Testing Puppet with Bitbucket Pipelines

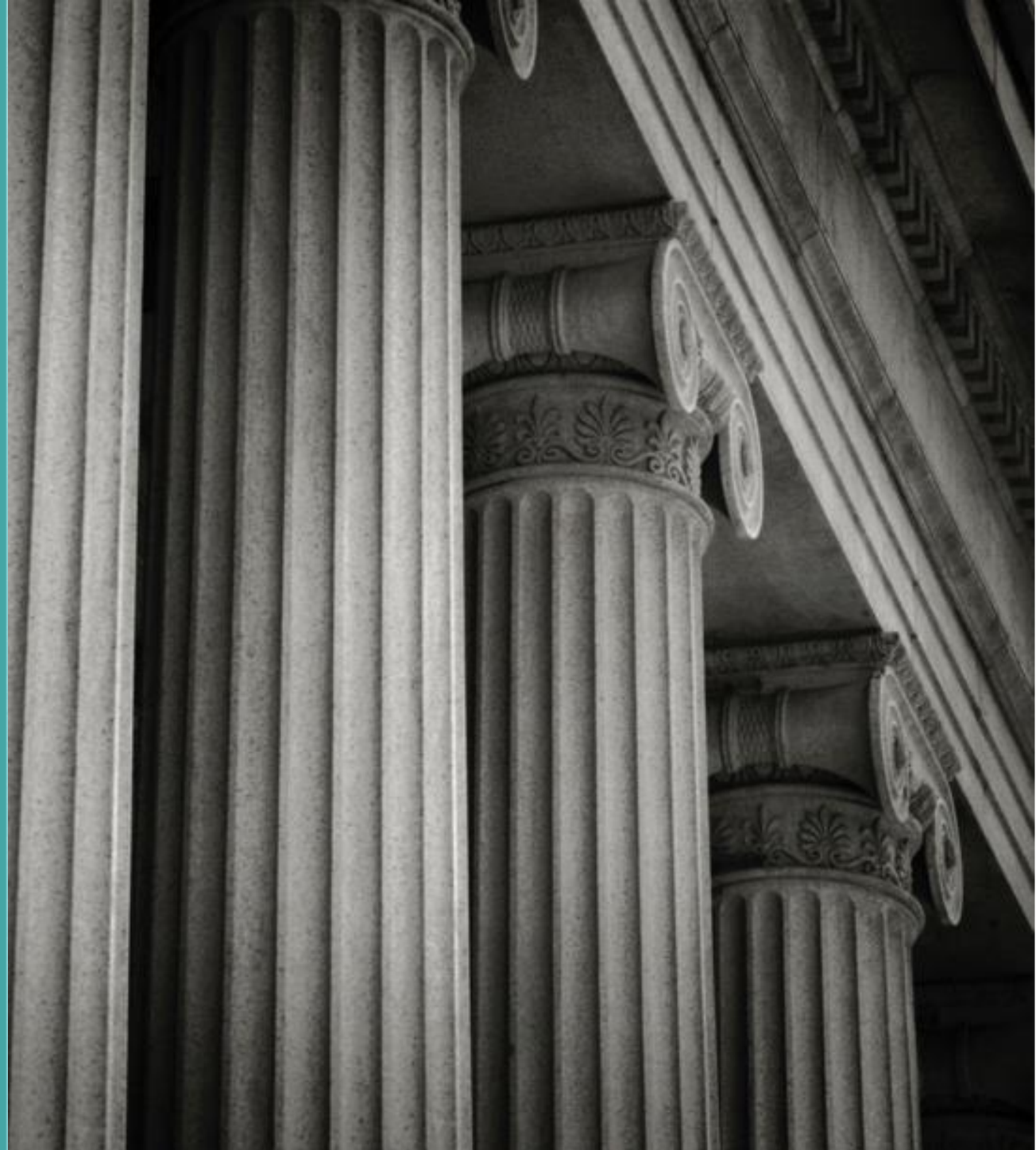
May 25, 2017

sourced

Overview

- Introduction
- The anatomy of the Puppet module
- Puppet code validation and testing
- Testing challenges
- Bitbucket Pipelines
- Bitbucket Pipelines demonstration
- Additional thoughts and comments

sourced



Who are Sourced?

Adopting cloud services within an enterprise requires experience

Historically

- Sourced Group were founded in 2009
- Significant Financial Services background
- Specialize in Configuration Management, Automation, Cloud Computing & Data Management
- Achieved a number of industry firsts in these fields
- Offices in Australia and Canada
- Delivery experience in Amazon Web Services, Microsoft Azure & IBM SoftLayer

Major in-flight Projects

- 80% data center migration to AWS for a large airline
  - Includes an Application Delivery Framework
  - Policy and guidance to underpin this activity
- Development of a strategic cloud environment for a global investment bank
  - Engage with internal stakeholders to define a public cloud environment that is capable of housing material workloads
- On-going assistance on the ‘cloud journey’ for large Canadian telco
  - Full business migration of electronics medical records suite of products to AWS

sourced

Our Partnerships

Strategic partnerships that align with our customer-centric approach

Me

Who is this guy anyway?

Keiran Sweet

- Senior Consultant with Sourced Group
- Previously Puppet lead for a large financial organisation
- Presented at multiple Puppet conferences and camps
- Background
  - Linux & UNIX System Administration and Architecture
  - Deployment & Integration with Cloud Providers (AWS / Azure / VMware)
  - Puppet user since ~2008/2009
- Dog Enthusiast

sourced

The Anatomy of the Puppet module

sourced

Puppet module anatomy

Quick recap of the foundations

- Puppet code is written as Puppet *manifests (*.pp)*
- Puppet code is distributed as modules
- A module should *do one thing and do it well*
- Manifests are compiled by Puppet into *catalogues*
- The Puppet compiler evaluates
  - *Facts*
  - *Parameters*
  - *Manifest logic*

sourced

Puppet module anatomy

Quick recap of the foundations

- The catalogue expressed as *JSON*
- It is consumed by the *Puppet agent* to invoke change
- Modules may also contain
  - *Data*
  - *Test cases*
  - *Documentation*

sourced

Puppet code validation and testing

sourced

Puppet syntax checks

Ensuring code validity

- Puppet parser validate
  - Ensure that the Puppet code is free of syntax errors
- Template syntax validation
  - Ensure that the ERB / EPP template code is free of syntax errors
- Hiera / YAML syntax validation
  - Ensure that the Hiera code is free of syntax errors (ie, Tabs)
- metadata.json syntax validation
  - Ensure the file is JSON compliant and contains all the required fields for Heira 5 and the Puppet module tool

sourced

Puppet style conformance

Keeping it clean

- Puppet Lint
  - Validates Puppet manifests against the style guide
- Puppet Strings
  - Validates Puppet manifests against YARD framework
  - Ensures that your code is documented
    - Warns* on undocumented
      - Classes
      - Parameters
      - Types

SPACES INSTEAD

OF TABS !? !?

sourced

Puppet module content tests

Ensuring the module repository is …

- Compliant with .gitignore
- Free of temporary files
  - .DS_Store
  - .idea
  - *.tmp
  - Symlinks
- Aligned with site specific requirements
  - ie, Free of binaries

FRIENDLY REMINDER

NO BINARIES IN THE GIT

sourced

Unit Testing is a level of software **testing** where individual components of software are tested. The purpose is to validate that each **unit** of the software performs as designed.

Puppet unit tests

Expecting the unexpected

- rspec-puppet
  - Ruby Unit Testing framework for Puppet catalogues
  - Define a set of test cases for the module
  - Compile suitable catalogs
  - Validate the contents
  - Fail on the unexpected / undesired

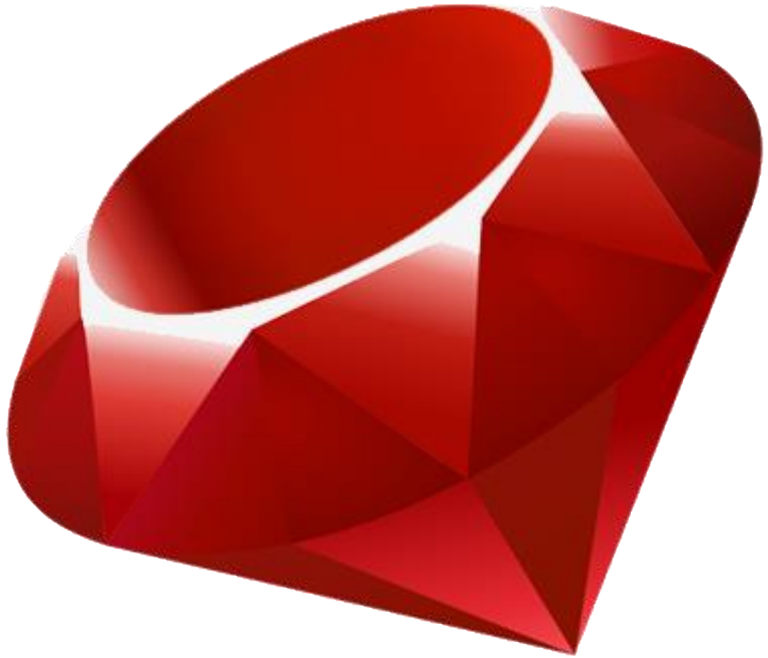
```
require 'spec_helper'
describe 'udeploy' do
  let(:title) { 'udeploy' }
  let(:node) { 'rhel.domain.tld' }

context 'RedHat6' do
  let(:facts) { {
      :osfamily => 'RedHat',
      :operatingsystem => 'RedHat',
      :architecture => 'x86_64',
      :path => '/usr/bin/:/usr/local/bin/',
      :is_opt_common_java   => true,
      :os => {
          'family' => 'RedHat',
          'release'=> {
              'major' => '6',
          }
      },
  } }

  it { is_expected.to contain_user('svc_udeploy').with(
    :ensure           => 'present',
    :gid              => 'svc_udeploy',
    :managehome       => true,
    :password         => '!!',
    :password_max_age => '-1',
    :password_min_age => '1',
    :shell            => '/bin/bash',
  ) }


....


end
```

sourced

Challenges

Some things make testing harder that we'd like

- *Ruby Ruby Ruby Ruby*
  - Operating system challenges
    - Windows Ruby pain (Official, RVM)
    - Linux Ruby pain (OS Native, RVM)
    - OSX Ruby pain (OS Native, RVM, Homebrew)
  - Ruby Gems
    - Cross platform support can be an issue
- IDE's are great – Tooling and editor wars .. Not so much..

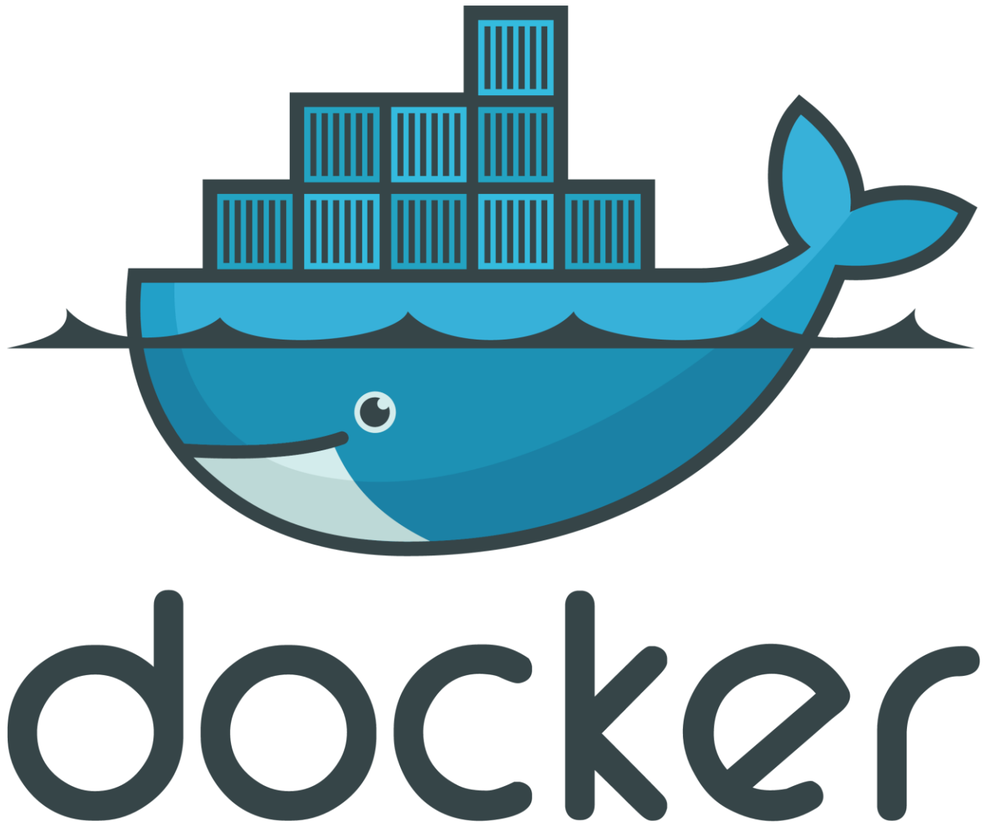


sourced

Solutions

Docker to the rescue !

- Enables rapid provisioning / teardown of runtime environments.
- Vast library of different images that provide language runtimes
- Workflow
    1. Launch a Ruby container
    2. Place the Puppet code within it
    3. Install the relevant dependencies
    4. Execute all required tests
    5. Mark it as a Success or a Failure
    6. Discard when completed



sourced

Still need to have each developer have a standard Docker environment on their workstation..

Would be ideal to have this as part of the workflow…

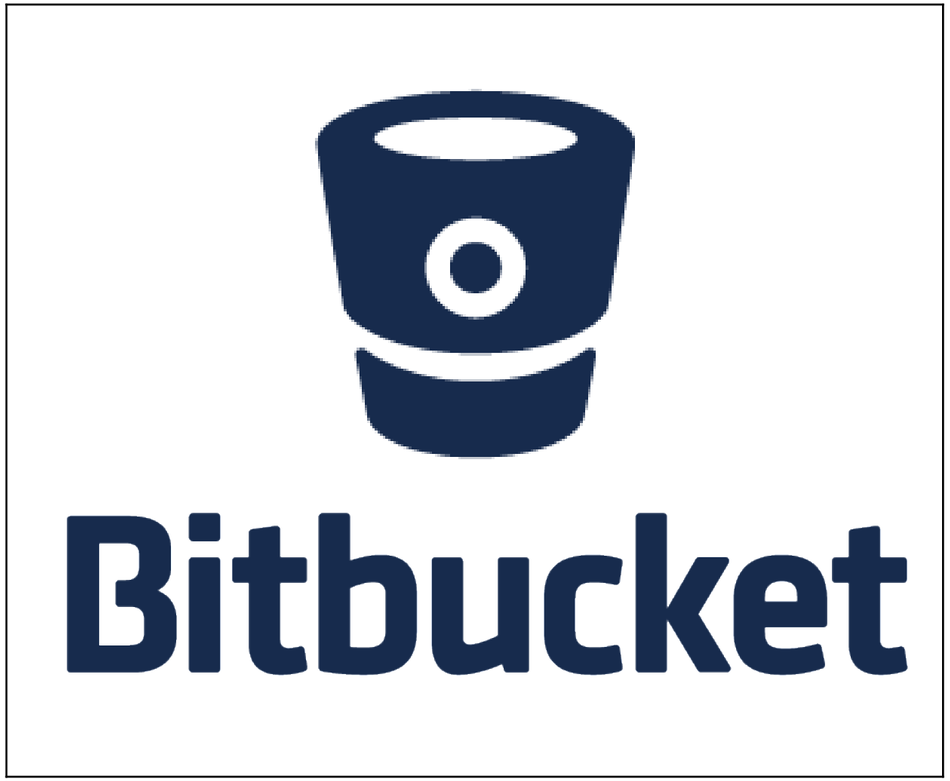
sourced

Bitbucket Pipelines

Bitbucket Pipelines

What is it?

- Bitbucket - Atlassian's hosted Git service
- Pipelines adds build capabilities to Bitbucket repositories
- Launches a Docker container on git events
- Handles placing the code in the container for you
- Provides a configuration file for you to specify
  - The container type
  - Tasks to execute within it
- Integrates with branch permissions



sourced

Bitbucket Pipelines

Setting it up

- Enable Pipelines on your git repository
- Setup Puppet module
  - Add your testing framework and tests
  - Create a *bitbucket-pipelines.yml*
    - Required Docker image
    - Commands to execute your tests
- Setup your branch permissions
- Push your code and check your results

```
image: ruby:2.3.0
pipelines:
    default:
       step:
          script:
            - ruby --version
            - bundler --version
            - bundle install
            - rake -T
            - rake lint
            - rake validate
            - rake check:dot_underscore
            - rake check:git_ignore
            - rake check:symlinks
            - TRUSTED_NODE_DATA=yes rake spec
            - rake strings:generate
```

sourced

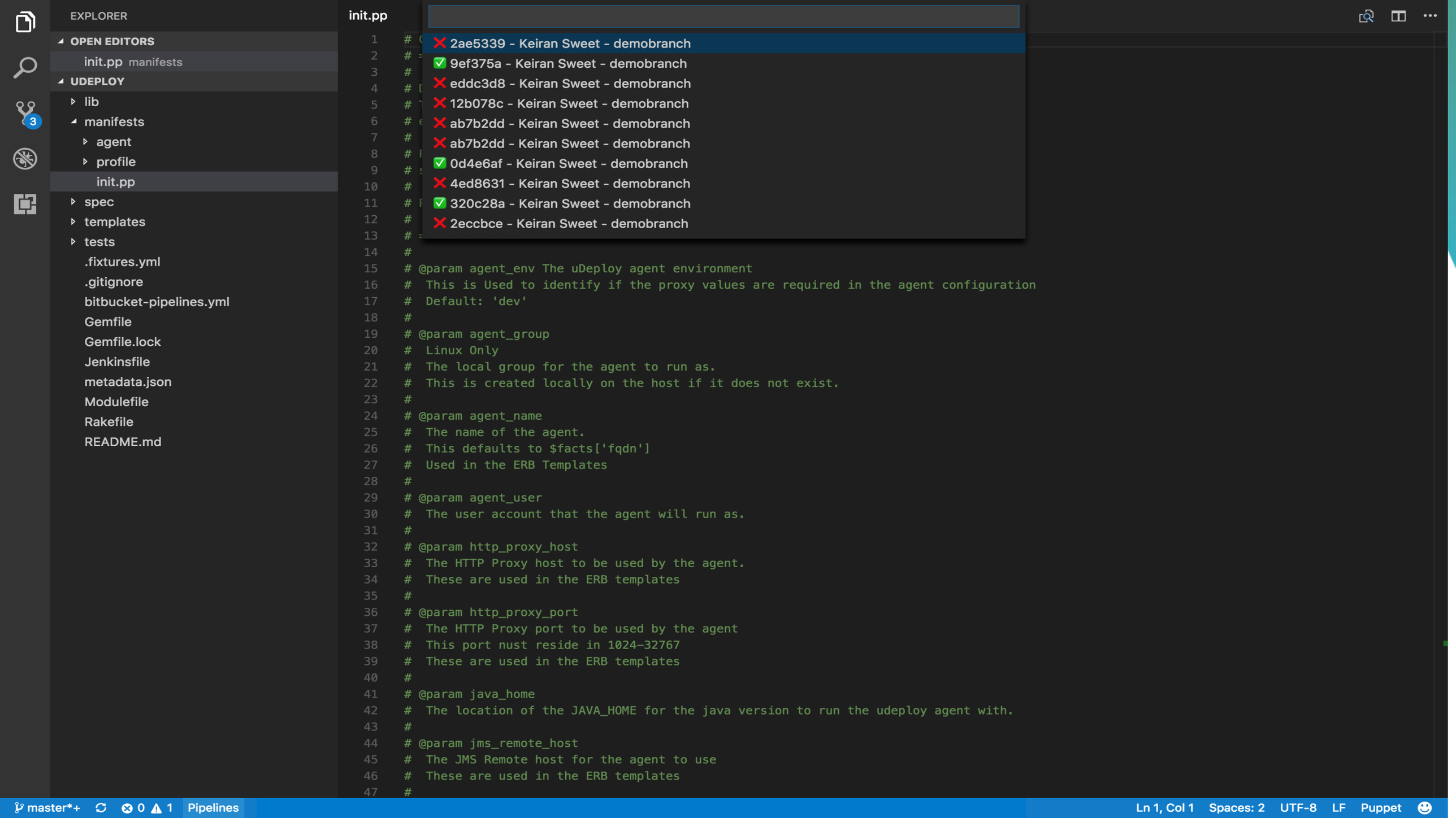
Demonstration

Additional thoughts and comments

Other things worth checking out

- Pipeline service containers
- IDE Capabilities
  - Ruby Mine
    - Mature Puppet support
    - New Docker Support
  - Visual Studio Code
    - Luke Bachelor's Pipelines plugin
- Thanks
  - Geoff Williams – Puppet / Declarative Systems

sourced


EXPLORER
OPEN EDITORS
init.pp manifests
UDEPLOY
lib
manifests
agent
profile
init.pp
spec
templates
tests
.fixtures.yml
.gitignore
bitbucket-pipelines.yml
Gemfile
Gemfile.lock
Jenkinsfile
metadata.json
Modulefile
Rakefile
README.md
init.pp
2ae5339 - Keiran Sweet - demobranch
9ef375a - Keiran Sweet - demobranch
eddc3d8 - Keiran Sweet - demobranch
12b078c - Keiran Sweet - demobranch
ab7b2dd - Keiran Sweet - demobranch
ab7b2dd - Keiran Sweet - demobranch
0d4e6af - Keiran Sweet - demobranch
4ed8631 - Keiran Sweet - demobranch
320c28a - Keiran Sweet - demobranch
2eccbce - Keiran Sweet - demobranch
@param agent_env The uDeploy agent environment
This is Used to identify if the proxy values are required in the agent configuration
Default: 'dev'
@param agent_group
Linux Only
The local group for the agent to run as.
This is created locally on the host if it does not exist.
@param agent_name
The name of the agent.
This defaults to $facts['fqdn']
Used in the ERB Templates
@param agent_user
The user account that the agent will run as.
@param http_proxy_host
The HTTP Proxy host to be used by the agent.
These are used in the ERB templates
@param http_proxy_port
The HTTP Proxy port to be used by the agent
This port nust reside in 1024-32767
These are used in the ERB templates
@param java_home
The location of the JAVA_HOME for the java version to run the udeploy agent with.
@param jms_remote_host
The JMS Remote host for the agent to use
These are used in the ERB templates
master*+
0 1
Pipelines
Ln 1, Col 1
Spaces: 2
UTF-8
LF
Puppet

Questions ?

sourced

Previous Presentations

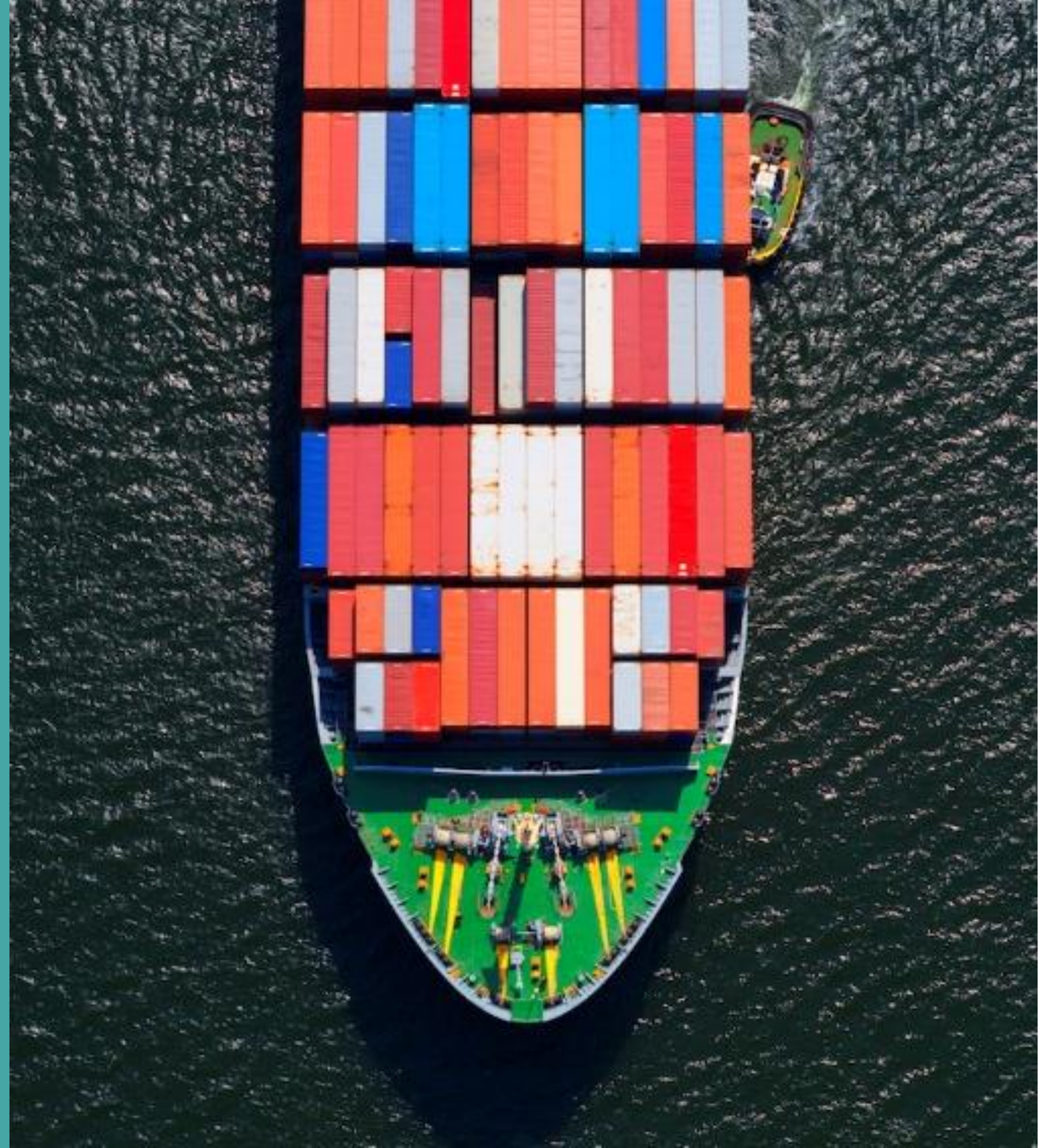
- Using Puppet in Automated Environments
  - *Unlocking Azure with Puppet Enterprise*

    Sourced Group, Puppetcamp 2016
  - *Order in a world of snowflakes*

    Sourced Group, Puppetconf 2015
- Using Puppet in Dynamic Environments
  - *The Evolving Design Patterns of Puppet Enterprise*

    Sourced Group, Puppetconf 2014
- Using Puppet with Multiple Cloud Providers
  - *Using Puppet as heterogeneous cloud glue*

    Sourced Group, Puppetconf 2012



sourced

Preventing merges without successful tests

sourced

Ensure repository is configured correctly

sourced

Create a new branch with some faulty code

sourced

Open a pull request and attempt to merge

sourced