ANSIBLE 2.0

Introduction to Ansible training

Marco Berube
sr. Cloud Solution Architect

Michael Lessard
Senior Solutions Architect

Martin Sauvé
Senior Solutions Architect
INTRODUCTION TO ANSIBLE
An **ansible** is a *fictional* machine capable of instantaneous or superluminal communication. It can send and receive messages to and from a corresponding device over any distance whatsoever with no delay. **Ansibles** occur as plot devices in **science fiction** literature
-- wikipedia
Intro to Ansible

Michael DeHaan (creator cobbler and func)
https://www.ansible.com/blog/2013/12/08/the-origins-of-ansible

“Ansible owes much of it's origins to time I spent at Red Hat's Emerging Technologies group, which was an R&D unit under Red Hat's CTO”
- Michael DeHaan

“...because Puppet was too declarative you couldn't use it to do things like reboot servers or do all the "ad hoc" tasks in between…”
- Michael DeHaan

Ansible
Simple
Can manage almost any *IX through SSH
requires Python 2.4
Windows (powershell, winrm python module)
It's been 18 months since I've been at an OpenStack summit. One of the most notable changes for me this summit has been Ansible. Everyone seems to be talking about Ansible, and it seems to be mainly customers rather than vendors. I'm sure if I look around hard enough I'll find someone discussing Puppet or Chef but I'd have to go looking ..... “
Andrew Cathrow, April 2016, on Google+
USE-CASES

Some examples...

Provisioning
Configuration management
Application deployments
Rolling upgrades - CD
Security and Compliance
Orchestration
BENEFITS

Why is Ansible popular?

➔ **Efficient**: Agentless, minimal setup
➔ **Fast**: Easy to learn/to remember, simple declarative language
➔ **Scalable**: Can managed thousands of nodes
➔ **Secure**: SSH transport
➔ **Large community**: thousands of roles on Ansible Galaxy
ANSIBLE - THE LANGUAGE OF DEVOPS

COMMUNICATION IS THE KEY TO DEVOPS.

Ansible is the first **automation language** that can be read and written across IT.

Ansible is the only **automation engine** that can automate the entire **application lifecycle** and **continuous delivery** pipeline.
KEY COMPONENTS

Understanding Ansible terms

★ Modules (Tools)
★ Tasks
★ Inventory
★ Plays
★ Playbook (Plan)
INSTALLING ANSIBLE

How-to

# ENABLE EPEL REPO
yum install epel-release

# INSTALL ANSIBLE
yum install ansible
MODULES

What is this?

*Bits of code copied to the target system.*

*Executed to satisfy the task declaration.*

*Customizable.*
MODULES

Lots of choice / Ansible secret power...

- Cloud Modules
- Clustering Modules
- Commands Modules
- Database Modules
- Files Modules
- Inventory Modules
- Messaging Modules
- Monitoring Modules
- Network Modules
- Notification Modules
- Packaging Modules
- Source Control Modules
- System Modules
- Utilities Modules
- Web Infrastructure Modules
- Windows Modules
# LIST ALL MODULES
ansible-doc -l

# VIEW MODULE DOCUMENTATION
ansible-doc <module_name>
MODULES

commonly used

- apt/yum
- copy
- file
- get_url
- git
- ping

- service
- synchronize
- template
- uri
- user
- wait_for
ANSIBLE COMMANDS
INVENTORY

Use the default one /etc/ansible/hosts or create a host file

[centos@centos1 ~]$ mkdir ansible ; cd ansible
[centos@centos1 ~]$ vim hosts

[all:vars]
ansible_ssh_user=centos

[web]
web1 ansible_ssh_host=centos2

[admin]
ansible ansible_ssh_host=centos1
COMMANDS

Run your first Ansible command...

```
# ansible all -i ./hosts -m command -a "uptime"

192.168.250.13 | success | rc=0 >>
18:57:01 up 11:03,  1 user,  load average: 0.00, 0.01, 0.05

192.168.250.11 | success | rc=0 >>
18:57:02 up 11:03,  1 user,  load average: 0.00, 0.01, 0.05
```
COMMANDS

Other example of commands

# INSTALL HTTPD PACKAGE
ansible web -s -i ./hosts -m yum -a "name=httpd state=present"

# START AND ENABLE HTTPD SERVICE
ansible web -s -i ./hosts -m service -a "name=httpd enabled=yes state=started"
LAB #1

Ansible commands

Objectives

Using Ansible commands, complete the following tasks:

1. Test Ansible connection to all your hosts using ping module
2. Install EPEL repo on all your hosts
3. Install HTTPD only on your web hosts
4. Change SELINUX to permissive mode

Modules documentation:

http://docs.ansible.com/ansible/list_of_all_modules.html
LAB #1 - SOLUTION

```bash
ansible all -i ../hosts -m ping
ansible all -i ../hosts -s -m yum -a "name=epel-release state=present"
ansible web -i ../hosts -s -m yum -a "name=httpd state=present"
ansible all -i ../hosts -s -m selinux -a "policy=targeted state=permissive"
```
ANSIBLE PLAYBOOKS
- name: This is a Play
  hosts: web-servers
  remote_user: mberube
  become: yes
  gather_facts: no
  vars:
    state: present

  tasks:
    - name: Install Apache
      yum: name=httpd state={{ state }}
PLAYS

Naming

- name: This is a Play
PLAYS

Host selection

- name: This is a Play
  hosts: web
PLAYS

Arguments

- name: This is a Play
  hosts: web
  remote_user: mberube
  become: yes
  gather_facts: no
FACTS
Gathers facts about remote host

- Ansible provides many facts about the system, automatically
- Provide by the setup module
- If facter (puppet) or ohai (chef) are installed, variables from these programs will also be snapshotted into the JSON file for usage in templating
  - These variables are prefixed with facter_ and ohai_ so it’s easy to tell their source.
- Using the ansible facts and choosing to not install facter and ohai means you can avoid Ruby-dependencies on your remote systems

http://docs.ansible.com/ansible/setup_module.html
PLAYS

Variables & tasks

- name: This is a Play
  hosts: web-servers
  remote_user: mberube
  become: yes
  gather_facts: no
  vars:
    state: present

  tasks:
    - name: Install Apache
      yum: name=httpd state={{ state }}
RUN AN ANSIBLE PLAYBOOK

[centos@centos7-1 ansible]$ ansible-playbook play.yml -i hosts
RUN AN ANSIBLE PLAYBOOK

Check mode “Dry run”

[centos@centos7-1 ansible]$ ansible-playbook play.yml -i hosts --check
- name: This is a Play
  hosts: web-servers
  remote_user: mberube
  become: yes
  gather_facts: no
  vars:
    state: present

tasks:
  - name: Install Apache and PHP
    yum: name={{ item }} state={{ state }}
    with_items:
    - httpd
    - php
LOOPS

Many types of general and special purpose loops

- with_nested
- with_dict
- with_fileglob
- with_together
- with_sequence
- until
- with_random_choice
- with_first_found
- with_indexed_items
- with_lines

http://docs.ansible.com/ansible/playbooks_loops.html
HANDLERS

Only run if task has a “changed” status

- name: This is a Play
  hosts: web-servers

  tasks:
  - yum: name={{ item }} state=installed
    with_items:
      - httpd
      - memcached
    notify: Restart Apache

  - template: src=templates/web.conf.j2 dest=/etc/httpd/conf.d/web.conf
    notify: Restart Apache

handlers:
- name: Restart Apache
  service: name=httpd state=restarted
TAGS

Example of tag usage

tasks:

  - yum: name={{ item }} state=installed
    with_items:
      - httpd
      - memcached

tags:
  - packages

  - template: src=templates/src.j2 dest=/etc/foo.conf

tags:
  - configuration
TAGS

Running with tags

```
ansible-playbook example.yml --tags "configuration"
ansible-playbook example.yml --skip-tags "notification"
```
TAGS

Special tags

```bash
ansible-playbook example.yml --tags "tagged"
ansible-playbook example.yml --tags "untagged"
ansible-playbook example.yml --tags "all"
```
RESULTS

Registering task outputs for debugging or other purposes

# Example setting the Apache version
- shell: httpd -v|grep version|awk '{print $3}'|cut -f2 -d'/'
  register: result

- debug: var=result
CONDITIONAL TASKS

Only run this on Red Hat OS

- name: This is a Play
  hosts: web-servers
  remote_user: mberube
  become: sudo

  tasks:
  - name: install Apache
    yum: name=httpd state=installed
    when: ansible_os_family == "RedHat"
Apply a condition to multiple tasks at once

```yaml
tasks:
  - block:
    - yum: name={{ item }} state=installed
      with_items:
        - httpd
        - memcached
    - template: src=templates/web.conf.j2 dest=/etc/httpd/conf.d/web.conf
    - service: name=bar state=started enabled=True
    when: ansible_distribution == 'CentOS'
```
By default, Ansible stops on errors. Add the ignore_error parameter to skip potential errors.

- name: ping host
  command: ping -c1 www.foobarn.com
  ignore_errors: yes
ERRORS

Defining failure

You can apply a special type of conditional that if true will cause an error to be thrown.

```yaml
- name: this command prints FAILED when it fails
  command: /usr/bin/example-command -x -y -z
  register: command_result
  failed_when: "'FAILED' in command_result.stderr"
```
ERRORS

Managing errors using blocks

tasks:

- block:
  - debug: msg='i execute normally'
  - command: /bin/false
  - debug: msg='i never execute, cause ERROR!'
rescue:
  - debug: msg='I caught an error'
  - command: /bin/false
  - debug: msg='I also never execute :-(' 
always:
  - debug: msg="this always executes"
LINEINFILE

Add, remove or update a particular line

- lineinfile: dest=/etc/selinux/config regexp:^SELINUX=
  line=SELINUX=enforcing

- lineinfile: dest=/etc/httpd/conf/httpd.conf regexp="^Listen 
  insertafter="^#Listen 
  line="Listen 8080"

Great example here:
https://relativkreativ.at/articles/how-to-use-ansibles-lineinfile-module-in-a-bulletproof-way

Note: Using template or a dedicated module is more powerful
LAB #2

Configure server groups using a playbook

Objectives

*Using an Ansible playbook:*

1. Change SELINUX to permissive mode on all your hosts
2. Install HTTPD on your web hosts only
3. Start and Enable HTTPD service on web hosts only if a new httpd package is installed.
4. Copy an motd file saying “Welcome to my server!” to all your hosts
5. Copy an “hello world” index.html file to your web hosts in /var/www/html
6. Modify the sshd.conf to set PermitRootLogin at no
---
- name: Lab2 - All server setup
  hosts: all
  become: yes
  vars:
    selinux: permissive

  tasks:
  - name: Configure selinux to {{ selinux }}
    selinux:
      policy: targeted
      state: "{{ selinux }}"

  - name: Copy motd file
    copy: src=motd dest=/etc/motd

- name: Lab2 - Web server setup
  hosts: web
  become: yes

  tasks:
  - name: Install Apache
    yum: name=httpd state=present
    notify: Restart Apache

  - name: Copy Index.html

  - name: Set ssh root login at no
    lineinfile:
      dest=/etc/sshd/sshd_config
      line="PermitRootLogin no"
      state=present
    notify: RestartSSH

handlers:
- name: Restart Apache
  service: name=httpd state=restarted enabled=yes
- name: RestartSSH
  Service: name=sshd state=restarted enabled=yes
LAB #2 - SOLUTION #2

```yaml
---
- name: Lab2 - All server setup
  hosts: all
  become: yes

  tasks:
  - name: Configure selinux to {{ selinux }}
    selinux:
      policy: targeted
      state: "{{ selinux }}"

  - name: Copy motd file
    copy: src=motd dest=/etc/motd

...
ANSIBLE VARIABLES AND CONFIGURATION MANAGEMENT
### VARIABLE PRECEDENCE

**Ansible v2**

1. extra vars
2. task vars (only for the task)
3. block vars (only for tasks in block)
4. role and include vars
5. play vars_files
6. play vars_prompt
7. play vars
8. set_facts
9. registered vars
10. host facts
11. playbook host_vars
12. playbook group_vars
13. inventory host_vars
14. inventory group_vars
15. inventory vars
16. role defaults
MAGIC VARIABLES

Ansible creates and maintains information about its current state and other hosts through a series of “magic” variables.

★ **hostvars[inventory_hostname]**

★ **hostvars[<any_hostname>]**

```
{{ hostvars['test.example.com']['ansible_distribution'] }}
```

★ **group_names**

is a list (array) of all the groups the current host is in

★ **groups**

is a list of all the groups (and hosts) in the inventory.
MAGIC VARIABLES

Using debug mode to view content

```yaml
- name: debug
  hosts: all

  tasks:
    - name: Show hostvars[inventory_hostname]
      debug: var=hostvars[inventory_hostname]

    - name: Show ansible_ssh_host variable in hostvars
      debug: var=hostvars[inventory_hostname].ansible_ssh_host

    - name: Show group_names
      debug: var=group_names

    - name: Show groups
      debug: var=groups
```

```bash
ansible-playbook -i ..:/hosts --limit <hostname> debug.yml
```
Template module
Using Jinja2

Templates allow you to create dynamic configuration files using variables.

- template: src=/mytemplates/foo.j2 dest=/etc/file.conf owner=bin group=wheel mode=0644

Documentation:
http://docs.ansible.com/ansible/template_module.html
JINJA2

Delimiters

Ansible uses Jinja2. Highly recommend reading about Jinja2 to understand how templates are built.

```{variable}
{% for server in groups.webservers %}
```
{% for server in groups.web %}
    {{ server }}  {{ hostvars[server].ansible_default_ipv4.address }}
{% endfor %}

web1 10.0.1.1
web2 10.0.1.2
web3 10.0.1.3
JINJA2

Conditional

{% if ansible_processor_cores >= 2 %}
-smp enable
{% else %}
-smp disable
{% endif %}
JINJA2

Variable filters

{% set my_var='this-is-a-test' %}
{{ my_var | replace('-', '_') }}

this_is_a_test
JINJA2

Variable filters

{% set servers = "server1,server2,server3" %}
{% for server in servers.split("","") %}
{{ server }}
{% endfor %}

server1
server2
server3
JINJA2, more filters

Lots of options...

# Combine two lists
{{ list1 | union(list2) }}

# Get a random number
{{ 59 | random }} * * * * root /script/from/cron

# md5sum of a filename
{{ filename | md5 }}

# Comparisons
{{ ansible_distribution_version | version_compare('12.04', '>=') }}

# Default if undefined
{{ user_input | default('Hello World') }}
JINJA2

Testing

{% if variable is defined %}
{% if variable is none %}
{% if variable is even %}
{% if variable is string %}
{% if variable is sequence %}
Jinja2

Template comments

{% for host in groups['app_servers'] %}
    {# this is a comment and won't display #}
    {{ loop.index }} {{ host }}
{% endfor %}
YAML vs. Jinja2 Template Gotchas

YAML values beginning with a template variable must be quoted

```yaml
vars:
  var1: {{ foo }} <<< ERROR!
  var2: "{{ bar }}"
  var3: Echoing {{ foo }} here is fine
```
# Example setting the Apache version
- shell: httpd -v|grep version|awk '{print $3}'|cut -f2 -d'/'
  register: result

- set_fact:
  apache_version: "{{ result.stdout }}"
LAB #3

Configuration management using variables

Objectives

Modify your `lab2 playbook` to add the following:

1. Convert your MOTD file in a template saying: “Welcome to `<hostname>`!”
2. Install facter to all your hosts using an ansible command
3. Convert your `index.html` file into a template to output the following information:

   **Web Servers**
   - `lab1 192.168.3.52` - free memory: 337.43 MB
   - `lab2 192.168.3.53` - free memory: 346.82 MB
LAB #3 - Help (debug file)

---

- name: debug
  hosts: all

  tasks:

  - name: Show hostvars[inventory_hostname]
    debug: var=hostvars[inventory_hostname]

  - name: Show hostvars[inventory_hostname].ansible_ssh_host
    debug: var=hostvars[inventory_hostname].ansible_ssh_host

  - name: Show group_names
    debug: var=group_names

  - name: Show groups
    debug: var=groups
LAB #3 - SOLUTION - playbook

---

- name: Lab3 - All server setup
  hosts: all
  become: yes

  tasks:
  - name: Configure selinux to permissive
    selinux:
      policy: targeted
      state: permissive

  - name: Copy motd template
    template: src=motd.j2 dest=/etc/motd

- name: Lab3 - Web server setup
  hosts: web
  become: yes

  tasks:
  - name: Install Apache
    yum: name=httpd state=present
    notify: Restart Apache

  - name: Copy Index.html template
    template: src=index.html.j2 dest=/var/www/html/index.html
    notify: Restart Apache

  handlers:
  - name: Restart Apache
    service: name=httpd state=restarted enabled=yes
LAB #3 - SOLUTION - template files

motd.j2

Welcome to {{ hostvars[inventory_hostname].inventory_hostname }}!

index.html.j2

Web Servers<br>
{% for server in groups.web %}
{{ server }}  {{ hostvars[server].ansible_default_ipv4.address }} - free memory: {{ hostvars[server].facter_memoryfree }}<br>
{% endfor %}
{% endif %}
ANSIBLE ROLES
ROLES

A redistributable and reusable collection of:

- tasks
- files
- scripts
- templates
- variables
ROLES

Often used to setup and configure services

➔ install packages
➔ copying files
➔ starting deamons

Examples: Apache, MySQL, Nagios, etc.
ROLES
Directory Structure

```
roles
  └── myapp
      ├── defaults
      ├── files
      ├── handlers
      ├── meta
      ├── tasks
      └── templates
          └── vars
```
ROLES
Create folder structure automatically

ansible-galaxy init <role_name>
ROLES

Playbook examples

---

- hosts: webservers
  roles:
    - common
    - webservers
ROLES

Playbook examples

---
- hosts: webservers
  roles:
    - common
    - { role: myapp, dir: '/opt/a', port: 5000 }
    - { role: myapp, dir: '/opt/b', port: 5001 }
ROLES

Playbook examples

---
- hosts: webservers
  roles:
    - { role: foo, when: "ansible_os_family == 'RedHat'" }
ROLES

Pre and Post - rolling upgrade example

```yaml
---
- hosts: webservers
  serial: 1

pre_tasks:
  - command: lb_rm.sh {{ inventory_hostname }}
    delegate_to: lb

  - command: mon_rm.sh {{ inventory_hostname }}
    delegate_to: nagios

roles:
  - myapp

post_tasks:
  - command: mon_add.sh {{ inventory_hostname }}
    delegate_to: nagios

  - command: lb_add.sh {{ inventory_hostname }}
    delegate_to: lb
```

http://docs.ansible.com/ansible/playbooks_delegation.html
ROLES - INTEGRATION WITH TRAVIS CI

Ansible 2+, magic is in .travis.yml

michaellessard / ansible-role-nginx

Worker information
  Build system information

07 $ export DEBPATH /home/travis/virtualenv/python2.7.9/lib/python2.7/site-packages (python 2.7)
123 $ git clone --depth=50 --branch=master https://github.com/michaellessard/ansible-role-nginx.git michaellessard/ansible-role-nginx
133 Installing APT Packages (BETA)
160 $ source ~/virtualenv/python2.7/bin/activate

162 $ python --version
163 Python 2.7.9
164 $ pip --version
165 pip 6.0.7 from /home/travis/virtualenv/python2.7.9/lib/python2.7/site-packages (python 2.7)
166 $ pip install ansible

playbook: tests/test.yml
The command "ansible-playbook tests/test.yml -i tests/inventory --syntax-check" exited with 0.
Done. Your build exited with 0.
LAB #4
Web server load-balancing over 3 roles

Objectives
1. Create 3 roles: common, apache and haproxy
2. Create a playbook to apply those roles.
   a. “common” should be applied to all servers
   b. “apache” should be applied to your “web” group
   c. “haproxy” should be applied to your “lb” group
3. Your index.html should return the web server name.
4. selinux state should be a set as a variable in group_vars “all”

HAPROXY role available here:
http://people.redhat.com/mlessard/qc/haproxy.tar.gz
LAB4 - File structure

```
.
├── group_vars
│   ├── all
│   │   └── lb
│   └── install.yml
└── roles
    ├── apache
    │   ├── handlers
    │   │   └── main.yml
    │   └── tasks
    │       └── main.yml
    │           └── templates
    │               └── index.html.j2
    └── common
        ├── defaults
        │   └── main.yml
        └── tasks
            └── main.yml
                └── templates
                    └── motd.j2
    └── haproxy
        ├── handlers
        │   └── main.yml
        └── tasks
            └── main.yml
                └── templates
                    └── haproxy.cfg.j2
```
Lab 4 : Example Solution

https://github.com/masauve/ansible-labs
ANSIBLE TOWER
What are the added values?

➔ Role based access control
➔ Push button deployment
➔ Centralized logging & deployment
➔ System tracking
➔ API
ANSIBLE TOWER
20 minutes demo : https://www.ansible.com/tower
THANK YOU

plus.google.com/+RedHat
linkedin.com/company/red-hat
youtube.com/user/RedHatVideos
facebook.com/redhatinc
twitter.com/RedHatNews
# yum install git (required for plug-vim)

$ cd


$ vim .vimrc

    call plug#begin('~/.vim/plugged')
    Plug 'pearofducks/ansible-vim'
    call plug#end()

$ vim

    :PlugInstall

When you edit a file type:

    :set ft=ansible
TRAVIS CI INTEGRATION

Setup

Procedure: https://galaxy.ansible.com/intro
[centos@centos7-1 nginx]$ vim .travis.yml

---
language: python
python: "2.7"

# Use the new container infrastructure
sudo: required

# Install ansible
addons:
  apt:
    packages:
    - python-pip

install:
  # Install ansible
  - pip install ansible

  # Check ansible version
  - ansible --version

  # Create ansible.cfg with correct roles_path
  - printf '[defaults]\nroles_path=../' >ansible.cfg

script:
  # Basic role syntax check
  - ansible-playbook tests/test.yml -i tests/inventory --syntax-check

notifications:
  webhooks: https://galaxy.ansible.com/api/v1/notifications/