Network Monitoring, Management and Automation

Introduction to Ansible

npNOG 5

Dec 8 - 12, 2019

What is Ansible?

- A ***configuratoin management*** tool
- Applies changes to your system to bring it to a desired state
- Similar applications include
  - SaltStack
  - Puppet
  - Chef
  - Juju
  - CFEngine

Why choose Ansible?

- Target system requires only sshd and python
  - No daemons or agents to install
- Security
  - Relies on ssh
- Easy to get started, compared to the others!

Ansible Modules

- Ansible ***modules*** are small pieces of code which perform one function
  - eg. copy a file, start or stop a daemon
- Most are ***idempotent***: running repeatedly has the same effect as running once
  - only makes a change when the system is not already in the desired state
- Many modules supplied as standard
  - https://docs.ansible.com/modules.html

Invoking modules from shell

```
          Host or Group                  Module Name
                |                             |
                V                             V
$ ansible vmX-gY.lab.workalaya.net -m service \
        -a "name=apache2 state=started"
           ----------------------------
                       ^
                       |
                 Module arguments
```

Configuring Ansible behaviour

- ***Tasks*** are modules called with specific arguments
- ***Handlers*** are triggered when something changes
  - e.g. restart daemon when a config file is changed
- ***Roles*** are re-usable bundles of tasks, handlers and templates
- All defined using YAML

YAML

- A way of storing structured data as text
- Conceptually similar to JSON
  - String and numeric values
  - Lists: ordered sequences
  - Hashes: unordered groups of key-value pairs
- String values don't normally need quotes
- Lists and hashes can be nested
- Indentation used to define nesting

YAML list (ordered sequence)

- single line form

```
[name, address, age]
```

- multi-line form

```
- name
- address
- age
 ^
 Space after dash required
```

YAML hash (key-value pairs)

- single line form

```
{item: shirt, colour: red, size: 40}
      ^
      Space after colon required
```

- multi-line form

```
item: shirt
colour: red
size: 40
description: |
  this is a very long multi-line
  text field which is all one value
```

Nesting: list of hashes

- compact

```
- {item: shirt, colour: red, size: 40}
- {item: shirt, colour: green, size: 44}
```

- multi-line form

```
- item: shirt
  colour: red
  size: 40
- item: shirt
  colour: green
  size: 44
  ^
  Note alignment
```

More complex YAML example

```
A list with 3 items
|
|  each item is a hash (key-value pairs)
|  |
V  V
- do: laundary  <-- simple value
  item:
    - shirts    <-- list value (note indentation)
    - trousers
- do: shopping
  item:
    - bread
    - jam
- do: relax
  eat:
    - chips
    - fruits
```

Ansible Playbook

```
Top level: a list of "plays"
|  Each play has "hosts" plus "tasks" and/or "roles"
|  |
V  V
- hosts:
    - vm1-g1.lab.workalaya.net
    - vm2-g2.lab.workalaya.net
  tasks:
    - name: install Apache
      action: package name=apache2 state=present
    - name: ensure Apache is started
      action: service name=apache2 state=started
- hosts: dns_servers
  roles:
    - dns_server
    - ntp
```

Ansible Roles

- A bundle of related tasks/handlers/templates

roles/ *<rolename>* **/tasks/main.yml**
roles/ *<rolename>* **/handlers/main.yml**
roles/ *<rolename>* **/defaults/main.yml**
roles/ *<rolename>* **/files/...**
roles/ *<rolename>* **/templates/...**

- Recommended way to make re-usable configs
- Not all these files need to be present

Ansible Tags

- Each role or individual task can be labelled with one or more "tags"
- When you run a playbook, you can tell it only to run tasks with a particular tag: -t <tag>
- Lets you selectively run parts of playbooks

Ansible Inventory

- Lists all hosts which Ansible may manage
- Simple "INI" format, not YAML
- Can define groups of hosts
- Default is /etc/ansible/hosts
  - Can override using -i <filename>

Inventory (hosts) example

```
[dns_servers]           <-- Name of group
ns1.lab.workalaya.net   <-- Hosts in this group
ns2.lab.workalaya.net

[vms]
vm1-g1.lab.workalaya.net
vm1-g1.lab.workalaya.net

[nagios_server]
noc.lab.workalaya.net
vm1-g1.lab.workalaya.net
vm1-g1.lab.workalaya.net
```

```
Note:
- the same host can be listed under multiple groups.
- Group "all" is created automatically
```

Inventory variables

- You can set variables on hosts or groups of hosts
- Variables can make tasks behave differently when applied to different hosts
- Variables can be inserted into templates
- Some variables control how Ansible connects

Setting host vars

- Directly in the inventory (hosts) file

```
[core_servers]
ns1.lab.workalaya.net ansible_connection=local
ns2.lab.workalaya.net
```

- In file host_vars/pc2.example.com

```
ansible_ssh_host: 10.10.0.241
ansible_ssh_user: root
flurble:
  - foo
  - bar
```

  - This is in YAML and is preferred

Setting group vars

- **group_vars/dns_servers**

```
# More YAML
flurble:
  - foo-foo
  - bar-foo
```

- **group_vars/all**

```
# More YAML, applies to every host
ansible_ssh_user: lab
ansible_beccome_pass: yourpass
```

```
Note: host vars take priority over group vars
```

Ansible Facts

- Facts are variables containing information collected automatically about the target host
- Things like what OS is installed, what interfaces it has, what disk drives it has
- Can be used to adapt roles automatically to the target system
- Gathered every time Ansible connects to a host (unless playbook has "gather_facts: no")

Showing facts

```
~$ ansible vmX-gY.lab.workalaya.net -m setup | less

vmX-gY.lab.workalaya.net | SUCCESS => {
    "ansible_facts": {
        "ansible_all_ipv4_addresses": [
            "100.68.X.21"
        ],
        "ansible_architecture": "x86_64",
        "ansible_bios_date": "12/12/2018",
        "ansible_bios_version": "6.00",
        "ansible_cmdline": {
            "BOOT_IMAGE": "/boot/vmlinuz-4.15.0-58-generic",
            "ro": true,
            "root": "/dev/mapper/lab--main--vg-root"
        },
        "ansible_date_time": {
            "date": "2019-11-13",
            "day": "13",
            "epoch": "1573634010",
```

jinja2 template examples

- Insert a variable into text

```
INTERFACES="{{ dhcp_interfaces }}"
```

- Looping over lists

```
search lab.workalaya.net
{% for host in dns_servers %}
nameserver {{ host }}
{% endfor %}
```

Many other cool features

- conditionals

```
- action: package name=apache2 state=present
  when: ansible_os_family=='Debian'
```

- Loops

```
- action: package name={{item}} state=present
  with_items:
    - openssh-server
    - rsync
    - telnet
```

Getting up-to-date Ansible

- Your package manager's version may be old
- For Ubuntu LTS: use the PPA

```
apt-get install python-software-properties
add-apt-repository ppa:rquillo/ansible
apt-get update
apt-get install ansible
```

- or, if using python venv

```
(venv) vmX-gY@ansible-gY:~/ansible-playbook$ pip install --upgrade ansible
```

More info and documentation

- https://docs.ansible.com/
- https://jinja.palletsprojects.com/
- https://yaml.org/