Create a Linux VM with infrastructure in Azure using Terraform

12/3/2020 • 8 minutes to read • Edit Online

Terraform allows you to define and create complete infrastructure deployments in Azure. You build Terraform templates in a human-readable format that create and configure Azure resources in a consistent, reproducible manner. This article shows you how to create a complete Linux environment and supporting resources with Terraform. You can also learn how to install and configure Terraform.

Prerequisites

- **Azure subscription**: If you don't have an Azure subscription, create a free account before you begin.

Create Azure connection and resource group

Let's go through each section of a Terraform template. You can also see the full version of the Terraform template that you can copy and paste.

The `provider` section tells Terraform to use an Azure provider. To get values for `subscription_id`, `client_id`, `client_secret`, and `tenant_id`, see Install and configure Terraform.

> **TIP**
> If you create environment variables for the values or are using the Azure Cloud Shell Bash experience , you don't need to include the variable declarations in this section.

```
provider "azurerm" {
    # The "feature" block is required for AzureRM provider 2.x.
    # If you're using version 1.x, the "features" block is not allowed.
    version = "~>2.0"
    features {}
}
```

The following section creates a resource group named `myResourceGroup` in the `eastus` location:

```
resource "azurerm_resource_group" "myterraformgroup" {
    name     = "myResourceGroup"
    location = "eastus"

    tags = {
        environment = "Terraform Demo"
    }
}
```

In additional sections, you reference the resource group with `azurerm_resource_group.myterraformgroup.name`.

Create virtual network

The following section creates a virtual network named `myVnet` in the `10.0.0.0/16` address space:

```
resource "azurerm_virtual_network" "myterraformnetwork" {
    name                = "myVnet"
    address_space       = ["10.0.0.0/16"]
    location            = "eastus"
    resource_group_name = azurerm_resource_group.myterraformgroup.name

    tags = {
        environment = "Terraform Demo"
    }
}
```

The following section creates a subnet named `mySubnet` in the `myVnet` virtual network:

```
resource "azurerm_subnet" "myterraformsubnet" {
    name                 = "mySubnet"
    resource_group_name  = azurerm_resource_group.myterraformgroup.name
    virtual_network_name = azurerm_virtual_network.myterraformnetwork.name
    address_prefixes       = ["10.0.2.0/24"]
}
```

Create public IP address

To access resources across the Internet, create and assign a public IP address to your VM. The following section creates a public IP address named `myPublicIP`:

```
resource "azurerm_public_ip" "myterraformpublicip" {
    name                         = "myPublicIP"
    location                     = "eastus"
    resource_group_name          = azurerm_resource_group.myterraformgroup.name
    allocation_method            = "Dynamic"

    tags = {
        environment = "Terraform Demo"
    }
}
```

Create Network Security Group

Network Security Groups control the flow of network traffic in and out of your VM. The following section creates a network security group named `myNetworkSecurityGroup` and defines a rule to allow SSH traffic on TCP port 22:

```
resource "azurerm_network_security_group" "myterraformnsg" {
    name                = "myNetworkSecurityGroup"
    location            = "eastus"
    resource_group_name = azurerm_resource_group.myterraformgroup.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 = "*"
    }

    tags = {
        environment = "Terraform Demo"
    }
}
```

Create virtual network interface card

A virtual network interface card (NIC) connects your VM to a given virtual network, public IP address, and network security group. The following section in a Terraform template creates a virtual NIC named `myNIC` connected to the virtual networking resources you've created:

```
resource "azurerm_network_interface" "myterraformnic" {
    name                        = "myNIC"
    location                    = "eastus"
    resource_group_name         = azurerm_resource_group.myterraformgroup.name

    ip_configuration {
        name                          = "myNicConfiguration"
        subnet_id                     = azurerm_subnet.myterraformsubnet.id
        private_ip_address_allocation = "Dynamic"
        public_ip_address_id          = azurerm_public_ip.myterraformpublicip.id
    }

    tags = {
        environment = "Terraform Demo"
    }
}

# Connect the security group to the network interface
resource "azurerm_network_interface_security_group_association" "example" {
    network_interface_id      = azurerm_network_interface.myterraformnic.id
    network_security_group_id = azurerm_network_security_group.myterraformnsg.id
}
```

Create storage account for diagnostics

To store boot diagnostics for a VM, you need a storage account. These boot diagnostics can help you troubleshoot problems and monitor the status of your VM. The storage account you create is only to store the boot diagnostics data. As each storage account must have a unique name, the following section generates some random text:

```
resource "random_id" "randomId" {
    keepers = {
        # Generate a new ID only when a new resource group is defined
        resource_group = azurerm_resource_group.myterraformgroup.name
    }

    byte_length = 8
}
```

Now you can create a storage account. The following section creates a storage account, with the name based on the random text generated in the preceding step:

```
resource "azurerm_storage_account" "mystorageaccount" {
    name                        = "diag${random_id.randomId.hex}"
    resource_group_name         = azurerm_resource_group.myterraformgroup.name
    location                    = "eastus"
    account_replication_type    = "LRS"
    account_tier                = "Standard"

    tags = {
        environment = "Terraform Demo"
    }
}
```

Create virtual machine

The final step is to create a VM and use all the resources created. The following section creates a VM named `myVM` and attaches the virtual NIC named `myNIC`. The latest `Ubuntu 18.04-LTS` image is used, and a user named `azureuser` is created with password authentication disabled.

SSH key data is provided in the `ssh_keys` section. Provide a public SSH key in the `key_data` field.

```
resource "tls_private_key" "example_ssh" {
  algorithm = "RSA"
  rsa_bits = 4096
}

output "tls_private_key" { value = tls_private_key.example_ssh.private_key_pem }

resource "azurerm_linux_virtual_machine" "myterraformvm" {
    name                  = "myVM"
    location              = "eastus"
    resource_group_name   = azurerm_resource_group.myterraformgroup.name
    network_interface_ids = [azurerm_network_interface.myterraformnic.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"
    disable_password_authentication = true

    admin_ssh_key {
        username       = "azureuser"
        public_key     = tls_private_key.example_ssh.public_key_openssh
    }

    boot_diagnostics {
        storage_account_uri = azurerm_storage_account.mystorageaccount.primary_blob_endpoint
    }

    tags = {
        environment = "Terraform Demo"
    }
}
```

Complete Terraform script

To bring all these sections together and see Terraform in action, create a file called `terraform_azure.tf` and paste the following content:

```
# Configure the Microsoft Azure Provider
provider "azurerm" {
    # The "feature" block is required for AzureRM provider 2.x.
    # If you're using version 1.x, the "features" block is not allowed.
    version = "~>2.0"
    features {}
}

# Create a resource group if it doesn't exist
resource "azurerm_resource_group" "myterraformgroup" {
    name     = "myResourceGroup"
    location = "eastus"

    tags = {
        environment = "Terraform Demo"
```

```
        environment = "Terraform Demo"
    }
}

# Create virtual network
resource "azurerm_virtual_network" "myterraformnetwork" {
    name                = "myVnet"
    address_space       = ["10.0.0.0/16"]
    location            = "eastus"
    resource_group_name = azurerm_resource_group.myterraformgroup.name

    tags = {
        environment = "Terraform Demo"
    }
}

# Create subnet
resource "azurerm_subnet" "myterraformsubnet" {
    name                 = "mySubnet"
    resource_group_name  = azurerm_resource_group.myterraformgroup.name
    virtual_network_name = azurerm_virtual_network.myterraformnetwork.name
    address_prefixes       = ["10.0.1.0/24"]
}

# Create public IPs
resource "azurerm_public_ip" "myterraformpublicip" {
    name                         = "myPublicIP"
    location                     = "eastus"
    resource_group_name          = azurerm_resource_group.myterraformgroup.name
    allocation_method            = "Dynamic"

    tags = {
        environment = "Terraform Demo"
    }
}

# Create Network Security Group and rule
resource "azurerm_network_security_group" "myterraformnsg" {
    name                = "myNetworkSecurityGroup"
    location            = "eastus"
    resource_group_name = azurerm_resource_group.myterraformgroup.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 = "*"
    }

    tags = {
        environment = "Terraform Demo"
    }
}

# Create network interface
resource "azurerm_network_interface" "myterraformnic" {
    name                      = "myNIC"
    location                  = "eastus"
    resource_group_name       = azurerm_resource_group.myterraformgroup.name

    ip_configuration {
        name                          = "myNicConfiguration"
        subnet_id                     = azurerm_subnet.myterraformsubnet.id
        private_ip_address_allocation = "Dynamic"
        public_ip_address_id          = azurerm_public_ip.myterraformpublicip.id
```

```
        public_ip_address_id          = azurerm_public_ip.myterraformpublicip.id
    }

    tags = {
        environment = "Terraform Demo"
    }
}

# Connect the security group to the network interface
resource "azurerm_network_interface_security_group_association" "example" {
    network_interface_id      = azurerm_network_interface.myterraformnic.id
    network_security_group_id = azurerm_network_security_group.myterraformnsg.id
}

# Generate random text for a unique storage account name
resource "random_id" "randomId" {
    keepers = {
        # Generate a new ID only when a new resource group is defined
        resource_group = azurerm_resource_group.myterraformgroup.name
    }

    byte_length = 8
}

# Create storage account for boot diagnostics
resource "azurerm_storage_account" "mystorageaccount" {
    name                        = "diag${random_id.randomId.hex}"
    resource_group_name         = azurerm_resource_group.myterraformgroup.name
    location                    = "eastus"
    account_tier                = "Standard"
    account_replication_type    = "LRS"

    tags = {
        environment = "Terraform Demo"
    }
}

# Create (and display) an SSH key
resource "tls_private_key" "example_ssh" {
  algorithm = "RSA"
  rsa_bits = 4096
}
output "tls_private_key" { value = tls_private_key.example_ssh.private_key_pem }

# Create virtual machine
resource "azurerm_linux_virtual_machine" "myterraformvm" {
    name                  = "myVM"
    location              = "eastus"
    resource_group_name   = azurerm_resource_group.myterraformgroup.name
    network_interface_ids = [azurerm_network_interface.myterraformnic.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"
    disable_password_authentication = true
```

```
    admin_ssh_key {
        username       = "azureuser"
        public_key     = tls_private_key.example_ssh.public_key_openssh
    }

    boot_diagnostics {
        storage_account_uri = azurerm_storage_account.mystorageaccount.primary_blob_endpoint
    }

    tags = {
        environment = "Terraform Demo"
    }
}
```

Build and deploy the infrastructure

With your Terraform template created, the first step is to initialize Terraform. This step ensures that Terraform has all the prerequisites to build your template in Azure.

```
terraform init
```

The next step is to have Terraform review and validate the template. This step compares the requested resources to the state information saved by Terraform and then outputs the planned execution. The Azure resources aren't created at this point.

```
terraform plan
```

After you execute the previous command, you should see something like the following screen:

```
Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan, but will not be
persisted to local or remote state storage.
...

Note: You didn't specify an "-out" parameter to save this plan, so when
"apply" is called, Terraform can't guarantee this is what will execute.
  + azurerm_resource_group.myterraform
      <snip>
  + azurerm_virtual_network.myterraformnetwork
      <snip>
  + azurerm_network_interface.myterraformnic
      <snip>
  + azurerm_network_security_group.myterraformnsg
      <snip>
  + azurerm_public_ip.myterraformpublicip
      <snip>
  + azurerm_subnet.myterraformsubnet
      <snip>
  + azurerm_virtual_machine.myterraformvm
      <snip>
Plan: 7 to add, 0 to change, 0 to destroy.
```

If everything looks correct and you're ready to build the infrastructure in Azure, apply the template in Terraform:

```
terraform apply
```

Once Terraform completes, your VM infrastructure is ready. Obtain the public IP address of your VM with az vm show:

```
az vm show --resource-group myResourceGroup --name myVM -d --query [publicIps] -o tsv
```

You can then SSH to your VM:

```
ssh azureuser@<publicIps>
```

Troubleshooting

For Terraform-specific support, use one of HashiCorp's community support channels to Terraform:

- Questions, use-cases, and useful patterns: Terraform section of the HashiCorp community portal
- Provider-related questions: Terraform Providers section of the HashiCorp community portal

Next steps

Learn more about using Terraform in Azure