day05

ansible

ansible模块

yum_repository

- 用于配置yum
- 常用选项:
  - file: 指定文件名
  - 其他选项，请与文件内容对照

```
# 在test组中的主机上，配置yum
[root@control ansible]# ansible test -m yum_repository
-a "file=myrepo name=myApp description='My App'
baseurl=ftp://192.168.4.254/rhel8/AppStream gpgcheck=no
enabled=yes"

[root@node1 ~]# cat /etc/yum.repos.d/myrepo.repo
[myApp]
baseurl = ftp://192.168.4.254/rhel8/AppStream
enabled = 1
gpgcheck = 0
name = My App

[root@control ansible]# ansible test -m yum_repository
-a "file=myrepo name=BaseOS description='Base OS'
baseurl=ftp://192.168.4.254/rhel8/BaseOS gpgcheck=no
enabled=yes"
```

```
[root@node1 ~]# cat /etc/yum.repos.d/myrepo.repo
[myApp]
baseurl = ftp://192.168.4.254/rhel8/AppStream
enabled = 1
gpgcheck = 0
name = My App

[BaseOS]
baseurl = ftp://192.168.4.254/rhel8/BaseOS
enabled = 1
gpgcheck = 0
name = Base OS
```

yum模块

- 用于rpm软件包管理，如安装、升级、卸载
- 常用选项：
  - name：包名
  - state：状态。present表示安装，如果已安装则忽略；latest表示安装或升级到最新版本；absent表示卸载。

```
# 在test组中的主机上安装tar
[root@control ansible]# ansible test -m yum -a
"name=tar state=present"

# 在test组中的主机上安装wget、net-tools
[root@control ansible]# ansible test -m yum -a
"name=wget,net-tools"

# 在test组中的主机上卸载wget
[root@control ansible]# ansible test -m yum -a
"name=wget state=absent"
```

service模块

- 用于控制服务。启动、关闭、重启、开机自启。
- 常用选项:
  - name：控制的服务名
  - state：started表示启动；stopped表示关闭；restarted表示重启
  - enabled：yes表示设置开机自启；no表示设置开机不要自启。

```
# 在test主机上安装httpd
[root@control ansible]# ansible test -m yum -a
"name=httpd state=latest"

#  在test主机上启动httpd，并设置它开机自启
[root@control ansible]# ansible test -m service -a
"name=httpd state=started enabled=yes"
```

逻辑卷相关模块

- 逻辑卷可以动态管理存储空间。可以对逻辑卷进行扩容或缩减。
- 可以把硬盘或分区转换成物理卷PV；再把1到多个PV组合成卷组VG；然后在VG上划分逻辑卷LV。LV可以像普通分区一样，进行格式化、挂载。
- 关闭虚拟机node1，为其添加2块20GB的硬盘
- LINUX 下 KVM 虚拟机新加的硬盘，名称是 /dev/vdb 和/dev/vdc
- vmware虚拟机新加的硬盘，名称是/dev/sdb和/dev/sdc
- 如果选nvme硬盘，名称可能是/dev/nvme0n1和/dev/nvme0n2

```
[root@node1 ~]# lsblk    # 可以查看到新加的硬盘vdb和vdc
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sr0     11:0    1 1024M  0 rom
vda    253:0    0   30G  0 disk
`-vda1 253:1    0   20G  0 part /
vdb    253:16   0   20G  0 disk
vdc    253:32   0   20G  0 disk
```

lvg模块

- 创建、删除卷组，修改卷组大小
- 常用选项：
  - vg：定义卷组名。vg：volume group
  - pvs：由哪些物理卷构成。pvs：physical volumes

```
# 在test组中的主机上安装lvm2，state不写，默认是present
```

```
[root@control ansible]# ansible test -m yum -a
"name=lvm2"

# 手工在node1上对vdb进行分区
[root@node1 ~]# fdisk /dev/vdb
Command (m for help): g    # 创建GPT分区表
Command (m for help): n    # 新建分区
Partition number (1-128, default 1):    # 回车，使用1号
分区
First sector (2048-41943006, default 2048):   # 起始位
置，回车
Last sector, +sectors or +size{K,M,G,T,P} (2048-
41943006, default 41943006): +5G   # 结束位置+5G

Command (m for help): n   # 新建分区
Partition number (2-128, default 2):   # 回车，使用2号
分区
First sector (10487808-41943006, default 10487808): #
起始位置，回车
Last sector, +sectors or +size{K,M,G,T,P} (10487808-
41943006, default 41943006): # 结束位置，回车，分区到结
尾
Command (m for help): w   # 存盘

[root@node1 ~]# lsblk    # vdb被分出来了两个分区
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sr0     11:0    1 1024M  0 rom
vda    253:0    0   30G  0 disk
`-vda1 253:1    0   20G  0 part /
vdb    253:16   0   20G  0 disk
|-vdb1 253:17   0    5G  0 part
```

```
`-vdb2 253:18   0   15G  0 part
vdc    253:32   0   20G  0 disk

# 在test组中的主机上创建名为myvg的卷组，该卷组
由/dev/vdb1组成
[root@control ansible]# ansible test -m lvg -a "vg=myvg
pvs=/dev/vdb1"

# 在node1上查看卷组
[root@node1 ~]# vgs
  VG   #PV #LV #SN Attr   VSize  VFree
  myvg   1   0   0 wz--n- <5.00g <5.00g

# 扩容卷组。卷组由PV构成，只要向卷组中加入新的PV，即可
实现扩容
[root@control ansible]# ansible test -m lvg -a "vg=myvg
pvs=/dev/vdb1,/dev/vdb2"

[root@node1 ~]# vgs  # 在node1上查看卷组
  VG   #PV #LV #SN Attr   VSize  VFree
  myvg   2   0   0 wz--n- 19.99g 19.99g
```

lvol模块

- 创建、删除逻辑卷，修改逻辑卷大小
- 常用选项：
  - vg：指定在哪个卷组上创建逻辑卷
  - lv：创建的逻辑卷名。lv：logical volume
  - size：逻辑卷的大小，不写单位，以M为单位

```
# 在test组中的主机上创建名为mylv的逻辑卷，大小为2GB
[root@control ansible]# ansible test -m lvol -a
"vg=myvg lv=mylv size=2G"

# 在node1上查看逻辑卷
[root@node1 ~]# lvs
  LV   VG   Attr       LSize Pool Origin Data%  Meta%
Move Log Cpy%Sync Convert
  mylv myvg -wi-a----- 2.00g

# mylv扩容至4GB
[root@control ansible]# ansible test -m lvol -a
"vg=myvg lv=mylv size=4G"

[root@node1 ~]# lvs  # 在node1上查看逻辑卷
  LV   VG   Attr       LSize Pool Origin Data%  Meta%
Move Log Cpy%Sync Convert
  mylv myvg -wi-a----- 4.00g
```

filesystem模块

- 用于格式化，也就是创建文件系统
- 常用选项：
  - fstype：指定文件系统类型
  - dev：指定要格式化的设备，可以是分区，可以是逻辑卷

```
#  在test组中的主机上，把/dev/myvg/mylv格式化为xfs
[root@control ansible]# ansible test -m filesystem -a
"fstype=xfs dev=/dev/myvg/mylv"

# 在node1上查看格式化结果
[root@node1 ~]# blkid /dev/myvg/mylv
/dev/myvg/mylv: UUID="46c0af72-e517-4b15-9e53-
ec72fbe1d96e" TYPE="xfs"
```

mount模块

- 用于挂载文件系统
- 常用选项：
  - path：挂载点。如果挂载点不存在，自动创建。
  - src：待挂载的设备
  - fstype：文件系统类型
  - state：mounted，表示永久挂载

```
# 在test组中的主机上，把/dev/myvg/mylv永久挂载到/data
[root@control ansible]# ansible test -m mount -a
"path=/data src=/dev/myvg/mylv state=mounted
fstype=xfs"

# 在node1上查看
[root@node1 ~]# tail -1 /etc/fstab
/dev/myvg/mylv /data xfs defaults 0 0
[root@node1 ~]# df -h /data/
Filesystem              Size  Used Avail Use% Mounted on
/dev/mapper/myvg-mylv  4.0G   61M  4.0G   2% /data
```

```
# 在test组中的主机上，卸载/dev/myvg/mylv
[root@control ansible]# ansible test -m mount -a
"path=/data state=absent"

# 在test组中的主机上，强制删除/dev/myvg/mylv
[root@control ansible]# ansible test -m lvol -a
"lv=mylv state=absent vg=myvg force=yes"   # force是强
制

# 在test组中的主机上，删除myvg卷组
[root@control ansible]# ansible test -m lvg -a "vg=myvg
state=absent"
```

Playbook剧本

- 常用于复杂任务的管理，以及管理经常要完成的任务
- playbook也是通过模块和它的参数，在特定主机上执行任务
- playbook是一个文件，该文件中需要通过yaml格式进行书写

YAML

- YAML Ain't a Markup Language：YAML不是一个标记语言

yaml语法规范

1. yaml文件的文件名，一般以yml或yaml作为扩展名
2. 文件一般以---作为第一行，不是必须的，但是常用
3. 键值对使用冒号:表示，冒号后面必须有空格。
4. 数组使用-表示，-后面必须有空格。
5. 相同的层级必须有相同的缩进。如果缩进不对，则有语法错误。每一级缩进，建议2个空格。
6. 全文不能使用tab，必须使用空格。

配置vim适应yaml语法

```
# 文件位置和名字是固定的，用于设置vim的格式
[root@control ansible]# vim ~/.vimrc
set ai          # 设置自动缩进
set ts=2        # 设置按tab键，缩进2个空格
set et          # 将tab转换成相应个数的空格
```

编写playbook

- 一个剧本（即playbook），可以包含多个play
- 每个play用于在指定的主机上，通过模块和参数执行相应的任务
- 每个play可以包含多个任务。
- 任务有模块和参数构成。

```
# 编写用于测试连通性的playbook，相当于执行ansible all -m ping
[root@control ansible]# vim test.yml
---
- hosts: all
  tasks:
    - ping:

[root@control ansible]# ansible-playbook test.yml  # 执行playbook

# 以上更规范的写法如下：
[root@control ansible]# vim test.yml
---
- name: test network     # play的名字，可选项
  hosts: all             # 作用于所有的主机
  tasks:                 # 任务
    - name: task 1       # 第1个任务的名字，可选项
      ping:              # 第1个任务使用的模块

[root@control ansible]# ansible-playbook test.yml  # 执行playbook
```

```
# 在test组的主机和node2上创建/tmp/demo目录，权限是0755。将控制端/etc/hosts拷贝到目标主机的/tmp/demo中
[root@control ansible]# vim fileop.yml
---
- name: create dir and copy file
  hosts: test,node2    # 这里的名称，必须出现在主机清单文件中
  tasks:
    - name: create dir
      file:
        path: /tmp/demo
        state: directory
        mode: '0755'

    - name: copy file
      copy:
        src: /etc/hosts
        dest: /tmp/demo/hosts

# 执行playbook
[root@control ansible]# ansible-playbook fileop.yml


# 在test组中的主机上，创建用户bob，附加组是adm；在node2主机上，创建/tmp/hi.txt，其内容为Hello World.
[root@control ansible]# vim two.yml
---
- name: create user
  hosts: test
```

```
  tasks:
    - name: create bob
      user:
        name: bob
        groups: adm

- name: create file
  hosts: node2
  tasks:
    - name: make file
      copy:
        dest: /tmp/hi.txt
        content: "Hello World\n"

[root@control ansible]# ansible-playbook two.yml
```

- |和>的区别：|它保留换行符，>把多行合并为一行

```
# 通过copy模块创建/tmp/1.txt，文件中有两行内容，分别是Hello World和ni hao
[root@control ansible]# vim f1.yml
---
- name: play 1
  hosts: test
  tasks:
    - name: mkfile 1.txt
      copy:
        dest: /tmp/1.txt
        content: |
          Hello World!
          ni hao.
```

```
[root@control ansible]# ansible-playbook f1.yml
# 查看结果
[root@node1 ~]# cat /tmp/1.txt
Hello World!
ni hao.


# 通过copy模块创建/tmp/2.txt，文件中有一行内容，分别是Hello World! ni hao
[root@control ansible]# vim f2.yml
---
- name: play 1
  hosts: test
  tasks:
    - name: mkfile 2.txt
      copy:
        dest: /tmp/2.txt
        content: >
          Hello World!
          ni hao.

[root@control ansible]# ansible-playbook f2.yml
[root@node1 ~]# cat /tmp/2.txt
Hello World! ni hao.
```

- playbook示例

```
# 在test组中的主机上创建john用户，它的uid是1040，主组是daemon，密码为123
[root@control ansible]# vim user_john.yml
```

```
---
- name: create user
  hosts: test
  tasks:
    - name: create user john
      user:
        name: john
        uid: 1040
        group: daemon
        password: "{{'123'|password_hash('sha512')}}"
[root@control ansible]# ansible-playbook user_john.yml

# 在test组中的主机上删除用户john
[root@control ansible]# vim del_john.yml
---
- name: delete user
  hosts: test
  tasks:
    - name: delete user john
      user:
        name: john
        state: absent
[root@control ansible]# ansible-playbook del_john.yml
```

硬盘管理

- 常用的分区表类型有：MBR（主引导记录）、GPT（GUID分区表）
- MBR最多支持4个主分区，或3个主分区加1个扩展分区。最大支持2.2TB左右的硬盘

- GPT最多支持128个主分区。支持大硬盘

parted模块

- 用于硬盘分区管理
- 常用选项：
  - device：待分区的设备
  - number：分区编号
  - state：present表示创建，absent表示删除
  - part_start：分区的起始位置，不写表示从开头
  - part_end：表示分区的结束位置，不写表示到结尾

```
# 在test组中的主机上，对/dev/vdc进行分区，创建1个1GB的主分区
[root@control ansible]# vim disk.yml
---
- name: disk manage
  hosts: test
  tasks:
    - name: create a partition
      parted:
        device: /dev/vdc
        number: 1
        state: present
        part_end: 1GiB

[root@control ansible]# ansible-playbook disk.yml

# 在目标主机上查看结果
[root@node1 ~]# lsblk
```

```
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
.. ...
vdc    253:32   0   20G  0 disk
`-vdc1 253:33   0 1023M  0 part

# 继续编辑disk.yml，对/dev/vdc进行分区，创建1个新的5GB的主分区
[root@control ansible]# vim disk.yml
... ...
    - name: add a new partition
      parted:
        device: /dev/vdc
        number: 2
        state: present
        part_start: 1GiB
        part_end: 6GiB

[root@control ansible]# ansible-playbook disk.yml
[root@node1 ~]# lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
... ...
vdc    253:32   0   20G  0 disk
|-vdc1 253:33   0 1023M  0 part
`-vdc2 253:34   0    5G  0 part

# 继续编辑disk.yml，创建名为my_vg的卷组，它由上面创建的vdc1和vdc2构成
[root@control ansible]# vim disk.yml
... ...
    - name: create my_vg
      lvg:
```

```
        vg: my_vg
        pvs: /dev/vdc1,/dev/vdc2

# 继续编辑disk.yml，在my_vg卷组上创建名为my_lv的逻辑卷，大小1G
[root@control ansible]# vim disk.yml
... ...
    - name: create my_lv
      lvol:
        vg: my_vg
        lv: my_lv
        size: 1G


# 继续编辑disk.yml，格式化my_lv为ext4
[root@control ansible]# vim disk.yml
... ...
    - name: mkfs my_lv
      filesystem:
        dev: /dev/my_vg/my_lv
        fstype: ext4


# 继续编辑disk.yml，将my_lv挂载到/data
[root@control ansible]# vim disk.yml
... ...
    - name: mount my_lv
      mount:
        path: /data
        src: /dev/my_vg/my_lv
        fstype: ext4
```

```
        state: mounted

# 完整的disk.yml如下
---
- name: disk manage
  hosts: test
  tasks:
    - name: create a partition
      parted:
        device: /dev/vdc
        number: 1
        state: present
        part_end: 1GiB

    - name: add a new partition
      parted:
        device: /dev/vdc
        number: 2
        state: present
        part_start: 1GiB
        part_end: 6GiB

    - name: create my_vg
      lvg:
        vg: my_vg
        pvs: /dev/vdc1,/dev/vdc2

    - name: create my_lv
      lvol:
        vg: my_vg
        lv: my_lv
```

```
      size: 1G

  - name: mkfs my_lv
    filesystem:
      dev: /dev/my_vg/my_lv
      fstype: ext4

  - name: mount my_lv
    mount:
      path: /data
      src: /dev/my_vg/my_lv
      fstype: ext4
      state: mounted
```