



# Red Hat Ansible Automation Platform 2.3

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

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学びます。



## Red Hat Ansible Automation Platform 2.3 Red Hat Ansible Automation Platform インストールガイド

---

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学びます。

## 法律上の通知

Copyright © 2023 Red Hat, Inc.

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

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

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

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

## 概要

フィードバックの提供: このドキュメントを改善するための提案がある場合、またはエラーを見つけた場合は、テクニカルサポート () に連絡し、Docs コンポーネントを使用して Ansible Automation Platform Jira プロジェクトで issue を作成してください。

## 目次

はじめに .....	3
多様性を受け入れるオープンソースの強化 .....	4
第1章 RED HAT ANSIBLE AUTOMATION PLATFORM インストールの概要 .....	5
1.1. 前提条件 .....	5
第2章 システム要件 .....	7
2.1. RED HAT ANSIBLE AUTOMATION PLATFORM のシステム要件 .....	7
2.2. AUTOMATION CONTROLLER のシステム要件 .....	8
2.3. AUTOMATION HUB のシステム要件 .....	11
2.4. POSTGRESQL の要件 .....	11
第3章 RED HAT ANSIBLE AUTOMATION PLATFORM のインストール .....	16
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 .....	16
3.2. RED HAT ANSIBLE AUTOMATION PLATFORM インストーラー設定スクリプトの実行 .....	29
3.3. AUTOMATION CONTROLLER のインストールの検証 .....	29
3.4. AUTOMATION HUB のインストールの検証 .....	30
3.5. インストール後の手順 .....	31
第4章 非接続インストール .....	34
4.1. 接続されていない RHEL への ANSIBLE AUTOMATION PLATFORM のインストール .....	34
4.2. REPOSYNC を使用した RPM リポジトリの同期 .....	34
4.3. リポジトリをホストする新しい WEB サーバーの作成 .....	35
4.4. ローカルにマウントされた DVD での RPM リポジトリへのアクセス .....	36
4.5. インターネット接続なしでサブスクリプションマニフェストを AAP に追加する方法 .....	37
4.6. AAP セットアップバンドルのインストール .....	38
4.7. インストール後のタスクの完了 .....	40
4.8. PRIVATE AUTOMATION HUB へのコレクションのインポート .....	41
4.9. コレクション名前空間の作成 .....	41
4.10. インポートされたコレクションの承認 .....	43
4.11. 非接続環境での実行環境のビルド .....	44
4.12. ANSIBLE-BUILDER RPM のインストール .....	44
付録A インベントリーファイル変数 .....	50
A.1. 一般的な変数 .....	50
A.2. ANSIBLE AUTOMATION HUB 変数 .....	51
A.3. RED HAT SINGLE SIGN-ON 変数 .....	59
A.4. 自動化サービスのカタログ変数 .....	62
A.5. オートメーションコントローラー変数 .....	64
A.6. ANSIBLE 変数 .....	67



## 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。

## 多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、[Red Hat CTO である Chris Wright のメッセージ](#)をご覧ください。



# 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は柔軟性を提供し、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。

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た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。

## インストール後の手順

インストールが正常に完了したら、Ansible Automation Platform の機能の使用を開始できます。

## 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

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ないと、エラーが発生する可能性があります。

- [レジストリーサービスアカウントの作成](#) ガイドの手順に従って Red Hat レジストリーサービスアカウントを作成している。

## 関連情報

プラットフォームインストーラーの取得または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必要があります。権限昇格の詳細は、[Understanding Privilege Escalation](#) を参照してください。
- AWX、PostgreSQL、Pulp など、root からユーザーへ権限を降格できる必要があります。

### 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	注記
サブスクリプション	有効な Red Hat Ansible Automation Platform	
OS	Red Hat Enterprise Linux 8.4 以降 64 ビット(x86)	Red Hat Ansible Automation Platform は OpenShift でもサポートされています。詳細は <a href="#">Red Hat Ansible Automation Platform Operator を OpenShift Container Platform 上にデプロイする</a> を参照してください。
Ansible	バージョン 2.14 (インストール用)	Ansible Automation Platform には、ansible-core 2.14 が含まれる実行環境が同梱されています。
Python	3.8 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。

- 以下のドメイン名が、接続成功のファイアウォールまたはプロキシの許可リストに含まれており、Automation Hub または Galaxy サーバーからコレクションをダウンロードできるようにしてください。
  - **galaxy.ansible.com**

- [cloud.redhat.com](https://cloud.redhat.com)
  - [console.redhat.com](https://console.redhat.com)
  - [sso.redhat.com](https://sso.redhat.com)
- 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する必要があります。



#### 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 ユーザーガイドの [スタートガイド](#) を参照してください。

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

- Ansible Automation Platform が管理するシステムの要件は Ansible と同じです。Ansible ユーザーガイドの [スタートガイド](#) を参照してください。
- Red Hat Ansible Automation Platform は Ansible Playbook に依存しており、Automation Controller をインストールする前に最新の安定したバージョンの Ansible をインストールする必要がありますが、Ansible の手動インストールは不要になりました。
- 新規インストールの場合、Automation Controller は Ansible 2.3 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は、インストールプログラムにより、バンドルから Ansible (およびその依存関係) のインストールが試行されます。
- Ansible を自身でインストールすることにした場合、Ansible Automation Platform インストールプログラムは Ansible がインストールされていることを検出して、再インストールを試行しません。



#### 注記

**yum** 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 2.3 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、複数のコンピューター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を設計できるように、ノードタイプの制御、ハイブリッド、実行、およびホップが抽象化として提供されます。

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

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。

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

## コントロールノード

イベントを処理し、プロジェクト更新およびクリーンアップジョブを含む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。

要件	必須
RAM	16 GB
CPU	4
ローカルディスク	<ul style="list-style-type: none"> <li>● 最小 40 GB。/var/lib/awx で少なくとも 20 GB が使用可能。</li> <li>● 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。</li> <li>● 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。</li> </ul>

## ハイブリッドノード

自動化ジョブとクラスタージョブの両方を実行します。実行ノードと制御ノードの CPU とメモリーに関するコメントも、このノードタイプに適用されます。

要件	必須
RAM	16 GB
CPU	4

要件	必須
ローカルディスク	<ul style="list-style-type: none"> <li>● 最小 40 GB。/var/lib/awx で少なくとも 20 GB が使用可能。</li> <li>● 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。</li> <li>● 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。</li> </ul>

## ホップノード

Automation Mesh の別の部分にトラフィックをルーティングします (たとえば、bastion ホストを別のネットワークにすることもできます)。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。

要件	必須
RAM	16 GB
CPU	4
ローカルディスク	40GB

- 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)。リソースの競合の可能性を回避するには、Ansible は 10 個のフォークごとに 1 GB のメモリーと、Automation Controller 用に 2 GB の予約を行うことを推奨します。詳細は、[Automation Controller Capacity Determination and Job impact](#) 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。
- 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。これらの RAM の制限は、ローリング更新を使用する場合、または構成を要求する各システムがキューに入り、可能な限り迅速に処理される 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する場合、または、Automation Controller が AMI などのイメージを作成または展開している場合は回避されます。これらはすべて、より大規模な環境を管理するための優れたアプローチです。詳細な質問は、Red Hat カスタマーポータル (<https://access.redhat.com/>) から Ansible サポートにお問い合わせください。

## 関連情報

- [サブスクリプションのインポート](#)

## 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	注記
RAM	最小 8GB	<ul style="list-style-type: none"> <li>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)</li> <li>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)</li> <li>設定のフォークに基づく容量については、追加情報を参照してください。</li> </ul>
CPU	最小 2 つ	設定のフォークに基づく容量については、追加情報を参照してください。
ローカルディスク	60GB ディスク	コレクションストレージとして、少なくとも 40 GB を /var 専用にする必要があります。

### 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、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 <https://pah.example.com> のような値で、**automationhub\_main\_url** インベントリー変数を使用します。

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については、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。

## 2.4. 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 の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判断するには、**awx-manage check\_db** コマンドを使用します。

表2.2 データベース

サービス	必須	注記
各 Automation Controller	40GB の専用ハードディスクスペース	<ul style="list-style-type: none"> <li>• ファイルおよび作業ディレクトリーのストレージ用に、<b>/var/</b> に最低 20 GB を割り当てます。</li> <li>• 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。</li> <li>• 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、<b>/var/lib/awx/projects</b> に 2 倍の GB を追加することを検討してください。</li> <li>• 150 GB 以上を推奨</li> <li>• 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。</li> </ul>
各 Automation Hub	60 GB の専用ハードディスクスペース	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。



サービス	必須	注記
データベース	20GB の専用ハードディスクスペース	<ul style="list-style-type: none"> <li>● 150 GB 以上を推奨</li> <li>● 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。</li> <li>● 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、管理対象ホストの数、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。</li> <li>● 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、<b>Automation Controller 管理ガイドの管理ジョブ</b>を参照してください。</li> </ul>

## 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。ただし、スタンドアロンのデータベースサーバーノードの PostgreSQL 設定を調整できます。**ansible\_memtotal\_mb** は、データベースサーバーの合計メモリーサイズになります。

```
max_connections == 1024
shared_buffers == ansible_memtotal_mb*0.3
work_mem == ansible_memtotal_mb*0.03
maintenance_work_mem == ansible_memtotal_mb*0.04
```

## 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。

## 2.4.1. 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

次の手順では、ストレージシステムの書き込み/読み取り IOPS パフォーマンスをベンチマークして、Ansible Automation Platform PostgreSQL データベースの最小要件が満たされているかを確認する方法について説明します。

### 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 テストデータのログファイルを保存するため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 60GB 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。

テストデータの量を減らすには、**size** パラメーターの値を調整します。

### 手順

- 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
```

- 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
```

- 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 はモジュラープラットフォームであり、Automation Hub などの他の Automation Platform コンポーネントとともに Automation Controller をデプロイできます。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 でサポートされているインストールシナリオは多数あります。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、以下の例のいずれかを使用してインストールシナリオを指定する必要があります。

- [内部データベースを備えたスタンドアロン 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](#)
- [外部 \(顧客提供\) データベースを備えた単一の Automation Controller](#)
- [外部 \(インストーラー管理\) データベースを使用する Ansible Automation Platform](#)
- [外部 \(顧客提供\) データベースを使用する Ansible Automation Platform](#)
- [内部データベースを備えたスタンドアロンの Automation Hub](#)
- [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 \(顧客提供\) データベースを備えた単一の Automation Controller](#)
- [Private Automation Hub での LDAP 設定](#)

### 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](#) のいずれかを使用して、**inventory** ファイルを更新します。

## 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。

### 3.1.1. 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 の多数のインストールシナリオをサポートしています。次の例を確認して、好みのインストールシナリオに適したものを選択してください。

#### 重要

- Red Hat Ansible Automation Platform または Automation Hub の場合: [automationhub] グループに Automation Hubホストを追加します。
- 内部データベースの場合: [database] を使用して、Ansible Automation Platform クラスター内の別のホストを指すことはできません。インストーラーによってインストールされるデータベースホストは、一意のホストである必要があります。
- 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 (FDQN) を提供して、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。「localhost」は使用しないでください。
- **pg\_password** には特殊文字を使用しないでください。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。

#### 3.1.1.1. 内部データベースを備えたスタンドアロン 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、内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.acme.org
```

```
[all:vars]
admin_password='<password>'
pg_host="
```

```

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

```

### 3.1.1.2. 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

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.acme.org

[database]
data.acme.org

[all:vars]
admin_password='<password>'
pg_host='data.acme.org'
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>'

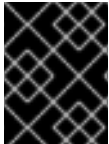
# SSL-related variables
# If set, this will install a custom CA certificate to the system trust store.
# custom_ca_cert=/path/to/ca.crt
# Certificate and key to install in nginx for the web UI and API
# web_server_ssl_cert=/path/to/tower.cert
# web_server_ssl_key=/path/to/tower.key
# Server-side SSL 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
```

### 3.1.1.3. 外部 (顧客提供) 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、プラットフォームインストーラーによって管理されない別のノード上の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。



#### 重要

この例では、データベースグループの下にホストがありません。これは、データベースがすでに存在し、別の場所で管理されていることをインストーラーに示します。

```
[automationcontroller]
controller.acme.org

[database]

[all:vars]
admin_password='<password>'

pg_host='data.acme.org'
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
```

### 3.1.1.4. 外部 (インストーラー管理) データベースを使用する Ansible Automation Platform

この例を使用して、インベントリーファイルに入力し、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、2つの Automation Controller ノード、2つの実行ノード、および外部管理データベースを備えた Automation Hub が含まれています。

```

# Automation Controller Nodes
# There are two valid node_types that can be assigned for this group.
# A node_type=control implies that the node will only be able to run
# project and inventory updates, but not regular jobs.
# A node_type=hybrid will have the ability to run everything.
# If you do not define the node_type, it defaults to hybrid.
#
# control.example node_type=control
# hybrid.example node_type=hybrid
# hybrid2.example <- this will default to hybrid

[automationcontroller]
controller1.acme.org node_type=control
controller2.acme.org node_type=control

# Execution Nodes
# There are two valid node_types that can be assigned for this group.
# A node_type=hop implies that the node will forward jobs to an execution node.
# A node_type=execution implies that the node will be able to run jobs.
# If you do not define the node_type, it defaults to execution.
#
# hop.example node_type=hop
# execution.example node_type=execution
# execution2.example <- this will default to execution

[execution_nodes]
execution1.acme.org node_type=execution
execution2.acme.org node_type=execution

[automationhub]
automationhub.acme.org

[database]
data.acme.org

[all:vars]
admin_password='<password>'
pg_host='data.acme.org'
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>'

# Receptor Configuration
#
receptor_listener_port=27199

# Automation Hub Configuration
#
automationhub_admin_password='<password>'
automationhub_pg_host='data.acme.org'

```



```

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 nginx for the web UI and API
# web_server_ssl_cert=/path/to/tower.crt
# web_server_ssl_key=/path/to/tower.key
# Certificate and key to install in Automation Hub node
# automationhub_ssl_cert=/path/to/automationhub.crt
# automationhub_ssl_key=/path/to/automationhub.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

```

### 3.1.1.5. 外部 (顧客提供) データベースを使用する Ansible Automation Platform

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、各ノードタイプが1つずつ含まれています (コントロール、ハイブリッド、ホップ、および実行、およびプラットフォームインストーラーが管理しない外部マネージドデータベースを使用する Automation Hub)。



#### 重要

この例では、データベースグループの下にホストがありません。これは、データベースがすでに存在し、別の場所で管理されていることをインストーラーに示します。

```

# Automation Controller Nodes
# There are two valid node_types that can be assigned for this group.
# A node_type=control implies that the node will only be able to run
# project and inventory updates, but not regular jobs.
# A node_type=hybrid will have the ability to run everything.
# If you do not define the node_type, it defaults to hybrid.
#
# control.example node_type=control
# hybrid.example node_type=hybrid
# hybrid2.example <- this will default to hybrid

```

```
[automationcontroller]
```

```

hybrid1.acme.org node_type=hybrid
controller1.acme.org node_type=control

# Execution Nodes
# There are two valid node_types that can be assigned for this group.
# A node_type=hop implies that the node will forward jobs to an execution node.
# A node_type=execution implies that the node will be able to run jobs.
# If you do not define the node_type, it defaults to execution.
#
# hop.example node_type=hop
# execution.example node_type=execution
# execution2.example <- this will default to execution

[execution_nodes]
hop1.acme.org node_type=hop
execution1.acme.org node_type=execution

[automationhub]
automationhub.acme.org

[database]

[all:vars]
admin_password='<password>'
pg_host='data.acme.org'
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>'

# Receptor Configuration
#
receptor_listener_port=27199

# Automation Hub Configuration
#
automationhub_admin_password='<password>'
automationhub_pg_host='data.acme.org'
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 nginx for the web UI and API
# web_server_ssl_cert=/path/to/tower.cert
# web_server_ssl_key=/path/to/tower.key
# Certificate and key to install in Automation Hub node
# automationhub_ssl_cert=/path/to/automationhub.cert
# automationhub_ssl_key=/path/to/automationhub.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
```

### 3.1.1.6. 内部データベースを備えたスタンドアロンの 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、内部データベースを使用して Automation Hub のスタンドアロンインスタンスをデプロイします。

```
[automationcontroller]

[automationhub]
automationhub.acme.org ansible_connection=local

[all:vars]
registry_url='registry.redhat.io'
registry_username='<registry username>'
registry_password='<registry password>'

automationhub_admin_password= <PASSWORD>

automationhub_pg_host="
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
```

### 3.1.1.7. 外部 (インストーラ管理) データベースを備えた単一の 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 (インストーラ管理) データベースを使用して Automation Hub の単一インスタンスをデプロイします。

```
[automationcontroller]

[automationhub]
automationhub.acme.org

[database]
data.acme.org

[all:vars]
registry_url='registry.redhat.io'
registry_username='<registry username>'
registry_password='<registry password>'

automationhub_admin_password= <PASSWORD>

automationhub_pg_host='data.acme.org'
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
```

### 3.1.1.8. 外部 (顧客提供) データベースを備えた単一の 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、プラットフォームインストーラーが管理しない外部データベースを使用して Automation Hub の単一インスタンスをデプロイメントします。

**重要**

この例では、データベースグループの下にホストがありません。これは、データベースがすでに存在し、別の場所で管理されていることをインストーラーに示します。

```
[automationcontroller]

[automationhub]
automationhub.acme.org

[database]

[all:vars]
registry_url='registry.redhat.io'
registry_username='<registry username>'
registry_password='<registry password>'

automationhub_admin_password= <PASSWORD>

automationhub_pg_host='data.acme.org'
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
```

### 3.1.1.9. 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しません。

### 3.1.1.9.1. 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. Private Automation Hub に追加のパラメーター (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) を設定する予定がある場合は、次のセクションに進んでください。

### 3.1.1.9.2. 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、それらを設定する 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. 次のような `ldap_extra_settings` を含む YAML ファイルを作成します。

■

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

次に、Private Automation Hub のインストール中に **setup.sh -e @ldapextras.yml** を実行します。

2. 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",}
...
```

次に、Private Automation Hub のインストール中に **setup.sh -e @ldapextras.yml** を実行します。

3. 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",}
...
```

次に、Private Automation Hub のインストール中に **setup.sh -e @ldapextras.yml** を実行します。

4. 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
...
```

次に、Private Automation Hub のインストール中に **setup.sh -e @ldapextras.yml** を実行します。

5. 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",}
...
```

次に、Private Automation Hub のインストール中に **setup.sh -e @ldapextras.yml** を実行します。

6. 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。

- a. 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-users,ou=groups,dc=example,dc=com'
...
```

- b. 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'
...
```

次に、Private Automation Hub のインストール中に **setup.sh -e @ldapextras.yml** を実行します。

7. 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
...
```

次に、Private Automation Hub のインストール中に **setup.sh -e @ldapextras.yml** を実行します。



### 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。

8. 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
```

```
...
```

次に、Private Automation Hub のインストール中に **setup.sh -e @ldapextras.yml** を実行します。

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ます。

## 3.2. 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。

## 3.3. 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。

### 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. **inventory** ファイルに設定した管理者認証情報を使用してログインします。



#### 注記

Automation Controller サーバーはポート 80 ([https://<CONTROLLER\\_SERVER\\_NAME>/](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るため、ポート 443 も使用する必要があります。



#### 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 までお問い合わせください。

Automation Controller へのログインに成功すると、Red Hat Ansible Automation Platform 2.3 のインストールが完了します。

### 3.3.1. 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、以下の資料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
<a href="#">Automation Controller クイックセットアップガイド</a>	Automation Controller を設定して最初の Playbook を実行します。
<a href="#">Automation Controller 管理ガイド</a>	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。
<a href="#">Red Hat Ansible Automation Platform のプロキシサポートの設定</a>	プロキシサーバーを使用して Automation Controller を設定します。
<a href="#">ユーザビリティアナリティクスおよびAutomation Controller からのデータ収集の管理</a>	Red Hat と共有する Automation Controller の情報を管理します。
<a href="#">Automation Controller ユーザーガイド</a>	Automation Controller の機能をより詳細に確認します。

## 3.4. 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. **inventory** ファイルに設定した管理者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 までお問い合わせください。

Automation Hub へのログインが成功すると、Red Hat Ansible Automation Platform 2.3 のインストールが完了します。

### 3.4.1. 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
<a href="#">Private Automation Hub でのユーザーアクセスの管理</a>	Automation Hub のユーザーアクセスを設定します。

リソースリンク	説明
<a href="#">Automation Hub での Red Hat 認定コレクションおよび Ansible Galaxy コレクションの管理</a>	Automation Hub にコンテンツを追加します。
<a href="#">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</a>	Automation Hub で社内で開発したコレクションを公開します。

## 3.5. インストール後の手順

Ansible Automation Platform を初めて使用するユーザーの場合でも、以前の Ansible コンテンツを最新版をインストールした Red Hat Ansible Automation Platform に移行する予定の既存の管理者の場合でも、次の手順を確認して、Ansible Automation Platform 2.3 の新機能を活用し始めてみてください。

### 3.5.1. Ansible Automation Platform 2.3 へのデータの移行

プラットフォーム管理者が Ansible Automation Platform 2.3 へのアップグレードを行う場合は、完了する前にデータを新しいインスタンスに移行する追加の手順が必要になる場合があります。

#### 3.5.1.1. 従来の仮想環境 (venvs) から自動化実行環境への移行

Ansible Automation Platform 2.3 は、カスタム Python 仮想環境 (venvs) よりも、自動化実行環境 (Ansible 自動化の実行およびスケーリングに必要なコンポーネントをパッケージ化するコンテナ化されたイメージ) を優先するようになっています。これには、Ansible Core、Ansible Content Collections、Python の依存関係、Red Hat Enterprise Linux UBI 8、およびその他のパッケージの依存関係が含まれます。

venv を実行環境に移行する場合は、(1) **awx-manage** コマンドを使用して、元のインスタンスから venv のリストを一覧表示してエクスポートし、(2) **ansible-builder** を使用して実行環境を作成する必要があります。

#### 関連情報

- [自動化実行環境へのアップグレード](#)
- [実行環境の作成および消費](#)

#### 3.5.1.2. AnsibleBuilder を使用した Ansible Engine 2.9 イメージへの移行

Ansible Automation Platform 2.3 で使用するために Ansible Engine 2.9 イメージを移行するには、**ansible-builder** ツールで、自動化実行環境で使用するためにイメージ (カスタムプラグインおよび依存関係を含む) を再構築するプロセスを自動化します。

#### 関連情報

Ansible Builder を使用して実行環境を構築する方法の詳細は、[実行環境の作成と消費](#) を参照してください。

#### 3.5.1.3. Ansible Core 2.13 への移行

Ansible Core 2.13 にアップグレードする場合に、最新版の Ansible Core でサポートされるようにするには、Playbook、プラグイン、または Ansible インフラストラクチャーの他の部分を更新する必要があります。Ansible コンテンツを更新して Ansible Core 2.13 との互換性を確保する手順は、[Ansible-core 2.13 移植ガイド](#) を参照してください。

### 3.5.2. 実行環境イメージの場所の更新

Private Automation Hub が個別にインストールされている場合は、Private Automation Hub を指すように実行環境イメージの場所を更新できます。この手順を使用して、実行環境イメージの場所を更新します。

#### 手順

1. **setup.sh** を含むディレクトリーに移動します。
2. 次のコマンドを実行して、**./group\_vars/automationcontroller** を作成します。

```
touch ./group_vars/automationcontroller
```

3. 次の内容を **./group\_vars/automationcontroller** に貼り付けます。環境に合わせて設定を調整してください。

```
# Automation Hub Registry
registry_username: 'your-automation-hub-user'
registry_password: 'your-automation-hub-password'
registry_url: 'automationhub.example.org'
registry_verify_ssl: False

## Execution Environments
control_plane_execution_environment: 'automationhub.example.org/ee-supported-rhel8:latest'

global_job_execution_environments:
- name: "Default execution environment"
  image: "automationhub.example.org/ee-supported-rhel8:latest"
- name: "Ansible Engine 2.9 execution environment"
  image: "automationhub.example.org/ee-29-rhel8:latest"
- name: "Minimal execution environment"
  image: "automationhub.example.org/ee-minimal-rhel8:latest"
```

4. **./setup.sh** スクリプトを実行します。

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

#### 検証

1. システム管理者アクセス権を持つユーザーとして Ansible Automation Platform にログインします。
2. **Administration** → **Execution Environments** に移動します。
3. イメージ列で、実行環境イメージの場所がデフォルト値の **<registry url>/ansible-automation-platform-<version>/<image name>:<tag>** から **<automation hub url>/<image name>:<tag>** に変更になっていることを確認します。

### 3.5.3. 自動化メッシュを使用した自動化スケールアップ

Red Hat Ansible Automation Platform の自動化メッシュコンポーネントは、マルチサイトのデプロイメント全体に自動化を分散するプロセスを簡素化します。IT 環境が複数に分離されている企業の場合、自動化メッシュは、ピアツーピアメッシュ通信ネットワークを使用して実行ノード全体に自動化をデプロイしてスケールアップするための一貫性があり、信頼性の高い方法を提供します。

バージョン 1.x から最新バージョンの Ansible Automation Platform にアップグレードする場合は、データをレガシーの分離ノードから自動化メッシュに必要な実行ノードに移行する必要があります。ハイブリッドノードとコントロールノードのネットワークを計画して、Ansible Automation Platform インストーラーにあるインベントリーファイルを編集して、メッシュ関連の値を各実行ノードに割り当てることで、自動化メッシュを実装できます。

分離ノードから実行ノードに移行する方法は、[Red Hat Ansible Automation Platform のアップグレードおよび移行ガイド](#)を参照してください。

自動化メッシュと、ご使用の環境に合わせて自動化メッシュを設計するさまざまな方法は、[Red Hat Ansible Automation Platform 化メッシュガイド](#)を参照してください。

## 第4章 非接続インストール

### 4.1. 接続されていない RHEL への ANSIBLE AUTOMATION PLATFORM のインストール

Ansible Automation Platform (AAP) オートメーションコントローラーと Private Automation Hub をインストールし、インストーラーによって管理されるデータベースを Automation Controller に配置します。インターネット接続は必要ありません。

#### 4.1.1. 前提条件

切断されたネットワークに AAP 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。

- サブスクリプションマニフェストを作成する。
- AAP セットアップバンドルをダウンロードする。
- Automation Controller と Private Automation Hub サーバーの DNS レコードを作成する。



#### 注記

セットアップバンドルには、切断された環境での AAP 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ます。これらには、AAP RPM とデフォルトの実行環境 (EE) イメージが含まれます。

#### 4.1.2. システム要件

ハードウェア要件は、Automation Platform インストールガイドに記載されています。お使いのバージョンの AAP の [AAP 製品ドキュメント](#) の Red Hat Ansible Automation Platform インストールガイドを参照してください。

#### 4.1.3. RPM ソース

BaseOS および AppStream リポジトリに由来する AAP の RPM 依存関係は、セットアップバンドルには含まれていません。これらの依存関係を追加するには、BaseOS および AppStream リポジトリへのアクセスを取得する必要があります。

- [Satellite](#) は、リポジトリを同期するために Red Hat が推奨する方法です。
- `reposync` - 必要な RPM リポジトリの完全なコピーを作成し、切断されたネットワーク上でそれらをホストします。
- RHEL Binary DVD - RHEL 8 Binary DVD で利用可能な RPM を使用します。



#### 注記

RHEL バイナリー DVD 方式の場合は、RHEL 8.4 以降のサポートされているバージョンの 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。

### 4.2. REPOSYNC を使用した RPM リポジトリの同期

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を移動できます。

## 手順

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

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

2. 再同期を実行します。

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

- a. reposync を使用する場合は、**--download-metadata** を使用し、**--newest-only** は使用しないでください。[RHEL 8] Reposync を参照してください。
  - b. **--newest-only** を使用しない場合、ダウンロードされるリポジトリは最大 90 GB になります。
  - c. **--newest-only** を使用する場合は、ダウンロードされるリポジトリは最大 14 GB になります。
3. Red Hat Single Sign-On (RHSSO) を使用する場合は、これらのリポジトリも同期する必要があります。
    - a. jb-eap-7.3-for-rhel-8-x86\_64-rpms
    - b. rh-sso-7.4-for-rhel-8-x86\_64-rpms
  4. reposync が完了すると、リポジトリを Web サーバーで使えるようになります。
  5. リポジトリを接続されていないネットワークに移動します。

## 4.3. リポジトリをホストする新しい WEB サーバーの作成

リポジトリ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を使用して作成します。

## 手順

新しい Web サーバーを作成する場合は、次の手順を使用します。

1. 前提条件をインストールします。

```
$ sudo dnf install httpd
```

2. リポジトリディレクトリを提供するように 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 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
```

## 4.4. ローカルにマウントされた DVD での RPM リポジトリへのアクセス

DVD からリポジトリにアクセスする場合は、ローカルリポジトリをセットアップする必要があります。このセクションでは、その方法を説明します。



## 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 rhel-8.6-x86_64-dvd.iso /media/rheldvd
```

2. **/etc/yum.repos.d/dvd.repo** に yum リポジトリ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]
```

リポジトリをセットアップするには、[Red Hat Enterprise Linux 8 でローカルにマウントされた DVD の yum リポジトリをセットアップする必要がある](#) を参照してください。

## 4.5. インターネット接続なしでサブスクリプションmanifestを AAP に追加する方法

インターネット接続なしで AAP 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manifestを作成してインポートします。

### 手順

1. [access.redhat.com](https://access.redhat.com) にログインします。

2. **Subscriptions** → **Subscriptions** に移動します。
3. **Subscription Allocations** をクリックします。
4. **Create New subscription allocation** をクリックします。
5. 新しいサブスクリプション割り当てに名前を付けます。
6. タイプとして **Satellite 6.8** → **Satellite 6.8** を選択します。
7. **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。
8. **Subscriptions** タブをクリックします。
9. **Add Subscription** をクリックします。
10. AAP 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。サーバー、ネットワークデバイスなど、AAP によって管理されるノードごとに1つのエンタイトルメントが必要です。
11. **Submit** をクリックします。
12. **Export Manifest** をクリックします。
13. 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。

## 4.6. AAP セットアップバンドルのインストール

“バンドル”バージョンには、AAP 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付属しているため、非接続インストールでは強く推奨されます。

### 4.6.1. セットアップバンドルのダウンロード

#### 手順

1. <https://access.redhat.com/downloads/content/480> に移動して Ansible Automation Platform 2.3 セットアップバンドルの **Download Now** をクリックし、AAP セットアップバンドルパッケージをダウンロードします。

#### 4.6.1.1. セットアップバンドルのインストール

セットアップバンドルのダウンロードとインストールは、Automation Controller 上に配置する必要があります。コントローラーからバンドルを展開し、インベントリーファイルを編集してセットアップを実行します。

1. バンドルを展開します。

```
$ tar xvf \
  ansible-automation-platform-setup-bundle-2.3-1.2.tar.gz
$ cd ansible-automation-platform-setup-bundle-2.3-1.2
```

2. 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'
```



#### 注記

インベントリーはバックアップ、復元、および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維持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することを検討してください。

3. root ユーザーとして AAP セットアップバンドルの実行可能ファイルを実行します。

```
$ sudo -i
# cd /path/to/ansible-automation-platform-setup-bundle-2.3-1.2
# ./setup.sh
```

4. 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。
5. インストールインベントリーファイルで指定された管理者認証情報を使用してログインします。

## 4.7. インストール後のタスクの完了

### 4.7.1. コントローラー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。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 admin でログインします。
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。

### 4.7.2. CA トラストストアの更新

#### 4.7.2.1. 自己署名証明書

デフォルトでは、AAP ハブとコントローラーは自己署名証明書を使用してインストールされます。これにより、コントローラーがハブの証明書を信頼せず、ハブから実行環境をダウンロードしないという問題が発生します。解決策は、ハブの CA 証明書を信頼できる証明書としてコントローラーにインポートすることです。SCP を使用するか、ファイルから別のファイルに直接コピーアンドペーストして、このアクションを実行できます。以下の手順は、ナレッジベースのアーティクル <https://access.redhat.com/solutions/6707451> からコピーしたものです。

#### 4.7.2.2. セキュアコピー (SCP) を使用して、Private Automation Hub 上のルート証明書を Automation Controller にコピーする

コントローラーと 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 の root 証明書をコントローラーにコピーし、そのコントローラー上で **update-ca-trust** を実行して CA トラストストアを更新します。

Automation Controller の場合

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

#### 4.7.2.3. コピーとペースト

Private Automation Hub とコントローラーの間で 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、コントローラー上にある `/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。新しいファイルが作成されたら、**update-ca-trust** コマンドを実行し、新しい証明書で CA トラストストアを更新します。

Private Automation Hub の場合

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

Automation Controller の場合

```
$ 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
```

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

コレクション tarball ファイルは、次のソースからダウンロードできます。

- Red Hat 認定コレクションは [Red Hat Automation Hub](#) にあります。
- コミュニティコレクションは [Ansible Galaxy](#) にあります。

### 4.8.1. Red Hat Automation Hub からのコレクションのダウンロード

このセクションでは、Red Hat Automation Hub からコレクションをダウンロードする方法を説明します。コレクションに依存関係がある場合は、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. <https://console.redhat.com/ansible/automation-hub/> に移動し、Red Hat 認証情報でログインします。
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** をクリックします。
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。

## 4.9. コレクション名前空間の作成

コレクションの名前空間がなければ、正常にインポートされません。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. Private Automation Hub Web コンソールにログインします。
2. **Collections** → **Namespaces** に移動します。
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。

### 4.9.1. GUI を使用したコレクション tarball のインポート

1. Private Automation Hub Web コンソールにログインします。
2. **Collections** → **Namespaces** に移動します。
3. コレクションをインポートする名前空間の **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。

#### 4.9.1.1. CLI 経由で ansible-galaxy 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。

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. 自己署名付き root 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** ファイルをハブ設定で更新します。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>
```



### 注記

コレクションが属する名前空間を事前に作成していない場合は、コレクションの公開に失敗します。

## 4.10. インポートされ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. Private Automation Hub Web コンソールにログインします。
2. **Collections** → **Approval** に移動します。
3. 承認するコレクションの **Approve** をクリックします。
4. コレクションは、Private Automation Hub で使用できるようになりました。



### 注記

コレクションは、ソースに関係なく公開リポジトリに追加されます。

1. 同じ手順で、コレクションの依存関係をインポートします。

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、次のコレクションを提供します。

- <https://console.redhat.com/ansible/automation-hub/repo/published/ansible>
- <https://console.redhat.com/ansible/automation-hub/repo/published/redhat>

### 4.10.1. カスタム実行環境

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 EE イメージは次の方法でビルドできます。

- インターネット向けシステムで EE イメージをビルドし、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、非接続環境で EE イメージをビルドします。
- 非接続環境に必要なすべての変更を含む最小限の基本コンテナイメージを作成し、基本コンテナイメージからカスタム EE イメージをビルドします。

#### 4.10.1.1. 非接続境界を越えたカスタム EE イメージの転送

カスタム実行環境イメージは、既存のドキュメントを使用して、インターネット向けマシン上にビルドできます。作成された実行環境は、ローカルの podman イメージキャッシュで使用できます。その後、非接続境界を越えてカスタム EE イメージを転送できます。非接続境界を越えてカスタム EE イメージを転送するには、まずイメージを保存します。

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、非接続境界を越えてファイルを転送します。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
```

## 4.11. 非接続環境での実行環境のビルド

カスタム実行環境をビルドする場合、ansible-builder ツールはデフォルトで、インターネットから次の要件をダウンロードします。

- EE イメージに追加されたコレクション用の Ansible Galaxy (galaxy.ansible.com) または Automation Hub (cloud.redhat.com)
- コレクションの依存関係として必要なすべての Python パッケージの PyPI (pypi.org)
- UBI ベースの EE イメージを更新するための UBI リポジトリ (cdn.redhat.com)
  - 特定のコレクション要件を満たすために、RHEL リポジトリも必要になる場合があります。
- ansible-builder-rhel8 コンテナイメージにアクセスするための registry.redhat.io

非接続環境で EE イメージをビルドするには、これらのミラーリングされた、または非接続ネットワークで使用可能にされたすべてのサブセットが必要です。Galaxy または Automation Hub から Private Automation Hub にコレクションをインポートする方法は、[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。

ハイサイド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を使用して利用可能にできます。

UBI リポジトリは、**reposync** などのツールを使用してローサイドでミラーリングし、非接続環境にインポートして、Satellite または単純な Web サーバーから利用可能にできます (コンテンツは自由に再配布可能であるため)。

**ansible-builder-rhel8** コンテナイメージは、カスタム EE のインポートと同じ方法で Private Automation Hub にインポートできます。[registry.redhat.io/ansible-automation-platform-21/ansible-builder-rhel8](#) を **localhost/custom-ee** に置き換える方法の詳細は、[非接続境界を越えたカスタム EE イメージの転送](#) を参照してください。これにより、ansible-builder-rhel8 イメージがデフォルトの EE イメージとともに Private Automation Hub レジストリーで使用可能になります。

ハイサイドネットワークですべての前提条件が利用可能になったら、ansible-builder と podman 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. ANSIBLE-BUILDER RPM のインストール

### 手順



1. RHEL システムで、ansible-builder RPM をインストールします。これは、いくつかの方法で行えます。
  - a. 非接続ネットワーク上の Satellite で RHEL ボックス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アタッチし、AAP リポジトリを有効にします。
  - c. ansible-builder RPM をインストールします。



### 注記

基礎となるビルドホストが登録されている場合、EE イメージは Satellite から RHEL コンテンツを利用できるため、Satellite が存在する場合はこれが推奨されます。

2. AAP セットアップバンドルをアーカイブから展開します。
3. 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 -xzf ansible-automation-platform-setup-bundle-2.3-1.2.tar.gz
$ cd ansible-automation-platform-setup-bundle-2.3-1.2/bundle/el8/repos/
$ sudo yum install ansible-builder-1.2.0-1.el9ap.noarch.rpm
python38-requirements-parser-0.2.0-4.el9ap.noarch.rpm
```

4. カスタム EE ビルドアーティファクト用のディレクトリを作成します。

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

5. <https://ansible-builder.readthedocs.io/en/stable/definition/> のドキュメントに従って、カスタム EE の要件を定義する `execution-environment.yml` ファイルを作成します。 **EE\_BASE\_IMAGE** 変数および **EE\_BUILDER\_IMAGE** 変数をオーバーライドして、Private Automation Hub で使用可能な EE を指すようにします。

```
$ cat execution-environment.yml
---
version: 1
build_arg_defaults:
  EE_BASE_IMAGE: '<hub_fqdn>/ee-supported-rhel8:latest'
  EE_BUILDER_IMAGE: '<hub_fqdn>/ansible-builder-rhel8:latest'

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

6. Private Automation Hub を指し、アップロードを許可する認証情報 (管理者ユーザートークンなど) を含む `ansible.cfg` ファイルを作成します。

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

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

7. 非接続 UBI リポジトリミラーを指す **ubi.repo** ファイルを作成します (UBI コンテンツがそこにホストされている場合、これは Satellite である可能性があります)。これは、UBI リポジトリをミラーリングするために **reposync** が使用された出力例です。

```
$ cat ubi.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
```

8. 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。
  - a. 自己署名証明書 (インストーラーのデフォルト) の場合は、Private Automation Hub からファイル **/etc/pulp/certs/root.crt** のコピーを作成し、**hub-root.crt** という名前を付けます。
  - b. Private Automation Hub Web サーバー証明書の要求と署名に内部認証局が使用された場合は、その CA 証明書のコピーを **hub-root.crt** という名前で作成します。
9. カスタム EE 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。必要なコレクションはすべて、Private Automation Hub にアップロードされている必要があることに注意してください。
10. ansible-builder を使用して、EE イメージのビルドに使用されるコンテキストディレクトリーを作成します。

```
$ ansible-builder create
Complete! The build context can be found at: /home/cloud-user/custom-ee/context
$ ls -1F
ansible.cfg
context/
execution-environment.yml
hub-root.crt
pip.conf
requirements.txt
requirements.yml
ubi.repo
```

11. インターネット向けのデフォルトをオーバーライドするために使用されるファイルをコンテキストディレクトリーにコピーします。

```
$ cp ansible.cfg hub-root.crt pip.conf ubi.repo context/
```

12. ファイル `context/Containerfile` を編集し、次の変更を追加します。

- a. 最初の `EE_BASE_IMAGE` ビルドセクションで、`ansible.cfg` および `hub-root.crt` ファイルを追加し、`update-ca-trust` コマンドを実行します。
- b. `EE_BUILDER_IMAGE` ビルドセクションで、`ubi.repo` および `pip.conf` ファイルを追加します。
- c. 最後の `EE_BASE_IMAGE` ビルドセクションで、`ubi.repo` および `pip.conf` ファイルを追加します。

```
$ cat context/Containerfile
ARG EE_BASE_IMAGE=<hub_fqdn>/ee-supported-rhel8:latest
ARG EE_BUILDER_IMAGE=<hub_fqdn>/ansible-builder-rhel8:latest

FROM $EE_BASE_IMAGE as galaxy
ARG ANSIBLE_GALAXY_CLI_COLLECTION_OPTS=
USER root

ADD _build /build
WORKDIR /build

# this section added
ADD ansible.cfg /etc/ansible/ansible.cfg
ADD hub-root.crt /etc/pki/ca-trust/source/anchors/hub-root.crt
RUN update-ca-trust
# end additions
RUN ansible-galaxy role install -r requirements.yml \
    --roles-path /usr/share/ansible/roles
RUN ansible-galaxy collection install \
    $ANSIBLE_GALAXY_CLI_COLLECTION_OPTS -r requirements.yml \
    --collections-path /usr/share/ansible/collections

FROM $EE_BUILDER_IMAGE as builder

COPY --from=galaxy /usr/share/ansible /usr/share/ansible

ADD _build/requirements.txt requirements.txt
RUN ansible-builder introspect --sanitize \
    --user-pip=requirements.txt \
    --write-bindep=/tmp/src/bindep.txt \
    --write-pip=/tmp/src/requirements.txt
# this section added
ADD ubi.repo /etc/yum.repos.d/ubi.repo
ADD pip.conf /etc/pip.conf
# end additions
RUN assemble

FROM $EE_BASE_IMAGE
USER root

COPY --from=galaxy /usr/share/ansible /usr/share/ansible
# this section added
ADD ubi.repo /etc/yum.repos.d/ubi.repo
ADD pip.conf /etc/pip.conf
# end additions
```

```
COPY --from=builder /output/ /output/
RUN /output/install-from-bindep && rm -rf /output/wheels
```

13. **podman** コマンドを使用して、ローカル podman キャッシュに EE イメージを作成します。

```
$ podman build -f context/Containerfile \
-t <hub_fqdn>/custom-ee:latest
```

14. カスタム EE イメージが正常にビルドされたら、それを Private Automation Hub にプッシュします。

```
$ podman push <hub_fqdn>/custom-ee:latest
```

#### 4.12.1. マイナー AAP リリース間のアップグレードワークフロー

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

##### 手順

1. 最新の AAP 2 セットアップバンドルをダウンロードし、アーカイブから展開します。
2. 既存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コントローラー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
<b>enable_insights_collection</b>	<p>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、<b>False</b> に設定します。</p> <p>デフォルト = <b>true</b></p>
<b>registry_password</b>	<p><b>registry_password</b> 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。</p> <p><b>registry_url</b> にアクセスするためのパスワード認証情報。</p> <p><b>[automationcontroller]</b> グループと <b>[automationhub]</b> グループの両方に使用されます。</p> <p><b>registry_username</b> および <b>registry_password</b> に Red Hat Registry Service Account の認証情報を入力し、Red Hat コンテナレジストリーにリンクします。</p> <p><b>registry_url</b> が <b>registry.redhat.io</b> 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。</p>
<b>registry_url</b>	<p><b>[automationcontroller]</b> グループと <b>[automationhub]</b> グループの両方に使用されます。</p> <p>デフォルト = <b>registry.redhat.io</b>。</p>

変数	説明
<b>registry_username</b>	<p><b>registry_username</b> 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。</p> <p><b>registry_url</b> にアクセスするためのユーザー認証情報。</p> <p><b>[automationcontroller]</b> および <b>[automationhub]</b> グループの両方に使用されますが、<b>registry_url</b> の値が <b>registry.redhat.io</b> である場合のみです。</p> <p><b>registry_username</b> および <b>registry_password</b> に Red Hat Registry Service Account の認証情報を入力し、Red Hat コンテナレジストリーにリンクします。</p>
<b>routable_hostname</b>	<p><b>routable hostname</b> 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 を使用することでは、対象ホストを解決できない場合) に使用されます。</p> <p><b>routable_hostname</b> が設定されていない場合は、デフォルトで <b>ansible_host</b> になります。次に、<b>ansible_host</b> が設定されていない場合に限り、<b>inventory_hostname</b> が最後の手段として使用されます。</p> <p>この変数は、特定のホストのホスト変数として使用され、<b>[all:vars]</b> セクションには使用されないことに注意してください。詳細は、<a href="#">1つのマシンへの変数の割り当て: ホスト変数</a> を参照してください。</p>

## A.2. ANSIBLE AUTOMATION HUB 変数

変数	説明
<b>automationhub_admin_password</b>	必須

変数	説明
<b>automationhub_api_token</b>	<p>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。</p> <ul style="list-style-type: none"> <li>● 既存の Ansible Automation Hub トークンを <b>Automationhub_api_token</b> として提供するか、</li> <li>● 新しいトークンを生成するには、<b>generate_automationhub_token</b> を <b>true</b> に設定します。</li> </ul> <p>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。</p>
<b>automationhub_authentication_backend</b>	<p>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、<b>ldap</b> に設定します。</p> <p>これが <b>ldap</b> 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。</p> <ul style="list-style-type: none"> <li>● <b>automationhub_ldap_server_uri</b></li> <li>● <b>automationhub_ldap_bind_dn</b></li> <li>● <b>automationhub_ldap_bind_password</b></li> <li>● <b>automationhub_ldap_user_search_base_dn</b></li> <li>● <b>automationhub_ldap_group_search_base_dn</b></li> </ul>
<b>automationhub_auto_sign_collections</b>	<p>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。</p> <p>このパラメーターを <b>true</b> に設定すると、デフォルトで署名されます。</p> <p>デフォルト = <b>false</b></p>
<b>automationhub_backup_collections</b>	<p><b>オプション</b></p> <p>Ansible Automation Hub は、<b>/var/lib/pulp</b> 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。</p> <p>また、<b>automationhub_backup_collections = false</b> を設定すると、バックアップ/復元プロセスは <b>/var/lib/pulp</b> をバックアップまたは復元しません。</p> <p>デフォルト = <b>true</b></p>



変数	説明
<b>automationhub_collection_seed_repository</b>	<p>バンドルインストーラーを実行すると、検証済みのコンテンツが <b>validated</b> リポジトリにアップロードされ、認定済みのコンテンツが <b>rh-certified</b> リポジトリにアップロードされます。</p> <p>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。</p> <p>この変数で使用可能な値は、'certified' または 'validated' です。</p> <p>コンテンツをインストールしない場合は、<b>automationhub_seed_collections</b> を <b>false</b> に設定してシードを無効にします。</p> <p>1つのタイプのコンテンツのみが必要な場合は、<b>automationhub_seed_collections</b> を <b>true</b> に設定し、<b>automationhub_collection_seed_repository</b> を追加するコンテンツのタイプに設定します。</p>
<b>automationhub_collection_signing_service_key</b>	<p>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。</p> <p><b>/absolute/path/to/key/to/sign</b></p>
<b>automationhub_collection_signing_service_script</b>	<p>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。</p> <p><b>/absolute/path/to/script/that/signs</b></p>
<b>automationhub_create_default_collection_signing_service</b>	<p>デフォルトのインストールでは、署名サービスは作成されません。<b>true</b> に設定すると、署名サービスが作成されます。</p> <p>デフォルト = <b>false</b></p>
<b>automationhub_container_signing_service_key</b>	<p>コンテナ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、コンテナが適切に署名されるようにする必要があります。</p> <p><b>/absolute/path/to/key/to/sign</b></p>
<b>automationhub_container_signing_service_script</b>	<p>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、コンテナが適切に署名されるようにする必要があります。</p> <p><b>/absolute/path/to/script/that/signs</b></p>

変数	説明
<b>automationhub_create_default_contaier_signing_service</b>	<p>デフォルトのインストールでは、署名サービスは作成されません。<b>true</b> に設定すると、署名サービスが作成されます。</p> <p>デフォルト = <b>false</b></p>
<b>automationhub_disable_hsts</b>	<p>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。Automation Hub が <b>HTTP Strict Transport Security (HSTS)</b> Web セキュリティーポリシーを有効にしてデプロイメントされている場合に使用します。特に指定がない限り、HSTS Web セキュリティーポリシーメカニズムは有効になっています。この設定により、必要に応じて無効にすることができます。</p> <p>デフォルト = <b>false</b></p>
<b>automationhub_disable_https</b>	<p><b>オプション</b></p> <p>Ansible Automation Hub が HTTPS を有効にしてデプロイされている場合。</p> <p>デフォルト = <b>false</b></p>
<b>automationhub_enable_api_access_log</b>	<p><b>True</b> に設定すると、<b>/var/log/galaxy_api_access.log</b> にログファイルが作成されます。このファイルは、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実行されたすべてのユーザーアクションをログに記録します。</p> <p>デフォルト = <b>false</b></p>
<b>automationhub_enable_analytics</b>	<p>Ansible Automation Platform 2.3 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。</p> <p>pulp 解析を有効にするには、<b>automationhub_enable_analytics = true</b> を設定します。</p> <p>デフォルトは <b>false</b> です。</p>
<b>automationhub_enable_unauthenticated_collection_access</b>	<p>承認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します。</p> <p>権限の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、<b>automationhub_enable_unauthenticated_collection_access = true</b> を設定します。</p> <p>デフォルトは <b>false</b> です。</p>

変数	説明
<b>automation_hub_enable_unauthenticated_collection_download</b>	<p>承認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します。</p> <p>権限の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、<b>automationhub_enable_unauthenticated_collection_download = true</b> を設定します。</p> <p>デフォルトは <b>false</b> です。</p>
<b>automationhub_importer_settings</b>	<p><b>オプション</b></p> <p>galaxy-importer に渡す設定のディクショナリー。</p> <p>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。</p> <p>動作は、<b>galaxy-importer.cfg</b> 設定によって駆動されます。</p> <p>例としては、<b>ansible-doc</b>、<b>ansible-lint</b>、および <b>flake8</b> があります。</p> <p>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。</p>
<b>automationhub_main_url</b>	<p>クライアントが接続するメインの {HubNameShort} URL。</p> <p>たとえば、https://&lt;load balancer host&gt; などです。</p> <p>指定しない場合は、<b>[automationhub]</b> グループの最初のノードが使用されます。</p> <p>Automation Hub 環境に Red Hat Single Sign-On を実装している場合は、<b>automationhub_main_url</b> 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。</p>
<b>automationhub_pg_database</b>	<p><b>必須</b></p> <p>データベース名です。</p> <p>デフォルト = <b>automationhub</b></p>
<b>automationhub_pg_host</b>	<p>内部データベースを使用しない場合は必須です。</p>
<b>automationhub_pg_password</b>	<p>Automation Hub PostgreSQL データベースのパスワード。</p> <p><b>Automationhub_pg_password</b> に特殊文字を使用しないでください。パスワードが失敗する可能性があります。</p>

変数	説明
<b>automationhub_pg_port</b>	<p>内部データベースを使用しない場合は必須です。</p> <p>デフォルト = 5432</p>
<b>automationhub_pg_sslmode</b>	<p>必須。</p> <p>デフォルト = <b>prefer</b></p>
<b>automationhub_pg_username</b>	<p>必須</p> <p>デフォルト = <b>automationhub</b></p>
<b>automationhub_require_content_approval</b>	<p>オプション</p> <p>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。</p> <p>デフォルトでは、コレクションを Automation Hub にアップロードする場合は、ユーザーがコレクションを使用できるようにする前に、管理者がコレクションを承認する必要があります。</p> <p>コンテンツ承認フローを無効にする場合は、変数を <b>false</b> に設定します。</p> <p>デフォルト = <b>true</b></p>
<b>automationhub_seed_collections</b>	<p>プリロードを有効にするかどうかを定義するブール値。</p> <p>バンドルインストーラーが実行されると、デフォルトで、<b>validated</b> という名前の Private Automation Hub に新しいリポジトリが作成され、検証済みコレクションのリストが更新されます。</p> <p>コンテンツをインストールしない場合は、<b>automationhub_seed_collections</b> を <b>false</b> に設定してシードを無効にします。</p> <p>1つのタイプのコンテンツのみが必要な場合は、<b>automationhub_seed_collections</b> を <b>true</b> に設定し、<b>automationhub_collection_seed_repository</b> を追加するコンテンツのタイプに設定します。</p> <p>デフォルト = <b>true</b></p>
<b>automationhub_ssl_cert</b>	<p>オプション</p> <p><b>/path/to/automationhub.cert</b>  <b>web_server_ssl_cert</b> と同じですが、Automation Hub の UI と API 用。</p>

変数	説明
<b>automationhub_ssl_key</b>	<p>オプション</p> <p><b>/path/to/automationhub.key</b></p> <p><b>web_server_ssl_key</b> と同じですが、Automation Hub UI および API 用です</p>
<b>automationhub_ssl_validate_certs</b>	<p>Red Hat Ansible Automation Platform 2.3 以降では、この値は使用されなくなりました。</p> <p>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。</p> <p>デフォルトは <b>false</b> です。</p>
<b>automationhub_upgrade</b>	<p>Deprecated</p> <p>Ansible Automation Platform 2.2.1 以降では、この値は true に固定されています。</p> <p>Automation Hub は常に最新のパッケージで更新されます。</p>
<b>generate_automationhub_token</b>	<p>Red Hat Ansible Automation Platform 2.0 以前からアップグレードする場合は、次のいずれかを行う必要があります。</p> <ul style="list-style-type: none"> <li>● 既存の Ansible Automation Hub トークンを <b>Automationhub_api_token</b> として提供するか、</li> <li>● 新しいトークンを生成するには、<b>generate_automationhub_token</b> を <b>true</b> 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。</li> </ul>
<b>nginx_hsts_max_age</b>	<p>この変数は、システムが <b>HTTP Strict Transport Security (HSTS)</b> 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 が通信専用で使用される期間です。</p> <p>デフォルト = 63072000 秒 (2 年)。</p>
<b>nginx_tls_protocols</b>	<p>Nginx での <b>ssl_protocols</b> のサポートを定義します。</p> <p>デフォルト = <b>TLSv1.2</b>。</p>

変数	説明
<b>pulp_db_fields_key</b>	インポートする Fernet 対称暗号化キー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。データベース内の特定のフィールド (認証情報など) 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。

Ansible Automation Hub が LDAP に直接接続するため。次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その他の LDAP 関連変数 (以下の **Automationhub\_ldap\_xxx** 変数には含まれていません) のリストは、<https://django-auth-ldap.readthedocs.io/en/latest/reference.html#settings> にあります。

変数	説明
<b>automationhub_ldap_bind_dn</b>	<b>Automationhub_ldap_bind_password</b> で LDAP サーバーにバインドするときに使用する名前。
<b>automationhub_ldap_bind_password</b>	必須  <b>Automationhub_ldap_bind_dn</b> で使用するパスワード。
<b>automationhub_ldap_group_search_base_dn</b>	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 LDAPSearch オブジェクト。設定で LDAP グループを参照する場合は、これと <b>Automationhub_ldap_group_type</b> を設定する必要があります。  デフォルト = <b>None</b>
<b>automationhub_ldap_group_search_filter</b>	オプション  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 = <b>(objectClass=Group)</b>
<b>automationhub_ldap_group_search_scope</b>	オプション  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 = <b>SUBTREE</b>

変数	説明
<b>automationhub_ldap_group_type_class</b>	<p>オプション</p> <p>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。</p> <p>デフォルト  <b>=django_auth_ldap.config:GroupOfNamesType</b></p>
<b>automationhub_ldap_server_uri</b>	<p>LDAP サーバーの URI。これは、基盤となる LDAP ライブラリーでサポートされている任意の URI にすることができます。</p>
<b>automationhub_ldap_user_search_base_dn</b>	<p>ディレクトリー内のユーザーを検索する LDAPSearch オブジェクト。フィルターパラメーターには、ユーザー名のプレースホルダー %(user)s を追加する必要があります。認証が成功するには、1 つの結果を返す必要があります。</p>
<b>automationhub_ldap_user-search_scope</b>	<p>オプション</p> <p>LDAP 認証用の 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。</p> <p>デフォルト = `SUBTREE`</p>

### A.3. RED HAT SINGLE SIGN-ON 変数

\* これらの変数は、**automationhub** または **automationcatalog** に使用します。

変数	説明
<b>sso_automation_platform_login_theme</b>	<p>オプション</p> <p>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。</p> <p>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。</p> <p>デフォルト = <b>ansible-automation-platform</b></p>

変数	説明
<b>sso_automation_platform_realm</b>	<p>オプション</p> <p>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。</p> <p>SSO のレルムの名前。</p> <p>デフォルト = <b>ansible-automation-platform</b></p>
<b>sso_automation_platform_realm_displayname</b>	<p>オプション</p> <p>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。</p> <p>レルムの表示名。</p> <p>デフォルト = <b>Ansible Automation Platform</b></p>
<b>sso_console_admin_username</b>	<p>オプション</p> <p>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。</p> <p>SSO 管理ユーザー名。</p> <p>デフォルト = <b>admin</b></p>
<b>sso_console_admin_password</b>	<p>必須</p> <p>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。</p> <p>SSO 管理パスワード。</p>
<b>sso_custom_keystore_file</b>	<p>オプション</p> <p>Ansible Automation Platform が管理する Red Hat Single Sign-On にのみ使用されます。</p> <p>お客様が提供した SSO のキーストア。</p>



変数	説明
<b>sso_host</b>	<p><b>必須</b></p> <p>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。</p> <p>Automation Hub および Automation サービスカタログでは、認証に SSO および SSO 管理の認証情報が必要です。</p> <p>アプリケーションに必要な自動化サービスカタログ固有のロールを設定するには、SSO 管理認証情報も必要です。</p> <p>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。</p>
<b>sso_keystore_file_remote</b>	<p><b>オプション</b></p> <p>Ansible Automation Platform が管理する Red Hat Single Sign-On にのみ使用されます。</p> <p>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 <b>true</b> に設定します。</p> <p>デフォルト = <b>false</b></p>
<b>sso_keystore_name</b>	<p><b>オプション</b></p> <p>Ansible Automation Platform が管理する Red Hat Single Sign-On にのみ使用されます。</p> <p>SSO のキーストアの名前。</p> <p>デフォルト = <b>ansible-automation-platform</b></p>
<b>sso_keystore_password</b>	<p>HTTPS 対応の SSO のキーストアのパスワード。</p> <p>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、<b>sso_use_https=True</b> で SSO をデプロイします</p>

変数	説明
<b>sso_redirect_host</b>	<p><b>オプション</b></p> <p>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。</p> <p><b>sso_redirect_host</b> 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。</p> <p>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。</p>
<b>sso_ssl_validate_certs</b>	<p><b>オプション</b></p> <p>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。</p> <p>接続中に証明書を検証する場合は <b>true</b> に設定します。</p> <p>デフォルト = <b>true</b></p>
<b>sso_use_https</b>	<p><b>オプション</b></p> <p>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。</p> <p>Single Sign On が https を使用する場合。</p> <p>デフォルト = <b>true</b></p>

## A.4. 自動化サービスのカタログ変数

変数	説明
<b>automationcatalog_controller_password</b>	<p>コントローラーホストからトークンを生成するために使用されます。</p> <p>同様に、<b>automation_controller_main_url</b> も定義する必要があります。</p>
<b>automationcatalog_controller_token</b>	<p>Automation Controller 用に事前に作成された OAuth トークンに使用されます。このトークンは、トークンを生成する代わりに使用されます。</p>
<b>automationcatalog_controller_username</b>	<p>コントローラーホストからトークンを生成するために使用されます。同様に、<b>automation_controller_main_url</b> も定義する必要があります。</p>

変数	説明
<b>automationcatalog_controller_verify_ssl</b>	<p>自動化サービスカタログから Automation Controller への SSL 検証を有効または無効にするために使用されます。</p> <p>デフォルト = <b>true</b></p>
<b>automationcatalog_disable_hsts</b>	<p>自動化サービスカタログの HSTS Web セキュリティポリシーを有効または無効にするために使用されます。</p> <p>デフォルト = <code>false</code></p>
<b>automationcatalog_disable_https</b>	<p>Services Catalog の HSTS Web セキュリティポリシーを有効または無効にするために使用されます。</p> <p>デフォルトは <b>false</b> です。</p>
<b>automationcatalog_enable_analytics_collection</b>	<p>自動化サービスカタログの分析コレクションのアクティブ化を制御するために使用されます</p>
<b>automationcatalog_main_url</b>	<p>SSO と自動化サービスカタログホストの間で使用する必要がある代替ホスト名がある場合は、Red Hat Single Sign-On ホスト設定によって使用されます。</p>
<b>automationcatalog_pg_database</b>	<p>自動化サービスカタログの postgres データベース URL。</p>
<b>automationcatalog_pg_host</b>	<p>自動化サービスカタログの PostgreSQL ホスト (データベースノード)。</p>
<b>automationcatalog_pg_password</b>	<p>自動化サービスカタログの PostgreSQL データベースのパスワード。</p> <p><b>Automationcatalog_pg_password</b> に特殊文字を使用しないでください。パスワードが失敗する可能性があります。</p>
<b>automationcatalog_pg_port</b>	<p>自動化サービスカタログに使用する PostgreSQL ポート。</p> <p>デフォルト = 5432</p>
<b>automationcatalog_pg_username</b>	<p>自動化サービスカタログの postgres ID。</p>
<b>automationcatalog_ssl_cert</b>	<p>カスタム提供の SSL 証明書ファイルへのパス。<b>automationcatalog_ssl_key</b> が必要です。提供されず、https が有効なままの場合は、内部で管理されている CA が証明書に署名して作成します。</p>

変数	説明
<b>automationcatalog_ssl_key</b>	<p>カスタム提供の SSL 証明書キーファイルへのパス。</p> <p><b>Automationcatalog_ssl_cert</b> が必要です。</p> <p>提供されず、https が有効なままの場合は、内部で管理されている CA が証明書に署名して作成します。</p>

## A.5. オートメーションコントローラー変数

変数	説明
<b>admin_password</b>	インストール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。
<b>automation_controller_main_url</b>	<p>自動化サービスカタログを使用した SSO 設定に必要な代替フロントエンド URL は、URL を指定します。</p> <p>自動化サービスカタログには、コントローラーを Automation Controller でインストールするか、この変数を使用してアクティブでルーティング可能なコントローラーサーバーへの URL を指定する必要があります。</p>
<b>automationcontroller_password</b>	Automation Controller インスタンスのパスワード。
<b>automationcontroller_username</b>	Automation Controller インスタンスのユーザー名。
<b>nginx_http_port</b>	<p>nginx HTTP サーバーは受信接続をリッスンします。</p> <p>デフォルト = 80</p>
<b>nginx_https_port</b>	<p>nginx HTTPS サーバーは、セキュアな接続をリッスンします。</p> <p>デフォルト = 443</p>
<b>nginx_hsts_max_age</b>	<p>この変数は、システムが <b>HTTP Strict Transport Security (HSTS)</b> 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 が通信専用で使用される期間です。</p> <p>デフォルト = 63072000 秒 (2 年)。</p>
<b>nginx_tls_protocols</b>	<p>Nginx での <b>ssl_protocols</b> のサポートを定義します。</p> <p>デフォルト = <b>TLSv1.2</b>。</p>

変数	説明
<b>node_state</b>	<p><b>オプション</b></p> <p>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、<b>active</b>、クラスターからノードを削除する <b>deprovision</b>、または従来の分離されたノードを実行ノードに移行する <b>iso_migrate</b> です。</p> <p>デフォルト = <b>active</b>。</p>
<b>node_type</b>	<p><b>[automationcontroller]</b> グループ用。</p> <p>このグループには、2つの有効な <b>node_types</b> を割り当てることができます。</p> <p><b>node_type=control</b> は、ノードが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ないことを意味します。</p> <p><b>node_type=hybrid</b> には、すべてを実行する機能があります。</p> <p>このグループのデフォルト = <b>hybrid</b>。</p> <p><b>[execution_nodes]</b> グループの場合</p> <p>このグループには、2つの有効な <b>node_type</b> を割り当てることができます。</p> <p><b>node_type=hop</b> 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。</p> <p><b>node_type=execution</b> は、ノードがジョブを実行できることを意味します。</p> <p>このグループのデフォルト = <b>execution</b>。</p>
<b>ピア</b>	<p><b>オプション</b></p> <p><b>peers</b> 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。ピア変数が定義されている場合は常に、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確立されます。</p> <p>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 <b>receptor.conf</b> ファイルに <b>tcp-peer</b> エントリーを追加するために使用されます。 <a href="#">ピアリング</a> を参照してください。</p> <p><b>peers</b> 変数は、インベントリーからのホストおよび/またはグループのコンマ区切りのリストにすることができます。これは、<b>receptor.conf</b> 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。</p>

変数	説明
<b>pg_database</b>	<p>postgres データベースの名前。</p> <p>デフォルト = <b>awx</b>。</p>
<b>pg_host</b>	<p>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。</p>
<b>pg_password</b>	<p>PostgreSQL データベースのパスワードを設定します。</p> <p><b>pg_password</b> には特殊文字を使用しないでください。パスワードが失敗する可能性があります。</p> <p>注記</p> <p>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 <b>pg_hashed_password</b> を指定する必要がなくなりました。</p> <p>インストーラのインベントリーファイルで <b>pg_password</b> 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。</p>
<b>pg_port</b>	<p>使用する PostgreSQL ポート。</p> <p>デフォルト = 5432</p>
<b>pg_ssl_mode</b>	<p><b>preferred</b> または <b>verify-full</b> のいずれか。</p> <p>クライアント側で強制される SSL の場合は、<b>verify-full</b> に設定します。</p> <p>デフォルト = <b>prefer</b>。</p>
<b>pg_username</b>	<p>postgres データベースのユーザー名。</p> <p>デフォルト = <b>awx</b>。</p>
<b>postgres_ssl_cert</b>	<p>postgres SSL 証明書の場所。</p> <p><b>/path/to/pgsql_ssl.cert</b></p>
<b>postgres_ssl_key</b>	<p>postgres ssl キーの場所。</p> <p><b>/path/to/pgsql_ssl.key</b></p>

変数	説明
<b>postgres_use_cert</b>	postgres ユーザー証明書場所。  <b>/path/to/pgsql.crt</b>
<b>postgres_use_key</b>	postgres ユーザーキー場所。  <b>/path/to/pgsql.key</b>
<b>postgres_use_ssl</b>	postgres が SSL を使用する場合。
<b>receptor_listener_port</b>	レセプター接続に使用するポート。  デフォルト = 27199.
<b>supervisor_start_retry_count</b>	指定すると (デフォルト値はなし)、 <b>startretries = &lt;指定された値&gt;</b> をスーパーバイザー設定ファイル (/etc/supervisord.d/tower.ini) に追加します。  <b>startretries</b> の詳細は、 <a href="#">program:x Section Values</a> を参照してください。
<b>web_server_ssl_cert</b>	オプション  <b>/path/to/webserver.cert</b>  <b>Automationhub_ssl_cert</b> と同じですが、Web サーバーの UI と API 用です。
<b>web_server_ssl_key</b>	オプション  <b>/path/to/webserver.key</b>  <b>Automationhub_server_ssl_key</b> と同じですが、Web サーバーの UI と API 用です。

## A.6. 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に関する追加情報は、<https://docs.ansible.com/ansible-core/devel/collections/ansible/builtin/index.html> にあります。

グローバル設定オプションのリストは、[https://docs.ansible.com/ansible-core/devel/reference\\_appendices/config.html](https://docs.ansible.com/ansible-core/devel/reference_appendices/config.html) にあります。

変数	説明
----	----

変数	説明
<b>ansible_connection</b>	<p>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。</p> <p>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 <b>smart</b>、<b>ssh</b>、または <b>paramiko</b> です。</p> <p>デフォルト = <b>smart</b></p>
<b>ansible_host</b>	<b>inventory_hostname</b> の代わりに使用するターゲットホストの IP または名前。
<b>ansible_port</b>	デフォルトではない場合 (ssh の場合は 22) は、接続ポート番号。
<b>ansible_user</b>	ホストに接続する際に使用するユーザー名。
<b>ansible_password</b>	<p>ホストへの認証に使用するパスワード。</p> <p>この変数をプレーンテキストで保存しないでください。</p> <p>常にボールドを使用してください。</p>
<b>ansible_ssh_private_key_file</b>	ssh で使用される秘密鍵ファイル。複数のキーを使用していて、SSH エージェントを使用しない場合に便利です。
<b>ansible_ssh_common_args</b>	この設定は、 <b>sftp</b> 、 <b>scp</b> 、および <b>ssh</b> 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。
<b>ansible_sftp_extra_args</b>	この設定は、デフォルトの <b>sftp</b> コマンドラインに常に付加されます。
<b>ansible_scp_extra_args</b>	この設定は、デフォルトの <b>scp</b> コマンドラインに常に付加されます。
<b>ansible_ssh_extra_args</b>	この設定は、デフォルトの <b>ssh</b> コマンドラインに常に付加されます。
<b>ansible_ssh_pipelining</b>	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、 <b>ansible.cfg</b> 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
<b>ansible_ssh_executable</b>	<p>(バージョン 2.2 で追加されています。)</p> <p>この設定は、システム 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 <b>ssh_executable</b> 設定をオーバーライドできます。</p>
<b>ansible_shell_type</b>	<p>ターゲットシステムのシェルタイプ。 <b>ansible_shell_executable</b> を非 Bourne (sh) 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 <b>csh</b> または <b>fish</b> 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。</p>
<b>ansible_shell_executable</b>	<p>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 <b>/bin/sh</b> に設定されている <b>ansible.cfg</b> の実行可能ファイルがオーバーライドされます。</p> <p><b>/bin/sh</b> を使用できない場合は、つまり、ターゲットマシンに <b>/bin/sh</b> がインストールされていないか、sudo から実行できない場合にのみ、変更する必要があります。</p>
<b>inventory_hostname</b>	<p>この変数は、マシンのホスト名をインベントリースクリプトまたは ansible 設定ファイルから取得します。</p> <p>この変数の値は設定できません。</p> <p>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。</p>