1. Ansible日常应用场景

1.1. 安装categraf监控采集器，n9e_addr为n9e监控服务器地址

1.1.1. 任务内容

```
[root@localhost ansible]# cat roles/categraf/tasks/install.yml
---
- name: 拷贝categraf安装包到/opt目录
  copy:
    src: categraf.tar.gz
    dest: /opt

- name: 安装categraf到/opt目录
  shell: |
    tar -zxvf categraf.tar.gz
    chmod -R 755 categraf/
  args:
    chdir: /opt/

- name: 添加systemd配置文件
  copy:
    src: categraf.service
    dest: /usr/lib/systemd/system/
    mode: 0600

- name: 配置categraf开机自启动
  systemd:
    name: categraf
    enabled: yes
    daemon_reload: yes

- import_tasks: start.yml
- import_tasks: config.yml
```

1.1.2. 任务执行

```
ansible-playbook -i inventory/ -e n9e_addr=192.168.77.129:17000 -e
operation=install categraf.yml
```

1.1.3. 配置检查

```
#启停categraf服务及查看服务状态
ansible-playbook -i inventory/ -e operation=start|stop|status categraf.yml
```

1.2. 为RHEL系列版本安装安全插件yum-security或yum-plugin-security

1.2.1. 任务内容

```
[root@localhost ansible]# cat roles/yum_security/tasks/install.yml
---

- name: 为RHEL5系列版本安装安全插件yum-security
  yum:
    name: yum-security
    state: latest
  when: ansible_distribution == "RedHat" and ansible_distribution_major_version
== "5"
- name: 为RHEL6系列版本安装安全插件yum-plugin-security
  yum:
    name: yum-plugin-security
    state: latest
  when: ansible_distribution == "RedHat" and ansible_distribution_major_version
== "6"
- import_tasks: check.yml
```

1.2.2. 任务执行

```
ansible-playbook -i inventory/ -e operation=install yum_security.yml
```

1.2.3. 配置检查

```
#检查是否安装安全插件yum-security或yum-plugin-security
ansible-playbook -i inventory/ -e operation=check yum_security.yml
```

1.3. 为指定网卡配置添加静态路由策略

1.3.1. 任务内容

```
[root@localhost ansible]# cat roles/route_add/tasks/config.yml
---
- name: 为ens33网卡添加静态路由策略
  copy:
    content: "192.168.2.1/24 via 192.168.77.2 dev ens33"
    dest: /etc/sysconfig/network-scripts/route-ens33
    backup: yes
- name: 重启网卡，使添加的静态路由策略生效
  service:
    name: NetworkManager
    state: restarted
    enabled: yes
    args: ens33
- import_tasks: check.yml
```

1.3.2. 任务执行

```
ansible-playbook -i inventory/ -e operation=config route_add.yml
```

1.3.3. 配置检查

```
#检查路由生效配置
ansible-playbook -i inventory/ -e operation=check route_add.yml
```

1.4. 新增|删除指定yum存储库

1.4.1. 任务内容

```
[root@localhost ansible]# cat roles/yum_repository/tasks/add.yml
---
- name: 新增名为yumlocal的yum存储库
  yum_repository:
    name: yumlocal
    description: yumlocal
    file: yumlocal
    baseurl: https://download.fedoraproject.org/pub/epel/$releasever/$basearch/
    gpgcheck: no
    enabled: yes
  notify: yum-clean-metadata
- import_tasks: check.yml
```

1.4.2. 任务执行

```
ansible-playbook -i inventory/ -e operation=add|remove yum_repository.yml
```

1.4.3. 配置检查

```
#检查yum存储库配置
ansible-playbook -i inventory/ -e operation=check yum_repository.yml
```

1.5. 对目标节点指定文件实现压缩备份并将备份文件拉取到Ansible控制机器

1.5.1. 任务内容

```
[root@localhost ansible]# cat roles/backup/tasks/backup.yml
---
- name: 定义存放备份文件的目录
  set_fact:
    backup_dir: "/tmp/backup/"

- name: 确保备份目录存在,如果不存在则创建
  file:
    path: "{{ backup_dir }}"
```

```
    state: directory
    mode: '0755'

- name: 定义时间格式为"YYYY-MM-DD"
  command: date +"%F"
  register: datetime

- name: 对/etc/*及/var/log/*进行压缩备份并存储在{{ backup_dir }}目录下
  archive:
    path:
    - /etc/*
    - /var/log/*
    dest: "{{ backup_dir }}/etc-varbak{{datetime.stdout}}.tar.bz2"
    format: bz2

- name: 将备份文件从远程主机拉取到Ansible控制机器{{ backup_dir }}目录下
  fetch:
    src: "{{ backup_dir }}/etc-varbak{{datetime.stdout}}.tar.bz2"
    dest: "{{ backup_dir }}/{{ inventory_hostname }}/"
    flat: yes
- import_tasks: check.yml
```

1.5.2. 任务执行

```
ansible-playbook -i inventory/ -e operation=backup backup.yml
```

1.5.3. 配置检查

```
#检查备份任务执行情况
ansible-playbook -i inventory/ -e operation=check backup.yml
```

1.6. 执行linux系统基线扫描任务，并将扫描结果拉取到本地的/tmp/jixian/目录

1.6.1. 任务内容

```
[root@localhost ansible]# cat roles/jixian_check/tasks/check_linux.yml
---
- name: 创建/tmp/jixian基线检查存在脚本目录
  file:
    path: /tmp/jixian
    state: directory
- name: 复制主机基线检查脚本文件到/tmp/jixian
  copy:
    src: '{{ item }}'
    dest: /tmp/jixian
  loop:
    - check_server_openeuler.pl
    - check_server_openeuler.sh

- name: 在/tmp/jixian目录下执行基线检查命令
  shell: perl check_server_openeuler.pl  "{{ inventory_hostname }}"
```

```
  args:
    chdir: /tmp/jixian
- name: 将基线检查产生的结果文件拉取到本地的/tmp/jixian/目录
  fetch:
    src: /tmp/jixian/{{ inventory_hostname }}_OpenEuler_chk.xml
    dest: /tmp/jixian/
    flat: yes

- import_tasks: check.yml
```

1.6.2. 任务执行

```
ansible-playbook -i inventory/ -e operation=check_linux jixian_check.yml
```

1.6.3. 配置检查

```
ansible-playbook -i inventory/ -e operation=check jixian_check.yml
```

1.7. 执行mysql数据库基线扫描任务，并将扫描结果拉取到本地的/tmp/jixian/目录

1.7.1. 任务内容

```
[root@localhost ansible]# cat roles/jixian_check/tasks/check_mysql.yml
---
- name: 创建/tmp/jixian基线检查存在脚本目录
  file:
    path: /tmp/jixian
    state: directory
- name:  复制数据库基线检查脚本文件到/tmp/jixian
  copy:
    src: '{{ item }}'
    dest: /tmp/jixian
  loop:
    - check_database_mysql_linux.pl
    - check_database_mysql_linux.sh
- name: 在/tmp/jixian目录下执行数据库基线检查命令
  shell: sh check_database_mysql_linux.sh  "{{ inventory_hostname }}" "{{
mysql_passwd }}" "{{ mysql_user }}" "{{ mysql_port }}" null
  args:
    chdir: /tmp/jixian/
- name: 将数据库基线检查产生的结果文件拉取到本地的/tmp/jixian/目录
  fetch:
    src: /tmp/jixian/{{ inventory_hostname }}_linux_mysql_chk.xml
    dest: /tmp/jixian/
    flat: yes

- import_tasks: check.yml
```

1.7.2. 任务执行

```
ansible-playbook -i inventory/ -e operation=check_mysql jixian_check.yml
```

1.7.3. 配置检查

```
ansible-playbook -i inventory/ -e operation=check jixian_check.yml
```

1.8. 对应用程序执行hash渗透扫描并拉取结果文件到控制节点

1.8.1. 任务内容

```
[root@localhost ansible]# cat roles/dumphash_check/tasks/dumphash_check.yml
---
- name: 复制安全检查脚本DumpHash文件到程序运行根目录/app
  copy:
    src: '{{ item }}'
    dest: /app
    owner: app
    group: app
    mode: 0750
  loop:
    - DumpHash

- name: 在程序根目录/app下运行DumpHash文件执行安全检查
  shell: ./DumpHash
  args:
    chdir: /app
- name: 将安全检查产生的结果文件拉取到本地的/tmp/jixian/DumpHash/目录
  fetch:
    src: /tmp/filehash.res
    dest: /tmp/jixian/DumpHash/{{ inventory_hostname }}_filehash.res
    flat: yes

- import_tasks: check.yml
```

1.8.2. 任务执行

```
 ansible-playbook -i inventory/ -e operation=dumphash_check dumphash_check.yml
```

1.8.3. 配置检查

```
#检查安全检查执行结果文件
ansible-playbook -i inventory/ -e  operation=check dumphash_check.yml
```

1.9. 关闭firewalld服务

1.9.1. 任务内容

```
[root@localhost ansible]# cat roles/firewalld/tasks/config.yml
- name: 关闭firewalld服务
  service:
    name: firewalld
    state: stopped
    enabled: no
- import_tasks: check.yml
```

1.9.2. 任务执行

```
ansible-playbook -i inventory/ -e operation=config firewalld.yml
```

1.9.3. 配置检查

```
#检查firewalld服务
ansible-playbook -i inventory/ -e operation=check firewalld.yml
```

1.10. 关闭selinux服务

1.10.1. 任务内容

```
[root@localhost ansible]# cat roles/selinux/tasks/config.yml
- name: 临时关闭selinux
  shell: setenforce 0
  ignore_errors: yes

- name: 关闭selinux服务
  selinux:
    state: disabled
  ignore_errors: yes

- import_tasks: check.yml
```

1.10.2. 任务执行

```
ansible-playbook -i inventory/ -e operation=config selinux.yml
```

1.10.3. 配置检查

```
#检查selinux服务
ansible-playbook -i inventory/ -e operation=check selinux.yml
```

1.11. 创建{{ username }}用户并设置密码，如果用户已经存在则修改密码。同时配置拥有sudo权限

1.11.1. 任务内容

```
[root@localhost ansible]# cat roles/useradd/tasks/useradd.yml
- name: 创建用户 {{ username }} 并设置密码，如果用户已经存在则修改密码
  user:
    name: "{{ username }}"
    password: "{{ chapass | string | password_hash('sha512') }}"
    state: present
    update_password: always
- name: 为 {{ username }} 用户配置sudo权限
  lineinfile:
    path: /etc/sudoers
    state: present
    regexp: '^{{ username }}'
    line: '{{ username }} ALL=(ALL) ALL'
    validate: '/usr/sbin/visudo -cf %s'
- import_tasks: check.yml
```

1.11.2. 任务执行

```
ansible-playbook -i inventory/ -e operation=useradd  -e username=weihu
useradd.yml
```

1.11.3. 配置检查

```
#检查指定用户{{ username }}是否存在，同时检查是否配置sudo权限
ansible-playbook -i inventory/ -e operation=check  -e username=weihu  useradd.yml
```

1.12. 使用自动生成的随机密码完成{{ user_name }}用户的创建及密码设置

1.12.1. 任务内容

```
[root@localhost ansible]#  cat roles/useradd_v1/tasks/useradd_v1.yml
---
- name: 生成一个12位数的随机密码
  shell: </dev/urandom tr -dc 'A-Za-z0-9!#$%^&' | head -c 12
  register: chapass
  delegate_to: localhost

- name: 创建{{ user_name }}用户并设置密码，如果用户存在则修改密码
  user:
    name: "{{ user_name }}"
    password: "{{ chapass.stdout | string | password_hash('sha512') }}"
    state: present
    update_password: always
```

```
- name: 存储{{ user_name }}用户密码信息到{{ password_file }}文件
  lineinfile:
    path: "{{ password_file }}"
    regexp: '^{{ inventory_hostname }}:{{ user_name }}:'
    line: "{{ inventory_hostname }}:{{ user_name }}:{{ chapass.stdout }}"
    create: yes
    mode: '0600'
  delegate_to: localhost

- import_tasks: check.yml
```

1.12.2. 任务执行

```
ansible-playbook -i inventory/ -e user_name=weihu -e operation=useradd_v1
useradd_v1.yml  -f 1
```

> 当多个任务几乎同时尝试向同一个文件写入数据时，可能会因ansible默认的并发写入特性从而导致数据丢失或不完整的情况出现，即使任务执行没有任何问题，所以这里通过在命令行添加参数`-f 1`来限制并发数避免该问题。

1.12.3. 配置检查

```
#检查指定用户{{ user_name }}是否存在，同时检查是否配置sudo权限
ansible-playbook -i inventory/ -e user_name=weihu -e operation=check
useradd_v1.yml
```

1.13. 格式化分区磁盘/dev/{{ disk_mount }}并挂载到/data

1.13.1. 任务内容

```
[root@localhost ansible]# cat roles/mountdisk/tasks/config.yml
---
- name: 对/dev/{{ disk_mount }}磁盘进行分区
  shell: |
    parted "/dev/{{ disk_mount }}" << EOF
    mklabel gpt
    yes
    mkpart primary 1 100%
    quit
    EOF
- name: 格式化磁盘/dev/{{ disk_mount }}并挂载到/data
  shell: |
    sleep 15
    mkfs.xfs "/dev/{{ disk_mount }}1"
    mkdir -p /data
    mount "/dev/{{ disk_mount }}1" /data
- name: 配置fstab，写入/dev/{{ disk_mount }}挂载信息
  lineinfile:
    dest: /etc/fstab
```

```
    regexp: "^/dev/{{ disk_mount }}1"
    backrefs: false
    backup: true
    line: "/dev/{{ disk_mount }}1    /data   xfs    defaults        0 0"
```

1.13.2. 任务执行

```
ansible-playbook -i inventory/ -e operation=config mountdisk.yml
```

1.13.3. 配置检查

```
#检查磁盘挂载情况
ansible-playbook -i inventory/ -e operation=check mountdisk.yml
```

1.14. 安装redis单实例并启动服务

1.14.1. 任务内容

```
[root@localhost ansible]# cat roles/redis/tasks/install.yml
- name: 安装yum仓库中最新版本的redis
  dnf:
    name: redis
    state: latest
- import_tasks: start.yml
```

1.14.2. 任务执行

```
ansible-playbook -i inventory/ -e operation=install redis.yml
```

1.14.3. 配置检查

```
#启停redis单实例服务及查看服务状态
ansible-playbook -i inventory/ -e operation=start|stop|status redis.yml
```

2. 使用ansible完成Linux安全基线加固

2.1. 针对root用户在/root/.bashrc文件中为ls命令设置别名ls='ls -al'

2.1.1. 任务内容

```
[root@localhost ansible]# cat roles/alias/tasks/config.yml
- name: 针对root用户在/root/.bashrc文件中为ls命令设置别名ls='ls -al'
  lineinfile:
    path: /root/.bashrc
    regexp: '^alias\s+ls='
    line: "alias ls='ls -al'"
    backup: yes
```

```
  ignore_errors: yes

- import_tasks: check.yml
```

2.1.2. 任务执行

```
ansible-playbook -i inventory/ -e operation=config alias.yml
```

2.1.3. 配置检查

```
#检查针对root用户在/root/.bashrc文件中为ls命令设置别名是否为ls='ls -al'
ansible-playbook -i inventory/ -e operation=check alias.yml
```

2.2. 在/etc/motd、/etc/issue及/etc/issue.net文件中配置系统banner提示信息

2.2.1. 任务内容

```
[root@localhost ansible]# cat roles/banner/tasks/config.yml
- name:  在/etc/motd文件中配置系统banner提示信息
  lineinfile:
    path: /etc/motd
    line: 'Authorized users only. All activity may be monitored and reported'
    backup: yes
  ignore_errors: yes
- name: 在/etc/issue文件中配置系统banner提示信息
  lineinfile:
    path: /etc/issue
    line: 'Authorized users only. All activity may be monitored and reported'
  ignore_errors: yes
- name: 在/etc/issue.net文件中配置系统banner提示信息
  lineinfile:
    path: /etc/issue.net
    line: 'Authorized users only. All activity may be monitored and reported'
  ignore_errors: yes

- import_tasks: check.yml
```

2.2.2. 任务执行

```
ansible-playbook -i inventory/ -e operation=config banner.yml
```

2.2.3. 配置检查

```
#检查在/etc/motd、/etc/issue及/etc/issue.net文件中配置的系统banner提示信息内容
ansible-playbook -i inventory/ -e operation=check banner.yml
```

2.3. 配置chronyd时钟同步服务器，ntp_server为时钟同步服务器地址

2.3.1. 任务内容

```
[root@localhost ansible]# cat roles/chrony/tasks/config.yml
- name: 配置chronyd时钟同步服务器为{{ ntp_server }}
  lineinfile:
    path: /etc/chrony.conf
    state: present
    regexp: '^server {{ ntp_server }} iburst'
    line: "server {{ ntp_server }} iburst"
    backup: yes
  notify: restart chronyd
  ignore_errors: yes

- import_tasks: check.yml
```

2.3.2. 任务执行

```
ansible-playbook -i inventory/ -e ntp_server=192.168.0.11 -e operation=config
chrony.yml
```

2.3.3. 配置检查

```
#检查chronyd时钟同步服务器配置
ansible-playbook -i inventory/ -e operation=check chrony.yml
```

2.4. 在/etc/csh.cshrc文件中设置csh shell 下的自动超时变量autologout为600s

2.4.1. 任务内容

```
[root@localhost ansible]# cat roles/csh/tasks/config.yml
- name: 在/etc/csh.cshrc文件中设置csh shell 下的自动超时变量autologout为600s
  lineinfile:
    path: /etc/csh.cshrc
    insertafter: 'EOF'
    line: "{{ item }}"
    backup: yes
  ignore_errors: yes
  with_items:
    - "set autologout=600"

- import_tasks: check.yml
```

2.4.2. 任务执行

```
ansible-playbook -i inventory/ -e operation=config csh.yml
```

2.4.3. 配置检查

```
#检查/etc/csh.cshrc文件中csh shell的自动超时变量autologout配置
ansible-playbook -i inventory/ -e operation=check csh.yml
```

2.5. 在/etc/host.conf文件中配置主机解析地址的顺序。先使用hosts，再使用BIND（DNS）进行解析

2.5.1. 任务内容

```
[root@localhost ansible]# cat roles/host_conf/tasks/config.yml
- name: 在/etc/host.conf文件中配置主机解析地址的顺序。先使用hosts，再使用BIND（DNS）进行解
析
  lineinfile:
    path: /etc/host.conf
    regexp: '^order'
    state: present
    line: 'order hosts,bind'
    backup: yes
  ignore_errors: yes
- import_tasks: check.yml
```

2.5.2. 任务执行

```
ansible-playbook -i inventory/ -e operation=config host_conf.yml
```

2.5.3. 配置检查

```
#检查在/etc/host.conf文件中配置主机解析地址的顺序配置
ansible-playbook -i inventory/ -e operation=check host_conf.yml
```

2.6. 在/etc/hosts.allow及/etc/hosts.deny文件中定义访问本地服务的远程主机或主机范围地址

2.6.1. 任务内容

```
[root@localhost ansible]# cat roles/hosts_auth/tasks/
check.yml   config.yml  main.yml
[root@localhost ansible]# cat roles/hosts_auth/tasks/config.yml
- name: 在/etc/hosts.allow文件中定义允许访问本地服务的远程主机或主机范围
  lineinfile:
    path: /etc/hosts.allow
    insertafter: 'EOF'
```

```
    line: "{{ item }}"
    backup: yes
  ignore_errors: yes
  loop:
    - 'sshd: all'
    - 'telnetd: all'
  tags: hosts_allow

- name: 在/etc/hosts.deny文件中定义禁止访问本地服务的远程主机或主机范围。
  lineinfile:
    path: /etc/hosts.deny
    insertafter: 'EOF'
    line: "{{ item }}"
    backup: yes
  ignore_errors: yes
  loop:
    - 'sshd: 192.168.182.2'
    - 'telnetd: 192.168.182.2'
  tags: hosts_deny

- import_tasks: check.yml
```

2.6.2. 任务执行

```
ansible-playbook -i inventory/ -e operation=config hosts_auth.yml
```

2.6.3. 配置检查

```
#检查在/etc/hosts.allow及/etc/hosts.deny文件中定义的访问本地服务的远程主机或主机范围地址
ansible-playbook -i inventory/ -e operation=check hosts_auth.yml
```

2.7. 在/etc/login.defs文件中配置

> 1、LASTLOG_ENAB：启用对用户的最后一次登录信息的记录。
>
> 2、FAILLOG_ENAB：启用对用户失败登录尝试的记录

2.7.1. 任务内容

```
[root@localhost ansible]# cat roles/login_defs/tasks/config.yml
- name: 在/etc/login.defs文件中配置1、LASTLOG_ENAB：启用对用户的最后一次登录信息的记录。
2、FAILLOG_ENAB：启用对用户失败登录尝试的记录
  lineinfile:
    path: /etc/login.defs
    insertafter: 'EOF'
    line: "{{ item }}"
    backup: yes
  ignore_errors: yes
  with_items:
    - "LASTLOG_ENAB yes"
    - "FAILLOG_ENAB yes"
- import_tasks: check.yml
```

2.7.2. 任务执行

```
ansible-playbook -i inventory/ -e operation=config login_defs.yml
```

2.7.3. 配置检查

```
#检查在/etc/login.defs文件中定义的LASTLOG_ENAB及FAILLOG_ENAB内容
ansible-playbook -i inventory/ -e operation=check login_defs.yml
```

2.8. 配置lo网卡禁止IP源路由net.ipv4.conf.lo.accept_source_route=0，启用路由转发net.ipv4.ip_forward=1

2.8.1. 任务内容

```
[root@localhost ansible]# cat roles/sysctl/tasks/config.yml
- name: 配置lo网卡禁止IP源路由net.ipv4.conf.lo.accept_source_route=0
  sysctl:
    name: net.ipv4.conf.lo.accept_source_route
    value: '0'
    sysctl_set: yes
    state: present
- name: 配置启用路由转发net.ipv4.ip_forward=1
  sysctl:
    name: net.ipv4.ip_forward
    value: '1'
    sysctl_set: yes
    state: present
    reload: yes
- import_tasks: check.yml
```

2.8.2. 任务执行

```
ansible-playbook -i inventory/ -e operation=config sysctl.yml
```

2.8.3. 配置检查

```
#检查lo网卡net.ipv4.conf.lo.accept_source_route以及net.ipv4.ip_forward路由相关参数配置
ansible-playbook -i inventory/ -e operation=check sysctl.yml
```

2.9. 配置/etc/security文件权限为600

2.9.1. 任务内容

```
[root@localhost ansible]# cat roles/chmod/tasks/config.yml
- name: 配置/etc/security文件权限为600
  file:
    path: "{{ item }}"
    state: directory
```

```
    mode: 0600
  ignore_errors: yes
  with_items:
    - /etc/security
- import_tasks: check.yml
```

2.9.2. 任务执行

```
ansible-playbook -i inventory/ -e operation=config chmod.yml
```

2.9.3. 配置检查

```
#检查/etc/security文件权限
ansible-playbook -i inventory/ -e operation=check chmod.yml
```

2.10. 配置将authpriv类的日志记录到/var/log/authlog文件中

2.10.1. 任务内容

```
[root@localhost ansible]# cat roles/rsyslog/tasks/config.yml
- name: 在/etc/rsyslog.conf文件中配置将authpriv类的日志记录到/var/log/authlog文件中
  lineinfile:
    path: /etc/rsyslog.conf
    insertafter: '^authpriv\.'
    line: 'authpriv.*
/var/log/authlog'
    backup: yes
  ignore_errors: yes
  notify: restart rsyslog
- import_tasks: check.yml
```

2.10.2. 任务执行

```
ansible-playbook -i inventory/ -e operation=config rsyslog.yml
```

2.10.3. 配置检查

```
#检查/etc/rsyslog.conf文件中对authpriv类的日志记录配置情况
ansible-playbook -i inventory/ -e operation=check rsyslog.yml
```

2.11. 配置远程日志服务器，logserver为远程日志服务器地址

2.11.1. 任务内容

```
[root@localhost ansible]# cat roles/logserver/tasks/config.yml
- name: 设置远程日志转发策略
  block:
    - name: 检查是否已经配置了远程日志服务器{{ logserver }}
      command: grep -q "{{ logserver }}" /etc/rsyslog.conf
      register: rsyslog_check
      failed_when: false  # 即使没有找到也不报错

    - name: 在/etc/rsyslog.conf文件中配置远程日志服务器为{{ logserver }}
      lineinfile:
        path: /etc/rsyslog.conf
        line: "*.* @{{ logserver }}"
        insertafter: EOF
        create: yes
        backup: yes
      when: rsyslog_check.rc != 0  # 只有当grep没有找到时才执行
      notify: restart rsyslog

- import_tasks: check.yml
```

2.11.2. 任务执行

```
ansible-playbook -i inventory/ -e logserver=192.168.0.11 -e operation=config
logserver.yml
```

2.11.3. 配置检查

```
#检查/etc/rsyslog.conf文件中是否配置{{ logserver }}为远程日志服务器地址
ansible-playbook -i inventory/ -e logserver=192.168.0.11 -e operation=check
logserver.yml
```

2.12. 在/etc/logrotate.d/目录下，将rsyslog文件重命名为syslog,并在该文件配置可以对日志按大小10M进行切割

2.12.1. 任务内容

```
[root@localhost ansible]# cat roles/logrotate/tasks/config.yml
- name: 在/etc/logrotate.d/目录下，将rsyslog文件重命名为syslog
  shell:
    cmd: mv rsyslog syslog
    chdir: /etc/logrotate.d/
  ignore_errors: yes
- name: 在/etc/logrotate.d/syslog文件中配置可以对日志按大小10M进行切割
  lineinfile:
    path: /etc/logrotate.d/syslog
```

```
    insertafter: '^{'
    line: '    size 10M'
    backup: yes
  ignore_errors: yes
- import_tasks: check.yml
```

2.12.2. 任务执行

```
ansible-playbook -i inventory/ -e operation=config logrotate.yml
```

2.12.3. 配置检查

```
#检查在/etc/logrotate.d/syslog文件中定义配置的日志切割策略
ansible-playbook -i inventory/ -e operation=check logrotate.yml
```

2.13. 在/etc/profile文件中设置命令行界面登录超时时间TMOUT为300s

2.13.1. 任务内容

```
[root@localhost ansible]# cat roles/profile/tasks/config.yml
- name: 在/etc/profile文件中设置命令行界面登录超时时间TMOUT为300s
  block:
    - name: 如果/etc/profile定义了TMOUT内容，则删除
      lineinfile:
        path: /etc/profile
        regexp: '^TMOUT='
        state: absent
        backup: yes
    - name: 如果/etc/profile定义了export TMOUT内容，则删除
      lineinfile:
        path: /etc/profile
        regexp: '^export\s+TMOUT$'
        state: absent
    - name: 在/etc/profile文件中更新或添加一行export TMOUT=300
      lineinfile:
        path: /etc/profile
        regexp: '^export\s+TMOUT='
        line: 'export TMOUT=300'
      notify: source profile
  tags: timeout
- import_tasks: check.yml
```

2.13.2. 任务执行

```
ansible-playbook -i inventory/ -e operation=config profile.yml  --tags=timeout
```

2.13.3. 配置检查

```
#检查在/etc/profile文件中定义配置的会话登录超时时间TMOUT设置
ansible-playbook -i inventory/ -e operation=check profile.yml  --tags=timeout
```

2.14. 在/etc/profile文件中设置用户缺省UMASK为027

2.14.1. 任务内容

```
[root@localhost ansible]# cat roles/profile/tasks/config.yml
- name: 在/etc/profile文件中设置用户缺省UMASK为027
  lineinfile:
    path: /etc/profile
    regexp: '^umask'
    line: 'umask 027'
    backup: yes
  notify: source profile
  tags: umask
- import_tasks: check.yml
```

2.14.2. 任务执行

```
ansible-playbook -i inventory/ -e operation=config profile.yml  --tags=umask
```

2.14.3. 配置检查

```
#检查在/etc/profile文件中定义配置umask设置
ansible-playbook -i inventory/ -e operation=check profile.yml  --tags=umask
```

2.15. 在/etc/profile文件中设置在.bash_history文件中保存命令的记录总数为5条

2.15.1. 任务内容

```
[root@localhost ansible]# cat roles/profile/tasks/config.yml
- name: 在/etc/profile文件中设置在.bash_history文件中保存命令的记录总数为5条
  lineinfile:
    path: /etc/profile
    regexp: '^HISTFILESIZE'
    line: '{{ item }}'
    backup: yes
  notify: source profile
  with_items:
    - HISTFILESIZE=5
  tags: bash_history
- import_tasks: check.yml
```

2.15.2. 任务执行

```
ansible-playbook -i inventory/ -e operation=config profile.yml  --
tags=bash_history
```

2.15.3. 配置检查

```
#检查在/etc/profile文件中定义配置的.bash_history文件中保存命令的记录总数HISTFILESIZE设置
ansible-playbook -i inventory/ -e operation=check profile.yml  --
tags=bash_history
```

2.16. 在/etc/profile文件中设置shell会话中history命令输出的记录总数为5条

2.16.1. 任务内容

```
[root@localhost ansible]# cat roles/profile/tasks/config.yml
- name: 在/etc/profile文件中设置shell会话中history命令输出的记录总数为5条
  lineinfile:
    path: /etc/profile
    regexp: '^HISTSIZE'
    line: '{{ item }}'
    backup: yes
  notify: source profile
  with_items:
    - HISTSIZE=5
  tags: history
- import_tasks: check.yml
```

2.16.2. 任务执行

```
ansible-playbook -i inventory/ -e operation=config profile.yml --tags=history
```

2.16.3. 配置检查

```
#检查在/etc/profile文件中定义配置的shell会话中history命令输出的记录总数设置
ansible-playbook -i inventory/ -e operation=check profile.yml --tags=history
```

2.17. /etc/security/limits.conf文件中配置core文件大小限制（* soft core 0和* hard core 0）

2.17.1. 任务内容

```
[root@localhost ansible]# cat roles/ulimit/tasks/config.yml
- name: 在/etc/security/limits.conf文件中配置core文件大小限制（* soft core 0和* hard
core 0）
  pam_limits:
    domain: '*'
    limit_type: '{{ item.type }}'
```

```
    limit_item: core
    value: 0
    dest: /etc/security/limits.conf
    backup: yes
  with_items:
    - { type: 'soft' }
    - { type: 'hard' }
- import_tasks: check.yml
```

2.17.2. 任务执行

```
ansible-playbook -i inventory/ -e operation=config ulimit.yml
```

2.17.3. 配置检查

```
#检查在/etc/security/limits.conf文件中定义的core文件的限制情况
ansible-playbook -i inventory/ -e operation=check ulimit.yml
```

2.18. 创建安全事件日志接收目录及文件/var/adm/messages并在/etc/rsyslog.conf文件中配置*.err;kern.debug;daemon.notice类的日志记录到/var/adm/messages文件

2.18.1. 任务内容

```
[root@localhost ansible]# cat roles/messages/tasks/config.yml
- name: 创建安全事件日志接收目录及文件/var/adm/messages并在/etc/rsyslog.conf文件中配置
*.err;kern.debug;daemon.notice类的日志记录到/var/adm/messages文件
  block:
    - name: 创建安全事件日志接收目录及文件/var/adm/messages
      file:
        path: /var/adm/messages
        state: touch
        mode: 0640

    - name: 在/etc/rsyslog.conf文件中配置*.err;kern.debug;daemon.notice类的日志记录
到/var/adm/messages文件
      lineinfile:
        path: /etc/rsyslog.conf
        insertafter: 'EOF'
        line: '*.err;kern.debug;daemon.notice /var/adm/messages'
        backup: yes
      notify: restart rsyslog
      ignore_errors: yes
  tags: adm_messages
- import_tasks: check.yml
```

2.18.2. 任务执行

```
ansible-playbook -i inventory/ -e operation=config messages.yml --
tags=adm_messages
```

2.18.3. 配置检查

```
#检查/etc/rsyslog.conf文件中关于messages文件定义的日志记录设置
ansible-playbook -i inventory/ -e operation=check messages.yml --
tags=adm_messages
```

2.19. 设置关键文件的属性，配置/var/log/messages文件只可追加不可修改

2.19.1. 任务内容

```
[root@localhost ansible]# cat roles/messages/tasks/config.yml
- name: 设置关键文件的属性，配置/var/log/messages文件只可追加不可修改
  shell: chattr +a /var/log/messages
  ignore_errors: yes
  tags: chattr_messages

- import_tasks: check.yml
```

2.19.2. 任务执行

```
ansible-playbook -i inventory/ -e operation=config messages.yml --
tags=chattr_messages
```

2.19.3. 配置检查

```
#检查/var/log/messages文件隐藏权限设置情况
ansible-playbook -i inventory/ -e operation=check messages.yml --
tags=chattr_messages
```

2.20. 配置ntp时钟同步服务器，ntp_server为时钟同步服务器地址

2.20.1. 任务内容

```
[root@localhost ansible]# cat roles/ntp/tasks/config.yml
- name: 安装ntp
  dnf:
    name: ntp
    state: latest

- name: 启动ntp服务并设置开机自启
  service: name=ntpd state=started enabled=yes
  ignore_errors: yes
```

```
- name: 配置ntp服务器为{{ ntp_server }}
  lineinfile:
    path: /etc/ntp.conf
    state: present
    regexp: '^server {{ ntp_server }}'
    line: "server {{ ntp_server }}"
    backup: yes
  notify: restart ntpd
  ignore_errors: yes
- import_tasks: check.yml
```

2.20.2. 任务执行

```
ansible-playbook -i inventory/ -e ntp_server=192.168.0.11 -e operation=config
ntp.yml
```

2.20.3. 配置检查

```
#检查ntp时钟同步服务器配置
ansible-playbook -i inventory/ -e operation=check ntp.yml
```

2.21. 配置限制除wheel组以外的用户通过su命令切换到root

2.21.1. 任务内容

```
[root@localhost ansible]# cat roles/pam_su/tasks/config.yml
- name: 配置限制除wheel组以外的用户通过su命令切换到root
  lineinfile:
    path: /etc/pam.d/su
    regexp: '^auth\s*required\s*pam_wheel.so\suse_uid'
    line: 'auth            required        pam_wheel.so use_uid'
    backup: yes
  ignore_errors: yes
- import_tasks: check.yml
```

2.21.2. 任务执行

```
ansible-playbook -i inventory/ -e operation=config pam_su.yml
```

2.21.3. 配置检查

```
#检查/etc/pam.d/su文件认证配置情况
ansible-playbook -i inventory/ -e operation=config pam_su.yml
```

2.22. /etc/pam.d/passwd配置使用pam_pwquality.so模块，并在/etc/security/pwquality.conf配置口令复杂度（大小写数字特殊字符至少包含一个）

2.22.1. 任务内容

```
[root@localhost ansible]# cat roles/pam_passwd/tasks/config.yml
- name: 在/etc/pam.d/passwd配置使用pam_pwquality.so模块，并
在/etc/security/pwquality.conf配置口令复杂度
  block:
    - name: 在/etc/pam.d/passwd配置使用pam_pwquality.so模块
      lineinfile:
        path: /etc/pam.d/passwd
        insertafter: EOF
        line: 'password required pam_pwquality.so retry=3'
        backup: yes
      ignore_errors: yes
      tags: pam_pwquality
    - name: 在/etc/security/pwquality.conf配置口令复杂度（小写lcredit、大写ucredit、数
字dcredit、特殊字符ocredit）
      lineinfile:
        path: /etc/security/pwquality.conf
        regexp: "{{ item.regexp }}"
        line: "{{ item.line }}"
        backup: yes
      loop:
        - {regexp: '^lcredit', line: 'lcredit = 1'}
        - {regexp: '^ucredit', line: 'ucredit = 1'}
        - {regexp: '^dcredit', line: 'dcredit = 1'}
        - {regexp: '^ocredit', line: 'ocredit = 1'}
      ignore_errors: yes
      tags: password
- import_tasks: check.yml
```

2.22.2. 任务执行

```
ansible-playbook -i inventory/ -e operation=config pam_passwd.yml
```

2.22.3. 配置检查

```
#检查在/etc/pam.d/passwd是否配置使用pam_pwquality.so模块以
及/etc/security/pwquality.conf关于口令复杂度（小写lcredit、大写ucredit、数字dcredit、特
殊字符ocredit）配置情况
ansible-playbook -i inventory/ -e operation=check pam_passwd.yml
```

2.23. 在/etc/login.defs文件中配置口令生存周期最长PASS_MAX_DAYS为90天，最小PASS_MIN_DAYS为10天及密码最小长度PASS_MIN_LEN为8

2.23.1. 任务内容

```
[root@localhost ansible]# cat roles/password/tasks/config.yml
- name: 在/etc/login.defs文件中配置口令生存周期最长为90天，最小为10天及密码最小长度为8
  lineinfile:
    path: /etc/login.defs
    regexp: "{{ item.regexp }}"
    line: "{{ item.line }}"
    backrefs: no
    backup: yes
  ignore_errors: yes
  with_items:
    - { regexp: '^PASS_MIN_LEN',  line: 'PASS_MIN_LEN    8' }
    - { regexp: '^PASS_MAX_DAYS', line: 'PASS_MAX_DAYS   90' }
    - { regexp: '^PASS_MIN_DAYS', line: 'PASS_MIN_DAYS   10' }
- import_tasks: check.yml
```

2.23.2. 任务执行

```
ansible-playbook -i inventory/ -e operation=config password.yml
```

2.23.3. 配置检查

```
#检查在/etc/login.defs文件中关于口令生存周期最长PASS_MAX_DAYS，最小PASS_MIN_DAYS及密码最小长度PASS_MIN_LEN配置情况
ansible-playbook -i inventory/ -e operation=check password.yml
```

2.24. 在/etc/pam.d/password-auth及/etc/pam.d/system-auth配置口令锁定策略,连续登录失败3次锁定账号

2.24.1. 任务内容

```
[root@localhost ansible]# cat roles/pam_auth/tasks/deny.yml
- name: system-auth配置口令锁定策略,连续登录失败3次锁定账号
  lineinfile:
    path: /etc/pam.d/system-auth
    regexp: '^auth\s*required\s*pam_faillock.so'
    line: 'auth        required      pam_faillock.so preauth audit deny=3
even_deny_root unlock_time=60'
    backup: yes
  ignore_errors: yes
- name: system-auth配置口令锁定策略,连续登录失败3次锁定账号
  lineinfile:
```

```
    path: /etc/pam.d/system-auth
    regexp: '^auth\s*\[default=die\]\s*pam_faillock.so'
    line: 'auth        [default=die] pam_faillock.so authfail audit deny=3
even_deny_root unlock_time=60'
    backup: yes
  ignore_errors: yes
- name: system-auth配置口令锁定策略,连续登录失败3次锁定账号
  lineinfile:
    path: /etc/pam.d/system-auth
    regexp: '^auth\s*sufficient\s*pam_faillock.so'
    line: 'auth        sufficient    pam_faillock.so authsucc audit deny=3
even_deny_root unlock_time=60'
    backup: yes
  ignore_errors: yes

- name: password-auth配置口令锁定策略,连续登录失败3次锁定账号
  lineinfile:
    path: /etc/pam.d/password-auth
    regexp: '^auth\s*required\s*pam_faillock.so'
    line: 'auth        required      pam_faillock.so preauth audit deny=3
even_deny_root unlock_time=60'
    backup: yes
  ignore_errors: yes
- name: password-auth配置口令锁定策略,连续登录失败3次锁定账号
  lineinfile:
    path: /etc/pam.d/password-auth
    regexp: '^auth\s*\[default=die\]\s*pam_faillock.so'
    line: 'auth        [default=die] pam_faillock.so authfail audit deny=3
even_deny_root unlock_time=60'
    backup: yes
  ignore_errors: yes
- name: password-auth配置口令锁定策略,连续登录失败3次锁定账号
  lineinfile:
    path: /etc/pam.d/password-auth
    regexp: '^auth\s*sufficient\s*pam_faillock.so'
    line: 'auth        sufficient    pam_faillock.so authsucc audit deny=3
even_deny_root unlock_time=60'
    backup: yes
  ignore_errors: yes

- import_tasks: check.yml
```

2.24.2. 任务执行

```
ansible-playbook -i inventory/ -e operation=deny pam_auth.yml
```

2.24.3. 配置检查

```
#检查在/etc/pam.d/password-auth及/etc/pam.d/system-auth文件中的配置认证情况
ansible-playbook -i inventory/ -e operation=check pam_auth.yml
```

2.25. 在/etc/pam.d/password-auth及/etc/pam.d/system-auth配置口令复杂度（大小写数字特殊字符至少包含一个）并限制到root

2.25.1. 任务内容

```
[root@localhost ansible]# cat roles/pam_auth/tasks/login-auth.yml
- name: system-auth文件中配置口令复杂度并限制到root
  lineinfile:
    path: /etc/pam.d/system-auth
    regexp: '^password\s+requisite\s+pam_pwquality.so'
    line: 'password    requisite     pam_pwquality.so minlen=8 minclass=3
enforce_for_root try_first_pass local_users_only retry=3 dcredit=1 ucredit=1
lcredit=1 ocredit=1'
    backup: yes
  ignore_errors: yes
- name: password-auth文件中配置口令复杂度并限制到root
  lineinfile:
    path: /etc/pam.d/password-auth
    regexp: '^password\s+requisite\s+pam_pwquality.so'
    line: 'password    requisite     pam_pwquality.so minlen=8 minclass=3
enforce_for_root try_first_pass local_users_only retry=3 dcredit=1 ucredit=1
lcredit=1 ocredit=1'
    backup: yes
  ignore_errors: yes
- import_tasks: check.yml
```

2.25.2. 任务执行

```
ansible-playbook -i inventory/ -e operation=login-auth pam_auth.yml
```

2.25.3. 配置检查

```
#检查在/etc/pam.d/password-auth及/etc/pam.d/system-auth文件中的配置认证情况
ansible-playbook -i inventory/ -e operation=check pam_auth.yml
```

2.26. 在/etc/pam.d/password-auth及/etc/pam.d/system-auth配置口令重复次数限制为5并限制到root

2.26.1. 任务内容

```
[root@localhost ansible]# cat roles/pam_auth/tasks/remember.yml
- name: system-auth文件中配置口令重复次数限制为5并限制到root
  lineinfile:
    path: /etc/pam.d/system-auth
    insertafter: '^password\s+requisite\s+pam_pwquality.so'
    line: 'password    required      pam_pwhistory.so use_authtok remember=5
enforce_for_root'
    backup: yes
```

```
  ignore_errors: yes
- name: password-auth文件中配置口令重复次数限制为5并限制到root
  lineinfile:
    path: /etc/pam.d/password-auth
    insertafter: '^password\s+requisite\s+pam_pwquality.so'
    line: 'password    required      pam_pwhistory.so use_authtok remember=5
enforce_for_root'
    backup: yes
  ignore_errors: yes
- name: system-auth文件中password    sufficient    pam_unix.so行配置口令重复次数限制为
5
  lineinfile:
    path: /etc/pam.d/system-auth
    regexp: '^password\s+sufficient'
    line: 'password    sufficient    pam_unix.so sha512 shadow nullok
try_first_pass use_authtok remember=5'
    backup: yes
  ignore_errors: yes
- import_tasks: check.yml
```

2.26.2. 任务执行

```
ansible-playbook -i inventory/ -e operation=remember pam_auth.yml
```

2.26.3. 配置检查

```
#检查在/etc/pam.d/password-auth及/etc/pam.d/system-auth文件中的配置认证情况
ansible-playbook -i inventory/ -e operation=check pam_auth.yml
```

2.27. 设置系统相关用户shell为/bin/false并进行锁定

2.27.1. 任务内容

```
[root@localhost ansible]# cat roles/lock_user/tasks/config.yml
- name: 设置lp|sync|halt|operator|games|nobody系统相关用户shell为/bin/false
  user:
    name: "{{ item }}"
    shell: /bin/false
  ignore_errors: yes
  with_items:
    - lp
    - sync
    - halt
    - operator
    - games
    - nobody
- name: 锁定lp|sync|halt|operator|games|nobody系统相关用户
  shell: /sbin/usermod -L {{ item }}
  ignore_errors: yes
  with_items:
    - lp
    - sync
```

```
    - halt
    - operator
    - games
    - nobody
- import_tasks: check.yml
```

2.27.2. 任务执行

```
ansible-playbook -i inventory/ -e operation=config lock_user.yml
```

2.27.3. 配置检查

```
#检查系统相关用户shell设置情况以及用户锁定状态
ansible-playbook -i inventory/ -e operation=check lock_user.yml
```

2.28. 配置ssh登录前警告Banner内容

2.28.1. 任务内容

```
[root@localhost ansible]# cat roles/ssh/tasks/ssh_banner.yml
- name: 创建/etc/ssh_banner文件，设置ssh登录前警告Banner内容
  copy:
    content: 'Authorized users only. All activity may be monitored and reported'
    dest: /etc/ssh_banner
    mode: '0644'
    owner: bin
    group: bin
    backup: yes
  ignore_errors: yes
  tags: ssh_banner

- name: 在/etc/ssh/sshd_config配置文件中应用/etc/ssh_banner配置
  lineinfile:
    path: /etc/ssh/sshd_config
    state: present
    regexp: '^Banner\s'
    line: 'Banner /etc/ssh_banner'
    backup: yes
  ignore_errors: yes
  notify: restart sshd
  tags: ssh_banner
- import_tasks: check.yml
```

2.28.2. 任务执行

```
ansible-playbook -i inventory/ -e operation=ssh_banner ssh.yml  --tags=ssh_banner
```

2.28.3. 配置检查

```
#检查ssh登录前警告Banner内容
ansible-playbook -i inventory/ -e operation=check ssh.yml --tags=ssh_banner
```

2.29. 配置禁止root用户通过SSH进行远程登录

2.29.1. 任务内容

```
[root@localhost ansible]# cat roles/ssh/tasks/ssh_config.yml
- name: 配置禁止root用户通过SSH进行远程登录
  lineinfile:
    path: /etc/ssh/sshd_config
    regexp: '^PermitRootLogin'
    line: 'PermitRootLogin no'
    backup: yes
  notify: restart sshd
  ignore_errors: yes
  tags: PermitRootLogin
- import_tasks: check.yml
```

2.29.2. 任务执行

```
ansible-playbook -i inventory/ -e operation=ssh_config ssh.yml --
tags=PermitRootLogin
```

2.29.3. 配置检查

```
#检查root用户远程登录限制情况
ansible-playbook -i inventory/ -e operation=check ssh.yml --tags=PermitRootLogin
```

3. 使用ansible完成Nginx、MySQL主从、Redis Cluster、MongoDB分片集群一键部署

3.1. 安装nginx应用并启动服务

3.1.1. 任务内容

```
[root@localhost ansible]# cat roles/nginx/tasks/config.yml
---
- name: 安装nginx
  shell: yum install nginx -y

- name: 下发nginx配置文件
  template:
    src: nginx.conf
    dest: /etc/nginx/
    mode: 0644
  notify:
```

```
    - reload nginx

- import_tasks: start.yml
```

3.1.2. 任务执行

```
ansible-playbook -i inventory/ -e operation=config nginx.yml
```

3.1.3. 配置检查

```
#针对nginx服务进行管理（服务启停及当前状态的检查）
ansible-playbook -i inventory/ -e operation=start|stop|status nginx.yml
```

3.2. 为nginx用途的节点安装keepalived服务并托管vip

3.2.1. 任务内容

```
[root@localhost ansible]# cat roles/keepalived/tasks/keepalived-nginx.yml
---
- name: 安装keepalived
  shell: yum install -y keepalived

- name: 获取网卡名
  shell: ip add | grep {{ inventory_hostname }}| awk '{print $NF}'
  register: interface
  tags: config

- name: 复制sysconfig文件
  template:
    src: sysconfig/keepalived
    dest: /etc/sysconfig/
    mode: 0644

- name: 复制keepalived配置文件
  template:
    src: keepalived.conf
    dest: /etc/keepalived/
    mode: 0644
  tags: config
  notify: reload keepalived

- name: 复制nginx检查脚本
  copy:
    src: check_nginx.sh
    dest: /etc/keepalived/check_nginx.sh
    mode: '0755'
  tags: nginx

- import_tasks: start.yml
- import_tasks: status.yml
```

3.2.2. 任务执行

```
ansible-playbook -i inventory/ -e operation=keepalived-nginx keepalived.yml
```

3.2.3. 配置检查

```
#检查keepalived服务状态以及vip托管情况
ansible-playbook -i inventory/ -e operation=status keepalived.yml
```

3.3. 为mysql用途的节点安装keepalived服务并托管vip

3.3.1. 任务内容

```
[root@localhost ansible]# cat roles/keepalived/tasks/keepalived-mysql.yml
---
- name: 安装keepalived
  shell: yum install -y keepalived

- name: 获取网卡名
  shell: ip add | grep {{ inventory_hostname }}| awk '{print $NF}'
  register: interface
  tags: config

- name: 复制sysconfig文件
  template:
    src: sysconfig/keepalived
    dest: /etc/sysconfig/
    mode: 0644

- name: 复制keepalived配置文件
  template:
    src: keepalived.conf
    dest: /etc/keepalived/
    mode: 0644
  tags: config
  notify: reload keepalived

- name: 复制mysql检查脚本
  copy:
    src: check_mysql.sh
    dest: /etc/keepalived/check_mysql.sh
    mode: '0755'
  tags: mysql

- import_tasks: start.yml
- import_tasks: status.yml
```

3.3.2. 任务执行

```
ansible-playbook -i inventory/ -e operation=keepalived-mysql keepalived.yml
```

3.3.3. 配置检查

```
#检查keepalived服务状态以及vip托管情况
ansible-playbook -i inventory/ -e operation=status keepalived.yml
```

3.4. 搭建mysql主从环境

3.4.1. 任务内容-安装mysql实例

```
[root@localhost ansible]# cat roles/mysql/tasks/install.yml
---

- name: 创建mysql数据目录/data/mysql{data,log,tmp,binlog,install}
  file:
    path: "{{ item }}"
    mode: '0755'
    owner: mysql
    group: mysql
    state: directory
  with_items:
    - /data/mysql
    - /data/mysql/data
    - /data/mysql/log
    - /data/mysql/tmp
    - /data/mysql/binlog
    - /data/mysql/install

- name: 检查mysql安装包是否存在
  stat:
    path: /data/mysql/mysql.tar
  register: mysql_package

- name: mysql安装包不存在，下载mysql软件包
  shell: wget https://downloads.mysql.com/archives/get/p/23/file/mysql-8.0.37-
1.el8.x86_64.rpm-bundle.tar -O /data/mysql/mysql.tar
  when: not mysql_package.stat.exists

- name: 解压mysql包
  unarchive:
    src: /data/mysql/mysql.tar
    dest: /data/mysql/install
    remote_src: yes

- name: 安装mysql
  shell: |
    cd /data/mysql/install/;yum install * -y

- name: 下发gtid模式的mysql配置文件my.cnf到/etc/目录
  template:
```

```
    src: my_gtid.cnf
    dest: /etc/my.cnf
    mode: 0644
  when: replication_mode=='gtid'

- name: 下发基于点位模式的mysql配置文件my.cnf到/etc/目录
  template:
    src: my_position.cnf
    dest: /etc/my.cnf
    mode: 0644
  when: replication_mode=='position'

- name: 初始化mysql数据库
  shell: mysqld --initialize

- import_tasks: start.yml

- name: 从日志中获取mysql数据库初始化密码
  shell: cat /data/mysql/log/mysqld.log |grep localhost|grep "temporary
password"|awk '{print $NF}'
  register: mysql_init_passwd

- name: 显示mysql数据库初始化密码
  debug:
    msg: "{{ mysql_init_passwd.stdout }}"

- name: 修改mysql数据库root用户密码
  shell: mysqladmin -u{{ mysql_user }}  -p'{{mysql_init_passwd.stdout}}' -S
/data/mysql/tmp/mysql.sock password '{{ mysql_passwd }}'

- import_tasks: master_slave.yml
```

3.4.2. 任务内容-构建主从关系

```
[root@localhost ansible]# cat roles/mysql/tasks/master_slave.yml
---
- name: 创建主从同步账号
  when: master is defined
  shell: |
      mysql -uroot -p{{ mysql_passwd }} -S /data/mysql/tmp/mysql.sock -e "CREATE
USER '{{ mysql_repl_user }}'@'%' IDENTIFIED BY '{{ mysql_repl_passwd }}';"
      mysql -uroot -p{{ mysql_passwd }} -S /data/mysql/tmp/mysql.sock -e "grant
REPLICATION SLAVE, REPLICATION CLIENT on *.* to '{{ mysql_repl_user }}'@'%';"
      mysql -uroot -p{{ mysql_passwd }} -S /data/mysql/tmp/mysql.sock -e "flush
privileges;"

- name: 查看master的binlog日志名和position位置信息
  block:
    - name: 查看master的binlog日志名
      shell: mysql -u{{ mysql_repl_user }} -p{{ mysql_repl_passwd }} -h '{{
play_hosts[0] }}' -e 'show master status ;' 2>/dev/null |grep binlog |awk '{print
$1}'
      register: master_bin_log

    - name: 查看master的position位置
```

```
      shell: mysql -u{{ mysql_repl_user }} -p{{ mysql_repl_passwd }} -h '{{
play_hosts[0] }}' -e 'show master status ;' 2>/dev/null |grep binlog |awk '{print
$2}'
      register: master_position

    - name: 打印master的binlog日志名和position位置信息
      debug:
        msg: "Master binlog file: {{ master_bin_log.stdout }}, Position: {{
master_position.stdout }}"

  rescue:
    - name: 错误处理
      debug:
        msg: "获取master的binlog日志名和position位置信息失败！！！"
  when: replication_mode=='position' and slave is defined

- name: 基于position模式创建slave与master的主从同步
  when: replication_mode=='position' and slave is defined
  shell: mysql -u{{ mysql_user }} -p{{ mysql_passwd }} -S
/data/mysql/tmp/mysql.sock -e "
    CHANGE MASTER TO MASTER_HOST='{{ play_hosts[0] }}',
    MASTER_PORT={{ mysql_port }},
    MASTER_USER='{{ mysql_repl_user }}',
    MASTER_PASSWORD='{{ mysql_repl_passwd }}',
    MASTER_LOG_FILE='{{ master_bin_log.stdout }}',
    MASTER_LOG_POS={{ master_position.stdout }};"

- name: 基于gtid模式创建slave与master的主从同步
  shell: mysql -u{{ mysql_user }} -p{{ mysql_passwd }} -S
/data/mysql/tmp/mysql.sock -e "
    CHANGE MASTER TO MASTER_HOST='{{ play_hosts[0] }}',
    MASTER_PORT={{ mysql_port }},
    MASTER_USER='{{ mysql_repl_user }}',
    MASTER_PASSWORD='{{ mysql_repl_passwd }}',
    master_auto_position=1,
    get_master_public_key=1;"
  when: replication_mode=='gtid' and slave is defined

- name: 启动slave
  when: slave is defined
  shell: |
    mysql -u{{ mysql_user }} -p{{ mysql_passwd }} -S /data/mysql/tmp/mysql.sock -
e "start slave;"
    sleep 10;

- import_tasks: show_slave_status.yml
```

3.4.3. 任务执行

```
#基于GTID模式搭建主从环境
ansible-playbook -i inventory/ -e replication_mode=gtid -e operation=install
mysql.yml
#基于position点位方式搭建主从环境
ansible-playbook -i inventory/ -e replication_mode=position -e operation=install
mysql.yml
```

3.4.4. 配置检查

```
#检查mysql主从环境同步情况,对mysql服务进行管理（服务启停及当前状态的检查）
ansible-playbook -i inventory/ -e operation=show_slave_status|start|stop|status
mysql.yml
```

3.5. 部署3节点3主3从redis cluster集群（服务端口：6379、6389）

3.5.1. 任务内容

```
[root@localhost ansible]# cat roles/redis_cluster/tasks/create.yml
---
- name: 创建数据目录/data/redis/
  file:
    path: '/data/redis/{{item.role}}/log'
    state: 'directory'
  with_items:
    - { role: 'master' }
    - { role: 'slave' }

- name: 创建配置文件目录/opt/redis/cluster-conf/
  file:
    path: '/opt/redis/cluster-conf/{{item.port}}'
    state: 'directory'
  with_items:
    - { port: '6379' }
    - { port: '6389' }

- name: 创建解压目录/tmp/redis
  file:
    path: '/tmp/redis'
    state: 'directory'

- name: 解压压缩包
  unarchive:
     src: 'files/redis-7.0.0.tar.gz'
     dest: '/tmp/redis'
- name: 安装gcc、make
  shell: yum install gcc make -y
  ignore_errors: true

- name: 编译安装redis到/opt/redis目录
```

```
  shell: cd /tmp/redis/redis-7.0.0 && make install PREFIX=/opt/redis

- name: 检查环境变量
  shell: grep redis /etc/profile
  ignore_errors: true
  register: redispath

- name: /etc/profile添加redis环境变量export PATH=$PATH:/opt/redis/bin/
  when: redispath.stdout == ""
  shell: |
    echo 'export PATH=$PATH:/opt/redis/bin/' >> /etc/profile

- import_tasks: config.yml

- import_tasks: start.yml

- name: redis实例启动中
  shell: sleep 5

- import_tasks: cluster.yml

- name: redis集群启动中
  shell: sleep 30

- import_tasks: cluster_status.yml
```

3.5.2. 任务执行

```
ansible-playbook -i inventory/ -e operation=create redis_cluster.yml
```

3.5.3. 配置检查

```
#针对3节点的redis cluster集群（3主：6379端口，3从：6389端口），进行服务的启停，节点及集群的状态检查
ansible-playbook -i inventory/ -e operation=start|stop|status|cluster_status redis_cluster.yml
```

3.6. 部署3节点的MongoDB分片集群

3.6.1. 任务内容

```
[root@localhost ansible]# cat roles/mongodb_shard/tasks/install.yml
---
- name: 下发mongodb安装依赖包compat-openssl
  copy:
    src: compat-openssl10-1.0.2o-4.el8_6.x86_64.rpm
    dest: /opt/

- name: 安装mongodb依赖包compat-openssl
  shell: yum install -y /opt/compat-openssl10-1.0.2o-4.el8_6.x86_64.rpm

- name: 下发mongodb安装包
```

```
  copy:
    src: mongodb-linux-x86_64-rhel70-4.4.20.tgz
    dest: /opt/

- name: 解压mongodb包
  unarchive:
    src: /opt/mongodb-linux-x86_64-rhel70-4.4.20.tgz
    dest: /opt/
    remote_src: yes

- name: 目录改名
  shell: mv /opt/mongodb-linux-x86_64-rhel70-4.4.20 /opt/mongodb
  ignore_errors: true

- name: 新建configserver目录
  when: configserver is defined
  file:
    path: /data/mongodb/configserver/{{ item }}
    state: directory
    mode: '0755'
  with_items:
    - data
    - log
    - conf

- name: 新建mongos目录
  when: mongos is defined
  file:
    path: /data/mongodb/mongos/{{ item }}
    state: directory
    mode: '0755'
  with_items:
    - data
    - log
    - conf

- name: 新建shard目录
  when: shardserver is defined
  file:
    path: /data/mongodb/{{ item }}
    state: directory
    mode: '0755'
  with_items:
    - /shard1/data
    - /shard1/conf
    - /shard1/log
    - /shard2/log
    - /shard2/data
    - /shard2/conf
    - /shard3/data
    - /shard3/conf
    - /shard3/log

- name: 关闭内存大页&配置环境变量
  shell: |
    #!/bin/bash
```

```
    echo never > /sys/kernel/mm/transparent_hugepage/enabled
    echo never > /sys/kernel/mm/transparent_hugepage/defrag
    echo "export PATH=\$PATH:/opt/mongodb/bin/" >> /etc/profile
    source /etc/profile

- name: key下发
  copy:
    src: mongo.key
    dest: /opt/mongodb/
    mode: '0400'

- import_tasks: config.yml
```

3.6.2. 任务执行

```
ansible-playbook -i inventory/ -e operation=install mongodb_shard.yml
```

3.6.3. 配置检查

```
#针对mongodb分配集群，进行服务的启停，节点及集群的状态检查
ansible-playbook -i inventory/ -e operation=start|stop|status|cluster_status
mongodb_shard.yml
```