




## Documentation for Project







### Project Overview:

This project is divided into three main parts, each contributing to different aspects of the overall solution:

1. **Terraform Part:**  Provisioning Azure resources, including a Linux Virtual Machine (VM), networking resources, and associated components.
2. **Application and Docker Part:**  Deploying a Flask API application using Docker containers.
3. **Ansible Part:**  Automating configuration and management tasks using Ansible playbooks.

### Terraform Part:

**Explanation for linux-vm-main.tf File:** This Terraform code provisions an Azure Linux VM along with related networking resources. The code is written in HashiCorp Configuration Language (HCL) and utilizes the Azure provider for interacting with Azure resources. Here's a detailed breakdown of each code section:

1.  **Random Password Generation:** The `random_password` resource generates a random password for VM authentication.
2.  **Random VM Name Generation:** The `random_string` resource generates a unique VM name.
3.  **Network Security Group Creation:** Creates a network security group (NSG) to control traffic.
4.  **Subnet-NSG Association:** Associates NSG rules with a specific subnet.
5.  **Static Public IP Creation:** Generates a static public IP address.
6.  **Network Interface Creation:** Creates a network interface associated with the public IP and subnet.

7. 🐧 **Linux VM Creation:** Creates the Linux VM using specified configurations.
8. 📄 **Template File for Cloud-Init:** Uses a template file for VM initialization.

### ⚙️ **How to Run Terraform Part:**

1. 💻 Open Ubuntu WSL.
2. 🌐 Install Azure CLI:  

```
sudo apt-get update
```

```
sudo apt-get install ca-certificates curl apt-transport-https lsb-release gnupg
```

# Follow the provided commands to add the Microsoft repository



```
sudo apt-get update
```

```
sudo apt-get install azure-cli
```
3. 🏗️ Install Terraform:  




```
sudo apt-get install terraform
```
4. 🔑 Log in to your Azure account using **az login**.
5. 🛡️ Generate an SSH key: **ssh-keygen -t rsa -b 4096**.
6. 📁 Navigate to the Terraform project directory.
7. 🏁 Initialize Terraform: **terraform init**.
8. 📋 Plan the deployment: **terraform plan**.
9. 🚀 Apply the Terraform configuration: **terraform apply --auto-approve**.
10. ✅ You'll receive important outputs, such as the server IP.

## Application and Docker Part:

This part involves deploying a simple Flask API application using Docker containers. To run the Dockerized application:

1.  In Ubuntu terminal, navigate to the application directory.
2.  Install Podman:  




```
sudo apt-get update
```

```
sudo apt-get install -y podman
```
3.  Creating a Docker Hub Account and Repository:
  - a. Visit Docker Hub and sign up for an account.
  - b. Once logged in, create a new repository:
    - Click on "Create Repository."
    - Choose a name for your repository and set its visibility (public or private).
4.  Building and Tagging the Docker Image:
  - a. In the terminal, navigate to the directory containing your application code and Dockerfile.
  - b.  Building the Image:


```
sudo podman build -t your-dockerhub-username/repository-name:tag-name .
```

### Replacing Values:

Before proceeding, make sure to replace placeholders with actual values for a personalized experience:


1.  **Your Docker Hub Username:** Replace **your-dockerhub-username** with your actual Docker Hub username.
2.  **Repository Name:** Choose an appropriate name for your repository and replace **repository-name**.
3.  **Tag Name:** Select a descriptive tag name and replace **tag-name**.

#### 4. Building and Tagging the Docker Image:


- a. In the terminal, navigate to the directory containing your application code and Dockerfile.
- b.  Building the Image:  
`sudo podman build -t your-dockerhub-username/repository-name:tag-name .`

Replace your-dockerhub-username, repository-name, and tag-name with appropriate values.

#### 5. Pushing the Image to Docker Hub:


- a.  Logging in to Docker Hub:  
`sudo podman login docker.io -u your-dockerhub-username`

6. Enter your Docker Hub password when prompted.

- b.  Pushing the Image:  
`sudo podman push your-dockerhub-username/repository-name:tag-name`



#### 7. Running the Container:

`sudo podman run -d -p 8086:3000 your-dockerhub-username/repository-name:tag-name`

 Open your browser and visit: <http://localhost:8086>.



## **Ansible Part:**

To run Ansible playbooks for automation:

1.  In Ubuntu terminal, navigate to the project directory.
2.  Install Ansible:  



```
sudo apt-get update
```

```
sudo apt-get install -y ansible
```
3.  Edit the hosts file with your server IP.
4.  Run a specific playbook: `ansible-playbook -i hosts playbook_name.yml`.



## **After Running the Containers:**


Once you've finished running the containers, take the following steps:


1.  **Open Your Browser:** Launch your web browser.
2.  **Type the URL:** In the browser's address bar, enter the following URL:  
[http://<server\\_ip>:<port>](http://<server_ip>:<port>)

## **Replace IP and Port for Access:**

For seamless access and interaction with your deployed Flask API application, personalize the URL by substituting placeholders with actual values:

1.  **Replace <server\_ip>:** Substitute <server\_ip> with the real IP address of your server.
2.  **Replace <port>:** Update <port> with the specific port number configured for your application.

By making these changes, you'll create a URL tailored to your deployment setup, enabling effortless browsing and interaction with your Flask API application in your web browser. 

 **Note:** This documentation provides a comprehensive guide to each part of the project, including installation steps, commands, and explanations. Always refer to official documentation for the most up-to-date instructions and details.