# Managing Multiple Environments with Terraform

/

When Terraform deploys resources, it stores metadata to track the resources in a state file. So if we want to deploy resources for multiple environments such as production/QA/Staging, we need to differentiate deployment actions and state files as well. However with Terraform workspaces and Reusable modules architecture we can achieve separate environment provisioning.

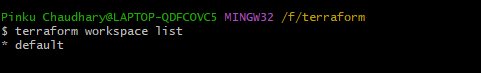
1. Terraform Workspaces- Terraform Workspaces allows us to manage separate state files for each workspace in the provided backend from a single source of configuration files
2. Reusable Modules- In this architecture, configuration files are stored inside a single directory. With module block we can source that directory and pass the variables.

So let's go in depth & explore Terraform workspaces,

Workspace does isolate the terraform.tfstate file from other workspaces in the same backend.

Below is an architecture with some configuration files for AWS resources and backend as s3 bucket-

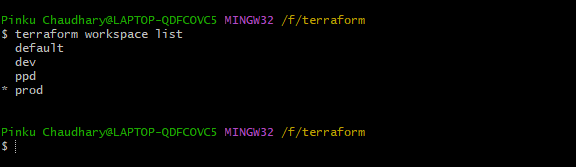
Right now, we have a workspace named default, which is default one and can’t be deleted ever-



Below are few [commands](https://www.terraform.io/docs/cli/commands/workspace/index.html) to manage workspaces,

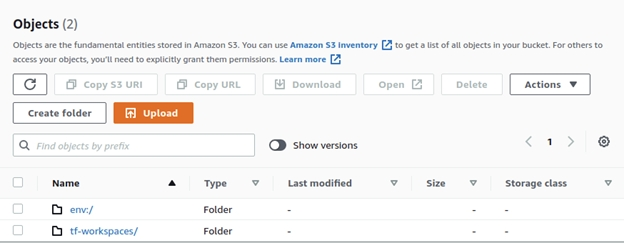
1. terraform workspace list --> To list workspaces
2. terraform workspace new --> To create a new workspace
3. terraform workspace select --> To select appropriate workspace
4. terraform workspace delete --> To delete workspace

I created some other workspaces. Dev & prod-

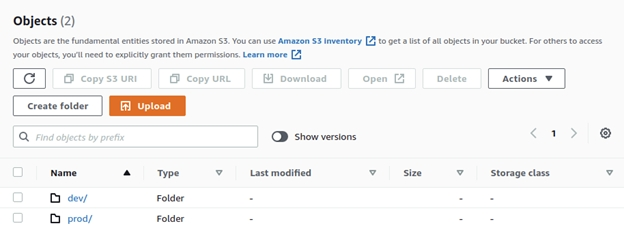


In this demo, we used s3 as the backend to store state files. Configure s3 bucket, region and key in the backend block of terraform. Configure AWS cli profile, region etc. in the AWS provider block

Based on the key which is defined in the backend block, inside the bucket it will create a defined path for default workspace and env folder for workspaces-



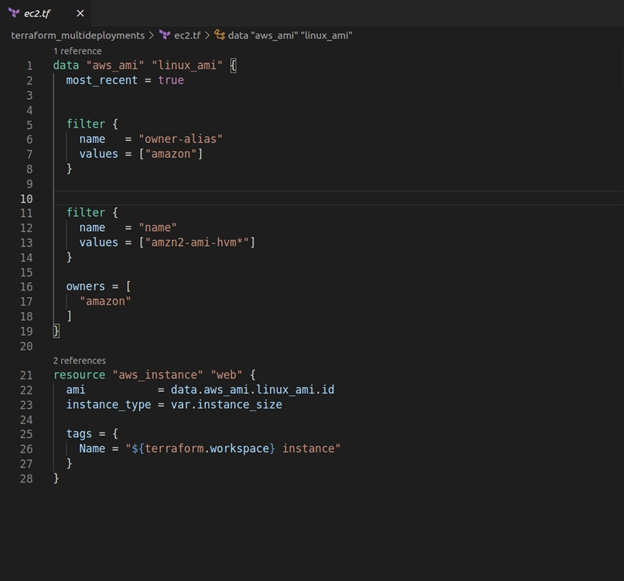
Inside env, it will create other folders for each workspace and store the terraform.tfstate file-



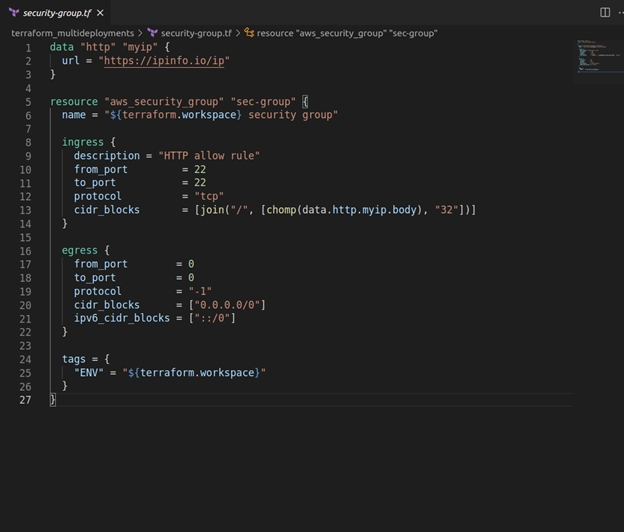
We have below resource blocks and variables configuration. Now let’s launch an ec2 instance and security group from terraform configuration files-

EC2 configuration:

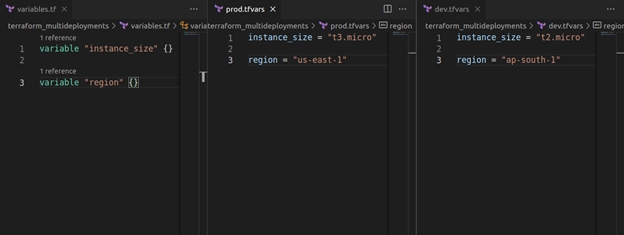
In the configuration below, we have a data block to fetch amazon linux ami id to launch the instance from that ami, and the resource block to configure launch of ec2 instance. In the resource block, we provided “${terraform.workspace}” to provide the current workspace name to the specific parameters in the specified block



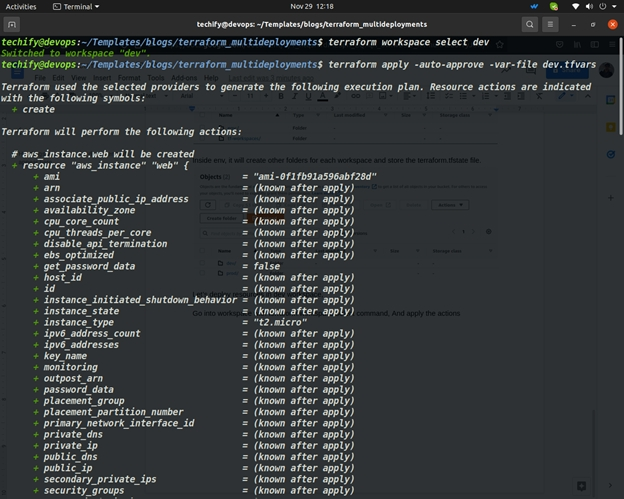
Below is the configuration of the security group, it will create a security group upon below configuration-



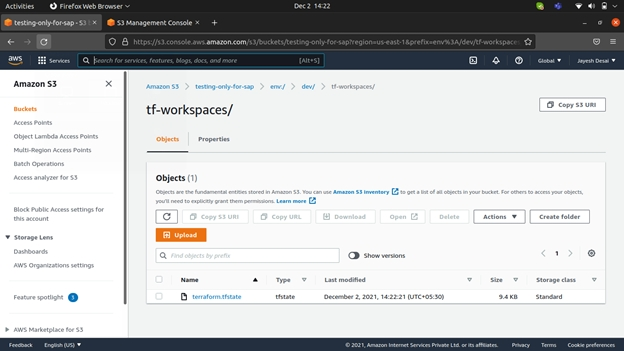
We have configured variables.tf file to expose variables and we will provide values of those variables in tfvars files. We created dev.tfvars for dev and prod.tfvars for prod workspace-



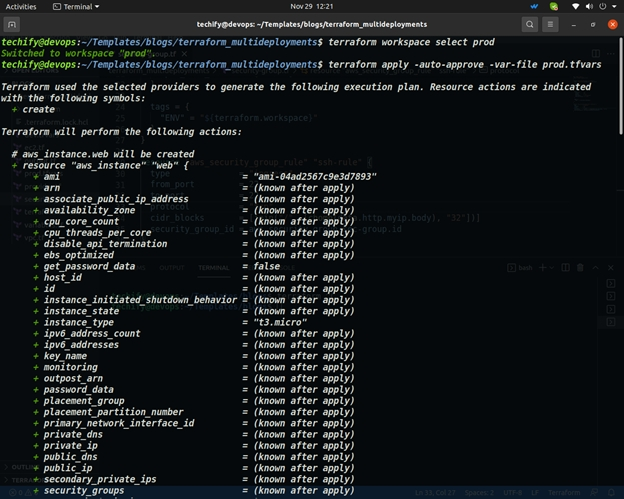
Let’s deploy resources in dev workspace, Let’s switch workspace to dev workspace with command “terraform workspace select dev”, and give terraform plan to view execution plan of defined AWS resources, then apply if everything is as expected-



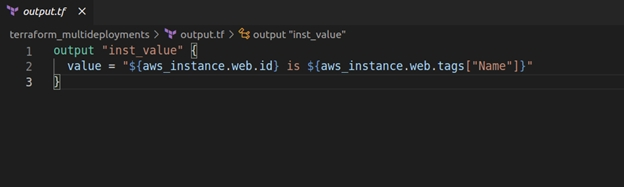
After apply completes, it will store the metadata in the state file-



Let’s deploy same resource in prod workspace-

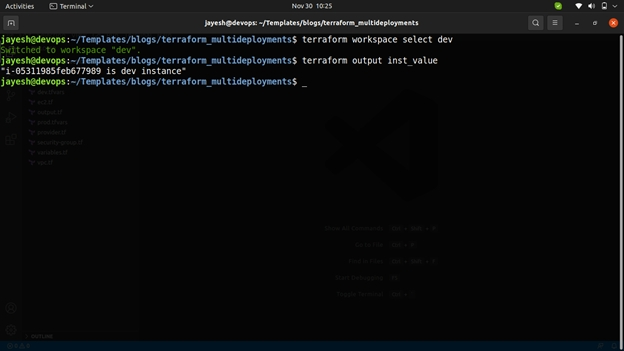


I created an output block for instance name and tags output-

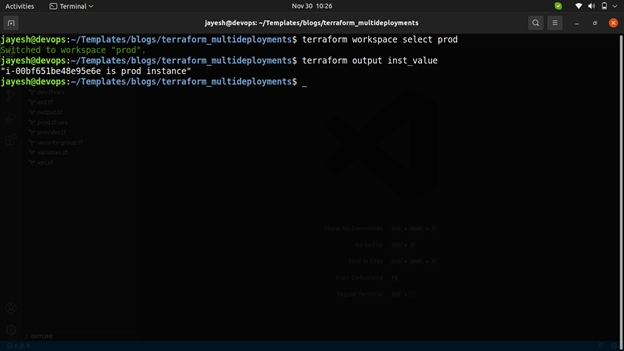


Let’s check that output value in both workspaces-

●Dev workspace:



●Prod workspace:



o, this is the first way to manage multiple environments deployments with terraform workspaces. Now let’s look at the other method to manage multiple environments.