**About** **Resources**

* Resource block describes one or more infrastructure objects, such as virtual networks, compute instances, or higher-level components such as DNS records.
* Each resource is associated with a single resource type, which determines the kind of infrastructure object it manages and what arguments and other attributes the resource supports.
* Each resource is associated with a single resource type, which determines the kind of infrastructure object it manages and what arguments and other attributes the resource supports.

**Resource Syntax**

resource "azurerm\_resource\_group" "my" {

name = "Sandeep-Terraform-rg"

location = "east us"

}

Most arguments in this section depend on the resource type.

Creating Azure App Services

Add the following to main.tf

resource "azurerm\_resource\_group" "terraform-rg" {

  name     = "sandeep-terraform-rg"

  location = "eastus"

}

resource "azurerm\_service\_plan" "demo-appservice-plan" {

  name                = "sandeep-demo-appservice-plan"

  location            = azurerm\_resource\_group.terraform-rg.location

  resource\_group\_name = azurerm\_resource\_group.terraform-rg.name

  os\_type             = "Linux"

  sku\_name         = "B1"

}

resource "azurerm\_linux\_web\_app" "demo-appservice" {

  name                = "sandeep-demo-appservice"

  location            = azurerm\_resource\_group.terraform-rg.location

  resource\_group\_name = azurerm\_resource\_group.terraform-rg.name

  service\_plan\_id = azurerm\_service\_plan.demo-appservice-plan.id

  site\_config {

    application\_stack {

      dotnet\_version = "7.0"

    }

    always\_on = false

  }

  app\_settings = {

    "Setting1" = "Value1"

    "Setting2" = "Value2"

  }

  connection\_string {

    name  = "Database"

    type  = "SQLServer"

    value = "Server=some-server.mydomain.com;Integrated Security=SSPI"

  }

}

**output** "app\_service\_default\_hostname" {

  value = azurerm\_linux\_web\_app.demo-appservice.default\_hostname

}

C:\terraformdemos>**terraform init**

C:\terraformdemos>**terraform plan -out=tfplan**

C:\terraformdemos>**terraform apply tfplan**

Create Azure VM

**Steps to Create a VM in Azure**

1. Create Resource Group
2. Create Virtual Network
3. Create Subnet
4. Create Public IP Address
5. Create Network Security Group
6. Create Network Interface Card
7. Connect Network Security Group to Network Interface
8. Create and Save SSH Key
9. Create VM

**Azure-vm.tf**

resource "azurerm\_resource\_group" "rg" {

  name     = "Terraform-rg"

  location = "east us"

}

# Create virtual network

resource "azurerm\_virtual\_network" "myNetwork" {

  name                = "myVnet"

  address\_space       = ["10.0.0.0/16"]

  resource\_group\_name = azurerm\_resource\_group.rg.name

  location             = azurerm\_resource\_group.rg.location

}

# Create subnet

resource "azurerm\_subnet" "mySubnet" {

  name                 = "mySubnet"

  resource\_group\_name  = azurerm\_resource\_group.rg.name

  virtual\_network\_name = azurerm\_virtual\_network.myNetwork.name

  address\_prefixes     = ["10.0.1.0/24"]

}

# Create public IPs

resource "azurerm\_public\_ip" "myVMPublicIP" {

  name                = "myPublicIP"

  location            = azurerm\_resource\_group.rg.location

  resource\_group\_name = azurerm\_resource\_group.rg.name

  allocation\_method   = "Dynamic"

}

# Create Network Security Group and rule

resource "azurerm\_network\_security\_group" "myNSG" {

  name                = "myNetworkSecurityGroup"

  location            = azurerm\_resource\_group.rg.location

  resource\_group\_name = azurerm\_resource\_group.rg.name

  security\_rule {

    name                       = "SSH"

    priority                   = 1001

    direction                   = "Inbound"

    access                     = "Allow"

    protocol                   = "Tcp"

    source\_port\_range       = "\*"

    destination\_port\_range = "22"

    source\_address\_prefix   = "\*"

    destination\_address\_prefix = "\*"

  }

  security\_rule {

    name                       = "HTTP"

    priority                   = 1000

    direction                   = "Inbound"

    access                     = "Allow"

    protocol                   = "Tcp"

    source\_port\_range          = "\*"

    destination\_port\_range = "80"

    source\_address\_prefix    = "\*"

    destination\_address\_prefix = "\*"

  }

}

# Create network interface

resource "azurerm\_network\_interface" "myNIC" {

  name                = "myNIC"

  location            = azurerm\_resource\_group.rg.location

  resource\_group\_name = azurerm\_resource\_group.rg.name

  ip\_configuration {

    name                           = "myNIC"

    subnet\_id                     = azurerm\_subnet.mySubnet.id

    private\_ip\_address\_allocation = "Dynamic"

    public\_ip\_address\_id          = azurerm\_public\_ip.myVMPublicIP.id

  }

}

# Connect the security group to the network interface

resource "azurerm\_network\_interface\_security\_group\_association" "example" {

  network\_interface\_id       = azurerm\_network\_interface.myNIC.id

  network\_security\_group\_id = azurerm\_network\_security\_group.myNSG.id

}

# Create (and display) an SSH key

resource "tls\_private\_key" "mySSH" {

  algorithm = "RSA"

  rsa\_bits  = 4096

}

# Save SSH to Login to Remote VM.

resource "local\_file" "private\_key" {

  content         = tls\_private\_key.mySSH.private\_key\_pem

  filename        = "azure.pem"

  file\_permission = "0600"

}

# Create virtual machine

resource "azurerm\_linux\_virtual\_machine" "my\_terraform\_vm" {

  name                  = "myVM"

  location              = azurerm\_resource\_group.rg.location

  resource\_group\_name   = azurerm\_resource\_group.rg.name

  network\_interface\_ids = [azurerm\_network\_interface.myNIC.id]

  size                  = "Standard\_DS1\_v2"

  os\_disk {

    name                 = "myOsDisk"

    caching              = "ReadWrite"

    storage\_account\_type = "Premium\_LRS"

  }

  source\_image\_reference {

    publisher = "Canonical"

    offer     = "UbuntuServer"

    sku       = "18.04-LTS"

    version   = "latest"

  }

  computer\_name                   = "myvm"

  admin\_username                  = "azureuser"

  admin\_password                  = "Password@123"

  disable\_password\_authentication = false

  admin\_ssh\_key {

    username   = "azureuser"

    public\_key = tls\_private\_key.mySSH.public\_key\_openssh

  }

  depends\_on = [

    azurerm\_network\_interface\_security\_group\_association.example

  ]

}

output azureVM\_public\_ip {

value = azurerm\_public\_ip.myVMPublicIP.ip\_address

depends\_on = [

azurerm\_linux\_virtual\_machine.my\_terraform\_vm

]

}

**Terraform Commands**

* terraform init
* terraform plan -out=tfplan
* terraform apply tfplan --auto-approve
* terraform destroy --auto-approve

**use below command to login to Azure VM**

**ssh -i azure.pem** [**azureuser@20.185.35.66**](mailto:azureuser@20.185.35.66)

Resources Meta-Arguments

The Terraform language defines several meta-arguments, which can be used with any resource type to change the behavior of resources.

The following meta-arguments are documented on separate pages:

* **count**, for creating multiple resource instances according to a count.
* **for\_each,** to create multiple instances according to a map, or set of strings.
* **depends\_on**, for specifying hidden dependencies.

The Count Meta Argument

If a resource or module block includes a count argument whose value is a whole number, Terraform will create that many instances.

Each instance has a distinct infrastructure object associated with it, and each is separately created, updated, or destroyed when the configuration is applied.

resource "azurerm\_resource\_group" "myGroups" {

  count=2

  name     =  **"Terraform${count.index}-rg"**

  location = "east us"

}

[**count.index**](https://developer.hashicorp.com/terraform/language/meta-arguments/count#count-index) — The distinct index number (starting with 0) corresponding to this instance.

**String formatting:**

format("%s - %s", "this is index ", count.index)

**Example:**

resource "azurerm\_resource\_group" "myGroups" {

  count = 2

  name     =  **format("Terraform%s-rg",count.index)**

  location = "east us"

}

**Referring to Instances**

When count is set, Terraform distinguishes between the block itself and the multiple resource or module instances associated with it. Instances are identified by an index number, starting with 0.

<TYPE>.<NAME>**[**<INDEX>**]**

Example:

azurerm\_resource\_group.myGroups[0]

azurerm\_resource\_group.myGroups[1]

Create Multiple Azure VM using Count

**Steps to Create a VM in Azure**

1. Create Resource Group (One)
2. Create Virtual Network (One)
3. Create Subnet (One)
4. Create Public IP Address (Multiple)
5. Create Network Security Group (One)
6. Create Network Interface Card (Multiple)
7. Connect Network Security Group to Network Interface (Multiple)
8. Create and Save SSH Key (One)
9. Create VM (Multiple)
   1. Disk (Multiple)

**create-vnet.tf**

resource "azurerm\_resource\_group" "rg" {

name = "Terraform-rg"

location = "east us"

}

# Create virtual network

resource "azurerm\_virtual\_network" "myNetwork" {

  name                 = "myVnet"

  address\_space = ["10.1.0.0/16"]

  location             = azurerm\_resource\_group.rg.location

  resource\_group\_name = azurerm\_resource\_group.rg.name

}

# Create subnet

resource "azurerm\_subnet" "mySubnet" {

  name                 = "mySubnet"

  resource\_group\_name  = azurerm\_resource\_group.rg.name

  virtual\_network\_name = azurerm\_virtual\_network.myNetwork.name

  address\_prefixes     = ["10.1.1.0/24"]

}

# Create public IPs

resource "azurerm\_public\_ip" "myVMPublicIPs" {

**count               = 2**

  name                = **format("myVM%s-ip", count.index)**

  location            = azurerm\_resource\_group.rg.location

  resource\_group\_name = azurerm\_resource\_group.rg.name

  allocation\_method   = "Dynamic"

}

# Create Network Security Group and rule

resource "azurerm\_network\_security\_group" "myNSG" {

  name                = "myNetworkSecurityGroup"

  location            = azurerm\_resource\_group.rg.location

  resource\_group\_name = azurerm\_resource\_group.rg.name

  security\_rule {

    name                       = "SSH"

    priority                   = 1001

    direction                  = "Inbound"

    access                     = "Allow"

    protocol                   = "Tcp"

    source\_port\_range          = "\*"

    destination\_port\_range     = "22"

    source\_address\_prefix      = "\*"

    destination\_address\_prefix = "\*"

  }

  security\_rule {

    name                       = "HTTP"

    priority                   = 1000

    direction                  = "Inbound"

    access                     = "Allow"

    protocol                   = "Tcp"

    source\_port\_range          = "\*"

    destination\_port\_range     = "80"

    source\_address\_prefix      = "\*"

    destination\_address\_prefix = "\*"

  }

}

# Create network interface

resource "azurerm\_network\_interface" "myNICs" {

**count = 2**

  name                = **format("myVM%s-nic", count.index)**

  location            = azurerm\_resource\_group.rg.location

  resource\_group\_name = azurerm\_resource\_group.rg.name

  ip\_configuration {

    name                          = "myNIC"

    subnet\_id                     = azurerm\_subnet.mySubnet.id

    private\_ip\_address\_allocation = "Dynamic"

    public\_ip\_address\_id          = **azurerm\_public\_ip.myVMPublicIPs[count.index].id**

  }

}

# Connect the security group to the network interface

resource "azurerm\_network\_interface\_security\_group\_association" "example" {

**count = 2**

  network\_interface\_id      = azurerm\_network\_interface.myNICs[**count.index**].id

  network\_security\_group\_id = azurerm\_network\_security\_group.myNSG.id

}

**Create-vm.tf**

# Create (and display) an SSH key

resource "tls\_private\_key" "mySSH" {

  algorithm = "RSA"

  rsa\_bits  = 4096

}

resource "local\_file" "private\_key" {

  content         = tls\_private\_key.mySSH.private\_key\_pem

  filename        = "azure.pem"

  file\_permission = "0600"

}

# Create virtual machine

resource "azurerm\_linux\_virtual\_machine" "my\_terraform\_vm" {

  count = 2

  name                  = **format("myVM%s-vm", count.index)**

  location              = azurerm\_resource\_group.rg.location

  resource\_group\_name   = azurerm\_resource\_group.rg.name

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

  size                  = "Standard\_DS1\_v2"

  os\_disk {

    name                 = format("myVM%s-OsDisk",count.index)

    caching              = "ReadWrite"

    storage\_account\_type = "Premium\_LRS"

  }

  source\_image\_reference {

    publisher = "Canonical"

    offer     = "UbuntuServer"

    sku       = "18.04-LTS"

    version   = "latest"

  }

  computer\_name                   = "myvm"

  admin\_username                  = "azureuser"

  admin\_password                  = "Password@123"

  disable\_password\_authentication = false

  admin\_ssh\_key {

    username   = "azureuser"

    public\_key = tls\_private\_key.mySSH.public\_key\_openssh

  }

  depends\_on = [

    azurerm\_network\_interface\_security\_group\_association.example

  ]

}

**output** "virtual\_machine\_ip" {

value = [

for vm in azurerm\_linux\_virtual\_machine.my\_terraform\_vm : vm.public\_ip\_address

]

depends\_on = [

azurerm\_linux\_virtual\_machine.my\_terraform\_vm

]

}

**Terraform Commands**

* terraform init
* terraform plan -out=tfplan
* terraform apply tfplan --auto-approve
* terraform destroy --auto-approve

**Steps to make the .pem file Readonly on Windows**

1. select .pem file -> right click -> properties
2. Security > Advanced > Disable inheritance
3. Remove all Users
4. Add > Select a principal
5. In "Enter the object name to select" type your Windows username > ok
6. Give all permissions > ok > apply

**Connect to VM using the below Command**

ssh -i azure.pem azureuser@<IPofVM>

The for\_each Meta Argument

If a resource or module block includes a for\_each argument whose value is a map or a set of strings, Terraform creates one instance for each member of that map or set.

Note: A given resource or module block cannot use both count and for\_each.

The for\_each meta-argument accepts a map or a set of strings, and creates an instance for each item in that map or set. Each instance has a distinct infrastructure object associated with it, and each is separately created, updated, or destroyed when the configuration is applied.

**each Object**

In blocks where for\_each is set, an additional each object is available in expressions, so you can modify the configuration of each instance. This object has two attributes:

* **each.key** — The map key (or set member) corresponding to this instance.
* **each.value** — The map value corresponding to this instance. (**If a set was provided, this is the same as each.key**)

For Each with Map:

Example: Create two resource groups in Azure

resource "azurerm\_resource\_group" "rg" {

**for\_each = {**

**Dev-rg = "eastus"**

**Prod-rg = "westus"**

**}**

name = each.key

location = each.value

}

**For Each with Set of strings:**

**Create multiple App Services for different environments.**

resource "azurerm\_linux\_web\_app" "demo-appservice" {

  for\_each = toset(["dev","qa","prod"])

  name                = "sandeep-demo-appservice-${each.key}"

. . .

**For each using Map and value as object:**

resource "azurerm\_linux\_virtual\_machine" "myVMs" {

**for\_each = {**

**vm1 = {**

**index = 0**

**name = "vm1"**

**size = "standard\_b2ms"**

**}**

**vm2 = {**

**index = 1**

**name = "dbserver1"**

**size = "Standard\_F2"**

**}**

  }

  name                  = **each.value.name**

  location              = azurerm\_resource\_group.rg.location

  resource\_group\_name   = azurerm\_resource\_group.rg.name

  network\_interface\_ids = [azurerm\_network\_interface.myNICs[**each.value.index**].id]

  size                  = each.value.size

  os\_disk {

    name                 = format("myVM%s-OsDisk", **each.value.index**)

    caching              = "ReadWrite"

    storage\_account\_type = "Premium\_LRS"

  }

  source\_image\_reference {

    publisher = "Canonical"

    offer     = "UbuntuServer"

    sku       = "18.04-LTS"

    version   = "latest"

  }

  computer\_name                   = **each.value.name**

  admin\_username                  = "azureuser"

  admin\_password                  = "Password@123"

  disable\_password\_authentication = false

  admin\_ssh\_key {

    username   = "azureuser"

    public\_key = tls\_private\_key.mySSH.public\_key\_openssh

  }

}

**When to Use for\_each instead of count**

If your instances are almost identical, count is appropriate. If some of their arguments need distinct values that can't be directly derived from an integer, it's safer to use for\_each.

Creation of Load Balancer in Azure

a) Create Standard SKU Public IP Address (azurerm\_public\_ip)

b) Create Standard Load Balancer (azurerm\_lb)

c) Create Backend Pool (azurerm\_lb\_backend\_address\_pool)

d) Add VMs IP Addresses to Backend Pool (azurerm\_lb\_backend\_address\_pool\_address)

e) Create Health Probe (azurerm\_lb\_probe)

f) Create Load Balancer Rule (azurerm\_lb\_rule)

**create-loadbalancer.tf**

resource "azurerm\_public\_ip" "testLoadBalancer-ip" {

  name                = "PublicIPForLB"

  location            = azurerm\_resource\_group.rg.location

  resource\_group\_name = azurerm\_resource\_group.rg.name

  allocation\_method   = "Static"

  sku = "Standard"

}

resource "azurerm\_lb" "testLoadBalancer" {

  name                = "TestLoadBalancer"

  location            = azurerm\_resource\_group.rg.location

  resource\_group\_name = azurerm\_resource\_group.rg.name

  sku = "Standard"

  frontend\_ip\_configuration {

    name                 = "PublicIPAddress"

    public\_ip\_address\_id = azurerm\_public\_ip.testLoadBalancer-ip.id

  }

}

resource "azurerm\_lb\_backend\_address\_pool" "testLoadBalancer-backendPool" {

  loadbalancer\_id = azurerm\_lb.testLoadBalancer.id

  name            = "BackEndAddressPool"

}

resource "azurerm\_lb\_backend\_address\_pool\_address" "example-1" {

  name                                = "address1"

  backend\_address\_pool\_id             = azurerm\_lb\_backend\_address\_pool.testLoadBalancer-backendPool.id

  virtual\_network\_id = azurerm\_virtual\_network.myNetwork.id

  ip\_address = azurerm\_network\_interface.myNICs[0].private\_ip\_address

}

resource "azurerm\_lb\_backend\_address\_pool\_address" "example-2" {

  name                                = "address2"

  backend\_address\_pool\_id             = azurerm\_lb\_backend\_address\_pool.testLoadBalancer-backendPool.id

  virtual\_network\_id = azurerm\_virtual\_network.myNetwork.id

  ip\_address = azurerm\_network\_interface.myNICs[1].private\_ip\_address

}

resource "azurerm\_lb\_probe" "ProbeA" {

  loadbalancer\_id = azurerm\_lb.testLoadBalancer.id

  name            = "ssh-running-probe"

  port            = 22

}

resource "azurerm\_lb\_rule" "WebRule" {

  loadbalancer\_id                = azurerm\_lb.testLoadBalancer.id

  name                           = "LBRule"

  protocol                       = "Tcp"

  frontend\_port                  = 80

  backend\_port                   = 80

  frontend\_ip\_configuration\_name = "PublicIPAddress"

  backend\_address\_pool\_ids = [azurerm\_lb\_backend\_address\_pool.testLoadBalancer-backendPool.id]

  probe\_id = azurerm\_lb\_probe.ProbeA.id

}

**output** loadbalancer\_public\_ip {

value = **azurerm\_public\_ip.testLoadBalancer-ip.ip\_address**

depends\_on = [

azurerm\_linux\_virtual\_machine.my\_terraform\_vm

]

}

**Terraform Commands**

* terraform init
* terraform plan -out=tfplan
* terraform apply tfplan --auto-approve
* terraform destroy --auto-approve