next run at Tuesday, January 28, 2025 at 12:05:22 PM Coordinated Universal Time.
GitHub hook trigger for GITScm polling
Poll SCM
Build Environment
Delete workspace before build starts
Use secret text(s) or file(s)
Add timestamps to the Console Output
Inspect build log for published build scans
Build Steps
Execute shell
Command
See the list of available environment variables
robot example_test.robot
Advanced
Add build step
Post-build Actions
Publish Robot Framework test results
Directory of Robot output
Path to directory containing robot xml and html files (relative to build workspace)
results/
Advanced



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

robot example_test.robot

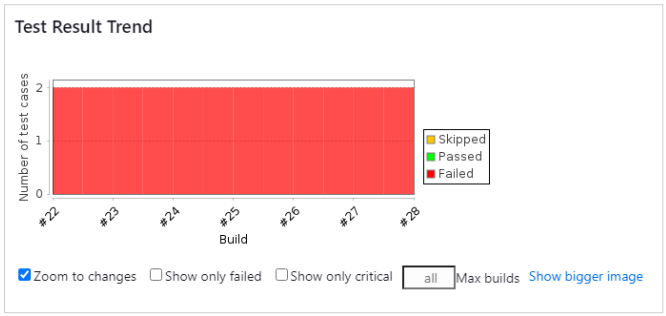
Post-build action: เพิ่ม Publish Robot Framework test results -> ระบุไดเร็คทอรีที่เก็บไฟล์ผลการทดสอบโดย Robot framework ในรูป xml และ html -> ตั้งค่า Threshold เป็น % ของการทดสอบที่ไม่ผ่านแล้วนับว่าซอฟต์แวร์มีปัญหา -> ตั้งค่า Threshold เป็น % ของการทดสอบที่ผ่านแล้วนับว่าซอฟต์แวร์มีอยู่ในสถานะที่สามารถนำไปใช้งานได้ (เช่น 20, 80)

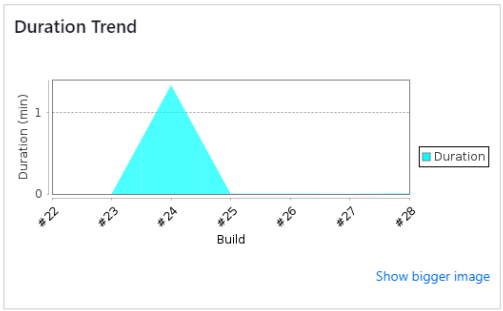
13. กด Apply และ Save
14. สั่ง Build Now

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

Robot Framework Test Results

Executed:	2025-01-28T13:36:01.014294
Duration:	0:00:00.427 (+0:00:00.259)
Status:	2 critical test, 0 passed, **2** failed, 0 skipped 2 test total (±0), 0 passed, **2** failed, 0 skipped
Results:	report.html log.html Original result files





Failed Test Cases

Name	Crit.	Duration	Age
Example Test.Test Example Page	no	0:00:00.069	7
Example Test.Test Search Google	no	0:00:00.025	7

Console Output

```
Started by user Peeraphong Poncha
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/PeeraphongPoncha/Lab8_SoftwareEN.git/ # timeout=10
Fetching upstream changes from https://github.com/PeeraphongPoncha/Lab8_SoftwareEN.git/
 > git --version # timeout=10
 > git --version # 'git version 2.39.5'
 > git fetch --tags --force --progress -- https://github.com/PeeraphongPoncha/Lab8_SoftwareEN.git/ +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision 29a7eec2a4c66618a81ef075d8ab83d90917237d (refs/remotes/origin/main)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f 29a7eec2a4c66618a81ef075d8ab83d90917237d # timeout=10
Commit message: "Delete README.md"
 > git rev-list --no-walk 29a7eec2a4c66618a81ef075d8ab83d90917237d # timeout=10
[UAT] $ /bin/sh -xe /tmp/jenkins13961099861280356203.sh
+ . /path/to/venv/bin/activate
+ deactivate nondestructive
+ [ -n  ]
+ [ -n  ]
+ [ -n  -o -n  ]
+ [ -n  ]
+ unset VIRTUAL_ENV
+ unset VIRTUAL_ENV_PROMPT
+ [ ! nondestructive = nondestructive ]
+ VIRTUAL_ENV=/path/to/venv
+ export VIRTUAL_ENV
+ PS1=(venv) $
+ export PS1
+ VIRTUAL_ENV_PROMPT=(venv)
+ export VIRTUAL_ENV_PROMPT
+ [ -n  -o -n  ]
+ robot --outputdir results example_test.robot
==============================================================================
Example Test
==============================================================================
Test Example Page :: เปิดหน้า Example.com และตรวจสอบว่ามีข้อความ E... | FAIL |
WebDriverException: Message: Service /var/jenkins_home/.cache/selenium/chromedriver/linux64/132.0.6834.110/chromedriver unexpectedly exited. Status code was: 127
------------------------------------------------------------------------------
Test Search Google :: ทดสอบการค้นหาบน Google                          | FAIL |
WebDriverException: Message: Service /var/jenkins_home/.cache/selenium/chromedriver/linux64/132.0.6834.110/chromedriver unexpectedly exited. Status code was: 127
------------------------------------------------------------------------------
Example Test                                                          | FAIL |
2 tests, 0 passed, 2 failed
==============================================================================
Output:  /var/jenkins_home/workspace/UAT/results/output.xml
Log:     /var/jenkins_home/workspace/UAT/results/log.html
Report:  /var/jenkins_home/workspace/UAT/results/report.html
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
```