# How to create with Azure SDK for .NET a Virtual Machine with Windows Server and Visual Studio 2022 preinstalled

**NOTE:** for more information about VM with Azure SDK for .NET visit the URL

https://github.com/Azure/azure-sdk-for-net/blob/main/sdk/compute/Azure.ResourceManager.Compute/samples/Sample2_ManagingVirtualMachines.md

## 0. Prerequisites

Install .NET 8 SDK: https://dotnet.microsoft.com/en-us/download/dotnet/8.0

Install Azure CLI: https://learn.microsoft.com/en-us/cli/azure/install-azure-cli

Install VSCode: https://code.visualstudio.com/download

## 1. Create a new C# console .Net 8 application in VSCode

Open VSCode and run the command:

```
dotnet new console --framework net8.0
```

## 2. Load the Azure SDK libraries.

From the Nuget web page copy the commands to load the libraries: https://www.nuget.org/

Run these commands to load the libraries:

```
dotnet add package Azure.Identity --version 1.10.4
dotnet add package Azure.ResourceManager --version 1.9.0
dotnet add package Azure.ResourceManager.Network --version 1.6.0
dotnet add package Azure.ResourceManager.Compute --version 1.2.1
```

Now run the command:

```
dotnet restore
```

# 3. Input the C# source code.

```
using System;
using System.Threading.Tasks;
using Azure;
using Azure.Core;
using Azure.Identity;
using Azure.ResourceManager;
using Azure.ResourceManager.Network.Models;
using Azure.ResourceManager.Network;
using Azure.ResourceManager.Resources;
using Azure.ResourceManager.Resources.Models;
using Azure.ResourceManager.Compute;
using Azure.ResourceManager.Compute.Models;


ArmClient armClient = new ArmClient(new DefaultAzureCredential());
SubscriptionResource subscription = await armClient.GetDefaultSubscriptionAsync();

ResourceGroupCollection rgCollection = subscription.GetResourceGroups();
// With the collection, we can create a new resource group with an specific name
string rgName = "myRgName";
AzureLocation location = AzureLocation.WestEurope;
ResourceGroupResource resourceGroup = await rgCollection.CreateOrUpdate(WaitUntil.Started, rgN

//------------------------------------------------------------------------------------------

PublicIPAddressCollection publicIPAddressCollection = resourceGroup.GetPublicIPAddresses();
string publicIPAddressName = "20.61.0.157";
PublicIPAddressData publicIPInput = new PublicIPAddressData()
{
    Location = resourceGroup.Data.Location,
    PublicIPAllocationMethod = NetworkIPAllocationMethod.Dynamic,
    DnsSettings = new PublicIPAddressDnsSettings()
    {
        DomainNameLabel = "mydomain12319741999"
    }
};
PublicIPAddressResource publicIPAddress = await publicIPAddressCollection.CreateOrUpdate(WaitU

VirtualNetworkCollection virtualNetworkCollection = resourceGroup.GetVirtualNetworks();

string vnetName = "myVnet";

// Use the same location as the resource group
VirtualNetworkData input = new VirtualNetworkData()
{
    Location = resourceGroup.Data.Location,
    AddressPrefixes = { "10.0.0.0/16", },
    DhcpOptionsDnsServers = { "8.8.8.8", "8.8.4.4", "10.1.1.1", "10.1.2.4" },
    Subnets = { new SubnetData() { Name = "mySubnet", AddressPrefix = "10.0.1.0/24", } }
```

```
};

VirtualNetworkResource vnet = await virtualNetworkCollection.CreateOrUpdate(WaitUntil.Complete

VirtualNetworkCollection virtualNetworkCollection1 = resourceGroup.GetVirtualNetworks();

VirtualNetworkResource virtualNetwork1 = await virtualNetworkCollection1.GetAsync("myVnet");
Console.WriteLine(virtualNetwork1.Data.Name);

//------------------------------------------------------------------------------------------

NetworkInterfaceCollection networkInterfaceCollection = resourceGroup.GetNetworkInterfaces();
string networkInterfaceName = "myNetworkInterface";
NetworkInterfaceData networkInterfaceInput = new NetworkInterfaceData()
{
    Location = resourceGroup.Data.Location,
    IPConfigurations = {
        new NetworkInterfaceIPConfigurationData()
        {
            Name = "ipConfig",
            PrivateIPAllocationMethod = NetworkIPAllocationMethod.Dynamic,
            PublicIPAddress = new PublicIPAddressData()
            {
                Id = publicIPAddress.Id
            },
            Subnet = new SubnetData()
            {
                // use the virtual network just created
                Id = virtualNetwork1.Data.Subnets[0].Id
            }
        }
    }
};

// Create NSG rule for SSH
NetworkSecurityGroupCollection nsgCollection = resourceGroup.GetNetworkSecurityGroups();
string nsgName = "myNetworkSecurityGroup";
NetworkSecurityGroupData nsgInput = new NetworkSecurityGroupData()
{
    Location = resourceGroup.Data.Location,
    SecurityRules =
            {
                new SecurityRuleData()
                {
                    Name = "AllowSSH",
                    Priority = 100,
                    Access = SecurityRuleAccess.Allow,
                    Direction = SecurityRuleDirection.Inbound,
                    Protocol = SecurityRuleProtocol.Tcp,
                    SourceAddressPrefix = "*",
                    SourcePortRange = "*",
                    DestinationAddressPrefix = "*",
                    DestinationPortRange = "22", // SSH port
```

```
                },
                new SecurityRuleData()
                {
                    Name = "AllowHTTP",
                    Priority = 110,
                    Access = SecurityRuleAccess.Allow,
                    Direction = SecurityRuleDirection.Outbound,
                    Protocol = SecurityRuleProtocol.Tcp,
                    SourceAddressPrefix = "*",
                    SourcePortRange = "*",
                    DestinationAddressPrefix = "*",
                    DestinationPortRange = "80", // HTTP port
                },
                new SecurityRuleData()
                {
                    Name = "AllowHTTPS",
                    Priority = 120,
                    Access = SecurityRuleAccess.Allow,
                    Direction = SecurityRuleDirection.Outbound,
                    Protocol = SecurityRuleProtocol.Tcp,
                    SourceAddressPrefix = "*",
                    SourcePortRange = "*",
                    DestinationAddressPrefix = "*",
                    DestinationPortRange = "443", // HTTPS port
                },
                new SecurityRuleData()
                {
                    Name = "AllowRDP",
                    Priority = 130,
                    Access = SecurityRuleAccess.Allow,
                    Direction = SecurityRuleDirection.Inbound,
                    Protocol = SecurityRuleProtocol.Tcp,
                    SourceAddressPrefix = "*",
                    SourcePortRange = "*",
                    DestinationAddressPrefix = "*",
                    DestinationPortRange = "3389", // RDP port
                }
            }
};

NetworkSecurityGroupResource nsg = await nsgCollection.CreateOrUpdate(WaitUntil.Completed, nsg

// Associate NSG with the network interface
networkInterfaceInput.NetworkSecurityGroup = new NetworkSecurityGroupData()
{
    Id = nsg.Id
};

NetworkInterfaceResource networkInterface = await networkInterfaceCollection.CreateOrUpdate(Wa


//------------------------------------------------------------------------------------------
// Now we get the virtual machine collection from the resource group
```

```
VirtualMachineCollection vmCollection = resourceGroup.GetVirtualMachines();
// Use the same location as the resource group
string vmName = "myVM";
VirtualMachineData input2 = new VirtualMachineData(resourceGroup.Data.Location)
{
    HardwareProfile = new VirtualMachineHardwareProfile()
    {
        //VmSize = VirtualMachineSizeType.StandardF2
        VmSize = VirtualMachineSizeType.StandardE2SV3
    },
    OSProfile = new VirtualMachineOSProfile()
    {
        AdminUsername = "luiscocoenrique1999",
        AdminPassword = "Luiscoco23421",
        ComputerName = "myVM",
        WindowsConfiguration = new WindowsConfiguration()
        {
            EnableAutomaticUpdates = true,
            ProvisionVmAgent = true,
        }
    },
    NetworkProfile = new VirtualMachineNetworkProfile()
    {
        NetworkInterfaces =
        {
            new VirtualMachineNetworkInterfaceReference()
            {
                Id = new ResourceIdentifier("/subscriptions/846901e6-da09-45c8-98ca-7cca2353ff
                Primary = true,
            }
        }
    },
    StorageProfile = new VirtualMachineStorageProfile()
    {
        OSDisk = new VirtualMachineOSDisk(DiskCreateOptionType.FromImage)
        {
            //OSType = SupportedOperatingSystemType.Linux,
            OSType = SupportedOperatingSystemType.Windows,
            Caching = CachingType.ReadWrite,
            ManagedDisk = new VirtualMachineManagedDisk()
            {
                StorageAccountType = StorageAccountType.StandardLrs
            }
        },
        ImageReference = new ImageReference()
        {
            Publisher = "microsoftvisualstudio",
            Offer = "visualstudio2022",
            Sku = "vs-2022-comm-latest-ws2022",
            Version = "latest",
        }
    },
    Priority = "Spot",
```

```
    EvictionPolicy = "Deallocate",
};
ArmOperation<VirtualMachineResource> lro = await vmCollection.CreateOrUpdateAsync(WaitUntil.Co
VirtualMachineResource vm = lro.Value;
```

## 4. Build and run the application in VSCode

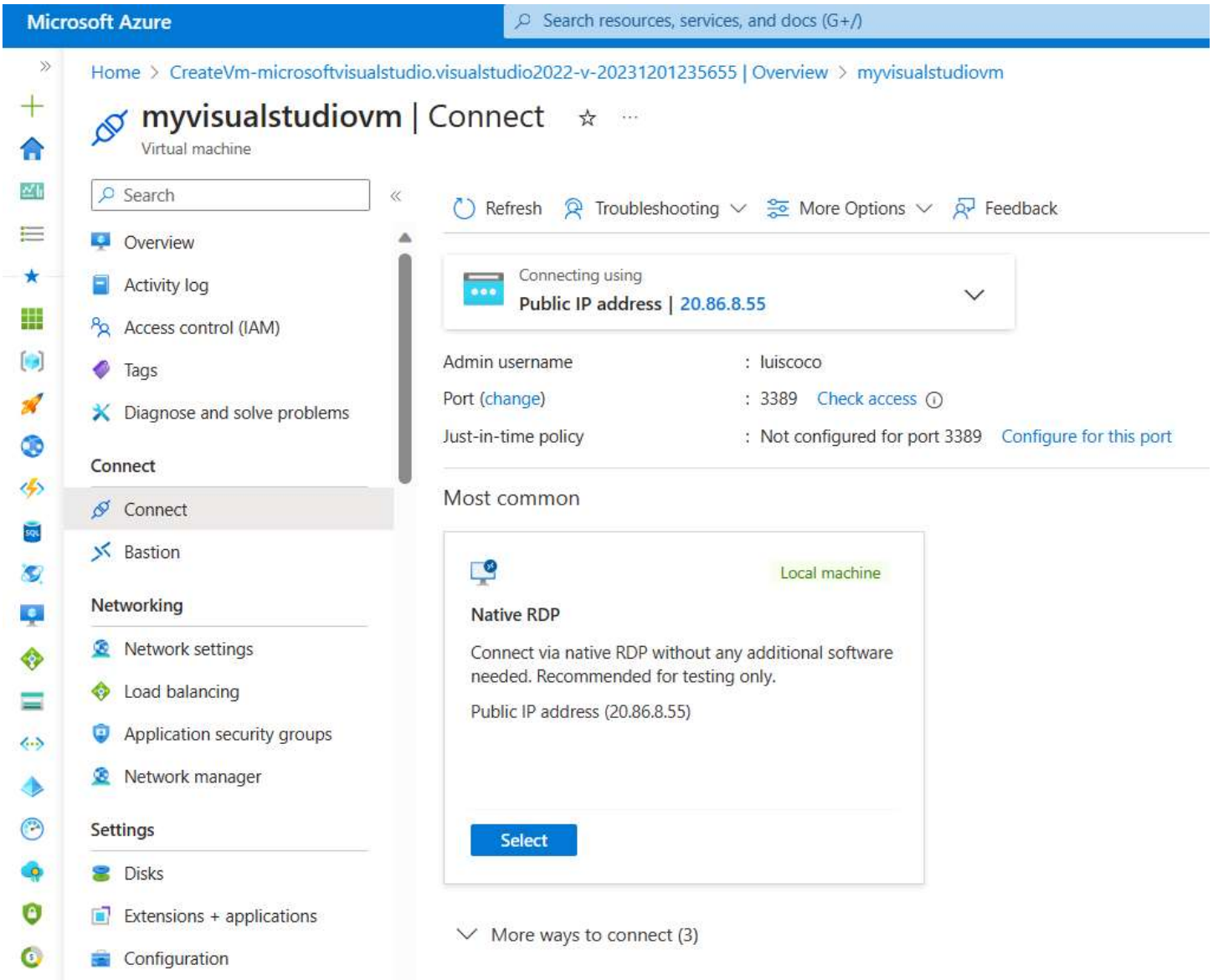
Type this command to build and run the application

```
dotnet run
```

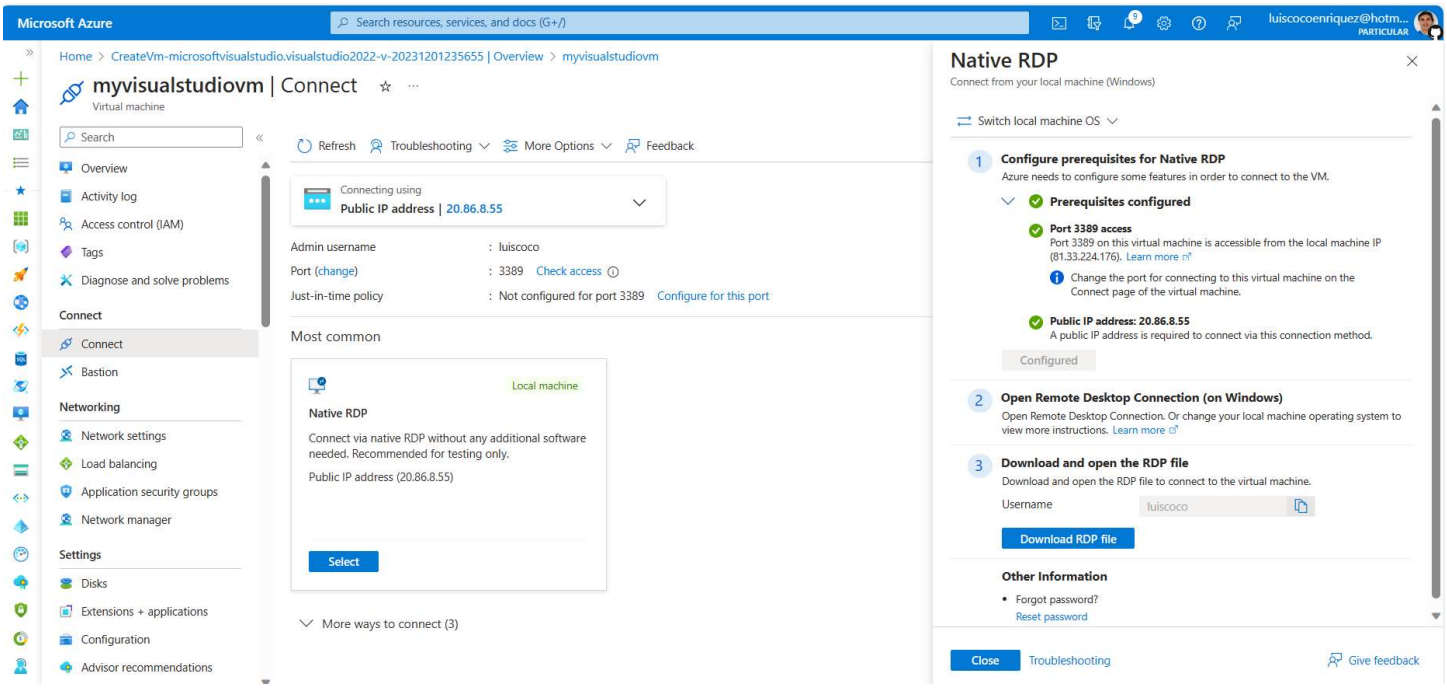
## 5. Connect to Azure portal and access to the VM

Go to the VM and download the RDP file in order to connect to the virtual machine.

Select Native RDP



## Download the RDP file




Double click on the RDP file and enter the **username** and **password** that we set in the application C# code:

```
username:luiscocoenrique1999
password:Luiscoco23421
```