Step-01: Introduction

Terraform Block

- Understand about Terraform Block and its importance

Provider Block

- What are Terraform Providers?

Step-02: Understand about Terraform Settings Block

- Terraform Settings Block
- Required Terraform Version
- Provider Requirements
- Terraform backends

Step-03: Create a simple terraform block and play with required_version

- `required_version` focuses on underlying Terraform CLI installed on your desktop
- If the running version of Terraform on your local desktop doesn't match the constraints specified in your terraform block, Terraform will produce an error and exit without taking any further actions.
- By changing the versions try `terraform init` and observe whats happening

```
# Play with Terraform CLI Version (We installed 1.0.0 version)
  required_version = "~> 0.14.3" - Will fail
  required_version = "~> 0.14"   - Will fail
  required_version = "= 0.14.4"  - Will fail
  required_version = ">= 0.13"   - will pass
  required_version = "= 1.0.0"   - will pass
  required_version = "1.0.0"     - will pass
  required_version = ">= 1.0.0"   - will pass

# Terraform Block
terraform {
  required_version = ">= 1.0.0"
}

# To view my Terraform CLI Version installed on my desktop
terraform version

# Initialize Terraform
terraform init
```

Step-04: Terraform Providers

- What are Terraform Providers?
- What Providers Do?

Step-05: Provider Requirements

- Define required providers in Terraform Block

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

Step-06: Provider Block

- Create a Provider Block for Azure Resource Management `azurerm`

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

- Understand about Features Block in Provider Block

```
# Initialize Terraform
terraform init

# Validate Terraform Configuration files
terraform validate

# Execute Terraform Plan
terraform plan
```

Step-07: Create a simple Resource Block - c2-resource-group.tf

```
# Resource Block
# Create a resource group
resource "azurerm_resource_group" "myrg" {
  name = "myrg-1"
  location = "East US"
}
```

Step-08: Execute Terraform commands

```
# Initialize Terraform
terraform init

# Validate Terraform Configuration files
terraform validate

# Execute Terraform Plan
terraform plan

# Create Resources using Terraform Apply
terraform apply -auto-approve
```

Step-09: Clean-Up

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

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

References

- Terraform Providers
- Azure Provider Documentation
- Azure Resource Group Terraform Resource
- Terraform Version Constraints
- Terraform Versions - Best Practices