Step-01: Introduction

- We will build a Terraform local module to host a static website on Azure Storage Account.
- We will understand how to call a Local Re-usable module in to a Root Module.
- We will understand how the local module variables becomes the arguments inside a module block when it is called in Root Module `c3-static-webiste.tf`
- We will understand how we define the output values for a local module in a Root module `c4-outputs.tf`
- Terraform Comamnd `terraform get`
- Understand the differences between `terraform init` and `terraform get`

Step-02: Create Module Folder Structure

- We are going to create `modules` folder and in that we are going to create a module named `azure-static-website`
- We will copy required files from previous section for this respective module `50-Terraform-Azure-Static-Website\terraform-manifests` .
  - **Terraform Working Directory:** 51-Terraform-Modules-Build-Local-Module\terraform-manifests

modules

- Module-1: azure-static-website

1. main.tf
2. variables.tf
3. outputs.tf
4. README.md
5. LICENSE

- Inside `modules/azure-static-website` , copy below listed three files from `50-Terraform-Azure-Static-Website\terraform-manifests`

1. main.tf
2. variables.tf
3. outputs.tf
4. versions.tf

Step-03: Root Module: c1-versions.tf

- Call Module from Terraform Work Directory
- Create Terraform Configuration in Root Module by calling the newly created module
- c1-versions.tf
- c2-variables.tf
- c3-static-website.tf
- c4-outputs.tf

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

# Provider Block
```

```
provider "azurerm" {
 features {}
}
```

Step-04: c2-variables.tf

- Place holder file, if you want you can define variables.
- For now focus is on Calling the Local Terraform Module in to Root Module so we are not going to complicate the stuff here.
- We will leave this placeholder file

Step-05: c3-static-website.tf

- Arguments for this module are going to be the variables defined in `variables.tf` of local module

```
# 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"
}
```

Step-06: c4-outputs.tf

- Understand how we are going to reference the output values from a local module
- The output names defined in local module `outputs.tf` will be the values in this `c4-outputs.tf`

```
# Output variable definitions
output "root_resource_group_id" {
  description = "resource group id"
  value       = module.azure_static_website.resource_group_id
}
output "root_resource_group_name" {
  description = "The name of the resource group"
  value       = module.azure_static_website.resource_group_name
}
output "root_resource_group_location" {
  description = "resource group location"
  value       = module.azure_static_website.resource_group_location
}
output "root_storage_account_id" {
  description = "storage account id"
  value       = module.azure_static_website.storage_account_id
}
output "root_storage_account_name" {
  description = "storage account name"
  value       = module.azure_static_website.storage_account_name
}
```

Step-07: Execute Terraform Commands

```
# Terraform Initialize
terraform init
Observation:
1. Verify ".terraform", you will find "modules" folder in addition to "providers" folder
2. Verify inside ".terraform/modules" folder too.

# 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-08: Destroy and Clean-Up

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

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

Step-09: Understand terraform get command

- We have used `terraform init` to download providers from terraform registry and at the same time to download `modules` present in local modules folder in terraform working directory.
- Assuming we already have initialized using `terraform init` and later we have created `module` configs, we can `terraform get` to download the same.
- Whenever you add a new module to a configuration, Terraform must install the module before it can be used.
- Both the `terraform get` and `terraform init` commands will install and update modules.
- The `terraform init` command will also initialize backends and install plugins.

```
# Delete modules in .terraform folder
ls -lrt .terraform/modules
rm -rf .terraform/modules
ls -lrt .terraform/modules

# Terraform Get
terraform get
ls -lrt .terraform/modules
```

Step10: Major difference between Local and Remote Module

- When installing a remote module, Terraform will download it into the `.terraform` directory in your configuration's root directory.
- When installing a local module, Terraform will instead refer directly to the source directory.
- Because of this, Terraform will automatically notice changes to local modules without having to re-run `terraform init` or `terraform get`.