Schedule Future Task Using Ansible

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

Session - 34 Agenda:

1. Schedule Future Tasks Using Ansible

Schedule/Manage Future Tasks in Linux:

We can schedule/manage future jobs (tasks) using at & crontab commands in Linux. At command is preferred for the one-time jobs which are non-repetitive in nature while the crontab is preferred over at command to schedule the repetitive jobs.

AT:

At command is used to schedule the one-time jobs which are non-repetitive.

```
# rpm -q at
# systemctl status atd.service
# dnf install -y at ; systemctl enable --now atd.service
# man at
# tty
# at 09:40 PM 17 July 2023
df -hT > /tmp/filesystems.txt
# atq
# at -l
# tail /var/log/cron
# rm -rf /tmp/*
# date
# ls -lh /tmp/
# cat /tmp/filesystems.txt
# at 21:43
lsblk > /tmp/blocks.txt
# atq
# date
# ls -lh /tmp/
# cat /tmp/filesystems.txt
# cat /tmp/blocks.txt
# echo "Nehra Classes Are Awesome" > /dev/pts/2 | at 09:45 PM
# atq
# date
# at Monday +20 minutes
ls -lh /tmp/
cat /tmp/blocks.txt
# atq
# atrm 5
# atq
# at now +5 hours
reboot
# atq
# echo "shutdown -h now" | at -m 4:30
# atq
# tail /var/log/cron
# at -c 5
# cd /var/spool/at
# ls -lh
# vim /etc/at.deny
vikasnehra
# su - vikasnehra
$ at now +2min
$ exit
# vim /etc/at.deny
#vikasnehra
# su - vikasnehra
$ at now +2min
$ exit
```

BATCH:

Batch command is similar to at command but the difference is at command is executed at a certain time while the batch command is executed at a certain load on the system.

```
# man batch
# echo NehraClasses > /tmp/output.txt | batch
# atq
# tail /var/log/cron
# ls -lh /tmp/output.txt
# cat /tmp/output.txt
# at -c 7
```

CRONTAB:

Crontab command is preferred over at command to schedule the repetitive jobs.

```
# rpm -q crontabs
# man crontab
# systemctl status crond.service
# dnf install -y crontabs
# systemctl enable --now crond.service
# cat /etc/crontab
min(0-59) hour(0-23) day(1-31) month(1-12) week(0-6)
# date
# crontab -e
55 21 * * * echo Nehra Classes Are Awesome > /dev/pts/0
# crontab -l
# ls -d /etc/cron.*
# crontab -l
# cat /etc/anacrontab
used in machines which are not running 24x7
# su - vikasnehra
$ crontab -e
59 21 * * * date >> /tmp/date.txt
*/2 * * * * cal >> /tmp/cal.txt
$ crontab -l
$ cat /tmp/date.txt
$ cat /tmp/cal.txt
$ exit
# crontab -l -u vikasnehra
# crontab -e -u vikasnehra
# tail /var/log/cron
# crontab -r
# crontab -l
# crontab -r -u vikasnehra
# crontab -l -u vikasnehra
# vim /etc/cron.deny
vikasnehra
# su - vikasnehra
$ crontab -e
$ exit
# vim /etc/cron.deny
#vikasnehra
# su - vikasnehra
$ crontab -e
```

Managing Future Tasks Using Ansible:

We can manage/schedule future tasks using Ansible in two ways by using:

1. Ansible Ad-hoc Commands
2. Ansible Playbooks

1. Ansible Ad-hoc Commands:

We can use command module, at module, cron module etc. in the Ansible ad-hoc commands to manage/schedule the future tasks in the managed node.

```
$ ansible-galaxy collection install ansible.posix
$ ansible node1 -m shell -a 'echo "Nehra Classes Are Awesome" > /tmp/file1.txt | at 21:55' -b
$ ansible node1 -m command -a 'atq' -b
$ ansible node1 -m command -a 'at -c 1' -b
$ ansible node1 -m command -a 'date' -b
$ ansible node1 -m command -a 'ls -lh /tmp/file1.txt' -b
$ ansible node1 -m command -a 'tail /var/log/cron' -b
$ ansible node1 -m command -a 'cat /tmp/file1.txt' -b
$ ansible node1 -m shell -a 'echo reboot | at now +4 hours' -b
$ ansible node1 -m command -a 'atq' -b
$ ansible node1 -m command -a 'atrm 2' -b
$ ansible node1 -m command -a 'atq' -b
$ ansible node1 -m at -a 'command=lsblk>/tmp/lsblk.txt count=1 units=minutes' -b
$ ansible node1 -m command -a 'atq' -b
$ ansible node1 -m command -a 'at -c 3' -b
$ ansible node1 -m command -a 'date' -b
$ ansible node1 -m command -a 'ls -lh /tmp/lsblk.txt' -b
$ ansible node1 -m command -a 'tail /var/log/cron' -b
$ ansible node1 -m at -a 'command=df>/tmp/df.txt count=3 units=days' -b
$ ansible node1 -m command -a 'atq' -b
$ ansible node1 -m command -a 'at -c 4' -b
$ ansible node1 -m command -a 'date' -b
$ ansible node1 -m command -a 'ls -lh /tmp/lsblk.txt' -b
$ ansible node1 -m command -a 'tail /var/log/cron' -b
$ ansible node1 -m at -a 'command=df>/tmp/df.txt count=3 units=days state=absent' -b
$ ansible node1 -m command -a 'atq' -b
$ ansible node1 -m cron -a 'atq' -b
$ ansible node1 -m cron -a 'name=hostnameinfo job=hostnamectl>/tmp/host.txt minute=10 hour=22' -b
$ ansible node1 -m command -a 'crontab -l' -b
$ ansible node1 -m cron -a 'name=fileread job="cat /etc/hosts" minute=15 hour=22 user=vikasnehra' -b
$ ansible node1 -m command -a 'crontab -l -u vikasnehra' -b
$ ansible node1 -m cron -a 'name=fileread job="cat /etc/hosts" minute=15 hour=22 user=vikasnehra state=absent' -b
$ ansible node1 -m command -a 'crontab -l -u vikasnehra' -b
$ ansible node1 -m cron -a 'name=update job="yum update" minute=10 hour=0 weekday=6 user=root' -b
$ ansible node1 -m command -a 'crontab -l' -b
$ ansible node1 -m command -a 'tail /var/log/cron' -b
$ ansible node1 -m cron -a 'name=memstat job=free>/tmp/mem.txt special_time=reboot user=root' -b
$ ansible node1 -m command -a 'crontab -l' -b
$ ansible node1 -m command -a 'tail /var/log/cron' -b
```

2. Ansible Playbooks:

We can use Ansible playbooks to schedule the future jobs in the managed nodes using at & cron modules.

AT Module:

```
$ vim at.yml
---
- name: Schedule Future Jobs
  hosts: node1
  become: true
  tasks:
    - name: Schedule a command to execute in 1 minutes as root
      ansible.posix.at:
         command: lsblk > /tmp/lsblk.txt
         count: 1
         units: minutes
...

$ ansible-playbook at.yml
$ ansible node1 -m command -a 'atq' -b
$ ansible node1 -m command -a 'ls -lh /tmp/lsblk.txt' -b
$ ansible node1 -m command -a 'cat /tmp/lsblk.txt' -b
```

```
$ vim at2.yml
---
- name: Schedule Future Jobs
  hosts: node1
  become: true
  tasks:
    - name: Schedule a command to execute in 3 days as root
      ansible.posix.at:
         command: uptime > /tmp/uptime.txt
         count: 3
         units: days
...

$ ansible-playbook at2.yml
$ ansible node1 -m command -a 'atq' -b
$ ansible node1 -m command -a 'at -c 3' -b
$ vim at3.yml
---
- name: Schedule Future Jobs
  hosts: node1
  become: true
  tasks:
    - name: Remove an existing task
      ansible.posix.at:
         command: uptime > /tmp/uptime.txt
         state: absent
...

$ ansible-playbook at3.yml
$ ansible node1 -m command -a 'atq' -b

$ vim at4.yml
---
- name: Schedule Future Jobs
  hosts: node1
  become: true
  tasks:
    - name: Schedule a command to execute in 20 minutes making sure it is unique in the queue
      ansible.posix.at:
        command: ls -d / >/dev/null
        count: 20
        units: minutes
        unique: true
...

$ ansible-playbook at4.yml
$ ansible node1 -m command -a 'atq' -b
$ ansible node1 -m command -a 'at -c 4' -b
```

CRON Module:

```
$ vim cron.yml
---
- name: Schedule Future Jobs
  hosts: node1
  become: true
  tasks:
    - name: Capture the users login information in /tmp/login.txt file at 2 & 5 hours.
      ansible.builtin.cron:
        name: "Users Login"
        minute: "0"
        hour: "2,5"
        job: "who > /tmp/login.txt"
...
```

```
$ ansible-playbook cron.yml
$ ansible node1 -m command -a 'crontab -l' -b

$ vim cron2.yml
---
- name: Schedule Future Jobs
  hosts: node1
  become: true
  tasks:
    - name: Remove an old job from crontab
      ansible.builtin.cron:
        name: "Users Login"
        state: absent
...

$ ansible-playbook cron2.yml
$ ansible node1 -m command -a 'crontab -l' -b

$ vim cron3.yml
---
- name: Schedule Future Jobs
  hosts: node1
  become: true
  tasks:
    - name: Show who is logged on and what they are doing at system reboot.
      ansible.builtin.cron:
        name: "System Info"
        special_time: reboot
        job: "w > /tmp/systeminfo.txt"
...

$ ansible-playbook cron3.yml
$ ansible node1 -m command -a 'crontab -l' -b

$ vim cron4.yml
---
- name: Schedule Future Jobs
  hosts: node1
  become: true
  tasks:
    - name: Creates a cron file under /etc/cron.d
      ansible.builtin.cron:
        name: yum autoupdate
        weekday: "2"
        minute: "0"
        hour: "12"
        user: root
        job: "YUMINTERACTIVE=0 /usr/sbin/yum-autoupdate"
        cron_file: ansible_yum-autoupdate
...
```

```
$ ansible-playbook cron4.yml
$ ansible node1 -m command -a 'ls -lh /etc/cron.d/' -b
$ ansible node1 -m command -a 'cat /etc/cron.d/ansible_yum-autoupdate' -b

$ vim cron5.yml
---
- name: Schedule Future Jobs
  hosts: node1
  become: true
  tasks:
    - name: Removes a cron file from under /etc/cron.d
      ansible.builtin.cron:
        name: yum autoupdate
        cron_file: ansible_yum-autoupdate
        state: absent
...

$ ansible-playbook cron5.yml
$ ansible node1 -m command -a 'ls -lh /etc/cron.d/' -b
$ ansible node1 -m command -a 'cat /etc/cron.d/ansible_yum-autoupdate' -b
```

Thank You