

Workshop

DEVOPS

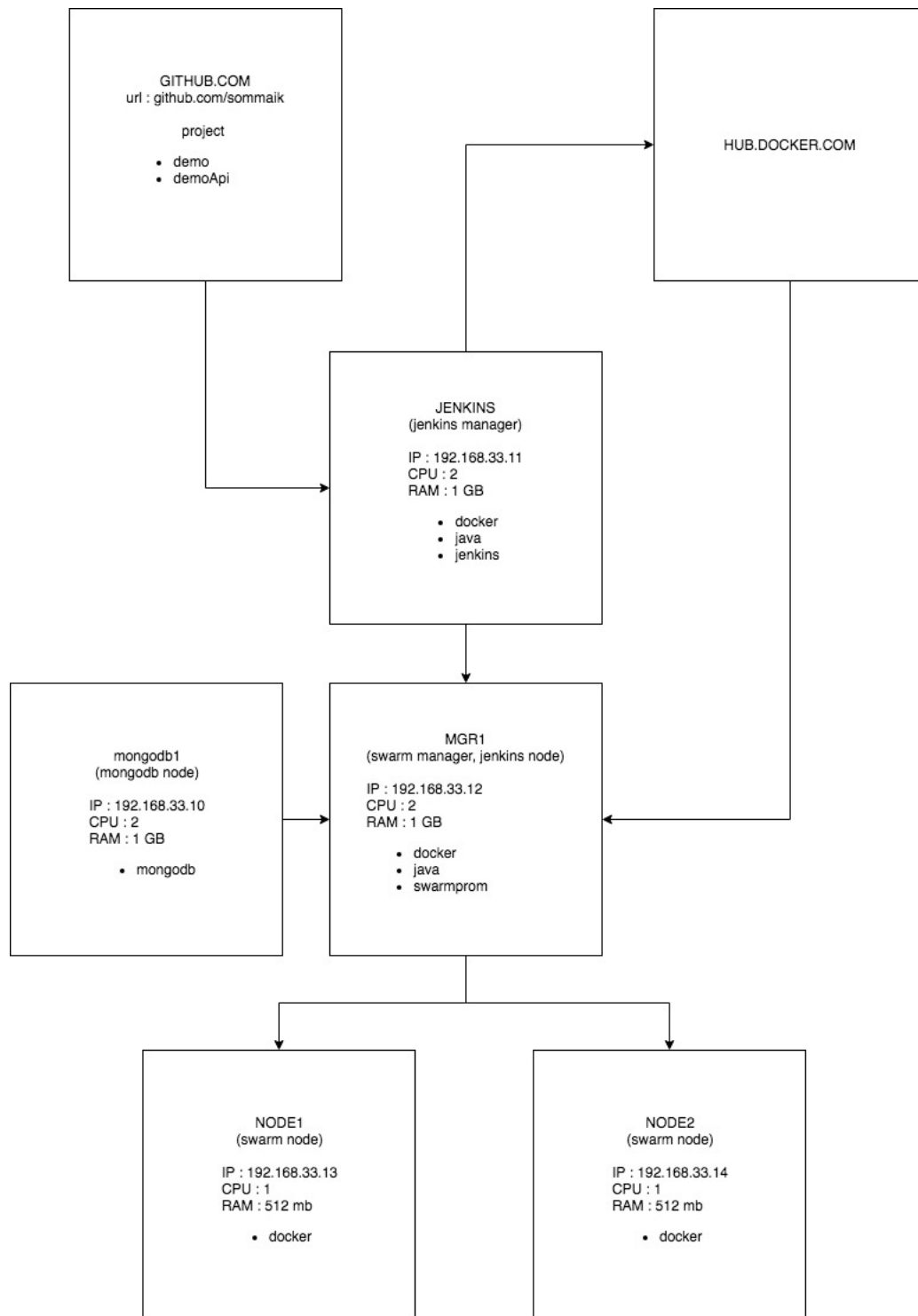
by

PnP Solution

www.pnpsw.com

facebook.com/pnpsolution

System Overview

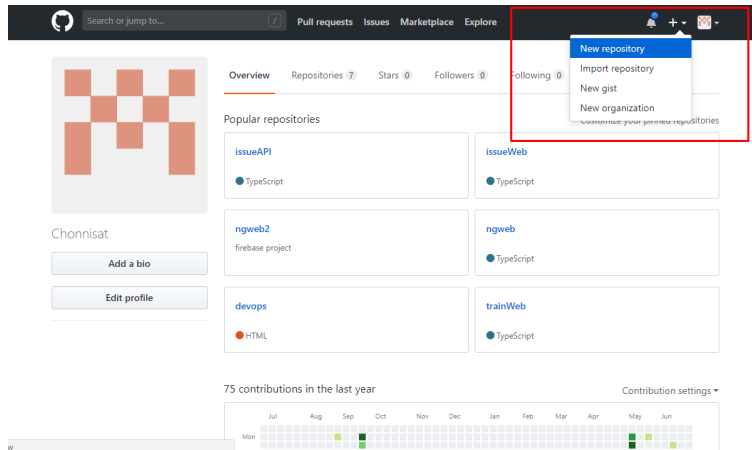


GIT

git#1 สร้าง project ใหม่ชื่อ demo sync code ขึ้นไปบน github.com

วิธีการ

- สร้าง project ใหม่บน github ชื่อ demo ตามขั้นตอนดังนี้
 - เข้า website github.com login ด้วยชื่อตัวเอง
 - กดปุ่ม New repository



- ใส่ชื่อ repository เป็น demo แล้วเลือกเป็น option เป็น public กดปุ่ม create repository

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: Chonnisat / Repository name: Devops-Training ✓

Great repository names are short and memorable. Need inspiration? How about **effective-octo-train**.

Description (optional):

☒ Public
Anyone can see this repository. You choose who can commit.

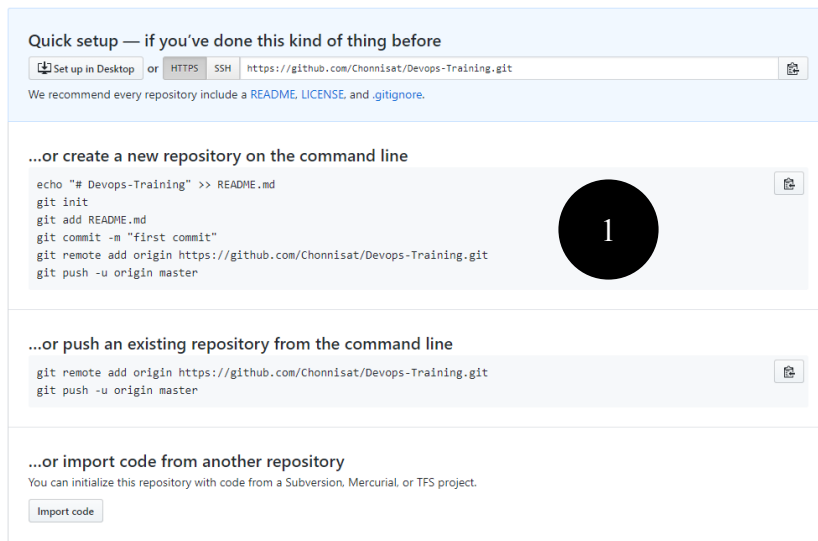
☐ Private
You choose who can see and commit to this repository.

☐ Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

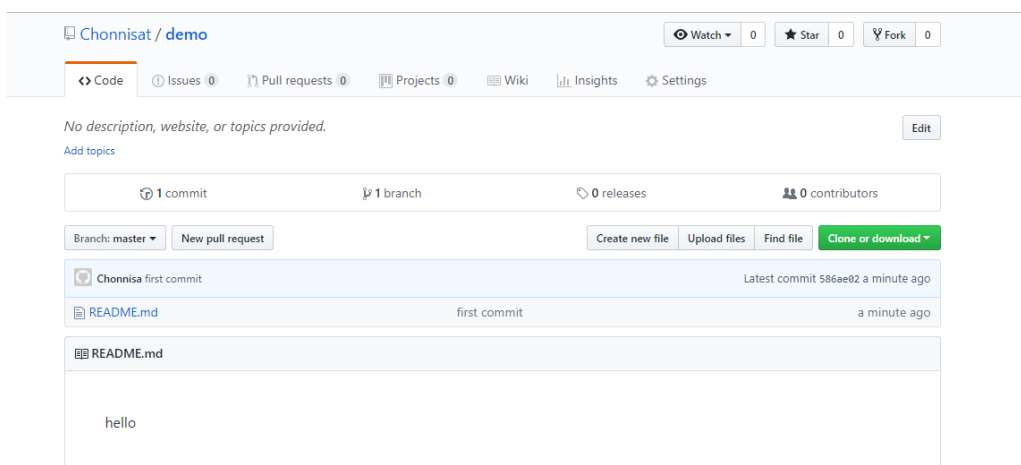
Add .gitignore: None | Add a license: None ⓘ

Create repository

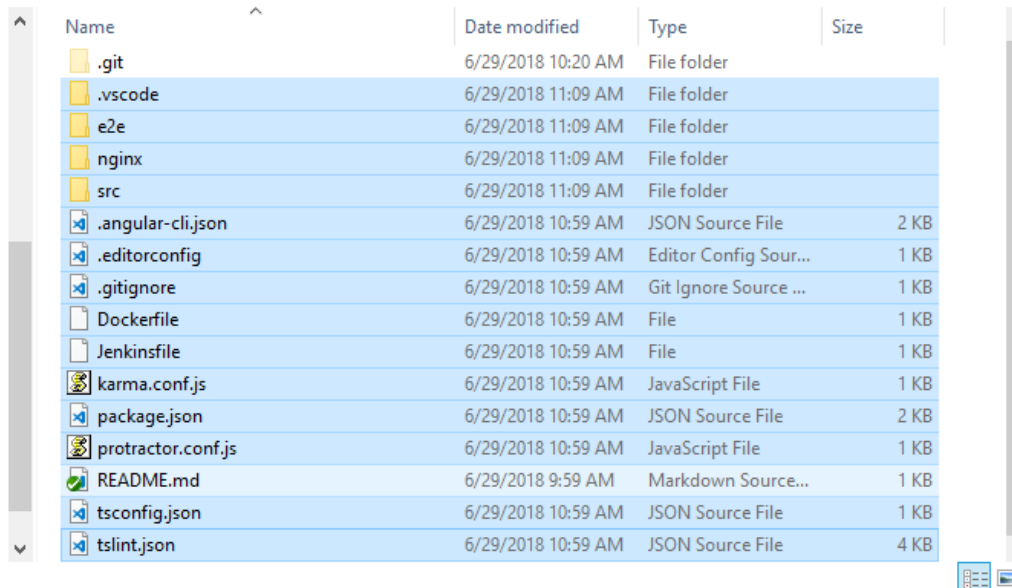
- จะได้ผลลัพธ์ดังนี้



- สร้าง project ใหม่บนเครื่อง ชื่อ demo
 - เปิด terminal ขึ้นมาแล้วรันคำสั่ง ดังนี้
 - mkdir demo
 - cd demo
 - echo hello > README.md
 - ทำให้ project sync กับ github ด้วยการนำคำสั่งในส่วนที่ 1 มา run ดังตัวอย่าง
 - git init
 - git add README.md
 - git commit -m "first commit"
 - git remote add origin <https://github.com/XXX/demo.git>
 - git push -u origin master
- ผลลัพธ์ที่ได้เมื่อเราเข้าไปที่หน้า project เราอีกครั้งบน github

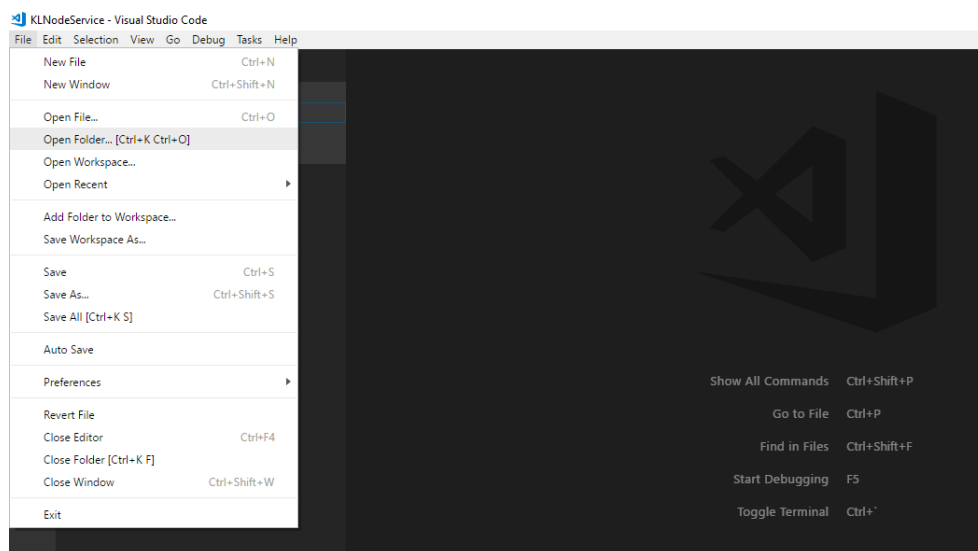


- คัดลอก source code จาก folder /devops/sourcecode/demo ใส่เข้าไปใน project demo ดังภาพ

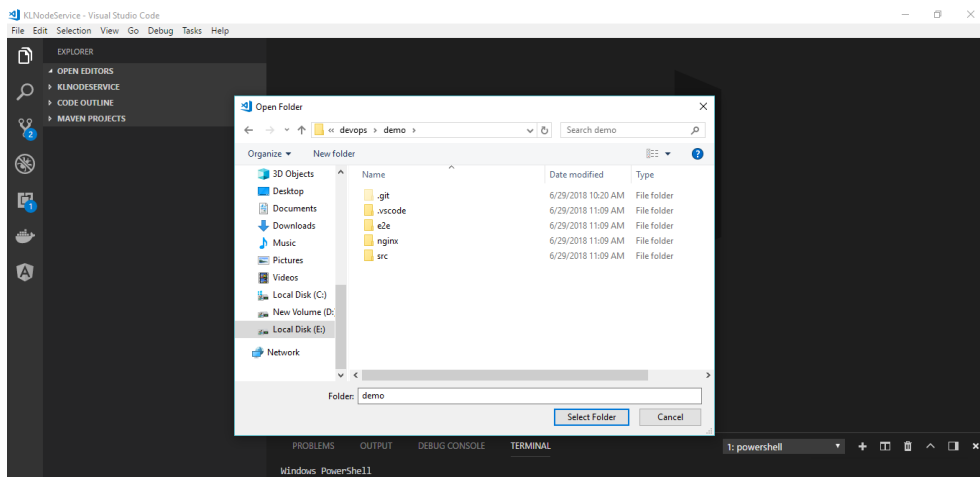


Name	Date modified	Type	Size
.git	6/29/2018 10:20 AM	File folder	
.vscode	6/29/2018 11:09 AM	File folder	
e2e	6/29/2018 11:09 AM	File folder	
nginx	6/29/2018 11:09 AM	File folder	
src	6/29/2018 11:09 AM	File folder	
.angular-cli.json	6/29/2018 10:59 AM	JSON Source File	2 KB
.editorconfig	6/29/2018 10:59 AM	Editor Config Sour...	1 KB
.gitignore	6/29/2018 10:59 AM	Git Ignore Source ...	1 KB
Dockerfile	6/29/2018 10:59 AM	File	1 KB
Jenkinsfile	6/29/2018 10:59 AM	File	1 KB
karma.conf.js	6/29/2018 10:59 AM	JavaScript File	1 KB
package.json	6/29/2018 10:59 AM	JSON Source File	2 KB
protractor.conf.js	6/29/2018 10:59 AM	JavaScript File	1 KB
README.md	6/29/2018 9:59 AM	Markdown Source...	1 KB
tsconfig.json	6/29/2018 10:59 AM	JSON Source File	1 KB
tslint.json	6/29/2018 10:59 AM	JSON Source File	4 KB

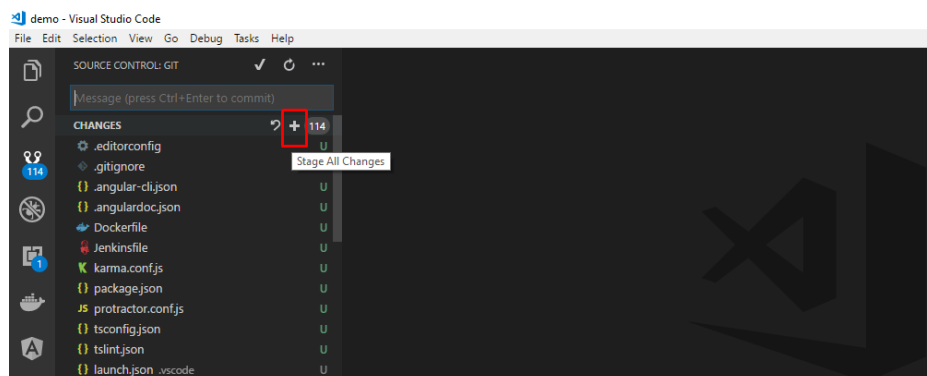
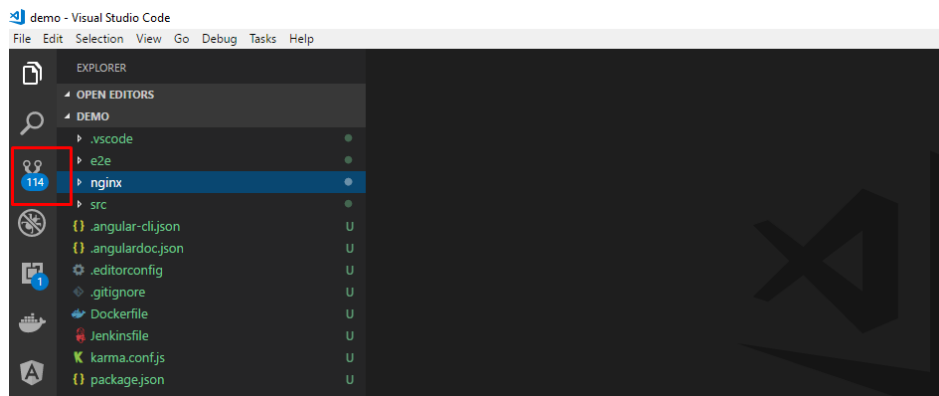
- เปิด program visual studio code เพื่อทำการแก้ไข project
 - เปิด project โดยการกดปุ่ม open



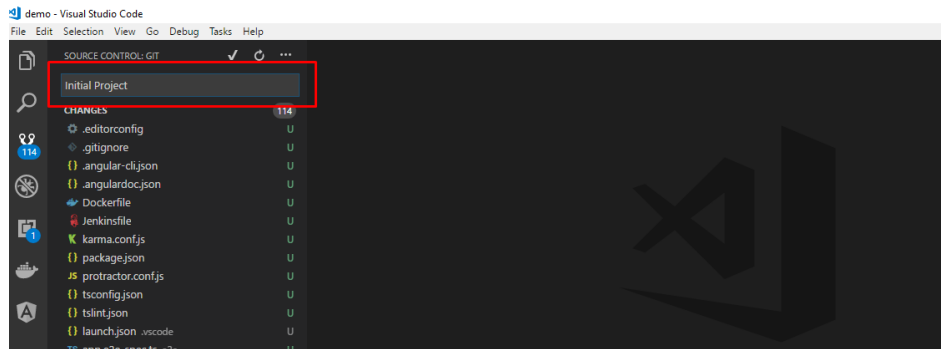
- แล้วเลือก folder ที่เราทำการสร้าง project ไว้



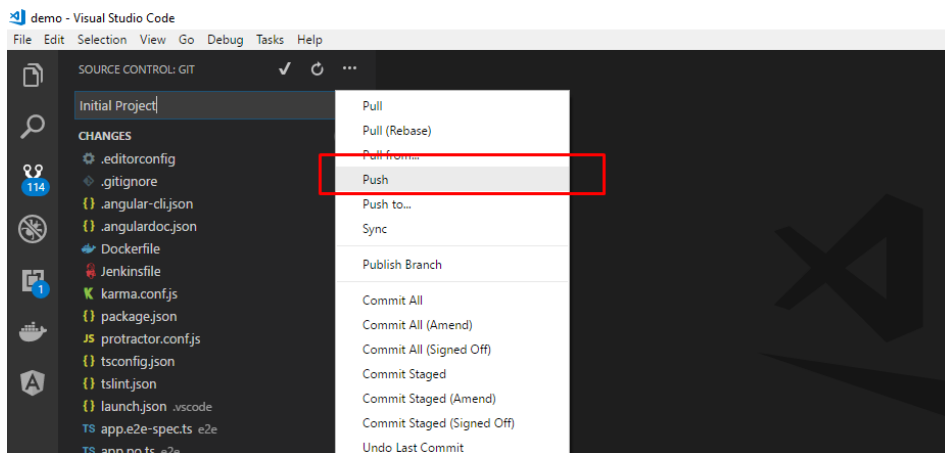
- sync code ขึ้น github ด้วย gui มีวิธีการดังนี้
 - เข้าไปในส่วนของการ sync code โดยการกดปุ่มดังนี้



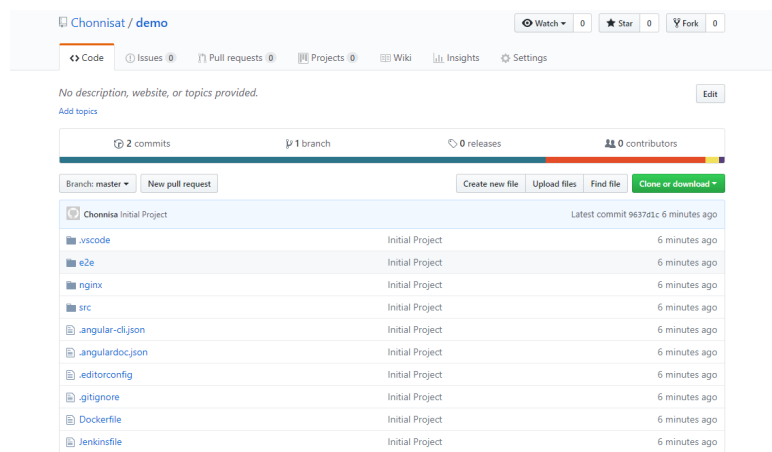
- พิมพ์ message ที่ต้องการจะ commit



- กดปุ่ม เพื่อทำการ push code ขึ้นไปบน github



- ตรวจสอบผลลัพธ์ที่ได้จากการ sync code



git#2 ถึง project vagrant ที่มีอยู่แล้วนำมาใช้
วิธีการ

- เปิด terminal ขึ้นมา แล้วรันคำสั่งดังนี้
 - git clone https://github.com/Sommaik/vagrant_proj vagrant_master

```
E:\Nooti3w\workspace_train\devops>git clone https://github.com/Sommaik/vagrant_proj.git vagrant_master
Cloning into 'vagrant_master'...
remote: Counting objects: 3, done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
Checking connectivity... done.

E:\Nooti3w\workspace_train\devops>
```

git#3 sync code จาก project ที่มีอยู่แล้ว

วิธีการ

- เปิด terminal ขึ้นมาแล้วเข้าไปที่ folder ที่ได้ทำการ clone project ไว้
- run คำสั่งดังนี้เพื่อ update code
 - a. git pull

```
E:\Nooti3w\workspace_train\devops\vagrant_master>git pull
remote: Counting objects: 3, done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/Sommaik/vagrant_proj
   56dd7be..24b0ac9  master    -> origin/master
Updating 56dd7be..24b0ac9
Fast-forward
 README.md | 1 +
 1 file changed, 1 insertion(+)
```


Excercise

git-exercise#1 จงสร้าง project ใหม่ชื่อ demoApi sync code ขึ้นไปบน github.com

**** source code**** เก็บอยู่ที่ /devops/sourcecode/demoApi

<<end git part>>

Docker

docker#1 สร้าง container nginx:alpine ชื่อ web_svr (แบบ full)

วิธีการ

- เปิด terminal ขึ้นมาแล้ว run คำสั่งดังนี้
 - ดึง image nginx:alpine มาเก็บไว้ในเครื่องเพื่อเก็บไว้ใช้ในคราวต่อไป
 - docker pull nginx:alpine

```
C:\Users\somma>docker pull nginx:alpine
alpine: Pulling from library/nginx
ff3a5c916c92: Pull complete
d81b148fab7c: Pull complete
f0fe12447daf: Pull complete
ad017fd52da2: Pull complete
Digest: sha256:4a85273d1e403fbf676960c0ad41b673c7b034204a48cb32779fbb2c18e3839d
Status: Downloaded newer image for nginx:alpine
```

- ตรวจสอบว่ามี images อะไรในเครื่องบ้าง

- docker images

```
C:\Users\somma>docker images
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
nginx                alpine         bc7fdec94612   3 weeks ago    18MB
```

- สร้าง container จาก image ที่ดึงมา

- docker create --name web_svr -p 80:80 nginx:alpine

```
C:\Users\somma>docker create --name web_svr -p 80:80 nginx:alpine
810ec0b2ba33780ecb1be62a275469e7e8867fdb8d2dcf51dbb6c6fff54074f
```

- ตรวจสอบ container ว่าถูกสร้างไว้แล้วหรือยัง

- docker ps -a

```
C:\Users\somma>docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS          NAMES
810ec0b2ba33   nginx:alpine   "nginx -g 'daemon of..." 49 seconds ago Created              web_svr
```

- start docker ที่สร้างเอาไว้

- docker start web_svr

```
C:\Users\somma>docker start web_svr
web_svr
```

- ตรวจสอบ container ว่าทำงานอยู่หรือไม่

- docker ps

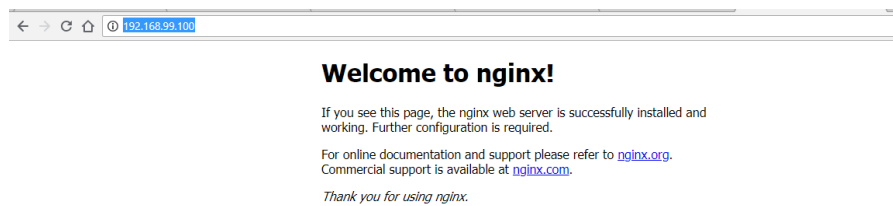
```
C:\Users\somma>docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS          NAMES
810ec0b2ba33   nginx:alpine   "nginx -g 'daemon of..." 3 minutes ago Up 38 seconds  0.0.0.0:80->80/tcp   web_svr
```

- ดู log การทำงานของ container

- docker logs -f web_svr

```
C:\Users\somma>docker logs -f web_svr
192.168.99.1 ~ - [29/Jun/2018:06:38:56 +0000] "GET / HTTP/1.1" 200 612 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.99 Safari/537.36" "-"
2018/06/29 06:38:57 [error] 6#6: *1 open() "/usr/share/nginx/html/favicon.ico" failed (2: No such file or directory), client: 192.168.99.1, server: localhost, request: "GET /favicon.ico HTTP/1.1", host: "192.168.99.100", referer: "http://192.168.99.100/"
192.168.99.1 ~ - [29/Jun/2018:06:38:57 +0000] "GET /favicon.ico HTTP/1.1" 404 571 "http://192.168.99.100/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.99 Safari/537.36" "-"
```

- ทดสอบ web_svr ว่าทำงานถูกหรือไม่
 - เปิด browser url : <http://localhost>, toolbox ให้พิมพ์ url : <http://192.168.99.100>



- ดู log ใน console ว่ามีผลลัพธ์ในการทำงานแสดงออกมา
- ปิดการทำงานของ logs ด้วยการกดปุ่ม ctrl + c

docker#2 สร้าง container nginx:alpine ชื่อ web_svr2 (แบบ short)

วิธีการ

- เปิด terminal ขึ้นมาแล้ว run คำสั่งดังนี้
 - สร้าง container ด้วยคำสั่งดังนี้
 - `docker run --name web_svr2 -p 81:80 -d nginx:alpine`

```
C:\Users\somma>docker run --name web_svr2 -p 81:80 -d nginx:alpine
2e615ca704337b480279748b08bca4c5bf6d9e975b8dcb3e885023a16ec65bd4
```

- ตรวจสอบ container ว่าถูกสร้างไว้แล้วหรือยัง
 - `docker ps -a`

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
2e615ca70433	nginx:alpine	"nginx -g 'daemon of..."	11 minutes ago	Up 11 minutes	0.0.0.0:81->80/tcp	web_svr2
810ec0b2ba33	nginx:alpine	"nginx -g 'daemon of..."	25 minutes ago	Up 22 minutes	0.0.0.0:80->80/tcp	web_svr

- ตรวจสอบ container ว่าทำงานอยู่หรือไม่
 - `docker ps`

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
2e615ca70433	nginx:alpine	"nginx -g 'daemon of..."	13 minutes ago	Up 13 minutes	0.0.0.0:81->80/tcp	web_svr2
810ec0b2ba33	nginx:alpine	"nginx -g 'daemon of..."	28 minutes ago	Up 25 minutes	0.0.0.0:80->80/tcp	web_svr

- ดู log การทำงานของ container
 - `docker logs -f web_svr2`
- ทดสอบ web_svr2 ว่าทำงานถูกหรือไม่
 - เปิด browser url : <http://localhost:81>, toolbox ให้พิมพ์ url : <http://192.168.99.100:81>



- ดู log ใน console ว่ามีผลลัพธ์ในการทำงานแสดงออกมา

```
C:\Users\somma>docker logs -f web_svr2
192.168.99.1 - - [29/Jun/2018:06:59:36 +0000] "GET / HTTP/1.1" 200 612 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.99 Safari/537.36" "-"
2018/06/29 06:59:36 [error] 6#6: *1 open() "/usr/share/nginx/html/favicon.ico" failed (2: No such file or directory), client: 192.168.99.1, server: localhost, request: "GET /favicon.ico HTTP/1.1", host: "192.168.99.100:81", referer: "http://192.168.99.100:81/"
192.168.99.1 - - [29/Jun/2018:06:59:36 +0000] "GET /favicon.ico HTTP/1.1" 404 571 "http://192.168.99.100:81/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.99 Safari/537.36" "-"
```

- ปิดการทำงานของ logs ด้วยการกดปุ่ม `ctrl + c`

docker#3 แก้ไข index.html ใน container web_svr

วิธีการ

- เข้าไปทำงานในเครื่อง container web_svr
 - เปิด terminal ขึ้นมา run คำสั่งดังนี้
 - `docker exec -it web_svr /bin/sh`
- ```
C:\Users\somma>docker exec -it web_svr /bin/sh
/#
```
- แก้ไข file index.html ด้วย vi texteditor
    - `vi /usr/share/nginx/html/index.html`

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
 body {
 width: 35em;
 margin: 0 auto;
 font-family: Tahoma, Verdana, Arial, sans-serif;
 }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
nginx.org.

Commercial support is available at
nginx.com.</p>

<p>Thank you for using nginx.</p>
</body>
</html>
~
~
~
~
```

- พิมพ์ i เพื่อเปิด mode edit
- พิมพ์ <h1>Devops</h1> ต่อจาก <h1>Welcome to Nginx</h1>

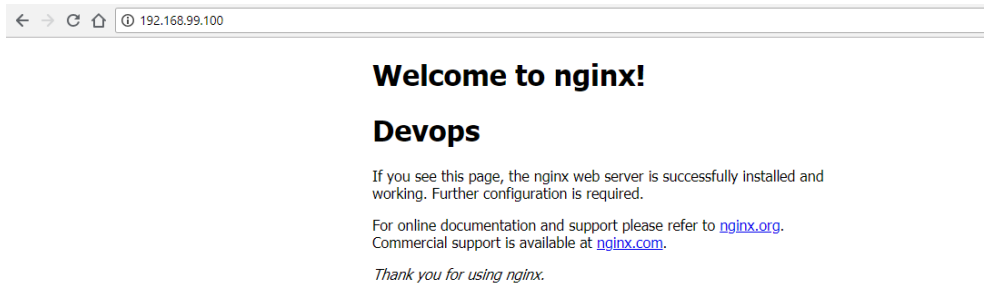
```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
 body {
 width: 35em;
 margin: 0 auto;
 font-family: Tahoma, Verdana, Arial, sans-serif;
 }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<h1>Devops</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
nginx.org.

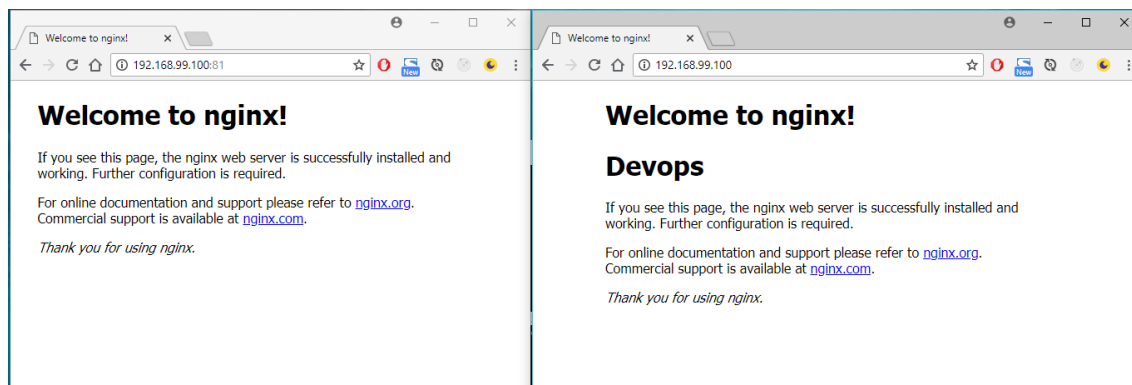
Commercial support is available at
nginx.com.</p>

<p>Thank you for using nginx.</p>
</body>
</html>
~
~
```

- กดปุ่ม esc เพื่อออกจาก mode edit
- พิมพ์ :wq! เพื่อ save และปิด editor
- ทดสอบ web\_svr ว่าทำงานถูกต้องหรือไม่
  - เปิด browser url : <http://localhost>



- เกี่ยวกับการทำงานของ web\_svr2 ว่าแสดงผลเหมือนหรือต่างกันอย่างไร
  - เปิด browser url : <http://localhost:81>



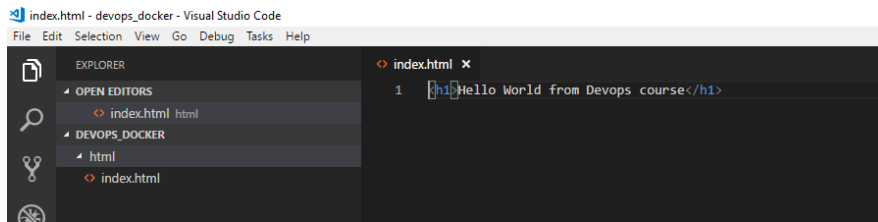
#### docker#4 สร้าง container โดยการอ้างอิงถึง file ที่อยู่ภายในเครื่อง <map volume> วิธีการ

- สร้าง folder เพื่อเก็บ project ชื่อ devops\_docker
  - เปิด terminal ขึ้นมา run คำสั่งดังนี้
    - mkdir devops\_docker
- สร้าง folder เพื่อเก็บ file index.html ของ nginx
  - เปิด terminal ขึ้นมา run คำสั่งดังนี้
    - cd devops\_docker
    - mkdir html

```
E:\Nooti3w\workspace_train\devops>mkdir devops_docker
E:\Nooti3w\workspace_train\devops>cd devops_docker
E:\Nooti3w\workspace_train\devops\devops_docker>mkdir html
```

- สร้าง file ชื่อ index ภายใน folder /devops\_docker/html
  - เปิด visual studio code ขึ้นมาแล้วเลือกไปที่ folder /devops\_docker
  - create new file ชื่อ index.html ภายใน folder /deops\_docker/html

- ใน file index.html ให้มีเนื้อหาดังนี้
- `<h1>Hello World from Devops course</h1>`



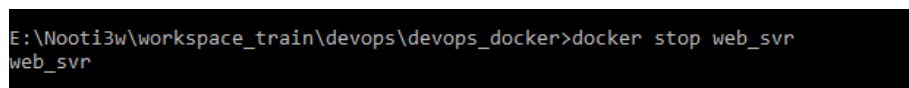
- เปิด terminal ขึ้นมาแล้ว run คำสั่งดังนี้ภายใน folder /devops\_docker
  - สร้าง container ด้วยคำสั่งดังนี้
    - `docker run --name web_svr3 -p 83:80 -v /c/Users/Public/workspace/devops_docker/html:/usr/share/nginx/html -d nginx:alpine`
- ทดสอบ web\_svr ว่าทำงานถูกหรือไม่
  - เปิด browser url : <http://localhost:83>



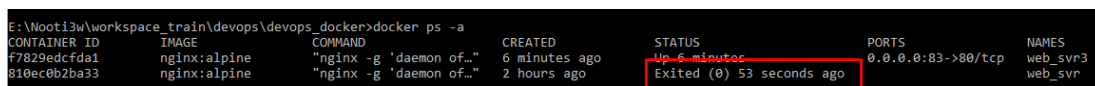
## docker#5 ปิดการทำงานของ container

### วิธีการ

- ปิดการทำงานของ container
  - เปิด terminal ขึ้นมา run คำสั่งดังนี้
    - `docker stop web_svr`



- ตรวจสอบว่า container ปิดไปแล้ว
  - `docker ps -a`



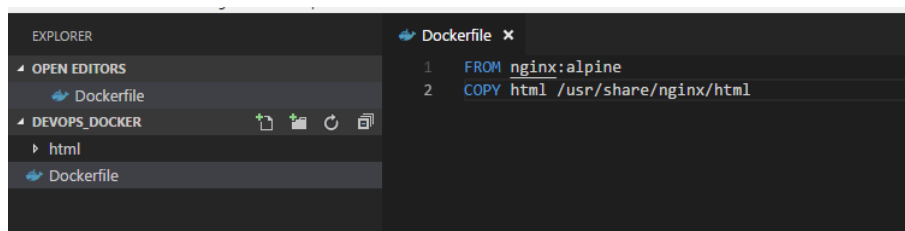
**docker#6** ลบ container web\_svr ออก**วิธีการ**

- ลบ container ออก
  - เปิด terminal ขึ้นมา run คำสั่งดังนี้
    - `docker rm web_svr`

```
E:\Wooti3w\workspace_train\devops\devops_docker>docker rm web_svr
web_svr
```

**docker#7** สร้าง image ของตัวเองขึ้นมา**วิธีการ**

- สร้าง file ชื่อ Dockerfile ภายใต้ folder /devops\_docker
  - เปิด visual studio code ขึ้นมาแล้วเลือกไปที่ folder /devops\_docker
  - create new file ชื่อ Dockerfile
  - ใน file Docker ให้มีเนื้อหาดังนี้
  - FROM nginx:alpine
  - COPY html /usr/share/nginx/html



- build image จาก Dockerfile
  - เปิด terminal ขึ้นมา run คำสั่งดังนี้
    - `docker build -t my_web .`

```
C:\Users\Public\workspace\devops_docker>docker build -t my_web .
Sending build context to Docker daemon 3.584kB
Step 1/2 : FROM nginx:alpine
--> bc7fdec94612
Step 2/2 : COPY html /usr/share/nginx/html
--> f661c36e0782
Successfully built f661c36e0782
Successfully tagged my_web:latest
SECURITY WARNING: You are building a Docker image from Windows against a non-Windows Docker host. All files and directories added to build context will have '-rwxr-xr-x' permissions. It is recommended to double check and reset permissions for sensitive files and directories.
```

- สร้าง container จาก image ตัวเอง โดยการเปิด terminal ขึ้นมาแล้ว run คำสั่งดังนี้
  - สร้าง container ด้วยคำสั่งดังนี้
    - `docker run --name my_web_svr -p 90:80 -d my_web`



```
C:\Users\Public\workspace\devops_docker>docker run --name my_web_svr -p 90:80 -d my_web_73ece97ffb0550a0704554d931702b1ffab8a6b221af77dbc70a5db0420a74f
```

- ตรวจสอบ container ว่าถูกสร้างไว้แล้วหรือยัง
  - `docker ps -a`
- ทดสอบ my\_web\_svr ว่าทำงานถูกหรือไม่
  - เปิด browser url : `http://localhost:90`



**Hello World from Devops course**

**docker#8** นำ image ที่สร้างขึ้นมาไปไว้บน docker hub

### วิธีการ

- สร้าง account บน docker hub
  - เข้า url : `hub.docker.com`
  - ในกรณีที่ยังไม่มี account ให้ทำการสมัครก่อน
- login เข้าใช้ docker hub ด้วยคำสั่งดังนี้
  - เปิด terminal ขึ้นมา run คำสั่งดังนี้
    - `docker login`
    - แล้วใส่ user / password เข้าไป
    - เมื่อสำเร็จจะได้ดังภาพ

```
C:\Users\Public\workspace\devops_docker>docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.
Username: nootiew
Password:
Login Succeeded
```

- สร้าง repository ชื่อ my\_web
  - กดปุ่ม create repository
  - กรอกข้อมูล

- เมื่อเสร็จแล้วจะได้ผลลัพธ์ดังนี้

- การ tag image
  - tag image ด้วยชื่อที่ได้สร้างไว้ เช่น สร้างไว้ชื่อ sommaik/my\_web ก็ใช้คำสั่งดังนี้
    - `docker tag my_web sommaik/my_web:latest`
  - ตรวจสอบว่าทำการ tag ถูก ด้วยคำสั่งดังนี้
    - `docker images`

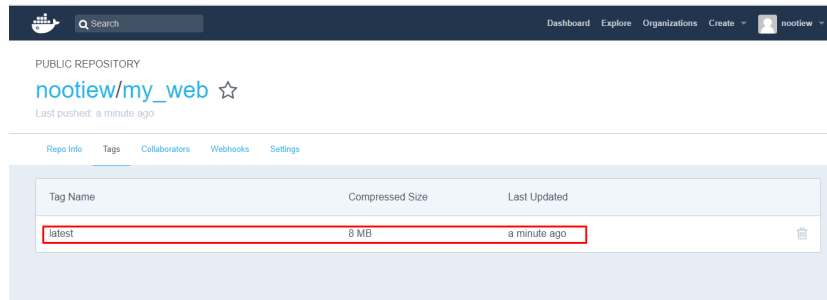
```
C:\Users\Public\workspace\devops_docker>docker tag my_web nootiew/my_web:latest

C:\Users\Public\workspace\devops_docker>docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
my_web latest f661c36e0782 22 minutes ago 18MB
nootiew/my_web latest f661c36e0782 22 minutes ago 18MB
nginx alpine bc7fdec94612 3 weeks ago 18MB
```

- push image ขึ้นไปเก็บบน docker hub repository
  - run คำสั่งดังนี้
    - `docker push sommaik/my_web`

```
C:\Users\Public\workspace\devops_docker>docker push nootiew/my_web
The push refers to repository [docker.io/nootiew/my_web]
c7199f4a251a: Pushed
951c1d7bace7: Mounted from library/nginx
91295ee17337: Mounted from library/nginx
423678709065: Mounted from library/nginx
cd7100a72410: Mounted from library/nginx
latest: digest: sha256:243f698aa5810fac9f678ce0d9c14629c7fb98f57fbfc0cf1863392a17a530b7 size: 1360
```

- ตรวจสอบว่า image อยู่บน docker hub ด้วยการเข้าไปที่ web แสดงรายการ image ของเราดังนี้



**docker#9** ลบ image my\_web ออก **\*\*จะลบออกได้เฉพาะในกรณีที่ไม่มีการใช้งานแล้ว\*\***  
**วิธีการ**

- ลบ image ออกด้วยคำสั่ง
  - เปิด terminal ขึ้นมา run คำสั่งดังนี้
    - `docker rmi my_web`

```
C:\Users\Public\workspace\devops_docker>docker rmi my_web
Untagged: my_web:latest
```

## Exercise

**docker-exercise#1** จงสร้าง nginx:alpine container ชื่อ web\_svr\_ex ใช้ port 88 โดยที่ file index.html มีเนื้อหาดังนี้

```
<html>
 <header>
 <title>Devops Workshops</title>
 </header>
 <body>
 <h1>Hello World</h1>
 <h2>This is a Devops Workshops</h2>
 <h3>Docker exercise #1</h3>
 </body>
</html>
```

**docker-exercise#2** จงสร้าง image ชื่อ my\_web\_ex แล้วเก็บไว้บน docker hub ชื่อ

<<your\_name>>/my\_web\_ex

**docker-exercise#3** จงสร้าง image ชื่อ demo แล้วเก็บไว้บน docker hub ชื่อ

<<your\_name>>/demo

**docker-exercise#4** จงสร้าง image ชื่อ demoApi แล้วเก็บไว้บน docker hub ชื่อ

<<your\_name>>/demoApi

**docker-exercise#5** จงนำ project devops\_docker ขึ้นไปเก็บไว้บน github ชื่อ devops\_docker

**\*\*<<your\_name>>\*\*** คือชื่อ account ที่สร้างไว้บน docker hub

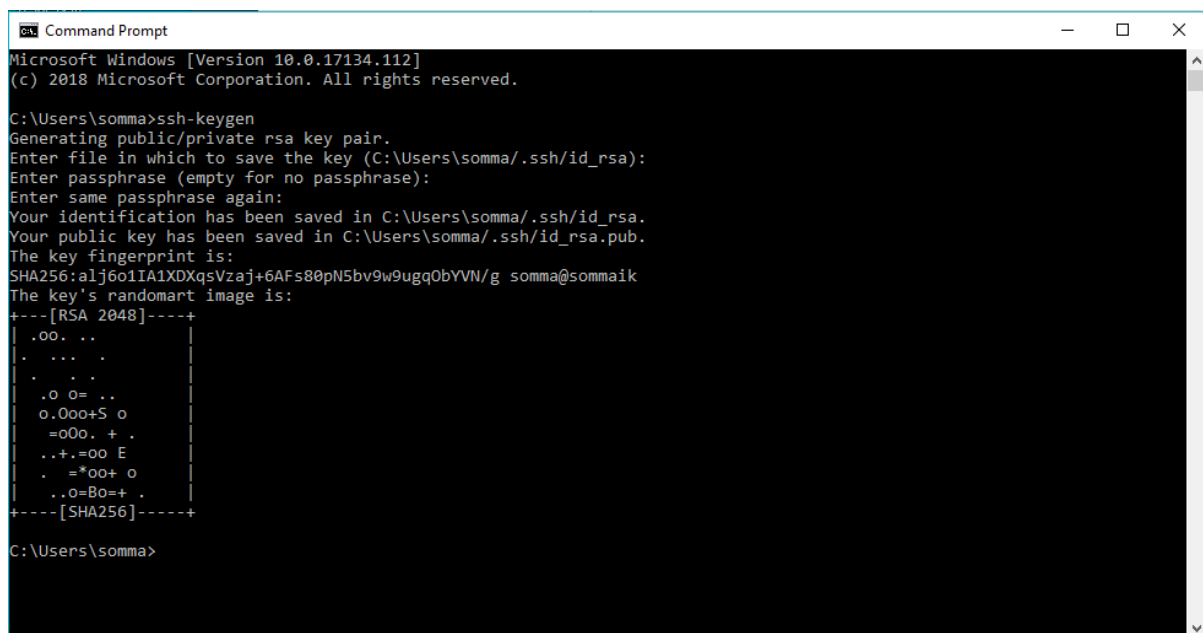
<<end docker part>>

## Vagrant

**vagrant#1** สร้าง project ใหม่เพื่อทำการเก็บข้อมูลการ config ของ vagrant

### วิธีการ

- initial vagrant project
  - เปิด terminal ขึ้นมาแล้ว run คำสั่ง ดังนี้
    - mkdir vagrant\_proj
    - cd vagrant\_proj
    - vagrant init
- สร้าง ssh private key เพื่อใช้ในการ authentication
  - run คำสั่งใน terminal ดังนี้
    - ssh-keygen
    - กด enter ไปตลอดจนจบขั้นตอน



```

C:\Users\somma>ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (C:\Users\somma\.ssh\id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in C:\Users\somma\.ssh\id_rsa.
Your public key has been saved in C:\Users\somma\.ssh\id_rsa.pub.
The key fingerprint is:
SHA256:alJ6o1IA1XDXqsVzaj+6AFs80pN5bv9w9ugqObYVN/g somma@sommaik
The key's randomart image is:
+---[RSA 2048]-----+
|.oo. ..
|. . . .
|. . . .
|.O O= ..
|.O.Oo+S o
|=Oo. + .
|..+.oo E
|. =*oo+ o
|.O=BO=+ .
+---[SHA256]-----+
C:\Users\somma>

```

- นำ ssh private key ไปเรียกใช้ใน Vagrant
  - แก้ไข file Vagrantfile ดังนี้

```

-*- mode: ruby -*-
vi: set ft=ruby :
VAGRANTFILE_API_VERSION = "2"
Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|
 config.vm.box = "ubuntu/xenial64"
 config.ssh.insert_key = false
 config.ssh.private_key_path = ["~/.ssh/id_rsa", "~/.vagrant.d/insecure_private_key"]
 config.vm.provision "file", source: "~/.ssh/id_rsa.pub", destination: "~/.ssh/authorized_keys"
 config.vm.provision "file", source: "~/.ssh/id_rsa", destination: "~/.ssh/id_rsa"
 config.vm.provision "file", source: "~/.ssh/id_rsa.pub", destination: "~/.ssh/id_rsa.pub"
 config.vm.provision "shell", inline: <<-EOC
 sudo sed -i -e "\/#PasswordAuthentication yes#PasswordAuthentication no#g" /etc/ssh/sshd_config
 sudo service ssh restart
 EOC
your code here
end

```

**vagrant #2** สร้าง vm สำหรับทำ Jenkins (เริ่มจาก base)**วิธีการ**

- config ขั้นตอนในการสร้าง vm
  - ใส่ script สำหรับการติดตั้ง Jenkins โดยการแก้ไข file Vagrantfile โดยเพิ่มวิธีการเข้าไป ดังนี้

```
$jenkins_script = <<-SCRIPT
 wget -q -O - https://pkg.jenkins.io/debian/jenkins-ci.org.key | sudo apt-key add -
 sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'
 sudo apt-get update
 sudo apt-get install default-jre -y
 sudo apt-get install jenkins -y
 sudo usermod -aG docker jenkins
 sudo service jenkins restart
SCRIPT

$docker_script = <<-SCRIPT
 sudo apt-get install apt-transport-https ca-certificates curl software-properties-common -y
 curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
 sudo apt-key fingerprint 0EBFCD88
 sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
 sudo apt-get update
 sudo apt-get install docker-ce -y
 sudo groupadd docker
 sudo usermod -aG docker vagrant
 sudo systemctl enable docker
 sudo curl -L https://github.com/docker/compose/releases/download/1.21.2/docker-compose-$(uname -s)-$(uname -m) -o
 /usr/local/bin/docker-compose --silent
 sudo chmod +x /usr/local/bin/docker-compose
SCRIPT

$jenkins_node_script = <<-SCRIPT
 sudo apt-get update
 sudo apt-get install default-jre -y
SCRIPT
```

- config รูปแบบการสร้าง vm ชื่อ jenkins ip : 192.168.33.11 default ram, cpu ดังนี้

```
config.vm.define "jenkins" do |machine|
 machine.vm.host_name = "jenkins"
 machine.vm.network :private_network, ip: "192.168.33.11"
 machine.vm.provision "shell", inline: $jenkins_script
 machine.vm.provision "shell", inline: $docker_script
end
```

- start vagrant ด้วยคำสั่งดังนี้
  - vagrant up jenkins
- remote เข้าไปที่เครื่อง jenkins ด้วยคำสั่ง ดังนี้
  - vagrant ssh jenkins
- ออกจาก remote ด้วยคำสั่งดังนี้
  - exit
- ดูสถานะของ vm ทั้งหมดด้วยคำสั่งดังนี้
  - vagrant status

**vagrant #3** เก็บ vagrant vm ที่สร้างเอาไว้เพื่อใช้ในการสร้าง vm อื่นๆ ที่คล้ายกัน**วิธีการ**

- pack vm ที่ต้องการ โดยมีขั้นตอนดังนี้

- run คำสั่งใน terminal ภายใต้ folder ที่เก็บ Vagrantfile
- พิมพ์คำสั่งดังนี้เพื่อ pack vm ที่ต้องการ
  - `vagrant package --output jenkins.box jenkins`

**vagrant #4** import box ที่สร้างไว้นำมาใช้งานใน vagrant

#### วิธีการ

- add box ที่สร้างไว้เข้าไปใน vagrant box
  - run คำสั่งใน terminal ภายใต้ folder ที่เก็บ jenkins.box
  - พิมพ์คำสั่งดังนี้เพื่อ add box เข้าไป
    - `vagrant box add --name jenkins jenkins.box`

## Exercise

**vagrant-exercise #1** สร้าง vm สำหรับทำ Docker swarm manager node โดยมีรายละเอียดดังนี้

- ชื่อ: mgr1
- script: \$jenkins\_node\_script, \$docker\_script
- ip: 192.168.33.12

**vagrant-exercise#2** สร้าง vm สำหรับทำ Docker swarm worker node โดยมีรายละเอียดดังนี้

- ชื่อ node1
- script: \$docker\_script
- ip: 192.168.33.13
- cpu: 1
- ram: 512

**vagrant-exercise#3** สร้าง vm สำหรับทำ Docker swarm worker node โดยมีรายละเอียดดังนี้

- ชื่อ node2
- script: \$docker\_script
- ip: 192.168.33.14
- cpu: 1
- ram: 512

**vagrant-exercise#4** สร้าง script สำหรับติดตั้ง mongodb โดยมีรายละเอียดดังนี้

```
$mongodb_script = <<-SCRIPT
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 9DA31620334BD75D9DCB49F368818C72E52529D4
echo "deb [arch=amd64,arm64] https://repo.mongodb.org/apt/ubuntu xenial/mongodb-org/4.0 multiverse" | sudo tee
/etc/apt/sources.list.d/mongodb-org-4.0.list
sudo apt-get update
sudo apt-get install -y mongodb-org
sudo service mongod start
SCRIPT
```

**vagrant-exercise#5** สร้าง vm สำหรับทำ mongodb โดยมีรายละเอียดดังนี้

- ชื่อ mongodb1
- script: \$mongodb\_script
- ip: 192.168.33.10

**vagrant-exercise#6** pack vm mgr1 และเก็บไว้ในชื่อ docker\_manager.box

**vagrant-exercise#7** pack vm node1 และเก็บไว้ในชื่อ docker\_node.box

**vagrant-exercise#8** pack vm mongodb และเก็บไว้ในชื่อ mongodb.box



**\*\*วิธีการ set ค่า cpu, ram\*\***

```
machine.vm.provider "virtualbox" do |v|
 v.memory = 512
 v.cpus = 1
end
```

<<end Vagrant part>>

## Docker Swarm

**swarm#1** สร้าง docker swarm manager

### วิธีการ

- remote เข้าเครื่อง mgr1
  - run คำสั่งใน terminal ดังนี้
    - cd vagrant\_proj
    - vagrant ssh mgr1
- กำหนดให้เครื่อง mgr เป็น swarm manager
  - run คำสั่งใน terminal ดังนี้
    - docker swarm init --advertise-addr=192.168.33.12
    - เมื่อ run เสร็จจะได้ command มาเพื่อให้เครื่อง worker ใช้ในการ join

**swarm#2** เพิ่ม worker node1 เข้าไปใน swarm manager

### วิธีการ

- remote เข้าเครื่อง node1
  - run คำสั่งใน terminal ดังนี้
    - cd vagrant\_proj
    - vagrant ssh node1
- กำหนดให้เครื่อง mgr เป็น swarm worker
  - run คำสั่งใน terminal ดังนี้
    - docker join --token=<your manager token>

**swarm#3** สร้าง service my\_web\_ex ที่ mgr1 (swarm manager)

### วิธีการ

- remote เข้าเครื่อง mgr1
  - run คำสั่งใน terminal ดังนี้
    - cd vagrant\_proj
    - vagrant ssh mgr1
- สร้าง service บนเครื่อง mgr1 โดยที่ให้ run เฉพาะเครื่อง worker เท่านั้น
  - run คำสั่งใน terminal ดังนี้
    - docker service create --replicas 2 --name my\_web\_ex --constraint "node.role != manager" --publish 80:80  
<<your\_docker\_hub\_id>>/my\_web\_ex:latest

**swarm#5** scale service my\_web\_ex ที่ mgr1 (swarm manager)

#### วิธีการ

- scale service my\_web\_ex ในเครื่อง mgr1 ให้เป็น 4
  - run คำสั่งใน terminal ดังนี้
    - docker service scale my\_web\_ex=4
- ตรวจสอบว่า สถานะ service
  - run คำสั่งใน terminal ดังนี้
    - docker service ls
    - docker service ps my\_web\_ex

**swarm#6** delete service my\_web\_ex ที่ mgr1 (swarm manager)

#### วิธีการ

- delete service my\_web\_ex ในเครื่อง mgr1
  - run คำสั่งใน terminal ดังนี้
    - docker service rm my\_web\_ex
- ตรวจสอบว่า สถานะ service
  - run คำสั่งใน terminal ดังนี้
    - docker service ls

## Exercise

**swarm-exercise#1** เพิ่ม worker node2 เข้าไปใน swarm manager

**swarm-exercise#2** สร้าง service ชื่อ demo โดยทำงานบน worker เท่านั้น 2 instance

**swarm-exercise#3** สร้าง service ชื่อ domoApi โดยทำงานบน worker เท่านั้น 2 instance

<<end swarm part>>

## Grafana

**grafana#1** ติดตั้ง grafana บนเครื่อง swarm manager

### วิธีการ

- remote เข้าเครื่อง mgr1
  - run คำสั่งใน terminal ดังนี้
    - cd vagrant\_proj
    - vagrant ssh mgr1
- ติดตั้ง grafana ในเครื่อง manager
  - run คำสั่งใน terminal ดังนี้
    - git clone https://github.com/stefanprodan/swarprom.git
    - cd swarprom
    - run คำสั่งเพื่อสร้าง grafana บนเครื่อง manager

ADMIN\_USER=admin \

ADMIN\_PASSWORD=admin \

SLACK\_URL=https://hooks.slack.com/services/TOKEN \

SLACK\_CHANNEL=devops-alerts \

SLACK\_USER=alertmanager \

docker stack deploy -c docker-compose.yml mon

- ทดสอบหลังการติดตั้ง
  - ตรวจสอบว่า service ได้ทำการ start ครบทุก service ไหม ด้วยคำสั่งดังนี้
    - docker service ls
    - โดยดูว่า service ได้มีครบไหม ดังภาพ

ID	NAME	MODE	REPLICAS	IMAGE	PORTS
7thxovmrlxq	mon_alertmanager	replicated	1/1	stefanprodan/swarprom-alertmanager:v0.14.0	
owuanok2h58s	mon_caddy	replicated	1/1	stefanprodan/caddy:latest	*:3000->3000/tcp, *:9090->9090/tcp, *:9090->9090/tcp, *:9090->9090/tcp
tf6h8v9mimjh	mon_cadvisor	global	2/2	google/cadvisor:latest	
i41z8uo8hqs3	mon_dockerd-exporter	global	2/2	stefanprodan/caddy:latest	
o0ju7fmdh9z3	mon_grafana	replicated	1/1	stefanprodan/swarprom-grafana:5.0.1	
32vbhg2s7jqm	mon_node-exporter	global	2/2	stefanprodan/swarprom-node-exporter:v0.15.2	
o4t5o44g190s	mon_prometheus	replicated	1/1	stefanprodan/swarprom-prometheus:v2.2.0-rc.0	

- เปิด browser
  - เข้าไปที่ url <http://192.168.33.12:3000>
  - ใส่ user = admin, password = admin













<<end grafana part>>

## Jenkins

**jenkins#1** เพิ่ม node mgr1 เข้าไปที่ jenkins

### วิธีการ

- remote เข้าเครื่อง jenkins
  - run คำสั่งใน terminal ดังนี้
    - `cd vagrant_proj`
    - `vagrant ssh jenkins`
- ทดสอบว่าเครื่อง jenkins เห็นเครื่อง mgr1 ไหม
  - remote ไปเครื่อง mgr1 จากเครื่อง jenkins
    - `ssh vagrant@192.168.33.12`
    - ตอบ yes
  - ทดสอบว่าอยู่ในเครื่อง mgr1 จริงด้วยคำสั่ง
    - `ip addr`
    - ดูว่าเป็น ip 192.168.33.12 หรือไม่
  - ออกจากเครื่อง mgr1 ด้วยคำสั่ง
    - `exit`
- หาค่า private key ของ ด้วยคำสั่ง ดังนี้
  - `cat ~/.ssh/id_rsa`
- เข้า jenkins web console
  - เปิด browser
    - เข้า url : <http://192.168.33.11:8080>
    - ใส่ user / password
  - สร้าง credentials โดยเข้าที่เมนู Credentials > global > Add Credentials

-  New Item
-  People
-  Build History
-  Project Relationship
-  Check File Fingerprint
-  Manage Jenkins
-  My Views
-  Open Blue Ocean
-  **Credentials** 
-  System
-  New View

**Build Queue**


No builds in the queue.

**Build Executor Status**

- 1 Idle
- 2 Idle


Stores scoped to **Jenkins**



P	Store ↓	Domains
	Jenkins	 (global) 




# Jenkins

[Jenkins](#) ▶ [Credentials](#) ▶ [System](#) ▶ [Global credentials \(unrestricted\)](#) ▶




 [Back to credential domains](#)

 [Add Credentials](#) 



## Global

Credentials that should

	<a href="#">tiewv</a>
	<a href="#">somi</a>
	<a href="#">core</a>

Icon: [S](#) [M](#) [L](#)

- กรอกข้อมูลดัง form ด้านล่าง

Scope

Global (Jenkins, nodes, items, all child items, etc)

Username

vagrant

Private Key

☒ Enter directly

Key

-----BEGIN RSA PRIVATE KEY-----  
MIIEpAIBAAKCAQEA5dW8Gr9579U/qrOdZdUxekdoBIBT2LpXVnmWLYkN0MUYFYg7  
UldH59y/WUj7vX3sMmguUke8aV4RA26EzMGHfWbydUbVMWMI29n7nN4ulI4NwAUt  
LlTUSu2wQ63Z5bNEQZ3CB/Pv9mFLjr1APFiV4nErqA1EJhuE6tirpyW6klwHe/G  
Ve01fw+x/WCxFZnUbO/18hxFdWRGnC717MtvFs5I4bEGZODzPjmtR6VONJmUE1Ce  
Vo8M0MRPEZVuuTd9PO2B4VhdTmnwUbsLCuFe12refeUTb5NVOYS37SP1VxbRK3b2  
5/rKYZphcJr0lZqBAu/fVj3bl0zMBolrkRFM3QIDAQABAOlBAQCmxOoNk3T9beX8  
jHAW/2Hx4X8voSD2O61doR3JmMM/k2Qwa5Ov4ck5QwJNhRm1m+OBh/rklUzvn+P0  
e4M9F8h6g/K2ilkz5sqMzJU4dDeUb6cmiBsKohZ98A6QbVvltVqT1B1eNnMCzIK  
L6SmpRiWxs7Cb1lR22k0OI34aPf/LfuWuue0ZINBqQ6PHLtps00itPJPw4WbZrk

Passphrase

ID

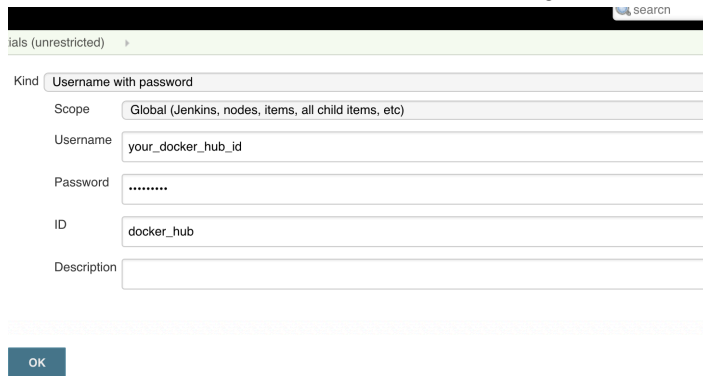
vagrant

Description

Save

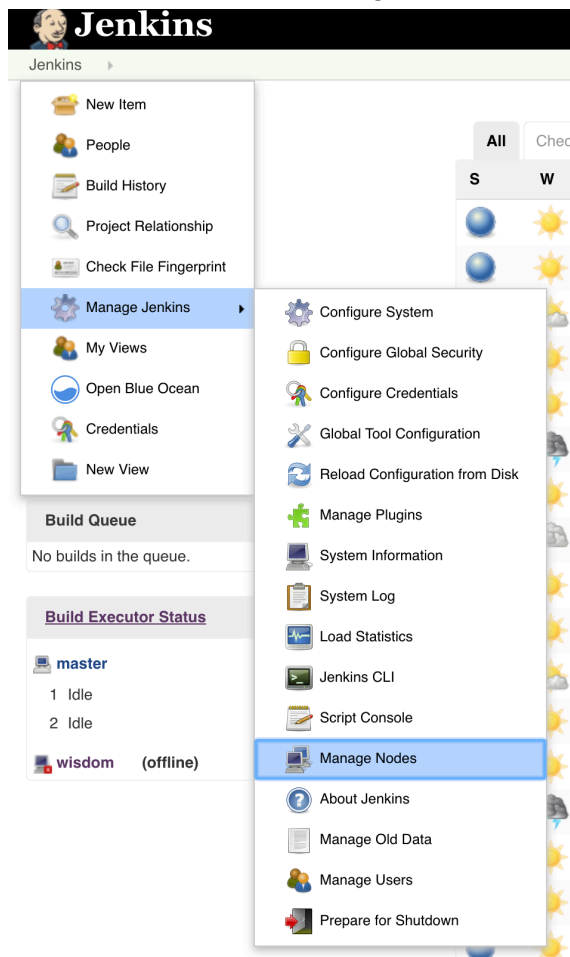


- สร้าง Credential เพื่อเก็บข้อมูล user / password สำหรับ dockerhub โดยมีขั้นตอนดังนี้
  - Add Credentials แล้วกรอกข้อมูลดัง form




The screenshot shows the 'Add Credentials' form in Jenkins. The 'Kind' is set to 'Username with password'. The 'Scope' is 'Global (Jenkins, nodes, items, all child items, etc)'. The 'Username' is 'your\_docker\_hub\_id'. The 'Password' is masked with dots. The 'ID' is 'docker\_hub'. The 'Description' field is empty. There is an 'OK' button at the bottom left.

- สร้าง node ใหม่ดังนี้
  - เข้า menu Manage Nodes



- เลือก New Node

 [Back to Dashboard](#) [Manage Jenkins](#) [New Node](#) [Configure](#)

S	Name ↓	Archite
	<a href="#">master</a>	Linux (e
Data obtained		

**Build Queue**

No builds in the queue.

**Build Executor Status**

1 Idle  
2 Idle

○ ตั้งชื่อเป็น mgr1

Node name

**Permanent Agent**

Adds a plain, permanent agent to Jenkins. This is called "permanent" because Jenkins provisioning. Select this type if no other agent types apply — for example such as whe Jenkins, etc.

**OK**

- ตั้งค่าตาม form ดังล่าง

Name	<input type="text" value="mgr1"/>		
Description	<input type="text"/>		
# of executors	<input type="text" value="1"/>		
Remote root directory	<input type="text" value="/home/vagrant"/>		
Labels	<input type="text" value="mgr1"/>		
Usage	<input type="text" value="Use this node as much as possible"/>		
Launch method	<input type="text" value="Launch slave agents via SSH"/>		
Host	<input type="text" value="192.168.1.12"/>		
Credentials	<input type="text" value="root"/>	<input type="button" value="Add"/>	
Host Key Verification Strategy	<input type="text" value="Non verifying Verification Strategy"/>		
Availability	<input type="text" value="Keep this agent online as much as possible"/>		

**Node Properties**

☐ Environment variables

☐ Tool Locations

- กด save จะได้ node ใหม่ขึ้นมา

**jenkins#2** สร้าง job ชื่อ my\_web\_ex โดยมีความสามารถดังนี้

1. build อัตโนมัติเมื่อ code ใน github มีการเปลี่ยนแปลงโดยทำการ check code ทุก ๆ 5 นาที
2. เมื่อ build เสร็จแล้วให้ deploy ไปที่ docker swarm (mgr1)

### วิธีการ

- สร้าง File Jenkinsfile เพื่อเก็บ code pipeline
  - สร้าง file Jenkinsfile แล้วใส่คำสั่งดังนี้
  - push code ขึ้น github
- สร้าง Job ที่ jenkins
  - กดปุ่ม New Item
  - ระบุชื่อ
  - เลือก Pipeline

- เลือก Poll SCM ใส่ค่าเป็น H/5 \* \* \* \*
- Pipeline
  - Defination เลือกเป็น Pipeline script from SCM
  - SCM เลือก Git
  - Repository Url : ใส่ <<your\_github\_url>>/devops\_docker
- ตรวจสอบการทำงานของ Job ว่ามีการทำงานหรือไม่
  - เมื่อเวลาผ่านไป 5 นาทีจะ job log เกิดขึ้น
- แก้ไข file index.html เพิ่มคำสั่งดังนี้
  - <h3>Edit for test jenkins</h3>
  - push ขึ้น github
- ตรวจสอบการทำงานของ Job ว่ามีการทำงานหรือไม่
  - เมื่อเวลาผ่านไป 5 นาทีจะมี job log เกิดขึ้น
- ตรวจสอบดู Changes ว่าแสดงผลอย่างไร

## **Exercise**

**jenkins-exercise#1** สร้าง job ชื่อ demo, demoApi โดยมีความสามารถดังนี้

1. build อัตโนมัติเมื่อ code ใน github มีการเปลี่ยนแปลงโดยทำการ check code ทุก ๆ 5 นาที
2. เมื่อ build เสร็จแล้วให้ deploy ไปที่ docker swarm (mgr1)

<<end jenkins part>>