Implementation 1

Goal: Complete all the problem statements and you are expected to upload end goal deployment screenshots in a simple doc to LMS.

Problem Statement: To Create a simple 3 node EC2 Virtual machine with Terraform and use ansible playbooks to deploy a sample LAMP web application on top of them. This should be done on Azure. The source code for this deployment should be checked into Git with proper branching strategies.

Expectation: Use Terraform to create 3 Node virtual machines on azure subscription given out to you.

1. State management should be in Azure storage account.
2. Use Ansible playbooks prewritten to deploy a sample LAMP stack on top of the Azure VMs created.
3. All the code l.e Terraform and Ansible playbooks to be committed to Azure repos in feature branches.

Solution:

This Project is implemented using Terraform and Ansible to display their capabilities

In this project we have used terraform to provision the infrastructure and install the LAMP stack on those VMs.

Followed below steps to complete this task:

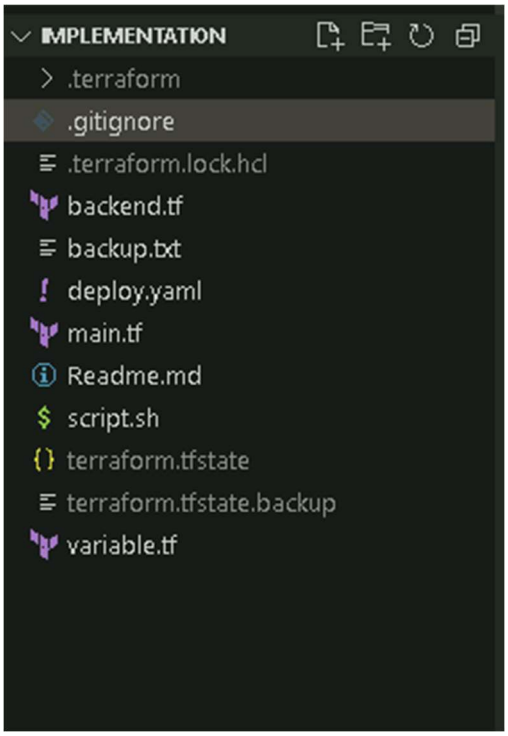
1. Used Terraform to provision below infra
    a. 1 Resource Group
    b. 1 Virtual Network
    c. 1 Subnet
    d. 1 Storage Account
    e. 3 Public Ips
    f. 3 Network Interface Card
    g. 3 Virtual Machines and copied public SSH key
2. Copied private key to master node.
3. Copied ansible file deploy.yaml to master node.
    a. Added code in the deploy.yaml file to Install the LAMP stack in all the hosts.
4. Copied ansible Inventory file to master node.
    a. Added all the VMs' private IPs in the Inventory file.
5. Copied script file to master node. The script has below code.
    a. Install Ansible
6. Update the private key permission.
    a. Run Ansible Playbook using inventory and deploy.yaml file.
7. Executed script file on the master node.

Source Code:

https://dev.azure.com/Shubham1700585806176/_git/Shubham_1700585806176

Project Structure:

Below is the file structure for the implementation:



Backend.tf

```
terraform {
    backend "azurerm" {
        resource_group_name = "state-resource-group"
        storage_account_name = "statestorageacc1812"
        container_name       = "state-container"
        key                  = "terraform.tfstate"
    }
}
```

Deploy.yaml

```
---
- hosts: all
  become: true
  tasks:
    - name: Update apt cache
      apt:
        update_cache: yes
```

```
    - name: Install Apache
      apt:
        name: apache2
        state: present

    - name: Install MySQL server
      apt:
        name: mysql-server
        state: present
      vars:
        mysql_root_password: root123

    - name: Install PHP and required modules
      apt:
        name: "{{ item }}"
        state: present
      with_items:
        - php
        - libapache2-mod-php
        - php-mysql  # PHP module for MySQL connectivity
        - php-curl   # Additional PHP modules as needed

    - name: Restart Apache
      service:
        name: apache2
        state: restarted
```

main.tf

```
provider "azurerm" {
  features {}
}

# ---------------------------------------------------------------------------
# Creating Storage Account For Terraform State
# ---------------------------------------------------------------------------

resource "azurerm_resource_group" "example_state" {
  name     = "state-resource-group"
  location = "East US"  # Change this to your desired Azure region
}

resource "azurerm_storage_account" "state_sa" {
  name = "statestorageacc1812"
  resource_group_name = azurerm_resource_group.example_state.name
  location = azurerm_resource_group.example_state.location
  account_tier = "${element(split("_", var.state_sa_type),0)}"
  account_replication_type = "${element(split("_", var.state_sa_type),1)}"
}

resource "azurerm_storage_container" "example" {
  name                  = "state-container"
  storage_account_name  = azurerm_storage_account.state_sa.name
}

# ---------------------------------------------------------------------------
# Creating Virtual Machine
# ---------------------------------------------------------------------------

resource "azurerm_resource_group" "example" {
  name     = "example-resource-group"
  location = "East US"  # Change this to your desired Azure region
}

resource "azurerm_virtual_network" "example" {
```

```
  name                = "example-vnet"
  address_space       = ["10.0.0.0/16"]
  location            = azurerm_resource_group.example.location
  resource_group_name = azurerm_resource_group.example.name
}

resource "azurerm_subnet" "example" {
  name                 = "example-subnet"
  resource_group_name  = azurerm_resource_group.example.name
  virtual_network_name = azurerm_virtual_network.example.name
  address_prefixes     = ["10.0.1.0/24"]
}

# ----------------------------------------------------------------------------
# This line is to follow company policy as boot diagnostics should be enabled
/*Create a storage account to create blob storage for the boot diag output*/

resource "azurerm_storage_account" "diagSA01" {
  name = "bootdiagsa021220232"
  resource_group_name = azurerm_resource_group.example.name
  location = azurerm_resource_group.example.location
  account_tier = "${element(split("_", var.boot_diagnostics_sa_type),0)}"
  account_replication_type = "${element(split("_", var.boot_diagnostics_sa_type),1)}"
}
# ----------------------------------------------------------------------------

resource "azurerm_public_ip" "example" {
    count = 3
    name = "example-ip-${count.index + 1}"
    location = azurerm_resource_group.example.location
    resource_group_name = azurerm_resource_group.example.name
    allocation_method = "Static"
}

resource "azurerm_network_interface" "example" {
  count               = 3
  name                = "example-nic-${count.index + 1}"
  location            = azurerm_resource_group.example.location
  resource_group_name = azurerm_resource_group.example.name

  ip_configuration {
    name                          = "example-nic-${count.index + 1}-ip"
    subnet_id                     = azurerm_subnet.example.id
    private_ip_address_allocation = "Dynamic"
    public_ip_address_id = azurerm_public_ip.example[count.index].id
  }
}

resource "azurerm_virtual_machine" "example" {
  count                 = 3
  name                  = "example-vm-${count.index + 1}"
  location              = azurerm_resource_group.example.location
  resource_group_name   = azurerm_resource_group.example.name
  network_interface_ids = [azurerm_network_interface.example[count.index].id]

  vm_size                  = "Standard_DS1_v2"  # Change this to your desired VM size
  delete_os_disk_on_termination = true

  # ----------------------------------------------------------------------------
  # This line is to follow company policy as boot diagnostics should be enabled
  boot_diagnostics {
    enabled = "true"
    storage_uri = azurerm_storage_account.diagSA01.primary_blob_endpoint
  }
  # ----------------------------------------------------------------------------

  storage_image_reference {
    publisher = "Canonical"
    offer     = "UbuntuServer"
```

```
    sku       = "18.04-LTS"
    version   = "latest"
  }

  storage_os_disk {
    name              = "example-osdisk-${count.index + 1}"
    caching           = "ReadWrite"
    create_option     = "FromImage"
    managed_disk_type = "Standard_LRS"
  }

  os_profile {
    computer_name  = "example-vm-${count.index + 1}"
    admin_username = "adminuser"  # Change this to your desired username
    admin_password = "Password1234!"  # Change this to your desired password
  }

  os_profile_linux_config {
    disable_password_authentication = true
    ssh_keys {
      path = "/home/adminuser/.ssh/authorized_keys"
      key_data = "ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQC1BDok0CFjh1KfPdhTwN+3mqXPUhVRp0+iUr2tINGH6Br/eKZik2+znbUZ0iWK3aDa5xtZ+m
bwRZ7XBUQlcFmy5TfKg687KAOlAVMJmXhZAHNjQJmZoPurqsr2UEkkMwZFz2ghakuj2yJfCPsJ7qfGIkpGtb2hg1HcX5e0PmJjVAPP
neiXxjNq64snreM9oig5pzWDiDuiZpley7A763U1malcYC+O12xiJ6dvyElDTuwj1ExPU4slCsk/XNP38tbfcCD5ItLhXbTeZ7BL3m
NYoBdmYP44n+5IazCqAMiexKU5MLVo+R9fXusSPwSoLUbHDJL12BeLyg71kdkCTuIB shubham.sharma3@cognizant.com"
    }
  }
}

resource "null_resource" "copy_private_key" {
  triggers = {
    always_run = timestamp()
  }

  provisioner "file" {
    source      = "C:/Users/VMUser/.ssh/id_rsa"
    destination = "/home/adminuser/.ssh/id_rsa"

    connection {
        type        = "ssh"
        user        = "adminuser"
        private_key = "${file("C:\\Users\\VMUser\\.ssh\\id_rsa")}"
        host        = azurerm_public_ip.example[0].ip_address
    }
  }
}

resource "null_resource" "copy_ansible_yaml" {
  triggers = {
    always_run = timestamp()
  }

  provisioner "file" {
    source      = "deploy.yaml"
    destination = "/tmp/deploy.yaml"

    connection {
        type        = "ssh"
        user        = "adminuser"
        private_key = "${file("C:\\Users\\VMUser\\.ssh\\id_rsa")}"
        host        = azurerm_public_ip.example[0].ip_address
    }
  }
}

resource "null_resource" "copy_ansible_inventory" {
  triggers = {
    always_run = timestamp()
```

```
  }

  provisioner "file" {
    content = <<EOF
      [localhost]
      ${azurerm_network_interface.example[0].private_ip_address} # Master Node Private IP

      [Node1]
      ${azurerm_network_interface.example[1].private_ip_address} # Node 1 Private IP

      [Node2]
      ${azurerm_network_interface.example[2].private_ip_address} # Node 2 Private IP

    EOF
    destination = "/tmp/inventory"

    connection {
        type        = "ssh"
        user        = "adminuser"
        private_key = "${file("C:\\Users\\VMUser\\.ssh\\id_rsa")}"
        host        = azurerm_public_ip.example[0].ip_address
    }
  }
}

resource "null_resource" "copy_script_file" {
  triggers = {
    always_run = timestamp()
  }

  provisioner "file" {
    source      = "script.sh"
    destination = "/tmp/script.sh"

    connection {
        type        = "ssh"
        user        = "adminuser"
        private_key = "${file("C:\\Users\\VMUser\\.ssh\\id_rsa")}"
        host        = azurerm_public_ip.example[0].ip_address
    }
  }
}

resource "null_resource" "execute_script" {
  triggers = {
    always_run = timestamp()
  }
  provisioner "remote-exec" {
    inline = [
        "chmod +x /tmp/script.sh ",
        "/tmp/script.sh"
    ]

    connection {
        type        = "ssh"
        user        = "adminuser"
        private_key = "${file("C:\\Users\\VMUser\\.ssh\\id_rsa")}"
        host        = azurerm_public_ip.example[0].ip_address
    }
  }
}
```

Variable.tf

```
variable "boot_diagnostics_sa_type" {
```

```
    default = "Standard_LRS"
}

variable "state_sa_type" {
    default = "Standard_LRS"
}
```

Script.sh

```
#!/bin/bash

# Update package lists
sudo apt update

# Install necessary dependencies
sudo apt install -y software-properties-common

# Add Ansible repository
sudo apt-add-repository --yes --update ppa:ansible/ansible

# Install Ansible
sudo apt install -y ansible

# Display Ansible version
ansible --version

echo "Ansible has been successfully installed."

# Set the desired filename for the SSH key
chmod 600 ~/.ssh/id_rsa
echo "Update the private key permission"

cd /tmp
ansible-playbook -i inventory deploy.yaml --ssh-extra-args='-o StrictHostKeyChecking=no'
echo "Run Ansible Playbook"
```

Commands Executed:

Apart from the commands mentioned in the above script.sh file executed below commands to initialize and apply terraform.

1. Below command to configure to save the tfstate file in the azure
    a. `terraform init –reconfigure`
2. Below command to inititalize terraform
    a. `terraform init`
3. Below command to apply terraform
    a. `terraform apply –auto-approve`