

ภาคผนวก

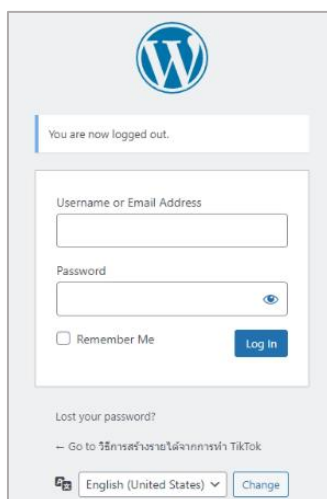
ภาคผนวก ก.
คู่มือการใช้งาน

ส่วนของ WordPress

ระบบหน้าบ้านของ WordPress หน้าเว็บไซต์ เป็นหน้าแรกไว้คอยต้อนรับผู้ที่เข้าชมเว็บไซต์

ระบบหลังบ้านของ WordPress คือ ระบบที่เอาไว้จัดการข้อมูลต่าง ๆ ภายในเว็บไซต์ตั้งแต่การตั้งค่าพื้นฐาน ลงข้อมูลในหน้าต่าง ๆ สร้างเมนู จัดการรูปภาพ ลงสินค้า จัดหน้า Layout จัดการ Theme และ Plugin โดยผู้ที่จะใช้ในระบบหลังบ้านได้จำเป็นต้องมีรหัสผ่านและ ชื่อผู้ใช้ ก่อนซึ่งสามารถใช้งานได้ดังนี้

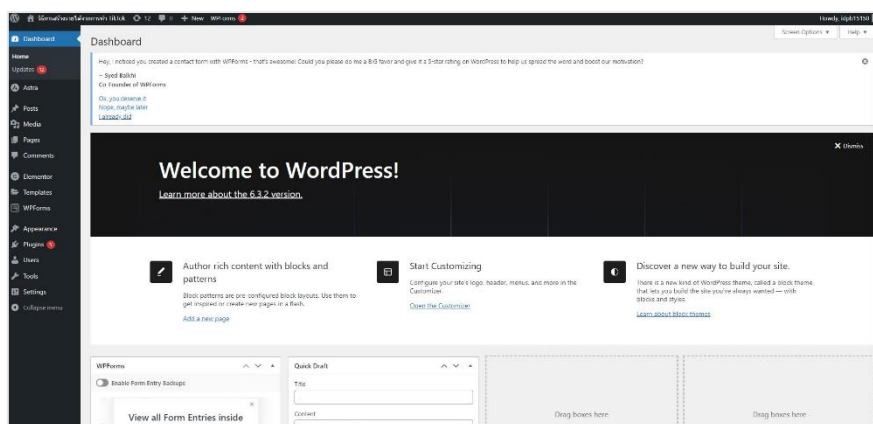
ระบบ login



The image shows the WordPress login interface. At the top is the WordPress logo. Below it, a message says "You are now logged out." The main section contains a login form with two input fields: "Username or Email Address" and "Password". There is a "Remember Me" checkbox and a "Log In" button. Below the form, there is a link for "Lost your password?" and a text link "Go to วิธีการสร้างรายได้จากการทำงาน TikTok". At the bottom, there is a language selector set to "English (United States)" and a "Change" button.

ภาพที่ ก.1 ระบบ login

Dashboard

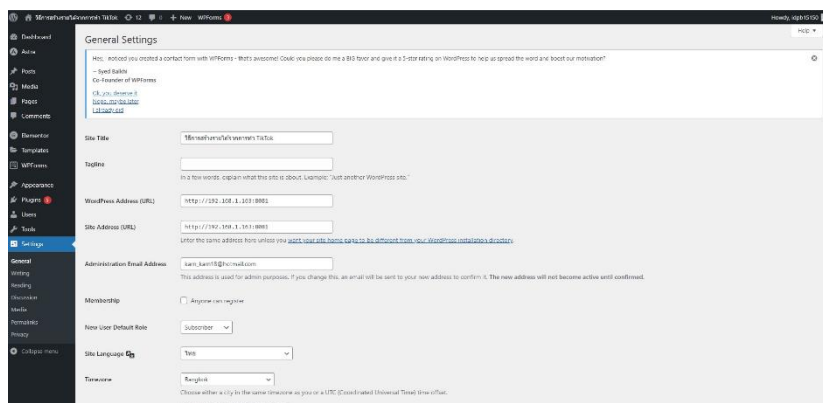


ภาพที่ ก.2 Dashboard

ทำได้โดย www.ตรงนี้ใส่ชื่อเว็บ .com/wp-admin จากนั้นก็ทำการใส่ข้อมูล ชื่อผู้ใช้และรหัสผ่าน ลงไปเพื่อเข้าสู่ระบบ เมื่อเข้าระบบหลังบ้านของ WordPress แล้วจะปรากฏหน้าจอสำหรับการปรับแต่งที่มีเมนูหลายอย่างให้เลือก

3. ตั้งค่า Site Identity & Tagline

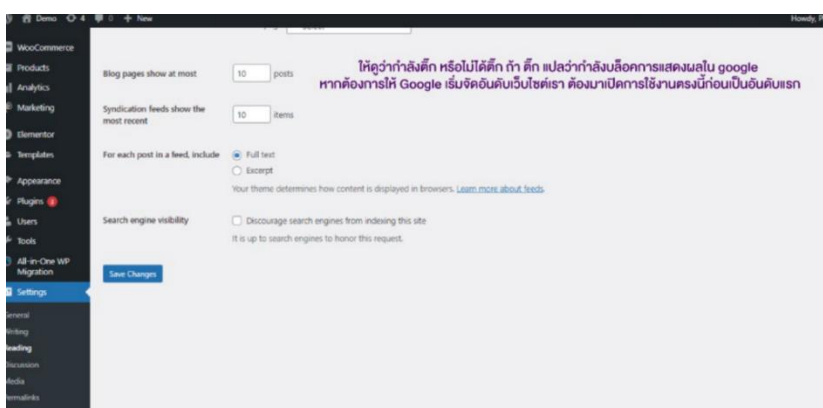
- Site Title : ใส่ชื่อเว็บ หรือชื่อ Brand ลงไป
- Tagline : เขียนคำอธิบายสั้นๆ เกี่ยวกับธุรกิจหรือร้านค้าของเรา
- Email Address : ใส่อีเมลที่ใช้งานจริง
- Site Language : เลือก ไทย
- Time zone : เลือกโซนเวลา เมืองไทยให้เลือกเป็น UTC+7
- Date Format : เลือกแบบ วัน/เดือน/ปี
- Week Starts On : วันเริ่มต้นสัปดาห์ ให้เลือกเป็น วันอาทิตย์
- Logo : จะใช้โลโก้ตัวนี้ไปแสดงเป็นตัวหลักไปเลยก็ได้
- Site Title : เป็นคำหลักของเว็บไซต์
- Tagline : เป็นคำโปรยหรือคำอธิบาย
- Favicon : เป็น icon ไอคอนเล็กๆที่ปรากฏอยู่บนแท็บของเบราว์เซอร์



ภาพที่ ก.5 General Settings

4. ตั้งค่าการมองเห็นของ Search Engine สำหรับ SEO เพื่อใช้ในการทำการตลาด

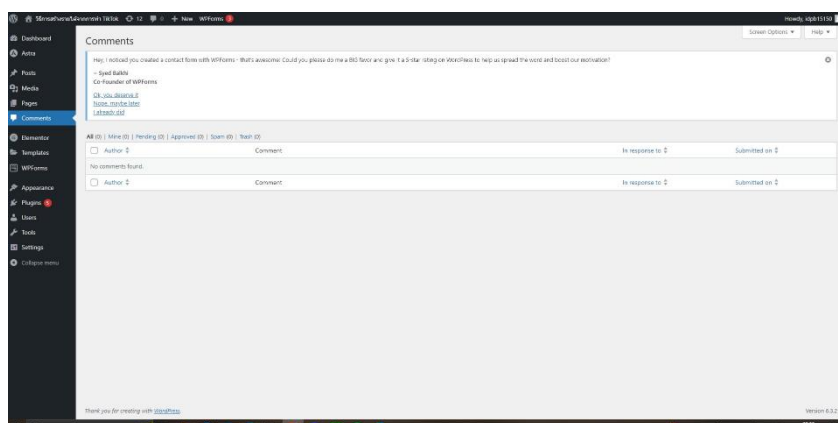
- ให้อ่านว่ากำลังติด หรือไม่ได้ติด ถ้าติด แปลว่ากำลังบล็อกการแสดงผลใน google หากต้องการให้ Google เริ่มจัดอันดับเว็บไซต์เรา ต้องมาเปิดการใช้งานตรงนี้ก่อนเป็นอันดับแรก



ภาพที่ ก.6 Settings

5. ตั้งค่าการ Comments

- โดยการตั้งค่าแบบนี้จะเป็นการปิดไม่ให้เห็นความเห็นบน Post ใหม่ๆ เท่านั้น ส่วน Post ก็เก่ายังแสดงความคิดเห็นได้อยู่

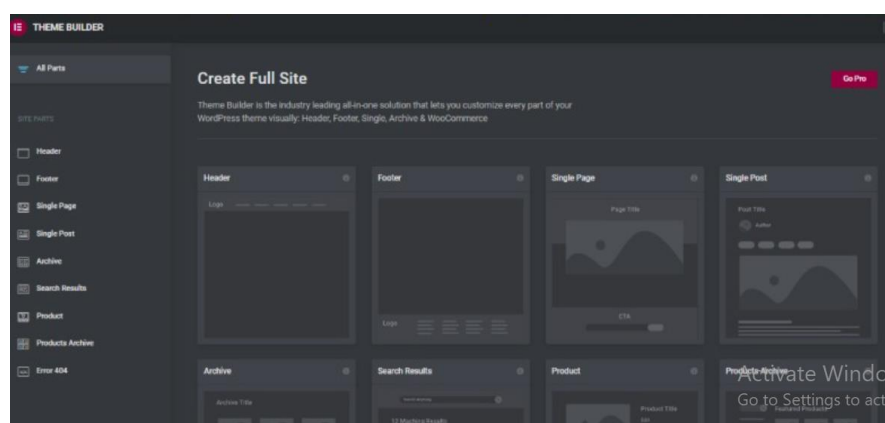


ภาพที่ ก.7 Comments

6. ตั้งค่า Theme

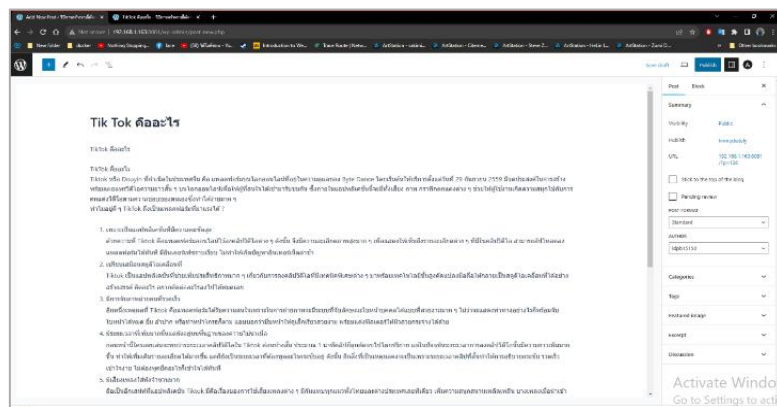
Themes ซึ่งทำหน้าที่ในส่วนของหน้าตาเว็บไซต์สามารถปรับแต่งธีมได้ดังนี้

- ตั้งค่าสีหลัก สีลิงค์ สีข้อความต่างๆ และ ฟอนต์
- ตั้งค่า Header, Footer
- ตั้งค่า Sidebar (Left, Right) *ส่วนด้านข้างซ้าย-ขวา
- ตั้งค่าหน้า Post, Page
- ตั้งค่าหน้า Shop
- การตั้งค่าขั้นสูง



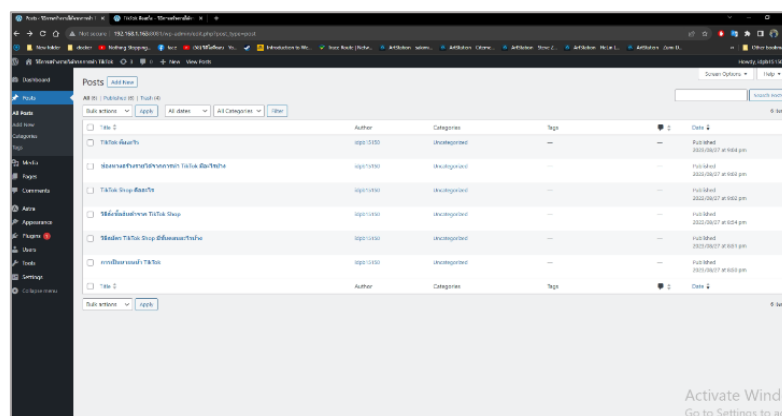
ภาพที่ ก.8 Theme

เมื่อตั้งค่าเสร็จเรียบร้อยแล้ว ไปที่ Post ทำการ Add New ใส่หัวข้อและเนื้อหา Publish เพื่อบันทึกเนื้อหา



ภาพที่ ก.9 Add New Post

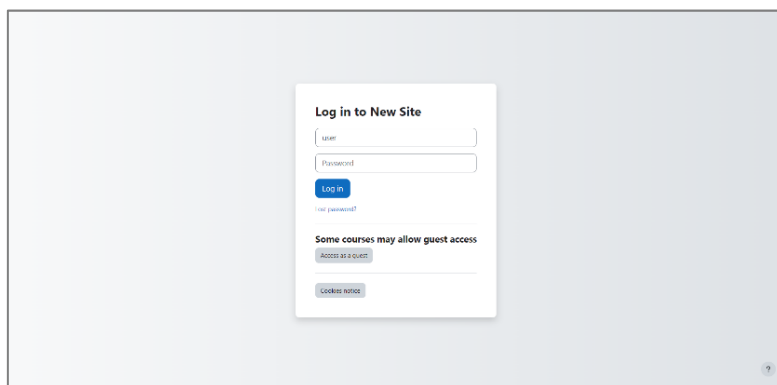
ในส่วนนี้จะเป็นหัวข้อทั้งหมดที่ได้ทำการสร้างไว้



ภาพที่ ก.10 หน้า Post

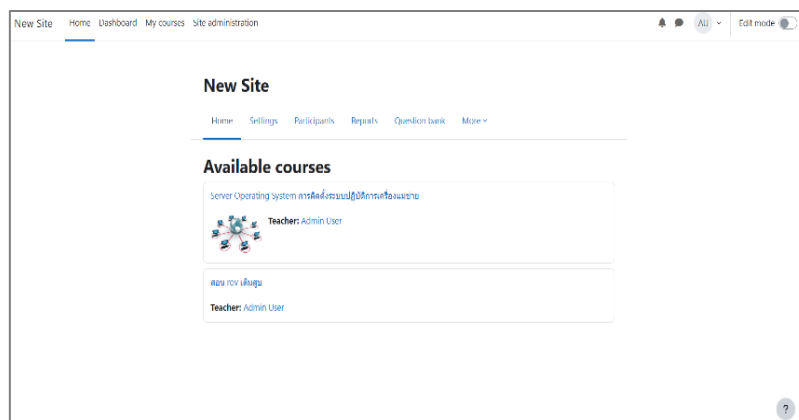
ส่วน ของ Moodle

ในส่วนของหน้า login จะใช้ User ที่แอดมินสร้างให้เข้าเท่านั้น



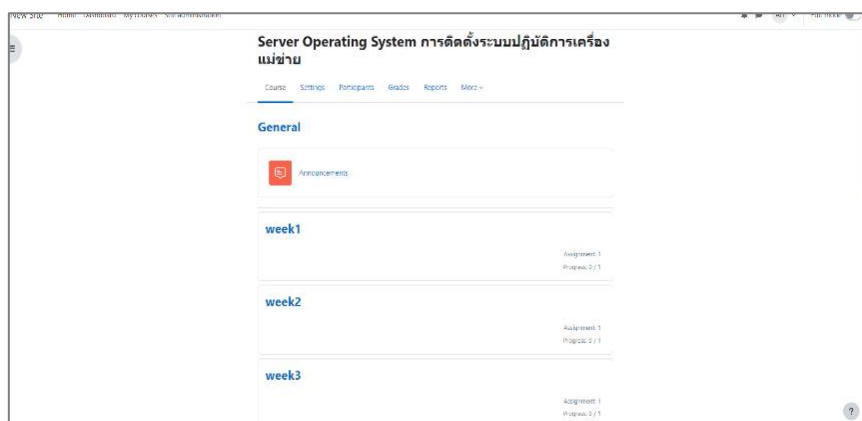
ภาพที่ ก.11 หน้า login

ส่วนของ หน้าแรกที่จะมีรายวิชาที่แอดมินเลือกไว้ให้ว่า User คนนั้นจะเห็นรายวิชาที่แอดมินกำหนดไว้



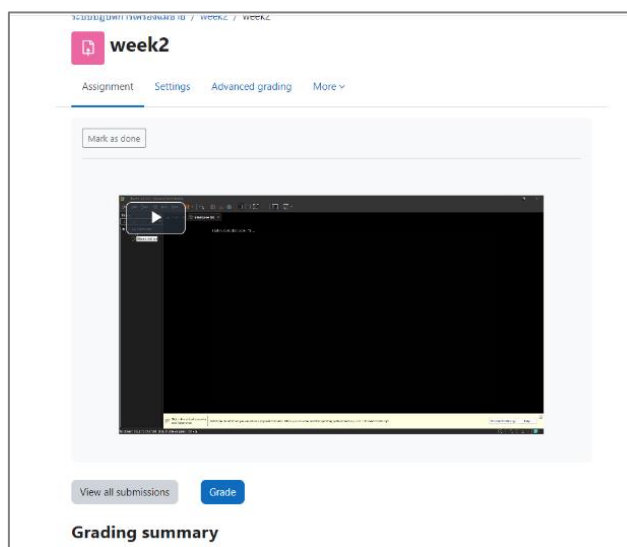
ภาพที่ ก.12 ส่วนของรายวิชา

ในส่วนของรายวิชาที่แอดมินกำหนดไว้ให้จะมีงานรายสัปดาห์ไว้ให้ผู้เรียนเข้าไปศึกษาทำตาม โดยจะมีทั้งหมด 15 สัปดาห์



ภาพที่ ก.13 ส่วนของงานรายสัปดาห์

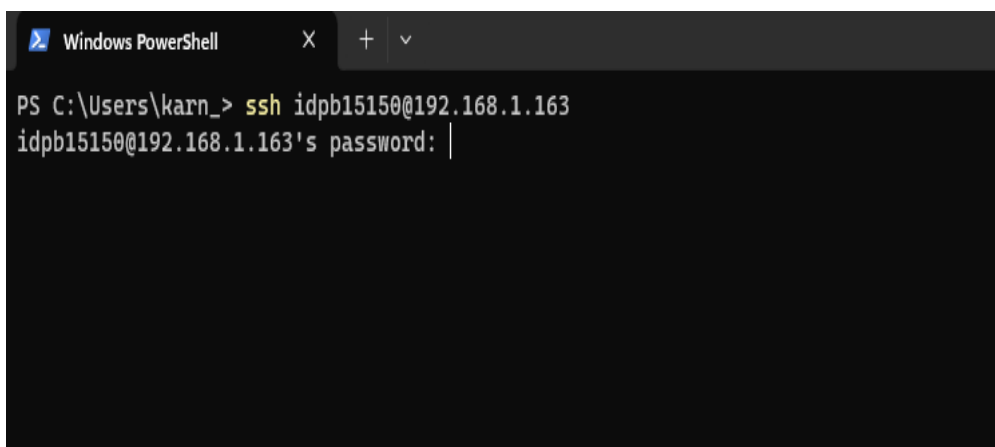
ส่วนของงานรายสัปดาห์จะมีคลิปการเรียนการสอนให้ทำตาม และมีงานรายสัปดาห์ให้ส่งเพื่อทดสอบผู้เรียนว่าเข้าใจหรือไม่



ภาพที่ ก.14 ส่วนของคลิปวิดีโอการเรียนการสอนใน Moodle

ในส่วนของ ระบบ

ในส่วนของการใช้งาน Login เข้าสู่ Server



ภาพที่ ก.15 ส่วนของคลิปวิดีโอการเรียนการสอน

หลังจาก login เข้ามา

```
idpb15150@idp15150: ~  
PS C:\Users\karn_> ssh idpb15150@192.168.1.163  
idpb15150@192.168.1.163's password:  
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.0-79-generic x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:        https://ubuntu.com/advantage  
  
System information as of Sat Nov 11 08:48:08 AM +07 2023  
  
System load:                0.0810546875  
Usage of /:                  20.9% of 86.44GB  
Memory usage:                31%  
Swap usage:                  0%  
Processes:                   290  
Users logged in:             0  
IPv4 address for br-254d5f20f1e7: 172.21.0.1  
IPv4 address for br-2debca4fe7ff: 172.18.0.1  
IPv4 address for br-ac2a727f66de: 172.27.0.1  
IPv4 address for br-b90a92f4c82c: 172.20.0.1  
IPv4 address for br-ea4dcebd4d10: 172.19.0.1  
IPv4 address for docker0:      172.17.0.1  
IPv4 address for ens33:        192.168.1.163  
  
* Strictly confined Kubernetes makes edge and IoT secure. Learn how  
  just raised the bar for easy, resilient and secure K8s cluster dep  
  
https://ubuntu.com/engage/secure-kubernetes-at-the-edge  
  
Expanded Security Maintenance for Applications is not enabled.  
  
68 updates can be applied immediately.  
To see these additional updates run: apt list --upgradable  
  
Enable ESM Apps to receive additional future security updates.  
See https://ubuntu.com/esm or run: sudo pro status  
  
*** System restart required ***  
Last login: Sat Nov 11 08:34:37 2023 from 192.168.1.128  
idpb15150@idp15150:~$ |
```

ภาพที่ ก.16 หลังจาก login เข้ามา

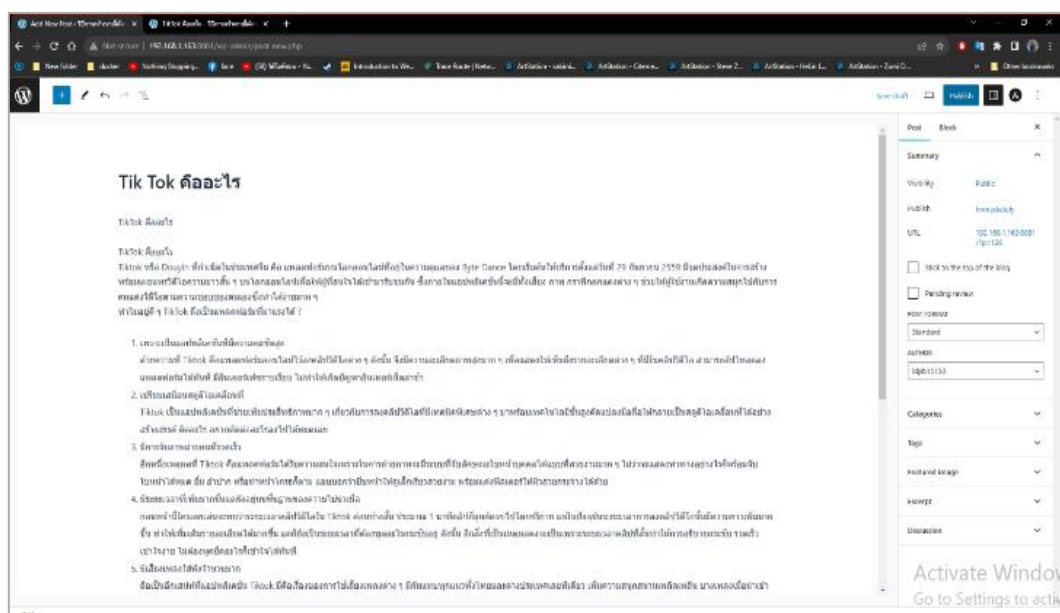
จากนั้นทำการเปิด Container ใน Docker ที่ได้มีการสร้างไว้แล้ว

```

idp015159@idp15159:~$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED     STATUS    PORTS                               NAMES
c5249a7d5b5d   dpag/pgadmin4  "/entrypoint.sh"        3 months ago Up 18 hours 443/tcp, 0.0.0.0:8888->88/tcp, :::8888->88/tcp   pantor-pg4-1
741fb2d77a81   postgres      "docker-entrypoint.s..." 3 months ago Up 18 hours 5432/tcp, 0.0.0.0:5432->5433/tcp, :::5432->5433/tcp   pantor-db-1
1cbb9df55924   nginx         "docker-entrypoint.s..." 3 months ago Up 18 hours 0.0.0.0:8099->80/tcp, :::8099->80/tcp   pantor-ngx-1
626c91718daa   wordpress     "docker-entrypoint.s..." 3 months ago Up 5 seconds 0.0.0.0:8081->80/tcp, :::8081->80/tcp   wpcontainer
87e97c86b6b5   mariadb      "docker-entrypoint.s..." 3 months ago Up 12 seconds 3306/tcp   wordpressb
14a9ea887c32   bitnami/noodle:latest "opt/bitnami/script..." 3 months ago Up 17 minutes 0.0.0.0:80->8088/tcp, :::80->8088/tcp, 0.0.0.0:443->8043/tcp, :::443->8043/tcp   noodle
805f289f159e   mariadb      "docker-entrypoint.s..." 3 months ago Up 17 minutes 3306/tcp   noodledb
idp015159@idp15159:~$
  
```

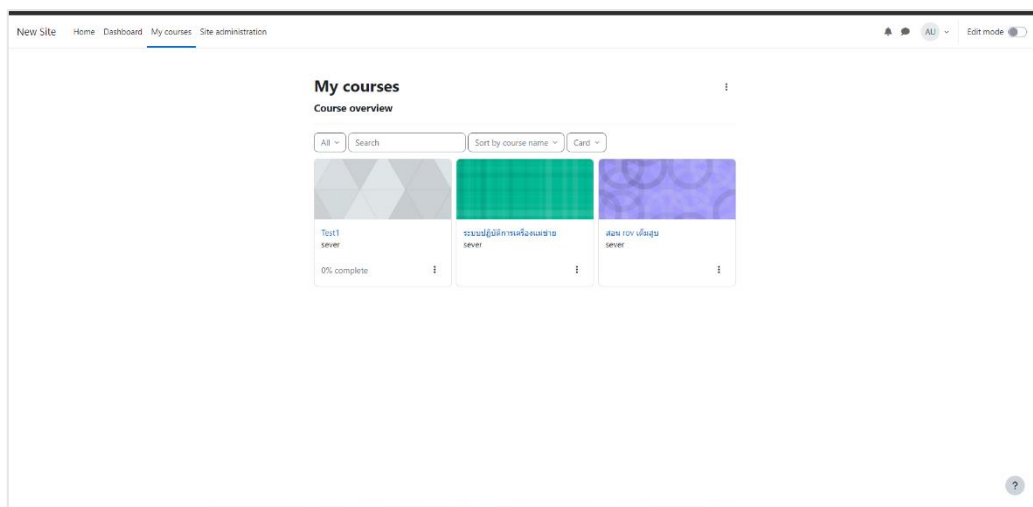
ภาพที่ ก.17 Container ใน Docker

หน้าต่างหลังจากเปิด Container
เนื้อหา WordPress



ภาพที่ ก.18 WordPress ใน Container

สื่อการเรียนการสอน Moodle



ภาพที่ ก.19 Moodle ใน Container

ภาคผนวก ข.
โค้ดโปรแกรม

คำสั่งในการสร้าง ระบบ

การตั้ง ip network

network: # เริ่มต้นของโค้ด

ethernets: # กำหนดค่าของ ethernets

ens33: # ชื่อของ interface เครือข่าย

addresses: # กำหนด IP address สำหรับ interface ens33

- 192.168.1.164/24 # IP address และ subnet mask

nameservers: # กำหนดข้อมูลของ nameservers

addresses: # กำหนด IP addresses ของ nameservers

- 192.168.1.1 # IP addresses ของ nameservers

search: [] # กำหนดค่าว่างในส่วนของการค้นหาชื่อโดเมน

routes: #กำหนดข้อมูลเส้นทาง

- to: default # กำหนดค่า default สำหรับเส้นทาง

via: 192.168.1.1 #กำหนด IP ของ gateway

version: 2 # กำหนดเวอร์ชันของ Network configuration

สร้าง postgres sql

postgres sql

install

การสร้าง PostgreSQL Container

docker run --name civicfe -e POSTGRES_PASSWORD=gtavc15150 -d -p 5432:5432

postgres

คำสั่งในการเริ่มต้น Docker container ของ PostgreSQL โดยกำหนดชื่อคอนเทนเนอร์ว่า"

civicfe" และกำหนดรหัสผ่านสำหรับผู้ใช้ postgres เป็น "gtavc15150"

login

sql

การเข้าสู่ PostgreSQL CLI

docker exec -it civic psql -U postgres

เป็นการเข้าสู่ PostgreSQL command-line interface (CLI) ในคอนเทนเนอร์ "civic" โดยใช้ผู้ใช้

postgres

```
bash
```

```
docker exec -it civicfe /bin/bash
```

```
dump database to ubuntu root
```

เป็นคำสั่งที่เรียกใช้งาน shell (/bin/bash) ภายใน Docker container ชื่อ "civicfe" โดยการเข้าถึงเป็นแบบ interactive mode (-it) เพื่อให้ผู้ใช้สามารถป้อนคำสั่งภายใน container ได้

```
# การสร้าง Dump Database
```

```
sudo docker exec -i civicfe /bin/bash -c "PGPASSWORD=gtavc15150 pg_dump --username postgres hit" > /root/home/dump.sql
```

คำสั่งนี้ใช้สร้างการสำรองข้อมูล (dump) ของฐานข้อมูลชื่อ "hit" โดยใช้คำสั่ง pg_dump ภายใน Docker container "civicfe" และบันทึกผลลัพธ์ลงในไฟล์ dump.sql ที่อยู่ในไดเรกทอรี /root/home ของโฮสต์

```
# สร้าง database
```

```
create database *name;
```

```
view database
```

```
\l
```

```
////////
```

```
# เข้า database
```

```
\c *namedatabase;
```

```
////////
```

```
# สร้าง table
```

```
create table menu(id int, descr text, price int);
```

```
////
```

```
ใน pathdatebase /d เพื่อดู table
```

```
////
```

```
# เพิ่มข้อมูล
```

```
insert into car values('1', 'six',45);
```

```
# การสร้าง pgadmin4
```

```
docker pg4
```

```
install container
```

```
docker run --name civicfk -p 80:80 -e 'PGADMIN_DEFAULT_EMAIL=karn@hotmail.com'
-e 'PGADMIN_DEFAULT_PASSWORD=gtavc15150' -d dpage/pgadmin4
```

```
# เพิ่ม disk
```

```
cfdisk
```

```
fdisk -l
```

```
df -h --total
```

```
lsblk
```

```
# เปลี่ยน user
```

```
to root sudo -i
```

```
to any user = su -*username
```

```
# เพิ่ม user
```

```
add user
```

```
sudo adduser ken
```

```
enter password
```

```
enter grop
```

```
หา user
```

```
find
```

```
find . -name 'namedata'
```

```
# ย้ายข้อไฟล์พร้อมข้อมูลด้านใน
```

```
move dir and file
```

```
dir
```

```
mv "dirname"/ to /home/idpb15150/to path
```

```
file
```

```
mv "filename" to /home/idpb15150/to path
```

```
# ลง moodle
```

```
moodle
```

```
docker run -d --name moodle -p 80:8080 -p 443:8443 -v moodle-
```

```
data:/bitnami/moodle --network moodle-network -e
```



```
MOODLE_DATABASE_HOST=moodledb -e MOODLE_DATABASE_USER=moodle -e
MOODLE_DATABASE_PASSWORD=gtavc15150 -e MOODLE_DATABASE_NAME=moodle
bitnami/moodle:latest
```

```
# ดู Vol
volum
mariadb-data
/var/lib/docker/volumes/mariadb-data/_data
wpcon
/var/www/html
wpdb
/var/lib/mysql
moodle
/var/lib/docker/volumes/moodle-data/_data
moodledb
/var/lib/mysql
```

```
ไฟล์ ip.py
from flask import Flask, render_template # เรียกใช้งาน Flask และ render_template
จาก Flask
import paramiko # เรียกใช้งาน paramiko เพื่อทำการเชื่อมต่อ SSH
app = Flask(__name__) # สร้างอ็อบเจกต์ Flask
class SSHConnection:
    def __init__(self, host, username, password):
        self.host = host # กำหนด host
        self.username = username # กำหนด username
        self.password = password # กำหนด password
        self.client = paramiko.SSHClient() # สร้างอ็อบเจกต์ SSHClient จาก paramiko
        self.client.set_missing_host_key_policy(paramiko.AutoAddPolicy()) # กำหนด
นโยบายการเพิ่มโฮสต์อัตโนมัติ
```

```

def connect(self):
    try:
        self.client.connect(self.host, username=self.username,
password=self.password) # เชื่อมต่อ SSH
โดยใช้ข้อมูลที่กำหนด
        print(f"Connected to {self.host}") # พิมพ์ข้อความแสดงว่าเชื่อมต่อสำเร็จ
    except Exception as e:
        print(f"Connection failed: {e}") # พิมพ์ข้อความแสดงว่าเชื่อมต่อล้มเหลว

def execute_command(self, command):
    try:
        stdin, stdout, stderr = self.client.exec_command(command) # ส่งคำสั่ง SSH
ไปยังอุปกรณ์
        return stdout.readlines() # อ่านผลลัพธ์ที่ได้กลับมา
    except Exception as e:
        print(f"Command execution failed: {e}") # พิมพ์ข้อความแสดงว่าการทำงานคำสั่งล้มเหลว
    return []

def close(self):
    self.client.close() # ปิดการเชื่อมต่อ SSH
    print("Connection closed") # พิมพ์ข้อความแสดงว่าการเชื่อมต่อถูกปิด
    # สร้างอ็อบเจกต์ SSHConnection
    ssh_connection = SSHConnection('192.168.88.1', 'admin', 'gtavc15150')
    # เชื่อมต่อ SSH
    ssh_connection.connect()
    # ดำเนินการทำงานคำสั่ง 'ip address print'
    output_lines = ssh_connection.execute_command('ip address print')
    # ปิดการเชื่อมต่อ SSH
    ssh_connection.close()

```

```
# สร้างเส้นทางของหน้าเว็บ
@app.route('/')
def index():
    return render_template('index.html', output_lines=output_lines)
if __name__ == '__main__':
    app.run(debug=True, host='127.0.0.2', port=5001) # เริ่มการทำงานของเซิร์ฟเวอร์ Flask
```

ไฟล์ template index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    # ระบุการตั้งค่า viewport เพื่อให้เว็บไซต์ปรับขนาดได้อย่างเหมาะสมกับอุปกรณ์
    <title>SSH Command Output</title>
    # กำหนดหัวข้อของเอกสาร
    <style>
        pre {
            white-space: pre-wrap; /* Preserve line breaks */
            # คำสั่ง CSS เพื่อให้รักษารูปแบบการขึ้นบรรทัดใหม่ใน <pre> อย่างถูกต้อง
            font-family: monospace;
            # กำหนดรูปแบบตัวอักษรให้เป็น monospace เพื่อความสมดุลในการแสดงผล
        }
    </style>
</head>
<body>
    <h1>SSH Command Output</h1> # ส่วนของหัวข้อหลักของหน้าเว็บ
    <pre> # แท็กที่ใช้แสดงข้อมูลโดยรักษารูปแบบการเรียงลำดับและการเว้นวรรคให้อยู่ในรูปแบบเดิม
        {% for line in output_lines %}
        # เริ่มต้นการวนลูปผ่าน output_lines ซึ่งอาจมาจากข้อมูลที่ส่งมาจากภาษา Python โดยใช้
        Flask หรือเฟรมเวิร์กเกอร์อื่น ๆ
```

```

        {{ line }} # แสดงค่าของแต่ละบรรทัดใน output_lines
    {% endfor %} # สิ้นสุดการวนลูป
</pre>
</body>
</html>

```

ไฟล์ interface.py

```

from flask import Flask, render_template # เรียกใช้งาน Flask และ render_template
จาก Flask

```

```

import paramiko # เรียกใช้งาน paramiko เพื่อทำการเชื่อมต่อ SSH
app = Flask(__name__) # สร้างอ็อบเจกต์ Flask
class SSHConnection:

```

```

    def __init__(self, host, username, password):
        self.host = host # กำหนด host
        self.username = username # กำหนด username
        self.password = password # กำหนด password
        self.client = paramiko.SSHClient() # สร้างอ็อบเจกต์ SSHClient จาก paramiko
        self.client.set_missing_host_key_policy(paramiko.AutoAddPolicy()) # กำหนด
นโยบายการเพิ่มโฮสต์อัตโนมัติ

```

```

    def connect(self):
        try:
            self.client.connect(self.host, username=self.username,
password=self.password) # เชื่อมต่อ SSH
โดยใช้ข้อมูลที่กำหนด
            print(f"Connected to {self.host}") # พิมพ์ข้อความแสดงว่าเชื่อมต่อสำเร็จ
        except Exception as e:
            print(f"Connection failed: {e}") # พิมพ์ข้อความแสดงว่าเชื่อมต่อล้มเหลว

```

```

def execute_command(self, command):
    try:
        stdin, stdout, stderr = self.client.exec_command(command) # ส่งคำสั่ง SSH
        ไปยังอุปกรณ์
        return stdout.readlines() # อ่านผลลัพธ์ที่ได้กลับมา
    except Exception as e:
        print(f"Command execution failed: {e}") # พิมพ์ข้อความแสดงว่าการท างานคำสั่งล้มเหลว
        return []

def close(self):
    self.client.close() # ปิดการเชื่อมต่อ SSH
    print("Connection closed") # พิมพ์ข้อความแสดงว่าการเชื่อมต่อถูกปิด
    # สร้างอ็อบเจกต์ SSHConnection
    ssh_connection = SSHConnection('192.168.88.1', 'admin', 'gtavc15150')
    # เชื่อมต่อ SSH
    ssh_connection.connect()
    # ดำเนินการทำงานคำสั่ง 'interface print'
    output_lines = ssh_connection.execute_command('interface print')
    # ปิดการเชื่อมต่อ SSH
    ssh_connection.close()
    # สร้างเส้นทางของหน้าเว็บ
    @app.route('/')
    def index():
        # ส่งผลลัพธ์ไปยัง HTML template
        return render_template('index.html', output_lines=output_lines)

if __name__ == '__main__':
    app.run(debug=True, host='127.0.0.2', port=5002) # เริ่มการทำงานของเซิร์ฟเวอร์ Flask

templates ไฟล์ index.html
<!DOCTYPE html> # ระบุประเภทของเอกสารว่าเป็น HTML5
<html lang="en"> # เปิดแท็ก HTML และระบุภาษาเป็นภาษาอังกฤษ
<head> # เริ่มต้นส่วนหัวของเอกสาร HTML
    <meta charset="UTF-8">
    # ระบุการเข้ารหัสเป็น UTF-8 เพื่อรองรับตัวอักษรและภาษาหลายภาษา

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
# ปรับขนาด viewport เพื่อให้เข้ากับขนาดอุปกรณ์และขนาดหน้าจอเริ่มต้น
<title>SSH Command Output</title> # กำหนดชื่อเว็บไซต์
<style> # เริ่มต้นส่วนสไตล์ของเอกสาร HTML
    pre {
        white-space: pre-wrap; /* Preserve line breaks */
        # คำสั่งนี้บอกให้เครื่องหมายว่างและข้อความที่ยาวเกินกว่าความกว้างของพื้นที่ที่กำหนดไว้
        ในแท็ก <pre> ให้มีการขึ้นบรรทัดใหม่เอง โดยไม่ต้องรอการพับบรรทัดจาก HTML
        font-family: monospace;
        # ระบุให้ข้อความภายในแท็ก <pre> ใช้แบบอักษร monospace เพื่อให้แสดงผลแบบ
        เดียวกับที่กำหนดไว้ ซึ่งจะช่วยให้ข้อความสามารถแสดงผลในลักษณะที่สม่ำเสมอ โดยที่ตัวอักษรแต่
        ละตัวมีขนาดเท่ากัน
    }
</style>
</head>
<body>
    <h1>SSH Command Output</h1> ‘
    <pre>
        {% for line in output_lines %}
        # เป็นส่วนของโค้ด Python (Jinja Template) ที่ใช้ในการวนซ้ำผ่านข้อมูลที่มีชื่อว่า
        output_lines และนำข้อมูลแต่ละบรรทัดมาแสดงผลใน HTML
        {{ line }}
        {% endfor %} # ใช้ในการแสดงข้อมูลในแต่ละบรรทัดที่ถูกรวนซ้ำ
    </pre>
</body>
</html>

```

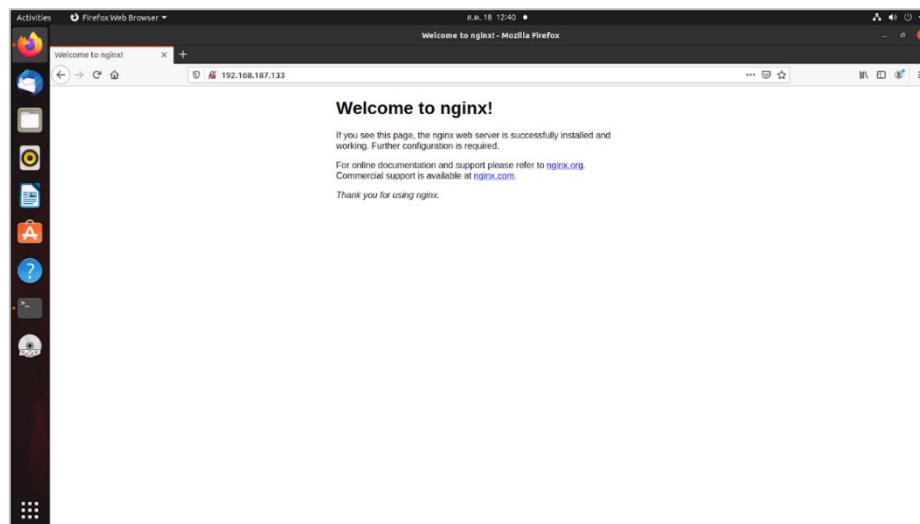
วิธีคอนฟิก loandblance

ติดตั้งแพ็คเกจ nginx ด้วยคำสั่งต่อไปนี้

sudo apt update

sudo apt install nginx

ทดลองใช้ browser เข้าเว็บไซต์ โดยระบุ IP ของเครื่องเซิร์ฟเวอร์

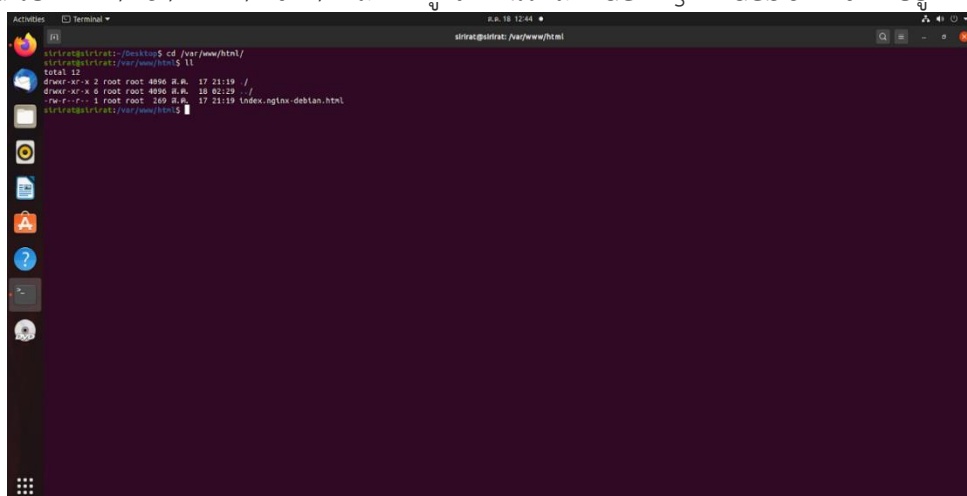


ภาพที่ ข.1 ตัวอย่างหน้าจอเว็บไซต์ฟอลต์ของ เว็บเซิร์ฟเวอร์ nginx

browser เข้าเว็บเซิร์ฟเวอร์โดยการระบุ IP ของตัวเซิร์ฟเวอร์ แต่ไม่ได้ระบุพาร หรือชื่อไฟล์ใดๆ
ต่อท้าย

เพราะฉะนั้น เว็บเซิร์ฟเวอร์ nginx จะไปเรียกไฟล์ที่อยู่ในไดเรกทอรีที่ระบุเป็น root แล้วไล่หาไฟล์
ที่อยู่ในคอนฟิก index ตามลำดับ

ถ้า cd ไปที่พาร /var/www/html/ และ ll ดู จะเห็นไฟล์ index.nginx-debian.html อยู่



ภาพที่ ข.2 ไฟล์ index.nginx-debian.html

ทำการแก้ไขไฟล์ index.nginx-debian.html เพื่อดูการเปลี่ยนแปลงของเว็บ โดยใช้คำสั่ง
sudo nano inden.nginx-debian.html

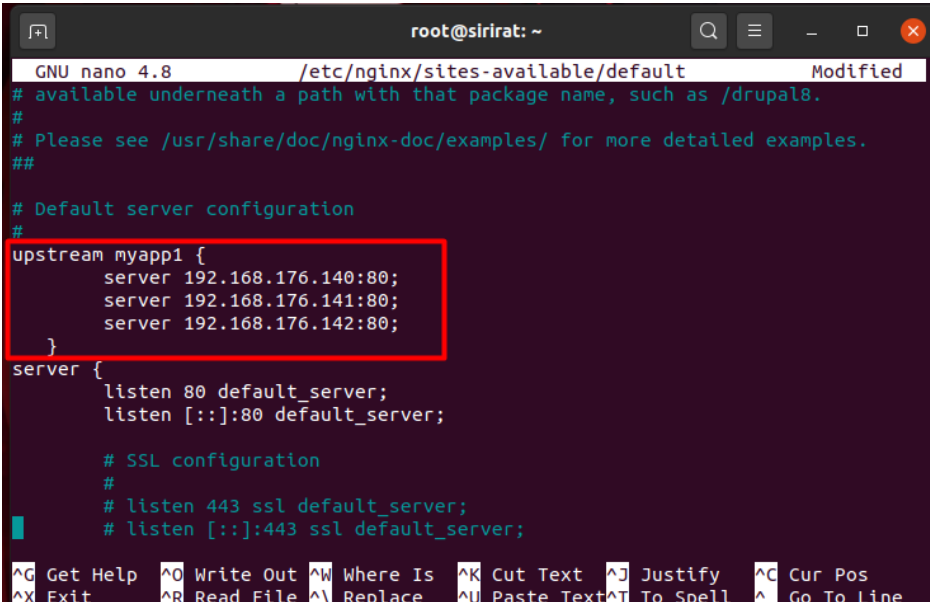
แก้ไขไฟล์เพื่อที่จะไว้สำหรับทดสอบ Load balance แต่ละเครื่องควรแตกต่างกัน เพื่อดูความเปลี่ยนแปลงตอนทดสอบ Load balance
ควรมีอย่างน้อย 2 เครื่อง 1 เครื่อง ต่อ 1 IP ไม่รวมกับเครื่องที่ Load balance

ในเครื่องที่เป็น Load balance
ติดตั้งแพ็คเกจ nginx โดยใช้คำสั่งต่อไปนี้
sudo apt update
sudo apt install nginx

แก้ไขไฟล์ default ที่พาท /etc/nginx/sites-available/ โดยใช้คำสั่ง
sudo nano /etc/nginx/sites-available/default

และเพิ่มข้อความต่อไปนี้ลงไป

```
upstream myapp1 {
    server 192.168.176.140(IP ของเครื่องเว็บเซิร์ฟเวอร์);
    server 192.168.176.141(IP ของเครื่องเว็บเซิร์ฟเวอร์);
    server 192.168.176.142(IP ของเครื่องเว็บเซิร์ฟเวอร์);
}
```



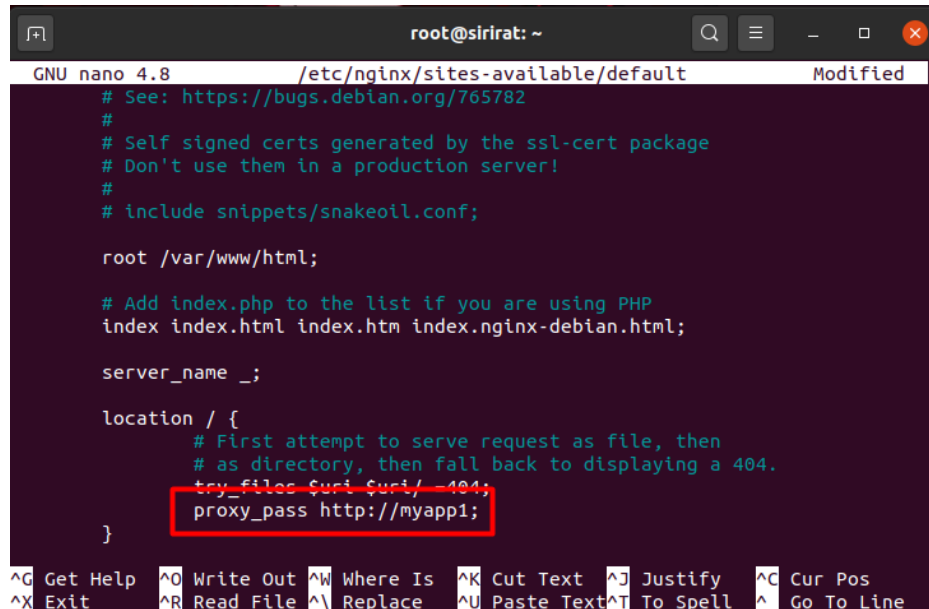
```

root@sirirat: ~
GNU nano 4.8 /etc/nginx/sites-available/default Modified
# available underneath a path with that package name, such as /drupal8.
#
# Please see /usr/share/doc/nginx-doc/examples/ for more detailed examples.
##
# Default server configuration
#
upstream myapp1 {
    server 192.168.176.140:80;
    server 192.168.176.141:80;
    server 192.168.176.142:80;
}
server {
    listen 80 default_server;
    listen [::]:80 default_server;

    # SSL configuration
    #
    # listen 443 ssl default_server;
    # listen [::]:443 ssl default_server;
  
```

ภาพที่ ข.3 upstream myapp1

#เพิ่มข้อความในส่วน location
 proxy_pass http://myapp1;



```

GNU nano 4.8 /etc/nginx/sites-available/default Modified
# See: https://bugs.debian.org/765782
#
# Self signed certs generated by the ssl-cert package
# Don't use them in a production server!
#
# include snippets/snakeoil.conf;

root /var/www/html;

# Add index.php to the list if you are using PHP
index index.html index.htm index.nginx-debian.html;

server_name _;

location / {
    # First attempt to serve request as file, then
    # as directory, then fall back to displaying a 404.
    try_files $uri $uri/ -- 404;
    proxy_pass http://myapp1;
}

^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Paste Text ^T To Spell  ^_ Go To Line
  
```

ภาพที่ ข.4 เพิ่มข้อความในส่วน location

บันทึกไฟล์ default และทำการรีสตาร์ท nginx โดยใช้คำสั่ง
 sudo systemctl restart nginx

ทดลองใช้ browser โดยระบุ IP ของเครื่องเซิร์ฟเวอร์ Load balance เมื่อรีเฟรชหน้า browser
 จะเห็นการเปลี่ยนแปลงตามที่ได้แก้ไขไว้