image

Why some companies do not test IaC

- Mindset / culture "It’s just infrastructure"
- Knowledge gaps, no background with tests mentality
- Tooling gaps
- Complexity of testing some components
- Cost and speed

Feature / Tool	tftest (Python) ⚰️	Terratest (Go)	Native Terraform Test Framework
Language	Python	Go	HCL (Terraform syntax)
What it does	Parses plans and uses Python to assert conditions	Runs Terraform commands, provisions infra, and verifies with Go assertions	Write `.tftest.hcl` files with runs, variables, mocks, and assertions
Support	Terraform only	Terraform, Helm, Docker, Kubernetes	Terraform only
Test type	Unit-style (plan output only, no apply)	Unit + Integration (plan + apply + external checks, HTTP requests, etc.)	Unit (plan) and Integration (apply) with mocking support
Ease of use	Just Python; no infra provisioning	Requires Go knowledge; steeper learning curve	Easiest (stays in HCL)

Plan-based Tests

- 🚀 Very fast (no infra provisioned)
- 💰 Cheap (no cloud resources spin up)
- ✅ No risk of accidentally touching production
- Limited: checks only configuration & plan output
- Quick unit/compliance checks;

Apply-based Tests

- 🐢 Slower (must provision and destroy)
- 💸 Can be expensive (for resources paid per hour)
- ⚠️ Risk if tests run against wrong env
- Full: can check real state + runtime behavior
- Integration, end-to-end, and functional validation of infra.

s3-bucket.tf

```
resource "aws_s3_bucket" "default" {
    bucket = "my-s3-bucket"

    tags = {
        Name        = "ExampleBucket"
        # Environment = "Dev"
    }
}

resource "aws_s3_bucket_versioning" "default" {
    bucket = aws_s3_bucket.default.id

    versioning_configuration {
        status = "Disabled"
    }
}
```

s3-bucket-unit.tftest.hcl

```
run "check-required-tags" {

  command = plan

  assert {
      condition     = contains(keys(aws_s3_bucket.default.tags), "Name") && contains(keys(aws_s3_bucket.def
      error_message = "Check required tags are present"
  }
}

run "check-s3-versioning" {

  command = plan

  assert {
      condition     = aws_s3_bucket_versioning.default.versioning_configuration[0].status == "Enabled"
      error_message = "Check s3 versioning is enabled"
  }
}
```

```
$ terraform test
s3-bucket-unit.tftest.hcl... in progress
  run "check-required-tags"... fail
╷
│ Error: Test assertion failed
│
│   on s3-bucket-unit.tftest.hcl line 8, in run "check-required-tags":
│    8:       condition     = contains(keys(aws_s3_bucket.default.tags), "Name") && contains(keys(aws_s3_bucket.default.tags), "Environment")
│     ├────────────────
│     │ Diff:
│     │ --- actual
│     │ +++ expected
│     │ - true
│     │ + false
│
│ Check required tags are present
╵
run "check-s3-versioning"... fail
╷
│ Error: Test assertion failed
│
│   on s3-bucket-unit.tftest.hcl line 18, in run "check-s3-versioning":
│   18:       condition     = aws_s3_bucket_versioning.default.versioning_configuration[0].status == "Enabled"
│     ├────────────────
│     │ Diff:
│     │ --- actual
│     │ +++ expected
│     │ - "Disabled"
│     │ + "Enabled"
│
│ Check s3 versioning is enabled
╵
```

Terraform linters and checkers

Tool	terraform validate (Native)	TFLint	Checkov
Focus	Syntax & config validation	Linting & best practices	Security & compliance
Strengths	Built-in, fast, ensures HCL is valid	Detects invalid/deprecated provider attributes, configurable, CI-friendly	Large rule set (CIS, NIST, etc.), multi-IaC, custom policies, rich CI integration
Limitations	No best practices or security checks	Doesn’t cover deep security/compliance	Slower, more complex, may give false positives

main.tf

```
variable "instance_type" { # This variable has no type constraint - TFLint will warn about this
  description = "EC2 instance type"
  default = "t2.micro"
}

resource "aws_instance" "problematic_ec2" { # Missing required ami - terraform validate will catch this
  instance_type = var.instance_type
  associate_public_ip_address = true   # Checkov will flag this as a security issue
  root_block_device {
    encrypted = false # Checkov will flag unencrypted volumes
    volume_size = 100
  }
  vpc_security_group_ids = [aws_security_group.wide_open.id] # Security group with all ports open - Checkov will flag this
}

resource "aws_security_group" "wide_open" {
  name        = "allow_all"
  description = "Allow all inbound traffic"

  ingress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"] # Dangerous ingress rule - Checkov will flag this
  }
}
```

```
$ terraform validate
```

```
╷
│ Error: Missing required argument
│
│   with aws_instance.problematic_ec2,
│   on main.tf line 8, in resource "aws_instance" "problematic_ec2":
│    8: resource "aws_instance" "problematic_ec2" {
│
│ "ami": one of `ami,launch_template` must be specified
```

```
$ tflint
```

```
3 issue(s) found:

Warning: terraform "required_version" attribute is required (terraform_required_version)

  on main.tf line 1:

Reference: https://github.com/terraform-linters/tflint-ruleset-terraform/blob/v0.13.0/docs/rules/terraform_required_vers

Warning: `instance_type` variable has no type (terraform_typed_variables)

  on main.tf line 3:
   3: variable "instance_type" {

Reference: https://github.com/terraform-linters/tflint-ruleset-terraform/blob/v0.13.0/docs/rules/terraform_typed_variabl

Warning: Missing version constraint for provider "aws" in `required_providers` (terraform_required_providers)

  on providers.tf line 1:
   1: provider "aws" {

Reference: https://github.com/terraform-linters/tflint-ruleset-terraform/blob/v0.13.0/docs/rules/terraform_required_prov
```

```
$ checkov  --file main.tf
```

```
Check: CKV_AWS_46: "Ensure no hard-coded secrets exist in EC2 user data"
        PASSED for resource: aws_instance.problematic_ec2
        File: /main.tf:8-20
        Guide: https://docs.prismacloud.io/en/enterprise-edition/policy-reference/aws-policies/secrets-policies/bc-aws-secrets-1
Check: CKV_AWS_88: "EC2 instance should not have public IP."
        FAILED for resource: aws_instance.problematic_ec2
        File: /main.tf:8-20
        Guide: https://docs.prismacloud.io/en/enterprise-edition/policy-reference/aws-policies/public-policies/public-12
               8  | resource "aws_instance" "problematic_ec2" {
               9  |   # Missing required ami - terraform validate will catch this
               10 |   instance_type = var.instance_type
               12 |   associate_public_ip_address = true   # Checkov will flag this as a security issue
               14 |   root_block_device {
               15 |     encrypted = false # Checkov will flag unencrypted volumes
               16 |     volume_size = 100
               17 |   }
               19 |   vpc_security_group_ids = [aws_security_group.wide_open.id] # Security group with all ports open - Checkov will flag this
               20 | }

Check: CKV_AWS_126: "Ensure that detailed monitoring is enabled for EC2 instances"
        FAILED for resource: aws_instance.problematic_ec2 File: /main.tf:8-20
Check: CKV_AWS_135: "Ensure that EC2 is EBS optimized"
        FAILED for resource: aws_instance.problematic_ec2 File: /main.tf:8-20
Check: CKV_AWS_8: "Ensure all data stored in the Launch configuration or instance Elastic Blocks Store is securely encrypted"
        FAILED for resource: aws_instance.problematic_ec2 File: /main.tf:8-20
Check: CKV_AWS_260: "Ensure no security groups allow ingress from 0.0.0.0:0 to port 80"
        FAILED for resource: aws_security_group.wide_open File: /main.tf:23-34
```