**Step-1**

Install Ansible in REDHAT 7

rpm -Uvh<https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm>

[yum install ansible](https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm)

**Step-2**

cd /etc/ansible

Place the ssh file to

vim /etc/ansible/hosts

[dashboard:vars]

ansible\_ssh\_private\_key\_file=./ansible-1.pem

[dashboard]

172.31.30.217 ansible\_ssh\_port=22 ansible\_ssh\_user=ec2-user

[apache]

172.31.30.217 ansible\_ssh\_port=22 ansible\_ssh\_user=ec2-user

Step-3

We have to verify the node

ansible apache -s -m ping or ansible -m ping all

ansible apache -s -m setup -a ‘filter=ansible\_os\_family’

ansible apache -s -m setup | grep distribution

**Install apache package through command line :**

ansible apache -s -m shell -a 'yum install httpd -y'

ansible apache -s -m shell -a 'service httpd start'

ansible apache -s -m shell -a 'yum remove httpd -y'

**Step-4**

How to create the playbook

cd /etc/ansible

vim first-playbook

---

- hosts: apache

connection: ssh

gather\_facts: no

tasks:

- name: Install Apache package

yum: pkg=httpd state=present update\_cache=true

**Execute the ansible through playbook**

ansible-playbook -s first-playbook.yml

**Task Section:**

---

- hosts: apache

connection: ssh

gather\_facts: no

tasks:

- name: Install Apache package

yum: pkg=httpd state=present update\_cache=true

**Note: there are different state are there**

latest :

absent:

present:

installed:

removed:

**Variable Section:**

vim vars.yml

package: telnet

package-2: php

vim first-playbook.yml

---

- hosts: apache

connection: ssh

gather\_facts: no

vars:

apache-pkg: httpd

vars\_files:

- vars.yml

tasks:

- name: Install apache package to host

yum: pkg=httpd state=installed update\_cache=true

- name: Install package through variable and variable file

yum: pkg={{ package }} state=installed update\_cache=true

- name: Install ano

ther package through variable file

yum: pkg={{ package-2 }} state=installed update\_cache=true

**Handler & Notify Section:**

vim first-playbook.yml

---

- hosts: apache

connection: ssh

gather\_facts: no

vars:

apache-pkg: httpd

vars\_files:

- vars.yml

tasks:

- name: Install apache package to host

yum: pkg=httpd state=installed update\_cache=true

Notify: Restart HTTPD

handlers:

- name: Restart HTTPD

service: name=httpd state=started ( started, stopped, restarted)

**You can run things in “Dry Run” Mode**

Ansible will not make any changes to your host, but simply report what changes would have been made if the playbook was run without this flag.

ansible-playbook -s first-playbook.yml --check

**Using with\_items might be a good idea**

When you use the **with\_items** clause, Ansible will create a variable called **{{item}}** containing the value for the current iteration. Some modules handle collections of items really well and are actually faster than running the same task multiple times with different parameters.

**# Installing all packages with one task (faster)**

- name: install required packages using the apt module

apt: package={{ item }} update\_cache=yes

with\_items:

- git

- memcached

- nginx

**# Installing packages individually (slower)**

- name: install git

apt: package=git update\_cache=yes

- name: install memcached

apt: package=memcached update\_cache=yes

- name: install nginx

apt: package=nginx update\_cache=yes

**Conditionals Section:**

vim first-playbook.yml

---

- hosts: apache

connection: ssh

gather\_facts: yes

vars:

centos-pkg: httpd

ubuntu-pkg: apache2

tasks:

- name: Install apache package to host

yum: pkg={{ centos-pkg }} state=installed update\_cache=true

when: ansible\_os\_family = “RedHat”

- name: Install apache package to host

yum: pkg={{ ubuntu-pkg }} state=installed update\_cache=true

when: ansible\_os\_family = “Debian”

**Until Section:**

tasks:

- name: Install apache webserver

yum: pkg=httpd state=latest

- name: Verify service status

shell: service https status

register: status\_result

until: status\_result.stdout.find ( “running” ) != -1

retries: 5

delay: 5

- debug: var=status\_result

**Use ansible-vault when you want to store sensitive information**

If one of your tasks requires sensitive information (let’s say the database user and password), it’s a good practice to keep this information encrypted, instead of storing it in plain text.

Ansible ships with a command line tool called ansible-vault, that allows you to create and manage encrypted files. This way you can commit the encrypted file to your source control and only users with the decryption password will be able to read it.

**# Encrypt an existing file. You'll need to create an encryption password.**

ansible-vault encrypt secrets.yml

**# Creates a new, encrypted file. You'll need to create an encryption password.**

ansible-vault create secrets.yml

**# Decrypt a file. You'll have to enter password used for encryption.**

**# Use it with caution! Don't leave your files encrypted.**

ansible-vault decrypt secrets.yml

**# Edit an encrypted file (uses vim by default, can be overridden by the environment variable $EDITOR)**

ansible-vault edit secrets.yml

**# Print the contents of the encrypted file**

ansible-vault edit secrets.yml

If you import the vars\_file **secrets.yml** in your playbook, Ansible will fail, as it will not know how to read the encrypted file. You’ll have to specify the command line argument **--ask-vault-pass,** which will make Ansible prompt you the password of the encrypted file.

ansible-playbook playbook.yml -i hosts --ask-vault-password

Another way is to store the password in a file (which should not be commited) and specify the path to the file using the **--vault-password-file** argument. If this file is marked as executable, Ansible will run it and use the output as the password.

**Prompt - Interactive playbook**

---

- hosts: apache

connection: ssh

gather\_facts: yes

vars\_prompt:

- name: centospkg

prompt: Install which package ?

default: telnet

tasks:

- name: Install apache package to host

yum: pkg={{ centospkg }} state=installed update\_cache=true

**Tags Section:**

tasks:

- name: Install apache package to host

yum: pkg=httpd state=installed update\_cache=true

tags:

- httpd

- name: Install telnet package to host

yum: pkg=telnet state=installed update\_cache=true

tags:

- telnet

**Start and stop section :**

tasks:

- name: Task-1

yum: pkg=httpd state=installed update\_cache=true

- name: Task-2

yum: pkg=telnet state=installed update\_cache=true

- name: Task-3

yum: pkg=telnet state=installed update\_cache=true

ansible-playbook first-playbook.yml --start-at-task=’Task-1’

ansible-playbook first-playbook.yml --step

**Pass variable in command line**

---

- hosts: {{ apache }}

connection: ssh

gather\_facts: yes

tasks:

- name: Install apache package to host

yum: pkg={{ centos-pkg }} state=installed update\_cache=true

ansible-playbook -s first-playbook.yml --extra-vars “ hosts=apache centos-pkg=httpd ”

**Jinja2 Template:**

tasks:

- name: customized configuration file

template: src=httpd.conf.j2 dest=/etc/httpd/conf owner=root group=root

**Module :**

setup module:

ansible apache -s -m setup

copy module:

- name : copy the files to remote server

copy: src=files/test.txt dest=/var/www/html/ owner=apache group=apache

wait\_for module:

- name: waiting for port 80 to listen

wait\_for:

port: 80

state: started

yum module:

- name: install apache webserver in centos

yum: pkg=httpd state=installed

apt module:

- name: install apache webserver in centos

apt: pkg=apache2 state=installed

service module:

- name: Restart HTTPD

service: name=httpd state=started

cron module:

- name: add a cron job to the node

Cron: name=”list files” minute=”0” hour=”1” job=”ls -a”

debug module:

tasks:

- name: Install apache webserver

yum: pkg=httpd state=latest

- name: Verify service status

shell: service https status

register: status\_result

- debug: var=status\_result

user module:

tasks:

- name: create the user call tomcat

user: name=tomcat shell=/bin/bash

script module:

- script: files/command.sh

raw module:

- tasks:

- name: find the date in hosts

raw: /usr/bin/uptime

ping module:

- tasks:

- name: ping all the hosts

ping:

unarchive module:

- name: copy and unarchive a file

unarchive: src=files/test.tar.tz dest=/var/www/html/

get\_url module:

- name: download the files from urls

get\_url: url=httpd://url dest=/var/www/html

group module:

tasks:

- name: create the user call tomcat

group: name=tomcat shell=/bin/bash

mail module:

tasks:

- name: send a email

mail:

host: “localhost”

port: “25”

to: “test@example.com”

subject: “Host is finished”

body: “system called {{ ansible\_hostname }} has been successful”

git module:

tasks:

-name: check out the git repo to remote host

git: rdest=/var/wwepo=ssh://giturl w/html

template module:

tasks:

- name: update the conf file to host node

template: src=httpd.conf.j2 dest=/etc/httpd/conf owner=root group=root

**Role:**

Roles are a further level of abstraction that can be useful for organizing playbooks. As you add more and more functionality and flexibility to your playbooks, they can become unwieldy and difficult to maintain as a single file. Roles allow you to create very minimal playbooks that then look to a directory structure to determine the actual configuration steps they need to perform.

**1 - Directory Structure:**

# mkdir role

# cd role

# mkdir webserver

# cd webserver

# mkdir files handlers meta templates tasks vars

These are the directories that will contain all of the code to implement our configuration. You may not use all of the directories, so in real practice, you may not need to create all of these directories.

**This is what they are all for:**

**files:** This directory contains regular files that need to be transferred to the hosts you are configuring for this role. This may also include script files to run.

**handlers:** All handlers that were in your playbook previously can now be added into this directory.

**meta:** This directory can contain files that establish role dependencies. You can list roles that must be applied before the current role can work correctly.

**templates:** You can place all files that use variables to substitute information during creation in this directory.

**tasks:** This directory contains all of the tasks that would normally be in a playbook. These can reference files and templates contained in their respective directories without using a path.

**vars:** Variables for the roles can be specified in this directory and used in your configuration files.

2 - Role Based Tasks:

# ls

webserver

# vim webserver.yml

---

- hosts: apache

connection: ssh

gather\_facts: yes

roles:

- webserver

# cd webserver

# ls

files

handlers

meta

templates

tasks

vars

# cd tasks

vim main.yml

- name: Install centos apache web server

yum: pkg=httpd state=latest

ansible-playbook -s webserver.yml

**3 - Task Order - Pre and Post Task**

# vim webserver.yml

---

- hosts: apache

connection: ssh

gather\_facts: yes

pre\_tasks:

- name: When did the role start

raw: date

roles:

- webserverl

post\_tasks:

- name: When did the role end

raw: date

**4 - Conditional execution:**

**webservers → tasks → main.yml**

- name: Install centos apache web server

yum: pkg=httpd state=latest

when: “ ansible\_os\_family == ‘RedHat’ “

- name: Install centos apache web server

yum: pkg=apache2 state=latest

when: “ ansible\_os\_family == ‘Debian’ ”

**5 - Variable Substitution:**

**webservers → vars → main.yml**

**redhat\_apache: httpd**

**debian\_apache: apache2**

**webservers → tasks → main.yml**

- name: Install centos apache web server

yum: pkg={{ redhat\_apache }} state=latest

when: “ ansible\_os\_family == ‘RedHat’ “

- name: Install centos apache web server

yum: pkg={{ debian\_apache }} state=latest

when: “ ansible\_os\_family == ‘Debian’ ”

**6 - Handlers & Notify**

**webservers → handlers → main.yml**

- name: Restart HTTPD

service: pkg={{ redhat\_apache }} state=restarted

when: “ ansible\_os\_family == ‘RedHat’ “

- name: Restart Apache2

service: pkg={{ debian\_apache }} state=restarted

when: “ ansible\_os\_family == ‘Debian’ ”

**webservers → tasks → main.yml**

- name: Install centos apache web server

yum: pkg={{ redhat\_apache }} state=latest

when: “ ansible\_os\_family == ‘RedHat’ “

Notify: Restart HTTPD

- name: Install centos apache web server

yum: pkg={{ debian\_apache }} state=latest

when: “ ansible\_os\_family == ‘Debian’ ”

notify: Restart Apache2

**7 - Configure alternative role paths:**

We have to configure the roles path in ansible.cfg and place the webservice.yml file to anywhere in server, then run the playbook

It’ll automatically take the role path from ansible.cfg file

**8 - Conditional Include Statement:**

**We have three roles “ webserver, redhat\_webservers, debian\_webservers”. Include redhat and debian role to main playbook**

# vim webserver.yml

---

- hosts: apache

connection: ssh

gather\_facts: yes

pre\_tasks:

- name: When did the role start

raw: date

roles:

- { role: redhat\_webservers, when: “ ansible\_os\_family == ‘RedHat’ ”}

- { role: debian\_webservers, when: “ ansible\_os\_family == ‘Debian’ ”}

post\_tasks:

- name: When did the role end

raw: date

**9 - Waiting for events**

**webservers → tasks → main.yml**

- name: Install centos apache web server

yum: pkg={{ redhat\_apache }} state=latest

when: “ ansible\_os\_family == ‘RedHat’ “

- name: waiting for port 80

wait\_for:

port: 80

state: started

- name: Install centos apache web server

yum: pkg={{ debian\_apache }} state=latest

when: “ ansible\_os\_family == ‘Debian’ ”

**10 - Execution a task until**

**webservers → tasks → main.yml**

- name: Install centos apache web server

yum: pkg={{ redhat\_apache }} state=latest

when: “ ansible\_os\_family == ‘RedHat’ “

- shell: systemctl status httpd

register: redhat\_result

until: redhat\_result.stdout.find ( “ active (running)” ) != -1

retries: 5

delay: 5

- name: waiting for port 80

wait\_for:

port: 80

state: started

- name: Install centos apache web server

yum: pkg={{ debian\_apache }} state=latest

when: “ ansible\_os\_family == ‘Debian’ ”

**11 - Using Tags**

**webservers → tasks → main.yml**

- name: Install centos apache web server

yum: pkg={{ redhat\_apache }} state=latest

when: “ ansible\_os\_family == ‘RedHat’ “

tags: Install\_List

- shell: systemctl status httpd

register: redhat\_result

until: redhat\_result.stdout.find ( “ active (running)” ) != -1

retries: 5

delay: 5

- name: waiting for port 80

wait\_for:

port: 80

state: started

- name: Install centos apache web server

yum: pkg={{ debian\_apache }} state=latest

when: “ ansible\_os\_family == ‘Debian’ ”

**12- Passing Variable from command line**

# vim webserver.yml

---

- hosts: {{ host }}

connection: ssh

gather\_facts: yes

pre\_tasks:

- name: When did the role start

raw: date

roles:

- webserver

post\_tasks:

- name: When did the role end

raw: date

ansible-playbook -s webserver.yml --extra-vars “ hosts=apache ”

**13 - Using Jinja2 Template**

* **templates: You can place all files that use variables to substitute information during creation in this directory.**

**webservers → tasks → main.yml**

- name: Install centos apache web server

yum: pkg={{ redhat\_apache }} state=latest

when: “ ansible\_os\_family == ‘RedHat’ “

- shell: systemctl status httpd

register: redhat\_result

until: redhat\_result.stdout.find ( “ active (running)” ) != -1

retries: 5

delay: 5

- name: waiting for port 80

wait\_for:

port: 80

state: started

- name: customized configuration file

template: src=httpd.conf.j2 dest=/etc/httpd/conf owner=root group=root

Place the httpd.conf.j2 template to template directory.

**14 - Task Order - Pre and Post Task**

Call multiple role to single playbook

# vim webserver.yml

---

- hosts: apache

connection: ssh

gather\_facts: yes

pre\_tasks:

- name: When did the role start

raw: date

roles:

- java

- tomcat

- webserver

post\_tasks:

- name: When did the role end

raw: date