Red Hat Ansible Automation Platform 2.4

Automation Controller 管理ガイド

Automation Controller の管理者ガイド

Last Updated: 2024-05-20

Red Hat Ansible Automation Platform 2.4 Automation Controller 管理ガイド

Automation Controller の管理者ガイド

法律上の通知

概要

カスタムスクリプト、管理ジョブなどを使用して Automation Controller を管理する方法について説明します。

目次

はじめに

Automation Controller 管理ガイドでは、カスタムスクリプト、管理ジョブなどを使用した Automation Controller の管理について説明します。DevOps エンジニアおよび管理者向けに作成された Automation Controller 管理ガイドでは、Automation Controller の使いやすいグラフィカルインターフェイスによる管理が必要なシステムに関する基本的な理解を前提としています。

多様性を受け入れるオープンソースの強化

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

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 AUTOMATION CONTROLLER のライセンス、更新、およびサポート

Automation Controller は、Red Hat Ansible Automation Platform 年間サブスクリプションの一部として提供されます。

Ansible はオープンソースソフトウェアのプロジェクトで、GNU 一般公衆利用許諾契約書バージョン 3 に基づいてライセンスが付与されます。これについては Ansible のソースコード に記載されています。

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。

詳細は、サブスクリプションの割り当て を参照してください。

1.1. 試用と評価

Automation Controller を実行するにはライセンスが必要です。まずは無料試用版ライセンスから始めることができます。

- Ansible Automation Platform の試用版ライセンスは http://ansible.com/license から入手できます。
- 試用版ライセンスや Automation Controller ソフトウェアの評価時には、サポートはありません。

1.2. サブスクリプションタイプ

Red Hat Ansible Automation Platform は、年間サブスクリプション契約をベースに、さまざまなサポートレベルおよびマシン数で提供されます。

- **Standard:**
  - あらゆる規模の環境の管理
  - エンタープライズサポート (週 5、1 日 8 時間) および SLA
  - メンテナンスおよびアップグレード込み
  - 製品サポート利用規約 で SLA を確認してください。
  - Red Hat サポートにおける重大度レベルの定義 を確認してください。
- **Premium:**
  - ミッションクリティカルな環境を含むあらゆる規模の環境の管理
  - プレミアムサポート (年中無休) および SLA
  - メンテナンスおよびアップグレード込み
  - 製品サポート利用規約 で SLA を確認してください。
  - Red Hat サポートにおける重大度レベルの定義 を確認してください。

すべてのサブスクリプションレベルには、Automation Controller、Ansible、および同プラットフォームの他のコンポーネントの通常の更新とリリースが含まれます。

詳細は、Red Hat カスタマーポータル または http://www.ansible.com/contact-us/ 経由で Ansible までお問い合わせください。

1.3. ライセンスでのノードの数え方

Automation Controller ライセンスでは、Red Hat Ansible Automation Platform サブスクリプション契約の一部として管理可能な管理対象ノードの数が定義されます。

通常のライセンスでは "ライセンス数: 500" と記載され、最大管理対象ノード数が 500 に設定されます。

管理対象ノードのライセンス要件に関する詳細は https://access.redhat.com/articles/3331481 を参照してください。

注記

Ansible は、ノード数を再利用したり、自動化されたホストをリセットしたりしません。

1.4. サブスクリプションの割り当て

Ansible Automation Platform をインストールする前に、有効な Ansible Automation Platform サブスクリプションを割り当てる **必要があります**。

注記

Red Hat アカウントで Simple Content Access Mode が有効になっている場合は、サブスクリプションを割り当てる必要はありません。ただし、Ansible Automation Platform をインストールする前に、**Red Hat Subscription Management**(RHSM) または Red Hat Satellite に登録する必要があります。

手順

1. サブスクリプションの **pool_id** を確認するために、次のコマンドを入力します。

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

このコマンドは以下を返します。

```
Subscription Name: Red Hat Ansible Automation Platform, Premium (5000 Managed Nodes)
Provides: Red Hat Ansible Engine
Red Hat Single Sign-On
Red Hat Ansible Automation Platform
SKU: MCT3695
Contract: ********
Pool ID: **********
Provides Management: No
Available: 4999
Suggested: 1
```

2. このサブスクリプションを割り当てるには、次のコマンドを入力します。

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

すべてのノードに割り当てられている場合は、リポジトリーが検出されます。

3. サブスクリプションが正常に割り当てられたかどうかを確認するには、次のコマンドを入力します。

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

4. このサブスクリプションを削除するには、次のコマンドを入力します。

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

1.5. コンポーネントライセンス

Automation Controller に含まれるコンポーネントのライセンス情報を表示するには、**/usr/share/doc/automation-controller-<version>/README** を参照してください。

ここで、**<version>** はインストールした Automation Controller のバージョンです。

特定のライセンスを表示するには、**/usr/share/doc/automation-controller-<version>/*.txt** を参照してください。

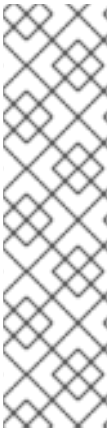
ここで、* は参照するライセンスファイル名です。

第2章 AUTOMATION CONTROLLER の起動、停止、および再起動

Automation Controller には、管理者ユーティリティースクリプト **automation-controller-service** が同梱されています。このスクリプトにより、現在の単一の Automation Controller ノード上で実行中のすべての Automation Controller サービスを起動、停止、再起動できます。統合インストールの場合、スクリプトにはメッセージキューコンポーネントとデータベースが含まれます。

外部データベースは、管理者が明示的に管理する必要があります。サービススクリプトは **/usr/bin/automation-controller-service** にあり、次のコマンドで呼び出すことができます。

```
root@localhost:~$ automation-controller-service restart
```



注記

クラスター化されたインストールの場合、**automation-controller-service restart** は、再起動するサービスとして PostgreSQL を含みません。これは、PostgreSQL が Automation Controller の外部に存在し、必ずしも再起動する必要がないためです。クラスター環境でサービスを再起動するには、代わりに **systemctl restart automation-controller** を使用してください。

また、特定の変更を維持するには、ローカルホストインストールの場合とは対照的に、シングルノードではなく各クラスターノードを再起動する必要があります。

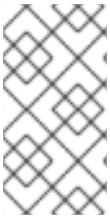
クラスター環境の詳細は、クラスタリング セクションを参照してください。

ディストリビューション固有のサービス管理コマンドを使用してサービススクリプトを呼び出すこともできます。ディストリビューションパッケージでは、多くの場合、サービスを管理するための同様のスクリプト (場合によっては init スクリプト) が提供されます。詳細は、お使いのディストリビューション固有のサービス管理システムを参照してください。

重要

コンテナー内で Automation Controller を実行する場合は、**automation-controller-service** スクリプトを使用しないでください。代わりに、コンテナー環境を使用して Pod を再起動します。

第3章 カスタムインベントリースクリプト



注記

インベントリースクリプトは廃止されました。

詳細は、**Automation Controller ユーザーガイド** の 古いインベントリースクリプトをエクスポートする を参照してください。

カスタムインベントリースクリプトを使用している場合は、これらのスクリプトをプロジェクトから取得するように移行してください。詳細は、**Automation Controller ユーザーガイド** の インベントリーファイルのインポート および インベントリーソース を参照してください。

インベントリーファイルをセットアップしている場合は、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 を参照して、当該セットアップに特化した例を見つけてください。

実行環境に移行する場合は、以下を参照してください。

- 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
- 自動化メッシュの設計パターン
- トポロジーを検証する場合は、**Ansible Automation Platform Upgrade and Migration Guide** の Mesh Topology を参照してください。

仮想マシンベースのインストールにおける自動化メッシュの詳細は、仮想マシンベースのインストール向け 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における自動化メッシュの詳細は、Operator ベースのインストール向け Red Hat Ansible Automation Platform 自動化メッシュ を参照してください。

メッシュトポロジーがすでにセットアップされており、ノードタイプ、ノードの健全性、および各ノードに関する具体的な情報を表示する必要がある場合は、トポロジービューアー を参照してください。

第4章 インベントリーファイルのインポート

Automation Controller を使用すると、インベントリーファイルを最初から作成するのではなく、ソースコントロールから選択することができます。この機能はカスタムインベントリースクリプトの場合と同様ですが、ブラウザーでコンテンツを編集するのではなく、ソースコントロールからコンテンツを取得する点で異なります。これは、ファイルが編集不可であることを意味します。ソースでインベントリーが更新されると、それに応じてプロジェクト内のインベントリー (**group_vars** ファイルと **host_vars** ファイル、またはそれらに関連付けられたディレクトリーを含む) も更新されます。SCM タイプは、インベントリーファイルとスクリプトの両方を使用できます。インベントリーファイルとカスタムインベントリータイプはどちらもスクリプトを使用します。

インポートされたホストには、デフォルトで **imported** という説明があります。これは、特定のホストで **_awx_description** 変数を設定することでオーバーライドできます。たとえば、ソースの **.ini** ファイルからインポートする場合は、次のホスト変数を追加できます。

```
[main]
127.0.0.1 _awx_description="my host 1"
127.0.0.2 _awx_description="my host 2"
```

同様に、グループの説明もデフォルトでは **imported** ですが、**_awx_description** でオーバーライドすることもできます。

ソースコントロールで古いインベントリースクリプトを使用するには、 **Automation Controller ユーザーガイド** の 古いインベントリースクリプトをエクスポートする を参照してください。

4.1. カスタムの動的インベントリースクリプト

バージョン管理に保存されているカスタム動的インベントリースクリプトをインポートして実行できます。これにより、インベントリースクリプトの変更がより簡単になります。スクリプトをコピーして Automation Controller にペーストする必要はなく、ソースコントロールから直接プルして実行します。スクリプトは、そのタスクに必要な認証情報を処理する必要があります。スクリプトに必要な Python ライブラリーをインストールすることが必要です (カスタム動的インベントリースクリプトにも同じ要件があります)。この点は、ユーザー定義のインベントリーソーススクリプトと SCM ソースの両方に当てはまります。この理由は、両方とも、Playbook に関連する Ansible **virtualenv** 要件にさらされるためです。

SCM インベントリーソースを編集する場合は環境変数を指定できます。スクリプトによっては、これで十分です。ただし、この方法では、クラウドプロバイダーやインベントリーへのアクセスを可能にするシークレット情報がセキュアに保存されません。

より良い方法は、使用するインベントリースクリプト用に新規の認証情報タイプを作成することです。当該認証情報タイプでは、必要な入力タイプをすべて指定する必要があります。当該タイプの認証情報を作成すると、シークレットは暗号化された形式で保存されます。その認証情報をインベントリーソースに適用すると、スクリプトはそれらの入力にアクセスできるようになります。

詳細は、「Automation Controller ユーザーガイド」の カスタム認証情報タイプ を参照してください。

4.2. SCM インベントリーソースのフィールド

使用するソースのフィールドは以下のとおりです。

- **source_project**: 使用するプロジェクト。
- **source_path**: ディレクトリーまたはファイルを示すプロジェクト内の相対パス。空白のままにしても、"" はプロジェクトの root ディレクトリーを示す相対パスを表します。

- **source_vars**: "file" タイプのインベントリーソースに設定されている場合には、実行時に環境変数に渡されます。

補足

- プロジェクトを更新すると、そのプロジェクトが使用されているインベントリーの更新が自動的にトリガーされます。
- プロジェクトの更新は、インベントリーソースの作成直後にスケジュールされます。
- インベントリー更新もプロジェクト更新も、関連ジョブの実行中にブロックされることはありません。
- 大規模なプロジェクト (約 10 GB) がある場合、**/tmp** のディスク領域が問題になる可能性があります。

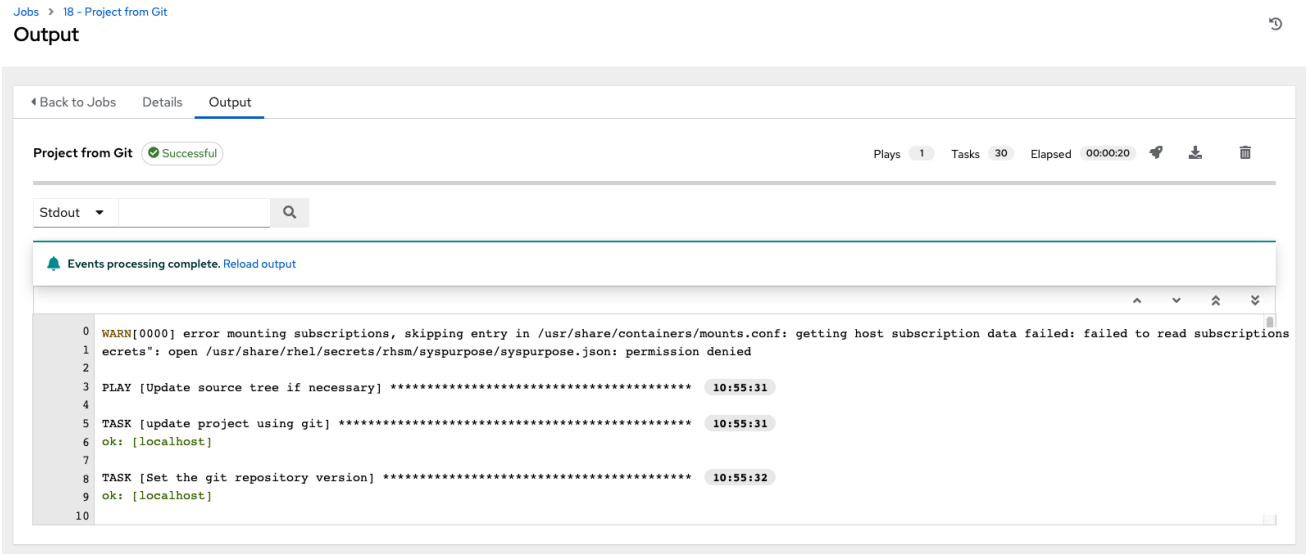
Automation Controller UI の **Create Inventory Source** ページから、手動で場所を指定できます。インベントリーソースの作成手順については、ソースの追加 を参照してください。

プロジェクトの更新時には、最新の SCM 情報を使用するようにリストを更新します。インベントリーソースが SCM インベントリーソースとしてプロジェクトを使用しない場合、更新時にインベントリーのリストが更新されない可能性があります。

インベントリーが SCM ソースを含む場合、インベントリー更新の **Job Details** ページには、プロジェクト更新のステータスインジケータとプロジェクト名が表示されます。

ステータスインジケーターはプロジェクト更新ジョブにリンクします。

プロジェクト名はプロジェクトにリンクします。



関連ジョブの実行中にインベントリーの更新を実行できます。

4.2.1. サポートされるファイルの構文

Automation Controller は、Ansible の **ansible-inventory** モジュールを使用してインベントリーファイルを処理し、Automation Controller で必要となる有効なインベントリー構文をすべてサポートします。

第5章 複数の認証情報の割り当て

Automation Controller は、ジョブテンプレートに対する 0 個以上の認証情報の割り当てをサポートします。

5.1. 背景情報

Automation Controller v3.3 より前のバージョンでは、ジョブテンプレートには認証情報に関する次の要件がありました。

- 全ジョブテンプレート (およびジョブ) は、マシン/SSH または Vault 認証情報 (またはそのいずれか) を **1つ** だけ指定する必要がありました。
- 全ジョブテンプレート (およびジョブ) は、0 以上の "追加の" 認証情報を指定することも可能でした。
- 追加の認証情報は、環境変数 (例: **AWS_ACCESS_KEY_ID**) 経由で外部サービスへの認証に使用可能な "Cloud" および "Network" 認証情報を表していました。

このモデルでは、ジョブテンプレートで認証情報を指定するためにさまざまなインターフェイス要素が必要で、また複数の Vault 認証情報を Playbook の実行に関連付ける機能がありませんでした。これは、Ansible 2.4 以降の Ansible Core でサポートされているユースケースです。

また、このモデルは、要件を満たすために "ダミーの" マシン/SSH 認証情報をジョブテンプレートに割り当てる必要があるなど、特定の Playbook 実行のワークフローに対する障害となっていました。

5.2. 重要な変更点

すべての Automation Controller 4.4 ジョブテンプレートには、認証情報の割り当て用にインターフェイスが 1 つ用意されています。

API エンドポイントから、以下を実行します。

```
GET /api/v2/job_templates/N/credentials/
```

POST 要求を使用して認証情報の割り当てや割り当て解除ができます。これは、非推奨になった **extra_credentials** エンドポイントの動作に似ています。

```
POST /api/v2/job_templates/N/credentials/ {'associate': true, 'id': 'X'}
POST /api/v2/job_templates/N/credentials/ {'disassociate': true, 'id': 'Y'}
```

このモデルでは、ジョブテンプレートに認証情報が割り当てられて **いない** 場合でも、ジョブテンプレートは有効であるとみなされます。また、このモデルは、ジョブテンプレートに複数の Vault 認証情報を割り当てることも可能にします。

5.3. 起動時の考慮事項

Automation Controller v3.3 より前のバージョンでは、ジョブテンプレートは設定可能な属性 **ask_credential_on_launch** を使用していました。この値を起動時に使用して、起動に必要な認証情報の値としてどの値が欠落しているかを判断していました。これは、認証情報の最小要件を満たすようにマシンまたは SSH 認証情報を指定する手段でした。

統合認証情報リストモデルでは、この属性はまだ存在しますが、認証情報は必要なくなりました。**ask_credential_on_launch** が **true** の場合、起動時に認証情報のリストを指定して、ジョブテンプレートで定義されている認証情報をオーバーライドできます。以下に例を示します。

```
POST /api/v2/job_templates/N/launch/ {'credentials': [A, B, C]}`
```

ask_credential_on_launch が **false** の場合、**POST /api/v2/job_templates/N/launch/** で指定されているカスタム認証情報が無視されます。

このモデルでは、**ask_credential_on_launch** は、起動時にユーザーに対して (任意で) 変更を行うかどうかのプロンプトを表示するように、API クライアントにシグナルを送るためだけのものです。

5.4. 複数の VAULT 認証情報

1 つのジョブに複数の認証情報を割り当てることができるため、ジョブテンプレートの実行時に復号化する複数の Vault 認証情報を指定できます。この機能は、Vault パスワードの管理 のサポートを反映しています。

Vault 認証情報に、オプションのフィールド **vault_id** が追加されました。これは、**ansible-playbook** の **--vault-id** 引数に似ています。

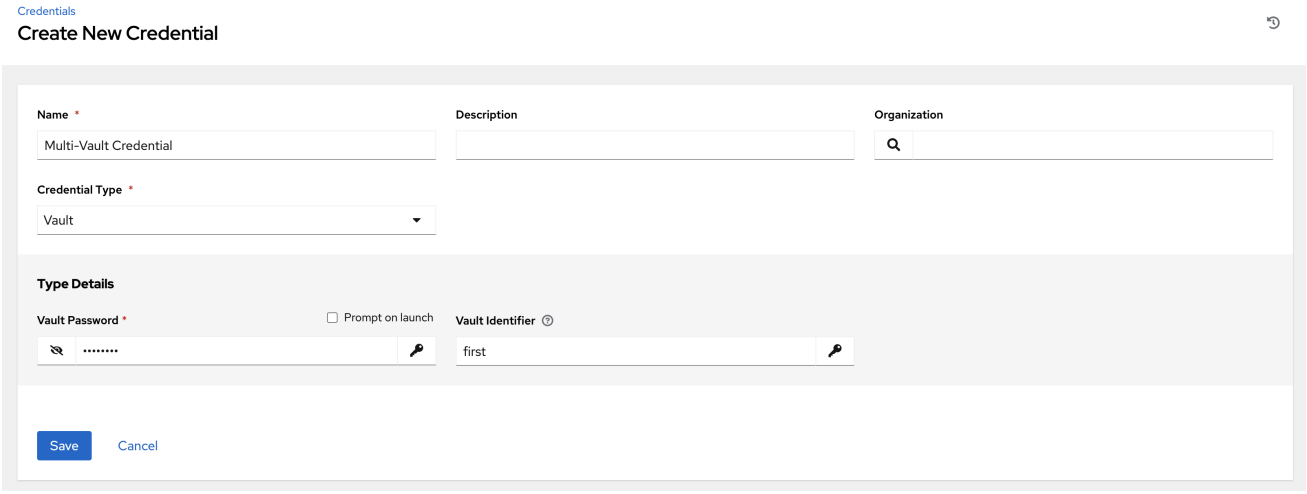
複数の Vault パスワードを使用する Playbook を実行するには、次の手順を使用します。

手順

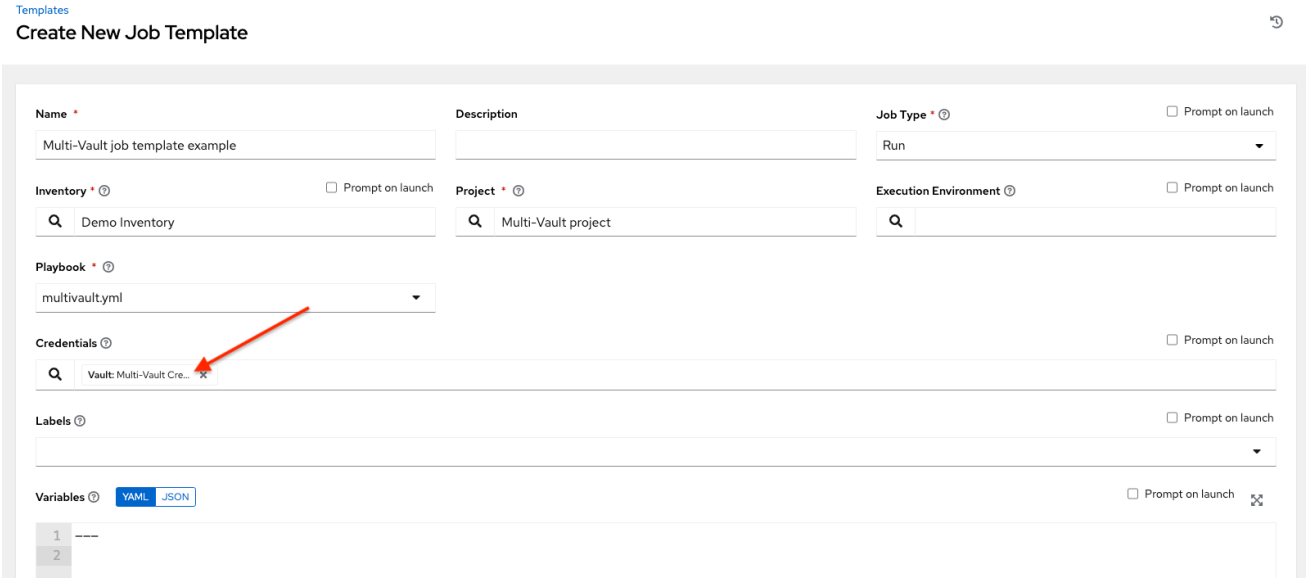
1. Automation Controller で Vault パスワードごとに Vault 認証情報を作成します。
2. 認証情報のフィールドとして Vault ID を指定し、パスワードを入力します (暗号化されて保存されます)。
3. 新しい認証情報のエンドポイントを使用して、ジョブテンプレートに複数の Vault 認証情報を割り当てます。

```
POST /api/v2/job_templates/N/credentials/

{
    'associate': true,
    'id': X
}
```

あるいは、Automation Controller UI の **Create Credential** ページで、同じ割り当てを実行することもできます。



この例では、作成した認証情報は、Vault 識別子 ("first") とパスワードのペアで使用されるシークレットを指定します。次の例のように、この認証情報をジョブテンプレートで使用する場合、この Vault ID "first" に関連付けられたシークレットのみが復号化されます。



従来の方法で、1 つの大きいファイルにすべてのシークレットを区別なく指定して Playbook をセットアップした場合には、Vault 認証情報の設定時には、**Vault Identifier** フィールドは空白のままにしてください。

5.4.1. プロンプト付きの Vault 認証情報

Vault 認証情報のパスワードに **Prompt on launch** とマークされている場合には、関連するジョブテンプレートの起動エンドポイントが **passwords_needed_to_start** パラメーターを使用して必要な Vault パスワードを通信します。

```
GET /api/v2/job_templates/N/launch/
{
    'passwords_needed_to_start': [
        'vault_password.X',
        'vault_password.Y',
    ]
}
```

ここで、**X** と **Y** は、関連する Vault 認証情報のプライマリーキーです。

```
POST /api/v2/job_templates/N/launch/
{
    'credential_passwords': {
        'vault_password.X': 'first-vault-password'
        'vault_password.Y': 'second-vault-password'
    }
}
```

5.4.2. 認証情報のリンク

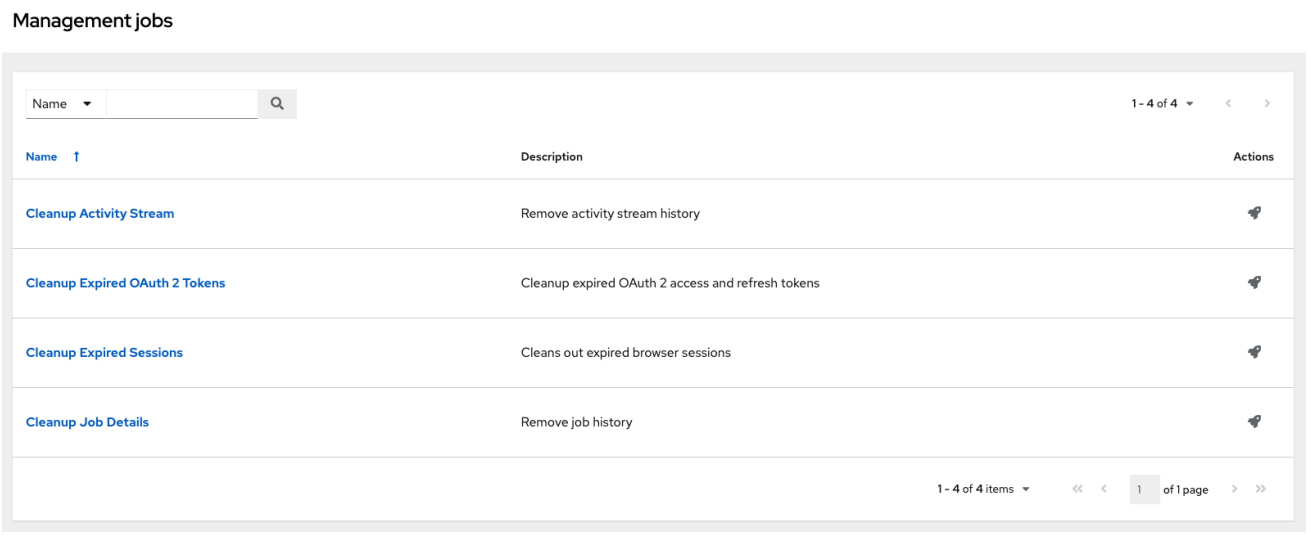
Automation Controller に機密の認証情報をアップロードする代わりに、認証情報フィールドを外部システムにリンクし、これを使用して Playbook を実行できます。

詳細は、Automation Controller ユーザーガイドの シークレット管理システム を参照してください。

第6章 管理ジョブ

Management Jobs は、システム追跡情報、トークン、ジョブ履歴、アクティビティーストリームなどの古いデータを Automation Controller から消去するのに役立ちます。これは、特定の保持ポリシーがある場合、または Automation Controller データベースで使用されるストレージを減らす必要がある場合に使用できます。

ナビゲーションパネルから、**Administration → Management Jobs** を選択します。

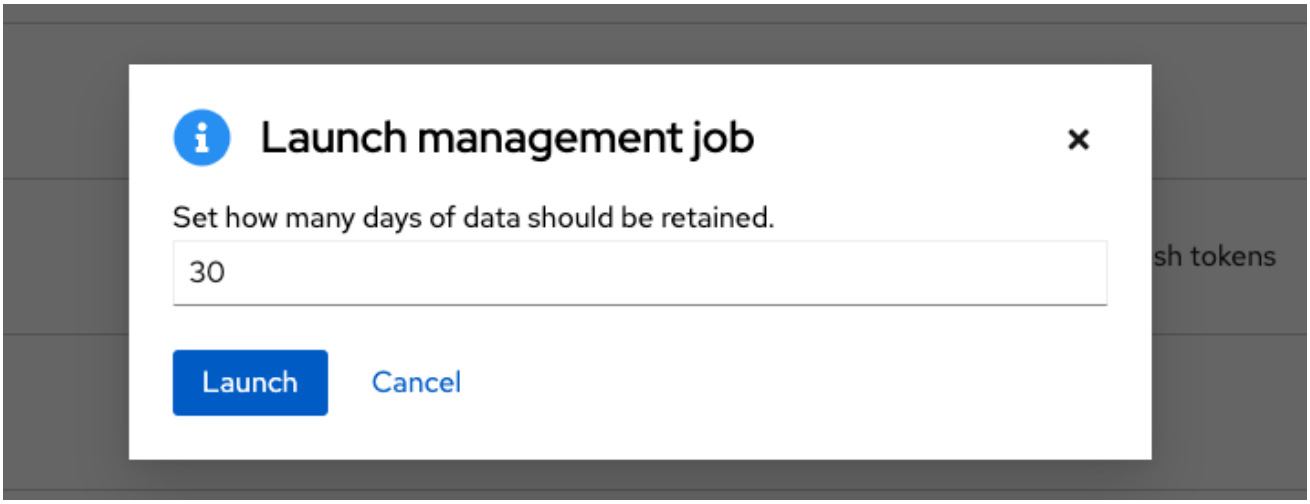


次のジョブタイプをスケジュールして起動できます。

- **Cleanup Activity Stream**: 指定した日数より前のアクティビティーストリーム履歴を削除します。
- **Cleanup Expired OAuth 2 Tokens** 期限切れの OAuth 2 アクセストークンを削除し、トークンを更新します。
- **Cleanup Expired Sessions**: データベースから期限切れのブラウザーセッションを削除します。
- **Cleanup Job Details**: 指定した日数より前のジョブ履歴を削除します。

6.1. 以前のアクティビティーストリームデータの削除

以前のアクティビティーストリームデータを削除するには、**Cleanup Activity Stream** の横にある起動アイコン ( ) をクリックします。



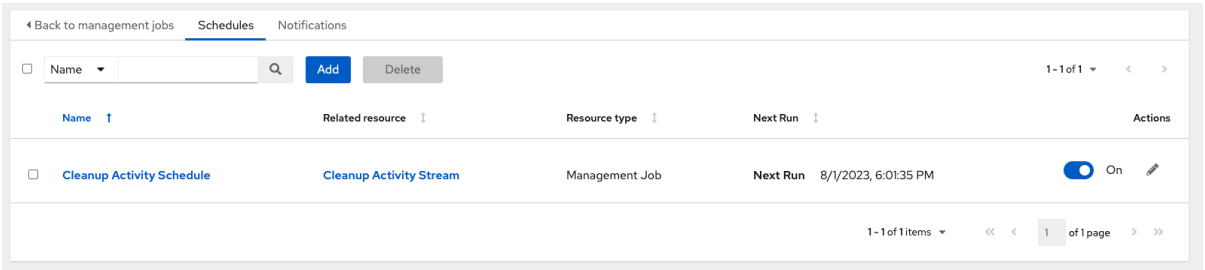
データを保存する日数を入力して **Launch** をクリックします。

6.1.1. 削除のスケジューリング

削除対象としてマークされたデータを削除するスケジュールを確認または設定するには、次の手順を使用します。

手順

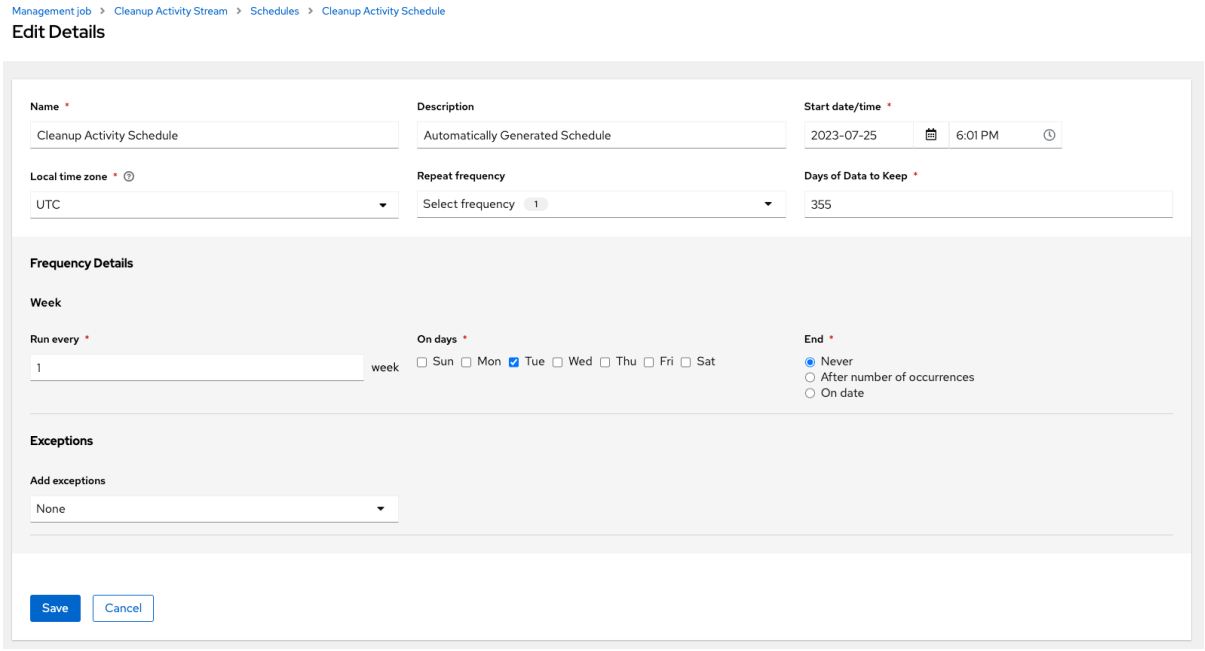
1. 特定のクリーンアップジョブの場合、**Schedules** タブをクリックします。



   ON/OFF 切り替えボタンを使用して、スケジュールされた管理ジョブをオンまたはオフにすることができます。

2. ジョブ名 (この例では "Cleanup Activity Schedule") をクリックして、スケジュール設定を確認します。

3. **Edit** をクリックして変更します。**Add** をクリックして、この管理ジョブの新しいスケジュールを作成することもできます。



4. 以下のフィールドに適切な情報を入力して、**Save** をクリックします。

    - **Name**: 必須
    - **Start Date**: 必須
    - **Start Time**: 必須
    - **Local time zone**: 入力した Start Time はこのタイムゾーンの時間になります。
    - **Repeat frequency**: 更新頻度を変更すると、適切なオプションが表示されます。例外を指定して、対象外とするデータを指定することもできます。
    - **Days of Data to Keep**: 必須。保持するデータの量を指定します。

Details タブでは、選択した Local Time Zone でのスケジュールの詳細と、スケジュール実行のリストを表示します。

注記

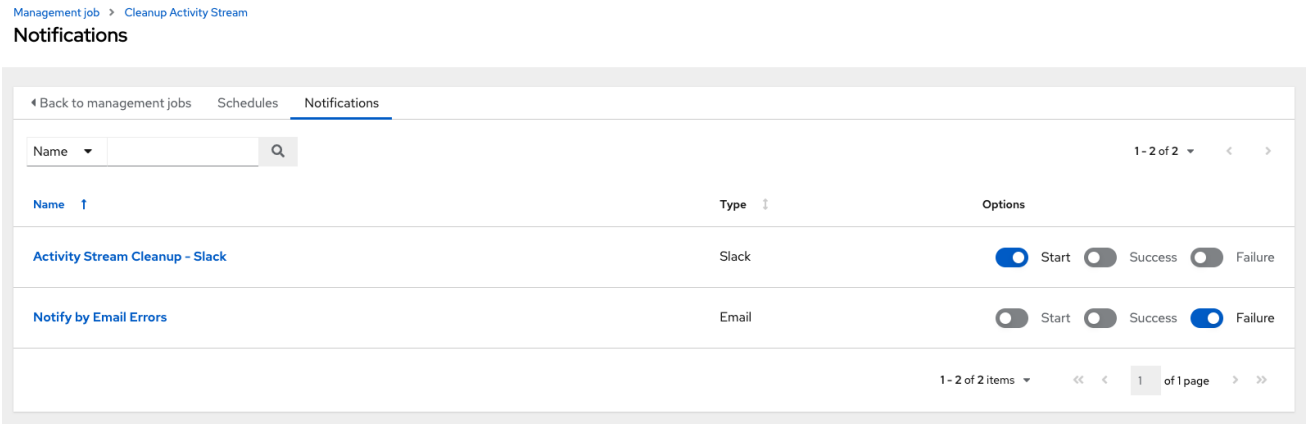
ジョブは UTC でスケジュールされます。夏時間への切り替えまたは夏時間からの切り替えが発生すると、特定の時刻に実行される繰り返しジョブは、ローカルタイムゾーンに合わせて移動する可能性があります。

6.1.2. 通知の設定

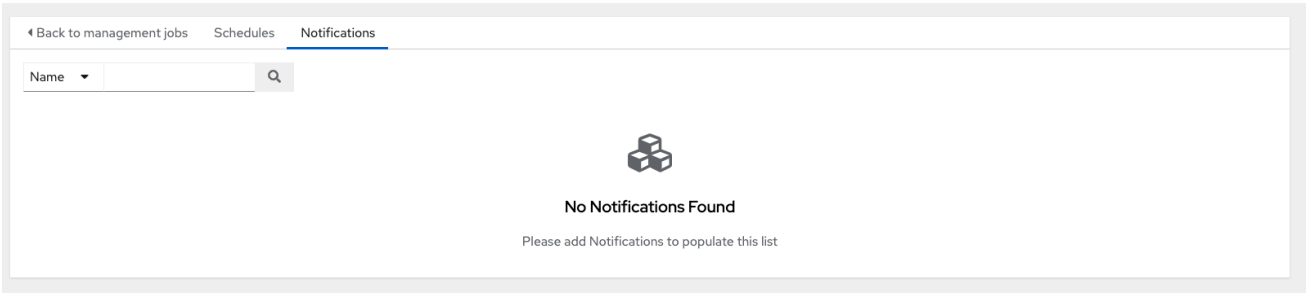
管理ジョブに関連する通知を確認または設定するには、次の手順を使用します。

手順

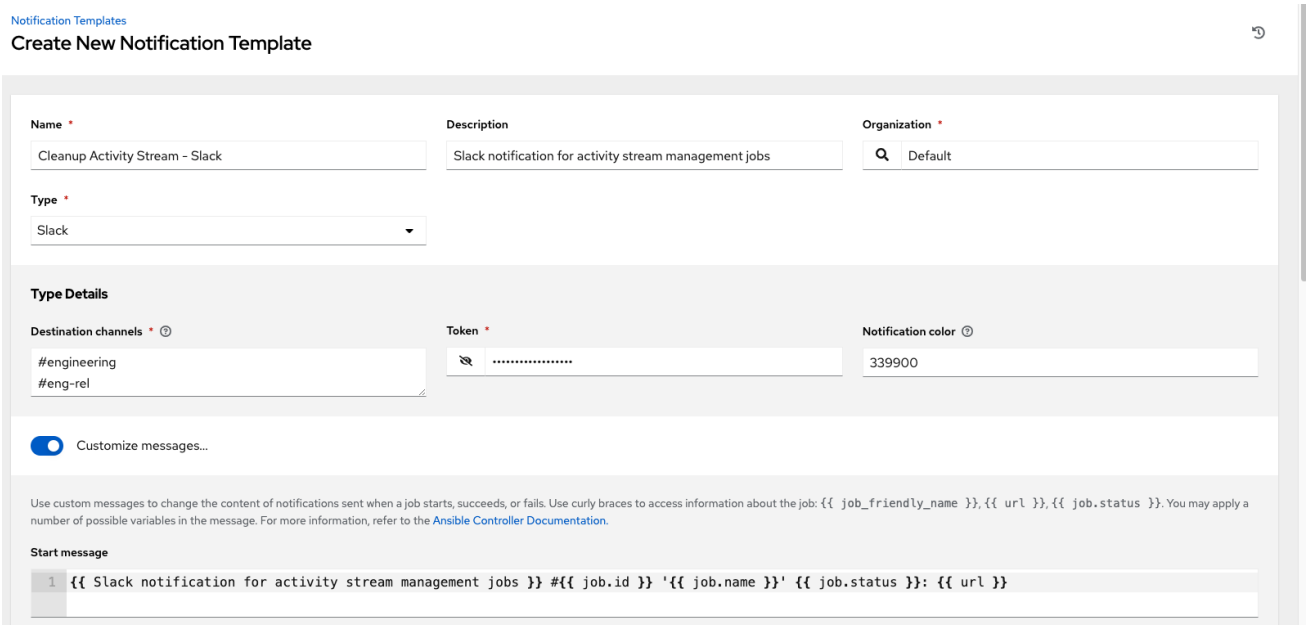
- 特定のクリーンアップジョブについては、**Notifications** タブを選択します。



通知が存在しない場合、詳細は、**Automation Controller ユーザーガイド** の 通知 を参照してください。



以下に、詳細が指定された通知の例を示します。



6.2. 期限切れの OAUTH2 トークンのクリーンアップ

期限切れの OAuth2 トークンを削除するには、**Cleanup Expired OAuth2 Tokens** の横にある起動アイコン ( ) をクリックします。

期限切れの OAuth2 トークンの消去スケジュールを確認または設定するには、アクティビティーストリーム管理ジョブ向けに記載されているのと同じ手順を実行します。

詳細は、削除のスケジューリング を参照してください。

アクティビティーストリーム管理ジョブの 通知の設定 で説明されているのと同じ方法で、この管理ジョブに関連する通知を設定または確認することもできます。

詳細は、**Automation Controller ユーザーガイド** の 通知 を参照してください。

6.2.1. 期限切れセッションのクリーンアップ

期限切れのセッションを削除するには、**Cleanup Expired Sessions** の横にある起動アイコン ( ) をクリックします。

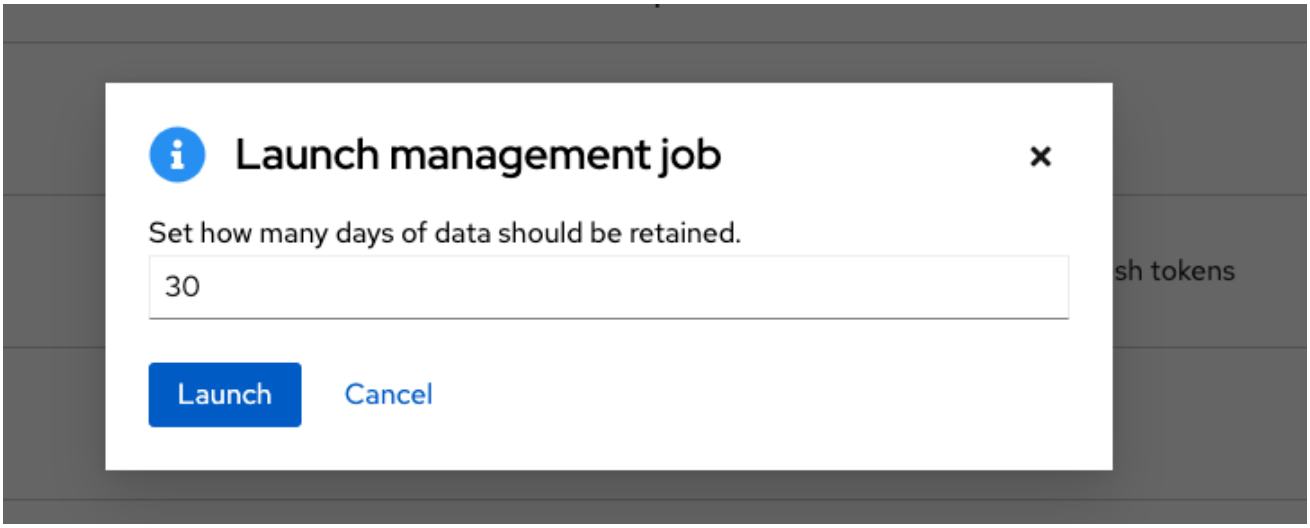
期限切れのセッションの消去スケジュールを確認または設定するには、アクティビティーストリーム管理ジョブ向けに記載されているのと同じ手順を実行します。詳細は、削除のスケジューリング を参照してください。

アクティビティーストリーム管理ジョブの 通知 で説明されているのと同じ方法で、この管理ジョブに関連する通知を設定または確認することもできます。

詳細は、**Automation Controller ユーザーガイド** の 通知 を参照してください。

6.2.2. 以前のジョブ履歴の削除

特定の日数以前のジョブ履歴を削除するには、**Cleanup Job Details** の横にある起動アイコン ( ) をクリックします。



データを保存する日数を入力して **Launch** をクリックします。

注記

Automation Controller リソース (プロジェクト、ジョブテンプレートなど) の最初のジョブ実行は、保持値に関係なく、**Cleanup Job Details** から除外されます。

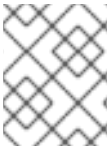
古いジョブ履歴の消去スケジュールを確認または設定するには、アクティビティーストリーム管理ジョブ向けに記載されているのと同じ手順を実行します。

詳細は、削除のスケジューリング を参照してください。

アクティビティーストリーム管理ジョブの 通知 で説明されているのと同じ方法で、この管理ジョブに関連する通知を設定または確認することもできます。詳細は、**Automation Controller ユーザーガイド** の 通知 を参照してください。

第7章 クラスタリング

クラスタリングでは、ホスト間で負荷が共有されます。各インスタンスは、UI および API アクセスのエントリーポイントとして機能できる必要があります。これにより、Automation Controller 管理者は必要な数のインスタンスの手前でロードバランサーを使用し、良好なデータ可視性を維持できるようになります。



注記

負荷分散はオプションで、必要に応じて 1 つまたはすべてのインスタンスで受信することも可能です。

各インスタンスは、Automation Controller のクラスターに参加して、ジョブを実行する機能を拡張できる必要があります。これは、ジョブをどこで実行するかを指定するのではなく、ジョブをどこでも実行できる簡単なシステムです。また、クラスター化されたインスタンスを、インスタンスグループ と呼ばれる異なるプールまたはキューにグループ化することもできます。

Ansible Automation Platform は、Kubernetes を使用したコンテナーベースのクラスターをサポートしています。つまり、このプラットフォームに新しい Automation Controller インスタンスを、機能の変更や転換を行うことなくインストールできます。Kubernetes コンテナーを指定するインスタンスグループを作成できます。詳細は、コンテナーおよびインスタンスグループ セクションを参照してください。

サポート対象オペレーティングシステム

クラスター環境の確立には、以下のオペレーティングシステムがサポートされます。

- Red Hat Enterprise Linux 8.7 以降である。

注記

OpenShift での分離インスタンスと Automation Controller の実行の組み合わせはサポートされません。

7.1. セットアップに関する考慮事項

クラスターの初期セットアップについて説明します。既存のクラスターをアップグレードするには、**Ansible Automation Platform Upgrade and Migration Guide**の Upgrade Planning を参照してください。

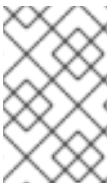
新規のクラスタリング環境では、次の重要な留意事項に注意してください。

- PostgreSQL はスタンドアロンインスタンスであり、クラスター化されていません。Automation Controller は、(ユーザーがスタンバイレプリカを設定した場合) レプリカ設定やデータベースフェイルオーバーを管理しません。
- クラスターを起動する場合は、データベースノードをスタンドアロンサーバーにする必要があります。PostgreSQL は Automation Controller ノードの 1 つにインストールしないでください。
- PgBouncer は、Automation Controller を使用した接続プールには推奨されません。Automation Controller は、さまざまなコンポーネント間でのメッセージの送信に **pg_notify** を使用しているため、PgBouncer をトランザクションプールモードですぐに使用することはできません。

- クラスターでサポートされるインスタンスの最大数は 20 です。
- 全インスタンスは他のすべてのインスタンスからアクセス可能であり、データベースにアクセスできる必要があります。ホストが安定したアドレスまたはホスト名を持つことも重要です (Automation Controller ホストの設定に応じて異なります)。
- 全インスタンスは、地理的に近い場所に配置する必要があります。インスタンス間はレイテンシーが低く信頼性のある接続を使用します。
- クラスター環境にアップグレードするには、プライマリーインスタンスがインベントリーの **default** グループに属し、**default** グループの最初のホストとしてリストされている必要があります。
- 手動のプロジェクトでは、顧客が手動で全インスタンスを同期して、一度に全インスタンスを更新する必要があります。
- プラットフォームデプロイメントの **inventory** ファイルは、保存または永続化する必要があります。新規インスタンスをプロビジョニングする場合は、パスワードと設定オプション、およびホスト名をインストーラーで使用できるようにする必要があります。

7.2. INSTALLING APICAST

新規インスタンスのプロビジョニングには、**inventory** ファイルの更新と設定用 Playbook の再実行が含まれます。インベントリーファイルには、クラスターのインストール時に使用するすべてのパスワードと情報が含まれていることが重要です。含まれていない場合には、他のインスタンスが再設定される場合があります。インベントリーファイルには、単一のインベントリーグループ **automationcontroller** が含まれています。



注記

インスタンスはすべて、ジョブの起動先や Playbook イベントの処理、定期的なクリーンアップなど、タスクのスケジュールに関連するさまざまなハウスキーピングタスクを担当します。

```
[automationcontroller]
hostA
hostB
hostC
[instance_group_east]
hostB
hostC
[instance_group_west]
hostC
hostD
```

注記

リソースに対してグループが選択されていない場合は **automationcontroller** グループが使用されますが、他のグループが選択されている場合には **automationcontroller** グループは使用されません。

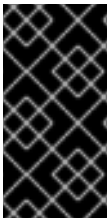
データベースグループは、外部 PostgreSQL を指定します。データベースホストを個別にプロビジョニングする場合、このグループは空である必要があります。

```
[automationcontroller]
hostA
hostB
hostC
[database]
hostDB
```

クラスター内の個々のコントローラーインスタンスで Playbook を実行すると、その Playbook の出力は、Automation Controller の websocket ベースのストリーミング出力機能の一部として、他のすべてのノードにブロードキャストされます。このデータブロードキャストは、インベントリー内の各ノードにプライベートルーティング可能なアドレスを指定し、内部アドレスを使用して処理する必要があります。

```
[automationcontroller]
hostA routable_hostname=10.1.0.2
hostB routable_hostname=10.1.0.3
hostC routable_hostname=10.1.0.4
routable_hostname
```

routable_hostname の詳細は、**Red Hat Ansible Automation Platform インストールガイド**の 一般的な変数 を参照してください。



重要

Automation Controller の以前のバージョンでは、変数名 **rabbitmq_host** を使用していました。以前のバージョンのプラットフォームからアップグレードしようとしており、インベントリーに **rabbitmq_host** を指定している場合は、アップグレードの前に、名前を **rabbitmq_host** から **routable_hostname** に変更してください。

7.2.1. Automation Controller と Automation Hub によって使用されるインスタンスとポート

Automation Controller で使用され、オンプレミスの Automation Hub ノードでも必要なポートとインスタンスは次のとおりです。

- ポート 80、443 (通常の Automation Controller ポートおよび Automation Hub ポート)
- ポート 22 (ssh - Ingress のみが必要)
- ポート 5432 (データベースインスタンス - データベースを外部インスタンスにインストールする場合は、Automation Controller インスタンスに対してこのポートを開く必要があります。)

7.3. ブラウザーの API によるステータスの確認およびモニタリング

Automation Controller は、クラスターの健全性を検証するために、**/api/v2/ping** にあるブラウザー API を使用して可能な限り多くのステータスを報告します。これには、以下のパラメーターが含まれます。

- HTTP 要求にサービスを提供するインスタンス
- クラスター内にある他の全インスタンスが出した最後のハートビートのタイムスタンプ
- これらのグループのインスタンスグループおよびインスタンスのメンバーシップ

実行中のジョブやメンバー情報など、インスタンスおよびインスタンスグループに関する詳細情報は、**/api/v2/instances/** および **/api/v2/instance_groups/** を参照してください。

7.4. インスタンスサービスおよび障害時の動作

各 Automation Controller インスタンスは、以下のさまざまなサービスで構成されており、これらのサービスは連携しています。

HTTP サービス

これには、Automation Controller アプリケーション自体と外部 Web サービスが含まれます。

Callback receiver

実行中の Ansible ジョブからジョブイベントを受信します。

Dispatcher

全ジョブを処理して実行するワーカーキューです。

Redis

このキー値ストアは、ansible-playbook からアプリケーションに伝搬されるイベントデータのキューとして使用されます。

Rsyslog

ログをさまざまな外部ロギングサービスに配信するために使用されるログ処理サービスです。

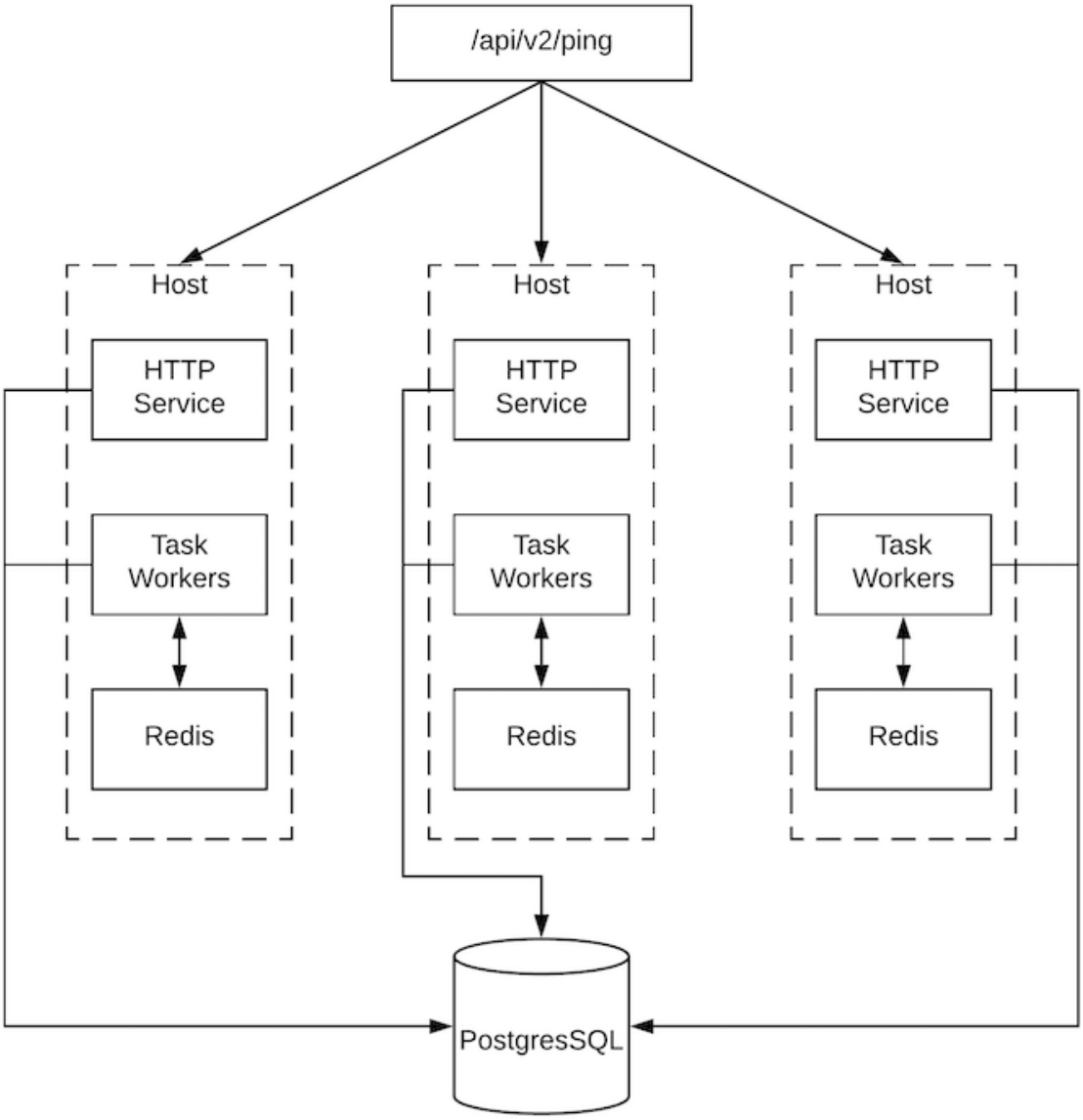
Automation Controller は、これらのサービスまたはそのコンポーネントのいずれかが失敗した場合に、すべてのサービスが再起動されるように設定されています。これらが短期間で頻繁に失敗する場合は、予期せぬ動作を引き起こすことなく修復できるように、インスタンス全体が自動的にオフラインになります。

クラスター環境のバックアップと復元については、クラスター環境のバックアップおよび復元 セクションを参照してください。

7.5. ジョブランタイムの動作

ジョブが実行され、Automation Controller の **通常の** ユーザーにレポートされる方法は変わりません。システム側では、次の違いに注意してください。

- ジョブが API インターフェイスから送信されると、ジョブはディスパッチャーキューにプッシュされます。各 Automation Controller インスタンスは、スケジューリングアルゴリズムを使用してそのキューに接続し、そこからジョブを受け取ります。クラスター内のどのインスタンスも同じ確率でジョブを受信してタスクを実行します。ジョブの実行中にインスタンスが失敗した場合、その作業は永続的に失敗したものとしてマークされます。



- プロジェクトの更新は、ジョブを実行する可能性のあるすべてのインスタンスで正常に実行されます。プロジェクトは、ジョブを実行する直前に、インスタンス上の正しいバージョンに同期されます。必要なリビジョンがすでにローカルでチェックアウトされており、Galaxy または Collections の更新が必要ない場合は、同期を実行できません。

- 同期が行われると、**launch_type = sync** および **job_type = run** のプロジェクト更新としてデータベースに記録されます。プロジェクトの同期によってプロジェクトのステータスやバージョンは変更されません。代わりに、プロジェクトが実行されているインスタンスでソースツリーのみが更新されます。

- Galaxy または Collections からの更新が必要な場合は、同期が実行されて必要なロールをダウンロードします。これにより、**/tmp file** の領域がより多く消費されます。大規模なプロジェクト (約 10 GB) がある場合、**/tmp** のディスク領域が問題になる可能性があります。

7.5.1. ジョブの実行

デフォルトでは、ジョブが Automation Controller キューに送信されると、どのワーカーでもジョブを取得できます。ただし、ジョブを実行するインスタンスを制限するなど、特定のジョブを実行する場所を制御することができます。

インスタンスの一時的なオフライン化をサポートするために、各インスタンスで有効なプロパティーが定義されています。このプロパティーが無効になっている場合、そのインスタンスにはジョブは割り当てられません。既存のジョブは終了しますが、新しいジョブは割り当てられません。

トラブルシューティング

実行中の Automation Controller ジョブに対して **cancel** 要求を発行すると、Automation Controller は ansible-playbook プロセスに **SIGINT** を発行します。これにより、Ansible は新しいタスクのディスパッチを停止して終了しますが、多くの場合、すでにリモートホストにディスパッチされたモジュールタスクは完了するまで実行されます。この動作は、コマンドラインでの Ansible の実行中に **Ctrl-c** を押した場合と似ています。

ソフトウェアの依存関係に関しては、実行中のジョブがキャンセルされた場合、そのジョブは削除されますが、依存関係は残ります。

7.6. インスタンスのプロビジョニング解除

設定用 Playbook を再実行しても、インスタンスのプロビジョニングが自動的に解除されるわけではありません。これは現在、インスタンスが意図的にオフラインにされたのか、障害が原因でオフラインになったのかをクラスターが識別できないためです。代わりに、Automation Controller インスタンスの全サービスをシャットダウンした後に、他のインスタンスからプロビジョニング解除ツールを実行します。

手順

1. コマンド **automation-controller-service stop** を使用して、インスタンスをシャットダウンするか、サービスを停止します。
2. 別のインスタンスから次のプロビジョニング解除コマンドを実行して、Automation Controller クラスターからインスタンスを削除します。

```
$ awx-manage deprovision_instance --hostname=<name used in inventory file>
```

例

```
awx-manage deprovision_instance --hostname=hostB
```

Automation Controller でインスタンスグループのプロビジョニングを解除しても、インスタンスグループは自動的にプロビジョニング解除されたり削除されたりしません。詳細は、インスタンスグループのプロビジョニング解除 セクションを参照してください。

第8章 インスタンスとコンテナーグループ

Automation Controller を使用すると、クラスターのメンバーで直接実行される Ansible Playbook を通じて、または必要なサービスアカウントがプロビジョニングされた OpenShift クラスターの namespace でジョブを実行できます。これをコンテナーグループと呼びます。コンテナーグループ内のジョブは、Playbook ごとに必要な場合にのみ実行できます。詳細は、コンテナーグループ を参照してください。

実行環境については、**Automation Controller ユーザーガイド** の 実行環境 を参照してください。

8.1. インスタンスグループ

インスタンスは 1 つ以上のインスタンスグループにグループ化できます。インスタンスグループは、以下にリストされているリソースの 1 つ以上に割り当てることができます。

- 組織
- インベントリー
- ジョブテンプレート

いずれかのリソースに関連付けられたジョブが実行されると、そのジョブは当該リソースに関連付けられたインスタンスグループに割り当てられます。実行プロセス中、ジョブテンプレートに関連付けられたインスタンスグループは、インベントリーに関連付けられたインスタンスグループよりも先にチェックされます。インベントリーに関連付けられたインスタンスグループは、組織に関連付けられたインスタンスグループよりも先にチェックされます。したがって、3 つのリソースに対するインスタンスグループの割り当てにより、以下の階層が形成されます。

ジョブテンプレート > インベントリー > 組織

インスタンスグループを使用する場合は、次の点を考慮してください。

- これらのグループ内に、他のグループおよびグループインスタンスを定義できます。これらのグループには、**instance_group_** という接頭辞を付ける必要があります。インスタンスは、他の **instance_group_** グループと共に、**automationcontroller** または **execution_nodes** グループに存在する必要があります。クラスター化されたセットアップでは、**automationcontroller** グループに少なくとも 1 つのインスタンスが存在する必要があります。これは、API インスタンスグループで **controlplane** として表示されます。詳細とシナリオ例については、**automationcontroller** のグループポリシー を参照してください。
- **controlplane** インスタンスグループを変更することはできません。変更しようとすると、すべてのユーザーに対してパーミッション拒否エラーが発生します。
  したがって、**controlplane** の Instances タブでは、**Disassociate** オプションを使用できません。
- **default** API インスタンスグループは、ジョブを実行可能なすべてのノードで自動的に作成されます。これは他のインスタンスグループと同様ですが、特定のインスタンスグループが特定のリソースに関連付けられていない場合、ジョブの実行は常にデフォルトのインスタンスグループにフォールバックします。デフォルトのインスタンスグループは常に存在し、削除したり名前を変更したりすることはできません。
- **instance_group_default** という名前のグループは作成しないでください。
- インスタンスにグループ名と同じ名前を指定しないでください。

8.1.1. automationcontroller のグループポリシー

ノードを定義する際には、以下の基準を使用します。

- **automationcontroller** グループ内のノードは、**node_type** ホスト変数を **hybrid** (デフォルト) または **control** と定義できます。
- **execution_nodes** グループ内のノードは、**node_type** ホスト変数を **execution** (デフォルト) または **hop** と定義できます。

インベントリーファイルでカスタムグループを定義するには、**instance_group_*** を使用してグループに名前を付けます。* は API のグループの名前になります。インストールの完了後に、API でカスタムインスタンスグループを作成することもできます。

現在の動作では、**instance_group_*** のメンバーが **automationcontroller** または **execution_nodes** グループのメンバーであることを想定しています。

例

```
[automationcontroller]
126-addr.tatu.home ansible_host=192.168.111.126  node_type=control

[automationcontroller:vars]
peers=execution_nodes

[execution_nodes]

[instance_group_test]
110-addr.tatu.home ansible_host=192.168.111.110 receptor_listener_port=8928
```

インストーラーを実行すると、次のエラーが表示されます。

```
TASK [ansible.automation_platform_installer.check_config_static : Validate mesh topology] ***
fatal: [126-addr.tatu.home -> localhost]: FAILED! => {"msg": "The host '110-addr.tatu.home' is not present in either [automationcontroller] or [execution_nodes]"}
```

これを修正するには、ボックス **110-addr.tatu.home** を **execution_node** グループに移動します。

```
[automationcontroller]
126-addr.tatu.home ansible_host=192.168.111.126  node_type=control

[automationcontroller:vars]
peers=execution_nodes

[execution_nodes]
110-addr.tatu.home ansible_host=192.168.111.110 receptor_listener_port=8928

[instance_group_test]
110-addr.tatu.home
```

その結果、以下のようになります。

```
TASK [ansible.automation_platform_installer.check_config_static : Validate mesh topology] ***
ok: [126-addr.tatu.home -> localhost] => {"changed": false, "mesh": {"110-addr.tatu.home":
```

{"node_type": "execution", "peers": [], "receptor_control_filename": "receptor.sock", "receptor_control_service_name": "control", "receptor_listener": true, "receptor_listener_port": 8928, "receptor_listener_protocol": "tcp", "receptor_log_level": "info"}, "126-addr.tatu.home": {"node_type": "control", "peers": ["110-addr.tatu.home"], "receptor_control_filename": "receptor.sock", "receptor_control_service_name": "control", "receptor_listener": false, "receptor_listener_port": 27199, "receptor_listener_protocol": "tcp", "receptor_log_level": "info"}}}

Automation Controller 4.0 以前のバージョンからアップグレードした後は、レガシーの **instance_group_** メンバーに awx コードがインストールされている可能性があります。これにより、ノードが **automationcontroller** グループに配置されます。

8.1.2. API からのインスタンスグループの設定

インスタンスグループは、システム管理者として **/api/v2/instance_groups** に POST 要求を出すことで作成できます。

インスタンスグループを作成したら、以下を使用してインスタンスをインスタンスグループに関連付けることができます。

```
HTTP POST /api/v2/instance_groups/x/instances/ {'id': y}`
```

インスタンスグループに追加したインスタンスは、グループのワークキューをリッスンするように自動で再設定されます。詳細は、次の **インスタンスグループのポリシー** セクションを参照してください。

8.1.3. インスタンスグループのポリシー

ポリシーを定義することにより、オンラインになると自動的にインスタンスグループに参加するように Automation Controller インスタンスを設定できます。これらのポリシーは、新しいインスタンスがオンラインになるたびに評価されます。

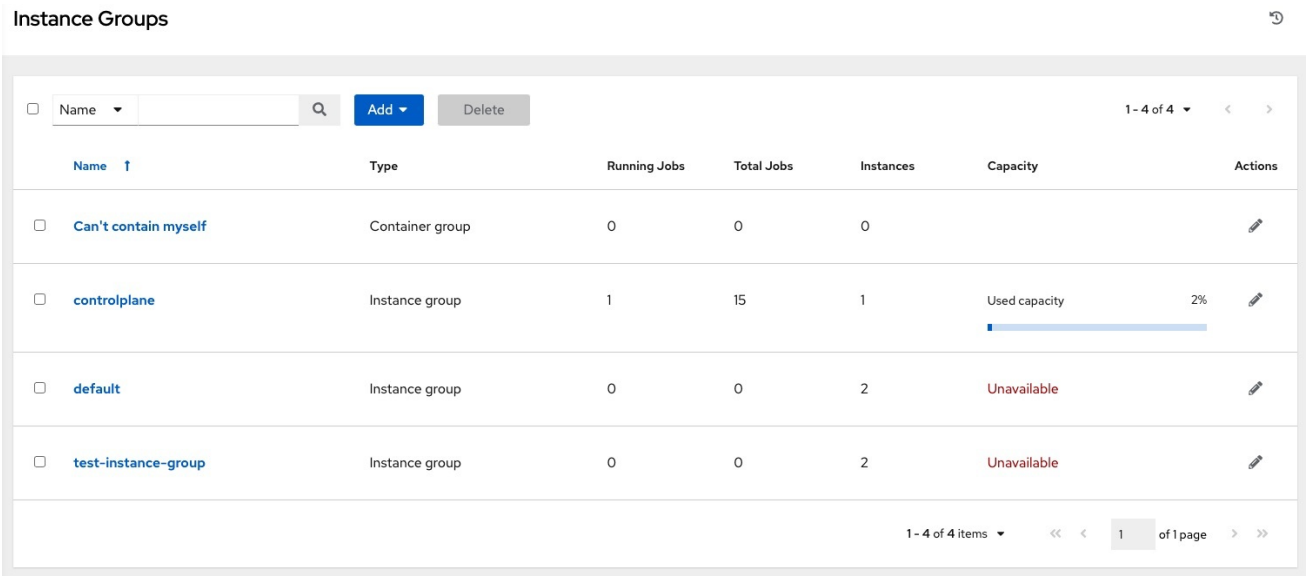
インスタンスグループポリシーは、**Instance Group** の次の 3 つの任意フィールドにより制御されます。

- **policy_instance_percentage**: これは、0 - 100 の間の数字で指定します。これにより、指定した割合のアクティブな Automation Controller インスタンスがこのインスタンスグループに確実に追加されます。新しいインスタンスがオンラインになると、インスタンスの総数に対してこのグループのインスタンス数が指定の割合より少ない場合には、指定の割合の条件を満たすまで、新しいインスタンスが追加されます。

- **policy_instance_minimum**: このポリシーは、インスタンスグループ内に保持するよう試行する最小インスタンス数を指定します。利用可能なインスタンスの数がこの最小数よりも少ない場合、すべてのインスタンスがこのインスタンスグループに配置されます。

- **policy_instance_list**: これは、このインスタンスグループに常に含めるインスタンス名の固定リストです。

Automation Controller ユーザーインターフェイス (UI) の **Instance Groups** リストビューには、インスタンスグループポリシーをもとにした各インスタンスグループのキャパシティーレベルの概要が表示されます。



関連情報

詳細は、**Automation Controller ユーザーガイド** の インスタンスグループの管理 セクションを参照してください。

8.1.4. 主なポリシーの考慮事項

ポリシーに関して以下の点を考慮してください。

- **policy_instance_percentage** と **policy_instance_minimum** は、どちらも最小割り当てレベルを設定します。グループに割り当てるインスタンスの数が多いルールが有効になります。たとえば、**policy_instance_percentage** が 50%、**policy_instance_minimum** が 2 の場合、6 つのインスタンスを起動すると、そのうち 3 つがインスタンスグループに割り当てられます。クラスター内のインスタンス総数を 2 に減らすと、**policy_instance_minimum** を満たすように、2 つのインスタンスがいずれもインスタンスグループに割り当てられます。こうすることで、利用可能なリソースの制限に合わせて、低い値を設定できます。

- ポリシーは、インスタンスが複数のインスタンスグループに割り当てられないように自発的に規制するわけではありませんが、割合が合計で 100 になるように指定すると、複数のインスタンスグループに割り当てないようにできます。4 つのインスタンスグループがある場合、各インスタンスグループに割合の値として 25 を割り当てると、インスタンスはグループ間で重複することなく分散されます。

8.1.5. 特定グループへのインスタンスの手動ピニング

インスタンスが特別で、特定のインスタンスグループだけに割り当てる必要があり、"percentage" または "minimum" のポリシーで他のグループに自動的に参加させない場合には、以下を行います。

手順

1. インスタンスを 1 つ以上のインスタンスグループの **policy_instance_list** に追加します。
2. インスタンスの **manage_by_policy** プロパティーを **False** に更新します。

これにより、インスタンスが percentage と minimum ポリシーに基づいて他のグループに自動的に追加されるのを防ぎます。当該インスタンスは、手動で割り当てたグループにのみ属します。

```
HTTP PATCH /api/v2/instance_groups/N/
```

```
{
"policy_instance_list": ["special-instance"]
}
HTTP PATCH /api/v2/instances/X/
{
"managed_by_policy": False
}
```

8.1.6. ジョブランタイムの動作

インスタンスグループに関連付けられたジョブを実行する場合は、次の動作に注意してください。

- クラスターを個別のインスタンスグループに分割した場合、その動作はクラスター全体の動作と同様になります。2 つのインスタンスを 1 つのグループに割り当てた場合、いずれのインスタンスも同じグループ内の他のインスタンスと同様の確率でジョブを受け取ります。
- Automation Controller インスタンスがオンラインになると、システムの作業容量が実質的に拡張されます。これらのインスタンスをインスタンスグループに配置すると、そのグループの容量も拡張されます。複数のグループのメンバーであるインスタンスが作業を実行している場合、インスタンスがメンバーとなっているすべてのグループから容量が削減されます。インスタンスのプロビジョニングを解除すると、そのインスタンスが割り当てられていたクラスターから容量が削除されます。詳細は、インスタンスグループのプロビジョニング解除 セクションを参照してください。

注記

すべてのインスタンスを同じ容量でプロビジョニングする必要はありません。

8.1.7. ジョブ実行場所の制御

インスタンスグループをジョブテンプレート、インベントリー、または組織に関連付けた場合、そのジョブテンプレートから実行したジョブはデフォルトの動作の対象になりません。つまり、これら 3 つのリソースに関連付けられたインスタンスグループ内のすべてのインスタンスが容量不足の場合、容量が利用可能になるまでジョブが保留状態になります。

ジョブ送信先のインスタンスグループを決定する際の優先順位は、以下のとおりです。

1. ジョブテンプレート
2. Inventory
3. 組織 (プロジェクトの形式)

インスタンスグループをジョブテンプレートに関連付け、すべてのインスタンスグループが容量に達している場合、ジョブはインベントリーで指定したインスタンスグループに送信され、次に組織で指定したインスタンスグループに送信されます。リソースが利用可能な場合、優先順位の高いグループから順にジョブを実行する必要があります。

Playbook で定義されたカスタムインスタンスグループと同様に、グローバルの **default** グループをリソースに関連付けることもできます。これを使用して、ジョブテンプレートまたはインベントリーで優先インスタンスグループを指定できます。一方で、容量が不足している場合には任意のインスタンスにジョブを送信できます。

例

- **group_a** をジョブテンプレートに関連付け、**default** グループをそのインベントリーに関連付けると、**group_a** が容量不足の場合に **default** グループをフォールバックとして使用できるようになります。

- さらに、インスタンスグループを1つのリソースに関連付けずに、別のリソースをフォールバックとして指定することもできます。たとえば、インスタンスグループをジョブテンプレートに関連付けず、インベントリーまたは組織のインスタンスグループにフォールバックさせます。

ここでは、次の 2 つの例を示します。

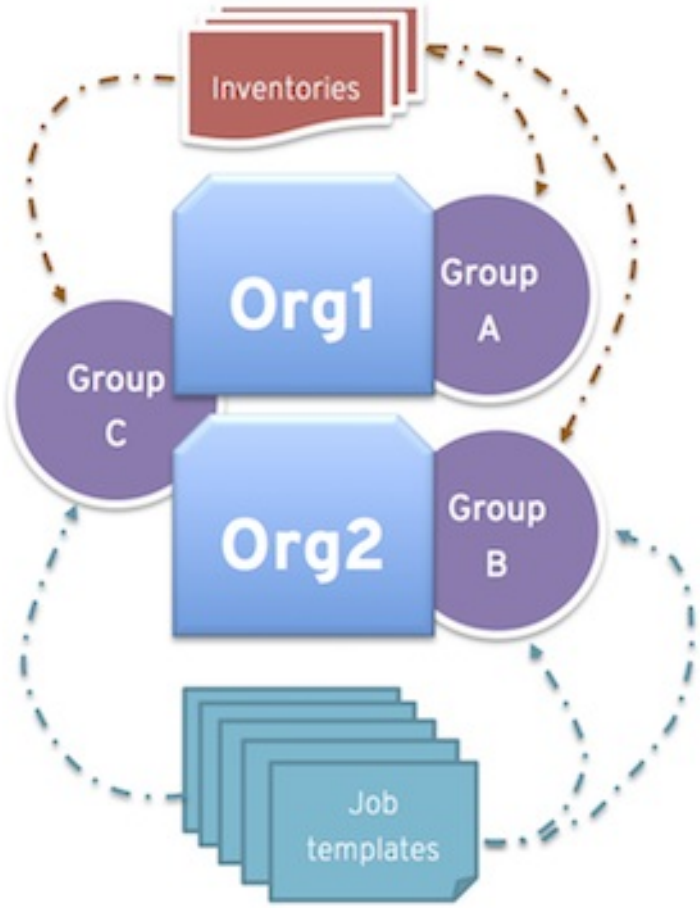
1. インスタンスグループをインベントリーに関連付けた場合 (インスタンスグループへのジョブテンプレートの割り当てを省略)、特定のインベントリーに対して Playbook を実行すると、そのインベントリーに関連付けられたグループでのみ実行されます。これは、それらのインスタンスのみが管理対象ノードへの直接リンクを有する状況で役立ちます。

2. 管理者はインスタンスグループを組織に割り当てることができます。これにより、管理者はインフラストラクチャー全体をセグメント化し、他の組織のジョブ実行能力を妨げることなくジョブを実行できる容量を、各組織が確保できるようになります。
   管理者は、次のシナリオのように、各組織に複数のグループを割り当てることができます。

   - **A**、**B**、および **C** の 3 つのインスタンスグループがあります。**Org1** と **Org2** の 2 つの組織があります。

   - 管理者は、グループ **A** を **Org1** に、グループ **B** を **Org2** に割り当てます。次に、必要になる可能性がある追加容量のオーバーフローとして、グループ **C** を **Org1** と **Org2** の両方に割り当てます。

   - 組織管理者は、必要なグループにインベントリーまたはジョブテンプレートを自由に割り当てたり、組織からデフォルトの順序を継承させたりすることができます。



このようにリソースを配置すると、柔軟性が得られます。また、インスタンスを 1 つだけ含むインスタンスグループを作成して、Automation Controller クラスター内の特定のホストに作業を割り振ることもできます。

8.1.8. インスタンスグループの容量制限

外部ビジネスロジックにより、インスタンスグループに送信したジョブの同時実行性や、消費するフォークの最大数を制限する必要性が高まる場合があります。

従来のインスタンスおよびインスタンスグループの場合、2 つの組織が同じ基盤インスタンス上でジョブを実行できるようにしながら、各組織の同時実行ジョブの総数を制限することが推奨されます。これは、組織ごとにインスタンスグループを作成し、**max_concurrent_jobs** の値を割り当てることで実現できます。

Automation Controller グループの場合、Automation Controller は通常、OpenShift クラスターのリソース制限を認識しません。namespaceの Pod 数に制限を設定することができます。自動スケーリングが設定されていない場合には、一度に特定の数の Pod をスケジュールするのに使用できるリソースのみに設定したりできます。この場合、**max_concurrent_jobs** の値を調整できます。

使用可能なもう 1 つのパラメーターは **max_forks** です。このパラメーターにより、インスタンスグループまたはコンテナーグループで消費される容量の上限をさらに柔軟に設定できます。このパラメーターは、さまざまなインベントリーサイズと "forks" 値を持つジョブが実行されている場合に使用できます。これにより、組織が同時に実行できるジョブの数を最大 10 個に制限し、一度に消費するフォークの数を最大 50 個に制限できます。

```
max_concurrent_jobs: 10
max_forks: 50
```

それぞれ 5 つのフォークを使用する 10 個のジョブを実行している場合、これらのジョブの 1 つがそのグループでの実行を終了する (または、容量のある別のグループでスケジュールされる) まで、11 番目のジョブは待機します。

それぞれ 20 個のフォークを使用する 2 つのジョブを実行している場合、これらのジョブの 1 つがそのグループでの実行を終了する (または、容量のある別のグループでスケジュールされる) まで、**task_impact** が 11 以上の 3 番目のジョブは待機します。

コンテナーグループの場合、ジョブの "forks" 値に関係なく、すべてのジョブが同じリソース要求で同じ **pod_spec** を使用して送信されるため、**max_forks** 値を使用すると便利です。デフォルトの **pod_spec** は、制限ではなく要求を設定します。そのため、Pod はスロットリングが適用されたりリープされたりすることなく、要求した値を超えて "バースト" できます。**max_forks** 値を設定することで、フォーク値の大きい多数のジョブが同時にスケジュールされ、OpenShift ノードが、要求した値よりも多くのリソースを使用して複数の Pod でオーバーサブスクライブされるという状況を防ぐことができます。

インスタンスグループ内の同時実行ジョブとフォークの最大値を設定するには、**Automation Controller ユーザーガイド** の インスタンスグループの作成 を参照してください。

8.1.9. インスタンスグループのプロビジョニング解除

設定用 Playbook を再実行しても、インスタンスのプロビジョニングは解除されません。これは現在、インスタンスが意図的にオフラインにされたのか、障害が原因でオフラインになったのかをクラスターが識別できないためです。代わりに、Automation Controller インスタンスの全サービスをシャットダウンした後に、他のインスタンスからプロビジョニング解除ツールを実行します。

手順

1. 次のコマンドで、インスタンスをシャットダウンするか、サービスを停止します。

```
automation-controller-service stop
```

2. 別のインスタンスから次のプロビジョニング解除コマンドを実行して、コントローラーのクラスターレジストリーからインスタンスを削除します。

```
awx-manage deprovision_instance --hostname=<name used in inventory file>
```

例

```
awx-manage deprovision_instance --hostname=hostB
```

Automation Controller でインスタンスグループのプロビジョニングを解除しても、インスタンスグループは自動的にプロビジョニング解除されたり削除されたりしません (再プロビジョニングによって当該インスタンスグループが使用されなくなることはしばしばあります)。インスタンスグループは、引き続き API エンドポイントや統計監視に表示される可能性があります。次のコマンドを使用すると、これらのグループを削除できます。

```
awx-manage unregister_queue --queuename=<name>
```

インベントリーファイルのインスタンスグループからインスタンスのメンバーシップを削除し、設定用 Playbook を再実行しても、インスタンスがグループに再度追加されなくなるわけではありません。インスタンスがグループに再度追加されないようにするには、API を介してインスタンスを削除し、インベントリーファイルからも削除します。インベントリーファイルでのインスタンスグループの定義をや

めることもできます。Automation Controller UI を通じてインスタンスグループのトポロジーを管理できます。UI でインスタンスグループを管理する方法の詳細は、**Automation Controller ユーザーガイド** の インスタンスグループの管理 を参照してください。

注記

古いバージョンの Automation Controller (3.8.x 以前) で作成した分離インスタンスグループがあり、それらを実行ノードに移行して自動化メッシュアーキテクチャーで使用できるようにする場合は、**Ansible Automation Platform Upgrade and Migration Guide** の Migrate isolated instances to execution nodes を参照してください。

8.2. コンテナーグループ

Ansible Automation Platform はコンテナーグループをサポートします。これにより、Automation Controller がスタンドアロン、仮想環境、コンテナーのいずれとしてインストールされているかに関係なく、Automation Controller でジョブを実行できます。コンテナーグループは、仮想環境内のリソースのプールとして機能します。OpenShift コンテナーを指すインスタンスグループを作成できます。OpenShift コンテナーは、Playbook の実行中にのみ存在する Pod としてオンデマンドでプロビジョニングされるジョブ環境です。これは一時的な実行モデルとして知られており、すべてのジョブの実行に対してクリーンな環境を確保します。

場合によっては、コンテナーグループを "常時オン" に設定することが必要な場合があります。これは、インスタンスの作成を通じて設定できます。

注記

Automation Controller 4.0 より前のバージョンからアップグレードしたコンテナーグループはデフォルトに戻り、古い Pod 定義は削除され、移行中のすべてのカスタム Pod 定義は消去されます。

実行環境はコンテナーイメージであり、仮想環境を使用しないという点で、実行環境とコンテナーグループは異なります。詳細は、**Automation Controller ユーザーガイド** の 実行環境 を参照してください。

8.2.1. コンテナーグループの作成

ContainerGroup は、OpenShift クラスターへの接続を可能にする認証情報が関連づけられた **InstanceGroup** の一種です。

前提条件

- 起動できる namespace がある。すべてのクラスターには "デフォルト" の namespace がありますが、特定の namespace を使用することもできます。
- この namespace で Pod を起動および管理可能なロールを持つサービスアカウントがある。
- プライベートレジストリーで実行環境を使用しており、Automation Controller でコンテナーレジストリー認証情報が実行環境に関連付けられている場合、サービスアカウントには、namespace でシークレットを取得、作成、削除するためのロールも必要です。これらのロールをサービスアカウントに付与しない場合は、**ImagePullSecrets** を事前に作成し、**ContainerGroup** の Pod 仕様で指定できます。この場合、実行環境にコンテナーレジストリーの認証情報を関連付けることはできません。関連付けられている場合、Automation Controller は namespace にシークレットを作成しようとします。

- サービスアカウントに関連付けられたトークンがある。OpenShift または Kubernetes Bearer トークン。
- クラスターに関連付けられた CA 証明書がある。

次の手順では、Automation Controller を使用してコンテナーグループでジョブを実行するためのサービスアカウントを OpenShift クラスターまたは Kubernetes に作成する方法を説明します。サービスアカウントを作成すると、その認証情報が OpenShift または Kubernetes API Bearer トークン認証情報の形式で Automation Controller に提供されます。

手順

1. サービスアカウントを作成するには、サンプルサービスアカウントである **containergroup sa** をダウンロードして使用し、認証情報を取得するために必要に応じて変更します。
2. **containergroup-sa.yml** から設定を適用します。

```
oc apply -f containergroup-sa.yml
```

3. サービスアカウントに関連付けられているシークレット名を取得します。

```
export SA_SECRET=$(oc get sa containergroup-service-account -o json | jq
'.secrets[0].name' | tr -d "")
```

4. シークレットからトークンを取得します。

```
oc get secret $(echo ${SA_SECRET}) -o json | jq '.data.token' | xargs | base64 --decode >
containergroup-sa.token
```

5. CA 証明書を取得します。

```
oc get secret $SA_SECRET -o json | jq '.data["ca.crt"]' | xargs | base64 --decode >
containergroup-ca.crt
```

6. **containergroup-sa.token** および **containergroup-ca.crt** の内容を使用して、コンテナーグループに必要な OpenShift または Kubernetes API Bearer トークン の情報を提供します。

コンテナーグループを作成するには、以下を実行します。

手順

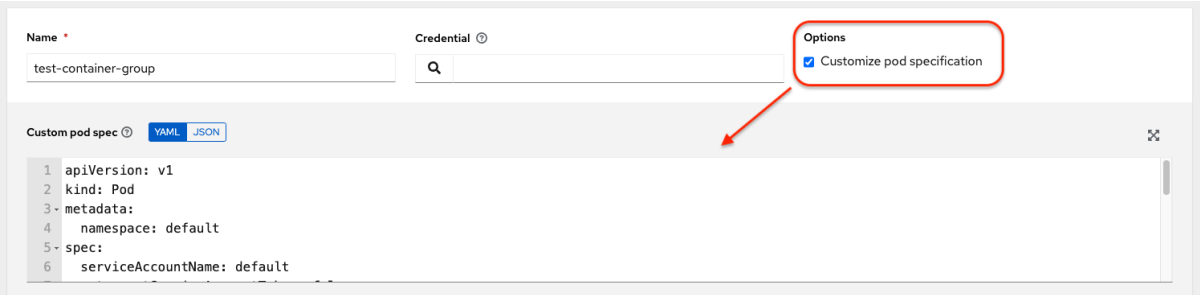
1. Automation Controller UI を使用して、コンテナーグループで使用する OpenShift または Kubernetes API Bearer トークン 認証情報を作成します。詳細は、**Automation Controller ユーザーガイド** の 認証情報の作成 を参照してください。
2. ナビゲーションパネルから **Administration → Instance Groups** を選択します。
3. **Add** をクリックし、**Create Container Group** を選択します。
4. 新しいコンテナーグループの名前を入力し、以前に作成した認証情報を選択して、コンテナーグループに割り当てます。

8.2.2. Pod 仕様のカスタマイズ

Ansible Automation Platform は単純なデフォルトの Pod 仕様を提供しますが、デフォルトの Pod 仕様をオーバーライドするカスタム YAML または JSON ドキュメントを提供できます。このフィールドは、有効な Pod JSON または YAML として "シリアル化" できる **ImagePullSecrets** などのカスタムフィールドを使用します。オプションの完全なリストは、OpenShift ドキュメントの Pod およびサービス セクションにあります。

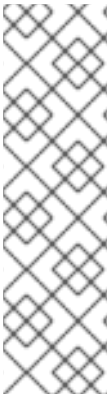
手順

1. Pod 仕様をカスタマイズするには、トグルを使用して **Pod Spec Override** フィールドで namespace を指定し、**Pod Spec Override** フィールドを有効にして展開します。



2. **Save** をクリックします。

必要に応じて、追加のカスタマイズを指定できます。**Expand** をクリックして、カスタマイズウィンドウ全体を表示します。

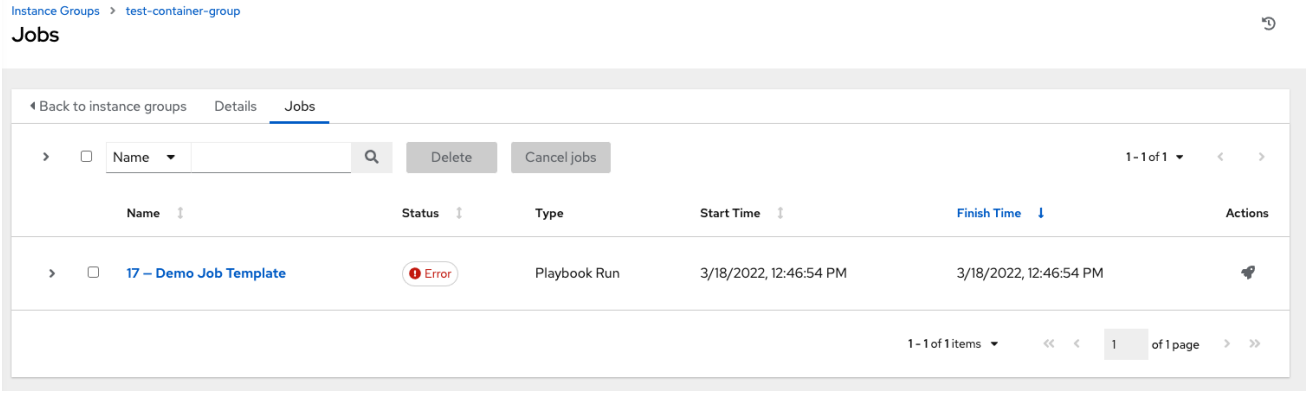


注記

ジョブ起動時のイメージは、どの実行環境がジョブに関連付けられているかによって決まります。コンテナーレジストリーの認証情報を実行環境に関連付ける場合、Automation Controller はイメージをプルするために **ImagePullSecret** を作成しようとします。シークレットを管理するパーミッションをサービスアカウントに付与しない場合は、**ImagePullSecret** を事前に作成してそれを Pod 仕様で指定し、使用する実行環境から認証情報を削除する必要があります。

詳細は、**Red Hat Container Registry Authentication**の Allowing Pods to Reference Images from Other Secured Registries セクションを参照してください。

コンテナーグループが正常に作成されると、新しく作成されたコンテナーグループの **Details** タブが表示され、そこでコンテナーグループ情報を確認および編集できます。これは、**Instance Group** リンクから をクリックした場合に開くメニューと同じです。**Instances** を編集し、このインスタンスグループに関連付けられた **Jobs** を確認することもできます。



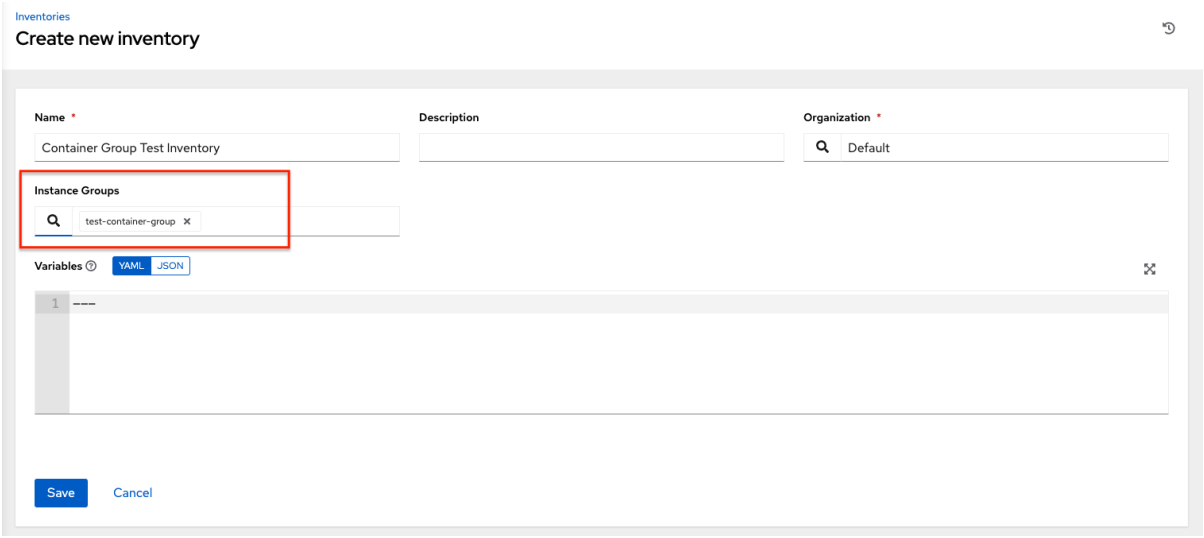
コンテナーグループとインスタンスグループは適宜ラベル付けされます。

8.2.3. コンテナーグループ機能の検証

コンテナーのデプロイと終了を確認するには、以下を実行します。

手順

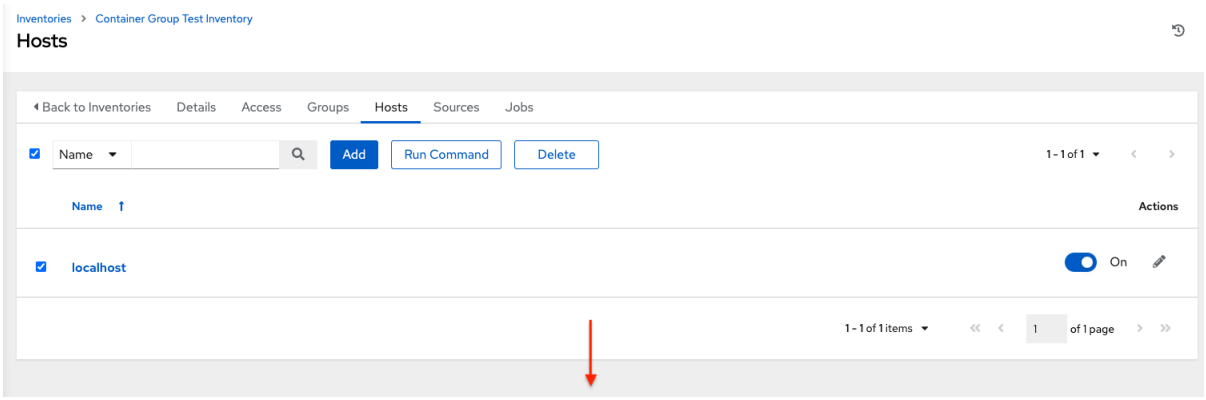
1. 模擬インベントリーを作成し、**Instance Group** フィールドにコンテナーグループの名前を入力して、コンテナーグループをそのインベントリーに関連付けます。詳細は、**Automation Controller ユーザーガイド** の 新規インベントリーの追加 を参照してください。

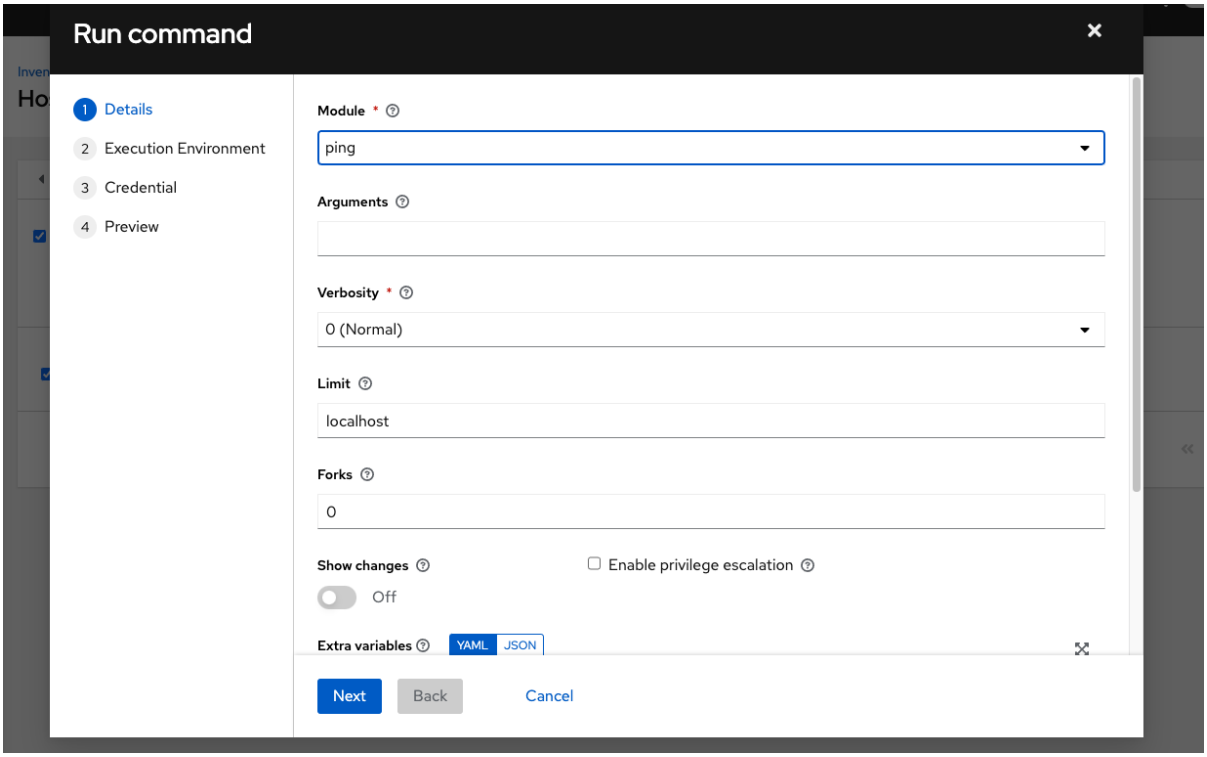


2. 以下の変数を使用して、インベントリーに **localhost** ホストを作成します。

```
{'ansible_host': '127.0.0.1', 'ansible_connection': 'local'}
```

3. **ping** または **setup** モジュールを使用して、ローカルホストに対してアドホックジョブを起動します。**Machine Credential** フィールドは必須ですが、このテストでどれを選択するかは重要ではありません。



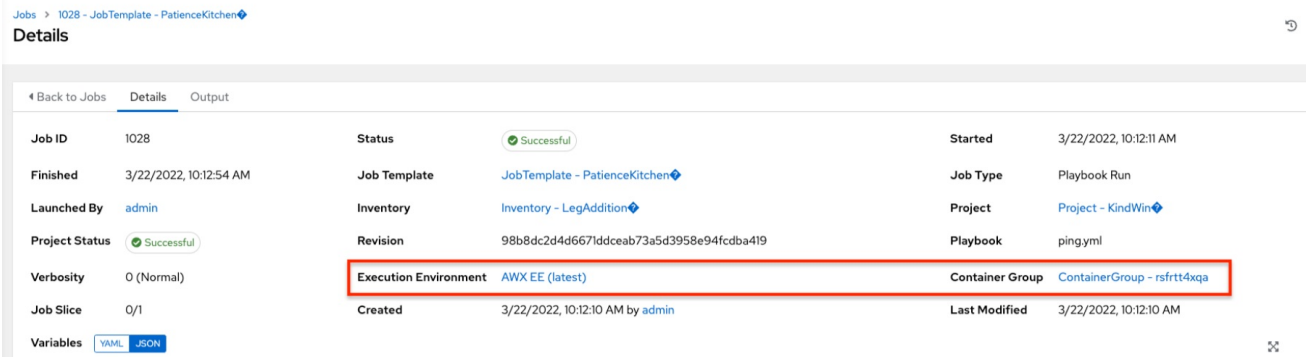


Jobs 詳細ビューで、アドホックジョブの 1 つを使用してコンテナーに正常にアクセスしたことが表示されます。

OpenShift UI を使用している場合、Pod はデプロイされると表示され、終了すると非表示になります。あるいは、CLI を使用して namespace で **get pod** 操作を実行し、同イベントの発生をリアルタイムで監視することもできます。

8.2.4. コンテナーグループジョブの表示

コンテナーグループに関連付けられたジョブを実行すると、そのジョブの詳細を、関連付けられたコンテナーグループおよび起動した実行環境とともに **Details** ビューで確認できます。



8.2.5. Kubernetes API の障害状態

コンテナーグループを実行し、Kubernetes API がリソースクォータを超過したと応答すると、Automation Controller はジョブを保留状態にします。その他の障害の場合は、次の例のように、**Error Details** フィールドのトレースバックに障害の理由が表示されます。

> Error creating pod: pods is forbidden: User "system: serviceaccount: aap:example" cannot create resource "pods" in API group "" in the namespace "aap"

8.2.6. コンテナーの容量制限

コンテナーの容量制限とクォータは、Kubernetes API のオブジェクトで定義します。

- 特定の namespace 内のすべての Pod に制限を設定するには、**LimitRange** オブジェクトを使用します。詳細は、OpenShift ドキュメントの クォータおよび制限範囲 セクションを参照してください。
- Automation Controller が起動する Pod 定義に制限を直接設定するには、Pod 仕様のカスタマイズ と OpenShift ドキュメントの コンピュートリソース セクションを参照してください。

注記

コンテナーグループは、通常のノードが使用する容量アルゴリズムを使用しません。ジョブテンプレートレベルでフォークの数を設定する必要があります。Automation Controller でフォークを設定すると、その設定はコンテナーに渡されます。

第9章 インスタンスによる容量の管理

自動化メッシュのスケーリングは、Red Hat Ansible Automation Platform の OpenShift デプロイメントで利用でき、インストールスクリプトを実行せずに、Automation Controller UI の **インスタンス** リソースを使用して、クラスターにノードを動的に追加または削除することで可能です。

インスタンスは、メッシュトポロジー内のノードとして機能します。自動化メッシュを使用すると、自動化のフットプリントを拡張できます。ジョブを起動する場所は、ansible-playbook が実行される場所とは異なる場合があります。

Automation Controller UI からインスタンスを管理するには、システム管理者またはシステム監査者の権限が必要です。

一般に、ノードのプロセッサーコア (CPU) とメモリー (RAM) が多いほど、そのノードで同時に実行するようにスケジュールできるジョブの数も多くなります。

詳細は、Automation Controller の容量決定とジョブへの影響 を参照してください。

9.1. 前提条件

自動化メッシュは、Red Hat Enterprise Linux (RHEL) 上で実行されるホップおよび実行ノードに依存します。Red Hat Ansible Automation Platform サブスクリプションにより、Ansible Automation Platform のコンポーネントの実行に使用できる 10 個の Red Hat Enterprise Linux ライセンスが付与されます。

Red Hat Enterprise Linux サブスクリプションの詳細は、Red Hat Enterprise Linux ドキュメントの システムの登録とサブスクリプションの管理 を参照してください。

次の手順では、自動化メッシュのデプロイメント用の RHEL インスタンスを準備します。

1. Red Hat Enterprise Linux オペレーティングシステムが必要です。メッシュ内の各ノードには、Automation Controller がアクセスできる静的 IP アドレス、または解決可能な DNS ホスト名が必要です。

2. 続行する前に、RHEL 仮想マシンの最小要件を満たしていることを確認している。詳細は、Red Hat Ansible Automation Platform のシステム要件 を参照してください。

3. 通信が必要なリモートネットワーク内に RHEL インスタンスをデプロイします。仮想マシンの作成については、**Red Hat Enterprise Linux** ドキュメントの 仮想マシンの作成 を参照してください。提案されたタスクを仮想マシン上で実行できるように、必ず仮想マシンの容量を十分にスケーリングしてください。

    - RHEL ISO は、access.redhat.com から入手できます。
    - RHEL クラウドイメージは、console.redhat.com の Image Builder を使用してビルドできます。

9.2. シークレットのプル

デフォルトの実行環境 (Automation Controller で提供) を使用してリモート実行ノードで実行している場合は、実行環境イメージをプルするための認証情報を含むプルシークレットを Automation Controller に追加する必要があります。

これを行うには、Automation Controller の namespace でプルシークレットを作成し、次のように Operator で **ee_pull_credentials_secret** パラメーターを設定します。

手順

1. シークレットを作成します。

```
oc create secret generic ee-pull-secret \
    --from-literal=username=<username> \
    --from-literal=password=<password> \
    --from-literal=url=registry.redhat.io


oc edit automationcontrollers <instance name>
```

2. **ee_pull_credentials_secret ee-pull-secret** を仕様に追加します。

```
spec.ee_pull_credentials_secret=ee-pull-secret
```

Automation Controller UI からインスタンスを管理するには、システム管理者またはシステム監査者の権限が必要です。

9.3. 自動化メッシュで使用するための仮想マシンのセットアップ

手順

1. 各 RHEL インスタンスに SSH で接続し、次の手順を実行します。ネットワークアクセスと制御によっては、SSH プロキシーまたは他のアクセスモデルが必要になる場合があります。
以下のコマンドを使用します。

```
ssh [username]@[host_ip_address]
```

たとえば、Amazon Web Services で実行されている Ansible Automation Platform インスタンスの場合は、以下のようになります。

```
ssh ec2-user@10.0.0.6
```

2. 後のステップで、ホップノードから実行ノードに接続するために使用できる SSH キーを作成またはコピーします。これは、自動化メッシュ設定のためだけに使用される一時キー、または長期間有効なキーの場合があります。SSH ユーザーとキーは、後の手順で使用されます。

3. **baseos** および **appstream** リポジトリーを使用して、RHEL サブスクリプションを有効にします。Ansible Automation Platform RPM リポジトリーは、**Red Hat Update Infrastructure** (RHUI) はなく、subscription-manager を通じてのみ利用できます。RHEL と RHUI を含む他の Linux フットプリントを使用しようとすると、エラーが発生します。

```
sudo subscription-manager register --auto-attach
```

アカウントで Simple Content Access が有効になっている場合は、次を使用します。

```
sudo subscription-manager register
```

Simple Content Access の詳細は、Simple Content Access の使用 を参照してください。

4. Ansible Automation Platform サブスクリプションと適切な Red Hat Ansible Automation Platform チャネルを有効にします。

```
# subscription-manager repos --enable ansible-automation-platform-2.4-for-rhel-8-x86_64-rpms for RHEL 8

# subscription-manager repos --enable ansible-automation-platform-2.4-for-rhel-9-x86_64-rpms for RHEL 9
```

5. パッケージが最新であることを確認します。

```
sudo dnf upgrade -y
```

6. ansible-core パッケージをインストールします。

```
sudo dnf install -y ansible-core
```

9.4. インスタンスの管理

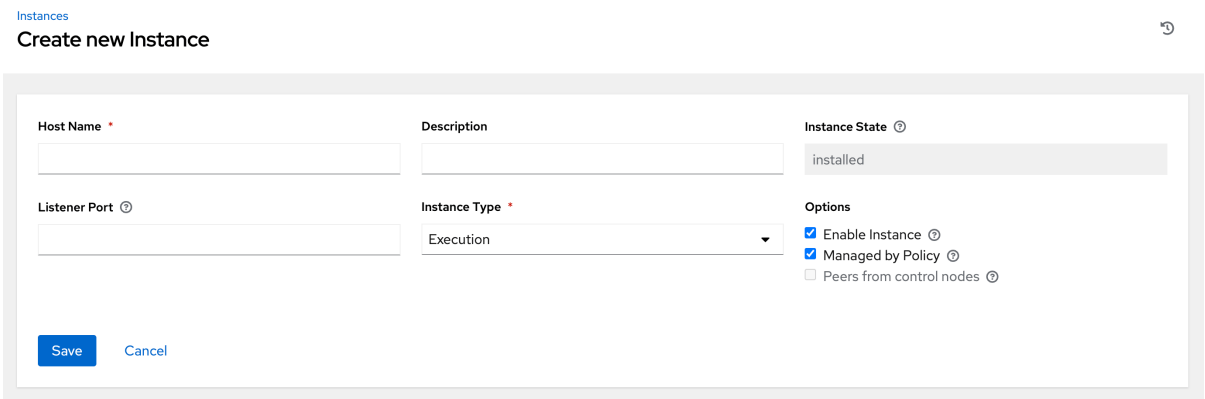
ジョブの容量を拡張するには、Automation Controller のデプロイメントと一緒に実行するように追加できるスタンドアロンの **実行ノード** を作成します。これらの実行ノードは、Automation ControllerKubernetes クラスターの一部ではありません。クラスター内で実行される制御ノードは、Receptor を介して実行ノードに接続し、作業を送信します。これらの実行ノードは、タイプ **execution** インスタンスとして、Automation Controller に登録されます。つまり、コントロールノードのように作業をディスパッチしたり、Web リクエストを処理したりするのではなく、ジョブを実行するためにのみ使用されます。

ホップノード は、Automation Controller のコントロールプレーンとスタンドアロン実行ノードの間に追加できます。これらのホップノードは Kubernetes クラスターの一部ではなく、タイプ **hop** のインスタンスとして Automation Controller に登録されます。つまり、別のネットワークまたはより厳密なネットワーク内で到達不可能なノードへのインバウンドおよびアウトバウンドのトラフィックのみを処理します。

次の手順は、ホストのノードタイプを設定する方法を示しています。

手順

1. ナビゲーションパネルから、**Administration → Instances** を選択します。
2. **Instances** リストページで、**Add** をクリックします。**Create new Instance** ウィンドウが開きます。



インスタンスには次の属性が必要です。

- **Host Name**: (必須) インスタンスの完全修飾ドメイン名 (パブリック DNS)) または IP アドレスを入力します。このフィールドは、インストーラーベースのデプロイメントの **hostname** に相当します。

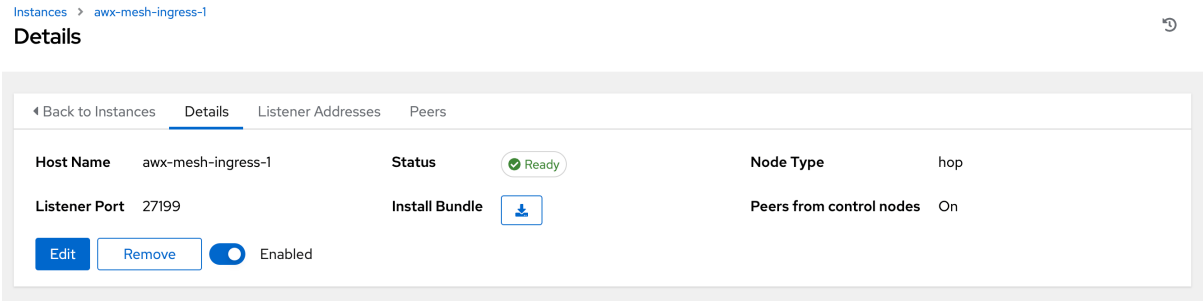
注記

インスタンスがコントロールクラスターから解決できないプライベート DNS を使用している場合、DNS ルックアップルーティングは失敗し、生成された SSL 証明書は無効になります。代わりに IP アドレスを使用してください。

- オプション: **Description**: インスタンスの説明を入力します。
- **Instance State**: このフィールドは自動入力され、インストール中であることを示します。変更はできません。
- **リスナーポート**: このポートは、受信接続をリッスンするために receptor に使用されます。ポートを設定に適したポートに設定できます。このフィールドは、API の **listen_port** に相当します。デフォルト値は 27199 ですが、独自のポート値を設定できます。
- **インスタンスタイプ**: **execution** ノードおよび **hop** ノードのみを作成できます。Operator ベースのデプロイメントは、コントロールノードまたはハイブリッドノードをサポートしません。
  オプション:
  - **インスタンスの有効化**: 実行ノード上でジョブを実行できるようにするには、このボックスをオンにします。
  - **Managed by Policy** ボックスをオンにして、ポリシーがインスタンスの割り当て方法を決定できるようにします。
  - **Peers from control nodes** ボックスをオンにして、コントロールノードがこのインスタンスに自動的にピアリングできるようにします。Automation Controller に接続されているノードの場合は、**Peers from Control** nodes ボックスをチェックして、そのノードと Automation Controller の間に直接通信リンクを作成します。その他のすべてのノードの場合:
    - ホップノードを追加しない場合は、**Peers from Control** がオンになっていることを確認してください。
    - ホップノードを追加する場合は、**Peers from Control** がチェックされていないことを確認してください。
    - ホップノードと通信する実行ノードの場合は、このボックスをオンにしないでください。
  - 実行ノードをホップノードとピアリングするには、 icon next to the **Peers** フィールドをクリックします。
    Select Peers ウィンドウが表示されます。

    実行ノードをホップノードにピアリングします。

3. **Save** をクリックします。

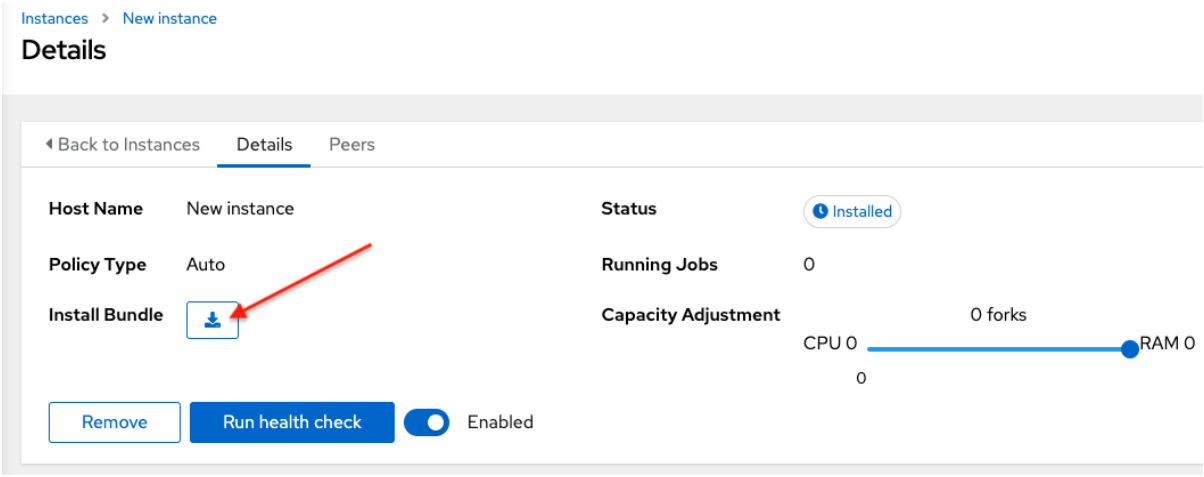


4. 更新したトポロジーをグラフィック表示するには、トポロジービューアー を参照してください。

注記

新しく作成したインスタンスに SSH アクセスできる任意のコンピューターから、次の手順を実行します。

5. **Install Bundle** の横にある  アイコンをクリックして、この新しいインスタンスと、作成されたノードを自動化メッシュにインストールするために必要なファイルを含む tar ファイルをダウンロードします。



インストールバンドルには、TLS 証明書とキー、認証局、および適切な Receptor 設定ファイルが含まれています。

```
receptor-ca.crt
work-public-key.pem
receptor.key
install_receptor.yml
inventory.yml
group_vars/all.yml
requirements.yml
```

6. ダウンロードした **tar.gz** インストールバンドルを、ダウンロードした場所から展開します。これらのファイルがリモートマシン上の正しい場所にあることを確認するために、インストールバンドルには **install_receptor.yml** Playbook が含まれています。Playbook には Receptor コレクションが必要です。次のコマンドを実行して、コレクションをダウンロードします。

```
ansible-galaxy collection install -r requirements.yml
```

7. **ansible-playbook** コマンドを実行する前に、**inventory.yml** ファイル内の次のフィールドを編集します。

```
all:
  hosts:
    remote-execution:
      ansible_host: 10.0.0.6
      ansible_user: <username> # user provided
  ansible_ssh_private_key_file: ~/.ssh/<id_rsa>
```

   - **ansible_host** が、ノードの IP アドレスまたは DNS に設定されていることを確認します。
   - **ansible_user** を、インストールを実行しているユーザー名に設定します。
   - **ansible_ssh_private_key_file** を設定して、インスタンスへの接続に使用される秘密鍵のファイル名を含めます。
   - **inventory.yml** ファイルの内容はテンプレートとして機能し、メッシュトポロジーでの receptor ノードのインストールおよび設定中に適用されるロールの変数が含まれています。他のフィールドの一部を変更したり、高度なシナリオではファイル全体を置き換えたりすることができます。詳細は、Role Variables を参照してください。

8. プライベート DNS を使用するノードの場合は、次の行を **inventory.yml** に追加します。

```
ansible_ssh_common_args: <your ssh ProxyCommand setting>
```

   これは、**install-ceptor.yml** Playbook に、proxy コマンドを使用してローカル DNS ノード経由でプライベートノードに接続するように指示します。

9. 属性を設定したら、**Save** をクリックします。作成したインスタンスの **Details** ページが開きます。

10. ファイルを保存して続行します。

11. インストールバンドルを実行してリモートノードをセットアップし、**ansible-playbook** を実行するシステムには、**ansible.receptor** コレクションがインストールされている必要があります。

```
ansible-galaxy collection install ansible.receptor
```

   または、以下を実行します。

```
ansible-galaxy install -r requirements.yml
```

   - **requirements.yml** ファイルから receptor コレクションの依存関係をインストールすると、そこで指定された receptor のバージョンが一貫して取得されます。さらに、今後必要になる可能性のある他のコレクションの依存関係も取得されます。
   - Playbook が実行されるすべてのノードに receptor コレクションをインストールします。インストールしない場合は、エラーが発生します。

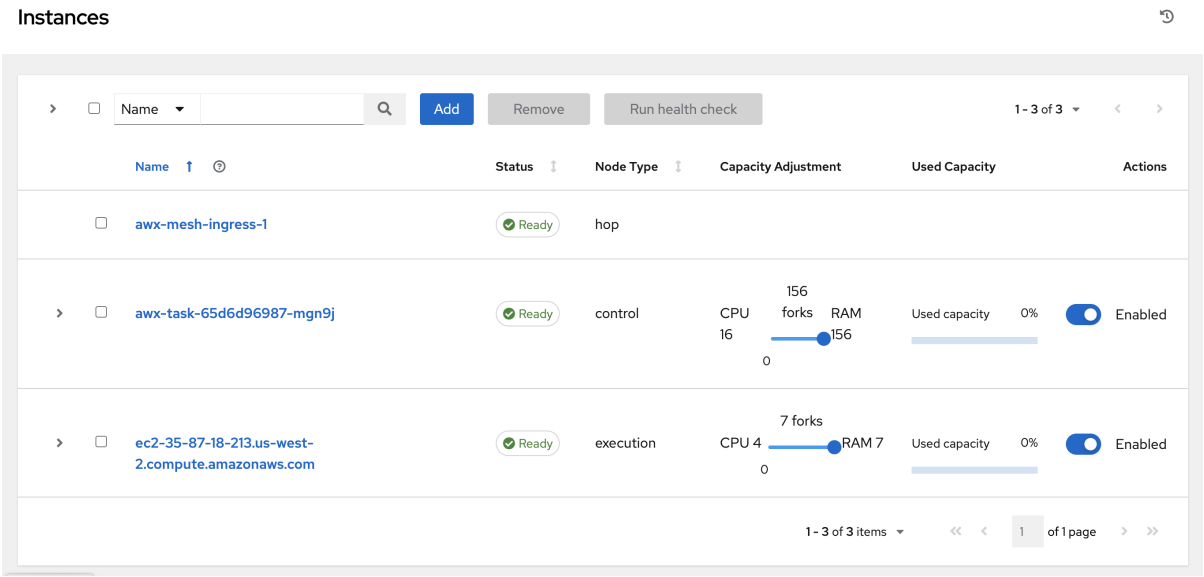
12. **receptor_listener_port** が定義されている場合、マシンには、受信 TCP 接続を確立するために使用可能なオープンポート (27199 など) も必要です。次のコマンドを実行して、receptor 通信用にポート 27199 を開きます。

```
sudo firewall-cmd --permanent --zone=public --add-port=27199/tcp
```

13. 自動化メッシュを更新するマシン上で、次の Playbook を実行します。

```
ansible-playbook -i inventory.yml install_receptor.yml
```

この Playbook を実行すると、自動化メッシュが設定されます。



メッシュからインスタンスを削除するには、インスタンスの削除 を参照してください。

第10章 トポロジービューアー

すでにメッシュトポロジーがデプロイされている場合、トポロジービューアーを使用すると、ノードタイプ、ノードの健全性、および各ノードに関する具体的な情報を表示できます。

Automation Controller UI からトポロジービューアーにアクセスするには、**システム管理者** または **システム監査者** のパーミッションが必要です。

仮想マシンベースのインストールにおける自動化メッシュの詳細は、仮想マシンベースのインストール向け Red Hat Ansible Automation Platform 自動化メッシュガイド を参照してください。

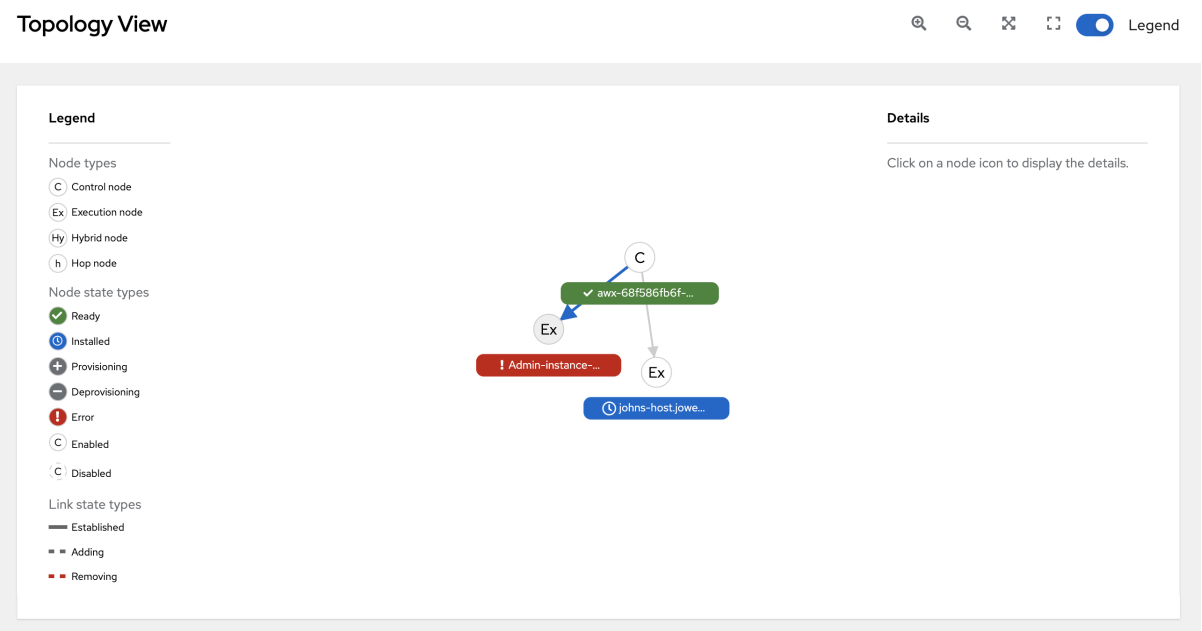
Operator ベースのインストールにおける自動化メッシュの詳細は、Operator ベースのインストール向け Red Hat Ansible Automation Platform 自動化メッシュ を参照してください。

10.1. トポロジービューアーへのアクセス

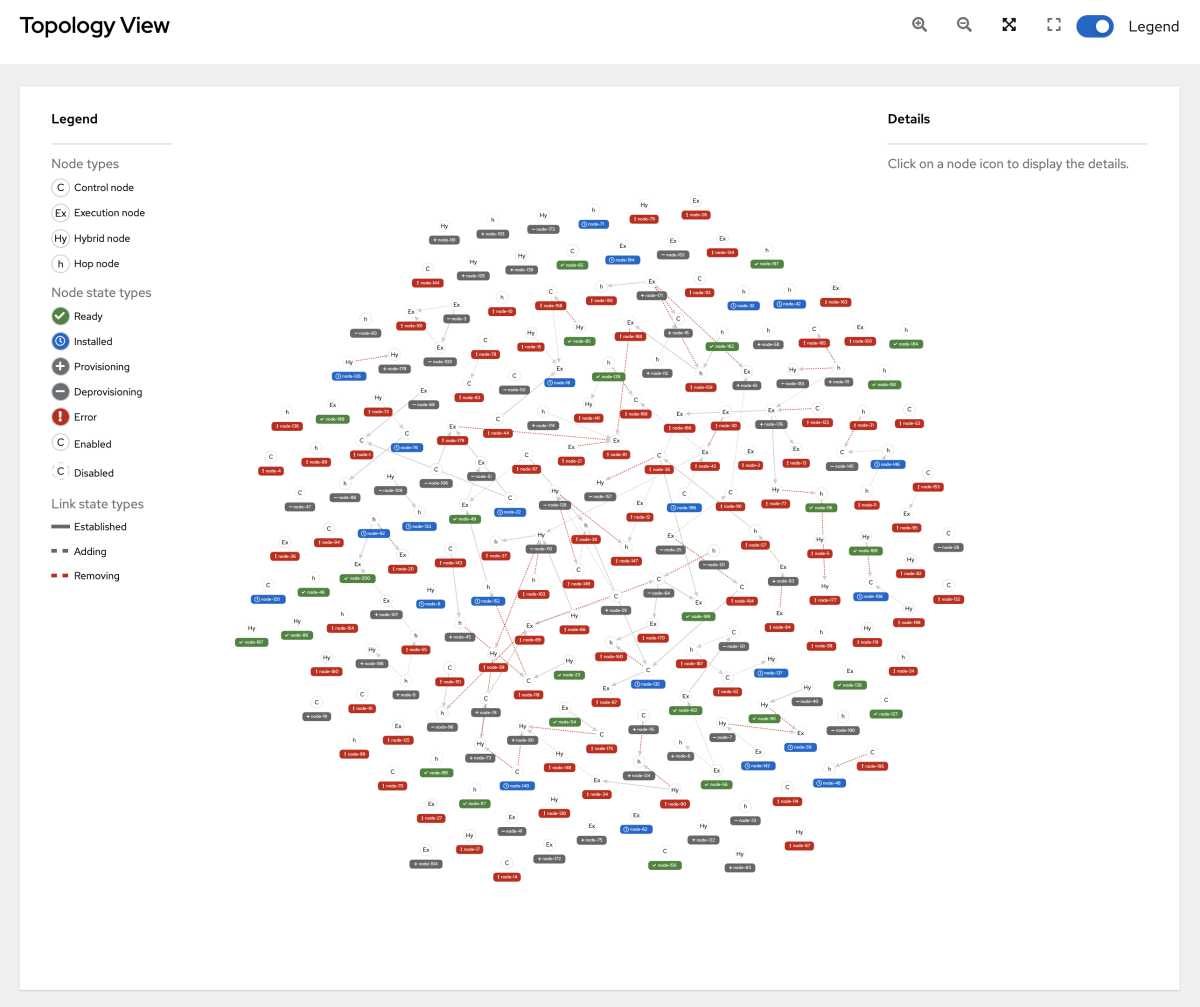
Automation Controller UI からトポロジービューアーにアクセスするには、次の手順を使用します。

手順

1. ナビゲーションパネルから、**Administration → Topology View** を選択します。**Topology View** が開き、各 receptor ノード間のリンクがグラフィック表示されます。



2. ズームレベルを調整したり、グラフィックビューを操作したりするには、ツールバーにあるズームイン ( )、ズームアウト ( )、拡大 ( )、リセット ( ) のコントロールアイコンを使用します。
クリックおよびドラッグして移動したり、マウスやトラックパッドでスクロールすることでズームしたりすることもできます。画面に合わせる機能は、画面に合わせて自動的にグラフィックを拡大縮小し、中央に再配置します。これは、大きなメッシュ全体を表示する場合に特に便利です。

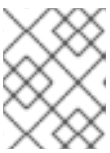


デフォルトのビューにリセットするには、**ズームリセット** ( ) アイコンをクリックします。

3. 表示されているノードのタイプを特定するには、**Legend** を参照します。
仮想マシンベースのインストールについては、コントロールプレーンおよび実行プレーン を参照してください。

   Operator ベースのインストールについては、コントロールプレーンおよび実行プレーン を参照してください。

   上記のリンクから各ノードタイプの詳細を参照してください。



注記

Legend が表示されていない場合は、上部のバーのトグルを使用して有効にします。

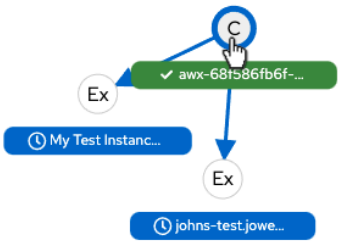
Legend には、**node status <node_statuses>** が色別に表示され、ノードの健全性を示します。Legend の **Error** ステータスには、**Unavailable** 状態 (Instances リストビューに表示されるもの) に加えて、Automation Controller の新しいバージョンで生じる今後のエラー状態が含まれます。

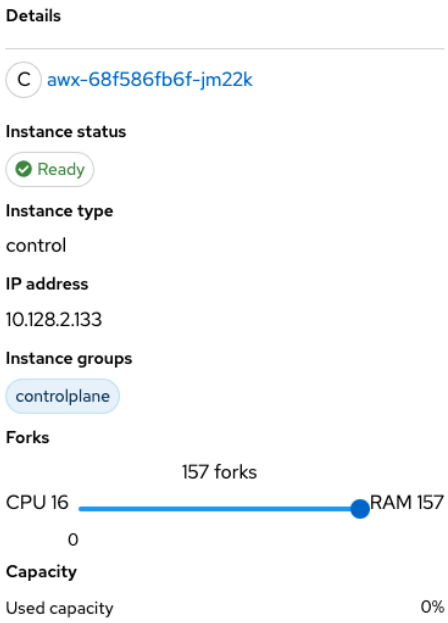
Legend には、次のリンクステータスも表示されます。

- **Established**: このリンク状態は、準備完了、使用不可、または無効であるノード間のピア接続を示します。

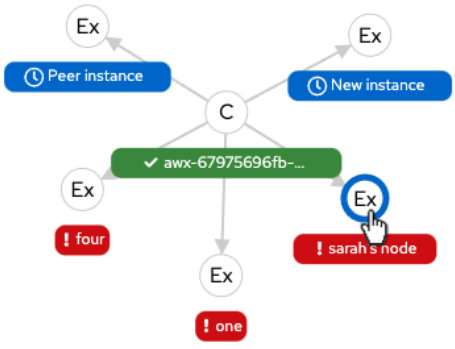
- **Adding**: このリンク状態は、メッシュトポロジーに追加するように選択されたノード間のピア接続を示します。
- **Removing**: このリンク状態は、トポロジーから削除するように選択されたノード間のピア接続を示します。

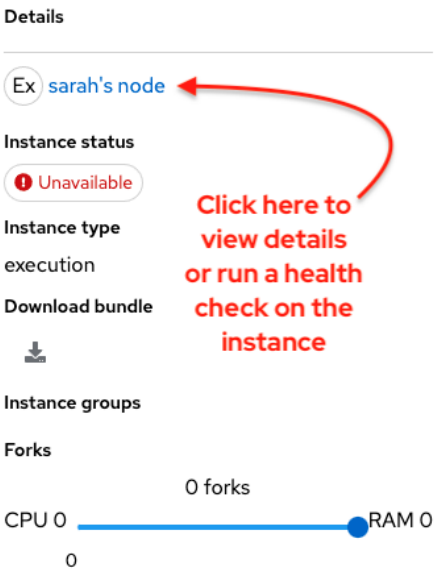
4. ノードにカーソルを合わせると、コネクターが強調表示されて直接接続されたノード (ピア) を示します。または、ノードをクリックして、ホスト名、ノードタイプ、ステータスなどホストの詳細を取得します。

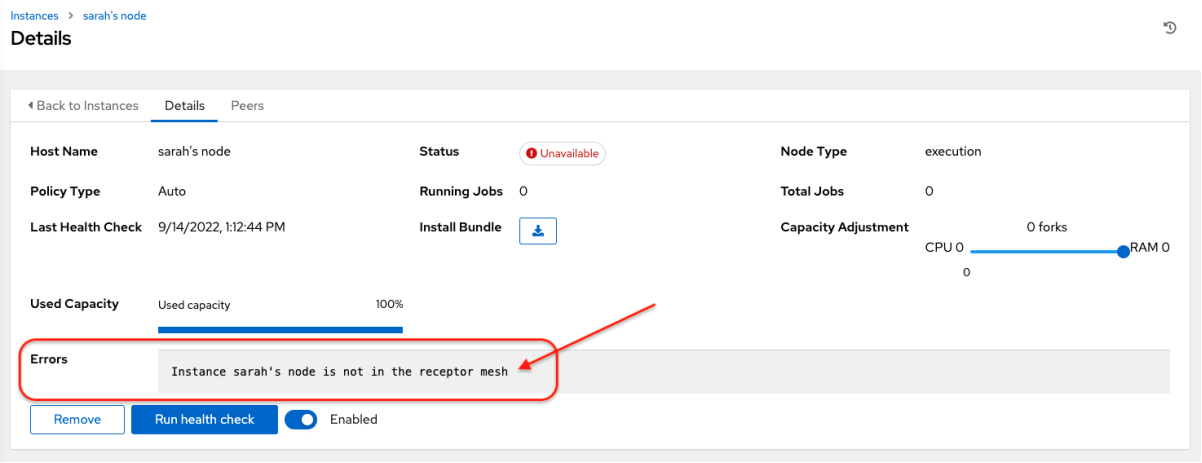




5. 表示された詳細からインスタンスのホスト名のリンクをクリックすると、**Details** ページにリダイレクトされ、そのノードに関する詳細情報、特に **Error** ステータスに関する情報が表示されます。次の例を参照してください。







Details ページを使用して、インスタンスを削除したり、必要に応じてインスタンスのヘルスチェックを実行したり、インスタンスからジョブの割り当てを解除したりできます。デフォルトでは、ジョブを各ノードに割り当てることができます。ただし、これを無効にして、ノードでジョブが実行されないようにすることもできます。

新しいノードの作成とメッシュのスケーリングの詳細は、インスタンスによる容量の管理 を参照してください。

第11章 AUTOMATION CONTROLLER ログファイル

Automation Controller ログファイルには、次の 2 つの一元化された場所からアクセスできます。

- **/var/log/tower/**
- **/var/log/supervisor/**

/var/log/tower/ ディレクトリーでは、次の方法で取得されたログファイルを確認できます。

- **tower.log:** ジョブの実行時に発生するランタイムエラーなどのログメッセージを取得します。
- **callback_receiver.log:** Ansible ジョブの実行時にコールバックイベントを処理するコールバックレシーバーログを取得します。
- **dispatcher.log:** Automation Controller ディスパッチャワーカーサービスのログメッセージを取得します。
- **job_lifecycle.log:** ジョブ実行の詳細、ブロックの有無、ブロックの条件などを取得します。
- **management_playbooks.log:** 管理 Playbook の実行、およびメタデータのコピーといった分離ジョブの実行のログを取得します。
- **rsyslog.err:** 外部ロギングサービスへのログ送信時に、外部ロギングサービスでの認証で生じた rsyslog エラーを取得します。
- **task_system.log:** バックグラウンドで Automation Controller が実行しているタスクのログを取得します。たとえば、クラスターインスタンスの追加や、分析のための情報収集/処理に関連するログなどです。
- **tower_rbac_migrations.log:** rbac データベースの移行またはアップグレードに関するログを取得します。
- **tower_system_tracking_migrations.log:** コントローラーシステムによる移行またはアップグレードの追跡に関するログを取得します。
- **wsbroadcast.log:** コントローラーノードの websocket 接続のログを取得します。

/var/log/supervisor/ ディレクトリーでは、次の方法で取得されたログファイルを確認できます。

- **awx-callback-receiver.log: supervisord** で管理されている Ansible ジョブの実行時にコールバックイベントを処理するコールバックレシーバーのログを取得します。
- **awx-daphne.log:** WebUI の Websocket 通信のログを取得します。
- **awx-dispatcher.log:** ジョブの実行時など、タスクを Automation Controller インスタンスにディスパッチするときに発生するログを取得します。
- **awx-rsyslog.log: rsyslog** サービスのログを取得します。
- **awx-uwsgi.log:** アプリケーションサーバーである uWSGI に関連するログを取得します。
- **awx-wsbroadcast.log:** Automation Controller が使用する websocket サービスのログを取得します。
- **failure-event-handler.stderr.log: /usr/bin/failure-event-handler** の supervisord サブプロセスの標準エラーを取得します。

- **supervisord.log:** **supervisord** 自体に関連するログを取得します。
- **wsrelay.log:** websocket リレーサーバー内の通信ログをキャプチャーします。
- **ws_heartbeat.log:** ホスト上で実行しているサービスの健全性に関する定期的なチェックをキャプチャーします。
- **rsyslog_configurer.log:** 外部ロギングサービスによる認証に関連する rsyslog 設定アクティビティーをキャプチャーします。

/var/log/supervisor/ ディレクトリーには、全サービスの **stdout** ファイルも含まれます。

Automation Controller (および Ansible Automation Platform) が使用するサービスにより、次のログパスが生成されると予想されます。

- **/var/log/nginx/**
- **/var/lib/pgsql/data/pg_log/**
- **/var/log/redis/**

トラブルシューティング

エラーログは次の場所にあります。

- Automation Controller サーバーエラーは **/var/log/tower** に記録されます。
- Supervisor のログは **/var/log/supervisor/** にあります。
- Nginx Web サーバーのエラーは httpd エラーログに記録されます。
- その他の Automation Controller のロギングニーズは **/etc/tower/conf.d/** で設定します。

大半のブラウザーに組み込まれている JavaScript コンソールを使用してクライアント側の問題をチェックし、Red Hat カスタマーポータル (https://access.redhat.com/) 経由で Ansible にエラーを報告してください。

第12章 ロギングおよびアグリゲーション

ロギングは、詳細なログをサードパーティーの外部ログ集約サービスに送信する機能を提供します。このデータフィードに接続されたサービスは、Automation Controller の使用や技術の傾向に関する洞察を得るための手段として機能します。このデータを使用して、インフラストラクチャー内のイベントを分析し、異常を監視し、あるサービスのイベントを別のサービスのイベントと関連付けることができます。

Automation Controller に最も役立つデータのタイプは、ジョブファクトデータ、ジョブイベントまたはジョブ実行、アクティビティーストリームデータ、およびログメッセージです。データは、カスタムハンドラーで、またはインポートされたライブラリーを介して設計された最小限のサービス固有の調整を使用して、HTTP 接続を介して JSON 形式で送信されます。

Automation Controller によってインストールされる **rsyslog** のバージョンは、次の **rsyslog** モジュールを含みません。

- rsyslog-udpspoof.x86_64
- rsyslog-libdbi.x86_64

Automation Controller 外部のロギングは、以前は RHEL 提供の **rsyslog** パッケージを使用して実行していた可能性がありますが、Automation Controller をインストールした後は、Automation Controller 提供の **rsyslog** パッケージのみを使用する必要があります。

Automation Controller インスタンスでのシステムログの記録にすでに **rsyslog** を使用している場合は、別の **rsyslog** プロセス (Automation Controller が使用するのと同じバージョンの rsyslog を使用) を実行し、別の **/etc/rsyslog.conf** ファイルを指定することで、引き続き **rsyslog** を使用して Automation Controller の外部からのログを処理できます。

外部ロガーがオフラインになった場合に Automation Controller の **rsyslog** プロセスで未送信のメッセージを処理する方法を、**/api/v2/settings/logging/** エンドポイントを使用して設定します。

- **LOG_AGGREGATOR_MAX_DISK_USAGE_GB**: 外部ログアグリゲーターの停止中に保存するデータ量を指定します (ギガバイト単位、デフォルトは 1)。**rsyslogd queue.maxdiskspace** 設定に相当します。
- **LOG_AGGREGATOR_MAX_DISK_USAGE_PATH**: 外部ログアグリゲーターの停止後に再試行する必要があるログを保存する場所を指定します (デフォルトは **/var/lib/awx**)。**rsyslogd queue.spoolDirectory** 設定に相当します。

たとえば、**Splunk** がオフラインになった場合、**rsyslogd** は **Splunk** がオンラインに戻るまでディスクにキューを保存します。デフォルトでは、(Splunk がオフラインの間) 最大 1 GB のイベントが保存されますが、必要に応じて 1 GB 超に増やすことも、キューを保存するパスを変更することもできます。

12.1. ロガー

以下は、予測可能な構造化形式または半構造化形式で大量の情報を提供する特別なロガー (汎用的なサーバーログを構成する **awx** は除く) です。これらは、API からデータを取得するときと同じ構造を使用します。

- **job_events**: Ansible コールバックモジュールから返されるデータを渡します。
- **activity_stream**: アプリケーション内のオブジェクトに対する変更の記録を表示します。

- **system_tracking**: Ansible **setup** モジュールによって収集されたファクトデータを提供します。つまり、**Enable Fact Cache** を選択した状態でジョブテンプレートを実行する場合は、**gather_facts: true** になります。
- **awx**: 通常はファイルに書き込まれるログを含む、汎用的なサーバーログを提供します。これには、すべてのログが有する標準メタデータが含まれます。ただし、含まれるのはログステートメントからのメッセージのみです。

これらのロガーは、**INFO** のログレベルのみを使用します。例外は **awx** ロガーで、これはどのレベルでも使用できます。

さらに、標準の Automation Controller ログも、同じメカニズムで配信できます。ローカルの設定ファイルで複雑なディクショナリーを操作せずに、これら 5 つのデータソースをそれぞれ有効または無効にする方法や、標準の Automation Controller ログから使用するログレベルを調整する方法は明らかです。

ナビゲーションパネルから **Settings** を選択します。次に、**System** オプションのリストから **Logging settings** を選択して、Automation Controller のログコンポーネントを設定します。

12.1.1. ログメッセージスキーマ

全ロガーに対する共通のスキーマ

- **cluster_host_id**: Automation Controller クラスター内のホストの一意識別子。
- **level**: イベントの重要性をほぼ反映する、標準の Python ログレベル。すべてのデータロガーは 'レベル' の一部として **INFO** レベルを使用しますが、他の Automation Controller ログは必要に応じて異なるレベルを使用します。
- **logger_name**: 設定で使用するロガー名 (たとえば "activity_stream")。
- **@timestamp**: ログの時刻。
- **path**: ログが生成されたコードのファイルパス。

12.1.2. アクティビティーストリームスキーマ

これは、ログメッセージスキーマ にリストされているすべてのロガーに共通するフィールドを使用します。

次の追加フィールドがあります。

- **actor**: ログに記述されたアクションを行ったユーザーのユーザー名。
- **changes**: 変更されたフィールド、および古い値または新しい値の概要 (JSON)。
- **operation**: "associate" など、アクティビティーストリームにログ記録された変更の基本カテゴリー。
- **object1**: 操作を行うプライマリーオブジェクトに関する情報。アクティビティーストリームに表示する内容と一貫性があります。
- **object2**: 該当する場合には、アクションに関連する 2 つ目のオブジェクト。

このロガーは、ジョブイベントに保存されているデータを反映します。ただし、ロガーの予期される標準フィールドと競合する場合を除きます (この場合、フィールドはネストされます)。特に、**job_event** モデルのフィールド host は、**event_host** として指定されます。ペイロード内にはサブディクショナ

リーフィールド **event_data** もあり、これには Ansible イベントの詳細に応じて異なるフィールドが含まれます。

このロガーには、ログメッセージスキーマ の共通フィールドも含まれます。

12.1.3. スキャン/ファクト/システム追跡データスキーマ

これらには、サービス、パッケージまたはファイルの詳細なディクショナリータイプのフィールドが含まれます。

- **services**: サービススキャンの場合、このフィールドが含まれ、サービスの名前に基づいたキーを含みます。

注記

名前の Elastic Search ではピリオドは使用できず、ログフォーマッターによって "_" に置き換えられます。

- **package**: パッケージスキャンからのログメッセージに含まれます。
- **files**: ファイルスキャンからのログメッセージに含まれます。
- **host**: スキャンを適用するホストの名前です。
- **inventory_id**: ホストが含まれるインベントリー ID です。

このロガーには、ログメッセージスキーマ の共通フィールドも含まれます。

12.1.4. ジョブステータスの変更

この情報ソースは、ジョブイベントと比較してジョブ状態の変化に関する情報は少量とし、ジョブテンプレートをベースとするジョブ以外で、統合されたジョブタイプに対する変更を取得します。

このロガーには、ログメッセージスキーマ の共通フィールドと、ジョブモデルに存在するフィールドも含まれます。

12.1.5. Automation Controller のログ

このロガーには、ログメッセージスキーマ の共通フィールドも含まれます。

さらに、ログメッセージを含む **msg** フィールドが含まれます。エラーには別の **traceback** フィールドが含まれます。ナビゲーションパネルから **Settings** を選択します。次に、**System** オプションのリストから **Logging settings** を選択し、**ENABLE EXTERNAL LOGGING** オプションを使用してログ記録コンポーネントを有効または無効にします。

12.1.6. ログアグリゲーターサービス

ログアグリゲーターサービスは、以下の監視またはデータ分析システムと連携します。

- Splunk
- Loggly
- Sumologic

- Elastic Stack (以前の ELK stack)

12.1.6.1. Splunk

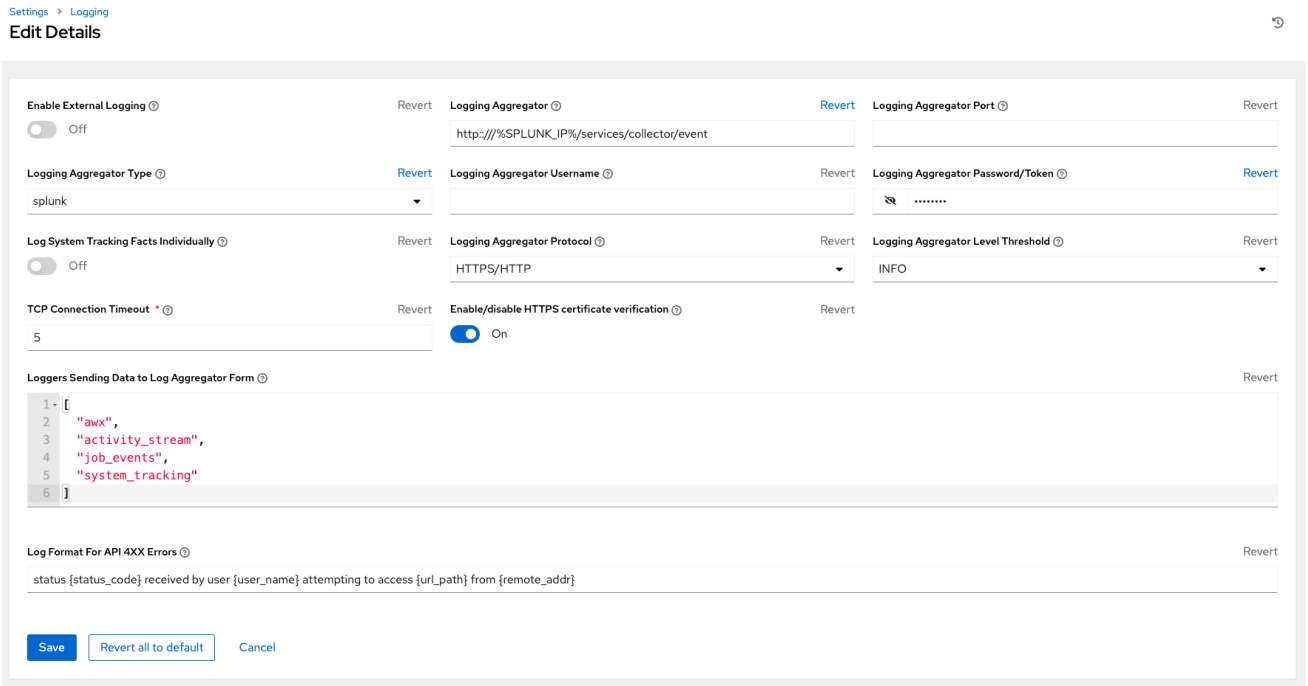
Automation Controller の Splunk ロギング統合では、Splunk HTTP Collector を使用します。SPLUNK ログアグリゲーターを設定する場合は、次の例のように、完全な URL を HTTP Event Collector ホストに追加します。

```
https://<yourcontrollerfqdn>/api/v2/settings/logging

{
    "LOG_AGGREGATOR_HOST": "https://<yoursplunk>:8088/services/collector/event",
    "LOG_AGGREGATOR_PORT": null,
    "LOG_AGGREGATOR_TYPE": "splunk",
    "LOG_AGGREGATOR_USERNAME": "",
    "LOG_AGGREGATOR_PASSWORD": "$encrypted$",
    "LOG_AGGREGATOR_LOGGERS": [
        "awx",
        "activity_stream",
        "job_events",
        "system_tracking"
    ],
    "LOG_AGGREGATOR_INDIVIDUAL_FACTS": false,
    "LOG_AGGREGATOR_ENABLED": true,
    "LOG_AGGREGATOR_CONTROLLER_UUID": ""
}
```

注記

Splunk HTTP Event Collector はデフォルトでポート 8088 をリッスンするため、受信要求が正常に処理されるように、**LOG_AGGREGATOR_HOST** に完全な HEC イベント URL (ポート番号を含む) を指定する必要があります。

一般的な値を次の例に示します。



HTTP Event Collector の設定の詳細は、Splunk のドキュメント を参照してください。

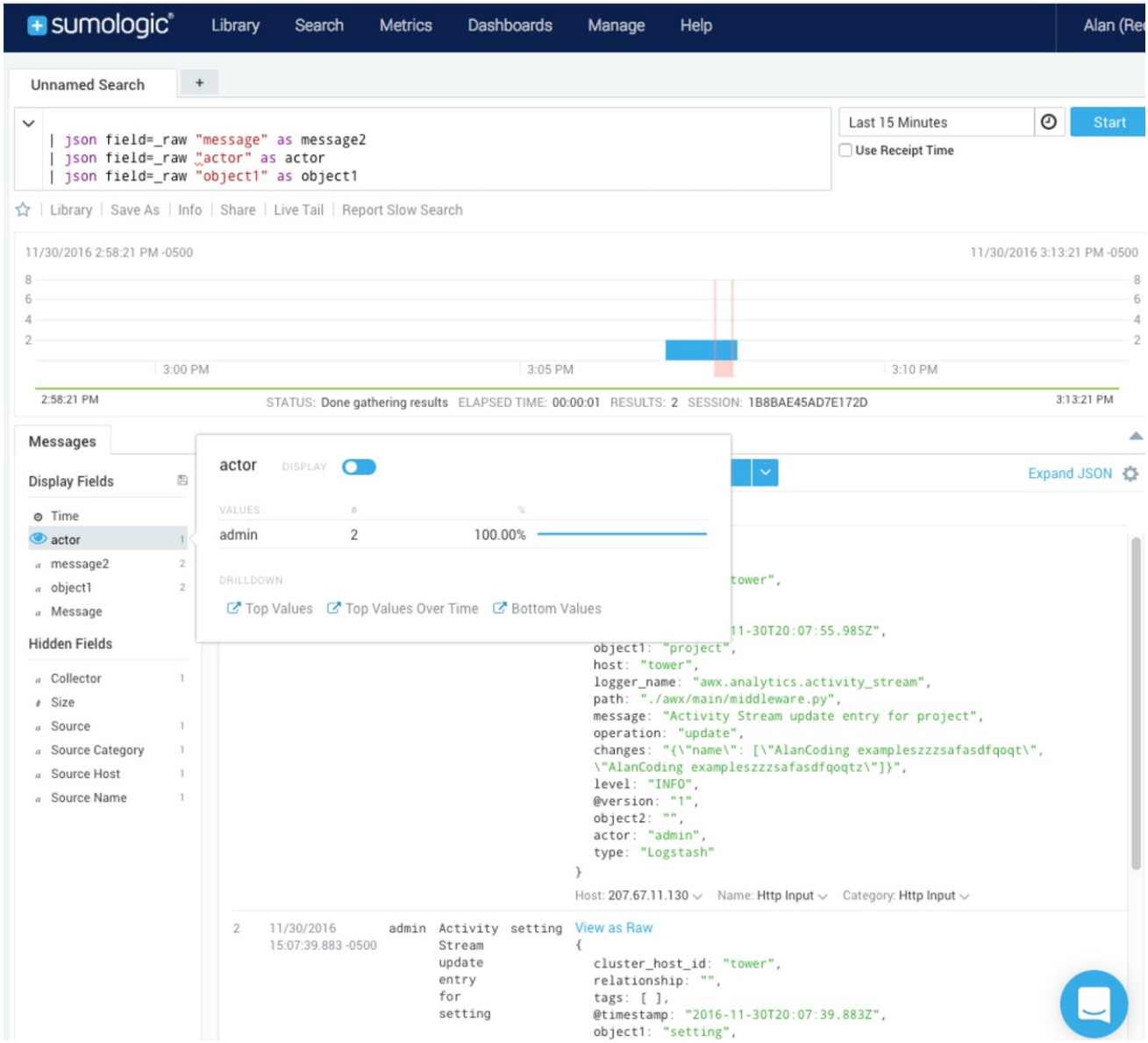
12.1.6.2. Loggly

Loggly の HTTP エンドポイントを介したログ送信の詳細は、Loggly のドキュメント を参照してください。

Loggly は、次の例の **Logging Aggregator** フィールドに示す URL 規則を使用します。

12.1.6.3. Sumologic

Sumologic では、JSON ファイルに含まれる検索基準を作成します。JSON ファイルには、必要なデータを収集するのに使用するパラメーターが指定されています。



12.1.6.4. Elastic stack (以前の ELK stack)

独自のバージョンの Elastic Stack をセットアップしている場合、必要な変更は、logstash **logstash.conf** ファイルに次の行を追加するだけです。

```
filter {
  json {
    source => "message"
  }
}
```

注記

Elastic 5.0.0 で後方互換性のない変更が導入され、使用するバージョンによっては、異なる設定が必要になる可能性があります。

12.2. ロギングのセットアップ

任意のアグリゲータタイプへのロギングをセットアップするには、次の手順を使用します。

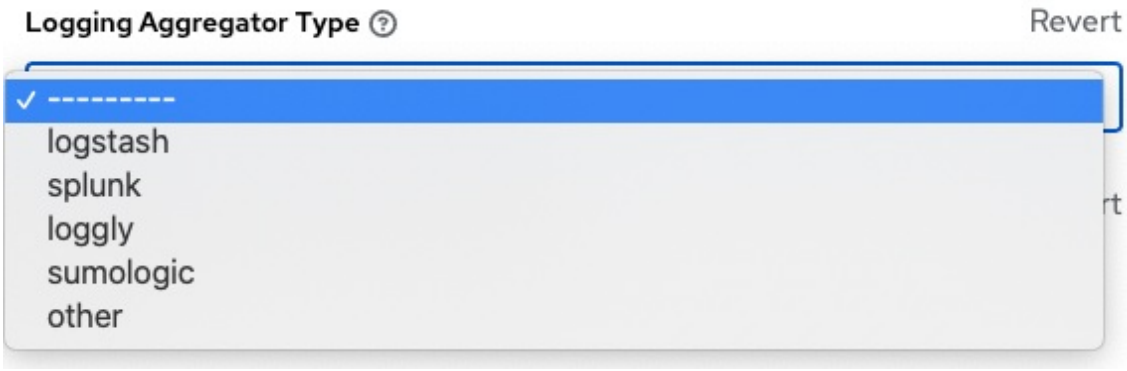
手順

1. ナビゲーションパネルから **Settings** を選択します。
2. **System** オプションのリストから **Logging settings** を選択します。
3. **Logging settings** 画面で、**Edit** をクリックします。
4. 次の設定可能なオプションを設定します。
   - **Enable External Logging**: ログを外部ログアグリゲーターに送信する場合は、トグルボタンをクリックして **ON** にします。
   - **Logging Aggregator**: ログを送信するホスト名または IP アドレスを入力します。
   - **Logging Aggregator Port**: アグリゲーターのポートが必要な場合は、ポートを指定します。

注記

接続タイプが HTTPS の場合、ホスト名をポート番号付きの URL として入力できます。その後はポートを再度入力する必要はありません。ただし、TCP 接続と UDP 接続は、URL ではなく、ホスト名とポート番号の組み合わせによって特定されます。したがって、TCP 接続または UDP 接続の場合は、指定されたフィールドでポートを指定します。代わりに URL が **Logging Aggregator** フィールドに入力された場合、そのホスト名部分がホスト名として抽出されます。

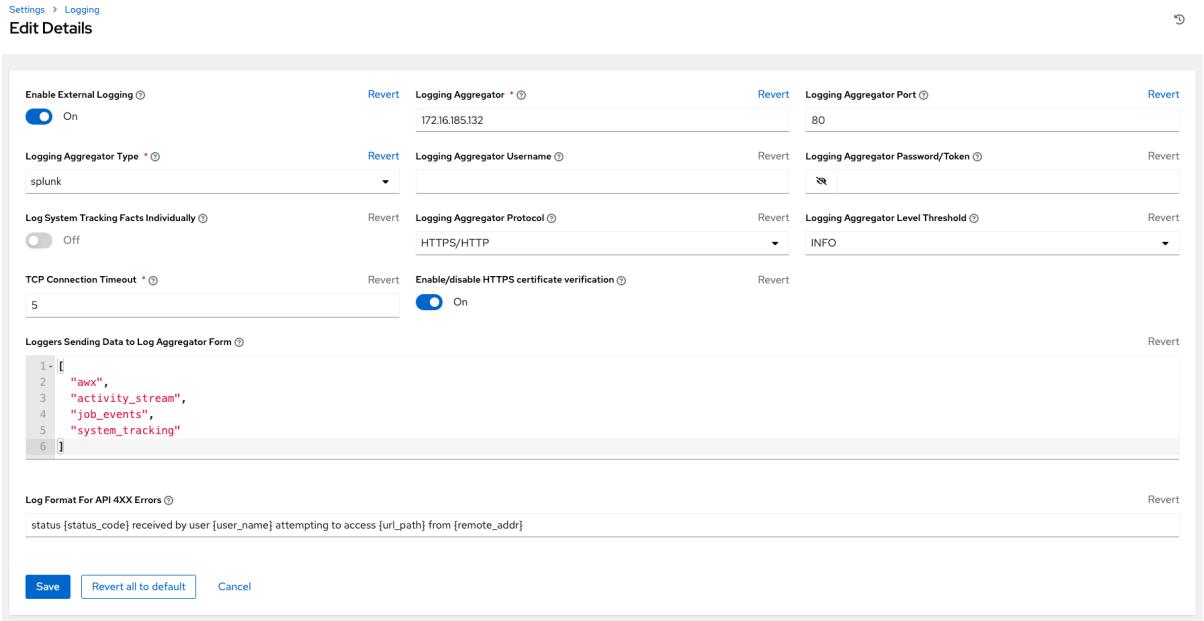
   - **Logging Aggregator Type**: クリックして、リストからアグリゲータサービスを選択します。



   - **Logging Aggregator Username**: 必要に応じて、ロギングアグリゲーターのユーザー名を入力します。
   - **Logging Aggregator Password/Token**: 必要に応じて、ロギングアグリゲーターのパスワードを入力します。
   - **Log System Tracking Facts Individually**: ツールチップアイコン ⓘ をクリックすると、オンに切り替えるか、デフォルトのままオフにしておくかなど、追加情報が表示されます。
   - **Logging Aggregator Protocol**: クリックして、ログアグリゲーターと通信するための接続タイプ (プロトコル) を選択します。その後に表示されるオプションは、選択したプロトコルによって異なります。

- **Logging Aggregator Level Threshold**: ログハンドラーで報告する重大度のレベルを選択します。
- **TCP Connection Timeout** 接続タイムアウトを秒単位で指定します。このオプションは、HTTPS および TCP ログアグリゲータープロトコルにのみ適用されます。
- **Enable/disable HTTPS certificate verification**: HTTPS ログプロトコルの証明書検証はデフォルトで有効になっています。接続を確立する前に、外部ログアグリゲーターによって送信された HTTPS 証明書をログハンドラーで検証しない場合は、トグルを **OFF** に設定します。
- **Loggers to Send Data to the Log Aggregator Form**デフォルトでは、4 種類のデータすべてが事前に入力されています。各データ型の追加情報を表示するには、フィールドの横にあるツールチップアイコン ⓘ をクリックします。不要なデータ型を削除します。
- **Log Format For API 4XX Errors** 特定のエラーメッセージを設定します。詳細は、API 4XX エラー設定 を参照してください。

5. 選択したロギングアグリゲーションのエントリーを確認します。次の例は Splunk 用にセットアップされています。



6. **Save** をクリックするか、変更を破棄する場合は **Cancel** をクリックします。

12.3. API 4XX エラー設定

要求で問題が発生すると、API は通常、エラーとともに 400 番台の HTTP エラーコードを返します。この場合、次のパターンに従ったエラーメッセージがログに生成されます。

> ' status {status_code} received by user {user_name} attempting to access {url_path} from {remote_addr} '

メッセージは必要に応じて設定できます。デフォルトの API 4XX エラーログメッセージの形式を変更するには、次の手順を使用します。

手順

1. ナビゲーションパネルから **Settings** を選択し、**Logging settings** を選択します。

2. **Logging settings** ページで、**Edit** をクリックします。
3. **Log Format For API 4XX Errors** フィールドを変更します。

{} で囲まれた項目はログエラー発生時に置換されます。次の変数を使用できます。

- **status_code**: API が返す HTTP ステータスコード。
- **user_name**: API 要求を行うときに認証されたユーザーの名前。
- **url_path**: 呼び出される URL のパス部分 (別名 API エンドポイント)
- **remote_addr** :Automation Controller が受信したリモートアドレス。
- **error**: API によって返されるエラーメッセージ。エラーが指定されていない場合は、HTTP ステータスをテキストで返します。

12.4. ロギングのトラブルシューティング

ロギングアグリゲーション
テストボタンを使用して、設定済みのロギングサービスに http または https 経由でメッセージを送信したものの、メッセージを受信しなかった場合は、**/var/log/tower/rsyslog.err** ログファイルを確認してください。http または https の外部ロギングサービスで rsyslog を認証するときにエラーが発生した場合、ここにエラーが保存されます。エラーがない場合、このファイルは存在しないことに注意してください。

API 4XX エラー
これらのメッセージのログ形式を変更することにより、4XX エラーの API エラーメッセージを含めることができます。API 4XX エラー設定 を参照してください。

LDAP
LDAP アダプターのロギングメッセージを有効にできます。詳細は、API 4XX エラー設定 を参照してください。

SAML
LDAP のログを有効にするのと同じ方法で、SAML アダプターのメッセージのロギングを有効にできます。詳細は、LDAP のロギングの有効化 セクションを参照してください。

第13章 メトリクス

API ではメトリクスエンドポイント **/api/v2/metrics/** が使用できます。このメトリクスエンドポイントは、Automation Controller に関する即時のメトリクスを生成します。メトリクスは、オープンソースプロジェクトの Prometheus といったシステム監視ソフトウェアで使用できます。

metrics/ エンドポイントに表示されるデータのタイプは、**Content-type: text/plain** および **application/json** です。

このエンドポイントには、アクティブなユーザーセッションの数や、各 Automation Controller ノードでアクティブに実行されているジョブの数などの有用な情報が含まれます。

Automation Controller メトリクスエンドポイントにアクセスし、このデータを時系列データベースに保存することで、Automation Controller からこれらのメトリクスを収集するように Prometheus を設定できます。

その後、クライアントは Prometheus を他のソフトウェア (Grafana や Metricbeat など) と組み合わせて使用して、データを視覚化したりアラートを設定したりできます。

13.1. PROMETHEUS のセットアップ

Prometheus をセットアップして使用するには、仮想マシンまたはコンテナーに Prometheus をインストールする必要があります。

詳細は、First steps with Prometheus ドキュメントを参照してください。

手順

1. Prometheus 設定ファイル (通常は **prometheus.yml**) で、**<token_value>**、作成した Automation Controller ユーザーの有効なユーザー名とパスワード、および **<controller_host>** を指定します。

注記

あるいは、OAuth2 トークン (**/api/v2/users/N/personal_tokens/** で生成可能) を指定することもできます。デフォルトでは、この設定はユーザー名 = **admin**、パスワード = **password** であるユーザーを想定しています。

次の例は、Automation Controller で Prometheus を認証するために **/api/v2/tokens** エンドポイントで作成した OAuth2 トークンを使用し、Automation Controller のメトリクスエンドポイントの URL が **https://controller_host:443/metrics** である場合の有効な収集設定です。

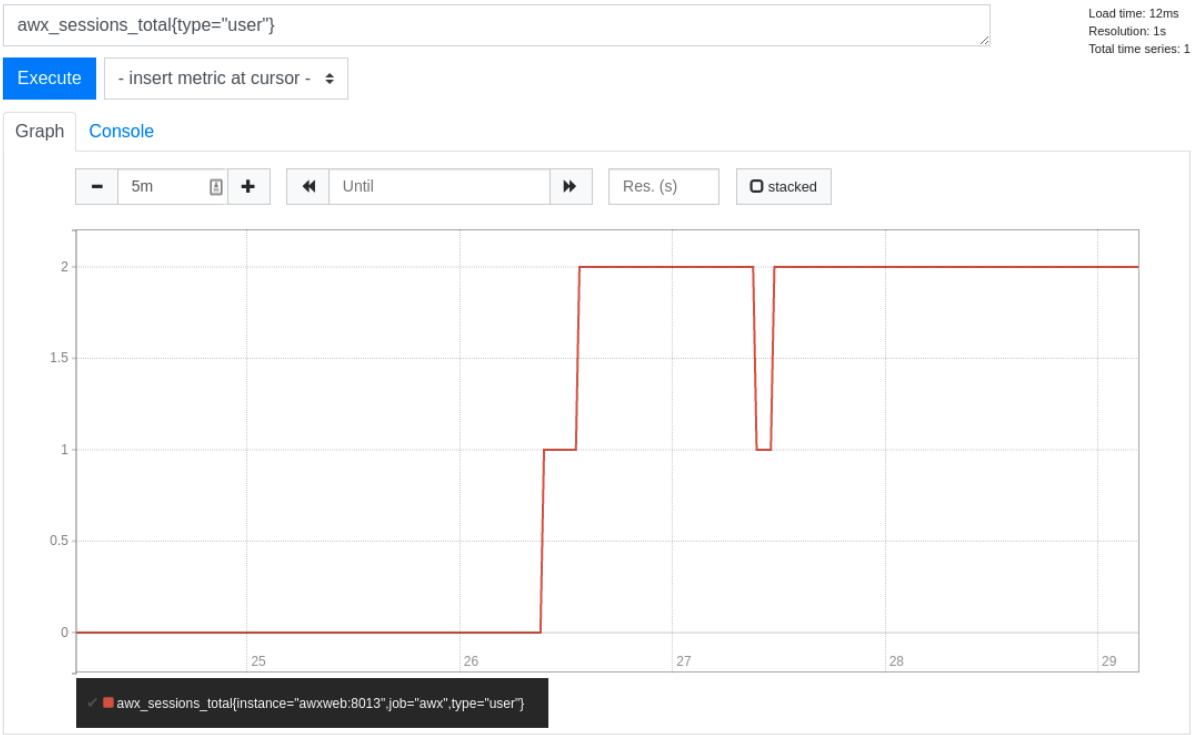
```
scrape_configs

  - job_name: 'controller'
    tls_config:
        insecure_skip_verify: True
    metrics_path: /api/v2/metrics
    scrape_interval: 5s
    scheme: https
    bearer_token: <token_value>
    # basic_auth:
    #   username: admin
    #   password: password
```

```
static_configs:
  - targets:
    - <controller_host>
```

アラートやサービスディスカバリー設定など、Prometheus の他の要素を設定する方法については、Prometheus configuration ドキュメントを参照してください。

Prometheus がすでに実行中の場合には、再読み込みしたエンドポイントに **POST** 要求を送信するか、Prometheus プロセスまたはサービスを強制終了し、Prometheus を再起動して設定の変更を適用する必要があります。

2. ブラウザーを使用して **http://<your_prometheus>:9090/graph** にある Prometheus UI のグラフに移動し、いくつかのクエリーをテストします。たとえば、**awx_sessions_total{type="user"}** を実行すると、現在のアクティブな Automation Controller ユーザーセッションの数をクエリーできます。



その他のクエリー方法については、インスタンスの Automation Controller API のメトリクスエンドポイント (**api/v2/metrics**) を参照してください。

第14章 AUTOMATION CONTROLLER のパフォーマンスチューニング

Automation Controller をチューニングしてパフォーマンスとスケーラビリティを最適化します。ワークロードを計画するときは、パフォーマンスとスケーリングのニーズを特定し、制限に合わせて調整し、デプロイメントを監視してください。

Automation Controller は、次のような複数の調整可能なコンポーネントを備えた分散システムです。

- ジョブのスケジューリングを担当するタスクシステム
- ジョブの制御と出力の処理を担当するコントロールプレーン
- ジョブが実行される実行プレーン
- API の提供を担当する Web サーバー
- websocket 接続とデータを提供およびブロードキャストする Websocket システム
- 複数のコンポーネントで使用されるデータベース

14.1. AUTOMATION CONTROLLER をデプロイするための容量のプランニング

Automation Controller の容量のプランニングでは、計画されたワークロードを実行する容量を確保できるように、デプロイメントのスケールと特性について計画します。容量のプランニングには次のフェーズが含まれます。

1. ワークロードの特徴付け
2. さまざまなノードタイプの機能の確認
3. ワークロードの要件に基づくデプロイメントのプランニング

14.1.1. ワークロードの特徴

デプロイメントを計画する前に、サポートするワークロードを確立します。Automation Controller ワークロードを特徴付けるために、次の要素を考慮してください。

- 管理対象ホスト
- ホストごとの 1 時間あたりのタスク数
- サポートする同時実行ジョブの最大数
- ジョブに設定するフォークの最大数。フォークによって、ジョブがいくつのホストで同時に動作するかが決まります。
- 1 秒あたりの API 要求の最大数
- デプロイするノードのサイズ (CPU/メモリー/ディスク)

14.1.2. Automation Controller のノードのタイプ

Automation Controller デプロイメントでは、4 種類のノードを設定できます。

- コントロールノード
- ハイブリッドノード
- 実行ノード
- ホップノード

14.1.2.1. コントロールノードをスケーリングする利点

コントロールノードとハイブリッドノードは、制御容量を提供します。これらのノードは、ジョブを開始し、データベースへの出力を処理する機能を提供します。すべてのジョブにはコントロールノードが割り当てられます。デフォルト設定では、各ジョブを制御するには 1 の容量単位が必要です。たとえば、100 の容量単位を持つコントロールノードは、最大 100 個のジョブを制御できます。

より多くのリソースを備えた大規模な仮想マシンをデプロイして、コントロールノードを垂直スケーリングすると、コントロールプレーンの次の機能が向上します。

- コントロールノードが制御タスクを実行できるジョブの数。この数を増やすには、より多くの CPU とメモリーが必要です。
- コントロールノードが同時に処理できるジョブイベントの数。

CPU とメモリーを同じ比率でスケーリングすることを推奨します (例: 1 CPU: 4 GB RAM)。メモリー消費量が多い場合でも、インスタンスの CPU を増やすことで負荷が軽減されることがよくあります。コントロールノードが消費するメモリーの大部分は、メモリーベースのキューに格納されている未処理のイベントによるものです。

注記

コントロールノードを垂直スケーリングしても、Web 要求を処理するワーカーの数は自動的に増加しません。

垂直スケーリングの代わりに、より多くのコントロールノードをデプロイして水平スケーリングを行うこともできます。これにより、ロードバランサーをプロビジョニングしてノード間で要求を分散する場合、制御タスクをより多くのノードに分散できるほか、Web トラフィックもより多くのノードに分散できます。より多くのコントロールノードをデプロイする水平スケーリングは、コントロールノードがダウンした場合や通常よりも高い負荷が発生した場合に、冗長性とワークロードの分離を強化するため、多くの点で望ましい場合があります。

14.1.2.2. 実行ノードをスケーリングする利点

実行ノードとハイブリッドノードは実行容量を提供します。ジョブが消費する容量は、ジョブテンプレートに設定されているフォークの数とインベントリー内のホストの数のいずれか少ない方に、メインの Ansible プロセスを考慮した 1 容量単位を加えたものと等しくなります。たとえば、デフォルトのフォーク値が 5 のジョブテンプレートが 50 台のホストを持つインベントリーで動作している場合、このジョブテンプレートは、割り当てられている実行ノードから 6 容量単位を消費します。

より多くのリソースを備えたより大規模な仮想マシンをデプロイすることで実行ノードを垂直スケーリングすると、ジョブ実行のためのフォークが増加します。これにより、インスタンスで実行できる同時実行ジョブの数が増加します。

一般に、CPU とメモリーを同じ比率でスケーリングすることを推奨します。コントロールノードやハイブリッドノードと同様に、各実行ノードには容量調整があります。容量調整を使用して、Automation Controller が作成する容量消費の推定値に合わせて実際の使用量を調整できます。デフォルトでは、す

べてのノードがその範囲の上限に設定されます。実際の監視データによりノードが過剰に使用されていると判明した場合、容量調整を減らすことで、実際の使用量と一致させることができます。

実行ノードを垂直スケーリングする代わりに、より多くの仮想マシンを実行ノードとしてデプロイして、実行プレーンを水平スケーリングすることもできます。水平スケーリングによりワークロードをさらに分離できるため、異なるインスタンスを異なるインスタンスグループに割り当てることができます。これらのインスタンスグループは、組織、インベントリー、またはジョブテンプレートに割り当てることができます。たとえば、特定のインベントリーに対してジョブを実行する場合にのみ使用できるインスタンスグループを設定できます。この場合、実行プレーンを水平スケーリングすることで、優先度の低いジョブが優先度の高いジョブをブロックしないようにすることができます。

14.1.2.3. ホップノードをスケーリングする利点

ホップノードが使用するメモリーと CPU は非常に少ないため、これらのノードを垂直スケーリングしても容量には影響しません。多くの実行ノードとコントロールプレーンの間の唯一の接続として機能するホップノードについては、ネットワーク帯域幅を監視してください。帯域幅の使用量が飽和している場合は、ネットワークの変更を検討してください。

ホップノードを追加して水平スケーリングすると、1 つのホップノードがダウンしても冗長性が提供され、コントロールプレーンと実行ノードの間でトラフィックを流し続けることができます。

14.1.2.4. 制御容量と実行容量の比率

デフォルト設定を前提とすると、従来の仮想マシンのデプロイメントでは、制御容量と実行容量の最大推奨比率は 1:5 です。この比率により、利用可能なすべての実行容量でジョブを実行して出力を処理するための、十分な制御容量が確保されます。実行容量に対して制御容量が少ないと、実行容量を使用する十分な数のジョブを起動できなくなります。

この比率を 1:1 に近づけることが推奨される場合があります。たとえば、ジョブが高レベルのジョブイベントを生成する場合、制御容量に対して実行容量を減らすと、出力を処理するためにコントロールノードに掛かる負荷が軽減されます。

14.2. 容量プランニングの演習例

サポートするワークロード容量を決定したら、ワークロードの要件に基づいてデプロイメントを計画する必要があります。デプロイメントに役立つように、次のプランニング演習を確認してください。

この例では、クラスターは次の容量をサポートする必要があります。

- 300 台の管理対象ホスト
- ホストごとに 1 時間あたり 1,000 タスク、またはホストごとに 1 分あたり 16 タスク
- 10 個の同時ジョブ
- Playbook でフォークが 5 に設定されている。これはデフォルトになります。
- 1 MB の平均イベントサイズ

仮想マシンには、4 つの CPU と 16 GB の RAM、および 3000 IOP のディスクが搭載されています。

14.2.1. ワークロード要件の例

この容量プランニングの演習例では、次のワークロード要件を使用します。

実行容量

- 10 個の同時実行ジョブを実行するには、少なくとも 60 単位の実行容量が必要です。
    - これは、(10 個のジョブ * 5 つのフォーク) + (10 個のジョブ * ジョブの 1 基本タスクインパクト) = 60 実行容量という式を使用して計算します。

制御容量

- 10 個の同時実行ジョブを制御するには、少なくとも 10 ユニットの制御容量が必要です。
- 300 台の管理対象ホストで、ホストごとに 1 時間あたり 1,000 個のタスクをサポートするために必要な 1 時間あたりのイベント数を計算するには、次の式を使用します。
    - 1 時間あたり 1000 個のタスク * 300 台の管理対象ホスト = 1 時間あたり少なくとも 300,000 個のイベント
    - ジョブが生成するイベントの数を正確に確認するには、ジョブを実行する必要があります。これは、イベントの数が特定のタスクと詳細度に依存するためです。たとえば、"Hello World" を出力するデバッグタスクは、1 つのホストで詳細度 1 のジョブイベントを 6 つ生成します。詳細度を 3 にすると、1 つのホストで 34 個のジョブイベントを生成します。したがって、タスクは少なくとも 6 つのイベントを生成すると推定する必要があります。これにより、1 時間あたり 3,000,000 個近くのイベント、つまり 1 秒あたり約 833 個のイベントが生成されることになります。

必要な実行ノードとコントロールノードの数の決定

必要な実行ノードとコントロールノードの数を決定するには、次の表に示す実験結果を参照してください。この表は、1 つのコントロールノードに対して同サイズの実行ノードが 5 つある場合に確認されたイベント処理速度を示しています (API 容量の列)。ジョブテンプレートのデフォルトの "フォーク" 設定は 5 です。そのため、このデフォルト値を使用すると制御容量と実行容量の比率が前述の 1:5 になり、ディスパッチ可能な最大数のジョブをコントロールノードが実行ノードにディスパッチした場合に、CPU/RAM が等しい 5 つの実行ノードが容量を 100% 使用することになります。

ノード	API 容量	デフォルトの実行容量	デフォルトの制御容量	容量使用率 100% での平均イベント処理速度	容量使用率 50% での平均イベント処理速度	容量使用率 40% での平均イベント処理速度
2.5 Ghz の 4 CPU、16 GB RAM コントロールノード、最大 3000 IOP のディスク	1 秒あたり約 10 の要求	N/A	137 ジョブ	1 秒あたり 1100	1 秒あたり 1400	1 秒あたり 1630
2.5 Ghz の 4 CPU、16 GB RAM 実行ノード、最大 3000 IOP のディスク	N/A	137	N/A	N/A	N/A	N/A
2.5 Ghz の 4 CPU、16 GB RAM データベースノード、最大 3000 IOP のディスク	N/A	N/A	N/A	N/A	N/A	N/A

ジョブの制御はコントロールノード上のジョブイベント処理と競合するため、制御容量をオーバープロビジョニングすると処理時間が短縮される可能性があります。処理時間が長い場合、ジョブの実行から API または UI で出力を表示できるようになるまでに遅延が発生する可能性があります。

この例において、300 台の管理対象ホストのワークロードがあり、ホストごとに 1 時間あたり 1000 個のタスクを実行し、Playbook でフォークを 5 に設定した 10 個の同時実行ジョブがあり、平均イベントサイズが 1 MB の場合、次の手順を使用します。

- 4 つの 2.5Ghz の CPU、16 GB の RAM、および約 3000 IOP のディスクを備えた 1 つの実行ノード、1 つのコントロールノード、1 つのデータベースノードをデプロイします。
- ジョブテンプレートで、フォーク設定をデフォルトの 5 に維持します。
- コントロールノードの UI のインスタンスビューで容量調整機能を使用して、容量を最低値の 16 まで減らし、イベント処理用に確保するコントロールノードの容量を増やします。

関連情報

- 高レベルの API インタラクションを伴うワークロードの詳細は、Scaling Automation Controller for API Driven Workloads を参照してください。
- インスタンスによる容量の管理の詳細は、インスタンスによる容量の管理 を参照してください。
- Operator ベースのデプロイメントの詳細は、Operator ベースのインストールにおける Red Hat Ansible Automation Platform のパフォーマンスに関する考慮事項 を参照してください。

14.3. AUTOMATION CONTROLLER のパフォーマンスのトラブルシューティング

要求のタイムアウト (504 または 503 エラー) が多く発生するか、一般に API レイテンシーが増大します。UI では、クライアントはログインに時間がかかり、ページがロードされるまでの待機時間が長くなります。どのシステムが原因であると考えられますか?

- これらの問題がログイン時にのみ発生し、外部認証を使用している場合は、外部認証プロバイダーの統合に問題がある可能性があります。エンタープライズ認証のセットアップ を参照するか、Red Hat サポートにお問い合わせください。
- タイムアウトや API レイテンシーの増大に関するその他の問題については、Web サーバーのチューニング を参照してください。

ジョブ出力がロードされるまでの待機時間が長くなります。

- ジョブ出力は、ansible-playbook を実際に実行する実行ノードから、関連するコントロールノードにストリーミングされます。次に、コールバックレシーバーがこのデータをシリアライズし、データベースに書き込みます。確認およびチューニングする関連設定については、ジョブイベント処理を管理するための設定 および Automation Controller の PostgreSQL データベースの設定およびメンテナンス を参照してください。
- 一般に、この症状を解決するには、コントロールノードの CPU とメモリーの使用状況を確認することが重要です。CPU またはメモリーの使用率が非常に高い場合は、より多くの仮想マシンをコントロールノードとしてデプロイし、コントロールプレーンを水平スケーリングすることで作業をさらに分散させるか、コントロールノードが一度に管理するジョブの数を変更します。詳細は、コントロールノードと実行ノードの容量設定 を参照してください。

Automation Controller が同時に実行できるジョブの数を増やすにはどうすればよいですか?

- ジョブが “保留” 状態のままになる要因は次のとおりです。
  - **“依存関係” が完了するのを待機している**: これには、“起動時の更新” 動作が有効になっている場合のプロジェクトの更新とインベントリーの更新が含まれます。
  - **ジョブテンプレートの "allow_simultaneous" 設定**: 同じジョブテンプレートの複数のジョブが "保留" 状態の場合は、ジョブテンプレートの "allow_simultaneous" 設定 (UI の "Concurrent Jobs" チェックボックス) を確認します。これが有効になっていない場合、ジョブテンプレートのジョブは一度に 1 つしか実行できません。
  - **ジョブテンプレートの “forks” 値**: デフォルト値は 5 です。ジョブの実行に必要な容量は、おおよそフォーク値と等しくなります (多少のオーバーヘッドは考慮されます)。フォーク値を非常に大きな数に設定すると、それを実行できるノードが制限されます。
  - **制御容量または実行容量の不足**: /api/v2/metrics で利用可能なアプリケーションメトリクスの “awx_instance_remaining_capacity” メトリクスを確認します。メトリクスを監視する方法の詳細は、Automation Controller アプリケーションを監視するためのメトリクス を参照してください。必要な数のジョブを処理するようにデプロイメントを計画する方法については、Automation Controller をデプロイするための容量のプランニング を参照してください。

ローカルマシンよりも Automation Controller の方がジョブの実行に時間がかかります。

- オーバーヘッドは多少増大すると想定されています。これは、Automation Controller が別のノードにジョブをディスパッチしている場合があるためです。この場合、Automation Controller はコンテナーを起動し、そこで ansible-playbook を実行し、すべての出力をシリアライズしてデータベースに書き込みます。
- 起動時のプロジェクト更新と、起動時のインベントリー更新の動作により、ジョブの開始時に遅延が増大する可能性があります。
- プロジェクトはコントロールノードで更新されて実行ノードに転送されるため、プロジェクトのサイズがジョブの開始にかかる時間に影響する可能性があります。内部クラスターリーティングは、ネットワークのパフォーマンスに影響を与える可能性があります。詳細は、内部クラスターリーティング を参照してください。
- コンテナーのプル設定は、ジョブの開始時間に影響を与える可能性があります。実行環境は、ジョブの実行に使用するコンテナーです。コンテナーのプル設定は、“Always”、“Never”、または “If not present” に設定できます。コンテナーが常にプルされている場合、遅延が発生する可能性があります。
- すべてのクラスターノード (実行ノード、コントロールノード、データベースノードを含む) が、最小要件の IOPS を満たすストレージを備えたインスタンスにデプロイされていることを確認してください。これは、Automation Controller が Ansible を実行してイベントデータをキャッシュする方法では、大量のディスク I/O が生じるためです。詳細は、Red Hat Ansible Automation Platform のシステム要件 を参照してください。

データベースストレージが増加し続けます。

- Automation Controller には、“Cleanup Job Details” というタイトルの管理ジョブがあります。デフォルトでは、120 日分のデータを保持し、週に 1 回実行するように設定されています。データベース内のデータ量を減らすために、保持期間を短縮できます。詳細は、以前のアクティビティーストリームデータの削除 を参照してください。
- クリーンアップジョブを実行すると、データベース内のデータが削除されます。ただし、データベースはいずれかの時点で、ストレージを解放するバキューム操作を実行する必要があります。データベースのバキューム処理の詳細は、Automation Controller の PostgreSQL データ

ベースの設定およびメンテナンス を参照してください。

14.4. AUTOMATION CONTROLLER を監視するためのメトリクス

Automation Controller ホストをシステムレベルとアプリケーションレベルで監視します。

システムレベルの監視では、次の情報が含まれます。

- ディスク I/O
- RAM 使用率
- CPU 使用率
- ネットワークトラフィック

アプリケーションレベルのメトリクスは、アプリケーションがシステムについて把握しているデータを提供します。このデータには次の情報が含まれます。

- 特定のインスタンスで実行されているジョブの数
- クラスター内のインスタンスに関する容量情報
- 存在するインベントリーの数
- インベントリーに含まれるホストの数

システムおよびアプリケーションのメトリクスを使用すると、サービスが低下したときにアプリケーションで何が起きていたのかを特定するのに役立ちます。Automation Controller の経時的なパフォーマンスに関する情報は、問題の診断や、将来の拡張に備えた容量プランニングを行うときに役立ちます。

14.4.1. Automation Controller アプリケーションを監視するためのメトリクス

アプリケーションレベルの監視のために、Automation Controller は、API エンドポイント **/api/v2/metrics** で Prometheus スタイルのメトリクスを提供します。これらのメトリクスを使用して、ジョブの出力処理やジョブのスケジューリングなどに関する、ジョブのステータスとサブシステムのパフォーマンスについての集計データを監視します。

メトリクスエンドポイントには、各メトリクスの説明が含まれます。パフォーマンスに関する特に重要なメトリクスの例は以下のとおりです。

- **awx_status_total**
  - 各ステータスにあるジョブの現在の合計です。他のイベントをシステム内のアクティビティーに関連付けるのに役立ちます。
  - エラーまたは失敗したジョブの増加を監視できます。
- awx_instance_remaining_capacity
  - 追加のジョブを実行するために残っている容量です。
- **callback_receiver_event_processing_avg_seconds**
  - 通称 “ジョブイベントラグ” です。
  - Ansible でタスクが発生してからユーザーがそれを確認できるようになるまでのラグタイム

の移動平均です。これは、コールバックレシーバーがイベントの処理でどれだけ遅延しているかを示します。この数が非常に大きい場合、ユーザーはコントロールプレーンをスケールアップするか、容量調整機能を使用してコントロールノードが制御するジョブの数を減らすことを検討できます。

- **callback_receiver_events_insert_db**
  - ノードによって挿入されたイベントのカウンターです。特定の期間におけるジョブイベント挿入率を計算するために使用できます。
- **callback_receiver_events_queue_size_redis**
  - コールバックレシーバーがイベントの処理でどれだけ遅延しているかを示すインジケーターです。値が高すぎると、Redis によってコントロールノードのメモリー不足 (OOM) が発生する可能性があります。

14.4.2. システムレベルの監視

インスタンスの容量管理では、ホストの実際のリソース使用量がイントロスペクトされないため、クラスターホストの CPU とメモリーの使用状況を監視することが重要です。自動化ジョブのリソースへの影響は、Playbook が実行する内容によって異なります。たとえば、多くのクラウドモジュールやネットワークモジュールは、Ansible Playbook を実行する実行ノードでほとんどの処理を行います。Automation Controller への影響は、“yum” のようなネイティブモジュールを実行している場合とは大きく異なります。このようなネイティブモジュールの場合、作業はターゲットホストで実行され、タスクの実行中、実行ノードは多くの時間を結果の待機に費やします。

CPU またはメモリーの使用率が非常に高い場合は、Automation Controller で、関連するインスタンスの容量調整を下げることを検討してください (インスタンスの詳細ページで行うことができます)。これにより、当該インスタンスで実行または制御するジョブの数が制限されます。

ディスク I/O とシステムの使用状況を監視します。Automation Controller ノードが Ansible を実行し、出力をファイルシステムにキャッシュして、最終的にデータベースに保存する方法では、多くのディスク読み取りと書き込みが発生します。ディスクのパフォーマンス低下を早期に特定すると、ユーザーエクスペリエンスの低下やシステムのパフォーマンス低下を防ぐのに役立ちます。

関連情報

- 監視の設定についての詳細は、メトリクス を参照してください。
- Automation Analytics のデータ収集を有効にすると、自動化の使用状況に関してさらなる洞察が得られます。詳細は、Red Hat Ansible Automation Platform 向けの Automation Analytics と Red Hat Insights を参照してください。

14.5. AUTOMATION CONTROLLER の POSTGRESQL データベースの設定およびメンテナンス

Automation Controller のパフォーマンスを向上させるために、データベースで次の設定パラメーターを設定できます。

メンテナンス

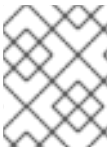
VACUUM タスクと **ANALYZE** タスクは、パフォーマンスに影響を与える可能性がある重要なメンテナンス作業です。通常の PostgreSQL 操作では、更新によって削除または不要になったタプルはテーブルから物理的に削除されません。これらは、**VACUUM** が完了するまで残ります。したがって、頻繁に更新されるテーブルでは特に、定期的に VACUUM を実行する必要があります。**ANALYZE** は、データベース内のテーブルの内容に関する統計情報を収集し、結果を **pg_statistic** システムカタログに保存し

ます。その後、クエリープランナーはこれらの統計情報を使用して、クエリーの最も効率的な実行計画を決定します。自動バキュームに関する PostgreSQL 設定パラメーターは、**VACUUM** コマンドおよび **ANALYZE** コマンドの実行を自動化します。自動バキュームは **true** に設定することを推奨します。ただし、データベースにアイドル時間がまったくない場合、自動バキュームは実行されません。自動バキュームによってデータベースディスクの領域が十分にクリーンアップされていないことが確認された場合、特定のメンテナンス期間中に特定のバキュームタスクをスケジューリングすることで解決する場合があります。

設定パラメーター

PostgreSQL サーバーのパフォーマンスを向上させるには、データベースメモリーを管理する次の **Grand Unified Configuration** (GUC) パラメーターを設定します。これらのパラメーターは、データベースサーバーの設定を管理する **postgresql.conf** ファイルの **$PDATA** ディレクトリー内にあります。

- **shared_buffers**: データをキャッシュするためにサーバーに割り当てるメモリーの量を決定します。このパラメーターのデフォルト値は 128 MB です。この値を変更する場合は、マシンの合計 RAM の 15% - 25% に設定する必要があります。

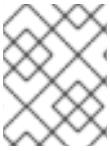


注記

shared_buffers の値を変更した後は、データベースサーバーを再起動する必要があります。

- **work_mem**: ディスクスワッピングの前に内部ソート操作とハッシュテーブルで使用されるメモリーの量を指定します。ソート操作は、order by、distinct、および merge join 操作に使用されます。ハッシュテーブルは、hash join と hash-based aggregation で使用されます。このパラメーターのデフォルト値は 4 MB です。**work_mem** パラメーターに正しい値を設定すると、ディスクスワッピングが減少し、検索速度が向上します。
  - 次の式を使用して、データベースサーバーの **work_mem** パラメーターの最適な値を計算します。

```
Total RAM * 0.25 / max_connections
```



注記

work_mem に大きな値を設定すると、データベースに対して開いている接続が多すぎる場合に、PostgreSQL サーバーがメモリー不足 (OOM) になる可能性があります。

- **max_connections**: データベースサーバーへの同時接続の最大数を指定します。
- **maintenance_work_mem**: vacuum、create index、および alter table add foreign key 操作などのメンテナンス操作で使用するメモリーの最大量を指定します。このパラメーターのデフォルト値は 64 MB です。このパラメーターの値を計算するには、次の式を使用します。

```
Total RAM * 0.05
```

注記

バキューム処理のパフォーマンスを向上させるには、**maintenance_work_mem** を **work_mem** よりも高く設定します。

関連情報

自動バキューム設定に関する詳細は、Automatic Vacuuming を参照してください。

14.6. AUTOMATION CONTROLLER のチューニング

Automation Controller UI、API、およびファイルベースの設定を使用して、次のような多くの Automation Controller の設定が可能です。

- Automation Controller UI のライブイベント
- ジョブイベント処理
- コントロールノードおよび実行ノードの容量
- インスタンスグループとコンテナーグループの容量
- タスク管理 (ジョブのスケジューリング)
- 内部クラスタールーティング
- Web サーバーのチューニング

14.6.1. Automation Controller UI でのライブイベントの管理

イベントは、ジョブにサブスクライブされた UI クライアントが存在するノードすべてに送信されます。このタスクはコストが高く、クラスターが生成するイベントの数が増加し、コントロールノードの数が増加するにつれて、さらにコストが高くなります。これは、特定のジョブにサブスクライブされたクライアントの数に関係なく、すべてのイベントがすべてのノードにブロードキャストされるためです。

UI にライブイベントを表示するオーバーヘッドを削減するために、管理者は次のいずれかを選択できます。

- ライブストリーミングイベントを無効にします。
- UI でイベントを縮小したり非表示にしたりする前に、1 秒あたりに表示されるイベントの数を減らします。

イベントのライブストリーミングを無効にすると、イベントは、ジョブの出力詳細ページのハードリフレッシュ時にのみロードされます。1 秒あたりに表示されるイベントの数を減らすと、ライブイベントを表示するオーバーヘッドは限定されますが、ハードリフレッシュを行わなくても UI でライブ更新が提供されます。

14.6.1.1. ライブストリーミングイベントの無効化

手順

1. 次のいずれかの方法を使用して、ライブストリーミングイベントを無効にします。
   a. API で、**UI_LIVE_UPDATES_ENABLED** を **False** に設定します。
   b. Automation Controller に移動します。**Miscellaneous System Settings** ウィンドウを開きます。**Enable Activity Stream** のトグルを **Off** に設定します。

14.6.1.2. イベントのレートとサイズを変更する設定

イベントのサイズが原因でイベントのライブストリーミングを無効にできない場合は、UI に表示されるイベントの数を減らします。次の設定を使用して、表示されるイベントの数を管理できます。

UI または API で編集できる設定

- **EVENT_STDOUT_MAX_BYTES_DISPLAY**: 表示する **stdout** の最大量 (バイト単位で測定)。これにより、UI に表示されるサイズが縮小されます。
- **MAX_WEBSOCKET_EVENT_RATE**: 1 秒あたりにクライアントに送信するイベントの数。

ファイルベースの設定を使用して利用可能な設定:

- **MAX_UI_JOB_EVENTS**: 表示するイベントの数。この設定により、リスト内の残りのイベントが非表示になります。
- **MAX_EVENT_RES_DATA**: Ansible コールバックイベントの "res" データ構造の最大サイズ。"res" はモジュールの完全な結果です。Ansible コールバックイベントの最大サイズに達すると、残りの出力は切り捨てられます。デフォルト値は 700000 バイトです。
- **LOCAL_STDOUT_EXPIRE_TIME**: **stdout** ファイルの有効期限が切れてローカルで削除されるまでの時間。

関連情報

ファイルベースの設定の詳細は、Automation Controller の追加設定 を参照してください。

14.6.2. ジョブイベント処理を管理するための設定

コールバックレシーバーはジョブのすべての出力を処理し、この出力をジョブイベントとして Automation Controller データベースに書き込みます。コールバックレシーバーには、イベントをバッチで処理するワーカーのプールがあります。ワーカーの数は、インスタンスで使用可能な CPU の数に応じて自動的に増加します。

管理者は、**JOB_EVENT_WORKERS** 設定を使用してコールバックレシーバーワーカーの数をオーバーライドできます。CPU ごとに複数のワーカーを設定することはできません。また、少なくとも 1 つのワーカーが必要です。値が大きいほど、Automation Controller にイベントがストリーミングされる際に、Redis キューをクリアするのに利用できるワーカーが増えますが、Web サーバーなどの他のプロセスと CPU 秒数で競合する可能性があります。また、より多くのデータベース接続 (ワーカーあたり 1 つ) を使用するほか、各ワーカーがコミットするイベントのバッチサイズが小さくなる可能性があります。

各ワーカーにより、バッチで書き込むイベントのバッファーが増大します。バッチを書き込む前に待機するデフォルトの時間は 1 秒です。これは、**JOB_EVENT_BUFFER_SECONDS** 設定によって制御されます。ワーカーがバッチ間で待機する時間を増やすと、バッチサイズが大きくなる可能性があります。

14.6.3. コントロールノードと実行ノードの容量設定

次の設定は、クラスターの容量計算に影響します。次のファイルベースの設定を使用して、すべてのコントロールノードでこれらを同じ値に設定します。

- **AWX_CONTROL_NODE_TASK_IMPACT**: ジョブの制御の影響を設定します。コントロールプレーンが望ましい使用量を超えて CPU またはメモリーを使用している場合に、この設定を使用して、コントロールプレーンが同時に実行可能なジョブの数を制御できます。

- **SYSTEM_TASK_FORKS_CPU** および **SYSTEM_TASK_FORKS_MEM**: Ansible の各フォークが消費すると推定されるリソースの数を制御します。デフォルトでは、Ansible の 1 フォークは、0.25 の CPU と 100 MB のメモリーを使用すると推定されます。

関連情報

ファイルベースの設定については、Automation Controller の追加設定 を参照してください。

14.6.4. インスタンスグループとコンテナーグループの容量設定

インスタンスグループで使用できる **max_concurrent_jobs** および **max_forks** 設定を使用して、インスタンスグループまたはコンテナーグループ全体で消費できるジョブとフォークの数を制限します。

- コンテナーグループで必要な **max_concurrent_jobs** を計算するには、そのコンテナーグループの **pod_spec** 設定を考慮してください。**pod_spec** では、自動化ジョブ Pod のリソース要求と制限を確認できます。次の式を使用して、必要な同時実行ジョブの最大数を計算します。

> ((number of worker nodes in kubernetes cluster) * (CPU available on each worker)) / (CPU request on pod_spec) = maximum number of concurrent jobs

- たとえば、**pod_spec** で、Pod が 250 mcpu を要求することが示され、Kubernetes クラスターに 2 CPU を備えたワーカーノードが 1 つある場合、開始する必要があるジョブの最大数は 8 です。
    - ジョブのフォークのメモリー消費も考慮することができます。次の式を使用して、**max_forks** の適切な設定を計算します。

> ((number of worker nodes in kubernetes cluster) * (memory available on each worker)) / (memory request on pod_spec) = maximum number of forks

- たとえば、8 GB のメモリーを備えたワーカーノードが 1 つあるとすれば、実行する **max forks** は 81 であると判断します。この場合、1 つのフォークを持つ 39 個のジョブを実行するか (タスクインパクトは常にフォーク +1 です)、またはフォークが 39 に設定されたジョブを 2 つ実行できます。
    - 他のビジネス要件では、**max_forks** または **max_concurrent_jobs** を使用して、コンテナーグループ内で起動するジョブの数を制限することが望ましい可能性があります。

14.6.5. ジョブのスケジューリングの設定

タスクマネージャーは、スケジュールする必要があるタスクを定期的に収集して、どのインスタンスに容量があり、タスクを実行可能かを判断します。タスクマネージャーのワークフローは、以下の通りです。

1. 制御インスタンスと実行インスタンスを見つけて割り当てます。
2. ジョブのステータスを更新して、待機中にします。
3. **pg_notify** を通じてコントロールノードにメッセージを送り、ディスパッチャーがタスクを取得して実行を開始できるようにします。

スケジューリングタスクが **TASK_MANAGER_TIMEOUT** 秒 (デフォルトは 300 秒) 以内に完了しない場合、タスクは早期に終了します。タイムアウトの問題は通常、保留中のジョブが多数存在する場合に発生します。

タスクマネージャーが 1 回の実行で処理できる作業量を制限する方法の 1 つに、**START_TASK_LIMIT** 設定があります。これにより、1 回の実行で開始できるジョブの数が制限されます。デフォルトは 100 ジョブです。さらに保留中のジョブがある場合は、新しいスケジューラータスクが直後に実行されるようにスケジュールされます。全体的なスループット向上のために、ジョブの起動から開始までのレイテンシーが潜在的に増大しても構わない場合は、**START_TASK_LIMIT** を引き上げることを検討できます。タスクマネージャーの個々の実行にかかる時間を確認するには、**/api/v2/metrics** で入手可能な Prometheus メトリクス **task_manager__schedule_seconds** を使用します。

タスクマネージャーによって実行の開始が選択されたジョブは、タスクマネージャープロセスが終了して変更がコミットされるまで実行されません。**TASK_MANAGER_TIMEOUT** 設定は、タスクマネージャーが変更をコミットするまでの 1 回の実行時間を決定します。タスクマネージャーは、タイムアウトに達すると進捗状況をコミットしようとします。タスクは、猶予期間 (**TASK_MANAGER_TIMEOUT_GRACE_PERIOD** により決定) が経過するまでは強制終了されません。

14.6.6. 内部クラスタールーティング

Automation Controller クラスターホストは、クラスター内のネットワークを介して通信します。従来の仮想マシンインストーラーのインベントリーファイルでは、クラスターノードへのルートを複数指定でき、それらはさまざまな方法で使用されます。

例:

```
[automationcontroller]
controller1 ansible_user=ec2-user ansible_host=10.10.12.11 node_type=hybrid
routable_hostname=somehost.somecompany.org
```

- **controller1** は、Automation Controller ホストのインベントリーホスト名です。インベントリーホスト名は、アプリケーションではインスタンスホスト名として表示されるものです。これは、バックアップ/復元方法を使用して異なる IP アドレスを持つ新しいホストのセットにクラスターを復元する、障害復旧シナリオに備える際に役立ちます。この場合、これらのインベントリーホスト名を IP アドレスにマッピングするエントリーを **/etc/hosts** に含めることができます。そして、内部 IP アドレスを使用して、パブリック DNS 名の解決に関する DNS の問題を軽減できます。

- **ansible_host=10.10.12.11** は、インストーラーがホスト (この場合は内部 IP アドレス) に到達する方法を示します。これはインストーラー以外では使用されません。

- **routable_hostname=somehost.somecompany.org** は、receptor メッシュ上でこのノードに接続するピアにとって解決可能なホスト名を示します。複数のネットワークをまたぐ可能性があるため、ホスト名は、receptor ピアで解決可能な IP アドレスにマッピングされるものを使用しています。

14.6.7. Web サーバーのチューニング

コントロールノードとハイブリッドノードはそれぞれ、Automation Controller の UI と API を提供します。WSGI トラフィックは、ローカルソケット上の uwsgi Web サーバーによって処理されます。ASGI トラフィックは、Daphne によって処理されます。NGINX はポート 443 でリッスンし、必要に応じてトラフィックをプロキシー処理します。

Automation Controller の Web サービスをスケーリングするには、次のベストプラクティスに従ってください。

- 複数のコントロールノードをデプロイし、ロードバランサーを使用して Web 要求を複数のサーバーに分散させます。

- Automation Controller ごとの最大接続数を 100 に設定します。

クライアント側で Automation Controller の Web サービスを最適化するには、次のガイドラインに従ってください。

- API を使用してインベントリーホストを個別に作成するのではなく、動的インベントリーソースを使用するようにユーザーに指示します。
- ジョブのステータスをポーリングする代わりに Webhook 通知を使用します。
- ホストの作成とジョブの起動に Bulk API を使用して、要求をバッチ処理します。
- トークン認証を使用します。多くの要求を迅速に行う必要がある自動化クライアントの場合、トークンを使用することがベストプラクティスです。これは、ユーザーのタイプによっては、Basic 認証を使用すると追加のオーバーヘッドが発生する可能性があるためです。

関連情報

- 高レベルの API インタラクションを伴うワークロードの詳細は、Scaling Automation Controller for API Driven Workloads を参照してください。
- Bulk API の詳細は、Bulk API in Automation Controller を参照してください。
- トークンの生成および使用方法の詳細は、トークンベースの認証 を参照してください。

第15章 シークレットの処理と接続セキュリティー

Automation Controller はシークレットと接続をセキュアに処理します。

15.1. シークレットの処理

Automation Controller は 3 つのシークレットのセットを管理します。

- ローカル Automation Controller ユーザーのユーザーパスワード。
- データベースのパスワードやメッセージバスのパスワードなど、Automation Controller の運用に使用するシークレット。
- SSH キー、クラウド認証情報、外部パスワード vault 認証情報など、自動化で使用するシークレット。

15.1.1. ローカルユーザー用のユーザーパスワード

Automation Controller は、SHA256 ハッシュを使用して、PBKDF2 アルゴリズムでローカル Automation Controller ユーザーのパスワードをハッシュします。LDAP、SAML、OAuth などの外部アカウントメカニズムで認証を行うユーザーの場合は、パスワードやシークレットが保存されません。

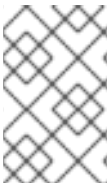
15.1.2. 運用に使用するシークレットの処理

Automation Controller には、運用に使用する以下のシークレットが存在します。

- **/etc/tower/SECRET_KEY**: データベース内の自動化シークレットを暗号化するために使用するシークレットキー。**SECRET_KEY** が変更された場合や不明な場合は、データベース内の暗号化されたフィールドにアクセスできません。
- **/etc/tower/tower.{cert,key}**: Automation Controller Web サービスの SSL 証明書とキー。自己署名証明書またはキーはデフォルトでインストールされますが、ローカルで適切な証明書とキーを指定できます。
- **/etc/tower/conf.d/postgres.py** にあるデータベースのパスワード、および **/etc/tower/conf.d/channels.py** にあるメッセージバスのパスワード

これらのシークレットは、暗号化されずに Automation Controller サーバーに保存されます。これは、Automation Controller サービスが起動時にこれらのシークレットをすべて自動的に読み取る必要があるためです。すべてのシークレットは UNIX パーミッションによって保護され、root および Automation Controller awx サービスユーザーに制限されます。

これらのシークレットを非表示にする必要がある場合、当該シークレットの読み取り元となるファイルは Python によって解釈されます。これらのファイルは、サービスが再起動するたびに、他のメカニズムでこれらのシークレットを取得するように調整できます。この変更はお客様により指定されるものであり、アップグレードのたびに再適用が必要な場合があります。Red Hat サポートと Red Hat コンサルティングは、そのような変更の例を有しています。



注記

シークレットシステムがダウンしていると、Automation Controller が情報を取得できず、サービスが復元されれば回復できるような障害が発生する可能性があります。システムで冗長性を使用することを強く推奨します。

Automation Controller によって生成された **SECRET_KEY** が侵害されており、再生成する必要があると思われる場合は、Automation Controller のバックアップおよび復元ツールとよく似た動作をするツールをインストーラーから実行できます。

重要

新しいシークレットキーを生成する前に、Automation Controller データベースを必ずバックアップしてください。

新しいシークレットキーを生成するには、以下を実行します。

1. バックアップおよび復元 セクションで説明されている手順に従います。
2. インストールのインベントリー (バックアップと復元の実行に使用したのと同じインベントリー) を使用して、次のコマンドを実行します。

```
setup.sh -k.
```

以前のキーのバックアップコピーは **/etc/tower/** に保存されます。

15.1.3. 自動化で使用するシークレットの処理

Automation Controller は、自動化に使用するシークレットや、自動化の結果であるさまざまなシークレットをデータベースに保存します。

これらのシークレットには以下が含まれます。

- すべての認証情報タイプの全シークレットフィールド (パスワード、シークレットキー、認証トークン、シークレットクラウド認証情報)。
- Automation Controller 設定で定義された外部サービスのシークレットトークンとパスワード。
- “password” タイプのサーベイフィールドのエントリー。

シークレットフィールドを暗号化するために、Automation Controller は、暗号化用の 256 ビットキーを使用した CBC モードの AES、PKCS7 パディング、および認証用の SHA256 を使用した HMAC を使用します。

暗号化や復号化のプロセスでは、**SECRET_KEY**、モデルフィールドのフィールド名、およびデータベースによって割り当てられた自動増分レコード ID から AES-256 ビット暗号化キーが導出されます。したがって、キー生成プロセスで使用される属性が変更された場合、Automation Controller はシークレットを正しく復号化できません。

Automation Controller は次のように設計されています。

- Automation Controller が起動する Playbook では、**SECRET_KEY** を読み取ることはできません。
- Automation Controller ユーザーは、これらのシークレットを読み取ることはできません。
- Automation Controller REST API によって、シークレットフィールド値が利用可能になることはありません。

Playbook でシークレット値が使用されている場合は、誤ってログに記録されないように、タスクで **no_log** を使用することを推奨します。

15.2. 接続セキュリティー

Automation Controller を使用すると、内部サービス、外部アクセス、および管理対象ノードへの接続が可能になります。

15.2.1. 内部サービス

Automation Controller は、内部操作の一環として次のサービスに接続します。

PostgreSQL データベース

PostgreSQL データベースへの接続は、ローカルホスト経由またはリモート (外部データベース経由) で、TCP を介したパスワード認証によって行われます。この接続では、PostgreSQL に組み込まれた SSL/TLS のサポートを使用できます。このサポートは、インストーラーのサポートによってネイティブに設定されます。SSL/TLS プロトコルは、デフォルトの OpenSSL 設定によって設定されます。

Redis キーまたは値のストア

Redis への接続は、ローカルの UNIX ソケットを介して行われ、awx サービスユーザーに制限されます。

15.2.2. 外部アクセス

Automation Controller は、Nginx が提供する標準ポート上の標準 HTTP/HTTPS を介してアクセスされます。自己署名証明書またはキーはデフォルトでインストールされますが、ローカルで適切な証明書とキーを指定できます。SSL/TLS アルゴリズムのサポートは、**/etc/nginx/nginx.conf** 設定ファイルで設定されます。デフォルトでは "intermediate" プロファイルが使用されますが、これは設定可能です。更新するたびに変更を再適用する必要があります。

15.2.3. 管理対象ノード

Automation Controller は、自動化の一環として管理対象マシンおよびサービスに接続します。管理対象マシンへの接続はすべて、SSH、WinRM、SSL/TLS など、標準のセキュアなメカニズムによって行われます。各メカニズムは、対象となる機能のシステム設定 (システム OpenSSL 設定など) から設定を継承します。

第16章 セキュリティーのベストプラクティス

Automation Controller をデプロイして、一般的な環境をセキュアに自動化できます。ただし、特定のオペレーティングシステム環境、自動化、および Automation Platform を管理するには、セキュリティーを確保するために追加のベストプラクティスが必要になる場合があります。

Red Hat Enterprise Linux をセキュリティー保護するには、以下の各リリースに適したセキュリティーガイドを参照してください。

- Red Hat Enterprise Linux 8 については、セキュリティーの強化 を参照してください。
- Red Hat Enterprise Linux 9 については、セキュリティーの強化 を参照してください。

16.1. ANSIBLE AUTOMATION PLATFORM と AUTOMATION CONTROLLER のアーキテクチャーについて

Ansible Automation Platform と Automation Controller は、汎用的で宣言型の自動化プラットフォームで構成されます。つまり、Ansible Playbook が (Automation Controller によって、またはコマンドラインで直接) 起動されると、Ansible に提供された Playbook、インベントリー、および認証情報は、信頼できるソースとみなされます。特定の Playbook コンテンツ、ジョブ定義、またはインベントリーコンテンツの外部検証に関するポリシーが必要な場合は、自動化を起動する前に、これらのプロセスを完了する必要があります。当該プロセスは、Automation Controller Web UI または Automation Controller API で行います。

ソースコントロールの使用、ブランチング、必須のコードレビューは、Ansible による自動化のベストプラクティスです。このようにソースコントロールを使用する際のプロセスフローを作成するために、役立つツールが存在します。

より高いレベルでは、自動化を含む任意のワークフローに関して、承認およびポリシーベースのアクションを作成できるツールが存在します。これらのツールは、Automation Controller の API 経由で Ansible を使用し、自動化を実行できます。

Automation Controller のインストール時には、セキュアなデフォルトの管理者パスワードを使用する必要があります。詳細は、Automation Controller 管理者パスワードの変更 を参照してください。

Automation Controller は、HTTP トラフィック用のポート 80 や HTTPS トラフィック用のポート 443 など、特定の既知のポートでサービスを公開します。オープンインターネットには Automation Controller を公開しないでください。オープンインターネットに公開しないことで、インストールの脅威が減少します。

16.1.1. アクセス権の付与

システムの特定の部分にアクセス権を付与すると、セキュリティーリスクが生じます。アクセスのセキュリティーを確保するために、次のプラクティスを適用してください。

- 管理アカウントを最小限に抑える
- ローカルシステムアクセスを最小限に抑える
- ユーザーから認証情報へのアクセスを削除する
- 職務の分離を強制する

16.1.2. 管理アカウントを最小限に抑える

システム管理アカウントへのアクセスを最小限に抑えることは、セキュアなシステムを維持するために非常に重要です。システム管理者または root ユーザーは、あらゆるシステムアプリケーションにアクセス、編集、および中断できます。可能な場合は、root アクセスを持つユーザーまたはアカウントの数を制限します。**root** や **awx** (Automation Controller ユーザー) への **sudo** を、信頼できないユーザーに付与しないでください。**sudo** などのメカニズムを使用して管理アクセスを制限した場合、特定のコマンドセットに制限しても、広範囲のアクセスが許可される可能性があることに注意してください。シェルまたは任意のシェルコマンドの実行を可能にするコマンド、またはシステム上のファイルを変更できるコマンドは、完全な root アクセスと同等です。

Automation Controller を使用すると、Automation Controller の "システム管理者" または "スーパーユーザー" アカウントは、Automation Controller のインベントリーまたは自動化定義を編集、変更、および更新できます。可能な限り、低レベルの Automation Controller 設定および障害復旧専用の最小限のユーザーだけに制限します。

16.1.3. ローカルシステムアクセスを最小限に抑える

ベストプラクティスに従って Automation Controller を使用する場合、管理目的以外のローカルユーザーアクセスは必要ありません。管理者以外のユーザーは、Automation Controller システムにアクセスできません。

16.1.4. 認証情報へのユーザーアクセスを削除する

Automation Controller 認証情報がコントローラーにのみ保存されている場合は、さらにセキュリティーを強化できます。OpenSSH などのサービスは、特定のアドレスからの接続時にのみ認証情報を許可するように設定できます。自動化で使用される認証情報は、システム管理者が障害復旧やその他のアドホック管理に使用する認証情報とは変えることができるため、監査が容易になります。

16.1.5. 職務の分離を強制する

自動化する部分に応じて、さまざまなレベルでシステムにアクセスすることが必要になる場合があります。たとえば、パッチを適用してセキュリティーベースラインチェックを実行する低レベルのシステム自動化を設定する一方で、アプリケーションをデプロイする高レベルの自動化を設定することが考えられます。自動化の各部分に異なるキーまたは認証情報を使用することにより、1 つのキーの脆弱性の影響が最小限に抑えられ、同時にベースライン監査も可能になります。

16.2. 利用可能なリソース

セキュアなプラットフォームを確保するために、Automation Controller などにいくつかのリソースが存在します。次の機能の使用を検討してください。

- 監査およびログの機能
- 既存のセキュリティー機能
- 外部アカウントストア
- Django パスワードポリシー

16.2.1. 監査およびログの機能

管理アクセスの場合、アクションの監査および監視を行うことが重要です。システム全体では、組み込みの監査サポートと組み込みのロギングサポートを使用してこれを実行できます。

Automation Controller の場合、Automation Controller 内のすべての変更を記録する組み込みのアクティビティーストリームサポートと、自動化ログを使用してこれを実行できます。

ベストプラクティスでは、ロギングと監査をローカルシステムで確認するのではなく、一元的に収集することが求められます。ご利用の環境で標準 ID またはロギングと監査 (Splunk) を使用するように、Automation Controller を設定する必要があります。Automation Controller には、Elastic Stack、Splunk、Sumologic、Loggly などの組み込みのロギング統合が含まれます。

関連情報

詳細は、ロギングおよびアグリゲーション を参照してください。

16.2.2. 既存のセキュリティー機能

SELinux または Automation Controller の既存のマルチテナントコンテインメントを無効にしないでください。Automation Controller のロールベースのアクセス制御 (RBAC) を使用して、自動化の実行に必要な最小限の権限を委譲します。Automation Controller のチームを使用して、個々のユーザーではなくユーザーのグループにパーミッションを割り当てます。

関連情報

詳細は、**Automation Controller ユーザーガイド** の ロールベースのアクセス制御 を参照してください。

16.2.3. 外部アカウントストア

Automation Controller ですべてのユーザーを維持することは、大規模な組織では時間のかかるタスクとなる場合があります。Automation Controller は、LDAP、SAML 2.0、および特定の OAuth プロバイダーによる外部アカウントソースへの接続をサポートします。これにより、パーミッションを操作する際のエラーの原因が排除されます。

16.2.4. Django パスワードポリシー

Automation Controller 管理者は、Django を使用して、作成時に **AUTH_PASSWORD_VALIDATORS** でパスワードポリシーを設定し、Automation Controller のユーザーパスワードを検証できます。Automation Controller インスタンスの **/etc/tower/conf.d** にある **custom.py** ファイルに、次のコードブロックの例を追加します。

```
AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
        'OPTIONS': {
            'min_length': 9,
        }
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]
```

関連情報

- 詳細は、前述の例に加えて、Django の Password validation を参照してください。
- 変更を有効にするには、必ず Automation Controller インスタンスを再起動してください。詳細は、Automation Controller の起動、停止、および再起動 を参照してください。

第17章 AWX-MANAGE ユーティリティー

awx-manage ユーティリティーは、Automation Controller の詳細な内部情報にアクセスするために使用します。**awx-manage** のコマンドは、**awx** ユーザーとしてのみ実行する必要があります。

17.1. インベントリーのインポート

awx-manage は、カスタムインベントリースクリプトを使用できないユーザー向けに、Automation Controller 管理者が Automation Controller にインベントリーを直接インポートできるメカニズムです。

awx-manage を正しく使用するには、まずインポート先として使用するインベントリーを Automation Controller に作成する必要があります。

awx-manage のヘルプが必要な場合は、次のコマンドを実行します。

```
awx-manage inventory_import [--help]
```

inventory_import コマンドは、コアの Ansible でサポートされている、テキストベースのインベントリーファイル、動的なインベントリースクリプト、またはファイルやスクリプトのディレクトリーと、Automation Controller のインベントリーオブジェクトを同期します。

コマンドの実行時に、**--inventory-id** または **--inventory-name**、および Ansible インベントリーのソースへのパス (**--source**) を指定します。

```
awx-manage inventory_import --source=/ansible/inventory/ --inventory-id=1
```

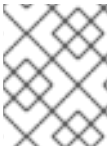
デフォルトでは、Automation Controller にすでに保存されているインベントリーデータと外部ソースからのデータは混在します。

外部データのみを使用するには、**--overwrite** を指定します。

既存のホストが **--source** からのみ変数データを取得するように指定するには、**--overwrite_vars** を指定します。

デフォルトの動作では、外部ソースから新しい変数を追加して既存のキーを上書きしますが、外部データソースから取得されていない変数は保持します。

```
awx-manage inventory_import --source=/ansible/inventory/ --inventory-id=1 --overwrite
```



注記

インベントリーホスト変数の編集と追加は、**--overwrite_vars** が設定されていない限り、インベントリーの同期後も維持されます。

17.2. 以前のデータの消去

awx-manage には、Automation Controller から古いデータを消去するためのさまざまなコマンドがあります。Automation Controller 管理者は、Automation Controller の **Management Jobs** インターフェイスを使用してアクセスするか、コマンドラインを使用できます。

```
awx-manage cleanup_jobs [--help]
```

このコマンドでは、ジョブの詳細や、指定の日数以前のジョブの出力が完全に削除されます。

```
awx-manage cleanup_activitystream [--help]
```

これにより、特定の日数以前の アクティビティーストリーム のデータが完全に削除されます。

17.3. クラスター管理

awx-manage provision_instance および **awx-manage deprovision_instance** コマンドの詳細は、クラスタリング を参照してください。

注記

Ansible サポートからの指示がない限り、他の **awx-manage** コマンドは実行しないようにしてください。

17.4. トークンおよびセッション管理

Automation Controller は、OAuth2 トークン管理用に次のコマンドをサポートします。

- **create_oauth2_token**
- **revoke_oauth2_tokens**
- **cleartokens**
- **expire_sessions**
- **clearsessions**

17.4.1. create_oauth2_token

次のコマンドを使用して OAuth2 トークンを作成します (**example_user** にはユーザー名を指定します)。

```
$ awx-manage create_oauth2_token --user example_user

New OAuth2 token for example_user: j89ia8OO79te6IAZ97L7E8bMgXCON2
```

トークンを作成するときは、必ず有効なユーザーを指定してください。そうしなければ、ユーザーを指定せずにコマンドを発行しようとしたことを示すエラーメッセージか、存在しないユーザー名を指定したことを示すエラーメッセージが表示されます。

17.4.2. revoke_oauth2_tokens

このコマンドを使用して、OAuth2 トークン (アプリケーショントークンとパーソナルアクセストークン (PAT) の両方) を取り消します。すべてのアプリケーショントークン (関連する更新トークンは除く) が取り消され、すべてのパーソナルアクセストークンも取り消されます。ただし、すべてのトークンを取り消すユーザーを指定することもできます。

既存の OAuth2 トークンをすべて取り消すには、次のコマンドを使用します。

```
$ awx-manage revoke_oauth2_tokens
```

すべての OAuth2 トークンとその更新トークンを取り消すには、次のコマンドを使用します。

```
$ awx-manage revoke_oauth2_tokens --revoke_refresh
```

id=example_user が指定されたユーザーの OAuth2 トークンをすべて取り消すには、以下を実行します (**example_user** にユーザー名を指定します)。

```
$ awx-manage revoke_oauth2_tokens --user example_user
```

id=example_user が指定されたユーザーの全 OAuth2 トークンと更新トークンを取り消すには、以下を実行します。

```
$ awx-manage revoke_oauth2_tokens --user example_user --revoke_refresh
```

17.4.3. cleartokens

このコマンドを使用して、すでに取り消されたトークンを消去します。

詳細は、Django の Oauth Toolkit ドキュメントの cleartokens を参照してください。

17.4.4. expire_sessions

このコマンドを使用して、すべてのセッションまたは特定のユーザーのすべてのセッションを終了します。

組織内でユーザーのロールが変更になった場合や、LDAP/AD の分類グループから削除された場合、これらのグループのメンバーシップが原因で管理者が対象ユーザーにジョブを実行させない場合などに、このコマンドの使用を検討してください。

```
$ awx-manage expire_sessions
```

このコマンドはデフォルトですべてのセッションを終了します。これらのセッションに関連付けられているユーザーはログアウトされます。特定のユーザーのセッションのみを期限切れにするには、**--user** フラグを使用してユーザー名を渡します (次の例では、**example_user** をユーザー名に置き換えます)。

```
$ awx-manage expire_sessions --user example_user
```

17.4.5. clearsessions

このコマンドを使用して、期限切れになったすべてのセッションを削除します。

詳細は、Django の Oauth Toolkit ドキュメントの Clearing the session store を参照してください。

UI での OAuth2 トークン管理の詳細は、「Automation Controller ユーザーガイド」の アプリケーション セクションを参照してください。

17.5. アナリティクスの収集

以下のコマンドを使用して、事前定義の時間枠 (デフォルトは 4 時間) 以外で、オンデマンドでアナリティクスを収集します。

```
$ awx-manage gather_analytics --ship
```

切断された環境を使用し、一定期間にわたって、自動化された一意のホストに関する使用情報を収集する場合は、次のコマンドを使用します。

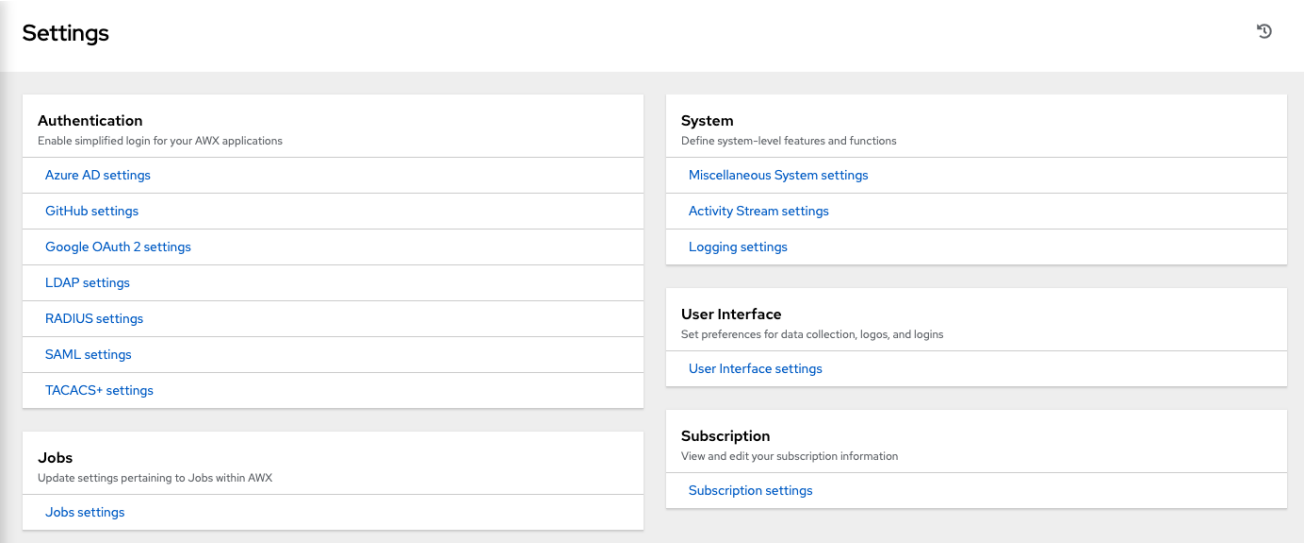
```
awx-manage host_metric --since YYYY-MM-DD --until YYYY-MM-DD --json
```

パラメーター **--since** および **--until** は、日付範囲を指定します。指定は任意ですが、いずれか 1 つは指定する必要があります。

--json フラグは出力形式を指定するもので、指定は任意です。

第18章 AUTOMATION CONTROLLER の設定

次のタブの **Settings** 画面で、Automation Controller の設定を行うことができます。



各タブには **Reset** オプションを備えたフィールドが含まれており、入力した値をデフォルト値に戻すことができます。**Reset All** は、すべての値を出荷時のデフォルト値に戻すことができます。

Save すると変更が適用されますが、編集ダイアログは終了しません。**Settings** ページに戻るには、ナビゲーションパネルから **Settings** を選択するか、現在のビューの上部にあるブレッドクラムを使用します。

18.1. AUTOMATION CONTROLLER の認証

Automation Controller UI を使用すると、GitHub、Google、LDAP、RADIUS、SAML などのさまざまな認証タイプによる簡素化されたログインをセットアップできます。適切なサービスで開発者アプリケーションを作成および登録した後に、それらに対する認証をセットアップできます。

手順

1. ナビゲーションパネルから **Settings** を選択します。
2. 次の **Authentication** オプションから選択します。
    - Azure AD の設定
    - Github の設定
    - Google OAuth2 の設定
    - LDAP 認証
    - RADIUS 設定
    - SAML 設定
        - 透過的な SAML ログイン
        - SAML のロギングを有効にする
    - TACACS+ 設定

- Generic OIDC 設定
  必要な情報がすべて含まれていることを確認してください。

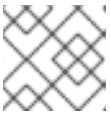
3. **Save** をクリックして設定を適用するか、**Cancel** をクリックして変更を破棄します。

18.2. ジョブの設定

Jobs タブでは、Automation Controller のアドホックコマンド機能で使用可能なモジュールタイプの設定、スケジュール可能なジョブ数に対する制限の設定、出力サイズの定義、および Automation Controller のジョブとの連携に関するその他の詳細の定義を行うことができます。

手順

1. ナビゲーションパネルから **Settings** を選択します。
2. **Jobs** オプションで **Jobs settings** を選択します。表示されたフィールドで設定可能なオプションを設定します。追加情報が必要な場合、対象のフィールドの横にあるツールチップアイコン をクリックします。
   Galaxy 設定の詳細は、**Automation Controller ユーザーガイド** の Ansible Galaxy サポート セクションを参照してください。



注記

すべてのタイムアウト値は秒単位です。

3. 設定を適用するには **Save** を、変更を破棄するには **Cancel** をクリックします。

18.3. システム設定の設定

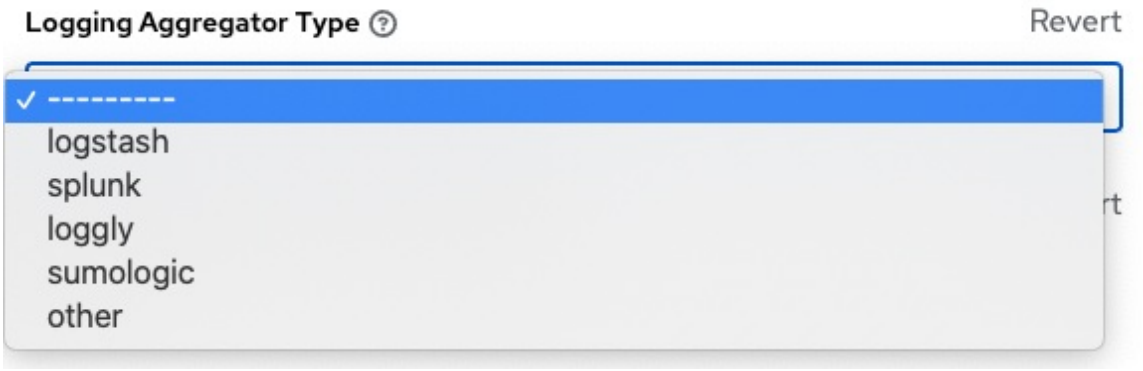
System タブでは、次のアクションを実行できます。

- Automation Controller ホストのベース URL の定義
- アラートの設定
- アクティビティーキャプチャーの有効化
- ユーザーの可視性の制御
- ライセンスファイルによる特定の Automation Controller 機能の有効化
- ロギングアグリゲーションオプションの設定

手順

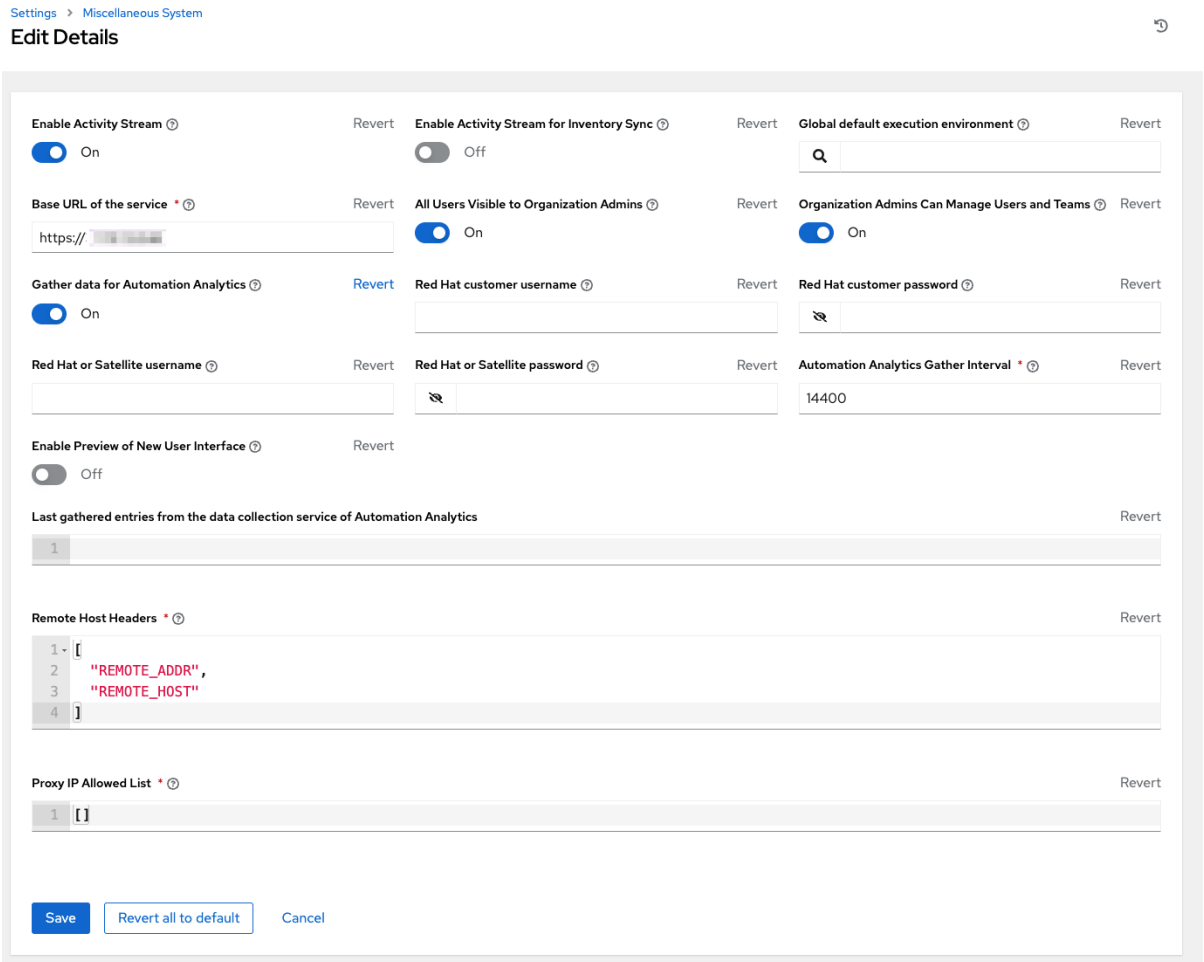
1. ナビゲーションパネルから **Settings** を選択します。
2. 次の **System** オプションから選択します。
   - **Miscellaneous System settings**: アクティビティーストリームの有効化、デフォルトの実行環境の指定、Automation Controller ホストのベース URL の定義、Automation Controller 管理アラートの有効化、ユーザーの可視性の設定、アナリティクスの定義、ユーザー名とパスワードの指定、およびプロキシーの設定を行います。

- **Miscellaneous Authentication settings**: 認証方法 (ビルトインまたは SSO)、セッション (タイムアウト、ログインしたセッション数、トークン)、およびソーシャル認証マッピングに関連するオプションを設定します。
- **Logging settings**: 選択したタイプをもとにロギングオプションを設定します。



各ロギングアグリゲーションタイプの詳細は、ロギングおよびアグリゲーション セクションを参照してください。

3. 表示されたフィールドで設定可能なオプションを設定します。追加情報が必要な場合、対象のフィールドの横にあるツールチップアイコン ⓘ をクリックします。
以下は **Miscellaneous System** の設定例です。



注記

Allow External Users to Create Oauth2 Tokens設定は、デフォルトでは無効になっています。これにより、外部ユーザーは独自のトークンを作成できなくなります。有効にしてから無効にすると、その間に外部ユーザーが作成したトークンは残り、自動的には取り消されません。

4. 設定を適用するには **Save** を、変更を破棄するには **Cancel** をクリックします。

18.4. ユーザーインターフェイスの設定

User Interface タブでは、Automation Controller の分析設定や、カスタムロゴやログインメッセージの設定が可能です。

手順

1. ナビゲーションパネルから **Settings** を選択します。
2. **User Interface** オプションから **User Interface settings** を選択します。
3. **Edit** をクリックして個人設定を設定します。

18.4.1. ユーザビリティーアナリティクスおよびデータ収集の設定

Automation Controller には、ユーザビリティーに関するデータ収集が含まれています。これにより、ユーザーがどのように Automation Controller と対話するかを理解し、今後のリリースを強化して、ユーザーエクスペリエンスを合理化するためのデータを収集します。

Red Hat Ansible Automation Platform の試用版をインストールしたユーザー、または Automation Controller を新規にインストールしたユーザーのみが、このデータ収集にオプトインされます。

Automation Controller は、製品の改善に役立てるためにユーザーデータを自動的に収集します。**User Interface settings** で参加レベルを設定することで、Automation Controller によるデータ収集のオプトアウトや制御が可能です。

手順

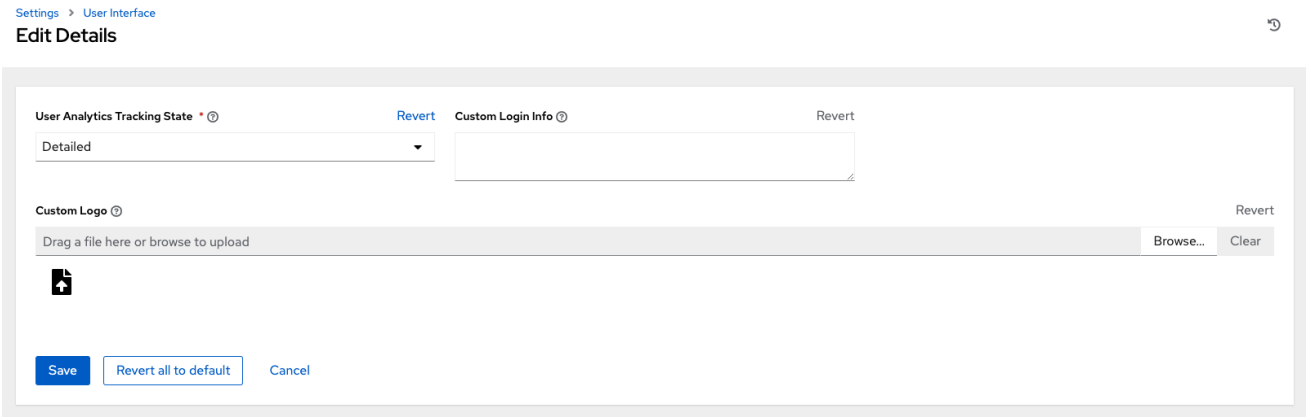
1. ナビゲーションパネルから **Settings** を選択します。
2. **User Interface** オプションから **User Interface settings** を選択します。
3. **Edit** をクリックします。
4. **User Analytics Tracking State** リストから目的のデータ収集レベルを選択します。
   - **Off**: データ収集を行いません。
   - **Anonymous**: ユーザー固有のデータを含めないデータ収集を有効化します。
   - **Detailed**: お使いのユーザー固有のデータを含めたデータ収集を有効化します。
5. **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。

関連情報

詳細は、Red Hat のプライバシーに関する声明 を参照してください。

18.4.2. カスタムロゴおよびイメージ

Automation Controller は、カスタムロゴの使用をサポートしています。カスタムロゴを追加するには、イメージをアップロードし、**User Interface settings** ページからカスタムログインメッセージを指定します。これらの設定にアクセスするには、ナビゲーションパネルから **Settings** を選択します。

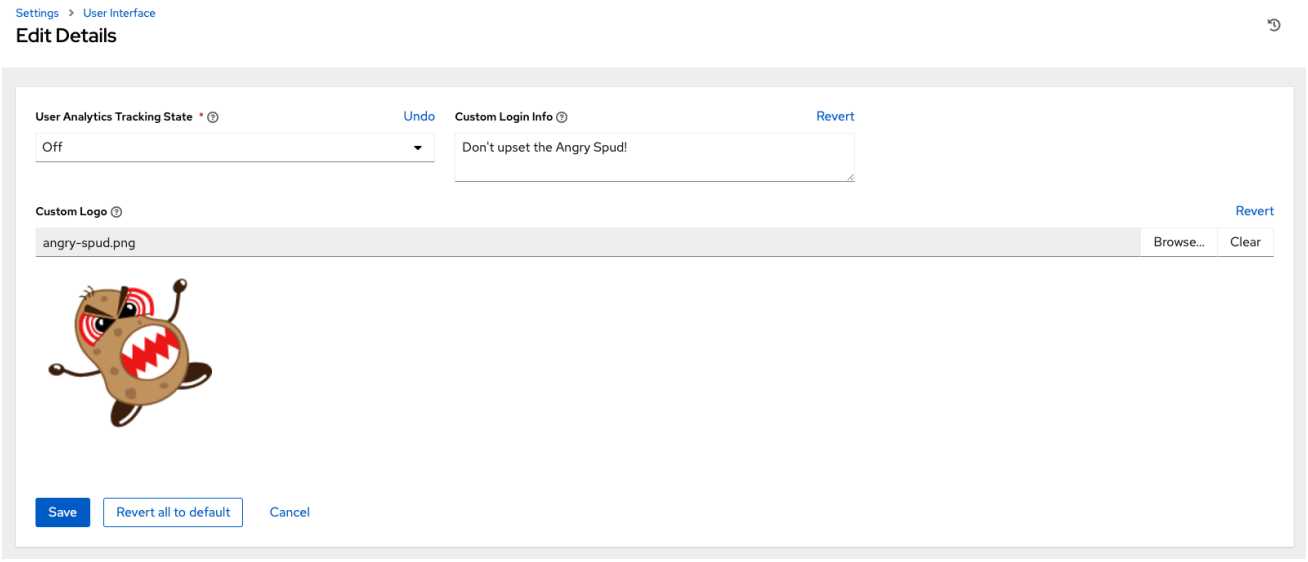


最良の結果を得るには、背景が透明な **.png** ファイルを使用します。GIF、PNG、JPEG 形式がサポートされています。

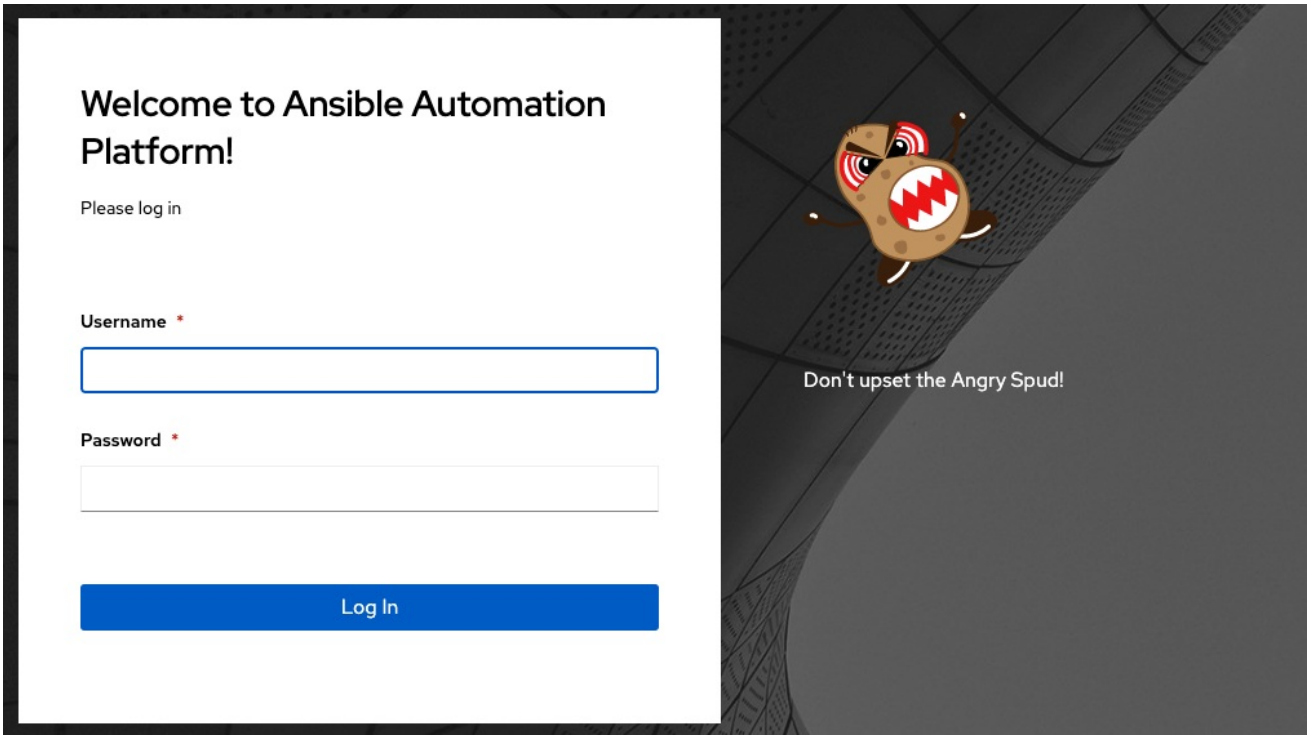
特定の情報 (法律上の通知や免責事項など) を **Custom Login Info** テキストフィールドに追加することで、ログインモーダルのテキストボックスに追加できます。

例

特定のロゴをアップロードし、次のテキストを追加したとします。



Ansible Automation Platform のログインダイアログは次のようになります。



標準の Automation Controller ロゴを使用するには、**Revert** を選択します。

18.5. AUTOMATION CONTROLLER の追加設定

Automation Controller の動作に影響を与える可能性のある追加の詳細設定が存在します。当該設定を Automation Controller UI で行うことはできません。

従来の仮想マシンベースのデプロイメントの場合、**/etc/tower/conf.d/custom.py** にファイルを作成することで、これらの設定を Automation Controller に提供できます。ファイルベースの設定を通じて Automation Controller に設定を提供する場合、設定ファイルはすべてのコントロールプレーンノードに存在する必要があります。これらのノードは、インストーラーインベントリーの **automationcontroller** グループに属するハイブリッドまたはコントロールタイプのノードすべてを含みます。

これらの設定を有効にするには、設定ファイルを有する各ノードで、**automation-controller-service** restart を使用してサービスを再起動します。このファイルで指定した設定が Automation Controller UI にも表示される場合、UI では "読み取り専用" としてマークされます。

18.6. 正式な ANSIBLE AUTOMATION CONTROLLER のサブスクリプションの取得

すでに Red Hat 製品のサブスクリプションをお持ちの場合は、そのサブスクリプションを通じて Automation Controller サブスクリプションを取得できます。Red Hat Ansible Automation Platform および Red Hat Satellite のサブスクリプションをお持ちでない場合は、試用版サブスクリプションをリクエストできます。

手順

- Red Hat Ansible Automation Platform サブスクリプションをお持ちの場合は、Automation Controller を起動するときに Red Hat の顧客認証情報を使用して、サブスクリプション情報にアクセスします。サブスクリプションのインポート を参照してください。
- Ansible 以外の Red Hat サブスクリプションまたは Satellite サブスクリプションをお持ちの場合は、次のいずれかの方法で Automation Controller にアクセスします。

- ライセンスページでユーザー名とパスワードを入力します。
- Red Hat カスタマーポータルの Subscription Allocations ページからサブスクリプションマニフェストを取得します。詳細は、**Automation Controller ユーザーガイド** の サブスクリプションマニフェストの取得 を参照してください。
- Red Hat Ansible Automation Platform のサブスクリプションをお持ちでない場合は、Try Red Hat Ansible Automation Platform に移動し、試用版サブスクリプションをリクエストしてください。

関連情報

サブスクリプションでサポートされる内容を確認するには、Automation Controller のライセンス、更新、およびサポート を参照してください。* サブスクリプションに問題がある場合は、セールスアカウントマネージャーまたは Red Hat カスタマーサービス (https://access.redhat.com/support/contact/customerService/) にお問い合わせください。

18.6.1. トラブルシューティング: サブスクリプションのコンプライアンスの維持

サブスクリプションには、次の 2 つのステータスがあります。

- **Compliant**: サブスクリプションが、サブスクリプション数以内の自動化したホストの数に対して適切であることを示します。
- **Out of compliance**: 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。

コンプライアンスは次のように計算されます。

```
managed > manifest_limit   => non-compliant
managed =< manifest_limit  => compliant
```

ここで、**managed** は、削除されていない一意のマネージドホストの数、**manifest_limit** はサブスクリプションマニフェスト内のマネージドホストの数に置き換えます。

表示されるその他の重要な情報は次のとおりです。

- **Hosts automated**: ライセンス数を消費する、ジョブで自動化されたホスト数。
- **Hosts imported**: インベントリーソースすべての一意のホスト名を考慮したホスト数。この数は、残りのホスト数 (Hosts remaining) には影響しません。
- **Hosts remaining**: 合計ホスト数から自動化されたホストを差し引いた数。
- **Hosts deleted**: 削除されたホスト (ライセンスの容量を解放します)。
- **Active hosts previously deleted**: 以前に削除され、現在アクティブになっているホストの数。

たとえば、ホスト 10 台分のサブスクリプションがあるとします。

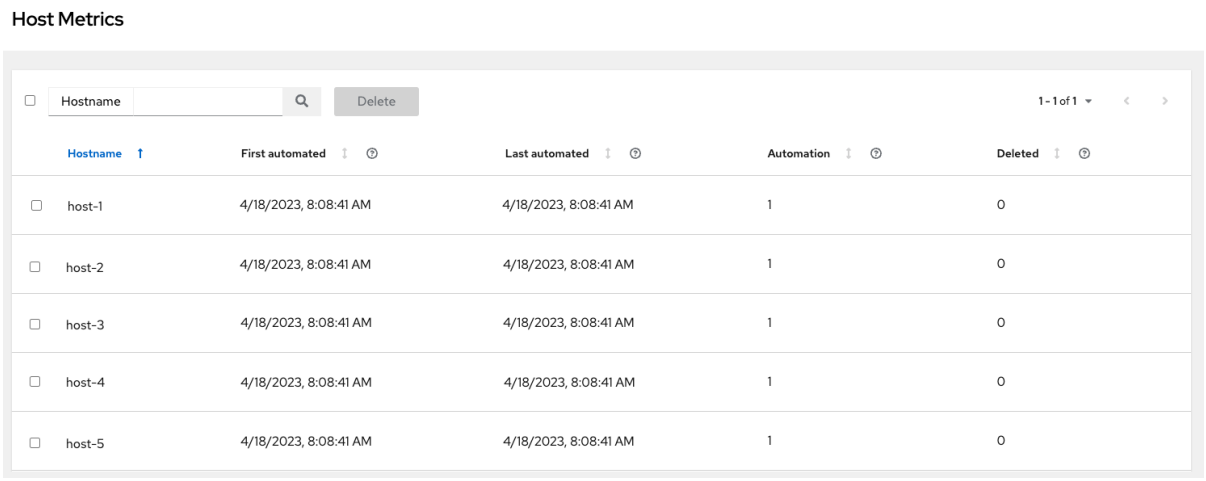
- 最初は 9 台のホストでしたが、ホストを 2 台追加してから、3 台削除しました。その結果、ホストが 8 台になりました (コンプライアンス準拠)。
- 3 台のホストを再び自動化したところ、ホストが 11 台になり、サブスクリプションの上限である 10 を超えました (コンプライアンス非準拠)。

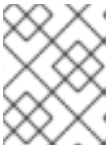
- ホストを削除する場合は、サブスクリプションの詳細を更新して、数とステータスの変更を確認します。

18.6.2. ホストアクティビティーの表示

手順

1. ナビゲーションパネルで **Host Metrics** を選択して、自動化されたものや削除されたものなど、ホストに関連するアクティビティーを表示します。
それぞれの一意のホスト名がリストされ、ユーザーの設定に応じて並べ替えられます。





注記

スケジュールされたタスクはこれらの値を毎週自動的に更新し、最後に自動化されたのが 1 年以上前であるホストのジョブを削除します。

2. 不要なホストを Host Metrics ビューから直接削除するには、目的のホストを選択して **Delete** をクリックします。
これらは論理的に削除されるため、レコードは削除されませんが、使用されないため、サブスクリプションにはカウントされません。

詳細は、**Automation Controller ユーザーガイド** の トラブルシューティング: サブスクリプションのコンプライアンスの維持 を参照してください。

18.6.3. ホストメトリクスユーティリティー

Automation Controller は、コマンドラインインターフェイス (CLI) を通じて、ホストメトリクスデータとホストメトリクス概要の CSV 出力を生成する方法を提供します。API を介してホストを一括して論理削除することもできます。

詳細は、**Automation controller User Guide** の Host metrics utilities セクションを参照してください。

第19章 分離機能および変数

Automation Controller は、コンテナーテクノロジーを使用してジョブを相互に分離します。デフォルトでは、現在のプロジェクトのみがジョブテンプレートを実行するコンテナーに公開されます。

追加のディレクトリーを公開するには、Playbook の実行のカスタマイズが必要な場合があります。

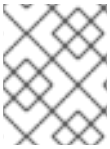
ジョブ分離の使用を微調整するために、特定の変数が設定可能です。

デフォルトでは、Automation Controller はシステムの **/tmp** ディレクトリーをステージングエリアとして使用します。これは、**Jobs settings** ページの **Job Execution Path** フィールド、または **/api/v2/settings/jobs** の REST API で以下を使用することで変更できます。

```
AWX_ISOLATION_BASE_PATH = "/opt/tmp"
```

ホストから Playbook を実行するコンテナーに追加のディレクトリーを公開する場合は、**Jobs settings** ページの **Paths to Expose to Isolated Jobs**フィールド、または **/api/v2/settings/jobs** の REST API で以下を使用することで指定できます。

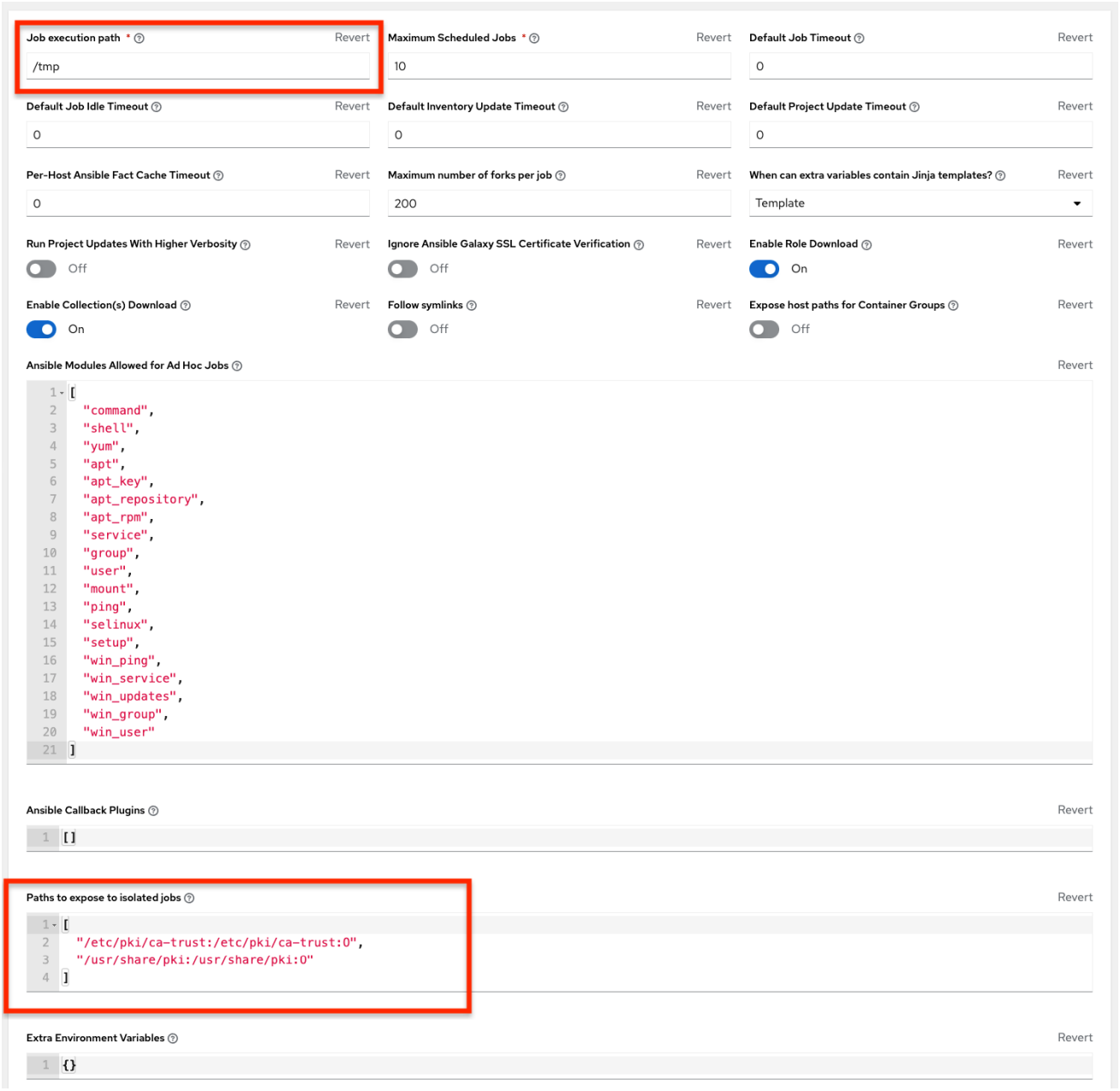
```
AWX_ISOLATION_SHOW_PATHS = ['/list/of/', '/paths']
```



注記

Playbook が **/var/lib/awx/.ssh** で定義されたキーまたは設定を使用する場合は、それを **AWX_ISOLATION_SHOW_PATHS** に追加する必要があります。

これらのフィールドは、**Jobs Settings** ページにあります。


Job execution path * Revert
/tmp
Maximum Scheduled Jobs * Revert
10
Default Job Timeout Revert
0
Default Job Idle Timeout Revert
0
Default Inventory Update Timeout Revert
0
Default Project Update Timeout Revert
0
Per-Host Ansible Fact Cache Timeout Revert
0
Maximum number of forks per job Revert
200
When can extra variables contain Jinja templates? Revert
Template
Run Project Updates With Higher Verbosity Revert
Off
Ignore Ansible Galaxy SSL Certificate Verification Revert
Off
Enable Role Download Revert
On
Enable Collection(s) Download Revert
On
Follow symlinks Revert
Off
Expose host paths for Container Groups Revert
Off
Ansible Modules Allowed for Ad Hoc Jobs Revert
[
"command",
"shell",
"yum",
"apt",
"apt_key",
"apt_repository",
"apt_rpm",
"service",
"group",
"user",
"mount",
"ping",
"selinux",
"setup",
"win_ping",
"win_service",
"win_updates",
"win_group",
"win_user"
]
Ansible Callback Plugins Revert
[]
Paths to expose to isolated jobs Revert
[
"/etc/pki/ca-trust:/etc/pki/ca-trust:O",
"/usr/share/pki:/usr/share/pki:O"
]
Extra Environment Variables Revert
{}

第20章 トークンベースの認証

OAuth 2 は、トークンベースの認証に使用されます。OAuth トークンとアプリケーション、つまりトークンの生成に使用する API クライアントのサーバー側の表現を管理できます。OAuth トークンを HTTP 認証ヘッダーの一部として含めることにより、自身を認証し、基本の RBAC パーミッションに加えて制限的なパーミッションのレベルを調整できます。

OAuth2 仕様の詳細は、The OAuth 2.0 Authorization Framework を参照してください。

manage ユーティリティーを使用してトークンを作成する方法の詳細は、トークンおよびセッション管理 を参照してください。

20.1. OAUTH 2 アプリケーションおよびトークンの管理

アプリケーションとトークンは、**/api/<version>/applications** および **/api/<version>/tokens** でトップレベルのリソースとして管理できます。これらのリソースには、**/api/<version>/users/N/<resource>** でユーザーごとにアクセスすることもできます。アプリケーションを作成するには、**api/<version>/applications** または **/api/<version>/users/N/applications** に **POST** を実行します。

各 OAuth 2 アプリケーションは、サーバー側で特定の API クライアントを表現します。API クライアントがアプリケーショントークンを介して API を使用するには、まずアプリケーションを用意し、アクセストークンを発行する必要があります。個々のアプリケーションには、**/api/<version>/applications/<pk>/** にあるプライマリーキーを使用してアクセスできます。

以下は、一般的なアプリケーションです。


```
{
    "id": 1,
    "type": "o_auth2_application",
    "url": "/api/v2/applications/2/",
    "related": {
        "tokens": "/api/v2/applications/2/tokens/"
    },
    "summary_fields": {
        "organization": {
            "id": 1,
            "name": "Default",
            "description": ""
        },
        "user_capabilities": {
            "edit": true,
            "delete": true
        },
        "tokens": {
            "count": 0,
            "results": []
        }
    },
    "created": "2018-07-02T21:16:45.824400Z",
    "modified": "2018-07-02T21:16:45.824514Z",
    "name": "My Application",
    "description": "",
    "client_id": "Ecmc6RjjhKUOWJzDYEP8TZ35P3dvsKt0AKdIjgHV",
    "client_secret":
```

```
"7Ft7ym8MpE54yWGUNvxxg6KqGwPFsyhYn9QQfYHlgBxai74Qp1GE4zsvJduOfSFkTfWFnPzYpxqcRsy1KacD0HH0vOAQUDJDCidByMiUIH4YQKtGFM1zE1dACYbpN44E",
    "client_type": "confidential",
    "redirect_uris": "",
    "authorization_grant_type": "password",
    "skip_authorization": false,
    "organization": 1
}
```


ここで、**name** は、人間が判読できるアプリケーションの識別子です。**client_id** や **redirect_uris** などの残りのフィールドは、主に OAuth2 認可に使用されます。この点は、パーソナルアクセストークン (PAT) 向けの OAuth2 トークンシステムの使用 で説明されています。

client_id および **client_secret** フィールドの値は作成時に生成されるアプリケーション識別子で、編集できません。一方で、**organization** および **authorization_grant_type** は作成時に必要で、その後に編集不可になります。

20.1.1. アプリケーションのアクセスルール

アプリケーションのアクセスルールは以下のとおりです。

- システム管理者は、システム内のすべてのアプリケーションを表示し、操作できます。
- 組織の管理者は、組織メンバーに属するすべてのアプリケーションを表示し、操作できます。
- 他のユーザーは、自身のアプリケーションの表示、更新、削除が可能ですが、新規アプリケーションは作成できません。

一方、トークンは、受信要求を認証し、基盤のユーザーのパーミッションをマスクするために使用するリソースです。

トークンを作成するには 2 つの方法があります。

- **/api/v2/tokens/** エンドポイントに POST 要求を送信します。**application** フィールドと **scope** フィールドは、関連アプリケーションを参照してトークンスコープを指定するように設定します。
- **scope** フィールドを使用して **/api/v2/applications/<pk>/tokens/** エンドポイントに POST 要求を送信します (親アプリケーションは自動的にリンクされます)。

個々のトークンは、**に/api/<version>/tokens/<pk>/** にあるプライマリーキーを使用してアクセスできます。

以下は一般的なトークンの例です。


```
{
    "id": 4,
    "type": "o_auth2_access_token",
    "url": "/api/v2/tokens/4/",
    "related": {
        "user": "/api/v2/users/1/",
        "application": "/api/v2/applications/1/",
        "activity_stream": "/api/v2/tokens/4/activity_stream/"
},
    "summary_fields": {
        "application": {
```

```
            "id": 1,
            "name": "Default application for root",
            "client_id": "mcU5J5uGQcEQMgAZyr5JUnM3BqBJpgbgL9fLOVch"
        },
        "user": {
            "id": 1,
            "username": "root",
            "first_name": "",
            "last_name": ""
        }
    },
    "created": "2018-02-23T14:39:32.618932Z",
    "modified": "2018-02-23T14:39:32.643626Z",
    "description": "App Token Test",
    "user": 1,
    "token": "**********",
    "refresh_token": "**********",
    "application": 1,
    "expires": "2018-02-24T00:39:32.618279Z",
    "scope": "read"
},
```


OAuth 2 トークンの場合、完全に編集可能なフィールドは、**scope** と **description** のみです。**application** フィールドは更新時に編集できず、他のフィールドはすべて編集できません。これらは作成時に自動で次のように入力されます。

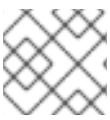
- **user** フィールドは、トークンの作成対象のユーザーに対応します。この場合、トークンを作成するユーザーでもあります。
- **expires** は、Automation Controller の設定 **OAUTH2_PROVIDER** に従って生成されます。
- **token** および **refresh_token** は、クラッシュなしの乱数文字列として自動生成されます。

アプリケーショントークンとパーソナルアクセストークンは両方とも **/api/v2/tokens/** エンドポイントに表示されます。パーソナルアクセストークンの **application** フィールドは常に **null** です。これにより、2 種類のトークンを区別できます。

20.1.2. トークンのアクセスルール

トークンのアクセスルールは以下のとおりです。

- ユーザーは、関連のアプリケーションを表示できる場合にはトークンを作成でき、自身の個人トークンも作成可能です。
- システム管理者は、システム内のすべてのトークンを表示し、操作できます。
- 組織の管理者は、組織メンバーに属するすべてのトークンを表示し、操作できます。
- システム監査者は、すべてのトークンおよびアプリケーションの表示が可能です。
- 他の通常ユーザーは、自身のトークンのみ表示し、操作できます。



注記

ユーザーは、作成時にのみトークンの表示またはトークンの値の更新が可能です。

20.2. パーソナルアクセストークン向けの OAUTH2 トークンシステムの使用

OAuth 2 トークンを取得する最も簡単で一般的な方法は、次の例に示すように、**/api/v2/users/<userid>/personal_tokens/** エンドポイントでパーソナルアクセストークン (PAT) を作成することです。

```
curl -XPOST -k -H "Content-type: application/json" -d '{"description":"Personal controller CLI token", "application":null, "scope":"write"}' https://<USERNAME>:<PASSWORD>@<CONTROLLER_SERVER>/api/v2/users/<USER_ID>/personal_tokens/ | python -m json.tool
```

インストールされている場合は、**jq** から JSON 出力をパイプすることもできます。

以下は、curl を使用して API エンドポイントにアクセスするために PAT を使用する例です。

```
curl -k -H "Authorization: Bearer <token>" -H "Content-Type: application/json" -X POST  -d '{}' https://controller/api/v2/job_templates/5/launch/
```

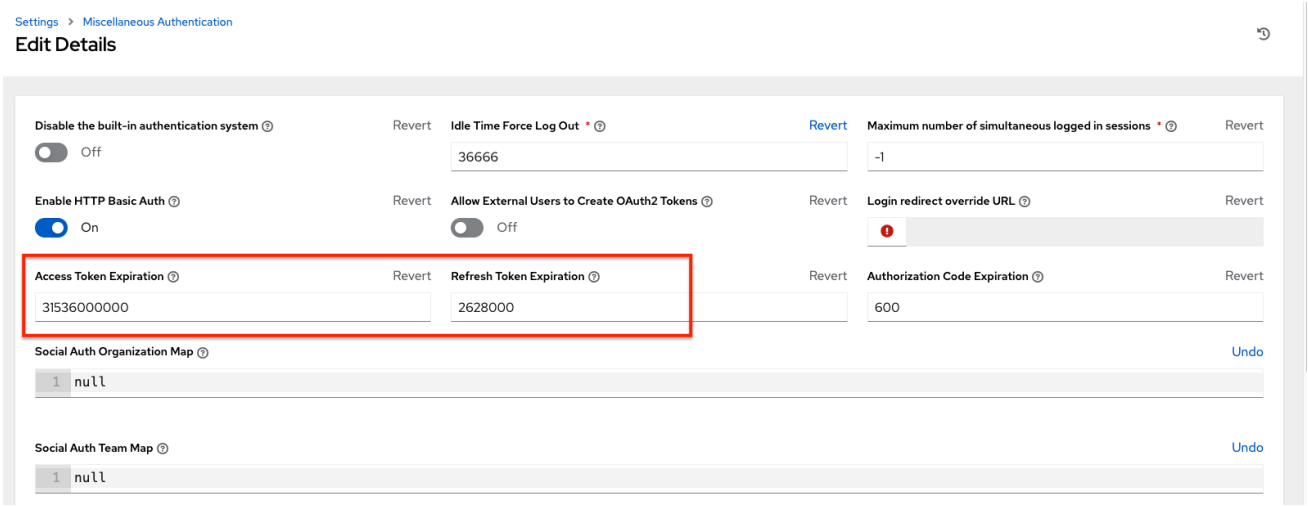
Automation Controller では、OAuth 2 システムは Django Oauth Toolkit の上にビルドされ、トークンの承認、取り消し、および更新のための専用エンドポイントを提供します。

これらのエンドポイントは **/api/v2/users/<USER_ID>/personal_tokens/** エンドポイントにあります。このエンドポイントでは、これらのエンドポイントの一般的な使用方法の例も示されています。これらの特別な OAuth 2 エンドポイントは、**x-www-form-urlencoded** という **Content-type** の使用のみをサポートするため、**api/o/*** エンドポイントで **application/json** を受け入れるものはありません。

注記

アプリケーションタイプに **null** を指定することで、**/api/o/token** エンドポイントを使用してトークンを要求することもできます。

あるいは、UI を使用してユーザーのトークンを追加する方法や、アクセストークンとそれに関連付けられたリフレッシュトークンの有効期限を設定する方法 (該当する場合) については、トークンの追加 を参照してください。



20.2.1. RBAC システム全体のトークンスコープのマスク

OAuth 2 トークンのスコープは、有効なスコープキーワード "read" と "write" で構成されるスペース区

切りの文字列です。これらのキーワードは設定可能であり、認証された API クライアントのパーミッションレベルを指定するのに使用されます。read および write スコープは、Automation Controller の **ロールベースのアクセス制御** (RBAC) パーミッションシステム上にマスク層を提供します。"write" スコープは、RBAC システムが提供する完全なパーミッションを認証ユーザーに付与します。一方 "read" スコープは、RBAC システムが提供する読み取りパーミッションのみを認証ユーザーに付与します。"write" パーミッションには "read" パーミッションも含まれることに注意してください。

たとえば、ジョブテンプレートに対する管理パーミッションがある場合、セッションまたは Basic 認証で認証されていれば、ジョブテンプレートを表示、変更、起動、および削除できます。

対照的に、OAuth 2 トークンで認証されており、関連するトークンスコープが "read" である場合は、管理者であってもジョブテンプレートの表示のみが可能で、操作や起動はできません。

トークンスコープが "write" または "read write" の場合、管理者としてジョブテンプレートを最大限に活用できます。

トークンを取得して使用するには、まずアプリケーショントークンを作成する必要があります。

手順

1. **authorization_grant_type** を **password** に設定してアプリケーションを作成します。
2. **/api/v2/applications/** エンドポイントに、次の HTTP POST 要求を送信します (自身の組織 ID を指定します)。

```
{
    "name": "Admin Internal Application",
    "description": "For use by secure services & clients. ",
    "client_type": "confidential",
    "redirect_uris": "",
    "authorization_grant_type": "password",
    "skip_authorization": false,
    "organization": <organization-id>
}
```

3. トークンを作成し、以下を使用して POST 要求を **/api/v2/tokens/** エンドポイントに送信します。

```
{
    "description": "My Access Token",
    "application": <application-id>,
    "scope": "write"
}
```

これにより、今後の要求の認証に使用できる <token-value> が返されます (これは今後表示されません)。

4. トークンを使用してリソースにアクセスします。以下では例として curl を使用しています。

```
curl -H "Authorization: Bearer <token-value>" -H "Content-Type: application/json" -X GET
https://<controller>/api/v2/users/
```

認証局をまだセットアップしておらず、SSL を使用している場合は、**-k** フラグが必要になる場合があります。

トークンを取り消すには、そのトークンの ID を使用して、トークンの **Details** ページで **DELETE** を使用します。

以下に例を示します。

```
curl -ku <user>:<password> -X DELETE https://<controller>/api/v2/tokens/<pk>/
```

同様に、トークンを使用する場合は次のようになります。

```
curl -H "Authorization: Bearer <token-value>" -X DELETE https://<controller>/api/v2/tokens/<pk>/ -k
```

20.3. アプリケーションの機能

いくつかの OAuth 2 ユーティリティーエンドポイントは、認可、トークンの更新、取り消しに使用されます。**/api/o/** エンドポイントはブラウザーで使用するためのものではなく、HTTP GET をサポートしません。ここで規定するエンドポイントは OAuth 2 の RFC 仕様に厳密に準拠しているため、詳細なリファレンスとして当該 RFC を使用してください。

以下は、Automation Controller におけるこれらのエンドポイントの一般的な使用例です。特に、さまざまな付与タイプを使用してアプリケーションを作成する場合の例を示します。

20.3.1. authorization code 付与タイプを使用したアプリケーション

アクセストークンを外部アプリケーションまたはサービスに直接発行する必要がある場合は、アプリケーションの **authorization code** 付与タイプを使用する必要があります。

注記

アプリケーション使用時のアクセストークンの取得には **authorization code** タイプのみ使用できます。外部 Web アプリケーションを Automation Controller と統合する場合、Web アプリケーションは、Web アプリケーションのユーザーに代わって OAuth2 トークンを作成する必要がある場合があります。次の理由で、Automation Controller で **authorization code** 付与タイプを使用してアプリケーションを作成することが推奨されます。

- 外部アプリケーションが、ユーザーの認証情報を使用してユーザーの Automation Controller からトークンを取得できるため。
- 特定のアプリケーション用に発行されたトークンが区画化されていることで、それらのトークンを簡単に管理できるため。たとえば、システム内の **すべて** のトークンを取り消すことなく、アプリケーションに関連付けられたすべてのトークンを取り消すことができます。

例

authorization-code 付与タイプを使用して **AuthCodeApp** という名前のアプリケーションを作成するには、**/api/v2/applications/** エンドポイントに対して POST を実行します。

```
{
    "name": "AuthCodeApp",
    "user": 1,
    "client_type": "confidential",
```

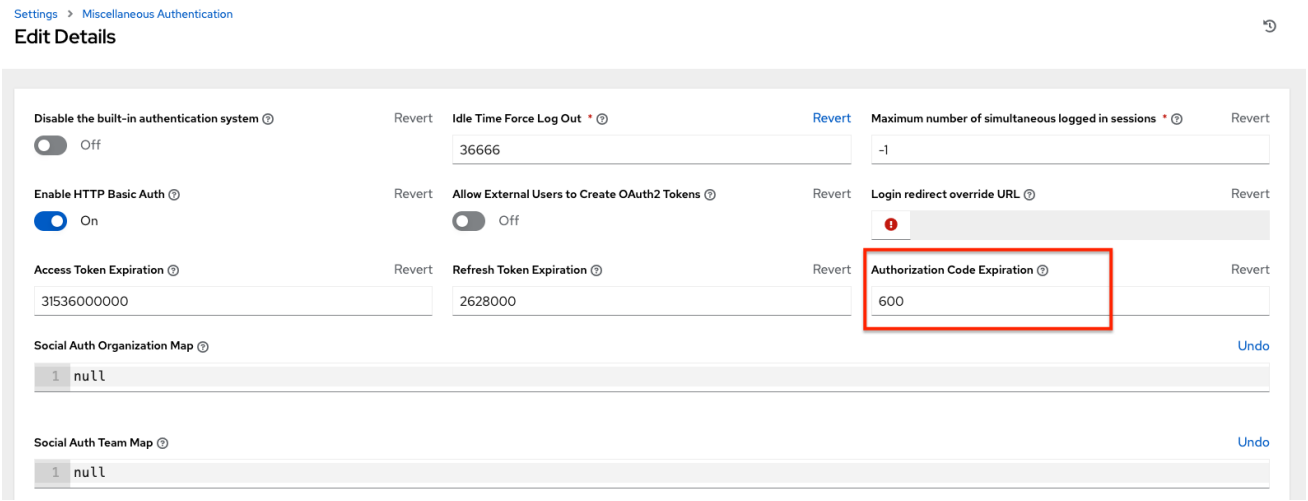
```
    "redirect_uris": "http://<controller>/api/v2",
    "authorization_grant_type": "authorization-code",
    "skip_authorization": false
}

.. _`Django-oauth-toolkit simple test application`: http://django-oauth-
toolkit.herokuapp.com/consumer/
```

response_type、**client_id**、**redirect_uris**、および **scope** を使用して、クライアントアプリケーションから **authorize** エンドポイントに **GET** 要求を送信する場合、以下のワークフローが発生します。

1. Automation Controller は、アプリケーションで指定された **redirect_uri** に対して、認可コードとステータスで応答します。
2. 次に、クライアントアプリケーションは、**code**、**client_id**、**client_secret**、**grant_type**、および **redirect_uri** を使用して、Automation Controller の **api/o/token/** エンドポイントに **POST** 要求を送信します。
3. Automation Controller は、**access_token**、**token_type**、**refresh_token**、および **expires_in** で応答します。

このワークフローの詳細およびテストについては、Django OAuth Toolkit の Test Your Authorization Server を参照してください。

System settings ページで、認可コードが有効な期間を秒単位で指定できます。



この期間の経過後にアクセストークンを要求すると失敗します。

期間のデフォルトは、RFC6749 の推奨に基づいて 600 秒 (10 分) です。

認可コード付与タイプを使用してアプリケーション統合をセットアップする最良の方法は、クロスサイト要求の発信元を許可リストに登録することです。より一般的には、アクセストークンを提供する対象である、Automation Controller と統合するサービスまたはアプリケーションを許可リストに登録する必要があります。

これを行うには、管理者に **/etc/tower/conf.d/custom.py** のローカル Automation Controller 設定にこの許可リストを追加してもらいます。

```
CORS_ORIGIN_ALLOW_ALL = True
CORS_ALLOWED_ORIGIN_REGEXES = [
    r"http://django-oauth-toolkit.herokuapp.com*",
```

```
    r"http://www.example.com*"
]
```

ここで、**http://django-oauth-toolkit.herokuapp.com** と **http://www.example.com** は、Automation Controller にアクセスするためにトークンが必要なアプリケーションです。

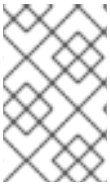
20.3.2. password 付与タイプを使用したアプリケーション

password 付与タイプまたは **Resource owner password-based** 付与タイプは、Web アプリケーションへのネイティブアクセスを持つユーザーに最適であり、クライアントがリソースオーナーの場合に使用する必要があります。以下では、付与タイプが **password** である "Default Application" というアプリケーションを前提としています。

```
{
    "id": 6,
    "type": "application",
    ...
    "name": "Default Application",
    "user": 1,
    "client_id": "gwSPoasWSdNkMDtBN3Hu2WYQpPWCO9SwUEsKK22l",
    "client_secret":
"fI6ZpfocHYBGfm1tP92r0yIgCyfRdDQt0Tos9L8a4fNsJjQQMwp9569eIaUBsaVDgt2eiwOGe0bg5m5vC
SstClZmtdy359RVx2rQK5YIIWyPlrolpt2LEpVeKXWaiybo",
    "client_type": "confidential",
    "redirect_uris": "",
    "authorization_grant_type": "password",
    "skip_authorization": false
}
```

password 付与タイプではログインは必要ないため、**curl** を使用して **/api/v2/tokens/** エンドポイント経由でパーソナルアクセストークンを取得できます。

```
curl -k --user <user>:<password> -H "Content-type: application/json" \
-X POST \
--data '{
    "description": "Token for Nagios Monitoring app",
    "application": 1,
    "scope": "write"
}' \
https://<controller>/api/v2/tokens/
```



注記

特別な OAuth 2 エンドポイントは、**x-www-form-urlencoded** という **Content-type** の使用のみをサポートします。したがって、**api/o/*** エンドポイントで **application/json** を受け入れるものはありません。

成功すると、アクセストークン、リフレッシュトークン、その他の情報を含む応答が JSON 形式で表示されます。

```
HTTP/1.1 200 OK
Server: nginx/1.12.2
Date: Tue, 05 Dec 2017 16:48:09 GMT
```

```
Content-Type: application/json
Content-Length: 163
Connection: keep-alive
Content-Language: en
Vary: Accept-Language, Cookie
Pragma: no-cache
Cache-Control: no-store
Strict-Transport-Security: max-age=15768000

{"access_token": "9epHOqHhnXUcgYK8QanOmUQPSgX92g", "token_type": "Bearer", "expires_in":
315360000000, "refresh_token": "jMRX6QvzOTf046KHee3TU5mT3nyXsz", "scope": "read"}
```

20.4. アプリケーショントークンの機能

/api/o/ エンドポイントのトークンの場合、トークンに関連付けられた **refresh** 機能および **revoke** 機能は、現在、アプリケーショントークンでのみ実行できます。

20.4.1. 既存のアクセストークンの更新

以下の例には、提供された更新トークンと既存のアクセストークンが示されています。

```
{
    "id": 35,
    "type": "access_token",
    ...
    "user": 1,
    "token": "omMFLk7UKpB36WN2Qma9H3gbwEBSOc",
    "refresh_token": "AL0NK9TTpv0qp54dGbC4VUZtsZ9r8z",
    "application": 6,
    "expires": "2017-12-06T03:46:17.087022Z",
    "scope": "read write"
}
```

/api/o/token/ エンドポイントは、アクセストークンの更新に使用されます。

```
curl -X POST \
    -d "grant_type=refresh_token&refresh_token=AL0NK9TTpv0qp54dGbC4VUZtsZ9r8z" \
    -u
"gwSPoasWSdNkMDtBN3Hu2WYQpPWCO9SwUEsKK22l:fI6ZpfocHYBGfm1tP92r0yIgCyfRdDQt0Tos
9L8a4fNsJjQQMwp9569eIaUBsaVDgt2eiwOGe0bg5m5vCSstClZmtdy359RVx2rQK5YlIWyPlrolpt2LEp
VeKXWaiybo" \
    http://<controller>/api/o/token/ -i
```

ここで、**refresh_token** は、前述のアクセストークンの **refresh_token** フィールドで指定されます。

認証情報の形式は、**<client_id>:<client_secret>** です。**client_id** と **client_secret** は、アクセストークンの基盤となる関連アプリケーションの対応するフィールドに置き換えます。

注記

特別な OAuth 2 エンドポイントは、**x-www-form-urlencoded** という **Content-type** の使用のみをサポートします。したがって、**api/o/*** エンドポイントで **application/json** を受け入れるものはありません。

成功すると、以前のものと同じスコープ情報を持つ新しい (更新された) アクセストークンを含む応答が、JSON 形式で表示されます。

```
HTTP/1.1 200 OK
Server: nginx/1.12.2
Date: Tue, 05 Dec 2017 17:54:06 GMT
Content-Type: application/json
Content-Length: 169
Connection: keep-alive
Content-Language: en
Vary: Accept-Language, Cookie
Pragma: no-cache
Cache-Control: no-store
Strict-Transport-Security: max-age=15768000

{"access_token": "NDInWxGJI4iZgqpsreujjbvzCfJqgR", "token_type": "Bearer", "expires_in":
315360000000, "refresh_token": "DqOrmz8bx3srlHkZNKmDpqA86bnQkT", "scope": "read write"}
```

更新操作では、元のトークンを削除した直後に、元のトークンと同じスコープと関連アプリケーションを持つ新しいトークンを作成することで、既存のトークンを置き換えます。

/api/v2/tokens/ エンドポイントで、新しいトークンが存在し、古いトークンが削除されていることを確認します。

20.4.2. アクセストークンの取り消し

/api/o/revoke-token/ エンドポイントを使用して、アクセストークンを取り消すことができます。

この方法によるアクセストークンの取り消しは、トークンリソースオブジェクトの削除と同じです。ただしこの方法では、トークン値と、関連付けられた **client_id** (アプリケーションが **confidential** の場合は加えて **client_secret**) を指定することで、トークンを削除できます。以下に例を示します。

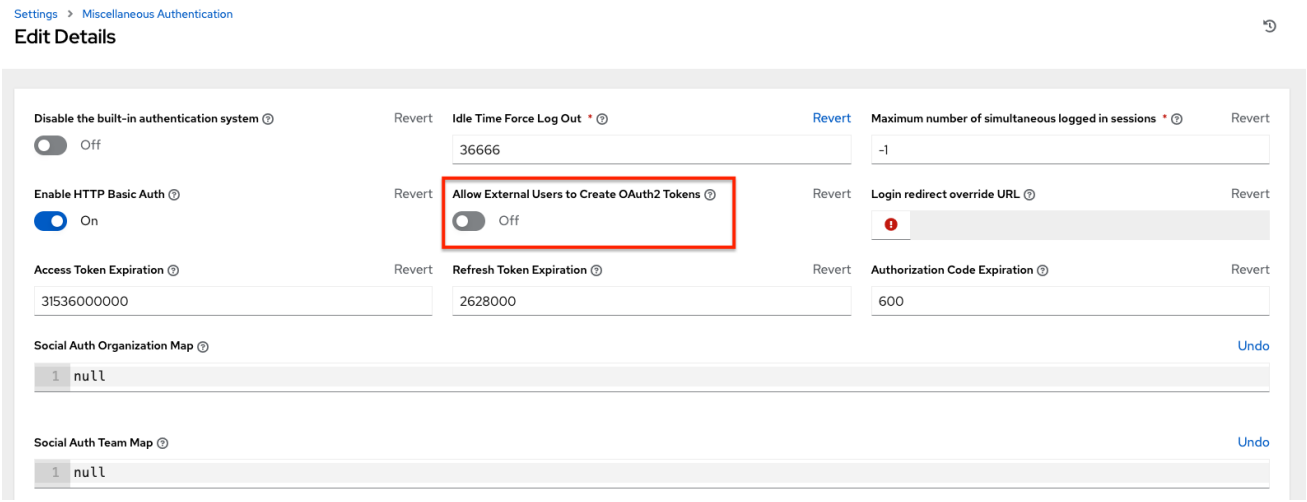
```
curl -X POST -d "token=rQONsve372fQwuc2pn76k3IHDCYpi7" \
-u
"gwSPoasWSdNkMDtBN3Hu2WYQpPWCO9SwUEsKK22l:fI6ZpfocHYBGfm1tP92r0yIgCyfRdDQt0Tos
9L8a4fNsJjQQMwp9569eIaUBsaVDgt2eiwOGe0bg5m5vCSstClZmtdy359RVx2rQK5YIlWyPlrolpt2LEp
VeKXWaiybo" \
http://<controller>/api/o/revoke_token/ -i
```

注記

- 特別な OAuth 2 エンドポイントは、**x-www-form-urlencoded** という **Content-type** の使用のみをサポートします。したがって、**api/o/*** エンドポイントで **application/json** を受け入れるものはありません。
- **Allow External Users to Create Oauth2 Tokens**(API の **ALLOW_OAUTH2_FOR_EXTERNAL_USERS**) の設定は、デフォルトでは無効になっています。外部ユーザーとは、LDAP などのサービスやその他の SSO サービスにより、外部で認証されたユーザーを指します。この設定により、外部ユーザーは独自のトークンを作成できなくなります。有効にしてから無効にすると、その間に外部ユーザーが作成したトークンは残り、自動的には取り消されません。

あるいは、OAuth2 トークンを取り消すために、**manage** ユーティリティーを使用できます。oauth2 トークンの取り消し を参照してください。

この設定は、UI からシステムレベルで設定できます。



成功すると、**200 OK** という応答が表示されます。トークンが **/api/v2/tokens/** エンドポイントに存在するかどうかをチェックして、削除されていることを確認します。

第21章 ソーシャル認証のセットアップ

認証方法を使用すると、エンドユーザーのログインを簡素化できます。専用の新しいログインアカウントを作成せずに、既存のログイン情報を使用してシングルサインオンでサードパーティーの Web サイトにサインインできます。

ユーザーインターフェイスでアカウント認証を設定し、それを PostgreSQL データベースに保存できます。

Automation Controller では、OAuth2 を一元的に使用するようにアカウント認証を設定できます。認証情報のソースとして SAML、RADIUS、さらには LDAP を使用して、エンタープライズレベルのアカウント認証を設定することもできます。

Microsoft Azure、Google、GitHub など、アカウント情報を提供する Web サイトの場合、アカウント情報はしばしば OAuth 標準を使用して実装されます。

OAuth はセキュアな認証プロトコルです。通常、これはアカウント認証と組み合わせて使用され、サードパーティーのアプリケーションに "セッショントークン" を付与して、アプリケーションにユーザーに代わってプロバイダーへの API 呼び出しを実行できるようにします。

Security Assertion Markup Language(SAML) は、アイデンティティープロバイダーとサービスプロバイダー間でアカウント認証と認可データを交換するための、オープン標準の XML データ形式です。

RADIUS 分散クライアント/サーバーシステムを使用すると、不正アクセスからネットワークを保護できます。このようなシステムを、高いレベルのセキュリティーを必要とするネットワーク環境に実装しつつ、リモートユーザーのネットワークアクセスを維持できます。

関連情報

詳細は、Automation Controller 設定 セクションを参照してください。

21.1. GITHUB 設定

GitHub のソーシャル認証をセットアップするには、Web アプリケーションの OAuth2 キーとシークレットを取得する必要があります。これを行うには、まず新しいアプリケーションを GitHub (https://github.com/settings/developers) に登録する必要があります。

アプリケーションを登録するには、ホームページの URL を指定する必要があります。この URL は、**GitHub default settings** ページの **Details** タブに表示される **Callback URL** です。OAuth2 キー (クライアント ID) とシークレット (クライアントシークレット) は、UI の必須フィールドに入力する際に使用します。

手順

1. ナビゲーションパネルから **Settings** を選択します。
2. **Settings** ページで、**Authentication** オプションのリストから **GitHub settings** を選択します。
3. **GitHub Default** タブを選択していない場合には、このタブを選択します。
   GitHub OAuth2 Callback URL フィールドには、事前に情報が入力されており、編集できません。アプリケーションの登録が済むと、GitHub ではクライアント ID およびクライアントシークレットが表示されます。
4. **Edit** をクリックし、GitHub のクライアント ID をコピーして **GitHub OAuth2 Key** フィールドにペーストします。

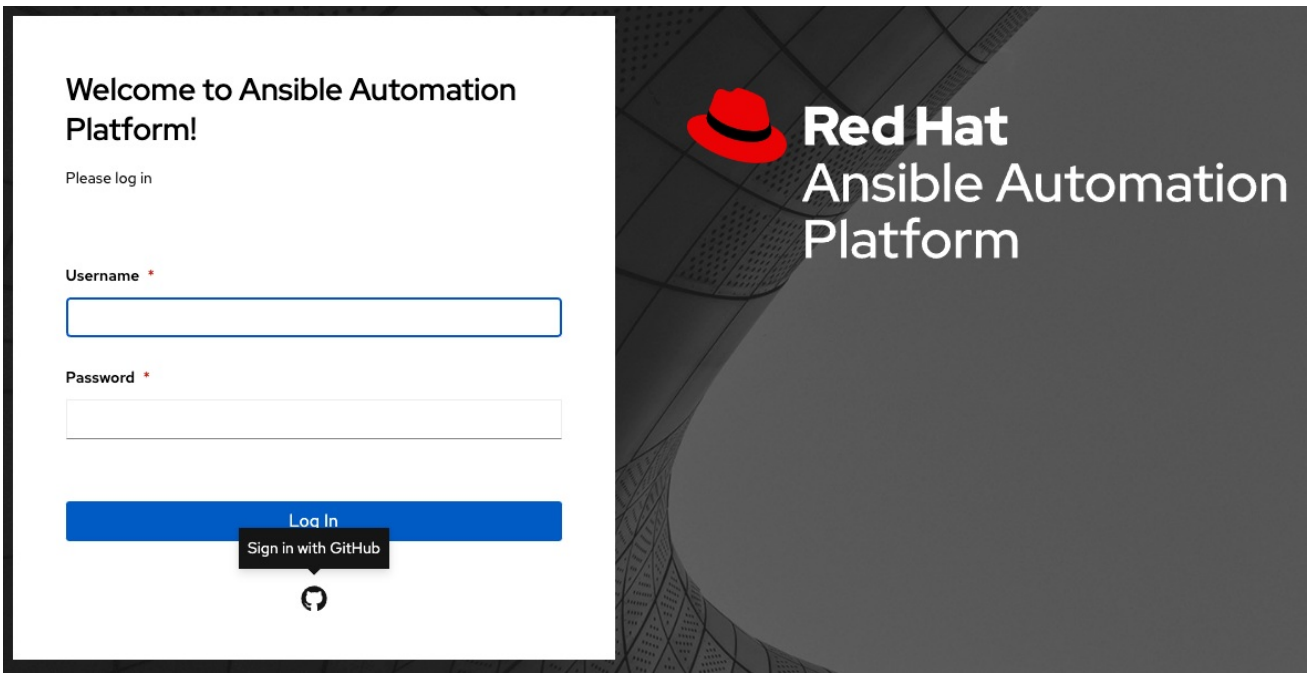
5. GitHub のクライアントシークレットをコピーして、**GitHub OAuth2 Secret** フィールドにペーストします。

6. マッピングフィールドに入力する方法の詳細は、組織マッピング および チームマッピング を参照してください。

7. **Save** をクリックします。

検証

認証が正しく設定されたことを確認するには、Automation Controller からログアウトします。ログイン画面に GitHub ロゴが表示され、これらの認証情報を使用してログインできるようになります。



21.1.1. GitHub Organization の設定

組織または組織内のチームでアカウント認証を定義する場合は、特定の組織およびチーム設定を使用する必要があります。アカウント認証は、組織ごと、または組織内のチームごとに制限を設定できます。

組織以外またはチーム以外の設定を指定して、すべてを許可するように設定することも可能です。

コントローラーにログインできるユーザーを、組織または組織内のチームに所属するユーザーのみに制限することができます。

GitHub Organization のソーシャル認証をセットアップするには、Web アプリケーションの OAuth2 キーとシークレットを取得する必要があります。これを行うには、まず組織が所有するアプリケーションを https://github.com/organizations/<yourorg>/settings/applications に登録する必要があります。

アプリケーションを登録するには、認可コールバック URL を指定する必要があります。この URL は、**Details** ページに表示される **Callback URL** です。各キーとシークレットは、一意のアプリケーションに属している必要があり、異なる認証バックエンド間で共有したり再利用したりすることはできません。OAuth2 キー (クライアント ID) とシークレット (クライアントシークレット) は、UI の必須フィールドに入力する際に使用します。

手順

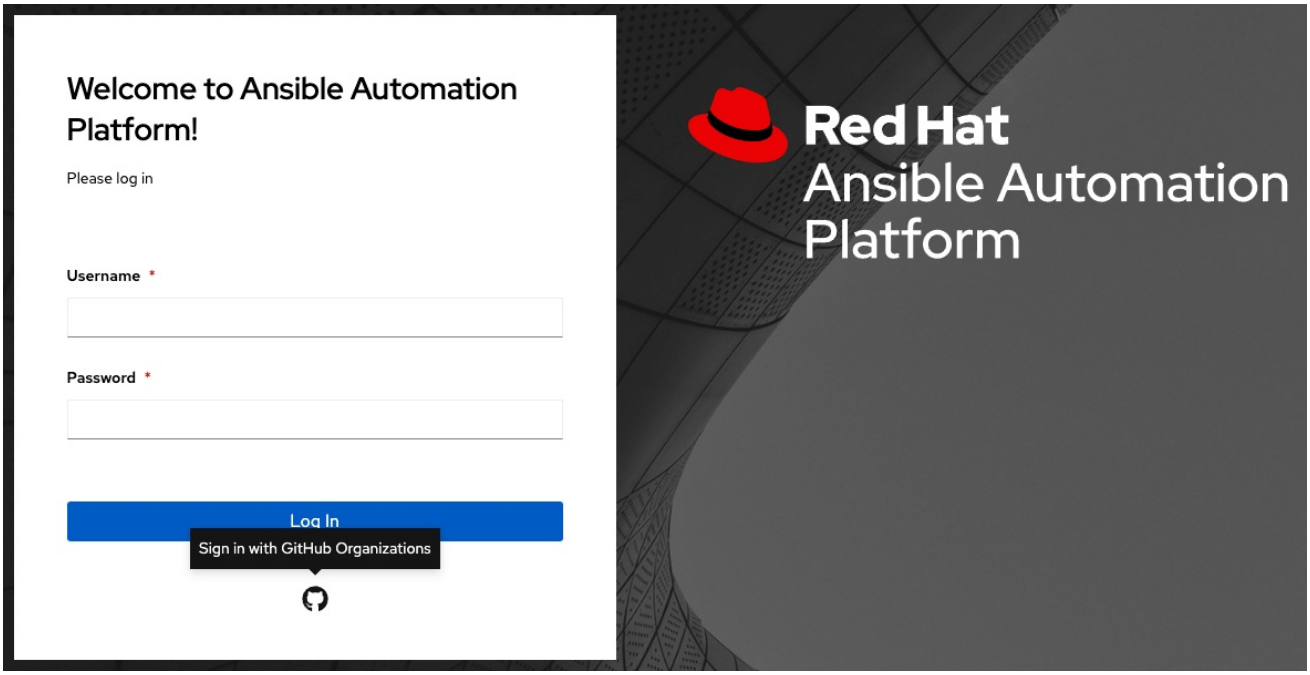
1. ナビゲーションパネルから **Settings** を選択します。

2. **Settings** ページで、**Authentication** オプションのリストから **GitHub settings** を選択します。
3. **GitHub Organization** タブを選択します。
   GitHub Organization OAuth2 Callback URL フィールドには、事前に情報が入力されており、編集できません。

   アプリケーションの登録が済むと、GitHub ではクライアント ID およびクライアントシークレットが表示されます。
4. **Edit** をクリックし、GitHub のクライアント ID をコピーして **GitHub Organization OAuth2 Key** フィールドにペーストします。
5. GitHub のクライアントシークレットをコピーして、**GitHub Organization OAuth2 Secret** フィールドにペーストします。
6. 組織の URL (例: https://github.com/<yourorg>/) で使用されている GitHub 組織の名前を、**GitHub Organization Name** フィールドに入力します。
7. マッピングフィールドに入力する方法の詳細は、組織マッピング および チームマッピング を参照してください。
8. **Save** をクリックします。

検証

認証が正しく設定されたことを確認するには、Automation Controller からログアウトします。ログイン画面に GitHub Organization のロゴが表示され、これらの認証情報を使用してログインできるようになります。



21.1.2. GitHub Team の設定

GitHub Team のソーシャル認証をセットアップするには、Web アプリケーションの OAuth2 キーとシークレットを取得する必要があります。これを行うには、まずチームが所有するアプリケーションを **https://github.com/organizations/<yourorg>/settings/applications** に登録する必要があります。アプリケーションを登録するには、認可コールバック URL を指定する必要があります。この URL は、**Details** ページに表示される **Callback URL** です。各キーとシークレットは、一意のアプリケー

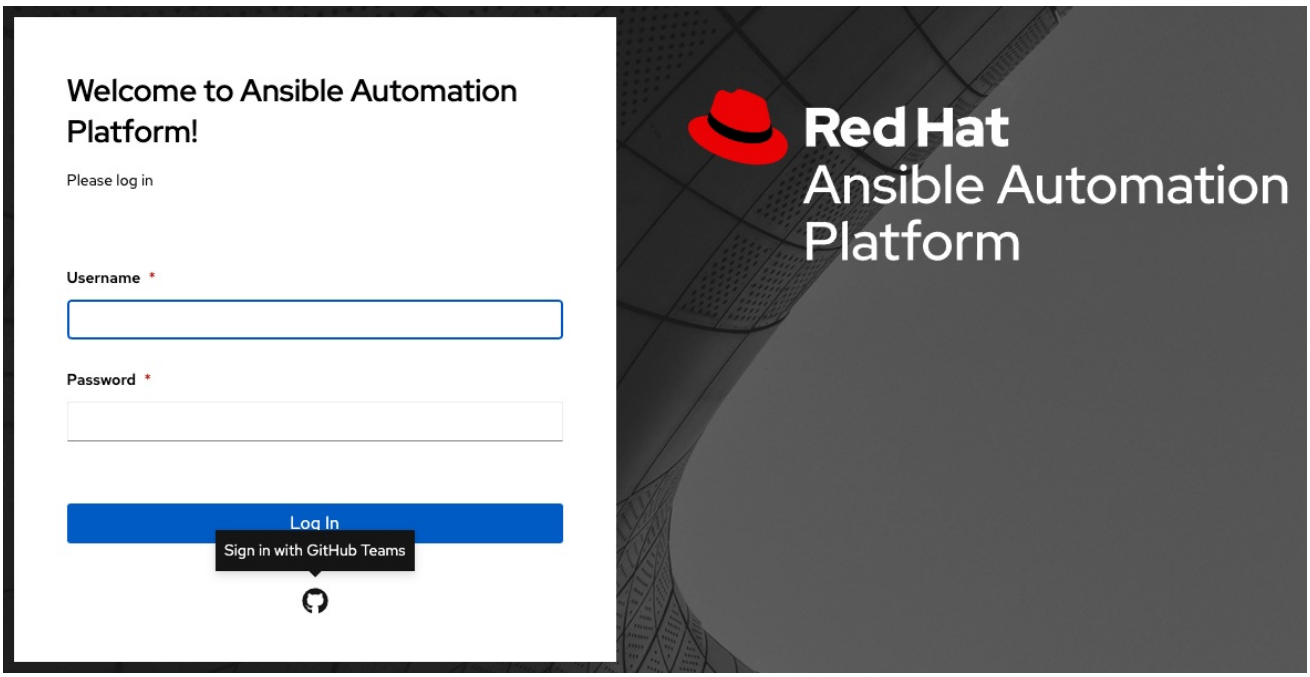
ションに属している必要があり、異なる認証バックエンド間で共有したり再利用したりすることはできません。OAuth2 キー (クライアント ID) とシークレット (クライアントシークレット) は、UI の必須フィールドに入力する際に使用します。

手順

1. GitHub API を使用して数値のチーム ID を検索します。チーム ID は、UI の必須フィールドに入力する際に使用します。
2. ナビゲーションパネルから **Settings** を選択します。
3. **Settings** ページで、**Authentication** オプションのリストから **GitHub settings** を選択します。
4. **GitHub Team** タブをクリックします。
   GitHub Team OAuth2 Callback URLフィールドには、事前に情報が入力されており、編集できません。アプリケーションの登録が済むと、GitHub ではクライアント ID およびクライアントシークレットが表示されます。
5. **Edit** をクリックし、GitHub のクライアント ID をコピーして **GitHub Team OAuth2 Key** フィールドにペーストします。
6. GitHub のクライアントシークレットをコピーして、**GitHub Team OAuth2 Secret**フィールドにペーストします。
7. GitHub のチーム ID をコピーして、**GitHub Team ID** フィールドにペーストします。
8. マッピングフィールドに入力する方法の詳細は、組織マッピング および チームマッピング を参照してください。
9. **Save**をクリックします。

検証

認証が正しく設定されたことを確認するには、Automation Controller からログアウトします。ログイン画面に GitHub Team のロゴが表示され、これらの認証情報を使用してログインできるようになります。



21.1.3. GitHub Enterprise の設定

GitHub Enterprise のソーシャル認証をセットアップするには、GitHub Enterprise URL、API URL、Web アプリケーションの OAuth2 キーとシークレットを取得する必要があります。

URL を取得するには、GitHub Enterprise administration ドキュメントを参照してください。

キーとシークレットを取得するには、まず企業が所有するアプリケーションを https://github.com/organizations/<yourorg>/settings/applications に登録する必要があります。

アプリケーションを登録するには、認可コールバック URL を指定する必要があります。この URL は、**Details** ページに表示される **Callback URL** です。これは **github.com** ではなくサイトでホストされているため、通信する認証アダプターを指定する必要があります。

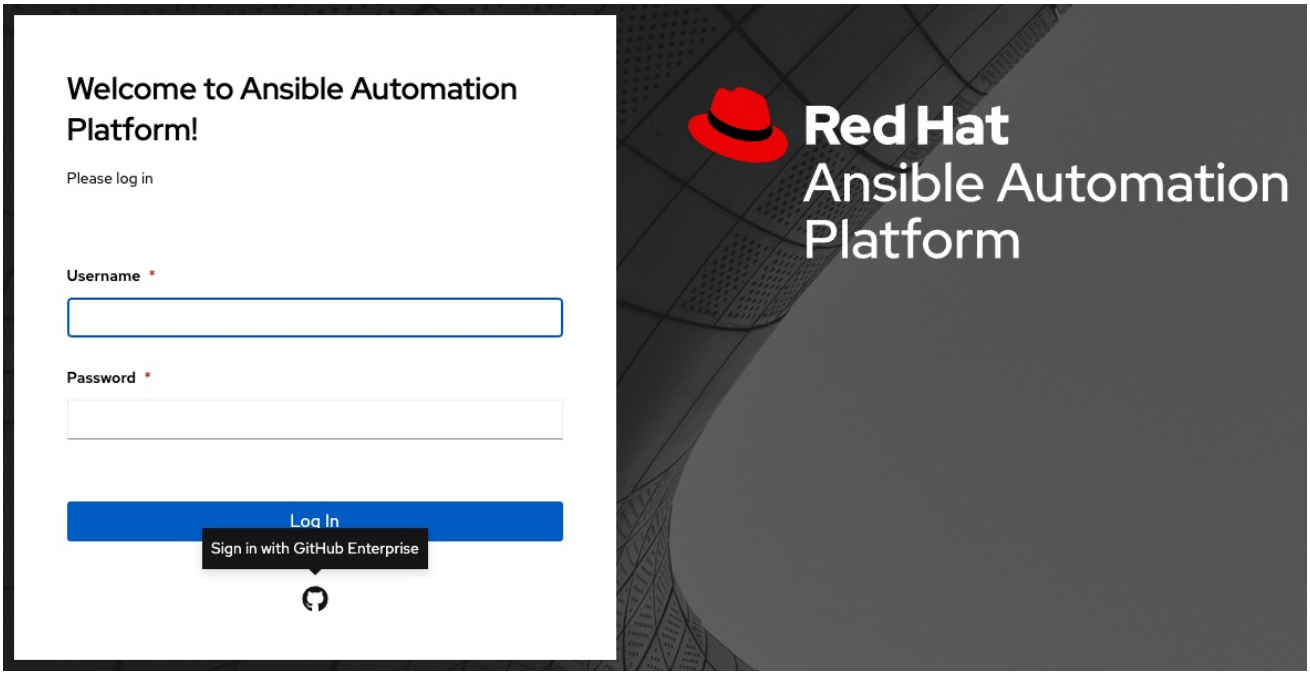
各キーとシークレットは、一意のアプリケーションに属している必要があり、異なる認証バックエンド間で共有したり再利用したりすることはできません。OAuth2 キー (クライアント ID) とシークレット (クライアントシークレット) は、UI の必須フィールドに入力する際に使用します。

手順

1. ナビゲーションパネルから **Settings** を選択します。
2. **Settings** ページで、**Authentication** オプションのリストから **GitHub settings** を選択します。
3. **GitHub Enterprise** タブをクリックします。
   GitHub Enterprise OAuth2 Callback URL フィールドには、事前に情報が入力されており、編集できません。アプリケーションの登録が済むと、GitHub ではクライアント ID およびクライアントシークレットが表示されます。
4. GitHub Enterprise の設定を行うには、**Edit** をクリックします。
5. **GitHub Enterprise URL** フィールドに、GitHub Enterprise インスタンスのホスト名を入力します (例: https://github.example.com)。
6. **GitHub Enterprise API URL** フィールドに、GitHub Enterprise インスタンスの API URL を入力します (例: https://github.example.com/api/v3)。
7. GitHub のクライアント ID をコピーして、**GitHub Enterprise OAuth2 Key** フィールドにペーストします。
8. GitHub のクライアントシークレットをコピーして、**GitHub Enterprise OAuth2 Secret** フィールドにペーストします。
9. マッピングフィールドに入力する方法の詳細は、組織マッピング および チームマッピング を参照してください。
10. **Save** をクリックします。

検証

認証が正しく設定されたことを確認するには、Automation Controller からログアウトします。ログイン画面に GitHub Enterprise のロゴが表示され、これらの認証情報を使用してログインできるようになります。



21.1.4. GitHub Enterprise Organization の設定

GitHub Enterprise Organization のソーシャル認証をセットアップするには、GitHub Enterprise Organization URL、Organization API URL、Web アプリケーションの Organization OAuth2 キーとシークレットを取得する必要があります。

URL を取得するには、GitHub ドキュメントの GitHub Enterprise administration を参照してください。

キーとシークレットを取得するには、まず企業の組織が所有するアプリケーションを **https://github.com/organizations/<yourorg>/settings/applications** に登録する必要があります。

アプリケーションを登録するには、認可コールバック URL を指定する必要があります。この URL は、**Details** ページに表示される **Callback URL** です。

これは **github.com** ではなくサイトでホストされているため、通信する認証アダプターを指定する必要があります。

各キーとシークレットは、一意のアプリケーションに属している必要があり、異なる認証バックエンド間で共有したり再利用したりすることはできません。OAuth2 キー (クライアント ID) とシークレット (クライアントシークレット) は、UI の必須フィールドに入力する際に使用します。

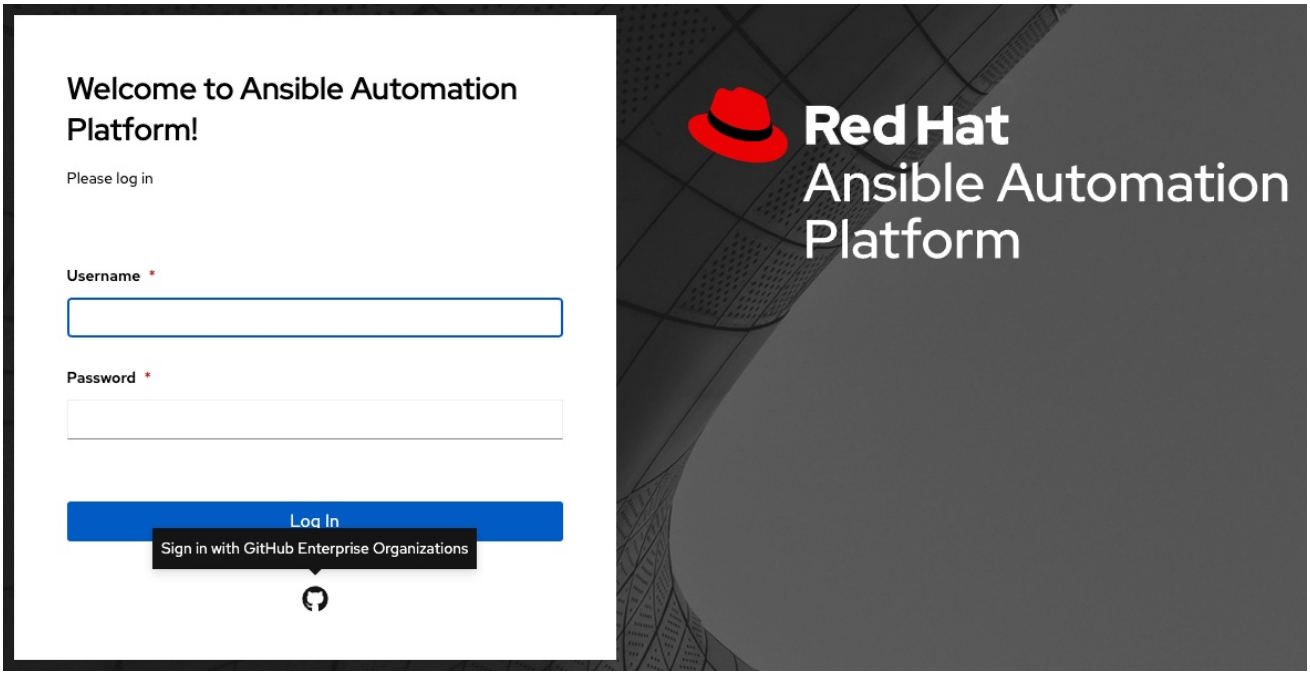
手順

1. ナビゲーションパネルから **Settings** を選択します。
2. **Settings** ページで、**Authentication** オプションのリストから **GitHub settings** を選択します。
3. **GitHub Enterprise Organization** タブをクリックします。
   GitHub Enterprise Organization OAuth2 Callback URLフィールドには、事前に情報が入力されており、編集できません。アプリケーションの登録が済むと、GitHub ではクライアント ID およびクライアントシークレットが表示されます。
4. GitHub Enterprise Organization の設定を行うには、**Edit** をクリックします。
5. **GitHub Enterprise Organization URL** フィールドに、GitHub Enterprise Organization インスタンスのホスト名を入力します (例: https://github.orgexample.com)。

6. **GitHub Enterprise Organization API URL** フィールドに、GitHub Enterprise Organization インスタンスの API URL を入力します (例: https://github.orgexample.com/api/v3)。
7. GitHub のクライアント ID をコピーして、**GitHub Enterprise Organization OAuth2 Key** フィールドにペーストします。
8. GitHub のクライアントシークレットをコピーして、**GitHub Enterprise Organization OAuth2 Secret** フィールドにペーストします。
9. 組織の URL (例: https://github.com/<yourorg>/) で使用されている GitHub Enterprise 組織の名前を、**GitHub Enterprise Organization Name** フィールドに入力します。
10. マッピングフィールドに入力する方法の詳細は、組織マッピング および チームマッピング を参照してください。
11. **Save** をクリックします。

検証

認証が正しく設定されたことを確認するには、Automation Controller からログアウトします。ログイン画面に GitHub Enterprise Organization のロゴが表示され、これらの認証情報を使用してログインできるようになります。



21.1.5. GitHub Enterprise Team の設定

GitHub Enterprise チームのソーシャル認証をセットアップするには、GitHub Enterprise Organization URL、Organization API URL、Web アプリケーションの Organization OAuth2 キーとシークレットを取得する必要があります。

URL を取得するには、GitHub ドキュメントの GitHub Enterprise administration を参照してください。

キーとシークレットを取得するには、まず企業のチームが所有するアプリケーションを **https://github.com/organizations/<yourorg>/settings/applications** に登録する必要があります。

アプリケーションを登録するには、認可コールバック URL を指定する必要があります。この URL は、**Details** ページに表示される **Callback URL** です。これは github.com ではなくサイトでホストされているため、通信する認証アダプターを指定する必要があります。

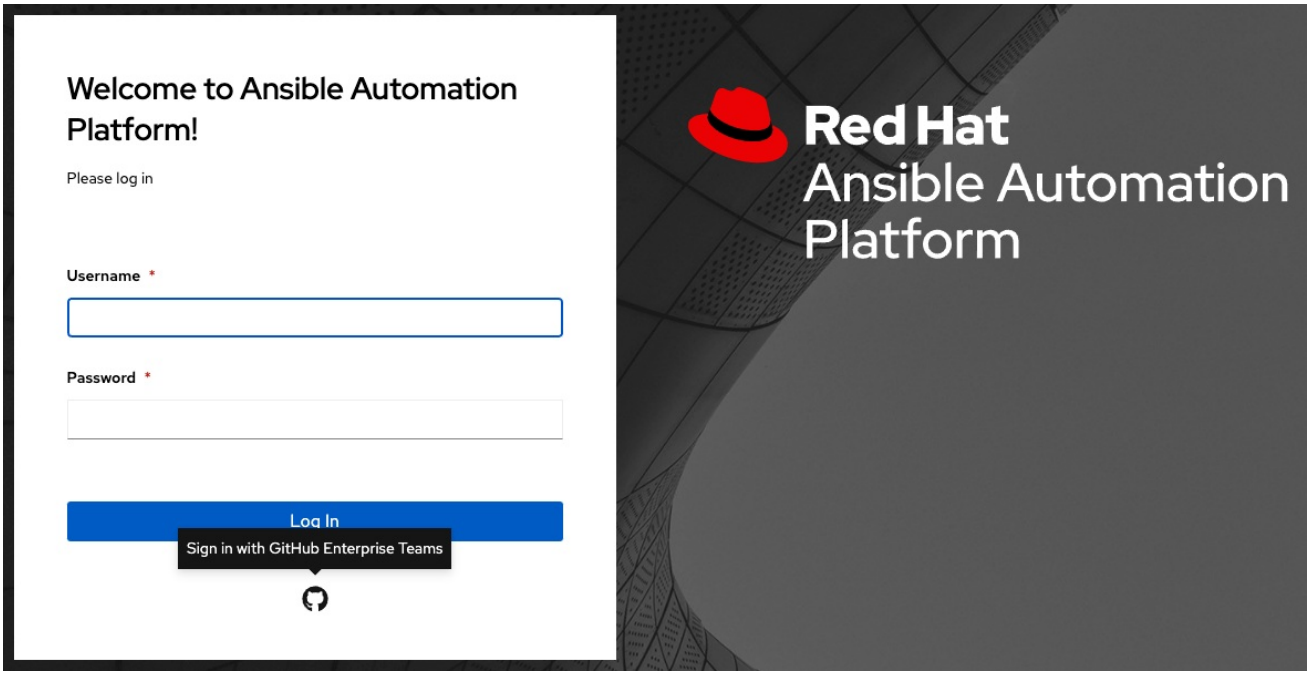
各キーとシークレットは、一意のアプリケーションに属している必要があり、異なる認証バックエンド間で共有したり再利用したりすることはできません。OAuth2key (クライアント ID) と シークレット (クライアントシークレット) は、UI の必須フィールドに入力する際に使用します。

手順

1. GitHub API を使用して数値のチーム ID を検索します。チーム ID は、UI の必須フィールドに入力する際に使用します。
2. ナビゲーションパネルから **Settings** を選択します。
3. **Settings** ページで、**Authentication** オプションのリストから **GitHub settings** を選択します。
4. **GitHub Enterprise Team** タブをクリックします。
GitHub Enterprise Team OAuth2 Callback URLフィールドには、事前に情報が入力されており、編集できません。アプリケーションの登録が済むと、GitHub ではクライアント ID およびクライアントシークレットが表示されます。
5. GitHub Enterprise Team の設定を行うには、**Edit** をクリックします。
6. **GitHub Enterprise Team URL**フィールドに、GitHub Enterprise チームインスタンスのホスト名を入力します (例: https://github.teamexample.com)。
7. **GitHub Enterprise Team API URL**フィールドに、GitHub Enterprise チームインスタンスの API URL を入力します (例: https://github.teamexample.com/api/v3)。
8. GitHub のクライアント ID をコピーして、**GitHub Enterprise Team OAuth2 Key**フィールドにペーストします。
9. GitHub のクライアントシークレットをコピーして、**GitHub Enterprise Team OAuth2 Secret** フィールドにペーストします。
10. GitHub のチーム ID をコピーして、**GitHub Enterprise Team ID**フィールドにペーストします。
11. マッピングフィールドに入力する方法の詳細は、組織マッピング および チームマッピング を参照してください。
12. **Save** をクリックします。

検証

認証が正しく設定されたことを確認するには、Automation Controller からログアウトします。ログイン画面に GitHub Enterprise Teams のロゴが表示され、これらの認証情報を使用してログインできるようになります。



21.2. GOOGLE OAUTH2 の設定

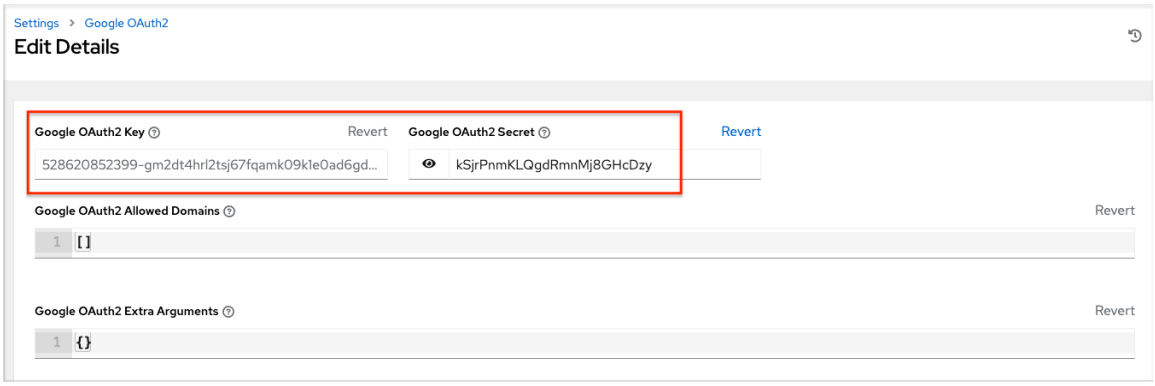
Google のソーシャル認証をセットアップするには、Web アプリケーションの OAuth2 キーとシークレットを取得する必要があります。これを行うには、まずプロジェクトを作成し、Google でセットアップする必要があります。

手順については、Google API Console Help ドキュメントの Setting up OAuth 2.0 を参照してください。

すでにセットアッププロセスが完了している場合は、Google API Manager Console の Credentials セクションに移動すると、これらの認証情報にアクセスできます。OAuth2 キー (クライアント ID) とシークレット (クライアントシークレット) は、UI の必須フィールドに入力する際に使用します。

手順

1. ナビゲーションパネルから **Settings** を選択します。
2. **Settings** ページで、**Authentication** オプションのリストから **Google OAuth 2 settings** を選択します。
   Google OAuth2 Callback URL フィールドには、事前に情報が入力されており、編集できません。
3. 以下のフィールドにも、事前に情報が入力されています。入力されていない場合は、Web アプリケーションのセットアッププロセスで Google から提供された認証情報を使用し、以下の例と同じ形式の値を探します。
   - **Edit** をクリックし、Google のクライアント ID をコピーして **Google OAuth2 Key** フィールドにペーストします。
   - Google のクライアントシークレットをコピーして、**Google OAuth2 Secret** フィールドにペーストします。



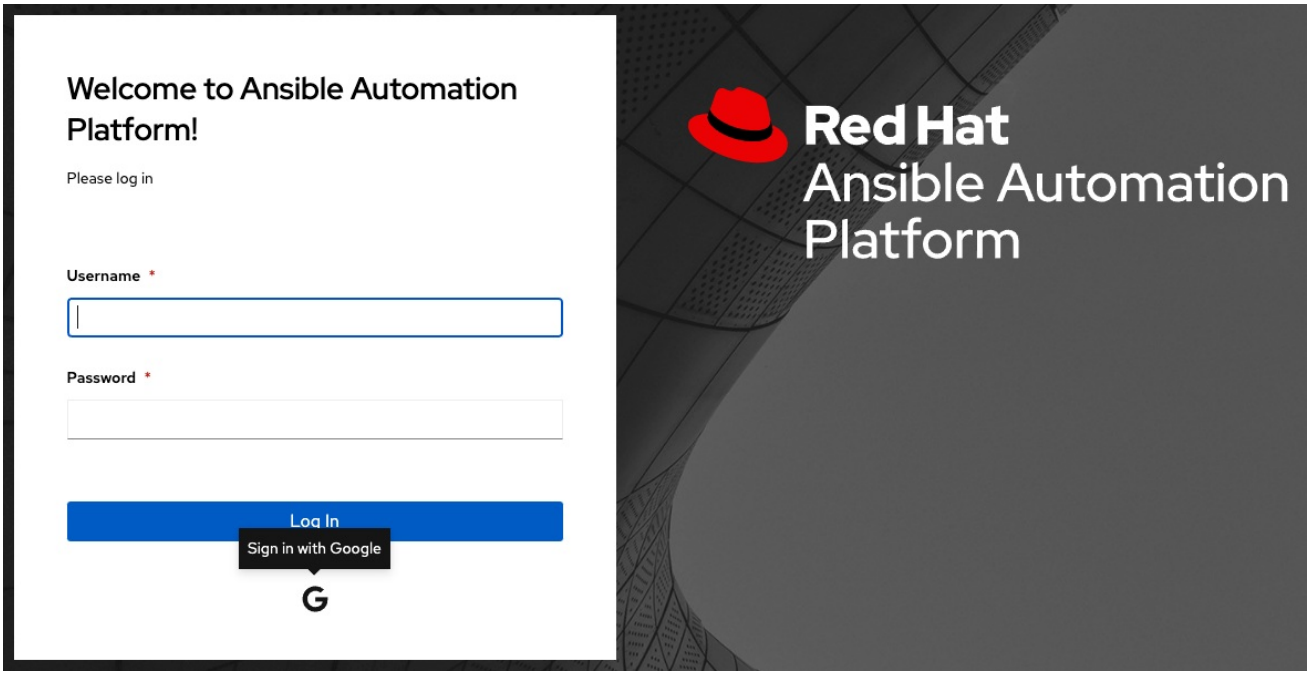
4. 残りの任意フィールドを入力するには、説明や必要な形式について各フィールドのヒントを参照してください。

5. マッピングフィールドに入力する方法の詳細は、組織マッピング および チームマッピング を参照してください。

6. **Save** をクリックします。

検証

認証が正しく設定されたことを確認するには、Automation Controller からログアウトします。ログイン画面には、Automation Controller にログインする別の方法として Google ロゴが表示されます。



21.3. 組織マッピング

ユーザー名とメールアドレスに基づいて、どのユーザーをどの Automation Controller 組織に配置するかを制御する必要があります (組織の管理者とユーザーをソーシャルレベルまたはエンタープライズレベルの認証アカウントから区別します)。

ディクショナリーキーは組織名です。組織がまだ存在しておらず、ライセンスで複数の組織が許可されている場合は、組織が作成されます。それ以外の場合は、キーに関係なく、単一のデフォルト組織が使用されます。

値は、各組織のメンバーシップのオプションを定義するディクショナリーとなります。どのユーザーが自動的に組織のユーザーとなり、どのユーザーが組織を管理できるかを、組織ごとに指定できます。

admins: None、True/False、文字列または文字列のリスト/タプル

- **None** の場合、組織管理者は更新されません。
- **True** の場合、アカウント認証を使用するすべてのユーザーが組織の管理者として自動的に追加されます。
- **False** の場合、いずれのアカウント認証ユーザーも組織の管理者として自動的に追加されません。
- 組織に追加するユーザーのユーザー名およびメールアドレスを文字列または文字列のリストで指定する場合、先頭および末尾が / である文字列は正規表現にコンパイルされます。修飾子 **i** (大文字と小文字の区別なし) および **m** (複数行) は、末尾の / の後に指定できます。

remove_admins: True/False。デフォルトは **True** です。

- **True** の場合、一致しないユーザーは組織の管理者リストから削除されます。
- **users**: None、True/False、文字列または文字列のリスト/タプル。**admins** と同じルールが適用されます。
- **remove_users**: True/False。デフォルトは **True** です。**remove_admins** と同じルールが適用されます。

```
{
    "Default": {
        "users": true
    },
    "Test Org": {
        "admins": ["admin@example.com"],
        "users": true
    },
    "Test Org 2": {
        "admins": ["admin@example.com", "/^controller-[^@]+?@.*$/i"],
        "users": "/^[^@].*?@example\\.com$/"
    }
}
```

組織マッピングは、アカウント認証バックエンドごとに個別に指定できます。定義した場合、上記のグローバル設定よりもこれらの設定が優先されます。

```
SOCIAL_AUTH_GOOGLE_OAUTH2_ORGANIZATION_MAP = {}
SOCIAL_AUTH_GITHUB_ORGANIZATION_MAP = {}
SOCIAL_AUTH_GITHUB_ORG_ORGANIZATION_MAP = {}
SOCIAL_AUTH_GITHUB_TEAM_ORGANIZATION_MAP = {}
SOCIAL_AUTH_SAML_ORGANIZATION_MAP = {}
```

21.4. チームマッピング

チームマッピングは、ソーシャル認証アカウントからのチームメンバー (ユーザー) のマッピングです。キーはチーム名です (存在しない場合は作成されます)。値は、各チームのメンバーシップに関するオプションのディクショナリーとなります。各値には、次のパラメーターを含めることができます。

- **organization**: 文字列。チームが所属する組織の名前です。組織とチームの組み合わせが存在しない場合は、チームが作成されます。組織が存在しない場合には、先に組織が作成されます。ライセンスで複数の組織が許可されていない場合、チームは常に単一のデフォルト組織に割り当てられます。

- **users**: None、True/False、文字列または文字列のリスト/タプル。

  - **None** の場合、チームメンバーは更新されません。

  - **True** の場合、すべてのソーシャル認証ユーザーがチームメンバーとして追加されます。

  - **False** の場合、すべてのソーシャル認証ユーザーがチームメンバーとして削除されます。

- ユーザーの照合に使用する表現を文字列または文字列のリストで指定する場合、ユーザー名またはメールアドレスが一致すると、そのユーザーはチームメンバーとして追加されます。先頭および末尾が / である文字列は、正規表現にコンパイルされます。修飾子 **i** (大文字と小文字の区別なし) および **m** (複数行) は、末尾の / の後に指定できます。

remove: True/False。デフォルトは **True** です。**True** の場合、前述のルールに一致しないユーザーはチームから削除されます。

```
{
    "My Team": {
        "organization": "Test Org",
        "users": ["/^[^@]+?@test\\.example\\.com$/"],
        "remove": true
    },
    "Other Team": {
        "organization": "Test Org 2",
        "users": ["/^[^@]+?@test\\.example\\.com$/"],
        "remove": false
    }
}
```

チームマッピングは、セットアップした内容に基づいて、アカウント認証バックエンドごとに個別に指定できます。定義した場合、上記のグローバル設定よりもこれらの設定が優先されます。

```
SOCIAL_AUTH_GOOGLE_OAUTH2_TEAM_MAP = {}
SOCIAL_AUTH_GITHUB_TEAM_MAP = {}
SOCIAL_AUTH_GITHUB_ORG_TEAM_MAP = {}
SOCIAL_AUTH_GITHUB_TEAM_TEAM_MAP = {}
SOCIAL_AUTH_SAML_TEAM_MAP = {}
```

次の行のコメントを解除します。つまり、**SOCIAL_AUTH_USER_FIELDS** を空のリストに設定して、新しいユーザーアカウントが作成されないようにします。

```
SOCIAL_AUTH_USER_FIELDS = []
```

以前にソーシャルまたはエンタープライズレベルの認証を使用して Automation Controller にログインしたことがあるユーザー、またはメールアドレスが一致するユーザーアカウントを持つユーザーのみが、ログインできるようになります。

第22章 エンタープライズ認証のセットアップ

次のエンタープライズシステムの認証をセットアップします。

- Azure AD の設定
- LDAP 認証
- RADIUS 設定
- SAML 設定
  - 透過的な SAML ログイン
  - SAML のロギングを有効にする
- TACACS+ 設定
- Generic OIDC 設定

注記

LDAP 認証については、LDAP 認証のセットアップ を参照してください。

SAML、RADIUS、および TACACS+ ユーザーは、"エンタープライズ" ユーザーとして分類されます。エンタープライズユーザーには次のルールが適用されます。

- エンタープライズユーザーは、リモートの認証バックエンドからのログインに初めて成功した場合にのみ作成できます。
- Automation Controller 内にエンタープライズ以外のユーザーが同じ名前で作成されている場合、エンタープライズユーザーを作成または認証することはできません。
- エンタープライズバックエンドが有効になっている場合は、エンタープライズユーザーの Automation Controller パスワードは常に空である必要があり、ユーザーによる設定はできません。
- エンタープライズバックエンドが無効になっている場合は、パスワードフィールドを設定することで、エンタープライズユーザーを通常の Automation Controller ユーザーに変換できます。

警告

変換された Automation Controller ユーザーをエンタープライズユーザーとして扱うことはできないため、この操作を元に戻すことはできません。

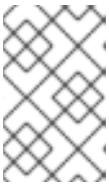
22.1. MICROSOFT AZURE ACTIVE DIRECTORY 認証

Microsoft Azure Active Directory (AD) のエンタープライズ認証をセットアップするには、組織が所有するアプリケーションを Azure から登録 (https://docs.microsoft.com/en-us/azure/active-directory/develop/quickstart-register-app) して、OAuth2 キーとシークレットを取得する必要があります。

各キーとシークレットは、一意のアプリケーションに属している必要があり、異なる認証バックエンド間で共有したり再利用したりすることはできません。アプリケーションを登録するには、Web ページの URL を指定する必要があります。この URL は、**Settings** 画面の **Authentication** タブに表示される Callback URL です。

手順

1. ナビゲーションパネルから **Settings** を選択します。
2. **Authentication** オプションのリストから **Azure AD settings** を選択します。



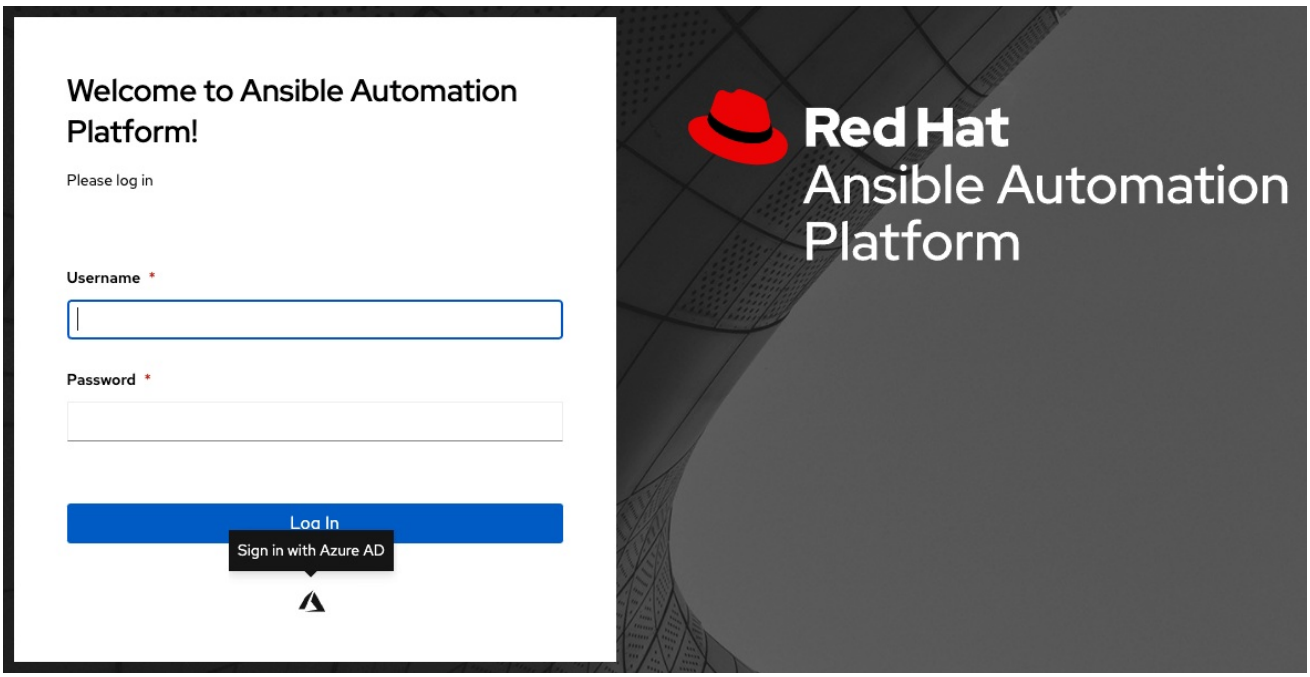
注記

Azure AD OAuth2 Callback URL フィールドには、事前に情報が入力されており、編集できません。アプリケーションが登録されると、Microsoft Azure にアプリケーション ID とオブジェクト ID が表示されます。

3. **Edit** をクリックし、Microsoft Azure のアプリケーション ID をコピーして、 **Azure AD OAuth2 Key** フィールドにペーストします。
アプリケーションと Microsoft Azure Active Directory のリンクに関する Microsoft Azure AD ドキュメントに従い、認証用に、キー (1 度だけ表示) をクライアントに提供します。
4. Microsoft Azure AD アプリケーション用に作成したシークレットキーをコピーして、**Settings - Authentication** 画面の **Azure AD OAuth2 Secret** フィールドにペーストします。
5. Microsoft Azure AD OAuth2 Organization Map フィールドと Microsoft Azure AD OAuth2 Team Map フィールドへの入力方法の詳細は、組織マッピング と チームマッピング を参照してください。
6. **Save** をクリックします。

検証

認証が正しく設定されたことを確認するには、Automation Controller からログアウトします。ログイン画面に Microsoft Azure ロゴが表示され、これらの認証情報でログインできるようになります。



関連情報

Microsoft Azure AD でのアプリケーション登録の基本については、What is the Microsoft identity platform? を参照してください。

22.2. RADIUS 認証

認証情報のソースとして RADIUS を一元的に使用するように Automation Controller を設定できます。

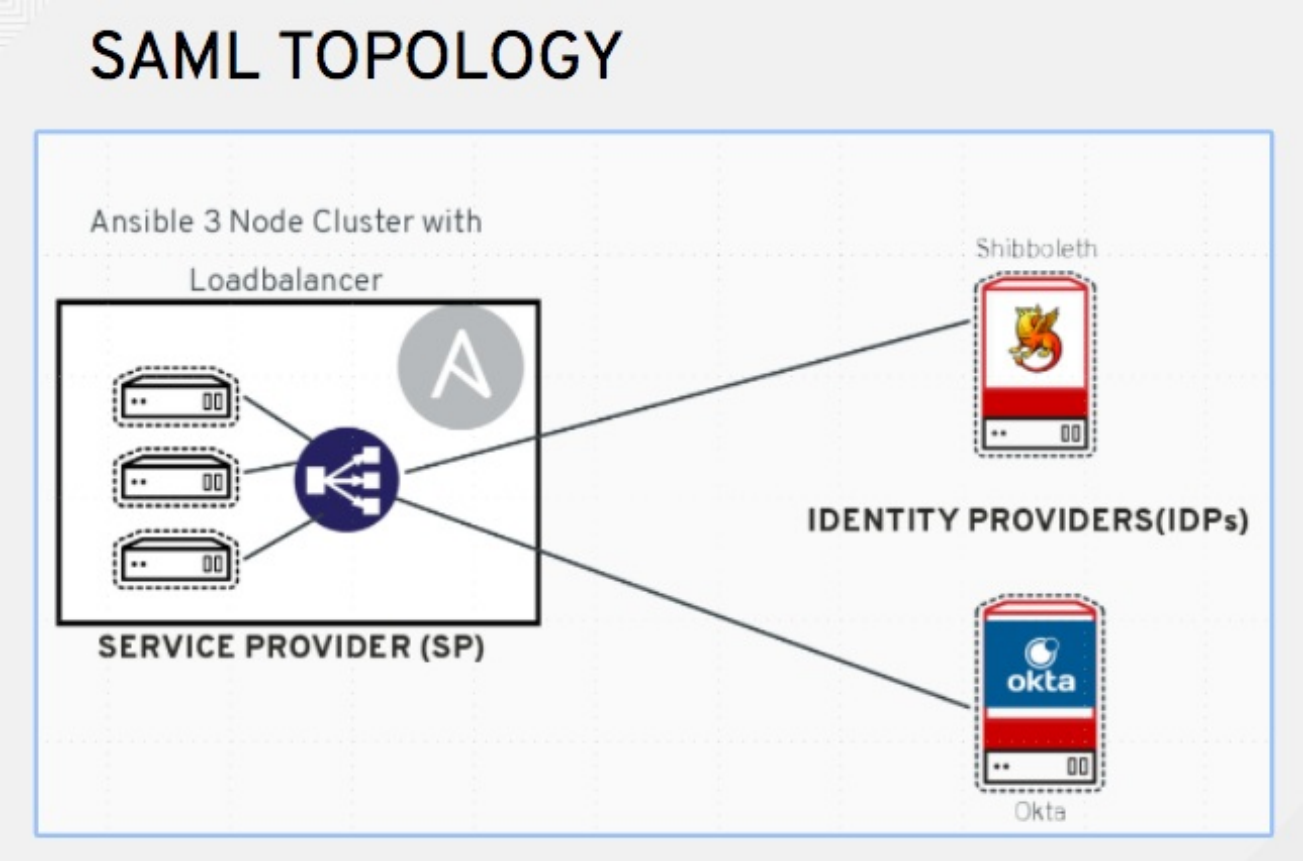
手順

1. ナビゲーションパネルから **Settings** を選択します。
2. **Authentication** オプションのリストから **RADIUS settings** を選択します。
3. **Edit** をクリックし、**RADIUS Server** フィールドに RADIUS サーバーのホストまたは IP を入力します。このフィールドを空白のままにすると、RADIUS 認証は無効になります。
4. 次の 2 つのフィールドには、ポートとシークレットの情報を入力します。
5. **Save** をクリックします。

22.3. SAML 認証

SAML を使用すると、アイデンティティープロバイダー (IdP - シングルサインオンサービスを提供するサーバーシステム) とサービスプロバイダー (今回の場合は Automation Controller) の間で認証および認可データを交換することができます。

Automation Controller は、SAML と通信して Automation Controller ユーザーを認証 (作成/ログイン/ログアウト) するように設定できます。ユーザー、チーム、および組織のメンバーシップは、Automation Controller に対する SAML の応答に埋め込むことができます。



次の手順では、サービスプロバイダーとしての Automation Controller について説明します。RHSSO (keycloak) を通じてユーザーを認証するには、Red Hat Single Sign On Integration with the Automation Controller を参照してください。

手順

1. ナビゲーションパネルから **Settings** を選択します。
2. **Authentication** オプションのリストから **SAML settings** を選択します。

注記

SAML Assertion Consume Service (ACS) URLおよび **SAML Service Provider Metadata URL** フィールドには、事前に情報が入力されており、編集できません。IdP 管理者に連絡し、これらのフィールドに入力されている情報を提供してください。

3. **Edit** をクリックし、**SAML Service Provider Entity ID**に、**Miscellaneous System settings** 画面にある Automation Controller ホストフィールドの **Base URL** を設定します。この情報は、**/api/v2/settings/system** の API の、**CONTROLLER_BASE_URL** 変数で確認できます。**Entity ID** は、個々の Automation Controller クラスターノードのいずれかに設定できますが、サービスプロバイダーの URL に設定することを推奨します。**Base URL** が、ロードバランサーの FQDN (使用されている場合) と一致していることを確認してください。

注記

Base URL は、クラスター内のノードごとに異なります。多くの場合、ロードバランサーは Automation Controller クラスターノードの手前に配置され、単一のエントリーポイントである Automation Controller クラスター FQDN を提供します。SAML サービスプロバイダーは、アウトバウンド接続を確立し、**SAML Service Provider Entity ID** で設定した Automation Controller クラスターノードまたは Automation Controller クラスター FQDN にルーティングできる必要があります。

次の例では、サービスプロバイダーは Automation Controller クラスターであるため、ID は Automation Controller クラスター FQDN に設定されます。

4. Ansible クラスターのサーバー証明書を作成します。通常、Ansible クラスターが設定されている場合、Automation Controller ノードは HTTP トラフィックのみを処理するように設定され、ロードバランサーは SSL 終端ポイントになります。この場合、SSL 証明書は個々の Automation Controller クラスターノードではなく、ロードバランサーに必要です。個々の Automation Controller ノードごとに SSL を有効または無効にできますが、SSL 終端ロードバランサーを使用する場合は無効にする必要があります。証明書の定期的な更新を回避するには、有効期限のない自己署名証明書を使用します。こうすることで、証明書の更新を忘れた場合でも認証が失敗しません。

注記

SAML Service Provider Public Certificateフィールドには、**-----BEGIN CERTIFICATE-----** や **-----END CERTIFICATE-----** を含む、証明書全体を含める必要があります。

証明書で CA バンドルを使用する場合には、このフィールドにバンドル全体の情報を追加してください。

例

```
-----BEGIN CERTIFICATE-----
... cert text ...
-----END CERTIFICATE-----
```

5. コントローラーをサービスプロバイダーとして使用するために任意の秘密鍵を作成して、**SAML Service Provider Private Key**フィールドに入力します。

例

```
-----BEGIN PRIVATE KEY-----
... key text ...
-----END PRIVATE KEY-----
```

6. **SAML Service Provider Organization Info**フィールドで、SSO プロセス中に Automation Controller クラスターに関する詳細を IdP に提供します。

```
{
  "en-US": {
    "url": "http://www.example.com",
    "displayname": "Example",
    "name": "example"
  }
}
```

重要

Automation Controller 内で SAML を正しく設定するには、これらのフィールドに入力する必要があります。

7. **SAML Service Provider Technical Contact**フィールドで、技術担当者の連絡先情報を IdP に提供します。このフィールドの内容は削除しないでください。

```
{
"givenName": "Some User",
"emailAddress": "suser@example.com"
}
```

8. **SAML Service Provider Support Contact**フィールドで、サポートの連絡先情報を IdP に提供します。このフィールドの内容は削除しないでください。

```
{
"givenName": "Some User",
"emailAddress": "suser@example.com"
}
```

9. **SAML Enabled Identity Providers** フィールドに、リストされている各 IdP への接続方法に関する情報を入力します。次の例は、Automation Controller が想定する SAML 属性を示しています。

```
Username(urn:oid:0.9.2342.19200300.100.1.1)
Email(urn:oid:0.9.2342.19200300.100.1.3)
FirstName(urn:oid:2.5.4.42)
LastName(urn:oid:2.5.4.4)
```

これらの属性が不明な場合は、既存の SAML 属性を **Username**、**Email**、**FirstName**、および **LastName** にマッピングします。

IdP ごとに必要なキーを設定します。

- **attr_user_permanent_id** - ユーザーの一意の識別子。IdP から送信された属性のいずれかと一致するように設定できます。Automation Controller ノードに **SAML:nameid** 属性が送信される場合、通常は **name_id** に設定されます。ユーザー名属性またはカスタムの一意の識別子にすることができます。

- **entity_id** - IdP 管理者によって提供されたエンティティー ID。管理者は Automation Controller の SAML プロファイルを作成し、一意の URL を生成します。

- **url** - シングルサインオン (SSO) が有効な場合に Automation Controller がユーザーをリダイレクトする SSO の URL。

- **x509_cert** - IdP で作成した SAML プロファイルから生成される、IdP 管理者が提供する証明書。**---BEGIN CERTIFICATE---** ヘッダーと **---END CERTIFICATE---** ヘッダーを削除し、改行のない 1 つの文字列として証明書を入力します。
  複数の SAML IdP がサポートされています。一部の IdP は、デフォルトの OID とは異なる属性名を使用してユーザーデータを提供する場合があります。SAML NameID は、一部の IdP が使用する特別な属性で、サービスプロバイダー (Automation Controller クラスター) に一意のユーザー識別子を伝えるために使用されます。使用する場合は、次の例に示すように、**attr_user_permanent_id** を **name_id** に設定します。他の属性名は IdP ごとにオーバーライドできます。

```
"myidp": {
  "entity_id": "https://idp.example.com",
  "url": "https://myidp.example.com/sso",
  "x509cert": ""
},
"onelogin": {
  "entity_id": "https://app.onelogin.com/saml/metadata/123456",
  "url": "https://example.onelogin.com/trust/saml2/http-post/sso/123456",
"x509cert": "",
  "attr_user_permanent_id": "name_id",
  "attr_first_name": "User.FirstName",
  "attr_last_name": "User.LastName",
  "attr_username": "User.email",
  "attr_email": "User.email"
  }
}
```

警告

別のユーザー (SAML 以外のユーザーを含む) とメールアドレスを共有する SAML ユーザーを作成しないでください。そのようなユーザーを作成すると、アカウントがマージされます。システム管理者の場合も同じ動作となることに注意してください。したがって、システム管理者と同じメールアドレスを使用して SAML ログインを行うと、システム管理者権限でのログインが可能になります。これを回避するには、SAML マッピングに基づいて管理者権限を削除 (または追加) します。

10. オプション: **SAML Organization Map** を提供します。詳細は、組織マッピング および チームマッピング を参照してください。

11. ユーザーが Automation Controller にログインするときに、ユーザーに関連付けるチームおよび組織のメンバーシップを含む特定の属性を検索するように Automation Controller を設定できます。属性名は、**SAML Organization Attribute Mapping** フィールドと **SAML Team Attribute Mapping** フィールドで定義します。

SAML 組織属性マッピングの例

以下の例は、属性 **member-of** にユーザー組織メンバーシップを埋め込む SAML 属性です。

```
<saml2:AttributeStatement>
   <saml2:Attribute FriendlyName="member-of" Name="member-of"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
      <saml2:AttributeValue>Engineering</saml2:AttributeValue>
      <saml2:AttributeValue>IT</saml2:AttributeValue>
      <saml2:AttributeValue>HR</saml2:AttributeValue>
      <saml2:AttributeValue>Sales</saml2:AttributeValue>
   </saml2:Attribute>
   <saml2:Attribute FriendlyName="admin-of" Name="admin-of"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
      <saml2:AttributeValue>Engineering</saml2:AttributeValue>
   </saml2:Attribute>
</saml2:AttributeStatement>
```

対応する Automation Controller の設定は次のとおりです。

```
{
  "saml_attr": "member-of",
  "saml_admin_attr": "admin-of",
  "remove": true,
  "remove_admins": false
}
```

- **saml_attr**: 組織アレイが含まれる SAML 属性名。**remove** を **true** に設定すると、組織のリストにユーザーを追加する前に、すべての組織からユーザーを削除します。ユーザーを組織にそのまま所属させた状態で SAML 属性の組織に追加する場合は、**remove** を **false** に設定します。

- **saml_admin_attr**: **saml_attr** 属性と同様ですが、この属性は組織のメンバーシップを指定するのではなく、管理者の組織パーミッションを指定します。

SAML チーム属性マッピングの例

次の例は、リストにチームメンバーシップを含む別の SAML 属性です。

```
<saml:AttributeStatement>
    <saml:Attribute
      xmlns:x500="urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500"
      x500:Encoding="LDAP"
      NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
      Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.1"
      FriendlyName="eduPersonAffiliation">
      <saml:AttributeValue
        xsi:type="xs:string">member</saml:AttributeValue>
      <saml:AttributeValue
        xsi:type="xs:string">staff</saml:AttributeValue>
      </saml:Attribute>
</saml:AttributeStatement>
{
    "saml_attr": "eduPersonAffiliation",
    "remove": true,
    "team_org_map": [
    {
```

```
        "team": "member",
        "organization": "Default1"
    },
    {
        "team": "staff",
        "organization": "Default2"
    }
  ]
}
```

- **saml_attr**: チームアレイが含まれる SAML 属性名です。
- **delete**: **remove** を **true** に設定すると、チームのリストにユーザーを追加する前に、すべてのチームからユーザーを削除します。ユーザーをチームにそのまま所属させた状態で SAML 属性のチームに追加する場合は、**remove** を **false** に設定します。
- **team_org_map**: コントローラーチーム → Automation Controller 組織のマッピングを定義する、**{ "team": "<AWX Team Name>", "organization": "<AWX Org Name>" }** 形式のディクショナリーのアレイ。Automation Controller の複数の組織に同じ名前のチームが存在できるため、このマッピングが必要になります。このマッピングがないと、SAML 属性にリストされているチームがどの組織に属しているか曖昧になります。
  エイリアスを作成して、**SAML Team Attribute Mapping** フィールドでチームと組織の両方をオーバーライドできます。この方法は、次の例に示すように、SAML バックエンドが複雑なグループ名を送信する場合に役立ちます。

```
{
 "remove": false,
 "team_org_map": [
  {
   "team": "internal:unix:domain:admins",
   "organization": "Default",
   "team_alias": "Administrators"
  },
  {
   "team": "Domain Users",
   "organization_alias": "OrgAlias",
   "organization": "Default"
  }
 ],
 "saml_attr": "member-of"
}
```

  ユーザーが認証されると、Automation Controller は組織とチームのエイリアスを作成します。

12. オプション: **SAML Team Map** フィールドに、チームのメンバーシップマッピングを入力します。詳細は、組織マッピング および チームマッピング を参照してください。
13. オプション: **SAML Security Config** フィールドに、セキュリティー設定を入力します。このフィールドは、API の **SOCIAL_AUTH_SAML_SECURITY_CONFIG** フィールドに相当します。詳細は、OneLogin の SAML Python Toolkit を参照してください。
    Automation Controller は、ユーザーが SAML 経由でログインするときに **python-social-auth** ライブラリーを使用します。このライブラリーは **python-saml** ライブラリーに依存し、次の 2 つのオプションのフィールド (**SAML Service Provider extra configuration data**と **SAML IDP to extra_data attribute mapping**) で設定を利用可能にします。

- SAML Service Provider extra configuration dataフィールドは、API の **SOCIAL_AUTH_SAML_SP_EXTRA** に相当します。詳細は、OneLogin の SAML Python Toolkit を参照して、有効なサービスプロバイダーの追加のパラメーター (**SP_EXTRA**) を確認してください。

- SAML IDP to extra_data attribute mappingフィールドは、API の **SOCIAL_AUTH_SAML_EXTRA_DATA** に相当します。詳細は、Python の SAML に関する Advanced Settings ドキュメントを参照してください。

- SAML User Flags Attribute Mappingフィールドを使用すると、SAML ロールと属性を特別なユーザーフラグにマッピングできます。このフィールドでは次の属性が有効です。

  - **is_superuser_role**: ユーザーにスーパーユーザーフラグを付与する 1 つ以上の SAML ロールを指定します。

  - **is_superuser_attr**: ユーザーにスーパーユーザーフラグを付与する SAML 属性を指定します。

  - **is_superuser_value**: ユーザーがスーパーユーザーになるために求められる **is_superuser_attr** に必要な値を 1 つ以上指定します。

  - **remove_superusers**: ユーザーのスーパーユーザーフラグを削除する必要があるかどうかを示すブール値です。デフォルトは **true** です。

  - **is_system_auditor_role**: ユーザーにシステム監査人フラグを付与する 1 つ以上の SAML ロールを指定します。

  - **is_system_auditor_attr**: ユーザーにシステム監査人フラグを付与する SAML 属性を指定します。

  - **is_system_auditor_value**: ユーザーがシステム監査人になるために求められる **is_system_auditor_attr** に必要な値を 1 つ以上指定します。

  - **remove_system_auditors**: ユーザーの **system_auditor** フラグを削除する必要があるかどうかを示すブール値です。デフォルトは **true** です。
    role フィールドと **value** フィールドはリストであり、'OR' ロジックが使用されます。2 つのロール ["Role 1", "Role 2"] を指定し、SAML ユーザーがいずれかのロールを持っている場合、このロジックでは、ユーザーがフラグに必要なロールを持っていると見なされます。これは、**value** フィールドでも同様です。["Value 1", "Value 2"] を指定し、SAML ユーザーの属性がどちらかの値を持っている場合、このロジックでは属性値が一致したと見なされます。

    superuser または **system_auditor** のいずれかに **role** と **attr** を指定した場合、**attr** の設定がロールより優先されます。システム管理者とシステム監査者のロールは、SAML ユーザーのログイン時に評価されます。これらのロールのいずれかを SAML 設定ではなく UI で SAML ユーザーに付与した場合、**remove** フラグを **false** に設定していない限り、当該ロールはユーザーの次回のログイン時に削除されます。**remove** フラグが **false** の場合、SAML アダプターは対応するフラグをユーザーから削除できません。次の表に、ロジックがどのように機能するかを示します。

1つ以上のロールがある	attr がある	1つ以上の attr Values がある	フラグを削除する	以前のフラグがある	フラグが立てられる
いいえ	いいえ	該当なし	True	False	いいえ
いいえ	いいえ	該当なし	False	False	いいえ
いいえ	いいえ	該当なし	True	True	いいえ
いいえ	いいえ	該当なし	False	True	はい
はい	いいえ	該当なし	True	False	はい
はい	いいえ	該当なし	False	False	はい
はい	いいえ	該当なし	True	True	はい
はい	いいえ	該当なし	False	False	はい
いいえ	はい	はい	True	True	はい
いいえ	はい	はい	True	False	はい
いいえ	はい	はい	False	False	はい
いいえ	はい	はい	True	True	はい
いいえ	はい	はい	False	True	はい
いいえ	はい	いいえ	True	False	いいえ
いいえ	はい	いいえ	False	False	いいえ
いいえ	はい	いいえ	True	True	いいえ
いいえ	はい	いいえ	False	True	はい
いいえ	はい	未設定	True	False	はい
いいえ	はい	未設定	False	False	はい
いいえ	はい	未設定	True	True	はい
いいえ	はい	未設定	False	True	はい

1つ以上のロールがある	attr がある	1つ以上の attr Values がある	フラグを削除する	以前のフラグがある	フラグが立てられる
はい	はい	はい	True	False	はい
はい	はい	はい	False	False	はい
はい	はい	はい	True	True	はい
はい	はい	はい	False	True	はい
はい	はい	いいえ	True	False	いいえ
はい	はい	いいえ	False	False	いいえ
はい	はい	いいえ	True	True	いいえ
はい	はい	いいえ	False	True	はい
はい	はい	未設定	True	False	はい
はい	はい	未設定	False	False	はい
はい	はい	未設定	True	True	はい
はい	はい	未設定	False	True	はい

SAML ユーザーが Automation Controller に対して認証されるたびに、これらのチェックが実行され、必要に応じてユーザーフラグが変更されます。UI 内で SAML ユーザーに対して **System Administrator** または **System Auditor** を設定した場合、SAML アダプターは前述のルールに基づいて UI 設定をオーバーライドします。SAML ユーザーのログイン時に SAML ユーザーのユーザーフラグを削除しないようにするには、**remove_** フラグを **false** に設定します。**remove** フラグを **false** に設定すると、UI、API、または SAML アダプター経由で **true** に設定したユーザーフラグは削除されません。ただし、ユーザーがフラグを持たず、前述のルールによってフラグを追加する必要があると判断された場合は、フラグが **false** であってもフラグが追加されます。

例

```
{
    "is_superuser_attr": "blueGroups",
    "is_superuser_role": ["is_superuser"],
    "is_superuser_value": ["cn=My-Sys-
Admins,ou=memberlist,ou=mygroups,o=myco.com"],
    "is_system_auditor_attr": "blueGroups",
    "is_system_auditor_role": ["is_system_auditor"],
```

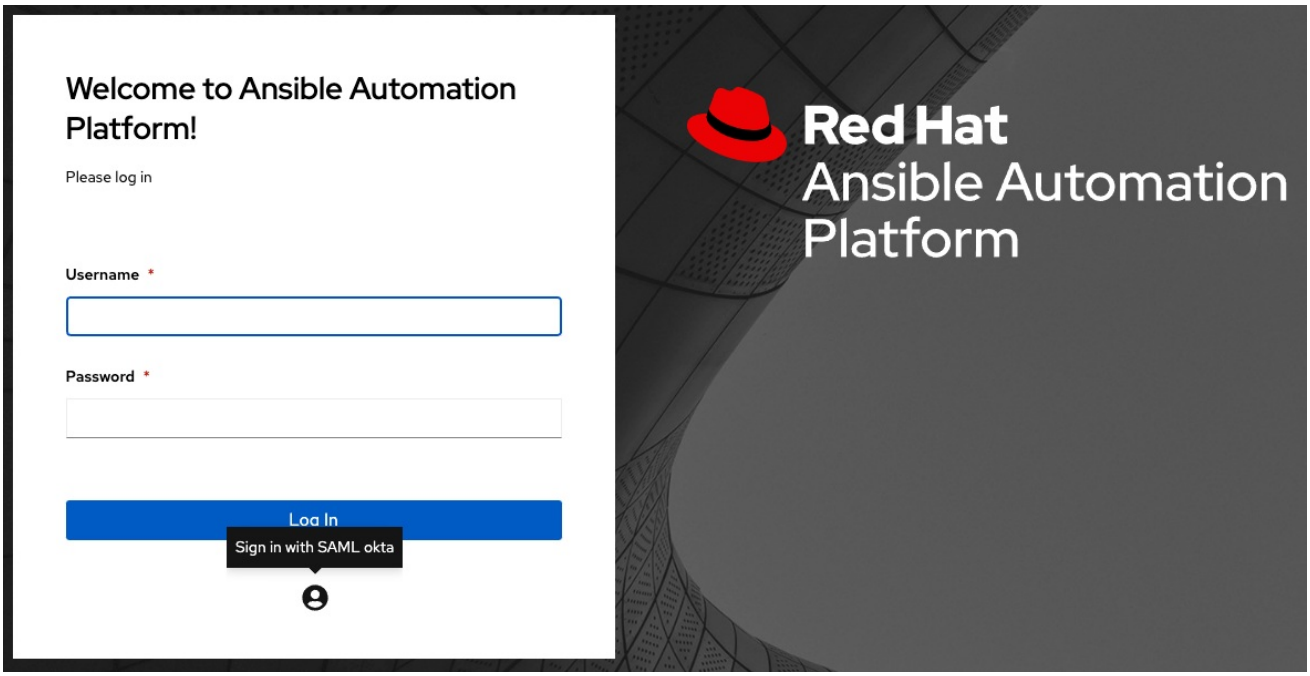
```
    "is_system_auditor_value": ["cn=My-
Auditors,ou=memberlist,ou=mygroups,o=myco.com"]
}
```

14. **Save** をクリックします。

検証

認証が正しく設定されたことを確認するには、**SAML Service Provider Metadata URL**にある自動生成された URL をブラウザーにロードします。XML 出力が得られない場合は、正しく設定されていません。

あるいは、Automation Controller からログアウトします。ログイン画面に、Automation Controller への別のログイン方法として SAML ロゴが表示されます。



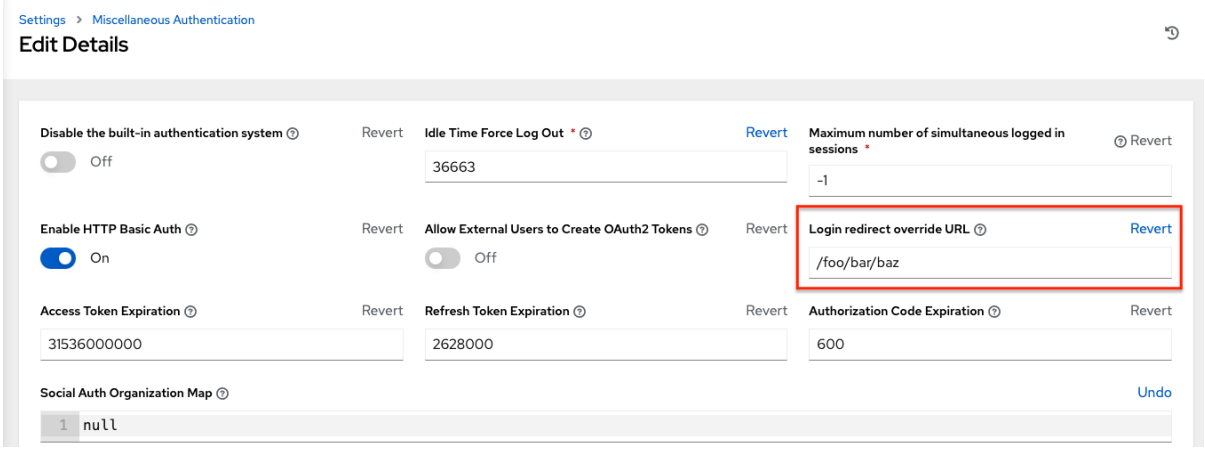
22.3.1. 透過的な SAML ログインの設定

透過的なログインを機能させるには、まず IdP-initiated ログインを機能させる必要があります。

手順

1. IdP の **RelayState** を、**SAML Enabled Identity Providers** フィールドの IdP 定義のキーに設定します。

2. これが機能したら、**Settings** メニューの **Miscellaneous Authentication** 設定にある **Login redirect override URL** フィールドを使用して、デフォルトの Automation Controller ログインページ以外の場所に、ログインしていないユーザーをリダイレクトする URL を指定します。透過的な SAML ログインでは、次の例に示すように、これを **/sso/login/saml/?idp=<name-of-your-idp>** に設定する必要があります。



注記

この例は一般的な IdP 形式を示していますが、特定のケースではこの形式は正しくない可能性があります。透過的なリダイレクト URL はすべての IdP で同じではないため、IdP に問い合わせて正しい URL を取得することが必要な場合があります。

3. 透過的な SAML ログインを設定した後に、ローカルの認証情報または別の SSO を使用してログインするには、**https://<your-tower-server>/login** に直接移動します。これにより、SSO 認証オプションを含む標準の Automation Controller ログインページが表示され、設定した方法でログインできるようになります。

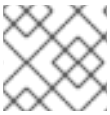
22.3.2. SAML のロギングの有効化

LDAP のロギングを有効にするのと同じ方法で、SAML アダプターのメッセージのロギングを有効にできます。

詳細は、LDAP のロギングの有効化 セクションを参照してください。

22.4. TACACS PLUS 認証

Terminal Access Controller Access-Control System Plus (TACACS+) は、ネットワークアクセス制御のためのリモート認証および関連サービスを、中央サーバーを介して処理するプロトコルです。TACACS+ は、認証、認可、アカウンティング (AAA) のサービスを提供します。このサービスでは、Automation Controller を認証のソースとして使用するように設定できます。



注記

この機能は非推奨となり、今後のリリースで削除されます。

手順

1. ナビゲーションパネルから **Settings** を選択します。
2. **Authentication** オプションのリストから **TACACs+ settings** を選択します。
3. **Edit** をクリックし、次の情報を入力します。
   - **TACACS+ Server**: 認証に使用する TACACS+ サーバーのホスト名または IP アドレスを指定します。このフィールドを空白のままにすると、TACACS+ 認証は無効になります。

- **TACACS+ Port**: TACACS+ はデフォルトでポート 49 を使用するので、事前に入力されています。
- **TACACS+ Secret**: TACACS+ 認証サーバーのシークレットキーです。
- **TACACS+ Auth Session Timeout** セッションタイムアウトの値 (秒単位) です。デフォルトは 5 秒です。
- **TACACS+ Authentication Protocol**: TACACS+ クライアントが使用するプロトコルです。オプションは **ascii** または **pap** です。

4. **Save** をクリックします。

22.5. 汎用 OIDC 認証

OpenID Connect (OIDC) は OAuth 2.0 フレームワークを使用します。これにより、サードパーティーのアプリケーションがアイデンティティーを検証し、基本的なエンドユーザー情報を取得できるようになります。OIDC と SAML の主な違いは、SAML にはサービスプロバイダー (SP) と IdP 間の信頼関係があるのに対し、OIDC はセキュリティートークンの取得に使用されるチャネル (HTTPS) との信頼関係を確立する点にあります。Automation Controller で OIDC をセットアップするために必要な認証情報を取得するには、OIDC をサポートする任意の IdP のドキュメントを参照してください。

手順

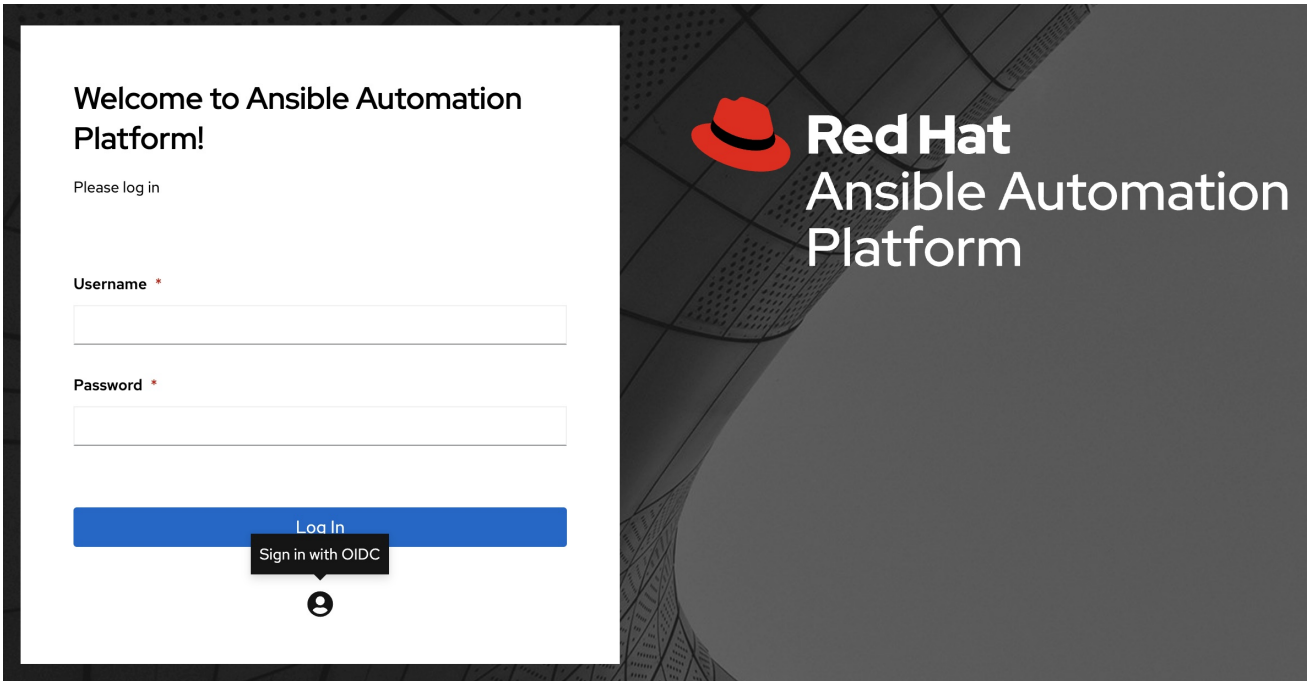
1. ナビゲーションパネルから **Settings** を選択します。
2. **Authentication** オプションのリストから **Generic OIDC settings** を選択します。
3. **Edit** をクリックし、次の情報を入力します。
   - **OIDC Key**: サードパーティー IdP からのクライアント ID。
   - **OIDC Secret**: IdP からのクライアントシークレット。
   - **OIDC Provider URL**: OIDC プロバイダーの URL。
   - **Verify OIDC Provider Certificate**: トグルを使用して、OIDC プロバイダーの SSL 証明書の検証を有効または無効にします。
4. **Save** をクリックします。

注記

現在、OIDC のチームと組織のマッピングはサポートされていません。OIDC アダプターは認証のみを行い、認可は行いません。ユーザーが本人であるかどうかの認証のみが可能です。このユーザーが実行可能な内容についての認可は行いません。汎用 OIDC を設定すると、2 つの異なるソースから発生した同じユーザー ID を区別するために、ID またはキーが追加された UserID が作成され、同じユーザー ID でも異なるユーザーと見なされます。したがって、1 つ目の ID はユーザー名のみとなり、2 つ目はユーザー名に乱数が追加されます。

検証

認証が正しく設定されたことを確認するには、Automation Controller からログアウトします。ログイン画面に、Automation Controller への別のログイン方法として OIDC ロゴが表示されます。


Welcome to Ansible Automation Platform!
Please log in
Username *
Password *
Log In
Sign in with OIDC
Red Hat
Ansible Automation Platform

第23章 LDAP 認証

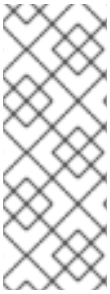
管理者は、Automation Controller ユーザーのアカウント認証情報のソースとして、**Lightweight Directory Access Protocol** (LDAP) を使用します。ユーザー認証は提供されますが、ユーザーのパーミッションと認証情報は同期されません。組織のメンバーシップとチームのメンバーシップは、組織管理者が同期できます。

23.1. LDAP 認証のセットアップ

設定すると、LDAP のユーザー名とパスワードを使用してログインするユーザーに対して、Automation Controller のアカウントが自動的に作成されます。このユーザーを、通常のユーザーまたは組織管理者として自動的に組織に配置できます。

ユーザーインターフェイス (ローカル) で作成されたユーザーは、別の認証ソリューションを使用して初めて Automation Controller にログインするユーザーよりも優先されます。LDAP などの別の認証方法で再利用する場合は、ローカルユーザーを削除する必要があります。

LDAP ログインで作成されたユーザーは、自分のユーザー名や姓名を変更できず、ローカルのパスワードも設定できません。他のフィールド名の編集を制限するように設定することも可能です。



注記

接続先の LDAP サーバーに自己署名証明書、または企業内部の認証局 (CA) で署名された証明書がある場合は、その CA 証明書をシステムの信頼された CA に追加する必要があります。そうしないと、LDAP サーバーに接続する際に、証明書の発行者が認識されないというエラーが発生します。詳細は、LDAPS 統合向けに Automation Controller に認証局をインポートする手順 を参照してください。プロンプトが表示されたら、Red Hat の顧客認証情報を使用してログインしてください。

手順

1. LDAP 構造全体の読み取り権限があるユーザーを LDAP に作成します。
2. **ldapsearch** コマンドを使用して、LDAP サーバーへのクエリーが正常に実行できるかどうかをテストします。このツールは、Automation Controller のシステムコマンドラインからインストールでき、他の Linux および OSX システムを使用してもインストール可能です。

   例

   ```
   ldapsearch -x  -H ldap://win -D "CN=josie,CN=Users,DC=website,DC=com" -b
   "dc=website,dc=com" -w Josie4Cloud
   ```

   この例では、**CN=josie,CN=users,DC=website,DC=com** が接続ユーザーの識別名です。

   注記

   ldapsearch ユーティリティーは、Automation Controller とともに自動的にプリインストールされるわけではありません。ただし、**openldap-clients** パッケージからインストールできます。

3. ナビゲーションパネルから、Automation Controller UI の **Settings** を選択します。
4. **Authentication** オプションのリストで **LDAP settings** を選択します。

LDAP サーバーごとに複数の LDAP 設定を行う必要はありませんが、このページから複数の LDAP サーバーを設定できます。そのように設定しない場合は、サーバーを **Default** のままにします。

同等の API エンドポイントは、**AUTH_LDAP_1_***、**AUTH_LDAP_2_***、**AUTH_LDAP_5_*** のように **AUTH_LDAP_*** を繰り返し表示し、サーバーの割り当てを示します。

5. LDAP サーバーアドレスを入力または変更するには、**Edit** をクリックし、テキストフィールドに事前に入力されているものと同じ形式を使用して **LDAP Server URI** フィールドに入力します。

注記

複数の LDAP サーバーを指定するには、各サーバーをスペースまたはコンマで区切ります。 ⓘ アイコンをクリックして、正しい構文とルールに準拠してください。

6. バインディングユーザーに使用するパスワードを **LDAP Bind Password** テキストフィールドに入力します。LDAP 変数の詳細は、Ansible Automation Hub 変数 を参照してください。

7. **LDAP Group Type** リストで、グループタイプをクリックして選択します。Automation Controller がサポートする LDAP グループタイプは、基盤となる django-auth-ldap ライブラリー を使用します。選択したグループタイプのパラメーターを指定するには、ステップ 15 を参照してください。

8. **LDAP Start TLS** は、デフォルトで無効になっています。LDAP 接続が SSL/TLS を使用していない場合に TLS を有効にするには、トグルを **On** に設定します。

9. **LDAP Bind DN** テキストフィールドに識別名を入力して、Automation Controller が LDAP サーバーへの接続 (バインド) に使用するユーザーを指定します。

   - 名前がキー **sAMAccountName** に保存されている場合、**LDAP User DN Template** は **(sAMAccountName=%(user)s)** から入力されます。Active Directory はユーザー名を **sAMAccountName** に保存します。OpenLDAP の場合、キーは **uid** で、行は **(uid=%(user)s)** になります。

10. **LDAP Require Group** フィールドに識別グループ名を入力して、当該グループ内のユーザーが Automation Controller にアクセスできるようにします。このとき、テキストフィールドに表示されているもの (**CN=controller Users,OU=Users,DC=website,DC=com**) と同じ形式を使用します。

11. **LDAP Deny Group** フィールドに識別グループ名を入力して、当該グループ内のユーザーが Automation Controller にアクセスできないようにします。このとき、テキストフィールドに表示されているものと同じ形式を使用します。

12. **LDAP User Search** フィールドに、認証時にユーザーを検索する場所を入力します。このとき、テキストフィールドに表示されているものと同じ形式を使用します。この例では、以下を使用します。

```
[
"OU=Users,DC=website,DC=com",
"SCOPE_SUBTREE",
"(cn=%(user)s)"
]
```

最初の行では、LDAP ツリーの中でユーザーを検索する場所を指定します。上記の例では、**DC=website,DC=com** から開始して再帰的にユーザーの検索が実行されます。

2 行目は、ユーザーを検索する範囲を指定します。

- **SCOPE_BASE**: この値は、基本 DN でエントリーのみを検索して、このエントリーだけが返されるように指定する時に使用します。
- **SCOPE_ONELEVEL**: この値は、基本 DN の 1 つ下のレベルにあるエントリーをすべて検索するように指定する時に使用します。ここでは、基本 DN や、基本 DN の 1 つ下のレベルよりも下のレベルにあるエントリーは検索には含めません。
- **SCOPE_SUBTREE**: この値は、指定した基本 DN も含めて、全レベルの全エントリーを検索するように指定する際に使用します。
3 行目は、ユーザー名を保存するキー名を指定します。

  複数の検索クエリーの場合、正しい構文は以下のとおりです。

```
[
  [
  "OU=Users,DC=northamerica,DC=acme,DC=com",
  "SCOPE_SUBTREE",
  "(sAMAccountName=%(user)s)"
  ],
  [
  "OU=Users,DC=apac,DC=corp,DC=com",
  "SCOPE_SUBTREE",
  "(sAMAccountName=%(user)s)"
  ],
  [
  "OU=Users,DC=emea,DC=corp,DC=com",
  "SCOPE_SUBTREE",
  "(sAMAccountName=%(user)s)"
  ]
]
```

13. **LDAP Group Search** テキストフィールドで、検索するグループと検索方法を指定します。この例では、以下を使用します。

```
[
"dc=example,dc=com",
"SCOPE_SUBTREE",
"(objectClass=group)"
 ]
```

- 最初の行は、グループを検索すべき基本 DN を指定します。
- 2 行目は、ユーザーディレクティブと同じで、範囲を指定します。
- 3 行目は、使用する LDAP において、グループオブジェクトの **objectClass** が何かを指定します。

14. **LDAP User Attribute Map** テキストフィールドに、ユーザー属性を入力します。この例では、以下を使用します。

```
{
```

```
"first_name": "givenName",
"last_name": "sn",
"email": "mail"
}
```

上記の例では、キー **sn** で姓によってユーザーを取得します。ユーザーに対して同じ LDAP クエリーを使用して、ユーザーがどのキーに保存されているかを判断できます。

選択した **LDAP Group Type** に応じて、異なるパラメーターが **LDAP Group Type Parameters** フィールドで使用できます。**LDAP_GROUP_TYPE_PARAMS** は、Automation Controller によって **kwargs** に変換されて、選択した **LDAP Group Type** クラスに渡されるディクショナリーです。どの **LDAP Group Type** でも使用される共通パラメーターが 2 つあります。**name_attr** と **member_attr** です。**name_attr** のデフォルトは cn で、 **member_attr** のデフォルトは member です。

```
{"name_attr": "cn", "member_attr": "member"}
```

特定の **LDAP Group Type** が必要とするパラメーターを確認するには、django_auth_ldap ドキュメントのクラス **init** パラメーターに関する箇所を参照してください。

15. **LDAP User Flags by Group** テキストフィールドにユーザープロファイルフラグを入力します。次の例では、LDAP ユーザーを "スーパーユーザー" および "監査人" として設定する構文を使用しています。

```
{
"is_superuser": "cn=superusers,ou=groups,dc=website,dc=com",
"is_system_auditor": "cn=auditors,ou=groups,dc=website,dc=com"
}
```

16. **LDAP Organization Map** および **LDAP Team Map** のマッピングフィールドへの入力に関する詳細は、LDAP 組織およびチームのマッピング セクションを参照してください。

17. **Save** をクリックします。

注記

Automation Controller はユーザーを積極的に同期しませんが、初回のログイン時にユーザーが作成されます。LDAP 認証に関連するパフォーマンスを向上させるには、ログインごとのLDAP属性更新の防止 を参照してください。

23.1.1. LDAP 組織およびチームのマッピング

LDAP 属性に基づいて、どのユーザーをどの Automation Controller 組織に配置するかを制御できます (組織の管理者、ユーザー、LDAP グループの間のマッピング)。

キーは組織名です。組織が存在しない場合は作成されます。値は、各組織のメンバーシップのオプションを定義するディクショナリーとなります。どのグループが自動的に組織のユーザーになり、どのグループが組織を管理できるかを、組織ごとに指定できます。

admins: **none**、**true**、**false**、**string**、または文字列の **list/tuple**:

- **none** の場合、組織管理者は LDAP 値に基づいて更新されません。
- **true** の場合、LDAP 内のすべてのユーザーが組織の管理者として自動的に追加されます。

- **false** の場合、いずれの LDAP ユーザーも組織の管理者として自動的に追加されません。
- 文字列または文字列のリストの場合は、グループ DN を指定します。指定したグループに一致する場合、組織に追加されます。

remove_admins: True/False。デフォルトは False です。

- **true** の場合、指定されたグループのメンバーではないユーザーは、組織の管理者リストから削除されます。

users: **none**、**true**、**false**、**string**、または文字列の **list/tuple**。管理者と同じルールが適用されます。

remove_users: **true** または **false**。デフォルトは **false** です。管理者と同じルールが適用されます。

例

```
{
"LDAP Organization": {
  "admins": "cn=engineering_admins,ou=groups,dc=example,dc=com",
  "remove_admins": false,
  "users": [
    "cn=engineering,ou=groups,dc=example,dc=com",
    "cn=sales,ou=groups,dc=example,dc=com",
    "cn=it,ou=groups,dc=example,dc=com"
  ],
  "remove_users": false
},
"LDAP Organization 2": {
  "admins": [
    "cn=Administrators,cn=Builtin,dc=example,dc=com"
  ],
  "remove_admins": false,
  "users": true,
"remove_users": false
}
}
```

ユーザーと LDAP グループをマッピングする場合、キーはチーム名であり、存在しない場合は作成されます。値は、各チームのメンバーシップに関するオプションのディクショナリーとなります。各値には、次のパラメーターを含めることができます。

organization: **string**.チームが所属する組織の名前です。組織とチームの組み合わせが存在しない場合は、チームが作成されます。組織が存在しない場合には、先に組織が作成されます。

users: **none**、**true**、**false**、**string**、または文字列の **list/tuple**:

- **none** の場合、チームメンバーは更新されません。
- **true** または **false** の場合、すべての LDAP ユーザーがチームメンバーとして追加または削除されます。
- 文字列または文字列のリストでグループ DN を指定する場合、ユーザーがこれらのグループの**いずれか** のメンバーであれば、そのユーザーはチームメンバーとして追加されます。

remove: **true** または **false**。デフォルトは **false** です。**true** の場合、指定されたグループのメンバーではないユーザーは、チームから削除されます。

例

```
{
"LDAP Engineering": {
  "organization": "LDAP Organization",
  "users": "cn=engineering,ou=groups,dc=example,dc=com",
  "remove": true
},
"LDAP IT": {
  "organization": "LDAP Organization",
  "users": "cn=it,ou=groups,dc=example,dc=com",
  "remove": true
},
"LDAP Sales": {
  "organization": "LDAP Organization",
  "users": "cn=sales,ou=groups,dc=example,dc=com",
  "remove": true
}
}
```

23.1.2. LDAP のロギングの有効化

LDAP のロギングを有効にするには、**Settings** 設定ウィンドウでレベルを **DEBUG** に設定する必要があります。

手順

1. ナビゲーションパネルから **Settings** を選択します。
2. **System** オプションのリストから **Logging settings** を選択します。
3. **Edit** をクリックします。
4. **Logging Aggregator Level Threshold** フィールドを **DEBUG** に設定します。
5. **Save** をクリックします。

23.1.3. ログインごとのLDAP属性更新の防止

デフォルトでは、LDAP ユーザーが認証されると、すべてのユーザー関連の属性がログインのたびにデータベースで更新されます。一部の環境では、パフォーマンスの問題のためにこの操作をスキップできます。これを回避するには、オプション **AUTH_LDAP_ALWAYS_UPDATE_USER** を無効にします。

警告

LDAP ユーザーの属性を更新しない場合は、このオプションを **false** に設定してください。属性は、ユーザーが初めて作成されたときにのみ更新されます。

手順

1. **/etc/tower/conf.d/custom-ldap.py** に次の内容のカスタムファイルを作成します。複数のノードがある場合は、すべてのノードで実行します。

```
AUTH_LDAP_ALWAYS_UPDATE_USER = False
```

2. すべてのノードで Automation Controller を再起動します。

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

このオプションを **False** に設定すると、LDAP ユーザーの属性に対する変更は Automation Controller にプッシュされません。新しいユーザーが作成され、当該ユーザーの初回ログイン時にその属性がデータベースにプッシュされることに注意してください。

デフォルトでは、LDAP ユーザーはログインするたびにデータベースで属性が更新されます。LDAP 認証情報を使用して複数回実行される Playbook の場合、これらのクエリーを回避できます。

検証

LDAP 認証に関連するクエリーで時間のかかっているものがないか、PostgreSQL を確認してください。

関連情報

詳細は、Django ドキュメントの AUTH_LDAP_ALWAYS_UPDATE_USER を参照してください。

23.1.4. LDAPS 統合向けに Automation Controller に認証局をインポートする手順

LDAP を使用して Automation Controller サーバーへの認証を行うことができますが、認証に LDAPS (SSL/TLS 経由の LDAP) を使用するように変更すると、次のいずれかのエラーで失敗します。

```
2020-04-28 17:25:36,184 WARNING  django_auth_ldap Caught LDAPError while authenticating e079127: SERVER_DOWN({'info': 'error:14090086:SSL routines:ssl3_get_server_certificate:certificate verify failed (unable to get issuer certificate)', 'desc': "Can't contact LDAP server"},)
```

```
2020-06-02 11:48:24,840 WARNING  django_auth_ldap Caught LDAPError while authenticating reinernippes: SERVER_DOWN({'desc': "Can't contact LDAP server", 'info': 'error:14090086:SSL routines:ssl3_get_server_certificate:certificate verify failed (certificate has expired)'},)
```

注記

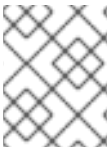
デフォルトでは、**django_auth_ldap** は、LDAPS トランザクションを開始する前に SSL 接続を検証します。**certificate verify failed** エラーが発生した場合、**django_auth_ldap** が証明書を検証できなかったことを意味します。SSL/TLS 接続を検証できない場合、接続の試行は中止されます。

手順

- LDAP CA をインポートするには、次のコマンドを実行します。

```
cp ldap_server-CA.crt /etc/pki/ca-trust/source/anchors/
```

```
update-ca-trust
```



注記

クラスター化されたセットアップ内のすべての Automation Controller ノードで、これら 2 つのコマンドを実行します。

23.1.5. 参照

Active Directory は、クエリーされたオブジェクトがデータベースで利用できない場合に備えて "参照" を使用します。参照は、django LDAP クライアントでは正しく機能しないため、無効にした方が有益です。

/etc/tower/conf.d/custom.py ファイルに次の行を追加して、LDAP 参照を無効にします。

```
AUTH_LDAP_GLOBAL_OPTIONS = {
    ldap.OPT_REFERRALS: False,
}
```

23.1.6. 認証用にデフォルトタイムアウトを変更する手順

Automation Controller UI の **Settings** 画面で、指定したトークンが有効であるデフォルトの時間 (秒単位) を変更できます。

手順

1. ナビゲーションパネルから **Settings** を選択します。
2. **System** オプションのリストから **Miscellaneous Authentication settings** を選択します。
3. **Edit** をクリックします。
4. **Idle Time Force Log Out** テキストフィールドに、タイムアウトの期間を秒単位で入力します。
5. **Save** をクリックします。

注記

Automation Controller にアクセスしてログインで問題が生じた場合は、Web ブラウザーのキャッシュをクリアしてください。一般的にそのような状況では、ブラウザーセッション中に認証トークンがキャッシュされています。続行するにはこれをクリアする必要があります。

第24章 KERBEROS でのユーザー認証

Automation Controller では、**Active Directory** (AD) を使用したユーザー認証 (Kerberos による認証とも呼ばれます) がサポートされます。

24.1. KERBEROS パッケージのセットアップ

まず、Kerberos チケットを正常に生成できるように、Automation Controller で Kerberos パッケージをセットアップします。

次のコマンドを使用してパッケージをインストールします。

```
yum install krb5-workstation
yum install krb5-devel
yum install krb5-libs
```

インストールされたら、**/etc/krb5.conf** ファイルを次のように編集して、AD のアドレス、ドメインなどの情報を指定します。

```
[logging]
 default = FILE:/var/log/krb5libs.log
 kdc = FILE:/var/log/krb5kdc.log
 admin_server = FILE:/var/log/kadmind.log

[libdefaults]
 default_realm = WEBSITE.COM
 dns_lookup_realm = false
 dns_lookup_kdc = false
 ticket_lifetime = 24h
 renew_lifetime = 7d
 forwardable = true

[realms]
 WEBSITE.COM = {
  kdc = WIN-SA2TXZOTVMV.website.com
  admin_server = WIN-SA2TXZOTVMV.website.com
 }

[domain_realm]
 .website.com = WEBSITE.COM
 website.com = WEBSITE.COM
```

設定ファイルが更新されたら、次のコマンドを使用して認証し、有効なトークンを取得します。

```
[root@ip-172-31-26-180 ~]# kinit username
Password for username@WEBSITE.COM:
[root@ip-172-31-26-180 ~]#
```

有効なチケットがあるかどうかを確認します。

```
[root@ip-172-31-26-180 ~]# klist
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: username@WEBSITE.COM
```

```
Valid starting     Expires            Service principal
01/25/23 11:42:56  01/25/23 21:42:53  krbtgt/WEBSITE.COM@WEBSITE.COM
  renew until 02/01/23 11:42:56
[root@ip-172-31-26-180 ~]#
```

有効なチケットがある場合は、すべてが期待どおりに動作していることをコマンドラインから確認できます。

これをテストするには、インベントリーは次のようにする必要があります。

```
[windows]
win01.WEBSITE.COM

[windows:vars]
ansible_user = username@WEBSITE.COM
ansible_connection = winrm
ansible_port = 5986
```

以下も行う必要があります。

- ホスト名が AD のエントリーと一致する適切なクライアントホスト名であり、IP アドレスではないことを確認してください。
- Kerberos では大文字と小文字が区別されるため、ユーザー名の宣言で、ドメイン名 (@ の後のテキスト) が大文字と小文字を区別して正しく入力されていることを確認してください。
- Automation Controller の場合は、インベントリーが同じであることも確認する必要があります。

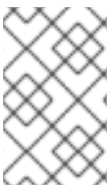
注記

Server not found in Kerberos database というエラーメッセージが表示され、インベントリーが (**IP アドレスではなく**) FQDN を使用して設定されている場合は、サービスプリンシパル名が欠落していないか、またサービスプリンシパル名の設定に間違いがないか確認してください。

Playbook は期待どおりに実行されるはずです。これをテストするには、**awx** ユーザーとして Playbook を実行します。

Playbook が適切に動作することを確認したら、Automation Controller と統合できます。

awx ユーザーとして Kerberos チケットを生成します。Automation Controller は、生成されたチケットを自動的に取得して認証します。



注記

Python の **kerberos** パッケージをインストールする必要があります。Ansible は、**kerberos** パッケージがインストールされているかどうかを確認し、インストールされている場合は Kerberos 認証を使用するように設計されています。

24.2. ACTIVE DIRECTORY および KERBEROS 認証情報

Active Directory のみ:

- マシンの認証情報として AD ユーザー名とパスワードを使用して Windows マシンに対してのみ Playbook を実行する予定の場合は、ユーザー名に "user@<domain>" 形式を使用できます。

Kerberos の場合:

- Kerberos がインストールされている場合には、 "user@<domain>" の形式のユーザー名とパスワードを使用してマシンの認証情報を作成することができます。

24.3. KEROBEROS チケットの使用

Ansible は、Kerberos 用に設定されたホストのマシン認証情報にユーザー名とパスワードの両方が指定されている場合、デフォルトで Kerberos チケットを自動的に管理します。各タスクが実行される前に、新しいチケットが各ホストの一時的な認証情報キャッシュに作成されます (チケットの有効期限が切れる可能性を最小限に抑えるため)。一時的な認証情報キャッシュは各タスクの後に削除され、デフォルトの認証情報キャッシュには影響しません。

チケットの自動管理を無効にする (つまり、既存の SSO チケットを使用するか、手動で **kinit** を呼び出してデフォルトの認証情報キャッシュを生成する) 場合は、インベントリーで **ansible_winrm_kinit_mode=manual** を設定します。

チケットの自動管理には、制御ホストシステムパスに標準の **kinit** バイナリーが必要です。別の場所またはバイナリー名を指定するには、**ansible_winrm_kinit_cmd** インベントリー変数を MIT krbv5 kinit 互換バイナリーへの完全修飾パスに設定します。

第25章 セッション制限

セッション制限を設定すると、管理者はユーザーまたは IP アドレス毎に同時に使用できるセッション数を制限できます。

25.1. セッション制限の使用

Automation Controller では、ユーザーがログインに使用するブラウザーごとにセッションが作成されます。これにより、管理者が定義した最大セッション数を超えると、ユーザーは余分なセッションから強制的にログアウトされます。

セットアップの内容によっては、セッション制限が重要になる場合があります。

例

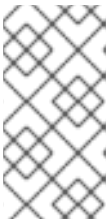
1 人のユーザーが、各デバイスで一度に 1 回だけシステムにログインできるように設定するとします (ユーザーは職場のラップトップ、電話、または自宅のコンピューターでログインできます)。この場合、セッション制限を 1 に設定します。たとえば、ユーザーがラップトップでログインした後、電話を使用してログインすると、ラップトップのセッションは期限切れ (タイムアウト) になり、電話でのログインのみが残ります。プロアクティブなセッション制限により、セッションがアイドル状態のときにユーザーがキックアウトされます。デフォルト値は **-1** で、許可される最大セッション数の上限が無効になります。これは、制限を課されることなく、いくつでもセッションを実行できることを意味します。

セッション数を非常に少なく設定することも可能ですが、組織が必要とするセッションログイン数に対応するように、セッション数を増やすことも可能です。

ユーザーがログインしたことが原因で他のユーザーがログアウトされてしまった場合、セッション制限に達し、ログアウトされてしまったユーザーには、ログアウトされた理由が通知されます。

手順

1. セッション制限を変更するには、ナビゲーションパネルから **Settings** を選択します。
2. **System** オプションのリストから **Miscellaneous Authentication settings** を選択します。
3. **Edit** をクリックします。
4. **Maximum number of simultaneous logged in sessions**設定を編集します。または、REST 要求を行う場合は Browsable API を使用します。



注記

セッション制限を最大限に活用するには、セッション制限の適用範囲外である **AUTH_BASIC_ENABLED** の値を **false** に変更し、無効にします。あるいは、**Miscellaneous Authentication settings** で、**Enable HTTP Basic Auth** をオフに切り替えます。

第26章 バックアップおよび復元

システムのバックアップと復元の機能は、Ansible Automation Platform の設定用 Playbook に統合されています。詳細は、クラスター環境のバックアップおよび復元 セクションを参照してください。

注記

必ずバックアップ元と同じバージョンに復元してください。ただし、お使いの Ansible Automation Platform インストールバージョンをバックアップまたは復元する際は、リリースの最新のマイナーバージョンを使用する必要があります。たとえば、現在使用している Ansible Automation Platform のバージョンが 2.0.x の場合は、最新の 2.0 インストーラーのみを使用してください。

バックアップと復元は、現在のプラットフォームバージョンでサポートされている PostgreSQL バージョンでのみ機能します。詳細は、**Red Hat Ansible Automation Platform インストールガイド** の Red Hat Ansible Automation Platform のシステム要件 を参照してください。

Ansible Automation Platform の設定用 Playbook は、プラットフォームインストーラーの tarball を展開したパスから **setup.sh** として呼び出します。インストール Playbook で使用されるものと同じインベントリーファイルを使用します。セットアップスクリプトでは、バックアップと復元用に以下の引数を指定できます。

- **-b**: データベースのインストールではなくバックアップを実行します。
- **-r**: データベースのインストールではなく復元を実行します。

root ユーザーとして、適切なパラメーターを指定して **setup.sh** を呼び出し、設定どおりに Ansible Automation Platform をバックアップまたは復元します。

```
root@localhost:~# ./setup.sh -b
root@localhost:~# ./setup.sh -r
```

バックアップファイルは、**setup.sh** スクリプトと同じパスに作成されます。パスは、次の **EXTRA_VARS** を指定することで変更できます。

```
root@localhost:~# ./setup.sh -e 'backup_dest=/path/to/backup_dir/' -b
```

次の例に示すように、**EXTRA_VARS** にデフォルト以外のパスを指定しない限り、デフォルトの復元パスが使用されます。

```
root@localhost:~# ./setup.sh -e 'restore_backup_file=/path/to/nondefault/backup.tar.gz' -r
```

オプションで、セットアップスクリプトに引数として渡すことで、使用したインベントリーファイルをオーバーライドすることができます。

```
setup.sh -i <inventory file>
```

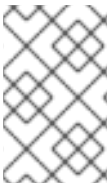
26.1. PLAYBOOK のバックアップおよび復元

setup.sh 設定用 Playbook に含まれる **install.yml** ファイルに加えて、バックアップおよび復元用の **backup.yml** および **restore.yml** ファイルもあります。

これらの Playbook はバックアップと復元に役立ちます。

- システム全体のバックアップでは、以下をバックアップします。
  - データベース
  - **SECRET_KEY** ファイル
- システム毎のバックアップには以下が含まれます。
  - カスタム設定ファイル
  - 手動のプロジェクト
- バックアップの復元では、バックアップされたファイルとデータが、新しくインストールして動作中の Automation Controller の 2 番目のインスタンスに復元されます。

システムを復元する際、インストーラーはバックアップファイルが存在するかどうか確認してから復元を開始します。バックアップファイルがない場合には、復元に失敗します。



注記

Automation Controller ホストが SSH キー、ホストファイル内のユーザー変数またはパスワード変数で適切にセットアップされていること、およびユーザーが **sudo** アクセス権を持っていることを確認してください。

26.2. バックアップおよび復元に関する留意事項

システムをバックアップおよび復元するときは、次の点を考慮してください。

ディスク領域

ディスク領域の要件を確認して、設定ファイル、キー、その他の関連ファイル、および Ansible Automation Platform インストールのデータベースをバックアップするのに十分な領域があることを確認します。

システム認証情報

ローカルデータベースまたはリモートデータベースを操作する場合は、必要なシステム認証情報があることを確認します。ローカルシステムでは、認証情報のセットアップ方法に応じて、**root** または **sudo** アクセスが必要になる場合があります。リモートシステムでは、バックアップまたは復元しようとしているリモートシステムへのアクセス権を取得するためには別の認証情報が必要になる場合があります。

バージョン

Ansible Automation Platform インストールバージョンをバックアップまたは復元する際は、常にリリースの最新のマイナーバージョンを使用する必要があります。たとえば、現在使用しているプラットフォームのバージョンが 2.0.x の場合は、最新の 2.0 インストーラーのみを使用します。

ファイルパス

setup.sh を使用してデフォルトの復元ファイルパス **/var/lib/awx** から復元する場合、復元を実行するには依然として **-r** が必要ですが、このオプションは引数を受け入れなくなりました。デフォルト以外の復元ファイルパスが必要な場合は、extra_var (**root@localhost:~# ./setup.sh -e 'restore_backup_file=/path/to/nondefault/backup.tar.gz' -r**) として指定する必要があります。

ディレクトリー

バックアップファイルが **setup.sh** インストーラーと同じディレクトリーに配置されている場合、復元 Playbook は復元ファイルを自動的に見つけます。この場合、バックアップファイルの場所を指定するために、**restore_backup_file** 追加変数を使用する必要はありません。

26.3. クラスター環境のバックアップおよび復元

クラスター環境のバックアップと復元の手順は、次のいくつかの留意事項を除き、単一インストールと同様です。

注記

クラスター環境のインストールの詳細は、インストールと設定 セクションを参照してください。

- 新規クラスターに復元する場合は、データベースにアクセスする際にはクラスター間で競合状態になる可能性があるので、以前のクラスターをシャットダウンしてから続行するようにしてください。
- ノードごとのバックアップは、バックアップと同じホスト名のノードにのみ復元されます。
- 既存のクラスターに復元する場合、復元には以下が含まれます。
  - PostgreSQL データベースのダンプ
  - データベースダンプに含まれる UI アーティファクト
  - Automation Controller 設定 (**/etc/tower** から取得)
  - Automation Controller シークレットキー
  - 手動のプロジェクト

26.3.1. 異なるクラスターへの復元

バックアップを別のインスタンスまたはクラスターに復元する場合、**/etc/tower** 配下の手動プロジェクトとカスタム設定は保持されます。ジョブ出力とジョブイベントはデータベースに保存されるため、影響を受けません。

復元プロセスでは、復元前に存在していたインスタンスグループは変更されません。新しいインスタンスグループが導入されることもありません。インスタンスグループに関連付けられた Automation Controller リソースを復元した場合、新しい Automation Controller クラスターのインスタンスグループに当該リソースを再度割り当てることが必要な可能性があります。

第27章 ユーザビリティーアナリティクスおよびデータ収集

Automation Controller にはユーザビリティーデータ収集が含まれており、ユーザーがどのように Automation Controller と対話するかをよりよく理解するためのデータを収集します。

試用版をインストールしたユーザーまたは新規にインストールしたユーザーのみが、このデータ収集にオプトインされます。

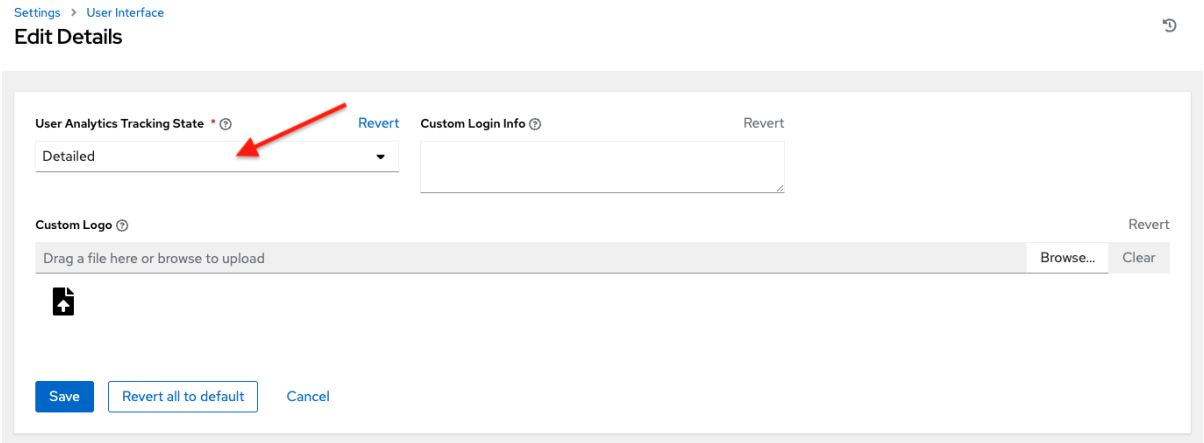
Automation Controller は、製品の改善に役立てるためにユーザーデータを自動的に収集します。**Settings** メニューの **User Interface settings** で参加レベルを設定することで、Automation Controller によるデータ収集のオプトアウトや制御が可能です。

27.1. データ収集への参加のセットアップ

データ収集の参加レベルを設定するには、次の手順を使用します。

手順

1. ナビゲーションパネルから **Settings** を選択します。
2. **User Interface** オプションから **User Interface settings** を選択します。
3. **Edit** をクリックします。



4. **User Analytics Tracking State** 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。
5. **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。

詳細は、Red Hat のプライバシーに関する声明 を参照してください。

27.2. AUTOMATION ANALYTICS

ライセンスの初回インポート時に、Automation Analytics (Ansible Automation Platform サブスクリプションに含まれるクラウドサービス) を強化するデータ収集に関するオプションが提供されます。

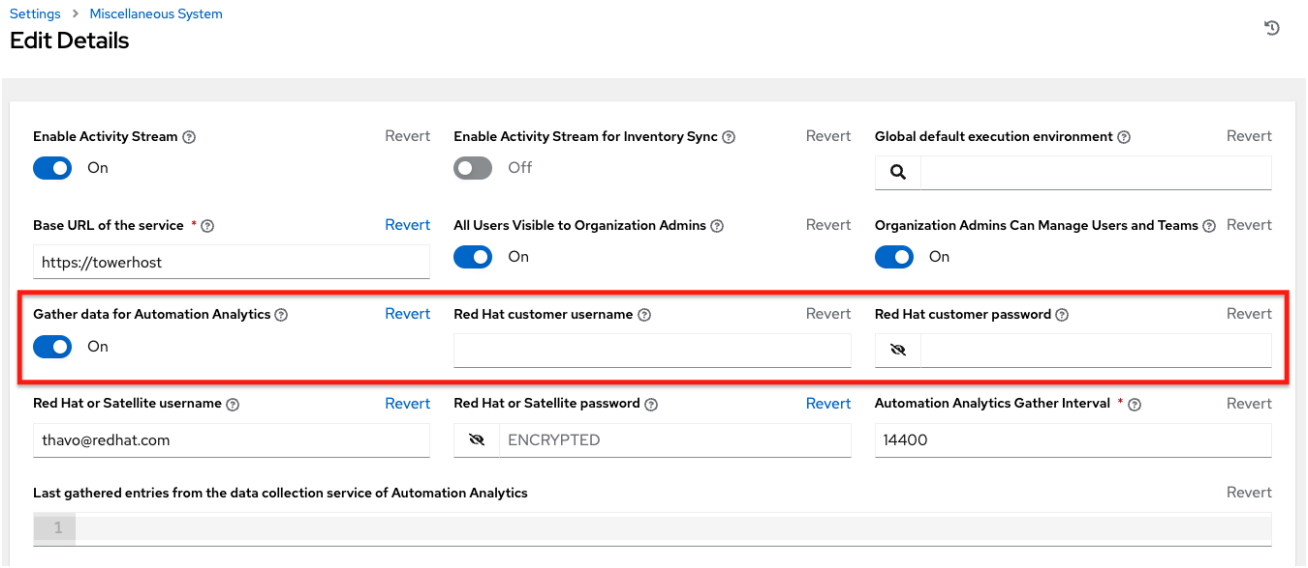
重要

Automation Analytics のオプトインを有効にするには、Automation Controller のインスタンスが Red Hat Enterprise Linux 上で実行されている必要があります。

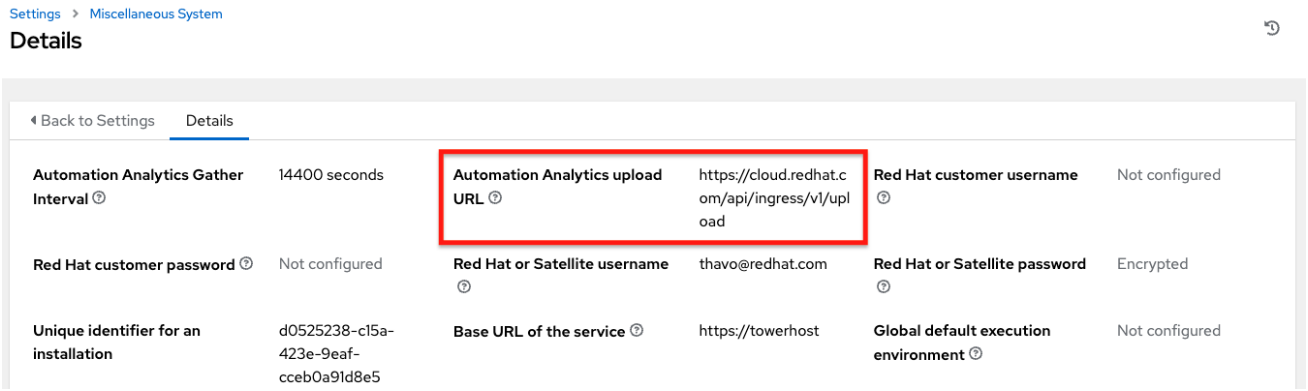
Red Hat Insights と同様に、Automation Analytics は必要最小限のデータを収集するようにビルドされています。認証情報のシークレット、個人データ、自動化変数、またはタスク出力は収集されません。

詳細は、データ収集の詳細 を参照してください。

この機能を有効にするには、Automation Analytics のデータ収集をオンにし、**Settings** メニューにあるオプションのシステム設定リストの **Miscellaneous System settings** に、Red Hat カスタマー認証情報を入力します。



洞察データのコレクションがアップロードされる場所は、**Details** ページの **Automation Analytics upload URL** フィールドで確認できます。

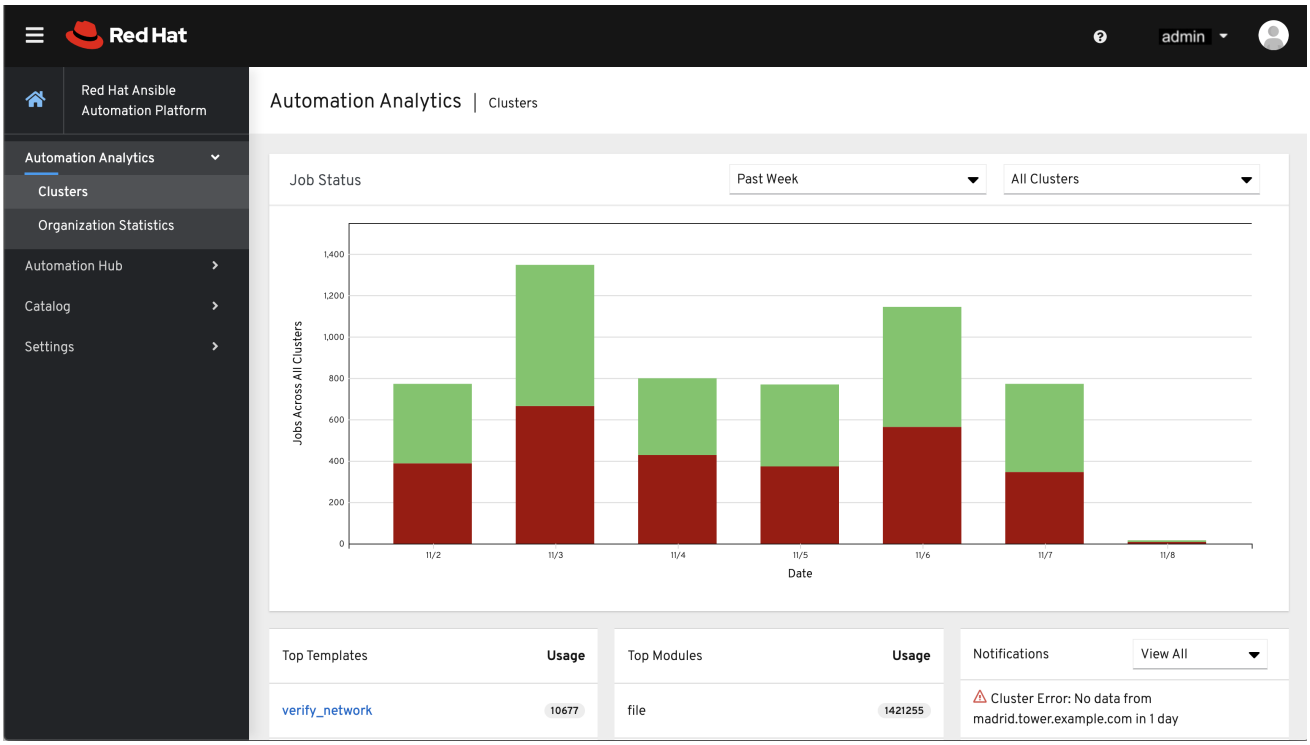


デフォルトでは、データは 4 時間ごとに収集されます。この機能を有効にすると、最大 1 カ月 (または前回の収集まで) 遡ってデータが収集されます。このデータ収集は、システム設定ウィンドウの **Miscellaneous System settings** でいつでもオフにできます。

この設定は、次のエンドポイントのいずれかで **INSIGHTS_TRACKING_STATE = true** を指定し、API 経由で有効にすることもできます。

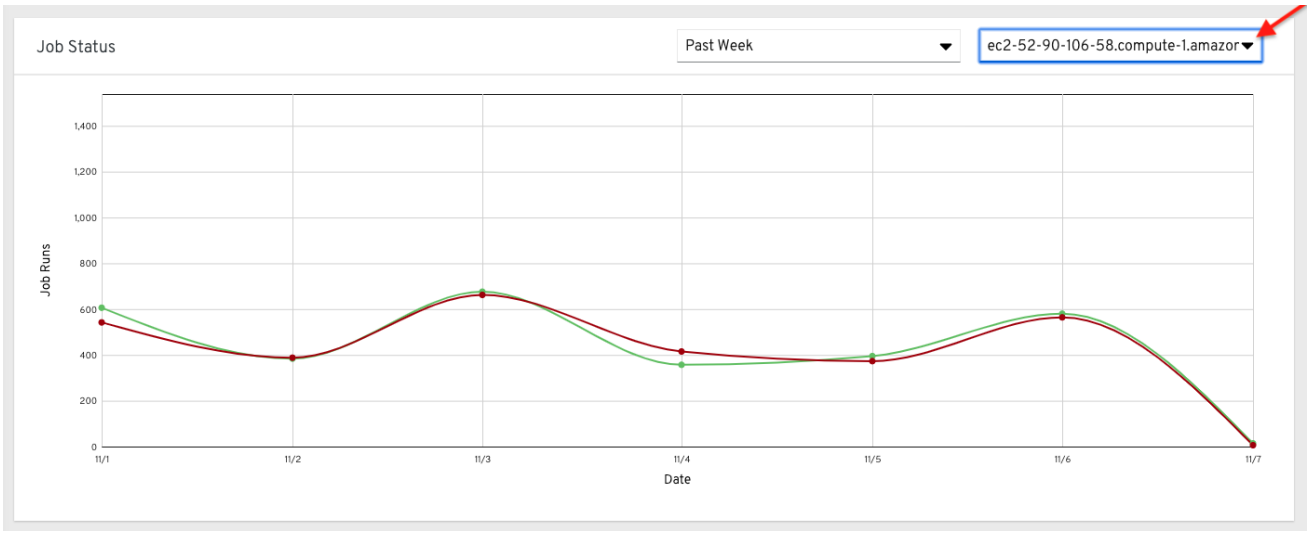
- **api/v2/settings/all**
- **api/v2/settings/system**

このデータ収集から生成された Automation Analytics は、Red Hat Cloud Services ポータルで確認できます。



デフォルトのビューは **Clusters** データです。このグラフは、一定期間における Automation Controller クラスター全体のジョブ実行数を表します。上記の例では、1 週間分の積み上げ棒グラフが表示されています。グラフは、正常に実行されたジョブの数 (緑色) と失敗したジョブの数 (赤色) で構成されています。

クラスターを 1 つ選択して、そのジョブステータス情報を表示することもできます。



この複数の折れ線グラフは、指定した期間における単一の Automation Controller クラスターのジョブ実行数を表します。上記の例では 1 週間分が示されており、正常に実行されたジョブの数 (緑色) と失敗したジョブの数 (赤色) で構成されています。選択したクラスターの成功したジョブ実行数と失敗したジョブ実行数を、1 週間単位、2 週間単位、および月次単位で表示するように指定できます。

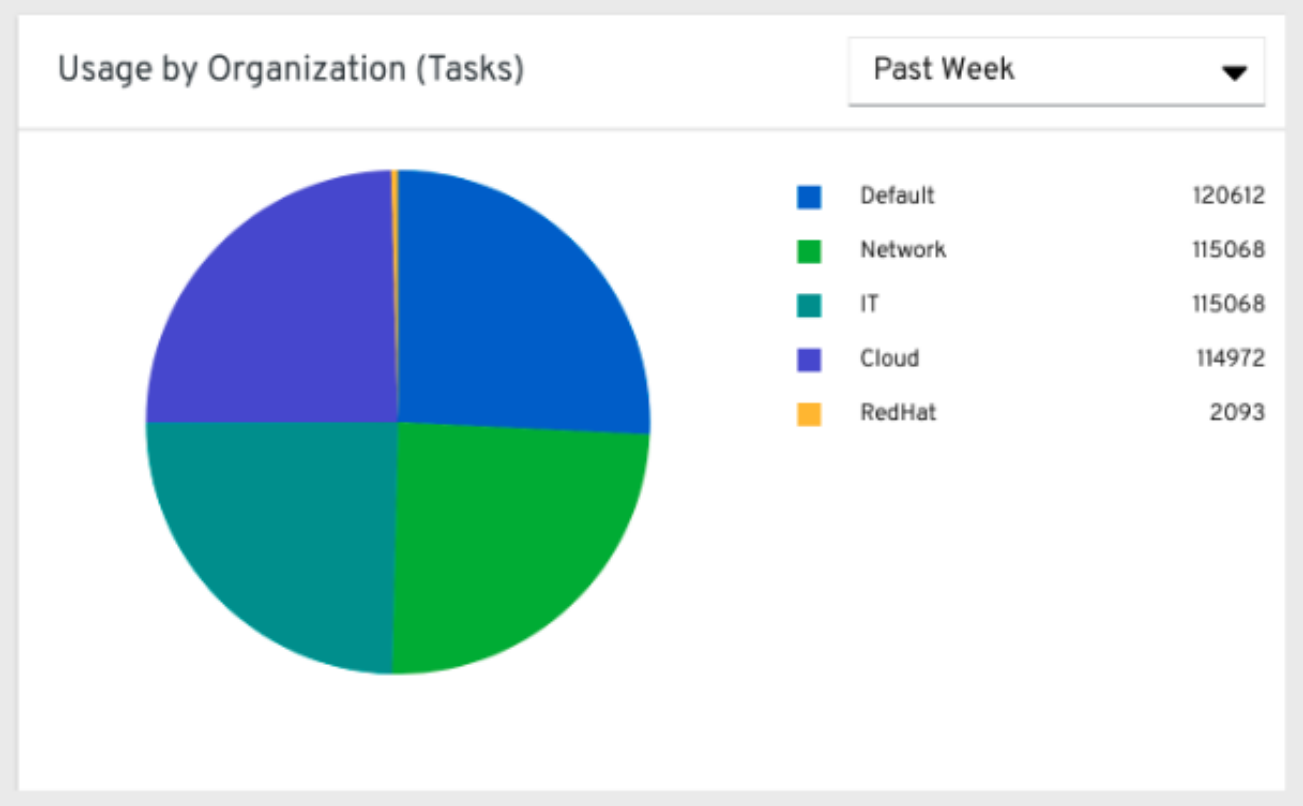
クラウドナビゲーションパネルで **Organization Statistics** を選択し、次の情報を表示します。

- 組織ごとの使用状況

- 組織ごとのジョブの実行数
- 組織の状態

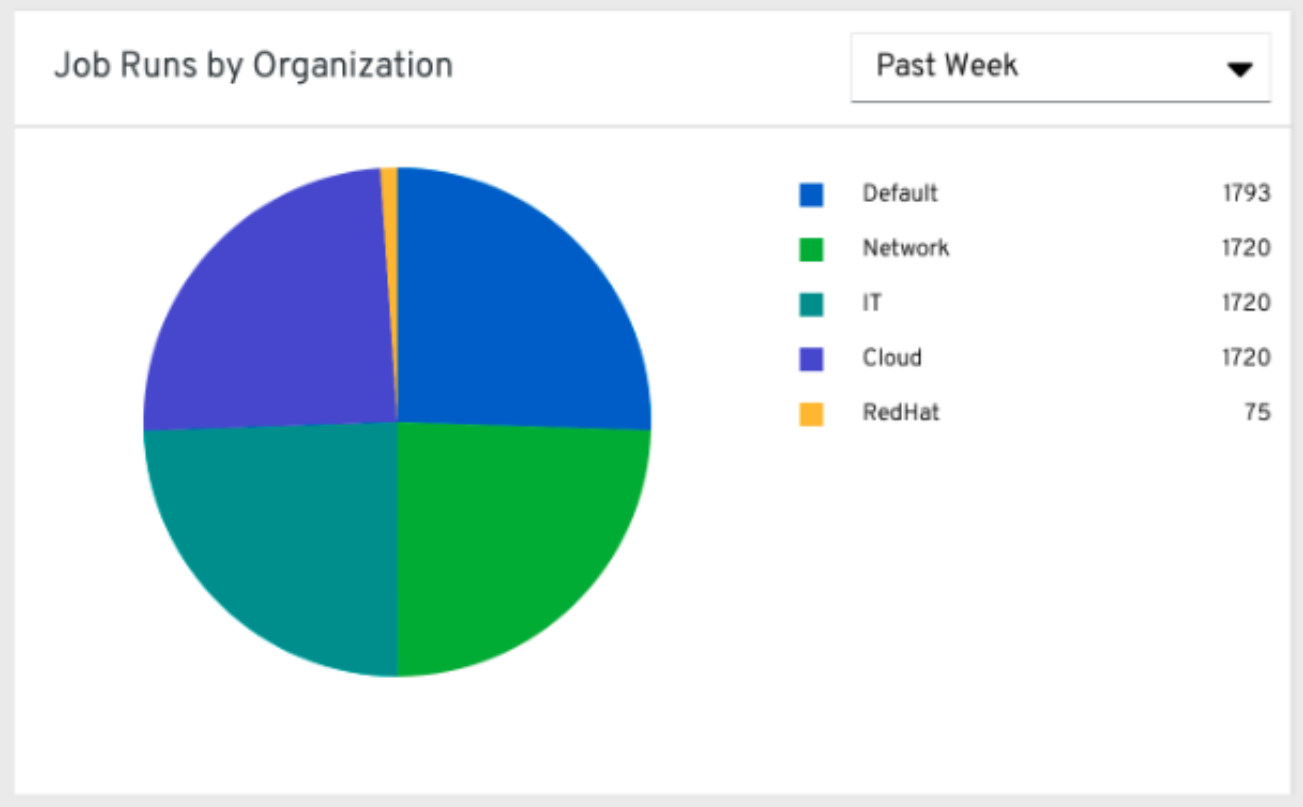
27.2.1. 組織ごとの使用状況

次のグラフは、特定の組織がすべてのジョブ内で実行したタスクの数を表します。



27.2.2. 組織ごとのジョブの実行数

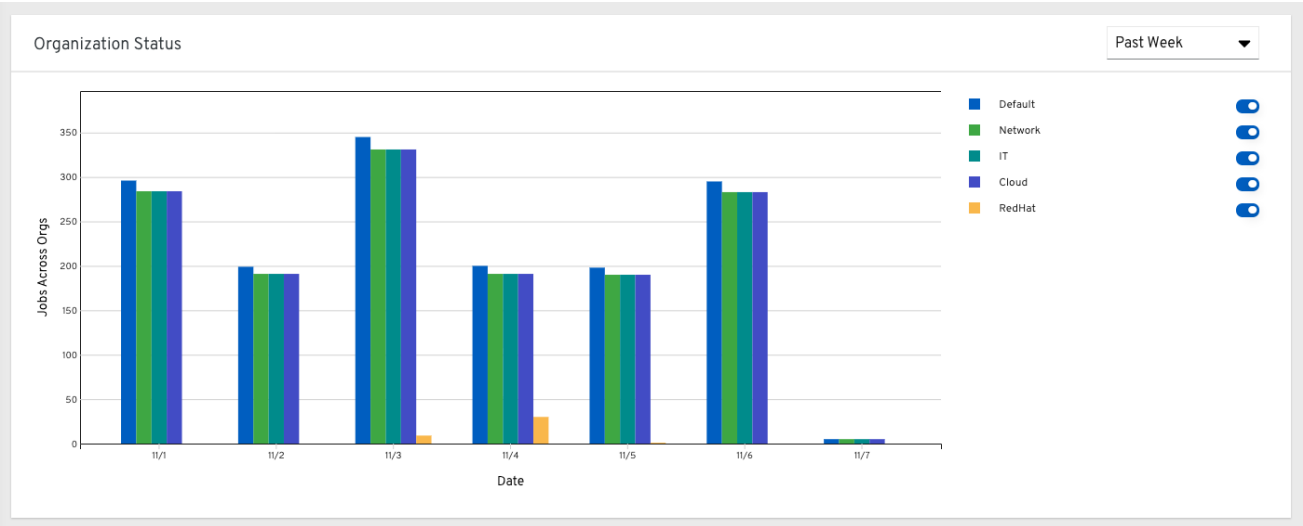
このグラフは、組織ごとのすべての Automation Controller クラスターでの Automation Controller の使用状況を表します。これは、組織が実行したジョブの数に基づいて計算されます。



27.2.3. 組織の状態

この棒グラフは、組織および日付別の Automation Controller の使用状況を表します。これは、組織が特定の日に実行したジョブの数に基づいて計算されます。

あるいは、組織ごとのジョブ実行数を 1 週間単位、2 週間単位、および月次単位で表示するように指定することもできます。



27.3. データ収集の詳細

Automation Analytics は、Automation Controller から次の種類のデータを収集します。

- 有効になっている機能や使用されているオペレーティングシステムなどの基本設定
- 容量と健全性など、Automation Controller 環境とホストのトポロジーおよびステータス

- 自動化リソースの数:
  - 組織、チーム、ユーザー
  - インベントリーとホスト
  - 認証情報 (タイプで索引付け)
  - プロジェクト (タイプで索引付け)
  - テンプレート
  - スケジュール
  - アクティブセッション
  - 実行中および保留中のジョブ
- ジョブ実行の詳細 (開始時間、終了時間、起動タイプ、成功)
- 自動化タスクの詳細 (成功、ホスト ID、Playbook/ロール、タスク名、使用モジュール)

awx-manage gather_analytics (**--ship** なし) を使用して Automation Controller が送信するデータを検査できるため、データ収集に関する懸念に対応できます。これにより、Red Hat に送信される分析データを含む tarball が作成されます。

このファイルには、多数の JSON ファイルと CSV ファイルが含まれています。各ファイルには、異なる分析データのセットが含まれています。

- manifest.json
- config.json
- instance_info.json
- counts.json
- org_counts.json
- cred_type_counts.json
- inventory_counts.json
- projects_by_scm_type.json
- query_info.json
- job_counts.json
- job_instance_counts.json
- unified_job_template_table.csv
- unified_jobs_table.csv
- workflow_job_template_node_table.csv
- workflow_job_node_table.csv

- events_table.csv

27.3.1. manifest.json

manifest.json は、分析データのマニフェストです。これは、コレクションに含まれる各ファイルと、ファイルのスキーマのバージョンについて記述しています。

以下は、**manifest.json** ファイルの例です。

```
  "config.json": "1.1",
  "counts.json": "1.0",
  "cred_type_counts.json": "1.0",
  "events_table.csv": "1.1",
  "instance_info.json": "1.0",
  "inventory_counts.json": "1.2",
  "job_counts.json": "1.0",
  "job_instance_counts.json": "1.0",
  "org_counts.json": "1.0",
  "projects_by_scm_type.json": "1.0",
  "query_info.json": "1.0",
  "unified_job_template_table.csv": "1.0",
  "unified_jobs_table.csv": "1.0",
  "workflow_job_node_table.csv": "1.0",
  "workflow_job_template_node_table.csv": "1.0"
}
```

27.3.2. config.json

config.json ファイルには、クラスターの設定エンドポイント **/api/v2/config** のサブセットが含まれています。config.json の例は次のとおりです。

```
{
    "ansible_version": "2.9.1",
    "authentication_backends": [
        "social_core.backends.azuread.AzureADOAuth2",
        "django.contrib.auth.backends.ModelBackend"
    ],
    "external_logger_enabled": true,
    "external_logger_type": "splunk",
    "free_instances": 1234,
    "install_uuid": "d3d497f7-9d07-43ab-b8de-9d5cc9752b7c",
    "instance_uuid": "bed08c6b-19cc-4a49-bc9e-82c33936e91b",
    "license_expiry": 34937373,
    "license_type": "enterprise",
    "logging_aggregators": [
        "awx",
        "activity_stream",
        "job_events",
        "system_tracking"
    ],
    "pendo_tracking": "detailed",
    "platform": {
        "dist": [
            "redhat",
            "7.4",
```

```
            "Maipo"
        ],
        "release": "3.10.0-693.el7.x86_64",
        "system": "Linux",
        "type": "traditional"
    },
    "total_licensed_instances": 2500,
    "controller_url_base": "https://ansible.rhdemo.io",
    "controller_version": "3.6.3"
}
```


これには次のフィールドが含まれます。

- **ansible_version**: ホスト上のシステム Ansible バージョン
- **authentication_backends**: 使用可能なユーザー認証バックエンド。詳細は、ソーシャル認証のセットアップ または LDAP 認証のセットアップ を参照してください。
- **external_logger_enabled**: 外部ロギングが有効かどうか
- **external_logger_type**: 有効になっている場合に使用されているロギングバックエンド。詳細は、ロギングおよびアグリゲーション を参照してください。
- **logging_aggregators**: 外部ロギングに送信されるロギングカテゴリー。詳細は、ロギングおよびアグリゲーション を参照してください。
- **free_instances**: ライセンスで使用できるホストの数。値がゼロの場合は、クラスターがライセンスを完全に消費していることを意味します。
- **install_uuid**: インストールの UUID (すべてのクラスターノードで同一)
- **instance_uuid**: インスタンスの UUID (クラスターノードごとに異なる)
- **license_expiry**: ライセンスの有効期限が切れるまでの時間 (秒単位)
- **license_type**: ライセンスのタイプ (ほとんどの場合は 'enterprise')
- **pendo_tracking**: **usability_data_collection** の状態
- **platform**: クラスターが実行されているオペレーティングシステム
- **total_licensed_instances**: ライセンス内のホストの合計数
- **controller_url_base**: クライアントが使用するクラスターのベース URL (Automation Analytics に表示)
- **controller_version**: クラスター上のソフトウェアのバージョン

27.3.3. instance_info.json

instance_info.json ファイルには、クラスターを構成するインスタンスに関する詳細情報がインスタンスの UUID ごとにまとめられています。

以下は、**instance_info.json** ファイルの例です。


```
{
    "bed08c6b-19cc-4a49-bc9e-82c33936e91b": {
```

```
        "capacity": 57,
        "cpu": 2,
        "enabled": true,
        "last_isolated_check": "2019-08-15T14:48:58.553005+00:00",
        "managed_by_policy": true,
        "memory": 8201400320,
        "uuid": "bed08c6b-19cc-4a49-bc9e-82c33936e91b",
        "version": "3.6.3"
    }
    "c0a2a215-0e33-419a-92f5-e3a0f59bfaee": {
        "capacity": 57,
        "cpu": 2,
        "enabled": true,
        "last_isolated_check": "2019-08-15T14:48:58.553005+00:00",
        "managed_by_policy": true,
        "memory": 8201400320,
        "uuid": "c0a2a215-0e33-419a-92f5-e3a0f59bfaee",
        "version": "3.6.3"
    }
}
```


これには次のフィールドが含まれます。

- **capacity**: タスクを実行するためのインスタンスの容量
- **cpu**: インスタンスのプロセッサーコア
- **memory**: インスタンスのメモリー
- **enabled**: インスタンスが有効でタスクを受け付けているかどうか
- **managed_by_policy**: インスタンスグループ内のインスタンスのメンバーシップがポリシーによって管理されているか、手動で管理されているか
- **version**: インスタンス上のソフトウェアのバージョン

27.3.4. counts.json

counts.json ファイルには、クラスター内の関連するカテゴリーごとの合計オブジェクト数が含まれています。

以下は、**counts.json** ファイルの例です。


```
{
    "active_anonymous_sessions": 1,
    "active_host_count": 682,
    "active_sessions": 2,
    "active_user_sessions": 1,
    "credential": 38,
    "custom_inventory_script": 2,
    "custom_virtualenvs": 4,
    "host": 697,
    "inventories": {
        "normal": 20,
        "smart": 1
    },
```

```
    "inventory": 21,
    "job_template": 78,
    "notification_template": 5,
    "organization": 10,
    "pending_jobs": 0,
    "project": 20,
    "running_jobs": 0,
    "schedule": 16,
    "team": 5,
    "unified_job": 7073,
    "user": 28,
    "workflow_job_template": 15
}
```

このファイルの各エントリーは、アクティブセッション数を除いて、**/api/v2** の対応する API オブジェクトに関するものです。

27.3.5. org_counts.json

org_counts.json ファイルには、クラスター内の各組織に関する情報と、その組織に関連付けられているユーザーとチームの数が含まれています。

以下は、**org_counts.json** ファイルの例です。

```
{
    "1": {
        "name": "Operations",
        "teams": 5,
        "users": 17
    },
    "2": {
        "name": "Development",
        "teams": 27,
        "users": 154
    },
    "3": {
        "name": "Networking",
        "teams": 3,
        "users": 28
    }
}
```

27.3.6. cred_type_counts.json

cred_type_counts.json ファイルには、クラスター内のさまざまな認証情報のタイプと、各タイプに存在する認証情報の数に関する情報が含まれています。

以下は、**cred_type_counts.json** ファイルの例です。

```
{
    "1": {
        "credential_count": 15,
        "managed_by_controller": true,
        "name": "Machine"
    },
```

```
    "2": {
        "credential_count": 2,
        "managed_by_controller": true,
        "name": "Source Control"
    },
    "3": {
        "credential_count": 3,
        "managed_by_controller": true,
        "name": "Vault"
    },
    "4": {
        "credential_count": 0,
        "managed_by_controller": true,
        "name": "Network"
    },
    "5": {
        "credential_count": 6,
        "managed_by_controller": true,
        "name": "Amazon Web Services"
    },
    "6": {
        "credential_count": 0,
        "managed_by_controller": true,
        "name": "OpenStack"
    },
```


27.3.7. inventory_counts.json

inventory_counts.json ファイルには、クラスター内のさまざまなインベントリーに関する情報が含まれています。

以下は、**inventory_counts.json** ファイルの例です。


```
{
    "1": {
        "hosts": 211,
        "kind": "",
        "name": "AWS Inventory",
        "source_list": [
            {
                "name": "AWS",
                "num_hosts": 211,
                "source": "ec2"
            }
        ],
        "sources": 1
    },
    "2": {
        "hosts": 15,
        "kind": "",
        "name": "Manual inventory",
        "source_list": [],
        "sources": 0
    },
    "3": {
```

```
        "hosts": 25,
        "kind": "",
        "name": "SCM inventory - test repo",
        "source_list": [
            {
                "name": "Git source",
                "num_hosts": 25,
                "source": "scm"
            }
        ],
        "sources": 1
    }
    "4": {
        "num_hosts": 5,
        "kind": "smart",
        "name": "Filtered AWS inventory",
        "source_list": [],
        "sources": 0
    }
}
```


27.3.8. projects_by_scm_type.json

project_by_scm_type.json ファイルは、クラスター内のすべてのプロジェクトの内訳を、ソースコントロールタイプごとに示します。

以下は、**projects_by_scm_type.json** ファイルの例です。


```
{
    "git": 27,
    "hg": 0,
    "insights": 1,
    "manual": 0,
    "svn": 0
}
```


27.3.9. query_info.json

query_info.json ファイルには、データ収集がいつどのように行われたかに関する詳細が記載されています。

以下は、**query_info.json** ファイルの例です。


```
{
    "collection_type": "manual",
    "current_time": "2019-11-22 20:10:27.751267+00:00",
    "last_run": "2019-11-22 20:03:40.361225+00:00"
}
```


collection_type は、**manual** または **automatic** のいずれかです。

27.3.10. job_counts.json

job_counts.json ファイルは、クラスターのジョブ履歴の詳細を提供し、ジョブの起動方法とジョブの終了ステータスの両方を記述します。

以下は、**job_counts.json** ファイルの例です。

```
    "launch_type": {
        "dependency": 3628,
        "manual": 799,
        "relaunch": 6,
        "scheduled": 1286,
        "scm": 6,
        "workflow": 1348
    },
    "status": {
        "canceled": 7,
        "failed": 108,
        "successful": 6958
    },
    "total_jobs": 7073
}
```

27.3.11. job_instance_counts.json

job_instance_counts.json ファイルは、**job_counts.json** と同じ詳細をインスタンスごとに分類して記載します。

以下は、**job_instance_counts.json** ファイルの例です。

```
{
    "localhost": {
        "launch_type": {
            "dependency": 3628,
            "manual": 770,
            "relaunch": 3,
            "scheduled": 1009,
            "scm": 6,
            "workflow": 1336
        },
        "status": {
            "canceled": 2,
            "failed": 60,
            "successful": 6690
        }
    }
}
```

このファイルのインスタンスは、**instance_info** のように UUID ごとに記載されているのではなく、ホスト名ごとに記載されていることに注意してください。

27.3.12. unified_job_template_table.csv

unified_job_template_table.csv ファイルは、システム内のジョブテンプレートに関する情報を提供します。各行には、ジョブテンプレートに関する次のフィールドが含まれます。

- **id**: ジョブテンプレート ID。

- name: ジョブテンプレート名。
- polymorphic_ctype_id: テンプレートのタイプの ID。
- model: テンプレートの **polymorphic_ctype_id** の名前。例として、**project**、**systemjobtemplate**、**jobtemplate**、**inventorysource**、**workflowjobtemplate** などがあります。
- created: テンプレートが作成された日時。
- updated: テンプレートが最後に更新された日時。
- created_by_id: テンプレートを作成した **userid**。システムが作成した場合は空白です。
- modified_by_id: テンプレートを最後に変更した **userid**。システムが変更した場合は空白です。
- current_job_id: テンプレートの現在実行中のジョブ ID (該当する場合)。
- last_job_id: ジョブの最後の実行。
- last_job_run: ジョブが最後に実行された時刻。
- last_job_failed: **last_job_id** が失敗したかどうか。
- status: **last_job_id** のステータス。
- next_job_run: テンプレートで次にスケジュールされている実行 (該当する場合)。
- next_schedule_id: **next_job_run** のスケジュール ID (該当する場合)。

27.3.13. unified_jobs_table.csv

unified_jobs_table.csv ファイルは、システムによって実行されるジョブに関する情報を提供します。

各行には、ジョブに関する次のフィールドが含まれます。

- id: ジョブ ID。
- name: ジョブ名 (テンプレートから)。
- polymorphic_ctype_id: ジョブのタイプの ID。
- model: ジョブの **polymorphic_ctype_id** の名前。例として、**job** や **workflow** などがあります。
- organization_id: ジョブの組織 ID。
- organization_name: **organization_id** の名前。
- created: ジョブレコードが作成された日時。
- starting: ジョブの実行が開始した日時。
- completed: ジョブが終了した日時。
- elapsed: ジョブの経過時間 (秒単位)。

- unified_job_template_id: このジョブのテンプレート。
- launch_type: **manual**、**scheduled**、**relaunched**、**scm**、**workflow**、**dependency** のいずれか。
- schedule_id: ジョブを起動したスケジュールの ID (該当する場合)
- instance_group_id: ジョブを実行したインスタンスグループ。
- execution_node: ジョブを実行したノード (UUID ではなくホスト名)。
- controller_node: ジョブの Automation Controller ノード (分離ジョブとして実行される場合、またはコンテナーグループ内で実行される場合)。
- cancel_flag: ジョブがキャンセルされたかどうか。
- status: ジョブのステータス。
- failed: ジョブが失敗したかどうか。
- job_explanation: 適切に実行できなかったジョブの追加の詳細。
- forks: ジョブに対して実行されたフォークの数。

27.3.14. workflow_job_template_node_table.csv

workflow_job_template_node_table.csv は、システムのワークフロージョブテンプレートで定義されるノードの情報を提供します。

各行には、ワークフローのジョブテンプレートノードに以下のフィールドが含まれます。

- id: ノード ID。
- created: ノードが作成された日時。
- modified: ノードが最後に更新された日時。
- unified_job_template_id: このノードのジョブテンプレート、プロジェクト、インベントリー、またはその他の親リソースの ID。
- workflow_job_template_id: このノードを含むワークフローのジョブテンプレート。
- inventory_id: このノードによって使用されるインベントリー。
- success_nodes: このノードが成功した後にトリガーされるノード。
- failure_nodes: このノードが失敗した後にトリガーされるノード。
- always_nodes: このノードの終了後に常にトリガーされるノード。
- all_parents_must_converge: このノードを起動するためにすべての親条件が満たされる必要があるかどうか。

27.3.15. workflow_job_node_table.csv

workflow_job_node_table.csv は、システム上のワークフローの一部として実行されたジョブに関する情報を提供します。

各行には、ワークフローの一部として実行されるジョブに次のフィールドが含まれています。

- **id**: ノード ID。
- **created**: ノードが作成された日時。
- **modified**: ノードが最後に更新された日時。
- **job_id**: このノードで実行されるジョブのジョブ ID。
- **unified_job_template_id**: このノードのジョブテンプレート、プロジェクト、インベントリー、またはその他の親リソースの ID。
- **workflow_job_template_id**: このノードを含むワークフローのジョブテンプレート。
- **inventory_id**: このノードによって使用されるインベントリー。
- **success_nodes**: このノードが成功した後にトリガーされるノード。
- **failure_nodes**: このノードが失敗した後にトリガーされるノード。
- **always_nodes**: このノードの終了後に常にトリガーされるノード。
- **do_not_run**: 開始条件がトリガーされなかったためにワークフローで実行されなかったノード。
- **all_parents_must_converge**: このノードを起動するためにすべての親条件が満たされる必要があるかどうか。

27.3.16. events_table.csv

events_table.csv ファイルは、システム内のすべてのジョブ実行からのすべてのジョブイベントに関する情報を提供します。

各行には、ジョブイベントに関する次のフィールドが含まれます。

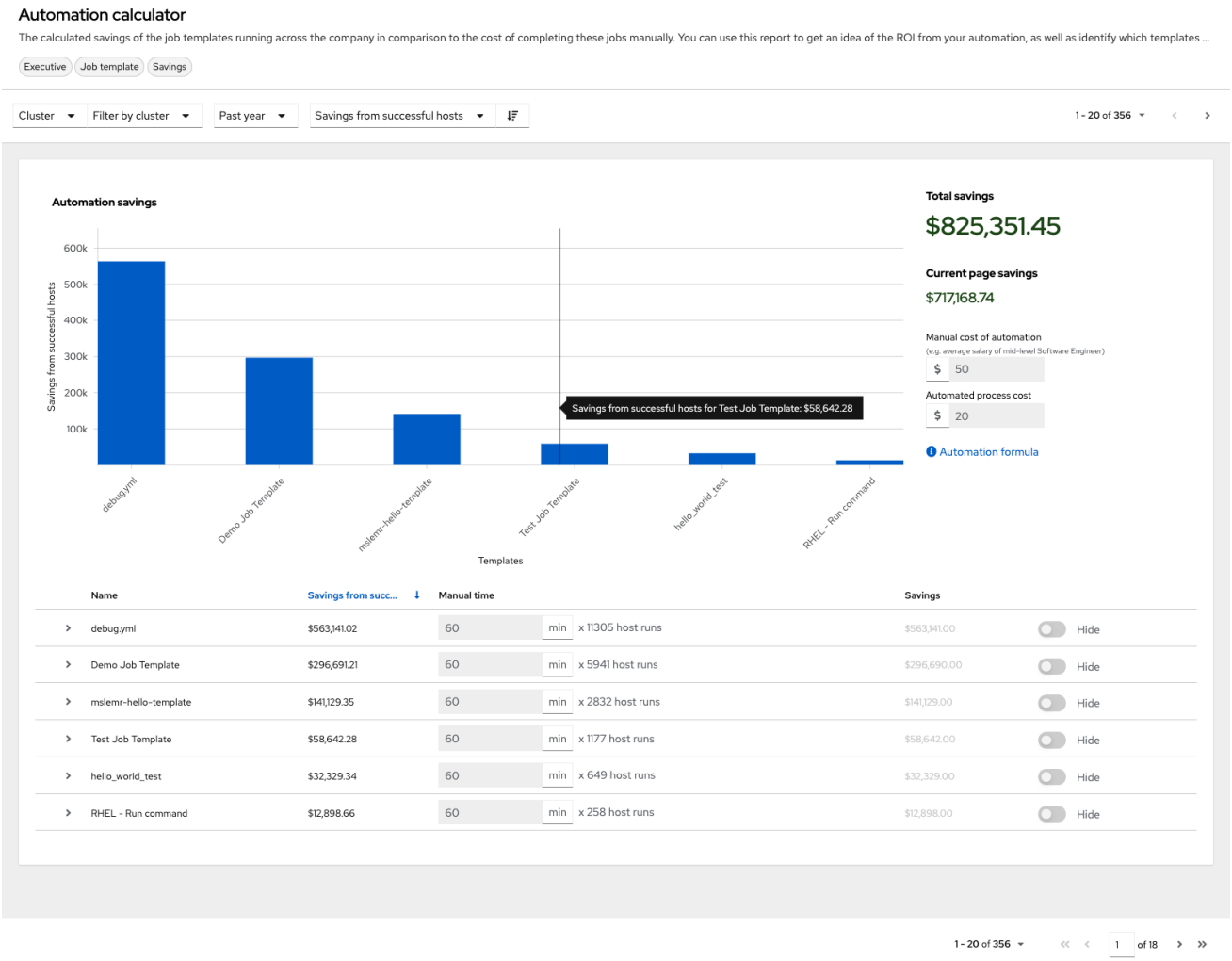
- **id**: イベント ID。
- **uuid**: イベント UUID。
- **created**: イベントが作成された日時。
- **parent_uuid**: このイベントの親 UUID (存在する場合)。
- **event**: Ansible イベントタイプ。
- **task_action**: このイベントに関連付けられたモジュール (存在する場合) (**command** や **yum** など)。
- **failed**: イベントが **failed** を返したかどうか。
- **changed**: イベントが **changed** を返したかどうか。
- **playbook**: イベントに関連付けられた Playbook。
- **play**: Playbook のプレイ名。

- **task**: Playbook のタスク名。
- **role**: Playbook のロール名。
- **job_id**: このイベントが発生したジョブの ID。
- **host_id**: このイベントが関連付けられているホストの ID (存在する場合)。
- **host_name**: このイベントが関連付けられているホストの名前 (存在する場合)。
- **start**: タスクの開始時刻。
- **end**: タスクの終了時刻。
- **duration**: タスクの期間。
- **warnings**: タスクまたはモジュールからの警告。
- **deprecations**: タスクまたはモジュールからの非推奨の警告。

27.4. 分析レポート

スーパーユーザーレベルのパーミッションがある場合は、Automation Controller UI でコレクションからのレポートにアクセスできます。最も便利な場所にオンプレミスで分析ビューを組み込むことで、日々の業務に影響を与える可能性のあるデータにアクセスできます。このデータは、console.redhat.com で提供される自動化から集約されます。

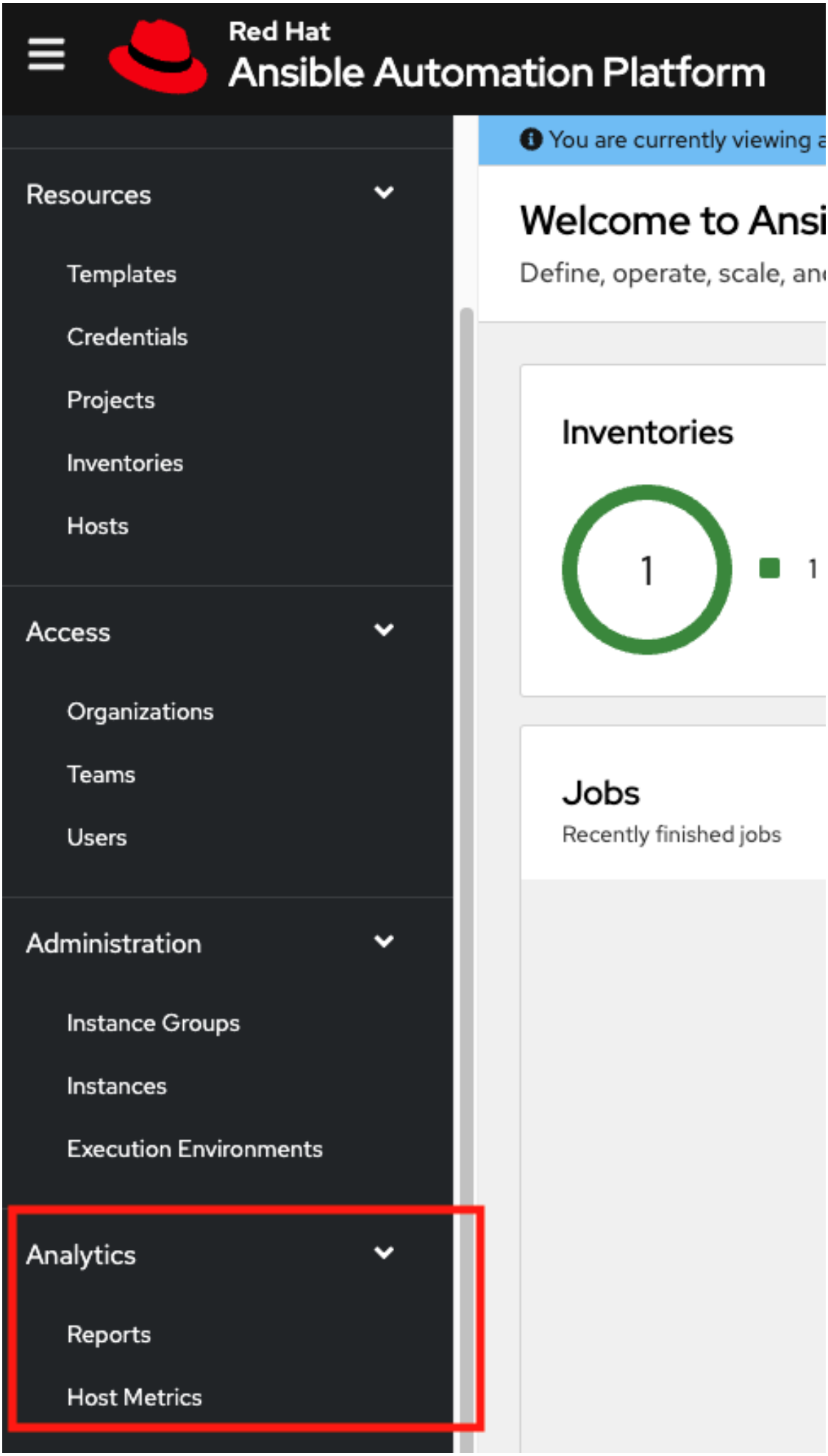
現在利用可能なのは、サブスクライバーに対して (潜在的な) 節約額を示すレポートを表示する Automation Calculator ユーティリティーの表示専用バージョンです。



注記

このオプションはテクニカルプレビューとして利用できますが、今後のリリースで変更される可能性があります。分析レポートビューをプレビューするには、**Settings** メニューの **Miscellaneous System Settings** オプションで **Enable Preview of New User Interface** トグルを **On** に設定します。

保存後、ログアウトして再度ログインし、ナビゲーションパネルの **Analytics** セクションのオプションにアクセスします。


Red Hat
Ansible Automation Platform
You are currently viewing a
Resources
Templates
Credentials
Projects
Inventories
Hosts
Access
Organizations
Teams
Users
Administration
Instance Groups
Instances
Execution Environments
Analytics
Reports
Host Metrics
Welcome to Ansi
Define, operate, scale, an
Inventories
1
1
Jobs
Recently finished jobs

Host Metrics は、ホストデータに関して収集される別の分析レポートです。UI の上記箇所からこのオプションにアクセスする機能は現在テクノロジープレビュー段階にあり、将来のリリースで変更される可能性があります。詳細は、Automation Controller の設定 の **Host Metrics ビュー** を参照してください。

第28章 AUTOMATION CONTROLLER のトラブルシューティング

以下に、Automation Controller に関する有用なトラブルシューティング情報を示します。

28.1. ホストに接続できない

ホストの接続エラーが原因で、**Automation Controller のスタートガイド** の プロジェクトの管理 セクションの **helloworld.yml** サンプル Playbook やその他の Playbook を実行できない場合は、以下をお試しください。

- ホストに **ssh** 接続できますか? Ansible では、管理するサーバーに SSH でアクセスできる必要があります。
- **hostnames** と IP がインベントリーファイルに正しく追加されていますか? タイプミスがないか確認してください。

28.2. HTTP 経由で AUTOMATION CONTROLLER にログインできない

Automation Controller へのアクセスは、セキュアなプロトコル (HTTPS) を使用して意図的に制限されています。Automation Controller ノードをロードバランサーやプロキシーの背後で "HTTP のみ" として実行するようにセットアップしており、SSL を使用せずに当該ノードにアクセスしたい場合 (トラブルシューティングのためなど)、Automation Controller インスタンスの **/etc/tower/conf.d** にある **custom.py** ファイルに次の設定を追加する必要があります。

```
SESSION_COOKIE_SECURE = False
CSRF_COOKIE_SECURE = False
```

これらの設定を **false** に変更すると、HTTP プロトコルの使用時に Automation Controller がクッキーとログインセッションを管理できるようになります。この変更は、クラスターインストールのすべてのノードで行う必要があります。

変更を適用するには、以下を実行します。

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

28.3. PLAYBOOK を実行できない

Playbook のエラーが原因で **Automation Controller のスタートガイド** の プロジェクトの管理 セクションの **helloworld.yml** サンプル Playbook を実行できない場合は、以下をお試しください。

- 現在コマンドを実行しているユーザーで認証していることを確認してください。そのように認証していない場合は、ユーザー名がどのようにセットアップされているかを確認するか、**--user=username** または **-u username** コマンドを渡してユーザーを指定します。
- YAML ファイルは正しくインデントされていますか? 空白を正しく配置することが必要な場合があります。YAML ではインデントのレベルが重要です。**yamlint** を使用して Playbook を確認できます。

28.4. ジョブを実行できない

Playbook からジョブを実行できない場合は、Playbook の YAML ファイルを確認します。手動またはソースコントロールメカニズムによって Playbook をインポートする場合、ホスト定義は Automation Controller によって制御され、**hosts:all** に設定する必要があることに注意してください。

28.5. PLAYBOOK がジョブテンプレートリストに表示されない

Playbook が **ジョブテンプレート** リストに表示されない場合は、次の点を確認してください。

- Playbook が有効な YML で、Ansible で解析できることを確認してください。
- プロジェクトパス (**/var/lib/awx/projects**) のパーミッションと所有権が、"awx" システムユーザーにファイルが表示されるようにセットアップされていることを確認してください。次のコマンドを実行して所有権を変更します。

```
chown awx -R /var/lib/awx/projects/
```

28.6. PLAYBOOK が保留状態で止まる

Playbook のジョブを実行しようとした際に、ジョブが **Pending** のままになる場合、以下の操作を試行してください。

- **supervisorctl status** を使用して、すべてのスーパーバイザーサービスが実行されていることを確認してください。
- **/var/** パーティションに 1 GB を超える空き領域があることを確認してください。**/var/** パーティションの領域が不十分な場合、ジョブは完了しません。
- Automation Controller サーバーで **automation-controller-service restart** を実行します。

問題が継続する場合には、Automation Controller サーバーで root として **sosreport** を実行し、その結果を添付した サポートリクエスト を提出してください。

28.7. 外部のデータベースを再利用するとインストールに失敗する

2 回目以降のノードのインストール中に外部データベースを再利用すると、インストールが失敗してしまう件が報告されています。

例

クラスターインストールを実行したとします。もう一度インストールを行う必要があり、2 回目のクラスターインストールは同じ外部データベースを再利用して実行するとします。その場合、この 2 回目のインストールだけが失敗します。

以前のインストールで使用した外部データベースをセットアップする場合、追加のインストールを成功させるには、クラスターノードで使用するデータベースを手動で消去する必要があります。

28.8. AUTOMATION CONTROLLER インベントリーのプライベート EC2 VPC インスタンスの表示

デフォルトでは、Automation Controller は、Elastic IP (EIP) が関連付けられている VPC のインスタンスのみを表示します。

手順

1. ナビゲーションパネルから、**Resources → Inventories** を選択します。

2. **Source** が **AWS** に設定されているグループを選択し、**Source** タブをクリックします。**Source variables** フィールドに次のように入力します。

```
vpc_destination_variable: private_ip_address
```

3. **Save** をクリックして、グループの更新をトリガーします。

これが完了すると、VPC インスタンスが表示されるようになります。

注記

インスタンスを設定する場合には、Automation Controller が、インスタンスへのアクセス権がある VPC 内で実行されている必要があります。

第29章 AUTOMATION CONTROLLER のヒントと裏技

- Automation Controller CLI ツールの使用
- Automation Controller 管理者パスワードの変更
- コマンドラインからの Automation Controller 管理者の作成
- Automation Controller で使用するジャンプホストのセットアップ
- Automation Controller の使用時に JSON コマンド用の Ansible 出力を表示する方法
- Ansbile 設定ファイルの場所特定および設定
- 全 ansible_ 変数のリストの表示
- ALLOW_JINJA_IN_EXTRA_VARS 変数
- 通知用の controllerhost ホスト名の設定
- curl でのジョブの起動
- Automation Controller の動的インベントリーソースによるインスタンスの絞り込み
- Ansible ソースから提供されるリリース前のモジュールを Automation Controller で使用する方法
- Automation Controller での callback プラグインの使用
- winrm での Windows への接続
- 既存のインベントリーファイルとホスト/グループ変数の Automation Controller へのインポート

29.1. AUTOMATION CONTROLLER CLI ツール

Automation Controller は、フル機能のコマンドラインインターフェイスを備えています。

設定と使用方法に関する詳細は、AWX Command Line Interface および AWX manage ユーティリティー セクションを参照してください。

29.2. AUTOMATION CONTROLLER 管理者パスワードの変更

インストールプロセス中に、Automation Controller によって作成された **admin** スーパーユーザーまたはシステム管理者用の管理者パスワードを入力するように求められます。SSH を使用してインスタンスにログインすると、プロンプトにデフォルトの管理者パスワードが表示されます。

このパスワードを変更する必要がある場合は、Automation Controller サーバーで root として次のコマンドを実行します。

```
awx-manage changepassword admin
```

次に、新しいパスワードを入力します。その後は入力したパスワードが Web UI の管理者パスワードとして機能します。

Django を使用して作成時にパスワード検証のポリシーを設定するには、Django パスワードポリシー を参照してください。

29.3. コマンドラインからの AUTOMATION CONTROLLER 管理者の作成

場合によっては、コマンドラインからシステム管理者 (スーパーユーザー) アカウントを作成すると便利な場合があります。

スーパーユーザーを作成するには、Automation Controller サーバーで root として次のコマンドを実行し、プロンプトに従って管理者の情報を入力します。

```
awx-manage createsuperuser
```

29.4. AUTOMATION CONTROLLER で使用するジャンプホストのセットアップ

Automation Controller が提供する認証情報は、ProxyCommand によってジャンプホストに送信されることはありません。認証情報は、トンネル接続がセットアップされている場合に、エンドノードにのみ使用されます。

ジャンプホスト経由の接続をセットアップする ProxyCommand 定義において、AWX ユーザーの SSH 設定で固定のユーザー/キーファイルを設定できます。

以下に例を示します。

```
Host tampa
Hostname 10.100.100.11
IdentityFile [privatekeyfile]

Host 10.100..
Proxycommand ssh -W [jumphostuser]@%h:%p tampa
```

インベントリー変数を使用して、Automation Controller インスタンスにジャンプホストを追加することもできます。

これらの変数は、インベントリー、グループ、またはホストレベルのいずれかで設定できます。変数を追加するには、インベントリーに移動し、選択したレベルの **variables** フィールドに次の変数を追加します。

```
ansible_user: <user_name>
ansible_connection: ssh
ansible_ssh_common_args: '-o ProxyCommand="ssh -W %h:%p -q
<user_name>@<jump_server_name>"'
```

29.5. AUTOMATION CONTROLLER の使用時に JSON コマンド用の ANSIBLE 出力を表示する方法

Automation Controller の使用時には、API を使用して JSON 形式のコマンド用の Ansible 出力を取得できます。

Ansible の出力を確認するには、https://<controller server name>/api/v2/jobs/<job_id>/job_events/ を参照します。

29.6. ANSBILE 設定ファイルの場所特定および設定

Ansible には設定ファイルは必要ありませんが、OS パッケージには多くの場合、カスタマイズできるようにデフォルトで **/etc/ansible/ansible.cfg** が含まれています。

カスタムの **ansible.cfg** ファイルを使用するには、それをプロジェクトの root に配置します。Automation Controller は、プロジェクトディレクトリーの root から **ansible-playbook** を実行し、そこでカスタムの **ansible.cfg** ファイルを見つけます。

注記

プロジェクトの他の場所にある **ansible.cfg** ファイルは無視されます。

このファイルで使用できる値については、Generating a sample ansible.cfg file を参照してください。

最初はデフォルトを使用することもできますが、デフォルトのモジュールパスや接続タイプなどをここで設定することも可能です。

Automation Controller は、いくつかの **ansible.cfg** のオプションをオーバーライドします。たとえば、Automation Controller は、SSH ControlMaster ソケット、SSH エージェントソケット、およびその他のジョブ実行ごとのアイテムを、ジョブごとの一時ディレクトリーに保存します。この一時ディレクトリーは、ジョブ実行に使用するコンテナーに渡されます。

29.7. 全 ANSIBLE_ 変数のリストの表示

Ansible は、管理下のマシンに関する "ファクト" をデフォルトで収集します。ファクトには、Playbook やテンプレートでアクセス可能です。

マシンに関するすべての使用可能なファクトを表示するには、**setup** モジュールを **ad hoc** アクションとして実行します。

```
ansible -m setup hostname
```

これにより、その特定のホストで使用可能なすべてのファクトのディクショナリーが出力されます。詳細は、information-discovered-from-systems-facts を参照してください。

29.8. ALLOW_JINJA_IN_EXTRA_VARS 変数

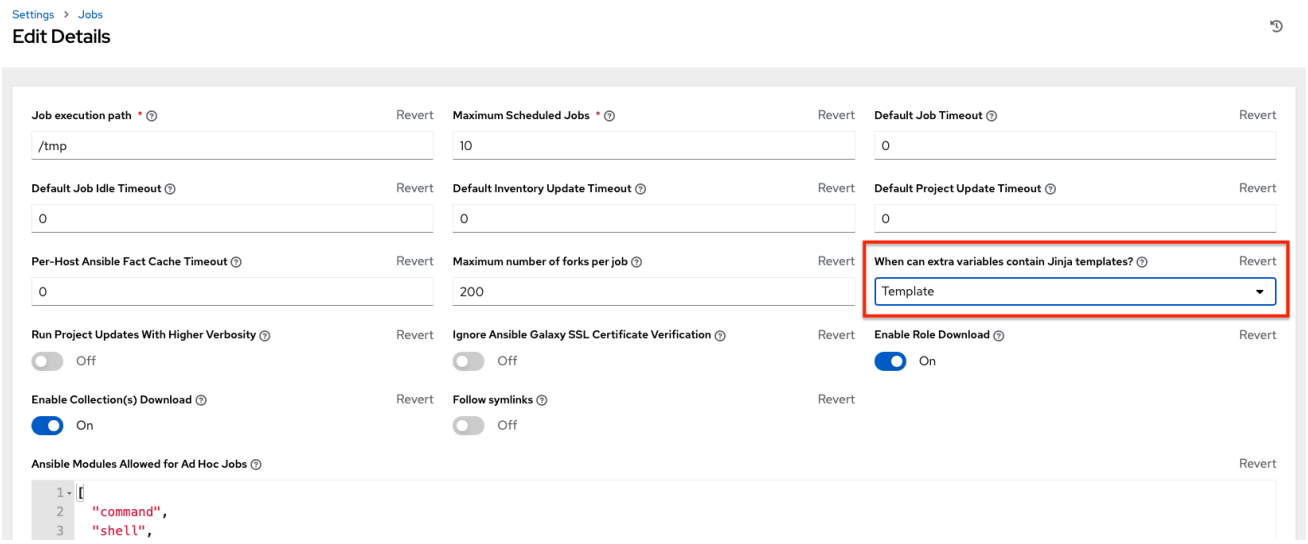
ALLOW_JINJA_IN_EXTRA_VARS = template 設定は、保存されたジョブテンプレートの追加変数に対してのみ機能します。

プロンプト変数と調査変数は 'template' の対象外です。

このパラメーターには 3 つの値があります。

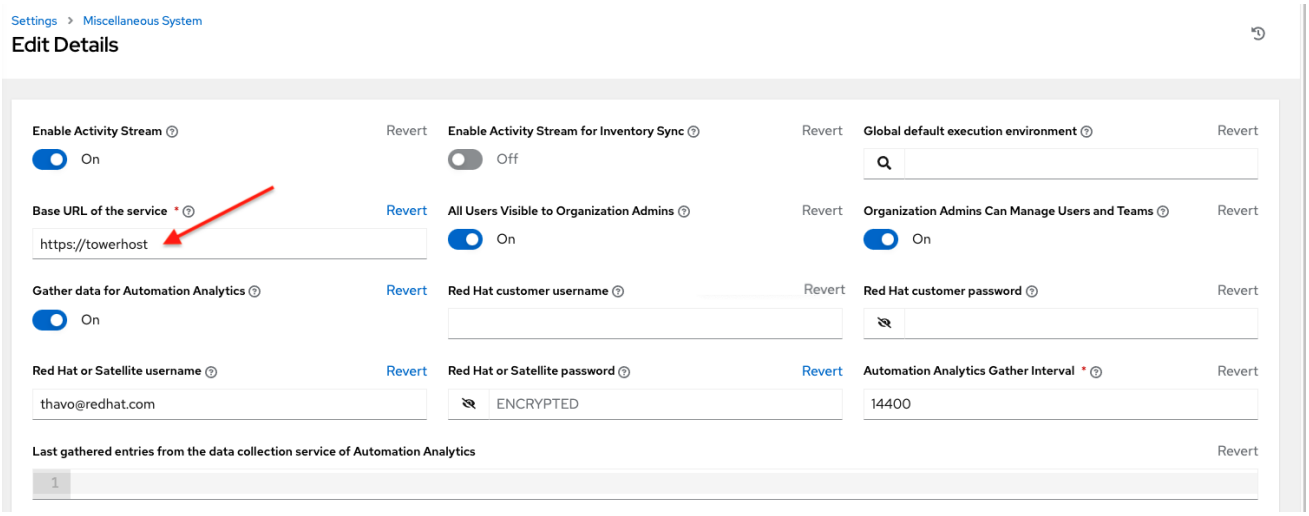
- ジョブテンプレート定義に直接保存された Jinja の使用を許可する **template** (デフォルト)。
- すべての Jinja の使用を無効にする **never** (推奨)。
- 常に Jinja を許可する **always** (極力避けてください。以前の互換性のためのオプションになります)。

このパラメーターは、Automation Controller UI の **Jobs Settings** ページで設定できます。



29.9. 通知用の CONTROLLERHOST ホスト名の設定

System settings で、**Base URL of The Controller Host**の **https://controller.example.com** を希望のホスト名に置き換えることで、通知ホスト名を変更できます。



Automation Controller のライセンスを更新すると、通知ホスト名も変更されます。Automation Controller の新規インストールでは、通知用のホスト名を設定する必要はありません。

29.10. CURL でのジョブの起動

Automation Controller API を使用してジョブを起動するのは簡単です。

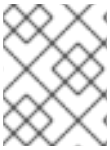
以下に、**curl** ツールを使用したわかりやすい例をいくつか示します。

ジョブテンプレート ID が '1' で、コントローラーの IP が 192.168.42.100、有効なログイン認証情報が **admin** および **awxsecret** である場合、以下のように新規ジョブを作成できます。

```
curl -f -k -H 'Content-Type: application/json' -XPOST \
    --user admin:awxsecret \
    ht p://192.168.42.100/api/v2/job_templates/1/launch/
```

これにより JSON オブジェクトが返されるので、解析し、新規作成したジョブの ID を 'id' フィールドから展開するのに使用してください。次の例のように、追加の変数をジョブテンプレートの呼び出しに渡すこともできます。

```
curl -f -k -H 'Content-Type: application/json' -XPOST \
    -d '{"extra_vars": "{\"foo\": \"bar\"}"}' \
    --user admin:awxsecret http://192.168.42.100/api/v2/job_templates/1/launch/
```



注記

extra_vars パラメーターは、単なる JSON ディクショナリーではなく、JSON を含む文字列である必要があります。引用符などをエスケープする場合は注意してください。

29.11. コントローラーの動的なインベントリーソースによるインスタンスの絞り込み

デフォルトでは、Automation Controller の動的インベントリーソース (AWS や Google など) は、使用するクラウド認証情報で利用可能なすべてのインスタンスを返します。インスタンスは、さまざまな属性に基づいて自動的にグループに参加します。たとえば、AWS インスタンスは、リージョン、タグ名、値、セキュリティーグループごとにグループ化されます。お使いの環境で特定のインスタンスを対象にするには、生成されたグループ名が対象となるように Playbook を記述します。

以下に例を示します。

```
---
- hosts: tag_Name_webserver
  tasks:
  ...
```

ジョブテンプレート設定の **Limit** フィールドを使用して、特定のグループ、ホスト、またはグループとホストの組み合わせのみを対象に Playbook を実行するように制限することもできます。構文は、ansible-playbook コマンドラインの **--limit** パラメーターと同じです。

自動生成されたグループをカスタムグループにコピーして、独自のグループを作成することもできます。動的インベントリーソースで **Overwrite** オプションが無効になっていることを確認してください。無効になっていないと、後から同期操作を行った際にカスタムグループが削除され、置き換えられます。

29.12. ANSIBLE ソースから提供されるリリース前のモジュールを AUTOMATION CONTROLLER で使用する方法

最新の Ansible Core ブランチで利用可能な機能を、Automation Controller システムに活用するのは非常に簡単です。

はじめに、利用可能な Ansible Core Module または Ansible Extra Modules GitHub リポジトリーから更新済みのモジュールで使用するものを決定します。

次に、Ansible のソース Playbook (/**library**) と同じディレクトリーレベルに新しいディレクトリーを作成します。

作成したら、使用するモジュールをコピーし、/**library** ディレクトリーにドロップします。このモジュールは、まずシステムモジュールによって消費され、通常のパッケージマネージャーで安定版を更新した後に削除できます。

29.13. AUTOMATION CONTROLLER での CALLBACK プラグインの使用

Ansible は、callback プラグインと呼ばれる方法で、Playbook の実行中にアクションを柔軟に処理できます。これらのプラグインを Automation Controller で使用すると、Playbook の実行または失敗時にサービスに通知したり、Playbook の実行後に毎回メールを送信したりできます。

callback プラグインアーキテクチャーに関する公式ドキュメントについては、Developing plugins を参照してください。

注記

Automation Controller は **stdout** callback プラグインをサポートしません。Ansible で許可されるのは 1 つだけであり、すでにイベントデータのストリーミングに使用されているためです。

https://github.com/ansible/ansible/tree/devel/lib/ansible/plugins/callback などでサンプルのプラグインも確認してみてください。これらは、各サイトの目的に合わせて変更する必要があります。

これらのプラグインを使用するには、Automation Controller プロジェクトの Playbook と併せて、callback プラグイン **.py** ファイルを **/callback_plugins** というディレクトリーに配置します。次に、ジョブ設定の **Ansible Callback Plugins** フィールドで、パス (1 行に 1 つのパス) を指定します。

注記

Ansible に同梱されているコールバックの大半をシステム全体に適用するには、それらを **ansible.cfg** の **callback_whitelist** セクションに追加する必要があります。

カスタムのコールバックがある場合は、Enabling callback plugins を参照してください。

29.14. WINRM での WINDOWS への接続

デフォルトでは、Automation Controller はホストへの **ssh** 接続を試みます。

Windows ホストが属するグループ変数には、**winrm** 接続情報を追加する必要があります。

まず、ホストが属する Windows グループを編集し、グループのソース画面または編集画面に変数を配置します。

winrm 接続情報を追加するには、以下を実行します。

- Windows サーバーを含むグループ名の編集アイコン をクリックして、選択したグループのプロパティーを編集します。"variables" セクションで、**ansible_connection: winrm** という接続情報を追加します。

完了したら、編集内容を保存します。Ansible が以前に SSH 接続を試みて失敗している場合は、ジョブテンプレートを再実行する必要があります。

29.15. 既存のインベントリーファイルとホスト/グループ変数の AUTOMATION CONTROLLER へのインポート

既存の静的インベントリーと付随のホスト変数およびグループ変数を Automation Controller にインポートするには、インベントリーが次のような構造になっている必要があります。

```
inventory/
|-- group_vars
|   `-- mygroup
|-- host_vars
|   `-- myhost
`-- hosts
```

これらのホストおよび変数をインポートするには **awx-manage** コマンドを実行します。

```
awx-manage inventory_import --source=inventory/ \
  --inventory-name="My Controller Inventory"
```

たとえば、ansible-hosts という名前のインベントリーのファイルがフラットファイルで 1 つしかない場合には、以下のようにインポートします。

```
awx-manage inventory_import --source=./ansible-hosts \
  --inventory-name="My Controller Inventory"
```

競合がある場合や、My Controller Inventory という名前のインベントリーを上書きする場合には、以下を実行します。

```
awx-manage inventory_import --source=inventory/ \
  --inventory-name="My Controller Inventory" \
  --overwrite --overwrite-vars
```

以下のようなエラーを受信した場合:

```
ValueError: need more than 1 value to unpack
```

ホストファイルおよび group_vars を保持するディレクトリーを作成します。

```
mkdir -p inventory-directory/group_vars
```

次に、 :vars のリストが含まれるグループごとに、**inventory-directory/group_vars/<groupname>** という名前のファイルを作成して、変数を YAML 形式にフォーマットします。

これにより、インポーターは変換を正しく処理するようになります。