



# Red Hat Ansible Automation Platform 2.4

## 创建和使用执行环境

使用 Ansible Builder 创建和使用执行环境



使用 Ansible Builder 创建和使用执行环境

## 法律通告

Copyright © 2024 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux<sup>®</sup> is the registered trademark of Linus Torvalds in the United States and other countries.

Java<sup>®</sup> is a registered trademark of Oracle and/or its affiliates.

XFS<sup>®</sup> is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL<sup>®</sup> is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js<sup>®</sup> is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack<sup>®</sup> Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

## 摘要

本指南演示了如何为 Red Hat Ansible Automation Platform 创建一致且可重复生成的自动化执行环境。

---

# 目录

前言 .....	3
对红帽文档提供反馈 .....	4
<b>第 1 章 自动化执行环境简介 .....</b>	<b>5</b>
1.1. 关于自动化执行环境 .....	5
<b>第 2 章 使用 ANSIBLE BUILDER .....</b>	<b>6</b>
2.1. 为什么使用 ANSIBLE BUILDER? .....	6
2.2. 安装 ANSIBLE BUILDER .....	6
2.3. 构建定义文件 .....	6
2.4. 构建自动化执行环境镜像 .....	8
2.5. 定义文件内容的分类 .....	8
2.6. 可选的 BUILD 命令参数 .....	12
2.7. CONTAINERFILE .....	12
2.8. 在不构建镜像的情况下创建容器文件 .....	12
<b>第 3 章 常见自动化执行环境场景 .....</b>	<b>13</b>
3.1. 更新自动化中心 CA 证书 .....	13
3.2. 在构建自动化执行环境时使用自动化中心身份验证详情 .....	13
3.3. 其他资源 .....	14
<b>第 4 章 发布自动化执行环境 .....</b>	<b>15</b>
4.1. 自定义现有自动化执行环境镜像 .....	15
4.2. 其他资源（或后续步骤） .....	17
<b>第 5 章 填充私有自动化中（AUTOMATION HUB）容器 REGISTRY .....</b>	<b>18</b>
5.1. 拉取 (PULL) 用于自动化中心的镜像 .....	18
5.2. 用于自动化中心的标记镜像 .....	19
5.3. 将容器镜像推送到私有自动化中心 .....	20
<b>第 6 章 设置容器存储库 .....</b>	<b>21</b>
6.1. 设置容器 REGISTRY 的先决条件 .....	21
6.2. 将 README 添加到容器存储库中 .....	21
6.3. 提供对容器存储库的访问 .....	21
6.4. 标记容器镜像 .....	22
6.5. 在自动化控制器中创建凭证 .....	22
<b>第 7 章 从容器存储库中拉取镜像 .....</b>	<b>24</b>
7.1. 拉取镜像 .....	24
7.2. 从容器存储库同步镜像 .....	24
<b>附录 A. 自动化执行环境优先级 .....</b>	<b>26</b>



## 前言

使用 Ansible Builder 为您的 Red Hat Ansible Automation Platform 需求创建一致且可重复生成的自动化环境。

## 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 自动化执行环境简介

使用依赖于非默认依赖项的 Ansible 内容可能会很复杂，因为必须在每个节点上安装软件包，并与主机系统上安装的其他软件进行交互，并保持同步。

自动化执行环境有助于简化这个过程，并可使用 Ansible Builder 轻松创建。

## 1.1. 关于自动化执行环境

Red Hat Ansible Automation Platform 中的所有自动化在称为自动化执行环境的容器镜像上运行。自动化执行环境创建用于传达自动化依赖项的通用语言，并提供构建和分发自动化环境的标准方法。

自动化执行环境应包含以下内容：

- Ansible Core 2.15 或更高版本
- Python 3.8-3.11
- Ansible Runner
- Ansible 内容集合及其依赖项
- 系统依赖项

### 1.1.1. 为什么使用自动化执行环境？

使用自动化执行环境时，Red Hat Ansible Automation Platform 已将 control plane 与 execution plane 分开来转换为分布式架构。与 control plane 独立进行自动化执行可加快开发周期，并提高跨环境的可扩展性、可靠性和可移植性。Red Hat Ansible Automation Platform 还包括对 Ansible 内容工具的访问，方便构建和管理自动化执行环境。

除了速度、可移植性和灵活性外，自动化执行环境还提供以下好处：

- 它们确保自动化在多个平台中持续运行，并能够纳入系统级依赖项和基于集合的内容。
- 它们可让 Red Hat Ansible Automation Platform 管理员提供和管理自动化环境以满足不同团队的需求。
- 它们通过提供构建和分发自动化环境的一种标准方式，在团队间轻松扩展和共享自动化。
- 它们使自动化团队能够自行定义、构建和更新其自动化环境。
- 自动化执行环境提供通用语言来传达自动化依赖项。

## 第 2 章 使用 ANSIBLE BUILDER

Ansible Builder（Ansible 构建器）是一个命令行工具，它通过使用各种 Ansible Collections 中定义的或由用户创建的元数据自动构建自动化执行环境的过程。

### 2.1. 为什么使用 ANSIBLE BUILDER？

在开发 Ansible Builder 之前，Red Hat Ansible Automation Platform 用户可以在创建自定义虚拟环境或容器（包含所有必要的依赖项）时遇到依赖项问题和错误。

现在，使用 Ansible Builder 时，您可以轻松地创建一个可自定义的自动化执行环境定义文件，该文件指定自动化执行环境中包含的内容，如 Ansible Core、Python、Collections、第三方 Python 要求和系统级别软件包。这可让您满足所有必要的要求和依赖项来获取作业运行。

### 2.2. 安装 ANSIBLE BUILDER

#### 先决条件

- 已安装 Podman 容器运行时。
- 您已在主机上附加了有效的订阅。这样，您可以访问安装 **ansible-builder** 所需的仅订阅资源，并确保自动启用 **ansible-builder** 所需的存储库。如需更多信息，请参阅附加 [Red Hat Ansible Automation Platform 订阅](#)。

#### 流程

1. 在终端中，运行以下命令来激活 Ansible Automation Platform 仓库：

```
# dnf install --enablerepo=ansible-automation-platform-2.4-for-rhel-9-x86_64-rpms ansible-builder
```

### 2.3. 构建定义文件

安装 Ansible Builder 后，您可以创建一个定义文件，Ansible Builder 用来创建自动化执行环境镜像。Ansible Builder 通过读取和验证定义文件来创建自动化执行环境镜像，然后创建一个 **Containerfile**，最后将 **Containerfile** 传递给 Podman，然后打包并创建自动化执行环境镜像。您创建的定义文件必须采用 **yaml** 格式，并包含不同的部分。默认定义文件名（如果未提供）是 **execution-environment.yml**。有关定义文件的部分的更多信息，请参阅 [定义文件内容的明细](#)。

以下是版本 3 定义文件的示例。每个定义文件必须指定其使用的 Ansible Builder 功能集的主版本号。如果没有指定，Ansible Builder 默认为版本 1，使大多数新功能和定义关键字不可用。

#### 例 2.1. 定义文件示例

```
version: 3

build_arg_defaults: ❶
  ANSIBLE_GALAXY_CLI_COLLECTION_OPTS: '--pre'

dependencies: ❷
  galaxy: requirements.yml
  python:
```

```

- six
- psutil
system: bindep.txt

images: ❸
base_image:
  name: registry.redhat.io/ansible-automation-platform-24/ee-minimal-rhel9:latest

# Custom package manager path for the RHEL based images
options: ❹
package_manager_path: /usr/bin/microdnf

additional_build_steps: ❺
prepend_base:
  - RUN echo This is a prepend base command!

prepend_galaxy:
  # Environment variables used for Galaxy client configurations
  - ENV ANSIBLE_GALAXY_SERVER_LIST=automation_hub
  - ENV
ANSIBLE_GALAXY_SERVER_AUTOMATION_HUB_URL=https://console.redhat.com/api/automati
on-hub/content/xxxxxxx-synclist/
  - ENV
ANSIBLE_GALAXY_SERVER_AUTOMATION_HUB_AUTH_URL=https://sso.redhat.com/auth/real
ms/redhat-external/protocol/openid-connect/token
  # define a custom build arg env passthru - we still also have to pass
  # `--build-arg ANSIBLE_GALAXY_SERVER_AUTOMATION_HUB_TOKEN` to get it to pick it
up from the env
  - ARG ANSIBLE_GALAXY_SERVER_AUTOMATION_HUB_TOKEN

prepend_final: |
  RUN whoami
  RUN cat /etc/os-release
append_final:
  - RUN echo This is a post-install command!
  - RUN ls -la /etc

```

- ❶ 列出构建参数的默认值。
- ❷ 指定各种要求文件的位置。
- ❸ 指定要使用的基础镜像。红帽支持仅针对 redhat.registry.io 基础镜像提供。
- ❹ 指定可能会影响构建器运行时功能的选项。
- ❺ 用于其他自定义构建步骤的命令。

### 其他资源

- 如需有关定义文件内容的更多信息，请参阅[定义文件内容的明细](#)。
- 要了解更多有关 Ansible Builder 版本 2 和 3 之间的区别的信息，请参阅[Ansible 3 端口指南](#)。

## 2.4. 构建自动化执行环境镜像

创建定义文件后，您可以继续构建自动化执行环境镜像。

### 先决条件

- 您已创建了定义文件。

### 流程

要构建自动化执行环境镜像，请从命令行运行以下命令：

```
$ ansible-builder build
```

默认情况下，Ansible Builder 会查找名为 **execution-environment.yml** 的定义文件，但可使用 **-f** 标志将不同的文件路径指定为参数：

```
$ ansible-builder build -f definition-file-name.yml
```

其中 *definition-file-name* 指定您的定义文件的名称。

## 2.5. 定义文件内容的分类

使用 Ansible Builder 构建自动化执行环境需要定义文件，因为它指定了自动化执行环境容器镜像中包含的内容。

以下小节细分了定义文件的不同部分。

### 2.5.1. 构建参数和基础镜像

定义文件的 **build\_arg\_defaults** 部分是一个字典，其键可为 Ansible Builder 的参数提供默认值。下表列出了 **build\_arg\_defaults** 中可以使用的值：

值	Description
<b>ANSIBLE_GALAXY_CLI_COLLECTION_OPTS</b>	允许用户在集合安装过程中将任意参数传递给 <code>ansible-galaxy CLI</code> 。例如， <code>-pre</code> 标志启用预发布集合的安装，或者 <code>-c</code> 禁用服务器的 SSL 证书的验证。
<b>ANSIBLE_GALAXY_CLI_ROLE_OPTS</b>	允许用户将任何标志（如 <code>-no-deps</code> ）传递给角色安装。

**build\_arg\_defaults** 中指定的值将硬编码到 **Containerfile** 中，因此这些值将在手动调用 **podman build** 时保留。



### 注意

如果在 CLI **--build-arg** 标志中指定相同的变量，CLI 值将具有更高的优先级。

您可以在定义文件的 **dependencies** 部分包含必须安装到最终镜像的依赖项。

为了避免自动化执行环境镜像出现问题，请确保 Galaxy、Python 和系统的条目指向有效的要求文件，或者是其相应的文件类型的有效内容。

### 2.5.1.1. Galaxy

**galaxy** 条目指向有效的要求文件，或者包括 **ansible-galaxy collection install -r ...** 命令的内联内容。

条目 **requirements.yml** 可以是来自自动化执行环境定义的文件夹目录的相对路径，也可以是绝对路径。

内容可能类似如下：

#### 例 2.2. Galaxy 条目

```
collections:
- community.aws
- kubernetes.core
```

### 2.5.1.2. Python

定义文件中的 **python** 条目指向有效的要求文件，或者指向 **pip install -r ...** 命令格式的 PEP508 格式的 Python 要求列表。

条目 **requirements.txt** 是一个文件，它会在集合已列出的 Python 依赖项之上安装额外的 Python 要求。它可以是对于自动化执行环境定义的文件夹目录的相对路径，也可以是绝对路径。**requirements.txt** 文件的内容应该类似以下示例，类似于 **pip freeze** 命令的标准输出：

#### 例 2.3. Python 条目

```
boto>=2.49.0
botocore>=1.12.249
pytz
python-dateutil>=2.7.0
awxkit
packaging
requests>=2.4.2
xmltodict
azure-cli-core==2.11.1
openshift>=0.6.2
requests-oauthlib
openstacksdk>=0.13
ovirt-engine-sdk-python>=4.4.10
```

### 2.5.1.3. System

定义中的 **system** 条目指向 **bindep** 要求文件，或指向 **bindep** 条目的内联列表，该条目安装集合已包含为依赖项的系统级依赖项。它可以是来自自动化执行环境定义的文件夹目录的相对路径，也可以列为绝对路径。至少，集合必须为 **[platform:rpm]** 指定必要的要求。

要演示这一点，以下是 **bindep.txt** 文件示例，该文件将 **libxml2** 和 **subversion** 软件包添加到容器中：

#### 例 2.4. 系统条目

```
libxml2-devel [platform:rpm]
subversion [platform:rpm]
```

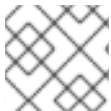
来自多个集合的条目合并到一个文件中。这通过 **bindep** 进行处理，然后传递到 **dnf**。镜像中仅会安装没有配置集或无运行时要求的要求。

## 2.5.2. 镜像

定义文件的 **images** 部分标识基础镜像。**podman** 容器运行时支持验证签名的容器镜像。

下表列出了您可以在 **镜像** 中使用的值：

value	Description
<b>base_image</b>	<p>指定自动化执行环境的父镜像，它允许基于现有镜像构建新镜像。这通常是一个受支持的执行环境基础镜像，如 <i>ee-minimal</i> 或 <i>ee-supported</i>，但它也可以是您创建的执行环境镜像，并希望进一步自定义。</p> <p>容器镜像需要使用 <b>name</b> 键。如果镜像在存储库中镜像，则指定 <b>签名</b> <b>_original_name</b> 密钥，但使用镜像的原始签名密钥签名。镜像名称必须包含标签，如 <b>:latest</b>。</p> <p>默认镜像为 <b>registry.redhat.io/ansible-automation-platform-24/ee-minimal-rhel8:latest</b>。</p>



### 注意

如果在 CLI **--build-arg** 标志中指定相同的变量，CLI 值将具有更高的优先级。

## 2.5.3. 其他构建文件

您可以将任何外部文件添加到构建上下文目录中，方法是将它们引用或复制到定义文件的 **additional\_build\_steps** 部分。格式是字典值的列表，每个值都有一个 **src** 和 **dest** 键和值。

每个列表项都必须是包含以下所需键的字典：

### src

指定要复制到构建上下文目录中的源文件。这可以是绝对路径（例如 **/home/user/.ansible.cfg**），也可以是相对于执行环境文件的路径。相对路径可以是 glob 表达式与一个或多个文件匹配（例如，**fileAttr.cfg**）。



### 注意

绝对路径不能包含正则表达式。如果 **src** 是目录，则该目录的整个内容都会复制到 **dest**。

### dest

指定构建上下文目录的 `_build` 子目录下的子目录路径，其中包含源文件（如 `files/configs`）。这不是绝对路径，也可以是在 **路径中包含 ..**。如果不存在，Ansible Builder 会为您创建此目录。

#### 2.5.4. 额外的自定义构建步骤

您可以在定义文件的 `additional_build_steps` 部分中为任何构建阶段指定自定义构建命令。这允许对构建阶段进行精细控制。

使用 `prepend_` 和 `append_` 命令将指令添加到在执行主构建步骤之前或之后运行的 `Containerfile`。命令必须符合运行时系统所需的任何规则。

下表列出了 `additional_build_steps` 中使用的值：

值	Description
<code>prepend_base</code>	允许您在构建基础镜像前插入命令。
<code>append_base</code>	允许您在构建基础镜像后插入命令。
<code>prepend_galaxy</code>	允许您在构建 galaxy 镜像前插入。
<code>append_galaxy</code>	允许您在构建 galaxy 镜像后插入。
<code>prepend_builder</code>	允许您在构建 Python 构建器镜像前插入命令。
<code>append_builder</code>	允许您在构建 Python 构建器镜像后插入命令。
<code>prepend_final</code>	允许您在构建最终镜像前插入。
<code>append_final</code>	允许您在构建最终镜像后插入。

`additional_build_steps` 的语法支持多行字符串和列表。请参见以下示例：

##### 例 2.5. 一个多行的字符串条目

```
prepend_final: |
  RUN whoami
  RUN cat /etc/os-release
```

##### 例 2.6. 一个列表条目

```
append_final:
- RUN echo This is a post-install command!
- RUN ls -la /etc
```

#### 2.5.5. 其他资源

- 有关常见场景的示例定义文件，请参阅 *Ansible Builder* 文档中的 [Common scenarios](#) 部分

## 2.6. 可选的 BUILD 命令参数

**-t** 标志将使用特定名称标记您的自动化执行环境镜像。例如，以下命令将构建名为 **my\_first\_ee\_image** 的镜像：

```
$ ansible-builder build -t my_first_ee_image
```



### 注意

如果您在构建中没有使用 **-t**，则创建名为 **ansible-execution-env'** 的镜像，并加载到本地容器 registry 中。

如果您有多个定义文件，可以通过包含 **-f** 标志来指定要使用的文件：

```
$ ansible-builder build -f another-definition-file.yml -t another_ee_image
```

在本例中，Ansible Builder 将使用名为 **another-definition-file.yml** 的文件中提供的规格，而不是默认的 **execution-environment.yml** 来构建名为 **another\_ee\_image** 的自动化执行环境镜像。

有关可用于 **build** 命令的其他规格和标志，请输入 **ansible-builder build --help** 来查看附加选项列表。

## 2.7. CONTAINERFILE

创建定义文件后，Ansible Builder 会读取并验证它，创建一个 **Containerfile** 和容器构建上下文，并选择性地将它们传递给 Podman 以构建自动化执行环境镜像。容器构建以几个不同的阶段发生：

**base**、**galaxy**、**builder** 和 **最终**。镜像构建步骤（以及 **additional\_build\_steps** 中定义的任何对应的自定义 **prepend\_** 和 **append\_** 步骤）包括：

1. 在 **基础** 构建阶段，指定的基础镜像是（可选）由其他构建阶段自定义的组件，包括 Python、**pip**、**ansible-core** 和 **ansible-runner**。然后，生成的镜像会被验证以确保所需的组件可用（因为它们可能已在基础镜像中存在）。生成的 **自定义基础镜像** 的临时副本用作所有其他构建阶段的基础。
2. 在 **galaxy** 构建阶段，由定义文件指定的集合会在 **最终** 构建阶段下载并供以后安装。还会收集集合声明的 Python 和系统依赖项（若有），以便稍后进行分析。
3. 在 **构建器** 构建阶段，集合声明的 Python 依赖项与定义文件中列出的 Python 依赖项合并。最后的 Python 依赖项被下载并构建为 Python wheels，并在 **最终** 构建阶段存储下来。
4. 在 **最终** 构建阶段，会安装之前下载的集合，以及系统软件包以及之前由集合声明为依赖项或定义文件中列出的依赖项的、以前构建的 Python 软件包。

## 2.8. 在不构建镜像的情况下创建容器文件

出于安全原因，您需要使用在沙盒环境中构建的共享容器镜像，您可以创建一个可共享 **Containerfile**。

```
$ ansible-builder create
```



## 第 3 章 常见自动化执行环境场景

使用以下示例定义文件来解决常见配置场景。

### 3.1. 更新自动化中心 CA 证书

使用本示例自定义默认定义文件，将 CA 证书包含到 **additional-build-files** 部分，将该文件移到适当的目录中，最后运行命令来更新 CA 证书的动态配置，以允许系统信任此 CA 证书。

#### 先决条件

- 自定义 CA 证书，如 **rootCA.crt**。



#### 注意

使用 **prepend\_base** 自定义 CA 证书意味着生成的 CA 配置出现在所有其他构建阶段和最终镜像中，因为所有其他构建阶段都继承自基础镜像。

```
additional_build_files:
  # copy the CA public key into the build context, we will copy and use it in the base image later
  - src: files/rootCA.crt
    dest: configs

additional_build_steps:
  prepend_base:
    # copy a custom CA cert into the base image and recompute the trust database
    # because this is in "base", all stages will inherit (including the final EE)
    - COPY _build/configs/rootCA.crt /usr/share/pki/ca-trust-source/anchors
    - RUN update-ca-trust

options:
  package_manager_path: /usr/bin/microdnf # downstream images use non-standard package
  manager

[galaxy]
server_list = automation_hub
```

### 3.2. 在构建自动化执行环境时使用自动化中心身份验证详情

使用以下示例自定义默认定义文件，将自动化中心身份验证详情传递给自动化执行环境构建，而无需在最终的自动化执行环境镜像中公开它们。

#### 先决条件

- 您已创建了自动化中心 API 令牌 并将其存储在安全位置，例如在名为 **token.txt** 的文件中。
- 定义使用自动化中心 API 令牌填充的构建参数：

```
export ANSIBLE_GALAXY_SERVER_AUTOMATION_HUB_TOKEN=$(cat <token.txt>)
```

```
additional_build_steps:
  prepend_galaxy:
```

```
# define a custom build arg env passthru- we still also have to pass
# `--build-arg ANSIBLE_GALAXY_SERVER_AUTOMATION_HUB_TOKEN` to get it to pick it up
from the host env
- ARG ANSIBLE_GALAXY_SERVER_AUTOMATION_HUB_TOKEN
- ENV ANSIBLE_GALAXY_SERVER_LIST=automation_hub
- ENV
ANSIBLE_GALAXY_SERVER_AUTOMATION_HUB_URL=https://console.redhat.com/api/automation-
hub/content/<yourhuburl>-synclist/
- ENV
ANSIBLE_GALAXY_SERVER_AUTOMATION_HUB_AUTH_URL=https://sso.redhat.com/auth/realms/
redhat-external/protocol/openid-connect/token
```

### 3.3. 其他资源

- 有关自动化执行环境定义文件的不同部分的信息，请参阅 [定义文件内容的明细](#)。
- 有关常见场景的其他定义文件示例，请参阅 *Ansible Builder* 文档中的 [常见场景部分](#)

## 第 4 章 发布自动化执行环境

### 4.1. 自定义现有自动化执行环境镜像

Ansible Controller 包括以下默认执行环境：

- **minimal** - 包括最新的 Ansible-core 2.15 发行版本以及 Ansible Runner，但不包括集合或其他内容
- **EE 支持** - Minimal，以及所有红帽支持的集合和依赖项

虽然这些环境涵盖了许多自动化用例，但您可以添加额外的项目来自定义这些容器，以满足您的特定需求。以下流程将 **kubernetes.core** 集合添加到 **ee-minimal** 默认镜像中：

#### 流程

1. 通过 Podman 登录到 **registry.redhat.io**：

```
$ podman login -u="[username]" -p="[token/hash]" registry.redhat.io
```

2. 确保您可以拉取所需的自动化执行环境基础镜像：

```
podman pull registry.redhat.io/ansible-automation-platform-24/ee-minimal-rhel8:latest
```

3. 配置 Ansible Builder 文件，以指定所需的基础镜像以及添加到新执行环境镜像的任何其他内容。

- a. 例如，要将来自 Galaxy 的 [Kubernetes Core Collection](#) 添加到镜像中，请使用 Galaxy 条目：

```
collections:
  - kubernetes.core
```

- b. 有关定义文件及其内容的更多信息，请参阅 [定义文件分类部分](#)。

4. 在执行环境定义文件中，在 **EE\_BASE\_IMAGE** 字段中指定原始 **ee-minimal** 容器 URL 和标签。在这样做时，您的最终 **execute-environment.yml** 文件将类似如下：

#### 例 4.1. 自定义 **execution-environment.yml** 文件

```
version: 3

images:
  base_image: 'registry.redhat.io/ansible-automation-platform-24/ee-minimal-rhel9:latest'

dependencies:
  galaxy:
    collections:
      - kubernetes.core
```



### 注意

由于本例使用 **kubernetes.core** 的社区版本，而不是来自自动化中心的认证集合，因此我们不需要创建 **ansible.cfg**，也不在定义文件中引用。

5. 使用以下命令构建新执行环境镜像：

```
$ ansible-builder build -t [username]/new-ee
```

其中 **[username]** 指定您的用户名，**new-ee** 指定新容器镜像的名称。



### 注意

如果您在构建中没有使用 **-t**，则创建名为 **ansible-execution-env'** 的镜像，并加载到本地容器 registry 中。

- 使用 **podman images** 命令确认您的新容器镜像在该列表中：

#### 例 4.2. 使用镜像 new-ee 的 podman images 命令的输出

```
REPOSITORY      TAG  IMAGE ID  CREATED  SIZE
localhost/new-ee latest f5509587efbb 3 minutes ago 769 MB
```

6. 验证是否安装了集合：

```
$ podman run [username]/new-ee ansible-doc -l kubernetes.core
```

7. 标记在自动化中心中使用的镜像：

```
$ podman tag [username]/new-ee [automation-hub-IP-address]/[username]/new-ee
```

8. 使用 Podman 登录到您的自动化 hub:



### 注意

您必须具有 Automation hub 的 **admin** 或适当的容器存储库权限才能推送容器。如需更多信息，请参阅管理 [自动化中心中的内容](#) 中的 [管理私有自动化中心](#) 中的 [管理容器](#) 部分。

```
$ podman login -u="[username]" -p="[token/hash]" [automation-hub-IP-address]
```

9. 将镜像推送到自动化 hub 中的容器 registry 中：

```
$ podman push [automation-hub-IP-address]/[username]/new-ee
```

10. 将新镜像拉取(pull)到自动化控制器实例中：

- a. 进入自动化控制器。
- b. 在导航面板中，选择 **Administration** → **Execution Environments**。

- c. 点 **Add**。
- d. 输入适当的信息，然后点 **Save** 以拉取新镜像。



### 注意

如果您的自动化中心实例受密码或令牌保护，请确保设置了适当的容器 registry 凭证。

## 4.2. 其他资源（或后续步骤）

有关基于常见场景自定义执行环境的更多详细信息，请参阅 *Ansible Builder* 文档中的以下主题：

- [将仲裁文件复制到执行环境中](#)
- [使用环境变量构建执行环境](#)
- [使用环境变量和 `ansible.cfg` 构建执行环境](#)

## 第 5 章 填充私有自动化中（AUTOMATION HUB）容器 REGISTRY

默认情况下，私有自动化中心不包括容器镜像。要填充容器 registry，您需要将容器镜像推送到其中。

您必须遵循特定的工作流程来填充私有自动化中心容器 registry：

- 从红帽生态系统目录 (registry.redhat.io) 中拉取镜像
- 标记它们
- 将它们推送到您的私有自动化中心容器 registry

### 重要

最初，镜像清单和文件系统 Blob 都直接由 **registry.redhat.io** 和 **registry.access.redhat.com** 提供。从 2023 年 5 月 1 日起，文件系统 Blob 由 **quay.io** 提供。

- 确保 [Table 5.10](#) 中列出的 [网络端口和协议](#)。执行环境(EE) 可用，以避免拉取容器镜像出现问题。

对任何特别启用到 **registry.redhat.io** 或 **registry.access.redhat.com** 的出站连接进行此更改。

在配置防火墙规则时使用主机名而不是 IP 地址。

完成此更改后，您可以继续从 **registry.redhat.io** 和 **registry.access.redhat.com** 拉取镜像。您不需要 **quay.io** 登录，或以任何方式直接与 **quay.io** registry 交互，以继续拉取红帽容器镜像。

但是，在基于 Web 的 Red Hat Subscription Management 上，清单（有时称为“订阅分配”）不再受支持（在 Red Hat Satellite 6.16 发布前）。对于 [Red Hat Satellite 6.16 的发布日期](#)，保持最新的 Red Hat Satellite 发行日期。

### 5.1. 拉取 (PULL) 用于自动化中心的镜像

在将容器镜像推送到私有自动化中心之前，您必须首先从现有 registry 中拉取容器镜像并标记它们以供使用。以下示例详细介绍了如何从红帽生态系统目录 (registry.redhat.io) 拉取镜像。

### 重要

从 2024 年初期，红帽不再支持 Red Hat Subscription Management Web 平台上的清单或清单列表，这些列表与“订阅分配”互换使用。红帽不再支持 Red Hat Satellite 中的大多数清单功能，但有一个例外：在 Red Hat Satellite 6.16 发布之前，Red Hat Satellite 用户处于封闭的网络或“air gapped”网络，目前仍然可以使用 **access.redhat.com**。

新的红帽帐户自动为其订阅工具使用简单内容访问。新的红帽帐户和可以连接到红帽服务器的现有 Satellite 客户可以在 **console.redhat.com** 上找到其清单。

#### 前提条件

- 有从 registry.redhat.io 拉取镜像的权限
- 启用简单内容访问的红帽帐户。

## 流程

1. 如果需要访问容器镜像的清单，请登录 [红帽控制台](#)。
2. 单击您容器镜像所需的清单的三 ot 菜单，再单击 **导出清单**。
3. 使用您的 registry.redhat.io 凭证登录到 Podman：

```
$ podman login registry.redhat.io
```

4. 输入您的用户名和密码。
5. 拉取容器镜像：

```
$ podman pull registry.redhat.io/<container_image_name>:<tag>
```

## 验证

要验证您最近拉取的镜像是否包含在列表中，请执行以下步骤：

1. 列出本地存储中的镜像：

```
$ podman images
```

2. 检查镜像名称，并验证标签是否正确。

## 其他资源

- 有关注册和获取镜像的信息，请参阅 [Red Hat Ecosystem Catalog Help](#)。
- 请参阅 [创建和管理连接的 Satellite 服务器的清单](#)，以了解更多有关红帽订阅工具的更改的信息。

## 5.2. 用于自动化中心的标记镜像

从 registry 中拉取镜像后，标记它们以便在私有自动化中心容器 registry 中使用。

### 前提条件

- 您已从外部 registry 中提取容器镜像。
- 有自动化中心实例的 FQDN 或 IP 地址。

### 流程

- 使用自动化中心容器存储库标记本地镜像：

```
$ podman tag registry.redhat.io/<container_image_name>:<tag>  
<automation_hub_hostname>/<container_image_name>
```

### 验证

1. 列出本地存储中的镜像：

```
$ podman images
```

2. 验证您最近使用自动化中心信息标记的镜像是否包含在列表中。

### 5.3. 将容器镜像推送到私有自动化中心

您可以将标记的容器镜像推送到私有自动化中心，以创建新容器并填充容器 registry。

#### 前提条件

- 有创建新容器的权限。
- 有自动化中心实例的 FQDN 或 IP 地址。

#### 流程

1. 使用您的自动化中心位置和凭证登录到 Podman :

```
$ podman login -u=<username> -p=<password> <automation_hub_url>
```

2. 将容器镜像推送到自动化中心容器 registry :

```
$ podman push <automation_hub_url>/<container_image_name>
```

#### 故障排除

**push** 操作会在上传期间重新编译镜像层，无法保证可重复生成，依赖于客户端实施。这可能会导致镜像层摘要的变化，并导致推送操作失败，**Error: Copying this image requires changing layer representation, which is not possible (image is signed or the destination specifies a digest)**。

#### 验证

1. 登录到您的自动化中心。
2. 进入 **Container Registry**。
3. 在 container 存储库列表中找到容器。



## 第 6 章 设置容器存储库

设置容器存储库时，您必须添加描述，包括 README，添加可访问存储库的组，以及标签镜像。

### 6.1. 设置容器 REGISTRY 的先决条件

- 已登录到一个私有自动化中心。
- 您有更改存储库的权限。

### 6.2. 将 README 添加到容器存储库中

将 README 添加到容器存储库，以向用户提供如何使用容器的说明。自动化中心容器存储库支持 Markdown 创建 README。默认情况下，README 为空。

#### 前提条件

- 有更改容器的权限。

#### 流程

1. 登录到自动化中心。
2. 在导航面板中，选择 **Execution Environments** → **Execution Environments**。
3. 选择您的容器存储库。
4. 在 **Details** 标签页中，点 **Add**。
5. 在 **Raw Markdown** 文本字段中，使用 Markdown 格式输入您的 README 文本。
6. 完成后点 **Save**。

添加 README 后，您可以随时通过点 **Edit** 并重复步骤 4 和 5 对其进行编辑。

### 6.3. 提供对容器存储库的访问

为需要使用镜像的用户提供容器存储库的访问权限。通过添加组，您可以修改组对容器存储库的权限。您可以使用这个选项根据组分配的内容来扩展或限制权限。

#### 前提条件

- 您具有改变容器命名空间的权限。

#### 流程

1. 登录到自动化中心。
2. 在导航面板中，选择 **Execution Environments** → **Execution Environments**。
3. 选择您的容器存储库。
4. 在 **Access** 选项卡中，点 **Select a group**。

5. 选择您要授予访问权限的组，然后点 **Next**。
6. 选择您要添加到此执行环境的角色，然后点 **Next**。
7. 点 **Add**。


## 6.4. 标记容器镜像

标记镜像，为存储在自动化中心容器存储库中的镜像添加额外名称。如果没有向镜像添加任何标签，则自动化中心名称默认为 **latest**。

### 前提条件

- 您有更改镜像标签的权限。

### 流程

1. 在导航面板中，选择 **Execution Environments** → **Execution Environments**。
2. 选择您的容器存储库。
3. 点 **Images** 选项卡。
4. 点 **More Actions** 图标 ，点 **Manage tags**。
5. 在文本字段中添加新标签，然后点 **Add**。
6. 可选：通过点该镜像的任何标签上的 **x** 来删除当前标签。
7. 点击 **Save**。

### 验证

- 点 **Activity** 选项卡，再检查最新的变化。

## 6.5. 在自动化控制器中创建凭证

要从密码保护的 registry 中拉取容器镜像，您必须在自动化控制器中创建凭证。

在早期版本的 Ansible Automation Platform 中，您需要部署 registry 来存储执行环境镜像。在 Ansible Automation Platform 2.0 及更新的版本中，系统会象您已启动并运行容器 registry 一样运行。要存储执行环境镜像，请仅添加所选容器 registry 的凭证。

### 流程

1. 进入自动化控制器。
2. 在导航面板中，选择 **Resources** → **Credentials**。
3. 点 **Add** 以创建新凭据。
4. 输入授权 **Name**、**Description** 和 **Organization**。
5. 选择**凭据类型**。

6. 输入**身份验证 URL**。这是容器 registry 地址。
7. 输入登录到容器 registry 所需的 **Username** 和 **Password or Token**。
8. 可选：要启用 SSL 验证，请选择 **Verify SSL**。
9. 点击 **Save**。

## 第 7 章 从容器存储库中拉取镜像

从自动化中心容器 registry 中拉取镜像，以便将镜像复制到本地机器。自动化中心为容器存储库中的每个 **latest** 镜像提供 **podman pull** 命令。您可以将此命令复制并粘贴到终端中，或者使用 **podman pull** 根据镜像标签复制镜像。

### 7.1. 拉取镜像

您可以从自动化中心容器 registry 中拉取镜像，以在本地机器中创建副本。

#### 前提条件

- 您需要有从私有容器存储库查看和拉取的权限。

#### 流程

1. 如果您要从密码保护的 registry 中拉取容器镜像，请在拉取镜像前 [在自动化控制器中创建凭证](#)。
2. 在导航面板中，选择 **Execution Environments** → **Execution Environments**。
3. 选择您的容器存储库。
4. 在 **Pull this image** 条目中，点 **Copy to clipboard**。
5. 在终端中粘贴并运行命令。

#### 验证

- 运行 **podman images** 以查看本地机器上的镜像。

### 7.2. 从容器存储库同步镜像

您可以从自动化中心容器 registry 拉取镜像，将镜像同步到本地机器。要从远程容器注册表同步镜像，您必须首先配置远程 registry。

#### 前提条件

您需要有从私有容器存储库查看和拉取的权限。

#### 流程

1. 在导航面板中，选择 **Execution Environments** → **Execution Environments**。
2. 将 <https://registry.redhat.io> 添加到 registry。
3. 添加所需的任何凭证进行验证。

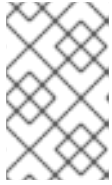


#### 注意

有些容器 registry 有速率限制。在 **Advanced Options** 下设置速率限制。

4. 在导航面板中，选择 **Execution Environments** → **Execution Environments**。

5. 在页面标头中点 **Add execution environment**。
6. 选择您要从中拉取的 registry。Name 字段显示本地 registry 中显示的镜像名称。



### 注意

**Upstream name** 字段是远程服务器上的镜像名称。例如，如果上游名称被设置为 "alpine"，并且 **Name** 字段为 "local/alpine"，则 alpine 镜像会从远程下载，并重命名为 "local/alpine"。

7. 设置要包含或排除的标签列表。将镜像与大量标签同步会非常耗时，并且需要使用大量磁盘空间。

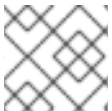
### 其他资源

- 如需 registry 列表，请参阅 [Red Hat Container Registry Authentication](#)。
- 有关拉取镜像时要使用的选项，请参阅 [Podman 是什么？](#) 文档。

## 附录 A. 自动化执行环境优先级

默认情况下，项目更新始终使用 control plane 自动化执行环境，但作业将使用第一个可用的自动化执行环境，如下所示：

1. 创建了该作业的模板（作业模板或清单源）上定义的 **execution\_environment**。
2. 作业使用的项目上定义的 **default\_environment**。
3. 在作业的组织上定义的 **default\_environment**。
4. 在作业使用的清单的组织上定义的 **default\_environment**。
5. 当前 **DEFAULT\_EXECUTION\_ENVIRONMENT** 设置（可在 **api/v2/settings/jobs/**中进行配置）
6. 来自 **GLOBAL\_JOB\_EXECUTION\_ENVIRONMENTS** 设置的任何镜像。
7. 任何其它全局执行环境。



### 注意

如果多个执行环境符合条件（适用于 6 和 7），则会使用最近创建的执行环境。