Red Hat Ansible Automation Platform 2.4

Red Hat Ansible Automation Platform 安装指南

安装 Ansible Automation Platform

Last Updated: 2024-07-03

Red Hat Ansible Automation Platform 2.4 Red Hat Ansible Automation Platform 安装指南

安装 Ansible Automation Platform

法律通告

摘要

本指南演示了如何根据支持的安装场景安装 Red Hat Ansible Automation Platform。

目录

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。

本指南帮助您了解安装 Ansible Automation Platform 后的安装要求和流程。本文档已更新，以包含 Ansible Automation Platform 最新版本的信息。

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 安装概述

Red Hat Ansible Automation Platform 安装程序为您提供了灵活性，允许您使用多个支持的安装场景安装 Ansible Automation Platform。从 Ansible Automation Platform 2.4 开始，安装场景包括了可选的 Event-Driven Ansible 控制器部署，它引进了对 IT 请求的自动解析。

无论您选择什么安装方式，安装 Ansible Automation Platform 都涉及以下步骤：

编辑 Red Hat Ansible Automation Platform 安装程序清单文件

Ansible Automation Platform 安装程序清单文件允许您指定安装方式，并描述 Ansible 的主机部署。本文档中提供的示例显示了为您的部署安装该场景所需的参数规格。

运行 Red Hat Ansible Automation Platform 安装程序设置脚本

设置脚本使用清单文件中定义的所需参数安装您的私有自动化中心。

验证自动化控制器安装

安装 Ansible Automation Platform 后，您可以通过登录到自动化控制器来验证安装是否成功。

验证自动化中心安装

安装 Ansible Automation Platform 后，您可以通过登录到自动化中心来验证安装是否成功。

验证 Event-Driven Ansible 控制器安装

安装 Ansible Automation Platform 后，您可以通过登录到 Event-Driven Ansible 控制器来验证安装是否成功。

其他资源

有关支持的安装场景的更多信息，请参阅 Red Hat Ansible Automation Platform 规划指南 。

1.1. 先决条件

- 从 Red Hat Ansible Automation Platform 产品软件 中选择并获取了平台安装程序。
- 您需要在满足基本系统要求的机器上安装。
- 您已将所有软件包更新至 RHEL 节点的当前版本。

警告

要防止错误，请在安装 Ansible Automation Platform 前完全升级 RHEL 节点。

- 已使用创建 Registry 服务帐户 中的说明创建了 Red Hat Registry Service Account。

其他资源

有关获取平台安装程序或系统要求的更多信息，请参阅 Red Hat Ansible Automation Platform *规划指南中的 Red Hat Ansible Automation Platform 系统要求*。

第 2 章 系统要求

在规划 Red Hat Ansible Automation Platform 安装并设计适合您的用例的自动化网格拓扑时，请使用此信息。

先决条件

- 您可以通过 **sudo** 命令或特权升级来获取 root 访问权限。如需有关特权升级的更多信息，请参阅了解特权升级。
- 您可以将特权从 root 降级到用户，例如： AWX、PostgreSQL、Event-Driven Ansible 或 Pulp。
- 您已在所有节点上配置了 NTP 客户端。如需更多信息，请参阅使用 Chrony 配置 NTP 服务器。

2.1. RED HAT ANSIBLE AUTOMATION PLATFORM 系统要求

您的系统必须满足以下最低系统要求才能安装和运行 Red Hat Ansible Automation Platform。

表 2.1. 基本系统

要求	必填	备注
Subscription	有效的 Red Hat Ansible Automation Platform	
OS	Red Hat Enterprise Linux 8.6 或更高的 64 位版本 (x86, ppc64le, s390x, aarch64)	OpenShift 也支持 Red Hat Ansible Automation Platform，请参阅在 OpenShift Container Platform 上部署 Red Hat Ansible Automation Platform Operator。
ansible-core	ansible-core 版本 2.14 或更高版本	Ansible Automation Platform 包括包含 ansible-core 2.15 的执行环境。
Python	3.9 或更高版本	
浏览器	当前支持的 Mozilla FireFox 或 Google Chrome 版本	
数据库	PostgreSQL 版本 13	

在使用项目更新和集合时，需要满足以下条件：

- 确保 *Table 5.9 中列出的 网络端口和协议。Automation Hub* 可用于成功连接并从自动化中心或 Ansible Galaxy 服务器下载集合。
- 在使用自签名证书或红帽域时禁用 SSL 检查。

注意

由 Ansible Automation Platform 管理的系统的要求与 Ansible 相同。请参阅 Ansible 社区文档中的安装 Ansible。

有关 Red Hat Ansible Automation Platform 要求的额外备注

- Red Hat Ansible Automation Platform 依赖于 Ansible Playbook，并且需要安装最新版本的 ansible-core。您可以手动下载 ansible-core，也可以作为 Red Hat Ansible Automation Platform 安装的一部分自动下载它。
- 对于新安装，自动化控制器会安装最新版本的 ansible-core。
- 如果执行捆绑的 Ansible Automation Platform 安装，安装 setup.sh 脚本会尝试从捆绑包中安装 ansible-core （及其依赖项）。
- 如果您手动安装 Ansible，Ansible Automation Platform 安装 setup.sh 脚本将检测到已安装了 Ansible，且不会尝试重新安装它。

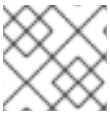
注意

您必须使用软件包管理器（如 **dnf** ）安装 Ansible，必须安装软件包管理器的最新稳定版本，以便 Red Hat Ansible Automation Platform 正常工作。版本 2.4 及更新的版本需要 Ansible 2.14 版本。

2.2. 自动化控制器系统要求

自动化控制器是一种分布式系统，不同的软件组件可以并置或部署到多个不同计算节点上。在安装程序中，提供四个节点类型作为抽象层，可帮助您设计适合您的用例的拓扑：控制、混合、执行和跃点节点。

对节点大小使用以下建议：



注意

在控制和混合节点上，为执行环境存储分配至少 20 GB 的 **/var/lib/awx**。

执行节点

执行节点运行自动化。增加内存和 CPU 以增加容量来运行更多分叉。

注意

- 执行节点上安装的软件包可能不需要 RAM 和 CPU 资源，但最低建议处理节点的作业负载，以便同时运行平均作业数量。
- 不提供推荐的 RAM 和 CPU 节点大小。所需的 RAM 或 CPU 取决于您在该环境中运行的作业数量。

有关所需 RAM 和 CPU 级别的更多信息，请参阅自动化控制器的性能调整。

表 2.2. 执行节点

要求	最低要求
RAM	16 GB
CPU	4
本地磁盘	最小 40GB

控制节点

控制节点处理事件并运行集群作业，包括项目更新和清理作业。增加 CPU 和内存有助于处理作业事件。

表 2.3. 控制节点

要求	最低要求
RAM	16 GB
CPU	4
本地磁盘	• 至少 40GB，在 /var/lib/awx 下至少有 20GB • 存储卷的最低基础线必须被定做 1500 IOPS • 项目存储在控制和混合节点上，作业持续时间也存储在执行节点上。如果集群有很多大型项目，请考虑在 /var/lib/awx/projects 中调整 GB，以避免磁盘空间错误。

hop 节点

hop 节点用于将流量从自动化网格的一个部分路由到另一个网络（例如，hop 节点可以是堡垒主机到另一网络）。RAM 可能会影响吞吐量，CPU 活动较低。网络带宽和延迟通常比 RAM 或 CPU 更重要。

表 2.4. 跃点（hop）节点

要求	最低要求
RAM	16 GB
CPU	4
本地磁盘	40 GB

- 实际 RAM 的要求取决于同时管理的主机自动化控制器数量（这由作业模板或系统 **ansible.cfg** 文件中的 **forks** 参数控制）。为避免可能的资源冲突，Ansible 建议每 10 个 fork 和 2 GB 保留内存用于自动化控制器。如需更多信息，请参阅自动控制器容量确定和作业影响。如果 **fork** 设为 400，则建议使用 42 GB 内存。

- 自动化控制器主机检查 **umask** 是否已设置为 0022。如果没有，则设置会失败。设置 **umask=0022** 以避免出现这个错误。
- 可以处理更多主机，但如果 fork 数量小于主机总数，则需要在主机间通过更多。您可以使用以下方法之一避免这些 RAM 限制：
  - 使用滚动更新。
  - 使用内置在自动化控制器中的置备回调系统，其中每个请求配置的系统都会进入队列，并尽快处理。
  - 如果自动化控制器正在生成或部署镜像，如 AMI。

其他资源

- 有关获取自动化控制器订阅的更多信息，请参阅 导入订阅。
- 如有疑问，请通过红帽客户门户网站联系 Ansible 支持。

2.3. AUTOMATION HUB 系统要求

通过自动化中心，您可以从 Red Hat Ansible 和认证合作伙伴发现并使用新的认证自动化内容。在 Ansible Automation Hub 上，您可以发现和管理由红帽及其合作伙伴开发的自动化内容的 Ansible 集合，用于云自动化、网络自动化和安全自动化等用例。

自动化中心有以下系统要求：

要求	必填	备注
RAM	最小 8 GB	• 8 GB RAM（Vagrant trial 版本安装的最小和推荐值） • 8 GB RAM（外部独立 PostgreSQL 数据库最小值） • 有关配置中基于 fork 的容量，请参阅 自动化控制器容量确定和作业影响。
CPU	最少 2 个	有关配置中基于 fork 的容量，请参阅 自动化控制器容量确定和作业影响。
本地磁盘	60 GB 磁盘	至少指定 40GB 到 **/var** 进行集合存储。

注意

私有自动化中心

如果您从内部地址安装私有自动化中心，并且具有仅包含外部地址的证书，这可能会导致安装无法用作容器 registry，且没有证书问题。

要避免这种情况，请使用 **automationhub_main_url** 清单变量和值，如 https://pah.example.com 链接到安装清单文件中的私有自动化中心节点。

这会将外部地址添加到 **/etc/pulp/settings.py**。这代表，您只使用外部地址。

有关清单文件变量的详情，请参考 *Red Hat Ansible Automation Platform 安装指南中的清单文件变量*。https://docs.redhat.com/en/documentation/red_hat_ansible_automation_platform/2.4/html-single/red_hat_ansible_automation_platform_installation_guide/index#inventory-files-vars

2.3.1. 高可用性自动化中心要求

在部署高可用性 (HA) 自动化中心前，请确保在您的环境中安装了共享文件系统，并配置了网络存储系统（如果适用）。

2.3.1.1. 所需的共享文件系统

高可用性自动化中心需要您在环境中安装一个共享文件系统，如 NFS。在运行 Red Hat Ansible Automation Platform 安装程序前，请验证作为共享文件系统安装的一部分，在集群中安装了 **/var/lib/pulp** 目录。如果在其中一个节点中没有检测到 **/var/lib/pulp**，则 Red Hat Ansible Automation Platform 安装程序会返回错误，从而导致高可用性自动化中心设置失败。

如果您收到一个节点上没有检测到 **/var/lib/pulp** 的错误，请确保 **/var/lib/pulp** 已在所有服务器中正确挂载并重新运行安装程序。

2.3.1.2. 为网络存储安装 firewalld

如果要使用自动化 hub 节点本身的网络存储安装 HA 自动化中心，您必须首先安装和使用 **firewalld**，以根据共享存储系统的要求打开共享存储系统所需的端口，然后才能运行 Ansible Automation Platform 安装程序。

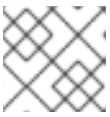
执行以下命令安装和配置 **firewalld** ：

1. 安装 **firewalld** 守护进程：

```
$ dnf install firewalld
```

2. 使用以下命令在 <service> 下添加网络存储：

```
$ firewall-cmd --permanent --add-service=<service>
```



注意

如需支持的服务列表，请使用 **$ firewall-cmd --get-services** 命令

3. 重新载入以应用配置：

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

2.4. 事件驱动 ANSIBLE 控制器系统要求

Event-Driven Ansible 控制器是一个可以处理长时间运行的进程数（如规则手册激活）的单节点系统，具体取决于 CPU 内核数。默认情况下，使用以下最低要求来运行，最多 12 个并发激活：

要求	必填
RAM	16 GB
CPU	4
本地磁盘	最小 40 GB

重要

- 如果您正在运行 Red Hat Enterprise Linux 8 并希望设置内存限值，则必须在安装 Event-Driven Ansible 前启用 cgroup v2。具体步骤请查看 知识库支持(KCS)文章，Red Hat Enterprise Linux 8 的 Ansible Automation Platform Event-Driven Ansible 控制器需要 cgroupv2。
- 当您在标准条件下激活 Event-Driven Ansible 规则手册时，它使用大约 250 MB 内存。但是，根据规则的复杂性以及处理事件的卷和大小，实际内存消耗可能会很大不同。在预计大量事件或规则手册复杂性很高的情况下，对暂存环境中的资源使用情况进行初步评估。这样可确保您的激活的最大数量取决于您的资源容量。如需设置 Event-Driven Ansible 控制器最大运行激活的示例，请参阅单一自动化控制器、单一自动化中心和带有外部（安装程序管理的）数据库 的单个 Event-Driven Ansible 控制器节点。

2.5. POSTGRESQL 要求

Red Hat Ansible Automation Platform 使用 PostgreSQL 13。在将 PostgreSQL 用户密码保存到数据库前，会使用 SCRAM-SHA-256 安全散列算法对其进行处理。

要确定您的自动化控制器实例是否可以访问数据库，您可以使用 **awx-manage check_db** 命令。

表 2.5. 数据库

Service	必填	备注

Service	必填	备注
数据库	• 20 GB 专用硬盘空间 • 4 个 CPU • 16 GB RAM	• 建议大于 150 GB • 存储卷必须为高基线 IOPS 进行评级 (1500 或更多)。 • 所有自动化控制器数据都存储在数据库中。通过管理的主机数量、作业运行数量、事实缓存中存储的 fact 数量以及单个作业中的任务数量，数据库存储会增加。例如，一个 playbook 在 250 个主机中每小时运行一次（一天 24 次），20 个任务每周会在数据库中存储超过 800000 个事件。 • 如果数据库中没有足够的空间，则必须定期清理旧作业运行和事实。如需更多信息，请参阅 *自动化控制器 管理指南中的管理作业*。

PostgreSQL 配置

另外，您可以将 PostgreSQL 数据库配置为不由 Red Hat Ansible Automation Platform 安装程序管理的独立节点。当 Ansible Automation Platform 安装程序管理数据库服务器时，它会使用通常为大多数工作负载推荐的默认值配置服务器。有关可以用来提高数据库性能的设置的更多信息，请参阅 数据库设置。

其他资源

有关调整 PostgreSQL 服务器的更多信息，请参阅 PostgreSQL 文档。

2.5.1. 设置外部（客户支持）数据库

重要

红帽不支持使用外部（客户支持）数据库，但客户使用它。以下有关 inital 配置的指导只从产品安装角度提供，以避免相关的支持请求。

要在外部 PostgreSQL 兼容数据库中创建用于自动化控制器的数据库、用户和密码，请使用以下步骤。

流程

1. 安装，然后连接到具有超级用户权限的 PostgreSQL 兼容数据库服务器。

```
# psql -h <db.example.com> -U superuser -p 5432 -d postgres <Password for user superuser>:
```

其中：

```
-h hostname
--host=hostname
```

指定运行服务器的机器的主机名。如果该值以斜杠开头，它将用作 Unix-domain 套接字的目录。

```
-d dbname
--dbname=dbname
```

指定要连接到的数据库的名称。这等同于将 **dbname** 指定为命令行中的第一个非选项参数。**dbname** 可以是连接字符串。如果是，则连接字符串参数会覆盖任何冲突的命令行选项。

```
-U username
--username=username
```

以用户 **用户名** 而非默认值连接到数据库。（您必须有权限才能这样做。）

2. 使用分配给用户的 **createDB** 或管理员角色，创建用户、数据库和密码。如需更多信息，请参阅数据库角色。

3. 将数据库凭据和主机详情作为外部数据库添加到自动化控制器清单文件。
以下示例中使用了默认值。

```
[database]
pg_host='db.example.com'
pg_port=5432
pg_database='awx'
pg_username='awx'
pg_password='redhat'
```

4. 运行安装程序。
如果您将 PostgreSQL 数据库与自动化控制器搭配使用，则数据库由 connect 用户所有，且必须为其分配 **createDB** 或管理员角色。

5. 检查您是否能够使用用户、密码和数据库名称连接到创建的数据库。

6. 检查用户的权限，该用户应具有 **createDB** 或 administrator 角色。

注意

在此过程中，您必须检查外部数据库覆盖。如需更多信息，请参阅
https://access.redhat.com/articles/4010491

2.5.2. 为自动化中心 PostgreSQL 数据库启用 hstore 扩展

在 Ansible Automation Platform 2.4 中，数据库迁移脚本使用 **hstore** 字段来存储信息，因此必须启用对自动化中心 PostgreSQL 数据库的 **hstore** 扩展。

使用 Ansible Automation Platform 安装程序和受管 PostgreSQL 服务器时，此过程是自动的。

如果 PostgreSQL 数据库是外部的，则必须在自动化中心安装前手动为自动化中心 PostreSQL 数据库启用 **hstore** 扩展。

如果在自动化中心安装前没有启用 **hstore** 扩展，在数据库迁移过程中会引发故障。

流程

1. 检查 PostgreSQL 服务器上是否有扩展（自动化 hub 数据库）。

```
$ psql -d <automation hub database> -c "SELECT * FROM pg_available_extensions WHERE
name='hstore'"
```

其中 **<automation hub database>** 的默认值为 **automationhub**。

带有 hstore 可用的输出示例 ：

```
name  | default_version | installed_version |comment
------+-----------------+-------------------+----------------------------------------------------
 hstore | 1.7           |                   | data type for storing sets of (key, value) pairs
(1 row)
```

带有 hstore 不可用的输出示例 ：

```
 name | default_version | installed_version | comment
------+-----------------+-------------------+---------
(0 rows)
```

2. 在基于 RHEL 的服务器上，**hstore** 扩展包含在 **postgresql-contrib** RPM 软件包中，该软件包在安装 PostgreSQL 服务器 RPM 软件包时不会自动安装。
要安装 RPM 软件包，请使用以下命令：

```
dnf install postgresql-contrib
```

3. 使用以下命令，在自动化中心数据库上创建 **hstore** PostgreSQL 扩展：

```
$ psql -d <automation hub database> -c "CREATE EXTENSION hstore;"
```

输出：

```
CREATE EXTENSION
```

4. 在以下输出中，**installed_version** 字段包含使用的 **hstore** 扩展，表示启用了 **hstore**。

```
name | default_version | installed_version | comment
-----+-----------------+-------------------+----------------------------------------------------
hstore |     1.7      |       1.7         | data type for storing sets of (key, value) pairs
(1 row)
```

2.5.3. Ansible Automation Platform PostgreSQL 数据库的基准测试存储性能

使用 Flexible I/O Tester (FIO)工具检查是否满足最低 Ansible Automation Platform PostgreSQL 数据库要求。FIO 是一个用于对存储系统的读取和写入 IOPS 性能进行基准测试的工具。

先决条件

- 您已安装了 Flexible I/O Tester (**fio**)存储性能基准工具。
  要安装 **fio**，请以 root 用户身份运行以下命令：

```
# yum -y install fio
```

- 您有足够的磁盘空间来存储 **fio** 测试数据日志文件。
  该流程中显示的示例至少需要 60GB 磁盘空间在 **/tmp** 目录中：
  - **numjobs** 设置由 命令运行的作业数量。
  - **size=10G** 设置每个作业生成的文件大小。
- 您已调整了 **size** 参数的值。调整此值可减少测试数据量。

流程

1. 运行随机写入测试：

```
$ fio --name=write_iops --directory=/tmp --numjobs=3 --size=10G \
--time_based --runtime=60s --ramp_time=2s --ioengine=libaio --direct=1 \
--verify=0 --bs=4K --iodepth=64 --rw=randwrite \
--group_reporting=1 > /tmp/fio_benchmark_write_iops.log \
2>> /tmp/fio_write_iops_error.log
```

2. 运行随机读测试：

```
$ fio --name=read_iops --directory=/tmp \
--numjobs=3 --size=10G --time_based --runtime=60s --ramp_time=2s \
--ioengine=libaio --direct=1 --verify=0 --bs=4K --iodepth=64 --rw=randread \
--group_reporting=1 > /tmp/fio_benchmark_read_iops.log \
2>> /tmp/fio_read_iops_error.log
```

3. 查看结果：
   在基准命令编写的日志文件中，搜索以 **iops** 开头的行。此行显示测试的最小、最大值和平均值。

   以下示例显示了日志文件中随机读取测试的行：

```
$ cat /tmp/fio_benchmark_read_iops.log
read_iops: (g=0): rw=randread, bs=(R) 4096B-4096B, (W) 4096B-4096B, (T) 4096B-4096B,
ioengine=libaio, iodepth=64
[…]
   iops        : min=50879, max=61603, avg=56221.33, stdev=679.97, samples=360
[…]
```

   您必须根据您自己的业务需求、应用程序工作负载和新需求审核、监控和重新查看日志文件。

第 3 章 安装 RED HAT ANSIBLE AUTOMATION PLATFORM

Ansible Automation Platform 是一个模块化平台。您可以使用其他自动化平台组件部署自动化控制器，如自动化中心和 Event-Driven Ansible 控制器。如需有关 Ansible Automation Platform 提供的组件的更多信息，请参阅 Red Hat Ansible Automation Platform 计划指南中的 Red Hat Ansible Automation Platform 组件。

Red Hat Ansible Automation Platform 有几个支持的安装场景。要安装 Red Hat Ansible Automation Platform，您必须编辑 inventory 文件参数来指定您的安装场景。您可以将以下之一用作您自己的清单文件的基础：

- 带有外部（安装程序管理的）数据库的单一自动化控制器
- 单个自动化控制器和带有外部（安装程序管理的）数据库的单一自动化中心
- 带有外部（安装程序管理的 ）数据库的单一自动化控制器、单一自动化中心和单一事件驱动的 ansible 控制器节点

3.1. 编辑 RED HAT ANSIBLE AUTOMATION PLATFORM 安装程序清单文件

您可以使用 Red Hat Ansible Automation Platform 安装程序清单文件指定您的安装场景。

流程

1. 进入安装程序：
   a. [RPM 安装的软件包]

   ```
   $ cd /opt/ansible-automation-platform/installer/
   ```

   b. [Bundled installer]

   ```
   $ cd ansible-automation-platform-setup-bundle-<latest-version>
   ```

   c. [Online installer]

   ```
   $ cd ansible-automation-platform-setup-<latest-version>
   ```

2. 使用文本编辑器打开**清单**文件。
3. 编辑**清单**文件参数，以指定您的安装场景。您可以使用其中一个支持的安装场景 示例 作为 **清单文件** 的基础。

其他资源

- 如需 Ansible 安装清单文件中使用的预定义变量的完整列表，请参阅清单文件变量。

3.2. 基于安装场景的清单文件示例

红帽支持多种 Ansible Automation Platform 安装场景。您可以使用示例文件作为基础开发自己的清单文件，也可以使用最接近您首选安装场景的示例。

3.2.1. 基于安装场景的清单文件建议

在为 Ansible Automation Platform 选择安装方法前，请查看以下建议。熟悉这些建议可简化安装过程。

- Red Hat Ansible Automation Platform 或 Automation hub：在 **[automationhub]** 组中添加一个 Automation hub 主机。
- 对于生产环境或客户环境中的 Ansible Automation Platform 版本，不要在同一节点上安装自动化控制器和自动化中心。这可能导致争用问题和大量资源的使用。
- 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自动化中心同步和安装内容。
  FQDN 不得包含 **-** 或 **_** 符号，因为它无法正确处理。

  不要使用 **localhost**。
- **admin** 是初始登录到 Ansible Automation Platform 的默认用户 ID，不能在清单文件中更改。
- 在 **pg_password** 中使用特殊字符是有限的。支持 **!**, **#**, **0** 和 **@** 字符。使用其他特殊字符可能会导致设置失败。
- 在 **registry_username** 和 **registry_password** 中输入 Red Hat Registry Service account 凭证以链接到 Red Hat 容器 registry。
- 只有在使用非捆绑包安装程序时，才需要清单文件变量 **registry_username** 和 **registry_password**。

3.2.1.1. 带有外部（安装程序管理的）数据库的单一自动化控制器

使用本示例填充清单文件来安装 Red Hat Ansible Automation Platform。此安装清单文件包含单一自动化控制器节点，单独节点上具有外部数据库。

```
[automationcontroller]
controller.example.com

[database]
data.example.com

[all:vars]
admin_password='<password>'
pg_host='data.example.com'
pg_port=5432
pg_database='awx'
pg_username='awx'
pg_password='<password>'
pg_sslmode='prefer'  # set to 'verify-full' for client-side enforced SSL

registry_url='registry.redhat.io'
registry_username='<registry username>'
registry_password='<registry password>'

# SSL-related variables
# If set, this will install a custom CA certificate to the system trust store.
# custom_ca_cert=/path/to/ca.crt
# Certificate and key to install in nginx for the web UI and API
# web_server_ssl_cert=/path/to/tower.cert
```

```
# web_server_ssl_key=/path/to/tower.key
# Server-side SSL settings for PostgreSQL (when we are installing it).
# postgres_use_ssl=False
# postgres_ssl_cert=/path/to/pgsql.crt
# postgres_ssl_key=/path/to/pgsql.key
```

3.2.1.2. 单个自动化控制器和带有外部（安装程序管理的）数据库的单一自动化中心

使用本示例填充清单文件，以使用外部（安装程序管理）数据库部署单一自动化控制器和自动化中心实例。

```
[automationcontroller]
controller.example.com

[automationhub]
automationhub.example.com

[database]
data.example.com

[all:vars]
admin_password='<password>'
pg_host='data.example.com'
pg_port='5432'
pg_database='awx'
pg_username='awx'
pg_password='<password>'
pg_sslmode='prefer'  # set to 'verify-full' for client-side enforced SSL

registry_url='registry.redhat.io'
registry_username='<registry username>'
registry_password='<registry password>'

automationhub_admin_password= <PASSWORD>

automationhub_pg_host='data.example.com'
automationhub_pg_port=5432

automationhub_pg_database='automationhub'
automationhub_pg_username='automationhub'
automationhub_pg_password=<PASSWORD>
automationhub_pg_sslmode='prefer'

# The default install will deploy a TLS enabled Automation Hub.
# If for some reason this is not the behavior wanted one can
# disable TLS enabled deployment.
#
# automationhub_disable_https = False
# The default install will generate self-signed certificates for the Automation
# Hub service. If you are providing valid certificate via automationhub_ssl_cert
# and automationhub_ssl_key, one should toggle that value to True.
#
# automationhub_ssl_validate_certs = False
# SSL-related variables
# If set, this will install a custom CA certificate to the system trust store.
```

```
# custom_ca_cert=/path/to/ca.crt
# Certificate and key to install in Automation Hub node
# automationhub_ssl_cert=/path/to/automationhub.cert
# automationhub_ssl_key=/path/to/automationhub.key

# Certificate and key to install in nginx for the web UI and API
# web_server_ssl_cert=/path/to/tower.cert
# web_server_ssl_key=/path/to/tower.key
# Server-side SSL settings for PostgreSQL (when we are installing it).
# postgres_use_ssl=False
# postgres_ssl_cert=/path/to/pgsql.crt
# postgres_ssl_key=/path/to/pgsql.key
```

3.2.1.2.1. 将自动化中心连接到 Red Hat Single Sign-On 环境

您可以进一步配置清单文件，将自动化中心连接到 Red Hat Single Sign-On 安装。

在连接到由 Ansible Automation Platform 管理的 Red Hat Single Sign-On 安装时，您必须配置不同的变量集合，而不是连接到外部 Red Hat Single Sign-On 安装。

如需有关这些清单变量的更多信息，请参阅为 Ansible Automation Platform 安装和配置中央身份验证 。

3.2.1.2.2. 高可用性自动化中心

使用以下示例填充清单文件来安装高度可用的自动化中心。此清单文件包含一个具有集群设置的高可用性自动化中心。

您可以进一步配置 HA 部署来实现 Red Hat Single Sign-On，并在 SELinux 中启用自动化中心的高可用性部署。

指定数据库主机 IP

- 使用 **automation_pg_host** 和 **automation_pg_port** 清单变量指定数据库主机的 IP 地址。例如：

```
automationhub_pg_host='192.0.2.10'
automationhub_pg_port=5432
```

- 另外，使用 **automationhub_pg_host** 清单变量中的值，在 [database] 部分中指定数据库主机的 IP 地址：

```
[database]
192.0.2.10
```

列出集群设置中的所有实例

- 如果安装集群设置，请将 [automationhub] 部分中的 **localhost ansible_connection=local** 替换为所有实例的主机名或 IP。例如：

```
[automationhub]
automationhub1.testing.ansible.com ansible_user=cloud-user ansible_host=192.0.2.18
automationhub2.testing.ansible.com ansible_user=cloud-user ansible_host=192.0.2.20
automationhub3.testing.ansible.com ansible_user=cloud-user ansible_host=192.0.2.22
```

后续步骤

检查每个私有自动化中心服务器的 **/etc/pulp/settings.py** 中是否存在以下指令：

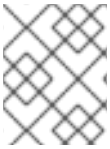
```
USE_X_FORWARDED_PORT = True
USE_X_FORWARDED_HOST = True
```

注意

如果没有指定 **Automationhub_main_url**，则默认使用 [automationhub] 组中的第一个节点。

3.2.1.2.3. 在 SELinux 中启用自动化中心的高可用性(HA)部署

您可以配置清单文件，以便在 SELinux 中启用自动化中心的高可用性部署。您必须为 **/var/lib/pulp** 和 **/var/lib/pulp/pulpcore_static** 创建两个挂载点，然后为每个上下文分配适当的 SELinux 上下文。



注意

您需要为 **/var/lib/pulp** 添加上下文，并在为 **/var/lib/pulp** 添加上下文前运行 Ansible Automation Platform 安装程序。

先决条件

- 您已在服务器上配置了 NFS 导出。

流程

1. 在 **/var/lib/pulp** 中创建挂载点：

```
$ mkdir /var/lib/pulp/
```

2. 使用文本编辑器打开 **/etc/fstab**，然后添加以下值：

```
srv_rhel8:/data /var/lib/pulp nfs defaults,_netdev,nosharecache 0 0
srv_rhel8:/data/pulpcore_static /var/lib/pulp/pulpcore_static nfs
defaults,_netdev,nosharecache,context="system_u:object_r:httpd_sys_content_rw_t:s0" 0 0
```

3. 运行 reload systemd manager configuration 命令：

```
$ systemctl daemon-reload
```

4. 为 **/var/lib/pulp** 运行 mount 命令：

```
$ mount /var/lib/pulp
```

5. 在 **/var/lib/pulp/pulpcore_static** 处创建一个挂载点：

```
$ mkdir /var/lib/pulp/pulpcore_static
```

6. 运行 mount 命令：

```
$ mount -a
```

7. 设置挂载点后，运行 Ansible Automation Platform 安装程序：

```
$ setup.sh -- -b --become-user root
```

8. 安装完成后，卸载 **/var/lib/pulp/** 挂载点。

后续步骤

1. 应用相应的 SELinux 上下文。
2. 配置 pulpcore.serivce。

其它资源

- 如需 SELinux 上下文列表，请参阅 Pulp 项目文档的 SELinux 要求。
- 有关 Pulp 文件夹的完整描述，请参阅 Filesystem Layout。

3.2.1.2.4. 配置 pulpcore.service

配置清单文件并应用 SELinux 上下文后，您需要配置 pulp 服务。

流程

1. 设置了两个挂载点，关闭 Pulp 服务以配置 **pulpcore.service** ：

```
$ systemctl stop pulpcore.service
```

2. 使用 **systemctl** 编辑 **pulpcore.service** ：

```
$ systemctl edit pulpcore.service
```

3. 将以下条目添加到 **pulpcore.service** 中，以确保自动化中心服务仅在启动网络并挂载远程挂载点后启动：

```
[Unit]
After=network.target var-lib-pulp.mount
```

4. 启用 **remote-fs.target** ：

```
$ systemctl enable remote-fs.target
```

5. 重启系统：

```
$ systemctl reboot
```

故障排除

pulpcore SELinux 策略中存在一个错误，可能会导致 **etc/pulp/certs/** 中的令牌身份验证的公钥/私钥没有正确的 SELinux 标签，从而导致 pulp 进程失败。当发生这种情况时，运行以下命令临时附加正确的标签：

```
$ chcon system_u:object_r:pulpcore_etc_t:s0 /etc/pulp/certs/token_{private,public}_key.pem
```

重复此命令，以便在重新标记系统时重新附加正确的 SELinux 标签。

3.2.1.2.5. 应用 SELinux 上下文

配置清单文件后，您必须应用上下文以便在 SELinux 上启用自动化中心的高可用性(HA)部署。

流程

1. 关闭 Pulp 服务：

```
$ systemctl stop pulpcore.service
```

2. 卸载 **/var/lib/pulp/pulpcore_static** ：

```
$ umount /var/lib/pulp/pulpcore_static
```

3. 卸载 **/var/lib/pulp/**:

```
$ umount /var/lib/pulp/
```

4. 使用文本编辑器打开 **/etc/fstab**，然后使用以下内容替换 **/var/lib/pulp** 的现有值：

```
srv_rhel8:/data /var/lib/pulp nfs
defaults,_netdev,nosharecache,context="system_u:object_r:pulpcore_var_lib_t:s0" 0 0
```

5. 运行 mount 命令：

```
$ mount -a
```

3.2.1.3. 在私有自动化 hub 中配置内容签名

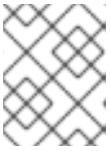
要成功签名并发布 Ansible 认证的内容集合，您必须配置私有自动化中心进行签名。

前提条件

- 您的 TIPC 密钥对已安全设置并管理您的机构。
- 您的公钥-私钥对有权在私有自动化中心上配置内容签名。

流程

1. 创建只接受文件名的签名脚本。



注意

此脚本充当签名服务，必须使用通过 **PULP_SIGNING_KEY_FINGERPRINT** 环境变量指定的密钥为该文件生成 ascii-armored 分离 **gpg** 签名。

该脚本打印一个 JSON 结构，其格式如下：

```
{"file": "filename", "signature": "filename.asc"}
```

所有文件名都是当前工作目录中的相对路径。对于分离的签名，文件名必须保持相同。

Example:

以下脚本为内容生成签名：

```
#!/usr/bin/env bash

FILE_PATH=$1
SIGNATURE_PATH="$1.asc"

ADMIN_ID="$PULP_SIGNING_KEY_FINGERPRINT"
PASSWORD="password"

# Create a detached signature
gpg --quiet --batch --pinentry-mode loopback --yes --passphrase \
  $PASSWORD --homedir ~/.gnupg/ --detach-sign --default-key $ADMIN_ID \
  --armor --output $SIGNATURE_PATH $FILE_PATH

# Check the exit status
STATUS=$?
if [ $STATUS -eq 0 ]; then
  echo {\"file\": \"$FILE_PATH\", \"signature\": \"$SIGNATURE_PATH\"}
else
  exit $STATUS
fi
```

部署私有自动化中心后，为 Ansible Automation Platform 集群启用了签名后，会在集合中会显示新的 UI。

2. 查看 Ansible Automation Platform 安装程序清单文件中的以 **automationhub_*** 开头的选项。

```
[all:vars]
.
.
.
automationhub_create_default_collection_signing_service = True
automationhub_auto_sign_collections = True
automationhub_require_content_approval = True
automationhub_collection_signing_service_key = /abs/path/to/galaxy_signing_service.gpg
automationhub_collection_signing_service_script = /abs/path/to/collection_signing.sh
```

两个新密钥(**automationhub_auto_sign_collections** 和 **automationhub_require_content_approval**)表示必须签名集合，并在上传到私有自动化中心后需要批准。

3.2.1.4. 私有自动化中心上的 LDAP 配置

您必须在 Red Hat Ansible Automation Platform 安装程序清单文件中设置以下六个变量，以配置用于 LDAP 身份验证的私有自动化中心：

- **automationhub_authentication_backend**
- **automationhub_ldap_server_uri**
- **automationhub_ldap_bind_dn**
- **automationhub_ldap_bind_password**
- **automationhub_ldap_user_search_base_dn**
- **automationhub_ldap_group_search_base_dn**

如果缺少其中任何这些变量，Ansible Automation 安装程序将无法完成安装。

3.2.1.4.1. 设置清单文件变量

当使用 LDAP 身份验证配置私有自动化中心时，您必须在安装过程中在清单文件中设置正确的变量。

流程

1. 根据 编辑 Red Hat Ansible Automation Platform 安装程序清单文件 中的步骤访问您的清单文件。
2. 使用以下示例来设置 Ansible Automation Platform 清单文件：

```
automationhub_authentication_backend = "ldap"

automationhub_ldap_server_uri = "ldap://ldap:389"   (for LDAPs use
automationhub_ldap_server_uri = "ldaps://ldap-server-fqdn")
automationhub_ldap_bind_dn = "cn=admin,dc=ansible,dc=com"
automationhub_ldap_bind_password = "GoodNewsEveryone"
automationhub_ldap_user_search_base_dn = "ou=people,dc=ansible,dc=com"
automationhub_ldap_group_search_base_dn = "ou=people,dc=ansible,dc=com"
```

注意

以下变量将使用默认值设置，除非您使用其他选项进行了设置。

```
auth_ldap_user_search_scope= 'SUBTREE'
auth_ldap_user_search_filter= '(uid=%(user)s)'
auth_ldap_group_search_scope= 'SUBTREE'
auth_ldap_group_search_filter= '(objectClass=Group)'
auth_ldap_group_type_class= 'django_auth_ldap.config:GroupOfNamesType'
```

3. 可选：在私有自动化中心中设置额外的参数，如用户组、超级用户访问或镜像。前往 配置额外的 LDAP 参数 以完成此可选步骤。

3.2.1.4.2. 配置额外的 LDAP 参数

如果您计划设置超级用户访问权限、用户组、镜像或其他额外参数，您可以创建一个在 **ldap_extra_settings** 字典中组成它们的 YAML 文件。

流程

1. 创建一个包含 **ldap_extra_settings** 的 YAML 文件。
   - Example:

```
#ldapextras.yml
---
ldap_extra_settings:
 <LDAP_parameter>: <Values>

...
```

2. 添加设置所需的任何参数。以下示例描述了您可以在 **ldap_extra_settings** 中设置的 LDAP 参数：
   - 使用本示例根据 LDAP 组中的成员资格设置超级用户标志。

```
#ldapextras.yml
---
ldap_extra_settings:
  AUTH_LDAP_USER_FLAGS_BY_GROUP: {"is_superuser": "cn=pah-admins,ou=groups,dc=example,dc=com",}
...
```

   - 使用本示例设置超级用户访问权限。

```
#ldapextras.yml
---
ldap_extra_settings:
  AUTH_LDAP_USER_FLAGS_BY_GROUP: {"is_superuser": "cn=pah-admins,ou=groups,dc=example,dc=com",}
...
```

   - 使用本示例来镜像（mirror）您属于的所有 LDAP 组。

```
#ldapextras.yml
---
ldap_extra_settings:
  AUTH_LDAP_MIRROR_GROUPS: True
...
```

   - 使用本示例映射 LDAP 用户属性（如用户名、姓氏和电子邮件地址）。

```
#ldapextras.yml
---
ldap_extra_settings:
  AUTH_LDAP_USER_ATTR_MAP: {"first_name": "givenName", "last_name": "sn", "email": "mail",}
...
```

- 使用以下示例根据 LDAP 组成员资格授予或拒绝访问权限：
  - 要授予私有自动化中心访问权限（例如，**cn=pah-nosoupforyou,ou=groups,dc=example,dc=com** 组的成员）：

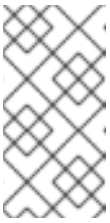
```
#ldapextras.yml
---
ldap_extra_settings:
  AUTH_LDAP_REQUIRE_GROUP: 'cn=pah-
nosoupforyou,ou=groups,dc=example,dc=com'
...
```

  - 拒绝私有自动化中心访问（例如，**cn=pah-nosoupforyou,ou=groups,dc=example,dc=com** 组的成员）：

```
#ldapextras.yml
---
ldap_extra_settings:
  AUTH_LDAP_DENY_GROUP: 'cn=pah-
nosoupforyou,ou=groups,dc=example,dc=com'
...
```

- 使用本示例启用 LDAP 调试日志记录。

```
#ldapextras.yml
---
ldap_extra_settings:
  GALAXY_LDAP_LOGGING: True
...
```



注意

如果重新运行 **setup.sh** 或在短时间内启用了调试日志，则可以将包含 **GALAXY_LDAP_LOGGING: True** 的行手动添加到私有自动化中心上的 **/etc/pulp/settings.py** 文件。重启 **pulpcore-api.service** 和 **nginx.service** 以使更改生效。为了避免因为人为错误而失败，请仅在需要时使用此方法。

- 通过设置变量 **AUTH_LDAP_CACHE_TIMEOUT**，使用本示例配置 LDAP 缓存。

```
#ldapextras.yml
---
ldap_extra_settings:
  AUTH_LDAP_CACHE_TIMEOUT: 3600
...
```

3. 在私有自动化中心安装过程中运行 **setup.sh -e @ldapextras.yml**。验证您已正确设置，请确认您可以在私有自动化中心上的 **/etc/pulp/settings.py** 文件中查看所有设置。

3.2.1.4.3. LDAP 引用

如果您的 LDAP 服务器返回引用，您可能需要禁用引用才能在私有自动化中心中使用 LDAP 成功进行身份验证。

如果没有，则返回以下信息：

```
Operation unavailable without authentication
```

要禁用 LDAP REFERRALS 查找，请设置：

```
GALAXY_LDAP_DISABLE_REFERRALS = true
```

这会将 **AUTH_LDAP_CONNECTIONS_OPTIONS** 设置为正确的选项。

3.2.1.5. 带有外部（安装程序管理的）数据库的单一自动化控制器、单一自动化中心和单个 Event-Driven Ansible 控制器节点

使用本示例填充清单文件，以使用外部（安装程序管理）数据库部署单一自动化控制器、自动化中心和 Event-Driven Ansible 控制器实例。

重要

- 此场景至少需要自动化控制器 2.4 才能成功部署 Event-Driven Ansible 控制器。
- event-Driven Ansible 控制器必须安装在单独的服务器上，且不能安装到与自动化中心和自动化控制器相同的主机上。
- 当您在标准条件下激活 Event-Driven Ansible rulebook 时，它使用大约 250 MB 内存。但是，根据规则的复杂性以及处理事件的卷和大小，实际内存消耗可能会很大不同。在预计大量事件或规则手册复杂性很高的情况下，对暂存环境中的资源使用情况进行初步评估。这样可确保您的激活的最大数量取决于您的资源容量。在以下示例中，默认的 **automationedacontroller_max_running_activations** 设置为 12，但您可以根据您的容量进行调整。

```
[automationcontroller]
controller.example.com

[automationhub]
automationhub.example.com

[automationedacontroller]
automationedacontroller.example.com

[database]
data.example.com

[all:vars]
admin_password='<password>'
pg_host='data.example.com'
pg_port='5432'
pg_database='awx'
pg_username='awx'
pg_password='<password>'
pg_sslmode='prefer'  # set to 'verify-full' for client-side enforced SSL

registry_url='registry.redhat.io'
registry_username='<registry username>'
registry_password='<registry password>'

# {HubNameStart} configuration
```

```
automationhub_admin_password= <PASSWORD>

automationhub_pg_host='data.example.com'
automationhub_pg_port=5432

automationhub_pg_database='automationhub'
automationhub_pg_username='automationhub'
automationhub_pg_password=<PASSWORD>
automationhub_pg_sslmode='prefer'

# Automation {EDAController} configuration

automationedacontroller_admin_password='<eda-password>'

automationedacontroller_pg_host='data.example.com'
automationedacontroller_pg_port=5432

automationedacontroller_pg_database='automationedacontroller'
automationedacontroller_pg_username='automationedacontroller'
automationedacontroller_pg_password='<password>'

# Keystore file to install in SSO node
# sso_custom_keystore_file='/path/to/sso.jks'

# This install will deploy SSO with sso_use_https=True
# Keystore password is required for https enabled SSO
sso_keystore_password=''

# This install will deploy a TLS enabled Automation Hub.
# If for some reason this is not the behavior wanted one can
# disable TLS enabled deployment.
#
# automationhub_disable_https = False
# The default install will generate self-signed certificates for the Automation
# Hub service. If you are providing valid certificate via automationhub_ssl_cert
# and automationhub_ssl_key, one should toggle that value to True.
#
# automationhub_ssl_validate_certs = False
# SSL-related variables
# If set, this will install a custom CA certificate to the system trust store.
# custom_ca_cert=/path/to/ca.crt
# Certificate and key to install in Automation Hub node
# automationhub_ssl_cert=/path/to/automationhub.cert
# automationhub_ssl_key=/path/to/automationhub.key

# Certificate and key to install in nginx for the web UI and API
# web_server_ssl_cert=/path/to/tower.cert
# web_server_ssl_key=/path/to/tower.key
# Server-side SSL settings for PostgreSQL (when we are installing it).
# postgres_use_ssl=False
# postgres_ssl_cert=/path/to/pgsql.crt
# postgres_ssl_key=/path/to/pgsql.key

# Boolean flag used to verify Automation Controller's
# web certificates when making calls from Automation {EDAcontroller}.
```

```
# automationedacontroller_controller_verify_ssl = true
#
# Certificate and key to install in Automation {EDAcontroller} node
# automationedacontroller_ssl_cert=/path/to/automationeda.crt
# automationedacontroller_ssl_key=/path/to/automationeda.key
```

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

在您使用用于安装您的私有自动化中心所需的参数更新了清单文档后，运行安装程序设置脚本。

流程

- 运行 **setup.sh** 脚本

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

安装 Red Hat Ansible Automation Platform 将开始。

3.4. 验证自动化控制器的安装

使用您插入到**清**单文件中的 admin 凭据登录来验证您是否已成功安装自动化控制器。

前提条件

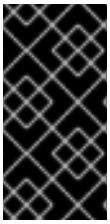
- 端口 443 可用

流程

1. 进入清单文件中为自动化控制器节点指定的 IP **地址**。
2. 使用您在清单文件中设置的用户 ID **admin** 和密码凭证登录。

注意

自动化控制器服务器可从端口 80 (https://<CONTROLLER_SERVER_NAME>/)访问，但重定向到端口 443。



重要

如果安装失败，且您购买了 Red Hat Ansible Automation Platform 的有效许可证，请通过红帽客户门户网站联络 Ansible。

当成功登录到自动化控制器后，则代表您的 Red Hat Ansible Automation Platform 2.4 安装已完成。

3.4.1. 其他自动化控制器配置和资源

请参阅以下资源来探索其他自动化控制器配置。

表 3.1. 配置自动化控制器的资源

资源链接	描述
自动化控制器快速设置指南	设置自动化控制器并运行第一个 playbook。
自动化控制器管理指南	通过客户脚本、管理作业等配置自动化控制器管理。
配置 Red Hat Ansible Automation Platform 的代理支持	使用代理服务器设置自动化控制器。
从自动化控制器管理可用性分析和数据收集	管理您与红帽共享的自动化控制器信息。
自动化控制器用户指南	更详细地查看自动化控制器功能。

3.5. 验证自动化中心安装

使用您插入到清单文件中的 admin 凭证登录来验证您是否已成功安装了自动化中心。

流程

1. 进入清单文件中为自动化中心节点指定的 IP 地址。
2. 使用您在清单文件中设置的用户 ID admin 和密码凭证登录。

重要

如果安装失败，且您购买了 Red Hat Ansible Automation Platform 的有效许可证，请通过红帽客户门户网站联络 Ansible。

当可以成功登录到自动化中心后，则代表您的 Red Hat Ansible Automation Platform 2.4 安装已完成。

3.5.1. 其他自动化中心配置和资源

请参阅以下资源来探索额外的自动化中心配置。

表 3.2. 配置自动化控制器的资源

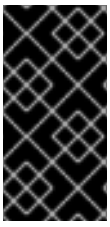
资源链接	描述
在私有自动化 hub 中管理用户访问权限	为自动化中心配置用户访问。
在自动化中心中管理红帽认证的、经过验证的 Ansible Galaxy 内容	在自动化中心中添加内容。
在自动化 hub 中发布专有内容集合	在您的自动化 hub 上发布内部开发的集合。

3.6. 验证 EVENT-DRIVEN ANSIBLE 控制器安装

使用您插入到清单文件中的 admin 凭据登录，验证您是否已成功安装了 Event-Driven Ansible 控制器。

流程

1. 进入到 inventory 文件中为 Event-Driven Ansible 控制器节点指定的 IP 地址。
2. 使用您在清单文件中设置的用户 ID admin 和密码凭证登录。



重要

如果安装失败，且您购买了 Red Hat Ansible Automation Platform 的有效许可证，请通过红帽客户门户网站联络 Ansible。

当成功登录到 Event-Driven Ansible 控制器后，则代表您的 Red Hat Ansible Automation Platform 2.4 安装已完成。

第 4 章 断开连接的安装

如果您没有连接到互联网，或者无法访问在线存储库，您可以在没有活跃互联网连接的情况下安装 Red Hat Ansible Automation Platform。

4.1. 先决条件

在断开连接的网络中安装 Ansible Automation Platform 前，您必须满足以下先决条件：

1. 创建的订阅清单。如需更多信息，请参阅获取清单文件。
2. 客户门户网站中的 Ansible Automation Platform 安装捆绑包已下载。
3. 创建自动化控制器和私有自动化中心服务器的 DNS 记录。

4.2. 在断开连接的 RHEL 上安装 ANSIBLE AUTOMATION PLATFORM

您可以使用自动化控制器中的安装程序管理的数据库安装 Ansible Automation Platform 自动化控制器和私有自动化中心，而无需互联网连接。使用安装捆绑包进行断开连接的安装，因为它包括可在断开连接的环境中安装 Ansible Automation Platform 的其他组件。这包括 Ansible Automation Platform 红帽软件包管理器(RPMs)和默认执行环境(EE)镜像。

4.2.1. 断开连接的安装的系统要求

在执行断开连接的安装 Ansible Automation Platform 前，请确定您的系统具有所有硬件要求。有关硬件要求的更多信息，请参阅 第 2 章。系统要求。

4.2.2. RPM 源

来自 BaseOS 和 AppStream 存储库的 Ansible Automation Platform 的 RPM 依赖项没有包括在安装捆绑包中。要添加这些依赖项，您必须首先获取 BaseOS 和 AppStream 软件仓库的访问权限。使用 Satellite 同步存储库并添加依赖项。如果您希望选择其他工具，您可以在以下选项之间进行选择：

- reposync

- RHEL Binary DVD

注意

RHEL Binary DVD 方法需要支持的 RHEL 版本的 DVD，包括 8.6 或更高版本。有关当前支持哪些 RHEL 版本的详情，请查看 Red Hat Enterprise Linux 生命周期。

其它资源

- Satellite

4.3. 使用 REPOSYNC 同步 RPM 存储库

要执行 reposync，您需要一个可以访问互联网的 RHEL 主机。同步仓库后，您可以将仓库移到从 Web 服务器托管的断开连接的网络中。

流程

1. 附加 BaseOS 和 AppStream 所需的软件仓库：

```
# subscription-manager repos \
    --enable rhel-8-for-x86_64-baseos-rpms \
    --enable rhel-8-for-x86_64-appstream-rpms
```

2. 执行 reposync ：

```
# dnf install yum-utils
# reposync -m --download-metadata --gpgcheck \
    -p /path/to/download
```

a. 使用带有 --download-metadata 且不带 --newest-only 的 reposync。请参阅 RHEL 8 Reposync。

- 如果您不使用 --newest-only， 则下载的仓库为 ~90GB。
- 如果您只使用 --newest-only， 则下载的仓库为 ~14GB。

3. 如果您计划使用 Red Hat Single Sign-On，请同步这些软件仓库：

   a. jb-eap-7.3-for-rhel-8-x86_64-rpms

   b. rh-sso-7.4-for-rhel-8-x86_64-rpms

   完成 reposync 后，您的仓库就可以与 web 服务器一起使用。

4. 将仓库移到断开连接的网络中。

4.4. 创建新的 WEB 服务器以托管仓库

如果您没有现有的 Web 服务器来托管您的存储库，您可以使用同步的存储库创建一个。

流程

1. 安装先决条件：

```
$ sudo dnf install httpd
```

2. 将 httpd 配置为提供 repo 目录：

```
/etc/httpd/conf.d/repository.conf

DocumentRoot '/path/to/repos'

<LocationMatch "^/+$">
    Options -Indexes
    ErrorDocument 403 /.noindex.html
</LocationMatch>

<Directory '/path/to/repos'>
    Options All Indexes FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>
```

3. **确保 apache 用户可读该目录：**

```
$ sudo chown -R apache /path/to/repos
```

4. **配置 SELinux：**

```
$ sudo semanage fcontext -a -t httpd_sys_content_t "/path/to/repos(/.*)?"
$ sudo restorecon -ir /path/to/repos
```

5. **启用 httpd ：**

```
$ sudo systemctl enable --now httpd.service
```

6. **打开防火墙：**

```
$ sudo firewall-cmd --zone=public --add-service=http –add-service=https --permanent
$ sudo firewall-cmd --reload
```

7. **在自动化控制器和自动化中心中，在 */etc/yum.repos.d/local.repo* 中添加一个仓库文件，并在需要时添加可选的仓库：**

```
[Local-BaseOS]
name=Local BaseOS
baseurl=http://<webserver_fqdn>/rhel-8-for-x86_64-baseos-rpms
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release

[Local-AppStream]
name=Local AppStream
baseurl=http://<webserver_fqdn>/rhel-8-for-x86_64-appstream-rpms
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
```

4.5. 从本地挂载的 DVD 访问 RPM 存储库

如果您计划从 RHEL 二进制 DVD 访问存储库，您必须首先设置本地存储库。

流程

1. **挂载 DVD 或者 ISO：**

   a. **DVD**

   ```
   # mkdir /media/rheldvd && mount /dev/sr0 /media/rheldvd
   ```

   b. **ISO**

   ```
   # mkdir /media/rheldvd && mount -o loop rhrhel-8.6-x86_64-dvd.iso /media/rheldvd
   ```

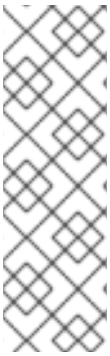
2. **在 /etc/yum.repos.d/dvd.repo 中创建 yum repo 文件**

   ```
   [dvd-BaseOS]
   name=DVD for RHEL - BaseOS
   baseurl=file:///media/rheldvd/BaseOS
   enabled=1
   gpgcheck=1
   gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release

   [dvd-AppStream]
   name=DVD for RHEL - AppStream
   baseurl=file:///media/rheldvd/AppStream
   enabled=1
   gpgcheck=1
   gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
   ```

3. **导入 gpg 密钥：**

   ```
   # rpm --import /media/rheldvd/RPM-GPG-KEY-redhat-release
   ```



注意

如果没有导入密钥，您将看到类似如下的错误

```
# Curl error (6): Couldn't resolve host name for
https://www.redhat.com/security/data/fd431d51.txt [Could not resolve host:
www.redhat.com]
```

其它资源

有关设置软件仓库的详情，请参考需要在 Red Hat Enterprise Linux 8 中为本地挂载的 DVD 设置 yum 软件仓库。

4.6. 在没有互联网连接的 ANSIBLE AUTOMATION PLATFORM 中添加订阅清单

要在没有互联网连接的 Ansible Automation Platform 中添加订阅，请创建并导入订阅清单。

流程

1. 登录到 红帽客户门户网站。
2. 从菜单栏中，选择 Subscriptions，再选择 Subscriptions。
3. 点 New subscription allocation。
4. 命名新的订阅分配。
5. 从 Type 列表中选择 Satellite 6.8。
6. 点 Create。Details 选项卡为您的订阅分配打开。
7. 选择 Subscriptions 选项卡。
8. 点 Add Subscriptions。
9. 找到 Ansible Automation Platform 订阅，并在 Entitlements 框中 添加 您要分配给您的环境的权利数量。每个由 Ansible Automation Platform 管理的节点都需要一个权利：服务器、网络设备等。
10. 点 Submit。

11. 点 Export Manifest。

这会下载在安装后使用自动化控制器导入的文件 *manifest_<allocation name>_<date>.zip*。

4.7. 下载并安装 ANSIBLE AUTOMATION PLATFORM 安装捆绑包

选择安装捆绑包为断开连接的安装下载 Ansible Automation Platform。此捆绑包包括 Ansible Automation Platform 的 RPM 内容以及在安装过程中上传到您的私有自动化中心的默认执行环境镜像。

流程

1. 进入 Red Hat Ansible Automation Platform 下载页面并点 Download Now for the Ansible Automation Platform 2.4 Setup Bundle 来下载 Ansible Automation Platform setup 捆绑包软件包。

2. 在自动化控制器中，解压捆绑包：

```
$ tar xvf \
  ansible-automation-platform-setup-bundle-2.4-1.tar.gz
$ cd ansible-automation-platform-setup-bundle-2.4-1
```

3. 编辑清单文件使其包含所需选项：

   a. **automationcontroller** 组

   b. **automationhub** 组

   c. **admin_password**

   d. **pg_password**

   e. **automationhub_admin_password**

f. **automationhub_pg_host, automationhub_pg_port**

g. **automationhub_pg_password**

清单文件示例

```
[automationcontroller]
automationcontroller.example.org ansible_connection=local

[automationcontroller:vars]
peers=execution_nodes

[automationhub]
automationhub.example.org

[all:vars]
admin_password='password123'

pg_database='awx'
pg_username='awx'
pg_password='dbpassword123'

receptor_listener_port=27199

automationhub_admin_password='hubpassword123'

automationhub_pg_host='automationcontroller.example.org'
automationhub_pg_port=5432

automationhub_pg_database='automationhub'
automationhub_pg_username='automationhub'
automationhub_pg_password='dbpassword123'
automationhub_pg_sslmode='prefer'
```

4. **以 root 用户身份运行 Ansible Automation Platform setup 捆绑包可执行文件：**

```
$ sudo -i
# cd /path/to/ansible-automation-platform-setup-bundle-2.4-1
# ./setup.sh
```

5. **安装完成后，进入安装清单文件中指定的自动化控制器节点的 Fully Qualified Domain Name (FQDN)。**

6. **使用安装清单文件中指定的管理员凭证登录。**

注意

清单文件必须在安装后保持不变，因为它用于备份、恢复和升级功能。将备份副本保存在安全位置，因为清单文件包含密码。

4.8. 完成安装后的任务

完成 Ansible Automation Platform 安装后，请确保自动化中心和自动化控制器正确部署。

4.8.1. 添加控制器订阅

流程

1. 进入 Automation 控制器的 FQDN。使用您在清单文件中的 admin_password 中指定的用户和密码登录。
2. 点 Browse，再选择您之前创建的 *manifest.zip*。
3. 点击 Next。
4. 取消选择 User analytics 和 Automation analytics。它们依赖于互联网连接，且必须关闭。
5. 点击 Next。
6. 阅读最终用户许可证协议，如果同意，点 Submit。

4.8.2. 更新 CA 信任存储

作为安装后任务的一部分，您必须更新软件的证书。默认情况下，Ansible Automation Platform 自动化中心和自动化控制器使用自签名证书安装。因此，控制器不信任 hub 的证书，且不会从 hub 下载执行环境。

为确保自动化控制器从自动化中心下载执行环境，您必须将 hub 的证书颁发机构(CA)证书导入为控制器上的可信证书。您可以通过以下两种方式之一进行此操作，具体取决于 SSH 是否作为自动化控制器和

私有自动化中心之间的 root 用户提供。

4.8.2.1. 以 root 用户身份使用安全副本(SCP)

如果 SSH 作为控制器和私有自动化中心之间的 root 用户提供，请使用 SCP 将私有自动化中心上的 root 证书复制到控制器。

流程

1. **在控制器上运行 update-ca-trust 以更新 CA 信任存储：**

```
$ sudo -i
# scp <hub_fqdn>:/etc/pulp/certs/root.crt
/etc/pki/ca-trust/source/anchors/automationhub-root.crt
# update-ca-trust
```

4.8.2.2. 以非 root 用户身份复制和粘贴

如果 SSH 在私有自动化中心和控制器之间不可用，请将文件 */etc/pulp/certs/root.crt* 复制到名为 */etc/pki/ca-trust/source/anchors/automationhub-root.crt* 的控制器上的新文件中，并将其粘贴到名为 /etc/pki/ca-trust/source/anchors/automationhub-root.crt 的控制器上的新文件。

流程

1. **运行 update-ca-trust 以使用新证书更新 CA 信任存储。在私有自动化中心中运行：**

```
$ sudo -i
# cat /etc/pulp/certs/root.crt
(copy the contents of the file, including the lines with 'BEGIN CERTIFICATE' and
'END CERTIFICATE')
```

1. **在自动化控制器中：**

```
$ sudo -i
# vi /etc/pki/ca-trust/source/anchors/automationhub-root.crt
(paste the contents of the root.crt file from the private automation hub into the new file and write to
disk)
# update-ca-trust
```

其它资源

-

如需有关未知证书颁发机构的更多信息，请参阅 Ansible Automation Platform 2.1 中的项目同步失败并显示未知证书颁发机构错误。

4.9. 将集合导入到私有自动化中心

您可以从 Ansible Automation hub 下载集合作为 tarball 文件，以便在私有自动化中心中使用。认证的集合 在自动化中心混合云控制台上 提供，社区集合则位于 Ansible Galaxy 上。您还必须下载并安装集合所需的所有依赖项。

流程

1. 导航到 link: console.redhat.com，并使用您的红帽凭证登录。
2. 点您要下载的 集合。
3. 点 Download tarball
4. 要验证集合是否有依赖项，请点 Dependencies 选项卡。
5. 下载此集合所需的所有依赖项。

4.10. 创建集合命名空间

在导入集合前，您必须首先在私有自动化中心中为集合创建一个命名空间。您可以通过查看集合 tarball filename 的第一个部分来查找命名空间名称。例如，集合 *ansible-netcommon-3.0.0.tar.gz* 的命名空间是 *ansible*。

流程

1. 登录到自动化中心混合云控制台。
2. 在导航面板中，选择 Collections → Namespaces。
3. 点 Create。

4. 提供命名空间名称。

5. 点 Create。

4.10.1. 使用 Web 控制台导入集合 tarball

创建命名空间后，您可以使用 Web 控制台导入集合。

流程

1. 登录到自动化中心混合云控制台。

2. 在导航面板中，选择 Collections → Namespaces。

3. 点您要将集合导入到的命名空间旁边的 View collections。

4. 点 Upload collection。

5. 点文件夹图标并选择集合的 tarball。

6. 点 Upload。

这将打开"我的导入"页面。您可以查看导入的状态，以及导入的文件和模块的各种详情。

4.10.2. 使用 CLI 导入集合 tarball

您可以使用命令行界面而不是 GUI 将集合导入到私有自动化中心。

流程

1. 将集合 tarball 复制到私有自动化中心。

2. 通过 SSH 登录到私有自动化中心服务器。

3. 将自签名 root CA 证书添加到自动化中心的信任存储中。

```
# cp /etc/pulp/certs/root.crt \
    /etc/pki/ca-trust/source/anchors/automationhub-root.crt
# update-ca-trust
```

4. 使用自动化中心配置更新 /etc/ansible/ansible.cfg 文件。使用令牌或用户名和密码进行身份验证。

```
[galaxy]
server_list = private_hub

[galaxy_server.private_hub]
url=https://<hub_fqdn>/api/galaxy/
token=<token_from_private_hub>
```

5. 使用 ansible-galaxy 命令导入集合。

```
$ ansible-galaxy collection publish <collection_tarball>
```

4.11. 批准导入的集合

使用 GUI 或 CLI 方法导入集合后，您必须使用 GUI 批准它们。批准后，就可以使用。

流程

1. 登录到自动化中心混合云控制台。

2. 在导航面板中，选择 Collections → Approval。

3. 对于您要批准的集合，点 Approve。

4. 这个集合现在可用于您的私有自动化中心。

5. 通过重复步骤 2 和 3，为集合导入所有依赖项。

注意

无论源是什么，集合都会被添加到 "Published" 仓库中。

推荐的集合取决于您的用例。Ansible 和红帽提供这些集合。

4.11.1. 自定义自动化执行环境

使用 ansible-builder 程序创建自定义执行环境镜像。对于断开连接的环境，可以使用以下方法构建自定义执行环境镜像：

- 在面向互联网的系统上构建执行环境镜像，并将其导入到断开连接的环境中。
- 完全在断开连接的环境中构建执行环境镜像，并对使用 ansible-builder 的正常流程进行一些修改。
- 创建一个最小的基础容器镜像，其中包含在断开连接的环境中的所有必要修改，然后从基本容器镜像构建自定义执行环境镜像。

4.11.1.1. 在断开连接的边界间传输自定义虚拟环境镜像

您可以在面向互联网的机器上构建自定义执行环境镜像。创建执行环境后，可在本地 podman 镜像缓存中提供。然后，您可以在断开连接的边界间传输自定义执行环境镜像。

流程

1. 保存镜像：

```
$ podman image save localhost/custom-ee:latest | gzip -c custom-ee-latest.tar.gz
```

使用现有机制（如 sneakernet 或单向 diode）在断开连接的边界间传输文件。

2. 当镜像在断开连接的环境中可用后，将其导入到本地 podman 缓存，标记它并将其推送到断开连接的 hub 中：

```
$ podman image load -i custom-ee-latest.tar.gz
$ podman image tag localhost/custom-ee <hub_fqdn>/custom-ee:latest
$ podman login <hub_fqdn> --tls-verify=false
$ podman push <hub_fqdn>/custom-ee:latest
```

4.12. 在断开连接的环境中构建执行环境

为 Ansible Automation Platform 创建执行环境是一个常见任务，它在断开连接的环境中工作不同。https://docs.redhat.com/en/documentation/red_hat_ansible_automation_platform/2.4/html/creating_and_consuming_execution_environments/index在构建自定义执行环境时，ansible-builder 工具默认为从互联网上以下位置下载内容：

- Red Hat Automation hub (console.redhat.com)或 Ansible Galaxy (galaxy.ansible.com) 用于添加到执行环境镜像的任何 Ansible 内容集合。
- PyPI (pypi.org)，适用于作为集合依赖项所需的任何 python 软件包。
- RPM 存储库，如 RHEL 或 UBI 软件仓库(cdn.redhat.com)，用于在执行环境镜像中添加或更新 RPM （如果需要）。
- registry.redhat.io 用于访问基本容器镜像。

在断开连接的环境中构建执行环境镜像需要从这些位置镜像内容。如需有关 将集合从 Ansible Ansible Galaxy 或自动化中心导入到私有自动化中心 的信息，请参阅将集合导入到私有自动化中心。

传送到断开连接的网络后镜像 PyPI 内容可以使用 Web 服务器或 Nexus 等工件存储库。RHEL 和 UBI 存储库内容可以从面向互联网的 Red Hat Satellite 服务器导出，复制到断开连接的环境中，然后导入到断开连接的 Satellite 中，以便构建自定义执行环境。详情请查看 Air-Gapped Scenario 中的 ISS 导出同步。

默认基础容器镜像 ee-minimal-rhel8 用于创建自定义执行环境镜像，并包含在捆绑的安装程序中。在安装时，此镜像添加到私有自动化中心。如果需要不同的基础容器镜像，如 ee-minimal-rhel9，则必须将其导入到断开连接的网络中，并添加到私有自动化中心容器 registry 中。

断开连接的网络上所有先决条件后，可以使用 ansible-builder 命令来创建自定义执行环境镜像。

4.12.1. 安装 Ansible Builder RPM

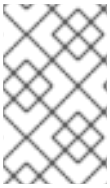
在构建自定义虚拟环境的 RHEL 系统上，您将使用环境中已存在的 Satellite 服务器安装 Ansible Builder RPM。此方法是首选的，因为执行环境镜像可以根据需要使用来自预先存在的 Satellite 的任何 RHEL 内容。

流程

1. 从 Ansible Automation Platform 存储库安装 Ansible Builder RPM。
   a. 将 RHEL 系统订阅到断开连接的网络上的 Satellite。
   b. 附加 Ansible Automation Platform 订阅并启用 AAP 存储库。存储库名称可以是 ansible-automation-platform-2.4-for-rhel-8-x86_64-rpms 或 ansible-automation-platform-2.4-for-rhel-9-x86_64-rpms，具体取决于底层系统上使用的 RHEL 版本。
   c. 安装 Ansible Builder RPM。Ansible Builder RPM 的版本必须是 3.0.0 或更高版本，才能使以下示例正常工作。
2. 从 Ansible Automation Platform 安装捆绑包安装 Ansible Builder RPM。如果断开连接的网络中没有 Satellite 服务器，则使用此方法。
   a. 解压缩 Ansible Automation Platform 安装捆绑包。
   b. 从包含的内容安装 Ansible Builder RPM 及其依赖项。

```
$ tar -xzvf ansible-automation-platform-setup-bundle-2.4-3-x86_64.tar.gz
$ cd ansible-automation-platform-setup-bundle-2.4-3-x86_64/bundle/packages/el8/repos/
$ sudo dnf install ansible-builder-3.0.0-2.el8ap.noarch.rpm \
    python39-requirements-parser-0.2.0-4.el8ap.noarch.rpm \
    python39-bindep-2.10.2-3.el8ap.noarch.rpm \
    python39-jsonschema-4.16.0-1.el8ap.noarch.rpm \
    python39-pbr-5.8.1-2.el8ap.noarch.rpm \
    python39-distro-1.6.0-3.el8pc.noarch.rpm \
    python39-packaging-21.3-2.el8ap.noarch.rpm \
    python39-parsley-1.3-2.el8pc.noarch.rpm \
```

```
python39-attrs-21.4.0-2.el8pc.noarch.rpm \
python39-pyrsistent-0.18.1-2.el8ap.x86_64.rpm \
python39-pyparsing-3.0.9-1.el8ap.noarch.rpm
```



注意

根据所使用的安装捆绑包版本，特定版本可能稍有不同。

其他资源

- 有关在断开连接的网络中创建 Satellite 环境的详情，请参考 在断开连接的网络环境中安装 Satellite 服务器。

4.12.2. 创建自定义执行环境定义

安装 Ansible Builder RPM 后，请按照以下步骤创建自定义执行环境。

1. 为创建自定义执行环境时使用的构建工件创建一个目录。所有使用以下步骤创建的新文件都将在此目录中创建。

```
$ mkdir $HOME/custom-ee $HOME/custom-ee/files
$ cd $HOME/custom-ee/
```

2. 创建一个 execution-environment.yml 文件，该文件定义自定义执行环境的要求。

注意

需要执行环境定义格式的版本 3，因此请确保 execution-environment.yml 文件包含 version: 3，然后再继续。

a. 覆盖基础镜像以指向私有自动化中心中可用的最小执行环境。

b. 定义额外的构建文件，以指向构建过程中要使用的任何断开连接的内容源。您的自定义 execution-environment.yml 文件应类似以下示例：

```
$ cat execution-environment.yml
---
```

```
version: 3

images:
  base_image:
    name: private-hub.example.com/ee-minimal-rhel8:latest

dependencies:
  python: requirements.txt
  galaxy: requirements.yml

additional_build_files:
  - src: files/ansible.cfg
    dest: configs
  - src: files/pip.conf
    dest: configs
  - src: files/hub-ca.crt
    dest: configs
  # uncomment if custom RPM repositories are required
  #- src: files/custom.repo
  #  dest: configs

additional_build_steps:
  prepend_base:
    # copy a custom pip.conf to override the location of the PyPI content
    - ADD _build/configs/pip.conf /etc/pip.conf
    # remove the default UBI repository definition
    - RUN rm -f /etc/yum.repos.d/ubi.repo
    # copy the hub CA certificate and update the trust store
    - ADD _build/configs/hub-ca.crt /etc/pki/ca-trust/source/anchors
    - RUN update-ca-trust
    # if needed, uncomment to add a custom RPM repository configuration
    #- ADD _build/configs/custom.repo /etc/yum.repos.d/custom.repo

  prepend_galaxy:
    - ADD _build/configs/ansible.cfg ~/.ansible.cfg

...
```

3. **在 files/ 子目录下创建一个 ansible.cfg 文件，指向您的私有自动化中心。**

```
$ cat files/ansible.cfg
[galaxy]
server_list = private_hub

[galaxy_server.private_hub]
url = https://private-hub.example.com/api/galaxy/
```

4. **在 files/ 子目录下创建一个 pip.conf 文件，指向内部 PyPI 镜像(Web 服务器或 Nexus)：**

```
$ cat files/pip.conf
[global]
```

```
index-url = https://<pypi_mirror_fqdn>/
trusted-host = <pypi_mirror_fqdn>
```

5. **可选：如果使用 bindep.txt 文件来添加 RPM 自定义执行环境，请在 files/ 子目录下创建一个 custom.repo 文件，指向断开连接的 Satellite 或托管 RPM 存储库的其他位置。如果此步骤是必需的，请取消与 custom.repo 文件对应的 execution-environment.yml 示例中的步骤。**

   以下示例是 UBI 仓库。也可以将其他本地仓库添加到此文件中。URL 路径可能需要根据镜像内容位于 web 服务器上的位置进行更改。

```
$ cat files/custom.repo
[ubi-8-baseos]
name = Red Hat Universal Base Image 8 (RPMs) - BaseOS
baseurl = http://<ubi_mirror_fqdn>/repos/ubi-8-baseos
enabled = 1
gpgkey = file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
gpgcheck = 1

[ubi-8-appstream]
name = Red Hat Universal Base Image 8 (RPMs) - AppStream
baseurl = http://<ubi_mirror_fqdn>/repos/ubi-8-appstream
enabled = 1
gpgkey = file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
gpgcheck = 1
```

6. **添加用于为私有自动化中心 web 服务器证书签名的 CA 证书。如果私有自动化中心使用安装程序提供的自签名证书：**

   a. **从您的私有自动化中心中复制 /etc/pulp/certs/pulp_webserver.crt，并将其命名为 hub-ca.crt。**

   b. **将 hub-ca.crt 文件添加到 files/ 子目录中。**

7. **如果私有自动化中心使用由证书颁发机构签名的用户提供的证书：**

   a. **复制该 CA 证书并将其命名为 hub-ca.crt。**

   b. **将 hub-ca.crt 文件添加到 files/ 子目录中。**

8. **完成预付步骤后，使用自定义执行环境镜像所需的内容创建 python requirements.txt 和**

Ansible 集合 requirements.yml 文件。

注意

任何所需的集合都必须已上传到您的私有自动化中心。

custom-ee/ 目录下应存在以下文件，其中 bindep.txt 和 files/custom.repo 是可选的：

```
$ cd $HOME/custom-ee
$ tree .
.
├── bindep.txt
├── execution-environment.yml
├── files
│   ├── ansible.cfg
│   ├── custom.repo
│   ├── hub-ca.crt
│   └── pip.conf
├── requirements.txt
└── requirements.yml

1 directory, 8 files
```

其他资源

如需有关版本 3 格式和要求的更多信息，请参阅 执行环境定义： 版本 3 格式。

4.12.3. 构建自定义执行环境

在创建新的自定义执行环境前，需要来自私有 hub 的 API 令牌来下载内容。

执行以下步骤生成令牌：

1. 登录到您的私有自动化中心。
2. 从左侧菜单中选择"集合"。

3. 在菜单的 "Collections" 部分下选择"API 令牌"。

4. 具有令牌后，设置以下环境变量以便 Ansible Builder 可以访问令牌：

```
$ export ANSIBLE_GALAXY_SERVER_PRIVATE_HUB_TOKEN=<your_token>
```

5. 使用以下命令创建自定义执行环境：

```
$ cd $HOME/custom-ee
$ ansible-builder build -f execution-environment.yml -t private-hub.example.com/custom-ee:latest -v 3
```

注意

如果构建失败，并显示私有 hub 证书由未知颁发机构签名的错误，您可以通过运行以下命令将所需的镜像拉取到本地镜像缓存中：

```
$ podman pull private-hub.example.com/ee-minimal-rhel8:latest --tls-verify=false
```

另外，您可以将私有 hub CA 证书添加到 podman 证书存储中：

```
$ sudo mkdir /etc/containers/certs.d/private-hub.example.com
$ sudo cp $HOME/custom-ee/files/hub-ca.crt /etc/containers/certs.d/private-hub.example.com
```

4.12.4. 将自定义执行环境上传到私有自动化中心

在新的执行环境镜像可用于自动化作业前，必须将其上传到私有自动化中心。

首先，验证执行环境镜像是否可以在本地 podman 缓存中看到：

```
$ podman images --format "table {{.ID}} {{.Repository}} {{.Tag}}"
IMAGE ID      REPOSITORY                    TAG
b38e3299a65e private-hub.example.com/custom-ee        latest
8e38be53b486 private-hub.example.com/ee-minimal-rhel8  latest
```

然后，登录到私有自动化中心的容器 registry 并推送镜像使其可用于作业模板和工作流：

```
$ podman login private-hub.example.com -u admin
Password:
Login Succeeded!
$ podman push private-hub.example.com/custom-ee:latest
```

4.13. 在 ANSIBLE AUTOMATION PLATFORM 次版本间升级

要在 Ansible Automation Platform 2 的次发行版本间升级，请使用这个常规工作流。

流程

1. 下载并解压缩最新的 Ansible Automation Platform 2 安装捆绑包。
2. 创建现有安装的备份。
3. 将现有安装清单文件复制到新的安装捆绑包目录中。
4. 运行 ./setup.sh 以升级安装。

例如，要从 2.2.0-7 升级到 2.3-1.2，请确保两个设置捆绑包都位于安装发生的初始控制器节点上：

```
 $ ls -1F
ansible-automation-platform-setup-bundle-2.2.0-7/
ansible-automation-platform-setup-bundle-2.2.0-7.tar.gz
ansible-automation-platform-setup-bundle-2.3-1.2/
ansible-automation-platform-setup-bundle-2.3-1.2.tar.gz
```

备份 2.2.0-7 安装：

```
$ cd ansible-automation-platform-setup-bundle-2.2.0-7
$ sudo ./setup.sh -b
$ cd ..
```

将 2.2.0-7 清单文件复制到 2.3-1.2 捆绑包目录中：

```
$ cd ansible-automation-platform-setup-bundle-2.2.0-7
$ cp inventory ../ansible-automation-platform-setup-bundle-2.3-1.2/
$ cd ..
```

使用 setup.sh 脚本从 2.2.0-7 升级到 2.3-1.2 ：

```
$ cd ansible-automation-platform-setup-bundle-2.3-1.2
$ sudo ./setup.sh
```

附录 A. 清单文件变量

下表包含 Ansible 安装清单文件中使用的预定义变量的信息。并非所有这些变量都是必需的。

A.1. 常规变量

变量	描述
enable_insights_collection	如果节点使用 Subscription Manager 注册，则默认安装会将节点注册到 Red Hat Ansible Automation Platform Service 中。设置为 **False** 以禁用。 默认为 **true**
nginx_user_http_config	在 http 部分下，列出 **/etc/nginx/nginx.conf** 的 nginx 配置列表。 列表中的每一元素作为单独的行提供给 **http nginx** 配置中。 默认为空列表
registry_password	只有在使用非捆绑包安装程序时，才需要 **registry_password**。 用于访问 **registry_url** 的密码凭证。 用于 **[automationcontroller]** 和 **[automationhub]** 组。 在 **registry_username** 和 **registry_password** 中输入 Red Hat Registry Service account 凭证以链接到 Red Hat 容器 registry。 当 **registry_url** 为 **registry.redhat.io** 时，如果没有使用捆绑包安装程序，则需要用户名和密码。
registry_url	用于 **[automationcontroller]** 和 **[automationhub]** 组。 默认为 **registry.redhat.io**。

变量	描述
registry_username	只有在使用非捆绑包安装程序时，才需要 **registry_username**。 用于访问 **registry_url** 的用户权限。 用于 **[automationcontroller]** 和 **[automationhub]** 组，但只有 **registry_url** 的值为 **registry.redhat.io** 时才使用。 在 **registry_username** 和 **registry_password** 中输入 Red Hat Registry Service account 凭证以链接到 Red Hat 容器 registry。
routable_hostname	如果运行安装程序的机器只能通过特定 URL 路由到目标主机，例如，如果您的清单中使用短名称，但运行安装程序的节点只能解析使用 FQDN 的主机时，使用 **routable hostname**。 如果没有设置 **routable_hostname**，它应默认为 **ansible_host**。如果没有设置 **ansible_host**，则**inventory_hostname** 将用作最后的手段。 此变量用作特定主机的主机变量，而不是在 **[all:vars]** 部分下。如需更多信息，请参阅 将变量分配给一个 machine:host 变量。

A.2. ANSIBLE AUTOMATION HUB 变量

变量	描述
automationhub_admin_password	必填 当密码在清单文件中以纯文本形式提供时，密码必须用引号括起。
automationhub_api_token	如果从 Ansible Automation Platform 2.0 或更早版本升级，您必须： - 提供一个现有的 Ansible Automation hub 令牌作为 **automationhub_api_token** 或 - 将 **generate_automationhub_token** 设置为 **true** 来生成新令牌 生成新令牌会导致现有令牌无效。

变量	描述
automationhub_authentication_backend	默认不设置此变量。将它设置为 **ldap** 以使用 LDAP 身份验证。 当它被设置为 **ldap** 时，还必须设置以下变量： ● **automationhub_ldap_server_uri** ● **automationhub_ldap_bind_dn** ● **automationhub_ldap_bind_password** ● **automationhub_ldap_user_search_base_dn** ● **automationhub_ldap_group_search_base_dn** 如果缺少其中任何一个，则安装将停止。
automationhub_auto_sign_collections	如果启用了集合签名服务，则集合默认不会自动签名。 将此参数设置为 **true** 默认禁用它们。 默认为 **false**。
automationhub_backup_collections	*可选* Ansible Automation hub 在 **/var/lib/pulp** 中提供工件。自动化控制器默认自动备份工件。 您还可以将 **automationhub_backup_collections** 设置为 false，备份/恢复进程不会备份或恢复 **/var/lib/pulp**。 默认为 **true**。
automationhub_collection_download_count	*可选* 决定是否在 UI 中显示下载计数。 默认为 **false**。

变量	描述
automationhub_collection_seed_repository	运行捆绑包安装程序时，验证的内容将上传到 **经过验证的** 存储库，认证的内容将上传到 **rh-certified** 存储库。 默认情况下，经过认证和验证的内容都会被上传。 此变量的可能值为 'certified' 或 'validated'。 如果您不想安装内容，请将 **automationhub_seed_collections** 设置为 **false** 来禁用 seeding。 如果您只需要一种类型的内容，请将 **automationhub_seed_collections** 设置为 **true**，将 **automationhub_collection_seed_repository** 设置为您要包含的内容类型。
automationhub_collection_signing_service_key	如果启用了集合签名服务，您必须提供此变量，以确保可以正确签名集合。 **/absolute/path/to/key/to/sign**
automationhub_collection_signing_service_script	如果启用了集合签名服务，您必须提供此变量，以确保可以正确签名集合。 **/absolute/path/to/script/that/signs**
automationhub_create_default_collection_signing_service	将此变量设置为 true 以创建集合签名服务。 默认为 **false**。
automationhub_container_signing_service_key	如果启用了容器签名服务，您必须提供此变量，以确保容器可以被正确签名。 **/absolute/path/to/key/to/sign**
automationhub_container_signing_service_script	如果启用了容器签名服务，您必须提供此变量，以确保容器可以被正确签名。 **/absolute/path/to/script/that/signs**
automationhub_create_default_container_signing_service	将此变量设置为 true 以创建容器签名服务。 默认为 **false**。

变量	描述
automationhub_disable_hsts	默认安装部署启用了 TLS 的 Ansible Automation hub。如果您部署启用了 *HTTP Strict Transport Security* (HSTS) web-security 策略的自动化中心，请使用此变量。此变量禁用 HSTS web-security 策略机制。 默认为 **false**。
automationhub_disable_https	*可选* 如果 Ansible Automation hub 部署启用了 HTTPS。 默认为 **false**。
automationhub_enable_api_access_log	当设置为 **true** 时，此变量会在 **/var/log/galaxy_api_access.log** 中创建一个日志文件，该文件会记录提供给平台的所有用户操作，包括其用户名和 IP 地址。 默认为 **false**。
automationhub_enable_analytics	指明是否为 Ansible Automation Platform 2.4 中自动化中心中使用的 pulpcore 版本启用 Pulp 分析的布尔值。 要启用 pulp 分析，请将 **automationhub_enable_analytics** 设置为 true。 默认为 **false**。
automationhub_enable_unauthenticated_collection_access	将此变量设置为 true，使未授权用户能够查看集合。 默认为 **false**。
automationhub_enable_unauthenticated_collection_download	将此变量设置为 true，使未授权用户能够下载集合。 默认为 **false**。
automationhub_importer_settings	*可选* 传递给 galaxy-importer 的设置字典。 在导入时，集合可以通过一系列检查。 行为由 **galaxy-importer.cfg** 配置驱动。 例如 **ansible-doc**、**ansible-lint** 和 **flake8**。 这个参数可让您驱动此配置。

变量	描述
automationhub_main_url	客户端连接到的主要自动化中心 URL。 例如： https://<load balancer host>。 如果要在自动化中心环境中实施 Red Hat Single Sign-On，使用 **automationhub_main_url** 指定客户端连接到的主自动化中心 URL。 如果没有指定，则使用 **[automationhub]** 组中的第一个节点。
automationhub_pg_database	*必需* 数据库名称。 默认为 **automationhub**。
automationhub_pg_host	如果不使用内部数据库，则需要此项。 Automation Hub 使用的远程 PostgreSQL 数据库的主机名。 默认 = **127.0.0.1**。
automationhub_pg_password	Automation hub PostgreSQL 数据库的密码。 对 **automationhub_pg_password** 使用特殊字符是有限的。支持 **!**, **#**, **0** 和 **@** 字符。使用其他特殊字符可能会导致设置失败。
automationhub_pg_port	如果不使用内部数据库，则需要此项。 默认 = 5432。
automationhub_pg_sslmode	必需。 默认为 **prefer**。
automationhub_pg_username	必填 默认为 **automationhub**。

变量	描述
automationhub_require_content_approval	*可选* 如果自动化中心在集合可用前强制实施批准机制，则值为 **true**。 默认情况下，当您将集合上传到自动化中心时，管理员必须批准它，然后才能供用户使用。 如果要禁用内容批准流，请将变量设置为 **false**。 默认为 **true**。
automationhub_seed_collections	定义是否启用预加载的布尔值。 运行捆绑包安装程序时，验证的内容将上传到 **经过验证的** 存储库，认证的内容将上传到 **rh-certified** 存储库。 默认情况下，经过认证和验证的内容都会被上传。 如果您不想安装内容，请将 **automationhub_seed_collections** 设置为 **false** 来禁用 seeding。 如果您只需要一种类型的内容，请将 **automationhub_seed_collections** 设置为 **true**，将 **automationhub_collection_seed_repository** 设置为您要包含的内容类型。 默认为 **true**。
automationhub_ssl_cert	*可选* **/path/to/automationhub.cert** 与 **web_server_ssl_cert** 相同，但用于自动化中心 UI 和 API。
automationhub_ssl_key	*可选* **/path/to/automationhub.key**. 与 **web_server_ssl_key** 相同，但用于自动化 hub UI 和 API
automationhub_ssl_validate_certs	对于 Red Hat Ansible Automation Platform 2.2 及更新的版本，不再使用这个值。 如果自动化中心因为默认设置而在请求自身时必须验证证书，则将值设为 **true**，Ansible Automation Platform 会使用自签名证书进行部署。 默认为 **false**。

变量	描述
automationhub_upgrade	**已弃用** 对于 Ansible Automation Platform 2.2.1 及更新的版本，这个值的值已固定为 **true**。 Automation hub 始终使用最新的软件包进行更新。
automationhub_user_headers	Ansible Automation hub web 服务器的 nginx 标头列表。 列表中的每一元素作为单独的行提供给 web 服务器的 nginx 配置。 默认为空列表
ee_from_hub_only	当使用自动化中心部署时，安装程序会将执行环境镜像推送到自动化中心，并将自动化控制器配置为从自动化中心 registry 中拉取镜像。 要使自动化中心唯一的 registry 从中拉取执行环境镜像，请将此变量设置为 **true**。 如果设置为 **false**，则也可直接从红帽获取执行环境镜像。 使用捆绑包安装程序时默认为 **true**。
generate_automationhub_token	如果从 Red Hat Ansible Automation Platform 2.0 或更早版本升级，请选择以下选项之一： ● 提供一个现有的 Ansible Automation hub 令牌作为 **automationhub_api_token** ● 将 **generate_automationhub_token** 设置为 **true** 来生成新令牌。生成新令牌会导致现有令牌无效。
nginx_hsts_max_age	此变量指定系统应被视为 *HTTP Strict Transport Security* (HSTS)主机的时间（以秒为单位）。也就是 HTTPS 专用于通信的时间。 默认为 63072000 秒，或两年。

变量	描述
nginx_tls_protocols	定义 Nginx 中对 **ssl_protocols** 的支持。 可用 **TLSv1、 TLSv1.1、'TLSv1.2、TLSv1.3** 只有在使用 OpenSSL 1.0.1 或更高版本时，TLSv1.1 和 TLSv1.2 参数才可以正常工作。 TLSv1.3 参数仅在使用 OpenSSL 1.1.1 或更高版本时才有效。 如果 **nginx_tls-protocols = ['TLSv1.3']** 仅启用 TLSv1.3。To set more than one protocol use **nginx_tls_protocols = ['TLSv1.2', 'TLSv.1.3']** 默认为 **TLSv1.2**。
pulp_db_fields_key	要导入的 Fernet 对称加密密钥的相对或绝对路径。该路径位于 Ansible 管理节点上。它用于加密数据库中的某些字段，如凭证。如果未指定，将生成一个新密钥。
sso_automation_platform_login_theme	*可选* 用于 Ansible Automation Platform 管理的和外部管理的 Red Hat Single Sign-On。 主题文件所在的目录的路径。如果更改此变量，您必须提供自己的主题文件。 默认 = **ansible-automation-platform**。
sso_automation_platform_realm	*可选* 用于 Ansible Automation Platform 管理的和外部管理的 Red Hat Single Sign-On。 SSO 中域的名称。 默认 = **ansible-automation-platform**。
sso_automation_platform_realm_displayname	*可选* 用于 Ansible Automation Platform 管理的和外部管理的 Red Hat Single Sign-On。 显示域的名称。 默认 = **Ansible Automation Platform**。

变量	描述
sso_console_admin_username	*可选* 用于 Ansible Automation Platform 管理的和外部管理的 Red Hat Single Sign-On。 SSO 管理用户名。 默认为 **admin**。
sso_console_admin_password	*必需* 用于 Ansible Automation Platform 管理的和外部管理的 Red Hat Single Sign-On。 SSO 管理密码。
sso_custom_keystore_file	*可选* 仅用于 Ansible Automation Platform 管理的 Red Hat Single Sign-On。 为 SSO 提供客户提供的密钥存储。
sso_host	*必需* 仅用于外部管理的 Red Hat Single Sign-On。 自动化中心需要 SSO 和 SSO 管理凭据进行身份验证。 如果清单中未提供用于配置的 SSO，则必须使用此变量来定义 SSO 主机。
sso_keystore_file_remote	*可选* 仅用于 Ansible Automation Platform 管理的 Red Hat Single Sign-On。 如果客户提供的密钥存储位于远程节点上，则设置为 **true**。 默认为 **false**。
sso_keystore_name	*可选* 仅用于 Ansible Automation Platform 管理的 Red Hat Single Sign-On。 SSO 的密钥存储名称。 默认 = **ansible-automation-platform**。

变量	描述
sso_keystore_password	启用 HTTPS 的 SSO 的密钥存储密码。 在使用 Ansible Automation Platform 管理的 SSO 以及启用 HTTPS 时，需要此项。默认安装使用 **sso_use_https=true** 部署 SSO。
sso_redirect_host	*可选* 用于 Ansible Automation Platform 管理的和外部管理的 Red Hat Single Sign-On。 如果设置了 **sso_redirect_host**，应用程序会使用它来连接到 SSO 以进行身份验证。 这必须可从客户端机器访问。
sso_ssl_validate_certs	*可选* 用于 Ansible Automation Platform 管理的和外部管理的 Red Hat Single Sign-On。 如果证书必须在连接过程中验证，则设置为 **true**。 默认为 **true**。
sso_use_https	*可选* 如果单点登录使用 HTTPS，则用于 Ansible Automation Platform 管理和外部管理的 Red Hat Single Sign-On。 默认为 **true**。

要使 Ansible 自动化中心直接连接到 LDAP，您必须配置以下变量：可以使用 ldap_extra_settings 变量传递的额外 LDAP 相关变量列表，请参阅 Django 参考文档。

变量	描述
automationhub_ldap_bind_dn	使用 **automationhub_ldap_bind_password** 绑定到 LDAP 服务器时使用的名称。 在将私有自动化中心与 LDAP 集成时，必须设置，否则安装将失败。

变量	描述
automationhub_ldap_bind_password	*必需* 与 **automationhub_ldap_bind_dn** 搭配使用的密码。 必须在集成私有自动化中心 LDAP 时设置，否则安装将失败。
automationhub_ldap_group_search_base_dn	一个 LDAP 搜索对象，用于查找用户可能属于的所有 LDAP 组。 如果您的配置对 LDAP 组有任何引用，您必须设置此变量和 **automationhub_ldap_group_type**。 在将私有自动化中心与 LDAP 集成时，必须设置，否则安装将失败。 默认为 **None**
automationhub_ldap_group_search_filter	*可选* 用于查找组成员资格的搜索过滤器。 变量标识了使用自动化中心和 LDAP 映射组要使用哪个 flavor 类型。用于通过 LDAP 安装自动化中心。 默认为 **(objectClass=Group)**
automationhub_ldap_group_search_scope	*可选* 范围使用 django 框架用于 LDAP 身份验证在 LDAP 树中搜索组。用于通过 LDAP 安装自动化中心。 默认为 **SUBTREE**
automationhub_ldap_group_type	描述 automationhub_ldap_group_search 返回的组类型。 这基于 automationhub_ldap_group_type_params 和 automationhub_ldap_group_type_class 的值动态设置，否则它是来自 django-ldap 的默认值，它是 'None' Default = **django_auth_ldap.config:GroupOfNamesType**

变量	描述
automationhub_ldap_group_type_class	*可选* django-ldap 组类型类的可导入路径。 变量标识在 django 框架内用于 LDAP 身份验证的组类型。用于通过 LDAP 安装自动化中心。 Default =**django_auth_ldap.config:GroupOfNamesType**
automationhub_ldap_server_uri	LDAP 服务器的 URI。 使用底层 LDAP 库支持的任何 URI。 必须在集成私有自动化中心 LDAP 时设置，否则安装将失败。
automationhub_ldap_user_search_base_dn	在目录中查找用户的 LDAP 搜索对象。filter 参数必须包含用户名的占位符 %(user)。它必须完全返回一个结果才能成功进行身份验证。 在将私有自动化中心与 LDAP 集成时，必须设置，否则安装将失败。
automationhub_ldap_user_search_filter	*可选* 默认为 **'(uid=%(user))'**
automationhub_ldap_user_search_scope	*可选* 范围通过使用用于 LDAP 身份验证的 django 框架在 LDAP 树中搜索用户。用于通过 LDAP 安装自动化中心。 默认为 **SUBTREE**

A.3. 自动化控制器变量

变量	描述
admin_password	安装完成后，用于访问 UI 的管理用户密码。 当密码在清单文件中以纯文本形式提供时，密码必须用引号括起。
automation_controller_main_url	对于 SSO 配置所需的替代前端 URL，请提供 URL。

变量	描述
automationcontroller_password	自动化控制器实例的密码。 当密码在清单文件中以纯文本形式提供时，密码必须用引号括起。
automationcontroller_username	自动化控制器实例的用户名。
nginx_http_port	nginx HTTP 服务器侦听入站连接。 默认为 80
nginx_https_port	nginx HTTPS 服务器侦听安全连接。 默认为 443
nginx_hsts_max_age	此变量指定系统必须被视为 *HTTP Strict Transport Security* (HSTS)主机的时间（以秒为单位）。也就是 HTTPS 专用于通信的时间。 默认为 63072000 秒，或两年。
nginx_tls_protocols	定义 Nginx 中对 **ssl_protocols** 的支持。 可用 **TLSv1、 TLSv1.1、'TLSv1.2、TLSv1.3** 只有在使用 OpenSSL 1.0.1 或更高版本时，TLSv1.1 和 TLSv1.2 参数才可以正常工作。 TLSv1.3 参数仅在使用 OpenSSL 1.1.1 或更高版本时才有效。 如果 **nginx_tls-protocols = ['TLSv1.3']** 仅启用 TLSv1.3。To set more than one protocol use **nginx_tls_protocols = ['TLSv1.2', 'TLSv.1.3']** 默认为 **TLSv1.2**。
nginx_user_headers	自动化控制器 web 服务器的 nginx 标头列表。 列表中的每一元素作为单独的行提供给 web 服务器的 nginx 配置。 默认为空列表
node_state	*可选* 节点或一组节点的状态。有效选项为 **active**，**取消置备** 从集群中删除节点，或 **iso_migrate** 将旧的隔离节点迁移到执行节点。 默认为 **active**。

变量	描述
node_type	对于 **[automationcontroller]** 组。 可以为这个组分配两个有效的 **node_types**。 **node_type=control** 表示节点仅运行项目和清单更新，但不运行常规作业。 **node_type=hybrid** 可以运行所有内容。 这个组的默认 = **hybrid** 对于 **[execution_nodes]** 组： 可以为这个组分配两个有效的 **node_types**。 **node_type=hop** 表示节点将作业转发到执行节点。 **node_type=execution** 表示节点可以运行作业。 此组的默认值为 **execution**。
peers	*可选* **peers** 变量用于指示特定主机或组连接到哪些节点。只要定义了此变量，都会建立到特定主机或组的出站连接。 此变量用于在 **receptor.conf** 文件中添加用于与其他节点建立网络连接的 **tcp-peer** 条目。 peers 变量可以是清单中以逗号分隔的主机和组的列表。这被解析为用来构造 **receptor.conf** 文件的一组主机。
pg_database	postgreSQL 数据库的名称。 默认为 **awx**。
pg_host	postgreSQL 主机，可以是外部管理的数据库。

变量	描述
pg_password	postgreSQL 数据库的密码。 在 **pg_password** 中使用特殊字符是有限的。支持 **!**, **#**, **0** 和 **@** 字符。使用其他特殊字符可能会导致设置失败。 注意 安装时，您将不再需要在清单文件中提供 **apg_hashed_password**，因为 PostgreSQL 13 现在可以更加安全地存储用户的密码。 当您在清单文件中为安装程序提供 **pg_password** 时，PostgreSQL 使用 SCRAM-SHA-256 哈希在安装过程中保护该密码。
pg_port	要使用的 postgreSQL 端口。 默认 = 5432
pg_ssl_mode	选择两种可用模式之一： **prefer** 和 **verify-full**。 设置为 **verify-full**，用于客户端强制执行 SSL。 默认为 **prefer**。
pg_username	您的 postgreSQL 数据库用户名。 默认为 **awx**。
postgres_ssl_cert	postgreSQL SSL 证书的位置。 **/path/to/pgsql_ssl.cert**
postgres_ssl_key	postgreSQL SSL 密钥的位置。 **/path/to/pgsql_ssl.key**
postgres_use_cert	postgreSQL 用户证书的位置。 **/path/to/pgsql.crt**
postgres_use_key	postgreSQL 用户密钥的位置。 **/path/to/pgsql.key**
postgres_use_ssl	如果 postgreSQL 使用 SSL，则使用此变量。

变量	描述
postgres_max_connections	如果您使用安装程序管理的 postgreSQL，则应用的最大数据库连接设置。 如需帮助选择值，请参阅自动化控制器管理指南中的 PostgreSQL 数据库配置。 对于基于虚拟机的安装，默认为 200 （单个节点）和 1024 （集群）。
receptor_listener_port	用于 receptor 连接的端口。 默认 = 27199
supervisor_start_retry_count	指定后，它会将 **startretries = <value specified>** 添加到 supervisor 配置文件 (/etc/supervisord.d/tower.ini)中。 有关 **startretries** 的更多信息，请参阅 program:x section 值。 不存在默认值。
web_server_ssl_cert	*可选* **/path/to/webserver.cert** 与 **automationhub_ssl_cert** 相同，但用于 Web 服务器 UI 和 API。
web_server_ssl_key	*可选* **/path/to/webserver.key** 与 **automationhub_server_ssl_key** 相同，但用于 Web 服务器 UI 和 API。

A.4. ANSIBLE 变量

以下变量控制 Ansible Automation Platform 与远程主机交互的方式。

有关特定于特定插件的变量的更多信息，请参阅 Ansible.Builtin 的文档。

有关全局配置选项列表，请参阅 Ansible 配置设置。

变量	描述
ansible_connection	用于目标主机上任务的连接插件。 这可以是任何 Ansible 连接插件的名称。SSH 协议类型是 **smart**、**ssh** 或 **paramiko**。 默认为 **smart**
ansible_host	要使用的目标主机的 ip 或名称，而不是 **inventory_hostname**。
ansible_port	连接端口号。 默认：ssh 的 22 个
ansible_user	连接到主机时使用的用户名。
ansible_password	用于向主机进行身份验证的密码。 从不以纯文本形式存储此变量。 始终使用密码库。
ansible_ssh_private_key_file	SSH 使用的私钥文件。在使用多个密钥且您不想使用 SSH 代理时很有用。
ansible_ssh_common_args	此设置始终附加到 **sftp**、**scp** 和 **ssh** 的默认命令行中。对于为特定主机或组配置 ProxyCommand 非常有用。
ansible_sftp_extra_args	此设置始终附加到默认的 **sftp** 命令行。
ansible_scp_extra_args	此设置始终附加到默认的 **scp** 命令行。
ansible_ssh_extra_args	此设置始终附加到默认的 **ssh** 命令行。
ansible_ssh_pipelining	确定是否使用 SSH pipelining。这可以覆盖 **ansible.cfg** 中的 pipelining 设置。如果使用基于 SSH 密钥的身份验证，密钥必须由 SSH 代理管理。
ansible_ssh_executable	在版本 2.2 中添加。 此设置覆盖使用系统 SSH 的默认行为。这可以覆盖 **ansible.cfg** 中的 ssh_executable 设置。

变量	描述
ansible_shell_type	目标系统的 shell 类型。除非将 **ansible_shell_executable** 设置为一个非兼容 shell，否则不要使用此设置。默认情况下，使用 sh 样式的语法对命令进行了格式化。把它设置为 **csh** 或 **fish** 会导致在目标系统上执行命令来遵循这些 shell 的语法。
ansible_shell_executable	这将设置 Ansible 控制器在目标机器上使用的 shell，并覆盖 **ansible.cfg** 中的可执行文件，默认为 **/bin/sh**。 不要更改此变量，除非目标计算机上没有安装 **/bin/sh**，否则无法从 sudo 运行。
inventory_hostname	此变量从清单脚本或 Ansible 配置文件获取机器的主机名。 您不能设置此变量的值。 由于从配置文件中获取了该值，实际运行时主机名值可能与此变量返回的不同。

A.5. EVENT-DRIVEN ANSIBLE 控制器变量

变量	描述
automationedacontroller_admin_password	Event-Driven Ansible 控制器实例使用的管理员密码。 当密码在清单文件中以纯文本形式提供时，密码必须用引号括起。
automationedacontroller_admin_username	django 用来识别并创建 Event-Driven Ansible 控制器的 admin 超级用户的用户名。 默认 = **admin**
automationedacontroller_admin_email	django 用于 Event-Driven Ansible 控制器的 admin 用户的电子邮件地址。 Default = **admin@example.com**
automationedacontroller_allowed_hostnames	启用对 Event-Driven Ansible 控制器访问权限的附加地址列表。 默认为空列表

变量	描述
automationedacontroller_controller_verify_ssl	在从 Event-Driven Ansible 控制器调用时验证自动化控制器 Web 证书的布尔值标志。验证为 **true** ；未验证为 **false**。 默认为 **false**
automationedacontroller_disable_https	禁用 HTTPS Event-Driven Ansible 控制器的布尔值标志。 默认为 **false**
automationedacontroller_disable_hsts	用于禁用 HSTS Event-Driven Ansible 控制器的布尔值标志。 默认为 **false**
automationedacontroller_gunicorn_workers	通过 gunicorn 提供的 API 的 worker 数量。 默认 =(dm of cores or threads)* 2 + 1
automationedacontroller_max_running_activations	每个节点同时运行的最大激活数。 这是一个必须大于 0 的整数。 默认 = 12
automationedacontroller_nginx_tls_files_remote	布尔值标志，用于指定证书源是否在远程主机 (true) 还是本地 (false)。 默认为 **false**
automationedacontroller_pg_database	Event-Driven Ansible 控制器使用的 Postgres 数据库。 默认为 **automtionedacontroller**。
automationnedacontroller_pg_host	Event-Driven Ansible 控制器使用的 Postgres 数据库的主机名，可以是外部管理的数据库。
automationedacontroller_pg_password	Event-Driven Ansible 控制器使用的 Postgres 数据库的密码。 对 **automationedacontroller_pg_password** 使用特殊字符是有限的。支持 **!**, **#**, **0** 和 **@** 字符。使用其他特殊字符可能会导致设置失败。

变量	描述
automationedacontroller_pg_port	Event-Driven Ansible 控制器使用的 Postgres 数据库的端口号。 默认 = **5432**。
automationedacontroller_pg_username	Event-Driven Ansible 控制器 Postgres 数据库的用户名。 默认为 **automationedacontroller**。
automationedacontroller_rq_workers	Event-Driven Ansible 控制器使用的 Redis Queue (RQ) worker 数量。RQ worker 是在后台运行的 Python 进程。 默认 =(dm of cores or threads)* 2 + 1
automationedacontroller_ssl_cert	*可选* **/root/ssl_certs/eda.*<example>*.com.crt** 与 **automationhub_ssl_cert** 相同，但用于 Event-Driven Ansible 控制器 UI 和 API。
automationedacontroller_ssl_key	*可选* **/root/ssl_certs/eda.*<example>*.com.key** 与 **automationhub_server_ssl_key** 相同，但用于 Event-Driven Ansible 控制器 UI 和 API。
automationedacontroller_user_headers	要添加到 Event-Driven Ansible 控制器 nginx 配置的额外 nginx 标头列表。 默认为空列表