Deploying Clincian Webapp to AWS EC2

GDP Group 15

Joseph Padden - jp3g20

Rory Coulson - rc3g20

Neeraja Jayaraj Menon - njm1g20

Dillon Geary - dgfg1g20

December 2023

Steps to Deploy on New Instance

Reference material:
https://github.com/calvin-puram/AWS-EC2-Demo
https://medium.com/@shefaliaj7/hosting-react-flask-mongodb-web-application-on-aws-part-4-hosting-web-application-b8e205c19e4

0.1 Setup EC2 Instance

1. Create an Ubuntu EC2 instance on AWS with configured inbound security rules to allow HTTP, HTTPS and SSH.
2. Update packages: `sudo apt update  sudo apt upgrade -y`
3. Install NodeJS, NPM, PostgreSQL, Nginx onto the instance
4. Clone the Clinician Webapp Repository via Git
5. Navigate and run `npm install` for both '/backend/server/' and '/frontend/' to install node dependencies.
6. If building locally run `sudo npm run build` within '/frontend/', make sure to grant permission to the build path via `chmod +x /path/to/build/`, granting each section individually

0.2 Deploying NodeJS, Express, React App

0.2.1 Install and configure PM2:

1. `sudo npm install pm2 -g`
2. `pm2 start /home/ubuntu/<path/to>/server/server.js --name web-app`
3. `pm2 startup` and run the output command it provides
4. check it's running with `pm2 status`
5. once running run `pm2 save`
6. `cd /etc/nginx/sites-available`
7. `sudo cp default clinician_webapp.xyz`
8. `sudo nano clinician_webapp.xyz` and paste in the following: (updating values where necessary)

```
server {
        listen 80;
        listen [::]:80;

        # example:
        # root
            /home/ubuntu/apps/GDP15ActivePoints_Clinician_WebApp/frontend/build;
        root /path/to/build;
        # Add index.php to the list if you are using PHP
        index index.html index.htm index.nginx-debian.html;

        # Change to instance public ip address at every reinitiation of
           instance (OR set to static elastic IP just once if available)
        server_name 13.41.226.171;

        location / {
               try_files $uri /index.html;
        }

        location /flask {
               include proxy_params;
               proxy_pass http://127.0.0.1:5002;
        }

        location /api {
            proxy_pass http://localhost:5001;
            proxy_http_version 1.1;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection 'upgrade';
            proxy_set_header Host $host;
            proxy_cache_bypass $http_upgrade;
        }

}
```

9. `sudo ln -s /etc/nginx/sites-available/clinician_webapp.xyz /etc/nginx/sites-enabled/`
10. `sudo systemctl restart nginx`
11. Use `pm2 logs` to debug for errors if needed

0.2.2 Setup environment variables

1. `sudo nano .bashrc`
2. `export ACTIVE_POINTS_DB_PSWD='<posgres_db_password>'`
3. `export ACTIVE_POINTS_DB_USERNAME='<posgres_db_username>'`
4. `source /etc/profile`
5. (alternatively define them in a .env file)

0.3 Deploying Flask API on Gunicorn Server:

1. Create a Python virtual environment in 'backend/ai-module/src/api/':

(a) `sudo apt install python3-venv`

(b) `python3 -m venv .venv`

(c) `source .venv/bin/activate`

2. Install required packages:

   (a) `pip3 install gunicorn`

   (b) `pip3 install Flask`

   (c) `pip3 install python-dotenv`

   (d) `pip3 install flask-cors`

   (e) `pip3 install flask-mongoengine`

   (f) `pip3 install requests`

3. Install project required packages:

   (a) `pip install psycopg2-binary`

   (b) `pip install shap`

   (c) `pip install xgboost --no-cache-dir`

   (d) `pip install pandas==2.1.2`

   (e) `pip install numpy==1.26.1`

   (f) `pip install matplotlib==3.7.2`

   (g) `pip install seaborn==0.12.2`

4. Navigate to 'ai-module' directory and run '`pip install .`'

5. (optionally run '`python apiInterface.py`' in 'ai-module/src/api/' to test it runs, then cancel after verifying)

6. `sudo nano /etc/systemd/system/clinician_webapp.service`

7. Example of server file (update paths and environment variables):

```
[Unit]
Description=Clinician Webapp Flask API
After=network.target

[Service]
User=ubuntu
WorkingDirectory=/home/ubuntu/apps/GDP15ActivePoints_Clinician_WebApp/backend/
ai-module/src/api

ExecStart=/home/ubuntu/apps/GDP15ActivePoints_Clinician_WebApp/backend/ai-module/src/
api/.venv/bin/gunicorn -b localhost:5002
--env ACTIVE_POINTS_DB_USERNAME=<username_value>
--env ACTIVE_POINTS_DB_PSWD=<password>   apiInterface:api
Restart=always

[Install]
```

```
WantedBy=multi-user.target

Restart=always

[Install]
WantedBy=multi-user.target
```

8. `sudo systemctl daemon-reload`

9. `sudo systemctl start clinician_webapp`

10. `sudo systemctl status clinician_webapp`

Open browser and type the public IP of EC2 instance (or elastic ip if available) to view the deployed webapp. If correctly configured it should show the react webapp and should pull the data from the database and display the AI predictions, badge and recommendations.

Steps to Update Deployed App

1. Pull latest version from Git

2. ‘`npm install`‘ where necessary if changes to dependencies

3. ‘`npm run build`‘ from ‘/frontend/’ if building locally

4. `sudo systemctl restart nginx` to see changes to the React App UI

5. `pm2 kill` and redo pm2 steps from above if changes to the express app

6. To update the AI API follow the steps above again, entering into the python vm, then run the reload functions to save the changes

7. If stopped and started the instance remember to update the IP value from the sites-available directory file unless using an elastic static IP