**ANSIBLE**

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

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

This is an open source configuration management tool created using python

The main machine where ansible is installed is called as "Controller"

and the remianing remote servers that we are configuring are called as

"managed nodes/hosts"

From the controller to the managed nodes we should have passwordless

shh connectivity

Ansible is called as "agentless" ie we need not install any client

s/w of ansible on the remote managed nodes.It uses "push" methodolgy

to push the configurations into the remote servers.

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

Setup of Ansible

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

1 Create 3 or 4 AWS ubuntu 18 instances

2 NAme the 1st one as controller and remaining 2 as server1 and server2

3 Establish Passwordless ssh from Controller to Server1 and Server2

a) Connect to server1 using gitbash

b) Setup password for the default user

sudo passwd ubuntu

c) Edit the ssh configuration file

sudo vim /etc/ssh/sshd\_config

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

d) Restart ssh

sudo service ssh restart

Repeat the above steps from a to d on Server2 managed node

e) Connect to Controller using git bash

f) Generate the ssh keys

ssh-keygen

g) Copy the ssh keys

ssh-copy-id ubuntu@private\_ip\_of\_server1

Repeat step g with ipaddress of Server2

4 Installing Ansible

a) Update the apt repository

sudo apt-get update

b) Install software-properties-common

sudo apt-get install -y 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

5 To check the verision of ansible

ansible --version

Ansible stores all the remote servers info in a file called as inventory file

We should open this file and store the ipaddress of all the managed nodes here

sudo vim /etc/ansible/hosts

Here copy and paste the ipaddresses of the managed nodes

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

Ansible performs remote configuration of servers in

3 different ways

1 Adhoc commands

2 Playbooks

3 Roles

Ansible uses prebuild Python modules for configuring remote

servers

Important modules in Ansible

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

1 command: This is used to execute linux commands on the remote managed

nodes.It is the default module of Ansible

2 shell: This is used to execute shell scripts on the remote managed nodes

it can execute command related to redirection and piping

3 user: This is used to perform user administartion on the remote servers

like creating users,assigning home dirs deleting users etc

4 file: Used for creating files/directories on the managed nodes

5 copy: This used to copy files/directories to the managed node

6 fetch: Used to copy files/directories from managed nodes to controller

7 apt: Used to for s/w package management like isntalling,deleting,upgrading

etc.It works on Ubuntu,Debain flvours of linux

8 yum: This is similar to apt but it works on Rehat linux,Centos,Fedora etc

flavours of Linux

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

10 uri: Used to check if a remote url is reachable or not

11 git: Used for perfroming git version controlling on the managed nodes

12 get\_url: Used for downloading files from remote servers into the managed nodes

13 stat: Used to capture detailed info about files/directories on the managed nodes

14 debug: Used to display the output in JSON file format

15 include: USed to call child playbooks from a parent playbook

16 replace: Used to replace specific portions of the text in a file

17 docker\_container: Used for container management on the managed nodes

18 docker\_image: Used to run command related to docker images

19 docker\_login: Used to login into the docker registry

20 docker\_swarm: Used to setup of docker swarm architecture

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

Adhoc command Syntax

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

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

CommandModule

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

Ansible command to see the memory info of all managed nodes

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

Note: /etc/ansible/hosts is the deafult inventory file and when working on it

we need not specify the -i option

ansible all -m command -a 'free -m'

Note: command module is the default module of Ansible and when working on it

we need not specify the -m option

ansible all -a 'free -m'

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

Shell Module

Ansible command to install docker on all managed nodes

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 store the memory info of all managed nodes in file1

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

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

UserModule

Ansible command to create a user and assign a password

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

Note: -b represents "become" it is used to giving higher previlages on the

remote managed nodes

User module can also assign home dirs ,default working shell ,uid etc

ansible all -m user -a 'name=Anu password=intelliqit uid=1234

home=/home/ubuntu/Anu shell=/bin/bash comment="A normal user"' -b

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

file module

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

Ansible command to create a file on all managed nodes

ansible all -m file -a 'name=/tmp/file14 state=touch'

Note: state= touch is for creating files

state=directory is for creating directories

state=absent is for deleting file/directories

Ansible command to create a file and also change the premissions

ownership and groupship

ansible all -m file -a 'name=/home/ubuntu/file56 state=touch

owner=sai group=Anu mode=770' -b

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

Copy Module

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

Ansible command to copy a file from controller to all managed nodes

ansible all -m copy -a 'src=file100 dest=/tmp'

Ansible command to copy a file and also change permissions ownership and group ownership

ansible all -m copy -a 'src=file100 dest=/tmp owner=root group=sai mode=764' -b

Copy module can also replace the existing content of a file

ansible all -m copy -a 'content="Hello IntelliQ\n" dest=file1'

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

apt Module

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

Ansible command to install tree on all managed nodes

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

Note: state=present for installing

state=absent for uninstalling

state=latest for upgrading to the latest version

Ansible command to uninstall git from all managed nodes

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

To update the apt repository we use

update\_cache=yes

Ansible comamnd to install tomcat9 after updating the apt repository

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

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

Service Module

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

Ansible command to restart ssh service

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

Note: state=restarted for restarting services

state=started for starting services

state=stopped for stopping services

Install tomcat9 ,copy tomcat-users.xml file and restart tomcat

1 Install tomcat9

ansible all -m apt -a 'name=tomcat9 state=present' -b

2 Create tomcat-users.xml file

vim tomcat-users.xml

<tomcat-users>

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

</tomcat-users>

3 Copy the tomcat-users.xmlf file the required location

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

4 Restart tomcat9

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

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

get\_url Module

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

Ansible command to downlaod jenkins.war into all managed nodes

ansible all -m get\_url -a

'url=http://mirrors.jenkins.io/war-stable/2.235.3/jenkins.war dest=/tmp'

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

Replace Module

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

Ansible command to change port of tomcat9 from 8080 to

9090 and restart tomcat9

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

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

git module

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

Ansible command to download from a remote git repository

ansible all -m git -a 'repo=https://github.com/intelliqittrainings/maven.git dest=/tmp/mygit' -b

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

uri module

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

Ansible command to check if google.com is reachable from all managed nodes

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

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

Configure apache2 on all managed nodes

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

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 to IntelliqIT" 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 all -m uri -a 'url=http://172.31.23.20 status\_code=200'

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

Configuring tomcat9

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

1 Install tomcat9 and tomcat9-admin

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

ansible all -m apt -a 'name=tomcat9-admin state=present' -b

2 Copy the tomcat-users.xml file

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

3 Restart tomcat

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

4 Check the url reponse of tomcat

ansible all -m uri -a 'url=http://172.31.28.60:8080 status\_code=200' -b

ansible all -m uri -a 'url=http://172.31.23.20:8080 status\_code=200' -b

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

Ansible Playbooks

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

Adhoc commands become difficutl to handle when working on complex

configurations of s/w applications.

Each adhoc command can work only on one module and one set of

arguments.In such cases we can use Ansible playbooks which

support greater reusability.

Playbooks are created using yaml and each playbook is a combination of

multiple plays.A play contains info about what module has to be

executed.These plays are designed to work on a single host or a

group of hosts or all the hosts

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

Anible playbook to create a user on all managed nodes

vim playbook1.yml

---

- name: Create user

hosts: all

tasks:

- name: User creation

user:

name: Anu

password: intelliqit

uid: 3456

home: /home/ubuntu/Anu

comment: "A regular user"

shell: /bin/bash

...

To check if the playbook is syntaxtically correct or not

ansible-playbook playbook1.yml --syntax-check

To execute the playbook

ansible-playbook playbook1.yml -b

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

Ansible playbook to configure 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: "IntelliQIT"

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

- name: Restart apache2

service:

name: apache2

state: restarted

- name: Check the url response of apache2 on server1

uri:

url: http://172.31.18.115

status\_code: 200

- name: Check the url response of apache2 on server2

uri:

url: http://172.31.30.86

status\_code: 200

...

To run the playbook

ansible-playbook playbook2.yml -b

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

Ansible playbook to configure tomcat9

- name: Configuring tomcat

hosts: all

tasks:

- name: Install tomcat9

apt:

name: tomcat9

state: present

update\_cache: yes

- name: Install tomcat9-admin

apt:

name: tomcat9-admin

state: present

update\_cache: no

- name: Copy tomcat-users.xml

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 3 mins

pause:

minutes: 3

- name: Check tomcat response on server1

uri:

url: http://172.31.30.86:9090

status\_code: 200

- name: Check tomcat response on server2

uri:

url: http://172.31.18.115:9090

status\_code: 200

...

To execute the playbook

ansible-playbook playbook3.yml -b

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

Variables in Ansible

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

Variables are categorised into 3 type

1 Global scope varaibles

2 Host Scope variables

3 Play scope variables

Global scope variables

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

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

and they have the highest level of priority

Ansible playbook to install or uninstall various s/w applications

vim playbook4.yml

---

- name: Install s/w applications

hosts: all

tasks:

- name: Install/uninstall s/w

apt:

name: "{{a}}"

state: "{{b}}"

update\_cache: "{{c}}"

...

To run the above playbook to uninstall git

ansible-playbook playbook4.yml --extra-vars "a=git b=absent c=no" -b

We can use the same playbook to work on some other set of s/w's like install java

ansible-playbook playbook4.yml --extra-vars "a=openjdk-8-jdk b=present c=no" -b

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

Ansible playbook to create users and files/dirs in users home dir

vim playbook5.yml

---

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

hosts: all

tasks:

- name: Create users

user:

name: "{{a}}"

password: "{{b}}"

home: "{{c}}"

- name: Create files/dirs in users home die

file:

name: "{{d}}"

state: "{{e}}"

...

To create multiple users and files/dirs

ansible-playbook playbook5.yml --extra-vars "a=Raju b=intelliqit

c=/home/Raju d=/home/Raju/file1 e=touch" -b

ansible-playbook playbook5.yml --extra-vars "a=Rani b=intelliqit

c=/home/Rani d=/home/Rani/dir1 e=directory" -b

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

Playscope varibles

These varibales are defined within a playbook and they have the

least priority

vim playbook6.yml

---

- name: Install/unistall sw applications

hosts: all

vars:

- a: tomcat9

- b: present

- c: no

tasks:

- name: Install/unisntall

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 make it work on some other application

by passing global scope variables

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

Grouping in inventory file

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

sudo vim /etc/ansible/hosts

[webserver]

172.31.30.86

172.31.18.115

[appserver]

172.31.92.137

[dbserver]

172.31.86.213

172.31.18.115

[server:children]

appserver

dbserver

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

Host Scope Variables

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

These variables are further classified into 2 types

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 create in a directory "group\_vars"

This directory is created in the same folder where the playbooks

are present.In the group\_vars directory we create a file whose

name is same as group name from the inventory file

1 Go to the folder where the playbooks are present

cd path\_of\_playbooks\_folder

2 Create a directory group\_vars and move into it

mkdir group\_vars

cd group\_vars

3 Create a file whose name is same as a group name from the inventory file

vim webserver

---

a: firewalld

b: present

c: no

...

4 Go back to the folder where the playbooks are present

cd ..

5 Create a playbook for using the above variables

vim playbook8.yml

---

- name: Install firewall using host scope variables

hosts: webserver

tasks:

- name: Install firewall

apt:

name: "{{a}}"

state: "{{b}}"

update\_cache: "{{c}}"

...

6 To execute the playbook

ansible-playbook playbook8.yml -b

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

Variables to work on a single host

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

These variables should be created in a file whose name is same as ip

address of a remote managed node and this file should be created in

a folder called "host\_vars" and this folder should be created in the folder

where all our playbooks are present

1 Go to the folder where the playbooks are present

cd path\_of\_playbooks\_folder

2 Create a directory host\_vars and move into it

mkdir host\_vars

cd host\_vars

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

from the inventory file

vim 172.31.56.218

---

a: Radha

b: intellqiit

c: 1243

d: /home/Radha

e: /bin/bash

...

4 Go back to the folder where the playbooks are present

cd ..

5 Create a playbook to use the above varibles

vim playbook9.yml

---

- name: create user using host scope varibles

hosts: 172.31.56.218

tasks:

- name: create user

user:

name: "{{a}}"

password: "{{b}}"

uid: "{{c}}"

home: "{{d}}"

shell: "{{e}}"

...

6 To execute the playbook

ansible-playbook playbook9.yml -b

...

6 To run the playbook

ansible-playbook playbook10.yml -b

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

Loops in ansible can be inmplemented using

with\_items,with\_sequence

Ansible playbook to install multiple s/w applications using with\_items

---

- name: Installing s/w applications

hosts: all

tasks:

- name: Install multiple s/w applications

apt:

name: "{{item}}"

state: present

update\_cache: no

with\_items:

- tree

- git

- maven

...

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

Alternate approach for the above playbook

---

- name: Install various s/w applications

hosts: all

tasks:

- name: Install tree

apt:

name: ["tree","git","maven"]

state: present

update\_cache: no

...

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

Ansible playbooks to install uninstall multiple s/w applications

---

- name: Installing/uninstalling/upgrading s/w applications

hosts: all

tasks:

- name: Install multiple s/w applications

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: maven,b: latest,c: yes}

...

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

Configuring apache2

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

---

- name: Configuring apache

hosts: all

tasks:

- name: Install tomcat9

apt:

name: apache2

state: present

update\_cache: yes

- name: Edit the idnex.html file

copy:

content: "IntelliQIT"

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

- name: Restart apache2

service:

name: apache2

state: restarted

- name: Check apache2 url response

uri:

url: "{{item}}"

status\_code: 200

with\_items:

- http://172.31.55.129

- http://172.31.48.61

...

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

Tags are like alias to modules in ansible playbooks

Using tags we can get a bettwr control on the flow of

the playbook execution

vim playbook14.yml

---

- name: Tagging in Ansible

hosts: all

tasks:

- name: Install tree

apt:

name: tree

state: present

tags: tree\_installation

- name: Create user

user:

name: Anu

password: intelliqit

tags: user\_creation

- name: Copy /etc/passwd file

copy:

src: /etc/passwd

dest: /tmp

...

To execute only 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

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

Ansible playbook to setup CI-CD environment for jenkins

---

- name: Setup of jenkins and required s/w's

hosts: jenkinsserver

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: http://mirrors.jenkins.io/war-stable/2.235.5/jenkins.war

dest: /tmp

- name: Setup tomcat on qa and prodservers

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 tomcat-users.xml file

copy:

src: tomcat-users.xml

dest: /etc/tomcat9

- name: Restart tomcat9

service:

name: tomcat9

state: restarted

...

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

Handlers

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

1 Handlers are modules that are executed if some other module is executed

succesfully and it has made some changes.

2 Handlers are only executed after all the modules in the tasks section 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 tasks section it will

be executed only once

---

- name: Implementing handlers

hosts: all

tasks:

- name: Install apache2

apt:

name: apache2

state: present

notify: Check url response

- name: Edit index.html file

copy:

content: "Welcome to my IntelliQIT\n"

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://172.31.48.56

- http://172.31.36.172

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

When conditions

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

This is "if" condtions and it helps us to execute modules based on a specific

condition

Create a file based on a condition

---

- name: Implementing when conditions

hosts: all

vars:

- a: 10

tasks:

- name: Copy passwd file

copy:

src: /etc/passwd

dest: /tmp

when: a == 10

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

---

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

hosts: all

tasks:

- name: Check for f1 directory

stat:

path: /home/ubuntu/f1

register: a

- name: Display output of abouve module

debug:

var: a

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

file:

name: /home/ubuntu/f1

state: touch

when: a.stat.exists == false

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

---

- name: Check ofr execute permissions and modify the permissions

hosts: all

tasks:

- name: Get detailed inof about file1

stat:

path: /tmp/file1

register: b

- name: Display output of the above module

debug:

var: b

- name: Check for execute permissions and change it

file:

name: /tmp/file1

mode: 740

when: b.stat.executable == false

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

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

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

Error Handling

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

Whenever a module in ansible playbook fails the execution

of the playbook stops there,if we know that a spcific module

can fail and still we want to continue the execution of the

playbook we can use error handling

The module that might fail should be given in the "block"

section,if it fails the control comes to the "rescue" section

"always" section is executed everytime

Ansible playbook to install tomcat8 on all managed nodes if

it fails then it should install tomcat9

vim playbook19.yml

---

- name: Error handling or Exception Handling

hosts: all

tasks:

- block:

- name: Install tomcat8

apt:

name: tomcat8

state: present

update\_cache: yes

rescue:

- name: Install tomcat9

apt:

name: tomcat9

state: present

update\_cache: yes

always:

- name: Display output

debug:

msg: Tomcat setup successfull

...

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

Ansible playbook implement CI-CD

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

---

- name: Install required s/w's for ci-cd

hosts: all

tasks:

- name: Install s/w's

apt:

name: "{{item.a}}"

state: present

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

with\_items:

- {a: git,b: yes}

- {a: openjdk-8-jdk,b: no}

- {a: maven,b: no}

- {a: tomcat9,b: no}

- name: Continuous Download and Build

hosts: devserver

tasks:

- name: Download the code created by developers

git:

repo: https://github.com/intelliqittrainings/maven.git

dest: /tmp/mygit

- name: Create an artifact from the above code

shell: cd /tmp/mygit;mvn package

- name: Fetch the artifact from devserver to controller

fetch:

src: /tmp/mygit/webapp/target/webapp.war

dest: /tmp

- name: Continuous Deployment and Testing

hosts: qaserver

tasks:

- name: Deploy artifact into tomcat on QaServer

copy:

src: /tmp/172.31.16.122/tmp/mygit/webapp/target/webapp.war

dest: /var/lib/tomcat9/webapps/testapp.war

- name: Restart tomcat

service:

name: tomcat9

state: restarted

- name: Downlaod the selenium test scripts

git:

repo: https://github.com/intelliqittrainings/FunctionalTesting.git

dest: /tmp/test-git

- name: Execute the seclenium test scripts

shell: java -jar /tmp/test-git/testing.jar

- name: Continuous Delivery

hosts: prodserver

tasks:

- name: Deploy the artifact into prodserver tomcat

copy:

src: /tmp/172.31.16.122/tmp/mygit/webapp/target/webapp.war

dest: /var/lib/tomcat9/webapps/prodapp.war

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

include module

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

This is used to call child playbooks from the level of a parnet

playbook

Child playbook

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

vim playbook20.yml

---

- name: Copy /etc/passwd file

copy:

src: /etc/passwd

dest: /tmp

...

Parent playbook

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

vim playbook21.yml

---

- name: Call child playbooks

hosts: all

tasks:

- name: Call child playbook

include: playbook20.yml

...

To execute

ansible-playbook playbook21.yml -b

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

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: "New intelliqit"

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

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

Configuring tomcat using child playbooks and group variables

Child playbooks

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

vim install\_tomcat.yml

---

- name: Install tomcat9 and tomcat9-admin

apt:

name: "{{item.a}}"

state: "{{item.b}}"

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

with\_items:

- {a: "{{a}}",b: "{{b}}",c: "{{c}}"}

- {a: "{{d}}",b: "{{b}}",c: "{{e}}"}

...

vim copy\_tomcat\_users.xml

---

- name: Copy tomcat-users.xml file

copy:

src: "{{f}}"

dest: "{{g}}"

...

vim change\_port.yml

---

- name: Change port of tomcat from 8080 to 9090

replace:

regexp: "{{h}}"

replace: "{{i}}"

path: "{{j}}"

...

vim restart\_tomcat.yml

---

- name: Restart tomcat9

service:

name: "{{a}}"

state: "{{k}}"

...

...

vim url\_response\_tomcat.yml

---

- name: Check url response of tomcat

uri:

url: "{{item.a}}"

status\_code: "{{item.b}}"

with\_items:

# - {a: http://172.31.48.61:9090,b: 200}

# - {a: http://172.31.55.129:9090,b: 200}

- {a: "{{l}}",b: "{{m}}"}

- {a: "{{n}}",b: "{{m}}"}

...

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

Parent playbooks

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

vim configure\_tomcat.yml

---

- name: Configuring tomcat using child playbooks

hosts: servers

tasks:

- name: Call child playbooks for tomcat

include: "{{item}}"

with\_items:

- install\_tomcat.yml

- copy\_tomcat\_users.yml

- change\_port.yml

- restart\_tomcat.yml

- url\_response\_tomcat.yml

...

Creating variables for the above playbooks

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

1 create a directory group\_vars and move into it

mkdir group\_vars

cd group\_vars

2 Create a file called servers to store the variables

vim servers

a: tomcat9

b: present

c: yes

d: tomcat9-admin

e: no

f: tomcat-users.xml

g: /etc/tomcat9

h: 8080

i: 9090

j: /etc/tomcat9/server.xml

k: restarted

l: http://172.31.55.129:9090

m: 200

n: http://172.31.48.61:9090

...

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

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

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 IntelliQ</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="intelliq" password="myintelliq" 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

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