Red Hat Enterprise Linux Automation with Ansible

Guided Exercise: Managing the Boot Process and Scheduled Processes ☆

In this exercise, you will manage the startup process, schedule recurring jobs, and reboot managed hosts.

Outcomes

You should be able to use a playbook to:

Schedule a `cron` job.

Remove a single specific `cron` job from a `crontab` file.

Schedule an `at` task.

Set the default boot target on managed hosts.

Reboot managed hosts.

Run the `lab system-process start` script from `workstation` to configure the environment for the exercise. The script creates the `system-process` working directory, and downloads the Ansible configuration file and the host inventory file needed for the exercise.

```
[student@workstation ~]$ lab system-process start
```

Procedure 9.3. Instructions

As the `student` user on `workstation`, change to the `/home/student/system-process` working directory.

```
[student@workstation ~]$ cd ~/system-process
[student@workstation system-process]$
```

Create a playbook, `create_crontab_file.yml`, in the current working directory. Configure the playbook to use the `cron` module to create the `/etc/cron.d/add-date-time` crontab file that schedules a recurring cron job. The job should run as the `devops` user every two minutes between `09:00` and `16:59` on `Monday` through `Friday`. The job should append the current date and time to the file /home/devops/my_datetime_cron_job

2.1. Create a new playbook, `create_crontab_file.yml`, and add the lines needed to start the play. It should target the managed hosts in the `webservers` group and enable privilege escalation.

```
---
- name: Recurring cron job
  hosts: webservers
  become: true
```

2.2. Define a task that uses the `cron` module to schedule a recurring cron job.

> **NOTE**
>
> The `cron` module provides a `name` option to uniquely describe the crontab file entry and to ensure expected results. The description is added to the crontab file. For example, the `name` option is required if you are removing a crontab entry using `state=absent`. Additionally, the `name` option prevents a new crontab entry from always being created when the default state, `state=present`, is set.

```
  tasks:
    - name: Crontab file exists
      cron:
        name: Add date and time to a file
```

2.3. Configure the job to run every two minutes between `09:00` and `16:59` on `Monday` through `Friday`.

```
        minute: "*/2"
        hour: 9-16
        weekday: 1-5
```

2.4. Use the `cron_file` parameter to use the `/etc/cron.d/add-date-time` crontab file instead of an individual user's crontab in `/var/spool/cron/`. A relative path will place the file in `/etc/cron.d` directory. If the `cron_file` parameter is used, you must also specify the `user` parameter.

```
      user: devops
      job: date >> /home/devops/my_date_time_cron_job
      cron_file: add-date-time
      state: present
```

2.5. When completed, the playbook should appear as follows. Review the playbook for accuracy.

```
---
- name: Recurring cron job
  hosts: webservers
  become: true

  tasks:
    - name: Crontab file exists
      cron:
        name: Add date and time to a file
        minute: "*/2"
        hour: 9-16
        weekday: 1-5
        user: devops
        job: date >> /home/devops/my_date_time_cron_job
        cron_file: add-date-time
        state: present
```

2.6. Verify playbook syntax by running the `ansible-playbook --syntax-check create_crontab_file.yml` command. Correct any errors before moving to the next step.

```
[student@workstation system-process]$ ansible-playbook --syntax-check \
> create_crontab_file.yml

playbook: create_crontab_file.yml
```

2.7. Run the playbook.

```
[student@workstation system-process]$ ansible-playbook create_crontab_file.yml

PLAY [Recurring cron job] ******************************************************

TASK [Gathering Facts] *********************************************************
ok: [servera.lab.example.com]

TASK [Crontab file exists] *****************************************************
changed: [servera.lab.example.com]

PLAY RECAP *********************************************************************
servera.lab.example.com    : ok=2    changed=1    unreachable=0    failed=0
```

2.8. Run an ad hoc command to verify that the `/etc/cron.d/add-date-time` cron file exists and its content is correct.

```
[student@workstation system-process]$ ansible webservers -u devops -b \
> -a "cat /etc/cron.d/add-date-time"
servera.lab.example.com | CHANGED | rc=0 >>
#Ansible: Add date and time to a file
*/2 9-16 * * 1-5 devops date >> /home/devops/my_date_time_cron_job
```

Create a playbook, `remove_cron_job.yml`, in the current working directory. Configure the playbook to use the cron module to remove the `Add date and time to a file` cron job from the `/etc/cron.d/add-date-time` crontab file

3.1. Create a new playbook, `remove_cron_job.yml`, and add the following lines:

```
---
- name: Remove scheduled cron job
  hosts: webservers
  become: true

  tasks:
    - name: Cron job removed
      cron:
        name: Add date and time to a file
        user: devops
        cron_file: add-date-time
        state: absent
```

3.2. Verify playbook syntax by running the `ansible-playbook --syntax-check remove_cron_job.yml` command. Correct any errors before moving to the next step.

```
[student@workstation system-process]$ ansible-playbook --syntax-check \
> remove_cron_job.yml

playbook: remove_cron_job.yml
```

3.3. Run the playbook.

```
[student@workstation system-process]$ ansible-playbook remove_cron_job.yml

PLAY [Remove scheduled cron job] ***************************************

TASK [Gathering Facts] *************************************************
ok: [servera.lab.example.com]

TASK [Cron job removed] ************************************************
changed: [servera.lab.example.com]

PLAY RECAP *************************************************************
servera.lab.example.com    : ok=2    changed=1    unreachable=0    failed=0
```

3.4. Run an ad hoc command to verify that the `/etc/cron.d/add-date-time` cron file continues to exist but the cron job has been removed.

```
[student@workstation system-process]$ ansible webservers -u devops -b \
> -a "cat /etc/cron.d/add-date-time"
servera.lab.example.com | CHANGED | rc=0 >>
```

Create a playbook, `schedule_at_task.yml`, in the current working directory. Configure the playbook to use the `at` module to schedule a task that runs one minute in the future. The task should run the `date` command and redirect its output to the `/home/devops/my_at_date_time` file. Use the `unique: yes` option to ensure that if the command already exists in the `at` queue, a new task is not added.

4.1. Create a new playbook, `schedule_at_task.yml`, and add the following lines:

```
---
- name: Schedule at task
  hosts: webservers
  become: true
  become_user: devops

  tasks:
    - name: Create date and time file
      at:
        command: "date > ~/my_at_date_time\n"
        count: 1
        units: minutes
        unique: yes
        state: present
```

4.2. Verify playbook syntax by running the `ansible-playbook -syntax-check schedule_at_task.yml` command. Correct any errors before moving to the next step.

```
[student@workstation system-process]$ ansible-playbook --syntax-check \
> schedule_at_task.yml

playbook: schedule_at_task.yml
```

4.3. Run the playbook.

```
[student@workstation system-process]$ ansible-playbook schedule_at_task.yml

PLAY [Schedule at task] ************************************************

TASK [Gathering Facts] *************************************************
ok: [servera.lab.example.com]

TASK [Create date and time file] ***************************************
changed: [servera.lab.example.com]

PLAY RECAP *************************************************************
servera.lab.example.com    : ok=2    changed=1    unreachable=0    failed=0
```

4.4. After waiting one minute for the `at` command to complete, run ad hoc commands to verify that the `/home/devops/my_at_date_time` file exists and has the correct contents.

```
[student@workstation system-process]$ ansible webservers -u devops \
> -a "ls -l my_at_date_time"
servera.lab.example.com | CHANGED | rc=0 >>
-rw-rw-r--. 1 devops devops 30 abr 17 06:15 my_at_date_time

[student@workstation system-process]$ ansible webservers -u devops \
> -a "cat my_at_date_time"
servera.lab.example.com | CHANGED | rc=0 >>
Thu Jul 22 13:24:34 PDT 2021
```

Create a playbook, `set_default_boot_target_graphical.yml`, in the current working directory. Configure the playbook to use the `file` module to change the symbolic link on managed hosts to reference the `graphical-target` boot target.

> **NOTE**
>
> In the following `file` module, the `src` parameter value is what the symbolic link references. The `dest` parameter value is the symbolic link.

5.1. Create a new playbook, `set_default_boot_target_graphical.yml`, and add the following lines:

```
---
- name: Change default boot target
  hosts: webservers
  become: true

  tasks:
    - name: Default boot target is graphical
      file:
        src: /usr/lib/systemd/system/graphical.target
        dest: /etc/systemd/system/default.target
        state: link
```

5.2. Verify the playbook syntax by running the `ansible-playbook --syntax-check set_default_boot_target_graphical.yml` command. Correct any errors before moving to the next step.

```
[student@workstation system-process]$ ansible-playbook --syntax-check \
> set_default_boot_target_graphical.yml

playbook: set_default_boot_target_graphical.yml
```

5.3. Before running the playbook, run an ad hoc command to verify that the current default boot target is `multi-user.target`:

```
[student@workstation system-process]$ ansible webservers -u devops -b \
> -a "systemctl get-default"
servera.lab.example.com | CHANGED | rc=0 >>
multi-user.target
```

5.4. Run the playbook.

```
[student@workstation system-process]$ ansible-playbook \
> set_default_boot_target_graphical.yml

PLAY [Change default boot target] ************************************

TASK [Gathering Facts] ***********************************************
ok: [servera.lab.example.com]

TASK [Default boot target is graphical] ******************************
changed: [servera.lab.example.com]

PLAY RECAP ***********************************************************
servera.lab.example.com    : ok=2    changed=1    unreachable=0    failed=0
```

5.5. Run an ad hoc command to verify that the default boot target is now `graphical.target`.

```
[student@workstation system-process]$ ansible webservers -u devops -b \
> -a "systemctl get-default"
servera.lab.example.com | CHANGED | rc=0 >>
graphical.target
```

Create a playbook, `reboot_hosts.yml`, in the current working directory that reboots the managed hosts. It is not required to reboot a server after changing the default target. However, knowing how to create a playbook that reboots managed hosts may prove useful.

6.1. Create a new playbook, `reboot_hosts.yml`, and add the following lines:

```
---
- name: Reboot hosts
  hosts: webservers
  become: true

  tasks:
    - name: Hosts are rebooted
      reboot:
```

6.2. Verify the playbook syntax by running the `ansible-playbook --syntax-check reboot_hosts.yml` command. Correct any errors before moving to the next step.

```
[student@workstation system-process]$ ansible-playbook --syntax-check \
> reboot_hosts.yml

playbook: reboot_hosts.yml
```

6.3. Before running the playbook, run an ad hoc command to determine the timestamp of the last system reboot.

```
[student@workstation system-process]$ ansible webservers -u devops -b \
> -a "who -b"
servera.lab.example.com | CHANGED | rc=0 >>
         system boot  2021-07-22 14:34
```

6.4. Run the playbook.

```
[student@workstation system-process]$ ansible-playbook reboot_hosts.yml

PLAY [Reboot hosts] ************************************************

TASK [Gathering Facts] *********************************************
ok: [servera.lab.example.com]

TASK [Hosts are rebooted] ******************************************
changed: [servera.lab.example.com]

PLAY RECAP *********************************************************
servera.lab.example.com    : ok=2    changed=1    unreachable=0    failed=0
```

6.5. Run an ad hoc command to determine the timestamp of the last system reboot. The timestamp displayed after the playbook runs should be later.

```
[student@workstation system-process]$ ansible webservers -u devops -b \
> -a "who -b"
servera.lab.example.com | CHANGED | rc=0 >>
         system boot  2021-07-22 14:52
```

6.6. Run a second ad hoc command to determine that the `graphical.target` boot target survived the reboot.

```
[student@workstation system-process]$ ansible webservers -u devops -b \
> -a "systemctl get-default"
servera.lab.example.com | CHANGED | rc=0 >>
graphical.target
```

To maintain consistency throughout the remaining exercises, change the default boot target back to its former setting, `multi-user.target`. Create a playbook, `set_default_boot_target_multi-user.yml`, in the current working directory. Configure the playbook to use the `file` module to change the symbolic link on managed hosts to reference the `multi-user.target` boot target.

7.1. Create a new playbook, `set_default_boot_target_multi-user.yml`, and add the following lines:

```
---
- name: Change default runlevel target
  hosts: webservers
  become: true

  tasks:
    - name: Default runlevel is multi-user target
      file:
        src: /usr/lib/systemd/system/multi-user.target
        dest: /etc/systemd/system/default.target
        state: link
```

7.2. Verify playbook syntax by running the `ansible-playbook --syntax-check set_default_boot_target_multi-user.yml` command. Correct any errors before moving to the next step.

```
[student@workstation system-process]$ ansible-playbook --syntax-check \
> set_default_boot_target_multi-user.yml

playbook: set_default_boot_target_multi-user.yml
```

7.3. Run the playbook.

```
[student@workstation system-process]$ ansible-playbook \
> set_default_boot_target_multi-user.yml

PLAY [Change default runlevel target] ************************************

TASK [Gathering Facts] ***************************************************
ok: [servera.lab.example.com]

TASK [Default runlevel is multi-user target] *****************************
changed: [servera.lab.example.com]

PLAY RECAP ***************************************************************
servera.lab.example.com    : ok=2    changed=1    unreachable=0    failed=0
```

7.4. Run an ad hoc command to verify that the default boot target is now `multi-user.target`.

```
[student@workstation system-process]$ ansible webservers -u devops -b \
> -a "systemctl get-default"
servera.lab.example.com | CHANGED | rc=0 >>
multi-user.target
```

Finish

On `workstation`, run the `lab system-process finish` script to clean up this exercise.

```
[student@workstation ~]$ lab system-process finish
```

This concludes the guided exercise.

← PREVIOUS (/ROL/APP/COURSES/RH294-8.4/PAGES/CH09S05)

→ NEXT (/ROL/APP/COURSES/RH294-8.4/PAGES/CH09S07)