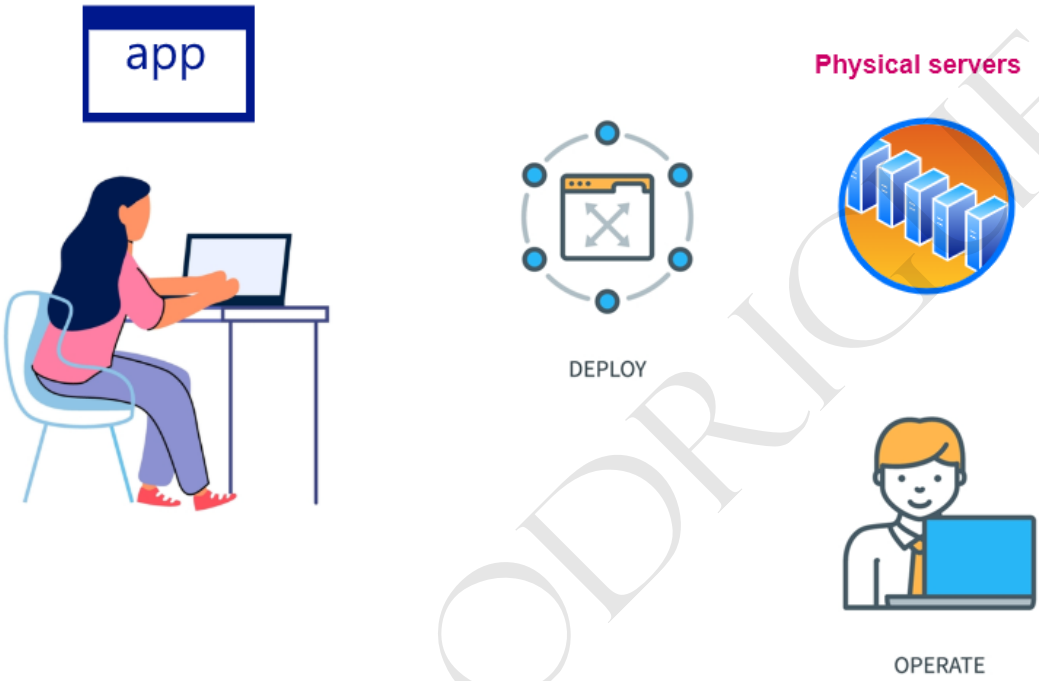
Introduction

Why Terraform



You need infrastructure in some form or the other for hosting your applications

Cloud adoption - Azure , AWS , Google

Deploying your infrastructure

Application Server

Database server

1. Deploy the infrastructure

2. Configure your infrastructure

Improvements to your application

Could warrant for changes to your infrastructure

Add autoscaling features to your infrastructure

Add high availability

Application Server

Database server

Production Environment

Application Server

Database server

Test Environment

Destroy the test environment after testing is complete.

Recreate the test environment again

That's really a task to recreate the entire environment again

Is there an easier way to recreate the environment

Infrastructure as code

Code file

Application Server

Database server

Change the code whenever required

Share the code

Create different versions of the code

Very important when deploying infrastructure on the cloud

Cloud follows the Pay-As-You-Go model

Different providers have been creating tools that help to facilitate the creation of Infrastructure as code

What makes Terraform popular

It works with a variety of cloud platforms.

The code is human-readable

Avid community

Open-source project

Terraform Workflow

1. Write your Terraform configuration file

This defines the resources that need to be deployed

Application Server

Database server

Environment

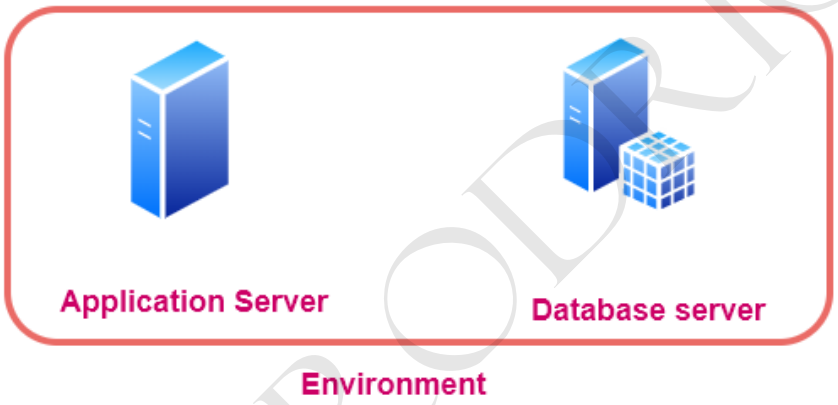
2. Terraform plan

Terraform looks at the configuration file and decides what needs to be deployed or changed.

It actually maintains a state file.

3. Terraform apply

Apply all of the changes as per the Terraform configuration file



Configuration File

Provision the infrastructure as per the details specified in the configuration file.

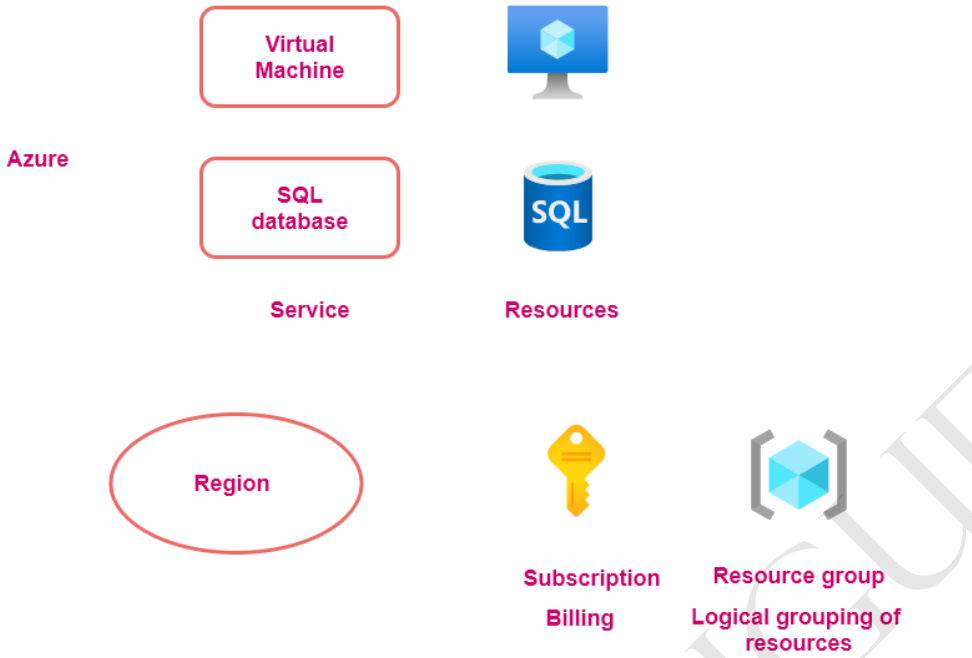
Application Server

Database server

Cloud follows the Pay-As-You-Go model

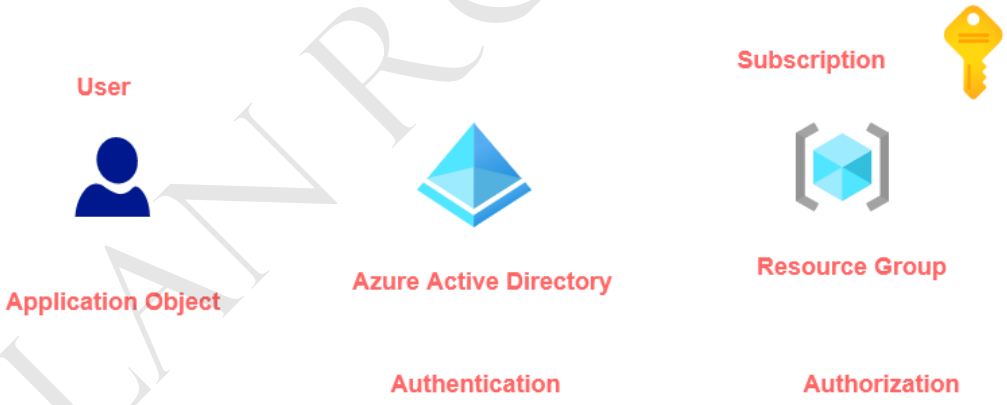
Azure Datacenters

Servers

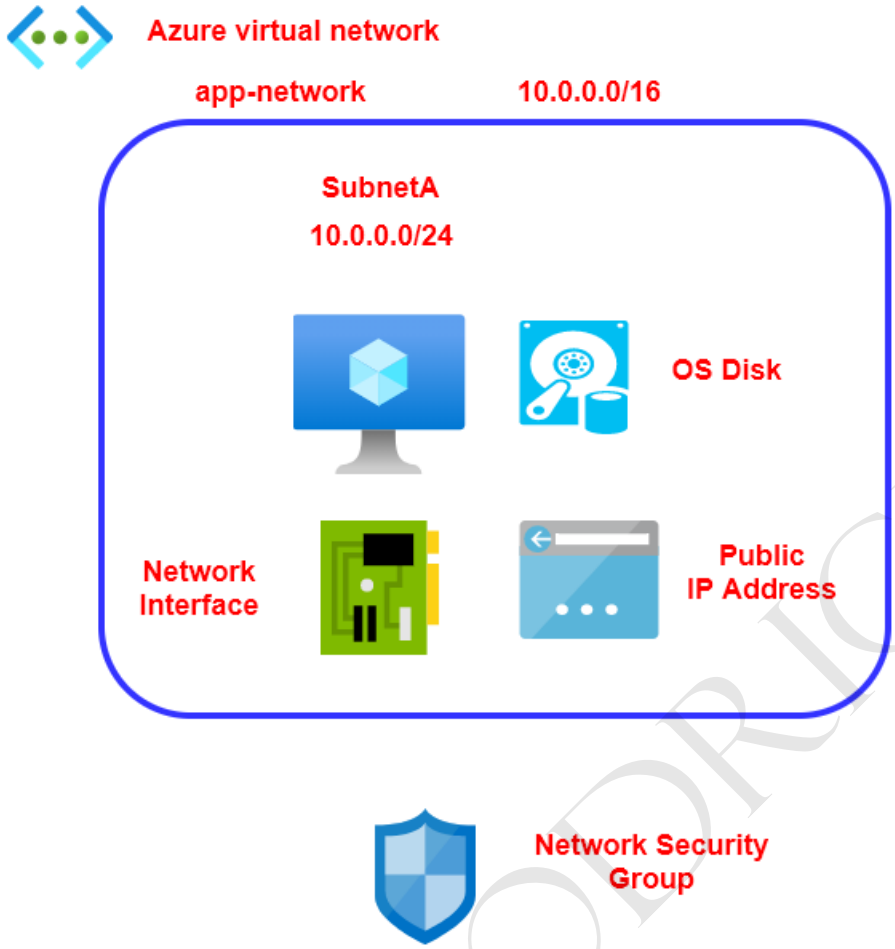


Azure Storage Accounts and Virtual Machines

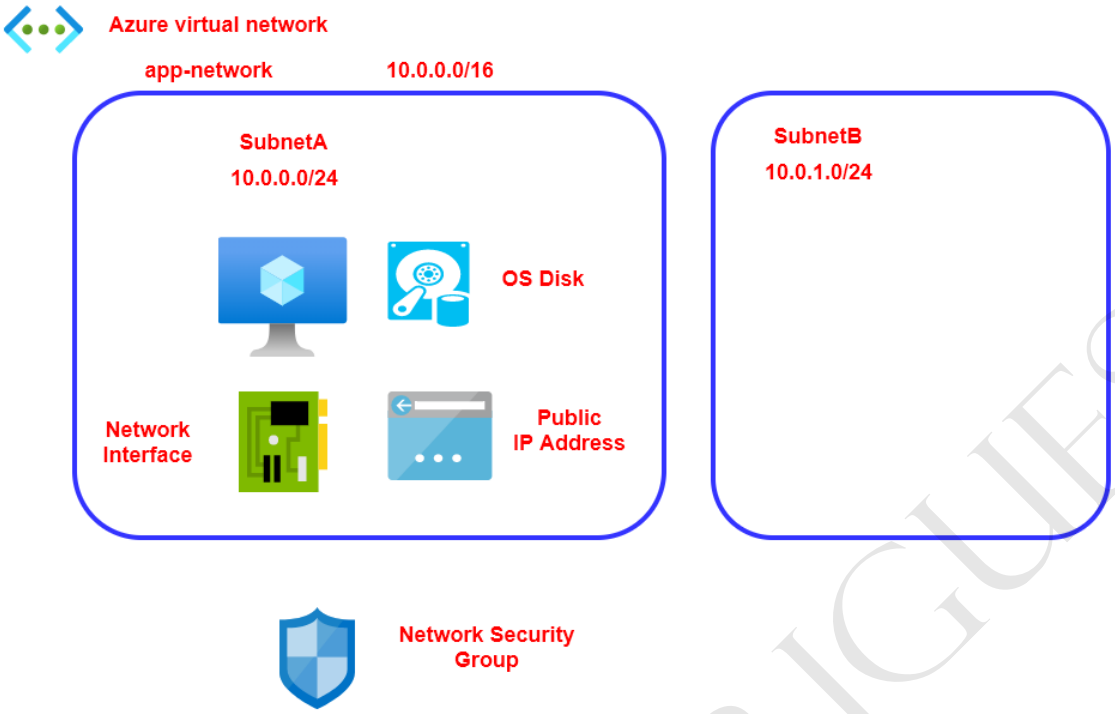
Azure AD Authorization



Reviewing the creation of an Azure Virtual Machine

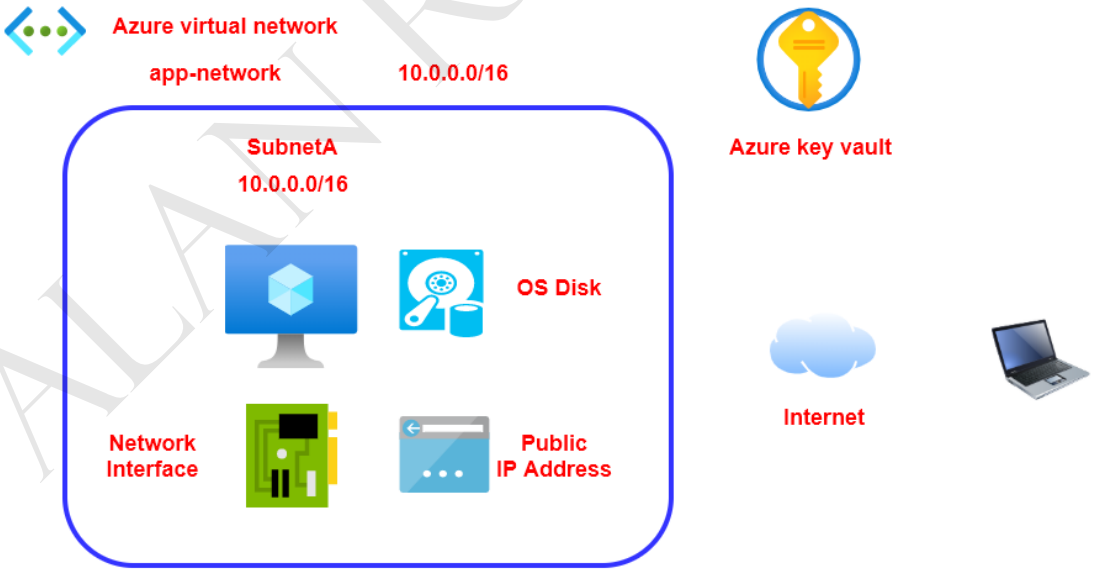


Lab - Creating a virtual network interface

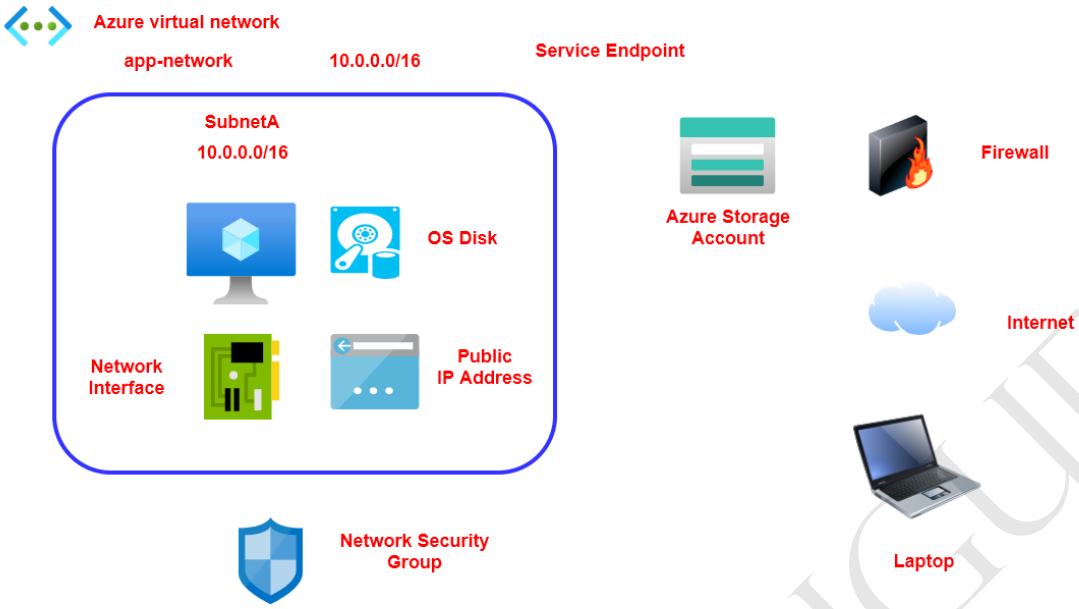


Azure Storage Accounts and Virtual Machines - Further aspects

About the Key vault service

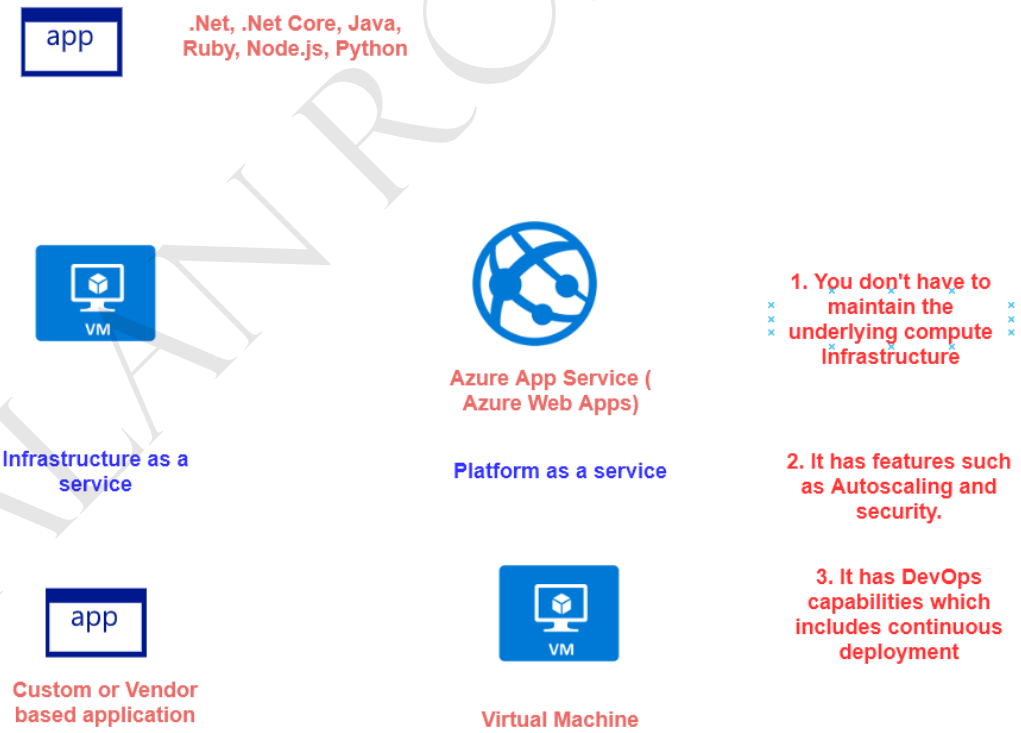


Storage Accounts - Firewall - What we want to achieve

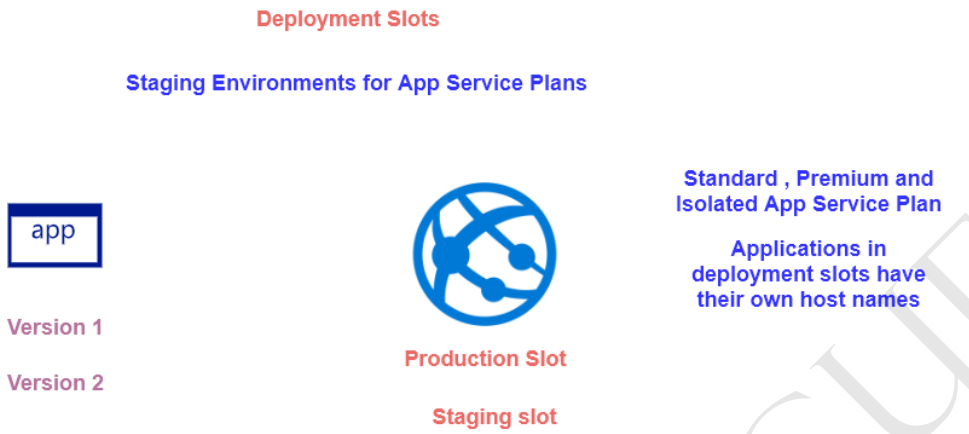


Azure Web App and SQL Database

Lab - Azure Web App - Manual process

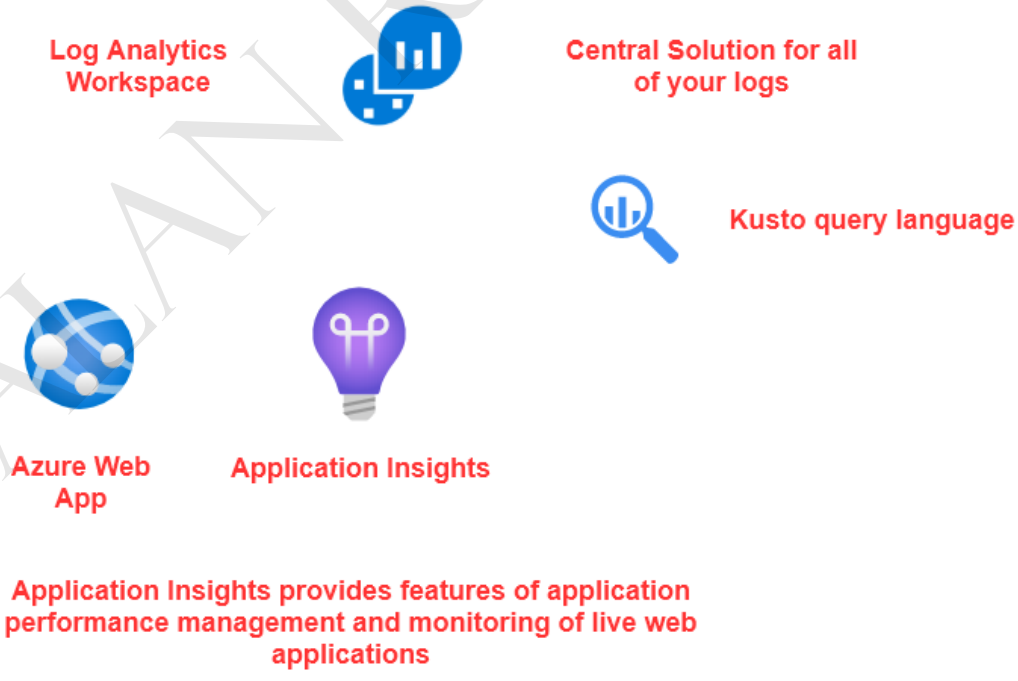


Understanding on deployment slots



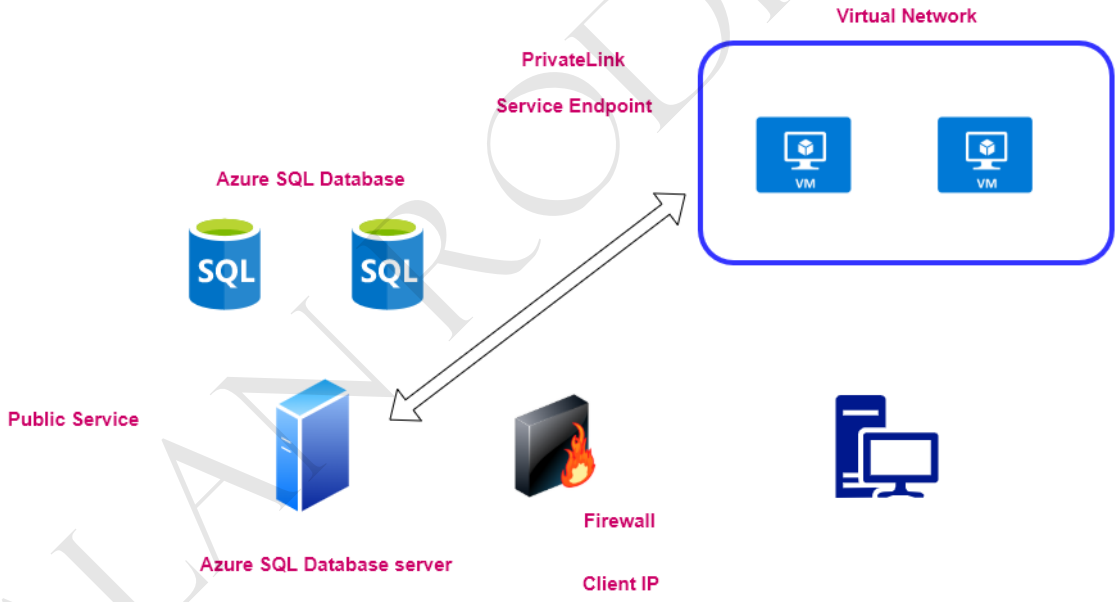
1. You have the chance to validate all application changes in the staging deployment slot
2. You can then swap the staging slot with the production slot
3. This helps eliminate the downtime for your application when new changes are deployed
4. You can also easily roll back the changes

Lab - Application Insights



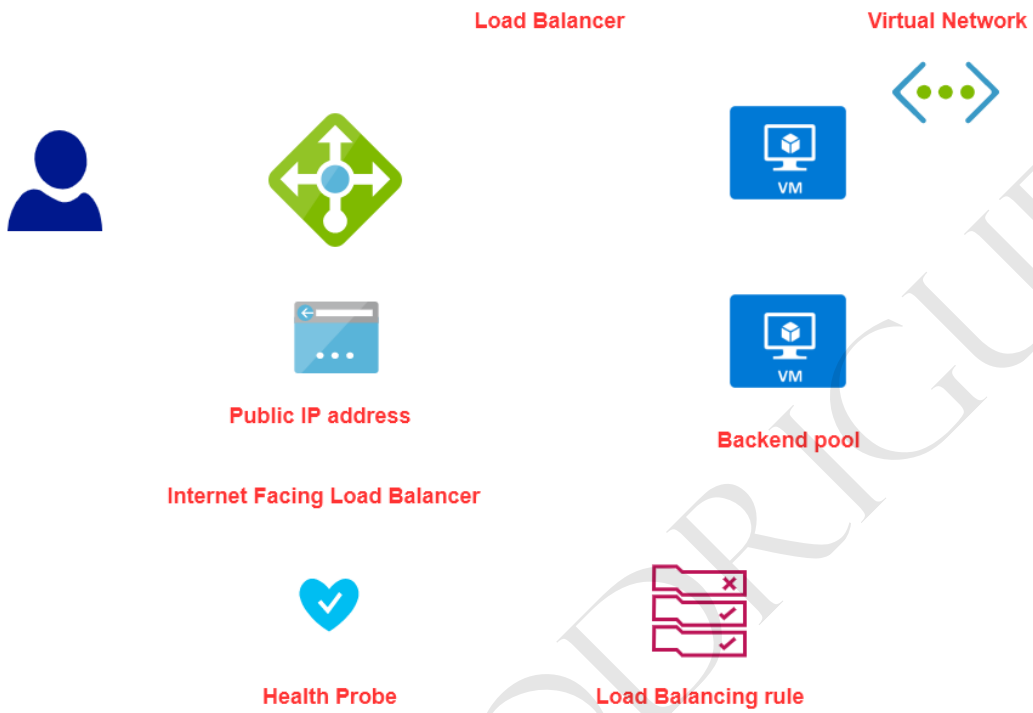
Lab - Azure SQL Database - Manual process

Lab - Azure SQL Database - Firewall rules

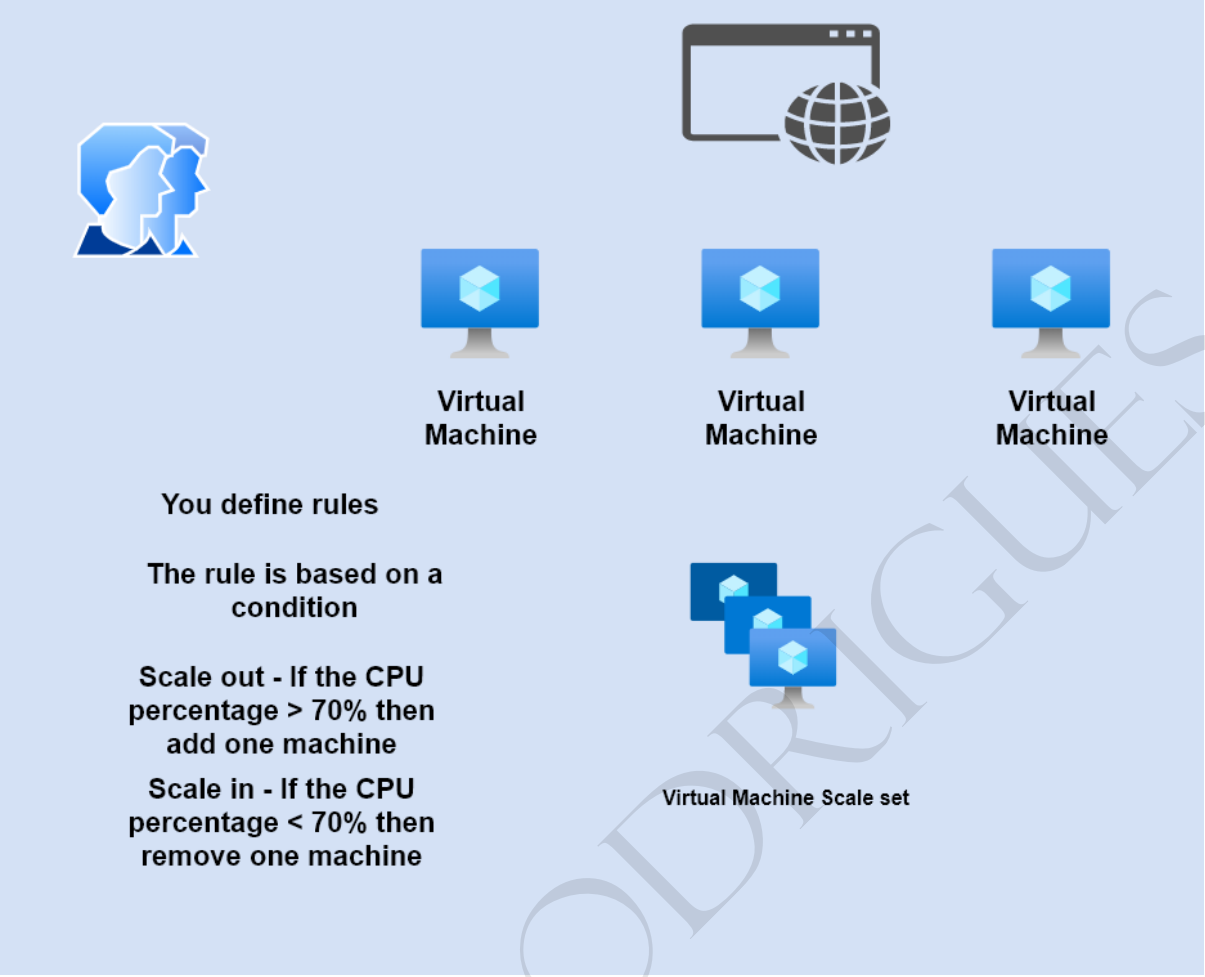


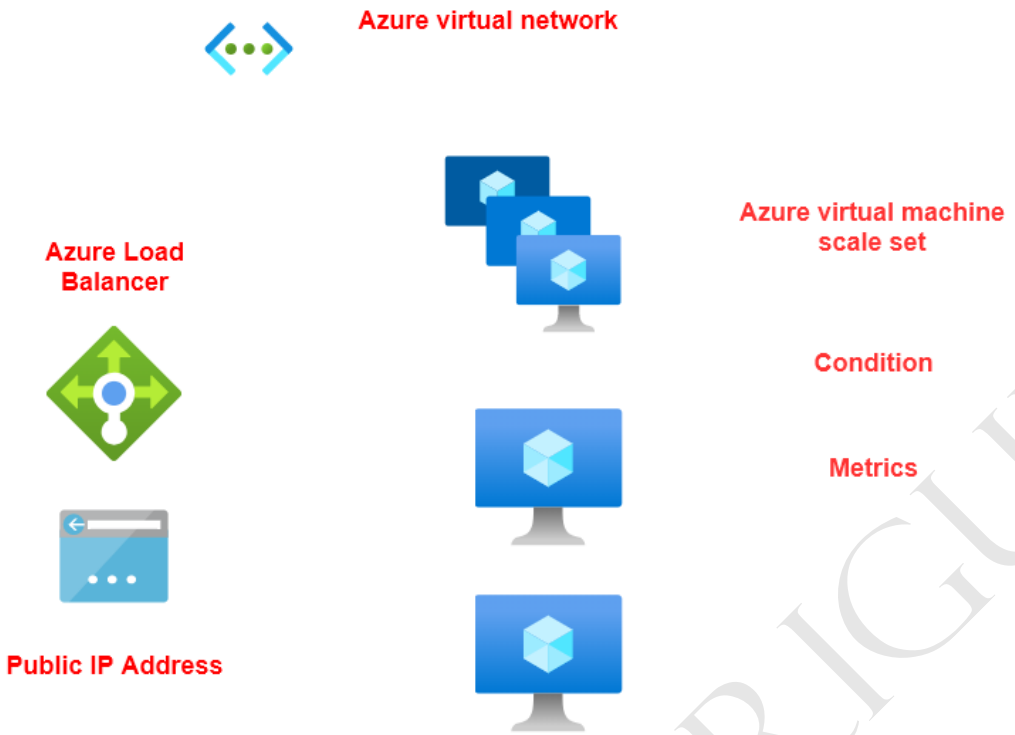
Azure Networking

Lab - Azure Load Balancer

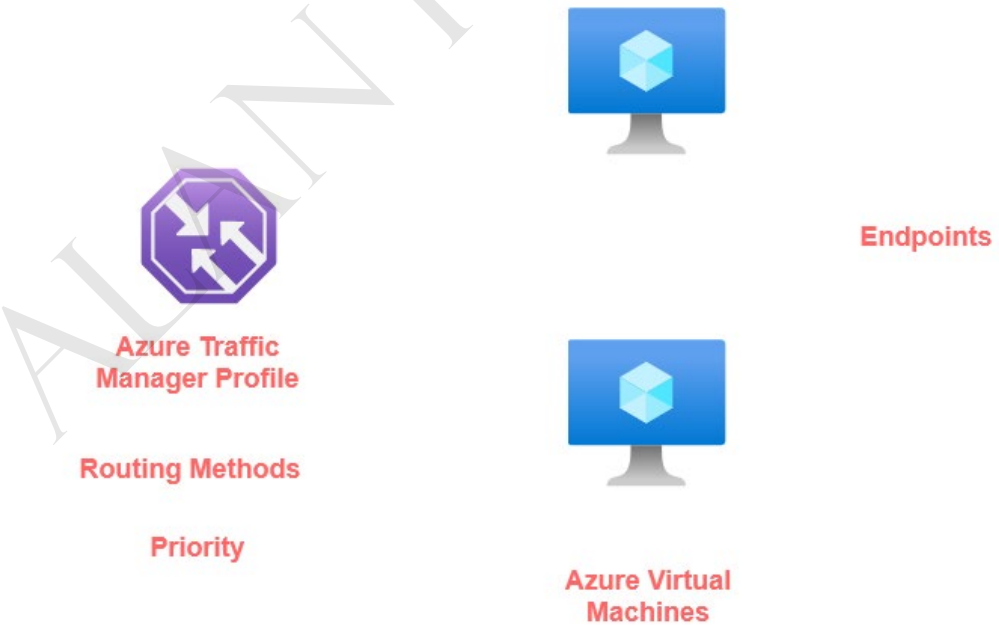


Lab - Virtual Machine Scale Set Deployment

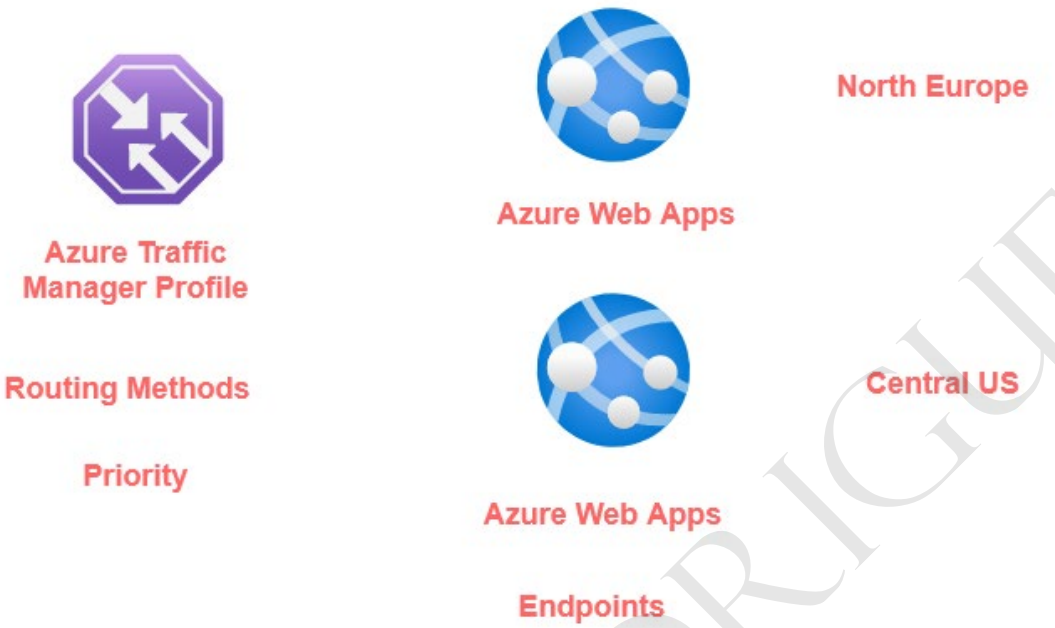




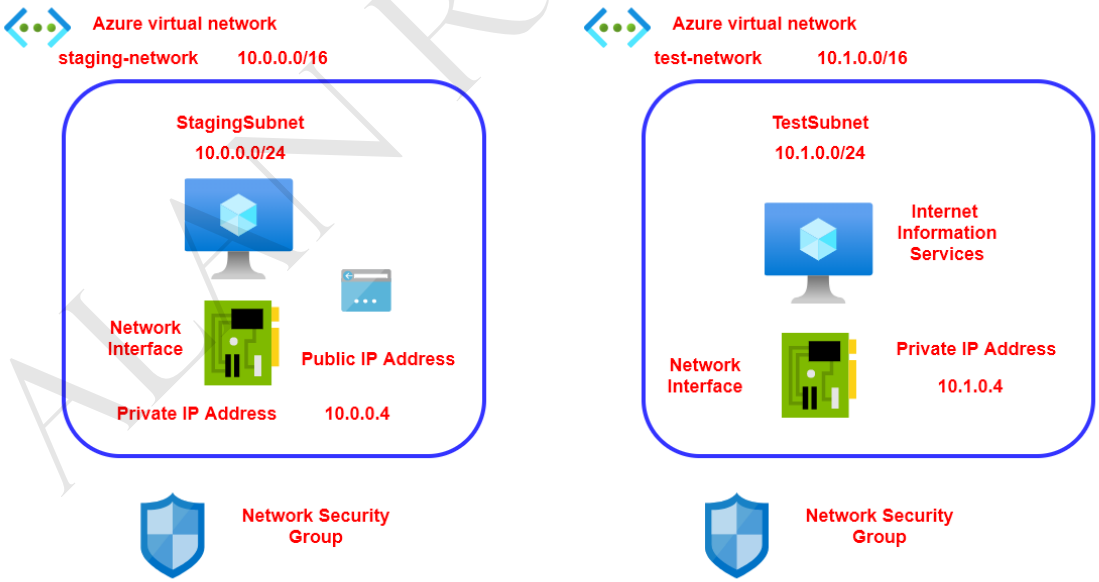
About Azure Traffic Manager



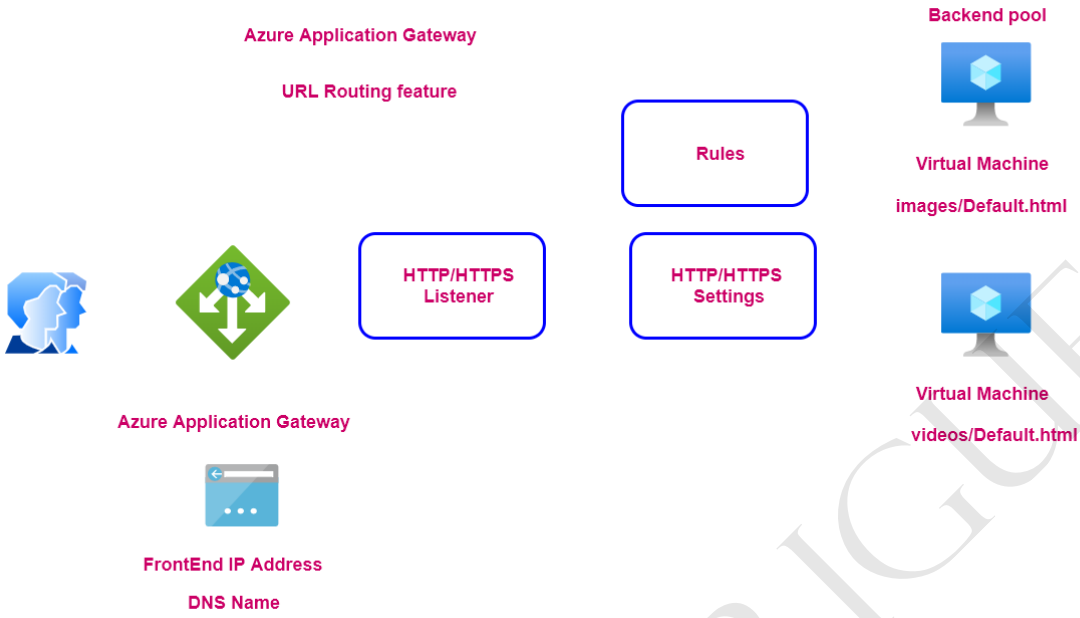
Implementing Azure Traffic Manager Manually



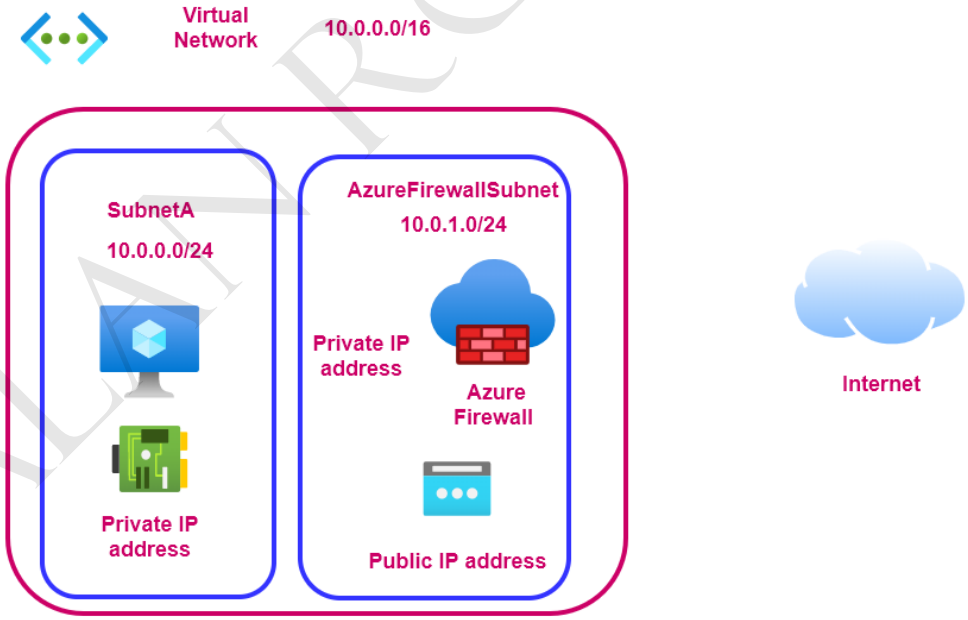
Lab - Virtual Network Peering - Manual deployment



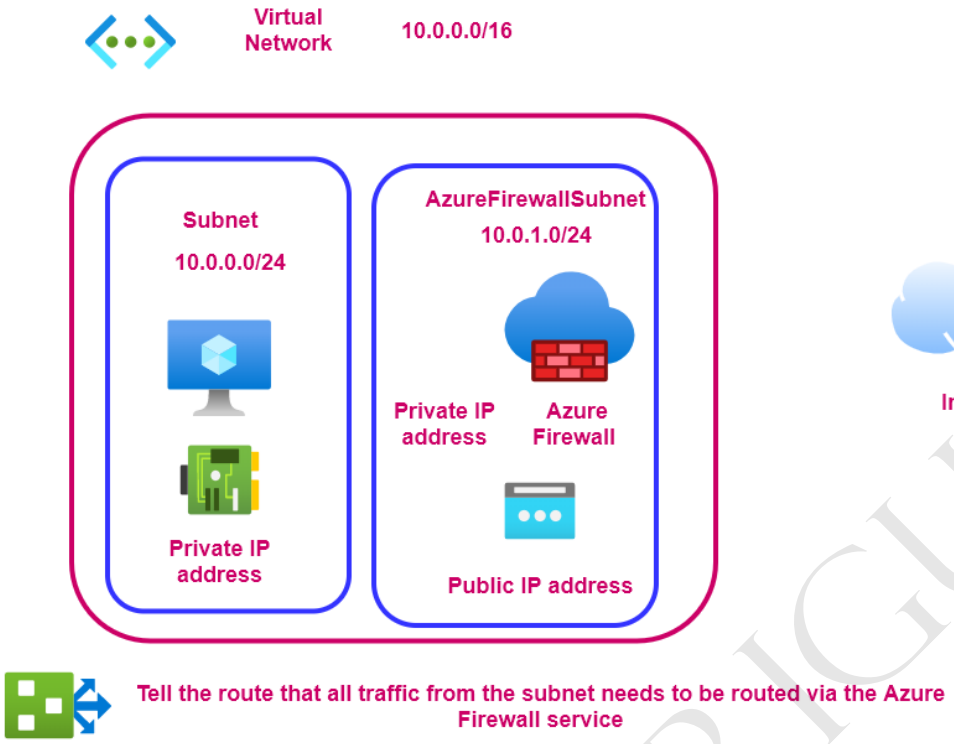
Lab - Deploying Azure Application Gateway Manually



Lab - Azure Firewall



Step 1 : Create our virtual network and Azure Firewall appliance

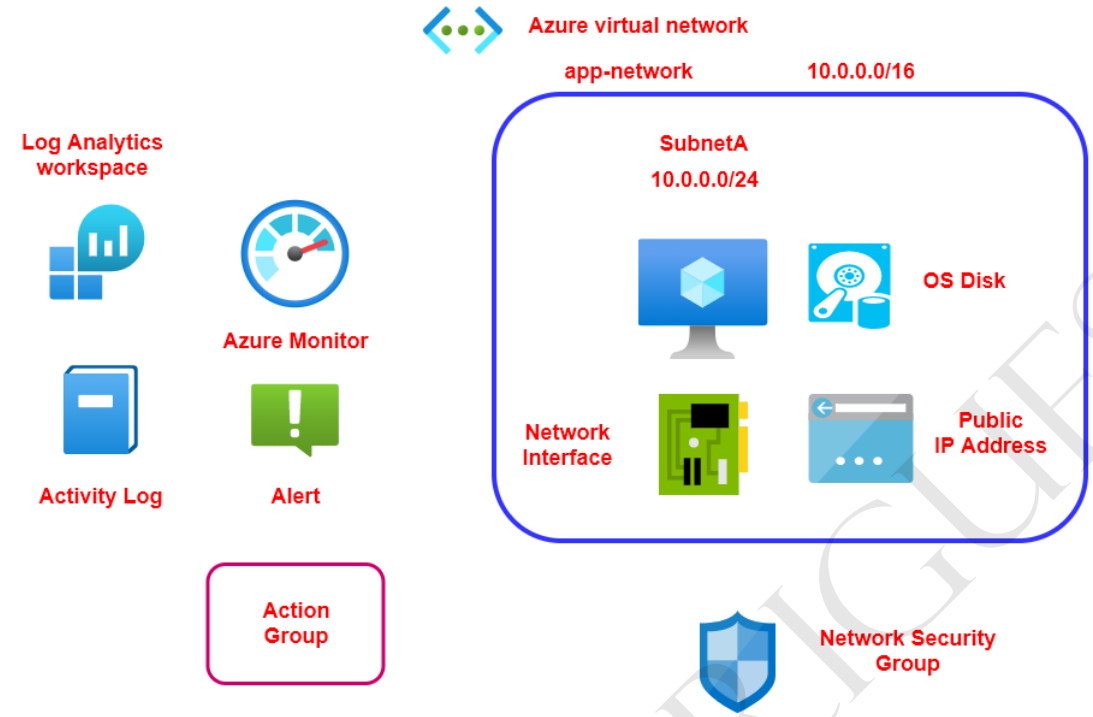


Tell the route that all traffic from the subnet needs to be routed via the Azure Firewall service

Step 2 : Create a route table and assign it to the Subnet hosting the virtual machine

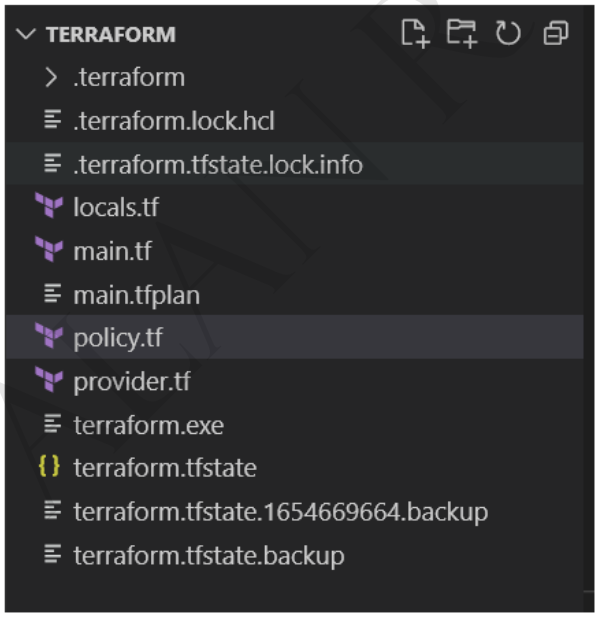
Identity Management, Monitoring and governance

Configuring a Log Analytics workspace



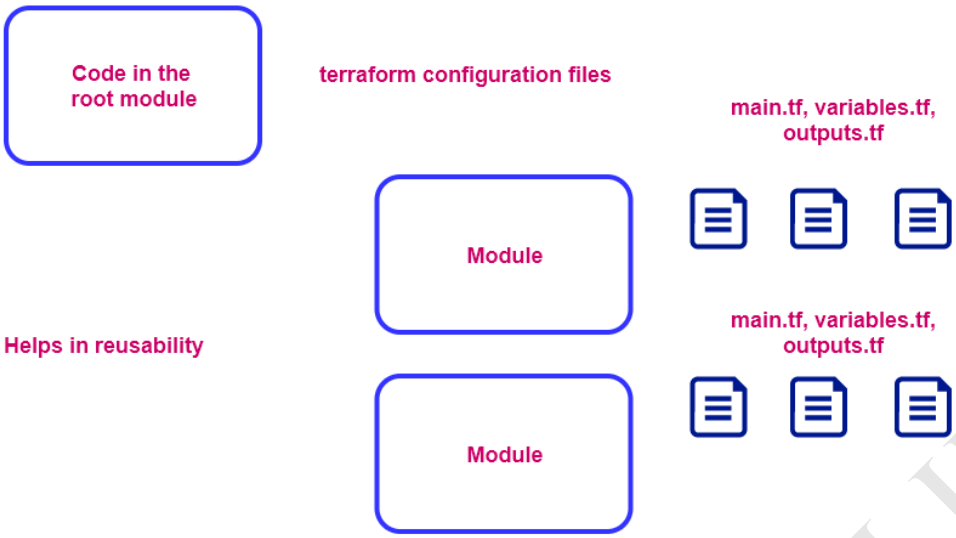
Terraform - More on features

Terraform modules



All this time we have already been working with modules

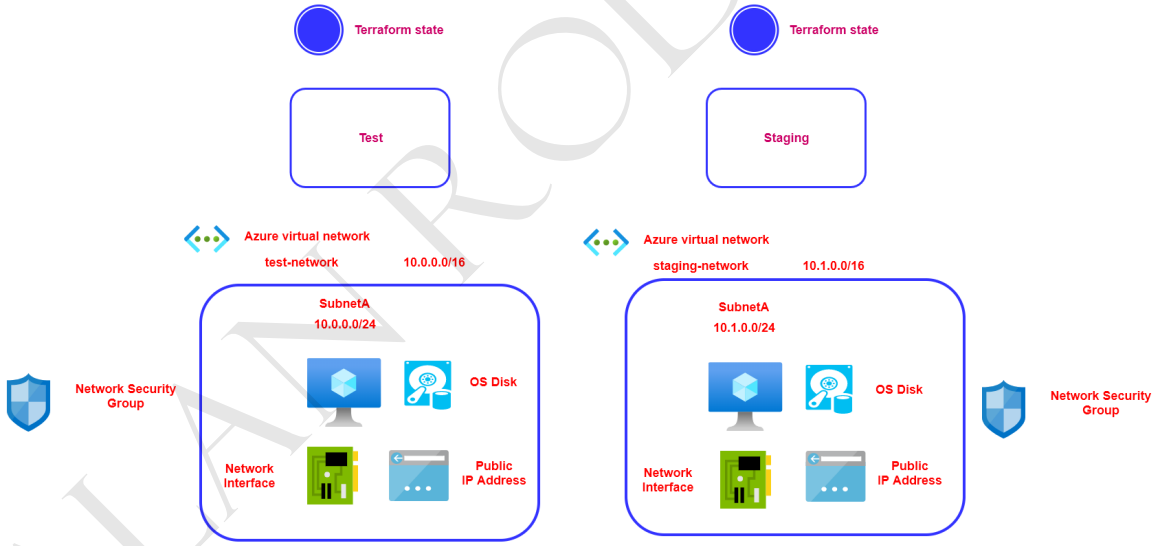
All of our code is already part of the root module



Modules are containers that can be used to host multiple resources that can be used together.

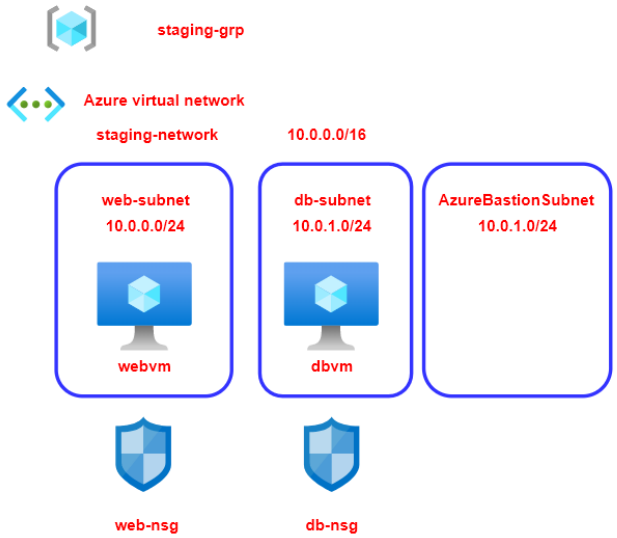
The root modules contains the resources defined in your terraform configuration file in the main working directory

Terraform workspaces



Mini Project

Our Architecture



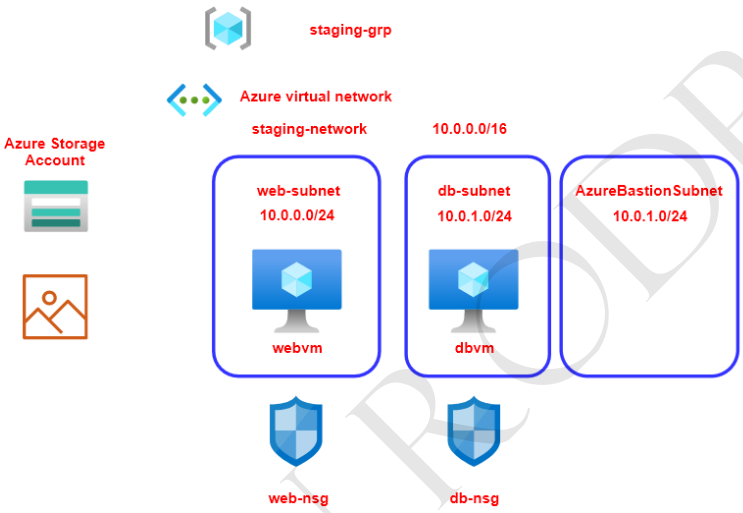
1. webvm would be hosting an ASP.NET application that would connect to the database hosted on db-subnet

2. dbvm would be hosting Micrsoft SQL Server 2019

3. webvm would have a Public IP address that would accept connections from the Internet

4. dbvm would not have a Public IP address assigned.

5. RDP connections need to be made via the Azure Bastion host



1. webvm would connect to an Azure storage account.

2. The Azure storage account would contain images that would be picked up by the web application.