Red Hat Ansible Automation Platform 2.5

コンテナーインストール

コンテナー化されたバージョンの Ansible Automation Platform のインストール

Last Updated: 2025-09-23

Red Hat Ansible Automation Platform 2.5 コンテナーインストール

コンテナー化されたバージョンの Ansible Automation Platform のインストール

Legal Notice

Abstract

このガイドは、コンテナー化されたバージョンの Ansible Automation Platform 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。

Table of Contents

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 からテクニカルサポートに連絡してリクエストを送信してください。

第1章 ANSIBLE AUTOMATION PLATFORM のライセンス、更新、サポートの管理

Ansible はオープンソースソフトウェアプロジェクトであり、Ansible ソースコード に記載されているように、GNU General Public License バージョン 3 に基づいてライセンスが付与されます。

Ansible Automation Platform をインストールする前に、有効なサブスクリプションが割り当てられている必要があります。

詳細は、サブスクリプションの割り当て を参照してください。

1.1. 試用と評価

Ansible Automation Platform を実行するにはライセンスが必要です。まずは無料試用版ライセンスから始めることができます。

- Ansible Automation Platform の試用版ライセンスは、Red Hat product trial center で入手できます。
- 試用版ライセンスや Ansible Automation Platform の評価時には、サポートはありません。

1.2. コンポーネントライセンス

Ansible Automation Platform に含まれるコンポーネントのライセンス情報を確認するには、**/usr/share/doc/automation-controller-<version>/README** を参照してください。

<version> はインストールした Automation Controller のバージョンです。

特定のライセンスを表示するには、**/usr/share/doc/automation-controller-<version>/*.txt** を参照してください。

ここで、* は参照するライセンスファイル名です。

1.3. ライセンスでのノードの数え方

Ansible Automation Platform のライセンスでは、サブスクリプションの一部として管理できる管理対象ノードの数が定義されています。

通常のライセンスでは「ライセンス数: 500」と記載され、最大管理対象ノード数が 500 に設定されます。

ライセンス付与の対象となる管理対象ノードの要件の詳細は、How are "managed nodes" defined as part of the Red Hat Ansible Automation Platform offering を参照してください。

注記

Ansible は、ノード数を再利用したり、自動化されたホストをリセットしたりしません。

1.4. サブスクリプションタイプ

Red Hat Ansible Automation Platform は、年間サブスクリプション契約をベースに、さまざまなサポートレベルおよびマシン数で提供されます。

- **Standard:**
  - あらゆる規模の環境の管理
  - エンタープライズサポート (週 5、1 日 8 時間) および SLA
  - メンテナンスおよびアップグレード込み
  - 製品サポート利用規約 で SLA を確認してください。
  - Red Hat サポートにおける重大度レベルの定義 を確認してください。
- **Premium:**
  - ミッションクリティカルな環境を含むあらゆる規模の環境の管理
  - プレミアムサポート (年中無休) および SLA
  - メンテナンスおよびアップグレード込み
  - 製品サポート利用規約 で SLA を確認してください。
  - Red Hat サポートにおける重大度レベルの定義 を確認してください。

すべてのサブスクリプションレベルに、Automation Controller、Ansible、および Ansible Automation Platform の他のコンポーネントの定期的な更新とリリースが含まれています。

詳細は、Red Hat カスタマーポータル または Ansible サイト から Ansible チームにお問い合わせください。

1.5. RED HAT ANSIBLE AUTOMATION PLATFORM サブスクリプションの割り当て

Red Hat Ansible Automation Platform をインストールする前に、全ノードに有効なサブスクリプションが割り当てられている **必要があります**。Ansible Automation Platform サブスクリプションを割り当てると、インストールを続行するために必要なサブスクリプション専用のリソースにアクセスできるようになります。

手順

1. システムが登録されていることを確認します。

```
$ sudo subscription-manager register --username <$INSERT_USERNAME_HERE> --password <$INSERT_PASSWORD_HERE>
```

2. Red Hat Ansible Automation Platform サブスクリプションの **pool_id** を取得します。

```
$ sudo subscription-manager list --available --all | grep "Ansible Automation Platform" -B 3 -A 6
```

注記

Ansible Automation Platform サブスクリプションのアタッチが失敗する可能性があるため、MCT4022 をサブスクリプションの **pool_id** として使用しないでください。

3. サブスクリプションを割り当てます。

```
$ sudo subscription-manager attach --pool=<pool_id>
```

これで、Red Hat Ansible Automation Platform サブスクリプションがすべてのノードに割り当てられました。

4. このサブスクリプションを削除するには、次のコマンドを入力します。

```
$ sudo subscription-manager remove --pool=<pool_id>
```

検証

- サブスクリプションが正常に割り当てられたことを確認します。

```
$ sudo subscription-manager list --consumed
```

トラブルシューティング

- Ansible Automation Platform インストーラーにバンドルされている特定のパッケージが見つからない場合や、**Repositories disabled by configuration** というメッセージが表示される場合は、次のコマンドを使用してリポジトリーを有効にしてみてください。
  Red Hat Ansible Automation Platform 2.5 for RHEL 8

```
$ sudo subscription-manager repos --enable ansible-automation-platform-2.5-for-rhel-8-x86_64-rpms
```

Red Hat Ansible Automation Platform 2.5 for RHEL 9

```
$ sudo subscription-manager repos --enable ansible-automation-platform-2.5-for-rhel-9-x86_64-rpms
```

1.6. マニフェストファイルの取得

サブスクリプションマニフェストは、Red Hat Subscription Management の サブスクリプション割り当て セクションで取得できます。サブスクリプションの割り当てを取得したら、そのマニフェストファイルをダウンロードしてアップロードし、Ansible Automation Platform のライセンス認証を行うことができます。

まず、管理者ユーザーアカウントを使用して Red Hat カスタマーポータル にログインし、このセクションの手順に従います。

1.6.1. サブスクリプションの割り当ての作成

新しいサブスクリプション割り当てを作成すると、現在オフラインまたはエアギャップ状態のシステムにサイドサブスクリプションとエンタイトルメントを設定できます。これは、マニフェストをダウンロードして Ansible Automation Platform にアップロードする前に必要です。

手順

1. サブスクリプションの割り当て ページで、**新規サブスクリプションの割り当て** をクリックします。

2. 割り当ての名前を入力し、後で検索できるようにします。
3. 管理アプリケーションとして、**Type: Satellite 6.16** を選択します。
4. **Create** をクリックします。

次のステップ

- サブスクリプションを追加します。

1.6.2. サブスクリプション割り当てへのサブスクリプションの追加

割り当てが作成されたら、Ansible Automation Platform を適切に実行するために必要なサブスクリプションを追加できます。この手順は、マニフェストをダウンロードして Ansible Automation Platform に追加する前に必要です。

手順

1. サブスクリプション割り当て ページで、サブスクリプションを追加する **サブスクリプション割り当て** の名前をクリックします。
2. **Subscriptions** タブをクリックします。
3. **Add Subscriptions** をクリックします。
4. 追加する予定の Ansible Automation Platform エンタイトルメントの数を入力します。
5. **Submit** をクリックします。

次のステップ

- マニフェストファイルをダウンロードします。

1.6.3. マニフェストファイルのダウンロード

割り当てを作成して、適切なサブスクリプションを取得したら、Red Hat サブスクリプション管理からマニフェストをダウンロードできます。

手順

1. サブスクリプションの割り当て ページで、マニフェストを生成する **サブスクリプション割り当て** の名前をクリックします。
2. **Subscriptions** タブをクリックします。
3. **マニフェストのエクスポート** をクリックして、マニフェストファイルをダウンロードします。**manifest**<allocation name>_<date>.zip_ というファイルが、デフォルトのダウンロードフォルダーにダウンロードされます。

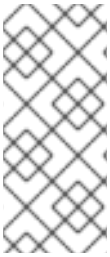
次のステップ

- マニフェストファイルをアップロードします。

1.7. RED HAT ANSIBLE AUTOMATION PLATFORM のライセンス認証

組織管理者の場合は、サービスアカウントを作成 し、クライアント ID とクライアントシークレットを使用してサブスクリプションを有効にする必要があります。

管理者アクセス権がない場合は、Client ID および Client secret フィールドに Red Hat のユーザー名とパスワードを入力して、サブスクリプションを検索し、Ansible Automation Platform インスタンスに追加できます。



注記

クライアント ID とクライアントシークレットを入力してもサブスクリプションが見つからない場合は、サービスアカウントに正しい権限が設定されていない可能性があります。サービスアカウントの詳細とトラブルシューティングのガイダンスは、Configure Ansible Automation Platform to authenticate through service account credentials を参照してください。

Red Hat Satellite の場合は、以下のフィールドに Satellite ユーザー名と Satellite パスワードを入力します。

Red Hat Ansible Automation Platform は、利用可能なサブスクリプションまたはサブスクリプションマニフェストを使用して、Ansible Automation Platform の使用を承認します。サブスクリプションを取得するには、次のいずれかを実行できます。

1. Ansible Automation Platform を起動するときに、Red Hat のユーザー名とパスワード、サービスアカウントの認証情報、または Satellite の認証情報を使用します。
2. Red Hat Ansible Automation Platform インターフェイスを使用するか、Ansible Playbook で手動でサブスクリプションマニフェストファイルをアップロードします。

1.7.1. 認証情報によるライセンス認証

Ansible Automation Platform を初めて起動すると、Ansible Automation Platform Subscription 画面が自動的に表示されます。組織管理者の場合は、Red Hat サービスアカウントを使用してサブスクリプションを取得し、Ansible Automation Platform に直接インポートできます。

管理者アクセス権がない場合は、Client ID および Client secret フィールドにそれぞれ Red Hat のユーザー名とパスワードを入力して、サブスクリプションを検索し、Ansible Automation Platform インスタンスに追加できます。

注記

初めてログインしてプラットフォームのライセンス認証を行うと、デフォルトで Automation Analytics がオプトインされます。これは、Red Hat がユーザーエクスペリエンスを大きく改善し、製品を改良する上で役立ちます。Ansible Automation Platform のライセンス認証後、次の操作を行うことでオプトアウトできます。

1. ナビゲーションパネルから、**Settings → Automation Execution → System** を選択します。
2. **Gather data for Automation Analytics**オプションのチェックボックスをオフにします。
3. **Save** をクリックします。

手順

1. Red Hat Ansible Automation Platform にログインします。
2. **Service Account / Red Hat Satellite**を選択します。
3. **Client ID / Satellite username**と **Client secret / Satellite password**を入力します。
4. **Subscription** リストからサブスクリプションを選択します。

注記

クラスターノードが Subscription Manager を通じて Satellite に登録されている場合は、Satellite のユーザー名とパスワードを使用することもできます。

5. 使用許諾契約書を確認し、**I agree to the End User License Agreement**を選択します。
6. **Finish** をクリックします。

検証

サブスクリプションが承認されると、サブスクリプションの詳細が表示されます。**Compliant** のステータスは、サブスクリプションが、サブスクリプションカウント内で自動化したホストの数に準拠していることを示します。それ以外の場合、ステータスは **Out of Compliance** と表示され、サブスクリプション内のホスト数を超えていることを示します。表示されるその他の重要な情報は次のとおりです。

自動化されたホスト
ライセンス数を消費するジョブによって自動化されたホスト数

インポートされたホスト
すべてのインベントリーソースを考慮したホスト数 (残りのホストには影響しません)

残りのホスト
合計ホスト数から自動化されたホストを差し引いた数

1.7.2. マニフェストファイルによるライセンス認証

サブスクリプションマニフェストがある場合は、Red Hat Ansible Automation Platform インターフェイスを使用してマニフェストファイルをアップロードできます。

注記

初めてログインしてプラットフォームのライセンス認証を行うと、デフォルトで Automation Analytics がオプトインされます。これは、Red Hat がユーザーエクスペリエンスを大きく改善し、製品を改良する上で役立ちます。Ansible Automation Platform のライセンス認証後、次の操作を行うことでオプトアウトできます。

1. ナビゲーションパネルから、**Settings → Automation Execution → System** を選択します。
2. **Gather data for Automation Analytics**オプションのチェックボックスをオフにします。
3. **Save** をクリックします。

前提条件

Red Hat カスタマーポータルから Red Hat サブスクリプションマニフェストファイルをエクスポートしている。詳細は、マニフェストファイルの取得 を参照してください。

手順

1. Red Hat Ansible Automation Platform にログインします。
2. マニフェストファイルの入力をすぐに求められなかった場合は、**Settings → Subscription** に移動します。
3. **Subscription manifest** を選択します。
4. **Browse** をクリックして、マニフェストファイルを選択します。
5. 使用許諾契約書を確認し、**I agree to the End User License Agreement**を選択します。
6. **Finish** をクリックします。

注記

ライセンスページで **BROWSE** ボタンが無効になっている場合は、**USERNAME** フィールドおよび **PASSWORD** フィールドをクリアします。

検証

サブスクリプションが承認されると、サブスクリプションの詳細が表示されます。**Compliant** のステータスは、サブスクリプションが、サブスクリプションカウント内で自動化したホストの数に準拠していることを示します。それ以外の場合、ステータスは **Out of Compliance** と表示され、サブスクリプション内のホスト数を超えていることを示します。表示されるその他の重要な情報は次のとおりです。

自動化されたホスト
ライセンス数を消費するジョブによって自動化されたホスト数

インポートされたホスト
すべてのインベントリーソースを考慮したホスト数 (残りのホストには影響しません)

残りのホスト
合計ホスト数から自動化されたホストを差し引いた数

次のステップ

- ナビゲーションパネルから **Settings → Subscription** を選択し、**Edit subscription** をクリックすると、ライセンス画面に戻ることができます。

第2章 ANSIBLE AUTOMATION PLATFORM のコンテナーインストール

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、コンテナー化されたバージョンの Ansible Automation Platform 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。

注記

2.4 コンテナー化された Ansible Automation Platform テクノロジープレビューから 2.5 コンテナー化された Ansible Automation Platform へのアップグレードはサポートされていません。

2.1. テスト済みのデプロイメントモデル

Red Hat は、定義済みのトポロジーセットを使用して Ansible Automation Platform 2.5 をテストし、推奨構成のデプロイメントオプションを提供しています。サポートされているトポロジーには、インフラストラクチャートポロジー図、テスト済みのシステム設定、サンプルインベントリーファイル、およびネットワークポート情報が含まれています。

コンテナー化された Ansible Automation Platform には、次の 2 つのインフラストラクチャートポロジー形態があります。

1. グロース - (オールインワン) Ansible Automation Platform の使用を開始する組織を対象としています。このトポロジーを使用すると、小さなフットプリントでデプロイできます。

2. エンタープライズ - 大規模な自動化のために冗長性や大きな計算能力を備えた Ansible Automation Platform をデプロイする必要がある組織を対象としています。これは、将来を考慮し、スケールアウトに対応したアーキテクチャーです。

コンテナー化された Ansible Automation Platform のテスト済みデプロイメントトポロジーの詳細は、**テスト済みのデプロイメントモデル** の コンテナートポロジー を参照してください。

2.2. システム要件

コンテナー化された Ansible Automation Platform のインストールを計画する際に、この情報を使用してください。

2.2.1. 前提条件

- Red Hat Enterprise Linux ホストに専用の非 root ユーザーが設定されている。
    - このユーザーには、インストール中に管理タスクを実行するために、**sudo** または Ansible でサポートされているその他の特権昇格 (**sudo** を推奨) が必要です。
    - 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。
    - このユーザーは、Ansible Automation Platform を実行するコンテナーのサービスアカウントでもあります。

- 各管理対象ノードに専用のユーザーが設定されている。Ansible Automation Platform は、このユーザーとして接続し、ノード上でタスクを実行します。各ノードで専用ユーザーを設定する方法の詳細は、コンテナーインストールのための管理対象ノードの準備 を参照してください。
- リモートホストのインストールで、非 root ユーザーに対して SSH 公開鍵認証が設定されていることを確認する。root 以外のユーザーに対する SSH 公開鍵認証の設定に関するガイドラインは、How to configure SSH public key authentication for passwordless login を参照してください。
- デフォルトのオンラインインストール方法を使用している場合、Red Hat Enterprise Linux ホストからインターネットにアクセスできることを確認する。
- ファイアウォールが設定されている場合、適切なネットワークポートが開いていることを確認する。開くポートの詳細は、**テスト済みのデプロイメントモデル** の コンテナートポロジー を参照してください。

重要

NFS 共有へのコンテナーイメージの保存は Podman ではサポートされていません。ユーザーのホームディレクトリーに NFS 共有を使用するには、NFS 共有の外部に Podman のストレージバックエンドパスを設定してください。詳細は、Rootless Podman and NFS を参照してください。

2.2.2. Ansible Automation Platform のシステム要件

お使いのシステムは、Red Hat Ansible Automation Platform をインストールして実行するために、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。

表2.1 システムの設定

Type	説明	注記
Subscription	• 有効な Red Hat Ansible Automation Platform サブスクリプション • 有効な Red Hat Enterprise Linux サブスクリプション (BaseOS および AppStream リポジトリーを使用するため)	
オペレーティングシステム	• Red Hat Enterprise Linux 9.2 またはそれ以降の Red Hat Enterprise Linux 9 のマイナーバージョン。 • Red Hat Enterprise Linux 10 またはそれ以降の Red Hat Enterprise Linux 10 のマイナーバージョン。	

Type	説明	注記
CPU アーキテクチャー	x86_64、AArch64、s390x (IBM Z)、ppc64le (IBM Power)	
ansible-core	• インストール用: **ansible-core** バージョン 2.14。 • Ansible Automation Platform の稼働用: **ansible-core** バージョン 2.16。	• インストールプログラムは、RHEL 9 AppStream リポジトリーの **ansible-core** 2.14 パッケージを使用します。 • Ansible Automation Platform には、その稼働のために使用される **ansible-core** バージョン 2.16 がバンドルされています。そのため、このバージョンを手動でインストールする必要はありません。
ブラウザー	現在サポートされている Mozilla Firefox または Google Chrome のバージョン。	
データベース	PostgreSQL 15	外部 (お客様がサポートする) データベースは ICU をサポートしている必要があります。

各仮想マシン (VM) のシステム要件を以下に示します。

表2.2 仮想マシンの要件

要件	最小要件
RAM	16 GB
CPU	4

要件	最小要件
ローカルディスク	• 使用可能なディスク容量合計: 60 GB • インストールディレクトリー: 15 GB (専用パーティションの場合) • オンラインインストール用の **/var/tmp**: 1 GB • オフラインまたはバンドルインストール用の **/var/tmp**: 3 GB • オフラインまたはバンドルインストール用の一時ディレクトリー (デフォルトは **/tmp**): 10 GB
ディスク IOPS	3000

注記

hub_seed_collections=true を使用してグロートポロジーのバンドルインストールを実行する場合は、32 GB RAM を使用することが推奨されます。この設定では、インストール時間が長くなり、コレクションのシードが完了するまで 45 分以上かかる場合があります。

2.2.3. データベース要件

Ansible Automation Platform は、次の 2 種類のデータベースに対応しています。

1. Ansible Automation Platform でインストールされたデータベース - このデータベースは、Red Hat が提供する PostgreSQL パッケージを使用して Ansible Automation Platform インストール中に行われた PostgreSQL インストールで構成されます。
2. お客様が用意または設定したデータベース - これは、ベアメタル、仮想マシン、コンテナー、またはクラウドホストサービス上の、お客様が用意する外部データベースです。

Ansible Automation Platform では、お客様が用意する (外部) データベースが ICU をサポートしている必要があります。

関連情報

- Red Hat Ansible Automation Platform データベースの対象範囲
- お客様が用意する (外部) データベースの設定

2.3. コンテナーインストールに向けた RED HAT ENTERPRISE LINUX ホストの準備

コンテナー化された Ansible Automation Platform は、Red Hat Enterprise Linux ホスト上で Podman ベースのコンテナーとしてコンポーネントサービスを実行します。インストールが正常に行われるように Red Hat Enterprise Linux ホストを準備します。

手順

1. 非 root ユーザーとして Red Hat Enterprise Linux ホストにログインします。
2. ホストに関連付けられたホスト名が完全修飾ドメイン名 (FQDN) として設定されていることを確認します。
   a. ホストに関連付けられているホスト名を確認するには、次のコマンドを実行します。

   ```
   hostname -f
   ```

   出力例:

   ```
   aap.example.org
   ```

   b. ホスト名が FQDN でない場合は、次のコマンドで設定できます。

   ```
   $ sudo hostnamectl set-hostname <your_hostname>
   ```

3. Red Hat Enterprise Linux ホストを **subscription-manager** に登録します。

   ```
   $ sudo subscription-manager register
   ```

4. ホスト上で BaseOS リポジトリーと AppStream リポジトリーのみが有効になっていることを確認します。

   ```
   $ sudo dnf repolist
   ```

   出力例:

   ```
   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)
   ```

   - 非接続インストールの場合は、RPM ソースの依存関係の取得と設定 の手順に従って、これらのリポジトリーにアクセスします。

5. ホストが DNS を使用してホスト名と IP アドレスを解決できることを確認します。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。
6. **ansible-core** をインストールします。

   ```
   $ sudo dnf install -y ansible-core
   ```

7. オプション: トラブルシューティングに役立つ追加のユーティリティー (**wget**、**git-core**、**rsync**、**vim** など) をインストールできます。

   ```
   $ sudo dnf install -y wget git-core rsync vim
   ```

8. オプション: インストールプログラムが自動的に Ansible Automation Platform サブスクリプ

8. オプション: インストールプログラムが自動的に Ansible Automation Platform サブスクリプションマニフェストライセンスを選択して適用するには、マニフェストファイルの取得 の手順に従います。

関連情報

- Red Hat Ansible Automation Platform サブスクリプションの割り当て
- アンバウンド DNS サーバーのセットアップ
- BIND DNS サーバーのセットアップおよび設定
- Ansible Core ドキュメント

2.4. コンテナーインストールに向けた管理対象ノードの準備

管理対象ノード (ホストとも呼ばれます) は、Ansible Automation Platform によって管理されるように設定されているデバイスです。

コンテナー化された Ansible Automation Platform を一貫してセキュアにセットアップするために、各ホストに専用のユーザーを作成してください。Ansible Automation Platform は、このユーザーとして接続し、ホスト上でタスクを実行します。

設定が完了したら、インベントリーファイルに **ansible_user=<username>** を追加して、各ホストの専用ユーザーを定義できます (例: **aap.example.org ansible_user=aap**)。

各ホストに対して次の手順を実行します。

手順

1. root ユーザーとしてホストにログインします。
2. 新しいユーザーを作成します。**<username>** を **aap** などの希望するユーザー名に置き換えます。

```
$ adduser <username>
```

3. 新しいユーザーのパスワードを設定します。**<username>** を作成したユーザー名に置き換えます。

```
$ passwd <username>
```

4. sudo コマンドを実行するようにユーザーを設定します。
   a. これを行うには、sudoers ファイルを開きます。

```
$ vi /etc/sudoers
```

   b. ファイルに次の行を追加します (**<username>** を作成したユーザー名に置き換えます)。

```
<username> ALL=(ALL) NOPASSWD: ALL
```

   c. ファイルを保存し、終了します。

2.5. ANSIBLE AUTOMATION PLATFORM のダウンロード

Red Hat Enterprise Linux 環境のインターネット接続に基づいて、必要なインストールプログラムを選択し、Red Hat Enterprise Linux ホストにインストールプログラムをダウンロードします。

前提条件

- 非 root ユーザーとして Red Hat Enterprise Linux ホストにログインしている。

手順

1. Ansible Automation Platform ダウンロードページ からコンテナー化された Ansible Automation Platform の最新バージョンをダウンロードします。
   a. オンラインインストールの場合: **Ansible Automation Platform 2.5 Containerized Setup**
   b. オフラインまたはバンドルインストールの場合: **Ansible Automation Platform 2.5 Containerized Setup Bundle**
2. インストールプログラムの **.tar.gz** ファイルと、必要に応じてマニフェスト **.zip** ファイルを Red Hat Enterprise Linux ホストにコピーします。
   a. **scp** コマンドを使用してファイルを安全にコピーできます。**scp** の基本構文は次のとおりです。

   ```
   scp [options] <path_to_source_file> <path_to_destination>
   ```

   次の **scp** コマンドを使用して、インストールプログラムの **.tar.gz** ファイルを秘密鍵を使用して AWS EC2 インスタンスにコピーします (プレースホルダーの **<>** 値は実際の情報に置き換えます)。

   ```
   scp -i <path_to_private_key> ansible-automation-platform-containerized-setup-<version_number>.tar.gz ec2-user@<remote_host_ip_or_hostname>:<path_to_destination>
   ```

3. インストールプログラムをファイルシステム上のどこに配置するか決定します。これはインストールディレクトリーと呼ばれます。
   a. インストール関連のファイルがこの場所に作成されます。初期インストールの場合、少なくとも 15 GB が必要です。
4. インストールプログラムの **.tar.gz** ファイルをインストールディレクトリーに展開し、展開し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を展開するには、次のコマンドを実行します。

   ```
   $ tar xfvz ansible-automation-platform-containerized-setup-<version_number>.tar.gz
   ```

   b. オフラインまたはバンドルインストーラーを展開するには次のコマンドを実行します。

   ```
   $ tar xfvz ansible-automation-platform-containerized-setup-bundle-<version_number>-<arch_name>.tar.gz
   ```

関連情報

- scp(1) - Linux man ページ

2.6. インベントリーファイルの設定

Ansible Automation Platform のインストールは、インベントリーファイルで制御できます。インベントリーファイルは、インストールをカスタマイズするために必要な情報を定義するものです。たとえば、ホストの詳細、証明書の詳細、さまざまなコンポーネント固有の設定などを定義します。

このドキュメントには、すぐに作業を開始できるように、コピーして変更できるサンプルインベントリーファイルが用意されています。

さらに、グロートポロジーとエンタープライズトポロジーのインベントリーファイルを次の場所で入手できます。

- ダウンロードしたインストールプログラムパッケージ内:
  - **inventory** という名前のデフォルトのインベントリーファイルは、エンタープライズトポロジーパターン用です。
  - グロートポロジー (オールインワン) パターンをデプロイするには、代わりに **inventory-growth** ファイルを使用してください。
- **テスト済みのデプロイメントモデル** の コンテナートポロジー。

インベントリーファイルの例を使用するには、プレースホルダー **< >** を実際の変数に置き換え、ホスト名を更新します。

任意および必須の変数の詳細は、インストールディレクトリーの **README.md** ファイルまたは インベントリーファイル変数 を参照してください。

2.6.1. コンテナー化されたグロートポロジー (オールインワン) のオンラインインストール用のインベントリーファイル

コンテナー化されたグロートポロジー (オールインワン) のオンラインインストールを実行するには、このインベントリーファイルの例を使用します。

```
# This is the Ansible Automation Platform installer inventory file intended for the container growth deployment topology.
# This inventory file expects to be run from the host where Ansible Automation Platform will be installed.
# Consult the Ansible Automation Platform product documentation about this topology's tested hardware configuration.
#
https://docs.redhat.com/en/documentation/red_hat_ansible_automation_platform/2.5/html/tested_deployment_models/container-topologies
#
# Consult the docs if you are unsure what to add
# For all optional variables consult the included README.md
# or the Ansible Automation Platform documentation:
#
https://docs.redhat.com/en/documentation/red_hat_ansible_automation_platform/2.5/html/containerized_installation

# This section is for your platform gateway hosts
# -----------------------------------------------------
```

```
[automationgateway]
aap.example.org

# This section is for your automation controller hosts
# ------------------------------------------------------
[automationcontroller]
aap.example.org

# This section is for your automation hub hosts
# ------------------------------------------------------
[automationhub]
aap.example.org

# This section is for your Event-Driven Ansible controller hosts
# ------------------------------------------------------
[automationeda]
aap.example.org

# This section is for the Ansible Automation Platform database
# ------------------------------------------------------
[database]
aap.example.org

[all:vars]
# Ansible
ansible_connection=local

# Common variables
#
https://docs.redhat.com/en/documentation/red_hat_ansible_automation_platform/2.5/html/containerized
_installation/appendix-inventory-files-vars#general-variables
# ------------------------------------------------------
postgresql_admin_username=postgres
postgresql_admin_password=<set your own>

registry_username=<your RHN username>
registry_password=<your RHN password>

redis_mode=standalone

# Platform gateway
#
https://docs.redhat.com/en/documentation/red_hat_ansible_automation_platform/2.5/html/containerized
_installation/appendix-inventory-files-vars#platform-gateway-variables
# ------------------------------------------------------
gateway_admin_password=<set your own>
gateway_pg_host=aap.example.org
gateway_pg_password=<set your own>

# Automation controller
#
https://docs.redhat.com/en/documentation/red_hat_ansible_automation_platform/2.5/html/containerized
_installation/appendix-inventory-files-vars#controller-variables
# ------------------------------------------------------
controller_admin_password=<set your own>
controller_pg_host=aap.example.org
```

```
controller_pg_password=<set your own>
controller_percent_memory_capacity=0.5

# Automation hub
#
https://docs.redhat.com/en/documentation/red_hat_ansible_automation_platform/2.5/html/containerized
_installation/appendix-inventory-files-vars#hub-variables
# ------------------------------------------------------
hub_admin_password=<set your own>
hub_pg_host=aap.example.org
hub_pg_password=<set your own>
hub_seed_collections=false

# Event-Driven Ansible controller
#
https://docs.redhat.com/en/documentation/red_hat_ansible_automation_platform/2.5/html/containerized
_installation/appendix-inventory-files-vars#event-driven-ansible-variables
# ------------------------------------------------------
eda_admin_password=<set your own>
eda_pg_host=aap.example.org
eda_pg_password=<set your own>
```

- **ansible_connection=local** - Ansible Automation Platform をホストする同じノードでインストールプログラムを実行するオールインワンインストールに使用されます。
    - インストールプログラムを別のノードから実行する場合は、**ansible_connection=local** を含めないでください。この場合は、代わりに SSH 接続を使用してください。

関連情報

- コンテナーグロートポロジー

2.6.2. コンテナー化されたエンタープライズトポロジーのオンラインインストール用のインベントリーファイル

コンテナー化されたエンタープライズトポロジーのオンラインインストールを実行するには、このインベントリーファイルの例を使用します。

```
# This is the Ansible Automation Platform enterprise installer inventory file
# Consult the docs if you are unsure what to add
# For all optional variables consult the included README.md
# or the Red Hat documentation:
#
https://docs.redhat.com/en/documentation/red_hat_ansible_automation_platform/2.5/html/containerized
_installation

# This section is for your platform gateway hosts
# ------------------------------------------------------
[automationgateway]
gateway1.example.org
gateway2.example.org

# This section is for your automation controller hosts
# ------------------------------------------------------
[automationcontroller]
```

```
controller1.example.org
controller2.example.org

# This section is for your Ansible Automation Platform execution hosts
# -----------------------------------------------------
[execution_nodes]
hop1.example.org receptor_type='hop'
exec1.example.org
exec2.example.org

# This section is for your automation hub hosts
# -----------------------------------------------------
[automationhub]
hub1.example.org
hub2.example.org

# This section is for your Event-Driven Ansible controller hosts
# -----------------------------------------------------
[automationeda]
eda1.example.org
eda2.example.org

[redis]
gateway1.example.org
gateway2.example.org
hub1.example.org
hub2.example.org
eda1.example.org
eda2.example.org

[all:vars]

# Common variables
#
https://docs.redhat.com/en/documentation/red_hat_ansible_automation_platform/2.5/html/containerized
_installation/appendix-inventory-files-vars#general-variables
# -----------------------------------------------------
postgresql_admin_username=<set your own>
postgresql_admin_password=<set your own>
registry_username=<your RHN username>
registry_password=<your RHN password>

# Platform gateway
#
https://docs.redhat.com/en/documentation/red_hat_ansible_automation_platform/2.5/html/containerized
_installation/appendix-inventory-files-vars#platform-gateway-variables
# -----------------------------------------------------
gateway_admin_password=<set your own>
gateway_pg_host=externaldb.example.org
gateway_pg_database=<set your own>
gateway_pg_username=<set your own>
gateway_pg_password=<set your own>

# Automation controller
#
https://docs.redhat.com/en/documentation/red_hat_ansible_automation_platform/2.5/html/containerized
```

```
_installation/appendix-inventory-files-vars#controller-variables
# ------------------------------------------------------
controller_admin_password=<set your own>
controller_pg_host=externaldb.example.org
controller_pg_database=<set your own>
controller_pg_username=<set your own>
controller_pg_password=<set your own>

# Automation hub
#
https://docs.redhat.com/en/documentation/red_hat_ansible_automation_platform/2.5/html/containerized
_installation/appendix-inventory-files-vars#hub-variables
# ------------------------------------------------------
hub_admin_password=<set your own>
hub_pg_host=externaldb.example.org
hub_pg_database=<set your own>
hub_pg_username=<set your own>
hub_pg_password=<set your own>

# Event-Driven Ansible controller
#
https://docs.redhat.com/en/documentation/red_hat_ansible_automation_platform/2.5/html/containerized
_installation/appendix-inventory-files-vars#event-driven-ansible-variables
# ------------------------------------------------------
eda_admin_password=<set your own>
eda_pg_host=externaldb.example.org
eda_pg_database=<set your own>
eda_pg_username=<set your own>
eda_pg_password=<set your own>
```

関連情報

- コンテナーエンタープライズトポロジー
- キャッシュおよびキューイングシステム

2.6.3. registry_username と registry_password の設定

オンラインの非バンドルインストールに **registry_username** 変数および **registry_password** 変数を使用する場合は、新しい Registry Service Account を作成する必要があります。

Registry Service Account は、デプロイメントシステムなど、認証情報が共有される環境で使用できる名前付きトークンです。

手順

1. https://access.redhat.com/terms-based-registry/accounts に移動します。
2. **Registry Service Account** ページで、**New Service Account** をクリックします。
3. 許可される文字のみを使用して、アカウントの名前を入力します。
4. 必要に応じて、アカウントの説明を入力します。
5. **Create** をクリックします。

6. 検索フィールドで名前を検索し、作成したアカウントをリストで確認します。
7. 作成したアカウントの名前をクリックします。
8. または、トークンの名前がわかっている場合は、URL を入力してページに直接移動することもできます。

```
https://access.redhat.com/terms-based-registry/token/<name-of-your-token>
```

9. **token** ページが開き、生成されたユーザー名 (アカウント名とは異なる) とトークンが表示されます。
    a. トークンが表示されない場合は、**Regenerate Token** をクリックします。これをクリックして、新しいユーザー名とトークンを生成することもできます。
10. ユーザー名 (例: "1234567|testuser") をコピーし、これを使用して変数 **registry_username** を設定します。
11. トークンをコピーし、これを使用して変数 **registry_password** を設定します。

2.7. 高度な設定オプション

外部データベースのセットアップやカスタム TLS 証明書の使用などの高度な設定オプションは、コンテナー化された Ansible Automation Platform の複雑なデプロイメントに使用できます。

これらの高度な設定オプションを使用しない場合は、コンテナー化された Ansible Automation Platform のインストール に進み、インストールを続行してください。

2.7.1. Event-Driven Ansible Controller への安全なプラグイン変数の追加

redhat.insights_eda または同様のプラグインを使用して、Event-Driven Ansible Controller でルールブックアクティベーションを実行する場合は、Ansible Automation Platform のディレクトリーに安全なプラグイン変数を追加する必要があります。これにより、Event-Driven Ansible Controller とソースプラグイン間の接続が確保され、ポートマッピングが正しく表示されます。

手順

1. 安全なプラグイン変数のディレクトリーを作成します。

```
mkdir -p ./group_vars/automationeda
```

2. そのディレクトリー内に新しい設定用のファイルを作成します (例: **touch ./group_vars/automationeda/custom.yml**)。
3. 有効にするプラグインのリストを含む変数 **eda_safe_plugins** を追加します。以下に例を示します。

```
eda_safe_plugins: ['ansible.eda.webhook', 'ansible.eda.alertmanager']
```

2.7.2. 実行ノードの追加

コンテナー化された Ansible Automation Platform は、リモート実行ノードをデプロイできます。

リモート実行ノードは、インベントリーファイルの **[execution_nodes]** グループで定義できます。

```
[execution_nodes]
<fqdn_of_your_execution_host>
```

デフォルトでは、実行ノードには次の設定が使用されます。この設定は必要に応じて変更できます。

```
receptor_port=27199
receptor_protocol=tcp
receptor_type=execution
```

- **receptor_port** - Receptor が他の Receptor ノードからの着信接続をリッスンするポート番号。
- **receptor_type** - ノードのロール。有効なオプションは、**execution** または **hop** です。
- **receptor_protocol** - 通信に使用するプロトコル。有効なオプションは、**tcp** または **udp** です。

デフォルトでは、**[execution_nodes]** グループ内のすべてのノードがコントローラーノードのピアとして追加されます。ピア設定を変更するには、**receptor_peers** 変数を使用します。

注記

receptor_peers の値は、ホスト名のコンマ区切りリストである必要があります。インベントリーグループ名は使用しないでください。

設定例:

```
[execution_nodes]
# Execution nodes
exec1.example.com
exec2.example.com
# Hop node that peers with the two execution nodes above
hop1.example.com receptor_type=hop
receptor_peers='["exec1.example.com","exec2.example.com"]'
```

2.7.3. Automation Hub のストレージの設定

Amazon S3、Azure Blob Storage、ネットワークファイルシステム (NFS) ストレージなどの Automation Hub のストレージバックエンドを設定します。

2.7.3.1. Automation Hub 用の Amazon S3 ストレージの設定

Amazon S3 ストレージは、コンテナー化されたインストールでサポートされるオブジェクトストレージの一種です。AWS S3 ストレージバックエンドを使用する場合は、**hub_storage_backend** を **s3** に設定します。インストールプログラムを実行する前に、AWS S3 バケットが必要です。

手順

1. インストールを続行する前に、AWS S3 バケットが存在することを確認してください。
2. インベントリーファイルの **[all:vars]** グループの下に次の変数を追加し、S3 ストレージを設定します。

- **hub_s3_access_key**
- **hub_s3_secret_key**
- **hub_s3_bucket_name**
- **hub_s3_extra_settings**
  Ansible **hub_s3_extra_settings** ディクショナリーを通じて追加のパラメーターを渡すことができます。以下に例を示します。

```
hub_s3_extra_settings:
  AWS_S3_MAX_MEMORY_SIZE: 4096
  AWS_S3_REGION_NAME: eu-central-1
  AWS_S3_USE_SSL: True
```

関連情報

- django-storages ドキュメント - Amazon S3

2.7.3.2. Automation Hub 用の Azure Blob Storage の設定

Azure Blob ストレージは、コンテナー化されたインストールでサポートされるオブジェクトストレージの一種です。Azure Blob ストレージバックエンドを使用する場合は、**hub_storage_backend** を **azure** に設定します。インストールプログラムを実行する前に、Azure コンテナーが存在している必要があります。

手順

1. インストールを続行する前に、Azure コンテナーが存在することを確認してください。
2. インベントリーファイルの **[all:vars]** グループの下に次の変数を追加し、Azure Blob ストレージを設定します。
   - **hub_azure_account_key**
   - **hub_azure_account_name**
   - **hub_azure_container**
   - **hub_azure_extra_settings**
     Ansible **hub_azure_extra_settings** ディクショナリーを通じて追加のパラメーターを渡すことができます。以下に例を示します。

```
hub_azure_extra_settings:
  AZURE_LOCATION: foo
  AZURE_SSL: True
  AZURE_URL_EXPIRATION_SECS: 60
```

関連情報

- django-storages ドキュメント - Azure Storage

2.7.3.3. Automation Hub のネットワークファイルシステム (NFS) ストレージの設定

NFS は、コンテナー化されたインストールでサポートされる共有ストレージの一種です。**file** ストレージバックエンドを使用する Automation Hub のインスタンスを複数インストールする場合は、共有ストレージが必要です。Automation Hub のインスタンスを 1 つだけインストールする場合、共有ストレージは任意です。

手順

1. Automation Hub の共有ストレージを設定するには、インベントリーファイルで **hub_shared_data_path** 変数を設定します。

```
hub_shared_data_path=<path_to_nfs_share>
```

   値は **host:dir** の形式と同じである必要があります (例: **nfs-server.example.com:/exports/hub**)。

2. (オプション) NFS 共有のマウントオプションを変更するには、**hub_shared_data_mount_opts** 変数を使用します。デフォルト値は **rw,sync,hard** です。

2.7.4. HAProxy ロードバランサーの設定

カスタム CA 証明書を使用してプラットフォームゲートウェイの前に HAProxy ロードバランサーを設定するには、**[all:vars]** グループの下に次のインベントリーファイル変数を設定します。

```
custom_ca_cert=<path_to_cert_crt>
gateway_main_url=<https://load_balancer_url>
```

注記

HAProxy SSL パススルーモードは、プラットフォームゲートウェイではサポートされていません。

2.7.5. 自動化コンテンツコレクションとコンテナー署名の有効化

自動化コンテンツ署名はデフォルトで無効になっています。これを有効にするには、インベントリーファイルに次のインストール変数が必要です。

```
# Collection signing
hub_collection_signing=true
hub_collection_signing_key=<full_path_to_collection_gpg_key>

# Container signing
hub_container_signing=true
hub_container_signing_key=<full_path_to_container_gpg_key>
```

鍵がパスフレーズで保護されている場合は、次の変数が必要です。

```
# Collection signing
hub_collection_signing_pass=<gpg_key_passphrase>

# Container signing
hub_container_signing_pass=<gpg_key_passphrase>
```

hub_collection_signing_key および **hub_container_signing_key** 変数では、インストールを実行する前に鍵を設定する必要があります。

現在、自動化コンテンツ署名は GnuPG (GPG) ベースの署名鍵のみをサポートしています。GPG の詳細は、GnuPG の man ページ を参照してください。

注記

使用するアルゴリズムと暗号はお客様の責任となります。

手順

1. RHEL9 サーバーで次のコマンドを実行して、コレクション署名用の新しい鍵ペアを作成します。

```
gpg --gen-key
```

2. "Real name" と "Email address" に、ユーザー自身の情報を入力します。
出力例:

```
gpg --gen-key
gpg (GnuPG) 2.3.3; Copyright (C) 2021 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Note: Use "gpg --full-generate-key" for a full featured key generation dialog.

GnuPG needs to construct a user ID to identify your key.

Real name: Joe Bloggs
Email address: jbloggs@example.com
You selected this USER-ID:
    "Joe Bloggs <jbloggs@example.com>"

Change (N)ame, (E)mail, or (O)kay/(Q)uit? O
```

これが失敗した場合、使用している環境に GPG に必要な前提条件パッケージがインストールされていません。続行するには必要なパッケージをインストールしてください。

3. ダイアログボックスが表示され、パスフレーズの入力を求められます。これはオプションですが、推奨されます。

4. 鍵が生成され、次のような出力が生成されます。

```
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
gpg: key 022E4FBFB650F1C4 marked as ultimately trusted
gpg: revocation certificate stored as '/home/aapuser/.gnupg/openpgp-
revocs.d/F001B037976969DD3E17A829022E4FBFB650F1C4.rev'
public and secret key created and signed.

pub   rsa3072 2024-10-25 [SC] [expires: 2026-10-25]
```

```
      F001B037976969DD3E17A829022E4FBFB650F1C4
uid                  Joe Bloggs <jbloggs@example.com>
sub   rsa3072 2024-10-25 [E] [expires: 2026-10-25]
```

有効期限は、会社の標準とニーズに基づき慎重に設定してください。

5. 次のコマンドを実行すると、すべての GPG 鍵を表示できます。

```
gpg --list-secret-keys --keyid-format=long
```

6. 公開鍵をエクスポートするには、次のコマンドを実行します。

```
gpg --export -a --output collection-signing-key.pub <email_address_used_to_generate_key>
```

7. 秘密鍵をエクスポートするには、次のコマンドを実行します。

```
gpg -a --export-secret-keys <email_address_used_to_generate_key> > collection-signing-key.priv
```

8. パスフレーズが検出されると、パスフレーズの入力を求められます。

9. 秘密鍵ファイルの内容を表示するには、次のコマンドを実行します。

```
cat collection-signing-key.priv
```

出力例:

```
-----BEGIN PGP PRIVATE KEY BLOCK-----

lQWFBGcbN14BDADTg5BsZGbSGMHypUJMuzmIffzzz4LULrZA8L/I616lzpBHJvEs
sSN6KuKY1TcIwIDCCa/U5Obm46kurpP2Y+vNA1YSEtMJoSeHeamWMDd99f49ItBp

<snippet>

j920hRy/3wJGRDBMFa4mlQg=
=uYEF
-----END PGP PRIVATE KEY BLOCK-----
```

10. 手順 1 - 9 を繰り返して、コンテナー署名用の鍵ペアを作成します。

11. 次の変数をインベントリーファイルに追加し、インストールを実行して署名サービスを作成します。

```
# Collection signing
hub_collection_signing=true
hub_collection_signing_key=/home/aapuser/aap/ansible-automation-platform-containerized-setup-<version_number>/collection-signing-key.priv
# This variable is required if the key is protected by a passphrase
hub_collection_signing_pass=<password>

# Container signing
hub_container_signing=true
hub_container_signing_key=/home/aapuser/aap/ansible-automation-platform-containerized-
```

```
setup-<version_number>/container-signing-key.priv
# This variable is required if the key is protected by a passphrase
hub_container_signing_pass=<password>
```

関連情報

- 署名済みコンテナーの操作

2.7.6. お客様が用意する (外部) データベースの設定

外部データベースを設定する場合、次の 2 つの状況が考えられます。

1. PostgreSQL 管理者認証情報がある外部データベース
2. PostgreSQL 管理者認証情報がない外部データベース

重要

- Ansible Automation Platform で外部データベースを使用する場合は、そのデータベースを作成および保守する必要があります。Ansible Automation Platform をアンインストールする際は、外部データベースを必ずクリアしてください。
- Red Hat Ansible Automation Platform では、お客様が用意する (外部) データベースが ICU をサポートしている必要があります。
- 外部データベースの設定時には、外部データベースの対象範囲を確認する必要があります。詳細は、Red Hat Ansible Automation Platform データベースの対象範囲 を参照してください。

2.7.6.1. PostgreSQL 管理者認証情報がある外部データベースの設定

PostgreSQL 管理者認証情報がある場合は、それをインベントリーファイルに入力すると、インストールプログラムによって各コンポーネントの PostgreSQL ユーザーとデータベースが自動的に作成されます。PostgreSQL 管理者アカウントには **SUPERUSER** 権限が必要です。

手順

- PostgreSQL 管理者認証情報を設定するには、インベントリーファイルの **[all:vars]** グループに次の変数を追加します。

```
postgresql_admin_username=<set your own>
postgresql_admin_password=<set your own>
```

2.7.6.2. PostgreSQL 管理者認証情報がない外部データベースの設定

PostgreSQL 管理者認証情報がない場合は、インストールプログラムを実行する前に、各コンポーネント (プラットフォームゲートウェイ、Automation Controller、Automation Hub、および Event-Driven Ansible) ごとに PostgreSQL ユーザーとデータベースを作成する必要があります。

手順

1. **SUPERUSER** 権限を持つユーザーを使用して、PostgreSQL 準拠のデータベースサーバーに接続します。

```
# psql -h <hostname> -U <username> -p <port_number>
```

以下に例を示します。

```
# psql -h db.example.com -U superuser -p 5432
```

2. パスワード付きのユーザーを作成し、**CREATEDB** ロールがユーザーに割り当てられていることを確認します。詳細は、Database Roles を参照してください。

```
CREATE USER <username> WITH PASSWORD <password> CREATEDB;
```

3. データベースを作成し、作成したユーザーを所有者として追加します。

```
CREATE DATABASE <database_name> OWNER <username>;
```

4. 各コンポーネントの PostgreSQL ユーザーとデータベースを作成したら、それらをインベントリーファイルの **[all:vars]** グループに指定できます。

```
# Platform gateway
gateway_pg_host=aap.example.org
gateway_pg_database=<set your own>
gateway_pg_username=<set your own>
gateway_pg_password=<set your own>

# Automation controller
controller_pg_host=aap.example.org
controller_pg_database=<set your own>
controller_pg_username=<set your own>
controller_pg_password=<set your own>

# Automation hub
hub_pg_host=aap.example.org
hub_pg_database=<set your own>
hub_pg_username=<set your own>
hub_pg_password=<set your own>

# Event-Driven Ansible
eda_pg_host=aap.example.org
eda_pg_database=<set your own>
eda_pg_username=<set your own>
eda_pg_password=<set your own>
```

2.7.6.3. Automation Hub PostgreSQL データベースの hstore 拡張機能の有効化

データベース移行スクリプトは、**hstore** フィールドを使用して情報を保存します。そのため、Automation Hub PostgreSQL データベースで **hstore** 拡張機能を有効にする必要があります。

Ansible Automation Platform インストーラーとマネージド PostgreSQL サーバーを使用する場合、このプロセスは自動的に行われます。

PostgreSQL データベースが外部にある場合は、インストール前に、Automation Hub PostgreSQL データベースで **hstore** 拡張機能を手動で有効にする必要があります。

インストール前に **hstore** 拡張機能が有効になっていないと、データベースの移行中にエラーが発生します。

手順

1. 拡張機能が PostgreSQL サーバー (Automation Hub データベース) で利用できるかどうかを確認します。

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

2. **<automation hub database>** のデフォルト値は **automationhub** です。
hstore が利用できる場合の出力例:

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

hstore が利用できない場合の出力例:

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

3. RHEL ベースのサーバーでは、**hstore** 拡張機能は **postgresql-contrib** RPM パッケージに含まれていますが、PostgreSQL サーバー RPM パッケージのインストール時に自動的にインストールされません。
RPM パッケージをインストールするには、次のコマンドを使用します。

```
dnf install postgresql-contrib
```

4. 次のコマンドを使用して、**hstore** PostgreSQL 拡張機能を Automation Hub データベースにロードします。

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

次の出力では、使用されている **hstore** 拡張機能が **installed_version** フィールドに表示されています。これは **hstore** が有効になっていることを示しています。

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

2.7.6.4. オプション: 外部データベースの相互 TLS (mTLS) 認証の設定

mTLS 認証はデフォルトで無効になっています。各コンポーネントのデータベースに mTLS 認証を設定するには、インベントリーファイルの **[all:vars]** グループの下に次の変数を追加し、各コンポーネントに異なる TLS 証明書および鍵があることを確認します。

手順

- インベントリーファイルの **[all:vars]** グループの下に次の変数を追加します。

```
# Platform gateway
gateway_pg_cert_auth=true
gateway_pg_tls_cert=/path/to/gateway.cert
gateway_pg_tls_key=/path/to/gateway.key
gateway_pg_sslmode=verify-full

# Automation controller
controller_pg_cert_auth=true
controller_pg_tls_cert=/path/to/awx.cert
controller_pg_tls_key=/path/to/awx.key
controller_pg_sslmode=verify-full

# Automation hub
hub_pg_cert_auth=true
hub_pg_tls_cert=/path/to/pulp.cert
hub_pg_tls_key=/path/to/pulp.key
hub_pg_sslmode=verify-full

# Event-Driven Ansible
eda_pg_cert_auth=true
eda_pg_tls_cert=/path/to/eda.cert
eda_pg_tls_key=/path/to/eda.key
eda_pg_sslmode=verify-full
```

2.7.7. カスタム TLS 証明書の使用

Red Hat Ansible Automation Platform は、X.509 証明書と鍵のペアを使用して、Ansible Automation Platform コンポーネント間の内部トラフィックと、パブリック UI および API 接続の外部トラフィックの両方を保護します。

Ansible Automation Platform デプロイメントの TLS 証明書を管理するには、主に次の 2 つの方法があります。

1. Ansible Automation Platform によって生成される証明書 (デフォルト)
2. ユーザーによって提供される証明書

2.7.7.1. Ansible Automation Platform によって生成される証明書

デフォルトでは、インストールプログラムは自己署名認証局 (CA) を作成し、それを使用してすべての Ansible Automation Platform サービス用の自己署名 TLS 証明書を生成します。自己署名 CA の証明書と鍵は、1 つのノードの **~/aap/tls/** ディレクトリーに生成され、他の全ノードの同じ場所にコピーされます。この CA は最初の作成日から 10 年間有効です。

自己署名証明書は、パブリック信頼チェーンの一部ではありません。インストールプログラムは、**~/aap/tls/extracted/** の下に自己署名 CA 証明書を含む証明書トラストストアを作成し、そのディレクトリーを **/etc/pki/ca-trust/extracted/** の下の各 Ansible Automation Platform サービスコンテナーにバインドマウントします。これにより、各 Ansible Automation Platform コンポーネントが、他の Ansible Automation Platform サービスの自己署名証明書を検証できるようになります。必要に応じて、CA 証明書を他のシステムまたはブラウザーのトラストストアに追加することもできます。

2.7.7.2. ユーザーによって提供される証明書

独自の TLS 証明書および鍵を使用して、インストール中に生成された自己署名証明書の一部またはすべてを置き換える場合は、インベントリーファイルで特定の変数を設定できます。独自の証明書と鍵は、インストールプロセス中に使用できるように、パブリック CA または組織の CA によって事前に生成する必要があります。

2.7.7.2.1. カスタム CA を使用してすべての TLS 証明書を生成する

Ansible Automation Platform ですべての証明書を生成するが、デフォルトの自己署名証明書ではなくカスタム CA で署名する必要がある場合は、この方法を使用します。

手順

- カスタム認証局 (CA) を使用してすべての Ansible Automation Platform サービスの TLS 証明書を生成するには、インベントリーファイルで次の変数を設定します。

```
ca_tls_cert=<path_to_ca_tls_certificate>
ca_tls_key=<path_to_ca_tls_key>
```

2.7.7.2.2. 各サービスにカスタム TLS 証明書を提供する

この方法は、組織が Ansible Automation Platform の外部で TLS 証明書を管理しており、手動でのプロビジョニングが必要な場合に使用します。

手順

- 各サービス (Automation Controller、Automation Hub、Event-Driven Ansible など) に TLS 証明書を手動で提供するには、インベントリーファイルで次の変数を設定します。

```
# Platform gateway
gateway_tls_cert=<path_to_tls_certificate>
gateway_tls_key=<path_to_tls_key>
gateway_pg_tls_cert=<path_to_tls_certificate>
gateway_pg_tls_key=<path_to_tls_key>
gateway_redis_tls_cert=<path_to_tls_certificate>
gateway_redis_tls_key=<path_to_tls_key>

# Automation controller
controller_tls_cert=<path_to_tls_certificate>
controller_tls_key=<path_to_tls_key>
controller_pg_tls_cert=<path_to_tls_certificate>
controller_pg_tls_key=<path_to_tls_key>

# Automation hub
hub_tls_cert=<path_to_tls_certificate>
hub_tls_key=<path_to_tls_key>
hub_pg_tls_cert=<path_to_tls_certificate>
hub_pg_tls_key=<path_to_tls_key>

# Event-Driven Ansible
eda_tls_cert=<path_to_tls_certificate>
eda_tls_key=<path_to_tls_key>
eda_pg_tls_cert=<path_to_tls_certificate>
eda_pg_tls_key=<path_to_tls_key>
eda_redis_tls_cert=<path_to_tls_certificate>
eda_redis_tls_key=<path_to_tls_key>
```

```
# PostgreSQL
postgresql_tls_cert=<path_to_tls_certificate>
postgresql_tls_key=<path_to_tls_key>

# Receptor
receptor_tls_cert=<path_to_tls_certificate>
receptor_tls_key=<path_to_tls_key>

# Redis
redis_tls_cert=<path_to_tls_certificate>
redis_tls_key=<path_to_tls_key>
```

2.7.7.2.3. サービスごとに証明書を提供する場合の考慮事項

個々のサービスごとにカスタム TLS 証明書を提供する場合は、次の点を考慮してください。

- ホストごとに固有の証明書を提供できます。これには、前述のインベントリーファイルの例に示されているように、インベントリーファイルで特定の **_tls_cert** および **_tls_key** 変数を定義する必要があります。
- 多数のノードにまたがってデプロイされるサービスの場合 (たとえば、エンタープライズトポロジーを採用する場合)、そのサービスに提供する証明書のサブジェクト代替名 (SAN) フィールドに、関連するすべてのノードの FQDN が含まれている必要があります。
- 外部向けサービス (Automation Controller やプラットフォームゲートウェイなど) が、SSL/TLS オフロードを実行するロードバランサーの背後にデプロイされている場合、サービスの証明書の SAN フィールドに、個々のサービスノードの FQDN に加えて、ロードバランサーの FQDN が含まれている必要があります。

関連情報

- ネットワークのセキュリティー保護

2.7.7.2.4. カスタム CA 証明書の提供

TLS 証明書を手動で提供する場合は、その証明書がカスタム CA によって署名されている可能性があります。環境内で適切な認証とセキュアな通信を確保するために、カスタム CA 証明書を提供してください。複数のカスタム CA 証明書がある場合は、それらを 1 つのファイルに結合する必要があります。

手順

- 手動で提供した TLS 証明書のいずれかがカスタム CA によって署名されている場合は、インベントリーファイルで次の変数を使用して CA 証明書を指定する必要があります。

  ```
  custom_ca_cert=<path_to_custom_ca_certificate>
  ```

  複数の CA 証明書がある場合は、それらを 1 つのファイルに結合し、結合した証明書を **custom_ca_cert** 変数で参照します。

2.7.7.3. Receptor 証明書の考慮事項

Receptor ノードにカスタム証明書を使用する場合、証明書のサブジェクト代替名 (SAN) の **otherName** フィールドに、値 **1.3.6.1.4.1.2312.19.1** が指定されている必要があります。詳細は、Above the mesh TLS を参照してください。

Receptor はワイルドカード証明書の使用をサポートしていません。さらに、TLS ホスト名検証を正しく実行するために、各 Receptor 証明書の SAN にホスト FQDN が指定されている必要があります。

2.7.7.4. Redis 証明書に関する考慮事項

Redis 関連サービスにカスタムの TLS 証明書を使用する際に、Extended Key Usage (EKU) を指定する場合は、相互 TLS (mTLS) 通信について次の点を考慮してください。

- Redis サーバー証明書 (**redis_tls_cert**) には、**serverAuth** (Web サーバー認証) および **clientAuth** (クライアント認証) EKU が含まれている必要があります。
- Redis クライアント証明書 (**gateway_redis_tls_cert**、**eda_redis_tls_cert**) には、**clientAuth** (クライアント認証) EKU が含まれている必要があります。

2.7.8. カスタム Receptor 署名鍵の使用

receptor_disable_signing=true が設定されていない Receptor 署名がデフォルトで有効になり、インストールプログラムによって RSA 鍵ペア (公開鍵と秘密鍵) が生成されます。ただし、以下の変数を使用して、カスタムの RSA 公開鍵と秘密鍵を設定できます。

```
receptor_signing_private_key=<full_path_to_private_key>
receptor_signing_public_key=<full_path_to_public_key>
```

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

Red Hat Enterprise Linux ホストを準備し、Ansible Automation Platform をダウンロードして、インベントリーファイルを設定したら、**install** Playbook を実行して、コンテナー化された Ansible Automation Platform をインストールします。

前提条件

以下を実行した。

- Red Hat Enterprise Linux ホストの準備
- 管理対象ノードの準備
- Ansible Automation Platform のダウンロード
- インベントリーファイルの設定
- 非 root ユーザーとして Red Hat Enterprise Linux ホストにログインしている

手順

1. Red Hat Enterprise Linux ホストのインストールディレクトリーに移動します。
2. **install** Playbook を実行します。

```
ansible-playbook -i <inventory_file_name> ansible.containerized_installer.install
```

以下に例を示します。

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

必要に応じて、インストールコマンドに追加のパラメーターを追加できます。

```
ansible-playbook -i <inventory_file_name> -e @<vault_file_name> --ask-vault-pass -K -v
ansible.containerized_installer.install
```

以下に例を示します。

```
ansible-playbook -i inventory -e @vault.yml --ask-vault-pass -K -v
ansible.containerized_installer.install
```

- **-i <inventory_file_name>** - インストールに使用するインベントリーファイル。
- **-e @<vault_file_name> --ask-vault-pass** - (オプション) 機密性の高い変数を保存するために vault を使用する場合は、これをインストールコマンドに追加します。
- **-K** - (オプション) 権限の昇格にパスワードの入力が必要な場合は、これをインストールコマンドに追加します。追加すると、BECOME パスワードの入力を求められます。
- **-v** - (オプション) インストールプロセスの詳細を表示するには、最大 4 つの v (**-vvvv**) を使用して詳細度を上げることができます。ただし、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 サポートから要求された場合にのみ使用してください。

3. コンテナー化された Ansible Automation Platform のインストールが開始します。

検証

- インストールが完了したら、次の URL で、デフォルトで利用できるプラットフォーム UI にアクセスできることを確認します。

```
https://<gateway_node>:443
```

- **gateway_admin_username** と **gateway_admin_password** 用に作成した認証情報を使用して、管理者ユーザーとしてログインします。
- Ansible Automation Platform で使用されるデフォルトのポートとプロトコルは、80 (HTTP) と 443 (HTTPS) です。ポートは次の変数を使用してカスタマイズできます。

```
envoy_http_port=80
envoy_https_port=443
```

- HTTPS を無効にする場合は、**envoy_disable_https** を **true** に設定します。

```
envoy_disable_https: true
```

関連情報

- 権限昇格の理解: become
- インストールインベントリー内の機密性の高い変数
- Ansible Automation Platform のスタートガイド
- コンテナー化された Ansible Automation Platform のトラブルシューティング

2.9. コンテナー化された ANSIBLE AUTOMATION PLATFORM の更新

Ansible Automation Platform のコンテナーベースのインストールを 2.5 から 2.5.x にパッチ更新します。

2.4 コンテナー化された Ansible Automation Platform テクノロジープレビューから 2.5 コンテナー化された Ansible Automation Platform へのアップグレードはサポートされていません。

前提条件

- 関連するパッチリリースのリリースノートを確認した。
- Ansible Automation Platform デプロイメントのバックアップを作成した。

手順

1. 専用の非 root ユーザーとして Red Hat Enterprise Linux ホストにログインします。
2. Ansible Automation Platform のダウンロード の手順に従い、コンテナー化された Ansible Automation Platform の最新バージョンをダウンロードします。
3. ダウンロードしたインストールプログラムを Red Hat Enterprise Linux ホストにコピーします。
4. **inventory** ファイルを必要な設定に一致するように編集します。既存の Ansible Automation Platform デプロイメントと同じパラメーターを維持することも、環境の変更に合わせてパラメーターを変更することもできます。
5. **install** Playbook を実行します。

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

   - 権限昇格にパスワードの入力が必要な場合は、コマンドに **-K** を追加します。その場合は、**BECOME** パスワードの入力を求められます。
   - インストールプロセスの詳細を表示するには、最大 4 つの v (**-vvvv**) を使用して詳細度を上げます。ただし、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ことに注意してください。そのため、必要な場合、または Red Hat サポートから要求された場合にのみ使用することを推奨します。
6. 更新が開始されます。

関連情報

- Ansible Automation Platform のリリースノート
- コンテナーベースの Ansible Automation Platform のバックアップ

2.10. コンテナー化された ANSIBLE AUTOMATION PLATFORM のバックアップ

Ansible Automation Platform のコンテナーベースのインストール環境をバックアップします。

注記

Ansible Automation Platform をバックアップする場合は、現在インストールされている Ansible Automation Platform のバージョンと同じバージョンのインストールプログラムを使用してください。

バックアップ機能は、現在お使いの Ansible Automation Platform バージョンでサポートされている PostgreSQL バージョンでのみ機能します。詳細は、システム要件 を参照してください。

前提条件

- 専用の非 root ユーザーとして Red Hat Enterprise Linux ホストにログインしている。

手順

1. Red Hat Enterprise Linux ホスト上の Red Hat Ansible Automation Platform インストールディレクトリーに移動します。
2. バックアップ操作を実行しているホストにバックアップアーティファクトを送信する前に、アーティファクトの圧縮を制御するには、インベントリーファイルで次の変数を使用できます。
   a. ファイルシステム関連のバックアップファイルの圧縮を制御するには、以下を使用します。

```
# For global control of compression for filesystem related backup files
use_archive_compression=true

# For component-level control of compression for filesystem related backup files
#controller_use_archive_compression=true
#eda_use_archive_compression=true
#gateway_use_archive_compression=true
#hub_use_archive_compression=true
#pcp_use_archive_compression=true
#postgresql_use_archive_compression=true
#receptor_use_archive_compression=true
#redis_use_archive_compression=true
```

   b. データベース関連のバックアップファイルの圧縮を制御するには、以下を使用します。

```
# For global control of compression for database related backup files
use_db_compression=true

# For component-level control of compression for database related backup files
#controller_use_db_compression=true
#eda_use_db_compression=true
#hub_use_db_compression=true
#gateway_use_db_compression=true
```

3. **backup** Playbook を実行します。

```
$ ansible-playbook -i <path_to_inventory> ansible.containerized_installer.backup
```

これにより、コンテナーインストーラーによってデプロイされた次のような重要なデータがバックアップされます。

- PostgreSQL データベース
- 設定ファイル
- データファイル

4. デフォルトでは、バックアップディレクトリーは **./backups** に設定されています。このディレクトリーは、**inventory** ファイルの **backup_dir** 変数を使用して変更できます。

次のステップ

バックアップをカスタマイズするには、**inventory** ファイルで次の変数を使用します。

- **backup_dir** 変数を使用して、バックアップ先ディレクトリーをデフォルトの **./backups** から変更します。
- **hub_data_path_exclude** 変数を使用して、スナップショットサブディレクトリーなど、重複データを含むパスを除外します。たとえば、.snapshots サブディレクトリーを除外するには、インベントリーファイルで hub_data_path_exclude=['**/.snapshots/**'] を指定します。
  - または、コマンドラインインターフェイスを使用して **-e** フラグを指定し、実行時にこの変数を渡すこともできます。

```
$ ansible-playbook -i inventory ansible.containerized_installer.backup -e
hub_data_path_exclude="['*/.snapshots/*']"
```

2.11. コンテナー化された ANSIBLE AUTOMATION PLATFORM の復元

Ansible Automation Platform のコンテナーベースのインストール環境を、バックアップから、または別の環境に復元します。

注記

Ansible Automation Platform を復元する場合は、復元時に利用可能な最新のインストールプログラムを使用してください。たとえば、バージョン **2.5-1** から作成したバックアップを復元する場合は、復元時に利用可能な最新の **2.5-x** インストールプログラムを使用してください。

復元機能は、現在お使いの Ansible Automation Platform バージョンでサポートされている PostgreSQL バージョンでのみ機能します。詳細は、システム要件 を参照してください。

前提条件

- 専用の非 root ユーザーとして Red Hat Enterprise Linux ホストにログインしている。

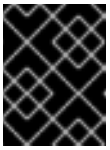
- Ansible Automation Platform デプロイメントのバックアップがある。詳細は、コンテナー化された Ansible Automation Platform のバックアップ を参照してください。
- 同じホスト名を持つ別の環境に復元する場合は、元の (ソース) 環境と同じトポロジーを持つターゲット環境で新規インストールを実行した。
- ターゲット環境の管理者の認証情報がソース環境の管理者の認証情報と一致していることを確認した。

手順

1. Red Hat Enterprise Linux ホストのインストールディレクトリーに移動します。
2. 適切な復元手順を実行します。
   - 同じホスト名を持つ同じ環境に復元する場合は、**restore** Playbook を実行します。

     ```
     $ ansible-playbook -i <path_to_inventory> ansible.containerized_installer.restore
     ```

     これにより、コンテナーインストーラーによってデプロイされた次のような重要なデータが復元されます。
     - PostgreSQL データベース
     - 設定ファイル
     - データファイル
       デフォルトでは、バックアップディレクトリーは **./backups** に設定されています。このディレクトリーは、**inventory** ファイルの **backup_dir** 変数を使用して変更できます。
   - 異なるホスト名を持つ別の環境に復元する場合は、**restore** Playbook を実行する前に次の追加手順を実行します。

     重要

     異なるホスト名を持つ別の環境に復元することは推奨されません。あくまで回避策として意図されたものです。

     i. 各コンポーネントについて、PostgreSQL のダンプファイルが含まれている、ソース環境からのバックアップファイルを特定します。
     以下に例を示します。

     ```
     $ cd ansible-automation-platform-containerized-setup-<version_number>/backups
     ```

     ```
     $ tar tvf gateway_env1-gateway-node1.tar.gz | grep db

     -rw-r--r-- ansible/ansible 4850774 2025-06-30 11:05 aap/backups/awx.db
     ```

     ii. バックアップファイルをソース環境からターゲット環境にコピーします。

     iii. 新しいノード名を反映するように、ターゲット環境のバックアップファイルの名前を変更します。
     以下に例を示します。

```
$ cd ansible-automation-platform-containerized-setup-<version_number>/backups
```

```
$ mv gateway_env1-gateway-node1.tar.gz gateway_env2-gateway-node1.tar.gz
```

iv. エンタープライズトポロジーの場合、**component.db** ファイルを含むコンポーネントのバックアップファイルが、インベントリーファイル内のグループの最初にリストされていることを確認します。
以下に例を示します。

```
$ cd ansible-automation-platform-containerized-setup-<version_number>
```

```
$ ls backups/gateway*

gateway_env2-gateway-node1.tar.gz
gateway_env2-gateway-node2.tar.gz
```

```
$ tar tvf backups/gateway_env2-gateway-node1.tar.gz | grep db

-rw-r--r-- ansible/ansible 416687 2025-06-30 11:05 aap/backups/gateway.db
```

```
$ tar tvf backups/gateway_env2-gateway-node2.tar.gz | grep db
```

```
$ vi inventory

[automationgateway]
env2-gateway-node1
env2-gateway-node2
```

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

Ansible Automation Platform のコンテナーベースのインストールをアンインストールします。

前提条件

- 専用の非 root ユーザーとして Red Hat Enterprise Linux ホストにログインしている。

手順

1. Ansible Automation Platform を再インストールし、保存されているデータベースを使用する場合は、次のコマンドを実行して既存のシークレットキーを収集する必要があります。

```
$ podman secret inspect --showsecret <secret_key_variable> | jq -r .[].SecretData
```

以下に例を示します。

```
$ podman secret inspect --showsecret controller_secret_key | jq -r .[].SecretData
```

2. **uninstall** 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し、コンテナーインストーラーによって使用される次のようなリソースがすべて削除されます。
  - 設定、データディレクトリーおよびファイル
  - 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** パラメーターを **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** パラメーターを **true** に設定します。

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

関連情報

- インベントリーファイル変数

2.13. コンテナー化された ANSIBLE AUTOMATION PLATFORM の再インストール

コンテナー化されたデプロイメントを、データベースを残してアンインストールした後、再インストールするには、コンテナー化された Ansible Automation Platform のインストール の手順を実行します。その際に、Playbook コマンドに既存のシークレットキーの値を含めます。

```
$ ansible-playbook -i inventory ansible.containerized_installer.install -e controller_secret_key=
<secret_key_value>
```

関連情報

- インベントリーファイル変数

第3章 非接続インストール

アクティブなインターネット接続がない環境に、コンテナー化された Ansible Automation Platform をインストールできます。これを行うには、非接続インストールを実行する前に、RPM ソースの依存関係を取得して設定する必要があります。

3.1. RPM ソースの依存関係の取得と設定

Ansible Automation Platform のコンテナーセットアップバンドルのインストールプログラムには、BaseOS および AppStream リポジトリーからの RPM ソースの依存関係は含まれていません。インストールプログラムは、ホストシステムのパッケージマネージャーを使用してこれらの依存関係を解決します。

非接続環境でこれらの依存関係にアクセスするには、次のいずれかの方法を使用できます。

- Red Hat Satellite を使用して、非接続環境内のリポジトリーを同期します。
- アクティブなインターネット接続がある Red Hat Enterprise Linux ホスト上で、**reposync** コマンドを使用して作成したローカルリポジトリーを使用します。
- マウントした Red Hat Enterprise Linux Binary DVD ISO イメージから作成したローカルリポジトリーを使用します。

3.1.1. reposync を使用したローカルリポジトリーの設定

reposync コマンドを使用すると、BaseOS リポジトリーと AppStream リポジトリーを、アクティブなインターネット接続がある Red Hat Enterprise Linux ホスト上のローカルディレクトリーに同期できます。その後、リポジトリーを非接続環境に転送できます。

前提条件

- アクティブなインターネット接続がある Red Hat Enterprise Linux ホスト。

手順

1. **subscription-manager** を使用して BaseOS および AppStream リポジトリーを割り当てます。

```
$ sudo subscription-manager repos \
    --enable rhel-9-baseos-rhui-rpms \
    --enable rhel-9-appstream-rhui-rpms
```

2. **yum-utils** パッケージをインストールします。

```
$ sudo dnf install yum-utils
```

3. **reposync** コマンドを使用してリポジトリーを同期します。**<path_to_download>** は適切な値に置き換えます。

```
$ sudo reposync -m --download-metadata --gpgcheck \
    -p <path_to_download>
```

以下に例を示します。

```
$ sudo reposync -m --download-metadata --gpgcheck \
    -p rhel-repos
```

- ダウンロード時間を最適にするために、**--download-metadata** オプションを指定し、**--newest-only** オプションを指定せずに reposync を使用します。

4. **reposync** 操作が完了したら、ディレクトリーを圧縮します。

```
$ tar czvf rhel-repos.tar.gz rhel-repos
```

5. 圧縮したアーカイブを非接続環境に移動します。

6. 非接続環境で、リポジトリーファイルを保存するディレクトリーを作成します。

```
$ sudo mkdir /opt/rhel-repos
```

7. アーカイブを **/opt/rhel-repos** ディレクトリーに展開します。次のコマンドは、アーカイブファイルがホームディレクトリーにあることを前提としています。

```
$ sudo tar xzvf ~/rhel-repos.tar.gz -C /opt
```

8. 次の内容を含む Yum リポジトリーファイルを **/etc/yum.repos.d/rhel.repo** に作成します。

```
[RHEL-BaseOS]
name=Red Hat Enterprise Linux BaseOS
baseurl=file:///opt/rhel-repos/rhel-9-baseos-rhui-rpms
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release

[RHEL-AppStream]
name=Red Hat Enterprise Linux AppStream
baseurl=file:///opt/rhel-repos/rhel-9-appstream-rhui-rpms
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
```

9. システムがパッケージを検証できるように、gpg 鍵をインポートします。

```
$ sudo rpm --import /opt/rhel-repos/rhel-9-baseos-rhui-rpms/RPM-GPG-KEY-redhat-release
```

10. リポジトリーの設定を確認します。

```
$ sudo yum repolist
```

3.1.2. マウントした ISO からのローカルリポジトリーの設定

Red Hat Enterprise Linux Binary DVD イメージを使用すると、非接続環境で、必要な RPM ソースの依存関係にアクセスできます。

前提条件

- Red Hat Enterprise Linux のダウンロードページ から Red Hat Enterprise Linux Binary DVD イメージをダウンロードし、非接続環境に移動した。

手順

1. 非接続環境で、ISO ファイルの場所として機能するマウントポイントディレクトリーを作成します。

```
$ sudo mkdir /media/rhel
```

2. ISO イメージをマウントポイントにマウントします。**<version_number>** と **<arch_name>** は、適切な値に置き換えます。

```
$ sudo mount -o loop rhel-<version_number>-<arch_name>-dvd.iso /media/rhel
```

   - 注記: ISO は読み取り専用状態でマウントされます。

3. 次の内容を含む Yum リポジトリーファイルを **/etc/yum.repos.d/rhel.repo** に作成します。

```
[RHEL-BaseOS]
name=Red Hat Enterprise Linux BaseOS
baseurl=file:///media/rhel/BaseOS
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release

[RHEL-AppStream]
name=Red Hat Enterprise Linux AppStream
baseurl=file:///media/rhel/AppStream
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
```

4. システムがパッケージを検証できるように、gpg 鍵をインポートします。

```
$ sudo rpm --import /media/rhel/RPM-GPG-KEY-redhat-release
```

5. リポジトリーの設定を確認します。

```
$ sudo yum repolist
```

3.2. 非接続インストールの実行

コンテナー化された Ansible Automation Platform の非接続インストールを実行するには、次の手順を使用します。

前提条件

以下を実行した。

- Red Hat Enterprise Linux ホストの準備

- RPM ソースの依存関係を取得して設定した。インストールプログラムは、ホストシステムの **dnf** パッケージマネージャーを使用してこれらの依存関係を解決します。
- 管理対象ノードの準備
- Ansible Automation Platform ダウンロードページ から、コンテナー化された Ansible Automation Platform のセットアップバンドルをダウンロードした。

手順

1. 非 root ユーザーとして Red Hat Enterprise Linux ホストにログインします。
2. インベントリーファイルの設定 の手順に従って、インベントリーファイルを更新します。
3. 次の変数が、インベントリーファイルの **[all:vars]** グループの下に含まれていることを確認します。

```
bundle_install=true
# The bundle directory must include /bundle in the path
bundle_dir='{{ lookup("ansible.builtin.env", "PWD") }}/bundle'
```

4. コンテナー化された Ansible Automation Platform のインストール の手順に従って、コンテナー化された Ansible Automation Platform をインストールし、インストールを検証します。

第4章 RED HAT ANSIBLE AUTOMATION PLATFORM での水平スケーリング

Ansible Automation Platform 全体のコンポーネントのマルチノードデプロイメントを設定できます。必要な水平スケーリングの対象が、自動化実行でも、自動化決定でも、自動化メッシュでも、組織のニーズに基づいてデプロイメントを拡張できます。

4.1. EVENT-DRIVEN ANSIBLE CONTROLLER での水平スケーリング

Event-Driven Ansible Controller では、イベント自動化の水平スケーリングを設定できます。この種のマルチノードデプロイメントでは、インストールプロセス中に必要な数のノードを定義できます。また、組織のニーズに応じて、いつでもノードの数を増減できます。

このデプロイメントでは、次のノードタイプが使用されます。

API ノードタイプ

Event-Driven Ansible Controller の HTTP REST API に応答します。

ワーカーノードタイプ

Event-Driven Ansible ワーカーを実行します。このワーカーは、プロジェクトとアクティベーションを管理するだけでなく、アクティベーション自体を実行する Event-Driven Ansible のコンポーネントです。

ハイブリッドノードタイプ

API ノードとワーカーノードを組み合わせたものです。

次の例は、ホストグループ名 **[automationeda]** とノードタイプ変数 **eda_type** を使用して、Red Hat Enterprise Linux 仮想マシン上の Event-Driven Ansible Controller の水平スケーリング用にインベントリーファイルを設定する方法を示しています。

```
[automationeda]

3.88.116.111 routable_hostname=automationeda-api.example.com eda_type=api

# worker node
3.88.116.112 routable_hostname=automationeda-api.example.com eda_type=worker
```

4.1.1. サイジングとスケーリングのガイドライン

API ノードは、ユーザーの要求 (UI または API とのやり取り) を処理します。一方、ワーカーノードは、Event-Driven Ansible が適切に機能するために必要なアクティベーションやその他のバックグラウンドタスクを処理します。必要な API ノードの数は、アプリケーションの必要なユーザー数と相関します。ワーカーノードの数は、実行するアクティベーションの必要な数と相関します。

アクティベーションは可変であり、ワーカーノードによって制御されます。そのため、スケーリング方法としてサポートされているのは、ハイブリッドノードではなく、別々の API ノードとワーカーノードを使用することです。これは、ワーカーノードによりハードウェアリソースを効率的に割り当てるためです。ノードを分けることで、特定のニーズに基づいて各タイプを個別に拡張でき、リソースの使用率とコスト効率が向上します。

ノードのデプロイメントのスケールアップを検討する事例としては、多数のアクティベーションを実行する少数のユーザーグループ向けに Event-Driven Ansible をデプロイする場合が挙げられます。この場合、1 つの API ノードで十分ですが、さらに必要な場合は、最大 3 つのワーカーノードを追加してスケールアップできます。

4.1.2. Event-Driven Ansible Controller の水平スケーリングの設定

スケールアップ (ノードを追加) またはスケールダウン (ノードを削除) するには、インベントリーファイルの内容を更新してノードを追加または削除し、インストールプログラムを再実行する必要があります。

手順

1. インベントリーを更新して、2 つのワーカーノードをさらに追加します。

```
[automationeda]

3.88.116.111 routable_hostname=automationeda-api.example.com eda_type=api

3.88.116.112 routable_hostname=automationeda-api.example.com eda_type=worker

# two more worker nodes
3.88.116.113 routable_hostname=automationeda-api.example.com eda_type=worker

3.88.116.114 routable_hostname=automationeda-api.example.com eda_type=worker
```

2. インストーラーを再実行します。

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

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うには、この情報を使用してください。

A.1. ANSIBLE AUTOMATION PLATFORM のログの収集

sos ユーティリティーを使用すると、設定、診断、およびトラブルシューティングのデータを収集し、それらのファイルを Red Hat テクニカルサポートに提供できます。**sos** レポートは、Red Hat テクニカルサポートエンジニアが Ansible Automation Platform のサービスリクエストの分析を実行する際に、出発点として一般的に使用されています。

適切なパラメーターを使用して **log_gathering** Playbook を実行すると、コンテナー化された Ansible Automation Platform デプロイメント内の各ホストの **sos** レポートを収集できます。

手順

1. Ansible Automation Platform のインストールディレクトリーに移動します。
2. **log_gathering** Playbook を実行します。この Playbook は、インベントリーファイル内の各ホストに接続し、**sos** ツールをインストールして、**sos** レポートを生成します。

```
$ ansible-playbook -i <path_to_inventory_file> ansible.containerized_installer.log_gathering
```

3. オプション: 追加のパラメーターを定義するには、**-e** オプションで指定します。以下に例を示します。

```
$ ansible-playbook -i <path_to_inventory_file> ansible.containerized_installer.log_gathering -e 'target_sos_directory=<path_to_files>' -e 'case_number=0000000' -e 'clean=true' -e 'upload=true' -s
```

   a. **-s** オプションを使用すると、Playbook 内の各タスクをステップ実行し、その実行を確認できます。これは任意ですが、デバッグに役立ちます。

   b. 以下は、**log_gathering** Playbook で使用できるパラメーターのリストです。

表A.1 パラメーターリファレンス

パラメーター名	説明	デフォルト
target_sos_directory	**sos** レポートファイルのデフォルトの場所を変更するために使用します。	現在のサーバーの **/tmp** ディレクトリー。
case_number	サポートケース番号を指定します (ログ収集に関連する場合)。	

パラメーター名	説明	デフォルト
clean	**sos** レポートに存在する可能性のある機密データを難読化します。	**false**
upload	**sos** レポートデータを Red Hat に自動的にアップロードします。	**false**

4. Playbook の出力に記載されている **sos** レポートファイルを収集し、サポートエンジニアと共有するか、**upload=true** 追加パラメーターを使用して **sos** レポートを Red Hat に直接アップロードします。

関連情報

- What is an sos report and how to create one in Red Hat Enterprise Linux?

A.2. 問題の診断

一般的なコンテナーベースのトラブルシューティングを実行する場合は、実行中のサービスのコンテナーログを検査すると、根本的な問題のトラブルシューティングに役立ちます。

実行中のコンテナーの特定

実行中のコンテナー名のリストを取得するには、次のコマンドを実行します。

```
$ podman ps --all --format "{{.Names}}"
```

表A.2 コンテナーの詳細

コンポーネントグループ	コンテナー名	目的
Automation Controller	**automation-controller-rsyslog**	Automation Controller の一元的なロギングを処理します。
Automation Controller	**automation-controller-task**	Playbook の実行やインベントリーの操作など、Automation Controller に関連するタスクを管理および実行します。
Automation Controller	**automation-controller-web**	Automation Controller 用の REST API を提供する Web サーバー。これは、ユーザーとのやり取りのために、プラットフォームゲートウェイを介してアクセスおよびルーティングされます。
Event-Driven Ansible	**automation-eda-api**	Event-Driven Ansible の API を公開し、外部システムがイベント駆動型の自動化をトリガーおよび管理できるようにします。

コンポーネントグループ	コンテナー名	目的
Event-Driven Ansible	**automation-eda-daphne**	WebSocket 接続を処理し、静的ファイルを提供する、Event-Driven Ansible 用の Web サーバー。
Event-Driven Ansible	**automation-eda-web**	Event-Driven Ansible 用の REST API を提供する Web サーバー。これは、ユーザーとのやり取りのために、プラットフォームゲートウェイを介してアクセスおよびルーティングされます。
Event-Driven Ansible	**automation-eda-worker-<number>**	これらのコンテナーは、受信イベントに基づいて自動化ルールと Playbook を実行します。
Event-Driven Ansible	**automation-eda-activation-worker-<number>**	これらのコンテナーは、自動化ルールのアクティブ化を管理し、特定の条件が満たされたときにルールが実行されるようにします。
Event-Driven Ansible	**automation-eda-scheduler**	定期的なタスクとルールアクティベーションのスケジュールと管理を担当します。
プラットフォームゲートウェイ	**automation-gateway-proxy**	リバースプロキシーとして機能し、受信リクエストを適切な Ansible Automation Platform サービスにルーティングします。
プラットフォームゲートウェイ	**automation-gateway**	プラットフォームの認証、認可、および全体的なリクエスト処理を担当します。これらはすべて REST API を通じて公開され、Web サーバーによって提供されます。
Automation Hub	**automation-hub-api**	Automation Hub 用の API を提供し、コレクションコンテンツ、ユーザー管理、およびその他の Automation Hub 機能とのやり取りを可能にします。
Automation Hub	**automation-hub-content**	Automation Hub に保存されている Ansible コンテンツコレクション、ロール、モジュールを管理および提供します。
Automation Hub	**automation-hub-web**	Automation Hub 用の REST API を提供する Web サーバー。これは、ユーザーとのやり取りのために、プラットフォームゲートウェイを介してアクセスおよびルーティングされます。
Automation Hub	**automation-hub-worker-<number>**	これらのコンテナーは、コンテンツの同期、インデックス作成、検証など、Automation Hub のバックグラウンドタスクを処理します。
Performance Co-Pilot	**pcp**	Performance Co-Pilot Monitoring が有効な場合、このコンテナーがシステムパフォーマンスの監視とデータ収集に使用されます。

コンポーネントグループ	コンテナー名	目的
PostgreSQL	**postgresql**	Ansible Automation Platform の PostgreSQL データベースをホストします。
Receptor	**receptor**	Ansible Automation Platform 内でセキュアで信頼性の高い通信を容易に行えるようにします。
Redis	**redis-<suffix>**	キャッシュ保存、リアルタイム分析、高速なデータ取得を担当します。

ログの検査

コンテナー化された Ansible Automation Platform は、Podman のロギングに **journald** を使用します。実行中のコンテナーログを検査するには、**journalctl** コマンドを実行します。

```
$ journalctl CONTAINER_NAME=<container_name>
```

コマンドと出力の例:

```
$ journalctl CONTAINER_NAME=automation-gateway-proxy

Oct 08 01:40:12 aap.example.org automation-gateway-proxy[1919]: [2024-10-08 00:40:12.885][2]
[info][upstream] [external/envoy/source/common/upstream/cds_ap>
Oct 08 01:40:12 aap.example.org automation-gateway-proxy[1919]: [2024-10-08 00:40:12.885][2]
[info][upstream] [external/envoy/source/common/upstream/cds_ap>
Oct 08 01:40:19 aap.example.org automation-gateway-proxy[1919]: [2024-10-08T00:40:16.753Z]
"GET /up HTTP/1.1" 200 - 0 1138 10 0 "192.0.2.1" "python->
```

実行中のコンテナーのログをリアルタイムで表示するには、**podman logs -f** コマンドを実行します。

```
$ podman logs -f <container_name>
```

コンテナー操作の制御

systemctl コマンドを実行することで、コンテナーの操作を制御できます。

```
$ systemctl --user status <container_name>
```

コマンドと出力の例:

```
$ systemctl --user status automation-gateway-proxy
● automation-gateway-proxy.service - Podman automation-gateway-proxy.service
    Loaded: loaded (/home/user/.config/systemd/user/automation-gateway-proxy.service; enabled;
preset: disabled)
```

```
    Active: active (running) since Mon 2024-10-07 12:39:23 BST; 23h ago
      Docs: man:podman-generate-systemd(1)
   Process: 780 ExecStart=/usr/bin/podman start automation-gateway-proxy (code=exited,
status=0/SUCCESS)
  Main PID: 1919 (conmon)
     Tasks: 1 (limit: 48430)
    Memory: 852.0K
       CPU: 2.996s
    CGroup: /user.slice/user-1000.slice/user@1000.service/app.slice/automation-gateway-
proxy.service
            └─1919 /usr/bin/conmon --api-version 1 -c
2dc3c7b2cecd73010bad1e0aaa806015065f92556ed3591c9d2084d7ee209c7a -u
2dc3c7b2cecd73010bad1e0aaa80>
Oct 08 11:44:10 aap.example.org automation-gateway-proxy[1919]: [2024-10-08T10:44:02.926Z]
"GET /api/galaxy/_ui/v1/settings/ HTTP/1.1" 200 - 0 654 58 47 ">
Oct 08 11:44:10 aap.example.org automation-gateway-proxy[1919]: [2024-10-08T10:44:03.387Z]
"GET /api/controller/v2/config/ HTTP/1.1" 200 - 0 4018 58 44 "1>
Oct 08 11:44:10 aap.example.org automation-gateway-proxy[1919]: [2024-10-08T10:44:03.370Z]
"GET /api/galaxy/v3/plugin/ansible/search/collection-versions/?>
Oct 08 11:44:10 aap.example.org automation-gateway-proxy[1919]: [2024-10-08T10:44:03.405Z]
"GET /api/controller/v2/organizations/?role_level=notification_>
Oct 08 11:44:10 aap.example.org automation-gateway-proxy[1919]: [2024-10-08T10:44:04.366Z]
"GET /api/galaxy/_ui/v1/me/ HTTP/1.1" 200 - 0 1368 79 40 "192.1>
Oct 08 11:44:10 aap.example.org automation-gateway-proxy[1919]: [2024-10-08T10:44:04.360Z]
"GET /api/controller/v2/workflow_approvals/?page_size=200&statu>
Oct 08 11:44:10 aap.example.org automation-gateway-proxy[1919]: [2024-10-08T10:44:04.379Z]
"GET /api/controller/v2/job_templates/7/ HTTP/1.1" 200 - 0 1356>
Oct 08 11:44:10 aap.example.org automation-gateway-proxy[1919]: [2024-10-08T10:44:04.378Z]
"GET /api/galaxy/_ui/v1/feature-flags/ HTTP/1.1" 200 - 0 207 81>
Oct 08 11:44:13 aap.example.org automation-gateway-proxy[1919]: [2024-10-08 10:44:13.856][2]
[info][upstream] [external/envoy/source/common/upstream/cds_ap>
Oct 08 11:44:13 aap.example.org automation-gateway-proxy[1919]: [2024-10-08 10:44:13.856][2]
[info][upstream] [external/envoy/source/common/upstream/cds_ap
```

実行プレーンに関するコンテナー情報の取得

Automation Controller、Event-Driven Ansible、および **execution_nodes** ノードに関するコンテナー情報を取得するには、Podman コマンドの前に以下を付けます。

```
CONTAINER_HOST=unix://run/user/<user_id>/podman/podman.sock
```

または

```
CONTAINERS_STORAGE_CONF=<user_home_directory>/aap/containers/storage.conf
```

例と出力:

```
$ CONTAINER_HOST=unix://run/user/1000/podman/podman.sock podman images

REPOSITORY                                                TAG         IMAGE ID      CREATED     SIZE
registry.redhat.io/ansible-automation-platform-25/ee-supported-rhel8  latest      59d1bc680a7c  6 days
ago  2.24 GB
registry.redhat.io/ansible-automation-platform-25/ee-minimal-rhel8    latest      a64b9fc48094  6 days
ago  338 MB
```

A.3. 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

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うには、この情報を使用してください。

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 システム要件 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ます。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遅延などの要因は、すべてインストール時間に影響を及ぼします。

2. デフォルトで **./aap_install.log** に配置されるインストールログファイルを確認します。インストールディレクトリーの **ansible.cfg** ファイル内でログファイルの場所を変更できます。

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]** セクションの下にコールバック行を追加します。次に例を示します。

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

Automation Controller が 413 エラーを返す

このエラーは、**manifest.zip** ライセンスファイルが **controller_nginx_client_max_body_size** の設定よりも大きい場合に発生します。このエラーが発生した場合は、インベントリーファイルを更新して次の変数を追加します。

```
controller_nginx_client_max_body_size=5m
```

デフォルト設定の **5m** でこの問題は回避できるはずですが、必要に応じて値を増やすことができます。

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る

```
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。この問題を解決するには、使用可能な領域を増やす必要があります。システム要

件の詳細は、システム要件 を参照してください。

パッケージが利用できないため "Install container tools" タスクが失敗する

このエラーは、インストールプロセスの出力では、次のように表示されます。

```
TASK [ansible.containerized_installer.common : Install container tools]
*********************************************************************************************************
fatal: [192.0.2.1]: FAILED! => {"changed": false, "failures": ["No package crun available.", "No
package podman available.", "No package slirp4netns available.", "No package fuse-overlayfs
available."], "msg": "Failed to install some of the specified packages", "rc": 1, "results": []}
fatal: [192.0.2.2]: FAILED! => {"changed": false, "failures": ["No package crun available.", "No
package podman available.", "No package slirp4netns available.", "No package fuse-overlayfs
available."], "msg": "Failed to install some of the specified packages", "rc": 1, "results": []}
fatal: [192.0.2.3]: FAILED! => {"changed": false, "failures": ["No package crun available.", "No
package podman available.", "No package slirp4netns available.", "No package fuse-overlayfs
available."], "msg": "Failed to install some of the specified packages", "rc": 1, "results": []}
fatal: [192.0.2.4]: FAILED! => {"changed": false, "failures": ["No package crun available.", "No
package podman available.", "No package slirp4netns available.", "No package fuse-overlayfs
available."], "msg": "Failed to install some of the specified packages", "rc": 1, "results": []}
fatal: [192.0.2.5]: FAILED! => {"changed": false, "failures": ["No package crun available.", "No
package podman available.", "No package slirp4netns available.", "No package fuse-overlayfs
available."], "msg": "Failed to install some of the specified packages", "rc": 1, "results": []}
```

このエラーを修正するには、ターゲットホストで次のコマンドを実行します。

```
sudo subscription-manager register
```

A.4. 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

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を行うには、この情報を使用してください。

インストール後、Ansible Automation Platform コンテンツを初期設定するときにエラーが発生することがある

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。

たとえば、**controller_configuration_async_delay** 変数を 2 秒に設定して、再試行の遅延を増やします。この変数は、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]** セクションで設定できます。

インストールプログラムを再実行して、すべてが期待どおりに動作することを確認します。

A.5. コンテナー化された ANSIBLE AUTOMATION PLATFORM のリファレンス

コンテナー化された Ansible Automation Platform デプロイメントのアーキテクチャーを理解するには、この情報を使用してください。

Ansible Automation Platform のコンテナー化設計のアーキテクチャーを詳しく教えていただけますか?

Red Hat では、基盤となるネイティブ Red Hat Enterprise Linux テクノロジーを可能な限り多用しています。コンテナーランタイムとサービスの管理には Podman を使用します。

システムで実行中のコンテナーをリスト表示するには、**podman ps** を使用します。

```
$ 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
```

ローカルに保存されているイメージに関する情報を表示するには、**podman images** を使用します。

```
$ 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
```

コンテナー化された Ansible Automation Platform は、セキュリティーを強化するために、デフォルトで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。

インストールプログラムは、基盤となる Red Hat Enterprise Linux ホスト上でインストールプログラムを実行するファイルシステムに、次のファイルを追加します。

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

インストールルートディレクトリーには、Podman ボリュームを使用する他のコンテナー化されたサービスが含まれています。

詳細な参照情報として、以下にいくつかの例を示します。

containers 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
```

controller 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
```

receptor ディレクトリーには、自動化メッシュの設定があります。

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

また、インストール後、ローカルユーザーの /**home** ディレクトリーには、**.cache** ディレクトリーなどの他のファイルがあります。

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

サービスはデフォルトでルートレス Podman を使用して実行されます。そのため、非特権ユーザーとして **systemd** を実行するなど、他のサービスを使用できます。**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。Podman を root ユーザーとして実行すると、デフォルトで **/var/lib/containers** が使用されます。標準ユーザーの場合、**$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
```

たとえば、**.local/storage/volumes** には、**podman volume ls** の出力に表示される内容が含まれています。

```
$ podman volume ls

DRIVER      VOLUME NAME
local       d73d3fe63a957bee04b4853fd38c39bf37c321d14fdab9ee3c9df03645135788
local       postgresql
local       redis_data
local       redis_etc
local       redis_run
```

実行プレーンは、メインサービスに影響を与えないように、コントロールプレーンのメインサービスから分離されています。

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

実行プレーンサービス (Automation Controller、Event-Driven Ansible、実行ノード) は、~/**aap/containers/storage.conf** にある専用の設定を使用します。この分離により、実行プレーンコンテナーがコントロールプレーンサービスに影響を与えることが防止されます。

実行プレーンの設定は、次のいずれかのコマンドで表示できます。

```
CONTAINERS_STORAGE_CONF=~/aap/containers/storage.conf podman <subcommand>
```

```
CONTAINER_HOST=unix://run/user/<user uid>/podman/podman.sock podman <subcommand>
```

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?

ホストリソースの使用率統計情報を表示するには、次のコマンドを実行します。

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

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) のインストールに基づく出力例 (約 1.8 GB の RAM を使用):

```
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%
```

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?

コンテナーボリュームストレージは、ローカルユーザー配下の **$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. 特定のボリュームに関する詳細情報を表示するには、次のコマンドを実行します。

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

以下に例を示します。

```
$ podman volume inspect postgresql
```

出力例:

```
[
    {
        "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}}"
```

出力例:

```
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. 以下のコマンドを実行します。

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

出力例:

```
/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、次のコマンドを実行してインストールします。

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

付録B インベントリーファイル変数

次の表には、Ansible Automation Platform のインストール **inventory** ファイルで使用される変数に関する情報が記載されています。表には、RPM ベースのインストールとコンテナーベースのインストールに使用できる変数が含まれています。

B.1. ANSIBLE 変数

以下の変数は、Ansible Automation Platform がリモートホストと対話する方法を制御します。

表B.1 Ansible 変数

変数	説明
ansible_connection	ターゲットホストでタスクに使用される接続プラグイン。これは、任意の Ansible 接続プラグインの名前にできます。 SSH プロトコルのタイプは、**smart**、**ssh**、または **paramiko** です。**local** を使用して、コントロールノード自体でタスクを実行することもできます。 デフォルト = **smart**
ansible_host	**inventory_hostname** の代わりに使用するターゲットホストの IP アドレスまたは名前。
ansible_password	ホストに対して認証するためのパスワード。 この変数をプレーンテキストで保存しないでください。必ず vault を使用してください。詳細は、Keep vaulted variables safely visible を参照してください。
ansible_port	接続ポート番号。 SSH のデフォルトは **22** です。
ansible_scp_extra_args	この設定は、デフォルトの **scp** コマンドラインに常に付加されます。
ansible_sftp_extra_args	この設定は、デフォルトの **sftp** コマンドラインに常に付加されます。
ansible_shell_executable	これにより、ターゲットマシンで Ansible Controller が使用するシェルが設定され、デフォルトで **/bin/sh** に設定されている **ansible.cfg** の実行可能ファイルがオーバーライドされます。

変数	説明
ansible_shell_type	ターゲットシステムのシェルタイプ。 **ansible_shell_executable** を Bourne (sh) 以外の互換シェルに設定していない限り、この設定を使用しないでください。デフォルトでは、コマンドは sh スタイルの構文を使用してフォーマットされます。これを **csh** または **fish** に設定すると、ターゲットシステムで実行されるコマンドが代わりにそれらのシェルの構文に従います。
ansible_ssh_common_args	この設定は、**sftp**、**scp**、および **ssh** のデフォルトのコマンドラインに常に追加されます。特定のホストまたはグループの **ProxyCommand** を設定する場合に便利です。
ansible_ssh_executable	この設定は、システムの **ssh** を使用するデフォルトの動作をオーバーライドします。これにより、**ansible.cfg** の **ssh_executable** 設定をオーバーライドできます。
ansible_ssh_extra_args	この設定は、デフォルトの **ssh** コマンドラインに常に付加されます。
ansible_ssh_pipelining	SSH **pipelining** を使用するかどうかを決定します。 これにより、**ansible.cfg** の **pipelining** 設定をオーバーライドできます。SSH キーベースの認証を使用する場合、そのキーは SSH エージェントで管理される必要があります。
ansible_ssh_private_key_file	SSH で使用される秘密鍵ファイル。 複数の鍵を使用していて、SSH エージェントを使用しない場合に便利です。
ansible_user	ホストに接続する際に使用するユーザー名。 **/bin/sh** がターゲットマシンにインストールされていない場合、または sudo から実行できない場合を除き、この変数を変更しないでください。
inventory_hostname	この変数は、インベントリースクリプトまたは Ansible 設定ファイルからマシンのホスト名を取得します。この変数の値は設定できません。値は設定ファイルから取得されるため、実際のランタイムホスト名の値は、この変数によって返される値とは異なる場合があります。

関連情報

- Ansible.Builtin
- Ansible 設定

B.2. AUTOMATION HUB の変数

Automation Hub のインベントリーファイル変数です。

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
automationhub_admin_password	**hub_admin_password**	Automation Hub の管理者パスワード。この変数では特殊文字の使用が制限されています。パスワードには、/、”、@ を除く、出力可能な任意の ASCII 文字を含めることができます。	必須	
automationhub_api_token		インストールプログラムの既存のトークンを設定します。たとえば、Automation Hub UI でトークンが再生成されると、既存のトークンは無効になります。この変数を使用して、次回インストールプログラムを実行するときにインストールプログラムでそのトークンを設定します。	任意	
automationhub_auto_sign_collections	**hub_collection_auto_sign**	コレクション署名サービスが有効になっている場合、デフォルトではコレクションは自動的に署名されません。コレクションにデフォルトで署名するには、この変数を **true** に設定します。	任意	**false**
automationhub_backup_collections		Ansible Automation Hub は、**/var/lib/pulp** にアーティファクトを提供します。このアーティファクトはデフォルトで自動的にバックアップされます。**/var/lib/pulp** のバックアップまたは復元を防ぐには、この変数を **false** に設定します。	任意	**true**
automationhub_client_max_body_size	**hub_nginx_client_max_body_size**	NGINX を介して Automation Hub に送信されるデータの最大許容サイズ。	任意	**20m**

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
automationhub_collection_download_count		コレクションのダウンロード数を UI に表示するかどうかを指定します。	任意	**false**
automationhub_collection_seed_repository		**hub_seed_collections** が **true** に設定されている場合にアップロードするコンテンツの種類を制御します。有効なオプションは、**certified**、**validated** です。	任意	デフォルトでは、certified と validated の両方が有効になっています。
automationhub_collection_signing_service_key	**hub_collection_signing_key**	コレクション署名鍵ファイルへのパス。	コレクション署名サービスが有効な場合に必須です。	
automationhub_container_repair_media_type		コマンド **pulpcore-manager container-repair-media-type** を実行するかどうかを指定します。有効なオプションは、**true**、**false**、**auto** です。	任意	**auto**
automationhub_container_signing_service_key	**hub_container_signing_key**	コンテナー署名鍵ファイルへのパス。	コンテナー署名サービスが有効な場合に必須です。	
automationhub_create_default_collection_signing_service	**hub_collection_signing**	コレクション署名サービスを有効にするには、この変数を **true** に設定します。	任意	**false**

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
automationhub_create_default_container_signing_service	**hub_container_signing**	コンテナー署名サービスを有効にするには、この変数を **true** に設定します。	任意	**false**
	hub_data_path_exclude	除外する Automation Hub バックアップパス。	任意	**[]**
automationhub_disable_hsts	**hub_nginx_disable_hsts**	Automation Hub に対して HTTP Strict Transport Security (HSTS) を有効にするか無効にするかを制御します。HSTS を無効にするには、この変数を **true** に設定します。	任意	**false**
automationhub_disable_https	**hub_nginx_disable_https**	Automation Hub に対して HTTPS を有効にするか無効にするかを制御します。HTTPS を無効にするには、この変数を **true** に設定します。	任意	**false**
automationhub_enable_api_access_log		**/var/log/galaxy_api_access.log** でロギングを有効にするか無効にするかを制御します。このファイルには、ユーザー名や IP アドレスを含め、プラットフォームに対して行われたすべてのユーザーアクションが記録されます。このロギングを有効にするには、この変数を **true** に設定します。	任意	**false**
automationhub_enable_unauthenticated_collection_access		Automation Hub のコレクションまたは名前空間を表示する権限のないユーザーに対して読み取り専用アクセスを有効にするか無効にするかを制御します。読み取り専用アクセスを有効にするには、この変数を **true** に設定します。	任意	**false**

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
automationhub_enable_unauthenticated_collection_download		権限のないユーザーが Automation Hub から読み取り専用コレクションをダウンロードできるかどうかを制御します。読み取り専用コレクションのダウンロードを有効にするには、この変数を **true** に設定します。	任意	**false**
automationhub_firewalld_zone	**hub_firewall_zone**	Automation Hub 関連のファイアウォールルールが適用されるファイアウォールゾーン。これは、ゾーンの信頼レベルに基づいて、Automation Hub にアクセスできるネットワークを制御します。	任意	RPM = デフォルト設定なし。コンテナー = **public**。
automationhub_force_change_admin_password		インストール中に Automation Hub のデフォルトの管理者パスワードの変更を要求するかどうかを指定します。インストール中にデフォルトの管理者パスワードを変更することをユーザーに要求するには、**true** に設定します。	任意	**false**
automationhub_importer_settings	**hub_galaxy_importer**	**galaxy-importer.cfg** 設定ファイルに渡す設定のディクショナリー。この設定は、**galaxy-importer** サービスが Ansible コンテンツを処理および検証する方法を制御します。値には、**ansible-doc**、**ansible-lint**、**flake8** などがあります。	任意	

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
automationhub_nginx_tls_files_remote		Web 証明書のソースが、インストールプログラムに対してローカルであるか (**false**)、リモートコンポーネントサーバー上にあるか (**true**) を指定します。	任意	**automationhub_tls_files_remote** で定義された値。
automationhub_pg_cert_auth	**hub_pg_cert_auth**	Automation Hub の PostgreSQL データベースでクライアント証明書認証を有効にするか無効にするかを制御します。クライアント証明書認証を有効にするには、この変数を **true** に設定します。	任意	**false**
automationhub_pg_database	**hub_pg_database**	Automation Hub で使用される PostgreSQL データベースの名前。	任意	RPM = **automationhub**。コンテナー = **pulp**
automationhub_pg_host	**hub_pg_host**	Automation Hub で使用される PostgreSQL データベースのホスト名。	必須	RPM = **127.0.0.1**。コンテナー = デフォルトなし。
automationhub_pg_password	**hub_pg_password**	Automation Hub の PostgreSQL データベースユーザーのパスワード。この変数では特殊文字の使用が制限されています。**!**、**#**、**0**、および **@** 文字がサポートされています。他の特殊文字を使用すると、セットアップが失敗する可能性があります。	任意	

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
automationhub_pg_port	**hub_pg_port**	Automation Hub で使用される PostgreSQL データベースのポート番号。	任意	**5432**
automationhub_pg_sslmode	**hub_pg_sslmode**	Automation Hub が PostgreSQL データベースに接続するときに使用する SSL/TLS モードを制御します。有効なオプションには、**verify-full**、**verify-ca**、**require**、**prefer**、**allow**、**disable** などがあります。	任意	**prefer**
automationhub_pg_username	**hub_pg_username**	Automation Hub の PostgreSQL データベースユーザーのユーザー名。	任意	RPM = **automationhub**。コンテナー = **pulp**.
automationhub_pgclient_sslcert	**hub_pg_tls_cert**	Automation Hub の PostgreSQL SSL/TLS 証明書ファイルへのパス。	クライアント証明書認証を使用する場合に必要です。	
automationhub_pgclient_sslkey	**hub_pg_tls_key**	Automation Hub の PostgreSQL SSL/TLS 鍵ファイルへのパス。	クライアント証明書認証を使用する場合に必要です。	

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
automationhub_pgclient_tls_files_remote		PostgreSQL クライアント証明書のソースが、インストールプログラムに対してローカルであるか (**false**)、リモートコンポーネントサーバー上にあるか (**true**) を指定します。	任意	**automationhub_tls_files_remote** で定義された値。
automationhub_require_content_approval		Automation Hub に対してコンテンツ署名を有効にするか無効にするかを制御します。デフォルトでは、コレクションを Automation Hub にアップロードした場合、ユーザーにコレクションを提供する前に、管理者がコレクションを承認する必要があります。コンテンツ承認フローを無効にするには、変数を **false** に設定します。	任意	**true**
automationhub_restore_signing_keys		既存の署名鍵をバックアップから復元するかどうかを制御します。既存の署名鍵の復元を無効にするには、**false** に設定します。	任意	**true**

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
automationhub_seed_collections	**hub_seed_collections**	コレクションのプリロードを有効にするかどうかを制御します。バンドルインストーラーを実行すると、検証済みコンテンツが **validated** リポジトリーにアップロードされ、認定済みコンテンツが **rh-certified** リポジトリーにアップロードされます。デフォルトでは、認定済みコンテンツと検証済みコンテンツの両方がアップロードされます。コンテンツをプリロードしない場合は、この変数を **false** に設定します。RPM ベースのインストーラーの場合、必要なコンテンツの種類が 1 つだけであれば、この変数を **true** に設定し、**automationhub_collection_seed_repository** 変数を、含めるコンテンツの種類に設定します。	任意	**true**
automationhub_ssl_cert	**hub_tls_cert**	Automation Hub の SSL/TLS 証明書ファイルへのパス。	任意	
automationhub_ssl_key	**hub_tls_key**	Automation Hub の SSL/TLS 鍵ファイルへのパス。	任意	
automationhub_tls_files_remote	**hub_tls_remote**	Automation Hub が提供する証明書ファイルが、インストールプログラムに対してローカルであるか (**false**)、リモートコンポーネントサーバー上にあるか (**true**) を指定します。	任意	**false**
automationhub_use_archive_compression	**hub_use_archive_compression**	Automation Hub のアーカイブ圧縮を有効にするか無効にするかを制御します。この機能は、**use_archive_compression** を使用してグローバルに制御できます。	任意	**true**

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
automationhub_use_db_compression	**hub_use_db_compression**	Automation Hub に対してデータベース圧縮を有効にするか無効にするかを制御します。この機能は、**use_db_compression** を使用してグローバルに制御できます。	任意	**true**
automationhub_user_headers	**hub_nginx_user_headers**	Automation Hub の NGINX 設定にさらに追加する NGINX ヘッダーのリスト。	任意	**[]**
generate_automationhub_token		インストール中に Automation Hub のトークンを生成するかどうかを制御します。デフォルトでは、新規インストール時にトークンが自動的に生成されます。**true** に設定すると、インストール中にトークンが再生成されます。	任意	**false**

	hub_extra_settings	インストール中に Automation Hub が使用する追加設定を定義します。
		
		以下に例を示します。
		
		`hub_extra_settings:`
		`- setting:`
		`REDIRECT_IS_HTTPS`
		`value: True`
			任意	**[]**

nginx_hsts_max_age	**hub_nginx_hsts_max_age**	Automation Hub に対して HTTP Strict Transport Security (HSTS) が適用される最大期間 (秒単位)。	任意	**63072000**
pulp_secret	**hub_secret_key**	Automation Hub がデータの署名と暗号化に使用するシークレットキーの値。	任意	

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
	hub_azure_account_key	Azure Blob ストレージアカウントキー。	Azure Blob ストレージバックエンドを使用する場合に必須です。	
	hub_azure_account_name	Azure Blob ストレージに関連付けられているアカウント名。	Azure Blob ストレージバックエンドを使用する場合に必要です。	
	hub_azure_container	Azure Blob ストレージコンテナーの名前。	任意	**pulp**
	hub_azure_extra_settings	Azure Blob ストレージバックエンドの追加パラメーターを定義します。パラメーターのリストの詳細は、django-storages ドキュメント - Azure Storage を参照してください。	任意	**{}**
	hub_collection_signing_pass	自動化コンテンツコレクション署名サービスのパスワード。	コレクション署名サービスがパスフレーズで保護されている場合に必須です。	

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
	hub_collection_signing_service	コレクションに署名するためのサービス。	任意	**ansible-default**
	hub_container_signing_pass	自動化コンテンツコンテナー署名サービスのパスワード。	コンテナー署名サービスがパスフレーズで保護されている場合に必須です。	
	hub_container_signing_service	コンテナーに署名するためのサービス。	任意	**container-default**
	hub_nginx_http_port	Automation Hub が HTTP リクエストをリッスンするポート番号。	任意	**8081**
	hub_nginx_https_port	Automation Hub が HTTPS リクエストをリッスンするポート番号。	任意	**8444**
nginx_tls_protocols	**hub_nginx_https_protocols**	HTTPS トラフィックを処理するときに Automation Hub がサポートするプロトコル。	任意	RPM = **[TLSv1.2]**。コンテナー = **[TLSv1.2, TLSv1.3]**。
	hub_pg_socket	Automation Hub が PostgreSQL データベースに接続するために使用する UNIX ソケット。	任意	

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
	hub_s3_access_key	AWS S3 アクセスキー。	AWS S3 ストレージバックエンドを使用する場合に必須です。	
	hub_s3_bucket_name	AWS S3 ストレージバケットの名前。	任意	**pulp**
	hub_s3_extra_settings	AWS S3 ストレージバックエンドの追加パラメーターを定義するために使用されます。パラメーターの一覧の詳細は、django-storages のドキュメント - Amazon S3 を参照してください。	任意	**{}**
	hub_s3_secret_key	AWS S3 シークレットキー。	AWS S3 ストレージバックエンドを使用する場合に必須です。	
	hub_shared_data_mount_opts	ネットワークファイルシステム (NFS) 共有のマウントオプション。	任意	**rw,sync,hard**

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
	hub_shared_data_path	読み取り、書き込み、実行 (RWX) アクセスを持つネットワークファイルシステム (NFS) 共有へのパス。値は **host:dir** の形式と同じである必要があります (例: **nfs-server.example.com:/exports/hub**)。	**file** ストレージバックエンドを備えた Automation Hub のインスタンスを複数インストールする場合に必須です。Automation Hub のインスタンスを 1 つだけインストールする場合は任意です。	
	hub_storage_backend	Automation Hub ストレージバックエンドタイプ。使用できる値には、**azure**、**file**、**s3** が含まれます。	任意	**file**
	hub_workers	Automation Hub ワーカーの数。	任意	**2**

B.3. AUTOMATION CONTROLLER の変数

Automation Controller のインベントリーファイル変数です。

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
admin_email	**controller_admin_email**	Django が Automation Controller の管理者ユーザーに使用するメールアドレス。	任意	**admin@example.com**
admin_password	**controller_admin_password**	Automation Controller の管理者パスワード。この変数では特殊文字の使用が制限されています。パスワードには、/、”、@ を除く、出力可能な任意の ASCII 文字を含めることができます。	必須	
admin_username	**controller_admin_user**	Automation Controller で管理者ユーザーを識別および作成するために使用するユーザー名。	任意	**admin**
automationcontroller_client_max_body_size	**controller_nginx_client_max_body_size**	NGINX を介して Automation Controller に送信されるデータの最大許容サイズ。	任意	**5m**
automationcontroller_use_archive_compression	**controller_use_archive_compression**	Automation Controller のアーカイブ圧縮を有効にするか無効にするかを制御します。この機能は、**use_archive_compression** を使用してグローバルに制御できます。	任意	**true**
automationcontroller_use_db_compression	**controller_use_db_compression**	Automation Controller に対してデータベース圧縮を有効にするか無効にするかを制御します。この機能は、**use_db_compression** を使用してグローバルに制御できます。	任意	**true**
awx_pg_cert_auth	**controller_pg_cert_auth**	Automation Controller の PostgreSQL データベースでクライアント証明書認証を有効にするか無効にするかを制御します。クライアント証明書認証を有効にするには、この変数を **true** に設定します。	任意	**false**

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
controller_firewalld_zone	**controller_firewall_zone**	Automation Controller 関連のファイアウォールルールが適用されるファイアウォールゾーン。これは、ゾーンの信頼レベルに基づいて、Automation Controller にアクセスできるネットワークを制御します。	任意	**public**
controller_nginx_tls_files_remote		Web 証明書のソースが、インストールプログラムに対してローカルであるか (**false**)、リモートコンポーネントサーバー上にあるか (**true**) を指定します。	任意	**controller_tls_files_remote** で定義された値。
controller_pgclient_tls_files_remote		PostgreSQL クライアント証明書のソースが、インストールプログラムに対してローカルであるか (**false**)、リモートコンポーネントサーバー上にあるか (**true**) を指定します。	任意	**controller_tls_files_remote** で定義された値。
controller_tls_files_remote	**controller_tls_remote**	Automation Controller が提供する証明書ファイルが、インストールプログラムに対してローカルであるか (**false**)、リモートコンポーネントサーバー上にあるか (**true**) を指定します。	任意	**false**
nginx_disable_hsts	**controller_nginx_disable_hsts**	Automation Controller に対して HTTP Strict Transport Security (HSTS) を有効にするか無効にするかを制御します。HSTS を無効にするには、この変数を **true** に設定します。	任意	**false**

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
nginx_disable_https	**controller_nginx_disable_https**	Automation Controller に対して HTTPS を有効にするか無効にするかを制御します。HTTPS を無効にするには、この変数を **true** に設定します。	任意	**false**
nginx_hsts_max_age	**controller_nginx_hsts_max_age**	Automation Controller に対して HTTP Strict Transport Security (HSTS) が適用される最大期間 (秒単位)。	任意	**63072000**
nginx_http_port	**controller_nginx_http_port**	Automation Controller が HTTP リクエストをリッスンするポート番号。	任意	RPM = **80**。コンテナー = **8080**
nginx_https_port	**controller_nginx_https_port**	Automation Controller が HTTPS リクエストをリッスンするポート番号。	任意	RPM = **443**。コンテナー = **8443**
nginx_tls_protocols	**controller_nginx_https_protocols**	HTTPS トラフィックを処理するときに Automation Controller がサポートするプロトコル。	任意	RPM = **[TLSv1.2]**。コンテナー = **[TLSv1.2, TLSv1.3]**
nginx_user_headers	**controller_nginx_user_headers**	Automation Controller の NGINX 設定にさらに追加する NGINX ヘッダーのリスト。	任意	**[]**
	controller_create_preload_data	インストール中にプリロードされるコンテンツを作成するかどうかを制御します。	任意	**true**

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
node_state		ノードまたはノードのグループのステータス。有効なオプションは、**active**、クラスターからノードを削除する **deprovision**、またはレガシーの分離ノードを実行ノードに移行する **iso_migrate** です。	任意	**active**
node_type	コンテナーの同等の変数は、**receptor_type** を参照してください。	**[automationcontroller]** グループの場合、以下の 2 つのオプションがあります。 ● **node_type=control**: ノードはプロジェクトとインベントリーの更新のみを実行し、通常のジョブは実行しません。 ● **node_type=hybrid**: ノードはすべてを実行します。 **[execution_nodes]** グループの場合、以下の 2 つのオプションがあります。 ● **node_type=hop**: ノードはジョブを実行ノードに転送します。 ● **node_type=execution**: ノードはジョブを実行できます。	任意	**[automationcontroller]** の場合 = **hybrid**、**[execution_nodes]** の場合 = **execution**
peers	コンテナーの同等の変数は、**receptor_peers** を参照してください。	特定のホストまたはグループがどのノードに接続するかを示すために使用されます。この変数が定義されている場合は、特定のホストまたはグループへの送信接続が常に確立されます。この変数には、インベントリーからのホストとグループのコンマ区切りリストを指定できます。これは、**receptor.conf** ファイルの作成に使用される一連のホストに解決されます。	任意	

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
pg_database	**controller_pg_database**	Automation Controller が使用する PostgreSQL データベースの名前。	任意	**awx**
pg_host	**controller_pg_host**	Automation Controller が使用する PostgreSQL データベースのホスト名。	必須	
pg_password	**controller_pg_password**	Automation Controller の PostgreSQL データベースユーザーのパスワード。この変数では特殊文字の使用が制限されています。**!**、**#**、**0**、および **@** 文字がサポートされています。他の特殊文字を使用すると、セットアップが失敗する可能性があります。	クライアント証明書認証を使用しない場合に必要です。	
pg_port	**controller_pg_port**	Automation Controller が使用する PostgreSQL データベースのポート番号。	任意	**5432**
pg_sslmode	**controller_pg_sslmode**	Automation Controller が PostgreSQL データベースに接続するときに使用する SSL/TLS モードを制御します。有効なオプションには、**verify-full**、**verify-ca**、**require**、**prefer**、**allow**、**disable** などがあります。	任意	**prefer**
pg_username	**controller_pg_username**	Automation Controller の PostgreSQL データベースユーザーのユーザー名。	任意	**awx**
pgclient_sslcert	**controller_pg_tls_cert**	Automation Controller の PostgreSQL SSL/TLS 証明書ファイルへのパス。	クライアント証明書認証を使用する場合に必要です。	

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
pgclient_sslkey	**controller_pg_tls_key**	Automation Controller の PostgreSQL SSL/TLS 鍵ファイルへのパス。	クライアント証明書認証を使用する場合に必要です。	
precreate_partition_hours		**pg_dump** によるロックを回避するために、バックアップ開始前に何時間分のイベントテーブルのパーティションを事前に作成しておくか。	任意	3
uwsgi_listen_queue_size	**controller_uwsgi_listen_queue_size**	**uwsgi_processes** がリクエストを処理できるようになるまでに、**uwsgi** が Automation Controller のキューで許可するリクエストの数。	任意	**2048**
web_server_ssl_cert	**controller_tls_cert**	Automation Controller の SSL/TLS 証明書ファイルへのパス。	任意	
web_server_ssl_key	**controller_tls_key**	Automation Controller の SSL/TLS 鍵ファイルへのパス。	任意	
	controller_event_workers	Automation Controller 内でジョブ関連のイベントを処理するイベントワーカーの数。	任意	**4**

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
	controller_extra_settings	インストール中に Automation Controller が使用する追加設定を定義します。 以下に例を示します。 controller_extra_settings: - setting: USE_X_FORWARDED_HOST value: true	任意	**[]**
	controller_license_file	Automation Controller のライセンスファイルへのパス。		
	controller_percent_memory_capacity	Automation Controller のメモリー割り当て。	任意	**1.0** (システムメモリー合計の 100% を Automation Controller に割り当てます)
	controller_pg_socket	Automation Controller が PostgreSQL データベースに接続するために使用する UNIX ソケット。	任意	
	controller_secret_key	Automation Controller がデータの署名と暗号化に使用するシークレットキーの値。	任意	

B.4. データベースの変数

Ansible Automation Platform で使用されるデータベースのインベントリーファイル変数です。

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
install_pg_port	**postgresql_port**	PostgreSQL データベースのポート番号。	任意	**5432**
postgres_extra_settings	**postgresql_extra_settings**	PostgreSQL で使用する追加設定を定義します。 RPM の場合の使用例: postgresql_extra_settings: ssl_ciphers: 'HIGH:!aNULL:!MD5' コンテナーの場合の使用例: postgresql_extra_settings: - setting: ssl_ciphers value: 'HIGH:!aNULL:!MD5'	任意	
postgres_firewalld_zone	**postgresql_firewall_zone**	PostgreSQL 関連のファイアウォールルールが適用されるファイアウォールゾーン。これは、ゾーンの信頼レベルに基づいて、PostgreSQL にアクセスできるネットワークを制御します。	任意	RPM = デフォルト設定なし。コンテナー = **public**。
postgres_max_connections	**postgresql_max_connections**	インストーラーによって管理されるデータベースを使用している場合のデータベース同時接続の最大数。詳細は、Automation Controller の PostgreSQL データベースの設定およびメンテナンス を参照してください。	任意	**1024**
postgres_ssl_cert	**postgresql_tls_cert**	PostgreSQL の SSL/TLS 証明書ファイルへのパス。	任意	
postgres_ssl_key	**postgresql_tls_key**	PostgreSQL の SSL/TLS 鍵ファイルへのパス。	任意	

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
postgres_use_ssl	**postgresql_disable_tls**	PostgreSQL データベースに対して SSL/TLS を有効にするか無効にするかを制御します。	任意	**false**
	postgresql_admin_database	PostgreSQL データベースサーバーへの接続に使用するデータベース名。	任意	**postgres**
	postgresql_admin_password	PostgreSQL 管理者ユーザーのパスワード。これを使用すると、インストールプログラムによって各コンポーネントのデータベースと認証情報が作成されます。	**postgresql_admin_username** を使用する場合に必須です。	
	postgresql_admin_username	PostgreSQL 管理者ユーザーのユーザー名。これを使用すると、インストールプログラムによって各コンポーネントのデータベースと認証情報が作成されます。	任意	**postgres**
	postgresql_effective_cache_size	データのキャッシュに使用可能なメモリー割り当て (MB 単位)。	任意	
	postgresql_keep_databases	アンインストール時にデータベースを保持するかどうかを制御します。この変数は、インストールプログラムによって管理されるデータベースにのみ適用されます。外部 (お客様が管理する) データベースには適用されません。アンインストール時にデータベースを保持するには **true** に設定します。	任意	**false**
	postgresql_log_destination	サーバーログ出力の宛先。	任意	**/dev/stderr**

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
	postgresql_password_encryption	パスワードを暗号化するためのアルゴリズム。	任意	**scram-sha-256**
	postgresql_shared_buffers	共有メモリーバッファーのメモリー割り当て (MB 単位)。	任意	
	postgresql_tls_remote	PostgreSQL が提供する証明書ファイルが、インストールプログラムに対してローカルであるか (**false**)、リモートコンポーネントサーバー上にあるか (**true**) を指定します。	任意	**false**
	postgresql_use_archive_compression	PostgreSQL のアーカイブ圧縮を有効にするか無効にするかを制御します。この機能は、**use_archive_compression** を使用してグローバルに制御できます。	任意	**true**

B.5. EVENT-DRIVEN ANSIBLE CONTROLLER の変数

Event-Driven Ansible Controller のインベントリーファイル変数です。

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
automationedacontroller_activation_workers	**eda_activation_workers**	Event-Driven Ansible の ansible-rulebook アクティベーション Pod に使用するワーカーの数。	任意	RPM = (コアまたはスレッドの数) x 2 + 1。コンテナー = **2**
automationedacontroller_admin_email	**eda_admin_email**	Django が Event-Driven Ansible の管理者ユーザーとして使用するメールアドレス。	任意	**admin@example.com**

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
automationedacontroller_admin_password	**eda_admin_password**	Event-Driven Ansible 管理者パスワード。この変数では特殊文字の使用が制限されています。パスワードには、/、”、@ を除く、出力可能な任意の ASCII 文字を含めることができます。	必須	
automationedacontroller_admin_username	**eda_admin_user**	Event-Driven Ansible で管理者ユーザーを識別および作成するために使用するユーザー名。	任意	**admin**
automationedacontroller_backend_gunicorn_workers		ワーカーノード上で Gunicorn を通じて提供される API を処理するワーカーの数。	任意	**2**
automationedacontroller_cache_tls_files_remote		キャッシュ証明書のソースが、インストールプログラムに対してローカルであるか (**false**)、リモートコンポーネントサーバー上にあるか (**true**) を示します。	任意	**false**
automationedacontroller_client_regen_cert		プラットフォームキャッシュの Event-Driven Ansible クライアント証明書を再生成するかどうかを制御します。Event-Driven Ansible クライアント証明書を再生成するには、**true** に設定します。	任意	**false**
automationedacontroller_default_workers	**eda_workers**	アプリケーションの処理のために Event-Driven Ansible で使用するワーカーの数。	任意	コアまたはスレッドの数
automationedacontroller_disable_hsts	**eda_nginx_disable_hsts**	Event-Driven Ansible に対して HTTP Strict Transport Security (HSTS) を有効にするか無効にするかを制御します。HSTS を無効にするには、この変数を **true** に設定します。	任意	**false**

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
automationedacontroller_disable_https	**eda_nginx_disable_https**	Event-Driven Ansible に対して HTTPS を有効にするか無効にするかを制御します。HTTPS を無効にするには、この変数を **true** に設定します。	任意	**false**
automationedacontroller_event_stream_path	**eda_event_stream_prefix_path**	プラットフォームゲートウェイを介した Event-Driven Ansible イベントストリームに使用される API 接頭辞パス。	任意	**/eda-event-streams**
automationedacontroller_firewalld_zone	**eda_firewall_zone**	Event-Driven Ansible 関連のファイアウォールルールが適用されるファイアウォールゾーン。これは、ゾーンの信頼レベルに基づいて、Event-Driven Ansible にアクセスできるネットワークを制御します。	任意	RPM = デフォルト設定なし。コンテナー = **public**。
automationedacontroller_gunicorn_event_stream_workers		Event-Driven Ansible のイベントストリーミングを処理するワーカーの数。	任意	**2**
automationedacontroller_gunicorn_workers	**eda_gunicorn_workers**	Gunicorn を通じて提供される API を処理するワーカーの数。	任意	(コアまたはスレッドの数) x 2 + 1
automationedacontroller_http_port	**eda_nginx_http_port**	Event-Driven Ansible が HTTP リクエストをリッスンするポート番号。	任意	RPM = **80**。コンテナー = **8082**。
automationedacontroller_https_port	**eda_nginx_https_port**	Event-Driven Ansible が HTTPS リクエストをリッスンするポート番号。	任意	RPM = **443**。コンテナー = **8445**。

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
automationedacontroller_max_running_activations	**eda_max_running_activations**	ノードごとに同時に実行されるアクティベーションの最大数。これは 0 より大きい整数である必要があります。	任意	**12**
automationedacontroller_nginx_tls_files_remote		Web 証明書のソースが、インストールプログラムに対してローカルであるか (**false**)、リモートコンポーネントサーバー上にあるか (**true**) を示します。	任意	**false**
automationedacontroller_pg_cert_auth	**eda_pg_cert_auth**	Event-Driven Ansible の PostgreSQL データベースでクライアント証明書認証を有効にするか無効にするかを制御します。クライアント証明書認証を有効にするには、この変数を **true** に設定します。	任意	**false**
automationedacontroller_pg_database	**eda_pg_database**	Event-Driven Ansible で使用される PostgreSQL データベースの名前。	任意	RPM = **automationedacontroller**。コンテナー = **eda**。
automationedacontroller_pg_host	**eda_pg_host**	Event-Driven Ansible で使用される PostgreSQL データベースのホスト名。	必須	
automationedacontroller_pg_password	**eda_pg_password**	Event-Driven Ansible の PostgreSQL データベースのパスワード。この変数では特殊文字の使用が制限されています。**!**、**#**、**0**、および **@** 文字がサポートされています。他の特殊文字を使用すると、セットアップが失敗する可能性があります。	クライアント証明書認証を使用しない場合に必要です。	
automationedacontroller_pg_port	**eda_pg_port**	Event-Driven Ansible で使用される PostgreSQL データベースのポート番号。	任意	**5432**

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
automationedacontroller_pg_sslmode	**eda_pg_sslmode**	クライアントサーバー接続の暗号化と認証のレベルを決定します。有効なオプションには、**verify-full**、**verify-ca**、**require**、**prefer**、**allow**、**disable** などがあります。	任意	**prefer**
automationedacontroller_pg_username	**eda_pg_username**	Event-Driven Ansible の PostgreSQL データベースのユーザー名。	任意	RPM = **automationedacontroller**。コンテナー = **eda**。
automationedacontroller_pgclient_sslcert	**eda_pg_tls_cert**	Event-Driven Ansible の PostgreSQL SSL/TLS 証明書ファイルへのパス。	クライアント証明書認証を使用する場合に必要です。	
automationedacontroller_pgclient_sslkey	**eda_pg_tls_key**	Event-Driven Ansible の PostgreSQL SSL/TLS 鍵ファイルへのパス。	クライアント証明書認証を使用する場合に必要です。	
automationedacontroller_pgclient_tls_files_remote		PostgreSQL クライアント証明書のソースが、インストールプログラムに対してローカルであるか (**false**)、リモートコンポーネントサーバー上にあるか (**true**) を示します。	任意	**false**
automationedacontroller_public_event_stream_url	**eda_event_stream_url**	イベントストリームに接続するための URL。URL は先頭が **http://** または **https://** である必要があります。	任意	

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
automationedacontroller_redis_host	**eda_redis_host**	Event-Driven Ansible で使用される Redis ホストのホスト名。	任意	**[automationgateway]** インベントリーグループの最初のノード
automationedacontroller_redis_password	**eda_redis_password**	Event-Driven Ansible の Redis のパスワード。	任意	ランダムに生成された文字列
automationedacontroller_redis_port	**eda_redis_port**	Event-Driven Ansible の Redis ホストのポート番号。	任意	RPM = プラットフォームゲートウェイの実装で定義された値 (**automationgateway_redis_port**)。コンテナー = **6379**
automationedacontroller_redis_username	**eda_redis_username**	Event-Driven Ansible の Redis のユーザー名。	任意	**eda**
automationedacontroller_secret_key	**eda_secret_key**	Event-Driven Ansible がデータの署名と暗号化に使用するシークレットキーの値。	任意	

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
automationedacontroller_ssl_cert	**eda_tls_cert**	Event-Driven Ansible の SSL/TLS 証明書ファイルへのパス。	任意	
automationedacontroller_ssl_key	**eda_tls_key**	Event-Driven Ansible の SSL/TLS 鍵ファイルへのパス。	任意	
automationedacontroller_tls_files_remote	**eda_tls_remote**	Event-Driven Ansible が提供する証明書ファイルが、インストールプログラムに対してローカルであるか (**false**)、リモートコンポーネントサーバー上にあるか (**true**) を指定します。	任意	**false**
automationedacontroller_trusted_origins		信頼できるクロスサイトリクエストフォージェリー (CSRF) 送信元の **<scheme>//:<address>:<port>** という形式のホストアドレスのリスト。	任意	**[]**
automationedacontroller_use_archive_compression	**eda_use_archive_compression**	Event-Driven Ansible のアーカイブ圧縮を有効にするか無効にするかを制御します。この機能は、**use_archive_compression** を使用してグローバルに制御できます。	任意	**true**
automationedacontroller_use_db_compression	**eda_use_db_compression**	Event-Driven Ansible に対してデータベース圧縮を有効にするか無効にするかを制御します。この機能は、**use_db_compression** を使用してグローバルに制御できます。	任意	**true**
automationedacontroller_user_headers	**eda_nginx_user_headers**	Event-Driven Ansible の NGINX 設定にさらに追加する NGINX ヘッダーのリスト。	任意	**[]**

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
automationedacontroller_websocket_ssl_verify		Podman が Pod からホストへの通信に使用する Daphne WebSocket に対して SSL 検証を実行するかどうかを制御します。SSL 検証を無効にするには **false** に設定します。	任意	**true**
eda_node_type	**eda_type**	Event-Driven Ansible のノードタイプ。有効なオプションは、**api**、**event-stream**、**hybrid**、**worker** です。	任意	**hybrid**
	eda_debug	Event-Driven Ansible のデバッグモードを有効にするか無効にするかを制御します。Event-Driven Ansible のデバッグモードを有効にするには、**true** に設定します。	任意	**false**

	eda_extra_settings	インストール中に Event-Driven Ansible が使用する追加設定を定義します。
		
		以下に例を示します。
		
		`eda_extra_settings:`
		`  - setting: RULEBOOK_READINESS_TIMEOUT_SECONDS`
		`  value: 120`
			任意	**[]**

	eda_nginx_client_max_body_size	NGINX を介して Event-Driven Ansible に送信されるデータの最大許容サイズ。	任意	**1m**
	eda_nginx_hsts_max_age	Event-Driven Ansible に対して HTTP Strict Transport Security (HSTS) が適用される最大期間 (秒単位)。	任意	**63072000**

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
nginx_tls_protocols	**eda_nginx_https_protocols**	Event-Driven Ansible が HTTPS トラフィックを処理するときにサポートするプロトコル。	任意	RPM = **[TLSv1.2]**。コンテナー = **[TLSv1.2, TLSv1.3]**。
	eda_pg_socket	Event-Driven Ansible が PostgreSQL データベースに接続するために使用する UNIX ソケット。	任意	
redis_disable_tls	**eda_redis_disable_tls**	Event-Driven Ansible の Redis に対して TLS を有効にするか無効にするかを制御します。TLS を無効にするには、この変数を true に設定します。	任意	**false**
	eda_redis_tls_cert	Event-Driven Ansible の Redis 証明書ファイルへのパス。	任意	
	eda_redis_tls_key	Event-Driven Ansible の Redis 鍵ファイルへのパス。	任意	
	eda_safe_plugins	Event-Driven Ansible 内での実行を許可するプラグインのリスト。 詳細は、Event-Driven Ansible Controller への安全なプラグイン変数の追加 を参照してください。	任意	**[]**

B.6. 一般的な変数

Ansible Automation Platform の一般的なインベントリーファイル変数です。

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
aap_ca_cert_file	**ca_tls_cert**	すべての Ansible Automation Platform サービスの SSL/TLS 証明書を生成するために使用するユーザー指定の CA 証明書ファイルへのパス。詳細は、カスタム TLS 証明書の使用 を参照してください。	任意	
aap_ca_cert_files_remote	**ca_tls_remote**	CA 証明書ファイルが、インストールプログラムに対してローカルであるか (**false**)、リモートコンポーネントサーバー上にあるか (**true**) を指定します。	任意	**false**
aap_ca_cert_size		内部的に管理される CA 証明書の秘密鍵のビットサイズ。	任意	**4096**
aap_ca_key_file	**ca_tls_key**	**aap_ca_cert_file** (RPM) および **ca_tls_cert** (コンテナー) で指定した CA 証明書の鍵ファイルへのパス。詳細は、カスタム TLS 証明書の使用 を参照してください。	任意	
aap_ca_passphrase_cipher		内部的に管理される CA 証明書の秘密鍵に署名するために使用する暗号。	任意	**aes256**
aap_ca_regenerate		内部的に管理される CA 証明書の鍵ペアを再生成するかどうかを指定します。	任意	**false**
aap_service_cert_size		内部 CA によって管理されるコンポーネント鍵ペアのビットサイズ。	任意	**4096**
aap_service_regen_cert		内部 CA によって管理されるコンポーネント鍵ペアを再生成するかどうかを指定します。	任意	**false**

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
aap_service_san_records		サービスに署名するための追加の SAN レコードのリスト。これらを、グループ変数またはすべての変数ではなく、ホスト変数としてインベントリーファイル内のコンポーネントに割り当ててください。すべての文字列に、**DNS:** や **IP:** など、対応する SAN オプションの接頭辞が含まれている必要があります。	任意	**[]**
backup_dest		**setup.sh** に対してローカルの最終バックアップファイルのディレクトリー。	任意	**setup_dir** で定義された値。
backup_dir	**backup_dir**	バックアップファイルを保存するために使用するディレクトリー。	任意	RPM = **/var/backups/automation-platform/**。コンテナー = **~/backups**
backup_file_prefix		最終バックアップファイルのファイルバックアップ名に使用する接頭辞。	任意	**automation-platform-backup**

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
bundle_install	**bundle_install**	オフラインまたはバンドルインストールを実行するかどうかを制御します。オフラインまたはバンドルインストールを有効にするには、この変数を **true** に設定します。	任意	セットアップインストールプログラムを使用する場合は **false** です。セットアップバンドルインストールプログラムを使用する場合は **true** です。
bundle_install_folder	**bundle_dir**	バンドルインストールを実行するときに使用するバンドルディレクトリーへのパス。	**bundle_install=true** の場合に必須	RPM = **/var/lib/ansible-automation-platform-bundle**。コンテナー = **<current_dir>/bundle**。

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
custom_ca_cert	**custom_ca_cert**	カスタム CA 証明書ファイルへのパス。手動で提供する TLS 証明書のいずれかがカスタム CA によって署名されている場合は、これが必要です。詳細は、カスタム TLS 証明書の使用 を参照してください。	任意	
enable_insights_collection		デフォルトのインストールでは、ノードが Subscription Manager に登録されている場合、そのノードは Red Hat Ansible Automation Platform サービスの Red Hat Insights for Red Hat Ansible Automation Platform に登録されます。この機能を無効にするには **false** に設定します。	任意	**true**
registry_password	**registry_password**	**registry_url** で定義されたレジストリーソースにアクセスするためのパスワード認証情報。詳細は、registry_username お registry_password の設定 を参照してください。	RPM = **registry_url** へのアクセスにパスワードが必要な場合に必須。コンテナー = **registry_auth=true** の場合に必須。	
registry_url	**registry_url**	実行環境イメージの取得元となるレジストリーソースの URL。	任意	**registry.redhat.io**

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
registry_username	**registry_username**	**registry_url** で定義されたレジストリーソースにアクセスするためのユーザー名認証情報。詳細は、registry_username お registry_password の設定 を参照してください。	RPM = **registry_url** へのアクセスにパスワードが必要な場合に必須。コンテナー = **registry_auth=true** の場合に必須。	
registry_verify_ssl	**registry_tls_verify**	HTTPS リクエストを行うときに SSL/TLS 証明書の検証を有効にするか無効にするかを制御します。	任意	**true**
restore_backup_file		プラットフォームの復元に使用する tar ファイルへのパス。	任意	**{{ setup_dir }}/automation-platform-backup-latest.tar.gz**
restore_file_prefix		一時的に準備される復元コンポーネントのパスの接頭辞。	任意	**automation-platform-restore**

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
routable_hostname	**routable_hostname**	インストールプログラムを実行しているマシンが特定の URL 経由でのみターゲットホストにルーティングできる場合に使用されます。たとえば、インベントリーで短縮名を使用しており、インストールプログラムを実行しているノードがそのホストを解決するのに FQDN を使用する必要がある場合などです。**routable_hostname** が設定されていない場合は、デフォルトで **ansible_host** に設定されます。**ansible_host** を設定しなかった場合、最後の手段として **inventory_hostname** が使用されます。この変数は、**[all:vars]** セクションではなく、特定のホストのホスト変数として使用されます。詳細は、Assigning a variable to one machine: host variables を参照してください。	任意	
use_archive_compression	**use_archive_compression**	ファイルシステム関連のバックアップファイルを、バックアップ操作を実行するためにホストに送信する前に圧縮するかどうかをグローバルレベルで制御します。**true** に設定すると、各 Ansible Automation Platform ホストで **tar.gz** ファイルが生成され、gzip 圧縮が使用されます。**false** に設定すると、単純な tar ファイルが生成されます。 この機能は、**<component_name>_use_archive_compression** 変数を使用してコンポーネントレベルで制御できます。	任意	**true**

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
use_db_compression	**use_db_compression**	データベース関連のバックアップファイルを、バックアップ操作を実行するためにホストに送信する前に圧縮するかどうかをグローバルレベルで制御します。 この機能は、**<component_name>_use_db_compression** 変数を使用してコンポーネントレベルで制御できます。	任意	**true**
	ca_tls_key_passphrase	**ca_tls_key** で指定した鍵を復号化するために使用するパスフレーズ。	任意	
	client_request_timeout	エンドユーザーの要求の HTTP タイムアウトを設定します。最小値は **10** 秒です。	任意	**30**
	container_compress	コンテナーイメージを圧縮するために使用する圧縮ソフトウェア。	任意	**gzip**
	container_keep_images	Ansible Automation Platform をアンインストールするときにコンテナーイメージを保持するかどうかを制御します。Ansible Automation Platform をアンインストールするときにコンテナーイメージを保持するには、**true** に設定します。	任意	**false**
	container_pull_images	インストール中に新しいコンテナーイメージをプルするかどうかを制御します。インストール中に新しいコンテナーイメージをプルしない場合は、**false** に設定します。	任意	**true**
	images_tmp_dir	インストール中にインストールプログラムがコンテナーイメージを一時的に保存するディレクトリー。	任意	システムの一時ディレクトリー。

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
	pcp_firewall_zone	Performance Co-Pilot 関連のファイアウォールルールが適用されるファイアウォールゾーン。これは、ゾーンの信頼レベルに基づいて、Performance Co-Pilot にアクセスできるネットワークを制御します。	任意	public
	pcp_use_archive_compression	Performance Co-Pilot のアーカイブ圧縮を有効にするか無効にするかを制御します。この機能は、**use_archive_compression** を使用してグローバルに制御できます。	任意	**true**
	registry_auth	レジストリー認証を使用するかどうかを設定します。この変数を true に設定すると、**registry_username** と **registry_password** が必須になります。	任意	**true**
	registry_ns_aap	Ansible Automation Platform レジストリーの名前空間。	任意	**ansible-automation-platform-26**
	registry_ns_rhel	RHEL レジストリーの名前空間。	任意	**rhel8**

B.7. イメージの変数

イメージのインベントリーファイル変数です。

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
extra_images		デプロイ中に設定されたコンテナーレジストリーからプルする追加のコンテナーイメージ。	任意	**ansible-builder-rhel8**
	controller_image	Automation Controller のコンテナーイメージ。	任意	**controller-rhel8:latest**
	de_extra_images	デプロイ中に設定されたコンテナーレジストリーからプルする追加の決定環境コンテナーイメージ。	任意	**[]**
	de_supported_image	サポートされている決定環境コンテナーイメージ。	任意	**de-supported-rhel8:latest**
	eda_image	Event-Driven Ansible のバックエンドコンテナーイメージ。	任意	**eda-controller-rhel8:latest**
	eda_web_image	Event-Driven Ansible のフロントエンドコンテナーイメージ。	任意	**eda-controller-ui-rhel8:latest**
	ee_extra_images	デプロイ中に設定されたコンテナーレジストリーからプルする追加の実行環境コンテナーイメージ。	任意	**[]**
	ee_minimal_image	最小限の実行環境コンテナーイメージ。	任意	**ee-minimal-rhel8:latest**

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
	ee_supported_image	サポートされている実行環境コンテナーイメージ。	任意	**ee-supported-rhel8:latest**
	gateway_image	プラットフォームゲートウェイのコンテナーイメージ。	任意	**gateway-rhel8:latest**
	gateway_proxy_image	プラットフォームゲートウェイプロキシーのコンテナーイメージ。	任意	**gateway-proxy-rhel8:latest**
	hub_image	Automation Hub のバックエンドコンテナーイメージ。	任意	**hub-rhel8:latest**
	hub_web_image	Automation Hub のフロントエンドコンテナーイメージ。	任意	**hub-web-rhel8:latest**
	pcp_image	Performance Co-Pilot のコンテナーイメージ。	任意	**pcp:latest**
	postgresql_image	PostgreSQL のコンテナーイメージ。	任意	**postgresql-15:latest**
	receptor_image	Receptor のコンテナーイメージ。	任意	**receptor-rhel8:latest**
	redis_image	Redis のコンテナーイメージ。	任意	**redis-6:latest**

B.8. プラットフォームゲートウェイの変数

プラットフォームゲートウェイのインベントリーファイル変数です。

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
automationgateway_admin_email	**gateway_admin_email**	Django がプラットフォームゲートウェイの管理者ユーザーに使用するメールアドレス。	任意	**admin@example.com**
automationgateway_admin_password	**gateway_admin_password**	プラットフォームゲートウェイ管理者のパスワード。この変数では特殊文字の使用が制限されています。パスワードには、/、”、@ を除く、出力可能な任意の ASCII 文字を含めることができます。	必須	
automationgateway_admin_username	**gateway_admin_user**	プラットフォームゲートウェイで管理者ユーザーを識別および作成するために使用するユーザー名。	任意	**admin**
automationgateway_cache_cert	**gateway_redis_tls_cert**	プラットフォームゲートウェイの Redis 証明書ファイルへのパス。	任意	
automationgateway_cache_key	**gateway_redis_tls_key**	プラットフォームゲートウェイの Redis 鍵ファイルへのパス。	任意	
automationgateway_cache_tls_files_remote		キャッシュクライアント証明書ファイルが、インストールプログラムに対してローカルであるか (**false**)、リモートコンポーネントサーバー上にあるか (**true**) を指定します。	任意	この値は、**automationgateway_tls_files_remote** で定義されます。デフォルトでは **false** に設定されます。

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
automationgateway_client_regen_cert		プラットフォームキャッシュのプラットフォームゲートウェイクライアント証明書を再生成するかどうかを制御します。プラットフォームゲートウェイクライアント証明書を再生成するには、**true** に設定します。	任意	**false**
automationgateway_control_plane_port	**gateway_control_plane_port**	プラットフォームゲートウェイのコントロールプレーンのポート番号。	任意	**50051**
automationgateway_disable_hsts	**gateway_nginx_disable_hsts**	プラットフォームゲートウェイに対して HTTP Strict Transport Security (HSTS) を有効にするか無効にするかを制御します。HSTS を無効にするには、この変数を **true** に設定します。	任意	**false**
automationgateway_disable_https	**gateway_nginx_disable_https**	プラットフォームゲートウェイに対して HTTPS を有効にするか無効にするかを制御します。HTTPS を無効にするには、この変数を **true** に設定します。	任意	RPM = **disable_https** で定義された値。デフォルトは **false** です。コンテナー = **false** 。
automationgateway_firewalld_zone	**gateway_proxy_firewall_zone**	プラットフォームゲートウェイ関連のファイアウォールルールが適用されるファイアウォールゾーン。これは、ゾーンの信頼レベルに基づいて、プラットフォームゲートウェイにアクセスできるネットワークを制御します。	任意	RPM = デフォルト設定なし。コンテナー = 'public' 。

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
automationgateway_grpc_auth_service_timeout	**gateway_grpc_auth_service_timeout**	プラットフォームゲートウェイ上の gRPC サービスに対するリクエストのタイムアウト期間 (秒単位)。	任意	**30s**
automationgateway_grpc_server_max_threads_per_process	**gateway_grpc_server_max_threads_per_process**	プラットフォームゲートウェイでリクエストを処理するために各 gRPC サーバープロセスが作成できるスレッドの最大数。	任意	**10**
automationgateway_grpc_server_processes	**gateway_grpc_server_processes**	プラットフォームゲートウェイで gRPC リクエストを処理するプロセスの数。	任意	**5**
automationgateway_http_port	**gateway_nginx_http_port**	プラットフォームゲートウェイが HTTP リクエストをリッスンするポート番号。	任意	RPM = **8080**。コンテナー = **8083**。
automationgateway_https_port	**gateway_nginx_https_port**	プラットフォームゲートウェイが HTTPS リクエストをリッスンするポート番号。	任意	RPM = **8443**。コンテナー = **8446**。
automationgateway_main_url	**gateway_main_url**	クライアントが接続するプラットフォームゲートウェイのメインインスタンスの URL。クラスターデプロイメントを実行し、コンポーネントのサーバーではなくロードバランサーの URL を使用する必要がある場合に使用します。URL は先頭が **http://** または **https://** である必要があります。	任意	

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
automationgateway_nginx_tls_files_remote		Web 証明書のソースが、インストールプログラムに対してローカルであるか (**false**)、リモートコンポーネントサーバー上にあるか (**true**) を示します。	任意	この値は、**automationgateway_tls_files_remote** で定義されます。デフォルトでは **false** に設定されます。
automationgateway_pg_cert_auth	**gateway_pg_cert_auth**	プラットフォームゲートウェイの PostgreSQL データベースでクライアント証明書認証を有効にするか無効にするかを制御します。クライアント証明書認証を有効にするには、この変数を **true** に設定します。	任意	**false**
automationgateway_pg_database	**gateway_pg_database**	プラットフォームゲートウェイで使用される PostgreSQL データベースの名前。	任意	RPM = **automationgateway**。コンテナー = **gateway**。
automationgateway_pg_host	**gateway_pg_host**	プラットフォームゲートウェイで使用される PostgreSQL データベースのホスト名。	必須	

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
automationgateway_pg_password	**gateway_pg_password**	プラットフォームゲートウェイの PostgreSQL データベースユーザーのパスワード。この変数では特殊文字の使用が制限されています。**!**、**#**、**0**、および **@** 文字がサポートされています。他の特殊文字を使用すると、セットアップが失敗する可能性があります。	任意	
automationgateway_pg_port	**gateway_pg_port**	プラットフォームゲートウェイで使用される PostgreSQL データベースのポート番号。	任意	**5432**
automationgateway_pg_sslmode	**gateway_pg_sslmode**	プラットフォームゲートウェイが PostgreSQL データベースに接続するときに使用する SSL モードを制御します。有効なオプションには、**verify-full**、**verify-ca**、**require**、**prefer**、**allow**、**disable** などがあります。	任意	**prefer**
automationgateway_pg_username	**gateway_pg_username**	プラットフォームゲートウェイの PostgreSQL データベースユーザーのユーザー名。	任意	RPM = **automationgateway**。コンテナー = **gateway**
automationgateway_pgclient_sslcert	**gateway_pg_tls_cert**	プラットフォームゲートウェイの PostgreSQL SSL/TLS 証明書ファイルへのパス。	クライアント証明書認証を使用する場合に必要です。	

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
automationgateway_pgclient_sslkey	**gateway_pg_tls_key**	プラットフォームゲートウェイの PostgreSQL SSL/TLS 鍵ファイルへのパス。	クライアント証明書認証を使用する場合に必要です。	
automationgateway_pgclient_tls_files_remote		PostgreSQL クライアント証明書のソースが、インストールプログラムに対してローカルであるか (**false**)、リモートコンポーネントサーバー上にあるか (**true**) を示します。	任意	この値は、**automationgateway_tls_files_remote** で定義されます。デフォルトでは **false** に設定されます。
automationgateway_redis_host	**gateway_redis_host**	プラットフォームゲートウェイで使用される Redis ホストのホスト名。	任意	**[automationgateway]** インベントリーグループの最初のノード。
automationgateway_redis_password	**gateway_redis_password**	プラットフォームゲートウェイの Redis のパスワード。	任意	ランダムに生成された文字列。

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
automationgateway_redis_username	**gateway_redis_username**	プラットフォームゲートウェイの Redis のユーザー名。	任意	**gateway**
automationgateway_secret_key	**gateway_secret_key**	プラットフォームゲートウェイがデータの署名と暗号化に使用するシークレットキーの値。	任意	
automationgateway_ssl_cert	**gateway_tls_cert**	プラットフォームゲートウェイの SSL/TLS 証明書ファイルへのパス。	任意	
automationgateway_ssl_key	**gateway_tls_key**	プラットフォームゲートウェイの SSL/TLS 鍵ファイルへのパス。	任意	
automationgateway_tls_files_remote	**gateway_tls_remote**	プラットフォームゲートウェイが提供する証明書ファイルが、インストールプログラムに対してローカルであるか (**false**)、リモートコンポーネントサーバー上にあるか (**true**) を指定します。	任意	**false**
automationgateway_use_archive_compression	**gateway_use_archive_compression**	プラットフォームゲートウェイのアーカイブ圧縮を有効にするか無効にするかを制御します。この機能は、**use_archive_compression** を使用してグローバルに制御できます。	任意	**true**
automationgateway_use_db_compression	**gateway_use_db_compression**	プラットフォームゲートウェイのデータベース圧縮を有効にするか無効にするかを制御します。この機能は、**use_db_compression** を使用してグローバルに制御できます。	任意	**true**
automationgateway_user_headers	**gateway_nginx_user_headers**	プラットフォームゲートウェイの NGINX 設定にさらに追加する NGINX ヘッダーのリスト。	任意	**[]**

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
automationgateway_verify_ssl		インストール中にプラットフォームゲートウェイから自身への呼び出しを行うときに、プラットフォームゲートウェイの Web 証明書を検証するかどうかを指定します。Web 証明書の検証を無効にするには **false** に設定します。	任意	**true**
automationgatewayproxy_disable_https	**envoy_disable_https**	プラットフォーム UI にアクセスするときに HTTPS を無効にするかどうかを制御します。HTTPS を無効にするには **true** に設定します (代わりに HTTP が使用されます)。	任意	RPM = **disable_https** で定義された値。デフォルトは **false** です。コンテナー = **false**。
automationgatewayproxy_http_port	**envoy_http_port**	Envoy プロキシーが HTTP 着信接続をリッスンするポート番号。	任意	**80**
automationgatewayproxy_https_port	**envoy_https_port**	Envoy プロキシーが HTTPS 着信接続をリッスンするポート番号。	任意	**443**
nginx_tls_protocols	**gateway_nginx_https_protocols**	HTTPS トラフィックを処理するときにプラットフォームゲートウェイがサポートするプロトコル。	任意	RPM = **[TLSv1.2]**。コンテナー = **[TLSv1.2, TLSv1.3]**。
redis_disable_tls	**gateway_redis_disable_tls**	プラットフォームゲートウェイの Redis に対して TLS を有効にするか無効にするかを制御します。TLS を無効にするには、この変数を **true** に設定します。	任意	**false**

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
redis_port	**gateway_redis_port**	プラットフォームゲートウェイの Redis ホストのポート番号。	任意	**6379**
	gateway_extra_settings	インストール中にプラットフォームゲートウェイが使用する追加設定を定義します。 以下に例を示します。 gateway_extra_settings: - setting: OAUTH2_PROVIDER['ACCESS_TOKEN_EXPIRE_SECONDS'] value: 600	任意	**[]**
	gateway_nginx_client_max_body_size	NGINX を介してプラットフォームゲートウェイに送信されるデータの最大許容サイズ。	任意	**5m**
	gateway_nginx_hsts_max_age	プラットフォームゲートウェイに対して HTTP Strict Transport Security (HSTS) が適用される最大期間 (秒単位)。	任意	**63072000**
	gateway_uwsgi_listen_queue_size	**uwsgi_processes** がリクエストを処理できるようになるまでに、**uwsgi** がプラットフォームゲートウェイのキューで許可するリクエストの数。	任意	**4096**

B.9. RECEPTOR の変数

Receptor のインベントリーファイル変数です。

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
receptor_datadir		Receptor がランタイムデータとローカルアーティファクトを保存するディレクトリー。ターゲットディレクトリーに **awx** ユーザーがアクセスできる必要があります。ターゲットディレクトリーが一時ファイルシステム **tmpfs** である場合は、再起動後に正しく再マウントされていることを確認してください。これを行わないと、Receptor の作業ディレクトリーがなくなります。	任意	**/tmp/receptor**
receptor_listener_port	**receptor_port**	Receptor が他の Receptor ノードからの着信接続をリッスンするポート番号。	任意	**27199**
receptor_listener_protocol	**receptor_protocol**	トラフィックを処理するときに Receptor がサポートするプロトコル。	任意	**tcp**
receptor_log_level	**receptor_log_level**	Receptor のロギングの詳細度を制御します。有効なオプションは、**error**、**warning**、**info**、または **debug** です。	任意	**info**
receptor_tls		Receptor に対して TLS を有効にするか無効にするかを制御します。TLS を無効にするには、この変数を **false** に設定します。	任意	**true**

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
RPM の同等の変数は、**node_type** を参照してください。	**receptor_type**	**[automationcontroller]** グループの場合、以下の 2 つのオプションがあります。 • **receptor_type=control**: ノードはプロジェクトとインベントリーの更新のみを実行しますが、通常のジョブは実行しません。 • **receptor_type=hybrid**: ノードはすべてを実行します。 **[execution_nodes]** グループの場合、以下の 2 つのオプションがあります。 • **receptor_type=hop**: ノードはジョブを実行ノードに転送します。 • **receptor_type=execution**: ノードはジョブを実行できます。	任意	**[automationcontroller]** グループの場合: **hybrid**。**[execution_nodes]** グループの場合: **execution**。
RPM の同等の変数は、**peers** を参照してください。	**receptor_peers**	特定のホストが接続するノードを示すために使用されます。この変数が定義されている場合は、特定のホストへの送信接続が常に確立されます。値は、ホスト名のコンマ区切りリストである必要があります。インベントリーグループ名は使用しないでください。 これは、**receptor.conf** ファイルの作成に使用される一連のホストに解決されます。 詳細は、実行ノードの追加 を参照してください。	任意	**[]**

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
	receptor_disable_signing	Receptor ノード間の通信の署名を有効にするか無効にするかを制御します。通信の署名を無効にするには、この変数を **true** に設定します。	任意	**false**
	receptor_disable_tls	Receptor に対して TLS を有効にするか無効にするかを制御します。TLS を無効にするには、この変数を **true** に設定します。	任意	**false**
	receptor_firewall_zone	Receptor 関連のファイアウォールルールが適用されるファイアウォールゾーン。これは、ゾーンの信頼レベルに基づいて、Receptor にアクセスできるネットワークを制御します。	任意	**public**
	receptor_mintls13	Receptor が受け入れる接続を、TLS 1.3 以上を使用する接続だけに限定するかどうかを制御します。TLS 1.3 以上を使用する接続のみを受け入れるには、**true** に設定します。	任意	**false**
	receptor_signing_private_key	ネットワーク内の他の Receptor ノードとの通信に署名するために Receptor が使用する秘密鍵へのパス。	任意	
	receptor_signing_public_key	ネットワーク内の他の Receptor ノードとの通信に署名するために Receptor が使用する公開鍵へのパス。	任意	
	receptor_signing_remote	Receptor 署名ファイルがインストールプログラムに対してローカルであるか (**false**)、リモートコンポーネントサーバー上にあるか (**true**) を示します。	任意	**false**
	receptor_tls_cert	Receptor の TLS 証明書ファイルへのパス。	任意	

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
	receptor_tls_key	Receptor の TLS 鍵ファイルへのパス。	任意	
	receptor_tls_remote	Receptor が提供する証明書ファイルが、インストールプログラムに対してローカルであるか (**false**)、リモートコンポーネントサーバー上にあるか (**true**) を指定します。	任意	**false**
	receptor_use_archive_compression	Receptor のアーカイブ圧縮を有効にするか無効にするかを制御します。この機能は、**use_archive_compression** を使用してグローバルに制御できます。	任意	**true**

B.10. REDIS の変数

Redis のインベントリーファイル変数です。

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
redis_cluster_ip	**redis_cluster_ip**	Redis クラスターがクラスター内の各ホストを識別するために使用する IPv4 アドレス。**[redis]** グループでホストを定義するときに、デフォルトが望ましくない場合は、この変数を使用して IPv4 アドレスを識別します。コンテナー固有: Redis クラスターではホスト名または IPv6 アドレスを使用できません。	任意	RPM = Ansible ファクトから検出された IPv4 アドレス。IPv4 アドレスが使用できない場合は、IPv6 アドレスが使用されます。コンテナー = Ansible ファクトから検出された IPv4 アドレス。
redis_disable_mtls		Redis に対して mTLS を有効にするか無効にするかを制御します。mTLS を無効にするには、この変数を **true** に設定します。	任意	**false**
redis_firewalld_zone	**redis_firewall_zone**	Redis 関連のファイアウォールルールが適用されるファイアウォールゾーン。これは、ゾーンの信頼レベルに基づいて、Redis にアクセスできるネットワークを制御します。	任意	RPM = デフォルト設定なし。コンテナー = **public**。

RPM の変数名	コンテナーの変数名	説明	必須または任意	デフォルト
redis_hostname		ホストを識別してルーティングするときに Redis クラスターによって使用されるホスト名。デフォルトでは **routable_hostname** が使用されます。	任意	**routable_hostname** で定義された値
redis_mode	**redis_mode**	Ansible Automation Platform のインストールに使用する Redis モード。有効なオプションは、**standalone** および **cluster** です。Redis の詳細は、**インストール計画** の キャッシュおよびキューイングシステム を参照してください。	任意	**cluster**
redis_server_regen_cert		Ansible Automation Platform によって管理される Redis 用の TLS 鍵ペアを再生成するかどうかを指定します。	任意	**false**
redis_tls_cert	**redis_tls_cert**	Redis サーバーの TLS 証明書へのパス。	任意	
redis_tls_files_remote	**redis_tls_remote**	Redis が提供する証明書ファイルが、インストールプログラムに対してローカルであるか (**false**)、リモートコンポーネントサーバー上にあるか (**true**) を指定します。	任意	**false**
redis_tls_key	**redis_tls_key**	Redis サーバーの TLS 証明書の鍵へのパス。	任意	
	redis_use_archive_compression	Redis のアーカイブ圧縮を有効にするか無効にするかを制御します。この機能は、**use_archive_compression** を使用してグローバルに制御できます。	任意	**true**