Test-Driven Development for Infrastructure

Rosemary Wang, @joatmon08
Velocity Berlin | Nov. 7, 2019
“TDD is dead.”

https://dhh.dk/2014/tdd-is-dead-long-live-testing.html
What’s Test-Driven Development (TDD)?

- Write tests firsts, then the feature.
- “Red, green, refactor”
- Benefits:
  - Only build defined features
  - Modular, testable code
“Ideal” Testing Pyramid

- Unit Tests
- Contract Tests
- Integration Tests
- End-to-End Tests
- Manual Testing

Cost (Time, $$)
Infrastructure Testing

Signpost

Manual Testing
- End-to-End Tests
- Integration Tests
- Contract Tests
- Unit Tests

Cost
(Time, $$)
Why is it a signpost?

- #YOLO-driven development
- Lack of education
- Lack of automation / interface
- Lack of tooling
Let’s try TDD for Infrastructure.

- Infrastructure configuration still requires good code practices
- Infrastructure architecture has definition of “feature complete”
- `terraform yolo isn’t a thing` (or insert deploy command here).
Target State
TDD Up the Pyramid

- Unit Tests
- Contract Tests
- Integration Tests
- End-to-End Tests
- Manual Testing

Cost
(Time, $$$)
Warning, Some Terraform

VS Code Live Share at:
hashi.co/tdd-velocity
Unit Testing
Unit Tests

- Test individual component
- Can be run without dependencies
- Infrastructure: check specific configuration & syntax
Live Code

- We’ll use `conftest` (Open Policy Agent) & `terraform validate`.
- Terraform-specific tools, compiled by Peter Souter:
  - `clarity`
  - `terraform-compliance`
  - `terraform_validate`
- Other infrastructure: native testing frameworks
Target State

VPC (10.128.0.0/25)

eu-central-1

eu-central-1a

Private subnet
10.128.0.32/28
database

Public subnet
10.128.0.0/28
application

80
8099
27017
Unit TDD

Fast feedback
Architectural Conformance
Lint and check syntax

Function logic?
Execution?
Resource dependencies?
Contract Testing
Contract Tests

- Validate interactions between 2 components (input & output)
- “Real” resources not required
- Compare infrastructure state
Live Code

- We’ll use `conftest` (Open Policy Agent) & `terraform plan`.
- Terraform-specific tools: `kitchen-terraform`
- Can also use native testing frameworks
  - Example for network switches with Python
  - Example for S3 Bucket Policy with Golang
  - If Kubernetes, check Kubernetes to external components (i.e., DNS)
Target State
Contract TDD

🤔 Resource dependencies?
😊 Interactions?

Fast feedback
Check functional logic
Catch our input errors
More holistic
Integration Testing
Integration Tests

- Confirms interactions between 2+ components
- Real resources required
- May also include acceptance tests
**IMPORTANT**

- Integration tests are about **interactions**.
- When using IaC tool, testing *deployment* is redundant.
  - In Terraform, covered by Provider Acceptance Tests
- Many types of integration tests, including Functional, Policy, & Security
Testing with Kitchen
Tools

- Terraform-specific tools: terratest, kitchen-terraform, Sentinel
- Other tools:
  - AWS localstack / mocking framework (USE WITH CAUTION)
  - Kubernetes local environments (KIND, Minikube, etc.)
  - Various Inspec resource packs
Fails!

> kitchen test

----> Starting Kitchen (v2.3.3)

... Waiting for SSH service on 54.93.35.169:22, retrying in 3 seconds
Waiting for SSH service on 54.93.35.169:22, retrying in 3 seconds
Waiting for SSH service on 54.93.35.169:22, retrying in 3 seconds
Waiting for SSH service on 54.93.35.169:22, retrying in 3 seconds
Waiting for SSH service on 54.93.35.169:22, retrying in 3 seconds
Passing the Test
Pass!

> kitchen test
-----> Starting Kitchen (v2.3.3)

Profile: Integration Tests for Application (default)

- db: Database: check routing from public to private subnet
- Host 10.128.0.43 port 27017 proto tcp should be reachable
- Host 10.128.0.43 port 27017 proto tcp should be resolvable
- Host 10.128.0.43 port 80 proto tcp should not be reachable
- outbound: Public Subnet: check routing out to public internet
- HTTP GET on https://hashicorp.com status should cmp == 301

Profile Summary: 2 successful controls, 0 control failures, 0 controls skipped
Test Summary: 3 successful, 0 failures, 0 skipped
Integration TDD

Convert theory to reality
Caught a missing component
Isolate sections of system

Not that fast
Requires real resources
End-to-End Testing
End-to-End Tests

- “The Real Deal”
- Can be manual, fully automated, or smoke tests
- If it starts smoking, it doesn’t work.
TDD it!
Did it fail?
Tools

- Terraform-specific tools: terratest, Sentinel
- Other:
  - Behavior-Driven Development (BDD) Frameworks
  - Can also use native testing frameworks
  - Kubernetes: Sonobuoy conformance tests, local environments (KIND, Minikube, etc.)
Final Thoughts
Ugh, so much effort.

- (Yes, it can be.)
- Unit testing is useless here!
- I don’t find testing tools for this!
- Stop with pedantic practices!
[Preaching] TDD is dead.
So is #YOLO-driven development.
Use TDD to Learn Infrastructure

- Confidence in infrastructure-as-code
- Develop knowledge of infrastructure
- More useful integration & end-to-end tests
- Tacit knowledge of a change’s blast radius
Eventually, YDD & TDD of infrastructure are dead...to you.
Please rate the session!

Rosemary Wang
she/her
Developer Advocate at HashiCorp
rosemary@hashicorp.com

@joatmon08
joatmon08
linkedin.com/in/rosemarywang/