Red Hat Ansible Automation Platform 2.4

Red Hat Ansible Automation Platform インストールガイド

Ansible Automation Platform のインストール

Last Updated: 2024-08-03

Red Hat Ansible Automation Platform 2.4 Red Hat Ansible Automation Platform インストールガイド

Ansible Automation Platform のインストール

法律上の通知

Copyright © 2024 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at
http://creativecommons.org/licenses/by-sa/3.0/
. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

概要

このガイドでは、サポートされているインストールシナリオに基づいて Red Hat Ansible Automation Platform をインストールする方法を説明します。

目次

はじめに

Red Hat Ansible Automation Platform に興味をお持ちいただきありがとうございます。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。

このガイドでは、Ansible Automation Platform のインストールにおけるインストール要件およびプロセスを説明します。このガイドの更新により、Ansible Automation Platform の最新リリースの情報が追加されました。

RED HAT ドキュメントへのフィードバック (英語のみ)

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

第1章 RED HAT ANSIBLE AUTOMATION PLATFORM インストールの概要

Red Hat Ansible Automation Platform インストールプログラムには柔軟性があり、サポートされている多数のインストールシナリオを使用して、Ansible Automation Platform をインストールできます。Ansible Automation Platform 2.4 以降、インストールシナリオには、IT リクエストの自動解決を導入する Event-Driven Ansible Controller のオプションのデプロイメントが含まれています。

選択したインストールシナリオに関係なく、Ansible Automation Platform のインストールには次の手順が含まれます。

Red Hat Ansible Automation Platform インストーラーのインベントリーファイルの編集

Ansible Automation Platform インストーラーのインベントリーファイルを使用すると、インストールシナリオを指定し、Ansible へのホストのデプロイを記述することができます。このドキュメントで提供されている例は、デプロイメントのシナリオをインストールするために必要なパラメーターの仕様を示しています。

Red Hat Ansible Automation Platform インストーラー設定スクリプトの実行

セットアップスクリプトは、インベントリーファイルで定義されている必須パラメーターを使用して、Private Automation Hub をインストールします。

Automation Controller インストールの確認

Ansible Automation Platform をインストールしたら、Automation Controller にログインして、インストールが成功したことを確認できます。

Automation Hub のインストールの確認

Ansible Automation Platform をインストールしたら、Automation Hub にログインして、インストールが成功したことを確認できます。

Event-Driven Ansible Controller のインストールの検証

Ansible Automation Platform をインストールしたら、Event-Driven Automation Controller にログインして、インストールが成功したことを確認できます。

関連情報

サポートされているインストールシナリオの詳細は、Red Hat Ansible Automation Platform 計画ガイドを参照してください。

1.1. 前提条件

- Red Hat Ansible Automation Platform Product Software からプラットフォームインストーラーを選択して入手している。
- ベースシステムの要件を満たすマシンにインストールしている。
- すべてのパッケージが RHEL ノードの最新バージョンに更新されている。

警告

エラーを防ぐために、Ansible Automation Platform をインストールする前に RHEL ノードを完全にアップグレードしてください。

- レジストリーサービスアカウントの作成 の手順に従って、Red Hat Registry Service Account を作成している。

関連情報

プラットフォームインストーラーの入手やシステム要件の詳細は、**Red Hat Ansible Automation Platform 計画ガイド** の Red Hat Ansible Automation Platform システム要件 を参照してください。

第2章 システム要件

この情報を使用して、Red Hat Ansible Automation Platform のインストールを計画し、ユースケースに適した自動化メッシュトポロジーを設計します。

前提条件

- **sudo** コマンドまたは権限昇格により、root アクセスを取得できる。権限昇格の詳細は、Understanding privilege escalation を参照してください。
- root から、AWX、PostgreSQL、Event-Driven Ansible、Pulp などのユーザーへ権限を降格できる。
- すべてのノードで NTP クライアントを設定している。詳細は、Chrony を使用した NTP サーバーの設定 を参照してください。

2.1. RED HAT 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 ベースシステム

要件	必須	備考
サブスクリプション	有効な Red Hat Ansible Automation Platform	
OS	Red Hat Enterprise Linux 8.8 以降 64 ビット (x86、ppc64le、s390x、aarch64)、または Red Hat Enterprise Linux 9.0 以降 64 ビット (x86、ppc64le、s390x、aarch64)	Red Hat Ansible Automation Platform は OpenShift でもサポートされています。詳細は、Red Hat Ansible Automation Platform Operator を OpenShift Container Platform 上にデプロイする を参照してください。
Ansible-core	Ansible-core バージョン 2.14 以降	Ansible Automation Platform には、ansible-core 2.15 を含む実行環境が含まれています。
Python	3.9 以降	
ブラウザー	Mozilla FireFox または Google Chrome の現行のサポートバージョン	
データベース	PostgreSQL バージョン 13	

プロジェクトの更新およびコレクションを使用するには、以下が必要です。

- ネットワークポートとプロトコル (**表 5.3 Automation Hub** に記載のもの) を使用して正常に接続し、Automation Hub または Ansible Galaxy サーバーからコレクションをダウンロードできることを確認します。
- 自己署名証明書または Red Hat ドメインを使用する場合に SSL インスペクションを無効にします。

注記

Ansible Automation Platform が管理するシステムの要件は Ansible と同じです。Ansible コミュニティードキュメントの Installing Ansible を参照してください。

Red Hat Ansible Automation Platform 要件に関する注意点

- Red Hat Ansible Automation Platform は Ansible Playbook に依存しており、最新の安定したバージョンの ansible-core のインストールが必要です。ansible-core は手動でダウンロードすることも、Red Hat Ansible Automation Platform のインストール中に自動的にダウンロードすることもできます。
- 新規インストールの場合、Automation Controller は ansible-core の最新リリースパッケージをインストールします。
- バンドルの Ansible Automation Platform インストールを実行する場合、インストール setup.sh スクリプトにより、バンドルから ansible-core (およびその依存関係) のインストールが試行されます。
- Ansible を手動でインストールした場合、Ansible Automation Platform インストールの setup.sh スクリプトは、Ansible がインストールされていることを検出し、再インストールを試行しません。

注記

dnf などのパッケージマネージャーを使用して Ansible をインストールする必要があります。また、Red Hat Ansible Automation Platform が正常に動作するには、最新の安定したバージョンのパッケージマネージャーをインストールする必要があります。バージョン 2.4 以降には、Ansible バージョン 2.14 が必要です。

2.2. AUTOMATION CONTROLLER のシステム要件

Automation Controller は分散システムであり、このシステムでは、異なるソフトウェアコンポーネントを同じ場所に配置したり、複数のコンピュートノードにデプロイしたりすることができます。インストーラーでは、ユースケースに適したトポロジーの設計に役立つ抽象化として、コントロールノード、ハイブリッドノード、実行ノード、ホップノードの 4 つのノードタイプが提供されます。

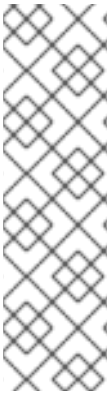
ノードのサイジングには、次の推奨事項を使用してください。

注記

コントロールノードとハイブリッドノードで、実行環境のストレージ用に、最小 20 GB を **/var/lib/awx** に割り当てます。

実行ノード

実行ノードは自動化を実行します。メモリーと CPU を増やし、フォークを多く実行できるように容量を増加します。



注記

- 記載されている RAM および CPU リソースは、実行ノードにインストールするパッケージには必要ない場合がありますが、ノードのジョブ負荷を処理して平均的な数のジョブを同時に実行するために推奨される最小リソースです。
- RAM および CPU ノードの推奨サイズはありません。必要な RAM または CPU は、その環境で実行しているジョブの数に直接依存します。

必要な RAM および CPU レベルの詳細は、Automation Controller のパフォーマンスチューニング を参照してください。

表2.2 実行ノード

要件	最小要件
RAM	16 GB
CPU	4
ローカルディスク	最小 40 GB

コントロールノード

コントロールノードはイベントを処理し、プロジェクトの更新やクリーンアップジョブなどのクラスタージョブを実行します。CPU およびメモリーを増やすと、ジョブイベントの処理に役立ちます。

表2.3 コントロールノード

要件	最小要件
RAM	16 GB
CPU	4
ローカルディスク	• 最小 40 GB。/var/lib/awx で少なくとも 20 GB が使用可能。 • ストレージボリュームは、最小ベースライン 1500 IOPS で評価される必要があります。 • プロジェクトは制御ノードとハイブリッドノードに保存され、ジョブの実行中は実行ノードにも保存されます。クラスターに大規模なプロジェクトが多数ある場合は、ディスク領域のエラーを回避するために、/var/lib/awx/projects に 2 倍の GB を追加することを検討してください。

ホップノード

ホップノードは、自動化メッシュの別の部分にトラフィックをルーティングする役割を果たします (たとえば、ホップノードは別のネットワークへの踏み台ホストにすることができます)。RAM はスループットに影響を与える可能性があり、CPU アクティビティーは低くなります。一般に、ネットワーク帯域幅と遅延は、RAM や CPU よりも重要な要素です。

表2.4 ホップノード

要件	最小要件
RAM	16 GB
CPU	4
ローカルディスク	40 GB

- 実際の RAM 要件は、同時に管理するホストの Automation Controller の数により異なります (これはジョブテンプレートまたはシステムの **ansible.cfg** ファイルの **forks** パラメーターによって制御されます)。リソースの競合を回避するために、Ansible では、10 フォークあたり 1 GB のメモリーと、Automation Controller 用に 2 GB を予約することを推奨しています。詳細は、Automation Controller の容量決定とジョブへの影響 を参照してください。**forks** が 400 に設定されている場合は、42 GB のメモリーが推奨されます。
- Automation Controller ホストは、**umask** が 0022 に設定されているかを確認します。設定されていない場合、セットアップが失敗します。このエラーを回避するには、**umask=0022** を設定します。
- より多くのホストにも対応できますが、フォーク数がホストの総数より少ない場合は、ホスト間でより多くのパスが必要になります。次のいずれかの方法を使用すると、このような RAM の制限を回避できます。
  - ローリング更新を使用します。
  - Automation Controller に組み込まれたプロビジョニングコールバックシステムを使用します。このシステムでは、設定を要求する各システムがキューに登録され、できるだけ早く処理されます。
  - Automation Controller が AMI などのイメージを作成またはデプロイしている場合。

関連情報

- Automation Controller サブスクリプションの取得の詳細は、サブスクリプションのインポート を参照してください。
- ご質問がある場合は、Red Hat カスタマーポータル から Ansible サポートにお問い合わせください。

2.3. AUTOMATION HUB のシステム要件

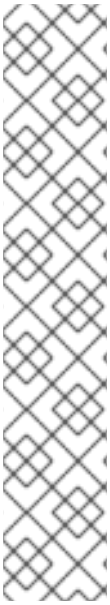
Automation Hub を使用すると、Red Hat Ansible および認定パートナーからの新しい認定自動化コンテンツを見つけて使用できます。Ansible Automation Hub では、クラウド自動化、ネットワーク自動化、セキュリティー自動化などのユースケースのために Red Hat とパートナーによって開発された、サポート対象自動化コンテンツである Ansible コレクションを検出して管理できます。

Automation Hub には、以下のシステム要件があります。

要件	必須	備考
RAM	最小 8 GB	• 8 GB のメモリー (Vagrant 試用版のインストールに推奨される最小要件) • 8 GB メモリー (外部のスタンドアロン PostgreSQL データベースの最小要件) • 設定内のフォークに基づく容量については、Automation Controller の容量決定とジョブへの影響 を参照してください。
CPU	最小 2 つ	設定内のフォークに基づく容量については、Automation Controller の容量決定とジョブへの影響 を参照してください。
ローカルディスク	60 GB ディスク	少なくとも 40 GB をコレクションストレージ用に **/var** に割り当てます。

重要

Ansible Automation 実行ノードと Automation Hub のシステム要件は異なるため、ネットワークのニーズを満たさない可能性があります。必要なメモリー量を決定するための一般的な式は、合計制御容量 = 合計メモリー (MB)/フォークサイズ (MB) です。



注記

Private Automation Hub

内部アドレスから Private Automation Hub をインストールし、外部アドレスしか記載されていない証明書を使用している場合は、インストールして証明書の問題がなくてもコンテナーレジストリーとして使用できなくなる可能性があります。

これを回避するには、**automationhub_main_url** インベントリー変数を使用し、インストールインベントリーファイル内の Private Automation Hub ノードにリンクする値 (https://pah.example.com など) を指定します。

これにより、外部アドレスが **/etc/pulp/settings.py** に追加されます。これは、外部アドレスのみを使用することを意味します。

インベントリーファイル変数の詳細は、**Red Hat Ansible Automation Platform インストールガイド** の インベントリーファイル変数 を参照してください。

2.3.1. 高可用性 Automation Hub の要件

高可用性 (HA) Automation Hub をデプロイする前に、環境に共有ファイルシステムがインストールされていること、および該当する場合はネットワークストレージシステムが設定されていることを確認してください。

2.3.1.1. 必要な共有ファイルシステム

高可用性 Automation Hub では、お使いの環境に、NFS などの共有ファイルシステムを設定しておく必要があります。Red Hat Ansible Automation Platform インストーラーを実行する前に、共有ファイルシステムのインストールの一部としてクラスター全体に **/var/lib/pulp** ディレクトリーをインストールしたことを確認します。いずれかのノードで **/var/lib/pulp** が検出されないと、Red Hat Ansible Automation Platform インストーラーはエラーを返し、高可用性 Automation Hub のセットアップが失敗します。

/var/lib/pulp がいずれかのノードで検出されないことを示すエラーが表示された場合は、**/var/lib/pulp** がすべてのサーバーに正しくマウントされていることを確認し、インストーラーを再実行します。

2.3.1.2. ネットワークストレージ用の firewalld のインストール

Automation Hub ノード自体にネットワークストレージを使用して HA Automation Hub をインストールする場合は、Ansible Automation Platform インストーラーを実行する前に、最初に **firewalld** をインストールして、共有ストレージシステムで必要なポートを開く必要があります。

以下のコマンドを実行して **firewalld** をインストールして設定します。

1. **firewalld** デーモンをインストールします。

```
$ dnf install firewalld
```

2. 以下のコマンドを使用して、<service> にネットワークストレージを追加します。

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

注記

対応しているサービスの一覧は **$ firewall-cmd --get-services** コマンドを使用します。

3. リロードして設定を適用します。

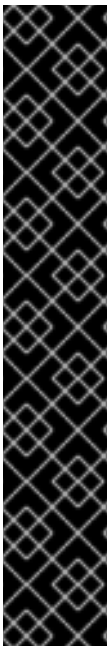
```
$ firewall-cmd --reload
```

2.4. EVENT-DRIVEN ANSIBLE CONTROLLER のシステム要件

Event-Driven Ansible コントローラーは、CPU コアの数に応じて、可変数の長時間実行プロセス (ルールブックアクティベーションなど) をオンデマンドで処理できるシングルノードシステムです。デフォルトで最大 12 個の同時アクティベーションを実行するには、次の最小要件を使用します。

要件	必須
RAM	16 GB

要件	必須
CPU	4
ローカルディスク	最小 40 GB



重要

- Red Hat Enterprise Linux 8 を実行していてメモリー制限を設定したい場合は、Event-Driven Ansible をインストールする前に cgroup v2 を有効にする必要があります。具体的な手順については、Knowledge-Centered Support (KCS) のソリューション記事 Ansible Automation Platform Event-Driven Ansible controller for Red Hat Enterprise Linux 8 requires cgroupv2 を参照してください。
- 標準条件下で Event-Driven Ansible ルールブックをアクティブ化すると、約 250 MB のメモリーが使用されます。ただし、実際のメモリー消費量は、ルールの複雑さと処理されるイベントのボリュームおよびサイズによって大幅に異なる可能性があります。大量のイベントが予想される場合やルールブックの複雑さが高いシナリオでは、ステージング環境でのリソース使用量の事前評価を行います。これにより、アクティベーションの最大数はリソースの容量に基づいて行われます。Event-Driven Ansible コントローラーの最大実行アクティベーションの設定例については、単一 Automation Controller、単一 Automation Hub、および外部 (インストーラー管理) データベースを備えた単一 Event-Driven Ansible Controller ノード を参照してください。

2.5. POSTGRESQL の要件

Red Hat Ansible Automation Platform は PostgreSQL13 を使用します。PostgreSQL ユーザーパスワードは、データベースに保存する前に SCRAM-SHA-256 のセキュアハッシュアルゴリズムでハッシュ化されます。

Automation Controller インスタンスがデータベースにアクセスできるかどうかを確認するには、**awx-manage check_db** コマンドを使用します。

表2.5 データベース

サービス	必須	備考

サービス	必須	備考
データベース	• 20 GB の専用ハードディスクスペース • 4 CPU • 16 GB RAM	• 150 GB 以上を推奨 • ストレージボリュームは、高いベースライン IOPS (1500 以上) に対応するように評価されている必要があります。 • すべての Automation Controller データはデータベースに保存されます。データベースストレージは、マネージドホストの数、ジョブ実行数、ファクトキャッシュに保存されているファクトの数、および個別ジョブのタスク数と共に増加します。たとえば、ホスト 250 台で 1 時間ごと (1 日に 24 回) に 20 個のタスクの Playbook を実行する場合は、毎週 800000 を超えるイベントを保存します。 • データベースに十分な容量が確保されていない場合は、以前のジョブ実行やファクトを定期的に消去する必要があります。詳細は、**Automation Controller 管理ガイド** の 管理ジョブ を参照してください。

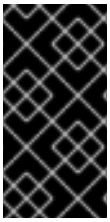
PostgreSQL の設定

必要に応じて、PostgreSQL データベースを、Red Hat Ansible Automation Platform インストーラーで管理されていない個別ノードとして設定できます。Ansible Automation Platform インストーラーがデータベースサーバーを管理する場合は、大半のワークロードで一般的に推奨されているデフォルト値を使用してサーバーを設定します。データベースのパフォーマンスを向上させるために使用できる設定の詳細は、データベース設定 を参照してください。

関連情報

PostgreSQL サーバーのチューニングの詳細は、PostgreSQL のドキュメント を参照してください。

2.5.1. 外部 (お客様がサポートする) データベースの設定



重要

Red Hat は外部 (お客様がサポートする) データベースの使用をサポートしていませんが、外部データベースはお客様によって使用されています。以下の初期設定に関するガイダンスは、関連するサポートリクエストを回避するために、製品インストールの観点からのみ提供されています。

Automation Controller で使用するために、外部の PostgreSQL 準拠データベースにデータベース、ユーザー、およびパスワードを作成するには、次の手順に従います。

手順

1. PostgreSQL 準拠のデータベースサーバーをインストールし、スーパーユーザー権限で接続します。

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

ここでは、以下のようになります。

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

サーバーが実行されているマシンのホスト名を指定します。値がスラッシュで始まる場合、その値は Unix ドメインソケットのディレクトリーとして使用されます。

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

接続するデータベースの名前を指定します。これは、コマンドラインで最初の非オプション引数として **dbname** を指定するのと同等です。**dbname** には接続文字列を指定できます。その場合、接続文字列パラメーターにより、競合するコマンドラインオプションがオーバーライドされます。

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

デフォルトではなく、ユーザー **username** としてデータベースに接続します。(これを行うには権限が必要です。)

2. ユーザーに割り当てられた **createDB** または管理者ロールを使用して、ユーザー、データベース、およびパスワードを作成します。詳細は、Database Roles を参照してください。

3. データベース認証情報とホストの詳細を、外部データベースとして Automation Controller インベントリーファイルに追加します。
次の例ではデフォルト値が使用されています。

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

4. インストーラーを実行します。
Automation Controller で PostgreSQL データベースを使用する場合、データベースは接続ユーザーが所有するものであり、**createDB** または管理者ロールがそのユーザーに割り当てられている必要があります。

5. 作成したデータベースにユーザー名、パスワード、データベース名で接続できることを確認します。

6. ユーザーの権限を確認します。ユーザーには **createDB** または管理者ロールが必要です。

注記

この手順の実行中、外部データベースの範囲を確認する必要があります。詳細は、https://access.redhat.com/articles/4010491 を参照してください。

2.5.2. Automation HubPostgreSQL データベースの hstore 拡張機能の有効化

Ansible Automation Platform 2.4 以降、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するため、Automation Hub PostgreSQL データベースの **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、Automation Hub をインストールする前に、Automation Hub PostreSQL データベースの **hstore** 拡張機能を手動で有効にする必要があります。

Automation Hub のインストール前に **hstore** 拡張機能が有効になっていない場合は、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'"
```

<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)
```

2. 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
```

3. 次のコマンドを使用して、Automation Hub データベースに **hstore** PostgreSQL 拡張機能を作成します。

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

その出力は次のとおりです。

```
CREATE EXTENSION
```

4. 次の出力では、使用されている **hstore** 拡張子が **installed_version** フィールドに含まれており、**hstore** が有効であ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)
```

2.5.3. Ansible Automation Platform PostgreSQL データベースのストレージパフォーマンスのベンチマーク

Flexible I/O Tester (FIO) ツールを使用して、Ansible Automation Platform PostgreSQL データベースの最小要件が満たされているかどうかを確認します。FIO は、ストレージシステムの読み取りおよび書き込み IOPS パフォーマンスをベンチマークするために使用されるツールです。

前提条件

- Flexible I/O Tester (**fio**) ストレージパフォーマンスベンチマークツールがインストールされている。
fio をインストールするには、root ユーザーとして次のコマンドを実行します。

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

- **fio** テストデータログファイルを保存するのに十分なディスク容量がある。
この手順に示す例では、**/tmp** ディレクトリーに少なくとも 60 GB のディスク領域が必要です。
  - **numjobs** は、コマンドによって実行されるジョブの数を設定します。
  - **size=10G** は、各ジョブによって生成されるファイルサイズを設定します。
- **size** パラメーターの値を調整済みである。この値を調整すると、テストデータの量が減ります。

手順

1. ランダムな書き込みテストを実行します。

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

2. ランダムな読み取りテストを実行します。

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

3. 結果を確認します。
   ベンチマークコマンドによって書き込まれたログファイルで、**iops** で始まる行を検索します。この行は、テストの最小値、最大値、および平均値を表示します。

   次の例は、ランダム読み取りテストのログファイル内の行を表示しています。

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

   独自のビジネス要件、アプリケーションのワークロード、および新しい要求に応じて、ログファイルを確認、監視、再検討する必要があります。

第3章 RED HAT ANSIBLE AUTOMATION PLATFORM のインストール

Ansible Automation Platform はモジュール式プラットフォームです。Automation Controller を、Automation Hub や Event-Driven Ansible Controller などの他の Automation Platform コンポーネントとともにデプロイできます。Ansible Automation Platform で提供されるコンポーネントの詳細は、Red Hat Ansible Automation Platform 計画ガイドの Red Hat Ansible Automation Platform コンポーネントを参照してください。

Red Hat Ansible Automation Platform では、サポートされているインストールシナリオがいくつかあります。Red Hat Ansible Automation Platform をインストールするには、インベントリーファイルのパラメーターを編集して、インストールシナリオを指定する必要があります。次のいずれかを独自のインベントリーファイルのベースとして使用できます。

- 外部 (インストーラー管理) データベースを備えた単一の Automation Controller
- 外部 (インストーラー管理) データベースを備えた単一の Automation Controller と単一の Automation Hub
- 単一の Automation Controller、単一の Automation Hub、および外部 (インストーラー管理) データベースを備えた単一の Event-Driven Ansible Controller ノード

3.1. RED HAT ANSIBLE AUTOMATION PLATFORM インストーラーのインベントリーファイルの編集

Red Hat Ansible Automation Platform インストーラーのインベントリーファイルを使用して、インストールシナリオを指定できます。

手順

1. インストーラーに移動します。
   a. [RPM インストールされたパッケージ]

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

   b. [バンドルのインストーラー]

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

   c. [オンラインインストーラー]

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

2. テキストエディターで **inventory** ファイルを開きます。
3. **inventory** ファイルのパラメーターを編集して、インストールシナリオを指定します。サポートされている インストールシナリオの例 の 1 つを **インベントリー** ファイルのベースとして使用できます。

関連情報

- Ansible インストールインベントリーファイルで使用される定義済み変数の包括的なリストについては、インベントリーファイル変数 を参照してください。

3.2. インストールシナリオに基づくインベントリーファイルの例

Red Hat は、Ansible Automation Platform のいくつかのインストールシナリオをサポートしています。サンプルファイルをベースとして使用し、独自のインベントリーファイルを開発することも、希望するインストールシナリオに最も近いサンプルを使用することもできます。

3.2.1. インストールシナリオに基づいたインベントリーファイルの推奨事項

Ansible Automation Platform のインストール方法を選択する前に、次の推奨事項を確認してください。これらの推奨事項をよく理解しておくと、インストールプロセスが効率化されます。

- Red Hat Ansible Automation Platform または自動化ハブの場合: **[automationhub]** グループに自動化ハブホストを追加します。
- 実稼働環境または顧客環境のバージョンの Ansible Automation Platform では、Automation Controller と Automation Hub を同じノードにインストールしないでください。これにより、競合の問題や大量のリソース使用が発生する可能性があります。
- **[automationhub]** および **[automationcontroller]** ホストに到達可能な IP アドレスまたは完全修飾ドメイン名 (FQDN) を指定して、ユーザーが別のノードから Automation Hub のコンテンツを同期してインストールできるようにします。
  FQDN には **-** 記号または _ 記号を含めることはできません。正しく処理されません。

  localhost は使用しないでください。
- **admin** は、Ansible Automation Platform への初回ログイン時のデフォルトのユーザー ID であり、インベントリーファイルで変更することはできません。
- **pg_password** での特殊文字の使用は制限されています。**!**、**#**、**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。
- **registry_username** および **registry_password** に Red Hat Registry Service Account の認証情報を入力し、Red Hat コンテナーレジストリーにリンクします。
- インベントリーファイル変数 **registry_username** および **registry_password** は、非バンドルインストーラーを使用する場合にのみ必要です。

3.2.1.1. 外部 (インストーラー管理) データベースを備えた単一の Automation Controller

この例を使用して、インベントリーファイルに入力し、Red Hat Ansible Automation Platform をインストールします。このインストールインベントリーファイルには、別のノードに外部データベースを持つ単一の Automation Controller ノードが含まれています。

```
[automationcontroller]
controller.example.com

[database]
data.example.com

[all:vars]
admin_password='<password>'
pg_host='data.example.com'
```

```
pg_port=5432
pg_database='awx'
pg_username='awx'
pg_password='<password>'
pg_sslmode='prefer'  # set to 'verify-full' for client-side enforced SSL

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

# SSL-related variables
# If set, this will install a custom CA certificate to the system trust store.
# custom_ca_cert=/path/to/ca.crt
# Certificate and key to install in nginx for the web UI and API
# web_server_ssl_cert=/path/to/tower.cert
# web_server_ssl_key=/path/to/tower.key
# Server-side SSL settings for PostgreSQL (when we are installing it).
# postgres_use_ssl=False
# postgres_ssl_cert=/path/to/pgsql.crt
# postgres_ssl_key=/path/to/pgsql.key
```

3.2.1.2. 外部 (インストーラー管理) データベースを備えた単一の Automation Controller と単一の Automation Hub

この例を使用して、インベントリーファイルにデータを入力し、外部 (インストーラー管理) データベースを備えた Automation Controller と Automation Hub の単一インスタンスをデプロイします。

```
[automationcontroller]
controller.example.com

[automationhub]
automationhub.example.com

[database]
data.example.com

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

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

automationhub_admin_password= <PASSWORD>

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

automationhub_pg_database='automationhub'
```

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

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

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

3.2.1.2.1. Automation Hub の Red Hat Single Sign-On 環境への接続

インベントリーファイルをさらに設定して、Automation Hub を Red Hat Single Sign-On インストールに接続できます。

Ansible Automation Platform が管理する Red Hat Single Sign-On インストールに接続するには、外部の Red Hat Single Sign-On インストールに接続するのではなく、別の変数セットを設定する必要があります。

これらのインベントリー変数の詳細は、Ansible Automation Platform の Central Authentication のインストールと設定 を参照してください。

3.2.1.3. 高可用性 Automation Hub

次の例を使用して、インベントリーファイルを設定し、高可用性 Automation Hub をインストールします。このインベントリーファイルには、クラスター設定を備えた高可用性 Automation Hub が含まれています。

HA デプロイメントをさらに設定して Red Hat Single Sign-On を実装し、SELinux 上で Automation Hub の高可用性デプロイメント を有効にできます。

データベースホスト IP の指定

- **automation_pg_host** および **automation_pg_port** インベントリー変数を使用して、データベースホストの IP アドレスを指定します。以下に例を示します。

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

- また、**automationhub_pg_host** インベントリー変数の値を使用して、[database] セクションでデータベースホストの IP アドレスを指定します。

```
[database]
192.0.2.10
```

クラスター設定のインスタンスのリスト表示

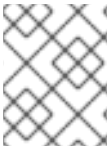
- クラスター設定をインストールする場合、[automationhub] セクションの **localhost ansible_connection=local** は、全インスタンスのホスト名または IP アドレスに置き換えます。以下に例を示します。

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

次のステップ

各 Private Automation Hub サーバーの **/etc/pulp/settings.py** に、次のディレクティブが存在することを確認します。

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

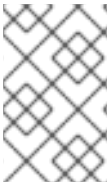


注記

automationhub_main_url を指定しないと、[automationhub] グループの最初のノードがデフォルトで使用されます。

3.2.1.4. SELinux 上での Automation Hub の高可用性 (HA) デプロイメントの有効化

インベントリーファイルを設定して、SELinux 上で Automation Hub の高可用性デプロイメントを有効化できます。**/var/lib/pulp** および **/var/lib/pulp/pulpcore_static** の 2 つのマウントポイントを作成し、それぞれに適切な SELinux コンテキストを割り当てる必要があります。



注記

/var/lib/pulp pulpcore_static のコンテキストを追加し、Ansible Automation Platform インストーラーを実行してから、**/var/lib/pulp** のコンテキストを追加する必要があります。

前提条件

- サーバーに NFS エクスポートを設定している。

手順

1. **/var/lib/pulp** にマウントポイントを作成します。

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

2. テキストエディターを使用して **/etc/fstab** を開き、次の値を追加します。

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

3. systemd マネージャー設定をリロードするコマンドを実行します。

```
$ systemctl daemon-reload
```

4. **/var/lib/pulp** のマウントコマンドを実行します。

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

5. **/var/lib/pulp/pulpcore_static** にマウントポイントを作成します。

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

6. マウントコマンドを実行します。

```
$ mount -a
```

7. マウントポイントを設定したら、Ansible Automation Platform インストーラーを実行します。

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

8. インストールが完了したら、**/var/lib/pulp/** マウントポイントをアンマウントします。

次のステップ

1. 適切な SELinux コンテキストを適用します。
2. pulpcore.service を設定します。

関連情報

- SELinux コンテキストリストは、SELinux Requirements on the Pulp Project documentation を参照してください。
- Pulp フォルダーの詳細は、ファイルシステムレイアウト を参照してください。

3.2.1.4.1. pulpcore.service の設定

インベントリーファイルを設定し、SELinux コンテキストを適用した後、Pulp サービスを設定する必要があります。

手順

1. 2 つのマウントポイントを設定したら、Pulp サービスをシャットダウンして **pulpcore.service** を設定します。

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

2. **systemctl** を使用して **pulpcore.service** を編集します。

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

3. 以下のエントリーを **pulpcore.service** に追加し、ネットワークを起動し、リモートマウントポイントをマウントすることで、Automation Hub サービスが起動するようにします。

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

4. **remote-fs.target** を有効にします。

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

5. システムを再起動します。

```
$ systemctl reboot
```

トラブルシューティング

pulpcore SELinux ポリシーのバグにより、**etc/pulp/certs/** のトークン認証公開鍵/秘密鍵に適切な SELinux ラベルがなく、パルププロセスが失敗する可能性があります。これが発生した場合は、次のコマンドを実行して、適切なラベルを一時的に貼り付けます。

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

システムのラベルを変更するたびに、このコマンドを繰り返して適切な SELinux ラベルを再割り当てします。

3.2.1.4.2. SELinux コンテキストの適用

インベントリーファイルを設定したら、コンテキストを適用して、SELinux 上で Automation Hub の高可用性 (HA) デプロイメントを有効化する必要があります。

手順

1. Pulp サービスをシャットダウンします。

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

2. **/var/lib/pulp/pulpcore_static** をアンマウントします。

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

3. **/var/lib/pulp/** をアンマウントします。

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

4. テキストエディターで **/etc/fstab** を開き、**/var/lib/pulp** の既存値を以下に置き換えます。

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

5. マウントコマンドを実行します。

```
$ mount -a
```

3.2.1.5. Private Automation Hub でのコンテンツ署名の設定

Ansible Certified Content Collections に正常に署名して公開するには、署名する Private Automation Hub を設定する必要があります。

前提条件

- GnuPG キーペアがセキュアに設定され、組織で管理されている。
- 公開鍵と秘密鍵のペアに、Private Automation Hub でコンテンツ署名を設定するのに適切なアクセス権がある。

手順

1. ファイル名のみを受け入れる署名スクリプトを作成します。

注記

このスクリプトは署名サービスとして機能し、**PULP_SIGNING_KEY_FINGERPRINT** 環境変数で指定された鍵を使用して、そのファイルの ASCII アーマー形式の **gpg** デタッチ署名を生成する必要があります。

スクリプトは、次の形式で JSON 構造を出力します。

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

すべてのファイル名は、現在の作業ディレクトリー内の相対パスです。ファイル名は、デタッチ署名でも同じにする必要があります。

以下に例を示します。

次のスクリプトはコンテンツの署名を生成します。

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

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

ADMIN_ID="$PULP_SIGNING_KEY_FINGERPRINT"
PASSWORD="password"

# Create a detached signature
gpg --quiet --batch --pinentry-mode loopback --yes --passphrase \
```

```
  $PASSWORD --homedir ~/.gnupg/ --detach-sign --default-key $ADMIN_ID \
  --armor --output $SIGNATURE_PATH $FILE_PATH

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

署名を有効にして Private Automation Hub を Ansible Automation Platform クラスターにデプロイすると、新しい UI が追加されたことがコレクションに表示されます。

2. **automationhub_*** で始まるオプションについては、Ansible Automation Platform インストーラーのインベントリーファイルを確認してください。

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

2 つの新しいキー (**automationhub_auto_sign_collections** および **automationhub_require_content_approval**) は、コレクションが Private Automation Hub にアップロードされた後に署名および承認される必要があることを示します。

3.2.1.6. Private Automation Hub での LDAP 設定

LDAP 認証用に Private Automation Hub を設定するには、Red Hat Ansible Automation Platform インストーラーインベントリーファイルで次の 6 つの変数を設定する必要があります。

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

これらの変数のいずれかが欠落している場合、Ansible Automation インストーラーはインストールを完了できません。

3.2.1.6.1. インベントリーファイル変数の設定

LDAP 認証を使用して Private Automation Hub を設定する場合は、インストールプロセス中にインベントリーファイルに適切な変数を設定する必要があります。

手順

1. Red Hat Ansible Automation Platform インストーラーインベントリーファイルの編集 の手順に従って、インベントリーファイルにアクセスします。
2. 次の例をガイドとして使用して、Ansible Automation Platform インベントリーファイルを設定します。

```
automationhub_authentication_backend = "ldap"

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

注記

次の変数は、他のオプションで設定しない限り、デフォルト値で設定されます。

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

3. オプション: ユーザーグループ、スーパーユーザーアクセス、ミラーリングなどの追加パラメーターを Private Automation Hub にセットアップします。このオプションの手順を完了するには、追加の LDAP パラメーターの設定 に進みます。

3.2.1.6.2. 追加の LDAP パラメーターの設定

スーパーユーザーアクセス、ユーザーグループ、ミラーリング、またはその他の追加パラメーターを設定する予定がある場合は、それらを含む YAML ファイルを **ldap_extra_settings** ディクショナリー内に作成できます。

手順

1. **ldap_extra_settings** を含む YAML ファイルを作成します。
   - 以下に例を示します。

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

...
```

2. セットアップに必要なパラメーターを追加します。次の例では、**ldap_extra_settings** で設定できる LDAP パラメーターについて説明します。

- この例を使用して、LDAP グループのメンバーシップに基づいてスーパーユーザーフラグを設定します。

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

- この例を使用して、スーパーユーザーアクセスを設定します。

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

- この例を使用して、所属するすべての LDAP グループをミラーリングします。

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

- この例を使用して、LDAP ユーザー属性 (ユーザーの名、姓、電子メールアドレスなど) をマップします。

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

- LDAP グループのメンバーシップに基づいて、アクセスを許可または拒否するには、次の例を使用します。
  - Private Automation Hub アクセスを許可する (たとえば、**cn=pah-nosoupforyou,ou=groups,dc=example,dc=com** グループのメンバー) には、以下を実行します。

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

  - Private Automation Hub アクセスを拒否する (たとえば、**cn=pah-nosoupforyou,ou=groups,dc=example,dc=com** グループのメンバー) には、以下を実行します。

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

- この例を使用して、LDAP デバッグログを有効にします。

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

注記

setup.sh を再実行することが現実的でない場合、またはデバッグログが短期間有効になっている場合は、Private Automation Hub の **/etc/pulp/settings.py** ファイルに **GALAXY_LDAP_LOGGING: True** を含む行を手動で追加できます。変更を有効にするには、**pulpcore-api.service** と **nginx.service** の両方を再起動します。人的ミスによる失敗を避けるため、この方法は必要な場合にのみ使用してください。

- この例では、変数 **AUTH_LDAP_CACHE_TIMEOUT** を設定して LDAP キャッシュを設定します。

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

3. Private Automation Hub のインストール中に **setup.sh -e @ldapextras.yml** を実行します。検証: 正しくセットアップされていることを検証するには、Private Automation Hub の **/etc/pulp/settings.py** ファイル内のすべての設定を表示できることを確認します。

3.2.1.6.3. LDAP 参照

LDAP サーバーが参照を返す場合、Private Automation Hub で LDAP を使用して正常に認証するには、参照を無効にする必要がある場合があります。

返さない場合は、次のメッセージが返されます。

```
Operation unavailable without authentication
```

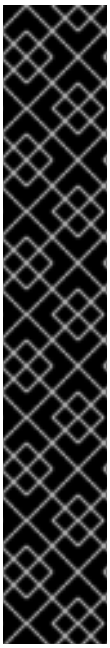
LDAP REFERRALS ルックアップを無効にするには、次のように設定します。

```
GALAXY_LDAP_DISABLE_REFERRALS = true
```

これにより、**AUTH_LDAP_CONNECTIONS_OPTIONS** が正しいオプションに設定されます。

3.2.1.7. 単一の Automation Controller、単一の Automation Hub、および外部 (インストーラー管理) データベースを備えた単一の Event-Driven Ansible Controller ノード

この例を使用して、インベントリーファイルにデータを入力し、外部 (インストーラー管理) データベースを備えた Automation Controller、Automation Hub、および Event-Driven Ansible Controller の単一インスタンスをデプロイします。



重要

- このシナリオでは、Event-Driven Ansible Controller を正常にデプロイするために、Automation Controller 2.4 以降が必要です。
- Event-Driven Ansible Controller は別のサーバーにインストールする必要があります。Automation Hub および Automation Controller と同じホストにインストールすることはできません。
- 標準条件下で Event-Driven Ansible ルールブックをアクティブ化すると、約 250 MB のメモリーが使用されます。ただし、実際のメモリー消費量は、ルールの複雑さと処理されるイベントのボリュームおよびサイズによって大幅に異なる可能性があります。大量のイベントが予想される場合やルールブックの複雑さが高いシナリオでは、ステージング環境でのリソース使用量の事前評価を行います。これにより、アクティベーションの最大数はリソースの容量に基づいて行われます。次の例では、デフォルトの **automationedacontroller_max_running_activations** 設定は 12 ですが、容量に応じて調整できます。

```
[automationcontroller]
controller.example.com

[automationhub]
automationhub.example.com

[automationedacontroller]
automationedacontroller.example.com

[database]
data.example.com

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

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

# Automation hub configuration

automationhub_admin_password= <PASSWORD>
```

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

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

# Automation Event-Driven Ansible controller configuration

automationedacontroller_admin_password='<eda-password>'

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

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

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

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

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

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

# Boolean flag used to verify Automation Controller's
# web certificates when making calls from Automation Event-Driven Ansible controller.
# automationedacontroller_controller_verify_ssl = true
#
```

```
# Certificate and key to install in Automation Event-Driven Ansible controller node
# automationedacontroller_ssl_cert=/path/to/automationeda.crt
# automationedacontroller_ssl_key=/path/to/automationeda.key
```

3.3. RED HAT ANSIBLE AUTOMATION PLATFORM インストーラー設定スクリプトの実行

Private Automation Hub をインストールするために必要なパラメーターでインベントリーファイルを更新した後、インストーラーのセットアップスクリプトを実行します。

手順

- **setup.sh** スクリプトを実行します。

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

Red Hat Ansible Automation Platform のインストールが開始します。

3.4. AUTOMATION CONTROLLER のインストールの検証

inventory ファイルに挿入した管理者認証情報でログインして、Automation Controller が正常にインストールされたことを確認します。

前提条件

- ポート 443 が利用可能である。

手順

1. **inventory** ファイルの Automation Controller ノードに指定した IP アドレスに移動します。
2. Red Hat Satellite の認証情報を入力します。インストール後に初めてログインする場合は、**manifest** ファイルをアップロードします。
3. ユーザー ID **admin** と、**インベントリー** ファイルに設定したパスワード認証情報を使用してログインします。

注記

Automation Controller サーバーにはポート 80 (**https://<CONTROLLER_SERVER_NAME>/**) からアクセスできますが、ポート 443 にリダイレクトされます。

重要

インストールに失敗し、Red Hat Ansible Automation Platform の有効なライセンスを購入済みのお客様は、Red Hat カスタマーポータル から Ansible までお問い合せください。

Automation Controller へのログインに成功すれば、Red Hat Ansible Automation Platform 2.4 のインストールは完了です。

3.4.1. 追加の Automation Controller の設定とリソース

追加の Automation Controller の設定については、次のリソースを参照してください。

表3.1 Automation Controller を設定するための資料

リソースリンク	説明
Automation Controller クイックセットアップガイド	Automation Controller を設定して最初の Playbook を実行します。
Automation Controller 管理ガイド	カスタマースクリプト、管理ジョブなどを使用して Automation Controller の管理を設定します。
Red Hat Ansible Automation Platform のプロキシーサポートの設定	プロキシーサーバーを使用して Automation Controller を設定します。
ユーザビリティーアナリティクスおよび Automation Controller からのデータ収集の管理	Red Hat と共有する Automation Controller の情報を管理します。
Automation Controller ユーザーガイド	Automation Controller の機能をより詳細に確認します。

3.5. AUTOMATION HUB のインストールの検証

inventory ファイルに挿入した管理者認証情報でログインして、Automation Hub が正常にインストールされたことを確認します。

手順

1. **inventory** ファイルの自動化ハブノードに指定した IP アドレスに移動します。
2. Red Hat Satellite の認証情報を入力します。インストール後に初めてログインする場合は、**manifest** ファイルをアップロードします。
3. ユーザー ID **admin** と、**インベントリー** ファイルに設定したパスワード認証情報を使用してログインします。

重要

インストールに失敗し、Red Hat Ansible Automation Platform の有効なライセンスを購入済みのお客様は、Red Hat カスタマーポータル から Ansible までお問い合せください。

Automation Hub へのログインに成功すれば、Red Hat Ansible Automation Platform 2.4 のインストールは完了です。

3.5.1. 追加の Automation Hub の設定とリソース

追加の Automation Hub 設定については、以下のリソースを参照してください。

表3.2 Automation Controller を設定するための資料

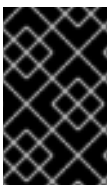
リソースリンク	説明
Private Automation Hub でのユーザーアクセスの管理	Automation Hub のユーザーアクセスを設定します。
Automation Hub での Red Hat 認定済み、検証済み、および Ansible Galaxy コンテンツの管理	Automation Hub にコンテンツを追加します。
Automation Hub でのプロプライエタリーコンテンツコレクションの公開	社内で開発したコレクションを Automation Hub に公開します。

3.6. EVENT-DRIVEN ANSIBLE CONTROLLER のインストールの検証

inventory ファイルに挿入した管理者認証情報でログインし、Event-Driven Ansible Controller が正常にインストールされたことを確認します。

手順

1. **inventory** ファイル内の Event-Driven Ansible Controller ノードに指定された IP アドレスに移動します。
2. Red Hat Satellite の認証情報を入力します。インストール後に初めてログインする場合は、**manifest** ファイルをアップロードします。
3. ユーザー ID **admin** と、**インベントリー** ファイルに設定したパスワード認証情報を使用してログインします。



重要

インストールに失敗し、Red Hat Ansible Automation Platform の有効なライセンスを購入済みのお客様は、Red Hat カスタマーポータル から Ansible までお問い合せください。

Event-Driven Automation Controller へのログインに成功すれば、Red Hat Ansible Automation Platform 2.4 のインストールは完了です。

第4章 非接続インストール

インターネットに接続していない場合、またはオンラインリポジトリーにアクセスできない場合、アクティブなインターネット接続がなくても Red Hat Ansible Automation Platform をインストールできます。

4.1. 前提条件

非接続ネットワークに Ansible Automation Platform をインストールするには、次の前提条件を満たす必要があります。

1. 作成済みのサブスクリプションマニフェスト。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。
2. カスタマーポータル の Ansible Automation Platform セットアップバンドルをダウンロードしている。
3. Automation Controller と Private Automation Hub サーバーの DNS レコード を作成している。

4.2. 非接続の RHEL への ANSIBLE AUTOMATION PLATFORM のインストール

Automation Controller 上にあるインストーラー管理のデータベースを使用すると、インターネット接続なしで Ansible Automation Platform Automation Controller と Private Automation Hub をインストールできます。非接続インストールには、セットアップバンドルを使用します。これには、非接続環境での Ansible Automation Platform のインストールを容易にする追加のコンポーネントが含まれているためです。コンポーネントには、Ansible Automation Platform Red Hat パッケージマネージャー (RPM) とデフォルトの実行環境 (EE) イメージが含まれます。

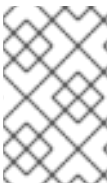
4.2.1. 非接続インストールのシステム要件

Ansible Automation Platform の非接続インストールを実行する前に、システムがすべてのハードウェア要件を満たしていることを確認してください。ハードウェア要件の詳細は、第 2 章 システム要件 を参照してください。

4.2.2. RPM ソース

BaseOS および AppStream リポジトリーからの Ansible Automation Platform の RPM 依存関係は、セットアップバンドルには含まれません。これらの依存関係を追加するには、まず BaseOS および AppStream リポジトリーへのアクセスを取得する必要があります。Satellite を使用してリポジトリーを同期し、依存関係を追加します。別のツールを使用する場合は、以下の中から選択できます。

- Reposync
- RHEL バイナリー DVD



注記

RHEL バイナリー DVD 方式では、バージョン 8.6 以降を含む、サポートされているバージョンの RHEL の DVD が必要です。現在サポートされている RHEL のバージョンに関する詳細は、Red Hat Enterprise Linux のライフサイクル を参照してください。

関連情報

- Satellite

4.3. REPOSYNC を使用した RPM リポジトリーの同期

reposync を実行するには、インターネットにアクセスできる RHEL ホストが必要です。リポジトリーが同期されたら、Web サーバーからホストされている非接続ネットワークにリポジトリーを移動できます。

手順

1. BaseOS と AppStream の必要なリポジトリーをアタッチします。

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

2. reposync を実行します。

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

   a. **--download-metadata** を指定し、**--newest-only** を指定せずに reposync を使用します。RHEL 8 Reposync を参照してください。

      - **--newest-only** を使用しない場合、ダウンロードされるリポジトリーは最大 90 GB になります。
      - **--newest-only** を使用する場合、ダウンロードされるリポジトリーは最大 14 GB になります。

3. Red Hat Single Sign-On を使用する予定がある場合は、次のリポジトリーを同期します。

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

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

   reposync が完了すると、リポジトリーを Web サーバーで使用できるようになります。

4. リポジトリーを非接続ネットワークに移動します。

4.4. リポジトリーをホストする新しい WEB サーバーの作成

リポジトリーをホストする既存の Web サーバーがない場合は、同期されたリポジトリーを使用して Web サーバーを作成できます。

手順

1. 前提条件をインストールします。

```
$ sudo dnf install httpd
```

2. リポジトリーディレクトリーを提供するように httpd を設定します。

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

DocumentRoot '/path/to/repos'

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

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

3. ディレクトリーが apache ユーザーによって読み取り可能であることを確認してください。

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

4. SELinux を設定します。

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

5. httpd を有効にします。

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

6. ファイアウォールを開きます。

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

7. Automation Controller と Automation Hub で、**/etc/yum.repos.d/local.repo** にリポジトリーファイルを追加し、必要に応じてオプションのリポジトリーを追加します。

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

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

4.5. ローカルにマウントした DVD からの RPM リポジトリーへのアクセス

RHEL バイナリー DVD からリポジトリーにアクセスする場合は、最初にローカルリポジトリーを設定する必要があります。

手順

1. DVD または ISO をマウントします。
   a. DVD

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

   b. ISO

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

2. **/etc/yum.repos.d/dvd.repo** に yum リポジトリーファイルを作成します。

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

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

3. gpg キーをエクスポートします。

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

注記

キーがインポートされていない場合は、次のようなエラーが表示されます。

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

関連情報

リポジトリーの設定の詳細は、Need to set up yum repository for locally-mounted DVD on Red Hat Enterprise Linux 8 を参照してください。

4.6. インターネット接続なしで ANSIBLE AUTOMATION PLATFORM にサブスクリプションマニフェストを追加する

インターネット接続なしで Ansible Automation Platform にサブスクリプションを追加するには、サブスクリプションマニフェストを作成してインポートします。

手順

1. Red Hat カスタマーポータル にログインします。
2. メニューバーから **Subscriptions** を選択し、**Subscriptions Allocations** タブを選択します。
3. **New Subscription Allocation** をクリックします。
4. 新しいサブスクリプション割り当てに名前を付けます。
5. **Type** の一覧から **Satellite 6.8** を選択します。
6. **Create** をクリックします。サブスクリプション割り当ての **Details** タブが開きます。
7. **Subscriptions** タブを選択します。
8. **Add Subscriptions** をクリックします。
9. Ansible Automation Platform サブスクリプションを見つけて、**Entitlements** ボックスに、環境に割り当てるエンタイトルメントの数を **追加** します。Ansible Automation Platform によって管理されるノード (サーバー、ネットワークデバイスなど) ごとに 1 つのエンタイトルメントが必要です。
10. **Submit** をクリックします。
11. **Export Manifest** をクリックします。

インストール後に Automation Controller でインポートされるファイル **manifest_<allocation name>_<date>.zip** がダウンロードされます。

4.7. ANSIBLE AUTOMATION PLATFORM セットアップバンドルのダウンロードとインストール

セットアップバンドルを選択して、非接続インストール用の Ansible Automation Platform をダウンロードします。このバンドルには、Ansible Automation Platform の RPM コンテンツと、インストールプロセス中に Private Automation Hub にアップロードされるデフォルトの実行環境イメージが含まれています。

手順

1. Red Hat Ansible Automation Platform のダウンロード ページに移動し、Ansible Automation Platform 2.4 セットアップバンドルの **Download Now** をクリックして、Ansible Automation Platform Setup Bundle パッケージをダウンロードします。
2. Automation Controller から、バンドルを展開します。

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

3. インベントリーファイルを編集して、必要なオプションを含めます。
   a. automationcontroller group

b. automationhub group

c. admin_password

d. pg_password

e. automationhub_admin_password

f. automationhub_pg_host, automationhub_pg_port

g. automationhub_pg_password
インベントリーファイルの例

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

[automationcontroller:vars]
peers=execution_nodes

[automationhub]
automationhub.example.org

[all:vars]
admin_password='password123'

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

receptor_listener_port=27199

automationhub_admin_password='hubpassword123'

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

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

4. Ansible Automation Platform セットアップバンドルの実行可能ファイルを root ユーザーとして実行します。

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

5. インストールが完了したら、インストールインベントリーファイルで指定された Automation Controller ノードの完全修飾ドメイン名 (FQDN) に移動します。

6. インストールインベントリーファイルで指定した管理者の認証情報を使用してログインします。

注記

インベントリーファイルはバックアップ、復元、アップグレード機能に使用されるため、インストール後もそのままの状態で保存する必要があります。インベントリーファイルにパスワードが含まれている場合は、バックアップコピーをセキュアな場所に保管してください。

4.8. インストール後のタスクの完了

Ansible Automation Platform のインストールが完了したら、Automation Hub と Automation Controller が適切にデプロイされていることを確認します。

4.8.1. Controller のサブスクリプションの追加

手順

1. Automation Controller の FQDN に移動します。ユーザー名 admin と、インベントリーファイルで **admin_password** として指定したパスワードを使用してログインします。
2. **Browse** をクリックして、作成済みの **manifest.zip** を選択します。
3. **Next** をクリックします。
4. **User analytics** と **Automation analytics** のチェックを外します。これらはインターネット接続に依存しているため、オフにする必要があります。
5. **Next** をクリックします。
6. 使用許諾契約書を読み、同意する場合は **Submit** をクリックします。

4.8.2. CA トラストストアの更新

インストール後のタスクの一環として、ソフトウェアの証明書を更新する必要があります。デフォルトでは、Ansible Automation Platform Automation Hub と Automation Controller は自己署名証明書を使用してインストールされます。そのため、Controller は Hub の証明書を信頼せず、Hub から実行環境をダウンロードしません。

Automation Hub から Automation Controller に実行環境をダウンロードするには、Hub の認証局 (CA) 証明書を信頼できる証明書として Controller にインポートする必要があります。これは、Automation Controller と Private Automation Hub の間で root ユーザーとして SSH を使用できるかどうかに応じて、2 つの方法のいずれかで実行できます。

4.8.2.1. root ユーザーとしてセキュアコピー (SCP) を使用する

Controller と Private Automation Hub の間で root ユーザーとして SSH を使用できる場合は、SCP を使用して Private Automation Hub 上のルート証明書を Controller にコピーします。

手順

1. Controller で **update-ca-trust** を実行して、CA トラストストアを更新します。

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

```
# update-ca-trust
```

4.8.2.2. root 以外のユーザーとしてコピーして貼り付ける

Private Automation Hub と Controller の間で root として SSH を使用できない場合は、Private Automation Hub 上のファイル **/etc/pulp/certs/root.crt** の内容をコピーし、Controller 上にある **/etc/pki/ca-trust/source/anchors/automationhub-root.crt** という名前の新しいファイルにペーストします。

手順

1. **update-ca-trust** を実行して、新しい証明書で CA トラストストアを更新します。Private Automation Hub で、次を実行します。

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

1. Automation Controller で、次を実行します。

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

関連情報

- 不明な認証局エラーの詳細は、Project sync fails with unknown certificate authority error in Ansible Automation Platform 2.1 を参照してください。

4.9. PRIVATE AUTOMATION HUB へのコレクションのインポート

Private Automation Hub で使用するために、Ansible Automation Hub からコレクションを tarball ファイルとしてダウンロードできます。認定コレクションは Automation Hub Hybrid Cloud Console で、コミュニティーコレクションは Ansible Galaxy で提供されています。コレクションに必要な依存関係もダウンロードしてインストールする必要があります。

手順

1. console.redhat.com に移動し、Red Hat 認証情報を使用してログインします。
2. ダウンロードする **コレクション** をクリックします。
3. **Download tarball** クリックします。
4. コレクションに依存関係があるかどうかを確認するには、**Dependencies** タブをクリックします。
5. このコレクションに必要な依存関係をダウンロードします。

4.10. コレクション名前空間の作成

コレクションをインポートする前に、まず Private Automation Hub にコレクションの名前空間を作成する必要があります。名前空間の名前は、コレクションの tarball ファイル名の最初にあります。たとえば、コレクション **ansible-netcommon-3.0.0.tar.gz** の名前空間は **ansible** です。

手順

1. Automation Hub Hybrid Cloud Console にログインします。
2. ナビゲーションパネルから、**Collections → Namespaces** を選択します。
3. **Create** をクリックします。
4. 名前空間の名前を指定します。
5. **Create** をクリックします。

4.10.1. Web コンソールを使用したコレクション tarball のインポート

名前空間を作成したら、Web コンソールを使用してコレクションをインポートできます。

手順

1. Automation Hub Hybrid Cloud Console にログインします。
2. ナビゲーションパネルから、**Collections → Namespaces** を選択します。
3. コレクションをインポートする名前空間の横にある **View collections** をクリックします。
4. **Upload collection** をクリックします。
5. **フォルダーアイコン** をクリックし、コレクションの tarball を選択します。
6. **Upload** をクリックします。

これにより、'My Imports' ページが開きます。インポートのステータスと、インポートされたファイルとモジュールのさまざまな詳細を確認できます。

4.10.2. CLI を使用したコレクション tarball のインポート

GUI ではなくコマンドラインインターフェイスを使用して、コレクションを Private Automation Hub にインポートできます。

手順

1. コレクションの tarball を Private Automation Hub にコピーします。
2. SSH 経由で Private Automation Hub サーバーにログインします。
3. 自己署名ルート CA 証明書を Automation Hub のトラストストアに追加します。

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

4. **/etc/ansible/ansible.cfg** ファイルを Automation Hub の設定で更新します。認証には、トークンまたはユーザー名とパスワードのいずれかを使用します。

```
[galaxy]
server_list = private_hub

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

5. ansible-galaxy コマンドを使用してコレクションをインポートします。

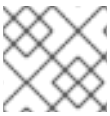
```
$ ansible-galaxy collection publish <collection_tarball>
```

4.11. インポートしたコレクションの承認

GUI または CLI のいずれかでコレクションをインポートした後、GUI を使用してコレクションを承認する必要があります。承認後、使用可能になります。

手順

1. Automation Hub Hybrid Cloud Console にログインします。
2. ナビゲーションパネルから、**Collections → Approval** を選択します。
3. 承認するコレクションの **Approve** をクリックします。
4. このコレクションが Private Automation Hub で使用できるようになります。
5. ステップ 2 と 3 を繰り返して、コレクションの依存関係をインポートします。



注記

コレクションは、ソースに関係なく「公開済み」リポジトリーに追加されます。

推奨されるコレクションは、ユースケースによって異なります。Ansible と Red Hat は こちらのコレクション を提供しています。

4.11.1. カスタム自動化実行環境

ansible-builder プログラムを使用して、カスタム実行環境イメージを作成します。非接続環境の場合、カスタム実行環境イメージは次の方法でビルドできます。

- インターネットに接続されたシステム上で実行環境イメージをビルドし、それを非接続環境にインポートします。
- ansible-builder を使用する通常のプロセスにいくつかの変更を加え、実行環境イメージを完全に非接続環境でビルドします。
- 非接続環境に必要なすべての変更を含む最小限のベースコンテナーイメージを作成し、ベースコンテナーイメージからカスタム実行環境イメージをビルドします。

4.11.1.1. 非接続環境の境界を越えたカスタム実行環境イメージの転送

カスタム実行環境イメージは、インターネットに接続されたマシン上でビルドできます。実行環境を作成すると、ローカルの podman イメージキャッシュでその実行環境を使用できるようになります。このカスタム実行環境イメージを、非接続環境の境界を越えて転送することができます。

手順

1. イメージを保存します。

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

   sneakernet、one-way diode などの既存メカニズムを使用して、非接続環境の境界を越えてファイルを転送します。

2. 非接続側でイメージが使用可能になったら、それをローカルの podman キャッシュにインポートし、タグを付けて非接続ハブにプッシュします。

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

4.12. 非接続環境での実行環境の構築

Ansible Automation Platform の 実行環境の作成 は一般的なタスクですが、非接続環境では動作が異なります。カスタム実行環境を構築する際に、ansible-builder ツールはデフォルトで、インターネットの以下の場所からコンテンツをダウンロードします。

- 実行環境イメージに追加する Ansible コンテンツコレクションの場合は、Red Hat Automation Hub (console.redhat.com) または Ansible Galaxy (galaxy.ansible.com)。
- コレクションの依存関係として必要なすべての Python パッケージについては、PyPI (pypi.org)。
- 必要に応じて、実行環境イメージに RPM を追加または更新する場合は、RPM リポジトリー (RHEL または UBI リポジトリー (cdn.redhat.com) など)。
- ベースコンテナーイメージへのアクセスは、registry.redhat.io。

非接続環境で実行環境イメージをビルドするには、これらの場所からコンテンツをミラーリングする必要があります。Ansible Galaxy または Automation Hub から Private Automation Hub へのコレクションのインポートに関する詳細は、Private Automation Hub へのコレクションのインポート を参照してください。

非接続ネットワークに転送されたミラーリングされた PyPI コンテンツは、Web サーバーまたは Nexus などのアーティファクトリポジトリーを使用して利用可能になります。RHEL および UBI リポジトリーのコンテンツは、インターネットに接続された Red Hat Satellite Server からエクスポートし、非接続環境にコピーして、非接続の Satellite にインポートできます。これにより、カスタム実行環境のビルドに使用できます。詳細は、エアギャップシナリオでの ISS エクスポート同期 を参照してください。

デフォルトのベースコンテナーイメージである ee-minimal-rhel8 は、カスタム実行環境イメージの作成に使用されます。これは、バンドルのインストーラーに含まれます。このイメージは、インストール時に Private Automation Hub に追加されます。ee-minimal-rhel9 などの別のベースコンテナーイメージが必要な場合は、それを非接続ネットワークにインポートし、Private Automation Hub コンテナーレジストリーに追加する必要があります。

非接続ネットワーク上ですべての前提条件が利用可能になったら、ansible-builder コマンドを使用してカスタム実行環境イメージを作成できます。

4.12.1. Ansible Builder RPM のインストール

カスタム実行環境がビルドされる RHEL システムでは、環境内の既存の Satellite Server を使用して Ansible Builder RPM をインストールします。必要に応じて、実行環境イメージで既存の Satellite の RHEL コンテンツを使用できるため、この方法が推奨されます。

手順

1. Ansible Automation Platform リポジトリーから Ansible Builder RPM をインストールします。
   a. 非接続ネットワーク上の Satellite に RHEL システムをサブスクライブします。
   b. Ansible Automation Platform サブスクリプションを割り当て、Ansible Automation Platform リポジトリーを有効にします。リポジトリー名は、基礎となるシステムで使用している RHEL のバージョンに応じて、**ansible-automation-platform-2.4-for-rhel-8-x86_64-rpms** または **ansible-automation-platform-2.4-for-rhel-9-x86_64-rpms** になります。
   c. Ansible Builder RPM をインストールします。以下の例が正しく動作するには、Ansible Builder RPM のバージョンが 3.0.0 以降である必要があります。
2. Ansible Automation Platform セットアップバンドルから Ansible Builder RPM をインストールします。この方法は、非接続ネットワークで Satellite Server が利用できない場合に使用してください。
   a. Ansible Automation Platform セットアップバンドルを展開します。
   b. 含まれているコンテンツから Ansible Builder RPM とその依存関係をインストールします。

```
$ tar -xzvf ansible-automation-platform-setup-bundle-2.4-3-x86_64.tar.gz
$ cd ansible-automation-platform-setup-bundle-2.4-3-x86_64/bundle/packages/el8/repos/
$ sudo dnf install ansible-builder-3.0.0-2.el8ap.noarch.rpm \
    python39-requirements-parser-0.2.0-4.el8ap.noarch.rpm \
    python39-bindep-2.10.2-3.el8ap.noarch.rpm \
    python39-jsonschema-4.16.0-1.el8ap.noarch.rpm \
    python39-pbr-5.8.1-2.el8ap.noarch.rpm \
    python39-distro-1.6.0-3.el8pc.noarch.rpm \
    python39-packaging-21.3-2.el8ap.noarch.rpm \
    python39-parsley-1.3-2.el8pc.noarch.rpm \
    python39-attrs-21.4.0-2.el8pc.noarch.rpm \
    python39-pyrsistent-0.18.1-2.el8ap.x86_64.rpm \
    python39-pyparsing-3.0.9-1.el8ap.noarch.rpm
```

注記

具体的なバージョンは、使用しているセットアップバンドルのバージョンに応じて若干異なる場合があります。

関連情報

- 非接続ネットワークで Satellite 環境を作成する方法の詳細は、オフラインネットワーク環境での Satellite Server のインストール を参照してください。

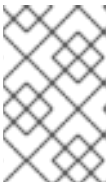
4.12.2. カスタム実行環境定義の作成

Ansible Builder RPM をインストールしたら、次の手順を使用してカスタム実行環境を作成します。

1. カスタム実行環境の作成時に使用するビルドアーティファクト用のディレクトリーを作成します。以下の手順で作成された新しいファイルはすべて、このディレクトリーに作成されます。

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

2. カスタム実行環境の要件を定義する **execution-environment.yml** ファイルを作成します。



注記

実行環境定義形式のバージョン 3 が必要です。そのため、続行する前に、**execution-environment.yml** ファイルに **version: 3** が明示的に含まれていることを確認してください。

a. Private Automation Hub で利用可能な最小実行環境を指すように、ベースイメージをオーバーライドします。

b. ビルドプロセスで使用する非接続のコンテンツソースを指すために必要な、追加のビルドファイルを定義します。カスタムの **execution-environment.yml** ファイルは次の例のようになります。

```
$ cat execution-environment.yml
---
version: 3

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

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

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

additional_build_steps:
  prepend_base:
    # copy a custom pip.conf to override the location of the PyPI content
    - ADD _build/configs/pip.conf /etc/pip.conf
    # remove the default UBI repository definition
    - RUN rm -f /etc/yum.repos.d/ubi.repo
    # copy the hub CA certificate and update the trust store
```

```
    - ADD _build/configs/hub-ca.crt /etc/pki/ca-trust/source/anchors
    - RUN update-ca-trust
    # if needed, uncomment to add a custom RPM repository configuration
    #- ADD _build/configs/custom.repo /etc/yum.repos.d/custom.repo

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

...
```

3. Private Automation Hub を指す **ansible.cfg** ファイルを **files/** サブディレクトリーに作成します。

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

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

4. 内部 PyPI ミラー (Web サーバーまたは Nexus など) を指す **pip.conf** ファイルを **files/** サブディレクトリーに作成します。

```
$ cat files/pip.conf
[global]
index-url = https://<pypi_mirror_fqdn>/
trusted-host = <pypi_mirror_fqdn>
```

5. オプション: カスタム実行環境に RPM を追加するために **bindep.txt** ファイルを使用する場合は、非接続の Satellite、または RPM リポジトリーをホストする他の場所を指す **files/** サブディレクトリーに、**custom.repo** ファイルを作成します。この手順が必要な場合は、**custom.repo** ファイルに対応するサンプルの **execution-environment.yml** ファイル内の手順をコメント解除します。
次の例は UBI リポジトリーの場合です。他のローカルリポジトリーもこのファイルに追加できます。ミラーコンテンツが Web サーバーのどこにあるかに応じて、URL パスの変更が必要な場合があります。

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

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

6. Private Automation Hub Web サーバー証明書の署名に使用される CA 証明書を追加します。Private Automation Hub が、インストーラーによって提供される自己署名証明書を使用する場合は、以下の手順を行います。

   a. Private Automation Hub からファイル **/etc/pulp/certs/pulp_webserver.crt** をコピーし、**hub-ca.crt** という名前を付けます。

   b. **hub-ca.crt** ファイルを **files/** サブディレクトリーに追加します。

7. Private Automation Hub が、認証局によって署名されたユーザー提供の証明書を使用する場合は、以下の手順を行います。

   a. その CA 証明書のコピーを作成し、**hub-ca.crt** という名前を付けます。

   b. **hub-ca.crt** ファイルを **files/** サブディレクトリーに追加します。

8. 上記の手順が完了したら、カスタム実行環境イメージに必要なコンテンツを含む Python の **requirements.txt** と Ansible コレクションの **requirements.yml** を作成します。

注記

必要なコレクションはすべて、Private Automation Hub に事前にアップロードする必要があります。

次のファイルは、**custom-ee/** ディレクトリーに存在する必要があります。**bindep.txt** と **files/custom.repo** は任意です。

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

1 directory, 8 files
```

関連情報

バージョン 3 の形式と要件の詳細は、Execution Environment Definition: Version 3 Format を参照してください。

4.12.3. カスタム実行環境の構築

新しいカスタム実行環境を作成する前に、コンテンツをダウンロードするためにプライベートハブからの API トークンが必要になります。

次の手順に従ってトークンを生成します。

1. プライベートハブにログインします。

2. 左側のメニューから "Collections" を選択します。
3. メニューの "Collections" セクションにある "API token" を選択します。
4. トークンを取得したら、Ansible Builder がトークンにアクセスできるように次の環境変数を設定します。

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

5. 次のコマンドを使用してカスタム実行環境を作成します。

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

注記

プライベートハブ証明書が不明な認証局によって署名されているというエラーで構築に失敗した場合は、次のコマンドを実行して、必要なイメージをローカルイメージキャッシュにプルできます。

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

あるいは、プライベートハブ CA 証明書を podman 証明書ストアに追加することもできます。

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

4.12.4. カスタム実行環境の Private Automation Hub へのアップロード

新しい実行環境イメージを自動化ジョブに使用するには、使用前にイメージを Private Automation Hub にアップロードする必要があります。

まず、実行環境イメージがローカルの podman キャッシュに表示されることを確認します。

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

次に、Private Automation Hub のコンテナーレジストリーにログインし、イメージをプッシュして、ジョブテンプレートとワークフローで使用できるようにします。

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

4.13. ANSIBLE AUTOMATION PLATFORM のマイナーリリース間のアップグレード

Ansible Automation Platform 2 のマイナーリリース間でアップグレードを実行するには、この一般的なワークフローを使用します。

手順

1. 最新の Ansible Automation Platform 2 セットアップバンドルをダウンロードして展開します。
2. 既存のインストールのバックアップを作成します。
3. 既存のインストールインベントリーファイルを新しいセットアップバンドルディレクトリーにコピーします。
4. **./setup.sh** を実行して、インストールをアップグレードします。

たとえば、バージョン 2.2.0-7 から 2.3-1.2 にアップグレードする場合は、インストールを実行した最初の Controller ノードに両方のセットアップバンドルがあることを確認します。

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

2.2.0-7 インストールをバックアップします。

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

2.2.0-7 インベントリーファイルを 2.3-1.2 バンドルディレクトリーにコピーします。

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

setup.sh スクリプトを使用して、2.2.0-7 から 2.3-1.2 にアップグレードします。

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

付録A インベントリーファイル変数

次の表には、Ansible インストールインベントリーファイルで使用される事前定義された変数に関する情報が含まれています。これらの変数のすべてが必要なわけではありません。

A.1. 一般的な変数

変数	説明
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は、ノードを Red Hat Insights for Red Hat Ansible Automation Platform Service に登録します。無効にするには、**False** に設定します。 デフォルト = **true**
nginx_user_http_config	**/etc/nginx/nginx.conf** の http セクションの nginx 設定のリスト。 リスト内の各要素が、**http nginx config** に別々の行として提供されます。 デフォルト = 空のリスト
registry_password	**registry_password** は、バンドル以外のインストーラーを使用する場合にのみ必要です。 **registry_url** にアクセスするためのパスワード認証情報。 **[automationcontroller]** グループと **[automationhub]** グループの両方に使用されます。 **registry_username** および **registry_password** に Red Hat Registry Service Account の認証情報を入力し、Red Hat コンテナーレジストリーにリンクします。 **registry_url** が **registry.redhat.io** の場合、バンドルインストーラーを使用しない場合はユーザー名とパスワードが必要です。
registry_url	**[automationcontroller]** グループと **[automationhub]** グループの両方に使用されます。 デフォルト = **registry.redhat.io**

変数	説明
registry_username	**registry_username** は、非バンドルインストーラーを使用する場合にのみ必要です。 **registry_url** にアクセスするためのユーザー認証情報。 **[automationcontroller]** および **[automationhub]** グループの両方に使用されますが、**registry_url** の値が **registry.redhat.io** である場合のみです。 **registry_username** および **registry_password** に Red Hat Registry Service Account の認証情報を入力し、Red Hat コンテナーレジストリーにリンクします。
routable_hostname	**routable hostname** は、インストーラーを実行しているマシンが特定の URL を介してのみターゲットホストにルーティングできる場合 (例: インベントリーで短縮名を使用しているにも関わらず、インストーラーを実行するノードは FQDN を使用することでしか、対象ホストを解決できない場合) に使用されます。 **routable_hostname** が設定されていない場合は、デフォルトで **ansible_host** になります。**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。

A.2. ANSIBLE AUTOMATION HUB 変数

変数	説明
automationhub_admin_password	必須 インベントリーファイルにパスワードをプレーンテキストで指定する場合は、パスワードを引用符で囲む必要があります。

変数	説明
automationhub_api_token	Ansible Automation Platform 2.0 以前からアップグレードする場合は、次のいずれかを行う必要があります。 ● 既存の Ansible Automation Hub トークンを **Automationhub_api_token** として提供するか、 ● 新しいトークンを生成するには、**generate_automationhub_token** を **true** に設定します。 新しいトークンを生成すると、既存のトークンが無効になります。
automationhub_authentication_backend	この変数はデフォルトでは設定されていません。LDAP 認証を使用するには、**ldap** に設定します。 これが **ldap** に設定されている場合は、次の変数も設定する必要があります。 ● **automationhub_ldap_server_uri** ● **automationhub_ldap_bind_dn** ● **automationhub_ldap_bind_password** ● **automationhub_ldap_user_search_base_dn** ● **automationhub_ldap_group_search_base_dn** これらのいずれかが存在しない場合、インストールは停止します。
automationhub_auto_sign_collections	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。 このパラメーターを **true** に設定すると、デフォルトで署名されます。 デフォルト = **false**

変数	説明
automationhub_backup_collections	**任意** Ansible Automation Hub は、**/var/lib/pulp** にアーティファクトを提供します。Automation Controller は、デフォルトでアーティファクトを自動的にバックアップします。 また、**automationhub_backup_collections** を false に設定すると、バックアップ/復元プロセスで **/var/lib/pulp** がバックアップまたは復元されなくなります。 デフォルト = **true**
automationhub_collection_download_count	**任意** ダウンロード数を UI に表示するかどうかを決定します。 デフォルト = **false**
automationhub_collection_seed_repository	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、認証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。 この変数で使用可能な値は、'certified' または 'validated' です。 コンテンツをインストールしない場合は、**automationhub_seed_collections** を **false** に設定してシードを無効にします。 1 つのタイプのコンテンツのみが必要な場合は、**automationhub_seed_collections** を **true** に設定し、**automationhub_collection_seed_repository** を追加するコンテンツのタイプに設定します。
automationhub_collection_signing_service_key	コレクション署名サービスが有効になっている場合は、この変数を指定して、コレクションが適切に署名されるようにする必要があります。 **/absolute/path/to/key/to/sign**

変数	説明
automationhub_collection_signing_service_script	コレクション署名サービスが有効になっている場合は、この変数を指定して、コレクションが適切に署名されるようにする必要があります。 **/absolute/path/to/script/that/signs**
automationhub_create_default_collection_signing_service	コレクション署名サービスを作成するには、この変数を true に設定します。 デフォルト = **false**
automationhub_container_signing_service_key	コンテナー署名サービスが有効になっている場合は、この変数を指定して、コンテナーが適切に署名されるようにする必要があります。 **/absolute/path/to/key/to/sign**
automationhub_container_signing_service_script	コンテナー署名サービスが有効になっている場合は、この変数を指定して、コンテナーが適切に署名されるようにする必要があります。 **/absolute/path/to/script/that/signs**
automationhub_create_default_container_signing_service	コンテナー署名サービスを作成するには、この変数を true に設定します。 デフォルト = **false**
automationhub_disable_hsts	デフォルトのインストールでは、TLS 対応の Ansible Automation Hub がデプロイされます。**HTTP Strict Transport Security** (HSTS) Web セキュリティーポリシーを有効にして Automation Hub をデプロイする場合は、この変数を使用します。この変数は、HSTS Web セキュリティーポリシーメカニズムを無効にします。 デフォルト = **false**
automationhub_disable_https	**任意** Ansible Automation Hub が HTTPS を有効にしてデプロイされている場合。 デフォルト = **false**

変数	説明
automationhub_enable_api_access_log	**true** に設定すると、**/var/log/galaxy_api_access.log** にログファイルが作成され、ユーザー名や IP アドレスなど、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。 デフォルト = **false**
automationhub_enable_analytics	Ansible Automation Platform 2.4 の Automation Hub で使用される pulpcore のバージョンに対して pulp 解析を有効にするかどうかを示すブール値。 pulp 解析を有効にするには、**automationhub_enable_analytics** を true に設定します。 デフォルト = **false**
automationhub_enable_unauthenticated_collection_access	許可されていないユーザーがコレクションを表示できるようにするには、この変数を true に設定します。 デフォルト = **false**
automationhub_enable_unauthenticated_collection_download	許可されていないユーザーがコレクションをダウンロードできるようにするには、この変数を true に設定します。 デフォルト = **false**
automationhub_importer_settings	**任意** galaxy-importer に渡す設定のディクショナリー。 インポート時に、コレクションは一連のチェックを受けることができます。 動作は、**galaxy-importer.cfg** 設定によって駆動されます。 例としては、**ansible-doc**、**ansible-lint**、および **flake8** があります。 このパラメーターを使用すると、この設定を駆動できます。

変数	説明
automationhub_main_url	クライアントが接続するメインの Automation Hub URL。 たとえば、https://<load balancer host> などです。 Automation Hub 環境に Red Hat Single Sign-On を実装している場合は、**automationhub_main_url** を使用して、クライアントが接続するメインの Automation Hub URL を指定します。 指定しない場合は、**[automationhub]** グループの最初のノードが使用されます。
automationhub_pg_database	**必須** データベース名です。 デフォルト = **automationhub**
automationhub_pg_host	内部データベースを使用しない場合は必須です。 Automation Hub が使用するリモート PostgreSQL データベースのホスト名。 デフォルト = **127.0.0.1**
automationhub_pg_password	Automation Hub PostgreSQL データベースのパスワード。 **automationhub_pg_password** での特殊文字の使用は制限されています。**!**、**#**、**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。
automationhub_pg_port	内部データベースを使用しない場合は必須です。 デフォルト = 5432
automationhub_pg_sslmode	必須。 デフォルト = **prefer**。
automationhub_pg_username	必須 デフォルト = **automationhub**

変数	説明
automationhub_require_content_approval	**任意** 値は、コレクションが使用可能になる前に、Automation Hub が承認メカニズムを強制する場合は **true** になります。 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場合は、変数を **false** に設定します。 デフォルト = **true**
automationhub_seed_collections	プリロードを有効にするかどうかを定義するブール値。 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、認証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。 コンテンツをインストールしない場合は、**automationhub_seed_collections** を **false** に設定してシードを無効にします。 1 つのタイプのコンテンツのみが必要な場合は、**automationhub_seed_collections** を **true** に設定し、**automationhub_collection_seed_repository** を追加するコンテンツのタイプに設定します。 デフォルト = **true**
automationhub_ssl_cert	**任意** **/path/to/automationhub.cert**: **web_server_ssl_cert** と同じですが、Automation Hub の UI と API 用です。
automationhub_ssl_key	**任意** **/path/to/automationhub.key** **web_server_ssl_key** と同じですが、Automation Hub の UI と API 用です。

変数	説明
automationhub_ssl_validate_certs	Red Hat Ansible Automation Platform 2.2 以降では、この値は使用されなくなりました。 デフォルトでは、Ansible Automation Platform は自己署名証明書を使用してデプロイされるため、Automation Hub がそれ自体を要求するときに証明書を検証する必要がある場合は、値を **true** に設定します。 デフォルト = **false**
automationhub_upgrade	**Deprecated** Ansible Automation Platform 2.2.1 以降では、この値は **true** に固定されています。 Automation Hub は常に最新のパッケージで更新されます。
automationhub_user_headers	Ansible Automation Hub の Web サーバーの nginx ヘッダーのリスト。 リスト内の各要素が、Web サーバーの nginx 設定に別々の行として提供されます。 デフォルト = 空のリスト
ee_from_hub_only	Automation Hub を使用してデプロイメントすると、インストーラーは実行環境イメージを Automation Hub にプッシュし、Automation Hub レジストリーからイメージをプルするように Automation Controller を設定します。 Automation Hub を実行環境イメージのプル元となる唯一のレジストリーにするには、この変数を **true** に設定します。 **false** に設定すると、実行環境イメージも Red Hat から直接取得されます。 バンドルインストーラーを使用する場合のデフォルト = **true**。

変数	説明
generate_automationhub_token	Red Hat Ansible Automation Platform 2.0 以前からアップグレードする場合は、次のいずれかの方法を選択してください。 ● 既存の Ansible Automation Hub トークンを **automationhub_api_token** として提供します。 ● **generate_automationhub_token** を **true** に設定し、新しいトークンを生成します。新しいトークンを生成すると、既存のトークンが無効になります。
nginx_hsts_max_age	この変数は、システムが **HTTP Strict Transport Security** (HSTS) ホストと見なされる時間を秒単位で指定します。つまり、HTTPS が通信専用に使用される期間です。 デフォルト = 63072000 秒 (2 年)。
nginx_tls_protocols	Nginx での **ssl_protocols** のサポートを定義します。 使用可能な値 **TLSv1**、**TLSv1.1**、**`TLSv1.2**、**TLSv1.3** TLSv1.1 および TLSv1.2 パラメーターは、OpenSSL 1.0.1 以降が使用されている場合にのみ機能します。 TLSv1.3 パラメーターは、OpenSSL 1.1.1 以降が使用されている場合にのみ機能します。 **nginx_tls-protocols = ['TLSv1.3']** の場合、TLSv1.3 のみが有効になります。複数のプロトコルを設定するには、**nginx_tls_protocols = ['TLSv1.2', 'TLSv.1.3']** を使用します。 デフォルト = **TLSv1.2**。
pulp_db_fields_key	インポートする Fernet 対称暗号鍵への相対パスまたは絶対パス。パスは Ansible 管理ノード上にあります。これは、認証情報など、データベース内の特定のフィールドを暗号化するために使用されます。指定しない場合は、新しいキーが生成されます。

変数	説明
sso_automation_platform_login_theme	**任意** Ansible Automation Platform の管理および外部で管理される Red Hat Single Sign-On に使用されます。 テーマファイルが配置されているディレクトリーへのパス。この変数を変更する場合は、独自のテーマファイルを指定する必要があります。 デフォルト = **ansible-automation-platform**
sso_automation_platform_realm	**任意** Ansible Automation Platform の管理および外部で管理される Red Hat Single Sign-On に使用されます。 SSO のレルムの名前。 デフォルト = **ansible-automation-platform**
sso_automation_platform_realm_displayname	**任意** Ansible Automation Platform の管理および外部で管理される Red Hat Single Sign-On に使用されます。 レルムの表示名。 デフォルト = **Ansible Automation Platform**
sso_console_admin_username	**任意** Ansible Automation Platform の管理および外部で管理される Red Hat Single Sign-On に使用されます。 SSO 管理ユーザー名。 デフォルト = **admin**
sso_console_admin_password	**必須** Ansible Automation Platform の管理および外部で管理される Red Hat Single Sign-On に使用されます。 SSO 管理パスワード。
sso_custom_keystore_file	**任意** Ansible Automation Platform が管理する Red Hat Single Sign-On にのみ使用されます。 お客様が提供した SSO のキーストア。

変数	説明
sso_host	**必須** Ansible Automation Platform の外部で管理される Red Hat Single Sign-On にのみ使用されます。 Automation Hub では、認証のために SSO および SSO 管理認証情報が必要です。 設定用のインベントリーで SSO が提供されていない場合は、この変数を使用して SSO ホストを定義する必要があります。
sso_keystore_file_remote	**任意** Ansible Automation Platform が管理する Red Hat Single Sign-On にのみ使用されます。 お客様が提供したキーストアがリモートノードにある場合は **true** に設定します。 デフォルト = **false**
sso_keystore_name	**任意** Ansible Automation Platform が管理する Red Hat Single Sign-On にのみ使用されます。 SSO のキーストアの名前。 デフォルト = **ansible-automation-platform**
sso_keystore_password	HTTPS 対応の SSO のキーストアのパスワード。 Ansible Automation Platform が管理する SSO を使用し、HTTPS が有効になっている場合に必要です。デフォルトのインストールで は、**sso_use_https=True** で SSO をデプロイします
sso_redirect_host	**任意** Ansible Automation Platform の管理および外部で管理される Red Hat Single Sign-On に使用されます。 **sso_redirect_host が** 設定されている場合は、アプリケーションが認証のために SSO に接続するために使用されます。 これは、クライアントマシンから到達可能である必要があります。

変数	説明
sso_ssl_validate_certs	**任意** Ansible Automation Platform の管理および外部で管理される Red Hat Single Sign-On に使用されます。 接続中に証明書を検証する必要がある場合は、**true** に設定します。 デフォルト = **true**
sso_use_https	**任意** Ansible Automation Platform で管理される Single Sign-On と外部で管理される Red Hat Single Sign-On に使用されます (Red Hat Single Sign-On が HTTPS を使用する場合)。 デフォルト = **true**

Ansible Automation Hub から LDAP に直接接続するには、以下の変数を設定する必要があります。**ldap_extra_settings** 変数を使用して渡すことができる追加の LDAP 関連変数のリストについては、Django リファレンスドキュメント を参照してください。

変数	説明
automationhub_ldap_bind_dn	**Automationhub_ldap_bind_password** で LDAP サーバーにバインドするときに使用する名前。 Private Automation Hub を LDAP と統合する際に設定する必要があります。設定しない場合、インストールは失敗します。
automationhub_ldap_bind_password	**必須** **Automationhub_ldap_bind_dn** で使用するパスワード。 Private Automation Hub を LDAP と統合する際に設定する必要があります。設定しない場合、インストールは失敗します。

変数	説明
automationhub_ldap_group_search_base_dn	ユーザーが属する可能性のあるすべての LDAP グループを検索する LDAP 検索オブジェクト。 設定で LDAP グループを参照する場合は、この変数と **automationhub_ldap_group_type** を設定する必要があります。 Private Automation Hub を LDAP と統合する際に設定する必要があります。設定しない場合、インストールは失敗します。 デフォルト = **None**
automationhub_ldap_group_search_filter	**任意** グループメンバーシップを検索するための検索フィルター。 変数は、Automation Hub および LDAP を使用してグループをマッピングするために使用する objectClass タイプを識別します。LDAP を使用して Automation Hub をインストールするのに使用されます。 デフォルト = **(objectClass=Group)**
automationhub_ldap_group_search_scope	**任意** LDAP 認証用の django フレームワークを使用して、LDAP ツリー内のグループを検索するスコープ。LDAP を使用して Automation Hub をインストールするのに使用されます。 デフォルト = **SUBTREE**
automationhub_ldap_group_type	**automationhub_ldap_group_search** によって返されるグループのタイプを説明します。 これは、**automationhub_ldap_group_type_params** および **automationhub_ldap_group_type_class** の値に基づいて動的に設定されます。そうでない場合は、django-ldap からのデフォルト値である 'None' が使用されます。 デフォルト = **django_auth_ldap.config:GroupOfNamesType**

変数	説明
automationhub_ldap_group_type_class	**任意** django-ldap グループタイプクラスのインポート可能なパス。 変数は、LDAP 認証のために django フレームワーク内でグループ検索中に使用されるグループタイプを識別します。LDAP を使用して Automation Hub をインストールするのに使用されます。 デフォルト =**django_auth_ldap.config:GroupOfNamesType**
automationhub_ldap_server_uri	LDAP サーバーの URI。 基礎となる LDAP ライブラリーでサポートされている任意の URI を使用してください。 Private Automation Hub を LDAP と統合する際に設定する必要があります。設定しない場合、インストールは失敗します。
automationhub_ldap_user_search_base_dn	ディレクトリー内のユーザーを検索する LDAP 検索オブジェクト。フィルターパラメーターに、ユーザー名のプレースホルダー %(user)s を含める必要があります。認証が成功するには、1 つの結果を返す必要があります。 Private Automation Hub を LDAP と統合する際に設定する必要があります。設定しない場合、インストールは失敗します。
automationhub_ldap_user_search_filter	**任意** デフォルト = **'(uid=%(user)s)'**
automationhub_ldap_user_search_scope	**任意** LDAP 認証の django フレームワークを使用して、LDAP ツリー内のユーザーを検索するスコープ。LDAP を使用して Automation Hub をインストールするのに使用されます。 デフォルト = **SUBTREE**

A.3. AUTOMATION CONTROLLER 変数

変数	説明
admin_password	インストールの完了時に管理ユーザーが UI にアクセスするためのパスワード。 インベントリーファイルにパスワードをプレーンテキストで指定する場合は、パスワードを引用符で囲む必要があります。
automation_controller_main_url	SSO 設定に必要な代替フロントエンド URL を指定します。
automationcontroller_password	Automation Controller インスタンスのパスワード。 インベントリーファイルにパスワードをプレーンテキストで指定する場合は、パスワードを引用符で囲む必要があります。
automationcontroller_username	Automation Controller インスタンスのユーザー名。
nginx_http_port	nginx HTTP サーバーは受信接続をリッスンします。 デフォルト = 80
nginx_https_port	nginx HTTPS サーバーは、セキュアな接続をリッスンします。 デフォルト = 443
nginx_hsts_max_age	この変数は、システムを **HTTP Strict Transport Security** (HSTS) ホストとして扱う必要がある期間を秒単位で指定します。つまり、HTTPS が通信専用に使用される期間です。 デフォルト = 63072000 秒 (2 年)。

変数	説明
nginx_tls_protocols	Nginx での **ssl_protocols** のサポートを定義します。 使用可能な値 **TLSv1**、**TLSv1.1**、**`TLSv1.2**、**TLSv1.3** TLSv1.1 および TLSv1.2 パラメーターは、OpenSSL 1.0.1 以降が使用されている場合にのみ機能します。 TLSv1.3 パラメーターは、OpenSSL 1.1.1 以降が使用されている場合にのみ機能します。 **nginx_tls-protocols = ['TLSv1.3']** の場合、TLSv1.3 のみが有効になります。複数のプロトコルを設定するには、**nginx_tls_protocols = ['TLSv1.2', 'TLSv.1.3']** を使用します。 デフォルト = **TLSv1.2**。
nginx_user_headers	Automation Controller Web サーバーの nginx ヘッダーのリスト。 リスト内の各要素が、Web サーバーの nginx 設定に別々の行として提供されます。 デフォルト = 空のリスト
node_state	**任意** 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。 デフォルト = **active**。

変数	説明
node_type	**[automationcontroller]** グループの場合: このグループには、2 つの有効な **node_types** を割り当てることができます。 **node_type=control** の場合、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** の場合、すべてを実行できます。 このグループのデフォルト = **hybrid** **[execution_nodes]** グループの場合: このグループには、2 つの有効な **node_types** を割り当てることができます。 **node_type=hop** は、ノードがジョブを実行ノードに転送することを意味します。 **node_type=execution** は、ノードがジョブを実行できることを意味します。 このグループのデフォルト = **execution**。
peers	**任意** **peers** 変数は、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アウトバウンド接続が常に確立されます。 この変数は、他のノードとのネットワーク接続を確立するために使用される **receptor.conf** ファイルに **tcp-peer** エントリーを追加するために使用されます。 peers 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の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。
pg_database	postgreSQL データベースの名前。 デフォルト = **awx**。
pg_host	外部で管理されたデータベースにすることができる postgreSQL ホスト。

変数	説明
pg_password	postgreSQL データベースのパスワードを設定します。 **pg_password** での特殊文字の使用は制限されています。**!**、**#**、**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。 注記 PostgreSQL 13 でユーザーパスワードをより安全に保存できるようになったため、インストール時にインベントリーファイルに **pg_hashed_password** を指定する必要がなくなりました。 インストーラのインベントリーファイルで **pg_password** を指定すると、PostgreSQL は SCRAM-SHA-256 ハッシュを使用して、インストールプロセスの一部としてそのパスワードを保護します。
pg_port	使用する postgreSQL ポート。 デフォルト = 5432
pg_ssl_mode	使用可能な 2 つのモード (**prefer** および **verify-full**) のいずれかを選択します。 クライアント側で強制される SSL の場合は、**verify-full** に設定します。 デフォルト = **prefer**。
pg_username	postgreSQL データベースのユーザー名。 デフォルト = **awx**。
postgres_ssl_cert	postgreSQL SSL 証明書の場所。 **/path/to/pgsql_ssl.cert**
postgres_ssl_key	postgreSQL SSL キーの場所。 **/path/to/pgsql_ssl.key**
postgres_use_cert	postgreSQL ユーザー証明書の場所。 **/path/to/pgsql.crt**
postgres_use_key	postgreSQL ユーザーキーの場所。 **/path/to/pgsql.key**

変数	説明
postgres_use_ssl	postgreSQL が SSL を使用する場合は、この変数を使用します。
postgres_max_connections	インストーラー管理の postgreSQL を使用している場合に適用する最大データベース接続設定。 値の選択方法は、Automation Controller 管理ガイドの PostgreSQL データベース設定 を参照してください。 仮想マシンベースのインストールのデフォルト = 200 (シングルノードの場合) および 1024 (クラスターの場合)
receptor_listener_port	レセプター接続に使用するポート。 デフォルト = 27199
supervisor_start_retry_count	指定すると、**startretries = <value specified>** がスーパーバイザー設定ファイル (/etc/supervisord.d/tower.ini) に追加されます。 **startretries** の詳細は、program:x Section Values を参照してください。 デフォルト値は存在しません。
web_server_ssl_cert	**任意** **/path/to/webserver.cert** **Automationhub_ssl_cert** と同じですが、Web サーバーの UI と API 用です。
web_server_ssl_key	**任意** **/path/to/webserver.key** **Automationhub_server_ssl_key** と同じですが、Web サーバーの UI と API 用です。

A.4. 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。

特定のプラグインに固有の変数の詳細は、Ansible.Builtin のドキュメントを参照してください。

グローバル設定オプションのリストについては、Ansible Configuration Settings を参照してください。

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することができます。SSH プロトコルタイプは **smart**、**ssh**、または **paramiko** です。 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 または名前。
ansible_port	接続ポート番号。 デフォルト: SSH の場合は 22
ansible_user	ホストに接続する際に使用するユーザー名。
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。 常にボールトを使用してください。
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_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 を設定するのに役立ちます。
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_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_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 パイプラインを使用するかどうかを決定します。これにより、**ansible.cfg** のパイプライン設定が上書きされる可能性があります。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。

変数	説明
ansible_ssh_executable	バージョン 2.2 で追加されました。 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_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_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。 **/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。

A.5. EVENT-DRIVEN ANSIBLE CONTROLLER の変数

変数	説明
automationedacontroller_admin_password	Event-Driven Ansible Controller インスタンスによって使用される admin パスワード。 インベントリーファイルにパスワードをプレーンテキストで指定する場合は、パスワードを引用符で囲む必要があります。

変数	説明
automationedacontroller_admin_username	Event-Driven Ansible Controller で管理者スーパーユーザーを識別して作成するために django によって使用されるユーザー名。 デフォルト = **admin**
automationedacontroller_admin_email	Event-Driven Ansible Controller の管理者ユーザーとして django によって使用される電子メールアドレス。 デフォルト = **admin@example.com**
automationedacontroller_allowed_hostnames	Event-Driven Ansible Controller へのユーザーアクセスを有効にする追加アドレスのリスト。 デフォルト = 空のリスト
automationedacontroller_controller_verify_ssl	Event-Driven Ansible Controller から呼び出しを行うときに Automation Controller の Web 証明書を検証するために使用されるブール値フラグ。検証済みは **true** です。未検証は **false** です。 デフォルト = **false**
automationedacontroller_disable_https	HTTPS Event-Driven Ansible Controller を無効にするブール値フラグ。 デフォルト = **false**
automationedacontroller_disable_hsts	HSTS Event-Driven Ansible Controller を無効にするブール値フラグ。 デフォルト = **false**
automationedacontroller_gunicorn_workers	gunicorn を通じて提供される API のワーカーの数。 デフォルト = (コアまたはスレッドの数) * 2 + 1
automationedacontroller_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。 デフォルト = 12

変数	説明
automationedacontroller_nginx_tls_files_remote	cert ソースがリモートホスト上 (true) にあるか、ローカル (false) にあるかを指定するブール値フラグ。 デフォルト = **false**
automationedacontroller_pg_database	Event-Driven Ansible Controller によって使用される Postgres データベース。 デフォルト = **automtionedacontroller**
automationnedacontroller_pg_host	Event-Driven Ansible Controller によって使用される Postgres データベースのホスト名。これは外部で管理されるデータベースにすることもできます。
automationedacontroller_pg_password	Event-Driven Ansible Controller によって使用される Postgres データベースのパスワード。 **automationedacontroller_pg_password** での特殊文字の使用は制限されています。**!**、**#**、**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。
automationedacontroller_pg_port	Event-Driven Ansible Controller によって使用される Postgres データベースのポート番号。 デフォルト = **5432**
automationedacontroller_pg_username	Event-Driven Ansible Controller の Postgres データベースのユーザー名。 デフォルト = **automationedacontroller**
automationedacontroller_rq_workers	Event-Driven Ansible Controller によって使用される Redis Queue (RQ) ワーカーの数。RQ ワーカーは、バックグラウンドで実行される Python プロセスです。 デフォルト = (コアまたはスレッドの数) * 2 + 1
automationedacontroller_ssl_cert	**任意** **/root/ssl_certs/eda.<example>.com.crt** **automationhub_ssl_cert** と同じですが、Event-Driven Ansible Controller UI および API 用です。

変数	説明
automationedacontroller_ssl_key	**任意** **/root/ssl_certs/eda.<example>.com.key** **automationhub_server_ssl_key** と同じですが、Event-Driven Ansible Controller UI および API 用です。
automationedacontroller_user_headers	Event-Driven Ansible Controller の nginx 設定に追加する追加の nginx ヘッダーのリスト。 デフォルト = 空のリスト