Testing gives the system a sense of reality, and when handled properly, it can help make your system fail-proof. In the past, testing was done manually, which meant it was error-prone. Today’s automated testing protocols are a boon to developers. Let’s dive into a few of the main tools that are being used to test Infrastructure as Code

1. Terratest

* Built specifically for Terraform.
* Terratest allows for unit testing or testing a section of the code at a time instead of testing the infrastructure. Furthermore, Terratest can perform multiple tests such as tests for credentials and addressing authentications.
* Automated end to end tests rather than just property matching.

1. Terrascan

* Terrascan is a static code analysis tool that can scan IAC from Terraform, K8s manifests, etc.
* The test results tell the severity of the code issues based on high, medium, and low, and provide a noticeable result in code form.
* Terrascan can also be used for local testing before deployment.

1. Kitchen Terraform

* Sponsored by Chef and used by all Chef-managed community workbooks.
* With Kitchen Terraform, the development cycle can now be completed with a test suite.
* It enables a test matrix that can vary in platforms, input variables, and even fixture modules.
* A driver plugin architecture is used to run code on various cloud providers and virtualization technologies such as Vagrant, Amazon EC2, Microsoft Azure, Google Compute Engine, Docker, and more.
* Set of Kitchen plugins to test Terraform code and verify with InSpec controls.

1. Checkov

* Checkov is a static code analysis tool for infrastructure as code (IaC) and also a software composition analysis (SCA) tool for images and open source packages.

1. Terraform compliance

* Lightweight tool which uses BDD syntax making it easy to code.
* Mainly focuses on negative testing instead of having fully-fledged functional tests.
* Terraform compliance requires a state file or a plan file to run against, which means there must be some type of terraform initialization and plan command. The beautiful thing about this tool is that it gives clear and concise test results.4

**Compare Tools**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Terratest | Kitchen-Terraform | Terraform compliance |
| Tech Stack | Go | Ruby | Python, HCL |
| Mani Idea | Automated end to end tests rather than just property matching. | Set of Kitchen plugins to test Terraform code and verify with InSpec controls. | Mainly focuses on negative testing instead of having fully-fledged functional tests. |
| Learning curve | Medium. Have to learn basics of Go. | Medium. Have to understand the Kitchen framework which is configured via yaml + basics of Ruby helps to write tests . | Easy. |
| Supports unit tests | YES | NO | YES |
| Supports integration tests | YES | YES | YES |
| Examples | https://terratest.gruntwork.io/examples/ | https://newcontext-oss.github.io/kitchen-terraform/tutorials/ | https://terraform-compliance.com/pages/Examples/ |

Decision:

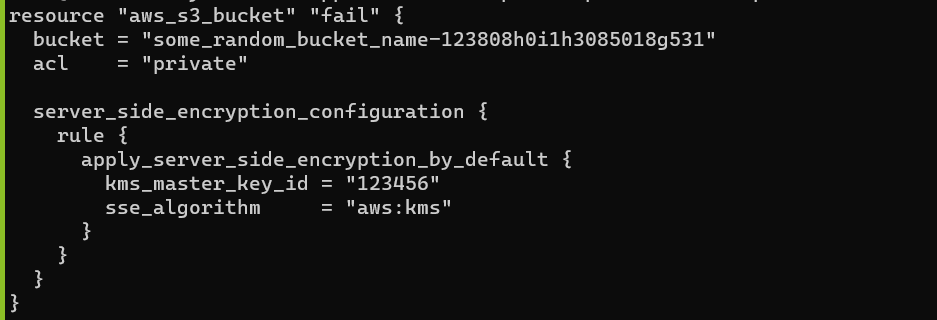
1. Terraform compliance requires a state file or a plan file to run test cases. It’s had easy learn curve. It’s best for unit tests.

Example:

<https://terraform-compliance.com/pages/Examples/>

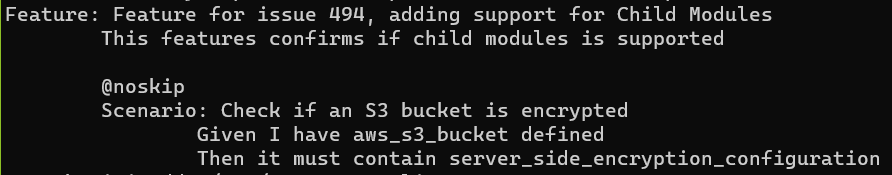
<https://github.com/terraform-compliance/cli/tree/master/tests/functional/test_child_modules>

1. Create main.tf file.



1. Create compliance file

cat compliance/test.feature



1. Run the terraform plan

terraform plan -out=plan.out

1. Run the terraform compliance

terraform-compliance -p plan.out -f chaild-compliance



**Issue:**

1. Code is not implemented for naming convention scenario. We need to write our own python scripts for that.

# Terraform Test

<https://terratest.gruntwork.io/examples/>

There is no proper documentation to implement.

# Kitchen Terraform

<https://newcontext-oss.github.io/kitchen-terraform/tutorials/>

There is no proper documentation to implement.