Building Scalable Infrastructure using GitOps with Terraform on GCP

HashiTalks Africa 2023

Basil Ndonga

Senior Software Engineer

Yunus Arif Said

Lead Software Engineer

in https://www.linkedin.com/in/basil-ndonga/

in https://www.linkedin.com/in/yunus-arif-said-57879351/

What we do

iProcure is an agri-tech company that offers last mile distribution services, currently supplying agri-inputs to over 1 Million commercial smallholder farmers in East Africa

Practically speaking, it means we design, implement and operate scalable microservices to support our customers.

Agenda

- The Manual Way
- Infrastructure as Code
- Lessons Learned
- The GitOps Way
- Demo

Imagine you want to deploy your newly created backend on the cloud.

Manual Way

- Go to cloud console e.g GCP or AWS
- Choose appropriate server, machine type, OS
- Configure Networking - VPCs, Subnets, Firewalls
- Spin up a database instance
- Set up Secrets, TLS certificates
- Spend hours on stack overflow
- Configure Application

Now imagine you want to do the same thing again. You have to replicate the same steps

Problems with Manual Way

- Ops teams occasionally make mistakes
- Releases become slow, repetitive and painful
- Outages become more frequent

Devs say “But it works on my machine”

What if we told you there was an better way?

Infrastructure as Code

The idea behind IaC is that, you can manage almost everything as code.

- Servers
- Databases
- Load balancers
- Networks
- Documentation

Tools

- Terraform
- Google Deployment Manager
- Azure Resource Manager
- AWS Cloud Formation
- Pulumi

Benefits

- Code reuse
- Documentation
- Automation
- Version Control
- Collaboration

Terraform 101

A tool for deploying your IaC using a simple declarative language. Works with a bunch of different providers.

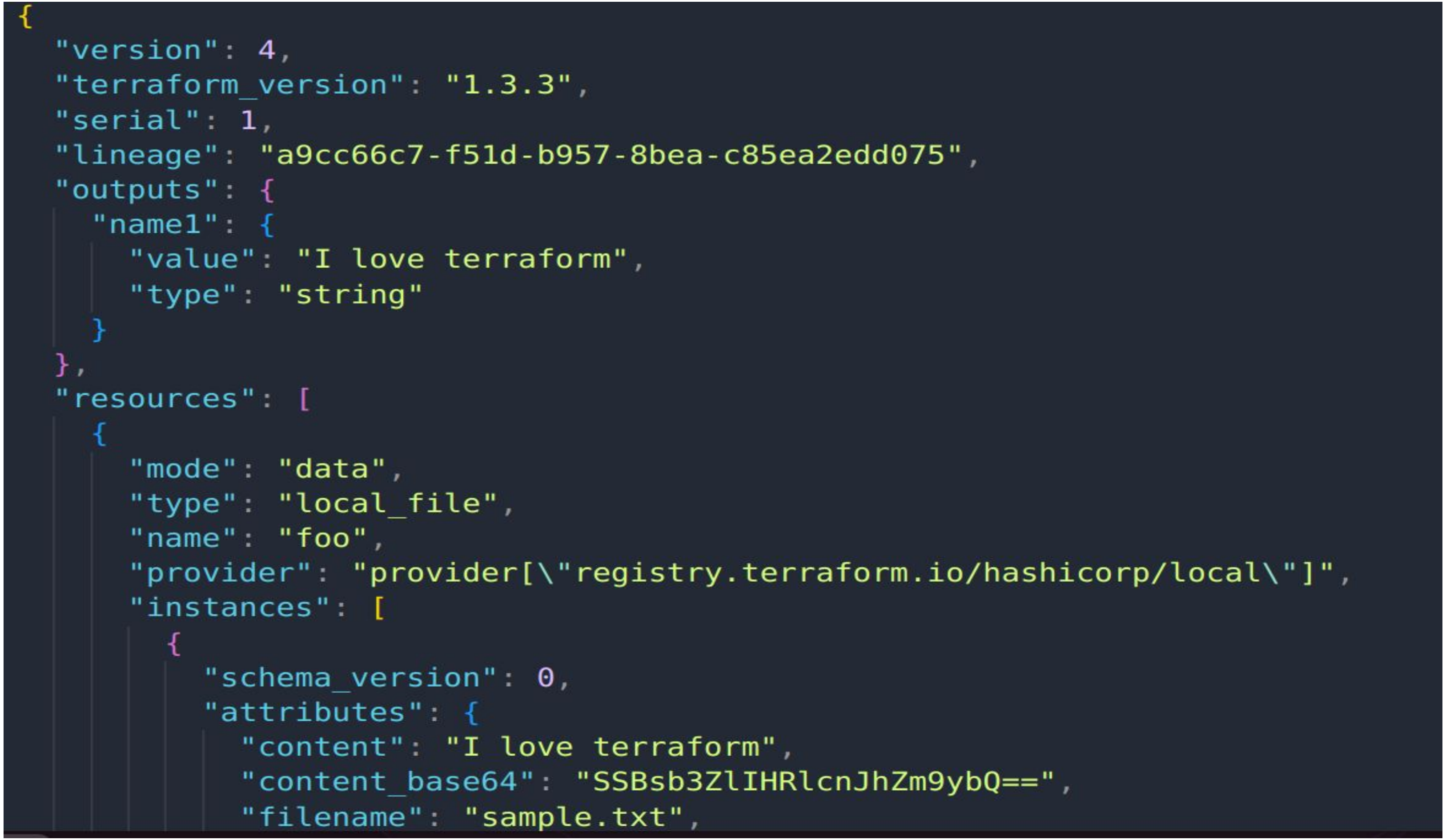
Lessons Learnt Implementing IaC

- Implement remote state
- Isolate multiple environments
- Create reusable modules
- Test your infrastructure code

Implement Remote State

Terraform can find resources it previously created, as it keeps track of what resources it created on a state file

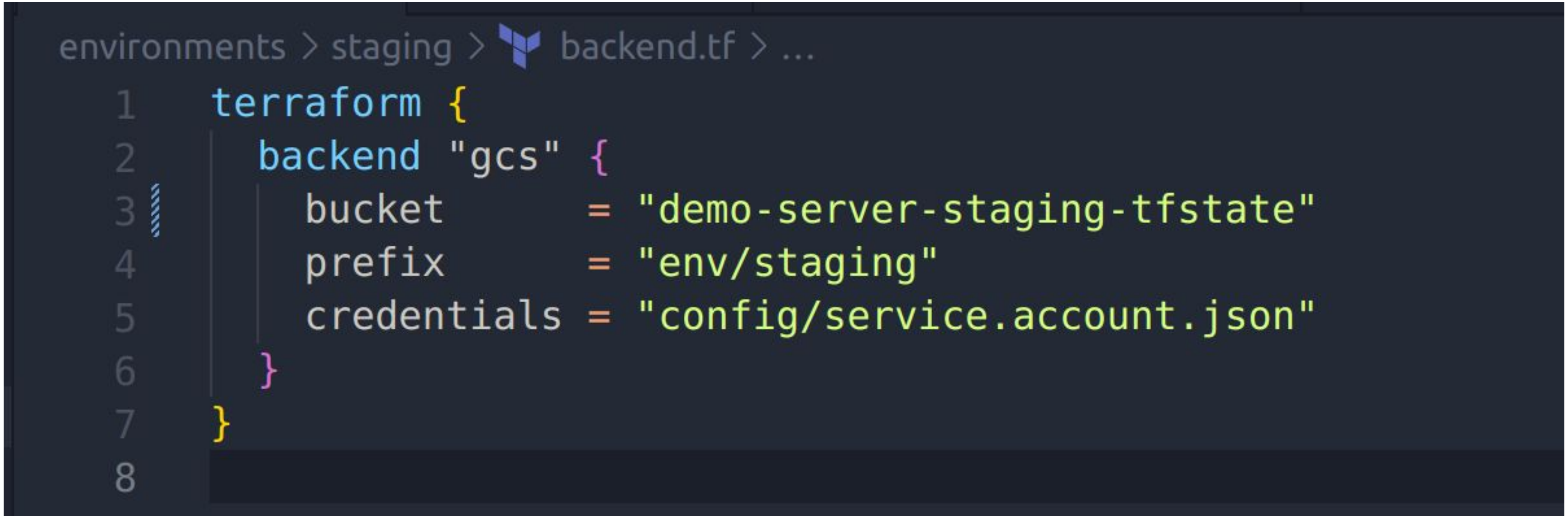
By default terraform uses a local backend. The state file (.tfstate) is stored in json format.



```
{
  "version": 4,
  "terraform_version": "1.3.3",
  "serial": 1,
  "lineage": "a9cc66c7-f51d-b957-8bea-c85ea2edd075",
  "outputs": {
    "name1": {
      "value": "I love terraform",
      "type": "string"
    }
  },
  "resources": [
    {
      "mode": "data",
      "type": "local_file",
      "name": "foo",
      "provider": "provider[\"registry.terraform.io/hashicorp/local\"]",
      "instances": [
        {
          "schema_version": 0,
          "attributes": {
            "content": "I love terraform",
            "content_base64": "SSBsb3ZlIHRlcnJhZm9ybQ==",
            "filename": "sample.txt",
```

If you are working with teams, each member needs access to same terraform state.

State needs to be stored in a remote location (remote backend). State locking prevents concurrent modification.



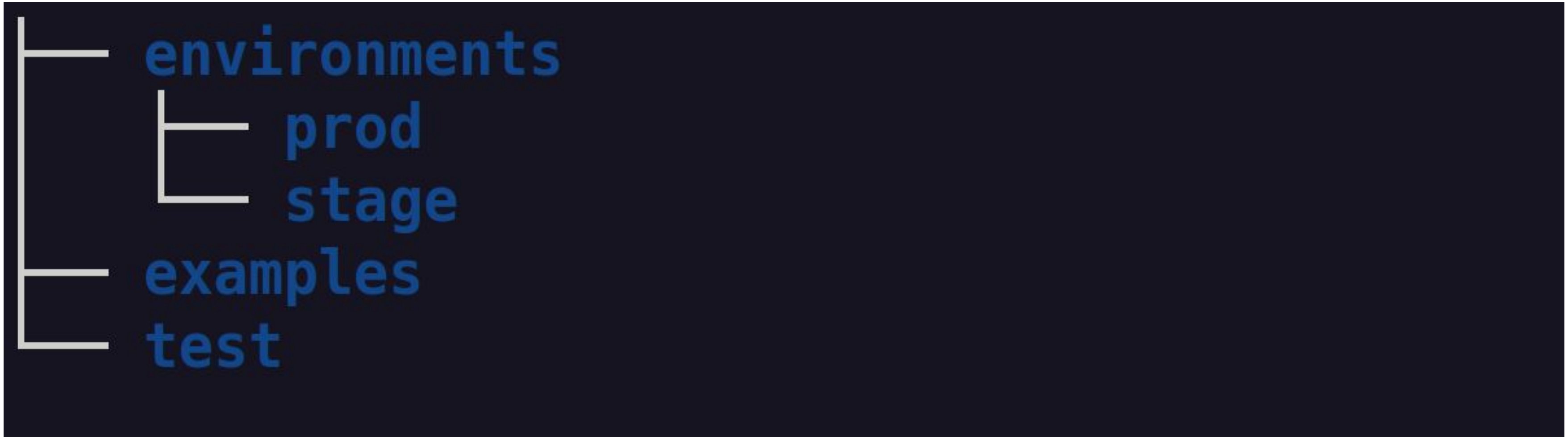
environments > staging > backend.tf > ...

```
terraform {
  backend "gcs" {
    bucket      = "demo-server-staging-tfstate"
    prefix      = "env/staging"
    credentials = "config/service.account.json"
  }
}
```

With remote backend enabled, terraform will now pull latest changes when running plan & apply

Multiple Environments

It's a good practice to have separate environments e.g stage and prod

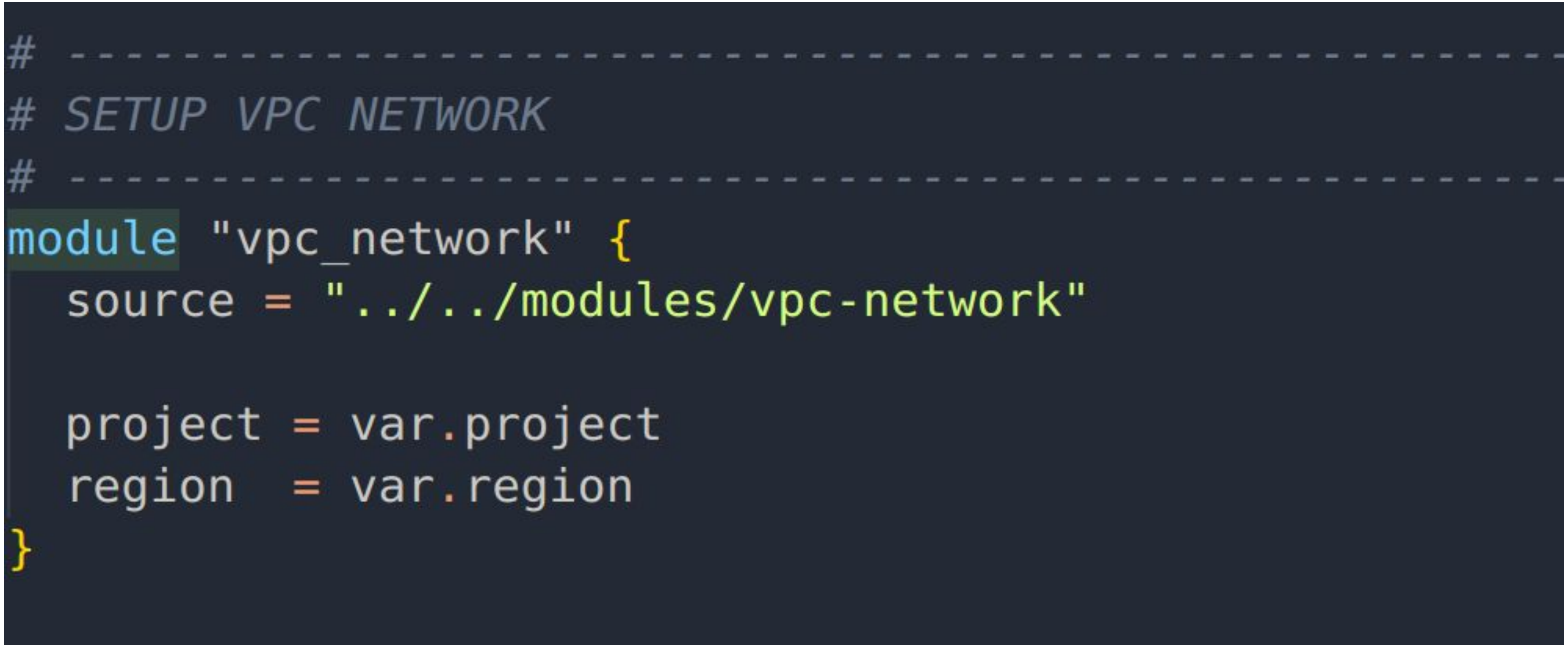


```
├── environments
│   ├── prod
│   └── stage
├── examples
└── test
```

Reusable Modules

Combines resources generally used together, essentially a blueprint.

You can now reuse the same module across multiple environments.



```
# -----------------------------------------------------
# SETUP VPC NETWORK
# -----------------------------------------------------
module "vpc_network" {
  source = "../../modules/vpc-network"

  project = var.project
  region  = var.region
}
```

Module Best Practices

- Place your modules in a git repository
- Semantically version modules using tags
- Validate module changes via code reviews
- Leverage terraform module registry

How do you unit test infrastructure code?

Writing tests gives you confidence that your infrastructure works as expected.

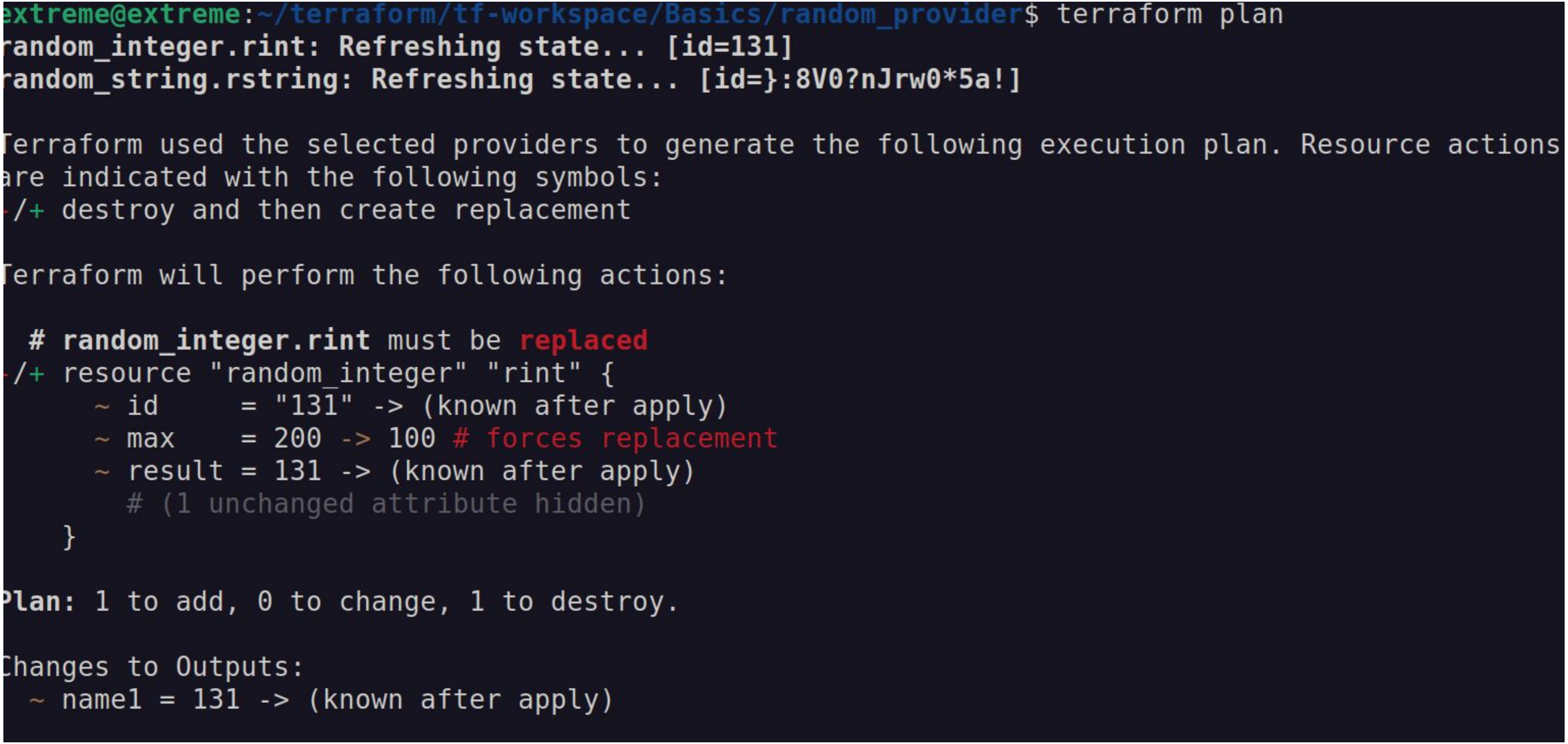
Test Strategies

- Static Analysis
- Unit Tests

Static Analysis

Test your code without deploying it. Most basic level of testing.

You can use *terraform plan* to review changes to your infrastructure.



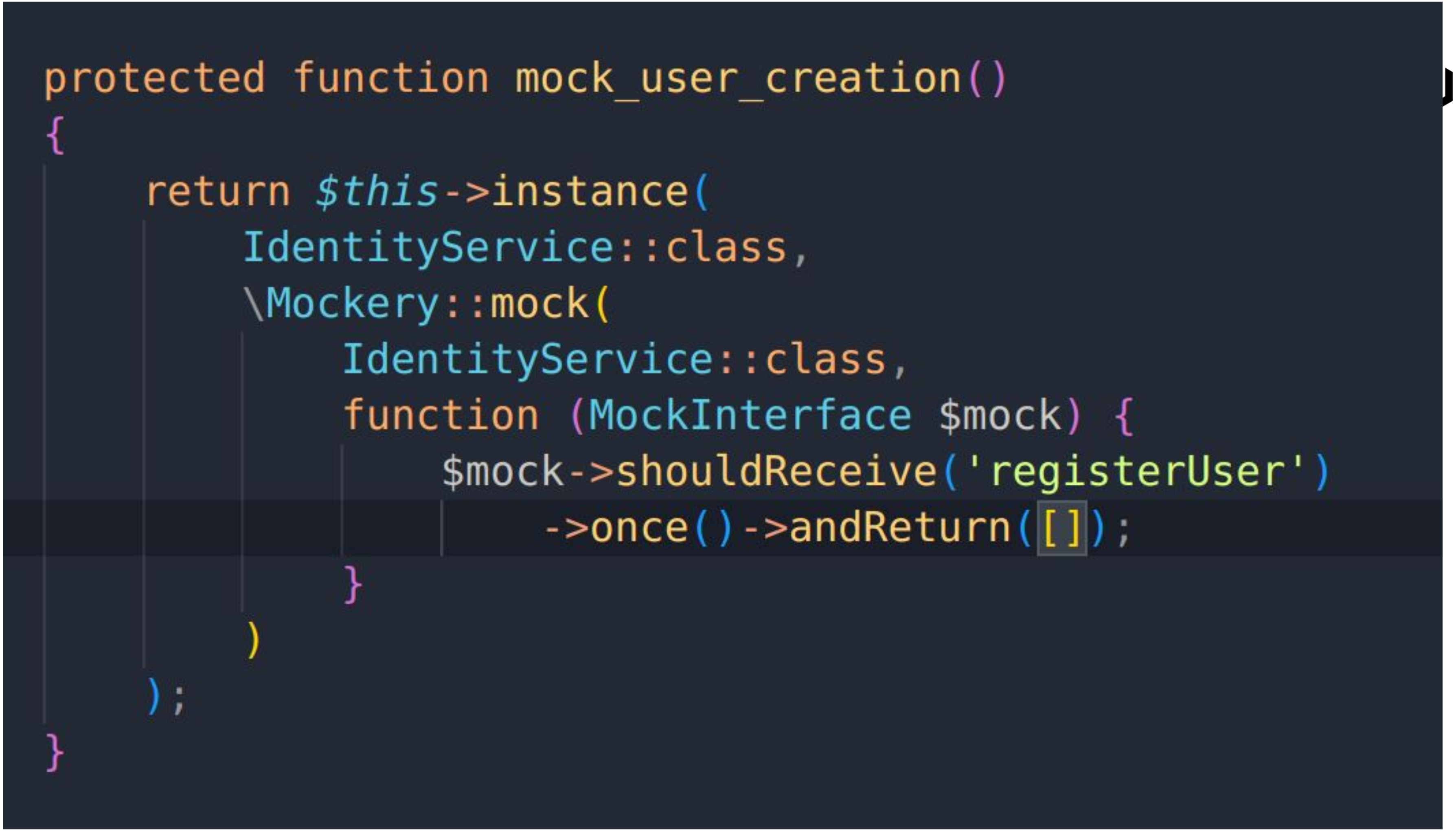
```
extreme@extreme:~/terraform/tf-workspace/Basics/random_provider$ terraform plan
random_integer.rint: Refreshing state... [id=131]
random_string.rstring: Refreshing state... [id=}:8V0?nJrw0*5a!]

Terraform used the selected providers to generate the following execution plan. Resource actions
are indicated with the following symbols:
-/+ destroy and then create replacement

Terraform will perform the following actions:

  # random_integer.rint must be replaced
-/+ resource "random_integer" "rint" {
      ~ id     = "131" -> (known after apply)
      ~ max    = 200 -> 100 # forces replacement
      ~ result = 131 -> (known after apply)
        # (1 unchanged attribute hidden)
    }

Plan: 1 to add, 0 to change, 1 to destroy.

Changes to Outputs:
  ~ name1 = 131 -> (known after apply)
```

You can also check for syntax errors with *terraform validate.* Helps catch a ton of common mistakes

Unit Testing

Implies testing individual modules.

In programming you can isolate tests from the outside environment and even mock external world.



```php
protected function mock_user_creation()
{
    return $this->instance(
        IdentityService::class,
        \Mockery::mock(
            IdentityService::class,
            function (MockInterface $mock) {
                $mock->shouldReceive('registerUser')
                    ->once()->andReturn([]);
            }
        )
    );
}
```

Infrastructure code communicates to cloud providers, essentially interacting with outside world.

This means deploying it to a real environment on GCP, AWS, et cetera. Best practice is to use a different project or sandbox environment.

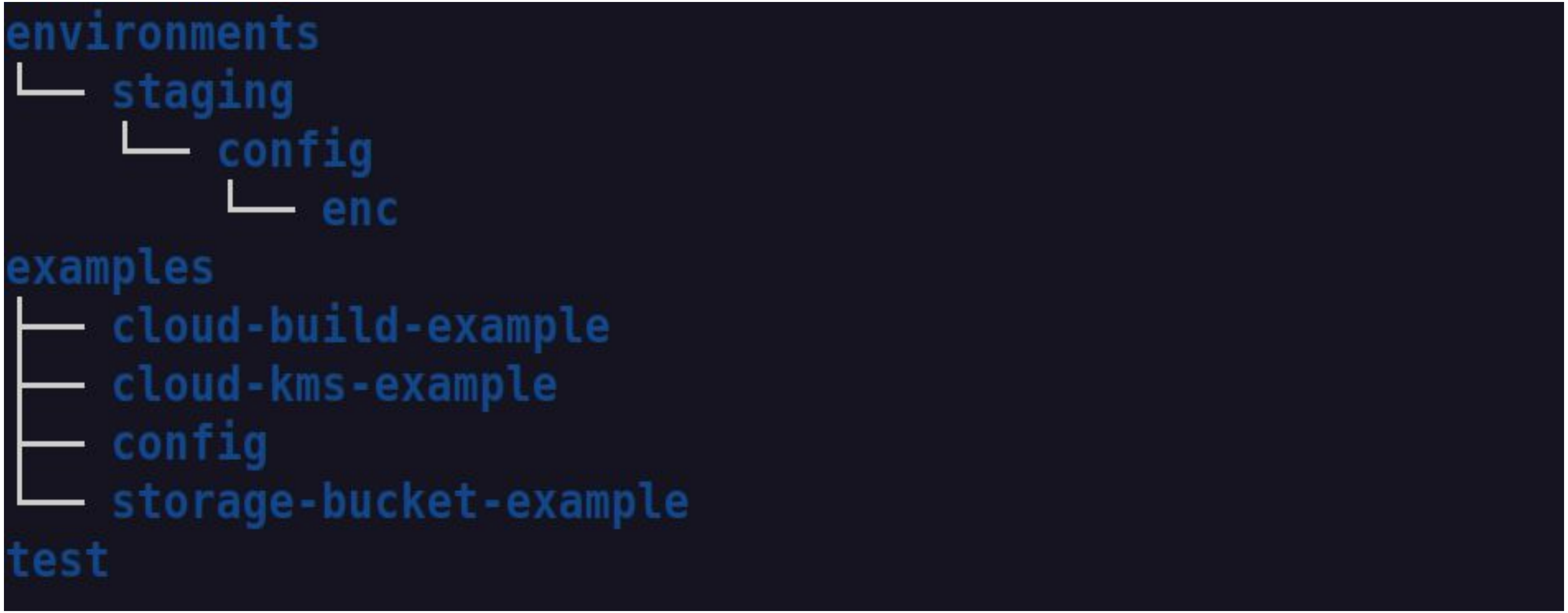
Testing modules

- Provision the test module
- Validate that the tests pass
- Destroy the infra once the test completes

Tools

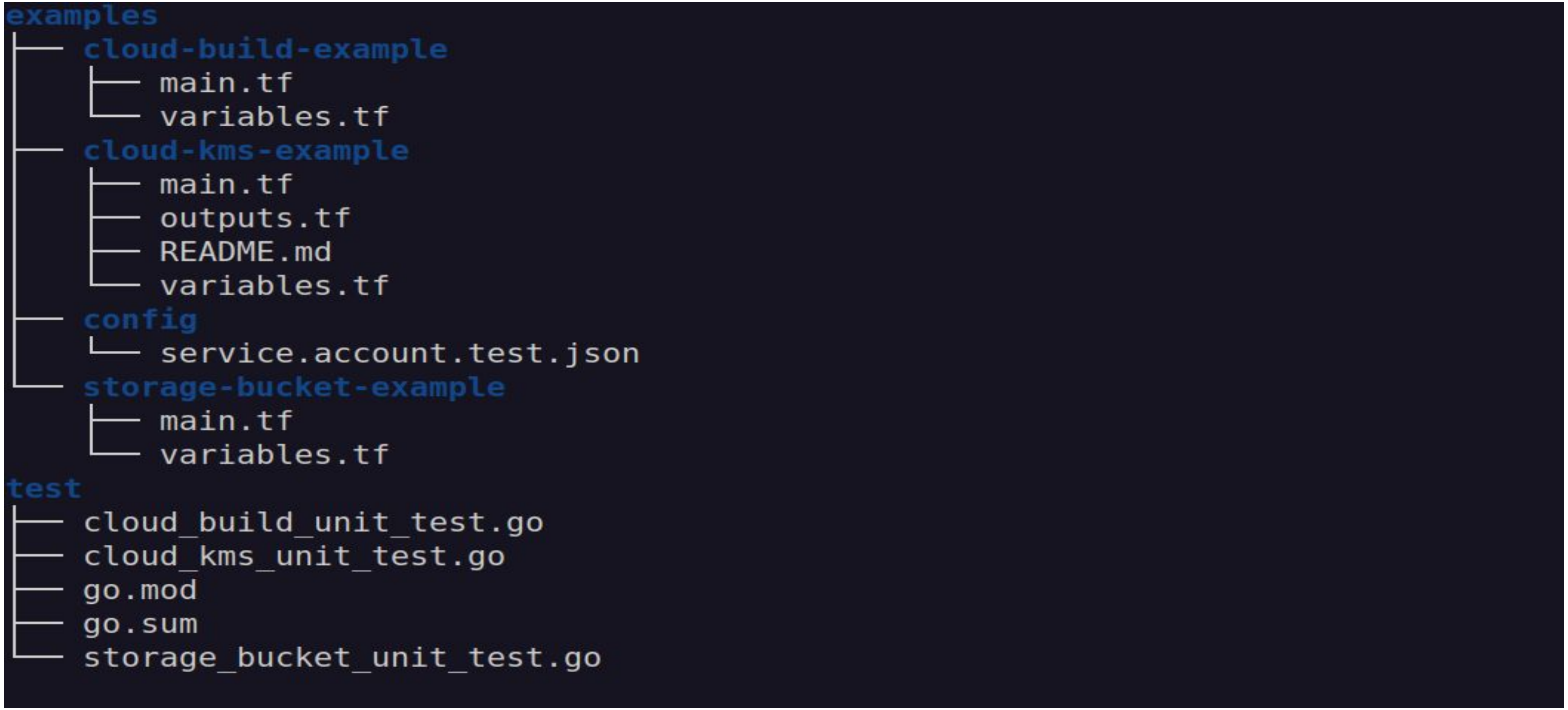
- Terratest - library written in Go
- Kitchen terraform

Directory Structure



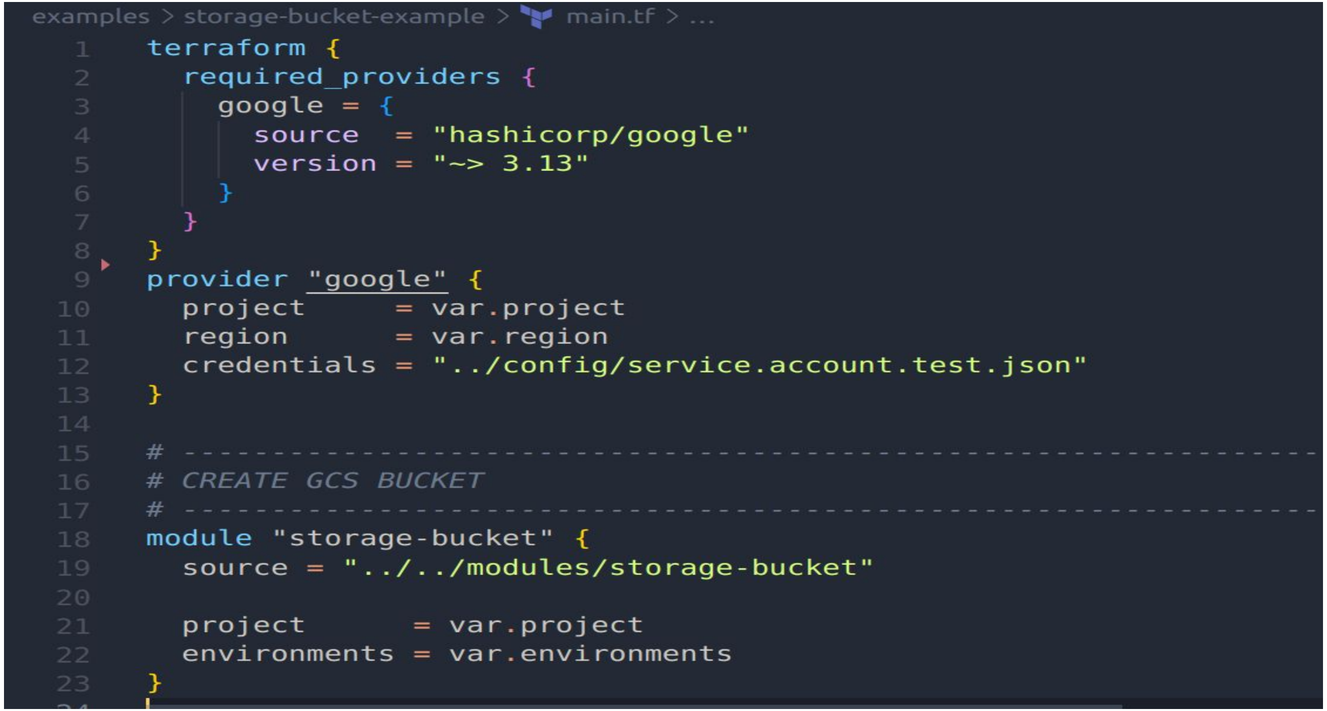
```
environments
└── staging
    └── config
        └── enc
examples
├── cloud-build-example
├── cloud-kms-example
├── config
└── storage-bucket-example
test
```

Test Directories



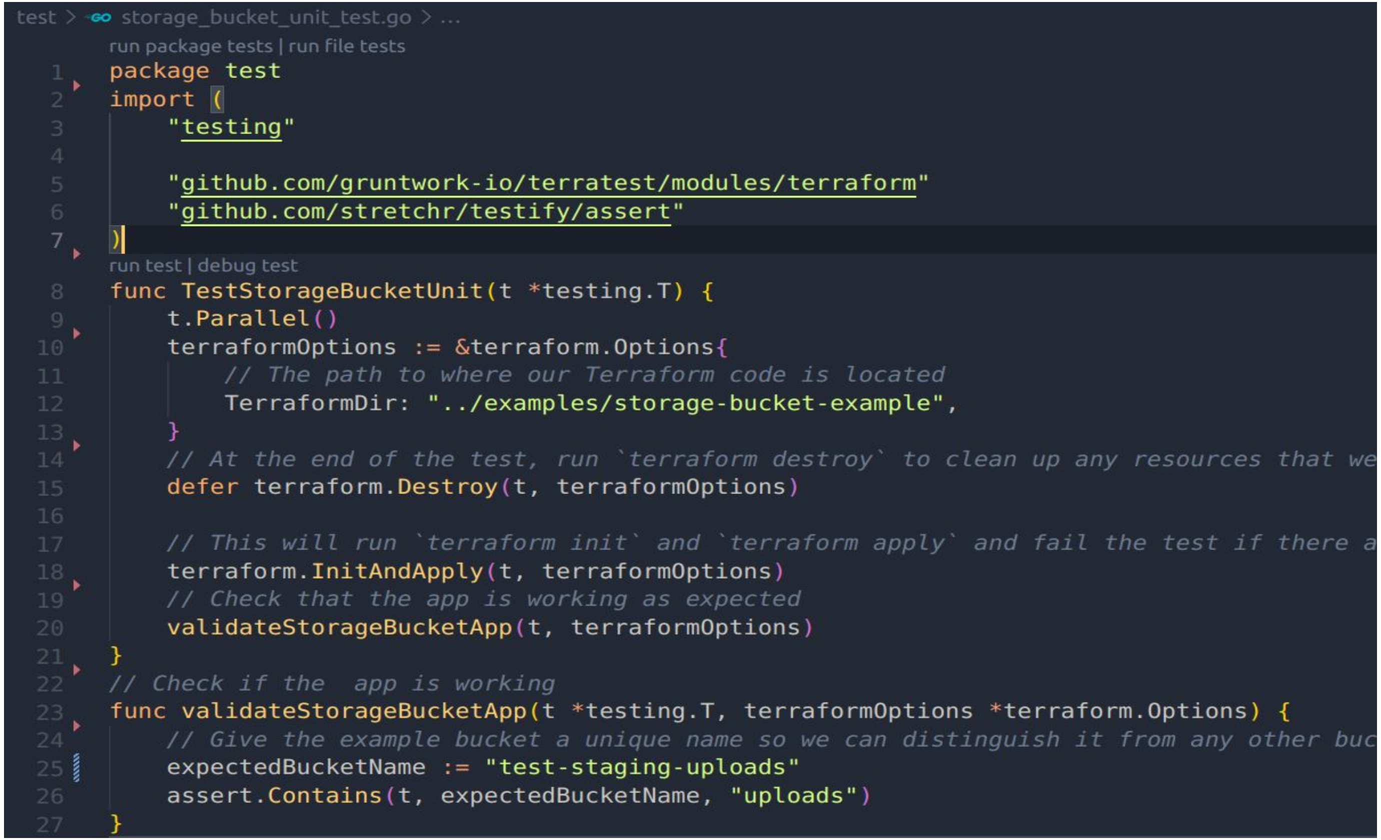
```
examples
├── cloud-build-example
│   ├── main.tf
│   └── variables.tf
├── cloud-kms-example
│   ├── main.tf
│   ├── outputs.tf
│   ├── README.md
│   └── variables.tf
├── config
│   └── service.account.test.json
└── storage-bucket-example
    ├── main.tf
    └── variables.tf
test
├── cloud_build_unit_test.go
├── cloud_kms_unit_test.go
├── go.mod
├── go.sum
└── storage_bucket_unit_test.go
```

Storage Bucket Example



```
terraform {
  required_providers {
    google = {
      source  = "hashicorp/google"
      version = "~> 3.13"
    }
  }
}
provider "google" {
  project     = var.project
  region      = var.region
  credentials = "../config/service.account.test.json"
}

# ---------------------------------------------------------------
# CREATE GCS BUCKET
# ---------------------------------------------------------------
module "storage-bucket" {
  source = "../../modules/storage-bucket"

  project      = var.project
  environments = var.environments
}
```

Storage bucket test.go

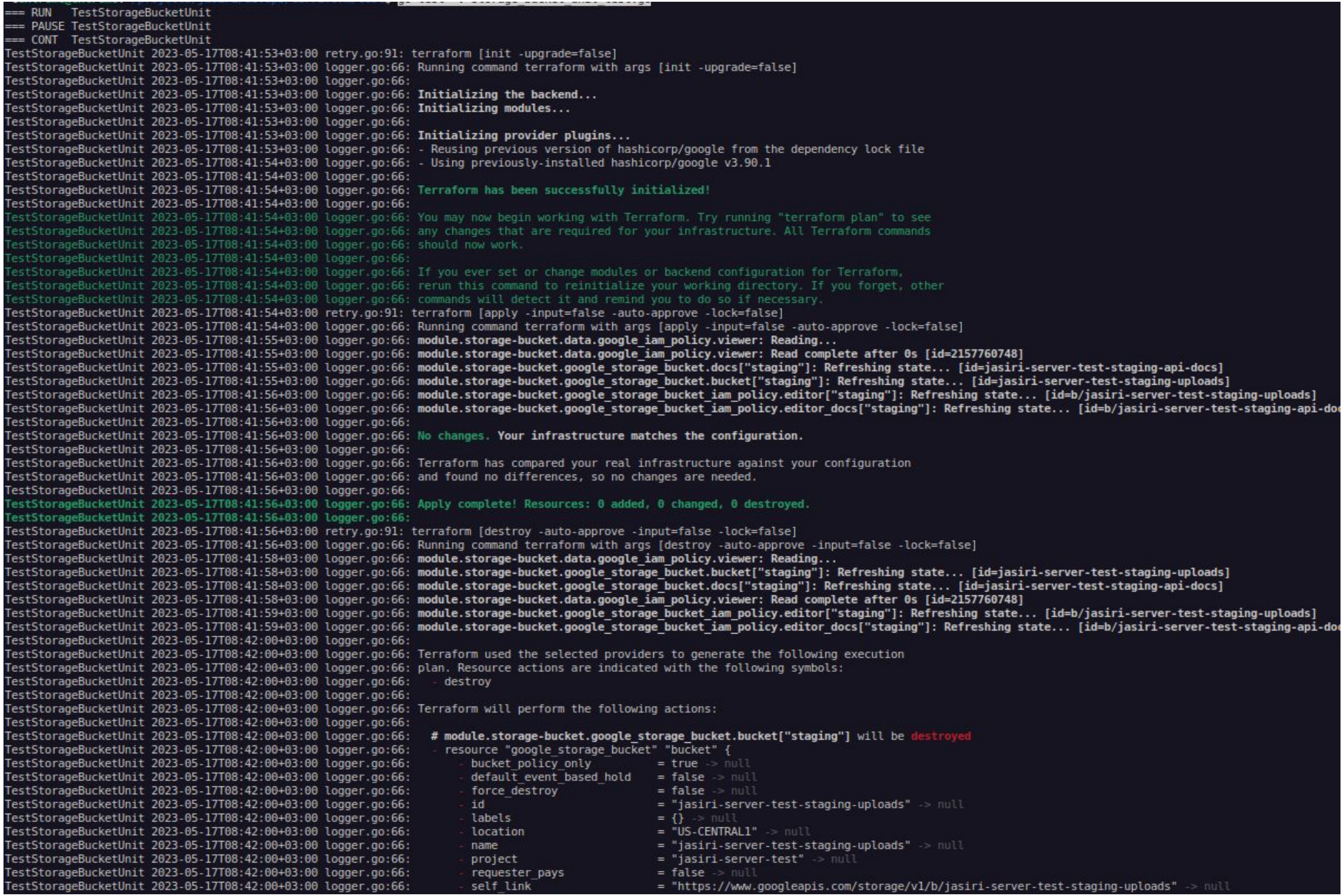

test > storage_bucket_unit_test.go > ...
run package tests | run file tests
run test | debug test

```go
package test
import (
    "testing"

    "github.com/gruntwork-io/terratest/modules/terraform"
    "github.com/stretchr/testify/assert"
)
func TestStorageBucketUnit(t *testing.T) {
    t.Parallel()
    terraformOptions := &terraform.Options{
        // The path to where our Terraform code is located
        TerraformDir: "../examples/storage-bucket-example",
    }
    // At the end of the test, run `terraform destroy` to clean up any resources that we
    defer terraform.Destroy(t, terraformOptions)

    // This will run `terraform init` and `terraform apply` and fail the test if there a
    terraform.InitAndApply(t, terraformOptions)
    // Check that the app is working as expected
    validateStorageBucketApp(t, terraformOptions)
}
// Check if the  app is working
func validateStorageBucketApp(t *testing.T, terraformOptions *terraform.Options) {
    // Give the example bucket a unique name so we can distinguish it from any other buc
    expectedBucketName := "test-staging-uploads"
    assert.Contains(t, expectedBucketName, "uploads")
}
```

Run Test

```
$ cd test
```

```
 $ go test -v storage_bucket_unit_test.go
```



```
=== RUN   TestStorageBucketUnit
=== PAUSE TestStorageBucketUnit
=== CONT  TestStorageBucketUnit
TestStorageBucketUnit 2023-05-17T08:41:53+03:00 retry.go:91: terraform [init -upgrade=false]
TestStorageBucketUnit 2023-05-17T08:41:53+03:00 logger.go:66: Running command terraform with args [init -upgrade=false]
TestStorageBucketUnit 2023-05-17T08:41:53+03:00 logger.go:66:
TestStorageBucketUnit 2023-05-17T08:41:53+03:00 logger.go:66: Initializing the backend...
TestStorageBucketUnit 2023-05-17T08:41:53+03:00 logger.go:66: Initializing modules...
TestStorageBucketUnit 2023-05-17T08:41:53+03:00 logger.go:66:
TestStorageBucketUnit 2023-05-17T08:41:53+03:00 logger.go:66: Initializing provider plugins...
TestStorageBucketUnit 2023-05-17T08:41:53+03:00 logger.go:66: - Reusing previous version of hashicorp/google from the dependency lock file
TestStorageBucketUnit 2023-05-17T08:41:54+03:00 logger.go:66: - Using previously-installed hashicorp/google v3.90.1
TestStorageBucketUnit 2023-05-17T08:41:54+03:00 logger.go:66:
TestStorageBucketUnit 2023-05-17T08:41:54+03:00 logger.go:66: Terraform has been successfully initialized!
TestStorageBucketUnit 2023-05-17T08:41:54+03:00 logger.go:66:
TestStorageBucketUnit 2023-05-17T08:41:54+03:00 logger.go:66: You may now begin working with Terraform. Try running "terraform plan" to see
TestStorageBucketUnit 2023-05-17T08:41:54+03:00 logger.go:66: any changes that are required for your infrastructure. All Terraform commands
TestStorageBucketUnit 2023-05-17T08:41:54+03:00 logger.go:66: should now work.
TestStorageBucketUnit 2023-05-17T08:41:54+03:00 logger.go:66:
TestStorageBucketUnit 2023-05-17T08:41:54+03:00 logger.go:66: If you ever set or change modules or backend configuration for Terraform,
TestStorageBucketUnit 2023-05-17T08:41:54+03:00 logger.go:66: rerun this command to reinitialize your working directory. If you forget, other
TestStorageBucketUnit 2023-05-17T08:41:54+03:00 logger.go:66: commands will detect it and remind you to do so if necessary.
TestStorageBucketUnit 2023-05-17T08:41:54+03:00 retry.go:91: terraform [apply -input=false -auto-approve -lock=false]
TestStorageBucketUnit 2023-05-17T08:41:54+03:00 logger.go:66: Running command terraform with args [apply -input=false -auto-approve -lock=false]
TestStorageBucketUnit 2023-05-17T08:41:55+03:00 logger.go:66: module.storage-bucket.data.google_iam_policy.viewer: Reading...
TestStorageBucketUnit 2023-05-17T08:41:55+03:00 logger.go:66: module.storage-bucket.data.google_iam_policy.viewer: Read complete after 0s [id=2157760748]
TestStorageBucketUnit 2023-05-17T08:41:55+03:00 logger.go:66: module.storage-bucket.google_storage_bucket.docs["staging"]: Refreshing state... [id=jasiri-server-test-staging-api-docs]
TestStorageBucketUnit 2023-05-17T08:41:55+03:00 logger.go:66: module.storage-bucket.google_storage_bucket.bucket["staging"]: Refreshing state... [id=jasiri-server-test-staging-uploads]
TestStorageBucketUnit 2023-05-17T08:41:56+03:00 logger.go:66: module.storage-bucket.google_storage_bucket_iam_policy.editor["staging"]: Refreshing state... [id=b/jasiri-server-test-staging-uploads]
TestStorageBucketUnit 2023-05-17T08:41:56+03:00 logger.go:66: module.storage-bucket.google_storage_bucket_iam_policy.editor_docs["staging"]: Refreshing state... [id=b/jasiri-server-test-staging-api-doc
TestStorageBucketUnit 2023-05-17T08:41:56+03:00 logger.go:66:
TestStorageBucketUnit 2023-05-17T08:41:56+03:00 logger.go:66: No changes. Your infrastructure matches the configuration.
TestStorageBucketUnit 2023-05-17T08:41:56+03:00 logger.go:66:
TestStorageBucketUnit 2023-05-17T08:41:56+03:00 logger.go:66: Terraform has compared your real infrastructure against your configuration
TestStorageBucketUnit 2023-05-17T08:41:56+03:00 logger.go:66: and found no differences, so no changes are needed.
TestStorageBucketUnit 2023-05-17T08:41:56+03:00 logger.go:66:
TestStorageBucketUnit 2023-05-17T08:41:56+03:00 logger.go:66: Apply complete! Resources: 0 added, 0 changed, 0 destroyed.
TestStorageBucketUnit 2023-05-17T08:41:56+03:00 logger.go:66:
TestStorageBucketUnit 2023-05-17T08:41:56+03:00 retry.go:91: terraform [destroy -auto-approve -input=false -lock=false]
TestStorageBucketUnit 2023-05-17T08:41:56+03:00 logger.go:66: Running command terraform with args [destroy -auto-approve -input=false -lock=false]
TestStorageBucketUnit 2023-05-17T08:41:58+03:00 logger.go:66: module.storage-bucket.data.google_iam_policy.viewer: Reading...
TestStorageBucketUnit 2023-05-17T08:41:58+03:00 logger.go:66: module.storage-bucket.google_storage_bucket.bucket["staging"]: Refreshing state... [id=jasiri-server-test-staging-uploads]
TestStorageBucketUnit 2023-05-17T08:41:58+03:00 logger.go:66: module.storage-bucket.google_storage_bucket.docs["staging"]: Refreshing state... [id=jasiri-server-test-staging-api-docs]
TestStorageBucketUnit 2023-05-17T08:41:58+03:00 logger.go:66: module.storage-bucket.data.google_iam_policy.viewer: Read complete after 0s [id=2157760748]
TestStorageBucketUnit 2023-05-17T08:41:59+03:00 logger.go:66: module.storage-bucket.google_storage_bucket_iam_policy.editor["staging"]: Refreshing state... [id=b/jasiri-server-test-staging-uploads]
TestStorageBucketUnit 2023-05-17T08:41:59+03:00 logger.go:66: module.storage-bucket.google_storage_bucket_iam_policy.editor_docs["staging"]: Refreshing state... [id=b/jasiri-server-test-staging-api-doc
TestStorageBucketUnit 2023-05-17T08:42:00+03:00 logger.go:66:
TestStorageBucketUnit 2023-05-17T08:42:00+03:00 logger.go:66: Terraform used the selected providers to generate the following execution
TestStorageBucketUnit 2023-05-17T08:42:00+03:00 logger.go:66: plan. Resource actions are indicated with the following symbols:
TestStorageBucketUnit 2023-05-17T08:42:00+03:00 logger.go:66:   - destroy
TestStorageBucketUnit 2023-05-17T08:42:00+03:00 logger.go:66:
TestStorageBucketUnit 2023-05-17T08:42:00+03:00 logger.go:66: Terraform will perform the following actions:
TestStorageBucketUnit 2023-05-17T08:42:00+03:00 logger.go:66:
TestStorageBucketUnit 2023-05-17T08:42:00+03:00 logger.go:66:   # module.storage-bucket.google_storage_bucket.bucket["staging"] will be destroyed
TestStorageBucketUnit 2023-05-17T08:42:00+03:00 logger.go:66:   - resource "google_storage_bucket" "bucket" {
TestStorageBucketUnit 2023-05-17T08:42:00+03:00 logger.go:66:       - bucket_policy_only          = true -> null
TestStorageBucketUnit 2023-05-17T08:42:00+03:00 logger.go:66:       - default_event_based_hold    = false -> null
TestStorageBucketUnit 2023-05-17T08:42:00+03:00 logger.go:66:       - force_destroy               = false -> null
TestStorageBucketUnit 2023-05-17T08:42:00+03:00 logger.go:66:       - id                          = "jasiri-server-test-staging-uploads" -> null
TestStorageBucketUnit 2023-05-17T08:42:00+03:00 logger.go:66:       - labels                      = {} -> null
TestStorageBucketUnit 2023-05-17T08:42:00+03:00 logger.go:66:       - location                    = "US-CENTRAL1" -> null
TestStorageBucketUnit 2023-05-17T08:42:00+03:00 logger.go:66:       - name                        = "jasiri-server-test-staging-uploads" -> null
TestStorageBucketUnit 2023-05-17T08:42:00+03:00 logger.go:66:       - project                     = "jasiri-server-test" -> null
TestStorageBucketUnit 2023-05-17T08:42:00+03:00 logger.go:66:       - requester_pays              = false -> null
TestStorageBucketUnit 2023-05-17T08:42:00+03:00 logger.go:66:       - self_link                   = "https://www.googleapis.com/storage/v1/b/jasiri-server-test-staging-uploads" -> null
```

GitOps

Gitops takes devops practices and applies them to infrastructure automation.

Infrastructure code can now be stored in Git Repositories, reviewed, merged and pushed to target environments

GitOps Flow

- Devs push code to git repositories
- Create pull request to merge to master
- Triggers an automated CICD pipeline
- Changes are automatically built, tested & pushed to the target infrastructure

GitOps Benefits

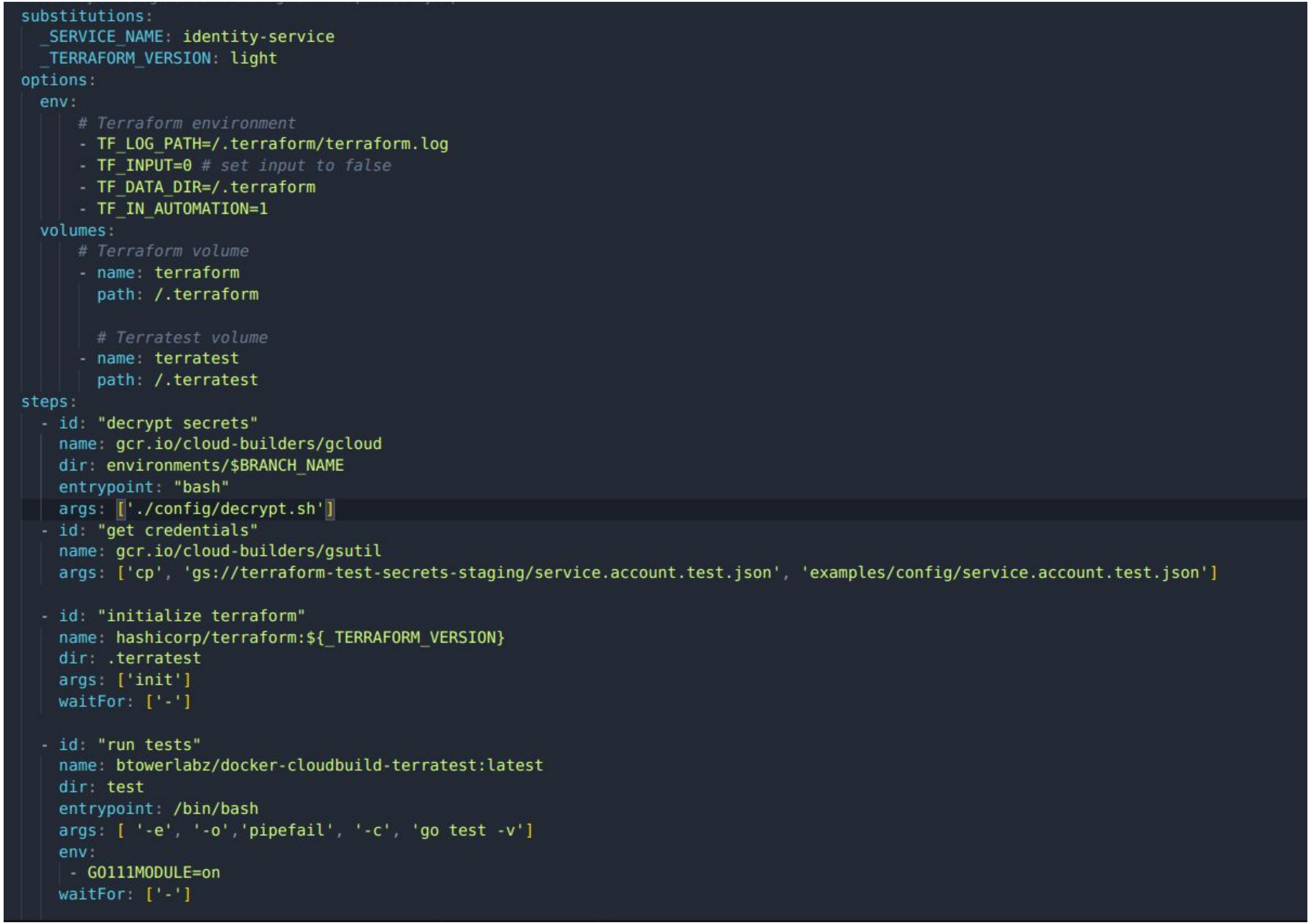
- Easily rollback infrastructure
- Track changes to your infrastructure
- Faster and more frequent deployments
- Code review - only authorized members make changes to infrastructure

Push Based Strategy

- Devs make changes and push to git
- Pipeline reacts and triggers a series of automated actions.
- Actions updates K8s manifests and triggers changes to infrastructure.

Build Tools

- Google Cloud Build
- Github Actions
- Gitlab CICD
- Circle CI



```yaml
substitutions:
  _SERVICE_NAME: identity-service
  _TERRAFORM_VERSION: light
options:
  env:
      # Terraform environment
      - TF_LOG_PATH=/.terraform/terraform.log
      - TF_INPUT=0 # set input to false
      - TF_DATA_DIR=/.terraform
      - TF_IN_AUTOMATION=1
  volumes:
      # Terraform volume
      - name: terraform
        path: /.terraform

        # Terratest volume
      - name: terratest
        path: /.terratest
steps:
  - id: "decrypt secrets"
    name: gcr.io/cloud-builders/gcloud
    dir: environments/$BRANCH_NAME
    entrypoint: "bash"
    args: ['./config/decrypt.sh']
  - id: "get credentials"
    name: gcr.io/cloud-builders/gsutil
    args: ['cp', 'gs://terraform-test-secrets-staging/service.account.test.json', 'examples/config/service.account.test.json']

  - id: "initialize terraform"
    name: hashicorp/terraform:${_TERRAFORM_VERSION}
    dir: .terratest
    args: ['init']
    waitFor: ['-']

  - id: "run tests"
    name: btowerlabz/docker-cloudbuild-terratest:latest
    dir: test
    entrypoint: /bin/bash
    args: [ '-e', '-o','pipefail', '-c', 'go test -v']
    env:
     - GO111MODULE=on
    waitFor: ['-']
```

Demo

- Deploy Infrastructure Code
- Deploy sample Nest Js App
- Deploy to GKE
- Test application deployment

Thank You

hugs@hashicorp.com | learn.hashicorp.com | discuss.hashicorp.com

Resources

Brikman Y. (2019). *Terraform Up and Running,* Writing Infrastructure as Code, 3rd Edition, USA, O'Reilly Media.

Github: https://github.com/Bascil/hashitalks-africa-resources.git