To tailor the **Deepseek-RAG-App** repository setup to your specific requirements and conditions, I will provide a **step-by-step plan** for deploying a locally hosted LLM chatbot on AWS Zurich servers. This plan will ensure the system can process 80-page PDF reports, work for one user, and meet the budget and scalability requirements. The setup will use **Cline** and **VSCode** for ease of deployment.

Step 1: Launch an EC2 Instance in AWS Zurich

1.1. Login to AWS and Navigate to EC2

- Go to the AWS Management Console and select the **Zurich (eu-central-1)** region.
- Navigate to the **EC2 Dashboard**.

1.2. Launch an EC2 Instance

- Click **Launch Instance** and configure the following:
  - **Name:** `Deepseek-RAG-App`
  - **AMI:** Select **Ubuntu 22.04 LTS** (64-bit).
  - **Instance Type:** Use a **G4dn.xlarge** instance (GPU-enabled) to handle the LLM efficiently.
    - Or G6.xlarge
  - **Key Pair:** Create or select an existing key pair for SSH access.
  - **Storage:** Allocate at least **64 GB** of storage (SSD).
  - **Security Group:** Allow the following inbound rules:
    - **SSH (port 22):** For remote access.
    - **HTTP (port 80):** For the reverse proxy.
    - **HTTPS (port 443):** For secure communication.
    - **Custom TCP (port 8501):** For the Streamlit app.
    - **Custom TCP (port 11434):** For the Ollama API.

1.3. Launch the Instance

- Review the configuration and launch the instance.
- Note the **Public IP Address** of the instance for later use.

Step 2: Connect to the EC2 Instance

2.1. SSH into the Instance

Open a terminal (or use Cline) and connect to the instance:

bash

Copy Code

```
ssh -i <your-key> ubuntu@<your-ec-public-ip>
```

2.2. Update the System

Run the following commands to update the system and install essential abilities:

bash
Copy Code

```
sudo apt update && sudo apt upgrade -y
sudo apt install -y python3 python3-pip git curl nginx
```

Step 3: Set Up the Reverse Proxy with Nginx

3.1. Install and Configure Nginx

- Install Nginx (already done in Step 2.2).

Configure Nginx to act as a reverse proxy for the Streamlit app and Ollama API:

bash
Copy Code

```
sudo nano /etc/nginx/sites-available/deepseek
```

Add the following configuration:

nginx
Copy Code

```
server {
    listen 80;
    server_name <your-ec-public-ip>;

    location / {
        proxy_pass http://localhost:8501; # Streamlit app
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }

    location /api/ {
        proxy_pass http://localhost:11434; # Ollama API
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }
}
```

Enable the configuration:

bash
Copy Code

```
sudo ln -s /etc/nginx/sites-available/deepseek /etc/nginx/sites-enabled/
sudo nginx -t
sudo systemctl restart nginx
```

3.2. Secure Nginx with HTTPS

Use **Certbot** to obtain a free SSL certificate:

bash
Copy Code

```
sudo apt install -y certbot python3-certbot-nginx
sudo certbot --nginx -d <your-ec-public-ip>
```

Step 4: Set Up the Deepseek-RAG-App Repository

4.1. Clone the Repository

Clone the repository into the EC2 instance:

bash
Copy Code

```
git clone https://github.com/Spidy20/Deepseek-RAG-App.git
cd Deepseek-RAG-App
```

4.2. Set Up a Python Virtual Environment

Create and activate a virtual environment:

bash
Copy Code

```
python3 -m venv venv
source venv/bin/activate
```

4.3. Install Dependencies

Install the required Python packages:

bash
Copy Code

```
pip install -r requirements.txt
```

Step 5: Install and Configure Ollama for the LLM

5.1. Install Ollama

Download and install Ollama:

bash
Copy Code

```
curl -fsSL https://ollama.com/install.sh | sh
```

5.2. Download the DeepSeek Model

Download the **DeepSeek R1** model (8B variant for efficiency):

bash
Copy Code

```
ollama run deepseek-r1:8b
```

5.3. Start the Ollama API

Start the Ollama API server:
bash
Copy Code

```
ollama serve
```

5.4. Test the API

Test the API to ensure it is running:
bash
Copy Code

```
curl http://localhost:11434/api/chat -d '{
    "model": "deepseek-r1:8b",
    "messages": [{ "role": "user", "content": "Write python script for
hello world" }],
    "stream": false
}'
```

Step 6: Run the Streamlit Application

6.1. Start the Application

Run the Streamlit app:
bash
Copy Code

```
streamlit run app.py
```

6.2. Access the Application

Open a browser and navigate to:
http://<your-ec-public-ip>

Step 7: Test the Application

7.1. Upload a PDF

- Use the Streamlit interface to upload an 80-page PDF document.

7.2. Ask Questions

- Interact with the chatbot by asking questions about the uploaded document.

Step 8: Optimize for Cost and Scalability

8.1. Use Spot Instances

- To reduce costs, consider using AWS Spot Instances for the EC2 instance.

8.2. Monitor Resource Usage

- Use AWS CloudWatch to monitor CPU, memory, and GPU usage.

8.3. Plan for Scaling

- For future scalability:
    - Use an **Auto Scaling Group** to add more EC2 instances as needed.
    - Deploy the application in a **private VPC** for enhanced security.

Step 9: Set Up VSCode for Development

9.1. Connect to EC2 via VSCode

- Install the **Remote - SSH** extension in VSCode.
- Configure the SSH connection:
    - Open the Command Palette (`Ctrl+Shift+P`) and select **Remote-SSH: Connect to Host**.

Enter the SSH details:
bash
Copy Code
```
ubuntu@<your-ec-public-ip>
```

9.2. Edit and Debug the Application

- Open the `Deepseek-RAG-App` folder in VSCode.

Use the integrated terminal to run the app:
bash
Copy Code
```
streamlit run app.py
```

- Set breakpoints in `app.py` to debug the application.

Step 10: Shut Down When Not in Use

To avoid unnecessary costs, stop or terminate the EC2 instance when not in use:
bash
Copy Code
```
aws ec2 stop-instances --instance-ids <instance>
```
