5. 运行 **motd.yml** 文件中包含的 playbook。
6. 检查 **motd.yml** 文件中包含的 playbook 是否已正确执行。

评估

在 workstation 上，运行 **lab file-review grade** 脚本来确认是否成功完成本练习。

```
[student@workstation ~]$ lab file-review grade
```

完成

在 workstation 上，运行 **lab file-review finish** 脚本来进行实验后清理。

```
[student@workstation ~]$ lab file-review finish
```

本引导式练习到此结束。

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

解决方案

将文件部署到受管主机

任务执行清单

在本实验中，您将运行一个 playbook，该 playbook 使用 Jinja2 模板在受管主机上创建自定义文件。

成果

您应能够：

- 构建模板文件。
- 在 playbook 中使用模板文件。

在你开始之前

以 `student` 用户身份并使用 `student` 作为密码登录 `workstation`。

在 `workstation` 上，运行 **`lab file-review start`** 命令。这可确保在 `workstation` 上安装 Ansible，创建 **`/home/student/file-review`** 目录，并将 **`ansible.cfg`** 文件下载到该目录。它也会将 **`motd.yml`**、**`motd.j2`**、**`issue`** 和 **`inventory`** 文件下载到 **`/home/student/file-review/files`** 目录。

```
[student@workstation ~]$ lab file-review start
```

> **注意**
> 本练习中使用的所有文件都位于 `workstation` 上的 **`/home/student/file-review/files`** 目录中。

1. 在 **`/home/student/file-review`** 目录中创建一个名为 **`inventory`** 的清单文件。此清单文件定义 **`servers`** 组，该组关联有 `serverb.lab.example.com` 受管主机。

   1.1. 在 workstation 上，更改到 **`/home/student/file-review`** 目录。

```
[student@workstation ~]$ cd ~/file-review/
```

   1.2. 在当前目录中创建 **`inventory`** 文件。此文件配置一个名为 **`servers`** 的组。将 `serverb.lab.example.com` 系统包含在 **`servers`** 组中。

```
[servers]
serverb.lab.example.com
```

2. 识别 `serverb.lab.example.com` 上的事实，其显示系统内存总量以及处理器数目。

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

使用 setup 模块获取 serverb.lab.example.com 受管主机的所有事实的列表。ansible_processor_count 和 ansible_memtotal_mb 事实提供有关受管主机资源限制的信息。

```
[student@workstation file-review]$ ansible serverb.lab.example.com -m setup
serverb.lab.example.com | SUCCESS => {
    "ansible_facts": {
...output omitted...
 "ansible_processor_count": 1,
...output omitted...
 "ansible_memtotal_mb": 821,
...output omitted...
    },
    "changed": false
}
```

3. 在当前目录中为当日消息创建一个名为 **motd.j2** 的模板。当 devops 用户登录 serverb.lab.example.com 时，应显示一条消息，其中显示系统的总内存和处理器数。使用 ansible_facts['memtotal_mb'] 和 ansible_facts['processor_count'] 事实为该消息提供内存信息。

```
System total memory: {{ ansible_facts['memtotal_mb'] }} MiB.
System processor count: {{ ansible_facts['processor_count'] }}
```

4. 在当前目录中创建一个名为 **motd.yml** 的新 playbook 文件。使用 template 模块，配置之前创建的 **motd.j2** Jinja2 模板文件，以映射到受管主机上的文件 **/etc/motd**。此文件的所有者和组为 root 用户，其权限为 0644。使用 stat 和 debug 模块，创建相应的任务来验证受管主机上是否存在 **/etc/motd** 并且显示 **/etc/motd** 的文件信息。使用 copy 模块，将 **files/issue** 放入受管主机上的 **/etc/** 目录，并且使用与 **/etc/motd** 相同的所有权和权限。使用 file 模块确保 **/etc/issue.net** 是受管主机上 **/etc/issue** 的符号链接。配置该 playbook 使其使用 devops 用户，并将 become 参数设置为 **true**。

```
---
- name: Configure system
  hosts: all
  remote_user: devops
  become: true
  tasks:
    - name: Configure a custom /etc/motd
      template:
        src: motd.j2
        dest: /etc/motd
        owner: root
        group: root
        mode: 0644

    - name: Check file exists
      stat:
        path: /etc/motd
      register: motd

    - name: Display stat results
      debug:
```

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

```
        var: motd

    - name: Copy custom /etc/issue file
      copy:
        src: files/issue
        dest: /etc/issue
        owner: root
        group: root
        mode: 0644

    - name: Ensure /etc/issue.net is a symlink to /etc/issue
      file:
        src: /etc/issue
        dest: /etc/issue.net
        state: link
        owner: root
        group: root
        force: yes
```

5. 运行 **motd.yml** 文件中包含的 playbook。

5.1. 在运行 playbook 之前，请使用 **ansible-playbook --syntax-check** 命令验证其语法。如果报告任何错误，请更正后再继续下一步。您应看到类似于下文的输出：

```
[student@workstation file-review]$ ansible-playbook --syntax-check motd.yml

playbook: motd.yml
```

5.2. 运行 **motd.yml** 文件中包含的 playbook。

```
[student@workstation file-review]$ ansible-playbook motd.yml

PLAY [Configure system] ********************************************************

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

TASK [Configure a custom /etc/motd] ********************************************
changed: [serverb.lab.example.com]

TASK [Check file exists] *******************************************************
ok: [serverb.lab.example.com]

TASK [Display stat results] ****************************************************
ok: [serverb.lab.example.com] => {
    "motd": {
        "changed": false,
        "failed": false,
...output omitted...

TASK [Copy custom /etc/issue file] *********************************************
changed: [serverb.lab.example.com]

TASK [Ensure /etc/issue.net is a symlink to /etc/issue] ************************
```

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

```
changed: [serverb.lab.example.com]

PLAY RECAP *********************************************************************
serverb.lab.example.com    : ok=6    changed=3    unreachable=0    failed=0
```

6. 检查 **motd.yml** 文件中包含的 playbook 是否已正确执行。

   以 devops 用户身份登录 serverb.lab.example.com，并验证登录时是否显示 **/etc/motd** 和 **/etc/issue** 内容。完成时注销。

```
[student@workstation file-review]$ ssh devops@serverb.lab.example.com
*------------------------------ PRIVATE SYSTEM ------------------------------*
*   Access to this computer system is restricted to authorised users only.   *
*                                                                            *
*      Customer information is confidential and must not be disclosed.       *
*----------------------------------------------------------------------------*
System total memory: 821 MiB.
System processor count: 1
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Thu Apr 25 22:09:33 2019 from 172.25.250.9
[devops@serverb ~]$ logout
```

评估

在 workstation 上，运行 **lab file-review grade** 脚本来确认是否成功完成本练习。

```
[student@workstation ~]$ lab file-review grade
```

完成

在 workstation 上，运行 **lab file-review finish** 脚本来进行实验后清理。

```
[student@workstation ~]$ lab file-review finish
```

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

总结

在本章中，您学到了：

- `Files` 模块库包含的模块允许您完成与文件管理相关的大多数任务，如创建、复制、编辑和修改文件的权限和其他属性。
- 您可以使用 Jinja2 模板动态构建文件以进行部署。
- Jinja2 模板由两个元素组成，即变量和表达式。在呈现 Jinja2 模板时，这些变量和表达式被替换为值。
- 通过 Jinja2 过滤器，模板表达式可以从一种数据格式转换为另一种数据格式。

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

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

章 7

管理大项目

目标

编写针对更大、更复杂的项目进行优化的 playbook。

培训目标

- 编写复杂的主机模式，以便高效地为 play 或临时命令选择主机。
- 描述动态清单的性质和用途，然后安装和使用现有脚本作为 Ansible 动态清单来源。
- 调整 Ansible 向受管主机开放的同时连接数量，以及 Ansible 如何通过 play 任务处理受管主机组。
- 通过无条件或基于条件测试导入或包含来自外部文件的 playbook 和任务来管理大型 playbook。

章节

- 利用主机模式选择主机（及引导式练习）
- 管理动态清单（及引导式练习）
- 配置并行（及引导式练习）
- 包含和导入文件（及引导式练习）

实验

- 管理大项目

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

利用主机模式选择主机

培训目标

学完本节后，您将能够编写复杂的主机模式，以便高效地为 play 或临时命令选择主机。

引用清单主机

主机模式用于指定要作为 play 或临时命令的目标的主机。在最简单的形式中，清单中受管主机或主机组的名称就是指定该主机或主机组的主机模式。

您已在本课程中使用主机模式。在 play 中，`hosts` 指定要针对其运行 play 的受管主机。对于临时命令，以命令行参数形式将主机模式提供给 **`ansible`** 命令。

仔细地使用主机模式并设有适当的清单组，而不在 play 的任务中设置复杂的条件，通常更容易控制作为 play 目标的主机。因此，对主机模式有一个可靠的理解非常重要。

本节中将通篇使用以下示例清单来演示主机模式。

```
[student@controlnode ~]$ cat myinventory
web.example.com
data.example.com

[lab]
labhost1.example.com
labhost2.example.com

[test]
test1.example.com
test2.example.com

[datacenter1]
labhost1.example.com
test1.example.com

[datacenter2]
labhost2.example.com
test2.example.com

[datacenter:children]
datacenter1
datacenter2

[new]
192.168.2.1
192.168.2.2
```

要演示如何解析主机模式，您将执行 Ansible Playbook **`playbook.yml`**，使用不同的主机模式来以此示例清单中受管主机的不同子集作为目标。

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

受管主机

最基本的主机模式是单一受管主机名称列在清单中。这将指定该主机是清单中 **ansible** 命令要执行操作的唯一主机。

在该 playbook 运行时，第一个 **Gathering Facts** 任务应在与主机模式匹配的所有受管主机上运行。此任务期间的故障可能导致受管主机从 play 中移除。

如果清单中明确列出了 IP 地址，而不是主机名，则可以将其用作主机模式。如果 IP 地址未列在清单中，您就无法用它来指定主机，即使该 IP 地址会在 DNS 中解析到这个主机名。

以下示例演示了如何使用主机模式来引用清单中包含的 IP 地址。

```
[student@controlnode ~]$ cat playbook.yml
---
- hosts: 192.168.2.1
...output omitted...

[student@controlnode ~]$ ansible-playbook playbook.yml

PLAY [Test Host Patterns] ******************************************************

TASK [Gathering Facts] *********************************************************
ok: [192.168.2.1]
...output omitted...
```

注意

在清单中通过 IP 地址引用受管主机存在一个问题，那就是难以记住 play 或临时命令所针对的主机使用了哪个 IP 地址。但是，如果主机没有可解析的主机名，您可能必须按 IP 地址指定主机以进行连接。

可以通过设置 ansible_host 主机变量，在清单中将某一别名指向特定的 IP 地址。例如，您的清单中可以有一个名为 **dummy.example** 的主机，然后通过创建含有以下主机变量的 **host_vars/dummy.example** 文件，将使用该名称的连接指向 IP 地址 192.168.2.1：

```
ansible_host: 192.168.2.1
```

使用组指定主机

您已使用清单主机组作为主机模式。当组名称用作主机模式时，它指定 Ansible 将对属于该组的成员的主机执行操作。

```
[student@controlnode ~]$ cat playbook.yml
---
- hosts: lab
...output omitted...
[student@controlnode ~]$ ansible-playbook playbook.yml

PLAY [Test Host Patterns] ******************************************************

TASK [Gathering Facts] *********************************************************
```

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

```
ok: [labhost1.example.com]
ok: [labhost2.example.com]
...output omitted...
```

记住有一个名为 all 的特别组，它匹配清单中的所有受管主机。

```
[student@controlnode ~]$ cat playbook.yml
---
- hosts: all
...output omitted...
[student@controlnode ~]$ ansible-playbook playbook.yml

PLAY [Test Host Patterns] ******************************************************

TASK [Gathering Facts] *********************************************************
ok: [labhost2.example.com]
ok: [test2.example.com]
ok: [web.example.com]
ok: [data.example.com]
ok: [labhost1.example.com]
ok: [192.168.2.1]
ok: [test1.example.com]
ok: [192.168.2.2]
```

还有一个名为 ungrouped 的特别组，它包括清单中不属于任何其他组的所有受管主机：

```
[student@controlnode ~]$ cat playbook.yml
---
- hosts: ungrouped
...output omitted...
[student@controlnode ~]$ ansible-playbook playbook.yml

PLAY [Test Host Patterns] ******************************************************

TASK [Gathering Facts] *********************************************************
ok: [web.example.com]
ok: [data.example.com]
```

使用通配符匹配多个主机

若要达成与 all 主机模式相同的目标，另一种方法是使用星号 (*) 通配符，它将匹配任意字符串。如果主机模式只是带引号的星号，则清单中的所有主机都将匹配。

```
[student@controlnode ~]$ cat playbook.yml
---
- hosts: '*'
...output omitted...
[student@controlnode ~]$ ansible-playbook playbook.yml

PLAY [Test Host Patterns] ******************************************************

TASK [Gathering Facts] *********************************************************
ok: [labhost2.example.com]
```

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

```
ok: [test2.example.com]
ok: [web.example.com]
ok: [data.example.com]
ok: [labhost1.example.com]
ok: [192.168.2.1]
ok: [test1.example.com]
ok: [192.168.2.2]
```

重要

一些在主机模式中使用的字符对 shell 也有意义。通过 **ansible** 使用主机模式从命令行运行临时命令时，这可能会有问题。建议您在命令行中用单引号括起使用的主机模式，防止它们被 shell 意外扩展。

类似地，如果您在 Ansible Playbook 中使用了任何特殊通配符或列表字符，您必须将主机模式放在单引号里，确保能够正确解析主机模式。

```
---
  hosts: '!test1.example.com,development'
```

也可使用星号字符，以匹配包含特定子字符串的受管主机或组。

例如，以下通配符主机模式匹配以 **.example.com** 结尾的所有清单名称：

```
[student@controlnode ~]$ cat playbook.yml
---
- hosts: '*.example.com'
...output omitted...
[student@controlnode ~]$ ansible-playbook playbook.yml

PLAY [Test Host Patterns] ***************************************************

TASK [Gathering Facts] ******************************************************
ok: [labhost1.example.com]
ok: [test1.example.com]
ok: [labhost2.example.com]
ok: [test2.example.com]
ok: [web.example.com]
ok: [data.example.com]
```

以下示例使用通配符主机模式来匹配开头为 **192.168.2.** 的主机或主机组的名称：

```
[student@controlnode ~]$ cat playbook.yml
---
- hosts: '192.168.2.*'
...output omitted...
[student@controlnode ~]$ ansible-playbook playbook.yml

PLAY [Test Host Patterns] ***************************************************
```

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

```
TASK [Gathering Facts] ****************************************************
ok: [192.168.2.1]
ok: [192.168.2.2]
```

以下示例使用通配符主机模式来匹配开头为 **datacenter** 的主机或主机组的名称。

```
[student@controlnode ~]$ cat playbook.yml
---
- hosts: 'datacenter*'
...output omitted...
[student@controlnode ~]$ ansible-playbook playbook.yml

PLAY [Test Host Patterns] *************************************************

TASK [Gathering Facts] ****************************************************
ok: [labhost1.example.com]
ok: [test1.example.com]
ok: [labhost2.example.com]
ok: [test2.example.com]
```

重要

通配符主机模式匹配所有清单名称、主机和主机组。它们不区别名称是 DNS 名、IP 地址还是组，这可能会导致一些意外的匹配。

例如，根据示例清单，比较上一示例中指定 **datacenter*** 主机模式的结果和 **data*** 主机模式的结果：

```
[student@controlnode ~]$ cat playbook.yml
---
- hosts: 'data*'
...output omitted...
[student@controlnode ~]$ ansible-playbook playbook.yml

PLAY [Test Host Patterns] *************************************************

TASK [Gathering Facts] ****************************************************
ok: [labhost1.example.com]
ok: [test1.example.com]
ok: [labhost2.example.com]
ok: [test2.example.com]
ok: [data.example.com]
```

列表

可以通过逻辑列表来引用清单中的多个条目。主机模式的逗号分隔列表匹配符合任何这些主机模式的所有主机。

如果提供受管主机的逗号分隔列表，则所有这些受管主机都将是目标：

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

```
[student@controlnode ~]$ cat playbook.yml
---
- hosts: labhost1.example.com,test2.example.com,192.168.2.2
...output omitted...
[student@controlnode ~]$ ansible-playbook playbook.yml

PLAY [Test Host Patterns] ***************************************************

TASK [Gathering Facts] ******************************************************
ok: [labhost1.example.com]
ok: [test2.example.com]
ok: [192.168.2.2]
```

如果提供组的逗号分隔列表，则属于任何这些组的所有主机都将是目标：

```
[student@controlnode ~]$ cat playbook.yml
---
- hosts: lab,datacenter1
...output omitted...
[student@controlnode ~]$ ansible-playbook playbook.yml

PLAY [Test Host Patterns] ***************************************************

TASK [Gathering Facts] ******************************************************
ok: [labhost1.example.com]
ok: [labhost2.example.com]
ok: [test1.example.com]
```

您也可以混合使用受管主机、主机组和通配符，如下所示：

```
[student@controlnode ~]$ cat playbook.yml
---
- hosts: lab,data*,192.168.2.2
...output omitted...
[student@controlnode ~]$ ansible-playbook playbook.yml

PLAY [Test Host Patterns] ***************************************************

TASK [Gathering Facts] ******************************************************
ok: [labhost1.example.com]
ok: [labhost2.example.com]
ok: [test1.example.com]
ok: [test2.example.com]
ok: [data.example.com]
ok: [192.168.2.2]
```

注意

也可以用冒号 (:) 来取代逗号。不过，逗号是首选的分隔符，特别是将 IPv6 地址用作受管主机名称时。您可能会在较旧的示例中看到冒号语法。

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

如果列表中的某一项以与符号 (&) 开头，则主机必须与该项匹配才能匹配主机模式。它的工作方式类似于逻辑 AND。

例如，根据我们的示例清单，以下主机模式将匹配 **lab** 组中同时也属于 datacenter1 组的计算机：

```
[student@controlnode ~]$ cat playbook.yml
---
- hosts: lab,&datacenter1
...output omitted...
[student@controlnode ~]$ ansible-playbook playbook.yml

PLAY [Test Host Patterns] ***************************************************

TASK [Gathering Facts] ******************************************************
ok: [labhost1.example.com]
```

您也可以通过主机模式 **&lab,datacenter1** 或 **datacenter1,&lab** 指定 **datacenter1** 组中的计算机只有在同时也属于组 lab 时才匹配。

您可以通过在主机模式的前面使用感叹号或惊叹号 (!)，从列表中排除匹配某一模式的主机。它的工作方式类似于逻辑 NOT。

根据示例清单，以下示例匹配 datacenter 组中定义的所有主机，但 **test2.example.com** 除外：

```
[student@controlnode ~]$ cat playbook.yml
---
- hosts: datacenter,!test2.example.com
...output omitted...
[student@controlnode ~]$ ansible-playbook playbook.yml

PLAY [Test Host Patterns] ***************************************************

TASK [Gathering Facts] ******************************************************
ok: [labhost1.example.com]
ok: [test1.example.com]
ok: [labhost2.example.com]
```

上一示例中也可使用模式 **'!test2.example.com,datacenter'** 来获得同样的结果。

最后一个示例演示了使用匹配测试清单中的所有主机的主机模式，datacenter1 组中的受管主机除外。

```
[student@controlnode ~]$ cat playbook.yml
---
- hosts: all,!datacenter1
...output omitted...
[student@controlnode ~]$ ansible-playbook playbook.yml

PLAY [Test Host Patterns] ***************************************************

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

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

```
ok: [data.example.com]
ok: [labhost2.example.com]
ok: [test2.example.com]
ok: [192.168.2.1]
ok: [192.168.2.2]
```

参考文献

使用模式 — Ansible 文档
https://docs.ansible.com/ansible/latest/user_guide/intro_patterns.html

使用清单 — Ansible 文档
https://docs.ansible.com/ansible/latest/user_guide/intro_inventory.html

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

指导练习

利用主机模式选择主机

在本练习中，您将探索如何使用主机模式为 play 或临时命令指定清单中的主机。我们将为您提供几个示例清单来探索主机模式。

成果

您将能够使用不同的主机模式来访问清单中的不同主机。

在你开始之前

以 `student` 用户身份并使用 `student` 作为密码登录 `workstation`。

在 `workstation` 上，运行 **`lab projects-host start`** 命令。该脚本将创建 **`projects-host`** 项目目录，并下载本练习所需的 Ansible 配置文件和主机清单文件。

```
[student@workstation ~]$ lab projects-host start
```

1. 在 `workstation` 上，更改到本练习的工作目录 **`/home/student/projects-host`**，并检查该目录的内容。

```
[student@workstation ~]$ cd ~/projects-host
[student@workstation projects-host]$
```

1.1. 列出该目录的内容。

```
[student@workstation projects-host]$ ls
ansible.cfg inventory1 inventory2 playbook.yml
```

1.2. 检查示例清单文件 **`inventory1`**。注意清单的组织方式。探索清单中有哪些主机和组，以及使用了哪些域。

```
srv1.example.com
srv2.example.com
s1.lab.example.com
s2.lab.example.com

[web]
jupiter.lab.example.com
saturn.example.com

[db]
db1.example.com
db2.example.com
db3.example.com
```

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

```
[lb]
lb1.lab.example.com
lb2.lab.example.com

[boston]
db1.example.com
jupiter.lab.example.com
lb2.lab.example.com

[london]
db2.example.com
db3.example.com
file1.lab.example.com
lb1.lab.example.com

[dev]
web1.lab.example.com
db3.example.com

[stage]
file2.example.com
db2.example.com

[prod]
lb2.lab.example.com
db1.example.com
jupiter.lab.example.com

[function:children]
web
db
lb
city

[city:children]
boston
london
environments

[environments:children]
dev
stage
prod
new

[new]
172.25.252.23
172.25.252.44
172.25.252.32
```

1.3. 检查示例清单文件 **inventory2**。注意清单的组织方式。探索清单中有哪些主机和组，以及使用了哪些域。

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

```
workstation.lab.example.com

[london]
servera.lab.example.com

[berlin]
serverb.lab.example.com

[tokyo]
serverc.lab.example.com

[atlanta]
serverd.lab.example.com

[europe:children]
london
berlin
```

1.4. 最后，检查 playbook **playbook.yml** 的内容。注意该 playbook 如何使用 debug 模块显示每个受管主机的名称。

```
---
- name: Resolve host patterns
  hosts:
  tasks:
    - name: Display managed host name
      debug:
        msg: "{{ inventory_hostname }}"
```

2. 使用临时命令，确定 db1.example.com 服务器是否在 **inventory1** 清单文件中。

```
[student@workstation projects-host]$ ansible db1.example.com -i inventory1 \
> --list-hosts
  hosts (1):
    db1.example.com
```

3. 使用临时命令，通过主机模式来引用 **inventory1** 清单中包含的 IP 地址。

```
[student@workstation projects-host]$ ansible 172.25.252.44 -i inventory1 \
> --list-hosts
 hosts (1):
    172.25.252.44
```

4. 通过临时命令，使用 all 组列出 **inventory1** 清单文件中的所有受管主机。

```
[student@workstation projects-host]$ ansible all -i inventory1 --list-hosts
  hosts (17):
    srv1.example.com
    srv2.example.com
    s1.lab.example.com
```

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

```
    s2.lab.example.com
    jupiter.lab.example.com
    saturn.example.com
    db1.example.com
    db2.example.com
    db3.example.com
    lb1.lab.example.com
    lb2.lab.example.com
    file1.lab.example.com
    web1.lab.example.com
    file2.example.com
    172.25.252.23
    172.25.252.44
    172.25.252.32
```

5. 通过临时命令，使用星号 (*) 字符列出 **inventory1** 清单文件中以 **.example.com** 结尾的所有主机。

```
[student@workstation projects-host]$ ansible '*.example.com' -i inventory1 \
> --list-hosts
  hosts (14):
    jupiter.lab.example.com
    saturn.example.com
    db1.example.com
    db2.example.com
    db3.example.com
    lb1.lab.example.com
    lb2.lab.example.com
    file1.lab.example.com
    web1.lab.example.com
    file2.example.com
    srv1.example.com
    srv2.example.com
    s1.lab.example.com
    s2.lab.example.com
```

6. 如前一命令的输出中所示，***.example.com** 域中有 14 个主机。修改上一个临时命令中的主机模式，以便忽略 ***.lab.example.com** 域中的主机。

```
[student@workstation projects-host]$ ansible '*.example.com, !*.lab.example.com' \
> -i inventory1 --list-hosts
  hosts (7):
    saturn.example.com
    db1.example.com
    db2.example.com
    db3.example.com
    file2.example.com
    srv1.example.com
    srv2.example.com
```

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

▶ **7.** 在不访问 **inventory1** 清单文件中的组的情况下，使用临时命令列出以下三个主机：lb1.lab.example.com、s1.lab.example.com 和 db1.example.com。

```
[student@workstation projects-host]$ ansible \
> lb1.lab.example.com,s1.lab.example.com,db1.example.com -i inventory1 \
> --list-hosts
  hosts (3):
    lb1.lab.example.com
    s1.lab.example.com
    db1.example.com
```

▶ **8.** 在临时命令中使用通配符主机模式，列出以下清单文件中 IP 地址以 **172.25.** 开头的主机：**inventory1**。

```
[student@workstation projects-host]$ ansible '172.25.*' -i inventory1 --list-hosts
  hosts (3):
    172.25.252.23
    172.25.252.44
    172.25.252.32
```

▶ **9.** 在临时命令中使用主机模式，列出 **inventory1** 清单文件中以字母“s”开头的所有主机。

```
[student@workstation projects-host]$ ansible 's*' -i inventory1 --list-hosts
  hosts (7):
    saturn.example.com
    srv1.example.com
    srv2.example.com
    s1.lab.example.com
    s2.lab.example.com
    file2.example.com
    db2.example.com
```

注意 file2.example.com 和 db2.example.com 主机在前一命令的输出中。它们都属于名为 stage 的组的成员，此组的名称也以字母“s”开头，所以它们会出现在列表中。

▶ **10.** 在临时命令中使用列表和通配符主机模式，列出 **inventory1** 清单中的 prod 组中的所有主机、IP 地址以 **172** 开头的主机，以及名称中含有 **lab** 的主机。

```
[student@workstation projects-host]$ ansible 'prod,172*,*lab*' -i inventory1 \
> --list-hosts
  hosts (11):
    lb2.lab.example.com
    db1.example.com
    jupiter.lab.example.com
    172.25.252.23
    172.25.252.44
    172.25.252.32
    lb1.lab.example.com
```

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

```
    file1.lab.example.com
    web1.lab.example.com
    s1.lab.example.com
    s2.lab.example.com
```

11. 使用临时命令，列出同时属于 db 和 london 组的所有主机。

```
[student@workstation projects-host]$ ansible 'db,&london' -i inventory1 \
> --list-hosts
  hosts (2):
    db2.example.com
    db3.example.com
```

12. 修改 **playbook.yml** 文件中的 **hosts** 值，以便将 london 组中的所有服务器都作为目标。使用 **inventory2** 清单文件执行此 playbook。

```
...output omitted...
  hosts: london
...output omitted...
```

```
[student@workstation projects-host]$ ansible-playbook -i inventory2 playbook.yml
...output omitted...
TASK [Gathering Facts] ************************************************
ok: [servera.lab.example.com]
...output omitted...
```

13. 修改 **playbook.yml** 文件中的 **hosts** 值，以便将 europe 嵌套组中的所有服务器都作为目标。使用 **inventory2** 清单文件执行此 playbook。

```
...output omitted...
  hosts: europe
...output omitted...
```

```
[student@workstation projects-host]$ ansible-playbook -i inventory2 playbook.yml
...output omitted...
TASK [Gathering Facts] ************************************************
ok: [servera.lab.example.com]
ok: [serverb.lab.example.com]
...output omitted...
```

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

▶ **14.** 修改 **playbook.yml** 文件中的 **hosts** 值，以便将不属于任何组的所有服务器都作为目标。使用 **inventory2** 清单文件执行此 playbook。

```
...output omitted...
  hosts: ungrouped
...output omitted...
```

```
[student@workstation projects-hosts]$ ansible-playbook -i inventory2 playbook.yml
...output omitted...
TASK [Gathering Facts] ***********************************************
ok: [workstation.lab.example.com]
...output omitted...
```

完成

在 workstation 上，运行 **lab projects-host finish** 脚本来清理本练习。

```
[student@workstation ~]$ lab projects-host finish
```

本引导式练习到此结束。

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

管理动态清单

培训目标

学完本节后，您将能够描述什么是动态清单，以及安装和使用现有脚本作为 Ansible 动态清单来源。

动态生成清单

您目前为止使用的静态清单文件很容易编写，对于管理小型基础架构而言也很方便。不过，如果要操作许多台计算机，或者在计算机更替非常快的环境中工作，可能难以让静态清单文件保持最新状态。

大多数大型 IT 环境中设有系统来跟踪可用的主机以及它们的组织方式。例如，可能有外部目录服务通过 Zabbix 等监控系统维护，或者位于 FreeIPA 或 Active Directory 服务器上。Cobbler 等安装服务器或红帽卫星等管理服务可能跟踪部署的裸机系统。类似地，Amazon Web Services EC2 或 OpenStack 部署等云服务，或者基于 VMware 或红帽虚拟化的虚拟机基础架构可能是有关那些更替的实例和虚拟机的信息来源。

Ansible 支持动态清单脚本，这些脚本在每当 Ansible 执行时从这些类型的来源检索当前的信息，使清单能够实时得到更新。这些脚本是可以执行的程序，能够从一些外部来源收集信息，并以 JSON 格式输出清单。

动态清单脚本的使用方式与静态清单文本文件一样。清单的位置可以直接在当前的 **`ansible.cfg`** 文件中指定，或者通过 **`-i`** 选项指定。如果清单文件可以执行，则它将被视为动态清单程序，Ansible 会尝试运行它来生成清单。如果文件不可执行，则它将被视为静态清单。

> **注意**
> 清单位置可以在 **`ansible.cfg`** 配置文件中通过 **`inventory`** 参数进行配置。默认情况下，它被配置为 **`/etc/ansible/hosts`**。

贡献的脚本

开源社区向 Ansible 项目贡献了大量现有的动态清单脚本。它们没有包含在 ansible 软件包中，也没有红帽的官方支持。这些脚本可从 Ansible GitHub 网站 (`https://github.com/ansible/ansible/tree/devel/contrib/inventory`) 获取。

贡献的动态清单脚本以一些数据源或平台作为对象，包括：

- 私有云平台，如红帽 OpenStack 平台。
- 公共云平台，如 Rackspace Cloud、Amazon Web Services EC2 和 Google Compute Engine。
- 虚拟化平台，如红帽虚拟化 (oVirt) 和 VMware vSphere。
- 平台即服务解决方案，如 OpenShift Container Platform。
- 生命周期管理工具，如 Foreman（搭配红帽卫星 6 或独立使用）和 Spacewalk（红帽卫星 5 的上游）。
- 主机托管提供程序，如 Digital Ocean 和 Linode。

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

要能发挥作用，每一脚本可能有自己的依赖项和要求。贡献的脚本通常以 Python 编写，但这不是动态清单脚本的要求。

编写动态清单程序

如果使用的目录系统或基础架构没有动态清单脚本，您可以编写自定义动态清单程序。可以使用任何编程语言编写自定义程序，但传递适当的选项时必须以 JSON 格式返回清单信息。

ansible-inventory 命令是学习如何以 JSON 格式编写 Ansible 清单的有用工具。您可以使用 Ansible 2.4 开始提供的 **ansible-inventory** 命令，以 JSON 格式查看清单文件。

要以 JSON 格式显示清单文件的内容，请运行 **ansible-inventory --list** 命令。您可以使用 **-i** 选项指定要处理的清单文件的位置，或仅使用当前 Ansible 配置设置的默认清单。

以下示例演示了如何使用 **ansible-inventory** 命令来处理 INI 样式的清单文件并以 JSON 格式输出。

```
[student@workstation projects-host]$ cat inventory
workstation1.lab.example.com

[webservers]
web1.lab.example.com
web2.lab.example.com

[databases]
db1.lab.example.com
db2.lab.example.com

[student@workstation projects-host]$ ansible-inventory -i inventory --list
{
    "_meta": {
        "hostvars": {
            "db1.lab.example.com": {},
            "db2.lab.example.com": {},
            "web1.lab.example.com": {},
            "web2.lab.example.com": {},
            "workstation1.lab.example.com": {}
        }
    },
    "all": {
        "children": [
            "databases",
            "ungrouped",
            "webservers"
        ]
    },
    "databases": {
        "hosts": [
            "db1.lab.example.com",
            "db2.lab.example.com"
        ]
    },
    "ungrouped": {
        "hosts": [
            "workstation1.lab.example.com"
```

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