DevOps Internship Challenge Report

Live Cloud Server Running on this IP address : http://13.201.166.239:8080/

1. Initial File Structure and Directory Misconfigurations

Issues: Initially, the application structure was set up with Dockerfiles in both the Python and Nginx directories. However, when trying to build the images, errors were encountered due to misspelled file names or incorrect paths. The structure was then validated to ensure all required files, such as `app.py` for Flask and `nginx.conf` for Nginx, were in the correct directories and the files are named correctly. There were errors of name like port numbers are expressed as numeration and file names were not correctly spelled like appy.py instead of app.py etc. which was corrected.

Resolution:

- Reviewed and confirmed directory structure.
- Corrected all the file names and all the ports are expressed in proper notation.
- Created an `index.html` file under an `html` subfolder in the Nginx directory for proper setup.

2. Orphaned Containers and Naming Conflicts

Issues : Errors occurred due to orphaned containers and conflicts with container names when rebuilding Docker Compose services. Docker showed errors like 'requested access to the resource is denied,' as old container names conflicted with new build containers.

Resolution:

- Used the `--remove-orphans` flag to remove any orphaned containers.
- Renamed services in `docker-compose.yml` to avoid naming conflicts.
- Deleted any residual containers and images with conflicting names using `docker rm` and `docker rmi` as needed.

3. Port Conflicts and Nginx Configuration

Issues : There was a port conflict because Nginx was configured to use a port i.e. **80** (quite a busy one) that was already occupied on the host machine.

Resolution:

- Changed the Nginx `docker-compose.yml` file to bind to port `8080` on the host. Here we assigned port 8080 to nginx to listen externally.
- Verified network setup in `docker-compose.yml` to ensure both services were connected to the same Docker bridge network.

4. Flask Application Connectivity

Issues: Although the Flask application was running on `0.0.0.0:8000`, there were connectivity issues when accessing it through Nginx.

Resolution:

- Confirmed Nginx was forwarding requests to the Flask service correctly by testing with `curl` on `localhost:8080`.

5. Missing Logs from Nginx

Issues : Nginx logs were initially not appearing as expected due to misconfigurations in the volume mounts for logging.

Resolution:

- Mounted a directory for Nginx logs in `docker-compose.yml` to `~/nginx-logs` on the host.
- Verified access and error logs were properly created in the mounted directory, confirming Nginx activity.

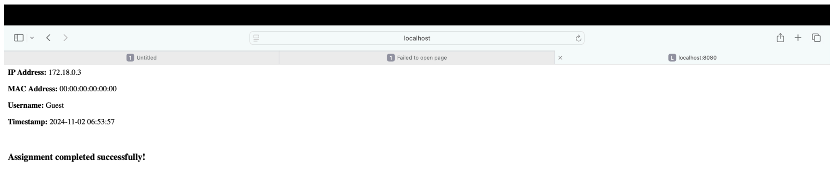
6. Favicon Error with Flask

Issue : Flask returned a 404 error for the `favicon.ico` request as no favicon was provided in the application.

Resolution:

- Suppressed `favicon.ico` requests by explicitly handling it in Flask.

PFA the images:



IP Address: 172.18.0.3

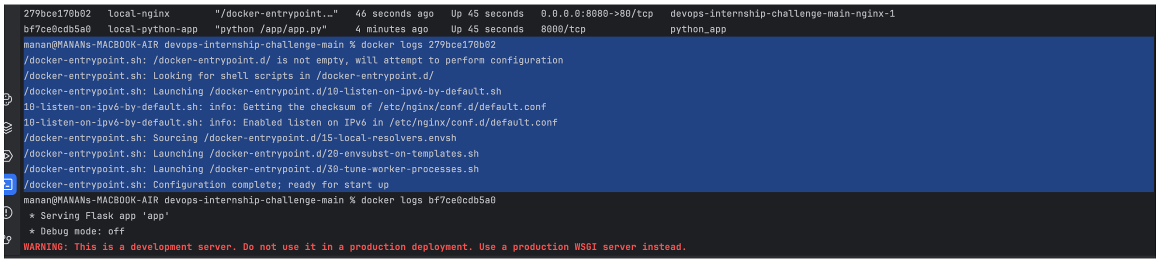
MAC Address: 00:00:00:00:00:00

Username: Guest

Timestamp: 2024-11-02 06:53:57

Assignment completed successfully!

Fig.1 Application running in the browser



```
279bce170b02   local-nginx        "/docker-entrypoint.…"   46 seconds ago   Up 45 seconds   0.0.0.0:8080->80/tcp   devops-internship-challenge-main-nginx-1
bf7ce0cdb5a0   local-python-app   "python /app/app.py"     4 minutes ago    Up 45 seconds   8000/tcp               python_app
manan@MANANs-MACBOOK-AIR devops-internship-challenge-main % docker logs 279bce170b02
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
manan@MANANs-MACBOOK-AIR devops-internship-challenge-main % docker logs bf7ce0cdb5a0
 * Serving Flask app 'app'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
```

Fig.2 Nginx Logs

Conclusion

Throughout this process, I resolved several issues related to file paths, Docker configuration, and networking. Each issue provided insights into Docker container management, service networking, and application deployment best practices. The final setup successfully runs a Dockerized Flask application with Nginx as a reverse proxy, accessible through port `8080` on the host.