Restful API And Microservices with Python

Day 14

Day 14 - Overview

- Production grade uWSGI server
- NGINIX reverse proxy

Prerequisite

- VM with windows OS
- Python 3.8 or >
- Visual Studio Code - Code Editor
- Postman
- GIT

https://github.com/saurav-samantray/flask-microservices-training/blob/main/slides/Setup%20GIT%20in%20your%20Local%20system.pdf

- Docker

Sync your fork for Day 14 activities

- Follow the below document to sync your fork and update local repository.

  [*https://github.com/saurav-samantray/flask-microservices-training/blob/main/slides/Setup%20GIT%20in%20your%20Local%20system.pdf*](https://github.com/saurav-samantray/flask-microservices-training/blob/main/slides/Setup%20GIT%20in%20your%20Local%20system.pdf)

- Stop and remove and already running container and volumes
- Navigate to below location

  C:\workspace\flask-microservices-training\day14\user-management-service

- Execute below command

  docker compose down -v

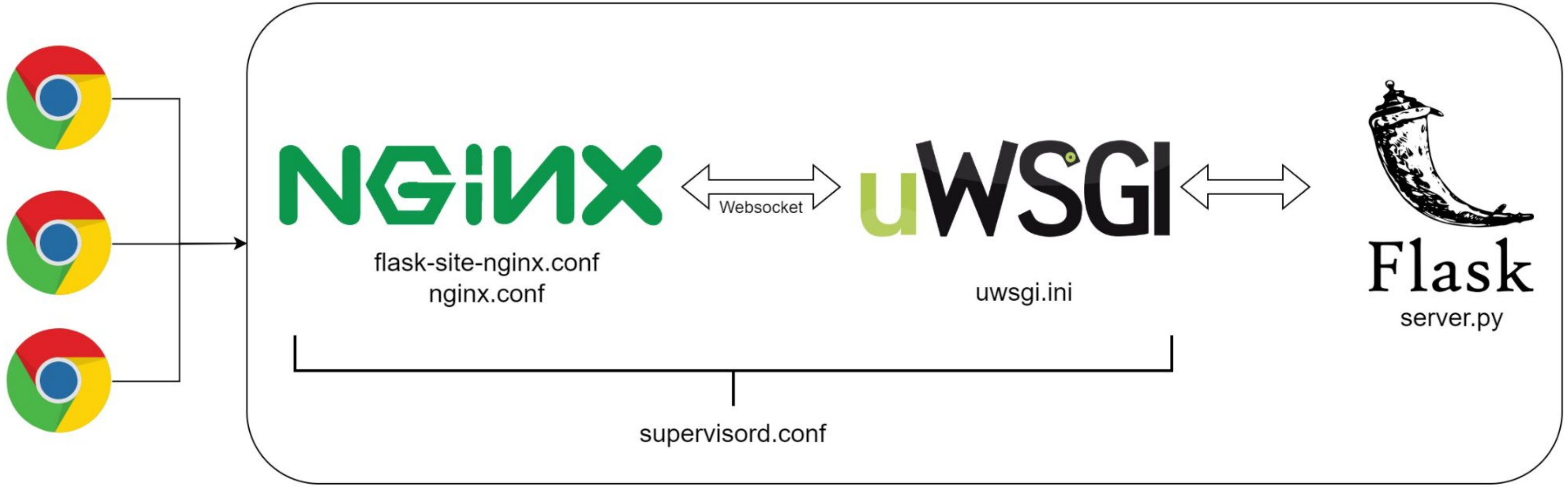
Development Server vs Production Server

when you application in development mode, is only single threaded.

It can only handle a single connection at a time. If two connections come in together the server will handle one and make the other wait.

A wsgi server with apache/nginx in front will handle many connections at once.

Architecture Diagram



Nginx Configuration

```
server {
   location / {
      try_files $uri @user-management-service;
   }
   location @user-management-service {
      include uwsgi_params;
      uwsgi_pass unix:///tmp/uwsgi.sock;
   }
}
```

uWSGI Configuration

```
[uwsgi]
module = server                 ; module/file that is used to access flask app object
callable = app                  ; variable that defines the flask app object
enable-threads = true           ; To run uWSGI in multithreading mode

uid = nginx                     ; docker environment user
gid = nginx                     ; docker environment user

socket = /tmp/uwsgi.sock
chown-socket = nginx:nginx
chmod-socket = 664

cheaper-algo = busyness         ; algorithim to distribute traffic
processes = 128                 ; Maximum number of workers allowed
cheaper = 1                     ; Minimum number of workers allowed - default 1
cheaper-initial = 4             ; Workers created at startup
cheaper-overload = 5            ; Will check busyness every 5 seconds.
cheaper-step = 3                ; How many workers to spawn at a time

auto-procname = true            ; Identify the workers
procname-prefix = "rhs-svc "    ; Note the space. uWSGI logs will be prefixed with "rhs-svc"
```

Dockerfile update

```
FROM python:3.10

#Install NGINX and Supervisor
RUN apt-get update
RUN apt-get install -y --no-install-recommends \
        libatlas-base-dev gfortran nginx supervisor

LABEL maintainer="saurav.samantray@gmail.com"
WORKDIR /user-management-service

COPY requirements.txt requirements.txt
RUN pip3 install -r requirements.txt

# create a new user called nginx. Avoid using root
RUN useradd --no-create-home nginx

#Remove default ngnix configuration
RUN rm /etc/nginx/sites-enabled/default
RUN rm -r /root/.cache

#Copy custom configuration to the image
COPY nginx.conf /etc/nginx/
COPY flask-site-nginx.conf /etc/nginx/conf.d/
COPY uwsgi.ini /etc/uwsgi/
COPY supervisord.conf /etc/

#Copy the source code from local to docker image
COPY . .

#Start the supervisord which will take care of starting ngnix and uWSGI server on container startup
CMD ["/usr/bin/supervisord"]
```

Building Running Docker container

Building Docker containers

- docker-compose build

Running Docker container using compose

- docker-compose run

Stopping all container and removing volumes

- docker-compose down -v

Testing the efficiency if the setup

```
import requests
import json
import time

url = "http://localhost:8080/api/auth"

payload = json.dumps({
  "email": "saurav@gmail.com",
  "password": "saurav"
})
headers = {
  'Content-Type': 'application/json'
}

iterations = 100
start = time.time()
for i in range(iterations):
    response = requests.request("POST", url, headers=headers, data=payload)
    print(response.text)
print(f"Time taken for {iterations} number requests: {time.time() - start} seconds")
```

Q and A