Lab Worksheet

ชื่อ-นามสกุล ธณภัทร ภูคงเดือน รหัสนักศึกษา 653380017-5 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

Lab Worksheet

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

(1) สิ่งที่อยู่ภายใต้คอลัมน์ Repository คืออะไร name of Repository

(2) Tag ที่ใช้บ่งบอกถึงอะไร version of images

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

6 – 9

```
PS E:\Software_en\Lab8_1> docker run busybox
PS E:\Software_en\Lab8_1> docker run -it busybox sh
/ # ls
bin    etc    lib    proc   sys    usr
dev    home   lib64  root   tmp    var
/ # ls -la
lrwxrwxrwx    1 root     root             3 Sep 26 21:31 lib64
-> lib
dr-xr-xr-x  212 root     root             0 Jan 27 06:36 proc
drwx------    1 root     root          4096 Jan 27 06:36 root
dr-xr-xr-x   11 root     root             0 Jan 27 06:36 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
```

10-11

```
PS E:\Software_en\Lab8_1> docker run busybox echo "Hello Thanap
hat Pookongdun from busybox"
Hello Thanaphat Pookongdun from busybox
PS E:\Software_en\Lab8_1> docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED
        STATUS                              PORTS     NAMES
a0a4328b683e   busybox   "echo 'Hello Thanaph…"   10 seconds ag
o       Exited (0) 10 seconds ago                     kind_solomon
edd37a61891e   busybox   "sh"                     About a minut
e ago   Exited (0) About a minute ago               dazzling_shte
rn
8541cf0689be   busybox   "sh"                     About a minut
e ago   Exited (0) About a minute ago               beautiful_mes
torf
```

(1) เมื่อใช้ option -it ในคำสั่ง run ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป
-it จะทำให้สามารถเปิด shell ที่ใช้งานแบบ interactive ได้

(2) คอลัมน์ STATUS จากการรันคำสั่ง docker ps -a แสดงถึงข้อมูลอะไร
ใช้เพื่อแสดงรายการของคอนเทนเนอร์ทั้งหมดบนเครื่อง

12. ป้อนคำสั่ง $ docker rm <container ID ที่ต้องการลบ>

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

```
PS E:\Software_en\Lab8_1> docker rm a0a4328b683e
a0a4328b683e
```

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

```
PS E:\Software_en\Lab8_2> docker build -t demo_image .
[+] Building 9.8s (6/6) FINISHED          docker:desktop-linux
 => [internal] load build definition from Dockerfile      0.0s
 => => transferring dockerfile: 154B                      0.0s
 => [internal] load metadata for docker.io/library/busyb  0.0s
 => [internal] load .dockerignore                         0.1s
 => => transferring context: 2B                           0.0s
 => [1/1] FROM docker.io/library/busybox:latest@sha256:a  9.3s
 => => exporting config sha256:61ba23e6529e2e9ec568af1b8   0.0s
 => => exporting attestation manifest sha256:1c1668c7ac6  0.1s
 => => exporting manifest list sha256:d88a42b72df95eb9fb   0.0s
 => => naming to docker.io/library/demo_image:latest      0.0s
 => => unpacking to docker.io/library/demo_image:latest   0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/n7wxp4h2xv0b9hn2g2n4j9u1a

 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
)
 - JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 3)
PS E:\Software_en\Lab8_2> docker run demo_image
Thanaphat Pookongduen 653380017-5 Sigco
```

(1) คำสั่งที่ใช้ในการ run คือ

docker run demo_image

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

ช่วยให้เราตั้งชื่อและแท็กให้กับ Image ที่สร้าง

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 E:\Software_en\Lab8\Lab8_3> docker build -t sigco/lab8 .
[+] Building 4.1s (6/6) FINISHED          docker:desktop-linux
 => [internal] load build definition from Dockerfile      0.1s
 => => transferring dockerfile: 185B                      0.0s
 => [internal] load metadata for docker.io/library/busyb  0.0s
 => [internal] load .dockerignore                         0.1s
 => => transferring context: 2B                           0.0s
 => CACHED [1/1] FROM docker.io/library/busybox:latest@s  3.5s
 => => resolve docker.io/library/busybox:latest@sha256:a  3.5s
 => [auth] library/busybox:pull token for registry-1.doc  0.0s
 => => exporting attestation manifest sha256:dc553b8ddb5  0.0s
 => => exporting manifest list sha256:24bd3ceafb68e6b220  0.0s
 => => naming to docker.io/sigco/lab8:latest              0.0s
 => => unpacking to docker.io/sigco/lab8:latest           0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/zraq1r54lpbf6oaasfj66x70s

 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
)
 - JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 3)
PS E:\Software_en\Lab8\Lab8_3> docker run sigco/lab8
>>
"Thanaphat Pookongduen 653380017-5"
```

6. ทำการ Push ตัว Docker image ไปไว้บน Docker Hub โดยการใช้คำสั่ง

   $ docker push <username ที่ลงทะเบียนกับ Docker Hub>/lab8

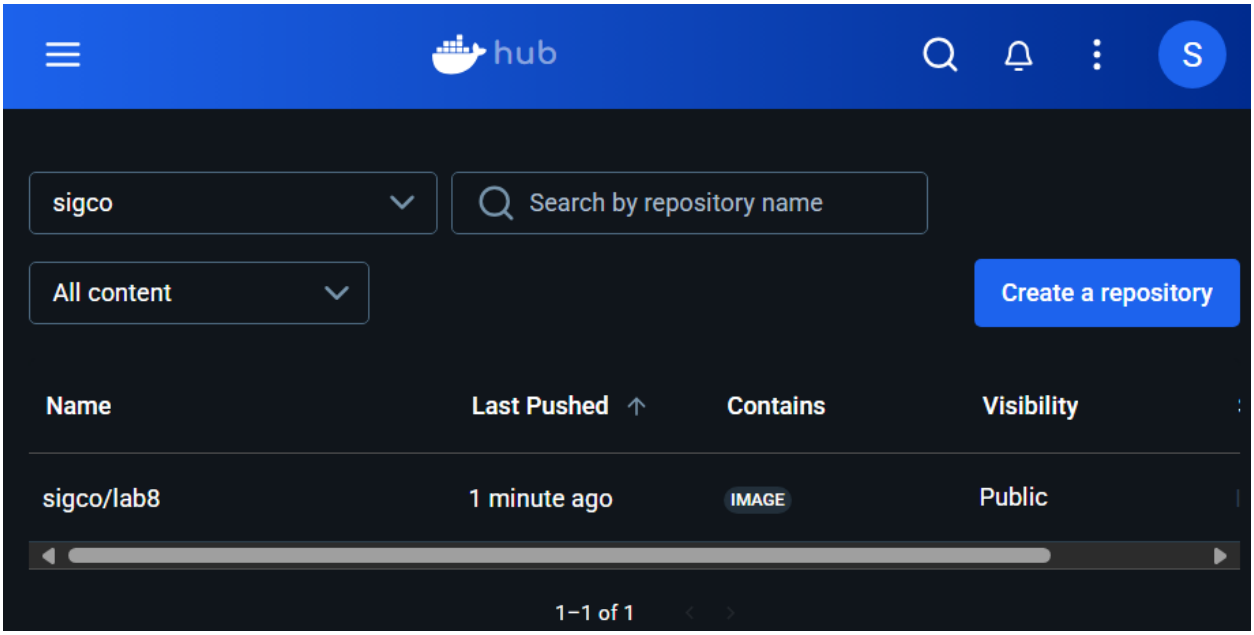
   ในกรณีที่ติดปัญหาไม่ได้ Login ไว้ก่อน ให้ใช้คำสั่งต่อไปนี้ เพื่อ Login ก่อนทำการ Push

   $ docker login แล้วป้อน Username และ Password ตามที่ระบุใน Command prompt หรือใช้คำสั่ง

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

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

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



แบบฝึกปฏิบัติที่ 8.4: การ Build แอปพลิเคชันจาก Container image และการ Update แอปพลิเคชัน

1. เปิด Command line หรือ Terminal จากนั้นสร้าง Directory ชื่อ Lab8_4
2. ทำการ Clone ซอร์สโค้ดของเว็บแอปพลิเคชันจาก GitHub repository https://github.com/docker/getting-started.git ลงใน Directory ที่สร้างขึ้น โดยใช้คำสั่ง
   $ git clone https://github.com/docker/getting-started.git
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

```
ker/getting-started.git
Cloning into 'getting-started'...
remote: Enumerating objects: 980, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 980 (delta 5), reused 1 (delta 1), pack-reused 971 (from 2)
Receiving objects: 100% (980/980), 5.28 MiB | 7.47 MiB/s, done.Resolving deltas: 100% (523/523), done.
```

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

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 E:\Software_en\Lab8\Lab8_4\getting-started> cd .\app\       PS E:\Software_en\Lab8\Lab8_4\getting-started\app> docker build -t myapp_65338001
75 .
[+] Building 33.6s (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                                                                    7.5s
 => [auth] library/node:pull token for registry-1.docker.io                                                                          0.0s
 => [internal] load .dockerignore                                                                                                    0.1s
 => => transferring context: 2B                                                                                                      0.0s
 => [1/4] FROM docker.io/library/node:18-alpine@sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d975e9059dd066be3e274fbb25               7.3s
 => => resolve docker.io/library/node:18-alpine@sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d975e9059dd066be3e274fbb25               0.0s
 => => sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce368990ca771 444B / 444B                                           0.4s
 => => sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15bf426d01bc1 1.26MB / 1.26MB                                       0.8s
 => => sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e0da02525e63167c 40.01MB / 40.01MB                                     6.0s
 => => sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c24403d592134f0 3.64MB / 3.64MB                                       1.8s
 => => extracting sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c24403d592134f0                                            0.2s
 => => extracting sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e0da02525e63167c                                            0.9s
 => => extracting sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15bf426d01bc1                                            0.1s
 => => extracting sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce368990ca771                                            0.0s
 => [internal] load build context                                                                                                    1.0s
 => => transferring context: 4.62MB                                                                                                  0.9s
 => [2/4] WORKDIR /app                                                                                                               0.1s
 => [3/4] COPY . .                                                                                                                   0.1s
 => [4/4] RUN yarn install --production                                                                                             12.4s
 => exporting to image                                                                                                               5.7s
 => => exporting layers                                                                                                              3.7s
 => => exporting manifest sha256:b8305faa4234c182b9f9415dee401192b40ba8d91cc5fa73442d1cd2c618f5e0                                     0.0s
 => => exporting config sha256:df68b1dfab2a8d015803a2b2e29112eea73ee5bccb1853f7c79e8dae8b910912                                       0.0s
 => => exporting attestation manifest sha256:2518aeec710e3b992be88f4f4a96ac8734e530f912b0579d5960a5ba7dfc3101                        0.0s
 => => exporting manifest list sha256:120db6c666ef17f080a1c1b0d000bbac20bb6fc7fdae3f5d971ac69d94ec9b1c                               0.0s
 => => naming to docker.io/library/myapp_6533800175:latest                                                                           0.0s
 => => unpacking to docker.io/library/myapp_6533800175:latest                                                                        1.8s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/uoxbi79muqgm8wvnqi5xqn9zx
```

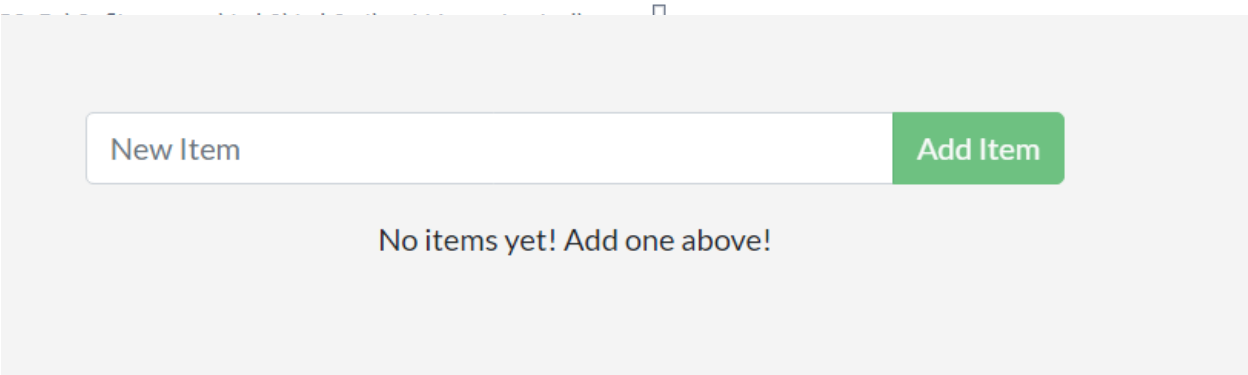
6. ทำการ Start ตัว Container ของแอปพลิเคชันที่สร้างขึ้น โดยใช้คำสั่ง

   $ 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 E:\Software_en\Lab8\Lab8_4\getting-started\app> docker run -dp 3000:3000 myapp_6533800175
14a8e5683258ba0fb735bab6ab0088f340267b9efd08e9be48e29cfc2cc00588
```



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

```
<p className="text-center">There is no TODO item. Please add one to the list. by Fullname and Student-ID</p>
PS E:\Software_en\Lab8\Lab8_4\getting-started> cd .\app\       PS E:\Software_en\Lab8\Lab8_4\getting-started\app> docker build -t myapp_65338001
75 .
[+] Building 33.6s (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                                                                    7.5s
 => [auth] library/node:pull token for registry-1.docker.io                                                                          0.0s
 => [internal] load .dockerignore                                                                                                    0.1s
 => => transferring context: 2B                                                                                                      0.0s
 => [1/4] FROM docker.io/library/node:18-alpine@sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d975e9059dd066be3e274fbb25               7.3s
 => => resolve docker.io/library/node:18-alpine@sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d975e9059dd066be3e274fbb25              0.0s
 => => sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce368990ca771 444B / 444B                                           0.4s
 => => sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15bf426d01bc1 1.26MB / 1.26MB                                       0.8s
 => => sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e0da02525e63167c 40.01MB / 40.01MB                                     6.0s
 => => sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c24403d592134f0 3.64MB / 3.64MB                                       1.8s
 => => extracting sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c24403d592134f0                                            0.2s
 => => extracting sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e0da02525e63167c                                            0.9s
 => => extracting sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15bf426d01bc1                                            0.1s
 => => extracting sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce368990ca771                                            0.0s
 => [internal] load build context                                                                                                    1.0s
 => => exporting manifest list sha256:a1acc45cd790dc9cf6bf48930290bedd3dc0f5f796296189052c8bab152bf3d0                               0.0s
 => => naming to docker.io/library/myapp_6533800175:latest                                                                           0.0s
 => => unpacking to docker.io/library/myapp_6533800175:latest                                                                        1.9s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/624o9q2qi3f2eejb5hskxfuu2
PS E:\Software_en\Lab8\Lab8_4\getting-started\app> docker run -dp 3000:3000 myapp_6533800175
549e87bb7c9a47f949d04ee3b662bd8e81f7d40cd942aceb543351a4db4a7739
docker: Error response from daemon: driver failed programming external connectivity on endpoint clever_neumann (ea137a5dba3ef28fd9a29058d3c6d7e6
841d6d4b6a8d3864a621d7095a8668a0): Bind for 0.0.0.0:3000 failed: port is already allocated.
```

(1) Error ที่เกิดขึ้นหมายความอย่างไร และเกิดขึ้นเพราะอะไร

ตัวเดิมยังไม่ได้ stop จึงทำให้ตัวใหม่ขอ port ไม่ได้

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

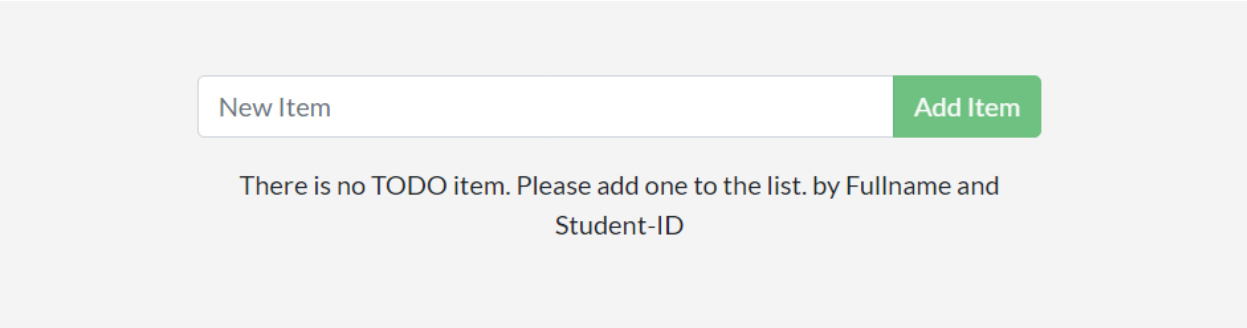
iii. ยืนยันโดยการกด Delete forever

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 E:\Software_en\Lab8\Lab8_4\getting-started\app> docker stop 549e87bb7c9a
549e87bb7c9a
PS E:\Software_en\Lab8\Lab8_4\getting-started\app> docker rm 549e87bb7c9a
549e87bb7c9a
PS E:\Software_en\Lab8\Lab8_4\getting-started\app> docker run -dp 3000:3000 myapp_6533800175
20f24cd8a58549657b10c2576ba5dc554b0ea36fe1244699940173d051115655
```



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

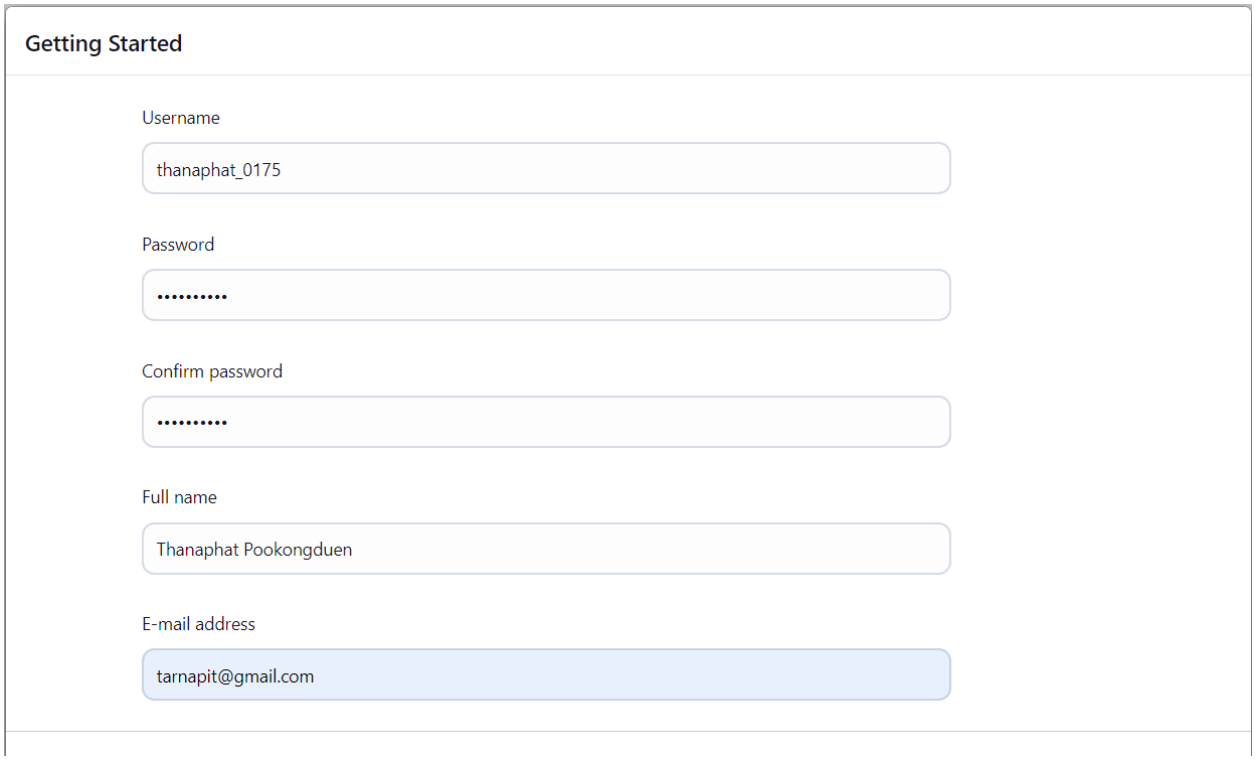
   หรือ

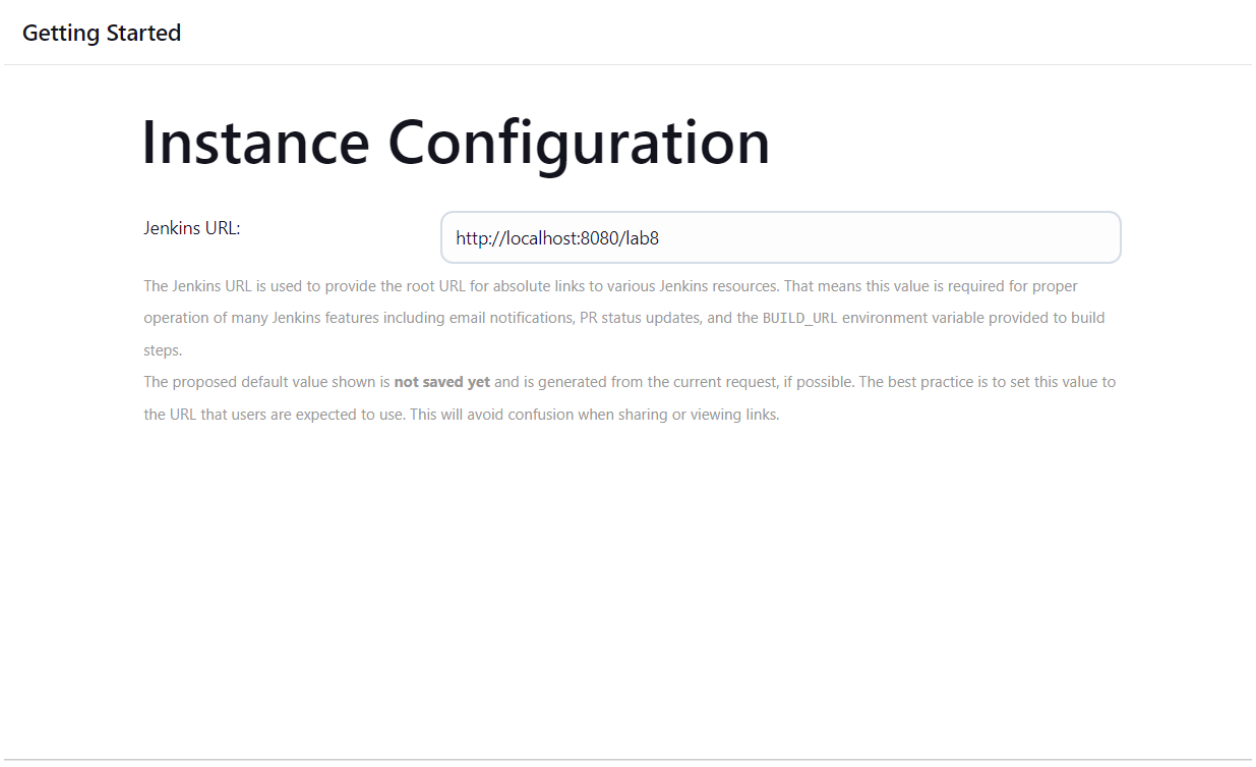
   $ docker run -p 8080:8080 -p 50000:50000 --restart=on-failure -v jenkins_home:/var/jenkins_home jenkins/jenkins:lts-jdk17
3. บันทึกรหัสผ่านของ Admin user ไว้สำหรับ log-in ในครั้งแรก

[Check point#12] Capture หน้าจอที่แสดงผล Admin password

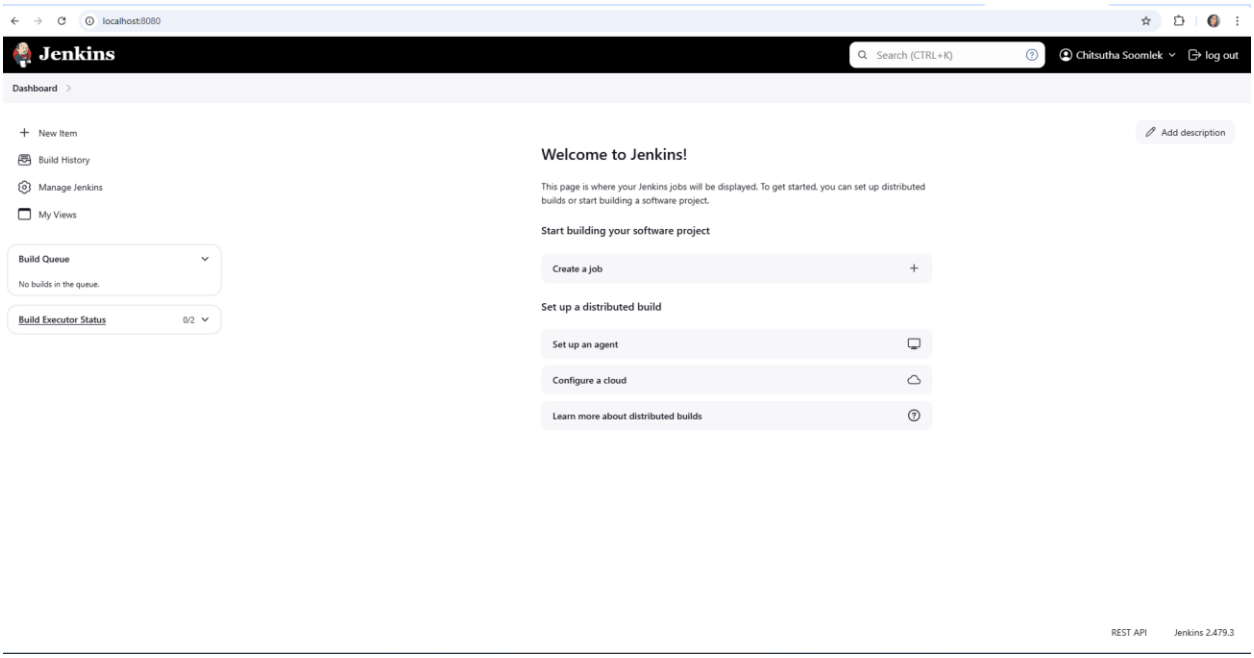
4. เมื่อได้รับการยืนยันว่า Jenkins is fully up and running ให้เปิดบราวเซอร์ และป้อนที่อยู่เป็น localhost:8080
5. ทำการ Unlock Jenkins ด้วยรหัสผ่านที่ได้ในข้อที่ 3
6. สร้าง Admin User โดยใช้ username เป็นชื่อจริงของนักศึกษาพร้อมรหัสสี่ตัวท้าย เช่น somsri_3062

[Check point#13] Capture หน้าจอที่แสดงผลการตั้งค่า

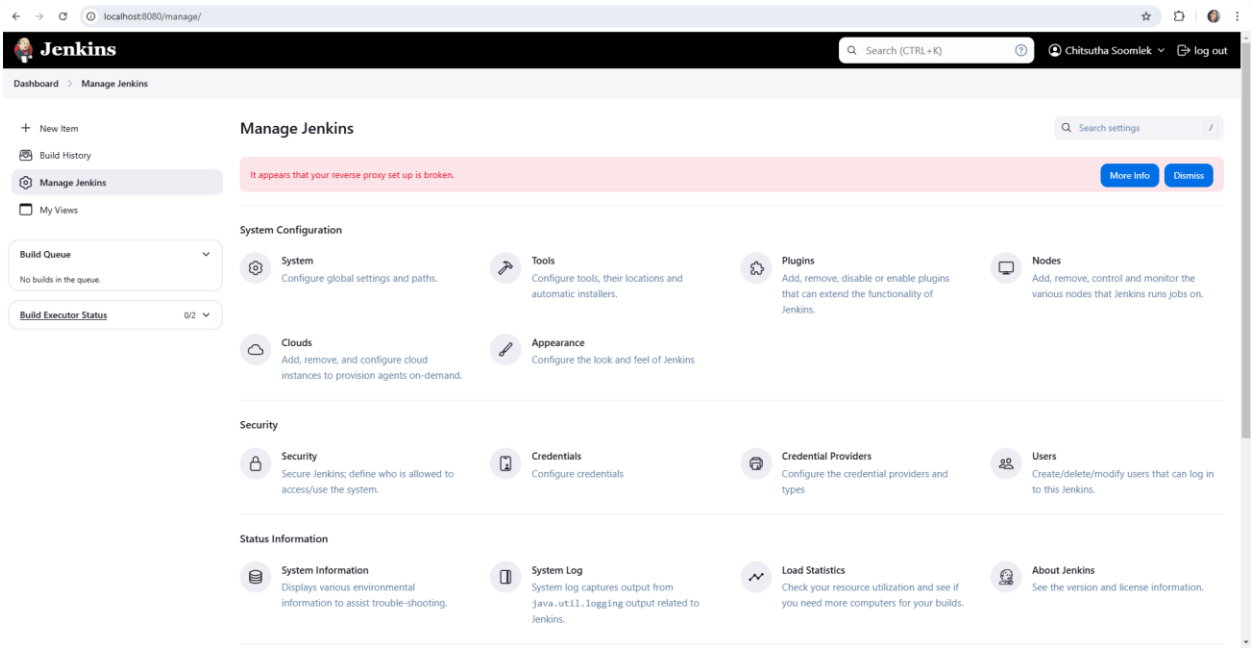




7. กำหนด Jenkins URL เป็น http://localhost:8080/lab8
8. เมื่อติดตั้งเรียบร้อยแล้วจะพบกันหน้า Dashboard ดังแสดงในภาพ

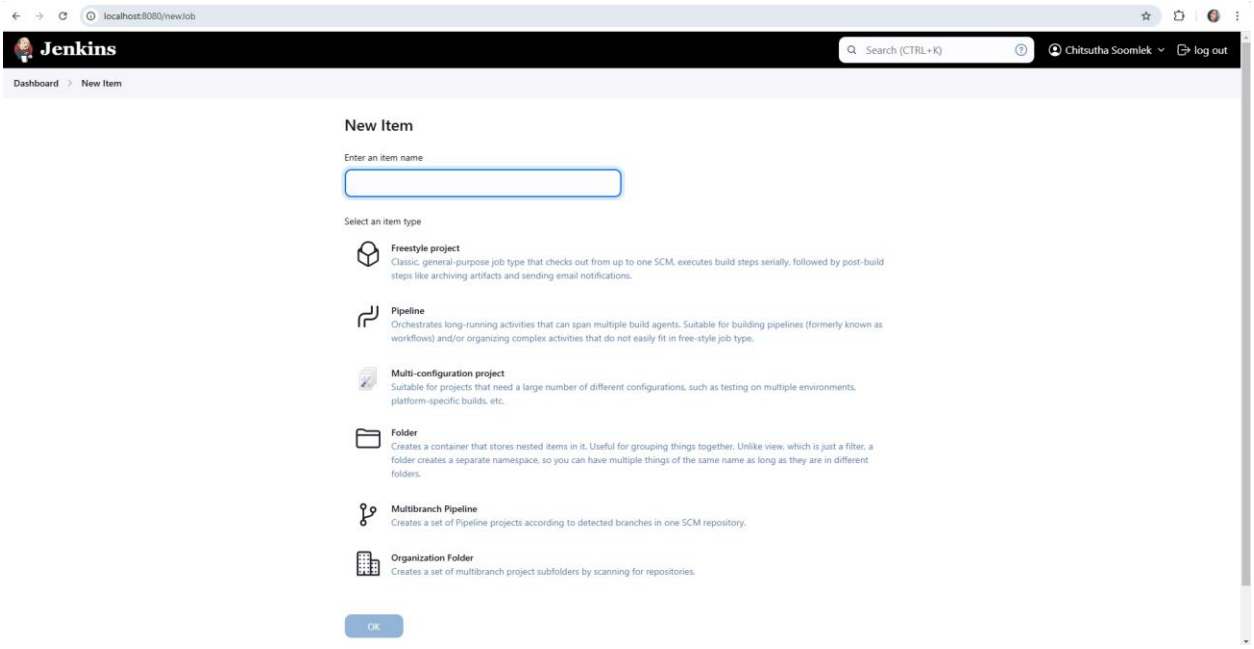


9. เลือก Manage Jenkins แล้วไปที่เมนู Plugins



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

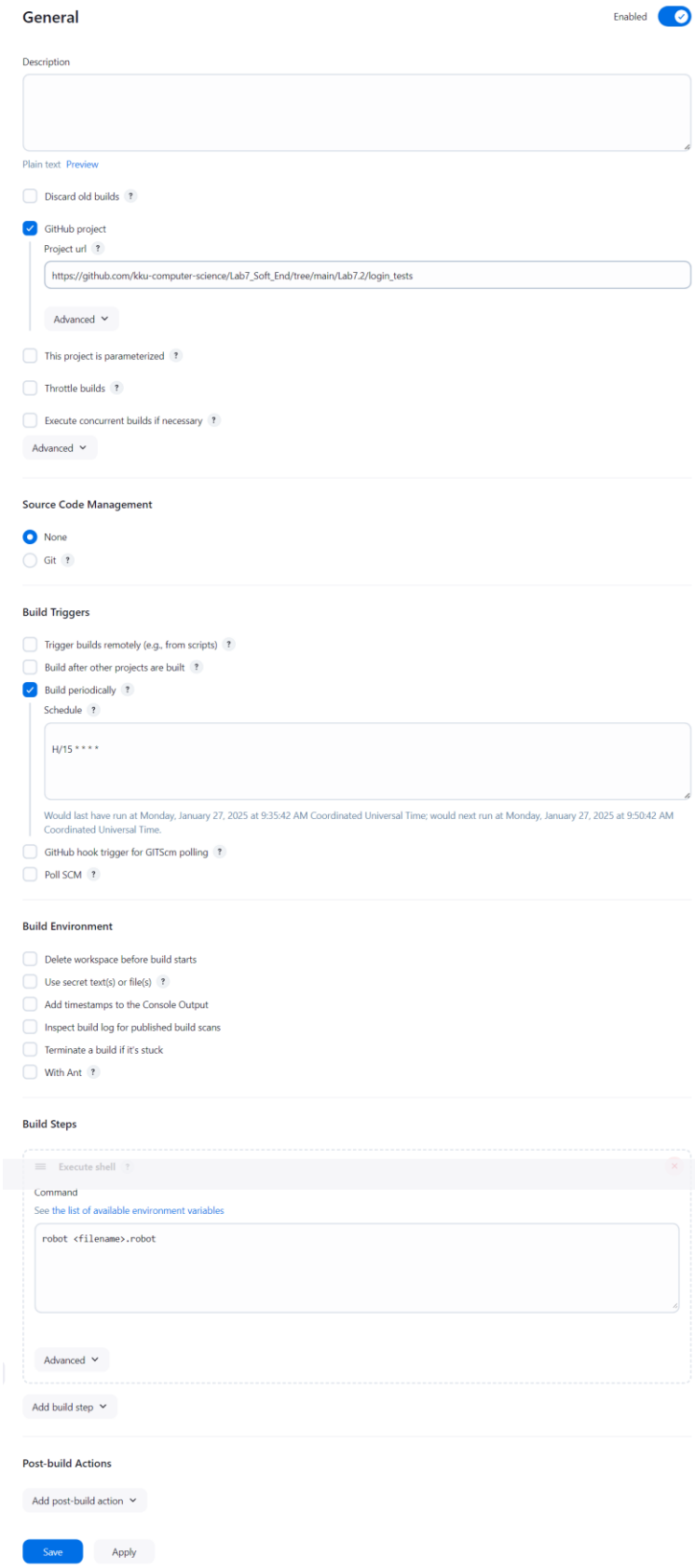
GitHub project: กดเลือก แล้วใส่ Project URL เป็น repository ที่เก็บโค้ด .robot (ดูขั้นตอนที่ 12)

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

Lab Worksheet



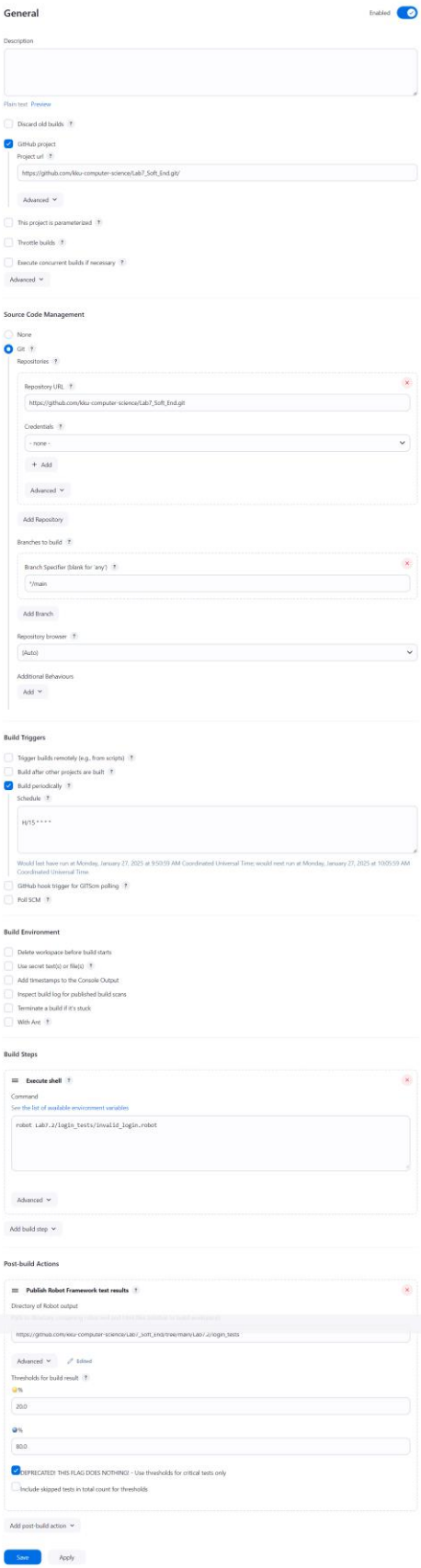
(1) คำสั่งที่ใช้ในการ Execute ไฟล์ .robot ใน Build Steps คือ

robot <filename>.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


General
Enabled
Description
Plain text Preview
Discard old builds
GitHub project
Project url
https://github.com/kku-computer-science/Lab7_Soft_End.git/
Advanced
This project is parameterized
Throttle builds
Execute concurrent builds if necessary
Advanced
Source Code Management
None
Git
Repositories
Repository URL
https://github.com/kku-computer-science/Lab7_Soft_End.git
Credentials
- none -
+ Add
Advanced
Add Repository
Branches to build
Branch Specifier (blank for 'any')
*/main
Add Branch
Repository browser
(Auto)
Additional Behaviours
Add
Build Triggers
Trigger builds remotely (e.g., from scripts)
Build after other projects are built
Build periodically
Schedule
H/15 * * * *
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
Terminate a build if it's stuck
With Ant
Build Steps
Execute shell
Command
See the list of available environment variables
robot Lab7.2/login_tests/invalid_login.robot
Advanced
Add build step
Post-build Actions
Publish Robot Framework test results
Directory of Robot output
Advanced
Edited
Thresholds for build result
20.0
80.0
DEPRECATED! THIS FLAG DOES NOTHING! - Use thresholds for critical tests only
Include skipped tests in total count for thresholds
Add post-build action
Save
Apply

Console Output

Download Copy View as plain text

```
Started by user Thanaphat Pookongduen
Started by user Thanaphat Pookongduen
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/kku-computer-science/Lab7_Soft_End.git # timeout=10
Fetching upstream changes from https://github.com/kku-computer-science/Lab7_Soft_End.git
 > git --version # timeout=10
 > git --version # 'git version 2.39.5'
 > git fetch --tags --force --progress -- https://github.com/kku-computer-science/Lab7_Soft_End.git +refs/heads/*:refs/remotes/origin/* #
timeout=10
 > git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision f71b0cd7e35b620741de61aecca04ce7e6e1237f (refs/remotes/origin/main)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f f71b0cd7e35b620741de61aecca04ce7e6e1237f # timeout=10
Commit message: "7.1 is demomo"
 > git rev-list --no-walk f71b0cd7e35b620741de61aecca04ce7e6e1237f # timeout=10
[UAT] $ /bin/sh -xe /tmp/jenkins4313624581443302604.sh
+ robot Lab7.2/login_tests/invalid_login.robot
/tmp/jenkins4313624581443302604.sh: 2: robot: not found
Build step 'Execute shell' marked build as failure
Robot results publisher started...
INFO: Checking test criticality is deprecated and will be dropped in a future release!
-Parsing output xml:
ERROR: Build step failed with exception
/var/jenkins_home/workspace/UAT/https:/github.com/kku-computer-science/Lab7_Soft_End/tree/main/Lab7.2/login_tests does not exist.
        at org.apache.tools.ant.types.AbstractFileSet.getDirectoryScanner(AbstractFileSet.java:512)
        at org.apache.tools.ant.types.AbstractFileSet.getDirectoryScanner(AbstractFileSet.java:489)
        at PluginClassLoader for robot//hudson.plugins.robot.RobotParser$RobotParserCallable.invoke(RobotParser.java:76)
        at PluginClassLoader for robot//hudson.plugins.robot.RobotParser$RobotParserCallable.invoke(RobotParser.java:52)
        at hudson.FilePath.act(FilePath.java:1234)
        at hudson.FilePath.act(FilePath.java:1217)
        at PluginClassLoader for robot//hudson.plugins.robot.RobotParser.parse(RobotParser.java:48)
        at PluginClassLoader for robot//hudson.plugins.robot.RobotPublisher.parse(RobotPublisher.java:262)
        at PluginClassLoader for robot//hudson.plugins.robot.RobotPublisher.perform(RobotPublisher.java:286)
        at hudson.tasks.BuildStepCompatibilityLayer.perform(BuildStepCompatibilityLayer.java:80)
        at hudson.tasks.BuildStepMonitor$1.perform(BuildStepMonitor.java:20)
        at hudson.model.AbstractBuild$AbstractBuildExecution.perform(AbstractBuild.java:818)
        at hudson.model.AbstractBuild$AbstractBuildExecution.performAllBuildSteps(AbstractBuild.java:767)
        at hudson.model.Build$BuildExecution.post2(Build.java:179)
        at hudson.model.AbstractBuild$AbstractBuildExecution.post(AbstractBuild.java:711)
        at hudson.model.Run.execute(Run.java:1854)
        at hudson.model.FreeStyleBuild.run(FreeStyleBuild.java:44)
        at hudson.model.ResourceController.execute(ResourceController.java:101)
        at hudson.model.Executor.run(Executor.java:445)
Build step 'Publish Robot Framework test results' marked build as failure
Finished: FAILURE
```