**Ansible Assignment 2**

1. How can a list of hosts in a group be looped over within a template?

* This can be done by accessing the “$groups” dictionary in the template as syntax:

{% for host in groups['db\_servers'] %}

{{ host }}

{% endfor %}

* First we need to ensure that the facts are populated, For instance a play that talks to db\_servers:1
  1. hosts: db\_servers

tasks:

* 1. debug: msg="Something to debug"
* Now, we can use within a template, eg.

{% for host in groups['db\_servers'] %}

{{ hostvars[host]['ansible\_eth0']['ipv4']['address'] }}

{% endfor %}.

1. What is Ansible's ad-hoc command?

* Basically Ad-hoc commands are just like one-line playbooks to perform a specific task only.
* it is an alternative to writing playbooks.
* Even we can say Ansible ad hoc command uses the /usr/bin/ansible command-line tool to automate a single task on one or more managed nodes. ad hoc commands are quick and easy, but they are not reusable.
* The syntax is ansible [pattern] -m [module] -a "[module options]"
* For example, we need to reboot all servers in the staging group
* You can use an ad hoc task to call the command module and reboot all web servers in lets say GUJARAT, 10 at a time then use below comamnd

ansible GUJARAT -a "/sbin/reboot" -f 10 -u username --become [--ask-become-pass]

* in most of the OS rebooting probably requires privilege access so that can be achieved by **keyword : become**

1. How do I set up Nginx using the Ansible playbook?

* Step 1: Let us first generate a public SSH key to connect to your host.

ssh-keygen

* Step 2: Now copy the public SSH key on your hosts eg.

ssh-copy-id -i root@IP\_of\_target\_machine

* Step 3: Now list the IP addresses of your hosts/nodes in your inventory.

vi /etc/ansible/hosts

add the server ip. in the inventory with group name let's say [test-server]

|  |
| --- |
| [test-server]  192.168.10.10 |

* Step 4: Now let's check if the connection has been established using ping

**ansible -m ping 'test-server'**

* Step 5: Now we create a playbook to install Nginx on the host machine.

create a file with yml extention vi nginx\_demo.yml

|  |
| --- |
| --  hosts: stagingwebservers  sudo: yes  vars:  - server\_port: 8080    tasks:  - name: install nginx  yum: pkg=nginx state=installed    - name: serve nginx config  template: src=../files/flask.conf dest=/etc/nginx/conf.d/  notify:  - restart nginx  handlers:  - name: restart nginx  service: name=nginx state=restarted |

Save the file exit

in above playbook first task is to get the required package for Nginx and then install it.

Ansible will check if the directory exists and create it if it’s not otherwise will do nothing.

Handlers are define to take a action only when notification of tasks or state changes.

In this playbook we have defined notify: restart Nginx handler which will restart Nginx once the files and templates are copied to hosts.

* Step 6: Run the playbook, using the command below:

ansible-playbook nginx\_demo.yml

* Step 7: Check if Nginx is installed on the machine. Use the following command:

ps waux | grep nginx

1. How do I programmatically access the name of a variable?

* Variable names can be built by adding strings together like "~".
* For example, if we have a need to get ipv4 address of an some arbitrary interface and the interface to be used may be supplied through a role parameter or other input then we can do it in this way.

{{ hostvars[inventory\_your\_hostname]['ansible\_' ~ which\_interface]['ipv4']['address'] }}

* if your interface names have dashes, you must replace them with underscores:

{{ hostvars[inventory\_your\_hostname]['ansible\_' ~ which\_interface | replace('\_', '-') ]['ipv4']['address'] }}

* hostvars is a hash with inventory hostnames as keys.
* One more eg. if you have :

[host\_group]

host-1 ansible\_ssh\_host=192.168.0.21 node\_name=foo

host-2 ansible\_ssh\_host=192.168.0.22 node\_name=bar

[host\_group:vars]

custom\_var=asdasdasd

* You can access host group vars using:

{{ hostvars['host\_group'].custom\_var }}

* If you need a specific value from specific host, you can use:

{{ hostvars[groups['host\_group'][0]].node\_name }}

1. How do Ansible and Puppet vary from one other?

* Let us see below comparison which helps to understand how does both vary from one another.

|  |  |
| --- | --- |
| Ansible | Puppet |
| Ansible is easy to setup and it is agentless | It is hard to setup when compared with ansible |
| the server pushes the configuration to the nodes | the client pulls the configuration from the server |
| It does not have that feature in the free version. | the puppet has an agent who polls every 30mins(default settings) to make sure all nodes are in a desirable state |
| Ansible has backup secondary nodes | puppet has more than one master node |
| It uses YAML and Python | It uses DSL(PuppetDSL) |

1. What is the purpose of Ansible Tower, and what are its characteristics?

* Ansible Tower is basically an enterprise-level commercial product by RedHat. It is GUI base.
* It provides a web-based console and REST API to manage Ansible across teams in an organization.
* provides the user-interface to centrally execute and monitor Ansible playbooks across a complex enterprise IT environment
* Ansible Tower provides RBAC(role based access control) credentials management, delegation of jobs, compliance, and many other higher-level management features critical to a successful deployment of Ansible in an enterprise
* There are many features such as
* We can set up different dependencies among playbooks, or running multiple playbooks maintained by different teams at once
* if we want to know the status of any play or tasks can be monitored easily and also we can check what’s going to run next
* It helps to tracking logs are very important so that we can easily revert back to a previous state if something goes wrong.
* We can use the tower to run any command to a host or group of hosts in our inventory.
* There are so many other things that can be done with tower such as Job Scheduling, Notification Integration, Multi-Playbook Workflows ,CLI, etc.
* We can connect multiple Ansible Tower nodes into an Ansible Tower cluster as the clusters add redundancy and capacity, which allow you to scale Ansible automation across the enterprise.
* Integrated Notifications – This feature lets you notify a person or team when a job succeeds or fails across the entire organization at once, or customize on a per-job basis.

1. Describe how you'll recursively copy files to a destination host.

* In Ansible we already have a built in copy module, we have with the name synchronize which is more efficient when we have huge amount of files

|  |
| --- |
| - synchronize:  src: /first/absolute/path  dest: /second/absolute/path  delegate\_to: "{{ inventory\_hostname }}" |

If we want to copy simple condition then can be achieved by

|  |
| --- |
| - name: to keep a backup  copy:  remote\_src: yes  dest: /x/y/z  src: "{{ updated['backup\_file'] }}" |

1. What is the most effective method for making content reusable and redistributable?

* In Ansible we have Ansible roles which we can use to make content reusable and redistributable
* Ansible roles are basically a level of abstraction to organize playbooks.
* They provide a skeleton for an independent and reusable collection of variables, tasks, templates, files, and modules which can be automatically loaded into the playbook. Playbooks are a collection of roles. Every role has specific functionality.
* For example, if we have a need to execute 20 tasks on 10 systems, writing all of them in the playbook will be complex and twill create too many confusion. so there is a way like if we create 20 roles and call them inside the playbook.

1. What are handlers, and what do they do?

* Handlers are used to trigger the status of a service such as restarting or stopping a service.
* sometimes when a task make a change to the system, a further task may need to be run.
* For example, a change to a service's configuration file may then require that the service be reloaded so that the changed configuration takes effect.
* Handlers are tasks that respond to a notification triggered by other tasks.
* for eg. after installation of let's say nginx we do need to start them. let see below eg.

tasks:

* + name: install nginx

apt: pkg=nginx state=installed update\_cache=true

notify:

* + start nginx

handlers:

* + name: start nginx

service: name=nginx state=started

1. How can a user module generate encrypted passwords?

* on most of the linux system we have utility called mkpasswd with the help of it we can generate. below is the command

ansible all -i localhost, -m debug -a "msg={{ 'mypassword' | password\_hash('sha512', 'mysecretsalt') }}"

* on system like mac we can use python

pip install passlib

python -c "from passlib.hash import sha512\_crypt; import getpass; print(sha512\_crypt.using(rounds=5000).hash(getpass.getpass()))"

1. What is the difference between dot notation and array notation for variables?

* The dot notation comes from Jinja and works fine for variables without special characters.
* If your variable contains dots (.), colons (:), or dashes (-), if a key begins and ends with two underscores, or if a key uses any of the known public attributes, it is safer to use the array notation.
* Also array notation allows for dynamic variable composition, see dynamic\_variables.
* Another problem with ‘dot notation’ is that some keys can cause problems because they collide with attributes and methods of python dictionaries.
* for eg.

item.update # this breaks if item is a dictionary, as 'update()' is a python method for dictionaries

item['update'] # this works

1. What is the purpose of the Ansible synchronize module?

* It is a module similar to rsync in Linux machines which we can use in playbooks.
* feature wise it is similar to rsync such as archive, compress, delete but also note that there are few limitations also such as
* Rsync must be installed on both source and target systems
* Need to specify delegate\_to to change the source from localhost to some other port
* Need to handle user permission as files are accessible as per remote user.
* We should always give the full path of the destination host location in case we use sudo otherwise files will be copied to the remote user home directory.
* Linux rsync limitations related to hard links are also applied here.
* It forces -delay-updates to avoid the broken state in case of connection failure
* in below eg. we are transferring files of /var/tmp/sync\_folder folder to remote machine /var/tmp/ineuron\_poc folder

|  |
| --- |
| ---  - hosts: host-remote tasks:  - name: sync from sync\_folder  synchronize:  src: /var/tmp/sync\_folder dest: /var/tmp/ineuron\_poc |

1. What is the purpose of the Ansible firewalld module?

* firewalld manages arbitrary ports and services using firewall rules to allow or block services from the port. it can be into two types: **Zones** and **Services**
* Zones refers to location for which we can control which services are exposed to or a location to which one the local network interface is connected.
* Services are typically a series of port/protocol combinations or sockets that your host may be listening on, which can then be placed in one or more zones
* below is few eg.

|  |
| --- |
| - name: permit traffic in default zone for https service  ansible.posix.firewalld:  service: https  permanent: yes  state: enabled    - name: do not permit traffic in default zone on port 8081/tcp  ansible.posix.firewalld:  port: 8081/tcp  permanent: yes  state: disabled |

1. What distinguishes the Ansible set fact module from vars, vars file, and include var?

* set\_fact is basically just like ansible fact and it used to set new variable values on a host by host basis discovered by the setup module. These variables are available to subsequent plays in a playbook.
* While vars, vars\_file, or include\_var we know the value beforehand whereas when using set\_fact, we can store the value after preparing it on the fly using certain tasks like using filters or taking subparts of another variable.
* We can also set a fact cache over it.
* set\_fact variable assignment is done by using key-pair values where the key is the variable name and the value is the assignment to it.
* Below is the example

- set\_fact:

one\_fact: value1

second\_fact:

value2

1. When is it risky to use a variable to bulk-set task arguments?

* In ansible all task arguments will be dictionary-typed variables which can be useful in some dynamic execution scenarios also Ansible will issues a warning since it introduces a security risk.

|  |
| --- |
| vars:  usermod\_args:  name: testuser  state: present  update\_password: always  tasks:  - user: '{{ usermod\_args }}' |

* In the above e.g the values passed to the variable is usermod\_args and this could be overwritten by some other malicious values in the host facts on a compromised target machine.
* To avoid this there below are the points to take care like:
* bulk variable precedence should be greater than host facts.
* we need to disable INJECT\_FACTS\_AS\_VARS configuration to avoid collision of fact values with variables.