Red Hat Ansible Automation Platform 2.4

Ansible Playbook 入门

ansible playbook 入门

Last Updated: 2025-07-23

Red Hat Ansible Automation Platform 2.4 Ansible Playbook 入门

ansible playbook 入门

Legal Notice

Abstract

本指南介绍了如何创建和使用 playbook 满足您的自动化要求。 本文档包括上游 docs.ansible.com 文档中的内容，该文档由 GNU GENERAL PUBLIC LICENSE v3.0 涵盖。

Table of Contents

前言

感谢您对 Red Hat Ansible Automation Platform 的关注。Ansible Automation Platform 是一个商业产品，它可以帮助团队通过增加控制、知识、协调基于 Ansible 的环境来更好地管理多阶的复杂部署环境。

本指南介绍了使用 Ansible Playbook。

对红帽文档提供反馈

如果您对本文档有任何改进建议，或发现错误，请通过 https://access.redhat.com 联系技术支持来创建一个请求。

第 1 章 简介

Ansible Playbook 是自动化任务的蓝图，这是在解决方案清单期间执行有限的手动工作操作。Playbook 告知 Ansible 在哪个设备上执行什么操作。除了在 IT 环境中手动应用相同操作到数百个或数千个类似技术的操作，而是自动执行 playbook 会自动完成对指定类型的清单（如一组路由器）的相同操作。

Playbook 定期用于自动化 IT 基础架构 - 例如操作系统和 Kubernetes 平台网络、安全系统和代码存储库（如 GitHub）。您可以使用 playbook 来编程应用程序、服务、服务器节点和其他设备，而无需从头开始创建所有操作。Playbook 以及它们内的条件、变量和任务可以无限期保存、共享或重复使用。这样，您可以更轻松地协调操作知识，并确保相同的操作是一致的执行。

1.1. ANSIBLE PLAYBOOK 的工作原理？

Ansible Playbook 是自动为指定清单或主机组执行的任务列表。可以合并一个或多个 Ansible 任务来制作一个 play，即映射到特定主机的任务分组。

任务按照它们写入的顺序执行。

一个 playbook 可以包含一个或多个 play。

playbook 由排序列表中的一个或多个 **play** 组成。

术语 **playbook** 和 **play** 是 sports analogies。

每个 play 执行 playbook 的整体目标的一部分，运行一个或多个任务。

每个任务都调用 Ansible 模块。

Playbook

定义 Ansible 从上到下执行操作的 play 列表，以实现总体目标。

Play

映射到清单中的受管节点的任务列表。

任务

对定义 Ansible 执行的操作的单个模块的引用。

角色

角色是一种将功能放在"库"中可重复利用代码的方法，然后根据需要用于任何 playbook。

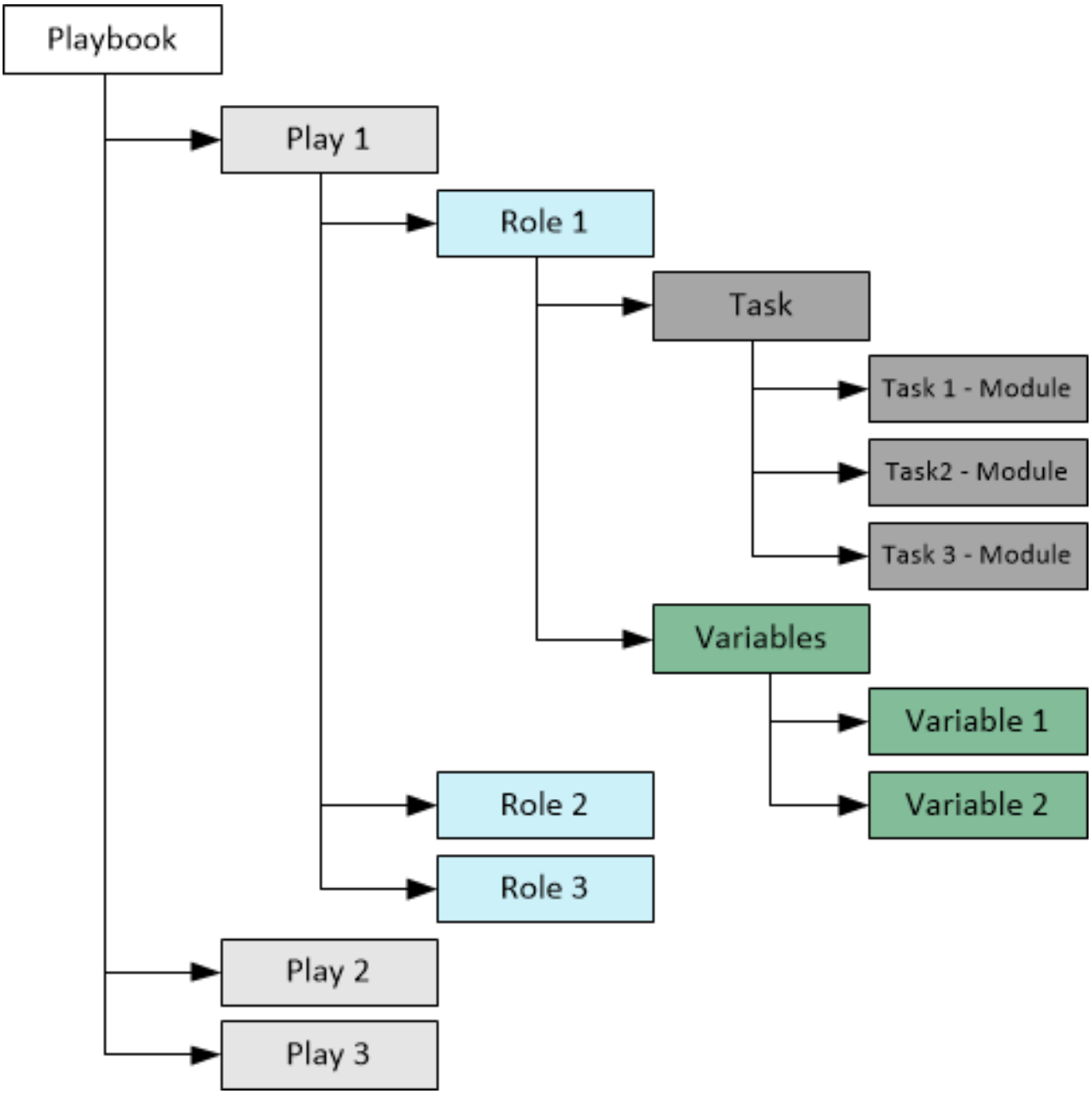
模块

Ansible 在受管节点上运行的代码或二进制代码单元。

Ansible 模块以集合的形式分组，*每个模块都有一个完全限定域名* (FQCN)。任务由模块执行，每个模块都在 playbook 中执行特定的任务。模块包含用于决定执行任务的时间和位置以及执行它的元数据。有数千个 Ansible 模块可以执行所有类型的 IT 任务，例如：

- 云管理
- 用户管理
- 网络
- 安全性
- 配置管理

- 通信



1.2. 如何使用 ANSIBLE PLAYBOOK？

Ansible 使用 YAML 语法。YAML 是一种人类可读的语言，可让您创建 playbook，而无需学习复杂的编码语言。

如需有关 YAML 的更多信息，请参阅 YAML 语法并考虑为您的文本编辑器安装附加组件，请参阅 其他工具和程序，以帮助您在 playbook 中编写干净的 YAML 语法。

使用 Ansible Playbook 的方法有两种：

- *使用命令行界面* (CLI)
- 使用 Red Hat Ansible Automation Platform 的按钮式部署。

1.2.1. 使用 CLI

安装开源 Ansible 项目或 Red Hat Ansible Automation Platform 后，使用以下命令

```
$ sudo dnf install ansible
```

在 Red Hat Enterprise Linux CLI 中，您可以使用 **ansible-playbook** 命令运行 Ansible Playbook。

1.2.2. 从平台内

Red Hat Ansible Automation Platform 用户界面提供按钮式 Ansible Playbook 部署，可用于更大作业或作业模板的一部分。这些部署附带额外的保护措施，对较新的 IT 自动化的用户特别有用，或者那些没有大量在 CLI 中工作经验的用户。

1.3. 使用 ANSIBLE 启动自动化

通过创建一个自动化项目、构建清单并创建 **Hello World** playbook 来开始使用 Ansible。

先决条件

- 必须安装 Ansible 软件包。

流程

- 在文件系统上创建一个项目文件夹。

```
mkdir ansible_quickstart
cd ansible_quickstart
```

通过使用单个目录结构，可以更轻松地添加到源控制，并重复使用和共享自动化内容。

1.4. 构建清单

清单将受管节点组织到集中文件中，该文件为 Ansible 提供系统信息和网络位置。使用清单文件，Ansible 可以通过单个命令管理大量主机。要完成以下步骤，您需要至少一个主机系统的 IP 地址或完全限定域名 (FQDN)。出于演示目的，主机可以在容器或虚拟机本地运行。

您还必须确保您的公共 SSH 密钥添加到每个主机上的 **authorized_keys** 文件中。使用以下步骤构建清单。

流程

在您创建的 **ansible_quickstart** 目录中，创建一个名为 **inventory.ini** 的文件。向 inventory.ini 文件中添加一个新的 **[myhosts]** 组，并指定每个主机系统的 IP 地址或完全限定域名(FQDN)。

```
[myhosts]
192.0.2.50
192.0.2.51
192.0.2.52
```

使用以下命令验证您的清单：

```
ansible-inventory -i inventory.ini --list
```

使用以下方法 ping 清单中的 **myhosts** 组：

```
ansible myhosts -m ping -i inventory.ini
```

如果在控制节点和受管节点上的用户名不同，请将 **-u** 选项与 Ansible 命令一起传递。

```
192.0.2.50 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
192.0.2.51 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
192.0.2.52 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

您已成功构建了一个清单。

1.4.1. 以 INI 或 YAML 格式的清单

您可以在 INI 文件或 YAML 中创建清单。在大多数情况下，如上例，CIN 文件对于少量受管节点来说比较简单，易于阅读。当受管节点的数量增加时，使用 YAML 格式创建清单会变得可行。

例如，以下等同于 **inventory.ini**，它为受管节点声明唯一名称，并使用 **ansible_host** 字段：

```
myhosts:
  hosts:
    my_host_01:
      ansible_host: 192.0.2.50
    my_host_02:
      ansible_host: 192.0.2.51
    my_host_03:
      ansible_host: 192.0.2.52
```

1.4.2. 构建清单的提示

- 确保组名称具有有意义的且唯一性。
- 组名称也区分大小写。
- 不要在组名称中使用空格、连字符或前面的数字（使用 floor_19，而不是 19th_floor）。
- 根据清单中的主机，逻辑上和何时对清单中的主机进行分组：

- what: 根据拓扑对主机进行分组，例如：db、web、leaf、spine。
- 其中：按地理位置对主机进行分组，例如： datacenter、region、floor、build。
- when ：按阶段分组主机，例如：development, test, staging, production。

1.4.3. 使用 metagroups

创建一个元组，以使用以下语法在清单中组织多个组：

```
metagroupname:
  children:
```

以下清单演示了数据中心的基本结构。这个示例清单包含一个网络 metagroup，其中包含所有网络设备和一个 datacenter metagroup，其中包含 network group 和 all webservers。

```
leafs:
  hosts:
    leaf01:
      ansible_host: 192.0.2.100
    leaf02:
      ansible_host: 192.0.2.110

spines:
  hosts:
    spine01:
      ansible_host: 192.0.2.120
    spine02:
      ansible_host: 192.0.2.130

network:
  children:
    leafs:
    spines:

webservers:
  hosts:
    webserver01:
      ansible_host: 192.0.2.140
    webserver02:
      ansible_host: 192.0.2.150

datacenter:
  children:
    network:
    webservers:
```

1.5. 创建变量

为受管节点设置值的变量，如 IP 地址、FQDN、操作系统和 SSH 用户，因此在运行 Ansible 命令时不需要传递它们。

变量可以应用到特定的主机。

```
webservers:
  hosts:
    webserver01:
      ansible_host: 192.0.2.140
      http_port: 80
    webserver02:
      ansible_host: 192.0.2.150
      http_port: 443
```

变量也可以应用到组中的所有主机。

```
webservers:
  hosts:
    webserver01:
      ansible_host: 192.0.2.140
      http_port: 80
    webserver02:
      ansible_host: 192.0.2.150
      http_port: 443
  vars:
    ansible_user: my_server_user
```

如需有关清单和 Ansible 清单变量的更多信息，请参阅关于安装程序 清单文件和 清单文件变量。

1.6. 创建第一个 PLAYBOOK

使用以下步骤创建 ping 主机并输出 "Hello world" 消息的 playbook。

流程

1. 在 **ansible_quickstart** 目录中创建一个名为 **playbook.yaml** 的文件，其内容如下：

```
- name: My first play
  hosts: myhosts
  tasks:
   - name: Ping my hosts
     ansible.builtin.ping:

   - name: Print message
     ansible.builtin.debug:
      msg: Hello world
```

2. 使用以下命令运行 playbook：

```
ansible-playbook -i inventory.ini playbook.yaml
```

3. Ansible 返回以下输出：

```
PLAY [My first play] ***********************************************************************

TASK [Gathering Facts] *********************************************************************
ok: [192.0.2.50]
ok: [192.0.2.51]
```

```
ok: [192.0.2.52]

TASK [Ping my hosts] ***********************************************************************
ok: [192.0.2.50]
ok: [192.0.2.51]
ok: [192.0.2.52]

TASK [Print message] ************************************************************************
ok: [192.0.2.50] => {
    "msg": "Hello world"
}
ok: [192.0.2.51] => {
    "msg": "Hello world"
}
ok: [192.0.2.52] => {
    "msg": "Hello world"
}

PLAY RECAP *********************************************************************************************
192.0.2.50: ok=3    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
192.0.2.51: ok=3    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
192.0.2.52: ok=3    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

在这个输出中，您可以看到：

- 提供 play 和每个任务的名称。始终使用描述性名称，使 playbook 易于验证和故障排除。
- Gather Facts 任务隐式运行。默认情况下，Ansible 会收集有关可以在 playbook 中使用的清单的信息。
- 每个任务的状态。每个任务都有一个 **ok** 状态，表示它成功运行。
- play 总结用于汇总每个主机 playbook 中所有任务的结果。在本例中，有三个任务，因此 **ok=3** 表示每个任务都成功运行。

第 2 章 使用 PLAYBOOK 建立与受管节点的连接

要确认您的凭证，您可以手动连接到网络设备并检索其配置。将示例用户和设备名称替换为您的实际凭证。

例如，对于 VyOS 路由器：

```
ssh my_vyos_user@vyos.example.net
show config
exit
```

2.1. 运行网络 ANSIBLE 命令

您可以使用单个 Ansible 命令检索其配置，而不是在网络设备上手动连接和运行命令。

```
ansible all -i vyos.example.net, -c ansible.netcommon.network_cli -u \
my_vyos_user -k -m vyos.vyos.vyos_facts -e \
ansible_network_os=vyos.vyos.vyos
```

此命令中的标志设置七值：

- 命令应应用到的主机组（本例中为 **all**）
- 清单(**-i**、目标的设备或设备 - 没有结尾的逗号 **-i** 指向清单文件)
- 连接方法(**-c**，连接和执行 ansible 的方法)
- 用户(**-u**，SSH 连接的用户名)
- SSH 连接方法(-**k**，提示输入密码)
- 模块(**-m**，要使用的 Ansible 模块，使用完全限定的集合名称(FQCN))
- 额外的变量（本例中为 **-e**，设置网络操作系统值）

注意

如果您将 **ssh-agent** 与 ssh 密钥一起使用，Ansible 会自动加载它们。您可以省略 **-k** 标志。

如果您在虚拟环境中运行 Ansible，还必须添加变量 **ansible_python_interpreter=/path/to/venv/bin/python**。

2.2. 运行网络 ANSIBLE PLAYBOOK

如果要每天运行特定命令，可以将其保存在 playbook 中，并使用 **ansible-playbook** 而不是 ansible 运行它。该 playbook 可以存储您在命令行中提供的许多参数，从而减少在命令行中键入的内容。您需要两个文件，即 playbook 和清单文件。

先决条件

从此处 下载 **first_playbook.yml**。

playbook 类似如下：

```
---

- name: Network Getting Started First Playbook
  connection: ansible.netcommon.network_cli
  gather_facts: false                          (1)
  hosts: all
  tasks:

    - name: Get config for VyOS devices
      vyos.vyos.vyos_facts:
        gather_subset: all

    - name: Display the config
      debug:
        msg: "The hostname is {{ ansible_net_hostname }} and the OS is {{ ansible_net_version }}"
```

标签	描述
gather_facts	Ansible 的原生事实收集(**ansible.builtin.setup**)已在此处禁用，因为 playbook 依赖于此网络集合中特定于平台的模块(**vyos.vyos_facts**)提供的事实。

playbook 从上面的命令行设置三个七个值：

- 组(**主机：all**)
- **连接方法（连接：ansible.netcommon.network_cli**）和
- 模块（在每个任务中）。

使用 playbook 中设置的这些值，您可以在命令行中省略它们。该 playbook 还会添加第二项任务来显示配置输出。

当从系统中收集事实时，通过特定于集合的事实模块（如 **vyos.vyos.vyos_facts** 或 **ansible.builtin.setup** ）收集到系统时，收集的数据会被保留在内存中，而不是写入到控制台。

当模块在 playbook 中运行时，输出会保留在内存中，供将来的任务使用，而不是写入控制台。在大多数其他模块中，您必须明确注册一个变量来存储和重复使用模块或任务的输出。

有关事实的更多信息，请参阅 *Ansiible Playbook 参考指南* 中的 [Ansible 事实]。

以下调试任务可让您在 shell 中看到结果。

流程

1. 使用以下命令运行 playbook：

```
ansible-playbook -i vyos.example.net, -u ansible -k -e ansible_network_os=vyos.vyos.vyos
first_playbook.yml
```

   playbook 包含一个含有两个任务的 play，并生成如下输出：

```
$ ansible-playbook -i vyos.example.net, -u ansible -k -e ansible_network_os=vyos.vyos.vyos
first_playbook.yml

PLAY [Network Getting Started First Playbook]
**********************************************************************************************************
**********

TASK [Get config for VyOS devices]
**********************************************************************************************************
**********
ok: [vyos.example.net]

TASK [Display the config]
**********************************************************************************************************
**********
ok: [vyos.example.net] => {
    "msg": "The hostname is vyos and the OS is VyOS 1.1.8"
}
```

2. 现在，您可以检索设备配置，您可以尝试使用 Ansible 更新它。

3. 从此处 下载 **first_playbook_ext.yml**，这是第一个 playbook 的扩展版本：
playbook 类似如下：

```
---

- name: Network Getting Started First Playbook Extended
  connection: ansible.netcommon.network_cli
  gather_facts: false
  hosts: all
  tasks:

    - name: Get config for VyOS devices
      vyos.vyos.vyos_facts:
        gather_subset: all

    - name: Display the config
      debug:
        msg: "The hostname is {{ ansible_net_hostname }} and the OS is {{ ansible_net_version
}}"

    - name: Update the hostname
      vyos.vyos.vyos_config:
        backup: yes
        lines:
          - set system host-name vyos-changed

    - name: Get changed config for VyOS devices
      vyos.vyos.vyos_facts:
        gather_subset: all

    - name: Display the changed config
      debug:
        msg: "The new hostname is {{ ansible_net_hostname }} and the OS is {{
ansible_net_version }}"
```

4. 扩展的第一个 playbook 在单个 play 中有五个任务。

5. 使用以下命令运行 playbook：

```
$ ansible-playbook -i vyos.example.net, -u ansible -k -e ansible_network_os=vyos.vyos.vyos
first_playbook_ext.yml
```

6. 输出显示您对配置所做的更改：

```
$ ansible-playbook -i vyos.example.net, -u ansible -k -e ansible_network_os=vyos.vyos.vyos
first_playbook_ext.yml

PLAY [Network Getting Started First Playbook Extended]
*********************************************************************************************************************
*******************

TASK [Get config for VyOS devices]
*********************************************************************************************************************
*****************
ok: [vyos.example.net]

TASK [Display the config]
*********************************************************************************************************************
*******************
ok: [vyos.example.net] => {
    "msg": "The hostname is vyos and the OS is VyOS 1.1.8"
}

TASK [Update the hostname]
*********************************************************************************************************************
*******************
changed: [vyos.example.net]

TASK [Get changed config for VyOS devices]
*********************************************************************************************************************
*******************
ok: [vyos.example.net]

TASK [Display the changed config]
*********************************************************************************************************************
*******************
ok: [vyos.example.net] => {
    "msg": "The new hostname is vyos-changed and the OS is VyOS 1.1.8"
}

PLAY RECAP
*********************************************************************************************************************
*******************
vyos.example.net           : ok=5    changed=1    unreachable=0    failed=0
```

2.3. 从网络设备收集事实

gather_facts 关键字支持在标准化键/值对中收集网络设备事实。您可以将这些网络事实传送到进一步的任务中，以管理网络设备。您还可以使用 **gather_network_resources** 参数和 network ***_facts** 模块（如 **arista.eos.eos_facts**）返回设备配置的子集，如下所示。

```
- hosts: arista
  gather_facts: True
  gather_subset: interfaces
  module_defaults:
    arista.eos.eos_facts:
      gather_network_resources: interfaces
```

playbook 返回以下接口事实：

```
"network_resources": {
      "interfaces": [
          {
              "description": "test-interface",
              "enabled": true,
              "mtu": "512",
              "name": "Ethernet1"
          },
          {
              "enabled": true,
              "mtu": "3000",
              "name": "Ethernet2"
          },
          {
              "enabled": true,
              "name": "Ethernet3"
          },
          {
              "enabled": true,
              "name": "Ethernet4"
          },
          {
              "enabled": true,
              "name": "Ethernet5"
          },
          {
              "enabled": true,
              "name": "Ethernet6"
          },
      ]
  }
```

注意

gather_network_resources 将配置数据呈现为所有支持资源的事实 (**interfaces/bgp/ospf/etc'**)，而 **gather_subset** 则主要用于获取操作数据。

您可以存储这些事实，并直接在另一个任务中使用它们，如 **eos_interfaces** 资源模块。

第 3 章 ANSIBLE PLAYBOOK 的实际示例

Ansible 可以与许多不同的设备分类通信，从基于云的 REST API 到 Linux 和 Windows 系统、网络硬件等。

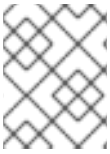
以下是两个 Ansible 模块示例，它会自动更新两种类型的服务器。

3.1. PLAYBOOK 执行

playbook 运行从上到下的顺序。在每个 play 中，任务也从上到下的顺序运行。具有多个"play"的 playbook 可以编配多计算机部署，在您的 webservers 上运行一个 play，然后在您的数据库服务器上运行另一个 play，然后在网络基础架构上运行第三个 play，等等。

每个 play 至少定义了两个内容：

- 指向目标的受管节点，使用模式
- 至少一个要执行的任务



注意

在 Ansible 2.10 及更高版本中，使用 playbook 中的完全限定集合名称来确保选择了正确的模块，因为多个集合可以包含具有相同名称的模块(**如用户**)。

如需更多信息，请参阅在 playbook 中使用集合。

在本例中，第一个 play 以 web 服务器为目标；第二个 play 以数据库服务器为目标。

```
---
- name: Update web servers
  hosts: webservers
  become: true

  tasks:
    - name: Ensure apache is at the latest version
      ansible.builtin.yum:
        name: httpd
        state: latest
    - name: Write the apache config file
      ansible.builtin.template:
        src: /srv/httpd.j2
        dest: /etc/httpd.conf
        mode: "0644"

- name: Update db servers
  hosts: databases
  become: true

  tasks:
    - name: Ensure postgresql is at the latest version
      ansible.builtin.yum:
        name: postgresql
        state: latest
    - name: Ensure that postgresql is started
      ansible.builtin.service:
```

```
    name: postgresql
    state: started
```

playbook 包含两个 play：

- 首先检查 Web 服务器软件是最新的，并在需要时运行更新。
- 第二个检查数据库服务器软件是否是最新的，并在需要时运行更新。

您的 playbook 不仅仅包含主机行和任务。

例如，此示例 playbook 为每个 play 设置 remote_user。这是 SSH 连接的用户帐户。您可以在 playbook、play 或任务级别添加其他 Playbook 关键字，以影响 Ansible 的行为方式。Playbook 关键字可以控制连接插件，是否使用特权升级、如何处理错误等。

为了支持各种环境，Ansible 可让您将这些参数设置为命令行标志、Ansible 配置或清单中。了解这些数据源的优先规则可帮助您扩展 Ansible 生态系统

第 4 章 开源许可证

GNU 通用公共许可证

版本 3，2007 年 6 月 29 日

任何人都可以复制和分发此许可证文档的动词副本，但不允许更改。

preamble
GNU 通用公共许可证是免费的、复制软件的版权许可证以及其他工作许可证。

大多数软件和其他实际工作的许可证旨在使您的自由共享和改变工作。相反，GNU 通用公共许可证旨在确保您自由共享和更改程序的所有版本 - 以确保其为所有用户保留了免费软件。自由软件基础将 GNU 通用公共许可证用于我们的大部分软件；它也适用于作者通过此方式发布的任何其他工作。您还可以将其应用到您的计划中。

当我们介绍免费软件时，我们指的是自由，而非价格。我们的通用公共许可证旨在确保您有自由地分发免费软件副本（如果需要，您可以收到源代码或获得源代码）。如果需要，您可以更改软件或在新的免费计划中使用部分软件，并且您知道您可以执行这些软件。

为了保护您的权利，我们需要防止其他人拒绝您这些权利，或要求您提高权利。因此，如果您分发软件副本或修改软件，或者修改它，则您有一定的职责。

例如，如果您分发此类程序的副本（无论是 gratis 还是付费），则必须传递给接收的相同自由。您必须确保它们（您也可以接收）或获取源代码。您必须向他们展示这些术语，以便他们知道它们的权限。

使用 GNU GPL 的开发人员通过两个步骤来保护您的权利：(1)作为软件版权，(2)向您提供此许可证，为您提供法律权限进行复制、分发和/或修改它。

对于开发人员和作者保护，GPL 清楚地说明了此免费软件没有保证。对于用户和作者而言，GPL 要求修改的版本标记为更改，因此它们的问题不会错误地归于以前版本的作者。

有些设备旨在拒绝用户访问或在它们中安装修改的软件版本，尽管制造商可以这样做。这从根本上不兼容，目的是防止用户修改软件的自由。此类滥用的系统模式发生在供个人使用的产品中，这是最不可接受的位置。因此，我们设计了此版本的 GPL 版本，以禁止对这些产品的实践。如果其他域中出现大量问题，我们只要为了保护用户的自由，我们准备将此配置扩展到这些域。

最后，每个计划都会受到软件隐患不断威胁。状态应该不允许对一般用途计算机的开发和使用软件进行限制，但希望在那些确实这样做的情况下，我们希望避免应用于免费程序的特殊危险程序，使它能够有效地进行专有。为了防止这种情况，GPL 保证无法用来使程序自由使用。

以下是复制、分发和修改的确切条件和条件。

条款和条件

0.定义。

"此许可证"是指 GNU 通用公共许可证的版本 3。

"版权"还意味着类似于版权的法律，适用于其他种类的工作，如半导体掩码。

"计划"是指根据许可证获得许可的任何可版权工作。每个许可证均以"您"的形式解决。"许可"和"接收"可能是个人或组织。

要"修改"工作意味着复制或调整所有或部分工作，需要版权权限，而不是进行精确的副本。生成的工作称为早期工作的"修改版本"，或称为"基于"之前工作的工作。

"接管工作"是指根据计划没有修改的程序或工作。

为了"传播"工作，即无需任何权限即可直接或对适用版权法的侵权，除在计算机上执行或修改私有副本外，进行任何操作。传播包括复制、分发（带有或不修改），供公众和其它活动使用。

要"整合"工作，这意味着任何类型的传播，使其他各方能够生成或接收副本。我通过计算机网络与用户交互（不转让副本）并不是传达。

交互式用户界面向扩展显示"相关法定法声明"，它包含方便且有明显的功能，(1)显示合适的版权通知，(2)告知用户本许可证下工作没有保证的副本（除提供该许可证的程度除外），该许可证可能会传达此许可证下的工作，以及查看此许可证的副本。如果接口显示用户命令或选项的列表，如菜单，则列表中的一个项目会满足此条件。

1.源代码.

工作中的"源代码"表示对它进行修改的首选形式。"目标代码"是指工作的任何非源形式。

"标准接口"是指由可识别的标准正文定义的官方标准，或者如果为特定编程语言指定的接口，一个接口广泛用于此语言。

可执行工作中的"系统库"包括任何一个整体工作（除作为整体工作外），(a)包含在一个 Major 组件中，但不是该主版本组件的一部分，(b)仅用于启用该主版本组件的使用，或实施一个标准接口，以便以公共代码形式提供给公共代码。在这种情况下，"Major 组件"是指运行该工作的特定操作系统（如内核、窗口系统等）的主要基本组件（如有）。

在对象代码表单中工作的"Corresponding Source"意味着生成、安装和（针对可执行工作）所需的所有源代码都会运行对象代码并修改工作，包括控制这些活动的脚本。但是，它不包括工作的系统库或通用的工具，或者通常可用的程序，这些程序在执行这些活动时未修改，但不是工作的一部分。例如，Corresponding Source 包括与工作源文件关联的接口定义文件，以及共享库的源代码，以及动态链接子代码，这些工作专门针对需要设计，如入侵数据通信或控制工作子部分之间的流。

Correspoing Source 不需要包括用户可以自动从 Corresponding Source 的其他部分重新生成的任何内容。

源代码形式的 Correspoing Source 是相同的工作。

2.基本权限.

在此许可证下授予所有权利，授予该计划版权条款，并在满足上述条件时提供不可撤销的。此许可证明确使您的无限制权限明确地运行未修改的程序。只有当输出给定其内容时，才会涵盖运行所涵盖工作的输出，此许可证才会构成了一系列的工作。根据版权法律提供的，此许可证确认您的公平使用或其他同等权利。

您可以进行、运行和传播涵盖的工作量，只要您的许可证处于强制状态，就没有条件。您可能出于单独您修改的唯一目的而让他们单独做修改，或者为您提供运行这些工作的工具，前提是您遵守本许可证条款，以消除您不控制版权的所有材料。这使得或运行覆盖的工作必须完全代表您的指示和控制，根据您的指导和控制，阻止他们与您联系外的版权材料之外的任何副本。

在任何其他情况下，完全根据以下规定的条件才允许。不允许使用 Sublicensing；第 10 节不需要。

3.保护用户来自 Anti-Circumvention Law 的法务权。

在 WIPO 版权处理在 20 月 1996 年 12 月 1996 日或类似法律禁止或限制此类措施中，任何适用法律规定的适用法律规定不得被认为是有效的技术措施的一部分。

当您传达了涵盖的工作时，您担心任何法律能力，防止出现技术法法的程度上，此类范围（如绕过、受此许可证权权，对所涵盖工作的权利）进行限制，您可以禁止以强制方式限制操作或修改工作，从而针对您的个人或第三方的法律权限来衡量。

4.传达 Verbatim Copies.

您可以在任何介质中收到程序的源代码的口头副本，并适当发布每个副本的版权通知；始终始终声明此许可证以及任何许可证所添加的任何非许可条款。

您可以收取每个副本的任何价格或无价格，并且您可以为付费提供支持或保证保护。

5.制作修改过的源版本.

您可以基于计划，或以第 4 节"第 4 节"条款中的源代码形式从程序生成它的工作，您也可以满足所有这些条件：

- a）工作必须执行相应的通知，表示您修改过，并给出相关日期。
- B）工作必须执行相应的通知，说明它在此许可证下发布，以及第 7 节下添加的所有条件。此要求将第 4 节中的要求修改为"保持完整所有通知"。
- C）您必须根据本许可证向拥有副本的任何人提供整个工作许可证。因此，无论如何打包它们，都会应用此许可证，以及任何适用的第 7 节。此许可证没有以其它方式许可证工作的权限，但如果单独收到了这个权限，则不会使此类权限无效。
- D）如果工作具有交互式用户界面，则每个操作都必须显示相关的法务法标语；但是，如果程序有没有显示适用法法法案的互动界面，则您的工作需求不会这样做。涵盖了其他单独和独立工作的编译是其他独立工作的，它们不是由所涵盖的工作的性质扩展，它们没有与其组成大型程序（例如，在存储或分发介质的卷中或存储或分发介质的卷中）称为"aggregate"（如果编译及其生成的版权被用来限制编译用户的访问权限或法律权利）。在聚合中包含涵盖的工作不会导致此许可证应用到聚合的其他部分。

6.传达非源表单.

您可根据第 4 和 5 节中的第 4 和 5 节"第 4 和 5 节"条款将涵盖的一系列工作以对象代码的形式编写，您也以以下方法之一传达机器可读的 Correspoing Source：

- a）遵循对象代码，或是物理产品（包括物理分发介质）提供的物理产品（包括物理分发介质），由 Correspoing Source 告知 Corresponding Source （自定义为软件干预）。
- B）遵循对象代码，或是物理产品（包括物理分发介质）、一个由书面提供的提供的、至少为三年有效的对象代码，只要为该产品提供备用部分或客户支持，为拥有对象代码((1))提供内容代码的任何人提供有效。在软件交换的持久化物理介质上，价格不多于您实际执行此类来源的合理成本，或者(2)访问从网络服务器复制 Corresponding 源，无需收费。
- c）将对象代码的单独副本与写入的产品副本一致，以提供 Corresponding Source。这个替代方法只在偶尔或非商业情况下被允许，只有在您收到了带有此类提供的对象代码时（根据第 6b 节），才允许此替代方法。
- D）通过提供指定位置(Gratis 或收费)访问对象代码，并提供与协调源相同的方式相同的访问，无需进一步收费。您不需要接收者来复制 Corresponding Source 以及目标代码。如果复制对象代码是一个网络服务器的位置，则 Correspoing Source 可能位于支持同等复制设施的不同服务器（由

您或第三方运行）上，只要您保持了对象代码旁边的清晰指示，表示在哪里找到 Correspoing Source。无论哪个服务器托管 Correspoing Source，您都会保持义务，只要需要满足这些要求。

- e）使用 peer-to-peer 传输来追踪对象代码，告知其他对等对象代码和协调工作源在第 6 节下不收取额外费用。

对象代码的隔离部分（其源代码不包括在 Correspoing Source 中作为系统库）中，不需要包含在聚合对象代码工作中。

"用户产品"是(1)个"消费者产品"，这意味着任何通常用于个人、家族或家族的个人属性，或者(2)任何专门为公司设计或出售的个人属性。在确定产品是否为消费者产品时，有疑的情况应该能被解决以覆盖范围。对于特定用户收到的特定产品，"常规使用"是指该类产品的典型或常见用途，无论特定用户或特定用户实际使用的方式，或者想要使用或希望使用该产品。产品是消费者产品，无论产品是否具有大量商业、工业还是非消费者用途，除非使用这种产品代表使用产品的唯一显著模式。

用户产品的"安装信息"是指从 Correspond Source 的修改版本，安装和执行用户产品中所涵盖工作所需的任何方法、流程、授权密钥或其他信息。信息必须足以确保修改的对象代码持续正常工作，没有因为进行了修改而无法完全阻止或干扰。

如果您在本节下传达对象代码工作，或者特别使用，或者特别用于用户产品，并作为拥有和使用 User Product 的事务的一部分而转移到固定术语（与事务字符相同），则 Corresnding Source 被传输至该部分的接收者，或用于固定术语（与事务字符相同）。但是，如果您和任何第三方都没有在 User 产品上安装修改的对象代码（例如，已在 ROM 中安装的工作），则此要求不适用。

提供安装信息的要求不包括继续提供支持服务、保证或更新已由接收者修改或安装的工作，或针对已修改或安装的 User 产品的要求。当修改本身材料并破坏网络操作或违反了网络间通信的规则和协议时，对网络的访问可能会被拒绝。

根据本节提供对应的 Sourceveyed 和 Installation Information，必须采用公开记录的格式（在源代码表单中面向公共的实施），且不需要特殊的密码或密钥来解包、读取或写入密钥。

7.其他条款.

"额外权限"是指通过从一个或多个条件中进行例外来补充此许可证条款的术语。适用于整个计划的其他权限应像在此许可证中包含的那样被视作，到他们在适用法律下有效的范围内。如果其他权限仅适用于计划的一部分，则该部分可以在这些权限下单独使用，但整个程序仍会受到这个许可证的管控，而无需考虑额外的权限。

当您传达了涵盖工作的副本时，您可以选择从该副本中删除任何其他权限，或从其中的任何部分中删除。（在某些情况下，可以编写其他权限，以便在修改工作时需要自己删除。）您可以将其他权限放在由您添加到覆盖工作中的材料上，或者可以授予适当的版权权限。

就此许可证的其他配置，对于添加到所涵盖工作的材料，您可以（如果由该材料的版权持有者授权）对许可证的条款加补充：

- a）与本许可证的第 15 和 16 部分的免除或限制性不同；或者
- B）获取指定合理的法律通知或本材料中的适用法律通知或作者，或通过包含该材料的适用法法标标标语显示，或者
- c）对材料的来源有误解，或要求此类材料的修改版本以合理的方式标记为与原始版本不同；或者
- D）限制用于公共目的的许可证或材料作者的名称；或者
- e）Declining 在商标法下授予使用某些交易名称、商标或服务标记的权利；或者

- f）遵守这些材料的许可者和作者由编写材料的材料（或修改的版本）以向接收方提供合同假设，保证这些合同假设直接对他们提出了任何责任。

所有其他非允许的其他术语在第 10 节的含义中都被视为"模糊限制"。如果您收到的程序或其中的任何部分，则包含一个通知，表示它由此许可证管理，以及作为进一步限制的术语，您可以删除该术语。如果许可证文档包含进一步的限制，但允许在此许可证下识别或传播，您可以添加到由该许可证文档条款管理的一系列工作材料中，只要进一步限制不会保留此类限制或传播。

如果您根据本节将术语添加到所涵盖的工作中，您必须在相关的源文件、适用于这些文件的其他术语声明中，或者指出在哪里找到适用的条款。

其他条款(permissive)可能会以单独写入许可证的形式声明，或按照例外情况声明；上述要求适用。

8.终止.

您可能无法传播或修改涵盖的工作，除非在此许可证下提供明确。否则，传播或修改它的任何尝试都无效，并将根据此许可证自动终止您的权限（包括第 11 节的第三个段落下授予的任何不权许可证）。

但是，如果您停止违反了此许可证，则来自特定版权持有者的许可证将退出(a)配置，除非和直到版权持有者明确终止您的许可证，并且(b)永久终止您的许可证，如果版权持有者未能通过某种合理的方式通知您，在校验之后的 60 天前，如果版权持有者无法通知您违反任何违反方式。

此外，如果版权持有者通知您某种合理的方式，则来自特定版权持有者的许可将永久恢复，这是您第一次收到此许可证的通知（针对任何工作，您会在收到通知后的 30 天前通知）。

在本节下终止您的权限不会终止您收到此许可证下的副本或权利的各方许可证。如果您的权利已被终止且没有永久恢复，则您没有在 10 节中接收相同材料的新许可证。

9.不需要接受 Having Copies。

您不需要接受此许可证才能接收或运行计划副本。由于使用对等传输来接收副本后，仅进行一系列工作的辅助传播，因此不需要接受。但是，除此许可证以外的其他操作都不授予您传播或修改任何涵盖的工作的权限。如果您不接受此许可证，则这些不侵权版权操作。因此，通过修改或传播涵盖的工作，您可以指示您接受此许可证。

10.自动许可 Downstream Recipients.

每次您传递了涵盖的工作时，接收方都会自动收到来自原始许可的许可证，以便运行、修改和传播该工作，受此许可证约束。您不负责强制使用此许可证的第三方合规性。

"用户交易"是机构传输控制的事务，或者大大传输一个机构或合并机构的所有资产。如果从实体事务传播所涵盖的工作结果，则每个方都会收到工作副本的事务，并接收任何许可证到方感兴趣的工作中，或者可能在前面的段落下给出或可以具有合理的努力的权利，并有权拥有与参与参与方的前身后期的工作源。

您不能对本许可证下授予或从属权利的练习进行进一步限制。例如，您不能为本许可证下授予的权利提出许可费、免除或其他费用，并且您可能无法启动许可（包括法律部门中的跨声明或计数器声明），以免除此许可证、使用、销售、分发或导入计划或其中任何部分的义务被侵犯。

11.免除法案。

"贡献者"是授权根据计划许可证使用的版权持有者，或计划所基于的工作。因此，许可证的工作被称为贡献者的 "contributor version"。

供稿者的"主办方声明"是指由贡献者拥有或控制的披露，无论是已获取或之后，这些声明都将受到某种程度的侵权，由此许可证允许、使用或销售其贡献者版本，但不包括仅在对贡献版本进行进一步修改的声明。就此定义而言，"控制"包括以与此许可证要求一致的方式授予敏感性子许可的权利。

每个贡献者都授予您全球非独占性、免除性、免除性、免除性、免除性、免除性、出版商的免除法索索赔、使用、销售、提供销售、导入和其他运行、修改和传播其贡献版本的内容。

在以下三个段落中，"patent 许可证"是指任何表达协议或承诺，但并没有声明，而不是强制冲突（例如，一种表达法权，以免于侵权权力或达成冲突）。为了"质量"此类向方的隐患许可证意味着进行此类协议或承诺不履行对方的隐患。

如果您委托了涵盖的工作，知道依赖加权许可证，并且对于任何人都无法复制、免费收费、通过公开可用的网络服务器或其他易访问的方式，您不得不提供该许可证的保证源。或(2)安排您去除此特定工作中此类敏感度许可证的福利，或是是与此许可证的要求保持一致的方式，将无数额许可证扩展到下游收件人。"了解"意味着，您拥有实际知识，但对于非法务许可、您在国家/地区涵盖的工作或您的收件人在某个国家/地区中涵盖的工作时，将对本国家/地区所涵盖的工作有侵权，导致您在该国家/地区有责任。

如果致力于或与单个交易或安排连接或连接，您同意或传播，由流程保证，覆盖的工作，并为接收所涵盖工作的部分工作授权授权，并且根据所涵盖的工作授权、传播、修改或传达或传达许可证，自动向所涵盖的工作提供方的授权。

如果免除覆盖范围内没有包括其覆盖范围、禁止练习或针对此许可证授予的一个或多个权利，则备受制约的许可证是"免除者"，如果其范围不包括在这一许可证下特别授予的权利。如果您与位于分发软件业务的第三方参与的第三方进行安排，则不得向第三方发出支付工作，根据参与工作的程度向第三方支付，以及第三方授予第三方，并且向第三方接收所涵盖工作的任何方做出付款。与包含所涵盖工作的副本（或从这些副本所做的副本）或(b)与包含所涵盖工作的副本的连接时或(b)主要与包含所涵盖产品或编译的特定产品或编译相连接，除非您进入该安排，否则在 2007 年 3 月 28 日之前被授权，或(b)与包含所涵盖工作的特定产品或编译相连接。

此许可证中的任何内容都应认为是排除或限制任何暗示的许可，或将其他防御权限制给您，否则可能会提供给您适用的法务法。

12.其他自由选择权益.

如果您对您实施条件（无论是根据订单、协议或以其他方式排列此许可证条件），则它们不会让您从此许可证的条件中移出。如果您无法同时满足许可证下的义务，以及其它相关义务，因此您可能根本不会让它满足。例如，如果您同意同意，为了进一步从那些您传达计划的人那里收集起来，那么您可以满足这些条款的唯一方式，并且此许可证将完全从计划传达。

13.与 GNU Affero General Public License 一起使用。

不满足此许可证的任何其他配置，您可以使用 GNU Affero General Public License 版本 3 下授权的工作链接或组合到单个组合工作中，并可使用生成的工作。此许可证条款将继续应用到涵盖的工作部分，但 GNU Affero General Public License （第 13 节）中的特殊要求将适用于通过网络的交互，因此它们将适用于组合。

14.修订了此许可证的版本。

Free Software Foundation 可能会及时发布修订版和/或新版本的 GNU 通用公共许可证。这些新版本将与当前的版本类似，但可能在解决新问题或关注方面有所不同。

每个版本都被赋予一个可分辨的版本号。如果计划指定 GNU General Public License "或更新的版本" 适用的某个编号版本"或任何更新的版本"，您可以选择遵循该版本或自由软件基础发布的任何更新版本的条款。如果计划没有指定 GNU General Public License 的版本号，您可以选择 Free Software Foundation 发布的任何版本。

如果计划指定代理可以决定可以使用 GNU General Public License 的未来版本，则该代理的公共声明会永久接受版本，则会让您为程序选择该版本。

之后的许可证版本可能会为您提供额外的或不同的权限。但是，由于您选择遵循更新的版本，不会对任何作者或版权持有者进行额外的义务。

15.免责声明.

在适用法律允许的范围内，该计划没有保证保证。除非写版权持有者和/或其他各方，否则其他各方将提供"原样"的保证计划，不保证特定目的的保证。计划质量和性能的整个风险与您一起。如果计划被缺陷证明，您需要假定所有必要服务、修复或纠正的成本。

16.责任限制.

除非适用法律或同意书面方，否则任何修改和/或传达计划的其他方都不必要，否则您不得承担破坏，包括任何常规、特权、事件或排序破坏，由此计划导致使用或无法使用计划。（包括但不仅限于丢失数据或数据被您或第三方或程序无法与任何其他程序操作而丢失的不准确或损失），即使建议此类拥有者或其他方，也会造成此类损坏的可能性。

17.第 15 和 16 段的解释.

如果上述责任的免除及上述责任限制无法根据条款授予本地法律效果，则审视者应采用本地法律要求，最接近于计划与计划相差的绝对绝对 waiver 责任，除非有保证或假设责任保证或假定计划返回的副本。

条款和条件结束

如何将条款应用到您的新计划
如果您开发了一个新计划，并且希望它成为公开的最大可能用途，那么实现此目标的最佳方法是使其免费软件，每个人都可以在这些条款下重新分发和更改。

为此，请将以下通知附加到程序：将它们附加到每个源文件的开头最安全，以最有效地说明 warranty 的排除；并且每个文件至少应"复制"行以及找到完整通知的指针。

> <one line to give the program's name and a brief idea of what it does.>
> Copyright (C) <year> <name of author>
>
> This program is free software: you can redistribute it and/or modify
> it under the terms of the GNU General Public License as published by
> the Free Software Foundation, either version 3 of the License, or
> (at your option) any later version.
>
> This program is distributed in the hope that it will be useful,
> but WITHOUT ANY WARRANTY; without even the implied warranty of
> MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
> GNU General Public License for more details.
>
> You should have received a copy of the GNU General Public License
> along with this program. If not, see <https://www.gnu.org/licenses/>.

另外，添加有关如何通过电子和纸质邮件与您联系的信息。

如果程序进行终端交互，使它在以互动模式启动时输出一个简短的通知：

> <program> Copyright (C) <year> <name of author>
> This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
> This is free software, and you are welcome to redistribute it
> under certain conditions; type `show c' for details.

假设命令"show w' 和"show c"应该显示通用公共许可证的相应部分。当然，您的程序的命令可能会有所不同；对于 GUI 界面，您将使用"关于框"。

如有情况，您也应让您的雇主（如果是作为程序员）或中介（如果有的话）为计划签名"版权免责声明"。有关这方面的更多信息，以及如何应用和遵循 GNU GPL，请参阅 <https://www.gnu.org/licenses/ >。

GNU 通用公共许可证不允许将程序合并到专有程序中。如果您的程序是一个子例程库，您可以考虑将专有应用程序与库的链接更为有用。如果这是您要做的，请使用 GNU Lesser General Public License 而不是这个许可证。请首先阅读 <https://www.gnu.org/licenses/why-not-lgpl.html>。