Azure RHEL 8 VM Provisioning Guide using CLI and Terraform

Contents

Using Azure CLI

Create Resource Group

Create RHEL 8 VM

SSH Access

Using Terraform

File Structure

Terraform Configuration

SSH Key Handling

Apply Terraform

SSH Access

Security Note

1.⁠ ⁠Using Azure CLI

Step-by-Step Guide

1.1 Create a Resource Group

az group create --name myresourcegroup --location eastus

1.2 Create the RHEL 8 VM with SSH Key Generation

az vm create \

--resource-group myresourcegroup \

--name myRHEL8VM \

--location eastus \

--image RedHat:RHEL:8-LVM:latest \

--admin-username azureuser \

--generate-ssh-keys \

--size Standard\_B1s

This generates keys at ~/.ssh/id\_rsa (private) and ~/.ssh/id\_rsa.pub (public) if they don’t exist.

1.3 Get Public IP Address

az vm list-ip-addresses \

--resource-group myresourcegroup \

--name myRHEL8VM \

--output table

1.4 SSH into the VM

ssh -i ~/.ssh/id\_rsa azureuser@<Public\_IP\_Address>

2. Using Terraform

📁 File Structure

terraform-rhel8-vm/

├── main.tf

├── variables.tf

├── outputs.tf

├── generate\_ssh.sh

🧾 2.1 main.tf

provider "azurerm" {

features {}

}

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

name = "myresourcegroup"

location = "eastus"

}

resource "azurerm\_virtual\_network" "vnet" {

name = "myVnet"

address\_space = ["10.0.0.0/16"]

location = azurerm\_resource\_group.rg.location

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

}

resource "azurerm\_subnet" "subnet" {

name = "mySubnet"

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

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

address\_prefixes = ["10.0.1.0/24"]

}

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

name = "myNSG"

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 = "0.0.0.0/0"

destination\_address\_prefix = "\*"

}

}

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

name = "myNIC"

location = azurerm\_resource\_group.rg.location

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

ip\_configuration {

name = "internal"

subnet\_id = azurerm\_subnet.subnet.id

private\_ip\_address\_allocation = "Dynamic"

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

}

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

}

resource "azurerm\_public\_ip" "public\_ip" {

name = "myPublicIP"

location = azurerm\_resource\_group.rg.location

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

allocation\_method = "Dynamic"

}

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

name = "myRHEL8VM"

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

location = azurerm\_resource\_group.rg.location

size = "Standard\_B1s"

admin\_username = "azureuser"

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

admin\_ssh\_key {

username = "azureuser"

public\_key = file("${path.module}/id\_rsa.pub")

}

os\_disk {

caching = "ReadWrite"

storage\_account\_type = "Standard\_LRS"

}

source\_image\_reference {

publisher = "RedHat"

offer = "RHEL"

sku = "8-LVM"

version = "latest"

}

}

🔧 2.2 variables.tf (Optional for customization)

📤 2.3 outputs.tf

output "public\_ip\_address" {

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

}

⚙️ 2.4 generate\_ssh.sh

#!/bin/bash

if [ ! -f "id\_rsa" ]; then

ssh-keygen -t rsa -b 2048 -f id\_rsa -q -N ""

echo "SSH key pair generated."

else

echo "SSH key already exists. Skipping generation."

fi

▶️ 2.5 Run Terraform

cd terraform-rhel8-vm

chmod +x generate\_ssh.sh

./generate\_ssh.sh

terraform init

terraform apply -auto-approve

2.6 SSH into the VM (from output)

ssh -i id\_rsa azureuser@<Public\_IP\_from\_output>

Important Security Note

Using 0.0.0.0/0 in the NSG rule opens SSH access from the entire internet.

This should only be used for testing purposes.

❌Never use 0.0.0.0/0 in production.

✅ In production, restrict to your specific IP or trusted IP ranges (e.g., 203.0.113.0/24).