Red Hat Enterprise Linux Automation with Ansible

← PREVIOUS (/ROL/APP/COURSES/RH294-8.4/PAGES/CH09S07)

→ NEXT (/ROL/APP/COURSES/RH294-8.4/PAGES/CH09S09)

Guided Exercise: Managing Storage

In this exercise you will partition a new disk, create logical volumes and format them with XFS file systems, and mount them immediately and automatically at boot time on your managed hosts.

Outcomes

You should be able to:

Use the `parted` module to configure block device partitions.

Use the `lvg` module to manage LVM volume groups.

Use the `lvol` module to manage LVM logical volumes.

Use the `filesystem` module to create file systems.

Use the `mount` module to control and configure mount points in `/etc/fstab`.

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

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

Procedure 9.4. Instructions

You are responsible for managing a set of web servers. A recommended practice for web server configuration is to store web server data on a separate partition or logical volume.

You will write a playbook to:

Manage partitions of the `/dev/vdb` device

Manage a volume group named `apache-vg` for web server data

Create two logical volumes named `content-lv` and `logs-lv`, both backed by the `apache-vg` volume group

Create an XFS file system on both logical volumes

Mount the `content-lv` logical volume at `/var/www`

Mount the `logs-lv` logical volume at `/var/log/httpd`

If the storage requirements for the web server change, update the appropriate playbook variables and re–execute the playbook. The playbook should be idempotent.

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

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

Review the skeleton playbook file `storage.yml` and the associated variables file `storage_vars.yml` in the project directory. Execute the playbook.

2.1. Review the `storage.yml` playbook.

```
---
- name: Ensure Apache Storage Configuration
  hosts: webservers
  vars_files:
    - storage_vars.yml
  tasks:
    - name: Correct partitions exist on /dev/vdb
      debug:
        msg: TODO
      loop: "{{ partitions }}"

    - name: Ensure Volume Groups Exist
      debug:
        msg: TODO
      loop: "{{  volume_groups }}"

    - name: Create each Logical Volume (LV) if needed
      debug:
        msg: TODO
      loop: "{{ logical_volumes }}"
      when: true

    - name: Ensure XFS Filesystem exists on each LV
      debug:
        msg: TODO
      loop: "{{ logical_volumes }}"

    - name: Ensure the correct capacity for each LV
      debug:
        msg: TODO
      loop: "{{ logical_volumes }}"

    - name: Each Logical Volume is mounted
      debug:
        msg: TODO
      loop: "{{ logical_volumes }}"
```

The name of each task acts as an outline of the intended procedure to implement. In later steps, you will update and change these six tasks.

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

```
---

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 `logs-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.

2.3. Execute the `storage.yml` playbook.

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

PLAY [Ensure Apache Storage Configuration] ***********************************

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

TASK [Correct partitions exist on /dev/vdb] **********************************
ok: [servera.lab.example.com] => (item={u'start': u'1MiB', u'end': u'257MiB', u'number': 1}) => {
    "msg": "TODO"
}

...output omitted...

TASK [Each Logical Volume is mounted] ****************************************
ok: [servera.lab.example.com] => (item={u'vgroup': u'apache-vg', u'size': u'64M', u'mount_path': u'/var/www', u'name': u'content-lv'}) =>
{
    "msg": "TODO"
}
ok: [servera.lab.example.com] => (item={u'vgroup': u'apache-vg', u'size': u'128M', u'mount_path': u'/var/log/httpd', u'name': u'logs-lv'})
=> {
    "msg": "TODO"
}

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

Change the first task 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 }}"
```

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 }}"
```

Change the third task of the play to use the `lvol` module to create a logical volume for each item. Use the item's keywords to create the new logical volume:

name
The name of the logical volume. Use this as the value of the `lv` keyword for the `lvol` module.

vgroup
The name of the volume group that provides storage for the logical volume.

size
The size of the logical volume. The value of this keyword is any acceptable value for the `-L` option of the `lvcreate` command.

Only execute the task if a logical volume does not already exist. Update the `when` statement to check that a logical volume does not exist with a name that matches the value of the item's `name` keyword.

5.1. 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 }}"
      when: true
```

5.2. The Ansible fact `ansible_lvm` contains information about Logical Volume Management objects on each hosts. Use an ad hoc command to see the current set of logical volumes on the remote host:

```
[student@workstation system-storage]$ ansible all -m setup -a \
> "filter=ansible_lvm"
servera.lab.example.com | SUCCESS => {
    "ansible_facts": {
        "ansible_lvm": {
            "lvs": {},
            "pvs": {},
            "vgs": {}
        },
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false
}
```

The value of the `lvs` keyword indicates that there are no logical volumes on the remote host.

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

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

PLAY [Ensure Apache Storage Configuration] ***********************************

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

TASK [Correct partitions exist on /dev/vdb] **********************************
changed: [servera.lab.example.com] => (item={...output omitted...})

TASK [Ensure Volume Groups Exist] ********************************************
changed: [servera.lab.example.com] => (item={...output omitted...})

TASK [Create each Logical Volume (LV) if needed] *****************************
changed: [servera.lab.example.com] => (item={...output omitted...})
changed: [servera.lab.example.com] => (item={...output omitted...})

TASK [Ensure XFS Filesystem exists on each LV] *******************************
ok: [servera.lab.example.com] => (item={...output omitted...}) => {
    "msg": "TODO"
}
...output omitted...
PLAY RECAP *******************************************************************
servera.lab.example.com    : ok=7    changed=3    unreachable=0    failed=0
```

5.4. Execute another ad hoc command to see the structure of the `ansible_lvm` variable when logical volumes exists on the remote host.

```
[student@workstation system-storage]$ ansible all -m setup -a \
> "filter=ansible_lvm"
servera.lab.example.com | SUCCESS => {
    "ansible_facts": {
        "ansible_lvm": {
            "lvs": {(1)
                "content-lv": {
                    "size_g": "0.06",
                    "vg": "apache-vg"
                },
                "logs-lv": {
                    "size_g": "0.12",
                    "vg": "apache-vg"
                }
            },
            "pvs": {(2)
                "/dev/vdb1": {
                    "free_g": "0.06",
                    "size_g": "0.25",
                    "vg": "apache-vg"
                }
            },
            "vgs": {(3)
                "apache-vg": {
                    "free_g": "0.06",
                    "num_lvs": "2",
                    "num_pvs": "1",
                    "size_g": "0.25"
                }
            }
        }
    },
    "changed": false
}
```

1. The value of the `lvs` keyword is a key–value pair data structure. The keys of this structure are the names of any logical volumes on the host. This indicates that both the `content-lv` and `logs-lv` logical volumes exist. For each logical volume, the corresponding volume group is provided by the `vg` keyword.
2. The `pvs` keyword contains information about physical volumes on the host. The information indicates that the `/dev/vdb1` partition belongs to the `apache-vg` volume group.
3. The `vgs` keyword contains information about volume groups on the host.

5.5. Update the `when` statement to check that a logical volume does not exist with a name that matches the value of the item's `name` keyword. 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 }}"
  when: item.name not in ansible_lvm["lvs"]
```

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 }}"
```

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 }}"
```

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
      opts: noatime
      state: mounted
    loop: "{{ logical_volumes }}"
```

Review the completed `storage.yml` playbook. Execute the playbook and verify that each logical volume is mounted.

9.1. Review the playbook:

```
---
- name: Ensure Apache Storage Configuration
  hosts: webservers
  vars_files:
    - storage_vars.yml
  tasks:
    - 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 }}"

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

    - name: Create each Logical Volume (LV) if needed
      lvol:
        vg: "{{ item.vgroup }}"
        lv: "{{ item.name }}"
        size: "{{ item.size }}"
      loop: "{{ logical_volumes }}"
      when: item.name not in ansible_lvm["lvs"]

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

    - 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 }}"

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

9.2. Execute the playbook.

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

PLAY [Ensure Apache Storage Configuration] **********************************

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

TASK [Correct partitions exist on /dev/vdb] *********************************
ok: [servera.lab.example.com] => (item={...output omitted...})

TASK [Ensure Volume Groups Exist] *******************************************
ok: [servera.lab.example.com] => (item={...output omitted...})
...output omitted...

TASK [Create each Logical Volume (LV) if needed] ****************************
skipping: [servera.lab.example.com] => (item={...output omitted...})
skipping: [servera.lab.example.com] => (item={...output omitted...})

TASK [Ensure XFS Filesystem exists on each LV] ******************************
changed: [servera.lab.example.com] => (item={...output omitted...})
changed: [servera.lab.example.com] => (item={...output omitted...})

TASK [Ensure the correct capacity for each LV] ******************************
ok: [servera.lab.example.com] => (item={...output omitted...})
ok: [servera.lab.example.com] => (item={...output omitted...})

TASK [Each Logical Volume is mounted] ***************************************
changed: [servera.lab.example.com] => (item={...output omitted...})
changed: [servera.lab.example.com] => (item={...output omitted...})

PLAY RECAP ******************************************************************
servera.lab.example.com    : ok=6    changed=2    unreachable=0    failed=0
skipped=1    rescued=0    ignored=0
```

A task is skipped during execution because the playbook was previously executed with the same variable values. The logical volumes did not need to be created.

9.3. Use an Ansible ad hoc command to run the `lsblk` command on the remote host. The output indicates the mount points for the logical volumes.

```
[student@workstation system-storage]$ ansible all -a lsblk
servera.lab.example.com | CHANGED | rc=0 >>
NAME                      MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sr0                        11:0    1 1024M  0 rom
vda                       252:0    0   10G  0 disk
└─vda1                    252:1    0   10G  0 part /
vdb                       252:16   0    1G  0 disk
└─vdb1                    252:17   0  256M  0 part
  ├─apache--vg-content--lv 253:0    0   64M  0 lvm  /var/www
  └─apache--vg-logs--lv    253:1    0  128M  0 lvm  /var/log/httpd
```

Increase the capacity of the `content-lv` logical volume to 128 MiB, and the `logs-lv` logical volume to 256 MiB. This requires increasing the capacity of the `apache-vg` volume group.

Create a new partition with a capacity of 256 MiB and add it to the `apache-vg` volume group.

10.1. Edit the `partitions` variable definition in the `storage_vars.yml` file to add a second partition to the `/dev/vdb` device. The content of the `partitions` variable should be:

```
partitions:
  - number: 1
    start: 1MiB
    end: 257MiB
  - number: 2
    start: 257MiB
    end: 513MiB
```

10.2. Edit the `volume_groups` variable definition in the `storage_vars.yml` file. Add the second partition to list of devices backing the volume group. The content of the `volume_groups` variable should be:

```
volume_groups:
  - name: apache-vg
    devices: /dev/vdb1,/dev/vdb2
```

10.3. Double the capacity of each logical volume defined in the `storage_vars.yml` file. The content of the `logical_volumes` variable should be:

```
logical_volumes:
  - name: content-lv
    size: 128M
    vgroup: apache-vg
    mount_path: /var/www

  - name: logs-lv
    size: 256M
    vgroup: apache-vg
    mount_path: /var/log/httpd
```

10.4. Execute the playbook. Verify the new capacity of each logical volume.

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

PLAY [Ensure Apache Storage Configuration] ***********************************

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

TASK [Correct partitions exist on /dev/vdb] **********************************
ok: [servera.lab.example.com] => (item={...output omitted...})
changed: [servera.lab.example.com] => (item={u'start': u'257MiB', u'end': u'513MiB', u'number': 2})

TASK [Ensure Volume Groups Exist] ********************************************
changed: [servera.lab.example.com] => (item={u'name': u'apache-vg', u'devices': u'/dev/vdb1,/dev/vdb2'})
...output omitted...

TASK [Create each Logical Volume (LV) if needed] *****************************
skipping: [servera.lab.example.com] => (item={u'vgroup': u'apache-vg', u'size': u'128M', u'mount_path': u'/var/www', u'name': u'content-l
v'})
skipping: [servera.lab.example.com] => (item={u'vgroup': u'apache-vg', u'size': u'256M', u'mount_path': u'/var/log/httpd', u'name': u'log
s-lv'})

TASK [Ensure XFS Filesystem exists on each LV] *******************************
ok: [servera.lab.example.com] => (item={...output omitted...})
ok: [servera.lab.example.com] => (item={...output omitted...})

TASK [Ensure the correct capacity for each LV] *******************************
changed: [servera.lab.example.com] => (item={u'vgroup': u'apache-vg', u'size': u'128M', u'mount_path': u'/var/www', u'name': u'content-l
v'})
changed: [servera.lab.example.com] => (item={u'vgroup': u'apache-vg', u'size': u'256M', u'mount_path': u'/var/log/httpd', u'name': u'logs
-lv'})

TASK [Each Logical Volume is mounted] ****************************************
ok: [servera.lab.example.com] => (item={...output omitted...})
ok: [servera.lab.example.com] => (item={...output omitted...})

PLAY RECAP *******************************************************************
servera.lab.example.com    : ok=6    changed=3    unreachable=0    failed=0
skipped=1    rescued=0    ignored=0
```

The output indicates changes to the partitions and volume group on the remote host, and that both logical volumes were resized.

10.5. Use an Ansible ad hoc command to run the `lsblk` command on the remote host.

```
[student@workstation system-storage]$ ansible all -a lsblk
servera.lab.example.com | CHANGED | rc=0 >>
NAME                      MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sr0                        11:0    1 1024M  0 rom
vda                       252:0    0   10G  0 disk
└─vda1                    252:1    0   10G  0 part /
vdb                       252:16   0    1G  0 disk
├─vdb1                    252:17   0  256M  0 part
│ ├─apache--vg-content--lv 253:0    0  128M  0 lvm  /var/www
│ └─apache--vg-logs--lv    253:1    0  256M  0 lvm  /var/log/httpd
└─vdb2                    252:18   0  256M  0 part
  ├─apache--vg-content--lv 253:0    0  128M  0 lvm  /var/www
  └─apache--vg-logs--lv    253:1    0  256M  0 lvm  /var/log/httpd
```

The output indicates that each logical volume is the correct size and mounted at the correct directory. Two entries exists for each logical volume because files stored on the logical volume may be physically located on either partition (`/dev/vdb1` or `/dev/vdb2`).

Finish

Run the `lab system-storage finish` command to cleanup the managed host.

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

This concludes the guided exercise.

← PREVIOUS (/ROL/APP/COURSES/RH294-8.4/PAGES/CH09S07) → NEXT (/ROL/APP/COURSES/RH294-8.4/PAGES/CH09S09)