Red Hat Ansible Automation Platform 2.4

容器化 Ansible Automation Platform 安装指南

容器化 Ansible Automation Platform 安装指南

Last Updated: 2024-07-03

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

容器化 Ansible Automation Platform 安装指南

法律通告

摘要

本指南帮助您了解我们新容器化版本的 Ansible Automation Platform 的安装要求和流程。

目录

对红帽文档提供反馈 3

第 1 章 ANSIBLE AUTOMATION PLATFORM 容器化安装 4
1.1. 系统要求 4
1.2. 为容器化安装准备 RHEL 主机 4
1.3. 安装 ANSIBLE-CORE 5
1.4. 下载 ANSIBLE AUTOMATION PLATFORM 5
1.5. 使用容器化 ANSIBLE AUTOMATION PLATFORM 的 POSTINSTALL 功能 6
1.6. 安装容器化 ANSIBLE AUTOMATION PLATFORM 7
1.7. 访问自动化控制器、自动化中心和 EVENT-DRIVEN ANSIBLE 控制器 10
1.8. 使用自定义 TLS 证书 11
1.9. 使用自定义 RECEPTOR 签名密钥 11
1.10. 启用 AUTOMATION HUB 集合和容器签名 11
1.11. 添加执行节点 12
1.12. 卸载容器化 ANSIBLE AUTOMATION PLATFORM 12

附录 A. 容器化 ANSIBLE AUTOMATION PLATFORM 故障排除 14
A.1. 容器化 ANSIBLE AUTOMATION PLATFORM 安装故障排除 14
A.2. 容器化 ANSIBLE AUTOMATION PLATFORM 配置故障排除 16
A.3. 容器化 ANSIBLE AUTOMATION PLATFORM 参考 17

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 ANSIBLE AUTOMATION PLATFORM 容器化安装

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。

本指南帮助您了解我们新容器化版本的 Ansible Automation Platform 的安装要求和流程。此初始版本基于 Ansible Automation Platform 2.4，并作为技术预览发布。请参阅 技术预览功能支持范围 以了解技术预览。

先决条件

- 基于 RHEL 9.2 的主机。建议使用最小操作系统基础安装。
- RHEL 主机的非 root 用户，具有 sudo 或其他 Ansible 支持的权限升级（建议使用 sudo）。此用户负责安装容器化 Ansible Automation Platform。
- 建议为非 root 用户设置一个 **SSH 公钥身份验证**。有关为非 root 用户设置 SSH 公钥身份验证的指南，请参阅如何为免密码登录配置 SSH 公钥身份验证。
- 只有在远程主机上安装时，才需要 SSH 密钥。如果是基于自我包含的本地虚拟机的安装，可以参照不需要 SSH 的示例，使用 **ansible_connection: local**。
- 如果使用默认的在线安装方法，可以从 RHEL 主机访问互联网。

1.1. 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 Containerized Ansible Automation Platform。

内存	16Gb RAM
CPU	4 CPU
磁盘空间	40Gb
磁盘 IOP	1500

1.2. 为容器化安装准备 RHEL 主机

流程

容器化 Ansible Automation Platform 作为基于 podman 的容器在 RHEL 主机之上运行组件服务。准备底层主机后，安装程序会负责此操作。执行以下操作。

1. 以您的非 root 用户身份登录到 RHEL 主机。
2. 运行 **dnf repolist** 以验证在主机上仅设置并启用了 BaseOS 和 appstream 存储库：

```
$ dnf repolist
Updating Subscription Management repositories.
repo id                                   repo name
rhel-9-for-x86_64-appstream-rpms          Red Hat Enterprise Linux 9 for x86_64 -
```

```
AppStream (RPMs)
rhel-9-for-x86_64-baseos-rpms                Red Hat Enterprise Linux 9 for x86_64 -
BaseOS (RPMs)
```

3. 确保这些仓库以及只有这些仓库可供主机操作系统使用。如果您需要了解如何执行本指南：Chapter 10。管理自定义软件存储库 Red Hat Enterprise Linux

4. 确保主机配置了 DNS，并且可以使用完全限定域名(FQDN)解析主机名和 IP。这对于确保服务可以互相通信非常重要。

使用 unbound DNS

要配置 unbound DNS，请参阅 Chapter 2。设置 unbound DNS 服务器 Red Hat Enterprise Linux 9 。

使用 BIND DNS

要使用 BIND 配置 DNS，请参阅 Chapter 1。设置和配置 BIND DNS 服务器 Red Hat Enterprise Linux 9 。

选填

要让安装程序自动获取并应用 Ansible Automation Platform 订阅清单许可证，请使用本指南生成可针对安装程序下载的清单文件： Chapter 2。获取 Red Hat Ansible Automation Platform 2. 的清单文件。

1.3. 安装 ANSIBLE-CORE

流程

1. 安装 ansible-core 和其他工具：

```
sudo dnf install -y ansible-core wget git rsync
```

2. 设置完全限定主机名：

```
sudo hostnamectl set-hostname your-FQDN-hostname
```

1.4. 下载 ANSIBLE AUTOMATION PLATFORM

流程

1. 从 access.redhat.com 下载最新的安装程序 tarball。这可以在 RHEL 主机内直接完成，从而节省操作的时间。

2. 如果您已将 tarball 和可选的 manifest zip 文件下载到您的笔记本电脑中，请将它们复制到 RHEL 主机上。
确定安装程序要保留在文件系统的什么位置。与安装相关的文件将在此位置中创建，需要至少 10Gb 来进行初始安装。

3. 在安装目录中解压安装程序 tarball，cd 到解压的目录中。

   a. 在线安装程序

```
$ tar xfvz ansible-automation-platform-containerized-setup-2.4-2.tar.gz
```

b. 捆绑的安装程序

```
$ tar xfvz ansible-automation-platform-containerized-setup-bundle-2.4-2-<arch
name>.tar.gz
```

1.5. 使用容器化 ANSIBLE AUTOMATION PLATFORM 的 POSTINSTALL 功能

使用容器化 Ansible Automation Platform 的实验性 postinstaller 功能，在初始安装过程中定义和加载配置。这使用 configuration-as-code 的方法，只需将配置定义为简单的 YAML 文件。

1. 要使用此可选功能，您需要在清单文件中取消注释以下变量：

```
controller_postinstall=true
```

2. 默认值为 false，因此您需要启用它来激活 postinstaller。对于必须驻留在本地文件系统中的此功能，您需要一个 Ansible Automation Platform 许可证，以便自动载入它：

```
controller_license_file=/full_path_to/manifest_file.zip
```

3. 您可以从基于 Git 的存储库中拉取配置作为代码。要做到这一点，将以下变量设置为指定从中拉取内容的位置，并将其上传到 Ansible Automation Platform 控制器的位置：

```
controller_postinstall_repo_url=https://your_cac_scm_repo
controller_postinstall_dir=/full_path_to_where_you_want_the pulled_content_to_reside
```

4. controller_postinstall_repo_url 变量可用于定义必须包含身份验证信息的 postinstall 存储库 URL。

```
http(s)://<host>/<repo>.git (public repository without http(s) authentication)
http(s)://<user>:<password>@<host>:<repo>.git (private repository with http(s)
authentication)
git@<host>:<repo>.git (public/private repository with ssh authentication)
```

注意

使用基于 ssh 的身份验证时，安装程序不会为您配置任何内容，因此您必须在安装程序节点上配置所有内容。

定义文件使用 infra 认证的集合。controller_configuration 集合已作为安装的一部分预安装，并使用您在清单文件中提供的安装控制器凭证来访问 Ansible Automation Platform 控制器。您只需要提供 YAML 配置文件。

您可以设置 Ansible Automation Platform 配置属性，如凭证、LDAP 设置、用户和团队、机构、项目、清单和主机、作业和工作流模板。

以下示例显示了定义和加载控制器作业模板的示例 **your-config.yml**。该示例演示了对 Ansible Automation Platform 安装提供的预加载演示示例的简单更改。

```
/full_path_to_your_configuration_as_code/
├── controller
    └── job_templates.yml
```

```
controller_templates:
 - name: Demo Job Template
   execution_environment: Default execution environment
   instance_groups:
  - default
   inventory: Demo Inventory
```

1.6. 安装容器化 ANSIBLE AUTOMATION PLATFORM

Ansible Automation Platform 的安装使用清单文件控制。清单文件定义使用和创建的主机和容器、组件的变量以及自定义安装所需的其他信息。

为方便提供示例清单文件，您可以复制和修改以快速启动。

注意

清单文件中未给出默认数据库选择。您必须按照清单文件中的说明，在内部提供的 postgres 之间进行适当的选择，或者提供您自己的外部管理和支持的数据库选项。

通过将 < **&gt**; 占位符替换为您的特定变量并取消注释所有特定于您的需要的行来编辑清单文件。

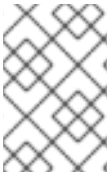
```
# This is the AAP installer inventory file
# Please consult the docs if you're unsure what to add
# For all optional variables please consult the included README.md

# This section is for your AAP Controller host(s)
# -----------------------------------------------
[automationcontroller]
fqdn_of_your_rhel_host ansible_connection=local

# This section is for your AAP Automation Hub host(s)
# -----------------------------------------------
[automationhub]
fqdn_of_your_rhel_host ansible_connection=local

# This section is for your AAP EDA Controller host(s)
# -----------------------------------------------
[automationeda]
fqdn_of_your_rhel_host ansible_connection=local

# This section is for your AAP Execution host(s)
# -----------------------------------------------
#[execution_nodes]
#fqdn_of_your_rhel_host

# This section is for the AAP database(s)
# -----------------------------------------
# Uncomment the lines below and amend appropriately if you want AAP to install and manage the
postgres databases
# Leave commented out if you intend to use your own external database and just set appropriate
_pg_hosts vars
# see mandatory sections under each AAP component
#[database]
#fqdn_of_your_rhel_host ansible_connection=local
```

```
[all:vars]

# Common variables needed for installation
# ----------------------------------------
postgresql_admin_username=postgres
postgresql_admin_password=<set your own>
# If using the online (non-bundled) installer, you need to set RHN registry credentials
registry_username=<your RHN username>
registry_password=<your RHN password>
# If using the bundled installer, you need to alter defaults by using:
#bundle_install=true
# The bundle directory must include /bundle in the path
#bundle_dir=<full path to the bundle directory>
# To add more decision environment images you need to set the de_extra_images variable
#de_extra_images=[{'name': 'Custom decision environment', 'image':
'<registry>/<namespace>/<image>:<tag>'}]
# To add more execution environment images you need to set the ee_extra_images variable
#ee_extra_images=[{'name': 'Custom execution environment', 'image':
'<registry>/<namespace>/<image>:<tag>'}]
# To use custom TLS CA certificate/key you need to set these variables
#ca_tls_cert=<full path to your TLS CA certificate file>
#ca_tls_key=<full path to your TLS CA key file>

# AAP Database - optional
# -----------------------
# To use custom TLS certificate/key you need to set these variables
#postgresql_tls_cert=<full path to your TLS certificate file>
#postgresql_tls_key=<full path to your TLS key file>

# AAP Controller - mandatory
# --------------------------
controller_admin_password=<set your own>
controller_pg_host=fqdn_of_your_rhel_host
controller_pg_password=<set your own>

# AAP Controller - optional
# -------------------------
# To use the postinstall feature you need to set these variables
#controller_postinstall=true
#controller_license_file=<full path to your manifest .zip file>
#controller_postinstall_dir=<full path to your config-as-code directory>
# When using config-as-code in a git repository
#controller_postinstall_repo_url=<url to your config-as-code git repository>
#controller_postinstall_repo_ref=main
# To use custom TLS certificate/key you need to set these variables
#controller_tls_cert=<full path to your TLS certificate file>
#controller_tls_key=<full path to your TLS key file>

# AAP Automation Hub - mandatory
# ------------------------------
hub_admin_password=<set your own>
hub_pg_host=fqdn_of_your_rhel_host
hub_pg_password=<set your own>

# AAP Automation Hub - optional
```

```
# ----------------------------
# To use the postinstall feature you need to set these variables
#hub_postinstall=true
#hub_postinstall_dir=<full path to your config-as-code directory>
# When using config-as-code in a git repository
#hub_postinstall_repo_url=<url to your config-as-code git repository>
#hub_postinstall_repo_ref=main
# To customize the number of worker containers
#hub_workers=2
# To use the collection signing feature you need to set these variables
#hub_collection_signing=true
#hub_collection_signing_key=<full path to your gpg key file>
# To use the container signing feature you need to set these variables
#hub_container_signing=true
#hub_container_signing_key=<full path to your gpg key file>
# To use custom TLS certificate/key you need to set these variables
#hub_tls_cert=<full path to your TLS certificate file>
#hub_tls_key=<full path to your TLS key file>

# AAP EDA Controller - mandatory
# ------------------------------
eda_admin_password=<set your own>
eda_pg_host=fqdn_of_your_rhel_host
eda_pg_password=<set your own>

# AAP EDA Controller - optional
# ----------------------------
# When using an external controller node unmanaged by the installer.
#controller_main_url=https://fqdn_of_your_rhel_host
# To customize the number of default/activation worker containers
#eda_workers=2
#eda_activation_workers=2
# To use custom TLS certificate/key you need to set these variables
#eda_tls_cert=<full path to your TLS certificate file>
#eda_tls_key=<full path to your TLS key file>

# AAP Execution Nodes - optional
# ----------------------------
#receptor_port=27199
#receptor_protocol=tcp
# To use custom TLS certificate/key you need to set these variables
#receptor_tls_cert=<full path to your TLS certificate file>
#receptor_tls_key=<full path to your TLS key file>
# To use custom RSA key pair you need to set these variables
#receptor_signing_private_key=<full path to your RSA private key file>
#receptor_signing_public_key=<full path to your RSA public key file>
```

使用以下命令安装容器化 Ansible Automation Platform：

```
ansible-playbook -i inventory ansible.containerized_installer.install
```



注意

If your privilege escalation requires a password to be entered, append *-K* to the command line. You will then be prompted for the *BECOME* password.

您可以提高输出的详细程度，最多可以使用 4 个 v (-vvvv) 来查看安装过程的详细信息。

注意

这可显著增加安装时间，因此建议您仅在红帽支持需要或要求时才使用它。

1.7. 访问自动化控制器、自动化中心和 EVENT-DRIVEN ANSIBLE 控制器

安装完成后，这是使用的默认协议和端口：

- HTTP/https 协议
- 自动化控制器的端口 8080/8443
- 用于自动化中心的端口 8081/8444
- Event-Driven Ansible 控制器的端口 8082/8445

这些设置可以被改变。详情请查看 **README.md**。建议您保留默认值，除非您因为端口冲突或其他因素而需要更改它们。

访问自动化控制器 UI

自动化控制器 UI 默认位于：

```
https://<your_rhel_host>:8443
```

以 admin 用户身份使用您为 **controller_admin_password** 创建的密码登录。

如果作为安装的一部分提供了许可证清单，则会显示 Ansible Automation Platform 仪表板。如果您没有提供许可证文件，则会显示 **Subscription** 屏幕，您需要在其中提供您的许可证详情。这里记录了：Chapter 1。激活 Red Hat Ansible Automation Platform 。

访问自动化中心 UI

Automation hub UI 默认位于：

```
https://<hub node>:8444
```

以 admin 用户身份使用您为 **hub_admin_password** 创建的密码登录。

访问 Event-Driven Ansible UI

Event-Driven Ansible UI 默认位于：

```
https://<eda node>:8445
```

以 admin 用户身份使用您为 **eda_admin_password** 创建的密码登录。

1.8. 使用自定义 TLS 证书

默认情况下，安装程序会为由自定义证书颁发机构(CA)签名的所有服务生成 TLS 证书和密钥。您可以为每个服务提供自定义 TLS 证书/密钥。如果该证书由自定义 CA 签名，您必须提供 CA TLS 证书和密钥。

- 证书颁发机构

```
ca_tls_cert=/full/path/to/tls/certificate
ca_tls_key=/full/path/to/tls/key
```

- 自动化控制器

```
controller_tls_cert=/full/path/to/tls/certificate
controller_tls_key=/full/path/to/tls/key
```

- Automation Hub

```
hub_tls_cert=/full/path/to/tls/certificate
hub_tls_key=/full/path/to/tls/key
```

- 自动化 EDA

```
eda_tls_cert=/full/path/to/tls/certificate
eda_tls_key=/full/path/to/tls/key
```

- Postgresql

```
postgresql_tls_cert=/full/path/to/tls/certificate
postgresql_tls_key=/full/path/to/tls/key
```

- Receptor

```
receptor_tls_cert=/full/path/to/tls/certificate
receptor_tls_key=/full/path/to/tls/key
```

1.9. 使用自定义 RECEPTOR 签名密钥

现在，Receptor 签名会被默认启用，除非设置了 **receptor_disable_signing=true**，否则安装程序会生成 RSA 密钥对(public/private)。但是，您可以通过设置 path 变量来提供自定义 RSA 公钥/私钥。

```
receptor_signing_private_key=/full/path/to/private/key
receptor_signing_public_key=/full/path/to/public/key
```

1.10. 启用 AUTOMATION HUB 集合和容器签名

Automation Hub 允许您为 Ansible 集合和容器镜像签名。默认情况下不启用此功能，您必须提供 GPG 密钥。

```
hub_collection_signing=true
hub_collection_signing_key=/full/path/to/collections/gpg/key
hub_container_signing=true
```

```
hub_container_signing_key=/full/path/to/containers/gpg/key
```

当 GPG 密钥受密码保护时，您必须提供密码短语。

```
hub_collection_signing_pass=<collections gpg key passphrase>
hub_container_signing_pass=<containers gpg key passphrase>
```

1.11. 添加执行节点

容器化安装程序可以部署远程执行节点。这由 ansible 清单文件中的 execution_nodes 组处理。

```
[execution_nodes]
fqdn_of_your_execution_host
```

执行节点默认配置为在端口 27199 (TCP)上运行的执行类型。这可以通过以下变量更改：

- receptor_port=27199
- receptor_protocol=tcp
- receptor_type=hop

Receptor 类型值可以是 execution 或 hop，而协议可以是 TCP 或 UDP。默认情况下，**execution_nodes** 组中的节点将添加为控制器节点的对等点。但是，您可以使用 **receptor_peers** 变量更改对等配置。

```
[execution_nodes]
fqdn_of_your_execution_host
fqdn_of_your_hop_host receptor_type=hop receptor_peers='["fqdn_of_your_execution_host"]'
```

1.12. 卸载容器化 ANSIBLE AUTOMATION PLATFORM

要卸载容器化部署，请执行 **uninstall.yml** playbook。

```
$ ansible-playbook -i inventory ansible.containerized_installer.uninstall
```

这将停止所有 systemd 单元和容器，然后删除容器化安装程序使用的所有资源，例如：

- 配置和数据目录/文件
- systemd 单元文件
- podman 容器和镜像
- RPM 软件包

要保留容器镜像，您可以将 **container_keep_images** 变量设置为 true。

```
$ ansible-playbook -i inventory ansible.containerized_installer.uninstall -e
container_keep_images=true
```

要保留 postgresql 数据库，您可以将 **postgresql_keep_databases** 变量设置为 true。

```
$ ansible-playbook -i </path/to/inventory> ansible.containerized_installer.uninstall -e
postgresql_keep_databases=true
```

注意

您必须使用相同的 django 机密密钥值，而不是自动生成的密钥。

附录 A. 容器化 ANSIBLE AUTOMATION PLATFORM 故障排除

使用此信息对容器化 Ansible Automation Platform 安装进行故障排除。

A.1. 容器化 ANSIBLE AUTOMATION PLATFORM 安装故障排除

安装需要很长时间，或者出现错误，我应检查什么内容？

1. 确保您的系统满足安装指南中所述的最低要求。在很多主机间分布不当的存储选择和高延迟等项目都将严重影响。
2. 在默认情况下，检查位于 **./aap_install.log** 的安装日志文件，除非在本地安装程序 **ansible.cfg** 中有其他变化。
3. 临时启用任务分析回调，以概述安装程序花费最多时间的概览。要做到这一点，请使用本地 **ansible.cfg** 文件。在 **[defaults]** 部分下添加回调行，例如：

```
$ cat ansible.cfg
[defaults]
callbacks_enabled = ansible.posix.profile_tasks
```

自动化控制器会返回 413 错误

此错误的原因是 **manifest.zip** 许可证文件大于 **nginx_client_max_body_size** 设置。如果发生这个错误，您需要更改安装清单文件使其包含以下变量：

```
nginx_disable_hsts: false
nginx_http_port: 8081
nginx_https_port: 8444
nginx_client_max_body_size: 20m
nginx_user_headers: []
```

20m 的当前默认设置应该足以避免此问题。

当进入控制器 UI 时，安装会失败并显示 "502 Bad Gateway"。

这个错误可能会在安装应用程序输出中发生，并在安装应用程序输出中清单本身：

```
TASK [ansible.containerized_installer.automationcontroller : Wait for the Controller API to te ready] ******************************************************
fatal: [daap1.lan]: FAILED! => {"changed": false, "connection": "close", "content_length": "150", "content_type": "text/html", "date": "Fri, 29 Sep 2023 09:42:32 GMT", "elapsed": 0, "msg": "Status code was 502 and not [200]: HTTP Error 502: Bad Gateway", "redirected": false, "server": "nginx", "status": 502, "url": "https://daap1.lan:443/api/v2/ping/"}
```

- 检查您是否已运行 **automation-controller-web** 容器和一个 systemd 服务。

注意

这在常规非特权用户而非系统范围内的级别使用。如果您使用 **su** 切换到运行容器的用户，您必须将 **XDG_RUNTIME_DIR** 环境变量设置为正确的值，以便能够与用户 **systemctl** 单元交互。

```
export XDG_RUNTIME_DIR="/run/user/$UID"
```

```
podman ps | grep web
systemctl --user | grep web
```

没有输出表示问题。

1. 尝试重启 **automation-controller-web** 服务：

```
systemctl start automation-controller-web.service --user
systemctl --user | grep web
systemctl status automation-controller-web.service --user
```

```
Sep 29 10:55:16 daap1.lan automation-controller-web[29875]: nginx: [emerg] bind() to
0.0.0.0:443 failed (98: Address already in use)
Sep 29 10:55:16 daap1.lan automation-controller-web[29875]: nginx: [emerg] bind() to
0.0.0.0:80 failed (98: Address already in use)
```

输出表明端口已经或仍然被其他服务使用。在本例中，**nginx**。

2. 运行：

```
sudo pkill nginx
```

3. 重新启动并状态检查 Web 服务。

普通服务输出应类似于如下，并且应该仍然在运行：

```
Sep 29 10:59:26 daap1.lan automation-controller-web[30274]: WSGI app 0 (mountpoint='/') ready in
3 seconds on interpreter 0x1a458c10 pid: 17 (default app)
Sep 29 10:59:26 daap1.lan automation-controller-web[30274]: WSGI app 0 (mountpoint='/') ready in
3 seconds on interpreter 0x1a458c10 pid: 20 (default app)
Sep 29 10:59:27 daap1.lan automation-controller-web[30274]: 2023-09-29 09:59:27,043 INFO     [-]
daphne.cli Starting server at tcp:port=8051:interface=127.0.>
Sep 29 10:59:27 daap1.lan automation-controller-web[30274]: 2023-09-29 09:59:27,043 INFO
Starting server at tcp:port=8051:interface=127.0.0.1
Sep 29 10:59:27 daap1.lan automation-controller-web[30274]: 2023-09-29 09:59:27,048 INFO     [-]
daphne.server HTTP/2 support not enabled (install the http2 >
Sep 29 10:59:27 daap1.lan automation-controller-web[30274]: 2023-09-29 09:59:27,048 INFO
HTTP/2 support not enabled (install the http2 and tls Twisted ex>
Sep 29 10:59:27 daap1.lan automation-controller-web[30274]: 2023-09-29 09:59:27,049 INFO     [-]
daphne.server Configuring endpoint tcp:port=8051:interface=1>
Sep 29 10:59:27 daap1.lan automation-controller-web[30274]: 2023-09-29 09:59:27,049 INFO
Configuring endpoint tcp:port=8051:interface=127.0.0.1
Sep 29 10:59:27 daap1.lan automation-controller-web[30274]: 2023-09-29 09:59:27,051 INFO     [-]
daphne.server Listening on TCP address 127.0.0.1:8051
Sep 29 10:59:27 daap1.lan automation-controller-web[30274]: 2023-09-29 09:59:27,051 INFO
Listening on TCP address 127.0.0.1:8051
Sep 29 10:59:54 daap1.lan automation-controller-web[30274]: 2023-09-29 09:59:54,139 INFO
success: nginx entered RUNNING state, process has stayed up for > th>
Sep 29 10:59:54 daap1.lan automation-controller-web[30274]: 2023-09-29 09:59:54,139 INFO
success: nginx entered RUNNING state, process has stayed up for > th>
Sep 29 10:59:54 daap1.lan automation-controller-web[30274]: 2023-09-29 09:59:54,139 INFO
success: uwsgi entered RUNNING state, process has stayed up for > th>
```

```
Sep 29 10:59:54 daap1.lan automation-controller-web[30274]: 2023-09-29 09:59:54,139 INFO
success: uwsgi entered RUNNING state, process has stayed up for > th>
Sep 29 10:59:54 daap1.lan automation-controller-web[30274]: 2023-09-29 09:59:54,139 INFO
success: daphne entered RUNNING state, process has stayed up for > t>
Sep 29 10:59:54 daap1.lan automation-controller-web[30274]: 2023-09-29 09:59:54,139 INFO
success: daphne entered RUNNING state, process has stayed up for > t>
Sep 29 10:59:54 daap1.lan automation-controller-web[30274]: 2023-09-29 09:59:54,139 INFO
success: ws-heartbeat entered RUNNING state, process has stayed up f>
Sep 29 10:59:54 daap1.lan automation-controller-web[30274]: 2023-09-29 09:59:54,139 INFO
success: ws-heartbeat entered RUNNING state, process has stayed up f>
Sep 29 10:59:54 daap1.lan automation-controller-web[30274]: 2023-09-29 09:59:54,139 INFO
success: cache-clear entered RUNNING state, process has stayed up fo>
Sep 29 10:59:54 daap1.lan automation-controller-web[30274]: 2023-09-29 09:59:54,139 INFO
success: cache-clear entered RUNNING state, process has stayed up
```

您可以再次运行安装程序，以确保所有安装都如预期安装。

在 Amazon Web Services 中安装容器化 Ansible Automation Platform 时，您会收到没有剩余空间的输出

```
TASK [ansible.containerized_installer.automationcontroller : Create the receptor container]
**************************************************
fatal: [ec2-13-48-25-168.eu-north-1.compute.amazonaws.com]: FAILED! => {"changed": false, "msg":
"Can't create container receptor", "stderr": "Error: creating container storage: creating an ID-mapped
copy of layer \"98955f43cc908bd50ff43585fec2c7dd9445eaf05eecd1e3144f93ffc00ed4ba\": error
during chown: storage-chown-by-maps: lchown usr/local/lib/python3.9/site-
packages/azure/mgmt/network/v2019_11_01/operations/__pycache__/_available_service_aliases_oper
ations.cpython-39.pyc: no space left on device: exit status 1\n", "stderr_lines": ["Error: creating
container storage: creating an ID-mapped copy of layer
\"98955f43cc908bd50ff43585fec2c7dd9445eaf05eecd1e3144f93ffc00ed4ba\": error during chown:
storage-chown-by-maps: lchown usr/local/lib/python3.9/site-
packages/azure/mgmt/network/v2019_11_01/operations/__pycache__/_available_service_aliases_oper
ations.cpython-39.pyc: no space left on device: exit status 1"], "stdout": "", "stdout_lines": []}
```

如果要将 **/home** 文件系统安装到默认的 Amazon Web Services marketplace RHEL 实例中，则可能太小，因为 **/home** 是 root / 文件系统的一部分。您需要留出更多可用空间。本文档为容器化 Ansible Automation Platform 的单节点部署指定最小 40GB。

A.2. 容器化 ANSIBLE AUTOMATION PLATFORM 配置故障排除

有时，安装用于 seeding my Ansible Automation Platform 内容错误。这可以将自身视为类似如下的输出：

```
TASK [infra.controller_configuration.projects : Configure Controller Projects | Wait for finish the
projects creation] ************************************
Friday 29 September 2023  11:02:32 +0100 (0:00:00.443)       0:00:53.521 ******
FAILED - RETRYING: [daap1.lan]: Configure Controller Projects | Wait for finish the projects creation
(1 retries left).
failed: [daap1.lan] (item={'failed': 0, 'started': 1, 'finished': 0, 'ansible_job_id': '536962174348.33944',
'results_file': '/home/aap/.ansible_async/536962174348.33944', 'changed': False,
'__controller_project_item': {'name': 'AAP Config-As-Code Examples', 'organization': 'Default',
'scm_branch': 'main', 'scm_clean': 'no', 'scm_delete_on_update': 'no', 'scm_type': 'git',
'scm_update_on_launch': 'no', 'scm_url': 'https://github.com/user/repo.git'}, 'ansible_loop_var':
'__controller_project_item'}) => {"__projects_job_async_results_item": {"__controller_project_item":
```

{"name": "AAP Config-As-Code Examples", "organization": "Default", "scm_branch": "main", "scm_clean": "no", "scm_delete_on_update": "no", "scm_type": "git", "scm_update_on_launch": "no", "scm_url": "https://github.com/user/repo.git"}, "ansible_job_id": "536962174348.33944", "ansible_loop_var": "__controller_project_item", "changed": false, "failed": 0, "finished": 0, "results_file": "/home/aap/.ansible_async/536962174348.33944", "started": 1}, "ansible_job_id": "536962174348.33944", "ansible_loop_var": "__projects_job_async_results_item", "attempts": 30, "changed": false, "finished": 0, "results_file": "/home/aap/.ansible_async/536962174348.33944", "started": 1, "stderr": "", "stderr_lines": [], "stdout": "", "stdout_lines": []}

infra.controller_configuration.dispatch 角色使用一个异步循环，30 次重试来应用每个配置类型，重试之间的默认延迟为 1 秒。如果配置较大，这可能没有足够的时间来在最后一次重试发生前应用所有内容。

通过将 **controller_configuration_async_delay** 变量设置为 1 秒以外的内容来增加重试延迟。例如，将其设置为 2 秒，使重试时间加倍。执行此操作的位置将位于定义控制器配置的存储库中。它还可以添加到安装程序清单文件的 **[all:vars]** 部分。

一些实例显示不需要额外的修改，然后再次重新运行安装程序。

A.3. 容器化 ANSIBLE AUTOMATION PLATFORM 参考

您能为 Ansible Automation Platform 容器化设计提供架构的详细信息？

我们尽可能使用尽可能多的底层原生 RHEL 技术。对于容器运行时和管理服务，我们使用 Podman。许多 Podman 服务和命令用于显示和调查解决方案。

例如，使用 **podman ps** 和 **podman images** 查看一些基础并运行片段：

```
[aap@daap1 aap]$ podman ps
CONTAINER ID  IMAGE                                                                COMMAND               CREATED
STATUS         PORTS       NAMES
88ed40495117  registry.redhat.io/rhel8/postgresql-13:latest                        run-postgresql        48
minutes ago  Up 47 minutes             postgresql
8f55ba612f04  registry.redhat.io/rhel8/redis-6:latest                              run-redis             47
minutes ago  Up 47 minutes             redis
56c40445c590  registry.redhat.io/ansible-automation-platform-24/ee-supported-rhel8:latest
/usr/bin/receptor...  47 minutes ago  Up 47 minutes               receptor
f346f05d56ee  registry.redhat.io/ansible-automation-platform-24/controller-rhel8:latest
/usr/bin/launch_a...  47 minutes ago  Up 45 minutes               automation-controller-rsyslog
26e3221963e3  registry.redhat.io/ansible-automation-platform-24/controller-rhel8:latest
/usr/bin/launch_a...  46 minutes ago  Up 45 minutes               automation-controller-task
c7ac92a1e8a1  registry.redhat.io/ansible-automation-platform-24/controller-rhel8:latest
/usr/bin/launch_a...  46 minutes ago  Up 28 minutes               automation-controller-web

[aap@daap1 aap]$ podman images
REPOSITORY                                                  TAG         IMAGE ID      CREATED      SIZE
registry.redhat.io/ansible-automation-platform-24/ee-supported-rhel8  latest      b497bdbee59e  10
days ago  3.16 GB
registry.redhat.io/ansible-automation-platform-24/controller-rhel8    latest      ed8ebb1c1baa  10 days
ago  1.48 GB
registry.redhat.io/rhel8/redis-6                            latest      78905519bb05  2 weeks ago  357 MB
registry.redhat.io/rhel8/postgresql-13                      latest      9b65bc3d0413  2 weeks ago  765
MB
[aap@daap1 aap]$
```

容器化 Ansible Automation Platform 作为无根容器运行，以实现最大开箱即用的安全性。这意味着您可以使用任何本地非特权用户帐户安装容器化的 Ansible Automation Platform。只有某些根级别任务需要特权升级，默认情况下不需要直接使用 root 用户。

安装后，您会注意到某些项目已在运行安装程序的文件系统上填充（底层 RHEL 主机）。

```
[aap@daap1 aap]$ tree -L 1
.
├── aap_install.log
├── ansible.cfg
├── collections
├── galaxy.yml
├── inventory
├── LICENSE
├── meta
├── playbooks
├── plugins
├── README.md
├── requirements.yml
├── roles
```

使用 Podman 卷等内容的其他容器化服务驻留在所使用的安装根目录下。以下是进一步参考的一些示例：

containers 目录包含一些用于执行平面的 Podman 细节：

```
containers/
├── podman
├── storage
│   ├── defaultNetworkBackend
│   ├── libpod
│   ├── networks
│   ├── overlay
│   ├── overlay-containers
│   ├── overlay-images
│   ├── overlay-layers
│   ├── storage.lock
│   └── userns.lock
└── storage.conf
```

控制器目录具有一些已安装的配置和运行时数据点：

```
controller/
├── data
│   ├── job_execution
│   ├── projects
│   └── rsyslog
├── etc
│   ├── conf.d
│   ├── launch_awx_task.sh
│   ├── settings.py
│   ├── tower.cert
│   └── tower.key
├── nginx
│   └── etc
├── rsyslog
```

```
│   └── run
└── supervisor
    └── run
```

receptor 目录有自动化网格配置：

```
receptor/
├── etc
│   └── receptor.conf
└── run
    ├── receptor.sock
    └── receptor.sock.lock
```

安装后，您还将在本地用户主目录（如 **.cache** 目录）中找到其他部分：

```
.cache/
├── containers
│   └── short-name-aliases.conf.lock
└── rhsm
    └── rhsm.log
```

正如我们默认以最安全的方式（如无根 Podman）运行一样，我们还可以使用其他服务，如将 **systemd** 作为非特权用户运行。在 **systemd** 下，您可以看到一些可用的组件服务控制：

.config 目录：

```
.config/
├── cni
│   └── net.d
│       └── cni.lock
├── containers
│   ├── auth.json
│   └── containers.conf
└── systemd
    └── user
        ├── automation-controller-rsyslog.service
        ├── automation-controller-task.service
        ├── automation-controller-web.service
        ├── default.target.wants
        ├── podman.service.d
        ├── postgresql.service
        ├── receptor.service
        ├── redis.service
        └── sockets.target.wants
```

这特定于 Podman，符合开放容器项目(OCI)规格。默认情况下，Podman 运行为 root 用户将使用 **/var/lib/containers**，而用于 **$HOME/.local** 下的层次结构。

.local 目录：

```
.local/
└── share
    └── containers
        ├── cache
        ├── podman
```

```
    └── storage

As an example `.local/storage/volumes` contains what the output from `podman volume ls` provides:

[aap@daap1 containers]$ podman volume ls
DRIVER      VOLUME NAME
local       d73d3fe63a957bee04b4853fd38c39bf37c321d14fdab9ee3c9df03645135788
local       postgresql
local       redis_data
local       redis_etc
local       redis_run
```

我们将执行平面与 control plane 主服务(PostgreSQL、Redis、automation controller、receptor、automation hub 和 Event-Driven Ansible 隔离)。

control plane 服务使用标准 Podman 配置(**~/.local/share/containers/storage**)运行。

执行平面服务使用专用配置或存储(**~/aap/containers/storage**)来避免执行 plane 容器可以与 control plane 交互。

如何查看主机资源利用率统计信息？

- 运行：

```
$ podman container stats -a
```

```
podman container stats -a
ID            NAME                        CPU %       MEM USAGE / LIMIT  MEM %       NET IO      BLOCK
IO    PIDS        CPU TIME     AVG CPU %
0d5d8eb93c18  automation-controller-web      0.23%       959.1MB / 3.761GB  25.50%      0B / 0B
0B / 0B     16          20.885142s  1.19%
3429d559836d  automation-controller-rsyslog  0.07%       144.5MB / 3.761GB  3.84%       0B / 0B
0B / 0B     6           4.099565s   0.23%
448d0bae0942  automation-controller-task     1.51%       633.1MB / 3.761GB  16.83%      0B / 0B     0B
/ 0B     33          34.285272s  1.93%
7f140e65b57e  receptor                    0.01%       5.923MB / 3.761GB  0.16%       0B / 0B     0B / 0B
7           1.010613s   0.06%
c1458367ca9c  redis                       0.48%       10.52MB / 3.761GB  0.28%       0B / 0B     0B / 0B     5
9.074042s   0.47%
ef712cc2dc89  postgresql                  0.09%       21.88MB / 3.761GB  0.58%       0B / 0B     0B / 0B
21          15.571059s  0.80%
```

前者是 Dell 销售并提供容器化的 Ansible Automation Platform 解决方案(DAAP)安装并利用 ~1.8Gb RAM 的示例。

使用多少存储以及在哪里？

当我们运行无根 Podman 时，容器卷存储位于位于 **$HOME/.local/share/containers/storage/volumes** 的本地用户下。

1. 查看每个卷运行的详情：

```
$ podman volume ls
```

2. 然后运行：

```
$ podman volume inspect <volume_name>
```

下面是一个示例：

```
$ podman volume inspect postgresql
[
    {
        "Name": "postgresql",
        "Driver": "local",
        "Mountpoint": "/home/aap/.local/share/containers/storage/volumes/postgresql/_data",
        "CreatedAt": "2024-01-08T23:39:24.983964686Z",
        "Labels": {},
        "Scope": "local",
        "Options": {},
        "MountCount": 0,
        "NeedsCopyUp": true
    }
]
```

安装程序创建的几个文件位于 **$HOME/aap/** 中，绑定挂载到不同的正在运行的容器中。

1. 查看与容器运行关联的挂载：

```
$ podman ps --format "{{.ID}}\t{{.Command}}\t{{.Names}}"
```

```
Example:
$ podman ps --format "{{.ID}}\t{{.Command}}\t{{.Names}}"
89e779b81b83 run-postgresql postgresql
4c33cc77ef7d run-redis redis
3d8a028d892d /usr/bin/receptor... receptor
09821701645c /usr/bin/launch_a... automation-controller-rsyslog
a2ddb5cac71b /usr/bin/launch_a... automation-controller-task
fa0029a3b003 /usr/bin/launch_a... automation-controller-web
20f192534691 gunicorn --bind 1... automation-eda-api
f49804c7e6cb daphne -b 127.0.0... automation-eda-daphne
d340b9c1cb74 /bin/sh -c nginx ... automation-eda-web
111f47de5205 aap-eda-manage rq... automation-eda-worker-1
171fcb1785af aap-eda-manage rq... automation-eda-worker-2
049d10555b51 aap-eda-manage rq... automation-eda-activation-worker-1
7a78a41a8425 aap-eda-manage rq... automation-eda-activation-worker-2
da9afa8ef5e2 aap-eda-manage sc... automation-eda-scheduler
8a2958be9baf gunicorn --name p... automation-hub-api
0a8b57581749 gunicorn --name p... automation-hub-content
68005b987498 nginx -g daemon o... automation-hub-web
cb07af77f89f pulpcore-worker automation-hub-worker-1
a3ba05136446 pulpcore-worker automation-hub-worker-2
```

2. 然后运行：

```
$ podman inspect <container_name> | jq -r .[].Mounts[].Source
```

```
Example:
```

```
/home/aap/.local/share/containers/storage/volumes/receptor_run/_data
/home/aap/.local/share/containers/storage/volumes/redis_run/_data
/home/aap/aap/controller/data/rsyslog
/home/aap/aap/controller/etc/tower.key
/home/aap/aap/controller/etc/conf.d/callback_receiver_workers.py
/home/aap/aap/controller/data/job_execution
/home/aap/aap/controller/nginx/etc/controller.conf
/home/aap/aap/controller/etc/conf.d/subscription_usage_model.py
/home/aap/aap/controller/etc/conf.d/cluster_host_id.py
/home/aap/aap/controller/etc/conf.d/insights.py
/home/aap/aap/controller/rsyslog/run
/home/aap/aap/controller/data/projects
/home/aap/aap/controller/etc/settings.py
/home/aap/aap/receptor/etc/receptor.conf
/home/aap/aap/controller/etc/conf.d/execution_environments.py
/home/aap/aap/tls/extracted
/home/aap/aap/controller/supervisor/run
/home/aap/aap/controller/etc/uwsgi.ini
/home/aap/aap/controller/etc/conf.d/container_groups.py
/home/aap/aap/controller/etc/launch_awx_task.sh
/home/aap/aap/controller/etc/tower.cert
```

3. 如果没有安装 **jq** RPM，请使用以下内容安装：

```
$ sudo dnf -y install jq
```