Ansible Control Node, Inventory & Managed Nodes

By: Er. Vikas Nehra (M. Tech, B. Tech), Experience: 15 + Years

Session - 6 Agenda:
1. Ansible Ad-Hoc Commands
2. Ansible Modules

Important Ansible Module Names:

1. *Copy*
2. *Command*
3. *Raw*
4. *Shell*
5. *File*
6. *Fetch*
7. *Get_url*
8. *Lineinfile*
9. *Replace*
10. *User*
11. *Group*
12. *Yum/Dnf/Apt*
13. *Yum_repository*
14. *Package*
15. *Stat*
16. *Mount*
17. *Setup*
18. *Service*
19. *Systemd*
20. *Debug*
21. *Uri*
22. *Parted*
23. *Filesystem*
24. *Cron*
25. *Script*

Quick Recap of Session – 4:

User Module:
Yum/Dnf/Apt Module:
Package Module:

User Module: (Manage user accounts and user attributes.) It is used for the same job that we perform using the commands like useradd, usermod, userdel etc.
Let’s create a user named ‘rahul’ having secondary group as wheel & uid as 1010 in all the RHEL managed nodes.

$ ansible node2 -m user -a 'name=rahul state=present uid=1010 groups=wheel'

Groups: for secondary group(s)
Group: for primary group

Verify the changes.

$ ansible node2 -m command -a 'id -a rahul'

OR

```
$ ansible node2 -m command -a 'tail -1 /etc/passwd'
```

Let’s create a user named ‘vikas’ having primary group as wheel with uid as 1015 in all the managed nodes.

```
$ ansible node2 -m user -a 'name=vikas state=present uid=1015 group=wheel'
```

Verify the changes.

```
$ ansible node2 -m command -a 'id -a vikas'
```

To remove the Rahul user:

```
$ ansible node2 -m user -a 'name=rahul state=absent'
```

Verify the changes.

```
$ ansible node2 -m command -a 'id -a rahul'
```

Let’s create a user account named admin with password on all the managed nodes.

```
$ ansible RHEL9 -m user -a "name=admin update_password=always password={{ newpassword|password_hash('sha512') }}" --extra-vars "newpassword=12345678"
```

Verify the changes.

```
$ ssh node2
$ su - admin
```

Yum/Dnf/Apt Module: (Installs, upgrade, downgrades, removes, and lists packages and groups with the `yum' package manager.) It is used to perform the jobs on the managed nodes that we perform on a machine using yum/dnf/apt commands.

Yum: It works on all Fedora based distributions like RHEL.

Dnf: It works on all Fedora based distributions like RHEL but not on RHEL 6 & previous versions.

Apt: It works on all the Debian based distributions like Ubuntu.

Let’s install a package e.g., zsh on the Fedora based managed nodes using yum module.

Let’s execute the ad-hoc command with an incorrect argument.

```
$ ansible all -m yum -a 'name=zsh state=presentss'
$ ansible all -m yum -a 'name=zsh state=present'
```

It requires subscription or yum repository already present on your node machines.

Here present or installed have the same meaning, similarly absent or removed have the same meaning. Latest means updated version of the packages need to be installed.

Verify the changes.

```
$ ansible all -m command -a 'rpm -qi zsh'
```

Let’s install a package e.g., httpd on the Fedora based managed nodes using dnf module.

```
$ ansible all -m dnf -a 'name=httpd state=latest'
```

Verify the changes.

```
$ ansible all -m command -a 'rpm -qi httpd'
```

Let’s Install zsh package on Debian based nodes:

Let’s try with installed argument first which is not supported by apt module.

```
$ ansible all -m apt -a 'name=zsh state=installed'
```

You can only use those arguments which are supported by that module. Eg., present or latest in this case of apt module. (not installed or removed)

```
$ ansible all -m apt -a 'name=zsh state=present'
```

Verify the changes.

```
$ ansible Ubuntu -m command -a 'apt list zsh'
$ ansible RHEL9 -m command -a 'yum list installed | grep zsh'
```

In case if we want to install any package on all the linux nodes we can use generic module called package module.

Package Module: (This module manages packages on a target without specifying a package manager module (like ansible.builtin.yum, ansible.builtin.apt, …).
It is convenient to use in a heterogeneous environment of machines without having to create a specific task for each package manager.)

```
$ ansible-doc package
```

Let's install ksh package on all the managed nodes using package module.

```
$ ansible all -m package -a 'name=ksh state=latest'
```

Verify the changes.

```
$ ansible Ubuntu -m command -a 'apt list ksh'
$ ansible RHEL9 -m command -a 'yum list installed | grep ksh'
```

Here keep in mind that with package module installed or removed arguments doesn't work for Debian based machines because they are not supported by apt.
Let's try to use installed argument.

```
$ ansible Ubuntu -m package -a 'name=apache2 state=installed'
$ ansible Ubuntu -m package -a 'name=apache2 state=latest'
```

Verify the changes.

```
$ ansible all -m command -a 'apt list apache2'
```

You can also specify the command in argument field which you want to use with package module to manage the packages on the managed nodes like yum, dnf and apt. But it should be only used if package module is not able to detect the target OS for some reason.

```
$ ansible RHEL9 -m package -a 'name=ftp state=latest use=dnf'
```

Verify the changes.

```
$ ansible RHEL9 -m command -a 'rpm -qi ftp'
```

---

15. Stat Module: (The stat command is used to print out the status of Linux files, directories and file systems. Unlike ls command, stat prints out a lot of information regarding files, directories and file systems such as their sizes, blocks, inodes, permissions, timestamps for modification, access, change dates etc.)

Let's check the existence of any file e.g., xyz.conf in /etc/ directory on all the nodes.

```
$ ansible all -m stat -a 'path=/etc/xyz.conf'
```

Let's check the existence of any existing file e.g., passwd in /etc/ directory on all the nodes.

```
$ ansible all -m stat -a 'path=/etc/passwd'
```

Let’s check the /dev/nvme0n1 file (disk).

```
$ ansible all -m stat -a 'path=/dev/nvme0n1'
```

---

16. Mount Module: (This module controls active and configured mount points in `/etc/fstab'.)

```
$ ansible-doc mount
```

Let’s make an entry in fstab file using mount module.

```
$ ansible node2 -m mount -a 'path=/opt/ src=/dev/sda fstype=xfs opts=defaults state=present'
```

Mount module is not available in ansible-core 2.12.2.

```
$ ansible-doc -l | grep mount
$ ansible-doc -l | wc -l
```

In some cases, you will find that this module is missing in your ansible environment. For such cases you can install ansible.posix from ansible galaxy using below command.

```
$ ansible-galaxy collection install ansible.posix
```

Verify that the module has been installed successfully.

```
$ ansible-doc -l | grep mount
$ ansible-doc -l | wc -l
$ ansible-doc mount
```

Try again:

```
$ ansible node2 -m mount -a 'path=/opt/ src=/dev/sda fstype=xfs opts=defaults state=present'
```

Verify the changes.

```
$ ansible node2 -m command -a 'cat /etc/fstab'
```

But it is currently not mounted.

```
$ ansible node2 -m command -a 'df -hT'
```

To mount we can mention mounted in state agrument.

```
$ ansible node2 -m mount -a 'path=/opt/ src=/dev/sda fstype=xfs opts=defaults state=mounted'
```

Create filesystem if not created earlier.

```
$ ansible node2 -m command -a 'mkfs.xfs /dev/sda'
$ ansible node2 -m mount -a 'path=/opt/ src=/dev/sda fstype=xfs opts=defaults state=mounted'
```

Verify the changes.

```
$ ansible node2 -m command -a 'cat /etc/fstab'
```

It will show up currently mounted.

```
$ ansible node2 -m command -a 'df -hT'
```

To unmount the filesystem:

```
$ ansible node2 -m mount -a 'path=/opt src=/dev/sda state=unmounted'
```

To remove the entry from fstab file:

```
$ ansible node2 -m mount -a 'path=/opt src=/dev/sda state=absent_from_fstab'
```

17. Setup Module: (This module is automatically called by playbooks to gather useful variables about remote hosts that can be used in playbooks.)
It is used to collect system information from the managed nodes in json format like we do with dmidecode command.

```
$ ansible all -m setup
```

Let’s collect the kernel version information from the managed nodes.

```
$ ansible all -m setup -a 'filter=*kernel*'
```

To save the information in a file.

```
$ ansible all -m setup -a 'filter=*kernel*' > kernel.txt
```

Exact keyword is: ansible_kernel

```
$ ansible all -m setup -a 'filter=ansible_kernel'
```

It can also be used for conditional operations using playbooks.

18. Service Module: (Ansible's service module controls services on remote hosts and is useful for these common tasks: Start, stop or restart a service on a remote host.)

```
$ ansible-doc service
```

Let’s start and enable the httpd service on all the RHEL nodes. (Install httpd packages first.)

```
$ ansible all -m service -a 'name=httpd state=started enabled=true'
```

It will work for all linux distributions and versions.

To restart the service:

```
$ ansible all -m service -a 'name=httpd state=restarted'
```

To stop & disable the service:

```
$ ansible all -m service -a 'name=httpd state=stopped enabled=false'
```

Thank You