Red Hat Ansible Automation Platform 2.4

コンテナー化された Ansible Automation Platform のインストールガイド

コンテナー化された Ansible Automation Platform のインストールガイド

Last Updated: 2024-10-16

Red Hat Ansible Automation Platform 2.4 コンテナー化された Ansible Automation Platform のインストールガイド

コンテナー化された Ansible Automation Platform のインストールガイド

法律上の通知

概要

このガイドは、Ansible Automation Platform の新しいコンテナー化されたバージョンのインストール要件とプロセスを理解するのに役立ちます。

目次

RED HAT ドキュメントへのフィードバック (英語のみ) 3

第1章 ANSIBLE AUTOMATION PLATFORM のコンテナー化されたインストール 4
1.1. システム要件 4
1.2. コンテナー化されたインストールのための RHEL ホストの準備 4
1.3. ANSIBLE-CORE のインストール 5
1.4. ANSIBLE AUTOMATION PLATFORM のダウンロード 5
1.5. コンテナー化された ANSIBLE AUTOMATION PLATFORM のポストインストール機能の使用 6
1.6. コンテナー化された ANSIBLE AUTOMATION PLATFORM のインストール 7
1.7. AUTOMATION CONTROLLER、AUTOMATION HUB、EVENT-DRIVEN ANSIBLE CONTROLLER へのアクセス 10
1.8. カスタム TLS 証明書の使用 11
1.9. カスタム RECEPTOR 署名キーの使用 12
1.10. AUTOMATION HUB コレクションとコンテナー署名の有効化 12
1.11. 実行ノードの追加 12
1.12. コンテナー化された ANSIBLE AUTOMATION PLATFORM のアンインストール 13

付録A コンテナー化された ANSIBLE AUTOMATION PLATFORM のトラブルシューティング 14
A.1. コンテナー化された ANSIBLE AUTOMATION PLATFORM のインストールのトラブルシューティング 14
A.2. コンテナー化された ANSIBLE AUTOMATION PLATFORM の設定のトラブルシューティング 17
A.3. コンテナー化された ANSIBLE AUTOMATION PLATFORM のリファレンス 17

RED HAT ドキュメントへのフィードバック (英語のみ)

このドキュメントの改善に関するご意見がある場合や、エラーを発見した場合は、https://access.redhat.com から Technical Support チームに連絡してください。

第1章 ANSIBLE AUTOMATION PLATFORM のコンテナー化されたインストール

Ansible Automation Platform は、Ansible を装備した環境に、制御、ナレッジ、委譲の機能を追加して、チームが複雑かつ複数層のデプロイメントを管理できるように支援する商用サービスです。

このガイドは、Ansible Automation Platform の新しいコンテナー化されたバージョンのインストール要件とプロセスを理解するのに役立ちます。この初期バージョンは Ansible Automation Platform 2.4 に基づいており、テクニカルプレビューとしてリリースされています。テクニカルプレビューの内容を理解するには、テクノロジプレビュー機能のサポート範囲 を参照してください。

前提条件

- RHEL 9.2 ベースのホスト。最小限の OS ベースのインストールを推奨します。
- sudo または他の Ansible でサポートされる権限昇格 (sudo を推奨) のある RHEL ホストの非 root ユーザー。このユーザーは、コンテナー化された Ansible Automation Platform のインストールを実行します。
- 非 root ユーザーには、**SSH 公開キー認証** をセットアップすることを推奨しています。非 root ユーザーの SSH 公開キー認証のセットアップに関するガイドラインについては、How to configure SSH public key authentication for passwordless login を参照してください。
- SSH キーは、リモートホストにインストールする場合にのみ必要です。自己完結型のローカル VM ベースのインストールを行う場合は、例のように SSH を必要としない **ansible_connection: local** を使用できます。
- デフォルトのオンラインインストール方法を使用する場合は、RHEL ホストからのインターネットアクセスが必要です。

1.1. システム要件

コンテナー化された Red Hat Ansible Automation Platform をインストールして実行するには、お使いのシステムが以下の最小システム要件を満たしている必要があります。

メモリー	16Gb RAM
CPU	4 CPU
ディスク領域	40 GB
ディスク IOP	1500

1.2. コンテナー化されたインストールのための RHEL ホストの準備

手順

コンテナー化された Ansible Automation Platform は、RHEL ホスト上でコンポーネントサービスを Podman ベースのコンテナーとして実行します。基盤となるホストの準備が完了すると、インストーラーがこれを処理します。以下の手順に従ってください。

1. RHEL ホストに非 root ユーザーとしてログインします。
2. **dnf repolist** を実行して、BaseOS と appstream リポジトリーのみがホスト上でセットアップされ、有効になっていることを検証します。

```
$ 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. これらのリポジトリーのみがホスト OS で利用可能であることを確認してください。確認方法の詳細は、第 10 章 カスタムソフトウェアリポジトリーの管理 を参照してください。
4. ホストに DNS が設定されており、完全修飾ドメイン名 (FQDN) を使用してホスト名と IP を解決できることを確認してください。これは、サービスが相互に通信できるようにするために不可欠です。

unbound DNS の使用

unbound DNS を設定するには、第 2 章 unbound DNS サーバー Red Hat Enterprise Linux 9 のセットアップ を参照してください。

BIND DNS の使用

BIND を使用して DNS を設定するには、第 1 章 BIND DNS サーバー Red Hat Enterprise Linux 9 のセットアップおよび設定 を参照してください。

任意

インストーラーが Ansible Automation Platform サブスクリプションのマニフェストライセンスを自動的に取得して適用できるようにするには、インストーラー用にダウンロードできるマニフェストファイルを生成します。詳細は、第 2 章 マニフェストファイル Red Hat Ansible Automation Platform 2 の取得 を参照してください。

1.3. ANSIBLE-CORE のインストール

手順

1. ansible-core およびその他のツールをインストールします。

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

2. 完全修飾ホスト名を設定します。

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

1.4. ANSIBLE AUTOMATION PLATFORM のダウンロード

手順

1. 最新のインストーラー tarball を access.redhat.com からダウンロードします。これは RHEL ホスト内で直接実行できるため、時間を節約できます。

2. tarball とオプションのマニフェスト zip ファイルをラップトップにダウンロードした場合は、それらを RHEL ホストにコピーします。
インストーラーをファイルシステム上のどこに配置するかを決定します。インストール関連のファイルはこの場所に作成され、初回インストールには少なくとも 10GB が必要です。

3. インストーラーの tarball をインストールディレクトリーに展開し、展開されたディレクトリーに移動します。

   a. オンラインインストーラー

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

   b. バンドルのインストーラー

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

1.5. コンテナー化された ANSIBLE AUTOMATION PLATFORM のポストインストール機能の使用

コンテナー化された Ansible Automation Platform の実験的なポストインストーラー機能を使用して、初回インストール中に設定を定義してロードします。これは configuration-as-code アプローチを使用しており、この場合、単純な YAML ファイルとしてロードされるように設定を定義するだけになります。

1. このオプション機能を使用するには、インベントリーファイル内にある以下の vars のコメントを解除する必要があります。

   ```
   controller_postinstall=true
   ```

2. デフォルトは false であるため、ポストインストーラーをアクティブ化するにはこれを有効にする必要があります。この機能を使用するには Ansible Automation Platform ライセンスが必要です。このライセンスは、自動的にロードできるようにローカルファイルシステム上に配置する必要があります。

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

3. Git ベースのリポジトリーから、configuration-as-code をプルできます。これを行うには、次の変数を設定して、コンテンツの取得元と、Ansible Automation Platform コントローラーにアップロードするためにコンテンツを保存する場所を指定します。

   ```
   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 変数を使用すると、認証情報を含める必要がある postinstall リポジトリーの URL を定義できます。

   ```
   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)
```

注記

ssh ベースの認証を使用する場合、インストーラーは何も設定しないため、インストーラーノードですべてを設定する必要があります。

定義ファイルは infra 認定コレクション を使用します。controller_configuration コレクションは、インストールの一部としてプレインストールされており、インベントリーファイルで指定されたインストールコントローラーの認証情報を使用して Ansible Automation Platform コントローラーにアクセスします。ユーザーは YAML 設定ファイルを指定するだけで済みます。

認証情報、LDAP 設定、ユーザーとチーム、組織、プロジェクト、インベントリーとホスト、ジョブとワークフローテンプレートなどの Ansible Automation Platform 設定属性をセットアップできます。

以下の例は、コントローラージョブテンプレートを定義およびロードするサンプルの **your-config.yml** ファイルを示しています。この例は、Ansible Automation Platform インストールで提供される事前にロードされたデモの例に対して加えられた簡単な変更を示しています。

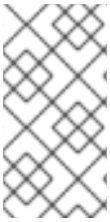
```
/full_path_to_your_configuration_as_code/
├── controller
    └── job_templates.yml
```

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

1.6. コンテナー化された ANSIBLE AUTOMATION PLATFORM のインストール

Ansible Automation Platform のインストールは、インベントリーファイルによって制御されます。インベントリーファイルは、使用および作成されるホストとコンテナー、コンポーネントの変数、およびインストールのカスタマイズに必要なその他の情報を定義します。

便宜上、サンプルのインベントリーファイルが提供されており、コピーおよび変更してすぐに使い始めることができます。



注記

インベントリーファイルには、デフォルトのデータベースはありません。インベントリーファイルの指示に従って、内部で提供される postgres を選択するか、外部で管理およびサポートされる独自のデータベースオプションを指定するかを適切に選択する必要があります。

< > プレースホルダーを特定の変数に置き換え、ニーズに応じて行のコメントを解除して、インベントリーファイルを編集します。

```
# This is the AAP installer inventory file
# Please consult the docs if you're unsure what to add
```

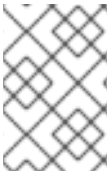
```
# For all optional variables please consult the included README.md

# 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>
```

以下のコマンドを使用して、コンテナー化された Ansible Automation Platform をインストールします。

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



注記

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.

最大 4 つの v (-vvvv) まで詳細度を上げて、インストールプロセスの詳細を表示できます。

注記

これによりインストール時間が大幅に長くなる可能性があるため、必要な場合または Red Hat サポートからリクエストされた場合にのみ使用することを推奨します。

1.7. AUTOMATION CONTROLLER、AUTOMATION HUB、EVENT-DRIVEN ANSIBLE CONTROLLER へのアクセス

インストールが完了すると、以下がデフォルトのプロトコルとポートとして使用されます。

- HTTP/https プロトコル
- Automation Controller 用のポート 8080/8443
- Automation Hub 用のポート 8081/8444
- Event-Driven Ansible Controller 用のポート 8082/8445

これらは変更可能です。詳細は、**README.md** を参照してください。ポートの競合またはその他の要因によりデフォルトを変更する必要がない限り、デフォルトのままにすることを推奨します。

Automation Controller UI へのアクセス

Automation Controller UI は、デフォルトで以下の場所から利用できます。

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

controller_admin_password 用に作成したパスワードを使用して、管理者ユーザーとしてログインします。

インストールの一部としてライセンスマニフェストを指定した場合は、Ansible Automation Platform ダッシュボードが表示されます。ライセンスファイルを指定しなかった場合は、**サブスクリプション** 画面が表示されるので、ライセンスの詳細を指定する必要があります。詳細は、第 1 章Red Hat Ansible Automation Platform のアクティブ化 を参照してください。

Automation Hub UI へのアクセス

Automation Hub UI は、デフォルトで以下の場所から利用できます。

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

hub_admin_password 用に作成したパスワードを使用して、管理者ユーザーとしてログインします。

Event-Driven Ansible UI へのアクセス

Event-Driven Ansible UI は、デフォルトで以下の場所から入手できます。

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

eda_admin_password 用に作成したパスワードを使用して、管理者ユーザーとしてログインします。

1.8. カスタム TLS 証明書の使用

デフォルトでは、インストーラーはカスタム認証局 (CA) によって署名されたすべてのサービスの TLS 証明書とキーを生成します。サービスごとにカスタム TLS 証明書/キーを指定できます。その証明書がカスタム CA によって署名されている場合は、CA TLS 証明書とキーを指定する必要があります。

- 認証局

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

- Automation Controller

```
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
```

-

- Automation 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. カスタム RECEPTOR 署名キーの使用

ceptor_disable_signing=true が設定されていない限り、receptor 署名がデフォルトで有効になり、RSA キーペア (パブリック/プライベート) がインストーラーによって生成されます。ただし、path 変数を設定することで、カスタムの RSA 公開鍵/秘密鍵を指定できます。

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

1.10. AUTOMATION HUB コレクションとコンテナー署名の有効化

Automation Hub を使用すると、ansible コレクションとコンテナーイメージに署名できます。この機能はデフォルトでは有効になっていないため、GPG キーを指定する必要があります。

```
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
```

GPG キーがパスフレーズで保護されている場合は、パスフレーズを指定する必要があります。

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

1.11. 実行ノードの追加

コンテナー化されたインストーラーは、リモート実行ノードをデプロイできます。これは、ansible インベントリーファイルの execution_nodes グループによって処理されます。

```
[execution_nodes]
fqdn_of_your_execution_host
```

実行ノードは、デフォルトでは、ポート 27199 (TCP) で実行される実行タイプとして設定されます。これは、以下の変数で変更できます。

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

receptor タイプの値は、プロトコルが TCP または UDP のいずれかですが、実行またはホップのいずれかになります。デフォルトでは、**execution_nodes** グループのノードはコントローラーノードのピアとして追加されます。ただし、**receptor_peers** 変数を使用してピア設定を変更できます。

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

1.12. コンテナー化された ANSIBLE AUTOMATION PLATFORM のアンインストール

コンテナー化されたデプロイメントをアンインストールするには、**uninstall.yml** Playbook を実行します。

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

これにより、すべての systemd ユニットとコンテナーが停止され、コンテナー化されたインストーラーが使用する以下のようなリソースがすべて削除されます。

- config およびデータのディレクトリー/ファイル
- systemd ユニットファイル
- podman のコンテナーとイメージ
- RPM パッケージ

コンテナーイメージを保持するには、**container_keep_images** 変数を true に設定します。

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

postgresql データベースを保持するには、**postgresql_keep_databases** 変数を true に設定します。

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

注記

自動生成されるものではなく、同じ django 秘密鍵値を使用する必要があります。

付録A コンテナー化された ANSIBLE AUTOMATION PLATFORM のトラブルシューティング

この情報を使用して、コンテナー化された Ansible Automation Platform のインストールのトラブルシューティングを行います。

A.1. コンテナー化された ANSIBLE AUTOMATION PLATFORM のインストールのトラブルシューティング

インストールに時間がかかったり、エラーが発生したりする場合は、何を確認すればよいですか?

1. システムがインストールガイドに記載されている最小要件を満たしていることを確認してください。不適切なストレージの選択や、多数のホストに分散する際の高レイテンシーなどの項目はすべて、大きな影響を及ぼします。

2. ローカルインストーラー **ansible.cfg** 内で特に変更されていない限り、デフォルトで **./aap_install.log** に配置されているインストールログファイルを確認します。

3. タスクプロファイリングコールバックをアドホックベースで有効にして、インストールプログラムが最も多くの時間を費やしている場所の概要を示します。これを行うには、ローカルの **ansible.cfg** ファイルを使用します。**[defaults]** セクションの下に次のようなコールバック行を追加します。

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

Automation Controller が 413 エラーを返します。

このエラーは、**manifest.zip** ライセンスファイルが **nginx_client_max_body_size** 設定より大きいために発生します。このエラーが発生した場合は、インストールインベントリーファイルを変更して次の変数を含める必要があります。

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

現在のデフォルト設定である **20m** でこの問題を回避できるはずです。

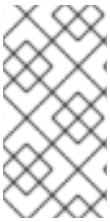
コントローラー UI に移動すると、“502 Bad Gateway” というエラーメッセージが表示され、インストールが失敗しました。

このようなエラーが発生する可能性があり、インストールアプリケーションの出力には、以下のように表示されます。

```
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/"}
```

- 実行中の **automation-controller-web** コンテナーと systemd サービスがあるか確認します。



注記

これは、システム全体のレベルではなく、通常の権限のないユーザーで使用されます。**su** を使用してコンテナーを実行しているユーザーに切り替えた場合は、ユーザーの **systemctl** ユニットと対話できるように、**XDG_RUNTIME_DIR** 環境変数を正しい値に設定する必要があります。

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

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

出力がない場合は、問題があることを示しています。

1. **automation-controller-web** サービスを再起動してみてください。

   ```
   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)
   ```

   出力は、ポートがすでに別のサービスによって使用されているか、引き続き使用されていることを示します。この場合は **nginx** を使用します。

2. 以下を実行します。

   ```
   sudo pkill nginx
   ```

3. Web サービスを再起動してステータスを再度確認します。

通常のサービス出力は次のようになり、引き続き実行されているはずです。

```
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
```

インストールプログラムを再度実行して、すべてが期待どおりにインストールされることを確認できます。

Amazon Web Services にコンテナー化された Ansible Automation Platform をインストールしようとすると、デバイスに空き容量がないという出力が表示されます。

```
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": []}
```

/home ファイルシステムをデフォルトの Amazon Web Services マーケットプレイスの RHEL インスタンスにインストールする場合、**/home** は root / ファイルシステムの一部であるため、サイズが小さすぎる可能性があります。より多くのスペースを確保する必要があります。ドキュメントでは、コンテナー化された Ansible Automation Platform のシングルノードデプロイメントに最低 40 GB を指定しています。

A.2. コンテナー化された ANSIBLE AUTOMATION PLATFORM の設定のトラブルシューティング

Ansible Automation Platform コンテンツをシードするためのインストール後にエラーが発生することがあります。これは次のような出力として現れる可能性があります。

```
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 ロールは、各設定タイプを適用するために 30 回の再試行を伴う非同期ループを使用し、再試行間のデフォルトの遅延は 1 秒です。設定が大きい場合、最後の再試行が発生する前にすべてを適用するには時間が足りない可能性があります。

controller_configuration_async_delay 変数を 1 秒以外の値に設定して、再試行の遅延時間を増やします。たとえば、2 秒に設定すると、再試行時間が 2 倍になります。これを行う場所は、コントローラー設定が定義されているリポジトリー内になります。インストールプログラムインベントリーファイルの **[all:vars]** セクションに追加することもできます。

いくつかの例では、追加の変更は必要なく、インストールプログラムを再実行すると機能することが示されています。

A.3. コンテナー化された ANSIBLE AUTOMATION PLATFORM のリファレンス

Ansible Automation Platform のコンテナー化設計のアーキテクチャーの詳細を教えていただけますか?

Red Hat では、基盤となるネイティブ RHEL テクノロジーを可能な限り多用しています。コンテナーのランタイムとサービスの管理には Podman を使用します。ソリューションを表示および調査するために、多くの Podman サービスとコマンドが使用されます。

たとえば、**podman ps** と **podman images** を使用して、基礎部分と実行部分の一部を確認します。

```
[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]$
```

コンテナー化された Ansible Automation Platform は、セキュリティーを最大限に高めてすぐに使用できるように、ルートレスコンテナーとして実行されます。つまり、ローカルの権限のないユーザーアカウントを使用して、コンテナー化された Ansible Automation Platform をインストールできます。権限の昇格は、特定の root レベルのタスクにのみ必要であり、デフォルトでは root を直接使用する場合には必要ありません。

インストールが完了すると、インストールプログラムが実行されるファイルシステム (基盤となる RHEL ホスト) に特定の項目が取り込まれていることを確認できます。

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

Podman ボリュームなどを利用するその他のコンテナー化されたサービスは、使用されるインストール root ディレクトリーの下に存在します。詳細な参照情報として、以下にいくつかの例を示します。

コンテナーディレクトリーには、実行プレーンに使用およびインストールされる Podman の詳細の一部が含まれています。

```
containers/
```

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

コントローラーディレクトリーには、インストールされた設定とランタイムデータポイントの一部が含まれています。

```
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
```

レセプターディレクトリーには、自動化メッシュ設定があります。

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

インストール後、ローカルユーザーのホームディレクトリーに、**.cache** ディレクトリーなどの他の部分も見つかります。

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

デフォルトでは、ルートレス Podman などの最も安全な方法で実行されるため、非特権ユーザーとして **systemd** を実行するなどの他のサービスも使用できます。**systemd** では、利用可能なコンポーネントサービスコントロールの一部を以下のように確認できます。

以下は、**.config** ディレクトリーです。

```
.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
```

これは Podman に固有のものであり、Open Container Initiative (OCI) 仕様に準拠しています。root ユーザーとして実行される Podman はデフォルトで **/var/lib/containers** を使用しますが、標準ユーザーの場合は **$HOME/.local** の下の階層が使用されます。

以下は、**.local** ディレクトリーです。

```
.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
```

実行プレーンをコントロールプレーンのメインサービス (PostgreSQL、Redis、Automation Controller、レセプター、Automation Hub、Event-Driven Ansible) から分離します。

コントロールプレーンサービスは、標準の Podman 設定 (~/**.local/share/containers/storage**) で実行されます。

実行プレーンサービスは、実行プレーンコンテナーがコントロールプレーンと対話できないように、専用の設定またはストレージ (~/**aap/containers/storage**) を使用します。

ホストリソースの使用率統計情報を確認するにはどうすればよいですか?

- 以下を実行します。

```
$ 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%
```

Dell が販売および提供しているコンテナー化された Ansible Automation Platform ソリューション (DAAP) のインストールにおける前の例では、約 1.8 Gb の RAM を使用しています。

ストレージは、どのくらいの量がどこで使用されていますか?

ルートレス Podman を実行すると、コンテナーボリュームストレージはローカルユーザーの **$HOME/.local/share/containers/storage/volumes** にあります。

1. 各ボリュームの詳細を表示するには、次のコマンドを実行します。

   ```
   $ podman volume ls
   ```

2. 次に、以下を実行します。

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

以下に例を示します。

```
$ 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
    }
]
```

インストールプログラムによって作成されたいくつかのファイルは **$HOME/aap/** にあり、実行中のさまざまなコンテナーにバインドマウントされます。

1. コンテナーに関連付けられたマウントを表示するには、次のコマンドを実行します。

```
$ 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. 次に、以下を実行します。

```
$ 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. **jq** RPM がインストールされていない場合は、次のコマンドでインストールします。

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