Red Hat Ansible Automation Platform 2.4

Red Hat Ansible Automation Platform 操作ガイド

Ansible Automation Platform インストールのスムーズなデプロイを確実にするためのインストール後の設定

Last Updated: 2024-06-17

Red Hat Ansible Automation Platform 2.4 Red Hat Ansible Automation Platform 操作ガイド

Ansible Automation Platform インストールのスムーズなデプロイを確実にするためのインストール後の設定

法律上の通知

概要

このガイドでは、Red Hat Ansible Automation Platform のインストール後のアクティビティーに関する指示とガイダンスを提供します。

目次

はじめに

Red Hat Ansible Automation Platform をインストールした後、デプロイメントがスムーズに実行するように、システムに追加の設定が必要になる場合があります。このガイドでは、Red Hat Ansible Automation Platform のインストール後に実行できる設定タスクの手順を説明します。

RED HAT ドキュメントへのフィードバック (英語のみ)

このドキュメント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 (https://access.redhat.com) に連絡し、**docs-product** 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。

第1章 RED HAT ANSIBLE AUTOMATION PLATFORM のアクティブ化

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

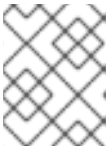
1. Ansible Automation Platform を起動するときに、Red Hat のお客様または Satellite の認証情報を使用します。
2. Red Hat Ansible Automation Platform インターフェイスを使用するか、Ansible Playbook で手動でサブスクリプションマニフェストファイルをアップロードします。

1.1. 認証情報を使用してアクティブ化する

Ansible Automation Platform を初めて起動すると、Ansible Automation Platform Subscription 画面が自動的に表示されます。Red Hat 認証情報を使用して、サブスクリプションを取得し、Ansible Automation Platform に直接インポートできます。

手順

1. Red Hat のユーザー名とパスワードを入力します。
2. **Get Subscriptions** をクリックします。



注記

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

3. 使用許諾契約書を確認し、**使用許諾契約書に同意します** を選択します。
4. 追跡と分析のオプションはデフォルトでオンになっています。これらの選択は、はるかに優れたユーザーエクスペリエンスを提供することで、Red Hat が製品を改善するのに役立ちます。オプションの選択を解除することで、オプトアウトできます。
5. **Submit** をクリックします。
6. サブスクリプションが受け入れられると、ライセンス画面が表示され、Ansible Automation Platform インターフェイスのダッシュボードに移動します。**Settings** アイコン ⚙ をクリックし、設定画面から **License** タブを選択すると、ライセンス画面に戻ることができます。

1.2. マニフェストファイルでアクティブ化する

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

前提条件

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

インターフェイスを使用したアップロード

1. マニフェストファイルを生成してダウンロードする手順を完了します。
2. Red Hat Ansible Automation Platform にログインします。
3. マニフェストファイルの入力をすぐに求められない場合は、**Settings → License** に移動します。
4. **Username** フィールドと **Password** フィールドが空であることを確認します。
5. **Browse** をクリックして、マニフェストファイルを選択します。
6. **Next** をクリックします。

注記

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

手動アップロード

Red Hat Ansible Automation Platform インターフェイスを使用してサブスクリプション情報を適用または更新できない場合は、**ansible.controller** コレクションの **license** モジュールを使用して、Ansible Playbook でサブスクリプションマニフェストを手動でアップロードできます。

```
- name: Set the license using a file
  license:
  manifest: "/tmp/my_manifest.zip"
```

第2章 マニフェストファイルの取得

サブスクリプションマニフェストは、Red Hat Subscription Management の サブスクリプション割り当て セクションで取得できます。サブスクリプションの割り当てを取得したら、そのマニフェストファイルをダウンロードしてアップロードし、Ansible Automation Platform をアクティブ化できます。

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

2.1. サブスクリプションの割り当ての作成

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

手順

1. サブスクリプションの割り当て ページで、**新規サブスクリプションの割り当て** をクリックします。
2. 割り当ての名前を入力し、後で検索できるようにします。
3. 管理アプリケーションとして、**Satellite 6.8** タイプを選択します。
4. **Create** をクリックします。

2.2. サブスクリプション割り当てへのサブスクリプションの追加

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

手順

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

検証

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

表示されるその他の重要な情報は次のとおりです。

自動化されたホスト

ライセンス数を消費するジョブによって自動化されたホスト数

インポートされたホスト

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

残りのホスト

合計ホスト数から自動化されたホストを差し引いた数

2.3. マニフェストファイルのダウンロード

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

手順

1. サブスクリプションの割り当て ページで、マニフェストを生成する **サブスクリプション割り当て** の名前をクリックします。
2. **Subscriptions** タブをクリックします。
3. **マニフェストのエクスポート** をクリックして、マニフェストファイルをダウンロードします。

注記

ファイルはデフォルトのダウンロードフォルダーに保存され、アップロードして Red Hat Ansible Automation Platform をアクティブ化できる ようになりました。

第3章 インストール後の手順

自動化を始めようとしている新しい Ansible Automation Platform ユーザーの方も、インストール済みの最新バージョンの Red Hat Ansible Automation Platform に以前の Ansible コンテンツを移行しようとしている既存の管理者の方も、次の手順を確認して、Ansible Automation Platform 2.4 の新機能を使い始めることをお勧めします。

3.1. データを ANSIBLE AUTOMATION PLATFORM 2.4 に移行する手順

プラットフォーム管理者が Ansible Automation Platform 2.4 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。

Ansible Automation Platform 2.4 へのアップグレードを完了するには、データを移行する必要があります。データを新しいインスタンスに移行するには、追加の手順が必要です。

3.1.1. 従来の仮想環境 (venvs) から自動化実行環境への移行

Ansible Automation Platform 2.4 は、カスタム Python 仮想環境 (venvs) よりも、自動化実行環境 (Ansible 自動化の実行と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。このコンポーネントには、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。

venvs を実行環境に移行するには、**awx-manage** コマンドを使用して元のインスタンスから venvs のリストを表示およびエクスポートしてから、**ansible-builder** を使用して実行環境を作成する必要があります。

関連情報

- 自動化実行環境へのアップグレード
- 実行環境の作成および消費

3.1.2. Ansible Builder を使用した Ansible Engine イメージの移行

Ansible Automation Platform 2.4 で使用するために以前の Ansible Engine イメージを移行するには、**ansible-builder** ツールを使用して、自動化実行環境で使用するイメージ (カスタムプラグインと依存関係を含む) を再ビルドするプロセスを自動化します。

関連情報

- Ansible Builder を使用して実行環境を構築する方法の詳細は、実行環境の作成および消費 を参照してください。

3.1.3. Ansible Core 2.13 への移行

ansible-core 2.13 にアップグレードする場合は、Playbook とプラグイン、または Ansible インフラストラクチャーの他の部分を、最新バージョンの ansible-core でサポートされるように更新する必要があります。

関連情報

Ansible-core 2.13 との互換性を確保するために Ansible コンテンツを更新する手順については、Ansible-core 2.13 Porting Guide を参照してください。

3.2. 実行環境イメージの場所の更新

Private Automation Hub を Ansible Automation Platform とは別にインストールした場合は、Private Automation Hub を参照するように実行環境イメージの場所を更新できます。

手順

1. **setup.sh** があるディレクトリーに移動します。
2. 次のコマンドを実行して、**./group_vars/automationcontroller** を作成します。

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

3. 次の内容を **./group_vars/automationcontroller** に貼り付けます。環境に合わせて設定を調整します。

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

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

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

4. **./setup.sh** スクリプトを実行します。

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

検証

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

3.3. 自動化メッシュの利点

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。

Ansible Automation Platform のバージョン 1.x から最新バージョンにアップグレードする場合は、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データを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。

関連情報

- 分離ノードから実行ノードに移行する方法は、Red Hat Ansible Automation Platform のアップグレードおよび移行ガイド を参照してください。
- 自動化メッシュと、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、以下を参照してください。
  - 仮想マシンベースのインストールは、仮想マシンベースのインストールに関する Red Hat Ansible Automation Platform 自動化メッシュガイド を参照してください。
  - Operator ベースのインストールは、Operator ベースのインストール用の Red Hat Ansible Automation Platform 自動化メッシュ を参照してください。

第4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ロールを果たします。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。

4.1. 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** に移動します。
2. **System** オプションのリストから **Miscellaneous System settings** を選択します。
3. **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。

4.2. 既知のプロキシー

Automation Controller 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、Automation Controller 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Automation Controller 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。既知のプロキシーリスト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。

4.2.1. 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** に移動し、 **System** オプションのリストから **Miscellaneous System settings** を選択します。
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

4.3. 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** に移動し、 **System** オプションのリストから **Miscellaneous System settings** を選択します。
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"
]
```

3. 以下の行を **/etc/tower/conf.d/custom.py** に追加して、アプリケーションが正しいヘッダーを使用していることを確認します。

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

4.4. スティッキーセッションの有効化

デフォルトでは、Application Load Balancer は、選択された負荷分散アルゴリズムに基づいて、登録済みのターゲットに各リクエストを個別にルーティングします。ロードバランサーの背後で Automation Hub の複数のインスタンスの実行時に認証エラーを回避するには、スティッキーセッションを有効にする必要があります。スティッキーセッションを有効にすると、ロードバランサーで設定された Cookie と一致するカスタムアプリケーション Cookie が設定され、スティッキーが有効になります。このカスタム Cookie には、アプリケーションで必要な Cookie 属性を含めることができます。

関連情報

- スティッキーセッションの有効化の詳細は、Application Load Balancer スティッキーセッション を参照してください。

免責事項: この注記に含まれる外部の Web サイトへのリンクは、お客様の利便性のみを目的として提供しています。Red Hat はリンクの内容を確認しておらず、コンテンツまたは可用性について責任を負わないものとします。外部 Web サイトへのリンクが含まれていても、Red Hat が Web サイトまたはその組織、製品、もしくはサービスを保証することを意味するものではありません。お客様は、外部サイトまたはコンテンツの使用 (または信頼) によって生じる損失または費用について、Red Hat が責任を負わないことに同意するものとします。

第5章 AUTOMATION CONTROLLER WEBSOCKET 接続の設定

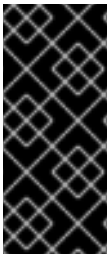
WebSocket の設定を nginx またはロードバランサー設定に合わせるために、Automation Controller を設定できます。

5.1. コントローラーの自動化用の WEBSOCKET 設定

Automation Controller ノードは、Websocket を介して相互接続され、Websocket が発行するすべてのメッセージをシステム全体に分散します。この設定セットアップにより、任意のブラウザークライアント Websocket が、任意の Automation Controller ノードで実行されている可能性がある任意のジョブにサブスクライブできるようになります。WebSocket クライアントは特定の Automation Controller ノードにルーティングされません。代わりに、すべての Automation Controller ノードが任意の Websocket 要求を処理できます。各 Automation Controller ノードは、全クライアントに宛てた全 Websocket メッセージを把握しておく必要があります。

すべての Automation Controller ノードの **/etc/tower/conf.d/websocket_config.py** で Websocket を設定でき、変更はサービスの再起動後に有効になります。

Automation Controller は、データベース内のインスタンスレコードを介して、他の Automation Controller ノードの検出を自動的に処理します。



重要

Automation Controller ノードは、(オープンインターネットではなく) プライベートで信頼できるサブネットを介して Websocket トラフィックをブロードキャストするように設計されています。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、Automation Controller ノード間で暗号化されずに送信されます。

5.1.1. 他の Automation Controller ノードの自動検出の設定

WebSocket 接続を設定して、Automation Controller 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 Automation Controller ノードの検出を自動的に処理できるようにします。

1. ポートとプロトコルの Automation Controller Websock 情報を編集し、Websocket 接続を確立するときに **True** または **False** で証明書を検証するかどうかを確認します。

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

2. 次のコマンドを使用して Automation Controller を再起動します。

```
$ automation-controller-service restart
```

第6章 ユーザビリティーアナリティクスおよび AUTOMATION CONTROLLER からのデータ収集の管理

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。

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、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。

6.1.1. Automation Controller からのデータ収集の制御

Settings メニューの **User Interface** タブで参加レベルを設定して、Automation Controller がデータを収集する方法を制御できます。

手順

1. Automation Controller にログインします。
2. **Settings** に移動し、**User Interface** オプションから **User Interface settings** を選択します。
3. **User Analytics Tracking State** ドロップダウンリストから目的のデータ収集レベルを選択します。
   - **Off**: データ収集を行いません。
   - **Anonymous**: 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。
   - **Detailed**: 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. **Save** をクリックして設定を適用するか、**Cancel** をクリックして変更を破棄します。

第7章 AUTOMATION CONTROLLER 設定ファイル内のプレーンテキストパスワードの暗号化

Automation Controller 設定ファイルに保存されるパスワードは、プレーンテキストで保存されます。**/etc/tower/conf.d/** ディレクトリーへのアクセス権を持つユーザーは、データベースへのアクセスに使用されるパスワードを表示できます。ディレクトリーへのアクセスは権限によって制御されるため、ディレクトリーは保護されていますが、セキュリティーに関する調査結果によっては、この保護は不十分であると考えられています。解決策は、パスワードを個別に暗号化することです。

7.1. POSTGRESQL パスワードハッシュの作成

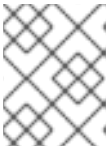
手順

1. Automation Controller ノードで、次のコマンドを実行します。

```
# awx-manage shell_plus
```

2. 続いて、python プロンプトから以下を実行します。

```
>>> from awx.main.utils import encrypt_value, get_encryption_key \
>>> postgres_secret = encrypt_value('$POSTGRES_PASS') \
>>> print(postgres_secret)
```



注記

$POSTGRES_PASS 変数を、暗号化する実際のプレーンテキストのパスワードに置き換えます。

出力は以下のようになります。

```
$encrypted$UTF8$AESCBC$Z0FBQUFBQmtLdGNRWXFjZGtkV1ZBR3hkNGVVbFFIU3hhY
21UT081eXFkR09aUWZLcG9TSmpndmZYQXFyRHVFQ3ZYSE15OUFuM1RHZHBqTFU3S
0MyNEo2Y2JWUURSYktsdmc9PQ==
```

3. これらのハッシュの完全な値をコピーして保存します。

- 次の例に示すように、ハッシュ値は **$encrypted$** で始まり、単なる文字列ではありません。

```
$encrypted$AESCBC$Z0FBQUFBQmNONU9BbGQ1VjJyNDJRVTRKaFRIR09Ib2U5TGd
aYVRfcXFXRjlmdmpZNjdoZVpEZ21QRWViMmNDOGJaM0dPeHN2b194NUxvQ1M5X3d
Sc1gxQ29TdDBKRkljWHc9PQ==
```

$*_PASS 値は、インベントリーファイル内ですでにプレーンテキストになっていることに注意してください。

これらの手順では、Automation Controller 設定ファイル内のプレーンテキストパスワードを置き換えるハッシュ値を提供します。

7.2. POSTGRES パスワードの暗号化

次の手順では、プレーンテキストのパスワードを暗号化された値に置き換えます。クラスター内の各ノードで次の手順を実行します。

手順

1. 以下を使用して **/etc/tower/conf.d/postgres.py** を編集します。

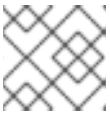
```
$ vim /etc/tower/conf.d/postgres.py
```

2. ファイルの先頭に次の行を追加します。

```
from awx.main.utils import decrypt_value, get_encryption_key
```

3. 'PASSWORD': の後にリストされているパスワード値を削除し、次の行に置き換えて、指定された **$encrytpted..** の値を独自のハッシュ値に置き換えます。

```
decrypt_value(get_encryption_key('value'),'$encrypted$AESCBC$Z0FBQUFBQmNONU9BbGQ1VjJyNDJRVTRKaFRIR09Ib2U5TGdaYVRfcXFXRjlmdmpZNjdoZVpEZ21QRWViMmNDOGJaM0dPeHN2b194NUxvQ1M5X3dSc1gxQ29TdDBKRkljWHc9PQ==='),
```



注記

このステップのハッシュ値は、**postgres_secret** の出力値です。

4. 完全な **postgres.py** は次のようになります。

```
# Ansible Automation platform controller database settings. from awx.main.utils import decrypt_value, get_encryption_key DATABASES = { 'default': { 'ATOMIC_REQUESTS': True, 'ENGINE': 'django.db.backends.postgresql', 'NAME': 'awx', 'USER': 'awx', 'PASSWORD': decrypt_value(get_encryption_key('value'),'$encrypted$AESCBC$Z0FBQUFBQmNONU9BbGQ1VjJyNDJRVTRKaFRIR09Ib2U5TGdaYVRfcXFXRjlmdmpZNjdoZVpEZ21QRWViMmNDOGJaM0dPeHN2b194NUxvQ1M5X3dSc1gxQ29TdDBKRkljWHc9PQ==='), 'HOST': '127.0.0.1', 'PORT': 5432, } }
```

7.3. AUTOMATION CONTROLLER サービスの再起動

手順

1. すべてのノードで暗号化が完了したら、以下を使用してクラスター全体でサービスの再起動を実行します。

```
# automation-controller-service restart
```

2. UI に移動し、すべてのノードでジョブを実行できることを確認します。

第8章 SSL 証明書の更新と変更

現在の SSL 証明書の有効期限が切れているか、まもなく期限切れになる場合は、Ansible Automation Platform で使用する SSL 証明書を更新または置き換えることができます。

新しいホストなどの新しい情報を使用して SSL 証明書を再生成する必要がある場合は、SSL 証明書を更新する必要があります。

内部の認証局によって署名された SSL 証明書を使用する場合は、SSL 証明書を置き換える必要があります。

8.1. 自己署名 SSL 証明書の更新

次の手順では、Automation Controller と Automation Hub の両方の新しい SSL 証明書を再生成します。

手順

1. **aap_service_regen_cert=true** をインベントリーファイルの **[all:vars]** セクションに追加します。

```
[all:vars]
aap_service_regen_cert=true
```

2. インストーラーを実行します。

検証

- Automation Controller の CA ファイルと server.crt ファイルを検証します。

```
openssl verify -CAfile ansible-automation-platform-managed-ca-cert.crt /etc/tower/tower.cert
openssl s_client -connect <AUTOMATION_HUB_URL>:443
```

- Automation Hub の CA ファイルと server.crt ファイルを検証します。

```
openssl verify -CAfile ansible-automation-platform-managed-ca-cert.crt
/etc/pulp/certs/pulp_webserver.crt
openssl s_client -connect <AUTOMATION_CONTROLLER_URL>:443
```

8.2. SSL 証明書の変更

SSL 証明書を変更するには、インベントリーファイルを編集してインストーラーを実行します。インストーラーは、すべての Ansible Automation Platform コンポーネントが動作していることを確認します。インストーラーの実行には時間がかかる場合があります。

あるいは、SSL 証明書を手動で変更することもできます。こちらのほうが迅速ですが、自動確認は行われません。

Red Hat では、Ansible Automation Platform インスタンスに変更を加える際にはインストーラーを使用することを推奨しています。

8.2.1. 前提条件

- 中間認証局がある場合は、それをサーバー証明書に追加する必要があります。
- Automation Controller と Automation Hub は両方とも NGINX を使用するため、サーバー証明書は PEM 形式である必要があります。
- 正しい証明書の順序を使用してください。最初にサーバー証明書を指定し、その後に中間認証局を指定します。

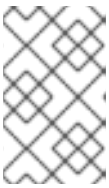
詳細は、NGINX ドキュメントの SSL 証明書のセクション を参照してください。

8.2.2. インストーラーを使用した SSL 証明書と鍵の変更

次の手順では、インベントリーファイル内の SSL 証明書と鍵を変更する方法について説明します。

手順

1. 新しい SSL 証明書と鍵を Ansible Automation Platform インストーラーに関連するパスにコピーします。
2. SSL 証明書と鍵の絶対パスをインベントリーファイルに追加します。これらの変数の設定に関するガイダンスについては、Red Hat Ansible Automation Platform インストールガイド の Automation Controller 変数、Automation Hub 変数、および Event-Driven Ansible Controller 変数 セクションを参照してください。
   - Automation Controller: **web_server_ssl_cert**、**web_server_ssl_key**、**custom_ca_cert**
   - Automation Hub: **automationhub_ssl_cert**、**automationhub_ssl_key**、**custom_ca_cert**
   - Event-Driven Ansible controller: **automationedacontroller_ssl_cert**、**automationedacontroller_ssl_key**、**custom_ca_cert**

   注記

   custom_ca_cert は、中間認証局に署名したルート認証局である必要があります。このファイルは **/etc/pki/ca-trust/source/anchors** にインストールされます。
3. インストーラーを実行します。

8.2.3. SSL 証明書の手動変更

8.2.3.1. Automation Controller での SSL 証明書と鍵の手動変更

次の手順では、Automation Controller で SSL 証明書と鍵を手動で変更する方法について説明します。

手順

1. 現在の SSL 証明書をバックアップします。

```
cp /etc/tower/tower.cert /etc/tower/tower.cert-$(date +%F)
```

2. 現在の鍵ファイルをバックアップします。

```
cp /etc/tower/tower.key /etc/tower/tower.key-$(date +%F)+
```

3. 新しい SSL 証明書を **/etc/tower/tower.cert** にコピーします。
4. 新しい鍵を **/etc/tower/tower.key** にコピーします。
5. SELinux コンテキストを復元します。

```
restorecon -v /etc/tower/tower.cert /etc/tower/tower.key
```

6. 証明書と鍵ファイルに適切なパーミッションを設定します。

```
chown root:awx /etc/tower/tower.cert /etc/tower/tower.key
chmod 0600 /etc/tower/tower.cert /etc/tower/tower.key
```

7. NGINX 設定をテストします。

```
nginx -t
```

8. NGINX をリロードします。

```
systemctl reload nginx.service
```

9. 新しい SSL 証明書と鍵がインストールされていることを確認します。

```
true | openssl s_client -showcerts -connect ${CONTROLLER_FQDN}:443
```

8.2.3.2. OpenShift Container Platform 上の Automation Controller の SSL 証明書と鍵の変更

次の手順では、OpenShift Container Platform で実行されている Automation Controller の SSL 証明書と鍵を変更する方法について説明します。

手順

1. 署名された SSL 証明書と鍵をセキュアな場所にコピーします。
2. OpenShift 内に TLS シークレットを作成します。

```
oc create secret tls ${CONTROLLER_INSTANCE}-certs-$(date +%F) --cert=/path/to/ssl.crt --
key=/path/to/ssl.key
```

3. Automation Controller カスタムリソースを変更して、**route_tls_secret** と新しいシークレットの名前を spec セクションに追加します。

```
oc edit automationcontroller/${CONTROLLER_INSTANCE}
```

```
...
spec:
  route_tls_secret: automation-controller-certs-2023-04-06
...
```

TLS シークレットの名前は任意です。この例では、Automation Controller インスタンスに適用される他の TLS シークレットと区別するために、シークレットを作成した日付のタイムスタンプが付けられています。

1. 変更が適用されるまで数分待ちます。
2. 新しい SSL 証明書と鍵がインストールされていることを確認します。

```
true | openssl s_client -showcerts -connect ${CONTROLLER_FQDN}:443
```

8.2.3.3. Event-Driven Ansible Controller での SSL 証明書と鍵の変更

次の手順では、Event-Driven Ansible Controller で SSL 証明書と鍵を手動で変更する方法について説明します。

手順

1. 現在の SSL 証明書をバックアップします。

```
cp /etc/ansible-automation-platform/eda/server.cert /etc/ansible-automation-platform/eda/server.cert-$(date +%F)
```

2. 現在の鍵ファイルをバックアップします。

```
cp /etc/ansible-automation-platform/eda/server.key /etc/ansible-automation-platform/eda/server.key-$(date +%F)
```

3. 新しい SSL 証明書を **/etc/ansible-automation-platform/eda/server.cert** にコピーします。
4. 新しい鍵を **/etc/ansible-automation-platform/eda/server.key** にコピーします。
5. SELinux コンテキストを復元します。

```
restorecon -v /etc/ansible-automation-platform/eda/server.cert /etc/ansible-automation-platform/eda/server.key
```

6. 証明書と鍵ファイルに適切なパーミッションを設定します。

```
chown root:eda /etc/ansible-automation-platform/eda/server.cert /etc/ansible-automation-platform/eda/server.key
```

```
chmod 0600 /etc/ansible-automation-platform/eda/server.cert /etc/ansible-automation-platform/eda/server.key
```

7. NGINX 設定をテストします。

```
nginx -t
```

8. NGINX をリロードします。

```
systemctl reload nginx.service
```

9. 新しい SSL 証明書と鍵がインストールされていることを確認します。

```
true | openssl s_client -showcerts -connect ${CONTROLLER_FQDN}:443
```

8.2.3.4. Automation Hub での SSL 証明書と鍵の手動変更

次の手順では、Automation Hub で SSL 証明書と鍵を手動で変更する方法について説明します。

手順

1. 現在の SSL 証明書をバックアップします。

```
cp /etc/pulp/certs/pulp_webserver.crt /etc/pulp/certs/pulp_webserver.crt-$(date +%F)
```

2. 現在の鍵ファイルをバックアップします。

```
cp /etc/pulp/certs/pulp_webserver.key /etc/pulp/certs/pulp_webserver.key-$(date +%F)
```

3. 新しい SSL 証明書を **/etc/pulp/certs/pulp_webserver.crt** にコピーします。
4. 新しい鍵を **/etc/pulp/certs/pulp_webserver.key** にコピーします。
5. SELinux コンテキストを復元します。

```
restorecon -v /etc/pulp/certs/pulp_webserver.crt /etc/pulp/certs/pulp_webserver.key
```

6. 証明書と鍵ファイルに適切なパーミッションを設定します。

```
chown root:pulp /etc/pulp/certs/pulp_webserver.crt /etc/pulp/certs/pulp_webserver.key
```

```
chmod 0600 /etc/pulp/certs/pulp_webserver.crt /etc/pulp/certs/pulp_webserver.key
```

7. NGINX 設定をテストします。

```
nginx -t
```

8. NGINX をリロードします。

```
systemctl reload nginx.service
```

9. 新しい SSL 証明書と鍵がインストールされていることを確認します。

```
true | openssl s_client -showcerts -connect ${CONTROLLER_FQDN}:443
```