

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') {  
            steps {  
                script {  
                    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 {  
                        ## Stage นี้ใช้เช็คว่ามี Image ที่ต้องการ  
                        อยู่แล้วหรือไม่ ถ้ามีจะข้าม Stage นี้ไป  
                        แต่ถ้าไม่มีจะลบ Container และ Image ซื่อซ้ำออก  
                    }  
                }  
            }  
        }  
    }  
}
```

```

        echo 'Skip this stage '
    }

}

}

}

```

```

stage('remove whole data') {
    steps {
        sh 'rm -rf *'
    }
}

```

ลบ Data บน Jenkins Workspace ทั้งหมด

```

stage('git clone') {
    steps {
        git branch: 'master',
        credentialsId: 'INT222CFAN',
        url: 'https://github.com/INT222-13-49-129/INT222_Integrated_Project_Front-
End.git'
    }
}

```

Clone secure บน Git เพื่อนำมา Deploy

```

stage('(deploy) start container') {
    steps {

```

Deploy ด้วย docker-compose up -d

```

        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
            ## อยู่แล้วหรือไม่ ถ้ามีจะข้าม Stage นี้ไป
            steps {
                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 container') {    ## Deploy ด้วย docker-compose up -d

  steps {

    sh 'docker-compose up -d'

  }

}
```

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

  steps {    ## ทำบน 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

INT222CFAN

Resource

POST Login Token

User Test

POST

{{MyURL}}/backend/api/general/login

Params

Authorization

Headers (7)

Body

Pre-request Script

Tests

```
1 pm.test("Status code is 200", function(){
2   pm.response.to.have.status(200);
3 });
4
5
6 pm.test("token???", function(){
7   pm.response.to.have.jsonBody("token");
8 });
9
10
11 pm.environment.set("token", pm.response.json().token);
12 console.log("token: "+pm.environment.get("token"))
```

2. Get User test

INT222CFAN

Resource

POST Login Token

User Test

GET GET User

GET GET Foodmenu

GET

{{MyURL}}/backend/api/user

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

```
1 pm.test("Status code is 200", function(){
2   pm.response.to.have.status(200);
3 });
4
```

3. Get Foodmenu test

INT222CFAN

Resource

POST Login Token

User Test

GET GET User

GET GET Foodmenu

GET

{{MyURL}}/backend/api/user/foodmenu

Params

Authorization

Headers (6)

Body

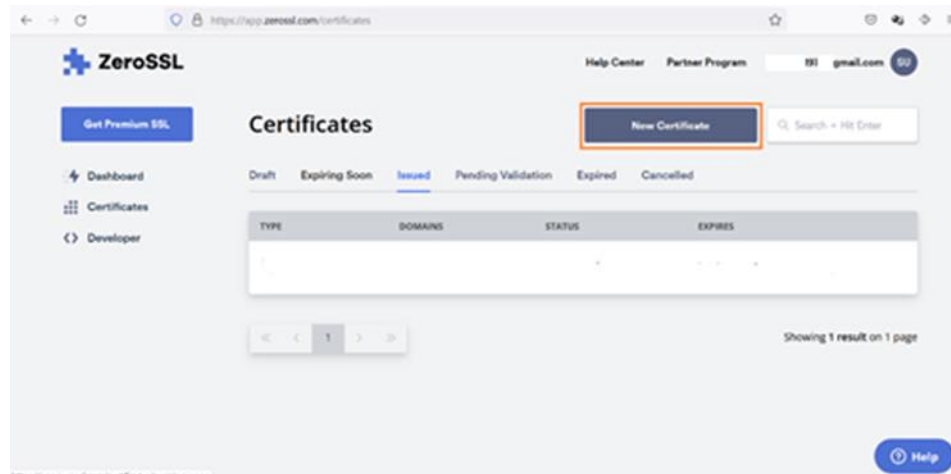
Pre-request Script

Tests

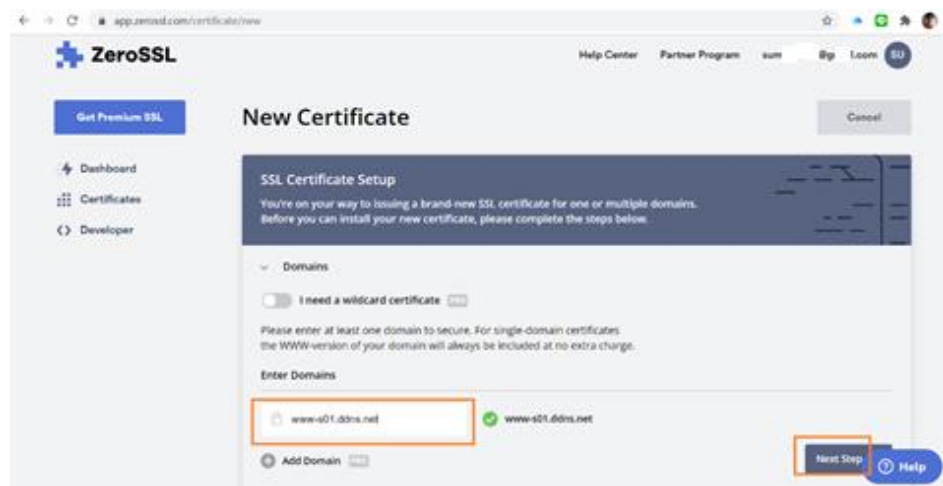
```
1 pm.test("Status code is 200", function(){
2   pm.response.to.have.status(200);
3 });
```

- Describe your work for HTTPS.

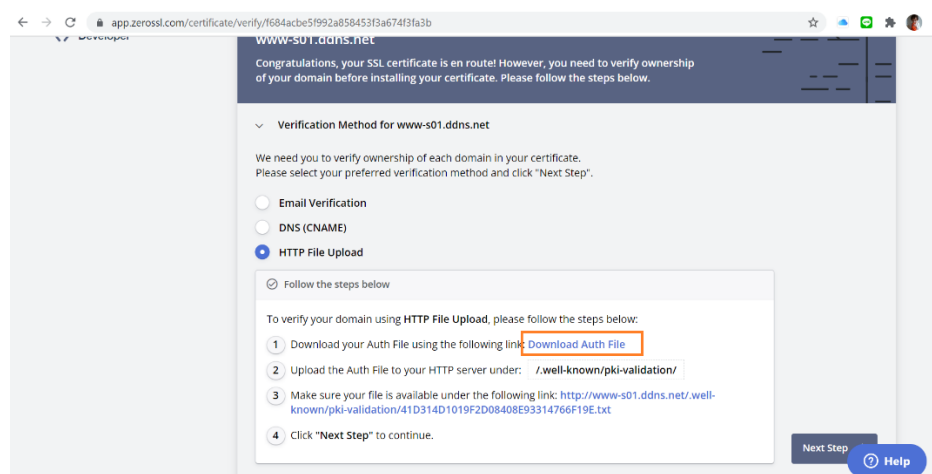
1. Go to URL: <https://app.zerossl.com/signup>



2.
3. ใส่ Domain name ลงไป

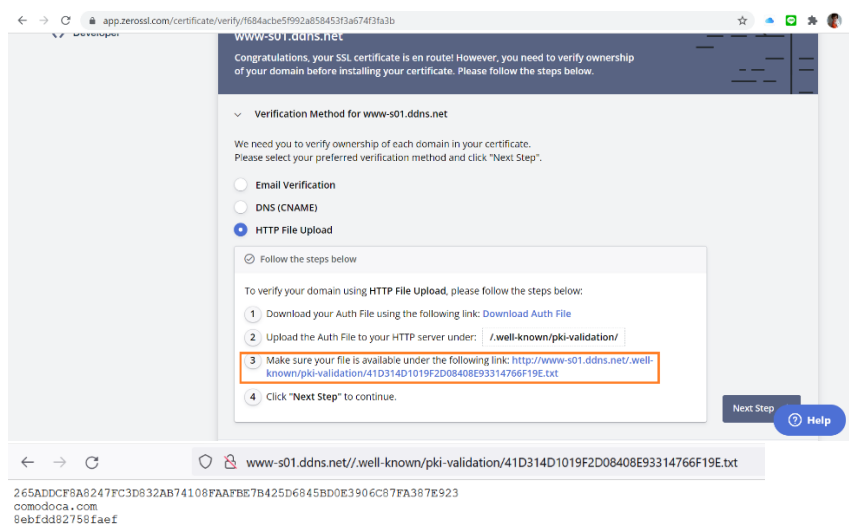


4. Download ตัว Auth file เพื่อตรวจสอบเช็ค ว่า Dns ใช้งานได้

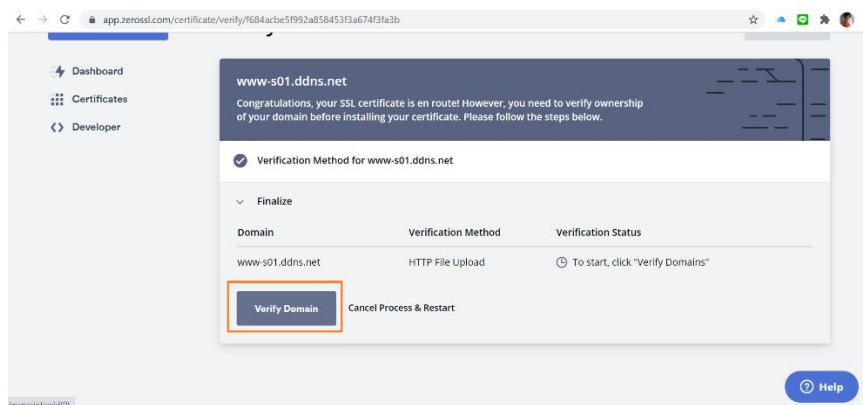


5. นำ Auth file ไปเก็บบน VM

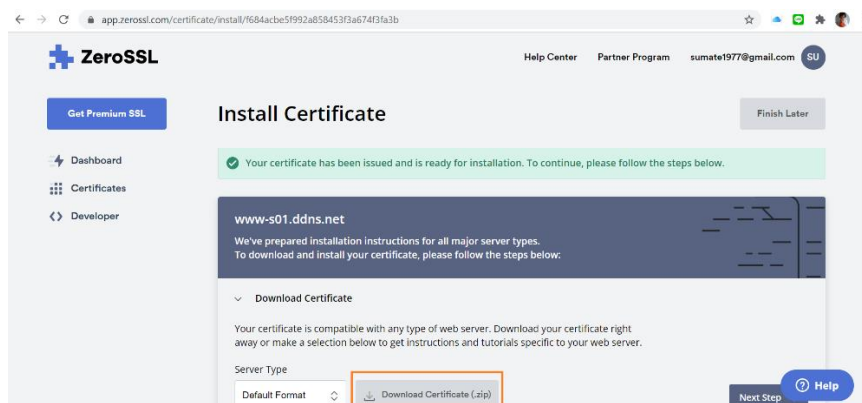
6. เช็คว่าเจอ Auth file หรือไม่

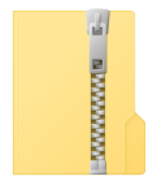


7. จดใบ Certificate ได้เลย



9. โหลดใบ Certificate เพื่อนำมายืนยันเว็บไซต์ด้วยวิธีการ นำไปวางไว้บน VM





www-s01.ddns.net

```

root@kali:~/Documents# ifconfig
eth0: flags=4096<UP,BROADCAST,MULTICAST> mtu 1500
    inet 10.10.10.10 netmask 255.255.255.0 broadcast 10.10.10.255
    ether 08:00:27:00:00:00 txqueuelen 1000 (0 bytes)
    RX packets 0 bytes 0 (0.0 B)
    TX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 (0.0 B)
    collisions 0
lo: flags=73<LOOPBACK,UP,APIGT> mtu 65536
    inet 127.0.0.1 netmask 255.255.255.255
    ether 00:00:00:00:00:00 txqueuelen 1000 (0 bytes)
    RX packets 0 bytes 0 (0.0 B)
    TX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 (0.0 B)
    collisions 0

```

```

C:\Program Files\Internet Explorer>certutil -getcert
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 0 (0x0)
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=TH, O=School of Information Technology RMUTT, CN=BITCertAuth
        Validity:
            Not Before: Feb 26 03:27:08 2019 GMT
            Not After : Feb 23 03:27:08 2029 GMT
        Subject: C=TH, O=School of Information Technology RMUTT, CN=BITCertAuth
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (4096 bit)
            Modulus (4096 bit):
                55:db:4b:43:5c:4e:55:d8:59:3b:7b:1a:de:cb:fb:f2:
                09:5d:84:f4:a5:2d:8f:0c:85:1d:5d:07:a4:24:a:7b:f8:
                f4:b5:c0:44:e1:0c:38:76:b6:49:f2:03:f1:de:1f:
                f8:b2:f6:10:a4:7d:0d:0c:0b:5c:27:a8:3c:b6:fb:0e:4d:
                44:07:70:5:40:9a:7a:ba:72:78:5b:7a:9c:c0:41:
                c1:10:3b:3c:25:62:28:1e:0c:f1:3c:b2:b6:7b:5a:ad:

```

10. 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;
}
```

11. สร้าง compose เพื่อสร้าง environment ให้ Https

version: '3'

services:

HttpsProxy:

container_name: HttpsProxy

image: nginx

volumes:

- /home/dohhttps/nginx/config/nginx.conf:/etc/nginx/nginx.conf:rw
- /home/dohhttps/nginx/config/conf.d/default.conf:/etc/nginx/conf.d/default.conf:rw
- /home/dohhttps/nginx/logs:/var/log/nginx:rw
- /home/dohhttps/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