WEB-DEPLOYMENT

Let us first understand the following keywords:

Nginx -NGINX is open source software for web serving, reverse proxying, caching, load balancing, media streaming, and more.For our purpose we are using web-server properties of it.

Flask- Flask is a web framework, it's a Python module that lets you develop web applications easily. It's has a small and easy-to-extend core.It does have many cool features like url routing, template engine.

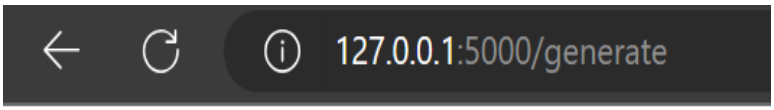
Requirement.txt- it is a file that is attached to the frontend and backend .It basically shows all the required files that will be used to run the corresponding files in that folder.

Before starting to apply, let us think that how would things work-

1. First we will explore the backend code (i.e, flask file named as app.py)

```python
# Import required modules
import random, string
from flask import Flask, render_template
from flask_cors import CORS

# Create a Flask app instance
app = Flask(__name__, template_folder='templates')

# Enable Cross-Origin Resource Sharing (CORS) for the app
cors = CORS(app, resources={r"*": {"origins": "*"}})

# Define a route that generates a random string
@app.route("/generate")
def index():
    # Generate a random string of length 10 using ASCII l
    random_string = ''.join(random.choice(string.ascii_le

    # Return the generated random string prefixed with "v
    return "v1-" + random_string

# Entry point for running the app
if __name__ == "__main__":
    # Start the Flask development server
    # Listen on all available network interfaces ("0.0.0.
    # Enable debugging mode for better error messages ("d
    app.run(host='0.0.0.0', debug=True)
```

This code simply prints the random string on path "generate" to show output when you run "python3 app.py" in the directory of the backend in the terminal.



v1-ylSyvTLwuo

Confused, why you are not able to get the output on the browser. Don't worry it is just to show u how the output will look like , further things will be discussed beneath.

2. Now we will discuss the frontend- it consists of two files the nginx.conf and the index.html.

   Nginx.conf

```
1  user  nginx;
2  worker_processes  1;
3
4  error_log  /var/log/nginx/error.log warn;
5  pid        /var/run/nginx.pid;
6
7
8  events {
9      worker_connections  1024;
10 }
11
12
13 http {
14     include       /etc/nginx/mime.types;
15     default_type  application/octet-stream;
16
17     log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
18                       '$status $body_bytes_sent "$http_referer" '
19                       '"$http_user_agent" "$http_x_forwarded_for"';
20
21     access_log  /var/log/nginx/access.log  main;
22
23     sendfile        on;
24     #tcp_nopush     on;
25
26     keepalive_timeout  65;
27
28     #gzip  on;
29
30     # include /etc/nginx/conf.d/*.conf;
31
32     server {
33         listen       80;
34         server_name  localhost;
35
36         #charset koi8-r;
37         #access_log  /var/log/nginx/host.access.log  main;
38
39         location / {
40             root   /usr/share/nginx/html;
41             index  index.html index.htm;
42         }
43
44         location /generate {
45             proxy_pass http://backend:5000;
46             proxy_http_version 1.1;
47             proxy_set_header Upgrade $http_upgrade;
48             proxy_set_header Connection "upgrade";
49             proxy_pass_request_headers on;
50         }
51     }
52 }
```

Do not need to focus very much on this file to understand.
It defines how Nginx should handle various aspects of incoming HTTP requests, server behavior, and proxying.

We need to understand some basic concepts like on line 32 the server block configures the virtual server. Here we specified the listen port as 80 and server_name as “localhost” to run it on the localhost of machine.
The location block configures settings for the "/generate" URL location.

- proxy_pass http://backend:5000:

This line indicates that requests to "/generate" will be proxied to the backend server which is built with flask, It will give a random string

Now looking inside index.html

```html
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
  initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Index</title>
  <script src="https://ajax.googleapis.com/ajax/libs/
  jquery/3.3.1/jquery.min.js"></script>
</head>

<body>
  <h1 style="color: blue" id="data"></h1>
</body>

<script>
  $(document).ready(function(){
    $.get(window.location.href + "generate", function
    (data, status){
      document.getElementById("data").innerText = data;
    })
  })
</script>

</html>
```

It is simple index page that will display the content of the backend
And additionally it will apply some of the css to make it appear beautiful
In our case the output will be shown in color blue as:

---

v1-CkaODBJgmh

It will give a random string, so every time the page gets refreshed , it will provide a different output.

So as an overview our approach will be

Running the backend → taking the address from their→ replacing it in the nginx.conf file at the place of of http://backend:5000 → then simply try to access the page

Its just an overview, we need to perform number of different thing while doing the actual implementation
So for what we are waiting for let's dive into it

IMPLEMENTATION →

1. I am going to perform it in a VM named "practice-web" created in GCP.

2. Run the command sudo apt-get update & sudo apt-get install python3-pip. It is necessary to have access to the pip package manager for Python 3, which is used to install and manage Python packages.

```
lakshya_datir2001@practice-web:~$ python3
Python 3.9.2 (default, Feb 28 2021, 17:03:44)
[GCC 10.2.1 20210110] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> exit()
```

3. Clone the repository to your VM.

4. Now run the backend first-

- go to the backend directory and run the command pip3 install -r requirements.txt which will install all the required dependencies
- Run the command python3 app.py to start backend server on port 5000-
  It will look like-

```
lakshya_datir2001@practice-web:~/backend$ python3 app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production dep
loyment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://10.0.1.7:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 102-613-752
```

5. To start the frontend we need to configure nginx

- go to the frontend directory and run the command pip3 install -r requirements.txt which will install all the required dependencies for frontend.
- Install nginx with command- sudo apt-get install nginx
- Replace the file at usr/share/nginx/html with our index.html
- In the nginx.conf file at location /generate give the backend address we got from above at proxy_pass
- Replace the file at etc/nginx/ with our nginx.conf
- Add user nginx with command- sudo adduser nginx
- Reload nginx with systemctl reload nginx
- Check the page with external ip address in your browser, you'll get the desired output.

---

v1-CkaODBJgmh