

Lab Worksheet

ชื่อ-นามสกุล นายอธิภัทร ภู่ด่านวัว รหัสนักศึกษา 653380349-0 Section 3

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

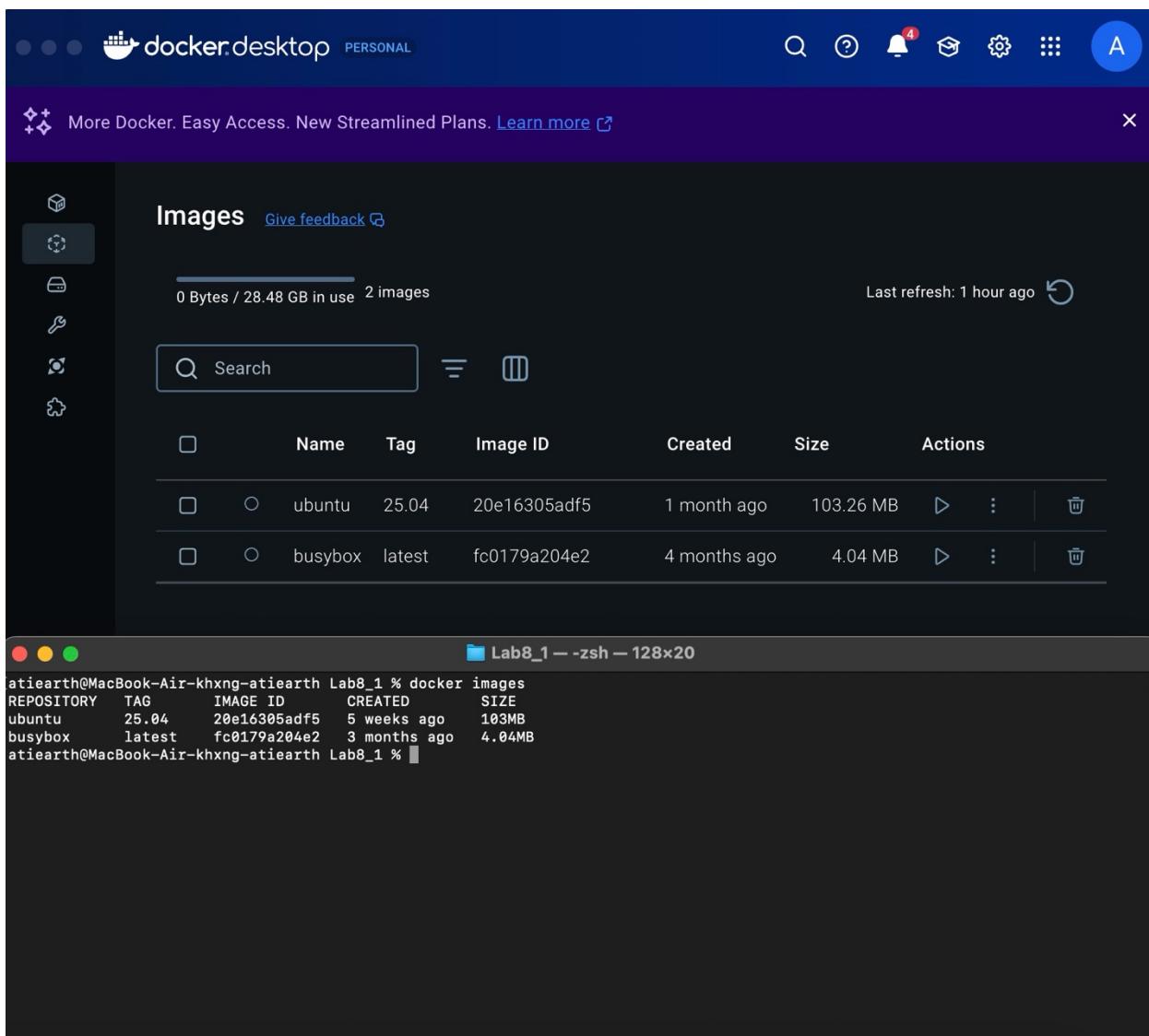
1. ติดตั้ง Docker desktop ลงบนเครื่องคอมพิวเตอร์ โดยดาวน์โหลดจาก <https://www.docker.com/get-started>
2. สร้าง Account บน Docker hub (<https://hub.docker.com/signup>)
3. กำหนดให้ \$ หมายถึง Command prompt และ <> หมายถึง ให้ป้อนค่าของพารามิเตอร์ที่กำหนด

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

Lab Worksheet



- (1) สิ่งที่อยู่ภายใต้คอลัมน์ Repository คืออะไร => shawn, ubuntu, busybox etc
- (2) Tag ที่ใช้บ่งบอกถึงอะไร => version ของ repository

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"

Lab Worksheet

11. ป้อนคำสั่ง \$ docker ps -a

[Check point#2] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงผลลัพธ์ที่ได้ตั้งแต่ขั้นตอนที่ 6-12 พร้อมกับตอบคำถามต่อไปนี้

```

[atiearth@MacBook-Air-khxng-atiearth Lab8_1 % docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
ubuntu 25.04 20e16305adf5 5 weeks ago 103MB
busybox latest fc0179a204e2 3 months ago 4.04MB
[atiearth@MacBook-Air-khxng-atiearth Lab8_1 % clear

[atiearth@MacBook-Air-khxng-atiearth Lab8_1 % docker run busybox
[atiearth@MacBook-Air-khxng-atiearth 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 23 03:08 .
drwxr-xr-x 1 root root 4096 Jan 23 03:08 ..
-rwxr-xr-x 1 root root 0 Jan 23 03:08 .dockerenv
drwxr-xr-x 2 root root 12288 Sep 26 21:31 bin
drwxr-xr-x 5 root root 360 Jan 23 03:08 dev
drwxr-xr-x 1 root root 4096 Jan 23 03:08 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 218 root root 0 Jan 23 03:08 proc
drwx----- 1 root root 4096 Jan 23 03:08 root
dr-xr-xr-x 11 root root 0 Jan 23 03:08 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
[atiearth@MacBook-Air-khxng-atiearth Lab8_1 % docker run busybox echo "Hello Athipat Poodanwua from busybox"
Hello Athipat Poodanwua from busybox
[atiearth@MacBook-Air-khxng-atiearth Lab8_1 % docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
78fd2054cb0 busybox "echo 'Hello Athipat..'" 6 seconds ago Exited (0) 5 seconds ago nice_heisenberg
3d826cdc07e3 busybox "sh" About a minute ago Exited (0) 44 seconds ago sad_meitner
6b7b1f7191cf busybox "sh" About a minute ago Exited (0) About a minute ago loving_tesla
atiearth@MacBook-Air-khxng-atiearth Lab8_1 %

```

(1) เมื่อใช้ option -it ในคำสั่ง run ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอกลังเขป

⇒ ส่งผลให้สามารถเข้าสู่ shell (sh) ภายใน container

(2) คอลัมน์ STATUS จากการรันคำสั่ง docker ps -a แสดงถึงข้อมูลอะไร

⇒ Exited (0) -> การรันเสร็จสมบูรณ์โดยไม่มีข้อผิดพลาด

Lab Worksheet

12. ป้อนคำสั่ง \$ docker rm <container ID ที่ต้องการลบ>

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

```

l-rwxrwxrwx 1 root      root          3 Sep 26 21:31 lib64 -> lib
drwxr-xr-x 218 root      root          0 Jan 23 03:08 proc
drwx----- 1 root      root        4096 Jan 23 03:08 root
drwxr-xr-x 11 root      root          0 Jan 23 03:08 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
[ atiearth@MacBook-Air-khxng-atiearth Lab8_1 % docker run busybox echo "Hello Athipat Poodanwua from busybox"
Hello Athipat Poodanwua from busybox
[ atiearth@MacBook-Air-khxng-atiearth Lab8_1 % docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
78fd2054ccb0 busybox "echo 'Hello Athipat..." 6 seconds ago Exited (0) 5 seconds ago
3d826cdc07e3 busybox "sh"     About a minute ago Exited (0) 44 seconds ago
6b7b1f7191cf busybox "sh"     About a minute ago Exited (0) About a minute ago
[ atiearth@MacBook-Air-khxng-atiearth Lab8_1 % docker rm 78fd2054ccb0
78fd2054ccb0
[ atiearth@MacBook-Air-khxng-atiearth Lab8_1 % docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
3d826cdc07e3 busybox "sh"     12 minutes ago Exited (0) 12 minutes ago
6b7b1f7191cf busybox "sh"     12 minutes ago Exited (0) 12 minutes ago
[ atiearth@MacBook-Air-khxng-atiearth Lab8_1 %

```

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

สำหรับเครื่องที่ใช้ระบบปฏิบัติการวินโดว์ส (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"

Lab Worksheet

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

```
[atiearth@MacBook-Air-khxng-atiearth Lab8_2 % docker build -t dockerfile .
[+] Building 0.0s (5/5) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 149B
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behav 0.0s
=> WARN: MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the 0.0s
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behav 0.0s
=> [internal] load metadata for docker.io/library/busybox:latest 0.0s
=> [internal] load .dockerrcignore 0.0s
=> => transferring context: 2B 0.0s
=> [1/1] FROM docker.io/library/busybox:latest 0.0s
=> exporting to image 0.0s
=> => exporting layers 0.0s
=> => writing image sha256:a2c766bbec935432de8554aac12813954522b788d2f963716363e3159688afba 0.0s
=> => naming to docker.io/library/dockerfile 0.0s
3 warnings found (use docker --debug to expand):
- JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to
OS signals (line 2)
- MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the same stage bec
ause 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)
[atiearth@MacBook-Air-khxng-atiearth Lab8_2 % docker run dockerfile
Athipat Poodanwua 653380349-0 Earth
atiearth@MacBook-Air-khxng-atiearth Lab8_2 % ]
```

- (1) คำสั่งที่ใช้ในการ run คือ

⇒ Docker run dockerfile

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

⇒ กำหนด tag ให้กับ file

Lab Worksheet**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

สำหรับเครื่องที่ใช้ระบบปฏิบัติการ Mac OS หรือ Linux บนหน้าต่าง Terminal และป้อนคำสั่งต่อไปนี้

FROM busybox

CMD echo "Hi there. My work is done. You can run them from my Docker image."

CMD echo "ชื่อ-นามสกุล รหัสนักศึกษา"

สำหรับเครื่องที่ใช้ระบบปฏิบัติการ Mac OS 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) และแสดงผลลัพธ์ที่ได้ในขั้นตอนที่ 5

Lab Worksheet

```

Start a build
[atiearth@MacBook-Air-khxng-atiearth Lab8_3 % docker build -t botun1438300004291/lab8 .
[+] Building 0.0s (5/5) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 170B
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals 0.0s
=> WARN: MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the same stage because only t 0.0s
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals 0.0s
=> [internal] load metadata for docker.io/library/busybox:latest 0.0s
=> [internal] load .dockerrcignore 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 sha256:48db37d95cd17b6e80154b248ef2019ed378eb92ba6e8f1aa303c3188b9c2371 0.0s
=> => naming to docker.io/botun1438300004291/lab8 0.0s

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

[atiearth@MacBook-Air-khxng-atiearth Lab8_3 % docker run botun1438300004291/lab8
Athipat Poodanwua 653380349-0
atiearth@MacBook-Air-khxng-atiearth Lab8_3 %

```

6. ทำการ Push ตัว Docker image ไปเว็บ Docker Hub โดยการใช้คำสั่ง

\$ docker push <username> ที่ลงทะเบียนกับ Docker Hub>/lab8

ในการนี้ต้องป้อนรหัสผ่านที่ได้ Login ไว้ก่อน ให้ใช้คำสั่งต่อไปนี้ เพื่อ Login ก่อนทำการ Push

\$ docker login และป้อน Username และ Password ตามที่ระบุใน Command prompt หรือใช้คำสั่ง

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

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

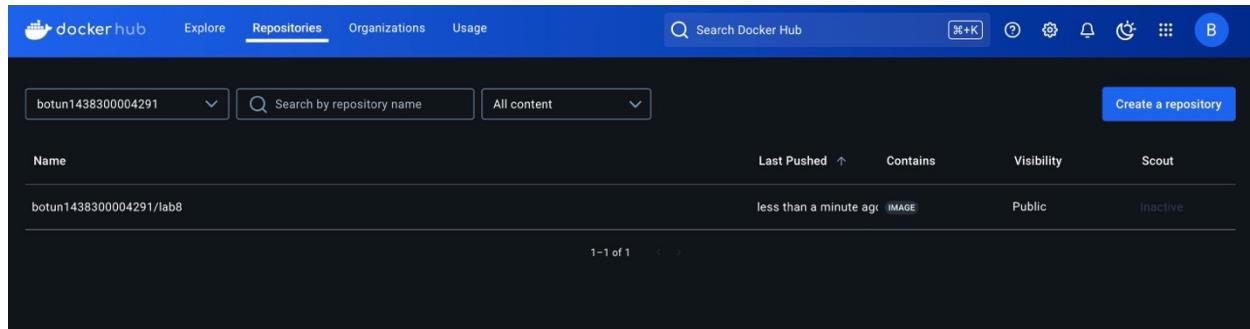
[Check point#6] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดง Repository ที่มี Docker image (<username>/lab8)

Lab Worksheet

```

>> => transferring dockerfile: 170B
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals 0.0s
=> WARN: MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the same stage because only t 0.0s
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals 0.0s
=> [internal] load metadata for docker.io/library/busybox:latest 0.0s
=> [internal] load .dockignore 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 sha256:48db37d95cd17b6e80154b248ef2019ed378eb92ba6e8f1aa303c3188b9c2371 0.0s
=> => naming to docker.io/botun1438300004291/lab8 0.0s

3 warnings found (use docker --debug to expand):
- JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 2)
- MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the same stage because only the last one wi 0.0s
ll be used (line 2)
- JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 3)
[atiearth@MacBook-Air-khxng-atiearth Lab8_3 % docker run botun1438300004291/lab8
Athipat Poodanwua 653380349-0
[atiearth@MacBook-Air-khxng-atiearth Lab8_3 % docker push botun1438300004291/lab8
Using default tag: latest
The push refers to repository [docker.io/botun1438300004291/lab8]
613e5fc50eb9: Mounted from library/busybox
latest: digest: sha256:4e7f55d132b2287b67fa9a96bdaf9a02a3dc8de5a4cf1793fb3fed978046d2f size: 527
atiearth@MacBook-Air-khxng-atiearth Lab8_3 %
]
```

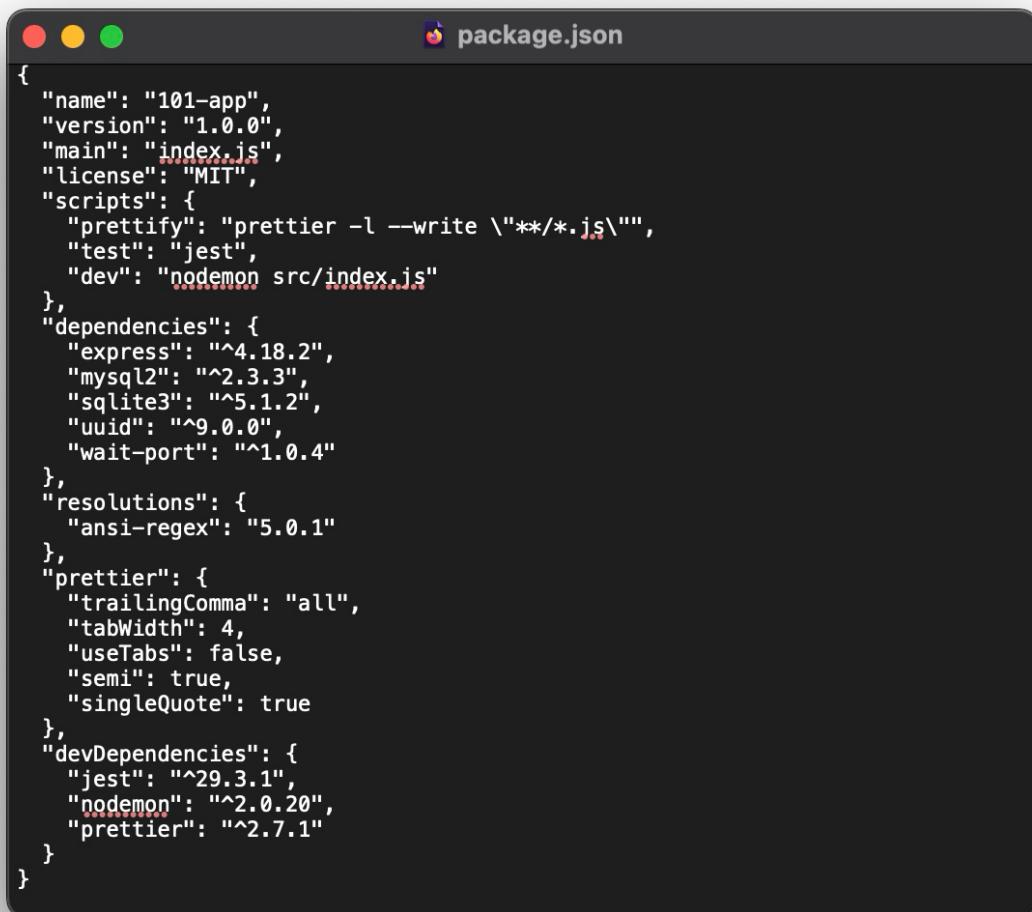


แบบฝึกปฏิบัติที่ 8.4: การ Build และ Update Container image และการ Update แอปพลิเคชัน

1. เปิด Command line หรือ Terminal จากนั้นสร้าง Directory ชื่อ Lab8_4
2. ทำการ Clone ซอฟต์แวร์ของเว็บแอปพลิเคชันจาก 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

Lab Worksheet



```

{
  "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 พร้อมกับใส่เนื้อหาดังต่อไปนี้ลงไว้ในไฟล์

```

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

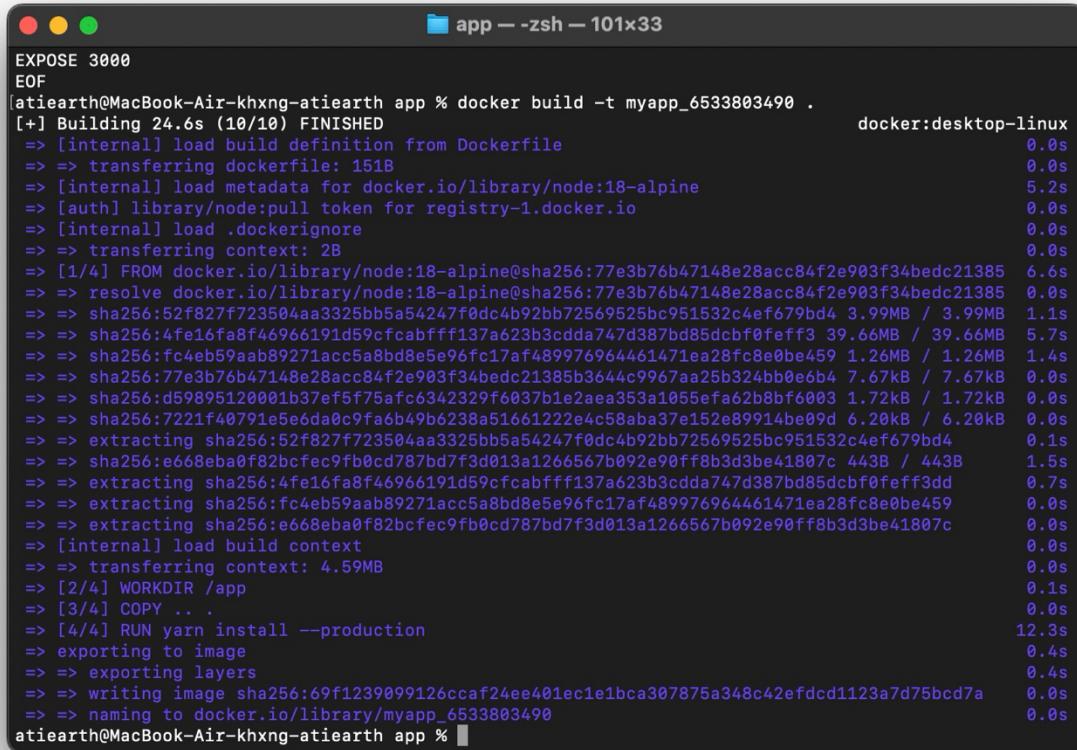
```

5. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้ โดยกำหนดใช้ชื่อ image เป็น myapp_ รหัสนศ.
ศ. ไม่มีขีด

Lab Worksheet

\$ docker build -t <myapp_รหัสคนศ. ไม่มีขีด> .

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

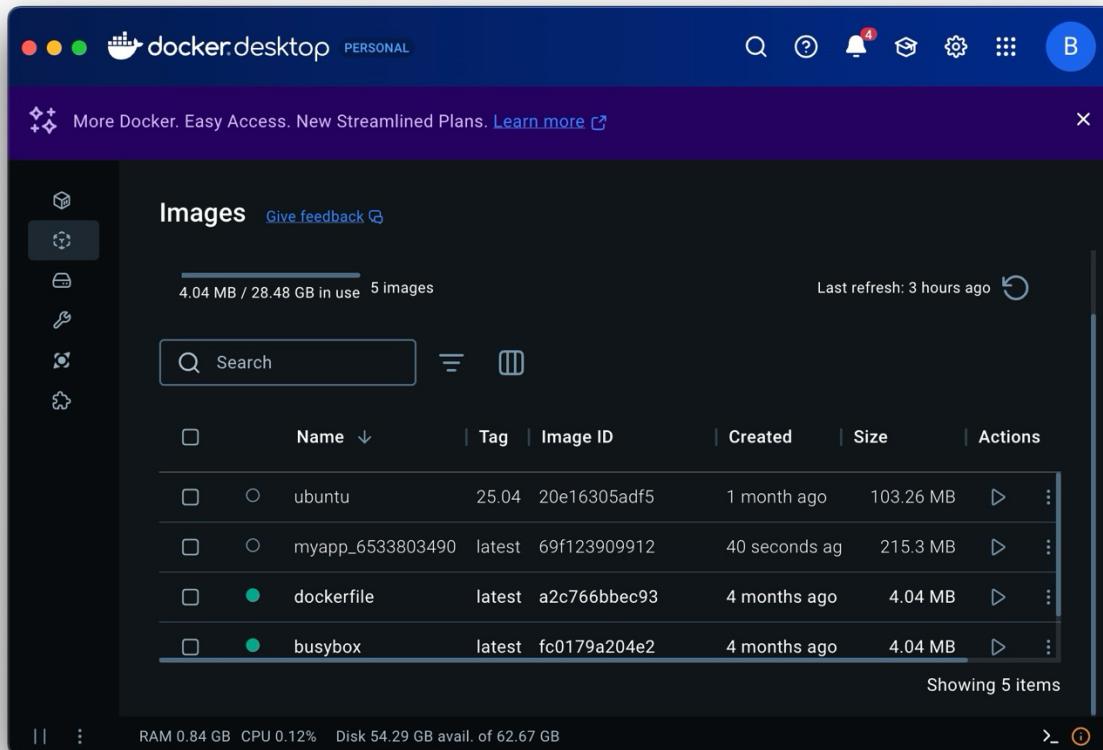


```

EXPOSE 3000
EOF
[+] Building 24.6s (10/10) FINISHED                                            docker:desktop-linux
=> [internal] load build definition from Dockerfile                         0.0s
=> => transferring dockerfile: 151B                                         0.0s
=> [internal] load metadata for docker.io/library/node:18-alpine           5.2s
=> [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@sha256:77e3b76b47148e28acc84f2e903f34bedc21385 6.6s
=> => resolve docker.io/library/node:18-alpine@sha256:77e3b76b47148e28acc84f2e903f34bedc21385 0.0s
=> => sha256:52f827f723504aa3325bb5a54247f0dc4b92bb72569525bc951532c4ef679bd4 3.99MB / 3.99MB 1.1s
=> => sha256:4fe16fa8f46966191d59cfcabfff137a623b3cdda747d387bd85dcfb0feff3 39.66MB / 39.66MB 5.7s
=> => sha256:f04eb59aab89271acc5a8bd8e5e96fc17af489976964461471ea28fc8e0be459 1.26MB / 1.26MB 1.4s
=> => sha256:77e3b76b47148e28acc84f2e903f34bedc21385b3644c9967aa25b324bb0e6b4 7.67KB / 7.67KB 0.0s
=> => sha256:d59895120001b37ef5f75afc6342329f0037b1e2aea353a1855efa62b8bf0003 1.72kB / 1.72kB 0.0s
=> => sha256:7221f40791e5e6da0c9fa6b49b6238a51661222e4c58aba37e152e89914be09d 6.20kB / 6.20kB 0.0s
=> => extracting sha256:52f827f723504aa3325bb5a54247f0dc4b92bb72569525bc951532c4ef679bd4 0.1s
=> => sha256:e668eba0f82bcfec9fb0cd787bd7f3d013a1266567b092e90ff8b3d3be41807c 443B / 443B 1.5s
=> => extracting sha256:4fe16fa8f46966191d59cfcabfff137a623b3cdda747d387bd85dcfb0feff3dd 0.7s
=> => extracting sha256:fc4eb59aab89271acc5a8bd8e5e96fc17af489976964461471ea28fc8e0be459 0.0s
=> => extracting sha256:e668eba0f82bcfec9fb0cd787bd7f3d013a1266567b092e90ff8b3d3be41807c 0.0s
=> [internal] load build context                                              0.0s
=> => transferring context: 4.59MB                                         0.0s
=> [2/4] WORKDIR /app                                                       0.1s
=> [3/4] COPY .. .                                                       0.0s
=> [4/4] RUN yarn install --production                                     12.3s
=> => exporting to image                                                 0.4s
=> => exporting layers                                                 0.4s
=> => writing image sha256:69f1239099126ccaf24ee401ec1e1bca307875a348c42efcd1123a7d75bcd7a 0.0s
=> => naming to docker.io/library/myapp_6533803490                      0.0s
atiearth@MacBook-Air-khxng-atiearth app %

```

Lab Worksheet



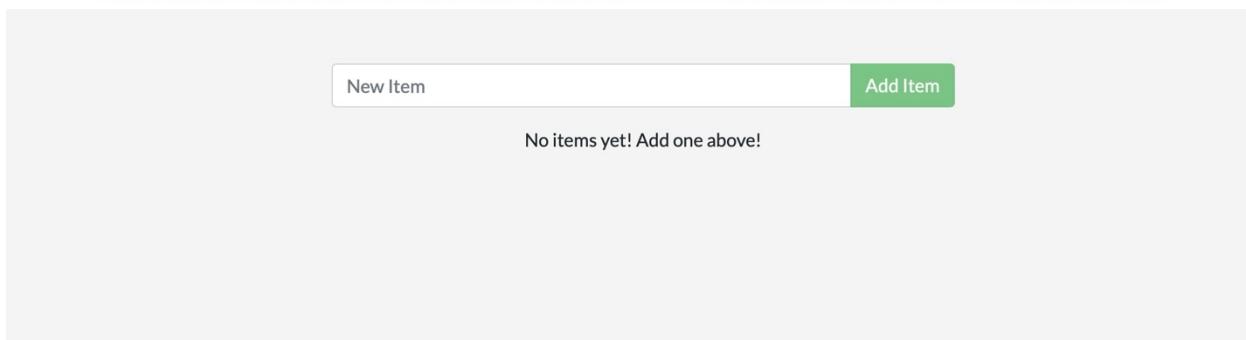
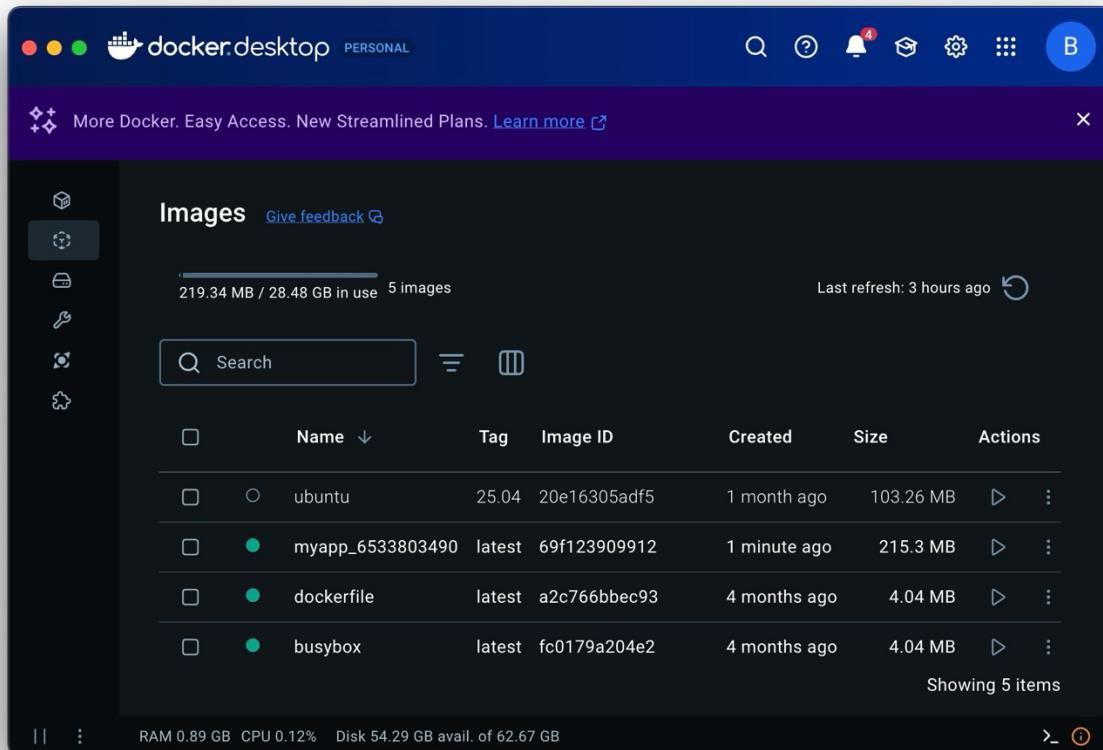
6. ทำการ Start ตัว Container ของแอปพลิเคชันที่สร้างขึ้น โดยใช้คำสั่ง

\$ docker run -dp 3000:3000 <myapp_รหัสศ.ไม่มีขีด>

7. เปิด Browser ไปที่ URL = <http://localhost:3000>

[Check point#9] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงผลลัพธ์ที่ได้บน Browser และ Dashboard ของ Docker desktop

Lab Worksheet



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

< p className="text-center" > No items yet! Add one above! </ p > เป็น

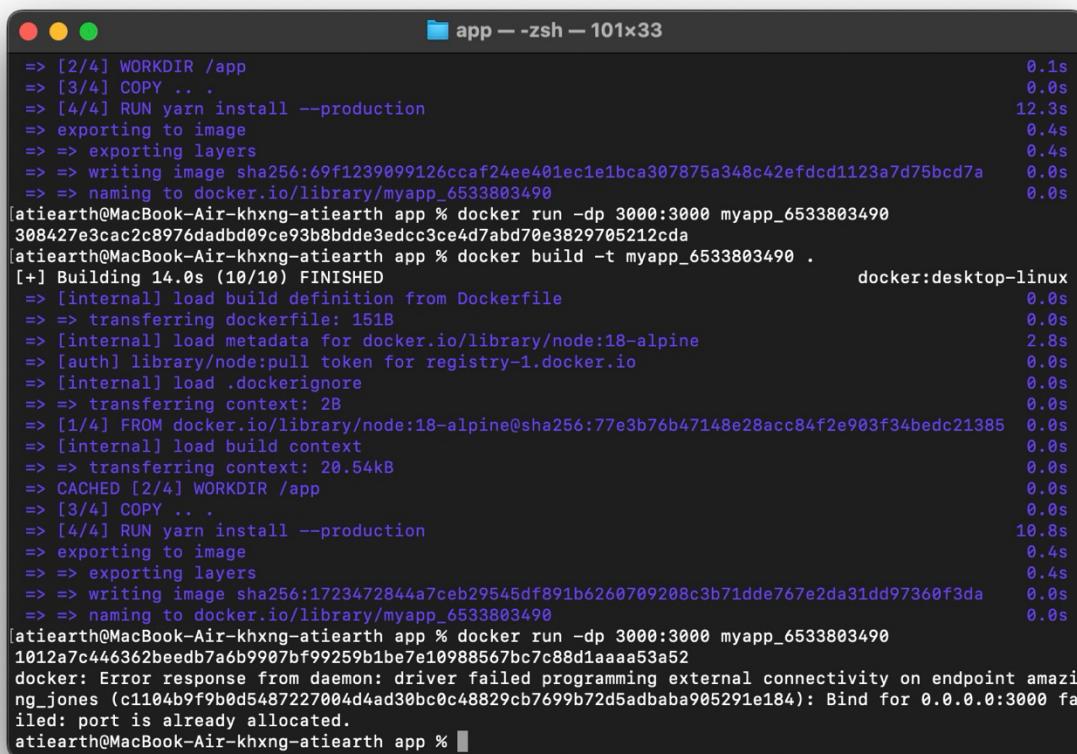
< p className="text-center" > There is no TODO item. Please add one to the list.

By ชื่อและนามสกุลของนักศึกษา

Lab Worksheet

- 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



```

app -- zsh -- 101x33
[2/4] WORKDIR /app
[3/4] COPY ... .
[4/4] RUN yarn install --production
=> exporting to image
=> >> exporting layers
=> >> writing image sha256:69f1239099126ccaf24ee401ec1e1bca307875a348c42efcd1123a7d75bcd7a
=> >> naming to docker.io/library/myapp_6533803490
[+] Building 14.0s (10/10) FINISHED
          docker:desktop-linux
=> [internal] load build definition from Dockerfile
=> >> transferring dockerfile: 151B
=> [internal] load metadata for docker.io/library/node:18-alpine
=> [auth] library/node:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> >> transferring context: 2B
=> [1/4] FROM docker.io/library/node:18-alpine@sha256:77e3b76b47148e28acc84f2e903f34bedc21385
=> [internal] load build context
=> >> transferring context: 20.54kB
=> CACHED [2/4] WORKDIR /app
=> [3/4] COPY ... .
=> [4/4] RUN yarn install --production
=> exporting to image
=> >> exporting layers
=> >> writing image sha256:1723472844a7ceb29545df891b6260709208c3b71dde767e2da31dd97360f3da
=> >> naming to docker.io/library/myapp_6533803490
[+] Building 10.8s (10/10) FINISHED
          docker:desktop-linux
=> [internal] load build definition from Dockerfile
=> >> transferring dockerfile: 151B
=> [internal] load metadata for docker.io/library/node:18-alpine
=> [auth] library/node:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> >> transferring context: 2B
=> [1/4] FROM docker.io/library/node:18-alpine@sha256:77e3b76b47148e28acc84f2e903f34bedc21385
=> [internal] load build context
=> >> transferring context: 20.54kB
=> CACHED [2/4] WORKDIR /app
=> [3/4] COPY ... .
=> [4/4] RUN yarn install --production
=> exporting to image
=> >> exporting layers
=> >> writing image sha256:1012a7c446362beedb7a6b9907bf99259b1be7e10988567bc7c88d1aaa53a52
=> >> naming to docker.io/library/myapp_6533803490
[+] Building 0.0s (10/10) FINISHED
          docker:desktop-linux
=> [internal] load build definition from Dockerfile
=> >> transferring dockerfile: 151B
=> [internal] load metadata for docker.io/library/node:18-alpine
=> [auth] library/node:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> >> transferring context: 2B
=> [1/4] FROM docker.io/library/node:18-alpine@sha256:77e3b76b47148e28acc84f2e903f34bedc21385
=> [internal] load build context
=> >> transferring context: 20.54kB
=> CACHED [2/4] WORKDIR /app
=> [3/4] COPY ... .
=> [4/4] RUN yarn install --production
=> exporting to image
=> >> exporting layers
=> >> writing image sha256:c1104b9f9b0d5487227004d4ad30bc0c48829cb7699b72d5adbaba905291e184
=> >> naming to docker.io/library/myapp_6533803490
[+] Building 0.0s (10/10) FINISHED
          docker:desktop-linux
=> [internal] load build definition from Dockerfile
=> >> transferring dockerfile: 151B
=> [internal] load metadata for docker.io/library/node:18-alpine
=> [auth] library/node:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> >> transferring context: 2B
=> [1/4] FROM docker.io/library/node:18-alpine@sha256:77e3b76b47148e28acc84f2e903f34bedc21385
=> [internal] load build context
=> >> transferring context: 20.54kB
=> CACHED [2/4] WORKDIR /app
=> [3/4] COPY ... .
=> [4/4] RUN yarn install --production
=> exporting to image
=> >> exporting layers
=> >> writing image sha256:amazing_jones (c1104b9f9b0d5487227004d4ad30bc0c48829cb7699b72d5adbaba905291e184): Bind for 0.0.0.0:3000 failed: port is already allocated.
[+] Building 0.0s (10/10) FINISHED
          docker:desktop-linux
=> [internal] load build definition from Dockerfile
=> >> transferring dockerfile: 151B
=> [internal] load metadata for docker.io/library/node:18-alpine
=> [auth] library/node:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> >> transferring context: 2B
=> [1/4] FROM docker.io/library/node:18-alpine@sha256:77e3b76b47148e28acc84f2e903f34bedc21385
=> [internal] load build context
=> >> transferring context: 20.54kB
=> CACHED [2/4] WORKDIR /app
=> [3/4] COPY ... .
=> [4/4] RUN yarn install --production
=> exporting to image
=> >> exporting layers
=> >> writing image sha256:atiearth@MacBook-Air-khxng-atiearth app %

```

(1) Error ที่เกิดขึ้นหมายความอย่างไร และเกิดขึ้นเพราะอะไร

⇒ เพราะว่า port 3000 มันยังถูกใช้อยู่อีก container นึง

11. ลบ Container ของ Web application เวอร์ชันก่อนแก้ไขออกจากระบบ โดยใช้วิธีเดวิธีหนึ่งดังต่อไปนี้

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 > เพื่อยุดการทำงานของ Container ดังกล่าว

Lab Worksheet

iv. ใช้คำสั่ง \$ docker rm <Container ID ที่ต้องการจะลบ> เพื่อทำการลบ

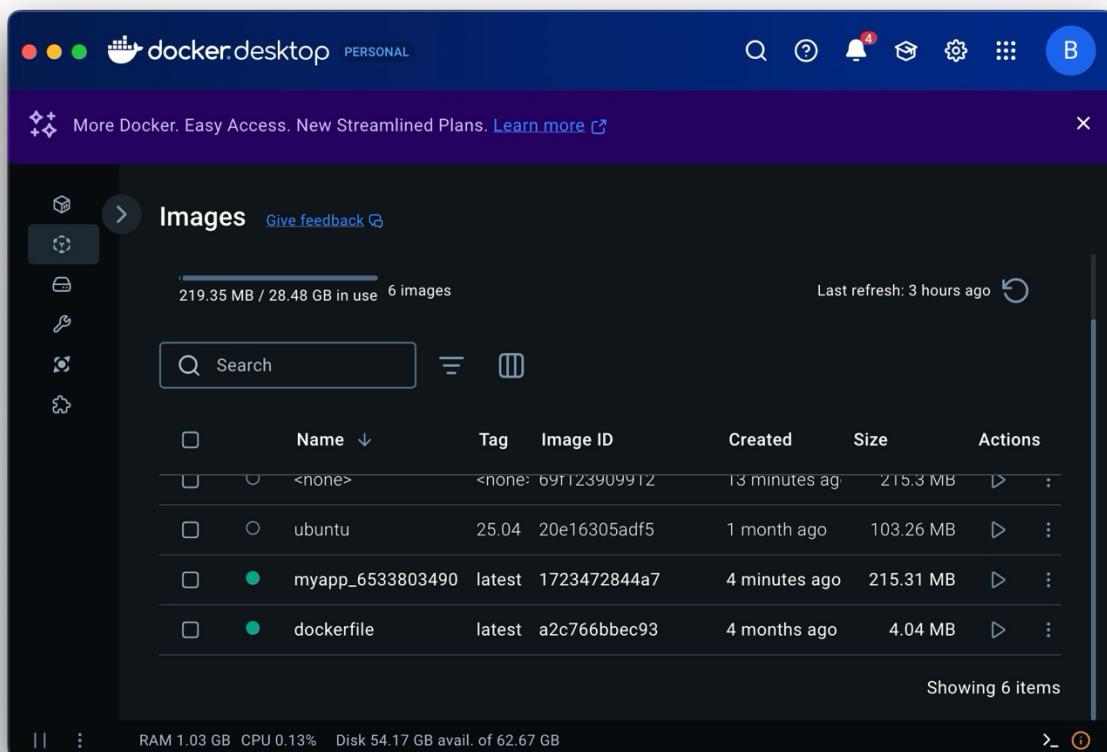
b. ผ่าน Docker desktop

- ไปที่หน้าต่าง Containers
- เลือกไอคอนถังขยะในແກ່ວຂອງ Container ที่ต้องการจะลบ
- ยืนยันโดยการกด Delete forever

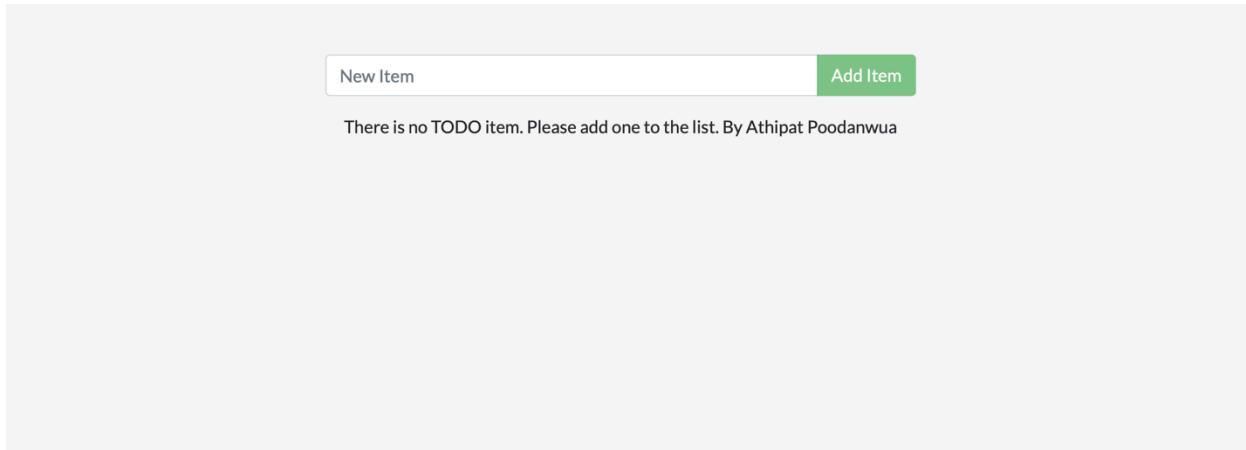
12. Start และรัน Container ตัวใหม่อีกรัง โดยใช้คำสั่งเดียวกันกับข้อ 6

13. เปิด Browser ไปที่ URL = <http://localhost:3000>

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



Lab Worksheet



แบบฝึกปฏิบัติที่ 8.5: เริ่มต้นสร้าง Pipeline อ่าย่างง่ายสำหรับการ 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

Lab Worksheet

```
Lab8_5 — docker run -p 8080:8080 -p 50000:50000 --restart=on-failure -v jenkins_home:/var/j...
adapted
2025-01-23 04:50:21.837+0000 [id=32]    INFO    jenkins.InitReactorRunner$1#onAttained: Loaded all jo
bs
2025-01-23 04:50:21.837+0000 [id=44]    INFO    jenkins.InitReactorRunner$1#onAttained: Configuration
for all jobs updated
2025-01-23 04:50:21.851+0000 [id=60]    INFO    hudson.util.Retrier#start: Attempt #1 to do the actio
n check updates server
2025-01-23 04:50:22.077+0000 [id=42]    INFO    jenkins.install.SetupWizard#init:

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

76eb8159dc1c46d78b3acd56c13d7e11

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

*****
2025-01-23 04:50:28.197+0000 [id=42]    INFO    jenkins.InitReactorRunner$1#onAttained: Completed ini
tialization
2025-01-23 04:50:28.205+0000 [id=25]    INFO    hudson.lifecycle.Lifecycle#onReady: Jenkins is fully
up and running
2025-01-23 04:50:30.437+0000 [id=60]    INFO    h.m.DownloadService$Downloadable#load: Obtained the u
pdated data file for hudson.tasks.Maven.MavenInstaller
2025-01-23 04:50:30.438+0000 [id=60]    INFO    hudson.util.Retrier#start: Performed the action check
updates server successfully at the attempt #1
[]
```

Lab Worksheet

The screenshot shows a 'Getting Started' page for Jenkins. The main title is 'Unlock Jenkins'. A note says: 'To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server: /var/jenkins_home/secrets/initialAdminPassword'. It instructs the user to 'Please copy the password from either location and paste it below.' Below this is a text input field labeled 'Administrator password' containing several asterisks ('*****'). At the bottom right is a blue 'Continue' button.

4. เมื่อได้รับการยืนยันว่า Jenkins is fully up and running ให้เปิดбраузอร์ และป้อนที่อยู่เป็น localhost:8080
5. ทำการ Unlock Jenkins ด้วยรหัสผ่านที่ได้ในข้อที่ 3
6. สร้าง Admin User โดยใช้ username เป็นชื่อจริงของนักศึกษาพร้อมรหัสสี่ตัวท้าย เช่น somsri_3062
[Check point#13] Capture หน้าจอที่แสดงผลการตั้งค่า

Lab Worksheet

Getting Started

Create First Admin User

Username
athipat_3490

Password
••••••

Confirm password
••••••

Full name
Athipat Poodanwua

E-mail address
athipat.p@kkumail.com

Jenkins 2.479.3

Skip and continue as admin

Save and Continue

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

Lab Worksheet

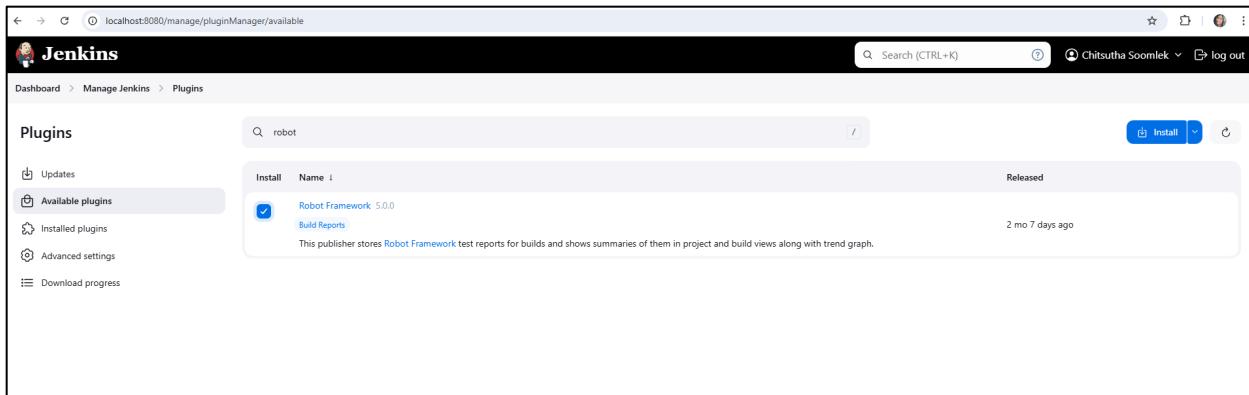
The screenshot shows the Jenkins dashboard at localhost:8080. The main header says "Jenkins". The left sidebar has links for "New Item", "Build History", "Manage Jenkins", and "My Views". A "Build Queue" section shows "No builds in the queue." Below it is a "Build Executor Status" section with "0/2". The central area has a "Welcome to Jenkins!" message: "This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project." It includes buttons for "Create a job" (with a plus sign), "Set up an agent" (with a monitor icon), "Configure a cloud" (with a cloud icon), and "Learn more about distributed builds" (with a question mark icon). At the bottom right, it says "REST API Jenkins 2.479.3".

9. เลือก Manage Jenkins แล้วไปที่เมนู Plugins

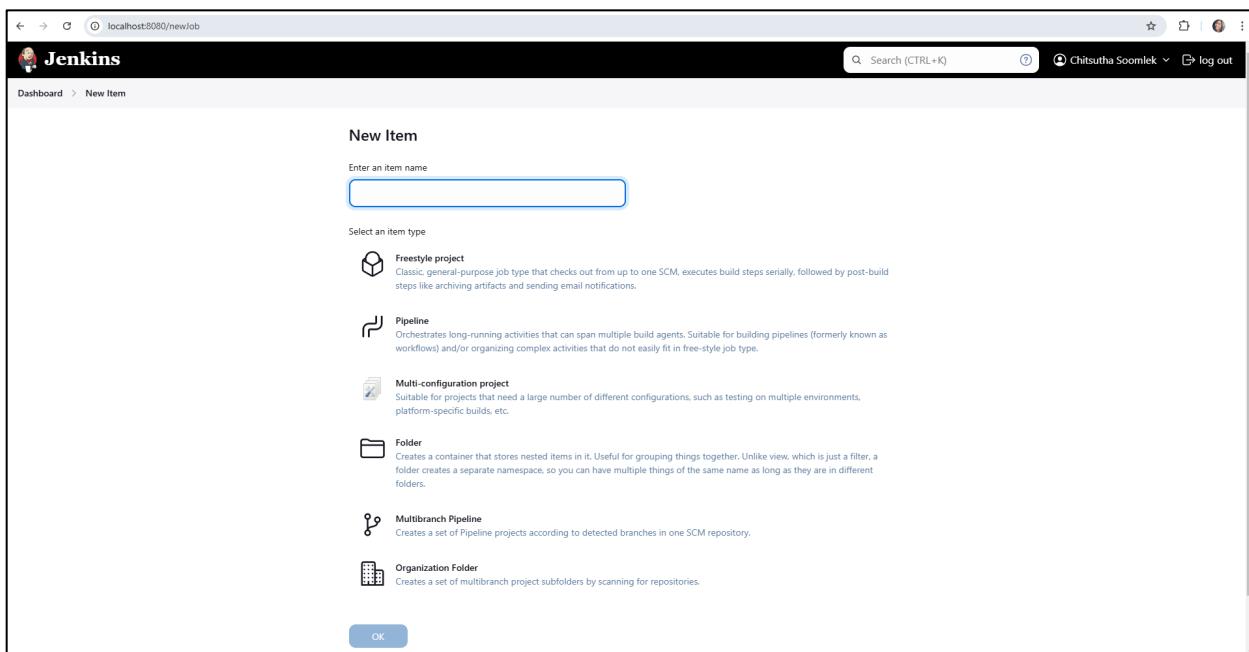
The screenshot shows the Jenkins "Manage Jenkins" page at localhost:8080/manage/. The top navigation bar includes "Dashboard > Manage Jenkins". The main title is "Manage Jenkins". A message box says "It appears that your reverse proxy set up is broken." with "More Info" and "Dismiss" buttons. The page is divided into several sections: "System Configuration" (with icons for System, Tools, Plugins, Nodes, Clouds, Appearance), "Security" (with icons for Security, Credentials, Credential Providers, and Users), and "Status Information" (with icons for System Information, System Log, Load Statistics, and About Jenkins). Each section contains a brief description of its function.

Lab Worksheet

10. ไปที่เมนู Available plugins แล้วเลือกติดตั้ง Robotframework เพิ่มเติม



11. กลับไปที่หน้า Dashboard และสร้าง Pipeline อย่างง่าย โดยกำหนด New item เป็น Freestyle project และตั้งชื่อเป็น UAT



12. นำไฟล์ .robot ที่ทำให้แบบฝึกปฏิบัติที่ 7 (Lab#7) ไปร่วม Repository ของนักศึกษา จากนั้นตั้งค่าที่จำเป็นในหน้านี้ทั้งหมด ดังนี้

Description: Lab 8.5

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

Build Trigger: เลือกแบบ Build periodically และกำหนดให้ build ทุก 15 นาที

Lab Worksheet

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

[Check point#14] Capture หน้าจอแสดงการตั้งค่า พร้อมกับตอบคำถามต่อไปนี้

The screenshot shows the Jenkins configuration interface for a new job. The 'Git' section is selected, displaying the repository URL as `https://github.com/653380349-0Athipat/UAT.git`. The 'Credentials' dropdown is set to '- none -'. An 'Advanced' button is visible. Below this, there's a 'Add Repository' button.

Build Triggers

- Trigger builds remotely (e.g., from scripts) ?
- Build after other projects are built ?
- Build periodically ?

The 'Schedule' field contains the cron expression `H/15 * * * *`.

Would last have run at Wednesday, January 29, 2025 at 4:35:30 AM Coordinated Universal Time; would next run at Wednesday, January 29, 2025 at 4:50:30 AM Coordinated Universal Time.

- GitHub hook trigger for GITScm polling ?
- Poll SCM ?

Lab Worksheet

Build Steps

Execute shell

Command

See [the list of available environment variables](#)

```
. /var/jenkins_home/workspace/UAT/myenv/bin/activate
export PATH=$PATH:/usr/bin
mkdir -p results
robot --outputdir results valid_login.robot
```

Advanced ▾

Add build step ▾

The screenshot shows a GitHub repository named "UAT". The repository has 1 branch and 0 tags. There are 2 commits in the main branch. The first commit was made by "653380349-0Athipat" and updated "resource.robot" and "valid_login.robot". The second commit was made by "resource.robot" and updated "valid_login.robot". The "About" section indicates no description, website, or topics provided. The "Activity" section shows 0 stars, 1 watching, and 0 forks.

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

- ⇒ . /var/jenkins_home/workspace/UAT/myenv/bin/activate
- ⇒ export PATH=\$PATH:/usr/bin
- ⇒ mkdir -p results
- ⇒ robot --outputdir results valid_login.robot

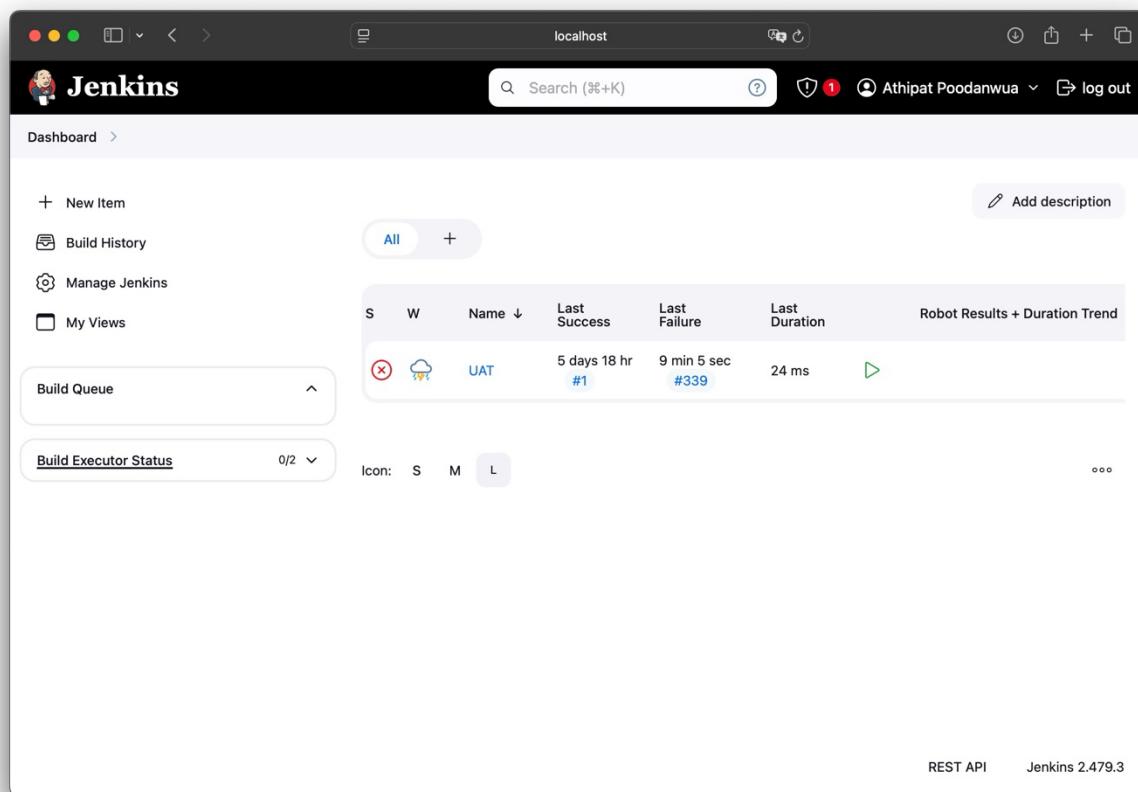
Lab Worksheet

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. สร้าง Build Now

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



The screenshot shows the Jenkins dashboard at localhost. The main area displays a table of builds. One build named "UAT" is highlighted with a red circle and a minus sign, indicating it has failed. The table columns include Status (S), Workstation (W), Name, Last Success, Last Failure, Last Duration, and Robot Results + Duration Trend. The "Last Failure" row for the UAT build shows "#1" and "#339". Below the table, there's a "Build Executor Status" section showing 0/2 executors available, with icons for S, M, and L.

Lab Worksheet

```
+ export PS1
+ VIRTUAL_ENV_PROMPT=(myenv)
+ export VIRTUAL_ENV_PROMPT
+ [ -n -o -n ]
+ export
PATH=/var/jenkins_home/workspace/UAT/myenv/bin:/opt/java/openjdk/bin:/usr/local/sbin:/us
r/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/bin
+ mkdir -p results
+ robot --outputdir results valid_login.robot
=====
Valid Login :: A test suite with a single test for valid login.
=====
Valid Login | FAIL |
Keyword name cannot be empty.
-----
Valid Login :: A test suite with a single test for valid login. | FAIL |
1 test, 0 passed, 1 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
Finished: FAILURE
```