create a terraform modules script for the number of count of virtual machines and install iis server using powershell from terraform script in Azure

3 / 3

ChatGPT

**ChatGPT**

Below is an example Terraform configuration for deploying a specified number of virtual machines (VMs) in Azure using modules. The VMs will have IIS installed using PowerShell provisioners:

1. **Create a directory structure**:

cssCopy code

├── main.tf ├── variables.tf ├── outputs.tf ├── vm\_module │ ├── main.tf │ └── variables.tf └── iis\_script └── install\_iis.ps1

1. **Define the main.tf**:

hclCopy code

provider "azurerm" { features {} } module "vms" { source = "./vm\_module" count = var.vm\_count vm\_name\_prefix = "vm" vm\_size = var.vm\_size subnet\_id = var.subnet\_id admin\_username = var.admin\_username admin\_password = var.admin\_password }

1. **Define the variables.tf**:

hclCopy code

variable "vm\_count" { description = "Number of virtual machines to deploy" type = number default = 2 } variable "vm\_size" { description = "Size of the virtual machines" type = string default = "Standard\_DS1\_v2" } variable "subnet\_id" { description = "ID of the subnet to deploy the VMs" type = string } variable "admin\_username" { description = "Admin username for the VMs" type = string } variable "admin\_password" { description = "Admin password for the VMs" type = string }

1. **Define the outputs.tf**:

hclCopy code

output "vm\_ips" { description = "Public IPs of the deployed VMs" value = module.vms.public\_ips }

1. **Define the vm\_module/main.tf**:

hclCopy code

resource "azurerm\_network\_interface" "nic" { count = var.vm\_count name = "${var.vm\_name\_prefix}-${count.index}-nic" location = azurerm\_subnet.subnet.location resource\_group\_name = azurerm\_subnet.subnet.resource\_group\_name ip\_configuration { name = "ipconfig1" subnet\_id = var.subnet\_id private\_ip\_address\_allocation = "Dynamic" } } resource "azurerm\_virtual\_machine" "vm" { count = var.vm\_count name = "${var.vm\_name\_prefix}-${count.index}" location = azurerm\_subnet.subnet.location resource\_group\_name = azurerm\_subnet.subnet.resource\_group\_name network\_interface\_ids = [azurerm\_network\_interface.nic[count.index].id] vm\_size = var.vm\_size delete\_os\_disk\_on\_termination = true storage\_image\_reference { publisher = "MicrosoftWindowsServer" offer = "WindowsServer" sku = "2019-datacenter" version = "latest" } storage\_os\_disk { name = "${var.vm\_name\_prefix}-${count.index}-osdisk" caching = "ReadWrite" create\_option = "FromImage" managed\_disk\_type = "Premium\_LRS" } os\_profile { computer\_name = "${var.vm\_name\_prefix}-${count.index}" admin\_username = var.admin\_username admin\_password = var.admin\_password } os\_profile\_windows\_config { provision\_vm\_agent = true } } provisioner "remote-exec" { inline = [ "Set-ExecutionPolicy Unrestricted -Force", "Invoke-Expression ((New-Object System.Net.WebClient).DownloadString('https://chocolatey.org/install.ps1'))", "choco install -y iis", "Restart-Service -Name 'W3SVC'" ] connection { type = "winrm" host = azurerm\_public\_ip.vm[count.index].ip\_address user = var.admin\_username password = var.admin\_password https = true port = 5986 timeout = "20m" cacert\_path = "./vm\_module/azurecert.pem" } }

1. **Define the vm\_module/variables.tf**:

hclCopy code

variable "vm\_name\_prefix" { description = "Prefix for the VM names" type = string } variable "vm\_size" { description = "Size of the virtual machines" type = string } variable "subnet\_id" { description = "ID of the subnet to deploy the VMs" type = string } variable "admin\_username" { description = "Admin username for the VMs" type = string } variable "admin\_password" { description = "Admin password for the VMs" type = string }

1. **Place the PowerShell script (install\_iis.ps1) in the iis\_script directory**:

powershellCopy code

Install-WindowsFeature -Name Web-Server -IncludeManagementTools

1. **Ensure to have Azure CLI configured and Terraform installed on your local machine.**
2. **Run the following commands**:

bashCopy code

terraform init terraform plan terraform apply

This setup will deploy the specified number of VMs in Azure, install IIS using the provided PowerShell script, and output the public IPs of the deployed VMs. Make sure to replace placeholder values like **var.subnet\_id**, **var.admin\_username**, and **var.admin\_password** with your actual values. Additionally, ensure that you have the necessary permissions and access to Azure resources.

1. ├── main.tf
2. ├── variables.tf
3. ├── outputs.tf
4. ├── vm\_module
5. │ ├── main.tf
6. │ └── variables.tf
7. └── iis\_script
8. └── install\_iis.ps1

2. provider "azurerm" {

features {}

}

module "vms" {

source = "./vm\_module"

count = var.vm\_count

vm\_name\_prefix = "vm"

vm\_size = var.vm\_size

subnet\_id = var.subnet\_id

admin\_username = var.admin\_username

admin\_password = var.admin\_password

}

3. variable "vm\_count" {

description = "Number of virtual machines to deploy"

type = number

default = 2

}

variable "vm\_size" {

description = "Size of the virtual machines"

type = string

default = "Standard\_DS1\_v2"

}

variable "subnet\_id" {

description = "ID of the subnet to deploy the VMs"

type = string

}

variable "admin\_username" {

description = "Admin username for the VMs"

type = string

}

variable "admin\_password" {

description = "Admin password for the VMs"

type = string

}

4. output "vm\_ips" {

description = "Public IPs of the deployed VMs"

value = module.vms.public\_ips

}

5. resource "azurerm\_network\_interface" "nic" {

count = var.vm\_count

name = "${var.vm\_name\_prefix}-${count.index}-nic"

location = azurerm\_subnet.subnet.location

resource\_group\_name = azurerm\_subnet.subnet.resource\_group\_name

ip\_configuration {

name = "ipconfig1"

subnet\_id = var.subnet\_id

private\_ip\_address\_allocation = "Dynamic"

}

}

resource "azurerm\_virtual\_machine" "vm" {

count = var.vm\_count

name = "${var.vm\_name\_prefix}-${count.index}"

location = azurerm\_subnet.subnet.location

resource\_group\_name = azurerm\_subnet.subnet.resource\_group\_name

network\_interface\_ids = [azurerm\_network\_interface.nic[count.index].id]

vm\_size = var.vm\_size

delete\_os\_disk\_on\_termination = true

storage\_image\_reference {

publisher = "MicrosoftWindowsServer"

offer = "WindowsServer"

sku = "2019-datacenter"

version = "latest"

}

storage\_os\_disk {

name = "${var.vm\_name\_prefix}-${count.index}-osdisk"

caching = "ReadWrite"

create\_option = "FromImage"

managed\_disk\_type = "Premium\_LRS"

}

os\_profile {

computer\_name = "${var.vm\_name\_prefix}-${count.index}"

admin\_username = var.admin\_username

admin\_password = var.admin\_password

}

os\_profile\_windows\_config {

provision\_vm\_agent = true

}

}

provisioner "remote-exec" {

inline = [

"Set-ExecutionPolicy Unrestricted -Force",

"Invoke-Expression ((New-Object System.Net.WebClient).DownloadString('https://chocolatey.org/install.ps1'))",

"choco install -y iis",

"Restart-Service -Name 'W3SVC'"

]

connection {

type = "winrm"

host = azurerm\_public\_ip.vm[count.index].ip\_address

user = var.admin\_username

password = var.admin\_password

https = true

port = 5986

timeout = "20m"

cacert\_path = "./vm\_module/azurecert.pem"

}

}

6. variable "vm\_name\_prefix" {

description = "Prefix for the VM names"

type = string

}

variable "vm\_size" {

description = "Size of the virtual machines"

type = string

}

variable "subnet\_id" {

description = "ID of the subnet to deploy the VMs"

type = string

}

variable "admin\_username" {

description = "Admin username for the VMs"

type = string

}

variable "admin\_password" {

description = "Admin password for the VMs"

type = string

}

7. Install-WindowsFeature -Name Web-Server -IncludeManagementTools

1. **8. Ensure to have Azure CLI configured and Terraform installed on your local machine.**

9 terraform init

terraform plan

terraform apply

This setup will deploy the specified number of VMs in Azure, install IIS using the provided PowerShell script, and output the public IPs of the deployed VMs. Make sure to replace placeholder values like **var.subnet\_id**, **var.admin\_username**, and **var.admin\_password** with your actual values. Additionally, ensure that you have the necessary permissions and access to Azure resources.

Top of Form

222222

create a terraform modules script for the number of count of virtual machines and install iis server using PowerShell, vnet creation, app service plan, app service to check dependency with app service plan before creation of app service, key vault. create a three environments like development, testing, and production and save statefile using backend in stoarge account from terraform script in Azure

Here's an example of how you can structure your Terraform configuration into modules and set up different environments (development, testing, production) with the specified resources including virtual machines, Azure App Service, Virtual Network, App Service Plan, and Key Vault. Additionally, it incorporates state file management using Azure Storage Account as a backend.

# main.tf

provider "azurerm" {

features {}

}

module "virtual\_machines" {

source = "./modules/virtual\_machines"

count = var.vm\_count

}

module "app\_service\_plan" {

source = "./modules/app\_service\_plan"

}

module "app\_service" {

source = "./modules/app\_service"

app\_service\_plan\_id = module.app\_service\_plan.id

}

module "vnet" {

source = "./modules/vnet"

}

module "key\_vault" {

source = "./modules/key\_vault"

}

# Backend configuration

terraform {

backend "azurerm" {

storage\_account\_name = "your\_storage\_account\_name"

container\_name = "your\_container\_name"

key = "terraform.tfstate"

}

}

You would need to create directories for each module (**virtual\_machines**, **app\_service\_plan**, **app\_service**, **vnet**, **key\_vault**) and within each directory, create **main.tf** files for configuring the resources specific to that module. Here's a basic structure:

root

├── main.tf

├── variables.tf

├── outputs.tf

├── modules

│ ├── virtual\_machines

│ │ ├── main.tf

│ │ ├── variables.tf

│ │ └── outputs.tf

│ ├── app\_service\_plan

│ │ ├── main.tf

│ │ ├── variables.tf

│ │ └── outputs.tf

│ ├── app\_service

│ │ ├── main.tf

│ │ ├── variables.tf

│ │ └── outputs.tf

│ ├── vnet

│ │ ├── main.tf

│ │ ├── variables.tf

│ │ └── outputs.tf

│ └── key\_vault

│ ├── main.tf

│ ├── variables.tf

│ └── outputs.tf

You need to fill in the configurations for each module according to your specific requirements. For instance, for the **virtual\_machines** module, you'll define the Azure VM resources and the necessary PowerShell scripts to install IIS Server.

Remember to define variables (**variables.tf**) and outputs (**outputs.tf**) in each module to make them reusable and to retrieve important information.

Additionally, make sure to replace **"your\_storage\_account\_name"** and **"your\_container\_name"** with the actual names of your Azure Storage Account and container where you want to store the Terraform state file.

333333333333333333333

terraform/

│

├── backend/ # Main module for backend configuration

│ ├── dev/ # Child module for dev environment

│ ├── qa/ # Child module for qa environment

│ └── prod/ # Child module for prod environment

│

└── main.tf # Main Terraform configuration file

module "backend\_dev" {

source = "./backend/dev"

// Define variables specific to dev environment here

}

module "backend\_qa" {

source = "./backend/qa"

// Define variables specific to qa environment here

}

module "backend\_prod" {

source = "./backend/prod"

// Define variables specific to prod environment here

}

Each child module (**backend/dev**, **backend/qa**, and **backend/prod**) will have its own **main.tf**, **variables.tf**, and other necessary files to configure the backend resources specific to that environment.

You can then execute the following commands to initialize, plan, and apply the Terraform configurations:

terraform init

terraform plan -out=tfplan

terraform apply tfplan

This will create separate Terraform state files for dev, qa, and prod environments, allowing you to manage them independently. Make sure to replace the comments with actual variable definitions and configurations specific to each environment.