**DevOps**

* **Explain your CI/CD stages/pipeline. Showing the execution flow among your tools. Briefly describe your work for each stage.**

**Frontend CI/CD pipeline by Jenkins**

<https://github.com/INT222-13-49-129/INT222_Integrated_Project_Front-End/blob/master/Jenkinsfile>

pipeline {

agent any

stages {

stage('stop and remove container, image') { **## Stage นี้ใช้เช็คว่ามี Image ที่ต้องการ**

steps { **อยู่แล้วหรือไม่ ถ้ามีจะข้าม Stage นี้ไป**

script { **แต่ถ้าไม่มีจะลบ Container และ Image ชื่อซ้ำออก**

def imageExists = sh(script: 'docker images -q frontend', returnStdout: true) == ""

println imageExists

if( !imageExists ){

sh 'docker stop frontend'

sh 'docker rm frontend'

sh 'docker image rm frontend'

}else {

echo 'Skip this stage '

}

}

}

}

stage('remove whole data') { **## ลบ Data บน Jenkins Workspace ทั้งหมด**

steps {

sh 'rm -rf \*'

}

}

stage('git clone') { **## Clone secure บน Git เพื่อนำมา Deploy**

steps {

git branch: 'master',

credentialsId: 'INT222CFAN',

url: 'https://github.com/INT222-13-49-129/INT222\_Integrated\_Project\_Front-End.git'

}

}

stage('(deploy) start contianer') { **## Deploy ด้วย docker-compose up -d**

steps {

sh 'docker-compose up -d'

}

}

}

}

**Backend CI/CD pipeline by Jenkins**

<https://github.com/INT222-13-49-129/INT222_Integrated_Project_Back-End/blob/master/Jenkinsfile>

pipeline {

agent any

tools { **## ใช้ Nodejs ทำ Test ด้วย newman**

nodejs "nodejs"

}

stages {

stage('stop and remove container, image') { **## Stage นี้ใช้เช็คว่ามี Image ที่ต้องการ**

steps { **อยู่แล้วหรือไม่ ถ้ามีจะข้าม Stage นี้ไป**

script { **แต่ถ้าไม่มีจะลบ Container และ Image ชื่อซ้ำออก**

def imageExists = sh(script: 'docker images -q backend', returnStdout: true) == ""

println imageExists

if( !imageExists ){

sh 'docker stop backend'

sh 'docker rm backend'

sh 'docker image rm backend'

}else {

echo 'Skip this stage '

}

}

}

}

stage('remove whole data') {  **## ลบ Data บน Jenkins Workspace ทั้งหมด**

steps {

sh 'rm -rf \*'

}

}

stage('git clone') { **## Clone secure บน Git เพื่อนำมา Deploy**

steps {

git branch: 'master',

credentialsId: 'INT222CFAN',

url: 'https://github.com/INT222-13-49-129/INT222\_Integrated\_Project\_Back-End.git'

}

}

stage('(deploy) start contianer') { **## Deploy ด้วย docker-compose up -d**

steps {

sh 'docker-compose up -d'

}

}

stage('test') { **## Install Test case ทุกตัว จาก postman collection**

steps { **ที่ทำบน postman save ลงเครื่อง แล้ว up ขึ้น Git**

sh 'node --version '

sh 'npm --version '

sh 'npm install -g newman'

sh 'newman run postman/postmantest/INT222CFAN.postman\_collection.json'

}

}

}

}

**ในส่วนของ Postman test**

1. **Login Token test**

Graphical user interface, text, application

Description automatically generated

1. **Get User test**

Graphical user interface

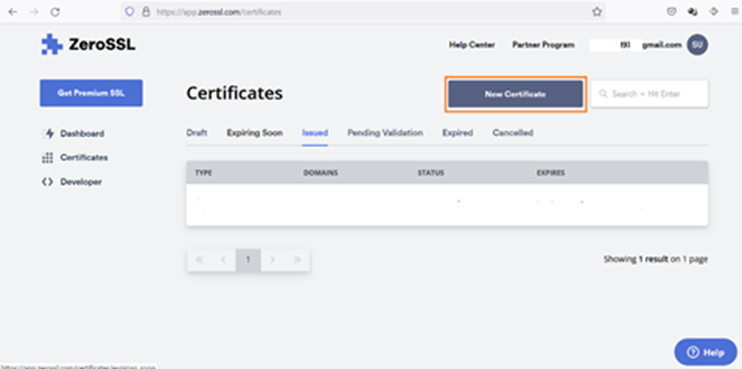
Description automatically generated

1. **Get Foodmenu test**

Graphical user interface, application

Description automatically generated

* **Describe your work for HTTPS.**

1. Go to URL: <https://app.zerossl.com/signup>
2. 
3. ใส่ Domain name ลงไป Graphical user interface, application, Teams

   Description automatically generated
4. Download ตัว Auth file เพื่อตรวจเช็คว่า Dns ใช้งานได้Graphical user interface, text, application

   Description automatically generated
5. นำ Auth file ไปเก็บบน VM
6. เช็คว่าเจอ Auth file หรือไม่Graphical user interface, text, application

   Description automatically generated
7. Graphical user interface, text, application, email

   Description automatically generated
8. จดใบ Certificate ได้เลยGraphical user interface, application

   Description automatically generated
9. โหลดใบ Certificate เพื่อนำมายืนยันเว็ปด้วยวิธีการ นำไปวางไว้บน VMGraphical user interface, text, application, email

   Description automatically generated

Graphical user interface, application, Word

Description automatically generated

Graphical user interface

Description automatically generated

Graphical user interface, text

Description automatically generated

1. หลังจากนั้น config ตัว default.conf เพื่อใส่ตัว proxy ลงไปและ redirect Public IP มาที่ Https

server { #Monitor port 443

listen 443 ssl;

server\_name cfan.ddnsking.com; #domain name

# Add ssl

ssl on; #If you force HTTP access, this line should be opened

ssl\_certificate /ssl /cfan.crt; #The crt file storage path of nginx for ss>

ssl\_certificate\_key /ssl /cfan.key; #Storage path of nginx key file of >

ssl\_session\_cache shared:SSL:1m;

ssl\_session\_timeout 5m;

# Specify the password as a format supported by openssl

ssl\_protocols SSLv2 SSLv3 TLSv1.2;

ssl\_ciphers HIGH:!aNULL:!MD5; # Password encryption method

ssl\_prefer\_server\_ciphers on; # Server passwords that rely on SSLv3 and>

# Define the index directory and name of the first page

location / {

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_pass http://172.20.0.2;

}

location /backend/ {

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_pass http://172.20.0.3:3000/;

}

#Redirect error page to / 50x.html

error\_page 500 502 503 504 /50x.html;

location = /50x.html {

root /usr/share/nginx/html;

}

}

server{ #Monitor port 80

listen 80;

server\_name cfan.ddnsking.com;

rewrite ^(.\*) https://$host$1 permanent; # All connections that ar>

}

server{

listen 443;

server\_name 20.212.82.102;

return 302 $scheme://cfan.ddnsking.com$request\_uri;

}

1. สร้าง compose เพื่อสร้าง environment ให้ Https

version: '3'

services:

HttpsProxy:

container\_name: HttpsProxy

image: nginx

volumes:

- /home/dohttps/nginx/config/nginx.conf:/etc/nginx/nginx.conf/:rw

- /home/dohttps/nginx/config/conf.d/default.conf:/etc/nginx/conf.d/default.conf:rw

- /home/dohttps/nginx/logs:/var/log/nginx/:rw

- /home/dohttps/nginx/ssl:/ssl/:rw

ports:

- "80:80"

- "443:443"

networks:

allnetwork:

ipv4\_address: 172.20.0.5

networks:

default:

external:

name: allnetwork

allnetwork:

external: true