Network Automation: Ansible 101

APRICOT - Feb 28th, 2017

Bronwyn Lewis and Matt Peterson

Our assumptions

- ➔ New to the world of “DevOps”
- ➔ No prior Ansible knowledge
- ➔ Want to stop hand-crafting your network configs

Agenda

Introduction

- ➔ Tutorial dependencies
- ➔ Introductions
- ➔ DevOps intro

Tutorial

- ➔ Ansible intro & concepts
- ➔ Configuration templating
- ➔ Homework, next steps

Tutorial repository

https://git.io/vZKZH

Required knowledge

1. basic familiarity with the command line
2. use of a command line text editor (e.g. vim or nano)

http://git.io/vZKZH

Technical requirements

Required

- ➔ Linux, MacOS, or Win10
- ➔ Python 2.7
- ➔ Ansible 2.2

Recommendations

- ➔ Ubuntu 16.04
- ➔ VM (VirtualBox, Vagrant)

http://git.io/vZKZH

Introductions

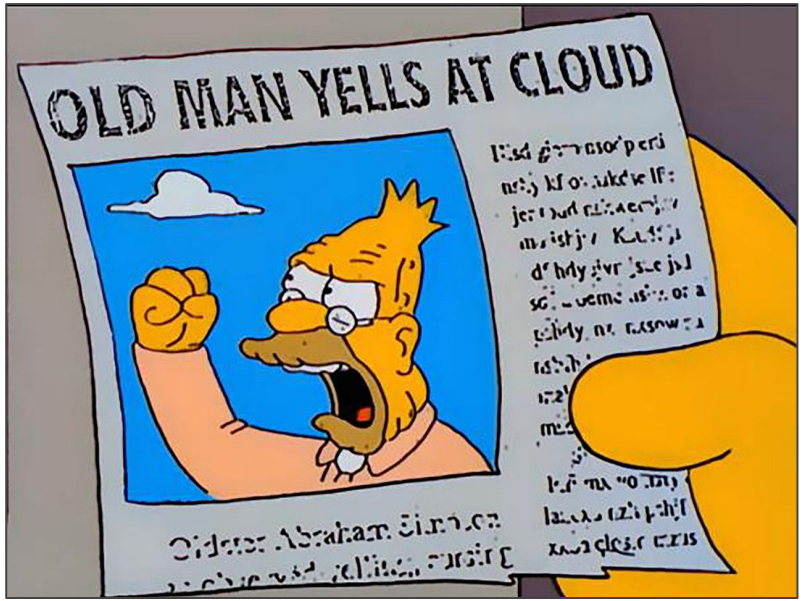
whois Bronwyn Lewis

- Technical advisor at SFMIX
- Networking, systems, and automation @ SFMIX and PCH for 3+ years
- Background in operations, project management, & international affairs

whois Matt Peterson

- Principal at Two P (network / systems)
- President at SFMIX (San Francisco IXP)
- Previously: Cumulus Networks, Tumblr, Square, Burning Man

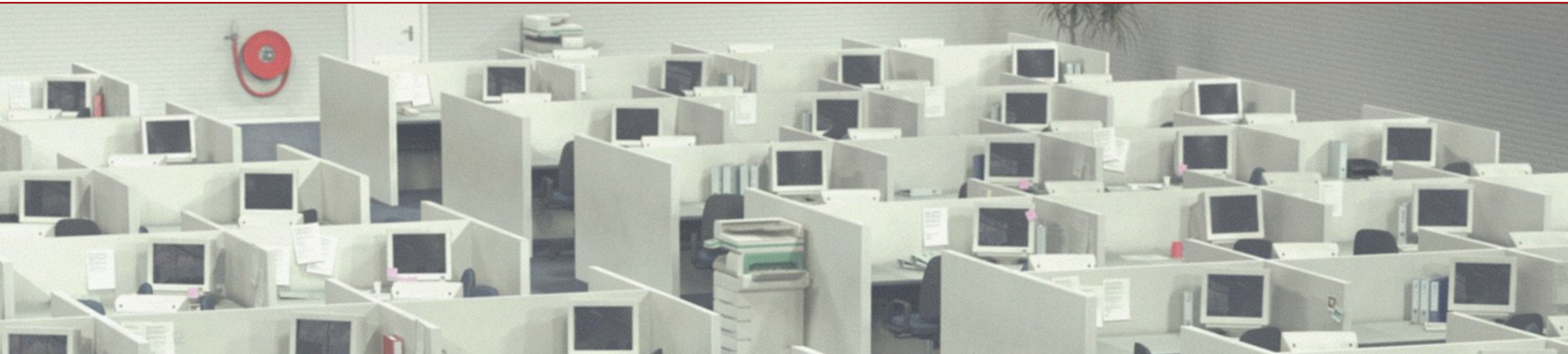
Got {Net}DevOps?



DevOps

- Unite people and *{organization appropriate}* methods
  - Typically Developers & Operations staff
  - Shared service(s) availability responsibility

- Not a specific software program, license, certification



{Net}DevOps

Leverage common DevOps tenants within Networking

- Configuration management (today's focus)
- Infrastructure as code
- Reactive to infrastructure as a whole
- Consistency *(sometimes viewed as transparency)*

This is *not* a DevOps talk

- DevOps Kung Fu
  https://github.com/chef/devops-kungfu
- Phoenix Project / IT Revolution
  http://itrevolution.com/
- DevOps Cafe podcast
  http://devopscafe.org/

Automation Tools

```
while true ; do cat ~/.history ; done
```

Automation tools aren’t new

- Expect (1990)
- CFEngine (1993)
- Puppet (2005)
- NETCONF (2006)
- OpenConfig (2014)
- Lots of homegrown tools

And much, much more...

Today’s frameworks

What’s great about frameworks?

Technical Benefits

- procedural
- repeatable
- idempotent

Other Benefits

- open source (majority)
- enterprise support
- community

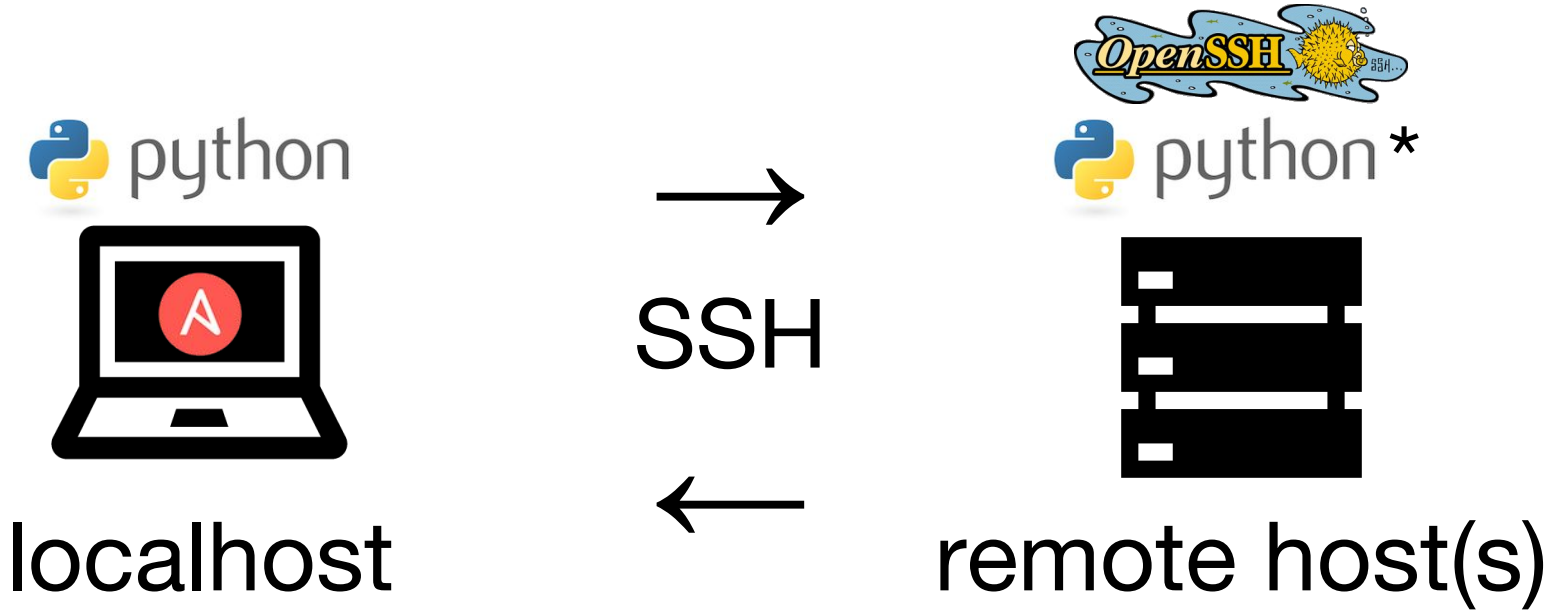
Why Ansible?

1. agent’less
2. low risk (run it locally)
3. small investment
4. easy to learn (abstraction!)

Abstraction

```
instructions:
    what: update pkgs
   where: myServer1, myServer5
    when: 23.00UTC

reference:
    pkgs: openssh, apache
```

How Ansible works



*default assumption, unless module exists for target host OS

(But we’re running it locally.)

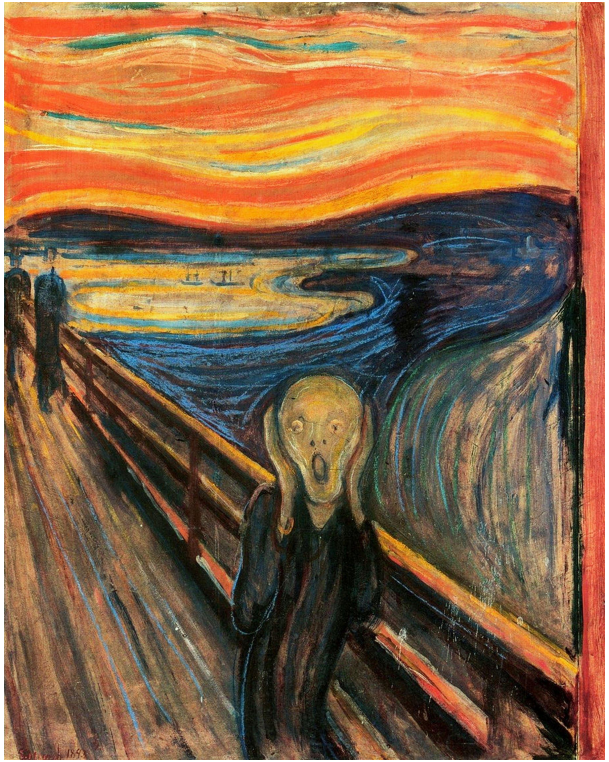
localhost

Terminology

WARNING!

Visually boring, but *important* information packed slides ahead.

(Sorry.)



JSON

- Data exchange format
- More powerful than CSV
  - Data can imply it’s a list, integer, string, etc.

```
{
  "roles": {
    "noc": {
      "name": "Alice"
    },
    "dev": {
      "name": "Ian"
    }
  }
}
```

YAML

- Human readable data format, subset of JSON
- Always starts with `---`
- Filename extension `.yml` or `.yaml`

```
# EXAMPLE DATA FILE 1
---
roles:
 - { who: dev, name: Ian }
 - { who: noc, name: Alice }

# EXAMPLE DATA FILE 2
---
roles:
  noc:
    name: Alice
  dev:
    name: Ian
```

Jinja2

```
# EXAMPLE TEMPLATE

Employees:
{% for k,v in roles %}
  Role: {% k %}
  Name: {% v %}
{% endfor %}
```

- Python template engine
- Enumerates files using variable data
- Supports conditionals:
  - If statements
  - Loops
  - Piping
- Ansible standard file extension `.j2`

Hosts

- Group host addresses, assign names, specify variables, etc.
- Default is /etc/ansible/hosts
  - can override this easily

```
# EXAMPLE HOSTS LIST

[dev]
test-switch1 mgmt_ip=10.1.10.1
100.0.0.42
dev-router4

[prod]
mywebsite.com
172.16.0.56 name=dev42.prod
172.16.0.17
```

Playbooks

```
---
- name: Generate configs
  hosts: localhost
  gather_facts: no

  roles:
   - router
   - switch
```

- Specifies execution
- Single or multiple OK
- You *can* write all tasks and vars *in* a playbook...
  - … but not recommended

Facts

- Gathers information on the remote host(s)
  - Hardware, OS, uptime, MAC address & more
- You can use this info like a regular variable data point

```
# EXAMPLE SYSTEM FACTS

"ansible_architecture":
"x86_64",
"ansible_bios_date":
"09/20/2012",
"ansible_bios_version":
"6.00",
```

Inventory

```
[EXAMPLE STRUCTURE]

myplaybook.yml
roles
inventory
  hosts
  group_vars
    sites.yml
```

- Allows you to pass in specific data with different playbooks
- Can specify hosts, group vars, and host-specific vars
- Can be accessed across multiple roles

Roles

- A built-in structure for compartmentalizing
- Roles make it easy / clean to manage execution
- Makes scaling and collaboration easier!

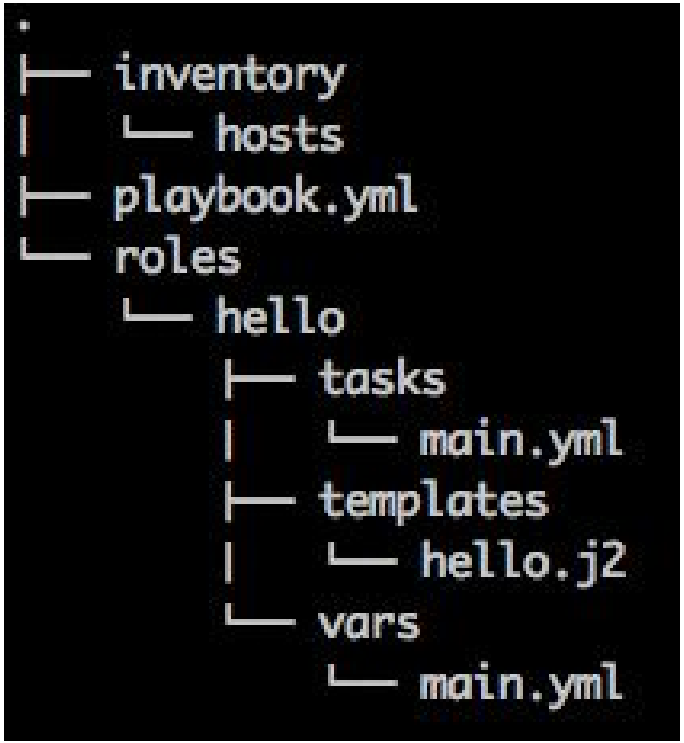
```
[EXAMPLE STRUCTURE]

ansible
  myplaybook.yml
  roles
    router
      tasks
      templates
    switch
      tasks
```

Hands-on

General outline

- Inventory + Roles
- Variables
- Templates
  - IP Address Filter
- Tasks
- Hosts
- Playbook

Hello world

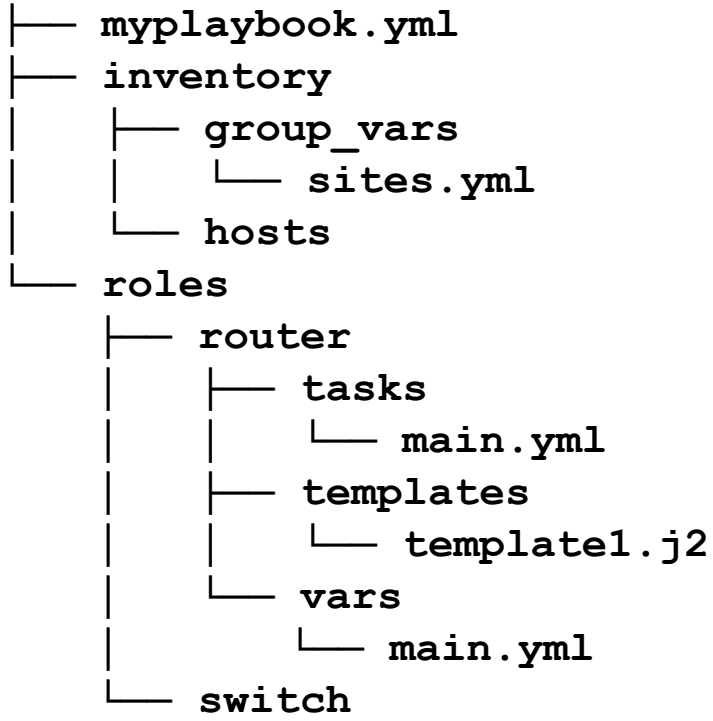


Hello world (before)

```
➜ hello_world git:(master) ansible-playbook -i inventory playbook.yml

PLAY [Hello world] ************************************************************

TASK: [hello | Verify compiled directory exists] *******************************
changed: [localhost]

TASK: [hello | Generate "hello"] **********************************************
changed: [localhost] => (item={'name': 'world', 'number': 1})
changed: [localhost] => (item={'name': 'RIPE71', 'number': 2})

PLAY RECAP ********************************************************************
localhost                  : ok=2    changed=2    unreachable=0    failed=0
```

Hello world (after)

Structure



- Lots of ways to structure
  - Use roles?
  - Use an inventory?
  - Global, group, host variables?
- Depends on your situation
- No “right” way

Reference files

Copy these from `workspace/reference/`

config1: we'll use this as our 1st template
config2: we'll use this as our 2nd template
config1-dhcp: advanced example template
config2-dhcp: advanced example template
ipaddress: RFC 5737 IP addresses (for demo/docs)
variables: we'll use these as our demo vars

Inventory + roles

- Inventory is an easy way to share variables across roles, as well as managing hosts & host-specific variables
- Roles make managing multiple templates and sets of tasks easier by compartmentalizing them

$ command-line

Variables

- Variables can be formatted individually, as a flat list, as a dictionary, or as an array
- Specific formatting can vary

⚠ Formatting impacts how you pass variables into templates and tasks — be careful here! ⚠

Templates

- You can template anything!
- Lots of neat advanced features, including:
  - If, when, and for statements/loops
  - Variable manipulation via filters

$ command-line

Tasks

- Procedural list of actions to execute, which combines templates and vars
- Conditions can be included, and are based on vars (i.e., only do X when Y is present)

$ command-line

IP address filter

- The `ipaddr()` filter is included in Ansible 1.9<
- Provides an interface to the `netaddr` Python package; does a lot of neat things including:
  - subnet manipulation
  - address validation
  - address conversion
  - MAC address formatting

Hosts

- What host we should be running the tasks on - normally this would be a remote host, but for us:

`localhost`

$ command-line

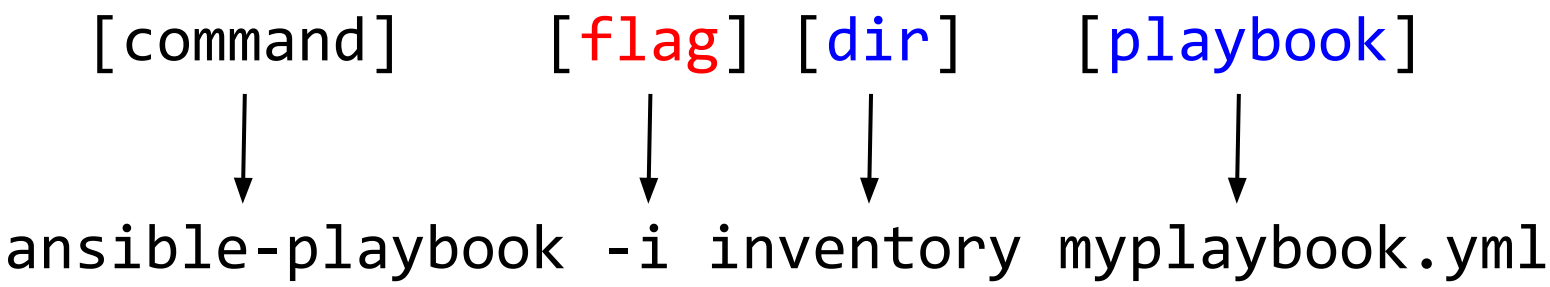
Playbook

- Brings it together:
  - Hosts
  - Roles
    - Tasks
    - Templates
  - Variables
- And executes!

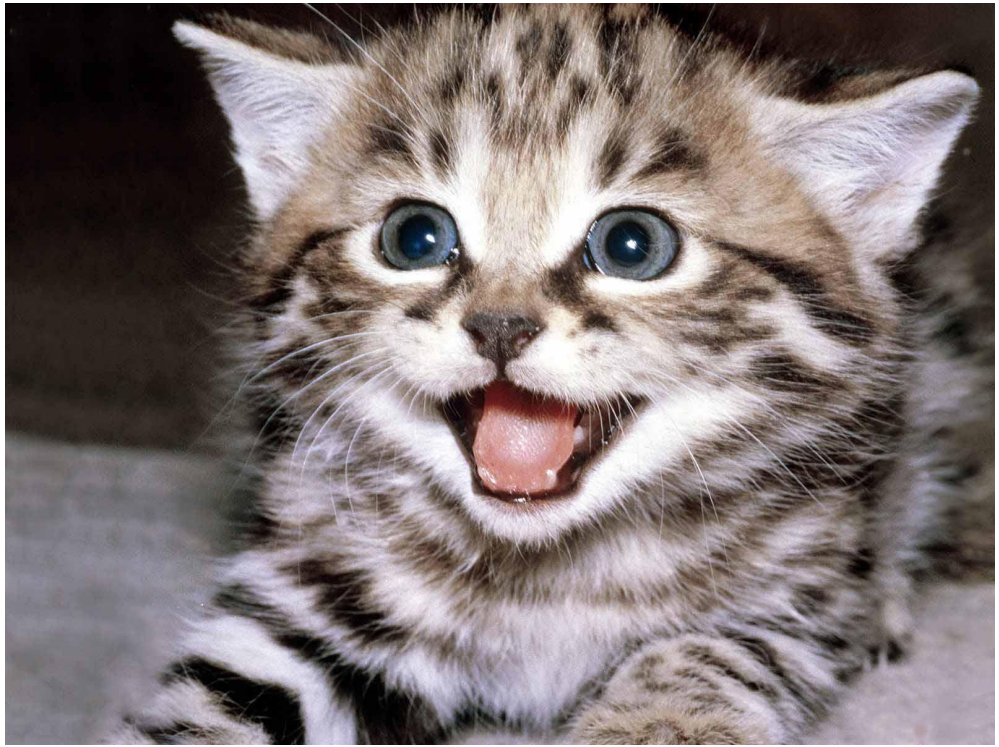
```
---
- name: Create files
  hosts: localhost
  connection: local
  gather_facts: no

  roles:
   - router
```

$ command-line

Running a play



You’ve got configs!



And if it didn’t work...

Common issues:

- Missing packages?
- Missing variables?
- Formatting weirdness?
- Typos?

Ansible can provide clues.

Ansible Debugging 101

Common Ansible debugging issues include:

One or more undefined variables: 'dict object' has no attribute 'hostname'

One or more undefined variables: 'hostname' is undefined

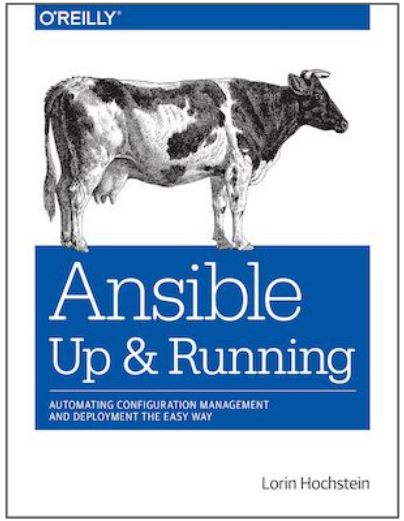
ERROR: Syntax Error while loading YAML script

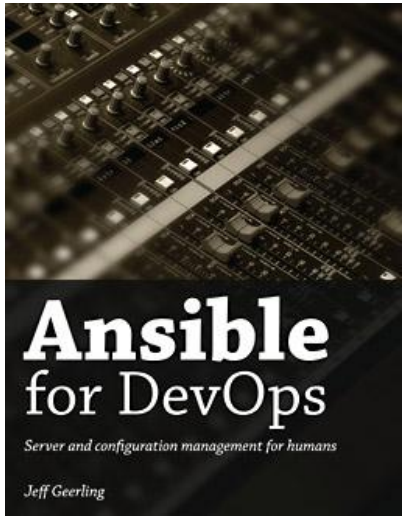
So… what’s next?

- Think how you can apply this to your work
- Start small; doesn’t need to be overly complex
- Check out more resources...

Some resources

books





blogs/sites

- http://jedelman.com/
- https://blog.tylerc.me/
- https://pynet.twb-tech.com/
- http://packetpushers.net/
- http://keepingitclassless.net/
- http://ansible-tips-and-tricks.rtfd.org/

stackoverflow

… and more!

Join us for 102 after the break...

- Advanced templating techniques
- Inventory + advanced variable & hosts management
- Dynamic inventory
- And more!

Thanks!

1. Questions? Comments?
2. Come talk to us!
3. Email or tweet us

me@bronwynlewis.com @bronwyn
matt@peterson.org @dorkmatt