Network Automation: Ansible 101

RIPE 71 - November 16th, 2015

Bronwyn Lewis and Matt Peterson

Our assumptions

- ➔ New to the world of “DevOps”
- ➔ No prior Ansible knowledge

Not within scope:

- ➔ Understand all encompassing DevOps fanaticism
- ➔ Automate yourself out of job

Agenda

Introduction

- ➔ Tutorial dependencies
- ➔ Introductions
- ➔ DevOps intro

Tutorial

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

Tutorial prerequisites

http://git.io/vZKH

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

Environment

Officially supported!

Not supported, but can work (Cygwin)

http://git.io/vZKZH

Virtual environment

http://git.io/vZKZH

Packages

Essentially…

http://git.io/vZKZH

What if I don't have any of that...

Use the command line?

Use a text editor?

You can still do most of this.

Introductions

`whois` Bronwyn Lewis

Packet Clearing House

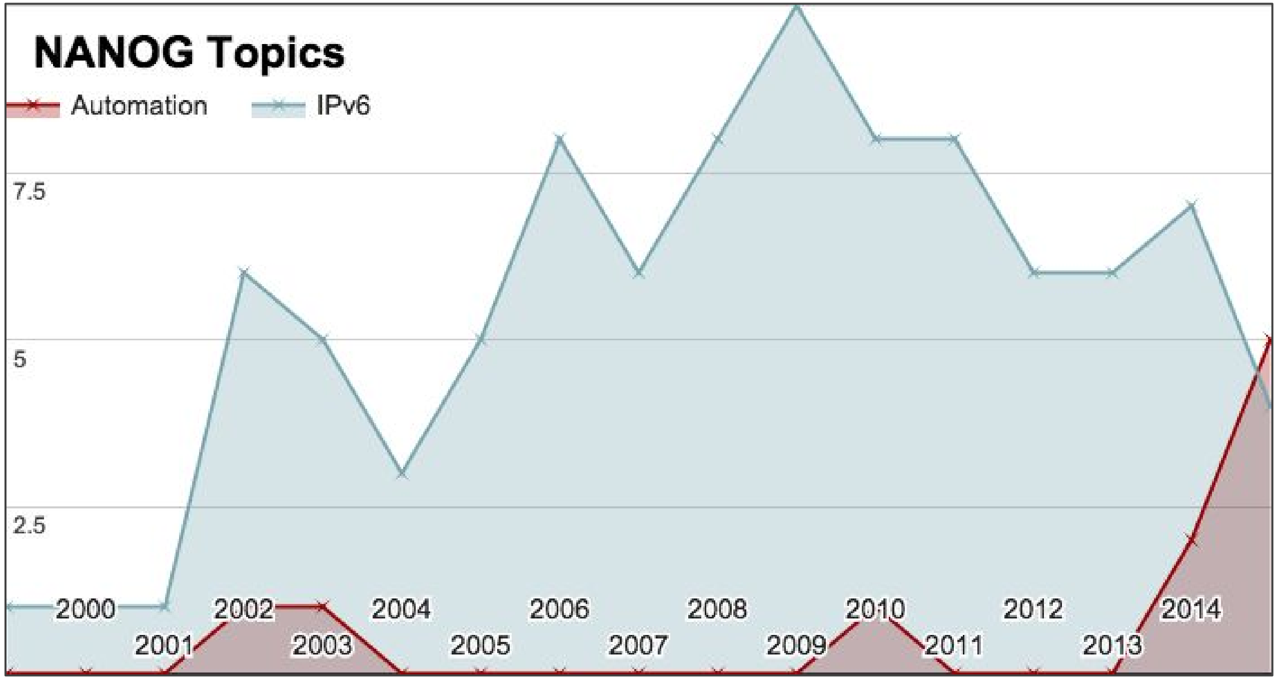
- Engineer @ Packet Clearing House
- Wearer of many hats (provisioning, network, systems, automation)
- Background in operations, project management, & international affairs

whois Matt Peterson

- Office of the CTO @ Cumulus Networks
- Network & systems engineer / architect for 15+ years
- Held enable @ Square, Tumblr, Burning Man, SFMIX

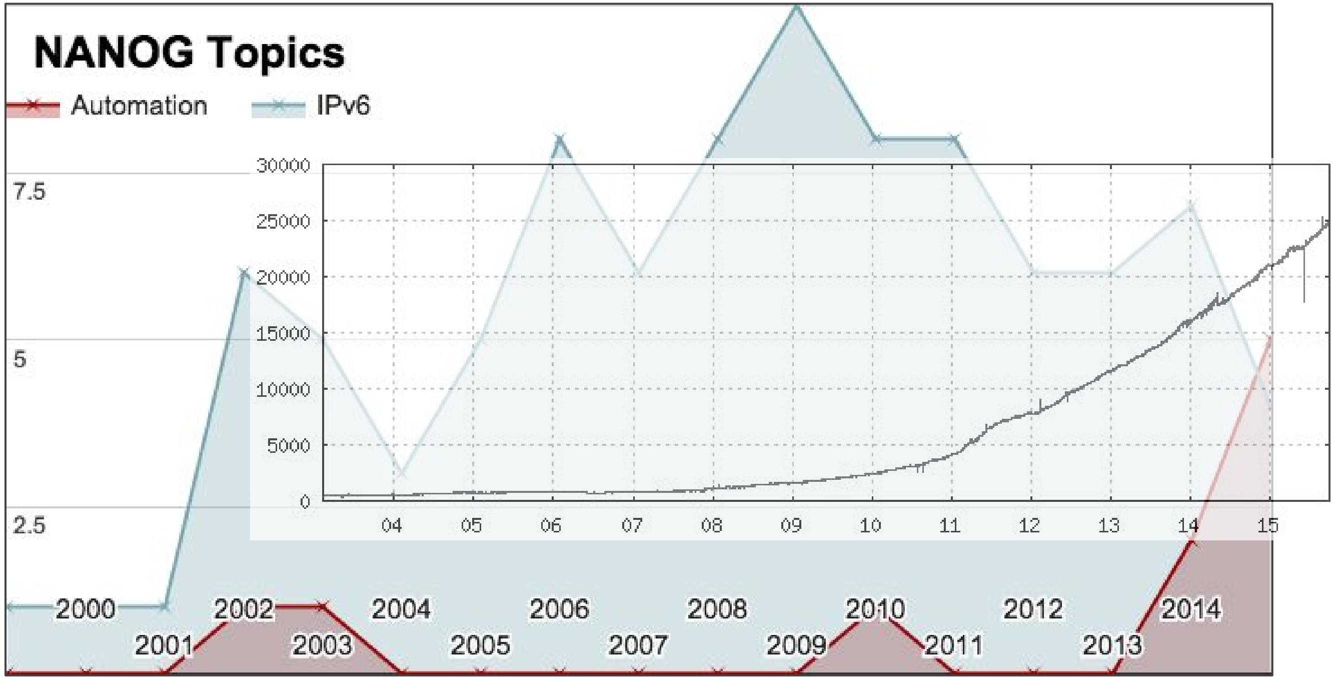
Why network automation?

Trending data



source: http://nanog.org/archives/presentations … query “ipv6”

Trending data

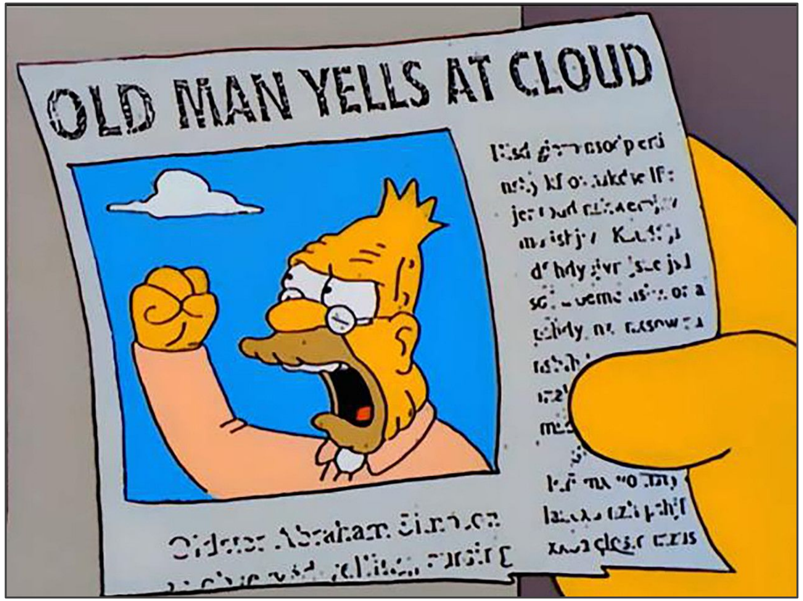


source: http://nanog.org/archives/presentations … query “ipv6” and http://bgp.potaroo.net/

Observations

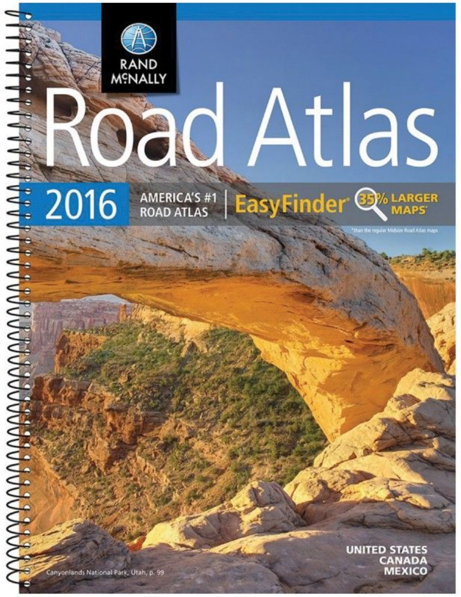
	International ISP (tier 2)	International ISP (tier 1)	National MSO
Source of truth	Database	Database, CSV	Many, reconciliation ...
Customer turn-up	Manual	Automated	Automated
Infrastructure	Route maps + lots hand tuning	Automated	Automated
Peering turn-up	Scripts for new, lots of hand tuning	Automated	Automated
SWIP / rwhois	Triggered script	Automated	Automated
DNS records	Automated nightly script	Automated	Automated
Monitoring	Scripts, based on importance	Vendor & home grown	Vendor & home grown

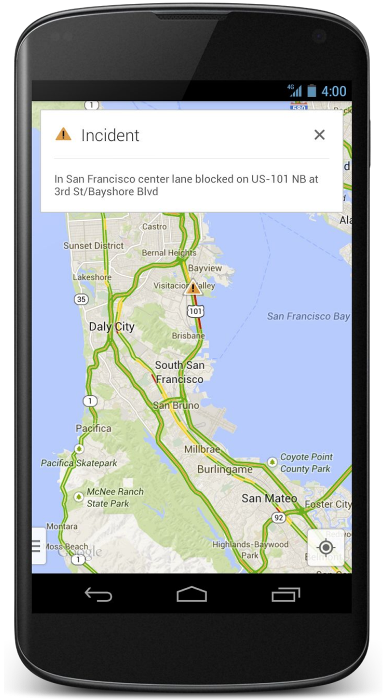
Got {Net}DevOps?



An industry in transition

Or… is your job in jeopardy?

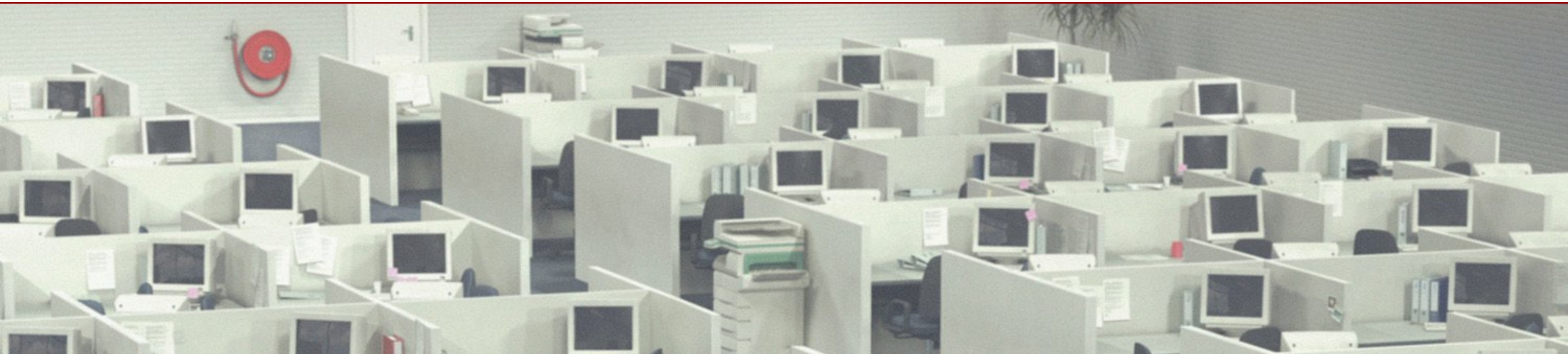




DevOps

- 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)*

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 frameworks aren't new

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

And much, much more...

History repeating itself

- *Pigeonhole pack* - Tcl, SLAX, NETCONF/Yang
  - Limited development community *(excl. prof. services)*
  - NOS specific implementation and/or niche language

- CLI (scraping) & SNMP still the norm... *Why?*
  - Approachable from many scripting languages
  - An “API” where faults are well understood

Lots of compute options

puppet labs®

SALTSTACK

CFEngine

CHEF™

ANSIBLE

… or write your own?

What's great about today's {compute automation} tools?

Technical Benefits

- procedural
- repeatable
- idempotent

Other Benefits

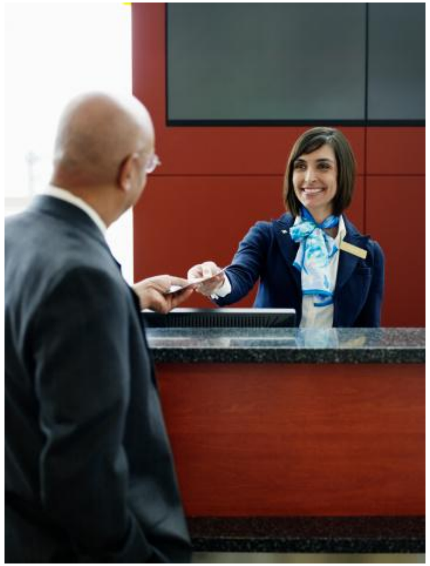
- open source (majority)
- enterprise support
- community

Abstraction

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

reference:
    pkgs: openssh, apache
```

Signalling



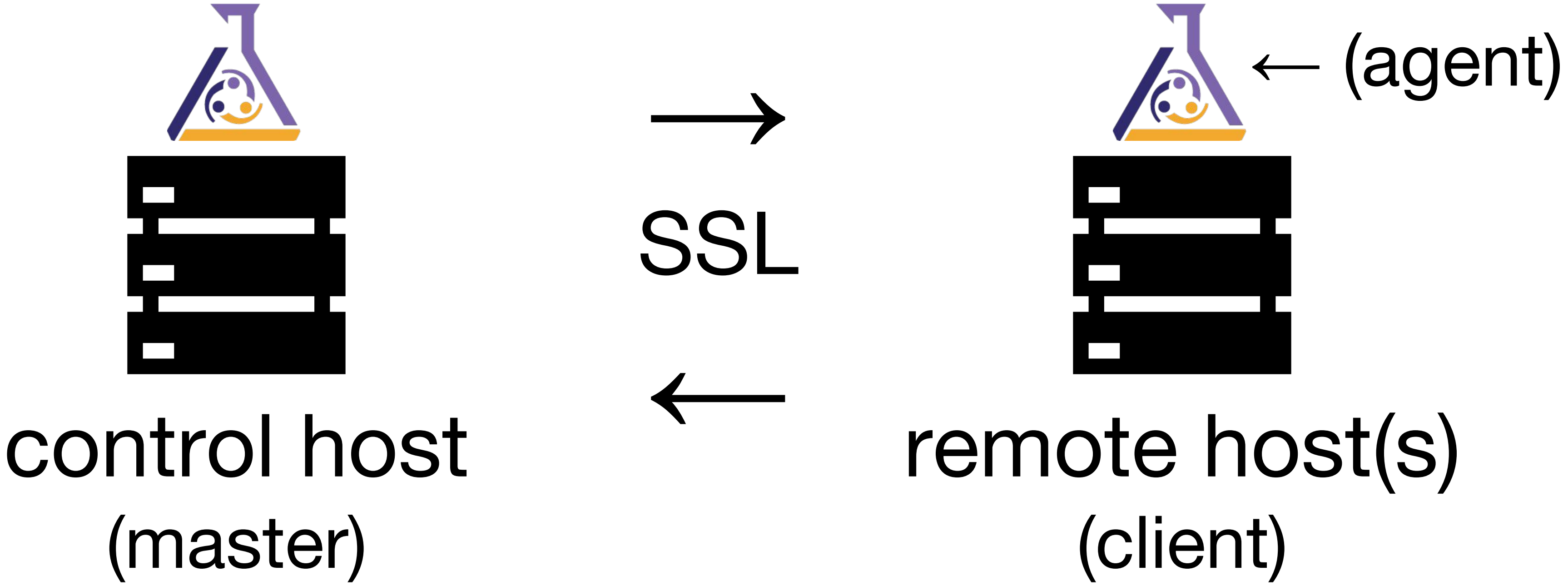
agent

software installed on remote hardware to interface with

agent'less

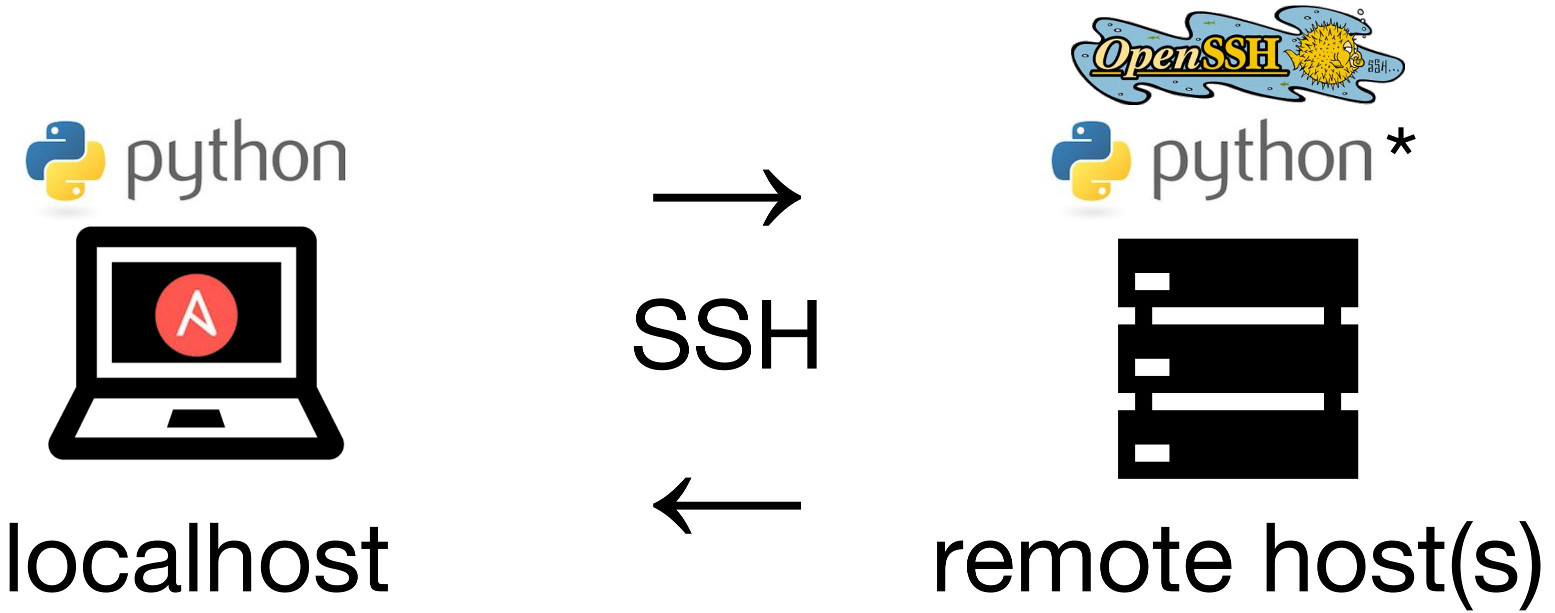
no specific software installed on remote hardware

How Puppet* works



*just one example

How Ansible works



*default assumption, unless module exists for target host OS

(But we’re running it locally.)

python

localhost

So, why did we pick Ansible?

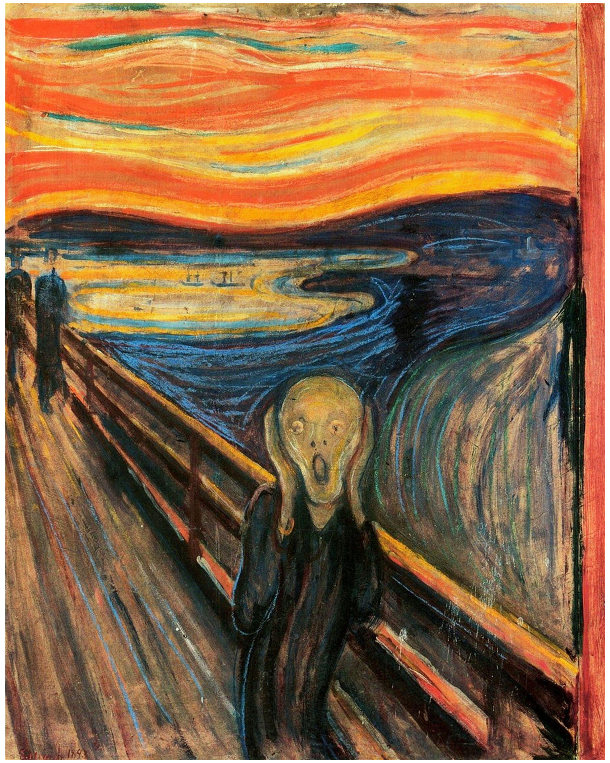
1. agent'less
2. low risk (run it locally)
3. small investment
4. easy to learn

Terminology

WARNING!

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

(Sorry.)



YAML

- Human readable data format / alternative to XML
- More powerful than CSV
    - Data can imply it's a list, integer, string, etc.
- Filename extension `.yml`

```
# 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 a,b in roles }
  Role: { item.a }
  Name: { item.b }
{ 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 DIRECTORY/FILE
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 DIRECTORY/FILE
STRUCTURE]

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

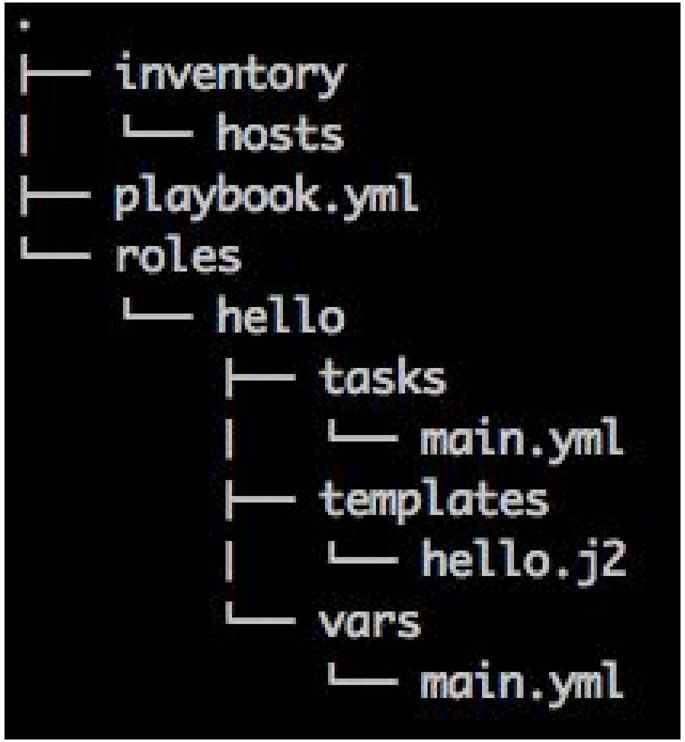
Hands-on: config generation

General outline

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

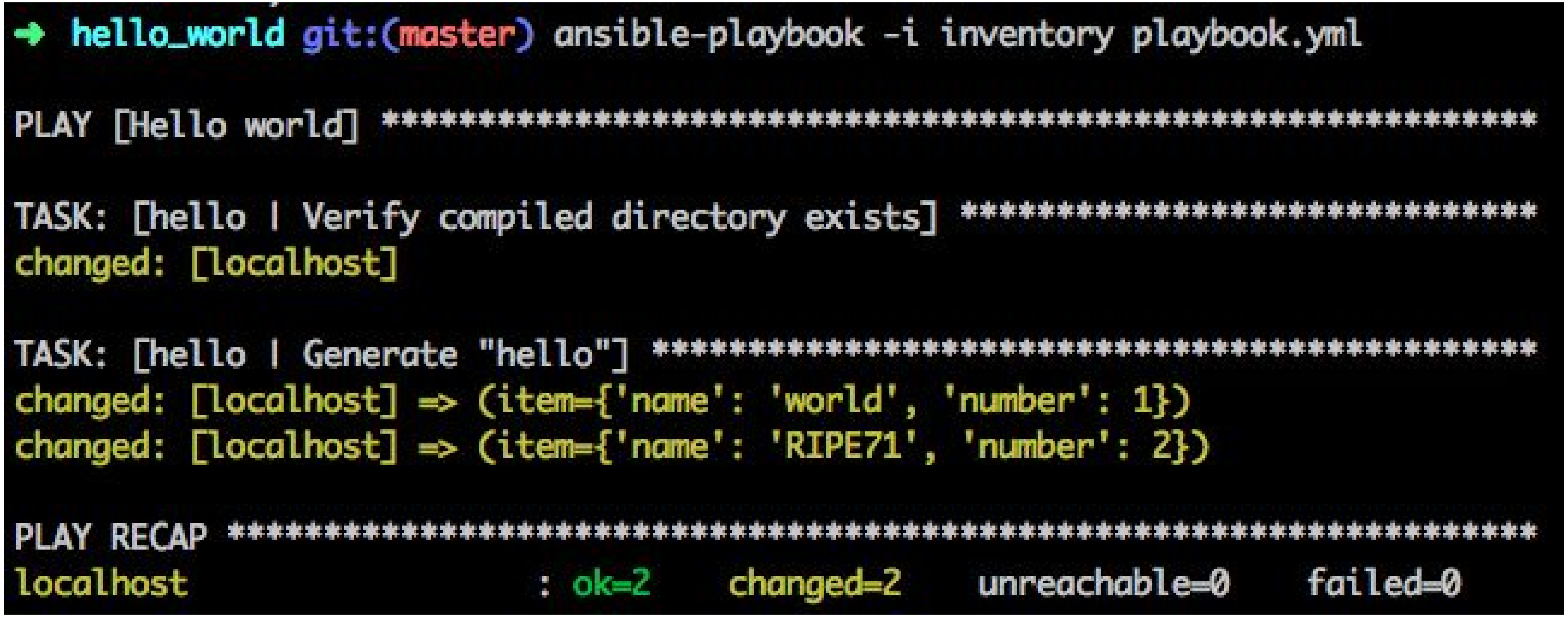
```
$ command-line
```

```
Hello world
```



```
.
├── inventory
│   └── hosts
├── playbook.yml
└── roles
    └── hello
        ├── tasks
        │   └── main.yml
        ├── templates
        │   └── hello.j2
        └── vars
            └── main.yml
```

Hello world (before)



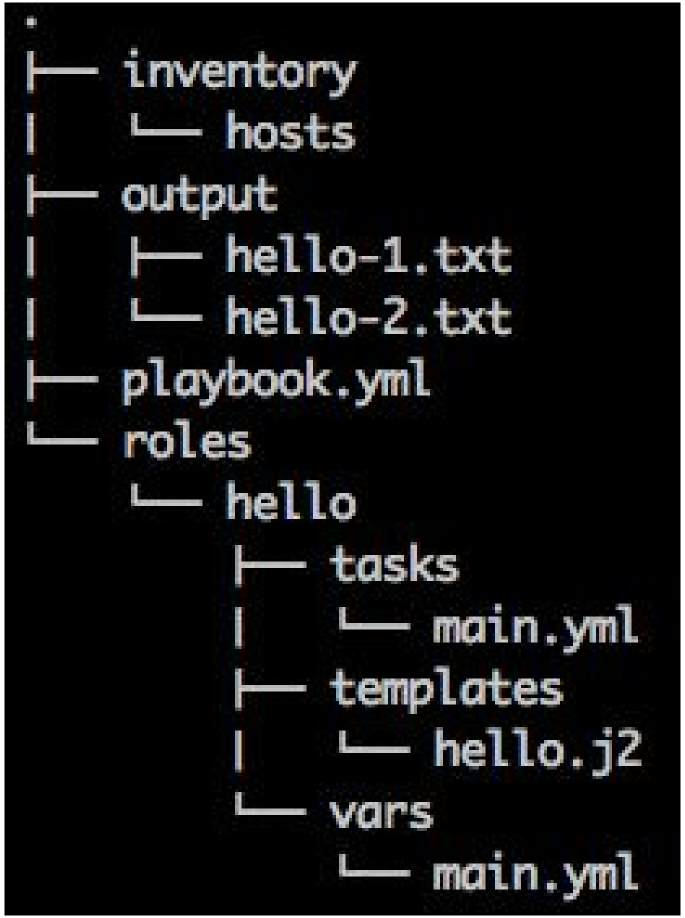
```
➜ hello_world git:(master) ansible-playbook -i inventory playbook.yml

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
```



```
.
├── inventory
│   └── hosts
├── output
│   ├── hello-1.txt
│   └── hello-2.txt
├── playbook.yml
└── roles
    └── hello
        ├── tasks
        │   └── main.yml
        ├── templates
        │   └── hello.j2
        └── vars
            └── main.yml
```

Hello world (after)

Structure

```
├── myplaybook.yml
├── inventory
│   ├── group_vars
│   │   └── sites.yml
│   └── hosts
└── roles
    ├── router
    │   ├── tasks
    │   │   └── main.yml
    │   ├── templates
    │   │   └── template1.j2
    │   └── vars
    │       └── main.yml
    └── switch
```

- 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 are our demo vars

```
$ command-line
```

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`

IP address filter

- The Jinja2 `ipaddr()` filter is included in Ansible as of version 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

```
$ 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)

Hosts

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

```
localhost
```

```
$ command-line
```

Playbook

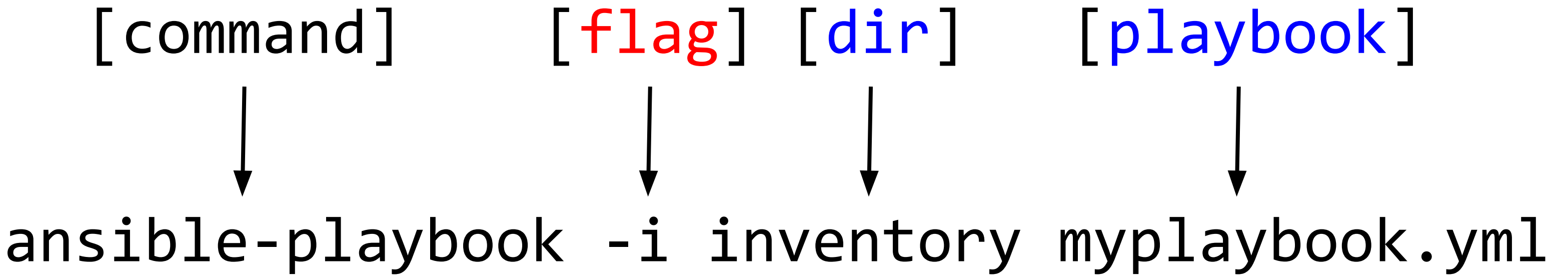
- Brings it together:
  - Hosts
  - Roles
    - Tasks
    - Templates
  - Variables
- And executes!

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

  roles:
   - router
```

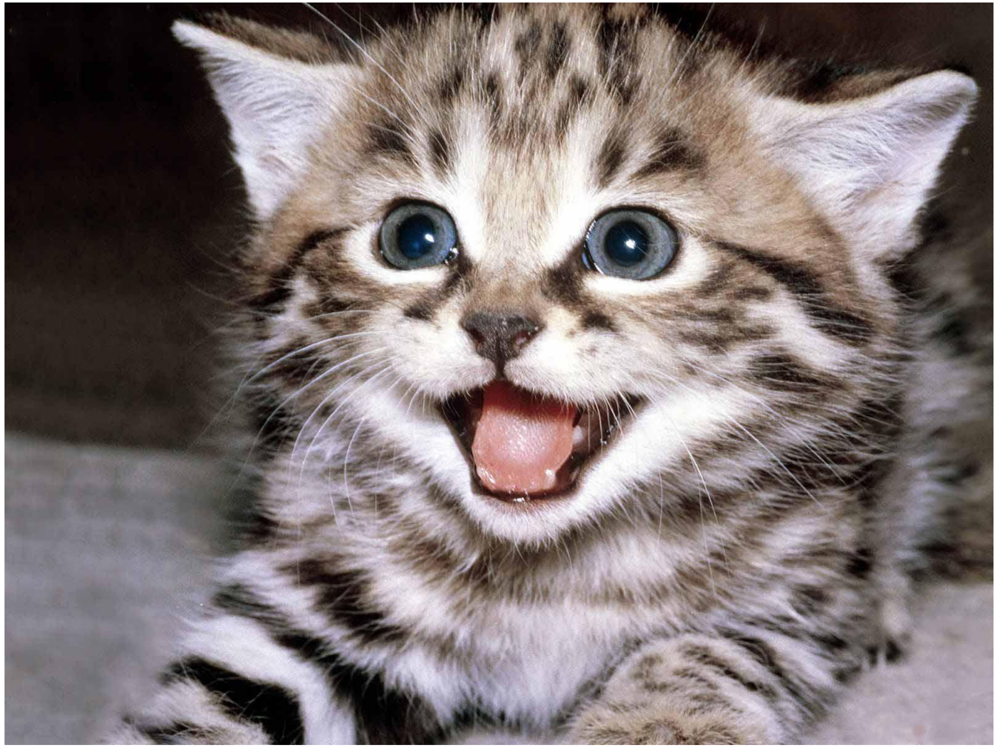
`$ command-line`

Running a play



```
ansible-playbook -i inventory myplaybook.yml
```

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***

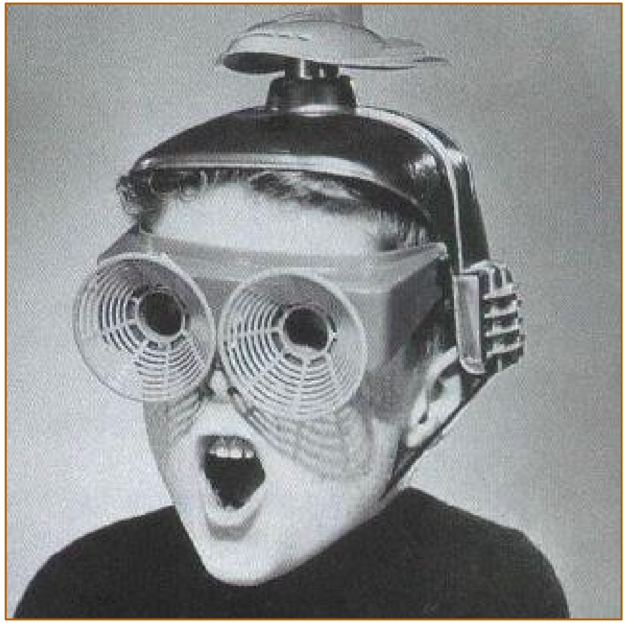
ERROR: Syntax Error while loading YAML script

$ command-line

More ipaddr & jinja2 fun

- Let’s try some loops!
  - multiple interfaces
- And add some logic!
  - if internal, enable cdp
- And do some IP address manipulation!
  - configure dhcp

Next steps



Deploy configs to your equipment

- NAPALM (NANOG64) multi-NOS *(EOS, JunOS, IOS-XR, FortiOS)*
- OS-specific modules: NX-OS, JunOS, Cumulus, Comware, ...

Learn more Ansible

Learn (a little) Python

- Custom filters/modules

Next steps (example deployment)

```
tasks:

    - name: “Compile templates”
      template: src=templates/routers.j2
      dest=configs/{{ inventory_hostname }}.cfg

    - name: “Deploy configuration”
      install_config:
      host={{ inventory_hostname }}
      file=configs/{{ inventory_hostname }}.cfg
```

Homework

High returns, low barrier effort

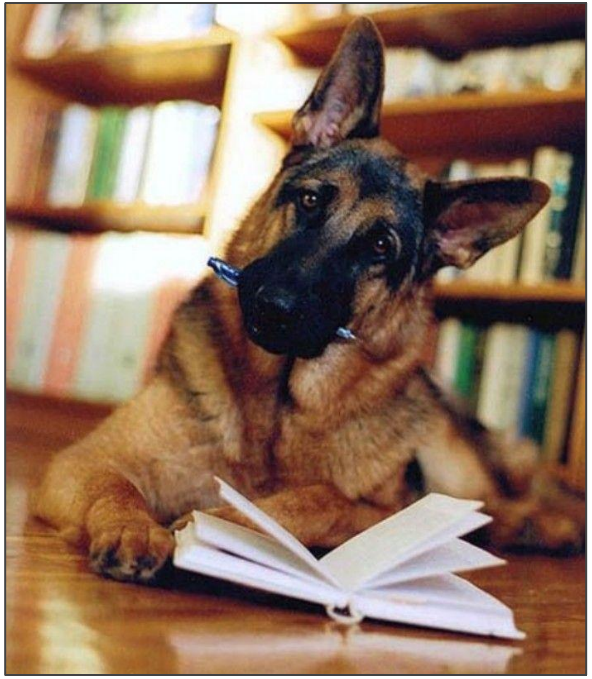
- Template *your* configs
- NMS or monitoring systems

Ratify a **source of truth**

- Database, IPAM, Rancid, spreadsheet... choose one!

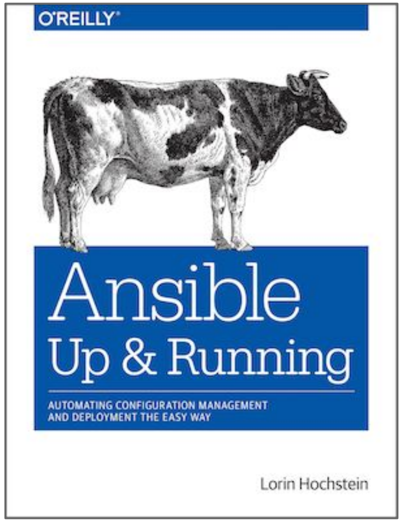
Prototype outside of production

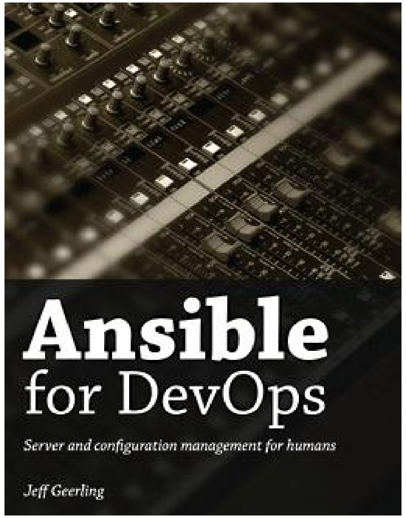
- VM's of your HW appliances



Some resources

books


O'REILLY
Ansible
Up & Running
AUTOMATING CONFIGURATION MANAGEMENT AND DEPLOYMENT THE EASY WAY
Lorin Hochstein



blogs/sites

- https://blog.tylerc.me/
- https://pynet.twb-tech.com/
- http://jedelman.com/
- http://packetpushers.net/
- http://keepingitclassless.net/

stackoverflow

… and more!

The future...

Maybe an Ansible 201 tutorial...?

- Advanced templating techniques
- Parsing existing configs
- Dynamic inventory & advanced variable management
- Interacting w/ network devices
- Using Ansible gathered Facts

Remember: this is an investment

1. This isn’t a panacea/cure-all

2. It takes time - start small and be iterative in effort

3. Don’t get discouraged

Give us feedback!

1. Come talk to us *(here all week!)*
2. Email or tweet us

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