

Getting Started with Ansible & ServerSpec

Open Repositories 2017

Brisbane, Australia

June 27, 2017

Half-day workshop, afternoon

tinyurl.com/OR17-AnsibleServerSpec



This work is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.

Getting Started with Ansible & ServerSpec

facilitated by

Hardy Pottinger, UCLA Library

additional collaborators

Alicia Cozine, Data Curation Experts

Anthony Vuong, UCLA Library

Francis Kayiwa, Princeton University Library

tinyurl.com/OR17-AnsibleServerSpec

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 2.0.x+ installed
`sudo gem install serverspec`
- You will need Rake installed, too:
`sudo gem install rake`
- 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.

Vagrant Up!

- The USB key has a Vagrant workspace and Virtualbox machine image in the folder named **HCTraining_start**
- Copy that folder to your notebook's hard drive
- -or-
- CD to that folder at its mount point, note down the path for later, e.g.
`/media/username/ANSIBLE`
- From your command prompt, run **vagrant up**

lost your USB stick? No worries, just use
<https://github.com/AnsibleWorkshops-R-Us/workshop-o-matic>

Vagrant ssh

- When the vagrant up command completes successfully, run `vagrant ssh`
- We will now ensure everyone is able to complete this step, work together in teams, it's OK to help other teams if necessary
- We are all in this together

TDD time: let's run the ServerSpec 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: 2
+ spec/
+ spec/localhost/
+ spec/localhost/sample_spec.rb
+ spec/spec_helper.rb
+ .rspec
```

TDD time: watch the tests fail

```
$ rake spec
```

```
...  
Port "80"  
  should be listening (FAILED - 1)
```

Failures:

```
1) Port "80" should be listening  
   On host `localhost`  
   Failure/Error: it { should be_listening  
 }  
...  
...
```

TDD time: watch the tests fail

...

```
Finished in 0.00616 seconds (files took  
0.40627 seconds to load)  
1 example, 1 failure
```

Failed examples:

```
rspec ./spec/localhost/sample_spec.rb:27 #  
Port "80" should be listening
```

Cool, let's fix this!

Getting Started with Ansible

- Based on Ansible for Hydra:

<https://tinyurl.com/DCE-Anisble4Hydra>

- Slides will function 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

The Simplest Ansible Playbook

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

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

Run the Simplest Playbook

```
$ ansible-playbook simplest_playbook.yml -i hosts
```

```
PLAY [simplest playbook] TODO: paste current output here
*****
```

```
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: simpler 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

```
$ ansible-playbook simpler_playbook.yml -i hosts
```


Try running simpler_ twice

```
$ ansible-playbook simpler_playbook.yml -i hosts
```

```
PLAY [simple playbook]
```

```
*****  
*****
```

```
TASK [Gathering Facts]
```

```
*****  
*****
```

```
ok: [controller]
```

```
TASK [install & maintain web server]
```

```
*****  
*****
```

```
ok: [controller]
```

```
TASK [restart web server]
```

```
*****  
*****
```

```
changed: [controller]
```

```
...
```

Try running simpler_ twice

PLAY RECAP

```
*****
*****
controller          : ok=3    changed=1    unreachable=0
failed=0
```

... wait ... what?

TASK [restart web server]

```
*****
*****
changed: [controller]
```

... every time we run this playbook, Apache will restart!

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
```

```
$ ansible-playbook simple_playbook.yml -i hosts
```

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

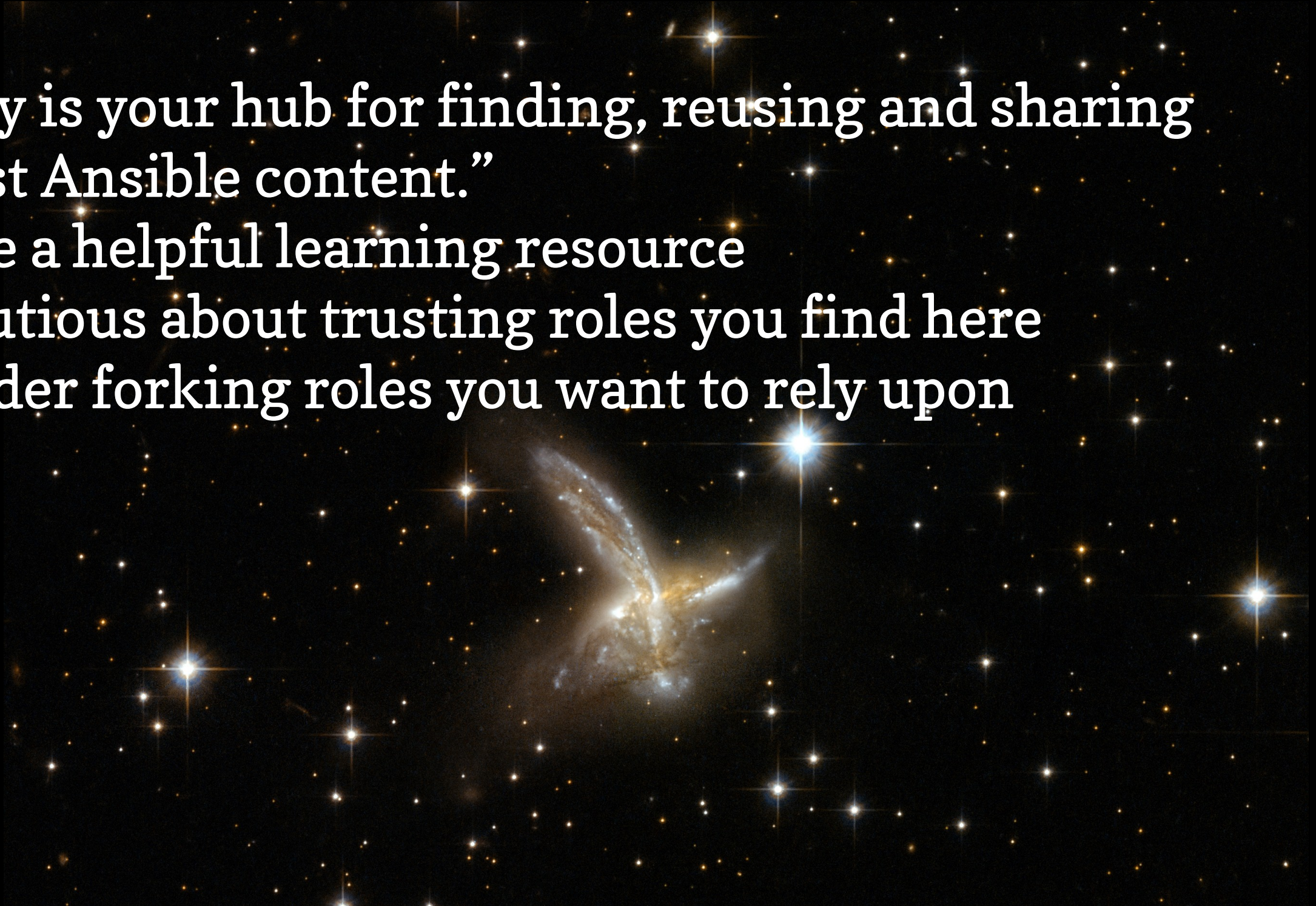
- reusable
- abstraction
- keep your focus on the big picture

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 can run against many hosts
- Inventory files help specify which ones
- The default inventory lives at `/etc/ansible/hosts`
- You can specify a hosts file with the `-i <path>` option on the command line

Multiple Hosts: Using an Inventory

- Inventories are in INI format
- You can group servers together in [groups]

```
mail.example.com
```

```
[webservers]
```

```
foo.example.com
```

```
bar.example.com
```

```
[dbservers]
```

```
one.example.com
```

```
two.example.com
```

```
three.example.com
```

Multiple Hosts: Using an Inventory

- There are lots of options for inventory management (dynamic, cloud-based)
- Start simple, save this for later, after you have some playbooks of your own
- Here's the one we've been using this whole time:

[web]

```
controller ansible_connection=local
```

Ansible Testing

- As you create 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

Ansible Testing

- Speaking of syntax checks, you have one available to you already

```
$ ansible-playbook simpler_playbook.yml -i hosts --syntax-check
```

```
playbook: simpler_playbook.yml
```

TDD time: watch the tests pass

```
$ rake spec
```

```
...
```

```
Package "apache2"  
  should be installed
```

```
Service "apache2"  
  should be enabled  
  should be running
```

```
Port "80"  
  should be listening
```

```
Finished in 0.05215 seconds (files took 0.28294 seconds to  
load)
```

```
4 examples, 0 failures
```

Combining Ansible and ServerSpec

Save this for later

tinyurl.com/OR17-AnsibleServerSpecMashup

Ansible Best Practices

- ansible.com/blog/ansible-best-practices-essentials
- “Name” your plays and tasks
- Use prefixes and human meaningful names with variables
- Use native YAML syntax
 - key=value pairs are supported but they are hard for humans to read
- Use modules before run commands
- If a playbook is starting to get complicated, you need a role

Ansible Best Practices

- Be careful with the flexibility Ansible gives you
 - Variables can be in many places
 - vars files
 - vars directories
 - hosts file
 - include files
 - command line
 - ENViroment variables
 - The order of precence will sneak up on you:
 - Too many options to fit on a slide
 - Look it up in the docs
 - Do not set a “default” in a vars file

Ansible Best Practices

- Put “become” (sudo) at the task level rather than the include or role level
- Minimize handler use (to keep sequence obvious)
- Make main.yml just for includes, so tasks are in usefully-named files (too many main.yml files!)

Ansible Inventory

Best Practices

- Unless you have a great reason not to, stick to a single hosts file
- Use groups
 - Group vars are really handy
- Be careful when running a playbook on a new host, it's very easy to run a playbook on more hosts than you intend
- Use the `-l` option to specifically limit playbook runs to just the host you intend
- `--syntax-check` is easy to add and will save you grief

Questions? Discussion?

tinyurl.com/OR17-AnsibleServerSpec

Hardy Pottinger
hpottinger@library.ucla.edu
[@hardypottinger](#)

Images

Sagrada Familia by pierpeter
<https://www.flickr.com/photos/vespeter/228136258>

Charlie Chaplin, Eating Machine
Still image from the 1936 movie *Modern Times*
<http://www.imdb.com/title/tt0027977>

Hubble Interacting Galaxy ESO 593-8
[https://commons.wikimedia.org/wiki/File:Hubble_Interacting_Galaxy_ESO_593-8_\(2008-04-24\).jpg](https://commons.wikimedia.org/wiki/File:Hubble_Interacting_Galaxy_ESO_593-8_(2008-04-24).jpg)