Automation with Ansible

Hands on workshop

Delivered by :

Khizer Naeem

Senior Systems Engineer
Professional Cloud Architect
Devops Enthusiast

khizernaeem@gmail.com

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
  - ....

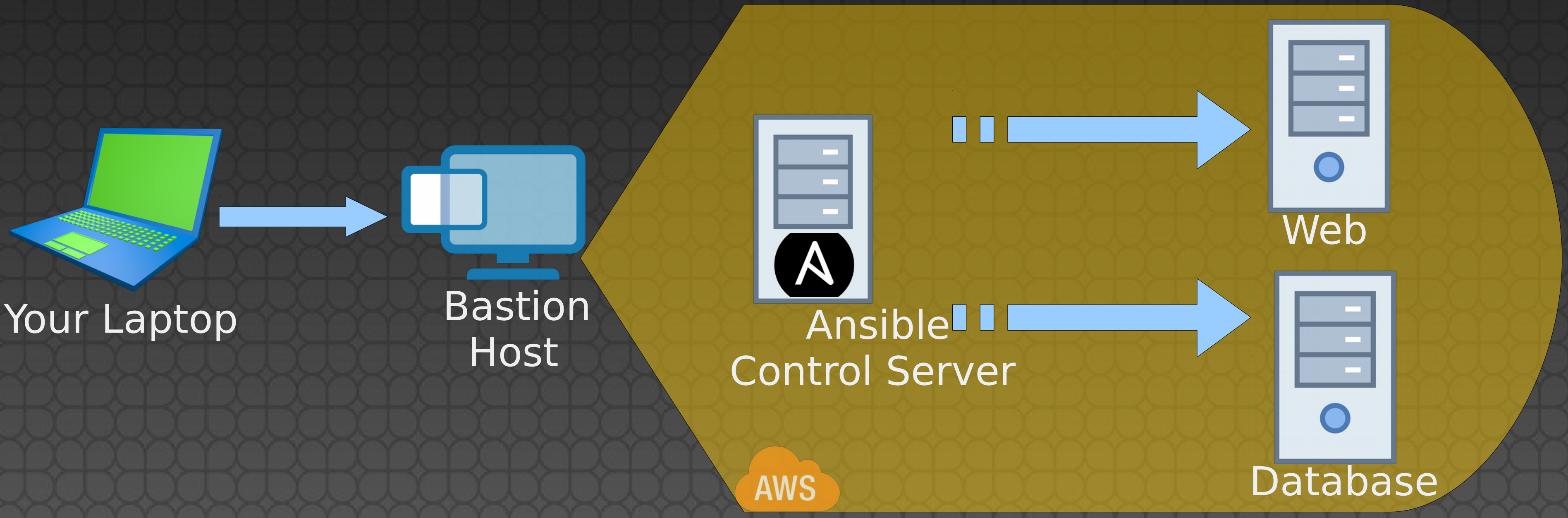
Why Ansible?

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

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
```

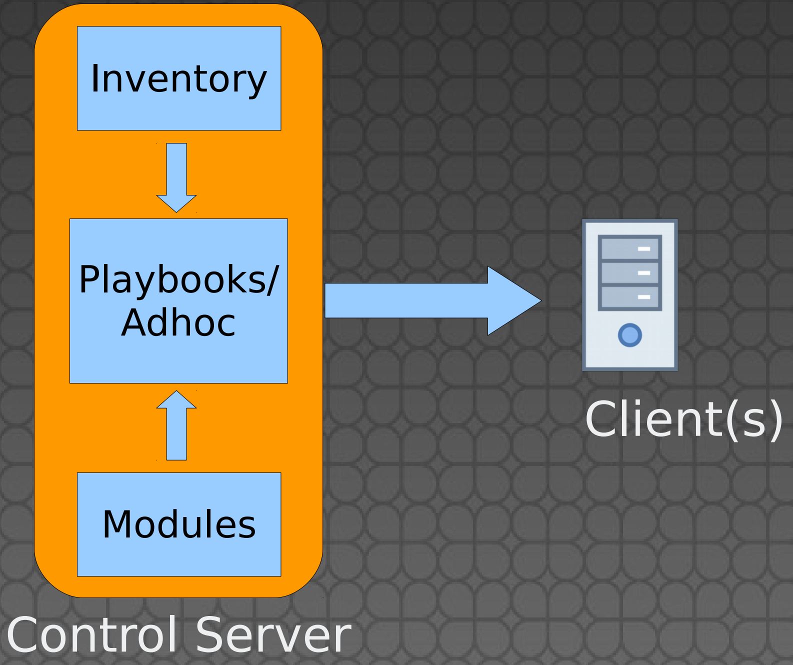
- Ansible is installed (that simple!)
- Ansible Configuration: /etc/ansible/ansible.cfg
- Default inventory: /etc/ansible/host
- Easily refer to the documentation:

```
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
- ansible_host=1.2.3.4

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

[webservers]
webserver.labXX

[dbservers]
database.labXX
```

Modules

- Ansible ships with ~500 Modules
  - You can write your own!
- Each modules 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-copyid webserver.labXX
```

```
ssh-copyid 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
  --filter “ansible_eth*”
  ```

- yum, apt
- copy, fetch
- hostname, timzone, 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

```
ansible *.labXX.local -m ping
```

- Regex ~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: 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
```

Real World Deployment I

Deployment Objectives

- Common
  - Disable selinux
  - Create some standard directories
  - Install vim
- Webserver
  - Install apache webserver
  - create webadmin user
- Database
  - Install mariadb database server
  - create dbadmin user
- Finally Reboot both servers
- Refer to /opt/workshop/rwd1/playbook.yml on your acs host

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

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

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
```

[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 webservers -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

Includes

Includes

Real World Deployment – I

Introduction to Ansible

- Lookups
- Handlers
- Tags
- Blocks
- Testing and Error Handling

Second Topic

Details

- Detail about second topic
- Another detail about second topic
    - Sub-detail

Summary

- Summarize first topic.
- Summarize second topic.

Questions?

Contact:
example@fedoraproject.org

Details

- Detail about first topic.
- Another detail about the first topic.

Second Topic

Details

- Detail about second topic
- Another detail about second topic
  - Sub-detail

Summary

- Summarize first topic.
- Summarize second topic.

Questions?

Contact:
example@fedoraproject.org

First Topic

Details

- Detail about first topic.
- Another detail about the first topic.

Second Topic

Details

- Detail about second topic
- Another detail about second topic
  - Sub-detail

Summary

- Summarize first topic.
- Summarize second topic.

Questions?

Contact:
example@fedoraproject.org