Red Hat Ansible Automation Platform 2.2

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

Red Hat Ansible Automation Platform でサポートされるインストールシナリオの手順および参考情報

Last Updated: 2023-06-02

Red Hat Ansible Automation Platform 2.2 Red Hat Ansible Automation Platform インストールガイド

Red Hat Ansible Automation Platform でサポートされるインストールシナリオの手順および参考情報

法律上の通知

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

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 プロパティー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 インストールの計画

Red Hat Ansible Automation Platform は、Red Hat Enterprise Linux と Red Hat Openshift の両方でサポートされます。このガイドを使用して、Red Hat Enterprise Linux への Red Hat Ansible Automation Platform のインストールを計画してください。

Red Hat OpenShift Container Platform 環境に Red Hat Ansible Automation Platform をインストールするには、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。

1.1. RED HAT ANSIBLE AUTOMATION PLATFORM のシステム要件

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。

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1.1 ベースシステム

	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 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	バージョン 2.2 が必要	Ansible がシステムに存在しない場合は、設定 Playbook で **ansible-core** 2.13 がインストールされます。
Python	3.8 以降	

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。

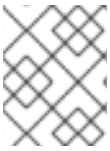
- 以下のドメイン名が、接続成功のファイアウォールまたはプロキシーの許可リストに含まれており、Automation Hub または Galaxy サーバーからコレクションをダウンロードするようにしてください。
  - **galaxy.ansible.com**
  - **cloud.redhat.com**
  - **console.redhat.com**

- **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。

1.1.1. Automation Controller

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、ユーザーが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。以下の表には、ノードのサイジングに関する推奨事項をまとめています。



注記

ホップノード以外のノードで、実行環境のストレージ用に、最低 20 GB を **/var/lib/awx** に割り当てます。

実行ノード	必須	注記
RAM	16 GB	
CPU	4	• 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。
コントロールノード	必須	注記
RAM	16 GB	
CPU	4	• 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。
ハイブリッドノード	必須	注記
RAM	16 GB	

CPU	4	• 自動化およびクラスタージョブの両方を実行します。実行およびコントロールノードの両方のコメントがこのノードタイプに適用されます。
ホップノード	必須	注記
RAM	16 GB	
CPU	4	• 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 アクティビティーは低くなります。ネットワーク帯域幅およびレイテンシーは通常、RAM/CPU のいずれかよりも重要な要素です。
ディスク: サービスノード	40GB の専用ハードディスクスペース	• **自動化コントローラー**: ファイルおよび作業ディレクトリーストレージ用に、最小 20 GB を **/var/** 専用にします。 • ストレージボリュームは、最低ベースラインとして IOPS が 1500 となるようにする必要があります。 • プロジェクトは、制御およびハイブリッドに保存され、ジョブの期間中も実行ノードに保存されます。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。

ディスク: データベースノード	20GB の専用ハードディスクスペース	• 150 GB 以上を推奨 • ストレージボリュームは、ベースライン IOPS を高くする (1500 以上) 必要があります。
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	

関連情報

- Automation Controller の使用を許可するには、Import a subscription を参照してください。

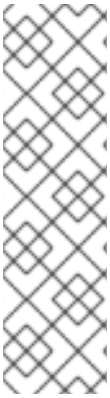
1.1.2. Automation Hub

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。

必須		注記
RAM	最小 8GB	• 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。
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。

	必須	注記
ディスク: サービスノード	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、最低ベースラインとして IOPS が 1500 となるようにする必要があります。
ディスク: データベースノード	20GB の専用ハードディスクスペース	• 150 GB 以上を推奨 • ストレージボリュームは、ベースライン IOPS を高くする (1500 以上) 必要があります。
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	



注記

- すべての自動化コントローラー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。
- 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、**自動化コントローラー管理ガイド**の 管理ジョブ を参照してください。

Amazon EC2

- インスタンスのサイズは m5.large 以上
- ホスト 100 台以上ある場合には m4.xlarge 以上

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

- 実際の RAM 要件は、同時に管理するホストの自動化コントローラー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、自動化コントローラー用に 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自動化コ

ントローラー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、自動化コントローラーが 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。

- 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。

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。
- 自動化コントローラー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。

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
```

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

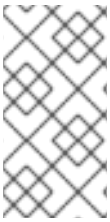
Red Hat Ansible Automation Platform は Ansible Playbook に依存しており、自動化コントローラーをインストールする前に最新の安定したバージョンの Ansible をインストールする必要がありますが、Ansible の手動インストールは不要になりました。

新規インストール時に、自動化コントローラーは Ansible 2.2 の最新の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。Red Hat Ansible Automation Platform が正しく機能するようにするには、**yum** などのパッケージマネージャーを使用して Ansible をインストールし、最新の安定したバージョンをインストールする必要があります。|at| バージョン 3.8 以降には、Ansible バージョン 2.9 が必要です。

- 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。



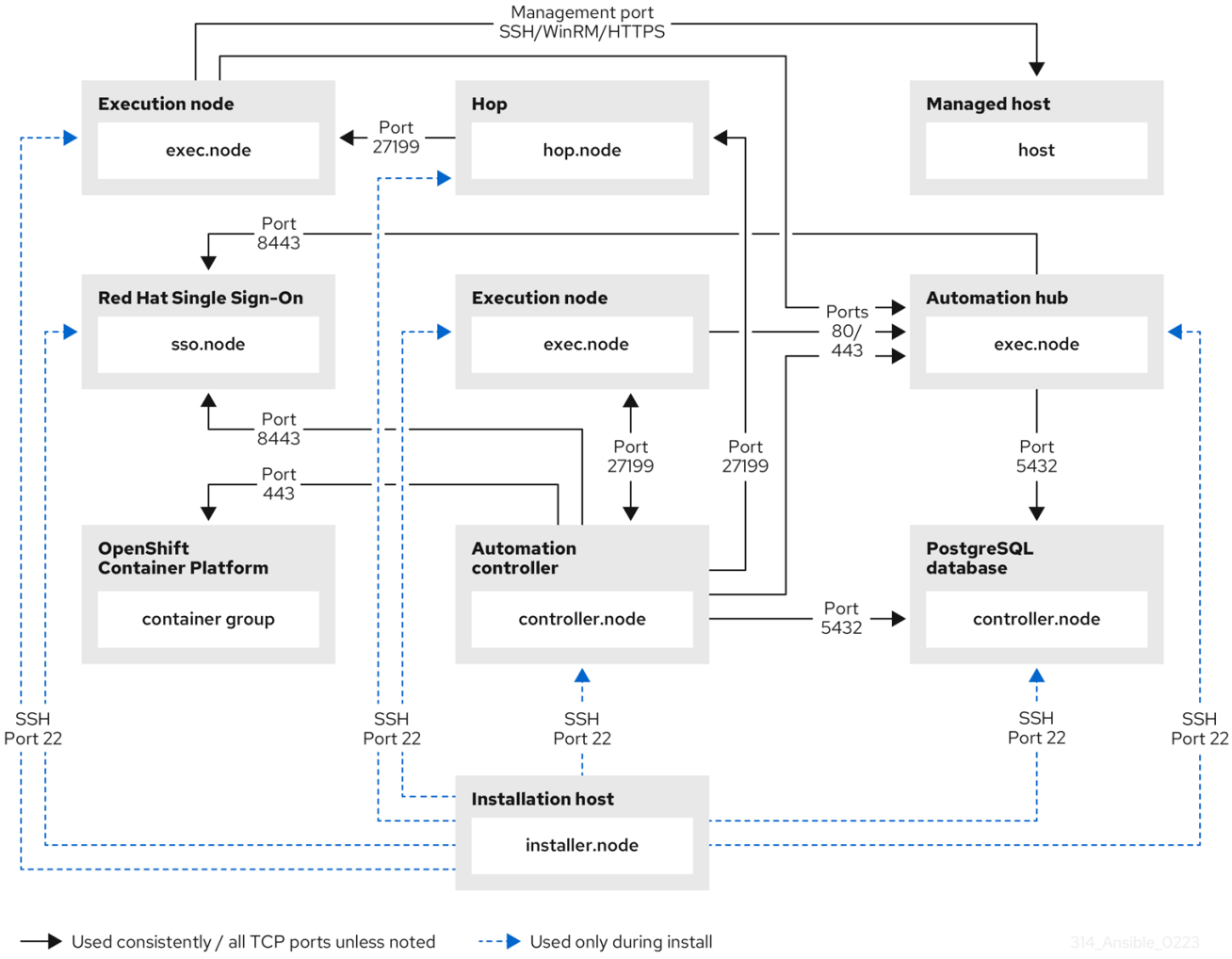
注記

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。|at| バージョン 3.8 以降には、Ansible バージョン 2.9 が必要です。

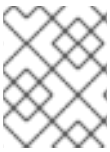
1.2. ネットワークポートおよびプロトコル

Red Hat Ansible Automation Platform (AAP) は、サービスとの通信に多くのポートを使用します。Red Hat Ansible Automation Platform サーバーへの着信接続を有効にするには、これらのポートを開いて利用できるようにする必要があります。これらのポートが利用可能で、サーバーのファイアウォールでブロックされていないことを確認してください。

以下のアーキテクチャー図は、すべての可能なコンポーネントと共に完全にデプロイされた Ansible Automation Platform の例です。



以下の表は、各アプリケーションに必要なデフォルトの Red Hat Ansible Automation Platform 宛先ポートを示しています。



注記

以下に記載のデフォルトの宛先ポートおよびインストーラーインベントリーは設定可能です。お使いの環境に合わせて設定すると、動作が変わる場合があります。

表1.2 PostgreSQL

ポート	プロトコル	サービス	方向	インストーラーのインベントリー変数	用途
22	TCP	SSH	受信および送信	**ansible_port**	インストール時のリモートアクセス
5432	TCP	Postgres	受信および送信	**pg_port**	デフォルトのポート コントローラーからデータベースポートへの接続を許可します。

表1.3 Automation Controller

ポート	プロトコル	サービス	方向	インストーラーのインベントリー変数	用途
22	TCP	SSH	受信および送信	**ansible_port**	インストール
80	TCP	HTTP	受信	**nginx_http_port**	UI/API
443	TCP	HTTPS	受信	**nginx_https_port**	UI/API
5432	TCP	PostgreSQL	受信および送信	**pg_port**	内部データベースが別のコンポーネントとともに使用されている場合に **のみ** 開きます。そうでない場合は、このポートを開放しないでください。 クラスター内のハイブリッドモード

ポート	プロトコル	サービス	方向	インストーラーのインベントリー変数	用途
27199	TCP	receptor	受信および送信	**receptor_listener_port**	必須および自動コントロールプレーンクラスタリング向けに全コントローラーで receptor リスナーポートを許可します。

表1.4 ホップノード

ポート	プロトコル	サービス	方向	インストーラーのインベントリー変数	用途
22	TCP	SSH	受信および送信	**ansible_port**	インストール
27199	TCP	receptor	受信および送信	**receptor_listener_port**	Mesh コントローラーから receptor ポートへの接続を許可します。

表1.5 実行ノード

ポート	プロトコル	サービス	方向	インストーラーのインベントリー変数	用途
22	TCP	SSH	受信および送信	**ansible_port**	インストール

ポート	プロトコル	サービス	方向	インストーラーのインベントリー変数	用途
27199	TCP	receptor	受信および送信	**receptor_listener_port**	Mesh: ノードは、コントローラーに直接ピア接続されます。ホップノードは使用しません。27199 は、実行ノードからの接続を双方向で許可します。 (ホップ接続ノード以外の場合) コントローラーからの receptor ポートへの接続を許可します。 (ホップノードを介してリレーされる場合) ホップノードから receptor ポートへの接続を許可します。

表1.6 コントロールノード

ポート	プロトコル	サービス	方向	インストーラーのインベントリー変数	用途
22	TCP	SSH	受信および送信	**ansible_port**	インストール

ポート	プロトコル	サービス	方向	インストーラーのインベントリー変数	用途
27199	TCP	receptor	受信および送信	**receptor_listener_port**	Mesh: ノードは、コントローラーに直接ピア接続されます。関係するダイレクトノード。27199 は、実行ノードからの接続を双方向で許可します。 (ホップ接続ノード以外の場合) コントローラーからの receptor ポートへの接続を有効にします。 (ホップノードを介してリレーされる場合) ホップノードから receptor ポートへの接続を有効にします。
443	TCP	Podman	受信	**nginx_https_port**	UI/API

表1.7 ハイブリッドノード

ポート	プロトコル	サービス	方向	インストーラーのインベントリー変数	用途
22	TCP	SSH	受信および送信	**ansible_port**	インストール

ポート	プロトコル	サービス	方向	インストーラーのインベントリー変数	用途
27199	TCP	receptor	受信および送信	**receptor_listener_port**	Mesh: ノードは、コントローラーに直接ピア接続されます。ホップノードは使用しません。27199 は、実行ノードからの接続を双方向で許可します。 (ホップ接続ノード以外の場合) コントローラーからの receptor ポートへの接続を有効にします。 (ホップノードを介してリレーされる場合) ホップノードから receptor ポートへの接続を有効にします。
443	TCP	Podman	受信	**nginx_https_port**	UI/API

表1.8 Automation Hub

ポート	プロトコル	サービス	方向	インストーラーのインベントリー変数	用途
22	TCP	SSH	受信および送信	**ansible_port**	インストール
80	TCP	HTTP	受信	固定値	ユーザーインターフェイス

ポート	プロトコル	サービス	方向	インストーラーのインベントリー変数	用途
443	TCP	HTTPS	受信	固定値	ユーザーインターフェイス
5432	TCP	PostgreSQL	受信および送信	**automationhub_pg_port**	内部データベースが別のコンポーネントとともに使用されている場合に **のみ** 開きます。そうでない場合は、このポートを開放しないでください。

表1.9 サービスカタログ

ポート	プロトコル	サービス	方向	インストーラーのインベントリー変数	用途
22	TCP	SSH	受信および送信	**ansible_port**	インストール
443	TCP	HTTPS	受信	**nginx_https_port**	サービスカタログユーザーインターフェイスへのアクセス
5432	TCP	PostgreSQL	受信および送信	**pg_port**	内部データベースが使用される場合に **のみ** 開きます。そうでない場合は、このポートを開放しないでください。

表1.10 Red Hat Insights for Red Hat Ansible Automation Platform

URL	用途
http://api.access.redhat.com:443	一般的なアカウントサービス、サブスクリプション
https://cert-api.access.redhat.com:443	Insights データのアップロード
https://cert.cloud.redhat.com:443	インベントリーのアップロードおよびクラウドコネクター接続
https://cloud.redhat.com	Insights ダッシュボードへのアクセス

表1.11 Automation Hub

URL	用途
https://console.redhat.com:443	一般的なアカウントサービス、サブスクリプション
https://sso.redhat.com:443	TCP
https://automation-hub-prd.s3.amazonaws.com	
https://galaxy.ansible.com	Ansible コミュニティーがキュレートされた Ansible コンテンツ
https://ansible-galaxy.s3.amazonaws.com	
https://registry.redhat.io:443	Red Hat およびパートナーが提供するコンテナーイメージへのアクセス
https://cert.cloud.redhat.com:443	Red Hat およびパートナーキュレートされた Ansible コレクション

表1.12 実行環境 (EE)

URL	用途
https://registry.redhat.io:443	Red Hat およびパートナーが提供するコンテナーイメージへのアクセス
cdn.quay.io:443	Red Hat およびパートナーが提供するコンテナーイメージへのアクセス
cdn01.quay.io:443	Red Hat およびパートナーが提供するコンテナーイメージへのアクセス

URL	用途
cdn02.quay.io:443	Red Hat およびパートナーが提供するコンテナーイメージへのアクセス
cdn03.quay.io:443	Red Hat およびパートナーが提供するコンテナーイメージへのアクセス

重要

イメージマニフェストとファイルシステム Blob は、**registry.redhat.io** から直接提供されます。ただし、2023 年 5 月 1 日以降、ファイルシステム Blob は代わりに **quay.io** から提供されます。コンテナーイメージのプルに関する問題を回避するには、一覧表示された **quay.io** ホスト名への送信接続を有効にする必要があります。この変更を、**registry.redhat.io** へのアウトバウンド接続を有効にするすべてのファイアウォール設定に変更を加えます。ファイアウォールルールを設定するときは、IP アドレスの代わりにホスト名を使用します。この変更を加えた後、引き続き **registry.redhat.io** からイメージをプルできます。Red Hat コンテナーイメージのプルを続行するために、**quay.io** にログインする必要も、**quay.io** レジストリーと直接やりとりする必要もありません。詳細は、こちら の記事を参照してください。

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

Red Hat Ansible Automation Platform をインストールする前に、全ノードに有効なサブスクリプションが割り当てられている **必要があります**。Ansible Automation Platform サブスクリプションを割り当てると、インストールを続行するのに必要なサブクリプションのみのリソースにアクセスできます。

注記

Red Hat アカウントで Simple Content Access Mode を有効にしている場合は、サブスクリプションを割り当てる必要はありません。有効にした場合は、Ansible Automation Platform をインストールする前にシステムを Red Hat Subscription Management (RHSM) または Satellite に登録する必要があります。詳細は、Simple Content Access Mode を参照してください。

手順

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

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

例

subsciption-manager list コマンドの出力例。**Pool ID:** セクションの説明に従って **pool_id** を取得します。

```
Subscription Name: Red Hat Ansible Automation, Premium (5000 Managed Nodes)
  Provides: Red Hat Ansible Engine
  Red Hat Ansible Automation Platform
  SKU: MCT3695
```

```
Contract: ````
Pool ID: <pool_id>
Provides Management: No
Available: 4999
Suggested: 1
```

2. サブスクリプションを割り当てます。

```
# subscription-manager attach --pool=<pool_id>
```

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

検証

- サブスクリプションが正常に割り当てられたことを確認します。

```
# subscription-manager list --consumed
```

トラブルシューティング

- Ansible Automation Platform インストーラーにバンドルされた特定のパッケージを見つけることができない場合や、**Repositories disabled by configuration** のメッセージが表示される場合は、以下のコマンドを使用してリポジトリーを有効化してみてください。
  Red Hat Ansible Automation Platform 2.2 for RHEL 8

```
subscription-manager repos --enable ansible-automation-platform-2.2-for-rhel-8-x86_64-rpms
```

  Red Hat Ansible Automation Platform 2.2 for RHEL 9

```
subscription-manager repos --enable ansible-automation-platform-2.2-for-rhel-9-x86_64-rpms
```

1.4. RED HAT ANSIBLE AUTOMATION PLATFORM プラットフォームコンポーネント

Red Hat Ansible Automation Platform は、以下のコンポーネントで構成されています。

Ansible Automation Hub

Ansible Content Collection の認定済みコンテンツのリポジトリー。Ansible Automation Hub は、Red Hat とそのパートナーがコンテンツを公開し、お客様が認定済みでサポートされている Ansible Content Collection を発見するための一元化されたリポジトリーです。Red Hat Ansible Certified Content は、Red Hat によってテストされ、サポートされているコンテンツをユーザーに提供します。

Private Automation Hub

プライベート自動化ハブは、コンテンツを同期するためのオフラインソリューションとオンプレミスソリューションの両方を提供します。Red Hat クラウド Automation Hub からコレクションと実行環境のイメージを同期し、独自のカスタム自動化コレクションと実行イメージを保存して提供できます。

Ansible Galaxy や他のコンテナーレジストリーなどの他のソースを使用して、プライベート Automation Hub にコンテンツを提供することもできます。プライベート自動化ハブは、エンタープライズディレクトリーと CI/CD パイプラインに統合できます。

Automation Controller

ユーザーインターフェイス (UI) と RESTful アプリケーションプログラミングインターフェイス (API) を使用して、Ansible Automation を制御、保護、および管理するためのエンタープライズフレームワーク。

Automation Services Catalog

Automation Services Catalog は、Red Hat Ansible Automation Platform 内のサービスです。自動化サービスカタログを使用すると、さまざまな環境で Ansible Automation コントローラー上の製品カタログソースを整理および管理できます。

Automation Services Catalog を使用すると、以下が可能になります。

- マルチレベルの承認を個々のプラットフォームインベントリーに適用します。
- プラットフォームの製品形式からポートフォリオにコンテンツを編成します。
- 特定のユーザーグループと共有するポートフォリオを選択します。
- ユーザー要求の実行に関して境界の値を設定します。

自動化メッシュ

自動化メッシュは、既存ネットワークを使用して互いにピアツーピア接続を確立しているノードを介して、大規模な分散ワーカーのコレクション全体で作業分散を容易にするオーバーレイネットワークです。

自動化メッシュは以下を提供します。

- 個別にスケーリングする動的クラスター容量。これにより、ダウンタイムを最小限に抑えてノードを作成、登録、グループ化、グループ化解除、および登録解除できます。
- コントロールプレーンと実行プレーンの分離。コントロールプレーンの容量とは関係なく Playbook の実行容量をスケーリングできます。
- 遅延に対する回復力があり、停止することなく再設定可能であり、停止が存在する場合は動的に再ルーティングして別のパスを選択するデプロイメントの選択肢。
- メッシュルーティングの変更。
- FIPS (Federal Information Processing Standards) に準拠する双方向、マルチホップのメッシュ通信の可能性を含む接続性。

自動化実行環境

Ansible 実行エンジンと、ユーザーが IT 環境とプロセスのあらゆる側面を自動化するのに役立つ数百のモジュールを含むソリューション。実行環境は、一般的に使用されるオペレーティングシステム、インフラストラクチャープラットフォーム、ネットワークデバイス、およびクラウドを自動化します。

Ansible Galaxy

Ansible コンテンツを検索、再利用、および共有するためのハブ。事前にパッケージ化されたロールの形式でコミュニティーが提供する Galaxy コンテンツは、自動化プロジェクトの開始に役立ちます。インフラストラクチャーのプロビジョニング、アプリケーションのデプロイ、およびその他のタスクを完了するためのロールは、Ansible Playbook にドロップして、顧客の環境にすぐに適用できます。

自動化コンテンツナビゲーター

自動化プラットフォームへの主要なコマンドラインインターフェイスとなる **テキストユーザーインターフェイス** (TUI)。コンテンツの構築、実行環境でのローカルでの自動化の実行、Ansible Automation Platform での自動化の実行、将来の **統合開発環境** (IDE) の基盤の提供などのユースケースを扱います。

1.5. RED HAT ANSIBLE AUTOMATION PLATFORM インストーラーの選択および取得

Red Hat Enterprise Linux 環境のインターネット接続に基づいて、必要な Ansible Automation Platform インストーラーを選択します。以下のシナリオを確認し、ニーズを満たす Red Hat Ansible Automation Platform インストーラーを決定してください。

注記

Red Hat カスタマーポータルで Red Hat Ansible Automation Platform インストーラーのダウンロードにアクセスするには、有効な Red Hat カスタマーアカウントが必要です。

インターネットアクセスを使用したインストール

Red Hat Enterprise Linux 環境をインターネットに接続している場合は、Ansible Automation Platform (AAP) インストーラーを選択します。インターネットアクセスを使用してインストールすると、必要な最新のリポジトリー、パッケージ、および依存関係を取得します。AAP インストーラーを設定するには、以下のいずれかの方法を選択します。

tarball インストール

1. https://access.redhat.com/downloads/content/480 に移動します。
2. **Ansible Automation Platform <latest-version> Setup** の **Download Now** をクリックします。
3. ファイルを展開します。

```
$ tar xvzf ansible-automation-platform-setup-<latest-version>.tar.gz
```

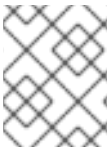
RPM インストール

1. Ansible Automation Platform インストーラーパッケージをインストールします。
   v.2.2 for RHEL 8 for x86_64

```
$ sudo dnf install --enablerepo=ansible-automation-platform-2.2-for-rhel-8-x86_64-rpms
ansible-automation-platform-installer
```

   v.2.2 for RHEL 9 for x86-64

```
$ sudo dnf install --enablerepo=ansible-automation-platform-2.2-for-rhel-9-x86_64-rpms
ansible-automation-platform-installer
```



注記

dnf install は、リポジトリーがデフォルトで無効になっているため、リポジトリーを有効にします。

RPM インストーラーを使用すると、ファイルは **/opt/ansible-automation-platform/installer** ディレクトリーに置かれます。

インターネットアクセスなしでのインストール

インターネットにアクセスできない場合や、オンラインリポジトリーから個別のコンポーネントおよび依存関係をインストールしない場合は、Red Hat Ansible Automation Platform (AAP) の **Bundle** インストーラーを使用します。Red Hat Enterprise Linux リポジトリーへのアクセスは依然として必要です。その他の依存関係はすべて tar アーカイブに含まれます。

1. https://access.redhat.com/downloads/content/480 に移動します。
2. **Ansible Automation Platform <latest-version> Setup Bundleの Download Now** をクリックします。
3. ファイルを展開します。

```
$ tar xvzf ansible-automation-platform-setup-bundle-<latest-version>.tar.gz
```

1.6. インストーラーインベントリーファイルについて

Red Hat Ansible Automation Platform は、インベントリーファイルを使用して、論理的に編成されたインフラストラクチャー内の管理対象ノードまたはホストのリストに対して機能します。Red Hat Ansible Automation Platform インストーラー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し、Ansible へのホストのデプロイについて説明できます。インベントリーファイルを使用することで、Ansible は単一のコマンドで多数のホストを管理できます。インベントリーは、指定する必要があるコマンドラインオプションの数を減らすことで、Ansible をより効率的に使用するのにも役立ちます。

インベントリーファイルは、所有するインベントリープラグインに応じて、多数ある形式のいずれかになります。最も一般的な形式は **INI** と **YAML** です。このドキュメントに記載されているインベントリーファイルは、INI 形式で示されています。

インベントリーファイルの場所は、使用したインストーラーによって異なります。次の表に、可能な場所を示します。

インストーラー	場所
Bundle tar	**/ansible-automation-platform-setup-bundle-<latest-version>**
Non-bundle tar	**/ansible-automation-platform-setup-<latest-version>**
RPM	**/opt/ansible-automation-platform/installer**

次のコマンドを使用して、インベントリー内のホストを確認できます。

```
ansible all -i <path-to-inventory-file. --list-hosts
```

インベントリーファイルの例

```
[automationcontroller]
host1.example.com
host2.example.com
Host4.example.com

[automationhub]
host3.example.com

[database]
Host5.example.com

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

pg_host=''
pg_port=''

pg_database='awx'
pg_username='awx'
pg_password='<password>'

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

インベントリーファイルの最初の部分は、Ansible が使用できるホストまたはグループを指定します。

1.6.1. ホストとグループのガイドライン

データベース

- 外部データベースを使用する場合は、インベントリーファイルの **database** セクションが正しく設定されていることを確認してください。
- パフォーマンスを向上させるために、データベースと自動化コントローラーを同じサーバーに配置しないでください。

Automation Hub

- **[automationhub]** グループに Ansible Automation Hub 情報を追加します
- Ansible Automation Hub と自動化コントローラーを同じノードにインストールしないでください。
- **[automationhub]** 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 Ansible 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。

重要

自動化コントローラーと Ansible Automation Hub を別々にインストールする必要があります。両方が同時にインストールされている場合、**[database]** グループは 2 つを区別しないからです。

[database] で 1 つの値を使用し、自動化コントローラーと Ansible Automation Hub の両方がそれを定義する場合、それらは同じデータベースを使用します。

automation controller

- オートメーションコントローラーは、使用するデータベースのレプリケーションまたはフェールオーバーを設定しません。自動化コントローラーは、所有しているすべてのレプリケーションで機能します。

クラスター化されたインストール

- 既存のクラスターをアップグレードする場合は、既存のインスタンスまたはインスタンスグループを省略するようにクラスターを再設定することもできます。インスタンスまたはインスタンスグループをインベントリーファイルから省略するだけでは、クラスターから削除するには不十分です。インベントリーファイルからインスタンスまたはインスタンスグループを除外するほかに、アップグレードを開始する前にインスタンスまたはインスタンスグループのプロビジョニングを解除する必要もあります。ノードまたはグループのプロビジョニング解除 を参照してください。そうしないと、省略されたインスタンスまたはインスタンスグループが引き続きクラスターと通信するため、アップグレード中に自動化コントローラーサービスで問題が発生する可能性があります。

- クラスター化されたインストールセットアップを作成している場合は、**[localhost]** をすべてのインスタンスのホスト名または IP アドレスに置き換える必要があります。自動化コントローラー、Automation Hub、および自動化サービスカタログのインストーラーは **[localhost]** を受け入れません。すべてのノードとインスタンスは、このホスト名またはアドレスを使用して他のノードに到達できる必要があります。つまり、いずれかのノードで localhost **ansible_connection=local** を使用することはできません。すべてのノードのホスト名に同じ形式を使用します。
  したがって、これは機能しません。

```
[automationhub]
localhost ansible_connection=local
hostA
hostB.example.com
172.27.0.4
```

  代わりに以下の形式を使用します。

```
[automationhub]
hostA
hostB
hostC
```

  または

```
[automationhub]
hostA.example.com
hostB.example.com
hostC.example.com
```

-

1.6.2. ノードまたはグループのプロビジョニング解除

Ansible Automation Platform インストーラーを使用して、ノードとインスタンスグループのプロビジョニングを解除できます。インストーラーを実行すると、グループ内のノードに割り当てられたすべての設定ファイルおよびログが削除されます。

注記

[automationcontroller] グループで指定されている最初のホストを除き、インベントリーの任意のホストのプロビジョニングを解除することができます。

ノードのプロビジョニングを解除するには、インベントリーファイル内のノードまたはグループに **node_state=deprovision** を追加します。

以下はその例です。

デプロイメントから単一のノードを削除するには、以下を実行します。

```
[automationcontroller]
host1.example.com
host2.example.com
host4.example.com   node_state=deprovision
```

または

デプロイからインスタンスグループ全体を削除するには、以下を実行します。

```
[instance_group_restrictedzone]
host4.example.com
host5.example.com

[instance_group_restrictedzone:vars]
node_state=deprovision
```

1.6.3. インベントリー変数

サンプルインベントリーファイルの **[all:vars]** に続く 2 番目の部分は、インストーラーによって使用される変数のリストです。**all** を使用すると、変数がすべてのホストに適用されます。

特定のホストに変数を適用するには、**[hostname:vars]** を使用します。たとえば、**[automationhub:vars]** です。

1.6.4. インベントリーファイルで変数を宣言するためのルール

文字列変数の値は、引用符で囲んで宣言します。以下はその例です。

```
pg_database='awx'
pg_username='awx'
pg_password='<password>'
```

:vars セクションで宣言すると、INI 値は文字列として解釈されます。たとえば、**var=FALSE** は **FALSE** に等しい文字列を作成します。ホスト行とは異なり、**:vars** セクションは行ごとに 1 つのエント

リーのみを受け入れるため、= の後のすべてがエントリーの値である必要があります。ホスト行は、行ごとに複数の **key=value** パラメーターを受け入れます。したがって、スペースがセパレーターではなく値の一部であることを示す方法が必要です。空白を含む値は引用符で囲むことができます (一重または二重)。詳細は、Python shlex parsing rules を参照してください。

INI インベントリーに設定された変数値が特定の型 (文字列やブール値など) でなければならない場合は、常にタスクでフィルターを使用して型を指定します。変数を使用するときは、INI インベントリーで設定されたタイプに依存しないでください。

注記

変数の実際の型に関する混乱を避けるために、インベントリーソースに YAML 形式を使用することを検討してください。YAML インベントリープラグインは、変数値を一貫して正しく処理します。

Ansible インベントリーファイルのパラメーター値に、#、{ または } などの特殊文字が含まれている場合は、値をダブルエスケープ (double-escape) する必要があります (値を単一と二重引用符で囲みます)。

たとえば、**mypasswordwith#hashsigns** を変数 **pg_password** の値として使用するには、Ansible ホストインベントリーファイルで **pg_password="'mypasswordwith#hashsigns'"** として宣言します。

1.6.5. インベントリーファイルでシークレットを保護する

Ansible Vault を使用して機密変数または秘密変数を暗号化できます。ただし、変数名と変数値を暗号化すると、値のソースを見つけるのが難しくなります。これを回避するには、**ansible-vault encrypt_string** を使用して変数を個別に暗号化するか、変数を含むファイルを暗号化します。

手順

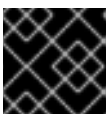
1. 暗号化された認証情報を保存するために、**credentials.yml** というラベルの付いたファイルを作成します。

```
$ cat credentials.yml

admin_password: my_long_admin_pw
pg_password: my_long_pg_pw
registry_password: my_long_registry_pw
```

2. **ansible-vault** を使用して **credentials.yml** ファイルを暗号化します。

```
$ ansible-vault encrypt credentials.yml
New Vault password:
Confirm New Vault password:
Encryption successful
```



重要

暗号化された vault パスワードを安全な場所に保管します。

3. **credentials.yml** ファイルが暗号化されていることを確認します。

```
$ cat credentials.yml
```

```
$ANSIBLE_VAULT;1.1;
AES256363836396535623865343163333339613833363064653364656138313534353135303
764646165393765393063303065323466663330646232363065316666310a37306230313337
633963383130303334313534383962613632303761636632623932653062343839613639653
635643365616233313365363661663931386430 0a353239373433313339613465326339313 0
356335653534643565386536316334643438353464323766386235336136663261363433323
131633436393939646132656164333634306335343039356462646330343839663623230330
65383763
```

4. Ansible Automation Platform 2.2 のインストールのために **setup.sh** を実行し、**credentials.yml** と **--ask-vault-pass オプション** の両方を渡します。

```
$ ANSIBLE_BECOME_METHOD='sudo' ANSIBLE_BECOME=True
ANSIBLE_HOST_KEY_CHECKING=False ./setup.sh -e @credentials.yml -- --ask-vault-pass
```

1.6.6. 追加のインベントリーファイル変数

インベントリーファイルに追加変数を追加して、Red Hat Ansible Automation Platform インストールをさらに設定できます。これらの設定では、Red Hat Ansible Automation Platform 管理用のオプション機能を追加します。テキストエディターでインベントリーファイルを編集して、これらの変数を追加します。

インベントリーファイル変数の定義済み値の表は、付録 A: インベントリーファイル変数 にあります。

1.7. サポート対象のインストールシナリオ

Red Hat は、Red Hat Ansible Automation Platform 向けに以下のインストールシナリオをサポートします。

1.7.1. 同一ノード上にあるデータベースを持つスタンドアロン自動化コントローラーまたはインストーラー以外が管理するデータベース

このシナリオでは、1 台のマシンに Web フロントエンド、REST API バックエンド、データベースを含む自動化コントローラーのインストールが含まれます。PostgreSQL をインストールし、そのデータベースとして使用するように自動化コントローラーを設定します。これは、標準の自動化コントローラーのインストールシナリオとみなされます。

開始するには、**1 台のマシンに Red Hat Ansible Automation Platform コンポーネントをインストール** の 同じノード上のデータベースを使用した自動化コントローラーのインストール を参照してください。

1.7.2. 外部管理データベースが設定されたスタンドアロン自動化コントローラー

このシナリオでは、単一のマシンに自動化コントローラーサーバーをインストールし、リモート PostgreSQL インスタンスとの通信をデータベースとして設定します。このリモート PostgreSQL は、管理するサーバーを使用することも、Amazon RDS などのクラウドサービスで提供することも可能です。

開始するには、**1 台のマシンに Red Hat Ansible Automation Platform コンポーネントをインストール** の 外部の管理データベースを使用した自動化コントローラーのインストール を参照してください。

1.7.3. 同じノード上にあるデータベースまたはインストーラー以外が管理するデータベースを使用するスタンドアロン Automation Hub

このシナリオでは、1台のマシンに Web フロントエンド、REST API バックエンド、データベースなど、Automation Hub のインストールが含まれます。PostgreSQL をインストールし、そのデータベースとして使用するように Automation Hub を設定します。

開始するには、**1台のマシンに Red Hat Ansible Automation Platform コンポーネントをインストール**の 同じノード上のデータベースを使用した Automation Hub のインストール を参照してください。

1.7.4. 外部管理データベースを使用するスタンドアロン Automation Hub

このシナリオでは、1台のマシンに Automation Hub サーバーをインストールし、Red Hat Ansible Automation Platform インストーラーが管理するリモート PostgreSQL データベースをインストールします。

開始するには、**1台のマシンに Red Hat Ansible Automation Platform コンポーネントをインストール**の 外部データベースに Automation Hub をインストール を参照してください。

1.7.5. 自動化コントローラーノードまたはインストーラー以外が管理するデータベースを使用したプラットフォームインストール

このシナリオには、自動化コントローラーノードにあるデータベース、またはインストーラー以外が管理するデータベースを使用した自動化コントローラーおよび Automation Hub のインストールが含まれます。

開始するには、**Red Hat Ansible Automation Platform のインストール**の 自動化コントローラノードまたはインストーラー以外が管理するデータベースへの、データベースを使用した Red Hat Ansible Automation Platform のインストール を参照してください。

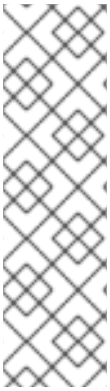
1.7.6. 外部管理データベースを使用したプラットフォームのインストール

このシナリオでは、自動化コントローラーと Automation Hub をインストールし、リモートの PostgreSQL インスタンスとの通信をデータベースとして設定します。このリモート PostgreSQL は、管理するサーバーを使用することも、Amazon RDS などのクラウドサービスで提供することも可能です。

開始するには、**Red Hat Ansible Automation Platform のインストール**の 外部の管理データベースを使用した Red Hat Ansible Automation Platform のインストール を参照してください。

1.7.7. 外部管理データベースを使用した複数マシンのクラスターのインストール

このシナリオでは、複数の自動化コントローラーノードおよび Automation Hub インスタンスをインストールし、リモート PostgreSQL インスタンスとの通信をデータベースとして設定します。このリモート PostgreSQL は、管理するサーバーを使用することも、Amazon RDS などのクラウドサービスで提供することも可能です。このシナリオでは、すべての自動化コントローラーがアクティブでジョブを実行でき、すべてのノードが HTTP 要求を受信できます。



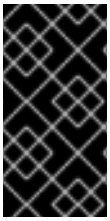
注記

- クラスター設定で実行するには、自動化コントローラーが外部のものである必要があります。PostgreSQL はプライマリーまたはセカンダリーの Tower ノードの 1 つではないマシンにインストールする必要があります。冗長設定の場合、リモートの PostgreSQL バージョン要件は **PostgreSQL 13** です。
  - クラスター化の設定に関する情報は、クラスタリング を参照してください。
- **[automationhub]** ホストの到達可能な IP アドレスを指定して、ユーザーが別のノードから Private Automation Hub のコンテンツを同期できるようにします。

開始するには、**マルチマシンクラスターインストール** の 外部の管理対象データベースを使用した複数ノードの Red Hat Ansible Automation Platform のインストール を参照してください。

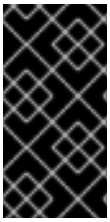
第2章 RED HAT ANSIBLE AUTOMATION PLATFORM のインストール

Red Hat Ansible Automation Platform のインストールには、自動化コントローラー Automation Hub をデプロイします。



重要

Ansible Automation Platform インストーラーでは、インベントリーごとに自動化ハブを **1つのみ** デプロイできます。Automation Hub のスタンドアロンインスタンスには Ansible Automation Platform インストーラーを使用し、任意の数の異なるインベントリーでインストーラーを実行して、複数の Automation Hub をデプロイできます。



重要

インストーラーでは、ユーザーが root としてログインして **./setup.sh** を実行する必要はありません。ユーザーは、root への特権エスカレーションの好ましい方法として、環境変数 **ANSIBLE_BECOME_METHOD** を適切に設定する必要があります。デフォルトのメソッドは **sudo** です。

このインストールオプションには、サポート対象のシナリオが 2 つ含まれています。

2.1. 自動化コントローラーノードまたはインストーラー以外が管理するデータベースを使用して RED HAT ANSIBLE AUTOMATION PLATFORM のインストール

この手順に従って、自動化コントローラーノード上のデータベース、またはインストーラー以外が管理するデータベースを使用して、Red Hat Ansible Automation Platform (自動化コントローラーおよび Automation Hub の両方) をインストールできます。

2.1.1. 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。
- 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。
- Ansible Automation Platform をインストールするにはコンテナーレジストリーサービスが必要です。コンテナーレジストリーにアクセスできると、自動化実行環境を Ansible Automation Platform に読み込み、Ansible Playbook およびロール実行する一貫性のあるコンテナー化された環境を提供できます。デフォルトでは、Ansible Automation Platform は Red Hat レジストリーサービスアカウントを必要とする **registry.redhat.io** を使用します。レジストリーサービスアカウントの作成については、レジストリーサービスアカウントの作成 を参照してください。

2.1.2. 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. [バンドルのインストーラー]

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

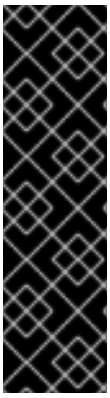
   b. [オンラインインストーラー]

```
$ 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。以下の例に従ってください。

2.1.3. 自動化コントローラーノードまたはインストーラー以外が管理するデータベースのインベントリーファイル例

この例では、Red Hat Ansible Automation Platform をインストールするためにインベントリーファイルを設定する方法を説明します。このインストールインベントリーファイルには、自動化コントローラーノードにあるデータベース、またはインストーラー以外が管理するデータベースを使用した自動化コントローラーおよび Automation Hub のインストールが含まれます。



重要

- 自動化コントローラーと Automation Hub を同じノードにインストールすることはできません。
- **[automationhub]** ホストの到達可能な IP アドレスを指定して、ユーザーが別のノードから Private Automation Hub のコンテンツを同期できるようにします。
- **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。

```
[automationcontroller]
controller.acme.org

[automationhub]
automationhub.acme.org

[all:vars]
admin_password='<password>'
pg_host=''
pg_port=''
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='controller.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
```

2.1.4. 設定スクリプトフラグおよび追加変数

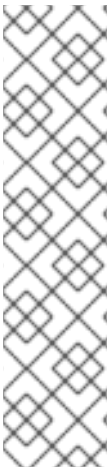
設定スクリプトを実行して自動化コントローラーをインストールする場合に、フラグおよび追加変数を指定することもできます。

表2.1 フラグ

引数	説明
-h	ヘルプメッセージを表示して終了します。
-i INVENTORY_FILE	Ansible インベントリーファイルへのパス (デフォルト: **inventory**)

引数	説明
-e EXTRA_VARS	追加の Ansible 変数を key=value または YAML/JSON として設定します。
-b	インストールの代わりにデータベースのバックアップを実行します。
-r	インストールの代わりにデータベースの復元を行います。
-k	SECRET_KEY を生成 **および** 配布します。

-- の区切り文字を使用して、適用する Ansible 引数を追加します。たとえば、**./setup.sh -i my_awesome_inventory.yml -e matburt_is_country_gold=True — -K** となります。



注記

- **-r** を渡してデータベース復元を実行する場合は、EXTRA_VARS でデフォルト以外のパスを指定しない限り、デフォルト復元パスが使用されます。復元パスを指定する EXTRA_VAR を渡す以下の例を参照してください。

```
./setup.sh -e 'restore_backup_file=/path/to/nondefault/location' -r
```

- **-e bundle_install=false** を渡すと、オンラインインストールを強制的に実行できます。

```
$ ./setup.sh -e bundle_install=false
```

表2.2 追加変数

変数	説明	デフォルト
upgrade_ansible_with_tower	自動化コントローラーのインストール時に、Ansible も最新の状態であることを確認します。	**False**
create_preload_data	Tower のインストール時に、デモ組織、プロジェクト、認証情報、ジョブテンプレートなども作成します。	**True**
bundle_install_folder	バンドルからインストールする場合に、バンドルされているリポジトリーを配置する場所	**var/lib/tower-bundle**

変数	説明	デフォルト
nginx_disable_https	nginx で HTTPS トラフィックを無効にします。HTTPS の負荷をロードバランサーに分散する場合に便利です。	**False**
nginx_disable_hsts	Web セキュリティーポリシーメカニズム HSTS を無効にします。	**False**
nginx_http_port	nginx が HTTP をリッスンするように設定するポート	**80**
nginx_https_port	nginx が HTTPS をリッスンするように設定するポート	**443**
backup_dir	バックアップ時に使用する一時的な場所	**/var/backups/tower/**
restore_backup_file	復元元として使用する別のバックアップファイルを指定します。	なし
required_ram	Tower のインストールに必要な最小メモリー (テストインストールの場合に限り変更してください)	**3750**
min_open_fds	表示ファイルの説明に使用できる最小リソース (テストインストールの場合に限り変更してください)	なし
ignore_preflight_errors	プリフライトチェックを無視します。これはテンプレートや他のシステム以外のイメージにインストールするときに便利です (**required_ram** および **min_open_fds** をオーバーライドします)。	**False**

例

- コアをアップグレードするには、以下を実行します。

```
./setup.sh -e upgrade_ansible_with_tower=1
```

- nginx で処理する https を無効化するには、以下を実行します。

```
./setup.sh -e nginx_disable_https=true
```

- バックアップファイルから復元時にデフォルトではないパスを指定する場合は、以下を実行します。

```
./setup.sh -e 'restore_backup_file=/path/to/nondefault/location' -r
```

2.1.5. 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 のインストールに必要なパラメーターを使用して **inventory** ファイルを更新したら、setup スクリプトを実行ができます。

手順

1. **setup.sh** スクリプトを実行します。

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

インストールが開始します。

2.1.6. 自動化コントローラーインストールの確認

インストールが完了したら、**inventory** ファイルに挿入した管理者認証情報でログインして、自動化コントローラーが正常にインストールしたことを確認できます。

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

自動化コントローラーサーバーはポート 80 (**https://<TOWER_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、443 も利用できる状態に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 (https://access.redhat.com/) から Ansible までお問い合せください。

自動化コントローラーへのログインに成功すると、Red Hat Ansible Automation Platform 2.2 のインストールが完了します。

2.1.6.1. 追加の自動化コントローラーの設定とリソース

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2.3 自動化コントローラーを設定するための資料

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自動化コントローラー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	自動化コントローラーの機能をより詳細に確認します。

2.1.7. Automation Hub のインストールの確認

インストールが完了したら、**inventory** ファイルに挿入した管理者認証情報でログインして、自動化ハブが正常にインストールしたことを確認できます。

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 (https://access.redhat.com/) から Ansible までお問い合せください。

Automation Hub へのログインが成功すると、Red Hat Ansible Automation Platform 2.2 のインストールが完了します。

2.1.7.1. 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2.4 自動化コントローラーを設定するための資料

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 認定コレクションおよび Ansible Galaxy コレクションの管理	Automation Hub にコンテンツを追加します。

リンク	説明
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	Automation Hub で社内で開発したコレクションを公開します。

2.1.8. Ansible Automation Platform 2.2 の次のステップ

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 2.2 の新機能を活用し始めてみてください。

2.1.8.1. Ansible Automation Platform 2.2 へのデータの移行

プラットフォーム管理者が Ansible Automation Platform 2.2 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。

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

Ansible Automation Platform 2.2 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ー化されたイメージ) 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。詳細は、自動化実行環境へのアップグレードガイド および Ansible ビルダーガイド を参照してください。

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

Ansible Automation Platform 2.2 で使用するために Ansible Engine2.9 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。Ansible Builder を使用して実行環境を構築する方法の詳細は、Ansible Builder ガイド を参照してください。

2.1.8.1.3. 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。

2.1.8.2. 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については、**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については、Red Hat Ansible Automation Platform 化メッシュガイド を参照してください。

2.2. 外部の管理データベースを使用した RED HAT ANSIBLE AUTOMATION PLATFORM のインストール

以下の手順に従って、外部の管理データベースを使用して Red Hat Ansible Automation Platform (自動化コントローラーおよび Automation Hub の両方) をインストールできます。

2.2.1. 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。
- 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。

2.2.2. 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. [バンドルのインストーラー]

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

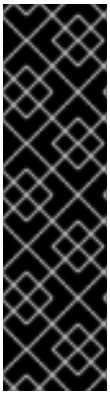
   b. [オンラインインストーラー]

   ```
   $ 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。以下の例に従ってください。

2.2.3. 外部管理データベースを含む Red Hat Ansible Automation Platform インベントリーファイルの例

この例では、Red Hat Ansible Automation Platform をインストールするためにインベントリーファイルを設定する方法を説明します。このインストールインベントリーファイルには、外部の管理データベースを備えた自動化コントローラーと Automation Hub の両方が含まれます。



重要

- 自動化コントローラーと Automation Hub を同じノードにインストールすることはできません。
- **[automationhub]** ホストの到達可能な IP アドレスを指定して、ユーザーが別のノードから Private Automation Hub のコンテンツを同期できるようにします。
- **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。

```
[automationcontroller]
controller.acme.org

[automationhub]
automationhub.acme.org

[database]
database-01.acme.org

[all:vars]
admin_password='<password>'
pg_host='database-01.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>'

# Automation Hub Configuration
#
automationhub_admin_password='<password>'
automationhub_pg_host='database-01.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
```

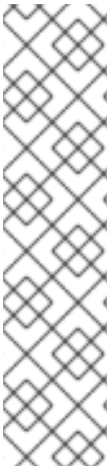
2.2.4. 設定スクリプトフラグおよび追加変数

設定スクリプトを実行して自動化コントローラーをインストールする場合に、フラグおよび追加変数を指定することもできます。

表2.5 フラグ

引数	説明
-h	ヘルプメッセージを表示して終了します。
-i INVENTORY_FILE	Ansible インベントリーファイルへのパス (デフォルト: **inventory**)
-e EXTRA_VARS	追加の Ansible 変数を key=value または YAML/JSON として設定します。
-b	インストールの代わりにデータベースのバックアップを実行します。
-r	インストールの代わりにデータベースの復元を行います。
-k	SECRET_KEY を生成 **および** 配布します。

-- の区切り文字を使用して、適用する Ansible 引数を追加します。たとえば、 **./setup.sh -i my_awesome_inventory.yml -e matburt_is_country_gold=True — -K** となります。



注記

- **-r** を渡してデータベース復元を実行する場合は、EXTRA_VARS でデフォルト以外のパスを指定しない限り、デフォルト復元パスが使用されます。復元パスを指定する EXTRA_VAR を渡す以下の例を参照してください。

```
./setup.sh -e 'restore_backup_file=/path/to/nondefault/location' -r
```

- **-e bundle_install=false** を渡すと、オンラインインストールを強制的に実行できます。

```
$ ./setup.sh -e bundle_install=false
```

表2.6 追加変数

変数	説明	デフォルト
upgrade_ansible_with_tower	自動化コントローラーのインストール時に、Ansible も最新の状態であることを確認します。	**False**
create_preload_data	Tower のインストール時に、デモ組織、プロジェクト、認証情報、ジョブテンプレートなども作成します。	**True**
bundle_install_folder	バンドルからインストールする場合に、バンドルされているリポジトリーを配置する場所	**var/lib/tower-bundle**
nginx_disable_https	nginx で HTTPS トラフィックを無効にします。HTTPS の負荷をロードバランサーに分散する場合に便利です。	**False**
nginx_disable_hsts	Web セキュリティーポリシーメカニズム HSTS を無効にします。	**False**
nginx_http_port	nginx が HTTP をリッスンするように設定するポート	**80**
nginx_https_port	nginx が HTTPS をリッスンするように設定するポート	**443**
backup_dir	バックアップ時に使用する一時的な場所	**/var/backups/tower/**
restore_backup_file	復元元として使用する別のバックアップファイルを指定します。	なし

変数	説明	デフォルト
required_ram	Tower のインストールに必要な最小メモリー (テストインストールの場合に限り変更してください)	**3750**
min_open_fds	表示ファイルの説明に使用できる最小リソース (テストインストールの場合に限り変更してください)	なし
ignore_preflight_errors	プリフライトチェックを無視します。これはテンプレートや他のシステム以外のイメージにインストールするときに便利です (**required_ram** および **min_open_fds** をオーバーライドします)。	**False**

例

- コアをアップグレードするには、以下を実行します。

```
./setup.sh -e upgrade_ansible_with_tower=1
```

- nginx で処理する https を無効化するには、以下を実行します。

```
./setup.sh -e nginx_disable_https=true
```

- バックアップファイルから復元時にデフォルトではないパスを指定する場合は、以下を実行します。

```
./setup.sh -e 'restore_backup_file=/path/to/nondefault/location' -r
```

2.2.5. 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 のインストールに必要なパラメーターを使用して **inventory** ファイルを更新したら、setup スクリプトを実行ができます。

手順

1. **setup.sh** スクリプトを実行します。

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

インストールが開始します。

2.2.6. 自動化コントローラーインストールの確認

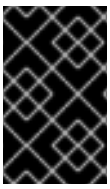
インストールが完了したら、**inventory** ファイルに挿入した管理者認証情報でログインして、自動化コントローラーが正常にインストールしたことを確認できます。

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

自動化コントローラーサーバーはポート 80 (**https://<TOWER_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、443 も利用できる状態に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 (https://access.redhat.com/) から Ansible までお問い合せください。

自動化コントローラーへのログインに成功すると、Red Hat Ansible Automation Platform 2.2 のインストールが完了します。

2.2.6.1. 追加の自動化コントローラーの設定とリソース

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2.7 自動化コントローラーを設定するための資料

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自動化コントローラー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	自動化コントローラーの機能をより詳細に確認します。

2.2.7. Automation Hub のインストールの確認

インストールが完了したら、**inventory** ファイルに挿入した管理者認証情報でログインして、自動化ハブが正常にインストールしたことを確認できます。

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 (https://access.redhat.com/) から Ansible までお問い合せください。

Automation Hub へのログインが成功すると、Red Hat Ansible Automation Platform 2.2 のインストールが完了します。

2.2.7.1. 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2.8 自動化コントローラーを設定するための資料

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 認定コレクションおよび Ansible Galaxy コレクションの管理	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	Automation Hub で社内で開発したコレクションを公開します。

2.2.8. Ansible Automation Platform 2.2 の次のステップ

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 2.2 の新機能を活用し始めてみてください。

2.2.8.1. Ansible Automation Platform 2.2 へのデータの移行

プラットフォーム管理者が Ansible Automation Platform 2.2 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。

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

Ansible Automation Platform 2.2 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ー化されたイメージ) 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。詳細は、自動化実行環境へのアップグレードガイド および Ansible ビルダーガイド を参照してください。

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

Ansible Automation Platform 2.2 で使用するために Ansible Engine2.9 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。Ansible Builder を使用して実行環境を構築する方法の詳細は、Ansible Builder ガイド を参照してください。

2.2.8.1.3. 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。

2.2.8.2. 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については、**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については、Red Hat Ansible Automation Platform 化メッシュガイド を参照してください。

第3章 RED HAT ANSIBLE AUTOMATION PLATFORM コンポーネントの単一マシンへのインストール

Red Hat Ansible Automation Platform コンポーネントは、以下のサポートされるシナリオのいずれかで 1 台のマシンにインストールすることができます。

3.1. データベースを使用した同一ノードへの自動化コントローラーのインストール

これらの手順に従って、同じノード上にデータベースを使用する自動化コントローラーのスタンドアロンインスタンス、またはインストーラー以外が管理するデータベースをインストールできます。このシナリオでは、1 台のマシンに Web フロントエンド、REST API バックエンド、データベースを含む自動化コントローラーのインストールが含まれます。PostgreSQL をインストールし、そのデータベースとして使用するように自動化コントローラーを設定します。これは、標準の自動化コントローラーのインストールシナリオとみなされます。

3.1.1. 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。
- 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。

3.1.2. 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. [バンドルのインストーラー]

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

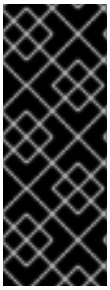
   b. [オンラインインストーラー]

   ```
   $ 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。以下の例に従ってください。

3.1.3. Red Hat Ansible Automation Platform の単一ノードインベントリーファイルの例

以下の例では、自動化コントローラーの単一ノードインストールにインベントリーファイルを追加する方法を説明します。



重要

- **pg_password** には特殊文字を使用しないでください。これにより、設定が失敗する場合があります。
- **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。

```
[automationcontroller]
controller.example.com (1)

[database]

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

pg_host=''
pg_port=''

pg_database='awx'
pg_username='awx'
pg_password='<password>'

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

(1) これは FQDN/IP として設定する必要があります。

3.1.4. 設定スクリプトフラグおよび追加変数

設定スクリプトを実行して自動化コントローラーをインストールする場合に、フラグおよび追加変数を指定することもできます。

表3.1 フラグ

引数	説明
-h	ヘルプメッセージを表示して終了します。
-i INVENTORY_FILE	Ansible インベントリーファイルへのパス (デフォルト: **inventory**)
-e EXTRA_VARS	追加の Ansible 変数を key=value または YAML/JSON として設定します。

引数	説明
-b	インストールの代わりにデータベースのバックアップを実行します。
-r	インストールの代わりにデータベースの復元を行います。
-k	SECRET_KEY を生成 **および** 配布します。

-- の区切り文字を使用して、適用する Ansible 引数を追加します。たとえば、**./setup.sh -i my_awesome_inventory.yml -e matburt_is_country_gold=True — -K** となります。

注記

- **-r** を渡してデータベース復元を実行する場合は、EXTRA_VARS でデフォルト以外のパスを指定しない限り、デフォルト復元パスが使用されます。復元パスを指定する EXTRA_VAR を渡す以下の例を参照してください。

```
./setup.sh -e 'restore_backup_file=/path/to/nondefault/location' -r
```

- **-e bundle_install=false** を渡すと、オンラインインストールを強制的に実行できます。

```
$ ./setup.sh -e bundle_install=false
```

表3.2 追加変数

変数	説明	デフォルト
upgrade_ansible_with_tower	自動化コントローラーのインストール時に、Ansible も最新の状態であることを確認します。	**False**
create_preload_data	Tower のインストール時に、デモ組織、プロジェクト、認証情報、ジョブテンプレートなども作成します。	**True**
bundle_install_folder	バンドルからインストールする場合に、バンドルされているリポジトリーを配置する場所	**var/lib/tower-bundle**
nginx_disable_https	nginx で HTTPS トラフィックを無効にします。HTTPS の負荷をロードバランサーに分散する場合に便利です。	**False**

変数	説明	デフォルト
nginx_disable_hsts	Web セキュリティーポリシーメカニズム HSTS を無効にします。	**False**
nginx_http_port	nginx が HTTP をリッスンするように設定するポート	**80**
nginx_https_port	nginx が HTTPS をリッスンするように設定するポート	**443**
backup_dir	バックアップ時に使用する一時的な場所	**/var/backups/tower/**
restore_backup_file	復元元として使用する別のバックアップファイルを指定します。	なし
required_ram	Tower のインストールに必要な最小メモリー (テストインストールの場合に限り変更してください)	**3750**
min_open_fds	表示ファイルの説明に使用できる最小リソース (テストインストールの場合に限り変更してください)	なし
ignore_preflight_errors	プリフライトチェックを無視します。これはテンプレートや他のシステム以外のイメージにインストールするときに便利です (**required_ram** および **min_open_fds** をオーバーライドします)。	**False**

例

- コアをアップグレードするには、以下を実行します。

```
./setup.sh -e upgrade_ansible_with_tower=1
```

- nginx で処理する https を無効化するには、以下を実行します。

```
./setup.sh -e nginx_disable_https=true
```

- バックアップファイルから復元時にデフォルトではないパスを指定する場合は、以下を実行します。

```
./setup.sh -e 'restore_backup_file=/path/to/nondefault/location' -r
```

3.1.5. 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 のインストールに必要なパラメーターを使用して **inventory** ファイルを更新したら、setup スクリプトを実行ができます。

手順

1. **setup.sh** スクリプトを実行します。

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

インストールが開始します。

3.1.6. 自動化コントローラーインストールの確認

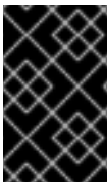
インストールが完了したら、**inventory** ファイルに挿入した管理者認証情報でログインして、自動化コントローラーが正常にインストールしたことを確認できます。

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

自動化コントローラーサーバーはポート 80 (**https://<TOWER_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、443 も利用できる状態に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 (https://access.redhat.com/) から Ansible までお問い合せください。

自動化コントローラーへのログインに成功すると、Red Hat Ansible Automation Platform 2.2 のインストールが完了します。

3.1.6.1. 追加の自動化コントローラーの設定とリソース

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3.3 自動化コントローラーを設定するための資料

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。

リンク	説明
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自動化コントローラー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	自動化コントローラーの機能をより詳細に確認します。

3.1.7. Ansible Automation Platform 2.2 の次のステップ

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 2.2 の新機能を活用し始めてみてください。

3.1.7.1. Ansible Automation Platform 2.2 へのデータの移行

プラットフォーム管理者が Ansible Automation Platform 2.2 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。

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

Ansible Automation Platform 2.2 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ー化されたイメージ) 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。詳細は、自動化実行環境へのアップグレードガイド および Ansible ビルダーガイド を参照してください。

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

Ansible Automation Platform 2.2 で使用するために Ansible Engine2.9 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。Ansible Builder を使用して実行環境を構築する方法の詳細は、Ansible Builder ガイド を参照してください。

3.1.7.1.3. 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。

3.1.7.2. 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については、**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については、Red Hat Ansible Automation Platform 化メッシュガイド を参照してください。

3.2. 外部管理データベースを備えた自動化コントローラーのインストール

以下の手順を使用して、リモート PostgreSQL インスタンスをデータベースとして通信するように設定された 1 台のマシンにスタンドアロンの自動化コントローラーサーバーをインストールできます。このリモート PostgreSQL は、管理するサーバーを使用することも、Amazon RDS などのクラウドサービスで提供することも可能です。

3.2.1. 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。
- 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。

3.2.2. 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. [バンドルのインストーラー]

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

   b. [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。以下の例に従ってください。

3.2.3. 外部管理データベースを使用するスタンドアロン自動化コントローラーのインベントリーファイル例

以下の例では、外部データベースを使用して自動化コントローラーのインストールをデプロイするためのインベントリーファイルの設定方法を説明します。

重要

- **pg_password** には特殊文字を使用しないでください。これにより、設定が失敗する場合があります。
- **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。

```
[automationcontroller]
controller.example.com (1)


[database]
database.example.com

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

pg_host='database.example.com'
pg_port='5432'

pg_database='awx'
pg_username='awx'

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

(1) これは FQDN/IP として設定する必要があります。

3.2.4. 設定スクリプトフラグおよび追加変数

設定スクリプトを実行して自動化コントローラーをインストールする場合に、フラグおよび追加変数を指定することもできます。

表3.4 フラグ

引数	説明
-h	ヘルプメッセージを表示して終了します。

引数	説明
-i INVENTORY_FILE	Ansible インベントリーファイルへのパス (デフォルト: **inventory**)
-e EXTRA_VARS	追加の Ansible 変数を key=value または YAML/JSON として設定します。
-b	インストールの代わりにデータベースのバックアップを実行します。
-r	インストールの代わりにデータベースの復元を行います。
-k	SECRET_KEY を生成 **および** 配布します。

-- の区切り文字を使用して、適用する Ansible 引数を追加します。たとえば、**./setup.sh -i my_awesome_inventory.yml -e matburt_is_country_gold=True — -K** となります。

注記

- **-r** を渡してデータベース復元を実行する場合は、EXTRA_VARS でデフォルト以外のパスを指定しない限り、デフォルト復元パスが使用されます。復元パスを指定する EXTRA_VAR を渡す以下の例を参照してください。

```
./setup.sh -e 'restore_backup_file=/path/to/nondefault/location' -r
```

- **-e bundle_install=false** を渡すと、オンラインインストールを強制的に実行できます。

```
$ ./setup.sh -e bundle_install=false
```

表3.5 追加変数

変数	説明	デフォルト
upgrade_ansible_with_tower	自動化コントローラーのインストール時に、Ansible も最新の状態であることを確認します。	**False**
create_preload_data	Tower のインストール時に、デモ組織、プロジェクト、認証情報、ジョブテンプレートなども作成します。	**True**
bundle_install_folder	バンドルからインストールする場合に、バンドルされているリポジトリーを配置する場所	**var/lib/tower-bundle**

変数	説明	デフォルト
nginx_disable_https	nginx で HTTPS トラフィックを無効にします。HTTPS の負荷をロードバランサーに分散する場合に便利です。	**False**
nginx_disable_hsts	Web セキュリティーポリシーメカニズム HSTS を無効にします。	**False**
nginx_http_port	nginx が HTTP をリッスンするように設定するポート	**80**
nginx_https_port	nginx が HTTPS をリッスンするように設定するポート	**443**
backup_dir	バックアップ時に使用する一時的な場所	**/var/backups/tower/**
restore_backup_file	復元元として使用する別のバックアップファイルを指定します。	なし
required_ram	Tower のインストールに必要な最小メモリー (テストインストールの場合に限り変更してください)	**3750**
min_open_fds	表示ファイルの説明に使用できる最小リソース (テストインストールの場合に限り変更してください)	なし
ignore_preflight_errors	プリフライトチェックを無視します。これはテンプレートや他のシステム以外のイメージにインストールするときに便利です (**required_ram** および **min_open_fds** をオーバーライドします)。	**False**

例

- コアをアップグレードするには、以下を実行します。

```
./setup.sh -e upgrade_ansible_with_tower=1
```

- nginx で処理する https を無効化するには、以下を実行します。

```
./setup.sh -e nginx_disable_https=true
```

- バックアップファイルから復元時にデフォルトではないパスを指定する場合は、以下を実行します。

```
./setup.sh -e 'restore_backup_file=/path/to/nondefault/location' -r
```

3.2.5. 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 のインストールに必要なパラメーターを使用して **inventory** ファイルを更新したら、setup スクリプトを実行ができます。

手順

1. **setup.sh** スクリプトを実行します。

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

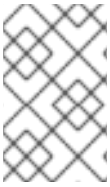
インストールが開始します。

3.2.6. 自動化コントローラーインストールの確認

インストールが完了したら、**inventory** ファイルに挿入した管理者認証情報でログインして、自動化コントローラーが正常にインストールしたことを確認できます。

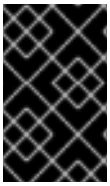
手順

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

自動化コントローラーサーバーはポート 80 (https://<TOWER_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、443 も利用できる状態に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 (https://access.redhat.com/) から Ansible までお問い合せください。

自動化コントローラーへのログインに成功すると、Red Hat Ansible Automation Platform 2.2 のインストールが完了します。

3.2.6.1. 追加の自動化コントローラーの設定とリソース

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3.6 自動化コントローラーを設定するための資料

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自動化コントローラー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	自動化コントローラーの機能をより詳細に確認します。

3.2.7. Ansible Automation Platform 2.2 の次のステップ

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 2.2 の新機能を活用し始めてみてください。

3.2.7.1. Ansible Automation Platform 2.2 へのデータの移行

プラットフォーム管理者が Ansible Automation Platform 2.2 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。

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

Ansible Automation Platform 2.2 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ー化されたイメージ) 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。詳細は、自動化実行環境へのアップグレードガイド および Ansible ビルダーガイド を参照してください。

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

Ansible Automation Platform 2.2 で使用するために Ansible Engine2.9 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。Ansible Builder を使用して実行環境を構築する方法の詳細は、Ansible Builder ガイド を参照してください。

3.2.7.1.3. 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。

3.2.7.2. 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については、**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については、Red Hat Ansible Automation Platform 化メッシュガイド を参照してください。

3.3. データベースを使用した同一ノードへの AUTOMATION HUB のインストール

これらの手順に従って、同じノード上のデータベース、またはインストーラー以外が管理する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インストールできます。

3.3.1. 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。
- 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。

3.3.2. 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. [バンドルのインストーラー]

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

   b. [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。以下の例に従ってください。

3.3.3. スタンドアロンの Automation Hub のインベントリーファイルの例

以下の例では、Automation Hub のスタンドアロンインスタンスをデプロイするためにインベントリーファイルを設定する方法を説明します。

重要

- 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。自動化コントローラーと Automation Hub を同じノードにインストールすることはできません。
- **[automationhub]** 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) を指定して、ユーザーが別のノードから自動化ハブ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。
- **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。

```
[automationcontroller]


[automationhub]
127.0.0.1 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=''

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
```

3.3.4. 設定スクリプトフラグおよび追加変数

設定スクリプトを実行して自動化コントローラーをインストールする場合に、フラグおよび追加変数を指定することもできます。

表3.7 フラグ

引数	説明
-h	ヘルプメッセージを表示して終了します。
-i INVENTORY_FILE	Ansible インベントリーファイルへのパス (デフォルト: **inventory**)
-e EXTRA_VARS	追加の Ansible 変数を key=value または YAML/JSON として設定します。
-b	インストールの代わりにデータベースのバックアップを実行します。
-r	インストールの代わりにデータベースの復元を行います。
-k	SECRET_KEY を生成 **および** 配布します。

-- の区切り文字を使用して、適用する Ansible 引数を追加します。たとえば、**./setup.sh -i my_awesome_inventory.yml -e matburt_is_country_gold=True — -K** となります。

注記

- **-r** を渡してデータベース復元を実行する場合は、EXTRA_VARS でデフォルト以外のパスを指定しない限り、デフォルト復元パスが使用されます。復元パスを指定する EXTRA_VAR を渡す以下の例を参照してください。

```
./setup.sh -e 'restore_backup_file=/path/to/nondefault/location' -r
```

- **-e bundle_install=false** を渡すと、オンラインインストールを強制的に実行できます。

```
$ ./setup.sh -e bundle_install=false
```

表3.8 追加変数

変数	説明	デフォルト
upgrade_ansible_with_tower	自動化コントローラーのインストール時に、Ansible も最新の状態であることを確認します。	**False**
create_preload_data	Tower のインストール時に、デモ組織、プロジェクト、認証情報、ジョブテンプレートなども作成します。	**True**
bundle_install_folder	バンドルからインストールする場合に、バンドルされているリポジトリーを配置する場所	**var/lib/tower-bundle**
nginx_disable_https	nginx で HTTPS トラフィックを無効にします。HTTPS の負荷をロードバランサーに分散する場合に便利です。	**False**
nginx_disable_hsts	Web セキュリティーポリシーメカニズム HSTS を無効にします。	**False**
nginx_http_port	nginx が HTTP をリッスンするように設定するポート	**80**
nginx_https_port	nginx が HTTPS をリッスンするように設定するポート	**443**
backup_dir	バックアップ時に使用する一時的な場所	**/var/backups/tower/**
restore_backup_file	復元元として使用する別のバックアップファイルを指定します。	なし
required_ram	Tower のインストールに必要な最小メモリー (テストインストールの場合に限り変更してください)	**3750**
min_open_fds	表示ファイルの説明に使用できる最小リソース (テストインストールの場合に限り変更してください)	なし

変数	説明	デフォルト
ignore_preflight_errors	プリフライトチェックを無視します。これはテンプレートや他のシステム以外のイメージにインストールするときに便利です (**required_ram** および **min_open_fds** をオーバーライドします)。	**False**

例

- コアをアップグレードするには、以下を実行します。

```
./setup.sh -e upgrade_ansible_with_tower=1
```

- nginx で処理する https を無効化するには、以下を実行します。

```
./setup.sh -e nginx_disable_https=true
```

- バックアップファイルから復元時にデフォルトではないパスを指定する場合は、以下を実行します。

```
./setup.sh -e 'restore_backup_file=/path/to/nondefault/location' -r
```

3.3.5. 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 のインストールに必要なパラメーターを使用して **inventory** ファイルを更新したら、setup スクリプトを実行ができます。

手順

1. **setup.sh** スクリプトを実行します。

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

インストールが開始します。

3.3.6. Automation Hub のインストールの確認

インストールが完了したら、**inventory** ファイルに挿入した管理者認証情報でログインして、自動化ハブが正常にインストールしたことを確認できます。

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 (https://access.redhat.com/) から Ansible までお問い合せください。

Automation Hub へのログインが成功すると、Red Hat Ansible Automation Platform 2.2 のインストールが完了します。

3.3.6.1. 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3.9 自動化コントローラーを設定するための資料

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 認定コレクションおよび Ansible Galaxy コレクションの管理	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	Automation Hub で社内で開発したコレクションを公開します。

3.3.7. Ansible Automation Platform 2.2 の次のステップ

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 2.2 の新機能を活用し始めてみてください。

3.3.7.1. Ansible Automation Platform 2.2 へのデータの移行

プラットフォーム管理者が Ansible Automation Platform 2.2 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。

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

Ansible Automation Platform 2.2 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ー化されたイメージ) 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。詳細は、自動化実行環境へのアップグレードガイド および Ansible ビルダーガイド を参照してください。

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

Ansible Automation Platform 2.2 で使用するために Ansible Engine2.9 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。Ansible Builder を使用して実行環境を構築する方法の詳細は、Ansible Builder ガイド を参照してください。

3.3.7.1.3. 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。

3.3.7.2. 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については、**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については、Red Hat Ansible Automation Platform 化メッシュガイド を参照してください。

3.4. 外部データベースを使用した AUTOMATION HUB のインストール

以下の手順に従って、外部の管理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インストールできます。これにより、Automation Hub サーバーが単一のマシンにインストールされ、Ansible Automation Platform インストーラーを使用してリモート PostgreSQL データベースがインストールされます。

3.4.1. 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。
- 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。

3.4.2. 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. [バンドルのインストーラー]

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

   b. [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。以下の例に従ってください。

3.4.3. スタンドアロンの Automation Hub のインベントリーファイルの例

以下の例では、Automation Hub のスタンドアロンインスタンスをデプロイするためにインベントリーファイルを設定する方法を説明します。

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。自動化コントローラーと Automation Hub を同じノードにインストールすることはできません。
- **[automationhub]** 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) を指定して、ユーザーが別のノードから自動化ハブ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。
- **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。

```
[automationcontroller]


[automationhub]
127.0.0.1 ansible_connection=local

[database]
host2

[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=''
```

```
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
```

3.4.4. 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。

3.4.4.1. 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。

前提条件

- システムが Red Hat Ansible Automation Platform 2.2.1 以降を実行している。
- Private Automation Hub 4.5.2 以降を使用している。

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。

3.4.4.2. 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。

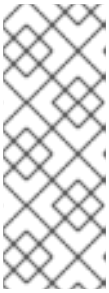
```
#ldapextras.yml
---
ldap_extra_settings:
  AUTH_LDAP_DENY_GROUP: 'cn=pah-nosoupforyou,ou=groups,dc=example,dc=com'
...
```

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。

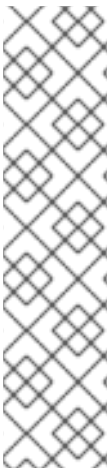
3.4.5. 設定スクリプトフラグおよび追加変数

設定スクリプトを実行して自動化コントローラーをインストールする場合に、フラグおよび追加変数を指定することもできます。

表3.10 フラグ

引数	説明
-h	ヘルプメッセージを表示して終了します。
-i INVENTORY_FILE	Ansible インベントリーファイルへのパス (デフォルト: **inventory**)
-e EXTRA_VARS	追加の Ansible 変数を key=value または YAML/JSON として設定します。
-b	インストールの代わりにデータベースのバックアップを実行します。
-r	インストールの代わりにデータベースの復元を行います。
-k	SECRET_KEY を生成 **および** 配布します。

-- の区切り文字を使用して、適用する Ansible 引数を追加します。たとえば、**./setup.sh -i my_awesome_inventory.yml -e matburt_is_country_gold=True — -K** となります。



注記

- **-r** を渡してデータベース復元を実行する場合は、EXTRA_VARS でデフォルト以外のパスを指定しない限り、デフォルト復元パスが使用されます。復元パスを指定する EXTRA_VAR を渡す以下の例を参照してください。

```
./setup.sh -e 'restore_backup_file=/path/to/nondefault/location' -r
```

- **-e bundle_install=false** を渡すと、オンラインインストールを強制的に実行できます。

```
$ ./setup.sh -e bundle_install=false
```

表3.11 追加変数

変数	説明	デフォルト
upgrade_ansible_with_tower	自動化コントローラーのインストール時に、Ansible も最新の状態であることを確認します。	**False**
create_preload_data	Tower のインストール時に、デモ組織、プロジェクト、認証情報、ジョブテンプレートなども作成します。	**True**

変数	説明	デフォルト
bundle_install_folder	バンドルからインストールする場合に、バンドルされているリポジトリーを配置する場所	**var/lib/tower-bundle**
nginx_disable_https	nginx で HTTPS トラフィックを無効にします。HTTPS の負荷をロードバランサーに分散する場合に便利です。	**False**
nginx_disable_hsts	Web セキュリティーポリシーメカニズム HSTS を無効にします。	**False**
nginx_http_port	nginx が HTTP をリッスンするように設定するポート	**80**
nginx_https_port	nginx が HTTPS をリッスンするように設定するポート	**443**
backup_dir	バックアップ時に使用する一時的な場所	**/var/backups/tower/**
restore_backup_file	復元元として使用する別のバックアップファイルを指定します。	なし
required_ram	Tower のインストールに必要な最小メモリー (テストインストールの場合に限り変更してください)	**3750**
min_open_fds	表示ファイルの説明に使用できる最小リソース (テストインストールの場合に限り変更してください)	なし
ignore_preflight_errors	プリフライトチェックを無視します。これはテンプレートや他のシステム以外のイメージにインストールするときに便利です (**required_ram** および **min_open_fds** をオーバーライドします)。	**False**

例

- コアをアップグレードするには、以下を実行します。

```
./setup.sh -e upgrade_ansible_with_tower=1
```

-

- nginx で処理する https を無効化するには、以下を実行します。

```
./setup.sh -e nginx_disable_https=true
```

- バックアップファイルから復元時にデフォルトではないパスを指定する場合は、以下を実行します。

```
./setup.sh -e 'restore_backup_file=/path/to/nondefault/location' -r
```

3.4.6. 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 のインストールに必要なパラメーターを使用して **inventory** ファイルを更新したら、setup スクリプトを実行ができます。

手順

1. **setup.sh** スクリプトを実行します。

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

インストールが開始します。

3.4.7. 自動化コントローラーインストールの確認

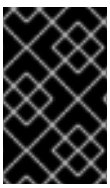
インストールが完了したら、**inventory** ファイルに挿入した管理者認証情報でログインして、自動化コントローラーが正常にインストールしたことを確認できます。

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

自動化コントローラーサーバーはポート 80 (**https://<TOWER_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、443 も利用できる状態に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 (https://access.redhat.com/) から Ansible までお問い合せください。

自動化コントローラーへのログインに成功すると、Red Hat Ansible Automation Platform 2.2 のインストールが完了します。

3.4.7.1. 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3.12 自動化コントローラーを設定するための資料

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 認定コレクションおよび Ansible Galaxy コレクションの管理	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	Automation Hub で社内で開発したコレクションを公開します。

3.4.8. Ansible Automation Platform 2.2 の次のステップ

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 2.2 の新機能を活用し始めてみてください。

3.4.8.1. Ansible Automation Platform 2.2 へのデータの移行

プラットフォーム管理者が Ansible Automation Platform 2.2 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。

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

Ansible Automation Platform 2.2 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ー化されたイメージ) 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。詳細は、自動化実行環境へのアップグレードガイド および Ansible ビルダーガイド を参照してください。

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

Ansible Automation Platform 2.2 で使用するために Ansible Engine2.9 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。Ansible Builder を使用して実行環境を構築する方法の詳細は、Ansible Builder ガイド を参照してください。

3.4.8.1.3. 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。

3.4.8.2. 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。

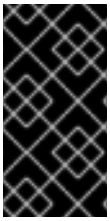
バージョン 1.x から最新バージョンの Ansible Automation Platform にアップグレードする場合は、データをレガシーの分離ノードから自動化メッシュに必要な実行ノードに移行する必要があります。ハイブリッドノードとコントロールノードのネットワークを計画して、Ansible Automation Platform インストーラーにあるインベントリーファイルを編集して、メッシュ関連の値を各実行ノードに割り当てることで、自動化メッシュを実装できます。

分離ノードから実行ノードに移行する方法については、**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については、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 マルチマシンクラスターのインストール

外部管理データベースを使用して、Automation Hub を使用して、クラスター化された自動化コントローラーとして Ansible Automation Platform をインストールできます。このモードでは、複数の自動化コントローラーノードがインストールされ、アクティブになります。任意のノードは HTTP 要求を受け取ることができ、すべてのノードがジョブを実行することができます。これにより、Ansible Automation Platform サーバーがクラスターにインストールされ、データベースとして PostgreSQL のリモートインスタンスと対話するように設定します。このリモート PostgreSQL は、管理するサーバーを使用することも、Amazon RDS などのクラウドサービスで提供することも可能です。



重要

Ansible Automation Platform インストーラーでは、インベントリーごとに自動化ハブを **1つのみ** デプロイできます。Automation Hub のスタンドアロンインスタンスには Ansible Automation Platform インストーラーを使用し、任意の数の異なるインベントリーでインストーラーを実行して、複数の Automation Hub をデプロイできます。

4.1. 外部管理データベースを使用した複数ノードの RED HAT ANSIBLE AUTOMATION PLATFORM のインストール

この手順に従い、外部管理データベースを使用して、Red Hat Ansible Automation Platform を複数の自動化コントローラーノードおよび Automation Hub としてインストールできます。

4.1.1. 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。
- 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。

4.1.2. 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. [バンドルのインストーラー]

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

   b. [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。以下の例に従ってください。

4.1.3. Red Hat Ansible Automation Platform の複数ノードのインベントリーファイル例

以下の例では、自動化コントローラーのマルチノードクラスターインストールのインベントリーファイルを追加する方法を説明します。

重要

- 自動化コントローラーと Automation Hub を同じノードにインストールすることはできません。
- **[automationhub]** ホストの到達可能な IP アドレスを指定して、ユーザーが別のノードから Private Automation Hub のコンテンツを同期できるようにします。
- **pg_password** には特殊文字を使用しないでください。これにより、設定が失敗する場合があります。
- **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。

```
[automationcontroller]
host1
host11
host12


[automationhub]
host2

[database]
(1)

[all:vars]
ansible_become=true

admin_password='<password>'

pg_host='dbnode.example.com'
pg_port='5432'

pg_database='tower'
pg_username='tower'
pg_password='<password>'

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

(1) フィールドは空でなければなりません。

4.1.4. 設定スクリプトフラグおよび追加変数

設定スクリプトを実行して自動化コントローラーをインストールする場合に、フラグおよび追加変数を指定することもできます。

表4.1 フラグ

引数	説明
-h	ヘルプメッセージを表示して終了します。
-i INVENTORY_FILE	Ansible インベントリーファイルへのパス (デフォルト: **inventory**)
-e EXTRA_VARS	追加の Ansible 変数を key=value または YAML/JSON として設定します。
-b	インストールの代わりにデータベースのバックアップを実行します。
-r	インストールの代わりにデータベースの復元を行います。
-k	SECRET_KEY を生成 **および** 配布します。

-- の区切り文字を使用して、適用する Ansible 引数を追加します。たとえば、 **./setup.sh -i my_awesome_inventory.yml -e matburt_is_country_gold=True — -K** となります。

注記

- **-r** を渡してデータベース復元を実行する場合は、EXTRA_VARS でデフォルト以外のパスを指定しない限り、デフォルト復元パスが使用されます。復元パスを指定する EXTRA_VAR を渡す以下の例を参照してください。

```
./setup.sh -e 'restore_backup_file=/path/to/nondefault/location' -r
```

- **-e bundle_install=false** を渡すと、オンラインインストールを強制的に実行できます。

```
$ ./setup.sh -e bundle_install=false
```

表4.2 追加変数

変数	説明	デフォルト
upgrade_ansible_with_tower	自動化コントローラーのインストール時に、Ansible も最新の状態であることを確認します。	**False**

変数	説明	デフォルト
create_preload_data	Tower のインストール時に、デモ組織、プロジェクト、認証情報、ジョブテンプレートなども作成します。	**True**
bundle_install_folder	バンドルからインストールする場合に、バンドルされているリポジトリーを配置する場所	**var/lib/tower-bundle**
nginx_disable_https	nginx で HTTPS トラフィックを無効にします。HTTPS の負荷をロードバランサーに分散する場合に便利です。	**False**
nginx_disable_hsts	Web セキュリティーポリシーメカニズム HSTS を無効にします。	**False**
nginx_http_port	nginx が HTTP をリッスンするように設定するポート	**80**
nginx_https_port	nginx が HTTPS をリッスンするように設定するポート	**443**
backup_dir	バックアップ時に使用する一時的な場所	**/var/backups/tower/**
restore_backup_file	復元元として使用する別のバックアップファイルを指定します。	なし
required_ram	Tower のインストールに必要な最小メモリー (テストインストールの場合に限り変更してください)	**3750**
min_open_fds	表示ファイルの説明に使用できる最小リソース (テストインストールの場合に限り変更してください)	なし
ignore_preflight_errors	プリフライトチェックを無視します。これはテンプレートや他のシステム以外のイメージにインストールするときに便利です (**required_ram** および **min_open_fds** をオーバーライドします)。	**False**

例

- コアをアップグレードするには、以下を実行します。

```
./setup.sh -e upgrade_ansible_with_tower=1
```

- nginx で処理する https を無効化するには、以下を実行します。

```
./setup.sh -e nginx_disable_https=true
```

- バックアップファイルから復元時にデフォルトではないパスを指定する場合は、以下を実行します。

```
./setup.sh -e 'restore_backup_file=/path/to/nondefault/location' -r
```

4.1.5. 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 のインストールに必要なパラメーターを使用して **inventory** ファイルを更新したら、setup スクリプトを実行ができます。

手順

1. **setup.sh** スクリプトを実行します。

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

インストールが開始します。

4.1.6. 自動化コントローラーインストールの確認

インストールが完了したら、**inventory** ファイルに挿入した管理者認証情報でログインして、自動化コントローラーが正常にインストールしたことを確認できます。

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

自動化コントローラーサーバーはポート 80 (**https://<TOWER_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、443 も利用できる状態に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 (https://access.redhat.com/) から Ansible までお問い合せください。

自動化コントローラーへのログインに成功すると、Red Hat Ansible Automation Platform 2.2 のインストールが完了します。

4.1.6.1. 追加の自動化コントローラーの設定とリソース

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4.3 自動化コントローラーを設定するための資料

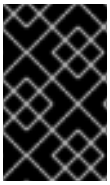
リンク	説明
Automation Controller クイックセットアップガイド	Automation Controller を設定して最初の Playbook を実行します。
Automation Controller 管理ガイド	カスタマースクリプト、管理ジョブなどを使用して Automation Controller の管理を設定します。
Red Hat Ansible Automation Platform のプロキシーサポートの設定	プロキシーサーバーを使用して Automation Controller を設定します。
ユーザビリティーアナリティクスおよび自動化コントローラーか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	自動化コントローラーの機能をより詳細に確認します。

4.1.7. Automation Hub のインストールの確認

インストールが完了したら、**inventory** ファイルに挿入した管理者認証情報でログインして、自動化ハブが正常にインストールしたことを確認できます。

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 (https://access.redhat.com/) から Ansible までお問い合せください。

Automation Hub へのログインが成功すると、Red Hat Ansible Automation Platform 2.2 のインストールが完了します。

4.1.7.1. 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4.4 自動化コントローラーを設定するための資料

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 認定コレクションおよび Ansible Galaxy コレクションの管理	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	Automation Hub で社内で開発したコレクションを公開します。

4.1.8. Ansible Automation Platform 2.2 の次のステップ

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 2.2 の新機能を活用し始めてみてください。

4.1.8.1. Ansible Automation Platform 2.2 へのデータの移行

プラットフォーム管理者が Ansible Automation Platform 2.2 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。

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

Ansible Automation Platform 2.2 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ー化されたイメージ) 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。詳細は、自動化実行環境へのアップグレードガイド および Ansible ビルダーガイド を参照してください。

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

Ansible Automation Platform 2.2 で使用するために Ansible Engine2.9 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。Ansible Builder を使用して実行環境を構築する方法の詳細は、Ansible Builder ガイド を参照してください。

4.1.8.1.3. 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。

4.1.8.2. 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については、**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については、Red Hat Ansible Automation Platform 化メッシュガイド を参照してください。

第5章 RED HAT ANSIBLE AUTOMATION PLATFORM のプロキシーサポートの設定

プロキシーを使用してトラフィックと通信できるように、Red Hat Ansible Automation Platform を設定できます。プロキシーサーバーは、リソースを別のサーバーから求めているクライアントが出した要求を仲介する役割を果たします。クライアントは、プロキシーサーバーに接続して、別のサーバーからサービスや利用可能なリソースを要求します。そして、このプロキシーサーバーは複雑な内容を簡素化して制御する方法の 1 つとして、その要求を評価します。次のセクションでは、サポート対象のプロキシー設定とその設定方法について説明します。

5.1. プロキシーサポートの有効化

プロキシーサーバーをサポートするために、Automation Controller は、Automation Controller 設定の **REMOTE_HOST_HEADERS** リスト変数を介してプロキシーされた要求 (Automation Controllerの前にある ALB、NLB、HAProxy、Squid、Nginx、tinyproxy など) を処理します。デフォルトでは、**REMOTE_HOST_HEADERS** は **["REMOTE_ADDR", "REMOTE_HOST"]** に設定されています。

プロキシーサーバーのサポートを有効にするには、Automation Controller の設定ページで **REMOTE_HOST_HEADERS** フィールドを編集します。

手順

1. Automation Controller で、**Settings → Miscellaneous System** に移動します。
2. **REMOTE_HOST_HEADERS** フィールドに、次の値を入力します。

```
[
  "HTTP_X_FORWARDED_FOR",
  "REMOTE_ADDR",
  "REMOTE_HOST"
]
```

Automation Controller はリモートホストの IP アドレスを判断するために、最初の IP アドレスが特定されるまで、**REMOTE_HOST_HEADERS** のヘッダー一覧を検索します。

5.2. 既知のプロキシー

オートメーションコントローラーを **REMOTE_HOST_HEADERS = ['HTTP_X_FORWARDED_FOR', 'REMOTE_ADDR', 'REMOTE_HOST']** で設定している場合は、**X-Forwarded-For** の値が、オートメーションコントローラーの前にあるプロキシーまたはロードバランサーから送られていることを前提としています。プロキシー/ロードバランサーを使用せずにオートメーションコントローラーに到達できる場合、またはプロキシーがヘッダーを検証しない場合は、**X-Forwarded-For** の値が偽造されて発信元の IP アドレスを偽装する可能性があります。**HTTP_X_FORWARDED_FOR** 設定で **REMOTE_HOST_HEADERS** を使用すると、脆弱性が発生します。

これを回避するには、Automation Controller の設定メニューの **PROXY_IP_ALLOWED_LIST** フィールドを使用して許可される既知のプロキシーのリストを設定できます。既知のプロキシー一覧に含まれていないロードバランサーおよびホストは、要求を拒否します。

5.2.1. 既知のプロキシーの設定

Automation Controller の既知のプロキシーのリストを設定するには、Automation Controller の設定ページの **PROXY_IP_ALLOWED_LIST** フィールドにプロキシー IP アドレスを追加します。

手順

1. Automation Controller で、**Settings → Miscellaneous System** に移動します。
2. **PROXY_IP_ALLOWED_LIST** フィールドに、以下の例の構文に従って、Automation Controller への接続を許可する IP アドレスを入力します。

PROXY_IP_ALLOWED_LIST エントリーの例

```
[
  "example1.proxy.com:8080",
  "example2.proxy.com:8080"
]
```

重要

- **PROXY_IP_ALLOWED_LIST** は、この一覧のプロキシーが適切にヘッダー入力をサニタイズし、**X-Forwarded-For** の値がクライアントの実際のソース IP と同等になるように正しく設定します。Automation Controller は、**PROXY_IP_ALLOWED_LIST** の IP アドレスとホスト名に依存して、**X-Forwarded-For** フィールドに偽装されていない値を提供できます。
- 以下の条件が **すべて** 満たされない限り **HTTP_X_FORWARDED_FOR** を 'REMOTE_HOST_HEADERS' のアイテムとして設定しないでください。
  - SSL Termination でプロキシー環境を使用している
  - プロキシーにより **X-Forwarded-For** ヘッダーのサニタイズまたは検証が行われクライアントの攻撃を防止することができる
  - **/etc/tower/conf.d/remote_host_headers.py** が信頼されたプロキシーまたはロードバランサーの送信元 IP のみを含む **PROXY_IP_ALLOWED_LIST** を定義している

5.3. リバースプロキシーの設定

Automation Controller 設定の **REMOTE_HOST_HEADERS** フィールドに **HTTP_X_FORWARDED_FOR** を追加して、リバースプロキシーサーバー設定をサポートできます。**X-Forwarded-For** (XFF) HTTP ヘッダーフィールドは、HTTP プロキシーまたはロードバランサー経由で Web サーバーに接続するクライアントの送信元 IP アドレスを識別します。

手順

1. Automation Controller で、**Settings → Miscellaneous System** に移動します。
2. **REMOTE_HOST_HEADERS** フィールドに、次の値を入力します。

```
[
  "HTTP_X_FORWARDED_FOR",
  "REMOTE_ADDR",
  "REMOTE_HOST"
]
```

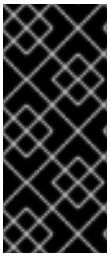
第6章 AUTOMATION CONTROLLER WEBSOCKET 接続の設定

Websocket の設定を nginx またはロードバランサー設定に合わせるために、Automation Controller を設定できます。

6.1. コントローラーの自動化用の WEBSOCKET 設定

自動化コントローラーノードは、websocket を介して他のすべての自動化コントローラーノードに接続されます。この相互接続では、WebSocket が出力されたメッセージをすべて他の自動化コントローラーノードに分散するために使用されます。これは、任意のブラウザークライアントの WebSocket が、どの自動化コントローラーノードで実行している可能性のあるジョブにサブスクライブできるためです。WebSocket クライアントは特定の自動化コントローラーノードにルーティングされません。すべての自動化コントローラーノードは、任意の Websocket 要求を処理でき、各自動化コントローラーノードは、すべてのクライアント宛てのすべての Websocket メッセージを把握しておく必要があります。

自動化コントローラーは、データベースのインスタンスレコードを使用して、他の自動化コントローラーノードの検出を自動的に処理します。



重要

- (オープンインターネットではなく) ノードがプライベートで信頼されるサブネットに Websocket トラフィックをブロードキャストしていることが意図されています。そのため、Websocket ブロードキャストの HTTPS をオフにすると、Ansible Playbook の標準出力 (stdout) の大部分で構成される Websocket トラフィックは、暗号化されない自動化コントローラーノード間で送信されます。

6.1.1. 他の自動化コントローラーノードの自動検出の設定

Websocket 接続を設定して、自動化コントローラーがデータベースのインスタンスレコードを使用して他の自動化コントローラーノードの検出を自動的に処理できるようにします。

- ポート、プロトコル、および Websocket 接続の確立時に証明書を検証するかどうかについて、自動化コントローラー Websocket 情報を編集します。

```
BROADCAST_WEBSOCKET_PROTOCOL = 'http'
BROADCAST_WEBSOCKET_PORT = 80
BROADCAST_WEBSOCKET_VERIFY_CERT = False
```

第7章 ユーザビリティーアナリティクスおよび自動化コントローラーからのデータ収集の管理

Automation Controller のユーザーインターフェイスをオプトアウトまたは変更することで、Automation Controller からユーザビリティーアナリティクスおよびデータ収集への参加方法を変更できます。

7.1. ユーザビリティーアナリティクスおよびデータ収集

ユーザビリティーのデータ収集は、Automation Controller に含まれており、Automation Controller ユーザーが Automation Controller とどのように相互作用するかをよりよく理解するためのデータを収集し、今後のリリースの強化に役立て、ユーザーエクスペリエンスの合理化を継続していきます。

Automation Controller のトライアルまたは Automation Controller の新規インストールのみが、このデータ収集でオプトインされます。

関連情報

- 詳細は、Red Hat プライバシーポリシー を参照してください。

7.1.1. 自動化コントローラーからのデータ収集の制御

設定メニューの **ユーザーインターフェイス** タブで参加レベルを設定して、自動化コントローラーがデータを収集する方法を制御できます。

手順

1. 自動化コントローラーにログインします。
2. **Settings → User Interface** に移動します。
3. アナリティクストラッキングの状態ドロップダウンリストから希望のデータ収集レベルを選択します。
    a. **オフ**: データ収集を行いません。
    b. **匿名**: ユーザー固有のデータを含めないデータ収集を有効化します。
    c. **詳細**: お使いのユーザー固有のデータを含めたデータ収集を有効化します。
4. **保存** をクリックして設定を適用するか、**キャンセル** をクリックして変更を破棄します。

第8章 サポート対象のインベントリープラグインテンプレート

アップグレード時に、既存の設定は、後方互換性のあるインベントリー出力を生成する新しい形式に変換されます。以下のテンプレートを使用して、インベントリーを新しいスタイルのインベントリープラグイン出力に移行するのに役に立ちます。

8.1. AMAZON WEB SERVICES EC2

```
compose:
  ansible_host: public_ip_address
  ec2_account_id: owner_id
  ec2_ami_launch_index: ami_launch_index | string
  ec2_architecture: architecture
  ec2_block_devices: dict(block_device_mappings | map(attribute='device_name') | list |
zip(block_device_mappings | map(attribute='ebs.volume_id') | list))
  ec2_client_token: client_token
  ec2_dns_name: public_dns_name
  ec2_ebs_optimized: ebs_optimized
  ec2_eventsSet: events | default("")
  ec2_group_name: placement.group_name
  ec2_hypervisor: hypervisor
  ec2_id: instance_id
  ec2_image_id: image_id
  ec2_instance_profile: iam_instance_profile | default("")
  ec2_instance_type: instance_type
  ec2_ip_address: public_ip_address
  ec2_kernel: kernel_id | default("")
  ec2_key_name: key_name
  ec2_launch_time: launch_time | regex_replace(" ", "T") | regex_replace("(\+)(\d\d):(\d)(\d)$",
".\g<2>\g<3>Z")
  ec2_monitored: monitoring.state in ['enabled', 'pending']
  ec2_monitoring_state: monitoring.state
  ec2_persistent: persistent | default(false)
  ec2_placement: placement.availability_zone
  ec2_platform: platform | default("")
  ec2_private_dns_name: private_dns_name
  ec2_private_ip_address: private_ip_address
  ec2_public_dns_name: public_dns_name
  ec2_ramdisk: ramdisk_id | default("")
  ec2_reason: state_transition_reason
  ec2_region: placement.region
  ec2_requester_id: requester_id | default("")
  ec2_root_device_name: root_device_name
  ec2_root_device_type: root_device_type
  ec2_security_group_ids: security_groups | map(attribute='group_id') | list |  join(',')
  ec2_security_group_names: security_groups | map(attribute='group_name') | list |  join(',')
  ec2_sourceDestCheck: source_dest_check | default(false) | lower | string
  ec2_spot_instance_request_id: spot_instance_request_id | default("")
  ec2_state: state.name
  ec2_state_code: state.code
  ec2_state_reason: state_reason.message if state_reason is defined else ""
  ec2_subnet_id: subnet_id | default("")
  ec2_tag_Name: tags.Name
  ec2_virtualization_type: virtualization_type
  ec2_vpc_id: vpc_id | default("")
```

```
filters:
  instance-state-name:
  - running
groups:
  ec2: true
hostnames:
  - network-interface.addresses.association.public-ip
  - dns-name
  - private-dns-name
keyed_groups:
  - key: image_id | regex_replace("[^A-Za-z0-9\_]", "_")
    parent_group: images
    prefix: ''
    separator: ''
  - key: placement.availability_zone
    parent_group: zones
    prefix: ''
    separator: ''
  - key: ec2_account_id | regex_replace("[^A-Za-z0-9\_]", "_")
    parent_group: accounts
    prefix: ''
    separator: ''
  - key: ec2_state | regex_replace("[^A-Za-z0-9\_]", "_")
    parent_group: instance_states
    prefix: instance_state
  - key: platform | default("undefined") | regex_replace("[^A-Za-z0-9\_]", "_")
    parent_group: platforms
    prefix: platform
  - key: instance_type | regex_replace("[^A-Za-z0-9\_]", "_")
    parent_group: types
    prefix: type
  - key: key_name | regex_replace("[^A-Za-z0-9\_]", "_")
    parent_group: keys
    prefix: key
  - key: placement.region
    parent_group: regions
    prefix: ''
    separator: ''
  - key: security_groups | map(attribute="group_name") | map("regex_replace", "[^A-Za-z0-9\_]", "_") |
list
    parent_group: security_groups
    prefix: security_group
  - key: dict(tags.keys() | map("regex_replace", "[^A-Za-z0-9\_]", "_") | list | zip(tags.values()
      | map("regex_replace", "[^A-Za-z0-9\_]", "_") | list))
    parent_group: tags
    prefix: tag
  - key: tags.keys() | map("regex_replace", "[^A-Za-z0-9\_]", "_") | list
    parent_group: tags
    prefix: tag
  - key: vpc_id | regex_replace("[^A-Za-z0-9\_]", "_")
    parent_group: vpcs
    prefix: vpc_id
  - key: placement.availability_zone
    parent_group: '{{ placement.region }}'
    prefix: ''
```

```
    separator: ''
plugin: amazon.aws.aws_ec2
use_contrib_script_compatible_sanitization: true
```

8.2. GOOGLE COMPUTE ENGINE

```
auth_kind: serviceaccount
compose:
  ansible_ssh_host: networkInterfaces[0].accessConfigs[0].natIP |
default(networkInterfaces[0].networkIP)
  gce_description: description if description else None
  gce_id: id
  gce_image: image
  gce_machine_type: machineType
  gce_metadata: metadata.get("items", []) | items2dict(key_name="key", value_name="value")
  gce_name: name
  gce_network: networkInterfaces[0].network.name
  gce_private_ip: networkInterfaces[0].networkIP
  gce_public_ip: networkInterfaces[0].accessConfigs[0].natIP | default(None)
  gce_status: status
  gce_subnetwork: networkInterfaces[0].subnetwork.name
  gce_tags: tags.get("items", [])
  gce_zone: zone
hostnames:
- name
- public_ip
- private_ip
keyed_groups:
- key: gce_subnetwork
  prefix: network
- key: gce_private_ip
  prefix: ''
  separator: ''
- key: gce_public_ip
  prefix: ''
  separator: ''
- key: machineType
  prefix: ''
  separator: ''
- key: zone
  prefix: ''
  separator: ''
- key: gce_tags
  prefix: tag
- key: status | lower
  prefix: status
- key: image
  prefix: ''
  separator: ''
plugin: google.cloud.gcp_compute
retrieve_image_info: true
use_contrib_script_compatible_sanitization: true
```

8.3. MICROSOFT AZURE RESOURCE MANAGER

```
conditional_groups:
  azure: true
default_host_filters: []
fail_on_template_errors: false
hostvar_expressions:
  computer_name: name
  private_ip: private_ipv4_addresses[0] if private_ipv4_addresses else None
  provisioning_state: provisioning_state | title
  public_ip: public_ipv4_addresses[0] if public_ipv4_addresses else None
  public_ip_id: public_ip_id if public_ip_id is defined else None
  public_ip_name: public_ip_name if public_ip_name is defined else None
  tags: tags if tags else None
  type: resource_type
keyed_groups:
- key: location
  prefix: ''
  separator: ''
- key: tags.keys() | list if tags else []
  prefix: ''
  separator: ''
- key: security_group
  prefix: ''
  separator: ''
- key: resource_group
  prefix: ''
  separator: ''
- key: os_disk.operating_system_type
  prefix: ''
  separator: ''
- key: dict(tags.keys() | map("regex_replace", "^(.*)$", "\1_") | list | zip(tags.values() | list)) if tags else
[]
  prefix: ''
  separator: ''
plain_host_names: true
plugin: azure.azcollection.azure_rm
use_contrib_script_compatible_sanitization: true
```

8.4. VMWARE VCENTER

```
compose:
  ansible_host: guest.ipAddress
  ansible_ssh_host: guest.ipAddress
  ansible_uuid: 99999999 | random | to_uuid
  availablefield: availableField
  configissue: configIssue
  configstatus: configStatus
  customvalue: customValue
  effectiverole: effectiveRole
  guestheartbeatstatus: guestHeartbeatStatus
  layoutex: layoutEx
  overallstatus: overallStatus
  parentvapp: parentVApp
  recenttask: recentTask
  resourcepool: resourcePool
  rootsnapshot: rootSnapshot
```

```
  triggeredalarmstate: triggeredAlarmState
filters:
- runtime.powerState == "poweredOn"
keyed_groups:
- key: config.guestId
  prefix: ''
  separator: ''
- key: '"templates" if config.template else "guests"'
  prefix: ''
  separator: ''
plugin: community.vmware.vmware_vm_inventory
properties:
- availableField
- configIssue
- configStatus
- customValue
- datastore
- effectiveRole
- guestHeartbeatStatus
- layout
- layoutEx
- name
- network
- overallStatus
- parentVApp
- permission
- recentTask
- resourcePool
- rootSnapshot
- snapshot
- triggeredAlarmState
- value
- capability
- config
- guest
- runtime
- storage
- summary
strict: false
with_nested_properties: true
```

8.5. RED HAT SATELLITE 6

```
group_prefix: foreman_
keyed_groups:
- key: foreman['environment_name'] | lower | regex_replace(' ', '') | regex_replace('[^A-Za-z0-9_]', '_') |
regex_replace('none', '')
  prefix: foreman_environment_
  separator: ''
- key: foreman['location_name'] | lower | regex_replace(' ', '') | regex_replace('[^A-Za-z0-9_]', '_')
  prefix: foreman_location_
  separator: ''
- key: foreman['organization_name'] | lower | regex_replace(' ', '') | regex_replace('[^A-Za-z0-9_]', '_')
  prefix: foreman_organization_
  separator: ''
```

```
- key: foreman['content_facet_attributes']['lifecycle_environment_name'] | lower | regex_replace(' ', '') |
regex_replace('[^A-Za-z0-9_]', '_')
  prefix: foreman_lifecycle_environment_
  separator: ''
- key: foreman['content_facet_attributes']['content_view_name'] | lower | regex_replace(' ', '') |
regex_replace('[^A-Za-z0-9_]', '_')
  prefix: foreman_content_view_
  separator: ''
legacy_hostvars: true
plugin: theforeman.foreman.foreman
validate_certs: false
want_facts: true
want_hostcollections: false
want_params: true
```

8.6. OPENSTACK

```
expand_hostvars: true
fail_on_errors: true
inventory_hostname: uuid
plugin: openstack.cloud.openstack
```

8.7. RED HAT VIRTUALIZATION

```
compose:
  ansible_host: (devices.values() | list)[0][0] if devices else None
keyed_groups:
- key: cluster
  prefix: cluster
  separator: _
- key: status
  prefix: status
  separator: _
- key: tags
  prefix: tag
  separator: _
ovirt_hostname_preference:
- name
- fqdn
ovirt_insecure: false
plugin: ovirt.ovirt.ovirt
```

8.8. AUTOMATION CONTROLLER

```
include_metadata: true
inventory_id: <inventory_id or url_quoted_named_url>
plugin: awx.awx.tower
validate_certs: <true or false>
```

第9章 カスタム通知でサポートされている属性

このセクションでは、サポート対象のジョブ属性リストと、通知用のメッセージテキスト作成に適した構文について説明します。サポートされるジョブ属性は以下のとおりです。

- **allow_simultaneous**: (ブール値) 複数のジョブが、このジョブに関連付けられた JT から同時に実行できるかどうかを示す
- **controller_node**: (文字列) 分離された実行環境を管理したインスタンス
- **created**: (日時) ジョブ作成時のタイムスタンプ
- **custom_virtualenv**: (文字列) ジョブの実行に使用されるカスタムの仮想環境
- **description**: (文字列) ジョブの説明 (任意)
- **diff_mode**: (ブール値) 有効になっている場合、ホストのテンプレート化されたファイルに追加されるテキストの変更を標準出力に表示
- **elapsed**: (10 進数) ジョブ実行の経過時間 (秒単位)
- **execution_node**: (文字列) ジョブが実行するノード
- **failed:**: (ブール値) ジョブが失敗した場合は true
- **finished**: (日時) ジョブが実行を完了した日時
- **force_handlers**: (ブール値) ハンドラーが強制されている場合、ホストでタスクが失敗しても、通知されるとハンドラーが実行する (ホストに到達できない場合でも、状況によってはハンドラーの実行が回避されることに留意)
- **forks**: (整数) ジョブに要求されたフォークの数
- **id**: (整数) このジョブのデータベース ID
- **job_explanation**: (文字列) stdout の実行およびキャプチャーを実行できない場合のジョブの状態を示すための状態フィールド
- **job_slice_count**: (整数) スライスされたジョブの一部として実行された場合には、スライスの合計数 (1 の場合はスライスされたジョブの一部ではない)
- **job_slice_number**: (整数) スライスされたジョブの一部として実行された場合には、スライス処理が行われたインベントリーの ID (スライスされたジョブの一部でなければ属性は使用されない)
- **job_tags**: (文字列) 指定されたタグを持つタスクのみが実行される
- **job_type**: (選択肢) run、check、または scan
- **launch_type**: (選択肢) manual、relaunch、callback、scheduled、dependency、workflow、sync、または scm
- **limit**: (文字列) 指定された場合は、このホストのセットに制限された Playbook を実行
- **modified**: (日時) ジョブの最終更新時のタイムスタンプ
- **name**: (文字列) このジョブの名前

- **Playbook**: (文字列) 実行された Playbook
- **scm_revision**: (文字列) このジョブに使用するプロジェクトからの SCM リビジョン (存在する場合)
- **skip_tags**: (文字列) 指定した場合は、このタグセットをスキップして Playbook を実行
- **start_at_task**: (文字列) 指定した場合は、この名前に一致するタスクで Playbook の実行を開始
- **started**: (日時) ジョブが開始するためにキューに入れられた日時
- **status**: (選択肢) new、pending、waiting、running、successful、failed、error、canceled
- **timeout**: (整数) タスクが取り消されるまでの実行時間 (秒数)
- **type**: (選択肢) このジョブのデータタイプ
- **url**: (文字列) ジョブの URL
- **use_fact_cache**: (ブール値) ジョブに対して有効化されている場合、Tower は Ansible ファクトキャッシュプラグインとして機能。データベースに対する Playbook 実行の終了時にファクトを保持し、Ansible で使用できるようにキャッシュ
- **verbosity**: (選択肢) 0 - 5 (正常 - WinRM デバッグに対応)
- **host_status_counts**: (各ステータスに一意に割り当てられたホスト数)
  - **skipped**: (整数)
  - **ok**: (整数)
  - **failures**: (整数)
  - **failures**: (整数)
  - **dark**: (整数)
  - **processed**: (整数)
  - **rescued**: (整数)
  - **ignored** (整数)
  - **failed** (ブール値)
- **summary_fields:**
  - **inventory**
    - **id**: (整数) インベントリーのデータベース ID
    - **name**: (文字列) インベントリーの名前
    - **description**: (文字列) インベントリーの説明 (任意)
    - **has_active_failures**: (ブール値) (非推奨) このインベントリーのホストが失敗したかどうかを示すフラグ

- **total_hosts**: (非推奨) (整数) このインベントリー内のホストの合計数
- **hosts_with_active_failures**: (非推奨) (整数) このインベントリー内のアクティブなエラーのあるホストの数
- **total_groups**: (非推奨) (整数) このインベントリー内のグループの合計数
- **groups_with_active_failures**: (非推奨) (整数) このインベントリー内のアクティブなエラーのあるホストの数
- **has_inventory_sources**: (非推奨) (ブール値) このインベントリーに外部のインベントリーソースがあるかどうかを示すフラグ
- **total_inventory_sources**: インベントリー内で設定される外部インベントリーソースの合計数 (整数)
- **inventory_sources_with_failures**: エラーのあるこのインベントリー内の外部インベントリーソースの数 (整数)
- **organization_id**: このインベントリーが含まれる組織
- **kind**: (選択肢) (空の文字列) (ホストにインベントリーとの直接リンクがあることを示す) または smart

- **project**
  - **id**: (整数) プロジェクトのデータベース ID
  - **name**: (文字列) プロジェクトの名前
  - **description**: (文字列) プロジェクトの説明 (任意)
  - **status**: (選択肢) new、pending、waiting、running、successful、failed、error、canceled、never updated、ok、missing のいずれか
  - **scm_type**: (選択肢) (空の文字列)、git、hg、svn、insights のいずれか
- **job_template**
  - **id**: (整数) ジョブテンプレートのデータベース ID
  - **name**: (文字列) ジョブテンプレートの名前
  - **description**: (文字列) ジョブテンプレートの説明 (任意)
- **unified_job_template**
  - **id**: (整数) 統合ジョブテンプレートのデータベース ID
  - **name**: (文字列) 統合ジョブテンプレートの名前
  - **description**: (文字列) 統合ジョブテンプレートの説明 (任意)
  - **unified_job_type**: (選択肢) 統合ジョブタイプ (job、workflow_job、project_update など)
- **instance_group**

        - **id**: (整数) インスタンスグループのデータベース ID
        - **name**: (文字列) インスタンスグループの名前
    - **created_by**
        - **id**: (int) 操作を開始したユーザーのデータベース ID
        - **username**: (文字列) 操作を開始したユーザー名
        - **first_name** - (文字列) 名
        - **last_name** - (文字列) 姓
    - **labels**
        - **count**: (整数) ラベルの数
        - **results**: ラベルを表すディクショナリーのリスト ({"id": 5, "name": "database jobs"} など)

ジョブに関する情報は、グループ化された中括弧 {{}} を使用してカスタム通知メッセージで参照できます。特定のジョブ属性にはドット表記を使用してアクセスされます (例: {{ job.summary_fields.inventory.name }})。中括弧の前または前後に使用されている文字、または平文は、明確にするために追加できます。たとえば、ジョブ ID の「#」や記述子を示す一重引用符などです。カスタムメッセージには、メッセージ全体で多数の変数を含めることができます。

```
{{ job_friendly_name }} {{ job.id }} ran on {{ job.execution_node }} in {{ job.elapsed }} seconds.
```

ジョブ属性に加えてテンプレートに追加できる変数は、他にも複数あります。

approval_node_name: (文字列) 承認ノード名

approval_status: (選択肢) approved、denied、timed_out のいずれか

url: (文字列) 通知が送信されるジョブの URL (開始、成功、失敗、および承認の通知に適用)

workflow_url: (文字列) 関連する承認ノードへの URL。これにより、通知の受信者は関連するワークフロージョブページに移動し、何が起こっているかを確認できます (つまり、このノードは {{ workflow_url }}) で確認できます)。承認関連の通知の場合は、url と workflow_url の両方が同じです。

job_friendly_name: (文字列) ジョブの分かりやすい名前

job_metadata: (文字列) ジョブのメタデータを JSON 文字列として置き換えます。以下に例を示します。

```
{'url': 'https://towerhost/$/jobs/playbook/13',
 'traceback': '',
 'status': 'running',
 'started': '2019-08-07T21:46:38.362630+00:00',
 'project': 'Stub project',
 'playbook': 'ping.yml',
 'name': 'Stub Job Template',
 'limit': '',
 'inventory': 'Stub Inventory',
 'id': 42,
 'hosts': {},
```

```
'friendly_name': 'Job',
'finished': False,
'credential': 'Stub credential',
'created_by': 'admin'}
```

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**
registry_password	**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	**[automation controller]** グループと **[automation hub]** グループの両方に使用されます。 デフォルト = **registry.redhat.io**。
registry_username	**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。

A.2. ANSIBLE AUTOMATION HUB 変数

変数	説明
automationhub_admin_password	必須。
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_seach_base_dn**
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	**オプション** 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 = false** を設定すると、バックアップ/復元プロセスは **/var/lib/pulp** をバックアップまたは復元しません。 デフォルト = **true**
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**
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	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。**true** に設定すると、署名サービスが作成されます。 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。Automation Hub が **HTTP Strict Transport Security** (HSTS) 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。 デフォルト: **false**
automationhub_disable_https	**オプション** 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。 デフォルト: **false**
automationhub_importer_settings	**オプション** 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。

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
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。
automationhub_ldap_bind_password	**必須** **Automationhub_ldap_bind_dn** で使用するパスワード。
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 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 **Automationhub_ldap_group_type** を設定する必要があります。 デフォルト = **None**

変数	説明
automatiohub_ldap_group_search_filter	**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。 デフォルト = **(objectClass=Group)**
automationhub_ldap_group_search_scope	**オプション** デフォルト = **SUBTREE**
automationhub_ldap_group_type_class	**オプション** デフォルト =**django_auth_ldap.config:GroupOfNamesType**
automationhub_ldap_server_uri	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。
automationhub_ldap_user_search_base_dn	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。
automationhub_main_url	Single Sign-On を使用する場合は、クライアントが接続するメインの Automation Hub URL を指定します (例 **https://<hubaddress.example.com>**)。これにより、外部アドレスが **/etc/pulp/settings.py** に入力されます。 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。
automationhub_pg_password	Automation Hub PostgreSQL データベースのパスワード。 **Automationhub_pg_password** 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。

変数	説明
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**
automationhub_require_content_approval	**オプション** 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。 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。 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_ssl_cert	**オプション** **/path/to/automationhub.cert**: **web_server_ssl_cert** と同じですが、Automation Hub の UI と API 用。
automationhub_ssl_key	**オプション** **/path/to/automationhub.key** **web_server_ssl_key** と同じですが、Automation Hub UI および API 用です
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。 デフォルト: **false**

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。
pulp_db_fields_key	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、新しいキーが生成されます。

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
sso_automation_platform_login_theme	**オプション** 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	**オプション** 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**

変数	説明
sso_automation_platform_realm_displayname	**オプション** 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	**オプション** 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	**オプション** 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。
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 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。 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。 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。

変数	説明
sso_keystore_file_remote	**オプション** 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	**オプション** 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	**オプション** 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。
sso_ssl_validate_certs	**オプション** 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場合は **true** に設定します。 デフォルト = **true**

変数	説明
sso_use_https	**オプション** 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。 Single Sign On が https を使用する場合。 デフォルト = **true**

A.4. 自動化サービスのカタログ変数

変数	説明
automationcatalog_controller_password	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、**automation_controller_main_url** も定義する必要があります。
automationcatalog_controller_token	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。
automationcatalog_controller_username	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、**automation_controller_main_url** も定義する必要があります。
automationcatalog_controller_verify_ssl	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。 デフォルト = **true**
automationcatalog_disable_hsts	自動化サービスカタログの HSTS Web セキュリティーポリシーを有効または無効にするために使用されます。 デフォルト = `false
automationcatalog_disable_https	Services Catalog の HSTS Web セキュリティーポリシーを有効または無効にするために使用されます。 デフォルト: **false**
automationcatalog_enable_analytics_collection	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

変数	説明
automationcatalog_main_url	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。
automationcatalog_pg_database	自動化サービスカタログの postgres データベース URL。
automationcatalog_pg_host	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)。
automationcatalog_pg_password	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。 **Automationcatalog_pg_password** 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。
automationcatalog_pg_port	自動化サービスカタログに使用する PostgreSQL ポート。 デフォルト = 5432
automationcatalog_pg_username	自動化サービスカタログの postgres ID。
automationcatalog_ssl_cert	カスタム提供の SSL 証明書ファイルへのパス。**automationcatalog_ssl_key** 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。
automationcatalog_ssl_key	カスタム提供の SSL 証明書キーファイルへのパス。 **Automationcatalog_ssl_cert** 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。

A.5. オートメーションコントローラー変数

変数	説明
admin_password	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。

変数	説明
automationcontroller_main_url	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。 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。
automationcontroller_password	Automation Controller インスタンスのパスワード。
automationcontroller_username	Automation Controller インスタンスのユーザー名。
node_state	**オプション** 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**、または従来の分離されたノードを実行ノードに移行する **iso_migrate** です。 デフォルト = **active**。
node_type	**[automationcontroller]** グループ用。 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** 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。 **node_type=hybrid** には、すべてを実行する機能があります。 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_type 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**。

変数	説明
ピア	**オプション** ピアの関係はノード間の接続を定義します。 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。ピアリング を参照してください。 peers 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、**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	postgres 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。
pg_password	postgreSQL データベースのパスワードを設定します。 **pg_password** 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。 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	**preferred** または **verify-full** のいずれか。 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**。

変数	説明
pg_username	postgres データベースのユーザー名。 デフォルト = **awx**。
postgres_ssl_cert	postgres SSL 証明書の場所。 **/path/to/pgsql_ssl.cert**
`postgres_ssl_key	postgres ssl キーの場所。 **/path/to/pgsql_ssl.key**
postgres_use_cert	postgres ユーザー証明書の場所。 **/path/to/pgsql.crt**
postgres_use_key	postgres ユーザーキーの場所。 **/path/to/pgsql.key**
postgres_use_ssl	postgres が SSL を使用する場合。
receptor_listener_port	レセプター接続に使用するポート。 デフォルト = 27199.
web_server_ssl_cert	**オプション** **/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	**オプション** **/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.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 にあります。

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	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。
ansible_user	ホストに接続する際に使用するユーザー名。
ansible_password	ホストへの認証に使用するパスワード。 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。複数のキーを使用していて、SSH エージェントを使用したく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。
ansible_ssh_pass	

変数	説明
ansible_ssh_user	この変数はインストーラーで使用する SSH ユーザーを設定します。これは、デフォルトでは root になります。このユーザーは、パスワードなしで SSH ベースの認証を許可する必要があります。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。
ansible_ssh_executable	(バージョン 2.2 で追加されています。) 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 **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 **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 コントローラー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** を使用できない場合は、つまり、ターゲットマシンに **/bin/sh** 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。

次の変数は、ユーザーが直接設定することはできません。Ansible は常にそれらをオーバーライドして、内部状態を反映します。

変数	説明
ansible_check_mode	チェックモードかどうかを指定するブール値
ansible_dependent_role_names	他のプレイの依存関係として現在のプレイに現在インポートされているロールの名前
ansible_limit	Ansible の現在の実行に対して、CLI オプション **--limit** として指定する内容
ansible_loop	**loop_control.extended** を使用して有効にすると、拡張ループ情報を含むディクショナリーまたはマップ

変数	説明
ansible_loop_var	loop_control.loop_var に提供される値の名前。2.8 で追加
ansible_index_var	**loop_control.index_var** に提供される値の名前。2.9 で追加
ansible_parent_role_names	現在のロールが **include_role** または **import_role** **action** によって実行されている場合、この変数にはすべての親ロールのリストと最新のロールが含まれます。つまり、このロールをインクルードまたはインポートしたロールは、リストの最初の項目です。複数の包含が発生した場合、このリストの最初の項目は最後のロール (このロールを含めたロール) です。特定のロールがこのリストに複数存在することも可能です。 例: ロール A にロール B が含まれている場合、ロール B 内では、**ansible_parent_role_names** は ['A'] と等しくなります。ロール B にロール C が含まれる場合、リストは ['B', 'A'] になります。
ansible_parent_role_paths	現在のロールが **include_role** または **import_role** アクションによって実行されている場合、この変数にはすべての親ロールパスのリストが含まれ、最新のロール (つまり、このロールをインクルード/インポートしたロール) が最初の要素になります。リストで。このリストの項目の順序については、**ansible_parent_role_names** を参照してください。
ansible_play_batch	シリアル (**batch**) によって制限された、現在のプレイ実行中のアクティブなホストのリスト。障害が発生したホストや到達不能なホストは、**active** とは見なされません。
ansible_play_hosts	シリアルに限定されない、現在のプレイ実行中のホストのリスト。失敗したホストまたは到達不能なホストは、このリストから除外されます。
ansible_play_hosts_all	プレイが対象としたホストの一覧
ansible_play_role_names	現在のプレイに現在インポートされているロールの名前。このリストには、依存関係によって暗黙的に含まれるロール名は含まれていません。
ansible_play_name	現在実行中のプレイの名前。2.8 で追加されました。(プレイブックのファイル名ではなく、プレイの name 属性。)

変数	説明
ansible_search_path	アクションプラグインとルックアップの現在の検索パス、つまり、テンプレートを実行するときに相対パスを検索する場所: src=myfile
ansible_version	現在実行している Ansible のバージョンに関する情報を含むディクショナリーまたはマップ。**full**、**major**、**minor**、**revision**、**string** などのキーが含まれます。