**ANSIBLE**

Day 1

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

Configuration Management

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

This is the process of handling activities of multiple

servers from one point of control

Advantages

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

1 Provisioning of Servers

Activities like installing s/w applications, deleting s/w, upgrading

configuring patches can be done very quickly

2 Resource Saving

The time taken to configure any number of servers become very less

and the number of people who are required for these server configurations

also becomes less

3 Useful in Disaster Recovery

To handle disaster recovery organizations maintain replica data centers

at different geographical locations. Creation of these replica data centers

can be done very easily

4 Handling snowflake servers

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

snowflakes i.e they run on slightly different h/w and s/w configurations.

Configuration Management tools can pick up this info and store in simple

config files which can be used later to setup similar environments

5 Idempotent

Configuration management tools are used to bring the remote to a state

called as the "desired state". If the remote servers are already in the

desired state configuration management tools will not reconfigure these

server

Important Configuration Management Tools

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

1 Ansible

2 Chef

3 Puppet

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 remaining 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

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

Day 2

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

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 administration 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 for s/w package management like installing, deleting, upgrading

etc. It works on Ubuntu, Debain flavours 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 performing 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'

|  |  |  |
| --- | --- | --- |
| Sr. no. | Module | description |
| 1 | command | To run any linux command on slaves.  Its default module of Ansible hence no need to specify  e.g  ansible all -a 'free -m' |
| 2 | shell | To run any linux script on slaves.  e.g Ansible command to store the memory info of all managed nodes in file1🡪  ansible all -m shell -a 'free -m > file1' |
| 3 | user | to create a user and assign a password, assign home dirs, default working shell, uid etc  e.g  ansible all -m user -a 'name=Anu password=intelliqit uid=1234  home=/home/ubuntu/Anu shell=/bin/bash comment="A normal user"' -b |
| 4 | file | to create a file on all managed nodes  e.g  ansible all -m file -a 'name=/tmp/file14 state=touch' |
| 5 | copy | to copy a file from controller to all managed nodes  e.g  ansible all -m copy -a 'src=file100 dest=/tmp' |

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 default 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

To check user actually created or not run following command in slave

cat /etc/passwd

Note: -b represents "become" it is used to giving higher privileges 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 permissions

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'

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

Day 3

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

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

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 command 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

Uri Module

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

Ansible command to check if the managed nodes are able to reach google.com

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

Note: status\_code=200 is success

status\_code=-1 is failure

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

git Module

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

Install git on all managed nodes and clone a remote git repository

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

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

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

fetch Module

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

Used to copy files from managed nodes to controller

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

UseCase

===========

Ansible command to copy files from one managed node to another

We have to use a combination of fetch and copy modules to perfrom this task

1 Create a file on first managed node

2 Fetch that file into the controller

ansible ipaddress\_of\_first\_managednode -m fetch -a 'src=path\_of\_file

dest=/tmp'

3 Copy thie into second managed node

ansible ipaddress\_of\_Second\_managednode -m copy -a 'src=path\_of\_file dest='/tmp'

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

Day 4

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

Configuring apache2

1 Install apache2

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

2 Edit the index.html file

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

3 Restart apache2

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

4 Check apache2 url response from all managed nodes

ansible all -m uri -a 'url=http://172.31.37.220 status\_code=200'

ansible all -m uri -a 'url=http://172.31.36.172 status\_code=200'

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

Configuring tomcat9

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

1 Install tomcat9 on all managed nodes

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

2 Copy tomcat-users.xml file

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

3 Change port from 9090 to 8080

ansible all -m replace -a 'regexp=9090 replace=8080 path=/etc/tomcat9/server.xml' -b

4 Restart tomcat9

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

5 Check the url response of tomcat9

ansible all -m uri -a 'url=http://172.31.55.129:8080 status\_code=200'

ansible all -m uri -a 'url=http://172.31.48.61:8080 status\_code=200'

Web server

* Apache2: On Ubuntu
* HTTPD: On RHEL
* Apache and tomcat both use for same purpose 🡪 hosting applications

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

Ansible Playbooks

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

Adhoc commands become difficult 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

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

XML and YAML files for storing data

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

<intelliqit>

<trainers>

<devops>sai</devops>

<azure>ramesh</azure>

</trainers>

<coordinators>

<devops>shalini</devops>

<azure>sania</azure>

</coordinators>

</intelliqit>

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

---

intelliqit:

trainers:

devops: sai

azure: ramesh

coordinators:

devops: shalini

azure: sania

...

Ansible playbook to create a user on all managed nodes

vim playbook1.yml

---

- name: Create users on managed node

hosts: all

tasks:

- name: Create a user

user:

name: Anu

password: intelliqit

uid: 1357

home: /home/Anu

shell: /bin/bash

...

To check if the playbook is created syntactically correct or not

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: "IntelliQIT"

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

- name: Restart apache2

service:

name: apache2

state: restarted

- name: Check url response of server1

uri:

url: http://172.31.37.220

status\_code: 200

- name: Check url response of server2

uri:

url: http://172.31.36.172

status\_code: 200

...

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

Ansible playbook for configuring tomcat

vim playbook3.yml

---

- name: Configuring tomcat

hosts: all

tasks:

- name: Install tomcat9

apt:

name: tomcat9

state: present

- name: Copy tomcat-users.xml file

copy:

src: /home/ubuntu/tomcat-users.xml

dest: /etc/tomcat9

- name: Change tomcat port from 9090 to 8080

replace:

regexp: 9090

replace: 8080

path: /etc/tomcat9/server.xml

- name: Restart tomcat9

service:

name: tomcat9

state: restarted

- name: Pause for some time

pause:

minutes: 2

- name: Check url response of tomcat on server1

uri:

url: http://172.31.37.220:8080

status\_code: 200

- name: Check url response of tomcat on server2

uri:

url: <http://172.31.36.172:8080>

* Its not possible to learn each and every module and its arguments. Instead of learning them we can have following approach🡪

ansible -doc user

scroll down and you will find use cases to refer

* Suppose you have one folder in all working nodes, from which you want to delete some files having .xyz as a extension

ansible all -m shell -a ‘rm -rf /var/lib/\*.xyz’ -b

Day 5

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

Ansible playbook to copy file from one manged node to another managed node

---

- name: Fetch file from server1

hosts: 172.31.55.129

tasks:

- name: Fetch the file

fetch:

src: myfile

dest: /tmp

- name: Copy file to server2

hosts: 172.31.48.61

tasks:

- name: Copy file

copy:

src: /tmp/172.31.55.129/myfile

dest: /home/ubuntu

...

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

Variables

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

These are used for achieving greater reusability of the playbooks

They are categorized into 3 types

1 Global Scope Variables

2 PlayScope Variables

3 Host Scope variables

Global Scope variables are passed from the command prompt

using "--extra-vars" and they have the highest level of

priority

Host scope variables are again classified into 2 type

a) Variables to work on a group of hosts

b) Variables to work on a single host

Host scope variables have the second level of priority after global scope

variables

Playscope variables are defined within the playbooks and they have the least

level of priority

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

Ansible playbook for installing/uninstalling s/w applications

using global scope variables

vim playbook5.yml

---

- name: Install s/w applications using variables

hosts: all

tasks:

- name: Install/uninstall s/w app

apt:

name: "{{a}}"

state: "{{b}}"

update\_cache: "{{c}}"

...

The above playbook can be used with different sets of data

ansible-playbook playbook5.yml --extra-vars "a=git b=present c=yes" -b

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

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

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

Ansible playbook to create users and create files and dirs

---

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

hosts: all

tasks:

- name: Create users

user:

name: "{{a}}"

password: "{{b}}"

home: "{{c}}"

- name: Create files/dirs in users home dir

file:

name: "{{d}}"

state: "{{e}}"

...

To execute the playbook with different sets of data

ansible-playbook playbook7.yml --extra-vars "a=Anu b=intelliqit c=/home/Anu d=/home/Anu/dir1 e=directory" -b

ansible-playbook playbook7.yml --extra-vars "a=Ramesh b=intelliqit c=/home/Ramesh d=/home/Ramesh/file1 e=touch" -b

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

PlayScope varibales

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

These variables are defined within a play and they can

work only on that play

vim playbook8.yml

---

- name: Install s/w applications

hosts: all

vars:

- a: tomcat9

- b: present

- c: yes

tasks:

- name: Install s/w

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 bypass that behaviour and make it work on

some other sets of data by passing global scope variables using

--extra-vars

ansible-playbook playbook8.yml --extra-vars "a=apache2 b=absent" -b

you can pass any variable and also can skip any variable as per your need. Ansible will take skipped variable from playbook.

ansible-playbook playbook8.yml --extra-vars "a=apache2 " -b

Day 6

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

Grouping in inventory file

One machine can be present in multiple groups

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

sudo vim /etc/ansible/hosts

Adhoc commands 🡪

ansible webservers -a ‘free’

ansible appserver -a ‘free’

ansible dbserver -a ‘free’

ansible servers -a ‘free’

[webservers]

172.31.55.129

172.31.48.61

[appserver]

172.31.50.184

[dbserver]

172.31.59.180

172.31.50.184

[servers:children]

webserver

apserver

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

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 created 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 variables

vim playbook9.yml

---

- name: create user using host scope variables

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

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

Loops in ansible can be implemented 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

...

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

Day 7

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

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

...

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

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

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

Day 10

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

when conditions

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

This is used to execute a playbook based on a specific

condition

vim playbook16.yml

---

- name: Implementing when conditions

hosts: all

vars:

- a: 10

tasks:

- name: Create a file

file:

name: /home/ubuntu/file12

state: touch

when: a == 20

...

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

---

- name: Check if a folder called d1 is present if not create a file called d1

hosts: all

tasks:

- name: Check for d1 directory

stat:

path: /home/ubuntu/d1

register: a

- name: Display output of abouve module

debug:

var: a

- name: Create file d1 if dir d1 is not present

file:

name: /home/ubuntu/d1

state: touch

when: a.stat.exists == false

Day 8

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

---

- name: Change ownership of file based on a condition

hosts: all

tasks:

- name: Capture info about a file

stat:

path: /home/ubuntu/file12

register: result

- name: Display output of above module

debug:

var: result

- name: Give execute permissions

file:

name: /home/ubuntu/file12

mode: 0770

when: result.stat.executable == false

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

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

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

---

- name: Create a user and set a password if user is not already exisiting

hosts: all

tasks:

- name: Create a user

user:

name: Sai

notify: Set password

handlers:

- name: Set password

user:

name: Sai

password: intelliqit

...