Terraform workspace

Background: Suppose, let say I have 3 environments, and all of them are managed by single subscription. [ say, Dev, Test and QA], Now when we would be running the command with environment specific things then statefile would be overwritten, to solve this concept we have terraform workspace concept.

Terraform workspaces are a feature that allow you to manage multiple environments (such as development, staging, and production) using a single Terraform configuration. Each workspace has its own state data, enabling isolated state management for different environments.

**Key Concepts of Terraform Workspaces**

1. **Default Workspace**: When you start using Terraform, there is a default workspace called default.
2. **Workspace Isolation**: Each workspace has its own state file.
3. **Commands for Managing Workspaces**: Terraform provides commands to create, switch, list, and delete workspaces.

**Common Commands**

**Create a New Workspace**

sh

terraform workspace new <workspace\_name>

Example:

sh

terraform workspace new dev

**List All Workspaces**

sh

terraform workspace list

**Select a Workspace**

sh

terraform workspace select <workspace\_name>

Example:

sh

terraform workspace select dev

**Show Current Workspace**

sh

terraform workspace show

**Delete a Workspace**

sh

terraform workspace delete <workspace\_name>

Note: You cannot delete the currently selected workspace. You must switch to another workspace first.

**Using Workspaces in Configuration**

When using workspaces, you can access the workspace name in your configuration using the terraform.workspace variable. This can be useful for dynamically setting variables or resource names based on the workspace.

**Example Configuration**

hcl

provider "aws" {

region = "us-west-2"

}

resource "aws\_s3\_bucket" "example" {

bucket = "my-app-${terraform.workspace}-bucket"

acl = "private"

}

output "bucket\_name" {

value = aws\_s3\_bucket.example.bucket

}

In this example, the S3 bucket name will include the workspace name, creating distinct buckets for different environments.

**Practical Use Case**

Suppose you have a Terraform configuration for an application, and you want to deploy it to separate environments (development, staging, production). You can use workspaces to manage this without duplicating your configuration.

**Step-by-Step Workflow**

**Create Workspaces**:

terraform workspace new dev

terraform workspace new staging

terraform workspace new prod

**Deploy to Development**:

terraform workspace select dev terraform apply

**Deploy to Staging**:

terraform workspace select staging

terraform apply

**Deploy to Production**:

terraform workspace select prod

terraform apply

**Best Practices**

* **Naming Conventions**: Use clear and consistent names for your workspaces, like dev, staging, and prod.
* **Workspace-specific Variables**: If you have variables that need to change per workspace, use conditionals in your Terraform configuration to set them based on the workspace name.
* **State Management**: Ensure that your state backend supports workspaces if you are using a remote backend. For example, S3 with DynamoDB for state locking.

**Example with Variable Overrides**

You can use workspace-specific variable files to override variables per workspace.

**Folder Structure**

.

├── main.tf

├── variables.tf

├── terraform.tfvars

├── dev.tfvars

├── staging.tfvars

└── prod.tfvars

**Variable Files**

**dev.tfvars**

hcl

instance\_type = "t2.micro"

**staging.tfvars**

hcl

instance\_type = "t2.medium"

**prod.tfvars**

hcl

instance\_type = "t2.large"

**main.tf**

hcl

variable "instance\_type" {

description = "Type of EC2 instance"

}

provider "aws" {

region = "us-west-2"

}

resource "aws\_instance" "example" {

ami = "ami-123456"

instance\_type = var.instance\_type

}

**Applying with Workspace-specific Variables**

sh

terraform workspace select dev

terraform apply -var-file="dev.tfvars"

terraform workspace select staging

terraform apply -var-file="staging.tfvars"

terraform workspace select prod

terraform apply -var-file="prod.tfvars"

Using workspaces helps manage different environments efficiently and keeps your Terraform configurations clean and organized.

Top of Form

Bottom of Form