ชื่อ-นามสกุล นางสาวฐิดาพร เติมสุข รหัสนักศึกษา 653380015-9 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

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

```
mink@MacBook-Pro--Mink Lab8_1 % docker images
REPOSITORY                     TAG       IMAGE ID       CREATED         SIZE
docker/labs-vscode-installer   0.0.8     d03c76d8a03f   3 months ago    31.2MB
busybox                        latest    fc0179a204e2   3 months ago    4.04MB
docker/labs-ai-tools-for-devs  0.0.13    63b2d43ecaa9   4 months ago    14.4MB
synthesizedio/whalesay         latest    07da125a0bc8   6 months ago    45.2MB
docker/welcome-to-docker       latest    648f93a1ba7d   14 months ago   19MB
```

(1) สิ่งที่อยู่ภายใต้คอลัมน์ Repository คืออะไร

ตอบ คอลัมน์ Repository แสดงชื่อของ Docker image ที่ดึงมาจาก Docker Hub หรือที่ถูกสร้างขึ้น เช่น busybox คือชื่อของ image

(2) Tag ที่ใช้บ่งบอกถึงอะไร

ตอบ Tag ใช้สำหรับระบุเวอร์ชันหรือสถานะเฉพาะของ Docker image เช่น latest หมายถึงเวอร์ชันล่าสุดของ image นั้น หรืออาจใช้ tag อื่นๆ เพื่อแยกเวอร์ชัน เช่น 0.0.8

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

```
mink@MacBook-Pro--Mink Lab8_1 % docker run busybox
```

```
mink@MacBook-Pro--Mink Lab8_1 % docker run -it busybox sh
/ # ls
bin    dev    etc    home   lib    lib64  proc   root   sys    tmp    usr    var
/ # ls -la
total 48
drwxr-xr-x    1 root     root          4096 Jan 22 07:39 .
drwxr-xr-x    1 root     root          4096 Jan 22 07:39 ..
-rwxr-xr-x    1 root     root             0 Jan 22 07:39 .dockerenv
drwxr-xr-x    2 root     root         12288 Sep 26 21:31 bin
drwxr-xr-x    5 root     root           360 Jan 22 07:39 dev
drwxr-xr-x    1 root     root          4096 Jan 22 07:39 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  222 root     root             0 Jan 22 07:39 proc
drwx------    1 root     root          4096 Jan 22 07:39 root
dr-xr-xr-x   11 root     root             0 Jan 22 07:39 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
```

```
mink@MacBook-Pro--Mink Lab8_1 % docker run busybox echo "Hello Tidaporn Teamsuk from busybox"
Hello Tidaporn Teamsuk from busybox
mink@MacBook-Pro--Mink Lab8_1 % docker ps -a
CONTAINER ID   IMAGE                          COMMAND                  CREATED          STATUS                          PORTS
NAMES
2b319341759f   busybox                        "echo 'Hello Tidapor…"   7 seconds ago    Exited (0) 6 seconds ago
zen_tesla
56aaa0a3c973   busybox                        "sh"                     2 minutes ago    Exited (0) About a minute ago
bold_cohen
e2f7209912dc   busybox                        "sh"                     2 minutes ago    Exited (0) 2 minutes ago
cranky_raman
b780427585db   synthesizedio/whalesay         "/usr/local/bin/cows…"   55 minutes ago   Exited (0) 55 minutes ago
interesting_varahamihira
c41a16fecf51   docker/welcome-to-docker:latest   "/docker-entrypoint.…"   3 months ago     Exited (0) 3 months ago
welcome-to-docker
```

(1) เมื่อใช้ option -it ในคำสั่ง run ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป

ตอบ การใช้ -it เป็นการเปิดใช้งาน Interactive mode (-i) และ จำลอง terminal ช่วยให้โต้ตอบได้เหมือน terminal จริง (-t) (-it) จึงทำให้สามารถโต้ตอบกับ container ได้โดยตรง เช่น การใช้ shell (sh) หรือการรันคำสั่งใน container โดยให้ input และรับ output แบบ real-time ผ่าน terminal

(2) คอลัมน์ STATUS จากการรันคำสั่ง docker ps -a แสดงถึงข้อมูลอะไร

ตอบ คอลัมน์ STATUS แสดงสถานะของ container เช่น

- Up <time> หมายถึง container กำลังทำงานอยู่และระบุเวลาที่ container ทำงาน
- Exited (code) <time> หมายถึง container หยุดทำงานแล้ว โดย code คือ exit code ที่ระบุสถานะของการทำงานก่อนหยุด

12. ป้อนคำสั่ง $ docker rm <container ID ที่ต้องการลบ>

```
mink@MacBook-Pro--Mink ~ % docker rm c41a16fecf51
c41a16fecf51
```

[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

```
mink@MacBook-Pro--Mink ~ % cd desktop
mink@MacBook-Pro--Mink desktop % mkdir Lab8_2
mink@MacBook-Pro--Mink desktop % cd Lab8_2
mink@MacBook-Pro--Mink Lab8_2 % touch Dockerfile

mink@MacBook-Pro--Mink Lab8_2 % nano Dockerfile

mink@MacBook-Pro--Mink Lab8_2 % docker build -t myfirstimage .

[+] Building 0.0s (5/5) FINISHED                                                                                  docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                                              0.0s
 => => transferring dockerfile: 184B                                                                                              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   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
 => CACHED [1/1] FROM docker.io/library/busybox:latest                                                                            0.0s
 => exporting to image                                                                                                            0.0s
 => => exporting layers                                                                                                           0.0s
 => => writing image sha256:210c3c0786d9db433591ce6cf294c858e81b58c6c049204223f77944e9fc201f                                      0.0s
 => => naming to docker.io/library/myfirstimage                                                                                   0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/frqho6d9a2zx2ytu04a66koko

 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 u
sed (line 2)
 - JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 3)

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview
mink@MacBook-Pro--Mink Lab8_2 % docker run myfirstimage

Tidaporn Teamsuk 653380015-9 Mink
mink@MacBook-Pro--Mink Lab8_2 %
```

(1) คำสั่งที่ใช้ในการ run คือ

ตอบ $ docker run <image_name> ในที่นี้คือ $ docker run myfirstimage

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

ตอบ ใช้ในการตั้งชื่อ (tag) ให้กับ image ที่สร้างขึ้น ชื่อที่กำหนดจะช่วยให้สามารถอ้างอิง Docker image ได้ง่ายขึ้นในขั้นตอนต่อไป เช่น การ run, push, หรือ pull image หากไม่ใช้ -t Docker จะสร้าง image โดยไม่มีชื่อและให้ชื่อเป็นค่า hash โดยอัตโนมัติ ซึ่งทำให้การจัดการกับ image นั้นยากขึ้น

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

```
mink@MacBook-Pro--Mink ~ % cd desktop
mink@MacBook-Pro--Mink desktop % cd Lab8_3

mink@MacBook-Pro--Mink Lab8_3 % touch Dockerfile
mink@MacBook-Pro--Mink Lab8_3 % nano Dockerfile

mink@MacBook-Pro--Mink Lab8_3 % docker build -t tidaporn/lab8 .

[+] Building 0.0s (5/5) FINISHED                                                                                    docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                                                0.0s
 => => transferring dockerfile: 205B                                                                                                0.0s
 => WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 2      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                                                                              0.0s
 => exporting to image                                                                                                              0.0s
 => => exporting layers                                                                                                             0.0s
 => => writing image sha256:80e4ccea0321b4e570ba041a18c46f56aca7bb2fef2e219f79bb795e68b8603b                                        0.0s
 => => naming to docker.io/tidaporn/lab8                                                                                            0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/rm2vkl6m0kb16oufabk331tny

 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

mink@MacBook-Pro--Mink Lab8_3 % docker run tidaporn/lab8
Hi there. My work is done. You can run them from my Docker image.
Tidaporn Teamsuk 653380015-9
mink@MacBook-Pro--Mink 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

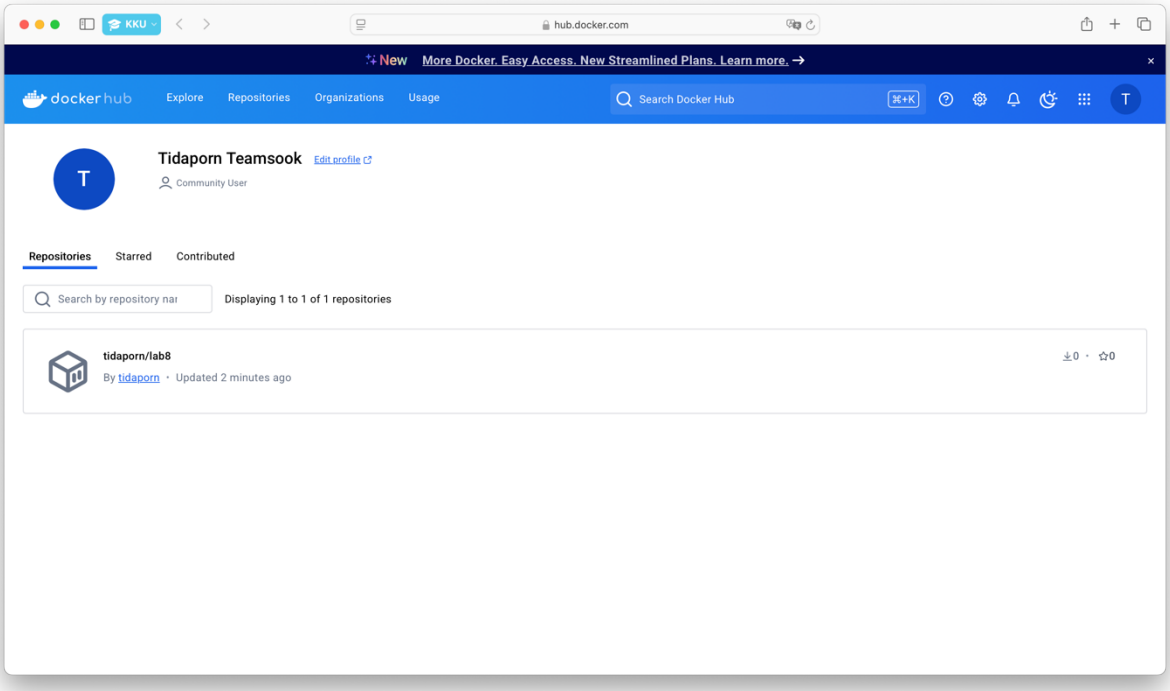
   หรือใช้คำสั่ง

   $ 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)

```
mink@MacBook-Pro--Mink Lab8_3 % docker push tidaporn/lab8
Using default tag: latest
The push refers to repository [docker.io/tidaporn/lab8]
613e5fc506b9: Mounted from library/busybox
latest: digest: sha256:14af8f95619d7b0821cf42758b8642317c9644ae2ac81fd2d3f44c4a72f0d8bf size: 527
mink@MacBook-Pro--Mink 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

```
mink@MacBook-Pro--Mink ~ % cd desktop
mink@MacBook-Pro--Mink desktop % mkdir Lab8_4

mink@MacBook-Pro--Mink desktop % cd Lab8_4

mink@MacBook-Pro--Mink Lab8_4 % git clone https://github.com/docker/getting-started.git

Cloning into 'getting-started'...
remote: Enumerating objects: 980, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 980 (delta 5), reused 1 (delta 1), pack-reused 971 (from 2)
Receiving objects: 100% (980/980), 5.28 MiB | 5.59 MiB/s, done.
Resolving deltas: 100% (523/523), done.
mink@MacBook-Pro--Mink Lab8_4 % cd getting-started/app

mink@MacBook-Pro--Mink app % nano package.json
mink@MacBook-Pro--Mink app %
```

```
  UW PICO 5.09                                        File: package.json

  "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"
  }
}
```

```
^G Get Help  ^O WriteOut  ^R Read File  ^Y Prev Pg  ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where is   ^V Next Pg  ^U UnCut Text ^T To Spell
```

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

```
mink@MacBook-Pro--Mink app % touch Dockerfile

mink@MacBook-Pro--Mink app % nano Dockerfile

mink@MacBook-Pro--Mink app % docker build -t myapp_6533800159 .

[+] Building 15.8s (10/10) FINISHED                                                                                 docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                                                0.0s
 => => transferring dockerfile: 188B                                                                                                0.0s
 => [internal] load metadata for docker.io/library/node:18-alpine                                                                   3.9s
 => [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@sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d975e9059dd066be3e274fbb25             2.6s
 => => resolve docker.io/library/node:18-alpine@sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d975e9059dd066be3e274fbb25             0.0s
 => => sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d975e9059dd066be3e274fbb25 7.67kB / 7.67kB                                      0.0s
 => => sha256:d59895120001b37ef5f75afc6342329f6037b1e2aea353a1055efa62b8bf6003 1.72kB / 1.72kB                                      0.0s
 => => sha256:7221f40791e5e6da0c9fa6b49b6238a51661222e4c58aba37e152e89914be09d 6.20kB / 6.20kB                                      0.0s
 => => sha256:52f827f723504aa3325bb5a54247f0dc4b92bb72569525bc951532c4ef679bd4 3.99MB / 3.99MB                                      0.6s
 => => sha256:4fe16fa8f46966191d59cfcabfff137a623b3cdda747d387bd85dcbf0feff3dd 39.66MB / 39.66MB                                    1.4s
 => => sha256:fc4eb59aab89271acc5a8bd8e5e96fc17af489976964461471ea28fc8e0be459 1.26MB / 1.26MB                                      1.2s
 => => extracting sha256:52f827f723504aa3325bb5a54247f0dc4b92bb72569525bc951532c4ef679bd4                                           0.1s
 => => sha256:e668eba0f82bcfec9fb0cd787bd7f3d013a1266567b092e90ff8b3d3be41807c 443B / 443B                                          1.3s
 => => extracting sha256:4fe16fa8f46966191d59cfcabfff137a623b3cdda747d387bd85dcbf0feff3dd                                           1.0s
 => => extracting sha256:fc4eb59aab89271acc5a8bd8e5e96fc17af489976964461471ea28fc8e0be459                                           0.0s
 => => extracting sha256:e668eba0f82bcfec9fb0cd787bd7f3d013a1266567b092e90ff8b3d3be41807c                                           0.0s
 => [internal] load build context                                                                                                   0.0s
 => => transferring context: 4.60MB                                                                                                 0.0s
 => [2/4] WORKDIR /app                                                                                                              0.1s
 => [3/4] COPY . .                                                                                                                  0.0s
 => [4/4] RUN yarn install --production                                                                                             8.8s
 => exporting to image                                                                                                              0.4s
 => => exporting layers                                                                                                             0.4s
 => => writing image sha256:8f0dbaf846427650fd260e4f7666aa2024a18069bef45556a5f56c14b04a2e9e                                        0.0s
 => => naming to docker.io/library/myapp_6533800159                                                                                 0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/rlfcfyerrqshb80xizmanqd5t

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview
mink@MacBook-Pro--Mink app %
```

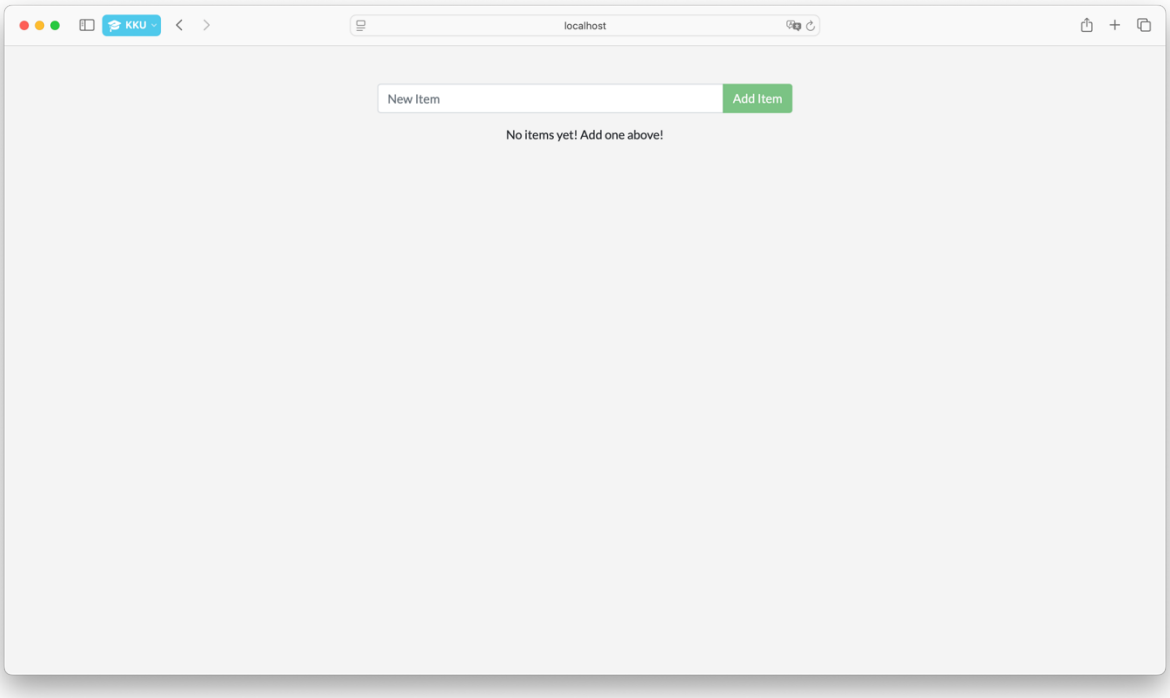
6. ทำการ Start ตัว Container ของแอปพลิเคชันที่สร้างขึ้น โดยใช้คำสั่ง

   $ 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

```
mink@MacBook-Pro--Mink app % docker run -dp 3000:3000 myapp_6533800159

66cea99dc0f90edaf422c914deefcc71c565b32240d5161fe87f0b33d23fdb0a
mink@MacBook-Pro--Mink 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

```
  UW PICO 5.09                                File: src/static/js/app.js

    const onItemRemoval = React.useCallback(
        item => {
            const index = items.findIndex(i => i.id === item.id);
            setItems([...items.slice(0, index), ...items.slice(index + 1)]);
        },
        [items],
    );

    if (items === null) return 'Loading...';

    return (
        <React.Fragment>
            <AddItemForm onNewItem={onNewItem} />
            {items.length === 0 && (
                <p className="text-center">There is no TODO item. Please add one to the list. By Tidaporn Teamsuk</p>
            )}
            {items.map(item => (
                <ItemDisplay

^G Get Help  ^O WriteOut  ^R Read File  ^Y Prev Pg  ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where is   ^V Next Pg  ^U UnCut Text ^T To Spell

mink@MacBook-Pro--Mink app % nano src/static/js/app.js
mink@MacBook-Pro--Mink app % docker build -t myapp_6533800159 .

[+] Building 2.4s (10/10) FINISHED                                  docker:desktop-linux
 => [internal] load build definition from Dockerfile                                0.0s
 => => transferring dockerfile: 188B                                                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@sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d975e9059dd066be3e274fbb25  0.0s
 => [internal] load build context                                                   0.0s
 => => transferring context: 9.76kB                                                 0.0s
 => CACHED [2/4] WORKDIR /app                                                       0.0s
 => CACHED [3/4] COPY . .                                                           0.0s
 => CACHED [4/4] RUN yarn install --production                                      0.0s
 => exporting to image                                                              0.0s
 => => exporting layers                                                             0.0s
 => => writing image sha256:75712bf9dc32cd98455ddfd5a265afc8bf31bc1e29f746aa1d9f99c2e5e18b7b  0.0s
 => => naming to docker.io/library/myapp_6533800159                                 0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/ri3v18cawr9p0e9k569mg449j

What's next:

    View a summary of image vulnerabilities and recommendations → docker scout quickview
mink@MacBook-Pro--Mink app % docker run -dp 3000:3000 myapp_6533800159

5e8bed460470d7f6b6e88139481d467df1726f71aa16cce626f14c954ff394bb
docker: Error response from daemon: driver failed programming external connectivity on endpoint optimistic_northcutt (77fd392f7121c88
cbbe0f269d2234b5850ceda46ebb7fca87a38452c802a6345): Bind for 0.0.0.0:3000 failed: port is already allocated.
```

(1) Error ที่เกิดขึ้นหมายความอย่างไร และเกิดขึ้นเพราะอะไร

ตอบ Port 3000 บนเครื่องยังถูกใช้งานอยู่จาก container ตัวก่อนหน้า สามารถทำการหยุด (stop) และลบ (remove) container ตัวเก่าเพื่อรันใหม่ได้

11. ลบ Container ของ Web application เวอร์ชันก่อนแก้ไขออกจากระบบ โดยใช้วิธีใดวิธีหนึ่งดังต่อไปนี้

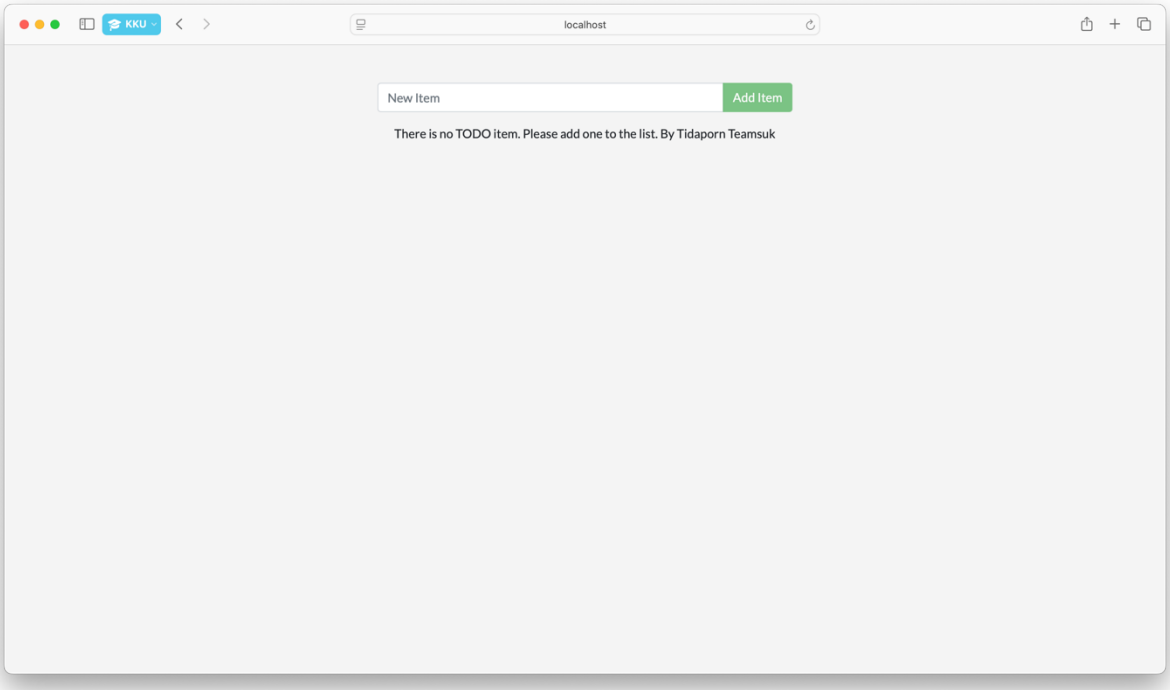
a. ผ่าน Command line interface
   i. ใช้คำสั่ง $ docker ps เพื่อดู Container ID ที่ต้องการจะลบ
   ii. Copy หรือบันทึก Container ID ไว้
   iii. ใช้คำสั่ง $ docker stop <Container ID ที่ต้องการจะลบ> เพื่อหยุดการทำงานของ Container ดังกล่าว
   iv. ใช้คำสั่ง $ docker rm <Container ID ที่ต้องการจะลบ> เพื่อทำการลบ

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

```
mink@MacBook-Pro--Mink app % docker ps

CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                    NAMES
66cea99dc0f9   8f0dbaf84642   "docker-entrypoint.s…"   23 minutes ago   Up 23 minutes   0.0.0.0:3000->3000/tcp   youthful_curran
mink@MacBook-Pro--Mink app % docker stop 66cea99dc0f9

66cea99dc0f9
mink@MacBook-Pro--Mink app % docker rm 66cea99dc0f9

66cea99dc0f9
mink@MacBook-Pro--Mink app %
```



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

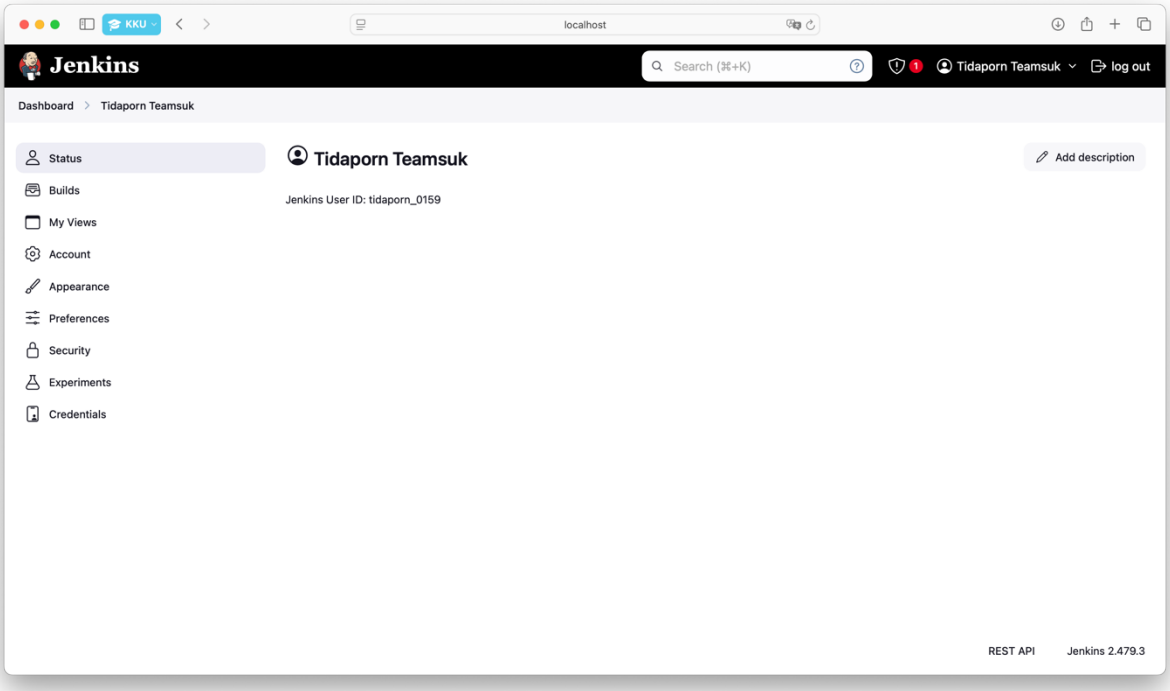
   $ docker run -p 8080:8080 -p 50000:50000 --restart=on-failure -v jenkins_home:/var/jenkins_home jenkins/jenkins:lts-jdk17
3. บันทึกรหัสผ่านของ Admin user ไว้สำหรับ log-in ในครั้งแรก

[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:

70126f384fc648ca83f0d2b3ac76bf46

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

*************************************************************
*************************************************************
*************************************************************
```

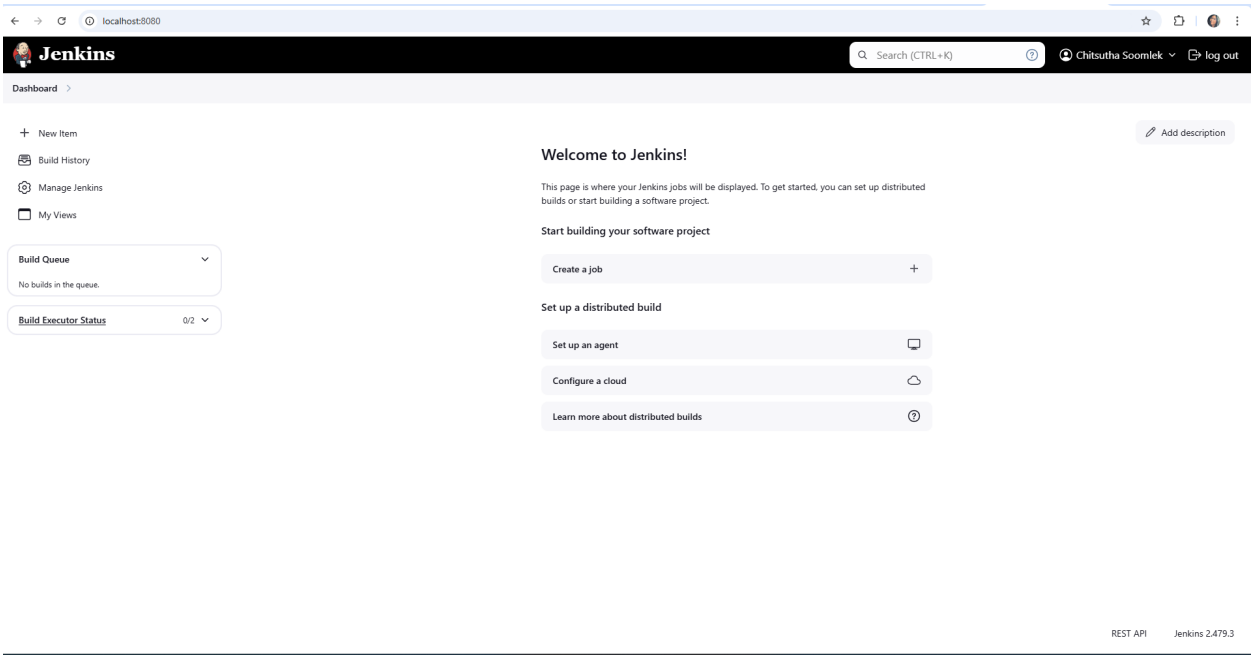
4. เมื่อได้รับการยืนยันว่า Jenkins is fully up and running ให้เปิดบราวเซอร์ และป้อนที่อยู่เป็น localhost:8080
5. ทำการ Unlock Jenkins ด้วยรหัสผ่านที่ได้ในข้อที่ 3
6. สร้าง Admin User โดยใช้ username เป็นชื่อจริงของนักศึกษาพร้อมรหัสสี่ตัวท้าย เช่น somsri_3062

[Check point#13] Capture หน้าจอที่แสดงผลการตั้งค่า

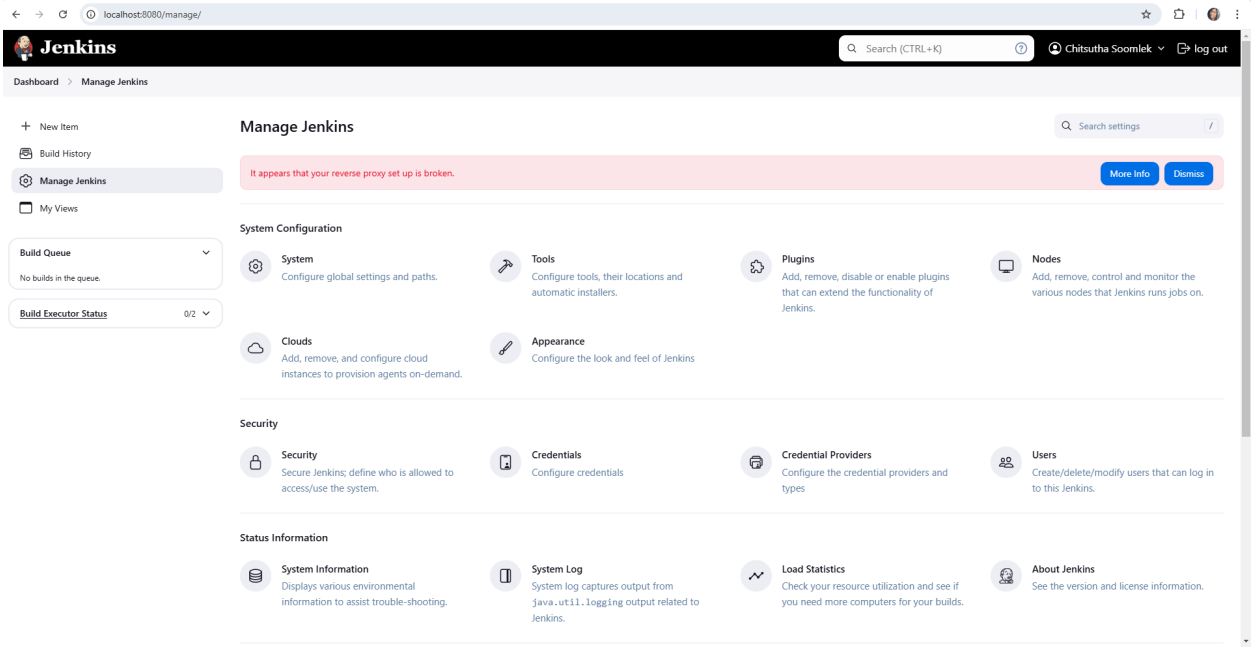


7. กำหนด Jenkins URL เป็น http://localhost:8080/lab8

8. เมื่อติดตั้งเรียบร้อยแล้วจะพบกันหน้า Dashboard ดังแสดงในภาพ

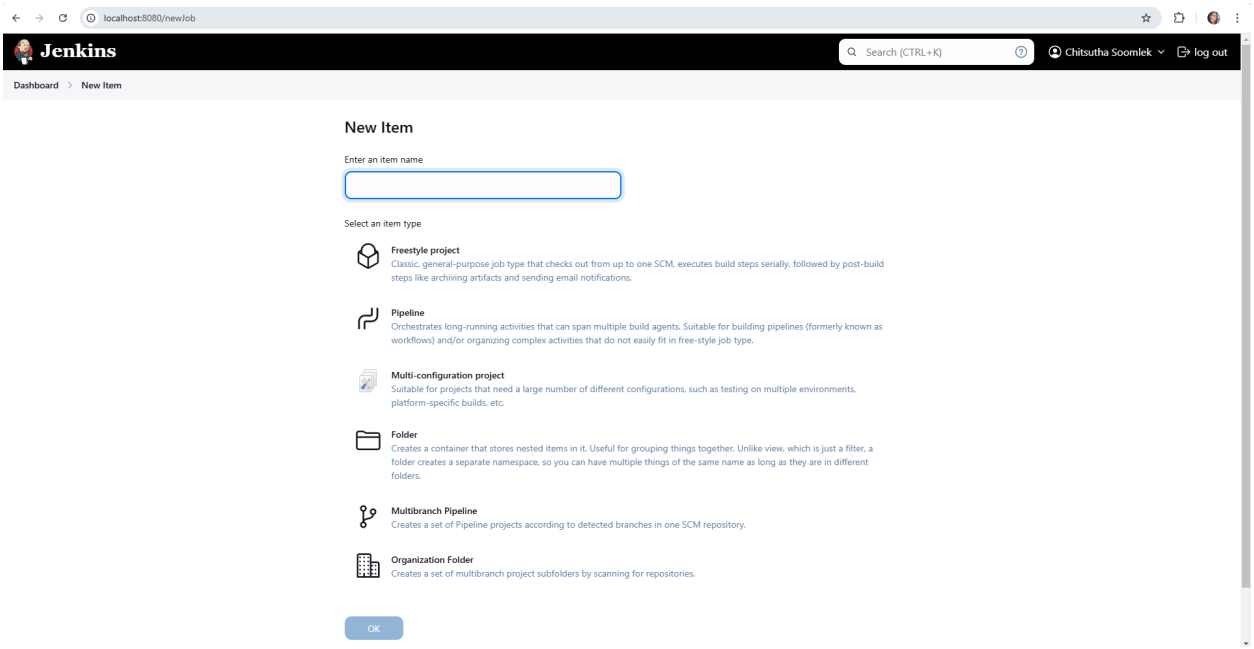


9. เลือก Manage Jenkins แล้วไปที่เมนู Plugins



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

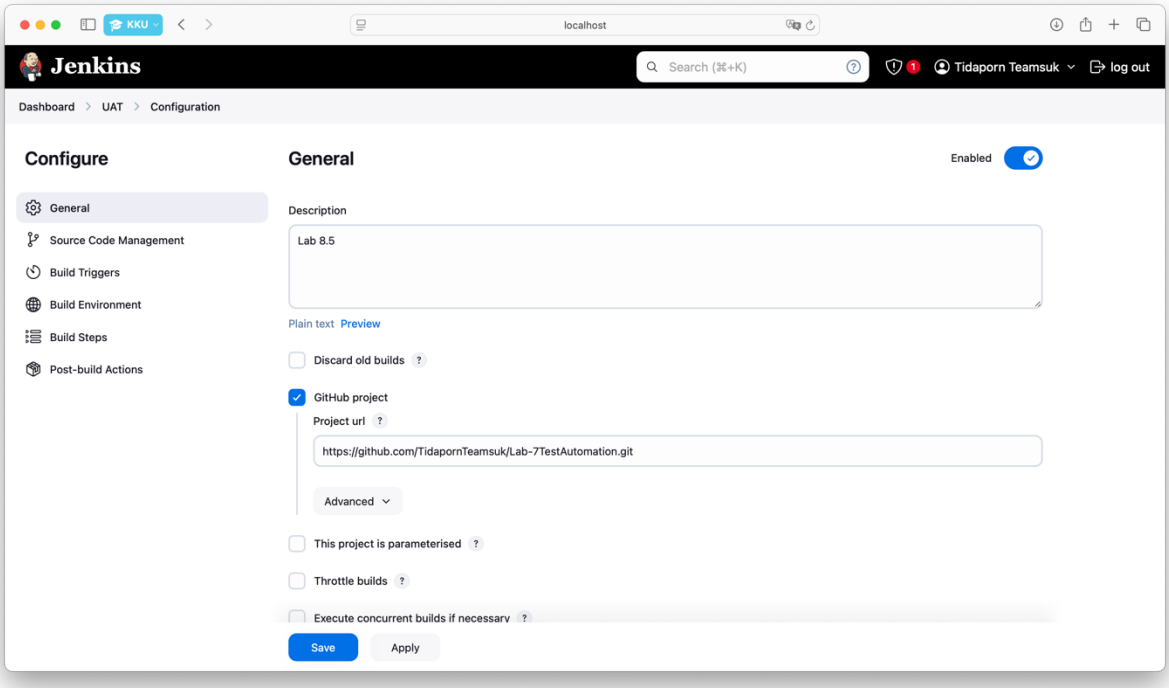
Description: Lab 8.5

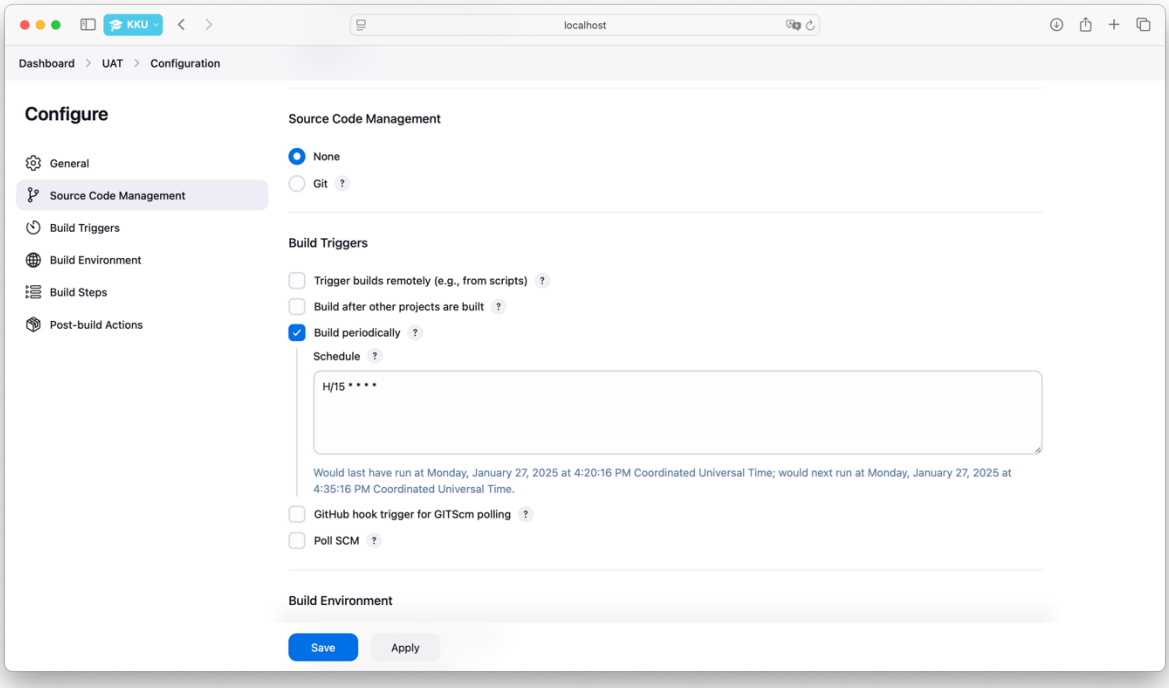
GitHub project: กดเลือก แล้วใส่ Project URL เป็น repository ที่เก็บโค้ด .robot (ดูขั้นตอนที่ 12)

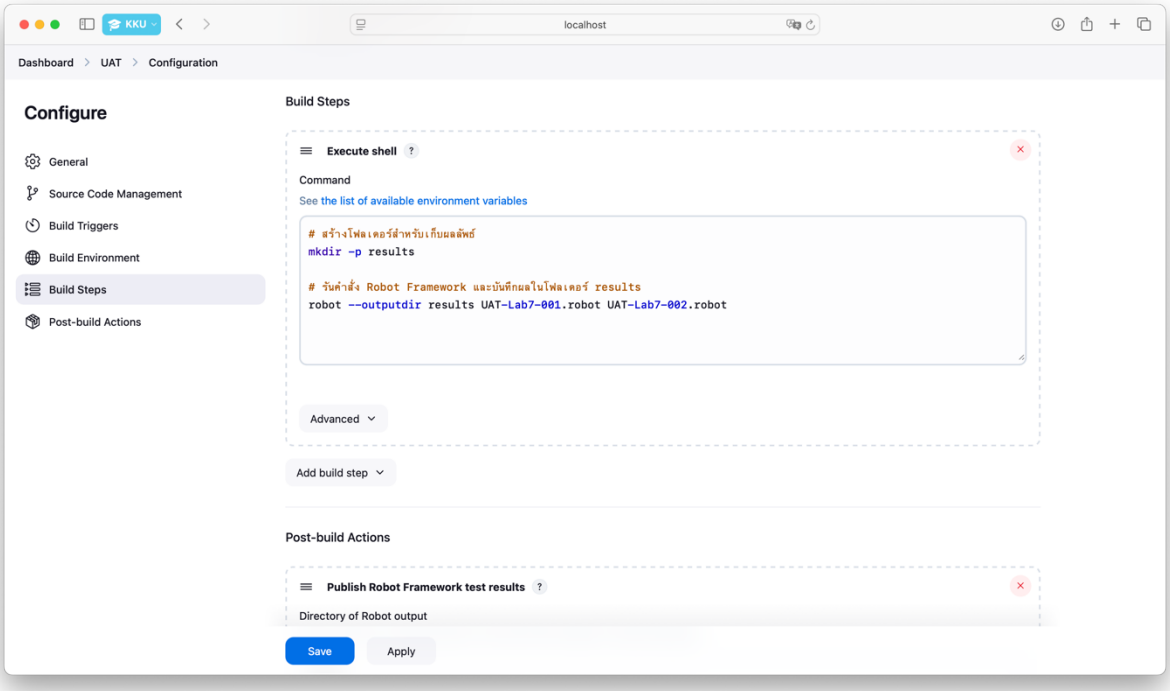
Build Trigger: เลือกแบบ Build periodically แล้วกำหนดให้ build ทุก 15 นาที

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

[Check point#14] Capture หน้าจอแสดงการตั้งค่า พร้อมกับตอบคำถามต่อไปนี้







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

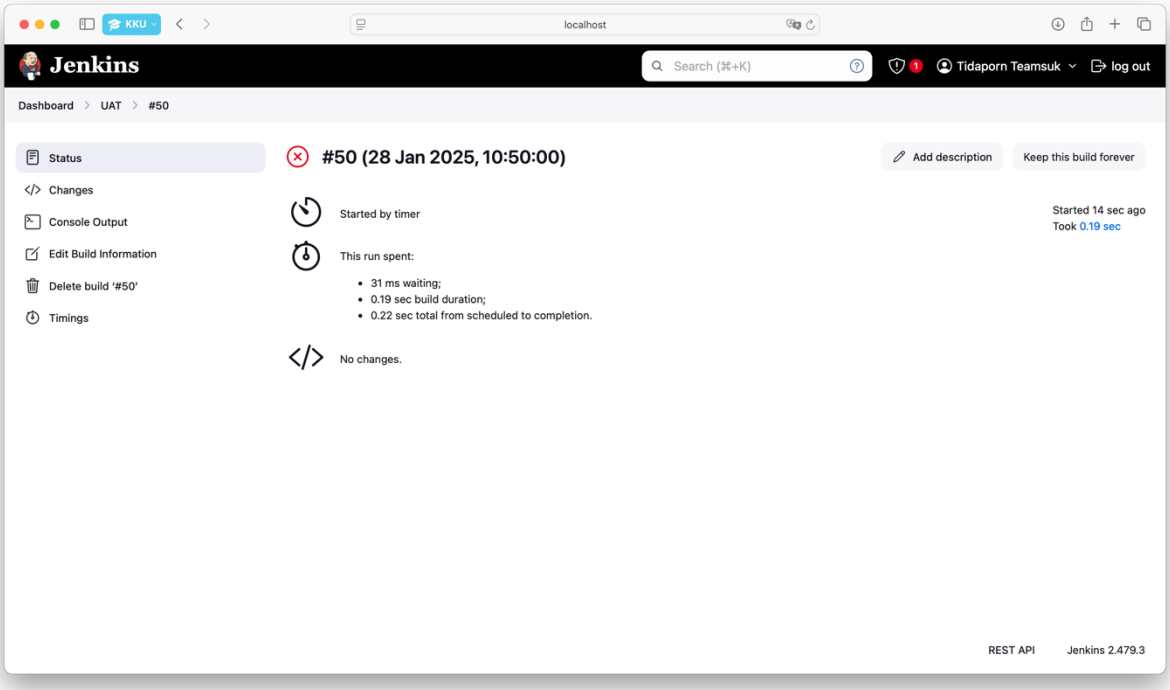
ตอบ ในที่นี้ ใช้คำสั่ง robot --outputdir results UAT-Lab7-001.robot UAT-Lab7-002.robot

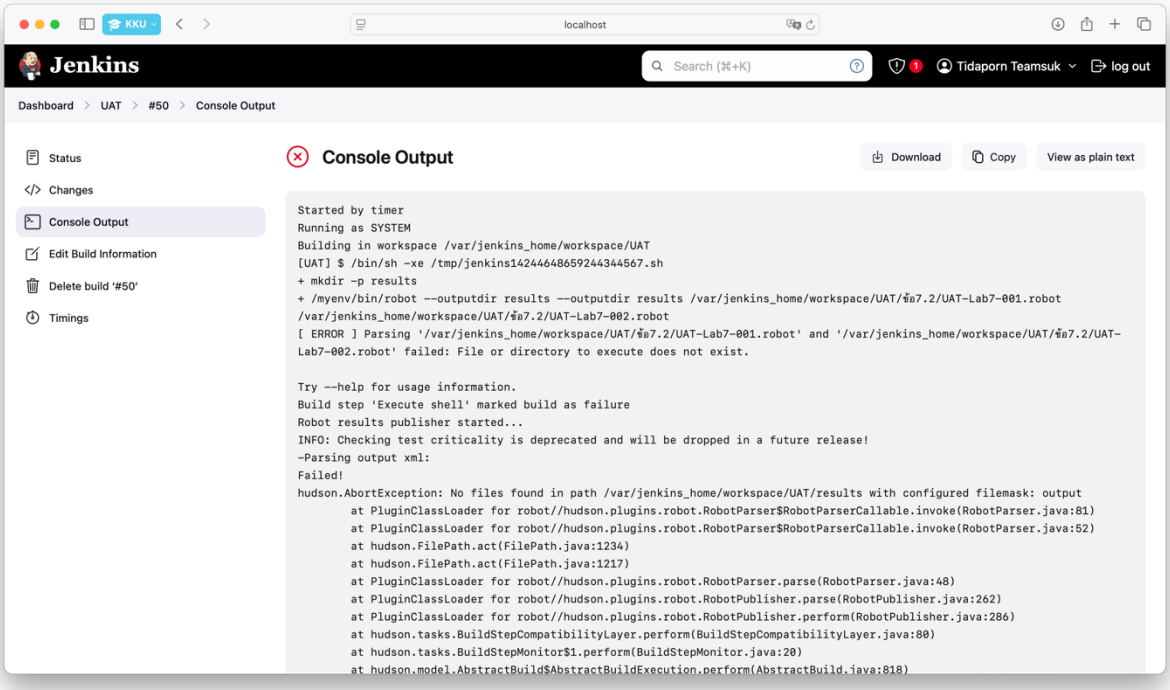
ซึ่งเป็นคำสั่ง robot ที่ใช้รันไฟล์ Robot Framework โดยตรงจาก command line

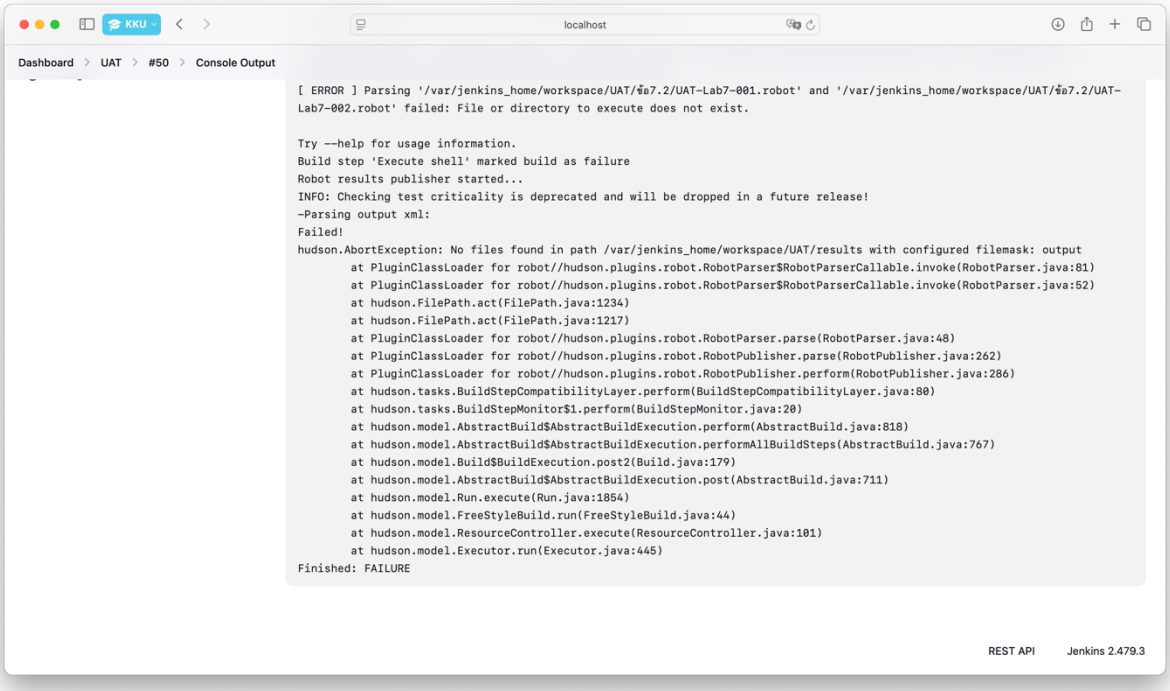
Post-build action: เพิ่ม Publish Robot Framework test results -> ระบุไดเร็คทอรีที่เก็บไฟล์ผลการทดสอบโดย Robot framework ในรูป xml และ html -> ตั้งค่า Threshold เป็น % ของการทดสอบที่ไม่ผ่านแล้วนับว่าซอฟต์แวร์มีปัญหา -> ตั้งค่า Threshold เป็น % ของการทดสอบที่ผ่านแล้วนับว่าซอฟต์แวร์มีอยู่ในสถานะที่สามารถนำไปใช้งานได้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

13. กด Apply และ Save
14. สั่ง Build Now

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




localhost
Jenkins
Search (⌘+K)
Tidaporn Teamsuk
log out
Dashboard > UAT > #50 > Console Output
Status
Changes
Console Output
Edit Build Information
Delete build '#50'
Timings
Console Output
Download
Copy
View as plain text
Started by timer
Running as SYSTEM
Building in workspace /var/jenkins_home/workspace/UAT
[UAT] $ /bin/sh -xe /tmp/jenkins14244648659244344567.sh
+ mkdir -p results
+ /myenv/bin/robot --outputdir results --outputdir results /var/jenkins_home/workspace/UAT/ข้อ7.2/UAT-Lab7-001.robot
/var/jenkins_home/workspace/UAT/ข้อ7.2/UAT-Lab7-002.robot
[ERROR] Parsing '/var/jenkins_home/workspace/UAT/ข้อ7.2/UAT-Lab7-001.robot' and '/var/jenkins_home/workspace/UAT/ข้อ7.2/UAT-Lab7-002.robot' failed: File or directory to execute does not exist.
Try --help for usage information.
Build step 'Execute shell' marked build as failure
Robot results publisher started...
INFO: Checking test criticality is deprecated and will be dropped in a future release!
-Parsing output xml:
Failed!
hudson.AbortException: No files found in path /var/jenkins_home/workspace/UAT/results with configured filemask: output
at PluginClassLoader for robot//hudson.plugins.robot.RobotParser$RobotParserCallable.invoke(RobotParser.java:81)
at PluginClassLoader for robot//hudson.plugins.robot.RobotParser$RobotParserCallable.invoke(RobotParser.java:52)
at hudson.FilePath.act(FilePath.java:1234)
at hudson.FilePath.act(FilePath.java:1217)
at PluginClassLoader for robot//hudson.plugins.robot.RobotParser.parse(RobotParser.java:48)
at PluginClassLoader for robot//hudson.plugins.robot.RobotPublisher.parse(RobotPublisher.java:262)
at PluginClassLoader for robot//hudson.plugins.robot.RobotPublisher.perform(RobotPublisher.java:286)
at hudson.tasks.BuildStepCompatibilityLayer.perform(BuildStepCompatibilityLayer.java:80)
at hudson.tasks.BuildStepMonitor$1.perform(BuildStepMonitor.java:20)
at hudson.model.AbstractBuild$AbstractBuildExecution.perform(AbstractBuild.java:818)


localhost
Dashboard > UAT > #50 > Console Output
[ERROR] Parsing '/var/jenkins_home/workspace/UAT/ข้อ7.2/UAT-Lab7-001.robot' and '/var/jenkins_home/workspace/UAT/ข้อ7.2/UAT-Lab7-002.robot' failed: File or directory to execute does not exist.
Try --help for usage information.
Build step 'Execute shell' marked build as failure
Robot results publisher started...
INFO: Checking test criticality is deprecated and will be dropped in a future release!
-Parsing output xml:
Failed!
hudson.AbortException: No files found in path /var/jenkins_home/workspace/UAT/results with configured filemask: output
at PluginClassLoader for robot//hudson.plugins.robot.RobotParser$RobotParserCallable.invoke(RobotParser.java:81)
at PluginClassLoader for robot//hudson.plugins.robot.RobotParser$RobotParserCallable.invoke(RobotParser.java:52)
at hudson.FilePath.act(FilePath.java:1234)
at hudson.FilePath.act(FilePath.java:1217)
at PluginClassLoader for robot//hudson.plugins.robot.RobotParser.parse(RobotParser.java:48)
at PluginClassLoader for robot//hudson.plugins.robot.RobotPublisher.parse(RobotPublisher.java:262)
at PluginClassLoader for robot//hudson.plugins.robot.RobotPublisher.perform(RobotPublisher.java:286)
at hudson.tasks.BuildStepCompatibilityLayer.perform(BuildStepCompatibilityLayer.java:80)
at hudson.tasks.BuildStepMonitor$1.perform(BuildStepMonitor.java:20)
at hudson.model.AbstractBuild$AbstractBuildExecution.perform(AbstractBuild.java:818)
at hudson.model.AbstractBuild$AbstractBuildExecution.performAllBuildSteps(AbstractBuild.java:767)
at hudson.model.Build$BuildExecution.post2(Build.java:179)
at hudson.model.AbstractBuild$AbstractBuildExecution.post(AbstractBuild.java:711)
at hudson.model.Run.execute(Run.java:1854)
at hudson.model.FreeStyleBuild.run(FreeStyleBuild.java:44)
at hudson.model.ResourceController.execute(ResourceController.java:101)
at hudson.model.Executor.run(Executor.java:445)
Finished: FAILURE
REST API
Jenkins 2.479.3