redhat.

ANSIBLE 2.0

Introduction to Ansible training

Marco Berube
sr. Cloud Solution Architect

Michael Lessard
Senior Solutions Architect

Martin Sauvé
Senior Solutions Architect

AGENDA

Ansible Training

1. **Introduction to Ansible**
   + DEMO
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
redhat.

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

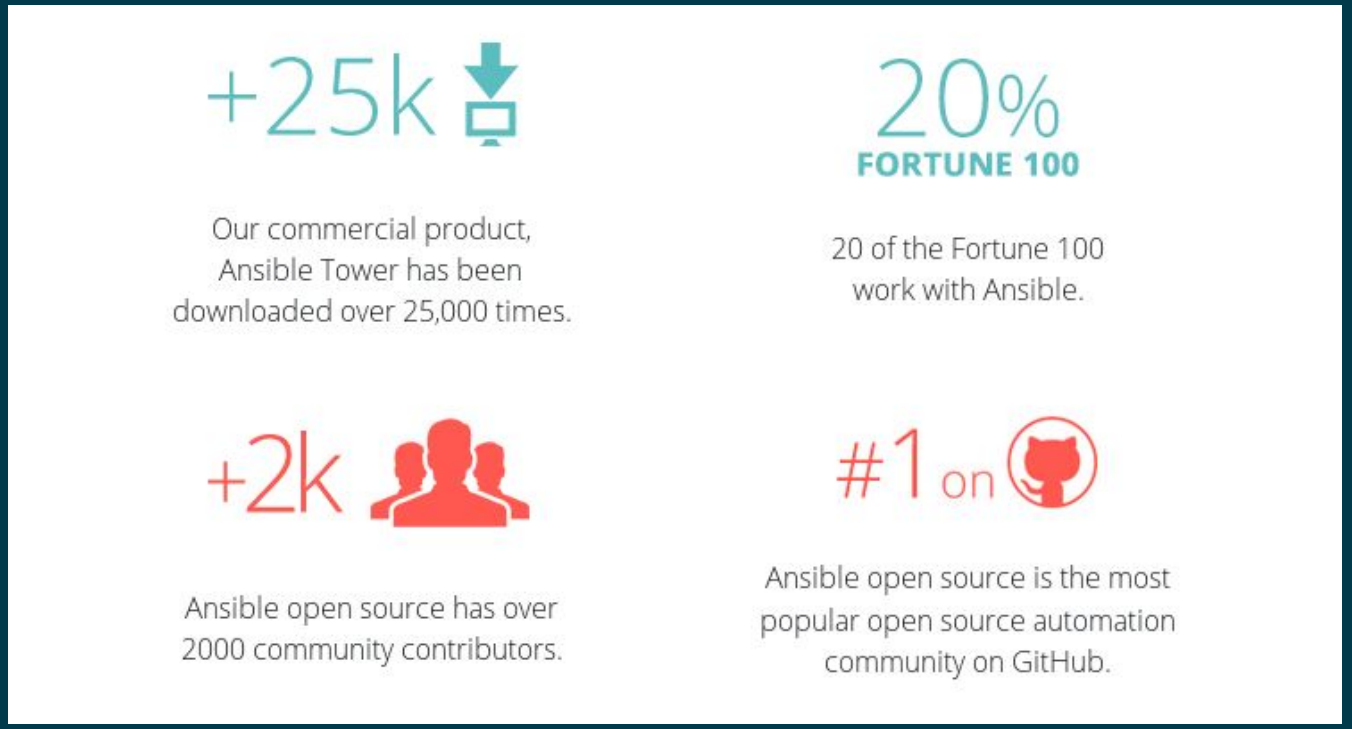
Simple

Can manage almost any *IX through SSH

requires Python 2.4

Windows (powershell, winrm python module)

Ansible growth



“ 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

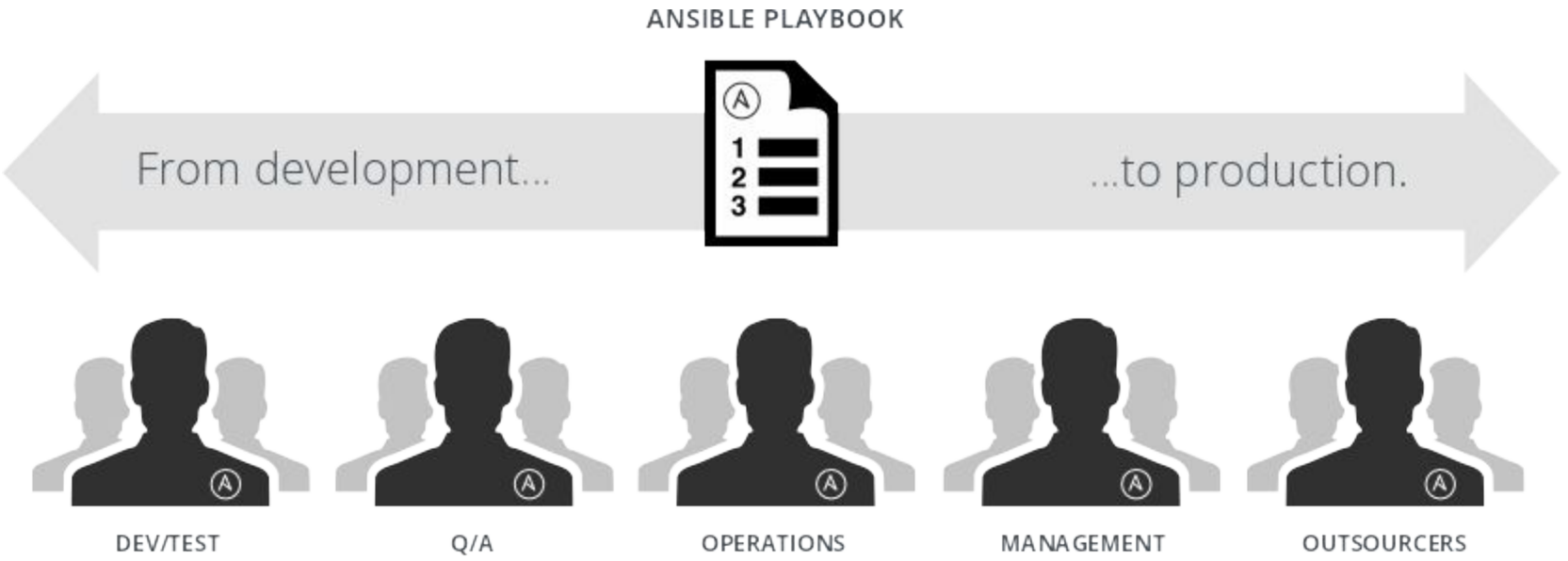
Orchestration

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**

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
```

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

```
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

PLAYBOOK EXAMPLE

```
- 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
```

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
```

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: Restart Apache

    - 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: Restart Apache
      service: name=httpd state=restarted enabled=yes
     - name: RestartSSH
       Service: name=sshd state=restarted enambles=yes
```

LAB #2 - SOLUTION #2

```
# ansible-playbook -i ../hosts lab2.yml -e "selinux=permissive"
```

```
---
- 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 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 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 %}
```

ANSIBLE ROLES

ROLES

A redistributable and reusable collection of:

- **tasks**
- **files**
- **scripts**
- **templates**
- **variables**

ROLES

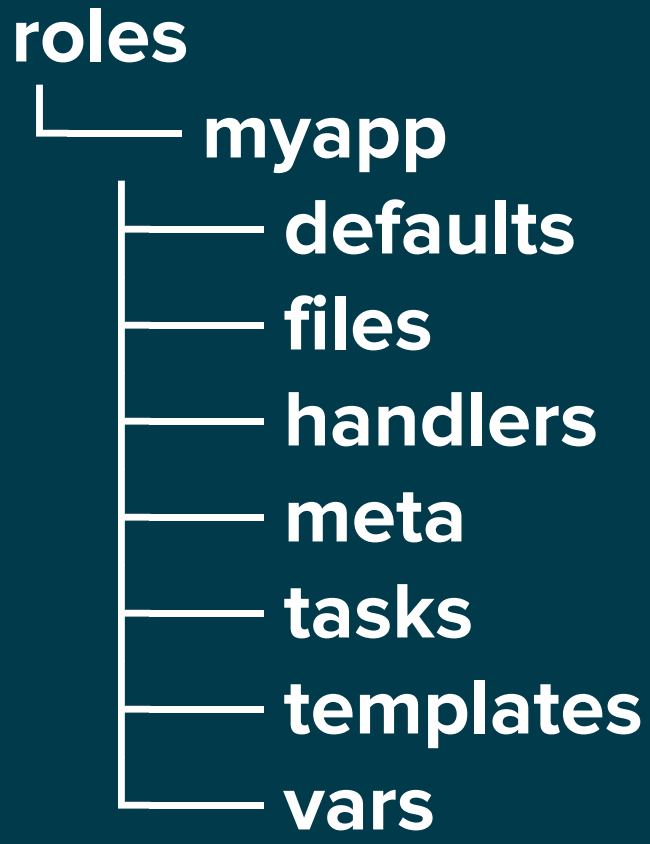
Often used to setup and configure services

- ➔ **install packages**
- ➔ **copying files**
- ➔ **starting deamons**

Examples: Apache, MySQL, Nagios, etc.

ROLES

Directory Structure



ROLES

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

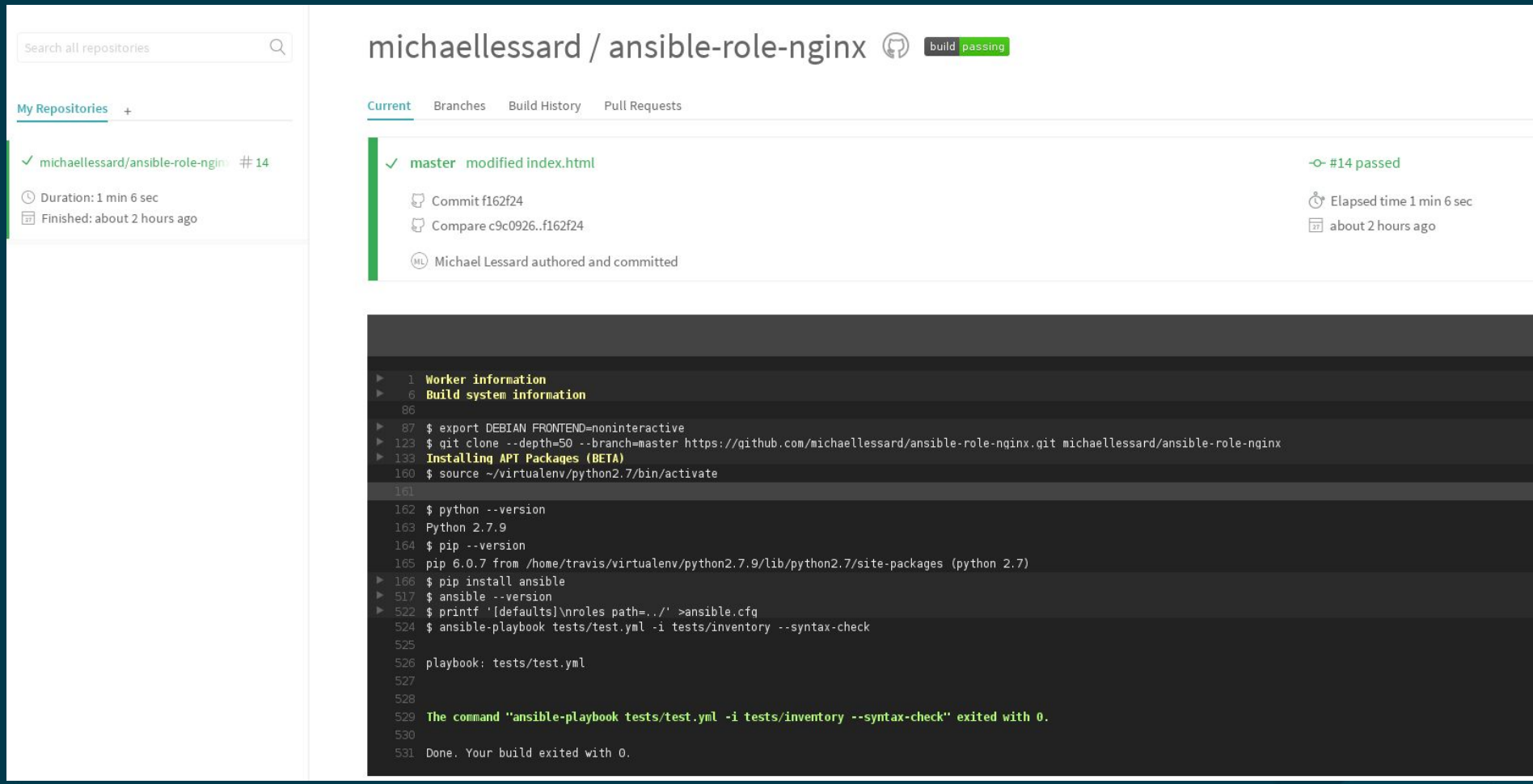
```
---
- hosts: webservers
  serial: 1

  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

ROLES - INTEGRATION WITH TRAVIS CI

Ansible 2+, magic is in .travis.yml



LAB #4

Web server load-balancing over 3 roles

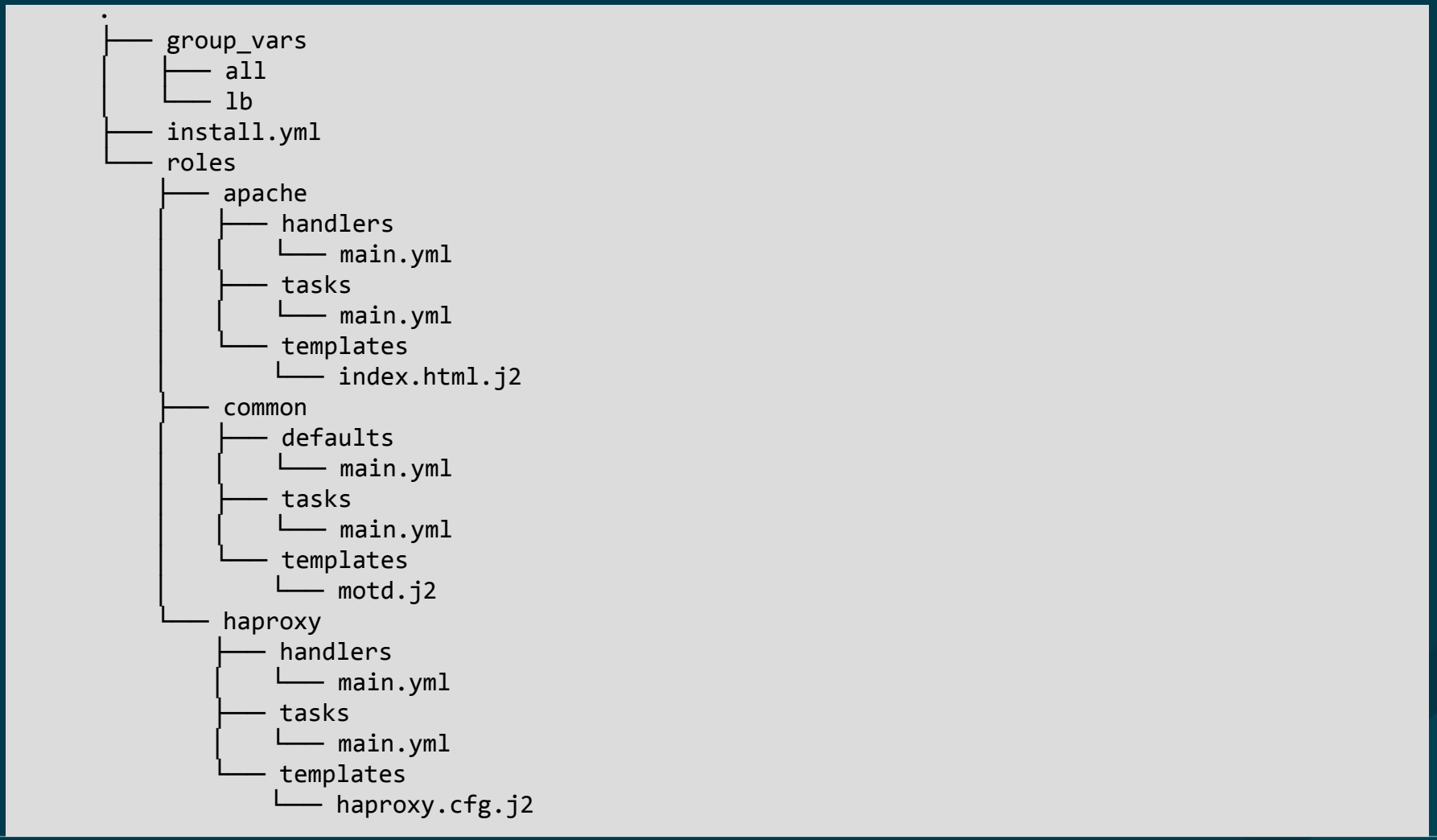
Objectives

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



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**

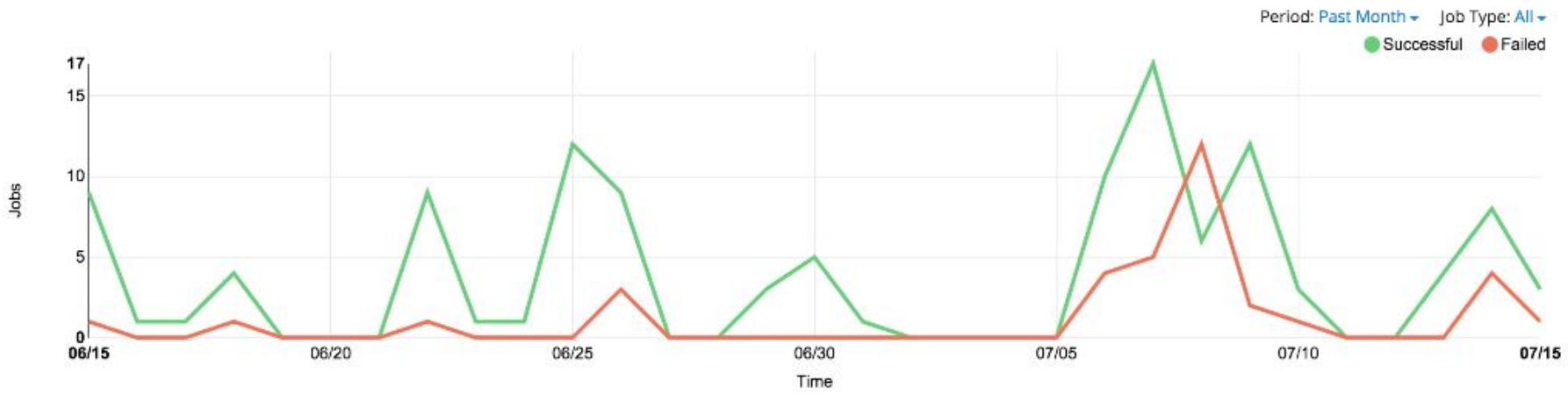
TOWER

Projects | Inventories | Job Templates | Jobs

admin

59	14	9	0	16	1
Hosts	Failed Hosts	Inventories	Inventory Sync Failures	Projects	Projects Sync Failures

Job Status | Host Status



Recently Used Job Templates

- Scan AWS Hosts
- Z - Launch LAMP demo instances to AWS
- Z - Terminate AWS Demo Instances
- Linux - Deploy Splunk
- Z - Launch AWS Demo Instances

See all job templates

Recent Job Runs

Scan AWS Hosts	28 minutes ago
Z - Launch LAMP demo instances to AWS	39 minutes ago
Z - Terminate AWS Demo Instances	an hour ago
Linux - Deploy Splunk	20 hours ago
Linux - Deploy Splunk	20 hours ago

See all job runs

Status running

Started 08/18/14 14:36:24

more

Plays

Play Name | All | Failed

Started	Elapsed	Status	Name
14:36:24	00:00:11	●	all
14:36:36	00:00:11	●	dbservers
14:36:47	00:00:07	●	webservers

Tasks

Task Name | All | Failed

Started	Elapsed	Status	Name	Host Status
14:36:47	00:00:00	●	Gathering Facts	2
14:36:48	00:00:06	●	Install http and php...	2
14:36:54	00:00:00	●	http service state	2
14:36:54		●	Install php and git	

Host Events

Host Name | All | Failed

Status	Host	Item	Message
●	web1		
●	web2		

Summary

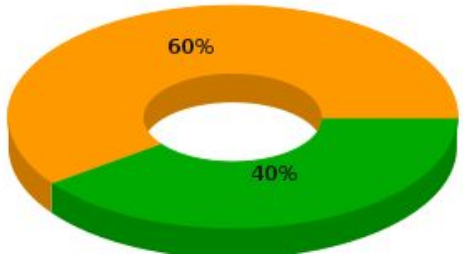
Host Name | All | Failed

● OK ● Changed ● Unreachable ● Failed

Host	Completed Tasks
db1	12 5
lb1	10
nagios	10
web1	11 2
web2	11 2

Status Summary

● OK ● Changed ● Unreachable ● Failed



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

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/
```