CDW Documentation

Bicep Terraform Comparison

Bicep Terraform Comparison

I worked on two scripts, both doing the same thing, one in Bicep and one in Terraform.

They both produced the exact same results. The biggest difference in the process is that bicep is more integrated with the Azure command line and needs fewer commands to get the same thing done. Terraform is easier to read through as Json is not the greatest for code review, although it's not difficult. Terraform is also better suited for multi-cloud support.

I was able to deploy the below main.bicep in one command:

```
az deployment group create --resource-group don-test-rg --template-file
main.bicep --parameters adminPassword="L3tm31n2025"
```

While the below main.tf for Terraform required 3 commands:

```
terraform init
terraform plan
terraform apply
```

main.bicep

```
param location string = resourceGroup().location
param adminUsername string = 'azureuser'
@secure()
param adminPassword string

var vnetName = 'myVnet'
var subnet1Name = 'subnet1'
var subnet2Name = 'subnet2'
var vm1Name = 'vm1'
var vm2Name = 'vm2'
var nsgName = 'webNsg'
var routeTableName = 'myRouteTable'

resource vnet 'Microsoft.Network/virtualNetworks@2023-04-01' = {
  name: vnetName
  location: location
  properties: {
    addressSpace: {
      addressPrefixes: ['10.0.0.0/16']
    }
    subnets: [
      {
        name: subnet1Name
        properties: {
          addressPrefix: '10.0.0.0/24'
          networkSecurityGroup: {
            id: nsg.id
```

```
          }
          routeTable: {
            id: routeTable.id
          }
        }
      }
      {
        name: subnet2Name
        properties: {
          addressPrefix: '10.0.1.0/24'
          networkSecurityGroup: {
            id: nsg.id
          }
          routeTable: {
            id: routeTable.id
          }
        }
      }
    ]
  }
}

resource nsg 'Microsoft.Network/networkSecurityGroups@2023-04-01' = {
  name: nsgName
  location: location
  properties: {
    securityRules: [
      {
        name: 'allow-ssh'
        properties: {
          priority: 1001
          direction: 'Inbound'
          access: 'Allow'
          protocol: 'Tcp'
          sourcePortRange: '*'
          destinationPortRange: '22'
          sourceAddressPrefix: '*'
          destinationAddressPrefix: '*'
        }
      }
      {
        name: 'allow-http'
        properties: {
          priority: 1002
          direction: 'Inbound'
          access: 'Allow'
          protocol: 'Tcp'
          sourcePortRange: '*'
          destinationPortRange: '80'
          sourceAddressPrefix: '*'
          destinationAddressPrefix: '*'
```

```
        }
      }
    ]
  }
}

resource routeTable 'Microsoft.Network/routeTables@2023-04-01' = {
  name: routeTableName
  location: location
  properties: {}
}

resource nic1 'Microsoft.Network/networkInterfaces@2023-04-01' = {
  name: '${vm1Name}-nic'
  location: location
  properties: {
    ipConfigurations: [
      {
        name: 'ipconfig1'
        properties: {
          subnet: {
            id: vnet.properties.subnets[0].id
          }
          privateIPAllocationMethod: 'Dynamic'
        }
      }
    ]
  }
}

resource nic2 'Microsoft.Network/networkInterfaces@2023-04-01' = {
  name: '${vm2Name}-nic'
  location: location
  properties: {
    ipConfigurations: [
      {
        name: 'ipconfig2'
        properties: {
          subnet: {
            id: vnet.properties.subnets[1].id
          }
          privateIPAllocationMethod: 'Dynamic'
        }
      }
    ]
  }
}

resource vm1 'Microsoft.Compute/virtualMachines@2023-07-01' = {
  name: vm1Name
  location: location
```

```
  properties: {
    hardwareProfile: {
      vmSize: 'Standard_B1s'
    }
    osProfile: {
      computerName: vm1Name
      adminUsername: adminUsername
      adminPassword: adminPassword
      linuxConfiguration: {
        disablePasswordAuthentication: false
      }
      customData: base64('''
        #!/bin/bash
        apt update
        apt install -y apache2
        systemctl enable apache2
        systemctl start apache2
      ''')
    }
    storageProfile: {
      imageReference: {
        publisher: 'Canonical'
        offer: '0001-com-ubuntu-server-focal'
        sku: '20_04-lts'
        version: 'latest'
      }
      osDisk: {
        createOption: 'FromImage'
        managedDisk: {
          storageAccountType: 'Standard_LRS'
        }
      }
    }
    networkProfile: {
      networkInterfaces: [
        {
          id: nic1.id
        }
      ]
    }
  }
}

resource vm2 'Microsoft.Compute/virtualMachines@2023-07-01' = {
  name: vm2Name
  location: location
  properties: {
    hardwareProfile: {
      vmSize: 'Standard_B1s'
    }
    osProfile: {
```

```
      computerName: vm2Name
      adminUsername: adminUsername
      adminPassword: adminPassword
      linuxConfiguration: {
        disablePasswordAuthentication: false
      }
      customData: base64('''
        #!/bin/bash
        apt update
        apt install -y apache2
        systemctl enable apache2
        systemctl start apache2
      ''')
    }
    storageProfile: {
      imageReference: {
        publisher: 'Canonical'
        offer: '0001-com-ubuntu-server-focal'
        sku: '20_04-lts'
        version: 'latest'
      }
      osDisk: {
        createOption: 'FromImage'
        managedDisk: {
          storageAccountType: 'Standard_LRS'
        }
      }
    }
    networkProfile: {
      networkInterfaces: [
        {
          id: nic2.id
        }
      ]
    }
  }
}
```

vm.bicep

```
param vmName string
param subnetId string
param location string
param adminUsername string
@secure()
param adminPassword string

resource nic 'Microsoft.Network/networkInterfaces@2023-04-01' = {
  name: '${vmName}-nic'
  location: location
  properties: {
```

```
    ipConfigurations: [
      {
        name: 'ipconfig1'
        properties: {
          subnet: {
            id: subnetId
          }
          privateIPAllocationMethod: 'Dynamic'
        }
      }
    ]
  }
}

resource vm 'Microsoft.Compute/virtualMachines@2023-07-01' = {
  name: vmName
  location: location
  properties: {
    hardwareProfile: {
      vmSize: 'Standard_B1s'
    }
    osProfile: {
      computerName: vmName
      adminUsername: adminUsername
      adminPassword: adminPassword
      linuxConfiguration: {
        disablePasswordAuthentication: false
      }
      customData: base64('
        #!/bin/bash
        apt update
        apt install -y apache2
        systemctl enable apache2
        systemctl start apache2
      ')
    }
    storageProfile: {
      imageReference: {
        publisher: 'Canonical'
        offer: 'UbuntuServer'
        sku: '20_04-lts'
        version: 'latest'
      }
      osDisk: {
        createOption: 'FromImage'
        managedDisk: {
          storageAccountType: 'Standard_LRS'
        }
      }
    }
    networkProfile: {
```

```
      networkInterfaces: [
        {
          id: nic.id
        }
      ]
    }
  }
}
```

main.tf

```
terraform {
  required_providers {
    azurerm = {
      source  = "hashicorp/azurerm"
      version = ">= 3.63.0"
    }
  }
}

provider "azurerm" {
  features {}
  subscription_id = "baa29726-b3e6-4910-bb9b-b585c655322c"
}

variable "location" {
  default = "eastus"
}



resource "azurerm_virtual_network" "vnet" {
  name                = "myVnet"
  address_space       = ["10.0.0.0/16"]
  location            = var.location
  resource_group_name = "don-test-rg"
}

resource "azurerm_network_security_group" "nsg" {
  name                = "webNsg"
  location            = var.location
  resource_group_name = "don-test-rg"

  security_rule {
    name                       = "allow-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                       = "allow-http"
    priority                   = 1002
    direction                  = "Inbound"
    access                     = "Allow"
    protocol                   = "Tcp"
    source_port_range          = "*"
    destination_port_range     = "80"
    source_address_prefix      = "*"
    destination_address_prefix = "*"
  }
}

resource "azurerm_route_table" "rt" {
  name                = "myRouteTable"
  location            = var.location
  resource_group_name = "don-test-rg"
}

resource "azurerm_subnet" "subnet1" {
  name                 = "subnet1"
  resource_group_name  = "don-test-rg"
  virtual_network_name = azurerm_virtual_network.vnet.name
  address_prefixes     = ["10.0.0.0/24"]
}

resource "azurerm_subnet" "subnet2" {
  name                 = "subnet2"
  resource_group_name  = "don-test-rg"
  virtual_network_name = azurerm_virtual_network.vnet.name
  address_prefixes     = ["10.0.1.0/24"]
}

resource "azurerm_subnet_network_security_group_association" "subnet1_nsg" {
  subnet_id                 = azurerm_subnet.subnet1.id
  network_security_group_id = azurerm_network_security_group.nsg.id
}

resource "azurerm_subnet_network_security_group_association" "subnet2_nsg" {
  subnet_id                 = azurerm_subnet.subnet2.id
  network_security_group_id = azurerm_network_security_group.nsg.id
}

resource "azurerm_subnet_route_table_association" "subnet1_route" {
  subnet_id      = azurerm_subnet.subnet1.id
  route_table_id = azurerm_route_table.rt.id
}
```

```
resource "azurerm_subnet_route_table_association" "subnet2_route" {
  subnet_id      = azurerm_subnet.subnet2.id
  route_table_id = azurerm_route_table.rt.id
}

resource "azurerm_network_interface" "nic1" {
  name                = "nic1"
  location            = var.location
  resource_group_name = "don-test-rg"
  ip_configuration {
    name                          = "ipconfig1"
    subnet_id                     = azurerm_subnet.subnet1.id
    private_ip_address_allocation = "Dynamic"
  }
}

resource "azurerm_network_interface" "nic2" {
  name                = "nic2"
  location            = var.location
  resource_group_name = "don-test-rg"
  ip_configuration {
    name                          = "ipconfig2"
    subnet_id                     = azurerm_subnet.subnet2.id
    private_ip_address_allocation = "Dynamic"
  }
}

resource "azurerm_linux_virtual_machine" "vm1" {
  name                  = "vm1"
  resource_group_name   = "don-test-rg"
  location              = var.location
  size                  = "Standard_B1s"
  admin_username        = "azureuser"
  admin_password        = "L3tm31n!2025"
  disable_password_authentication = false
  network_interface_ids = [azurerm_network_interface.nic1.id]

  os_disk {
    caching              = "ReadWrite"
    storage_account_type = "Standard_LRS"
    name                 = "vm1-osdisk"
  }

  source_image_reference {
    publisher = "Canonical"
    offer     = "0001-com-ubuntu-server-focal"
    sku       = "20_04-lts"
    version   = "latest"
  }

  custom_data = base64encode(<<EOF
```

```
#!/bin/bash
apt update
apt install -y apache2
systemctl enable apache2
systemctl start apache2
EOF
  )
}

resource "azurerm_linux_virtual_machine" "vm2" {
  name                = "vm2"
  resource_group_name   = "don-test-rg"
  location            = var.location
  size                = "Standard_B1s"
  admin_username      = "azureuser"
  admin_password      = "L3tm31n!2025"
  disable_password_authentication = false
  network_interface_ids = [azurerm_network_interface.nic2.id]

  os_disk {
    caching              = "ReadWrite"
    storage_account_type = "Standard_LRS"
    name                 = "vm2-osdisk"
  }

  source_image_reference {
    publisher = "Canonical"
    offer     = "0001-com-ubuntu-server-focal"
    sku       = "20_04-lts"
    version   = "latest"
  }

  custom_data = base64encode(<<EOF
#!/bin/bash
apt update
apt install -y apache2
systemctl enable apache2
systemctl start apache2
EOF
  )
}
```

Bonus to show how similar AWS and Azure are with Terraform:

main_aws.tf

```
provider "aws" {
  region = "us-east-1"
}

variable "vpc_cidr" {
```

```
  default = "10.0.0.0/16"
}

variable "subnet1_cidr" {
  default = "10.0.0.0/24"
}

variable "subnet2_cidr" {
  default = "10.0.1.0/24"
}

resource "aws_vpc" "main" {
  cidr_block = var.vpc_cidr
  tags = {
    Name = "main-vpc"
  }
}

resource "aws_subnet" "subnet1" {
  vpc_id     = aws_vpc.main.id
  cidr_block = var.subnet1_cidr
  availability_zone = "us-east-1a"
  tags = {
    Name = "subnet-1"
  }
}

resource "aws_subnet" "subnet2" {
  vpc_id     = aws_vpc.main.id
  cidr_block = var.subnet2_cidr
  availability_zone = "us-east-1b"
  tags = {
    Name = "subnet-2"
  }
}

resource "aws_internet_gateway" "gw" {
  vpc_id = aws_vpc.main.id
}

resource "aws_route_table" "rt" {
  vpc_id = aws_vpc.main.id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.gw.id
  }
}

resource "aws_route_table_association" "a1" {
  subnet_id      = aws_subnet.subnet1.id
```

```
  route_table_id = aws_route_table.rt.id
}

resource "aws_route_table_association" "a2" {
  subnet_id      = aws_subnet.subnet2.id
  route_table_id = aws_route_table.rt.id
}

resource "aws_security_group" "web_sg" {
  name        = "web-sg"
  description = "Allow SSH and HTTP"
  vpc_id      = aws_vpc.main.id

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

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

  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
}

resource "aws_instance" "vm1" {
  ami                         = "ami-xxxxxxxxxxxxxxxxx" # Replace with valid
Ubuntu AMI
  instance_type               = "t2.micro"
  subnet_id                   = aws_subnet.subnet1.id
  vpc_security_group_ids      = [aws_security_group.web_sg.id]
  key_name                    = "my-key" # Replace with your key pair name
  associate_public_ip_address = true

  user_data = <<-EOF
              #!/bin/bash
              apt update
              apt install -y apache2
              systemctl enable apache2
              systemctl start apache2
              EOF
```

```
  tags = {
    Name = "vm1"
  }
}

resource "aws_instance" "vm2" {
  ami                         = "ami-xxxxxxxxxxxxxxxxx" # Replace with valid
Ubuntu AMI
  instance_type               = "t2.micro"
  subnet_id                   = aws_subnet.subnet2.id
  vpc_security_group_ids      = [aws_security_group.web_sg.id]
  key_name                    = "my-key" # Replace with your key pair name
  associate_public_ip_address = true

  user_data = <<-EOF
              #!/bin/bash
              apt update
              apt install -y apache2
              systemctl enable apache2
              systemctl start apache2
              EOF

  tags = {
    Name = "vm2"
  }
}
```

AI Knowledge