Red Hat Ansible Automation Platform 2.0-ea

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

本ガイドでは、Red Hat Ansible Automation Platform の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を提供します。

Last Updated: 2021-07-21

Red Hat Ansible Automation Platform 2.0-ea Red Hat Ansible Automation Platform インストールガイド

本ガイドでは、Red Hat Ansible Automation Platform の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を提供します。

法律上の通知

概要

フィードバックの提供: 本書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、http://issues.redhat.com で案件を作成してください。Ansible Automation Platform プロジェクトを選択し、Documentation コンポーネントを使用します。

目次

序文

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。

本書では、Ansible Automation Platform のインストールにおけるインストール要件とプロセスについて説明します。本書の更新により、Ansible Automation Platform の最新リリースの情報が追加されました。

第1章 RED HAT ANSIBLE AUTOMATION PLATFORM インストールの計画

本セクションを使用すると、Red Hat Ansible Automation Platform インストールの計画に役立ちます。インストールの前に、セットアップインストーラー、システム要件、およびサポートされるインストールシナリオに関する情報を確認してください。

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

Red Hat Ansible Automation Platform インストールを計画する際に、この情報を使用します。インストール済みの各 Ansible Automation Platform コンポーネント: 自動化ハブおよび自動化コントローラーには、以下の要件があります。

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 Red Hat Ansible Automation Platform のシステム要件

	必須	備考
サブスクリプション	有効な Red Hat Ansible Automation Platform	
OS	Red Hat Enterprise Linux 8.2 以降 64 ビット (x86)	
Ansible	バージョン 2.9 が必要です。	
RAM	最小 4 GB	• 4 GB のメモリー (Vagrant 試用版のインストールには推奨最小要件) • 4 GB メモリー (外部のスタンドアロン PostgreSQL データベースには最小要件) • 設定のフォークに基づく容量については、追加のリソース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、追加のリソースを参照してください。

	必須	備考
ディスク: サービスノード	40 GB の専用ハードディスク領域	• **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 750 となるようにする必要があります。 • 自動化コントローラーストレージの自動化の詳細は、以下の注記を参照してください。
ディスク: データベースノード	20 GB の専用ハードディスク領域	• 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 (1000 以上) 必要があります。
Browser	Mozilla Firefox または Google Chrome の現行のサポートバージョン	
Database	PostgreSQL バージョン 12	

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、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 m4.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 つのフォークごとに 1 GB のメモリーと、自動化コントローラー用に 2 GB のメモリーを予約すことを推奨しています。詳細は、:ref:`容量アルゴリズム <userguide:ug_job_concurrency>` を参照してください。**forks** が 400 に設定されている場合、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 の主な変更

Red Hat Ansible Automation Platform は PostgreSQL 12 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。

- PostgreSQL 12 ではユーザーのパスワードをより安全に保存できるため、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。インストーラーのインベントリーファイルでパスワードを指定した場合 (**pg_password**) に、このパスワードは、インストールプロセスの一部として PostgreSQL により SCRAM-SHA-256 にハッシュ化されます。**pg_password** で特殊文字を使用すると設定が失敗する場合があるため、使用 **しない** でください。

- 自動化コントローラーおよび自動化ハブは、3.8 で PostgreSQL の Software Collections バージョンを使用しているため、データベースにアクセスするために **rh-postgresql10** scl を有効にする必要があります。管理者は **awx-manage dbshell** コマンドを使用して、PostgreSQL SCL を自動的に有効にできます。

- 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必要がある場合は、**awx-manage check_db** コマンドで実行できます。

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。

Ansible のソフトウェア要件

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。

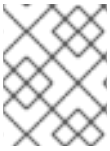
新規インストールの実行時に、コントローラーの自動化は Ansible 2.9 の最新のリリースパッケージをインストールします。

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。

1.2. 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 環境がインターネットに接続している場合は、Ansible Automation Platform 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、および依存関係が取得されます。

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
```

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たくない場合は、Red Hat Ansible Automation Platform 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
```

1.3. 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、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、自動化ハブリポジトリーが有効になります。

有効なサブスクリプションは、自動化ハブにのみ割り当てる必要があります。**[automationhub]** グループが空白の場合でも、他のノードに有効なサブスクリプションを割り当てる必要はありません。これは、**repos_el** ロールレベルで行われ、このロールは **[automationcontroller] および '[automationhub]** ホストの両方で実行します。

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
```

コマンドは以下を返します。

```
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: ``````````
  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>
```

全ノードには、Red Hat Ansible Automation Platform が割り当てられ、自動化ハブリポジトリーを検索します。

検証

1. 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
```

1.4. 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。

1.4.1. 同じ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。

1.4.2. 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。

1.4.3. 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自動化ハブ

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、自動化ハブ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ハブを設定します。

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自動化ハブのインストール」を参照してください。

1.4.4. 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ハブ

このシナリオでは、1台のマシンに自動化ハブ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自動化ハブをインストール」を参照してください。

1.4.5. 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への、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自動化ハブ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。

1.4.6. 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自動化ハブをインストールし、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。

1.4.7. 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自動化ハブ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 12** 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 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。

2.1. 同じノードへ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

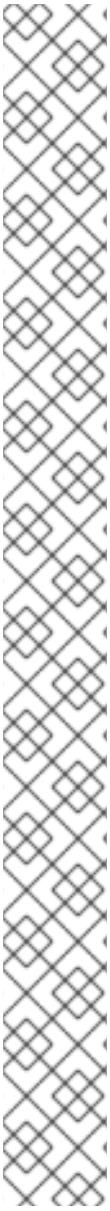
これらの手順に従って、同じノード上にデータベースを使用して自動化コントローラーのスタンドアロンインスタンス、またはインストーラー以外が管理するデータベースをインストールできます。このシナリオでは、1 台のマシンに Web フロントエンド、REST API バックエンド、データベースを含む自動化コントローラーのインストールが含まれます。PostgreSQL をインストールし、そのデータベースとして使用するように自動化コントローラーを設定します。これは、標準の自動化コントローラーのインストールシナリオとみなされます。

2.1.1. 前提条件

- プラットフォームインストーラーを選択し、取得している。
- 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。

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。



注記

- **外部データベース** の使用: インベントリーファイルのデータベースセクションが正しく設定されていることを確認します。
- **[automationhub]** グループに自動化ハブの情報を追加
- 自動化ハブと自動化コントローラーは、同じノードにインストールできません。
- 自動化コントローラーは、使用するデータベースのレプリケーションやフェイルオーバーを設定しません。自動化コントローラーは、所有するレプリケーションと連携する必要があります。
- パフォーマンス上の理由から、データベースサーバーは、自動化コントローラーサーバーと同じネットワーク上または同じデータセンターに置く必要があります。
- **既存のクラスターのアップグレード** の場合: クラスターのアップグレード時に、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ためにクラスターを再設定しないといけない必要があります。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省略すると、クラスターからインスタンスを削除するだけでは不十分です。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他に、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。それ以外の場合には、省略されたインスタンスまたはインスタンスグループはクラスターと通信し続けます。これにより、アップグレード中に Tower サービスに関する問題が生じる可能性があります。
- **クラスター化したインストールの場合**: クラスター設定を作成する場合は、**localhost** を全インスタンスのホスト名または IP アドレスに置き換える必要があります。すべてのノードまたはインスタンスには、このホスト名や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** を使用できません。また、すべてのノードがホスト名と同じ形式を使用する必要があります。

重要

- リモートマシンへの root アクセスが必要です。Ansible では、これはさまざまな方法で実行できます。
- **ansible_user=root ansible_ssh_pass="your_password_here"** インベントリーのホスト変数またはグループ変数
- **ansible_user=root ansible_ssh_private_key_file="path_to_your_keyfile.pem"** インベントリーのホスト変数またはグループ変数
- **ANSIBLE_BECOME_METHOD='sudo' ANSIBLE_BECOME=True ./setup.sh**

become プラグインの詳細は、「特権昇格について」を参照してください。

手順

1. インストーラーに移動します。
   a. [bundled installer]
      ■

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

b. [online installer]

```
$ 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. 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。

```
[automationcontroller]
127.0.0.1 ansible_connection=local

[database]

[all:vars]
admin_password='password'

pg_host=''
pg_port=''

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

2.1.4. フラグおよび追加変数

自動化コントローラーのインストール時に、フラグと追加変数を渡すことができます。

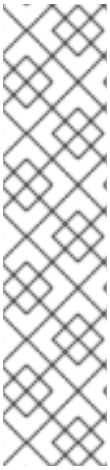
:

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う
-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。例: **./setup.sh -i my_awesome_inventory.yml -e matburt_is_country_gold=True — -K**.



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 の無効化	**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時のみ変更してください)	**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時のみ変更してください)	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方法:

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

- nginx で処理する https を無効化する方法:

```
./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方法:

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

*

```
To override an inventory file used by passing it as an argument to the setup script:
```

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
```

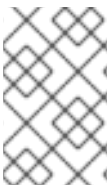
インストールが開始されます。

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。

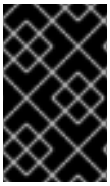
手順

1. **inventory** ファイルの自動化コントローラーノードに指定した 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。

これで自動化コントローラーが初期設定可能になりました。

2.1.6.1. 次のステップ

- 顧客スクリプトや管理ジョブなどの管理については、『Automation Controller 管理ガイド』を参照してください。
- 自動化コントローラーのクイック設定については、『Automation Controller クイック設定ガイド』を参照してください。

- 『自動化コントローラーユーザーガイド』で自動化コントローラー機能を確認してください。

2.1.7. 次のステップ

- プロキシーサーバーを使用して自動化コントローラーを設定する場合は、「**Red Hat Ansible Automation Platform のプロキシー設定**」を参照してください。
- Red Hat と共有する情報を制御する方法は、「**自動化コントローラーからのユーザビリティー、アナリティクスおよびデータ収集の管理**」を参照してください。
- 自動化コントローラーの使用方法の詳細は、『Ansible Tower クイック設定ガイド』を参照してください。

2.2. 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。

2.2.1. 前提条件

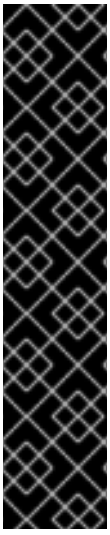
- プラットフォームインストーラーを選択し、取得している。
- ベースシステムの要件を満たすマシンにインストールしている。

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。

注記

- **外部データベース** の使用: インベントリーファイルのデータベースセクションが正しく設定されていることを確認します。
- **[automationhub]** グループに自動化ハブの情報を追加
- 自動化ハブと自動化コントローラーは、同じノードにインストールできません。
- 自動化コントローラーは、使用するデータベースのレプリケーションやフェイルオーバーを設定しません。自動化コントローラーは、所有するレプリケーションと連携する必要があります。
- パフォーマンス上の理由から、データベースサーバーは、自動化コントローラーサーバーと同じネットワーク上または同じデータセンターに置く必要があります。
- **既存のクラスターのアップグレード** の場合: クラスターのアップグレード時に、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ためにクラスターを再設定しないといけない必要があります。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省略すると、クラスターからインスタンスを削除するだけでは不十分です。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他に、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。それ以外の場合には、省略されたインスタンスまたはインスタンスグループはクラスターと通信し続けます。これにより、アップグレード中に Tower サービスに関する問題が生じる可能性があります。
- **クラスター化したインストールの場合**: クラスター設定を作成する場合は、**localhost** を全インスタンスのホスト名または IP アドレスに置き換える必要があります。すべてのノードまたはインスタンスには、このホスト名や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** を使用できません。また、すべてのノードがホスト名と同じ形式を使用する必要があります。



重要

- リモートマシンへの root アクセスが必要です。Ansible では、これはさまざまな方法で実行できます。
- **ansible_user=root ansible_ssh_pass="your_password_here"** インベントリーのホスト変数またはグループ変数
- **ansible_user=root ansible_ssh_private_key_file="path_to_your_keyfile.pem"** インベントリーのホスト変数またはグループ変数
- **ANSIBLE_BECOME_METHOD='sudo' ANSIBLE_BECOME=True ./setup.sh**

become プラグインの詳細は、「特権昇格について」を参照してください。

手順

1. インストーラーに移動します。
   a. [bundled installer]
      ■

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

b. [online installer]

```
$ 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。

```
[automationcontroller]
127.0.0.1 ansible_connection=local


[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'
```

2.2.4. フラグおよび追加変数

自動化コントローラーのインストール時に、フラグと追加変数を渡すことができます。

:

表2.3 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う
-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。例: **./setup.sh -i my_awesome_inventory.yml -e matburt_is_country_gold=True — -K**.

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2.4 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 の無効化	**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時のみ変更してください)	**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時のみ変更してください)	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方法:

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

- nginx で処理する https を無効化する方法:

```
./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方法:

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

*

```
To override an inventory file used by passing it as an argument to the setup script:
```

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されます。

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** ファイルの自動化コントローラーノードに指定した 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。

これで自動化コントローラーが初期設定可能になりました。

2.2.6.1. 次のステップ

- 顧客スクリプトや管理ジョブなどの管理については、『Automation Controller 管理ガイド』を参照してください。
- 自動化コントローラーのクイック設定については、『Automation Controller クイック設定ガイド』を参照してください。

- 『自動化コントローラーユーザーガイド』で自動化コントローラー機能を確認してください。

2.2.7. 次のステップ

- プロキシーサーバーを使用して自動化コントローラーを設定する場合は、「**Red Hat Ansible Automation Platform のプロキシー設定**」を参照してください。
- Red Hat と共有する情報を制御する方法は、「**自動化コントローラーからのユーザビリティー、アナリティクスおよびデータ収集の管理**」を参照してください。
- 自動化コントローラーの使用方法の詳細は、『Ansible Tower クイック設定ガイド』を参照してください。

2.3. 同じノードへのデータベースを使用した自動化ハブのインストール

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自動化ハブ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。

2.3.1. 前提条件

- プラットフォームインストーラーを選択し、取得している。
- 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。

2.3.2. Red Hat Ansible Automation Platform インストール設定

自動化ハブのインストール時に、以下の設定を使用できます。

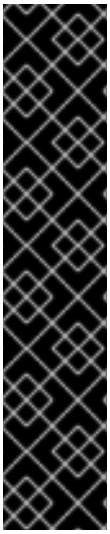
- **automationhub_importer_settings**: **galaxy-importer** に渡す設定および構成のディクショナリー。**/etc/galaxy-importer/galaxy-importer.cfg** で終了します。
- **automationhub_require_content_approval**: コレクションが利用可能になる前に、自動化ハブが承認メカニズムを適用するかどうか
- **automationhub_disable_https**: TLS を有効にして自動化ハブをデプロイするかどうか
- **automationhub_disable_hsts**: 自動化ハブが HTTP Strict Transport Security (HSTS) の Web セキュリティーポリシーメカニズムを有効にしてデプロイする必要があるかどうか。
- **automationhub_ssl_validate_certs**: デフォルトでは、プラットフォームは自己署名証明書を使用してデプロイするため、自動化ハブが自らを要求する際に証明書を検証するかどうか (デフォルト = False)
- **automationhub_ssl_cert**: **web_server_ssl_cert** と同じですが、自動化ハブの UI と API
- **automationhub_ssl_key**: 自動化ハブの UI と API 用の **web_server_ssl_key** と同じです。
- **automationhub_backup_collections**: オートメーションハブは、**/var/lib/pulp** のアーティファクトを提供します。デフォルトでは、これは **true** に設定され、自動化コントローラー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。パーティション (LVM、NFS、CephFS など) がそこにマウントされている場合、企業組織は常にバックアップされるようになります。この場合は、**automationhub_backup_collections = false** を設定すると、バックアップまたは復元プロセスでは **/var/lib/pulp** をバックアップまたは復元する必要はありません。

2.3.3. 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。

注記

- **外部データベース** の使用: インベントリーファイルのデータベースセクションが正しく設定されていることを確認します。
- **[automationhub]** グループに自動化ハブの情報を追加
- 自動化ハブと自動化コントローラーは、同じノードにインストールできません。
- 自動化コントローラーは、使用するデータベースのレプリケーションやフェイルオーバーを設定しません。自動化コントローラーは、所有するレプリケーションと連携する必要があります。
- パフォーマンス上の理由から、データベースサーバーは、自動化コントローラーサーバーと同じネットワーク上または同じデータセンターに置く必要があります。
- **既存のクラスターのアップグレード** の場合: クラスターのアップグレード時に、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ためにクラスターを再設定しないといけない必要があります。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省略すると、クラスターからインスタンスを削除するだけでは不十分です。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他に、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。それ以外の場合には、省略されたインスタンスまたはインスタンスグループはクラスターと通信し続けます。これにより、アップグレード中に Tower サービスに関する問題が生じる可能性があります。
- **クラスター化したインストールの場合**: クラスター設定を作成する場合は、**localhost** を全インスタンスのホスト名または IP アドレスに置き換える必要があります。すべてのノードまたはインスタンスには、このホスト名や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** を使用できません。また、すべてのノードがホスト名と同じ形式を使用する必要があります。



重要

- リモートマシンへの root アクセスが必要です。Ansible では、これはさまざまな方法で実行できます。
- **ansible_user=root ansible_ssh_pass="your_password_here"** インベントリーのホスト変数またはグループ変数
- **ansible_user=root ansible_ssh_private_key_file="path_to_your_keyfile.pem"** インベントリーのホスト変数またはグループ変数
- **ANSIBLE_BECOME_METHOD='sudo' ANSIBLE_BECOME=True ./setup.sh**

become プラグインの詳細は、「特権昇格について」を参照してください。

手順

1. インストーラーに移動します。

   a. [bundled installer]

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

   b. [online installer]

   ```
   $ 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。

2.3.4. スタンドアロン自動化ハブのインベントリーファイルの例

以下の例では、自動化ハブ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自動化ハブ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。

```
[automationcontroller]


[automationhub]
127.0.0.1 ansible_connection=local

[all:vars]
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
```

2.3.5. フラグおよび追加変数

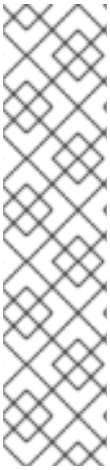
自動化コントローラーのインストール時に、フラグと追加変数を渡すことができます。

:

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う
-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。例: **./setup.sh -i my_awesome_inventory.yml -e matburt_is_country_gold=True — -K**.



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 の無効化	**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時のみ変更してください)	**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時のみ変更してください)	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方法:

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

- nginx で処理する https を無効化する方法:

```
./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方法:

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

*

```
To override an inventory file used by passing it as an argument to the setup script:
```

2.3.6. 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されます。

2.3.7. 自動化ハブ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。

これで、自動化ハブの初期設定の準備が整いました。

2.3.8. 次のステップ

- 自動化ハブにユーザーアクセスを設定するには、「プライベート自動化ハブでのユーザーアクセスの管理」を参照してください。
- 自動化ハブにコンテンツを追加する方法については、「自動化ハブにおける Red Hat Certified and Ansible Galaxy コレクションの管理」を参照してください。
- 自動化ハブで内部的に開発されたコレクションを公開する方法の詳細は、「{ハブ名}で独自のコンテンツコレクションを公開」を参照してください。

2.4. 外部データベースを使用した自動化ハブのインストール

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、自動化ハブ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、{HubNam} 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。

2.4.1. 前提条件

- プラットフォームインストーラーを選択し、取得している。
- 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。

2.4.2. Red Hat Ansible Automation Platform インストール設定

自動化ハブのインストール時に、以下の設定を使用できます。

- **automationhub_importer_settings**: **galaxy-importer** に渡す設定および構成のディクショナリー。**/etc/galaxy-importer/galaxy-importer.cfg** で終了します。
- **automationhub_require_content_approval**: コレクションが利用可能になる前に、自動化ハブが承認メカニズムを適用するかどうか
- **automationhub_disable_https**: TLS を有効にして自動化ハブをデプロイするかどうか

- **automationhub_disable_hsts**: 自動化ハブが HTTP Strict Transport Security (HSTS) の Web セキュリティーポリシーメカニズムを有効にしてデプロイする必要があるかどうか。

- **automationhub_ssl_validate_certs**: デフォルトでは、プラットフォームは自己署名証明書を使用してデプロイするため、自動化ハブが自らを要求する際に証明書を検証するかどうか (デフォルト = False)

- **automationhub_ssl_cert**: **web_server_ssl_cert** と同じですが、自動化ハブの UI と API

- **automationhub_ssl_key**: 自動化ハブの UI と API 用の **web_server_ssl_key** と同じです。

- **automationhub_backup_collections**: オートメーションハブは、**/var/lib/pulp** のアーティファクトを提供します。デフォルトでは、これは **true** に設定され、自動化コントローラー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。パーティション (LVM、NFS、CephFS など) がそこにマウントされている場合、企業組織は常にバックアップされるようになります。この場合は、**automationhub_backup_collections = false** を設定すると、バックアップまたは復元プロセスでは **/var/lib/pulp** をバックアップまたは復元する必要はありません。

2.4.3. 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。

注記

- **外部データベース** の使用: インベントリーファイルのデータベースセクションが正しく設定されていることを確認します。
- **[automationhub]** グループに自動化ハブの情報を追加
- 自動化ハブと自動化コントローラーは、同じノードにインストールできません。
- 自動化コントローラーは、使用するデータベースのレプリケーションやフェイルオーバーを設定しません。自動化コントローラーは、所有するレプリケーションと連携する必要があります。
- パフォーマンス上の理由から、データベースサーバーは、自動化コントローラーサーバーと同じネットワーク上または同じデータセンターに置く必要があります。
- **既存のクラスターのアップグレード** の場合: クラスターのアップグレード時に、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ためにクラスターを再設定しないといけない必要があります。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省略すると、クラスターからインスタンスを削除するだけでは不十分です。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他に、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。それ以外の場合には、省略されたインスタンスまたはインスタンスグループはクラスターと通信し続けます。これにより、アップグレード中に Tower サービスに関する問題が生じる可能性があります。
- **クラスター化したインストールの場合**: クラスター設定を作成する場合は、**localhost** を全インスタンスのホスト名または IP アドレスに置き換える必要があります。すべてのノードまたはインスタンスには、このホスト名や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** を使用できません。また、すべてのノードがホスト名と同じ形式を使用する必要があります。

重要

- リモートマシンへの root アクセスが必要です。Ansible では、これはさまざまな方法で実行できます。
- **ansible_user=root ansible_ssh_pass="your_password_here"** インベントリーのホスト変数またはグループ変数
- **ansible_user=root ansible_ssh_private_key_file="path_to_your_keyfile.pem"** インベントリーのホスト変数またはグループ変数
- **ANSIBLE_BECOME_METHOD='sudo' ANSIBLE_BECOME=True ./setup.sh**

become プラグインの詳細は、「特権昇格について」を参照してください。

手順

1. インストーラーに移動します。
   a. [bundled installer]
      ■

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

b. [online installer]

```
$ 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。

2.4.4. スタンドアロン自動化ハブのインベントリーファイルの例

以下の例では、自動化ハブ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自動化ハブ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。

```
[automationcontroller]


[automationhub]
127.0.0.1 ansible_connection=local

[database]
host2

[all:vars]
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
```

2.4.5. フラグおよび追加変数

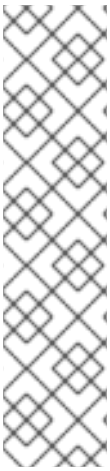
自動化コントローラーのインストール時に、フラグと追加変数を渡すことができます。

:

表2.7 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う
-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。例: **./setup.sh -i my_awesome_inventory.yml -e matburt_is_country_gold=True — -K**.



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2.8 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 の無効化	**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時のみ変更してください)	**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時のみ変更してください)	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方法:

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

- nginx で処理する https を無効化する方法:

```
./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方法:

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

*

```
To override an inventory file used by passing it as an argument to the setup script:
```

2.4.6. 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
```

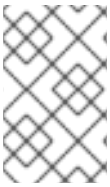
インストールが開始されます。

2.4.7. 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** ファイルの自動化コントローラーノードに指定した 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。

これで自動化コントローラーが初期設定可能になりました。

2.4.7.1. 次のステップ

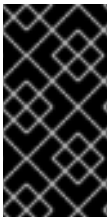
- 顧客スクリプトや管理ジョブなどの管理については、『Automation Controller 管理ガイド』を参照してください。
- 自動化コントローラーのクイック設定については、『Automation Controller クイック設定ガイド』を参照してください。
- 『自動化コントローラーユーザーガイド』で自動化コントローラー機能を確認してください。

2.4.8. 次のステップ

- 自動化ハブにユーザーアクセスを設定するには、「プライベート自動化ハブでのユーザーアクセスの管理」を参照してください。
- 自動化ハブにコンテンツを追加する方法については、「自動化ハブにおける Red Hat Certified and Ansible Galaxy コレクションの管理」を参照してください。
- 自動化ハブで内部的に開発されたコレクションを公開する方法の詳細は、「{ハブ名}で独自のコンテンツコレクションを公開」を参照してください。

第3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と自動化ハブをデプロイします。



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。自動化ハブ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、複数の自動化ハブをデプロイできます。

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。

3.1. 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に、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自動化ハブの両方) をインストールできます。

3.1.1. 前提条件

- プラットフォームインストーラーを選択し、取得している。
- 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。

3.1.2. Red Hat Ansible Automation Platform インストール設定

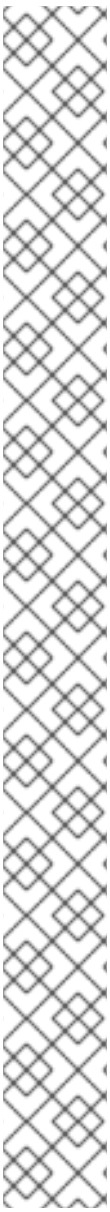
自動化ハブのインストール時に、以下の設定を使用できます。

- **automationhub_importer_settings**: **galaxy-importer** に渡す設定および構成のディクショナリー。**/etc/galaxy-importer/galaxy-importer.cfg** で終了します。
- **automationhub_require_content_approval**: コレクションが利用可能になる前に、自動化ハブが承認メカニズムを適用するかどうか
- **automationhub_disable_https**: TLS を有効にして自動化ハブをデプロイするかどうか
- **automationhub_disable_hsts**: 自動化ハブが HTTP Strict Transport Security (HSTS) の Web セキュリティーポリシーメカニズムを有効にしてデプロイする必要があるかどうか。
- **automationhub_ssl_validate_certs**: デフォルトでは、プラットフォームは自己署名証明書を使用してデプロイするため、自動化ハブが自らを要求する際に証明書を検証するかどうか (デフォルト = False)
- **automationhub_ssl_cert**: **web_server_ssl_cert** と同じですが、自動化ハブの UI と API
- **automationhub_ssl_key**: 自動化ハブの UI と API 用の **web_server_ssl_key** と同じです。
- **automationhub_backup_collections**: オートメーションハブは、**/var/lib/pulp** のアーティファ

クトを提供します。デフォルトでは、これは **true** に設定され、自動化コントローラー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。パーティション (LVM、NFS、CephFS など) がそこにマウントされている場合、企業組織は常にバックアップされるようになります。この場合は、**automationhub_backup_collections = false** を設定すると、バックアップまたは復元プロセスでは **/var/lib/pulp** をバックアップまたは復元する必要はありません。

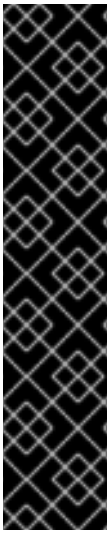
3.1.3. Red Hat Ansible Automation Platform インストーラーのインベントリーファイルの編集

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。



注記

- **外部データベース** の使用: インベントリーファイルのデータベースセクションが正しく設定されていることを確認します。
- **[automationhub]** グループに自動化ハブの情報を追加
- 自動化ハブと自動化コントローラーは、同じノードにインストールできません。
- 自動化コントローラーは、使用するデータベースのレプリケーションやフェイルオーバーを設定しません。自動化コントローラーは、所有するレプリケーションと連携する必要があります。
- パフォーマンス上の理由から、データベースサーバーは、自動化コントローラーサーバーと同じネットワーク上または同じデータセンターに置く必要があります。
- **既存のクラスターのアップグレード** の場合: クラスターのアップグレード時に、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ためにクラスターを再設定しないといけない必要があります。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省略すると、クラスターからインスタンスを削除するだけでは不十分です。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他に、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。それ以外の場合には、省略されたインスタンスまたはインスタンスグループはクラスターと通信し続けます。これにより、アップグレード中に Tower サービスに関する問題が生じる可能性があります。
- **クラスター化したインストールの場合**: クラスター設定を作成する場合は、**localhost** を全インスタンスのホスト名または IP アドレスに置き換える必要があります。すべてのノードまたはインスタンスには、このホスト名や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** を使用できません。また、すべてのノードがホスト名と同じ形式を使用する必要があります。



重要

- リモートマシンへの root アクセスが必要です。Ansible では、これはさまざまな方法で実行できます。
- **ansible_user=root ansible_ssh_pass="your_password_here"** インベントリーのホスト変数またはグループ変数
- **ansible_user=root ansible_ssh_private_key_file="path_to_your_keyfile.pem"** インベントリーのホスト変数またはグループ変数
- **ANSIBLE_BECOME_METHOD='sudo' ANSIBLE_BECOME=True ./setup.sh**

become プラグインの詳細は、「特権昇格について」を参照してください。

手順

1. インストーラーに移動します。
   a. [bundled installer]

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

   b. [online installer]

```
$ 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。

3.1.4. 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、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自動化ハブ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自動化ハブ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。

```
[automationcontroller]
tower.acme.org

[automationhub]
automationhub.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

# 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
# Isolated Tower nodes automatically generate an RSA key for authentication;
# To disable this behavior, set this value to false
# isolated_key_generation=true
# SSL-related variables
# If set, this will install a custom CA certificate to the system trust store.
# custom_ca_cert=/path/to/ca.crt
# Certificate and key to install in nginx for the web UI and API
# web_server_ssl_cert=/path/to/tower.cert
# web_server_ssl_key=/path/to/tower.key
# Certificate and key to install in Automation Hub node
# automationhub_ssl_cert=/path/to/automationhub.cert
# automationhub_ssl_key=/path/to/automationhub.key
# Server-side SSL settings for PostgreSQL (when we are installing it).
# postgres_use_ssl=False
# postgres_ssl_cert=/path/to/pgsql.crt
# postgres_ssl_key=/path/to/pgsql.key
```

3.1.5. フラグおよび追加変数

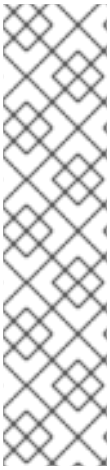
自動化コントローラーのインストール時に、フラグと追加変数を渡すことができます。

:

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。
-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う
-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。例: **./setup.sh -i my_awesome_inventory.yml -e matburt_is_country_gold=True — -K**.



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**

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 の無効化	**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時のみ変更してください)	**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時のみ変更してください)	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方法:

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

- nginx で処理する https を無効化する方法:

```
./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方法:

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

*

```
To override an inventory file used by passing it as an argument to the setup script:
```

3.1.6. 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
```

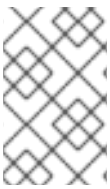
インストールが開始されます。

3.1.7. 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。

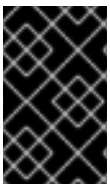
手順

1. **inventory** ファイルの自動化コントローラーノードに指定した 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。

これで自動化コントローラーが初期設定可能になりました。

3.1.7.1. 次のステップ

- 顧客スクリプトや管理ジョブなどの管理については、『Automation Controller 管理ガイド』を参照してください。
- 自動化コントローラーのクイック設定については、『Automation Controller クイック設定ガイド』を参照してください。

- 『自動化コントローラーユーザーガイド』で自動化コントローラー機能を確認してください。

3.1.8. 自動化ハブ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。

これで、自動化ハブの初期設定の準備が整いました。

3.1.9. 次のステップ

- プロキシーサーバーを使用して自動化コントローラーを設定する場合は、「**Red Hat Ansible Automation Platform のプロキシー設定**」を参照してください。
- Red Hat と共有する情報を制御する方法は、「**自動化コントローラーからのユーザビリティーアナリティクスおよびデータ収集の管理**」を参照してください。
- 自動化コントローラーの使用方法の詳細は、『Ansible Tower クイック設定ガイド』を参照してください。
- 自動化のハブ設定に関する完全なドキュメントは、Ansible Automation Platform の製品ドキュメントを参照してください。

3.2. 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自動化ハブの両方) をインストールできます。

3.2.1. 前提条件

- プラットフォームインストーラーを選択し、取得している。
- 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。

3.2.2. Red Hat Ansible Automation Platform インストール設定

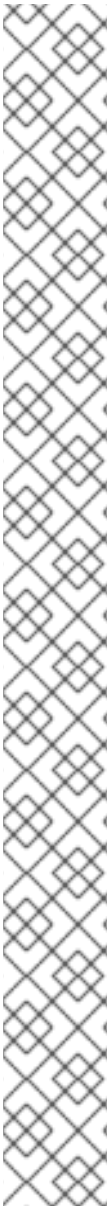
自動化ハブのインストール時に、以下の設定を使用できます。

- **automationhub_importer_settings**: **galaxy-importer** に渡す設定および構成のディクショナリー。**/etc/galaxy-importer/galaxy-importer.cfg** で終了します。

- **automationhub_require_content_approval**: コレクションが利用可能になる前に、自動化ハブが承認メカニズムを適用するかどうか
- **automationhub_disable_https**: TLS を有効にして自動化ハブをデプロイするかどうか
- **automationhub_disable_hsts**: 自動化ハブが HTTP Strict Transport Security (HSTS) の Web セキュリティーポリシーメカニズムを有効にしてデプロイする必要があるかどうか。
- **automationhub_ssl_validate_certs**: デフォルトでは、プラットフォームは自己署名証明書を使用してデプロイするため、自動化ハブが自らを要求する際に証明書を検証するかどうか (デフォルト = False)
- **automationhub_ssl_cert**: **web_server_ssl_cert** と同じですが、自動化ハブの UI と API
- **automationhub_ssl_key**: 自動化ハブの UI と API 用の **web_server_ssl_key** と同じです。
- **automationhub_backup_collections**: オートメーションハブは、**/var/lib/pulp** のアーティファクトを提供します。デフォルトでは、これは **true** に設定され、自動化コントローラー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。パーティション (LVM、NFS、CephFS など) がそこにマウントされている場合、企業組織は常にバックアップされるようになります。この場合は、**automationhub_backup_collections = false** を設定すると、バックアップまたは復元プロセスでは **/var/lib/pulp** をバックアップまたは復元する必要はありません。

3.2.3. 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。



注記

- **外部データベース** の使用: インベントリーファイルのデータベースセクションが正しく設定されていることを確認します。
- **[automationhub]** グループに自動化ハブの情報を追加
- 自動化ハブと自動化コントローラーは、同じノードにインストールできません。
- 自動化コントローラーは、使用するデータベースのレプリケーションやフェイルオーバーを設定しません。自動化コントローラーは、所有するレプリケーションと連携する必要があります。
- パフォーマンス上の理由から、データベースサーバーは、自動化コントローラーサーバーと同じネットワーク上または同じデータセンターに置く必要があります。
- **既存のクラスターのアップグレード** の場合: クラスターのアップグレード時に、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ためにクラスターを再設定しないといけない必要があります。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省略すると、クラスターからインスタンスを削除するだけでは不十分です。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他に、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。それ以外の場合には、省略されたインスタンスまたはインスタンスグループはクラスターと通信し続けます。これにより、アップグレード中に Tower サービスに関する問題が生じる可能性があります。
- **クラスター化したインストールの場合**: クラスター設定を作成する場合は、**localhost** を全インスタンスのホスト名または IP アドレスに置き換える必要があります。すべてのノードまたはインスタンスには、このホスト名や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** を使用できません。また、すべてのノードがホスト名と同じ形式を使用する必要があります。

重要

- リモートマシンへの root アクセスが必要です。Ansible では、これはさまざまな方法で実行できます。
- **ansible_user=root ansible_ssh_pass="your_password_here"** インベントリーのホスト変数またはグループ変数
- **ansible_user=root ansible_ssh_private_key_file="path_to_your_keyfile.pem"** インベントリーのホスト変数またはグループ変数
- **ANSIBLE_BECOME_METHOD='sudo' ANSIBLE_BECOME=True ./setup.sh**

become プラグインの詳細は、「特権昇格について」を参照してください。

手順

1. インストーラーに移動します。
   a. [bundled installer]
      ■

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

b. [online installer]

```
$ 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。

3.2.4. 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自動化ハブの両方が含まれます。

重要

- 自動化コントローラーと自動化ハブ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。

```
[automationcontroller]
tower.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

# 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
# Isolated Tower nodes automatically generate an RSA key for authentication;
# To disable this behavior, set this value to false
# isolated_key_generation=true
# SSL-related variables
# If set, this will install a custom CA certificate to the system trust store.
# custom_ca_cert=/path/to/ca.crt
# Certificate and key to install in nginx for the web UI and API
# web_server_ssl_cert=/path/to/tower.cert
# web_server_ssl_key=/path/to/tower.key
# Certificate and key to install in Automation Hub node
# automationhub_ssl_cert=/path/to/automationhub.cert
# automationhub_ssl_key=/path/to/automationhub.key
# Server-side SSL settings for PostgreSQL (when we are installing it).
# postgres_use_ssl=False
# postgres_ssl_cert=/path/to/pgsql.crt
# postgres_ssl_key=/path/to/pgsql.key
```

3.2.5. フラグおよび追加変数

自動化コントローラーのインストール時に、フラグと追加変数を渡すことができます。

:

表3.3 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う
-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。例: **./setup.sh -i my_awesome_inventory.yml -e matburt_is_country_gold=True — -K**.

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3.4 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 の無効化	**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/**

変数	説明	デフォルト
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時のみ変更してください)	**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時のみ変更してください)	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方法:

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

- nginx で処理する https を無効化する方法:

```
./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方法:

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

*

```
To override an inventory file used by passing it as an argument to the setup script:
```

3.2.6. 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
```

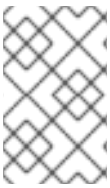
インストールが開始されます。

3.2.7. 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** ファイルの自動化コントローラーノードに指定した 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。

これで自動化コントローラーが初期設定可能になりました。

3.2.7.1. 次のステップ

- 顧客スクリプトや管理ジョブなどの管理については、『Automation Controller 管理ガイド』を参照してください。
- 自動化コントローラーのクイック設定については、『Automation Controller クイック設定ガイド』を参照してください。
- 『自動化コントローラーユーザーガイド』で自動化コントローラー機能を確認してください。

3.2.8. 次のステップ

- プロキシーサーバーを使用した自動化コントローラーの設定については、*5章Red Hat Ansible Automation Platform のプロキシーサポートの設定* を参照してください。
- Red Hat と共有する自動化コントローラーデータ分析を管理するには、xref:assembly-controlling-data-collection を参照してください。
- 自動化コントローラーの使用方法の詳細は、『Ansible Tower クイック設定ガイド』を参照してください。
- 自動化のハブ設定に関する完全なドキュメントは、Ansible Automation Platform の製品ドキュメント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、自動化ハブを使用して、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。自動化ハブ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、複数の自動化ハブ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自動化ハブとしてインストールできます。

4.1.1. 前提条件

- プラットフォームインストーラーを選択し、取得している。
- 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。

4.1.2. Red Hat Ansible Automation Platform インストール設定

自動化ハブのインストール時に、以下の設定を使用できます。

- **automationhub_importer_settings**: **galaxy-importer** に渡す設定および構成のディクショナリー。**/etc/galaxy-importer/galaxy-importer.cfg** で終了します。
- **automationhub_require_content_approval**: コレクションが利用可能になる前に、自動化ハブが承認メカニズムを適用するかどうか
- **automationhub_disable_https**: TLS を有効にして自動化ハブをデプロイするかどうか
- **automationhub_disable_hsts**: 自動化ハブが HTTP Strict Transport Security (HSTS) の Web セキュリティーポリシーメカニズムを有効にしてデプロイする必要があるかどうか。
- **automationhub_ssl_validate_certs**: デフォルトでは、プラットフォームは自己署名証明書を使用してデプロイするため、自動化ハブが自らを要求する際に証明書を検証するかどうか (デフォルト = False)
- **automationhub_ssl_cert**: **web_server_ssl_cert** と同じですが、自動化ハブの UI と API
- **automationhub_ssl_key**: 自動化ハブの UI と API 用の **web_server_ssl_key** と同じです。
- **automationhub_backup_collections**: オートメーションハブは、**/var/lib/pulp** のアーティファクトを提供します。デフォルトでは、これは **true** に設定され、自動化コントローラーはデフォ

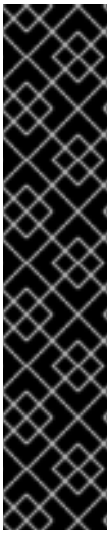
ルトでアーティファクトを自動的にバックアップします。パーティション (LVM、NFS、CephFS など) がそこにマウントされている場合、企業組織は常にバックアップされるようになります。この場合は、**automationhub_backup_collections = false** を設定すると、バックアップまたは復元プロセスでは **/var/lib/pulp** をバックアップまたは復元する必要はありません。

4.1.3. 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。

注記

- **外部データベース** の使用: インベントリーファイルのデータベースセクションが正しく設定されていることを確認します。
- **[automationhub]** グループに自動化ハブの情報を追加
- 自動化ハブと自動化コントローラーは、同じノードにインストールできません。
- 自動化コントローラーは、使用するデータベースのレプリケーションやフェイルオーバーを設定しません。自動化コントローラーは、所有するレプリケーションと連携する必要があります。
- パフォーマンス上の理由から、データベースサーバーは、自動化コントローラーサーバーと同じネットワーク上または同じデータセンターに置く必要があります。
- **既存のクラスターのアップグレード** の場合: クラスターのアップグレード時に、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ためにクラスターを再設定しないといけない必要があります。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省略すると、クラスターからインスタンスを削除するだけでは不十分です。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他に、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。それ以外の場合には、省略されたインスタンスまたはインスタンスグループはクラスターと通信し続けます。これにより、アップグレード中に Tower サービスに関する問題が生じる可能性があります。
- **クラスター化したインストールの場合**: クラスター設定を作成する場合は、**localhost** を全インスタンスのホスト名または IP アドレスに置き換える必要があります。すべてのノードまたはインスタンスには、このホスト名や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** を使用できません。また、すべてのノードがホスト名と同じ形式を使用する必要があります。



重要

- リモートマシンへの root アクセスが必要です。Ansible では、これはさまざまな方法で実行できます。
- **ansible_user=root ansible_ssh_pass="your_password_here"** インベントリーのホスト変数またはグループ変数
- **ansible_user=root ansible_ssh_private_key_file="path_to_your_keyfile.pem"** インベントリーのホスト変数またはグループ変数
- **ANSIBLE_BECOME_METHOD='sudo' ANSIBLE_BECOME=True ./setup.sh**

become プラグインの詳細は、「特権昇格について」を参照してください。

手順

1. インストーラーに移動します。
   a. [bundled installer]

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

   b. [online installer]

```
$ 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。

4.1.4. 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自動化ハブ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。

```
[automationcontroller]
host1
host11
host12
```

```
[automationhub]
host2

[database]
host3

[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'
```

4.1.5. フラグおよび追加変数

自動化コントローラーのインストール時に、フラグと追加変数を渡すことができます。

:

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う
-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。例: **./setup.sh -i my_awesome_inventory.yml -e matburt_is_country_gold=True — -K**.

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**
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 の無効化	**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時のみ変更してください)	**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時のみ変更してください)	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方法:

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

- nginx で処理する https を無効化する方法:

```
./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方法:

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

*

To override an inventory file used by passing it as an argument to the setup script:

4.1.6. 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
```

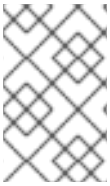
インストールが開始されます。

4.1.7. 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** ファイルの自動化コントローラーノードに指定した 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。

これで自動化コントローラーが初期設定可能になりました。

4.1.7.1. 次のステップ

- 顧客スクリプトや管理ジョブなどの管理については、『Automation Controller 管理ガイド』を参照してください。
- 自動化コントローラーのクイック設定については、『Automation Controller クイック設定ガイド』を参照してください。
- 『自動化コントローラーユーザーガイド』で自動化コントローラー機能を確認してください。

4.1.8. 次のステップ

- プロキシーサーバーを使用して自動化コントローラーを設定する場合は、「**Red Hat Ansible Automation Platform のプロキシー設定**」を参照してください。
- Red Hat と共有する情報を制御する方法は、「**自動化コントローラーからのユーザビリティーアナリティクスおよびデータ収集の管理**」を参照してください。
- 自動化コントローラーの使用方法の詳細は、『Ansible Tower クイック設定ガイド』を参照してください。
- 自動化のハブ設定に関する完全なドキュメントは、Ansible Automation Platform の製品ドキュメント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。本セクションでは、サポートされているプロキシー設定と設定方法を説明します。

5.1. 既知のプロキシーの設定

自動化コントローラーに到達できる既知のプロキシー一覧を設定できます。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。

5.1.1. 既知のプロキシー

Tower を **REMOTE_HOST_HEADERS = ['HTTP_X_FORWARDED_FOR', 'REMOTE_ADDR', 'REMOTE_HOST']** で設定している場合には、**X-Forwarded-For** の値が Tower 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 Tower にアクセスできる場合や、プロキシーがヘッダーを検証しない場合、**X-Forwarded-For** は比較的簡単になり、送信元の IP アドレスを偽装できます。 **REMOTE_HOST_HEADERS** 設定で **HTTP_X_FORWARDED_FOR** を使用すると、本質的にユーザーが持つべきではない特定のリソースにアクセスできる脆弱性が発生します。

脆弱性の例:

- ジョブテンプレートのホスト設定キー
- ジョブテンプレートのリンクされたインベントリーのホスト名または ansible_(ssh_)host
- ジョブテンプレートのプロビジョニングコールバックの URL

5.1.1.1. 既知のプロキシーの設定

設定 API から **PROXY_IP_WHITELIST** 設定を使用して許可される「既知のプロキシー」の一覧を設定できます。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。

重要

- **PROXY_IP_WHITELIST** 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。{ControllerNameStart} は、**PROXY_IP_WHITELIST** の IP アドレスとホスト名に依存して、**X-Forwarded-For** フィールドに攻撃を受けない値を提供します。
- 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_WHITELIST** を定義している

手順

1. 自動化コントローラーノードで、テキストエディターを使用して **/etc/tower/conf.d/remote_host_headers.py** を開きます。

追加情報

- 手順モジュールの内容に密接に関連するその他の資料へのリンクの箇条書きの一覧。
- 現在、モジュールには xref を含めることができないため、コレクションに他のコンテンツへのリンクを含めることができません。別のアセンブリーへのリンクが必要な場合は、このモジュールを含むアセンブリーに xref を追加します。
- 手順モジュールの作成に関する詳細は、Modular Documentation Reference Guide を参照してください。
- ファイル名、ID、およびタイトルには一貫性のあるシステムを使用します。ヒントについては、Modular Documentation Reference Guide の **Anchor Names and File Names** を参照してください。

5.2. リバースプロキシーの設定

HTTP_X_FORWARDED_FOR のヘッダーフィールドを使用したリバースプロキシーサーバーの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. 自動化コントローラーノードで、テキストエディターで **remote_host_headers.py** を開きます。

```
$ vi /etc/tower/conf.d/remote_host_headers.py`
```

2. 以下のように **REMOTE_HOST_HEADERS** を設定します。

```
`REMOTE_HOST_HEADERS = ['HTTP_X_FORWARDED_FOR', 'REMOTE_ADDR',
'REMOTE_HOST']`
```

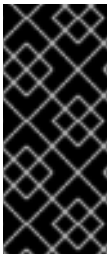
5.3. 自動化コントローラー WEBSOCKET 接続の設定

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、自動化コントローラーを設定できます。

5.3.1. コントローラーの自動化用の WebSocket 設定

{ControllerNameStart} 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。

{ControllerNameStart} 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。

5.3.1.1. 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、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6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

自動化コントローラーのユーザーインターフェースをオプトアウトまたは変更することで、自動化コントローラーからの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。

6.1. 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、自動化コントローラーに含まれており、自動化コントローラーユーザーが自動化コントローラー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。

自動化コントローラーのトライアルまたは自動化コントローラー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。

追加情報

- 詳細は、「Red Hat プライバシーポリシー」を参照してください。

6.1.1. 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ース** 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7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。

7.1. 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
```

7.2. 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
```

7.3. 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
```

7.4. 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
```

7.5. 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
```

7.6. OPENSTACK

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

7.7. 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
```

7.8. 自動コントローラー

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

第8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は、ホストでタスクが失敗した場合でも、通知された場合は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、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) 指定した場合には、このタグセットをスキップして Playbook を実行
- **start_at_task**: (文字列) 指定した場合には、この名前に一致するタスクで Playbook の実行を開始
- **started**: (日時) ジョブが開始の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には、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:**
  - **インベントリー**
    - **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、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には、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'}
 -----

:context: {parent-context}

:leveloffset!:
```