MINI PROJECT: 1

TASKS:

1. Create an Ansible role to setup LAMP or XAMP
2. Create a Terraform script to provision stateful 3 tier architecture

TERRAFORM CONFIGURATION:

```
provider "aws" {
  region = "ap-south-1"
}

# Generate SSH Key
resource "tls_private_key" "ans_key" {
  algorithm = "RSA"
  rsa_bits  = 4096
}

resource "aws_key_pair" "ans_key" {
  key_name   = "ans_key"
  public_key = tls_private_key.ans_key.public_key_openssh
}

resource "local_file" "private_key" {
  content  = tls_private_key.ans_key.private_key_pem
  filename = "ans_key1.pem"
}

# VPC
```

```
resource "aws_vpc" "new_vpc" {
  cidr_block = "10.0.0.0/16"
}

# Internet Gateway
resource "aws_internet_gateway" "igw" {
  vpc_id = aws_vpc.new_vpc.id
}

# Subnets
resource "aws_subnet" "public_subnet" {
  vpc_id                  = aws_vpc.new_vpc.id
  cidr_block              = "10.0.1.0/24"
  availability_zone       = "ap-south-1a"
  map_public_ip_on_launch = true
  tags = {
    Name = "Public Subnet"
  }
}

resource "aws_subnet" "private_subnet" {
  vpc_id            = aws_vpc.new_vpc.id
  cidr_block        = "10.0.2.0/24"
  availability_zone = "ap-south-1b"
  tags = {
    Name = "Private Subnet"
  }
}

# Public Route Table
resource "aws_route_table" "public_route_table" {
  vpc_id = aws_vpc.new_vpc.id
  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.igw.id
  }
```

```
}

resource "aws_route_table_association" "public_association" {
  subnet_id      = aws_subnet.public_subnet.id
  route_table_id = aws_route_table.public_route_table.id
}

# Security Group
resource "aws_security_group" "ansible_sg" {
  name        = "ansible_sg"
  vpc_id      = aws_vpc.new_vpc.id
  description = "SG for Ansible allowing SSH, HTTP, HTTPS,
8080, ICMP"

  ingress {
    description = "allow ssh"
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  ingress {
    description = "allow http"
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  ingress {
    description = "allow https"
    from_port   = 443
    to_port     = 443
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
```

```
  }

  ingress {
    description = "allow port 8080"
    from_port   = 8080
    to_port     = 8080
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  ingress {
    description = "allow all icmp"
    from_port   = -1
    to_port     = -1
    protocol    = "icmp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  egress {
    description = "allow all outbound traffic"
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
}

# EC2 Instances
resource "aws_instance" "ansible_server" {
  ami                         = "ami-0d0ad8bb301edb745"
  instance_type               = "t2.micro"
  subnet_id                   = aws_subnet.public_subnet.id
  vpc_security_group_ids      =
[aws_security_group.ansible_sg.id]
  key_name                    = aws_key_pair.ans_key.key_name
  associate_public_ip_address = true
```

```
  tags = {
    Name = "Ansible Control Server"
  }
}

resource "aws_instance" "managerial_node" {
  ami                         = "ami-0d0ad8bb301edb745"
  instance_type               = "t2.micro"
  subnet_id                   = aws_subnet.public_subnet.id
  vpc_security_group_ids      =
[aws_security_group.ansible_sg.id]
  key_name                    = aws_key_pair.ans_key.key_name
  associate_public_ip_address = true
  tags = {
    Name = "Managerial Node 1"
  }
}

resource "aws_instance" "managerial_node2" {
  ami                         = "ami-0d0ad8bb301edb745"
  instance_type               = "t2.micro"
  subnet_id                   = aws_subnet.public_subnet.id
  vpc_security_group_ids = [aws_security_group.ansible_sg.id]
  key_name                    = aws_key_pair.ans_key.key_name
  associate_public_ip_address = true
  tags = {
    Name = "Managerial Node 2"
  }
}
```

main.tf

3-tier-arechitecture > main.tf > ...

```
133  resource "aws_instance" "managerial_node" {
137    vpc_security_group_ids      = [aws_security_group.ansible_sg.id]
138    key_name                    = aws_key_pair.ans_key.key_name
139    associate_public_ip_address = true
140    tags = {
141      Name = "Managerial Node 1"
142    }
143  }
144
145  resource "aws_instance" "managerial_node2" {
146    ami                         = "ami-0d0ad8bb301edb745"
147    instance_type               = "t2.micro"
148    subnet_id                   = aws_subnet.public_subnet.id
149    vpc_security_group_ids = [aws_security_group.ansible_sg.id]
150    key_name                    = aws_key_pair.ans_key.key_name
151    associate_public_ip_address = true
152    tags = {
153      Name = "Managerial Node 2"
154    }
155  }
156
157
```

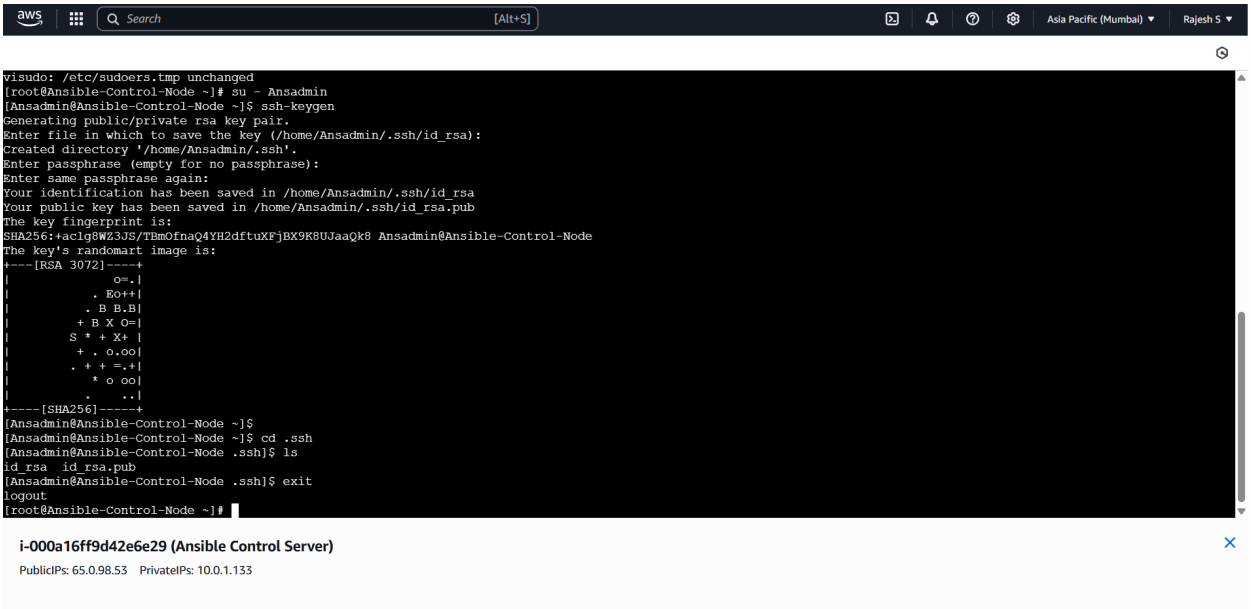
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS CODE REFERENCE LOG

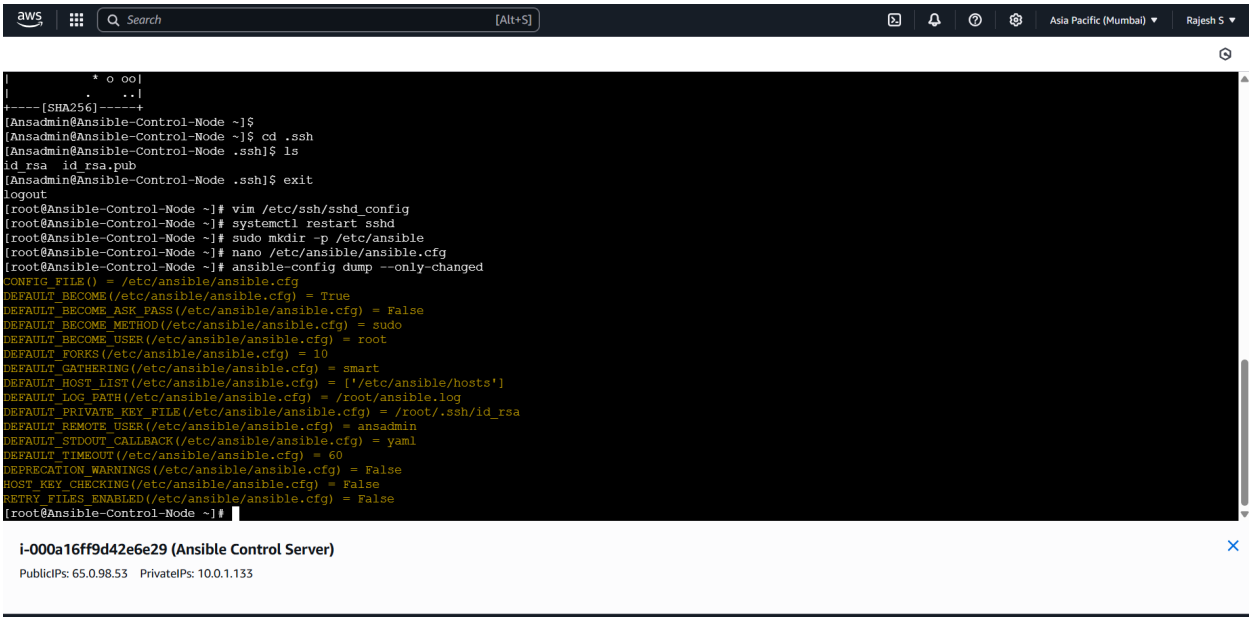
powershell

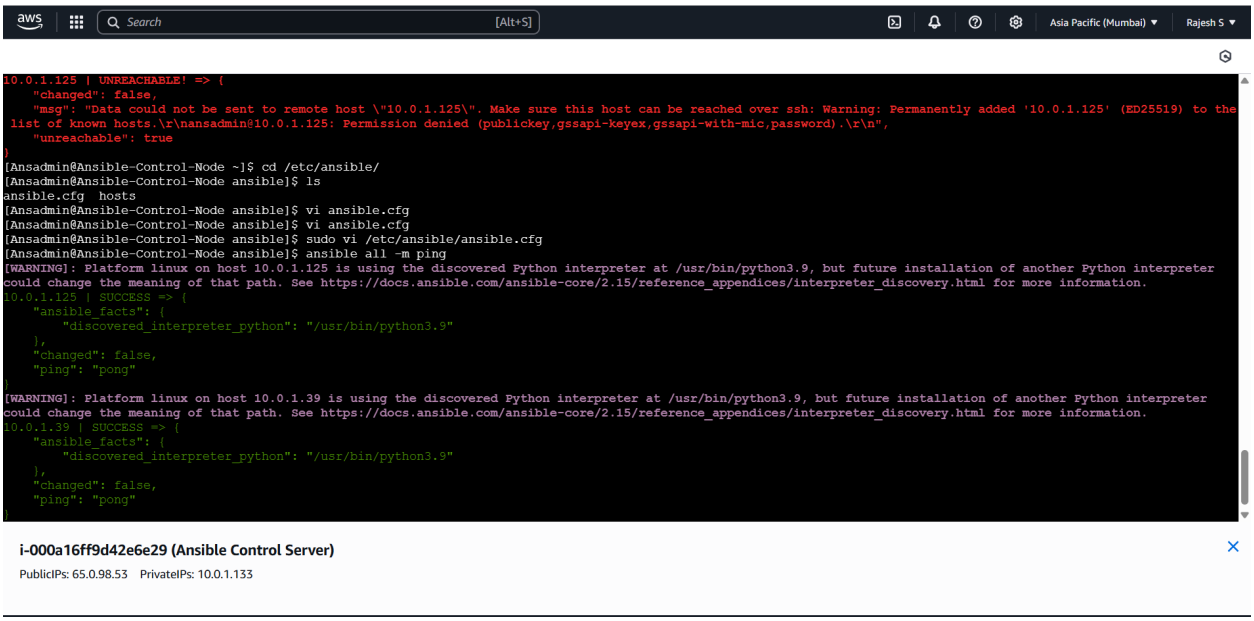
```
aws_instance.managerial_node: Still creating... [00m10s elapsed]
aws_instance.ansible_server: Creation complete after 17s [id=i-000a16ff9d42e6e29]
aws_instance.managerial_node2: Creation complete after 17s [id=i-028b519d397943310]
aws_instance.managerial_node: Still creating... [00m20s elapsed]
aws_instance.managerial_node: Still creating... [00m30s elapsed]
aws_instance.managerial_node: Creation complete after 33s [id=i-07c025fb3199fe3c3]

Apply complete! Resources: 13 added, 0 changed, 0 destroyed.
PS D:\Terraform\3-tier-arechitecture>
```

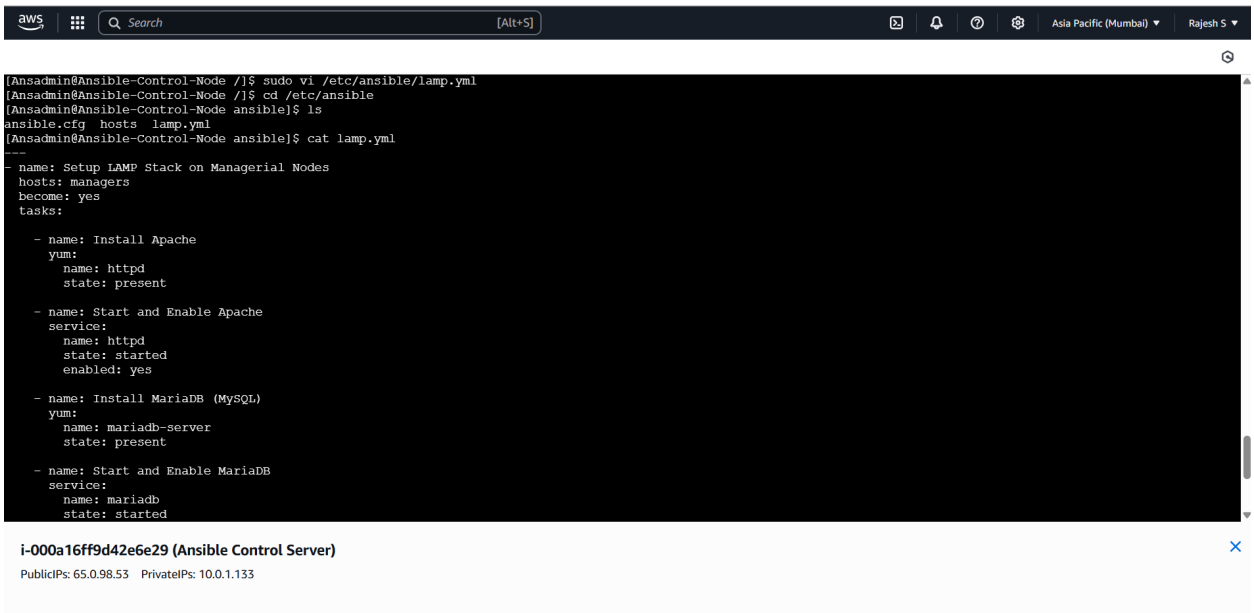
Ln 157, Col 1 Spaces: 2 UTF-8 CRLF Terraform Prettier

ANSIBLE CONFIGURATION:






aws
Search
[Alt+S]
Asia Pacific (Mumbai)
Rajesh S
10.0.1.125 | UNREACHABLE! => {
"changed": false,
"msg": "Data could not be sent to remote host \"10.0.1.125\". Make sure this host can be reached over ssh: Warning: Permanently added '10.0.1.125' (ED25519) to the list of known hosts.\r\nansadmin@10.0.1.125: Permission denied (publickey,gssapi-keyex,gssapi-with-mic,password).\r\n",
"unreachable": true
}
[Ansadmin@Ansible-Control-Node ~]$ cd /etc/ansible/
[Ansadmin@Ansible-Control-Node ansible]$ ls
ansible.cfg hosts
[Ansadmin@Ansible-Control-Node ansible]$ vi ansible.cfg
[Ansadmin@Ansible-Control-Node ansible]$ vi ansible.cfg
[Ansadmin@Ansible-Control-Node ansible]$ sudo vi /etc/ansible/ansible.cfg
[Ansadmin@Ansible-Control-Node ansible]$ ansible all -m ping
[WARNING]: Platform linux on host 10.0.1.125 is using the discovered Python interpreter at /usr/bin/python3.9, but future installation of another Python interpreter could change the meaning of that path. See https://docs.ansible.com/ansible-core/2.15/reference_appendices/interpreter_discovery.html for more information.
10.0.1.125 | SUCCESS => {
"ansible_facts": {
"discovered_interpreter_python": "/usr/bin/python3.9"
},
"changed": false,
"ping": "pong"
}
[WARNING]: Platform linux on host 10.0.1.39 is using the discovered Python interpreter at /usr/bin/python3.9, but future installation of another Python interpreter could change the meaning of that path. See https://docs.ansible.com/ansible-core/2.15/reference_appendices/interpreter_discovery.html for more information.
10.0.1.39 | SUCCESS => {
"ansible_facts": {
"discovered_interpreter_python": "/usr/bin/python3.9"
},
"changed": false,
"ping": "pong"
}
i-000a16ff9d42e6e29 (Ansible Control Server)
PublicIPs: 65.0.98.53 PrivateIPs: 10.0.1.133


aws
Search
[Alt+S]
Asia Pacific (Mumbai)
Rajesh S
[Ansadmin@Ansible-Control-Node /]$ sudo vi /etc/ansible/lamp.yml
[Ansadmin@Ansible-Control-Node /]$ cd /etc/ansible
[Ansadmin@Ansible-Control-Node ansible]$ ls
ansible.cfg hosts lamp.yml
[Ansadmin@Ansible-Control-Node ansible]$ cat lamp.yml

- name: Setup LAMP Stack on Managerial Nodes
hosts: managers
become: yes
tasks:
- name: Install Apache
yum:
name: httpd
state: present
- name: Start and Enable Apache
service:
name: httpd
state: started
enabled: yes
- name: Install MariaDB (MySQL)
yum:
name: mariadb-server
state: present
- name: Start and Enable MariaDB
service:
name: mariadb
state: started
i-000a16ff9d42e6e29 (Ansible Control Server)
PublicIPs: 65.0.98.53 PrivateIPs: 10.0.1.133

MANAGERIAL NODE :