

Module 5

Templates, State and Drift



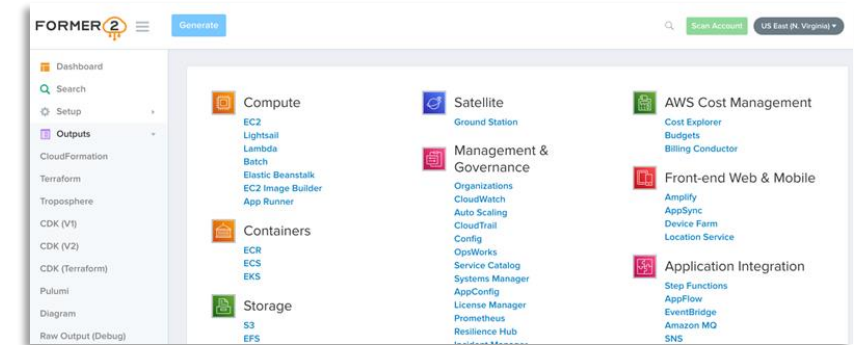
Agenda

- Importing, Recreating and Terminating Resources
- Templates
- OSS Workspaces
- Local vs. Remote State files
- Managing State and Configuration Drift
- Logging
- Lab 5



Importing, Recreating and Terminating Resources

- Resources may already have been created and configured outside of terraform
- Identify Resources which remain independent of Terraform and those which need to be brought under Terraform control
- Consider importing resources into terraform



Terraform Import

- Terraform can import existing infrastructure resources. This functionality lets you bring existing resources under Terraform management.
- Terraform import can only import resources into the state. Importing does not generate configuration.
- Before you run terraform import you must manually write a resource configuration block for the resource. The resource block describes where Terraform should map the imported object.
- Must run locally, not supported in Terraform Cloud
- Be careful.. Here be dragons!! <https://developer.hashicorp.com/terraform/cli/import>

<https://spacelift.io/blog/importing-existing-infrastructure-into-terraform>

This is a third-party blog, and no warranty/assurance is given as to its validity



Templates

- Questions to consider

How large is your main.tf?

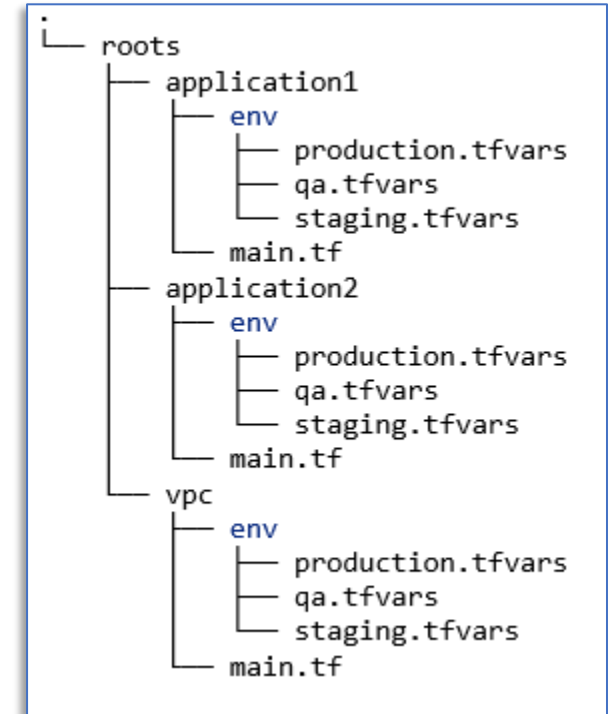
How complex is your architecture that is being deployed and maintained by Terraform?

How complex is your team?

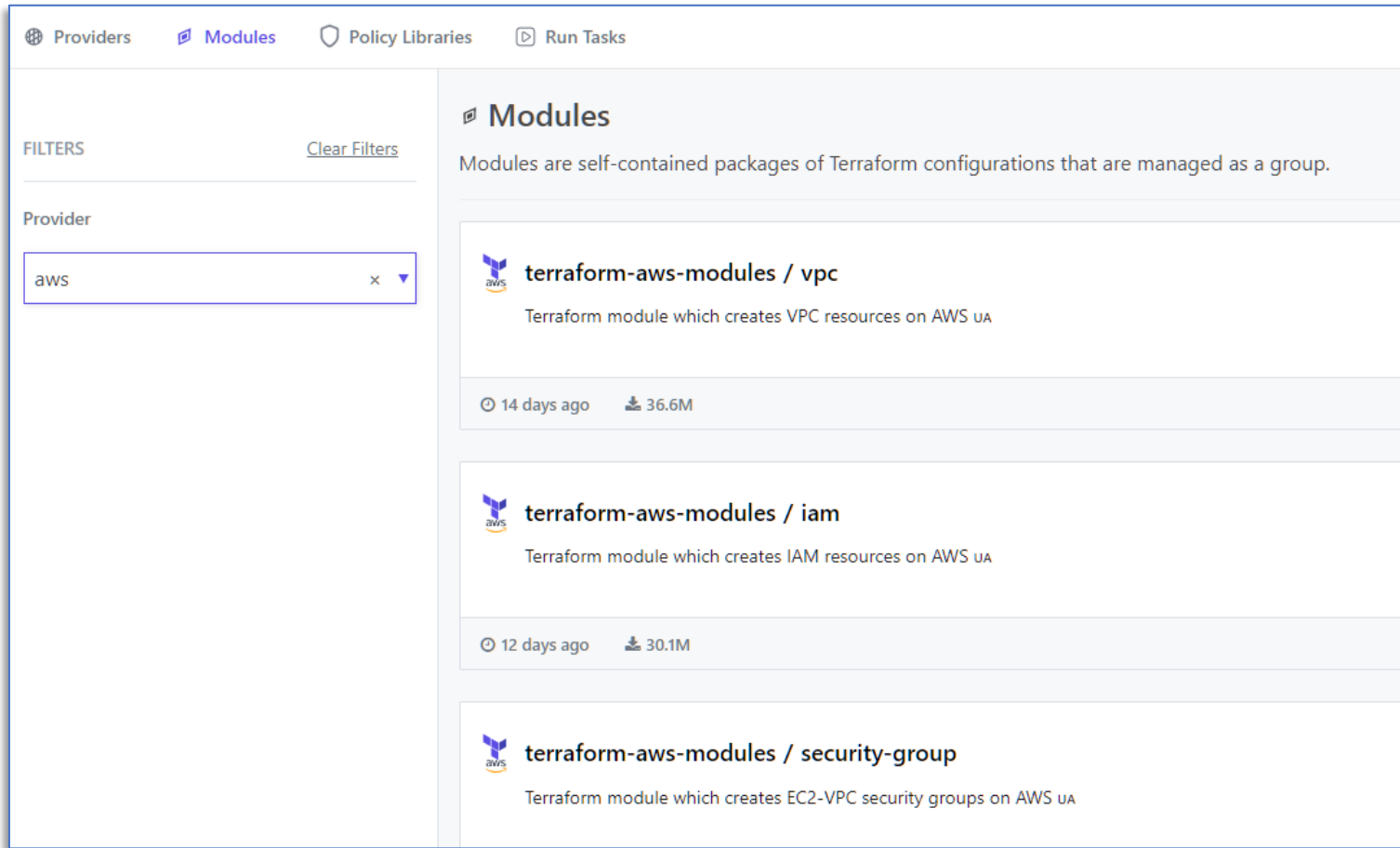
Do you require separation of concerns?

Can you re-use your code segments in other projects?

- Pros/Cons of one large configuration (single Root Module) vs. Parent / Child modules to define your configuration ?



Don't re-invent the wheel



<https://registry.terraform.io/browse/modules?provider=aws>



Root & Child Modules

Child Modules use the same configuration language as the Root

Input variables

output values

Resources

Location

local – create a new directory and create one or more .tf files

remote - (public vs private registry)

Referencing the child modules

local – by relative path

remote – by path to registry / module

```
module "servers" {  
  source = "./app-cluster"  
  servers = 5  
}
```

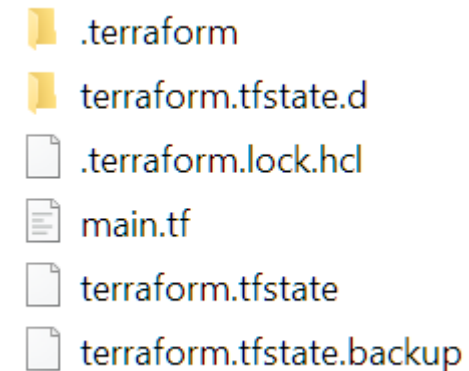


HashiCorp
Terraform

Terraform CLI Workspaces

- Terraform CLI workspaces differ from Terraform Cloud Workspaces
- Terraform CLI workspaces – one by default called ‘default’
- The ‘default’ cannot be deleted
- Each workspace holds our ‘state’
- To create a new workspace called Test: `terraform workspace new test`
- To switch to a workspace called Test: `terraform workspace select test`

```
resource "aws_instance" "WebEC2" {  
  count = "${terraform.workspace == "test" ? 2 : 10}"  
  
  # ... more arguments as required  
}
```



- .terraform
- terraform.tfstate.d
- .terraform.lock.hcl
- main.tf
- terraform.tfstate
- terraform.tfstate.backup



Local vs Remote State files

- State may be recorded locally or remote

Local by default

May be migrated

May be 'moved' without data migration

- Locations

Local: may use default location or you may specify path

Remote: many options available

```
terraform {  
  backend "local" {  
    path = "relative/path/to/terraform.tfstate"  
  }  
}
```

```
terraform {  
  backend "s3" {  
    bucket = "mybucket"  
    key    = "path/to/my/key"  
    region = "us-east-1"  
  }  
}
```



Backends

- Not all backends provide the same functionality, some research may be required
- Even a particular backend may provide different features depending on its setup. S3 may simply store the state or use DynamoDB to perform state locking and consistency checking.

<https://developer.hashicorp.com/terraform/language/settings/backends/configuration>

local
remote
artifactory
azurerm
consul
cos
etcd
etcdv3
gcs
http
Kubernetes
S3



Managing State and Configuration Drift

- Resources should be managed by our IaC tooling
- Human intervention may accidentally introduce changes to configuration
- Deliberate intervention may be required for Priority fixes
- Configuration as Code tools may intentionally introduce changes due to bad planning
- Configuration Drift needs to be detected and fixed



<https://pixabay.com/photos/drifting-sport-drift-car-speed-2396992/>



Taint/Replace an existing resource

- Taint/Replace is used to inform Terraform that a resource has become damaged or degraded and should be replaced at next `apply`

Taint < v0.15.2

```
terraform taint [options] aws_instance.vm1
```

Replace => v0.15.2

```
terraform apply -replace="aws_instance.vm1"
```



Logging

- Enable logging using TF_LOG environment variable. This will direct any logging output to *stderr*
- You can set TF_LOG to TRACE, DEBUG, INFO, WARN or ERROR to change the verbosity of the logs. If you set TF_LOG to JSON, the output logs at the TRACE level in a json format.
- If you wish to persist the output to your own log, then set TF_LOG_PATH in order to append to a specific file when logging is enabled with TF_LOG
- Logging can be enabled separately for terraform itself and the provider plugins using the TF_LOG_CORE or TF_LOG_PROVIDER environment variables. These take the same level arguments as TF_LOG, but only activate a subset of the logs.

<https://developer.hashicorp.com/terraform/internals/debugging>



HashiCorp
Terraform

Lab 5

Create an Application Load Balancer with Autoscaling on AWS using Terraform

In this lab you will deploy an application load balancer to distribute web traffic across an auto-scaling group of EC2 instances. You will be provided with a networking module.

