ANSIBLE 2

Introduction to Ansible - workshop

Michael Lessard
Sr. Solutions Architect

Martin Sauvé
Sr. Solutions Architect

AGENDA

Ansible Training

1 **Introduction to Ansible**

2 **Ansible commands**
+ LAB

3 **Ansible playbooks**
+ LAB

4 **Ansible variables**
+ LAB

5 **Ansible roles**
+ LAB

6 **Ansible Tower**

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

Simple

AUTOMATE ANYTHING

Can manage almost any *IX through SSH

requires Python 2.4

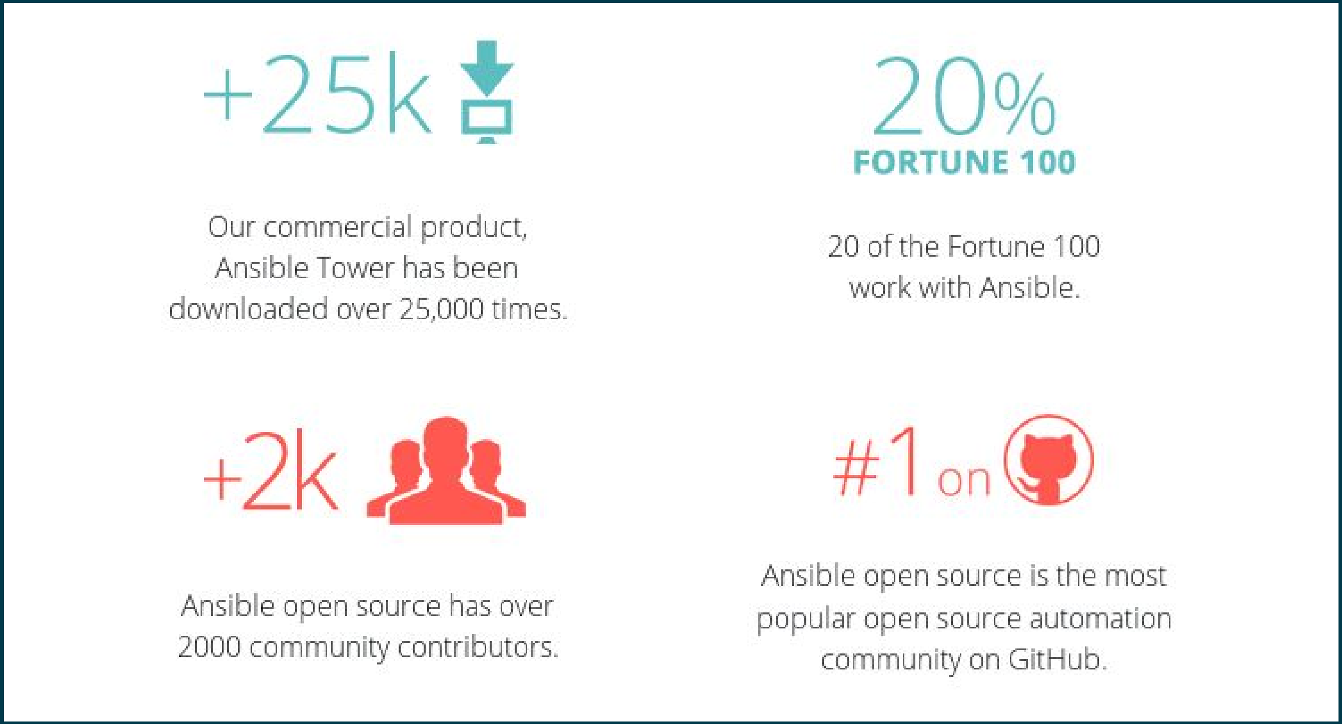
Windows (powershell, winrm python module)

Cloud, Virtualization, Container and Network components

“ 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 growth


+25k

Our commercial product, Ansible Tower has been downloaded over 25,000 times.

20%
FORTUNE 100

20 of the Fortune 100 work with Ansible.

+2k

Ansible open source has over 2000 community contributors.

#1 on

Ansible open source is the most popular open source automation community on GitHub.

" 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

CONFIG MANAGEMENT

Centralizing configuration file management and deployment is a common use case for Ansible, and it's how many power users are first introduced to the Ansible automation platform.

APP DEPLOYMENT

When you define your application with Ansible, and manage the deployment with Tower, teams are able to effectively manage the entire application lifecycle from development to production.

PROVISIONING

Your apps have to live somewhere. If you're PXE booting and kickstarting bare-metal servers or VMs, or creating virtual or cloud instances from templates, Ansible and Ansible Tower help streamline the process.

NETWORK AUTOMATION

Ansible's simple automation framework means that previously isolated network administrators can finally speak the same language of automation as the rest of the IT organization, extending the capabilities of Ansible to include native support for both legacy and open network infrastructure devices.

CONTINUOUS DELIVERY

Creating a CI/CD pipeline requires buy-in from numerous teams. You can't do it without a simple automation platform that everyone in your organization can use. Ansible Playbooks keep your applications properly deployed (and managed) throughout their entire lifecycle.

SECURITY & COMPLIANCE

When you define your security policy in Ansible, scanning and remediation of site-wide security policy can be integrated into other automated processes and instead of being an afterthought, it'll be integral in everything that is deployed.

ORCHESTRATION

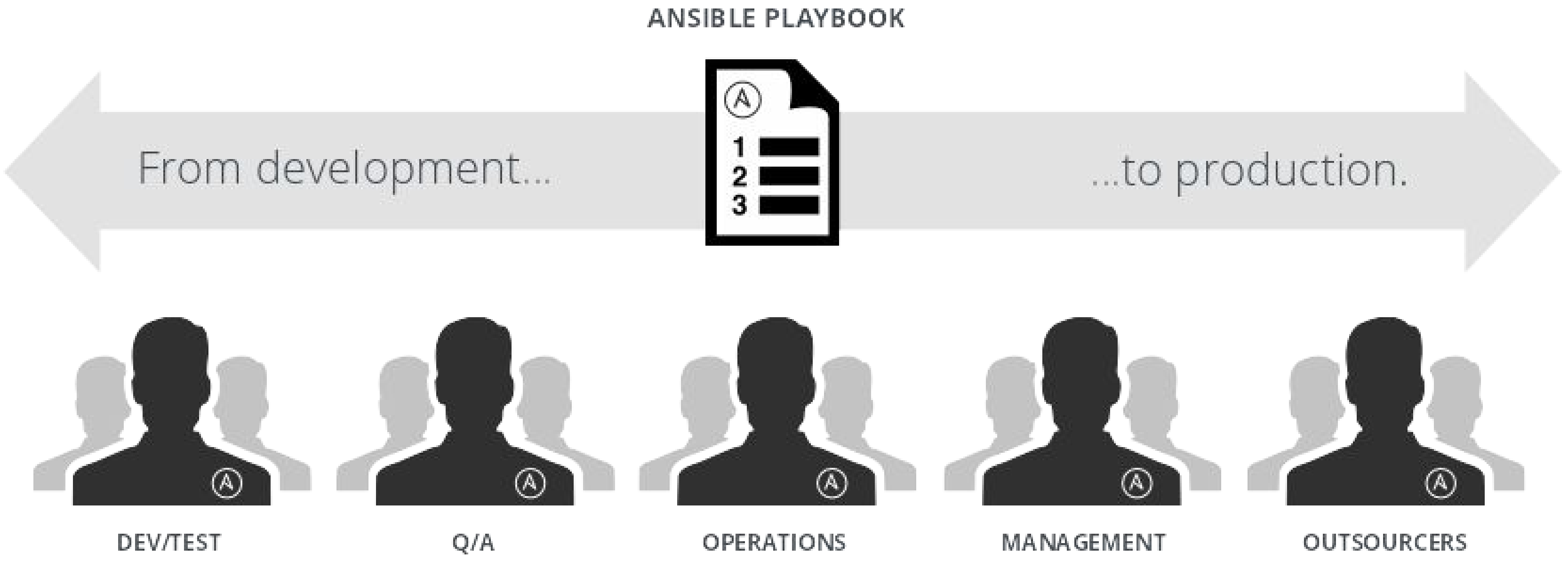
Configurations alone don't define your environment. You need to define how multiple configurations interact and ensure the disparate pieces can be managed as a whole. Out of complexity and chaos, Ansible brings order.

BENEFITS

Why is Ansible popular?

- ➔ Efficient : Agentless, minimal setup, desired state (no unnecessary change), push-Based architecture, Easy Targeting Based on Facts
- ➔ Fast : Easy to learn/to remember, simple declarative language
- ➔ Scalable : Can managed thousands of nodes, extensible and modular
- ➔ 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.

LET'S START !

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
```

https://access.redhat.com/articles/2271461

MODULES

What is this?

Bits of code copied to the target system.
Executed to satisfy the task declaration.
Customizable.

The modules that ship with Ansible all are written in Python, but modules can be written in any language.

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

MODULES

Documentation

```
# 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

[centos@centos1 ~]$ ansible all -i ./hosts -m command -a "uptime"
```

INVENTORY - ALTERNATIVE

```
[centos@centos1 ~]$ cd ansible
[centos@centos1 ansible]$ vim ansible.cfg

[defaults]
inventory=/home/centos/ansible/hosts


[centos@centos1 ~]$ ansible all -m command -a "uptime"
```

COMMANDS

Run your first Ansible command...

```
        (which)            (module)  (arguments)
# 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 (all hosts)

Modules documentation:

http://docs.ansible.com/ansible/list_of_all_modules.html

LAB #1 - SOLUTION

```
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

YAML

1. Designed primarily for the representation of data structures
2. Easy to write, human readable format
3. Design objective : abandoning traditional enclosure syntax

YAML Validator : yamllint.com

PLAYBOOK EXAMPLE

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

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

**** When a variable is used as the first element to start a value, quotes are mandatory.

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
```

PLAYS

Loops

```
- 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

```
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    (will display the 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"
```

BLOCKS

Apply a condition to multiple tasks at once

```
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'
```

ERRORS

Ignoring errors

By default, Ansible stop on errors. Add the ingore_error parameter to skip potential errors.

```
- name: ping host
  command: ping -c1 www.foobar.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.

```
- 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

LAB #2 - SOLUTION #1

```
---
- 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: RestartApache

       - name: Copy Index.html
         copy: src=index.html dest=/var/www/html/index.html

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

  handlers:
       - name: RestartApache
         service: name=httpd state=restarted enabled=yes
       - name: RestartSSH
         service: name=sshd state=restarted enabled=yes
```

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 it's 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

```
- 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
```

```
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 %}
```

JINJA2

LOOPS

```
{% 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

```
vars:
  var1: {{ foo }} <<< ERROR!
  var2: “{{ bar }}”
  var3: Echoing {{ foo }} here is fine
```

Facts

Setting facts in a play

```
# 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 you 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
3. Convert your index.html file into a template to output the following information:

   Web Servers

   centos1 192.168.3.52 - free memory: 337.43 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
```

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

```
---
- 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

http://galaxy.ansible.com

ANSIBLE GALAXY

```
# ansible-galaxy search 'install git' --platforms el
Found 77 roles matching your search:
Name

# ansible-galaxy install davidkarban.git -p roles/

# ansible-galaxy list

# ansible-galaxy remove
```

LAB #4

Convert the lab3 playbook in two roles

Objectives

1. Create 2 roles: common and apache
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
3. Put the jinja2 templates in the appropriate folder.

LAB #4 - SOLUTION 1/3

Create directories using Ansible galaxy template

```
[centos@centos1 ~]$ cd ansible ; mkdir roles ; cd roles
[centos@centos1 roles]$ ansible-galaxy init common ; ansible-galaxy init apache
```

Setup the apache role

```
[centos@centos1 roles]$ cd apache
[centos@centos1 apache]$ vim tasks/main.yml
---
# tasks file for apache
      - 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

[centos@centos1 apache]$ vim handlers/main.yml
---
# handlers file for apache
      - name: Restart Apache
        service: name=httpd state=restarted enabled=yes
[centos@centos1 apache]$ cp ../../index.html.j2 templates/
```

LAB #4 - SOLUTION 2/3

Setup the common role

```
[centos@centos1 apache]$ cd ../common
[centos@centos1 common]$ vim tasks/main.yml
---
# tasks file for common
      - name: Configure selinux to permissive
        selinux:
        policy: targeted
        state: permissive

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

      - name: install facter
        yum: name=facter state=present

[centos@centos1 apache]$ cp ../../motd.j2 templates/
```

LAB #4 - SOLUTION 3/3

Setup the playbook

```
[centos@centos1 common]$ cd ../../
[centos@centos1 lab4]$ vim install.yml
---
- name: Lab4 - All server setup
  hosts: all
  become: yes

  roles:
       - common

- name: Lab4 - Web server setup
  hosts: web
  become: yes

  roles:
       - apache

[centos@centos1 lab4]$ ansible-playbook -i ../hosts install.yml
```

ANSIBLE TOWER

What are the added values ?

- ➔ **Role based access control**
- ➔ **Satellite and Cloudforms integration**
- ➔ **Push button deployment**
- ➔ **Centralized logging & deployment**
- ➔ **Notification to Slack, Twilio, Irc, webooks, ...**
- ➔ **System tracking**
- ➔ **API**

ANSIBLE TOWER

20 minutes demo :

https://www.ansible.com/tower

redhat

THANK YOU

plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHatNews

youtube.com/user/RedHatVideos

FIXING VIM FOR YAML EDITION

```
# yum install git (required for plug-vim)
$ cd
$ curl -fLo ~/.vim/autoload/plug.vim --create-dirs
https://raw.githubusercontent.com/junegunn/vim-plug/master/plug.vim
$ 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
```

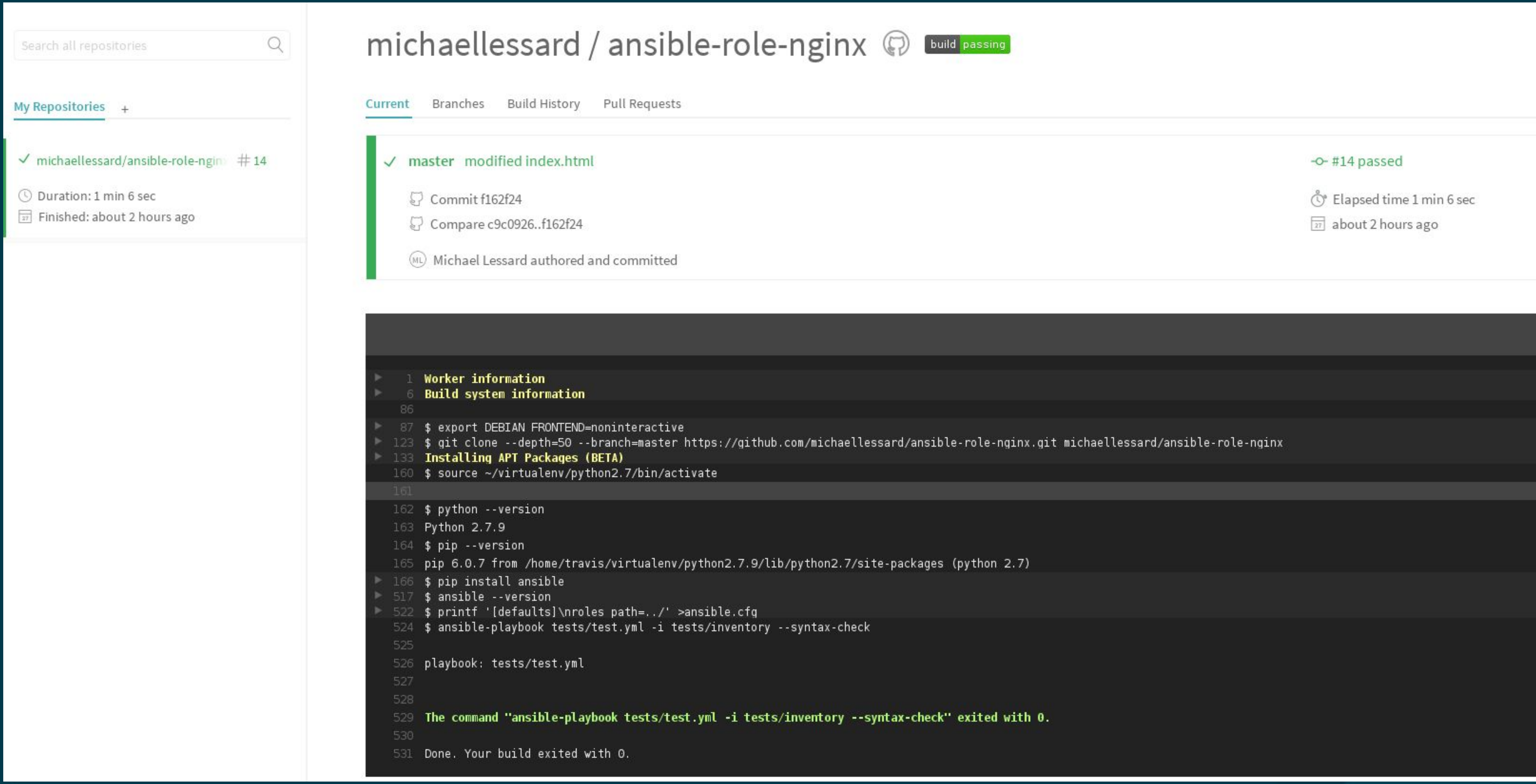
TRAVIS CI INTEGRATION

Setup

Procedure : https://galaxy.ansible.com/intro

ROLES - INTEGRATION WITH TRAVIS CI

Ansible 2+, the magic is in .travis.yml



Search all repositories

My Repositories +

✓ michaellessard/ansible-role-ngin # 14

Duration: 1 min 6 sec
Finished: about 2 hours ago

michaellessard / ansible-role-nginx build passing

Current Branches Build History Pull Requests

✓ master modified index.html

Commit f162f24
Compare c9c0926..f162f24
Michael Lessard authored and committed

#14 passed
Elapsed time 1 min 6 sec
about 2 hours ago

```
Worker information
Build system information

$ export DEBIAN_FRONTEND=noninteractive
$ git clone --depth=50 --branch=master https://github.com/michaellessard/ansible-role-nginx.git michaellessard/ansible-role-nginx
Installing APT Packages (BETA)
$ source ~/virtualenv/python2.7/bin/activate

$ python --version
Python 2.7.9
$ pip --version
pip 6.0.7 from /home/travis/virtualenv/python2.7.9/lib/python2.7/site-packages (python 2.7)
$ pip install ansible
$ ansible --version
$ printf '[defaults]\nroles_path=../' >ansible.cfg
$ ansible-playbook tests/test.yml -i tests/inventory --syntax-check

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.
```

TRAVIS CI INTEGRATION

```
[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/
```