DevOps Assessment Documentation

1. Introduction

This document outlines the steps taken to complete the deployment of a Node.js application (using express-rest-boilerplate) with the following technologies:

- **Node.js**
- **NGINX** (Reverse Proxy)
- **MongoDB** (Database)
- **Redis** (Caching)
- **AWS EC2** (Hosting)

The application is deployed on **AWS EC2 instance** using NGINX as a reverse proxy, with the application running on **port 3000**.

2. Environment Setup

- **AWS EC2 Instance:** Ubuntu 20.04
- **Node.js Version:** 18.20.8
- **NGINX Version:** 1.24.0
- **MongoDB Version:** 7.0
- **Redis Version:** 7.0

3. Steps to Complete the Assessment

Step 1: Application Setup

1. **Clone the Project**
    - First, clone the express-rest-boilerplate repository into your EC2 instance:
    - git clone https://github.com/danielfsousa/express-rest-boilerplate.git
    - cd express-rest-boilerplate
2. **Install Dependencies**
    - Install necessary npm dependencies:
        1. npm install
3. **Setup .env File**
    - Add the required environment variables in the .env file. Example:
    - NODE_ENV=development

- PORT=3000
- JWT_SECRET=your_secret_key
- JWT_EXPIRATION_MINUTES=15
- MONGO_URI=mongodb://localhost:27017/appname
- REDIS_URL=redis://localhost:6379

4. **Run the Application**
   - Start the Node.js app using nodemon:
     1. npm run dev

Step 2: NGINX Setup (Reverse Proxy)

1. **Install NGINX**
   - If NGINX is not installed, run:
   - sudo apt update
   - sudo apt install nginx
2. **Configure NGINX for Reverse Proxy**
   - Edit the NGINX config file:
   - sudo nano /etc/nginx/sites-available/default
   - Update it to use the following configuration:
   - server {
   - listen 80;
   - server_name _;
   - location / {
   - proxy_pass http://localhost:3000;
   - 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;
   - }

- }
- **Save** and exit the editor.

3. **Restart NGINX**
    - Restart the NGINX service to apply changes:
    - sudo systemctl restart nginx

Step 3: MongoDB & Redis Setup

1. **Install MongoDB**
    - For MongoDB, run:
    - sudo apt install -y mongodb
    - sudo systemctl enable mongodb
    - sudo systemctl start mongodb
2. **Install Redis**
    - For Redis, run:
    - sudo apt install -y redis-server
    - sudo systemctl enable redis-server
    - sudo systemctl start redis-server
3. **Verify MongoDB and Redis are Running**
    - Check MongoDB:
    - systemctl status mongodb
    - Check Redis:
    - systemctl status redis-server

Step 4: Test the Application

1. **Check Health Endpoint**
    - Open your browser or use curl to test the status of the app:
    - curl http://localhost:3000/v1/status
    - The expected response should be OK.
2. **Test NGINX Reverse Proxy**

- Access the application through the NGINX reverse proxy:
- curl http://<your-ec2-public-ip>/v1/status
- The expected output should be OK.

Step 5: Troubleshooting

1. **If NGINX gives a "502 Bad Gateway" error:**
   - Ensure that the Node.js app is running on the correct port (3000).
   - Verify NGINX proxy settings and confirm proxy_pass points to the correct address (http://localhost:3000).
2. **If Node.js app shows a 404 Not Found error:**
   - Check the routes defined in the application (e.g., v1/status and others).
   - Verify the routes are correctly configured in index.js.
3. **If Email Error Occurs:**
   - Check the .env file for email configuration (like EMAIL_HOST, EMAIL_PORT, etc.).
   - If no email is configured, you can disable email features temporarily or set empty values.

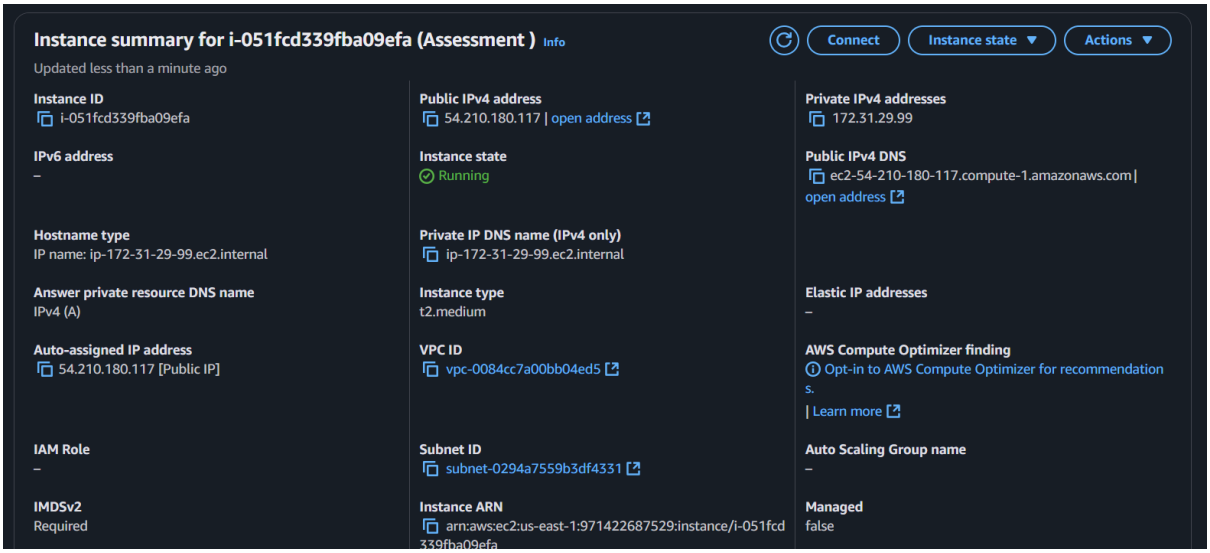
Step 6: Conclusion

Once all steps are complete, the Node.js application should be fully deployed on the EC2 instance, with NGINX as the reverse proxy. The app should be accessible via the public IP of your EC2 instance on port 80 (HTTP).

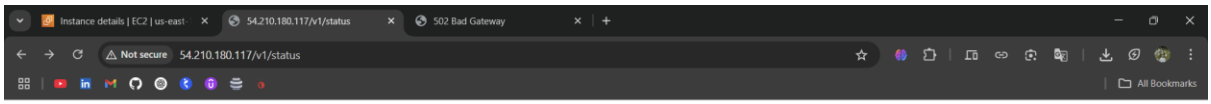
If you encounter any errors, follow the troubleshooting steps above to resolve them.

Screenshots:

1. Instance:



2. Node output in the browser: http://<public-ip>:3000/v1/status



OK

---

3. Node output in the terminal:

```
ubuntu@ip-172-31-29-99:~/node/express-rest-boilerplate/src/api/routes/v1$ npm run dev

> express-rest-es2017-boilerplate@1.2.2 dev
> nodemon ./src/index.js

[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node ./src/index.js`
info: server started on port 3000 (development)
```

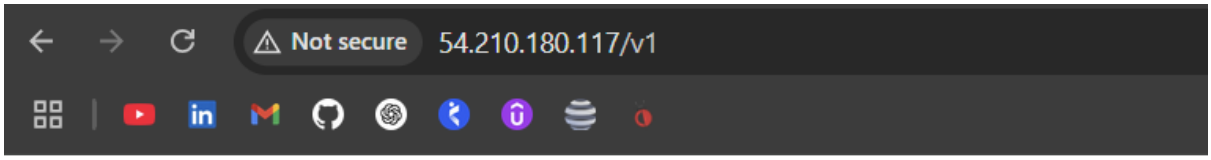
4. Nginx reverse proxy configuration:

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

    server_name _;

    location / {
        proxy_pass http://54.210.180.117:3000;
        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;
    }
}
```

5. Node output after reverse proxy through nginx: http://<public-ip>/v1



Welcome to my Node.js App!