Infrastructure Testing with Molecule

Elana Hashman
Rackspace Managed Security
AnsibleFest 2017 – San Francisco, CA

@ehashdn

Outline

- Define “Infrastructure as Code”
- Motivate the importance of infrastructure testing
- Discuss available infrastructure testing solutions for Ansible
- Walk through Molecule, a testing platform for Ansible
- Demo!
- Conclusion, Q&A

✨ INFRASTRUCTURE AS CODE ✨

Using techniques from software development to build computing infrastructure.

Traditional Operations Team

- Human operators manage computers
- Rely on runbooks, escalation paths, and on-call rotations
- Much of team culture is oral/implicit
  - Communication skills and collaboration are very important
- Team scales linearly with the number of computers supported
- Transitioning to a DevOps/SRE model is seen as costly

Bringing Software Dev Best Practices to Ops

- Codifying runbooks inside software
- Identifying reusable common logic
- Using version control
- Performing collaborative reviews
- Building automated tests

Benefits of Codifying your Runbooks

- Implicit knowledge/oral tradition becomes explicit/written
- Accuracy and scalability over human operators
- Reduction or removal of ambiguity in runbooks
- Ease of identification of areas for code reuse

Benefits of Using Version Control

- Centralized repository of the deployment specifications for your infrastructure
- Ease of rollbacks
- Bisection between commits to diagnose when bugs were introduced
- Branching to independently work on many new features
- Explicit revisions and change sets identified for change management

Benefits of Performing Collaborative Reviews

- Significant reduction in defect count
- Enables group ownership of infrastructure
- Facilitates discussions around technical decision-making
- Provides a written record of why things changed, with context
- Gives peers the opportunity to hold each other accountable
- Provides a learning opportunity for junior team members

Benefits of Building Automated Tests

- Marginal cost of adding new tests is nearly constant
- Exponentially greater test combinations possible
- Ease of regression testing, for faster development cycle
- Improved quality and confidence in (low) defect rates

“Infrastructure as Code” in the Ansible Community

✔ Codifying runbooks inside software

✔ Identifying reusable common logic

✔ Using version control

? Performing collaborative reviews

✘ Building automated tests

What tools are available for Ansible infrastructure testing?

Testing Ansible with Ansible

- Benefits:
  - As flexible and powerful as you need it to be
- Issues:
  - High development cost (“not invented here”)
  - Assertions in production code can lead to unexpected failures
  - Ansible can't detect bugs in Ansible
  - Need to write your own provisioner

Testing Ansible with ansible-test

- Benefits:
  - Simple tool with small codebase
  - Written in Python
  - Solves the provisioning problem in “test Ansible with Ansible”
- Issues:
  - Doesn't support verifiers, just executes Ansible and checks result
  - Only supports a Docker provisioner on Debian-based images
  - Does not appear to be actively maintained

Testing Ansible with Test Kitchen

- Benefits:
  - Large contributor community: well-maintained and robust
  - Supports Ansible
  - Supports Windows testing for Ansible
- Issues:
  - Written in Ruby
  - Verifiers are Ruby or bash based
  - Installs Ansible on the target host and runs it locally

Testing Ansible with Molecule

- Benefits:
  - Ansible-native, written in Python
  - Two year old stable project with established community
  - Support for Python-based verification with testinfra
- Issues:
  - No Windows support
  - Only supports Ansible provisioning (e.g. no Chef support)

How Molecule Testing Works

1) Creates nodes for testing (docker, VMs, cloud, etc.)
2) Runs your playbook on the nodes
3) Runs your playbook again to test for idempotence
4) Lints your Ansible code with ansible-lint and flake8
5) Runs your verifier tests on your nodes to ensure state is what we expect

Using Molecule for Infrastructure Testing

- What sorts of things can we test for?
  - Files exist and have the correct permissions
  - Services are running
  - A user exists and is the member of the correct groups
  - Software interactions work as expected, i.e. GET from the web server requires basic authentication
- What can this automation be used for?
  - Checking against compliance benchmarks
  - Service status verification

Molecule Demonstration

- Let's try it!
  - Create a node
  - Converge a node
  - Check for idempotence
  - Lint and verify role against tests
- **Github Repo:** https://github.com/ehashman/ansiblefest

❇ LIVE DEMO ❇

^ kind of looks like the Molecule logo

Conclusion

- Treating infrastructure as code offers many benefits
- Writing automated tests for your Ansible automation gives you more confidence in your infrastructure and lets you test much more
- There are a number of different testing solutions available for Ansible: Molecule is an Ansible-native, robust option
- Molecule allows you to create, converge, lint, verify, and idempotence-check your infrastructure

Questions?

Thank you!

Thanks to: Noah Kantrowitz, Clifton Sigler, Rackspace

Talk links, references, and resources can be found at
https://hashman.ca/ansiblefest-2017/