Project Documentation

Overview

Our company is launching a new assets management system for our facility and site operations team across global sites. Each site/region has its own interface/application window. We have successfully deployed our first interface for the HQ Site. However, some internal features are still missing, such as a Mailing Server, Security Code Scanning, and Deployment Automation. We are expanding our solution across global sites and need to finalize these missing features on a single server due to budget constraints.

Technical Steps

1. Server Provisioning Using Terraform

1. **Install Git**
    - On Ubuntu, follow the instructions in this article.
2. **Install Terraform**
    - Follow the steps in this tutorial.
3. **Create Terraform Files for Server Provisioning**
    - Define your infrastructure in `.tf` files to create an EC2 instance.
4. **Install AWS CLI**
    - Install AWS CLI to register the AccessKey or manually enter it in `~/.aws/credentials`.
    - Follow the AWS documentation.
5. **Create and register the AWS AccessKey**
    - Generate and register an AWS AccessKey as per the AWS documentation.
6. **Create and Install the SSH Key Pair**
    - Generate and download the SSH Key Pair from the AWS console following this guide.
    - Change the permissions of the key file: `chmod 400 <your-key-file.pem>`

Notes:

- **Security Group Configuration:** Ensure the security group allows inbound traffic on necessary ports (22 for SSH, 80 for HTTP, 5000 for Apache, and 9000 for SonarQube).
- **Instance Type:** Use an instance type with at least 2GB of RAM (e.g., `t2.medium`) to efficiently run SonarQube. As explained in the docs
- **Key Pair Permissions:** Change permissions of the SSH key file to `chmod 400 <your-key-file.pem>` to ensure it is secure.

2. Server Configuration Management

1. **Install Ansible**
   - Use the package manager to install Ansible on your machine.
2. **Create Ansible Playbooks**
   - Create playbooks for Nginx, Apache, and Docker. Refer to the following guides:
     - Install Apache in Ubuntu using Ansible
     - Installing Nginx Web server on Ubuntu with Ansible
3. **Create Ansible Configuration File**
   - Create an `ansible.cfg` file to hold configuration settings.
     - Introduction to Ansible Configuration Files
4. **Create Inventory File**
   - Create an `inventory.ini` file to hold the hosts' data.
5. **Run the Playbooks**
   - Execute the playbooks using Ansible. If issues arise, consider the following steps:

- **Nginx and Apache both use port 80. Forward one of them to another port, e.g., Apache to port 5000.**
- **Ensure Python and the six module are installed:**

```
sudo apt-get update
sudo apt-get install python3-pip
pip3 install six
```

- **Ensure Ansible is installed or updated:**

```
pip install --upgrade ansible
```

3. Deployment Using Docker Containers

1. **Create "sonar.properties" File containing the following**

```
sonar.web.context=/sonarqube
```

2. **Create DockerFile to create of the SonarQube with some edit in sonar.properties:**

```
# Use the official SonarQube image as base
FROM sonarqube:latest

# Copy custom sonar.properties file to the appropriate location
COPY sonar.properties /opt/sonarqube/conf/sonar.properties
```

3. **Create Docker Compose File**
    - Define the Docker Compose file to start SonarQube and PostgreSQL containers. Refer to this guide.
4. **Create Ansible Playbook for Docker Compose**
    - Write an Ansible playbook to run the Docker Compose on the server. See this Stack Overflow post for reference.

4. Port-Forwarding Configuration Using Nginx

1. **Create Reverse Proxy Configuration**
    - Configure Nginx to forward requests:
        - `/apache` to port 5000
        - `/sonarqube` to port 9000
    - Refer to this Stack Overflow post for setting up SonarQube behind a reverse proxy.
2. **Create Ansible Playbook for Nginx Configuration**
    - Write an Ansible playbook to apply the reverse proxy configuration on the server.

5. CI/CD Automation

1. **Setup Terraform Cloud**
    - Follow this video and tutorial to store remote state in Terraform Cloud.
2. **Automate Terraform with GitHub Actions**
    - Use this tutorial to automate the Terraform workflow with GitHub Actions.