# Import the needed credential and management objects from the libraries.

import os

from datetime import datetime

from azure.identity import ClientSecretCredential

#from azure.identity import AzureCliCredential

from azure.mgmt.compute import ComputeManagementClient

from azure.mgmt.network import NetworkManagementClient

from azure.mgmt.resource import ResourceManagementClient

from azure.identity import DefaultAzureCredential

from azure.mgmt.network.v2017\_03\_01.models import NetworkSecurityGroup

from azure.mgmt.network.v2017\_03\_01.models import SecurityRule

print("Provisioning a virtual machine...some operations might take a \

minute or two.")

dttime = f'{datetime.now():%d-%H-%M-%S%z}'

# Step 1: Credential Management

#----------------------------------------------

# Actual ids have not been provided here, during assignments provision of the ids is a must.

tenantid = '<tenant-id>’

subscription\_id = ‘<subscription\_id >'

clientid = ‘<clientid >'

secretkey = ‘<secretkey>'

credential = ClientSecretCredential(

tenant\_id=tenantid, client\_id=clientid,

client\_secret=secretkey)

# Step 2: Provision a resource group

#-----------------------------------------------------------------------------

# Obtain the management object for resources, using the credentials

# from the CLI login.

resource\_client = ResourceManagementClient(credential, subscription\_id)

# Constants we need in multiple places: the resource group name and

# the region in which we provision resources. You can change these

# values however you want.

RESOURCE\_GROUP\_NAME = "RG"

SECURITY\_RULE\_NAME = "SecurityRuleAllowRDP"

LOCATION = "eastus"

NETWORK\_SECUIRITY\_GROUP = RESOURCE\_GROUP\_NAME + "-" + "NSG"

rg\_result = resource\_client.resource\_groups.create\_or\_update(

RESOURCE\_GROUP\_NAME, {"location": LOCATION}

)

print(f"Provisioned Resource Group {RESOURCE\_GROUP\_NAME}")

#Step-3 - Create Network security group

network\_client = NetworkManagementClient(credential,subscription\_id)

poller = network\_client.network\_security\_groups.begin\_create\_or\_update(

resource\_group\_name=RESOURCE\_GROUP\_NAME,

network\_security\_group\_name=NETWORK\_SECUIRITY\_GROUP,

parameters={"location": LOCATION,

"properties": {

"securityRules": [

{

"name": "rule1",

"properties": {

"access": "Allow",

"destinationAddressPrefix": "\*",

"destinationPortRange": "3389",

"direction": "Inbound",

"priority": 100,

"protocol": "TCP",

"sourceAddressPrefix": "\*",

"sourcePortRange": "\*",

},

}

]

},

},

)

nsg\_result = poller.result();

print(f"Provisioned Network Security Group {NETWORK\_SECUIRITY\_GROUP}")

# Step-4 Provision the VNET

#----------------------------

# Obtain the management object for networks

IP\_NAME = "ip" + "-" + dttime

IP\_CONFIG\_NAME = "ip-config" + "-" + dttime

NIC\_NAME = "nic" + "-" + dttime

VNET\_NAME = "VNET" + "-" + dttime

SUBNET\_NAME = "SUBNET" + "-" + dttime

## Provision the virtual network and wait for completion

poller = network\_client.virtual\_networks.begin\_create\_or\_update(

RESOURCE\_GROUP\_NAME,

VNET\_NAME,

{

"location": LOCATION,

"address\_space": {"address\_prefixes": ["10.0.0.0/16"]},

}

)

vnet\_result = poller.result()

#

print(

f"Provisioned virtual network {vnet\_result.name} with address \

#prefixes {vnet\_result.address\_space.address\_prefixes}"

)

#

## Step 5: Provision the subnet and wait for completion

#-------------------------------------------------------

poller = network\_client.subnets.begin\_create\_or\_update(

RESOURCE\_GROUP\_NAME,

VNET\_NAME,

SUBNET\_NAME,

{

"address\_prefix": "10.0.0.0/21",

},

)

subnet\_result = poller.result()

print(

f"Provisioned virtual subnet {subnet\_result.name} with address \

prefix {subnet\_result.address\_prefix}"

)

# Step 6: Provision an IP address and wait for completion

#--------------------------------------------------------

poller = network\_client.public\_ip\_addresses.begin\_create\_or\_update(

RESOURCE\_GROUP\_NAME,

IP\_NAME,

{

"location": LOCATION,

"sku": {"name": "Standard"},

"public\_ip\_allocation\_method": "Static",

"public\_ip\_address\_version": "IPV4",

},

)

ip\_address\_result = poller.result()

print(

f"Provisioned public IP address {ip\_address\_result.name} \

with address {ip\_address\_result.ip\_address}"

)

# Step 7: Provision the network interface client

#-------------------------------------------------

poller = network\_client.network\_interfaces.begin\_create\_or\_update(

RESOURCE\_GROUP\_NAME,

NIC\_NAME,

{

"location": LOCATION,

"ip\_configurations": [

{

"name": IP\_CONFIG\_NAME,

"subnet": {"id": subnet\_result.id},

"public\_ip\_address": {"id": ip\_address\_result.id},

"networkSecurityGroup": nsg\_result.id,

}

],

},

)

nic\_result = poller.result()

# Step 8: Provision the VM

#-------------------------------------------------

# Provision the VM specifying only minimal arguments, which defaults

# to an Ubuntu 18.04 VM on a Standard DS1 v2 plan with a public IP address

# and a default virtual network/subnet.

VM\_NAME = "A" + "-" + dttime

USERNAME = "azureuser"

PASSWORD = "Admin123456#"

print( f"Provisioning virtual machine {VM\_NAME}; this operation might \

take a few minutes."

)

# Obtain the management object for virtual machines

compute\_client = ComputeManagementClient(credential, subscription\_id)

poller = compute\_client.virtual\_machines.begin\_create\_or\_update(

RESOURCE\_GROUP\_NAME,

VM\_NAME,

{ "location": LOCATION,

"storage\_profile": {

"imageReference": {

"publisher": "MicrosoftWindowsServer",

"offer": "WindowsServer",

"sku": "2022-datacenter-azure-edition",

"version": "latest",

"exactVersion": "20348.1547.230207",

#"publisher": "Canonical",

#"offer": "UbuntuServer",

#"sku": "16.04.0-LTS",

#"version": "latest",

}

},

"hardware\_profile": {"vm\_size": "Standard\_DS1\_v2"},

"os\_profile": {

"computer\_name": VM\_NAME,

"admin\_username": USERNAME,

"admin\_password": PASSWORD,

},

"network\_profile": {

"network\_interfaces": [

{

"id": nic\_result.id,

"networkSecurityGroup": nsg\_result.id,

},

]

},

},

)

vm\_result = poller.result()

print(f"Provisioned virtual machine {vm\_result.name}")