**Ansible installation in Linux Master**

* You can consider **ansible ad-hoc commands as shell commands** and a **playbook as a shell script.**

**Steps to install Ansible in Master node:**

1. **Sudo su**
2. Create hostname **“hostname Ansible-Master”**
3. Change hostname permanently **“ vi /etc/hostname”**
4. To use hostname follow bellow commands

[root@ip-172-31-19-198 ec2-user]# **exit**

exit

[ec2-user@ip-172-31-19-198 ~]$ **sudo su**

[root@Ansible-Master ec2-user]#

1. Now create user **“useradd ansimaster**”
2. For to see howmany users are there use **“cat /etc/passwd”**
3. To create password for user use **“passwd ansimaster”**

[root@Ansible-Master ec2-user]# **passwd ansimaster**

**Changing password for user ansimaser.**

New password: **pavan@140697**

Retype new password: **pavan@140697**

passwd: **all authentication tokens updated successfully.**

1. I need to give ROOT permissions to USER

**Visudo**

Add user

root ALL=(ALL) ALL

ansimaster ALL=(ALL) NOPASSWD: ALL

1. Now provide Password Authentication to user

**vi /etc/ssh/sshd\_config**

# To disable tunneled clear text passwords, change to no here!

PasswordAuthentication yes

#PermitEmptyPasswords no

#PasswordAuthentication no

1. Now restart the sshd service

**service sshd restart/reload**

1. Now change as USER

**Sudo su – ansimaster**

[ec2-user@Ansible-Master ~]$ **sudo su - ansimaster**

[**ansi@**Ansible-Master ~]$

1. After changed to USER generate SSH keys

**ssh-keygen** and then click on 3 times on **enter**

**To Install Ansible we need Pre-requisites:**

1. Python and pip

Python already there and to install pip “**sudo yum install python-pip**”

1. Now install ANSIBLE

(sudo amazon-linux-extras install ansible2)

**RedHat/CentOS:** sudo yum install ansible

**Fedora:** sudo dnf install ansible

**Ubuntu:** sudo apt-get install ansible

**PIP:** sudo pip install ansible

Sudo yum install epel-release

Sudo yum install python-pip

sudo pip install ansible==2.4 or

sudo pip install --upgrade ansible

1. I am used **“sudo pip install ansible”**
2. Check version **“ansible –version”**

**Note:** if you used yum command by default it creates directory **“/etc/ansible”** but we used pip so I need to create MANUALLY.

1. **Sudo mkdir /etc/ansible**
2. **cd /etc/ansible**
3. I need **to create configuration file** in this directory **“sudo touch ansible.cfg”**

Go to this link <https://docs.ansible.com/ansible/latest/installation_guide/intro_configuration.html>

or

click on <https://github.com/ansible/ansible/blob/devel/examples/ansible.cfg>

and copy the content paste in **“sudo vi** **ansible.cfg”**

1. Create INVENTORY/HOST file **“sudo touch hosts”**

**Note:**

In host file we provide **IP address** and create **Groups.**

**Create Nodes/Slaves**

I created 2 instances. One is “Linux” and “Ubuntu”.

Create user in each instance.

1. **Sudo su**
2. Create hostname **“hostname Ansible-Manage-node01/02”**
3. Change hostname permanently **“ vi /etc/hostname”**
4. To use hostname follow bellow commands

[root@ip-172-31-19-198 ec2-user]# **exit**

exit

[ec2-user@ip-172-31-19-198 ~]$ **sudo su**

[root@Ansible-Master ec2-user]#

1. Now create user **“useradd ansimaster**”

For Ubuntu

**Useradd -m -d /home/ansimaster ansimaster**

1. For to see howmany users are their use **“cat /etc/passwd”**
2. To create password for user use **“passwd ansimaster”**

[root@Ansible-Master ec2-user]# **passwd** **ansimaster**

**password for user ansimaster ===> adapala303**

1. I need to give ROOT permissions to USER

**Visudo**

Add user

root ALL=(ALL) ALL

ansimaster ALL=(ALL) NOPASSWD: ALL

1. Now provide Password Authentication to user

**vi /etc/ssh/sshd\_config**

# To disable tunneled clear text passwords, change to no here!

PasswordAuthentication yes

#PermitEmptyPasswords no

#PasswordAuthentication no

1. Now restart the sshd service

**service sshd restart/reload or systemctl restart sshd**

1. Now change as USER

**Sudo su – ansimaster**

**TEST: master connecting to slave nodes or not**

* ssh-copy-id ansimaster

enter password of slave user “**adapala303”**

* **ssh** [**10.10.11.127**](mailto:ansi-manage01@10.10.11.127) **===> we will connect to slave node without any key.**

Note:

If are getting any error “check you are login as an ansible user or not. If not “ sudo su - ansible user”. Then try once again.

**Now do some tasks in Nodes/Slaves by using Ansible Master**

* Go to “ansible master” instance switch to user -> go to “ cd /etc/ansible” -> sudo vi hosts -> create groups and add “private IP of Instances”.
* From my master I used this command
* In hosts file we need to enter

10.10.11.127

* **Use some Ad-Hoc commands:**

1. ansible all -m ping
2. ansible --list-hosts all
3. ansible all -m command -a "uptime"
4. ansible all -m command -a "hostname"
5. [anadmin@ansible-master-node ansible]$ ansible all -m command -a "uptime" -o =>for single line O/P

Output: 10.10.11.127 | CHANGED | rc=0 | (stdout) 08:58:01 up 47 min, 2 users, load average: 0.03, 0.01, 0.00

1. To know about module ===> ansible-doc <modulename>
2. ansible –help
3. ansible all -m shell -a "ip addr" -b

**my note:** in ansible master and slaves the username is same then only we can connect to slaves.

ssh-copy-id <privateip>

password: slave password

**Question:** can we connect to ‘managed node’ without python in it?

**Ans:** yes, by using ‘raw’ module it is possible.

ansible -u root -i inventory <node server IP> --ask-pass -m raw -a ‘yum install python3 -y’

**Ad-Hoc Commands: ( these are idempotent ) ===> the commands used for testing**

1. ansible-doc <modulename>
2. ansible --help
3. ansible all --list-hosts
4. ansible --list-hosts ungrouped => ungrouped server details
5. ansible-inventory --graph ==> to know how many groups are presented
6. ansible all -m ping
7. ansible <groupname> -m ping
8. ansible all -m command -a “uptime”
9. ansible all -m command -a “hostname”
10. ansible all -m command -a “uptime” -o
11. **To create user**===>ansible all -m user -a “name=pavan” -b
12. ansible <groupname> -m user -a “name=pavan” -b
13. if you have custom inventory/host file then use
14. ansible -i <hostfile with full path> -m user -a “name=pavan” -b
15. **To copy the file**===>ansible all -m copy -a “src=/home/ec2-user/abc.sh dest=/var/tmp/abc.sh” -b
16. ansible all -m copy -a “src=/home/ec2-user/abc.sh dest=/var/tmp/abc.sh mode=0755” -b
17. ansible all -m file -a “src=/home/ec2-user/abc.sh mode=0755” -b
18. **To make file** ===>ansible all -m file -a “path=/var/tmp/text01.txt state=touch” -b
19. ansible all -m file -a “path=/var/tmp/text01.txt state=touch mode=0755” -b
20. **for delete file**===> ansible all -m file -a “path=/var/tmp/text01.txt state=absent” -b
21. **To make Directory**===> ansible all -m file -a “path=/var/tmp/test state=directory” -b
22. **To know memory details**===> ansible all -m shell -a “cat /proc/meminfo | head -5”
23. **To know free memory**===> ansible all -m shell -a “free -m”
24. **To know statics of file**===> ansible all -m stat -a “path=filepath”
25. **To know IP address===>** ansible all -m shell -a "ip addr" -b
26. **To install software’s===>** ansible all -m yum -a “name=git” -b
27. ansible <group-name> -m git -a "repo=https://foo.example.org/repo.git dest=/srv/myapp version=HEAD"
28. ansible <group-name> -m **apt -a 'update\_cache=yes' -b**
29. ansible <group-name> -m **apt -a 'upgrade=dist update\_cache=yes' -b**
30. **To check CPU usage**===>ansible all -m shell -a 'mpstat -P ALL' --become
31. **To check open ports===>**ansible all -m shell -a 'netstat -plntu' --become

**Ad-Hoc commands to install packages in slaves by using master**

* To install “httpd/apache2”

1. ansible <group-name> -m **yum -a “name=httpd state=present” -b**
2. ansible <group-name> -m **apt -a “name=apache2 state=present” -b**
3. ansible <group-name> -m service -a "name=httpd enabled=yes state=started" -b
4. ansible <group-name> -m **service -a "name=httpd state=started" -b**
5. ansible <group-name> -m service -a "name=httpd enabled=yes state=started" -b

**note:**

state= present, started, latest, installed, absent, stopped, restarted

modules= **yum, apt, service**

* To install “git”

1. ansible <group-name> -m **apt -a "name=git state=latest" -b**
2. ansible <group-name> -m **yum -a "name=git state=latest" -b**
3. ansible <group-name> -m **apt -a "name=git state=present" -b**
4. ansible <group-name> -m **yum -a "name=git state=present" -b**

* To install “Nginx”

1. ansible <group-name> -m apt -a 'name=nginx state=latest' -b
2. ansible <group-name> -m yum -a 'name=nginx state=latest' -b
3. ansible <group-name> -m apt -a 'name=nginx state=**absent** purge=yes' -b
4. ansible <group-name> -m apt -a 'name=nginx state=absent purge=yes **autoremove=yes**' -b
5. ansible <group-name> -m service -a 'name=nginx **state=started** enabled=yes' -b
6. ansible <group-name> -m service -a 'name=nginx **state=restarted'** --become
7. ansible <group-name> -m service -a 'name=nginx **state=stopped'** –become

* To install “Mysql”

1. ansible db\_servers -m yum -a 'name=mysql-server **state=present'** -b
2. ansible web\_servers -m apt -a 'name=mysql-server **state=present'** -b

**note:**

to check status of mysql

**for ubuntu:** service mysql status

**for linux:** service mysqld status

1. for linux we need to start mysql no need for ubuntu

ansible db\_servers -m service -a 'name=mysqld enabled=yes **state=started'** -b

1. ansible web\_servers -m service -a 'name=mysql **state=stopped'** -b
2. ansible web\_servers -m service -a 'name=mysql **state=restarted'** -b

**Playbook commands:**

**Note:**

* ansible-playbook <playbook file name with path> --syntax-check
* ansible-playbook <playbook file name with path > --check
* time ansible-playbook <playbook file name with path > --check
* ansible-playbook <playbook file name with path > --list-hosts --list-tasks
* ansible-playbook <playbook name> -v (or) -vv (or) -vvv (or) -vvvv ===> displays the task results
* ansible-playbook -C <playbook name> ===> for dry run

1. sudo su <ansible-user>
2. cd /etc/ansible
3. sudo mkdir playbooks
4. cd /playbooks
5. sudo touch sample.yml
6. sudo vi sample.yml
7. ansible all -m setup ===> to know about ansible server
8. ansible all -m setup |grep ansible\_os\_family
9. ansible-playbook package.yml
10. ansible-playbook package.yml **--list-tags**
11. ansible-playbook package.yml **--tags 'git install'**

**Ansible:**

Pre-requisite:

1. Python3

Files and directories in ansible:

1. Hosts
2. Ansile.cfg
3. Playbooks
4. Roles

**Note:**

Linux machine

Required:

Python 3.5 or later

Linux Machine

As

Central node

Windows machine

Required:

Powershell v 3.0

.net v 4

**Host/inventory file types:**

1. **Static /default inventory file:** the path is /etc/ansible/hosts
2. **Dynamic inventory file:** ex: ansible -i inventory file path --list-hosts

**best way is:** create host file in project directory. This means each project having individual inventory file.

* In Host or Inventory can create GROUPS and NESTED GROUPS

[group A]

ip[1:10]

[db]

ip3

ip4

[web]

ip5

[india:children]

db

web

[usa:children]

group A

* How to run NESTED GROUPS in Ad-Hoc Command

**ansible india --list-hosts**

**ansible.cfg:**

1. [defaults]
2. [privilege\_escalation]

[defaults]

# some basic default values...

inventory = /etc/ansible/good/hosts

remote\_user = ansiblem

host\_key\_checking = false

deprecation\_warning = false

[privilege\_escalation]

become=True

become\_method=sudo

become\_user=root

become\_ask\_pass=False

**Playbooks:**

Playbooks are used to run multiple plays and also tasks.

**sample playbook:**

---

- name: this is sample playbook to install some packages in slave nodes

hosts: all

become: true

tasks:

- name: install httpd

yum:

name: httpd

state: installed

when: ansible\_os\_family == "RedHat"

- name: start httpd

service:

name: httpd

state: restarted

when: ansible\_os\_family == "RedHat"

- name: insatll apache2

apt:

name: apache2

state: present

when: ansible\_os\_family == "Debian"

- name: start apache2

service:

name: apache2

state: started

when: ansible\_os\_family == "Debian"

- name: install git

yum:

name: git

state: present

when: ansible\_os\_family == "RedHat"

**Playbook with tags:**

---

- name: this is sample playbook to install some packages in slave nodes

hosts: all

become: true

tasks:

- name: install httpd

yum:

name: httpd

state: installed

when: ansible\_os\_family == "RedHat"

tags: install httpd

- name: start httpd

service:

name: httpd

state: restarted

when: ansible\_os\_family == "RedHat"

tags: started httpd

- name: insatll apache2

apt:

name: apache2

state: present

when: ansible\_os\_family == "Debian"

tags: install apache2

- name: start apache2

service:

name: apache2

state: started

when: ansible\_os\_family == "Debian"

tags: started apache2

- name: install git

yum:

name: git

state: present

when: ansible\_os\_family == "RedHat"

tags: git install

…

**note:**

1. ansible-playbook package.yml **--list-tags**
2. ansible-playbook package.yml **--tags 'git install'**

**Variables:**

---

- name: creating user

hosts: all

vars:

user: lisa

tasks:

- name: create a user {{ user }} # if it is occurring some ware in the sentence use {{varname}}

user:

name: ‘’{{ user }}” # if it is occurring first must use “{{ varname }}”

**note**: we can also use variable file. ( vars\_file: )**:**

**variable precedence:**

1. host scope – inclusion file
2. play scope – in playbook
3. global scope - variable define in inventory file, **command line**

**ex:**

[server]

ansiblenode01 web\_server=httpd ===> this belong to only “ansiblenode01” server **(assign variable to host)**

or

[server: vars]

Web\_server=httpd ===> this belong to [server] group **(assign variable to hostgroup)**

Note:

If we are using variable files ===> project\_dir/host\_vars/varfilename

For groups ===> project\_dir/group\_vars/varfilename

**Multi value variables:**

We can write in two ways: 1. arrays or list 2. Dictionaries

**List:**

Variable file name: **userslist**

users:

  - username: Pavan

    homedir: /home/Pavan

    shell: /bin/bash

  - username: Kumar

    homedir: /home/Kumar

    shell: /bin/bash

  - username: Adapala

    homedir: /home/Adapala

    shell: /bin/sh

**How to use this variable file in “playbook”:**

Playbook name: simple.yml

---

- name: simple playbook for how to use multi value variable

  hosts: all

  vars\_file:

    - userslist

  tasks:

    - name: print array values

      debug:

        msg: "user {{ item.username }} has homedirectory {{ item.homedir }} and shell {{ item.shell }}"

      loop: "{{ users }}"

    - name: print the second array value

      debug:

        msg: "the second item is {{ users[1] }}"

---

- name: "playbook for practice"

hosts: all

become: true

#gather\_facts: false

vars:

userlist:

- kumar

- adapala

tasks:

- name: "create {{ name }} in redhat server"

user:

name: "{{ item }}"

loop: "{{ userlist }}"

…

**dictionary: [ loop is can’t possible ]**

variable file name: usersindict

users:

  Pavan:

    username: Pavan

    homedir: /home/Pavan

    shell: /bin/bash

  Kumar:

    username: Kumar

    homedir: /home/Kumar

    shell: /bin/bash

  Adapala:

    username: Adapala

    homedir: /home/Adapala

    shell: /bin/sh

Playbook :

---

- name: simple playbook for how to use multi value variable

  hosts: all

  vars\_file:

    - usersindict

  tasks:

    - name: print dictionary values

      debug:

        msg: "user {{ users.Pavan.username }} has homedirectory {{ users.Pavan.homedir }} and shell {{ users.Pavan.shell }}"

        or

        msg: "user {{ users['Pavan']['username'] }} has homedirectory {{ users['Pavan']['homedir'] }} and shell {{ users['Pavan']['shell'] }}"

**Ansible vault:**

Ansible vault is used to store “secrets” like credentials, passwords.

Note: make 2 separate files for “encrypted” and “non-encrypted”

**Commands:**

ansible-vault create playbook.yml

ansible-vault view playbook.yml

ansible-vault edit playbook.yml

ansible-vault encrypt filename

ansible-vault decrypt filename

ansible-vault rekey filename ===> to change password

**ex: playbook name “ansiblevault.yml”**

---

- name: simple playbook for how to use multi value variable

  hosts: all

  vars\_file:

    - secret.yml

  tasks:

    - name: print dictionary values

      user:

        name: "{{ username }}"

        password: "{{ pwhash }}"

**note:**

secret.yml is created by using “ansible-vault” ===> **ansible-vault create secret.yml ===>** enter password (set new one) ===> after that it will open “vi” editor ===> enter variable and values.

username: Pavan

pwhash: password

**Run the Playbook:**

**ansible-playbook --ask-vault-pass ansiblevault.yml ===> enter Vault password**

when I run the above command “I need to enter vault password so I will keep my password in a file”

**filename:** vault.pass ===> chmod 400 vault.pass

**password into a file:** echo “pavankumar” > vault.pass

**ansible-playbook --vault-password-vault.pass ansiblevault.yml**

**Loops:** this is new option before (ansible version below 2.5 ) used “with\_” (with\_items).

**Example01**

---

- name: postfix and dovecat starting

service:

name: “{{ item }}”

state: started

loop:

- postfix

- dovecat

**Example02**

---

vars:

mail\_services:

- postfix

- dovecat

tasks:

- name: postfix and dovecat starting

service:

name: “{{item}}”

state: started

loop: “{{ mail\_services }}” ===>passed a variable

**Best Example:**

---

- name: create users using loop

hosts: all

tasks:

- name: create users

user:

name: “{{ item.name }}”

state: present

groups: “{{ item.groups }}”

loop:

- name: anna

groups: wheel

- name: linda

groups: users

- name: bob

groups: users

**How to use “register variable” with “loop”:**

* “register” is a key-word.
* It stores the command output in a variable.
* It also used with loops. Ex: **loop: “{{ echovar.stdout\_lines }}”**

**Example:**

---

- name: demo how register works

hosts: all

tasks:

- name: show loop register

shell: “echo {{ item }}”

loop:

- “one”

- “two”

register: echovar

- name: show register results

debug:

var: echovar

**Example:** ( this example shows how to use register variable output as condition ) **=> when & register**

---

- name: demo how register works

hosts: all

tasks:

- shell: cat /etc/passwd

register: echovar

- debug:

msg: echo “passwd contains user lisa”

when: echovar.stdout.find(‘lisa’) != -1

**Example:** moving users directory to backup directory

---

- name: registered variable usage as aloop list

  hosts: all

  tasks:

    - name: create target directory

      file:

        path: /mnt/bkspool

        state: directory

    - name: retrieve the list of home directories

      command: ls /home

      register: home\_dir

    - name: add home dirs to the backup spooler

      file:

        path: /mnt/bkspool/{{ item }}

        src: /home/{{ item }}

        state: link

      loop: "{{ home\_dir.stdout\_lines }}"

...

Note: (when statements) are used for conditions.

Example:

---

- name: when demo

  hosts: all

  vars:

    supported\_distros:

      - RedHat

      - CentOS

      - Fedora

  tasks:

    - name: install nginx pacakge in {{ supported\_distros }}

      yum:

        name: nginx

        state: present

      when: ansible\_distribution in supported\_distros

...

**How to use Multiple conditions with when:**

Example:

---

- name: when demo

  hosts: all

  tasks:

    - name: install nginx package

      yum:

        name: nginx

        state: present

      when:

        - ansible\_distribution == "RedHat"

        - ansible\_memfree\_mb > 512

...

Example:

---

- name: when demo

  hosts: all

  tasks:

    - name: install nginx package

      yum:

        name: nginx

        state: present

      when:

        ( ansible\_distribution == "RedHat" and ansible\_memfree\_mb > 512 )

        or

        ( ansible\_distribution == "CentOS" and ansible\_memfree\_mb > 600 )

...

**Handlers:**

* Handlers are used when “ dependent tasks are taking place in playbook ”.
* Handlers are executed after all tasks are successfully executed otherwise handlers won’t execute.
* When the playbook having **“force handlers : true”** than it executed when particular task successfully executed. (this means if other tasks are not executed but handlers will run)
* Handlers are run only “single time only”.

**Note:**

We can use multiple handlers for single task but it happens in rare scenarios.

**Playbook with “lineinfile”, “handlers”, “tags”, “var”:**

---

- name: this is sample playbook to install some packages in slave nodes

hosts: all

become: true

vars:

port: 800

tasks:

- name: install httpd

yum:

name: httpd

state: present

when: ansible\_os\_family == "RedHat"

- name: start httpd

service:

name: httpd

state: started

when: ansible\_os\_family == "RedHat"

- name: changes in configuration port

lineinfile:

path: /etc/httpd/conf/httpd.conf

regexp: '^Listen'

insertafter: '^#Listen'

line: Listen {{port}}

when: ansible\_os\_family == "RedHat"

notify: restart httpd

- name: insatll apache2

apt:

name: apache2

state: present

when: ansible\_os\_family == "Debian"

- name: start apache2

service:

name: apache2

state: started

when: ansible\_os\_family == "Debian"

- name: changes in configuration port

lineinfile:

path: /etc/httpd/conf/httpd.conf

regexp: '^Listen'

insertafter: '^#Listen'

line: Listen {{port}}

when: ansible\_os\_family == "RedHat"

notify: restart httpd

- name: insatll apache2

apt:

name: apache2

state: present

when: ansible\_os\_family == "Debian"

- name: start apache2

service:

name: apache2

state: started

when: ansible\_os\_family == "Debian"

- name: changes in configuration port

lineinfile:

path: /etc/apache2/ports.conf

regexp: '^Listen'

insertafter: "^# /etc/apache2/sites-enabled/000-default.conf"

line: Listen {{port}}

when: ansible\_os\_family == "Debian"

tags: change in port

notify: restart apache2

- name: install git

yum:

name: git

state: present

when: ansible\_os\_family == "RedHat"

handlers:

- name: restart httpd

service:

name: httpd

state: restarted

- name: restart apache2

service:

name: apache2

state: restarted

…

**Block:**

In blocks we can run multiple tasks and we can’t use loops in blocks.

Block can be called as **super set of tasks.**

Example:

---

- name: this is sample playbook to install some packages in slave nodes

hosts: all

become: true

tasks:

- name: setting up http

block:

- name: install httpd

yum:

name: httpd

state: present

- name: start httpd

service:

name: httpd

state: started

when: ansible\_distribution == "RedHat"

Example:

---

- name: using blocks

  hosts: all

  tasks:

- name: intended to be successful

block:

    - name: remove a file

      shell:

       cmd: /usr/bin/rm /var/www/html/index.html

rescue:

    - name: create a file

      shell:

       cmd: /usr/bin/touch /tmp/rescuefile

always:

    - name: always write a message to logs

      shell:

       cmd: /usr/bin/logger hello

**Note:** when the block task failed then “rescue task will run if block task success than rescue won’t run”. But always task will run every time.

Note:

When we are using failure module then we must use “ignore\_errors: yes “ in tasks.

---

- name: demonstrating failed\_when

  hosts: all

  tasks:

   - name: run a script

     command: echo “helloworld”

     ignore\_errors: yes

     register: command\_result

     failed\_when: “ ‘world’ in command\_result.stdout”

**Playbook for Nginx:**

---

- name: This is sample playbook for install nginx

  hosts: all

  become: true

  tasks:

  # removing httpd

   - name: removed httpd

     yum:

      name: httpd

      state: absent

     when: ansible\_os\_family == "RedHat"

  # removing httpd

   - name: install nginx

     apt:

      name: apache2

      state: absent

     when: ansible\_os\_family == "Debian"

   # installing Nginx in redhat

   - name: install nginx

     yum:

      name: nginx

      state: present

     when: ansible\_os\_family == "RedHat"

   # nginx starting stage in RedHat

   - name: start nginx

     service:

      name: nginx

      state: started

     when: ansible\_os\_family == "RedHat"

   # install nginx in Debian

   - name: installation nginx

     apt:

      name: nginx

      state: present

     when: ansible\_os\_family == "Debian"

   # nginx starting in Debian

   - name: start nginx

     service:

      name: nginx

      state: started

     when: ansible\_os\_family == "Debian"

**Modules used to change the file/file content**

1. lineinfile ===> it used for single line modification
2. blockinfile ===> more lines
3. fetch ===> it fetch the file from remote machine to managed node
4. copy
5. file

Example:

---

- name: using file module

  hosts: all

  tasks:

    - name: create a file

      file:

        path: /tmp/removeme

        owner: ansible

        mode: 0640

        state: touch

        setype: public\_content\_rw\_t

...

# playbook 2

---

- name: some file modules

  hosts: all

  tasks:

    - name: using copy module

      copy:

        src: /etc/hosts

        dest: /tmp/

    - name: add some lines to /tmp/hosts

      blockinfile:

        path: /tmp/hosts

        block: |

          192.168.4.110 host1.example.com

          192.168.4.120 host2.example.com

        state: present

    - name: verify file checksum

      stat:

        path: /tmp/hosts

        checksum\_algorithm: md5

      register: result

    - debug:

        msg: "The checksum of /tmp/hosts is {{result.stat.checksum }}"

    - name: fetch a file

      fetch:

        src: /tmp/hosts

        dest: /tmp/

...

**Custom files with jinja2:**

* by using “variables, facts” we can make changes in target file.

Elements in jinja2:

1. Data
2. Variables
3. Expressions
4. **Control structures**

**Note:**

Create a directory called “templates” in this create template files.

**Vfstpd.j2 ===> template file name**

anonymous\_enable={{ anonymous\_enable }}

local\_enable={{ local\_enable }}

write\_enable={{ write\_enable }}

anon\_upload\_enable={{ anon\_upload\_enable }}

dirmessage\_enable=YES

xferlog\_enable=YES

connect\_from\_port\_20=YES

pam\_servicce\_name=vsftpd

userlist\_enable=YES

#my IP address={{ ansible\_facts[‘default\_ipv4’][‘address’] }}

Playbook:

---

- name: configure VSFTPD using a template

  hosts: all

  vars:

    anonymous\_enable: yes

    local\_enable: yes

    write\_enable: yes

    anon\_upload\_enable: yes

  tasks:

    - name: install vsftpd

      yum:

        name: vsftpd

    - name: use template to copy FTP config

      template:

        src: templates/vsftpd.j2

        dest: /etc/vsftpd/vsftpd.conf

...

Example02:

hosts.j2

{% for hosts in groups['all'] %}

{{hostvars[host]['ansible\_facts']['default\_ipv4']['address']}} {{hostvars[host]['ansible\_facts']['fqdn']}} {{hostvars[host]['ansible\_facts']['hostname']}}

{% endfor %}

Playbook:

---

- name: update /etc/hosts file dynamically

  hosts: all

  tasks:

    - name: update /etc/hosts

      template:

        src: templates/hosts.j2

        dest: /etc/hosts

**ROLES:**

1. cd /etc/ansible/roles (or) cd /etc/ansible
2. mkdir roles
3. cd roles
4. mkdir apache-git ===> role name (or) sudo ansible-galaxy init apache-git
5. cd apache-git
6. mkdir files handlers tasks vars meta templates
7. in each directory we need to create “sudo touch main.yml”

like below: [anadmin@ansible-master roles]$ tree

└── apache-git

├── defaults

│   └── main.yml

├── handlers

│   └── main.yml

├── tasks

│   └── main.yml

└── vars

└── main.yml

1. cd playbooks
2. sudo vi package.yml

---

- name: this is sample playbook to install some packages in slave nodes

hosts: all

become: true

roles:

- apache-git

1. ansible-playbook package.yml --syntax-check
2. ansible-playbook package.yml –check
3. If all ok “ansible-playbook package.yml”

**Ansible also used as IAC (infrastructure as code):**

* By using we can create AWS resources like EC2, VPC, S3 etc…
* Check below link

https://docs.ansible.com/ansible/latest/collections/amazon/aws/ec2\_module.html

* Use this command “ansible-galaxy collection install amazon.aws”

**if any permission error in httpd start**

**like** Permission denied: AH00072: make\_sock: could not bind to address [::]:8000

**fix by running where httpd running:**

**Disable SELinux**

Disable SELinux temporarily

sudo setenforce 0

Restart httpd service

service httpd restart

Disable SELinux persistently (after reboot)

vi /etc/selinux/config

Add line and save

SELINUX=disabled

1. **Variables [ list, dictionaries ]**
2. **Loops**
3. Combination of **variable and loops**