Red Hat Ansible Automation Platform 2.4

Red Hat Ansible Automation Platform 升级和迁移指南

升级并迁移 Ansible Automation Platform 的传统部署

Last Updated: 2024-07-03

Red Hat Ansible Automation Platform 2.4 Red Hat Ansible Automation Platform 升级和迁移指南

升级并迁移 Ansible Automation Platform 的传统部署

法律通告

摘要

本指南介绍了如何升级到最新的 Ansible Automation Platform 版本，并将旧的虚拟环境迁移到自动化执行环境。

目录

对红帽文档提供反馈

如果您对本文档有任何改进建议，或发现了任何错误，请通过 https://access.redhat.com 联系技术支持，以使用 **docs-product** 组件在 Ansible Automation Platform JIRA 项目中创建一个问题。

第 1 章 RED HAT ANSIBLE AUTOMATION PLATFORM 2.4 升级

通过设置清单并运行安装脚本，升级到 Red Hat Ansible Automation Platform 2.4。Ansible 随后将您的部署升级到 2.4。如果您计划从 Ansible Automation Platform 2.0 或更早版本升级，则必须迁移 Ansible 内容以便与 2.4 的兼容性。

1.1. ANSIBLE AUTOMATION PLATFORM 升级

从 Ansible Automation Platform 2.1 或更高版本升级到 2.4 版本涉及下载安装软件包，然后执行以下步骤：

- 设置清单以匹配您的安装环境。
- 通过当前的 Ansible Automation Platform 安装运行 2.4 安装程序。

其他资源

- 升级到 Red Hat Ansible Automation Platform 2.4

1.2. ANSIBLE AUTOMATION PLATFORM 传统升级

从 Ansible Automation Platform 2.0 或更早版本升级到 2.4 版本需要您迁移 Ansible 内容以获得兼容性。

以下步骤提供旧的升级过程概述：

- 使用 **awx-manage** 命令将自定义虚拟环境复制到自动化执行环境中。
- 通过并行升级将数据从隔离的旧节点迁移到执行节点，以便节点与最新的自动化网格功能兼容。
- 导入或生成新的自动化中心 API 令牌。
- 重新配置您的 Ansible 内容以包含 Fully Qualified Collection Names (FQCN) 以匹配 **ansible-core** 2.15。

其他资源

- 将虚拟环境迁移到自动化执行环境
- 将隔离节点迁移到执行节点
- 迁移 Ansible 内容

第 2 章 升级到 RED HAT ANSIBLE AUTOMATION PLATFORM 2.4

要升级 Red Hat Ansible Automation Platform，请先检查规划信息以确保成功升级。然后您可以下载 Ansible Automation Platform 安装程序所需的版本，在安装捆绑包中配置清单文件以反映您的环境，然后运行安装程序。

2.1. ANSIBLE AUTOMATION PLATFORM 升级计划

在开始升级过程前，请查看以下注意事项，以计划并准备 Ansible Automation Platform 部署：

自动化控制器

- 即使您拥有之前版本的有效许可证，在升级到最新版本的自动化控制器时，必须提供您的凭证或订阅清单。
- 如果您需要升级 Red Hat Enterprise Linux 和自动化控制器，您必须首先备份和恢复自动化控制器数据。
- 在升级前，集群升级需要特别注意实例和实例组。

其他资源

- 导入订阅
- 备份和恢复
- 集群

Automation hub

- 升级到 Ansible Automation Platform 2.4 时，您可以添加现有的自动化中心 API 令牌或生成新的令牌，并无效任何现有令牌。

其他资源

- 设置清单文件

Event-Driven Ansible 控制器

- 如果您当前正在运行 Event-Driven Ansible 控制器，并计划在升级到 Ansible Automation Platform 2.4 时进行部署，建议您在升级前禁用所有 Event-Driven Ansible 激活，以确保在升级过程完成后只运行新的激活。这可防止在以前的版本中运行激活的孤立容器的可能性。

2.2. 选择并获取 RED HAT ANSIBLE AUTOMATION PLATFORM 安装程序

根据您的 Red Hat Enterprise Linux 环境互联网连接，选择您需要的 Red Hat Ansible Automation Platform 安装程序。查看以下场景，并决定哪个 Red Hat Ansible Automation Platform 安装程序满足您的需要。

注意

需要有效的红帽客户帐户才能访问红帽客户门户上的 Red Hat Ansible Automation Platform 安装程序下载。

使用互联网访问进行安装

如果您的 Red Hat Enterprise Linux 环境连接到互联网，请选择 Red Hat Ansible Automation Platform 安装程序。使用互联网访问进行安装会检索最新的软件仓库、软件包和依赖项。选择以下方法之一来设置 Ansible Automation Platform 安装程序。

Tarball 安装

1. 进入 Red Hat Ansible Automation Platform 下载 页面。
2. 为 **Ansible Automation Platform <latest-version> Setup** 点 **Download Now**。
3. 解压文件：

```
$ tar xvzf ansible-automation-platform-setup-<latest-version>.tar.gz
```

RPM 安装

1. 安装 Ansible Automation Platform 安装程序软件包
   v.2.4 for RHEL 8 for x86_64

```
$ sudo dnf install --enablerepo=ansible-automation-platform-2.4-for-rhel-8-x86_64-rpms
ansible-automation-platform-installer
```

   v.2.4 for RHEL 9 for x86-64

```
$ sudo dnf install --enablerepo=ansible-automation-platform-2.4-for-rhel-9-x86_64-rpms
ansible-automation-platform-installer
```

注意

dnf install 启用存储库，因为默认禁用存储库。

使用 RPM 安装程序时，文件位于 **/opt/ansible-automation-platform/installer** 目录下。

在没有互联网访问的情况下安装

如果您无法访问互联网，或者不想从在线存储库安装独立的组件和依赖项，请使用 Red Hat Ansible Automation **PlatformBundle** 安装程序。仍然需要访问 Red Hat Enterprise Linux 软件仓库。所有其他依赖项都包含在 tar 归档中。

1. 进入 Red Hat Ansible Automation Platform 下载 页面。
2. 为 **Ansible Automation Platform <latest-version> Setup Bundle** 点 **Download Now**。
3. 解压文件：

```
$ tar xvzf ansible-automation-platform-setup-bundle-<latest-version>.tar.gz
```

2.3. 设置清单文件

在升级 Red Hat Ansible Automation Platform 安装前，请编辑 **inventory** 文件，使其与您的所需配置匹配。您可以从现有 Ansible Automation Platform 部署中保留相同的参数，或者修改参数以匹配环境的任何更改。

流程

1. 进入安装程序目录。

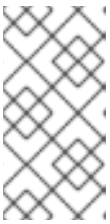
   捆绑的安装程序

   ```
   $ cd ansible-automation-platform-setup-bundle-2.4-1-x86_64
   ```

   在线安装程序

   ```
   $ cd ansible-automation-platform-setup-2.4-1
   ```

2. 打开 **inventory** 文件以进行编辑。
3. 修改 **inventory** 文件，以置备新节点、取消置备节点或组，以及导入或生成自动化中心 API 令牌。
   如果没有更改环境，您可以使用现有 Ansible Automation Platform 2.1 安装中的同一 **inventory** 文件。



注意

为 **[automationhub]** 和 **[automationcontroller]** 主机提供可访问 IP 地址或完全限定域名(FQDN)，以确保用户可以从不同节点从 Ansible 自动化中心同步和安装内容。不要使用 **localhost**。如果使用 **localhost**，则升级将作为 preflight 检查的一部分停止。

在集群中置备新节点

- 添加新节点以及 **inventory** 文件中的现有节点，如下所示：

```
[controller]
clusternode1.example.com
clusternode2.example.com
clusternode3.example.com

[all:vars]
admin_password='password'

pg_host=''
pg_port=''

pg_database='<database_name>'
pg_username='<your_username>'
pg_password='<your_password>'
```

在集群中取消置备节点或组

- 将 **node_state-deprovision** 附加到 **inventory** 文件中的节点或组。

导入和生成 API 令牌

当从 Red Hat Ansible Automation Platform 2.0 或更早版本升级到 Red Hat Ansible Automation Platform 2.1 或更高版本时，您可以使用现有的自动化中心 API 令牌或生成新的令牌。在清单文件中，在运行 Red Hat Ansible Automation Platform 安装程序设置脚本 **setup.sh** 前编辑以下字段之一：

- 使用 **automationhub_api_token** 标志导入现有的 API 令牌，如下所示：

```
automationhub_api_token=<api_token>
```

- 生成一个新的 API 令牌，并使用 **generate_automationhub_token** 标志使任何现有令牌无效，如下所示：

```
generate_automationhub_token=True
```

其他资源

- Red Hat Ansible Automation Platform 安装指南
- 取消置备单个节点或实例组

2.4. 运行 RED HAT ANSIBLE AUTOMATION PLATFORM 安装程序设置脚本

在更新了 **inventory** 文件后，您可以运行设置脚本。

流程

1. 运行 **setup.sh** 脚本

```
$ ./setup.sh
```

安装将开始。

第 3 章 迁移到自动化执行环境

3.1. 为什么需要升级到自动化执行环境？

Red Hat Ansible Automation Platform 2.4 引入了自动化执行环境。自动化执行环境是容器镜像，可以通过包括在单一容器内运行 Ansible 自动化所需的所有内容来更轻松地管理 Ansible。自动化执行环境包括：

- RHEL UBI 8
- ansible-core 2.14 或更高版本
- Python 3.9 或更高版本.
- 任何 Ansible 内容集合
- 集合 python 或二进制依赖项

通过包含这些元素，Ansible 为平台管理员提供标准化的方法来定义、构建和分发自动化运行环境。

由于新的自动化执行环境，管理员不再需要创建自定义插件和自动化内容。管理员现在只需较少的时间即可启动较小的自动化执行环境，以创建其内容。

现在，所有自定义依赖项都在开发阶段定义，而不是在管理和部署阶段定义。与 control plane 分离，可加快跨环境的开发周期、可扩展性、可靠性和可移动性。借助自动化执行环境，Ansible Automation Platform 可以移至分布式架构，允许管理员在组织内扩展自动化。

3.2. 关于将旧的 VENVS 迁移到自动化执行环境

当从旧版本的自动化控制器升级到 4.0 或更高版本时，控制器可以检测之前与机构、清单和作业模板关联的虚拟环境版本，并告知您迁移到新的自动化执行环境模型。新的自动化控制器安装在安装过程中创建两个 virtualenv；一个运行控制器，另一个运行 Ansible。与传统虚拟环境一样，自动化执行环境允许控制器在稳定的环境中运行，同时允许您根据需要在自动化执行环境中添加或更新模块，以运行 playbook。

通过将设置迁移到新的自动化执行环境，您可以将设置复制到以前的自定义虚拟环境中。使用本节中的 **awx-manage** 命令：

- 所有当前自定义虚拟环境及其路径列表（**list_custom_venvs**）
- 查看依赖特定自定义虚拟环境的资源（**custom_venv_associations**）
- 将特定的自定义虚拟环境导出到可用于迁移到自动化执行环境（**export_custom_venv**）的格式.

以下工作流描述了如何使用 **awx-manage** 命令从旧的 venvs 迁移到自动化执行环境。

3.3. 将虚拟环境迁移到自动化执行环境

升级到 Red Hat Ansible Automation Platform 2.0 和自动化控制器 4.0 后，请使用以下部分协助迁移过程中的其他步骤。

3.3.1. 列出自定义虚拟环境

您可以使用 **awx-manage** 命令列出自动化控制器实例上的虚拟环境。

流程

1. SSH 到自动化控制器实例并运行：

```
$ awx-manage list_custom_venvs
```

这时将显示已发现的虚拟环境列表。

```
# Discovered virtual environments:
/var/lib/awx/venv/testing
/var/lib/venv/new_env

To export the contents of a virtual environment, re-run while supplying the path as an argument:
awx-manage export_custom_venv /path/to/venv
```

3.3.2. 查看与自定义虚拟环境关联的对象

使用 **awx-manage** 命令，查看与自定义虚拟环境关联的组织、作业和清单源。

流程

1. SSH 到自动化控制器实例并运行：

```
$ awx-manage custom_venv_associations /path/to/venv
```

这时将显示相关对象的列表。

```
inventory_sources:
- id: 15
  name: celery
job_templates:
- id: 9
  name: Demo Job Template @ 2:40:47 PM
- id: 13
  name: elephant
organizations
- id: 3
  name: alternating_bongo_meow
- id: 1
  name: Default
projects: []
```

3.3.3. 选择要导出的自定义虚拟环境

选择您要使用 **awx-manage export_custom_venv** 命令导出的自定义虚拟环境。

流程

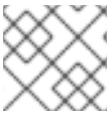
1. SSH 到自动化控制器实例并运行：

```
$ awx-manage export_custom_venv /path/to/venv
```

此命令的输出将显示在指定虚拟环境中的 **pip freeze** 状态。此信息可复制到 Ansible Builder 的 **requirements.txt** 文件中，用于创建新的自动化执行环境镜像

```
numpy==1.20.2
pandas==1.2.4
python-dateutil==2.8.1
pytz==2021.1
six==1.16.0

To list all available custom virtual environments run:
awx-manage list_custom_venvs
```



注意

在运行 **awx-manage list_custom_venvs** 时传递 **-q** 标志来减少输出。

第 4 章 将隔离节点迁移到执行节点

从版本 1.x 升级到最新版本 Red Hat Ansible Automation Platform 需要平台管理员将数据从隔离的旧节点迁移到执行节点。这个迁移是部署自动化网格所必需的。

本指南介绍了如何对外迁移进行并排迁移。这样可确保在迁移过程中不会修改原始自动化环境中的数据。

迁移过程涉及以下步骤：

1. 验证升级配置。
2. 备份原始实例。
3. 为并行升级部署新实例。
4. 使用 ansible 控制器在新实例中重新创建实例组。
5. 将原始备份恢复到新实例。
6. 设置执行节点，并将实例升级到 Red Hat Ansible Automation Platform 2.4。
7. 配置升级的控制器实例。

4.1. 升级 ANSIBLE AUTOMATION PLATFORM 的先决条件

在开始升级 Ansible Automation Platform 前，请确保您的环境满足以下要求。

4.1.1. 节点要求

Ansible Automation Platform 升级过程涉及的节点需要以下规格：

- 16 GB RAM，控制器节点、数据库节点、执行节点和跃点 (hop) 节点。
- 4 个 CPU，用于控制器节点、数据库节点、执行节点和跃点节点。
- 数据库节点有 150 GB+ 磁盘空间。
- 非数据库节点有 40 GB+ 磁盘空间。
- DHCP 保留使用无限租期来部署集群使用静态 IP 地址。
- 所有节点的 DNS 记录。
- 为所有节点安装 Red Hat Enterprise Linux 8 或更高的 64 位版本(x86)。
- 为所有节点配置 chrony。
- Python 3.9 或更高版本，用于所有内容依赖项。

4.1.2. 自动化控制器配置要求

在进行 Ansible Automation Platform 升级过程前，需要以下自动化控制器配置：

使用 Chrony 配置 NTP 服务器

集群中的每个 Ansible Automation Platform 节点都必须有权访问 NTP 服务器。使用 **chronyd** 将系统时钟与 NTP 服务器同步。这样可确保如果节点间的日期和时间没有同步，则使用需要验证的 SSL 证书的集群节点不会失败。

这对升级的 Ansible Automation Platform 集群中使用的所有节点都需要：

1. 安装 **chrony** ：

```
# dnf install chrony --assumeyes
```

2. 使用文本编辑器打开 **/etc/chrony.conf**。
3. 找到公共服务器池部分，并将其修改为包含相应的 NTP 服务器地址。只需要一个服务器，但建议使用三个服务器。添加 'iburst' 选项以加快与服务器同步所需的时间：

```
# Use public servers from the pool.ntp.org project.
# Please consider joining the pool (http://www.pool.ntp.org/join.html).
server <ntp-server-address> iburst
```

4. 在 **/etc/chrony.conf** 文件中保存更改。
5. 启动主机并启用 **chronyd** 守护进程：

```
# systemctl --now enable chronyd.service
```

6. 验证 **chronyd** 守护进程状态：

```
# systemctl status chronyd.service
```

在所有节点上附加红帽订阅

Red Hat Ansible Automation Platform 要求您将有效的订阅附加到所有节点。您可以运行以下命令来验证当前节点是否具有红帽订阅：

```
# subscription-manager list --consumed
```

如果没有附加到节点的红帽订阅，请参阅 附加 Ansible Automation Platform 订阅 以了解更多信息。

创建具有 sudo 权限的非 root 用户

在升级 Ansible Automation Platform 前，建议为部署过程创建一个具有 sudo 权限的非 root 用户。此用户用于：

- SSH 连接。
- 在安装过程中进行免密码身份验证。
- 权限升级(sudo)权限。

以下示例使用 **ansible** 命名此用户。在升级的 Ansible Automation Platform 集群的所有节点上，创建一个名为 **ansible** 的非 root 用户，并生成 SSH 密钥：

1. 创建非 root 用户：

```
# useradd ansible
```

2. 为您的用户设置密码：

```
# passwd ansible 1
Changing password for ansible.
Old Password:
New Password:
Retype New Password:
```

1 如果使用其他名称，将 **ansible** 替换为第 1 步中的非 root 用户

3. 以用户身份生成 **ssh** 密钥：

```
$ ssh-keygen -t rsa
```

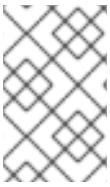
4. 进行以下操作，以使用户在使用 **sudo** 时无需输入密码：

```
# echo "ansible ALL=(ALL) NOPASSWD:ALL" | sudo tee -a /etc/sudoers.d/ansible
```

将 SSH 密钥复制到所有节点

创建 **ansible** 用户后，将 **ssh** 密钥复制到升级 Ansible Automation Platform 集群中使用的所有节点上。这样可确保 Ansible Automation Platform 安装运行时，它可以在没有密码的情况下 **ssh** 到所有节点：

```
$ ssh-copy-id ansible@node-1.example.com
```



注意

如果在云供应商中运行，您可能需要创建一个 **~/.ssh/authorized_keys** 文件，其中包含所有节点上的 **ansible** 用户的公钥，并将权限设置为 **authorized_keys** 文件，使只有所有者（**ansible**）具有读写权限（权限 600）。

配置防火墙设置

在升级的 Ansible Automation Platform 集群的所有节点上配置防火墙设置，以允许访问成功 Ansible Automation Platform 升级的适当服务和端口。对于 Red Hat Enterprise Linux 8 或更高版本，启用 **firewalld** 守护进程以启用所有节点所需的访问权限：

1. 安装 **firewalld** 软件包：

```
# dnf install firewalld --assumeyes
```

2. 启动 **firewalld** 服务：

```
# systemctl start firewalld
```

3. 启用 **firewalld** 服务：

```
# systemctl enable --now firewalld
```

4.1.3. Ansible Automation Platform 配置要求

在进行 Ansible Automation Platform 升级过程前，需要以下 Ansible Automation Platform 配置：

配置执行和跃点节点的防火墙设置

升级 Red Hat Ansible Automation Platform 实例后，在网格节点上添加自动化网格端口（execution 和 hop 节点）以启用自动化网格功能。用于所有节点上的网格网络的默认端口为 **27199/tcp**。您可以将网格网络配置为使用不同的端口，方法是将 **recptor_listener_port** 指定为清单文件中每个节点的变量。

在跃点和执行节点中，设置用于安装的 **firewalld** 端口。

1. 确保 **firewalld** 正在运行：

```
$ sudo systemctl status firewalld
```

2. 将 **firewalld** 端口添加到控制器数据库节点（例如，端口 27199）：

```
$ sudo firewall-cmd --permanent --zone=public --add-port=27199/tcp
```

3. 重新加载 **firewalld** ：

```
$ sudo firewall-cmd --reload
```

4. 确认端口已打开：

```
$ sudo firewall-cmd --list-ports
```

4.2. 备份 ANSIBLE AUTOMATION PLATFORM 实例

使用 **backup_dir** 标志运行 **.setup.sh** 脚本备份现有 Ansible Automation Platform 实例，它会保存当前环境的内容和配置：

1. 进入 **ansible-tower-setup-latest** 目录。
2. 按照以下示例运行 **./setup.sh** 脚本：

```
$ ./setup.sh -e ‘backup_dir=/ansible/mybackup’ -e ‘use_compression=True’ @credentials.yml -b (1) (2)
```

(1) **backup_dir** 指定将备份保存到的目录。

(2) **@credentials.yml** 传递密码变量及其值通过 **ansible-vault** 加密。

通过成功备份，会在 **/ansible/mybackup/tower-backup-latest.tar.gz** 处创建一个备份文件。

之后，需要进行这个备份，将内容从您的旧实例迁移到新实例。

4.3. 为并行（SIDE-BY-SIDE）升级部署新实例

要进行并行升级过程，请部署第二个 Ansible Tower 3.8.x 实例，并使用相同的实例组配置。此新实例将收到来自您原始实例的内容和配置，之后再升级到 Red Hat Ansible Automation Platform 2.4。

4.3.1. 部署 Ansible Tower 的新实例

要部署新的 Ansible Tower 实例，请执行以下操作：

1. 进入 Ansible Tower 安装程序页面，下载与您的原始 Tower 实例匹配的 Tower 安装程序版本。
2. 进入安装程序，使用一个文本编辑器打开 **inventory** 文件，为 Tower 安装配置 **inventory** 文件：
   a. 除了 Tower 配置外，删除包含 **isolated_group** 或 **instance_group** 的任何字段。

注意

有关使用 Ansible Automation Platform 安装程序安装 Tower 的更多信息，请参阅 Ansible Automation Platform 安装指南。

3. 运行 **setup.sh** 脚本来开始安装。

安装了新实例后，配置 Tower 设置以匹配来自原始 Tower 实例的实例组。

4.3.2. 在新实例中重新创建实例组

要在新实例中重新创建您的实例组，请执行以下操作：

注意

记录来自您原始 Tower 实例的所有实例组。您需要在新实例中重新创建这些组。

1. 登录到 Tower 的新实例。
2. 在导航窗格中，选择 **Administration → Instance Groups**。
3. 点 **Create instance group**。
4. 输入与原始实例的实例组匹配的 **Name**，然后点 **Save**
5. 重复上述实例中的所有实例组。

4.4. 将备份恢复到新实例

使用 **restore_backup_file** 标志运行 **./setup.sh** 脚本，将内容从原始 1.x 实例的备份文件迁移到新实例。这会有效地迁移所有作业历史记录、模板和其他 Ansible Automation Platform 相关内容。

流程

1. 运行以下命令:

```
$ ./setup.sh -r -e 'restore_backup_file=/ansible/mybackup/tower-backup-latest.tar.gz' -e
'use_compression=True' -e @credentials.yml -r -- --ask-vault-pass 1 2 3
```

1. **restore_backup_file** 指定 Ansible Automation Platform 备份数据库的位置
2. **use_compression** 设置为 **True**，因为备份过程中使用要使用压缩功能
3. **-r** 将 restore 数据库选项设置为 **True**

2. 登录到新的 RHEL 8 Tower 3.8 实例，验证来自您原始实例的内容是否已恢复：

   a. 进入 **Administration → Instance Groups**。重新创建的实例组现在应包含来自原始实例的 **Total Jobs**。

   b. 使用侧导航面板，检查您的内容是否已从您的原始实例导入，包括作业、模板、清单、凭证和用户。

现在，您有一个新的 Ansible Tower 实例，其中包含来自您原始实例的所有 Ansible 内容。

您将把这个新实例升级到 Ansible Automation Platform 2.4，以便保留所有之前的数据，而无需覆盖原始实例。

4.5. 升级到 ANSIBLE AUTOMATION PLATFORM 2.4

要将 Ansible Tower 实例升级到 Ansible Automation Platform 2.4，请将您的原始 Tower 实例中的 **inventory** 文件复制到您的新 Tower 实例，并运行安装程序。Red Hat Ansible Automation Platform 安装程序检测到 pre-2.4，并提供升级的清单文件以继续升级过程：

1. 从 Red Hat Ansible Automation Platform 下载 页面， 下载 Red Hat Ansible Automation Platform 的最新安装程序。

2. 解压文件：

```
$ tar xvzf ansible-automation-platform-setup-<latest_version>.tar.gz
```

3. 进入 Ansible Automation Platform 安装目录：

```
$ cd ansible-automation-platform-setup-<latest_version>/
```

4. 将原始实例中的 **inventory** 文件复制到最新安装程序的目录中：

```
$ cp ansible-tower-setup-3.8.x.x/inventory ansible-automation-platform-
setup-<latest_version>
```

5. 运行 **setup.sh** 脚本：

```
$ ./setup.sh
```

   设置脚本将暂停并表明检测到了"pre-2.x"清单文件，但提供了名为 **inventory.new.ini** 的新文件，允许您继续升级您的原始实例。

6. 编辑使用文本编辑器打开 **inventory.new.ini**。

注意

通过运行设置脚本，安装程序修改了原始清单文件中的几个字段，如将 [tower] 重命名为 [automationcontroller]。

7. 通过分配相关变量、节点和相关节点对等连接来更新新生成的 **inventory.new.ini** 文件来配置自动化网格：

注意

自动化网格拓扑的设计取决于您的环境自动化需求。提供所有可能场景的设计超出了本文档的范围。以下是一个自动化网格设计示例。

包含三个使用跃点节点的标准 control plane 的清单文件示例：

```
[automationcontroller]
control-plane-1.example.com
control-plane-2.example.com
control-plane-3.example.com

[automationcontroller:vars]
node_type=control (1)
peers=execution_nodes (2)


[execution_nodes]
execution-node-1.example.com peers=execution-node-2.example.com
execution-node-2.example.com peers=execution-node-3.example.com
execution-node-3.example.com peers=execution-node-4.example.com
execution-node-4.example.com peers=execution-node-5.example.com node_type=hop
execution-node-5.example.com peers=execution-node-6.example.com node_type=hop (3)
execution-node-6.example.com peers=execution-node-7.example.com
execution-node-7.example.com

[execution_nodes:vars]
node_type=execution
```

(1) 指定运行项目和清单更新和系统作业的控制节点，但不运行常规作业。这些节点上禁用了执行功能。

(2) 在 **[execution_nodes]** 组中为节点连接指定对等关系。

(3) 指定将流量路由到其他执行节点的跃点节点。hop 节点无法执行自动化。

8. 导入或生成自动化中心 API 令牌。
   - 使用 **automationhub_api_token** 标志导入现有的 API 令牌：

     ```
     automationhub_api_token=<api_token>
     ```

   - 生成一个新的 API 令牌，并通过将 **generate_automationhub_token** 标志设置为 **True** 来使任何现有令牌无效：

     ```
     generate_automationhub_token=True
     ```

9. 为自动化网格配置 **inventory.new.ini** 后，使用 **inventory.new.ini** 运行设置脚本：

   ```
   $ ./setup.sh -i inventory.new.ini -e @credentials.yml -- --ask-vault-pass
   ```

10. 安装完成后，通过在所有自动化控制器节点中登录 Ansible Automation Platform 控制面板 UI 来验证 Ansible Automation Platform 是否已成功安装。

其他资源

- 有关使用 Ansible Automation Platform 安装程序的常规信息，请参阅 Red Hat Ansible Automation Platform 安装指南。

4.6. 配置升级的 ANSIBLE AUTOMATION PLATFORM

4.6.1. 配置自动化控制器实例组

升级 Red Hat Ansible Automation Platform 实例后，通过在自动化控制器 UI 中配置设置将原始实例关联到对应的实例组：

1. 登录新的 Controller 实例。
2. 来自旧实例的内容，如凭证、作业和清单，现在在控制器实例上可见。
3. 进入 **Administration → Instance Groups**。
4. 点实例组来关联执行节点，然后点**实例**选项卡。
5. 点**关联**。选择要关联此实例组的节点，然后点 **Save**。
6. 您还可以修改默认实例来解除新的执行节点。

第 5 章 ANSIBLE 内容迁移

如果您要从 **ansible-core** 版本迁移到 **ansible-core** 2.13，请考虑检查 Ansible 核心移植指南，以熟悉每个版本之间的更改和更新。在查看 Ansible 核心端口指南时，请确保选择最新版本的 **ansible-core** 或 **devel**，它位于指南的左上方。

有关完全支持和经认证的 Ansible 内容集合列表，请参阅 console.redhat.com 上的 Ansible Automation hub。

5.1. 安装 ANSIBLE 集合

作为从较早 Ansible 版本迁移到更新的版本的一部分，您需要查找并下载包含您使用的模块的集合。找到该集合列表后，您可以使用以下选项之一在本地包含集合：

1. 使用 **ansible-builder** 将集合下载并安装到您的运行时或执行环境。
2. 更新 Automation Controller 项目中的 'requirements.yml' 文件安装角色和集合。这样，每次同步 Automation Controller 中的项目时，将下载角色和集合。

注意

在很多情况下，上游和下游集合可能相同，但始终从 Automation Hub 下载您的认证集合。

5.2. 将 ANSIBLE PLAYBOOK 和角色迁移到 CORE 2.13

当您从基于集合的内容迁移到基于集合的内容时，您应该在 playbook 和角色中使用 Fully Qualified Collection Names(FQCN)以避免意外行为。

带有 FQCN 的 playbook 示例：

```
- name: get some info
  amazon.aws.ec2_vpc_net_info:
    region: "{{ec2_region}}"
  register: all_the_info
  delegate_to: localhost
  run_once: true
```

如果您使用 ansible-core 模块，且没有从不同的集合调用模块，您应该使用 FQCN **ansible.builtin.copy**。

带有 FQCN 的模块示例：

```
- name: copy file with owner and permissions
  ansible.builtin.copy:
  src: /srv/myfiles/foo.conf
  dest: /etc/foo.conf
  owner: foo
  group: foo
  mode: '0644'
```

5.3. 转换 PLAYBOOK 示例

例子

这个示例是名为 **/mydata** 的共享目录，我们希望可以在作业运行期间读取和写入文件。请记住，必须在执行节点上已存在，我们将用于自动化运行。

您将以 *aape1.local* 执行节点为目标来运行此作业，因为底层主机已经有这个问题。

```
[awx@aape1 ~]$ ls -la /mydata/
total 4
drwxr-xr-x.  2 awx  awx   41 Apr 28 09:27 .
dr-xr-xr-x. 19 root root 258 Apr 11 15:16 ..
-rw-r--r--.  1 awx  awx   33 Apr 11 12:34 file_read
-rw-r--r--.  1 awx  awx    0 Apr 28 09:27 file_write
```

您将使用简单的 playbook 启动自动化，并定义 sleep 来供您访问，并了解进程，并演示读取和写入文件。

```
# vim:ft=ansible:
```

```
- hosts: all
  gather_facts: false
  ignore_errors: yes
  vars:
    period: 120
    myfile: /mydata/file
  tasks:
    - name: Collect only selected facts
      ansible.builtin.setup:
        filter:
          - 'ansible_distribution'
          - 'ansible_machine_id'
          - 'ansible_memtotal_mb'
          - 'ansible_memfree_mb'
    - name: "I'm feeling real sleepy..."
      ansible.builtin.wait_for:
        timeout: "{{ period }}"
      delegate_to: localhost
    - ansible.builtin.debug:
        msg: "Isolated paths mounted into execution node: {{ AWX_ISOLATIONS_PATHS }}"
    - name: "Read pre-existing file..."
      ansible.builtin.debug:
        msg: "{{ lookup('file', '{{ myfile }}_read'
    - name: "Write to a new file..."
      ansible.builtin.copy:
        dest: "{{ myfile }}_write"
        content: |
          This is the file I've just written to.

    - name: "Read written out file..."
      ansible.builtin.debug:
        msg: "{{ lookup('file', '{{ myfile }}_write') }}"
```

在 Ansible Automation Platform 2 导航面板中，选择 **Settings**。然后从 **Jobs** 选项中选择 **Job settings**。

公开隔离任务的路径：

```
[
"/mydata:/mydata:rw"
]
```

卷挂载在容器中使用相同的名称进行映射，并具有读写功能。这将在启动作业模板时使用。

prompt on launch 应该为 *extra_vars* 设置，以便您可以调整每次运行的 sleep 持续时间，默认为 30 秒。

启动后，会调用 *wait_for* 模块作为睡眠状态，您可以进入执行节点并查看正在运行的内容。

要验证运行是否已成功完成，请运行以下命令以获得作业的输出：

```
$ podman exec -it 'podman ps -q' /bin/bash
bash-4.4#
```

您现在位于运行的执行环境容器中。

查看权限，您会看到 **awx** 已变为 'root'，但这并不像在超级用户中一样真正是 root 用户，因为您使用无根 Podman，这会将用户映射到与沙盒类似的内核命名空间中。了解更多有关 rootless Podman 工作方式的信息，适用于 shadow-utils。

```
bash-4.4# ls -la /mydata/
Total 4
drwxr-xr-x. 2 root root 41 Apr 28 09:27 .
dr-xr-xr-x. 1 root root 77 Apr 28 09:40 ..
-rw-r—--r–. 1 root root 33 Apr 11 12:34 file_read
-rw-r—--r–. 1 root root  0 Apr 28 09:27 file_write
```

根据结果，此作业会失败。为了了解原因，需要检查剩余的输出。

```
TASK [Read pre-existing file…]****************************** 10:50:12
ok: [localhost] =>  {
    “Msg”: “This is the file I am reading in.”

TASK {Write to a new file...}********************************** 10:50:12
An exception occurred during task execution. To see the full traceback, use -vvv. The error was:
PermissionError: [Errno 13] Permission denied: b'/mydata/.ansible_tmpazyqyqdrfile_write' -> b'
/mydata/file_write'
Fatal: [localhost]: FAILED! => {"changed": false, :checksum":
"9f576o85d584287a3516ee8b3385cc6f69bf9ce", "msg": "Unable to make
b'/root/.ansible/tmp/anisible-tim-1651139412.9808054-40-91081834383738/source' into
/mydata/file_write, failed final rename from b'/mydata/.ansible_tmpazyqyqdrfile_write': [Errno 13]
Permission denied: b'/mydata/.ansible_tmpazyqyqdrfile_write' -> b'/mydata/file_write}
...ignoring

TASK [Read written out file...] ****************************** 10:50:13
Fatal: [localhost]: FAILED: => {"msg": "An unhandled exception occurred while running the lookup
plugin 'file'. Error was a <class 'ansible.errors.AnsibleError;>, original message: could not locate file in
lookup: /mydate/file_write. Vould not locate file in lookup: /mydate/file_write"}
...ignoring
```

作业失败，即使设置了 **:rw**，因此它应具有写入功能。进程能够读取现有的文件，但不能写出。这是因为 SELinux 保护要求将正确的标签放在挂载到容器中的卷内容上。如果缺少标签，SELinux 可能会阻止进程在容器内运行。Podman 不会更改操作系统设置的标签。如需更多信息，请参阅 Podman 文档。

这可以是常见的错误解释器。我们已将默认值设置为 **:z**，它会告知 Podman 在共享卷上重新标记文件对象。

因此，我们可以添加 **:z** 或将其关闭。

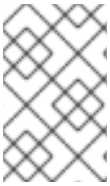
公开隔离任务的路径：

```
[
  "/mydata:/mydata"
]
```

playbook 现在可以正常工作：

```
PLAY [all] ********************************************************* 11:05:52
TASK [I'm feeling real sleepy. . .] ************************** 11:05:52
ok: [localhost]
TASK [Read pre-existing file...] ***************************** 11:05:57
ok: [localhost] => {
 "Msg": "This is the file I'm reading in."
}
TASK [Write to a new file…] ********************************** 11:05:57
ok: [localhost]
TASK [Read written out file…] ******************************** 11:05:58
ok: [localhost] => {
      “Msg”: “This is the file I’ve just written to.”
```

返回到底层执行节点主机，我们有新写入的内容。



注意

如果您使用容器组在 Red Hat OpenShift 中启动自动化作业，您也可以告诉 Ansible Automation Platform 2 向那个环境公开相同的路径，但您必须在设置下将默认值切换为 **On**。

启用后，这将将其注入用于执行的 pod 规格中的 *volumeMounts* 和 *卷*。它类似如下：

```
apiVersion: v1
kind: Pod
Spec:
   containers:
   - image: registry.redhat.io/ansible-automation-platform-24/ee-minimal-rhel8
  args:
    - ansible runner
    - worker
    - –private-data-dir=/runner
  volumeMounts:
mountPath: /mnt2
name: volume-0
readOnly: true
mountPath: /mnt3
name: volume-1
```

```
readOnly: true
mountPath: /mnt4
name: volume-2
readOnly: true
volumes:
hostPath:
  path: /mnt2
  type: “”
name: volume-0
hostPath:
  path: /mnt3
  type: “”
name: volume-1
hostPath:
  path: /mnt4
  type: “”
name: volume-2
```

运行的容器内的存储使用覆盖文件系统。运行的容器中的任何修改都会在作业完成后销毁，就像被卸载的 tmpfs 一样。

第 6 章 为 AAP2 转换 PLAYBOOK

使用 Ansible Automation Platform 2 及其容器化执行环境时，*localhost* 的使用已被修改。在以前的 Ansible Automation Platform 版本中，作业将针对 *localhost* 运行，它转换为底层 Automation Controller 主机上运行。这可用于存储数据和持久工件。

使用 Ansible Automation Platform 2 时，localhost 意味着您在容器内运行，这是临时的。*localhost* 不再与特定主机关联，使用可移植的执行环境，这意味着它可以在已嵌入到执行环境容器中的正确环境和软件先决条件的任何位置运行。

6.1. 从自动运行保留数据

将本地自动化控制器文件系统视为计数器产品，因为这会将数据绑定到该主机。如果您有多节点集群，那么您可以在每次联系不同的主机，如果创建依赖于其他和创建的目录的工作流，则会导致问题。例如，如果在另一个节点运行 playbook 时只在一个节点中创建目录，则结果将不一致。

该解决方案是使用某种形式的共享存储解决方案，如 Amazon S3、Gist 或您的数据端点的 rsync 数据的角色。

选项存在在运行时将数据或配置注入容器中。这可以通过使用自动化控制器的隔离作业路径选项来实现。

这提供了在运行时将目录和文件挂载到执行环境的方法。这可以通过自动化网格来实现，使用 ansible-runner 将它们注入 Podman 容器以启动自动化。以下是使用隔离作业路径的一些用例：

- 在运行时提供 SSL 证书，而不是将它们纳入执行环境。
- 传递运行时配置数据，如 SSH 配置设置，但可能是您要在自动化期间使用的任何数据。
- 在自动化运行期间和之后读取和写入之前使用的文件。

使用率有一些注意事项：

- 卷挂载必须在所有能够执行自动化的节点上(hybrid control plane 节点和所有执行节点) 预先存在。
- 如果启用 SELinux (Ansible Automation Platform 默认)是文件权限的。
    - 这很重要，因为无根 Podman 在基于非 OCP 安装上运行。

需要仔细观察到注意事项。强烈建议您对无根 Podman 和 Podman 卷挂载选项、隔离作业路径的 [:OPTIONS] 部分进行读取，因为这是 Ansible Automation Platform 2 中使用的内容。

其他资源

- 了解无根 Podman.
- podman 卷挂载运行时选项.

6.1.1. 转换 playbook 示例

例子

这个示例是名为 **/mydata** 的共享目录，我们希望可以在作业运行期间读取和写入文件。请记住，必须在执行节点上已存在，我们将用于自动化运行。

您将以 *aape1.local* 执行节点为目标来运行此作业，因为底层主机已经有这个问题。

```
[awx@aape1 ~]$ ls -la /mydata/
total 4
drwxr-xr-x.  2 awx  awx   41 Apr 28 09:27 .
dr-xr-xr-x. 19 root root 258 Apr 11 15:16 ..
-rw-r--r--.  1 awx  awx   33 Apr 11 12:34 file_read
-rw-r--r--.  1 awx  awx    0 Apr 28 09:27 file_write
```

您将使用简单的 playbook 启动自动化，并定义 sleep 来供您访问，并了解进程，并演示读取和写入文件。

```
# vim:ft=ansible:
```

```
- hosts: all
  gather_facts: false
  ignore_errors: yes
  vars:
    period: 120
    myfile: /mydata/file
  tasks:
    - name: Collect only selected facts
      ansible.builtin.setup:
        filter:
          - 'ansible_distribution'
          - 'ansible_machine_id'
          - 'ansible_memtotal_mb'
          - 'ansible_memfree_mb'
    - name: "I'm feeling real sleepy..."
      ansible.builtin.wait_for:
        timeout: "{{ period }}"
      delegate_to: localhost
    - ansible.builtin.debug:
        msg: "Isolated paths mounted into execution node: {{ AWX_ISOLATIONS_PATHS }}"
    - name: "Read pre-existing file..."
      ansible.builtin.debug:
        msg: "{{ lookup('file', '{{ myfile }}_read'
    - name: "Write to a new file..."
      ansible.builtin.copy:
        dest: "{{ myfile }}_write"
        content: |
          This is the file I've just written to.

    - name: "Read written out file..."
      ansible.builtin.debug:
        msg: "{{ lookup('file', '{{ myfile }}_write') }}"
```

在 Ansible Automation Platform 2 导航面板中，选择 **Settings**。然后从 **Jobs** 选项中选择 **Job settings**。

公开隔离任务的路径：

```
[
"/mydata:/mydata:rw"
]
```

- 卷挂载在容器中使用相同的名称进行映射，并具有读写功能。这将在启动作业模板时使用。

prompt on launch 应该为 *extra_vars* 设置，以便您可以调整每次运行的 sleep 持续时间，默认为 30 秒。

启动后，会调用 *wait_for* 模块作为睡眠状态，您可以进入执行节点并查看正在运行的内容。

要验证运行是否已成功完成，请运行以下命令以获得作业的输出：

```
$ podman exec -it 'podman ps -q' /bin/bash
bash-4.4#
```

您现在位于运行的执行环境容器中。

查看权限，您会看到 **awx** 已变为 'root'，但这并不像在超级用户中一样真正是 root 用户，因为您使用无根 Podman，这会将用户映射到与沙盒类似的内核命名空间中。了解更多有关 rootless Podman 工作方式的信息，适用于 shadow-utils。

```
bash-4.4# ls -la /mydata/
Total 4
drwxr-xr-x. 2 root root 41 Apr 28 09:27 .
dr-xr-xr-x. 1 root root 77 Apr 28 09:40 ..
-rw-r—--r–. 1 root root 33 Apr 11 12:34 file_read
-rw-r—--r–. 1 root root  0 Apr 28 09:27 file_write
```

根据结果，此作业会失败。为了了解原因，需要检查剩余的输出。

```
TASK [Read pre-existing file…]****************************** 10:50:12
ok: [localhost] => {
    “Msg”: “This is the file I am reading in.”

TASK {Write to a new file...}******************************** 10:50:12
An exception occurred during task execution. To see the full traceback, use -vvv. The error was:
PermissionError: [Errno 13] Permission denied: b'/mydata/.ansible_tmpazyqyqdrfile_write' -> b'
/mydata/file_write'
Fatal: [localhost]: FAILED! => {"changed": false, :checksum":
"9f576o85d584287a3516ee8b3385cc6f69bf9ce", "msg": "Unable to make
b'/root/.ansible/tmp/anisible-tim-1651139412.9808054-40-91081834383738/source' into
/mydata/file_write, failed final rename from b'/mydata/.ansible_tmpazyqyqdrfile_write': [Errno 13]
Permission denied: b'/mydata/.ansible_tmpazyqyqdrfile_write' -> b'/mydata/file_write}
...ignoring

TASK [Read written out file...] ****************************** 10:50:13
Fatal: [localhost]: FAILED: => {"msg": "An unhandled exception occurred while running the lookup
plugin 'file'. Error was a <class 'ansible.errors.AnsibleError;>, original message: could not locate file in
lookup: /mydate/file_write. Vould not locate file in lookup: /mydate/file_write"}
...ignoring
```

作业失败，即使设置了 **:rw**，因此它应具有写入功能。进程能够读取现有的文件，但不能写出。这是因为 SELinux 保护要求将正确的标签放在挂载到容器中的卷内容上。如果缺少标签，SELinux 可能会阻止进程在容器内运行。Podman 不会更改操作系统设置的标签。如需更多信息，请参阅 Podman 文档。

这可以是常见的错误解释器。我们已将默认值设置为 **:z**，它会告知 Podman 在共享卷上重新标记文件对象。

因此，我们可以添加 **:z** 或将其关闭。

公开隔离任务的路径：

```
[
   "/mydata:/mydata"
]
```

playbook 现在可以正常工作：

```
PLAY [all] ********************************************************* 11:05:52
TASK [I'm feeling real sleepy. . .] ************************** 11:05:52
ok: [localhost]
TASK [Read pre-existing file...] ****************************** 11:05:57
ok: [localhost] => {
 "Msg": "This is the file I'm reading in."
}
TASK [Write to a new file…] ********************************** 11:05:57
ok: [localhost]
TASK [Read written out file…] ******************************** 11:05:58
ok: [localhost] => {
      “Msg”: “This is the file I’ve just written to.”
```

返回到底层执行节点主机，我们有新写入的内容。

注意

如果您使用容器组在 Red Hat OpenShift 中启动自动化作业，您也可以告诉 Ansible Automation Platform 2 向那个环境公开相同的路径，但您必须在设置下将默认值切换为 **On**。

启用后，这将将其注入用于执行的 pod 规格中的 *volumeMounts* 和 *卷*。它类似如下：

```
apiVersion: v1
kind: Pod
Spec:
   containers:
   - image: registry.redhat.io/ansible-automation-platform-24/ee-minimal-rhel8
  args:
    - ansible runner
    - worker
    - –private-data-dir=/runner
  volumeMounts:
mountPath: /mnt2
name: volume-0
readOnly: true
mountPath: /mnt3
name: volume-1
readOnly: true
mountPath: /mnt4
name: volume-2
readOnly: true
volumes:
hostPath:
```

```
  path: /mnt2
  type: “”
name: volume-0
hostPath:
  path: /mnt3
  type: “”
name: volume-1
hostPath:
  path: /mnt4
  type: “”
name: volume-2
```

运行的容器内的存储使用覆盖文件系统。运行的容器中的任何修改都会在作业完成后销毁，就像被卸载的 tmpfs 一样。