Red Hat Enterprise Linux Automation with Ansible

Lab: Automating Linux Administration Tasks ☆

Performance Checklist

In this lab, you will configure and perform administrative tasks on managed hosts using a playbook.

Outcomes

You should be able to create playbooks for configuring on a managed host a software repository, users and groups, logical volumes, cron jobs, and additional network interfaces.

On `workstation`, run the lab start script to confirm that the environment is ready for the lab to begin. The script creates the working directory, called `system-review`, and populates it with an Ansible configuration file, a host inventory, and lab files.

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

Procedure 9.6. Instructions

Create and execute on the `webservers` host group a playbook which configures the Yum internal repository located at `http://materials.example.com/yum/repository`, and installs the example-motd package available in that repository. All RPM packages are signed with an organizational GPG key pair. The GPG public key is available at `http://materials.example.com/yum/repository/RPM-GPG-KEY-example`.

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

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

1.2. Create the `repo_playbook.yml` playbook which runs on the managed hosts at the `webservers` host group. Add a task that uses the `yum_repository` module to ensure the configuration of the internal yum repository on the remote host. Ensure that:

- The repository's configuration is stored in the file `/etc/yum.repos.d/example.repo`
- The repository ID is `example-internal`
- The base URL is `http://materials.example.com/yum/repository`
- The repository is configured to check RPM GPG signatures
- The repository description is `Example Inc. Internal YUM repo`

The playbook contains the following:

```
---
- name: Repository Configuration
  hosts: webservers
  tasks:
    - name: Ensure Example Repo exists
      yum_repository:
        name: example-internal
        description: Example Inc. Internal YUM repo
        file: example
        baseurl: http://materials.example.com/yum/repository/
        gpgcheck: yes
```

1.3. Add a second task to the play that uses the `rpm_key` module to ensure that the repository public key is present on the remote host. The repository public key URL is `http://materials.example.com/yum/repository/RPM-GPG-KEY-example`.

The second task appears as follows:

```
    - name: Ensure Repo RPM Key is Installed
      rpm_key:
        key: http://materials.example.com/yum/repository/RPM-GPG-KEY-example
        state: present
```

1.4. Add a third task to install the example-motd package available in the Yum internal repository.

The third task appears as follows:

```
    - name: Install Example motd package
      yum:
        name: example-motd
        state: present
```

1.5. Execute the playbook:

```
[student@workstation system-review]$ ansible-playbook repo_playbook.yml

PLAY [Repository Configuration] ************************************************

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

TASK [Ensure Example Repo exists] **********************************************
changed: [serverb.lab.example.com]

TASK [Ensure Repo RPM Key is Installed] ****************************************
changed: [serverb.lab.example.com]

TASK [Install Example motd package] ********************************************
changed: [serverb.lab.example.com]

PLAY RECAP *********************************************************************
serverb.lab.example.com    : ok=4    changed=3    unreachable=0    failed=0
```

HIDE SOLUTION

Create and execute on the `webservers` host group a playbook which creates the `webadmin` user group, and add two users to that group, `ops1` and `ops2`.

2.1. Create a `vars/users_vars.yml` variable file, which defines two users, `ops1` and `ops2`, which belong to the `webadmin` user group. You may need to create the `vars` subdirectory.

```
[student@workstation system-review]$ mkdir vars
[student@workstation system-review]$ vi vars/users_vars.yml
---
users:
  - username: ops1
    groups: webadmin
  - username: ops2
    groups: webadmin
```

2.2. Create the `users.yml` playbook. Define a single play in the playbook that targets the `webservers` host group. Add a `vars_files` clause that defines the location of the `vars/users_vars.yml` filename. Add a task which uses the `group` module to create the `webadmin` user group on the remote host.

```
---
- name: Create multiple local users
  hosts: webservers
  vars_files:
    - vars/users_vars.yml
  tasks:
    - name: Add webadmin group
      group:
        name: webadmin
        state: present
```

2.3. Add a second task to the playbook that uses the `user` module to create the users. Add a `loop: "{{ users }}"` clause to the task to loop through the variable file for every username found in the `vars/users_vars.yml` file. As the `name:` for the users, use the `item.username` the variable name. This way the variable file may contain additional information that might be useful for creating the users, such as the groups that the users should belong to. The second task contains the following:

```
    - name: Create user accounts
      user:
        name: "{{ item.username }}"
        groups: webadmin
      loop: "{{ users }}"
```

2.4. Execute the playbook:

```
[student@workstation system-review]$ ansible-playbook users.yml

PLAY [Create multiple local users] ***************************************

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

TASK [Add webadmin group] ************************************************
changed: [serverb.lab.example.com]

TASK [Create user accounts] **********************************************
changed: [serverb.lab.example.com] => (item={'username': 'ops1', 'groups': 'webadmin'})
changed: [serverb.lab.example.com] => (item={'username': 'ops2', 'groups': 'webadmin'})

PLAY RECAP ***************************************************************
serverb.lab.example.com    : ok=3    changed=2    unreachable=0    failed=0
```

HIDE SOLUTION

Create and execute on the `webservers` host group a playbook that uses the `/dev/vdb` device to create a volume group named `apache-vg`. This playbook also creates two logical volumes, named `content-lv` and `logs-lv`, both backed by the `apache-vg` volume group. Finally, it creates an XFS file system on each logical volume, and mounts the `content-lv` logical volume at `/var/www`, and the `logs-lv` logical volume at `/var/log/httpd`. The lab script populates two files in `~/system-review`, `storage.yml` which provides an initial skeleton for the playbook, and `storage_vars.yml` which provides values to all the variables required by the different modules.

3.1. Review the `storage_vars.yml` variables file.

```
[student@workstation system-review]$ cat storage_vars.yml
---

partitions:
  - number: 1
    start: 1MiB
    end: 257MiB

volume_groups:
  - name: apache-vg
    devices: /dev/vdb1

logical_volumes:
  - name: content-lv
    size: 64M
    vgroup: apache-vg
    mount_path: /var/www

  - name: logs-lv
    size: 128M
    vgroup: apache-vg
    mount_path: /var/log/httpd
```

This file describes the intended structure of partitions, volume groups, and logical volumes on each web server. The first partition begins at an offset of 1 MiB from the beginning of the `/dev/vdb` device, and ends at an offset of 257 MiB, for a total size of 256 MiB.

Each web server has one volume group, named `apache-vg`, containing the first partition of the `/dev/vdb` device.

Each web server has two logical volumes. The first logical volume is named `content-lv`, with a size of 64 MiB, attached to the `apache-vg` volume group, and mounted at `/var/www`. The second logical volume is named `content-lv`, with a size of 128 MiB, attached to the `apache-vg` volume group, and mounted at `/var/log/httpd`.

> **NOTE**
>
> The `apache-vg` volume group has a capacity of 256 MiB, because it is backed by the `/dev/vdb1` partition. It provides enough capacity for both of the logical volumes.

3.2. Change the first task in the `storage.yml` playbook to use the `parted` module to configure a partition for each loop item. Each item describes an intended partition of the `/dev/vdb` device on each web server:

number
The partition number. Use this as the value of the `number` keyword for the `parted` module.

start
The start of the partition, as an offset from the beginning of the block device. Use this as the value of the `part_start` keyword for the `parted` module.

end
The end of the partition, as an offset from the beginning of the block device. Use this as the value of the `part_end` keyword for the `parted` module.

The content of the first task should be:

```
- name: Correct partitions exist on /dev/vdb
  parted:
    device: /dev/vdb
    state: present
    number: "{{ item.number }}"
    part_start: "{{ item.start }}"
    part_end: "{{ item.end }}"
  loop: "{{ partitions }}"
```

3.3. Change the second task of the play to use the `lvg` module to configure a volume group for each loop item. Each item of the `volume_groups` variable describes a volume group that should exist on each web server:

name
The name of the volume group. Use this as the value of the `vg` keyword for the `lvg` module.

devices
A comma-separated list of devices or partitions that form the volume group. Use this as the value of the `pvs` keyword for the `lvg` module.

The content of the second task should be:

```
- name: Ensure Volume Groups Exist
  lvg:
    vg: "{{ item.name }}"
    pvs: "{{ item.devices }}"
  loop: "{{  volume_groups }}"
```

3.4. Change the third task to use the `lvol` module. Set the volume group name, logical volume name, and logical volume size using each item's keywords. The content of the third task is now:

```
- name: Create each Logical Volume (LV) if needed
  lvol:
    vg: "{{ item.vgroup }}"
    lv: "{{ item.name }}"
    size: "{{ item.size }}"
  loop: "{{ logical_volumes }}"
```

3.5. Change the fourth task to use the `filesystem` module. Configure the task to ensure that each logical volume is formatted as an XFS file system. Recall that a logical volume is associated with the logical device `/dev/<volume group name>/<logical volume name>`.

The content of the fourth task should be:

```
- name: Ensure XFS Filesystem exists on each LV
  filesystem:
    dev: "/dev/{{ item.vgroup }}/{{ item.name }}"
    fstype: xfs
  loop: "{{ logical_volumes }}"
```

3.6. Configure the fifth task to ensure each logical volume has the correct storage capacity. If the logical volume increases in capacity, be sure to force the expansion of the volume's file system.

> **WARNING**
>
> If a logical volume needs to decrease in capacity, this task will fail because an XFS file system does not support shrinking capacity.

The content of the fifth task should be:

```
- name: Ensure the correct capacity for each LV
  lvol:
    vg: "{{ item.vgroup }}"
    lv: "{{ item.name }}"
    size: "{{ item.size }}"
    resizefs: yes
    force: yes
  loop: "{{ logical_volumes }}"
```

3.7. Use the `mount` module in the sixth task to ensure that each logical volume is mounted at the corresponding mount path and persists after a reboot.

The content of the sixth task should be:

```
  - name: Each Logical Volume is mounted
    mount:
      path: "{{ item.mount_path }}"
      src: "/dev/{{ item.vgroup }}/{{ item.name }}"
      fstype: xfs
      state: mounted
    loop: "{{ logical_volumes }}"
```

3.8. Execute the playbook to create the logical volumes on the remote host.

```
[student@workstation system-review]$ ansible-playbook storage.yml
PLAY [Ensure Apache Storage Configuration] ************************************

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

TASK [Correct partitions exist on /dev/vdb] ***********************************
changed: [serverb.lab.example.com] => (item={'number': 1, 'start': '1MiB', 'end': '257MiB'})

TASK [Ensure Volume Groups Exist] *********************************************
changed: [serverb.lab.example.com] => (item={'name': 'apache-vg', 'devices': '/dev/vdb1'})
...output omitted...

TASK [Create each Logical Volume (LV) if needed] ******************************
changed: [serverb.lab.example.com] => (item={'name': 'content-lv', 'size': '64M', 'vgroup': 'apache-vg', 'mount_path': '/var/www'})
changed: [serverb.lab.example.com] => (item={'name': 'logs-lv', 'size': '128M', 'vgroup': 'apache-vg', 'mount_path': '/var/log/http
d'})

TASK [Ensure XFS Filesystem exists on each LV] *********************************
changed: [serverb.lab.example.com] => (item={'name': 'content-lv', 'size': '64M', 'vgroup': 'apache-vg', 'mount_path': '/var/www'})
changed: [serverb.lab.example.com] => (item={'name': 'logs-lv', 'size': '128M', 'vgroup': 'apache-vg', 'mount_path': '/var/log/http
d'})

TASK [Ensure the correct capacity for each LV] *********************************
ok: [serverb.lab.example.com] => (item={'name': 'content-lv', 'size': '64M', 'vgroup': 'apache-vg', 'mount_path': '/var/www'})
ok: [serverb.lab.example.com] => (item={'name': 'logs-lv', 'size': '128M', 'vgroup': 'apache-vg', 'mount_path': '/var/log/httpd'})

TASK [Each Logical Volume is mounted] *****************************************
changed: [serverb.lab.example.com] => (item={'name': 'content-lv', 'size': '64M', 'vgroup': 'apache-vg', 'mount_path': '/var/www'})
changed: [serverb.lab.example.com] => (item={'name': 'logs-lv', 'size': '128M', 'vgroup': 'apache-vg', 'mount_path': '/var/log/http
d'})

PLAY RECAP ********************************************************************
serverb.lab.example.com    : ok=7    changed=5    unreachable=0    failed=0
```

HIDE SOLUTION

Create and execute on the `webservers` host group a playbook which uses the `cron` module to create the `/etc/cron.d/disk_usage` 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 disk usage to the file `/home/devops/disk_usage`.

4.1. 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
```

4.2. 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, when the default state, `state=present` is set, the `name` option prevents a new crontab entry from always being created, regardless of existing ones.

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

4.3. 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
```

4.4. Use the `cron_file` parameter to use the `/etc/cron.d/disk_usage` 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: df >> /home/devops/disk_usage
        cron_file: disk_usage
        state: present
```

4.5. 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: df >> /home/devops/disk_usage
        cron_file: disk_usage
        state: present
```

4.6. Run the playbook.

```
[student@workstation system-review]$ ansible-playbook create_crontab_file.yml
PLAY [Recurring cron job] ******************************************************

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

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

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

HIDE SOLUTION

Create and execute on the `webservers` host group a playbook which uses the `linux-system-roles.network` role to configure with the `172.25.250.40/24` IP address the spare network interface, `eth1`.

5.1. Use `ansible-galaxy` to verify that system roles are available. If not, you need to install the rhel-system-roles package.

```
[student@workstation system-review]$ ansible-galaxy list
# /usr/share/ansible/roles
- linux-system-roles.kdump, (unknown version)
- linux-system-roles.network, (unknown version)
- linux-system-roles.postfix, (unknown version)
- linux-system-roles.selinux, (unknown version)
- linux-system-roles.timesync, (unknown version)
- rhel-system-roles.kdump, (unknown version)
- rhel-system-roles.network, (unknown version)
- rhel-system-roles.postfix, (unknown version)
- rhel-system-roles.selinux, (unknown version)
- rhel-system-roles.timesync, (unknown version)
# /etc/ansible/roles
 [WARNING]: - the configured path /home/student/.ansible/roles does not exist.
```

5.2. Create a playbook, `network_playbook.yml`, with one play that targets the `webservers` host group. Include the `rhel-system-roles.network` role in the `roles` section of the play.

```
---
- name: NIC Configuration
  hosts: webservers

  roles:
    - rhel-system-roles.network
```

5.3. Create the `group_vars/webservers` subdirectory.

```
[student@workstation system-review]$ mkdir -pv group_vars/webservers
mkdir: created directory 'group_vars'
mkdir: created directory 'group_vars/webservers'
```

5.4. Create a new file `network.yml` to define role variables. Because these variable values apply to the hosts on the `webservers` host group, you need to create that file in the `group_vars/webservers` directory. Add variable definitions to support the configuration of the `eth1` network interface. The file now contains:

```
[student@workstation system-review]$ vi group_vars/webservers/network.yml
---
network_connections:
  - name: eth1
    type: ethernet
    ip:
      address:
        - 172.25.250.40/24
```

5.5. Run the playbook to configure the secondary network interface.

```
[student@workstation system-review]$ ansible-playbook network_playbook.yml

PLAY [NIC Configuration] ********************************************************

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

TASK [rhel-system-roles.network : Check which services are running] *************
ok: [serverb.lab.example.com]

TASK [rhel-system-roles.network : Check which packages are installed] **********
ok: [serverb.lab.example.com]

TASK [rhel-system-roles.network : Print network provider] ***********************
ok: [serverb.lab.example.com] => {
    "msg": "Using network provider: nm"
}

TASK [rhel-system-roles.network : Install packages] *****************************
skipping: [serverb.lab.example.com]

TASK [rhel-system-roles.network : Enable network service] ***********************
ok: [serverb.lab.example.com]

TASK [rhel-system-roles.network : Configure networking connection profiles] ****
 [WARNING]: [002] <info>  #0, state:None persistent_state:present, 'eth1': add connection
eth1, 38d63afd-e610-4929-ba1b-1d38413219fb

changed: [serverb.lab.example.com]

TASK [rhel-system-roles.network : Re-test connectivity] *************************
ok: [serverb.lab.example.com]

PLAY RECAP **********************************************************************
serverb.lab.example.com    : ok=7    changed=1    unreachable=0    failed=0
```

5.6. Verify that the `eth1` network interface uses the `172.25.250.40` IP address. It may take up to a minute to configure the IP address.

```
[student@workstation system-review]$ ansible webservers -m setup \
> -a 'filter=ansible_eth1'
serverb.lab.example.com | SUCCESS => {
    "ansible_facts": {
        "ansible_eth1": {
...output omitted...
            "ipv4": {
                "address": "172.25.250.40",
                "broadcast": "172.25.250.255",
                "netmask": "255.255.255.0",
                "network": "172.25.250.0"
            },
...output omitted...
```

HIDE SOLUTION

Evaluation

Run `lab system-review grade` on `workstation` to grade your work.

```
[student@workstation ~]$ lab system-review grade
```

Finish

From `workstation`, run the `lab system-review finish` script to clean up the resources created in this lab.

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

This concludes the lab.

← PREVIOUS (/ROL/APP/COURSES/RH294-8.4/PAGES/CH09S10)

→ NEXT (/ROL/APP/COURSES/RH294-8.4/PAGES/CH09S12)