TAMING IAC PROJECTS WITH STACKS

MOIN

IT since 2001

Solutions Architect/Teamlead/Automation Nerd/...

First proper automation project in 2003

INTRODUCTION TO TERRAFORM

- Declarative solution for Infrastructure as Code
- HCL (Hashicorp Configuration Language)
- State files store current state
- Modules, Providers, Resources
- Wide adoption with Cloud Deployments (e.g. AWS, Azure, GCP)

WHEN DO WE USE TERRAFORM?

- IaC adoption in automation design
- Intent-based orchestrators (e.g. ACI, Meraki)
    - Needs 99%+ model-driven configuration coverage.
    - Terraform provider is vendor provided and maintained.
    - Terraform provider supports all necessary features
- Customer requirement

VANILLA TERRAFORM PROJECT STRUCTURES

...AND THEIR CHALLENGES

SINGLE STATE FILE

- Performance: More objects = longer run times
- Resiliency: State file corruption effects whole environment

MULTIPLE STATE FILES

- Multiple State Files = Multiple Terraform projects
- Maintainability: Complex project structure
- Not DRY: Code duplication, e.g. providers, modules

STACKS

BENEFITS

- Segmentation of the project into multiple chunks (stacks)
- State file per stack -> smaller blast radius
- Stack size and structure is flexible
- Each stack is like an independent Terraform project
- Efficient dependency management

TOOLS

- Terragrunt - Slim Wrapper for Terraform
- Terramate - Orchestrate Terraform native stacks
- Terraspace - Framework similar to Terramate

STACKS IN ACTION - ACI+TERRAFORM+TERRAMATE

PROJECT STRUCTURE

```
.
├── config
├── modules
├── stacks
│   ├── fabric
│   ├── snapshot
│   ├── tenants
│   └── vn
│   └── backend.tm.hcl
├── README.md
├── credentials.tm.hcl
├── globals.tm.hcl
├── terramate.tm.hcl
└── ...
```

BACKEND CONFIGURATION

```
generate_hcl "_versions.tf" {
  content {
    terraform {
    required_version = ">= 1.6"

    required_providers {
      aci = {
      source  = "ciscodevnet/aci"
      version = "~> 2.17.0"
      }
      vault = {
      source  = "hashicorp/vault"
      version = "~> 5.3.0"
      }
    }
```

MODULE CONFIGURATION

```
# contracts.tf.hcl

generate_hcl "_contracts.tf" {
  stack_filter {
    project_paths = [
    "stacks/tenants/*"
    ]
  }

  content {
    locals {
    contracts = fileexists( \
      tm_format("../../../config/tenants/%s/contracts.yml",
      ? yamldecode(file( \
      tm_format("../../../config/tenants/%s/contracts.yml",
```

STACK CONFIGURATION

```
# stack.tm.hcl

stack {
  name        = "103_Tn1"
  description = "103_Tn1"
  id          = "6c9efb97-xxxx-xxxx-xxxx-a84fb0843bd5"
  tags        = ["tenant", "mytenant", "tn_103", "prod"]
  after       = ["tag:snapshot"]
  }

  import {
  source = "/modules/tenant/tenant.tm.hcl"
  }
  ...
  import {
```

GENERATE TERRAFORM STACKS

```
terramate generate
```

```
.
├── ...
├── stacks
│   ├── ...
│   ├── tenants
│   │   ├── 103_Tn1_Mig
│   │   │   ├── 103_VRF1
│   │   │   │   ├── ...
│   │   │   ├── _contracts.tf
│   │   │   ├── _credentials.auto.tfvars
│   │   │   ├── globals.tm.hcl
│   │   │   ├── _providers.tf
│   │   │   ├── stack.tm.hcl
│   │   │   ├── _variables.tf
│   │   │   ├── _versions.tf
│   │   │   └── ...
│   │   ├── 106_Tn1_Mig
│   │   │   └── ...
│   │   └── ...
│   └── ...
│   └── backend.tm.hcl
└── terramate.tm.hcl
```

RUN THE DEPLOYMENT

Initialize Terraform Stacks

```
terramate run terraform init
```

Terraform Plan on all stacks

```
terramate run terraform plan -o terraform.tfplan
```

Terraform Apply on all stacks

```
terramate run terraform apply -auto-approve terraform.tfplan
```

OTHER COMMANDS

Deploy only Tenant and VRF stacks

```
terramate run --tags tenant,vrf terraform apply -auto-approve
```

Deploy Tenants except Common Tenant

```
terramate run --tags tenant,vrf --no-tags common terraform app
```

Terraform Apply stacks with config changes (git diff)

```
terramate run --changed terraform apply -auto-approve terrafor
```

MERAKI EXAMPLE

```
.
├── ...
├── orgs
│   ├── org-tyrell
│   │   ├── ...
│   ├── org-weylandyutani
│   │   ├── site-london
│   │   ├── site-tokyo
│   │   │   ├── _appliance.tf
│   │   │   ├── _network.tf
│   │   │   ├── _wireless.tf
│   │   │   ├── globals.tm.hcl
│   │   │   ├── _providers.tf
│   │   │   ├── stack.tm.hcl
│   │   │   ├── _variables.tf
```

STACK RELATED RECOMMENDATIONS

- Align stacks with infrastructure blocks/segments.
- Mirror config and stack structure.
- Use stack tool features, e.g. Terramate globals, functions, scripts.

TERRAFORM RELATED RECOMMENDATIONS

- Centralized statefile storage with encryption.
- Vault for secrets/keys/certificates.
- Store plan files and logs as artifacts.

SUMMARY

- Terraform/Terramate for NetAuto is possible.
- Validate, if the setup fits the project.
- Project structure needs proper design.
- Stacks improve maintainability and speed.