海量视频题库 myitpub.com QQ:5565462
www.52myit.com

章 10

自动执行 LINUX 管理任务

目标

利用 Ansible，实现常见 Linux 系统管理任务自动化。

培训目标

- 订阅系统，配置软件通道和存储库，以及管理受管主机上的 RPM 软件包。
- 管理 Linux 用户和组，配置 SSH，以及修改受管主机上的 Sudo 配置。
- 管理服务启动，使用 at、cron 和 systemd 调度进程，重新引导，以及控制受管主机上的默认启动目标。
- 对存储设备进行分区，配置 LVM，格式化分区或逻辑卷，挂载文件系统，以及添加交换文件或空间。

章节

- 管理软件和订阅（引导式练习）
- 管理用户和身份验证（引导式练习）
- 管理启动过程和调度的进程（引导式练习）
- 管理存储（引导式练习）
- 管理网络配置（引导式练习）

实验

- 自动执行 Linux 管理任务

海量视频题库 myitpub.com QQ:5565462
www.52myit.com

管理软件和订阅

培训目标

学完本节后，您应能够订阅系统，配置软件频道和存储库，以及管理受管主机上的 RPM 软件包。

使用 ANSIBLE 管理软件包

yum Ansible 模块在受管主机上使用 Yum 软件包管理器来处理软件包操作。下例中的 playbook 可将 httpd 软件包安装到 `servera.lab.example.com` 受管主机上。

```
---
- name: Install the required packages on the web server
  hosts: servera.lab.example.com
  tasks:
    - name: Install the httpd packages
      yum:
        name: httpd      ❶
        state: present   ❷
```

❶ `name` 关键字提供要安装的软件包的名称。

❷ `state` 关键字指明该软件包在受管主机上的预期状态：

`present`
如果尚不存在，Ansible 将安装该软件包。

`absent`
如果已安装，Ansible 将删除该软件包。

`latest`
如果软件包还不是最新的可用版本，Ansible 将更新该软件包。如果软件包尚未安装，则 Ansible 会安装它。

下表对 yum Ansible 模块和同等作用的 **yum** 命令的一些用法进行了对比。

ANSIBLE 任务	YUM 命令

`- name: Install httpd`
`  yum:`
`  name: httpd`
`  state: present`
	`yum install httpd`

`- name: Install or update httpd`
`  yum:`
`  name: httpd`
`  state: latest`
	`yum update httpd`，或者 **`yum install httpd`**（若软件包尚未安装）。

海量视频题库 myitpub.com QQ:5565462
www.52myit.com

ANSIBLE 任务	YUM 命令

`- name: Update all packages`
`  yum:`
`  name: '*'`
`  state: latest`
	yum update

`- name: Remove httpd`
`  yum:`
`  name: httpd`
`  state: absent`
	yum remove httpd

`- name: Install Development Tools`
`  yum:`
`  name: '@Development Tools'` ❶
`  state: present`
 	
❶ 使用 yum Ansible 模块时，您必须在组名称前面添加前缀 @。请记住，您可以使用 **yum group list** 命令来检索组列表。
	yum group install "Development Tools"

`- name: Remove Development Tools`
`  yum:`
`  name: '@Development Tools'`
`  state: absent`
	yum group remove "Development Tools"

`- name: Inst perl AppStream module`
`  yum:`
`  name: '@perl:5.26/minimal'` ❶
`  state: present`
 	
❶ 若要管理 Yum AppStream 模块，请在其名称前加上前缀 @。语法与使用 **yum** 命令相同。例如，您可以省略配置集部分以使用默认配置集：**@perl:5. 26**。请记住，您可以使用 **yum module list** 命令列出可用的 Yum AppStream 模块。
	yum module install perl:5.26/minimal

运行 **ansible-doc yum** 命令查看其他参数和 playbook 示例。

优化多软件包安装

如要对几个软件包进行操作，`name` 关键字可接受列表。下例演示了一个 playbook，它可将三个软件包安装到 `servera.lab.example.com` 上。

```
---
- name: Install the required packages on the web server
  hosts: servera.lab.example.com
  tasks:
```

海量视频题库 myitpub.com QQ:556546 2
www.52myit.com

```
    - name: Install the packages
      yum:
        name:
          - httpd
          - mod_ssl
          - httpd-tools
        state: present
```

通过利用这种语法，Ansible 可在单个 Yum 事务中安装这些软件包。这等同于运行 **`yum install httpd mod_ssl httpd-tools`** 命令。

此任务的常见版本是使用循环，但效率较低且速度较慢。

```
---
- name: Install the required packages on the web server
  hosts: servera.lab.example.com
  tasks:
    - name: Install the packages
      yum:
        name: "{{ item }}""
        state: present
      loop:
        - httpd
        - mod_ssl
        - httpd-tools
```

请避免使用此方法，因为它需要模块执行三个单独的事务，每个事务对应一个软件包。

收集有关已安装软件包的事实

`package_facts` Ansible 模块收集受管主机上已安装软件包的详细信息。它使用软件包详细信息来设置 `ansible_facts.packages` 变量。

以下 playbook 调用 `package_facts` 模块，再调用 `debug` 模块来显示 `ansible_facts.packages` 变量的内容，然后再次调用 `debug` 模块来查看已安装的 NetworkManager 软件包的版本。

```
---
- name: Display installed packages
  hosts: servera.lab.example.com
  tasks:
    - name: Gather info on installed packages
      package_facts:
        manager: auto

    - name: List installed packages
      debug:
        var: ansible_facts.packages

    - name: Display NetworkManager version
      debug:
        msg: "Version {{ansible_facts.packages['NetworkManager'][0].version}}"
      when: "'NetworkManager' in ansible_facts.packages"
```

海量视频题库 nvitpub.com QQ:5565462
www.52nvit.com

运行时，该 playbook 将显示软件包列表和 NetworkManager 软件包的版本：

```
[user@controlnode ~]$ ansible-playbook lspackages.yml

PLAY [Display installed packages] ****************************************

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

TASK [Gather info on installed packages] *********************************
ok: [servera.lab.example.com]

TASK [List installed packages] *******************************************
ok: [servera.lab.example.com] => {
    "ansible_facts.packages": {
        "NetworkManager": [
            {
                "arch": "x86_64",
                "epoch": 1,
                "name": "NetworkManager",
                "release": "14.el8",
                "source": "rpm",
                "version": "1.14.0"
            }
        ],
...output omitted...
        "zlib": [
            {
                "arch": "x86_64",
                "epoch": null,
                "name": "zlib",
                "release": "10.el8",
                "source": "rpm",
                "version": "1.2.11"
            }
        ]
    }
}

TASK [Display NetworkManager version] ************************************
ok: [servera.lab.example.com] => {
    "msg": "Version 1.14.0"
}

PLAY RECAP ***************************************************************
servera.lab.example.com    : ok=4    changed=0    unreachable=0    failed=0
```

查看用于管理软件包的其他模块

`yum` Ansible 模块适用于使用 Yum 软件包管理器的受管主机。对于其他软件包管理器，Ansible 通常提供一个专用模块。例如，`dnf` 模块可管理利用 DNF 软件包管理器的操作系统（如 Fedora）中的软件包。`apt` 模块使用 Debian 或 Ubuntu 中可用的 APT 软件包工具。`win_package` 模块可以在 Microsoft Windows 系统中安装软件。

海量视频题库 in.ytnub.com QQ:5565462 www.52mvvt.com

以下 playbook 使用条件在由红帽企业 Linux 和 Fedora 系统组成的环境中选择适当的模块。

```
---
- name: Install the required packages on the web servers
  hosts: webservers
  tasks:
    - name: Install httpd on RHEL
      yum:
        name: httpd
        state: present
      when: "ansible_distribution == 'RedHat'"

    - name: Install httpd on Fedora
      dnf:
        name: httpd
        state: present
      when: "ansible_distribution == 'Fedora'"
```

作为替代选择，通用的 package 模块可自动检测并使用受管主机上可用的软件包管理器。使用 package 模块时，您可以按照如下所示重写前面的 playbook。

```
---
- name: Install the required packages on the web servers
  hosts: webservers
  tasks:
    - name: Install httpd
      package:
        name: httpd
        state: present
```

但请注意，package 模块不支持更为专业的模块所提供的所有功能。此外，操作系统通常对它们提供的软件包使用不同的名称。例如，用于安装 Apache HTTP Server 的软件包在红帽企业 Linux 中名为 httpd，在 Ubuntu 中则名为 apache2。在这种情况下，您仍需要一个条件来选择正确的软件包名称，具体取决于受管主机的操作系统。

使用红帽订阅管理注册和管理系统

为了让您的新红帽企业 Linux 系统获得产品订阅权利，Ansible 提供了 redhat_subscription 和 rhsm_repository 模块。这些模块与受管主机上的红帽订阅管理工具进行接口。

注册和订阅新系统

使用红帽订阅管理工具时，执行的前两个任务通常是注册新系统和附加可用订阅。

如果没有 Ansible，您可以使用 **subscription-manager** 命令来执行这些任务：

```
[user@host ~]$ subscription-manager register --username=yourusername
> --password=yourpassword
[user@host ~]$ subscription-manager attach --pool=poolID
```

请记住，您可通过 **subscription-manager list --available** 命令列出帐户中可用的池。

redhat_subscription Ansible 模块可在一个任务中执行注册和订阅。

海量视频题库 mvitpub.com QQ:5565462
www.52mvit.com

```
- name: Register and subscribe the system
  redhat_subscription:
    username: yourusername
    password: yourpassword
    pool_ids: poolID
    state: present
```

state 关键字设为 **present** 时表示注册并订阅系统。如果设为 **absent**，模块将取消注册系统。

启用红帽软件存储库

订阅之后，下一任务是在新系统上启用红帽软件存储库。

如果没有 Ansible，您通常要执行 **subscription-manager** 命令来实现这一目标：

```
[user@host ~]$ subscription-manager repos \
> --enable "rhel-8-for-x86_64-baseos-rpms" \
> --enable "rhel-8-for-x86_64-baseos-debug-rpms"
```

请记住，您可以使用 **subscription-manager repos --list** 命令列出可用的存储库。

对于 Ansible，请使用 rhsm_repository 模块：

```
- name: Enable Red Hat repositories
  rhsm_repository:
    name:
      - rhel-8-for-x86_64-baseos-rpms
      - rhel-8-for-x86_64-baseos-debug-rpms
    state: present
```

配置 YUM 存储库

为了在受管主机上启用对第三方存储库的支持，Ansible 提供了 yum_repository 模块。

声明 Yum 存储库

运行以下 playbook 时，它将声明一个位于 servera.lab.example.com 的新存储库。

```
---
- name: Configure the company Yum repositories
  hosts: servera.lab.example.com
  tasks:
    - name: Ensure Example Repo exists
      yum_repository:
        file: example        ❶
        name: example-internal
        description: Example Inc. Internal YUM repo
        baseurl: http://materials.example.com/yum/repository/
        enabled: yes
        gpgcheck: yes        ❷
        state: present       ❸
```

海量视频题库 myitpub.com QQ:5565462
www.52myit.com

❶ file 关键字提供要在 **/etc/yum.repos.d/** 目录下创建的文件的名称。模块将自动为该名称添加 **.repo** 扩展名。
❷ 通常，软件提供商使用 GPG 密钥对 RPM 软件包进行数字签名。通过将 gpgcheck 关键字设为 **yes**，RPM 系统将确认软件包已由相应的 GPG 密钥签名以验证软件包的完整性。如果 GPG 签名不匹配，它将拒绝安装软件包。使用 rpm_key Ansible 模块安装所需的 GPG 公钥，稍后将介绍这一模块。
❸ 当 state 关键字设为 **present** 时，Ansible 会创建或更新 **.repo** 文件。当 state 设为 **absent** 时，Ansible 会删除该文件。

servera.lab.example.com 上生成的 **/etc/yum.repos.d/example.repo** 文件如下方所示。

```
[example-internal]
baseurl = http://materials.example.com/yum/repository/
enabled = 1
gpgcheck = 1
name = Example Inc. Internal YUM repo
```

yum_repository 模块将大多数 Yum 存储库配置参数公开为关键字。运行 **ansible-doc yum_repository** 命令查看其他参数和 playbook 示例。

> **注意**
> 一些第三方存储库将一个配置文件和 GPG 公钥作为 RPM 软件包的一部分提供，该软件包可以使用 **yum install** 命令来下载和安装。例如， Extra Packages for Enterprise Linux (EPEL) 项目提供了 https://dl.fedoraproject.org/pub/epel/epel-release-latest-VER.noarch.rpm 软件包，该软件包会部署 **/etc/yum.repos.d/epel.repo** 配置文件。对于此存储库，请使用 yum Ansible 模块来安装 EPEL 软件包，而不要使用 yum_repository 模块。

导入 RPM GPG 密钥

当 yum_repository 模块中的 gpgcheck 关键字设为 **yes** 时，您还需要在受管主机上安装 GPG 密钥。下例中的 rpm_key 模块会在 servera.lab.example.com 上部署托管于远程 Web 服务器的 GPG 公钥。

```
---
- name: Configure the company Yum repositories
  hosts: servera.lab.example.com
  tasks:
    - name: Deploy the GPG public key
      rpm_key:
        key: http://materials.example.com/yum/repository/RPM-GPG-KEY-example
        state: present

    - name: Ensure Example Repo exists
      yum_repository:
        file: example
        name: example-internal
        description: Example Inc. Internal YUM repo
        baseurl: http://materials.example.com/yum/repository/
```

海量视频题库 myitpub.com QQ:556462 www.52myit.com

```
enabled: yes
gpgcheck: yes
state: present
```

参考文献

yum(8)、**yum.conf**(5) 和 **subscription-manager**(8) 帮助手册

yum — 使用 yum 软件包管理器管理软件包 — Ansible 文档
https://docs.ansible.com/ansible/latest/modules/yum_module.html

package_facts — 作为事实的软件包信息 — Ansible 文档
https://docs.ansible.com/ansible/latest/modules/package_facts_module.html

redhat_subscription — 使用 subscription-manager 命令管理 RHSM 注册和订阅 — Ansible 文档
https://docs.ansible.com/ansible/latest/modules/redhat_subscription_module.html

rhsm_repository — 使用 subscription-manager 命令管理 RHSM 存储库 — Ansible 文档
https://docs.ansible.com/ansible/latest/modules/rhsm_repository_module.html

yum_repository — 添加或删除 YUM 存储库 — Ansible 文档
https://docs.ansible.com/ansible/latest/modules/yum_repository_module.html

rpm_key — 从 rpm 数据库添加或删除 gpg 密钥 — Ansible 文档
https://docs.ansible.com/ansible/latest/modules/rpm_key_module.html

海量视频题库 myitpub.com QQ:5565462
www.52myit.com

指导练习

管理软件和订阅

在本练习中，您将配置一个新的 Yum 存储库，并在受管主机上安装该存储库中的软件包。

成果

您应能够：

- 使用 `yum_repository` 模块来配置 yum 存储库。
- 使用 `rpm_key` 模块来管理 RPM GPG 密钥。
- 使用 `package_facts` 模块来获取主机上安装的软件包的信息。

在你开始之前

在 `workstation` 上运行实验起始脚本，以确认环境已就绪，可开始实验。该脚本将创建名为 **`system-software`** 的工作目录，并为它填充 Ansible 配置文件、主机清单和实验文件。

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

场景概述

您的组织要求所有主机都安装 example-motd 软件包。这个软件包由内部的 Yum 存储库提供，您的组织维护此存储库来托管内部开发的软件包。

您的任务是编写一个 playbook，以确保远程主机上安装了 example-motd 软件包。该 playbook 必须确保配置了内部 Yum 存储库。

该存储库位于 `http://materials.example.com/yum/repository`。所有 RPM 软件包都使用组织 GPG 密钥对签名。GPG 公钥位于 `http://materials.example.com/yum/repository/RPM-GPG-KEY-example`。

1. 以 `student` 用户身份，在 `workstation` 上更改到 **`/home/student/system-software`** 工作目录。

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

2. 开始编写 **`repo_playbook.yml`** playbook。在 playbook 中定义一个以所有主机为目标的 play。添加 **`vars`** 子句，以定义单个变量 `custom_pkg` 并将值设为 **`example-motd`**。将 **`tasks`** 子句添加到 playbook 中。

   该 playbook 现在包含：

海量视频题库 myitpub.com QQ:556546 2
www.52myit.com

```
---
- name: Repository Configuration
  hosts: all
  vars:
    custom_pkg: example-motd
  tasks:
```

3. 在 playbook 中添加两个任务。

在第一个任务中，使用 `package_facts` 模块收集远程主机上安装的软件包的信息。此任务将填充 `ansible_facts.packages` 事实。

在第二个任务中，使用 `debug` 模块通过相关的 `custom_pkg` 变量来显示已经安装的包的版本。仅在 `ansible_facts.packages` 事实中找到自定义软件包时执行此任务。

执行 **`repo_playbook.yml`** playbook。

3.1. 在 playbook 中添加第一个任务。使用值 **`auto`**，配置 `package_facts` 模块的 `manager` 关键字。第一个任务包含下列内容：

```
- name: Gather Package Facts
  package_facts:
    manager: auto
```

3.2. 在 playbook 中添加第二个任务，以使用 `debug` 模块来显示 `ansible_facts.packages[custom_pkg]` 变量的值。在任务中添加 **`when`** 子句，以检查 `custom_pkg` 变量的值是否包含在 `ansible_facts.packages` 变量中。第二个任务包含下列内容：

```
- name: Show Package Facts for the custom package
  debug:
    var: ansible_facts.packages[custom_pkg]
  when: custom_pkg in ansible_facts.packages
```

3.3. 执行 playbook：

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

PLAY [Repository Configuration] ************************************************

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

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

TASK [Show Package Facts for the custom package] *****************************
skipping: [servera.lab.example.com]

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

海量视频题库 myitpub.com QQ:5565462
www.52myit.com

调试任务已被跳过，因为远程主机上没有安装 example-motd 软件包。

4. 添加第三个任务，以使用 `yum_repository` 模块来确保远程主机上配置了内部 yum 存储库。确保：

   - 存储库的配置存储在文件 **`/etc/yum.repos.d/example.repo`**中
   - 存储库 ID 为 **`example-internal`**
   - 基础 URL 为 `http://materials.example.com/yum/repository`
   - 存储库已配置为检查 RPM GPG 签名
   - 存储库描述为 **`Example Inc. Internal YUM repo`**

   第三个任务包含下列内容：

```
- name: Ensure Example Repo exists
  yum_repository:
    name: example-internal
    description: Example Inc. Internal YUM repo
    file: example
    baseurl: http://materials.example.com/yum/repository/
    gpgcheck: yes
```

5. 在 play 中添加第四个任务，以使用 `rpm_key` 模块来确保远程主机上存在存储库公钥。存储库公钥 URL 为 `http://materials.example.com/yum/repository/RPM-GPG-KEY-example`。

   第四个任务如下所示：

```
- name: Ensure Repo RPM Key is Installed
  rpm_key:
    key: http://materials.example.com/yum/repository/RPM-GPG-KEY-example
    state: present
```

6. 添加第五个任务，以确保远程主机上安装了 `custom_pkg` 变量引用的软件包。

   第五个任务如下所示：

```
- name: Install Example motd package
  yum:
    name: "{{ custom_pkg }}"
    state: present
```

7. 远程主机上安装新软件包时，不更新 `ansible_facts.packages`事实。

   复制第二个任务，并将它作为 play 中的第六个任务添加。执行该 playbook，并验证 `ansible_facts.packages` 事实不包含远程主机上安装的 example-motd 的信息。

   7.1. 第六个任务包含第二个任务的副本：

海量视频题库 myitpub.com QQ:556462
www.52myit.com

```
    - name: Show Package Facts for the custom package
      debug:
        var: ansible_facts.packages[custom_pkg]
      when: custom_pkg in ansible_facts.packages
```

整个 playbook 现在如下方所示：

```
---
- name: Repository Configuration
  hosts: all
  vars:
    custom_pkg: example-motd
  tasks:
    - name: Gather Package Facts
      package_facts:
        manager: auto

    - name: Show Package Facts for the custom package
      debug:
        var: ansible_facts.packages[custom_pkg]
      when: custom_pkg in ansible_facts.packages

    - name: Ensure Example Repo exists
      yum_repository:
        name: example-internal
        description: Example Inc. Internal YUM repo
        file: example
        baseurl: http://materials.example.com/yum/repository/
        gpgcheck: yes

    - name: Ensure Repo RPM Key is Installed
      rpm_key:
        key: http://materials.example.com/yum/repository/RPM-GPG-KEY-example
        state: present

    - name: Install Example motd package
      yum:
        name: "{{ custom_pkg }}"
        state: present

    - name: Show Package Facts for the custom package
      debug:
        var: ansible_facts.packages[custom_pkg]
      when: custom_pkg in ansible_facts.packages
```

7.2. 执行 playbook。

```
[student@workstation system-software]$ ansible-playbook repo_playbook.yml
PLAY [Repository Configuration] ***********************************************

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

海量视频题库 myitpub.com QQ:5565462
www.52myit.com

```
TASK [Gather Package Facts] ***************************************************
ok: [servera.lab.example.com]❶

TASK [Show Package Facts for the custom package] ****************************
skipping: [servera.lab.example.com]

TASK [Ensure Example Repo exists] *********************************************
changed: [servera.lab.example.com]

TASK [Ensure Repo RPM Key is Installed] ***************************************
changed: [servera.lab.example.com]

TASK [Install Example motd package] *******************************************
changed: [servera.lab.example.com]

TASK [Show Package Facts for the custom package] ****************************
skipping: [servera.lab.example.com]❷

PLAY RECAP ********************************************************************
servera.lab.example.com     : ok=5    changed=3    unreachable=0    failed=0
skipped=2    rescued=0    ignored=0
```

❶ **Gather Package Facts** 任务决定 ansible_facts.packages 事实中包含的数据。

❷ 该任务已被跳过，因为在 **Gather Package Facts** 任务后安装了 example-motd 软件包。

8. 紧接在 **Install Example motd package** 任务后插入一个任务，以使用 package_facts 模块更新软件包事实。使用值 **auto** 设置模块的 manager 关键字。完整的 playbook 如下方所示：

```
---
- name: Repository Configuration
  hosts: all
  vars:
    custom_pkg: example-motd
  tasks:
    - name: Gather Package Facts
      package_facts:
        manager: auto

    - name: Show Package Facts for the custom package
      debug:
        var: ansible_facts.packages[custom_pkg]
      when: custom_pkg in ansible_facts.packages

    - name: Ensure Example Repo exists
      yum_repository:
        name: example-internal
        description: Example Inc. Internal YUM repo
        file: example
        baseurl: http://materials.example.com/yum/repository/
```

海量视频题库 myitpub.com QQ:5565462 www.52myit.com

```
      gpgcheck: yes

    - name: Ensure Repo RPM Key is Installed
      rpm_key:
        key: http://materials.example.com/yum/repository/RPM-GPG-KEY-example
        state: present

    - name: Install Example motd package
      yum:
        name: "{{ custom_pkg }}"
        state: present

    - name: Gather Package Facts
      package_facts:
        manager: auto

    - name: Show Package Facts for the custom package
      debug:
        var: ansible_facts.packages[custom_pkg]
      when: custom_pkg in ansible_facts.packages
```

9. 使用一个 Ansible 临时命令，删除上次执行 Playbook 期间安装的 example-motd 软件包。执行插入了 `package_facts` 任务的 playbook，并使用其输出来验证 example-motd 软件包的安装。

9.1. 若要从所有主机删除 example-motd 软件包，可使用 **`-m yum`** 和 **`-a 'name=example-motd state=absent'`** 选项来运行 **`ansible all`** 命令。

```
[student@workstation system-software]$ ansible all -m yum \
> -a 'name=example-motd state=absent'
servera.lab.example.com | CHANGED => {
...output omitted...
    "changed": true,
    "msg": "",
    "rc": 0,
    "results": [
        "Removed: example-motd-1.0-1.el7.x86_64"
    ]
...output omitted...
```

9.2. 执行 playbook。

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

PLAY [Repository Configuration] ***********************************************

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

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

TASK [Show Package Facts for the custom package] ******************************
```

海量视频题库 myitpub.com QQ:5565462
www.52myit.com

```
skipping: [servera.lab.example.com]❶

...output omitted...

TASK [Install Example motd package] ****************************************
changed: [servera.lab.example.com]❷

TASK [Gather Package Facts] ************************************************
ok: [servera.lab.example.com]❸

TASK [Show Package Facts for example-motd] *********************************
ok: [servera.lab.example.com] => {
    "ansible_facts.packages[custom_pkg]": [❹
        {
            "arch": "x86_64",
            "epoch": null,
            "name": "example-motd",
            "release": "1.el7",
            "source": "rpm",
            "version": "1.0"
        }
    ]
}

PLAY RECAP *****************************************************************
servera.lab.example.com    : ok=7    changed=1    unreachable=0    failed=0
skipped=1    rescued=0    ignored=0
```

❶ 不存在针对 example-motd 软件包的软件包事实，因为远程主机上没有安装该软件包。

❷ 作为此任务的结果，安装了 example-motd 软件包，如 **changed** 状态所示。

❸ 此任务使用有关 example-motd 软件包的信息更新了软件包事实。

❹ example-motd 软件包事实存在，并且指出仅安装了一个 example-motd 软件包。所安装软件包的版本是 **1.0**。

完成

在 workstation 上，运行 **lab system-software finish** 脚本来清理本练习中创建的资源。

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

本引导式练习到此结束。

海量视频题库 myitpub.com QQ:556546 2
www.52myit.com

管理用户和身份验证

培训目标

学完本节后，您应能够：

· 在多个服务器上调配多个用户帐户。

· 管理 Linux 用户和组，配置 SSH，以及修改受管主机上的 Sudo 配置。

USER 模块

Ansible user 模块允许您管理远程主机上的用户帐户。您可以管理许多参数，包括删除用户、设置主目录、设置系统帐户的 UID，以及管理密码和关联的分组。若要创建可以登录计算机的用户，您需要为 password 参数提供哈希密码。请参见参考资料部分中“如何为 user 模块生成加密密码”的链接。

user 模块示例

```
- name: Add new user to the development machine and assign the appropriate groups.
  user:
    name: devops_user ❶
    shell: /bin/bash ❷
    groups: sys_admins, developers ❸
    append: yes
```

❶ **name** 参数是 user 模块中的唯一必要参数，通常是服务帐户或用户帐户。

❷ **shell** 参数用于设置用户的 shell（可选）。在其他操作系统上，默认 shell 由正在使用的工具决定。

❸ **groups** 参数及 **append** 参数告知计算机，我们要为此用户附加 groups sys_asmins 和 developers 组。如果不使用 append 参数，则组将覆盖现有的组。

创建用户时，您可以将其指定为 **generate_ssh_key**。这不会覆盖现有的 SSH 密钥。

使用 user 模块生成 ssh 密钥的示例

```
- name: Create a SSH key for user1
  user:
    name: user1
    generate_ssh_key: yes
    ssh_key_bits: 2048
    ssh_key_file: .ssh/id_my_rsa
```

注意

user 模块也提供一些返回值。Ansible 模块可以取用返回值并将它们注册到变量中。您可通过 ansible-doc 或在主要文档网站上了解更多信息。

海量视频题库 myitpub.com QQ:556462 www.52myit.com

一些常用的参数

参数	注释
comment	（可选）设置用户帐户的描述。
group	（可选）设置用户的主要组。
groups	多个组的列表。设置为空值时，将删除除主组之外的所有组。
home	（可选）设置用户的主目录。
create_home	取布尔值 yes 或 no。如果值设置为 yes，将为用户创建主目录。
system	在创建 state=present 的帐户时，将此参数设为 yes 可使该用户成为系统帐户。无法对现有用户更改此设置。
uid	设置用户的 UID。

GROUP 模块

group 模块允许您管理（添加、删除或修改）受管主机上的组。您需要具有 **groupadd**、**groupdel** 或 **groupmod**。对于 Windows 目标，请使用 **win_group** 模块。

group 模块示例

```
- name: Verify that auditors group exists
  group:
    name: auditors
    state: present
```

group 模块参数

参数	注释
gid	为组设置的可选 GID。
local	强制在实施它的平台上使用“本地”命令替代选择。
name	要管理的组的名称。
state	远程主机上是否应存在该组。
system	如果设置为 yes，则表示创建的组是系统组。

已知主机模块

如果要管理大量主机密钥，则需要使用**known_hosts**模块。**known_hosts** 模块允许您在受管主机上的 known_hosts 文件中添加或删除主机密钥。

海量视频题库 myitpub.com QQ:5565462 www.52myit.com

known_hosts 任务示例

```
- name: copy host keys to remote servers
  known_hosts:
    path: /etc/ssh/ssh_known_hosts
    name: user1
    key: "{{ lookup('file', 'pubkeys/user1') }}"❶
```

❶ **lookup** 插件允许 Ansible 访问外部来源的数据。

AUTHORIZED_KEY 模块

authorized_key 模块允许您为各个用户帐户添加或删除 SSH 授权密钥。在向大量服务器添加和减少用户时，您需要能够管理 ssh 密钥。

authorized_key 任务示例

```
- name: Set authorized key
  authorized_key:
    user: user1
    state: present
    key: "{{ lookup('file', '/home/user1/.ssh/id_rsa.pub') }}❶
```

❶ 也可以从 url 提取密钥：https://github.com/user1.keys。

参考文献

user 模块 Ansible 文档
http://docs.ansible.com/ansible/latest/modules/user_module.html#user-module

如何为 user 模块生成加密密码
https://docs.ansible.com/ansible/latest/reference_appendices/faq.html#how-do-i-generate-crypted-passwords-for-the-user-module

group 模块 Ansible 文档
https://docs.ansible.com/ansible/latest/modules/group_module.html#group-module

SSH 已知主机模块 Ansible 文档
https://docs.ansible.com/ansible/latest/modules/known_hosts_module.html#known-hosts-module

Authorized_key 模块 Ansible 文档
https://docs.ansible.com/ansible/latest/modules/authorized_key_module.html#authorized-key-module

Lookup 插件 Ansible文档
https://docs.ansible.com/ansible/latest/plugins/lookup.html?highlight=lookup

海量视频题库 myitpub.com QQ:556462
www.52myit.com

指导练习

管理用户和身份验证

在本练习中，您将在受管主机上创建多个用户，并为其填充授权的 SSH 密钥。

成果

您应能够：

- 创建一个新用户组。
- 使用 `user` 模块管理用户。
- 使用 `authorized_key` 模块填充 SSH 授权密钥。
- 使用 `lineinfile` 模块修改 **`sudoers`** 和 **`sshd_config`** 文件。

在你开始之前

在 `workstation` 上，运行实验起始脚本，以确认环境已就绪，可开始实验。该脚本将创建名为 **`system-users`** 的工作目录，并为它填充 Ansible 配置文件、主机清单和一些实验文件。

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

场景概述

您的组织要求所有主机都具有相同的本地用户。这些用户应从属于 `webadmin` 用户组，该组能够在不指定密码的前提下使用 **`sudo`** 命令。此外，用户的 SSH 公钥应该在环境中分发，并且不允许 `root` 用户使用 SSH 直接登录。

您的任务是编写一个 playbook，以确保远程主机上存在用户和用户组。该 playbook 必须确保用户可以使用授权 SSH 密钥登录，并且能够在不指定密码的前提下使用 **`sudo`**，同时还确保 `root` 用户无法使用 SSH 直接登录。

1. 以 `student` 用户身份，在 `workstation` 上更改到 **`/home/student/system-users`** 工作目录。

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

2. 查看现有的 **`vars/users_vars.yml`** 变量文件。

```
[student@workstation system-users]$ cat vars/users_vars.yml
---
users:
  - username: user1
    groups: webadmin
  - username: user2
    groups: webadmin
```

海量视频题库 myitpub.com QQ:5565462
www.52myit.com

```
  - username: user3
    groups: webadmin
  - username: user4
    groups: webadmin
  - username: user5
    groups: webadmin
```

它使用了 `username` 变量名称来设置正确的用户名，并且使用 `groups` 变量来定义用户应属于的其他组。

3. 开始编写 **`users.yml`** playbook。在 playbook 中定义一个以 `webservers` 主机组为目标的 play。添加一个 **`vars_files`** 子句来定义 **`vars/users_vars.yml`** 文件名的位置，该文件已为您创建好，其包含本练习所需的所有用户名。将 **`tasks`** 子句添加到 playbook 中。

使用文本编辑器创建 **`users.yml`** playbook。Playbook 应包含如下内容：

```
---
- name: Create multiple local users
  hosts: webservers
  vars_files:
    - vars/users_vars.yml
  tasks:
```

4. 在 playbook 中添加两个任务。

在第一个任务中，使用 `group` 模块在远程主机上创建 `webadmin` 用户组。此任务将创建 `webadmin` 用户组。

在第二个任务中，使用 `user` 模块从 **`vars/users_vars.yml`** 文件创建用户。

执行 **`users.yml`** playbook。

4.1. 在 playbook 中添加第一个任务。第一个任务包含下列内容：

```
  - name: Add webadmin group
    group:
      name: webadmin
      state: present
```

4.2. 在 playbook 中添加第二个任务，以使用 `user` 模块来创建用户。在任务中添加 **`loop: "{{ users }}"`** 子句，在变量文件中循环遍历 **`vars/users_vars.yml`** 文件中找到的每个用户名。对于用户的 **`name:`**，使用 `item.username` 作为变量名称。这样，变量文件可以包含可能对创建用户有用的其他信息，例如用户应从属于的组。第二个任务包含下列内容：

```
  - name: Create user accounts
    user:
      name: "{{ item.username }}"
      groups: webadmin
    loop: "{{ users }}"
```

4.3. 执行 playbook：

海量视频题库 myitpub.com QQ:556546 2
www.52myit.com

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

PLAY [Create multiple local users] *********************************************

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

TASK [Add webadmin group] ******************************************************
changed: [servera.lab.example.com]

TASK [Create user accounts] ****************************************************
changed: [servera.lab.example.com] => (item={u'username': u'user1', u'groups':
 u'webadmin'})
changed: [servera.lab.example.com] => (item={u'username': u'user2', u'groups':
 u'webadmin'})
changed: [servera.lab.example.com] => (item={u'username': u'user3', u'groups':
 u'webadmin'})
changed: [servera.lab.example.com] => (item={u'username': u'user4', u'groups':
 u'webadmin'})
changed: [servera.lab.example.com] => (item={u'username': u'user5', u'groups':
 u'webadmin'})

PLAY RECAP *********************************************************************
servera.lab.example.com    : ok=3    changed=2    unreachable=0    failed=0
```

5. 添加第三个任务，以使用 `authorized_key` 模块来确保 SSH 公钥已正确分发到远程主机上。在 **files** 目录中，每个用户都有一个唯一的 SSH 公钥文件。该模块循环遍历用户列表，利用 `username` 变量查找相应的密钥，并将密钥推送到远程主机。

   第三个任务包含下列内容：

```
- name: Add authorized keys
  authorized_key:
    user: "{{ item.username }}"
    key: "{{ lookup('file', 'files/'+ item.username + '.key.pub') }}"
  loop: "{{ users }}"
```

6. 在 play 中添加第四个任务，以使用 `copy` 模块修改 **sudo** 配置文件并允许 `webadmin` 组成员在远程主机上免密码使用 **sudo**。

   第四个任务如下所示：

```
- name: Modify sudo config to allow webadmin users sudo without a password
  copy:
    content: "%webadmin ALL=(ALL) NOPASSWD: ALL"
    dest: /etc/sudoers.d/webadmin
    mode: 0440
```

海量视频题库 myitpub.com QQ:5565462
www.52myit.com

7. 添加第五个任务，以确保不允许 `root` 用户使用 SSH 直接进行登录。使用 **`notify: "Restart sshd"`** 触发处理程序，以重新启动 SSH。

第五个任务如下所示：

```
- name: Disable root login via SSH
  lineinfile:
    dest: /etc/ssh/sshd_config
    regexp: "^PermitRootLogin"
    line: "PermitRootLogin no"
  notify: Restart sshd
```

8. 在变量文件位置之后的第一行中，添加新的处理程序定义。将它命名为 **`Restart sshd`**。

8.1. 该处理程序的定义应如下所示：

```
...output omitted...
    - vars/users_vars.yml
  handlers:
  - name: Restart sshd
    service:
      name: sshd
      state: restarted
```

整个 playbook 现在如下方所示：

```
---
- name: Create multiple local users
  hosts: webservers
  vars_files:
    - vars/users_vars.yml
  handlers:
  - name: Restart sshd
    service:
      name: sshd
      state: restarted

  tasks:

  - name: Add webadmin group
    group:
      name: webadmin
      state: present

  - name: Create user accounts
    user:
      name: "{{ item.username }}"
      groups: webadmin
    loop: "{{ users }}"

  - name: Add authorized keys
    authorized_key:
      user: "{{ item.username }}"
      key: "{{ lookup('file', 'files/'+ item.username + '.key.pub') }}"
```

海量视频题库 myitpub.com QQ:556462
www.52myit.com

```
    loop: "{{ users }}"

  - name: Modify sudo config to allow webadmin users sudo without a password
    copy:
      content: "%webadmin ALL=(ALL) NOPASSWD: ALL"
      dest: /etc/sudoers.d/webadmin
      mode: 0440

  - name: Disable root login via SSH
    lineinfile:
      dest: /etc/ssh/sshd_config
      regexp: "^PermitRootLogin"
      line: "PermitRootLogin no"
    notify: "Restart sshd"
```

8.2. 执行 playbook。

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

PLAY [Create multiple local users] ******************************************

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

TASK [Add webadmin group] ***************************************************
ok: [servera.lab.example.com]

TASK [Create user accounts] *************************************************
ok: [servera.lab.example.com] => (item={u'username': u'user1', u'groups':
 u'webadmin'})
ok: [servera.lab.example.com] => (item={u'username': u'user2', u'groups':
 u'webadmin'})
ok: [servera.lab.example.com] => (item={u'username': u'user3', u'groups':
 u'webadmin'})
ok: [servera.lab.example.com] => (item={u'username': u'user4', u'groups':
 u'webadmin'})
ok: [servera.lab.example.com] => (item={u'username': u'user5', u'groups':
 u'webadmin'})

TASK [Add authorized keys] **************************************************
changed: [servera.lab.example.com] => (item={u'username': u'user1', u'groups':
 u'webadmin'})
changed: [servera.lab.example.com] => (item={u'username': u'user2', u'groups':
 u'webadmin'})
changed: [servera.lab.example.com] => (item={u'username': u'user3', u'groups':
 u'webadmin'})
changed: [servera.lab.example.com] => (item={u'username': u'user4', u'groups':
 u'webadmin'})
changed: [servera.lab.example.com] => (item={u'username': u'user5', u'groups':
 u'webadmin'})

TASK [Modify sudo config to allow webadmin users sudo without a password] ***
changed: [servera.lab.example.com]
```

海量视频题库 myitpub.com QQ:5565462
www.52myit.com