作者@@老马

整理时间：2023-12

RHCE9 考试

请查阅以下重要配置信息，以了解考试环境相关的信息。

考试时间4小时。

请注意，考试过程中不能与其他考生交流，您也不得连接到其他考生的主机，测验系统及其网络会受到监控，不当使用可导致成绩判为零分。

请执行下方列出的任务。在开始操作前，您不妨先通读整个列表。这些项目将使用一个分数来评分，这些任务的总分为 300 分，您必须达到 210 分或以上才能获得认证。

重要配置信息

在考试期间，除了您就坐位置的台式机之外，还将使用多个虚拟系统。您不具有台式机系统的 `root` 访问权，但具有对虚拟系统的完整 `root` 访问权。

系统信息

在本考试期间，您将操作下列虚拟系统：

系统	IP地址	角色
bastion.lab.example.com	172.25.250.254	Ansible control node
workstation.lab.example.com	172.25.250.9	Ansible managed node
servera.lab.example.com	172.25.250.10	Ansible managed node
serverb.lab.example.com	172.25.250.11	Ansible managed node
serverc.lab.example.com	172.25.250.12	Ansible managed node
serverd.lab.example.com	172.25.250.13	Ansible managed node

这些系统的 IP 地址采用静态设置。请勿更改这些设置。主机名称解析已配置为解析上方列出的完全限定主机名，同时也解析主机短名称。

帐户信息

- 所有系统的 `root` 密码是 `redhat`。请勿更改 `root` 密码。
- 除非另有指定，否则这将是用于访问其他系统和服务的密码。
- 除非另有指定，否则此密码也应用于您创建的所有帐户或者任何需要设置密码的服务。
- 为方便起见，所有系统上已预装了 SSH 密钥，允许在不输入密码的前提下通过 SSH 进行访问。请勿对系统上的 SSH 配置文件进行任何修改。
- Ansible 控制节点上已创建了用户帐户 `devops`。此帐户预装了 SSH 密钥，允许在 Ansible 控制节点和各个 Ansible 受管节点之间进行 SSH 登录。请勿对系统上的 SSH 配置文件进行任何修改。

其他信息

- 一些考试项目可能需要修改 Ansible 主机清单。您要负责确保所有以前的清单组和项目保留下来，与任何其他更改共存。您还要有确保清单中所有默认的组和主机保留您进行的任何更改。
- 考试系统上的**防火墙默认为不启用**，SELinux 则处于**强制模式**。
- 如果需要安装其他软件，您的物理系统和 Ansible 控制节点可能已设置为指向 content 上的下述存储库：
  - http://content.example.com/rhel9.0/x86_64/dvd/BaseOS
  - http://content.example.com/rhel9.0/x86_64/dvd/AppStream
- 一些项目需要额外的文件，这些文件已在以下位置提供：http://materials.example.com/laoma/
- 产品文档可从以下位置找到：http://materilas.example.com/docs/
- 容器镜像仓库信息：`utility.lab.example.com`
  - 账号：`admin`
  - 密码：`redhat`
- 其他资源也进行了配置，供您在考试期间使用。关于这些资源的具体信息将在需要这些资源的项目中提供。

重要评测信息

除非另有指定，否则您的所有工作（包括 Ansible playbook、配置文件和主机清单等）应当保存在控制节点上的目录 **/home/devops/ansible** 中，并且应当归 `devops` 用户所有。所有 Ansible 相关的命令应当由 `devops` 用户从 Ansible 控制节点上的这个目录运行。

请注意，在评分之前，您的 Ansible 受管节点系统将重置为考试开始时的初始状态，您编写的 Ansible playbook 将通过以 `devops` 用户身份从控制节点上的目录 `/home/devops/ansible` 目录运行来应用。在 playbook 运行后，系统会对您的受管节点进行评估，以判断它们是否按照规定进行了配置。

虚拟系统管理

考试期间，您可以随时关闭或重新引导虚拟机系统。您可以从虚拟系统本身进行这项操作，也可以从物理系统控制虚拟系统。要从物理系统访问或控制考试系统，单击桌面上VM控制台图标。这会显示一个表格，包含每个虚拟机系统的对应按钮，单击特定虚拟机系统的按钮将弹出一个菜单，包含用来控制该系统的选项：

- **启动节点VM**：如果指定的虚拟系统未在运行，该选项将启动指定系统。如果系统已经在运行-则该选项无任何作用。
- **重新引导节点VM**：正常关闭考试虚拟系统，然后重启。
- **关闭节点VM**：正常关闭指定虚拟系统。
- **关闭节点VM电源**：立即关闭指定虚拟系统。
- **控制台**：这将打开一个窗口，用于连接到指定虚拟系统的控制台。请注意，如果将焦点移动到此窗口，控制台将抓住您的鼠标。要恢复鼠标，同时键入`Ctrl+Alt`。
- **重建节点VM**：将当前 VM 还原为原始状态。系统将弹出一个单独的窗口，要求您确认操作。

  警告！！！ 您在 VM 上完成的所有操作都将丢失。仅当系统无法使用时才应使用这个功能。在使用这个功能之前，确保关闭 VM。

考试小技巧

1. 仔细检查鼠标和键盘，特别是鼠标右键和中间。
2. 尽可能使用 `tab` 键补全。
3. 考试过程中用到的名称、路径等字符串，使用复制粘贴。
4. 保持终端清洁，每做完一个题目，执行 `clear` 命令或使用快捷键 `CTRL+L` 清屏。
5. 根据需要，打开多个终端，例如查看帮助。

练习环境准备

将文件 **rhce9_prepare.sh、rhce9-exam294.zip** 上传到foundation0上root用户家目录，然后执行以下命令：

```
[root@foundation0 ~]# chmod +x /root/rhce9_prepare.sh
[root@foundation0 ~]# /root/rhce9_prepare.sh
Reset all VMs:
  Wait for classroom running ..................................... DONE
  Wait for bastion running ....................................... DONE
  Wait for workstation running ................................... DONE
  Wait for utility running ....................................... DONE
  Wait for servera running ....................................... DONE
  Wait for serverb running ....................................... DONE
```

```
  Wait for serverc running ..................................... DONE
  Wait for serverd running ..................................... DONE
Prepare foundation ............................................. DONE
Setup bastion .................................................. DONE
Setup servera .................................................. DONE
Setup serverb .................................................. DONE

--------------------------------------------------------------------------------
          按 Super 键，搜索 EXAM，打开试题，按回车键开始答题
--------------------------------------------------------------------------------
```

脚本执行完成后开始做题。

练习题

1. 安装和配置 Ansible

按照下方所述，在控制节点 **bastion.lab.example.com** 上安装和配置 Ansible：

- 安装所需的软件包
- 创建名为 `inventory` 的静态清单文件，以满足以下要求：
  - `workstation` 是 `balancers` 主机组的成员
  - `servera` 是 `dev` 主机组的成员
  - `serverb` 是 `test` 主机组的成员
  - `serverc` 和 `serverd` 是 `prod` 主机组的成员
  - `prod` 组是 `webservers` 主机组的成员
- 创建名为 `ansible.cfg` 的配置文件，以满足以下要求：
  - 主机清单文件为 `inventory`
  - `playbook` 中使用的**角色**位置为 `roles`
  - `playbook` 中使用的**内容集**位置为 `mycollections`

解答：

```
[kiosk@foundation0 ~]$ ssh devops@bastion

# 安装ansible软件包
[devops@bastion ~]$ sudo yum install -y ansible-core ansible-navigator

# 创建项目目录，并切换到项目目录
[devops@bastion ~]$ mkdir ansible && cd ansible/

# 创建inventory文件
[devops@bastion ansible]$ vim inventory
[balancers]
workstation
```

```
[dev]
servera

[test]
serverb

[prod]
serverc
serverd

[webservers:children]
prod

# 创建 ansible.cfg，内容参考 /etc/ansible/ansible.cfg 中命令
[devops@bastion ansible]$ head -n 3 /etc/ansible/ansible.cfg
# Since Ansible 2.12 (core):
# To generate an example config file (a "disabled" one with all default settings, commented out):
#               $ ansible-config init --disabled > ansible.cfg

# 临时生成一个ansible.cfg.ori，参考ansible.cfg.ori 创建ansible.cfg
[devops@bastion ansible]$ ansible-config init --disabled > ansible.cfg.ori

[devops@bastion ansible]$ vim ansible.cfg
[defaults]
inventory=inventory
remote_user=devops
roles_path=roles
collections_path=mycollections

[privilege_escalation]
become=True
become_method=sudo
become_user=root
become_ask_pass=False

# 创建角色目录和内容集目录
[devops@bastion ansible]$ mkdir roles mycollections

# 验证ansible配置
[devops@bastion ansible]$ ansible all -m command -a id

# 验证inventory结构
[devops@bastion ansible]$ ansible-inventory --graph
@all:
  |--@balancers:
  |  |--workstation
  |--@dev:
  |  |--servera
  |--@test:
```

```
|  |--serverb
|--@ungrouped:
|--@webservers:
|  |--@prod:
|  |  |--serverc
|  |  |--serverd

# 查看仓库地址
[devops@bastion ansible]$ grep image ~/.ansible-navigator.yml
    image: utility.lab.example.com/ee-supported-rhel8:latest

# 登录 registry，下载镜像
[devops@bastion ansible]$ podman login -u admin -p redhat
utility.lab.example.com
[devops@bastion ansible]$ podman pull utility.lab.example.com/ee-
supported-rhel8:latest
```

2. 配置您的系统以使用默认存储库

作为系统管理员，您需要在受管节点上安装软件。

请按照下方所述，创建一个名为 `yum_repo.yml` 的 playbook，在各个受管节点上安装 yum 存储库：

- 存储库1：
  - 存储库的名称为：EX294_BASE
  - 描述为：EX294 base software
  - 基础URL为：http://content.example.com/rhel9.0/x86_64/dvd/BaseOS
  - GPG签名检查为：启用状态
  - GPG密钥URL为：http://content.example.com/rhel9.0/x86_64/dvd/RPM-GPG-KEY-redhat-release
    存储库状态为：启用状态
- 存储库2：
  - 存储库的名称为：EX294_STREAM
  - 描述为：EX294 stream software
  - 基础URL为：http://content.example.com/rhel9.0/x86_64/dvd/AppStream
  - GPG签名检查为启用状态
  - GPG密钥URL为：http://content.example.com/rhel9.0/x86_64/dvd/RPM-GPG-KEY-redhat-release
  - 存储库状态为：启用状态

解答：

vim 工具配置

```
# yaml文件使用vim工具编辑，配置vim工具
# 设置ai，自动缩进auotoindex
# 设置ts=2，tab键用2个空格代替
[devops@bastion ansible]$ vim ~/.vimrc
set ai ts=2
```

准备 Playbook 模板

```
[devops@bastion ansible]$ vim tpl.yml
```

```
---
- name:
  hosts:
  tasks:
    - name:
```

查找模块帮助

> 以 yum 为例

```
# 查找模块名
[devops@bastion ansible]$ ansible-doc -l|grep yum
yum                     Manages packages with the `yum' package manager

yum_repository          Add or remove YUM repositories

# 查看模块使用说明，搜索EXAMPLE（/EXAMPLE），抄示例
[devops@bastion ansible]$ ansible-doc yum_repository
......
- name: Add multiple repositories into the same file (1/2)
  yum_repository:
    name: epel
    description: EPEL YUM repo
    file: external_repos
    baseurl:
https://download.fedoraproject.org/pub/epel/$releasever/$basearch/
    gpgcheck: no
......
```

创建 Playbook

```
[devops@bastion ansible]$ cp tpl.yml yum_repo.yml
[devops@bastion ansible]$ vim yum_repo.yml
```

```
---
- name: configuration yum repository
  hosts: all
```

```
  tasks:
    - name: BaseOS
      yum_repository:
        name: EX294_BASE
        description: EX294 base software
        baseurl: http://content.example.com/rhel9.0/x86_64/dvd/BaseOS
        gpgcheck: yes
        enabled: yes
        gpgkey: http://content.example.com/rhel9.0/x86_64/dvd/RPM-GPG-KEY-redhat-release

    - name: AppStream
      yum_repository:
        name: EX294_STREAM
        description: EX294 stream software
        baseurl: http://content.example.com/rhel9.0/x86_64/dvd/AppStream
        gpgcheck: yes
        enabled: yes
        gpgkey: http://content.example.com/rhel9.0/x86_64/dvd/RPM-GPG-KEY-redhat-release
```

```
# 执行脚本验证效果
[devops@bastion ansible]$ ansible-navigator run -m stdout yum_repo.yml
[kiosk@foundation0 ~]$ ssh root@servera ls /etc/yum.repos.d/
EX294_BASE.repo
EX294_STREAM.repo
```

3. 使用 Ansible Galaxy 安装内容集

站点 http://materials.example.com/laoma 中存放多个内容集，将下面3个内容集安装到 `mycollections` 目录：

- ansible-posix-1.4.0.tar.gz
- community-general-4.3.0.tar.gz
- redhat-rhel_system_roles-1.16.2.tar.gz

解答：

```
# 安装内容集
[devops@bastion ansible]$ ansible-galaxy collection install http://materials.example.com/laoma/ansible-posix-1.4.0.tar.gz
[devops@bastion ansible]$ ansible-galaxy collection install http://materials.example.com/laoma/community-general-4.3.0.tar.gz
[devops@bastion ansible]$ ansible-galaxy collection install http://materials.example.com/laoma/redhat-rhel_system_roles-1.16.2.tar.gz
```

```
# 查看内容集清单
[devops@bastion ansible]$ ansible-galaxy collection list

# /home/devops/ansible/mycollections/ansible_collections
Collection                Version
------------------------- -------
ansible.posix             1.4.0
community.general         4.3.0
redhat.rhel_system_roles 1.16.2
```

4. 安装软件包

创建一个名为 `packages.yml` 的 playbook:

- 将 `php` 和 `mariadb` 软件包安装到 `dev`、`test` 和 `prod` 主机组中的主机上
- 将 `RPM Development Tools` 软件包组安装到 `dev` 主机组中的主机上
- 将 `dev` 主机组中主机上的所有软件包更新为最新版本

解答:

```
# 创建 Playbook
[devops@bastion ansible]$ cp tpl.yml packages.yml
[devops@bastion ansible]$ vim packages.yml
```

```
---
- name: Installation software packages
  hosts: dev,test,prod
  tasks:
    - name: install php and mariadb
      yum:
        name: php,mariadb
        state: present
    - name: install RPM Development Tools
      yum:
        name: "@RPM Development Tools"
        state: present
      when: inventory_hostname in groups.dev
    - name: Upgrage all packages
      yum:
        name: "*"
        state: latest
      when: inventory_hostname in groups.dev
```

```
# 执行Playbook
[devops@bastion ansible]$ ansible-navigator run -m stdout packages.yml

# 验证
[devops@bastion ansible]$ ansible dev -a 'yum list php mariadb'
```

5-1. 使用 RHEL 系统角色-timesync

安装 RHEL 系统角色软件包，并创建名称为 `timesync.yml` 的 playbook，符合以下条件：

- 在所有受管节点上运行
- 使用 `timesync` 角色
- 配置该角色，以使用当前有效的NTP提供商
- 配置该角色，以使用时间服务器172.25.254.254
- 配置该角色，以启用iburst参数

> **提示：考试时，5-1 和5-2 只考其中1题**

解答：

```
# 安装rhel-system-roles
[devops@bastion ansible]$ sudo yum install -y rhel-system-roles

# 复制rhel-system-roles.timesync角色到项目角色目录，并取名为timesync
[devops@bastion ansible]$ cp -r /usr/share/ansible/roles/rhel-system-roles.timesync roles/timesync

# 查看角色清单
[devops@bastion ansible]$ ansible-galaxy list
# /home/devops/ansible/roles
- timesync, (unknown version)

# 创建 Playbook
[devops@bastion ansible]$ cp tpl.yml timesync.yml

# 编辑Playbook，参照roles/timesync/README.md文件查看变量
[devops@bastion ansible]$ vim timesync.yml
```

```
---
- name: Use RHEL system role
  hosts: all
  vars:
    timesync_ntp_servers:
      - hostname: 172.25.254.254
        iburst: yes
  roles:
    - timesync
```

```
# 执行Playbook
[devops@bastion ansible]$ ansible-navigator run -m stdout timesync.yml

# 验证，以dev主机为例
[devops@bastion ansible]$ ansible dev -a 'cat /etc/chrony.conf'
servera | CHANGED | rc=0 >>
#
# Ansible managed
#

server 172.25.254.254 iburst

......
```

5-2. 使用 RHEL 系统角色-selinux

安装RHEL系统角色软件包，并创建名称为 `selinux.yml` 的 playbook，符合以下条件：

- 在所有受管节点上运行
- 使用 `selinux` 角色
- `selinux` 策略为 `targeted`
- `selinux` 模式为 `enforcing`

> **提示：考试时，5-1 和5-2 只考其中1题**

解答：

参考 `/usr/share/doc/rhel-system-roles/selinux/example-selinux-playbook.yml`

```
# 安装rhel-system-roles
[devops@bastion ansible]$ sudo yum install -y rhel-system-roles

# 复制rhel-system-roles.selinux角色到项目角色目录，并取名为selinux
[devops@bastion ansible]$ cp -r /usr/share/ansible/roles/rhel-system-
roles.selinux roles/selinux

# 查看角色清单
[devops@bastion ansible]$ ansible-galaxy list
# /home/devops/ansible/roles
- selinux, (unknown version)

# 创建 Playbook
[devops@bastion ansible]$ cp /usr/share/doc/rhel-system-
roles/selinux/example-selinux-playbook.yml selinux.yml

# 编辑Playbook，参照roles/timesync/README.md文件查看变量
[devops@bastion ansible]$ vim selinux.yml
```

```yaml
---
- name: Use RHEL system role
  hosts: all
  vars:
    selinux_policy: targeted
    selinux_state: enforcing
    selinux_reboot_required: true
  tasks:
    - block:
        - name: Apply SELinux role
          include_role:
            name: selinux
      rescue:
        - name: Check for failure for other reasons than required reboot
          fail:
          when: not selinux_reboot_required
        - reboot:
        - name: Reapply SELinux role
          include_role:
            name: selinux
```

```
# 执行Playbook
[devops@bastion ansible]$ ansible-navigator run -m stdout selinux.yml

# 验证，以 test 主机组为例
[devops@bastion ansible]$ ansible test -a 'getenforce'
serverb | CHANGED | rc=0 >>
Enforcing
```

6. 使用 Ansible Galaxy 安装角色

使用 Ansible Galaxy 和要求文件 `roles/requirements.yml`。

从以下 URL 下载角色并安装到 `roles` 目录：

- 来源：http://materials.example.com/laoma/haproxy.tar，角色名称为： `balancer`
- 来源：http://materials.example.com/laoma/phpinfo.tar，角色名称为： `phpinfo`

解答：

```
# 创建角色安装文件
[devops@bastion ansible]$ vim roles/requirements.yml
```

```
---
- src: http://materials.example.com/laoma/haproxy.tar
  name: balancer
- src: http://materials.example.com/laoma/phpinfo.tar
  name: phpinfo
```

```
# 安装角色
[devops@bastion ansible]$ ansible-galaxy install -r
roles/requirements.yml
- downloading role from http://materials.example.com/laoma/haproxy.tar
- extracting balancer to /home/devops/ansible/roles/balancer
- balancer was installed successfully
- downloading role from http://materials.example.com/laoma/phpinfo.tar
- extracting phpinfo to /home/devops/ansible/roles/phpinfo
- phpinfo was installed successfully

# 查看角色清单
[devops@bastion ansible]$ ansible-galaxy role list
# /home/devops/ansible/roles
- timesync, (unknown version)
- balancer, (unknown version)
- phpinfo, (unknown version)
```

7. 创建和使用角色

根据下列要求，在 `roles` 中创建名为 `apache` 的角色：

- `httpd` 软件包已安装，设为在系统启动时启用并启动
- 防火墙已启用并正在运行，并使用允许访问 `web` 服务器的规则
- 模板文件 `index.html.j2` 用于创建文件 `/var/www/html/index.html` 具有以下输出内容：**Welcome to HOSTNAME on IPADDRESS**。`HOSTNAME` 是受管节点的完全限定域名，`IPADDRESS` 则是受管节点的IP地址。

按照下方所述，创建一个名为 `newrole.yml` 的 playbook，使用此角色在 `webservers` 主机组上运行。

解答：

```
# 创建角色
[devops@bastion ansible]$ ansible-galaxy init apache --init-path=roles
- apache was created successfully

# 编辑角色任务
[devops@bastion ansible]$ vim roles/apache/tasks/main.yml
```

```
---
# tasks file for apache
```

```yaml
- name: install package httpd
  yum:
    name: httpd
    state: present

# 防火墙服务必须启动
- name: set service
  service:
    name: "{{ item }}"
    state: started
    enabled: yes
  loop:
    - httpd
    - firewalld

- name: set firewall allow http
  ansible.posix.firewalld:
    service: http
    permanent: yes
    immediate: yes
    state: enabled

- name: prepare index.html
  template:
    src: index.html.j2
    dest: /var/www/html/index.html
    owner: apache
    group: apache
```

```
# web主页模板文件
# facts 变量通过 ansible servera -m setup |less命令查找
[devops@bastion ansible]$ vim roles/apache/templates/index.html.j2
Welcome to {{ ansible_fqdn }} on {{ ansible_default_ipv4.address }}

# 创建 Playbook
[devops@bastion ansible]$ cp timesync.yml newrole.yml
[devops@bastion ansible]$ vim newrole.yml
```

```yaml
---
- name: use apache roles
  hosts: webservers
  roles:
    - apache
```

```
# 执行Playbook
[devops@bastion ansible]$ ansible-navigator run -m stdout newrole.yml

# 验证
[devops@bastion ansible]$ curl http://serverc.lab.example.com
Welcome to serverc.lab.example.com on 172.25.250.12
[devops@bastion ansible]$ curl http://serverd.lab.example.com
Welcome to serverd.lab.example.com on 172.25.250.13
```

8. 从 Ansible Galaxy 使用角色

根据下列要求，创建一个名为 `roles.yml` 的 playbook：

- playbook 中包含一个 play，该 play 在 `balancers` 主机组中的主机上运行并将使用 `balancer` 角色。此角色配置一项服务，以在 `webservers` 主机组中的主机之间平衡 `Web` 服务器请求的负载。
  - 浏览到 `balancers` 主机组中的主机（例如http://workstation.lab.example.com）将生成以下输出：`Welcom to serverc.lab.example.com on 172.25.250.12`
  - 重新加载浏览器将从另一 Web服务器生成输出：`Welcom to serverd.lab.example.com on 172.25.250.13`
- playbook 中包含另外一个 play，该 play 在 `webservers` 主机组中的主机上运行并将使用 `phpinfo` 角色。通过 `URL/hello.php` 浏览到 `webservers` 主机组中的主机，将生成以下输出：`Hello PHP World from FQDN`。其中，`FQDN`是主机的完全限定名称，

  例如，浏览到 http://serverc.lab.example.com/hello.php，会生成以下输出：

  ```
  Hello PHP World from serverc.lab.example.com
  ```

  同样，浏览到 http://serverd.lab.example.com/hello.php，会生成以下输出：

  ```
  Hello PHP World from serverd.lab.example.com
  ```

  另外还有 PHP 配置的各种详细信息，如安装的 PHP 版本等。

解答：

```
# 创建 Playbook
[devops@bastion ansible]$ cp timesync.yml roles.yml
[devops@bastion ansible]$ vim roles.yml
```

```
---
# 先运行使用 phpinfo 角色的剧本，然后在运行使用 balancers 角色的剧本，
# 这样操作balancer才会把后端服务器设置为 webservers，否则 balancers 角色将执行失败
- name: use phpinfo roles
```

```
  hosts: webservers
  roles:
    - phpinfo

- name: use balancer roles
  hosts: balancers
  roles:
    - balancer

  tasks:
    - name: Start firewalld, if not started
      service:
        name: firewalld
        state: started
        enabled: yes

    - name: permit apache
      ansible.posix.firewalld:
        service: http
        permanent: yes
        state: enabled
        immediate: yes
```

```
# 执行Playbook
[devops@bastion ansible]$ ansible-navigator run -m stdout roles.yml

# 验证
[devops@bastion ansible]$ curl http://workstation.lab.example.com
[devops@bastion ansible]$ curl
http://workstation.lab.example.com/hello.php
```

9-1. 创建和使用逻辑卷

创建一个名为 `lv.yml` 的 playbook，它将在所有受管节点上运行以执行下列任务：

- 在 `research` 卷组中创建逻辑卷
- 逻辑卷名称为 `data`
- 逻辑卷大小为 `6000 MiB`
- 使用 `ext4` 文件系统格式化逻辑卷
- 如果无法创建请求的逻辑卷大小，应显示错误信息：

  ```
  Could not create logical volume of that size
  ```

  并将使用大小改为 `800 MiB`。
- 如果卷组 `research` 不存在，应显示错误信息：

```
Volume group does not exist
```

- 不要以任何方式挂载逻辑卷

> **提示：考试时，9-1和9-2只考其中1题**

解答：

```
# 创建 Playbook
[devops@bastion ansible]$ cp tpl.yml lv.yml
[devops@bastion ansible]$ vim lv.yml
```

```
---
- name: Create and use lv
  hosts: all
  tasks:
    - block:
        - name: Create a logical volume of 6000m
          community.general.lvol:
            vg: research
            lv: data
            size: 6000
      rescue:
        - debug:
            msg: Could not create logical volume of that size
        - name: Create a logical volume of 800m
          community.general.lvol:
            vg: research
            lv: data
            size: 800
      always:
        - name: Create a ext4
          community.general.filesystem:
            fstype: ext4
            dev: /dev/research/data
      # 根据变量是否定义判断卷组是否存在
      when: ansible_lvm.vgs.research is defined
    - debug:
        msg: Volume group does not exist
      when: ansible_lvm.vgs.research is not defined
```

```
# 执行Playbook
[devops@bastion ansible]$ ansible-navigator run -m stdout lv.yml

# 验证
[devops@bastion ansible]$ ansible all -a 'lvs'
```

9-2. 创建和使用分区

创建一个名为 `partition.yml` 的 playbook，它将在所有受管节点上运行以执行下列任务：

- 在设备 `vdb` 中创建主分区，`id` 为 `1`，大小是 `1500 MiB`
- 使用 `ext4` 文件系统格式化
- 仅在 `dev` 主机组中节点，挂载在 `/data`
- 如果无法创建请求的分区大小，应显示错误信息：

```
Could not create partition of that size
```

  并将使用大小改为 `800 MiB`。

- 如果设备 `vdd` 不存在，应显示错误信息：

```
Disk does not exist
```

> **提示：考试时，9-1和9-2只考其中1题**

解答：

```
# 创建 Playbook
[devops@bastion ansible]$ cp tpl.yml partition.yml
[devops@bastion ansible]$ vim partition.yml
```

```yaml
---
- name: config partition
  hosts: all
  tasks:
    - block:
        - name: create part
          community.general.parted:
            device: /dev/vdb
            number: 1
            state: present
            part_start: 1MiB
            part_end: 1501MiB
      rescue:
        - name: debug part
          debug:
            msg: Can not create partition of that size
        - name: recreate part
          community.general.parted:
            device: /dev/vdb
            number: 1
            state: present
            part_start: 1MiB
            part_end: 801MiB
      always:
```

```
        - name: create ext4
          filesystem:
            fstype: ext4
            dev: /dev/vdb1
        - name: /data
          file:
            path: /data
            state: directory
          when: inventory_hostname in groups.dev
        - name: mount
          mount:
            path: /data
            fstype: ext4
            src: /dev/vdb1
            state: mounted
          when: inventory_hostname in groups.dev
      when: ansible_devices.vdb is defined
    - name: debug vdd
      debug:
        msg:  Disk does not exist
      when: ansible_devices.vdd is not defined
```

```
# 执行Playbook
[devops@bastion ansible]$ ansible-navigator run -m stdout partition.yml

# 验证
[devops@bastion ansible]$ ansible all -a 'lsblk /dev/vdb'
```

分区练习

如果大家想在练习环境完成分区练习，将题目改写如下：

创建一个名为 `partition.yml` 的 playbook，它将在 **`prod`** 主机组上运行以执行下列任务：

- 在设备 `vdb` 中创建主分区，`id` 为 `1`，大小是 `1500 MiB`
- 使用 `ext4` 文件系统格式化
- 仅在 **`serverc`** 主机节点，挂载在 `/data`
- 如果无法创建请求的分区大小，应显示错误信息：

  ```
  Could not create partition of that size
  ```

  并将使用大小改为 `800 MiB`。
- 如果设备 `vdd` 不存在，应显示错误信息：

  ```
  Disk does not exist
  ```

解答：

```
# 创建 Playbook
[devops@bastion ansible]$ cp tpl.yml partition.yml
[devops@bastion ansible]$ vim partition.yml
```

```yaml
---
- name: config partition
  hosts: prod
  tasks:
    - block:
        - name: create part
          community.general.parted:
            device: /dev/vdb
            number: 1
            state: present
            part_start: 1MiB
            part_end: 1501MiB
      rescue:
        - name: debug part
          debug:
            msg: Can not create partition of that size
        - name: recreate part
          community.general.parted:
            device: /dev/vdb
            number: 1
            state: present
            part_start: 1MiB
            part_end: 801MiB
      always:
        - name: create ext4
          filesystem:
            fstype: ext4
            dev: /dev/vdb1
        - name: /data
          file:
            path: /data
            state: directory
          when: "inventory_hostname == 'serverc'"
        - name: mount
          mount:
            path: /data
            fstype: ext4
            src: /dev/vdb1
            state: mounted
          when: "inventory_hostname == 'serverc'"
      when: ansible_devices.vdb is defined
    - name: debug vdd
      debug:
        msg:  Disk does not exist
      when: ansible_devices.vdd is not defined
```

```
# 执行Playbook
[devops@bastion ansible]$ ansible-navigator run -m stdout partition.yml

# 验证
[devops@bastion ansible]$ ansible all -a 'lsblk /dev/vdb'
```

10. 生成主机文件

下载初始模板文件 http://materials.example.com/laoma/hosts.j2 到 `/home/devops/ansible`。完成该模板，以便用它生成以下文件，针对每个清单主机包含一行内容，其格式与 `/etc/hosts` 相同。

创建名为 `hosts.yml` 的 playbook，它将使用此模板在 `dev` 主机组中的主机上生成文件 `/etc/myhosts`。

该 playbook 运行后，`dev` 主机组中主机上的文件 `/etc/myhosts` 应针对每个受管主机包含一行内容：

```
127.0.0.1 localhost localhost.localdomain localhost4
localhost4.localdomain4
::1 localhost localhost.localdomain localhost6 localhost6.localdomain6

172.25.250.9 workstation.lab.example.com workstation
172.25.250.10 servera.lab.example.com servera
172.25.250.11 serverb.lab.example.com serverb
172.25.250.12 serverc.lab.example.com serverc
172.25.250.13 serverd.lab.example.com serverd
```

> **注：清单主机名称的显示顺序不重要。**

解答：

```
# 下载模版文件
[devops@bastion ansible]$ wget
http://materials.example.com/laoma/hosts.j2

## 编辑模版文件
# 变量hostvars[host]中host不要加单引号
[devops@bastion ansible]$ vim hosts.j2
127.0.0.1 localhost localhost.localdomain localhost4
localhost4.localdomain4
::1 localhost localhost.localdomain localhost6 localhost6.localdomain6

{% for host in groups.all %}
{{ hostvars[host].ansible_default_ipv4.address }} {{
hostvars[host].ansible_fqdn }} {{ hostvars[host].ansible_hostname }}
{% endfor %}
```

```
######################以上是文件内容#############################

# 创建 Playbook
[devops@bastion ansible]$ cp tpl.yml hosts.yml
[devops@bastion ansible]$ vim hosts.yml
```

```
---
- name: Generate a host file
  hosts: all
  tasks:
    - name: template /etc/myhosts
      template:
        src: hosts.j2
        dest: /etc/myhosts
      when: inventory_hostname in groups.dev
```

```
# 执行Playbook
[devops@bastion ansible]$ ansible-navigator run -m stdout hosts.yml

# 验证
[devops@bastion ansible]$ ansible dev -a 'cat /etc/myhosts'
```

11. 修改文件内容

按照下方所述，创建一个名为 `issue.yml` 的 playbook：

- 该 playbook 将在所有清单主机上运行。
- 该 playbook 会将 `/etc/issue` 的内容替换为下方所示的一行文本：
  - 在 `dev` 主机组中的主机上，这行文本显示为：`Development`
  - 在 `test` 主机组中的主机上，这行文本显示为：`Test`
  - 在 `prod` 主机组中的主机上，这行文本显示为：`Production`

解答：

```
# 创建 Playbook
[devops@bastion ansible]$ cp tpl.yml issue.yml
[devops@bastion ansible]$ vim issue.yml
```

```
---
- name: Modify /etc/issue content
  hosts: all
  tasks:
    - name: host dev
      copy:
        content: "Development"
        dest: /etc/issue
```

```
    when: inventory_hostname in groups.dev
  - name: host test
    copy:
      content: "Test"
      dest: /etc/issue
    when: inventory_hostname in groups.test
  - name: host prod
    copy:
      content: "Production"
      dest: /etc/issue
    when: inventory_hostname in groups.prod
```

```
# 执行Playbook
[devops@bastion ansible]$ ansible-navigator run -m stdout issue.yml

# 验证
[devops@bastion ansible]$ ansible dev,test,prod -a 'cat /etc/issue'
```

12. 创建 Web 内容目录

按照下方所述，创建一个名为 `webcontent.yml` 的 playbook：

- 该 playbook 在 `dev` 主机组中的受管节点上运行。
- 创建符合下列要求的目录 `/webdev`：
  - 所有者为 `webdev` 组
  - 具有常规权限：

    ```
    owner=read+write+execute, group=read+write+execute, other=read+execute
    ```

  - 具有特殊权限：设置组ID
  - 用符号链接将 `/var/www/html/webdev` 链接到 `/webdev`
  - 创建文件 `/webdev/index.html`，其中包含如下所示的单行文件：`Development`
  - 在 `dev` 主机组中主机上浏览此目录（例如http://servera.lab.example.com/webdev/）将生成以下输出：`Development`

解答：

```
# 创建 Playbook
[devops@bastion ansible]$ cp tpl.yml webcontent.yml
[devops@bastion ansible]$ vim webcontent.yml
```

```
---
- name: Create a Web Content Directory
  hosts: dev
```

```yaml
  # 首先使用apache角色部署dev主机组为web服务器
  roles:
    - apache

  tasks:
    - name: ensure webdev exit
      group:
        name: webdev
        state: present

    - name: create /webdev
      file:
        path: /webdev
        state: directory
        # mode参数值必须使用单引号
        mode: '2775'
        group: webdev
        setype: httpd_sys_content_t

    - name: create a link
      file:
        src: /webdev
        dest: /var/www/html/webdev
        state: link

    - name: index.html
      copy:
        content: "Development\n"
        dest: /webdev/index.html
        setype: httpd_sys_content_t
```

```
# 执行Playbook
[devops@bastion ansible]$ ansible-navigator run -m stdout webcontent.yml

# 验证
[devops@bastion ansible]$ curl http://servera.lab.example.com/webdev/
Development
```

13. 生成硬件报告

创建一个名为 `hwreport.yml` 的 playbook：

- 在所有受管节点上生成含有以下信息的输出文件 `/root/hwreport.txt`：
  - 清单主机名称
  - 以 `MB` 表示的总内存大小
  - `BIOS` 版本

    - 磁盘设备 `vda` 的大小
    - 磁盘设备 `vdb` 的大小
- 输出文件中的每一行都是 `key=value` 对。
- 您的 playbook应当从http://materials.example.com/laoma/hwreport.empty下载文件，并将它保存为 `/root/hwreport.txt`
- 使用正确的值更改 `/root/hwreport.txt`。
- 如果硬件项不存在，相关的值应设为 `NONE`

解答：

```
# 下载硬件报告模板文件，查看文件内容
[devops@bastion ansible]$ wget
http://materials.example.com/laoma/hwreport.empty
[devops@bastion ansible]$ cat hwreport.empty
# Hardware report
HOST=inventoryhostname
MEMORY=memory_in_MB
BIOS=BIOS_version
DISK_SIZE_VDA=disk_vda_size
DISK_SIZE_VDB=disk_vdb_size

# 创建 Playbook
[devops@bastion ansible]$ cp tpl.yml hwreport.yml
[devops@bastion ansible]$ vim hwreport.yml
```

```
---
- name: Create hardware report
  hosts: all
  tasks:
    - name: download hwreport.txt
      get_url:
        url: http://materials.example.com/laoma/hwreport.empty
        dest: /root/hwreport.txt
        force: yes
    - name: hostname
      lineinfile:
        path: /root/hwreport.txt
        regexp: '^HOST='
        line: 'HOST={{ inventory_hostname }}'
    - name: memory
      lineinfile:
        path: /root/hwreport.txt
        regexp: '^MEMORY='
        line: 'MEMORY={{ ansible_memtotal_mb }}'
    - name: BIOS version
      lineinfile:
        path: /root/hwreport.txt
        regexp: '^BIOS='
        line: 'BIOS={{ ansible_bios_version }}'
```

```
    - name: vda size
      lineinfile:
        path: /root/hwreport.txt
        regexp: '^DISK_SIZE_VDA='
        line: 'DISK_SIZE_VDA={{ ansible_devices.vda.size | default("NONE")}}'
    - name: vdb size
      lineinfile:
        path: /root/hwreport.txt
        regexp: '^DISK_SIZE_VDB='
        line: 'DISK_SIZE_VDB={{ ansible_devices.vdb.size | default("NONE")}}'
```

```
# 执行Playbook
[devops@bastion ansible]$ ansible-navigator run -m stdout hwreport.yml

# 验证
[devops@bastion ansible]$ ansible all -a 'cat /root/hwreport.txt'
```

14. 创建密码库

按照下方所述，创建一个 Ansible Vault 文件来存储用户密码：

- Vault 文件名称为 `locker.yml`
- Vault 文件中含有两个变量：
  - `pw_developer` 值为 `Imadev`
  - `pw_manager` 值为 `Imamgr`
- 用于加密和解密该库的密码为：`laoma`
- 密码存储在文件 `secret.txt` 中

解答：

```
# 创建Ansible库
[devops@bastion ansible]$ vim locker.yml
---
pw_developer: Imadev
pw_manager: Imamgr

# 将密码存储在文件中
[devops@bastion ansible]$ echo -n laoma > secret.txt
[devops@bastion ansible]$ chmod 600 secret.txt

# 加密Ansible库
[devops@bastion ansible]$ ansible-vault encrypt locker.yml --vault-password-file=secret.txt

# 验证
```

```
[devops@bastion ansible]$ ansible-vault view locker.yml --vault-password-file=secret.txt
```

> **不要在ansible.cfg中配置vault-password-file参数。**

15. 创建用户帐户

从http://materials.example.com/laoma/user_list.yml下载要创建的用户的列表，并将它保存到 `/home/devops/ansible` 目录。

在本次考试中使用在其他位置创建的密码库 `locker.yml`。创建名为 `users.yml` 的 playbook，从而按以下所述创建用户帐户：

- 职位描述为 `developer` 的用户应当：
  - 在 `dev` 和 `test` 主机组中的受管节点上创建
  - 从 `pw_developer` 变量分配密码
  - 是补充组 `devops` 的成员
  - 设置用户 `UID`
  - 设置密码最大有效期
- 职位描述为 `manager` 的用户应当：
  - 在 `prod` 主机组中的受管节点上创建
  - 从 `pw_manager` 变量分配密码
  - 是补充组 `opsmgr` 的成员
  - 设置用户 `UID`
  - 设置密码最大有效期
- 密码采用 `SHA512` 哈希格式。

您的 playbook 应能够在本次考试中使用在其他位置创建的库密码文件 `secret.txt` 正常运行。

解答：

```
# 下载变量文件并查看文件内容
[devops@bastion ansible]$ wget http://materials.example.com/laoma/user_list.yml
[devops@bastion ansible]$ cat user_list.yml
---
users:
  - name: james
    job: manager
    uid: 2001
    PASS_MAX_DAYS: 100
  - name: mary
    job: manager
    uid: 2002
    PASS_MAX_DAYS: 100
```

```
  - name: john
    job: developer
    uid: 3001
    PASS_MAX_DAYS: 90

# 创建 Playbook
[devops@bastion ansible]$ cp tpl.yml users.yml
[devops@bastion ansible]$ vim users.yml
```

```yaml
---
- name: q14-1
  hosts: dev,test
  vars_files:
    - user_list.yml
    - locker.yml
  tasks:
    - name: create group
      group:
        name: devops
        state: present

    - name: create user
      user:
        name: "{{ item.name }}"
        uid: "{{ item.uid }}"
        password: "{{ pw_developer | password_hash('sha512') }}"
        password_expire_max: "{{ item.PASS_MAX_DAYS }}"
        groups: devops
        state: present
      loop: "{{ users }}"
      when: 'item.job == "developer"'

- name: q14-2
  hosts: prod
  vars_files:
    - user_list.yml
    - locker.yml
  tasks:
    - name: create group
      group:
        name: opsmgr
        state: present

    - name: create user
      user:
        name: "{{ item.name }}"
        uid: "{{ item.uid }}"
        password: "{{ pw_manager | password_hash('sha512') }}"
        password_expire_max: "{{ item.PASS_MAX_DAYS }}"
        groups: opsmgr
        state: present
```

```
    loop: "{{ users }}"
    when: 'item.job == "manager"'
```

```
# 执行Playbook
[devops@bastion ansible]$ ansible-navigator run -m stdout users.yml --vault-password-file=secret.txt

# 验证用户账户
[devops@bastion ansible]$ ansible servera -a 'id john'
servera | CHANGED | rc=0 >>
uid=1002(john) gid=1003(john) groups=1003(john),1001(devops)

# 验证用户账户密码有效期，是否是30天后
[devops@bastion ansible]$ ansible servera -a 'chage -l john'|grep Password

# 验证用户密码
[devops@bastion ansible]$ ssh john@servera
[john@servera ~]$ su - john
```

16. 更新 Ansible 库的密钥

按照下方所述，更新现有Ansible库的密钥：

- 从http://materials.example.com/laoma/salaries.yml下载Ansible库到/home/devops/ansible目录
- 当前的库密码为 `laoma`
- 新的库密码为 `redhat`
- 库使用新密码保持加密状态

解答：

```
# 下载Ansible库
[devops@bastion ansible]$ wget http://materials.example.com/laoma/salaries.yml

# 设置新密码
[devops@bastion ansible]$ ansible-vault rekey salaries.yml
Vault password: `laoma`
New Vault password: `redhat`
Confirm New Vault password: `redhat`
Rekey successful

# 查看文件内容，验证新密码
[devops@bastion ansible]$ ansible-vault view salaries.yml
Vault password: redhat
hello laoma
```

17. 配置 cron 作业

创建一个名为 `cron.yml` 的 playbook，它将在所有受管节点上运行，配置一个计划任务：

- 运行身份 `devops`
- 运行时间间隔 2 分钟
- 运行命令：`logger "EX294 in progress"`

解答：

```
# 创建 Playbook
[devops@bastion ansible]$ cp tpl.yml cron.yml
[devops@bastion ansible]$ vim cron.yml
```

```
---
- name: Create a cron job
  hosts: all
  tasks:
    - name: ensure user devops is exist
      user:
        name: devops
        state: present
    - name: Ensure a job that runs every 2 minutes
      cron:
        name: devops job
        minute: "*/2"
        user: devops
        job: 'logger "EX294 in progress"'
```

```
# 执行Playbook
[devops@bastion ansible]$ ansible-navigator run -m stdout cron.yml

# 验证
[devops@bastion ansible]$ ansible all -a 'crontab -lu devops'
```

检查成绩

提示：

- 考试时，不提供评分脚本检查错误。
- 评分脚本，针对 5-1 和 9-1
- **使用 root 执行 grade 命令。**

```
[root@foundation0 ~]# grade
```