Setting up ansible:

On the ansible master/controller vm running RHEL/Centos run ->

subscription-manager repos --enable rhel-\*-optional-rpms \

--enable rhel-\*-extras-rpms \

--enable rhel-ha-for-rhel-\*-server-rpms

yum install https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm

Then run:

yum install ansible

Thats it .

Ansible Inventory:

Ansible is agentless . It connects using ssh for unix target and winrm for windows clients.

The targets are defined in an inventory file.

The default location of the file is

/etc/ansible/hosts

For each project we can have separate inventory file in the working directory of the project .

To setup inventory:

First setup the targets in /etc/hosts of the controller

[root@dsb-dev-web-vm-0 project]# cat /etc/hosts

127.0.0.1 localhost ansiblecontroller

::1 localhost ansiblecontroller

10.0.1.4 ansible-target1

10.0.1.8 ansible-target2

Then login to the targets via ssh using the desired user which will be used for ansilble connection . Default is root

Then setup the inventory.txt(or any name) in the local working directory where you are going to have your playbook as:

web\_vm1 ansible\_host=ansible-target1 ansible\_connection=ssh ansible\_user=root ansible\_ssh\_pass=root@123

web\_vm2 ansible\_host=ansible-target2 ansible\_connection=ssh ansible\_user=root ansible\_ssh\_pass=root@123

[webvms]

web\_vm1

web\_vm2

* webvm1/2 are the aliases for the actual targets which have been ansible-target1/2
* ansible\_host -> defines the host to which the alias is mapped (this host must have an entry in the /etc/hosts file of the controller )
* ansible\_connection -> defines the protocol of logging in . In case of linux ssh , windows winrm
* ansible\_user -> Defines the user that is used to make the connection . default root for unix . administrator for windows .
* ansible\_ssh\_pass -> defines password for connection i.e password of root on target machine .
* In case of windows it will be ansible\_password
* ansible\_port -> used when ssh or windows remote connection is not using the default ports 22 or 5986

Test the connection by :

ansible web\_vm2 -m ping -i inventory.txt

ansible web\_vm1 -m ping -i inventory.txt

It should return a successful ping result like:

web\_vm2 | SUCCESS => {

"ansible\_facts": {

"discovered\_interpreter\_python": "/usr/bin/python"

},

"changed": false,

"ping": "pong"

}

You can also run it for the group

ansible webvms -m ping -i inventory.txt

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

Playbooks:

Playbooks are a set of instructions that ansible runs on the desired host or group of hosts to get to the desired state . All playbooks are written in yaml . A play is a set of activities to be run on a group of hosts/host . A task is a single activity run on a host/group of hosts

A play book has the following structure:

-

name: <name of playbook>

hosts: < on which hosts the plays will run /can be a group or a single hosts or all hosts in the inventory file>

become: true < if you are using any user other than root as ansible\_user and it has sudo access , then this line prompts sudo to root >

tasks:

* name : name of task

<module name>: <attribute>

* name: <name of second task>

<module name>: <attribute>

-

name: <name of second playbook>

hosts: < on which hosts the plays will run /can be a group or a single hosts or all hosts in the inventory file>

tasks:

* name : name of task

<module name>: <attribute>

* name: <name of second task>

<module name>:

name: < what to do with the module>

state: < what is the desired state of the attribute that is being worked upon by module>

Tasks is a list . So the hierarchy of task declaration within a play defines which tasks happen first and which task happens next .

To check syntax of a playbook

ansible-playbook connection\_playbook.yaml -i inventory.txt --syntax-check

To run a playbook:

Ansible-playbook <playbook\_name> -i <inventory-file-name> # if not specified then default inventory list is querried

There is a separate way of running ansible command:

Ansible < target host/group as mentioned in inventory> -m <module\_name> -i <inventory-file-name>

* ansible web\_vm2 -m ping -i inventory.txt

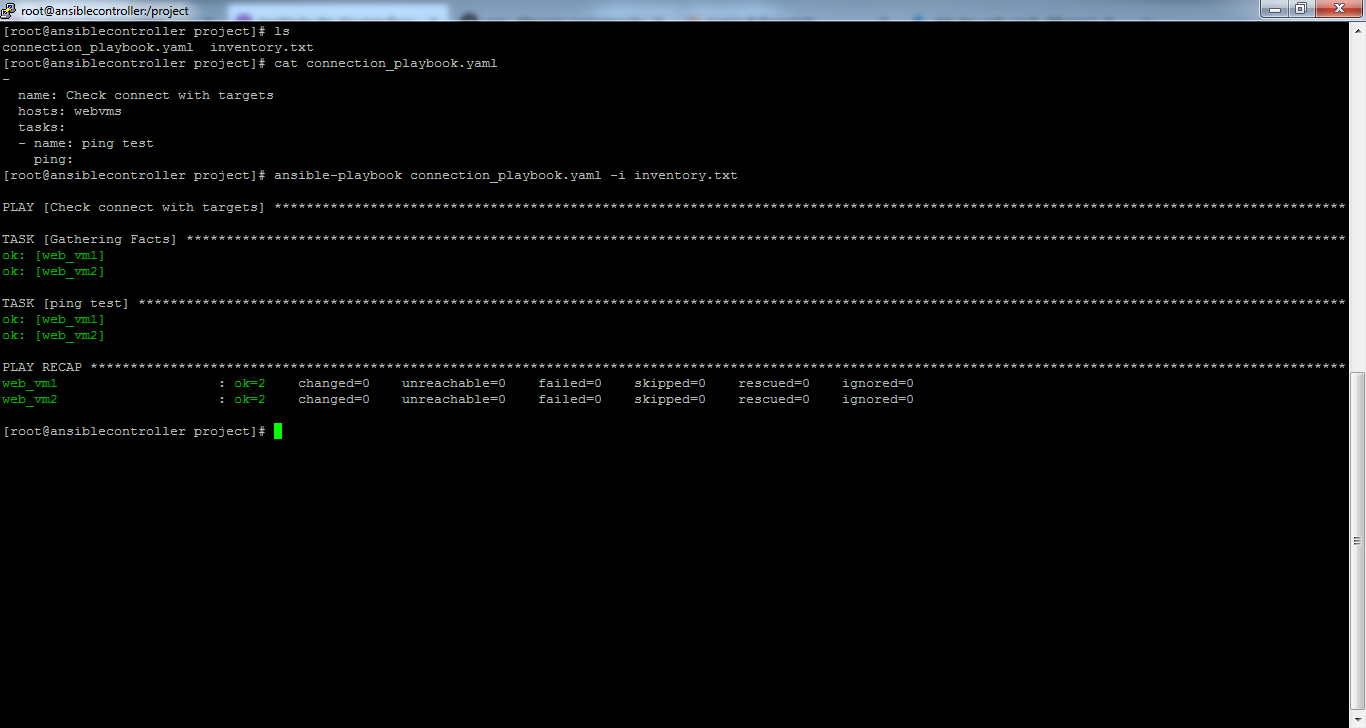
All commands run in a ansible are ansible modules . All modules are available in

<https://docs.ansible.com/ansible/2.3/yum_module.html>

You can also run a command or a script. For command in tasks mention the command after the command: key word in tasks , in case of scripts mention the script name in target after the script: key word .

The different actions run by tasks are ansible modules.

For example see the below setup:



The following is the contents of the playbook:

-

name: Check connect with targets

hosts: webvms

tasks:

- name: ping test

ping:

Now updated the playbook with two more tasks . Installing httpd and starting the service in the target servers.

-

name: Install webservers

hosts: webvms

tasks:

- name: ping test

ping:

- name: install http

yum:

name: httpd

state: latest

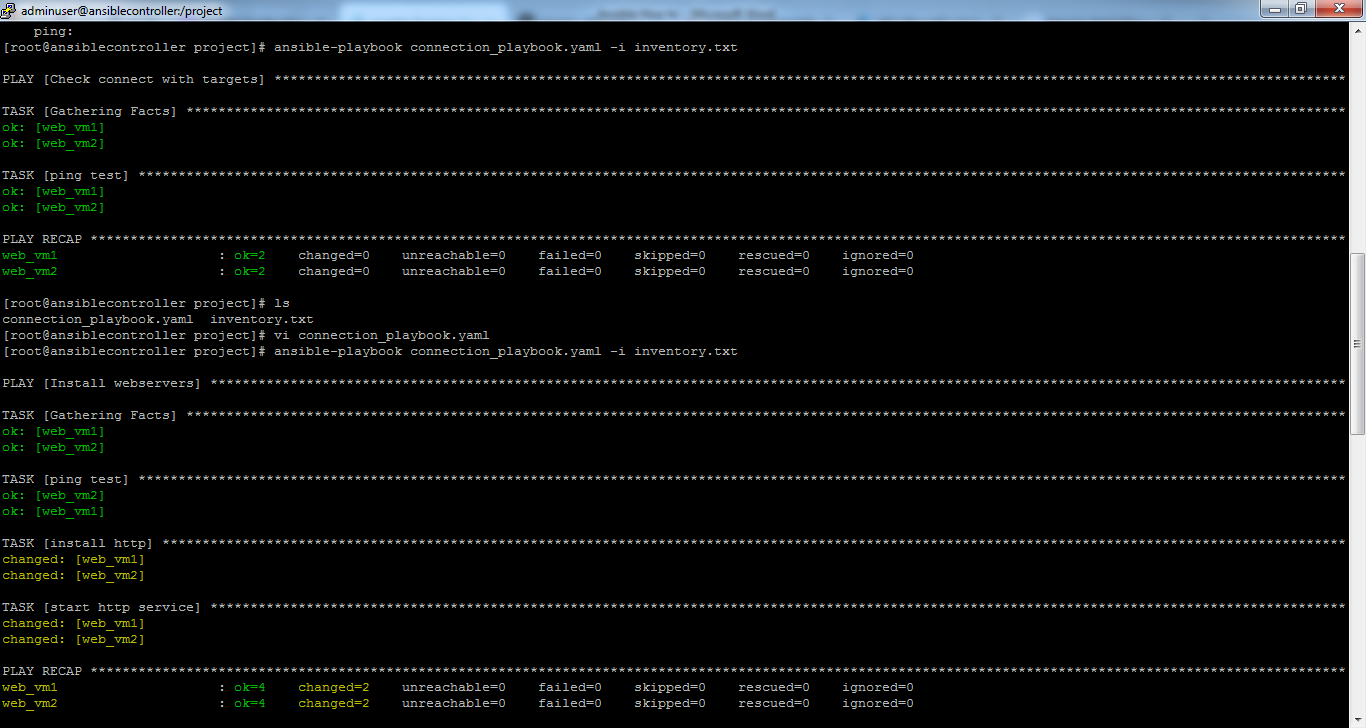
- name: start http service

service:

name: httpd

state: started

Output



Few modules for use:

lineinfile -> This is used to append a line in a file:

lineinfile:

path: <file path>

line: ‘<line to add>’

copy -> Copying a file from source to destination

copy:

source: <source file path>

destination: <destination file path>

script -> Executes a script on a remote machine after transferring it:

script: <local script path> -arg1 –arg2

User -> To create a user:

- name: 'Create a web user'

user:

name: web\_user

uid: 1040

group: developers

There are many modules . We go ahead with them as we code .

Input variables:

Input variables can be declared within the playbook itself or in a separate file .

Inside a playbook it is defined as:

-

name: <name of playbook>

hosts: <reqd hosts>

vars:

var: value

It is referred to by {{ var }}

If at the beginning of a line it is referred to by “{{ var }}”

Our playbook can be modified as follows:

[root@ansiblecontroller project]# cat connection\_playbook.yaml

-

name: Install webservers

hosts: webvms

vars:

pkg: httpd

tasks:

- name: ping test

ping:

- name: install http

yum:

name: "{{ pkg }}"

state: latest

- name: start http service

service:

name: httpd

state: started

This can also be done by setting the variables in the inventory file beside the relevant hosts.

Loops in Ansible:

Loops in ansible uses either the loop function or the with\_\* (i.e with\_items, with\_files etc ) function .

A typical loop with a variable containing a list of values looks like below:

name: 'Install required packages'

hosts: localhost

vars:

packages:

- httpd

- binutils

- glibc

- ksh

- libaio

- libXext

- gcc

- make

- sysstat

- unixODBC

- mongodb

- nodejs

- grunt

tasks:

-

yum:

name: '{{ item }}'

state: present

with\_items: '{{ packages }}'

item is the iterator . so {{ item }} accesses each of the values . we have used with\_items function here . It uses the variable packages for the iteration . The same can be done by loop: {{ <var\_name>}}

If the variable is a map(dictionary) instead of a list , then the iteration works as below:

name: 'Install required packages'

hosts: webvms

vars:

Users:

- name: abind

Uid: 007

-name: archie

Uid: 008

- name: 'Create a web user'

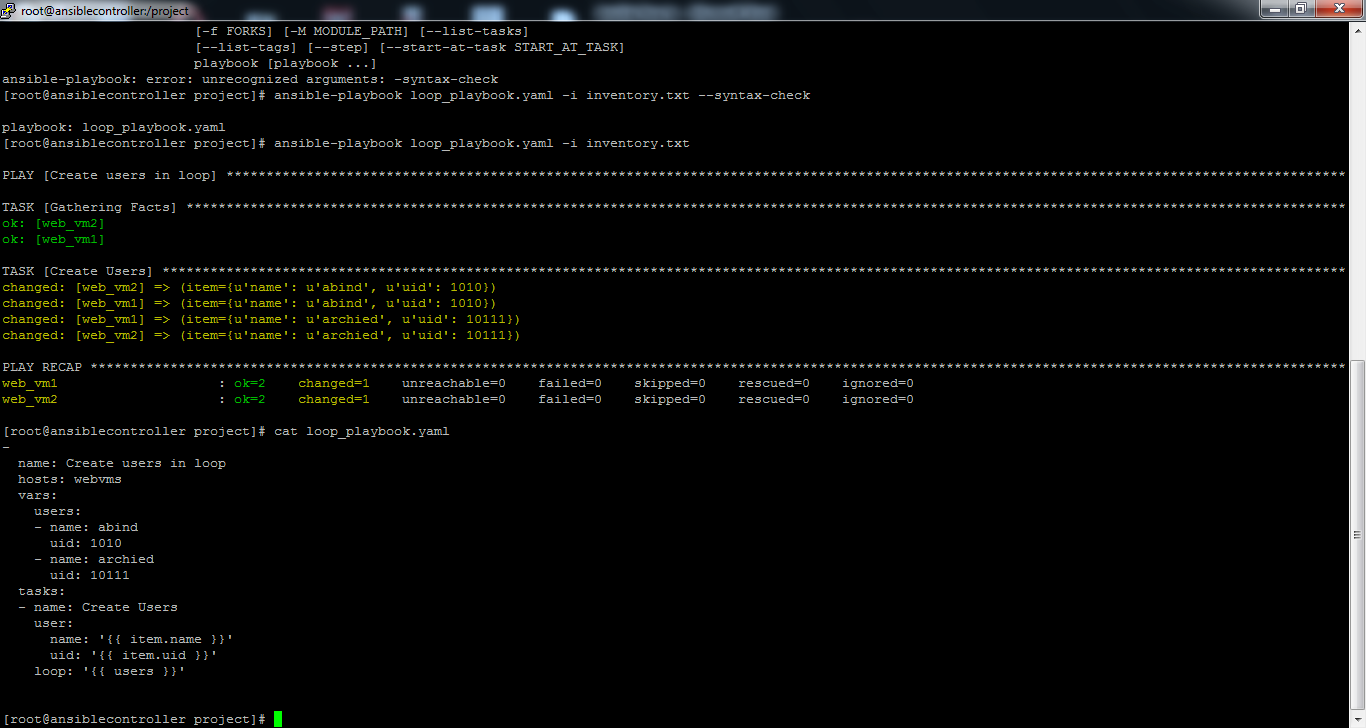
user:

name: ‘{{ item.name }}’’

uid: ‘{{ item.uid }}’

group: developers

loop: ‘{{ users}}’



Conditions in ansible:

A task can be done or not done based on a condition “when” .

The below example uses two loops . In one of the loops a user is created optionally if the required field is set to true:

-

name: Create users in loop

hosts: webvms

vars:

users:

- name: abind

uid: 1010

linux\_group: sysadmin

required: True

- name: archied

uid: 10111

linux\_group: developer

required: False

user\_group:

- name: sysadmin

gid: 555

- name: developers

gid: 666

tasks:

- name: Create Groups

group:

name: '{{ item.name }}'

gid: '{{ item.gid }}'

state: present

loop: '{{ user\_group }}'

- name: Create Users

user:

name: '{{ item.name }}'

uid: '{{ item.uid }}'

group: '{{ item.linux\_group }}'

when: item.required == True

loop: '{{ users }}'

Here the user is created only if it is in developer group

-

name: Create users in loop

hosts: webvms

vars:

users:

- name: abind

uid: 1010

linux\_group: sysadmin

- name: archied

uid: 10111

linux\_group: developers

user\_group:

- name: sysadmin

gid: 555

- name: developers

gid: 666

tasks:

- name: Create Groups

group:

name: '{{ item.name }}'

gid: '{{ item.gid }}'

state: present

loop: '{{ user\_group }}'

- name: Create Users

user:

name: '{{ item.name }}'

uid: '{{ item.uid }}'

group: '{{ item.linux\_group }}'

when: item.linux\_group == "developers"

loop: '{{ users }}'

=================================================================================================================A complete project:

-

name: Configure mariadb on database servers

hosts: dbvms

vars:

packages:

- MySQL-python

- mariadb-server

tasks:

- name: Install mariadb

yum:

name: "{{ item }}"

state: latest

loop: '{{ packages }}'

- name: Start mariadb service

service:

name: mariadb

state: started

- name: Allow port 3306 on server firewall

firewalld:

port: 3306/tcp

permanent: true

state: enabled

- name: Create a new database with name ecomdb

mysql\_db:

name: ecomdb

state: present

- name: Create a new user and grant priviledge

mysql\_user:

name: ecomuser

password: ecompassword

priv: '\*.\*:ALL,GRANT'

state: present

- name: Create the db-load-script.sql

#command: touch '{{ item }}' ; chmod +x '{{ item }}'

shell: |

touch '{{ item }}'

chmod +x '{{ item }}'

loop:

- /db-load-script.sql

- /script.sh

- name: Populate the script with data

blockinfile:

path: /db-load-script.sql

block: |

USE ecomdb;

CREATE TABLE products (id mediumint(8) unsigned NOT NULL auto\_increment,Name varchar(255) default NULL,Price varchar(255) default NULL, ImageUrl varchar(255) de

fault NULL,PRIMARY KEY (id)) AUTO\_INCREMENT=1;

INSERT INTO products (Name,Price,ImageUrl) VALUES ("Laptop","100","c-1.png"),("Drone","200","c-2.png"),("VR","300","c-3.png"),("Tablet","50","c-5.png"),("Watch"

,"90","c-6.png"),("Phone Covers","20","c-7.png"),("Phone","80","c-8.png"),("Laptop","150","c-4.png");

- name: Run script.sh

script: /project/new\_project/mysqlscript.sh

-

name: Deploy and Configure Web

hosts: webvms

vars:

web\_packages:

- httpd

- php

- php-mysql

- git

tasks:

- name: Install the required Web packages

yum:

name: "{{ item }}"

state: latest

loop: '{{ web\_packages }}'

- name: Replace a line in httpd.conf

lineinfile:

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

regexp: '^DirectoryIndex index.html'

line: 'DirectoryIndex index.php'

backrefs: yes

- name: STart and enable httpd service

service:

name: httpd

state: started

enabled: yes

- name: to check if the file /var/www/html/flag\_new exists

stat:

path: /var/www/html/flag\_new

register: file\_data

- name: Clone a Git repo

git:

repo: 'https://github.com/kodekloudhub/learning-app-ecommerce.git'

dest: /var/www/html/

when: not file\_data.stat.exists

- name: Point the web server to the database server

replace:

path: /var/www/html/index.php

regexp: '172.20.1.101'

replace: '10.0.1.8'

- name: Read the contents of index.html

shell: cat /var/www/html/index.php

register: index\_content

- name: touch a file when the index is already replaced with 10.0.1.8

command: touch /var/www/html/flag\_new

when: index\_content.stdout.find('10.0.1.8') != -1

- name: Allow port 80 on server firewall

firewalld:

port: 80/tcp

permanent: true

state: enabled

ghp\_lLDCitiXIzuQH1Q7rQx3OE8OmZUTCf314E5P

set\_fact -> module used to initialise a variable within a script with a value

CPU monitoring script

-

name: Monitor CPU Utilisation

hosts: all

tasks:

- name: Monitor the CPU utilisation of a host for first second

shell: "vmstat 1 2|tail -1|awk '{print $15}'"

register: cpuutil\_1

- name: Monitor the CPU utilisation of a host for 2nd second

shell: "vmstat 1 3|tail -1|awk '{print $15}'"

register: cpuutil\_2

- name: Monitor the CPU utilisation of a host for 3rd second

shell: "vmstat 1 4|tail -1|awk '{print $15}'"

register: cpuutil\_3

- name: Generate alert if CPU utilisation is above 90%

debug:

msg: "CPU utilisation on {{ ansible\_hostname }} is above 90%"

when: cpuutil\_1.stdout|int < 10 and cpuutil\_2.stdout|int < 10 and cpuutil\_3.stdout|int < 10

The above script uses three separate registers and for each new second iteration a new register is to be added. This doesnot provide enough flexibility . Hence I have developed a different way of dealing with it .

We take the iterations in a variable and loop it and register all data in a single variable. This variable now cannot be referenced as var.stdout as it has the iterative data it needs to be accessed in a separate way .

I first debugged the variable for checking its contents by

Debug:

Var: <register name>

I found out that the register now pronts out values as <register>.results.<attributes> format . My interest is <register>.results.stdout . Now register.results is a list of map (list of dictionary) . So to access it we have to use <register>.results[0].stduout or <register>.results[1].stduout , so on and so forth

The script now looks like this::

- hosts: all

vars:

images:

- 2

- 3

- 4

- 5

- 6

- 7

tasks:

- shell: "vmstat 1 {{ item }}|tail -1|awk '{print $15}'"

register: "r"

with\_items: "{{images}}"

# - debug: var=r

# - debug: msg="item.item={{item.item}}, item.stdout={{item.stdout}}"

# with\_items: "{{r.results}}"

- debug:

msg: CPU utilisation in {{ ansible\_hostname }} is more than 90% for over 5 secs

when: r.results[0].stdout|int < 10 and r.results[1].stdout|int < 10 and r.results[2].stdout|int < 10 and r.results[3].stdout|int < 10 and r.results[4].stdout|int

< 10 and r.results[5].stdout|int < 10

**Variable and Variable files**:

Variables can be declared in various ways:

Variables can be declared within playbooks using vars:

Variables can also be declared in files and specified in vars\_files

There is though a more clean and organised way of passing variables to a play book:

This is done by group\_vars and host\_vars.

* Group\_vars -> For each group in inventory a file with the syntax <groupname>.yaml is created inside the group\_vars directory and inside that the variable values are declared .
* Host\_vars -> Under the host\_vars directory for each host a host.yaml file is created and host specific variables are passed .

Other than this extra vars are passed with –e flag while running the playbook . The precedence of the variable callings are as follows from lowest to highest:

1. Variables decalred in hosts.yaml (inventory file) -> lowest precedence
2. Group\_vars
3. Host\_vars
4. Vars: inside playbook
5. Vars\_file inside playbook
6. –e custom variable while executing runbook .

**Ansible-Roles:**

The bigger play book can be divided into smaller setups called a role . A role has a specific directory structure. Inside the directory which hold the main play book , the host\_vars/group\_vars etc , create another directory called roles . Inside the roles create a directory with the role name . Inside the role name directory , the following directories are created:

* Tasks -> Under tasks there is a main.yaml file where the tasks only are written . This tasks can further be broken by using import\_tasks option within the main.yaml file .
* Vars and defaults -> This two directories are used to hold variable values for the role. Each when populated should contain a main.yaml file
* Templates -> The files contained in this directory will be used to dynamically set values using the content of group\_vars /host\_vars variables . This will contain all dynamic files also which may be needed to be moved to a destination location in different servers
* Files -> Contains static files for transfer to various locations
* Handlers -> Contains a main.yaml file which defines the handler activity in case of a notification event that happens in a playbook
* Meta -> May contain meta data .

I have here used templates files to reference host\_vars/localhost.yaml and create a dynamic variable file and move it to role/plan/vars/main.yaml . This is then used by the role . The role is called in the main playbook .

[root@ansiblecontroller rolebased\_project]# ls -lrt

total 8

-rw-r--r--. 1 root root 335 Jan 12 13:26 inventory.txt

drw-r--r--. 3 root root 18 Jan 12 13:35 roles

drw-r--r--. 2 root root 25 Jan 12 15:00 group\_vars

drw-r--r--. 2 root root 28 Jan 13 06:34 host\_vars

-rw-r--r--. 1 root root 65 Jan 13 06:34 main.yaml

[root@ansiblecontroller rolebased\_project]# cd roles

[root@ansiblecontroller roles]# cd \*

[root@ansiblecontroller plan]# ls -lrt

total 0

drw-r--r--. 2 root root 6 Jan 13 06:26 files

drw-r--r--. 2 root root 6 Jan 13 06:26 handlers

drw-r--r--. 2 root root 21 Jan 13 06:29 templates

drw-r--r--. 2 root root 23 Jan 13 06:31 tasks

drw-r--r--. 2 root root 23 Jan 13 06:35 vars

[root@ansiblecontroller plan]# ls -lrt tasks

total 4

-rw-r--r--. 1 root root 279 Jan 13 06:31 main.yaml

[root@ansiblecontroller plan]# cat tasks/main.yaml

---

- name: Copy template to ariable file location

template:

src: ../templates/temp.j2

dest: /project/rolebased\_project/roles/plan/vars/main.yaml

- name: Install a package

yum:

name: "{{ item }}"

state: latest

loop: "{{ packages }}"

[root@ansiblecontroller plan]# ls -lrt templates

total 4

-rw-r--r--. 1 root root 30 Jan 13 06:29 temp.j2

[root@ansiblecontroller plan]# cat templates/temp.j2

---

packages: {{ packages }}

[root@ansiblecontroller plan]# ls -lrt vars

total 4

-rw-r--r--. 1 root root 35 Jan 13 06:35 main.yaml

[root@ansiblecontroller rolebased\_project]# ls -lrt

total 8

-rw-r--r--. 1 root root 335 Jan 12 13:26 inventory.txt

drw-r--r--. 3 root root 18 Jan 12 13:35 roles

drw-r--r--. 2 root root 25 Jan 12 15:00 group\_vars

drw-r--r--. 2 root root 28 Jan 13 06:34 host\_vars

-rw-r--r--. 1 root root 65 Jan 13 06:34 main.yaml

[root@ansiblecontroller rolebased\_project]# pwd

/project/rolebased\_project

[root@ansiblecontroller rolebased\_project]# cat main.yaml

-

name: Install packages

hosts: localhost

roles:

- plan

[root@ansiblecontroller rolebased\_project]# cd host\_vars

[root@ansiblecontroller host\_vars]# ls -lrt

total 4

-rw-r--r--. 1 root root 25 Jan 13 06:34 localhost.yaml

[root@ansiblecontroller host\_vars]# cat localhost.yaml

packages:

- httpd

- wget

[root@ansiblecontroller host\_vars]#