**PowerShell Script Documentation: Azure Resource Deployment**

**Overview:**

This PowerShell script automates the deployment of Azure resources for a virtual machine (VM) along with associated networking resources. It creates or checks the existence of a resource group, network security group (NSG), storage account, public IP address, virtual network (VNet), subnet, network interface (NIC), and finally provisions a VM.

**Prerequisites**

- Azure PowerShell module should be installed (`Az` module).

- Azure account with appropriate permissions to create resources.

Script Flow

1. Resource Group Creation/Validation:

- Checks if the specified resource group exists. If not, creates a new one in the specified location (`$resource\_group\_Location`).

2. Network Security Group Creation:

- Defines two inbound security rules for RDP and HTTP traffic.

- Creates a new NSG (`MYNSG`) in the specified resource group.

3. Storage Account Creation:

- Creates a new storage account in the specified resource group and location.

4. Public IP Address Creation:

- Creates a static public IP address in the specified resource group and location.

5. Virtual Network and Subnet Creation:

- Creates a new virtual network (`MyVNet`) with a subnet (`MySubnet`) in the specified resource group and location.

6. Network Interface Creation:

- Creates a new network interface (`MyNIC`) associated with the previously created resources (NSG, public IP, subnet).

7. Virtual Machine Deployment:

- Prompts the user for credentials.

- Defines parameters for VM creation including resource group, name, location, VNet, subnet, NSG, credentials, size, and image.

- Creates the VM in the specified resource group.

8. Wait for VM Creation:

- Monitors the VM creation process until completion.

**Code:**

|  |
| --- |
| $resource\_group\_name = "MyResourceGroup"  $resource\_group\_Location = "centralus"  # Check if the resource group exists, if not create it  $existingResourceGroup = Get-AzResourceGroup -Name $resource\_group\_name -ErrorAction SilentlyContinue  if (-not $existingResourceGroup) {  Write-Host "Creating resource group $resource\_group\_name..."  New-AzResourceGroup -Name $resource\_group\_name -Location $resource\_group\_Location  # Wait for the resource group creation  Write-Host "Waiting for the resource group to be created..."  do {  $existingResourceGroup = Get-AzResourceGroup -Name $resource\_group\_name -ErrorAction SilentlyContinue  Start-Sleep -Seconds 10  } while (-not $existingResourceGroup)  }  else {  Write-Host "Resource group $resource\_group\_name already exists. Proceeding..."  }  # Create a detailed network security group  $rule1 = New-AzNetworkSecurityRuleConfig -Name rdp-rule -Description "Allow RDP" `  -Access Allow -Protocol Tcp -Direction Inbound -Priority 300 -SourceAddressPrefix `  Internet -SourcePortRange \* -DestinationAddressPrefix \* -DestinationPortRange 3389  $rule2 = New-AzNetworkSecurityRuleConfig -Name web-rule -Description "Allow Http" `  -Access Allow -Protocol Tcp -Direction Inbound -Priority 400 -SourceAddressPrefix `  Internet -SourcePortRange \* -DestinationAddressPrefix \* -DestinationPortRange 80  $NSG = New-AzNetworkSecurityGroup -ResourceGroupName $resource\_group\_name -Location $resource\_group\_Location -Name "MYNSG" -SecurityRules $rule1,$rule2  Write-Host "Waiting for NSG to be created..."  do {  $NSG = Get-AzNetworkSecurityGroup -Name 'MYNSG' -ResourceGroupName $resource\_group\_name -ErrorAction SilentlyContinue  Start-Sleep -Seconds 10  } while (-not $NSG)  # Check if NSG was successfully created  if (-not $NSG) {  Write-Error "Failed to create NSG. Exiting..."  exit  }  # Create a storage account  $storage\_acc\_name = "jobansstorageacc"  $storage\_acc\_location = "centralus"  $storageacc = New-AzStorageAccount -ResourceGroupName $resource\_group\_name -Name $storage\_acc\_name -Location $storage\_acc\_location -SkuName "Standard\_LRS" -Kind "StorageV2"  Write-Host "Waiting for storage account to be created..."  do {  $storageacc = Get-AzStorageAccount -ResourceGroupName $resource\_group\_name -Name $storage\_acc\_name -ErrorAction SilentlyContinue  Start-Sleep -Seconds 10  } while (-not $storageacc)  # Check if storage account was successfully created  if (-not $storageacc) {  Write-Error "Failed to create storage account. Exiting..."  exit  }  # Create a Public IP address  $publicIp = New-AzPublicIpAddress -ResourceGroupName $resource\_group\_name -Name "MyPublicIP" -AllocationMethod Static -Location $resource\_group\_Location  # Wait for Public IP creation  Write-Host "Waiting for Public IP to be created..."  do {  $publicIp = Get-AzPublicIpAddress -ResourceGroupName $resource\_group\_name -Name "MyPublicIP" -ErrorAction SilentlyContinue  Start-Sleep -Seconds 10  } while (-not $publicIp)  # Check if Public IP was successfully created  if (-not $publicIp) {  Write-Error "Failed to create Public IP. Exiting..."  exit  }  # Create a subnet configuration  $subnetConfig = New-AzVirtualNetworkSubnetConfig -Name "MySubnet" -AddressPrefix "10.0.0.0/24"  # Create a virtual network  $vnet = New-AzVirtualNetwork -ResourceGroupName $resource\_group\_name -Location $resource\_group\_Location -Name "MyVNet" -AddressPrefix "10.0.0.0/16" -Subnet $subnetConfig  # Wait for Virtual Network creation  Write-Host "Waiting for Virtual Network to be created..."  do {  $vnet = Get-AzVirtualNetwork -ResourceGroupName $resource\_group\_name -Name "MyVNet" -ErrorAction SilentlyContinue  Start-Sleep -Seconds 10  } while (-not $vnet)  # Check if Virtual Network was successfully created  if (-not $vnet) {  Write-Error "Failed to create Virtual Network. Exiting..."  exit  }  # Create a network interface and associate it with NSG, public IP, and subnet  $nic = New-AzNetworkInterface -Name "MyNIC" -ResourceGroupName $resource\_group\_name -Location $resource\_group\_Location -SubnetId $vnet.Subnets[0].Id -PublicIpAddressId $publicIp.Id -NetworkSecurityGroupId $NSG.Id  # Wait for NIC creation  Write-Host "Waiting for NIC to be created..."  do {  $nic = Get-AzNetworkInterface -Name "MyNIC" -ResourceGroupName $resource\_group\_name -ErrorAction SilentlyContinue  Start-Sleep -Seconds 10  } while (-not $nic)  # Check if NIC was successfully created  if (-not $nic) {  Write-Error "Failed to create NIC. Exiting..."  exit  }  # Create the VM configuration  $VM\_name = "jobans-vm"  $cred = Get-Credential -Message "Enter a username and password for the virtual machine."  $VM = New-AzVMConfig -VMName $VM\_name -VMSize 'Standard\_DS1\_v2'  $VM = Set-AzVMOperatingSystem -VM $VM -Windows -ComputerName $VM\_name -Credential $cred -ProvisionVMAgent -EnableAutoUpdate  $VM = Add-AzVMNetworkInterface -VM $VM -Id $nic.Id  # Create the OS disk  $VM = Set-AzVMOSDisk -VM $VM -Name "osdisk1" -CreateOption FromImage -Windows  # Create the VM  $GETVM=New-AzVM -ResourceGroupName $resource\_group\_name -Location $resource\_group\_Location -VM $VirtualMachine  # Wait for NIC creation  Write-Host "Waiting for VM to be created..."  do {  $GETVM = Get-AzVM -Name $VM\_name -ResourceGroupName $resource\_group\_name -ErrorAction SilentlyContinue  Start-Sleep -Seconds 10  } while (-not $GETVM)  # Check if NIC was successfully created  if (-not $GETVM) {  Write-Error "Failed to create VM. Exiting..."  exit  }  Write-Host "All resources created Successfully" |

**Usage**

1. Ensure Azure PowerShell module is installed.

2. Copy the script into a PowerShell environment or editor.

3. Update variables `$resource\_group\_name` and `$resource\_group\_Location` with desired values.

4. Execute the script.

5. Follow the prompts to enter credentials for the VM.

6. Monitor the script execution for any errors or warnings.

Demo:

Run the script

A screenshot of a computer

Description automatically generated

Verify that resources have been created

A screenshot of a computer

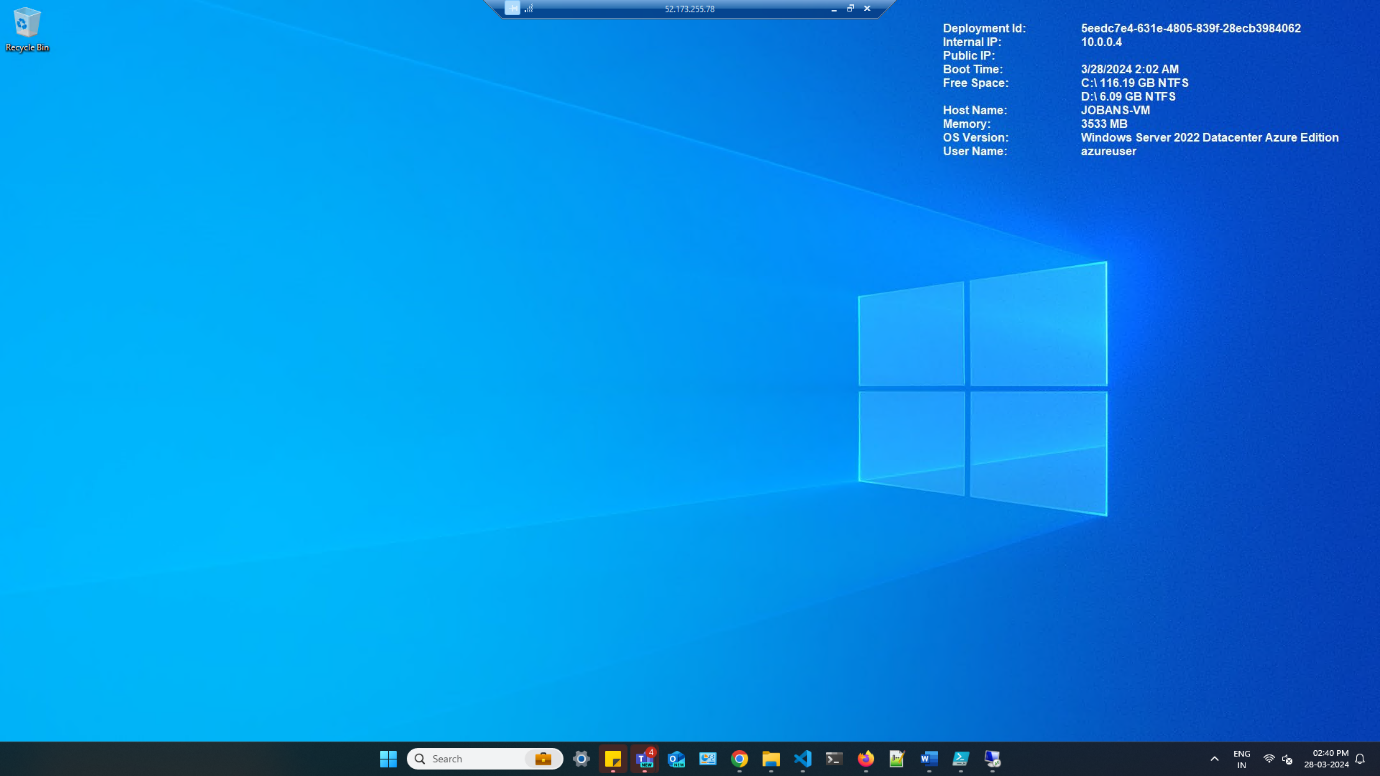
Description automatically generated

Check the result

A blue screen with white text

Description automatically generated

Connect to the vm using rdp



check the details of vm

A screenshot of a computer

Description automatically generated

**Notes**

- The script contains wait loops to ensure the completion of resource creation operations.

- Error handling is implemented to exit the script in case of failures during resource creation.

- Users may customize the script by modifying variables or adding additional resource creation steps as needed.