


Deploying Nodejs App on AWS EC2 Ubuntu Instance using ansible

📅 When	@August 14, 2024 → August 17, 2024
🔖 Tags	<div>Ansible Deployment EC2 Nginx Nodejs RDS S3 Website git & github</div>
☰ Description	Automated deployment of a nodejs app from a master instance to worker instance using ansible. Tech stack: AWS (EC2,RDS,S3), ansible, nodejs, git and github, sql, nginx, pm2
👤 Employer	Personal
🌟 Status	Finished
📎 Thumbnail	
📌 Type	Personal Project

Project Documentation: Deployment of Node.js App with AWS RDS and S3 Integration using Ansible, Nginx, pm2, Github.

1. Introduction

Brief Overview

This document details the automated deployment process of a Node.js application integrated with AWS RDS (Relational Database Service) and AWS S3 (Simple Storage Service) using **ansible**. The application allows users to input their ID, name, and profile picture. The ID and name are stored in AWS RDS, while the profile picture is uploaded to AWS S3. This project

encompasses local development, environment setup, deployment on EC2 instance through other instance, and configuration using Ansible for automation.

Purpose and Functionality

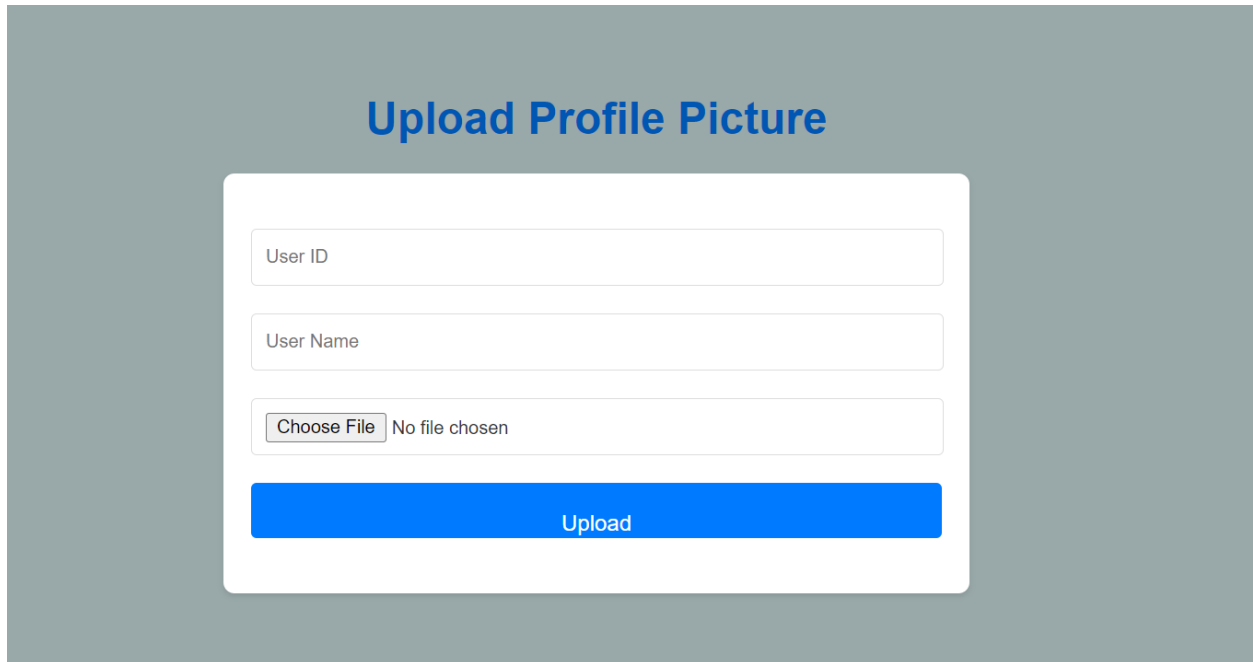
The primary objective of this project was to deploy a robust profile management system created with the help of ChatGPT to an EC2 instance. The application facilitates user profile submissions and ensures that the data is securely stored in AWS services. The document outlines each step taken to develop, configure, and deploy the application, including handling issues related to SQL integration, env variable changes, Key issues and using Ansible for automated deployment.

2. Development

Node.js App Development

1. Application Functionality:

- **Description:** Developed a Node.js application to handle user inputs for ID, name, and profile picture. The backend processes these inputs and interacts with AWS RDS for database operations and AWS S3 for file storage.
- **Local Testing:** The application was tested locally to verify functionality. Issues were encountered with SQL integration due to difficulties connecting to AWS RDS from a local environment. Testing was later performed in a AWS instance by connecting it with RDS and ensured dataflow.
- **Screenshot**



Description: The application running in a local development environment, showcasing its view only.

2. Integration with AWS RDS and S3:

- **AWS RDS Integration:** Configured the application to connect to an AWS RDS instance. This involved setting up the database schema and ensuring connectivity from the application.
- **AWS S3 Integration:** Set up AWS S3 to handle file uploads for profile pictures. The application uses AWS SDK to interact with S3 for file storage operations.
- **Errors Encountered:**
 - During master ec2 testing, there were issues related to connecting to AWS RDS. The errors were primarily due to incorrect credentials or network access settings.

3. Environment Setup

.env File Creation

1. Purpose of .env File:

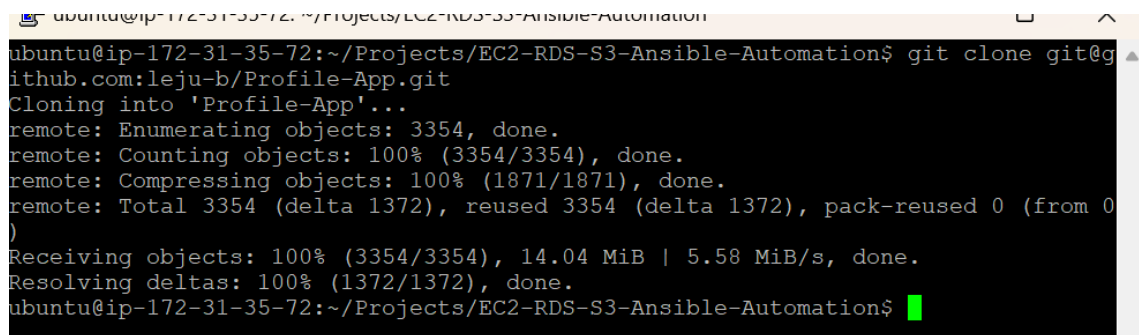
- **Description:** The `.env` file is crucial for securely storing sensitive information such as AWS credentials and database connection details. It helps in keeping these details separate from the codebase.
 - **Steps to Create:**
 - Created a `.env` file in the project's root directory.
 - Added key-value pairs for AWS credentials, database connection strings, and other environment-specific variables.
-

4. Deployment on Master Instance for testing

Initial Setup

1. Cloning the Repository:

- **Description:** Cloned the project repository onto the master EC2 instance to begin the deployment process.
- **Steps Taken:**
 - Accessed the master instance using SSH.
 - Cloned the repository with the command `git clone <repourl>`.



```
ubuntu@ip-172-31-35-72: ~/Projects/EC2-RDS-S3-Ansible-Automation
ubuntu@ip-172-31-35-72:~/Projects/EC2-RDS-S3-Ansible-Automation$ git clone git@github.com:leju-b/Profile-App.git
Cloning into 'Profile-App'...
remote: Enumerating objects: 3354, done.
remote: Counting objects: 100% (3354/3354), done.
remote: Compressing objects: 100% (1871/1871), done.
remote: Total 3354 (delta 1372), reused 3354 (delta 1372), pack-reused 0 (from 0)
Receiving objects: 100% (3354/3354), 14.04 MiB | 5.58 MiB/s, done.
Resolving deltas: 100% (1372/1372), done.
ubuntu@ip-172-31-35-72:~/Projects/EC2-RDS-S3-Ansible-Automation$
```

- ▼ Installed necessary dependencies and prepared the environment for deployment.

```
npm install
dotenv
muster
express
```

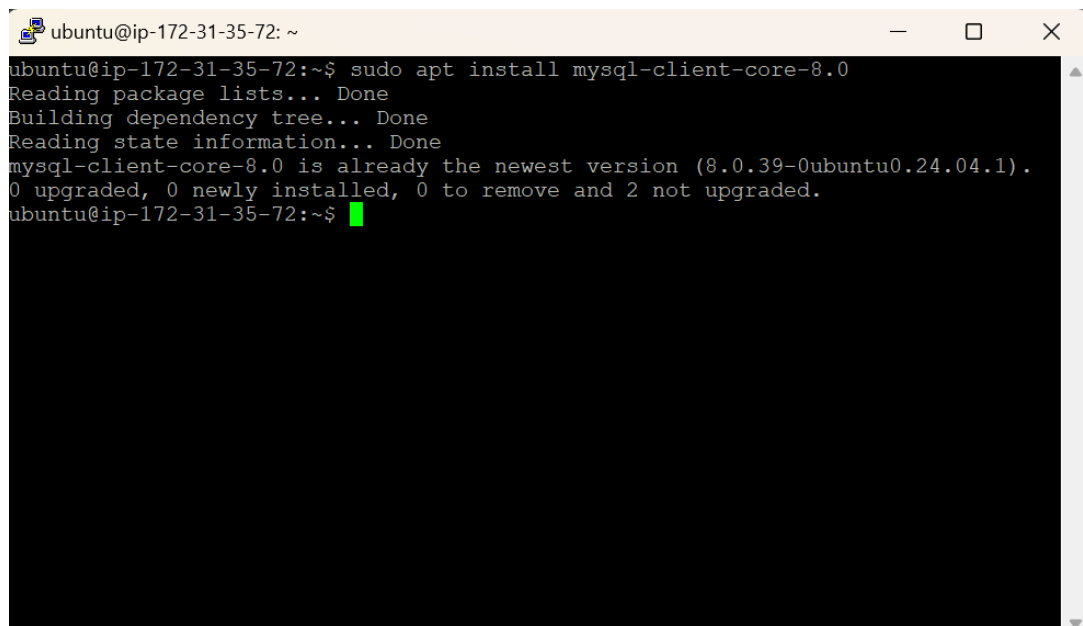
mysql2

aws-sdk

multer

2. Configuring the Instance:

- **Description:** Configured the master EC2 instance to connect to AWS RDS and set up the necessary environment for running the application.
- **Steps Taken:**
 - Established connection with RDS.
 - Configured security groups to allow connectivity to AWS RDS and to run the App.
 - Installed Node.js and other dependencies required for the application.
 - Installed my-sql client to login to rds and create database and table for the testing.

A terminal window screenshot from an Ubuntu instance. The window title is 'ubuntu@ip-172-31-35-72: ~'. The terminal shows the command 'sudo apt install mysql-client-core-8.0' being executed. The output indicates that the package is already the newest version (8.0.39-0ubuntu0.24.04.1) and that 0 packages were upgraded, 0 newly installed, and 0 to be removed. The prompt returns to 'ubuntu@ip-172-31-35-72:~\$' with a green cursor.

```
ubuntu@ip-172-31-35-72:~$ sudo apt install mysql-client-core-8.0
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
mysql-client-core-8.0 is already the newest version (8.0.39-0ubuntu0.24.04.1).
0 upgraded, 0 newly installed, 0 to remove and 2 not upgraded.
ubuntu@ip-172-31-35-72:~$
```

3. Running the App:

- **Command Used:** `sudo node server.js` or `sudo npm start`

```
ubuntu@ip-172-31-35-72: ~/Projects/EC2-RDS-S3-Ansible-Automation/Profile-App
ubuntu@ip-172-31-35-72:~/Projects/EC2-RDS-S3-Ansible-Automation/Profile-App$ npm start

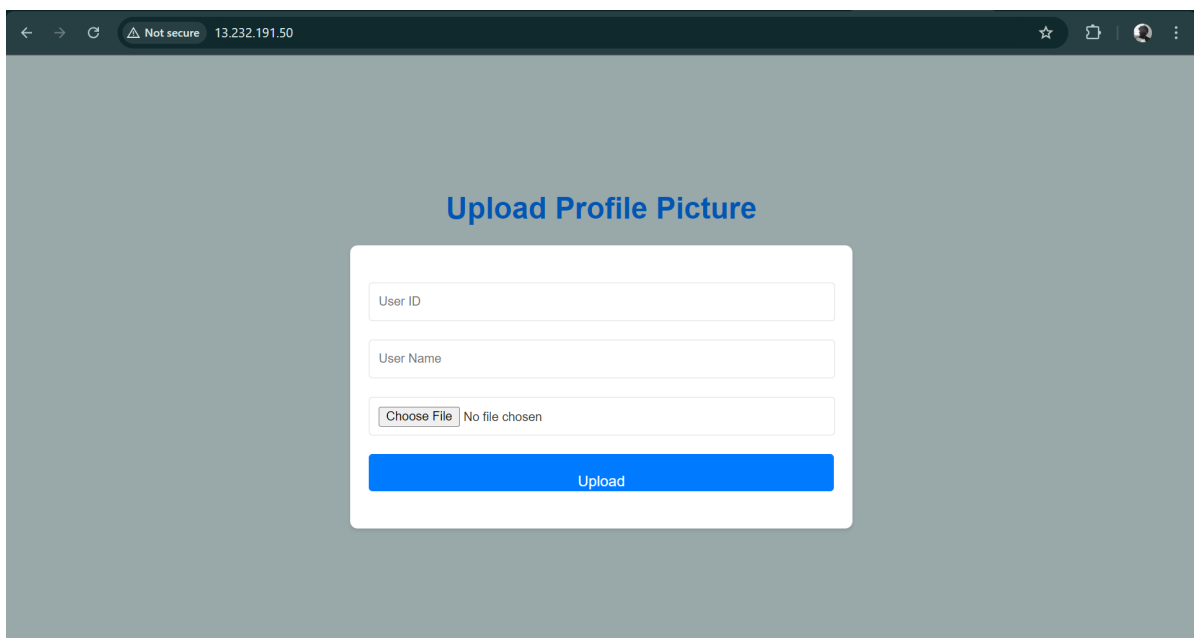
> profile-app@1.0.0 start
> node server.js

Server running on port 3000
(node:1642) NOTE: The AWS SDK for JavaScript (v2) will enter maintenance mode
on September 8, 2024 and reach end-of-support on September 8, 2025.

Please migrate your code to use AWS SDK for JavaScript (v3).
For more information, check blog post at https://a.co/cUPnyil
(Use `node --trace-warnings ...` to show where the warning was created)
```

if like to use npm run npm install first

- **Description:** Executed the application on the master instance to verify that it was operational.
- **Screenshot:**



The application running on the master instance with successful execution.

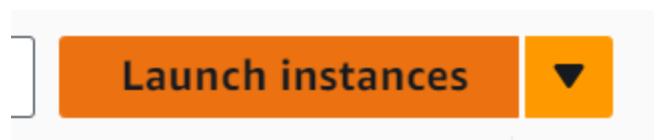
```
mysql> SELECT * FROM users;
+----+-----+-----+
| id  | name          | profile_picture_url |
+----+-----+-----+
| 92929292 | justsample@gmail.com | https://bucket-profile-app.s3.ap-south-1.amazonaws.com/profiles/1343c47440f09338118d4fd074477687.JPG |
| 111122333 | justsample@gmail.com | https://bucket-profile-app.s3.ap-south-1.amazonaws.com/profiles/33d49ca8542df162c194f89606211ab4.JPG |
+----+-----+-----+
```

5. Configuration of New Target Instance

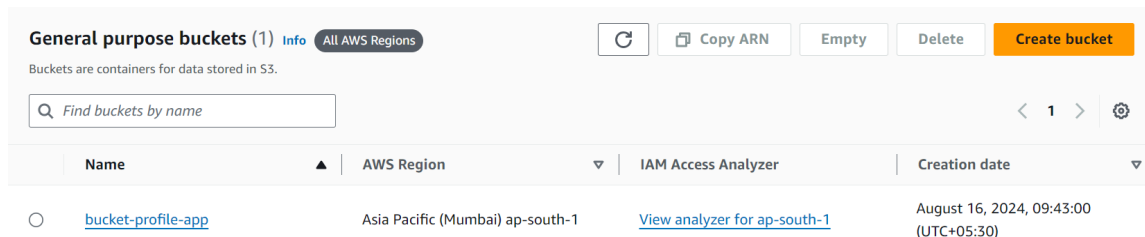
Setup of New Target Instance and RDS

1. Creating and Configuring New Instances:

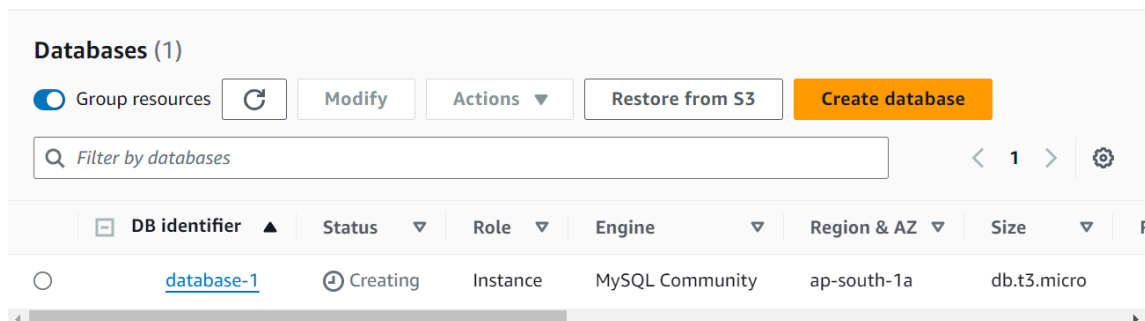
- **Description:** Provisioned a new EC2 instance and an additional RDS instance for scaling the deployment.
- **Steps Taken:**
 - Created a new target EC2 instance using AWS Management Console.
 - Configured the new RDS instance to handle database operations in target instance.
 - Set up security groups and access controls for proper communication between instances.
- **Screenshots:**



Launch target instance



Bucket Created.



database created

Inbound rules (3)							
<input type="text" value="Search"/>				< 1 >		⚙	
<input type="checkbox"/>	Name	Security group rule...	IP version	Type	Protocol	Port range	
<input type="checkbox"/>	-	sgr-072b718cf34f92efc	IPv4	HTTP	TCP	80	
<input type="checkbox"/>	-	sgr-0d384c357ddb64...	IPv4	Custom TCP	TCP	3000	
<input type="checkbox"/>	-	sgr-0f8bdb915a3b7e11e	IPv4	SSH	TCP	22	

security groups added, while connecting to RDS, SQL port will automatically added to outbound rules if not manually add it.

2. Adding Public Keys:

- **Description:** Updated authorized keys to enable secure communication between the master instance and the new target instance.
- **Steps Taken:**
 - Added the master instance's public key to the new instance's `authorized_keys` file.
- **Screenshots:** Configuration of SSH keys for secure instance communication.

```
ubuntu@ip-172-31-35-72:~$ MasterInstance^C
ubuntu@ip-172-31-35-72:~$ cat .ssh/id_rsa.pub
```

copy RSA public key from master instance

```
ubuntu@ip-172-31-36-246:~$ target instance^C
ubuntu@ip-172-31-36-246:~$ nano .ssh/authorized_keys
```

paste on authorised keys folder on target

Ansible Configuration

1. Structure of the Ansible folder


```

ubuntu@ip-172-31-35-72:~/Projects/EC2-RDS-S3-Ansible-Automation/Profile-App/ansible$ tree
.
├── ansible.cfg
├── inventory
│   └── hosts.yml
├── playbooks
│   └── site.yml
└── roles
    ├── common
    │   └── tasks
    │       └── main.yml
    ├── deploy
    │   └── tasks
    │       └── main.yml
    ├── env
    │   └── tasks
    │       └── main.yml
    ├── nginx
    │   ├── tasks
    │   │   └── main.yml
    │   └── templates
    │       └── nginx_profile_app.j2
    ├── pm2
    │   └── tasks
    │       └── main.yml
    └── sql
        └── tasks
            └── main.yml
17 directories, 10 files

```

2. Ansible Installation:

- **Description:** Installed Ansible on the master instance to automate the deployment process.
- **Steps Taken:**
 - Used package managers (e.g., `apt-get`) to install Ansible.
 - Verified installation and version to ensure compatibility.
- **Screenshot:**

```

ubuntu@ip-172-31-35-72:~$ sudo apt install ansible
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ansible is already the newest version (9.2.0+dfsg-0ubuntu5).
0 upgraded, 0 newly installed, 0 to remove and 2 not upgraded.
ubuntu@ip-172-31-35-72:~$

```

Installation process of Ansible on the master instance.

3. Configuration Files:

- **ansible.cfg:**

- **Description:** Configured Ansible settings, including roles and inventory paths.
- **Screenshot:**

```
ubuntu@ip-172-31-35-72:~/Projects/EC2-RDS-S3-Ansible-Automation/Profile-App/ansible$ cat ansible.cfg
[defaults]
inventory = inventory/hosts.yaml
roles_path = roles
```

Configuration settings in `ansible.cfg`.

- **Inventory File:**

- **Description:** Defined target instances and their details for Ansible to manage.
- **Screenshot:**

```
7 directories, 10 files
ubuntu@ip-172-31-35-72:~/Projects/EC2-RDS-S3-Ansible-Automation/Profile-App/ansible$ cat inventory/hosts.yml
all:
  hosts:
    13.232.191.50:
      ansible user: ubuntu
      ansible_ssh_private_key_file: /home/ubuntu/.ssh/id_rsa
ubuntu@ip-172-31-35-72:~/Projects/EC2-RDS-S3-Ansible-Automation/Profile-App/ansible$
```

Inventory file listing target instances and their connection details.

- **Roles Overview:**

- **Role 1: common**

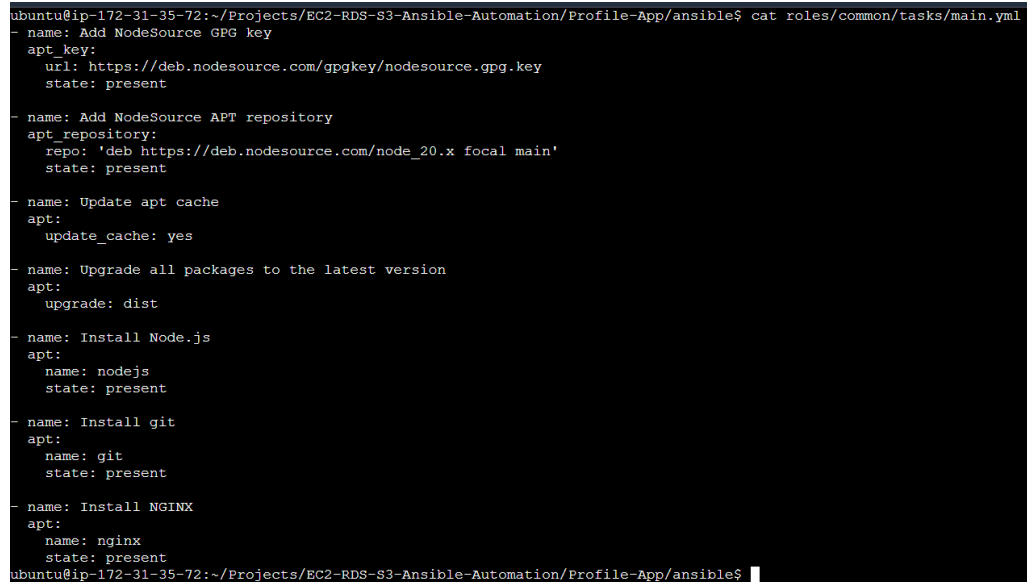
- **Description:**

The Common role is designed to set up the foundational environment required for the application to run. It begins by adding the NodeSource GPG key, which is essential for verifying the authenticity of the Node.js packages that will be installed. This is followed by adding the NodeSource APT repository to the server's package manager. This repository provides access to the latest stable version of Node.js, ensuring that the application will run on a supported version. After adding the repository, the server's package list is updated to make sure it has the latest information on available packages. The role then upgrades all existing packages on the server, ensuring everything is up-to-date and compatible with the application. Finally, the role installs Node.js, Git, and NGINX. Node.js is essential for running the application, Git allows for version control and easy deployment, and NGINX will be used as the reverse proxy server to manage incoming web traffic.

Note: If you're using a different Linux distribution or an older version of Ubuntu, you may need to update the repository URL and package names to match your

system's requirements.

- **Screenshot:**



```
ubuntu@ip-172-31-35-72:~/Projects/EC2-RDS-S3-Ansible-Automation/Profile-App/ansible$ cat roles/common/tasks/main.yml
- name: Add NodeSource GPG key
  apt_key:
    url: https://deb.nodesource.com/gpgkey/nodesource.gpg.key
    state: present

- name: Add NodeSource APT repository
  apt_repository:
    repo: 'deb https://deb.nodesource.com/node_20.x focal main'
    state: present

- name: Update apt cache
  apt:
    update_cache: yes

- name: Upgrade all packages to the latest version
  apt:
    upgrade: dist

- name: Install Node.js
  apt:
    name: nodejs
    state: present

- name: Install git
  apt:
    name: git
    state: present

- name: Install NGINX
  apt:
    name: nginx
    state: present
ubuntu@ip-172-31-35-72:~/Projects/EC2-RDS-S3-Ansible-Automation/Profile-App/ansible$
```

- **Role 2: deploy**

- **Description:**

The Deploy role is responsible for getting the application code onto the server and preparing it to run. The process starts with cloning the application's GitHub repository to the server. This allows the server to pull the latest version of the codebase directly from GitHub, ensuring that the most current version of the app is deployed. After the repository is cloned, the role installs the necessary npm dependencies by running `npm install` in the project directory. This command downloads and installs all the required packages listed in the `package.json` file. Additionally, the role explicitly installs the `dotenv` package, which is used to manage environment variables within the application. This is crucial for securely handling sensitive information like database credentials and API keys. Finally, the role installs other essential Node.js dependencies needed by the application, such as `express`, `mysql2`, `aws-sdk`, and `multer`, which provide the necessary functionality for server-side logic, database interaction, AWS integration, and file handling.

Note: If you're deploying a different application, be sure to update the Git repository URL and modify the list of dependencies in the `npm install` commands to match

your project's requirements.

- **Screenshot:** Tasks defined in the `deploy` role.

```
ubuntu@ip-172-31-35-72:~/Projects/EC2-RDS-S3-Ansible-Automation/Profile-App/ansible$ cat roles/deploy/tasks/main.yml
---
- name: Clone repository
  git:
    repo: 'https://ghp_gea2834ptWroRZoDCIxohSun7M5nhK2z0JUA@github.com/leju-b/Profile-App.git'
    dest: /home/ubuntu/Profile-App
    version: main
    accept_hostkey: yes
    update: yes

- name: Install npm dependencies
  command: npm install
  args:
    chdir: /home/ubuntu/Profile-App

- name: Install dotenv package
  command: npm install dotenv
  args:
    chdir: /home/ubuntu/Profile-App

- name: Install Node.js dependencies
  command: npm install express mysql2 aws-sdk multer
  args:
    chdir: /home/ubuntu/Profile-App
ubuntu@ip-172-31-35-72:~/Projects/EC2-RDS-S3-Ansible-Automation/Profile-App/ansible$
```

change the repo link into your link, create and add the access token in the link for authorisation.

- **Role 3: env**

- **Description:**

The Env role is focused on securely configuring the application's environment variables. It creates a `.env` file in the application directory, which stores sensitive information like database credentials, AWS access keys, and the S3 bucket name. This file is crucial for the application to interact securely with external services such as AWS RDS and S3. The `.env` file is created with restricted permissions, ensuring that only the necessary users have access to this sensitive information. The owner and group of the file are set to the user running the application, and the file permissions are set to `0600` to prevent unauthorized access.

Note: When replicating this role, it's important to edit the credentials in the `.env` file to match your own RDS, S3, and AWS configurations.

- **Screenshot:** Tasks defined in the `env` role.*

```

ubuntu@ip-172-31-35-72:~/Projects/EC2-RDS-S3-Ansible-Automation/Profile-App/ansible$ cat roles/env/tasks/main.yml
---
- name: Create .env file
  copy:
    dest: /home/ubuntu/Profile-App/.env
    content: |
      DB_HOST=database-1.c3qw2oqme9gi.ap-south-1.rds.amazonaws.com
      DB_USER=admin
      DB_PASSWORD=passRDS02
      DB_NAME=database-1
      S3_BUCKET=bucket-profile-app
      AWS_ACCESS_KEY_ID=AKIA6ODU4Y7HEPP4BDKF
      AWS_SECRET_ACCESS_KEY=WFDdF+D3F1CuOFGUr2o8wjlbaqBNZ68nV3kJWig
      AWS_REGION=ap-south-1
    mode: '0600'
    owner: ubuntu
    group: ubuntu

```

Edit this file, add your credentials. Best practice is to add all the credentials in a env file within ansible, but still not for production purposes.

◦ **Role 4: nginx**

▪ **Description:**

The NGINX role sets up and configures the NGINX web server to act as a reverse proxy for the Node.js application. It starts by ensuring that NGINX is installed on the server, then removes the default NGINX configuration to prevent conflicts. The role then creates a new NGINX configuration file specifically for the Profile App. This configuration file is templated using Jinja2, which allows for dynamic insertion of variables such as the server's IP address. The NGINX configuration is set up to listen on port 80 and proxy all incoming requests to the Node.js application running on port 3000. After creating the configuration file, the role enables it by creating a symbolic link in the `sites-enabled` directory, which tells NGINX to use this configuration. Finally, the role restarts the NGINX service to apply the changes and ensure that the server is ready to handle incoming traffic.

Note: If you're using a different domain or IP address, you'll need to edit the NGINX configuration template to match your setup. Also, make sure that the NGINX configuration file accurately reflects the correct paths and settings for your specific application.

▪ **Screenshot:** Tasks and template defined in the `nginx` role.

```

ubuntu@ip-172-31-35-72:~/Projects/EC2-RDS-S3-Ansible-Automation/Profile-App/ansible$ cat roles/nginx/tasks/main.yml
---
- name: Install NGINX
  apt:
    name: nginx
    state: present
    update_cache: yes

- name: Remove default NGINX configuration
  file:
    path: /etc/nginx/sites-enabled/default
    state: absent

- name: Create NGINX configuration file for the Profile App
  template:
    src: nginx_profile_app.j2
    dest: /etc/nginx/sites-available/profile_app
    mode: '0644'

- name: Enable the Profile App configuration
  file:
    src: /etc/nginx/sites-available/profile_app
    dest: /etc/nginx/sites-enabled/profile_app
    state: link

- name: Restart NGINX
  service:
    name: nginx
    state: restarted

```

```

ubuntu@ip-172-31-35-72:~/Projects/EC2-RDS-S3-Ansible-Automation/Profile-App/ansible$ cat roles/nginx/templates/nginx_profile_app.j2
server {
    listen 80;
    server_name 13.200.254.63;

    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;
    }
}

```

◦ Role 5: sql

▪ Description:

The SQL role focuses on preparing the database environment on AWS RDS to store user information for the application. It begins by ensuring that the necessary Python MySQL client libraries (`pymysql` and `mysqldb`) are installed, which are required for Ansible to interact with the MySQL database on RDS. The role then checks if the target database exists on the RDS instance and creates it if it does not. Following this, the role creates the `users` table within the database if it doesn't already exist. This table is structured to store user data, including an ID, name, and the URL of the user's profile picture. By ensuring the database and table are in place, this role prepares the environment for the application to start saving user data securely in the RDS.

Note: When replicating this project, you'll need to update the database credentials and table structure in the SQL role to match your own database schema and security requirements. Make sure the RDS instance is properly configured and accessible from your application server.

▪ Screenshot: Tasks defined in the `sql` role.

```

ubuntu@ip-172-31-35-72:~/Projects/EC2-RDS-S3-Ansible-Automation/Profile-App/ansible$ cat roles/sql/tasks/main.yml
- name: Ensure Python MySQL client is installed (for Python 3)
  ansible.builtin.apt:
    name:
      - python3-pymysql
      - python3-mysqldb
    state: present
    become: true

- name: Ensure the database exists
  community.mysql.mysql_db:
    login_host: database-1.c3qw2oqme9gi.ap-south-1.rds.amazonaws.com
    login_user: admin
    login_password: passRDS02
    name: database-1
    state: present

- name: Create users table if it does not exist
  community.mysql.mysql_query:
    login_host: database-1.c3qw2oqme9gi.ap-south-1.rds.amazonaws.com
    login_user: admin
    login_password: passRDS02
    login_db: database-1
    query: |
      CREATE TABLE IF NOT EXISTS users (
        id INT PRIMARY KEY,
        name VARCHAR(100) NOT NULL,
        profile_picture_url VARCHAR(255) DEFAULT NULL
      );

```

Change the user details.

◦ Role 6: pm2

▪ Description:

The PM2 role ensures that the Node.js application is managed and run efficiently using PM2, a production process manager for Node.js applications. This role begins by installing PM2 globally on the server, allowing it to be used system-wide. PM2 is then used to start the application with a specific name (`myapp`), ensuring that the application is continuously monitored and can automatically restart in case of failures. The role also configures PM2 to start at system boot by generating and executing the necessary startup command for systemd, the system and service manager. This ensures that the application will automatically start whenever the server is rebooted. After PM2 is set up, the role saves the current PM2 process list, which includes the application, so that PM2 can restore this process list upon reboot. The role concludes by reloading the NGINX service to ensure that it is correctly configured to work with the PM2-managed application.

Note: If you're deploying a different application, you'll need to modify the `pm2 start` command to point to the correct entry point (e.g., `server.js`) for your application. Also, make sure the PM2 and NGINX settings align with the specific requirements of your environment.

▪ Screenshot: Tasks defined in the `pm2` role.

```

ubuntu@ip-172-31-35-72:~/Projects/EC2-RDS-S3-Ansible-Automation/Profile-App/ansible$ cat roles/pm2/tasks/main.yml
---
- name: Install PM2 globally
  npm:
    name: pm2
    state: present
    global: yes

- name: Install application dependencies
  npm:
    path: /home/ubuntu/Profile-App
    state: present

- name: Start the application with PM2
  shell: pm2 start server.js -f --name myapp
  args:
    chdir: /home/ubuntu/Profile-App
    register: pm2_start_result

- name: Get the PM2 startup command for systemd
  command: pm2 startup systemd
  register: pm2_startup_command
  changed_when: false

- name: Execute the PM2 startup command
  shell: |
    sudo env PATH=$PATH:/usr/bin:/usr/lib/node_modules/pm2/bin/pm2 startup systemd -u ubuntu --hp /home/ubuntu
  become: yes

- name: Save PM2 process list
  shell: pm2 save
  when: pm2_start_result.changed

- name: Reload NGINX to apply changes
  service:
    name: nginx
    state: reloaded
    become: yes
    when: pm2_start_result.changed

- name: Ensure NGINX is running
  service:
    name: nginx
    state: started
    enabled: yes
    become: yes
ubuntu@ip-172-31-35-72:~/Projects/EC2-RDS-S3-Ansible-Automation/Profile-App/ansible$

```

6. Running Ansible Playbooks

Sites.yml file for executing the roles defined in the playbooks

```

ubuntu@ip-172-31-35-72:~/Projects/EC2-RDS-S3-Ansible-Automation/Profile-App/ansible$ cat playbooks/site.yml
---
- hosts: all
  become: yes
  roles:
    - common
    - deploy
    - env
    - nginx
    - sql
    - pm2

```

Execution

1. Commands Used:

- **Description:** Executed Ansible playbooks to automate the deployment process.
- **Commands:**
 - `ansible-playbook -i /inventory /playbook/sites.yml` - To run the playbook

- **Screenshot:** Execution of Ansible playbooks for deployment.

```

ubuntu@ip-172-31-35-72:~/Projects/EC2-RDS-S3-Ansible-Automation/Profile-App/ansible$ ansible-playbook -i inventory/ playbooks/site.yml
PLAY [all] *****
TASK [Gathering Facts] *****
ok: [13.232.191.50]

TASK [common : Add NodeSource GPG key] *****
ok: [13.232.191.50]

TASK [common : Add NodeSource APT repository] *****
ok: [13.232.191.50]

TASK [common : Update apt cache] *****
changed: [13.232.191.50]

TASK [common : Upgrade all packages to the latest version] *****
ok: [13.232.191.50]

TASK [common : Install Node.js] *****
ok: [13.232.191.50]

TASK [common : Install git] *****
ok: [13.232.191.50]

TASK [common : Install NGINX] *****
ok: [13.232.191.50]

TASK [deploy : Clone repository] *****
changed: [13.232.191.50]

TASK [deploy : Install npm dependencies] *****
changed: [13.232.191.50]

TASK [deploy : Install dotenv package] *****
changed: [13.232.191.50]

TASK [deploy : Install Node.js dependencies] *****
changed: [13.232.191.50]

TASK [env : Create .env file] *****
changed: [13.232.191.50]

TASK [nginx : Install NGINX] *****
ok: [13.232.191.50]

TASK [nginx : Remove default NGINX configuration] *****
ok: [13.232.191.50]

TASK [nginx : Create NGINX configuration file for the Profile App] *****
ok: [13.232.191.50]

TASK [nginx : Enable the Profile App configuration] *****
ok: [13.232.191.50]

TASK [nginx : Restart NGINX] *****
changed: [13.232.191.50]

TASK [sql : Ensure Python MySQL client is installed (for Python 3)] *****
ok: [13.232.191.50]

TASK [sql : Ensure the database exists] *****
ok: [13.232.191.50]

TASK [sql : Create users table if it does not exist] *****
changed: [13.232.191.50]

TASK [pm2 : Install PM2 globally] *****
ok: [13.232.191.50]

TASK [pm2 : Install application dependencies] *****
ok: [13.232.191.50]

TASK [pm2 : Start the application with PM2] *****
changed: [13.232.191.50]

TASK [pm2 : Get the PM2 startup command for systemd] *****
ok: [13.232.191.50]

TASK [pm2 : Execute the PM2 startup command] *****
changed: [13.232.191.50]

TASK [pm2 : Save PM2 process list] *****
changed: [13.232.191.50]

TASK [pm2 : Reload NGINX to apply changes] *****
changed: [13.232.191.50]

TASK [pm2 : Ensure NGINX is running] *****
ok: [13.232.191.50]

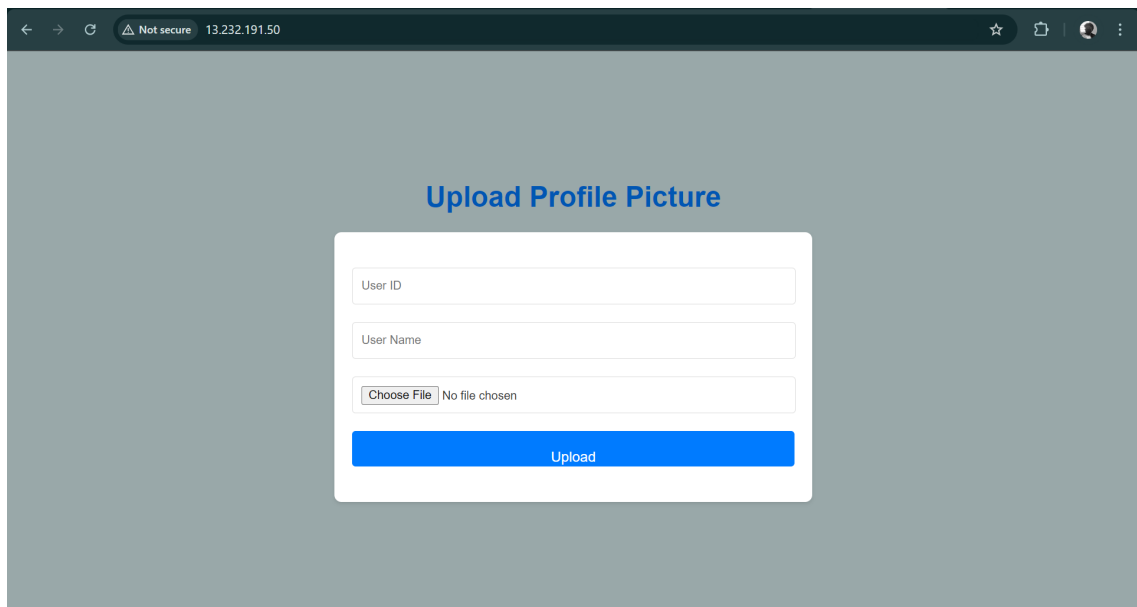
PLAY RECAP *****
13.232.191.50 : ok=29 changed=12 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

ubuntu@ip-172-31-35-72:~/Projects/EC2-RDS-S3-Ansible-Automation/Profile-App/ansible$

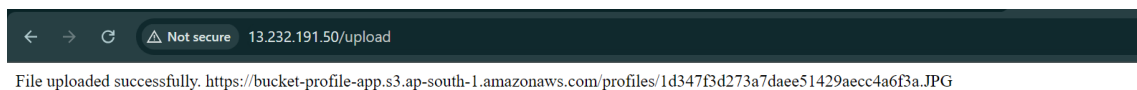
```

2. App Deployment:

- **Description:** Verified the successful deployment of the application on the target instance.
- **Steps Taken:**
 - Checked application status, ensured it was running correctly, and verified connectivity to AWS RDS and S3



Successfully Running



Successfully uploading files to S3 and RDS database

7. Conclusion

Summary

This project involved the development, environment setup, and deployment of a Node.js application with AWS RDS and S3 integration. The application was successfully deployed using Ansible for automation, with the entire setup being tested and verified on target instances and

master instance. The application now operates as intended, with data being stored in AWS RDS and files in AWS S3.

- **Screenshots:** Application in its final state, including successful file uploads and database entries.*

```
ubuntu@ip-172-31-36-246:~/Profile-App$ mysql -h database-1.c3qw2oqme9gi.ap-south-1.rds.amazonaws.com -u admin -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 80
Server version: 8.0.35 Source distribution

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

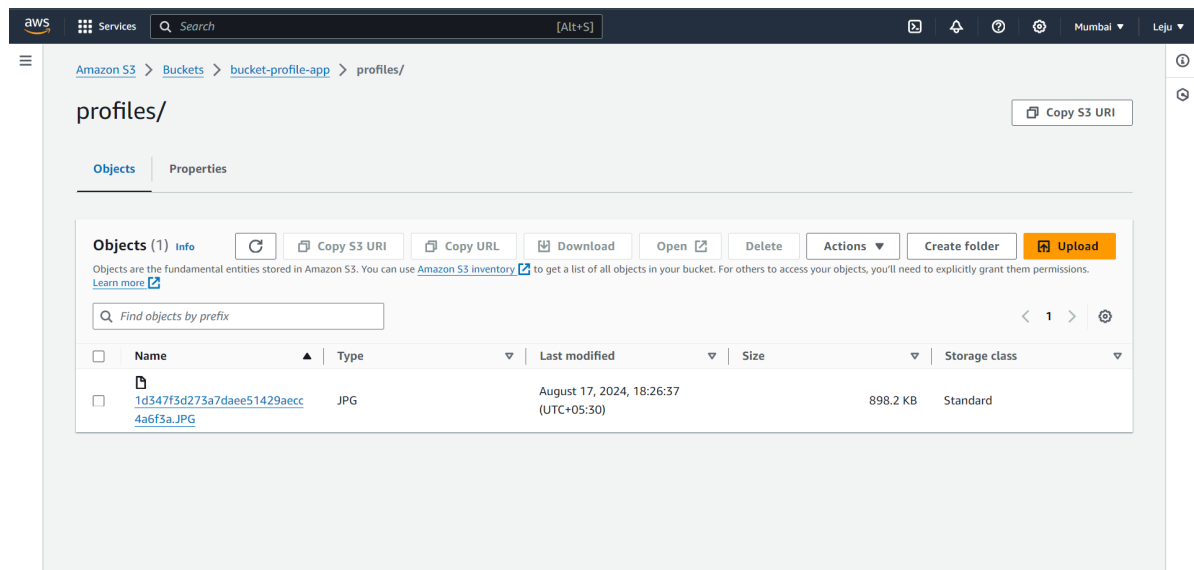
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> USE database-1;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SELECT * FROM users;
+-----+-----+-----+
| id    | name          | profile_picture_url |
+-----+-----+-----+
| 92929292 | justsample@gmail.com | https://bucket-profile-app.s3.ap-south-1.amazonaws.com/profiles/87db0bf736709a21ef3fa7cbca34c72b.JPG |
| 92929294 | justsample@gmail.com | https://bucket-profile-app.s3.ap-south-1.amazonaws.com/profiles/1d347f3d273a7daee51429aecc4a6f3a.JPG |
+-----+-----+-----+
2 rows in set (0.00 sec)

mysql>
```

Data after successfull running of the app.



files uploaded by app on s3.

Appendix

- **Troubleshooting Tips:**
 - ▼ Common issues with RDS connections and solutions.

- Mostly issues will be at security group configurations and credential integration.
- Make sure to connect your RDS with the target instance.
- ▼ Debugging tips for AWS S3 integration issues.
 - Create access keys, while env file editing make sure about spellings.
- ▼ Steps to verify Ansible configurations and troubleshoot playbook errors.
 - Use -v : verboros for detailed error log
 - change nodejs installation versions if wanted

Feel free to ping me if you have any questions, doubts or stuck at any point. find my contact details @<https://leju-b.github.io/my-Portfolio/>