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.
   1. 1 Resource Group
   2. 1 Virtual Network
   3. 1 Subnet
   4. 1 Storage Account
   5. 3 Public Ips
   6. 3 Network Interface Card
   7. 3 Virtual Machines and copied public SSH key.
2. Copied private key to master node.
3. Copied ansible file deploy.yaml to master node.
   1. Added code in the deploy.yaml file to Install the LAMP stack in all the hosts.
4. Copied ansible Inventory file to master node.
   1. Added all the VMs' private IPs in the Inventory file.
5. Copied script file to master node. The script has below code.
   1. Install Ansible
6. Update the private key permission.
   1. 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:

A screenshot of a computer program

Description automatically generated

### 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+mbwRZ7XBUQlcFmy5TfKg687KAOlAVMJmXhZAHNjQJmZoPurqsr2UEkkMwZFz2ghakuj2yJfCPsJ7qfGIkpGtb2hg1HcX5e0PmJjVAPPneiXxjNq64snreM9oig5pzWDiDuiZpley7A763U1malcYC+O12xiJ6dvyElDTuwj1ExPU4slCsk/XNP38tbfcCD5ItLhXbTeZ7BL3mNYoBdmYP44n+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
   1. terraform init –reconfigure
2. Below command to inititalize terraform
   1. terraform init
3. Below command to apply terraform
   1. terraform apply –auto-approve