Terraform Documentatoin

Description

This Terraform code defines and provisions an AWS Virtual Private Cloud (VPC) infrastructure along with associated resources, including subnets, an Internet Gateway, route table, route table associations, security group, and an EC2 instance. This infrastructure is designed to host a web application and provide network isolation and security.

Resources

1. **AWS VPC (myapp-vpc):**
    - Creates a Virtual Private Cloud.
    - CIDR Block: Defined by **var.vpc_cidr_block**.
    - Tags: Name tag is set to **${var.env_prefix}-vpc**.
2. **AWS Subnet (dev-subnet-1):**
    - Creates a subnet within the VPC.
    - Associated with Availability Zone: Defined by **var.avail_zone**.
    - CIDR Block: Defined by **var.subnet_cidr_block**.
    - Tags: Name tag is set to **${var.env_prefix}-subnet-1**.
3. **AWS Internet Gateway (myapp-igw):**
    - Creates an Internet Gateway.
    - Attached to the VPC.
4. **AWS Route Table (myapp-route-table):**
    - Creates a route table associated with the VPC.
    - Defines a default route to the Internet Gateway.
    - Tags: Name tag is set to **${var.env_prefix}-rtb**.
5. **AWS Route Table Association (myapp-ass-rtb-subnet):**
    - Associates the subnet **dev-subnet-1** with the **myapp-route-table**.
6. **AWS Security Group (myapp-sg):**
    - Creates a security group to control inbound and outbound traffic.
    - Inbound Rules: Allows incoming traffic on Ports 22 (SSH), 80 (HTTP), 8080, and 8083.
    - Outbound Rule: Allows all outbound traffic.

- Tags: Name tag is set to **${var.env_prefix}-sg**.

7. **AWS EC2 Instance (myapp-server):**
    - Launches an EC2 instance.
    - Uses the specified Amazon Machine Image (AMI).
    - Associated with **dev-subnet-1**.
    - Assigned a public IP address.
    - Uses the defined security group **myapp-sg**.
    - Key pair: Defined by **key_name**.
    - Tags: Name tag is set to **${var.env_prefix}-server**.

How to Run

1. Make sure you have Terraform installed and configured on your machine.
2. Create a **.tf** file with the provided code.
3. Modify the variables (e.g., **var.vpc_cidr_block**, **var.subnet_cidr_block**, **var.avail_zone**, **var.instance_type**, etc.) to match your requirements.
4. Open a terminal and navigate to the directory containing your **.tf** file.
5. Run the following Terraform commands:

    terraform init

    terraform plan

    terraform apply

    Terraform will display a plan of the resources to be created. Review it and proceed with **apply** to provision the infrastructure.

Once the deployment is complete, Terraform will provide output indicating the resources created.

You can access the EC2 instance using its public IP or DNS name.

Remember to manage your infrastructure responsibly. Use **terraform destroy** to remove the resources when they are no longer needed.

Please ensure you have AWS credentials configured properly on your machine and that the provided variables are accurate before running Terraform.

Ansible Playbook Documentation: Install Podman and Run Nginx Container

Description

This Ansible playbook automates the process of provisioning an AWS EC2 instance, installing Podman, and running an Nginx container on the instance. The playbook aims to create an isolated environment for hosting a web application.

Playbook Steps

1. **Install Podman on RHEL:**
    - The playbook uses the **yum** Ansible module to install Podman on the target RHEL instance.
    - The task ensures that Podman is available for running containers.
2. **Start Nginx Container:**
    - The playbook runs a Docker Nginx container using the **podman** command.
    - The **-d** flag detaches the container, **--name** sets the container name, and **-p** maps port 8083 on the host to port 80 in the container.
    - The Nginx container is pulled from the **docker.io/nginx** repository.

How to Run

1. Make sure you have Ansible installed and configured on your control machine.
2. Create a **.yml** file with the provided playbook.
3. Modify the **hosts** parameter to target the desired host(s) where you want to install Podman and run the Nginx container.
4. Open a terminal and navigate to the directory containing your **.yml** file.
5. Run the Ansible playbook using the following command:
    - Sudo ansible-playbook -i hosts playbook.yml
6. Ansible will execute the playbook tasks on the specified host(s).
7. Once the playbook execution is complete, an Nginx container will be running on the specified host(s), accessible via port 8083.

Note

- Ensure that you have SSH access to the target host(s) and the necessary permissions to execute tasks with elevated privileges (**become: yes**).
- Make sure the specified host(s) meet the requirements for running Podman and Docker containers.
- Validate the playbook on a test environment before running it in a production environment.

  Remember to manage your infrastructure responsibly. You can extend the playbook with additional tasks to configure security groups, firewall rules, and more, depending on your requirements.