

Azure Virtual Machine Provisioning using Pulumi

- We will provision the Azure Virtual Machine using Pulumi as an Infrastructure as Code.
 - We will deploy it in a custom Virtual Network for isolation.
 - We will SSH into the Virtual Machine, and install the docker.
 - Then, we will deploy the Nginx Container and try accessing it on the Web Browser.
-

Prerequisites

1. An Azure account.
 2. Azure CLI installed and configured with the appropriate Azure User or Service Principal.
 3. Pulumi Installed.
 4. Kubectl Installed.
-

Write Pulumi Configuration files

First, we will initiate and edit Pulumi configuration files for Azure resources using predefined Pulumi Library available on the internet.

Steps

1. Create a Pulumi Project directory.
2. Open the PowerShell.
3. Change the directory to the above-created Pulumi Project.
4. Run the `pulumi new azure-python` command to initialize the *pulumi*.
5. Provide the appropriate values to prompts such as *project-name*, *project-description*, *stack-name*, *toolchain*, *region-name*, etc.
6. This will generate some Pulumi files in this directory.
7. Now we will install predefined Pulumi modules.
8. Activate the `venv` by running `venv\Scripts\activate`.
9. Run `pip install git+https://github.com/sahilphule/pulumi.git` to install the modules.
10. Deactivate the `venv` by running `deactivate`.
11. Now open the directory in the preferred IDE.
12. Create *commons* folder
13. Inside the folder create *init.py* file.
14. Import the following in the *init.py* file:
 - `from inflection_zone_pulumi.modules.azure.resource_group import resource_group`
 - `from inflection_zone_pulumi.modules.azure.vnet import vnet`
 - `from inflection_zone_pulumi.modules.azure.virtual_machine import virtual_machine` \15. Click [code](#) for reference.
15. Definition of *init.py* is complete.
16. Now create the *values.py* file in the root folder of the above-created project directory.
17. Define the following values:

- `resource_group_properties`
- `vnet_properties`
- `virtual_machine_properties`

18. Click [code](#) for reference.

19. The definition of `values.py` is complete.

20. Now navigate to the `main.py` file present in the root folder of the above-created project directory.

21. Clear the sample code if present.

22. Import the following:

- `from commons import resource_group, vnet, virtual_machine`
- `values`

23. Define the following objects and pass the values & dependencies as an argument:

- `RESOURCE_GROUP`
- `VNET`
- `VM`

24. Click [code](#) for reference.

25. Definition of `main.py` is complete.

Provisioning the Infrastructure

Now we will provision the infrastructure by applying the above-created configuration files.

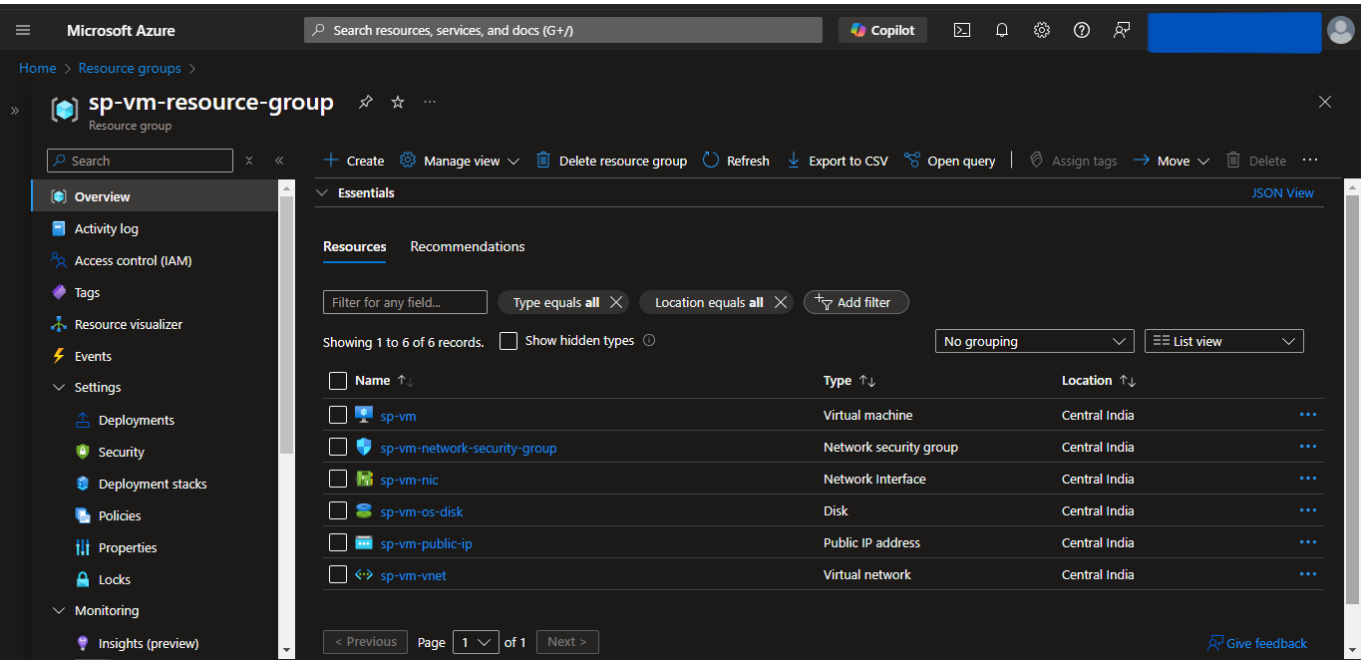
Ensure Azure CLI is configured with the appropriate Azure User or Service Principal.

Steps:

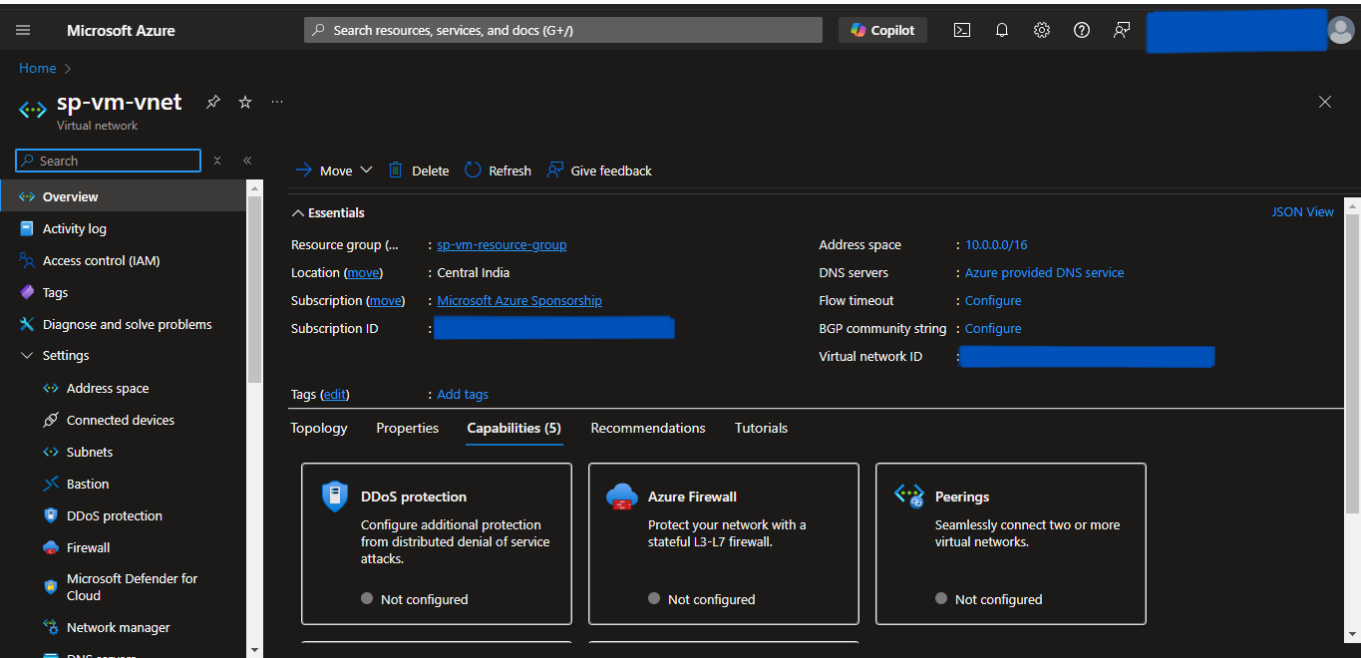
1. Open the PowerShell.
 2. Change the directory to the above-created Pulumi Project.
 3. Run the `pulumi up` command and if prompted, select `yes` to provision the infrastructure onto the Azure Cloud.
 4. Head to the Azure Console, and verify the created resources.
-

Screenshots of Provisioned Infrastructure

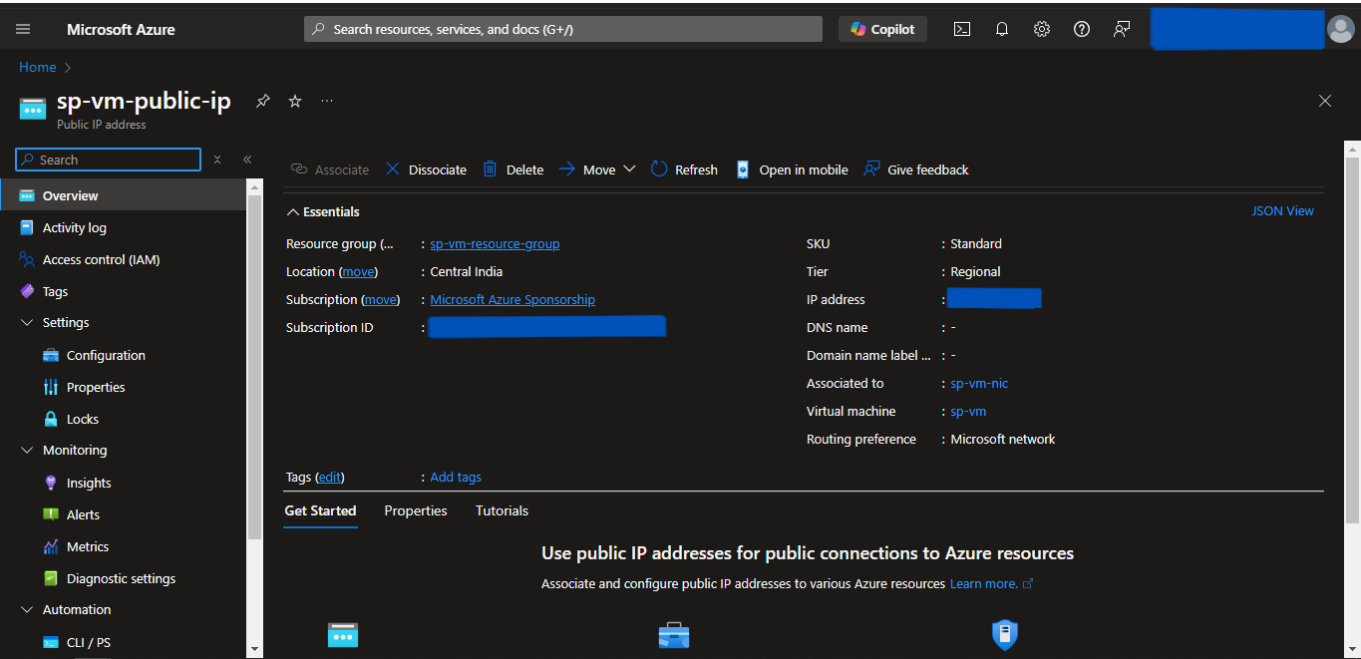
Resource Group Image



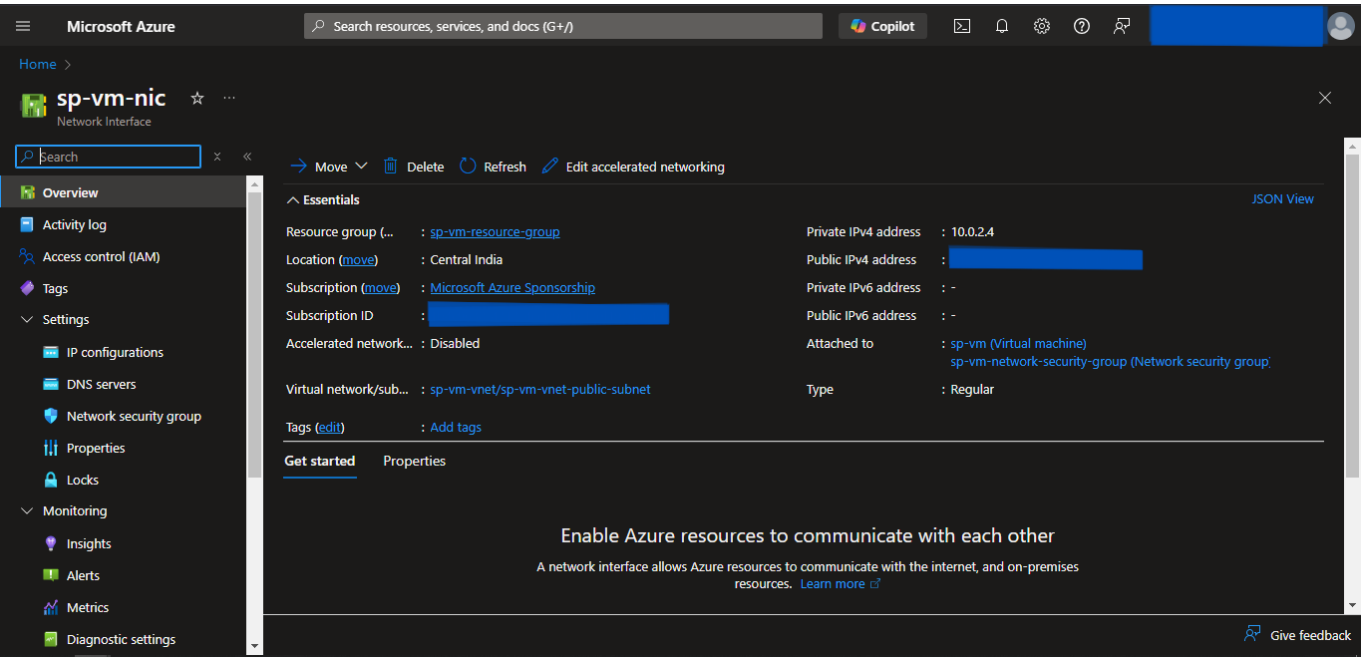
VNet Image



Public IP Image



Network Interface Card Image



Network Security Group Image

Microsoft Azure

Search resources, services, and docs (G+/)

Copilot

Home > sp-vm-nic >

sp-vm-network-security-group

Network security group

Search

Move

Delete

Refresh

Give feedback

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Settings

Inbound security rules

Outbound security rules

Network interfaces

Subnets

Properties

Locks

Monitoring

Alerts

D diagnostic settings

Essentials

JSON View

Resource group (move) : sp-vm-resource-group

Location : Central India

Subscription (move) : Microsoft Azure Sponsorship

Subscription ID :

Tags (edit) : Add tags

Custom security rules : 3 inbound, 0 outbound

Associated with : 0 subnets, 1 network interfaces

Filter by name

Port == all

Protocol == all

Source == all

Destination == all

Action == all

Priority	Name	Port	Protocol	Source	Destination	Action
Inbound Security Rules						
100	ssh-inbound	22	Tcp	Any	Any	Allow
200	http-inbound	80	Tcp	Any	Any	Allow
300	https-inbound	443	Tcp	Any	Any	Allow
65000	AllowVnetInBound	Any	Any	VirtualNetwork	VirtualNetwork	Allow
65001	AllowAzureLoadBalan...	Any	Any	AzureLoadBalancer	Any	Allow
65500	DenyAllInBound	Any	Any	Any	Any	Deny

Virtual Machine Image

Microsoft Azure

Search resources, services, and docs (G+/)

Copilot

Home >

sp-vm

Virtual machine

Search

Help me copy this VM in any region

Connect

Start

Restart

Stop

Hibernate

Capture

Delete

Refresh

Open in mobile

Feedback

CU / PS

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Connect

Connect

Bastion

Networking

Network settings

Load balancing

Application security groups

Network manager

Settings

Disks

Essentials

JSON View

Resource group (move) : SP-VM-RESOURCE-GROUP

Status : Running

Location : Central India

Subscription (move) : Microsoft Azure Sponsorship

Subscription ID :

Operating system : Linux (ubuntu 22.04)

Size : Standard DS1 v2 (1 vcpu, 3.5 GiB memory)

Public IP address :

Virtual network/subnet : sp-vm-vnet/sp-vm-vnet-public-subnet

DNS name : Not configured

Health state : -

Time created : 1/9/2025, 3:48 AM UTC

Tags (edit) : Add tags

Properties

Monitoring

Capabilities (7)

Recommendations

Tutorials

Virtual machine

Computer name : sp-vm

Operating system : Linux (ubuntu 22.04)

Networking

Public IP address : (Network interface sp-vm-nic)

Public IP address (IPv6) : -

SSH Into Azure VM

Now we will SSH into the Azure VM and configure it for Nginx container deployment.

Steps

1. Open the Powershell Window.
2. Run the following command to SSH into Azure VM and substitute the `<admin-username>` with the value provided in `values.py` file under `<virtual_machine_properties>` section and `<vm-public-ip>` with the Azure VM Public IP received from `pulumi stack output vm-public-ip` command:

```
ssh -o StrictHostKeyChecking=no <admin-username>@<vm-public-ip>
```

3. It will prompt for password, enter the `<admin-password>` provided in the `values.py` file under `<virtual_machine_properties>` section.
4. Once you enter the server, run the following commands to install the necessary dependencies for deployment and run the nginx container:

```
sudo apt update
sudo apt install -y docker.io
sudo docker run -d -p 80:80 nginx
```

9. Try accessing it on the browser using `<vm-public-ip>` received from `pulumi stack output vm-public-ip` command.

Nginx Image

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Destroy the provisioned infrastructure

Lastly, we will destroy the above-created resources.

Steps

1. To destroy infrastructure, open the Powershell Window and change the directory to the above-created Pulumi Project using the `cd` command.
 2. Run `pulumi destroy` & if prompted, select `yes`.
 3. Infrastructure will be destroyed.
-