Ansible Control Node, Inventory & Managed Nodes

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

Session - 7 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. *LVG*
25. *LVOL*
26. *Cron*

19. Systemd Module: (Controls systemd units (services, timers, and so on) on the remote hosts.)
Read the documentation:
$ ansible-doc systemd

Let's force systemd to reread configurations on all the managed nodes in RHEL9 group.
$ ansible RHEL9 -m systemd -a 'daemon-reload=true'

Just force systemd to re-execute itself on all the managed nodes in RHEL9 group.
$ ansible RHEL9 -m systemd -a 'daemon_reexec=true'

Let's start and enable the httpd service using systemd module on the managed nodes. (Make sure httpd package is already installed.)
$ ansible RHEL9 -m package -a 'name=httpd state=latest'

$ ansible RHEL9 -m systemd -a 'name=httpd state=started enabled=true'

Ansible Control Node, Inventory & Managed Nodes

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

Verify the changes.
$ ansible RHEL9 -m command -a 'systemctl status httpd'

Let's restart the httpd service in all the managed nodes.
$ ansible RHEL9 -m systemd -a 'name=httpd state=restarted'

Restart Vs Reload:
Restart shuts the service down and then starts it up again, whereas reload instructs the daemon to reload its configuration.

Let's reload the httpd service in all the managed nodes.
$ ansible RHEL9 -m systemd -a 'name=httpd state=reloaded'

Let's stop the httpd service in all the managed nodes.
$ ansible RHEL9 -m systemd -a 'name=httpd state=stopped'

Let's disable the httpd service in all the managed nodes.
$ ansible RHEL9 -m systemd -a 'name=httpd enabled=false'

Verify the changes.
$ ansible RHEL9 -m command -a 'systemctl status httpd'

Masking:
When masking a service, the symbolic links are moved and then point to /dev/null. When you simply disable a service, it can still be started manually. When you mask a service, it cannot be started manually. In other words, masking a service renders the service permanently unusable until it's unmasked.

Let's explain it using below example of httpd service.
 # dnf install -y httpd

 # echo Nehra Classes Are Awesome > /var/www/html/index.html

 # systemctl start httpd

 # systemctl status httpd
- ***httpd.service - The Apache HTTP Server***
  Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; vendor preset: disabled)

 # systemctl enable httpd
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /usr/lib/systemd/system/httpd.service.

 # ll /etc/systemd/system/multi-user.target.wants/httpd.service

 # systemctl status httpd
- ***httpd.service - The Apache HTTP Server***
  Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset: disabled)

 # curl localhost
Nehra Classes Are Awesome

 # systemctl disable --now httpd
Removed /etc/systemd/system/multi-user.target.wants/httpd.service.

 # systemctl status httpd
○ httpd.service - The Apache HTTP Server
Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; vendor preset: disabled)

 # curl localhost
curl: (7) Failed to connect to localhost port 80: Connection refused

 # systemctl enable --now httpd
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /usr/lib/systemd/system/httpd.service.

 # curl localhost
Nehra Classes Are Awesome

 # systemctl disable --now httpd
Removed /etc/systemd/system/multi-user.target.wants/httpd.service.

 # systemctl status httpd
○ httpd.service - The Apache HTTP Server
Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; vendor preset: disabled)

 # curl localhost
curl: (7) Failed to connect to localhost port 80: Connection refused

 # systemctl mask httpd
Created symlink /etc/systemd/system/httpd.service → /dev/null.

 # systemctl status httpd
○ httpd.service
Loaded: masked (Reason: Unit httpd.service is masked.)
Active: inactive (dead)

 # systemctl enable --now httpd
Failed to enable unit: Unit file /etc/systemd/system/httpd.service is masked.

 # curl localhost
curl: (7) Failed to connect to localhost port 80: Connection refused

 # systemctl unmask httpd
Removed /etc/systemd/system/httpd.service.

 # systemctl enable --now httpd

 # systemctl status httpd
● httpd.service - The Apache HTTP Server
Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset: disabled)

 # curl localhost
Nehra Classes Are Awesome

 # systemctl list-units
 # systemctl list-units | grep masked

 # systemctl mask httpd
Created symlink /etc/systemd/system/httpd.service → /dev/null.

 # systemctl list-units | grep masked
● httpd.service masked active running httpd.service

Let's mask the httpd service in all the managed nodes.
$ ansible RHEL9 -m systemd -a 'name=httpd masked=true'

Verify the changes.
$ ansible RHEL9 -m command -a 'systemctl status httpd'

Let's unmask the httpd service in all the managed nodes.
$ ansible RHEL9 -m systemd -a 'name=httpd masked=false'

---

20. Debug Module: (This module prints statements during execution and can be useful for debugging variables or expressions without necessarily halting the playbook.)
Let's print hello world on the managed nodes using debug module.
$ ansible all -m debug -a 'msg="Hello World"'

Similarly, we can print Nehra Classes Are Awesome.
$ ansible all -m debug -a 'msg="Nehra Classes Are Awesome."'

Let's print the ansible core version using variable in debug module.
$ ansible localhost -m debug -a 'msg="{{ ansible_version.full }}"'

Let's print the inventory hostname of all the managed nodes using debug module.
$ ansible all -m debug -a 'msg="{{ inventory_hostname }}"'

---

21. Uri Module: (Interacts with HTTP and HTTPS web services and supports Digest, Basic and WSSE HTTP authentication mechanisms.)
URI module is used for URL, port number, IP addresses testing etc.
Let's test a URL whether it is available at port number 9898 on all the managed nodes.
$ ansible all -m uri -a 'url=http://api.nehraclasses.com:9898'

Let's test whether https://www.nehraclasses.in URL is accessible from all nodes or not?
$ ansible all -m uri -a 'url=https://www.nehraclasses.in'

---

22. Parted Module: (This module allows configuring block device partition using the parted command line tool.)
$ ansible-doc parted

Ansible Control Node, Inventory & Managed Nodes

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

This module is not available in ansible-core.
You might already have this collection installed if you are using the ansible package. It is not included in ansible-core.

To install ansible.posix we can execute below command.
$ ansible-galaxy collection install ansible.posix

To check whether it is installed:
$ ansible-galaxy collection list

Ansible Posix Modules:

acl module –	*Set and retrieve file ACL information.*
at module –	*Schedule the execution of a command or script file via the at command*
authorized_key module –	*Adds or removes an SSH authorized key*
firewalld module –	*Manage arbitrary ports/services with firewalld*
firewalld_info module –	*Gather information about firewalld*
mount module –	*Control active and configured mount points*
patch module –	*Apply patch files using the GNU patch tool*
rhel_facts module –	*Facts module to set or override RHEL specific facts.*
rhel_rpm_ostree module –	*Ensure packages exist in a RHEL for Edge rpm-ostree based system*
rpm_ostree_upgrade module –	*Manage rpm-ostree upgrade transactions*
seboolean module –	*Toggles SELinux booleans*
selinux module –	*Change policy and state of SELinux*
synchronize module –	*A wrapper around rsync to make common tasks in playbooks*
sysctl module –	*Manage entries in sysctl.conf.*

Unfortunately parted module is not there in ansible.posix collection as well. Now we have to create or install this module from community.general collection of ansible galaxy.
$ ansible-galaxy collection install community.general

Now, we will have many more additional modules installed in our ansible server machine.
$ ansible-doc -l | wc -l

Now we can go check its documentation as well.
$ ansible-doc parted

Check & identify the disk where partition is to be created (e.g. /dev/sda) on the managed nodes.
$ ansible node2 -m command -a 'lsblk'

Let's create the first partition of 10GB on /dev/sda disk on the managed nodes.
$ ansible node2 -m parted -a 'device=/dev/sda number=1 part_end=10GiB state=present'
We can also define the type of the partition there like primary or secondary.

Verify the changes.
$ ansible node2 -m command -a 'lsblk'

Let's create another partition (number 2) of 5GB on /dev/sda disk on the managed nodes.
$ ansible node2 -m parted -a 'device=/dev/sda number=2 part_start=10GiB part_end=15GiB state=present'

Verify the changes.
$ ansible node2 -m command -a 'lsblk'

Let's create one extended partition (number 3) of 4GB on /dev/sda disk on the managed nodes.

$ ansible node2 -m parted -a 'device=/dev/sda number=3 part_start=15GiB part_end=19GiB part_type=extended state=present'

ansible rhel -m parted -a 'device=/dev/sdb number=3 part_start=2GiB part_end=100% part_type=extended state=present' -b

Verify the changes.

$ ansible node2 -m command -a 'lsblk'

ansible rhel -m parted -a "device=/dev/sdb number=5 part_start=2049MiB part_end=4096MiB part_type=logical state=present" -b

Let's remove the extended partition created earlier on the managed nodes.

$ ansible node2 -m parted -a 'device=/dev/sda number=3 state=absent'

Verify the changes.

$ ansible node2 -m command -a 'lsblk'

---

23. Filesystem Module: (This module creates a filesystem.)

Let's check the partition first, whether it has the filesystem or not on the managed nodes on /dev/sda1.

$ ansible node2 -m command -a 'lsblk -f'

OR

$ ansible node2 -m command -a 'blkid'

Let's create the XFS filesystem on the first partition created earlier on /dev/sda disk on the managed node using filesystem module.

$ ansible node2 -m filesystem -a 'device=/dev/sda1 fstype=xfs'

Verify the changes.

$ ansible node2 -m command -a 'lsblk -f'

OR

$ ansible node2 -m command -a 'blkid'

---

24. LVG Module: (This module creates, removes or resizes volume groups.)

$ ansible-doc lvg

Let's create a volume group having name as VolGrp on the /dev/sda2 partition created earlier on the managed nodes.

$ ansible node2 -m lvg -a 'pvs=/dev/sda2 vg=VolGrp state=present'

Verify the changes.

$ ansible node2 -m command -a 'vgs'

Similar it can be resized as well. Let's increase the VolGrp using /dev/sda1 on the managed nodes.

$ ansible node2 -m lvg -a 'pvs=/dev/sda1,/dev/sda2 vg=VolGrp state=present'

Verify the changes.

$ ansible node2 -m command -a 'vgs'

To remove the VolGrp volume group:

$ ansible node2 -m lvg -a 'vg=VolGrp state=absent'

Verify the changes.

$ ansible node2 -m command -a 'vgs'

Ansible Control Node, Inventory & Managed Nodes

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

Let's create the VolGrp again on the managed nodes.
$ ansible node2 -m lvg -a 'pvs=/dev/sda1,/dev/sda2 vg=VolGrp state=present'

Verify the changes.
$ ansible node2 -m command -a 'vgs'

---

25. LVOL Module: (This module creates, removes or resizes logical volumes.)
$ ansible-doc lvol

Let's create an LVM of 4GB with name as LogVol on the VolGrp created earlier on the managed nodes.
$ ansible node2 -m lvol -a 'vg=VolGrp lv=LogVol size=4G state=present'

Verify the changes.
$ ansible node2 -m command -a 'lvs'

Let's extend this LogVol with 2G on the managed nodes.
$ ansible node2 -m lvol -a 'vg=VolGrp lv=LogVol size=+2G state=present'

Verify the changes.
$ ansible node2 -m command -a 'lvs'

Let's create one more logical volume having all free space on the volume group in the managed nodes.
$ ansible node2 -m lvol -a 'vg=VolGrp lv=test size=100%FREE state=present'

Verify the changes.
$ ansible node2 -m command -a 'lvs'
$ ansible node2 -m command -a 'vgs'

To remove the test logical volume from the managed nodes:
$ ansible node2 -m lvol -a 'vg=VolGrp lv=test state=absent force=true'

Verify the changes.
$ ansible node2 -m command -a 'lvs'
$ ansible node2 -m command -a 'vgs'

Let's check the partitions to create the filesystem on the LogVol on the managed nodes.
$ ansible node2 -m command -a 'lsblk -f'

Let's create the ext4 filesystem on LogVol on the managed nodes.
$ ansible node2 -m filesystem -a 'device=/dev/VolGrp/LogVol fstype=ext4'

Verify the changes.
$ ansible node2 -m command -a 'lsblk -f'

Now, we can use the mount module to mount any directory on this filesystem on the managed nodes.
$ ansible node2 -m command -a 'df -Th'

$ ansible node2 -m mount -a 'path=/opt/ src=/dev/VolGrp/LogVol fstype=ext4 opts=defaults state=mounted'

ansible rhel -b -m community.general.lvol -a "vg=VolGrp lv=LogVol size=100%VG resizefs=true state=present" -b

Verify the changes.

$ ansible node2 -m command -a 'df -Th'

---

26. Cron Module: (Use this module to manage crontab and environment variables entries.)

$ ansible-doc cron

Let's check there is no cron job scheduled for the root user on the managed nodes.

$ ansible all -m command -a 'crontab -l -u root'

Let's create a cronjob on all the managed nodes to append a line in /var/log/messages file 'NehraClasses' on every Monday at 10:30 PM under root user.

$ ansible all -m cron -a 'name=nehra weekday=1 minute=30 hour=22 user=root job="echo NehraClasses > /var/log/messages"'

Verify the changes.

$ ansible all -m command -a 'crontab -l -u root'

Now you can also check & test it on the managed node, by manually changing the date and time of the machine.

$ sudo timedatectl set-ntp false

$ timedatectl set-time "2023-03-06 22:29:40"

$ sudo tail -f /var/log/messages

Thank You

ansible all -b -m cron -a 'name=nehra minute="*/1" job="echo ganesh >> /var/log/messages" user=root' -b