Ansible

Ansible

Installation and configuration

Communication framework

Password less Authentication

Ansible command line

Ansible playbooks

Ansible roles

Ansible tags

Ansible Vault

Ansible variables- precedence

Ansible jinja2 templates

Ansible galaxy

Chef,Puppet,Ansible, Saltstack these tools are for the same purpose – configuration management.

System admin/Operational tasks

provision server

deploy

configure

Monitor

Scale

Secure

Application reliability and scalability directly affect the business profits

Difficult to manage thousands of servers.

As the application grows in functionality the day to day activities takes more time

We need to control and flexibility, as well as ease of use.

Automation:

Automation of these tasks is becoming extremely important, hence DevOps

Smart Automation is the key

Recognize the most common tasks and automate

-Defined way of doing things is Recipes

In today’s world everything is code

kernel

OS

shell

Application

Configuration management tools

Provisioning resources

Ansible

Ansible in short is an IT Automation, configuration management and provisioning tool.

Simple to start

Scales up on demand

No agents are needed

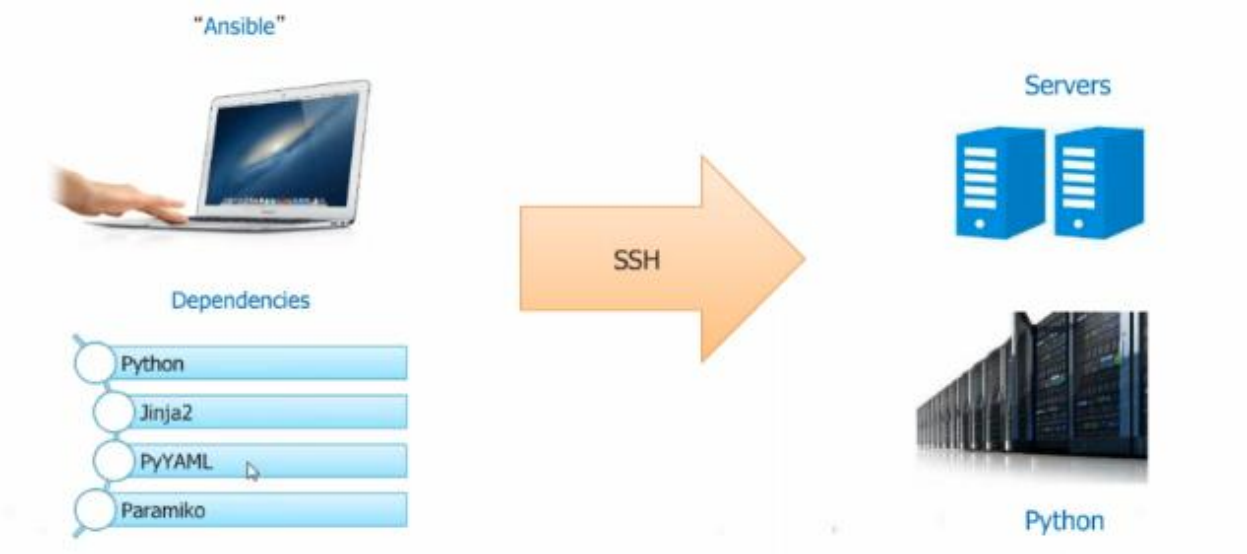
sshd on remote machine can be used

Parallel by default

Automaton language is like English

212 core modules

Ansible is a tool to manage systems and their configuration. Without the need for a client installed agent and with the ability to launch programs with command line, it seems to fit between classic configuration management like Puppet on one hand and ssh/dsh on the other.



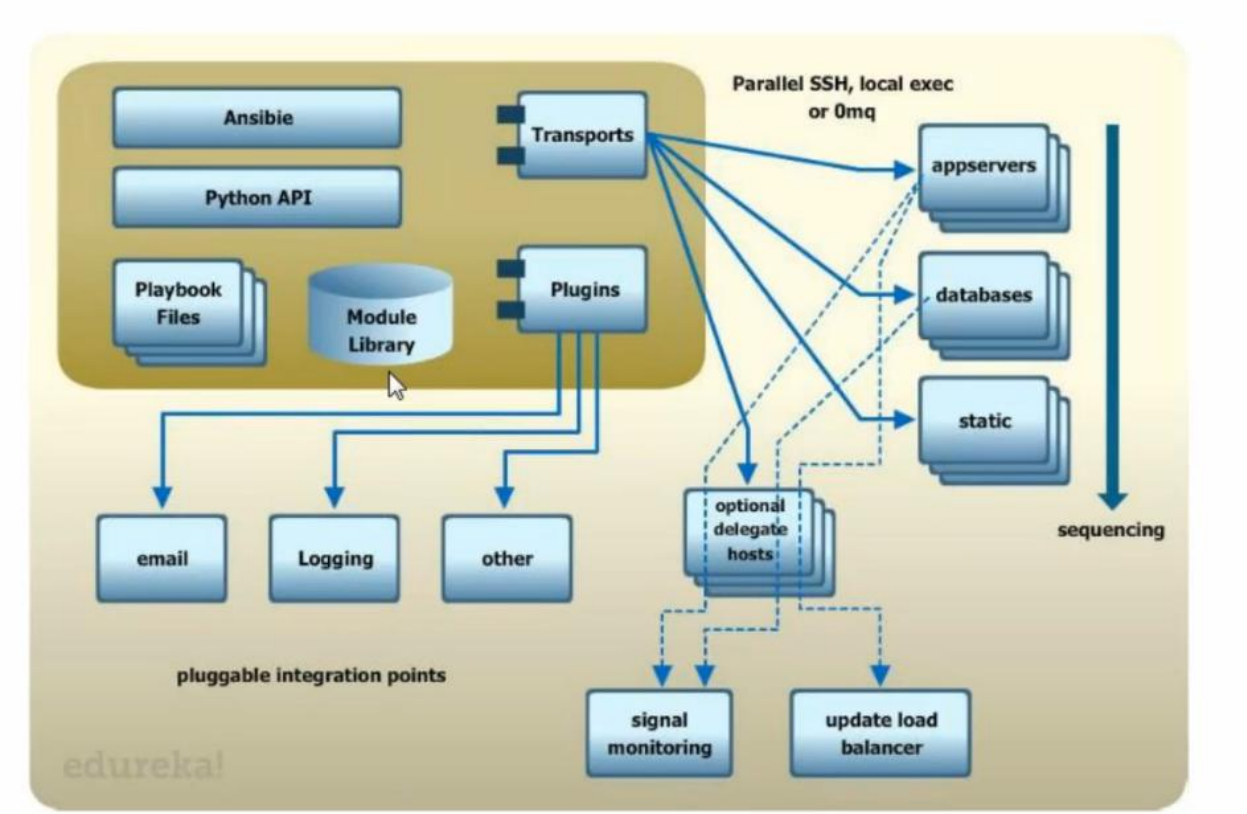
Ansible based on python

Jinja2 supports templates for Ansible – templating engine

PyYAML – yet another markup language – playbooks are written in this language

Paramiko – kind of ssh implementation.

Ansible Architecture



Playbook files – infrastructure as code is maintained in this files

Python API – execute commands using API

Module – is a resource – file/package/command/item trying to configure

transports – by default ssh – to talk to servers

plugins – email logging other

playbooks are executed on parallel fashion

Ansible terminology

Inventory: file contains list of hosts, groups or variables.

Modules: Actually do the work

Plugins: call back actions, and other hooks

Facts: Data gathered from target hosts

Playbooks: Collection of plays

Plays: Loops over a list of takes mapped to list of hosts

Tasks: Invokes a module to do the work

Ansible Install:

ssh-key has to setup on both the nodes

Ansible server talks to managed nodes using ssh

Default location of inventory: /etc/ansible/hosts

Its in INI format

Ansible repository Ansible Galaxy

How to use Ansible

ansbile inventory –m modulename

ansible-playbook

Ansible-playbook playbook.yml

Ansible command mode:

add hosts to /etc/ansible/hosts

and configure password less authentication

Ansible needs to know the hosts its going to serve. They can be managed on the central server in /etc/ansible/hosts or in a file configured in the shell variable ANSIBLE_HOSTS. The hosts can be listed as IP addresses or host names, and can contain additional information like user names, ssh port and so on:

Installation and configuration:

1. Install ansible on the node using

   yum install ansible

2. Add nodes to ansible server using inventory file

   /etc/ansible/hosts

3. Generate ssh keys and setup password less authentication between server and clients
4. Perform jobs either using ansible command line or playbooks.

Ansible command line:

[root@ctx1p05 RHEL5]# ansible all -m ping

ctx1p10.in.vp.com | SUCCESS => {

"changed": false,

```
    "ping": "pong"
}
ctx1p11.in.vp.com | SUCCESS => {
    "changed": false,
    "ping": "pong"
```

[root@ctx1p05 RHEL5]# ansible all -a "touch /tmp/hello "

ctx1p10.in.vp.com | SUCCESS | rc=0 >>

ctx1p11.in.vp.com | SUCCESS | rc=0 >>

to execute command on local machine (ansible server)

ansible all -i "localhost," -c local -m shell -a 'echo hello world'

to execute command on remote single machine (ansible client)

ansible all -i "ctx1p10" -m shell -a 'echo hello world'

Grouping the nodes:

Nodes can be grouped into groups like webserver dbserver application server etc in hosts file

[root@ctx1p05 RHEL5]# ansible webservers -m ping

ctx1p10.in.vp.com | SUCCESS => {

    "changed": false,

```
    "ping": "pong"
}
[root@ctx1p05 RHEL5]#



[webservers]
ctx1p10.in.vp.com

[dbservers]
ctx1p11.in.vp.com

to execute a command on local/ansible server
ansible all -i "localhost," -c local -m shell -a 'hostname'

root@ctx1p05 ~]# ansible all -i "localhost," -c local  -a 'hostname'
localhost | SUCCESS | rc=0 >>
ctx1p05.in.vpath.com
to execute command on single host
[root@ctx1p05 ~]# ansible all -i "reviewb.in.vpath.com,"   -a 'hostname'
reviewb.in.vpath.com | SUCCESS | rc=0 >>
reviewb.in.vpath.com
```

Ansible Playbooks:

A sample playbook:

```
[root@ctx1p05 playbooks]# cat httpd.yaml
---
- hosts: webservers
  tasks:
  - name: install httpd
    yum: pkg=httpd state=installed
[root@ctx1p05 playbooks]#
```

```
[root@ctx1p05 playbooks]# ansible-playbook httpd.yaml

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

TASK [setup]
**********************************************************************
ok: [ctx1p10.in.vp.com]
```

TASK [install httpd]
****

changed: [ctx1p10.in.vp.com]

PLAY RECAP
****

ctx1p10.in.vp.com : ok=2 changed=1 unreachable=0 failed=0

[root@ctx1p05 playbooks]#

Playbooks:

hosts defined on which target this configuration is pushed

user: to login to remote server

vars:

http_port:80

max_client: 200

Variables are declared and can be passed , wherever http_port is used, 80 will be replaced

Tasks: collection of actions we need to perform

each action will have a name

action will call the module

name: ensure apache is at the latest version

action: yum pkg=httpd state=latest

here yum is the module for rhel

we are passing some variables to yum

1st one is package name, 2nd one is action to install/remove/latest

for specific version we need to replace state with version

action: template src=httpd.j2 dest=/etc/httpd.conf

we want to create a file /etc/httpd.conf with template file httpd.j2

here we are using template

notify:

- restart apache

  Whenever you perform this action (when template is notified), this action has to be performed

  -name ensure httpd is running

  action=service name =httpd state=started

  handlers:

  -name: restart apache

  action: service name=httpd state=restarted

to understand handles, execute below playbook 2 times

[root@ctx1p05 playbooks]# cat template.yaml

```
---
- hosts: webservers
  tasks:
  - name: copy application code to destination on clients
    template: src=index.html.j2 dest=/var/www/html/index.htm
    notify: restart apache
```

```
  handlers:
   - name: restart apache
     service: name=httpd state=restarted
```

1st run

```
[root@ctx1p05 playbooks]# ansible-playbook template.yaml

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

TASK [setup]
**********************************************************************
ok: [ctx1p11.in.ibm.com]

TASK [copy application code to destnation on clients]
**************************

ok: [ctx1p11.in.ibm.com]

PLAY RECAP
**********************************************************************
ctx1p11.in.ibm.com         : ok=2    changed=0    unreachable=0    failed=0
```

2nd run:

Modify something in index.html on ansible master and deploy

[root@ctx1p05 playbooks]# ansible-playbook template.yaml

PLAY [webservers]
****

TASK [setup]
****

ok: [ctx1p11.in.ibm.com]

TASK [copy application code to destnation on clients]
****

changed: [ctx1p11.in.ibm.com]

RUNNING HANDLER [restart apache]
****

changed: [ctx1p11.in.ibm.com]

PLAY RECAP
****

ctx1p11.in.ibm.com : **ok=3** **changed=2** unreachable=0 failed=0

Note: if there is any change only handles will be used to restart service

To list all tasks in a module

[root@ctx1p05 playbooks]# ansible-playbook apache.yaml --list-tasks

playbook: apache.yaml

```
  play #1 (webservers): webservers      TAGS: []
    tasks:
      ensure apache is installed        TAGS: []
      ensure apache is running  TAGS: []
      copy files to document root       TAGS: []
      copy application code to document root    TAGS: []
[root@ctx1p05 playbooks]#


[root@ctx1p05 playbooks]# ansible-playbook apache.yaml --list-tasks

playbook: apache.yaml

  play #1 (webservers): webservers      TAGS: []
    tasks:
      ensure apache is installed        TAGS: []
      ensure apache is running  TAGS: []
      copy files to document root       TAGS: []
      copy application code to document root    TAGS: []
[root@ctx1p05 playbooks]# ansible-playbook --check apache.yaml

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

```
TASK [setup]
*******************************************************************

ok: [ctx1p10.in.ibm.com]


TASK [ensure apache is installed]
************************************************

ok: [ctx1p10.in.ibm.com]


TASK [ensure apache is running]
************************************************

ok: [ctx1p10.in.ibm.com]


TASK [copy files to document root]
*********************************************

ok: [ctx1p10.in.ibm.com]


TASK [copy application code to document root]
***********************************

ok: [ctx1p10.in.ibm.com]


PLAY RECAP
*******************************************************************************

ctx1p10.in.ibm.com         : ok=5    changed=0    unreachable=0    failed=0
```

ansible-playbook php.yaml --^Cmit @/etc/ansible/playbooks/php.retry

to retry a failed playbook execution

[root@ctx1p05 playbooks]# ansible-playbook php.yaml --limit @/etc/ansible/playbooks/intaractive.retry

PLAY [webservers]
****

TASK [setup]
****

ok: [ctx1p10.in.ibm.com]

TASK [xyz]
****

changed: [ctx1p10.in.ibm.com]

PLAY RECAP
****

ctx1p10.in.ibm.com : ok=2 changed=1 unreachable=0 failed=0

Online Ansbile Playbook repository : Ansible – galaxy

Ansible tower – licensed version of Ansible

Ansible is not supported on Windows, its supported only on Linux

in Ansbile modules are tools

playbooks are instruction manuals

5 basic components in Ansible

1)Inventory:

A text file that contains information of servers and systems in the infrastructure

Group hosts into categories

Define any variables required by specific hosts (ex: ssh username and password)

2)Ansible configuration:

custom configuration can be defined here

3)Modules

It is the command center of the system

2 types

core modules – developed by Ansible

other modules – developed by community

4)Playbooks

Play: It is a single task or set of tasks which can be executed on hosts with the help of modules

it Is set of plays built in specific order sequence to produce an expected outcome

5) Ansible global configuration

It is a file that contains global variables

execution types:

remote

local : when python is not installed on the agent

ex: router

Handles: wait for event to occur

once event occurs with notify action, handler which is mapped that event will gets executed

--

hosts: webservers

user: root

example playbooks

[root@ctx1p05 playbooks]# cat apache.yaml

- hosts: webservers

  tasks:

  - name: ensure apache is installed

```
    yum: pkg=httpd state=latest
  - name: ensure apache is running
    service: name=httpd state=running enabled=yes
  - name: copy files to document root
    copy: src=cloud.png dest=/var/www/html/cloud.png
  - name: copy application code to document root
    template: src=index.html.j2 dest=/var/www/html/index.html
    notify: restart apache
  handlers:
   - name: restart apache
     service: name=httpd state=restarted
[root@ctx1p05 playbooks]#
```

```
[root@ctx1p05 playbooks]# cat index.html.j2
<head>
    <title>Vp Ansible Demo</title>
</head>
<body>
    <h1>
        Welcome to Visual path Ansible demo.
    </h1>
    <br/><br/><br/>
```

```
    <p>
        <img src="cloud.png" alt="Logo"/>
    </p>
</body>
</html>
[root@ctx1p05 playbooks]#


roles:
[root@ctx1p05 playbooks]# cat myrole.yaml
---
- hosts: webservers
  roles:
   - webservers
[root@ctx1p05 playbooks]#


include:
[root@ctx1p05 playbooks]# cat myrole.yaml
---
- hosts: webservers
  roles:
   - webservers
[root@ctx1p05 playbooks]# cat sample.yaml
---
```

```
- hosts: webservers
  tasks:
    - include: /etc/ansible/tasks/file.yaml

- hosts: dbservers
  tasks:
    - include: /etc/ansible/tasks/file.yaml
[root@ctx1p05 playbooks]#
```

include is for making the code look simpler, manageable

small playbooks can be created and can be included

To see what hosts would be affected by a playbook before running it, try

```
root@ctx1p05 playbooks]# ansible-playbook apache.yaml --list-hosts
playbook: apache.yaml
  play #1 (webservers): webservers      TAGS: []
    pattern: [u'webservers']
    hosts (1):
      ctx1p10.in.vp.com
```

Ansible all –list-hosts

shows the nodes

Ansible all –s –m shell –a “mount –a”

installed – will install

latest - will upgrade if available

absent – will remove

with –check option – dry run method, will not do changes to machine

deprication_warnings = make it to no in cfg

-v is for verbose

--vv explain completely

--vvv full versbose

tags

to execute based on the task

with_loop

to execute in a loop

Interactive play book

include:

[root@ctx1p05 playbooks]# cat sample.yaml

```
---
- hosts: webservers
  tasks:
    - include: /etc/ansible/tasks/file.yaml

- hosts: dbservers
  tasks:
    - include: /etc/ansible/tasks/file.yaml
```

Roles:

Yaml file will grow with time as and when the tasks are increased

Directory structures and templates

```
[root@ctx1p05 playbooks]# cat myrole.yaml
---
- hosts: webservers
  roles:
   - webservers
[root@ctx1p05 playbooks]#
```

Using ansible facts to get os family

```
---
- hosts: ext1
  tasks:
  - name: install httpd
    yum: name=httpd state=present
    when: ansible_os_family == "RedHat"
  - name: install httpd
    apt: name=httpd state=present
    when: ansible_os_family == "Debian"
[root@ctx1p05 playbooks]#
```

Getting info about the facts

ansible ext1 -m setup --tree /tmp/facts

using user module to add user

ansible webservers –s –m user –a “name=user1 uid=20000 shell=/bin/bash”

including multiple roles in playbook:

```
[root@reviewb playbooks]# cat role.yaml
---
- hosts: webservers
  roles:
   - common
   - webservers
[root@reviewb playbooks]#
```

```
[root@reviewb playbooks]# cat /etc/ansible/roles/common/tasks/main.yaml
- name: create a dummy file for common role
  command: touch /tmp/dummy2role
```

```
[root@reviewb playbooks]# cat
/etc/ansible/roles/webservers/tasks/main.yaml
- name: ensure apache is installed
  yum: pkg=httpd state=latest
- name: ensure apache is running
  service: name=httpd state=running enabled=yes
- name: copy files to document root
  copy: src=cloud.png dest=/var/www/html/cloud.png
- name: copy application code to document root
```

```
  template: src=index.html.j2 dest=/var/www/html/index.html
```

[root@reviewb playbooks]# cat /etc/ansible/roles/webservers/templates/index.html.j2

```
<head>
    <title> Ansible </title>
</head>
<body>
    <h1>
        Welcome to Ansible Roles    </h1>
    <br/><br/><br/>
    <p>
        <img src="cloud.png" alt="Logo"/>
    </p>
</body>
</html>
```

parallel vs serial execution:

change forks=15 in cfg file

[root@ctx1p05 playbooks]# cat parallel.yaml

Parallel execution:

```
---
- hosts: all
  tasks:
    - name: sleep for 5 seconds
```

```
      shell: /bin/sleep 5
```

Serial execution:

```
[root@ctx1p05 playbooks]# cat serial.yaml
---
- hosts: all
  serial: 1
  tasks:
    - name: sleep for 5 seconds
      shell: /bin/sleep 5
[root@ctx1p05 playbooks]#
```

Serial:1 is for executing the playbook on single machine (one by one)

Serial:2 will execute on each of 2 machines (first 2 machines parallelly, then goes to third,forth)

pre and post tasks execution

```
[root@ctx1p05 playbooks]# cat preposttask.yaml
---
- hosts: webservers

  serial: 1
  pre_tasks:
    - name: install package
      yum: name=httpd state=installed
    - name: copy new template
```

```
      copy: src=/index1.html dest=/var/www/html/index.html
    - name: restart http
      service: name=httpd state=restarted
    - name: sleep for 5 sec
      shell: /bin/sleep 5
  tasks:
    - name: deploy website content
      copy: src=/index2.html dest=/var/www/html/index.html
    - name: restart http
      service: name=httpd state=restarted
  post_tasks:
    - name:
      service: name=httpd state=restarted
[root@ctx1p05 playbooks]#
```

Roles:

Roles define what each type of server has to perform

Sample:

```
- name: install and configure webservers
hosts: webservers
remote_user: ec2-user
sudo: yes
roles:
   - webservers
   - common
```

Here common and webservers role is associated with hosts webservers

[root@ctx1p05 playbooks]# cat myrole.yaml

```
---
- hosts: webservers
  roles:
   - webservers
```

[root@ctx1p05 playbooks]# cat /etc/ansible/roles/webservers/tasks/main.yaml

```
- name: ensure apache is installed
  yum: pkg=httpd state=latest
- name: ensure apache is running
  service: name=httpd state=running enabled=yes
- name: copy files to document root
  copy: src=cloud.png dest=/var/www/html/cloud.png
- name: copy application code to document root
  template: src=index.html.j2 dest=/var/www/html/index.html
  notify: restart apache
```

Playbook to create users

[root@ctx1p05 playbooks]# cat create.yaml

```
---
- hosts: webservers
  tasks:
```

```
    - name: add several users
      user: name={{ item.name }} state=present groups={{ item.groups }}
      with_items:
      - { name: 'testuser1', groups: 'root' }
      - { name: 'testuser2', groups: 'root' }
[root@ctx1p05 playbooks]#
```

Playbook to explain notify:

imagine a setup where you copy a file, and if that file was copied (so not there before or changed in the meantime) , we need to restart sshd:

```
---
- hosts: webservers
  remote_user: liquidat
  tasks:
      - name: copy file
        copy: src=~/tmp/test.txt dest=~/test.txt
        notify:
            - restart sshd
  handlers:
      - name: restart sshd
        service: name=sshd state=restarted
        sudo: yes
```

Using Inventory file to provide the variables/values:

[root@ctx1p05 playbooks]# ***cat inventory***

web1 ansible_ssh_host=9.184.184.145 ansible_ssh_user=john ansible_ssh_pass=root123

[root@ctx1p05 playbooks]# ***ansible web1 -i inventory -a "touch /tmp/hello15thfeb2017 "***

web1 | SUCCESS | rc=0 >>

Executing a playbook/command line ansible using non-root user

Create the user called user1 on Ansible server, assign the password and generate the keys

Create another user called user1 on Ansible agent, assign the password.

Copy keys from server to agent using ssh-copy-id for user1

Then execute:

[user1@reviewb ~]$ **ansible webservers1 -i inventory -a "touch /tmp/hello15thfeb2017 "**

webservers1 | SUCCESS | rc=0 >>

[root@ctx1p05 playbooks]# ***cat echo2.yaml***

- hosts: all

  tasks:

    - name: run echo command

command: /usr//bin/touch /tmp/hellofeb

ansible-playbook -i inventory echo2.yaml

Ansible Vault

Installing Ansible vault:

Comes by default with ansible

From Ansible 1.5, “Vault” is a feature that allows keeping sensitive data such as passwords or keys in encrypted files, rather than as plaintext in playbooks or roles. These vault files can then be distributed or placed in source control.

How to encrypt data

Encrypt files with ansible vault

- AES 256 encryption

Ansible will decrypt at the runtime

To encrypt the passwords for privacy ansible vault can be used

ansible-vault encrypt intaractive.yaml

ansible-playbook intaractive.yaml --ask-vault-pass

[root@ctx1p05 playbooks]# ansible-playbook intaractive.yaml --ask-vault-pass

Vault password:

Install which package? [telnet]:

PLAY [webservers]
****

TASK [setup]
****

ok: [ctx1p11.in.ibm.com]

TASK [ensure apache is installed]
****

ok: [ctx1p11.in.ibm.com]

PLAY RECAP
****

ctx1p11.in.ibm.com : ok=2 changed=0 unreachable=0 failed=0

ansible-vault encrypt <> - to encrypt the file

ansible-valut edit <> - to edit the file

ansible-vault rekey – to change password

Ansible until loop:

Sometimes we may want to retry a task until a certain condition is met. Here's an example:

[root@ctx1p05 playbooks]# cat until.yaml

- hosts: webservers

  tasks:

    - action: shell ifconfig

      register: result

```
    until: result.stdout.find("ens192") != -1
    retries: 5
    delay: 10
[root@ctx1p05 playbooks]# ansible-playbook until.yaml

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

TASK [setup]
**********************************************************************
ok: [ctx1p11.in.ibm.com]

TASK [command]
**********************************************************************
changed: [ctx1p11.in.ibm.com]

PLAY RECAP
**********************************************************************
ctx1p11.in.ibm.com         : ok=2    changed=1    unreachable=0    failed=0

[root@ctx1p05 playbooks]#
```

The above example run the shell module recursively till the module's result has "all systems go" in its stdout or the task has been retried for 5 times with a delay of 10 seconds. The default value for "retries" is 3 and "delay" is 5.

The task returns the results returned by the last task run. The results of individual retries can be viewed by -vv option. The registered variable will also have a new key "attempts" which will have the number of the retries for the task.

Ansible if loop:

Jinja2 template language can be used to populate the template files on the destination with loops as below:

If loop

For loop

If loop can be used to populate the destination template file using input variables we specify during playbook execution.

[root@ctx1p05 role1]# pwd

/etc/ansible/roles/role1

[root@ctx1p05 role1]# ls -lrt

total 0

drwxr-xr-x. 2 root root 21 Feb 17 02:37 templates

drwxr-xr-x. 2 root root 21 Feb 17 02:42 vars

[root@ctx1p05 role1]#

[root@ctx1p05 role1]# cd templates/

[root@ctx1p05 templates]# ls -lrt

total 4

-rw-r--r--. 1 root root 295 Feb 17 02:37 myvar.j2

[root@ctx1p05 templates]# cat myvar.j2

{% if param1 is defined and param2 is defined and param3 is defined %}

```
  myvariable: 'param1,param2,param3'
{% elif param1 is defined and param2 is defined %}
  myvariable: 'param1,param2'
{% elif param1 is defined %}
  myvariable: 'param1'
{% else %}
  myvariable: 'default-param'
{% endif %}
[root@ctx1p05 templates]#

[root@ctx1p05 role1]# cd vars/
[root@ctx1p05 vars]# ls
main.yml
[root@ctx1p05 vars]# cat main.yml
myvariable: 'param1,param2'
[root@ctx1p05 vars]#



[root@ctx1p05 playbooks]# cat role1.yml
- hosts: xyz
  gather_facts: yes
#  connection: local
  become: yes

  tasks:
```

```
  - template: src=/etc/ansible/roles/role1/templates/myvar.j2
dest=/etc/ansible/roles/role1/vars/main.yml


  - debug: var=myvariable


  roles:

        - { role: role1 }

[root@ctx1p05 playbooks]#




[root@ctx1p05 playbooks]# ansible-playbook role1.yml -e "param1=value1
param2=value2 param3=value3"


PLAY [xyz]
*********************************************************************


TASK [setup]
*********************************************************************

ok: [ctx1p18.in.ibm.com]


TASK [template]
*********************************************************************

changed: [ctx1p18.in.ibm.com]


TASK [debug]
*********************************************************************

ok: [ctx1p18.in.ibm.com] => {
```

```
    "myvariable": "param1,param2"
}

PLAY RECAP
*********************************************************************
ctx1p18.in.ibm.com         : ok=3    changed=1    unreachable=0    failed=0
```

On the destination machine:

```
root@ctx1p18:/etc/ansible/roles/role1/vars# cat main.yml
  myvariable: 'param1,param2,param3'
```

example 2:

```
[root@ctx1p05 playbooks]# ansible-playbook role1.yml

PLAY [xyz]
*********************************************************************

TASK [setup]
*********************************************************************
ok: [ctx1p18.in.ibm.com]

TASK [template]
*********************************************************************
changed: [ctx1p18.in.ibm.com]

TASK [debug]
*********************************************************************
```

```
ok: [ctx1p18.in.ibm.com] => {
    "myvariable": "param1,param2"
}

PLAY RECAP
*********************************************************************
ctx1p18.in.ibm.com         : ok=3    changed=1    unreachable=0    failed=0

[root@ctx1p05 playbooks]#
```

On destination machine:

```
root@ctx1p18:/etc/ansible/roles/role1/vars# cat main.yml
  myvariable: 'default-param'
```

Template if loop:

```
[root@ctx1p05 playbooks]# pwd
/etc/ansible/playbooks
[root@ctx1p05 playbooks]# cat role2.yml
- hosts: xyz
  gather_facts: yes
  become: yes

  tasks:

  - template: src=/etc/ansible/roles/role2/templates/myvar.j2
dest=/etc/ansible/roles/role2/vars/main.yml
```

```
  - debug: var=myvariable

  roles:
        - { role: role2 }
[root@ctx1p05 playbooks]#



[root@ctx1p05 role2]# pwd
/etc/ansible/roles/role2
[root@ctx1p05 role2]# ls -rlt
total 0
drwxr-xr-x. 2 root root 21 Feb 17 05:47 vars
drwxr-xr-x. 2 root root 21 Feb 17 06:00 templates
[root@ctx1p05 role2]# cat templates/myvar.j2
Tower name is: {{ hostvars['ctx1p18.in.ibm.com']['ansible_hostname'] }}
The epochs of the clients are:
{% for host in groups['xyz'] %}
- {{ hostvars[host]['ansible_date_time']['epoch'] }}
{% endfor %}
[root@ctx1p05 role2]#



[root@ctx1p05 playbooks]# ansible-playbook role2.yml

PLAY [xyz]
*********************************************************************
```

```
TASK [setup]
*******************************************************************

ok: [ctx1p18.in.ibm.com]


TASK [template]
*******************************************************************

changed: [ctx1p18.in.ibm.com]


TASK [debug]
*******************************************************************

ok: [ctx1p18.in.ibm.com] => {

    "myvariable": "param1,param2"

}


PLAY RECAP
*******************************************************************

ctx1p18.in.ibm.com         : ok=3    changed=1    unreachable=0    failed=0


[root@ctx1p05 playbooks]#
```

```
root@ctx1p18:/etc/ansible/roles/role1/vars# cat
/etc/ansible/roles/role2/vars/main.yml

Tower name is: ctx1p18

The epochs of the clients are:

- 1487329302
```

Writing variables to index.html using roles:

```
[root@ctx1p05 playbooks]# cat /etc/ansible/playbooks/role3.yml
- hosts: webservers
  vars:
    http_port: 80
    company: example.com
  roles:
        - { role: role3 }

[root@ctx1p05 playbooks]# cat /etc/ansible/roles/role3/tasks/main.yml
---
 - name: Copy index file.
   template:
    src: /etc/ansible/roles/role3/templates/myvar.j2
    dest: /var/www/html/index.html



[root@ctx1p05 playbooks]# cat /etc/ansible/roles/role3/templates/myvar.j2
```

```
<head>
   <title> Ansible </title>
</head>
<body>
   <h1>
      Welcome to Ansible {{ http_port }} {{ company }}
   </h1>
   <br/><br/><br/>
   <p>
      <img src="cloud.png" alt="Logo"/>
   </p>
</body>
</html>
```

How to use facts variables inside template:

Get the facts from the below command:

ansible -m setup ctx1p11.in.ibm.com

example:

```
ctx1p11.in.ibm.com | SUCCESS => {
   "ansible_facts": {
      "ansible_all_ipv4_addresses": [
         "9.182.76.101"
      ],
      "ansible_all_ipv6_addresses": [
         "fe80::d039:46f6:236:45f4"
```

```
        ],
        "ansible_architecture": "x86_64",
        "ansible_bios_date": "10/13/2009",
        "ansible_bios_version": "6.00",
        "ansible_cmdline": {
            "BOOT_IMAGE": "/vmlinuz-3.10.0-514.el7.x86_64",
            "LANG": "en_US.UTF-8",
            "crashkernel": "auto",
            "quiet": true,
            "rd.lvm.lv": "rhel/swap",
            "rhgb": true,
            "ro": true,
            "root": "/dev/mapper/rhel-root"
```


These variables can be directly used in tempaltes

Example: To display on site using index.html

Galaxy:

Download a role nfs server setup and use it to export a file system

Download the role and copy , extract it to roles

Create a playbook

[root@ctx1p05 playbooks]# cat nfs.yaml

- hosts: webservers

  roles:

   - ansible-role-nfs-master

Go thr readme

Provide directory name in defaults/main.yml

[root@ctx1p05 ansible-role-nfs-master]# cat defaults/main.yml

---

nfs_exports: [/abc]

nfs_exports: { "/home/public *(rw,sync,no_root_squash)" }

Ansible galaxy role mysql

Download the cookbook

Check the compatible versions/OS matrix

Download and create a playbook and use it to install mysql

geerlingguy.mysql

ansible-galaxy install geerlingguy.mysql

Ansible variable precedence:

Variables precedence or priority is in this order:

Variables declared in playbooks along with role

Variables declared in vars directory in the role

Variables defined in defaults

How to declare variables in playbook

Example 1:

```
[root@ctx1p05 playbooks]# cat role3.yml-bk
- hosts: webservers
  vars:
    http_port: 80
    company: example.com
  roles:
        - { role: role3 }

[root@ctx1p05 role3]# cat templates/myvar.j2
<head>
   <title> Ansible </title>
</head>
<body>
   <h1>
      Welcome to Ansible ROLES
   </h1>
   <h1>
      Welcome to {{ company }}
   </h1>
   <h1>
      Port is  {{ port_num }}
```

```
    </h1>

    <br/><br/><br/>
    <p>
        <img src="cloud.png" alt="Logo"/>
    </p>
</body>
[root@ctx1p05 role3]# cat tasks/main.yml
---
 - name: Copy index file.
   template:
    src: /etc/ansible/roles/role3/templates/myvar.j2
    dest: /var/www/html/index.html
```

Example2:

```
[root@ctx1p05 playbooks]# cat role3.yml
- hosts: webservers
  roles:
        - role3
[root@ctx1p05 playbooks]# cd /etc/ansible/roles/role3/
[root@ctx1p05 role3]# ls
tasks  templates  vars
[root@ctx1p05 role3]# cat vars/main.yml
port_num: 8081
company: XYZ123

[root@ctx1p05 role3]# cat /etc/ansible/playbooks/role3.yml
```

```
- hosts: webservers
  roles:
       - { role: role3, port_num: 9090 }     this has the highest precedence



[root@ctx1p05 role3]# cat vars/main.yml
port_num: 8081
company: XYZ123
[root@ctx1p05 role3]# cat templates/myvar.j2
<head>
   <title> Ansible </title>
</head>
<body>
   <h1>
      Welcome to Ansible ROLES
   </h1>
   <h1>
      Welcome to {{ company }}
   </h1>
   <h1>
      Port is  {{ port_num }}
   </h1>
   <h1>
      My host name is
{{ hostvars[groups['webservers'][0]].ansible_default_ipv4 }}
   </h1>
   <br/><br/><br/>
```

```
    <p>
        <img src="cloud.png" alt="Logo"/>
    </p>
</body>
```

File module to create a file/directory

```
[root@ctx1p05 playbooks]# cat file.yml
---
- hosts: webservers
  tasks:
    - name: create a directory
      file:
        path: /etc/xyz
        state: directory
        mode: 755
```

if state=directory creates directory

if nothing is given -> creates file

Register module to store value from command line:

```
[root@ctx1p05 playbooks]# cat register.yml
```

```
---
- hosts: webservers
  tasks:
    - shell: cat /etc/motd
      register: motd_contents

    - name: motd contains the word hi
      shell: echo "motd contains the word hi"
      when: motd_contents.stdout.find('hi') != -1
```

```
[root@ctx1p05 playbooks]# ansible-playbook register.yml

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

TASK [setup]
*****************************************************************
ok: [ctx1p11.in.ibm.com]

TASK [command]
*****************************************************************
changed: [ctx1p11.in.ibm.com]

TASK [motd contains the word hi]
*********************************************
```

changed: [ctx1p11.in.ibm.com]

PLAY RECAP
****

ctx1p11.in.ibm.com : ok=3 changed=2 unreachable=0 failed=0

Ansible cheatsheat

ANSIBLE CHEAT SHEET

$ansible

call a shell command:
```
-a "$command"
```
call with sudo rights:
```
--sudo
```
test if machine responses:
```
-m ping
```
call an arbitrary module:
```
-m $module -a "$argument"
```
gather specific facts:
```
-m setup -a "filter=*distri*"
```

$ansible-playbook

call a book:
```
$host some/playbook.yaml
```
run in parallel:
```
-f 10
```
list affected hosts:
```
--list-hosts"
```

basic playbook.yaml

YAML format:
```
---
- hosts:all
```
launch task:
```
- name: check avail
  ping:
```
variables:
```
vars:
  rack: green
```
include tasks:
```
tasks:
 - { include: tasks/my.yml, user=alice }
```
include handlers:
```
handlers:
 - { include: handlers/my.yml }
```
call depedency:
```
tasks:
 - name: copy file
   copy: src=a dest=b
   notify:
    - restart sshd
handlers:
 - name: restart sshd
```

inventory

general inventory:
```
/etc/ansible/hosts
```
shell variable:
```
$ANSIBLE_HOSTS
```
normal entry:
```
www.example.com
```
multi-definition:
```
db[0-9].example.com
```
custom ip:
```
ansible_ssh_host
```
grouping:
```
[group]
```
specific user:
```
ansible_ssh_user
```
specific port:
```
ansible_ssh_port
```

complex playbook.yaml

loop over items:
```
- name: copy file
  copy: src={{item.src}} \
        dest={{item.dst}}
  with_items:
     - {src: a, dst: b}
     - {src: k, dst: l}
```
conditionals:
```
- name: reboot Debian
  command: shutdown -r now
  when: ansible_os_family == "Debian"
```
Roles:
```
- hosts: webserver
  roles:
    - common
    - dbservers
  roles/
    common/
       files/
       templates/
       tasks/main.yml
       handlers/main.yml
       vars/main.yml
       meta/main.yml
   webservers/
   ...
```

Create user yaml:

```
---
- hosts: webservers
  tasks:
  - name: create user
    user:
      name: user12345
      comment: "user12345 "
      uid: 1090
      shell: "/bin/bash"
      createhome: yes
      home: "/home/user12345"
      password: kieVINJhMcitY
```

to generate password in encrypted format:

```
openssl passwd
```

run_once keyword:

run_once: true to run task only once per playbook run.

Example:

```
[root@reviewb playbooks]# cat run_once.yaml
---
- hosts: webservers
  tasks:
  - name: create a dummy file
```

```
    command: touch /tmp/dummymay30

    run_once: true
```

even though there are 2 hosts in webservers, it will be executed only on 1 host.