Terraform Project Structuring

HashiCorp
Terraform

Terraform Project Structures

- Project structuring and organization is important
- As infrastructure gets larger, more environments, multi region - greater emphasis on organization
- Project structuring is continuous journey
  - No single answer fits all
  - No right or wrong way, but some have more advantages than others
  - Depends on where you are, and how large your infrastructure is

HashiCorp Terraform

Terraform Project Structures

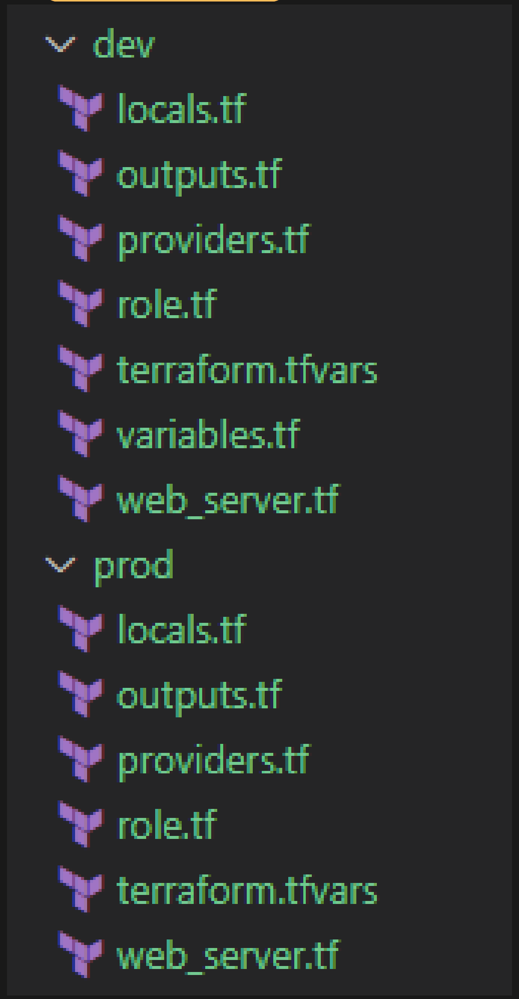
- The way project is structured and organized, directly affects your state file
  - Troubleshooting or editing state files, gets harder the larger the state becomes
- Good organized projects can help:
  - Makes infrastructure easier to read and create
  - Allow state files to be isolated
  - Provide context as to how environments are isolated, and responsibility of infrastructure

HashiCorp Terraform

Terraform State

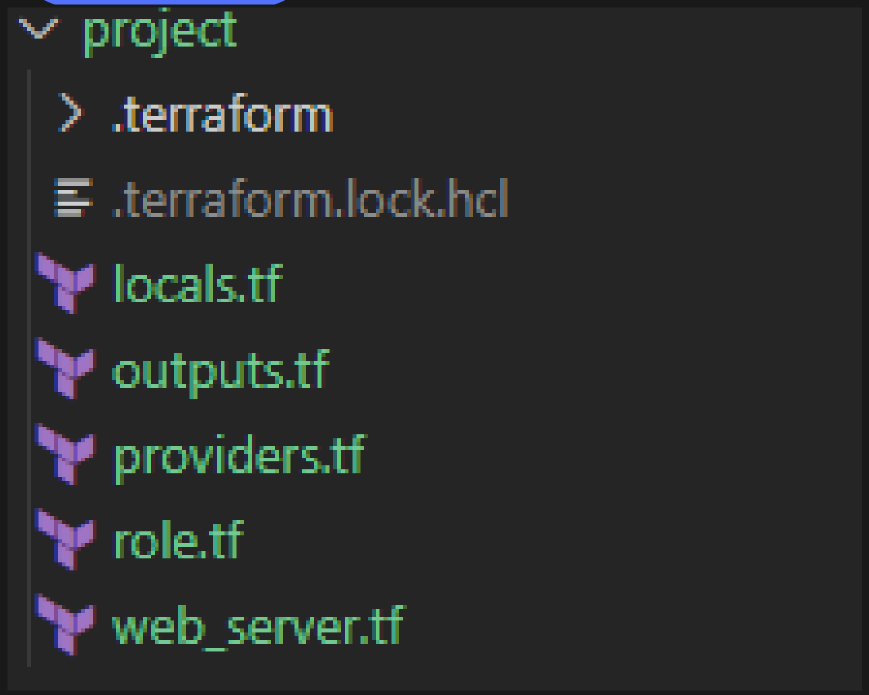
- There are three general organizational layouts - Simple, Flat, Segmented
- Simple organization:
  - Single region and environment
  - Place all code in single directory
  - One state file for all infrastructure
  - For bare bones, light weight infrastructure
- Flat organization:
  - Divide infrastructure based on environment and region
  - State file per environment and region
  - Code lives in multiple directories
  - For infrastructure spanning multiple environments and region

Flat



```
dev
  locals.tf
  outputs.tf
  providers.tf
  role.tf
  terraform.tfvars
  variables.tf
  web_server.tf
prod
  locals.tf
  outputs.tf
  providers.tf
  role.tf
  terraform.tfvars
  web_server.tf
```

Simple

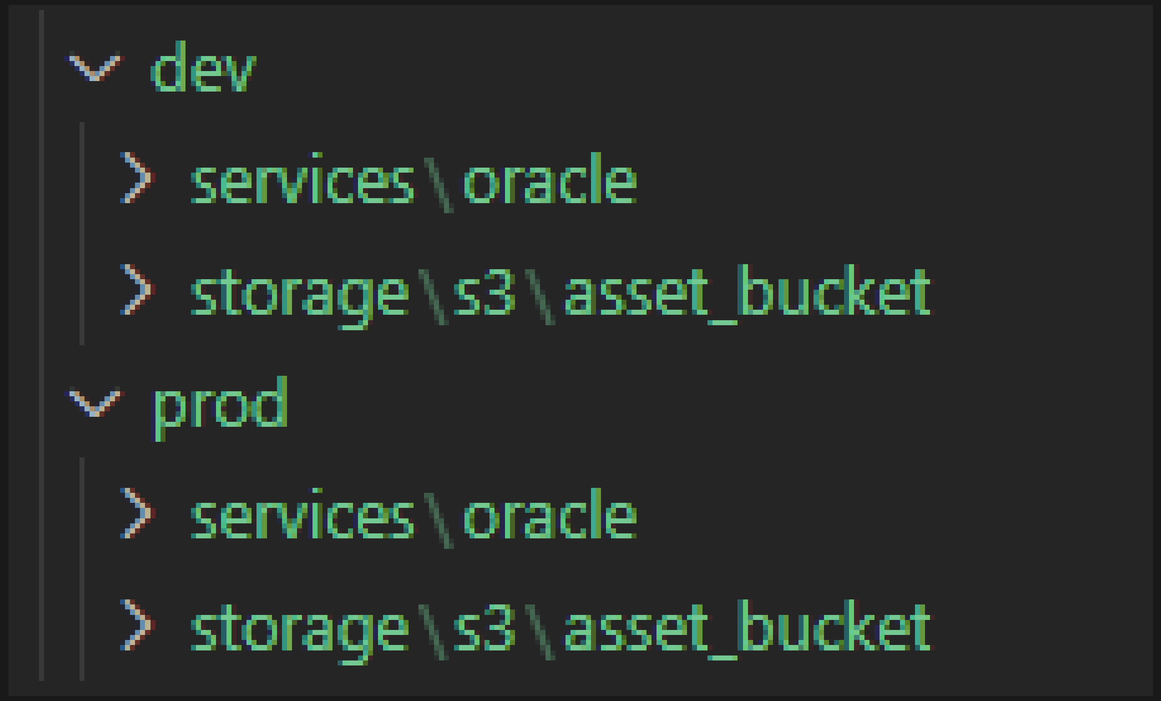


```
project
  .terraform
  .terraform.lock.hcl
  locals.tf
  outputs.tf
  providers.tf
  role.tf
  web_server.tf
```

HashiCorp Terraform

Terraform Project Structures

- Segmented organization:
  - Organize based on environment, region, and category
  - Category is abstract - can be resource type, resource class, application services, etc.
  - Category can be as granular as you want
  - State files broken down into smaller pieces
  - Each project / configuration services specific purpose
  - Relies heavily on automation
  - Difficult to manage if running Terraform manually



```
v dev
  > services\oracle
  > storage\s3\asset_bucket
v prod
  > services\oracle
  > storage\s3\asset_bucket
```

HashiCorp Terraform

Terraform Project Structures

- Simple Organization
  - ✔ Advantages
    - Great for starting out, small infrastructure in single environment
  - Disadvantage:
    - Code gets difficult to read as infrastructure gets larger
    - Relies heavily on provider aliases
    - State file grows larger, making it more difficult to read and troubleshoot
    - Error blast radius affects entire infrastructure
    - Mixing environment infrastructure in one project
    - Difficult to scale as more people work on same project
      - Continuous state locking

HashiCorp
Terraform

Terraform Project Structures

- Flat Organization
  - ✔ Advantages
    - Separate infrastructure per environment and region
      - Clear distinction of infrastructure between environment
    - No more single State file - State file per environment and region
    - Code is easier to manage and read, until a certain point
    - Relies on creating Modules to reduce duplication

    Disadvantage:
    - As infrastructure gets larger, state file becomes difficult to manage
      - Code readability and management suffers
    - Blast radius is still an issue - mistake affects entire environment
    - State locking can be an issue
      - One person working on Dev infrastructure locks state
      - Dev environment is blocked until state is unlocked

Terraform Project Structures

- Segmented Organization
  - ✔ Advantages
    - Clear distinction of infrastructure responsibility
      - Categorized by environment, region, and type
    - State files kept small, easier to read and manage
      - State locking is non-issue
    - Blast radius is contained to only specific infrastructure
    - Relies on creating Modules to reduce duplication
    - Relies heavily on automation
    - Easy to scale
  - Disadvantages
    - Requires good engineering discipline and creating best practices
    - Can be confusing to understand initially
    - Relies heavily on automation - needs CICD to deploy infrastructure
      - Terraform Cloud, Spacelift, Atlantis, etc.

HashiCorp Terraform