Getting Started with Ansible & ServerSpec

Open Repositories 2017
Brisbane, Australia
June 27, 2017
Half-day workshop, afternoon

Agenda

- 1/2 hour: Intro to ServerSpec
- 1 1/2 hours: Intro to Ansible
- 1/2 hour: Testing Strategies for Ansible
- 1/2 hour: questions and discussion

A word about cathedrals

- Ansible and ServerSpec are fantastic tools, and once you get into using them a bit, you will have grand plans on how to use them more
- You will have those ideas today
- We will not build any cathedrals today
- We may show you hints at the ways some cathedrals have been built, however…

We will not build cathedrals today

We are constantly learning more about our environment

- Developers shop jobs a lot
- Have you seen the mailing lists?
- We change jobs a lot
- We are always the newbie

Why Write ServerSpec Tests?

- Tests are documentation
- Your work group may or may not survive
- Provisioning tools come and go
- No matter what happens to the tools or your workforce, these tests will persist as documentation of your intentions and proof that the service is configured as you expected

ServerSpec

- Extension of RSpec
- Yep, it's a Ruby gem
- Is a great way to force yourself to think about your intentions before you provision a new service
- Is a great way to get to know your existing services

Installing ServerSpec

- It's a Ruby gem, you'll need Ruby 1.9.x+ installed
- gem install serverspec
- you'll also need SSH access to the servers you want to check
- For Ansible you'll probably need Sudo privileges on these servers

Start simple

```
require 'spec_helper'
describe package('httpd') do
 it { should be_installed }
end
describe service('httpd') do
 it { should be_enabled   }
 it { should be_running   }
end
describe port(80) do
 it { should be_listening }
end
```

Yes, there is an init script

```
$ serverspec-init
Select OS type:
 1) UN*X
 2) Windows
Select number: 1
Select a backend type:
 1) SSH
 2) Exec (local)
Select number: 1
Vagrant instance y/n: n
Input target host name: www.example.jp
+ spec/
+ spec/www.example.jp/
+ spec/www.example.jp/sample_spec.rb
+ spec/spec_helper.rb
+ Rakefile
+ .rspec
```

Run the tests

```
$ rake spec
/usr/bin/ruby -S rspec
spec/www.example.jp/sample_spec.rb
Package "httpd"
 should be installed
Service "httpd"
 should be enabled
 should be running
Port "80"
 should be listening
Finished in 0.21091 seconds (files took 6.37
seconds to load)
4 examples, 0 failures
```

What's next?

- Start simple, with the services you already know
- Consider writing tests before you deploy a new service
- As you use these tools, look for ways to consolidate your effort

Time for some Pair Programming!

- Form two lines:
  - Ever used Vagrant? You go on the left side.
  - New to Vagrant? You go on the right side.
  - Whomever is at the head of each line, you are now a team, you will work together for the rest of this workshop.
  - Keep forming teams until lines are empty.

Getting Started with Ansible

- Based on Ansible for Hydra:

  https://tinyurl.com/DCE-Anisble4Hydra

- Slides will funcion as bookmarks, but do follow along with the wiki, it will be easier to copy/paste code samples from there.

Why Ansible?

- **Agentless**: no “puppetmaster,” no “client”
- **Simple and legible**
- **Sequentially executed**: devs will understand it
- **Well documented**: docs.ansible.com
- **Well integrated**: works with Vagrant, and anything you can SSH to
- **Well extended**: database and utility connections are in Ansible core, not add-ons or plugins

A Simple Ansible Playbook

- Playbooks are YAML files
- Playbooks invoke modules
  - package
  - service

```
---
- name: simple playbook
  hosts: web

  tasks:
    - name: restart web server
      become: yes
      service: name=apache2 state=restarted
```

Run the Simple Playbook

```
$ ansible-playbook simple_playbook.yml --user=myuser --private-
key=~/.ssh/my_key.pem -i server1.domain.ext

PLAY [simple playbook]
**********************************************************************

TASK [setup]
**********************************************************************
ok: [server1.domain.ext]

TASK [restart web server]
**********************************************************************
changed: [server1.domain.ext]

PLAY RECAP
**********************************************************************
server1.domain.ext : ok=2    changed=1    unreachable=0    failed=0
```

Oh no! Apache2 isn't installed!

```
---
- name: simple playbook
  hosts: web

  tasks:
    - name: install & maintain web server
      become: yes
      package: name=apache2 state=latest

  - name: restart web server
    become: yes
    service: name=apache2 state=restarted
```

Wait! only restart Apache2 if it changes

- Notify a Handler

```
---
- name: simple playbook
  hosts: web

  tasks:
    - name: install & maintain web server
      become: yes
      package: name=apache2 state=latest
      notify:
        - restart web server

  handlers:
    - name: restart web server
      become: yes
      service: name=apache2 state=restarted
```

More Complex Playbooks

Use more modules!

- package
- file
- template
- cron
- postgresql_user
- copy
- shell

and {{variables}} ! (not a module, just a feature)

More Complex Playbooks: The Danger Zone

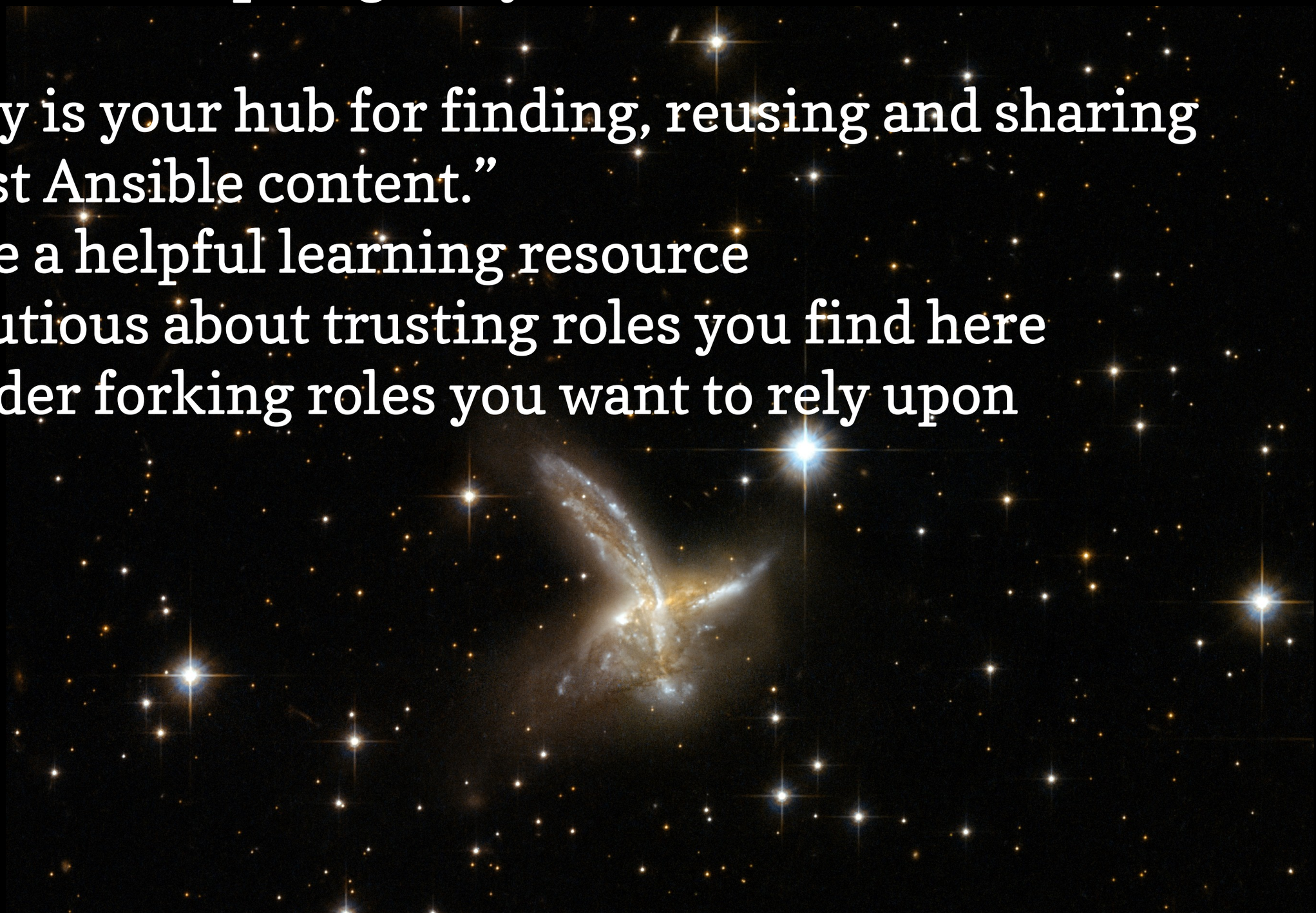
Ansible Roles

“Roles in Ansible build on the idea of include files and combine them to form clean, reusable abstractions – they allow you to focus more on the big picture and only dive down into the details when needed.” – Ansible Docs

- reusable
- abstraction
- keep your focus on the big picture

Ansible Galaxy

https://galaxy.ansible.com/



"Galaxy is your hub for finding, reusing and sharing the best Ansible content."

- can be a helpful learning resource
- be cautious about trusting roles you find here
- consider forking roles you want to rely upon

Multiple Hosts: Using an Inventory

Ansible Testing

- As you create more roles, you'll want to keep an eye on them to ensure they are still usable
- A simple syntax check in a .travis.yml file (using Travis CI) is a great place to start
- UCLA Library has an example role template you are welcome to borrow, that has this strategy built in:

github.com/UCLALibrary/uclalib_role_template

Combining Ansible and ServerSpec