Step-00: Introduction

- **End Goal:** Build a Terraform Local Module

1. Create a Terraform module
2. Use local Terraform modules in your configuration
3. Configure modules with variables
4. Use module outputs
5. We are going to write a local re-usable module for the following usecase.

- **Usecase: Hosting a static website using Azure Storage Account**

1. Create a Azure Storage Account
2. Enable `Static webstite` option
3. Upload Static Content and test
4. For steps, 1 and 2 we are going to create a re-usable module in Terraform

- **How are we going to do this?**
- We are going to do this in 3 sections
- **Section-1 - Full Manual:** Create Static Website on Azure Storage Account using Azure Portal Management Console and host static content and test
- **Section-2 - Terraform Resources:** Automate section-1 using Terraform Resources
- **Section-3 - Terraform Modules:** Create a re-usable module for hosting static website by referencing section-2 terraform configuration files.

Module-1: Manual: Hosting a Static Website with Azure Storage Account

Step-01: Create Azure Storage Account

- Login to Azure Portal Console
- Go to Storage Accounts -> Create
- **Resource Group:** myrg-sw-1
- **Storage account name:** staticwebsitek123 (Should be unique across Azure)
- **Region:** East US
- **Performance:** Standard
- **Redundancy:** LRS
- Rest all leave to defaults
- Click on **Review + Create**
- Click on **Create**

Step-02: Enable Static Website

- Goto Storage Account -> staticwebsitek123 -> Data Management -> Static Website
- Click on **Enabled**
- **Index document name:** index.html
- **Error document path:** error.html

Step-03: Upload Static Content

- Goto Storage Account -> staticwebsitek123 -> Data Storage -> Containers -> $web
- Upload Static files from `static-content` folder

1. index.html
2. error.html

Step-05: Access Static Website

- Goto Storage Account -> staticwebsitek123 -> Data Management -> Static Website
- Get the endpoint name `Primary endpoint`

```
# Primary Endpoint
https://staticwebsitek123.z13.web.core.windows.net/
```

Step-06: Conclusion

- We have used multiple manual steps to host a static website on Azure Storage Account
- Now all the above manual steps automate using Terraform in next step

Module-2: Create Terraform Configuration to Host a Static Website on Azure

- We are going to host a static website on Azure Storage Account using general terraform configuration files
- Below will be the naming convention for Terraform Configs

1. versions.tf
2. main.tf
3. variables.tf
4. outputs.tf
5. terraform.tfvars

Step-01: versions.tf

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

Step-02: variables.tf

```
# Input variable definitions
variable "Location" {
```

```
  description = "The Azure Region in which all resources groups should be created."
  type = string
}
variable "resource_group_name" {
  description = "The name of the resource group"
  type = string
}
variable "storage_account_name" {
  description = "The name of the storage account"
  type = string
}
variable "storage_account_tier" {
  description = "Storage Account Tier"
  type = string
}
variable "storage_account_replication_type" {
  description = "Storage Account Replication Type"
  type = string
}
variable "storage_account_kind" {
  description = "Storage Account Kind"
  type = string
}
variable "static_website_index_document" {
  description = "static website index document"
  type = string
}
variable "static_website_error_404_document" {
  description = "static website error 404 document"
  type = string
}
variable "static_website_source_folder" {
  description = "static website source folder"
  type = string
}
```

Step-03: main.tf

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

# Random String Resource
resource "random_string" "myrandom" {
  length = 6
  upper = false
  special = false
  number = false
}

# Create Resource Group
resource "azurerm_resource_group" "resource_group" {
  name     = var.resource_group_name
  location = var.location
}

# 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
```

```
  }
}
```

Step-04: terraform.tfvars

```
location = "eastus"
resource_group_name = "myrg1"
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"
static_website_source_folder = "../static-content"
```

Step-05: outputs.tf

```
# Output variable definitions
output "resource_group_id" {
  description = "resource group id"
  value       = azurerm_resource_group.resource_group.id
}
output "resource_group_name" {
  description = "The name of the resource group"
  value       = azurerm_resource_group.resource_group.name
}
output "resource_group_location" {
  description = "resource group location"
  value       = azurerm_resource_group.resource_group.location
}
output "storage_account_id" {
  description = "storage account id"
  value       = azurerm_storage_account.storage_account.id
}
output "storage_account_name" {
  description = "storage account name"
  value       = azurerm_storage_account.storage_account.name
}
```

Step-06: Execute Terraform Commands

```
# Terraform Initialize
terraform init

# Terraform Validate
terraform validate

# Terraform Format
terraform fmt

# Terraform Plan
terraform plan

# Terraform Apply
terraform apply -auto-approve

# Upload Static Content
1. Go to Storage Accounts -> staticwebsitexxxxxx -> Containers -> $web
2. Upload files from folder "static-content"

# Verify
```

```
1. Azure Storage Account created
2. Static Website Setting enabled
3. Verify the Static Content Upload Successful
4. Access Static Website: Goto Storage Account -> staticwebsitek123 -> Data Management -> Static Website
5. Get the endpoint name `Primary endpoint`
https://staticwebsitek123.z13.web.core.windows.net/
```

Step-07: Destroy and Clean-Up

```
# Terraform Destroy
terraform destroy -auto-approve

# Delete Terraform files
rm -rf .terraform*
rm -rf terraform.tfstate*
```

Step-08: Conclusion

- Using above terraform configurations we have hosted a static website in Azure Storage Account in seconds.
- In next step, we will convert these **terraform configuration files** to a Module which will be re-usable just by calling it.