



Automation with Ansible

Hands on workshop

Delivered by :

Khizer Naeem

Senior Systems Engineer
Professional Cloud Architect
Devops Enthusiast

khizernaem@gmail.com

Prerequisites

1. A PC/Laptop with Linux/MAC/Windows
2. Internet Connected
3. SSH Client e.g, Putty, Terminal
4. Web browser
5. Basic knowledge of Linux and command line
6. Functional knowledge of a Linux text editor

Agenda

1. Introduction to Ansible
2. Installing Ansible
3. Ansible Components
4. Real world deployment I
5. Ansible Advance topics
6. Real world deployment II
7. Ansible Roles
8. Real world deployment III

Introduction to Ansible

What is Ansible?

- Automation
- Change Management
- Provisioning
- Orchestration

Automation

- Core of Ansible
- Run tasks
 - Update a software package
 - Create a user
 - Open/Close ports
- Conditions
- Scale

Change Management

- System State
 - Define
 - Enforce
 - Example
 - Apache web server version 2.4.x installed
 - PHP 5.4.x installed
 - Apache web server started
 - webadmin user exist with authorized key
 - Deviation from the state would warrant a change
 - Ansible operations are Idempotent

Provisioning

- Built on top of Automation and Change Management
- Preparing a system
- Installing, updating, configuring software
- For Example:
 - Start with a basic installation of OS
 - Update the operating system
 - Install the web server
 - Deploy the application
 - Configure the application
 - Start the web server

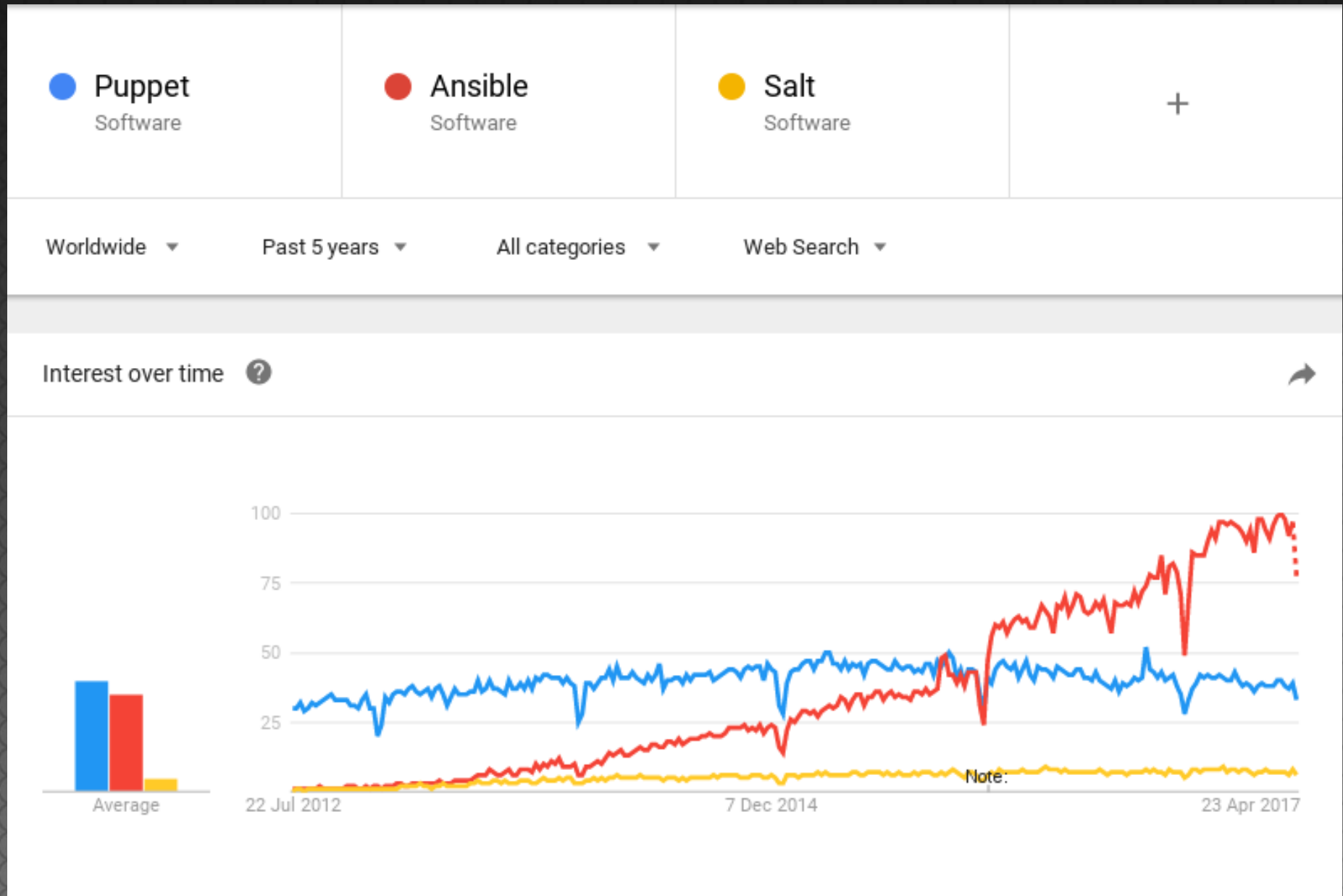
Orchestration

- Orchestration is not Automation
- Coordination between systems
- Order sensitive tasks
- For example:
 - Remove web1 from LB
 - Run tasks on web1
 - Add web1 to LB
 -

Why Ansible?

- Simple and Lightweight
 - No Agents
 - No database
- Multi-platform
 - Windows, Linux, Unix, Mac ..
- YAML
- Built-in Security
- Extendable

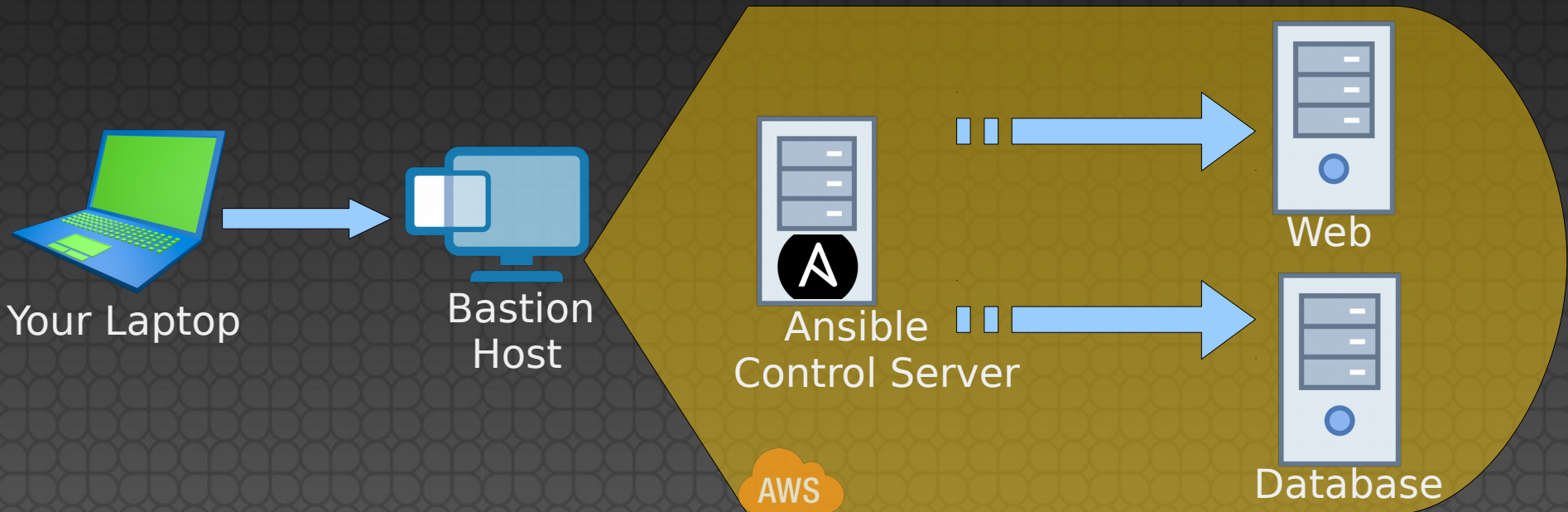
Why Ansible?



Installing Ansible

Lab Environment

- You will be assigned a group number. 10 - 99
- Substitute your group number with XX in the table below
- SSH into the bastion host to access your environment



Server	IP Address	User	Password
Bastion Host	lab.kxr.me	labXX	Lab3nvXX
acs.labXX	10.0.XX.10	root	ansibleXX
webserver.labXX	10.0.XX.11	root	webserverXX
database.labXX	10.0.XX.12	root	databaseXX

Setup your laptop

- Your laptop should be connected to the internet
- Use your favorite SSH client on you laptop (e.g. Putty)
- Create 3 separate connections to the bastion host for each node: acs, webserver, database
- Each node is having a basic installation of CentOS 7
- Make sure you are on the correct nodes
- The color of the prompt should help you identify each node:

```
[root@acs ~]
```

```
[root@webserver ~]
```

```
[root@database ~]
```


Installing Ansible

On the Control Server `[root@acs ~]`

```
yum -y update
```

```
yum -y install epel-release
```

```
yum -y install ansible
```

- Ansible is installed (that simple!)
- Ansible Configuration: `/etc/ansible/ansible.cfg`
- Default inventory: `/etc/ansible/host`
- Easily refer to the documentation:

```
ansible-doc -l
```

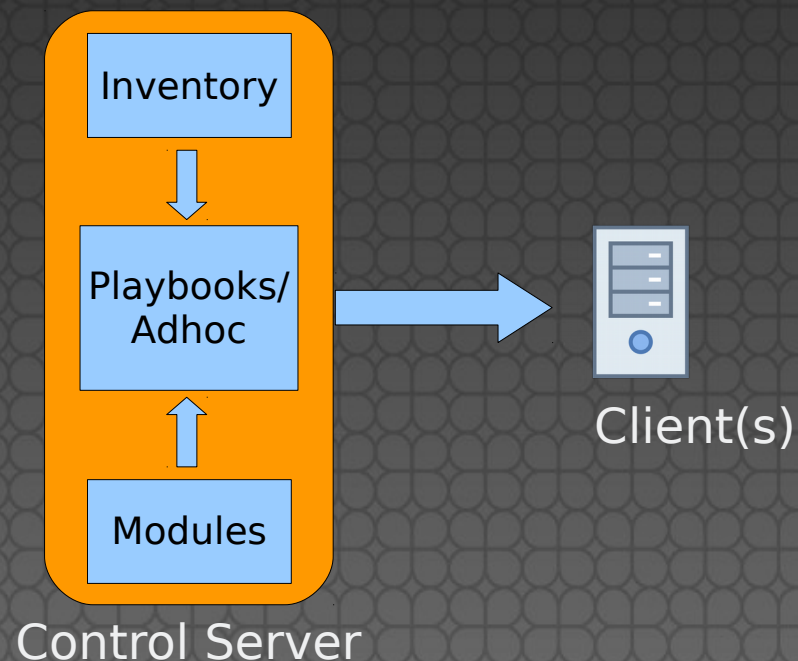
```
ansible-doc <module>
```

```
ansible-doc -s <module>
```

Ansible Components

Ansible Components

- Architecture
 - Control Server → Client
 - Gathers facts from clients
- Control Server
 - Inventory
 - Modules
 - Playbooks
- Client
 - SSH
 - WinRM



Inventory

- Default: /etc/ansible/hosts
- Custom inventory using -i switch
 - Use custom inventories to isolate environments e.g. Prod, dev, US
- Hosts, Groups, Variables
- Default Groups: all, ungrouped
- For example:

```
mail.example.com
```

```
[webservers]
```

```
one.example.com
```

```
alpha.example.com ansible_host=192.0.2.50
```

```
[dbservers]
```

```
one.example.com
```

```
two.example.com
```

```
three.example.com ansible_host=192.0.2.99
```

Setup inventory

- /etc/ansible/hosts
- Make sure the hostnames are reachable

```
yum -y install nano vim emacs
```

```
cd /etc/ansible
```

```
<editor> hosts
```

```
[app]  
webserver.labXX  
database.labXX
```

```
[webservers]  
webserver.labXX
```

```
[dbservers]  
database.labXX
```

Modules

- Ansible ships with ~500 Modules
 - You can write your own!
- Each module is automating a task for you.
- Modules for configuring network devices.
- Module Support (Read the docs)
 - Core
 - Curated
 - Community
- Lets see some in action

Ansible Ad-hoc commands

- Running quick task
- Inventory, Module, Arguments
 - `ansible <inventory> -m <module> -a <arguments>`
- Examples:
 - add client finger prints to known_hosts
 - Use `-k` or `--ask-pass` to be prompted for password

```
ansible webserver.labXX -m ping
```

```
ansible webserver.labXX -a "ip addr"
```

```
ansible webserver.labXX -a "w"
```

```
ansible webserver.labXX -m yum -a "name=vim state=present"
```

Authorize ssh

- Lets generate our ssh key:

```
ssh-keygen
```

- Authorize our key:

```
ssh-copy-id webserver.labXX
```

```
ssh-copy-id webserver.labXX
```

- We can use ansible :)

```
ansible webserver.labXX -m authorized_key -a \
"user=root key={{lookup('file', '/root/.ssh/id_rsa.pub')}}" -k
```

```
ansible database.labXX -m authorized_key -a \
"user=root key={{lookup('file', '/root/.ssh/id_rsa.pub')}}" -k
```

Commonly used Modules

- setup

```
ansible webserver.labXX -m setup  
-a "filter=ansible_eth*"
```

- yum, apt
- copy, fetch
- hostname, timezone, service
- user, authorized_key
- template, file, lineinfile

Targeting hosts and groups

- OR group1:group2

```
ansible webservers:dbservers -m ping
```

- AND group1:&group2

```
ansible 'app:&dbservers' -m ping
```

- NOT !group1

```
ansible 'app:!dbservers' -m ping
```

- Combination group1:&group2:!group3

- Wildcard and Regex

```
ansible *.lab* -m ping
```

```
~web[0-9] +
```

Playbooks

- Written in YAML
 - Watch the whitespaces!
- Playbooks: Collection of Plays
 - Plays: Collection of tasks
 - Tasks: Collection of modules
- Sequential order of execution
- Stops further execution on failure
 - `ignore_errors: yes`
 - `retry` file for failed hosts
- You can include other playbooks

Playbooks - Example

```
---
- hosts: webservers

  tasks:
    - name: Install Apache Webserver
      yum: name=httpd state=present

    - name: Start Apache Webserver
      service: name=httpd state=started enabled: yes

- hosts: dbservers

  tasks:
    - name: Install MariaDB Server
      yum: name=mariadb-server state=present

    - name: Start MariaDB Server
      service: name=mariadb-server state=started enabled: yes
```


Playbooks - Example

- hosts: webservers

tasks:

- name: Install Apache Webserver
yum: name=httpd state=present

- name: Start Apache Webserver
service: name=httpd state=started enabled: yes

- hosts: dbservers

tasks:

- name: Install MariaDB Server
yum: name=mariadb-server state=present

- name: Start MariaDB Server
service: name=mariadb-server state=started enabled: yes

Playbooks - Example

- hosts: webservers

tasks:

- name: Install Apache Webserver
yum: name=httpd state=present

- name: Start Apache Webserver
service: name=httpd state=started enabled: yes

- hosts: dbservers

tasks:

- name: Install MariaDB Server
yum: name=mariadb-server state=present

- name: Start MariaDB Server
service: name=mariadb-server state=started enabled: yes

Playbooks – Example

```
---  
- hosts: all  
  
  tasks:  
  
    - name: Disable SELinux  
      selinux:  
        state: disabled  
  
    - name: Reboot  
      command: /sbin/reboot
```

```
cd /etc/ansible
```

```
cp /opt/workshop/examples/disable_selinux_reboot.yml .
```

```
ansible-playbook disable_selinux_reboot.yml --check
```


Real World Deployment I

Deployment Objectives

- Common
 - Disable selinux
 - Create a standard directory
 - Install vim
- Webserver
 - Install apache webserver
 - create webadmin user
- Database
 - Install mariadb database server
 - create dbadmin user

Deployment I – Play 1

```
- hosts: all
  tasks:
    - name: Disable SELinux
      selinux:
        state: disabled
    - name: Create MyFiles Directory
      file:
        path: /root/MyFiles
        state: directory
        owner: root
        group: root
        mode: 0755
    - name: Install Vim
      yum:
        name: vim
        state: present
```

Deployment I – Play 2

```
- hosts: webservers
  tasks:
    - name: Install Apache Webserver
      yum:
        name: httpd
        state: present

    - name: Start Apache Webserver
      service:
        name: httpd
        state: started
        enabled: yes

    - name: Create webadmin user
      user:
        name: webadmin
        comment: "Web Admin User"
        groups: apache
```


Deployment I – Play 3

```
- hosts: dbservers
  tasks:
  - name: Install MariaDB Server
    yum:
      name: mariadb-server
      state: present

  - name: Start MariaDB Server
    service:
      name: mariadb-server
      state: started
      enabled: yes

  - name: Create dbadmin user
    user:
      name: dbadmin
      comment: "DB Admin User"
      groups: mysql
```

Deployment I

- Copy and run the playbook from /opt/workshop/rwd1

```
[root@acs ~]
```

```
cd /etc/ansible
```

```
cp /opt/workshop/rwd1/playbook1.yml
```

```
ansible-playbook playbook1.yml --check
```

```
ansible-playbook playbook1.yml
```

- Test services on both nodes

Playbook Output

```
PLAY [all] *****
```

```
TASK [Gathering Facts] *****
```

```
ok: [webserver.lab10]
```

```
ok: [database.lab10]
```

```
TASK [Disable SELinux] *****
```

```
changed: [webserver.lab10]
```

```
changed: [database.lab10]
```

```
.. ..
```

```
PLAY [webservers] *****
```

```
TASK [Gathering Facts] *****
```

```
ok: [webserver.lab10]
```

```
TASK [Install Apache Webserver] *****
```

```
changed: [webserver.lab10]
```

```
.. ..
```

```
PLAY RECAP *****
```

```
database.lab10      : ok=8    changed=6    unreachable=0    failed=0
```

```
webserver.lab10     : ok=8    changed=6    unreachable=0    failed=0
```

[Break]
Questions?

Agenda

1. Introduction to Ansible
2. Installing Ansible
3. Ansible Components
4. Real world deployment I
5. Ansible Advance topics
6. Real world deployment II
7. Ansible Roles
8. Real world deployment III

Ansible Advance Topics

Ansible Advance Topics

- Variables
- Conditions
- Handlers
- Loops
- Templates
- Includes

Variables

- Facts

```
ansible webserver -m setup
```

- Magic Variables

- hostvars, group_names, groups

- Variables Defined in:

- Inventory
 - Playbook
 - Include files
 - Roles

Variables

- Variables in inventory

```
webserver.labXX ansible_port=2992 ansible_host=1.2.3.4
```

```
webserver.labXX http_port=80 maxRequestsPerChild=100
```

```
[app]
webserver.labXX
database.labXX
```

```
[webservers]
webserver.labXX
```

```
[dbservers]
database.labXX
```

```
...
```

```
...
[app:vars]
ntp_server=1.2.3.4

[webservers:vars]
http_port=80
htdocs=/var/www/html

[dbservers:vars]
mariadb_port=3306
db_user = dbadmin
```

- Inventory variables in files:

- /etc/ansible/host_vars/webserver.labXX.yml
- /etc/ansible/group_vars/app.yml

Variables

- Variables in playbook

```
- hosts: webservers
```

```
vars:
```

```
  http_port: 80
```

```
  htdocs: /var/www/html
```

```
tasks:
```

```
  - name: Blah blah
```

```
    module:
```

```
      ...
```

Variables

- Register variables

```
- hosts: webservers
```

```
tasks:
```

```
  - name: Run shell script  
    shell: /root/script.sh  
    register: script_output  
    ...
```

Conditions

- When Statement

```
- hosts: webservers
```

```
tasks:
```

- name: Run shell script
yum: name=httpd state=present
when: ansible_os_family == "RedHat"
- name: Run shell script
apt: name=apache2 state=present
when: ansible_os_family == "Debian"

```
...
```


Conditions

- “When” on Register variables

```
- hosts: all

tasks:
  - name: Check apache vhost conf file
    stat:
      path: /etc/httpd/conf.d/app.conf
      register: appconf

  - name: Copy appconf file
    copy:
      src: /opt/application/apache/app.conf
      dest: /etc/httpd/conf.d/app.conf
      when: not appconf.stat.exists

  - name: Restart Apache
    service:
      name: httpd
      state: restarted
```

Handlers

- Running Operations On Change

```
- hosts: all

tasks:
  - name: Check apache vhost conf file
    stat:
      path: /etc/httpd/conf.d/app.conf
      register: appconf

  - name: Copy appconf file
    copy:
      src: /opt/application/apache/app.conf
      dest: /etc/httpd/conf.d/app.conf
    when: not appconf.stat.exists
    notify: Restart Apache

handlers:
  - name: Restart Apache
    service:
      name: httpd
      state: restarted
```

Loops

- Standard Loops using “with_items:”

```
- hosts: all
```

```
tasks:
```

```
- name: Add user user1
```

```
  user:
```

```
    name: "user1"
```

```
    state: present
```

```
    groups: "wheel"
```

```
- name: Add user user2
```

```
  user:
```

```
    name: "user2"
```

```
    state: present
```

```
    groups: "wheel"
```

```
- hosts: all
```

```
tasks:
```

```
- name: add users user1 and 2
```

```
  user:
```

```
    name: "{{ item }}"
```

```
    state: present
```

```
    groups: "wheel"
```

```
with_items:
```

```
- user1
```

```
- user2
```

Loops

- File iteration using “with_file”

```
- hosts: all
```

```
tasks:
```

```
- name: Copy app.php
```

```
  copy:
```

```
    src: /opt/app/app.php
```

```
    dest: /var/www/html/
```

```
    owner: apache
```

```
    mode: 600
```

```
- name: Copy config.php
```

```
  copy:
```

```
    src: /opt/app/config.php
```

```
    dest: /var/www/html/
```

```
    owner: apache
```

```
    mode: 600
```

```
- hosts: all
```

```
tasks:
```

```
- name: Copy app files
```

```
  copy:
```

```
    src: “{{ item }}”
```

```
    dest: /var/www/html/
```

```
    owner: apache
```

```
    mode: 600
```

```
  with_file:
```

```
    - “/opt/app/app.php”
```

```
    - “/opt/app/config.php”
```


Loops

- File iteration using “with_fileglob”

```
- hosts: all
```

```
tasks:
```

- ```
- name: Copy app.php
```

```
 copy:
```

```
 src: /opt/app/app.php
```

```
 dest: /var/www/html/
```

```
 owner: apache
```

```
 mode: 600
```

- ```
- name: Copy config.php
```

```
  copy:
```

```
    src: /opt/app/config.php
```

```
    dest: /var/www/html/
```

```
    owner: apache
```

```
    mode: 600
```

```
- hosts: all
```

```
tasks:
```

- ```
- name: Copy app files
```

```
 copy:
```

```
 src: “{{ item }}”
```

```
 dest: /var/www/html/
```

```
 owner: apache
```

```
 mode: 600
```

```
 with_fileglob:
```

- ```
    - “/opt/app/*.php”
```

Templates

- Ansible uses jinja2 templating engine
- Template modules
 - Similar to copy module
 - Replaces the variables
 - Can contain loops and conditions
- Check the official Jinja2 docs:
<http://jinja.pocoo.org/docs/2.9/>

Templates

- Jinja2 Basics
 - `{% ... %}` for Statements
 - `{{ ... }}` for Expressions
 - `{# ... #}` for Comments
- Variables
 - `{{ foo.bar }}`
- Filters
 - `{{ htmldata | striptags | title }}`
 - `{{ list | join(', ') }}`

Templates

- Example: ntp.conf.j2

```
driftfile /var/lib/ntp/drift

restrict 127.0.0.1
restrict -6 ::1

server {{ ntpserver }}

includefile /etc/ntp/crypto/pw

keys /etc/ntp/keys
```


Templates

- Example: my.cnf.j2

```
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
user=mysql
# Disabling symbolic-links is recommended to
prevent assorted security risks
symbolic-links=0
port={{ mysql_port }}

[mysqld_safe]
log-error=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid
```

Templates

- Using the templates

```
...  
- name: Configure ntp file  
  template:  
    src: ntp.conf.j2  
    dest: /etc/ntp.conf  
    notify: Restart ntp
```

```
...  
- name: Configure MariaDB  
  template:  
    src: my.cnf.j2  
    dest: /etc/my.cnf  
    notify: restart mariadb
```

```
...
```

Includes

- Break up bits of configuration policy into smaller files
- Simplify, organize and reuse plays
- Task includes
 - Inclusions under the “tasks” directive
- Play includes
 - Inclusions along the same level of tasks
- You can pass variables when calling the include statement
 - One “template” playbook with variables can be used multiple times with different variables
- Example:

Includes

tasks/common.yml

- name: Disable SELinux
selinux:
state: disabled
- name: Install Vim
yum:
name: vim
state: present

tasks/httpd.yml

- name: Install httpd
yum:
name: httpd
state: present
- name: Start httpd
service:
name: httpd
state: started
enabled: yes

tasks/mariadb.yml

- name: Install mariadb
yum:
name: mariadb-server
state: present
- name: Start mariadb
service:
name: mariadb
state: started
enabled: yes

- hosts: all
tasks:
 - include tasks/common.yml
- hosts: webservers
tasks:
 - include: tasks/httpd.yml
- hosts: dbservers
tasks:
 - include: tasks/httpd.yml

Real World Deployment – II

Deployment Objectives

- Common
 - Disable selinux
 - Create some standard directories
 - Setup NTP service and timezone
 - Install vim, screen, tcpdump, mysql, wget
- Webserver
 - Install apache webserver, with vhost configuration
 - create webadmin user
 - Deploy a simple one page application
- Database
 - Install mariadb database server, with configuration
 - create dbadmin user
 - Configure database for the application

Deployment

```
cd /etc/ansible
```

```
cp -av /opt/workshop/rwd2 .
```

```
cd rwd2
```

- Templates
 - index.php.j2: A simple web app
 - my.cnf.j2: mariadb configuration
 - ntp.cnf.j2: ntp server configuration
 - vhost.conf.j2: Apache configuration
- Includes
 - common.yml, webserver.yml, database.yml
- Playbook2.yml
 - `ansible-playbook playbook2.yml --check`
 - `ansible-playbook playbook2.yml`

```
rwd2/  
├── templates/  
│   ├── index.php.j2  
│   ├── my.cnf.j2  
│   ├── ntp.conf.j2  
│   └── vhost.conf.j2  
├── common.yml  
├── database.yml  
├── playbook2.yml  
└── webserver.yml
```

[Break]
Questions?

Agenda

1. Introduction to Ansible
2. Installing Ansible
3. Ansible Components
4. Real world deployment I
5. Ansible Advance topics
6. Real world deployment II
7. Ansible Roles
8. Real world deployment III

Ansible Roles

Ansible Roles

- Best way to organize your playbooks
- Includes on steroids
 - no additional magic except directory structure and search path handling
- Special directory structure
 - You don't need to have all the directories
 - Only what you need
 - A simple role will only have tasks/main.yml
- main.yml
 - can have include files beside it

```
examplerole/  
├── defaults/  
│   └── main.yml  
├── files/  
│   └── app.zip  
├── handlers/  
│   └── main.yml  
├── meta/  
│   └── main.yml  
├── tasks/  
│   └── main.yml  
├── templates/  
│   └── conf.ini.j2  
└── vars/  
    └── main.yml
```


Ansible Roles – Rules

- If tasks/main.yml exists, tasks listed therein will be added to the play.
- If handlers/main.yml exists, handlers listed therein will be added to the play.
- If vars/main.yml exists, variables listed therein will be added to the play.
- If defaults/main.yml exists, variables listed therein will be added to the play.
- If meta/main.yml exists, any role dependencies listed therein will be added.
- Any copy, script, template or include tasks in the role can reference files in: files, templates, tasks without having to path them relatively or absolutely

```
examplerole/  
├── defaults/  
│   └── main.yml  
├── files/  
│   └── app.zip  
├── handlers/  
│   └── main.yml  
├── meta/  
│   └── main.yml  
├── tasks/  
│   └── main.yml  
├── templates/  
│   └── conf.ini.j2  
└── vars/  
    └── main.yml
```


Ansible Roles – Example

```
# roles/memcached/tasks/main.yml
```

- name: Install memcached
yum:
 name: memcached
 state: present
- name: Enable memcached service
service:
 name: memcached
 state: started

```
# roles/httpd/tasks/main.yml
```

- name: Install httpd
yum:
 name: httpd
 state: present
- name: Enable httpd service
service:
 name: httpd
 state: started

```
# playbook.yml
```

- hosts: webservers

 roles:
 - memcached
 - httpd

```
roles/  
├── httpd/  
│   └── tasks/  
│       └── main.yml  
└── memcached/  
    └── tasks  
        └── main.yml
```

Ansible Roles – Example

```
# roles/apache2/tasks/main.yml
```

- name: Install apache2
yum:
 name: apache2
 state: present
- name: Enable apache2 service
service:
 name: apache2
 state: started

```
# roles/httpd/tasks/main.yml
```

- name: Install httpd
yum:
 name: httpd
 state: present
- name: Enable httpd service
service:
 name: httpd
 state: started

```
# playbook.yml
```

- hosts: webservers
- roles:
- { role: httpd, when: ansible_distribution == 'RedHat' }
 - { role: apache2, when: ansible_distribution == 'Debian' }

```
roles/  
├── httpd/  
│   └── tasks/  
│       └── main.yml  
└── apache2/  
    └── tasks  
        └── main.yml
```

Ansible Galaxy

- Free repository of community developed roles
- You can also use the site to share roles that you create
- Uses github authentication
- You can deploy your own internal Galaxy server
- Installing Roles

```
ansible-galaxy install username.role_name
```

- By default installs to /etc/ansible/roles
- A good reference point for writing your own

Ansible Roles

- Let us do the same deployment again, this time with roles
- Hardly any change in the code
- Three Roles
 - common
 - webserver
 - database
- Variable appname set in playbook
- Variables set in roles

Real World Deployment – II

Deployment

- Roles and Playbook are available:

/opt/workshop/rwd3/common

/opt/workshop/rwd3/webserver

/opt/workshop/rwd3/database

/opt/workshop/rwd3/playbook3.yml

- Copy the three roles

- `cd /etc/ansible`

- `cp -r /opt/workshop/rwd3/common roles/`

- `cp -r /opt/workshop/rwd3/webserver roles/`

- `cp -r /opt/workshop/rwd3/database roles/`

- Copy and execute the playbook

- `cp /opt/workshop/rwd3/playbook3.yml .`

- `ansible-playbook playbook3.yml`

```
roles/
├── common
│   ├── tasks
│   │   └── main.yml
│   ├── templates
│   │   └── ntp.conf.j2
│   └── vars
│       └── main.yml
├── database
│   ├── handlers
│   │   └── main.yml
│   ├── tasks
│   │   └── main.yml
│   ├── templates
│   │   └── my.cnf.j2
│   └── vars
│       └── main.yml
└── webserver
    ├── handlers
    │   └── main.yml
    ├── tasks
    │   └── main.yml
    ├── templates
    │   ├── index.php.j2
    │   └── vhost.conf.j2
    └── vars
        └── main.yml
```

The End

Questions?

Contact:
khizernaeeem@gmail.com