MERN Stack AWS Deployment

can you access the below link and make a document on it

https://medium.com/@jijinvj/how-to-deploy-a-mern-stack-app-on-aws-ec2-with-ssl-nginx-and-subdomain-c597d3075d83

Certainly! Here's a comprehensive guide based on the Medium article by Jijin VJ on deploying a MERN stack application on AWS EC2 with SSL, NGINX, and a subdomain:LinkedIn+1

🚀 Deploying a MERN Stack App on AWS EC2 with SSL, NGINX, and Subdomain

Step 1: Launch an EC2 Instance

1. **Create an AWS Account**:
    - Sign up at AWS.
2. **Launch an EC2 Instance**:
    - Navigate to the EC2 Dashboard.
    - Click on “Launch Instance”.
    - Choose **Ubuntu 20.04 LTS** as the OS (eligible for the free tier).
    - Name your instance appropriately.Medium+1
3. **Create a Key Pair**:
    - Click on “Create New Key Pair”.
    - Select **RSA** as the key pair type and **.pem** as the file format.
    - Download and securely store the `` `.pem` `` file; it's essential for SSH access. Medium+2
4. **Configure Network Settings**:
    - Allow **HTTP (port 80)** and **HTTPS (port 443)** traffic.
5. **Adjust Storage**:
    - Allocate up to 30GB of storage as needed.Medium
6. **Launch the Instance**:
    - Click on “Launch Instance” and monitor its status.Medium

Step 2: Allocate and Associate an Elastic IP

1. In the EC2 Dashboard, navigate to **Elastic IPs**.

2. Click on “Allocate new address” and confirm.
3. Select the newly allocated IP and click on “Associate this Elastic IP address”.
4. Choose your EC2 instance and associate the IP.Medium

Step 3: Configure Security Group Inbound Rules

1. Select your EC2 instance and navigate to the **Security** tab.
2. Click on the associated security group.
3. Edit inbound rules to allow:
    - **SSH (port 22)**
    - **HTTP (port 80)**
    - **HTTPS (port 443)**
    - **Custom ports** used by your Node.js and React applications (e.g., 9000).Medium

Step 4: Connect to the EC2 Instance via SSH

1. Open your terminal.
2. Navigate to the directory containing your `` `.pem` `` file.
3. Modify the permissions of the `` `.pem` `` file:

    ```bash
    chmod 400 your-key.pem
    ```

4. Connect to your instance:

    ```bash
    ssh -i "your-key.pem" ubuntu@your-elastic-ip
    ```

MediumMedium

Step 5: Install Necessary Software on Ubuntu

1. **Update Package Lists**:

    ```bash
    sudo apt update
    ```

2. **Install Node.js and npm**:

    ```bash
    sudo apt install nodejs npm
    ```

Medium

3. **Verify Installations**:

```
bash

nodejs -v
npm -v
```

YouTube

4. **Install NVM (Node Version Manager)** (optional, for managing Node.js versions):

```
bash

curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.35.3/install.sh | bash
```

5. **Install PM2 (Process Manager)**:

```
bash

sudo npm install -g pm2
```

DevOps Stack Exchange+1

6. **Install NGINX**:

```
bash

sudo apt install nginx
```

LinkedIn+1

7. **Install UFW (Uncomplicated Firewall)**:

```
bash

sudo apt install ufw
```

Medium

Step 6: Configure Domain and Subdomain

1. **Purchase a Domain**:
   - Use providers like **Hostinger** or **GoDaddy**.Medium
2. **Set Up DNS Records**:
   - In your domain provider's dashboard, navigate to DNS settings.
   - Add the following **A records**:
     - **Type**: A
     - **Name**: @
     - **Points to**: Your Elastic IP

        - **TTL**: Default
        - **Type**: A
        - **Name**: api
        - **Points to**: Your Elastic IP
        - **TTL**: Default
    - This setup points `yourdomain.com` to your frontend and `api.yourdomain.com` to your backend.

Step 7: Configure NGINX

1. **Navigate to NGINX Sites-Available Directory**:

```bash
cd /etc/nginx/sites-available
```

2. **Create Configuration Files**:
    - For frontend (`yourdomain.com`):

    ```bash
    sudo nano yourdomain.com
    ```

    Add the following configuration:

    ```nginx
    server {
        listen 80;
        server_name yourdomain.com www.yourdomain.com;

        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;
        }
    }
    ```

    - For backend (`api.yourdomain.com`):

    ```bash
    sudo nano api.yourdomain.com
    ```

    Add the following configuration:

    ```nginx
    server {
        listen 80;
        server_name api.yourdomain.com;
    ```

```
    location / {
        proxy_pass http://localhost:9000;
        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;
    }
}
```

3. **Enable the Configuration Files**:

```bash
sudo ln -s /etc/nginx/sites-available/yourdomain.com /etc/nginx/sites-enabled/
sudo ln -s /etc/nginx/sites-available/api.yourdomain.com /etc/nginx/sites-enabled/
```

DevOps Stack Exchange

4. **Test NGINX Configuration**:

```bash
sudo nginx -t
```

5. **Restart NGINX**:

```bash
sudo systemctl restart nginx
```

Step 8: Install SSL Certificates with Certbot

1. **Install Certbot**:

```bash
sudo apt install certbot python3-certbot-nginx
```

YouTube

2. **Obtain and Install SSL Certificates**:

```bash
sudo certbot --nginx -d yourdomain.com -d www.yourdomain.com
sudo certbot --nginx -d api.yourdomain.com
```

YouTube

3. **Verify SSL Installation**:
    - Visit `` `https://yourdomain.com` `` and `` `https://api.yourdomain.com` `` to ensure SSL is active.

4. **Set Up Automatic Certificate Renewal**:

```bash
sudo certbot renew --dry-run
```

By following these steps, you will have successfully deployed your MERN stack application on AWS EC2 with SSL, NGINX, and a subdomain.

For a more detailed walkthrough, you can refer to the original Medium article by Jijin VJ: Medium LinkedIn+1

Sources