# Network Automation: Ansible 102

APRICOT - Feb 28th, 2017

Bronwyn Lewis and Matt Peterson

## Agenda

**Tutorial** (add-on to previous 101 session)

- ➔ Inventory
- ➔ Templating
- ➔ Dynamic inventory
- ➔ Vendor & community network modules
- ➔ Miscellaneous & final advice

## Tutorial repository

https://git.io/vZKZH

## Quick notes

# “Best practices”

- ➔ Several different ways to approach most things in Ansible
- ➔ We consider these approaches our “best practices”
- ➔ Keep it simple; things will get complex without your help!

# Variables & accessing them

Variables

```
location: sfo
```

Lists

```
location:
  - sfo
  - sgn
```

Dictionaries

```
sites:
  - location: sfo
    type: customer
  - location: sgn
    type: backbone
```

# Variables & accessing them

Variables

```
    {{ location }}
```

Lists (looping)

```
    {{ item }}
```

Dictionaries (looping)

```
    {{ item.location }}
    {{ item.x.location }}
```

## Inventory

# `inventory`

- ➔ Directory of files to be used when executing a playbook:
  **`ansible-playbook -i inventory playbook.yml`**

- ➔ Can contain (but does not require):
  - `hosts`
  - `host_vars`
  - `group_vars`

- ➔ Overrules “roles” for variable precedence, but can be overridden by other variable locations (see docs)

# `hosts`

- ➔ Default: `/etc/ansible/hosts`
    - Hard to manage across systems
    - Usually need to be root to edit
    - Can’t easily keep in revision control

- ➔ Inventory: `./inventory/hosts`
    - Easily manage in revision control
    - No root needed!
    - Ensure everyone is using correct hosts

# hosts

```
[all-routers:children]
customer-routers
peering-routers

[customer-routers]
car01.sgn01     ansible_host=192.0.2.20   model=mx104
car01.hkg01     ansible_host=192.0.2.21   model=mx104

[peering-routers]
pat01.lax01     ansible_host=192.0.3.40   model=asr1k
pat01.tyo01     ansible_host=192.0.3.41   model=ask9k

[websites]
bronwynlewis.com
```

# hosts & [host|group]_vars

***% tree***

```
├ ansible
│  ├── inventory
│  │   ├── group_vars
│  │   │   ├── all.yml
│  │   │   ├── development.yml
│  │   │   └── production.yml
│  │   ├── hosts
│  │   └── host_vars
│  │       ├── jumphost.yml
│  │       └── mail.yml
│  ├── playbook.yml
│  └── roles
```

***% cat inventory/hosts***

```
[production:children]
admin
website

[admin]
mail
jumphost

[website]
web1
web2
db1
```

## `host_vars`

- ➔ Host-specific variables
- ➔ `host_vars/mail.yml`
    - Variables to be used by the `mail` host
- ➔ `host_vars/jumphost.yml`
    - Variables to be used by the `jumphost` host

# host_vars

***<u>% cat inventory/hosts</u>*** (before)

```
[customer-routers]
car01.sgn01     ansible_host=192.0.2.20   model=mx104
```

***<u>% cat inventory/host_vars/car01.sgn01.yml</u>***

```
---
ansible_host: 192.0.2.20
model: mx104
location: sgn
```

***<u>% cat inventory/hosts</u>*** (after)

```
[customer-routers]
car01.sgn01
```

# `group_vars`

- ➔ Host group-specific variables
- ➔ `group_vars/production.yml`
  - Vars to be used by any host in `production` group
- ➔ `group_vars/all.yml`
  - Contains vars to be used by ALL hosts

# Ansible system facts

```
$ command-line
```

Run the following to display facts about your local system:

```
ansible all -i localhost, -m setup --connection=local
```

## Ansible system facts

`$ command-line`

Example output:

```
[...]
"ansible_distribution": "Ubuntu",
"ansible_distribution_major_version": "16",
"ansible_distribution_release": "xenial",
"ansible_distribution_version": "16.04",
"ansible_dns": {
    "nameservers": [
        "8.8.8.8",
        "4.2.2.2"
    ]
},
[...]
```

# Ansible JunOS facts

Install prerequisite Python & Ansible modules:

```
% sudo -H pip install passlib

% sudo -H ansible-galaxy install Juniper.junos

% ansible-playbook -i bordergw01.hkg.domain.tld, test.yml
```

## Ansible JunOS facts

Example play:

```
---
- name: Sample play
  hosts: bordergw01.hkg.domain.tld
  roles:
    - Juniper.junos
  gather_facts: no
  connection: local

  vars:
    ansible_user: matt

  vars_prompt:
    name: passwd
    prompt: JunOS password
    private: yes
```

# Ansible JunOS facts

(continued) example play:

```
tasks:
  - name: Collect JunOS facts
    junos_get_facts:
        host={{ inventory_hostname }}
        user={{ ansible_user }}
        passwd={{ passwd }}
    register: junos

  - name: Dump JunOS facts to stdout
    debug: msg="{{ junos.facts }}"
```

# Ansible JunOS facts

```
% ansible-playbook -i bordergw01.hkg.domain.tld, test.yml
JunOS password:

PLAY [Sample play] ***********************************

TASK [Collect JunOS facts] ***************************

TASK [Dump JunOS facts to stdout] ********************
ok: [bordergw01.hkg.domain.tld] => {
    "msg": {
```

# Ansible JunOS facts

Example output:

```
[...]
"hostname": "bordergw01.hkg.domain.tld",
"ifd_style": "CLASSIC",
"master": "RE0",
"model": "MX104",
"personality": "MX",
"serialnumber": "H7931",
"switch_style": "BRIDGE_DOMAIN",
"vc_capable": false,
"version": "13.3R6.5",
"version_RE0": "13.3R6.5",
[...]
```

## Templating

## Variables in templates

- ➔ Filters allow variable manipulation
    - ipaddr filter is a great example
    - Perform regex, perform math, etc. against vars
- ➔ Can set variables inside of templates
    - Can be direct variables
    - Can be variables pulled in

## Filters

Default if variable not defined:

```
{{ foo | default(‘bar’) }}
```

Requiring a variable exist:

```
{{ foo | mandatory }}
```

# `{{ foo | mandatory }}`

When a `mandatory` variable is missing, Ansible will tell you:

```
TASK [hello : Generate "hello"] ************************************
failed: [localhost] (item={u'name': u'world', u'number': 1}) =>
{"failed": true, "item": {"name": "world", "number": 1}, "msg":
"AnsibleFilterError: Mandatory variable not defined."}
```

## Regex filters

Convert “airport.domain.tld” to “AIRPORT:

```
{{ “hkg.foo.com” | regex_replace('^(.*)\.(\w.*)$', '\\2') | upper }}
```

Print last octet of IP address (result is “123”):

```
{{ “198.168.0.123/24”|ipaddr('address')|splitext| \
     last|regex_replace('^[.]','') }}
```

# ipaddr filter

Example of ipaddr filter on **192.0.2.5/24**:

```
{{ item.ipv4 | ipaddr(‘network’) }}
```

```
ipaddr(‘address’)    → 192.0.2.5
ipaddr(‘network’)    → 192.0.2.0
ipaddr(‘netmask’)    → 255.255.255.0
ipaddr(‘broadcast’)  → 192.0.2.255
```

# ipaddr filter

Usable IP’s from variable **203.0.113.0/24**:

```
ipaddr(‘net’)                      ⟹ 203.0.113.0/24
ipaddr(‘net’)|ipaddr(‘1’)          ⟹ 203.0.113.1/24
ipaddr(‘net’)|ipaddr(‘1’)|ipaddr(‘address’)
                                   ⟹ 203.0.113.1
```

Validate subnet:

```
‘266.1.2.888/24’ | ipaddr(‘net’)  ⟹ False
‘192.168.0.3/24’ | ipaddr(‘net’)  ⟹ True
```

# Quick test filter

`$ command-line`

```
% ansible all -i localhost, -m debug -a \
"msg={{'203.0.113.0/24'|ipaddr('net')|ipaddr('-1')|ipaddr('address')}}"
localhost | SUCCESS => {
    "msg": "203.0.113.255"
}

% ansible all -i localhost, -m debug -a
"msg={{'266.1.2.3/24'|ipaddr('net')}}"
localhost | SUCCESS => {
    "msg": false
}

% ansible all -i localhost, -m debug -a \
"msg={{'2001:db8::/32'|ipaddr('net')|ipaddr('2')}}"
localhost | SUCCESS => {
    "msg": "2001:db8::2/32"
```

## Setting vars

Variable in template:

```
{{ item.ipv4 | ipaddr(‘network’) }}
```

Setting variable “alias” at beginning of template:

```
{% set network = item.ipv4 | ipaddr('network') %}

{{ network }}
```

## Setting vars

***<u>% cat templates/dns.j2</u>***

```
{% set dnsservers = [‘192.168.1.2’, ‘172.16.100.4’] %}

{% for nameserver in dnsservers %}
ip name-server {{ nameserver }}
{% endfor %}
```

***<u>% cat output</u>***

```
ip name-server 192.168.1.2
ip name-server 172.16.100.4
```

## Include in template

- ➔ Include allows you to pull in other templates to be included in your main template
- ➔ Makes it easy to break up & reuse template components for maintainability and clarity
- ➔ Should always be wrapped in conditional logic

# Include

***<u>% cat templates/base-config.j2</u>***

```
{% if dhcp is True %}
 {% include dhcp.j2 %}
{% endif %}
```

***<u>% cat templates/dhcp.j2</u>***

```
   service dhcp
   ip dhcp pool {{ item.hostname | upper }}
    [...]
```

## Tests

***% cat templates/base-config.j2***

```
{% if dhcp is True %}
 {% include dhcp.j2 %}
{% endif %}
```

***% cat inventory/host_vars/router-sfo01.yml***

```
dhcp: True
```

## Loops

- ➔ It’s just like a loop in other programming languages
- ➔ Iterate over lists/dicts of data
- ➔ Great for looping over interfaces, for example

# Loops

***% cat vars/main.yml***

```
 -  users:
   - first_name: david
     last_name: bowie
     team: accounting
   - first_name: grace
     last_name: jones
     team: engineering
```

***% cat templates/user_details.j2***

```
 {% for x in users %}
  {{ x.first_name | title }}, {{ x.team | title }}
 {% endfor %}
```

***% cat output***

```
David, Accounting
Grace, Engineering
```

## Templating:
## Advanced examples

# Variable within a template

```
{% set macros = {
    "DNS-SERVERS": "system name-server <*>",
    "NTP-SERVERS": "system ntp server <*>"
    "SNMP-MANAGERS": "snmp community <*> clients <*>",
%}

{% for key, value in macros.iteritems() %}
        <prefix-list>
          <name>{{ key }}</name>
          <apply-path>{{ value|escape }}</apply-path>
        </prefix-list>
{% endfor %}
```

# Output

```
<prefix-list>
  <name>DNS-SERVERS</name>
  <apply-path>system name-server &lt;*&gt;</apply-path>
</prefix-list>
<prefix-list>
  <name>NTP-SERVERS</name>
  <apply-path>system ntp server &lt;*&gt;</apply-path>
</prefix-list>
<prefix-list>
  <name>SNMP-MANAGERS</name>
  <apply-path>snmp community &lt;*&gt; clients &lt;*&gt;</apply-path>
</prefix-list>
```

# Input (host & group vars)

***% cat inventory/host_vars/car01.sgn01.yml***

```
---
interfaces:
  lo0:
    description: "Loopback"
    v4_address: 192.0.2.20/32
```

***% cat inventory/host_vars/car01.hkg01.yml***

```
---
interfaces:
  lo0:
    description: "Loopback"
    v4_address: 192.0.2.21/32
```

***% cat inventory/host_vars/pat01.lax01.yml***

```
---
interfaces:
  lo0:
    description: "Loopback"
    v4_address: 192.0.3.40/32
```

***% cat inventory/hosts***

```
[customer-routers]
car01.sgn01
car01.hkg01

[peering-routers]
pat01.lax01

[all-routers:children]
customer-routers
peering-routers
```

# Input (template)

```
{% for router in groups['all-routers']|sort %}
{# build iBGP sessions to all routers, except self #}
{% if hostvars[router]['inventory_hostname'] != inventory_hostname %}
 <neighbor>
   <name>{{ hostvars[router]['interfaces']['lo0']['v4_address']|ipaddr('address') }}</name>
   <description>{{ router }}</description>
 </neighbor>
{% endif %}
{% endfor %}
```

# Output

***% cat output/cat01.sgn01.conf***

```
<bgp>
 <neighbor>
   <name>192.0.2.21</name>
   <description>car01.hkg01</description>
 </neighbor>
 <neighbor>
   <name>192.0.3.40</name>
   <description>pat01.lax01</description>
 </neighbor>
</bgp>
```

***% cat output/car02.hkg.conf***

```
<bgp>
 <neighbor>
   <name>192.0.2.20</name>
   <description>car01.sgn01</description>
 </neighbor>
 <neighbor>
   <name>192.0.3.40</name>
   <description>par01.lax02</description>
 </neighbor>
</bgp>
```

***% cat output/pat01.lax01.conf***

```
<bgp>
 <neighbor>
   <name>192.0.2.20</name>
   <description>car01.sgn01</description>
 </neighbor>
 <neighbor>
   <name>192.0.2.21</name>
   <description>car01.hkg01</description>
 </neighbor>
</bgp>
```

## Dynamic Inventory

# Dynamic inventory

- ➔ JSON output of a script within `inventory` directory
- ➔ Any scripting language (Python, Ruby, bash, etc)
- ➔ Warning: Be aware of duplicate variables or keys

# Input (CSV file)

| A_Device | A_Port | Z_Device | Z_Port | v4_Network | v6_Network | Type | Description |
|---|---|---|---|---|---|---|---|
| pat01.lax | xe-1/0/2 | | | 198.51.100.47/24 | 2001:db8:51::47/64 | Peering | Angeles-IX |
| car04.sfo | ge-2/0/8 | | | 192.0.2.248/30 | 2001:db8:0::/56 | Customer | 6Ten Stores |
| br02.sgn | xe-1/2/6 | | | 203.0.113.2/30 | 2001:db8:0:910::1/127 | Transit | PTT (Ckt #123) |
| bbr01.hkg | he-1/4/8 | bbr01.sin | he-2/1/7 | 192.0.2.4/31 | 2001:db8:ba:24::/127 | Backbone | |

# Dynamic inventory

```
% ./inventory/csv2json.py --list
---
{
   "_meta": {
      "hostvars": {
         "pat01.lax": {
            "interfaces": {
               "xe-1/0/2": {
                  "description": "PEERING: Angeles-IX”,
                  "type": "Peering",
                  "v4_address": "198.51.100.47/24",
                  "v6_address": "2001:Db8:51::47/647"
               }
            }
         },
```

# Dynamic inventory

```
"car04.sfo": {
   "interfaces": {
      "ge-2/0/8": {
         "description": "CUSTOMER: 6Ten Stores”,
         "type": "Customer",
         "v4_address": "192.0.2.249/30",
         "v6_address": "2001:db8:0::/56"
      }
   }
},
```

# Dynamic inventory

```
"bar02.sgn": {
   "interfaces": {
      "xe-1/2/6": {
         "description": "TRANSIT: PTT (Ckt #123)”,
         "type": "Transit",
         "v4_address": "203.0.113.2/30",
         "v6_address": "2001:db8:0:910::1/127"
      }
   }
},
```

# Dynamic inventory

```
      "bbr01.hkg": {
        "interfaces": {
           "he-1/4/8": {
              "description": "BACKBONE: bbr01.sin:he-2/1/7”,
              "type": "Backbone",
              "v4_address": "192.0.2.4/31",
              "v6_address": "2001:db8:ba:24::/127"
...
     "bbr01.sin": {
        "interfaces": {
           "he-2/1/7": {
              "description": "BACKBONE: bbr01.hkg:he-1/4/8”,
              "type": "Backbone",
              "v4_address": "192.0.2.5/31",
              "v6_address": "2001:db8:ba:24::1/127"
```

# Dynamic inventory

```
...
  "ungrouped": {
      "vars": {
         "dns_entries": [
            {
               "domain": "myisp.com.",
               "record_name": "he-1-4-8.bbr01.hkg.myisp.com”
               "record_type": "A",
               "record_value": "192.0.2.4"
            },
            {
               "domain": "2.0.192.in-addr.arpa",
               "record_name": "4",
               "record_type": "PTR",
               "record_value": "he-1-4-8.bbr01.myisp.com."
```

## Vendor & Community Modules

# Modules

- ➔ Mixture of vendor & community  authored code
  *JunOS, Cisco {IOS, IOSxr, NX-OS, ASA), Arista EOS, F5 BigIP, A10, Cumulus, DellOS, VyOS, and more!*

  *NAPALM (* above + FortiOS, Mikrotik, and more!)*

- ➔ Transport: SSH, HTTP, SNMP
- ➔ Interface: custom API (REST), NETCONF, OpenConfig
- ➔ Warning: Syntax is not consistent between modules

# JunOS modules

| Module | Input formats | Password directive | Notes |
|---|---|---|---|
| Juniper “core” | Indented, set, or XML | “passwd” | |
| Juniper “junos-stdlib” | Indented, set, or XML | “passwd” | Serial console, telnet support |
| NAPALM | Indented | “password” | Powerful validation & facts support |

# JunOS modules example

```
---
- name: Backup running configuration
  junos_get_config: >
    host={{ ansible_host }} format=xml
    dest=myrouters/{{ inventory_hostname }}_running.xml

- name: Compile configuration
  template: >
    src=juniper.conf.j2
    dest=myrouters/{{ inventory_hostname }}_compiled.xml

- name: Deploy configuration
  junos_install_config: >
    host={{ ansible_host }} replace=yes timeout=45
    file=myrouters/{{ inventory_hostname }}_compiled.xml
```

## Miscellaneous

## Conditionals

```
- name: Create new user account
  user: name={{ item.name }} password={{ item.password }}
  when: ansible_distribution == "CentOS"

- name: Create new user account
  user: name={{ item.name }} password={{ item.password }}
  when:
    - ansible_distribution == "CentOS"
    - “{{ item.user_type }}” == “admin”
```

# Comments: Input

```
% cat ansible.cfg
[defaults]
ansible_managed = %a %b %d %H:%M:%S %z %Y

% cat roles/juniper/templates/main.j2
<configuration operation="replace" xmlns:junos="junos">
    <junos:comment>
##############################################################################
# HEADER: This file was generated by Ansible on {{ ansible_managed }}
# HEADER: Source {{ template_path }}
# HEADER: Built by {{ template_uid }} on {{ template_host }}
##############################################################################
    </junos:comment>
    <system>
        <host-name>{{ inventory_hostname | mandatory }}</host-name>
...
```

# Comments: Output

```
% cat output/edge01.fmt01.conf
<configuration operation="replace" xmlns:junos="junos">
    <junos:comment>
##############################################################################
# HEADER: This file was generated by Ansible on Mon Feb 20 18:34:26 -0800 2017
# HEADER: Source /Users/matt/work/ansible/roles/juniper/templates/main.j2
# HEADER: Built by matt on BSD4lyfe.local
##############################################################################
    </junos:comment>
    <system>
        <host-name>edge01.fmt01</host-name>
...
```

## Comments: Output

```
matt@edge01.fmt01> show configuration
/*
###############################################################################
# HEADER: This file was generated by Ansible on Mon Feb 20 18:34:26 -0800 2017
# HEADER: Source /Users/matt/work/ansible/roles/juniper/templates/main.j2
# HEADER: Built by matt on BSD4lyfe.local
##############################################################################
*/
system {
    host-name edge01.fmt01;
...
```

# Debugging

- ➔ Variables - flat host/group vars, dynamic, templates, ...
- ➔ Consider ansible-dumpall to dump all variables, per each host and/or group to local text file
- ➔ Enable `error_on_undefined_vars` in `ansible.cfg`
- ➔ Run `ansible -v` verbose mode

## Gameplan

# Planning to prototype

Ratify a **source of truth**

- Database, IPAM, CSV file, spreadsheet... choose one!
- Consider `ntc_show_command` to parse CLI

Archive

- Backup current configs with Oxidized, Rancid, …

# Prototype to testing

Don’t break production!

- Consider VM’s (simulation)
- Lab or retired equipment

“Unit testing”

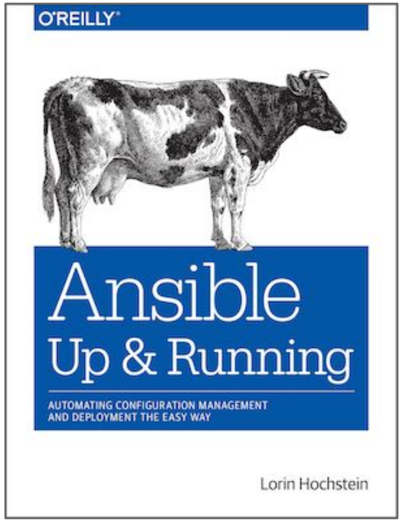
- Validate syntax (lint)

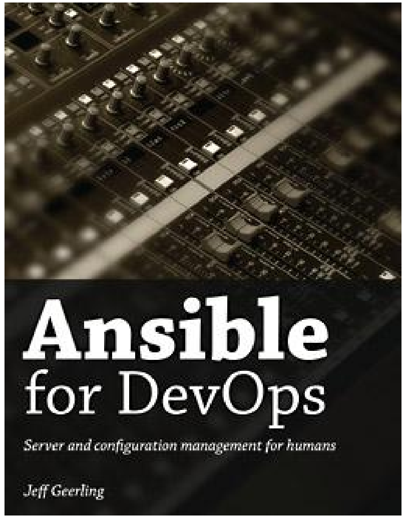
“Functional testing”

- Validate configs build prescribed IGP routing, correct MTU, working MD5 hashes, etc.

# Some resources

## books


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



## 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!

# Thanks!

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

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