**Project Name:** AWS Infrastructure Deployment for Python Flask App

This project automates the deployment of a Python Flask web application on Amazon Web Services (AWS) infrastructure using Terraform. It includes the creation of a Virtual Private Cloud (VPC), public subnet, route table with an internet gateway, and an EC2 instance configured to host the Flask app. The infrastructure setup ensures scalability, security, and high availability for the web application.

Key Features:

* Automated provisioning of AWS resources with Terraform.
* Configuration of networking components including VPC, subnet, and route table.
* Deployment of a secure EC2 instance for hosting the Python Flask app.
* Integration of AWS services for reliable and scalable application hosting.
* Infrastructure-as-Code (IaC) approach for version-controlled and repeatable deployments.

Technologies Used:

* AWS (Amazon Web Services)
* Terraform
* Python Flask

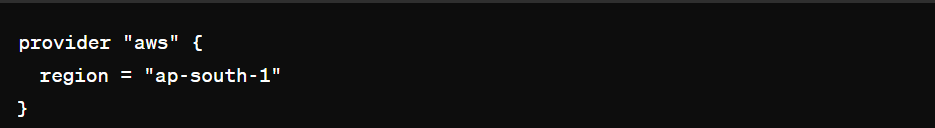
Usage:

* Clone the repository.
* Navigate to the project directory.
* Initialize Terraform (terraform init).
* Review and apply the Terraform configuration (terraform apply).
* Access the deployed Flask application via the EC2 instance public IP.

This project demonstrates best practices for deploying modern web applications on AWS infrastructure efficiently and securely.

**Procedure :**

1. Provider Configuration:



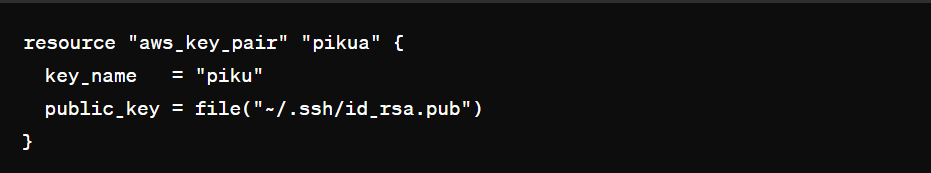
* This block configures Terraform to use the AWS provider for resources in the specified AWS region (**ap-south-1** in this case).

2. Variable Declaration:



* This block declares a variable named **cidr** with a default value of 10.0.0.0/16. This variable will be used later in resource definitions.

3. Key Pair Creation:



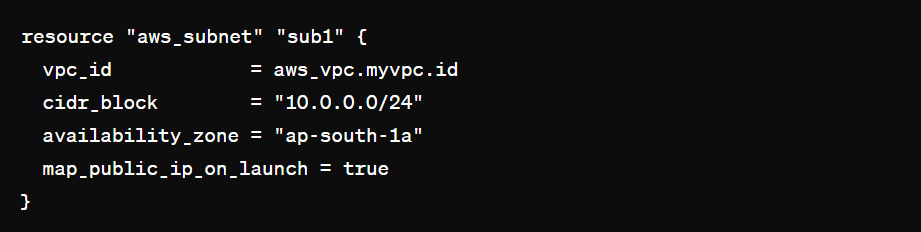
* This block creates an AWS key pair named piku using the public key from the (~/.ssh/id\_rsa.pub) file on your local machine
* You have to generate from your terminal using : ssh-keygen –t rsa

4. VPC Creation:



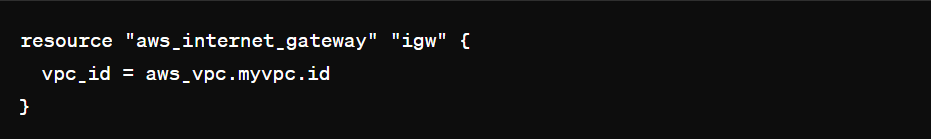
* This block creates an AWS Virtual Private Cloud (VPC) with the CIDR block specified by the **cidr** variable.

5. Subnet Creation:



* This block creates a subnet (sub1) within the VPC created earlier, with the specified CIDR block and availability zone. **map\_public\_ip\_on\_launch** is set to true to assign public IPs to instances launched in this subnet.

6. Internet Gateway Creation:



* This block creates an internet gateway and attaches it to the VPC to allow outbound internet traffic.

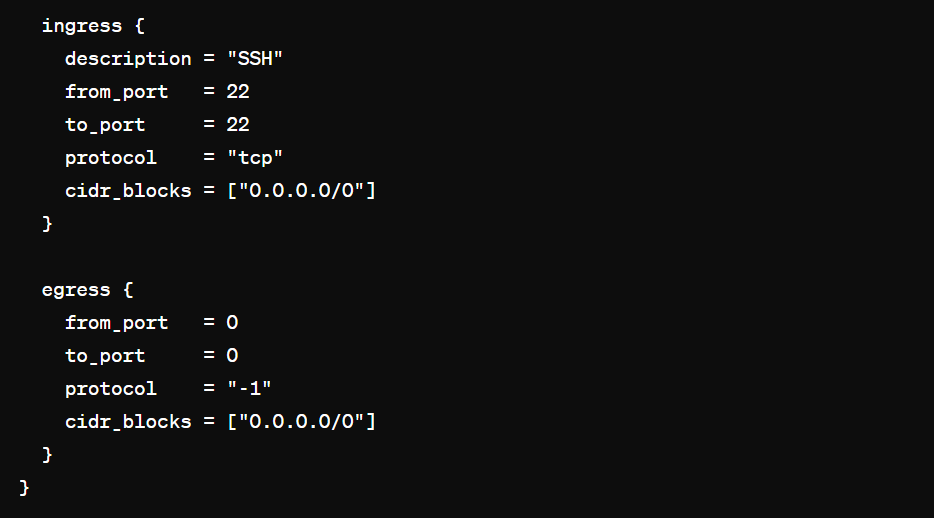
7. Route Table and Association:



* These blocks create a route table (RT) and associate it with the subnet sub1. The route table directs traffic destined for **0.0.0.0/0 (all traffic)** to the internet gateway created earlier.

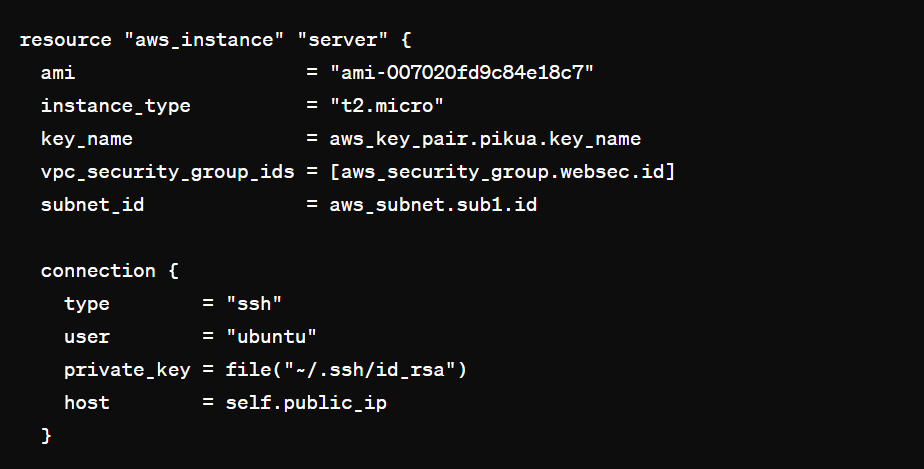
8. Security Group Creation:



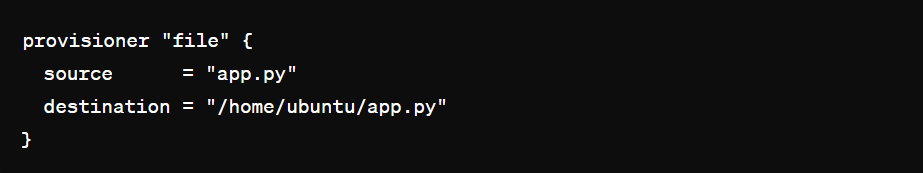


* This block creates a security group named webpage allowing HTTP (port 80) and SSH (port 22) traffic from any source (0.0.0.0/0).

9. EC2 Instance Creation:

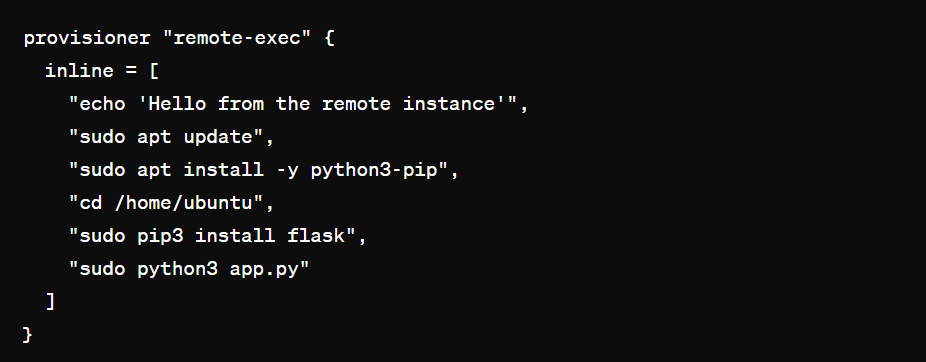


10. File Provisioner:



* This file provisioner block is used to transfer files from your local machine (source) to the EC2 instance (destination) during provisioning.
* **source:** Specifies the path to the file **(app.py)** on your local machine relative to the location of your Terraform configuration file.
* **destination:** Specifies the path on the EC2 instance where the file will be copied **(/home/ubuntu/app.py in this case).**

11. Remote-Exec Provisioner:



* This **remote-exec provisioner** block is used to execute commands on the EC2 instance remotely after it has been created and the file has been transferred.
* **inline:** Contains a list of commands to be executed sequentially on the EC2 instance.
* echo 'Hello from the remote instance':
* Outputs a message to indicate successful execution of the provisioner.
* ***sudo apt update:*** Updates the package lists on the instance.
* ***sudo apt install -y python3-pip***: Installs Python 3 pip package manager.
* ***cd /home/ubuntu:*** Changes the working directory to /home/ubuntu.
* ***sudo pip3 install flask:*** Installs Flask using pip3.
* ***sudo python3 app.py:*** Runs the Python script app.py using Python 3.
* These commands are typically used to set up the environment and dependencies required to run the Flask application (app.py) on the EC2 instance after it's provisioned.