Add Continuous Compliance to Your CI/CD Pipelines

November 4, 2019
Overview

Software development teams have long been able to take advantage of unit, integration, and functional testing as an integral part of a robust, test and behavior-driven development environment. Infrastructure as Code (IaC) provides new capabilities for DevOps teams to utilize new frameworks to build ephemeral environments with integrated compliance testing before, during, and after deployment.
Problem Statement: I need to ensure compliance in my cloud environments.
## Actively Monitoring Environments

<table>
<thead>
<tr>
<th>Provider Solutions</th>
<th>Provider Solutions</th>
<th>External Monitoring</th>
</tr>
</thead>
<tbody>
<tr>
<td>• AWS Config</td>
<td>• Azure Advisor</td>
<td>• CloudCheckr</td>
</tr>
<tr>
<td>Cost structure, near real time, 100% customizable</td>
<td>Cost, security, and configuration optimization</td>
<td>• Prisma Cloud</td>
</tr>
<tr>
<td>• AWS Inspector</td>
<td>• AWS Trusted Advisor</td>
<td>• DivvyCloud</td>
</tr>
<tr>
<td>Access to EC2 instances, software vulnerabilities</td>
<td></td>
<td>• Qualys</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Rapid7</td>
</tr>
</tbody>
</table>
Problem solved....ish
Monitoring Active Environments

Tools are validating the configuration of active environments.

- Vulnerabilities are found
- Tickets for the backlog
- Prioritized based on risk
- More often than not, left exposed for a period of time.
Refined Problem Statement:
I need to ensure compliance in my cloud environments before they are created.
Test-Driven Development (TDD)

- Add a test
- Run test to see the test fail
- Write code to satisfy the conditions of the test
- Run test to see the test pass
- Refactor
- Repeat as necessary
If your infrastructure is in code, why is development of your IaC environments any different than the process developers use to write applications?
# Infrastructure as Code Toolbox

<table>
<thead>
<tr>
<th>Kitchen</th>
<th>LocalStack</th>
<th>Clever Thinking</th>
</tr>
</thead>
</table>
| https://kitchen.ci
Kitchen provides a test harness to execute infrastructure code on one or more platforms in isolation. | https://localstack.cloud
LocalStack provides a test framework on your local machine providing the same functionality and APIs as the real AWS cloud environment. | ⚫ Start with the mentality tests are required.
⚫ Utilize orchestration like Jenkins or CircleCI to build private, ephemeral environments to test and validate changes |
Let’s walk through an example

- AMI Build Process
- ChefSpec Unit Tests
- Kitchen for Integration and Functional Testing
- Jenkins Pipeline
Components of the Example:

- Chef Cookbook - cis_mitigation
  - ChefSpec Unit Testing
  - Kitchen Configuration
- Packer Machine Definition
- Chef Cookbook - ami_builder
  - ChefSpec Unit Testing
  - Kitchen Configuration

WAIT!! We don’t use Chef, we use <insert name of tool>!
ChefSpec Example - Chef Cookbook cis_mitigation

ChefSpec Tests

```ruby
require 'spec_helper'

describe 'cis_mitigation::cis' do
  before do
    stub_command([]).and_return(false)
  end

  it 'configure selinux' do
    expect(chef_run).to create_cookbook_file('/etc/selinux/config')
  end

  it 'configure logrotate' do
    expect(chef_run).to create_template('/etc/logrotate.d/syslog')
  end

  it 'disable x11 forwarding' do
    expect(chef_run).to run_exec('ensure_x11_forwading_disabled')
  end
end
```

Bash Output:

```
call cis_mitigation::var_tmp_mount
  mount /tmp
  should mount mount "/tmp"
  should mount mount "/tmp"
  should mount mount "/tmp"

 cis_mitigation::var_tmp_mount
  mount /var/tmp
  should mount mount "/var/tmp"
  should mount mount "/var/tmp"
  should mount mount "/var/tmp"

Finished in 9.25 seconds (files took 2.99 seconds to load)
14 examples, 0 failures

ChefSpec Coverage report generated...
Total Resources:   9
Touched Resources: 9
Touch Coverage:    100.0%

You are awesome and so is your test coverage! Have a fantastic day!
```

kitchen.yml

---
driver:
  name: ec2
aws_ssh_key_id: chef-test-kitchen-20191020
security_group_ids: ["sg-asdf1234"]
region: us-west-2
subnet_id: subnet-1234asdf
associate_public_ip: false
interface: private
tags:
  Name: test-kitchen-cis-mitigation
  Owner: "DevOps Group"
provisioner:
  name: chef_zero
transport:
  ssh_key: ~/.ssh/chef-test-kitchen-20191020
connection_timeout: 10
connection_retries: 5
name: sftp
---
platforms:
- name: ubuntu-18.04
driver:
  transport:
    username: ubuntu
image_search:
  owner-id: "099720109477"
  name: "*ubuntu-bionic-18.04-amd64-server*"
block_device_mappings:
- device_name: /dev/sda1
ebs:
  volume_type: gp2
  volume_size: 20
  delete_on_termination: true
suites:
- name: default
  provisioner:
    policyfile: policyfiles/kitchen.rb
verifier:
  name: inspec
Kitchen Example - Chef Cookbook cis_mitigation

Kitchen Commands

- kitchen list
  ○ Lists all of the test suites available for each platform.
- kitchen create
  ○ Create the test instance using the kitchen driver
Kitchen Example - Chef Cookbook cis_mitigation

**Kitchen Commands**

- **kitchen converge**
  - Apply the Chef run list to the test instance
Kitchen Example - Chef Cookbook cis_mitigation

Kitchen Commands

- `kitchen verify`
  - Run the Inspec tests on the test instance

Version: (not specified)
Target: ssh://ubuntu@172.16.1.94:22

File /var/tmp
  - should exist
  - should be mounted

File /etc/selinux/config
  - should exist
  - content should match /SELINUX=enforcing/

File /etc/logrotate.d/syslog
  - content should match /var/log/cron/
  - content should match /var/log/maillog/
  - content should match /var/log/messages/
  - content should match /var/log/secure/
  - content should match /var/log/spooler/
  - content should match /var/log/boot.log/

File /etc/sshd_config
  - content should match /X11Forwarding no/

File /root/images/tmpfile.bin
  - should exist

File /tmp
  - should exist
  - should be mounted

Test Summary: 14 successful, 0 failures, 0 skipped
Finished verifying <default-ubuntu-1804> (0m8.69s).
-----> Kitchen is finished. (0m14.10s)
eggerling@bender:~/summit-2019/chef/cis_mitigations$
Kitchen Example - Chef Cookbook cis_mitigation

Kitchen Commands

- kitchen login
  - Connect to the test instance and allow manual inspection of the instance
- kitchen destroy
  - Destroy the test instance
Example Inspec Test

# CIS 1.6.1 Configure SELinux
describe file('/etc/selinux/config') do
  it { should exist }
  its('content') { should match(/^SELINUX=enforcing/) }
end

# CIS 4.3 Ensure logrotate is configured
describe file('/etc/logrotate.d/syslog') do
  its('content') { should match(%r{^/var/\log/cron}) }
end

Example Inspec Test

# Test for Amazon SSM Agent installed and active
describe package('amazon-ssm-agent') do
  it { should be_installed }
end

if os[:family] == 'debian' || (os[:family] == 'redhat' && os[:release].to_i == 7)
describe systemd_service('amazon-ssm-agent') do
  it { should be_installed }
  it { should be_enabled }
end
end


# CIS 1.1.9 Ensure noexec option set on /var/tmp partition
if os[:family] == 'ubuntu'
    describe command('/tmp/testfile.sh') do
        its('exit_status') { should eq 3 }
    end
end

if os[:family] == 'redhat'
    describe command('/tmp/testfile.sh') do
        its('exit_status') { should eq 1 }
    end
end

if os[:family] == 'redhat'
    describe command('/usr/local/bin/aws --version') do
        its('exit_status') { should eq 0 }
    end
end

if os[:family] == 'redhat' && os[:release].to_i == 6
    describe upstart_service('amazon-ssm-agent') do
        it { should be_installed }
        it { should be_enabled }
        it { should be_running }
    end
end
Kitchen Building Blocks

Now we have the general building blocks, how can we use them?

- Integration Tests
- Functional Tests
Kitchen Building Blocks

Start with the Chef Cookbooks

- ChefSpec Unit Tests
- Kitchen Integration Tests
Kitchen Building Blocks

AMI Builder

- Packer Machine Definition
- AMI Builder Cookbook
  - ChefSpec Unit Tests
  - Kitchen Integration Tests
- AMI Functional Tests
Wrap It Up With Jenkins

Using a Jenkins Pipeline script, automate the entire process checking for failure along the way.
Jenkins Pipeline Example

```groovy
stage("Lint") {
    sh "foodcritic -f correctness ami/cookbooks/ami-builder"
    dir ('ami/cookbooks/ami-builder') {
        sh "rubocop -r cookstyle ami/cookbooks/ami-builder"
    }
}

stage("Spec") {
    dir ('ami/cookbooks/ami-builder') {
        sh "chef exec rspec"
    }
}

stage("Integration Kitchen-Create") {
    dir ('ami/cookbooks/ami-builder') {
        sh "kitchen create integration-pre-packer-ubuntu-1804"
    }
}

stage("Integration Kitchen-Converge") {
    dir ('ami/cookbooks/ami-builder') {
        sh "kitchen converge int-pre-packer-ubuntu-1804"
    }
}

stage("Integration Kitchen-Verify") {
    dir ('ami/cookbooks/ami-builder') {
        sh "kitchen verify integration-pre-packer-ubuntu-1804"
    }
}

stage("Integration Kitchen-Destroy") {
    dir ('ami/cookbooks/ami-builder') {
        sh "kitchen destroy integration-pre-packer-ubuntu-1804"
    }
}

stage("Berks Vendor Cookbooks") {
    dir ('ami') {
        sh "rm -rf berks_vendor/
    }
}
```
Jenkins Pipeline Example

```groovy
stage("Berks Vendor Cookbooks") {
  dir ('ami') {
    sh "rm -rf berks_vendor/
    sh "mkdir -p berks_vendor"
    sh "
      for DIR in `ls cookbooks/`; do
        berks vendor berks_vendor -b "cookbooks/$DIR/Berksfile"
      done
    ""
  }
}

stage("Functional Kitchen - Create") {
  dir ('ami/cookbooks/ami-builder') {
    sh "sed -i 's/===ubuntu-1804ImageId===/${amiId}/g' .kitchen.yml"
    sh "kitchen create functional-post-packer-ubuntu-1804"
  }
}

def amiId = ""

stage("Run Packer") {
  dir ('ami') {
    sh ""
    [ packer build -machine-readable -var branch=${env.BRANCH} base_ami.json; echo "\$?"; ] | tee packer_build_output.txt
    tail -n 1 output.txt > exit_code.txt
    grep "artifact,0,id" output.txt | cut -d":" -f 2 > ami_id.txt
    packerExitCode = readFile "exit_code.txt"
    if (packerExitCode.trim() != "0") {
      error("Packer failed")
    }
    amiId = readFile "ami_id.txt"
  }
}
```
Jenkins Pipeline Example

stage("Functional Kitchen-Create") {
dir ('ami/cookbooks/ami-builder') {
  sh 'sed -i s/===ImageId===/${amiId}/g .kitchen.yml'
  sh "kitchen create functional-post-packer-ubuntu-1804"
}
}

stage("Functional Kitchen-Verify") {
dir ('ami/cookbooks/ami-builder') {
  sh "kitchen verify functional-post-packer-ubuntu-1804"
}
}

stage("Functional Kitchen-Destroy") {
dir ('ami/cookbooks/ami-builder') {
  sh "kitchen destroy functional-post-packer-ubuntu-1804"
}
}
We have a Jenkins Pipeline to build and test an AMI. What’s next?

- Rapid7
- Qualys
- Encryption
- Sharing With Accounts
Thank you!

Eric Gerling, CTO, Trility Consulting
eric@trility.io

https://github.com/Trility/cloud-devops-security-summit-2019