

注意

在红帽企业 Linux 7 上，**/etc/sudoers** 的默认配置允许 **wheel** 组中的所有用户在输入密码后使用 **sudo** 成为 **root** 用户。

若要让用户（以下示例中为 **someuser**）在不输入密码的前提下使用 **sudo** 成为 **root**，一种方式是将含有适当指令的文件安装到 **/etc/sudoers.d** 目录（归 **root** 所有，八进制权限为 0400）：

```
## password-less sudo for Ansible user
someuser ALL=(ALL) NOPASSWD:ALL
```

仔细思索您选择的特权升级方法的安全影响。不同的组织和部署可能要考虑不同的权衡。

下方的示例 **ansible.cfg** 文件假定您可以通过基于 SSH 密钥的身份验证以 **someuser** 用户身份连接受管主机，并且 **someuser** 可以使用 **sudo** 以 **root** 用户身份运行命令而不必输入密码：

```
[defaults]
inventory = ./inventory
remote_user = someuser
ask_pass = false

[privilegeEscalation]
become = true
become_method = sudo
become_user = root
become_ask_pass = false
```

非 SSH 连接

默认情况下，Ansible 用于连接受管主机的协议设置为 **smart**，它会确定使用 SSH 的最高效方式。可以通过多种方式将其设置为其他的值。

例如，默认使用 SSH 的规则有一个例外。如果您的目录中没有 **localhost**，Ansible 将设置一个隐式 **localhost** 条目以允许您运行以 **localhost** 为目标的临时命令和 playbook。这一特殊清单条目不包括在 **all** 或 **ungrouped** 主机组中。此外，Ansible 不使用 **smart** SSH 连接类型，而是利用默认的特殊 **local** 连接类型来进行连接。

```
[user@controlnode ~]$ ansible localhost --list-hosts
[WARNING]: provided hosts list is empty, only localhost is available

hosts (1):
localhost
```

local 连接类型忽略 **remote_user** 设置，并且直接在本地系统上运行命令。如果使用了特权升级，它会在运行 **sudo** 时使用运行 Ansible 命令的用户帐户，而不是 **remote_user**。如果这两个用户具有不同的 **sudo** 特权，这可能会导致混淆。

如果您要确保像其他受管主机一样使用 SSH 连接 **localhost**，一种方法是在清单中列出它，但是，这会将它包含在 **all** 和 **ungrouped** 组中，而您可能不希望如此。

另一种办法是更改用于连接 `localhost` 的协议。执行此操作的最好方法是为 `localhost` 设置 `ansible_connection` 主机变量。为此，您要在运行 Ansible 命令的目录中创建 `host_vars` 子目录。在该子目录中，创建名为 `localhost` 的文件，其应含有 `ansible_connection: smart` 这一行。这将确保对 `localhost` 使用 `smart` (SSH) 连接协议，而非 `local`。

您也可以通过另一种变通办法来使用它。如果您的清单中列有 `127.0.0.1`，则默认情况下，您将使用 `smart` 来连接它。您也可以创建一个含有 `ansible_connection: local` 这一行的 `host_vars/127.0.0.1` 文件，它会改为使用 `local`。

本课程的后续部分中将更详细地阐述主机变量。



注意

您也可以使用组变量来更改整个主机组的连接类型。若要这么做，可以将名称与组名相同的文件放入 `group_vars` 目录，并确保这些文件含有连接变量的设置。

例如，您可能希望所有 Microsoft Windows 受管主机使用 `winrm` 协议和端口 5986 进行连接。若要这样配置，可以将所有这些受管主机放入 `windows` 组，然后创建一个名为 `group_vars/windows` 的文件并使它包含下列几行：

```
ansible_connection: winrm  
ansible_port: 5986
```

配置文件注释

Ansible 配置文件允许使用两种注释字符：井号或编号符号 (#) 以及分号 (;)。

位于行开头的编号符号会注释掉整行。它不能和指令位于同一行中。

分号字符可以注释掉所在行中其右侧的所有内容。它可以和指令位于同一行中，只要该指令在其左侧。



参考文献

`ansible(1)`、`ansible-config(1)`、`ssh-keygen(1)` 和 `ssh-copy-id(1)` 帮助手册

配置文件：Ansible 文档

https://docs.ansible.com/ansible/latest/installation_guide/intro_configuration.html

► 指导练习

管理 ANSIBLE 配置文件

在本练习中，您将通过编辑 Ansible 配置文件自定义您的 Ansible 环境。

成果

您应当能够创建配置文件，从而利用持久自定义设置来配置您的 Ansible 环境。

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 deploy-manage start** 命令。此脚本将确保受管主机 servera 可在网络上访问。

```
[student@workstation ~]$ lab deploy-manage start
```

- 1. 创建目录 **/home/student/deploy-manage**，用于包含本练习的文件。更改到这个新建的目录。

```
[student@workstation ~]$ mkdir ~/deploy-manage  
[student@workstation ~]$ cd ~/deploy-manage
```

- 2. 在 **/home/student/deploy-manage** 目录中，使用文本编辑器开始编辑新文件 **ansible.cfg**。

在该文件中创建 **[defaults]** 部分。在这一部分中，添加一行，以使用 **inventory** 指令将 **./inventory** 文件指定为默认清单。

```
[defaults]  
inventory = ./inventory
```

保存您的作业，再退出文本编辑器。

- 3. 在 **/home/student/deploy-manage** 目录中，使用文本编辑器开始编辑新的静态清单文件 **inventory**。

静态清单中应含有四个主机组：

- **[myself]** 应包含主机 localhost。
- **[intranetweb]** 应包含主机 servera.lab.example.com。
- **[internetweb]** 应包含主机 serverb.lab.example.com。
- **[web]** 应包含 intranetweb 和 internetweb 主机组。

- 3.1. 在 **/home/student/deploy-manage/inventory** 中，通过添加下列几行创建 myself 主机组：

```
[myself]
localhost
```

3.2. 在 **/home/student/deploy-manage/inventory** 中，通过添加下列几行创建 **intranetweb** 主机组：

```
[intranetweb]
servera.lab.example.com
```

3.3. 在 **/home/student/deploy-manage/inventory** 中，通过添加下列几行创建 **internetweb** 主机组：

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

3.4. 在 **/home/student/deploy-manage/inventory** 中，通过添加下列几行创建 **web** 组：

```
[web:children]
intranetweb
internetweb
```

3.5. 确认您的最终 **inventory** 文件是否如下方所示：

```
[myself]
localhost

[intranetweb]
servera.lab.example.com

[internetweb]
serverb.lab.example.com

[web:children]
intranetweb
internetweb
```

保存您的作业，再退出文本编辑器。

► 4. 使用 **ansible** 命令及 **--list-hosts** 选项，测试清单文件的主机组配置。这不会实际连接这些主机。

```
[student@workstation deploy-manage]$ ansible myself --list-hosts
hosts (1):
localhost
[student@workstation deploy-manage]$ ansible intranetweb --list-hosts
hosts (1):
servera.lab.example.com
[student@workstation deploy-manage]$ ansible internetweb --list-hosts
hosts (1):
```

```
serverb.lab.example.com
[student@workstation deploy-manage]$ ansible web --list-hosts
hosts (2):
    servera.lab.example.com
    serverb.lab.example.com
[student@workstation deploy-manage]$ ansible all --list-hosts
hosts (3):
    localhost
    servera.lab.example.com
    serverb.lab.example.com
```

- 5. 在文本编辑器中打开 `/home/student/deploy-manage/ansible.cfg` 文件。添加 `[privilege_escalation]` 部分，将 Ansible 配置为在受管主机上运行任务时自动使用 `sudo` 命令从 `student` 切换到 `root`。Ansible 还应配置为提示您输入 `student` 用户用于 `sudo` 命令的密码。

5.1. 通过添加下列条目，在 `/home/student/deploy-manage/ansible.cfg` 配置文件中创建 `[privilege_escalation]` 部分：

```
[privilege_escalation]
```

5.2. 通过将 `become` 指令设置为 `true` 来启用特权升级。

```
become = true
```

5.3. 通过将 `become_method` 指令设置为 `sudo`，将特权升级设置为使用 `sudo` 命令。

```
become_method = sudo
```

5.4. 通过将 `become_user` 指令设置为 `root` 来设置特权升级用户。

```
become_user = root
```

5.5. 通过将 `become_ask_pass` 指令设置为 `true` 来启用提示输入特权升级密码。

```
become_ask_pass = true
```

5.6. 确认完整的 `ansible.cfg` 文件是否如下方所示：

```
[defaults]
inventory = ./inventory

[privilege_escalation]
become = true
become_method = sudo
become_user = root
become_ask_pass = true
```

保存您的作业，再退出文本编辑器。

- 6. 再次运行 **ansible --list-hosts** 命令，以验证现在是否提示您输入 **sudo** 密码。
提示输入 **sudo** 密码时，请输入 **student**，即使它不用于本次空运行。

```
[student@workstation deploy-manage]$ ansible intranetweb --list-hosts  
BECOME password: student  
hosts (1):  
    servera.lab.example.com
```

完成

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

```
[student@workstation ~]$ lab deploy-manage finish
```

本引导式练习到此结束。

运行临时命令

培训目标

学完本节后，您应能够使用临时命令运行单个 Ansible 自动化任务，并解释临时命令的一些用例。

使用 ANSIBLE 运行临时命令

使用临时命令可以快速执行单个 Ansible 任务，您不需要将它保存下来供以后再次运行。它们是简单的在线操作，无需编写 playbook 即可运行。

临时命令对快速测试和更改很有用。例如，您可以使用临时命令确保一组服务器上的 **/etc/hosts** 文件中存在某一特定的行。您可以使用另一个临时命令在许多不同的计算机上高效重启一项服务，或者确保特定的软件包为最新版本。

临时命令对于通过 Ansible 快速执行简单的任务非常有用。它们确实也存在局限，而且总体而言，您要使用 Ansible Playbook 来充分发挥 Ansible 的作用。但在许多情形中，临时命令正是快速执行简单任务时所需要的工具。

运行临时命令

使用 **ansible** 命令来运行临时命令：

```
ansible host-pattern -m module [-a 'module arguments'] [-i inventory]
```

host-pattern 参数用于指定应在其中运行临时命令的受管主机。它可以是清单中的特定受管主机或主机组。您已看到过它与 **--list-hosts** 选项结合使用，此选项显示通过特定主机模式匹配的主机。您也了解过可以使用 **-i** 选项来指定要使用的其他清单位置，取代当前 Ansible 配置文件中的默认位置。

-m 选项将 Ansible 应在目标主机上运行的 **module** 的名称作为参数。模块是为了实施您的任务而执行的小程序。一些模块不需要额外的信息，但其他模块需要使用额外参数来指定其操作详情。**-a** 选项以带引号字符串形式取这些参数的列表。

一种最简单的临时命令使用 **ping** 模块。此模块不执行 ICMP ping，而是检查能否在受管主机上运行基于 Python 的模块。例如，以下临时命令确定清单中的所有受管主机能否运行标准的模块：

```
[user@controlnode ~]$ ansible all -m ping
servera.lab.example.com | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}
```

使用临时命令通过模块来执行任务

模块是临时命令用于完成任务的工具。Ansible 提供了数百个能够完成不同任务的模块。通常，您可以查找一个经过测试的专用模块，作为标准安装的一部分来完成您所需的任务。

ansible-doc -l 命令可列出系统上安装的所有模块。您可以使用 **ansible-doc** 来按名称查看特定模块的文档，再查找关于模块将取什么参数作为选项的信息。例如，以下命令显示 **ping** 模块的文档：

```
[user@controlnode ~]$ ansible-doc ping
> PING      (/usr/lib/python3.6/site-packages/ansible/modules/system/ping.py)

    A trivial test module, this module always returns `pong` on successful
    contact. It does not make sense in playbooks, but it is useful from `/usr/bin/
    ansible` to

        verify the ability to login and that a usable Python is configured. This
        is NOT ICMP ping, this is just a trivial test module that requires Python on the
        remote-node. For Windows targets, use the [win_ping] module instead. For
        Network targets, use the [net_ping] module instead.

    * This module is maintained by The Ansible Core Team
    OPTIONS (= is mandatory):

        - data
            Data to return for the `ping` return value.
            If this parameter is set to `crash`, the module will cause an exception.
            [Default: pong]
            type: str

SEE ALSO:
    * Module net_ping
        The official documentation on the net_ping module.
        https://docs.ansible.com/ansible/latest/modules/net_ping_module.html
    * Module win_ping
        The official documentation on the win_ping module.
        https://docs.ansible.com/ansible/latest/modules/win_ping_module.html

AUTHOR: Ansible Core Team, Michael DeHaan
METADATA:
    status:
        - stableinterface
    supported_by: core

EXAMPLES:
# Test we can logon to 'webservers' and execute python with json lib.
# ansible webservers -m ping

# Example from an Ansible Playbook
- ping:

# Induce an exception to see what happens
```

```
- ping:
  data: crash
```

RETURN VALUES:

```
ping:
  description: value provided with the data parameter
  returned: success
  type: str
  sample: pong
```

要了解有关模块的更多信息，请访问在线 Ansible 文档，网址：http://docs.ansible.com/ansible/latest/modules/modules_by_category.html。

下表列举了许多有用的模块。还存在很多其他模块。

Ansible 模块

模块类别	模块
文件模块	<ul style="list-style-type: none"> · copy: 将本地文件复制到受管主机 · file: 设置文件的权限和其他属性 · lineinfile: 确保特定行是否在文件中 · synchronize: 使用 rsync 同步内容
软件包模块	<ul style="list-style-type: none"> · package: 使用操作系统本机的自动检测软件包管理器管理软件包 · yum: 使用 YUM 软件包管理器管理软件包 · apt: 使用 APT 软件包管理器管理软件包 · dnf: 使用 DNF 软件包管理器管理软件包 · gem: 管理 Ruby gem · pip: 从 PyPI 管理 Python 软件包
系统模块	<ul style="list-style-type: none"> · firewalld: 使用 firewalld 管理任意端口和服务 · reboot: 重新启动计算机 · service: 管理服务 · user: 添加、删除和管理用户帐户
Net Tools 模块	<ul style="list-style-type: none"> · get_url: 通过 HTTP、HTTPS 或 FTP 下载文件 · nmcli: 管理网络 · uri: 与 Web 服务交互

大部分模块会取用参数。可在模块的文档中找到可用于该模块的参数列表。临时命令可以通过 **-a** 选项向模块传递参数。无需参数时，可以从临时命令中省略 **-a** 选项。如果需要指定多个参数，请以引号括起的空格分隔列表形式提供。

例如，以下临时命令使用 **user** 模块来确保 **newbie** 用户存在于 **servera.lab.example.com** 上，并且其 UID 为 4000：

```
[user@controlnode ~]$ ansible -m user -a 'name=newbie uid=4000 state=present' \
> servera.lab.example.com
servera.lab.example.com | SUCCESS => {
  "ansible_facts": {
```

```

    "discovered_interpreter_python": "/usr/libexec/platform-python"
},
"changed": true,
"comment": "",
"createhome": true,
"group": 4000,
"home": "/home/newbie",
"name": "newbie",
"shell": "/bin/bash",
"state": "present",
"system": false,
"uid": 4000
}

```

大多数模块为 idempotent，这表示它们可以安全地多次运行；如果系统已处于正确的状态，它们不会进行任何操作。例如，如果您再次运行前面的临时命令，它应该不会报告任何更改：

```

[user@controlnode ~]$ ansible -m user -a 'name=newbie uid=4000 state=present' \
> servera.lab.example.com
servera.lab.example.com | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
},
"append": false,
"changed": false
"comment": "",
"group": 4000,
"home": "/home/newbie",
"move_home": false,
"name": "newbie",
"shell": "/bin/bash",
"state": "present",
"uid": 4000
}

```

在受管主机上运行任意命令

`command` 模块允许管理员在受管主机的命令行中运行任意命令。要运行的命令通过 `-a` 选项指定为该模块的参数。例如，以下命令将对由 `mymanagedhosts` 主机模式引用的受管主机运行 `hostname` 命令。

```

[user@controlnode ~]$ ansible mymanagedhosts -m command -a /usr/bin/hostname
host1.lab.example.com | CHANGED | rc=0 >>
host1.lab.example.com
host2.lab.example.com | CHANGED | rc=0 >>
host2.lab.example.com

```

上一临时命令为每个受管主机返回两行输出。第一行是状态报告，显示对其运行该临时操作的受管主机名称及操作的结果。第二行是使用 Ansible `command` 模块远程执行的命令的输出。

若要改善临时命令输出的可读性和解析，管理员可能会发现使对受管主机执行的每一项操作具有单行输出十分有用。使用 `-o` 选项以单行格式显示 Ansible 临时命令的输出。

```
[user@controlnode ~]$ ansible mymanagedhosts -m command -a /usr/bin/hostname -o  
host1.lab.example.com | CHANGED | rc=0 >> (stdout) host1.lab.example.com  
host2.lab.example.com | CHANGED | rc=0 >> (stdout) host2.lab.example.com
```

command 模块允许管理员对受管主机快速执行远程命令。这些命令不是由受管主机上的 shell 加以处理。因此，它们无法访问 shell 环境变量，也不能执行重定向和传送等 shell 操作。



注意

如果临时命令没有指定哪个模块与 **-m** 选项一起使用，红帽 Ansible 引擎将默认使用 **命令** 模块。

在命令需要 shell 处理的情形中，管理员可以使用 **shell** 模块。与 **command** 模块类似，您可以在临时命令中将要执行的命令作为参数传递给该模块。Ansible 随后对受管主机远程执行该命令。与 **command** 模块不同的是，这些命令将通过受管主机上的 shell 进行处理。因此，可以访问 shell 环境变量，也可使用重定向和传送等 shell 操作。

以下示例演示了 **command** 模块和 **shell** 模块的区别。如果您尝试使用这两个模块执行内建的 Bash 命令 **set**，只有使用 **shell** 模块时才会成功。

```
[user@controlnode ~]$ ansible localhost -m command -a set  
localhost | FAILED | rc=2 >>  
[Errno 2] No such file or directory  
[user@controlnode ~]$ ansible localhost -m shell -a set  
localhost | CHANGED | rc=0 >>  
BASH=/bin/sh  
BASHOPTS=cmdhist:extquote:force_fignore:hostcomplete:interact  
ive_comments:progcomp:promptvars:sourcepath  
BASH_ALIASES=()  
...output omitted...
```

command 和 **shell** 模块都要求受管主机上安装正常工作的 Python。第三个模块是 **raw**，它可以绕过模块子系统，直接使用远程 shell 运行命令。在管理无法安装 Python 的系统（如网络路由器）时，可以利用这个模块。它也可用于将 Python 安装到主机上。



重要

在大多数情形中，建议您避免使用 **command**、**shell** 和 **raw** 这三个“运行命令”模块。

其他模块大部分都是幂等的，可以自动进行更改跟踪。它们可以测试系统的状态，在这些系统已处于正确的状态时不执行任何操作。相反，以幂等方式使用“运行命令”模块要复杂得多。依靠它们，您更难以确信再次运行临时命令或 playbook 不会造成意外的失败。当 **shell** 或 **command** 模块运行时，通常会基于它是否认为影响了计算机状态而报告 **CHANGED** 状态。

有时候，“运行命令”模块是有用的工具，也是解决问题的好办法。如果确实需要使用它们，可能最好先尝试使用 **command** 模块，只有在需要 **shell** 或 **raw** 模块的特殊功能时才利用它们。

配置临时命令的连接

受管主机连接和特权升级的指令可以在 Ansible 配置文件中配置，也可使用临时命令中的选项来定义。使用临时命令中的选项定义时，它们将优先于 Ansible 配置文件中配置的指令。下表显示了与各项配置文件指令类同的命令行选项。

Ansible 命令行选项

配置文件指令	命令行选项
inventory	-i
remote_user	-u
become	--become、 -b
become_method	--become-method
become_user	--become-user
become_ask_pass	--ask-become-pass、 -K

在使用命令行选项配置这些指令前，可以通过查询 **ansible --help** 的输出来确定其当前定义的值。

```
[user@controlnode ~]$ ansible --help
...output omitted...
-b, --become           run operations with become (nopasswd implied)
--become-method=BECOME_METHOD
                      privilege escalation method to use (default=sudo),
                      valid choices: [ sudo | su | pbrun | pfexec | runas |
                      doas ]
--become-user=BECOME_USER
...output omitted...
-u REMOTE_USER, --user=REMOTE_USER
                      connect as this user (default=None)
```



参考文献

ansible(1) 帮助手册

使用模式：Ansible 文档

https://docs.ansible.com/ansible/latest/user_guide/intro_patterns.html

临时命令简介：Ansible 文档

http://docs.ansible.com/ansible/latest/user_guide/intro_adhoc.html

模块索引：Ansible 文档

http://docs.ansible.com/ansible/latest/modules/modules_by_category

command - 对远程节点执行命令：Ansible 文档

http://docs.ansible.com/ansible/latest/modules/command_module.html

shell - 在节点中执行命令：Ansible 文档

http://docs.ansible.com/ansible/latest/modules/shell_module.html

► 指导练习

运行临时命令

在本练习中，您将对多个受管主机执行临时命令。

成果

您应当能够利用特权升级以临时命令为基础对受管主机执行命令。

您将利用 `devops` 用户帐户对 `workstation` 和 `servera` 执行临时命令。此帐户在 `workstation` 和 `servera` 上具有相同的 `sudo` 配置。

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 deploy-adhoc start` 命令。此脚本将确保受管主机 `servera` 可在网络上访问。它也会创建 `/home/student/deploy-adhoc` 工作目录并填充本练习中使用的材料。

```
[student@workstation ~]$ lab deploy-adhoc start
```

► 1. 确定 `workstation` 和 `servera` 上 `devops` 帐户的 `sudo` 配置。

- 1.1. 确定 `devops` 帐户的 `sudo` 配置，该帐户已在 `workstation` 构建时配置。如果提示输入 `student` 帐户的密码，则输入 `student`。

```
[student@workstation ~]$ sudo cat /etc/sudoers.d/devops
[sudo] password for student: student
devops ALL=(ALL) NOPASSWD: ALL
```

请注意，该用户具有完整的 `sudo` 特权，但无需密码身份验证。

- 1.2. 确定 `devops` 帐户的 `sudo` 配置，该帐户已在 `servera` 构建时配置。

```
[student@workstation ~]$ ssh devops@servera.lab.example.com
[devops@servera ~]$ sudo cat /etc/sudoers.d/devops
devops ALL=(ALL) NOPASSWD: ALL
[devops@servera ~]$ exit
```

请注意，该用户具有完整的 `sudo` 特权，但无需密码身份验证。

► 2. 将目录更改为 `/home/student/deploy-adhoc`，再检查 `ansible.cfg` 和 `inventory` 文件的内容。

```
[student@workstation ~]$ cd ~/deploy-adhoc
[student@workstation deploy-adhoc]$ cat ansible.cfg
[defaults]
```

```
inventory=inventory
[student@workstation deploy-adhoc]$ cat inventory
[control_node]
localhost

[intranetweb]
servera.lab.example.com
```

配置文件将该目录中的 **inventory** 文件配置为 Ansible 清单。请注意，Ansible 尚未配置为使用特权升级。

- 3. 使用 **all** 主机组和 **ping** 模块，执行临时命令，确保所有受管主机都可以运行使用 Python 的 Ansible 模块。

```
[student@workstation deploy-adhoc]$ ansible all -m ping
servera.lab.example.com | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}
localhost | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}
```

- 4. 使用 **command** 模块，对 **workstation** 执行临时命令，以确定 Ansible 用来对受管主机执行操作的用户帐户。使用 **localhost** 主机模式来连接 **workstation**，以执行临时命令。由于您是在本地连接，所以 **workstation** 既是控制节点也是受管主机。

```
[student@workstation deploy-adhoc]$ ansible localhost -m command -a 'id'
localhost | CHANGED | rc=0 >>
uid=1000(student) gid=1000(student) groups=1000(student),10(wheel)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

注意，该临时命令是以 **student** 用户身份对受管主机执行的。

- 5. 对 **workstation** 执行上一临时命令，但通过使用 **-u** 选项以 **devops** 用户帐户进行连接并执行操作。

```
[student@workstation deploy-adhoc]$ ansible localhost -m command -a 'id' -u devops
localhost | CHANGED | rc=0 >>
uid=1001(devops) gid=1001(devops) groups=1001(devops)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

注意，该临时命令是以 **devops** 用户身份对受管主机执行的。

- 6. 使用 `copy` 模块，对 `workstation` 执行临时命令以更改 `/etc/motd` 文件的内容，使其包含字符串“Managed by Ansible”并且后跟一个换行符。使用 `devops` 帐户执行命令，但不要使用 `--become` 选项切换到 `root`。因为权限不足，该临时命令应该会失败。

```
[student@workstation deploy-adhoc]$ ansible localhost -m copy \
> -a 'content="Managed by Ansible\n" dest=/etc/motd' -u devops
localhost | FAILED! => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "checksum": "4458b979ede3c332f8f2128385df4ba305e58c27",
    "msg": "Destination /etc not writable"
}
```

临时命令失败的原因是，`devops` 用户没有写入文件的权限。

- 7. 使用特权升级再次运行该命令。您可以在 `ansible.cfg` 文件中修正设置，但在本例中，我们仅使用 `ansible` 命令的相应命令行选项。

使用 `copy` 模块，对 `workstation` 执行上一命令以更改 `/etc/motd` 文件的内容，使其包含字符串“Managed by Ansible”并且后跟一个换行符。使用 `devops` 用户连接受管主机，但通过 `--become` 选项以 `root` 用户身份执行该操作。使用 `--become` 选项就够了，因为 `become_user` 指令的默认值在 `/etc/ansible/ansible.cfg` 文件中设置为 `root`。

```
[student@workstation deploy-adhoc]$ ansible localhost -m copy \
> -a 'content="Managed by Ansible\n" dest=/etc/motd' -u devops --become
localhost | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": true,
    "checksum": "4458b979ede3c332f8f2128385df4ba305e58c27",
    "dest": "/etc/motd",
    "gid": 0,
    "group": "root",
    "md5sum": "65a4290ee5559756ad04e558b0e0c4e3",
    "mode": "0644",
    "owner": "root",
    "secontext": "system_u:object_r:etc_t:s0",
    "size": 19,
    "src": "/home/devops/.ansible/tmp/ansible-
tmp-1558954193.0260043-131348380629718/source",
    "state": "file",
    "uid": 0
}
```

注意这一次命令执行成功，因为该临时命令是使用特权升级执行的。

- 8. 使用 all 主机组在所有主机上再次运行之前的临时命令。这会确保 workstation 和 servera 上的 /etc/motd 都包含文本 “Managed by Ansible”。

```
[student@workstation deploy-adhoc]$ ansible all -m copy \
> -a 'content="Managed by Ansible\n" dest=/etc/motd' -u devops --become
servera.lab.example.com | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": true,
    "checksum": "4458b979ede3c332f8f2128385df4ba305e58c27",
    "dest": "/etc/motd",
    "gid": 0,
    "group": "root",
    "md5sum": "65a4290ee5559756ad04e558b0e0c4e3",
    "mode": "0644",
    "owner": "root",
    "secontext": "system_u:object_r:etc_t:s0",
    "size": 19,
    "src": "/home/devops/.ansible/tmp/ansible-
tmp-1558954250.7893758-136255396678462/source",
    "state": "file",
    "uid": 0
}
localhost | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "checksum": "4458b979ede3c332f8f2128385df4ba305e58c27",
    "dest": "/etc/motd",
    "gid": 0,
    "group": "root",
    "mode": "0644",
    "owner": "root",
    "path": "/etc/motd",
    "secontext": "system_u:object_r:etc_t:s0",
    "size": 19,
    "state": "file",
    "uid": 0
}
```

您应该会看到 localhost 显示 **SUCCESS**, servera 显示 **CHANGED**。不过, localhost 应该会报告 **"changed": false**, 因为该文件已处于正确的状态。相反, servera 应该会报告 **"changed": true**, 因为临时命令已将此文件更新为正确的状态。

- 9. 使用 `command` 模块，执行临时命令来运行 `cat /etc/motd`，以验证 `workstation` 和 `servera` 上文件的内容都已成功修改。使用 `all` 主机组和 `devops` 用户来指定并连接受管主机。不需要特权升级就能让此命令正常运行。

```
[student@workstation deploy-adhoc]$ ansible all -m command \
> -a 'cat /etc/motd' -u devops
servera.lab.example.com | CHANGED | rc=0 >>
Managed by Ansible

localhost | CHANGED | rc=0 >>
Managed by Ansible
```

完成

在 `workstation` 上，运行 `lab deploy-adhoc finish` 脚本来清理本练习。

```
[student@workstation ~]$ lab deploy-adhoc finish
```

本引导式练习到此结束。

▶ 开放研究实验

部署 ANSIBLE

任务执行清单

在本实验中，您将配置连接清单主机的 Ansible 控制节点，并使用临时命令对受管主机执行操作。

成果

您应能够配置控制节点，以在受管主机上运行临时命令。

您将以 `student` 用户身份使用 Ansible 来管理 `workstation.lab.example.com` 中的多个主机。您将设置一个项目目录，其中包含一个具有特定默认值的 `ansible.cfg` 文件，还将设置一个包含清单文件的 `inventory` 目录。

您将使用临时命令来确保所有受管主机上的 `/etc/motd` 文件都包含指定的内容。

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 deploy-review start` 命令。此脚本将确保受管主机可在网络上访问。

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

1. 验证控制节点上已安装了 ansible 软件包，再运行 `ansible --version` 命令。
2. 在 `workstation` 上的 `student` 用户主目录 `/home/student` 中，创建名为 `deploy-review` 的新目录。更改到该目录。
3. 在 `deploy-review` 目录上创建 `ansible.cfg` 文件，您将使用该文件来设置下列 Ansible 默认值：
 - 以 `devops` 用户身份连接受管主机。
 - 使用 `inventory` 子目录来存放清单文件。
 - 默认禁用特权升级。如果从命令行启用特权升级，请配置默认设置，使 Ansible 使用 `sudo` 方法来切换到 `root` 用户帐户。Ansible 应该不会提示输入 `devops` 登录密码或 `sudo` 密码。

已经为受管主机配置了 `devops` 用户，该用户可以使用基于 SSH 密钥的身份验证方式登录，并且可以 `root` 用户身份使用 `sudo` 命令运行任何命令，而无需提供密码。

4. 创建 `/home/student/deploy-review/inventory` 目录。
下载 `http://materials.example.com/labs/deploy-review/inventory` 文件，并将它保存为名为 `/home/student/deploy-review/inventory/inventory` 的静态清单文件。

5. 以 `all` 主机组为目标作为临时命令执行 `id` 命令，以验证 `devops` 是远程用户并且特权升级默认为禁用。
6. 以 `all` 主机组为目标执行一个临时命令，使用 `copy` 模块修改所有主机上的 `/etc/motd` 文件的内容。

使用 `copy` 模块的 `content` 选项，确保 `/etc/motd` 文件在单独一行中含有字符串 `This server is managed by Ansible.\n`。（`\n` 与 `content` 选项搭配使用，可使该模块在字符串末尾放入一个换行符。）

您必须从命令行请求特权升级，以确保这适用于您的当前 `ansible.cfg` 默认值。
7. 如果您再次运行同样的临时命令，您应该会看到 `copy` 模块检测到文件已经正确，因此不更改它们。查看该临时命令是否对每一受管主机报告 `SUCCESS` 并显示有 `"changed": false` 这一行。
8. 若要以另一种方式确认，请以 `all` 主机组为目标运行一个临时命令，以使用 `command` 模块来执行 `cat /etc/motd` 命令。`ansible` 的输出应当为所有主机都显示字符串 `This server is managed by Ansible.`。运行此临时命令不需要特权升级。
9. 在 `workstation` 上运行 `lab deploy-review grade` 以检查您的工作。

```
[student@workstation deploy-review]$ lab deploy-review grade
```

完成

在 `workstation` 上，运行 `lab deploy-review finish` 脚本来清理本实验中创建的资源。

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

本实验到此结束。

► 解决方案

部署 ANSIBLE

任务执行清单

在本实验中，您将配置连接清单主机的 Ansible 控制节点，并使用临时命令对受管主机执行操作。

成果

您应能够配置控制节点，以在受管主机上运行临时命令。

您将以 `student` 用户身份使用 Ansible 来管理 `workstation.lab.example.com` 中的多个主机。您将设置一个项目目录，其中包含一个具有特定默认值的 `ansible.cfg` 文件，还将设置一个包含清单文件的 `inventory` 目录。

您将使用临时命令来确保所有受管主机上的 `/etc/motd` 文件都包含指定的内容。

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 deploy-review start` 命令。此脚本将确保受管主机可在网络上访问。

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

- 验证控制节点上已安装了 ansible 软件包，再运行 `ansible --version` 命令。

1. 验证是否安装了 ansible 软件包。

```
[student@workstation ~]$ yum list installed ansible
Installed Packages
ansible.noarch      2.8.0-1.el8ae      @rhel-8-server-ansible-2.8-rpms
```

2. 运行 `ansible --version` 命令，以确认所安装的 Ansible 版本。

```
[student@workstation ~]$ ansible --version
ansible 2.8.0
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/home/student/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3.6/site-packages/ansible
  executable location = /usr/bin/ansible
  python version = 3.6.8 (default, Apr  3 2019, 17:26:03) [GCC 8.2.1 20180905 (Red Hat 8.2.1-3)]
```

3. 在 `workstation` 上的 `student` 用户主目录 `/home/student` 中，创建名为 `deploy-review` 的新目录。更改到该目录。

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

3. 在 **deploy-review** 目录上创建 **ansible.cfg** 文件，您将使用该文件来设置下列 Ansible 默认值：

- 以 **devops** 用户身份连接受管主机。
- 使用 **inventory** 子目录来存放清单文件。
- 默认禁用特权升级。如果从命令行启用特权升级，请配置默认设置，使 Ansible 使用 **sudo** 方法来切换到 **root** 用户帐户。Ansible 应该不会提示输入 **devops** 登录密码或 **sudo** 密码。

已经为受管主机配置了 **devops** 用户，该用户可以使用基于 SSH 密钥的身份验证方式登录，并且可以 **root** 用户身份使用 **sudo** 命令运行任何命令，而无需提供密码。

- 3.1. 使用文本编辑器创建 **/home/student/deploy-review/ansible.cfg** 文件。创建 **[defaults]** 部分。添加 **remote_user** 指令，使 Ansible 在连接受管主机时使用 **devops** 用户。添加 **inventory** 指令，将 Ansible 配置为使用 **/home/student/deploy-review/inventory** 目录作为清单文件的默认位置。

```
[defaults]
remote_user = devops
inventory = inventory
```

- 3.2. 在 **/home/student/deploy-review/ansible.cfg** 文件中，创建 **[privilege_escalation]** 部分，再添加下列条目来禁用特权升级。将特权升级方法设置为使用 **root** 帐户及 **sudo** 且无需密码身份验证。

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

- 3.3. 完成的 **ansible.cfg** 文件应当如下所示：

```
[defaults]
remote_user = devops
inventory = inventory

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

保存您的工作，再退出编辑器。

4. 创建 **/home/student/deploy-review/inventory** 目录。

下载 `http://materials.example.com/labs/deploy-review/inventory` 文件，并将它保存为名为 `/home/student/deploy-review/inventory/inventory` 的静态清单文件。

4.1. 创建 `/home/student/deploy-review/inventory` 目录。

```
[student@workstation deploy-review]$ mkdir inventory
```

4.2. 将 `http://materials.example.com/labs/deploy-review/inventory` 文件下载到 `/home/student/deploy-review/inventory` 目录。

```
[student@workstation deploy-review]$ wget -O inventory/inventory \
> http://materials.example.com/labs/deploy-review/inventory
```

4.3. 检查 `/home/student/deploy-review/inventory/inventory` 文件的内容。

```
[student@workstation deploy-review]$ cat inventory/inventory
[internetweb]
serverb.lab.example.com
```

```
[intranetweb]
servera.lab.example.com
serverc.lab.example.com
serverd.lab.example.com
```

5. 以 all 主机组为目标作为临时命令执行 `id` 命令，以验证 `devops` 是远程用户并且特权升级默认认为禁用。

```
[student@workstation deploy-review]$ ansible all -m command -a 'id'
serverb.lab.example.com | CHANGED | rc=0 >>
uid=1001(devops) gid=1001(devops) groups=1001(devops) context=unconfined_u:
unconfined_r:unconfined_t:s0-s0:c0.c1023

serverc.lab.example.com | CHANGED | rc=0 >>
uid=1001(devops) gid=1001(devops) groups=1001(devops) context=unconfined_u:
unconfined_r:unconfined_t:s0-s0:c0.c1023

servera.lab.example.com | CHANGED | rc=0 >>
uid=1001(devops) gid=1001(devops) groups=1001(devops) context=unconfined_u:
unconfined_r:unconfined_t:s0-s0:c0.c1023

serverd.lab.example.com | CHANGED | rc=0 >>
uid=1001(devops) gid=1001(devops) groups=1001(devops) context=unconfined_u:
unconfined_r:unconfined_t:s0-s0:c0.c1023
```

您的结果可能会以不同顺序返回。

6. 以 all 主机组为目标执行一个临时命令，使用 `copy` 模块修改所有主机上的 `/etc/motd` 文件的内容。

使用 `copy` 模块的 `content` 选项，确保 `/etc/motd` 文件在单独一行中含有字符串 `This server is managed by Ansible.\n`。（`\n` 与 `content` 选项搭配使用，可使该模块在字符串末尾放入一个换行符。）

您必须从命令行请求特权升级，以确保这适用于您的当前 `ansible.cfg` 默认值。

```
[student@workstation deploy-review]$ ansible all -m copy \
> -a 'content="This server is managed by Ansible.\n" dest=/etc/motd' --become
serverd.lab.example.com | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": true,
    "checksum": "93d304488245bb2769752b95e0180607effc69ad",
    "dest": "/etc/motd",
    "gid": 0,
    "group": "root",
    "md5sum": "af74293c7b2a783c4f87064374e9417a",
    "mode": "0644",
    "owner": "root",
    "secontext": "system_u:object_r:etc_t:s0",
    "size": 35,
    "src": "/home/devops/.ansible/tmp/ansible-
tmp-1558954517.7426903-24998924904061/source",
    "state": "file",
    "uid": 0
}
servera.lab.example.com | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": true,
    "checksum": "93d304488245bb2769752b95e0180607effc69ad",
    "dest": "/etc/motd",
    "gid": 0,
    "group": "root",
    "md5sum": "af74293c7b2a783c4f87064374e9417a",
    "mode": "0644",
    "owner": "root",
    "secontext": "system_u:object_r:etc_t:s0",
    "size": 35,
    "src": "/home/devops/.ansible/tmp/ansible-
tmp-1558954517.7165847-103324013882266/source",
    "state": "file",
    "uid": 0
}
serverc.lab.example.com | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": true,
    "checksum": "93d304488245bb2769752b95e0180607effc69ad",
    "dest": "/etc/motd",
    "gid": 0,
    "group": "root",
    "md5sum": "af74293c7b2a783c4f87064374e9417a",
    "mode": "0644",
    "owner": "root",
```

```

    "secontext": "system_u:object_r:etc_t:s0",
    "size": 35,
    "src": "/home/devops/.ansible/tmp/ansible-tmp-1558954517.75727-94151722302122/
source",
    "state": "file",
    "uid": 0
}
serverb.lab.example.com | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": true,
    "checksum": "93d304488245bb2769752b95e0180607effc69ad",
    "dest": "/etc/motd",
    "gid": 0,
    "group": "root",
    "md5sum": "af74293c7b2a783c4f87064374e9417a",
    "mode": "0644",
    "owner": "root",
    "secontext": "system_u:object_r:etc_t:s0",
    "size": 35,
    "src": "/home/devops/.ansible/tmp/ansible-
tmp-1558954517.6649802-53313238077104/source",
    "state": "file",
    "uid": 0
}

```

7. 如果您再次运行同样的临时命令，您应该会看到 `copy` 模块检测到文件已经正确，因此不更改它们。查看该临时命令是否对每一受管主机报告 `SUCCESS` 并显示有 `"changed": false` 这一行。

```

[student@workstation deploy-review]$ ansible all -m copy \
> -a 'content="This server is managed by Ansible.\n" dest=/etc/motd' --become
serverb.lab.example.com | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "checksum": "93d304488245bb2769752b95e0180607effc69ad",
    "dest": "/etc/motd",
    "gid": 0,
    "group": "root",
    "mode": "0644",
    "owner": "root",
    "path": "/etc/motd",
    "secontext": "system_u:object_r:etc_t:s0",
    "size": 35,
    "state": "file",
    "uid": 0
}
serverc.lab.example.com | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },

```