**ANSIBLE-O1**

Configuration Management

============================

This is the process of configuring servers from one point

of control

Advantages

==================

1 Provisioning of Servers

Setup of s/w's on servers can be done very easily from one point

2 Reduction of usage of resources

We require less amount of time,money and human resources to configure

servers

3 Handling Snowflake servers

After a point of time all servers in the data center behave like

snowflake servers ie they might be running on slightly different

h/w and s/w confurations.Configuaration Management tools can pick

up this info in simple setup file which can be reused to setup

similar environments

4 Disaster Recovery

In case of disaster recovery where we can loose an entire data center

we can recreate similar data center with greater ease

5 Idempotent

Configuration Management tools are used to bring the servers to a

specific state called as "desired state",If the rmeote server is

already in the desired state CM tools will not reconfigure that server

===========================================================================

Popular CM tools

======================

Ansible

Chef

Puppet

Saltstack

=========================================================================

Ansible is installed on one machine that is called as "Controller"

all the remaining servers that we want to configure are called

as "managed nodes/hosts"

Ansible uses "agentless" policy to configure the remote servers

ie we don't require any client side s/w of ansible to be present

on the managed nodes

Ansible uses "push" methodology to push the configuration changes

via passwordless ssh

----------------------------------------------------------------------

=========================================================================

Setup of Ansible

=============================

1 Create 3-4 AWS ubuntu instances

2 Name the first one Controller and the remaining as MAnaged nodes

3 Establish passwordless ssh between Controller and managed nodes

a) Connect to managed node

b) Setup password of ubuntu user

sudo passwd ubuntu

c) Edit the sshd\_config file

sudo vim /etc/ssh/sshd\_config

Search for "PasswordAuthentication" and change it from no to yes

d) Restart ssh

sudo service ssh restart

e) Connect to Controller

f) Generate the ssh keys

ssh-keygen

g) Copy the public keys to authoried\_keys on managed nodes

ssh-copy-id ubuntu@private\_ip\_of\_managednode

4 Install Ansible

a) Update the apt repository

sudo apt-get update

b) Install softwares required for ansible

sudo apt-get install software-properties-common

c) Add the latest version of ansible to apt repository

sudo apt-add-repository ppa:ansible/ansible

d) Update the apt repository

sudo apt-get update

e) Install ansible

sudo apt-get install -y ansible

Ansible uses a files known as "inventory" file to read info about

the managed nodes,Here we should add the rmeote managed nodes

ipaddress

sudo vim /etc/ansible/hosts

Copy paste the private ip address of all managed nodes

================================================================

=========================================================================

Important modules in Ansible

=====================================

1 command: This is used to run basic linux commands on the managed

nodes.This is the default module of Ansible

2 shell: This is used to run shell scripts or python scripts

on the managed nodes.It is also used to run commands related

to redirection and piping

3 ping: This is used to check if the rmeote servers are pingable or not

4 user: This is used to perform user administration on managed nodes

ie creating users,assigning home dirs,deleting users etc

5 copy: Used to copy files and directories to the managed nodes

6 fetch: Used to copy files and directories from the managed nodes to

controller

7 apt: Used to s/w package managemrn on DEbain,Ubuntu based machines

8 yum: similar to apt but it works on Redhat linux,centos etc

9 service: USed to start,stop or restart services on the managed nodes

10 git: Used to perform git version controlling on managed nodes

11 get\_url: Used to downlaod files from remote service

12 uri: Used to check if the remote url is reachable or not

13 debug : Used to display some output in JSON file format

14 stat: Used to capture info about files and folders present on managed nodes

15 include: This is used to call child playbooks from a parent playbook

16 replace: This is used to replace specific text in a file

17 lineinfile: This is also similar to replace

18 docker\_container: Used to manage docker containers on the managed nodes

19 docker\_image: USed to manage docker images

20 docker\_login: Used to login into docker registr

21 file: Used to create delete files/directories

22 k8s: Used to manage Kubernetes services

========================================================================

Ansible perfrom remote configuration in 3 ways

1 Adhoc commands

2 Playbooks

3 Roles

.**ANSIBLE-02**

Adhoc commands

====================

Syntax of Adhoc commands

--------------------------------------

ansible all/ipaddress/group\_name -i path\_of\_inventory -m module\_name

-a ' '

Command Module

==================

Ansible command to see the memory information of all managed nodes

ansible all -i /etc/ansible/hosts -m command -a 'free m'

/etc/ansible/hosts is the default inventory file and when using it

we need not give -i

ansible all -m command -a 'free m'

command module is the default module os ansible and we need not use

-m option when working on command module

ansible all -a 'free m'

====================================================================

Shell Module

-----------------

Ansible commands to downlaod the docker script and execute it to install docker

ansible all -m shell -a ' curl -fsSL https://get.docker.com -o get-docker.sh'

ansible all -m shell -a 'sh get-docker.sh'

--------------------------------------------------------------------------

Ansible command to capture memory statics into a file called as file1

ansible all -m shell -a 'free -m > file1'

=====================================================================

User Module

=================

Ansible command to create a user and assign a password

ansible all -m user -a 'name=sai password=levelup' -b

Ansible command to create a user,assign home dir,default shell

uid,comment etc

ansible all -m user -a 'name=Anu password=levelup

home=/home/ubuntu/Anu uid=12345 shell=/bin/bash

comment="A normal user"' -b

-------------------------------------------------------------------

Ansible command to create file on all managed nodes

ansible all -m file -a 'name=/tmp/file1 state=touch'

state=touch create files

state=directory creates directories

state=absent delete files/directories

===========================================================================

Ansible command to create file and also specify the owner,group owner

and permissions

ansible all -m file -a 'name=/tmp/file1 state=touch owner=Anu

group=Anu mode=750' -b

===========================================================================

Ansible command to copy /etc/passwd file to all managed nodes

ansible all -m copy -a 'src=/etc/passwd dest=/tmp'

Create a chain of directories on the controller and copy all that to

the managed nodes

mkdir -p d1/d2/d3/d4/d5

ansible all -m copy -a 'src=d1 dest=/home/ubuntu'

Copy module can also change the permissions,ownership and group ownership

of files that are copied

ansible all -m copy -a 'src=file1 dest=/tmp owner=root

group=anu mode=700' -b

===========================================================================

Ansible command to install git on all managed nodes

ansible all -m apt -a 'name=git state=present' -b

state=present is for installation

state=absent is for uninstallation

state=latest is for upgrading to a later version

To update the apt repository we can use

update\_cache-yes

Ansible command to install tomcat9 after updating the apt repository

ansible all -m apt -a 'name=tomcat9 state=present update\_cache=yes' -b

========================================================================

Service module

===================

Ansible command to restart tomcat on all managed nodes

ansible all -m service -a 'name=tomcat9 state=restarted' -b

state=restarted is for restarting

state=stopped is for stopping

state=started is for starting

========================================================================

Fetch Module

=====================

Ansible command to copy /etc/group file from all managed nodes to controller

ansible all -m fetch -a 'src=/etc/group dest=/tmp'

==========================================================================

Git Module

================

Ansible command to clone a remote git repository into all managed nodes

ansible all -m git -a 'repo=https://github.com/AnupamaSoma/maven-project.git dest=/tmp/git'

=========================================================================

Replace Module

==================

Ansible command to change port of tomcat from 8080 to 9090

ansible all -m replace -a 'regexp=8080 replace=9090

path=/etc/tomcat9/server.xml' -b

ansible all -m service -a 'name=tomcat9 state=restarted' -b

=========================================================================

Uri Module

===================

Ansible comamand to check if google is reachbale from all managed nodes

ansible all -m uri -a 'url=http://google.com status\_code=200'

status\_code=200 is success

status\_code=-1 is failure

========================================================================

get\_url

==============

Ansible command to download jenkins.war into all managed nodes

ansible all -m get\_url -a

'url=https://get.jenkins.io/warstable/2.277.2/jenkins.war dest=/tmp'

============================================================================

Configuring apache2

========================

1 Install apache2 on all managed nodes

ansible all -m apt -a 'name=apache2 state=present' -b

2 Edit the index.html file

ansible all -m copy -a 'content="Welcome" dest=/var/www/html/index.html' -b

3 Restart apache2

ansible all -m service -a 'name=apache2 state=restarted' -b

4 Check the url response of apache2

ansible all -m uri -a 'url=http://172.31.28.60 status\_code=200'

**ANSIBLE-03**

Configuring tomcat9

==========================

1 Install tomcat9

ansible all -m apt -a 'name=tomcat9 state=present update\_cache=yes' -b

2 Copy the tomcat-users.xml file

ansible all -m copy -a 'src=tomcat-users.xml dest=/etc/tomcat9' -b

3 Change port of tomcat

ansible all -m replace -a 'regexp=9090 replace=8080

path=/etc/tomcat9/server.xml' -b

4 Restart tomcat

ansible all -m service -a 'name=tomcat9 state=restarted' -b

5 Check the url reponse of tomcat

ansible all -m uri -a 'url=http://172.31.28.60:8080 status\_code

=======================================================================

Ansible playbook to create a user on all managed nodes

vim playbook1.yml

---

- name: Create a user

hosts: all

tasks:

- name: User Creation

user:

name: Hari

password: lavelup

uid: 1256

home: /home/Hari

shell: /bin/bash

comment: "A regular user"

...

To check if the playbook is syntactically correct

ansible-playbook playbook1.yml --syntax-check

To execute the playbook

ansible-playbook playbook1.yml -b

==============================================================

Ansible playbook for configuring apache2

vim playbook2.yml

---

- name: Configuring apache2

hosts: all

tasks:

- name: Install apache2

apt:

name: apache2

state: present

update\_cache: yes

- name: Edit the index.html file

copy:

content: "Welcome to Apache"

dest: /var/www/html/index.html

- name: Restart apache2

service:

name: apache2

state: restarted

- name: Check apache response on server1

uri:

url: http://172.31.23.20

status\_code: 200

- name: Check apache response on server2

uri:

url: http://172.31.28.60

status\_code: 200

=================================================================

Ansible playbook for configuring tomcat9

vim playbook3.yml

---

- name: Configuring tomcat

hosts: all

tasks:

- name: Install tomcat9

apt:

name: tomcat9

state: present

- name: Copy the tomcat-users.xml file

copy:

src: tomcat-users.xml

dest: /etc/tomcat9

- name: Change port of tomcat from 8080 to 9090

replace:

regexp: 8080

replace: 9090

path: /etc/tomcat9/server.xml

- name: Restart tomcat9

service:

name: tomcat9

state: restarted

- name: Pause for 5 min

pause:

minutes: 5

- name: Check url response of tomcat on server1

uri:

url: http://172.31.28.60:9090

status\_code: 200

- name: Check url response of tomcat on server2

uri:

url: http://172.31.23.20:9090

status\_code: 200

...

=======================================================================

Ansible playbook to copy file from one managed node to another

vim playbook4.yml

---

- name: Fetch file from server1 to controller

hosts: 172.31.23.20

tasks:

- name: Fetch file

fetch:

src: file2

dest: /tmp

- name: Copy file to server2

hosts: 172.31.28.60

tasks:

- name: Copy file

copy:

src: /tmp/172.31.23.20/file2

dest: /home/ubuntu

...

==============================================================

Variables in Ansible

========================

Ansible uses three types of variables

1 Global Scope variabels

2 Host Scope Variables

3 Play Scope Varibales

Global Scope Variables

============================

These variables are passed from the command prompt using "--extra-vars"

and they have the highest priority

Ansible playbook to install or uninstall s/w applications

vim playbook5.yml

---

- name: Install s/w packages using variables

hosts: all

tasks:

- name: Install/Uninstall

apt:

name: "{{a}}"

state: "{{b}}"

update\_cache: "{{c}}"

...

To run the playbook for installing tree

ansible-playbook playbook5.yml --extra-vars "a=tree b=present c=no" -b

We can use the same playbook to uninstall git

ansible-playbook playbook5.yml --extra-vars "a=git b=absent c=no" -b

===========================================================================

vim playbook6.yml

---

- name: Create users and create files/dirs in the user home dir

hosts: all

tasks:

- name: Create user

user:

name: "{{a}}"

password: "{{b}}"

home: "{{c}}"

- name: Create files/dirs in users home dir

file:

name: "{{d}}"

state: "{{e}}"

...

To create users from the above playbook

ansible-playbook playbook6.yml --extra-vars "a=Ramesh b=password c=/home/Ramesh d=/home/Ramesh/file1 e=touch" -b

ansible-playbook playbook6.yml --extra-vars "a=Radha b=password c=/home/ubuntu/Radha d=/home/ubuntu/Radha/dir1 e=directory" -b

**ANSIBLE-04**

Playscope Variables

=======================

These variables are defined in the playbook in the "vars" section and

they have the least priority

vim playbook7.yml

---

- name: Install some s/w applications

hosts: all

vars:

- a: tomcat9

- b: present

- c: yes

tasks:

- name: Install/uninstall

apt:

name: "{{a}}"

state: "{{b}}"

update\_cache: "{{c}}"

...

The above playbook works like a template whose default behaviour is

to install tomcat9 but we can use the same playbook and make it

work on some other s/w application

ansible-playbook playbook7.yml --extra-vars "a=tree" -b

====================================================================

Host Scope variables

============================

These variables are classified into 2 type

1 Variables to work on a group of hosts

2 Variables to work on a single host

Variables to work on a group of hosts

=============================================

These variables are created ina folder called group\_vars

This folder should be created in the location where the playbooks

are present and in the group\_vars we create a file whose name is

same as group name from inventory file

1 Move into the folder where the playbooks are present

cd path\_of\_playbooks\_folder

2 Create a folder group\_vars and move into it

mkdir group\_vars

cd group\_vars

3 Create a file whose name is same as some group name in inventory file

vim webserver

---

a: firewalld

b: present

c: yes

...

4 Move back to the playbooks folder

cd ..

5 Create a playbook to use the above varibales

vim playbook8.yml

---

- name: Install firewall using host scope variables

hosts: webserver

tasks:

- name: Install firewalld

apt:

name: "{{a}}"

state: "{{b}}"

update\_cache: "{{c}}"

6 To execute the playbook

ansible-playbook playbook8.yml -b

===============================================================

Variable to work on a single host

======================================

These variables are created in host\_vars folder and this folder

should be created in the fodler where the playbooks are present

In this folder create a file whose name is ipaddress of one managed

node

1 Move into the playbooks folder

cd path\_of\_playbooks\_folder

2 Create a folder called host\_vars and move into it

mkdir host\_vars

cd host\_vars

3 Create file whose name is same as ipaddress of one managed node

vim 172.31.95.178

---

a: Radha

b: levelup

c: /home/Radha

d: /bin/bash

4 Move back to the playbooks folder

cd ..

5 Create aplybook to use the above varibales

vim playbook9.yml

---

- name: User create uisng hsot scope varibales

hosts: 172.31.95.178

tasks:

- name: USer creation

user:

name: "{{a}}"

password: "{{b}}"

home: "{{c}}"

shell: "{{d}}"

...

6 To execute the playbook

ansible-playbook playbook9.yml -b

==================================================================

Loops in ansible

====================

Loops can be implemented using "with\_items" and

"with\_sequence"

Ansible playbook to install s/w applications

vim playbook10.yml

---

- name: Install s/w applications

hosts: all

tasks:

- name: Install s/w

apt:

name: "{{item}}"

state: present

update\_cache: no

with\_items:

- tree

- git

- apache2

...

========================================================================

Ansible playbook to install/uninstall/upgrade s/w applications

vim playbook11.yml

---

- name: Install s/w applications

hosts: all

tasks:

- name: Install s/w

apt:

name: "{{item.a}}"

state: "{{item.b}}"

update\_cache: "{{item.c}}"

with\_items:

- {a: tree,b: present,c: no}

- {a: git,b: absent,c: no}

- {a: apache2,b: latest,c: yes}

...

===================================================================

vim playbook12.yml

---

- name: Create multiple users and copy files into user home dirs

hosts: all

tasks:

- name: Create users

user:

name: "{{item.a}}"

password: "{{item.b}}"

home: "{{item.c}}"

with\_items:

- {a: Rajesh,b: password,c: /home/Rajesh}

- {a: Usha,b: password,c: /home/ubuntu/Usha}

- name: Copy files into users home dirs

copy:

src: "{{item.a}}"

dest: "{{item.b}}"

with\_items:

- {a: /etc/passwd,b: /home/Rajesh}

- {a: /etc/group,b: /home/ubuntu/Usha}

**ANSIBLE-05**

Tags in Ansible

=======================

Tags are used to get more modular control on the execution

of the playbooks

vim playbook14.yml

---

- name: Implementing tags

hosts: all

tasks:

- name: Install tree

apt:

name: tree

state: present

tags: tree\_installation

- name: Create user

user:

name: Ravi

password: levelup

tags: user\_creation

- name: Copy /etc/passwd file

copy:

src: /etc/passwd

dest: /tmp

...

To execute only the the tagged modules

ansible-playbook playbook14.yml --tags=tagged -b

To execute only the untagged modules

ansible-playbook playbook14.yml --tags=untagged -b

To execute modules with a specific tag name

ansible-playbook playbook14.yml --tags=user\_creation -b

===================================================================

When conditions

======================

These are similar to "if" condiotions ie we want to execute

a module when a specific condition is true

---

- name: Implementing when conditions

hosts: all

vars:

- a: 10

tasks:

- name: Download from git

git:

repo: git-url

dest: /tmp/git

when: a == 10

...

=======================================================================

---

- name: Check if file f1 is present if not create a folder f1

hosts: all

tasks:

- name: check if f1 file is present

stat:

path: /tmp/f1

register: a

- name: Display output of the above module

debug:

var: a

- name: Create dir f1 if file f1 is not present

file:

name: /tmp/f1

state: directory

when: a.stat.exists == false

...

==========================================================

Handlers

================

1 Handlers are modules that are executed if some other module

is executed successfully and it has made some changes

2 Handlers are executed only after all the tasks are executed

3 Handlers are executed in the order that they are mentioned

in the handlers section and not in the order that they are

called in the tasks section

4 Even if a handler is called multiple times in the task section

it will be executed only once

vim playbook17.yml

---

- name: Implementing Handlers

hosts: all

tasks:

- name: Install apache2

apt:

name: apache2

state: present

notify: Check url response

- name: Edit the index.html file

copy:

content: "IntelliQIT Home Page"

dest: /var/www/html/index.html

notify: Restart apache2

handlers:

- name: Restart apache2

service:

name: apache2

state: restarted

- name: Check url response

uri:

url: "{{item}}"

status\_code: 200

with\_items:

- http://172.31.95.178

- http://172.31.88.236

...

**ANSIBLE-06**

Error Handling or Exception Handling

========================================

If we want the ansbile playbook to continue its execution

even if a specific module fails then we can use Error Handling

The section of code that might fail is given in the "block" section

If it fails the control goes to the "rescue" section,"always"

section is executed everytime

vim playbook18.yml

---

- name: Setup of tomcat

hosts: all

tasks:

- block:

- name: Install tomcat8

apt:

name: tomcat8

state: present

rescue:

- name: Install tomcat9

apt:

name: tomcat9

state: present

always:

- name: Display tomcat success msg

debug:

msg: "Tomcat setup is successfull"

...

=================================================================

Ansible playbook to install git using apt on ubuntu based machines

and yum on centos based machines

vim playbook15.yml

---

- name: Install using apt or yum

hosts: all

tasks:

- block:

- name: Install git using apt

apt:

name: git

state: present

rescue:

- name: Install git using yum

yum:

name: git

state: present

...

=================================================================

Ansible playbook for setting up Jenkins on one server and

tomcat on qa and prodservers

vim playbook.yml

---

- name: Install required s/w for Jenkins

hosts: jenkins

tasks:

- name: Install required s/w

apt:

name: "{{item.a}}"

state: present

update\_cache: "{{item.b}}"

with\_items:

- {a: openjdk-8-jdk,b: yes}

- {a: git,b: no}

- {a: maven,b: no}

- name: Download jenkins.war

get\_url:

url: https://get.jenkins.io/war-stable/2.277.2/jenkins.war

dest: /home/ubuntu

- name: Install tomcat and tomcat-admin

hosts: servers

tasks:

- name: Install tomcat9 and tomcat9-admin

apt:

name: "{{item.a}}"

state: present

update\_cache: "{{item.b}}"

with\_items:

- {a: tomcat9,b: yes}

- {a: tomcat9-admin,b: no}

- name: Copy the tomcat-users.xml file

copy:

src: tomcat-users.xml

dest: /etc/tomcat9

notify: Restart tomcat9

handlers:

- name: Restart tomcat9

service:

name: tomcat9

state: restarted

...

------------------------------------------------------------

========================================================================

include module

========================

This is a module of Ansible that is used to call child playbooks

Create a child playbook for user creation anc all if from a

parent playbook

Child playbook

======================

vim playbook22.yml

---

- name: Create a user

user:

name: Ramesh

password: levelup

uid: 1590

home: /home/Ramesh

shell: /bin/bash

comment: "A regular user"

...

Parent Playbook

=======================

vim playbook23.yml

---

- name: USer creation using child playbooks

hosts: all

tasks:

- name: call child playbook

include: playbook22.yml

=======================================================================

=====================================================================

Configuring apache2 using child playbooks

Childplaybooks

=================

vim install\_apache.yml

---

- name: Install apache2

apt:

name: apache2

state: present

update\_cache: yes

...

vim edit\_index.yml

---

- name: Edit index.html file

copy:

content: "Welcome"

dest: /var/www/html/index.html

...

vim service.yml

---

- name: Restart apache2

service:

name: apache2

state: restarted

...

vim check\_url\_response.yml

---

- name: Check url response of apache2 on all managed nodes

uri:

url: "{{item}}"

status\_code: 200

with\_items:

- http://172.31.89.80

- http://172.31.30.86

...

Parent playbook

==================

vim configure\_apache.yml

---

- name: Configuring apache using child playbooks

hosts: all

tasks:

- name: Call multiple child playbooks

include: "{{item}}"

with\_items:

- install\_apache.yml

- edit\_index.yml

- service.yml

- check\_url\_response.yml

...

To run the playbook

ansible-playbook configure\_apache.yml -b

**ANSIBLE-07**

Ansible Vault

===================

This is a feature of ansible which allows us to protect the playbooks

via a password.Playbooks created using vault can be viewed,edited or

executed only if we know the password

1 To create a vault playbook

ansible-vault create playbook\_name.yml

2 To view the content of a vault playbook

ansible-vault view playbook\_name.yml

3 To edit the content of a vault playbook

ansible-vault edit playbook\_name.yml

4 To convert an ordinary playbook into a vault playbook

ansible-vault encrypt playbook\_name.yml

5 To convert a vault playbook into an ordinary playbook

ansible-vault decrypt playbook\_name.yml

6 To reset the password of a vault playbook

ansible-vault rekey playbook\_name.yml

===============================================================

======================================================================

Ansible Docker Automation:

---

- name: Install python-pip,docker,docker-py

hosts: all

tasks:

- name: Install python-pip

apt:

name: python3-pip

state: present

- name: install docker-py, download docker and install docker

shell: "{{item}}"

with\_items:

- pip install docker-py

- curl -fsSL https://get.docker.com -o get-docker.sh

- sh get-docker.sh

======================================================================

---

- name: download postgres image

hosts: all

tasks:

- name: download postgre

docker\_image:

name: pacur/centos-7

state: absent

======================================================================

---

- name: working on tomcat conatainers

hosts: all

tasks:

- name: start tomcat as container

docker\_container:

image: tomcat

name: mytomcat

- name: stop the container

docker\_container:

name: mytomcat

state: stopped

- name: Remove tomcat container

docker\_container:

name: mytomcat

state: absent

===================================================================

---

- name: start 10 nginx containers

hosts: all

tasks:

- name: start nginx

docker\_container:

image: nginx

name: "nginx{{item}}"

with\_sequence: count=10

==================================================================

---

- name: start ubuntu as a container

hosts: all

tasks:

- name: start ubuntu container

docker\_container:

image: ubuntu

name: c1

interactive: yes

tty: yes

volumes:

- /data

==================================================================

---

- name: implement docker-compose

hosts: all

tasks:

- name: start mysql as a container

docker\_container:

image: mysql:5

name: mydb

env:

MYSQL\_ROOT\_PASSWORD: password

- name: start wordpres container

docker\_container:

image: wordpress

name: mywordpress

ports:

- 9999:80

links:

- "mydb:mysql"

====================================================================

---

- name: create ci-cd architecture

hosts: all

tasks:

- name: start jenkins

docker\_container:

image: jenkins/jenkins

name: devserver

ports:

- 8181:8080

- name: start tomcat as acontainer

docker\_container:

image: tomcat

name: "{{item.a}}"

ports:

"{{item.b}}"

links:

- "devserver:jenkins"

with\_items:

- {a: qaserver, b: "5050:8080"}

- {a: prodserver, b: "6060:8080"}

==========================================================================

---

- name: push image to docker hub

hosts: 172.31.24.221

tasks:

- name: login into docker hub

docker\_login:

username: dockerhub\_username

password: Hpassword

- name: push the image

docker\_image:

name: username/image\_name

push: yes

source: local

===========================================================================

**ANSIBLE-08**

=======================================================================

Roles in Ansibles

========================

Roles provide greater reusability then playbooks

Generally roles are used to configure s/w applications

Everything necessary to configure a s/w applications should be

present with the folder structure of a role

This aids in easy understanding and maintainance of CM activites

Roles should be create in /etc/ansible/roles folder

To create roles in some other locations

sudo vim /etc/ansible/ansible.config

Search for roles\_path and give the path of the directory where

we want to create the role and uncomment it

Folder structrue of roles

================================

README.MD : This is a simple text file that is used to store info about

the role in plain English

defaults: This stores info about the application that we are configuring

and it also stores varibales of lesser priority

files: All the static files that are required for configuring a s/w application

are stored here

meta: Data about the data is called as metadata and this is used to store info about

the roles like when it was created who created it what versions it supports etc

handlers: handlers are modules that are executed when some other module is

successfull and it has made some changes,all such handlers are stored in

this folder

tasks: The actual configuration management activity that has to be perfromed on the

remote servers is stored in this folder

templates: This is used to store dynamic configuration files

tests: All the modules that are used to check if the remote configurations

are successfull or not are stored in this folder

vars: This is used to store all the variables that are required for configuring

a specific s/w application.These variables have higher priority than the

variables in defaults folder.

Apache Role

========================

1 Go into the /etc/ansible/roles folder

cd /etc/ansible/roles

2 Create a new role for apache2

ansible-galaxy init apache2 --offline

3 check the tree structure of the role that we created

tree apache2

4 Go to tasks folder in role and create the task for configuring apache2

cd apache2/tasks

vim main.yml

---

- include: install.yml

- include: configure.yml

- include: check\_url\_response.yml

...

Save and quit Esc :wq Enter

vim install.yml

---

- name: install apache2

apt:

name: apache2

state: present

Save and quit Esc :wq Enter

vim configure.yml

---

- name:copy index.html

copy:

src: index.html

dest: /var/www/html/index.html

notify:

Restart apache2

...

Save and quit Esc :wq Enter

vim check\_url\_response.yml

---

- name: Check url response

uri:

url: "{{item}}"

status: 200

with\_items:

- http://172.31.18.210

- http://172.31.31.227

...

Save and quit Esc :wq Enter

Go to files folder to create the index.html file

cd ..

cd files

sudo vim index.html

<html>

<body>

<h1>This is levelup</h1>

</body>

</html>

Save and quit Esc :wq Enter

Go to handlers folder

cd ..

cd handlers

sudo vim main.yml

---

# handlers file for apache2

- name: Restart apache2

service:

name: apache2

state: restarted

...

Save and quit Esc :wq Enter

CREATE the parent playbook to call the roles

cd ..

cd ..

sudo vim apache\_role.yml

---

- name: Implementing roles for apache2

hosts: all

roles:

- apache2

...

Save and quit Esc :wq Enter

To execute the role

ansible-playbook apache\_role.yml -b

===============================================================================

Creating roles for tomcat

---------------------------

1 cd /etc/ansible/roles

2 ansible-galaxy init tomcat --offline

3 Create tasks for tomcat

a) cd tomcat/tasks

b) sudo vim main.yml

---

- name: Calling child playbooks

include: "{{item}}"

with\_items:

- install.yml

- configure.yml

- restart.yml

...

Save and quit

c) sudo vim install.yml

---

- name: Installing tomcat8 and tomcat8-admin

apt:

name: "{{item.a}}"

state: "{{item.b}}"

update\_cache: "{{item.c}}"

with\_items:

- {a: "{{pkg1}}",b: "{{state1}}",c: "{{cache1}}"}

- {a: "{{pkg2}}",b: "{{state1}}",c: "{{cache2}}"}

...

d) sudo vim configure.yml

---

- name: Copy tomcat-user.xml

copy:

src: "{{file1}}"

dest: "{{destination1}}"

- name: Change port of tomcat from 8080 to 9090

replace:

path: "{{path1}}"

regexp: "{{port1}}"

replace: "{{port2}}"

notify:

- check\_url\_response

...

e) sudo vim restart.yml

---

- name: Restart tomcat8

service:

name: "{{pkg1}}"

state: "{{state3}}"

...

4) Create the handlers

cd ..

cd handlers

sudo vim main.yml

---

# handlers file for tomcat

- name: check\_url\_response

uri:

url: "{{item.a}}"

status: "{{item.b}}"

with\_items:

- {a: "{{server1}}",b: "{{status1}}"}

- {a: "{{server2}}",b: "{{status1}}"}

...

5) create static files

cd ..

cd files

a) sudo vim tomcat-users.xml

<tomcat-users>

<user username="admin" password="password" roles="manager-script"/>

</tomcat-users>

Save and quit

6) Define the variables

cd ..

cd vars

sudo vim main.yml

---

# vars file for tomcat

pkg1: tomcat8

pkg2: tomcat8-admin

state1: present

state2: absent

state3: restarted

cache1: yes

cache2: no

file1: tomcat-users.xml

destination1: /etc/tomcat8

server1: http://172.31.87.8:9090

server2: http://172.31.84.59:9090

status1: 200

status2: -1

path1: /etc/tomcat8/server.xml

port1: 8080

port2: 9090

...

7 Come out of the tomcat roles

cd ../..

8 Create a playbook to call that role

sudo vim configure\_tomcat.yml

---

- name: Configuring tomcat using roles

hosts: all

roles:

- tomcat

...

9 To run the playbook for the above role

ansible-playbook configure\_tomcat.yml -b