

Terraform on Azure Cloud

Author: Nho Luong

Skill: DevOps Engineer Lead



A screenshot of a web browser displaying the Microsoft Learn profile page for "nholuong-0040". The page shows four active certifications under the "Credentials" tab. 1. Microsoft Certified: DevOps Engineer Expert - Expires June 15, 2025. 2. Microsoft Certified: Azure Solutions Architect Expert - Expires June 15, 2025. 3. Microsoft Certified: Azure Administrator Associate - Expires June 15, 2025. 4. Microsoft Certified: Azure Fundamentals - Earned on May 24, 2024. Each certification entry includes a "View certification details" link and status indicators for "Online Verifiable" and "Active".



AWS EKS
Kubernetes

Azure AKS
Kubernetes

AWS ECS
Docker on AWS

AWS CloudFormation

STACK

**DevOps &
SRE
Roadmap**

AWS Lambda

Python

**DevOps on
AWS & Azure**

**HashiCorp Certified
Vault & Consul**

**Kubernetes
Certifications
CKAD, CKA, CKS**

Jenkins
Ansible

**Azure AKS
Part-2**

**Azure Certs
AZ-900, 104, 204, 400**

**Google Cloud
Associate &
Professional Certs**

HashiCorp Certified Terraform Associate on AWS with Practical Demos

Terraform on AWS with SRE and IaC DevOps with Real-World Demos

HashiCorp Certified Terraform Associate on Azure with Practical Demos

Terraform on Azure with IaC DevOps SRE with Real-World Demos

Terraform on AWS EKS Elastic Kubernetes Service with Real-World Demos

Terraform Fundamentals (Commands, Language, Settings, Providers, Resources)

Azure Virtual Network, Subnets and Network Security Groups

Azure Virtual Machines, Network Interface and Public IP

Azure Bastion Host and Bastion Service

Azure Standard Load Balancer with Inbound NAT Rules

Azure Virtual Machine Scale Sets with Manual Scaling

Azure VMSS with Autoscaling Default, Recurrence and Fixed Profiles

Azure Standard Load Balancer (External and Internal)

Azure Private and Public DNS Zones

Terraform On Azure Cloud

Real-World Approach

Step by Step Documentation On GitHub

Incremental way to Build Complex Infra

Azure Traffic Manager

Azure Application Gateway Basics

Azure Application Gateway Context Path based Routing and Multisite Hosting

Azure Application Gateway SSL, HTTP to HTTPS Redirect, SSL with Key Vault

Terraform Local Modules – Leverage Public Registry

Terraform Local Modules – Build from scratch and Publish to Public TF Registry

Terraform Remote State Storage & Remote State Datasource

Azure MySQL Single Server with Azure Application Gateway

IaC DevOps with Azure DevOps for Terraform Project with Build and Release Pipelines

GitHub Step-by-Step Documentation

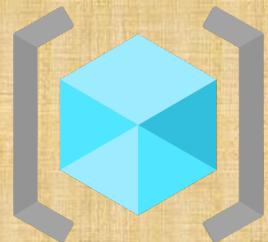
terraform-on-azure-cloud

- > 01-Infrastructure-as-Code-IaC-Basics
- > 02-Install-Tools-TerraformCLI-AzureCLI-VSCodeIDE
- > 03-Terraform-Command-Basics
- > 04-Terraform-Language-Syntax
- > 05-Terraform-Provider-Resource-Block-Basics
- > 06-Azure-Terraform-VsCode-Plugin
- > 07-Multiple-Provider-Configurations
- > 08-Providers-Dependency-Lock-File
- > 09-Resource-Syntax-and-Behavior
- > 10-Azure-Virtual-Network-4Tier
- > 11-Azure-Linux-Virtual-Machine
- > 12-Azure-Bastion-Service-and-Host
- > 13-Azure-Standard-LoadBalancer-using-Portal
- > 14-Azure-Standard-LoadBalancer-Basic
- > 15-Azure-Standard-LoadBalancer-Inbound-NATRules
- > 16-Azure-SLB-VM-with-MetaArgument-Count
- > 17-Azure-SLB-VM-with-for-each-and-for-loops
- > 18-Azure-VM-ScaleSets-Manual-scaling
- > 19-Azure-VM-ScaleSets-Auto-scaling
- > 20-Azure-External-and-Internal-LB-with-VMSS

TF Configs
well kept
on GitHub

- > 21-Azure-Private-DNS-Zones
- > 22-Delegate-DNS-Domain-to-Azure-DNS
- > 23-Azure-Public-DNS-Zone
- > 24-Terraform-Remote-State-Storage
- > 25-Azure-Traffic-Manager
- > 26-Azure-Application-Gateway-using-Portal
- > 27-Azure-Application-Gateway-Basics
- > 28-Azure-Application-Gateway-Path-Based-Routing
- > 29-Azure-Application-Gateway-Multisite-Hosting
- > 30-Azure-Application-Gateway-SSL-SelfSigned
- > 31-Azure-Application-Gateway-SSL-SelfSigned-KeyVault
- > 32-Azure-IaC-DevOps
- > 33-Azure-MySQL-Single-Server
- > 34-Terraform-Modules-use-Public-Module
- > 35-Terraform-Azure-Static-Website
- > 36-Terraform-Modules-Build-Local-Module
- > 37-Terraform-Module-Publish-to-Public-Registry
- > 38-Terraform-Module-Sources

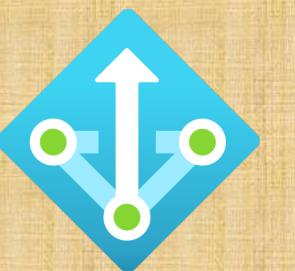
20 Azure Services



Resource Group



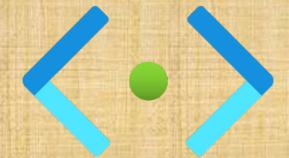
Traffic Manager



NAT Gateway



Virtual Network



Subnet



Network Security Group



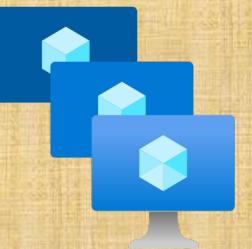
Network Interface



Virtual Machine



Azure Disk



VM Scale Set



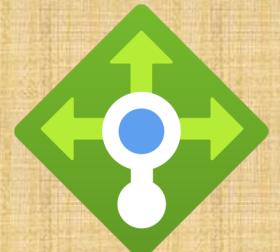
Public IP



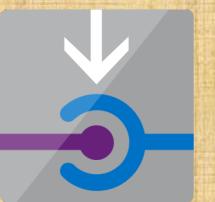
DNS Zone



MySQL Server



Load Balancer



LB Inbound NAT



Application Gateway



Storage Account



Azure DevOps Pipelines

Terraform Concepts covered

1. Settings Block
2. Providers Block
3. Multiple Providers usage
4. Dependency Lock File Importance
5. Resources Syntax and Behavior
6. Resources Meta-Argument - depends_on
7. Resources Meta-Argument - count
8. Resources Meta-Argument - for_each
9. Resources Meta-Argument - lifecycle
10. Input Variables - Basics
11. Input Variables - Assign When Prompted
12. Input Variables - Assign with terraform.tfvars
13. Input Variables - Assign with tfvars var-file argument
14. Input Variables - Assign with auto tfvars
15. Input Variables - Lists

Terraform Concepts

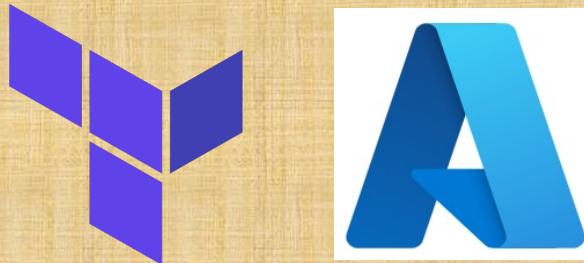
16. Input Variables - Maps
17. Input Variables - Sensitive Input Variables
18. File Function
19. Output Values
20. Local Values
21. Datasources
22. Backends - Remote State Storage
23. File Provisioner
24. remote-exec Provisioner
25. Null Resource
26. Modules from Public Registry
27. Build Local Module
28. Dynamic Blocks
29. base64encode function
30. filebase64 function
31. element function
32. lookup function
33. Remote State Datasource

Terraform Concepts

Azure Cloud

&

Azure Resources



GitHub Step-by-Step Documentation

Terraform Configs

terraform-manifests

- > app-scripts
- > ssh-keys
- ↳ c1-versions.tf
- ↳ c2-generic-input-variables.tf
- ↳ c3-locals.tf
- ↳ c4-random-resources.tf
- ↳ c5-resource-group.tf
- ↳ c6-01-vnet-input-variables.tf
- ↳ c6-02-virtual-network.tf
- ↳ c6-03-web-subnet-and-nsg.tf
- ↳ c6-04-app-subnet-and-nsg.tf
- ↳ c6-05-db-subnet-and-nsg.tf
- ↳ c6-06-bastion-subnet-and-nsg.tf
- ↳ c6-07-vnet-outputs.tf

- ↳ c7-01-web-linux-vmss-input-variables.tf
- ↳ c7-02-web-linux-vmss-nsg-inline-basic.tf
- ↳ c7-03-web-linux-vmss-resource.tf
- ↳ c7-04-web-linux-vmss-outputs.tf
- ↳ c7-05-web-linux-vmss-autoscaling-default-profile.tf

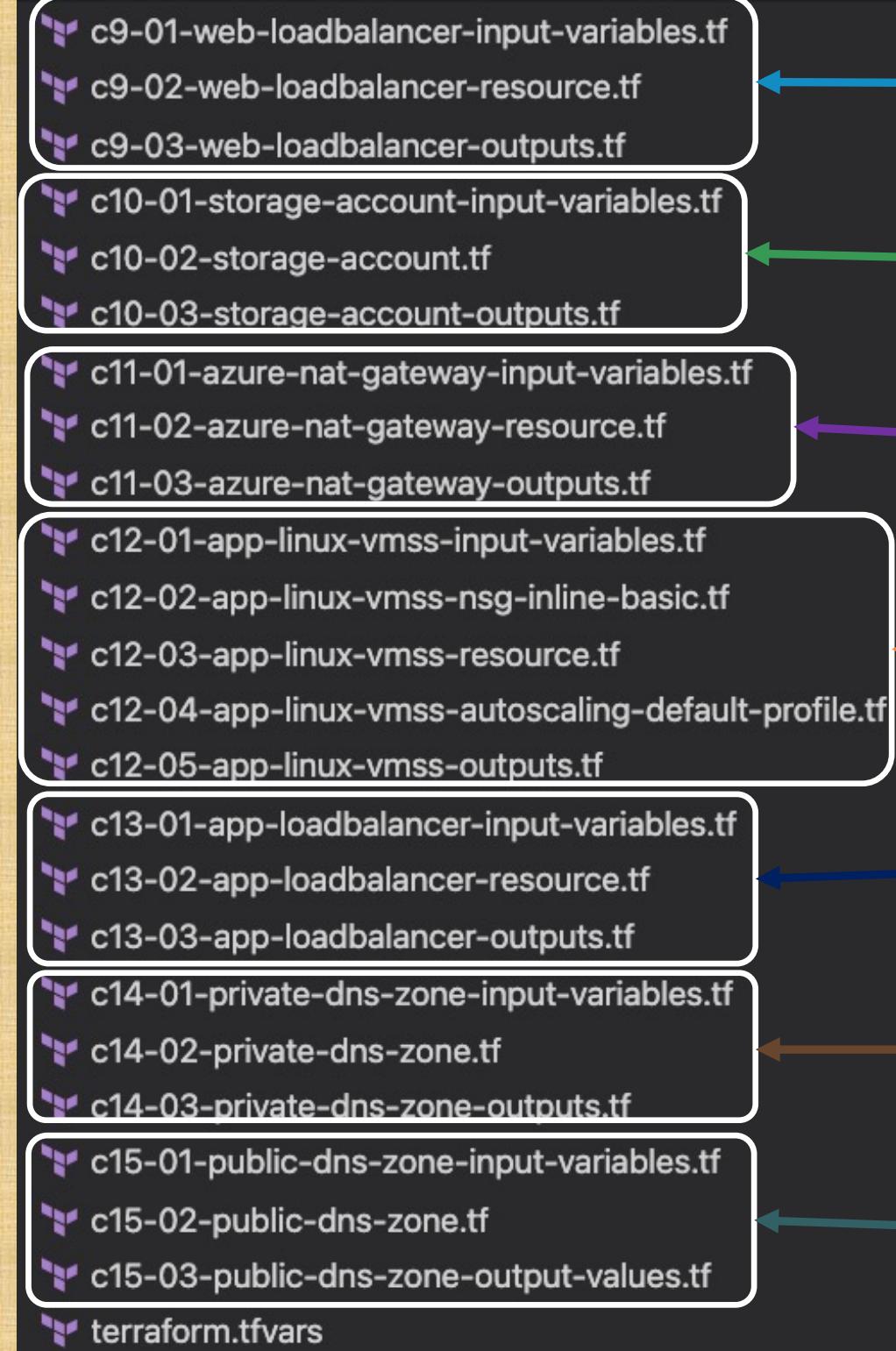
- ↳ c8-01-bastion-host-input-variables.tf
- ↳ c8-02-bastion-host-linuxvm.tf
- ↳ c8-03-move-ssh-key-to-bastion-host.tf
- ↳ c8-04-AzureBastionService.tf
- ↳ c8-05-bastion-outputs.tf

Azure Virtual Network with Subnets and Network Security Groups

Azure Virtual Machine Scale Set with Auto Scaling Profile - Web Tier

Azure Bastion Host Linux VM – Enabled
Azure Bastion Service - Disabled

Terraform Configs



Azure Standard Load Balancer – External
(Web Tier Load Balancer)

Azure Storage Account – Deploy httpd conf

Azure NAT Gateway – Outbound
communication for App VMSS

Azure Virtual Machine Scale Set with Auto
Scaling Profile - App Tier

Azure Standard Load Balancer – Internal
(App Tier Load Balancer)

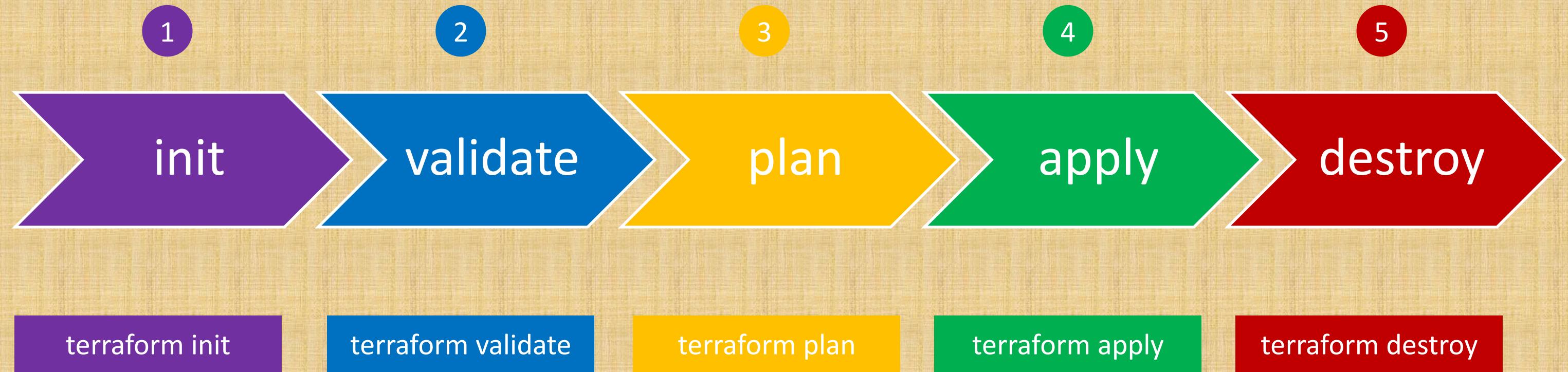
Azure Private DNS Zone for App Tier Load
Balancer internal DNS Name in VNET

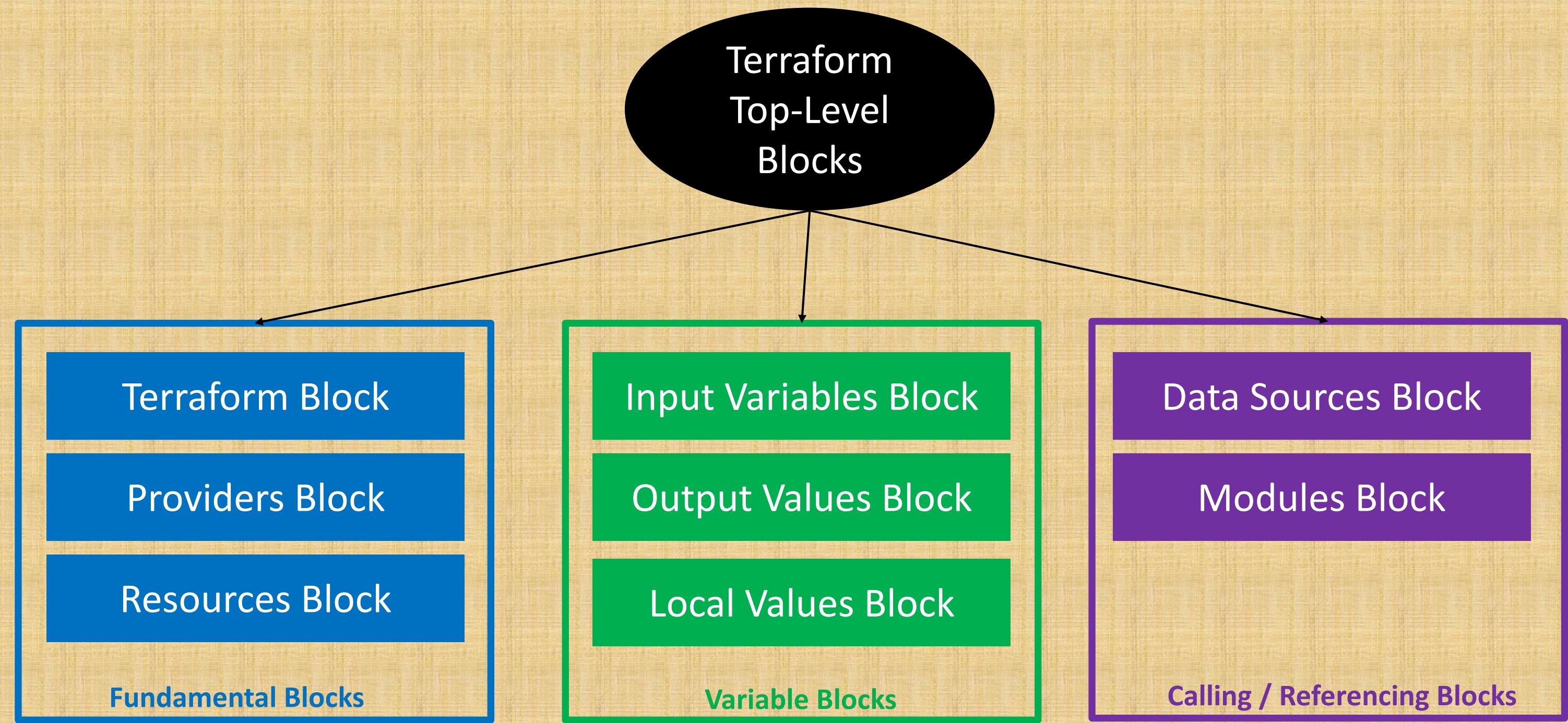
Azure Public DNS Zone – Access Applications
via Internet using Registered Domain

GitHub Repositories

Repository Used For	Repository URL
Course Main Repository with step-by-step documentation	https://github.com/nholuongut/terraform-on-azure-cloud
Azure IaC DevOps with Build and Release Pipelines	https://github.com/nholuongut/terraform-on-azure-with-azure-devops

Terraform Workflow





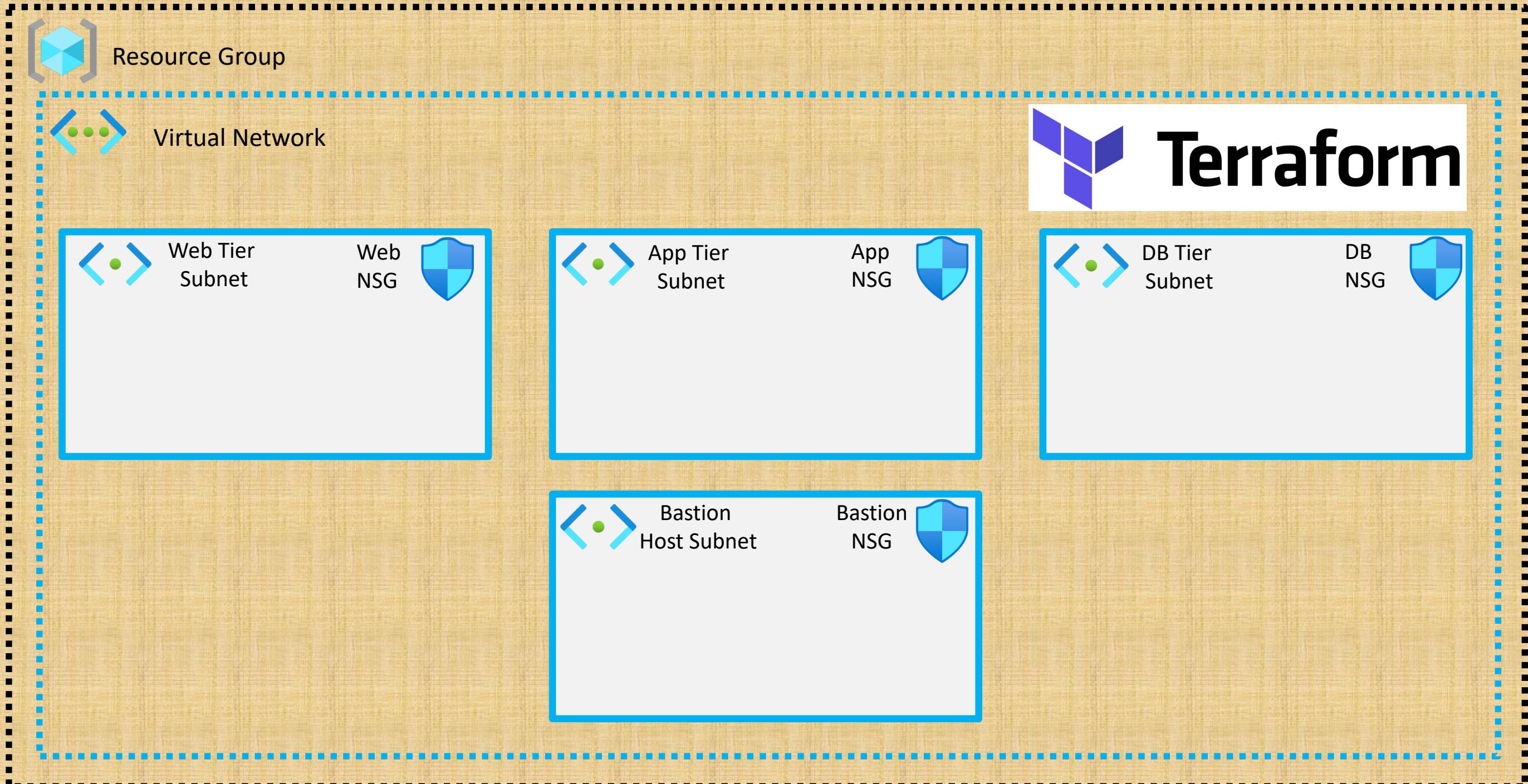
Terraform Fundamentals

✓ Terraform - Install Tools	5 lectures • 31min
✓ Terraform Command Basics	3 lectures • 25min
✓ Terraform Language Basics	3 lectures • 21min
✓ Terraform Settings & Providers Block	5 lectures • 44min
✓ Terraform Multiple Providers	2 lectures • 12min
✓ Terraform Dependency Lock File	4 lectures • 30min
✓ Terraform Resource Syntax, Behaviour and State	8 lectures • 49min

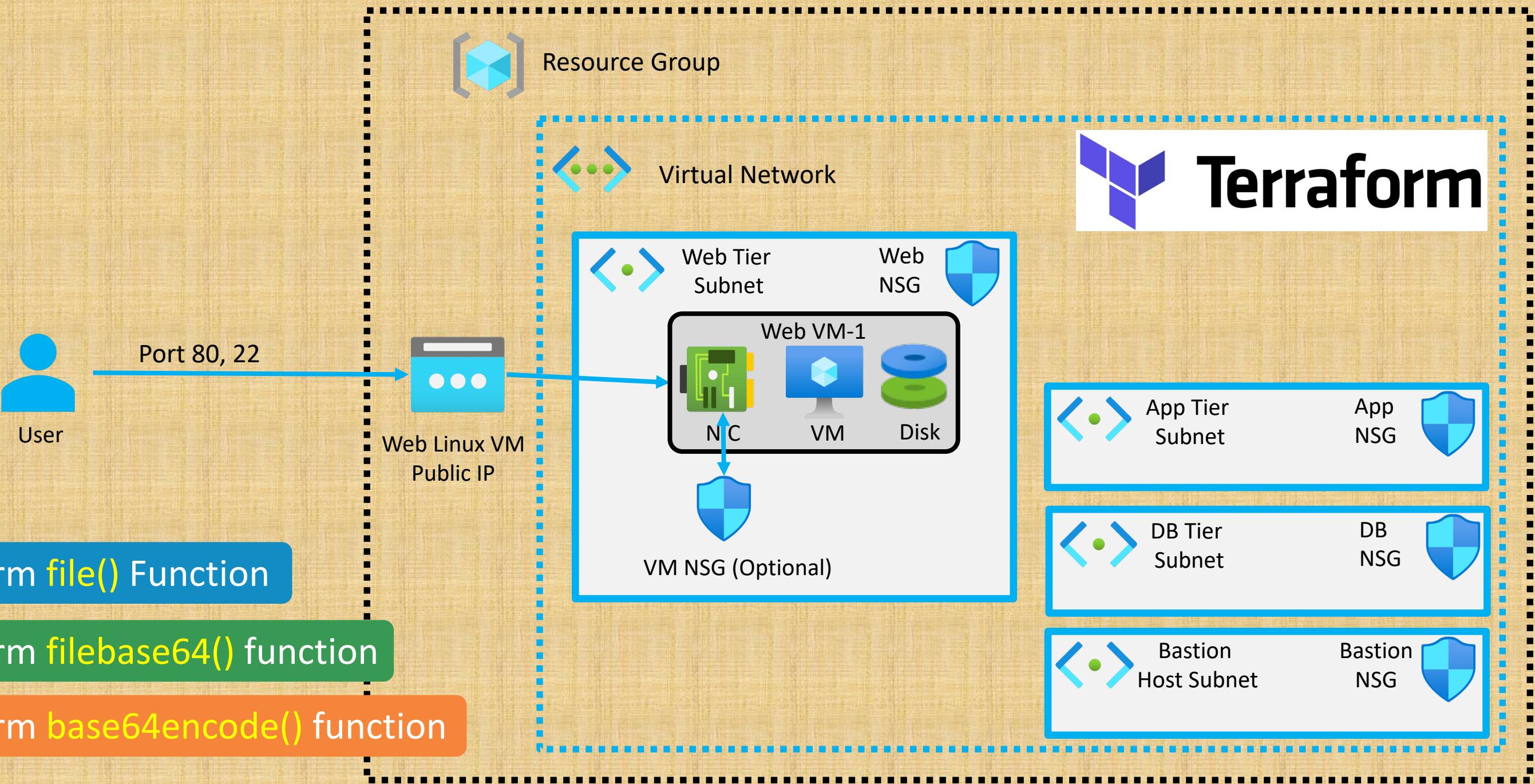
We are going to learn Terraform Fundamentals for 3 hours

Continue with Real-World Demos

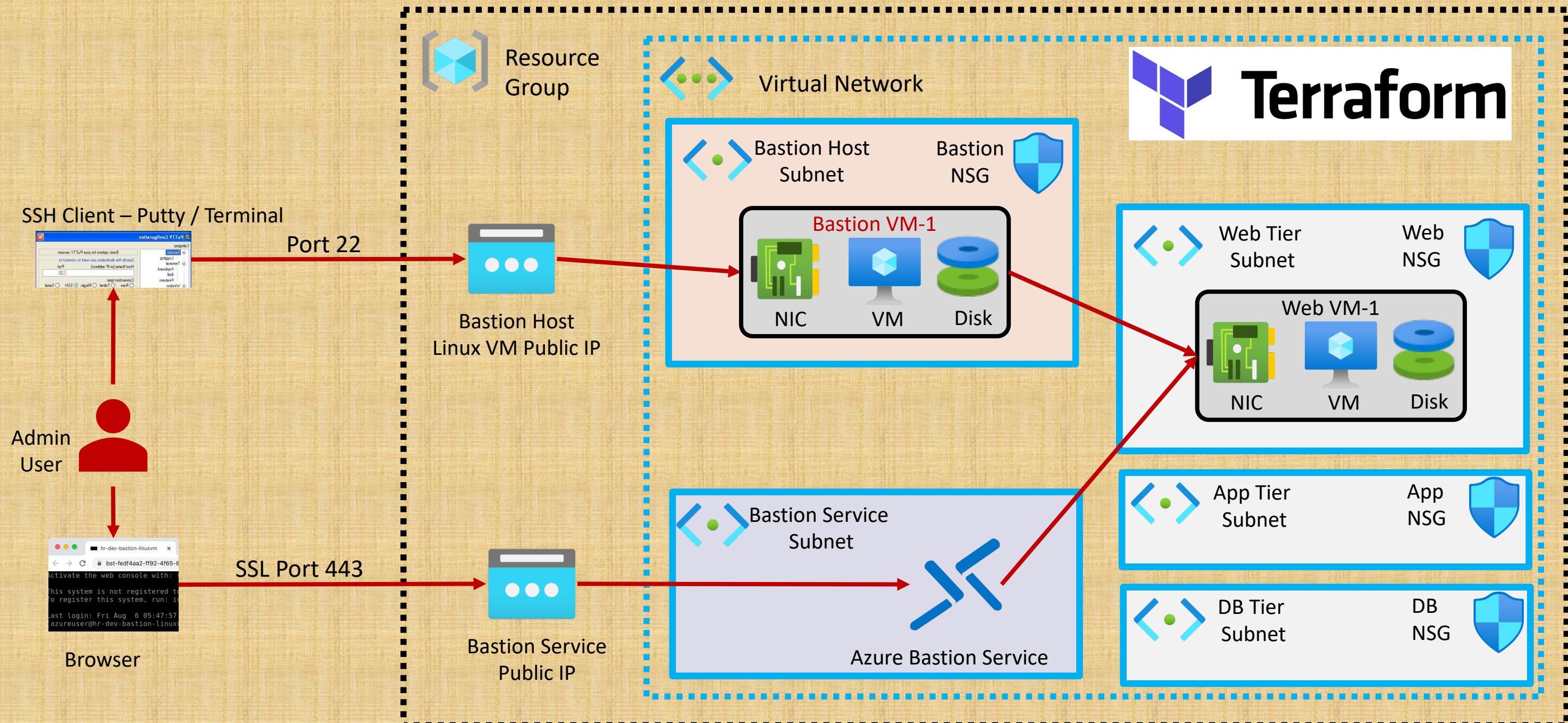
Azure Virtual Network – 4 Tier Design



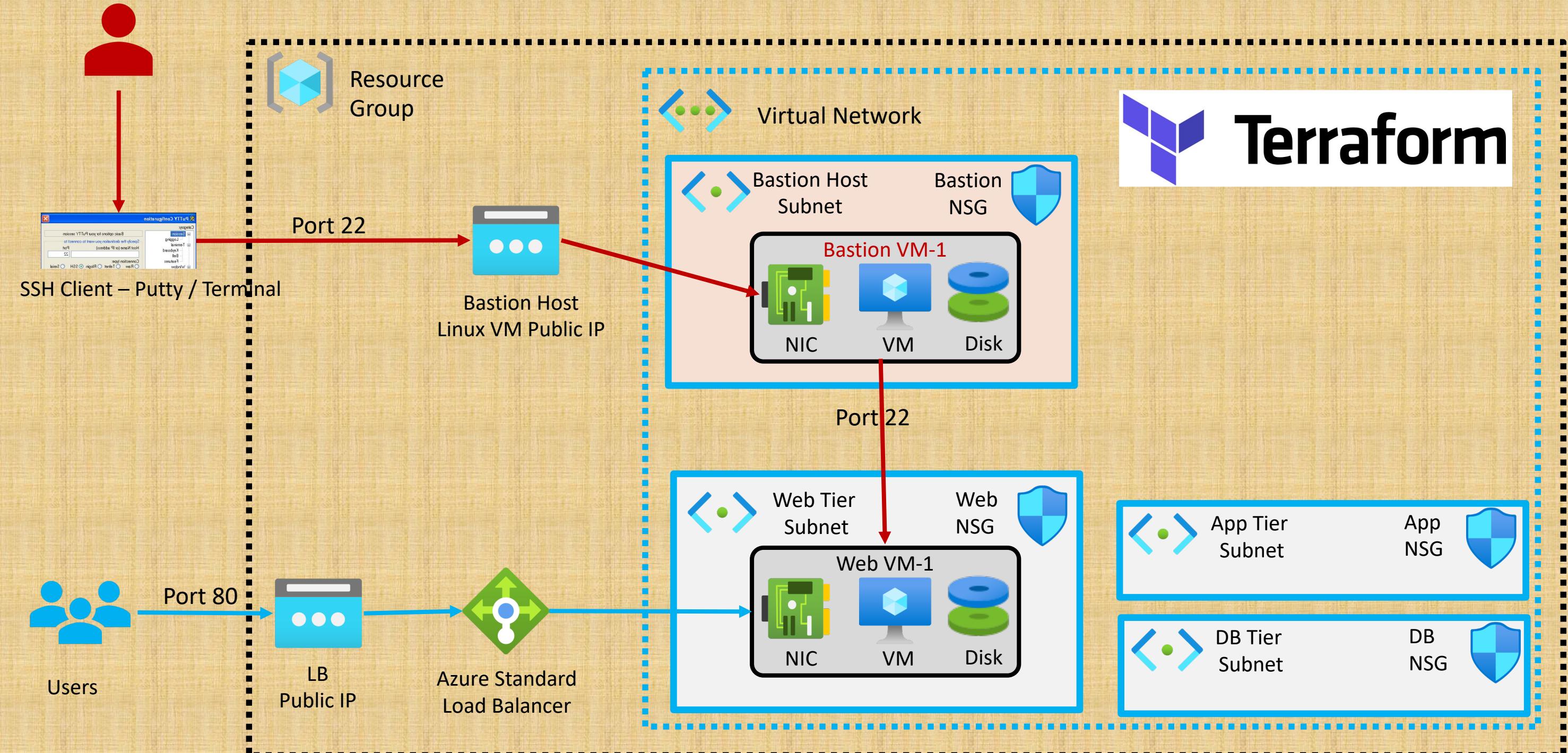
Azure Linux Virtual Machine

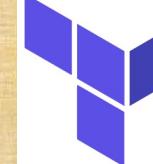


Azure Bastion Host Linux VM & Bastion Service

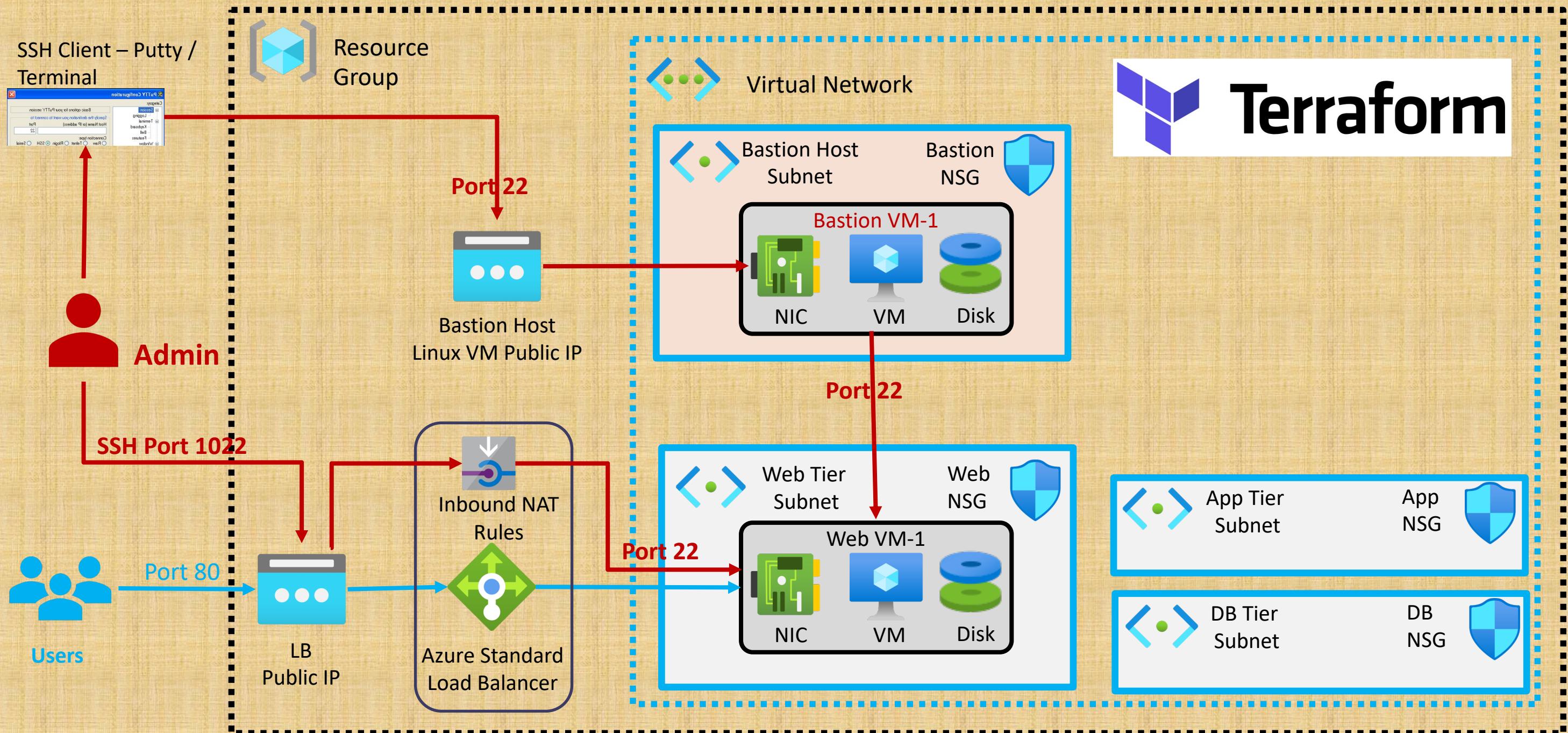


Azure Standard Load Balancer – Internet Facing

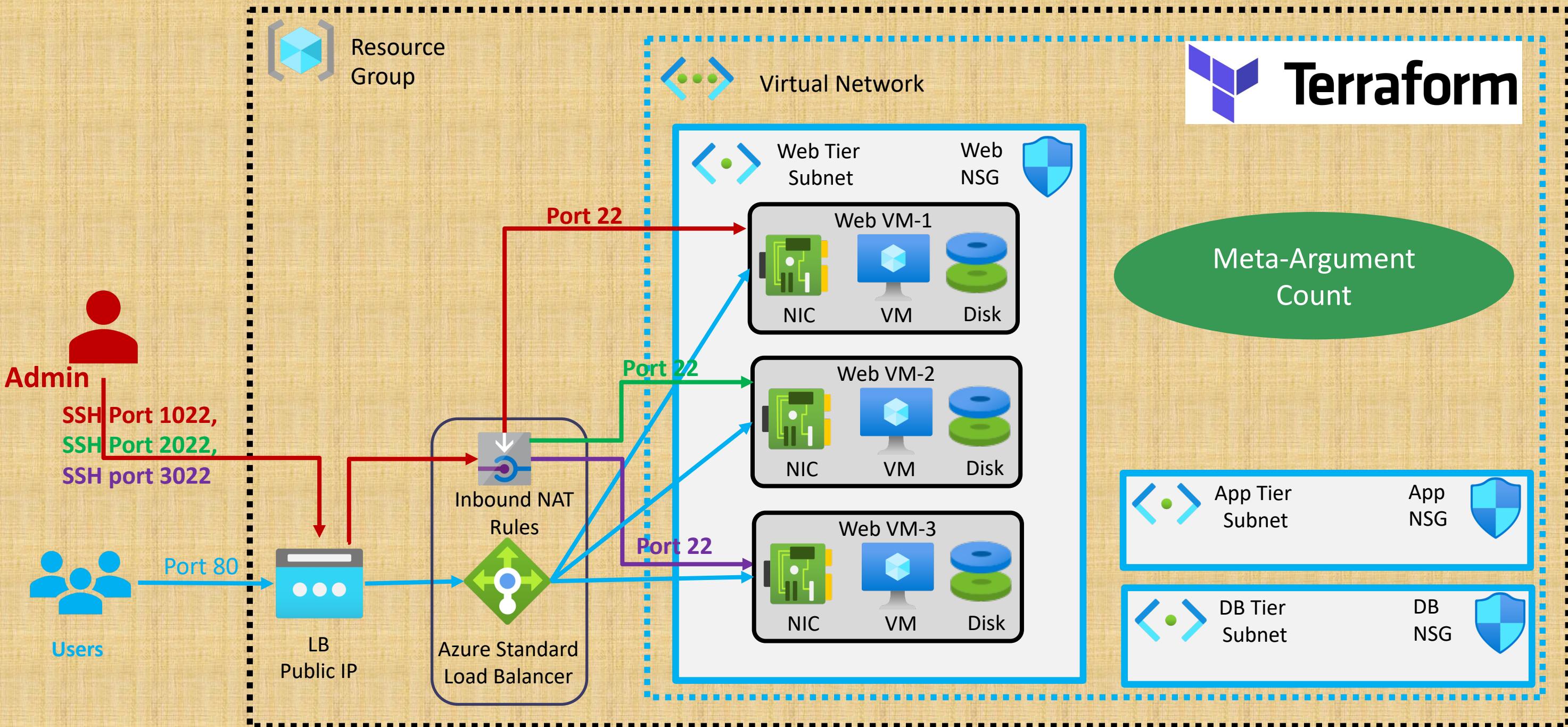


 **Terraform**

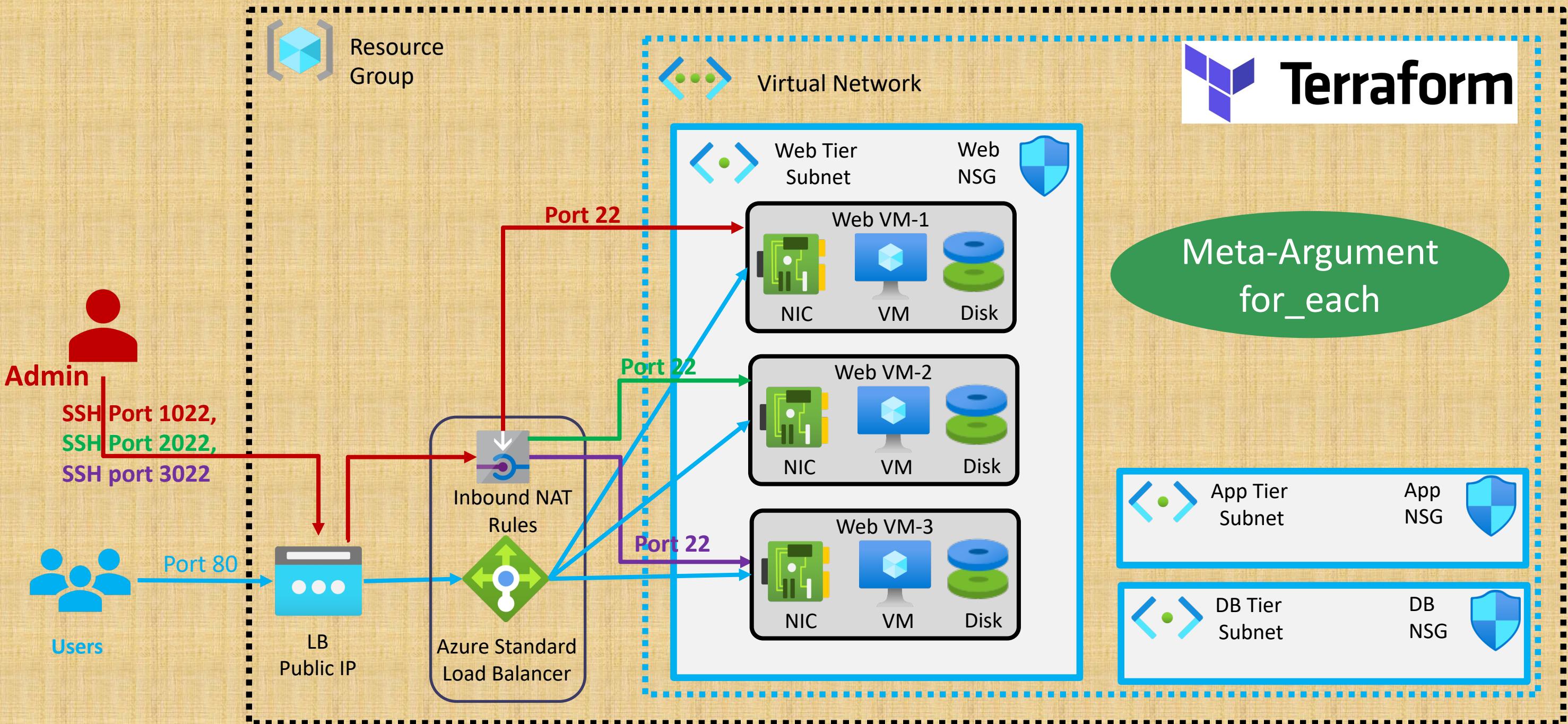
Azure Standard Load Balancer – Inbound NAT Rules



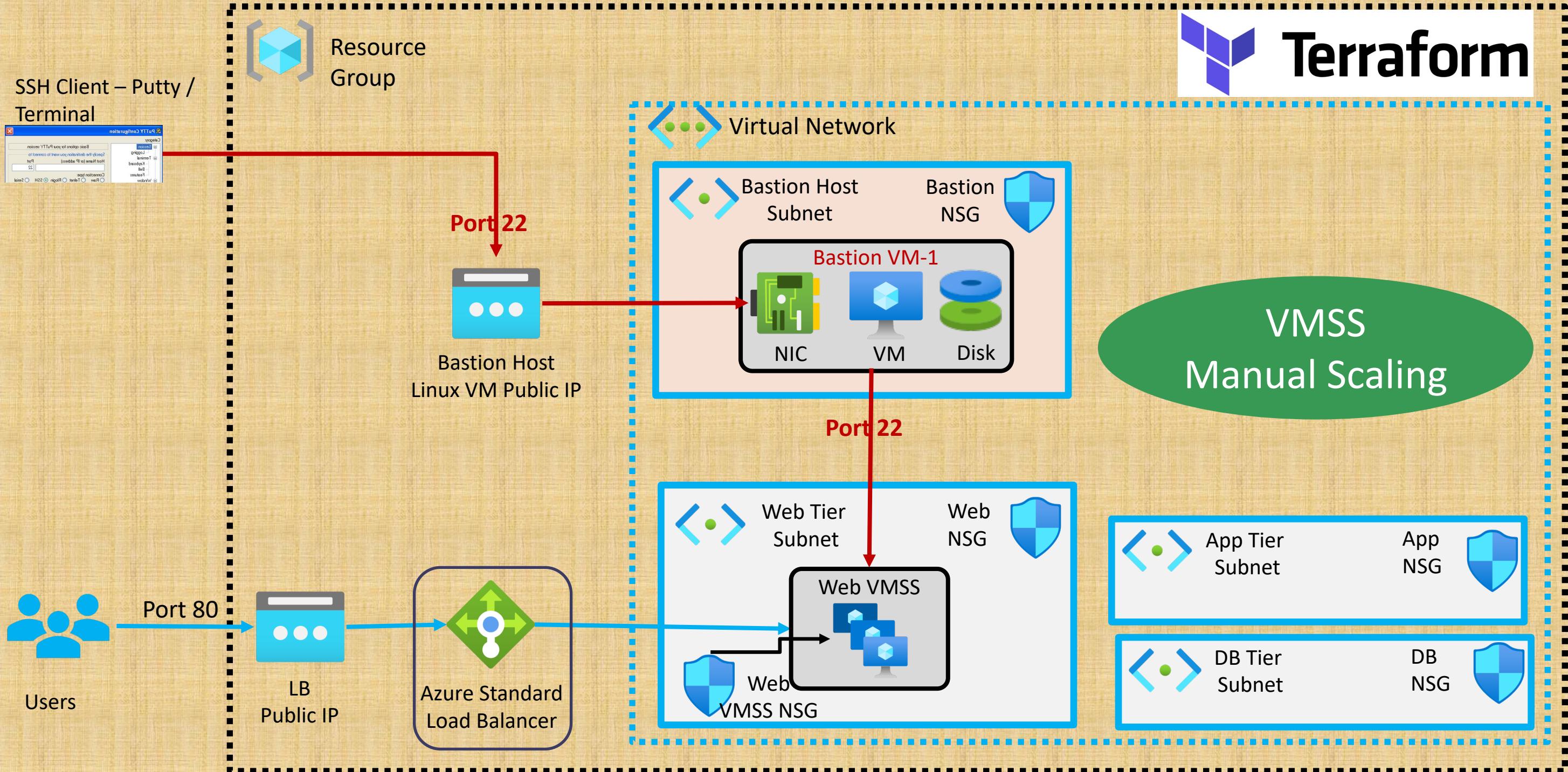
Azure Standard Load Balancer – Meta-Argument Count



Azure Standard Load Balancer – Meta-Argument for_each



Azure Standard Load Balancer – VMSS Manual Scaling



Azure VMSS - Autoscaling

Autoscaling Default Profile

Mandatory Profile

Defaults to round the clock schedule

Will not execute if Recurrence or Fixed Profile exists

P3

Autoscaling Recurrence Profile

Recur on those days with start time specified

Week Day and Weekend profiles

Business Hour and Non-Business Hour profile

P2

Autoscaling Fixed Profiles

Executes on that specific day

Fixed profile takes priority 1 for execution on that day if exists

P1

Priority Execution Order for Autoscaling Profiles

Autoscaling Default Profile

Default* default 

 Delete warning  The very last or default recurrence rule cannot be deleted. Instead, you can disable autoscale to turn off autoscale.

Scale mode Scale based on a metric Scale to a specific instance count

Rules

Scale out

When	hr-dev-web-vmss	(Average) Percentage CPU > 75	Increase count by 1
Or	hr-dev-web-vmss	(Average) Available Memory By...	Increase count by 1
Or	hr-dev-web-lb	(Average) SYNCount > 10	Increase count by 1

Scale in

When	hr-dev-web-vmss	(Average) Percentage CPU < 25	Decrease count by 1
And	hr-dev-web-vmss	(Average) Available Memory By...	Decrease count by 1
And	hr-dev-web-lb	(Average) SYNCount < 10	Decrease count by 1

+ Add a rule

Instance limits

Minimum 	Maximum 	Default 
2 	6 	2 

Schedule  This scale condition is executed when none of the other scale condition(s) match

Scale Out

Scale In

Default Profile

Autoscaling Recurrence Profile Week Days

profile-2-weekdays 



Scale mode Scale based on a metric Scale to a specific instance count

Rules

Scale out

When	hr-dev-web-vmss	(Average) Percentage CPU > 75	Increase count by 1
Or	hr-dev-web-vmss	(Average) Available Memory By...	Increase count by 1
Or	hr-dev-web-lb	(Average) SYNCCount > 10	Increase count by 1

Scale in

When	hr-dev-web-vmss	(Average) Percentage CPU < 25	Decrease count by 1
And	hr-dev-web-vmss	(Average) Available Memory By...	Decrease count by 1
And	hr-dev-web-lb	(Average) SYNCCount < 10	Decrease count by 1

+ Add a rule

Instance limits

Minimum 	4	
Maximum 	20	
Default 	4	

Schedule

Specify start/end dates Repeat specific days

Repeat every

Monday Tuesday Wednesday Thursday
Friday Saturday Sunday

Timezone

Start time

End time

Specify an end time, else this scale condition will apply for all days until it reaches the start time of another scale condition

Recurrence Week Days

Scale mode

 Scale based on a metric Scale to a specific instance count

Rules

Scale out

When	hr-dev-web-vmss	(Average) Percentage CPU > 75	Increase count by 1
Or	hr-dev-web-vmss	(Average) Available Memory By...	Increase count by 1
Or	hr-dev-web-lb	(Average) SYNCCount > 10	Increase count by 1

Scale in

When	hr-dev-web-vmss	(Average) Percentage CPU < 25	Decrease count by 1
And	hr-dev-web-vmss	(Average) Available Memory By...	Decrease count by 1
And	hr-dev-web-lb	(Average) SYNCCount < 10	Decrease count by 1

[+ Add a rule](#)

Instance limits

Minimum 3 Maximum 6 Default 3 

Schedule

Repeat every

Timezone

Start time

End time

 Specify start/end dates Repeat specific days

- Monday Tuesday Wednesday Thursday
 Friday Saturday Sunday

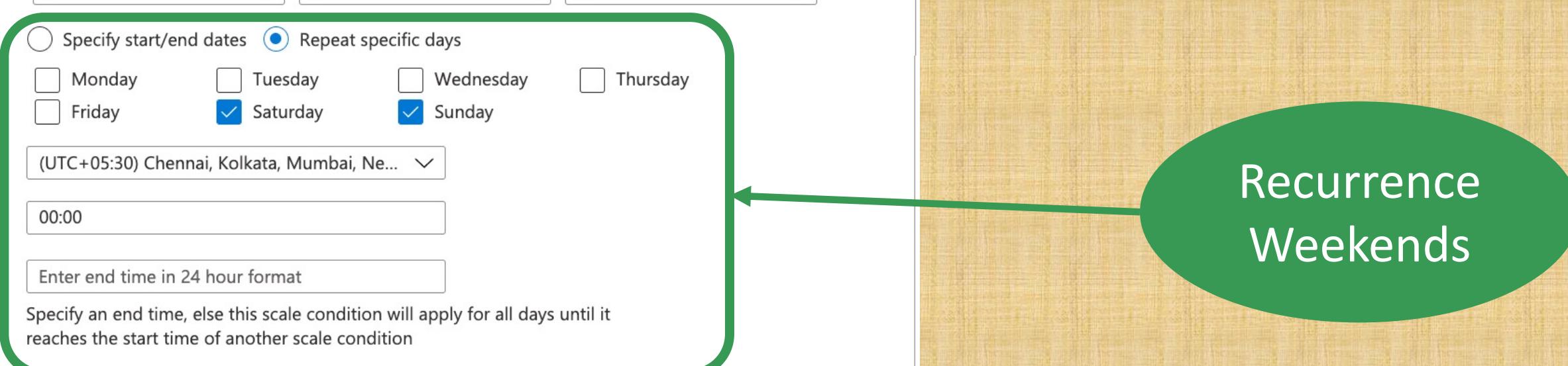
(UTC+05:30) Chennai, Kolkata, Mumbai, Ne...

00:00

Enter end time in 24 hour format

Specify an end time, else this scale condition will apply for all days until it reaches the start time of another scale condition

Autoscaling Recurrence Profile Weekends



Recurrence Weekends

Autoscaling Fixed Profile

profile-4-fixed-profile 



Scale mode Scale based on a metric Scale to a specific instance count

Rules

Scale out

When	hr-dev-web-vmss	(Average) Percentage CPU > 75	Increase count by 1
Or	hr-dev-web-vmss	(Average) Available Memory By...	Increase count by 1
Or	hr-dev-web-lb	(Average) SYNCCount > 10	Increase count by 1

Scale in

When	hr-dev-web-vmss	(Average) Percentage CPU < 25	Decrease count by 1
And	hr-dev-web-vmss	(Average) Available Memory By...	Decrease count by 1
And	hr-dev-web-lb	(Average) SYNCCount < 10	Decrease count by 1

+ Add a rule

Instance limits

Minimum 	40 	Default 	6 
---	--	---	---

Schedule

Timezone 

Start date 

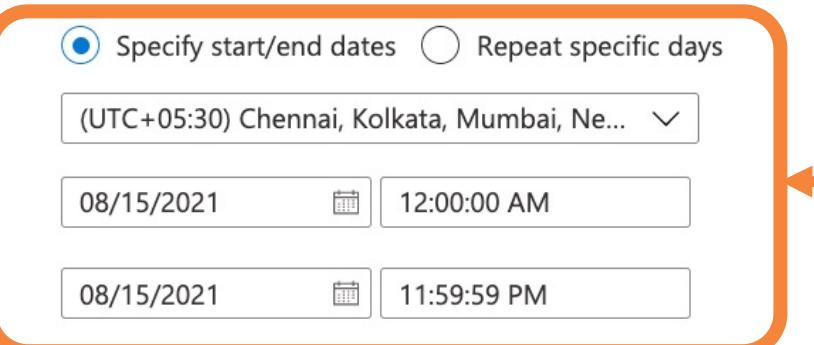
End date 

Specify start/end dates Repeat specific days

(UTC+05:30) Chennai, Kolkata, Mumbai, Ne...

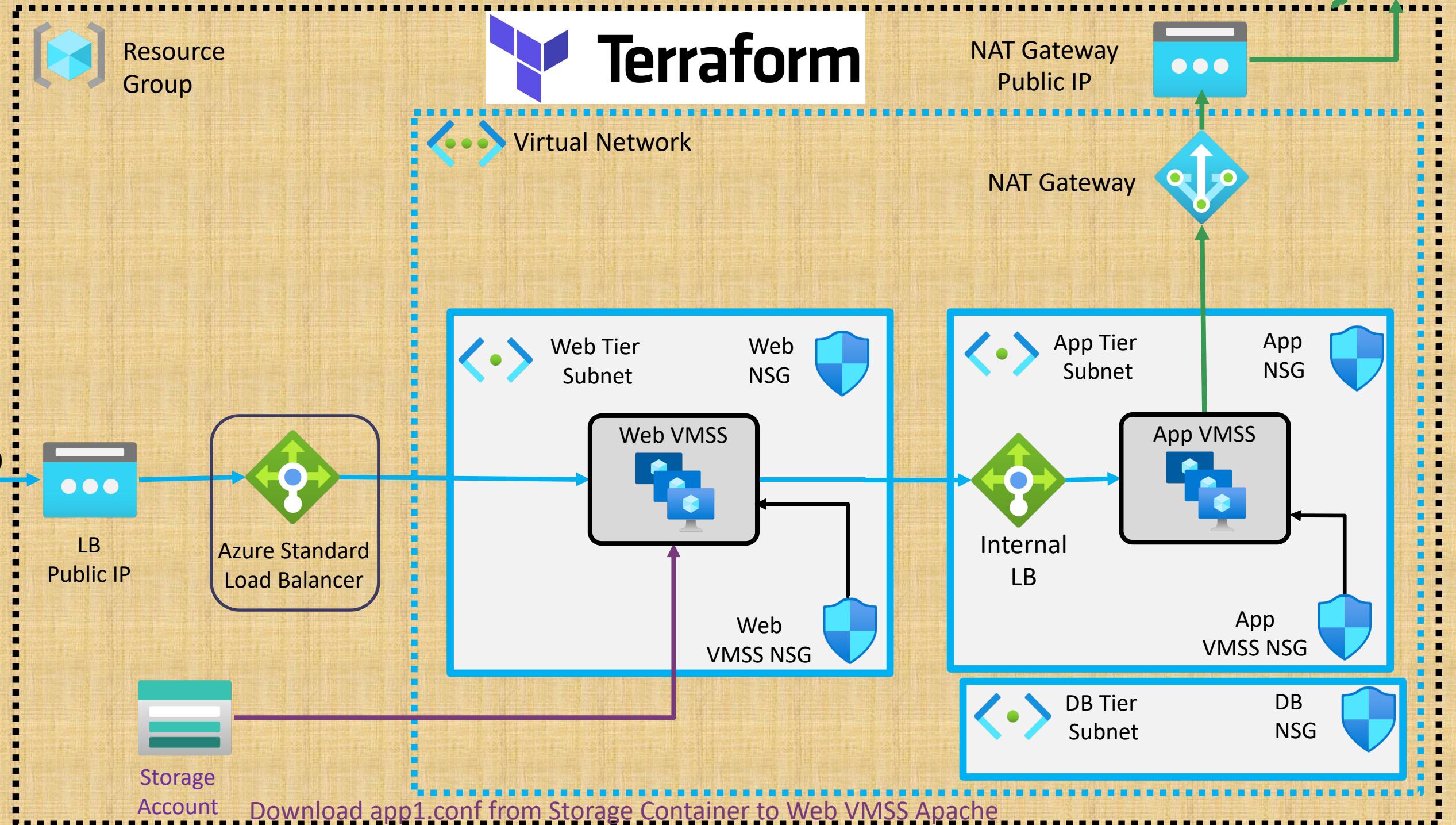
08/15/2021 12:00:00 AM

08/15/2021 11:59:59 PM



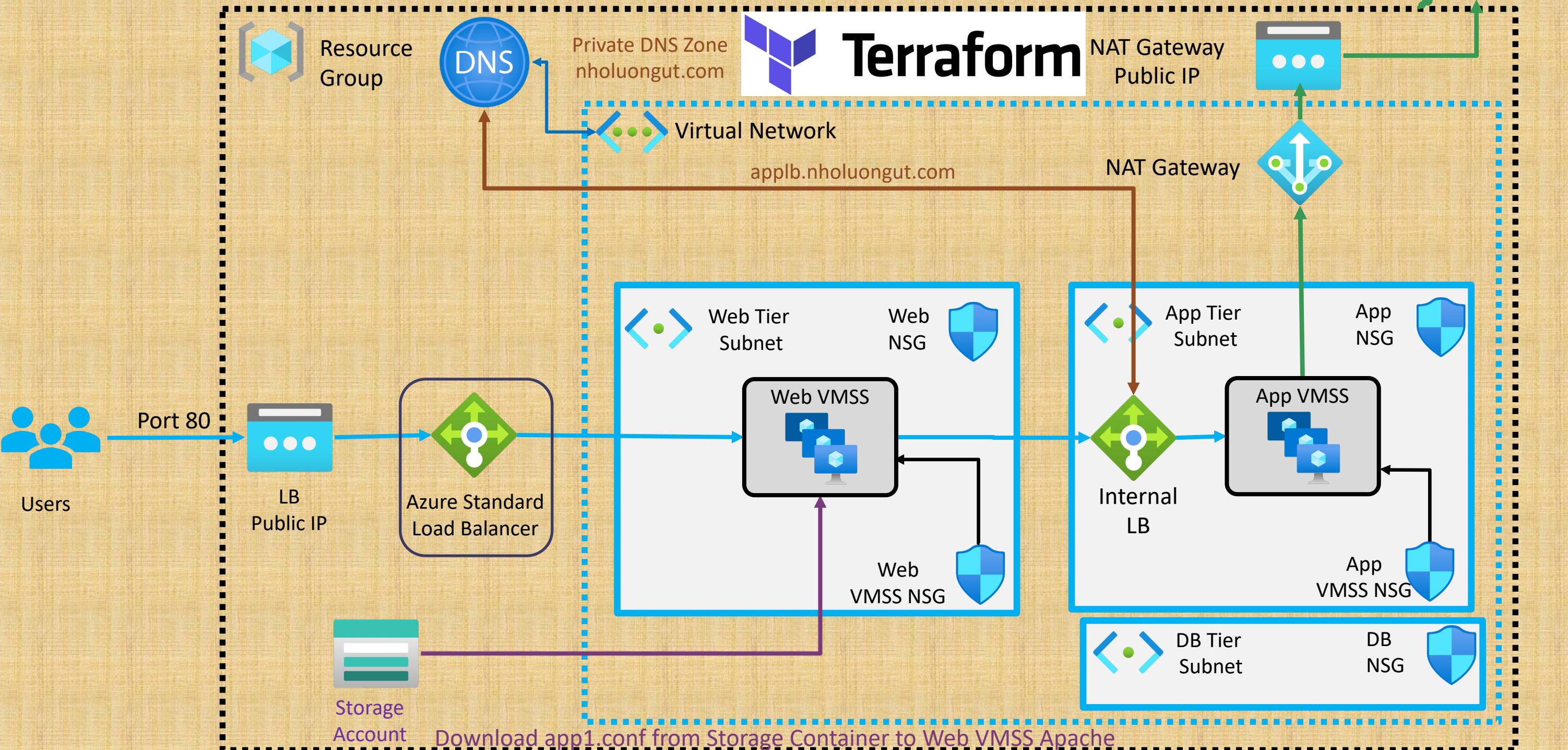
Azure - External LB + Web VMSS + Internal LB + App VMSS

Internet



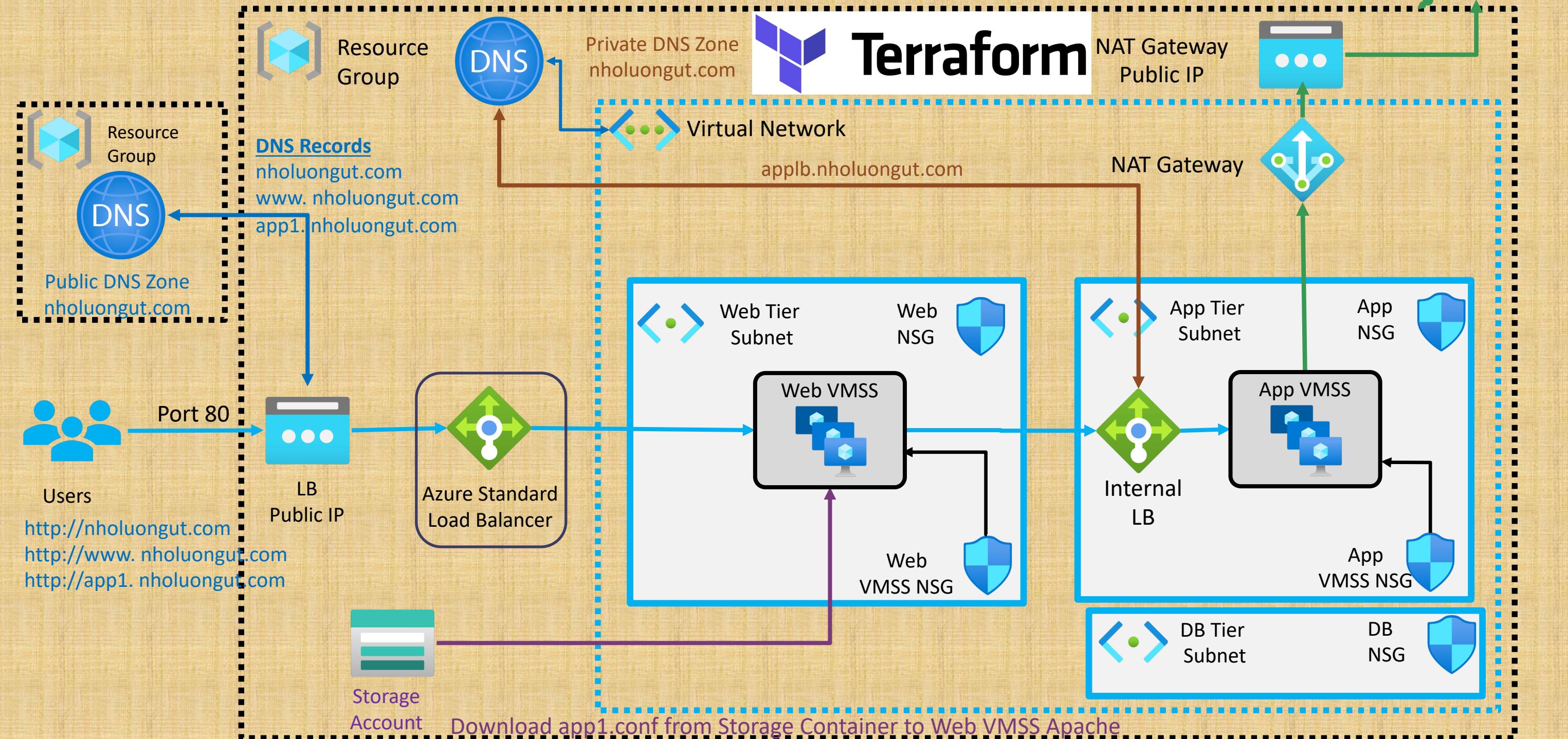
Azure - Private DNS Zones

Terraform



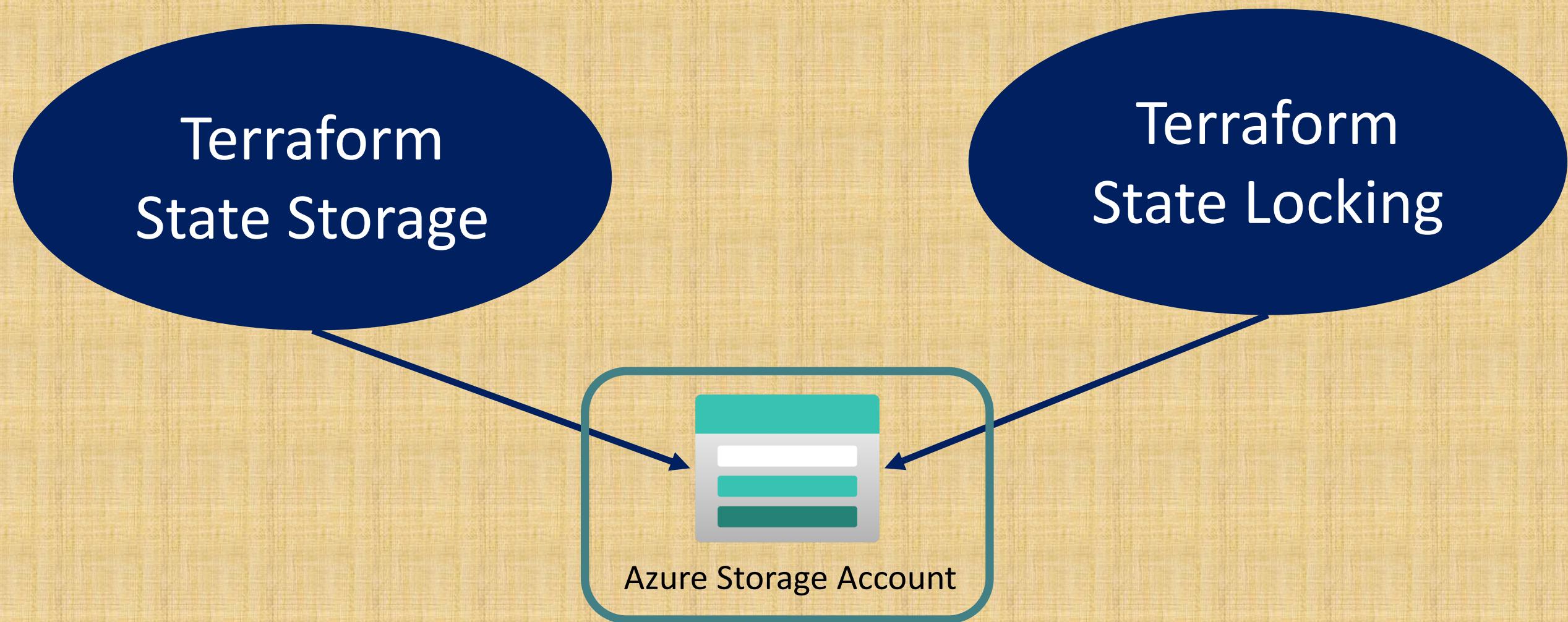
Azure - Public DNS Zones

Terraform

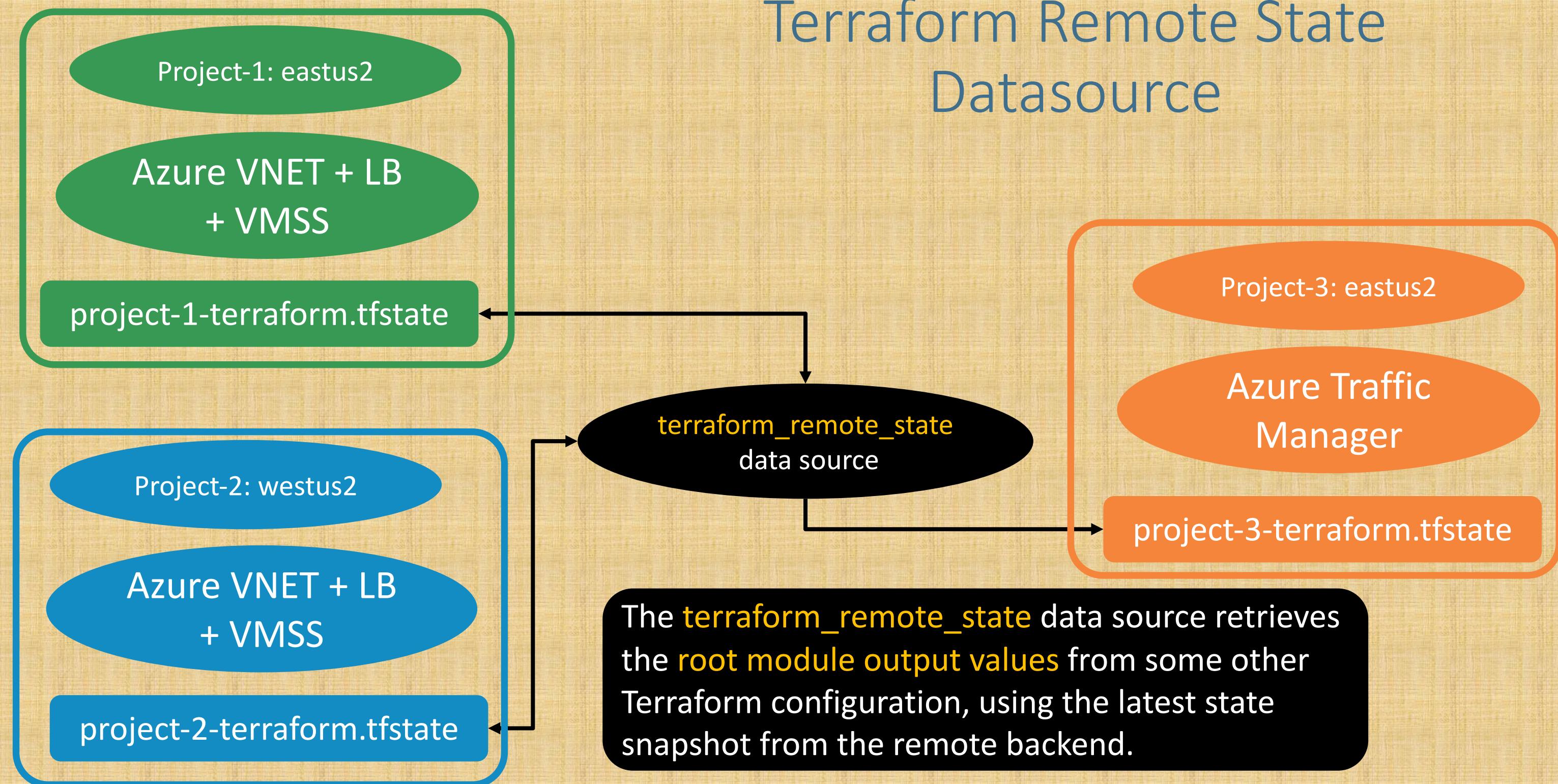


What is Terraform Backend ?

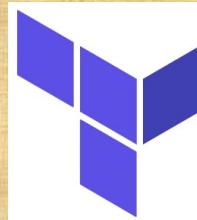
Backends are responsible for storing state and providing an API for state locking.



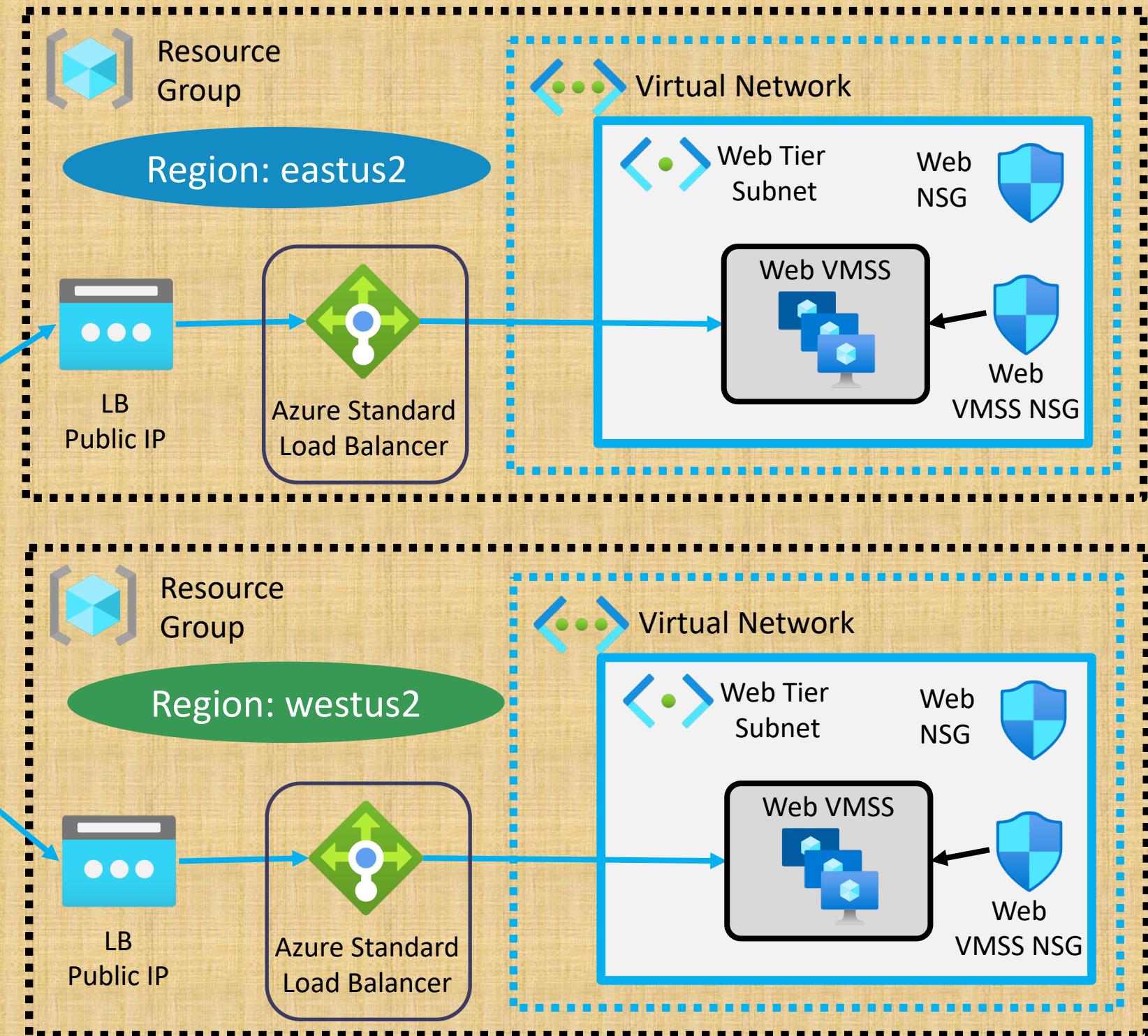
Terraform Remote State Datasource



Azure – Traffic Manager



Terraform



Create Azure Application
Gateway manually

AG – Backend Pools

AG – Frontend IP Configs

AG – Listeners

AG – HTTP Settings

AG – Rules

AG – Health Probes

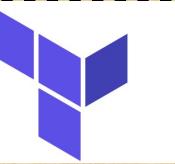


Port 80

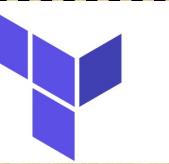


LB Public IP

Azure Application Gateway – using Azure Portal



Resource
Group

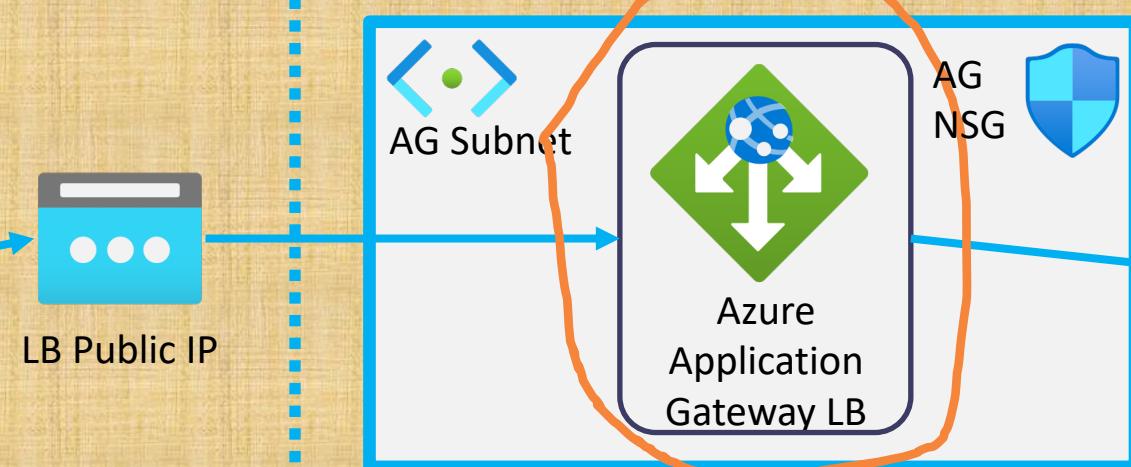
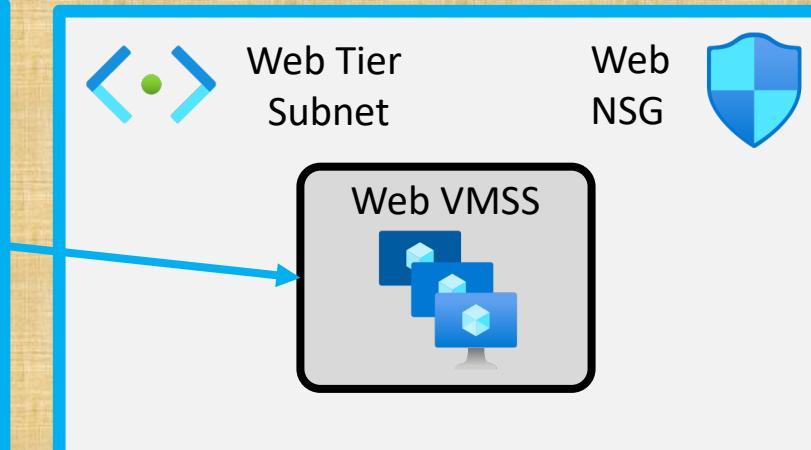
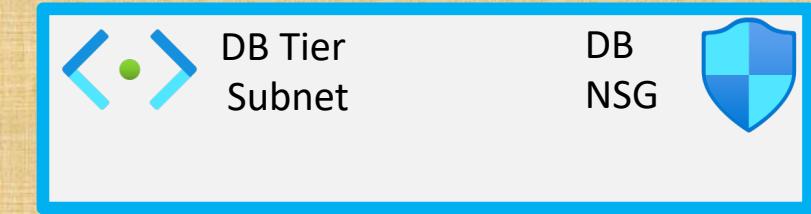
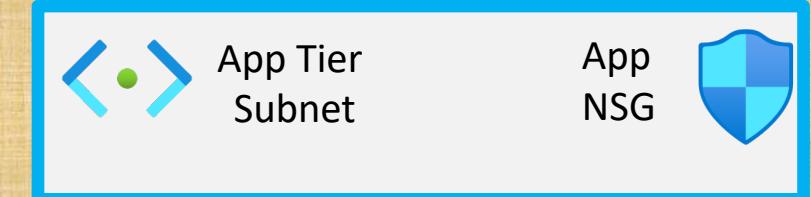


Terraform



Virtual Network

Create AG using
Azure Portal



Create Azure Application Gateway using Terraform

AG – Backend Pools

AG – Frontend IP Configs

AG – Listeners

AG – HTTP Settings

AG – Rules

AG – Health Probes



Users

Port 80



LB Public IP

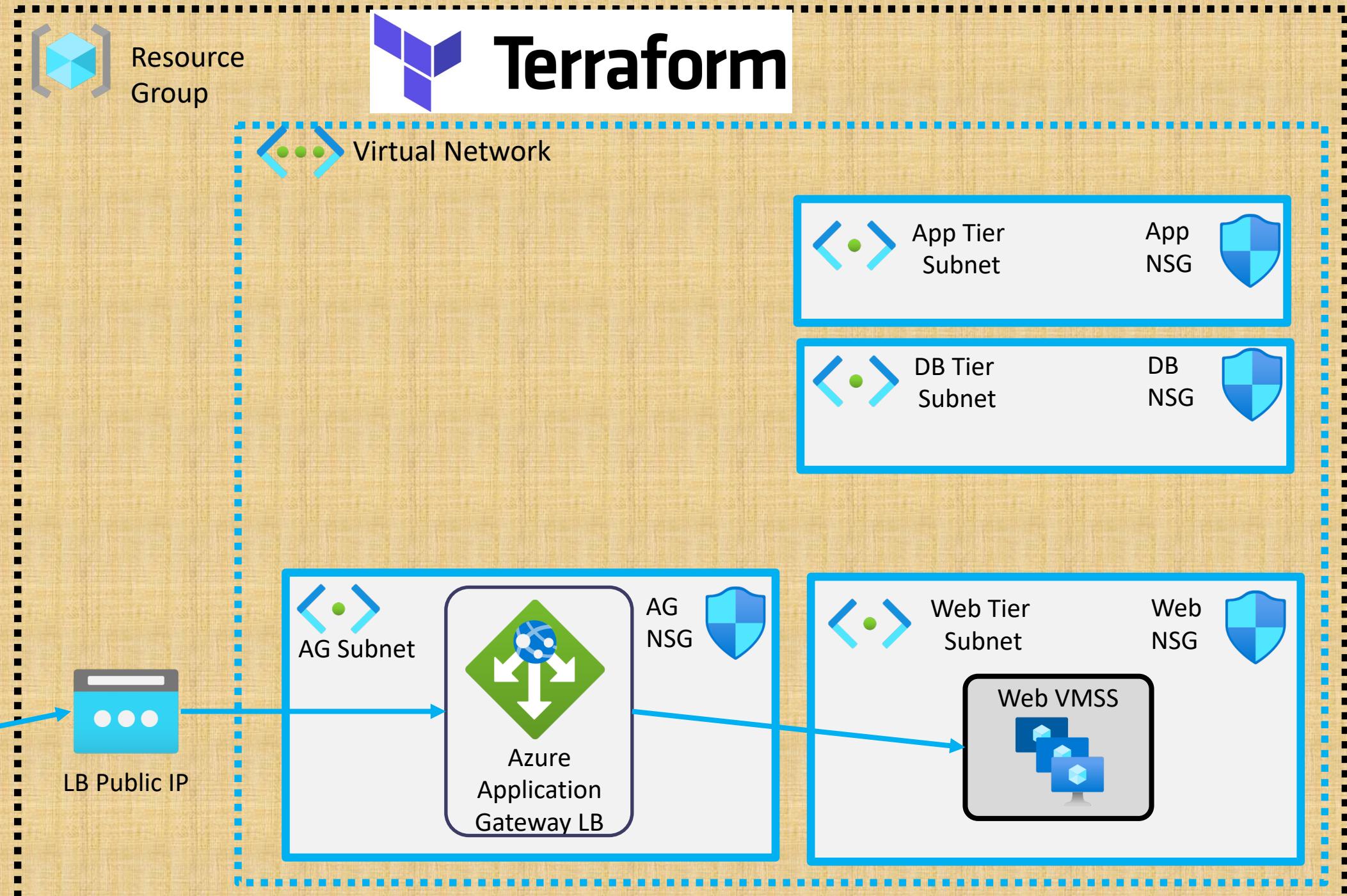
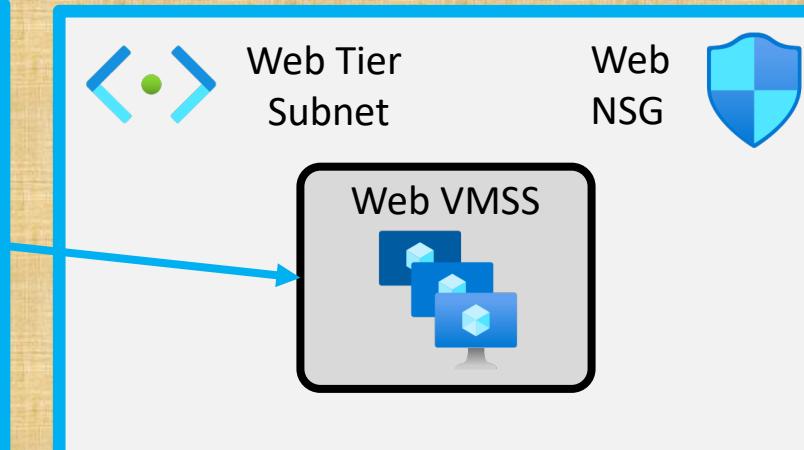
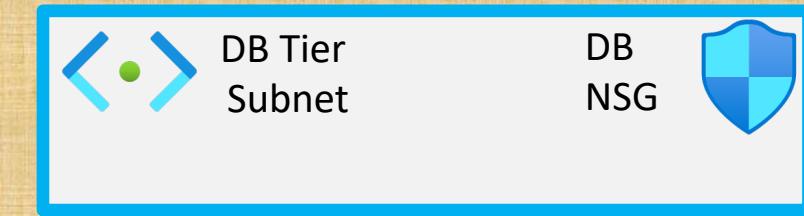
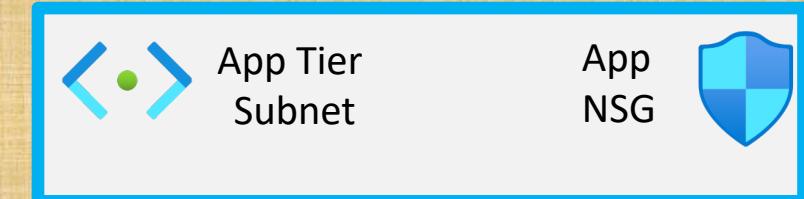
Azure Application Gateway – using Terraform



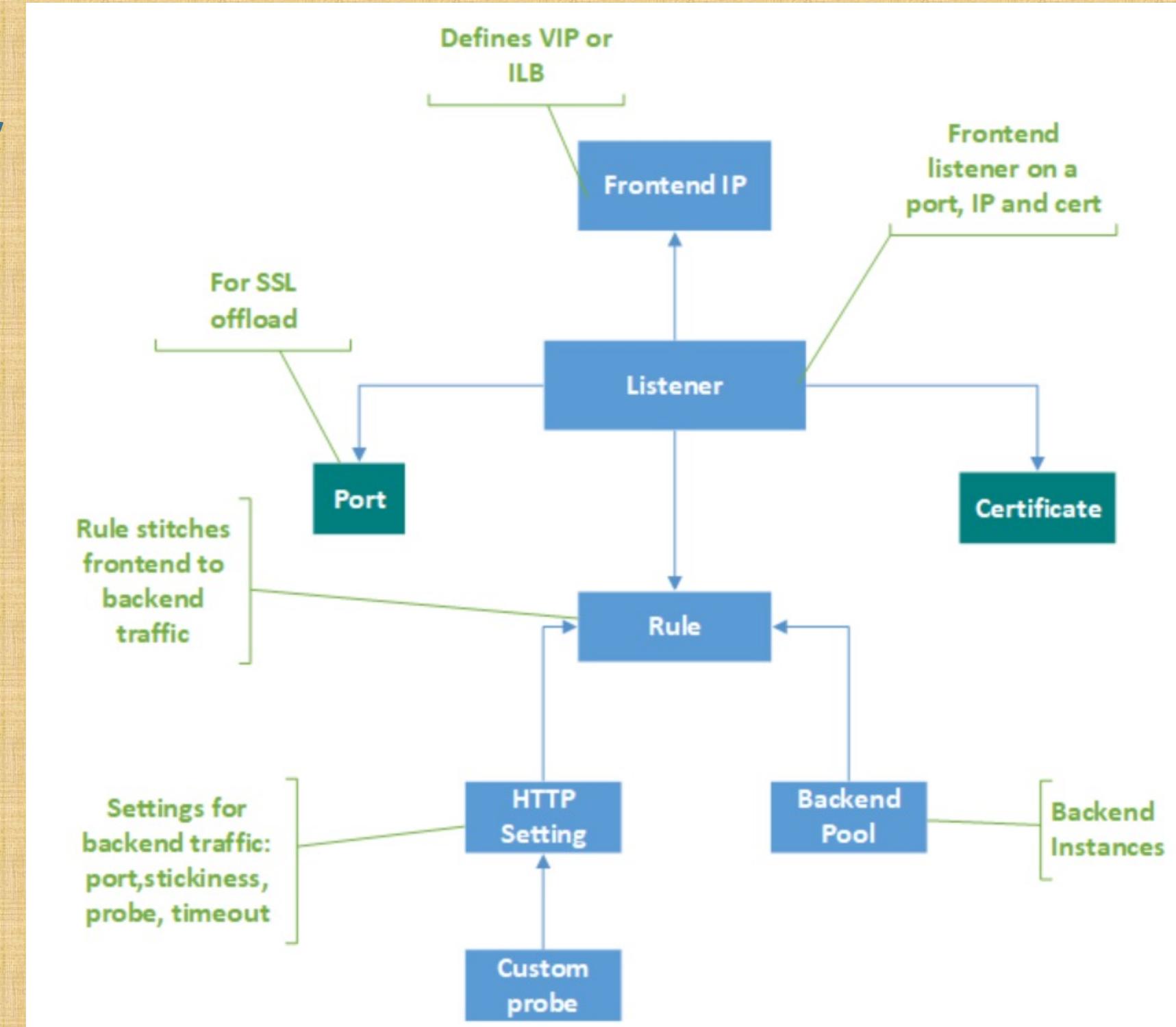
Terraform

Resource Group

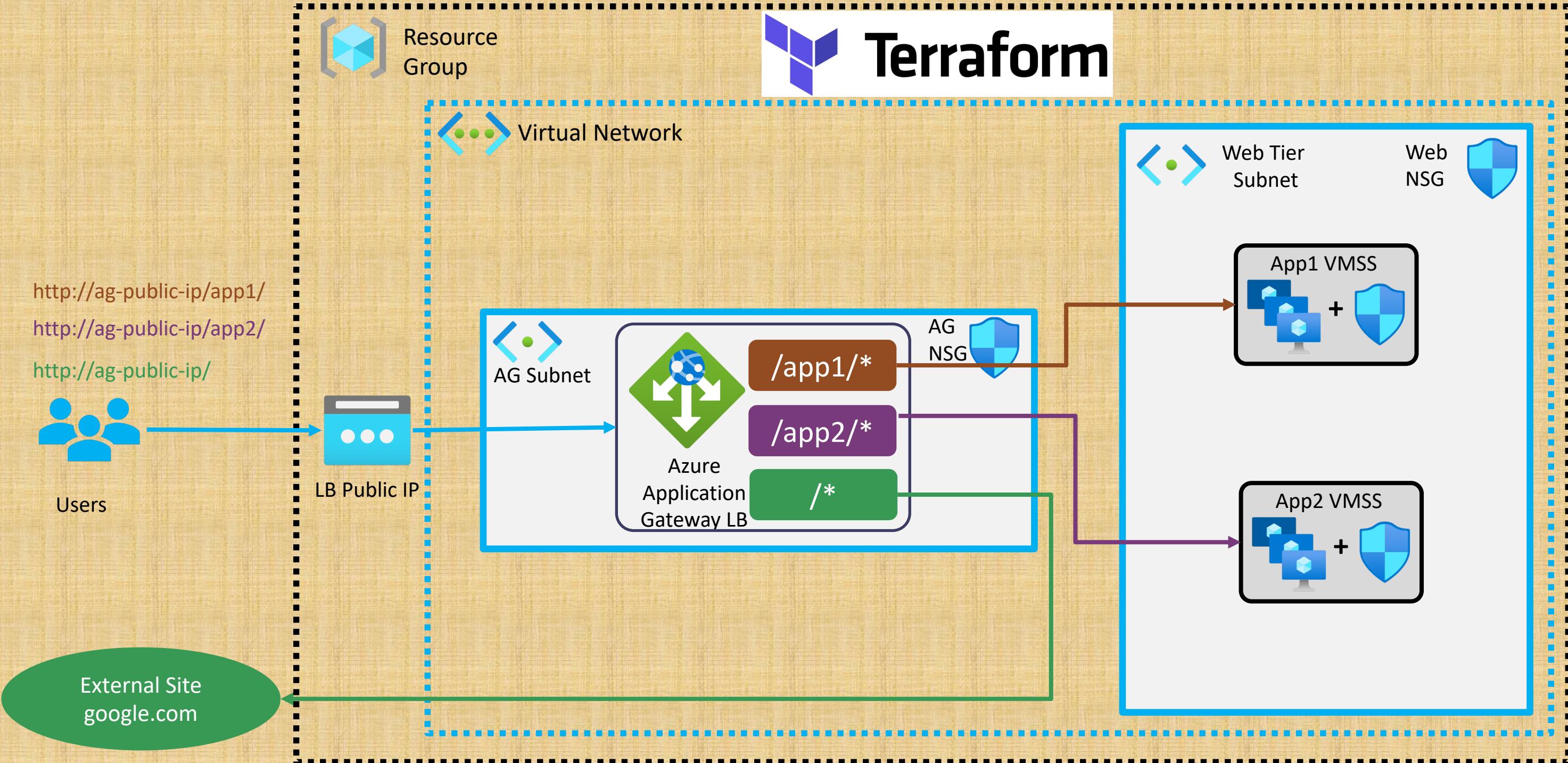
Virtual Network



Azure Application Gateway Components



Azure Application Gateway – Context Path based Routing



Path Based Routing

Root Context
External
Redirect

Context Path
based Routing

hr-dev-vnet-rqrt-1

hr-dev-web-ag

* Listener * Backend targets

Choose a backend pool to which this routing rule will send traffic. You will also need to specify a set of HTTP settings that define the behavior of the routing rule.

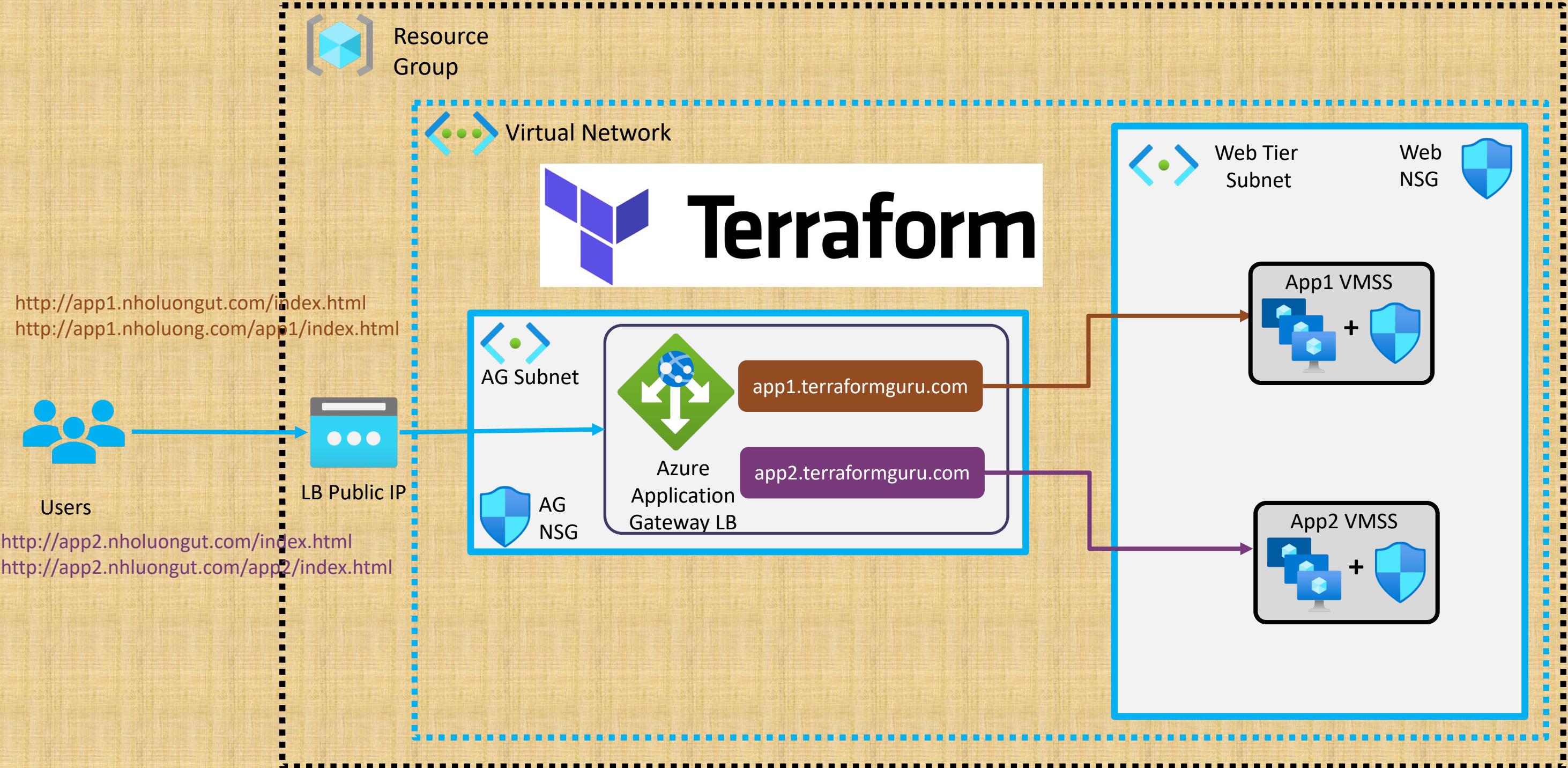
Path-based routing

You can route traffic from this rule's listener to different backend targets based on the URL path of the request. You can also apply a different set of HTTP settings based on the URL path.

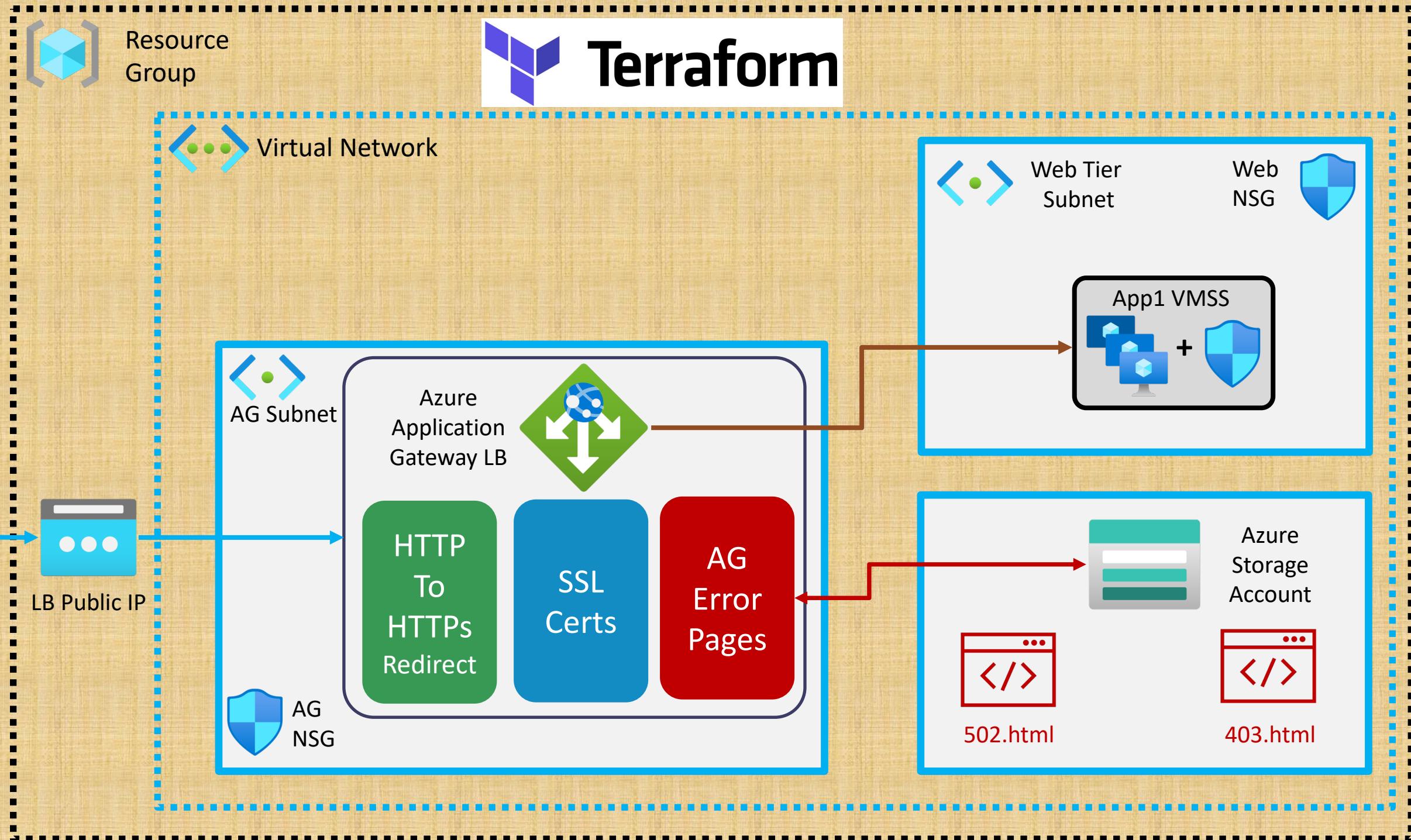
Path based rules

Path	Target name	HTTP setting name	Backend pool	...
/app1/*	app1-rule	hr-dev-vnet-be-htst-app1	hr-dev-vnet-beap-app1	...
/app2/*	app2-rule	hr-dev-vnet-be-htst-app2	hr-dev-vnet-beap-app2	...

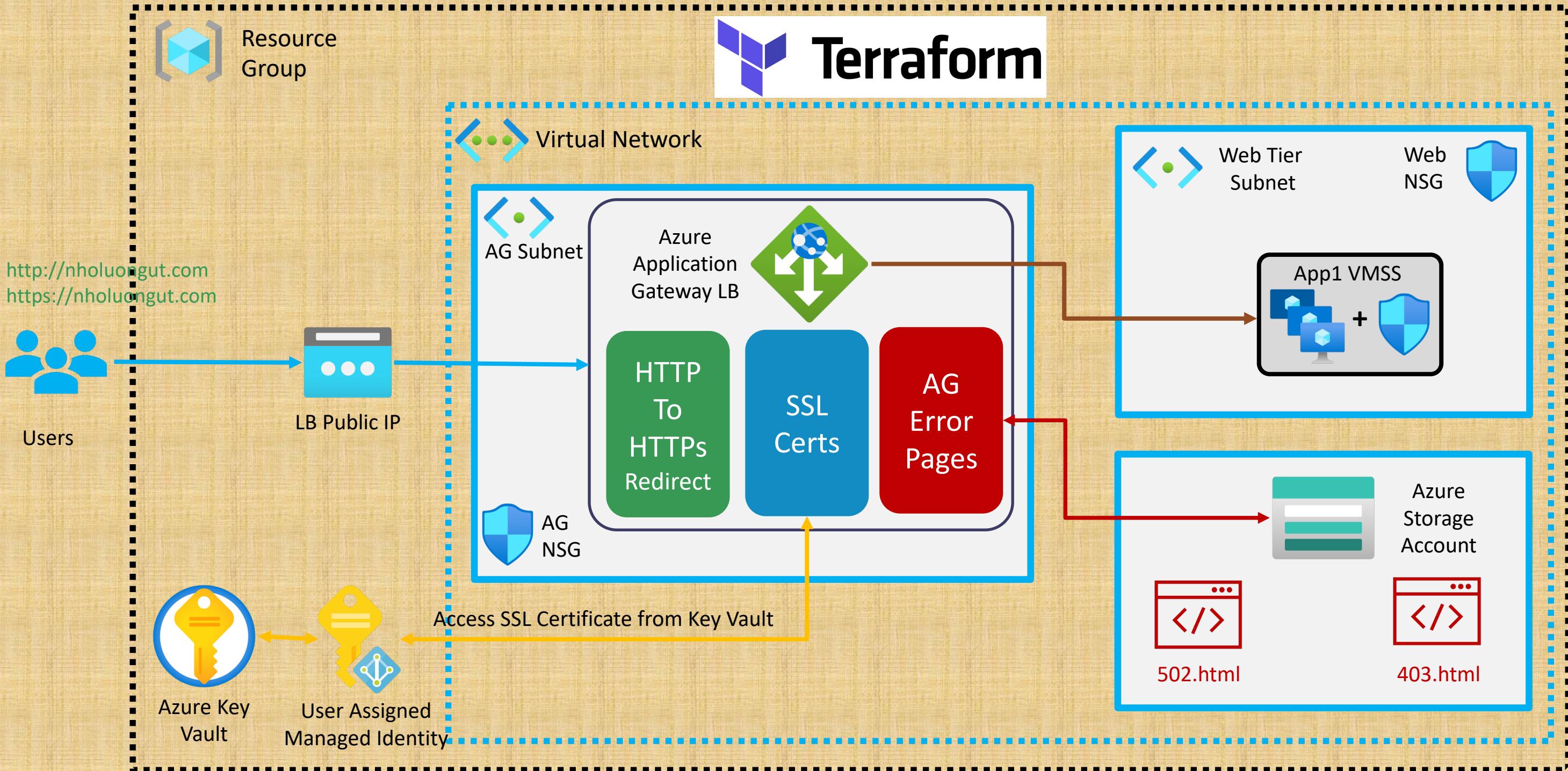
Azure Application Gateway – Multisite Hosting



Azure Application Gateway – SSL Self-signed

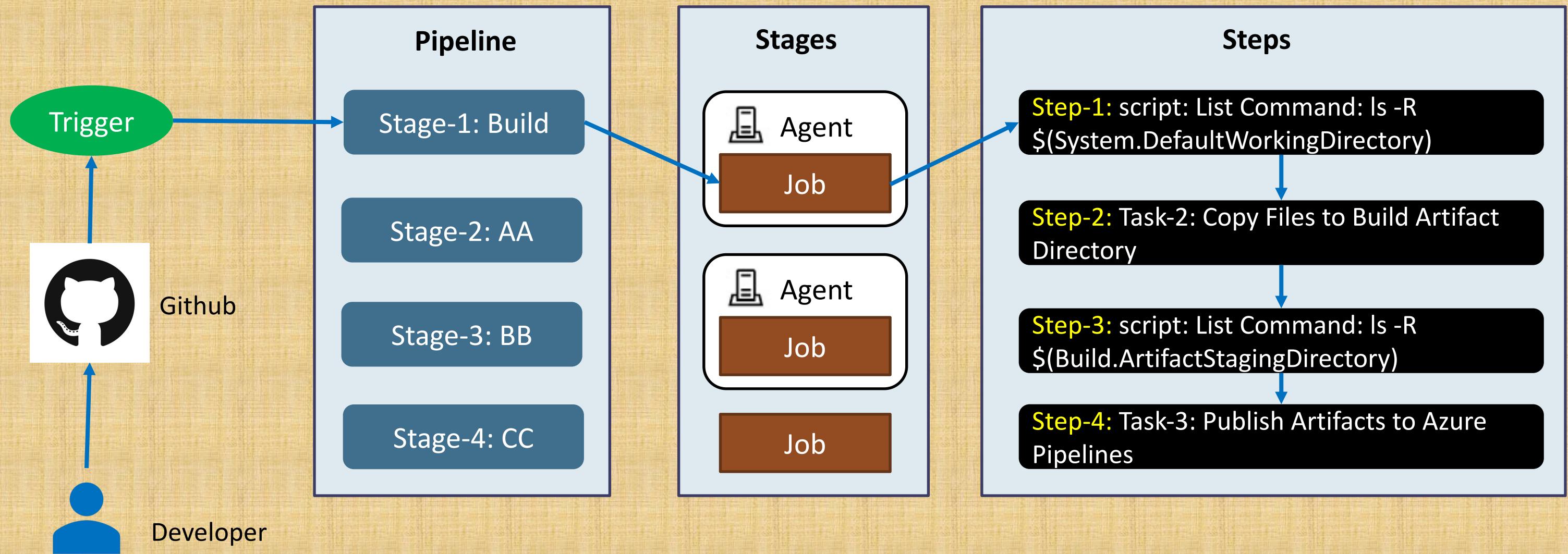


Azure Application Gateway – SSL from Key Vault





Azure Pipelines – Key Concepts



Azure DevOps – Build Pipeline

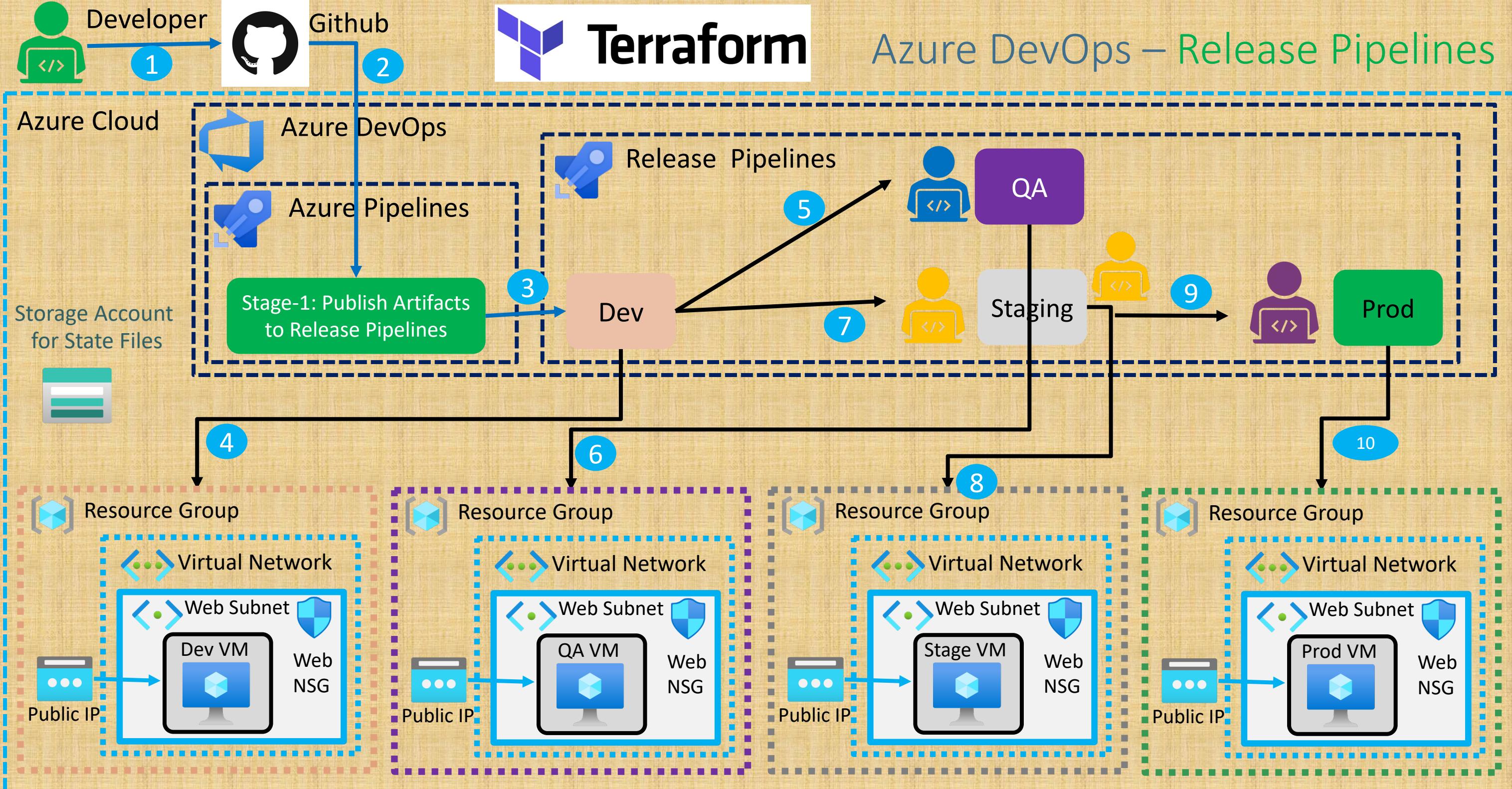
Continuous Integration Pipeline

Task-1

Copy files (terraform-manifests folder) from System default working Directory to Build Artifact Directory

Task-2

Publish Build Artifacts to Azure Pipelines, so that we can use them in Release Pipelines



Azure IaC DevOps – Release Pipeline

terraform-on-azure-w... + ⌂ Help

Overview Boards Repos Pipelines Pipelines Environments Releases Library Task groups Deployment groups Test Plans Artifacts

↑ Terraform-CD > Release-6

Pipeline Variables History + Deploy Cancel Refresh Edit ...

Release

Continuous deployment for Stack Simplify 25/08/2021, 15:32

Artifacts

_Terraform Contin... 20210825.8 main

Stages

```
graph LR; Dev[Dev] --> QA[QA]; QA --> Stage[Stage]; Stage --> Prod[Prod]
```

Dev ✓ Succeeded on 25/08/2021, 15:36

QA ✓ Succeeded on 25/08/2021, 15:40

Stage ✓ Succeeded on 25/08/2021, 15:46

Prod ✓ Succeeded on 25/08/2021, 15:50

The screenshot shows the Azure DevOps Release Pipeline interface. On the left, there's a sidebar with various navigation options like Overview, Boards, Repos, Pipelines, Environments, Releases, Library, Task groups, Deployment groups, Test Plans, and Artifacts. The Pipelines option is selected. In the center, the pipeline details for 'terraform-on-azure-w...' are shown under 'Terraform-CD > Release-6'. The 'Pipeline' tab is active. A 'Release' section on the left displays a 'Continuous deployment' card for 'Stack Simplify' on 25/08/2021, 15:32, with an artifact named '_Terraform Contin... 20210825.8' in the 'main' branch. To the right, the 'Stages' section shows a sequential flow of stages: Dev, QA, Stage, and Prod. Each stage is represented by a card with a green success icon and the word 'Succeeded'. The stages are timestamped: Dev at 15:36, QA at 15:40, Stage at 15:46, and Prod at 15:50. The Stage stage is explicitly labeled 'Stage'.

Azure IaC DevOps Release Pipelines

All pipelines > **Terraform-CD**

Pipeline Tasks Variables Retention Options History

Dev Deployment process

Agent job Run on agent +

Install Terraform 1.0.5 Terraform tool installer

Terraform : Init Terraform

Terraform : validate Terraform

Terraform : plan Terraform

Terraform : apply -auto-approve Terraform

Stage name
Dev

The screenshot shows the Azure DevOps Pipelines interface for a pipeline named "Terraform-CD". The left sidebar has a "Pipelines" section selected. The main area shows a single stage named "Dev" with a deployment process. Inside the stage, there is an "Agent job" step labeled "Run on agent" followed by five Terraform tasks: "Install Terraform 1.0.5", "Terraform : Init", "Terraform : validate", "Terraform : plan", and "Terraform : apply -auto-approve". A "Stage name" dropdown is set to "Dev".

Azure IaC DevOps Releases

The screenshot shows the Azure DevOps interface for a pipeline named "Terraform-CD". The left sidebar highlights the "Releases" section. The main area displays the pipeline configuration and a detailed history of releases.

Pipeline Configuration:

- Search bar: Search all pipelines
- Toolbar: Overview, Boards, Repos, Pipelines, Environments, Releases (selected), Library, Task groups, Deployment groups, Test Plans, Artifacts.
- Release name: Terraform-CD
- Status: Prod
- Stages: Dev, QA, Stage, Prod

Release History:

Release	Created	Stages
Release-7	25/08/2021, 15:59:34	Dev, QA, Stage, Prod
Release-6	25/08/2021, 15:32:50	Dev, QA, Stage, Prod
Release-5	25/08/2021, 15:14:29	Dev
Release-4	25/08/2021, 15:01:11	Dev
Release-3	25/08/2021, 14:55:53	Dev
Release-2	25/08/2021, 14:51:55	Dev
Release-1	25/08/2021, 14:43:41	Dev

Azure IaC DevOps Environment TF State Files

Microsoft Azure Search resources, services, and docs (G+)

Home > Resource groups > terraform-storage-rg > terraformstate201 >

 **tfstatefiles** ...

» Upload Change access level Refresh | Delete | Change tier | Acquire lease Brea

Authentication method: Access key ([Switch to Azure AD User Account](#))
Location: tfstatefiles

Search blobs by prefix (case-sensitive)

Add filter

Name	Modified	Access tier
<input type="checkbox"/> dev-terraform.tfstate	8/25/2021, 4:01:20 PM	Hot (Inferred)
<input type="checkbox"/> prod-terraform.tfstate	8/25/2021, 3:50:26 PM	Hot (Inferred)
<input type="checkbox"/> qa-terraform.tfstate	8/25/2021, 4:03:15 PM	Hot (Inferred)
<input type="checkbox"/> stage-terraform.tfstate	8/25/2021, 4:05:37 PM	Hot (Inferred)

^ Azure IaC DevOps for Terraform Projects

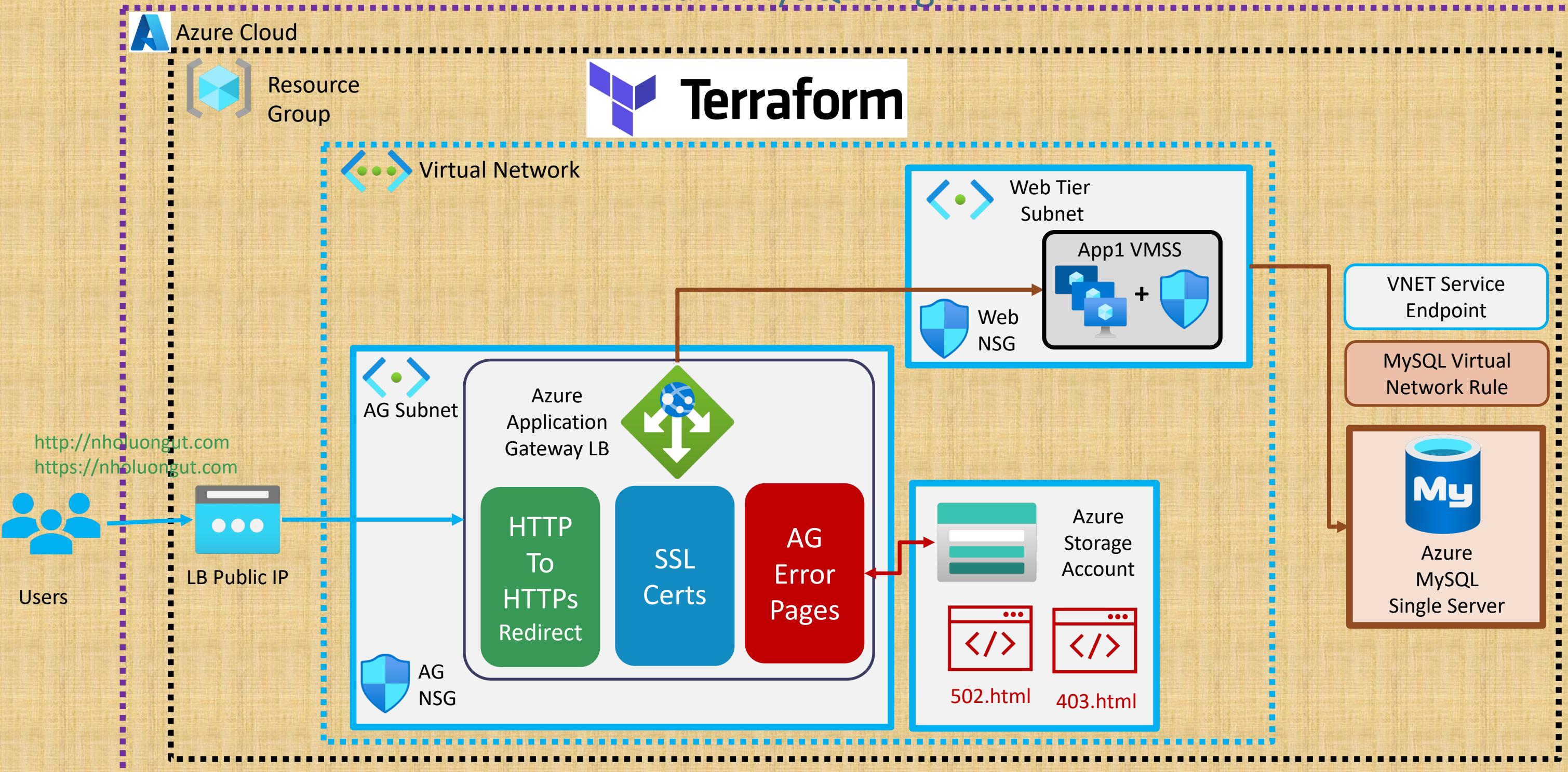
14 lectures • 2hr 37min

- Step-01: Introduction to Azure IaC DevOps
- ▶ Step-02: Review Terraform Configs 16:04
- ▶ Step-03: Create Git Repo and Check-In Terraform Configs 06:16
- ▶ Step-04: GIT SSH Connection Note 01:49
- ▶ Step-05: Terraform Dependency Lock File check-in to Git Repo 19:29
- ▶ Step-06: Create Azure DevOps Organization and Install TF Extension 10:02
- ▶ Step-07: Azure DevOps Build Pipelines Introduction 12:10
- ▶ Step-08: Create Azure Build Pipeline 17:25
- ▶ Step-09: Azure DevOps Release Pipelines Introduction 09:34
- ▶ Step-10: Create Service Connection and Storage Account for TFState Files 08:33
- ▶ Step-11: Create Releaes Pipeline, Artificats, Dev Stage Install and Init Tasks 12:13
- ▶ Step-12: Create Validate, Plan, Apply Tasks for Dev Stage 05:49
- ▶ Step-13: Run end to end pipelines and verify dev resources 08:05
- ▶ Step-14: Create QA, Stage and Prod stages and Verify Resources 17:54
- ▶ Step-15: Make changes to prod.tfvars, verify and clean-up 11:23

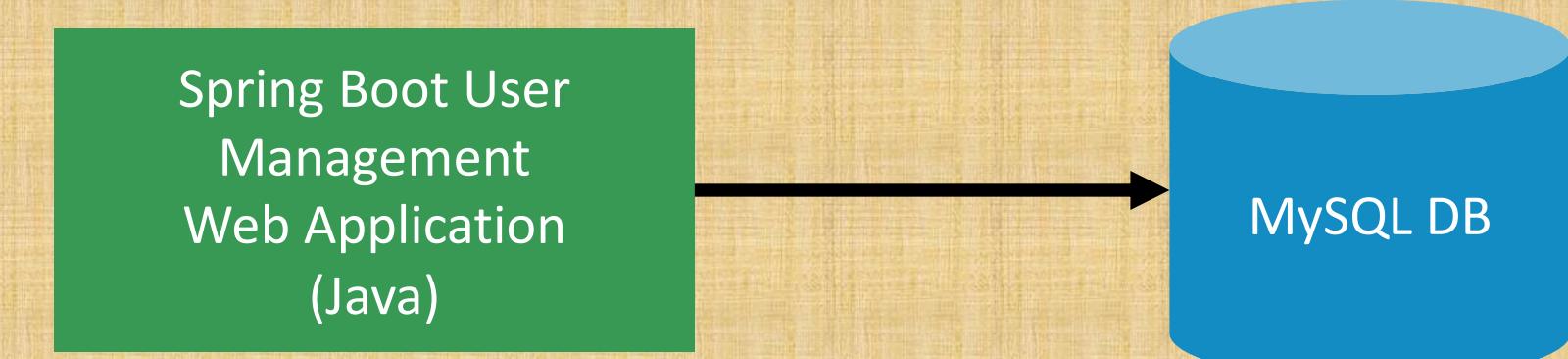
Time it takes to complete this Demo

Real-World
Demo

Azure MySQL Single Server



User Management Web Application



UMS Web App with Create User, List User, Login and Logout Features

UMS Web App Listens on Port 8080

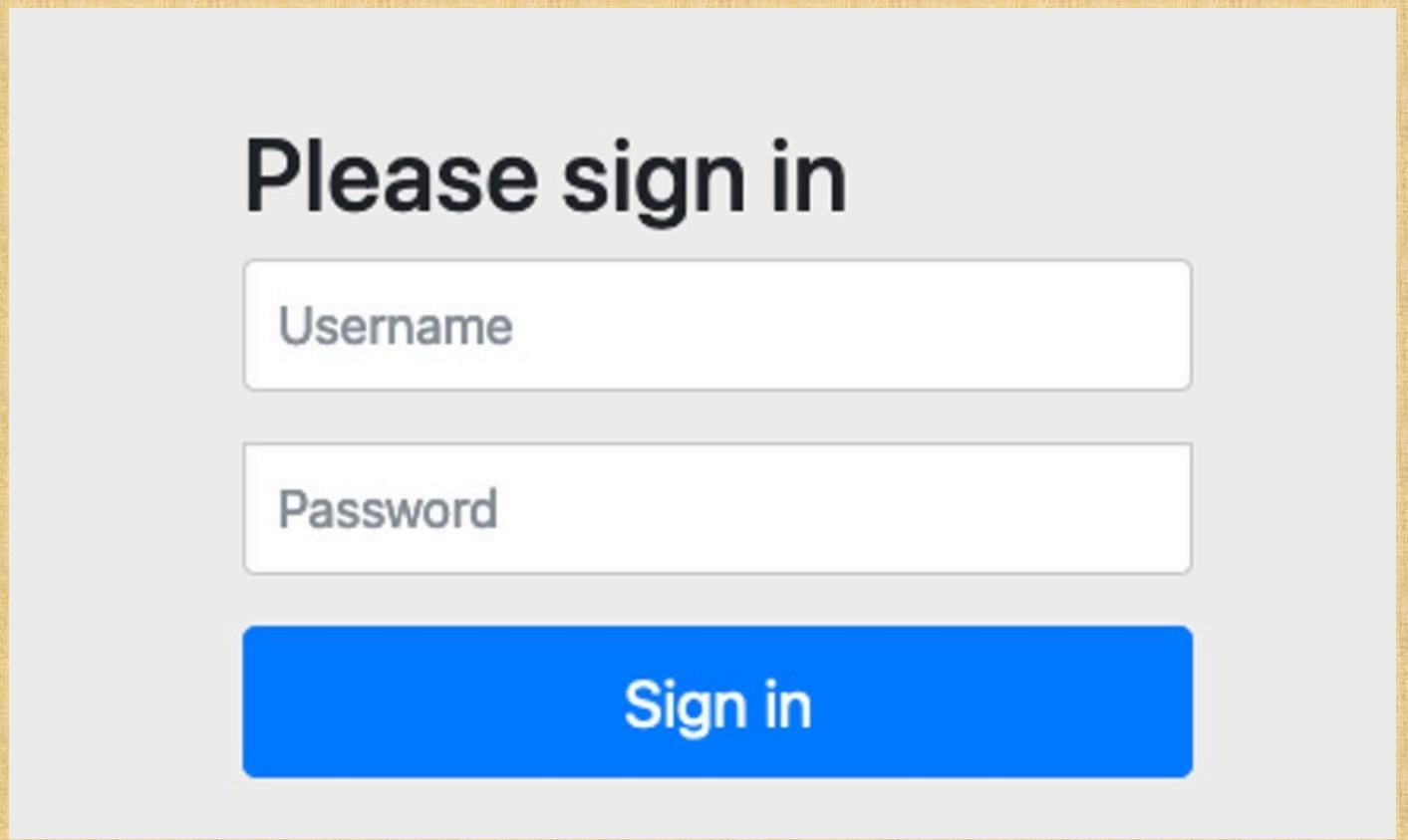
UMS Web App needs MySQL DB to store its users. If connection to DB fails, it cannot start

UMS Web App DB information can be passed via Environment Variables (DB Name, Port User, Pass)

New users created will be stored in MySQL DB

We can login with new users created to UMS Web App

UMS – Login Screen



UMS – Landing Page Post Login

Kubernetes on Cloud

Masterclass Series on AWS, Azure and GKE

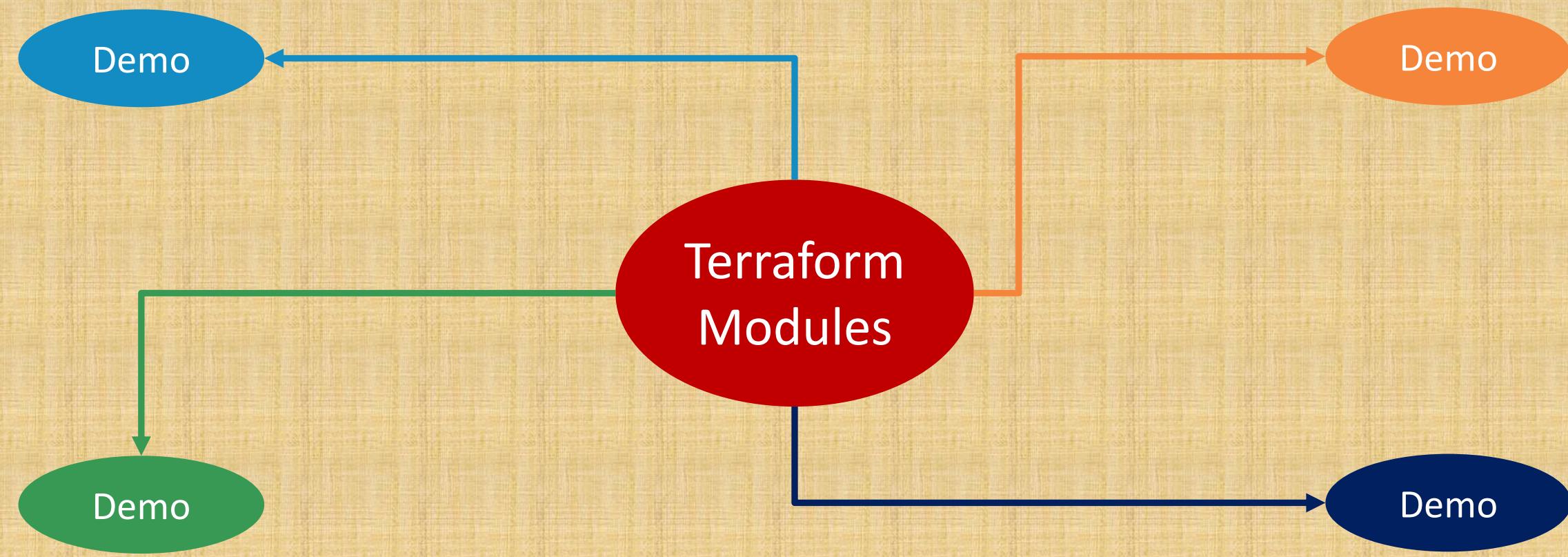
User Management

List Users

Create Users

Use a Public Registry module in our TF Configs

Build a Local Terraform Module and call it from Root Module and Test



1. Build a Static Website using Azure manually using Azure Portal.
2. Automate it using Terraform Resources

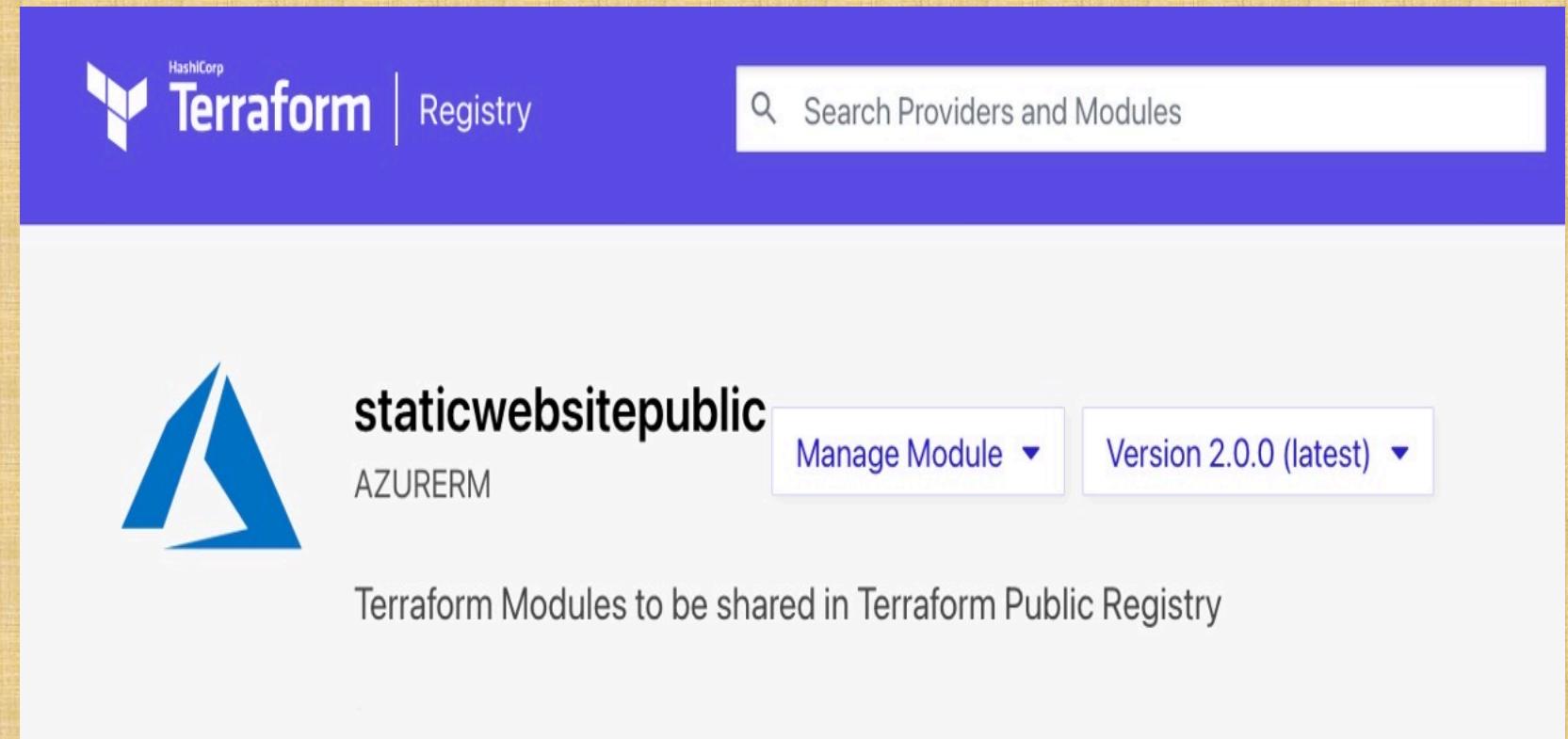
Publish to Public Terraform Registry and Access it from Public Registry and Test.

Publish to Public Terraform Registry

Demo

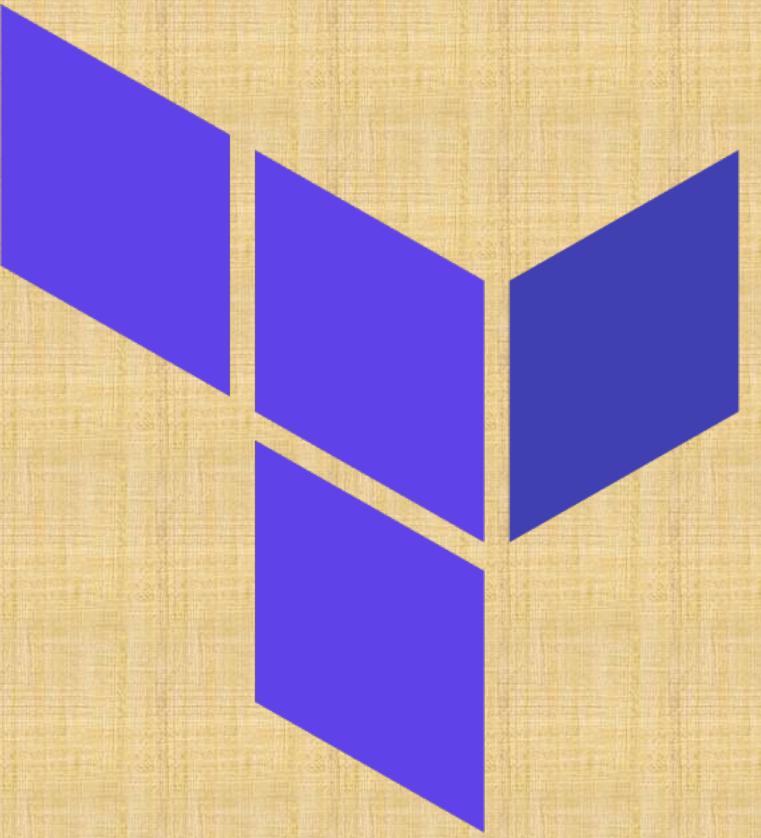
Publish to Public Terraform Registry and Access it from Public Registry and Test.

```
✓ 52-Terraform-Module-Publish-to-Public-Registry
  > static-content
  ✓ terraform-azure-static-website-module-manifests
    LICENSE
    main.tf
    outputs.tf
    README.md
    variables.tf
    versions.tf
  > terraform-manifests
    README.md
```





Terraform Fundamentals Pre-requisite Note



Terraform Fundamentals

✓ Terraform - Install Tools	5 lectures • 31min
✓ Terraform Command Basics	3 lectures • 25min
✓ Terraform Language Basics	3 lectures • 21min
✓ Terraform Settings & Providers Block	5 lectures • 44min
✓ Terraform Multiple Providers	2 lectures • 12min
✓ Terraform Dependency Lock File	4 lectures • 30min
✓ Terraform Resource Syntax, Behaviour and State	8 lectures • 49min

We are going to learn
Terraform Fundamentals

This section is common for
both Azure Terraform

Azure –Terraform Associate Demos

Continue with remaining
Demos

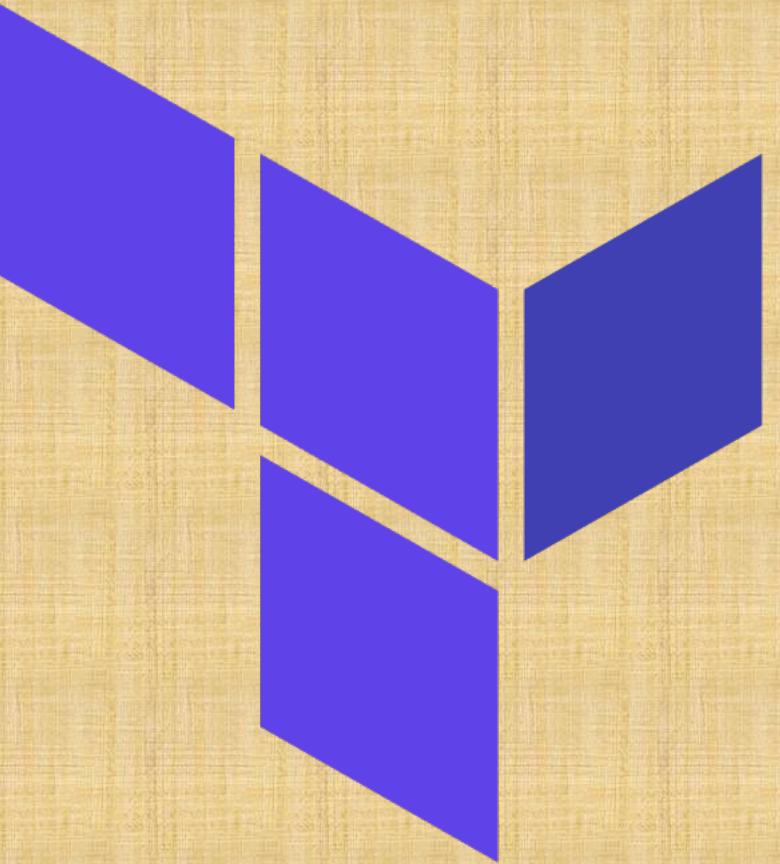
Terraform on Azure with IaC DevOps SRE | Real-World Demos

Continue with Real-World
Demos

Terraform

Infrastructure as Code

IaC



What is Infrastructure as Code ?

Traditional Way of Managing Infrastructure



This is fullflow infrastructure management process

Traditional Way of Managing Infrastructure

Admin-1



Admin-1



Admin-2



Admin-2



Admin-2



No CI

Delays

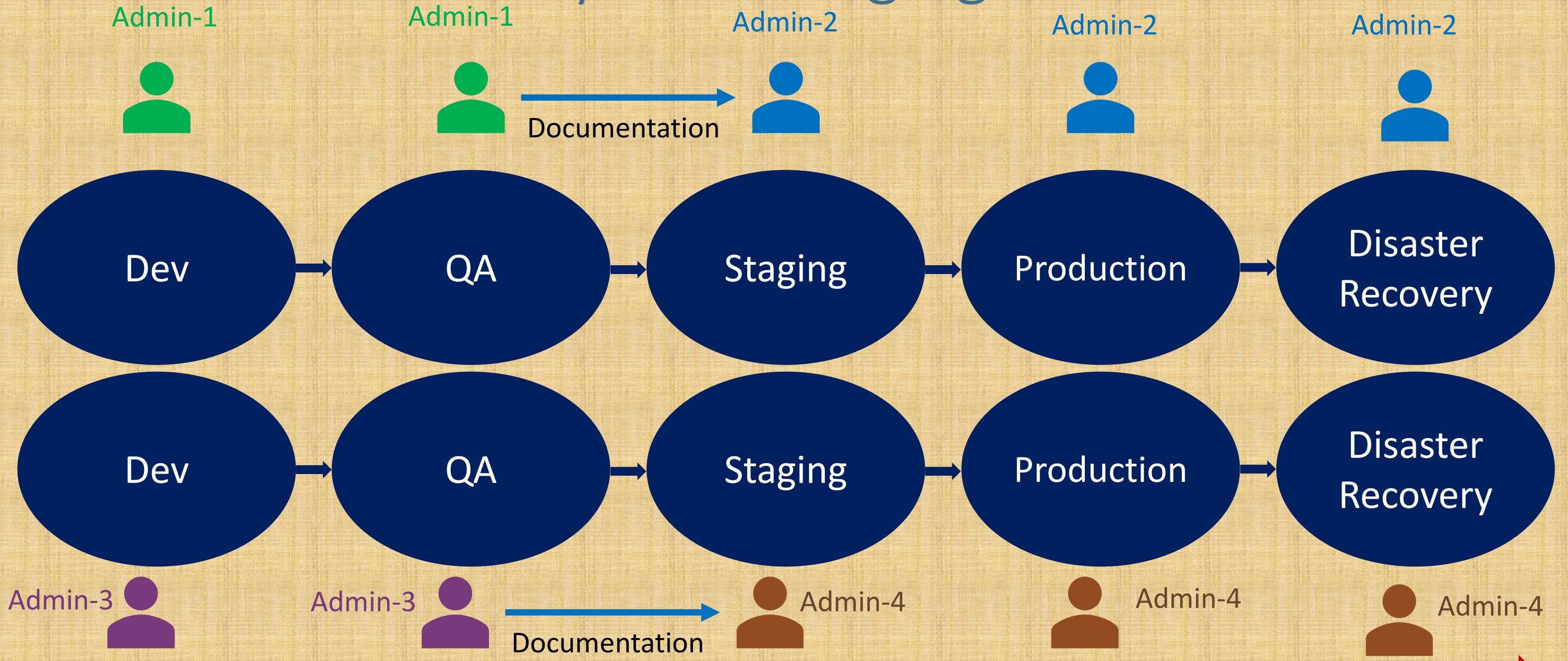
Issues

Outages

Not-in-Sync

Many Problems at many places in manual process

Traditional Way of Managing Infrastructure



Infrastructure scalability – Workforce need to be increased to meet the timelines

Traditional Way of Managing Infrastructure

Prod-1

Prod-2

Prod-3

Prod-4

Scale Up

Prod-1

Prod-2

Scale Down

On-Demand Scale-Up and Scale-Down is not an option

5 Days

Admin-1



Check-In TF Code

Github



Triggers
TF Runs



Terraform
Cloud

DevOps / CI CD for IaC

Scale-Up and Scale-Down On-Demand

Creates Infra

Dev

QA

Staging

Production

Disaster
Recovery

One-Time
Work

Re-Use
Template
s

Quick &
Fast

Reliable

Tracked
for Audit

Total Time: 25 Days reduced to 5 days, Provisioning environments will be in minutes or seconds

Manage using IaC with Terraform

Visibility

IaC serves as a very **clear reference** of what resources we created, and what their settings are. We don't have to **navigate** to the web console to check the parameters.

Stability

If you **accidentally** change the **wrong** setting or delete the **wrong** resource in the web console you can **break things**. IaC helps **solve this**, especially when it is combined with **version control**, such as Git.

Scalability

With IaC we can **write it once** and then **reuse it many times**. This means that one well written template can be used as the **basis for multiple services**, in multiple regions around the world, making it much easier to horizontally scale.

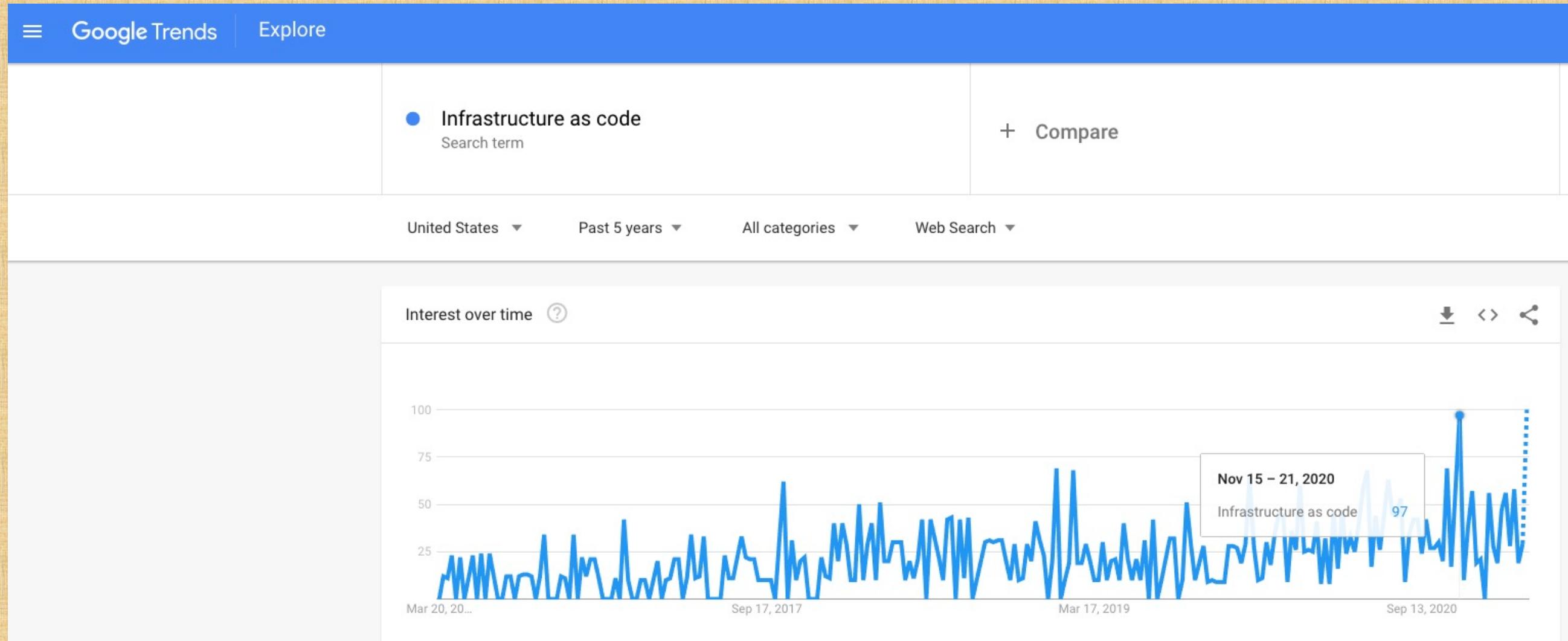
Security

Once again IaC gives you a **unified template** for how to deploy our architecture. If we create one **well secured architecture** we can reuse it multiple times, and know that each deployed version is following the same settings.

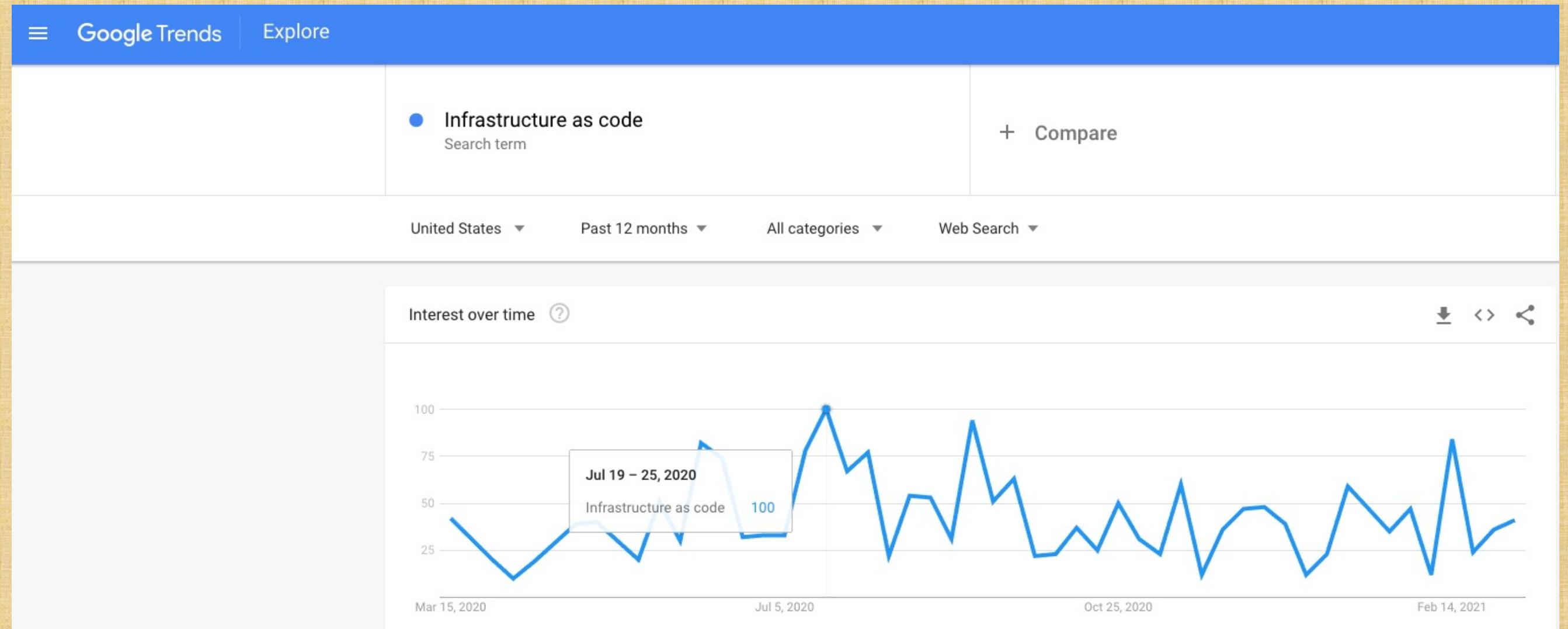
Audit

Terraform not only creates resources it also **maintains the record** of what is created in real world cloud environments using its State files.

Google Trends – Past 5 Years

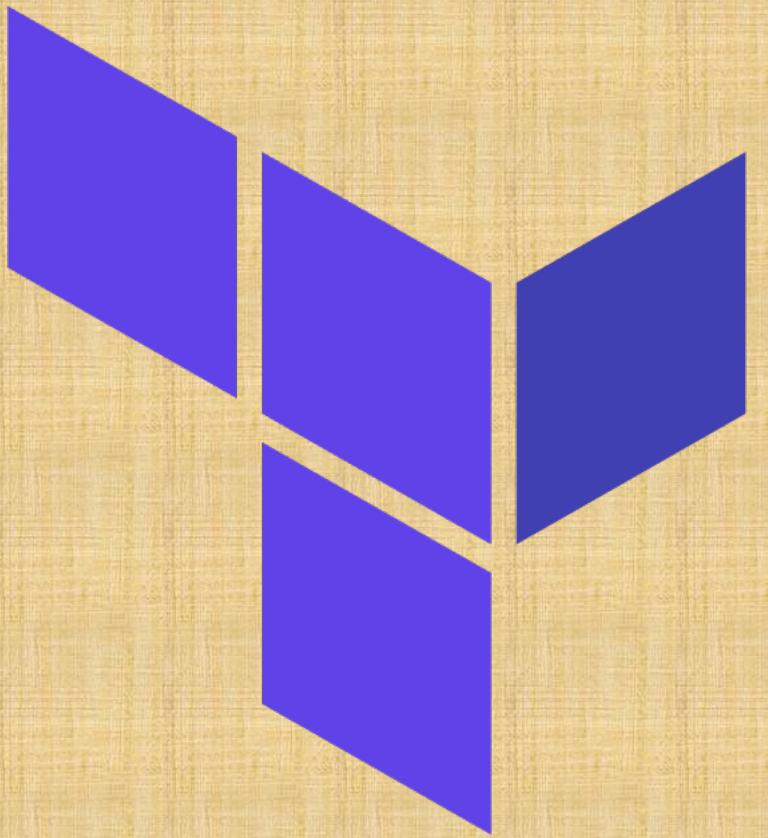


Google Trends – Past 1 Year





Terraform Installation



Terraform Installation

Terraform CLI

Azure CLI

VS Code Editor

Terraform plugin
for VS Code

GIT Client

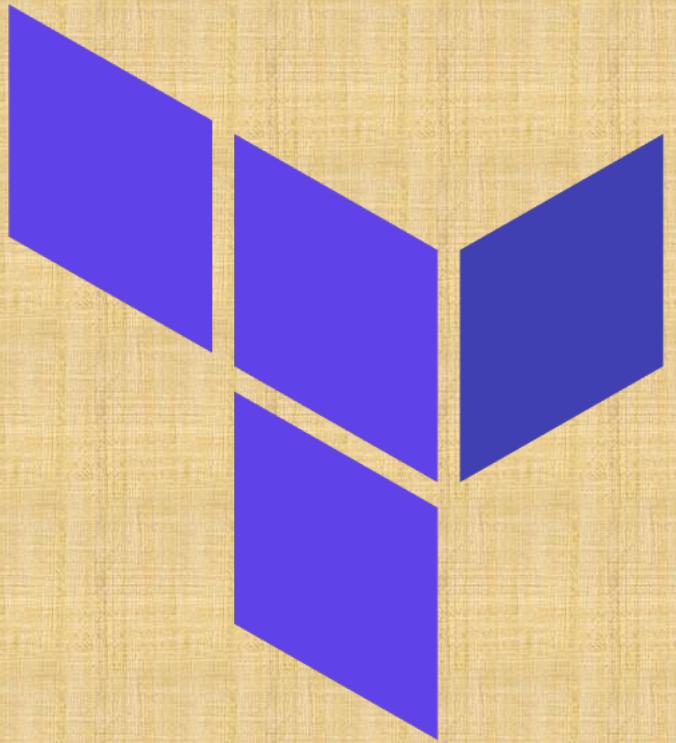
Mac OS

Windows OS

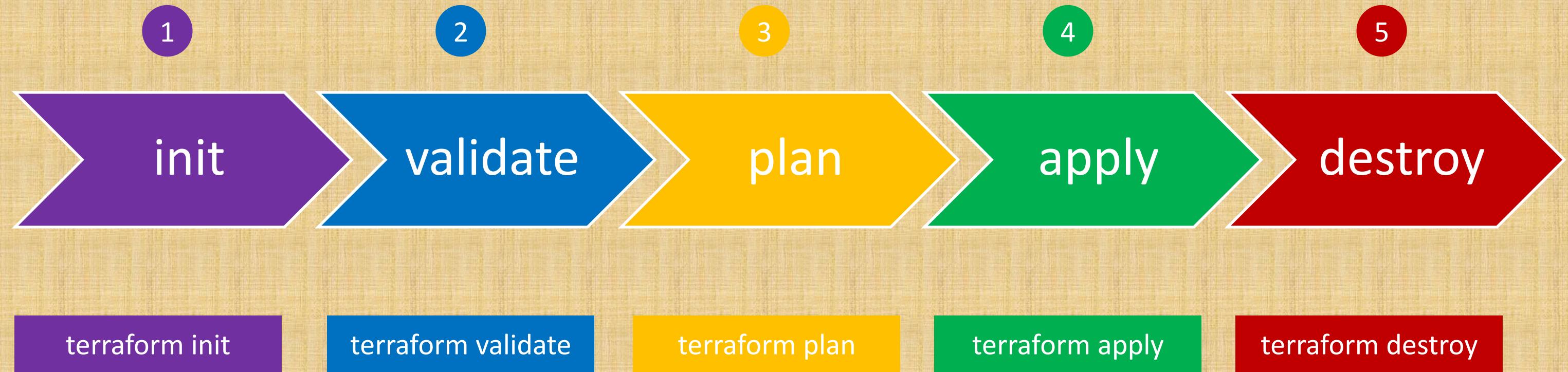
Linux OS



Terraform Command Basics



Terraform Workflow



Terraform Workflow

1

init

2

validate

3

plan

4

apply

5

destroy

- Used to Initialize a working directory containing terraform config files
- This is the first command that should be run after writing a new Terraform configuration
- Downloads Providers

- Validates the terraform configurations files in that respective directory to ensure they are syntactically valid and internally consistent.

- Creates an execution plan
- Terraform performs a refresh and determines what actions are necessary to achieve the desired state specified in configuration files

- Used to apply the changes required to reach the desired state of the configuration.
- By default, apply scans the current directory for the configuration and applies the changes appropriately.

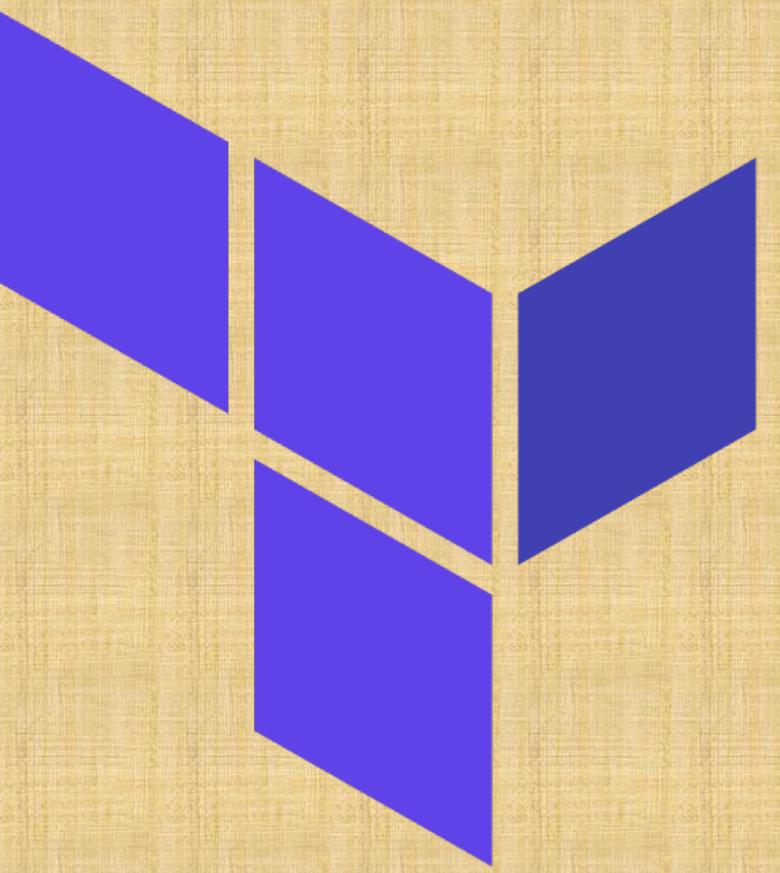
- Used to destroy the Terraform-managed infrastructure
- This will ask for confirmation before destroying.



Terraform

Language Basics

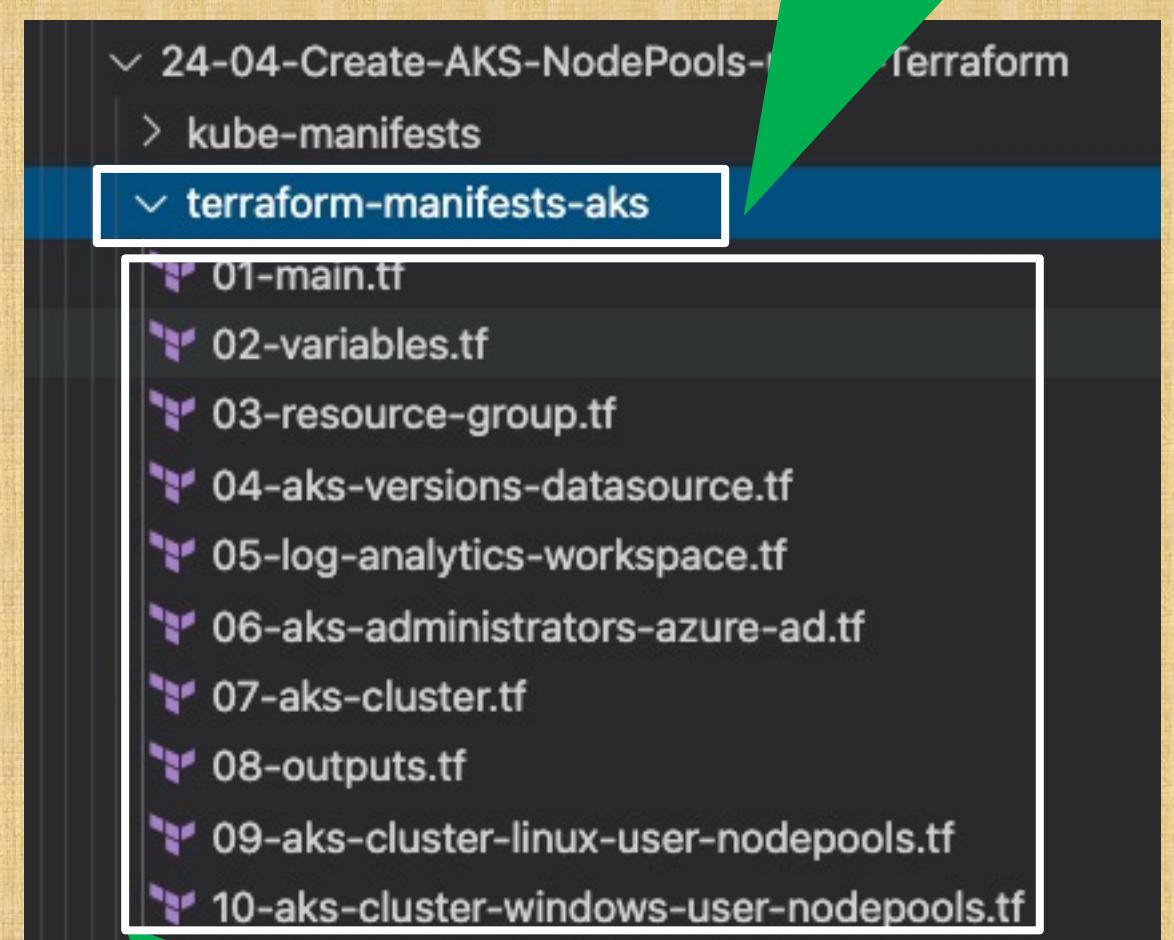
Theoretical but Very Important



Terraform Language Basics – Files

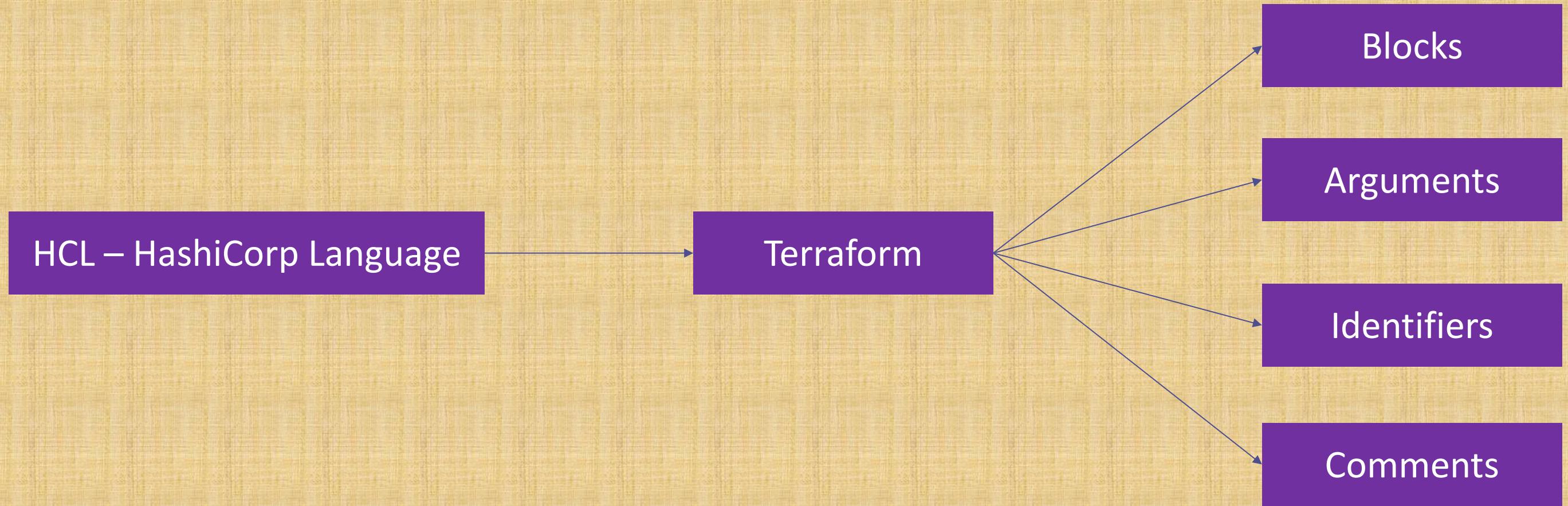
- Code in the Terraform language is stored in plain text files with the `.tf` file extension.
- There is also a JSON-based variant of the language that is named with the `.tf.json` file extension.
- We can call the files containing terraform code as Terraform Configuration Files or Terraform Manifests

Terraform Working Directory



Terraform Configuration Files ending with `.tf` as extension

Terraform Language Basics – Configuration Syntax



Terraform Language Basics – Configuration Syntax

```
# Template
<BLOCK TYPE> "<BLOCK LABEL>" "<BLOCK LABEL>"  {
    # Block body
    <IDENTIFIER> = <EXPRESSION> # Argument
}

# Azure Example
# Create a resource group
resource 'azurerm_resource_group' "myrg" { # Resource BLOCK
    name = "myrg-1" # Argument
    location = "East US" # Argument
}
```

Block Type

Top Level &
Block inside
Blocks

Top Level Blocks: resource, provider

Block Inside Block: provisioners,
resource specific blocks like tags

Block Labels

Based on Block
Type block labels
will be 1 or 2

Example:
Resource – 2
labels

Variables – 1 label

Arguments

Terraform Language Basics – Configuration Syntax

Argument
Name
[or]
Identifier

Argument
Value
[or]
Expression

```
# Template
<BLOCK TYPE> "<BLOCK LABEL>" "<BLOCK LABEL>"  {
    # Block body
    <IDENTIFIER> = <EXPRESSION> # Argument
}

# Azure Example
# Create a resource group
resource "azurerm_resource_group" "myrg" { # Resource BLOCK
    name = "myrg-1" # Argument
    location = "East US" # Argument
}
```

Terraform Language Basics – Configuration Syntax

Single Line Comments with # or //

Multi-line
comment

```
# Create a resource group
resource "azurerm_resource_group" "myrg" {
    name = "myrg-1"
    location = "eastus"
}

/*
Multi-line Comments
Line-1
Line-2
*/
// Azure Resource Group
```

Terraform language uses a **limited** number of **top-level block** types, which are **blocks** that can appear **outside** of any other **block** in a TF configuration file.

Terraform Top-Level Blocks

Most of **Terraform's features** are implemented as **top-level** blocks.

Terraform Block

Providers Block

Resources Block

Fundamental Blocks

Input Variables Block

Output Values Block

Local Values Block

Variable Blocks

Data Sources Block

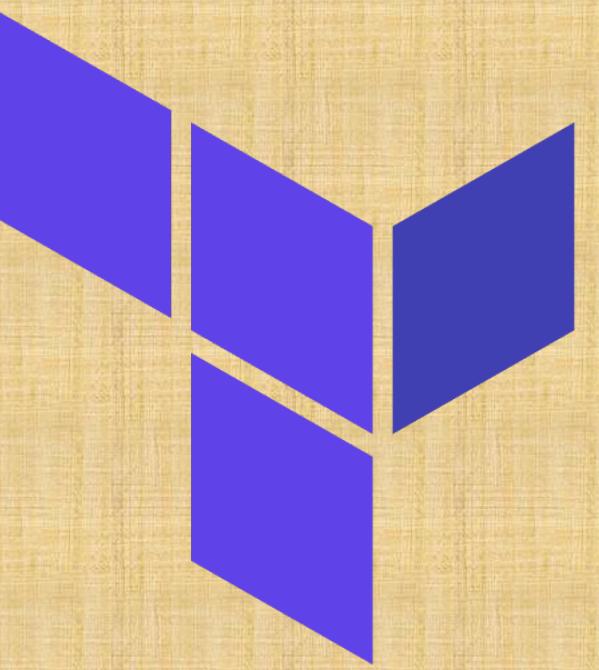
Modules Block

Calling / Referencing Blocks



Terraform Fundamental Blocks

Terraform, Provider, Resources



Terraform Basic Blocks

Terraform Block

Special block used to configure some **behaviors**

Specifying a **required Terraform CLI Version**

Specifying **Provider Requirements & Versions**

Configuring a Terraform Backend (**Terraform State**)

Provider Block

HEART of Terraform

Terraform relies on providers to **interact** with Remote Systems

Declare providers for Terraform to **install** providers & use them

Provider configurations belong to **Root Module**

Resource Block

Each Resource Block describes one or more Infrastructure Objects

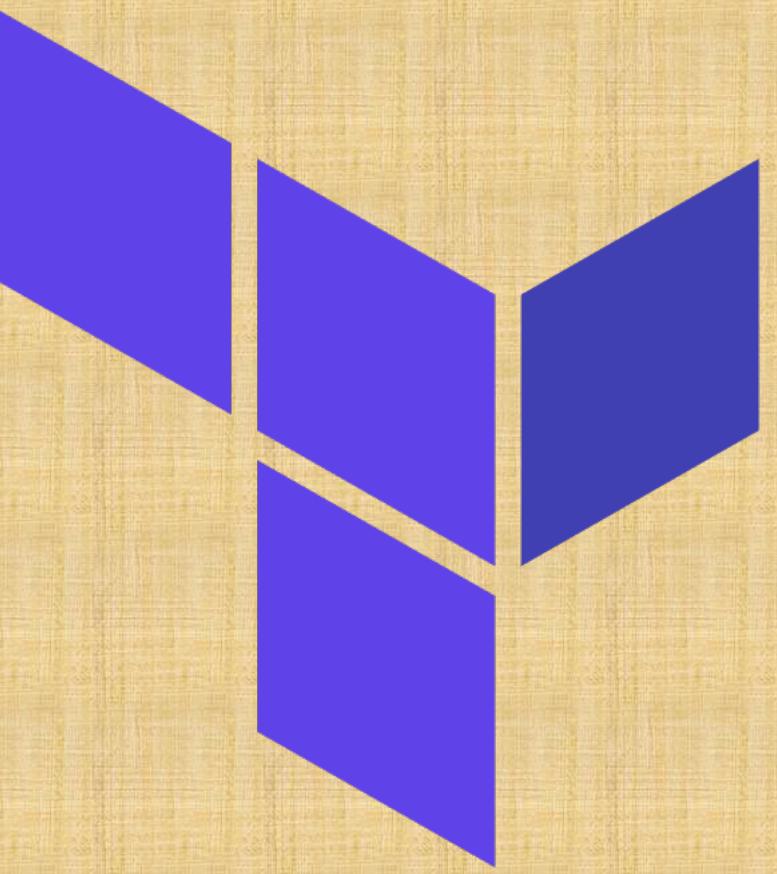
Resource Syntax: How to declare Resources?

Resource Behavior: How Terraform handles resource declarations?

Provisioners: We can configure Resource post-creation actions



Terraform Block



Terraform Block

- This block can be called in 3 ways. All means the same.
 - Terraform Block
 - Terraform Settings Block
 - Terraform Configuration Block
- Each terraform block can contain a number of settings related to Terraform's behavior.
- Within a terraform block, **only constant values can be used**; arguments **may not refer** to named objects such as resources, input variables, etc, and **may not use any** of the Terraform language built-in functions.

Terraform Block from 0.13 onwards

Terraform 0.12 and earlier:

```
# Configure the AWS Provider
provider "aws" {
    version = "~> 3.0"
    region  = "us-east-1"
}

# Create a VPC
resource "aws_vpc" "example" {
    cidr_block = "10.0.0.0/16"
}
```

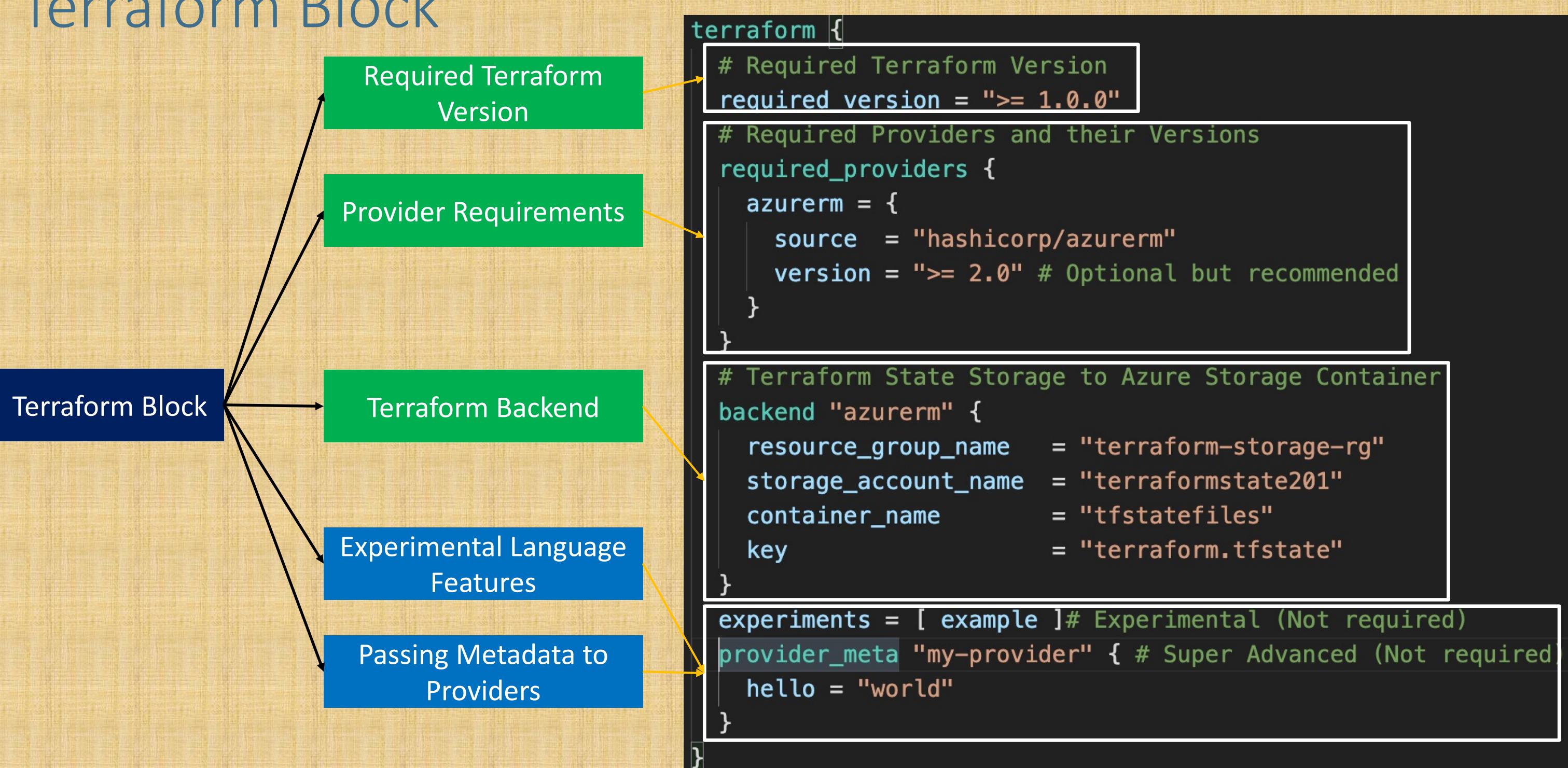
Terraform 0.13 and later:

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 3.0"
    }
  }
}

# Configure the AWS Provider
provider "aws" {
    region = "us-east-1"
}

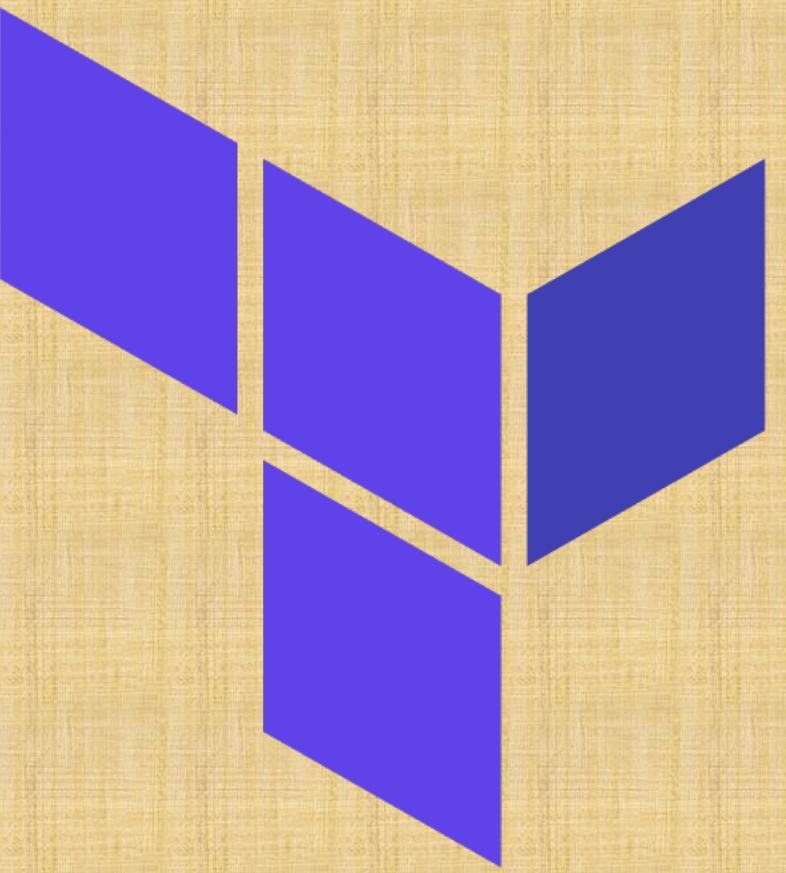
# Create a VPC
resource "aws_vpc" "example" {
    cidr_block = "10.0.0.0/16"
}
```

Terraform Block





Terraform Providers



Terraform Providers

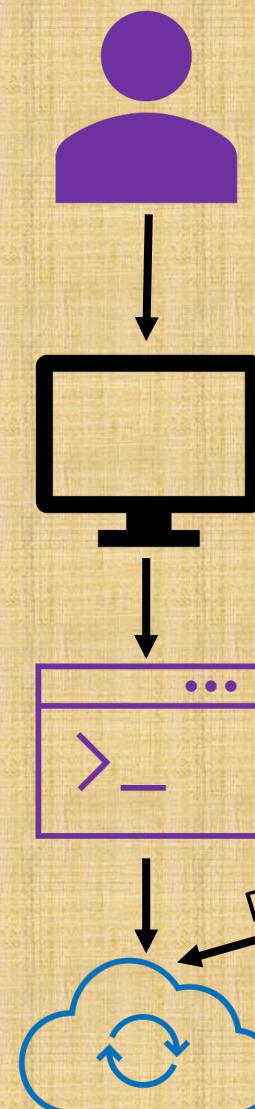
Terraform Admin

Local Desktop

Terraform CLI

Terraform Azure Provider

2 terraform validate



Providers are **HEART** of Terraform

Every **Resource Type** (example: Azure Resource Group), is implemented by a Provider

Without Providers Terraform **cannot** manage any infrastructure.

Providers are distributed separately from Terraform and each provider has its own **release cycles** and **Version Numbers**

Terraform **Registry** is publicly available which contains many Terraform Providers for most **major** Infra Platforms

Azure Cloud

Azure APIs

Resource Group

Provider Requirements

```
# Terraform Block
terraform {
  required_version = ">= 1.0.0"
  required_providers {
    azurerm = {
      source = "hashicorp/azurerm"
      version = ">= 2.0"
    }
  }
}
```

Terraform Providers

Provider Configuration

```
# Provider Block
provider "azurerm" {
  features {}
}
```

Dependency Lock File

```
└─ terraform-manifests
    └─ .terraform
        └─ .terraform.lock.hcl
    └─ c1-versions.tf
    └─ c2-resource-group.tf
```

```
# This file is maintained automatically by "terraform init".
# Manual edits may be lost in future updates.

provider "registry.terraform.io/hashicorp/azurerm" {
  version      = "2.64.0"
  constraints = ">= 2.0.0"
  hashes = [
    "h1:6qwrh6hD19XCreyps0ojY1Vp9VI5vr1bCbl1EZGy6Io=",
    "zh:048da64c3d173f3467e908ca8b28962bbf6e3e06597614474b46d4c09ec8ca6c",
    "zh:08d5baa31e498b2c7761d88a8ff5875066566f9b3644a5c12c8ea305e1a7c85d",
    "zh:0a452f95795f56c16f5b0febe05539f44638895f387973f594ac3de179f22150",
    "zh:1b6dd54a023ef22c9fadef2c6b6e8e66e2c9a29d23921706e10b35b1bd2a47ed3",
    "zh:3260fdc14d2a33c0c0e7b230687d303e12852c54aa1120918e7b77f954ed1b",
    "zh:b36dd823f543fb45b31a623ff68be5fd49fdcc50c2e032dd44828a26f9d41b9",
    "zh:ba6514590b1be102438cc3632795965f3e271044635bf05ec0f0e46c4795b06c",
    "zh:bf663c286ba4198111d8b9f4987a277f1a378d0ef40a824c8fc25f6e4de1866d",
    "zh:d8c122295c29c9078804cdbcab53e1fc5c75071d6600b5389372c8f8bda1967",
    "zh:dffbb1af6ab61b875cbd4fd198d3b12c28261a216c1a32f393b8c795b62bb30",
    "zh:f8dca9bf566e7869412a5c10e44f64f3a19eee7c4aa18e7ce5db9a6f145a2a4f",
  ]
}
```

Dependency Lock File

```
# This file is maintained automatically by "terraform init".  
# Manual edits may be lost in future updates.  
  
provider "registry.terraform.io/hashicorp/azurerm" {  
    version      = "2.64.0"  
    constraints = ">= 2.0.0"
```

Required Providers

```
# Terraform Block
terraform {
  required_version = ">= 1.0.0"
  required_providers {
    azurerm = {
      source = "hashicorp/azurerm"
      version = ">= 2.0"
    }
  }
}

# Provider Block
provider "azurerm" {
  features {}
}
```

Local Names

Local Names are **Module specific** and should be **unique per-module**

Terraform configurations always refer to **local name** of provider **outside** required_provider block

Users of a provider can choose **any local name** for it (myazure, azure1, azure2).

Recommended way of choosing local name is to use preferred local name of that provider (For Azure Provider: hashicorp/azurerm, **preferred local name** is azurerm)

Source

It is the **primary location** where we can download the Terraform Provider

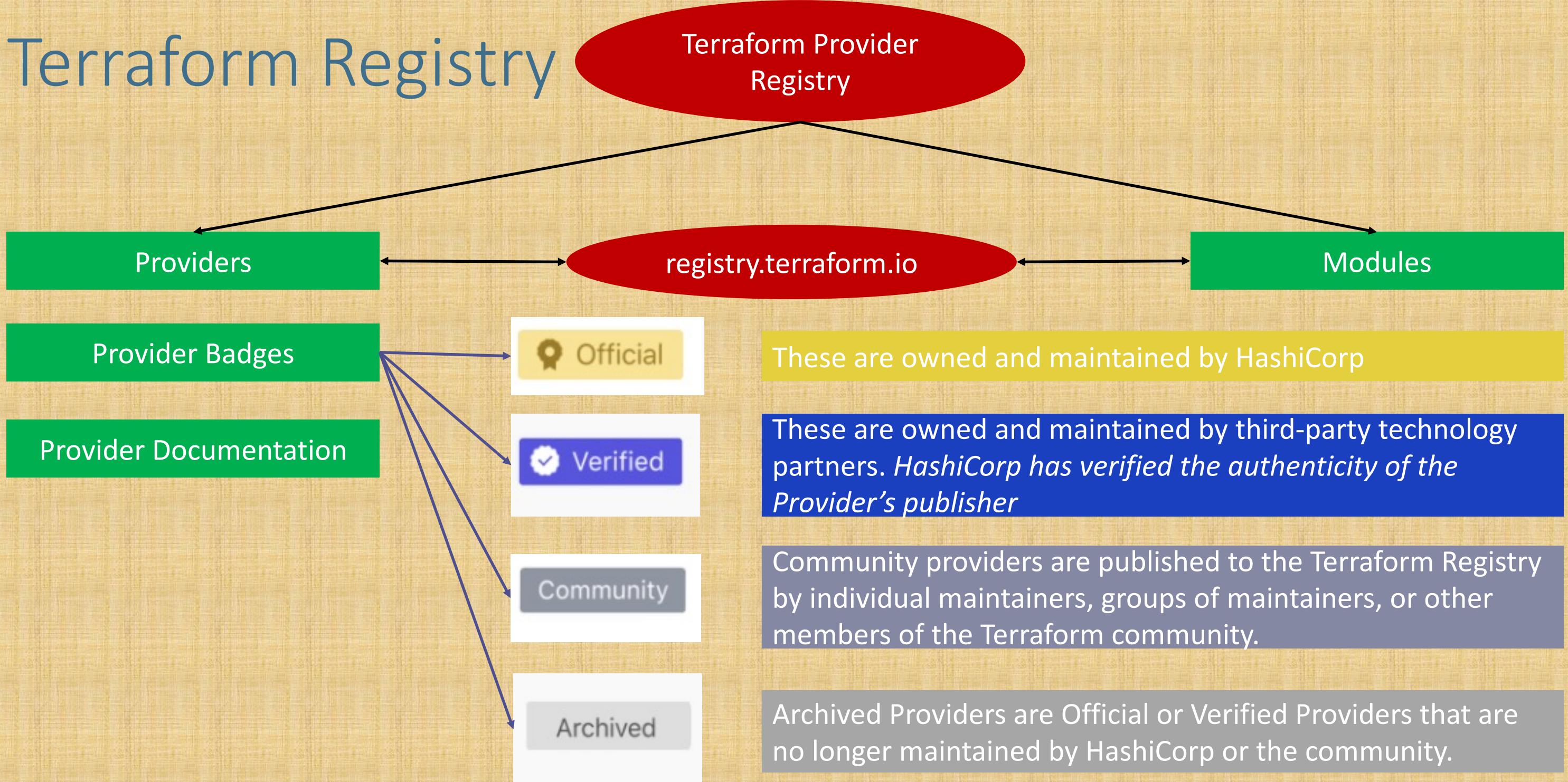
Source addresses consist of **three parts** delimited by **slashes (/)**

[<HOSTNAME>/]<NAMESPACE>/<TYPE>

registry.terraform.io/hashicorp/azurerm

Registry Name is **optional** as default is going to be Terraform Public Registry

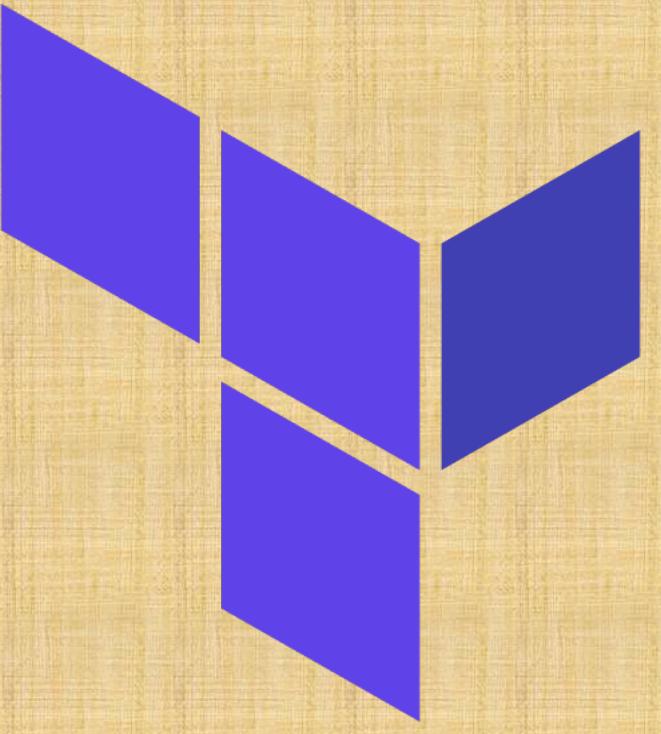
Terraform Registry





Terraform

Multiple Providers



Multiple Providers

We can define **multiple configurations** for the same **provider**, and select which one to use on a **per-resource** or **per-module** basis.

The primary reason for this is to support **multiple regions** for a cloud platform

```
# Provider-1 for EastUS
# Default Provider
provider "azurerm" {
    features {}
}

# Provider-2 for WestUS
provider "azurerm" {
    features {}
    alias = "provider2-westus"
}
```

Multiple Providers

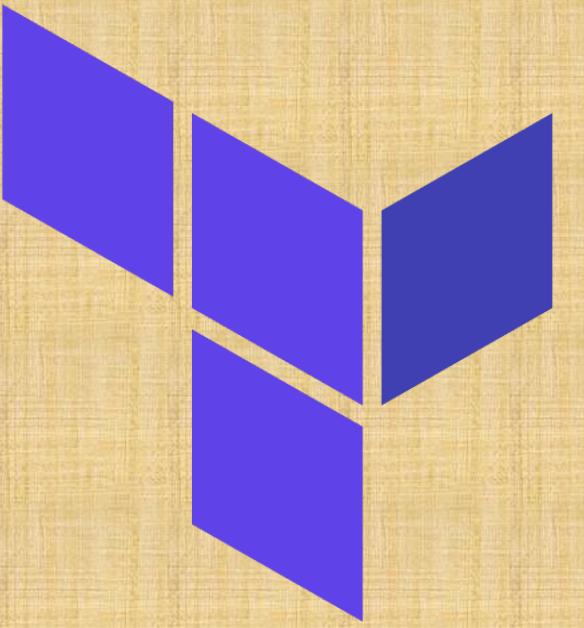
We can use the alternate provider in a resource, data or module by referencing it as <PROVIDER NAME>.<ALIAS>

```
# Uses Default Provider
resource "azurerm_resource_group" "myrg1" {
    name = "myrg-1"
    location = "East US"
}

# Uses "provider2-westus" provider
resource "azurerm_resource_group" "myrg2" {
    name = "myrg-2"
    location = "West US"
    provider = azurerm.provider2-westus
}
```



Terraform Dependency Lock File



Dependency Lock File

Terraform

New feature added
from Terraform v0.14
& later

Providers

Terraform configuration refers to two different kinds of external dependency that come from outside of its own codebase

Modules

Version Constraints within the configuration itself determine which versions of dependencies are *potentially compatible*

Dependency Lock File: After selecting a **specific version** of each dependency using **Version Constraints** Terraform remembers the **decisions it made** in a **dependency lock file** so that it can (by default) make the same decisions again in future.

Location of Lock File: Current Working Directory

Very Important: Lock File currently **tracks only Provider Dependencies**. For modules continue to use **exact version constraint** to ensure that Terraform will always select the same module version.

Checksum Verification: Terraform will also verify that each package it installs matches at least one of the checksums it previously recorded in the lock file, if any, returning an error if none of the checksums match

Dependency Lock File

```
# This file is maintained automatically by "terraform init".
# Manual edits may be lost in future updates.

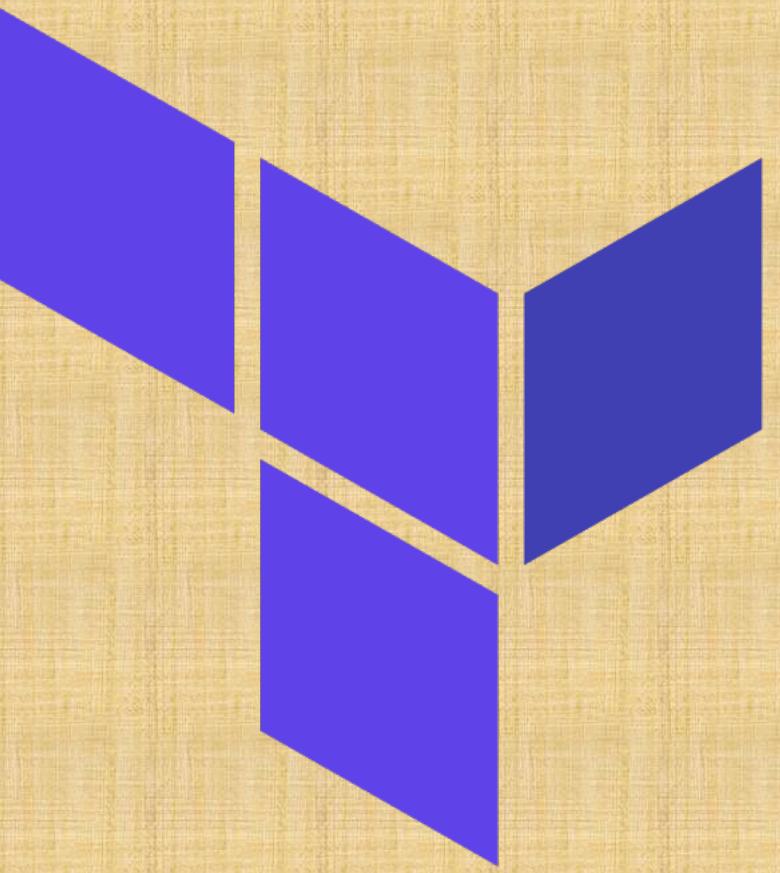
provider "registry.terraform.io/hashicorp/azurerm" {
    version      = "2.64.0"
    constraints = ">= 2.0.0"
```

Importance of Dependency Lock File

If Terraform did not find a lock file, it would download the latest versions of the providers that fulfill the version constraints you defined in the required_providers block inside Terraform Settings Block.

If we have lock file, the lock file causes Terraform to always install the same provider version, ensuring that runs across your team or remote sessions will be consistent.

Terraform Resources Introduction



Terraform Resources

Terraform
Language Basics

Terraform
Resource Syntax

Terraform
Resource Behavior

Terraform
State

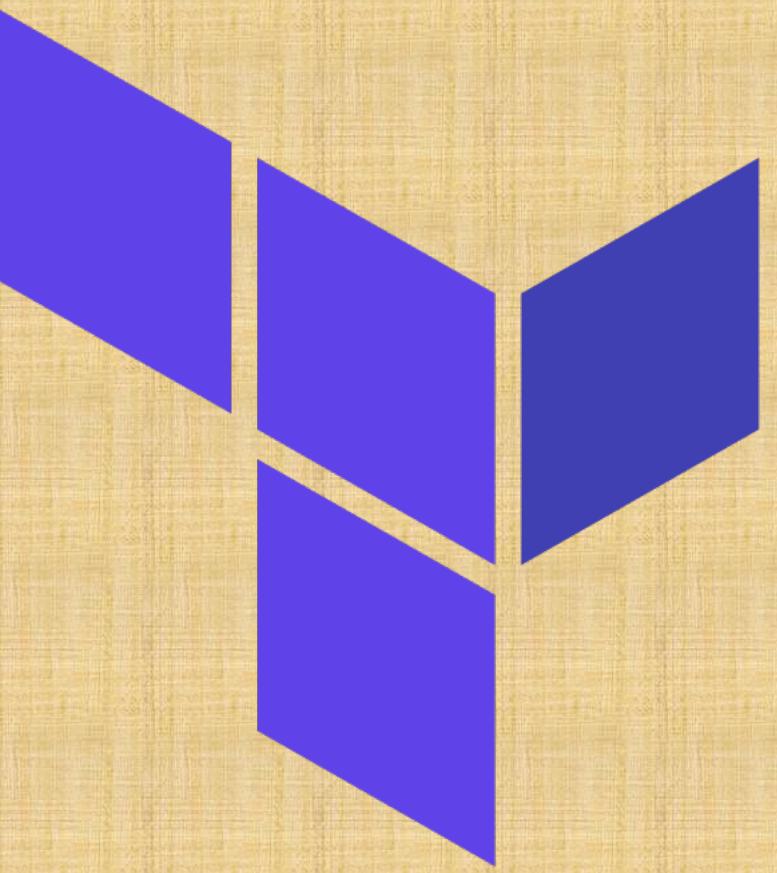
Resource
Meta-Argument
count

Resource
Meta-Argument
depends_on

Resource
Meta-Argument
for_each

Resource
Meta-Argument
lifecycle

Terraform Resource Syntax



Terraform Language Basics – Configuration Syntax

```
# Template
<BLOCK TYPE> "<BLOCK LABEL>" "<BLOCK LABEL>"  {
    # Block body
    <IDENTIFIER> = <EXPRESSION> # Argument
}

# Azure Example
# Create a resource group
resource 'azurerm_resource_group' "myrg" { # Resource BLOCK
    name = "myrg-1" # Argument
    location = "East US" # Argument
}
```

Block Type

Top Level &
Block inside
Blocks

Top Level Blocks: resource, provider

Block Inside Block: provisioners,
resource specific blocks like tags

Block Labels

Based on Block
Type block labels
will be 1 or 2

Example:
Resource – 2
labels

Variables – 1 label

Arguments

Terraform Language Basics – Configuration Syntax

Argument
Name
[or]
Identifier

Argument
Value
[or]
Expression

```
# Template
<BLOCK TYPE> "<BLOCK LABEL>" "<BLOCK LABEL>"  {
    # Block body
    <IDENTIFIER> = <EXPRESSION> # Argument
}

# Azure Example
# Create a resource group
resource "azurerm_resource_group" "myrg" { # Resource BLOCK
    name = "myrg-1" # Argument
    location = "East US" # Argument
}
```

Resource Syntax

Resource Type: It determines the kind of **infrastructure object** it manages and what arguments and other attributes the resource supports.

Resource Local Name: It is used to refer to this resource from elsewhere in the same Terraform module, but has **no significance** outside that module's scope.
The resource type and name together serve as an identifier for a given resource and so must be **unique** within a module

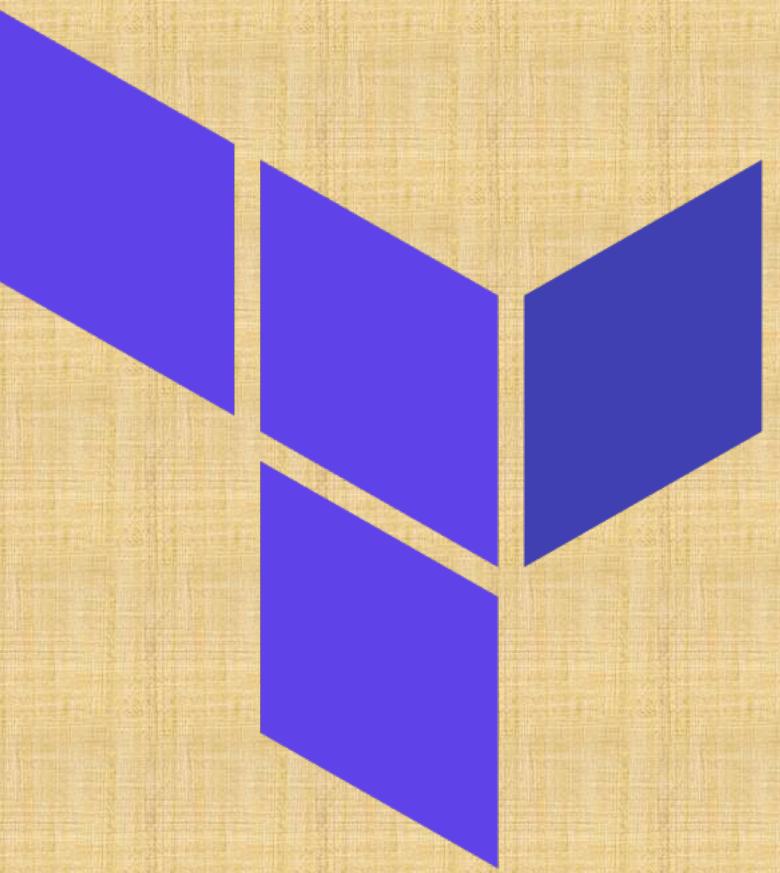
Meta-Arguments: Can be used with any resource to change the behavior of resources

Resource Arguments: Will be specific to resource type. Argument Values can make use of **Expressions** or other Terraform **Dynamic Language Features**

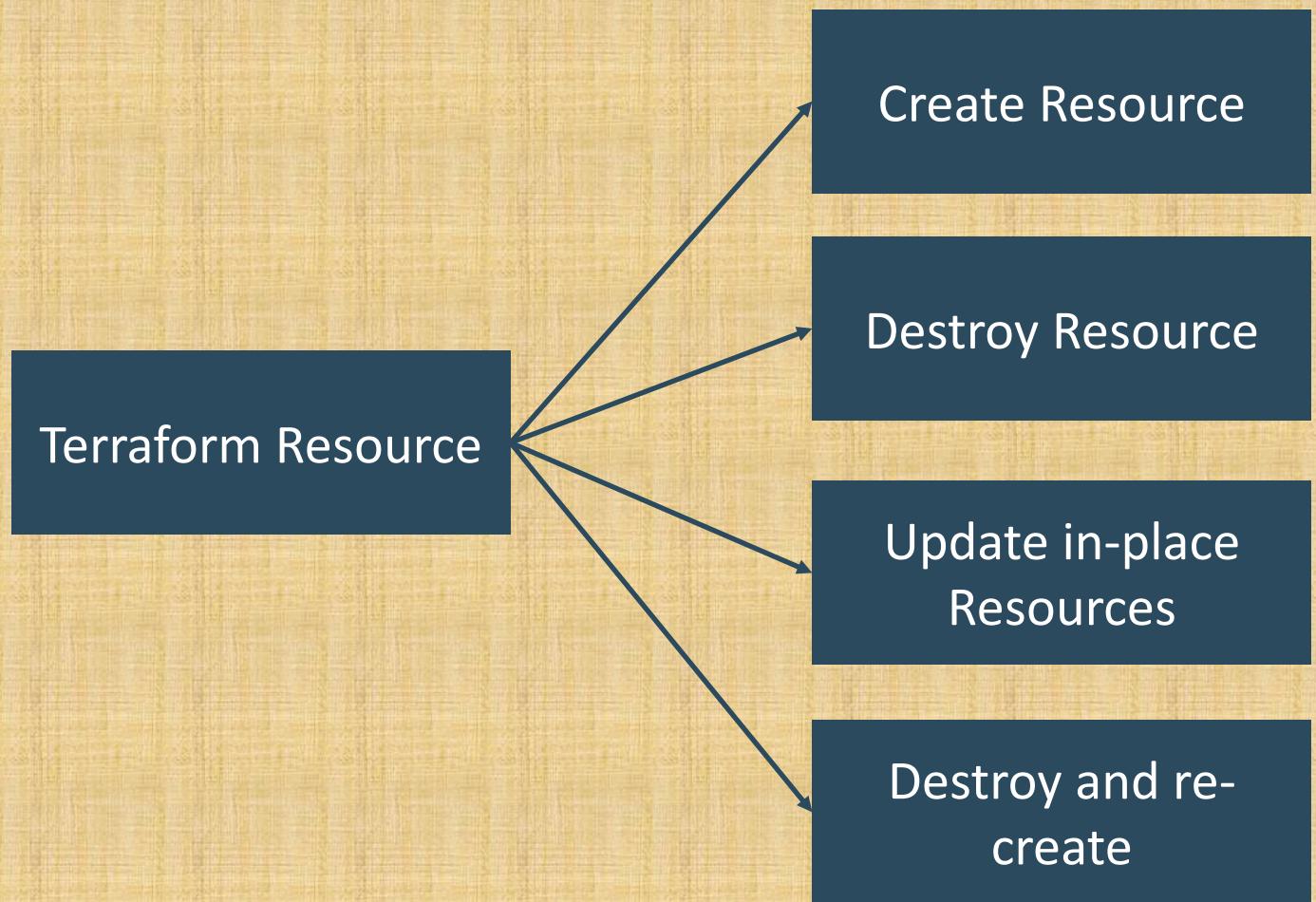
```
# Uses "provider2-westus" provider
resource "azurerm_resource_group" "myrg2" {
    name = "myrg-2"
    location = "West US"
    provider = azurerm.provider2-westus
    count = 2
}
```

Terraform

Resource Behavior



Resource Behavior



Create resources that exist in the configuration but are **not associated** with a real infrastructure object in the state.

Destroy resources that **exist in the state** but no longer exist in the configuration.

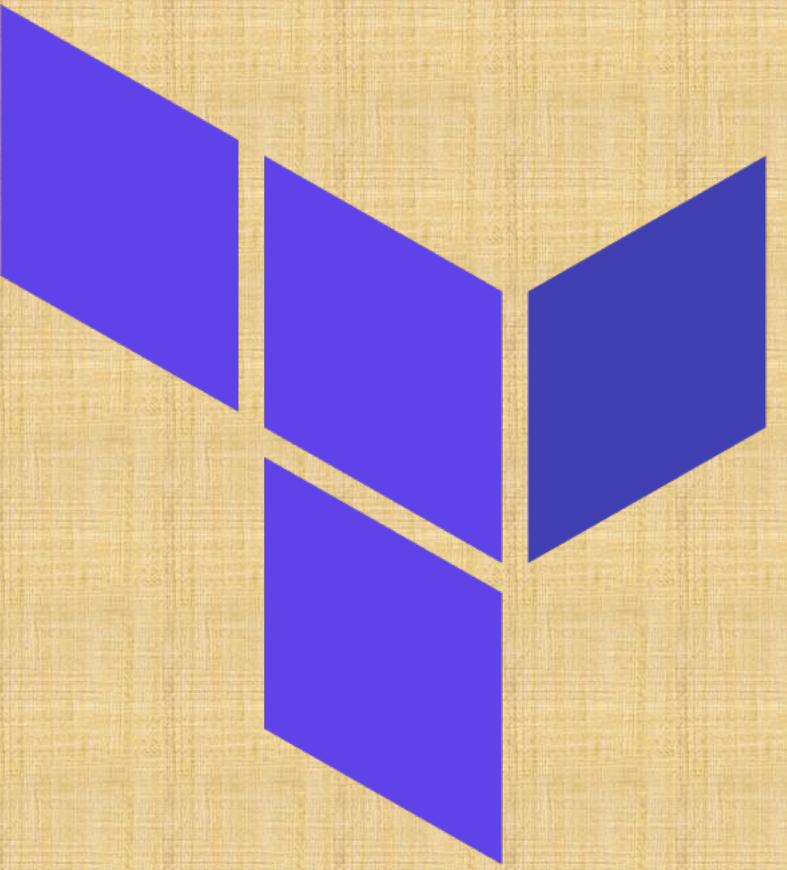
Update **in-place resources** whose arguments have changed.

Destroy and re-create resources whose arguments have changed but which **cannot be updated in-place** due to remote API limitations.

Terraform State



Terraform State



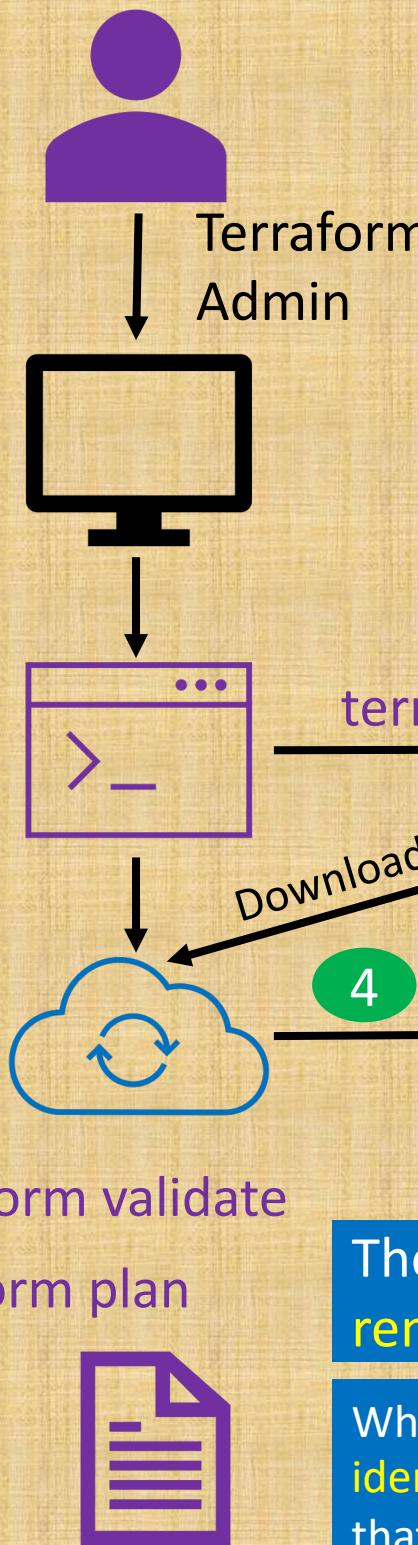
Terraform State

Local Desktop

Terraform CLI

Terraform Azure Provider

Terraform State File
`terraform.tfstate`



Terraform must **store state** about your managed infrastructure and configuration

This state is used by Terraform to map **real world resources** to your **configuration** (**.tf files**), keep track of metadata, and to improve performance for large infrastructures.

This state is stored by default in a local file named "**terraform.tfstate**", but it can also be stored **remotely**, which works better in a **team** environment.

The **primary purpose** of Terraform state is to store **bindings** between objects in a **remote system** and resource instances **declared** in your configuration.

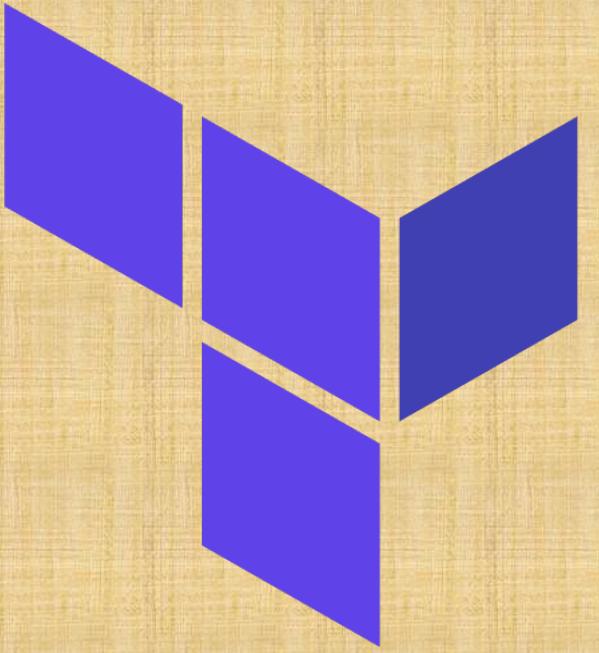
When Terraform creates a remote object in response to a change of configuration, it will record the **identity** of that remote object against a particular resource instance, and then **potentially update or delete** that object in response to future configuration changes.



Terraform

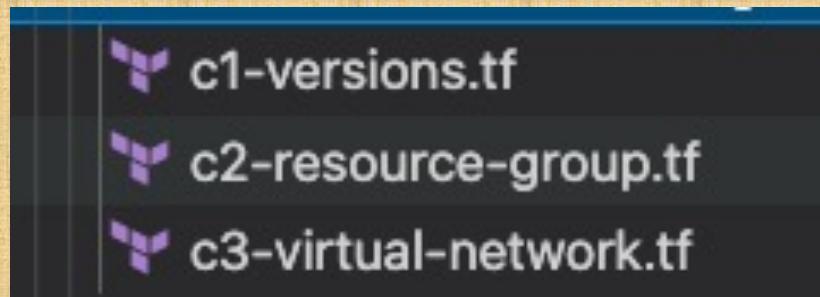
State

Desired & Current



Desired & Current Terraform States

Terraform Configuration Files



Real World Resources

A screenshot of the Azure portal showing the 'myrg-1' Resource Group overview. The page includes sections for Overview, Essentials (Subscription, Tags), Settings (Deployments, Security, Policies, Properties, Locks), and Cost Management. The 'Essentials' section shows the following details:

Deployment	Type	Location
mypublicip-1	Public IP address	East US
mynet-1	Virtual network	East US
vmnic-1	Network interface	East US

Desired State



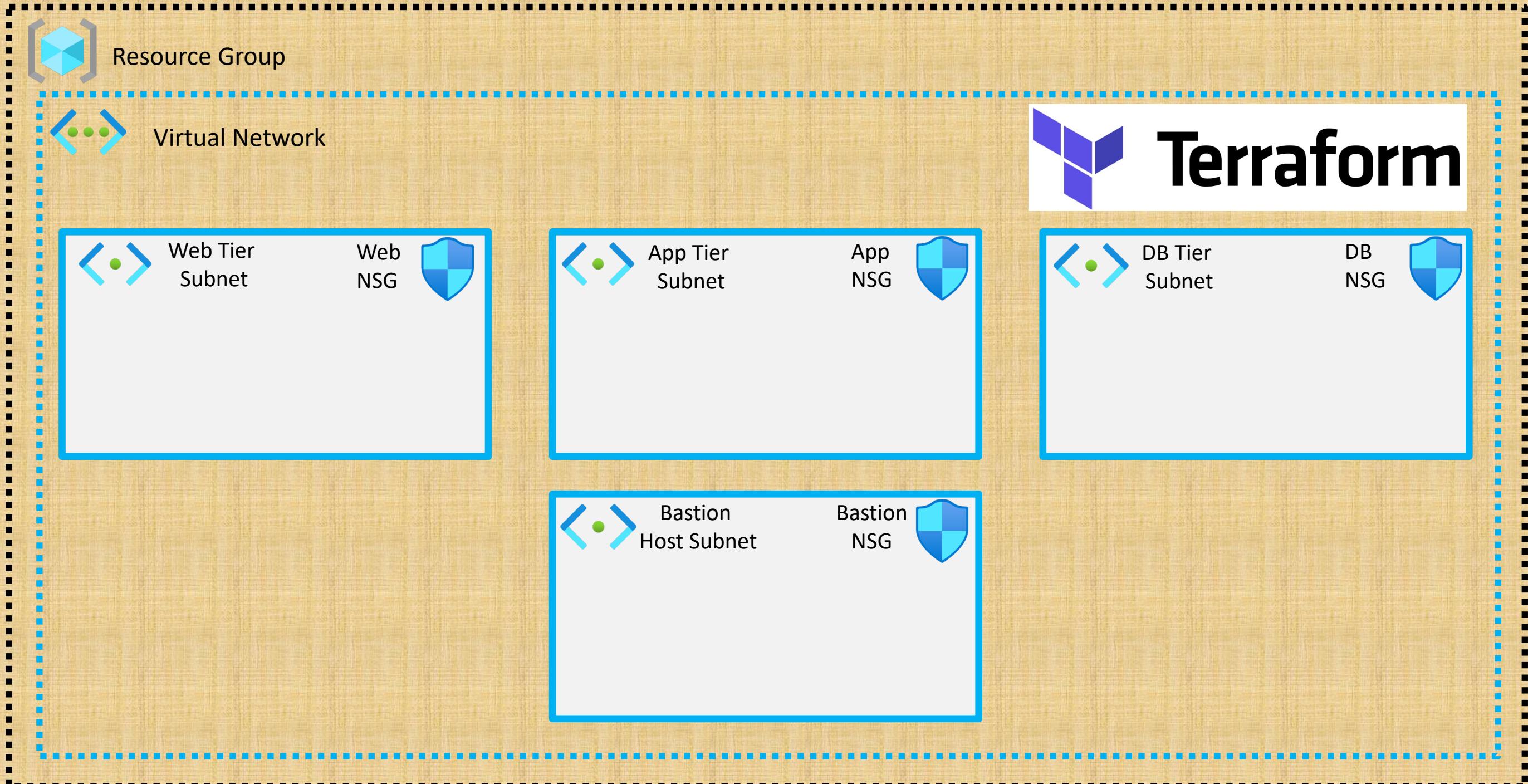
Current State

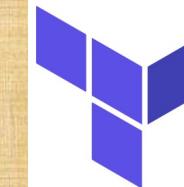
Real-World
Demo 1

Azure Virtual Network Subnets

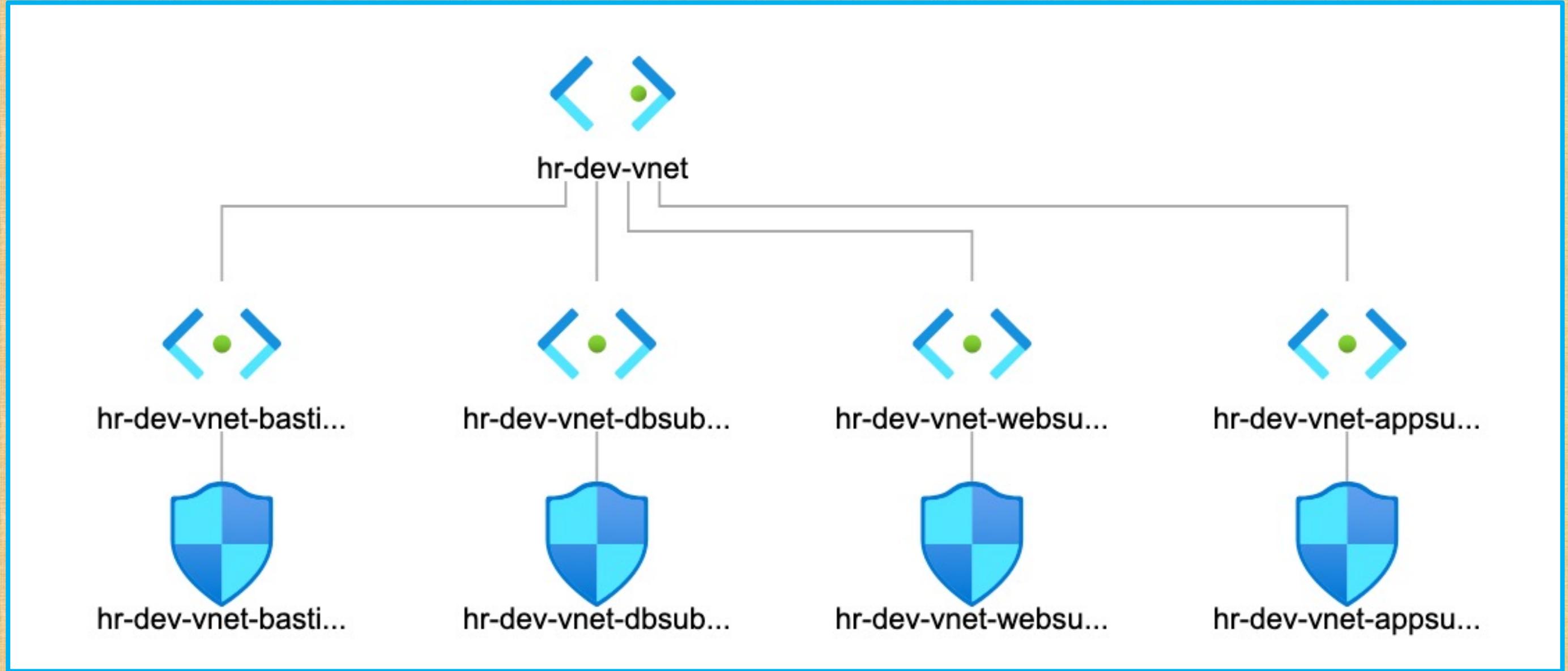
Network Security Group

Azure Virtual Network – 4 Tier Design



 **Terraform**

Azure Virtual Network Topology



Terraform Concepts & Azure Resources

Terraform Settings Block

Terraform Provider Block

Terraform Input Variables

Terraform Local Values Block



azurerm_resource_group



azurerm_virtual_network



azurerm_subnet

Terraform Random Resource

Terraform **for_each** Meta-Argument

Terraform **depends_on** Meta-Argument

Terraform Output Values



azurerm_network_security_group

azurerm_network_security_rule

azurerm_subnet_network_security_group_association

What are we going to Learn ?

```
✓ 10-Azure-Virtual-Network-4Tier
  ✓ terraform-manifests
    ✓ c1-versions.tf
    ✓ c2-generic-input-variables.tf
    ✓ c3-locals.tf
    ✓ c4-random-resources.tf
    ✓ c5-resource-group.tf
    ✓ c6-01-vnet-input-variables.tf
    ✓ c6-02-virtual-network.tf
    ✓ c6-03-web-subnet-and-nsg.tf
    ✓ c6-04-app-subnet-and-nsg.tf
    ✓ c6-05-db-subnet-and-nsg.tf
    ✓ c6-06-bastion-subnet-and-nsg.tf
    ✓ c6-07-vnet-outputs.tf
    ✓ terraform.tfvars
  ⓘ README.md
```

Terraform Input Variables

Terraform Local Values

Azure Virtual Network with Subnets
and Network Security Groups

Virtual Network 4-Tier
Design

Terraform Output Values

Terraform Meta-Arguments
(`for_each` and `depends_on`)

Terraform Variable – `terraform.tfvars`

^ Azure Virtual Network, Subnets and Network Security Groups

10 lectures • 1hr 40min

- Step-01: Introduction to Azure Virtual Network 4-Tier Design using Terraform
- Step-02: Implement Terraform Settings and Provider Blocks 05:43
- Step-03: Define Generic Input Variables 05:40
- Step-04: Define Terraform Local Values and Understand its importance 08:39
- Step-05: Create Random String and Resource Group 11:39
- Step-06: Create VNET Input Variables and VNET Resource 11:33
- Step-07: Create Web Subnet, NSG and NSG-Subnet Associate Resources 10:48
- Step-08: Create NSG Rules, Implement for_each and depends_on Meta-Arguments 18:18
- Step-09: Create VNET Output Values 08:30
- Step-10: Create terraform.tfvars file 06:07
- Step-11: Execute TF Commands, Verify 4Tier VNET and Destroy Resources 13:19



Time it takes to complete this Demo



Real-World Demo

Input variables serve as **parameters** for a Terraform module, allowing aspects of the module to be **customized** without altering the module's own source code, and allowing modules to be **shared** between different configurations.

Demos

Terraform Input Variables

1 Input Variables - Basics

2 Provide Input Variables when prompted during `terraform plan` or `apply`

3 Override default variable values using CLI argument `-var`

4 Override default variable values using Environment Variables (`TF_var_aa`)

5 Provide Input Variables using `terraform.tfvars` files

11 Input Variables using Structural Type: `Object`

6

Provide Input Variables using `<any-name>.tfvars` file with CLI argument `-var-file`

7

Provide Input Variables using `somefilename.auto.tfvars` files

8

Implement complex type **constructors** like `List` & `Map` in Input Variables

9

Implement **Custom Validation Rules** in Variables

10

Protect **Sensitive** Input Variables

13

12 Input Variables using Collection Type: `set`

12

Input Variables using Structural Type: `tuple`

Terraform Variables – Output Values

Output values are like the **return values** of a Terraform module and have several uses

1

A root module can use outputs to **print** certain values in the **CLI output** after running **terraform apply**.

Terraform
Variables
Outputs

2

A child module can use outputs to **expose a subset** of its resource attributes to a **parent module**.

When using **remote state**, root module outputs can be accessed by other configurations via a **terraform_remote_state** data source.

3

Advanced

Output Values – 3 Demos

Demo-1

Basics

Demo-2

Count and
Splat
Expression

Demo-3

for_each and
for loops

Over the process master the **for loops** in Terraform with **Lists** and
Maps

Terraform Variables – Local Values

A local value assigns a **name to an expression**, so you can use that name multiple times within a module without repeating it.

Local values are like a **function's temporary local variables**.

Once a local value is declared, you can reference it in expressions as **local.<NAME>**.

Local values can be helpful to avoid **repeating** the same values or expressions **multiple times** in a configuration

If **overused** they can also make a configuration **hard to read** by future maintainers **by hiding** the actual values used

The ability to easily change the value in a central place is the **key advantage** of local values.

In short, Use local values only in moderation

```
locals {  
    service_name = "forum"  
    owner        = "Community Team"  
}
```

```
locals {  
    # Common tags to be assigned to all resources  
    common_tags = {  
        Service = local.service_name  
        Owner   = local.owner  
    }  
}
```

```
resource "aws_instance" "example" {  
    # ...  
  
    tags = local.common_tags  
}
```

Resource Meta-Arguments – depends_on

Use the `depends_on` meta-argument to handle **hidden** resource or module dependencies that Terraform can't automatically infer.

Explicitly specifying a dependency is only necessary when a resource or module relies on some other resource's behavior but *doesn't access* any of that resource's data in its arguments.

This argument is available in **module blocks** and in all **resource blocks**, regardless of resource type.

Resource
Meta-Argument
`depends_on`

The `depends_on` meta-argument, if present, must be a list of references to **other resources** or **child modules** in the same calling module.

Arbitrary expressions are **not allowed** in the `depends_on` argument value, because its value must be known before Terraform knows resource relationships and thus before it can safely evaluate expressions.

The `depends_on` argument should be used only as a **last resort**. Add comments for future reference about why we added this.

Resource Meta-Arguments – `for_each`

If a resource or module block includes a `for_each` argument whose value is a map or a set of strings, Terraform will create one instance for each member of that map or set.

Each instance has a distinct infrastructure object associated with it, and each is separately created, updated, or destroyed when the configuration is applied.

A given resource or module block **cannot** use both `count` and `for_each`

Resource
Meta-Argument
`for_each`

For set of Strings, `each.key = each.value`
`for_each = toset(["Jack", "James"])`
`each.key = Jack`
`each.key = James`

For Maps, we use `each.key & each.value`
`for_each = {`
 `dev = "myapp1"`
`}`
`each.key = dev`
`each.value = myapp1`

In blocks where `for_each` is set, an additional `each` object is available in expressions, so you can modify the configuration of each instance.
`each.key` — The map key (or set member) corresponding to this instance.
`each.value` — The map value corresponding to this instance. (If a set was provided, this is the same as `each.key`.)

`for_each` with Maps

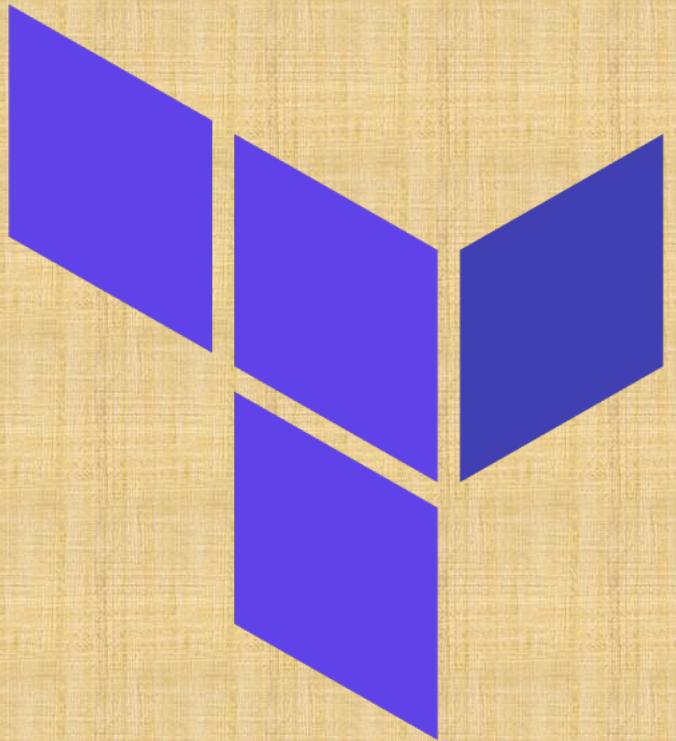
`for_each` with Set of Strings

`for_each` Chaining

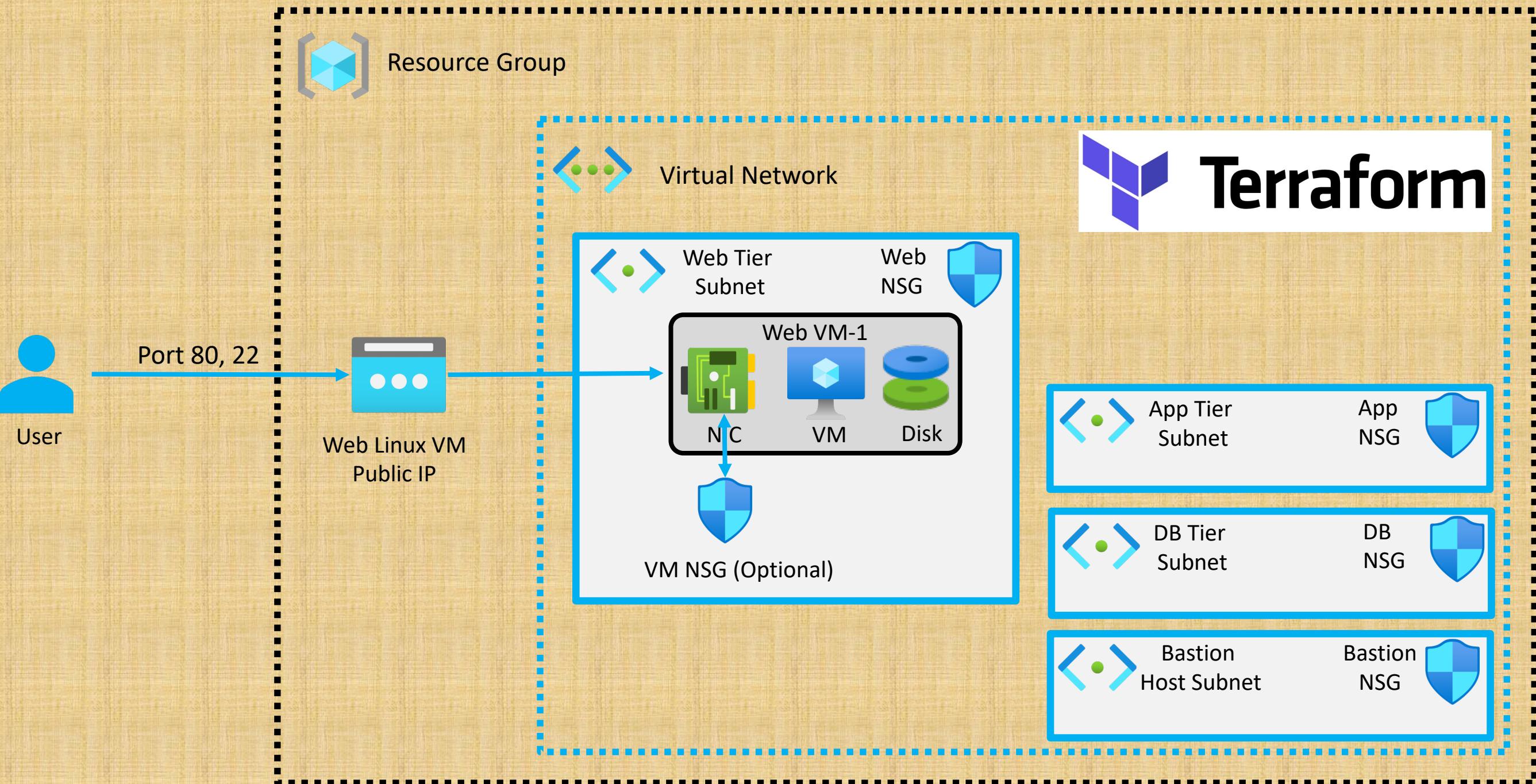


Real-World
Demo 2

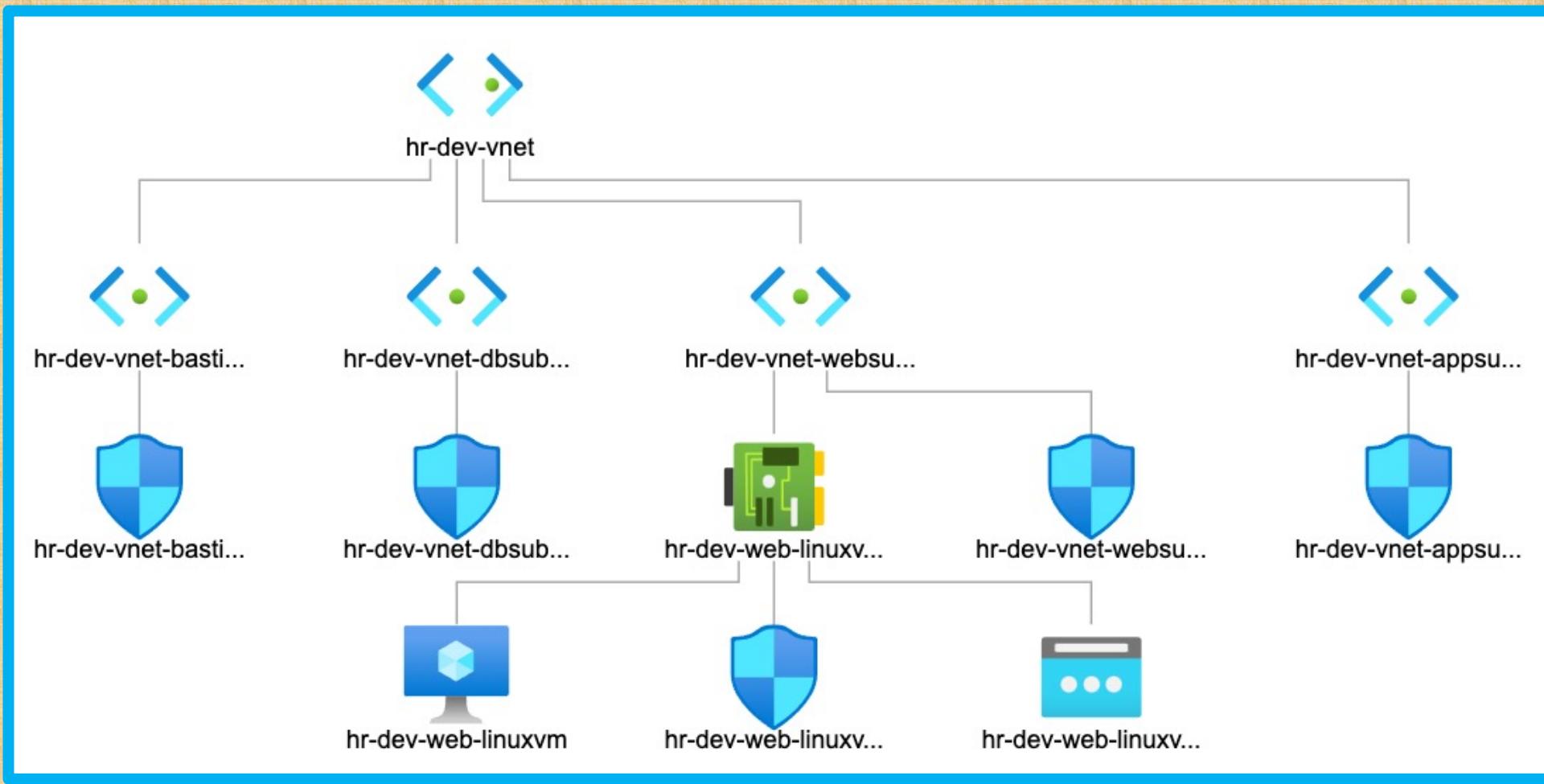
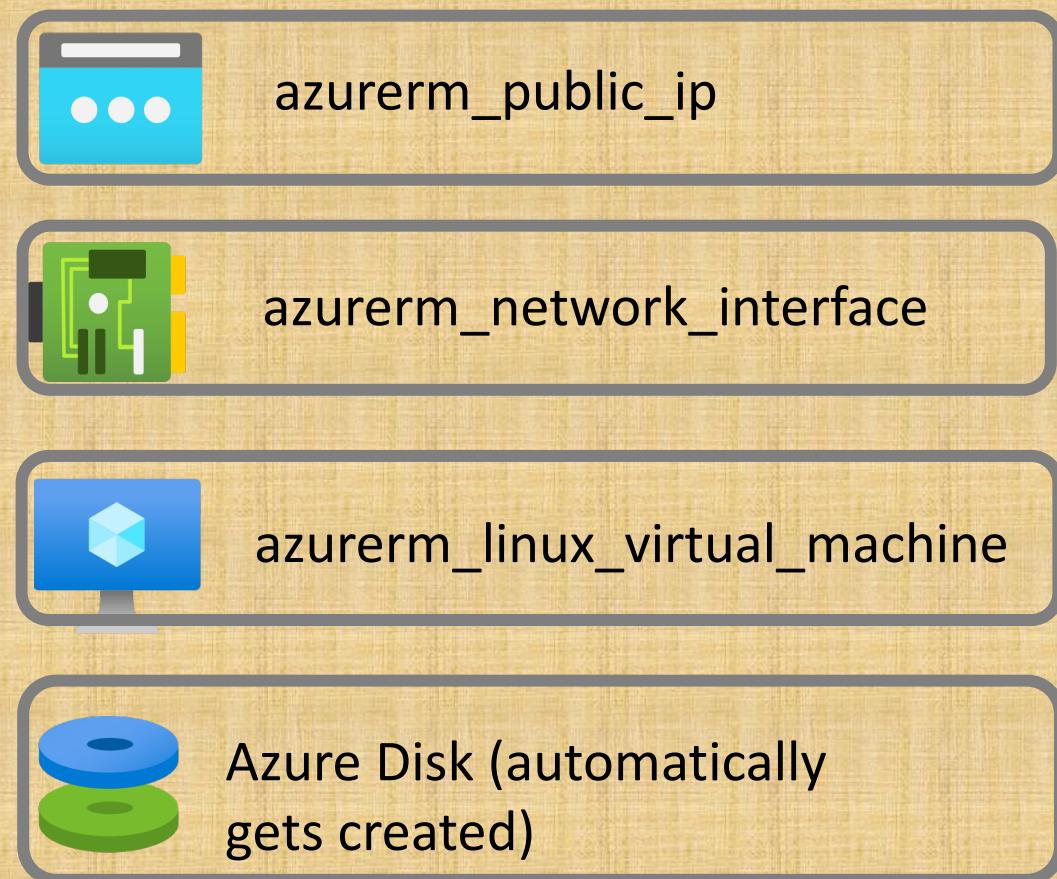
Azure Virtual Machines Linux



Azure Linux Virtual Machine



Azure Resources and Topology



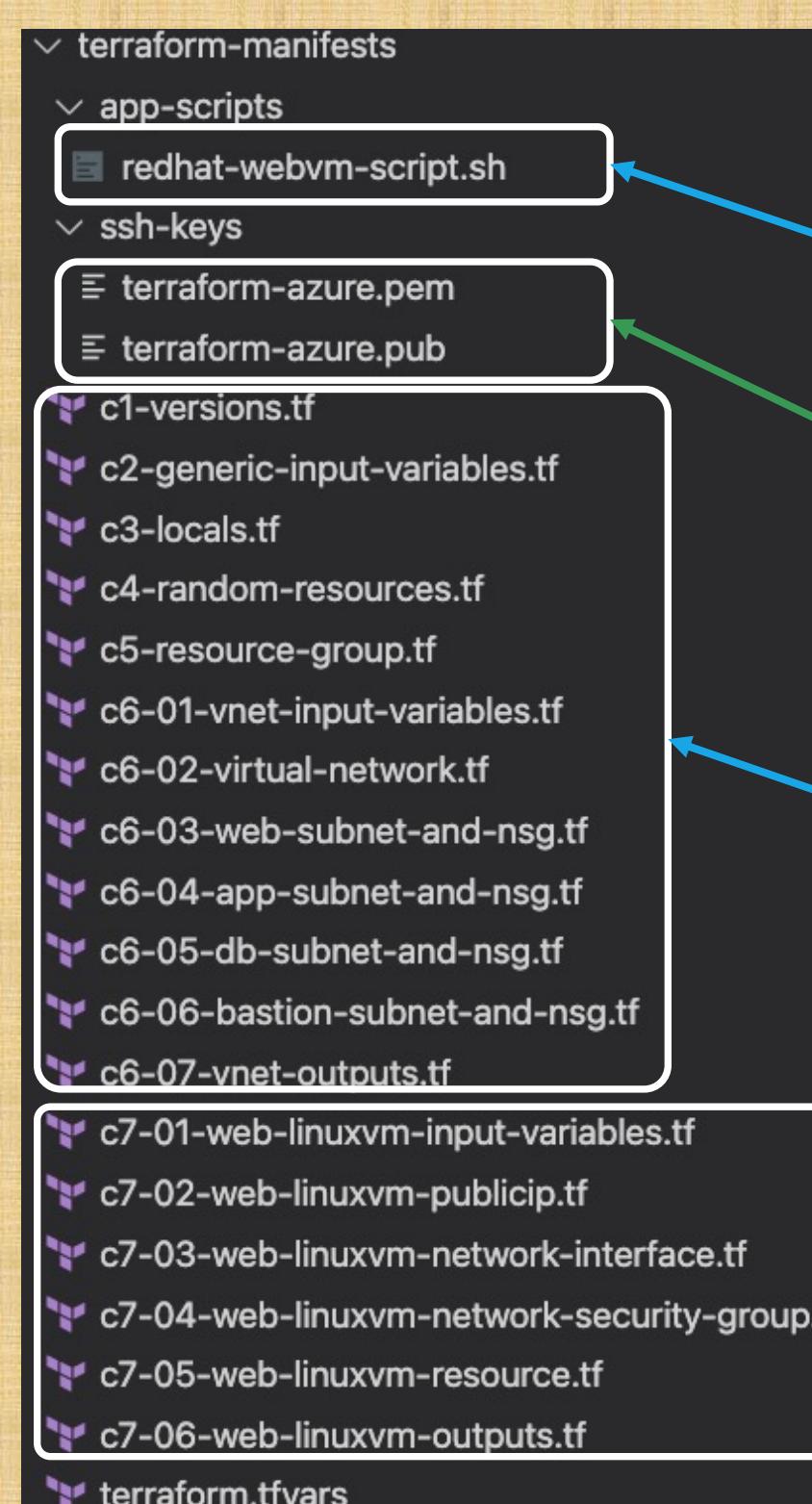
azurerm_network_security_group
azurerm_network_security_rule
azurerm_network_interface_security_group_association

Terraform **file()** Function

Terraform **filebase64()** function

Terraform **base64encode()** function

Terraform Configs



VM Custom Data Script to Bootstrap
Apache Webserver

SSH Keys to Connect to Linux VM

Azure 4-Tier Virtual Network

Azure Linux VM with Public IP and
Network Interface in Web Subnet

^ Azure Linux Virtual Machine using Terraform

9 lectures • 1hr 27min

- Step-01: Introduction to Azure Linux Virtual Machine using Terraform
- ▶ Step-02: Create SSH Keys and Public IP for Linux VM 11:17
- ▶ Step-03: Create Network Interface for Linux VM 10:34
- ▶ Step-04: Review VM NIC Network Security Group 04:31
- ▶ Step-05: Create Linux Virtual Machine Resource, learn file function 13:42
- ▶ Step-06: Create VM Custom Data using Locals Block, learn filebase64, base64encod 12:25
- ▶ Step-07: Review Output Values for Linux VM 05:04
- ▶ Step-08: Execute TF Commands, Verify all the Resources created in Azure Portal 14:50
- ▶ Step-09: Connect via SSH to VM, Verify cloud-init-output log, httpd files and lo 09:07
- ▶ Step-10: Comment VM NIC NSG and Destroy Resources 05:39

Time it takes to complete this Demo

Real-World
Demo

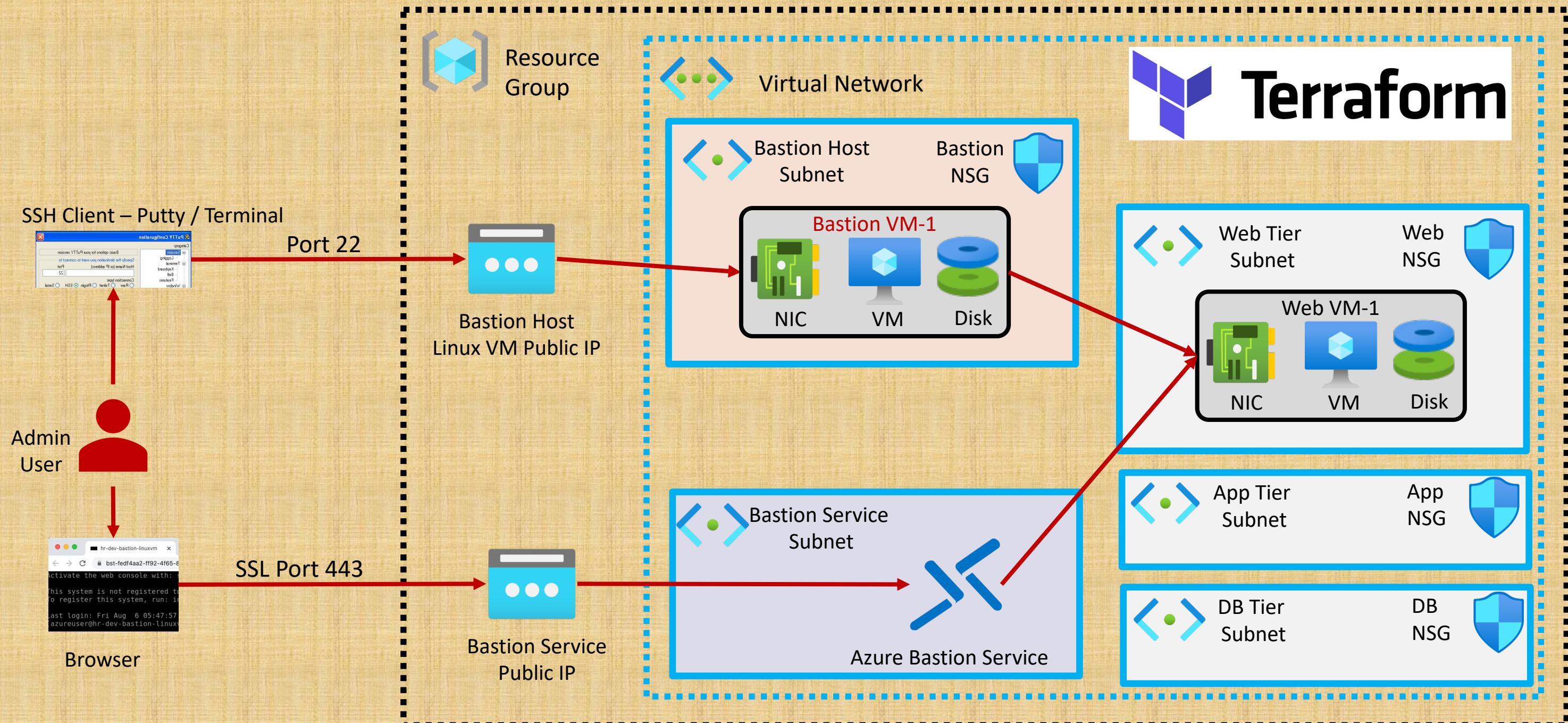


Real-World
Demo

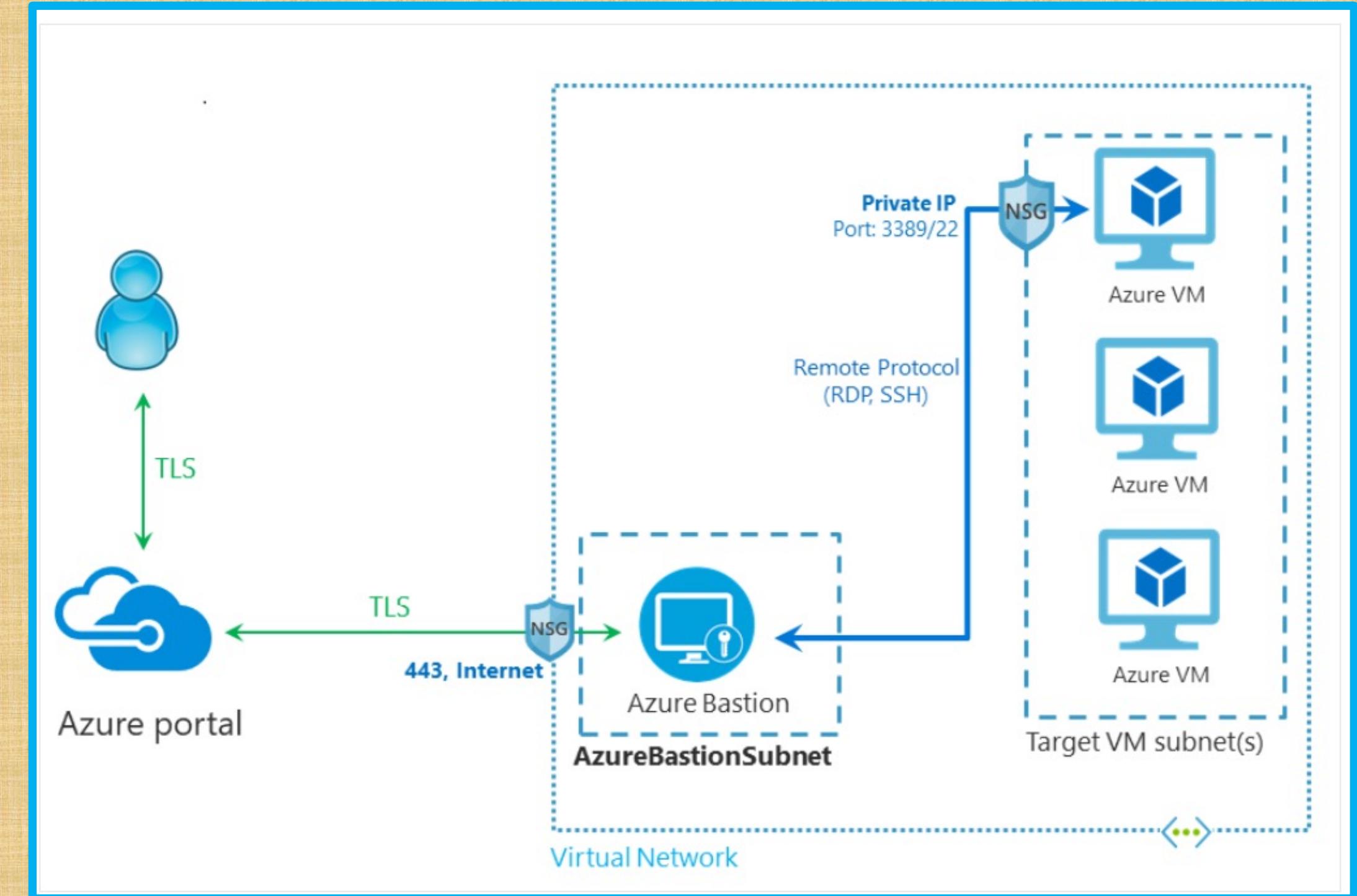
Azure Bastion Host Service and Bastion Host Linux VM



Azure Bastion Host Linux VM & Bastion Service



Azure Bastion Service



Option-1: Azure Bastion Host Linux VM



azurerm_public_ip



azurerm_network_interface



azurerm_linux_virtual_machine



Azure Disk (automatically gets created)



azurerm_network_security_group



azurerm_network_security_rule



azurerm_network_interface_security_group_association

Optional

Azure Resources

Option-2: Azure Bastion Service



azurerm_public_ip



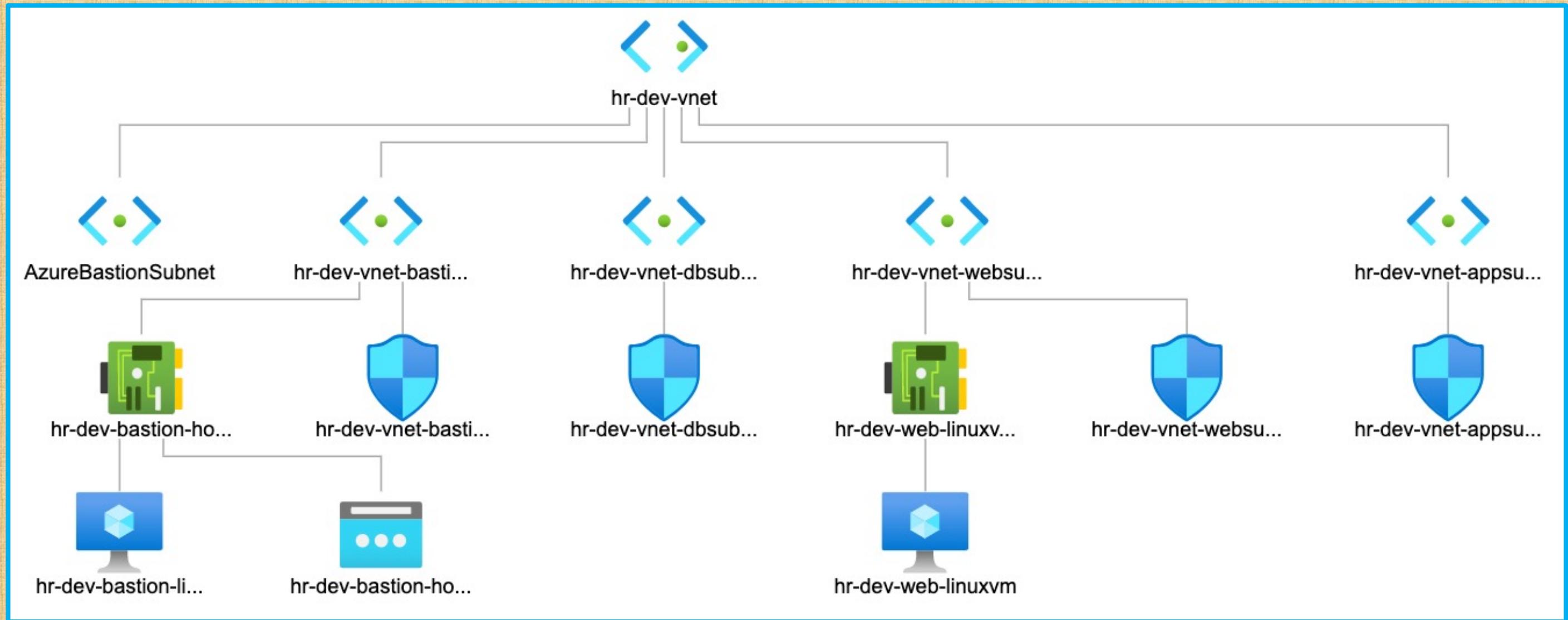
azurerm_subnet

Dedicated Subnet for Azure Bastion Service



azurerm_bastion_host

Azure Bastion Options - Topology



Terraform Configs

```
└── terraform-manifests
    ├── app-scripts
    ├── ssh-keys
    └── c1-versions.tf
    └── c2-generic-input-variables.tf
    └── c3-locals.tf
    └── c4-random-resources.tf
    └── c5-resource-group.tf
    └── c6-01-vnet-input-variables.tf
    └── c6-02-virtual-network.tf
    └── c6-03-web-subnet-and-nsg.tf
    └── c6-04-app-subnet-and-nsg.tf
    └── c6-05-db-subnet-and-nsg.tf
    └── c6-06-bastion-subnet-and-nsg.tf
    └── c6-07-vnet-outputs.tf
    └── c7-01-web-linuxvm-input-variables.tf
    └── c7-02-web-linuxvm-publicip.tf
    └── c7-03-web-linuxvm-network-interface.tf
    └── c7-04-web-linuxvm-network-security-group.tf
    └── c7-05-web-linuxvm-resource.tf
    └── c7-06-web-linuxvm-outputs.tf
    └── c8-01-bastion-host-input-variables.tf
    └── c8-02-bastion-host-linuxvm.tf
    └── c8-03-move-ssh-key-to-bastion-host.tf
    └── c8-04-AzureBastionService.tf
    └── c8-05-bastion-outputs.tf
    └── terraform.tfvars
```

Azure 4-Tier Virtual Network

Azure Linux VM with **Public IP** and Network Interface in Web Subnet

Azure **Bastion Host Linux VM** in Bastion Subnet

Azure **Bastion Service** in AzureBastionSubnet

^ Azure Bastion Host Linux VM and Azure Bastion Service

6 lectures • 59min

- Step-01: Introduction to Azure Bastion Host and Azure Bastion Service
- Step-02: Review Input Variables and Bastion Host Public IP, VM NIC and Linux VM 09:52
- Step-03: Understand TF Concepts - Null Resource, Connection Block, File and Remo 15:09
- Step-04: Create Null Resource, Connection Block, File and Remote Exec Provisione 07:20
- Step-05: Create Azure Bastion Service Resources using Terraform 09:42
- Step-06: Execute TF Commands, Verify Bastion Host VM to Web VM Connect using SSH 09:55
- Step-07: Verify Bastion Service to Connect to Web Linux VM and Destroy Resources 07:20

Time it takes to complete this Demo

Real-World Demo

Terraform Provisioners

Provisioners can be used to model specific actions on the local machine or on a remote machine in order to prepare servers

Passing data into virtual machines and other compute resources

Running configuration management software (packer, chef, ansible)

Creation-Time Provisioners

Failure Behaviour: Continue: Ignore the error and continue with creation or destruction.

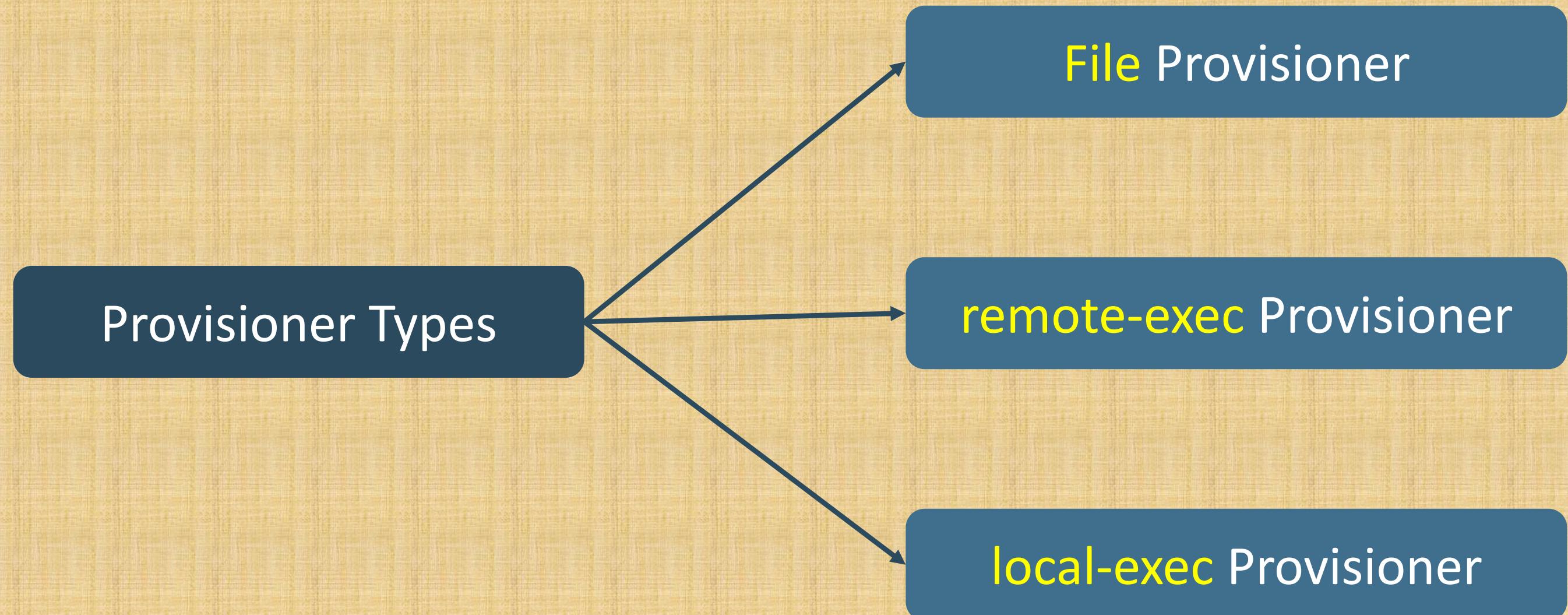
Provisioners are a Last Resort

First-class Terraform provider functionality may be available

Destroy-Time Provisioners

Failure Behaviour: Fail: Raise an error and stop applying (the default behavior). If creation provisioner, taint resource

Types of Provisioners



Terraform Null Resource

Terraform
Null Resource

Terraform
Connection
Block

Terraform File
Provisioner

Terraform
remote-exec
Provisioner

Terraform New Concepts Introduced

Connect to Azure Bastion Host VM from Terraform CLI
Terminal

Push `terraform-azure.pem` to Bastion Host VM

Provide permissions `chmod 400` to `terraform-azure.pem`
after copied to Bastion Host VM

File Provisioner

- File Provisioner is used to **copy files or directories** from the machine executing Terraform to the **newly created resource**.
- The file provisioner supports both **ssh** and **winrm** type of connections

```
## File Provisioner: Copies the terraform-key.pem
provisioner "file" {
  source      = "ssh-keys/terraform-azure.pem"
  destination = "/tmp/terraform-azure.pem"
}
```

remote-exec Provisioner

remote-exec Provisioner

- The remote-exec provisioner invokes a script on a remote resource after it is created.
- This can be used to run a configuration management tool, bootstrap into a cluster, etc.

```
## Remote Exec Provisioner: Using remote-exec provisioner
provisioner "remote-exec" {
  inline = [
    "sudo chmod 400 /tmp/terraform-azure.pem"
  ]
}
```

Connection Block

Connection Block

Most provisioners require **access** to the **remote resource** via **SSH** or **WinRM**, and expect a nested **connection block** with details about how to **connect**.

Expressions in connection blocks **cannot** refer to their parent resource by **name**. Instead, they can use the special **self** object.

Connection Block for Provisioners to connect to Azure VM Instance

```
connection {  
    type = "ssh"  
    host = azurerm_linux_virtual_machine.bastion_host_linuxvm.public_ip_address  
    user = azurerm_linux_virtual_machine.bastion_host_linuxvm.admin_username  
    private_key = file("${path.module}/ssh-keys/terraform-azure.pem")  
}
```

Terraform Null Resource

If you need to run provisioners that aren't directly associated with a specific resource, you can associate them with a `null_resource`.

Instances of `null_resource` are treated like normal resources, but they don't do anything.

Same as other resource, you can configure provisioners and connection details on a `null_resource`.

local-exec Provisioner

local-exec Provisioner

- The local-exec provisioner invokes a local executable after a resource is created.
- This invokes a process on the machine running Terraform, not on the resource.

```
# local-exec provisioner-1 (Creation-Time Provisioner - Triggered during Create Resource)
provisioner "local-exec" {
    command = "echo ${azurerm_linux_virtual_machine.mylinuxvm.public_ip_address} >> creation-
    working_dir = "local-exec-output-files/"
}

# local-exec provisioner-2 - (Destroy-Time Provisioner - Triggered during Destroy Resource)
provisioner "local-exec" {
    when      = destroy
    command   = "echo Destroy-time provisioner Instance Destroyed at `date` >> destroy-time.tx
    working_dir = "local-exec-output-files/"
}
```

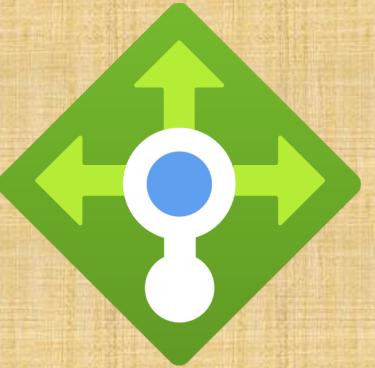
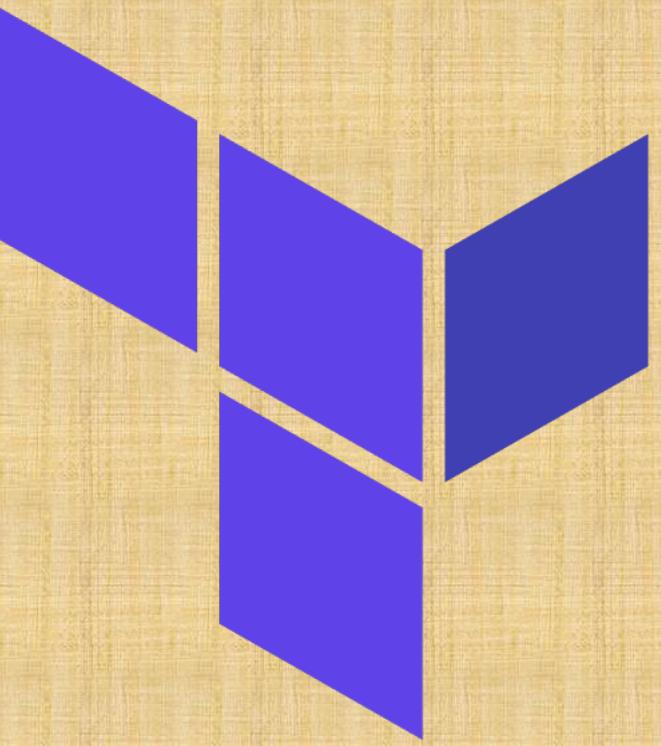
Creation Time Provisioner (by default)

Destroy Time Provisioner (when = destroy)

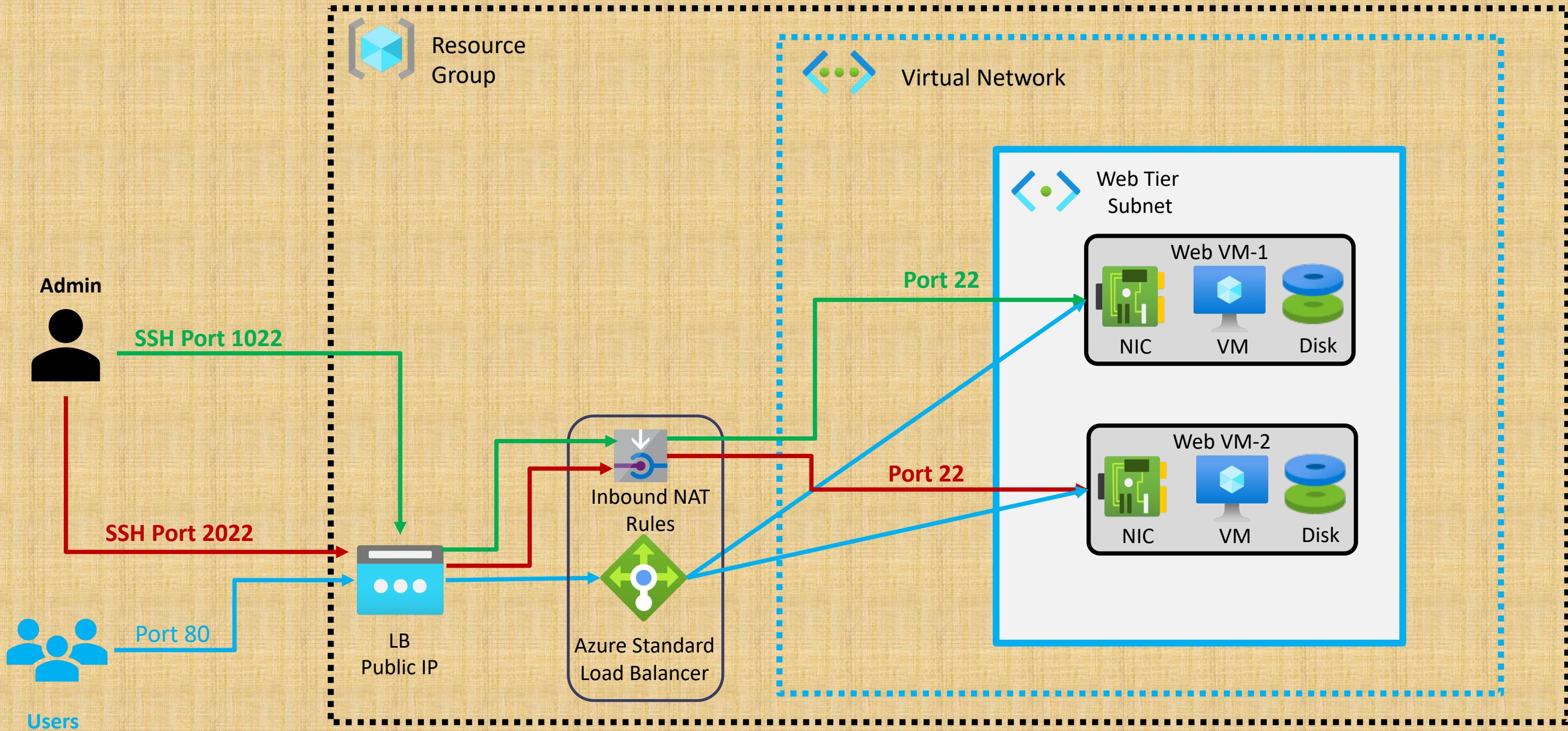


Azure Standard Load Balancer (using Portal)

Real-World
Demo



Azure Standard Load Balancer



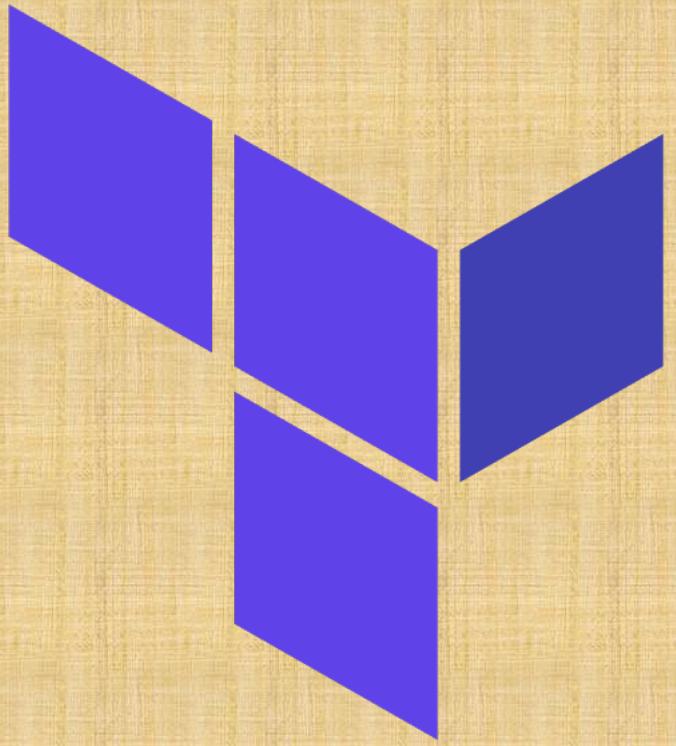


Load Balancer



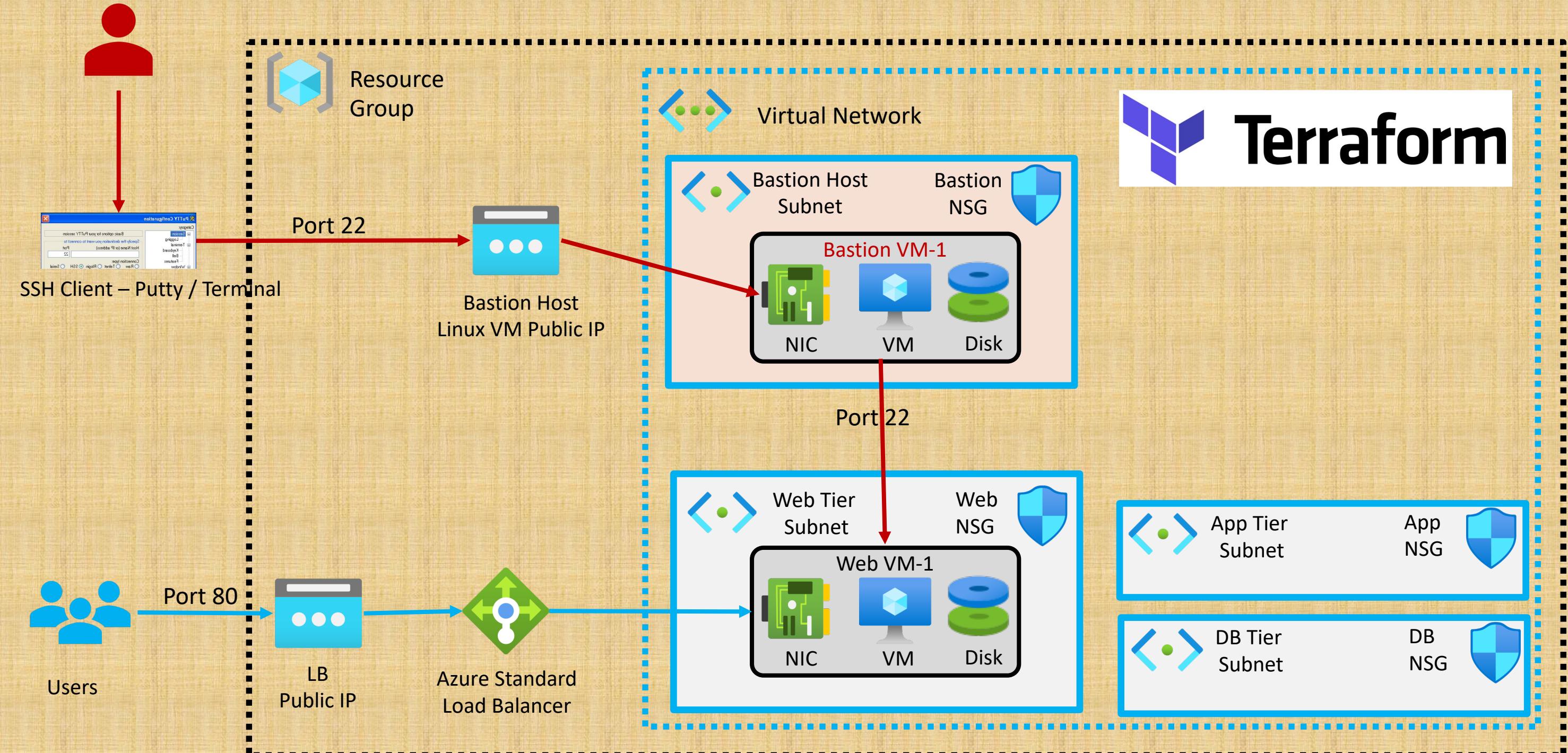
Azure Standard Load Balancer

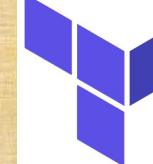
Real-World
Demo



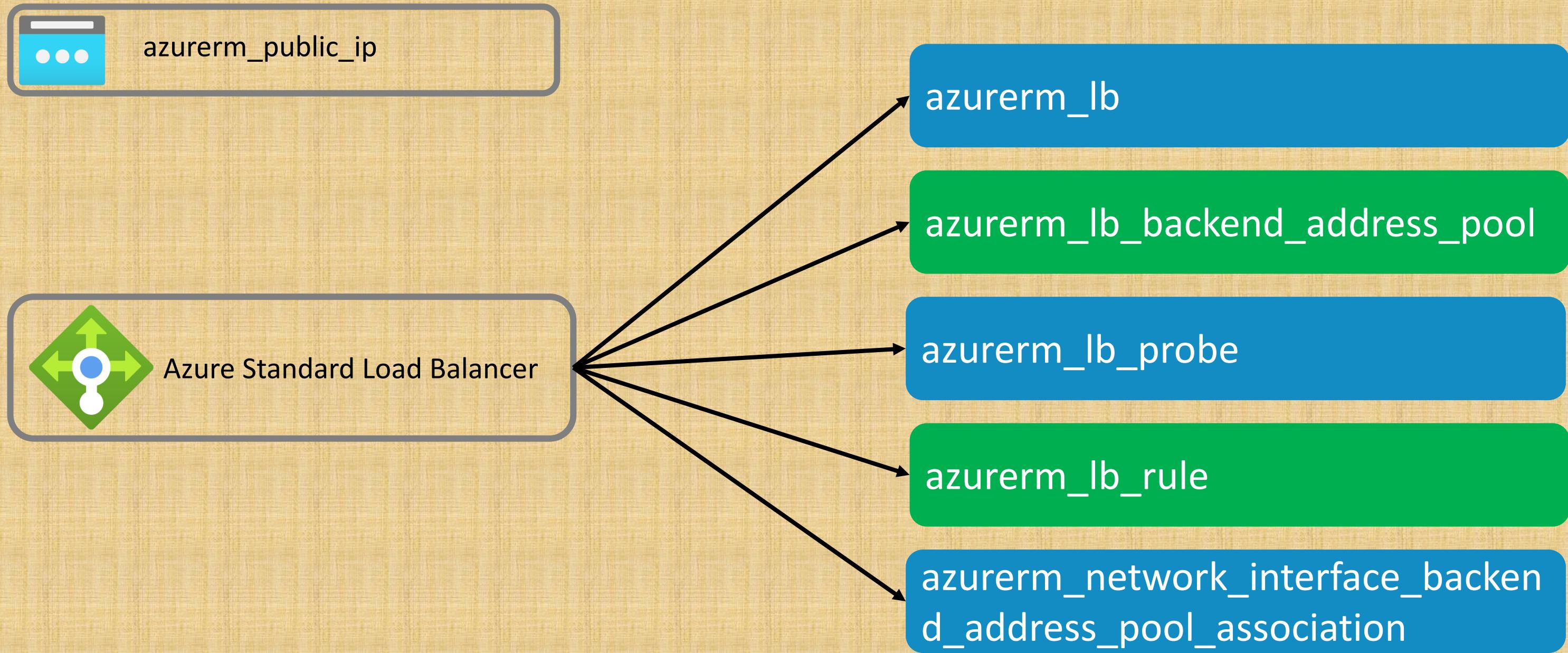
Virtual Machines

Azure Standard Load Balancer – Internet Facing

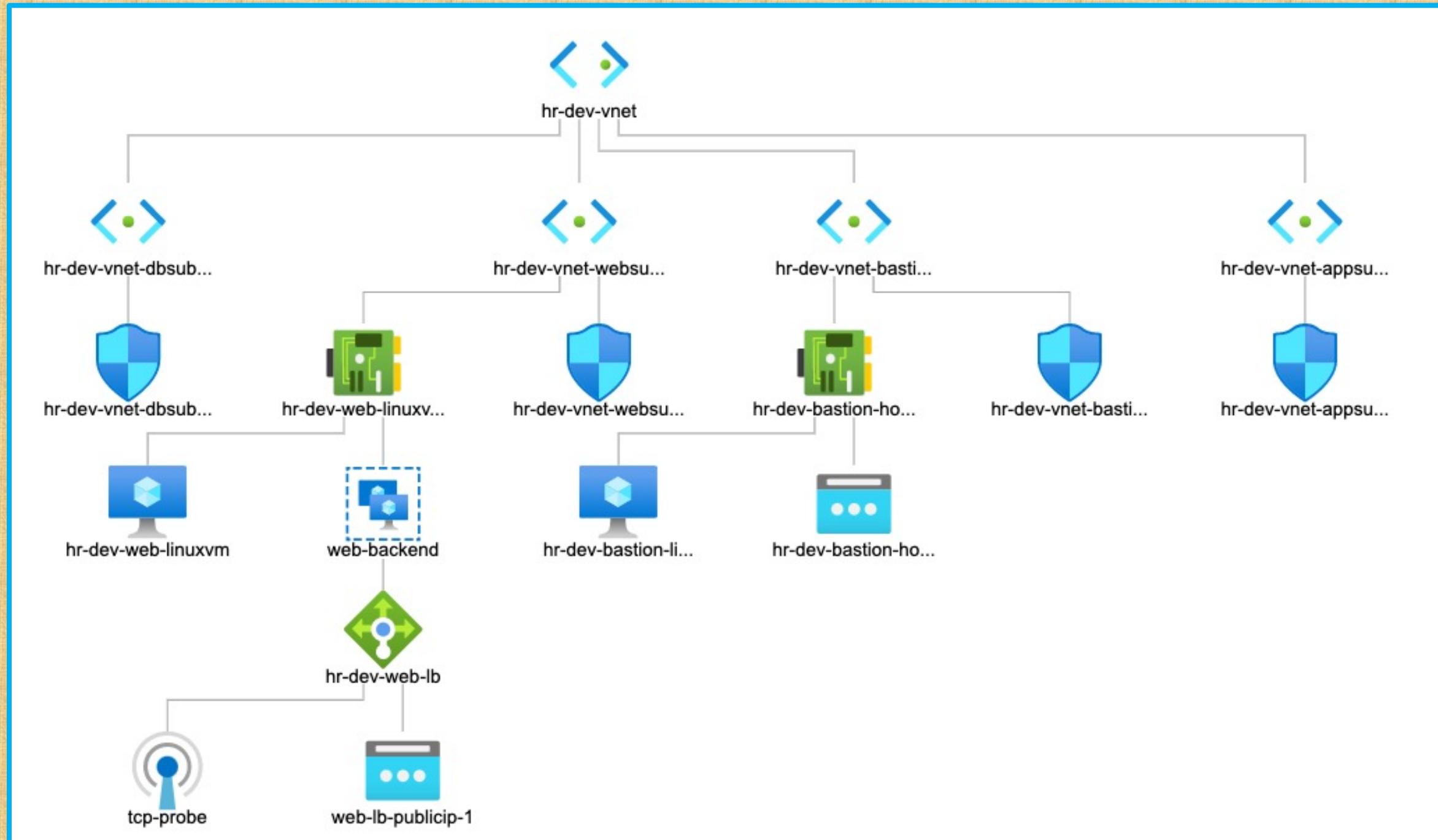


 **Terraform**

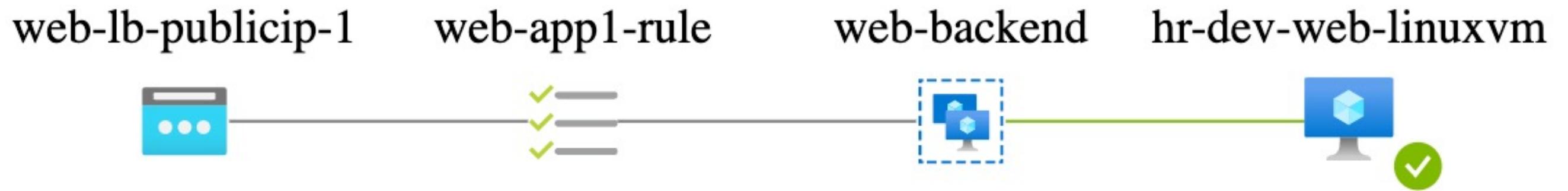
Azure Resources - Internet Facing



Azure Load Balancer - Topology



Azure Load Balancer – Topology



Terraform Configs

terraform-manifests

- > app-scripts
- > ssh-keys
- ↳ c1-versions.tf
- ↳ c2-generic-input-variables.tf
- ↳ c3-locals.tf
- ↳ c4-random-resources.tf
- ↳ c5-resource-group.tf

- ↳ c6-01-vnet-input-variables.tf
- ↳ c6-02-virtual-network.tf
- ↳ c6-03-web-subnet-and-nsg.tf
- ↳ c6-04-app-subnet-and-nsg.tf
- ↳ c6-05-db-subnet-and-nsg.tf
- ↳ c6-06-bastion-subnet-and-nsg.tf
- ↳ c6-07-vnet-outputs.tf

- ↳ c7-01-web-linuxvm-input-variables.tf
- ↳ c7-02-web-linuxvm-publicip.tf
- ↳ c7-03-web-linuxvm-network-interface.tf
- ↳ c7-04-web-linuxvm-network-security-group.tf
- ↳ c7-05-web-linuxvm-resource.tf
- ↳ c7-06-web-linuxvm-outputs.tf

- ↳ c8-01-bastion-host-input-variables.tf
- ↳ c8-02-bastion-host-linuxvm.tf
- ↳ c8-03-move-ssh-key-to-bastion-host.tf
- ↳ c8-04-AzureBastionService.tf
- ↳ c8-05-bastion-outputs.tf

Azure Virtual Network with Subnets and Network Security Groups

Azure Virtual Machines + Network Interface + Public IP

Azure Bastion Host Linux VM – Enabled
Azure Bastion Service - Disabled

Azure Standard Load Balancer - Web

- ↳ c9-01-web-loadbalancer-input-variables.tf
- ↳ c9-02-web-loadbalancer-resource.tf
- ↳ c9-03-web-loadbalancer-outputs.tf
- ↳ terraform.tfvars

^ **Azure Standard Load Balancer Basics using Terraform**

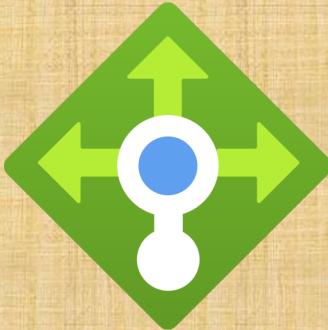
3 lectures • 36min

- Step-01: Introduction to Standard Load Balancer using Terraform
- Step-02: Create Public IP, LB, Backend Pool Resources 09:57
- Step-03: Create LB Probe, LB Rule, LB Backend Pool and VM Nic Associate Resource 13:56
- Step-04: Execute TF Commands, Verify LB Resources and Destroy Resources 11:42



Time it
takes to
complete
this Demo

Real-World
Demo



Load Balancer

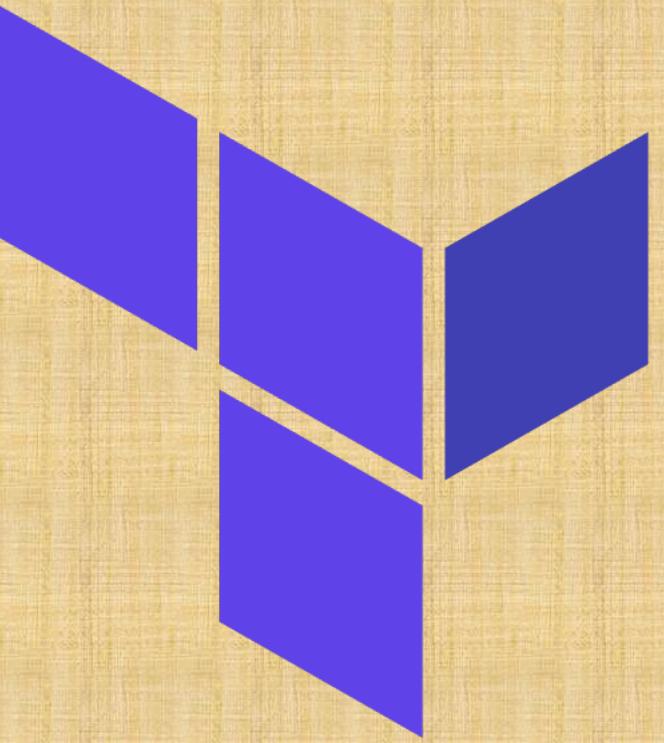


Virtual Machines

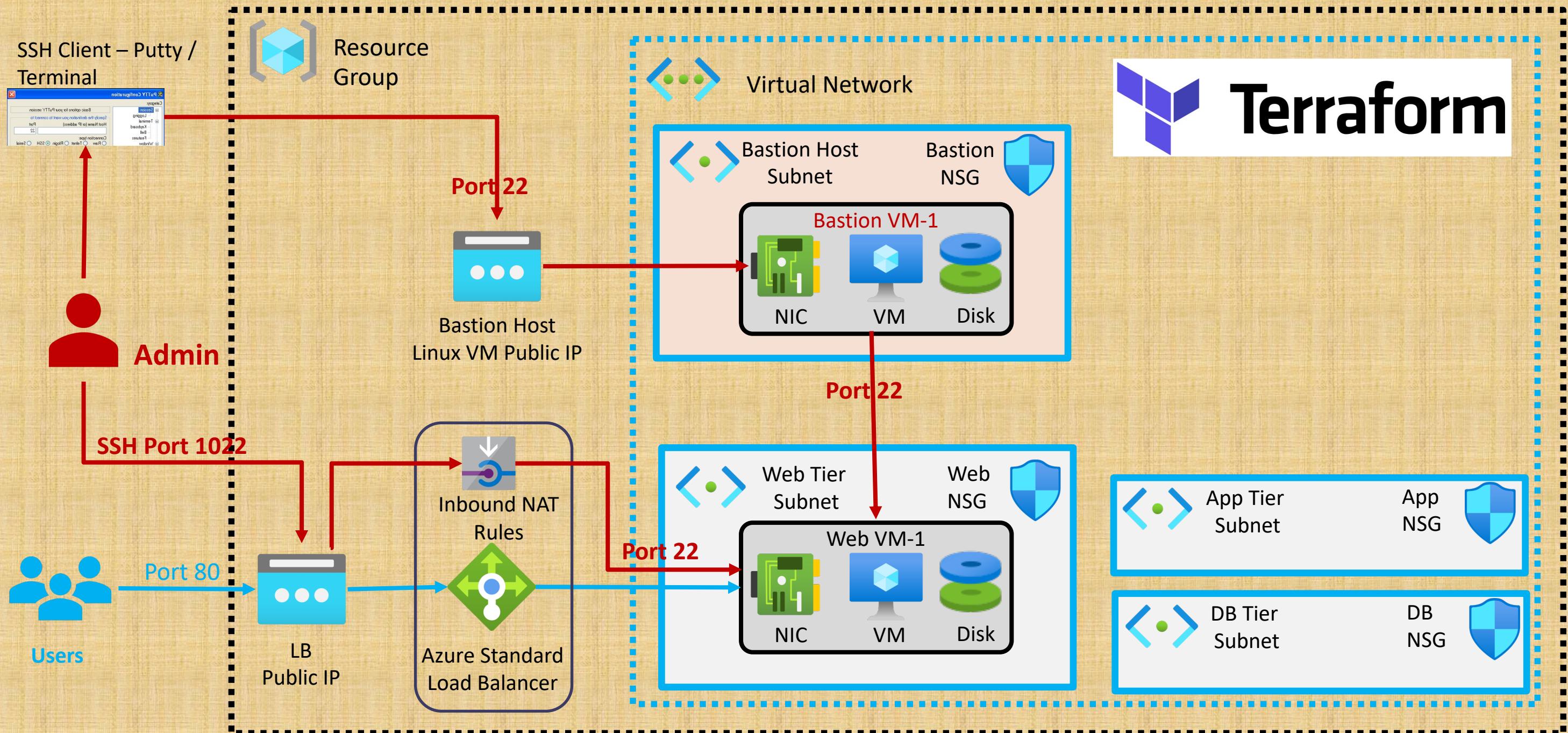


Azure Standard Load Balancer Inbound NAT Rules

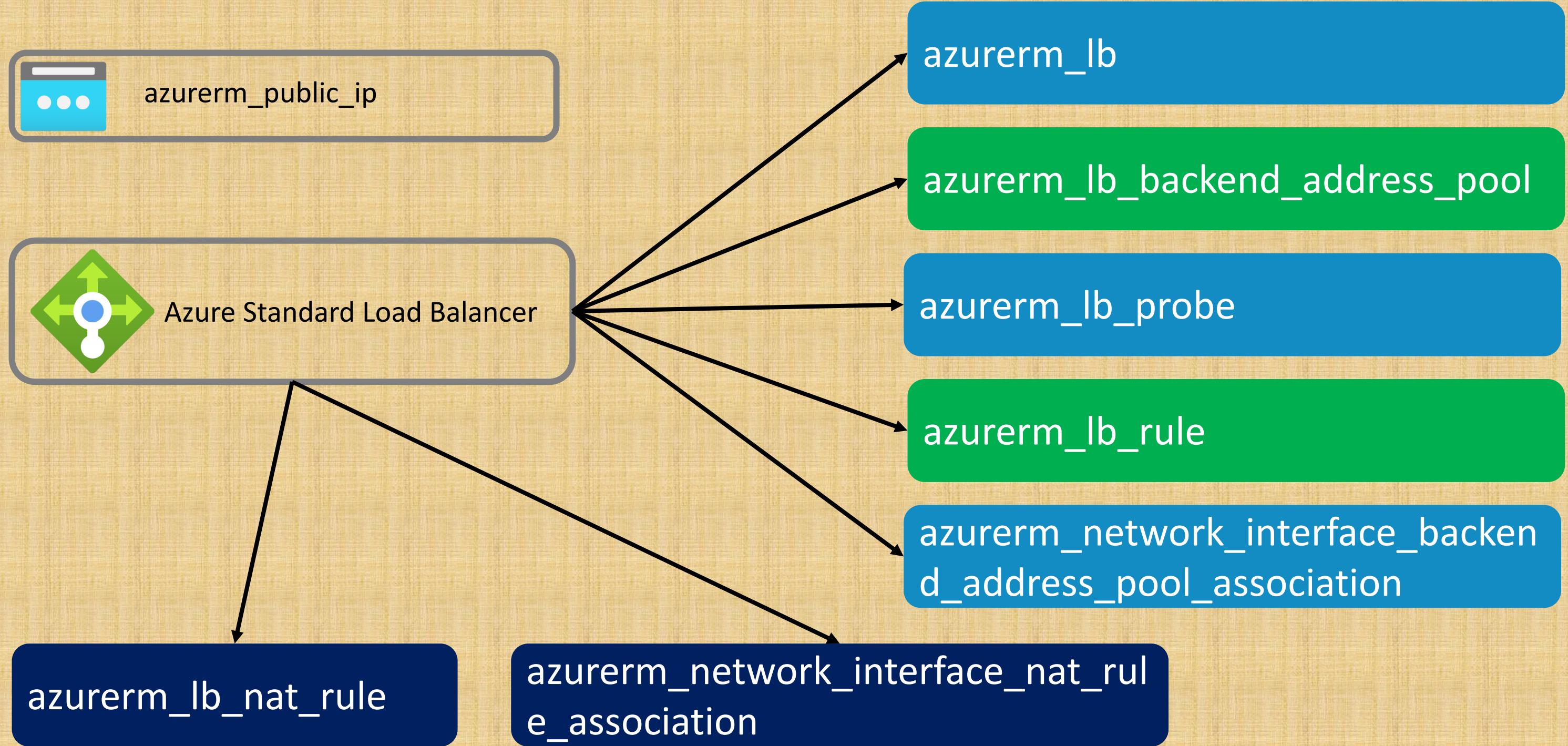
Real-World
Demo



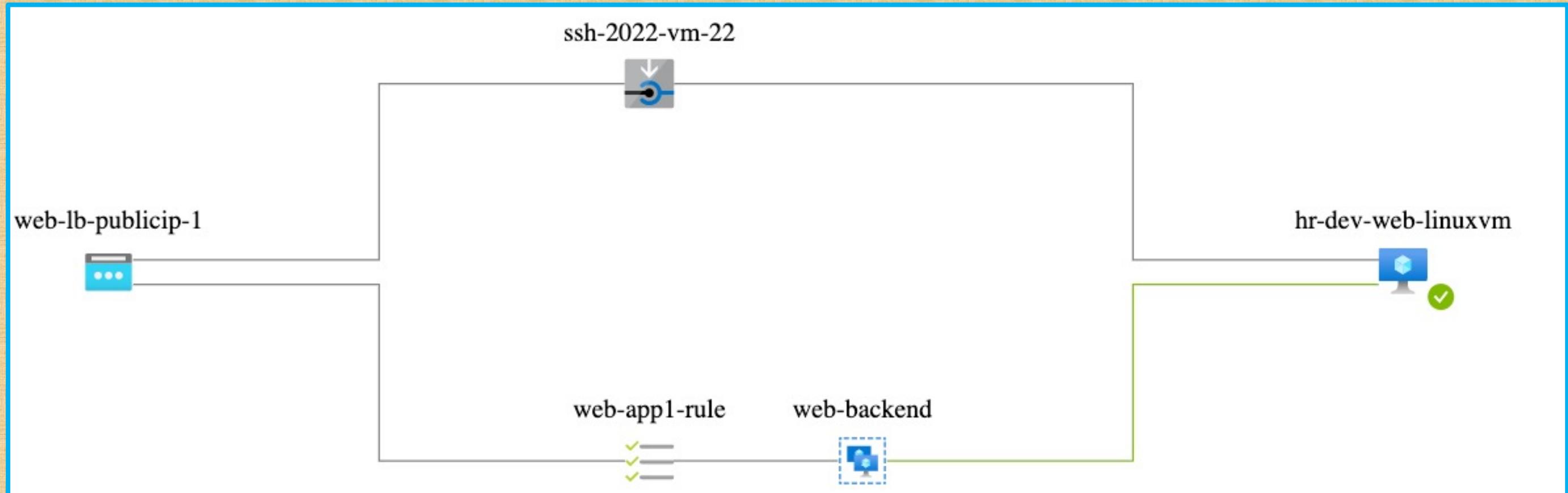
Azure Standard Load Balancer – Inbound NAT Rules



Azure Resources



Azure Load Balancer – Inbound NAT Rules Topology



Terraform Configs

terraform-manifests

- > app-scripts
- > ssh-keys
- ↳ c1-versions.tf
- ↳ c2-generic-input-variables.tf
- ↳ c3-locals.tf
- ↳ c4-random-resources.tf
- ↳ c5-resource-group.tf

- ↳ c6-01-vnet-input-variables.tf
- ↳ c6-02-virtual-network.tf
- ↳ c6-03-web-subnet-and-nsg.tf
- ↳ c6-04-app-subnet-and-nsg.tf
- ↳ c6-05-db-subnet-and-nsg.tf
- ↳ c6-06-bastion-subnet-and-nsg.tf
- ↳ c6-07-vnet-outputs.tf

- ↳ c7-01-web-linuxvm-input-variables.tf
- ↳ c7-02-web-linuxvm-publicip.tf
- ↳ c7-03-web-linuxvm-network-interface.tf
- ↳ c7-04-web-linuxvm-network-security-group.tf
- ↳ c7-05-web-linuxvm-resource.tf
- ↳ c7-06-web-linuxvm-outputs.tf

- ↳ c8-01-bastion-host-input-variables.tf
- ↳ c8-02-bastion-host-linuxvm.tf
- ↳ c8-03-move-ssh-key-to-bastion-host.tf
- ↳ c8-04-AzureBastionService.tf
- ↳ c8-05-bastion-outputs.tf

Azure Virtual Network with Subnets and Network Security Groups

Azure Virtual Machines + Network Interface + Public IP

Azure Bastion Host Linux VM – Enabled
Azure Bastion Service - Disabled

Azure Standard Load Balancer – Web + Inbound NAT Rules

- ↳ c9-01-web-loadbalancer-input-variables.tf
- ↳ c9-02-web-loadbalancer-resource.tf
- ↳ c9-03-web-loadbalancer-outputs.tf
- ↳ c9-04-web-loadbalancer-inbound-nat-rules.tf

^ Azure Standard Load Balancer - Inbound NAT Rules

- ❑ Step-01: Introduction to Load Balancer Inbound NAT Rules
- Step-02: Create LB NAT Rule Resources
- Step-03: Execute TF Commands, Verify Inbound NAT via LB and Destroy Resources

2 lectures • 12min

Time it takes to complete this Demo

Real-World Demo



Load Balancer

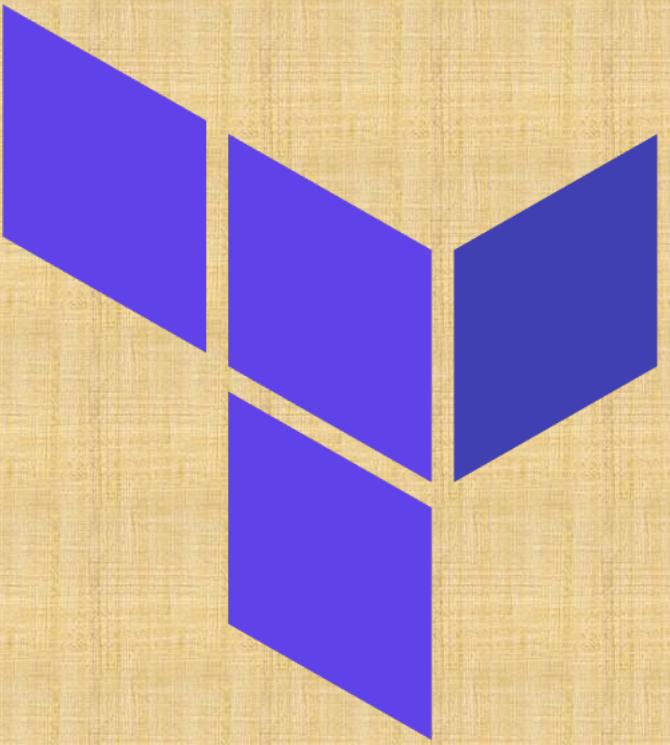


Virtual Machines

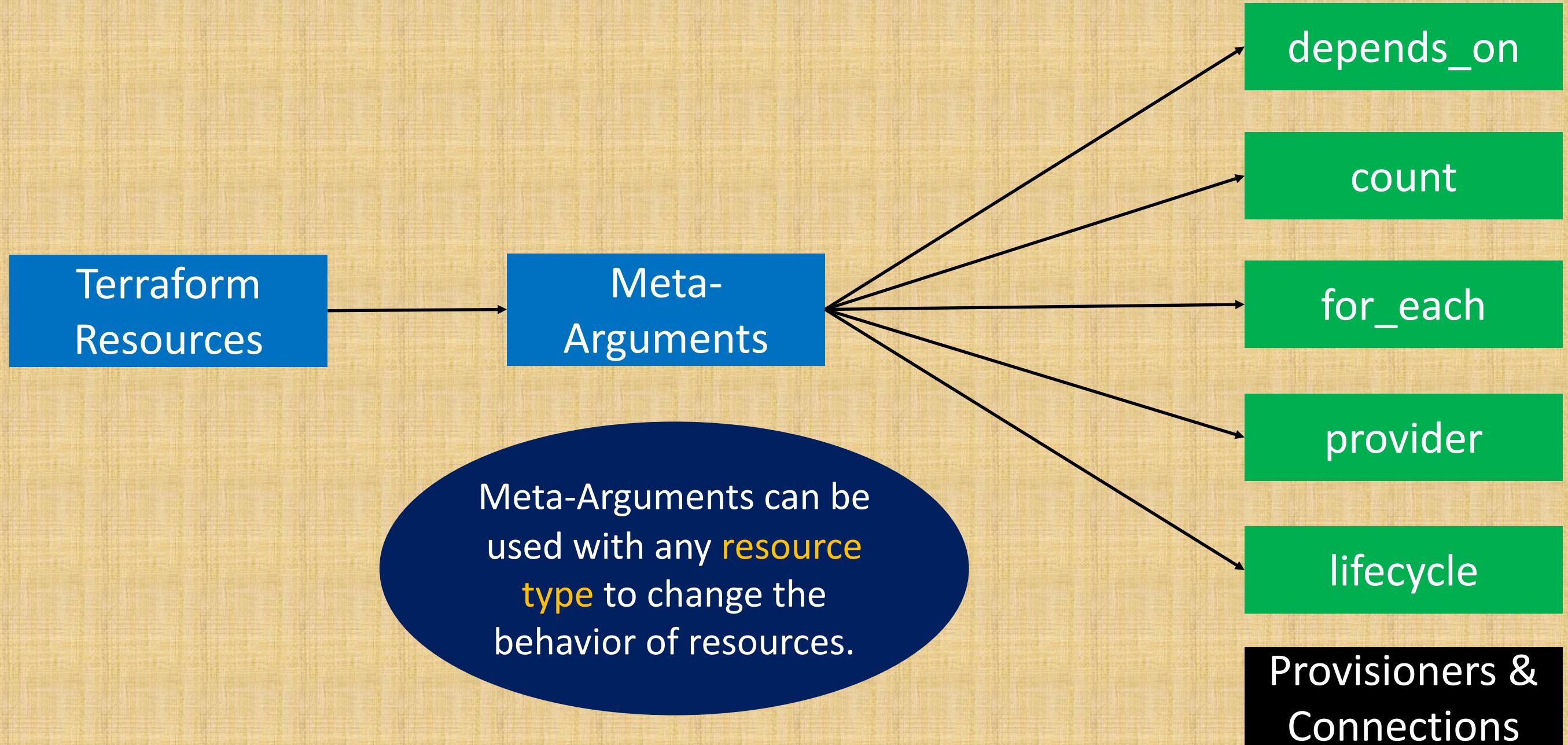


Real-World
Demo

Azure Multiple VMs Meta-Argument Count



Resource Meta-Arguments



Practical Example with Step-by-Step Documentation on Github

Demos for Meta-Arguments

- > 10-Meta-Argument-dependends_on
- > 11-01-Terraform-Azure-Linux-Virtual-Machine
- > 11-02-Meta-Argument-count
- > 12-Meta-Argument-for_each-Maps
- > 13-Meta-Argument-for_each-ToSet
- > 14-Meta-Argument-for_each-Chaining
- > 15-Meta-Argument-lifecycle-create_before_destroy
- > 16-Meta-Argument-lifecycle-prevent_destroy
- > 17-Meta-Argument-lifecycle-ignore_changes

- ✓ **Terraform Meta-Argument depends_on**
- ✓ **Provision Azure Linux VM using Terraform, file and filebase64 functions**
- ✓ **Terraform Meta-Argument count with Element Function and Splat Expression**
- ✓ **Terraform Meta-Argument for_each with Maps, Set of Strings and Chaining**
- ✓ **Meta-Argument lifecycle create_before_destroy, prevent_destroy and ignore_change**

3 lectures • 21min

4 lectures • 25min

6 lectures • 31min

6 lectures • 39min

6 lectures • 34min

Resource Meta-Arguments

depends_on

To handle **hidden resource or module** dependencies that Terraform can't automatically infer.

count

For creating **multiple** resource **instances** according to a **count**

for_each

To create **multiple instances** according to a **map**, or **set** of strings

provider

For selecting a **non-default provider** configuration

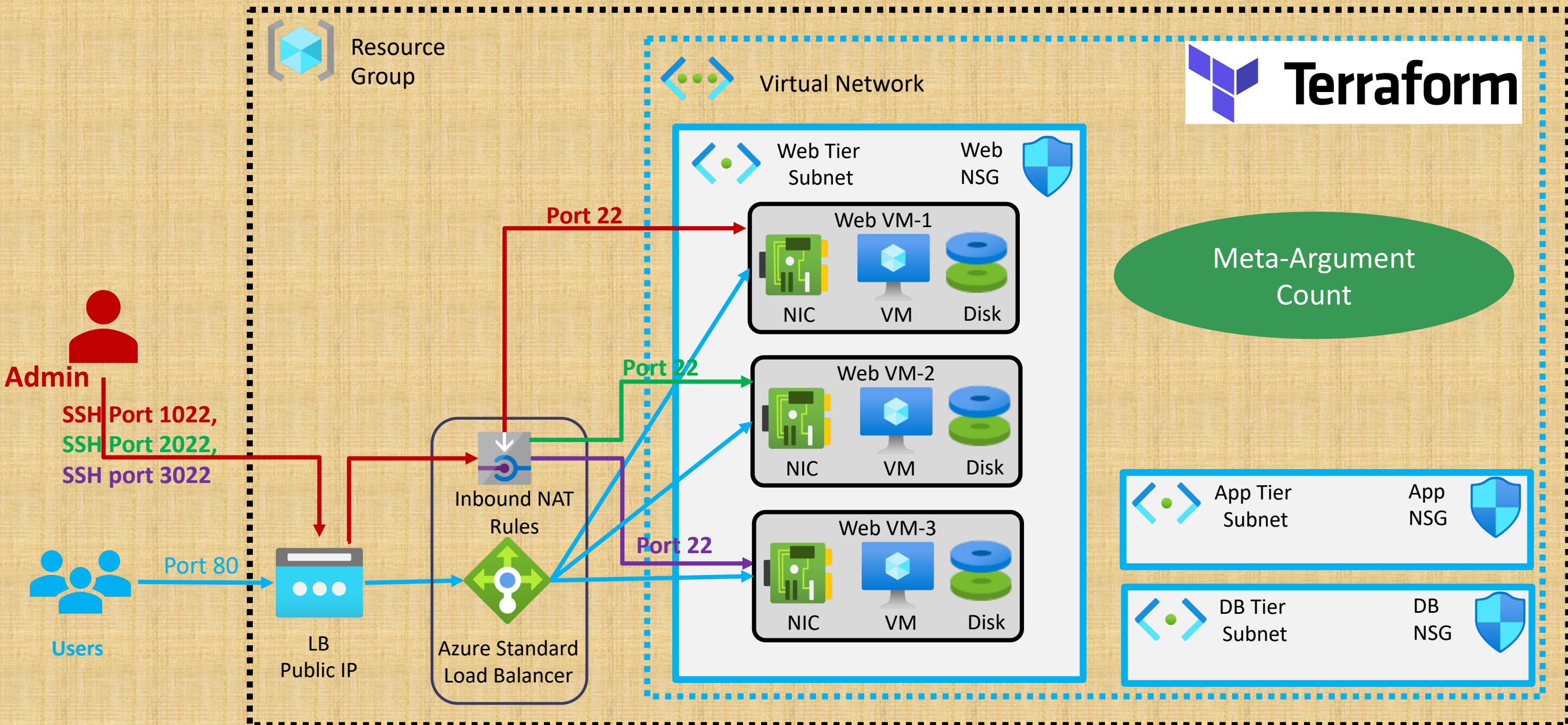
lifecycle

Standard **Resource behavior** can be altered using special nested **lifecycle block** within a resource block body

Provisioners & Connections

For taking **extra actions** after resource creation (Example: **install** some app on server or do something on **local desktop** after resource is created at **remote destination**)

Azure Standard Load Balancer – Meta-Argument Count



Meta-Argument Count – Input Variable

```
# Web Linux VM Instance Count
5 references
variable "web_linuxvm_instance_count" {
  description = "Web Linux VM Instance Count"
  type = number
  default = 1
}

# Web LB Inbound NAT Port for All VMs
variable "lb_inbound_nat_ports" {
  description = "Web LB Inbound NAT Ports List"
  type = list(string)
  default = ["1022", "2022", "3022", "4022", "5022"]
}
```

```
terraform.tfvars ×

terraform-on-azure-cloud > 16-Azure-SLB-VM-with-MetaArgument-Count > terraform-manifests > terraform.tfvars > business_division
-- 
23 web_linuxvm_instance_count = 5
24 lb_inbound_nat_ports = ["1022", "2022", "3022", "4022", "5022"]
```

Meta-Argument Count – Network Interface

```
# Resource-2: Create Network Interface
1 reference

resource "azurerm_network_interface" "web_linuxvm_nic" {
    count          = var.web_linuxvm_instance_count
    name           = "${local.resource_name_prefix}-web-linuxvm-nic-${count.index}"
    location       = azurerm_resource_group.rg.location
    resource_group_name = azurerm_resource_group.rg.name

    ip_configuration {
        name          = "web-linuxvm-ip-1"
        subnet_id     = azurerm_subnet.websubnet.id
        private_ip_address_allocation = "Dynamic"
        #public_ip_address_id = azurerm_public_ip.web_linuxvm_publicip.id
    }
}
```

Meta-Argument Count – Virtual Machine

```
# Resource: Azure Linux Virtual Machine
4 references
resource "azurerm_linux_virtual_machine" "web_linuxvm" {
    count = var.web_linuxvm_instance_count
    name = "${local.resource_name_prefix}-web-linuxvm-${count.index}"
    #computer_name = "web-linux-vm"    # Hostname of the VM (Optional)
    resource_group_name = azurerm_resource_group.rg.name
    location = azurerm_resource_group.rg.location
    size = "Standard_DS1_v2"
    admin_username = "azureuser"
    network_interface_ids = [ element(azurerm_network_interface.web_linuxvm_nic[*].id, count.index) ]
    admin_ssh_key {
        username = "azureuser"
        public_key = file("${path.module}/ssh-keys/terraform-azure.pub")
    }
}
```

Meta-Argument Count – LB Backend Address Pool

```
# Resource-6: Associate Network Interface and Standard Load Balancer
# https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/network\_interface\_backend
resource "azurerm_network_interface_backend_address_pool_association" "web_nic_lb_associate" {
    count = var.web_linuxvm_instance_count
    network_interface_id  = element(azurerm_network_interface.web_linuxvm_nic[*].id, count.index)
    ip_configuration_name = azurerm_network_interface.web_linuxvm_nic[count.index].ip_configuration[0].name
    backend_address_pool_id = azurerm_lb_backend_address_pool.web_lb_backend_address_pool.id
}
```

Meta-Argument Count – LB NAT Rule

```
# Azure LB Inbound NAT Rule
resource "azurerm_lb_nat_rule" "web_lb_inbound_nat_rule_22" {
  depends_on = [azurerm_linux_virtual_machine.web_linuxvm] # To effectively handle az
  count = var.web_linuxvm_instance_count
  name = "vm-${count.index}-ssh-${var.lb_inbound_nat_ports[count.index]}-vm-22"
  protocol = "Tcp"
  frontend_port = element(var.lb_inbound_nat_ports, count.index)
  backend_port = 22
  frontend_ip_configuration_name = azurerm_lb.web_lb.frontend_ip_configuration[0].name
  resource_group_name = azurerm_resource_group.rg.name
  loadbalancer_id = azurerm_lb.web_lb.id
}
```

Meta-Argument Count – LB NAT Rule & NIC Association

```
# Associate LB NAT Rule and VM Network Interface
resource "azurerm_network_interface_nat_rule_association" "web_nic_nat_rule_associate" {
    count = var.web_linuxvm_instance_count
    network_interface_id  = element(azurerm_network_interface.web_linuxvm_nic[*].id, count.index)
    ip_configuration_name = element(azurerm_network_interface.web_linuxvm_nic[*].ip_configuration[0].name, count.i
    nat_rule_id           = element(azurerm_lb_nat_rule.web_lb_inbound_nat_rule_22[*].id, count.index)
}
```

Terraform Configs

terraform-manifests

- > app-scripts
- > ssh-keys
- ↳ c1-versions.tf
- ↳ c2-generic-input-variables.tf
- ↳ c3-locals.tf
- ↳ c4-random-resources.tf
- ↳ c5-resource-group.tf

- ↳ c6-01-vnet-input-variables.tf
- ↳ c6-02-virtual-network.tf
- ↳ c6-03-web-subnet-and-nsg.tf
- ↳ c6-04-app-subnet-and-nsg.tf
- ↳ c6-05-db-subnet-and-nsg.tf
- ↳ c6-06-bastion-subnet-and-nsg.tf
- ↳ c6-07-vnet-outputs.tf

- ↳ c7-01-web-linuxvm-input-variables.tf
- ↳ c7-02-web-linuxvm-publicip.tf
- ↳ c7-03-web-linuxvm-network-interface.tf
- ↳ c7-04-web-linuxvm-network-security-group.tf
- ↳ c7-05-web-linuxvm-resource.tf
- ↳ c7-06-web-linuxvm-outputs.tf

- ↳ c8-01-bastion-host-input-variables.tf
- ↳ c8-02-bastion-host-linuxvm.tf
- ↳ c8-03-move-ssh-key-to-bastion-host.tf
- ↳ c8-04-AzureBastionService.tf
- ↳ c8-05-bastion-outputs.tf

Azure Virtual Network with Subnets and Network Security Groups

Azure Virtual Machines + Network Interface + Public IP

Azure Bastion Host Linux VM – Enabled
Azure Bastion Service - Disabled

Azure Standard Load Balancer – Web + Inbound NAT Rules

- ↳ c9-01-web-loadbalancer-input-variables.tf
- ↳ c9-02-web-loadbalancer-resource.tf
- ↳ c9-03-web-loadbalancer-outputs.tf
- ↳ c9-04-web-loadbalancer-inbound-nat-rules.tf

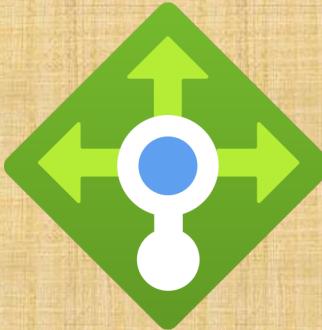
^ Terraform Meta-Argument Count with Azure LB and VMs

6 lectures • 1hr 2min

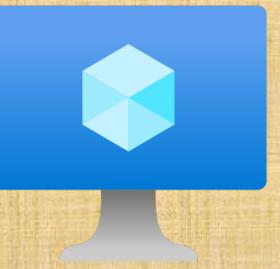
- Step-01: Introduction to implementing multiple VMs using Meta-Argument Count
- ▶ Step-02: Review changes to be performed and create variables 08:48
- ▶ Step-03: Create VM NIC with Meta-Argument Count 05:44
- ▶ Step-04: Create Linux VM with Meta-Argument Count with Element Function and Spla 11:45
- ▶ Step-05: Create VM Outputs with Splat Expr and LBNICAssociate with Count Meta-Ar 08:26
- ▶ Step-06: Craeate LB Inbound NAT Rules with Meta-Argument Count 11:25
- ▶ Step-07: Execute TF Commands, Verify Resources and Destroy Resources 15:34

Time it takes to complete this Demo

Real-World Demo



Load Balancer

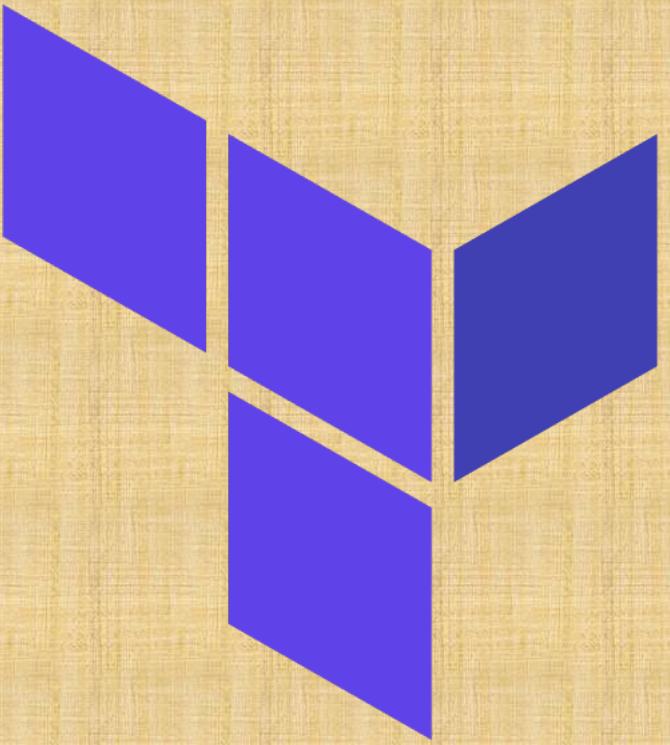


Virtual Machines

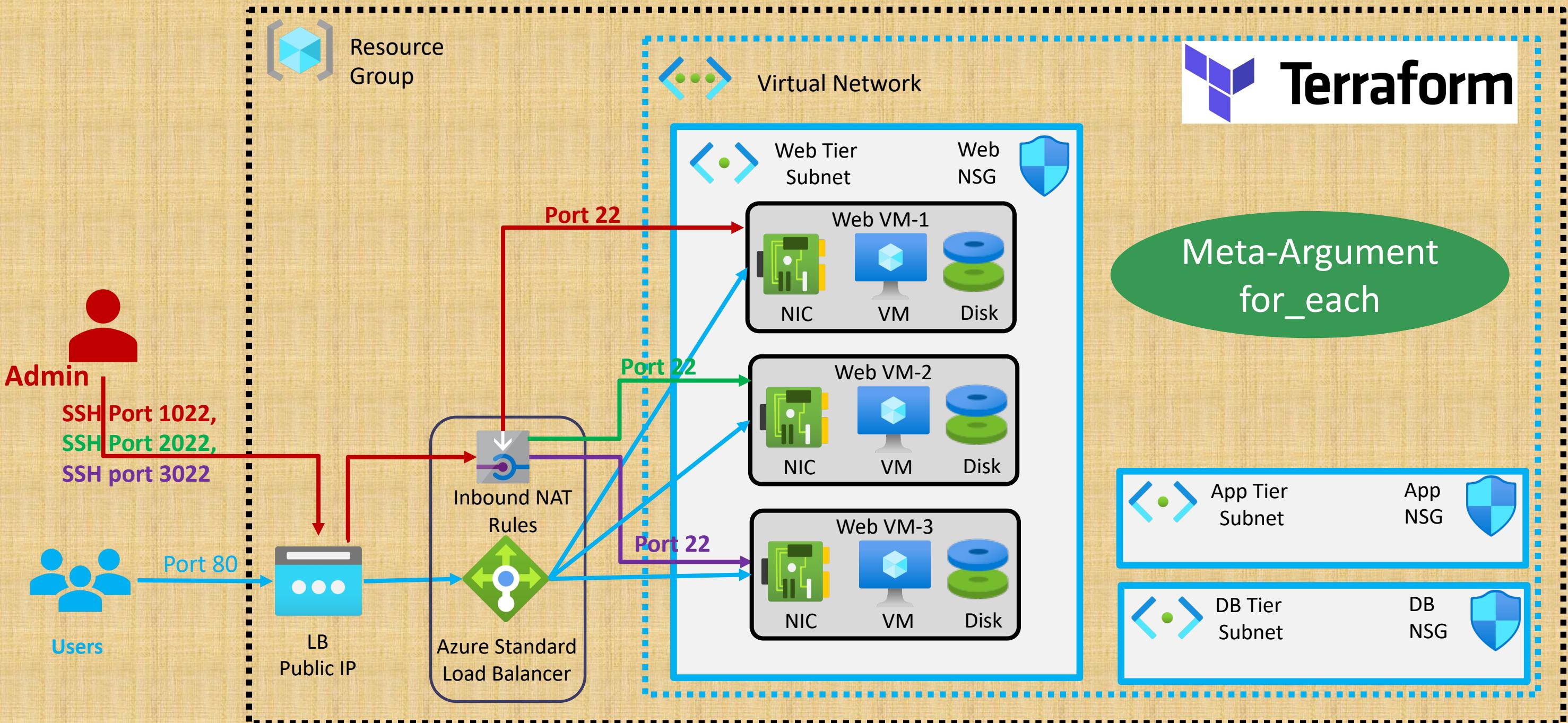


Real-World
Demo

Azure
Multiple VMs
Meta-Argument for_each



Azure Standard Load Balancer – Meta-Argument for_each



Meta-Argument for_each – Input Variable

```
# Linux VM Input Variables Placeholder file.  
# Web Linux VM Instance Count  
5 references  
variable "web_linuxvm_instance_count" {  
  description = "Web Linux VM Instance Count"  
  type = map(string)  
  default = {  
    "vm1" = "1022",  
    "vm2" = "2022"  
  }  
}
```

```
terraform.tfvars ×  
terraform-on-azure-cloud > 17-Azure-SLB-VM-with-for-each-and-for-loops  
23   web_linuxvm_instance_count = {  
24     "vm1" = "1022",  
25     "vm2" = "2022"  
26   }
```

Meta-Argument for `_each` – Network Interface

```
# Resource-2: Create Network Interface
2 references
resource "azurerm_network_interface" "web_linuxvm_nic" {
  for_each = var.web_linuxvm_instance_count
  name      = "${local.resource_name_prefix}-web-linuxvm-nic-${each.key}"
  location   = azurerm_resource_group.rg.location
  resource_group_name = azurerm_resource_group.rg.name

  ip_configuration {
    name          = "web-linuxvm-ip-1"
    subnet_id     = azurerm_subnet.websubnet.id
    private_ip_address_allocation = "Dynamic"
    #public_ip_address_id = azurerm_public_ip.web_linuxvm_publicip.id
  }
}
```

Meta-Argument for_each – Virtual Machine

```
# Resource: Azure Linux Virtual Machine
```

5 references

```
resource "azurerm_linux_virtual_machine" "web_linuxvm" {
    for_each = var.web_linuxvm_instance_count
    name = "${local.resource_name_prefix}-web-linuxvm-${each.key}"
    #computer_name = "web-linux-vm"  # Hostname of the VM (Optional)
    resource_group_name = azurerm_resource_group.rg.name
    location = azurerm_resource_group.rg.location
    size = "Standard_DS1_v2"
    admin_username = "azureuser"
    network_interface_ids = [ azurerm_network_interface.web_linuxvm_nic[each.key].id ]
    admin_ssh_key {
        username = "azureuser"
        public_key = file("${path.module}/ssh-keys/terraform-azure.pub")
    }
}
```

Meta-Argument for_each – LB Backend Pool

```
# Resource-6: Associate Network Interface and Standard Load Balancer
# https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/network_interface_backend_pool_association
resource "azurerm_network_interface_backend_address_pool_association" "web_nic_lb_associate" {
    for_each = var.web_linuxvm_instance_count
    network_interface_id      = azurerm_network_interface.web_linuxvm_nic[each.key].id
    ip_configuration_name     = azurerm_network_interface.web_linuxvm_nic[each.key].ip_configuration[0].name
    backend_address_pool_id   = azurerm_lb_backend_address_pool.web_lb_backend_address_pool.id
}
```

Meta-Argument for_each – Inbound NAT Rules

```
# Azure LB Inbound NAT Rule
resource "azurerm_lb_nat_rule" "web_lb_inbound_nat_rule_22" {
    depends_on = [azurerm_linux_virtual_machine.web_linuxvm] # To effectively handle azurerm provider related errors
    for_each = var.web_linuxvm_instance_count
    name = "${each.key}-ssh-${each.value}-vm-22"
    protocol = "Tcp"
    frontend_port = each.value
    #frontend_port = lookup(var.web_linuxvm_instance_count, each.key)
    backend_port = 22
    frontend_ip_configuration_name = azurerm_lb.web_lb.frontend_ip_configuration[0].name
    resource_group_name = azurerm_resource_group.rg.name
    loadbalancer_id = azurerm_lb.web_lb.id
}

# Associate LB NAT Rule and VM Network Interface
resource "azurerm_network_interface_nat_rule_association" "web_nic_nat_rule_associate" {
    for_each = var.web_linuxvm_instance_count
    network_interface_id = azurerm_network_interface.web_linuxvm_nic[each.key].id
    ip_configuration_name = azurerm_network_interface.web_linuxvm_nic[each.key].ip_configuration[0].name
    nat_rule_id = azurerm_lb_nat_rule.web_lb_inbound_nat_rule_22[each.key].id
}
```

Terraform – For Loops using Outputs

```
# Output List - Single Input to for loop
output "web_linuxvm_private_ip_address_list" {
  description = "Web Linux Virtual Machine Private IP"
  #value = azurerm linux virtual machine.web_linuxvm.private_ip_address
  value = [for vm in azurerm_linux_virtual_machine.web_linuxvm: vm.private_ip_address]
}

# Output Map - Single Input to for loop
output "web_linuxvm_private_ip_address_map" {
  description = "Web Linux Virtual Machine Private IP"
  #value = azurerm linux virtual machine.web_linuxvm.private_ip_address
  value = {for vm in azurerm_linux_virtual_machine.web_linuxvm: vm.name => vm.private_ip_address}
}

# Terraform keys() function: keys takes a map and returns a list containing the keys from that map.
output "web_linuxvm_private_ip_address_keys_function" {
  description = "Web Linux Virtual Machine Private IP"
  value = keys({for vm in azurerm_linux_virtual_machine.web_linuxvm: vm.name => vm.private_ip_address})
}
```

Terraform – For Loops using Outputs

```
# Terraform values() function: values takes a map and returns a list containing the values of the elements
output "web_linuxvm_private_ip_address_values_function" {
  description = "Web Linux Virtual Machine Private IP"
  value = values({for vm in azurerm_linux_virtual_machine.web_linuxvm: vm.name => vm.private_ip_address})
}

# Output List - Two Inputs to for loop (here vm is Iterator like "i")
output "web_linuxvm_network_interface_id_list" {
  description = "Web Linux VM Network Interface ID"
  #value = azurerm_network_interface.web_linuxvm_nic.id
  value = [for vm, nic in azurerm_network_interface.web_linuxvm_nic: nic.id ]
}

# Output Map - Two Inputs to for loop (here vm is Iterator like "i")
output "web_linuxvm_network_interface_id_map" {
  description = "Web Linux VM Network Interface ID"
  #value = azurerm_network_interface.web_linuxvm_nic.id
  value = {for vm, nic in azurerm_network_interface.web_linuxvm_nic: vm => nic.id }
}
```

Terraform Configs

terraform-manifests

- > app-scripts
- > ssh-keys
- ↳ c1-versions.tf
- ↳ c2-generic-input-variables.tf
- ↳ c3-locals.tf
- ↳ c4-random-resources.tf
- ↳ c5-resource-group.tf

- ↳ c6-01-vnet-input-variables.tf
- ↳ c6-02-virtual-network.tf
- ↳ c6-03-web-subnet-and-nsg.tf
- ↳ c6-04-app-subnet-and-nsg.tf
- ↳ c6-05-db-subnet-and-nsg.tf
- ↳ c6-06-bastion-subnet-and-nsg.tf
- ↳ c6-07-vnet-outputs.tf

- ↳ c7-01-web-linuxvm-input-variables.tf
- ↳ c7-02-web-linuxvm-publicip.tf
- ↳ c7-03-web-linuxvm-network-interface.tf
- ↳ c7-04-web-linuxvm-network-security-group.tf
- ↳ c7-05-web-linuxvm-resource.tf
- ↳ c7-06-web-linuxvm-outputs.tf

- ↳ c8-01-bastion-host-input-variables.tf
- ↳ c8-02-bastion-host-linuxvm.tf
- ↳ c8-03-move-ssh-key-to-bastion-host.tf
- ↳ c8-04-AzureBastionService.tf
- ↳ c8-05-bastion-outputs.tf

Azure Virtual Network with Subnets and Network Security Groups

Azure Virtual Machines + Network Interface + Public IP

Azure Bastion Host Linux VM – Enabled
Azure Bastion Service - Disabled

Azure Standard Load Balancer – Web + Inbound NAT Rules

- ↳ c9-01-web-loadbalancer-input-variables.tf
- ↳ c9-02-web-loadbalancer-resource.tf
- ↳ c9-03-web-loadbalancer-outputs.tf
- ↳ c9-04-web-loadbalancer-inbound-nat-rules.tf

^ Terraform Meta-Argument for_each with Azure LB and VMs

4 lectures • 40min

- Step-01: Introduction to implementing multiple VMs using Meta-Argument for_each
- Step-02: Create Variables, VM NIC and Linux VM with Meta-Argument for_each 09:17
- Step-03: Create Outputs with for loops 12:20
- Step-04: Create LB Backend Associate with VMNIC, LB NAT Rules with MA for_each 09:07
- Step-05: Execute TF Commands, Verify Outputs, Verify and Destroy Resources 09:37

Time it takes to complete this Demo

Real-World Demo



Load Balancer

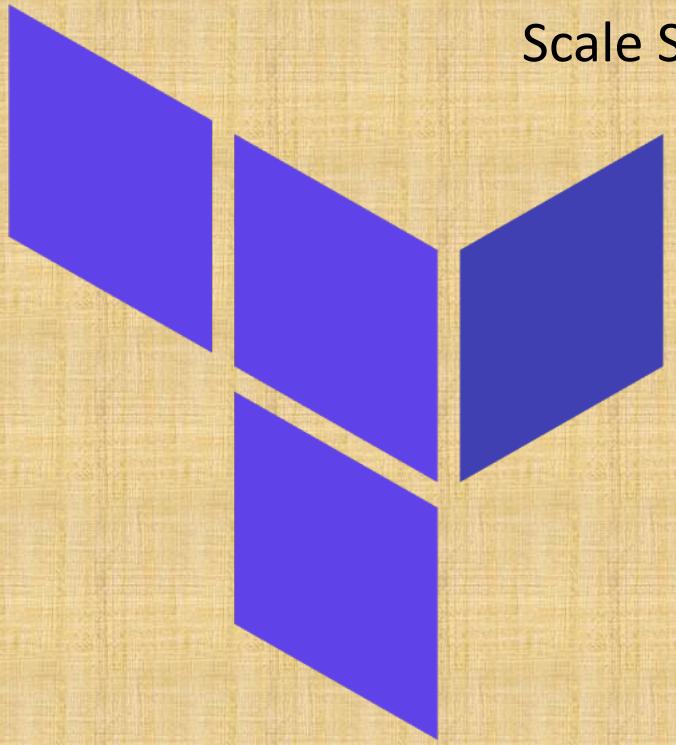


Azure Virtual Machine Scale Sets

Manual Scaling External LB

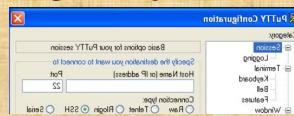


VM
Scale Sets



Azure Standard Load Balancer – VMSS Manual Scaling

SSH Client – Putty / Terminal



Users

Port 80

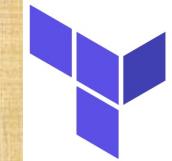
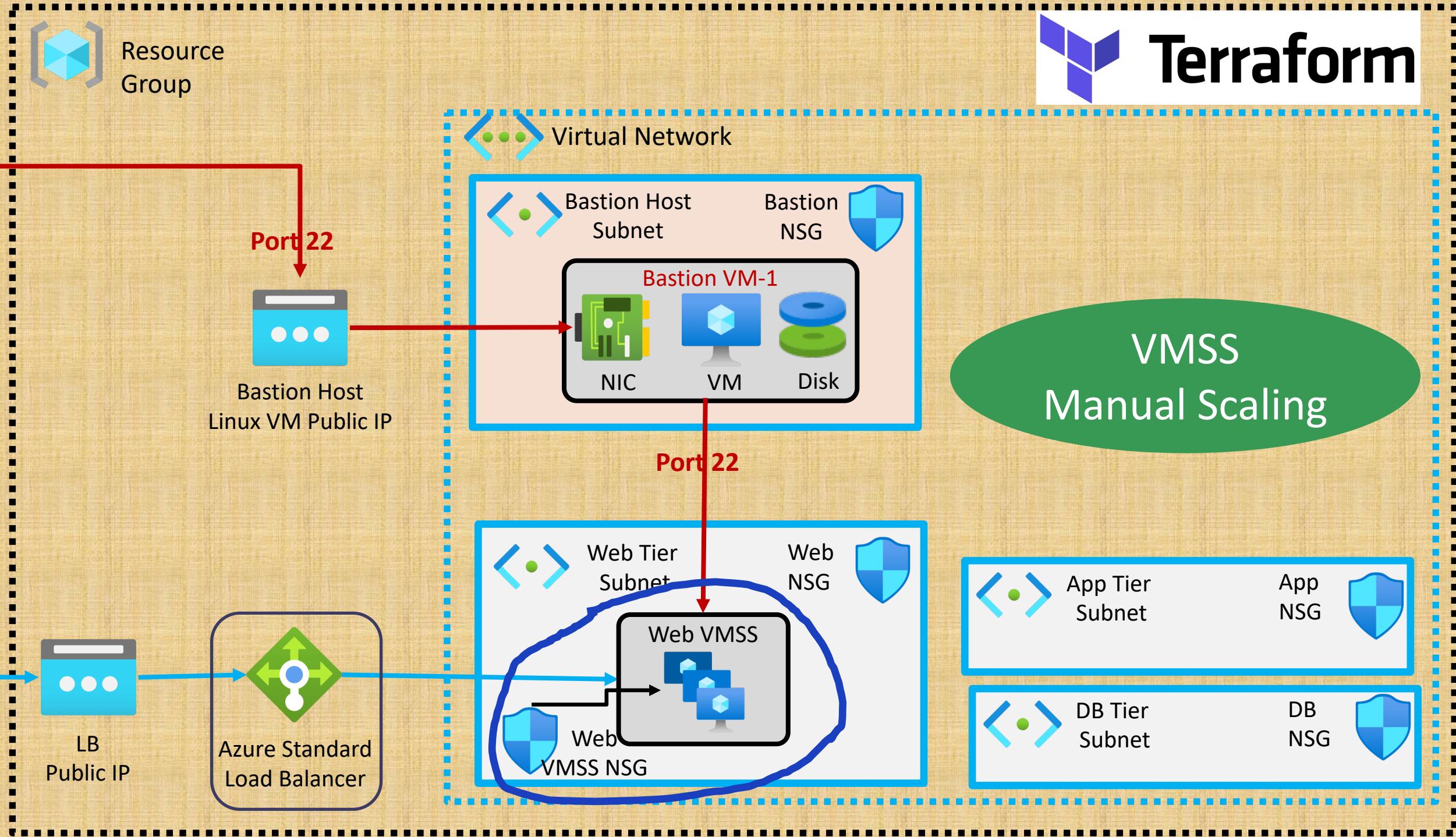
LB Public IP

Azure Standard Load Balancer

Port 22

Bastion Host Linux VM Public IP

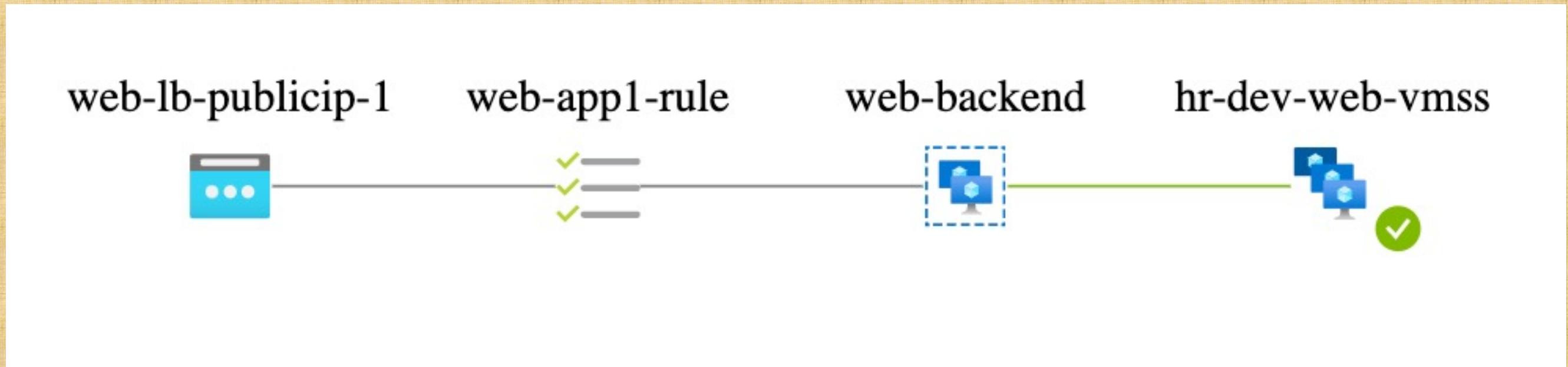
Resource Group



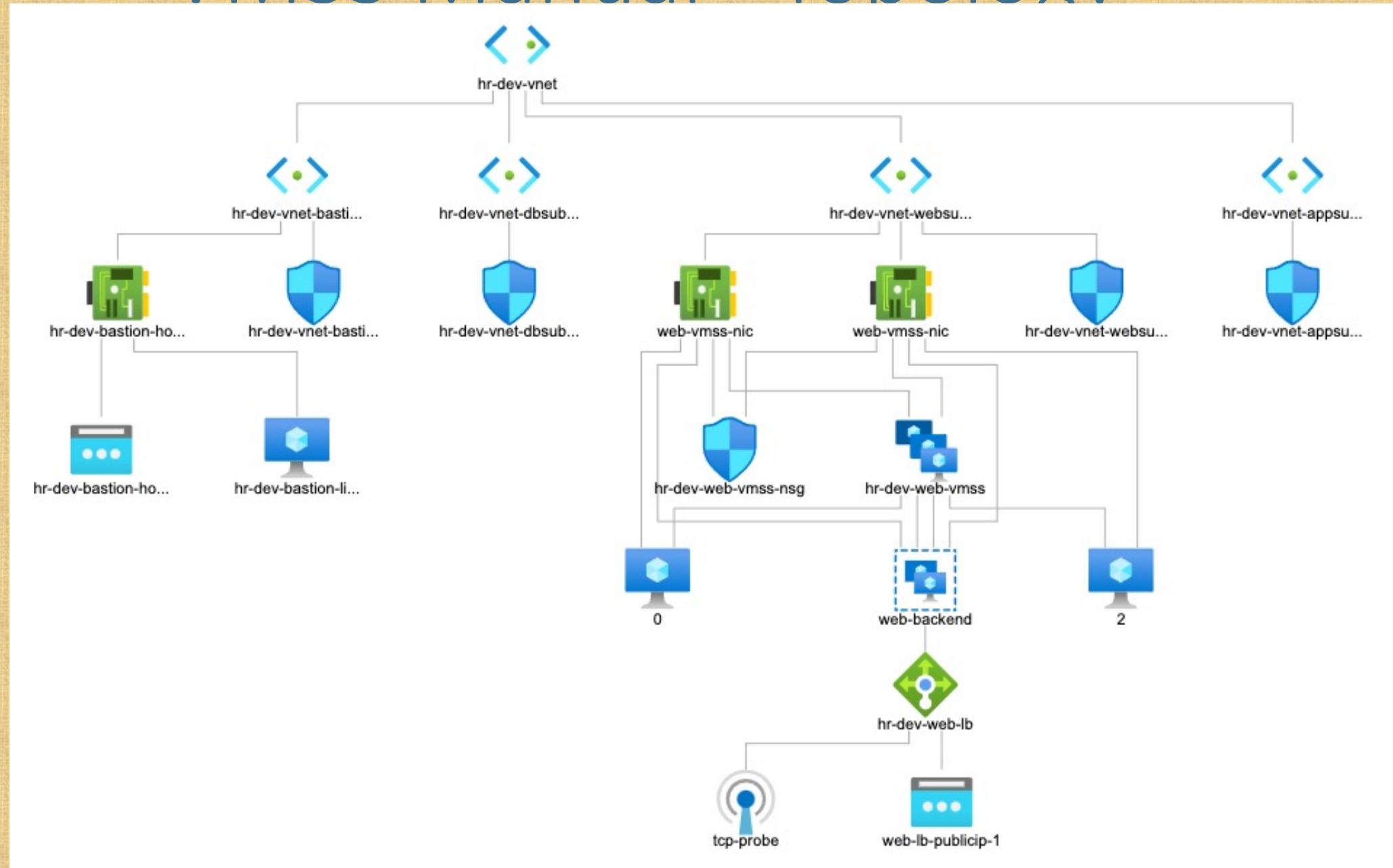
Terraform

VMSS
Manual Scaling

LB Topology



VMSS Manual - Topology



Terraform Dynamic Blocks for VMSS Network Security Group

```
# Linux VM Input Variables Placeholder file.  
variable "web_vmss_nsg_inbound_ports" {  
    description = "Web VMSS NSG Inbound Ports"  
    type = list(string)  
    default = [22, 80, 443]  
}
```

```
# Create Network Security Group using Terraform Dynamic Blocks  
1 reference  
resource "azurerm_network_security_group" "web_vmss_nsg" {  
    name          = "${local.resource_name_prefix}-web-vmss-nsg"  
    location      = azurerm_resource_group.rg.location  
    resource_group_name = azurerm_resource_group.rg.name  
  
    dynamic "security_rule" {  
        for_each = var.web_vmss_nsg_inbound_ports  
        content {  
            name          = "inbound-rule-${security_rule.key}"  
            description   = "Inbound Rule ${security_rule.key}"  
            priority      = sum([100, security_rule.key])  
            direction     = "Inbound"  
            access         = "Allow"  
            protocol      = "Tcp"  
            source_port_range = "*"  
            destination_port_range = security_rule.value  
            source_address_prefix = "*"  
            destination_address_prefix = "*"  
        }  
    }  
}
```

VMSS Resource – Associate Load Balancer

```
network_interface {  
    name = "web-vmss-nic"  
    primary = "true"  
    network_security_group_id = azurerm_network_security_group.web_vmss_nsg.id  
    ip_configuration {  
        name = "internal"  
        primary = true  
        subnet_id = azurerm_subnet.websubnet.id  
        load_balancer_backend_address_pool_ids = [azurerm_lb_backend_address_pool.web_lb_backend_address_pool.id]  
    }  
}
```

```
/*  
# Resource-6: Associate Network Interface and Standard Load Balancer  
# https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/network\_interface  
resource "azurerm_network_interface_backend_address_pool_association" "web_nic_lb_associate" {  
    network_interface_id      = azurerm_network_interface.web_linuxvm_nic.id  
    ip_configuration_name    = azurerm_network_interface.web_linuxvm_nic.ip_configuration[0].name  
    backend_address_pool_id = azurerm_lb_backend_address_pool.web_lb_backend_address_pool.id  
}  
*/
```

Not needed
when using
VMSS

Terraform Configs

```
✓ terraform-manifests
  > app-scripts
  > ssh-keys
  ✓ c1-versions.tf
  ✓ c2-generic-input-variables.tf
  ✓ c3-locals.tf
  ✓ c4-random-resources.tf
  ✓ c5-resource-group.tf
  ✓ c6-01-vnet-input-variables.tf
  ✓ c6-02-virtual-network.tf
  ✓ c6-03-web-subnet-and-nsg.tf
  ✓ c6-04-app-subnet-and-nsg.tf
  ✓ c6-05-db-subnet-and-nsg.tf
  ✓ c6-06-bastion-subnet-and-nsg.tf
  ✓ c6-07-vnet-outputs.tf
  ✓ c7-01-web-linux-vmss-input-variables.tf
  ✓ c7-02-web-linux-vmss-nsg-inline-basic.tf
  ✓ c7-03-web-linux-vmss-resource.tf
  ✓ c7-04-web-linux-vmss-autoscaling-cpu-usage.tf
  ✓ c7-05-web-linux-vmss-outputs.tf
  ✓ c8-01-bastion-host-input-variables.tf
  ✓ c8-02-bastion-host-linuxvm.tf
  ✓ c8-03-move-ssh-key-to-bastion-host.tf
  ✓ c8-04-AzureBastionService.tf
  ✓ c8-05-bastion-outputs.tf
```

Azure Virtual Network with Subnets and Network Security Groups

Azure Virtual Machine Scale Sets with Manual Scaling + Terraform Dynamic Blocks for VMSS NSG

Azure Bastion Host Linux VM – Enabled
Azure Bastion Service - Disabled

Azure Standard Load Balancer – Web + Inbound NAT Rules

c9-01-web-loadbalancer-input-variables.tf
c9-02-web-loadbalancer-resource.tf
c9-03-web-loadbalancer-outputs.tf
terrafrom.tfvars

^ Azure Virtual Machine Scale Sets Basics - Manual Scaling

4 lectures • 56min

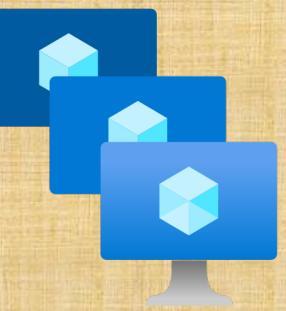
- Step-01: Introduction to implementing VMSS using Terraform
- ▶ Step-02: Review Variables and VMSS NSG Basic Rule via Inline Security Rule Neste 08:52
- ▶ Step-03: Create VMSS NSG with Terraform Dynamic Block and Inline Security Rules 12:39
- ▶ Step-04: Create Web VMSS Resource and its outputs 14:04
- ▶ Step-05: Execute TF Commands, Verify Resources and Destroy VMSS 20:41

Time it takes to complete this Demo

Real-World Demo



Load Balancer

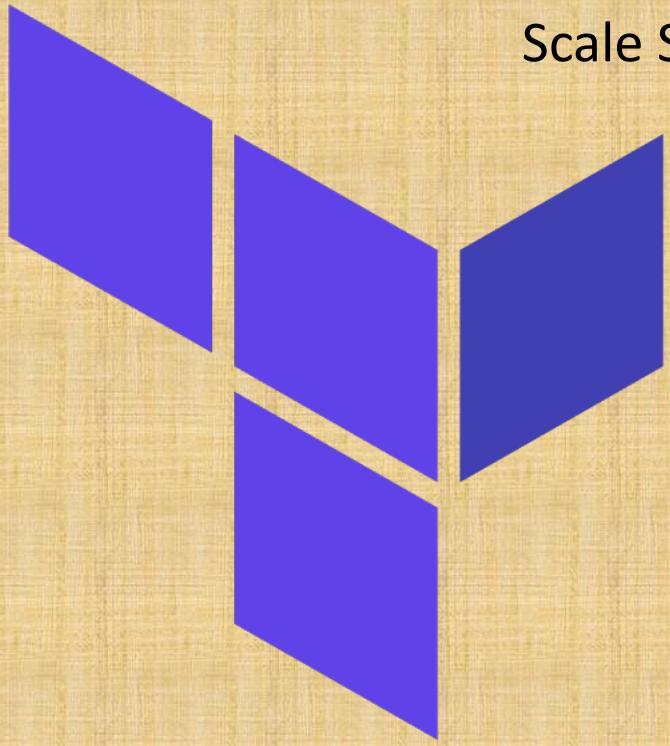


VM
Scale Sets



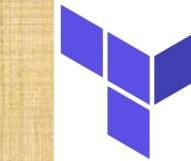
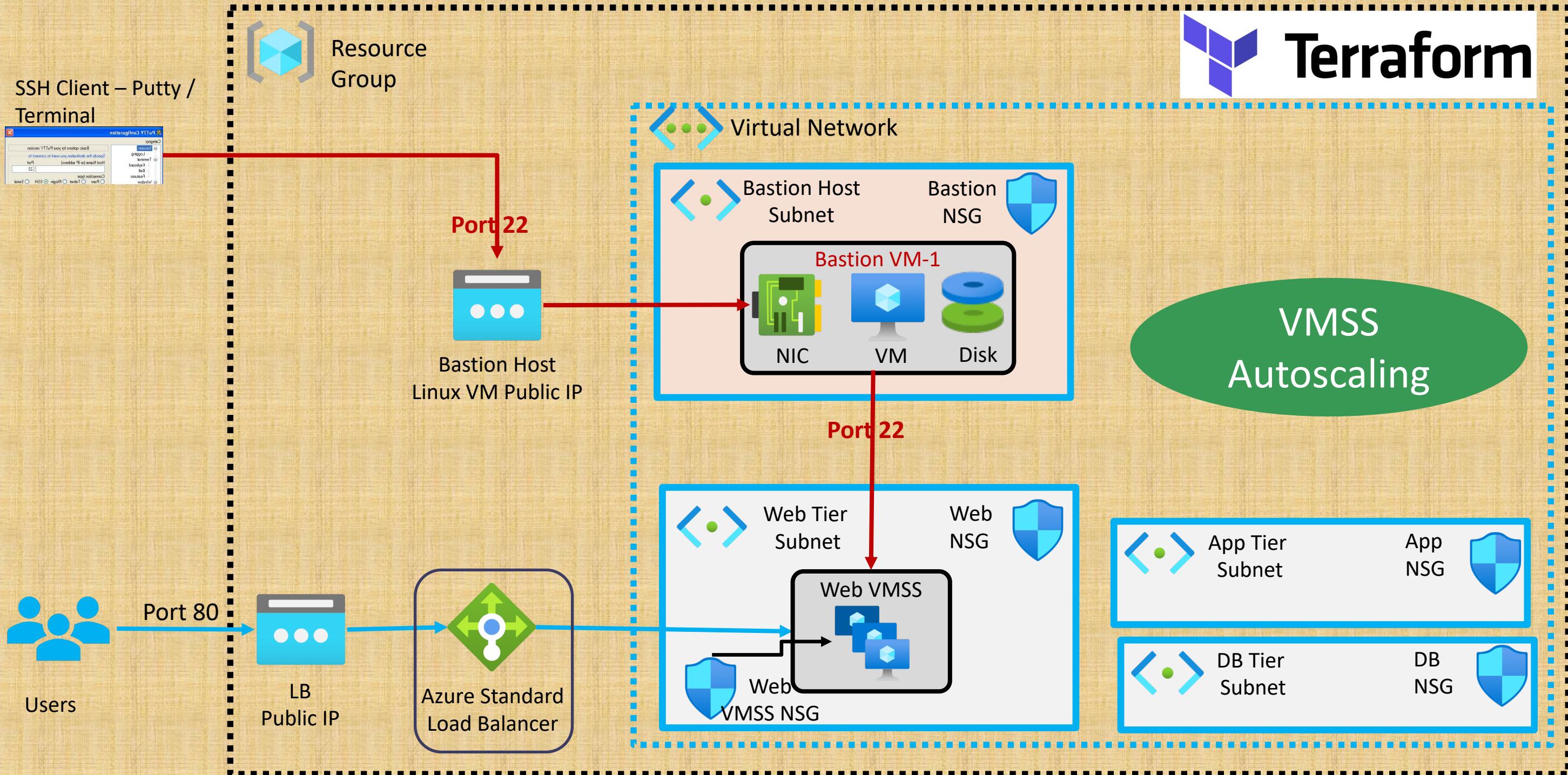
Real-World
Demo

Azure Virtual Machine Scale Sets Autoscaling



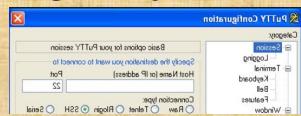
External LB

Azure Standard Load Balancer + VMSS Auto Scaling



Terraform

SSH Client – Putty / Terminal



Users

Port 80

LB Public IP

Azure Standard Load Balancer

Port 22

Bastion Host Linux VM Public IP

Virtual Network

Bastion Host Subnet

Bastion NSG

Bastion VM-1

- NIC
- VM
- Disk

Web Tier Subnet

Web NSG

Web VMSS

- Web VMSS NSG

VMSS Autoscaling

App Tier Subnet

App NSG

DB Tier Subnet

DB NSG

Azure VMSS - Autoscaling

Autoscaling Default Profile

Mandatory Profile

Defaults to round the clock schedule

Will not execute if Recurrence or Fixed Profile exists

P3

Autoscaling Recurrence Profile

Recur on those days with start time specified

Week Day and Weekend profiles

Business Hour and Non-Business Hour profile

P2

Autoscaling Fixed Profiles

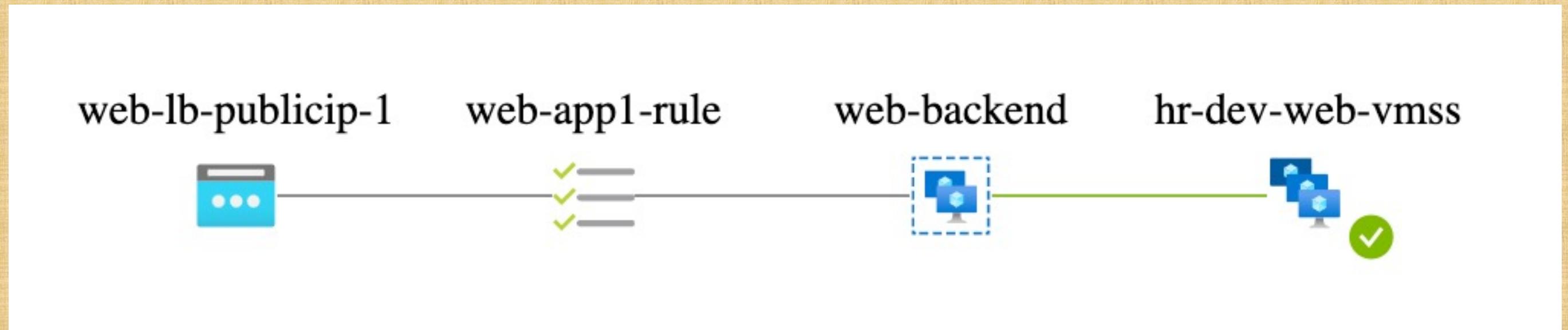
Executes on that specific day

Fixed profile takes priority 1 for execution on that day if exists

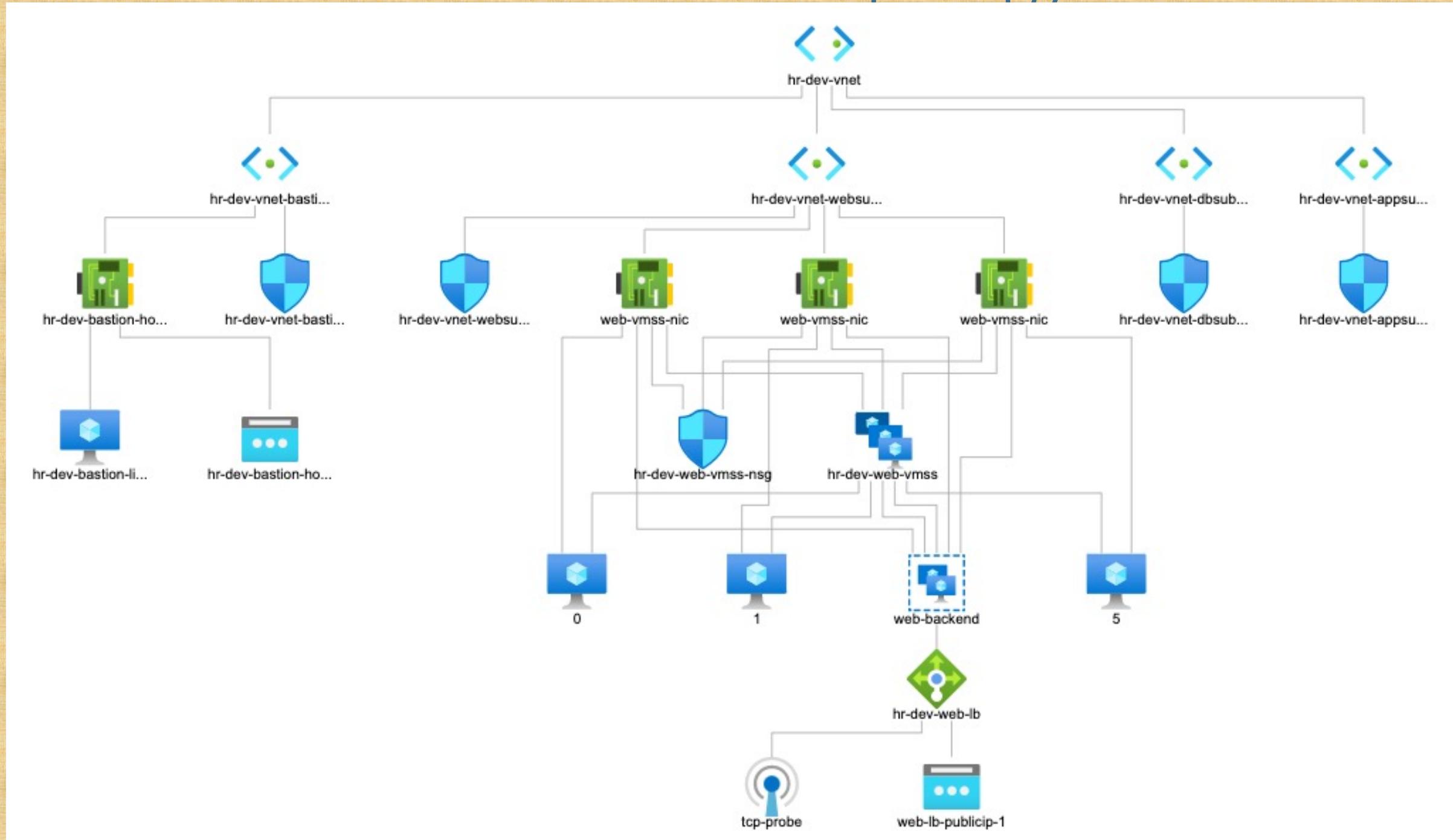
P1

Priority Execution Order for Autoscaling Profiles

LB Topology



VMSS Manual - Topology



Autoscaling – Default Profile

The screenshot shows the Azure portal interface for managing a Virtual Machine Scale Set (VMSS). The top navigation bar includes 'Home > Virtual machine scale sets > hr-dev-web-vmss'. The main title is 'hr-dev-web-vmss | Scaling'. Below the title, there's a search bar and standard navigation buttons: Save, Discard, Refresh, Logs, and Feedback.

The left sidebar contains several navigation items under 'hr-dev-web-vmss': Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Instances, Networking, Scaling (which is selected), Disks, Operating system, Security, Guest + host updates, and Size.

The main content area has a tab bar with 'Configure' (selected), Scale-In Policy, Run history, JSON, Notify, and Diagnostic settings. A descriptive text block explains what Autoscale does: "Autoscale is a built-in feature that helps applications perform their best when demand changes. You can choose to scale your resource manually to a specific instance count, or via a custom Autoscale policy that scales based on metric(s) thresholds, or schedule instance count which scales during designated time windows. Autoscale enables your resource to be performant and cost effective by adding and removing instances based on demand." It links to "Learn more about Azure Autoscale" and "view the how-to video".

The 'Choose how to scale your resource' section offers two options: 'Manual scale' (with a green icon) and 'Custom autoscale' (with a blue icon). The 'Custom autoscale' option is highlighted with a red box. Below this, the 'Custom autoscale' section displays the following settings:

Autoscale setting name	hr-dev-web-vmss-autoscale-profiles
Resource group	hr-dev-rg-nfrxug
Instance count	3

Autoscaling Default Profile

Default* default 

 Delete warning  The very last or default recurrence rule cannot be deleted. Instead, you can disable autoscale to turn off autoscale.

Scale mode Scale based on a metric Scale to a specific instance count

Rules

Scale out

When	hr-dev-web-vmss	(Average) Percentage CPU > 75	Increase count by 1
Or	hr-dev-web-vmss	(Average) Available Memory By...	Increase count by 1
Or	hr-dev-web-lb	(Average) SYNCCount > 10	Increase count by 1

Scale in

When	hr-dev-web-vmss	(Average) Percentage CPU < 25	Decrease count by 1
And	hr-dev-web-vmss	(Average) Available Memory By...	Decrease count by 1
And	hr-dev-web-lb	(Average) SYNCCount < 10	Decrease count by 1

+ Add a rule

Instance limits

Minimum 	Maximum 	Default 
2 	6 	2 

Schedule  This scale condition is executed when none of the other scale condition(s) match

Scale Out

Scale In

Default Profile

Autoscaling – What happens to default profile when other profiles present ?

Default* default 



Scale mode Scale based on a metric Scale to a specific instance count

Rules

Scale out

When	hr-dev-web-vmss	(Average) Percentage CPU > 75	Increase count by 1
Or	hr-dev-web-vmss	(Average) Available Memory By...	Increase count by 1
Or	hr-dev-web-lb	(Average) SYNCount > 10	Increase count by 1

Scale in

When	hr-dev-web-vmss	(Average) Percentage CPU < 25	Decrease count by 1
And	hr-dev-web-vmss	(Average) Available Memory By...	Decrease count by 1
And	hr-dev-web-lb	(Average) SYNCount < 10	Decrease count by 1

+ Add a rule

Instance limits

Minimum 	2 	Maximum 	6 	Default 	2 
---	---	---	---	---	---

Schedule

This scale condition is not running because you have other scale condition(s) without an end time.

Autoscaling Recurrence Profile Week Days

profile-2-weekdays 



Scale mode Scale based on a metric Scale to a specific instance count

Rules

Scale out

When	hr-dev-web-vmss	(Average) Percentage CPU > 75	Increase count by 1
Or	hr-dev-web-vmss	(Average) Available Memory By...	Increase count by 1
Or	hr-dev-web-lb	(Average) SYNCCount > 10	Increase count by 1

Scale in

When	hr-dev-web-vmss	(Average) Percentage CPU < 25	Decrease count by 1
And	hr-dev-web-vmss	(Average) Available Memory By...	Decrease count by 1
And	hr-dev-web-lb	(Average) SYNCCount < 10	Decrease count by 1

+ Add a rule

Instance limits

Minimum 	4	
Maximum 	20	
Default 	4	

Schedule

Specify start/end dates Repeat specific days

Repeat every

Monday Tuesday Wednesday Thursday
Friday Saturday Sunday

Timezone

Start time

End time

Specify an end time, else this scale condition will apply for all days until it reaches the start time of another scale condition

Recurrence Week Days



Scale mode

 Scale based on a metric Scale to a specific instance count

Rules

Scale out

When	hr-dev-web-vmss	(Average) Percentage CPU > 75	Increase count by 1
Or	hr-dev-web-vmss	(Average) Available Memory By...	Increase count by 1
Or	hr-dev-web-lb	(Average) SYNCCount > 10	Increase count by 1

Scale in

When	hr-dev-web-vmss	(Average) Percentage CPU < 25	Decrease count by 1
And	hr-dev-web-vmss	(Average) Available Memory By...	Decrease count by 1
And	hr-dev-web-lb	(Average) SYNCCount < 10	Decrease count by 1

[+ Add a rule](#)

Instance limits

Minimum 3 Maximum 6 Default 3 

Schedule

Repeat every

Timezone

Start time

End time

 Specify start/end dates Repeat specific days

- Monday Tuesday Wednesday Thursday
 Friday Saturday Sunday

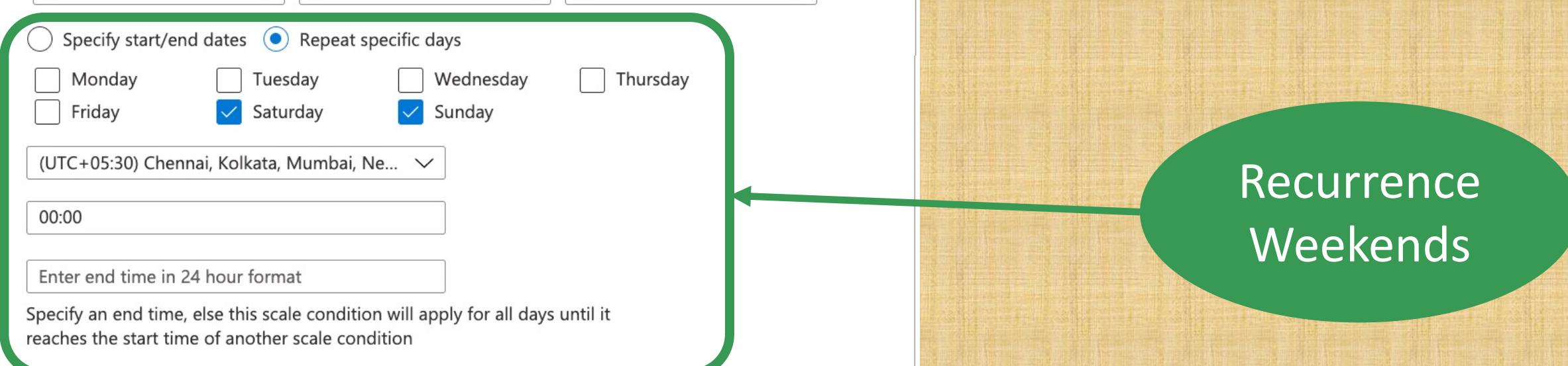
(UTC+05:30) Chennai, Kolkata, Mumbai, Ne...

00:00

Enter end time in 24 hour format

Specify an end time, else this scale condition will apply for all days until it reaches the start time of another scale condition

Autoscaling Recurrence Profile Weekends



Recurrence Weekends

Autoscaling Fixed Profile

profile-4-fixed-profile 



Scale mode Scale based on a metric Scale to a specific instance count

Rules

Scale out

When	hr-dev-web-vmss	(Average) Percentage CPU > 75	Increase count by 1
Or	hr-dev-web-vmss	(Average) Available Memory By...	Increase count by 1
Or	hr-dev-web-lb	(Average) SYNCCount > 10	Increase count by 1

Scale in

When	hr-dev-web-vmss	(Average) Percentage CPU < 25	Decrease count by 1
And	hr-dev-web-vmss	(Average) Available Memory By...	Decrease count by 1
And	hr-dev-web-lb	(Average) SYNCCount < 10	Decrease count by 1

+ Add a rule

Instance limits

Minimum 	40 	Default 	6 
---	--	---	---

Schedule

Timezone

Start date

End date

Specify start/end dates Repeat specific days

(UTC+05:30) Chennai, Kolkata, Mumbai, Ne...

08/15/2021 12:00:00 AM

08/15/2021 11:59:59 PM

Fixed Date
Profile

Autoscaling Run History

Scaling

...

X

Save Discard Refresh Logs Feedback

Configure Scale-In Policy Run history JSON Notify Diagnostic settings

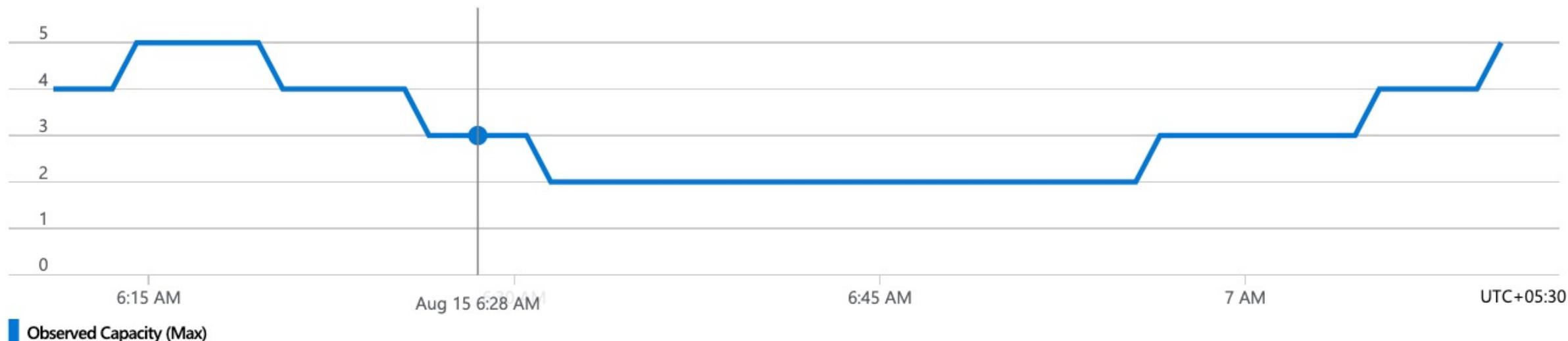
i Observed instance count - this chart plots the instance count as observed by the auto scale engine. If the chart is empty it either means auto scale is in cool down period or auto scale was disabled over a period of time or auto scale was not configured.

Show data for last

1 hour 6 hours 12 hours 1 day 7 days Custom

Pin to dashboard

Observed resource instance count



Observed Capacity (Max)
hr-dev-web-vmss-autoscale-profiles

3

Autoscaling Events – Activity Log

Home > Virtual machine scale sets > hr-dev-web-vmss >

Activity log



Activity Edit columns Refresh Diagnostics settings Download as CSV Logs | Pin current filters Reset filters

Search

Quick Insights

Subscription : StackSimplify-Paid-Subscription

Event severity : All

Timespan : Last 1 hour

Resource group : hr-dev-rg-nfrxug X

Resource : hr-dev-web-vmss X

Event category : Autoscale X

+ Add Filter

7 items.

Operation name	Status	Time	Time stamp	Subscription	Event initiated by
> ScaleupResult	Succeeded	3 minutes a...	Sun Aug 15 ...	StackSimplify-Paid-Subscription	Microsoft.Insights/autosc...
> Autoscale scale up completed	Succeeded	8 minutes a...	Sun Aug 15 ...	StackSimplify-Paid-Subscription	Microsoft.Insights/autosc...
> Autoscale scale up completed	Succeeded	17 minutes ...	Sun Aug 15 ...	StackSimplify-Paid-Subscription	Microsoft.Insights/autosc...
> Autoscale scale down completed	Succeeded	42 minutes ...	Sun Aug 15 ...	StackSimplify-Paid-Subscription	Microsoft.Insights/autosc...
> Autoscale scale down completed	Succeeded	48 minutes ...	Sun Aug 15 ...	StackSimplify-Paid-Subscription	Microsoft.Insights/autosc...
Autoscale scale down completed	Succeeded	53 minutes ...	Sun Aug 15 ...	StackSimplify-Paid-Subscription	Microsoft.Insights/autosc...
> Autoscale scale up completed	Succeeded	59 minutes ...	Sun Aug 15 ...	StackSimplify-Paid-Subscription	Microsoft.Insights/autosc...

Azure Autoscaling Resource – VMSS, App Services

```
resource "azurerm_monitor_autoscale_setting" "web_vmss_autoscale" {
    name          = "${local.resource_name_prefix}-web-vmss-autoscale-profile"
    resource_group_name = azurerm_resource_group.rg.name
    location      = azurerm_resource_group.rg.location
    target_resource_id = azurerm_linux_virtual_machine_scale_set.web_vmss.id
```

Terraform Configs

```
✓ terraform-manifests-autoscaling
  > app-scripts
  > ssh-keys
  ✓ c1-versions.tf
  ✓ c2-generic-input-variables.tf
  ✓ c3-locals.tf
  ✓ c4-random-resources.tf
  ✓ c5-resource-group.tf
  ✓ c6-01-vnet-input-variables.tf
  ✓ c6-02-virtual-network.tf
  ✓ c6-03-web-subnet-and-nsg.tf
  ✓ c6-04-app-subnet-and-nsg.tf
  ✓ c6-05-db-subnet-and-nsg.tf
  ✓ c6-06-bastion-subnet-and-nsg.tf
  ✓ c6-07-vnet-outputs.tf
```

```
✓ c7-01-web-linux-vmss-input-variables.tf
  ✓ c7-02-web-linux-vmss-nsg-inline-basic.tf
  ✓ c7-03-web-linux-vmss-resource.tf
  ✓ c7-04-web-linux-vmss-outputs.tf
  ✓ c7-05-web-linux-vmss-autoscaling-default-profile.tf
  ✓ c7-06-web-linux-vmss-autoscaling-default-and-recurrence-profiles.tf
  ✓ c7-07-web-linux-vmss-autoscaling-default-recurrence-fixed-profiles.tf
  ✓ c8-01-bastion-host-input-variables.tf
  ✓ c8-02-bastion-host-linuxvm.tf
  ✓ c8-03-move-ssh-key-to-bastion-host.tf
  ✓ c8-04-AzureBastionService.tf
  ✓ c8-05-bastion-outputs.tf
```

Azure Virtual Network with Subnets and Network Security Groups

Azure Virtual Machine Scale Sets with **Autoscaling + Terraform Dynamic Blocks** for VMSS NSG

Azure Bastion Host Linux VM – Enabled
Azure Bastion Service - Disabled

Azure Standard Load Balancer - Web

✓ c9-01-web-loadbalancer-input-variables.tf
 ✓ c9-02-web-loadbalancer-resource.tf
 ✓ c9-03-web-loadbalancer-outputs.tf
 ✓ **terraform.tfvars**

^ Azure Virtual Machine Scale Sets Basics - Auto Scaling 4 Profiles

8 lectures • 1hr 36min

- Step-01: Introduction to VMSS Autoscaling
- ▶ Step-02: Understand about autoscaling usecase we are going to implement 06:35
- ▶ Step-03: Learn Autoscaling Default Profile using Azure Portal 14:50
- ▶ Step-04: Learn Autoscaling Recurrence and Fixed Profiles 11:36
- ▶ Step-05: Review Bastion Host changes and create basic autoscale resource 10:35
- ▶ Step-06: Create Default Profile Metric Rules CPU Usage, Available Memory and LB 14:40
- ▶ Step-07: Test ScaleOut and ScaleIn Events for Autoscaling Default Profile 17:47
- ▶ Step-08: Create and Test Autoscaling Recurrence Profiles 12:37
- ▶ Step-09: Create Autoscaling Fixed Profile, Test and Delete Resources 06:51

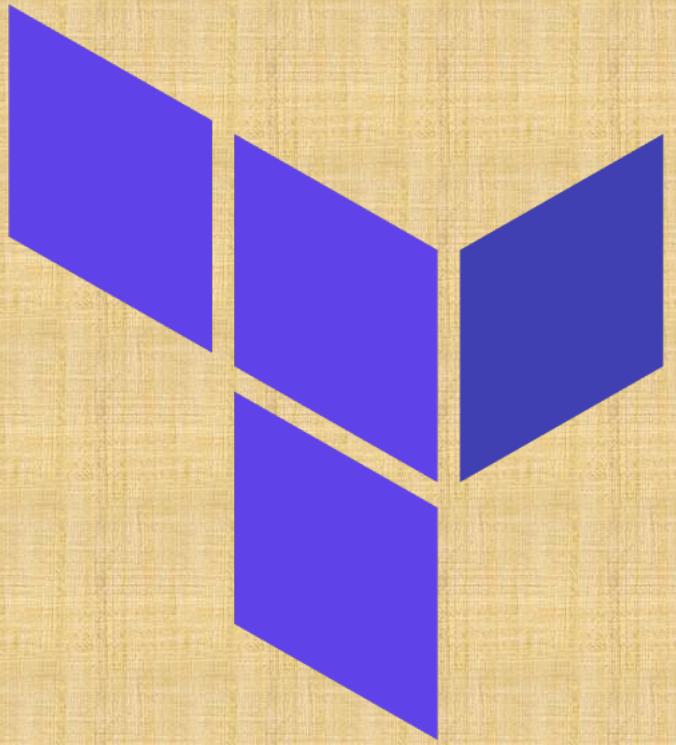
Time it takes to complete this Demo

Real-World Demo

External and Internal LB

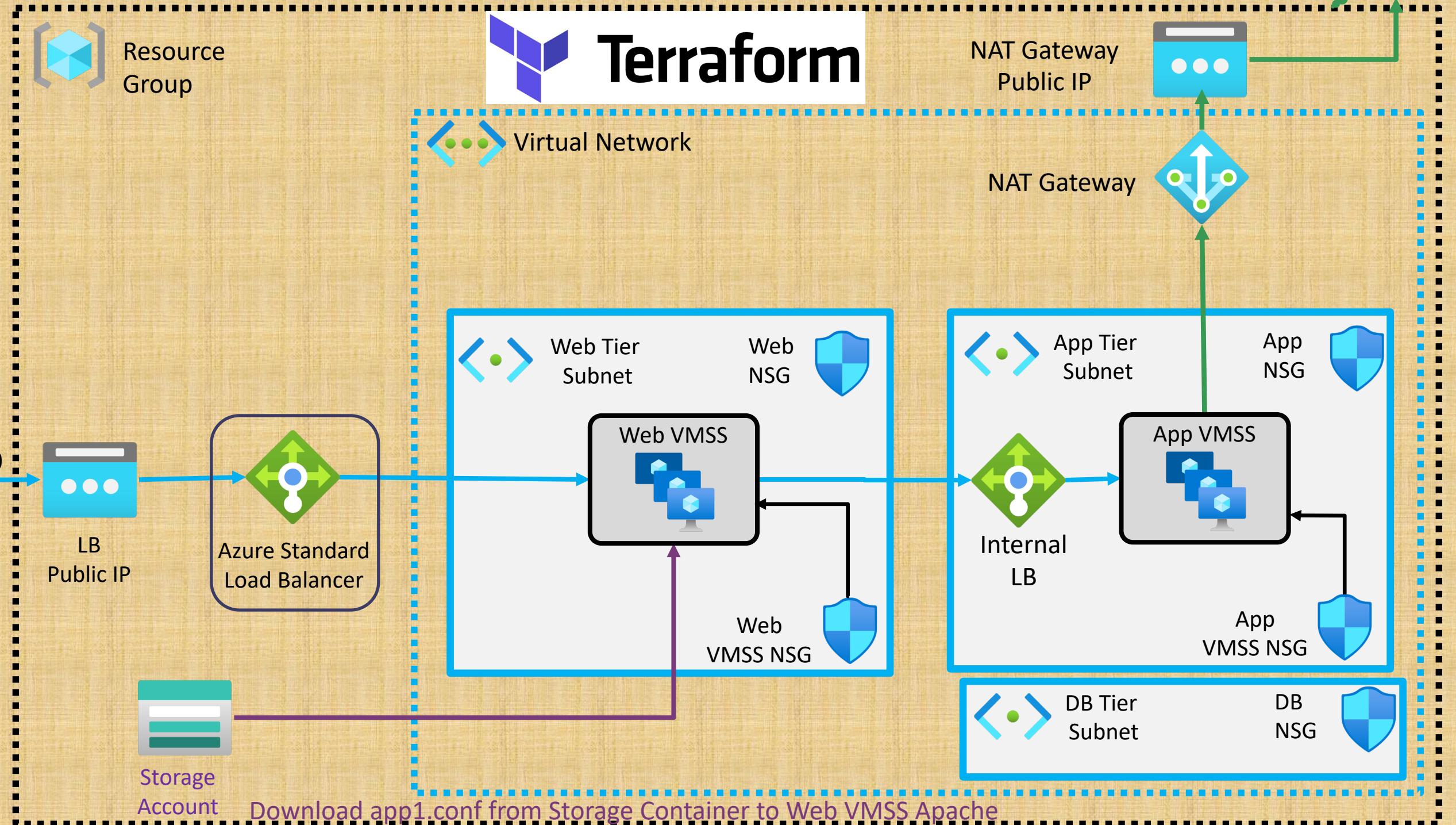
Real-World
Demo

Azure
Standard
Load Balancer



Azure - External LB + Web VMSS + Internal LB + App VMSS

Internet



Azure - External LB + Web VMSS + Internal LB + App VMSS

Internet

SSH Client – Putty / Terminal



Resource Group

Terraform

Virtual Network

Storage Account

Bastion Host
Linux VM Public IP

Port 22



Users

Port 80

LB
Public IP

Azure Standard Load Balancer

Bastion Host Subnet

Bastion NSG



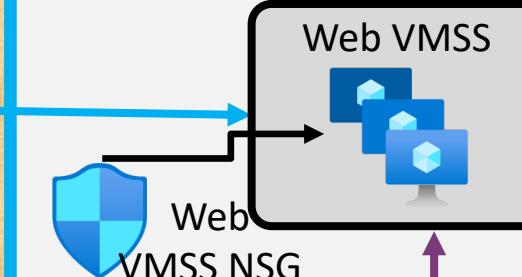
Bastion VM-1

NIC VM Disk

Port 22

Web Tier Subnet

Web NSG



Web VMSS

Web VMSS NSG

Port 22

NAT Gateway
Public IP

NAT Gateway



App Tier Subnet

App NSG

App VMSS

App VMSS NSG

App NSG

DB NSG

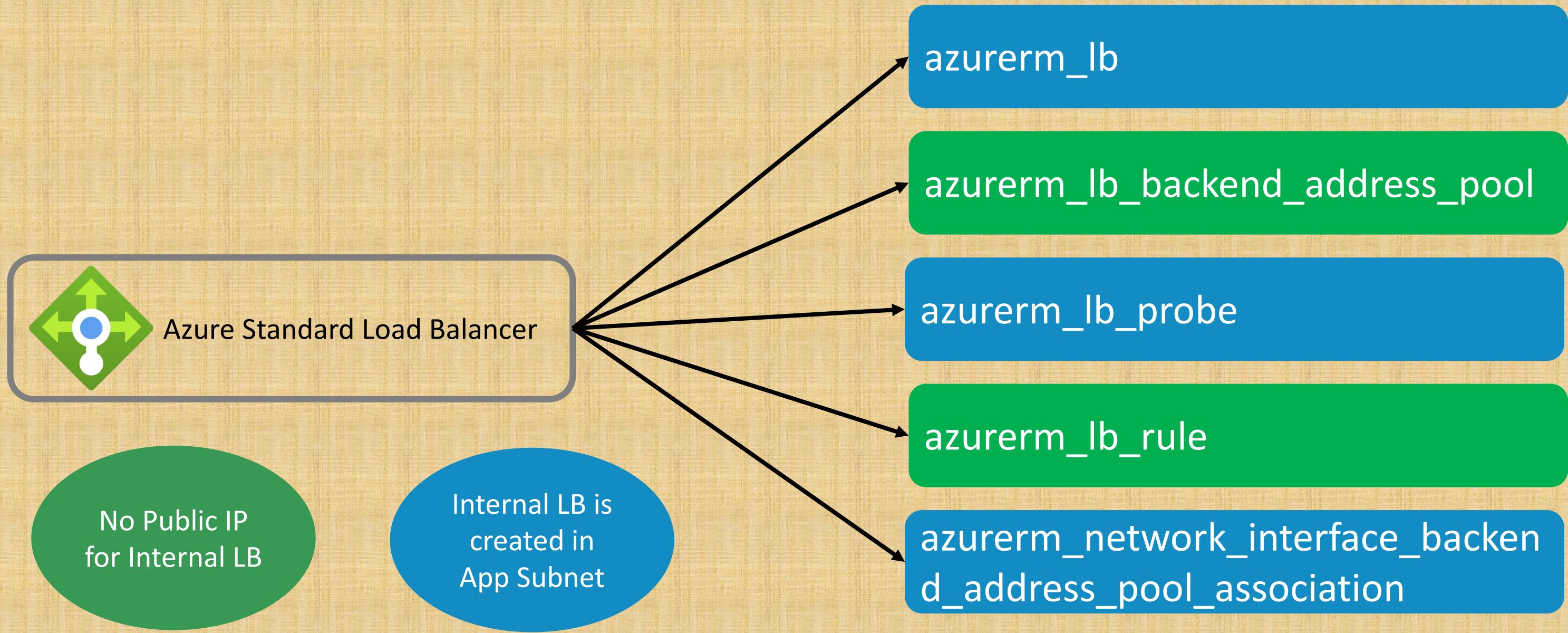
DB Tier Subnet

DB NSG

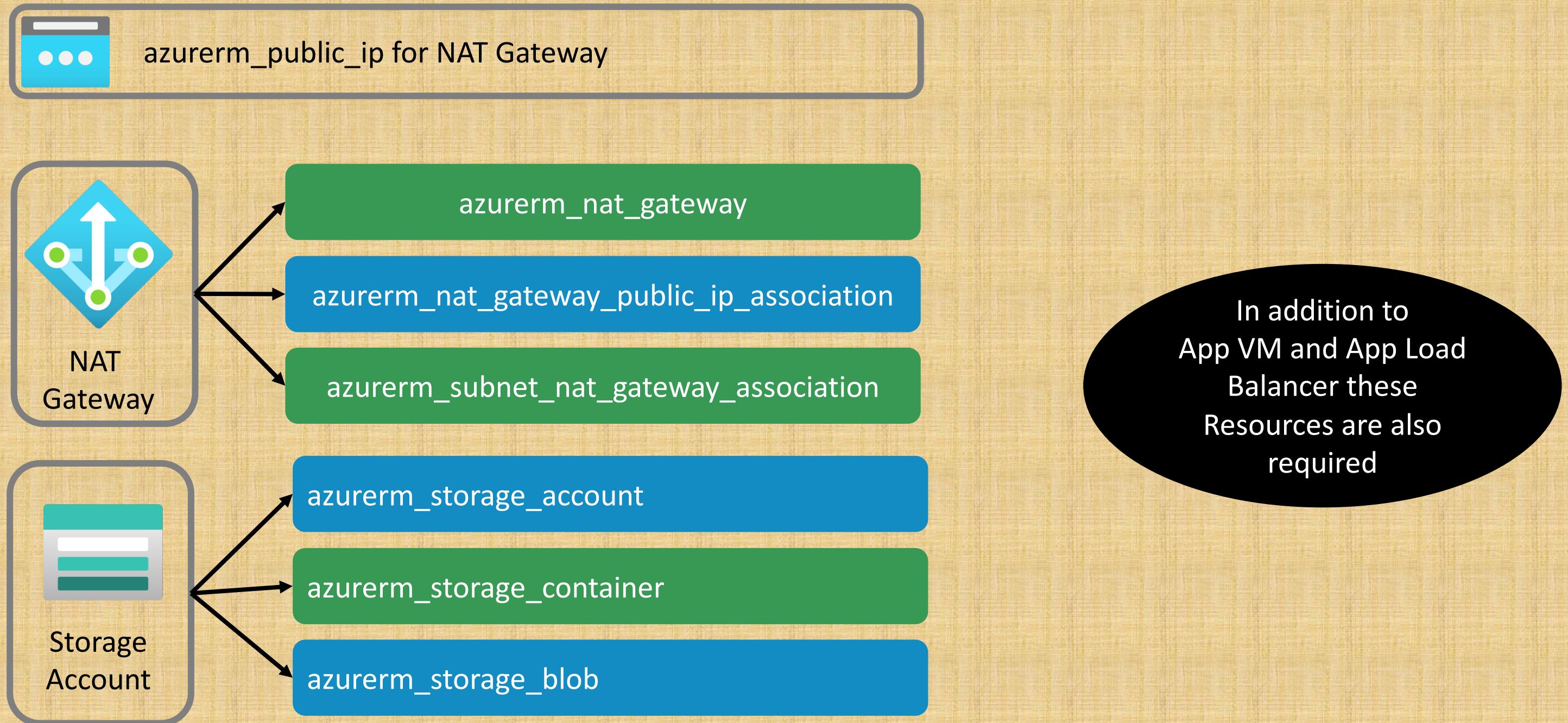
Internal LB

Download app1.conf from Storage Container to Apache

Azure Resources for Internal LB



Azure Resources



Azure Load Balancer – External LB and Internal LB

External LB

web-lb-publicip-1 web-app1-rule web-backend hr-dev-web-vmss

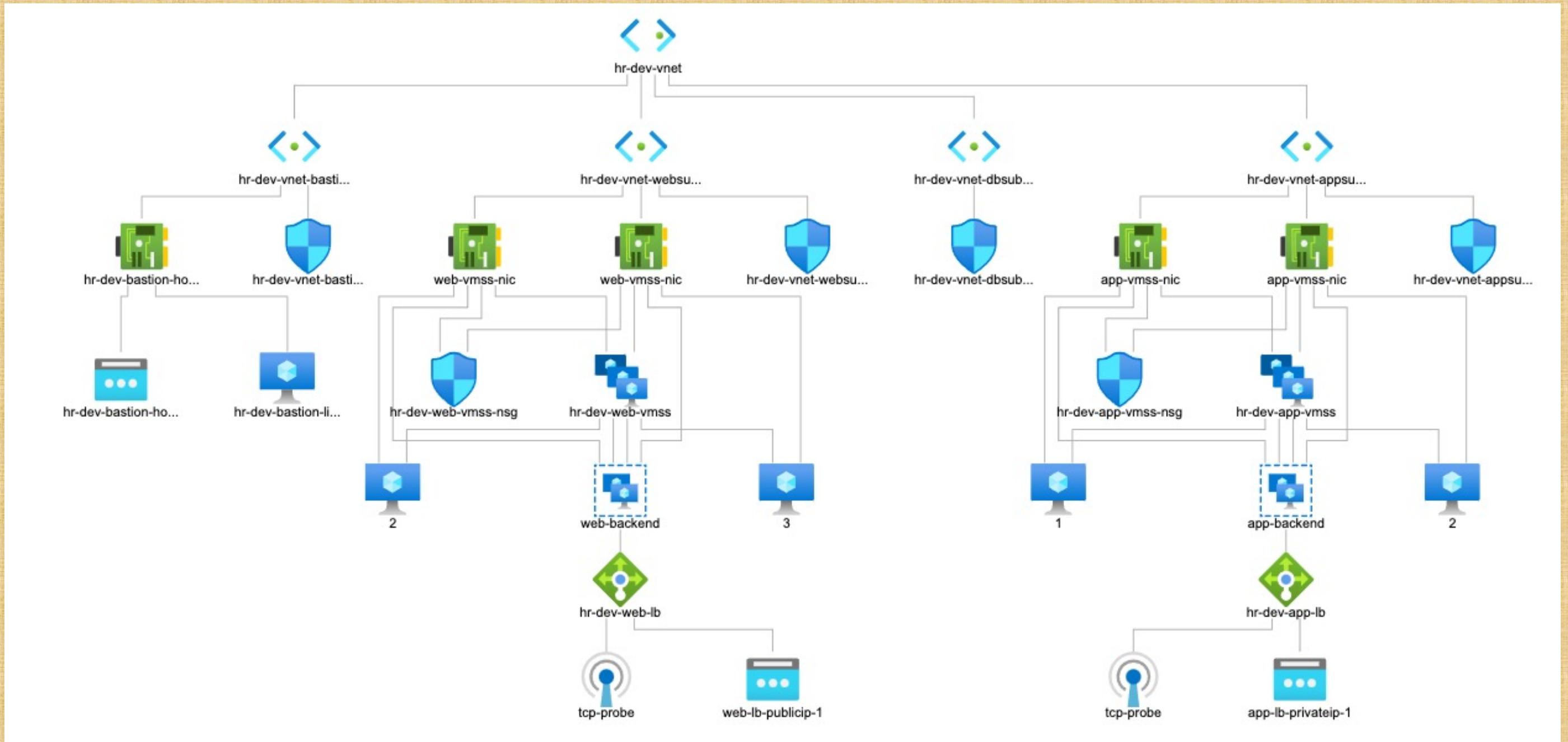


Internal LB

app-lb-privateip-1 app-app1-rule app-backend hr-dev-app-vmss



Azure Resources - Topology



Terraform Configs

terraform-manifests

- > app-scripts
- > ssh-keys
- ↳ c1-versions.tf
- ↳ c2-generic-input-variables.tf
- ↳ c3-locals.tf
- ↳ c4-random-resources.tf
- ↳ c5-resource-group.tf
- ↳ c6-01-vnet-input-variables.tf
- ↳ c6-02-virtual-network.tf
- ↳ c6-03-web-subnet-and-nsg.tf
- ↳ c6-04-app-subnet-and-nsg.tf
- ↳ c6-05-db-subnet-and-nsg.tf
- ↳ c6-06-bastion-subnet-and-nsg.tf
- ↳ c6-07-vnet-outputs.tf

- ↳ c7-01-web-linux-vmss-input-variables.tf
- ↳ c7-02-web-linux-vmss-nsg-inline-basic.tf
- ↳ c7-03-web-linux-vmss-resource.tf
- ↳ c7-04-web-linux-vmss-outputs.tf
- ↳ c7-05-web-linux-vmss-autoscaling-default-profile.tf

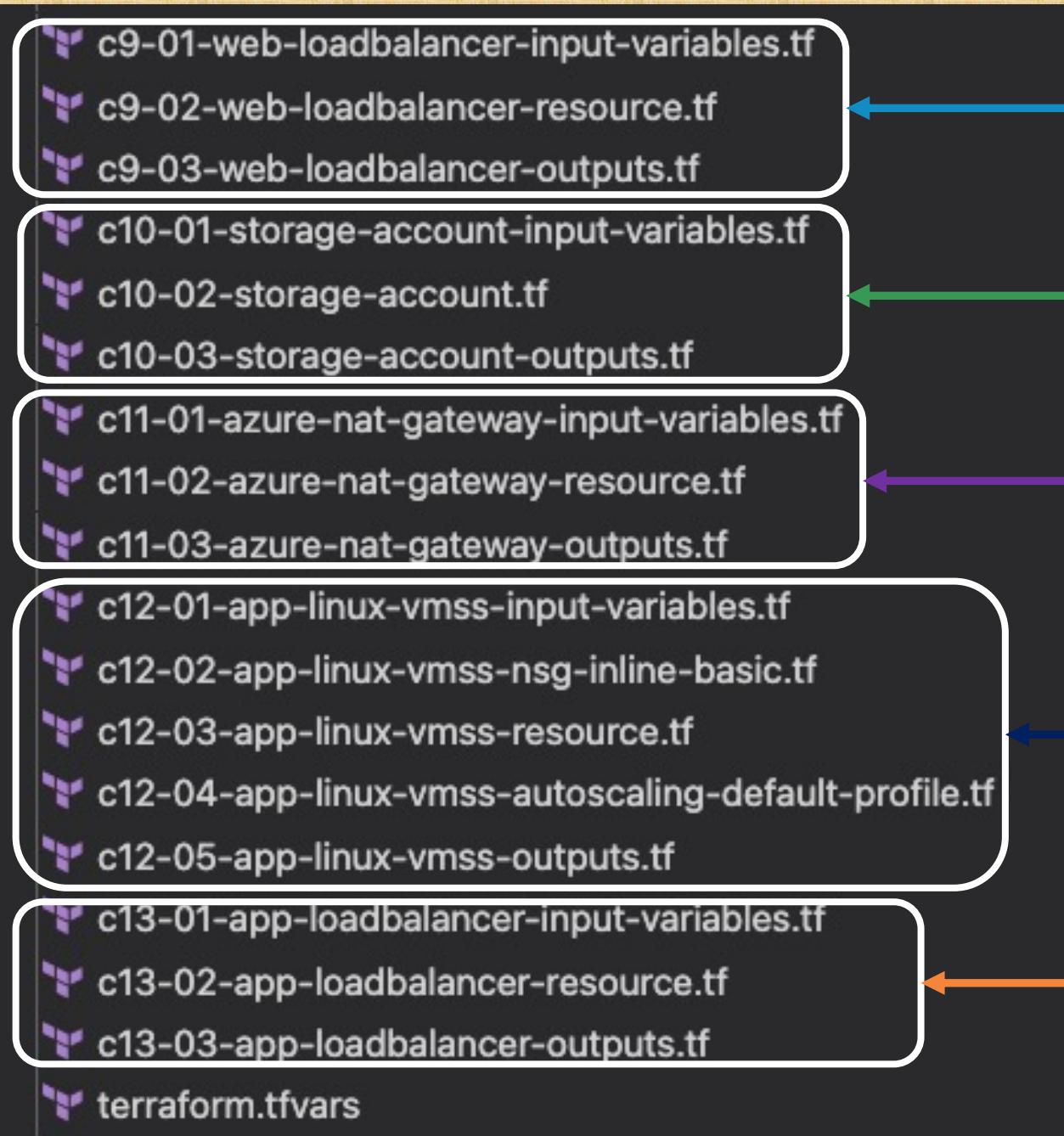
- ↳ c8-01-bastion-host-input-variables.tf
- ↳ c8-02-bastion-host-linuxvm.tf
- ↳ c8-03-move-ssh-key-to-bastion-host.tf
- ↳ c8-04-AzureBastionService.tf
- ↳ c8-05-bastion-outputs.tf

Azure Virtual Network with Subnets and Network Security Groups

Azure Virtual Machine Scale Set with Auto Scaling Profile - Web Tier

Azure Bastion Host Linux VM – Enabled
Azure Bastion Service - Disabled

Terraform Configs



Azure Standard Load Balancer – External Internet facing for Web Tier VMSS Load Balancing

Azure Storage Account – Httpd conf Deployment

Azure NAT Gateway – Outbound communication for App VMSS

Azure Virtual Machine Scale Set with Auto Scaling Profile - App Tier

Azure Standard Load Balancer – Internal for App Tier VMSS Load Balancing

^ Azure Internet LB + Web VMSS + Internal LB + App VMSS

8 lectures • 1hr 16min

- Step-01: Introduction to Internal LB Implementation Usecase
- ▶ Step-02: Create Storage Account and Container 11:50
- ▶ Step-03: Understan app1.conf and create upload files resource to storage contain 08:15
- ▶ Step-04: Create NAT Gateway Resources and associate to App Subnet 05:27
- ▶ Step-05: Review App VMSS Resources TF Configs 07:58
- ▶ Step-06: Review App Load Balancer Resources and Terraform TFPARS file 08:30
- ▶ Step-07: Execute TF Commands, Verify Resources on Azure Portal 10:03
- ▶ Step-08: Review Web VM Custom Data in detail 10:00
- ▶ Step-09: Verify Resources using Bastion Host (Point to Point) and Clean-Up 14:03

Time it takes to complete this Demo

Real-World Demo

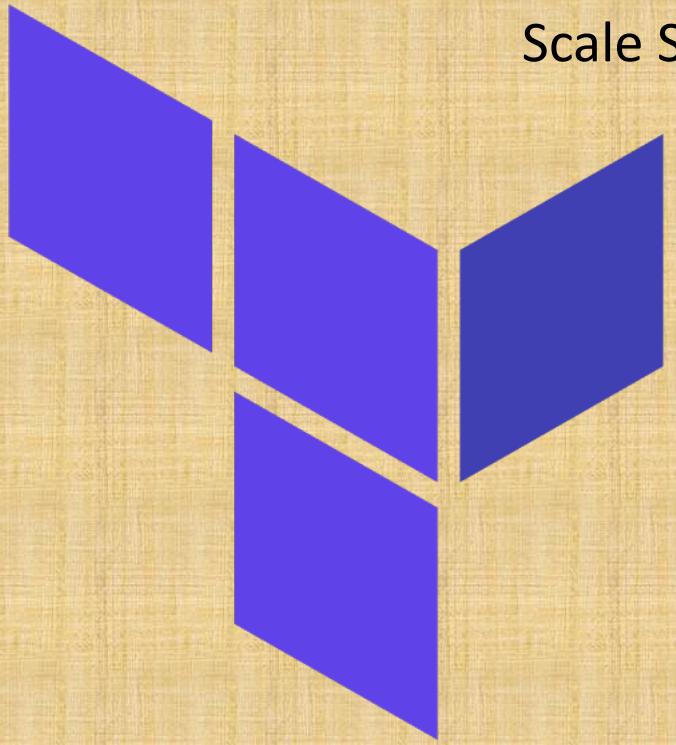


Azure
Load Balancer

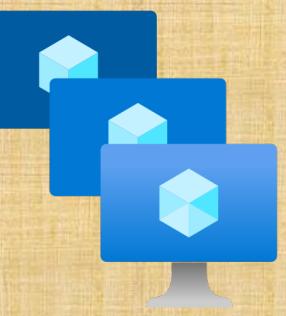


Azure Private DNS Zones

Real-World
Demo

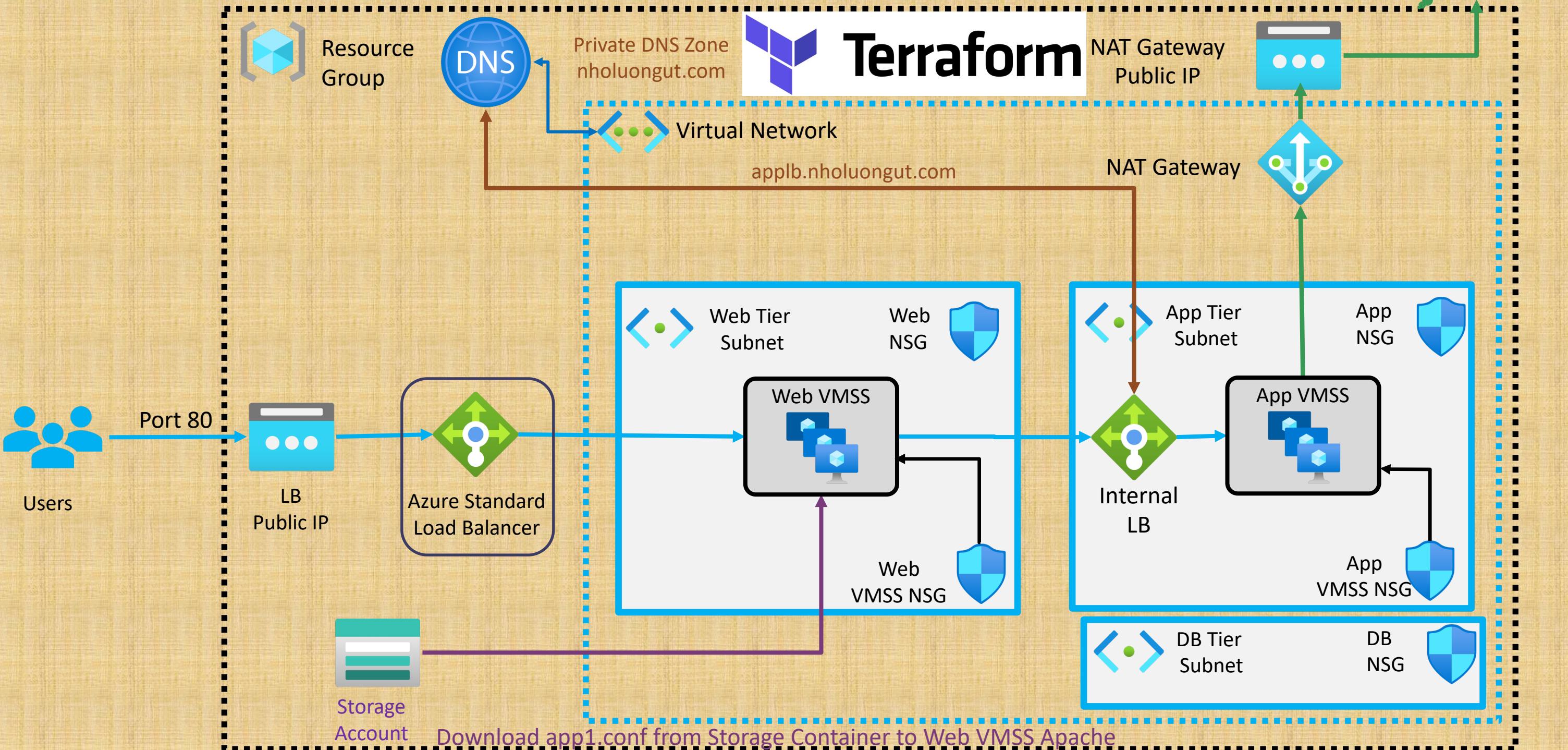


VM
Scale Sets



Azure - Private DNS Zones

Terraform

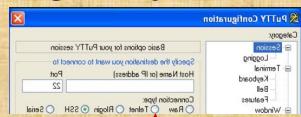


Azure - Private DNS Zones

applb.nholuongut.com

Internet

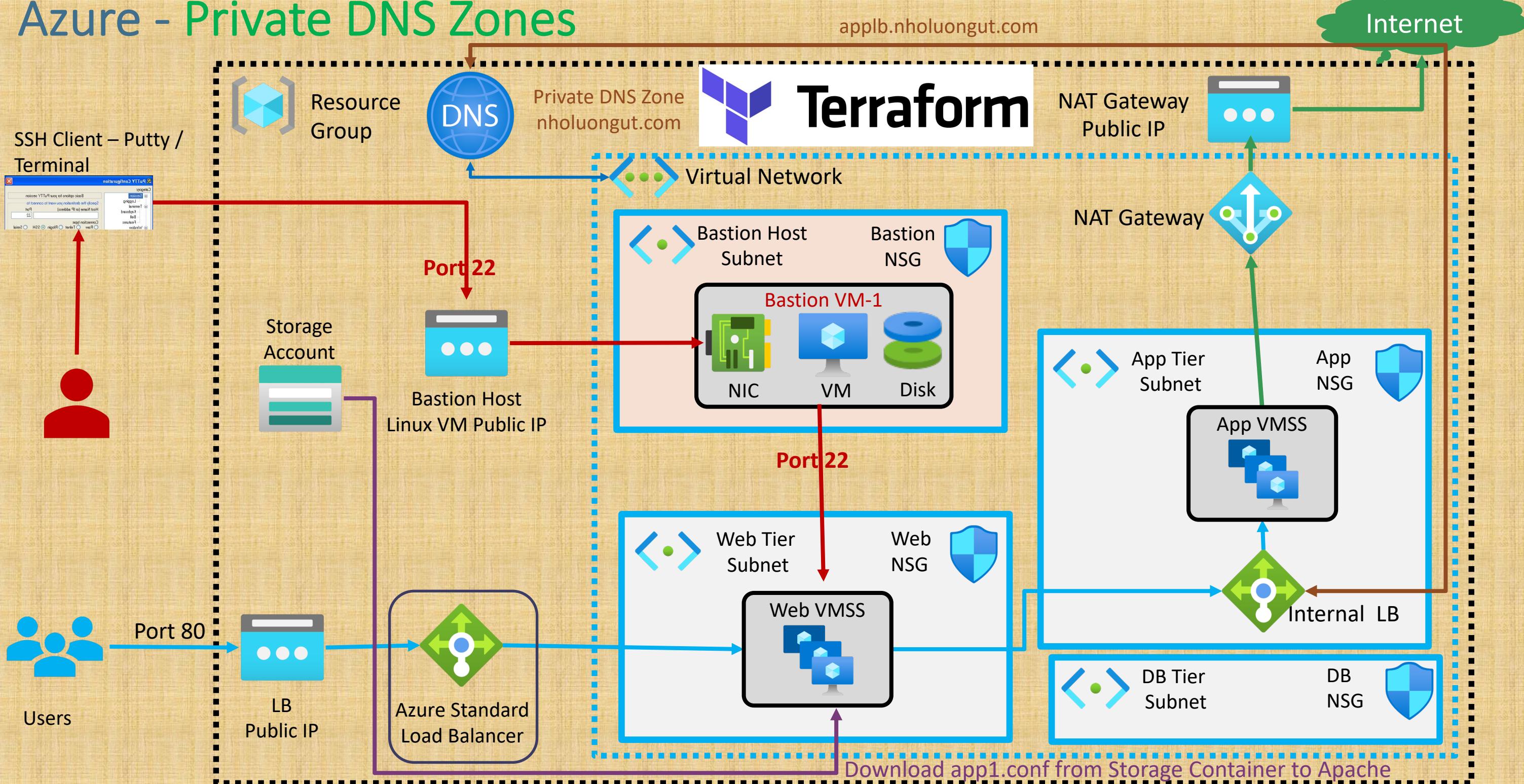
SSH Client – Putty / Terminal



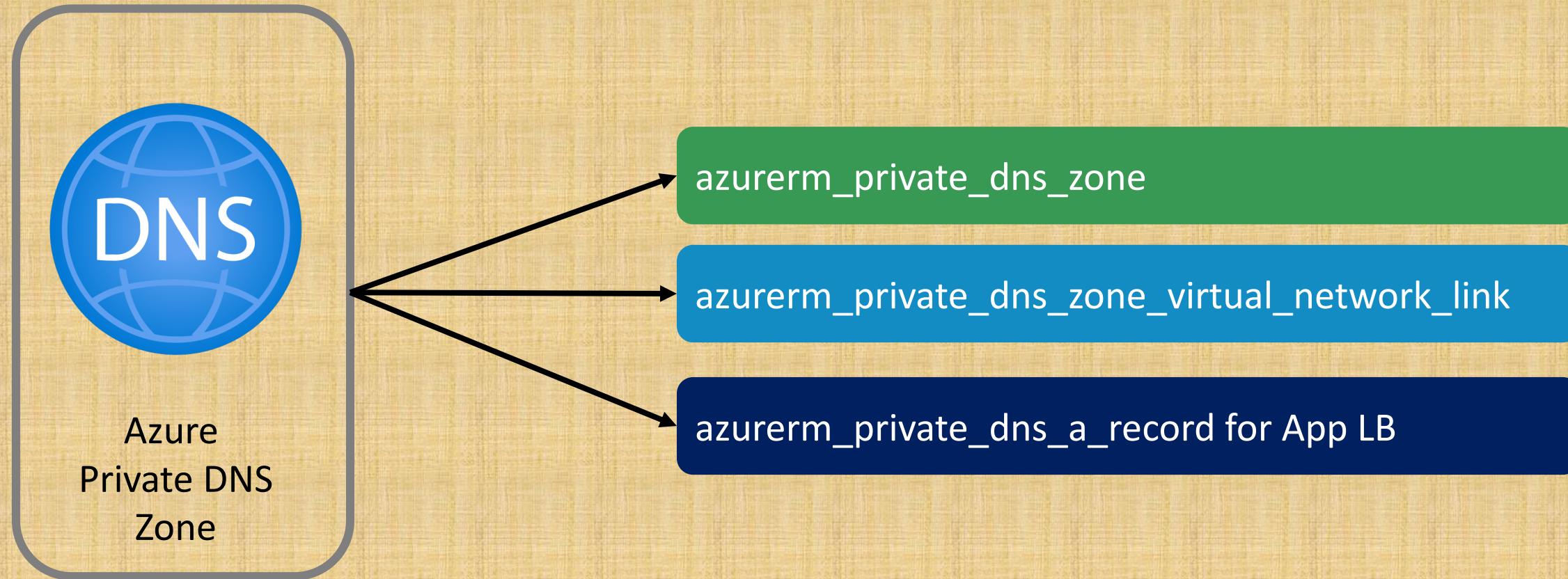
Users

Author: Nho Luong

Skill: DevOps Engineer Lead



Azure Resources



Terraform Configs

terraform-manifests

- > app-scripts
- > ssh-keys
- ↳ c1-versions.tf
- ↳ c2-generic-input-variables.tf
- ↳ c3-locals.tf
- ↳ c4-random-resources.tf
- ↳ c5-resource-group.tf
- ↳ c6-01-vnet-input-variables.tf
- ↳ c6-02-virtual-network.tf
- ↳ c6-03-web-subnet-and-nsg.tf
- ↳ c6-04-app-subnet-and-nsg.tf
- ↳ c6-05-db-subnet-and-nsg.tf
- ↳ c6-06-bastion-subnet-and-nsg.tf
- ↳ c6-07-vnet-outputs.tf

- ↳ c7-01-web-linux-vmss-input-variables.tf
- ↳ c7-02-web-linux-vmss-nsg-inline-basic.tf
- ↳ c7-03-web-linux-vmss-resource.tf
- ↳ c7-04-web-linux-vmss-outputs.tf
- ↳ c7-05-web-linux-vmss-autoscaling-default-profile.tf

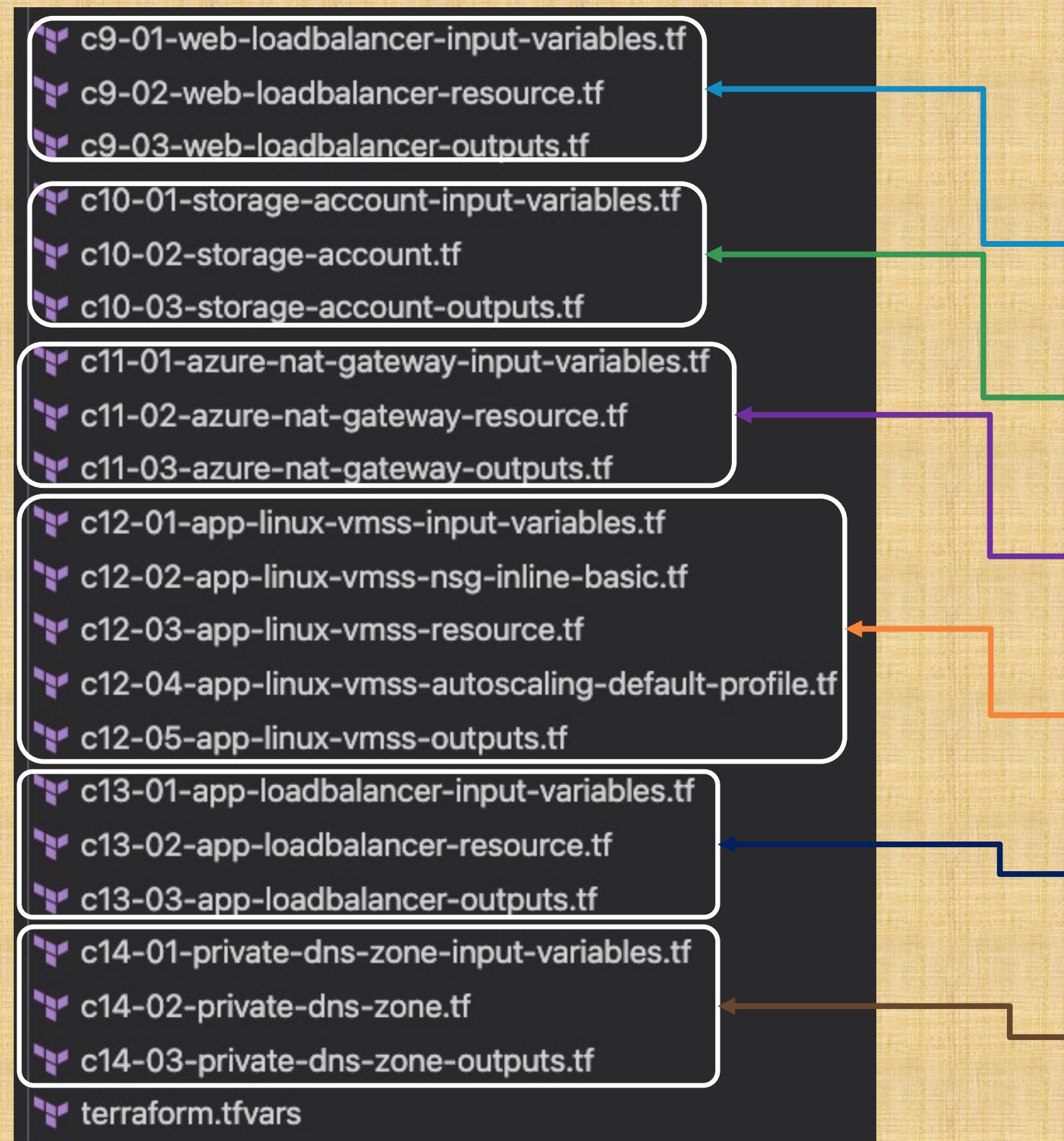
- ↳ c8-01-bastion-host-input-variables.tf
- ↳ c8-02-bastion-host-linuxvm.tf
- ↳ c8-03-move-ssh-key-to-bastion-host.tf
- ↳ c8-04-AzureBastionService.tf
- ↳ c8-05-bastion-outputs.tf

Azure Virtual Network with Subnets and Network Security Groups

Azure Virtual Machine Scale Set with Auto Scaling Profile - Web Tier

Azure Bastion Host Linux VM – Enabled
Azure Bastion Service - Disabled

Terraform Configs



^ Azure Private DNS Zones

- Step-01: Introduction to Azure Private DNS Zones
- ▶ Step-02: Create Private DNS Zone Resources and Update app1.conf
- ▶ Step-03: Execute TF Commands, Verify Private DNS Zones functioning and Delete Re

2 lectures • 22min

13:47

08:01

Time it takes to complete this Demo

Real-World Demo



Real-World
Demo

Azure Delegate DNS Domain To Azure Public DNS Zone



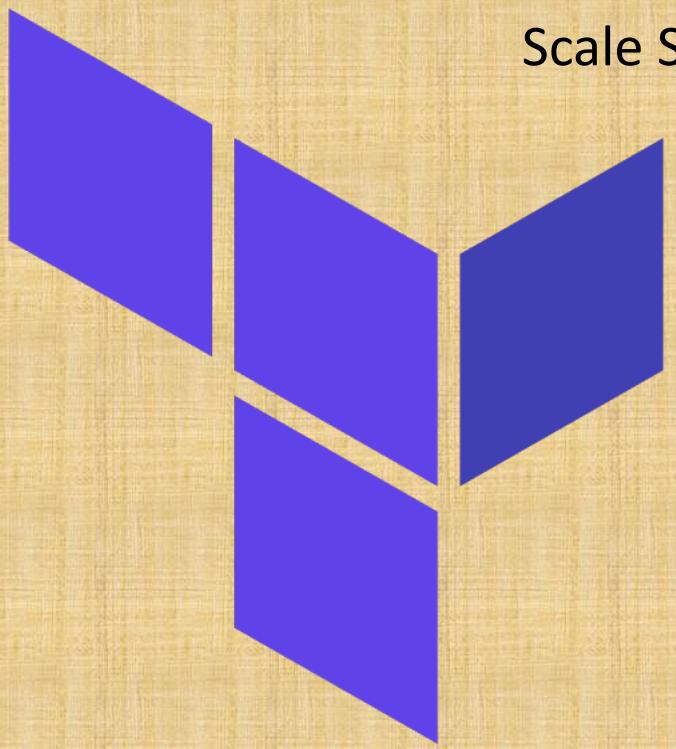


Azure
Load Balancer



Azure Public DNS Zones

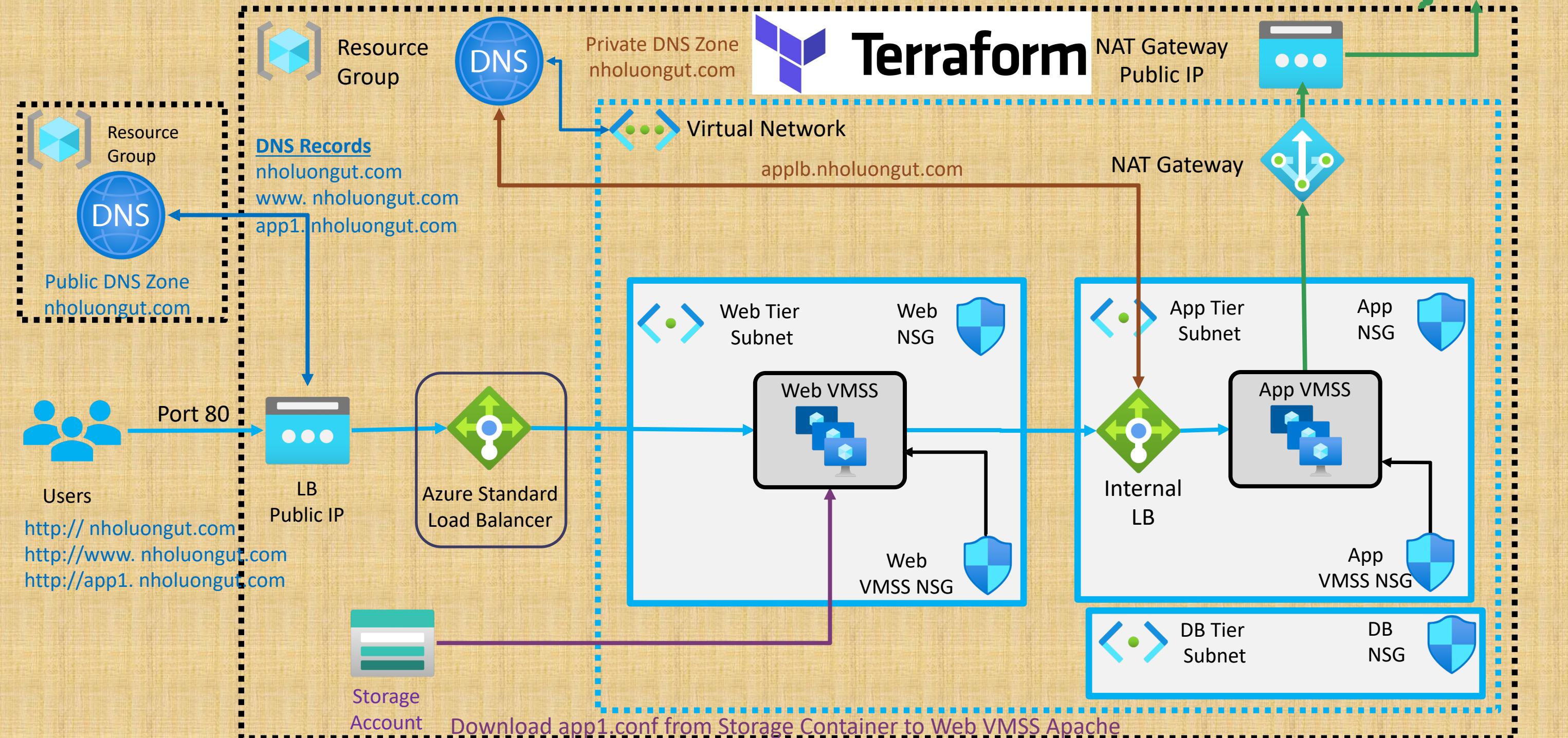
Real-World
Demo



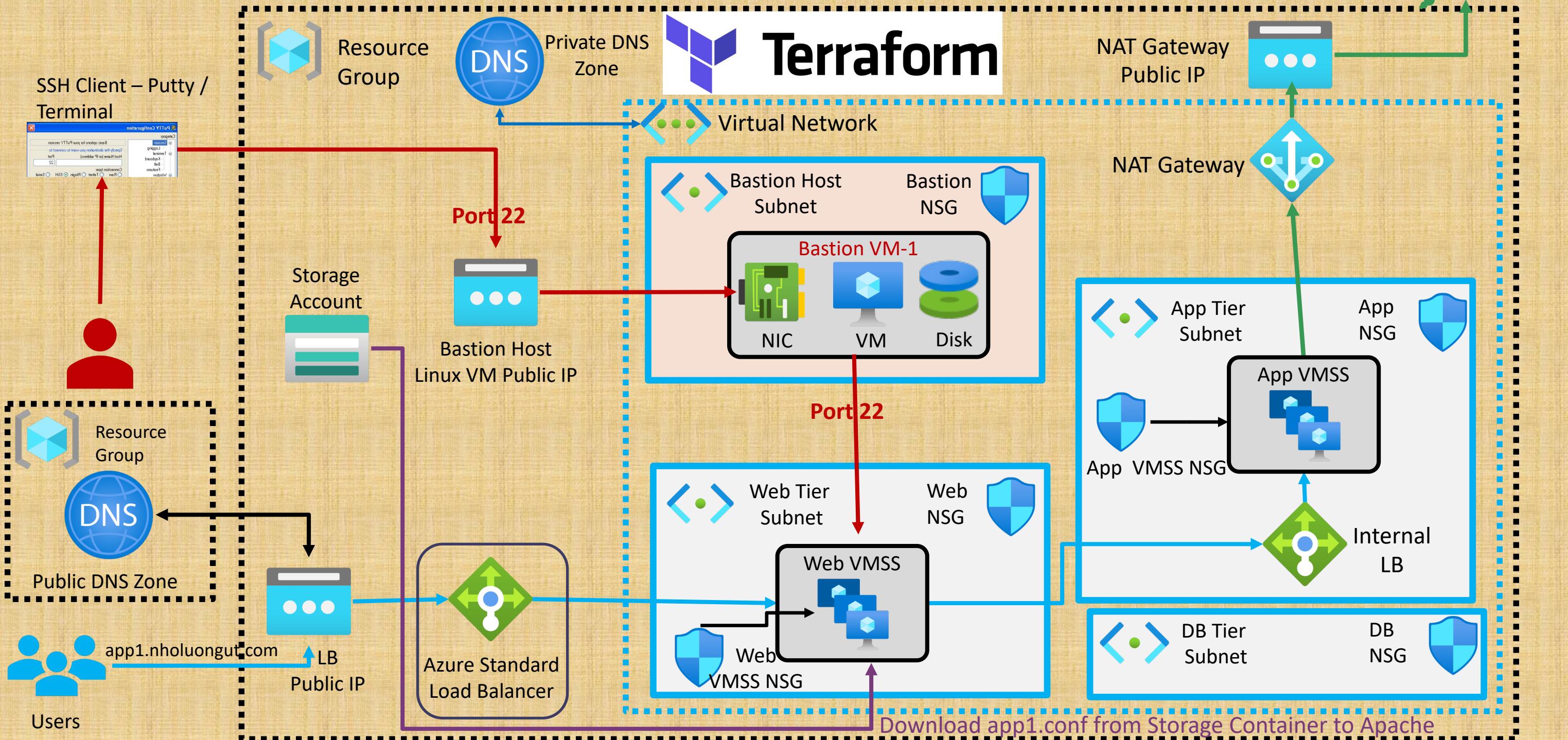
VM
Scale Sets

Azure - Public DNS Zones

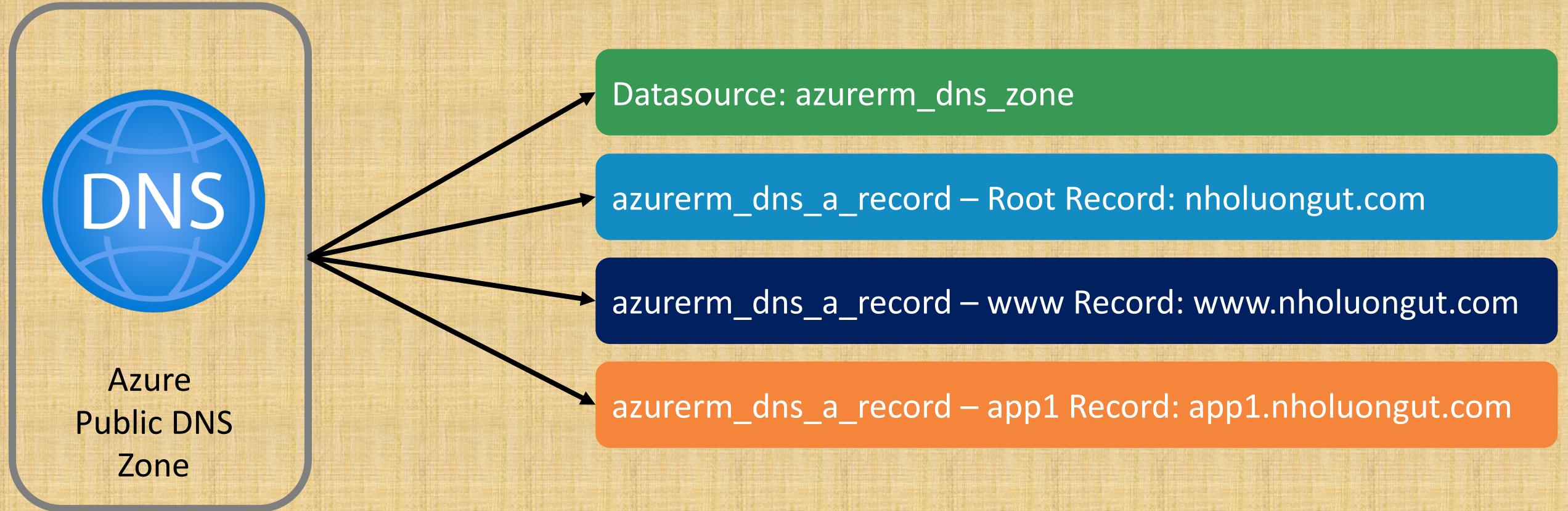
Terraform



Azure - Public DNS Zone



Azure Resources



Terraform Configs

terraform-manifests

- > app-scripts
- > ssh-keys
- ↳ c1-versions.tf
- ↳ c2-generic-input-variables.tf
- ↳ c3-locals.tf
- ↳ c4-random-resources.tf
- ↳ c5-resource-group.tf
- ↳ c6-01-vnet-input-variables.tf
- ↳ c6-02-virtual-network.tf
- ↳ c6-03-web-subnet-and-nsg.tf
- ↳ c6-04-app-subnet-and-nsg.tf
- ↳ c6-05-db-subnet-and-nsg.tf
- ↳ c6-06-bastion-subnet-and-nsg.tf
- ↳ c6-07-vnet-outputs.tf

- ↳ c7-01-web-linux-vmss-input-variables.tf
- ↳ c7-02-web-linux-vmss-nsg-inline-basic.tf
- ↳ c7-03-web-linux-vmss-resource.tf
- ↳ c7-04-web-linux-vmss-outputs.tf
- ↳ c7-05-web-linux-vmss-autoscaling-default-profile.tf

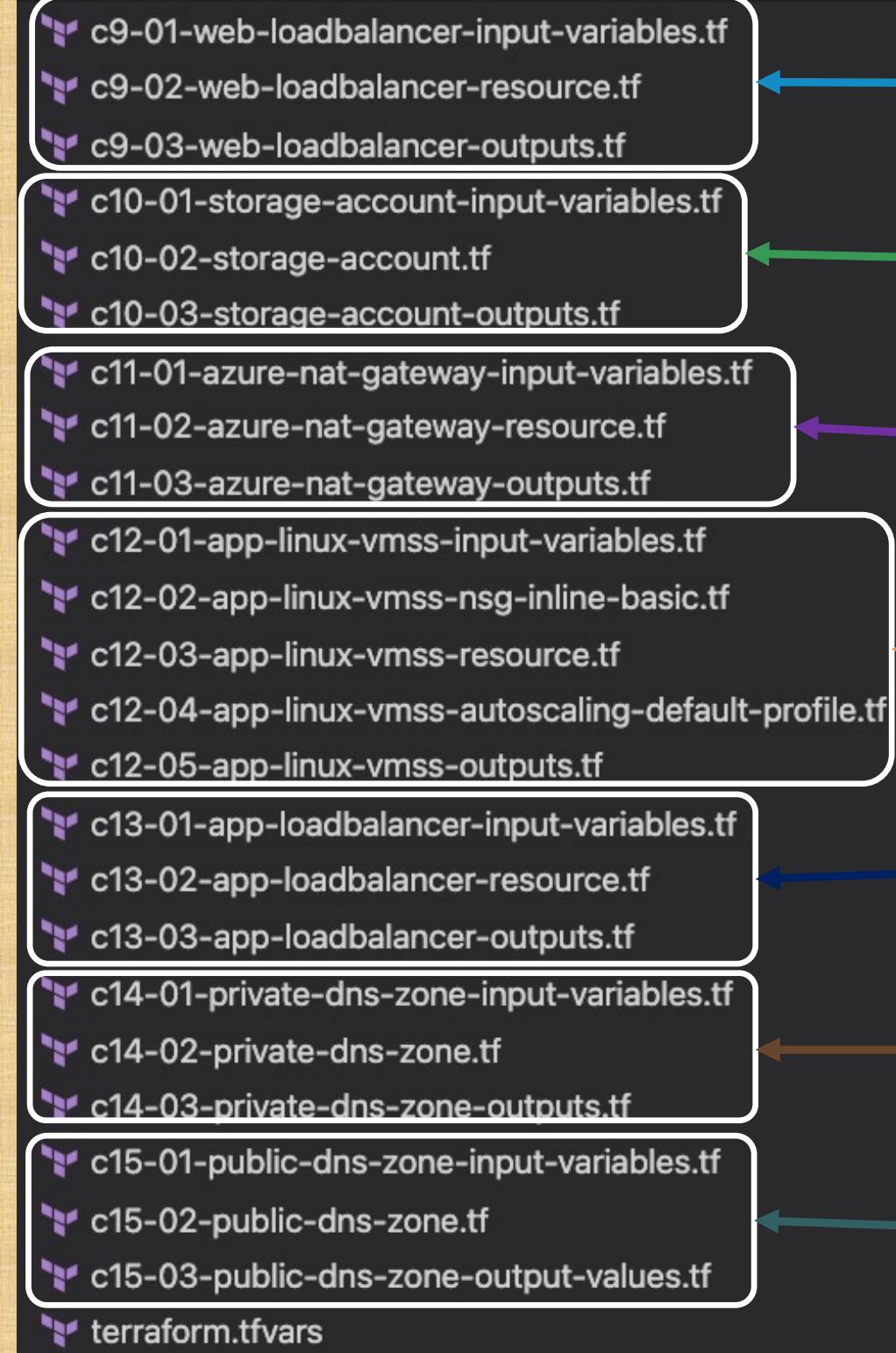
- ↳ c8-01-bastion-host-input-variables.tf
- ↳ c8-02-bastion-host-linuxvm.tf
- ↳ c8-03-move-ssh-key-to-bastion-host.tf
- ↳ c8-04-AzureBastionService.tf
- ↳ c8-05-bastion-outputs.tf

Azure Virtual Network with Subnets and Network Security Groups

Azure Virtual Machine Scale Set with Auto Scaling Profile - Web Tier

Azure Bastion Host Linux VM – Enabled
Azure Bastion Service - Disabled

Terraform Configs



Azure Standard Load Balancer – External
(Web Tier Load Balancer)

Azure Storage Account – Deploy httpd conf

Azure NAT Gateway – Outbound
communication for App VMSS

Azure Virtual Machine Scale Set with Auto
Scaling Profile - App Tier

Azure Standard Load Balancer – Internal
(App Tier Load Balancer)

Azure Private DNS Zone for App Tier Load
Balancer internal DNS Name in VNET

Azure Public DNS Zone – Access Applications
via Internet using Registered Domain

^ Azure Public DNS Zones

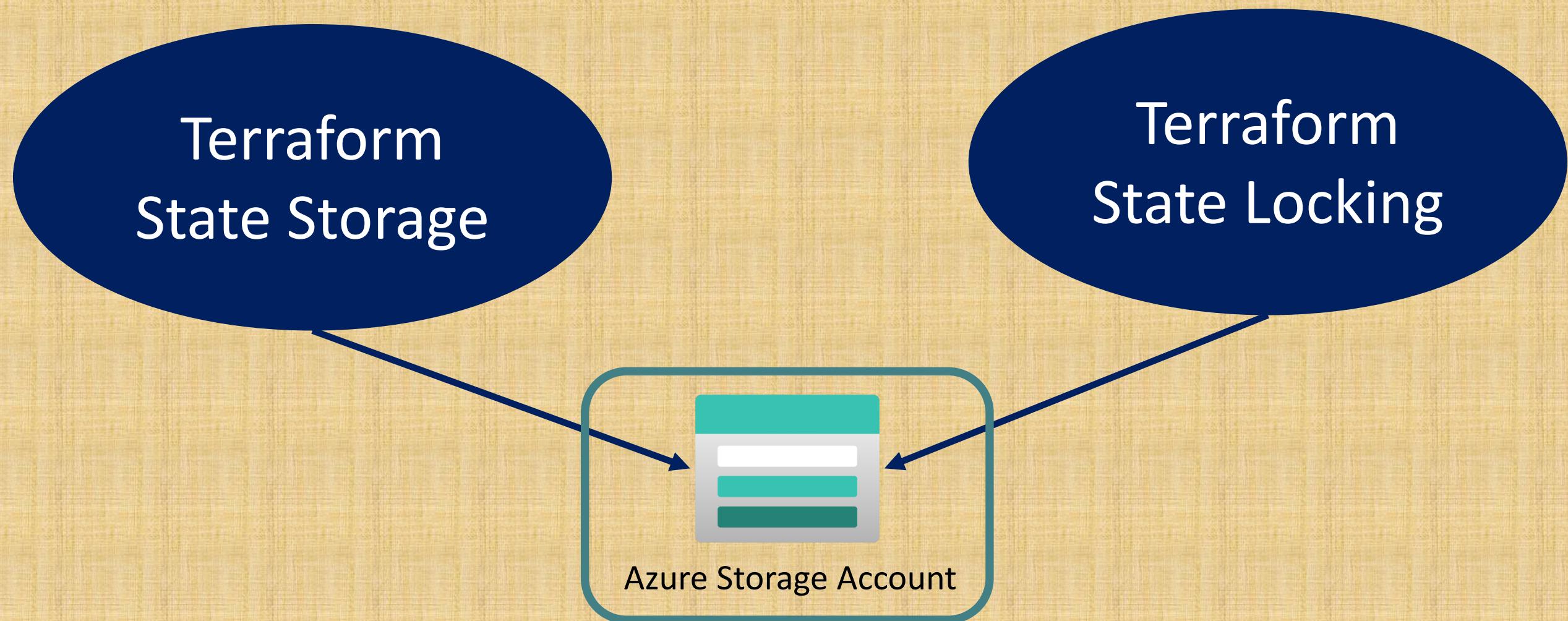
- ❑ Step-01: Introduction to Azure Public DNS Zone
- Step-02: Create Public DNS Records using DNS Zone
Datasource
- Step-03: Execute TF Commands, Verify Public DNS Records
and Clean-Up

2 lectures • 18min



What is Terraform Backend ?

Backends are responsible for storing state and providing an API for state locking.

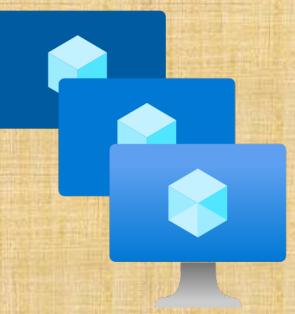




Azure
Load Balancer



Real-World
Demo

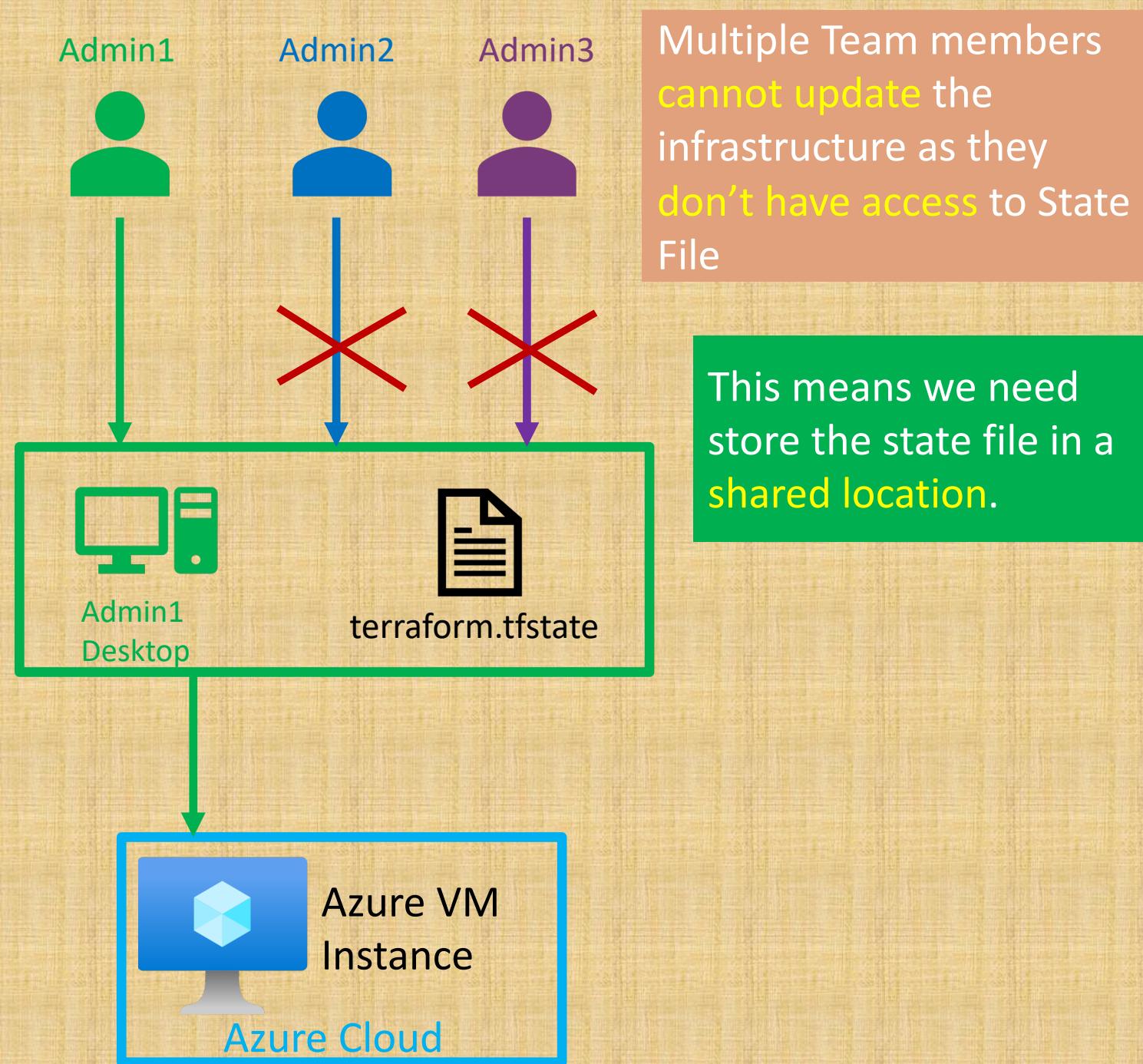


VM
Scale Sets

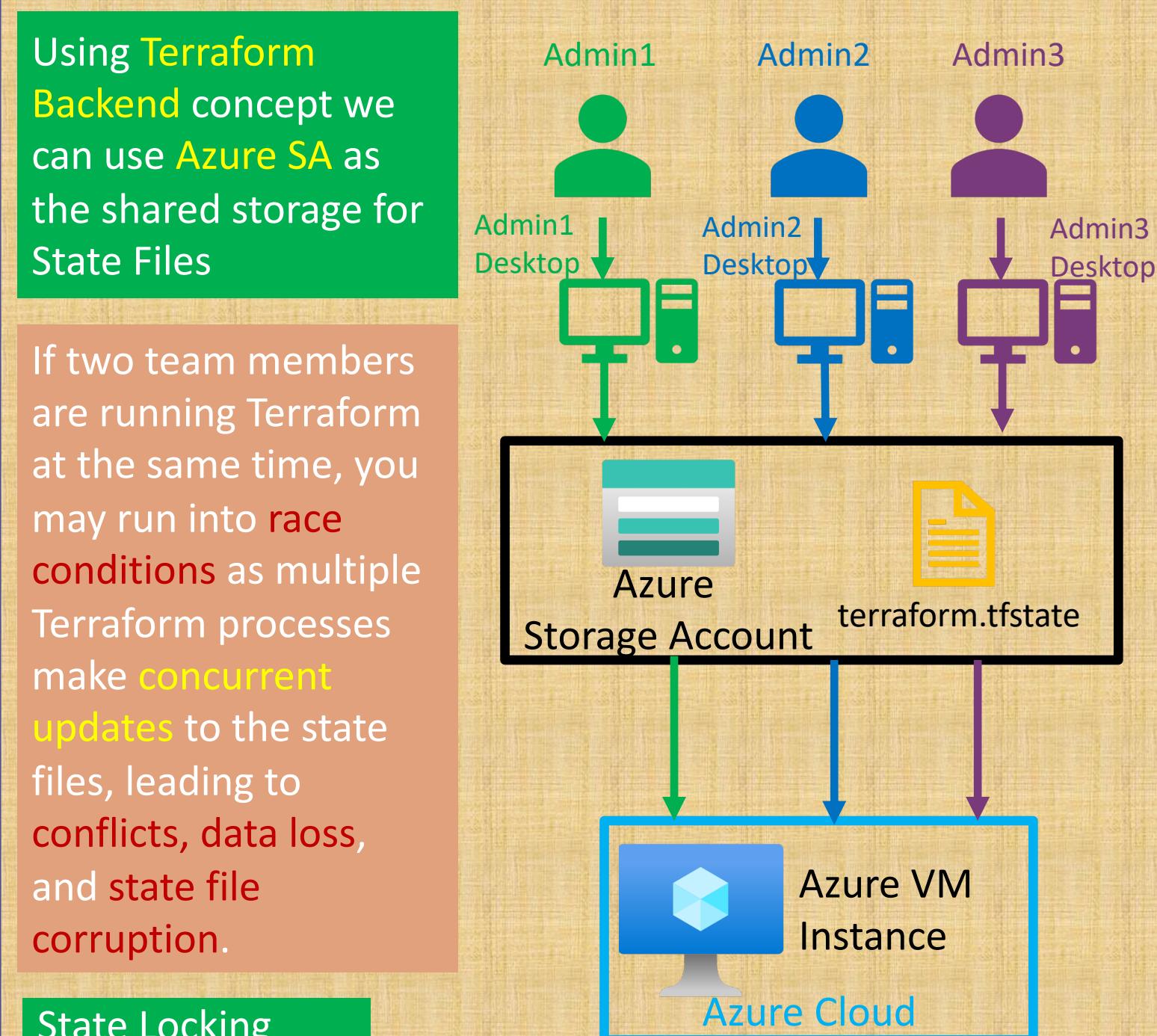
Terraform Backend Remote State Storage with Azure Storage Account



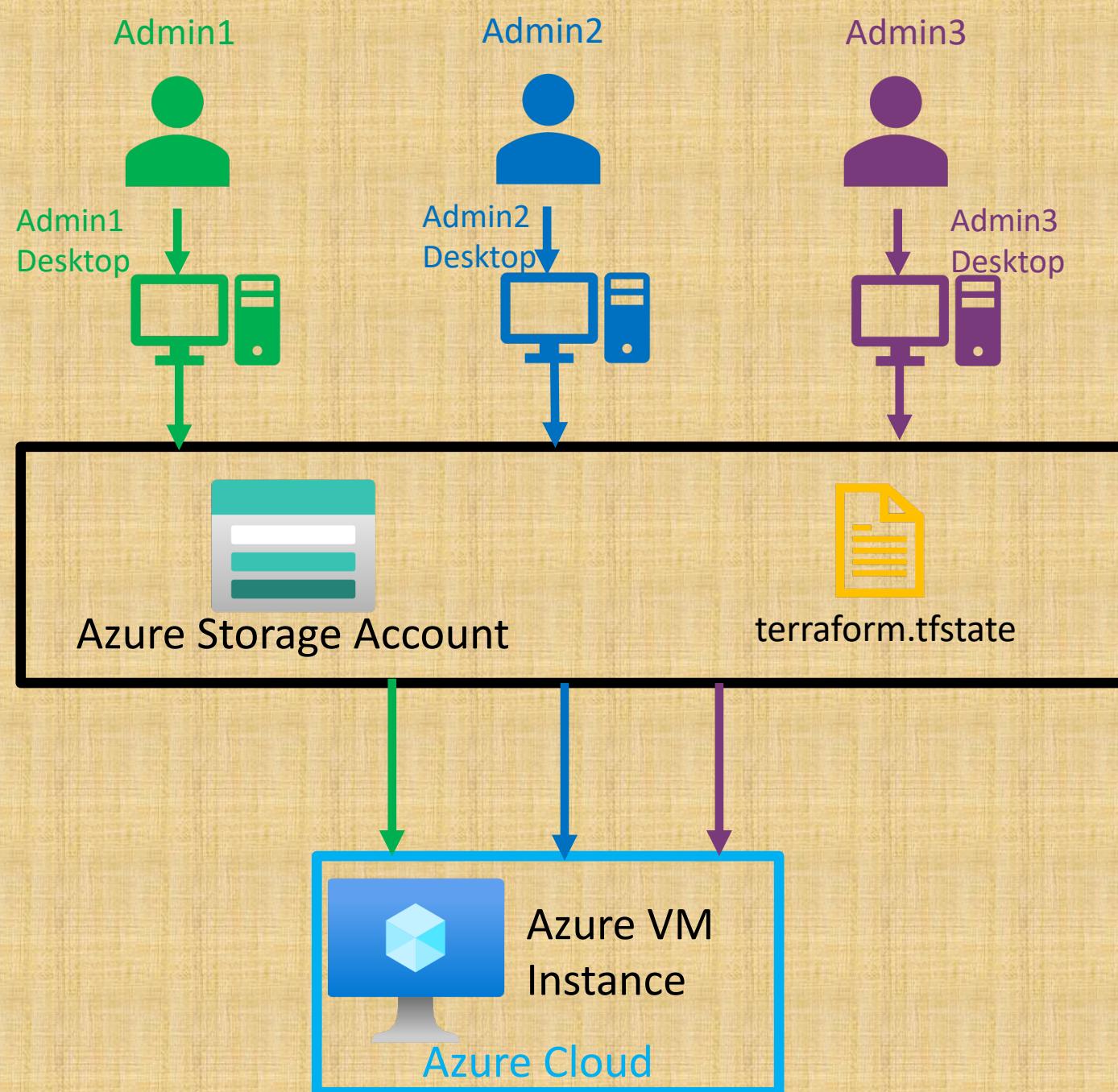
Local State File



Remote State File



Terraform Remote State File with State Locking



Not all backends support State Locking. Azure Storage Account supports State Locking

State locking happens automatically on all operations that could **write state**.

If state locking fails, Terraform **will not continue**.

You can **disable** state locking for most commands with the **-lock flag** but it is **not recommended**.

If acquiring the lock is taking **longer** than expected, Terraform will output a **status message**.

If Terraform doesn't output a message, state locking is still **occurring** if your backend supports it.

Terraform has a **force-unlock command** to manually unlock the state if unlocking failed.

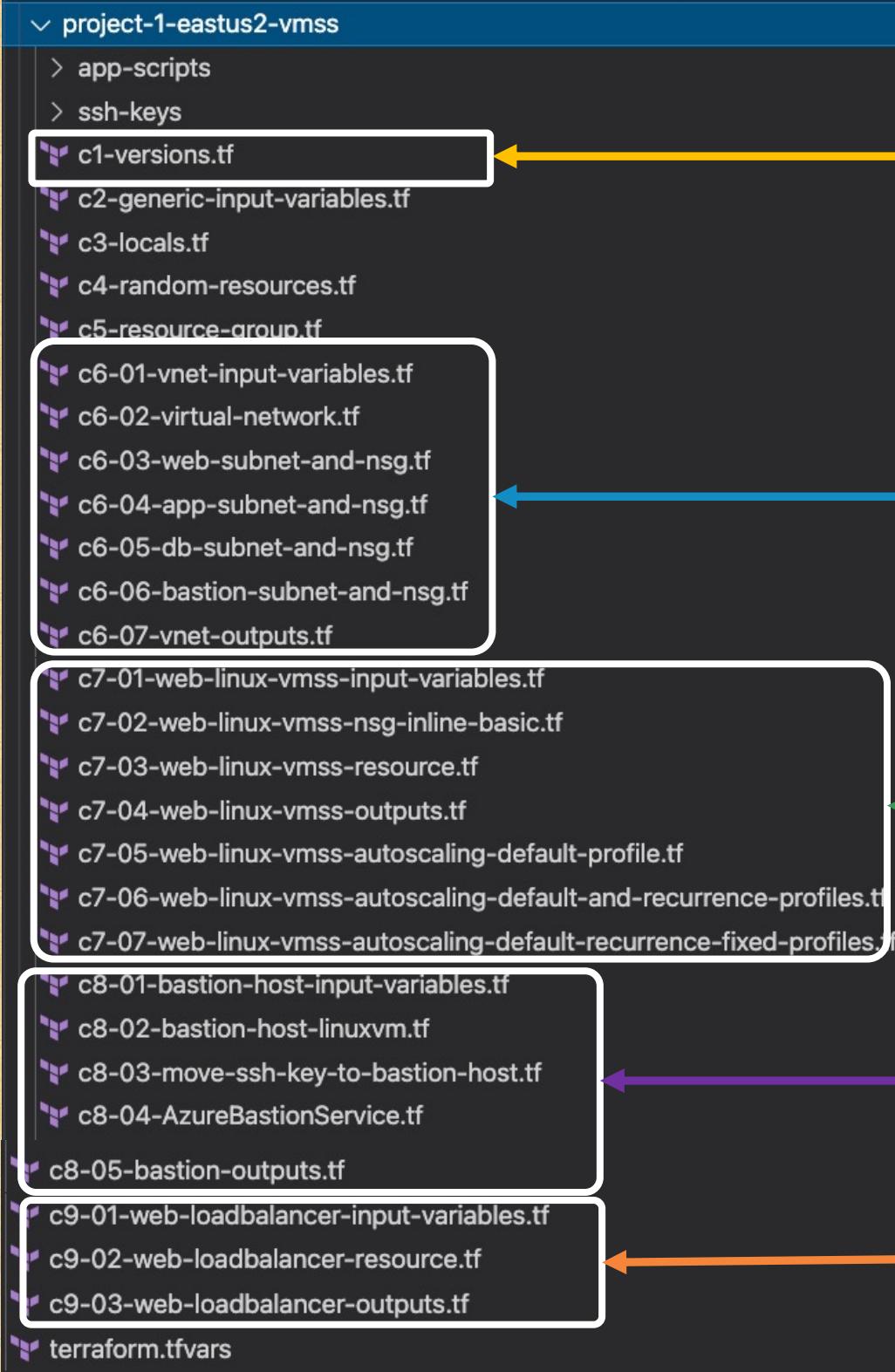
Terraform Remote State File with State Locking

Terraform Remote State Storage

```
# Terraform Block
terraform {
    required_version = ">= 1.0.0"
    required_providers {
        azurerm = {
            source = "string"
            version = ">= 2.0"
        }
        random = {
            source = "hashicorp/random"
            version = ">= 3.0"
        }
    }
}

# Terraform State Storage to Azure Storage Container
backend "azurerm" {
    resource_group_name      = "terraform-storage-rg"
    storage_account_name     = "terraformstate201"
    container_name           = "tfstatefiles"
    key                      = "terraform.tfstate"
}
```

Terraform Configs - Project 1



Terraform Remote Backends – Azure Storage Account

Azure Virtual Network with Subnets and Network Security Groups

Azure Virtual Machine Scale Set with Auto Scaling Profile

Azure Bastion Host Linux VM – Disabled
Azure Bastion Service - Disabled

Azure Standard Load Balancer

^ Terraform Remote State Storage using Azure Storage Account

3 lectures • 31min

- ☐ Step-01: Introduction to Terraform Remote State Storage using Azure Storage Acco

- ▶ Step-02: Create Storage Account, Container and TF Backend config

07:35

- ▶ Step-03: Review remaining TF Configs which needs changes

08:05

- ▶ Step-04: Execute TF Commands, Verify Storage Account TF State files

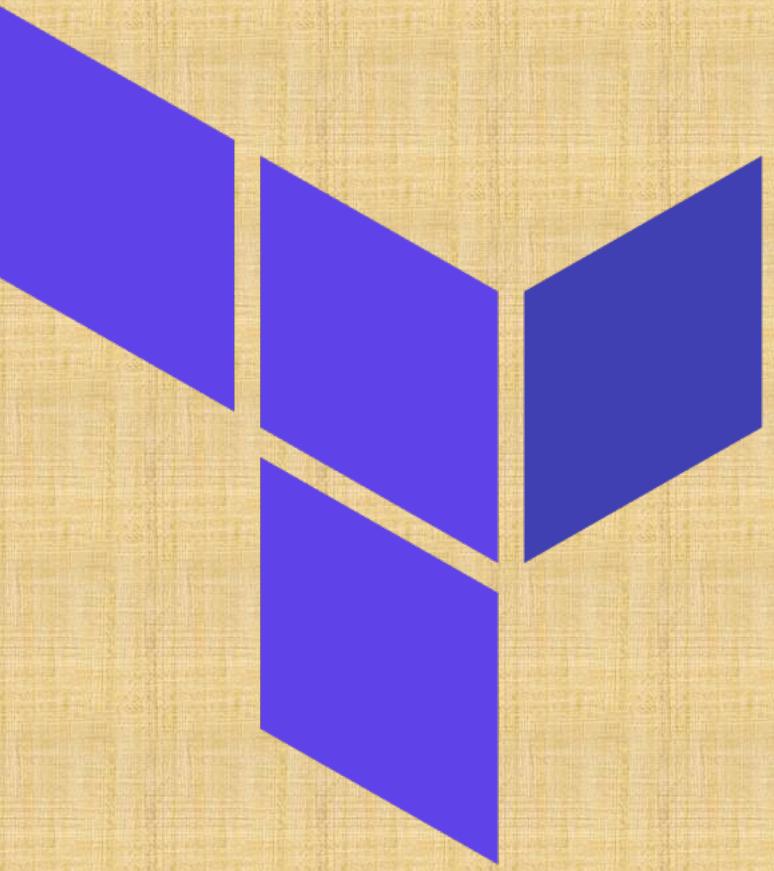
15:05

Time it takes to complete this Demo

Real-World Demo



Terraform Backends



Terraform Backends

Each Terraform configuration can specify a **backend**, which defines where and how operations are performed, where **state** snapshots are stored, etc.

Where Backends are Used

Backend configuration is only used by Terraform CLI.

Terraform Cloud and Terraform Enterprise always use their **own state storage** when performing **Terraform runs**, so they ignore any **backend block** in the configuration.

For Terraform Cloud users also it is always recommended to use **backend block** in Terraform configuration for commands like **terraform taint** which can be executed only using Terraform CLI

Terraform Backends

What Backends Do

1. Where state is stored
2. Where operations are performed.

Store State

Terraform uses persistent state data to keep track of the resources it manages.

Everyone working with a given collection of infrastructure resources must be able to access the same state data (shared state storage).

State Locking

State Locking is to prevent conflicts and inconsistencies when the operations are being performed

Operations

"Operations" refers to performing API requests against infrastructure services in order to create, read, update, or destroy resources.

Not every terraform subcommand performs API operations; many of them only operate on state data.

Only two backends actually perform operations: local and remote.

The remote backend can perform API operations remotely, using Terraform Cloud or Terraform Enterprise.

What are Operations ?
`terraform apply`
`terraform destroy`

Terraform Backends

Backend Types

Enhanced Backends

Enhanced backends can both **store state** and **perform operations**. There are only two enhanced backends: **local** and **remote**

Example for Remote Backend
Performing Operations : Terraform Cloud, Terraform Enterprise

Standard Backends

Standard backends **only store state**, and **rely** on the local backend for performing operations.

Example: AWS S3, Azure RM, Consul, etcd, gcs http and many more



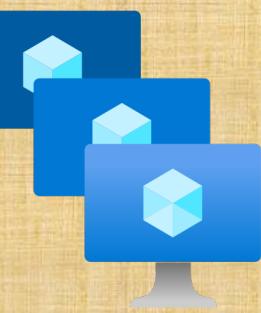
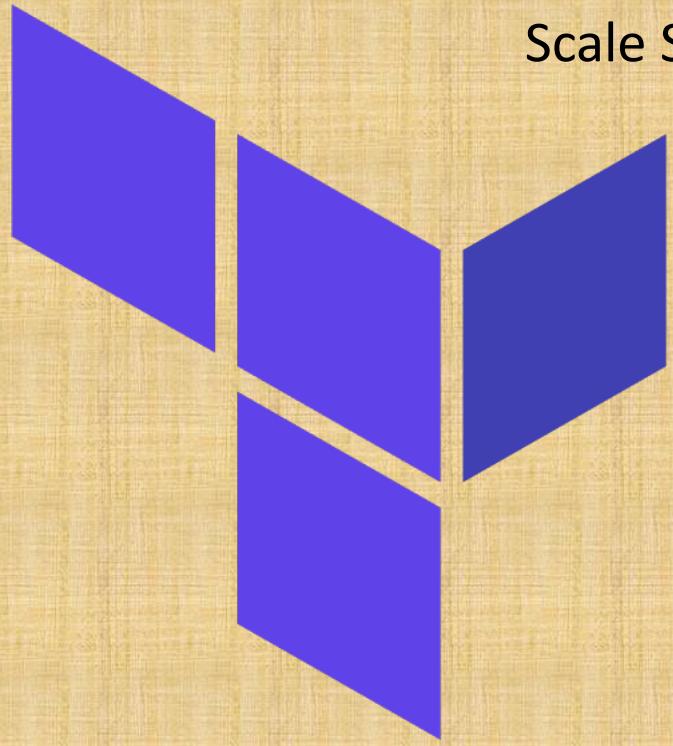
Azure
Load Balancer



Azure
Traffic Manager

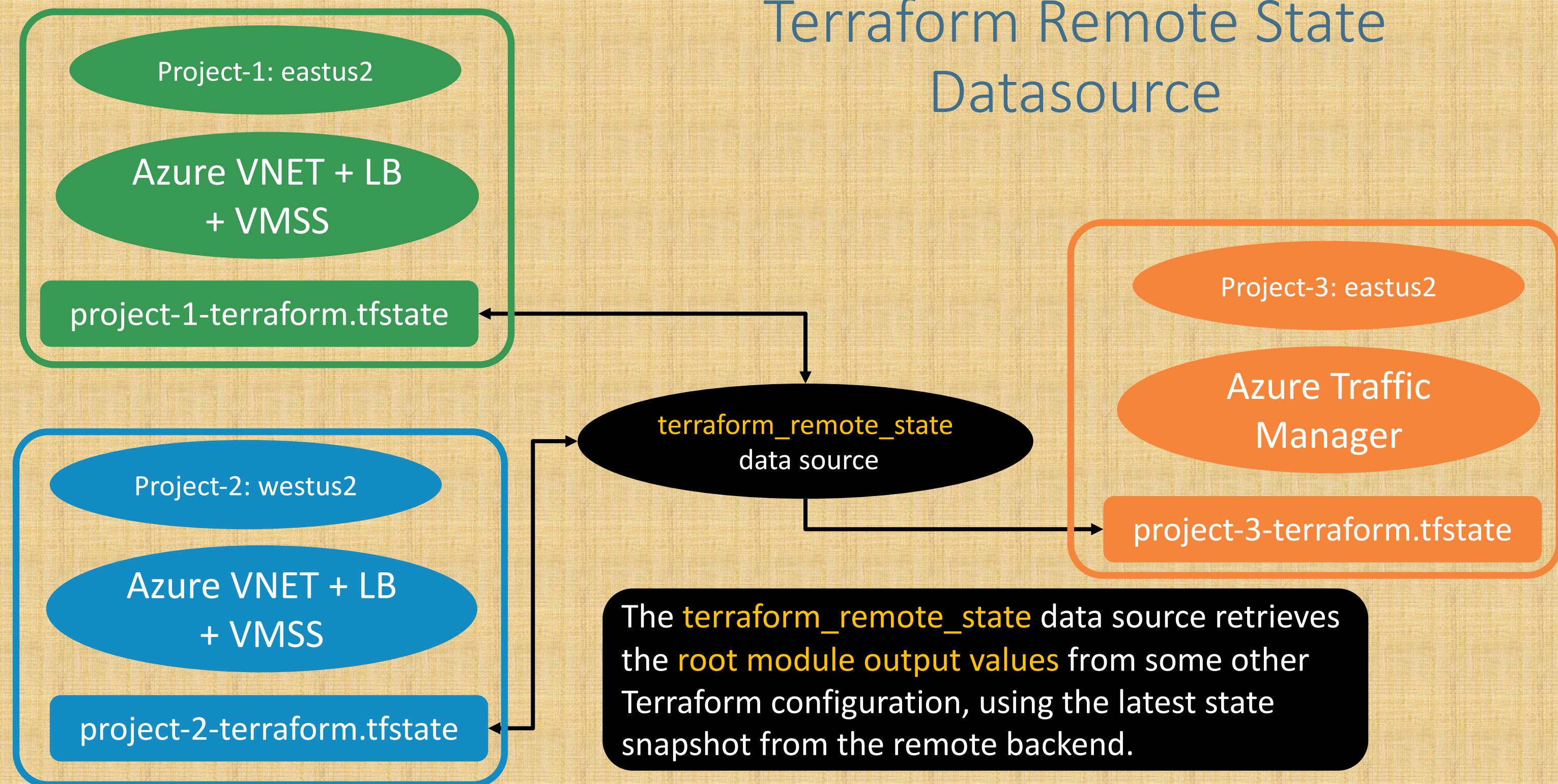
Real-World
Demo

Azure Traffic Manager Terraform Remote State Datasource



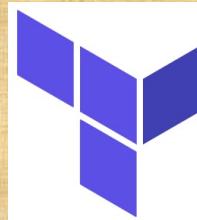
VM
Scale Sets

Terraform Remote State Datasource

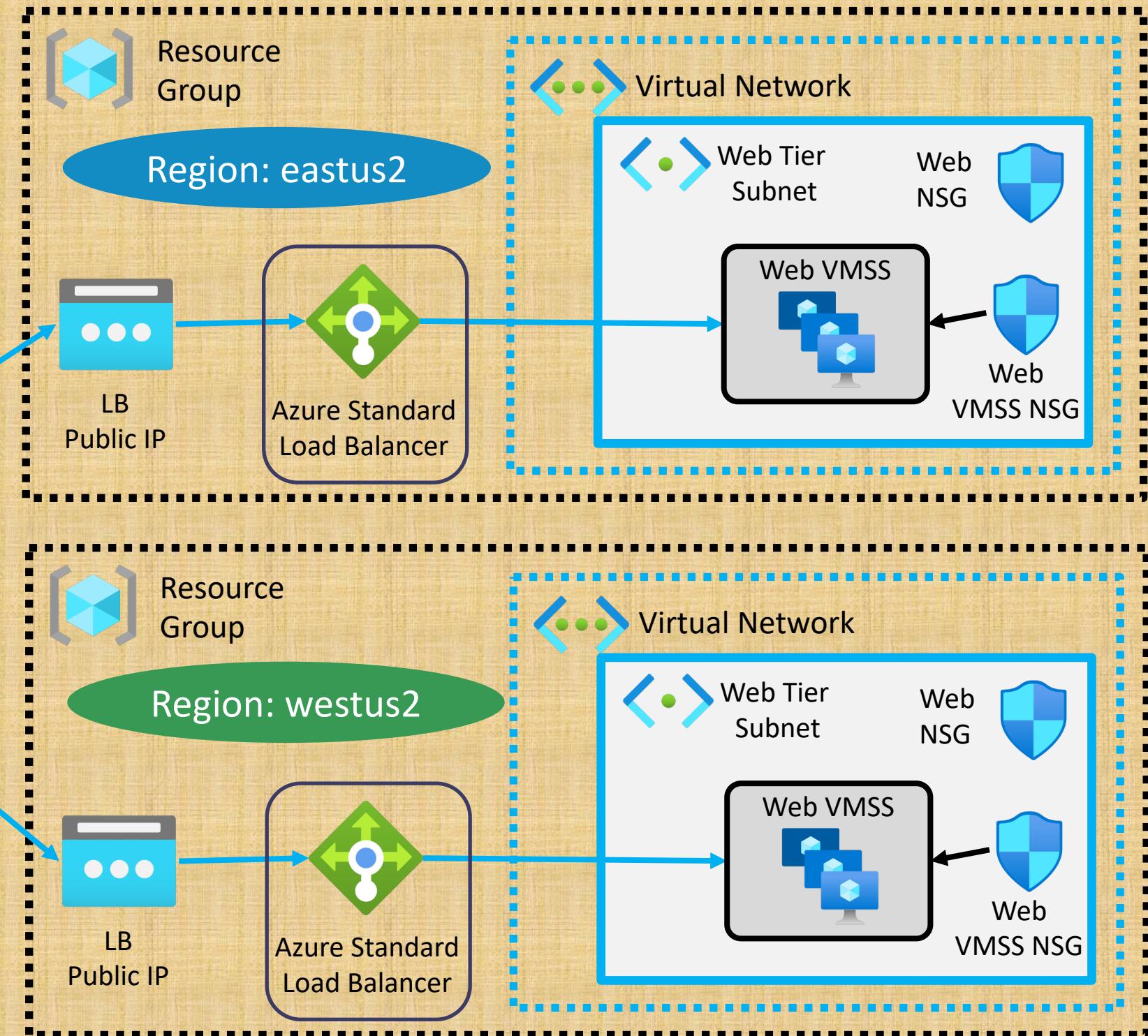


The `terraform_remote_state` data source retrieves the `root module output values` from some other Terraform configuration, using the latest state snapshot from the remote backend.

Azure – Traffic Manager



Terraform



Traffic Manager Endpoints

Home > Resource groups > tm-eastus2-hr-dev-rg-wquift > mytfdemo-wquift

mytfdemo-wquift | Endpoints

Traffic Manager profile

Search (Cmd+/) Add Refresh

Overview Activity log Access control (IAM) Tags Diagnose and solve problems

Search endpoints

Name ↑↓	Status ↑↓	Monitor status ↑↓	Type ↑↓	Weight ↑↓	...
tm-endpoint-project2-westus2	Enabled	Online	Azure endpoint	50	...
tm-endpoint-project1-eastus2	Enabled	Online	Azure endpoint	50	...

Terraform Backend – Three Projects

```
# Terraform State Storage to Azure Storage Container
backend "azurerm" {
  resource_group_name      = "terraform-storage-rg"
  storage_account_name     = "terraformstate201"
  container_name           = "tfstatefiles"
  key                      = "project-1-eastus2-terraform.tfstate"
}
```

Project-1 Terraform State File

```
# Terraform State Storage to Azure Storage Container
backend "azurerm" {
  resource_group_name      = "terraform-storage-rg"
  storage_account_name     = "terraformstate201"
  container_name           = "tfstatefiles"
  key                      = "project-2-westus2-terraform.tfstate"
}
```

Project-2 Terraform State File

```
# Terraform State Storage to Azure Storage Container
backend "azurerm" {
  resource_group_name      = "terraform-storage-rg"
  storage_account_name     = "terraformstate201"
  container_name           = "tfstatefiles"
  key                      = "project-3-traffic-manager-terraform.tfstate"
}
```

Project-3 Terraform State File

✓ project-1-eastus2-vmss

- > app-scripts
- > ssh-keys
- ↳ c1-versions.tf
- ↳ c2-generic-input-variables.tf
- ↳ c3-locals.tf
- ↳ c4-random-resources.tf
- ↳ c5-resource-group.tf
- ↳ c6-01-vnet-input-variables.tf
- ↳ c6-02-virtual-network.tf
- ↳ c6-03-web-subnet-and-nsg.tf
- ↳ c6-04-app-subnet-and-nsg.tf
- ↳ c6-05-db-subnet-and-nsg.tf
- ↳ c6-06-bastion-subnet-and-nsg.tf
- ↳ c6-07-vnet-outputs.tf

- ↳ c7-01-web-linux-vmss-input-variables.tf
- ↳ c7-02-web-linux-vmss-nsg-inline-basic.tf
- ↳ c7-03-web-linux-vmss-resource.tf
- ↳ c7-04-web-linux-vmss-outputs.tf
- ↳ c7-05-web-linux-vmss-autoscaling-default-profile.tf
- ↳ c7-06-web-linux-vmss-autoscaling-default-and-recurrence-profiles.tf
- ↳ c7-07-web-linux-vmss-autoscaling-default-recurrence-fixed-profiles.tf

- ↳ c8-01-bastion-host-input-variables.tf
- ↳ c8-02-bastion-host-linuxvm.tf
- ↳ c8-03-move-ssh-key-to-bastion-host.tf
- ↳ c8-04-AzureBastionService.tf

- ↳ c8-05-bastion-outputs.tf

- ↳ c9-01-web-loadbalancer-input-variables.tf
- ↳ c9-02-web-loadbalancer-resource.tf
- ↳ c9-03-web-loadbalancer-outputs.tf

- ↳ terraform.tfvars

Terraform Configs - Project 1 and 2

Project-1 Region: eastus2
Project-2 Region: westus2

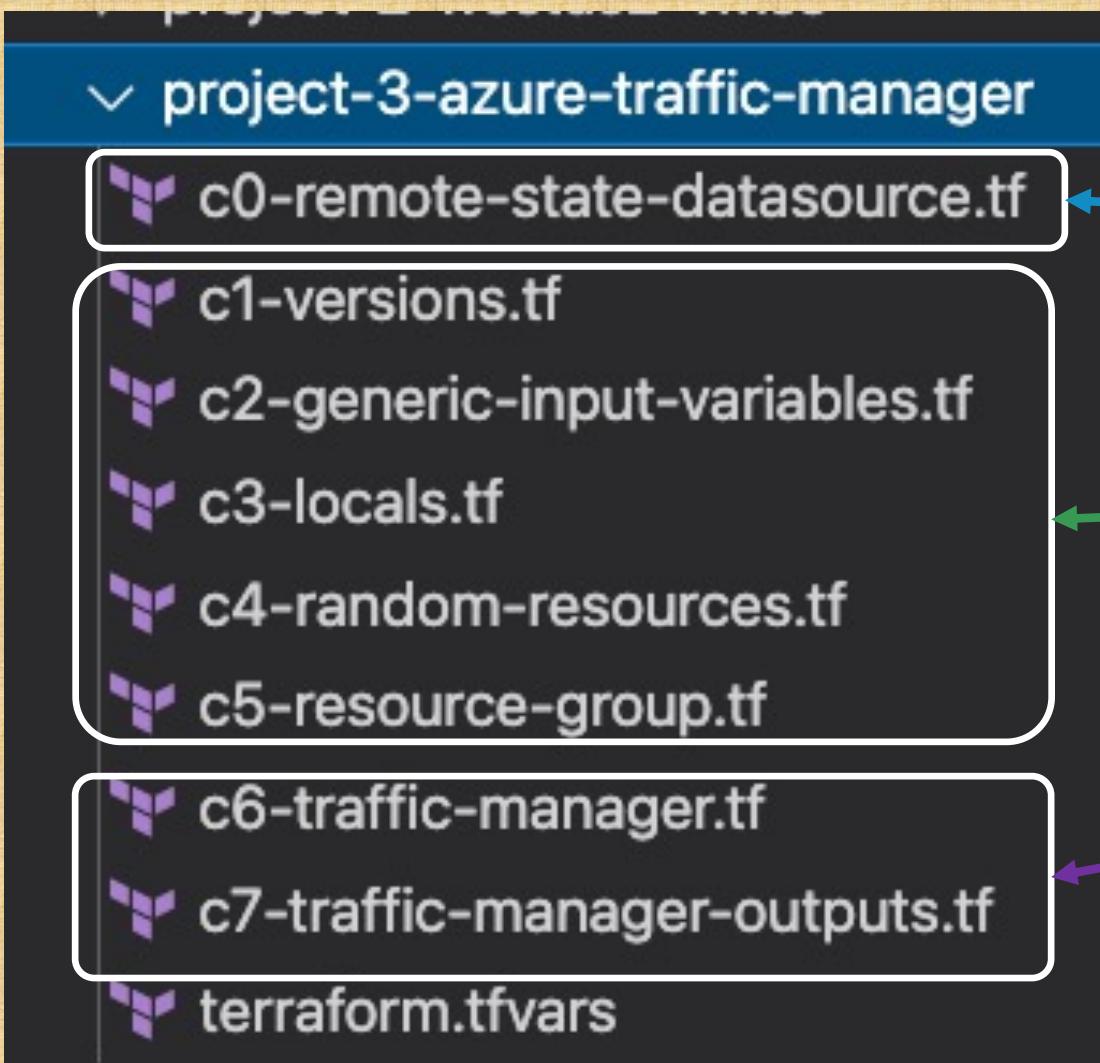
Azure Virtual Network with Subnets and Network Security Groups

Azure Virtual Machine Scale Set with Auto Scaling Profile

Azure Bastion Host Linux VM – Disabled
Azure Bastion Service - Disabled

Azure Standard Load Balancer

Terraform Configs – Project 3



Terraform Remote State Datasource to access Project-1 and Project-2 Public IP SLB

Traffic Manager Project Generic Resources (RG, locals, Input Variables, Random Resources)

Azure Traffic Manager Resource

Terraform Remote State Data source

```
# Project-1: East US2 Datasource
data "terraform_remote_state" "project1_eastus2" {
    backend = "azurerm"
    config = {
        resource_group_name      = "terraform-storage-rg"
        storage_account_name    = "terraformstate201"
        container_name           = "tfstatefiles"
        key                      = "project-1-eastus2-terraform.tfstate"
    }
}

# Project-2: West US2 Datasource
data "terraform_remote_state" "project2_westus2" {
    backend = "azurerm"
    config = {
        resource_group_name      = "terraform-storage-rg"
        storage_account_name    = "terraformstate201"
        container_name           = "tfstatefiles"
        key                      = "project-2-westus2-terraform.tfstate"
    }
}
```

^ Azure Traffic Manager and Terraform Remote State Datasource

5 lectures • 45min

- Step-01: Introduction to Azure Traffic Manager and Terraform Remote State Dataso
- ▶ Step-02: Review TF Configs and Execute TFCommands eastus2 and westus2 regions 09:27
- ▶ Step-03: Define Remote State Datasource TF Configs for Project-1 and 2 in Projec 08:48
- ▶ Step-04: Create TFConfig for Traffic Manager Profile 08:28
- ▶ Step-05: Create TFConfig for Traffic Manager Endpoints 08:13
- ▶ Step-06: Execute Project-3 TF Command, Verify and Clean-Up 3 projects 10:06

Time it takes to complete this Demo

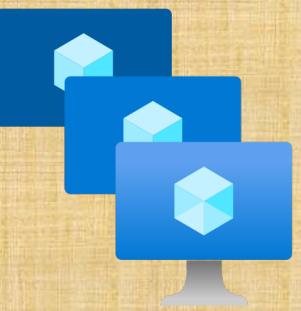
Real-World Demo



Application
Gateway

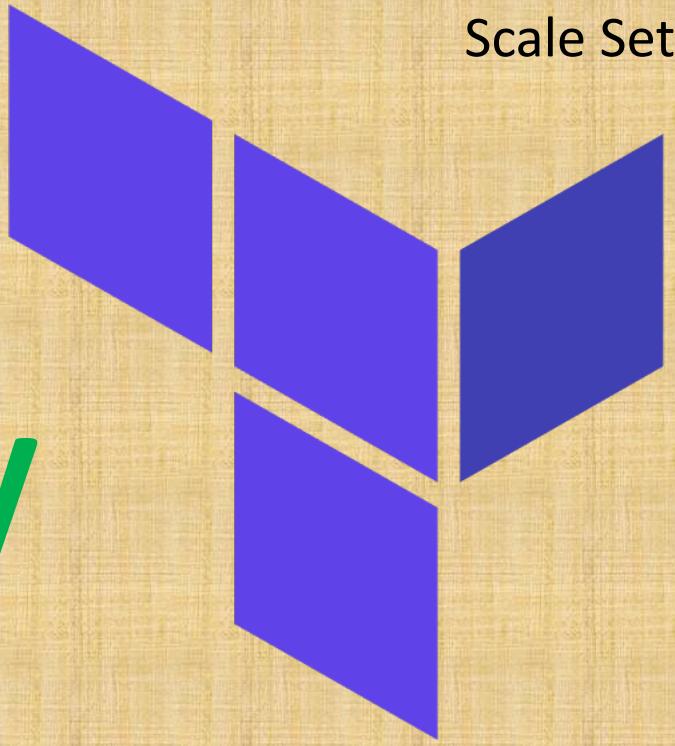


Real-World
Demo

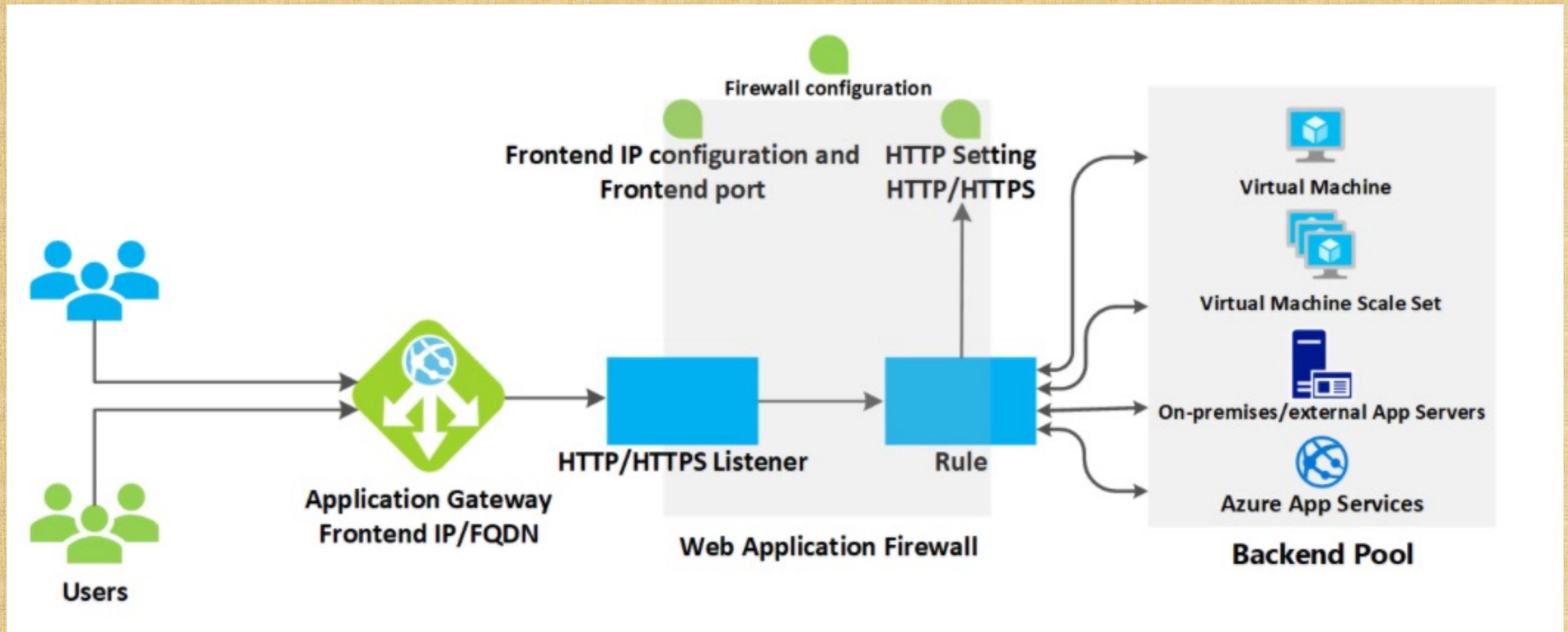


VM
Scale Sets

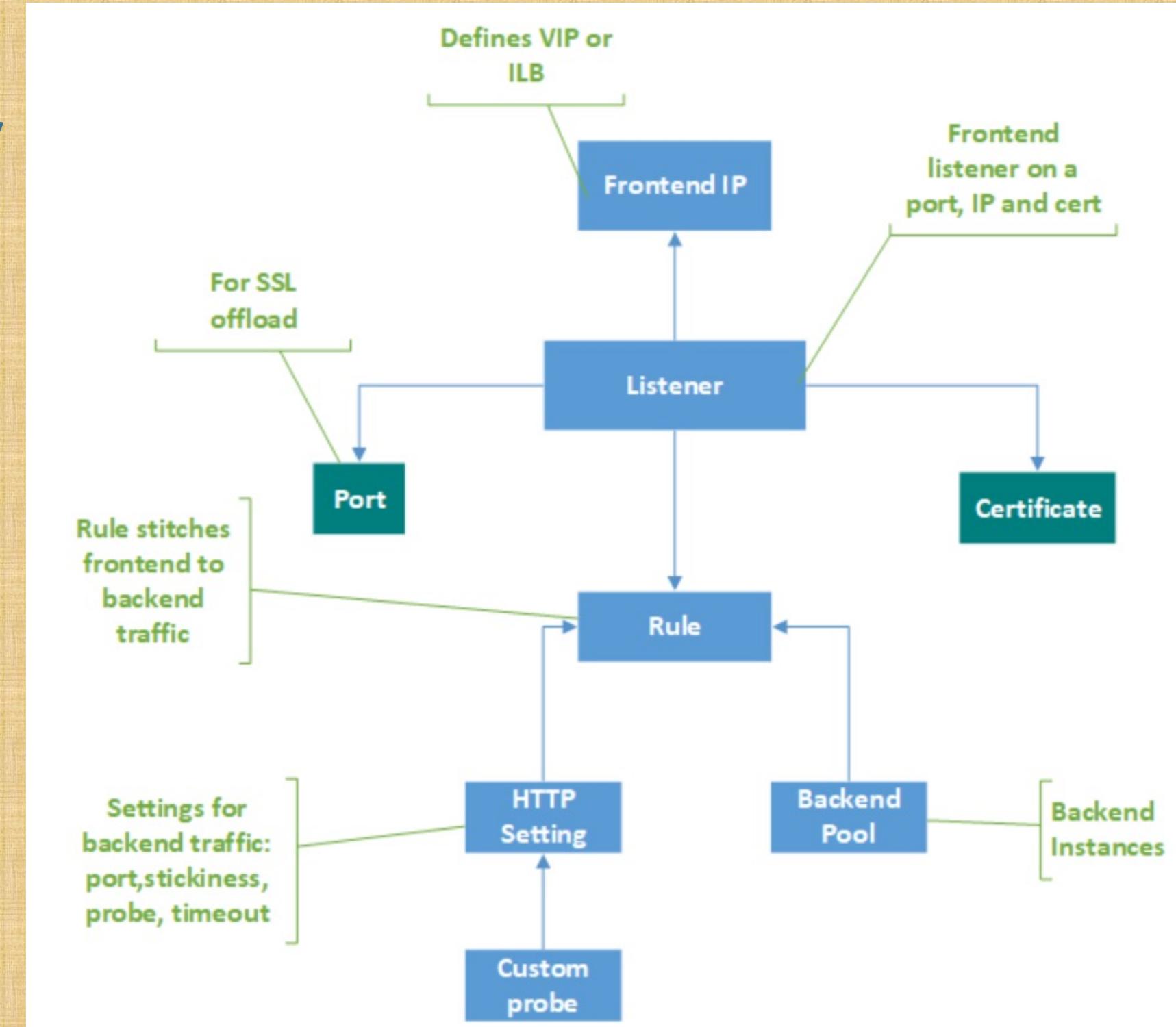
Azure Application Gateway using Azure Portal & Terraform



Azure Application Gateway



Azure Application Gateway Components



Create Azure Application
Gateway manually

AG – Backend Pools

AG – Frontend IP Configs

AG – Listeners

AG – HTTP Settings

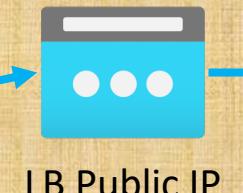
AG – Rules

AG – Health Probes



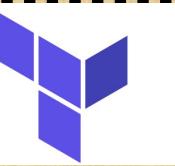
Users

Port 80



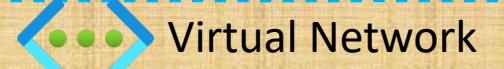
LB Public IP

Azure Application Gateway – using Azure Portal



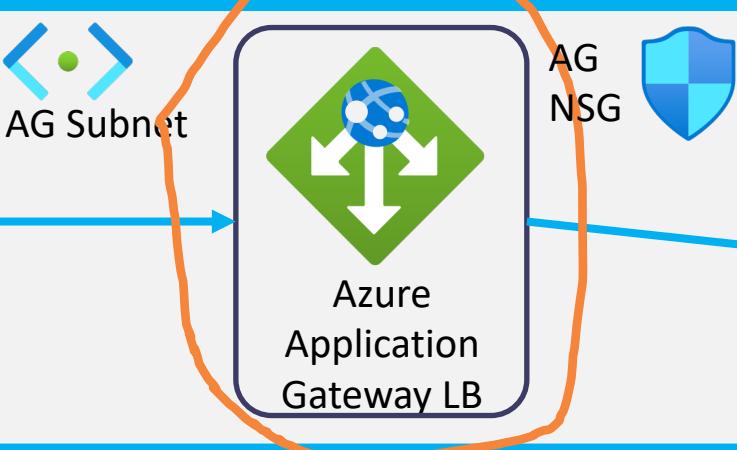
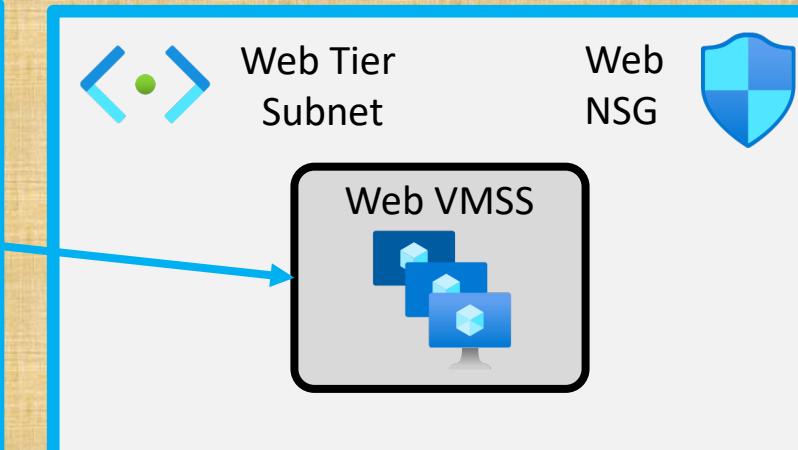
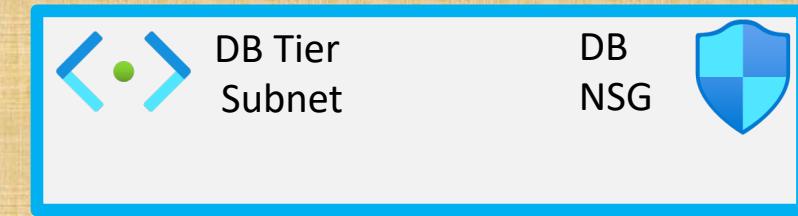
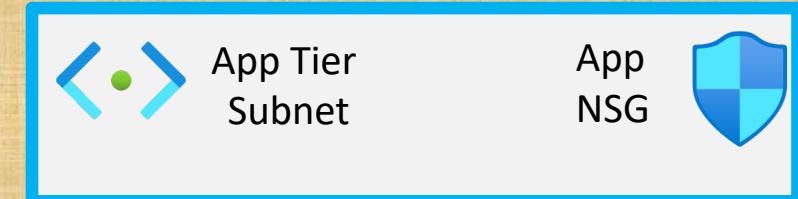
Resource
Group

Terraform



Virtual Network

Create AG using
Azure Portal



Create Azure Application Gateway using Terraform

AG – Backend Pools

AG – Frontend IP Configs

AG – Listeners

AG – HTTP Settings

AG – Rules

AG – Health Probes



Users

Port 80



LB Public IP

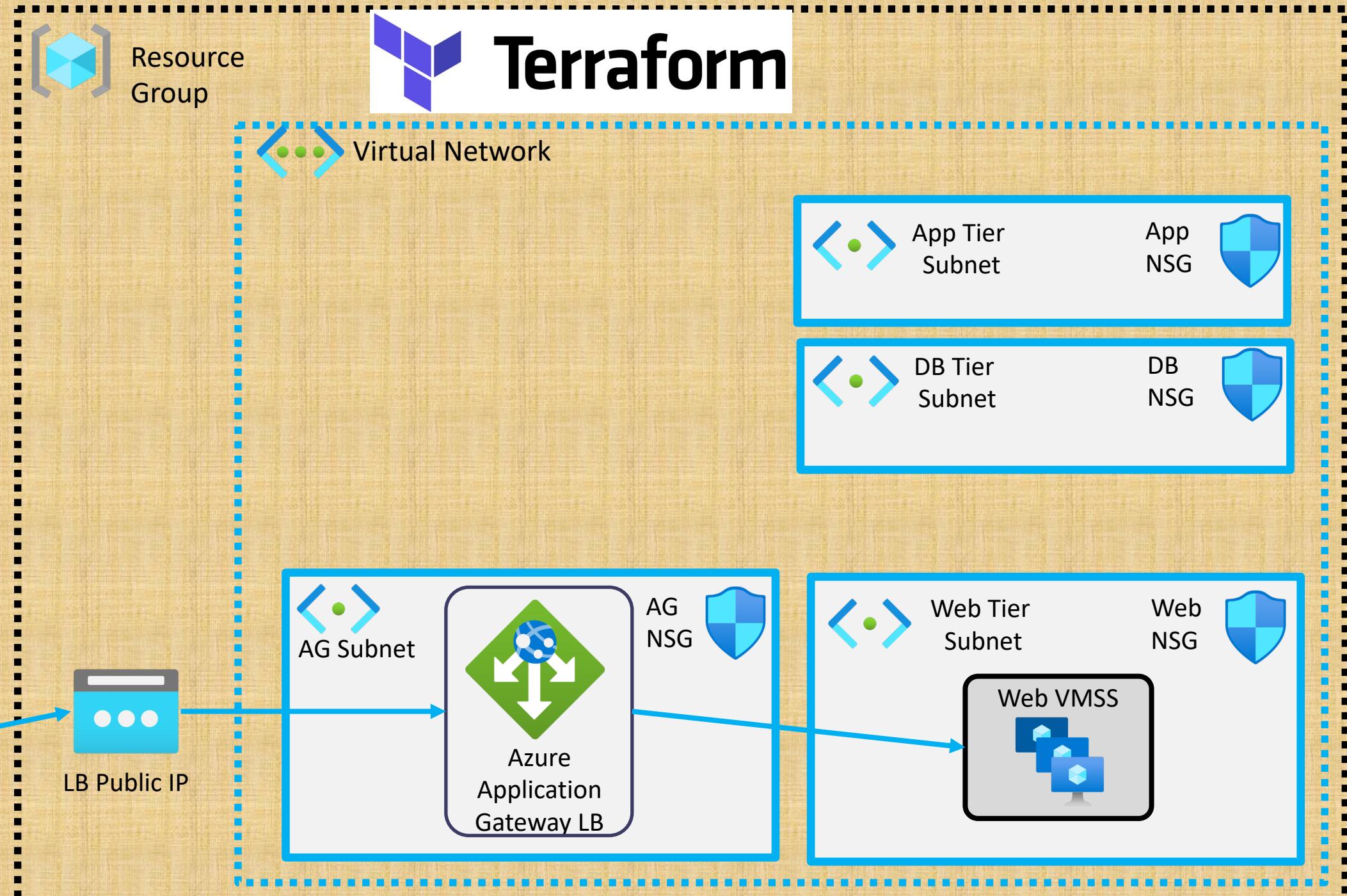
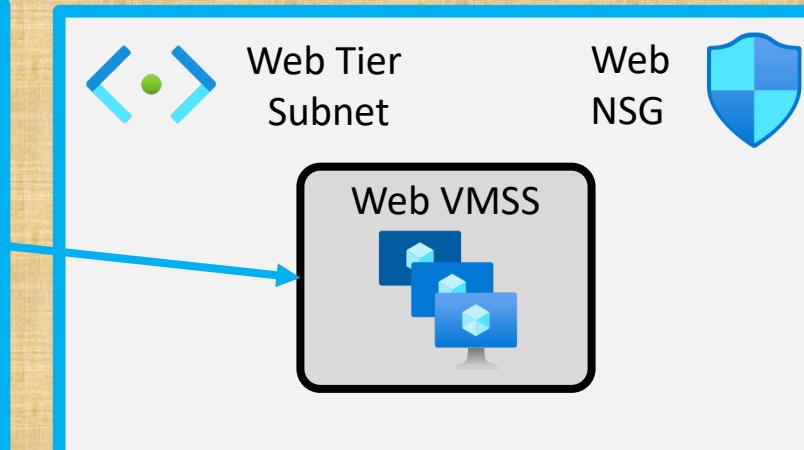
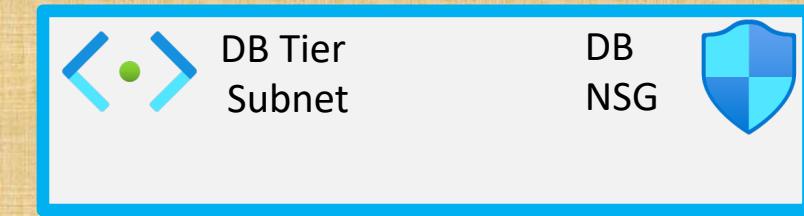
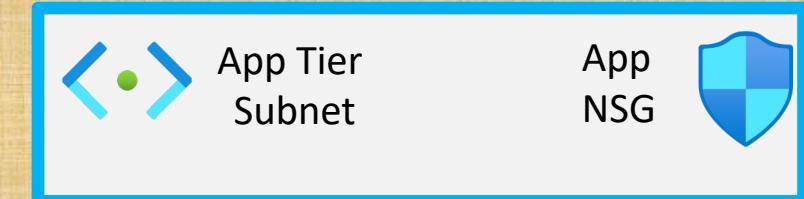
Azure Application Gateway – using Terraform



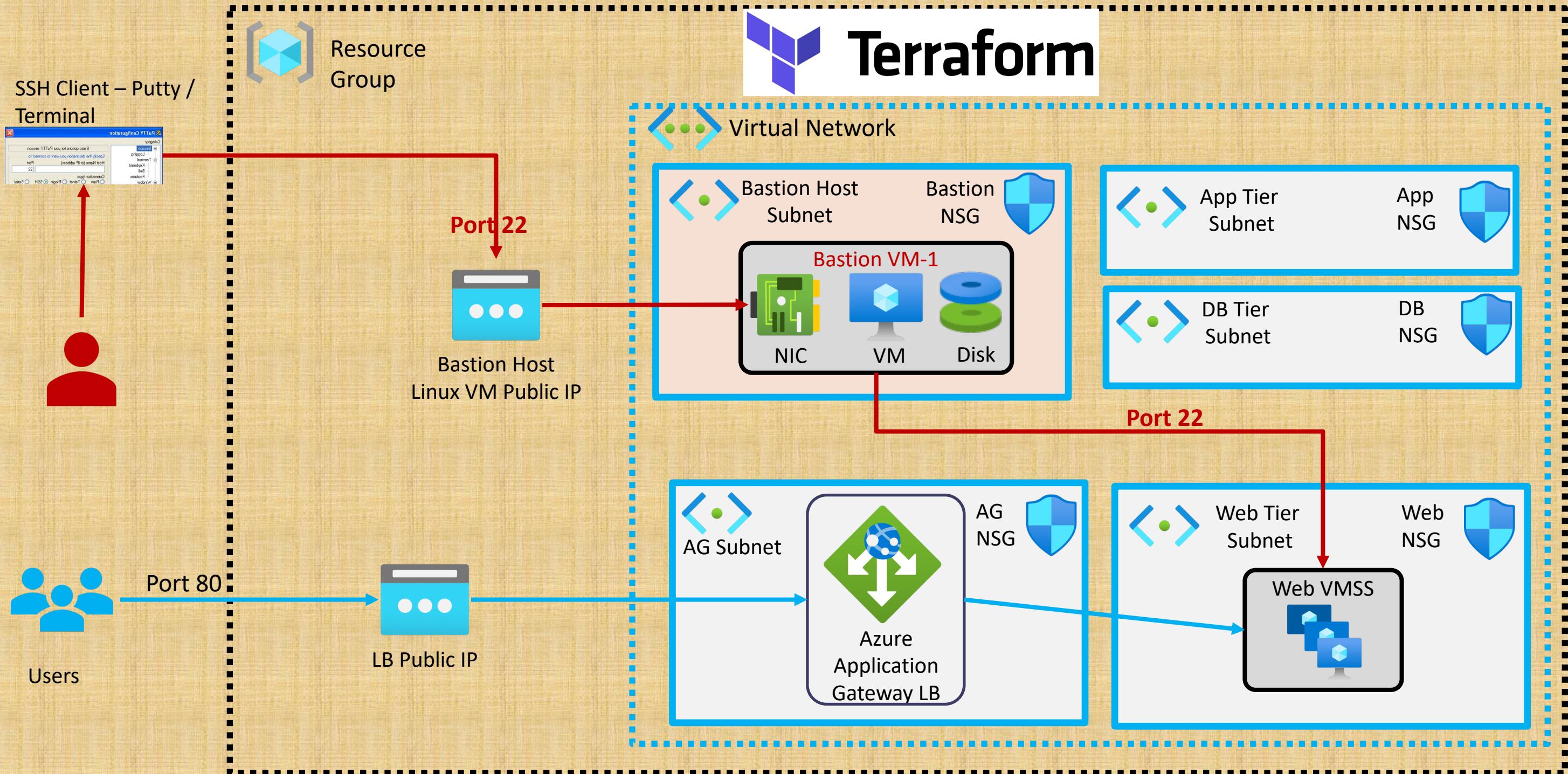
Terraform

Resource Group

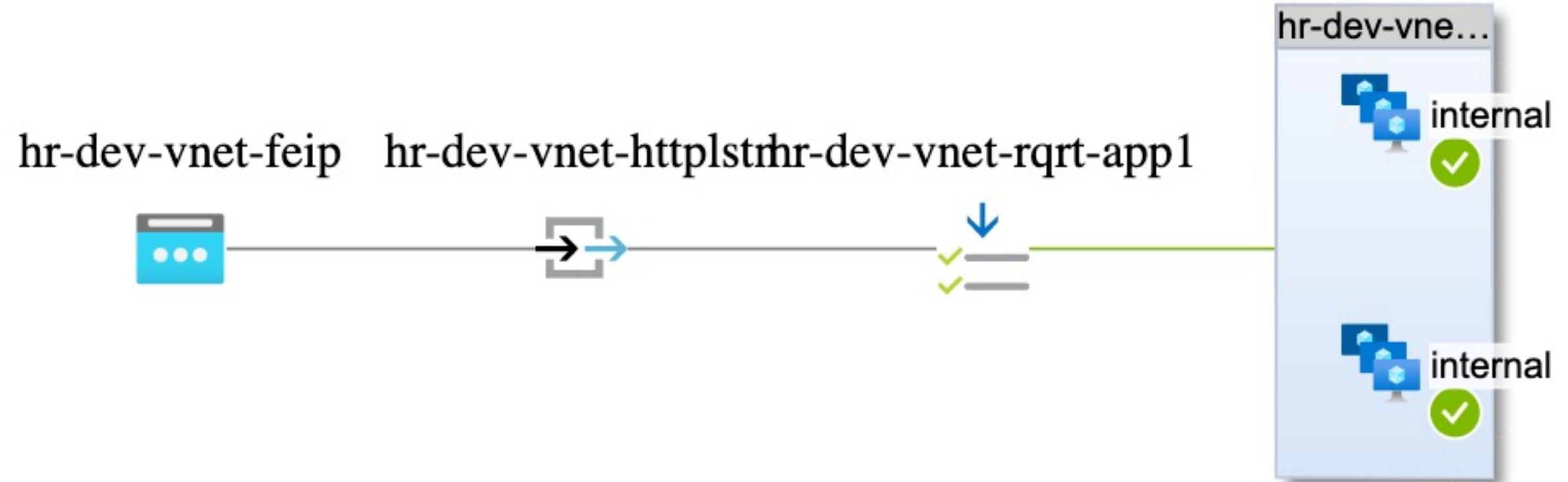
Virtual Network



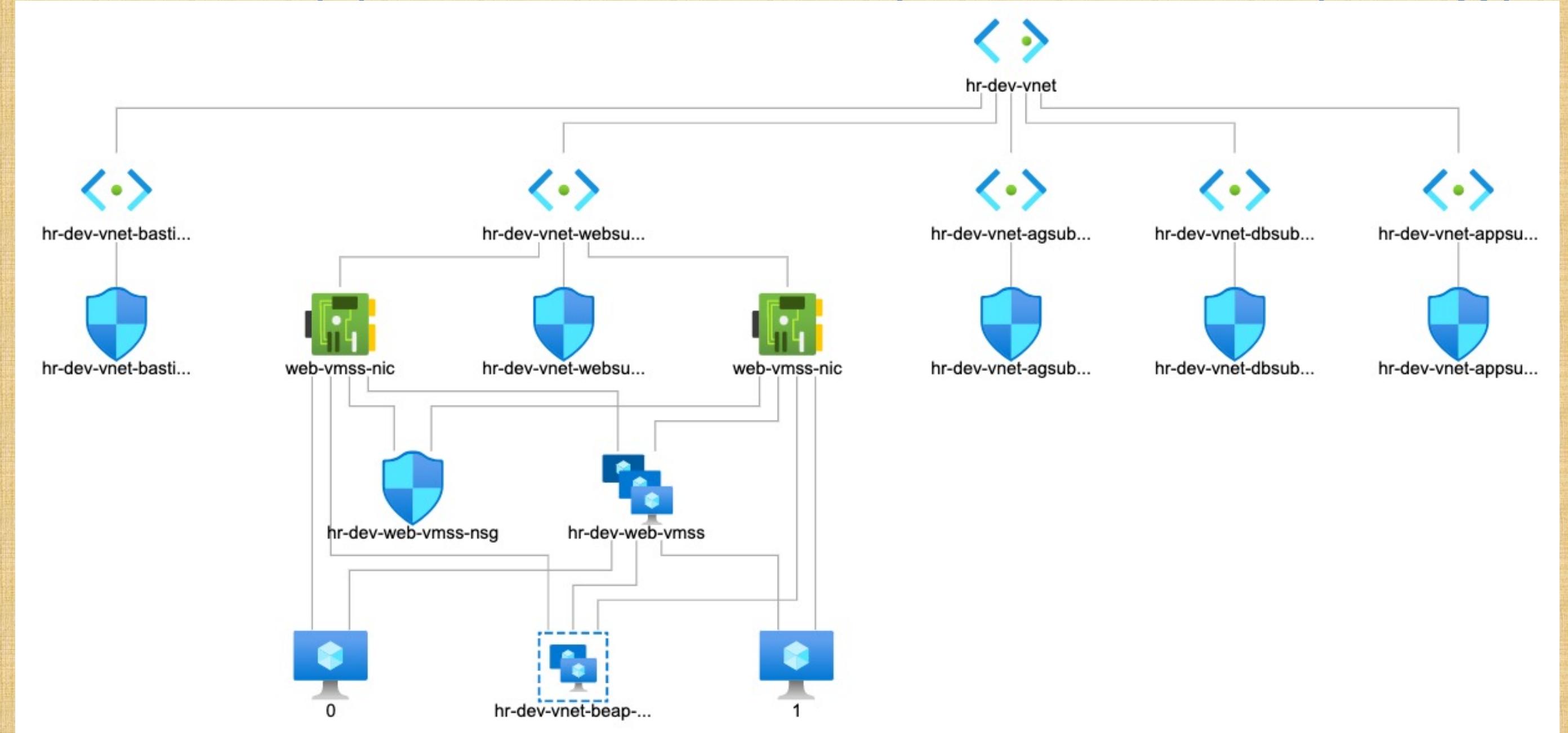
Azure Application Gateway



Azure Application Gateway - Topology



Azure Application Gateway + VMSS - Topology



```
✓ terraform-manifests
  > app-scripts
  > ssh-keys
  ✓ c1-versions.tf
  ✓ c2-generic-input-variables.tf
  ✓ c3-locals.tf
  ✓ c4-random-resources.tf
  ✓ c5-resource-group.tf
  ✓ c6-01-vnet-input-variables.tf
  ✓ c6-02-virtual-network.tf
  ✓ c6-03-web-subnet-and-nsg.tf
  ✓ c6-04-app-subnet-and-nsg.tf
  ✓ c6-05-db-subnet-and-nsg.tf
  ✓ c6-06-bastion-subnet-and-nsg.tf
  ✓ c6-07-ag-subnet-and-nsg.tf
  ✓ c6-08-vnet-outputs.tf
```

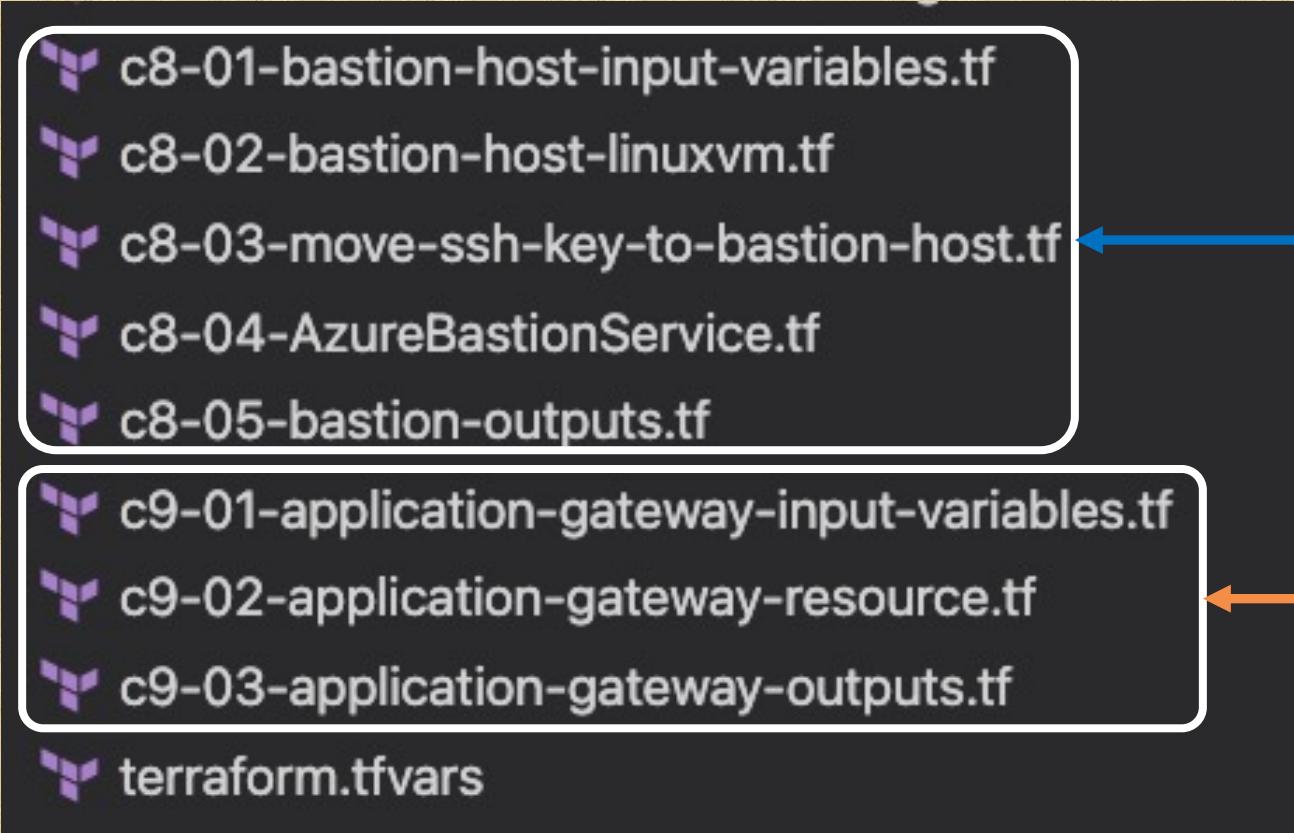
```
✓ c7-01-web-linux-vmss-input-variables.tf
  ✓ c7-02-web-linux-vmss-nsg-inline-basic.tf
  ✓ c7-03-web-linux-vmss-resource.tf
  ✓ c7-04-web-linux-vmss-outputs.tf
  ✓ c7-05-web-linux-vmss-autoscaling-default-profile.tf
  ✓ c7-06-web-linux-vmss-autoscaling-default-and-recurrence-profiles.tf
  ✓ c7-07-web-linux-vmss-autoscaling-default-recurrence-fixed-profiles.tf
```

Terraform Configs

Azure Virtual Network with Subnets and Network Security Groups

Azure Virtual Machine Scalesets with Autoscaling enabled

Terraform Configs



Azure Bastion Linux VM – Disabled
Azure Bastion Service - Disabled

Azure Application Gateway

^ Azure Application Gateway using Azure Portal

4 lectures • 40min

- Step-01: Introduction to Azure Application Gateway
- ▶ Step-02: Review TF Configs and Execute TF Commands to create VNET + VMSS + AG Su 11:37
- ▶ Step-03: Create Application Gateway using Azure Portal 12:07
- ▶ Step-04: Verif Application Gateway Features and Access Application using AG PIP 12:56
- ▶ Step-05: Destroy Resources 03:06

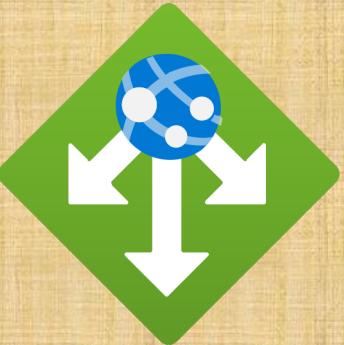
^ Azure Application Gateway using Terraform

4 lectures • 42min

- ▶ Step-01: Create Application Gateway TF Config - Part-1 12:50
- ▶ Step-02: Create Application Gateway TF Config - Part-2 12:51
- ▶ Step-03: Create Application Gateway TF Config - Part-3 07:18
- ▶ Step-04: Execute TF Commands, Verify AG Resources and Clean-Up 09:17

Time it takes to complete this Demo

Real-World Demo

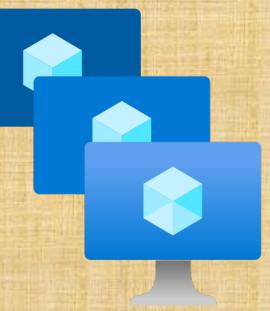
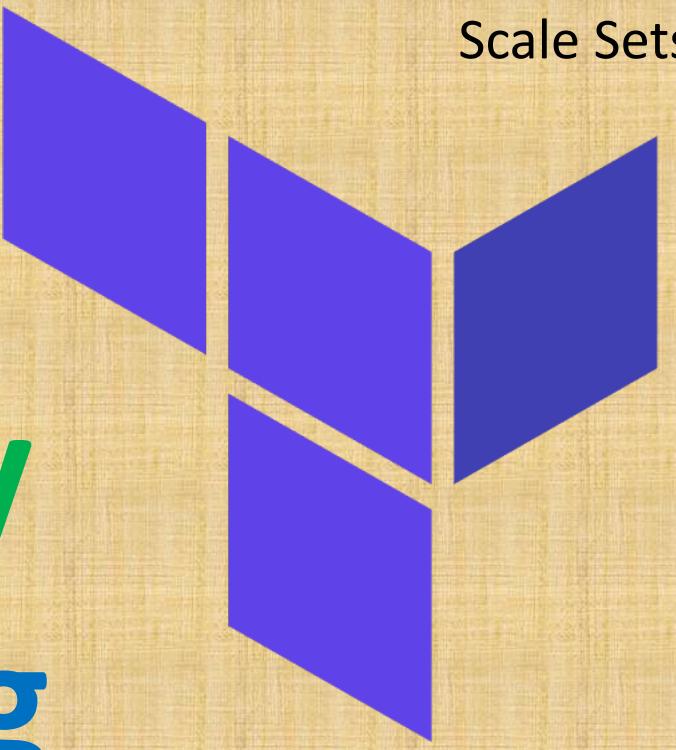


Application
Gateway



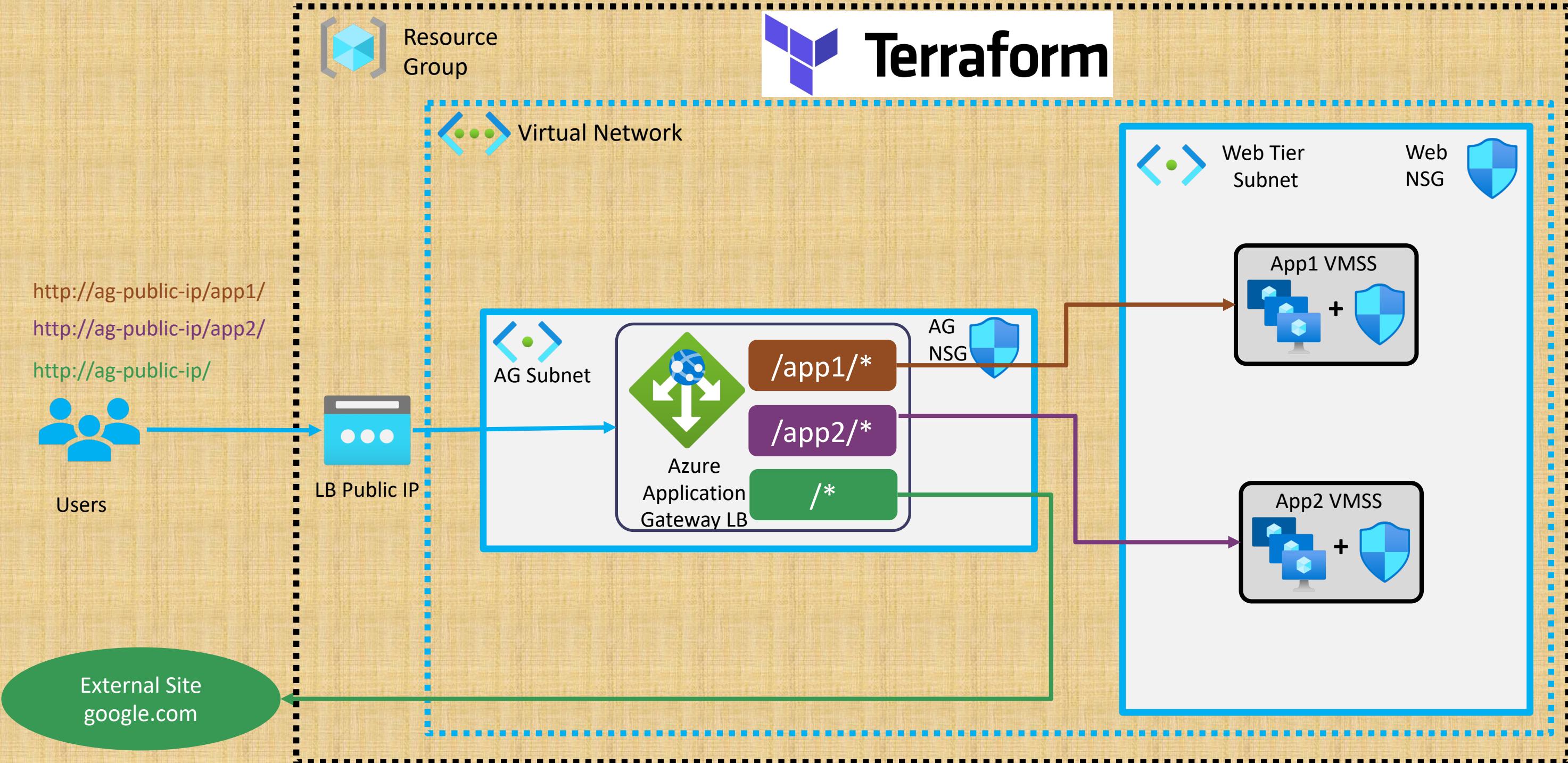
Real-World
Demo

Azure Application Gateway Context Path Routing

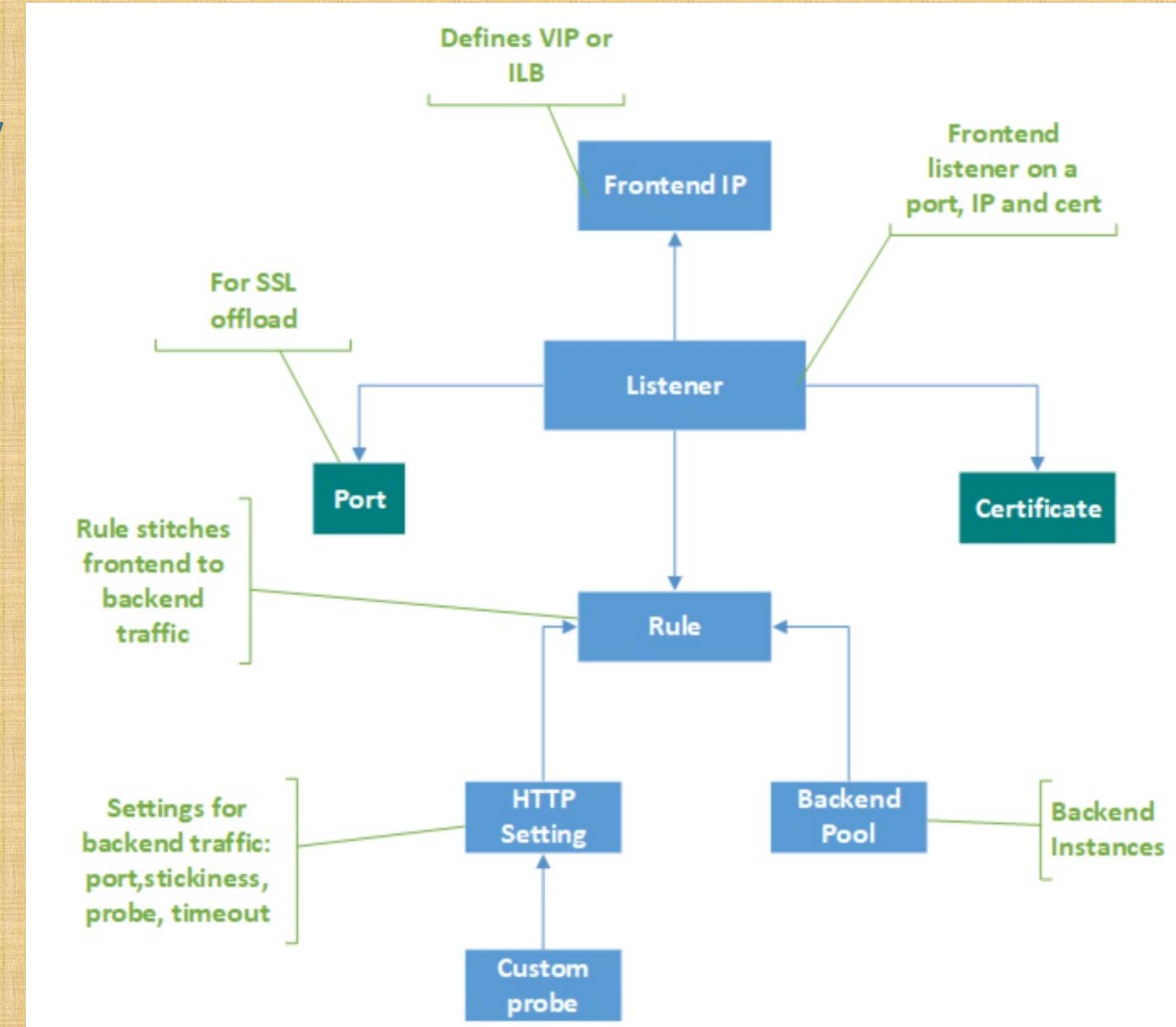


VM
Scale Sets

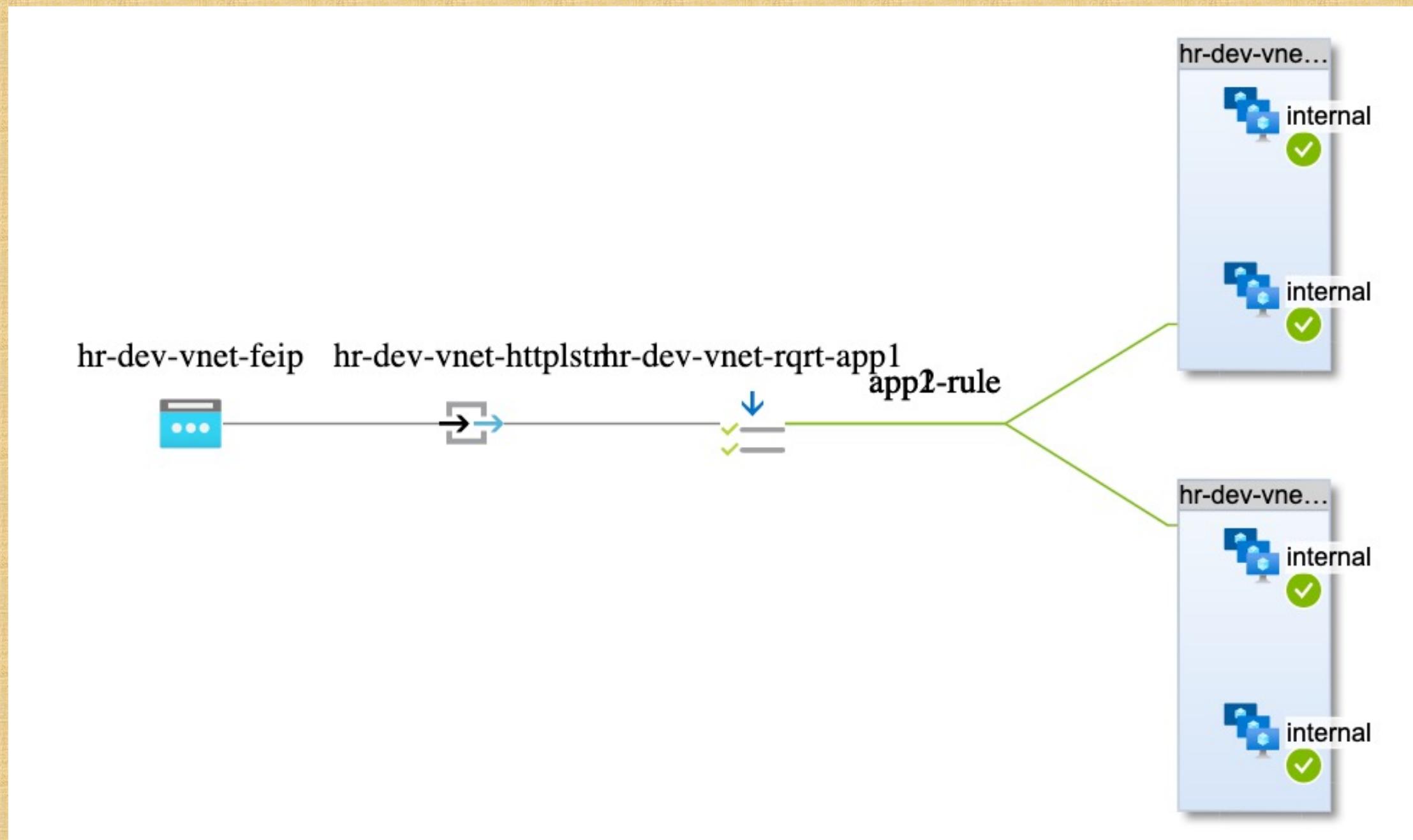
Azure Application Gateway – Context Path based Routing



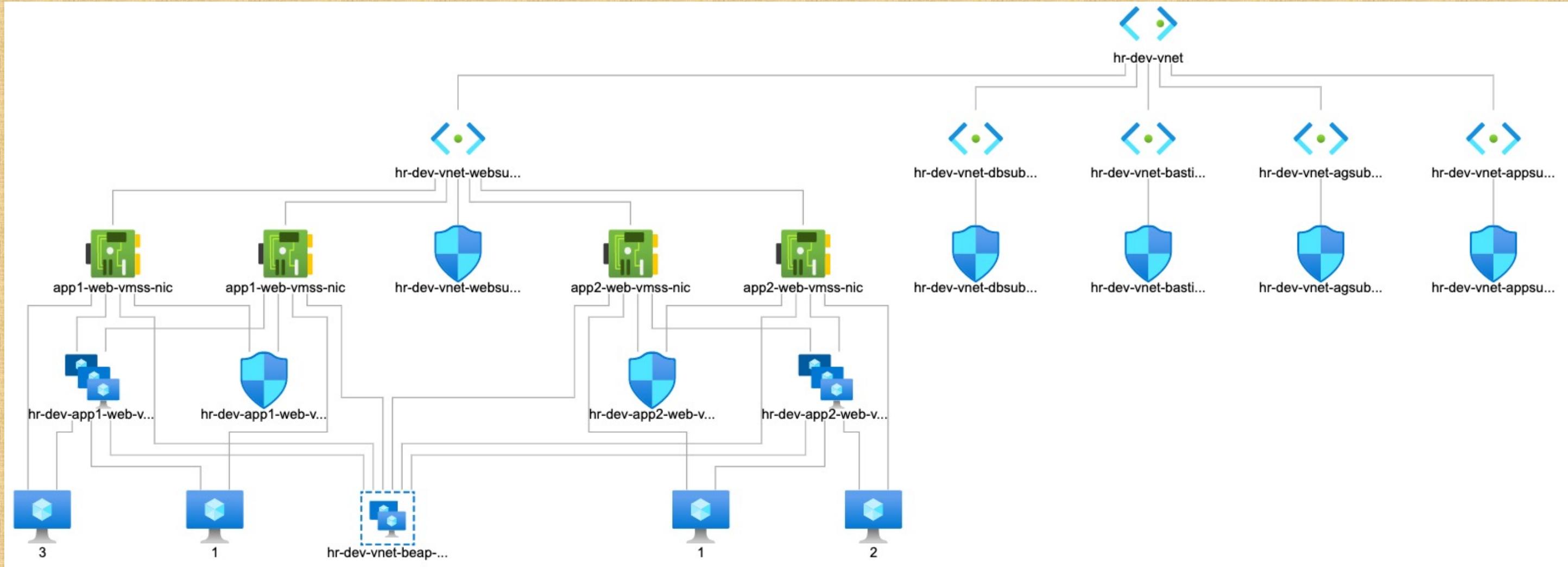
Azure Application Gateway Components



Azure Application Gateway Context Path Routing



Azure Application Gateway Context Path Routing



Path Based Routing

Root Context External Redirect

Context Path based Routing

hr-dev-vnet-rqrt-1

hr-dev-web-ag

* Listener * Backend targets

Choose a backend pool to which this routing rule will send traffic. You will also need to specify a set of HTTP settings that define the behavior of the routing rule.

Target type Backend pool Redirection

Redirection type Permanent

Redirection target Listener External site

Target URL * https://stacksimplify.com/azure-aks/azure-kubernetes-service-introduction/

Include query string Yes No

Include path Yes No

Path-based routing

You can route traffic from this rule's listener to different backend targets based on the URL path of the request. You can also apply a different set of HTTP settings based on the URL path.

Path based rules

Path	Target name	HTTP setting name	Backend pool	...
/app1/*	app1-rule	hr-dev-vnet-be-htst-app1	hr-dev-vnet-beap-app1	...
/app2/*	app2-rule	hr-dev-vnet-be-htst-app2	hr-dev-vnet-beap-app2	...

Path based Routing Rule

```
# Path based Routing Rule
request_routing_rule {
    name
    rule_type
    http_listener_name
    url_path_map_name
}
```

= local.request_routing_rule1_name
= "PathBasedRouting"
= local.listener_name
= local.url_path_map

Path based Routing Rule

```
# URL Path Map – Define Path based Routing
url_path_map {
    name = local.url_path_map
    default_redirect_configuration_name = local.redirect_configuration_name
    path_rule {
        name = "app1-rule"
        backend_address_pool_name optional, string
        backend_address_pool_name = local.backend_address_pool_name_app1
        backend_http_settings_name = local.http_setting_name_app1
    }
    path_rule {
        name = "app2-rule"
        paths = ["/app2/*"]
        backend_address_pool_name = local.backend_address_pool_name_app2
        backend_http_settings_name = local.http_setting_name_app2
    }
}
```

Default Root Context - Redirection

```
# Default Root Context (/ - Redirection Config)
redirect_configuration {
    name = local.redirect_configuration_name
    redirect_type = "Permanent"
    target_url =
}
```

terraform-manifests

- > app-scripts
- > ssh-keys
- ↳ c1-versions.tf
- ↳ c2-generic-input-variables.tf
- ↳ c3-locals.tf
- ↳ c4-random-resources.tf
- ↳ c5-resource-group.tf

- ↳ c6-01-vnet-input-variables.tf
- ↳ c6-02-virtual-network.tf
- ↳ c6-03-web-subnet-and-nsg.tf
- ↳ c6-04-app-subnet-and-nsg.tf
- ↳ c6-05-db-subnet-and-nsg.tf
- ↳ c6-06-bastion-subnet-and-nsg.tf
- ↳ c6-07-ag-subnet-and-nsg.tf
- ↳ c6-08-vnet-outputs.tf

- ↳ c7-01-web-linux-vmss-input-variables.tf
- ↳ c7-02-web-linux-vmss-app1-nsg-inline-basic.tf
- ↳ c7-03-web-linux-vmss-app1-resource.tf
- ↳ c7-04-web-linux-vmss-app1-autoscaling-default-profile.tf
- ↳ c7-05-web-linux-vmss-app2-nsg-inline-basic.tf
- ↳ c7-06-web-linux-vmss-app2-resource.tf
- ↳ c7-07-web-linux-vmss-app2-autoscaling-default-profile.tf
- ↳ c7-08-web-linux-vmss-outputs.tf

- ↳ c8-01-bastion-host-input-variables.tf
- ↳ c8-02-bastion-host-linuxvm.tf
- ↳ c8-03-move-ssh-key-to-bastion-host.tf
- ↳ c8-04-AzureBastionService.tf
- ↳ c8-05-bastion-outputs.tf

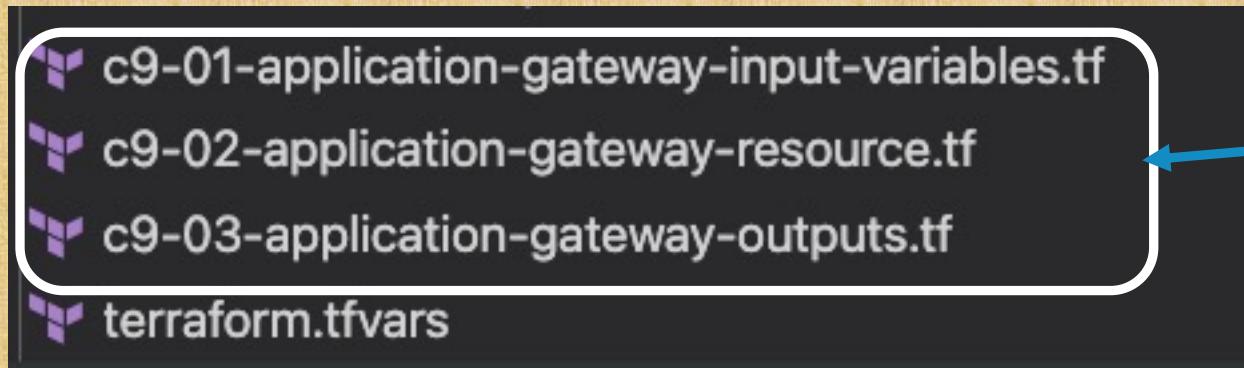
Terraform Configs

Azure Virtual Network with Subnets and Network Security Groups

Azure Virtual Machine Scalesets with Autoscaling enabled

Azure Bastion Linux VM – Disabled
Azure Bastion Service - Disabled

Terraform Configs



Azure Application Gateway

^ Azure Application Gateway Context Path Routing using Terraform

- Step-01: Introduction to AG Context Path based Routing
- Step-02: Review App1 and App2 VMSS TF Configs 11:57
- Step-03: Create App2Backend and Routing Rule TF Configs in AG Resource 08:19
- Step-04: Create Url Path Map, Redirect Config and VMSS-AG Integration TF Configs 10:45
- Step-05: Execute TF Commands, Verify AG Path Routing and CleanUp 12:31

4 lectures • 44min



Time it takes to complete this Demo

Real-World Demo

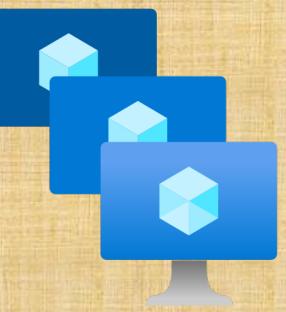


Application
Gateway

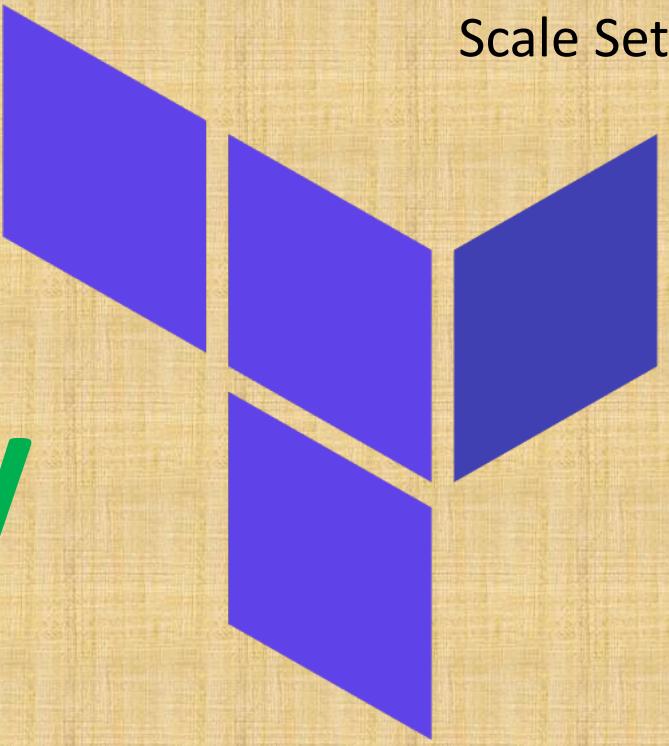


Real-World
Demo

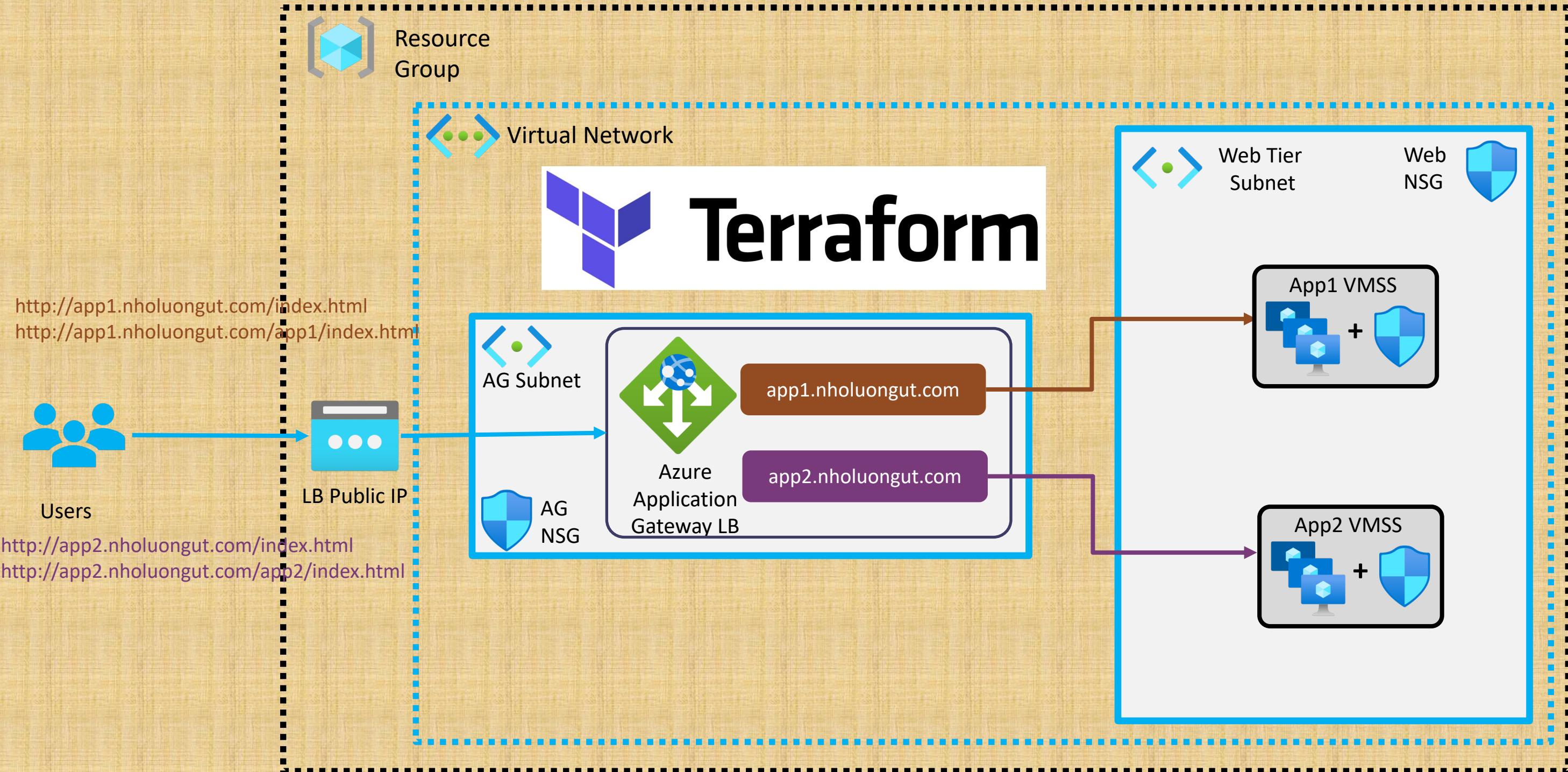
Azure Application Gateway Multisite Hosting



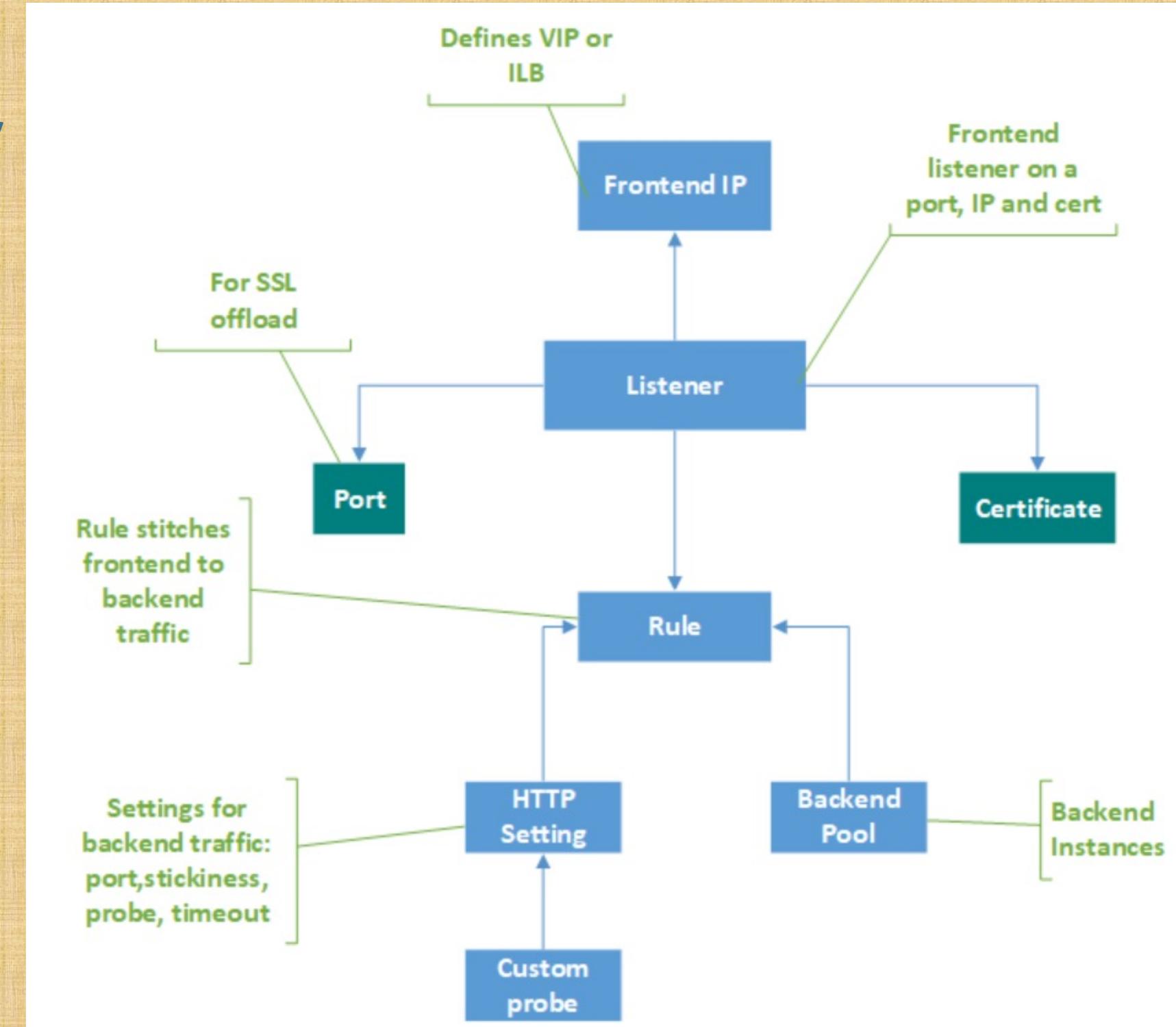
VM
Scale Sets



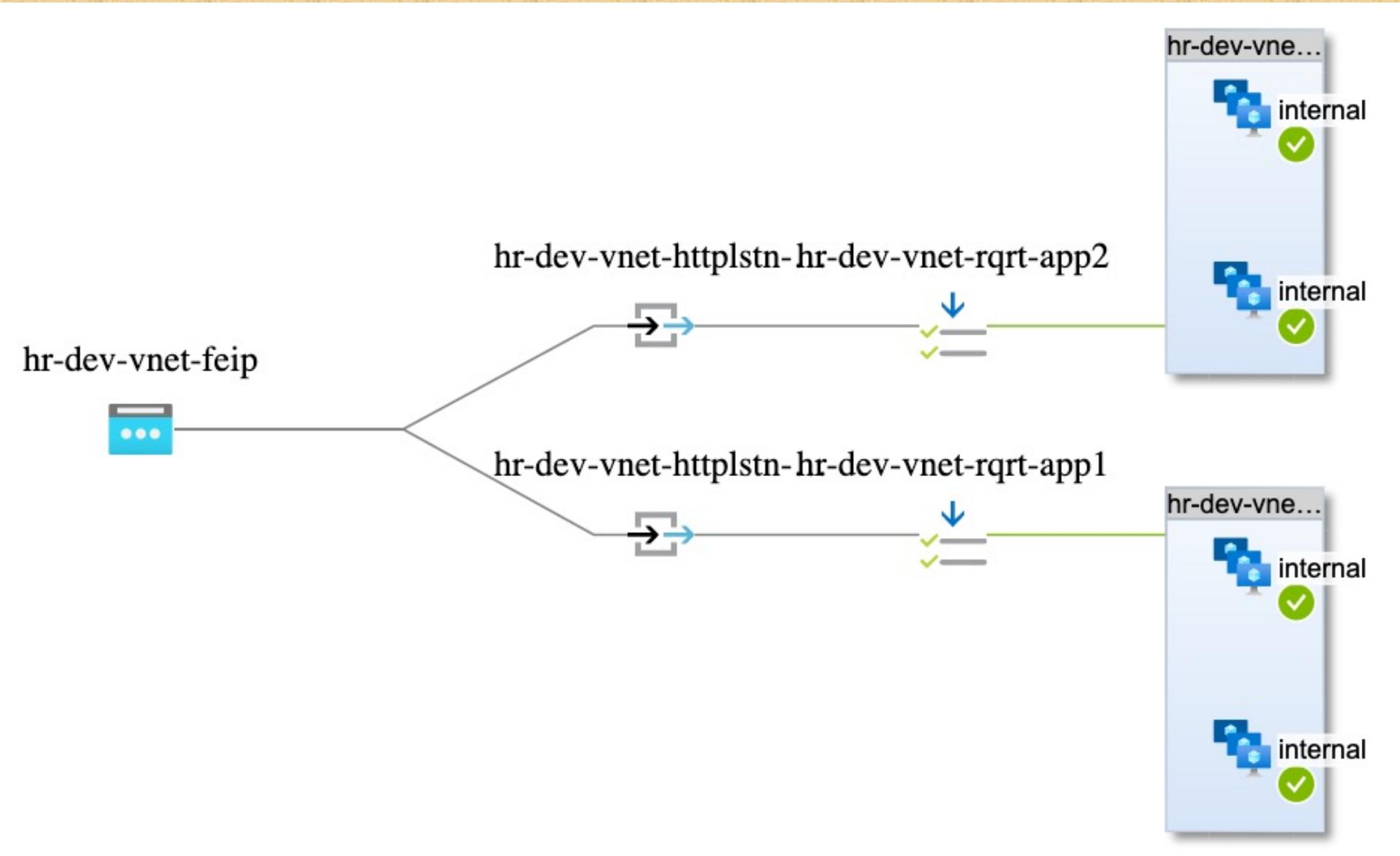
Azure Application Gateway – Multisite Hosting



Azure Application Gateway Components



Azure Application Gateway Multisite Hosting



Azure Application Gateway - Routing Rules

hr-dev-vnet-rqrt-app1

hr-dev-web-ag

Configure a routing rule to send traffic from a given frontend IP address to contain a listener and at least one backend target.

Rule name

hr-dev-vnet-rqrt-app1

* Listener * Backend targets

A listener "listens" on a specified port and IP address for traffic that uses the application gateway will apply this routing rule.

Listener *

hr-dev-vnet-httplstn-app1

hr-dev-vnet-rqrt-app2

hr-dev-web-ag

Configure a routing rule to send traffic from a given frontend IP address to contain a listener and at least one backend target.

Rule name

hr-dev-vnet-rqrt-app2

* Listener * Backend targets

A listener "listens" on a specified port and IP address for traffic that uses the application gateway will apply this routing rule.

Listener *

hr-dev-vnet-httplstn-app2

Azure Application Gateway - Routing Rules

hr-dev-vnet-rqrt-app1

hr-dev-web-ag

Configure a routing rule to send traffic from a given frontend IP address to one or more backend targets. A routing rule must contain a listener and at least one backend target.

Rule name

hr-dev-vnet-rqrt-app1

* Listener * Backend targets

Choose a backend pool to which this routing rule will send traffic. You will also define the behavior of the routing rule.

Target type

Backend pool Redirection

Backend target * ⓘ

hr-dev-vnet-beap-app1

HTTP settings * ⓘ

hr-dev-vnet-be-htst-app1

hr-dev-vnet-rqrt-app2

hr-dev-web-ag

Configure a routing rule to send traffic from a given frontend IP address to one or more backend targets. A routing rule must contain a listener and at least one backend target.

Rule name

hr-dev-vnet-rqrt-app2

* Listener * Backend targets

Choose a backend pool to which this routing rule will send traffic. You will also define the behavior of the routing rule.

Target type

Backend pool Redirection

hr-dev-vnet-beap-app2

hr-dev-vnet-be-htst-app2

Azure Application Gateway - Routing Rules

```
# Routing Rule - app1.terraformguru.com
request_routing_rule {
    name                      = local.request_routing_rule_name_app1
    rule_type                 = "Basic"
    http_listener_name        = local.listener_name_app1
    backend_address_pool_name = local.backend_address_pool_name_app1
    backend_http_settings_name = local.http_setting_name_app1
}

# Routing Rule - app2.terraformguru.com
request_routing_rule {
    name                      = local.request_routing_rule_name_app2
    rule_type                 = "Basic"
    http_listener_name        = local.listener_name_app2
    backend_address_pool_name = local.backend_address_pool_name_app2
    backend_http_settings_name = local.http_setting_name_app2
}
```

Azure Application Gateway – Health Probes

Home > Load balancing - help me choose (Preview) > hr-dev-web-ag

hr-dev-web-ag | Health probes

Application gateway

Search (Cmd+/) << + Add ⏪ Refresh ⏷ Delete

Web application firewall

Backend pools

HTTP settings

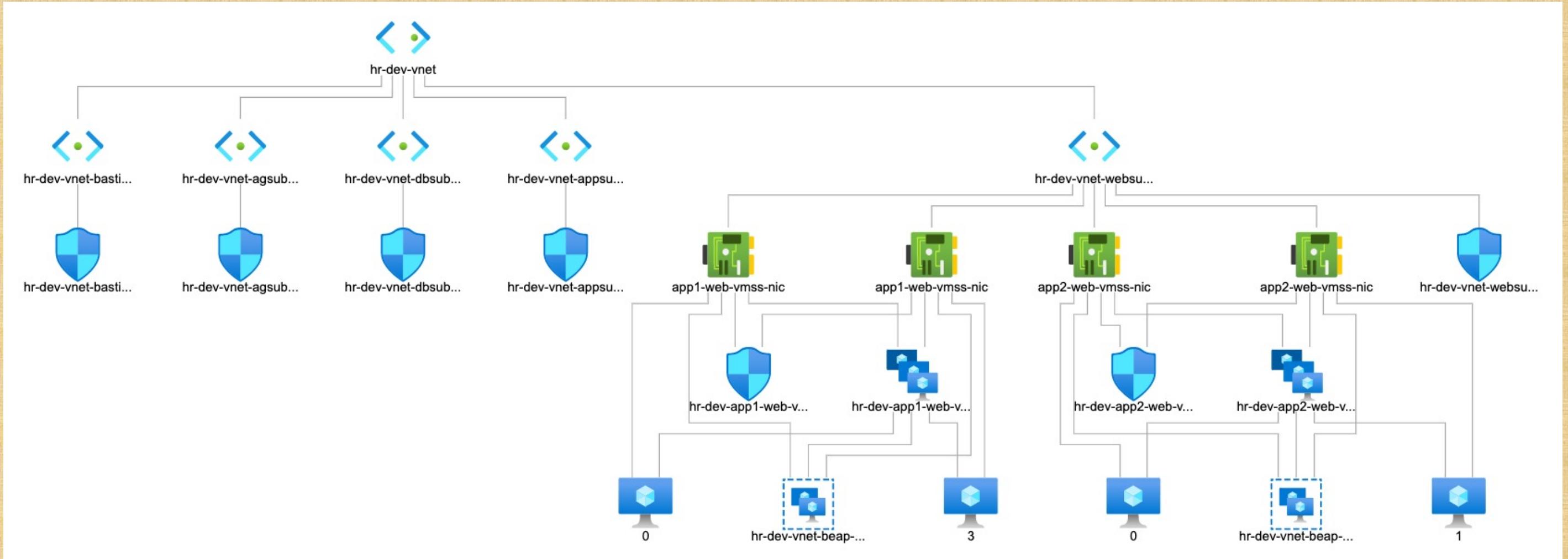
Frontend IP configurations

SSL settings (Preview)

Search probes

Name	Protocol	Host	Path	Timeout (seconds)	⋮
<input type="checkbox"/> hr-dev-vnet-be-probe-app1	Http	127.0.0.1	/app1/status.html	30	⋮
<input type="checkbox"/> hr-dev-vnet-be-probe-app2	Http	127.0.0.1	/app2/status.html	30	⋮

VMSS Topology App1 and App2



terraform-manifests

- > app-scripts
- > ssh-keys
- ↳ c1-versions.tf
- ↳ c2-generic-input-variables.tf
- ↳ c3-locals.tf
- ↳ c4-random-resources.tf
- ↳ c5-resource-group.tf

- ↳ c6-01-vnet-input-variables.tf
- ↳ c6-02-virtual-network.tf
- ↳ c6-03-web-subnet-and-nsg.tf
- ↳ c6-04-app-subnet-and-nsg.tf
- ↳ c6-05-db-subnet-and-nsg.tf
- ↳ c6-06-bastion-subnet-and-nsg.tf
- ↳ c6-07-ag-subnet-and-nsg.tf
- ↳ c6-08-vnet-outputs.tf

- ↳ c7-01-web-linux-vmss-input-variables.tf
- ↳ c7-02-web-linux-vmss-app1-nsg-inline-basic.tf
- ↳ c7-03-web-linux-vmss-app1-resource.tf
- ↳ c7-04-web-linux-vmss-app1-autoscaling-default-profile.tf
- ↳ c7-05-web-linux-vmss-app2-nsg-inline-basic.tf
- ↳ c7-06-web-linux-vmss-app2-resource.tf
- ↳ c7-07-web-linux-vmss-app2-autoscaling-default-profile.tf
- ↳ c7-08-web-linux-vmss-outputs.tf

- ↳ c8-01-bastion-host-input-variables.tf
- ↳ c8-02-bastion-host-linuxvm.tf
- ↳ c8-03-move-ssh-key-to-bastion-host.tf
- ↳ c8-04-AzureBastionService.tf
- ↳ c8-05-bastion-outputs.tf

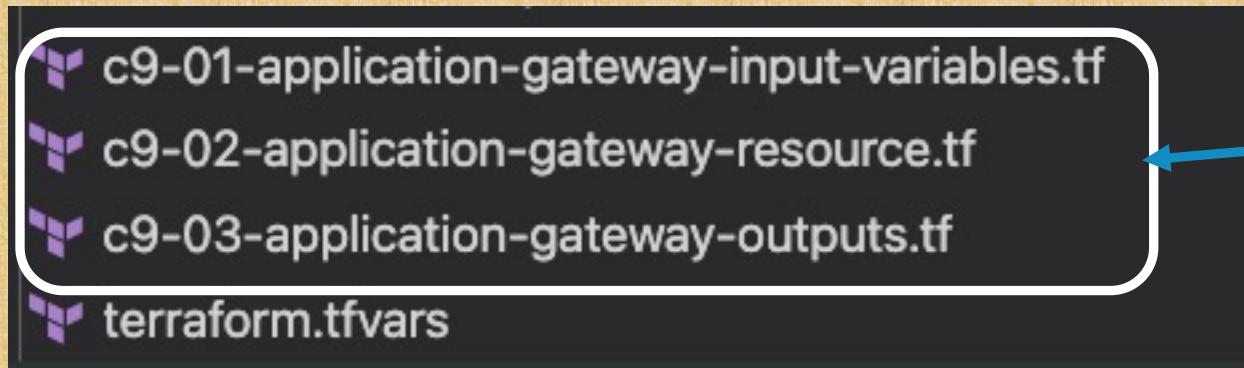
Terraform Configs

Azure Virtual Network with Subnets and Network Security Groups

Azure Virtual Machine Scalesets with Autoscaling enabled

Azure Bastion Linux VM – Disabled
Azure Bastion Service - Disabled

Terraform Configs



Azure Application Gateway

^ Azure Application Gateway Multisite Hosting using Terraform

- Step-01: Introduction to AG Multisite Hosting
- ▶ Step-02: Create TF Configs for Two Listeners and Two Routings
for App1 and App2
- ▶ Step-03: Execute TF Commands, Verify AG Multisite Hosting
and CleanUp

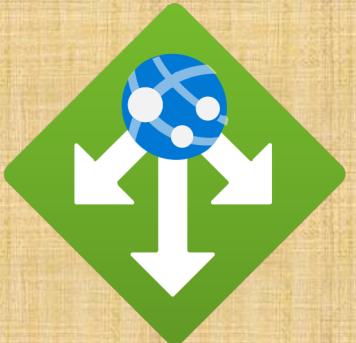
2 lectures • 18min

08:53

09:14

Time it
takes to
complete
this Demo

Real-World
Demo



Application
Gateway



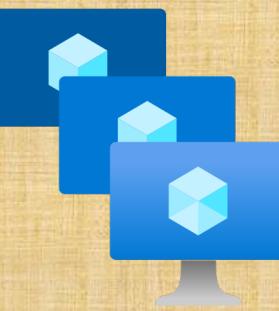
Real-World
Demo

Azure Application Gateway

Self Signed SSL

Http toHttps Redirect

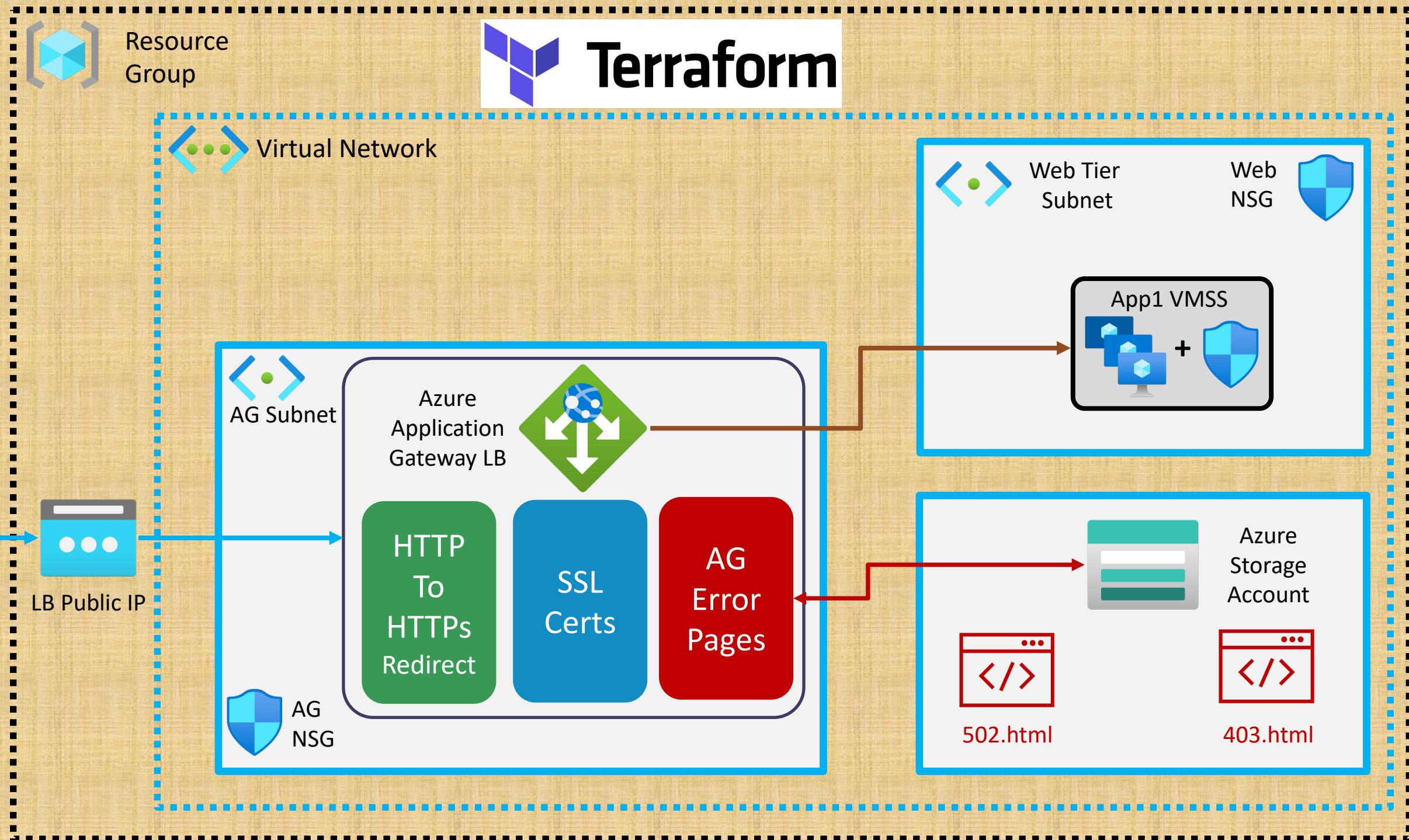
Error Pages 502 and 403



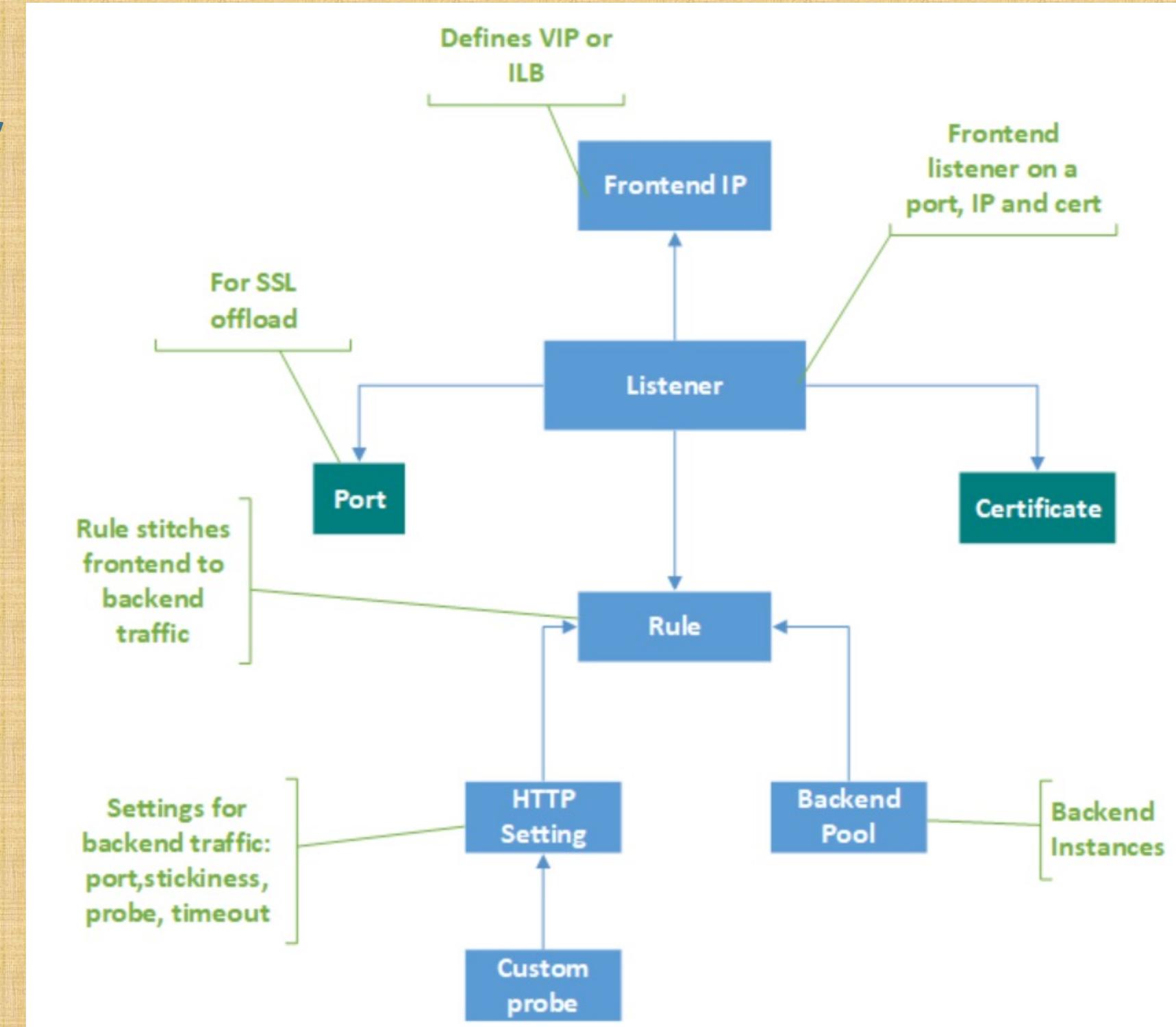
VM
Scale Sets



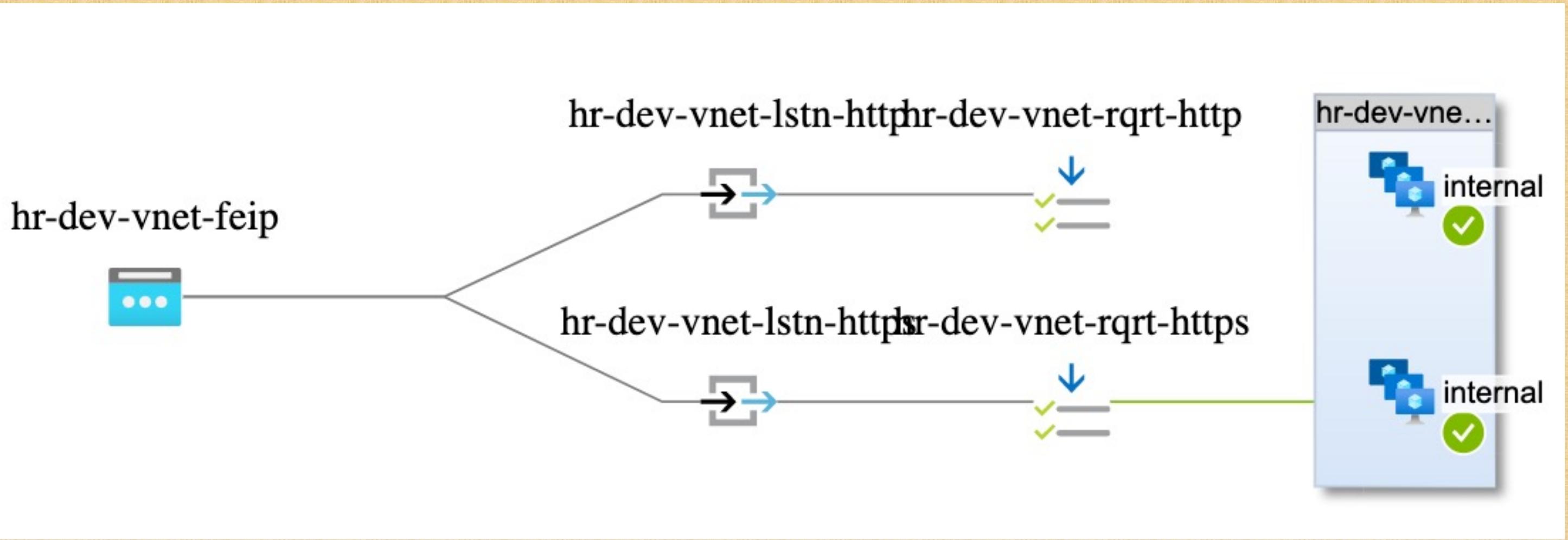
Azure Application Gateway – SSL Self-signed



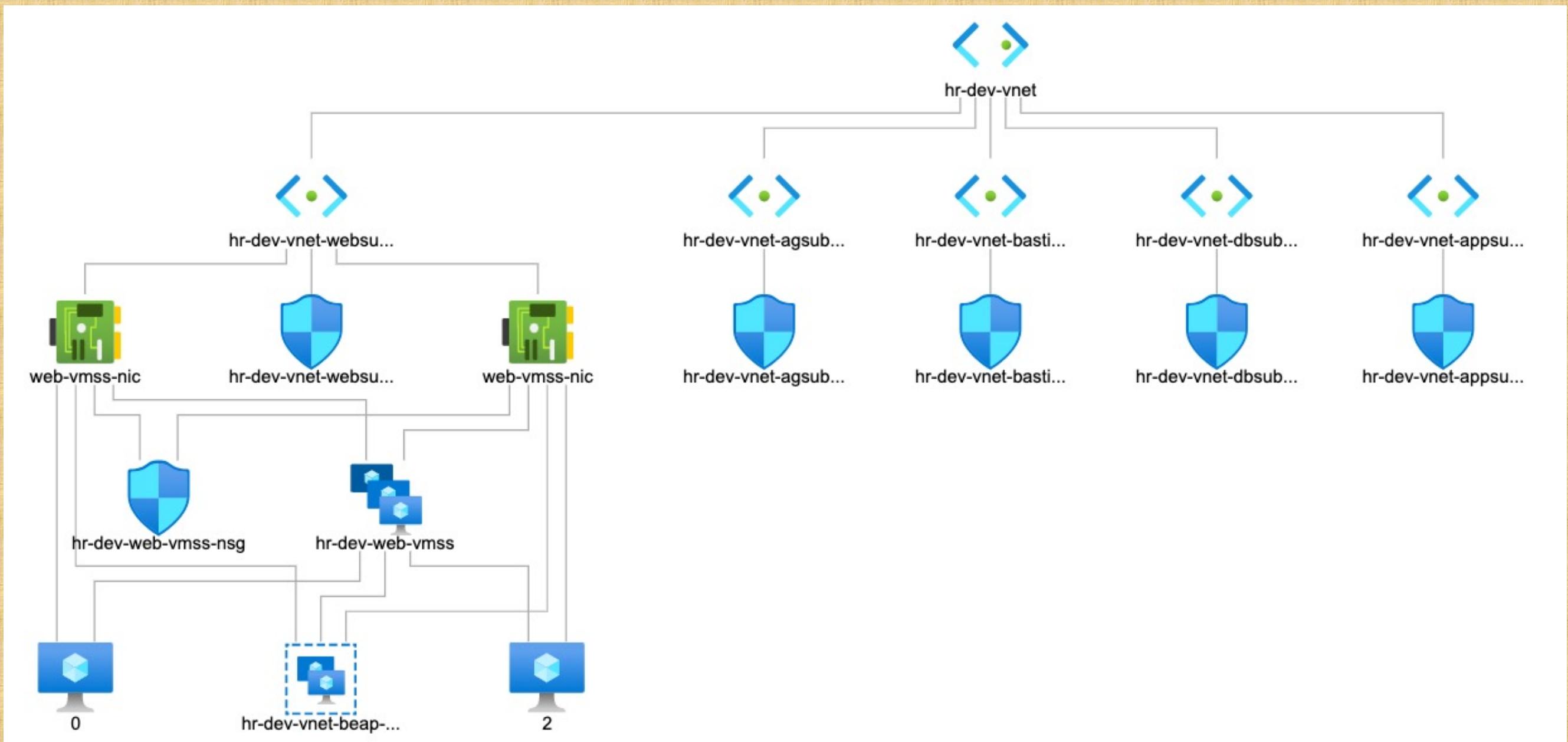
Azure Application Gateway Components



Azure Application Gateway – SSL



VMSS Topology



HTTPS Listener Error Pages

Home > Load balancing - help me choose (Preview) > hr-dev-web-ag >

hr-dev-vnet-lstn-https

hr-dev-web-ag

my-cert-1

Renew or edit selected certificate

Enable SSL Profile ⓘ

Associated rule

hr-dev-vnet-rqrt-https

Additional settings

Listener type ⓘ

Basic Multi site

Error page url

Yes No

Bad gateway - 502

`https://staticwebsitefxpxgn.z13.web.core.windows.net/502.html`

Forbidden - 403

`https://staticwebsitefxpxgn.z13.web.core.windows.net/403.html`

Save

Cancel

HTTP to HTTPS Redirect Routing Rule

hr-dev-vnet-rqrt-http
hr-dev-web-ag

Configure a routing rule to send traffic from a given frontend IP address contain a listener and at least one backend target.

Rule name **hr-dev-vnet-rqrt-http**

*** Listener * Backend targets**

A listener "listens" on a specified port and IP address for traffic that uses the application gateway will apply this routing rule.

Listener **hr-dev-vnet-lstn-http**

hr-dev-vnet-rqrt-http
hr-dev-web-ag

Configure a routing rule to send traffic from a given frontend IP address to one or more backend targets. A routing rule must contain a listener and at least one backend target.

Rule name **hr-dev-vnet-rqrt-http**

*** Listener * Backend targets**

Choose a backend pool to which this routing rule will send traffic. You will also need to specify a set of HTTP settings that define the behavior of the routing rule.

Target type

Backend pool Redirection

Redirection type

Permanent External site

Redirection target

hr-dev-vnet-lstn-https

Target listener *

Yes No

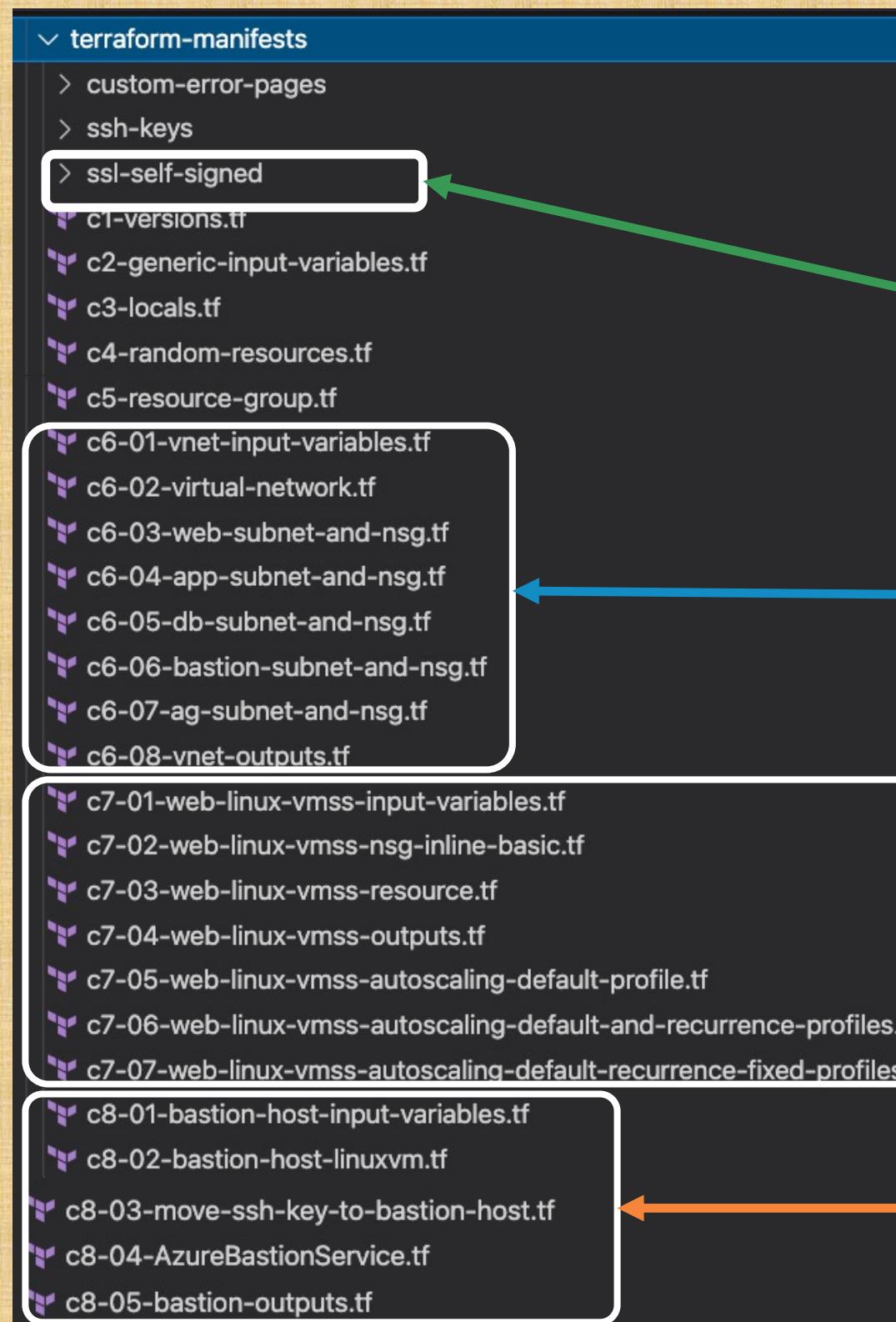
Include query string

Yes No

Include path

Yes No

Terraform Configs



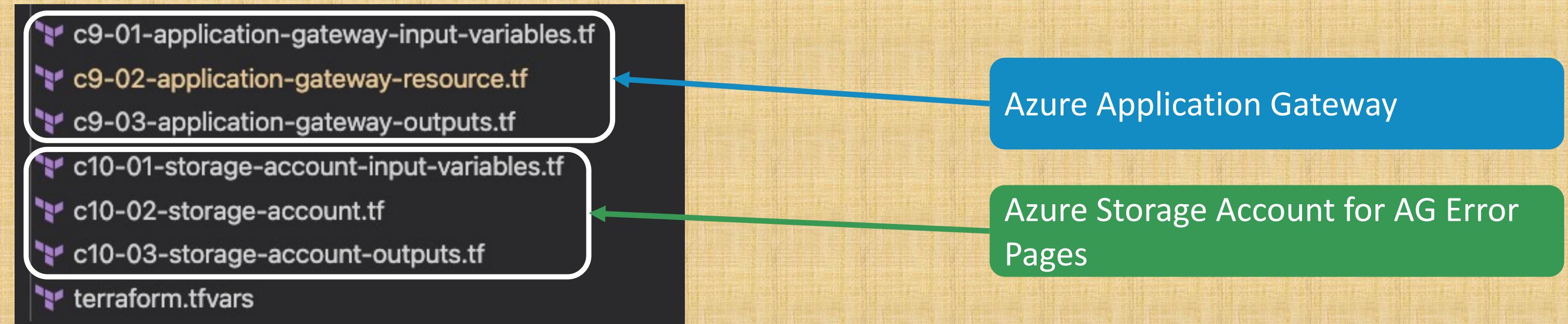
SSL Self-Signed Certificates

Azure Virtual Network with Subnets and Network Security Groups

Azure Virtual Machine Scalesets with Autoscaling enabled

Azure Bastion Linux VM – Disabled
Azure Bastion Service - Disabled

Terraform Configs



^ Azure Application Gateway SSL + HTTP to HTTPS Redirect + AG Error Pages

5 lectures • 54min

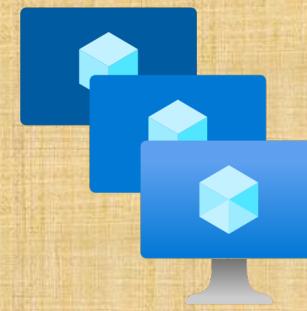
- Step-01: Introduction to AG SSL, HTTP to HTTPS Redirect and AG Error Pages
- Step-02: Generate SSL Self-Signed Certificates and Convert to pfx 05:35
- Step-03: Create Storage Account TF Configs for AG Error Pages 09:48
- Step-04: Create TF Configs for AG HTTP Listener, FE Port, Rule and Redirect Conf 11:12
- Step-05: Create TF Configs for AG HTTPS Listener, Rule and SSL Block 09:47
- Step-06: Execute TF Commands, Verify 3 Usecases and CleanUp 17:19

Time it takes to complete this Demo

Real-World Demo



Application
Gateway



VM
Scale Sets

Azure

Application Gateway

Self Signed SSL

Azure Key Vault



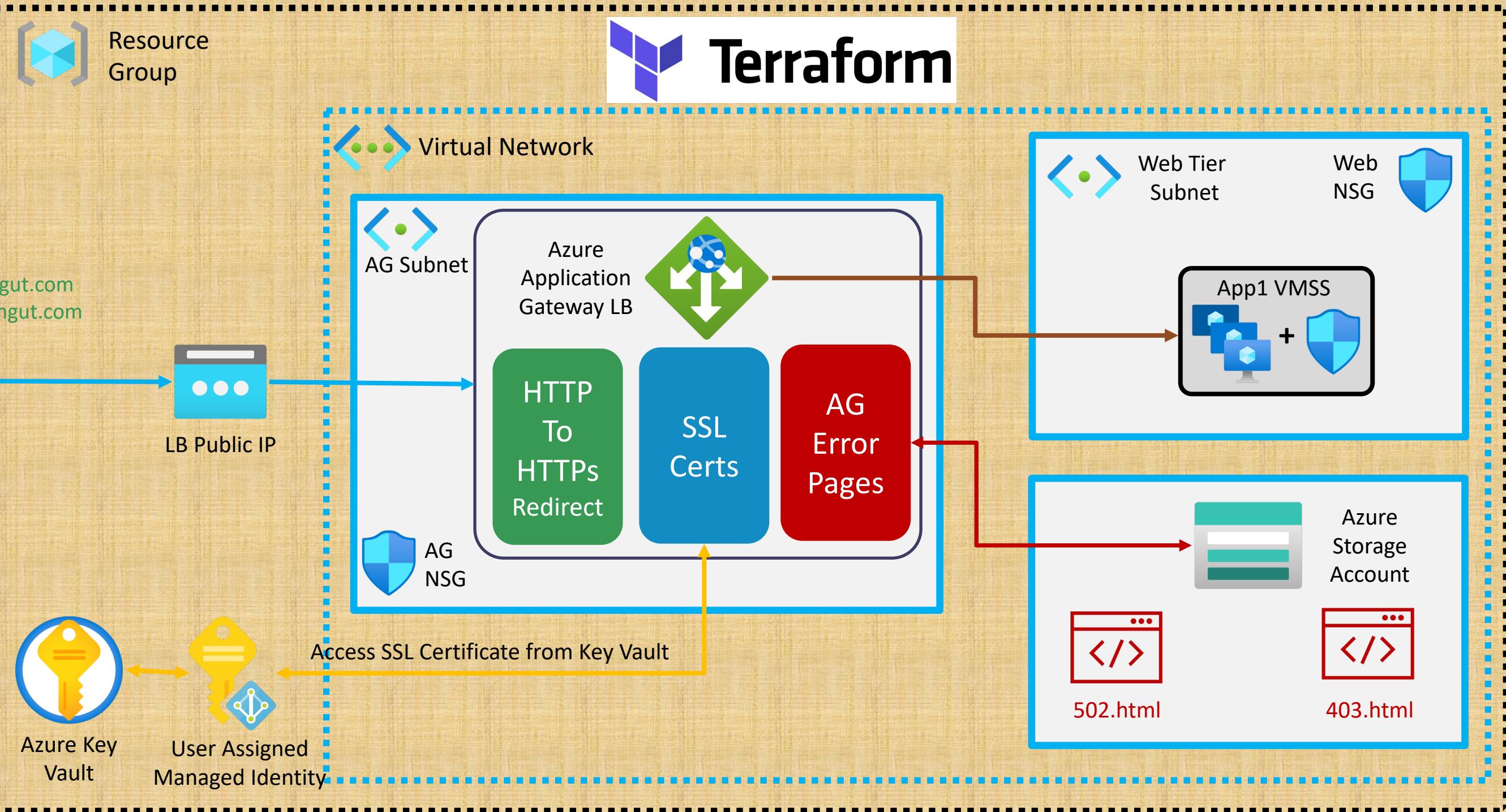
User Managed
Identity

Real-World
Demo



Azure Key Vault

Azure Application Gateway – SSL from Key Vault



What is User-assigned Managed Identity?

How can I use managed identities for Azure resources?

4

I can use Managed Identities when...

As a Developer, I want to build an application using

Source:

Azure Resources
Azure VMs
Azure App Services
Azure Functions
Azure Container instances
Azure Kubernetes Service
Azure Logic Apps
Azure Storage
....

that accesses

Target:

Any target that supports Azure Active Directory Authentication:
- Your applications
- Azure Services:

- Azure Key Vault
- Azure Storage
- Azure SQL...

without having to manage any credentials!

For example, I want to build an application using **Azure App Services** that accesses **Azure Storage** without having to manage any credentials.

Azure Application Gateway HTTPS Listener SSL Certificate associated from Azure Key Vault

Microsoft Azure Search resources, services, and docs (G+)

Home > Load balancing - help me choose (Preview) > hr-dev-web-ag > **hr-dev-vnet-lstn-https** ...

hr-dev-web-ag

Listener name hr-dev-vnet-lstn-https

Frontend IP * Public

Port * 443

Protocol HTTP HTTPS

Choose a certificate Create new Select existing

Certificate * keyvault-my-cert-1

Renew or edit selected certificate

Enable SSL Profile

Azure Key Vault – SSL Certificate Import

Home > Key vaults > keyvault1tawdrc

keyvault1tawdrc | Certificates

Key vault

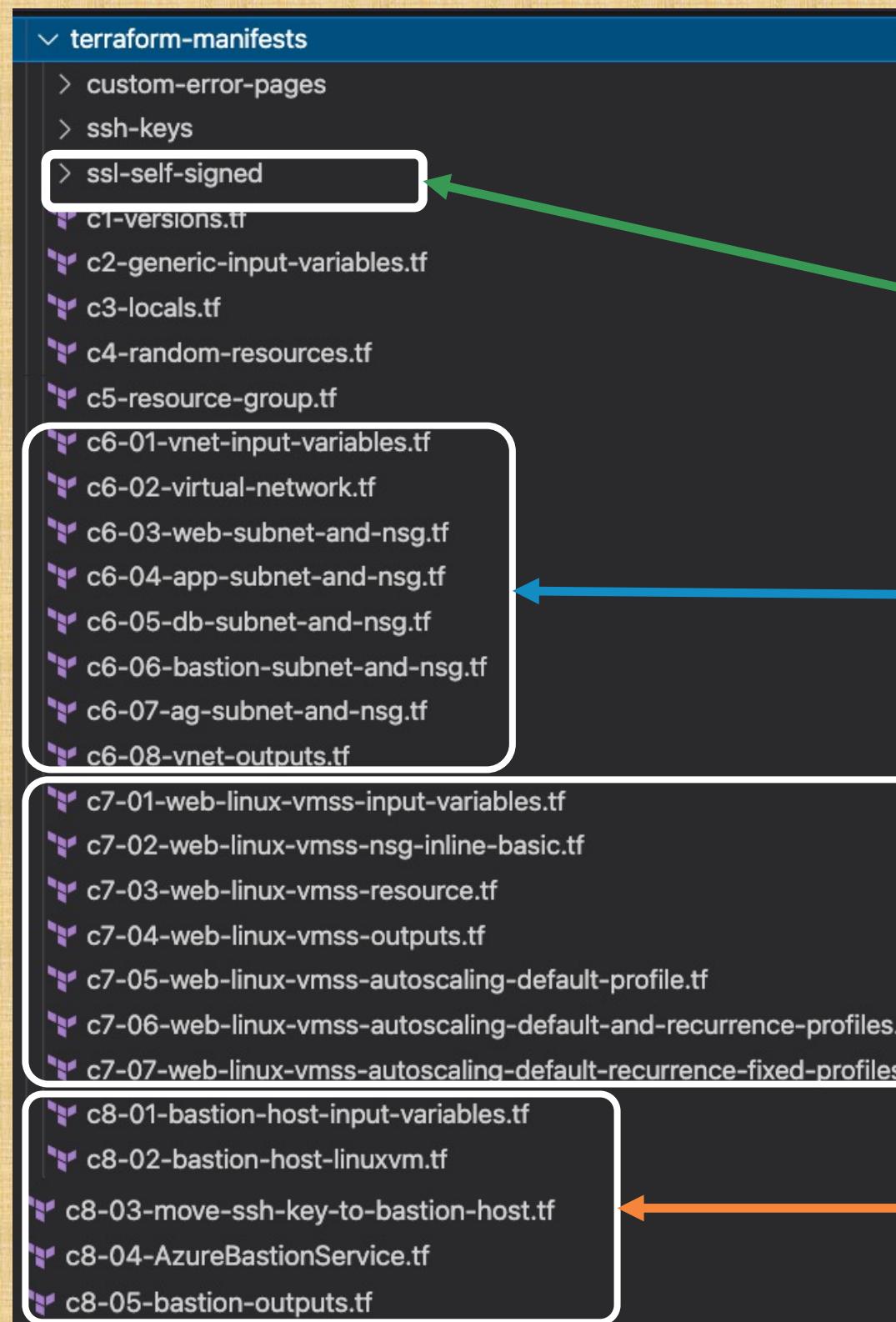
Search (Cmd+/) <> + Generate/Import ⏪ Refresh ⏹ Restore Backup ⚡ Manage deleted certificates 📩 Certificate Contacts 🗃 Certificate Authorities

Name	Thumbprint	Status	Expiration date
Completed			
my-cert-1	31BCB0900198CC10F88700EC0FE8CE...	✓ Enabled	8/19/2041
In progress, failed or cancelled			
There are no certificates available.			

Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Events

Settings
Keys
Secrets
Certificates
Access policies

Terraform Configs



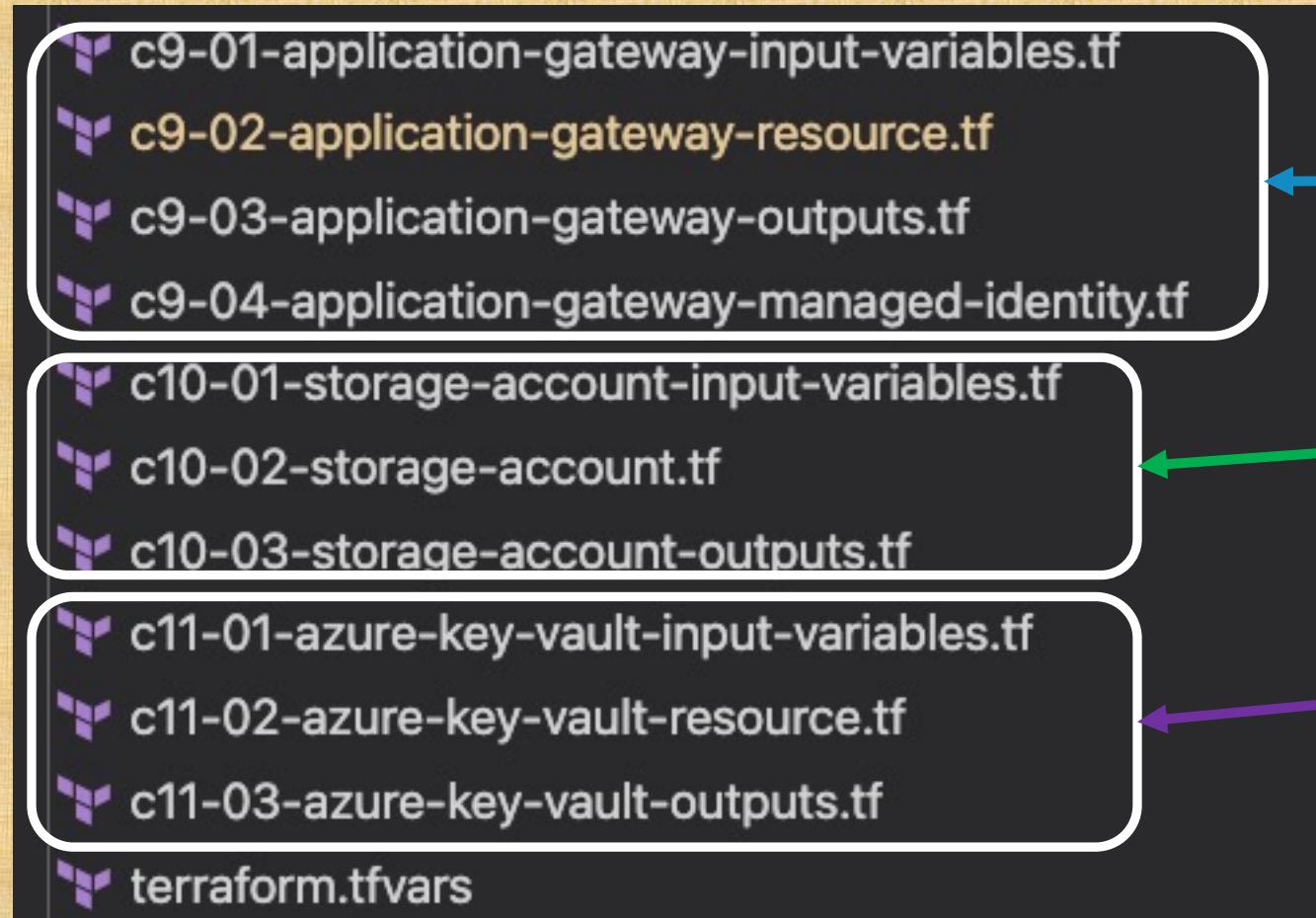
SSL Self-Signed Certificates

Azure Virtual Network with Subnets and Network Security Groups

Azure Virtual Machine Scalesets with Autoscaling enabled

Azure Bastion Linux VM – Disabled
Azure Bastion Service - Disabled

Terraform Configs



Azure Application Gateway + User Managed Identity

Azure Storage Account for AG Error Pages

Azure Key Vault for SSL Certs

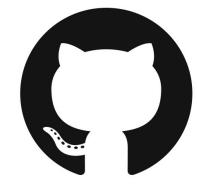
^ Azure Application Gateway SSL from Azure Key Vault using Terraform

- ❑ Step-01: Introduction to AG SSL Integration with Azure Key Vault
- ▶ Step-02: Review TF Configs for UMID, KeyVault, KV Access Policy Resources and Li 18:20
- ▶ Step-03: Review TF Configs for KV Access Policy for UMID, KV Cert Import Resourc 15:23
- ▶ Step-04: Review Application Gateway TF Configs required for Key Vault Usecase 04:57
- ▶ Step-05: Execute TF Commands, Verify keyVault Resources and CleanUp 09:36

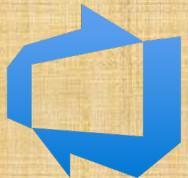
4 lectures • 48min

Time it takes to complete this Demo

Real-World Demo



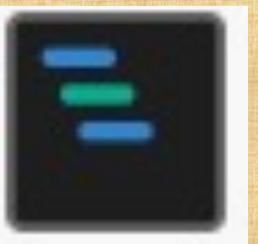
Github



Azure
DevOps



Azure
Pipelines



Starter
Pipelines

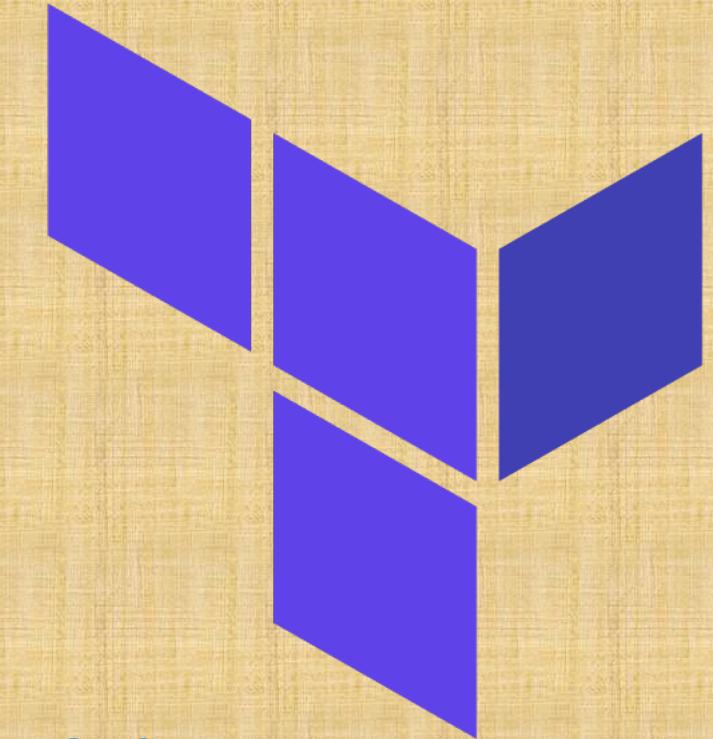


Real-World
Demo

Azure IaC DevOps

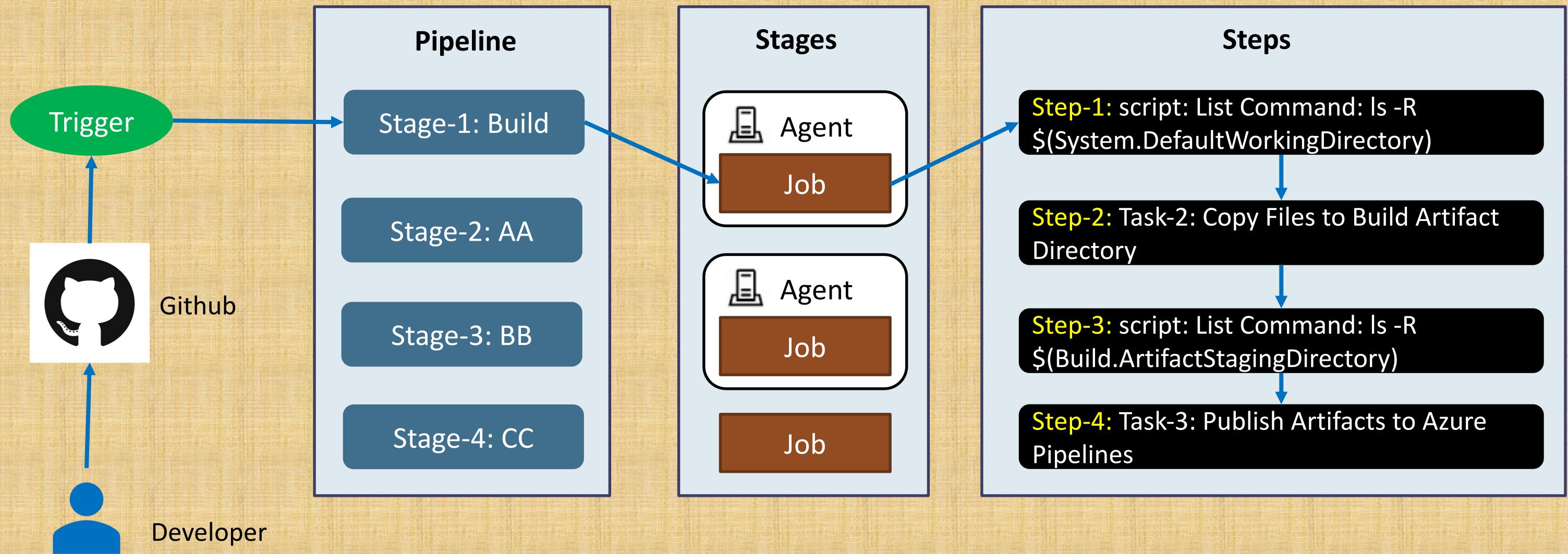
Continuous Integration & Delivery Pipelines

Azure Build & Release Pipelines





Azure Pipelines – Key Concepts



Azure DevOps – Build Pipeline

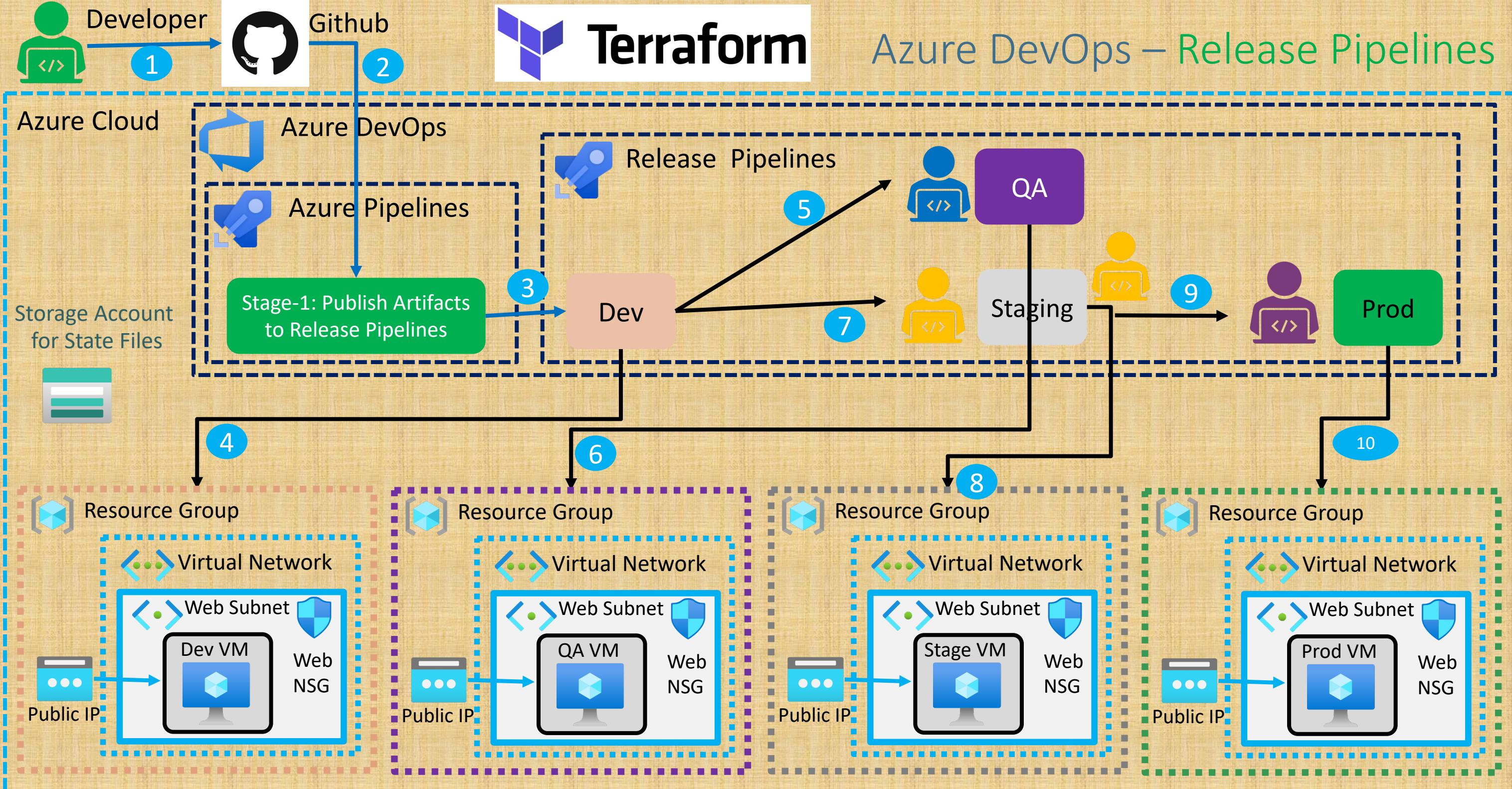
Continuous Integration Pipeline

Task-1

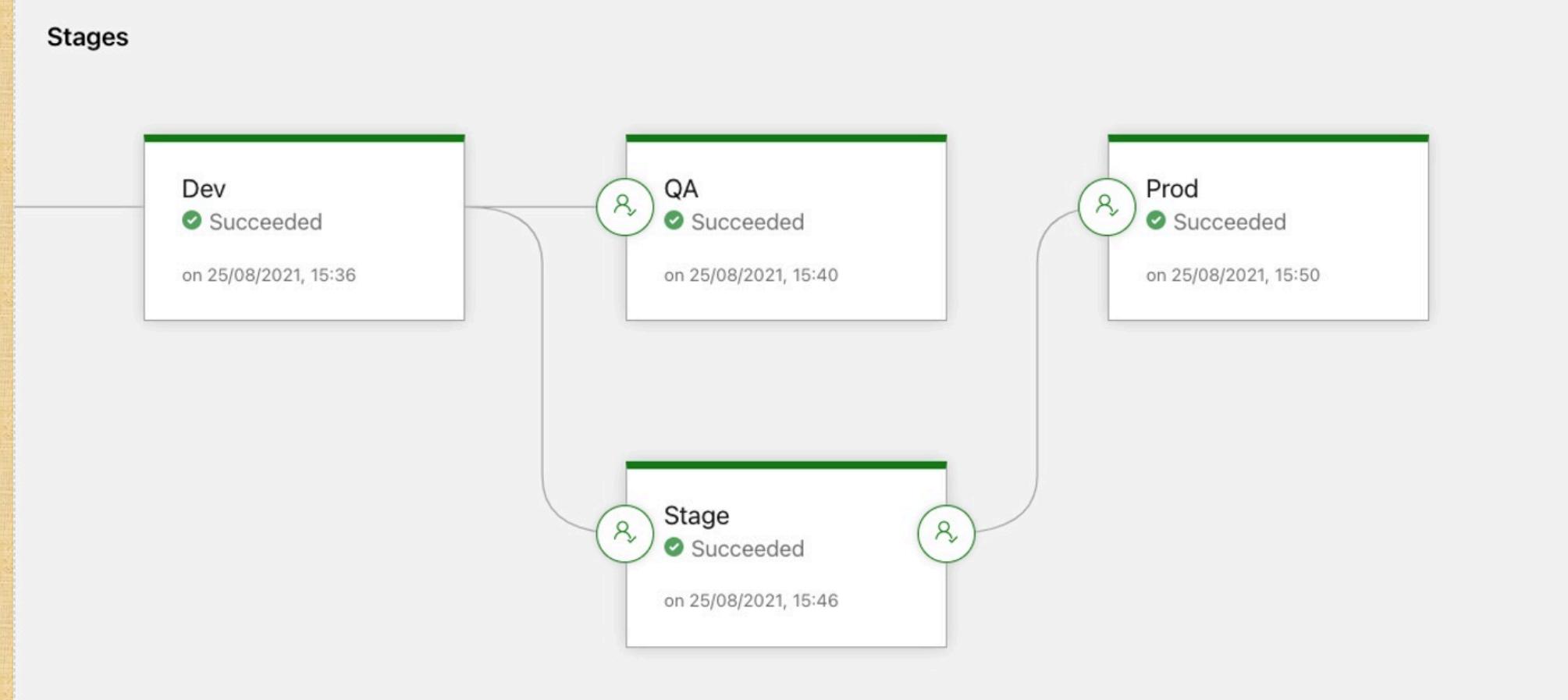
Copy files (terraform-manifests folder) from System default working Directory to Build Artifact Directory

Task-2

Publish Build Artifacts to Azure Pipelines, so that we can use them in Release Pipelines



Azure IaC DevOps – Release Pipeline



Azure IaC DevOps Release Pipelines

All pipelines > `↑ Terraform-CD`

Pipeline Tasks Variables Retention Options History

Dev Deployment process

Agent job Run on agent +

- Install Terraform 1.0.5 Terraform tool installer
- Terraform : Init Terraform
- Terraform : validate Terraform
- Terraform : plan Terraform
- Terraform : apply -auto-approve Terraform

Stage name
Dev

Pipelines

- Pipelines
- Environments
- Releases
- Library
- Task groups
- Deployment groups
- Test Plans
- Artifacts

Azure IaC DevOps Releases

The screenshot shows the Azure DevOps interface. On the left, the sidebar is open with the 'Releases' option selected. In the center, the 'Terraform-CD' pipeline is displayed under the 'Pipelines' section. At the top right of the pipeline view, there are 'Edit' and 'Create release' buttons. Below the pipeline name, there's a status indicator showing 'Prod'. The main area shows a table of releases:

Release	Created	Stages
Release-7	25/08/2021, 15:59:34	Dev, QA, Stage, Prod
Release-6	25/08/2021, 15:32:50	Dev, QA, Stage, Prod
Release-5	25/08/2021, 15:14:29	Dev
Release-4	25/08/2021, 15:01:11	Dev
Release-3	25/08/2021, 14:55:53	Dev
Release-2	25/08/2021, 14:51:55	Dev
Release-1	25/08/2021, 14:43:41	Dev

Azure IaC DevOps Releases

The screenshot shows the Azure DevOps interface. On the left, the sidebar includes links for Overview, Boards, Repos, Pipelines, Environments, Releases (selected), Library, Task groups, Deployment groups, Test Plans, and Artifacts. The main area displays the 'Terraform-CD' pipeline under 'Pipelines'. The pipeline details show it's a 'Terraform-CD' pipeline for 'Prod' environment. The 'Releases' tab is selected, showing a list of seven releases:

Release	Created	Stages
Release-7	25/08/2021, 15:59:34	Dev, QA, Stage, Prod
Release-6	25/08/2021, 15:32:50	Dev, QA, Stage, Prod
Release-5	25/08/2021, 15:14:29	Dev
Release-4	25/08/2021, 15:01:11	Dev
Release-3	25/08/2021, 14:55:53	Dev
Release-2	25/08/2021, 14:51:55	Dev
Release-1	25/08/2021, 14:43:41	Dev

Azure IaC DevOps Environment TF State Files

Microsoft Azure Search resources, services, and docs (G+)

Home > Resource groups > terraform-storage-rg > terraformstate201 >

 **tfstatefiles** ...

» Upload Change access level Refresh | Delete | Change tier | Acquire lease Brea

Authentication method: Access key ([Switch to Azure AD User Account](#))
Location: tfstatefiles

Search blobs by prefix (case-sensitive)

Add filter

Name	Modified	Access tier
<input type="checkbox"/> dev-terraform.tfstate	8/25/2021, 4:01:20 PM	Hot (Inferred)
<input type="checkbox"/> prod-terraform.tfstate	8/25/2021, 3:50:26 PM	Hot (Inferred)
<input type="checkbox"/> qa-terraform.tfstate	8/25/2021, 4:03:15 PM	Hot (Inferred)
<input type="checkbox"/> stage-terraform.tfstate	8/25/2021, 4:05:37 PM	Hot (Inferred)

Virtual Network Subnets - 4 Environments

	Name ↑↓	IPv4 ↑↓	
Dev	hr-dev-vnet-websubnet	10.1.1.0/24	QA
	hr-dev-vnet-bastionsu...	10.1.100.0/24	
	hr-dev-vnet-dbsubnet	10.1.21.0/24	
	hr-dev-vnet-appsubnet	10.1.11.0/24	
Stage	hr-stage-vnet-bastion...	10.3.100.0/24	Prod
	hr-stage-vnet-dbsubnet	10.3.21.0/24	
	hr-stage-vnet-websub...	10.3.1.0/24	
	hr-stage-vnet-appsub...	10.3.11.0/24	
	hr-qatest-vnet-bastionsub...	10.2.100.0/24	
	hr-qatest-vnet-dbsubnet	10.2.21.0/24	
	hr-qatest-vnet-websubnet	10.2.1.0/24	
	hr-qatest-vnet-appsubnet	10.2.11.0/24	
	hr-prod-vnet-bastionsub...	10.4.100.0/24	
	hr-prod-vnet-dbsubnet	10.4.21.0/24	
	hr-prod-vnet-websubnet	10.4.1.0/24	
	hr-prod-vnet-appsubnet	10.4.11.0/24	

4 Environments - Resources

Resource Groups

Name ↑↓
 hr-dev-rg-fzmqwz
 hr-prod-rg-qhddgg
 hr-qa-rg-vmsjwa
 hr-stage-rg-btcjcz

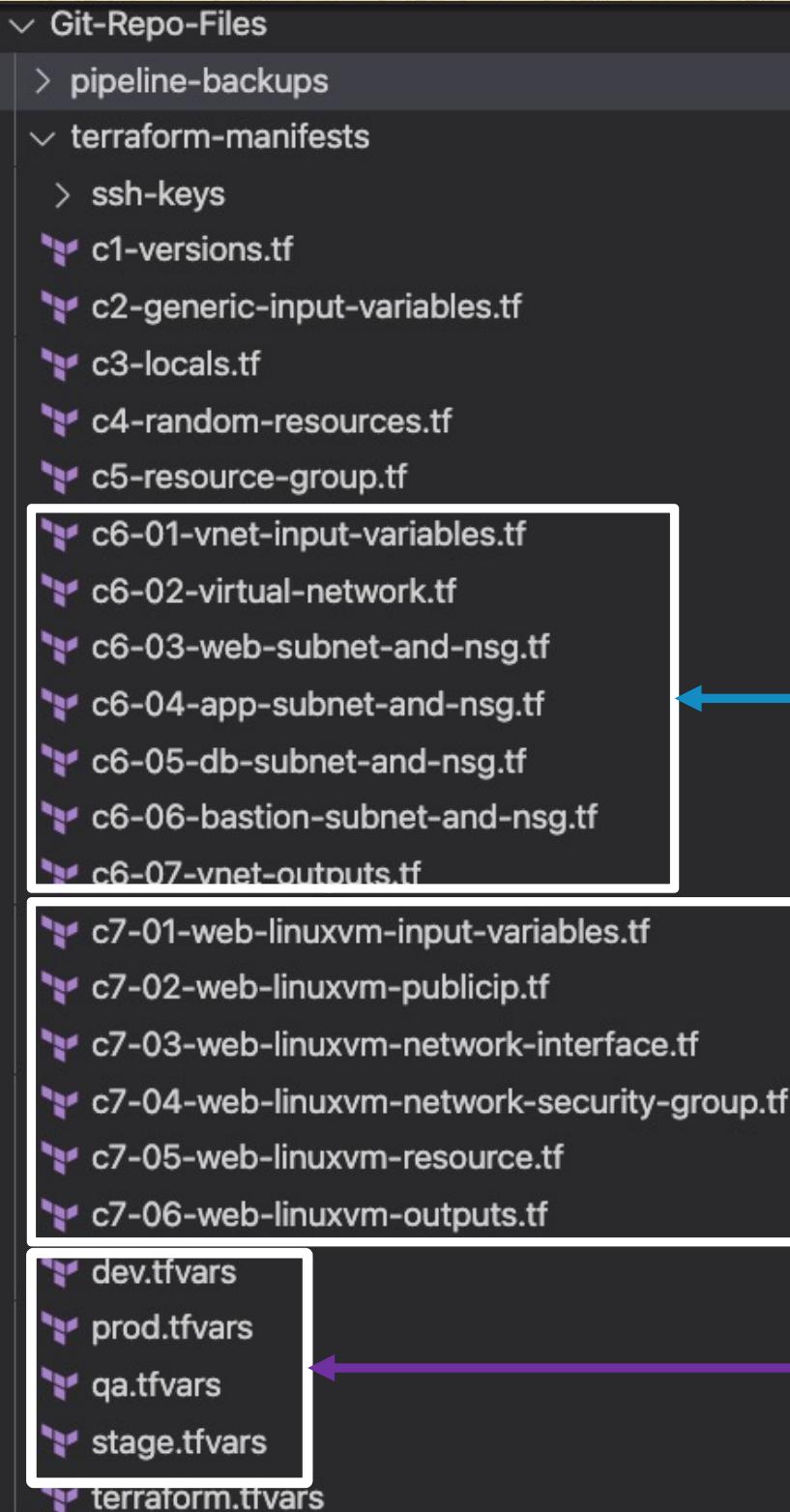
Public IP

Name ↑↓
 hr-dev-web-linuxvm-publicip
 hr-prod-web-linuxvm-publicip
 hr-qa-web-linuxvm-publicip
 hr-stage-web-linuxvm-publicip

Virtual Machines

Name ↑↓
 hr-dev-web-linuxvm
 hr-prod-web-linuxvm
 hr-qa-web-linuxvm
 hr-stage-web-linuxvm

Terraform Configs



Azure Virtual Network with Subnets and Network Security Group Resources

Azure Web Linux VM with Public IP and Network Interface Resources

Environment specific tfvar files
terraform apply -auto-approve -var-file=dev.tfvars
terraform apply -auto-approve -var-file=qa.tfvars

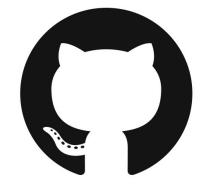
^ Azure IaC DevOps for Terraform Projects

14 lectures • 2hr 37min

- Step-01: Introduction to Azure IaC DevOps
- ▶ Step-02: Review Terraform Configs 16:04
- ▶ Step-03: Create Git Repo and Check-In Terraform Configs 06:16
- ▶ Step-04: GIT SSH Connection Note 01:49
- ▶ Step-05: Terraform Dependency Lock File check-in to Git Repo 19:29
- ▶ Step-06: Create Azure DevOps Organization and Install TF Extension 10:02
- ▶ Step-07: Azure DevOps Build Pipelines Introduction 12:10
- ▶ Step-08: Create Azure Build Pipeline 17:25
- ▶ Step-09: Azure DevOps Release Pipelines Introduction 09:34
- ▶ Step-10: Create Service Connection and Storage Account for TFState Files 08:33
- ▶ Step-11: Create Releaes Pipeline, Artificats, Dev Stage Install and Init Tasks 12:13
- ▶ Step-12: Create Validate, Plan, Apply Tasks for Dev Stage 05:49
- ▶ Step-13: Run end to end pipelines and verify dev resources 08:05
- ▶ Step-14: Create QA, Stage and Prod stages and Verify Resources 17:54
- ▶ Step-15: Make changes to prod.tfvars, verify and clean-up 11:23

Time it takes to complete this Demo

Real-World
Demo



Github



Azure
DevOps



Azure
Pipelines



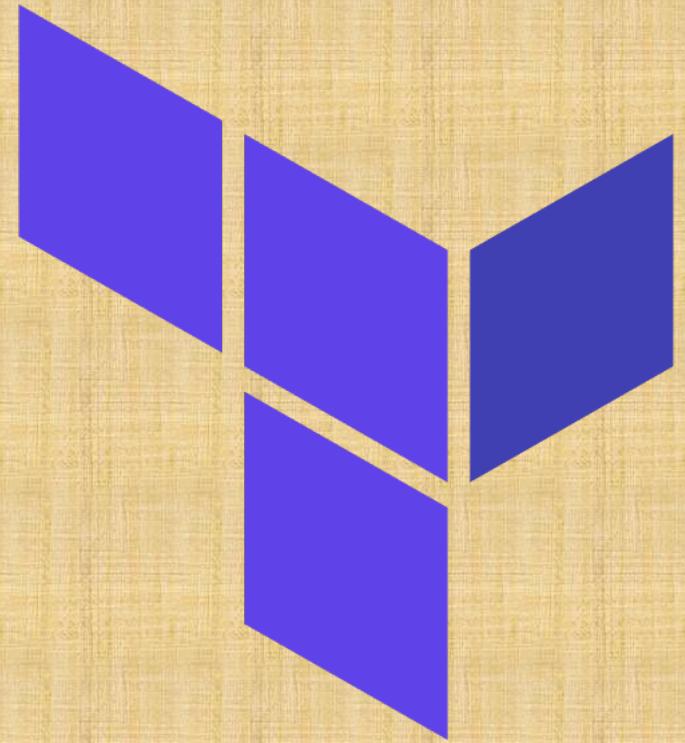
Starter
Pipelines

Real-World
Demo

Azure IaC DevOps

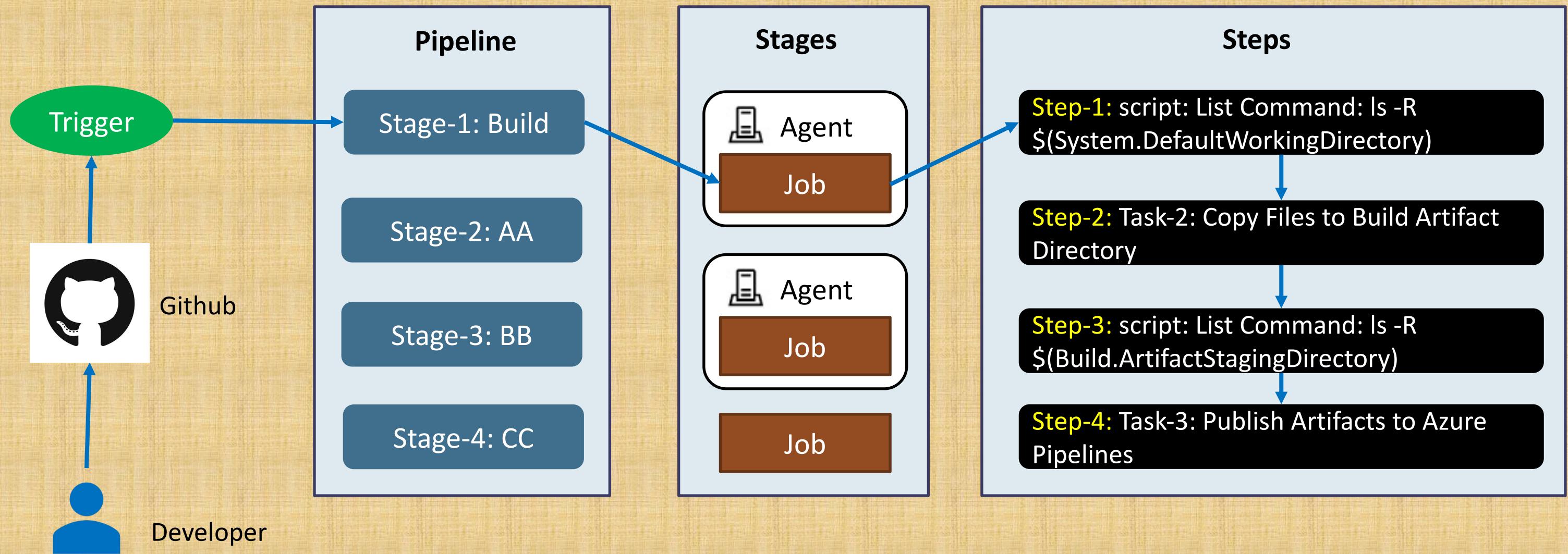
Continuous Integration Pipelines

Build Pipelines

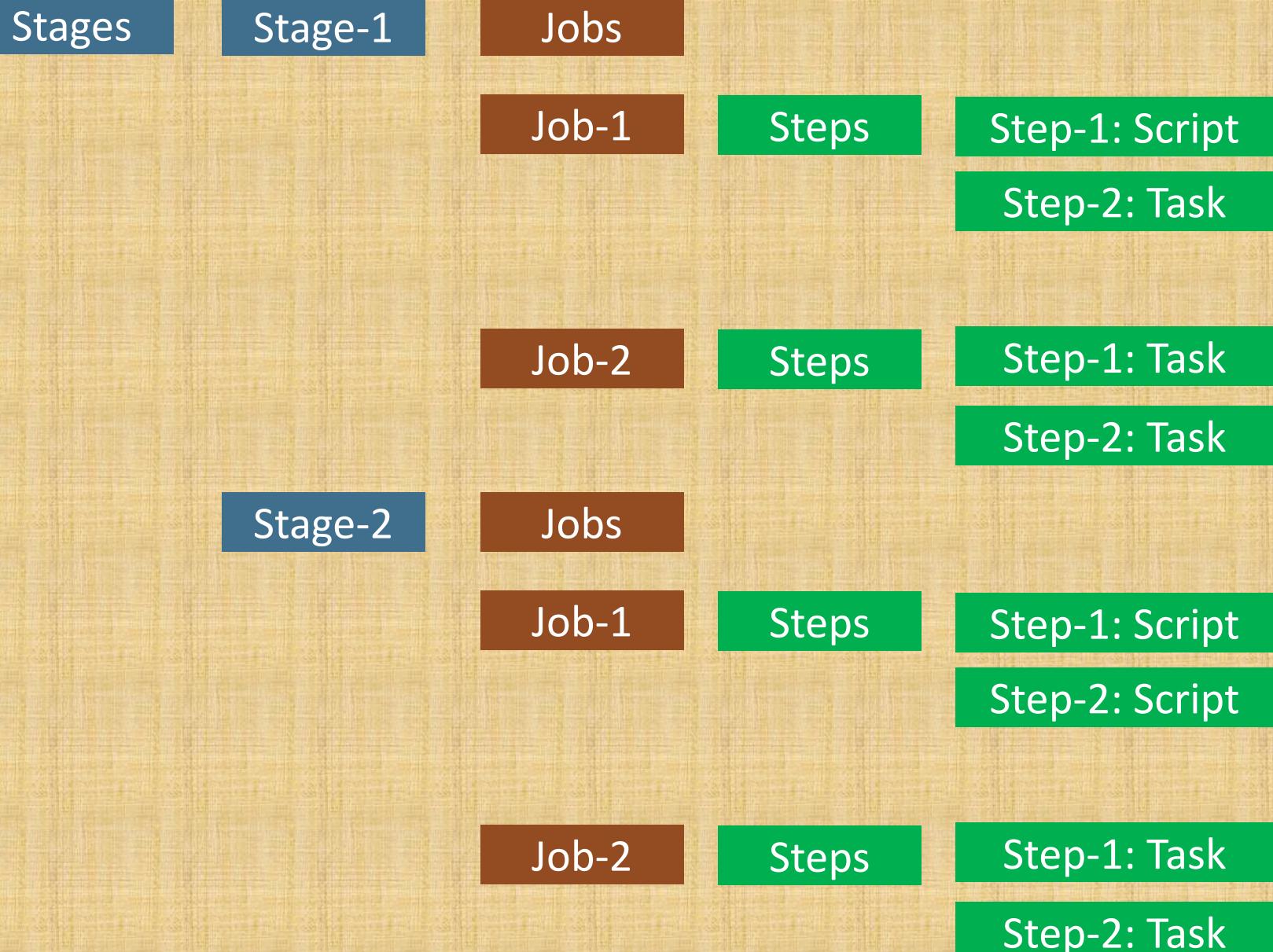




Azure Pipelines – Key Concepts



Azure Pipelines – Key Concepts



```
stages:  
- stage: Stage-1  
  jobs:  
    - job: Job-1  
      steps:  
        - script: echo Step-1  
        - script: echo Step-2  
    - job: Job-2  
      steps:  
        - task: some task step-1  
        - task: some task step-2  
- stage: Stage-2  
  jobs:  
    - job: Job-1  
      steps:  
        - task: some task step-1  
        - task: some task step-2  
    - job: Job-2  
      steps:  
        - script: echo Step-1  
        - script: echo Step-2
```

Azure DevOps – Build Pipeline

Continuous Integration Pipeline

Task-1

Copy files (terraform-manifests folder) from System default working Directory to Build Artifact Directory

Task-2

Publish Build Artifacts to Azure Pipelines, so that we can use them in Release Pipelines

Azure IaC DevOps – CI/Build Pipeline

```
1 trigger:
2   - main
3
4   # Stages
5   # Stage-1:
6     # Task-1: Copy terraform-manifests files to Build Artifact Directory
7     # Task-2: Publish build artifacts to Azure Pipelines
8   # Pipeline Hierarchical Flow: Stages -> Stage -> Jobs -> Job -> Steps -> Task1, Task2, Task3 ...
9
10  stages:
11    # Build Stage
12    - stage: Build
13      displayName: Build Stage
14      jobs:
15        - job: Build
16          displayName: Build Job
17          pool:
18            vmImage: 'ubuntu-latest'
19          steps:
20            ## Publish Artifacts pipeline code in addition to Build and Push .....
21            - bash: echo Contents in System Default Working Directory; ls -R $(System.DefaultWorkingDirectory)
22            - bash: echo Before copying Contents in Build Artifact Directory; ls -R $(Build.ArtifactStagingDirectory)
23            # Task 2: Copy files (Copy files from a source folder to target folder)
```

Azure IaC DevOps – CI/Build Pipeline

← Terraform Continuous Integration CI Pipeline Disabled Variables Run :

main stack simplify/terraform-on-azure-with-azure-devops-internal / azure-pipelines.yml Show assistant

```
23     # Task-2: Copy files (Copy files from a source folder to target folder)
24     # Source Directory: $(System.DefaultWorkingDirectory)/terraform-manifests
25     # Target Directory: $(Build.ArtifactStagingDirectory)

    Settings
26     - task: CopyFiles@2
27         inputs:
28             SourceFolder: '$(System.DefaultWorkingDirectory)/terraform-manifests'
29             Contents: '**'
30             TargetFolder: '$(Build.ArtifactStagingDirectory)'
31             OverWrite: true
32             # List files from Build Artifact Staging Directory – After Copy
33             - bash: echo After copying to Build Artifact Directory; ls -R $(Build.ArtifactStagingDirectory)
34             # Task-3: Publish build artifacts (Publish build to Azure Pipelines)

    Settings
35     - task: PublishBuildArtifacts@1
36         inputs:
37             PathToPublish: '$(Build.ArtifactStagingDirectory)'
38             ArtifactName: 'terraform-manifests'
39             publishLocation: 'Container' ...
40
```

Azure IaC DevOps – CI/Build Pipeline

terraform-on-azure-w... +

- Overview
- Boards
- Repos
- Pipelines
- Pipelines**
- Environments
- Releases
- Library
- Task groups
- Deployment groups
- Test Plans
- Artifacts

← **Jobs in run #20210825.8**
Terraform Continuous Integration CI Pipeline

Build Stage

Job	Duration
Build Job	15s
Initialize job	1s
Checkout stacksimplicity/terraform-on-azure-with-azure-devops-internal@main	5s
Bash	3s
Bash	<1s
CopyFiles	2s
Bash	<1s
PublishBuildArtifacts	1s
Post-job: Checkout st...	<1s
Finalize Job	<1s

```
1 Starting: Checkout stacksimplicity/terraform-on-azure-with-azure-devops-internal@main to s
2 =====
3 Task      : Get sources
4 Description: Get sources from a repository. Supports Git, TfsVC, and SVN repositories.
5 Version   : 1.0.0
6 Author    : Microsoft
7 Help     : [More Information](https://go.microsoft.com/fwlink/?LinkId=798199)
8 =====
9 Cleaning any cached credential from repository: stacksimplicity/terraform-on-azure-with-azure-devops-internal (GitHub)
10 Finishing: Checkout stacksimplicity/terraform-on-azure-with-azure-devops-internal@main to s
```

Azure DevOps Parallelism Request

Agent: 1800
Minutes Free

dev.azure.com/terraform-on-azure-prep/_settings/billing

Azure DevOps terraform-on-azure-prep / Settings / Billing

Organization Settings

terraform-on-azure-prep

Search Settings

General

- Overview
- Projects
- Users
- Billing**
- Auditing
- Global notifications
- Usage
- Extensions
- Azure Active Directory

Security

- Policies
- Permissions

Boards

- Process

Pipelines

- Agent pools
- Settings

Billing

Billing has not been set up for this organization. Access will be available up to [free tier limits](#).

Set up billing

Pipelines for private projects	Free	Paid parallel jobs
MS Hosted CI/CD [?]	1800 minutes	0
Self-Hosted CI/CD [?]	1	0

Visit [parallel jobs](#) for full details on free pipelines and public concurrency

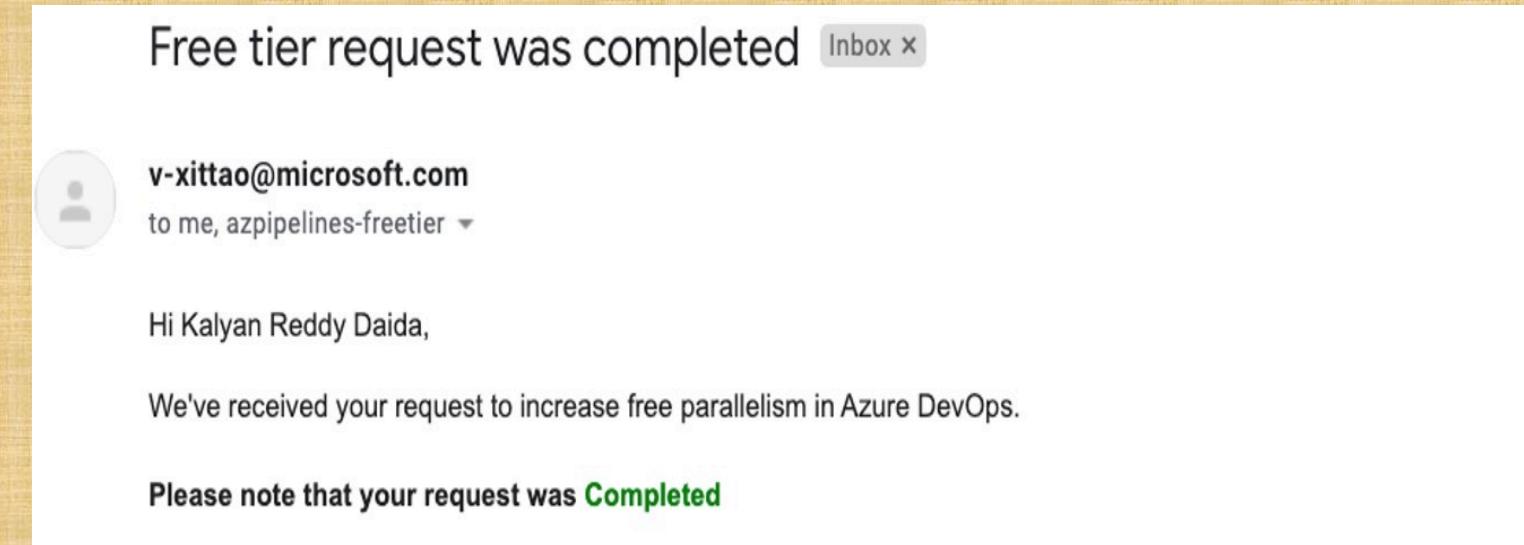
Boards, Repos and Test Plans	Free
Basic users [?]	5
Basic + Test Plans [?]	Start free trial

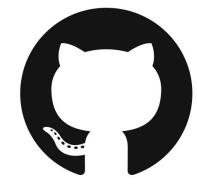
Settings	Access level
Default access level for new users [?]	Stakeholder

Azure DevOps Parallelism Request

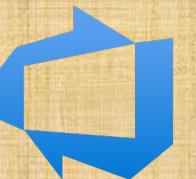
Usually, it takes 2 to 3 business days for approval.

It got approved for me in 24 hours.





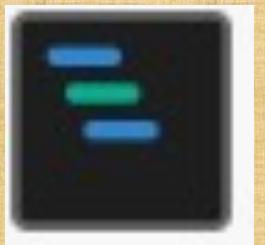
Github



Azure
DevOps



Azure
Pipelines



Starter
Pipelines



Azure
Release
Pipelines

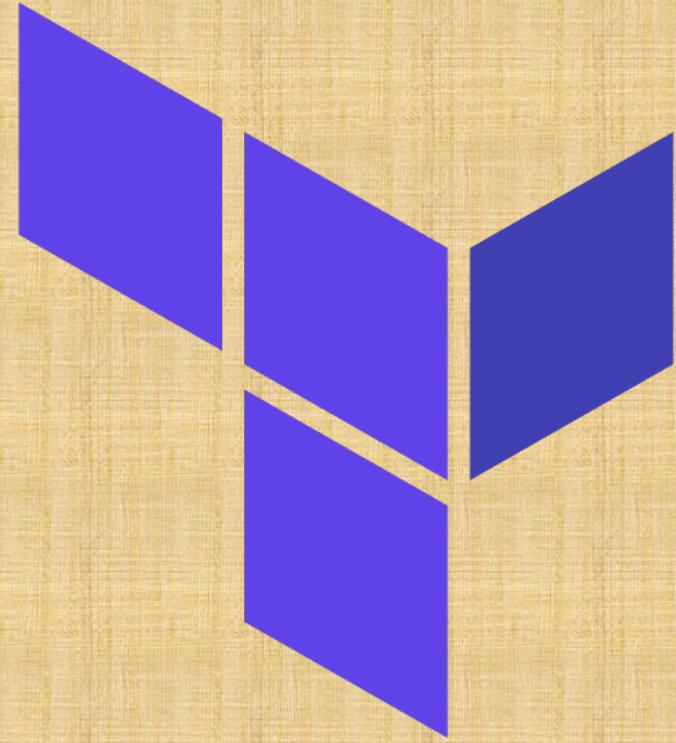


Real-World
Demo

Azure IaC DevOps

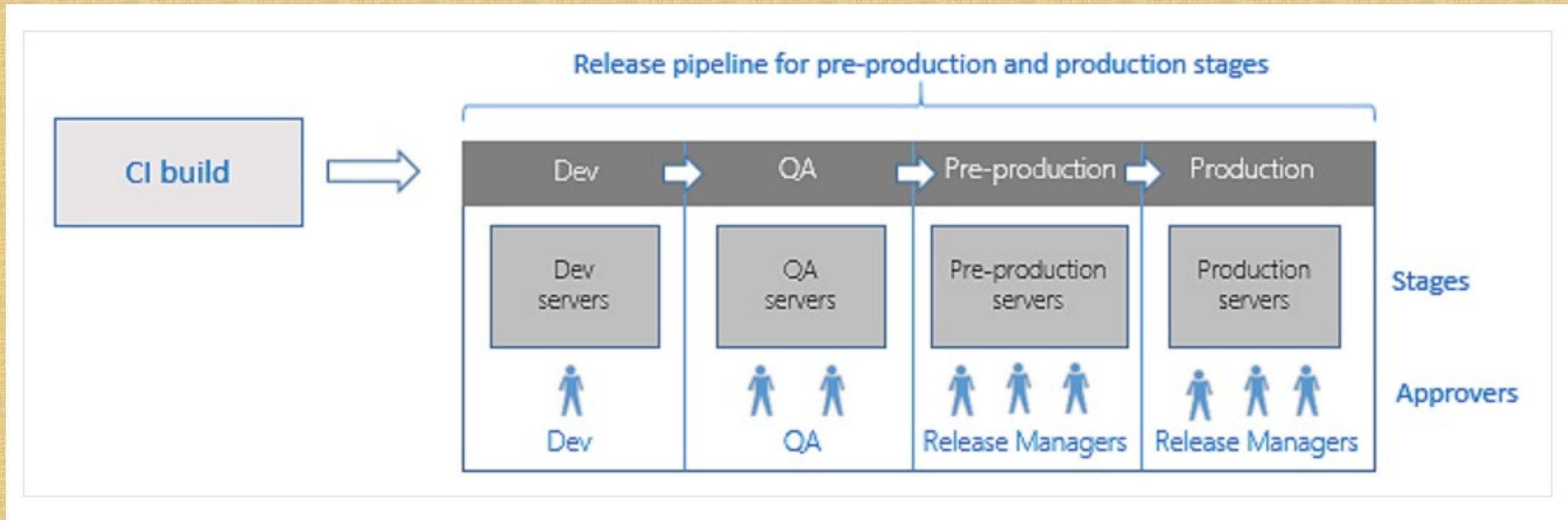
Continuous Delivery Pipelines

Release Pipelines

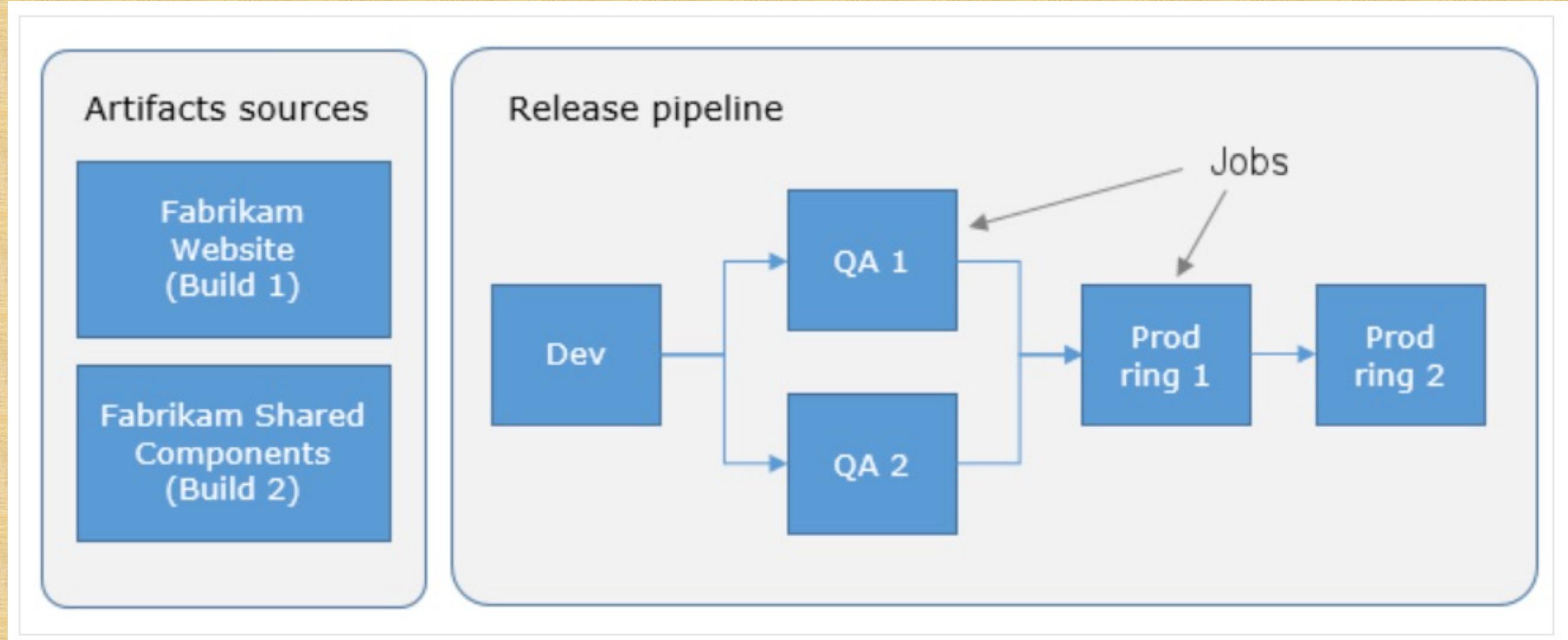


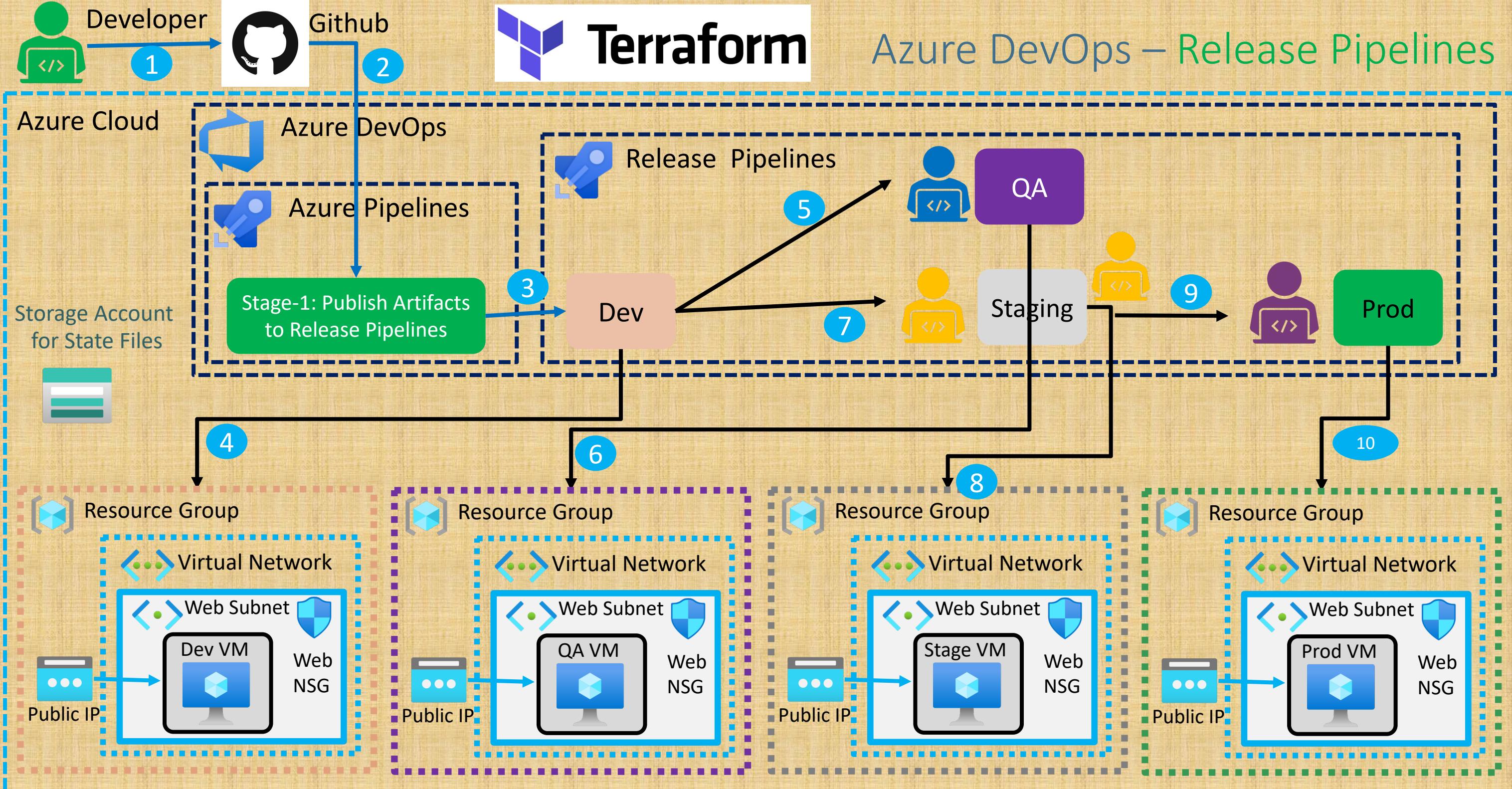
Azure DevOps – Release Pipelines

To achieve **Continuous Delivery** we use Release Pipelines

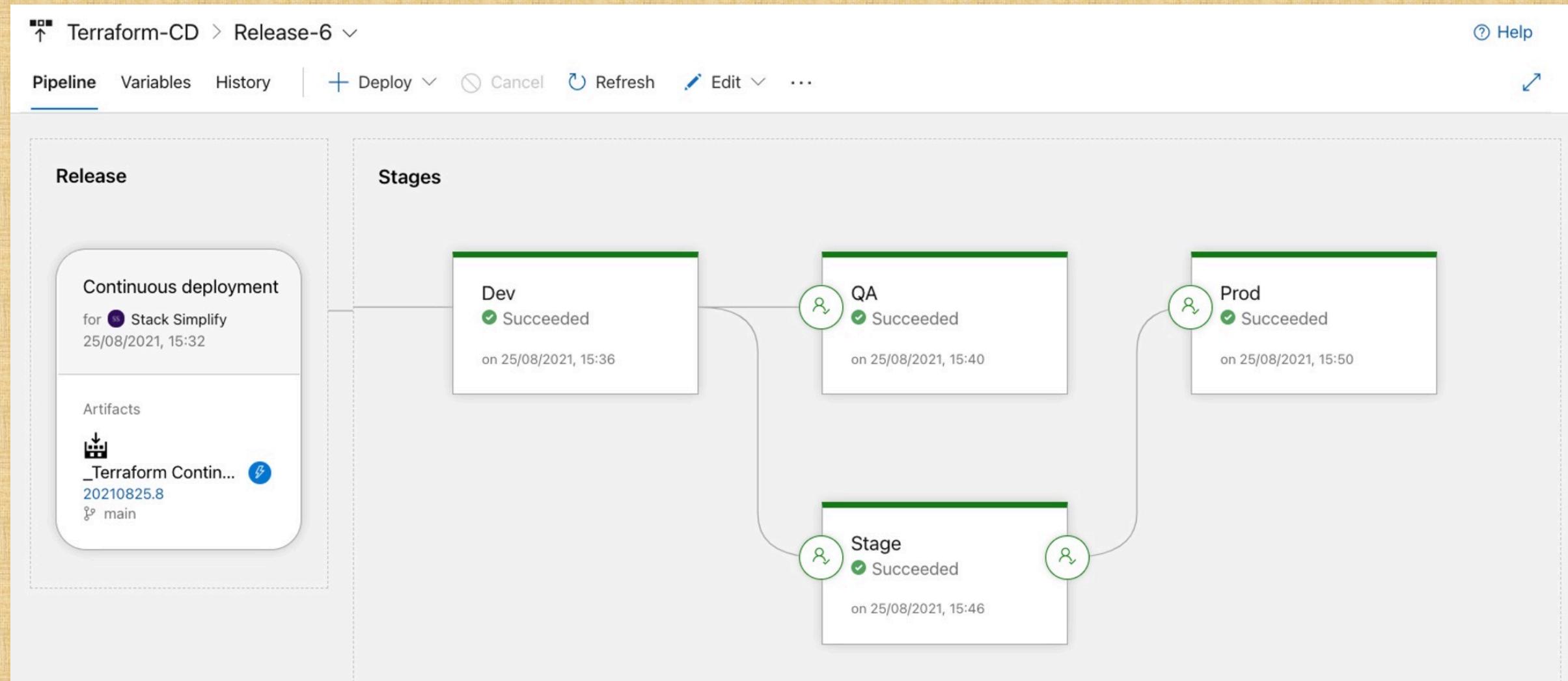


Azure DevOps – Release Pipelines





Azure IaC DevOps – Release Pipeline



Azure IaC DevOps Release Pipelines

All pipelines > `↑ Terraform-CD`

Pipeline Tasks **Variables** Retention Options History

Dev Deployment process ...

Agent job +
Run on agent

Install Terraform 1.0.5
Terraform tool installer

Terraform : Init
Terraform

Terraform : validate
Terraform

Terraform : plan
Terraform

Terraform : apply -auto-approve
Terraform

Stage name
Dev

The screenshot shows the Azure DevOps Pipeline interface. At the top, it displays the pipeline path: All pipelines > ↑ Terraform-CD. Below this is a navigation bar with tabs: Pipeline, Tasks (which is selected), Variables, Retention, Options, and History. The main area shows a single stage named 'Dev' with the subtitle 'Deployment process'. Within this stage, there is an 'Agent job' section labeled 'Run on agent' with a '+' button. Below this are five sequential Terraform tasks: 'Install Terraform 1.0.5' (Terraform tool installer), 'Terraform : Init' (Terraform), 'Terraform : validate' (Terraform), 'Terraform : plan' (Terraform), and 'Terraform : apply -auto-approve' (Terraform). To the right of the pipeline details, there is a sidebar titled 'Stage name' containing a single entry: 'Dev'.

Azure IaC DevOps Releases

The screenshot shows the Azure DevOps Releases page for the "Terraform-CD" pipeline. The pipeline is currently set to "Prod". The page includes a search bar, navigation icons, and tabs for "Releases", "Deployments", and "Analytics". The main area displays a list of releases, each with a status badge (SS), release name, creation date, and stage status indicators.

Release	Created	Stages
Release-7 2021082...	25/08/2021, 15:59:34	Dev QA Stage Prod
Release-6 2021082...	25/08/2021, 15:32:50	Dev QA Stage Prod
Release-5 2021082...	25/08/2021, 15:14:29	Dev
Release-4 2021082...	25/08/2021, 15:01:11	Dev
Release-3 2021082...	25/08/2021, 14:55:53	Dev
Release-2 2021082...	25/08/2021, 14:51:55	Dev
Release-1 2021082...	25/08/2021, 14:43:41	Dev

Azure IaC DevOps Environment TF State Files

Microsoft Azure Search resources, services, and docs (G+)

Home > Resource groups > terraform-storage-rg > terraformstate201 >

 **tfstatefiles** ...

» Upload Change access level Refresh | Delete | Change tier | Acquire lease Brea

Authentication method: Access key ([Switch to Azure AD User Account](#))
Location: tfstatefiles

Search blobs by prefix (case-sensitive)

Add filter

Name	Modified	Access tier
<input type="checkbox"/> dev-terraform.tfstate	8/25/2021, 4:01:20 PM	Hot (Inferred)
<input type="checkbox"/> prod-terraform.tfstate	8/25/2021, 3:50:26 PM	Hot (Inferred)
<input type="checkbox"/> qa-terraform.tfstate	8/25/2021, 4:03:15 PM	Hot (Inferred)
<input type="checkbox"/> stage-terraform.tfstate	8/25/2021, 4:05:37 PM	Hot (Inferred)

Virtual Network Subnets - 4 Environments

	Name ↑↓	IPv4 ↑↓	
Dev	hr-dev-vnet-websubnet	10.1.1.0/24	QA
	hr-dev-vnet-bastionsu...	10.1.100.0/24	
	hr-dev-vnet-dbsubnet	10.1.21.0/24	
	hr-dev-vnet-appsubnet	10.1.11.0/24	
Stage	hr-stage-vnet-bastion...	10.3.100.0/24	Prod
	hr-stage-vnet-dbsubnet	10.3.21.0/24	
	hr-stage-vnet-websub...	10.3.1.0/24	
	hr-stage-vnet-appsub...	10.3.11.0/24	
	hr-qatest-vnet-bastionsub...	10.2.100.0/24	
	hr-qatest-vnet-dbsubnet	10.2.21.0/24	
	hr-qatest-vnet-websubnet	10.2.1.0/24	
	hr-qatest-vnet-appsubnet	10.2.11.0/24	
	hr-prod-vnet-bastionsub...	10.4.100.0/24	
	hr-prod-vnet-dbsubnet	10.4.21.0/24	
	hr-prod-vnet-websubnet	10.4.1.0/24	
	hr-prod-vnet-appsubnet	10.4.11.0/24	

4 Environments - Resources

Resource Groups

Name ↑↓
 hr-dev-rg-fzmqwz
 hr-prod-rg-qhddgg
 hr-qa-rg-vmsjwa
 hr-stage-rg-btcjcz

Public IP

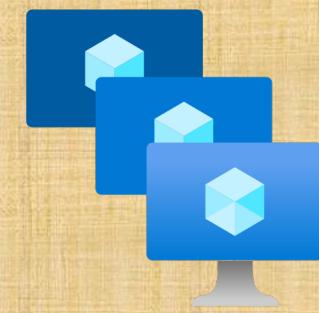
Name ↑↓
 hr-dev-web-linuxvm-publicip
 hr-prod-web-linuxvm-publicip
 hr-qa-web-linuxvm-publicip
 hr-stage-web-linuxvm-publicip

Virtual Machines

Name ↑↓
 hr-dev-web-linuxvm
 hr-prod-web-linuxvm
 hr-qa-web-linuxvm
 hr-stage-web-linuxvm



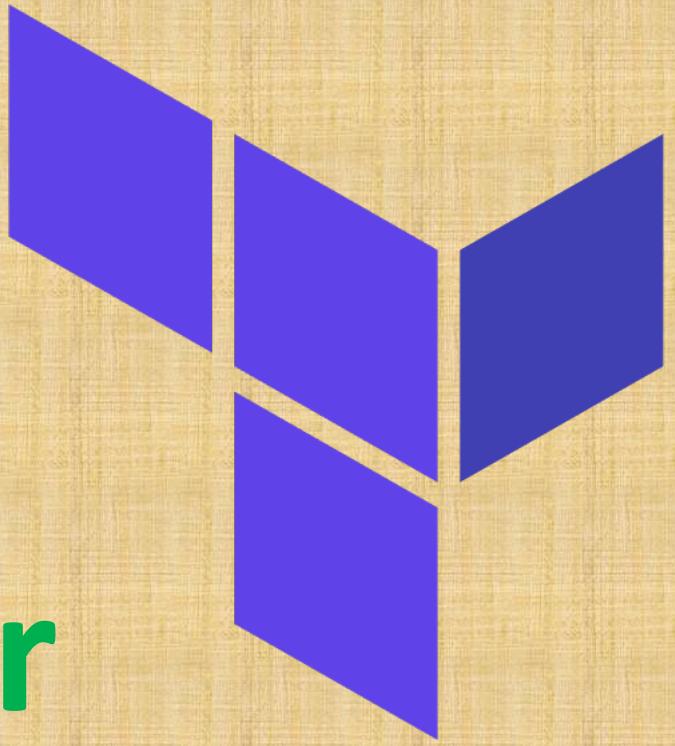
Application
Gateway



VM
Scale Sets



Azure MySQL Single Server



MySQL Server

Azure MySQL Deployment Options

Home > Azure Database for MySQL servers >

Select Azure Database for MySQL deployment option

Microsoft

How do you plan to use the service?



Single server

Best for a broad range of transactional workloads and web applications.

Production-ready, cost-optimized with built-in high availability, and backed by a service level agreement. Intelligent features to help optimize query performance. Advanced Threat Protection available to help secure your database.

[Create](#)

[Learn More](#)



Flexible server (Preview)

Best for workloads that require advanced customization and cost optimization.

Maximum control with a simplified developer experience. Supports custom maintenance windows, zone redundant high availability, and simple cost optimization. Flexible server is currently in preview.

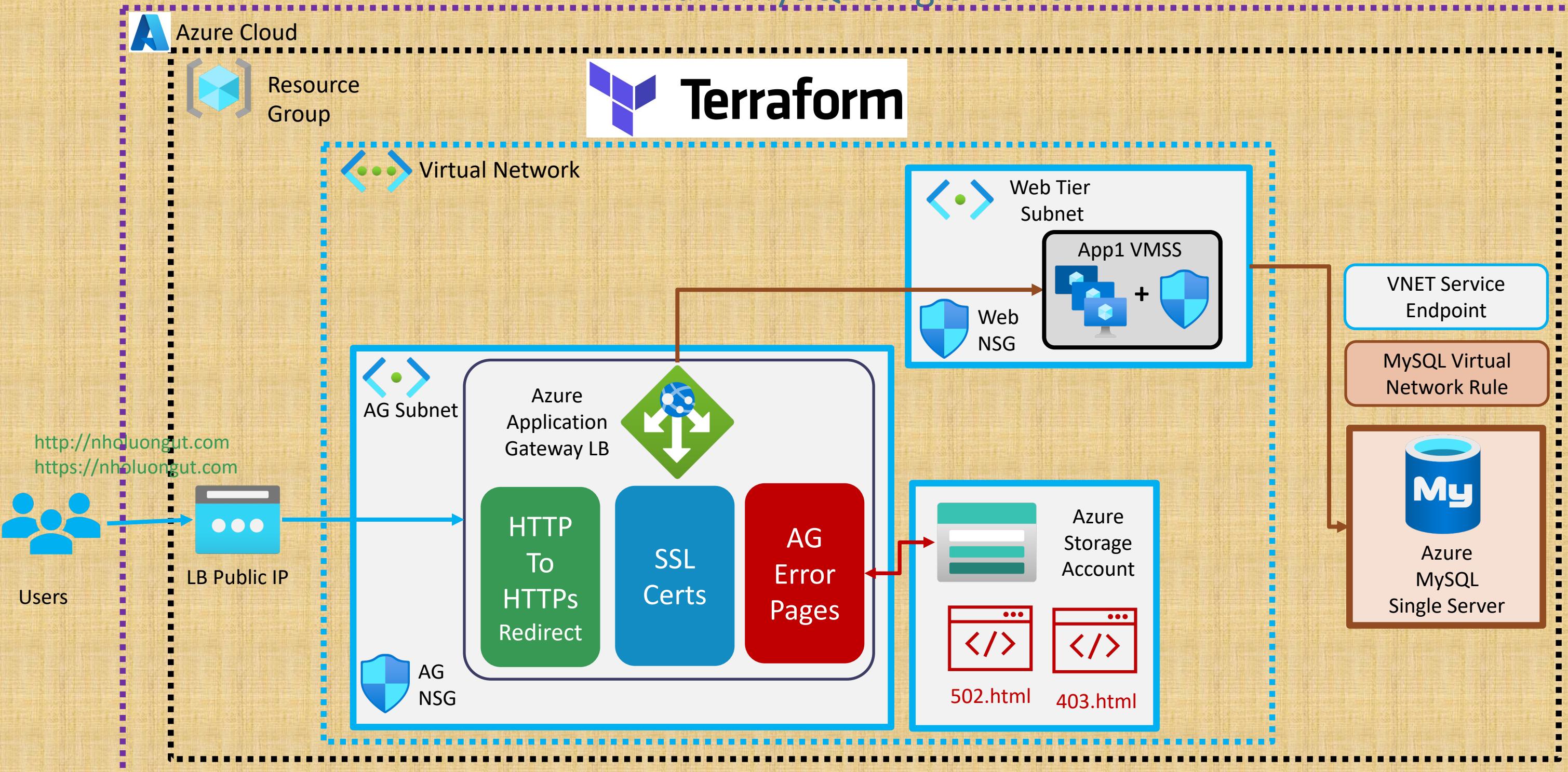
[Create](#)

[Learn More](#)

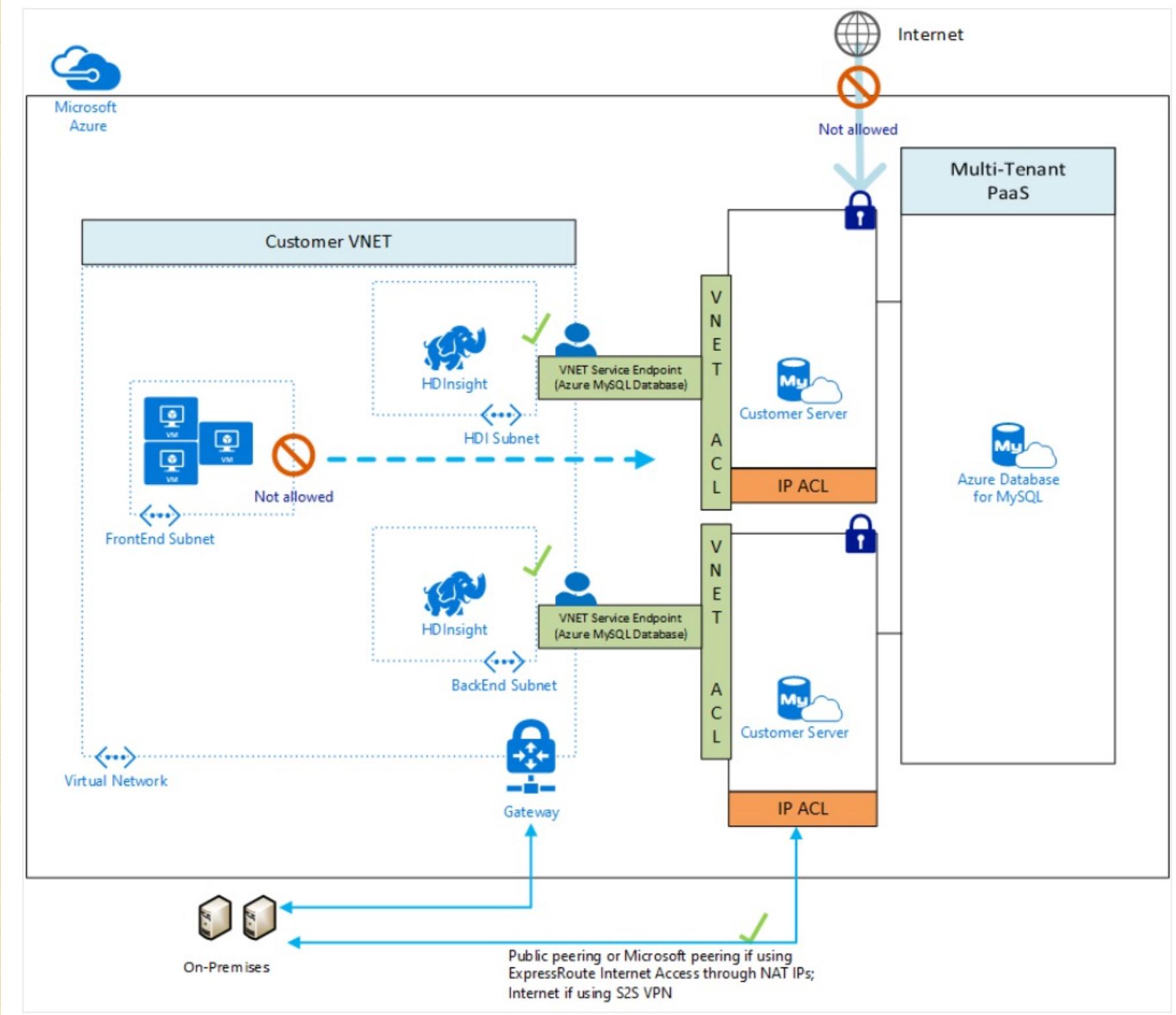
Terraform Resource not available as on today

...

Azure MySQL Single Server



Azure MySQL Single Server with Virtual Network Rules



Azure Resources

Azure MySQL Single Server



azurerm_mysql_server



azurerm_mysql_database



azurerm_mysql_firewall_rule



azurerm_mysql_virtual_network_rule

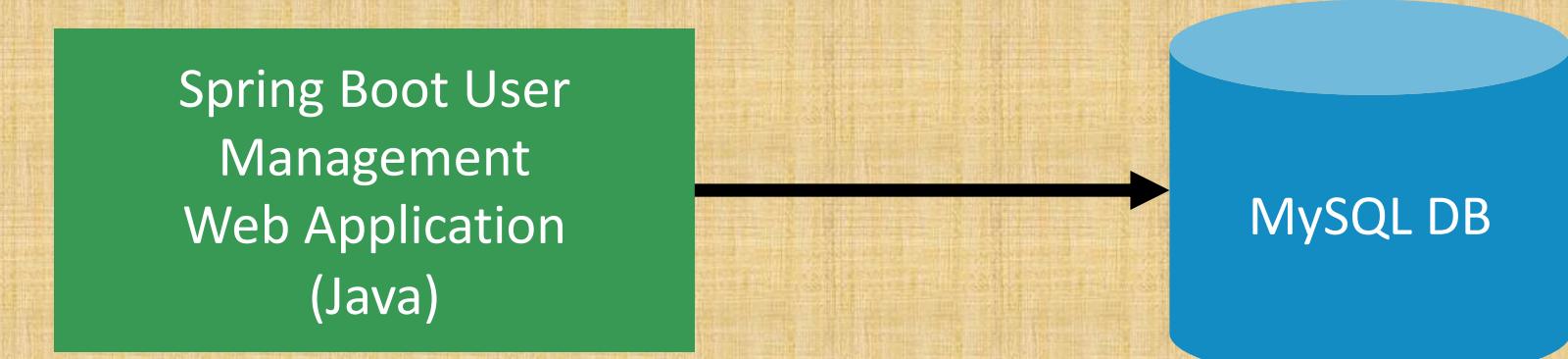
Terraform Concepts

Input Variables – name.auto.tfvars

Input Variables – -var-file=secrets.tfvars

Input Variables Sensitive Flag = True

User Management Web Application



UMS Web App with Create User, List User, Login and Logout Features

UMS Web App Listens on Port 8080

UMS Web App needs MySQL DB to store its users. If connection to DB fails, it cannot start

UMS Web App DB information can be passed via Environment Variables (DB Name, Port User, Pass)

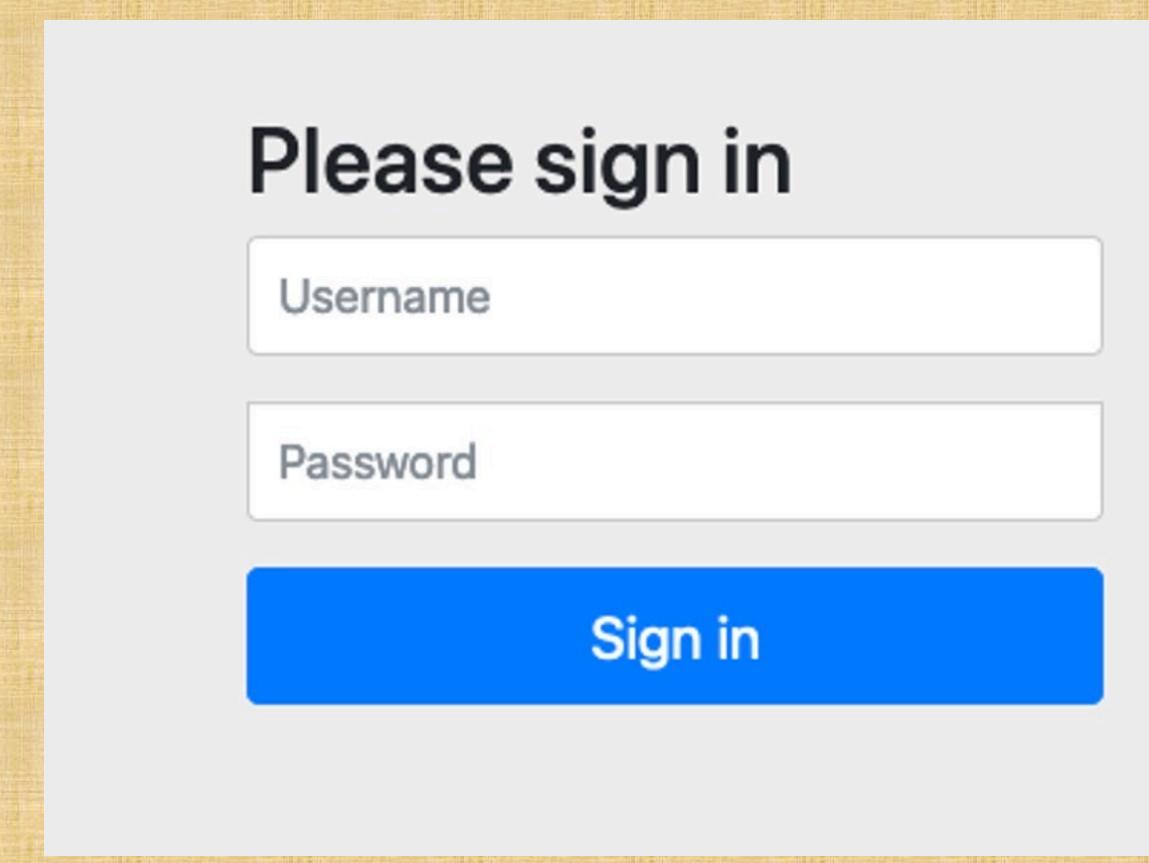
New users created will be stored in MySQL DB

We can login with new users created to UMS Web App

User Management Web Application Custom Data

```
webvm_custom_data = <<CUSTOM_DATA
#!/bin/sh
#sudo yum update -y
# Stop Firewall and Disable it
sudo systemctl stop firewalld
sudo systemctl disable firewalld
# Java App Install
sudo yum -y install java-11-openjdk
sudo yum -y install telnet
sudo yum -y install mysql
mkdir /home/azureuser/app3-usermgmt && cd /home/azureuser/app3-usermgmt
export DB_HOSTNAME=${azurerm_mysql_server.mysql_server.fqdn}
export DB_PORT=3306
export DB_NAME=${azurerm_mysql_database.webappdb.name}
export DB_USERNAME="${azurerm_mysql_server.mysql_serveradministrator_login}@${azurerm_mysql_
export DB_PASSWORD=${azurerm_mysql_server.mysql_serveradministrator_login_password}
java -jar /home/azureuser/app3-usermgmt/usermgmt-webapp.war > /home/azureuser/app3-usermgmt/
CUSTOM_DATA
```

UMS – Login Screen



UMS

Create Users Screen

Create User

User Name

Password

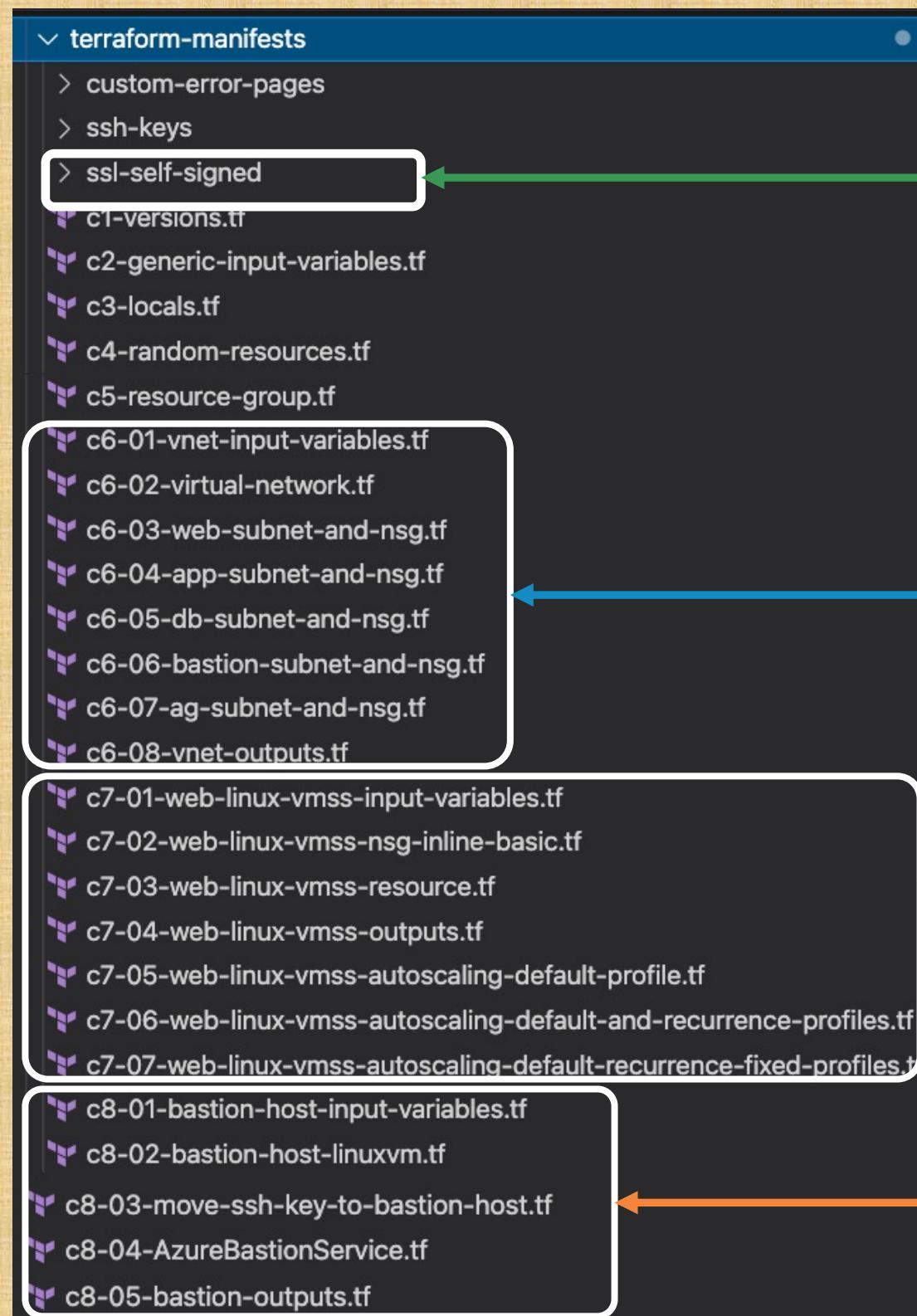
First Name

Last Name

Email Address

Social Security Number

Add Cancel



Terraform Configs

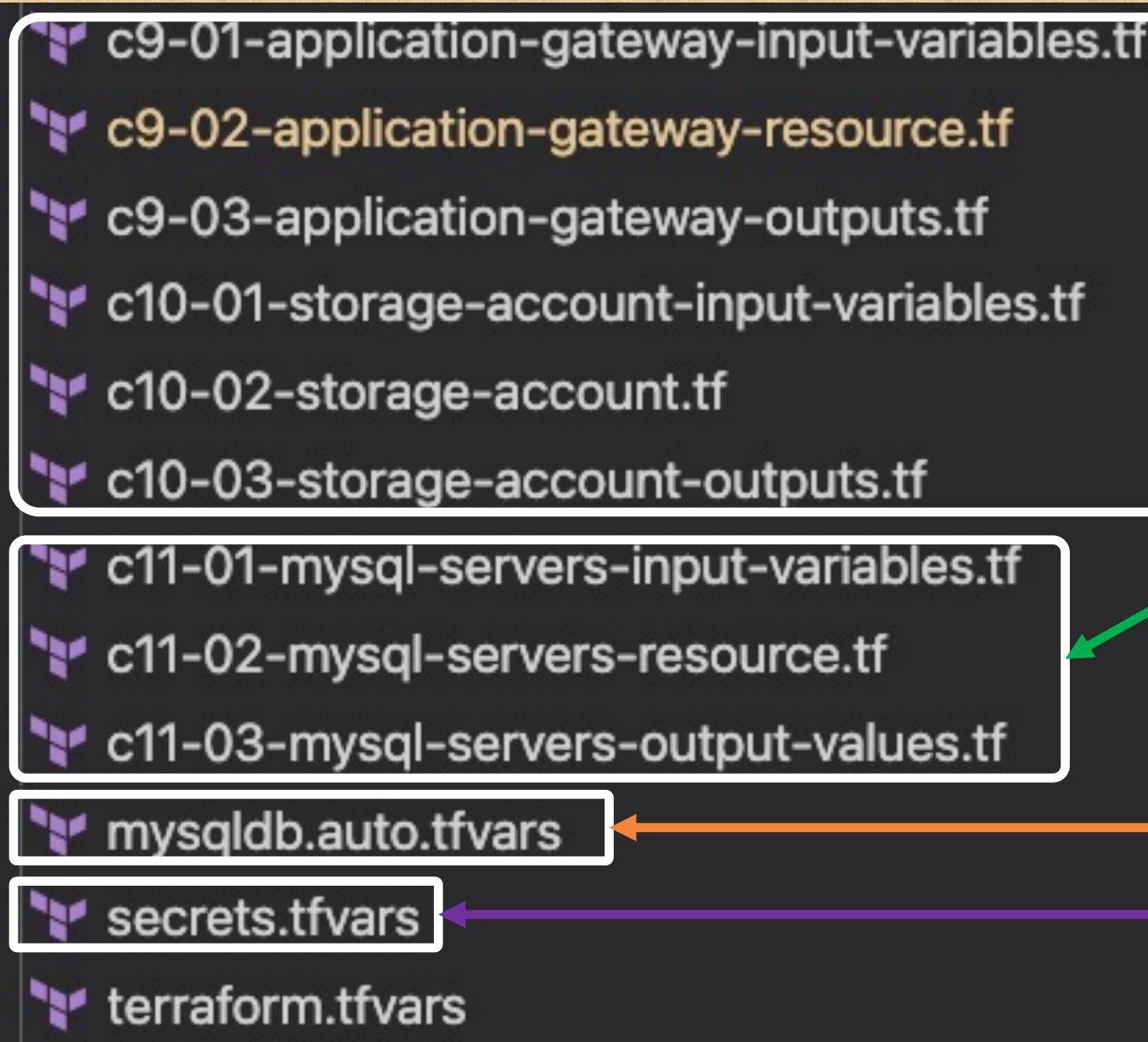
SSL Self-Signed Certificates

Azure Virtual Network with Subnets and Network Security Groups

Azure Virtual Machine Scalesets with Autoscaling enabled

Azure Bastion Linux VM – Enabled
Azure Bastion Service - Disabled

Terraform Configs



Azure Application Gateway +
Storage Account for AG Error Pages

Azure MySQL Single Server

Terraform Input Variables
mysqldb.auto.tfvars

Terraform Input Variables
secrets.tfvars

^ Azure MySQL Single Server with Terraform

- Step-01: Introduction to Azure MySQL Single Server using Terraform
- Step-02: Create MySQL Server Input Variables TF Configs 14:04
- Step-03: Create MySQL Server TF Configs 13:21
- Step-04: Create MySQL DB, FW Rule, Vnet Rule and Web Subnet Service Endpoint TF 09:01
- Step-05: Review VMSS Custom Data for Java App and VMSS Resource TF Configs 14:00
- Step-06: Review Application Gateway and Bastion Host TF Configs 09:14
- Step-07: Execute TF Commands and Verify Resources 22:48
- Step-08: Destroy Resources 08:00

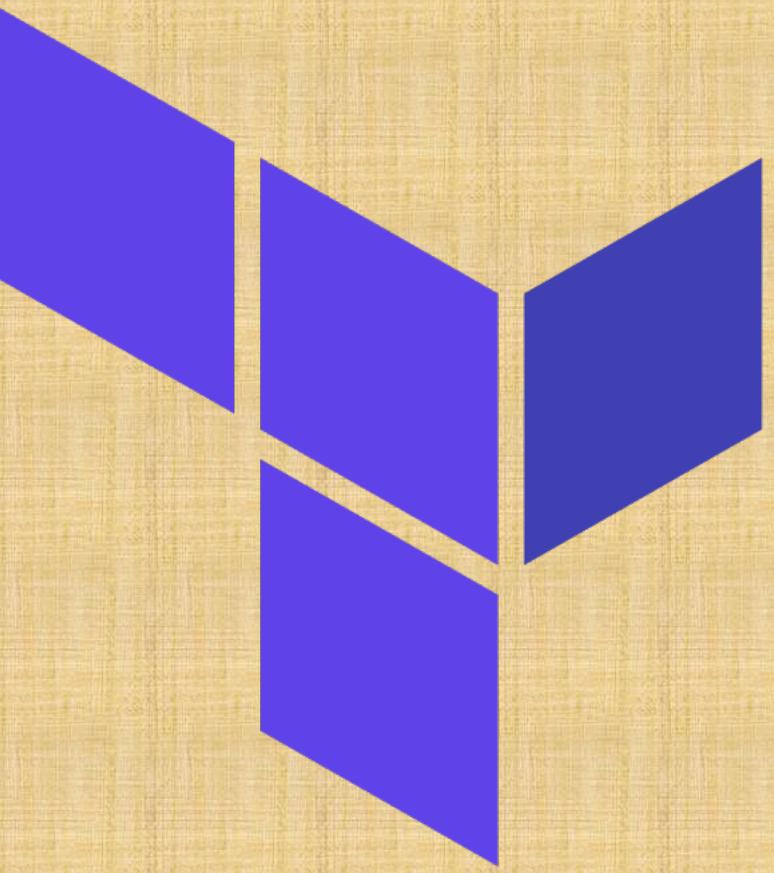
7 lectures • 1hr 30min

Time it takes to complete this Demo

Real-World Demo



Terraform Modules



Terraform Modules

Modules are **containers for multiple resources** that are used together. A module consists of a collection of .tf files kept together in a directory.

Modules are the main way to **package and reuse** resource configurations with Terraform.

Every Terraform configuration has at least one module, known as its **root module**, which consists of the resources defined in the **.tf files** in the **main working directory**.

A Terraform module (usually the **root module** of a configuration) can call **other modules** to include their resources into the configuration.

A module that has been **called by another module** is often referred to as a **child module**.

Child modules **can be called multiple times** within the same configuration, and **multiple configurations** can use the same child module.

In addition to modules from **the local filesystem**, Terraform can load modules from a **public or private registry**.

This makes it possible to **publish modules for others to use**, and to use modules that others have published.

Terraform Registry – Use Publicly Available Modules

The Terraform Registry hosts a broad collection of publicly available Terraform modules for configuring many kinds of common infrastructure.

Demo

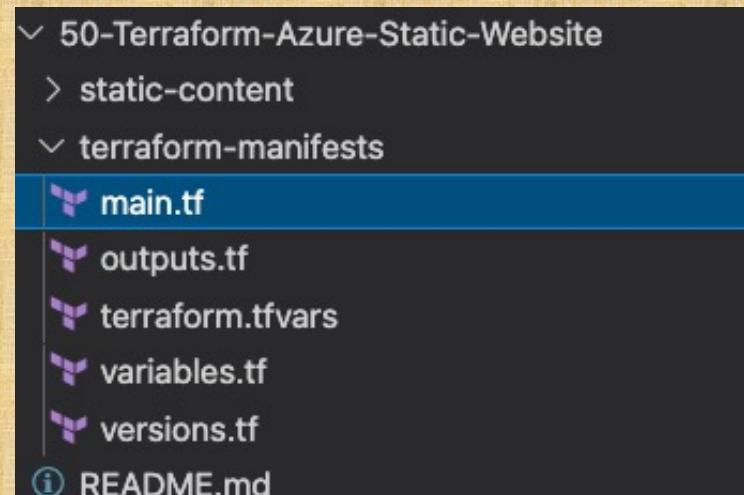
These modules are free to use, and Terraform can download them automatically if you specify the appropriate source and version in a module call block.

```
# Create Virtual Network and Subnets using Terraform Public Registry Modules
module "vnet" {
  source  = "Azure/vnet/azurerm"
  version = "2.5.0" # No versions constraints for production grade implementation
  vnet_name = local.vnet_name
  resource_group_name = azurerm_resource_group.myrg.name
  address_space      = ["10.0.0.0/16"]
  subnet_prefixes    = ["10.0.1.0/24", "10.0.2.0/24", "10.0.3.0/24"]
  subnet_names       = ["subnet1", "subnet2", "subnet3"]
  subnet_service_endpoints = {
    subnet2 = ["Microsoft.Storage", "Microsoft.Sql"],
    subnet3 = ["Microsoft.AzureActiveDirectory"]
  }
  tags = {
    environment = "dev"
    costcenter  = "it"
  }
  depends_on = [azurerm_resource_group.myrg]
}
```

Build manually and then automate using Terraform

1. Build a Static Website using Azure manually using Azure Portal.
2. Automate it using Terraform Resources

Demo



```
# Create Azure Storage account
resource "azurerm_storage_account" "storage_account" {
    name          = "${var.storage_account_name}${random_string.myrandom.id}"
    resource_group_name = azurerm_resource_group.resource_group.name

    location          = var.location
    account_tier       = var.storage_account_tier
    account_replication_type = var.storage_account_replication_type
    account_kind        = var.storage_account_kind

    static_website {
        index_document      = var.static_website_index_document
        error_404_document = var.static_website_error_404_document
    }
}
```

Build a Local Terraform Module

Demo

Build a **Local** Terraform Module and call it from **Root** Module and Test

```
✓ 51-Terraform-Modules-Build-Local-Module
  > static-content
  ✓ terraform-manifests
    > modules
      ✓ c1-versions.tf
      ✓ c2-variables.tf
      ✓ c3-static-website.tf
      ✓ c4-outputs.tf
  ⓘ README.md
```

```
# Call our Custom Terraform Module which we built earlier
module "azure_static_website" {
  source = "./modules/azure-static-website" # Mandatory

  # Resource Group
  location = "eastus"
  resource_group_name = "myrg1"

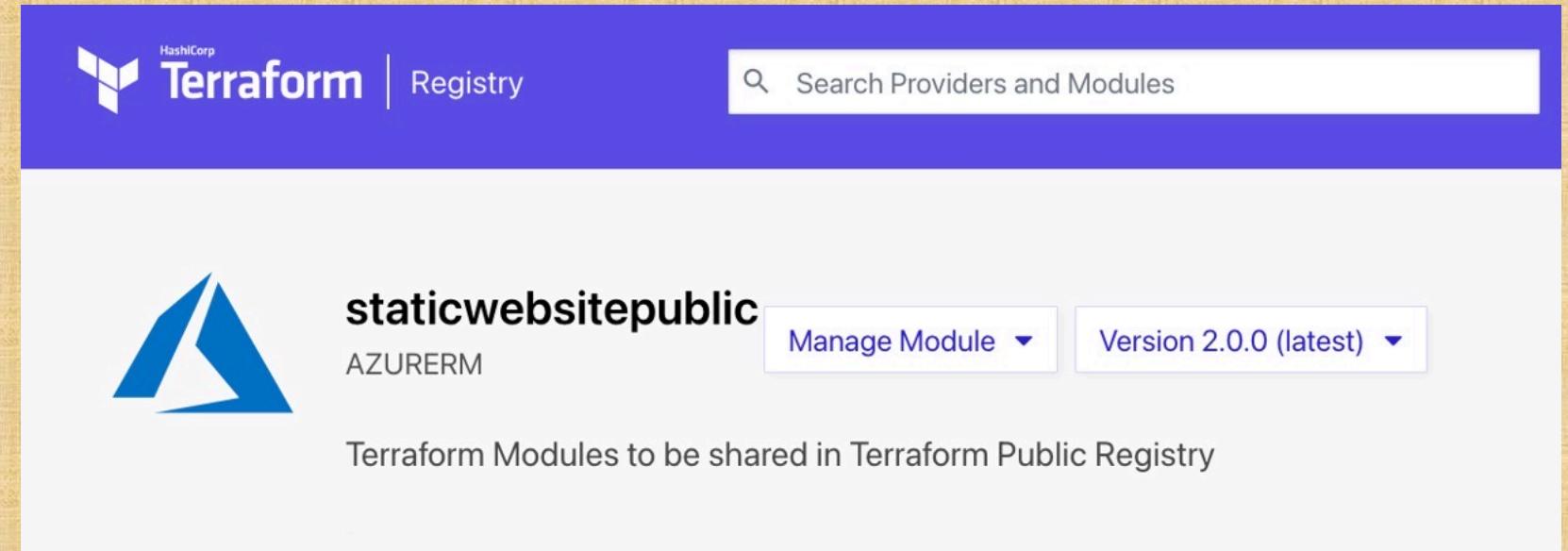
  # Storage Account
  storage_account_name = "staticwebsite"
  storage_account_tier = "Standard"
  storage_account_replication_type = "LRS"
  storage_account_kind = "StorageV2"
  static_website_index_document = "index.html"
  static_website_error_404_document = "error.html"
}
```

Publish to Public Terraform Registry

Demo

Publish to Public Terraform Registry and Access it from Public Registry and Test.

```
✓ 52-Terraform-Module-Publish-to-Public-Registry
  > static-content
  ✓ terraform-azure-static-website-module-manifests
    LICENSE
    main.tf
    outputs.tf
    README.md
    variables.tf
    versions.tf
  > terraform-manifests
    README.md
```



Use various Module Sources

Demo

In addition to Terraform Public and Private Registry use various module sources.

```
✓ 53-Terraform-Module-Sources
  ✓ terraform-manifests
    ✓ c1-versions.tf
    ✓ c2-variables.tf
    ✓ c3-static-website.tf
    ✓ c4-outputs.tf
  ⓘ README.md
```

```
# Call our Custom Terraform Module which we built earlier
module "azure_static_website" {

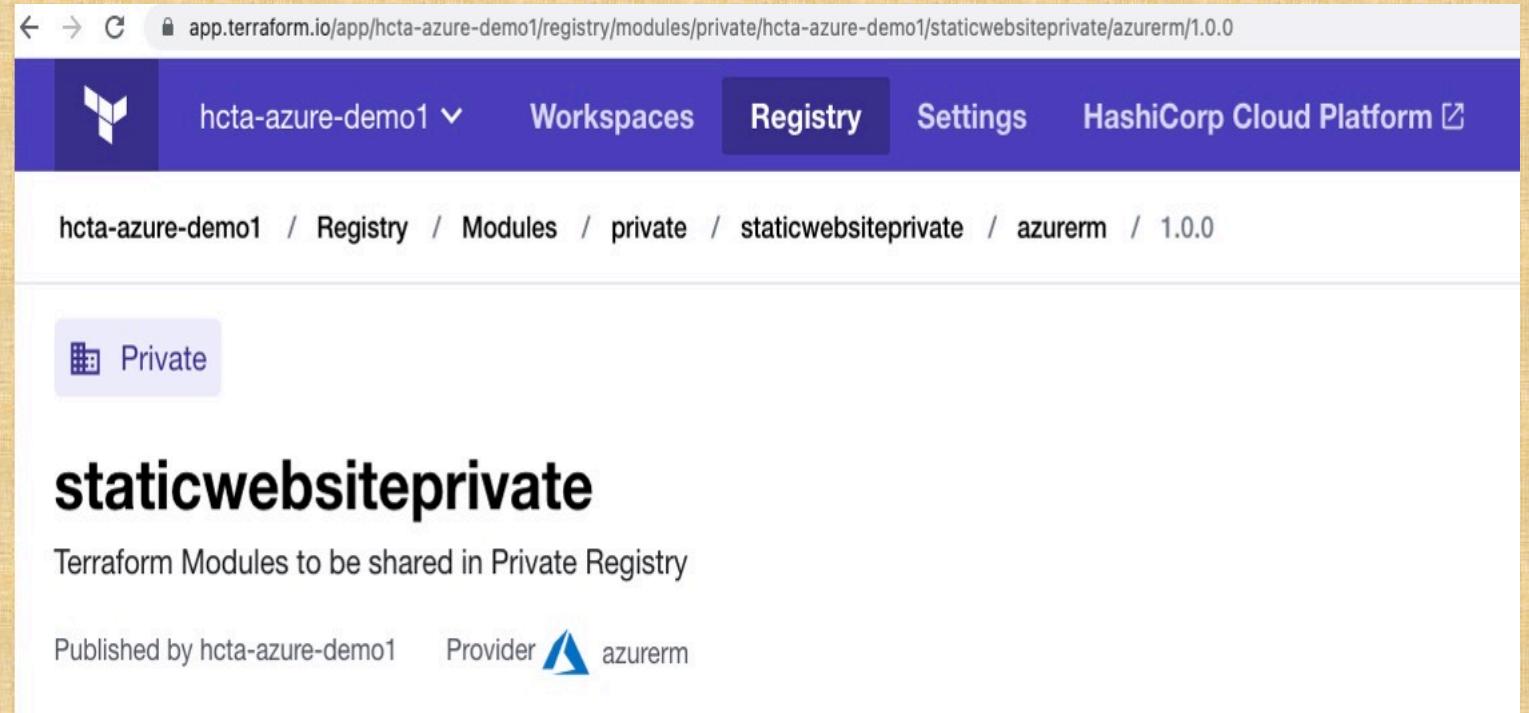
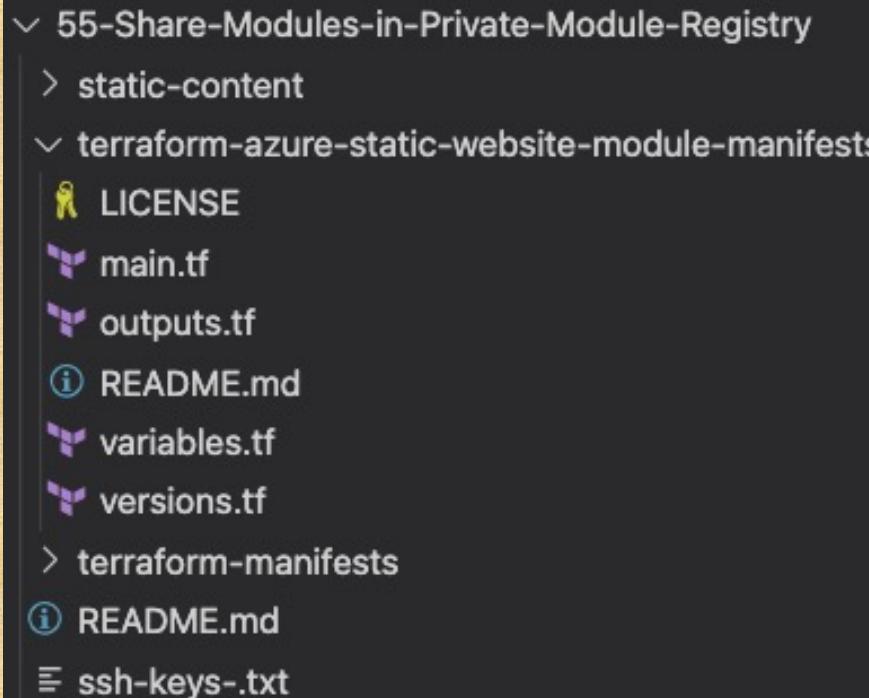
  # Terraform Local Module (Local Paths)
  #source = "./modules/azure-static-website"

  # Terraform Public Registry
  #source  = "stacksimplify/staticwebsitepb/azurerm"
  #version = "2.0.0"
```

Private Module Registry in Terraform Cloud & Enterprise

Members of your organization might **produce modules** specifically crafted for your own infrastructure needs.

Terraform Cloud and Terraform Enterprise both include a private module registry for sharing modules **internally** within your organization.



Demo

Practical Examples & Step-by-Step Documentation on Github

- ✓ 49-Terraform-Modules-use-Public-Module
 - > `terraform-manifests`
 - ❶ `README.md`
- ✓ 50-Terraform-Azure-Static-Website
 - > `static-content`
 - > `terraform-manifests`
 - ❶ `README.md`
- ✓ 51-Terraform-Modules-Build-Local-Module
 - > `static-content`
 - > `terraform-manifests`
 - ❶ `README.md`
- ✓ 52-Terraform-Module-Publish-to-Public-Registry
 - > `static-content`
 - > `terraform-azure-static-website-module-manifests`
 - > `terraform-manifests`
 - ❶ `README.md`
- ✓ 53-Terraform-Module-Sources
 - > `terraform-manifests`
 - ❶ `README.md`

- ✓ **Terraform Modules - Use Public Registry Module**
- ✓ **Terraform Azure Static Website**
- ✓ **Terraform Modules - Build Local Terraform Module**
- ✓ **Terraform Modules - Publish to Terraform Public Registry**
- ✓ **Terraform Module Sources**

4 lectures • 42min

3 lectures • 21min

4 lectures • 26min

4 lectures • 25min

1 lecture • 12min



Thank You