Red Hat Enterprise Linux Automation with Ansible

Managing Storage

Objectives

After completing this section, you should be able to partition storage devices, configure LVM, format partitions or logical volumes, mount file systems, and add swap files or spaces.

Configuring Storage with Ansible Modules

Red Hat Ansible Automation Platform provides a collection of modules to configure storage devices on managed hosts. Those modules support partitioning devices, creating logical volumes, and creating and mounting filesystems.

The `parted` Module

The `parted` module supports the partition of block devices. This module includes the functionality of the `parted` command, and allows to create partitions with a specific size, flag, and alignment. The following table lists some of the parameters for the `parted` module.

Parameter name	Description
align	Configures partition alignment.
device	Block device.
flags	Flags for the partition.
number	The partition number.
part_end	Partition size from the beginning of the disk specified in `parted` supported units.
state	Creates or removes the partition.
unit	Size units for the partition information.

The following example creates a new partition of 10 GB.

```
- name: New 10GB partition
  parted:
    device: /dev/vdb (1)
    number: 1 (2)
    state: present (3)
    part_end: 10GB (4)
```

(1) Uses `vdb` as the block device to partition.

(2) Creates the partition number one.

(3) Ensures the partition is available.

(4) Sets the partition size to 10 GB.

The `lvg` and `lvol` Modules

The `lvg` and `lvol` modules support the creation of logical volumes, including the configuration of physical volumes, and volume groups. The `lvg` takes as parameters the block devices to configure as the back end physical volumes for the volume group. The following table lists some of the parameters for the `lvg` module.

Parameter name	Description
pesize	The size of the physical extent. Must be a power of 2, or multiple of 128 KiB.
pvs	List of comma-separated devices to be configured as physical volumes for the volume group.
vg	The name of the volume group.
state	Creates or removes the volume.

The following task creates a volume group with a specific physical extent size using a block device as a back end.

```
- name: Creates a volume group
  lvg:
    vg: vg1 (1)
    pvs: /dev/vda1 (2)
    pesize: 32 (3)
```

(1) The volume group name is `vg1`.

(2) Uses `/dev/vda1` as the back end physical volume for the volume group.

(3) Sets the physical extent size to `32`.

In the following example, if the `vg1` volume group is already available with `/dev/vdb1` as a physical volume, the volume is enlarged adding a new physical volume with `/dev/vdc1`.

```
- name: Resize a volume group
  lvg:
    vg: vg1
    pvs: /dev/vdb1,/dev/vdc1
```

The `lvol` module creates logical volumes, and supports the resizing and shrinking of those volumes, and the filesystems on top of them. This module also supports the creation of snapshots for the logical volumes. The following table lists some of the parameters for the `lvol` module.

Parameter name	Description
lv	The name of the logical volume.
resizefs	Resizes the filesystem with the logical volume.
shrink	Enable logical volume shrink.
size	The size of the logical volume.
snapshot	The name of the snapshot for the logical volume.
state	Create or remove the logical volume.
vg	The parent volume group for the logical volume.

The following task creates a logical volume of 2 GB.

```
- name: Create a logical volume of 2GB
  lvol:
    vg: vg1 (1)
    lv: lv1 (2)
    size: 2g (3)
```

(1) The parent volume group name is `vg1`.

(2) The logical volume name is `lv1`.

(3) The size of the logical volume is 2 GB.

The `filesystem` Module

The `filesystem` module supports both creating and resizing a filesystem. This module supports filesystem resizing for `ext2`, `ext3`, `ext4`, `ext4dev`, `f2fs`, `lvm`, `xfs`, and `vfat`. The following table lists some of the parameters for the `filesystem` module.

Parameter name	Description
dev	Block device name.
fstype	Filesystem type.
resizefs	Grows the filesystem size to the size of the block device.

The following example creates a filesystem on a partition.

```
- name: Create an XFS filesystem
  filesystem:
    fstype: xfs (1)
    dev: /dev/vdb1 (2)
```

(1) Uses the `XFS` filesystem.

(2) Uses the `/dev/vdb1` device.

The `mount` Module

The `mount` module supports the configuration of mount points on `/etc/fstab`. The following table lists some of the parameters for the `mount` module.

Parameter name	Description
fstype	Filesystem type.
opts	Mount options.
path	Mount point path.
src	Device to be mounted.
state	Specify the mount status. If set to `mounted`, the system mounts the device, and configures `/etc/fstab` with that mount information. To unmount the device and remove it from `/etc/fstab` use `absent`.

The following example mounts a device with an specific ID.

```
- name: Mount device with ID
  mount:
    path: /data (1)
    src: UUID=a8063676-44dd-409a-b584-68be2c9f5570 (2)
    fstype: xfs (3)
    state: present (4)
```

(1) Uses `/data` as the mount point path.

(2) Mounts the device with the `a8063676-44dd-409a-b584-68be2c9f5570` ID.

(3) Uses the `XFS` filesystem.

(4) Mounts the device and configures `/etc/fstab` accordingly.

The following example mounts the NFS share available at `172.25.250.100:/share` on the `/nfsshare` directory at the managed host.

```
- name: Mount NFS share
  mount:
    path: /nfsshare
    src: 172.25.250.100:/share
    fstype: nfs
    opts: defaults
    dump: '0'
    passno: '0'
    state: mounted
```

Configuring swap with Modules

Red Hat Ansible Automation Platform does not currently include modules to manage swap memory. To add swap memory to a system with Ansible with logical volumes you need to create a new volume group and logical volume with the `lvg` and `lvol` modules. When ready, you need to format as swap the new logical volume using the `command` module with the `mkswap` command. Finally, you need to activate the new swap device using the `command` module with the `swapon` command. Ansible includes the `ansible_swaptotal_mb` variable which includes the total swap memory. You can use this variable to trigger swap configuration and enablement when swap memory is low. The following tasks, create a volume group and a logical volume for swap memory, format that logical volume as swap, and activates it.

```
- name: Create new swap VG
  lvg:
    vg: vgswap
    pvs: /dev/vda1
    state: present

- name: Create new swap LV
  lvol:
    vg: vgswap
    lv: lvswap
    size: 10g

- name: Format swap LV
  command: mkswap /dev/vgswap/lvswap
  when: ansible_swaptotal_mb < 128

- name: Activate swap LV
  command: swapon /dev/vgswap/lvswap
  when: ansible_swaptotal_mb < 128
```

Ansible Facts for Storage Configuration

Ansible uses facts to retrieve information to the control node about the configuration of the managed hosts. You can use the `setup` Ansible module to retrieve all the Ansible facts for a managed host.

```
[user@controlnode ~]$ ansible webservers -m setup
host.lab.example.com | SUCCESS => {
    "ansible_facts": {
...output omitted...
    }
```

The `filter` option for the `setup` module supports fine-grained filtering based on shell-style wildcards.

The `ansible_devices` element includes all the storage devices available on the managed host. The element for each storage device includes additional information like partitions or total size. The following example displays the `ansible_devices` element for a managed host with three storage devices: `sr0`, `vda`, and `vdb`.

```
[user@controlnode ~]$ ansible webservers -m setup -a 'filter=ansible_devices'
host.lab.example.com | SUCCESS => {
    "ansible_facts": {
        "ansible_devices": {
            "sr0": {
                "holders": [],
                "host": "IDE interface: Intel Corporation 82371SB PIIX3 IDE [Natoma/Triton II]",
                "links": {
                    "ids": [
                        "ata-QEMU_DVD-ROM_QM00003"
                    ],
                    "labels": [],
                    "masters": [],
                    "uuids": []
                },
                "model": "QEMU DVD-ROM",
                "partitions": {},
                "removable": "1",
                "rotational": "1",
                "sas_address": null,
                "sas_device_handle": null,
                "scheduler_mode": "mq-deadline",
                "sectors": "2097151",
                "sectorsize": "512",
                "size": "1024.00 MB",
                "support_discard": "0",
                "vendor": "QEMU",
                "virtual": 1
            },
            "vda": {
                "holders": [],
                "host": "SCSI storage controller: Red Hat, Inc. Virtio block device",
                "links": {
                    "ids": [],
                    "labels": [],
                    "masters": [],
                    "uuids": []
                },
                "model": null,
                "partitions": {
                    "vda1": {
                        "holders": [],
                        "links": {
                            "ids": [],
                            "labels": [],
                            "masters": [],
                            "uuids": [
                                "a8063676-44dd-409a-b584-68be2c9f5570"
                            ]
                        },
                        "sectors": "20969439",
                        "sectorsize": 512,
                        "size": "10.00 GB",
                        "start": "2048",
                        "uuid": "a8063676-44dd-409a-b584-68be2c9f5570"
                    }
                },
                "removable": "0",
                "rotational": "1",
                "sas_address": null,
                "sas_device_handle": null,
                "scheduler_mode": "mq-deadline",
                "sectors": "20971520",
                "sectorsize": "512",
                "size": "10.00 GB",
                "support_discard": "0",
                "vendor": "0x1af4",
                "virtual": 1
            },
            "vdb": {
                "holders": [],
                "host": "SCSI storage controller: Red Hat, Inc. Virtio block device",
                "links": {
                    "ids": [],
                    "labels": [],
                    "masters": [],
                    "uuids": []
                },
                "model": null,
                "partitions": {},
                "removable": "0",
                "rotational": "1",
                "sas_address": null,
                "sas_device_handle": null,
```

```
                "scheduler_mode": "mq-deadline",
                "sectors": "10485760",
                "sectorsize": "512",
                "size": "5.00 GB",
                "support_discard": "0",
                "vendor": "0x1af4",
                "virtual": 1
            }
        }
    },
    "changed": false
}
```

The `ansible_device_links` element includes all the links available for each storage device. The following example displays the `ansible_device_links` element for a managed host with two storage devices, `sr0` and `vda1`, which have an associated ID.

```
[user@controlnode ~]$ ansible webservers -m setup -a 'filter=ansible_device_links'
host.lab.example.com | SUCCESS => {
    "ansible_facts": {
        "ansible_device_links": {
            "ids": {
                "sr0": [
                    "ata-QEMU_DVD-ROM_QM00003"
                ]
            },
            "labels": {},
            "masters": {},
            "uuids": {
                "vda1": [
                    "a8063676-44dd-409a-b584-68be2c9f5570"
                ]
            }
        }
    },
    "changed": false
}
```

The `ansible_mounts` element includes information about the current mounted devices on the managed host, like the mounted device, the mount point, and the options. The following output displays the `ansible_mounts` element for a managed host with one active mount, `/dev/vda1` on the `/` directory.

```
[user@controlnode ~]$ ansible webservers -m setup -a 'filter=ansible_mounts'
host.lab.example.com | SUCCESS => {
    "ansible_facts": {
        "ansible_mounts": [
            {
                "block_available": 2225732,
                "block_size": 4096,
                "block_total": 2618619,
                "block_used": 392887,
                "device": "/dev/vda1",
                "fstype": "xfs",
                "inode_available": 5196602,
                "inode_total": 5242304,
                "inode_used": 45702,
                "mount": "/",
                "options": "rw,seclabel,relatime,attr2,inode64,noquota",
                "size_available": 9116598272,
                "size_total": 10725863424,
                "uuid": "a8063676-44dd-409a-b584-68be2c9f5570"
            }
        ]
    },
    "changed": false
}
```

REFERENCES

parted – Configure block device partitions — Ansible Documentation (https://docs.ansible.com/ansible/2.9/modules/parted_module.html)

lvg – Configure LVM volume groups — Ansible Documentation (https://docs.ansible.com/ansible/2.9/modules/lvg_module.html)

lvol – Configure LVM logical volumes — Ansible Documentation (https://docs.ansible.com/ansible/2.9/modules/lvol_module.html)

filesystem – Makes a filesystem — Ansible Documentation (https://docs.ansible.com/ansible/2.9/modules/filesystem_module.html)

mount – Control active and configured mount points — Ansible Documentation (https://docs.ansible.com/ansible/2.9/modules/mount_module.html)

← PREVIOUS (/ROL/APP/COURSES/RH294-8.4/PAGES/CH09S06)

→ NEXT (/ROL/APP/COURSES/RH294-8.4/PAGES/CH09S08)