Red Hat Ansible Automation Platform 2.4

Automation Controller ユーザーガイド

Automation controller のユーザーガイド

Last Updated: 2024-04-30

Red Hat Ansible Automation Platform 2.4 Automation Controller ユーザーガイド

Automation controller のユーザーガイド

法律上の通知

概要

このガイドでは、Red Hat Ansible Automation Platform Controller (Automation Controller) の使用方法を説明します。

目次

はじめに 8

多様性を受け入れるオープンソースの強化 9

RED HAT ドキュメントへのフィードバック (英語のみ) 10

第1章 AUTOMATION CONTROLLER の概要 11
1.1. リアルタイムの PLAYBOOK 出力と調査 11
1.2. "PUSH BUTTON" による自動化 11
1.3. 簡素化されたロールベースのアクセス制御と監査 11
1.4. クラウドと自動スケーリングの柔軟性 12
1.5. 最適な RESTFUL API 12
1.6. バックアップおよび復元 12
1.7. ANSIBLE GALAXY 統合 12
1.8. OPENSTACK のインベントリーサポート 12
1.9. リモートコマンド実行 12
1.10. システムトラッキング 12
1.11. 通知の統合 13
1.12. インテグレーション 13
1.13. カスタムの仮想環境 13
1.14. 認証の機能拡張 14
1.15. クラスター管理 14
1.16. ワークフローの機能拡張 14
1.17. ジョブの説明 14
1.18. FIPS が有効な環境でのデプロイメントのサポート 15
1.19. 組織別のホスト数の制限 15
1.20. インベントリープラグイン 15
1.21. シークレット管理システム 15

第2章 AUTOMATION CONTROLLER のライセンス、更新、およびサポート 16
2.1. 試用と評価 16
2.2. コンポーネントライセンス 16
2.3. ライセンスでのノードの数え方 16

第3章 インストール後の AUTOMATION CONTROLLER へのログイン 17

第4章 ANSIBLE AUTOMATION CONTROLLER のサブスクリプションの管理 18
4.1. サブスクリプションタイプ 18
4.2. 正式な ANSIBLE AUTOMATION CONTROLLER のサブスクリプションの取得 18
4.3. サブスクリプションマニフェストの取得 19
4.4. サブスクリプションのインポート 21
4.5. サブスクリプションの手動による追加 24
4.6. サブスクリプションの割り当て 24
4.7. 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 25
4.8. ホストアクティビティーの表示 26
4.9. ホストメトリクスユーティリティー 26

第5章 ユーザーインターフェイス 28
5.1. ビュー 28
5.2. リソースメニュー 32
5.3. アクセスメニュー 32
5.4. 管理 32
5.5. 設定メニュー 33

第6章 検索 **34**
6.1. 検索のルール 34
6.2. 並び替え 36

第7章 組織 **37**
7.1. 組織の作成 37
7.2. 組織へのアクセス 39

第8章 AUTOMATION CONTROLLER でのユーザーの管理 **44**
8.1. ユーザーの作成 44
8.2. ユーザーの削除 45
8.3. ユーザー組織の表示 46
8.4. ユーザーのチームの表示 46
8.5. ユーザーのロールの表示 46
8.6. ユーザーのトークンの作成 48

第9章 チームの管理 **50**
9.1. チームの作成 50

第10章 ユーザー認証情報の管理 **54**
10.1. 認証情報の仕組み 54
10.2. 新しい認証情報の作成 54
10.3. 既存の認証情報に新しいユーザーとジョブテンプレートを追加する 55
10.4. 認証情報タイプ 55
10.5. PLAYBOOK での AUTOMATION CONTROLLER 認証情報の使用 70

第11章 カスタム認証情報タイプ **72**
11.1. コレクションからのコンテンツ収集 72
11.2. 後方互換 API の留意事項 73
11.3. 内容検証 74
11.4. 認証情報タイプの使用開始 74
11.5. 新しい認証情報タイプの作成 74

第12章 シークレット管理システム **80**
12.1. シークレットルックアップの設定とリンク 80

第13章 アプリケーション **91**
13.1. アプリケーションの使用開始 91
13.2. 新規アプリケーションの作成 92

第14章 実行環境 **95**
14.1. 実行環境の構築 95
14.2. ジョブテンプレートへの実行環境の追加 99

第15章 実行環境の設定リファレンス **102**
15.1. 実行環境定義の例 102
15.2. 設定オプション 102
15.3. AWX のデフォルトの実行環境 110

第16章 プロジェクト **111**
16.1. 新しいプロジェクトの追加 113
16.2. ソースコントロールからのプロジェクトの更新 118
16.3. 権限の使用 118
16.4. ANSIBLE GALAXY サポート 120
16.5. コレクションのサポート 121

第17章 プロジェクトの署名と検証 **126**
17.1. 前提条件 127
17.2. AUTOMATION CONTROLLER への GPG キーの追加 127
17.3. ANSIBLE-SIGN CLI ユーティリティーのインストール 128
17.4. プロジェクトの署名 129
17.5. プロジェクトの検証 130
17.6. 署名の自動化 131

第18章 インベントリー **133**
18.1. スマートインベントリー 134
18.2. 構築されたインベントリー 139
18.3. INVENTORY プラグイン 144
18.4. 新規インベントリーの追加 145
18.5. 完了したジョブの表示 164
18.6. アドホックコマンドの実行 165

第19章 サポートされているインベントリープラグインテンプレート **169**
19.1. AMAZON WEB SERVICES EC2 169
19.2. GOOGLE COMPUTE ENGINE 171
19.3. MICROSOFT AZURE RESOURCE MANAGER 172
19.4. VMWARE VCENTER 172
19.5. RED HAT SATELLITE 6 174
19.6. OPENSTACK 174
19.7. RED HAT VIRTUALIZATION 174
19.8. RED HAT ANSIBLE AUTOMATION PLATFORM 175

第20章 ジョブテンプレート **176**
20.1. ジョブテンプレートの作成 176
20.2. テンプレートへのパーミッションの追加 186
20.3. ジョブテンプレートの削除 188
20.4. 通知の使用 188
20.5. 完了したジョブの表示 189
20.6. ジョブテンプレートのスケジュール設定 190
20.7. ジョブテンプレートの SURVEY 191
20.8. ジョブテンプレートの起動 193
20.9. ジョブテンプレートのコピー 194
20.10. スキャンジョブテンプレート 195
20.11. クラウドインベントリーでのクラウド認証情報の使用 199
20.12. プロビジョニングコールバック 202
20.13. 追加変数 205

第21章 ジョブスライス **208**
21.1. ジョブスライスの留意事項 208
21.2. ジョブスライスの実行動作 209
21.3. ジョブスライスの検索 210

第22章 AUTOMATION CONTROLLER のワークフロー **211**
22.1. ワークフローのシナリオおよび留意事項 211
22.2. ワークフローの追加変数 214
22.3. ワークフローの状態 216
22.4. ロールベースのアクセス制御 216

第23章 ワークフロージョブテンプレート **217**
23.1. ワークフローテンプレートの作成 217
23.2. 権限の使用 222

23.3. 通知の使用 223
23.4. 完了したワークフロージョブの表示 223
23.5. ワークフロージョブテンプレートのスケジュール設定 224
23.6. ワークフロージョブテンプレートでの SURVEY 224
23.7. ワークフロービジュアライザー 224
23.8. ワークフロージョブテンプレートの起動 235
23.9. ワークフロージョブテンプレートのコピー 236
23.10. ワークフロージョブテンプレートの追加変数 236

第24章 インスタンスグループの管理 237
24.1. インスタンスグループの作成 237

第25章 AUTOMATION CONTROLLER のジョブ 241
25.1. インベントリー同期ジョブ 242
25.2. SCM インベントリージョブ 244
25.3. PLAYBOOK 実行ジョブ 245
25.4. AUTOMATION CONTROLLER の容量決定とジョブへの影響 250
25.5. ジョブブランチの上書き 253

第26章 WEBHOOK の使用 256
26.1. GITHUB WEBHOOK の設定 256
26.2. GITLAB WEBHOOK の設定 259
26.3. ペイロード出力の表示 260

第27章 通知 262
27.1. 通知の階層 262
27.2. 通知ワークフロー 262
27.3. 通知テンプレートの作成 263
27.4. 通知タイプ 263
27.5. カスタム通知の作成 272
27.6. 通知の有効化と無効化 276
27.7. 通知用のホスト名設定 277
27.8. 通知 API 278

第28章 カスタム通知でサポートされている属性 280

第29章 スケジュール 285
29.1. 新しいスケジュールの追加 285

第30章 RED HAT ANSIBLE AUTOMATION PLATFORM 修復のための RED HAT INSIGHTS のセットアップ 287
30.1. RED HAT INSIGHTS 認証情報の作成 287
30.2. RED HAT INSIGHTS プロジェクトの作成 288
30.3. INSIGHTS インベントリーの作成 289
30.4. RED HAT INSIGHTS インベントリーの修復 289

第31章 AUTOMATION CONTROLLER のベストプラクティス 292
31.1. ソースコントロールの使用 292
31.2. ANSIBLE ファイルおよびディレクトリー構造 292
31.3. 動的インベントリーソースの使用 292
31.4. インベントリーの変数管理 292
31.5. 自動スケーリング 293
31.6. 大規模なホスト数 293
31.7. 継続的インテグレーションまたは継続的デプロイ 293

第32章 セキュリティー 294
32.1. PLAYBOOK のアクセスおよび情報共有 294

32.2. ロールベースのアクセス制御 295
32.3. ロールの機能: 編集および作成 301

第33章 用語集 305
アドホック 305
Callback プラグイン 305
コントロールグループ 305
Check Mode (チェックモード) 305
コンテナーグループ 305
Credentials 305
認証情報プラグイン 305
分散ジョブ 305
外部認証情報タイプ 305
ファクト 305
Forks 306
グループ 306
グループ変数 306
ハンドラー 306
ホスト 306
ホスト指定子 306
インスタンスグループ 306
Inventory 306
インベントリースクリプト 306
インベントリーソース 306
ジョブ 306
ジョブの詳細 307
ジョブスライス 307
ジョブテンプレート 307
JSON 307
Mesh 307
メタデータ 307
ノード 307
通知テンプレート 307
通知 307
Notify (通知) 308
Organization 308
Organization Administrator 308
パーミッション 308
プレイ 308
Playbook 308
ポリシー 308
プロジェクト 308
ロール 308
シークレット管理システム 308
スケジュール 308
スライスされたジョブ 308
Sudo 309
スーパーユーザー 309
Survey 309
ターゲット認証情報 309
チーム 309
ユーザー 309
Webhook 309

ワークフロージョブテンプレート 309
YAML 309

はじめに

Red Hat Ansible Automation Platform Automation Controller にご興味をお持ちいただきありがとうございます。Automation Controller は、Ansible を利用した環境に制御、ナレッジ、および委譲の機能を追加することにより、複数層からなる複雑なデプロイメントをチームが管理する際に役立ちます。

Automation Controller ユーザーガイドには、Automation Controller で使用できるすべての機能が説明されています。Playbook、変数、タグなどの概念を含め、Ansible についてある程度の知識があることを前提としています。これらおよびその他の Ansible の概念に関する詳細は、Ansible のドキュメントを参照してください。

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 の概要

Ansible Automation Platform を使用すると、シンプルで強力なエージェントレスの技術実装により、組織全体のユーザーが自動化コンテンツを共有、精査、管理できるようになります。IT マネージャーは、自動化を個々のチームに適用する方法に関するガイドラインを提供できます。自動化開発者は、複雑なツールやフレームワークに準拠するための運用上のオーバーヘッドを発生させることなく、既存の知識を使用してタスクを作成できます。これは、ハイブリッドクラウドからエッジまでエンドツーエンドの自動化ソリューションをデプロイメントするための、より安全で安定した基盤です。

Ansible Automation Platform には、企業全体で自動化を定義、操作、拡張、委任できる Automation controller が含まれています。

1.1. リアルタイムの PLAYBOOK 出力と調査

Automation controller を使用すると、Playbook の実行をリアルタイムで監視し、各ホストがチェックインするときに確認できます。特定のタスクとホストの結果を確認し、詳細に調査できます。特定のプレイまたはホストを検索して、該当する結果だけを表示したり、修正が必要なエラーを特定したりできます。

1.2. "PUSH BUTTON" による自動化

Automation controller を使用すると、Web インターフェイスからお気に入りのプロジェクトにアクセスし、実行を再トリガーできます。Automation controller は入力変数を要求し、認証情報の入力を求め、ジョブを開始および監視し、結果とホスト履歴を表示します。

1.3. 簡素化されたロールベースのアクセス制御と監査

Automation controller を使用すると、次のことが可能になります。

- **ロールベースのアクセス制御 (RBAC)** を使用して、複数の異なるチームや明示的なユーザーに対し、特定のタスクを実行する権限を付与します。タスクの例には、ファイルの表示、作成、変更などがあります。
- 一部のプロジェクトを非公開にして、一部のユーザーがインベントリーを編集できるようにし、他のユーザーがチェック (ドライラン) モードまたはライブモードで特定のシステムに対して Playbook を実行できるようにします。
- 特定のユーザーが認証情報を公開せずに認証情報を使用できるようにします。

Automation Controller は、編集されたオブジェクトや起動されたジョブなど、操作の履歴と操作の実行者を記録します。

ユーザーまたはチームにジョブテンプレートを使用するパーミッションを付与する場合は、ジョブテンプレートにパーミッションを直接割り当てることができます。認証情報は Automation controller の RBAC システム内の完全なオブジェクトであり、複数のユーザーまたはチームに割り当てて使用できます。

Automation Controller には **監査者** タイプが用意されています。システムレベルの監査者は、システム自動化のすべての側面を確認できますが、自動化を実行または変更する権限はありません。監査者は、REST API から自動化情報を収集するサービスアカウントに役立ちます。

関連情報

- ユーザーロールの詳細は、ロールベースのアクセス制御 を参照してください。

1.4. クラウドと自動スケーリングの柔軟性

Automation Controller には、ノードがオンデマンドで設定を要求できるようにする強力なオプションのプロビジョニングコールバック機能が組み込まれています。これはクラウドの自動スケーリングシナリオに最適なソリューションであり、次の機能が含まれています。

- Cobbler のようなプロビジョニングサーバーと統合され、稼働時間が予測できないマネージドシステムに対処します。
- リモートノードに管理ソフトウェアをインストールする必要はありません。
- コールバックソリューションは、**curl** または **wget** の呼び出しによってトリガーでき、**init** スクリプト、キックスタート、または preseed に埋め込むことができます。
- インベントリーにリストされているマシンのみが設定を要求できるようにアクセスを制御できます。

1.5. 最適な RESTFUL API

Automation controller REST API は、システム管理アプリケーションにとって理想的な RESTful API であり、すべてのリソースが完全に検出可能でページ分割され、検索可能で、適切にモデル化されています。スタイル付きの API ブラウザーを使用すると **http://<server name>/api/** にある API ルートから API を探索でき、すべてのリソースとの関係が表示されます。ユーザーインターフェイスで実行できることはすべて API で実行できます。

1.6. バックアップおよび復元

Ansible Automation Platform はシステムのバックアップと復元ができるため、必要に応じてインスタンスのバックアップとレプリケーションを簡単に行うことができます。

1.7. ANSIBLE GALAXY 統合

Ansible Galaxy **required.yml** ファイルをプロジェクトディレクトリーに追加することで、Automation controller は Playbook に必要なロールを Galaxy、GitHub、またはローカルのソースコントロールから自動的に取得します。詳細は、Ansible Galaxy サポート を参照してください。

1.8. OPENSTACK のインベントリーサポート

動的インベントリーのサポートは OpenStack で利用できます。これにより、OpenStack クラウドで実行されている任意の仮想マシンまたはイメージをターゲットにすることができます。

詳細は、Openstack を参照してください。

1.9. リモートコマンド実行

リモートコマンド実行を使用して、1 人のユーザーの追加、1 つのセキュリティー脆弱性の更新、障害が発生したサービスの再起動など、単純なタスクを実行します。単一の Ansible プレイとして記述できるタスクはすべて、インベントリー内のホストまたはホストのグループで実行できるため、システムを迅速かつ簡単に管理できます。RBAC エンジンと詳細な監査ログにより、どのユーザーが特定のタスクを完了したかがわかります。

1.10. システムトラッキング

ファクトキャッシュ機能を使用してファクトを収集できます。詳細は、ファクトキャッシュ を参照してください。

1.11. 通知の統合

自動化のステータスを追跡します。

次の通知を設定できます。

- ジョブテンプレート、プロジェクト、または組織全体でスタック可能な通知
- ジョブの開始、ジョブの成功、ジョブの失敗、およびジョブの承認に関するさまざまな通知 (ワークフローノードの場合)

次の通知ソースがサポートされています。

- メール
- Grafana
- IRC
- Mattermost
- PagerDuty
- Rocket.Chat
- Slack
- Twilio
- Webhook (他のツールと統合するには、任意の Webhook に POST 要求を送信)

前述の通知タイプごとに通知メッセージをカスタマイズすることもできます。

1.12. インテグレーション

Automation Controller は次の統合をサポートしています。

- Red Hat Satellite 6 の動的インベントリーソース

詳細は、Red Hat Satellite 6 を参照してください。

- Red Hat Insights の統合により、Insights Playbook を Ansible Automation Platform プロジェクトとして使用できるようになります。

詳細は、Insights 修復のセットアップ を参照してください。

- Automation Hub は Automation controller のコンテンツプロバイダーとして機能し、Automation controller デプロイメントと Automation Hub デプロイメントの両方を並行して実行する必要があります。

1.13. カスタムの仮想環境

カスタムの Ansible 環境サポートでは、さまざまなチームやジョブに対して、各種 Ansible 環境を設定し、カスタムのパスを指定できるようになりました。

1.14. 認証の機能拡張

Automation controller は以下をサポートします。

- LDAP
- SAML
- トークンベースの認証

LDAP および SAML のサポートにより、エンタープライズアカウント情報をより柔軟な方法で統合できます。

トークンベースの認証では、統合された OAuth 2 トークンのサポートを通じて、Automation controller を使用したサードパーティーのツールとサービスの認証が可能になります。

1.15. クラスター管理

クラスターグループのランタイム管理により、スケーリングの設定操作が可能になります。

1.16. ワークフローの機能拡張

複雑なプロビジョニング、デプロイメント、およびオーケストレーションのワークフローをモデル化するには、Automation controller 拡張ワークフローをいくつかの方法で使用できます。

- **ワークフローのインベントリーの上書き** ワークフローの定義時または起動時に、ワークフロー全体でインベントリーを上書きできます。Automation controller を使用すると、アプリケーションデプロイメントワークフローを定義して複数の環境で再利用できます。
- **ワークフローの収束ノード** 複雑なプロセスをモデル化する場合、続行する前に複数のステップが完了するまで待機する必要がある場合があります。Automation controller ワークフローはこれを再現でき、ワークフローステップは、前のワークフローステップが適切に完了するまで任意の数だけ待機してから続行できます。
- **ワークフローのネスト** 個々のワークフローを、より大きなワークフローのコンポーネントとして、再利用できます。たとえば、プロビジョニングとアプリケーションデプロイメントのワークフローを単一のワークフローに結合することなどが挙げられます。
- **ワークフローの一時停止と承認** ユーザーの介入が必要な承認ノードを含むワークフローを構築できます。これにより、Playbook 間でワークフローを一時停止して、ユーザーがワークフローの次のステップに進むことを承認 (または拒否) できるようになります。

詳細は、Automation controller のワークフロー を参照してください。

1.17. ジョブの説明

数千台のマシンで実行されているファクト収集または設定ジョブを取得して、Automation Controller クラスター全体に分散できるスライスに分割し、信頼性の向上、ジョブ完了の高速化、およびクラスターの使用率の向上を実現します。

たとえば、大きなスケールで 15,000 台のスイッチすべてのパラメーターを変更したり、数千ノードの RHEL 環境全体の情報を収集したりできます。

詳細は、ジョブのスライス を参照してください。

1.18. FIPS が有効な環境でのデプロイメントのサポート

Automation controller は、FIPS などの限定的なモードでデプロイメントおよび実行されます。

1.19. 組織別のホスト数の制限

多くの大規模組織では、多くの組織間でインスタンスを共有しています。この機能を使用するとスーパーユーザーは、各組織に割り当て可能なライセンス付きのホストの上限を指定でき、ライセンス付きのホストすべてが 1 つの組織で使用されないようにします。Automation controller アルゴリズムでは、組織の上限および全組織の合計ホスト数の変更について考慮します。インベントリーの同期により組織がポリシーに準拠しなくなる場合、インベントリーの更新は失敗します。さらに、スーパーユーザーは、警告を付けてライセンスを余分に割り当てることができます。

1.20. インベントリープラグイン

アップストリームコレクションからの次のインベントリープラグインが使用されます。

- **amazon.aws.aws_ec2**
- **community.vmware.vmware_vm_inventory**
- **azure.azcollection.azure_rm**
- **google.cloud.gcp_compute**
- **theforeman.foreman.foreman**
- **openstack.cloud.openstack**
- **ovirt.ovirt.ovirt**
- **awx.awx.tower**

1.21. シークレット管理システム

シークレット管理システムを使用すると、Automation controller で使用するために外部認証情報が保存され、提供されるため、外部認証情報を直接提供する必要はありません。

第2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。

2.1. 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。

2.2. 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。

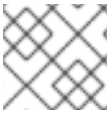
ここで、* は参照するライセンスファイル名です。

2.3. 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。

第3章 インストール後の AUTOMATION CONTROLLER へのログイン

Automation controller をインストールした後、ログインする必要があります。

手順

1. インストールの完了後に指定したログイン情報を使用して、Web ブラウザーを開き、サーバー URL (https://<CONTROLLER_SERVER_NAME>/) に移動して Automation Controller にログインします。
2. インストールプロセス中に指定した認証情報を使用してログインします。
   - デフォルトのユーザー名は **admin** です。
   - **admin** のパスワードは指定した値です。
3. 目的のユーザーの横にある **More Actions** アイコン ⋮ をクリックします。
4. **Edit** をクリックします。
5. 必要な詳細を編集し、**Save** をクリックします。

第4章 ANSIBLE AUTOMATION CONTROLLER のサブスクリプションの管理

Automation Controller を使用するには、その使用を許可する有効なサブスクリプションが必要です。

4.1. 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。

4.2. 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、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。

4.3. サブスクリプションマニフェストの取得

サブスクリプションマニフェストをアップロードするには、まずサブスクリプションの割り当てを設定します。

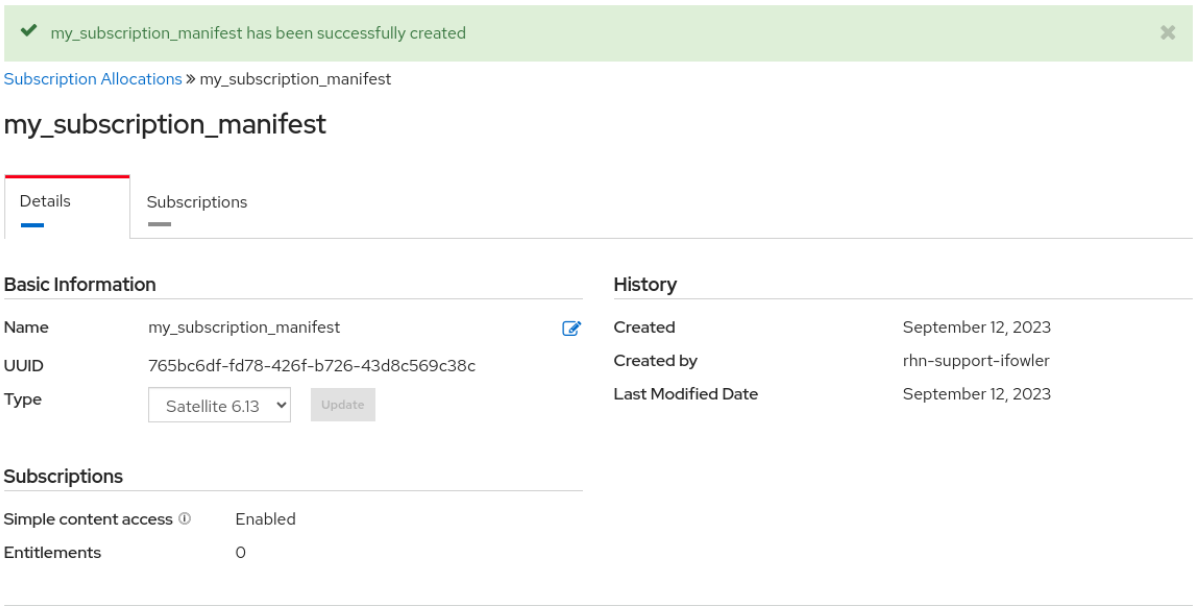
手順

1. https://access.redhat.com/management/subscription_allocations に移動します。**サブスクリプションの割り当て** ページには、作成されるまでサブスクリプションは含まれません。
2. **Create New subscription allocation** をクリックします。

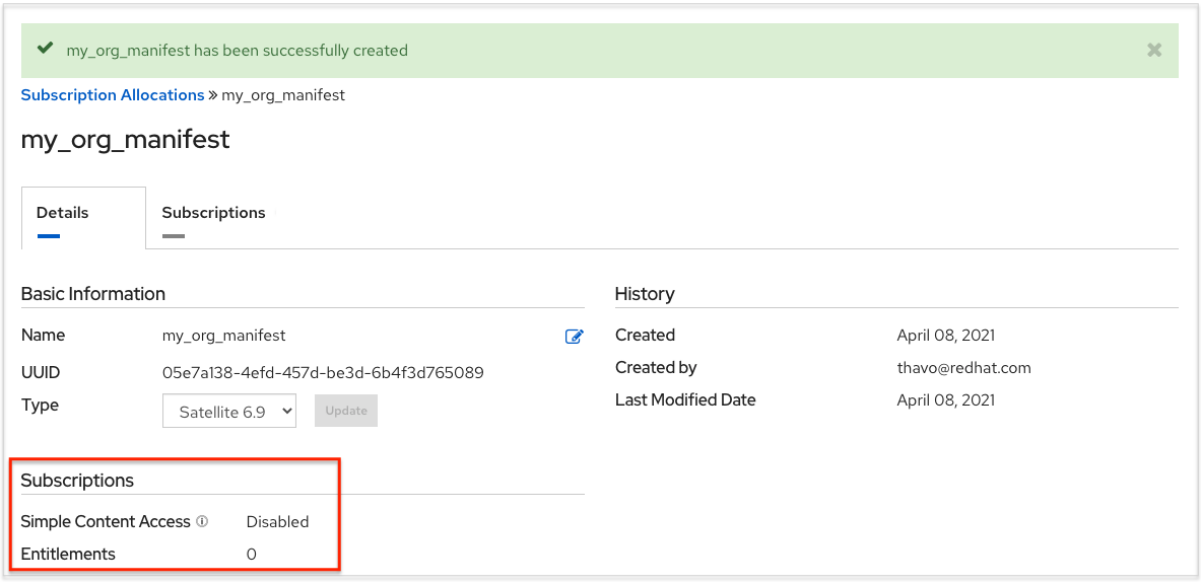
注記

Create New subscription allocationが表示されないか無効になっている場合は、サブスクリプション割り当てを作成するための適切な権限がありません。サブスクリプションの割り当てを作成するには、カスタマーポータルの管理者であるか、サブスクリプションの管理のロールが必要です。サブスクリプションを管理する権限を付与できる **access.redhat.com** の管理者または組織管理者に問い合わせてください。

3. サブスクリプションの **Name** を入力し、**Type** ドロップダウンメニューから **6.15** を選択します。



4. **Create** をクリックします。
サブスクリプションマニフェストが正常に作成されたときに **Entitlements** の横に表示される数字は、サブスクリプションに関連付けられているエンタイトルメントの数を示します。



4.3.1. サブスクリプションマニフェストの設定

サブスクリプションマニフェストを取得するには、**Subscriptions** タブを使用してサブスクリプションにエンタイトルメントを追加する必要があります。

手順

1. **Subscriptions** タブをクリックします。
2. サブスクリプションが表示されない場合は、**Add Subscriptions** をクリックします。
3. 次の画面では、マニフェストファイルに含めるエンタイトルメント選択して追加できます。

Red Hat Ansible Automation Platform for Certified Cloud and Service Providers	12003868	2019-09-05	2021-09-05	4999	
Red Hat Ansible Automation, Premium (100 Managed Nodes)	12009552	2019-09-18	2021-09-19	100	100
Red Hat Ansible Automation, Premium (100 Managed Nodes, Embedded Billing)	12009552	2019-09-18	2021-09-19	100	

サブスクリプション割り当てで複数の Ansible Automation Platform サブスクリプションを選択できます。有効な Ansible Automation Platform サブスクリプションは通常、**"Red Hat Ansible Automation..."** という名前になります。

4. マニフェストファイルに含めるエンタイトルメントまたはマネージドノードの数を指定します。これにより、サブスクリプションを分割できます (例: 1000 ノードのサブスクリプションのうち、開発クラスターに 400 ノード、実稼働クラスターに 600 ノード)。

注記

同じタイプの複数のサブスクリプションをマニフェストファイルに追加してアップロードして、複数のサブスクリプションを 1 つのインストールに適用できます。同様に、マニフェストの作成時にサブスクリプションの一部を割り当てるだけで、サブスクリプションのサブセットを適用できます。

5. **Submit** をクリックします。
指定した割り当ては、正常に追加されると、**Subscriptions** タブに表示されます。
6. **Details** タブをクリックし、サブスクリプションマニフェストファイルにアクセスします。
7. **Export Manifest** をクリックして、このサブスクリプションのマニフェストファイルをエクスポートします。先頭に **manifest_** が付いたフォルダーがローカルドライブにダウンロードされます。SKU が同じサブスクリプションが複数集約されます。
8. サブスクリプションマニフェストがある場合は、サブスクリプション画面に移動します。
9. **Browse** をクリックしてマニフェストファイル全体をアップロードします。
10. ファイルが保存されている場所に移動します。ファイルを開いたり、その個々の部分をアップロードしたりしないでください。

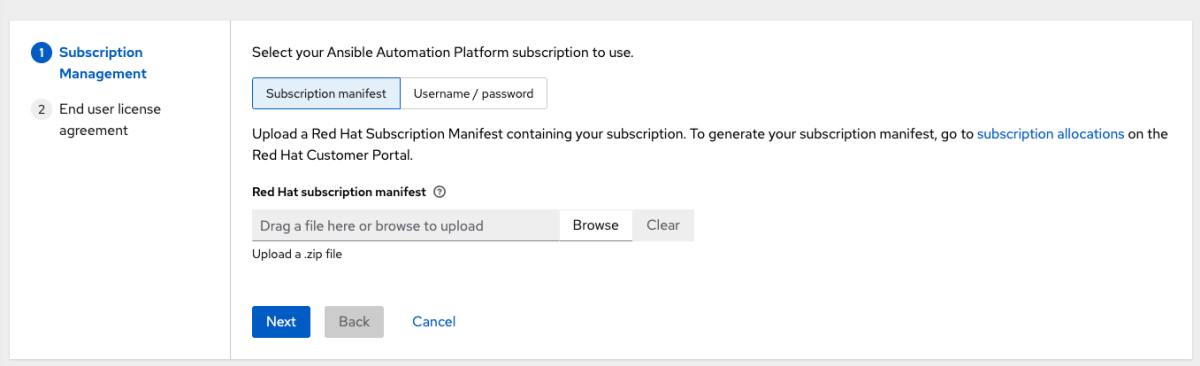
4.4. サブスクリプションのインポート

正式な Ansible Automation Platform サブスクリプションを取得したら、Automation Controller を使用する前に、サブスクリプションを Automation Controller システムにインポートする必要があります。
前提条件

- サブスクリプションマニフェストを取得している。詳細は、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。

手順

1. Automation Controller を初回起動します。**Subscription Management** 画面が表示されます。



2. 次のいずれかの手順を実行して、サブスクリプションを取得してインポートします。

   a. サブスクリプションマニフェストを取得した場合は、ファイルが保存されている場所に移動してアップロードします。アップロードするサブスクリプションマニフェストは、その一部だけでなく、**.zip** ファイル全体です。

   注記

   Subscription manifest オプションの **Browse** オプションが無効になっている場合は、**username** フィールドと **password** フィールドをクリアして有効にしてください。

   次に、サブスクリプションメタデータが RHSM/Satellite API または指定したマニフェストから取得されます。1 つのインストールに多くのサブスクリプション数が適用されると、自動化コントローラーはカウントを組み合わせますが、最も早い有効期限を有効期限として使用します（サブスクリプションを更新する必要がある時点）。

   b. Red Hat の顧客認証情報を使用している場合は、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。Automation Controller クラスターノードがサブスクリプションマネージャーを通じて Satellite に登録されている場合は、Satellite のユーザー名またはパスワードを使用します。認証情報を入力したら、**Get Subscriptions** をクリックします。Automation Controller は、ユーザーが設定したサブスクリプションサービスを取得します。続いて、実行するサブスクリプションを選択するためのプロンプトをユーザーに表示し、そのメタデータを Automation Controller に適用します。ユーザーは長期間にわたってログインし、サブスクリプションを更新した場合は、新しいサブスクリプションを取得できます。

3. **Next** をクリックして、**Tracking and Insights** ページに進みます。
   Tracking and Insights では、Red Hat 製品の改善やユーザーエクスペリエンスの向上に役立つデータを収集します。データ収集の詳細は、**Automation Controller 管理ガイド** の ユーザビリティー分析とデータ収集 を参照してください。

   このオプションはデフォルトでオンになっていますが、次のいずれかをオプトアウトできます。

   - **User analytics**。コントローラー の UI からデータを収集します。
   - **Insights Analytics**。Automation Controller を使用して、オートメーションの高度な分析を提供します。これは、コントローラーの傾向と異常な使用を特定するのに役立ちます。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。詳細は、Automation Analytics セクションを参照してください。

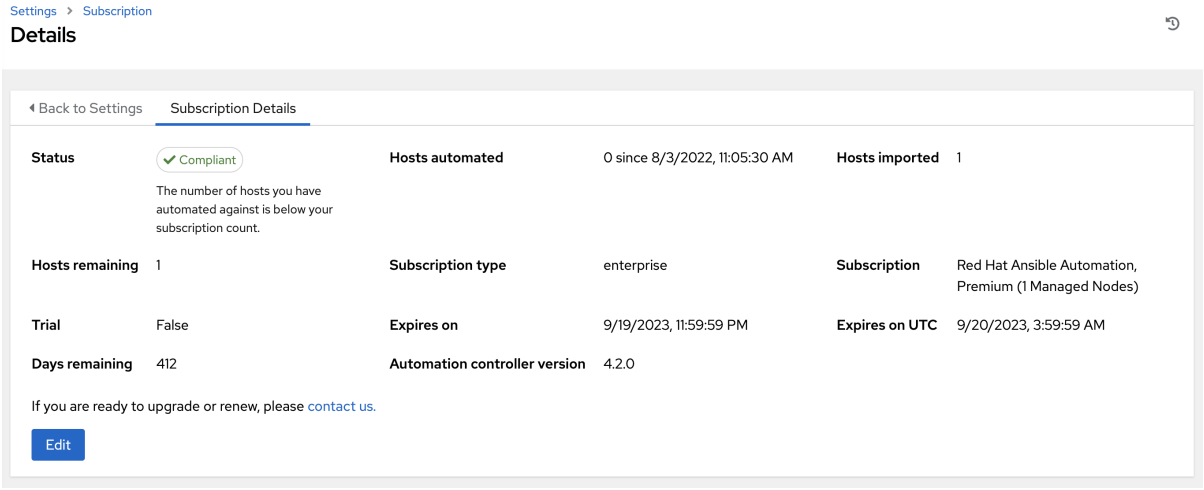
注記

分析データ収集の設定はいつでも変更できます。

4. Tracking and Insights の設定を指定したら、**Next** をクリックして使用許諾契約書に進みます。

5. 使用許諾契約書を確認し、**I agree to the End User License Agreement**チェックボックスを選択して、**Submit** をクリックします。
サブスクリプションが承認されると、Automation Controller にサブスクリプションの詳細が表示され、ダッシュボードが開きます。ダッシュボードからサブスクリプション設定画面に戻るには、ナビゲーションパネルの **Subscription** オプションから **Settings → Subscription settings** を選択します。

6. オプション：Dashboard から Subscription settings 画面に戻るには、ナビゲーションパネルの **Settings → Subscription settings** オプションを選択します。



サブスクリプションのトラブルシューティング

サブスクリプションの期限が切れると（サブスクリプション設定ウィンドウのサブスクリプションの詳細で確認可能）、Automation Controller で更新する必要があります。これを行うには、新しいサブスクリプションをインポートするか、新しいサブスクリプションを設定します。

Error fetching licenses のメッセージをすべて満たす場合は、Satellite ユーザーに必要な適切なパーミッションがあることを確認します。サブスクリプションを適用するには、Automation Controller 管理者にこのアクセス権が必要です。

Satellite のユーザー名とパスワードは、既存のサブスクリプションについて Satellite API をクエリーするために使用されます。Automation Controller は、Satellite API からそれらのサブスクリプションに関するメタデータを受信し、フィルタリングして、適用できる有効なサブスクリプションを検出します。このサブスクリプションは、有効なサブスクリプションオプションとして UI に表示されます。

以下の Satellite ロールが、適切なアクセス権を付与します。

- **view_subscriptions** および **view_organizations** フィルターを使用したカスタムロール
- Viewer
- Administrator
- Organization Administrator
- Manager

Automation Controller との統合には、最も制限が厳しい **Custom** ロールを使用します。詳細は、ユーザーとロールの管理に関する Satellite ドキュメント を参照してください。

注記

System Administrator ロールは **Administrator** user チェックボックスと同等ではないため、サブスクリプション API ページにアクセスするのに十分なパーミッションがありません。

4.5. サブスクリプションの手動による追加

Automation Controller ユーザーインターフェイスを使用してサブスクリプション情報を適用または更新できない場合は、Ansible Playbook でサブスクリプションマニフェストを手動でアップロードできます。

ansible.controller コレクション内のライセンスモジュールを使用します。

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

詳細は、Automation controller ライセンスモジュール を参照してください。

4.6. 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>
```

4.7. 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。

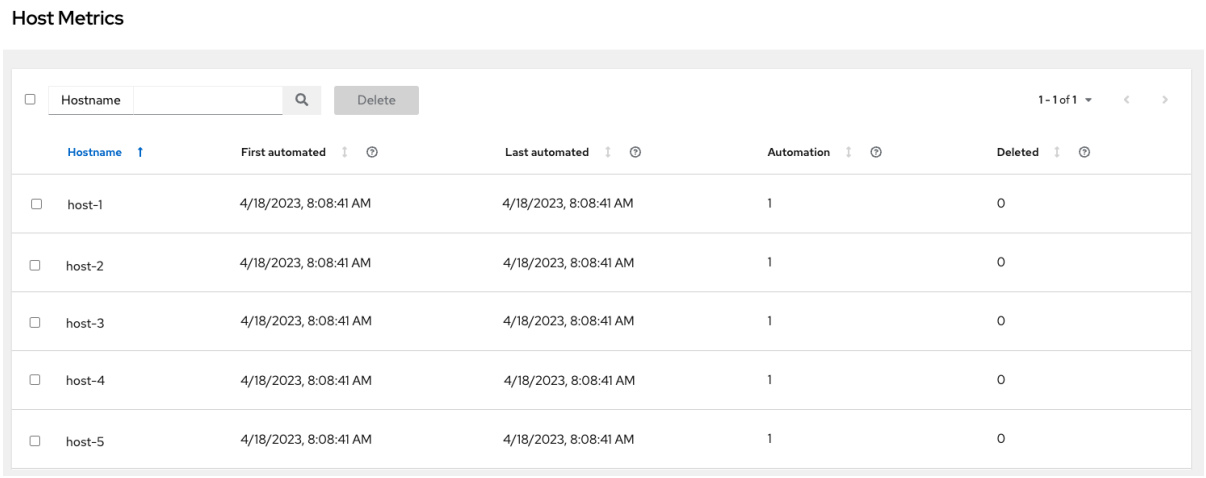
- 最初は 9 台のホストでしたが、ホストを 2 台追加してから、3 台削除しました。その結果、ホストが 8 台になりました (コンプライアンス準拠)。
- 3 台のホストを再び自動化したところ、ホストが 11 台になり、サブスクリプションの上限である 10 を超えました (コンプライアンス非準拠)。

- 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。

4.8. 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。

4.9. 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。

4.9.1. awx-manage ユーティリティー

awx-manage ユーティリティーは次のオプションをサポートしています。

```
awx-manage host_metric --csv
```

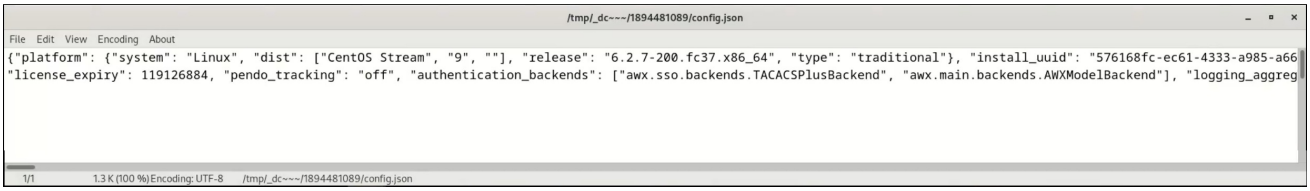
このコマンドは、ホストメトリクスデータ、ホストメトリクス概要ファイル、およびクラスター情報ファイルを生成します。配布および共有用にすべてのファイルを 1 つの tarball にパッケージ化するには、次のようにします。

```
awx-manage host_metric --tarball
```

各ファイルに出力する行数 (**<n>**) を指定するには、次のようにします。

```
awx-manage host_metric --tarball --rows_per_file <n>
```

以下は設定ファイルの例です。



Automation Analytics は JSON ファイルを受け取り、使用します。

4.9.2. API エンドポイント関数

API は削除されていないレコードのみをリストし、**last_automation** 列と **used_in_inventories** 列で並べ替えることができます。

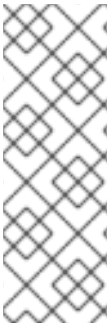
ホストメトリクス API エンドポイント **api/v2/host_metric** を使用して、ホストを論理削除することもできます。

```
api/v2/host_metric <n> DELETE
```

月次でスケジュールされたタスクでは、ホストを使用するジョブで最後に自動化されたのが 1 年以上前のジョブが、ホストメトリクステーブルから自動的に削除されます。

第5章 ユーザーインターフェイス

Automation controller **ユーザーインターフェイス** (UI) は、IT オーケストレーション要件に対応するグラフィカルフレームワークを提供します。ナビゲーションパネルを使用すると、**Projects**、**Inventories**、**Job Templates**、および **Jobs** などの Automation controller リソースに簡単にアクセスできます。



注記

Automation Controller UI は、テクニカルプレビューとして提供されており、今後のリリースで変更される可能性があります。新しい UI をプレビューするには、**Settings** メニューの **Miscellaneous System** オプションから **Enable Preview of New User Interface** をクリックして **On** に切り替えます。

保存後、ログアウトして再度ログインし、プレビューバナーから新しい UI にアクセスします。現在の UI に戻るには、示されている上部バナーのリンクをクリックします。

ユーザープロファイルや **About** ページにアクセスしたり、関連ドキュメントを表示したり、ページヘッダーのアイコンを使用してログアウトしたりできます。

アクティビティーストリームをクリックすると、そのユーザーの **アクティビティーストリーム** (アイコン) を表示できます。

5.1. ビュー

Automation controller UI には、情報を表示するためのいくつかのオプションが用意されています。

- ダッシュボードビュー
- ジョブビュー
- スケジュールビュー
- アクティビティーストリーム
- ワークフローの承認
- ホストメトリクス

5.1.1. ダッシュボードビュー

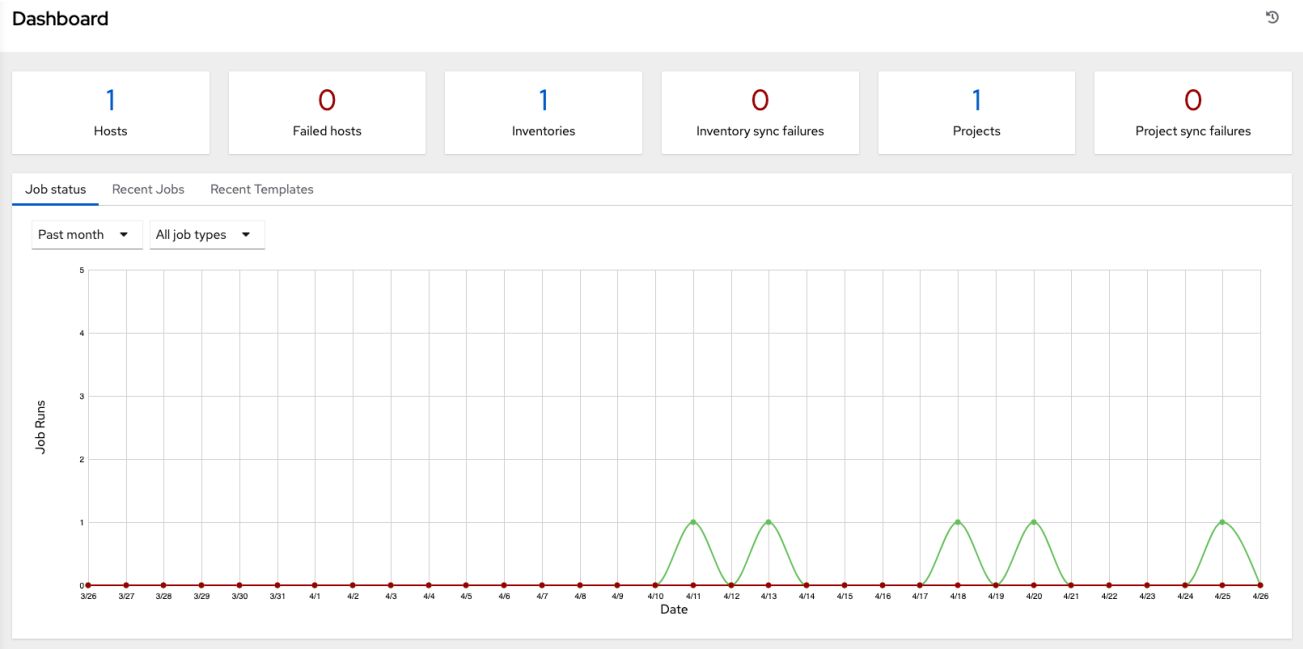
ナビゲーションメニューを使用して、次のタスクを実行します。

- さまざまなビューを表示する
- リソースに移動する
- ユーザーにアクセスを許可する
- UI で Automation Controller の機能を管理する

手順

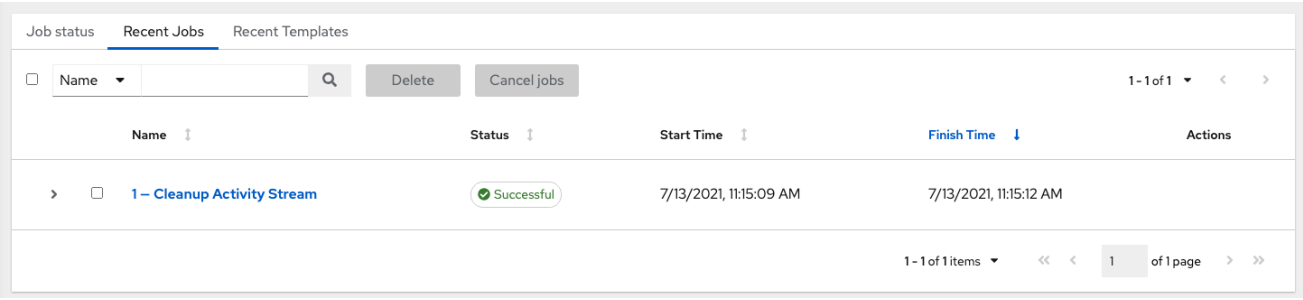
- ナビゲーションパネルから **Views** を選択して、**Views** オプションを非表示または表示します。

- ダッシュボードには、現在の **Job status** の概要が表示されます。
  - ジョブのステータスは、期間またはジョブの種類でフィルタリングできます。

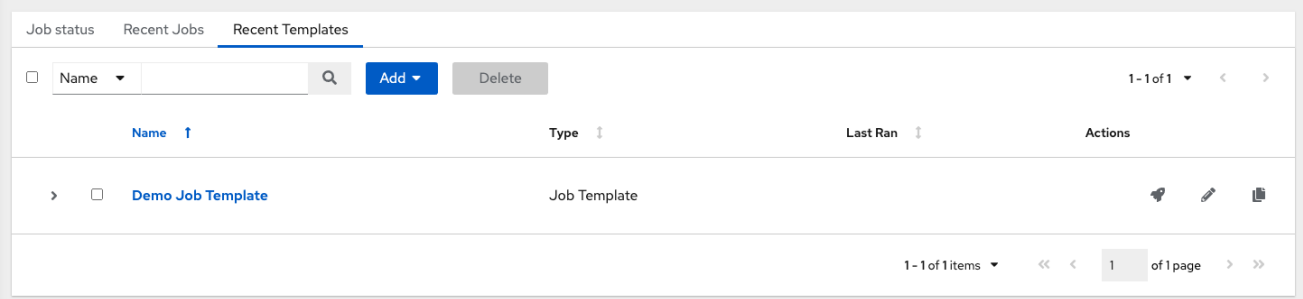


- **Recent Jobs** や **Recent Templates** の概要を表示することもできます。

Recent Jobs タブには、最近実行されたジョブ、そのステータス、および実行時間が表示されます。



Recent Templates タブには、最近使用したテンプレートの概要が表示されます。ナビゲーションメニューから **Templates** を選択して、この概要にアクセスすることもできます。

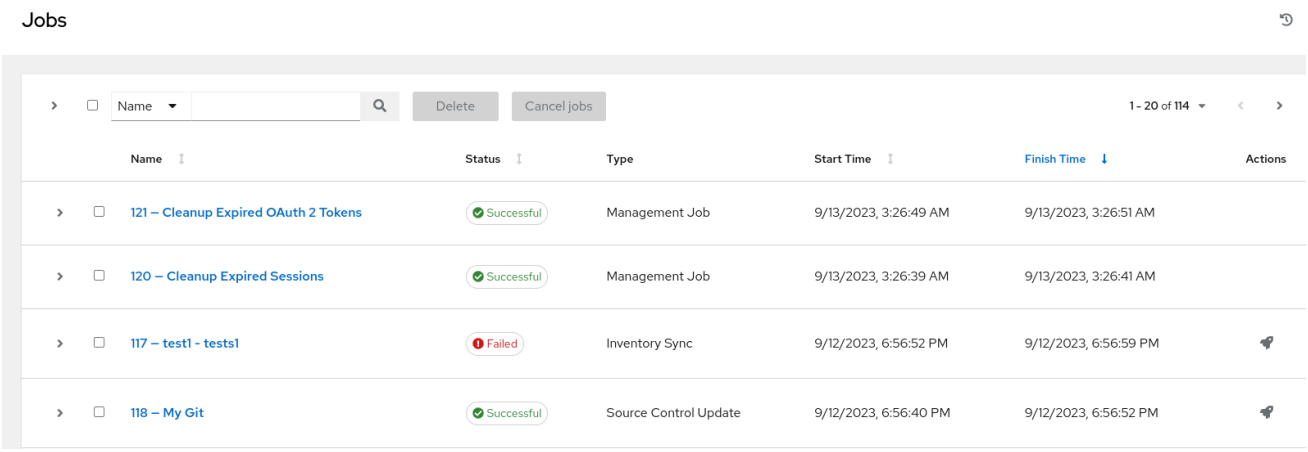


注記

ナビゲーションパネルの **Dashboard** をクリックするか、Ansible Automation Platform のロゴをクリックすると、いつでもダッシュボードに戻ることができます。

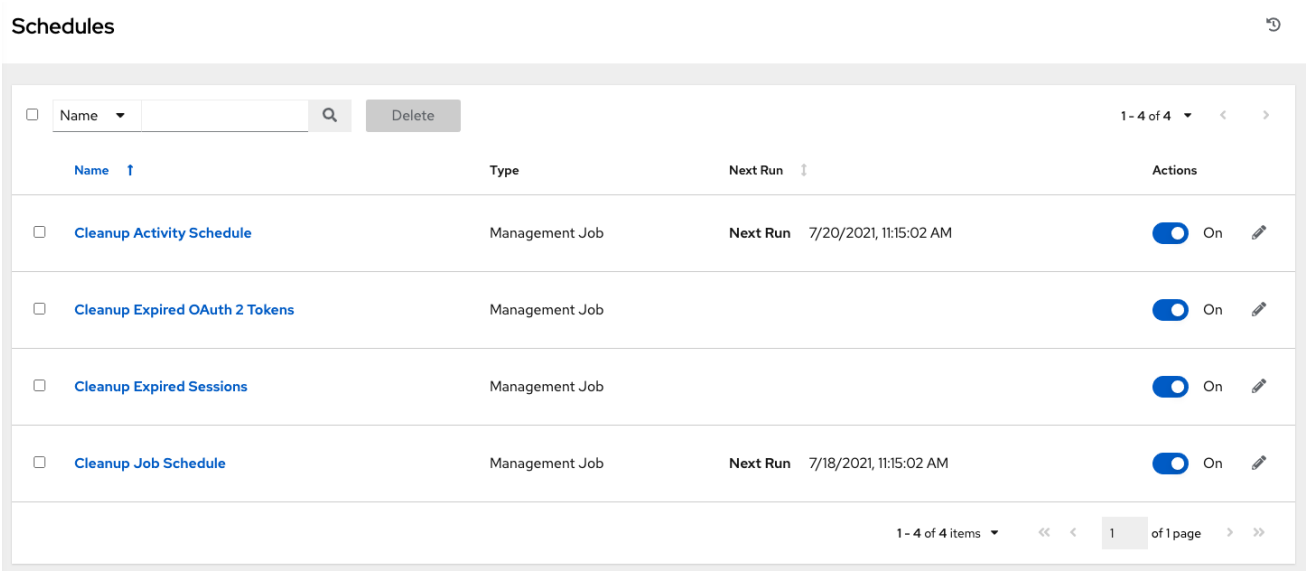
5.1.2. ジョブビュー

- ナビゲーションパネルから、**Jobs** を選択します。このビューには、プロジェクト、テンプレート、管理ジョブ、SCM 更新、Playbook の実行など、実行されたジョブが表示されます。



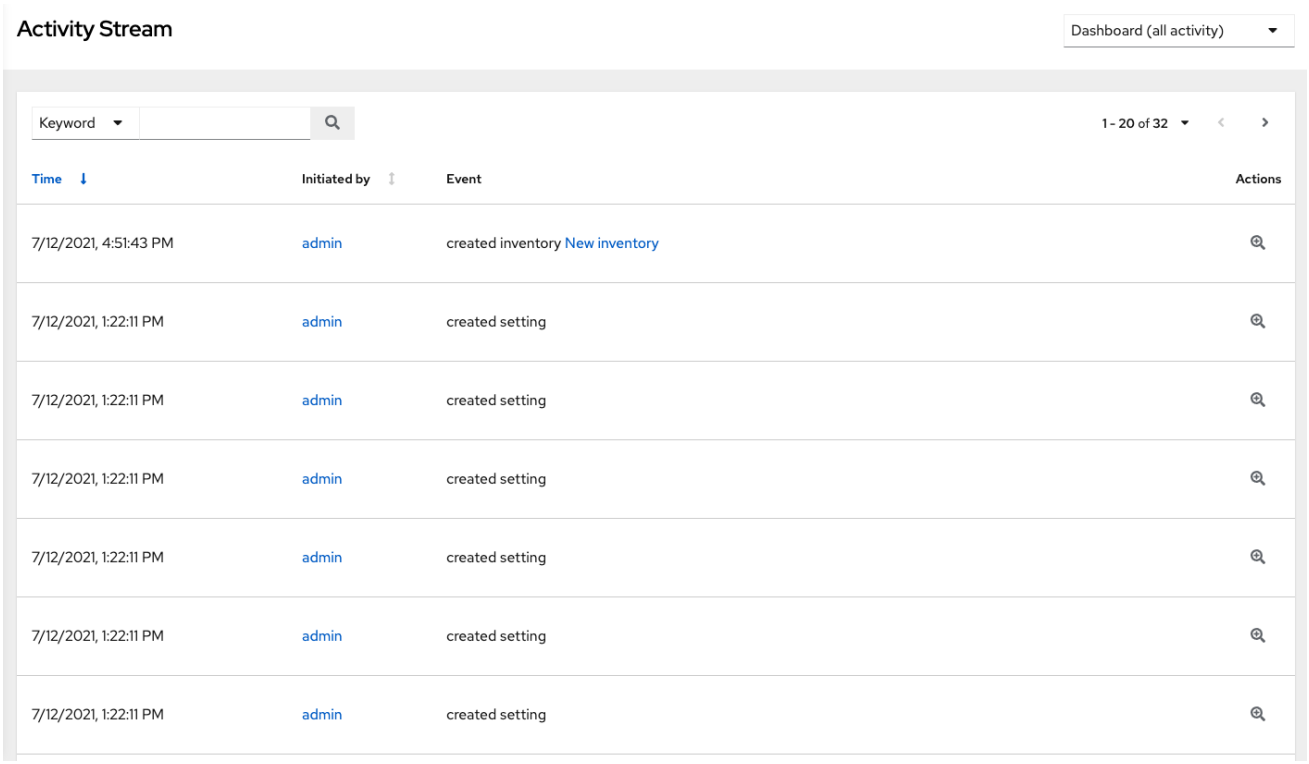
5.1.3. スケジュールビュー

このビューには、設定されているスケジュールされたジョブがすべて表示されます。

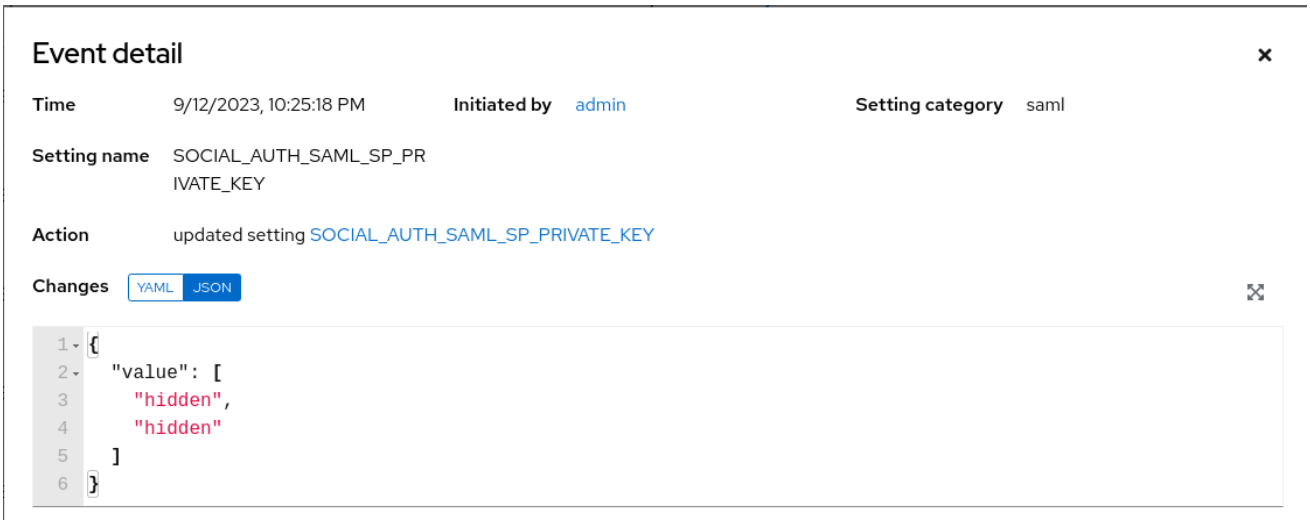


5.1.4. アクティビティーストリーム

- ナビゲーションパネルから **Activity Stream** を選択して、アクティビティーストリームを表示します。ほとんどの画面にはアクティビティーストリーム (アイコン) があります。



アクティビティーストリームには、特定のオブジェクトの変更がすべて表示されます。アクティビティーストリームには、変更ごとに、イベントの時刻、イベントを開始したユーザー、およびアクションが表示されます。表示される情報はイベントの種類によって異なります。**Examine** (アイコン) をクリックすると、変更のイベントログが表示されます。



アクティビティーストリームは、開始ユーザー、システム (システムによって開始された場合)、または認証情報、ジョブテンプレート、スケジュールなどの関連オブジェクトによってフィルタリングできます。

メインダッシュボードのアクティビティーストリームには、インスタンス全体のアクティビティーストリームが表示されます。ほとんどのページで、対象となる特定のオブジェクトに対してフィルタリングされたアクティビティーストリームを表示できます。

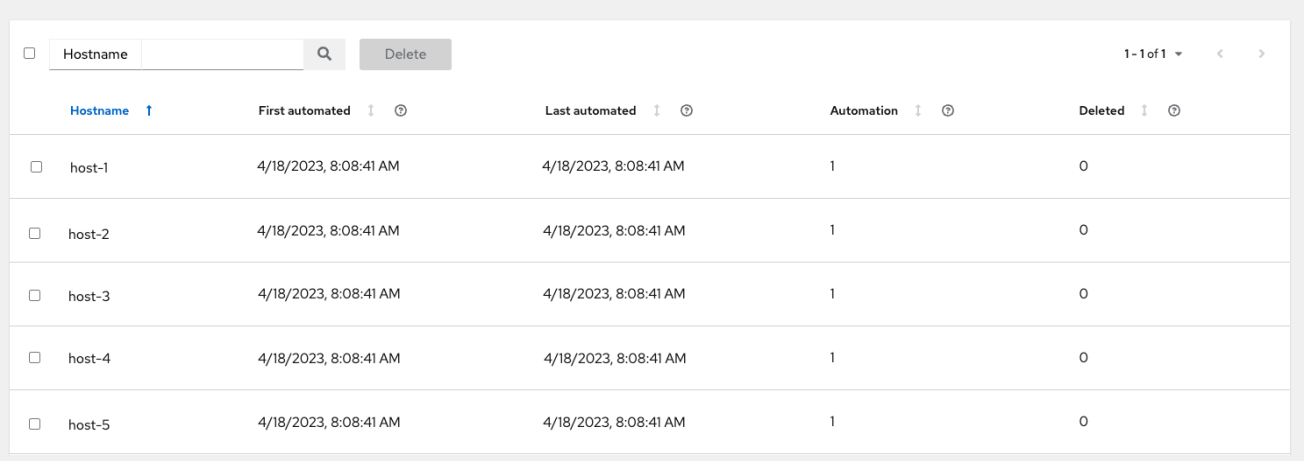
5.1.5. ワークフローの承認

- ナビゲーションパネルから **ワーWorkflow Approvals** を選択して、ワークフローの承認キューを表示します。リストには、ジョブを続行する前に承認または拒否する必要があるアクションが含まれています。

5.1.6. ホストメトリクス

- ナビゲーションパネルで **Host Metrics** を選択すると、ホストに関連付けられたアクティビティーが表示されます。これには、自動化されたもの、インベントリーで使用されたもの、削除されたものなどの数が含まれます。

Host Metrics



詳細は、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。

5.2. リソースメニュー

Resources メニューでは、Automation controller の次のコンポーネントにアクセスできます。

- テンプレート
- Credentials
- プロジェクト
- インベントリー
- Hosts

5.3. アクセスメニュー

Access メニューを使用すると、Automation controller リソースに対するパーミッションがあるユーザーを設定できます。

- 組織
- ユーザー
- チーム

5.4. 管理

Administration メニューでは、Automation controller の管理オプションにアクセスできます。ここから、以下を作成、表示、編集できます。

- 認証情報タイプ
- 通知
- Management_jobs
- インスタンスグループ
- インスタンス
- アプリケーション
- 実行環境
- Topology ビュー

5.5. 設定メニュー

Settings メニューを使用して、グローバルおよびシステムレベルの設定を設定します。**Settings** メニューでは、Automation controller 設定にアクセスできます。

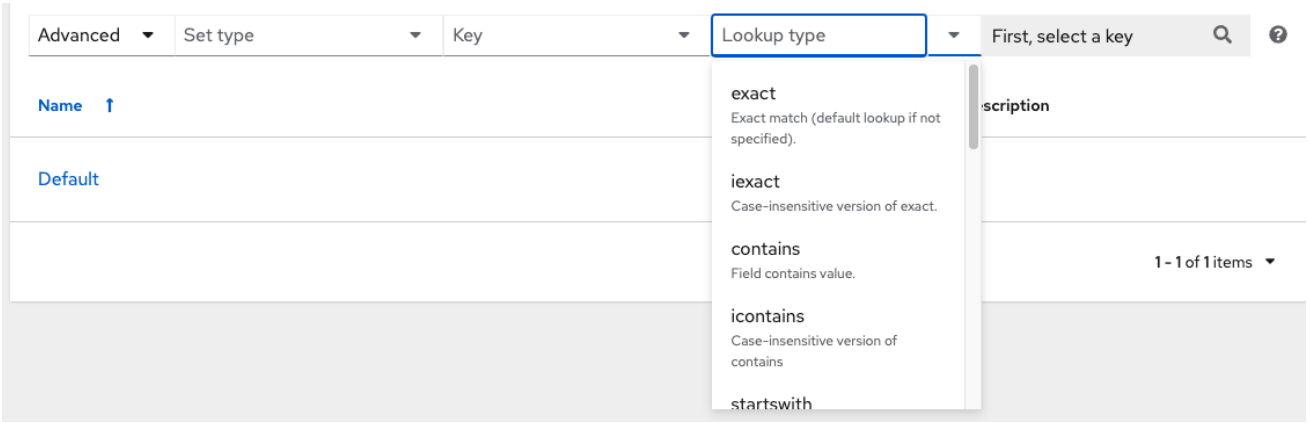
Settings ページでは、管理者は次の項目を設定できます。

- 認証
- ジョブ
- システムレベルの属性
- UI のカスタマイズ
- 製品ライセンス情報

第6章 検索

複数の機能にわたる検索およびフィルター機能については、Automation controller の検索ツールを使用します。検索フィールドの **Name** メニューの **Advanced** オプションから、リストを展開して検索条件を使用できます。

そこから、**Set Type**、**Key**、および **Lookup type** の組み合わせを使用してフィルタリングします。



6.1. 検索のルール

これらの検索のヒントは、ホストの検索を前提にはしていません。このセクションの大部分は、ホストにも適用できますが、いくつか違いがあります。

- 検索の一般的な構文は、フィールド (左側) と値 (右側) で設定されます。
- コロンは、検索するフィールドと値を区切るために使用されます。
- 検索にコロンがない場合 (例 3 を参照) **?search=foobar** が送信される単純な文字列検索として扱われます。

以下は、検索に使用される構文の例です。

1. **name:localhost** この例では、name 属性で文字列 'localhost' を検索します。この文字列が **Fields** または **Related Fields** の内容と一致しない場合は、検索全体が文字列として扱われます。

2. **Organization.name:Default** この例は、関連フィールドの検索を示しています。**Organization.name** のピリオドは、モデルとフィールドを区切ります。検索の深さまたは複雑さに応じて、クエリーの対象部分に複数のピリオドを含めることができます。

3. **foobar** これは、単純な文字列 (キーワード) 検索で、名前および説明フィールドに対して **icontains** 検索を使用して検索用語のすべてのケースを検出します。**foo bar** のように用語の間にスペースを使用すると、両方の用語を含む結果が返されます。**"foo bar"** のように用語が引用符で囲まれている場合、Automation controller は、これらの用語が一緒に出現する文字列を検索します。

固有名の検索は、API 名に対して検索を行います。例: ユーザーインターフェイスの **Management job** は API の **system_job** と同じです。**Organization:Default** この例は、関連フィールドの検索を示していますが、組織に関連するフィールドは指定されていません。これは API でサポートされており、単純な文字列検索に似ていますが、組織に対して実行されます (**icontains** は名前と説明の両方に対する検索が含まれます)。

6.1.1. 検索フィールドの値

特定のフィールドの値を見つけるには、API エンドポイントで広範なオプションとその有効な値を参照してください。たとえば、**/api/v2/jobs > type** フィールドに対して検索する場合は、**/api/v2/jobs** に対して **OPTIONS** リクエストを実行し、API で **"type"** のエントリーを検索して値を見つけることができます。さらに、各画面の一番下までスクロールすると、関連する検索を表示できます。**/api/v2/jobs** の例では、関連する検索に次の結果が表示されます。

```
"related_search_fields": [
        "modified_by__search",
        "project__search",
        "project_update__search",
        "credentials__search",
        "unified_job_template__search",
        "created_by__search",
        "inventory__search",
        "labels__search",
        "schedule__search",
        "webhook_credential__search",
        "job_template__search",
        "job_events__search",
        "dependent_jobs__search",
        "launch_config__search",
        "unifiedjob_ptr__search",
        "notifications__search",
        "unified_job_node__search",
        "instance_group__search",
        "hosts__search",
        "job_host_summaries__search"
```

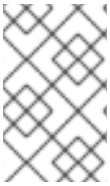
フィールドの値は、**GET** リクエストのキーから取得されます。**url**、**related**、および **summary_fields** は使用されません。関連フィールドの値も **OPTIONS** の応答から取得されますが、別の属性から取得されます。関連フィールドは、**relation_search_fields** からすべての値を取得し、末尾から **__search** を削除することによって設定されます。

フィールドの値または関連フィールドの値で始まらない検索は、一般的な文字列検索として扱われます。たとえば、**localhost** を検索すると、UI は **?search=localhost を** クエリーパラメーターとして API エンドポイントに送信します。これは、名前フィールドと説明フィールドで **icontains** を検索するショートカットです。

6.1.2. 関連フィールドの値を使用した検索

関連フィールドを検索するには、検索文字列を関連フィールドの値で始める必要があります。次の例では、関連フィールドの値、**organization** を使用して検索する方法を説明します。

検索文字列の左側は、**organization:Default** のように、**organization** で始める必要があります。関連フィールドに応じて、2 番目および 3 番目のフィールドを指定して、より具体的な検索方向を指定できます。この例としては、特定の名前に一致するプロジェクトを使用するすべてのジョブテンプレートを検索するように指定することが挙げられます。この構文は、**job_template.project.name:"A Project"** のようになります。



注記

このクエリーは、**unified_job_templates** エンドポイントに対して実行されるため、**job_template** で始まります。**job_templates** エンドポイントに対して検索を行っていた場合、クエリーの **job_template** の部分は必要ありません。

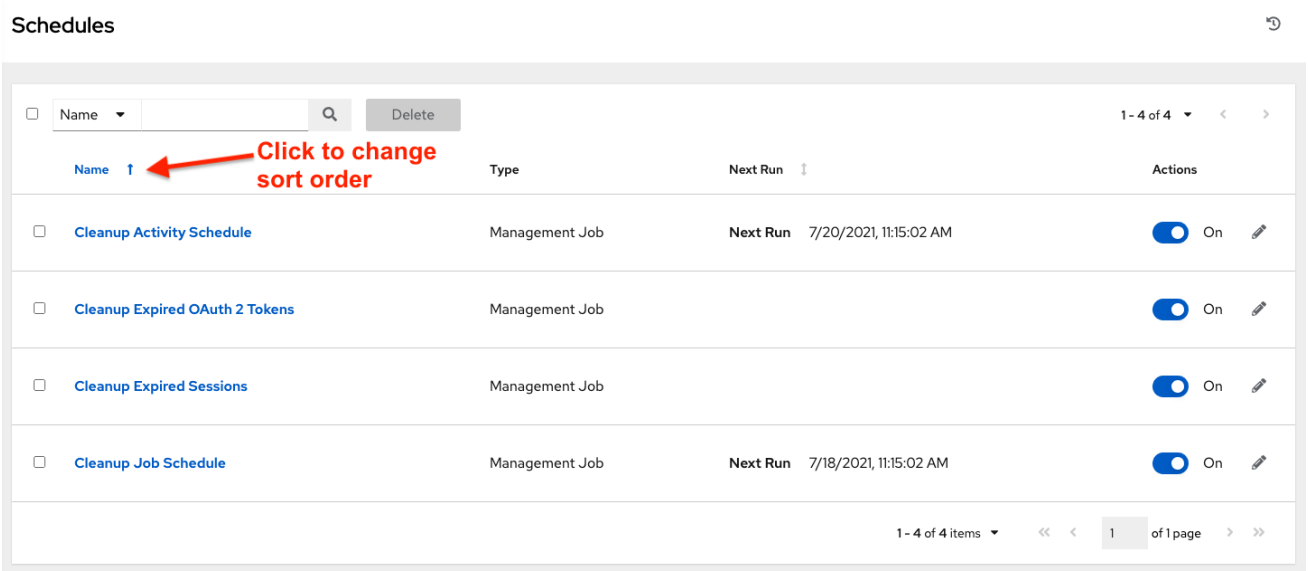
6.1.3. その他の検索に関する留意事項

Automation controller で検索する場合は、次の問題に注意してください。

- 現在、**OR** クエリーでサポートされている構文はありません。すべての検索語はクエリーパラメーター内で **AND** 演算が実行されます。
- スペースを含む文字列検索に対応するには、検索パラメーターの左側の部分を引用句で括ることができます。詳細は、検索のヒント を参照してください。
- 現在、Field 内の値は、**GET** リクエストで返されることが想定されている直接属性です。いずれかの値に対して検索するたびに、Automation controller は **__icontains** 検索を実行します。したがって、たとえば、**name:localhost** は **?name__icontains=localhost** を返します。Automation controller は現在、すべての Field 値 (**id** も含む) に対してこの検索を実行します。

6.2. 並び替え

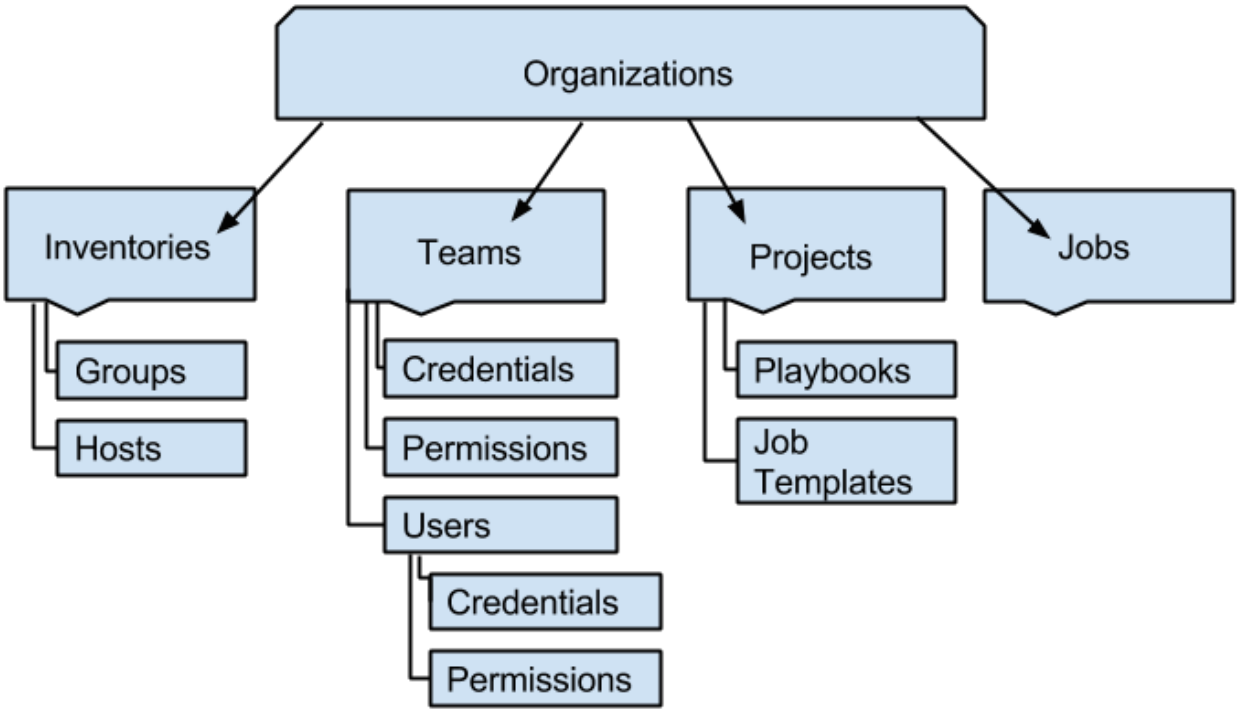
必要に応じて、各列の矢印を使用して昇順で並べ替えます。以下はスケジュールリストの例です。



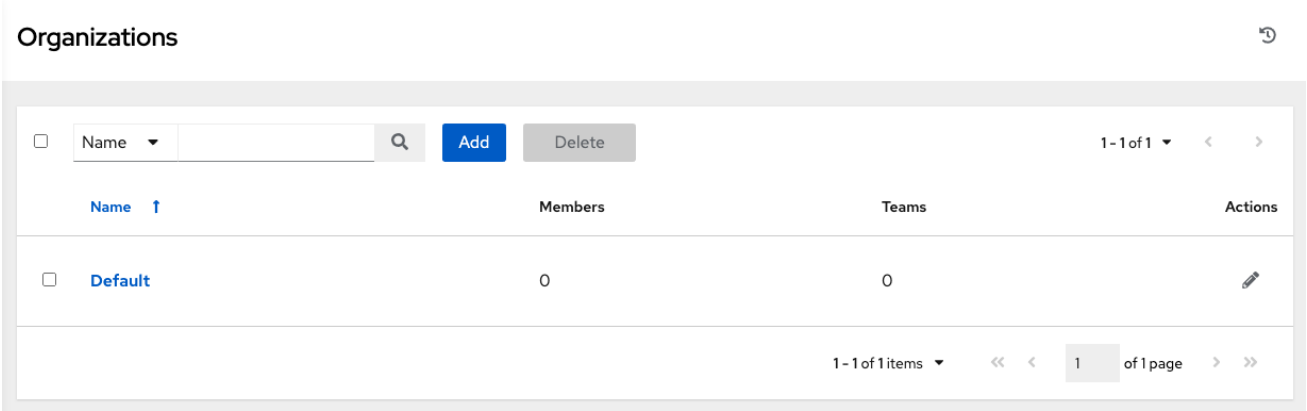
矢印の方向は、列の並べ替え順序を示します。

第7章 組織

組織は、ユーザー、チーム、プロジェクト、およびインベントリーを論理的にまとめたものです。これは、コントローラーオブジェクト階層の最上位のオブジェクトです。



ナビゲーションメニューから **Organizations** を選択し、インストールの既存の組織を表示します。



組織は **名前** または **説明** で検索できます。

アイコンを使用して組織を変更します。選択した組織を削除するには、**Delete** をクリックします。

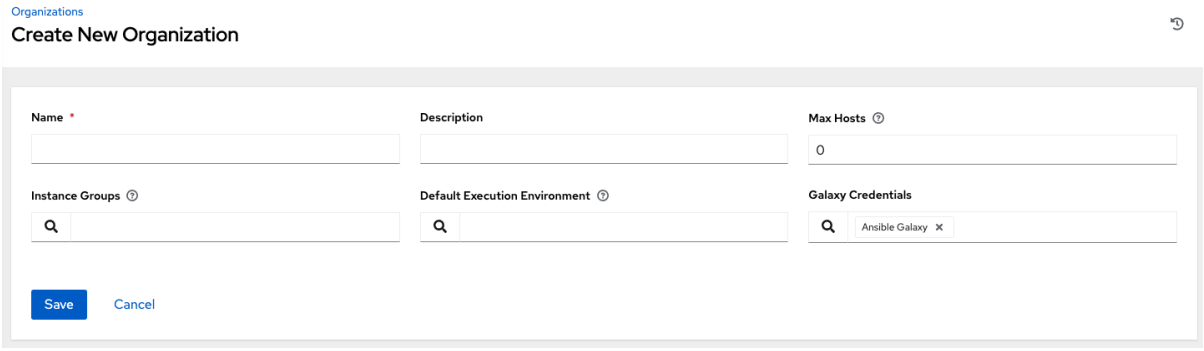
7.1. 組織の作成

注記

Automation Controller はデフォルトの組織を自動的に作成します。Self-Support レベルのライセンスをお使いの場合は、デフォルトの組織しか使用できないため、デフォルトの組織を削除しないでください。

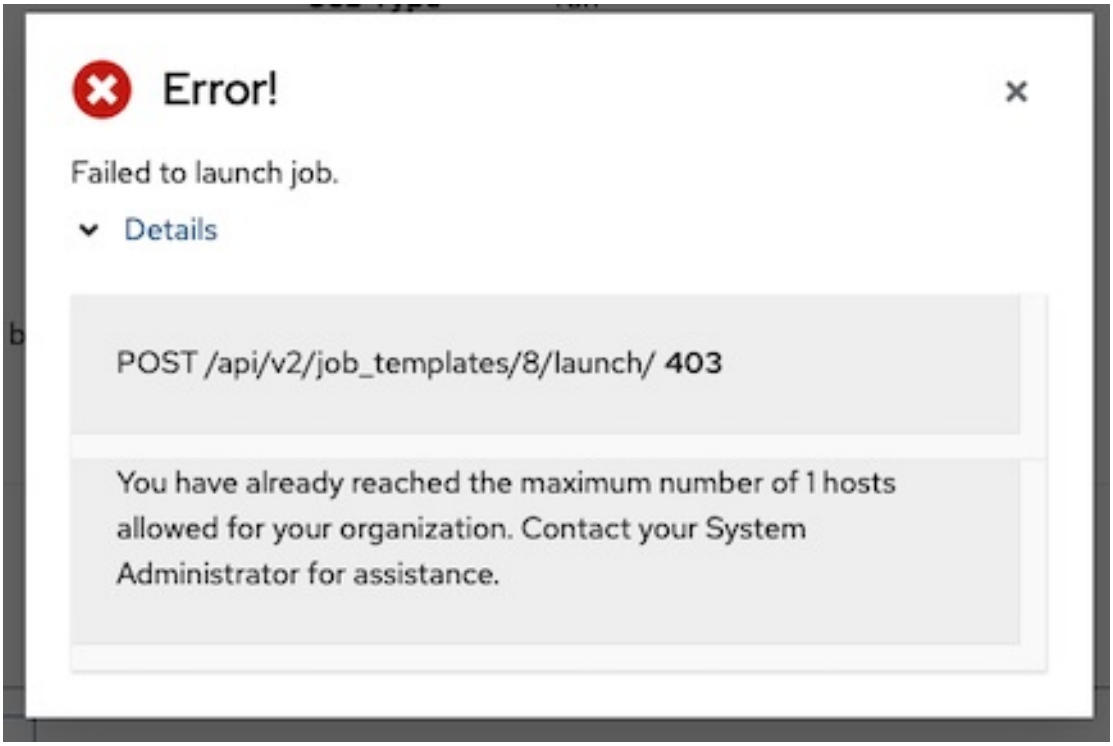
デフォルトの組織は初期設定のまま使用できます。後で編集することもできます。

1. **Add** をクリックして新しい組織を作成します。



2. 組織の属性を複数、設定できます。

   - 組織の **Name** を入力します (必須)。
   - 組織の **Description** を入力します。
   - **Max Hosts** を編集できるのはスーパーユーザーのみで、組織が保有できるライセンスホスト数の上限を設定できます。この値を 0 に設定すると、制限がないことを意味します。ホストの上限に達したか、上限を超えた組織にホストを追加しようとすると、次のエラーメッセージが表示されます。
     インベントリー同期出力ビューには、ホスト制限エラーも表示されます。

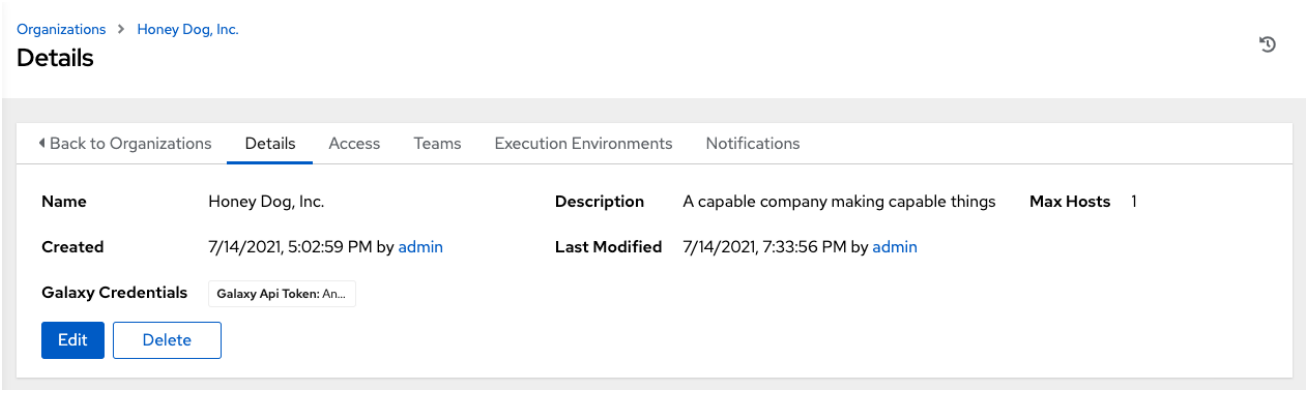


     エラーに関する追加情報を表示するには、**Details** をクリックします。

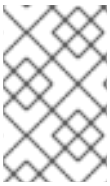
   - この組織を実行する **Instance Groups** の名前を入力します。
   - 実行環境の名前を入力するか、この組織を実行するために存在する実行環境を検索します。詳細は、実行環境へのアップグレード を参照してください。
   - 使用する場合は、**Galaxy Credentials** を入力するか、既存の認証情報のリストから検索します。

3. 組織の作成を完了するには、**Save** をクリックします。

組織が作成されると、Automation controller に組織の詳細が表示され、組織のアクセス環境と実行環境を管理できるようになります。



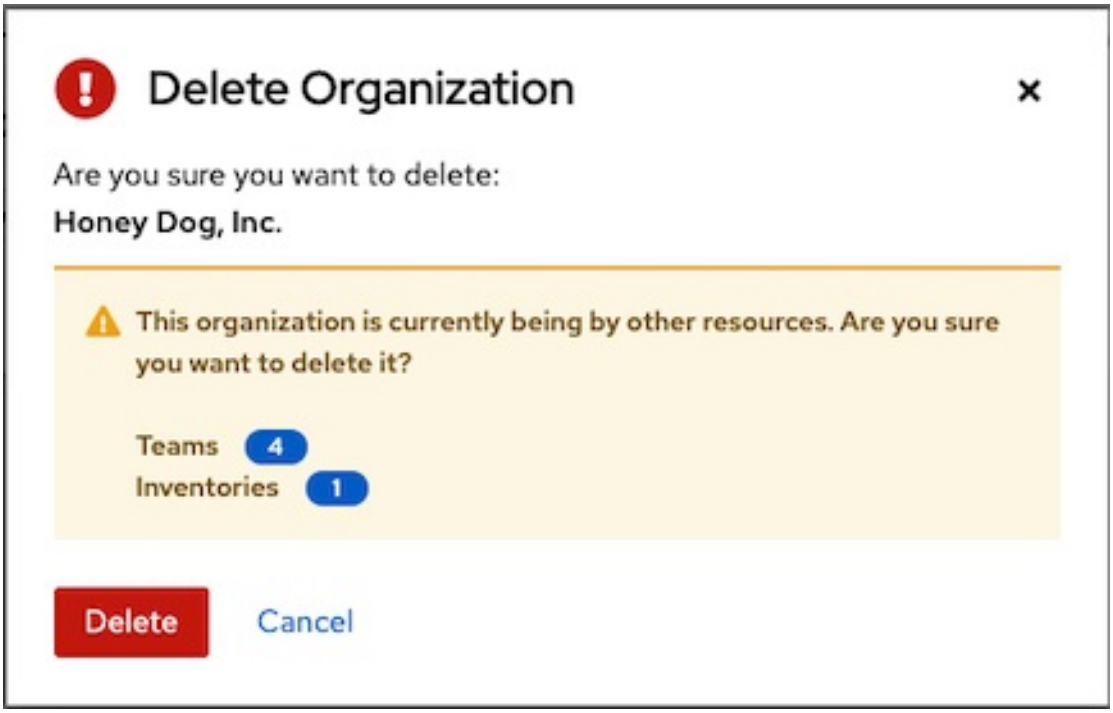
Details タブでは、組織の編集や削除を行うことができます。



注記

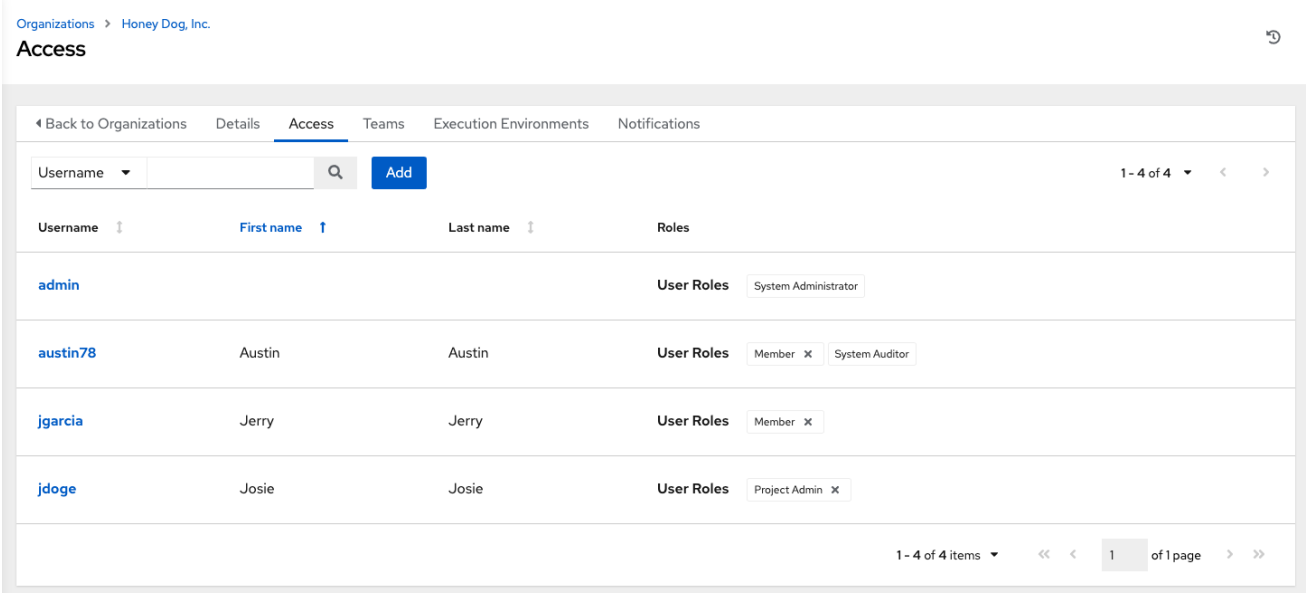
他の作業項目で使用されている項目を削除しようとすると、削除の影響を受ける項目がリストされ、削除の確認を求めるメッセージが表示されます。一部の画面には、無効な項目または以前に削除された項目が含まれているため、実行に失敗します。

以下はそのようなメッセージの例です。



7.2. 組織へのアクセス

- 組織を表示するときに **Access** を選択すると、この組織に関連付けられているユーザーとそのロールが表示されます。



このページを使用して、次のタスクを完了します。

- この組織のユーザーメンバーシップを管理します。ナビゲーションパネルで **Users** をクリックし、**Users** ページからユーザーごとにユーザーメンバーシップを管理します。
- 組織内の特定のユーザーに特定のレベルのパーミッションを割り当てます。
- 特定のリソースの管理者として機能できるようにします。詳細は、ロールベースのアクセス制御 を参照してください。

ユーザーをクリックすると、そのユーザーの詳細が表示されます。そのユーザーに関連付けられたパーミッションを確認、付与、編集、削除できます。詳細は、ユーザー を参照してください。

7.2.1. ユーザーまたはチームの追加

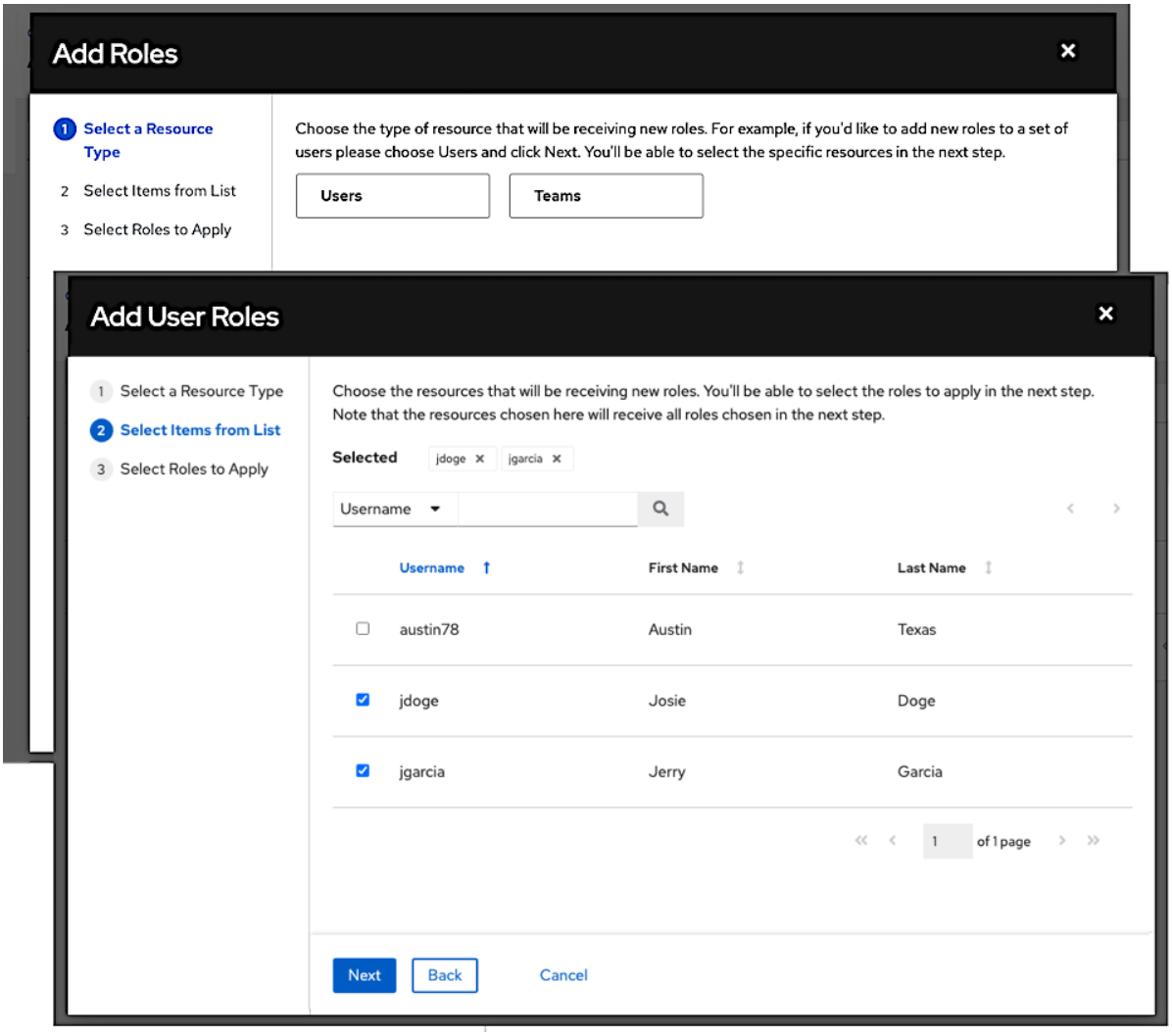
ユーザーまたはチームを組織に追加するには、そのユーザーまたはチームがすでに存在している必要があります。

詳細は、ユーザーの作成 および チームの作成 を参照してください。

既存のユーザーまたはチームを組織に追加するには以下を実行します。

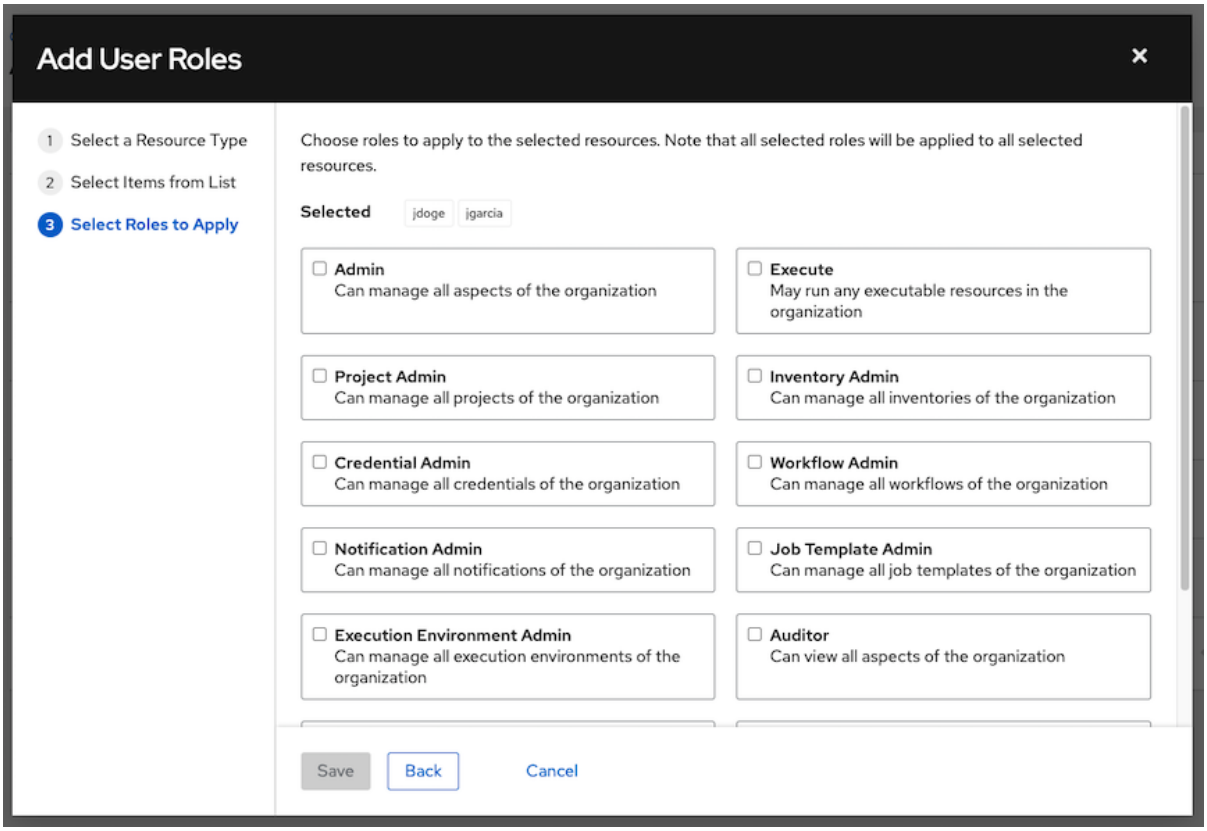
手順

1. **Organization** ページの **Access tab** で、**Add** をクリックします。
2. 追加するユーザーまたはチームを選択します。
3. **Next** をクリックします。
4. 名前の横にあるチェックボックスをクリックしてリストからユーザーまたはチームを 1 つ以上選択し、メンバーとして追加します。
5. **Next** をクリックします。

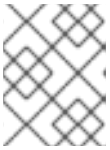


この例では、ユーザーが 2 つ選択されています。

6. 選択したユーザーまたはチームに付与するロールを選択します。下にスクロールして、ロールの完全なリストを表示します。リソースが異なれば、利用可能なオプションも異なります。



7. **Save** をクリックして、選択したユーザーまたはチームにロールを適用し、メンバーとして追加します。**Add Users** または **Add Teams** ウィンドウには、各ユーザーおよびチームに割り当てられた更新されたロールが表示されます。



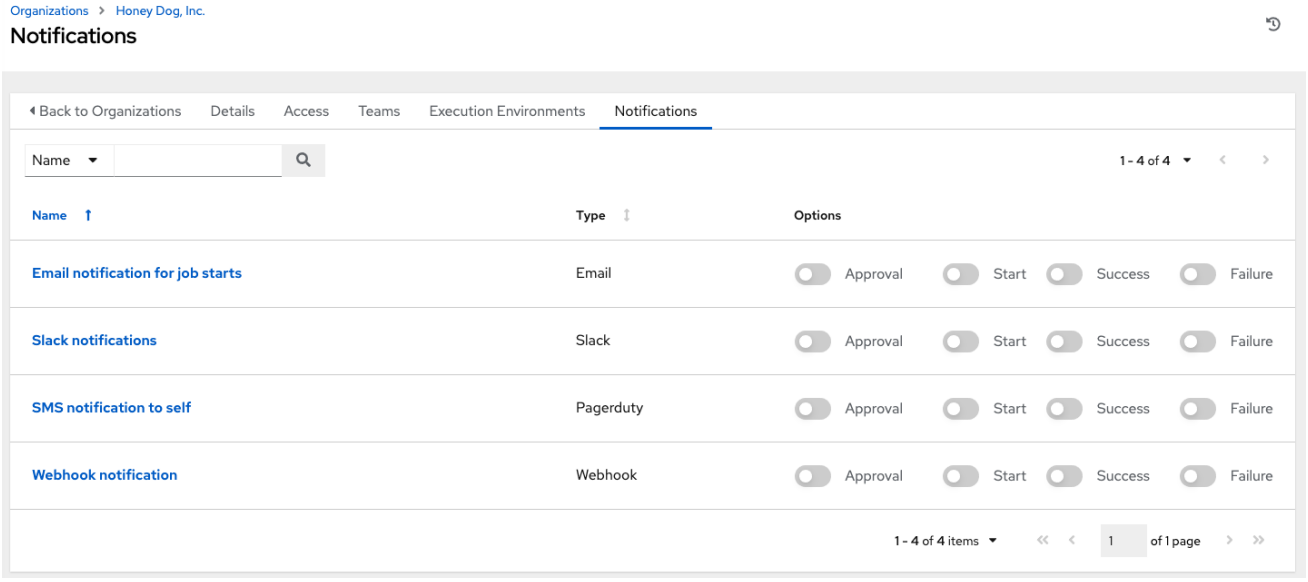
注記

ロールが関連付けられているユーザーまたはチームは、別の組織に再割り当てされた場合でも、そのロールを保持します。

8. 特定ユーザーのロールを削除するには、そのリソースの横にある関連付けの解除 (✖ アイコン) をクリックします。これにより確認ダイアログが起動し、関連付けの解除を確定するように求められます。

7.2.2. 通知の使用

Notifications タブを選択すると、設定した通知統合を確認できます。



トグルを使用して、特定の組織で使用する通知を有効または無効にします。詳細は、通知の有効化と無効化 を参照してください。

通知が設定されていない場合は、ナビゲーションパネルから **Notifications** を選択します。

通知タイプの設定についてはは、通知タイプ を参照してください。

第8章 AUTOMATION CONTROLLER でのユーザーの管理

組織に関連付けられているユーザーは、組織の **Access** タブに表示されます。

Normal User、**System Auditor**、**System Administrator** など、その他のユーザーを組織に追加することもできますが、先にそれらのユーザーを作成する必要があります。

ユーザーリストを **Username**、**First Name**、または **Last Name** で並べ替えたり検索したりできます。ヘッダーをクリックして並べ替え設定を切り替えます。

Users ページのユーザー名の横に、ユーザーパーミッションとユーザーのタイプを表示できます。

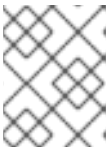
8.1. ユーザーの作成

Automation Controller で新しいユーザーを作成し、ロールを割り当てます。

手順

1. **Users** ページで、**Add** をクリックします。
Create User ダイアログが開きます。

2. 新しいユーザーに関する適切な情報を入力します。アスタリスク (*) の付いたフィールドは必須です。

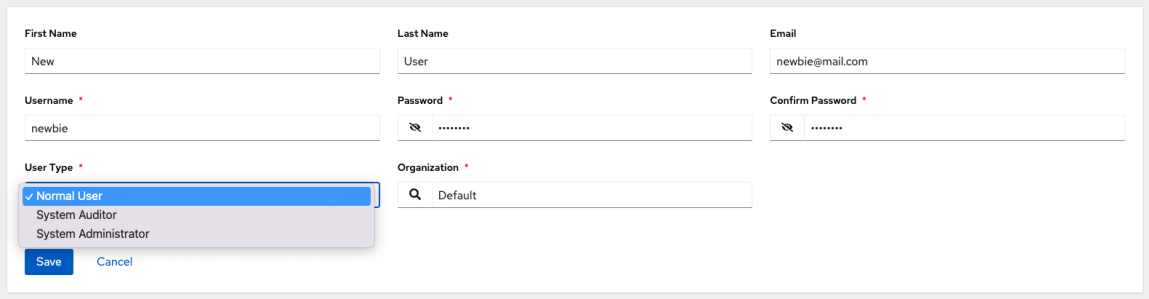


注記

自分のパスワードを変更する場合は、ログアウトして再度ログインし、変更を有効にします。

次の 3 種類のユーザーを割り当てることができます。

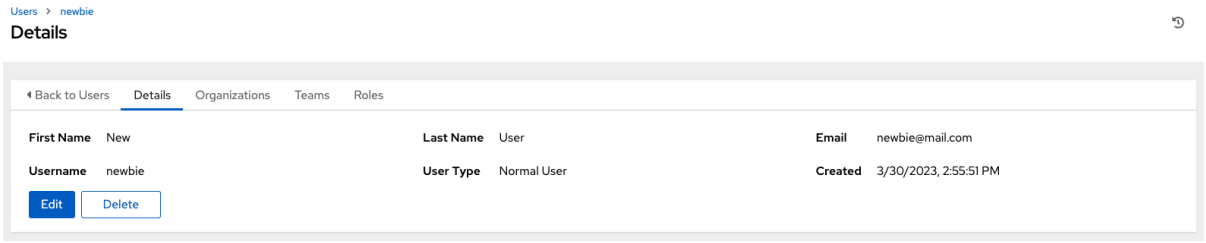
- **Normal User**: 標準ユーザーには、適切なロールや権限が付与されているリソース (インベントリー、プロジェクト、およびジョブテンプレートなど) に限定される読み取りおよび書き込みアクセスがあります。
- **System Auditor**: 監査者は、環境内のすべてのオブジェクトの読み取り専用機能を暗黙的に継承します。
- **System Administrator**: システム管理者 (スーパーユーザーとも呼ばれる) は、インストール全体に対する読み取りおよび書き込み権限をすべて持つ、完全なシステム管理権限を持っています。システム管理者は通常、日常業務のあらゆる側面を管理し、その責任をさまざまなユーザーに委任します。



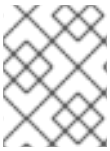
注記

インストールプロセス中に、**System Administrator** ロールを持つデフォルトの管理者が自動的に作成されます。このユーザーは、Automation Controller のすべてのユーザーが使用できます。**System Administrator** は必ず 1 つ存在させておく必要があります。**System Administrator** アカウントを削除するには、まず別の **System Administrator** アカウントを作成する必要があります。

3. **Save** をクリックします。
ユーザーが正常に作成されると、**User** ダイアログが開きます。



4. **Delete** をクリックしてユーザーを削除するか、現在のユーザーのリストからユーザーを削除できます。詳細は、ユーザーの削除 を参照してください。
ユーザー名をクリックしても、ユーザーの横にある Edit (アイコン) をクリックしても、同じウィンドウが開きます。このウィンドウを使用して、ユーザーの **Organizations**、**Teams**、**Roles**、およびその他のユーザーメンバーシップの詳細を確認および変更できます。



注記

新規作成されたユーザーでない場合、詳細画面にはそのユーザーの最後のログインアクティビティーが表示されます。

自分自身としてログインし、ユーザープロファイルの詳細を表示すると、ユーザープロファイルからトークンを管理できます。

詳細は、ユーザートークンの追加 を参照してください。

8.2. ユーザーの削除

ユーザーを削除するには、ユーザーパーミッションが必要です。ユーザーアカウントを削除すると、ユーザーの名前とメールが Automation Controller から完全に削除されます。

手順

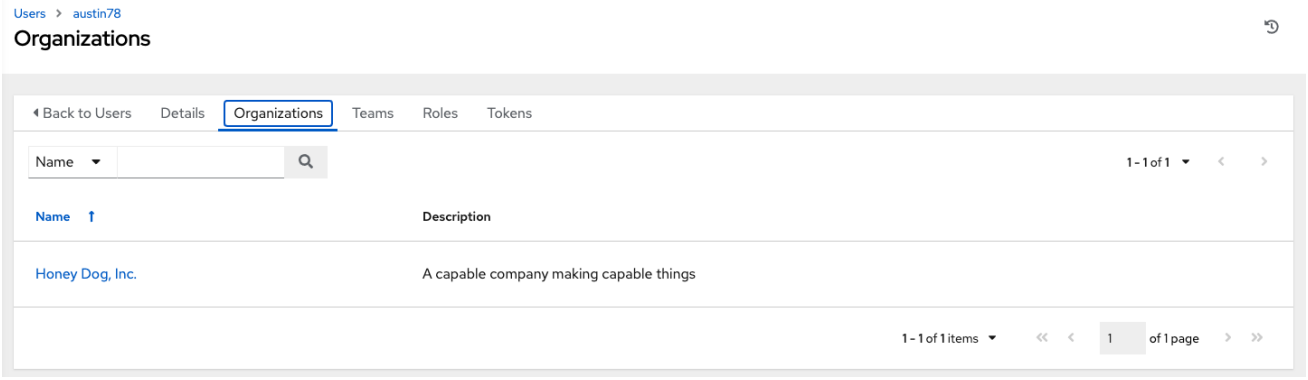
1. ナビゲーションパネルから **Access** を選択します。
2. **Users** をクリックして、現在のユーザーの一覧を表示します。
3. 削除するユーザーのチェックボックスをオンにします。
4. **Delete** をクリックします。
5. 確認警告メッセージの **Delete** をクリックして、ユーザーを完全に削除します。

8.3. ユーザー組織の表示

特定のユーザーを選択すると、**Details** ページが表示されます。**Organizations** タブを選択すると、そのユーザーがメンバーとなっている組織のリストが表示されます。

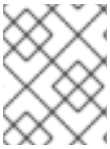
注記

組織のメンバーシップは、この表示パネルから変更できません。



8.4. ユーザーのチームの表示

Users > **Details** ページから **Teams** タブを選択して、ユーザーが所属するチームの一覧を表示します。



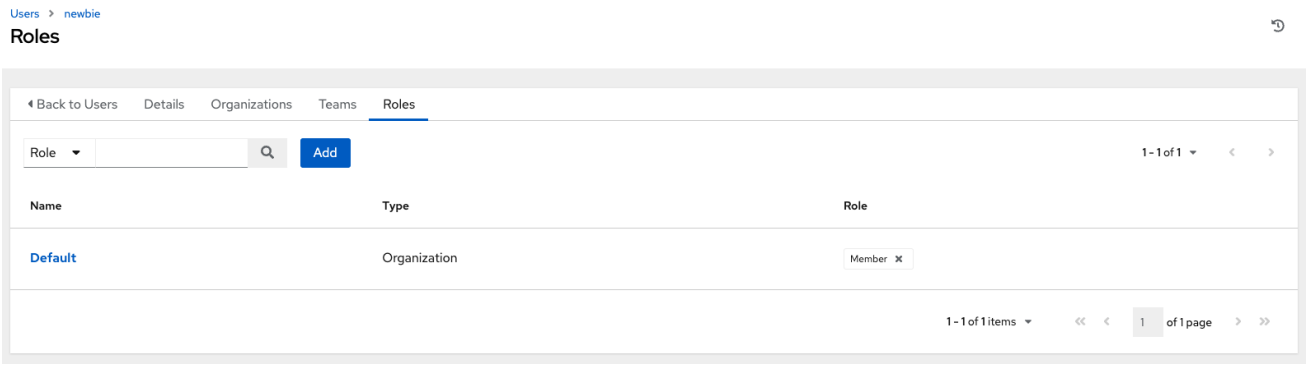
注記

チームのメンバーシップは、この表示パネルから変更できません。詳細は、チーム を参照してください。

チームを作成してそのチームにユーザーを割り当てるまで、そのユーザーに割り当てられたチームの詳細が表示されます。

8.5. ユーザーのロールの表示

Users > **Details** ページから **Roles** タブを選択して、このユーザーに割り当てられているユーザーのセットを表示します。ロールにより、プロジェクト、インベントリー、ジョブテンプレートおよびその他の要素の読み取り、変更、および管理が可能になります。

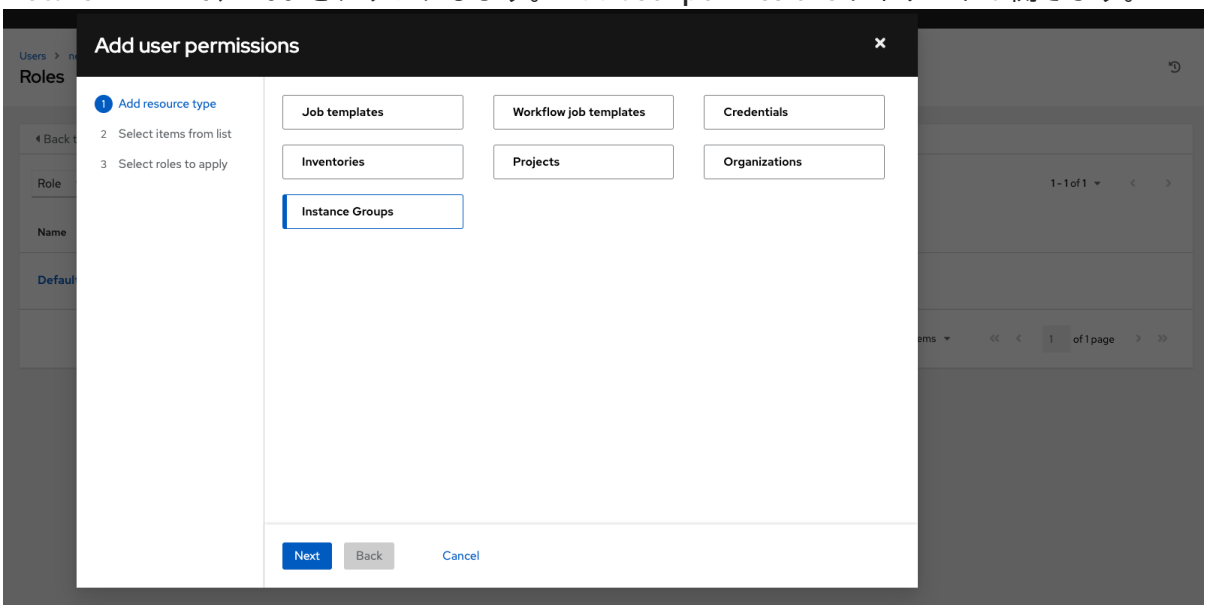


8.5.1. ユーザー権限の追加と削除

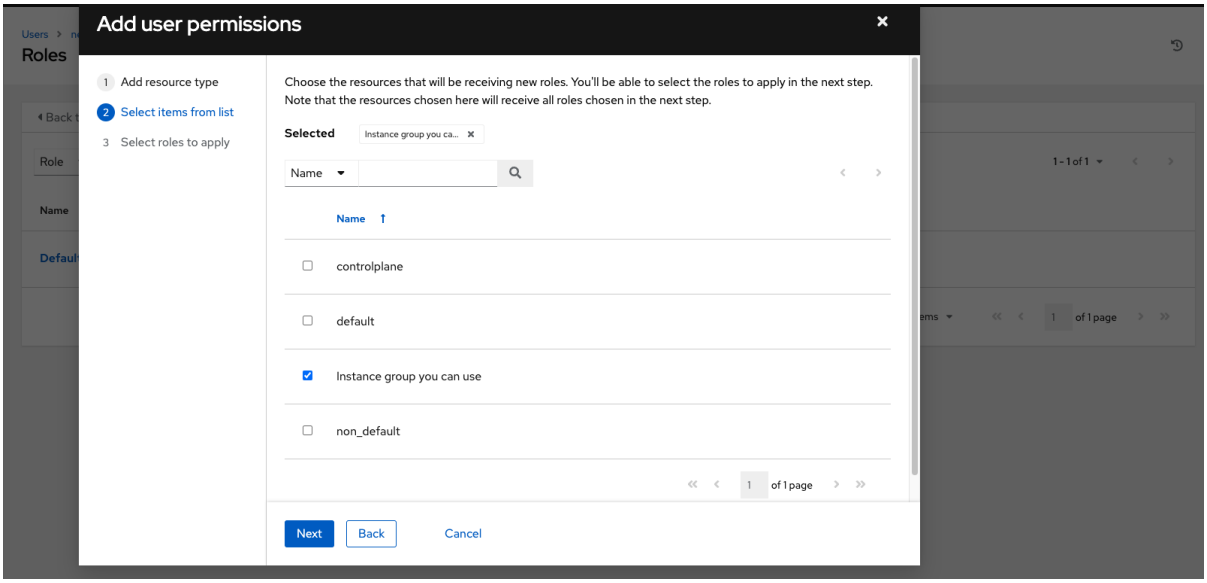
特定のユーザーに権限を追加するには、以下を実行します。

手順

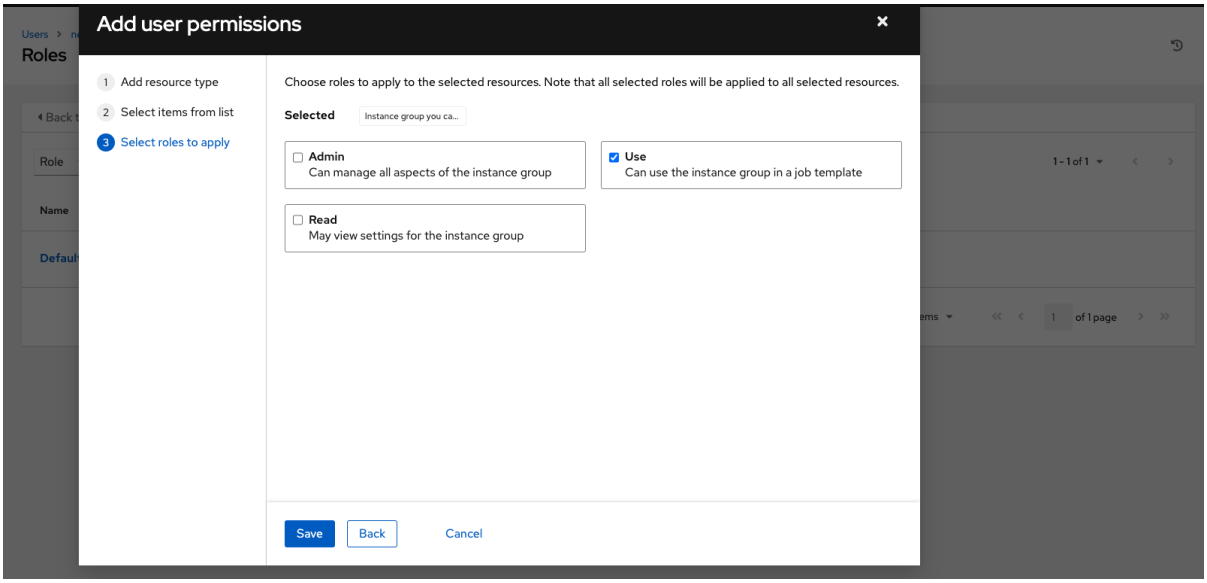
1. Users リストビューから、ユーザーの名前をクリックします。
2. Details ページで、**Add** をクリックします。Add user permissions ウィザードが開きます。



3. ユーザーがアクセスできる権限を割り当てるオブジェクトを選択します。
4. **Next** をクリックします。
5. チームロールを割り当てるリソースを選択し、**Next** をクリックします。



6. 権限を割り当てるリソースを選択します。リソースが異なれば、利用可能なオプションも異なります。



7. **Save** をクリックします。
8. **Roles** ページに、選択した各リソースに割り当てられた権限を持つユーザーの更新されたプロファイルが表示されます。

注記

チーム、個人、または複数のユーザーを追加し、オブジェクトレベルでパーミッションを割り当てることもできます。これには、テンプレート、認証情報、インベントリー、プロジェクト、組織、またはインスタンスグループが含まれます。この機能により、組織で多数のユーザーを一括でオンボーディングする時間を短縮できます。

パーミッションを削除する方法:

- リソースの横にある ✖ アイコンをクリックします。これにより確認ダイアログが起動し、関連付けの解除を確定するように求められます。

8.6. ユーザーのトークンの作成

Tokens タブは、自分用に作成したユーザーに対してのみ表示されます。

ユーザーのトークンを追加する前に、トークンをアプリケーションに関連付ける場合は、アプリケーションを作成する ことを推奨します。

アプリケーションに関連付けずに **Personal Access Token** (PAT) を作成することもできます。

手順

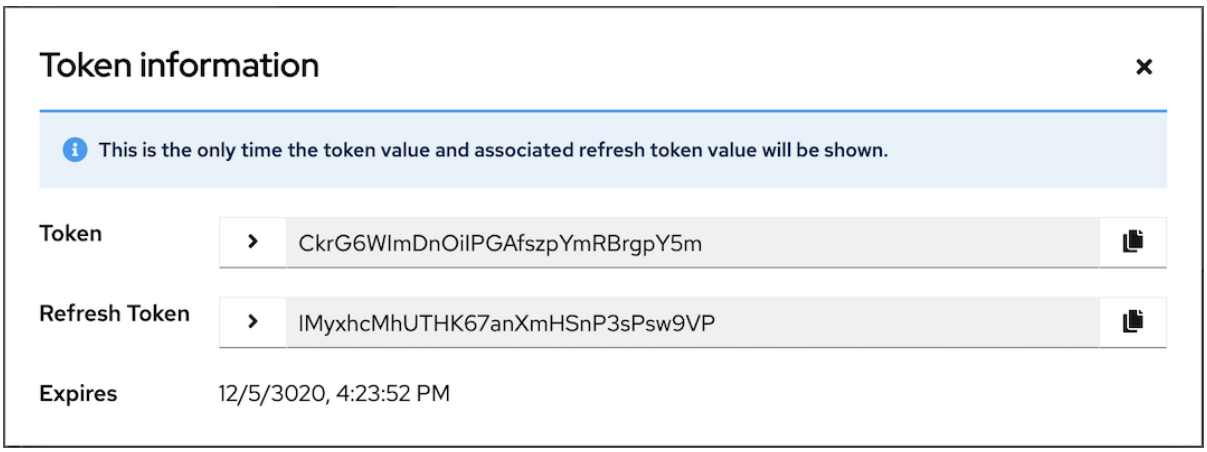
1. **Users** リストビューからユーザーを選択して、OAuth 2 トークンを設定します。
2. ユーザーのプロファイルから **Tokens** タブをクリックします。
3. **Add** をクリックして **Create Token** ウィンドウを開きます。
4. 以下の情報を入力します。
   - **アプリケーション**: トークンを関連付けるアプリケーションの名前を入力します。

または、 アイコンをクリックしてアプリケーション名を検索することもできます。これにより、別のウィンドウが開き、利用可能なオプションから選択できるようになります。リストが膨大な場合は、検索バーを使用して名前でフィルタリングします。

どのアプリケーションにもリンクされていない PAT を作成する場合は、このフィールドを空白のままにしておきます。

- オプション: **説明**: トークンに関する簡単な説明を入力します。
- **スコープ**: 対象のトークンに付与するアクセスレベルを指定します。

5. 変更を破棄するには、**Save** または **Cancel** をクリックします。
6. トークンが保存されると、ユーザー用に新しく作成されたトークンが表示されます。



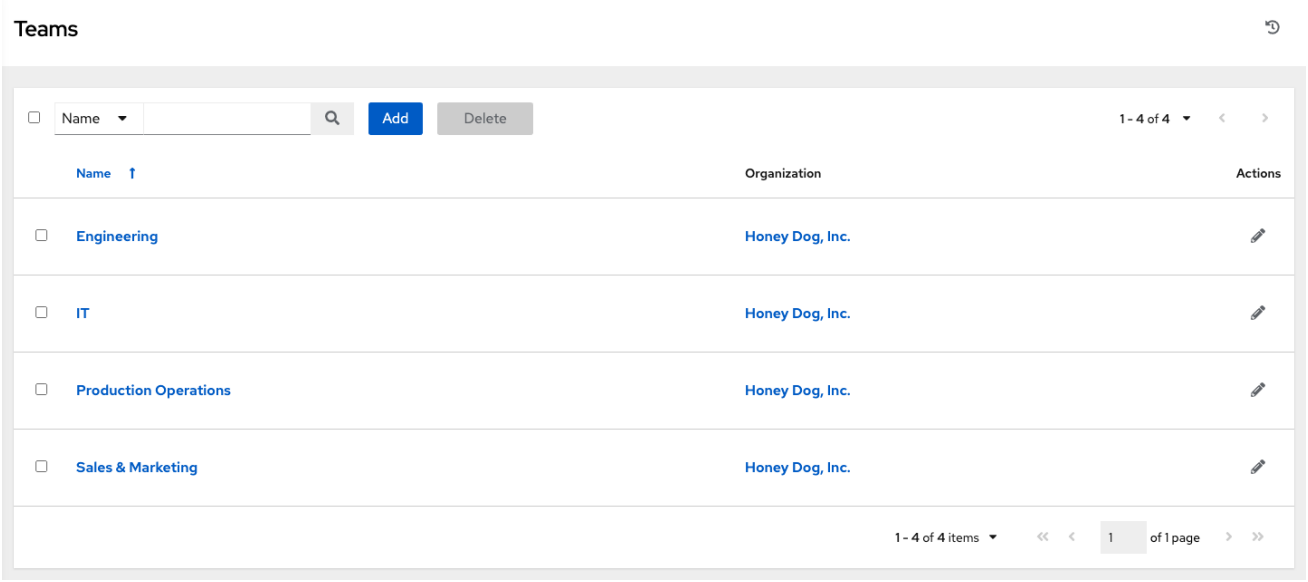
重要

この時だけ唯一、トークンの値と、関連する更新トークンの値が表示されます。

第9章 チームの管理

チーム は、ユーザー、プロジェクト、認証情報、パーミッションが関連付けられた組織の下位部門です。チームは、ロールベースのアクセス制御スキームを実装し、組織全体で責任を委譲する手段を提供します。たとえば、チーム内の各ユーザーではなく、チーム全体に権限を付与できます。

ナビゲーションパネルから **Access** → **Teams** を選択します。



チームのリストを、**Name** や **Organization** で並べ替えおよび検索できます。

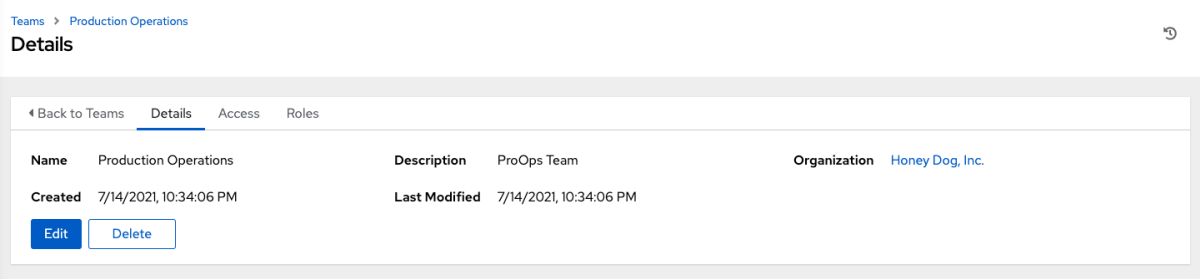
エントリーの横にある Edit (アイコン) を使用して、チームに関する情報を編集します。このチームに関連付けられている **ユーザー** と **権限** を確認することもできます。

9.1. チームの作成

組織に必要な数のユーザーチームを作成できます。ユーザーと同様に、各チームに権限を割り当てることができます。チームには認証情報の所有権を割り当てることもできるため、同じ認証情報を同じユーザーに割り当てる手順が最小限に抑えられます。

手順

1. **Teams** ページで、**Add** をクリックします。
2. 以下のフィールドに該当する詳細を入力します。
   - **Name**
   - オプション: **Description**
   - **Organization**: 既存の組織を選択する必要があります。
3. **Save** をクリックします。**Details** ダイアログが開きます。
4. チームの情報を確認して編集します。



9.1.1. チームへのユーザーの追加または削除

ユーザーをチームに追加するには、ユーザーがすでに作成されている必要があります。詳細は ユーザーの作成 を参照してください。ユーザーは、チームに対してメンバーとしてのみ追加されます。**Access** タブを使用して、さまざまなリソースに対するユーザーのロールを指定します。

手順

1. **Details** ページの **Access** タブで、**Add** をクリックします。
2. プロンプトに従ってユーザーを追加し、ロールに割り当てます。
3. **Save** をクリックします。

9.1.2. ユーザーのロール削除

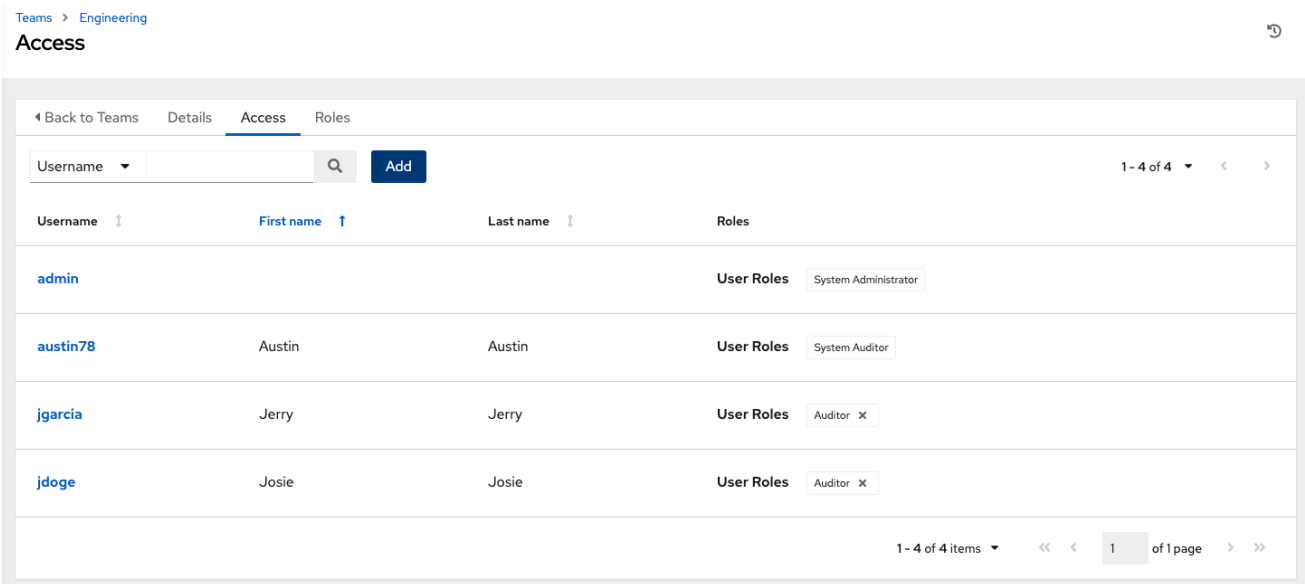
手順

- 特定のユーザーのロールを削除するには、リソースの横にある ✖ アイコンをクリックします。

これにより確認ダイアログが起動し、関連付けの解除を確定するように求められます。

9.1.3. チームのアクセス

Access タブには、特定のチームのメンバーであるユーザーのリストが表示されます。



このリストは、**Username**、**First Name**、または **Last Name** で検索できます。詳細は、ユーザー を参照してください。

9.1.4. チームのロールと権限

Details → Roles を選択すると、このチームで現在使用可能な権限のリストが表示されます。

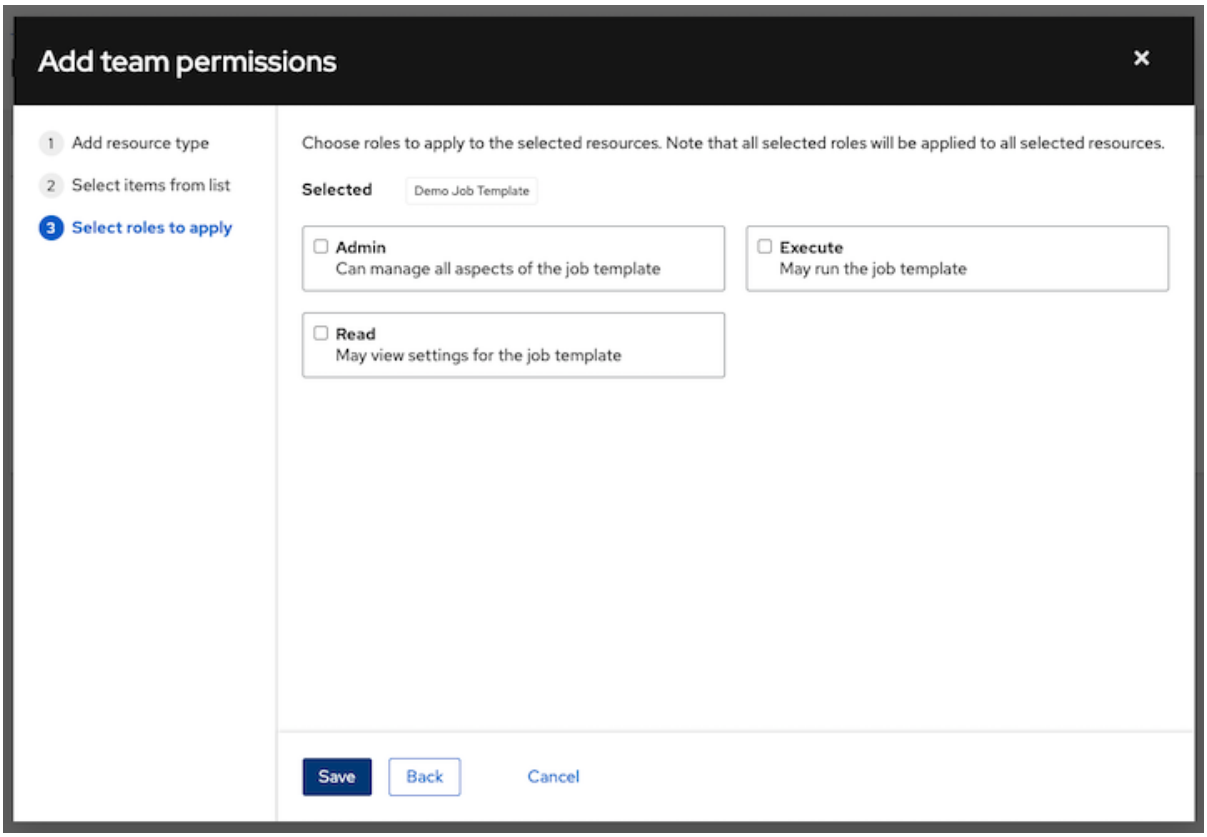
9.1.5. チームパーミッションの追加と削除

デフォルトでは、作成したすべてのチームに読み取り権限が与えられます。これに加えて、プロジェクト、インベントリー、その他の要素の編集や管理などの権限を割り当てることができます。

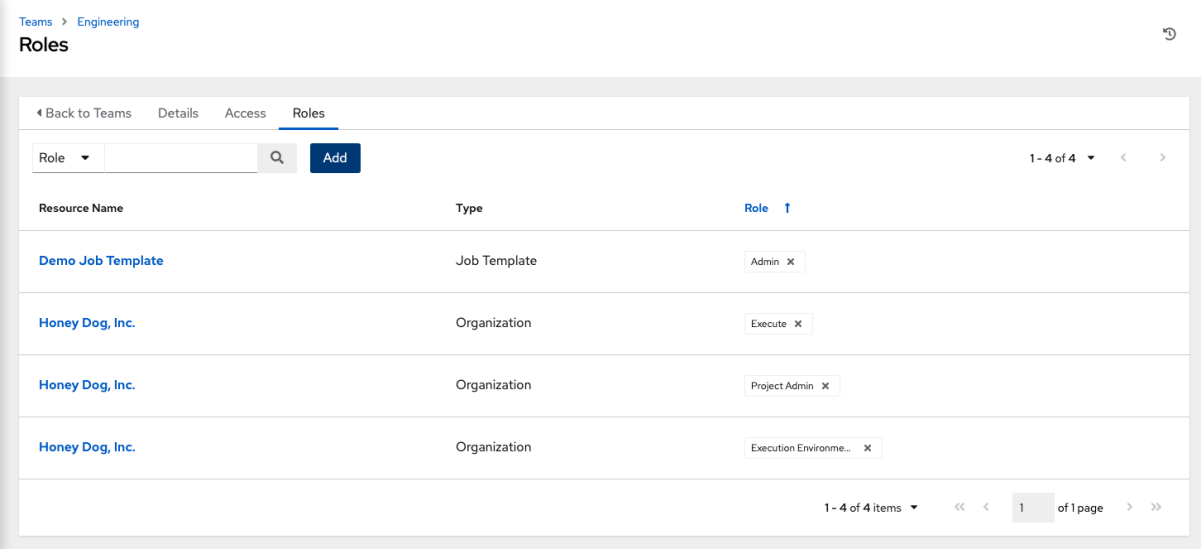
権限は、インベントリー、プロジェクト、ジョブテンプレート、または Organizations ビューから設定できます。

手順

1. **Team** リストビューから、必要なユーザーをクリックします。
2. **Details** ページで、**Add** をクリックします。**Add team permissions** ウィザードが開きます。
3. チームがアクセスする必要のあるオブジェクトを選択します。
4. **Next** をクリックします。
5. チームロールを割り当てるリソースを選択します。
6. **Next** をクリックします。
7. ロールの横にあるチェックボックスをクリックすると、選択したタイプのリソースにそのロールが割り当てられます。リソースが異なれば、利用可能なオプションも異なります。



8. **Save** をクリックします。
9. 選択したリソースごとに割り当てられたロールと、そのユーザーの更新されたプロファイルが表示されます。



9.1.5.1. チームパーミッションの削除

- 特定のリソースの権限を削除するには、リソースの横にある ✖ アイコンをクリックします。これにより確認ダイアログが起動し、関連付けの解除を確定するように求められます。

注記

チーム、個人、または多数のユーザーを追加し、オブジェクトレベルで権限を割り当てることもできます。これには、プロジェクト、インベントリー、ジョブテンプレート、ワークフローテンプレートが含まれます。この機能により、組織で多数のユーザーを一括でオンボーディングする時間を短縮できます。

第10章 ユーザー認証情報の管理

認証情報を使用して、マシンに対するジョブの起動、インベントリーソースの同期、バージョン管理システムからのプロジェクトコンテンツのインポートを行う場合に、Automation Controller のユーザーを認証します。

認証情報をユーザーに公開せずに、ユーザーとチームにこれらの認証情報を使用する権限を付与できます。ユーザーが別のチームに異動する場合や、組織から離職する場合は、対象の認証情報が Automation Controller で利用できるからといって、システムずべてのキー登録をやり直す必要はありません。

注記

Automation controller はデータベース内のパスワードと鍵情報を暗号化し、API を通じて機密情報が公開されることはありません。詳細は、**Automation Controller 管理ガイド** を参照してください。

10.1. 認証情報の仕組み

Automation controller は SSH を使用してリモートホストに接続します。鍵を Automation Controller から SSH に渡すには、鍵を復号化してから名前付きパイプに書き込む必要があります。Automation Controller はそのパイプを使用して鍵を SSH に送信します。そのため、鍵がディスクに書き込まれることはありません。パスワードが使用されている場合、Automation Controller はパスワードプロンプトに直接応答し、パスワードを復号化してからプロンプトに書き込むことでパスワードを処理します。

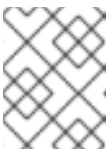
10.2. 新しい認証情報の作成

チームに追加された認証情報は、チームのすべてのメンバーが利用できるようになります。個々のユーザーに認証情報を追加することもできます。

初期設定の一環として、デモ認証情報と Ansible Galaxy の 2 つの認証情報を使用できます。Ansible Galaxy 認証情報は、テンプレートとして使用してください。この認証情報はコピーできますが、編集することはできません。必要に応じて認証情報を追加してください。

手順

1. ナビゲーションパネルから **Resources → Credentials** を選択します。
2. **Add** をクリックします。
3. 以下の情報を入力します。
   - 新しい認証情報の名前。
   - オプション: 新しい認証情報の説明。
   - オプション: 認証情報を関連付ける組織の名前。



注記

ある組織に関連付けられている一連の権限が含まれる認証情報は、別の組織に再割り当てされた後も残ります。

4. **Credential Type** フィールドで、作成する認証情報のタイプを入力または選択します。

5. 認証情報タイプ の説明に従い、選択した認証情報タイプに応じて適切な詳細を入力します。
6. **Save** をクリックします。

10.3. 既存の認証情報に新しいユーザーとジョブテンプレートを追加する

手順

1. ナビゲーションパネルから **Resources → Credentials** を選択します。
2. 追加のユーザーに割り当てる認証情報を選択します。
3. **Access** タブをクリックします。この認証情報に関連付けられているユーザーとチーム、およびそのロールを確認できます。
4. ユーザーを選択し、**Add** をクリックします。ユーザーが存在しない場合は、**Users** メニューから追加します。詳細は、ユーザー を参照してください。
5. **Job Templates** を選択すると、この認証情報に関連付けられたジョブテンプレートと、この認証情報を使用して最近実行されたジョブが表示されます。
6. ジョブテンプレートを選択し、**Add** をクリックして、追加のジョブテンプレートに認証情報を割り当てます。新しいジョブテンプレートの作成の詳細は、ジョブテンプレート セクションを参照してください。

10.4. 認証情報タイプ

Automation controller は次の種類の認証情報をサポートしています。

- Amazon Web Services
- Ansible Galaxy/Automation Hub API Token
- Centrify Vault Credential Provider Lookup
- Container Registry
- CyberArk Central Credential Provider Lookup
- CyberArk Conjur Secrets Manager Lookup
- GitHub Personal Access Token
- GitLab Personal Access Token
- Google Compute Engine
- GPG Public Key
- HashiCorp Vault Secret Lookup
- HashiCorp Vault Signed SSH
- Insights
- Machine

- Microsoft Azure Key Vault
- Microsoft Azure Resource Manager
- Network
- OpenShift or Kubernetes API Bearer Token
- OpenStack
- Red Hat Ansible Automation Platform
- Red Hat Satellite 6
- Red Hat Virtualization
- Source Control
- Thycotic DevOps Secrets Vault
- Thycotic Secret Server
- Vault
- VMware vCenter

Centrify、CyberArk、HashiCorp Vault、Microsoft Azure Key Vault、および Thycotic に関連付けられている認証情報タイプは、認証情報プラグイン機能の一部であり、外部システムによるシークレット情報の検索を可能にするものです。

詳細は、シークレット管理システム を参照してください。

10.4.1. Amazon Web Services 認証情報タイプ

クラウドインベントリーと Amazon Web Services の同期を有効にするには、この認証情報を選択します。

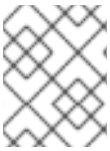
Automation controller は、AWS 認証情報に次の環境変数を使用します。

```
AWS_ACCESS_KEY_ID
AWS_SECRET_ACCESS_KEY
AWS_SECURITY_TOKEN
```

これらは、ユーザーインターフェイスでプロンプトが表示されるフィールドです。

Amazon Web Services の認証情報は、AWS の **アクセスキー** と **シークレットキー** で構成されます。

Automation Controller は、アイデンティティーおよびアクセス管理 (IAM) STS 認証情報とも呼ばれる EC2 STS トークンのサポートを提供します。**Security Token Service** (STS) は、AWS IAM ユーザーのために、権限が限られた一時的な認証情報を要求できる Web サービスです。



注記

EC2 のタグの値にブール値 (**yes**/**no**/**true**/**false**) が含まれている場合は、その値を引用符で囲む必要があります。

警告

暗黙的な IAM ロール認証情報を使用するには、IAM ロールに依存して AWS API にアクセスするときに、Automation controller で AWS クラウド認証情報をアタッチしないでください。

AWS クラウド認証情報をジョブテンプレートにアタッチすると、IAM ロールの認証情報ではなく、AWS 認証情報が強制的に使用されます。

関連情報

IAM/EC2 STS トークンの詳細は、IAM の一時的なセキュリティー認証情報 を参照してください。

10.4.1.1. Ansible Playbook で Amazon EC2 の認証情報にアクセスする

ジョブランタイム環境から AWS 認証情報パラメーターを取得できます。

```
vars:
  aws:
    access_key: '{{ lookup("env", "AWS_ACCESS_KEY_ID") }}'
    secret_key: '{{ lookup("env", "AWS_SECRET_ACCESS_KEY") }}'
    security_token: '{{ lookup("env", "AWS_SECURITY_TOKEN") }}'
```

10.4.2. Ansible Galaxy/Automation Hub API トークン認証情報タイプ

Ansible Galaxy にアクセスするか、Private Automation Hub のインスタンスで公開されたコレクションを使用するには、この認証情報を選択します。

この画面で Galaxy サーバーの URL を入力します。

Red Hat Hybrid Cloud Console の **Galaxy Server URL** フィールドに **Server URL** フィールドの内容を入力します。Red Hat Hybrid Cloud Console で、**Auth Server URL** フィールドに **SSO URL** フィールドの内容を入力します。

関連情報

詳細は、Automation Hub でのコレクションの使用 を参照してください。

10.4.3. Centrify Vault Credential Provider Lookup 認証情報タイプ

これはシークレット管理機能の一部とみなされます。詳細は、Centrify Vault Credential Provider Lookup を参照してください。

10.4.4. Container Registry 認証情報タイプ

Automation controller がコンテナーイメージのコレクションにアクセスできるようにするには、この認証情報を選択します。詳細は、What is a container registry? を参照してください。

名前を指定する必要があります。**Authentication URL** フィールドにはデフォルト値が事前に入力されています。値を変更するには、別のコンテナーレジストリーの認証エンドポイントを指定します。

10.4.5. CyberArk Central Credential Provider Lookup 認証情報タイプ

これはシークレット管理機能の一部とみなされます。

詳細は、CyberArk Central Credential Provider (CCP) Lookup を参照してください。

10.4.6. CyberArk Conjur Secrets Manager Lookup 認証情報タイプ

これはシークレット管理機能の一部とみなされます。

詳細は、CyberArk Conjur Secrets Manager Lookup を参照してください。

10.4.7. GitHub Personal Access Token 認証情報タイプ

この認証情報を選択すると、GitHub から取得できる **Personal Access Token** (PAT) を使用して GitHub にアクセスできるようになります。

詳細は、Webhook の使用 を参照してください。

GitHub PAT の認証情報では、**Token** フィールドに値が必要です。これは GitHub プロファイル設定で提供されます。

この認証情報は、Webhook リスナージョブで使用する GitHub への API 接続を確立し、ステータスの更新を送信 (Post) するために使用できます。

10.4.8. GitLab Personal Access Token 認証情報タイプ

この認証情報を選択すると、GitLab から取得できる **Personal Access Token** (PAT) を使用して GitLab にアクセスできるようになります。

詳細は、Webhook の使用 を参照してください。

GitLab PAT の認証情報では、**Token** フィールドに値が必要です。これは GitLab プロファイル設定で提供されています。

この認証情報は、Webhook リスナージョブで使用する GitLab への API 接続を確立し、ステータスの更新を送信 (Post) するために使用できます。

10.4.9. Google Compute Engine 認証情報タイプ

この認証情報を選択すると、クラウドインベントリーと Google Compute Engine (GCE) との同期が可能になります。

Automation controller は、GCE 認証情報に次の環境変数を使用します。

```
GCE_EMAIL
GCE_PROJECT
GCE_CREDENTIALS_FILE_PATH
```

ユーザーインターフェイスでプロンプトが表示されるフィールドは次のとおりです。

GCE 認証情報には次の情報が必要です。

- **サービスアカウントのメールアドレス**: Google Compute Engine **サービスアカウント** に割り当てられるメールアドレス。

- オプション: **Project**: GCE によって割り当てられた ID またはプロジェクト作成時に指定した一意のプロジェクト ID を指定します。
- オプション: **Service Account JSON File** GCE サービスアカウントファイルをアップロードします。**Browse** をクリックして、GCE インスタンスで実行されているサービスやアプリケーションが他の Google Cloud Platform API とやり取りするために使用できる特別なアカウント情報を含むファイルを参照します。これにより、サービスアカウントと仮想マシンインスタンスにパーミッションが付与されます。
- **RSA 秘密鍵**: サービスアカウントメールに関連付けられる PEM ファイル。

10.4.9.1. Ansible Playbook で Google Compute Engine の認証情報にアクセスする

GCE 認証情報パラメーターはジョブランタイム環境から取得できます。

```
vars:
  gce:
    email: '{{ lookup("env", "GCE_EMAIL") }}'
    project: '{{ lookup("env", "GCE_PROJECT") }}'
    pem_file_path: '{{ lookup("env", "GCE_PEM_FILE_PATH") }}'
```

10.4.10. GPG Public Key 認証情報タイプ

この認証情報タイプを選択すると、ソースコントロールから同期するときに Automation Controller がプロジェクトの整合性を検証できるようになります。

有効なキーペアを生成する方法、CLI ツールを使用してコンテンツに署名する方法、およびコントローラーに公開鍵を追加する方法の詳細は、プロジェクトの署名と検証 を参照してください。

10.4.11. HashiCorp Vault Secret Lookup 認証情報タイプ

これはシークレット管理機能の一部とみなされます。

詳細は、HashiCorp Vault Secret Lookup を参照してください。

10.4.12. HashiCorp Vault Signed SSH 認証情報タイプ

これはシークレット管理機能の一部とみなされます。

詳細は、HashiCorp Vault Signed SSH を参照してください。

10.4.13. Insights 認証情報タイプ

Red Hat Insights とクラウドインベントリーの同期を有効にするには、この認証情報タイプを選択します。

Insights 認証情報は Insights の **ユーザー名** と **パスワード** で、これはユーザーの Red Hat カスタマーポータルアカウントのユーザー名とパスワードです。

Insights の **extra_vars** および **env** インジェクターは次のとおりです。

```
ManagedCredentialType(
    namespace='insights',
....
```

```
....
....

injectors={
        'extra_vars': {
            "scm_username": "{{username}}",
            "scm_password": "{{password}}",
        },
        'env': {
            'INSIGHTS_USER': '{{username}}',
            'INSIGHTS_PASSWORD': '{{password}}',
        },
```

10.4.14. Machine 認証情報タイプ

Machine 認証情報を使用すると、Automation Controller は管理下のホスト上で Ansible を呼び出すことができます。SSH ユーザー名を指定して、必要に応じてパスワード、SSH 鍵、キーパスワードを指定したり、デプロイ時に Automation Controller がユーザーにパスワードの入力を求めるようにしたりできます。これらは、Playbook に対する SSH およびユーザーレベルの権限昇格アクセスを定義し、リモートホストで Playbook を実行するジョブを送信するときに使用されます。

ネットワーク接続 **httpapi**、**netconf**、**network_cli** では、認証情報タイプとして **Machine** を使用します。

Machine/SSH 認証情報は、環境変数を使用しません。これらの認証情報は、ansible **-u** フラグを介してユーザー名を渡し、基盤となる SSH クライアントが SSH パスワードを要求したときにパスワードを対話的に書き込みます。

Machine 認証情報には次の入力が必要です。

- **Username**: SSH 認証に使用するユーザー名。
- **Password**: SSH 認証に使用するパスワード。このパスワードは、入力されると暗号化されてデータベースに保存されます。あるいは、**Prompt on launch** を選択して、起動時にユーザーにパスワードを要求するように Automation Controller を設定することもできます。このような場合、ジョブの起動時にダイアログが開き、ユーザーにパスワードとパスワードの確認の入力を求めます。
- **SSH Private Key**: マシン認証情報の SSH 秘密鍵をコピーまたはドラッグアンドドロップします。
- **Private Key Passphrase**: 使用する SSH 秘密鍵がパスワードで保護されている場合は、秘密鍵のキーパスフレーズを設定できます。このパスワードは、入力されると暗号化されてデータベースに保存されます。**Prompt on launch** を選択して、起動時にユーザーにキーパスフレーズの入力を求めるように Automation Controller を設定することもできます。選択した場合、ジョブの起動時にダイアログが開き、ユーザーにキーパスフレーズの入力と確認を求めます。
- **Privilege Escalation Method** 特定のユーザーに割り当てる昇格権限のタイプを指定します。これは、**--become-method=BECOME_METHOD** パラメーターを指定するのと同じです。**BECOME_METHOD** は、既存の方法のいずれか、または作成したカスタムの方法です。方法の名前の入力を開始すると、適切な名前が自動入力されます。
- **選択なし**: タスクまたはプレイの **become** が **yes** に設定されており、何も選択されていない場合は、デフォルトで **sudo** に設定されます。
- **sudo**: スーパーユーザー (root ユーザー) 権限で単一のコマンドを実行します。

- **su**: スーパーユーザー (root ユーザー) アカウント (または他のユーザーアカウント) に切り替えます。
- **pbrun**: 制御されたアカウントでアプリケーションまたはコマンドが実行されるように要求し、詳細レベルの root 権限の委譲およびキーのロギングを可能にします。
- **pfexec**: 特定のユーザー ID やグループ ID など、定義済みのプロセス属性を使用してコマンドを実行します。
- **dzdo**: Centrify の Active Directory サービスの RBAC 情報を使用する sudo の拡張バージョン。詳細は、DZDO のサイト を参照してください。
- **pmrun**: 制御されたアカウントでアプリケーションが実行されるように要求します。Privilege Manager for Unix 6.0 を参照してください。
- **runas**: 現在のユーザーとして実行できるようにします。
- **enable**: ネットワークデバイスで昇格された権限に切り替えます。
- **doas**: リモート/ログインユーザーが **doas** (Do as user) ユーティリティーを通じて別のユーザーとしてコマンドを実行できるようにします。
- **ksu**: リモート/ログインユーザーが Kerberos アクセスで別のユーザーとしてコマンドを実行できるようにします。
- **machinectl**: **systemd** マシンマネージャーを使用してコンテナーを管理できるようにします
- **sesu**: リモート/ログインユーザーが CA Privileged Access Manager を使用して別のユーザーとしてコマンドを実行できるようにします。

注記

カスタムの **become** プラグインは Ansible 2.8 以降から利用可能です。詳細は、 権限昇格について と become プラグイン のリストを参照してください。

- **Privilege Escalation Username**: 権限昇格のオプションを選択した場合にのみ、このフィールドが表示されます。リモートシステム上で昇格権限で使用するユーザー名を入力します。
- **Privilege Escalation Password**: 権限昇格のオプションを選択した場合にのみ、このフィールドが表示されます。選択した権限昇格タイプによりリモートシステムでユーザーを認証するために使用するパスワードを入力します。このパスワードはデータベースに暗号化されて保存されます。**Prompt on launch** を選択して、起動時にユーザーにパスワードを要求するように Automation Controller を設定することもできます。このような場合、ジョブの起動時にダイアログが開き、ユーザーにパスワードとパスワードの確認の入力を求めます。

注記

sudo パスワードは SSH パスワードまたは SSH 秘密鍵と組み合わせて使用する必要があります。Automation Controller は、**sudo** を呼び出して sudo ユーザーに変更する前に、まずホストとの認証された SSH 接続を確立する必要があるためです。

警告

スケジュール済みジョブで使用される認証情報は、**Prompt on launch** として設定しないでください。

10.4.14.1. Ansible Playbook でマシンの認証情報にアクセスする

ユーザー名とパスワードは Ansible fact から取得できます。

```
vars:
  machine:
    username: '{{ ansible_user }}'
    password: '{{ ansible_password }}'
```

10.4.15. Microsoft Azure Key Vault 認証情報タイプ

これはシークレット管理機能の一部とみなされます。

詳細は、Microsoft Azure Key Vault を参照してください。

10.4.16. Microsoft Azure Resource Manager 認証情報タイプ

Microsoft Azure Resource Manager とクラウドインベントリーの同期を有効にするには、この認証情報タイプを選択します。

Microsoft Azure Resource Manager の認証情報には次を入力する必要があります。

- **Subscription ID**: Microsoft Azure アカウントのサブスクリプション UUID。
- **Username**: Microsoft Azure アカウントに接続するために使用するユーザー名。
- **Password**: Microsoft Azure アカウントに接続するために使用するパスワード。
- **Client ID**: Microsoft Azure アカウントのクライアント ID。
- **Client Secret**: Microsoft Azure アカウントのクライアントシークレット。
- **Tenant ID**: Microsoft Azure アカウントのテナント ID。
- **Azure Cloud Environment** Azure クラウドまたは Azure スタック環境に関連付けられた変数。

これらのフィールドは、API の変数に相当します。

サービスプリンシパルの認証情報を渡すには、以下の変数を定義します。

```
AZURE_CLIENT_ID
AZURE_SECRET
AZURE_SUBSCRIPTION_ID
AZURE_TENANT
AZURE_CLOUD_ENVIRONMENT
```

Active Directory のユーザー名およびパスワードのペアを渡すには、以下の変数を定義します。

```
AZURE_AD_USER
AZURE_PASSWORD
AZURE_SUBSCRIPTION_ID
```

認証情報をパラメーターとして Playbook 内のタスクに渡すこともできます。優先順位は、パラメーター、次に環境変数、最後にホームディレクトリーにあるファイルになります。

認証情報をパラメーターとしてタスクに渡すには、サービスプリンシパルの認証情報についての以下のパラメーターを使用します。

```
client_id
secret
subscription_id
tenant
azure_cloud_environment
```

あるいは、Active Directory のユーザー名/パスワードに次のパラメーターを渡します。

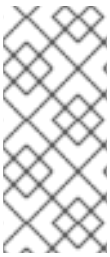
```
ad_user
password
subscription_id
```

10.4.16.1. Ansible Playbook で Microsoft Azure リソースマネージャーの認証情報にアクセスする

Microsoft Azure 認証情報パラメーターはジョブランタイム環境から取得できます。

```
vars:
  azure:
    client_id: '{{ lookup("env", "AZURE_CLIENT_ID") }}'
    secret: '{{ lookup("env", "AZURE_SECRET") }}'
    tenant: '{{ lookup("env", "AZURE_TENANT") }}'
    subscription_id: '{{ lookup("env", "AZURE_SUBSCRIPTION_ID") }}'
```

10.4.17. Network 認証情報タイプ



注記

プロバイダー との **ローカル** 接続で Ansible ネットワークモジュールを使用してネットワークデバイスに接続および管理する場合は、Network 認証情報タイプを選択します。

ネットワークデバイスに接続する場合、認証情報タイプが接続の種類と一致する必要があります。

- **provider** を使用した **local** 接続の場合は、認証情報タイプは **Network** に指定する必要があります。
- それ以外のネットワーク接続 (**httpapi**、**netconf**、および **network_cli**) については、認証情報タイプは **Machine** に指定する必要があります。

ネットワークデバイスで使用できる接続タイプの詳細は、複数の通信プロトコル を参照してください。

Automation controller は、ネットワーク認証情報に次の環境変数を使用します。

```
ANSIBLE_NET_USERNAME
ANSIBLE_NET_PASSWORD
```

ネットワーク認証情報として次の情報を提供します。

- **Username**: ネットワークデバイスと組み合わせて使用するユーザー名。
- **Password**: ネットワークデバイスと組み合わせて使用するパスワード。
- **SSH Private Key**: ユーザーを SSH 経由でネットワークに対して認証するために使用される実際の SSH 秘密鍵をコピーするか、またはドラッグアンドドロップします。
- **Private Key Passphrase**: SSH 経由でネットワークに対してユーザーを認証するための秘密鍵のパスフレーズ。
- **Authorize**: Options フィールドからこれを選択して、特権モードに入るかどうかを制御します。
- **Authorize** チェックボックスをオンにした場合は、**Authorize Password** フィールドに、特権モードにアクセスするためのパスワードを入力ます。

詳細は、新しい接続プラグインを使用した Ansible Network Playbook の移植 を参照してください。

10.4.18. Ansible Playbook でネットワークの認証情報にアクセスする

ユーザー名とパスワードのパラメーターはジョブランタイム環境から取得できます。

```
vars:
  network:
    username: '{{ lookup("env", "ANSIBLE_NET_USERNAME") }}'
    password: '{{ lookup("env", "ANSIBLE_NET_PASSWORD") }}'
```

10.4.19. OpenShift or Kubernetes API Bearer Token 認証情報タイプ

Kubernetes または OpenShift コンテナーを参照するインスタンスグループを作成するには、この認証情報タイプを選択します。

詳細は、**Automation controller 管理ガイド** の コンテナーとインスタンスグループ を参照してください。

コンテナーの認証情報として次の情報を提供します。

- **OpenShift or Kubernetes API Endpoint**(必須): OpenShift または Kubernetes コンテナーへの接続に使用されるエンドポイントです。
- **API Authentication Bearer Token**(必須): 接続の認証に使用されるトークンです。
- オプション: **Verify SSL**: このオプションをオンにすると、サーバーの SSL/TLS 証明書が有効で信頼できるかどうかを検証できます。内部またはプライベートの **認証局** (CA) を使用する環境で検証を無効にするには、このオプションをオフのままにする必要があります。
- **Certificate Authority Data**: 指定されている場合、証明書を貼り付けるときに **BEGIN CERTIFICATE** と **END CERTIFICATE** の行を含めます。

コンテナーグループは、OpenShift クラスターへの接続を可能にする認証情報が関連付けられているインスタンスグループの一種です。コンテナーグループを設定するには、次の項目が必要です。

- 最初に入ることができる namespace。すべてのクラスターにデフォルトの 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 を起動および管理できるロールを持つサービスアカウント。
- 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その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、コンテナーグループの Pod 仕様でそれらのロールを指定できます。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。
- そのサービスアカウントに関連付けられたトークン (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

10.4.19.1. Openshift クラスターでのサービスアカウントの作成

Automation Controller を介してコンテナーグループ内のジョブを実行するために使用するサービスアカウントを、Openshift または Kubernetes クラスターに作成します。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。

サービスアカウントを作成したら、新しいサービスアカウントの情報を使用して Automation Controller を設定します。

手順

1. サービスアカウントを作成するために、サンプルサービスアカウント ダウンロードして使用します。必要に応じて、上記の認証情報を取得するために変更します。
2. サンプルサービスアカウント 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。

10.4.20. OpenStack 認証情報タイプ

クラウドインベントリーと OpenStack の同期を有効にするには、この認証情報タイプを選択します。

OpenStack 認証情報として次の情報を提供します。

- **ユーザー名**: OpenStack への接続に使用するユーザー名。
- **パスワード (API キー)**: OpenStack に接続するために使用するパスワードまたは API キー
- **ホスト (認証 URL)**: 認証に使用するホスト。
- **プロジェクト (テナント名)**: OpenStack に使用されるテナント名またはテナント ID。この値は通常、ユーザー名と同じです。
- オプション: **プロジェクト (ドメイン名)**: ドメインに関連付けられたプロジェクト名を入力します。
- オプション: **ドメイン名**: OpenStack への接続に使用する FQDN を指定します。

OpenStack クラウド認証情報を使用する場合は、クラウドインベントリーでのクラウド認証情報の使用を参照してください。これにはサンプル Playbook が含まれています。

10.4.21. Red Hat Ansible Automation Platform 認証情報タイプ

別の Automation controller インスタンスにアクセスするには、この認証情報を選択します。

Ansible Automation Platform の認証情報には次の入力が必要です。

- **Red Hat Ansible Automation Platform**: 接続先の他のインスタンスのベース URL または IP アドレス。
- **Username**: 接続に使用するユーザー名。
- **Password**: 接続に使用するパスワード。
- **Oauth Token**: ユーザー名とパスワードが使用されない場合は、認証に使用する OAuth トークンを指定します。

Ansible Automation Platform の **env** インジェクターは次のとおりです。

```
ManagedCredentialType(
   namespace='controller',

....
....
....

injectors={
       'env': {
          'TOWER_HOST': '{{host}}',
          'TOWER_USERNAME': '{{username}}',
          'TOWER_PASSWORD': '{{password}}',
```

```
        'TOWER_VERIFY_SSL': '{{verify_ssl}}',
        'TOWER_OAUTH_TOKEN': '{{oauth_token}}',
        'CONTROLLER_HOST': '{{host}}',
        'CONTROLLER_USERNAME': '{{username}}',
        'CONTROLLER_PASSWORD': '{{password}}',
        'CONTROLLER_VERIFY_SSL': '{{verify_ssl}}',
        'CONTROLLER_OAUTH_TOKEN': '{{oauth_token}}',
    }
```

10.4.21.1. Ansible Playbook で Automation Controller の認証情報にアクセスする

ホスト、ユーザー名、パスワードのパラメーターはジョブランタイム環境から取得できます。

```
vars:
  controller:
    host: '{{ lookup("env", "CONTROLLER_HOST") }}'
    username: '{{ lookup("env", "CONTROLLER_USERNAME") }}'
    password: '{{ lookup("env", "CONTROLLER_PASSWORD") }}'
```

10.4.22. Red Hat Satellite 6 認証情報タイプ

Red Hat Satellite 6 とクラウドインベントリーの同期を有効にするには、この認証情報タイプを選択します。

Automation controller は、ユーザーインターフェイスで要求されたフィールドをもとに Satellite 設定ファイルを書き込みます。ファイルへの絶対パスは、次の環境変数に設定されます。

```
FOREMAN_INI_PATH
```

Satellite 認証情報には次の入力が必要になります。

- **Satellite 6 URL**: 接続先の Satellite 6 URL または IP アドレス。
- **ユーザー名**: Satellite 6 への接続に使用するユーザー名。
- **パスワード**: Satellite 6 への接続に使用するパスワード。

10.4.23. Red Hat Virtualization 認証情報タイプ

この認証情報を選択すると、Automation Controller が **Red Hat Virtualization** によって管理される Ansible の **oVirt4.py** 動的インベントリープラグインにアクセスできるようになります。

Automation controller は、Red Hat Virtualization 認証情報に次の環境変数を使用します。ユーザーインターフェイスのフィールドは次のとおりです。

```
OVIRT_URL
OVIRT_USERNAME
OVIRT_PASSWORD
```

Red Hat Virtualization 認証情報として次の情報を提供します。

- **ホスト (認証 URL)**: 接続するホスト URL または IP アドレス。インベントリーと同期するには、認証情報 URL に **ovirt-engine/api** パスが含まれている必要があります。

- **ユーザー名**: oVirt4 への接続に使用するユーザー名。正常に実行させるには、ドメインプロファイル (例: **username@ovirt.host.com)** が含まれている必要があります。
- **Password**: 接続に使用するパスワード。
- オプション: **CA File**: oVirt 証明書ファイルへの絶対パスを指定します (拡張子は **.pem**、**.cer**、**.crt** である可能性がありますが、一貫性を保つために **.pem** を推奨します)。

10.4.23.1. Ansible Playbook で Virtualization の認証情報にアクセスする

ジョブランタイム環境から Red Hat Virtualization 認証情報パラメーターを取得できます。

```
vars:
  ovirt:
    ovirt_url: '{{ lookup("env", "OVIRT_URL") }}'
    ovirt_username: '{{ lookup("env", "OVIRT_USERNAME") }}'
    ovirt_password: '{{ lookup("env", "OVIRT_PASSWORD") }}'
```

Red Hat Virtualization の **file** および **env** インジェクターは次のとおりです。

```
ManagedCredentialType(
    namespace='rhv',

....
....
....

injectors={
        # The duplication here is intentional; the ovirt4 inventory plugin
        # writes a .ini file for authentication, while the ansible modules for
        # ovirt4 use a separate authentication process that support
        # environment variables; by injecting both, we support both
        'file': {
            'template': '\n'.join(
                [
                    '[ovirt]',
                    'ovirt_url={{host}}',
                    'ovirt_username={{username}}',
                    'ovirt_password={{password}}',
                    '{% if ca_file %}ovirt_ca_file={{ca_file}}{% endif %}',
                ]
            )
        },
        'env': {'OVIRT_INI_PATH': '{{tower.filename}}', 'OVIRT_URL': '{{host}}', 'OVIRT_USERNAME':
'{{username}}', 'OVIRT_PASSWORD': '{{password}}'},
    },
)
```

10.4.24. Source Control 認証情報タイプ

Source Control 認証情報は、Git や Subversion などのリモートリビジョン管理システムからローカルソースコードリポジトリーを複製および更新するためにプロジェクトで使用されます。

ソースコントロールの認証情報には次の入力が必要です。

- **ユーザー名**: ソースコントロールシステムと併用するユーザー名。
- **パスワード**: ソースコントロールシステムと併用するパスワード。
- **SCM 秘密鍵**: ユーザーを SSH 経由でソースコントロールシステムに対して認証するために使用される実際の SSH 秘密鍵をコピーするか、またはドラッグアンドドロップします。
- **秘密鍵のパスフレーズ**: 使用される SSH 秘密鍵がパスフレーズで保護される場合、秘密鍵のパスフレーズを設定できます。

注記

Source Control 認証情報を **Prompt on launch** として設定することはできません。

Source Control 認証情報に GitHub アカウントを使用しており、アカウントで **2FA (2 要素認証)** が有効になっている場合は、パスワードフィールドで、アカウントのパスワードではなく Personal Access Token を使用する必要があります。

10.4.25. Thycotic DevOps Secrets Vault 認証情報タイプ

これはシークレット管理機能の一部とみなされます。

詳細は、Thycotic DevOps Secrets Vault を参照してください。

10.4.26. Thycotic Secret Server 認証情報タイプ

これはシークレット管理機能の一部とみなされます。

詳細は、Thycotic Secret Server を参照してください。

10.4.27. Ansible Vault 認証情報タイプ

Ansible Vault とのインベントリーの同期を有効にするには、この認証情報タイプを選択します。

Vault 認証情報では、**Vault パスワード** と、オプションで複数の Vault 認証情報が適用される場合には **Vault 識別子** が必要です。

Multi-Vault サポートの詳細は、**Automation controller 管理ガイド** の Multi-Vault 認証情報 セクションを参照してください。

あるいは、**Prompt on launch** を選択して、起動時にユーザーにパスワードを要求するように Automation Controller を設定することもできます。

Prompt on Launch を選択すると、ジョブの起動時にダイアログが開き、ユーザーにパスワードの入力を求めます。

警告

スケジュール済みジョブで使用される認証情報は、**Prompt on launch** として設定しないでください。

Ansible Vault の詳細は、Ansible Vault による機密データの保護 を参照してください。

10.4.28. VMware vCenter 認証情報タイプ

VMware vCenter とのインベントリーの同期を有効にするには、この認証情報タイプを選択します。

Automation controller は、VMware vCenter 認証情報に次の環境変数を使用します。

```
VMWARE_HOST
VMWARE_USER
VMWARE_PASSWORD
VMWARE_VALIDATE_CERTS
```

これらは、ユーザーインターフェイスでプロンプトが表示されるフィールドです。

VMware 認証情報には次の入力が必要です。

- **vCenter ホスト**: 接続先の vCenter ホスト名または IP アドレス。
- **ユーザー名**: vCenter への接続に使用するユーザー名。
- **パスワード**: vCenter への接続に使用するパスワード。

注記

VMware ゲストツールがインスタンスで実行されていない場合、VMware インベントリーの同期により、そのインスタンスの IP アドレスが返されません。

10.4.28.1. Ansible Playbook で VMware vCenter の認証情報にアクセスする

VMware vCenter 認証情報パラメーターはジョブランタイム環境から取得できます。

```
vars:
  vmware:
    host: '{{ lookup("env", "VMWARE_HOST") }}'
    username: '{{ lookup("env", "VMWARE_USER") }}'
    password: '{{ lookup("env", "VMWARE_PASSWORD") }}'
```

10.5. PLAYBOOK での AUTOMATION CONTROLLER 認証情報の使用

次の Playbook は、Playbook で Automation controller 認証情報を使用する方法の例です。

```
- hosts: all

  vars:
    machine:
      username: '{{ ansible_user }}'
      password: '{{ ansible_password }}'
    controller:
      host: '{{ lookup("env", "CONTROLLER_HOST") }}'
      username: '{{ lookup("env", "CONTROLLER_USERNAME") }}'
      password: '{{ lookup("env", "CONTROLLER_PASSWORD") }}'
    network:
      username: '{{ lookup("env", "ANSIBLE_NET_USERNAME") }}'
```

```
    password: '{{ lookup("env", "ANSIBLE_NET_PASSWORD") }}'
  aws:
    access_key: '{{ lookup("env", "AWS_ACCESS_KEY_ID") }}'
    secret_key: '{{ lookup("env", "AWS_SECRET_ACCESS_KEY") }}'
    security_token: '{{ lookup("env", "AWS_SECURITY_TOKEN") }}'
  vmware:
    host: '{{ lookup("env", "VMWARE_HOST") }}'
    username: '{{ lookup("env", "VMWARE_USER") }}'
    password: '{{ lookup("env", "VMWARE_PASSWORD") }}'
  gce:
    email: '{{ lookup("env", "GCE_EMAIL") }}'
    project: '{{ lookup("env", "GCE_PROJECT") }}'
  azure:
    client_id: '{{ lookup("env", "AZURE_CLIENT_ID") }}'
    secret: '{{ lookup("env", "AZURE_SECRET") }}'
    tenant: '{{ lookup("env", "AZURE_TENANT") }}'
    subscription_id: '{{ lookup("env", "AZURE_SUBSCRIPTION_ID") }}'

tasks:
  - debug:
      var: machine

  - debug:
      var: controller

  - debug:
      var: network

  - debug:
      var: aws

  - debug:
      var: vmware

  - debug:
      var: gce

  - shell: 'cat {{ gce.pem_file_path }}'
    delegate_to: localhost

  - debug:
      var: azure
```

delegate_to と任意の検索変数の使用

```
- command: somecommand
  environment:
    USERNAME: '{{ lookup("env", "USERNAME") }}'
    PASSWORD: '{{ lookup("env", "PASSWORD") }}'
  delegate_to: somehost
```

第11章 カスタム認証情報タイプ

システム管理者は、YAML または JSON のような定義を使用して、標準形式でカスタム認証情報タイプを定義できます。既存の認証情報タイプと同様に機能するカスタム認証情報タイプを定義できます。たとえば、カスタム認証情報タイプを使用すると、Playbook やカスタムインベントリースクリプトで使用できるように、サードパーティーの Web サービスの API トークンを環境変数に挿入できます。

カスタム認証情報は、認証情報を挿入する以下の方法をサポートします。

- 環境変数
- Ansible の追加変数
- ファイルベースのテンプレート (認証情報の値が含まれる **.ini** または **.conf** ファイルの生成)

1 つの SSH 認証情報と複数のクラウド認証情報をジョブテンプレートにアタッチできます。クラウド認証情報ごとに、違う種類を指定する必要があります。各タイプの認証情報は 1 つだけ許可されます。Vault の認証情報とマシンの認証情報は別個のエンティティーです。

注記

- 新しい認証情報タイプを作成するときは、**extra_vars**、**env**、およびファイルの namespace で競合しないようにする必要があります。
- 環境変数または追加変数の名前は予約されているため、**ANSIBLE_** で始めることはできません。
- 認証情報タイプ (**CredentialType**) の作成および編集と **CredentialType.injection** フィールドの表示をできるようにするには、システム管理者 (スーパーユーザー) パーミッションが必要です。

11.1. コレクションからのコンテンツ収集

"マネージド型" の認証情報タイプである **kind=galaxy** は、プロジェクトの更新を実行するときに **requirements.yml** で定義されたコレクションを取得するためのコンテンツソースを表します。コンテンツソースの例としては、galaxy.ansible.com、console.redhat.com、オンプレミス Automation Hub などがあります。この新しい認証情報タイプは、Ansible ドキュメントの Configuring the ansible-galaxy client で説明されているように、プロジェクトの更新で **ansible-galaxy collection install** 実行時に環境変数を構築するために必要な URL と (オプションの) 認証の詳細を表します。この認証情報タイプには、Ansible Galaxy CLI に公開される設定オプション (サーバーごとなど) に直接マップするフィールドがあります。

API のエンドポイントは、組織レベルでのこれらの認証情報の順序付きリストを反映します。

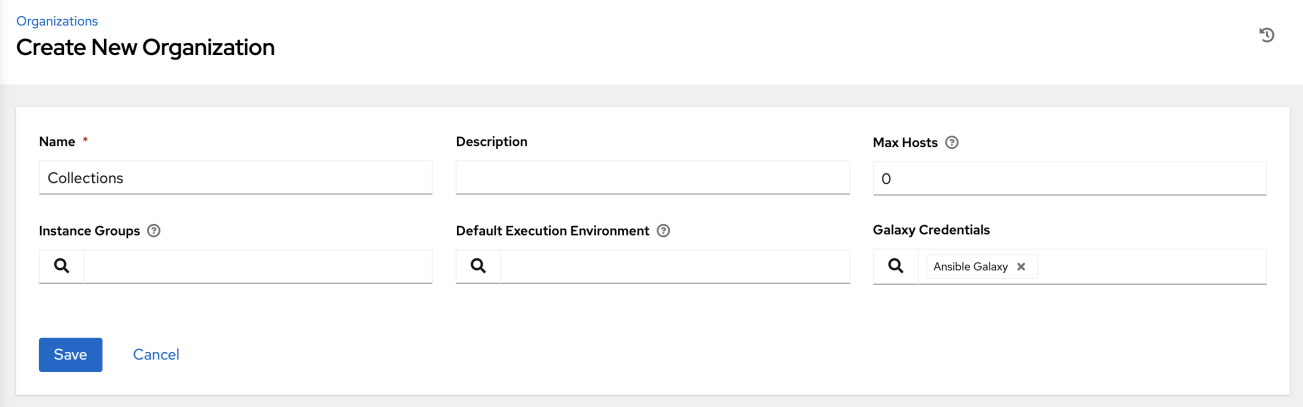
```
/api/v2/organizations/N/galaxy_credentials/
```

Automation controller のインストールで既存の Galaxy 指向の設定値が移行されると、アップグレード後の適切な認証情報が作成され、すべての組織にアタッチされます。最新バージョンにアップグレードすると、アップグレード前に存在していたすべての組織に、関連付けられた 1 つ以上の Galaxy 認証情報のリストが作成されます。

さらに、アップグレード後は、これらの設定は **/api/v2/settings/jobs/** エンドポイントから表示 (または編集) できなくなります。

Automation Controller は、**galaxy.ansible.com** が組織のリストの最初の認証情報でない場合でも、パブリック Galaxy から直接ロールを取得し続けます。グローバルな Galaxy 設定は、ジョブレベルではなく、ユーザーインターフェイスの組織レベルで設定されます。

組織の **Add** および **Edit** ウィンドウに、**kind=galaxy** の認証情報用のオプションの **Credential** 検索フィールドがあります。



順序によりコンテンツの同期と検索の優先順位が設定されるため、これらの認証情報の順序を指定することが重要です。詳細は、組織の作成 を参照してください。

コレクションを使用してプロジェクトを設定する方法の詳細は、Automation Hub でのコレクションの使用 を参照してください。

11.2. 後方互換 API の留意事項

バージョン 2 の API (**api/v2/**) がサポートされている場合、ジョブテンプレートと認証情報の一対多の関係を使用できます (マルチクラウドサポートを含む)。

v2 の API では、認証情報をフィルタリングできます。

```
curl "https://controller.example.org/api/v2/credentials/?credential_type__namespace=aws"
```

V2 CredentialType モデルでは、リレーションシップは以下のように定義されます。

マシン	SSH
Vault	Vault
Network	環境変数を設定します (**ANSIBLE_NET_AUTHORIZE** など)。
SCM	ソースコントロール
クラウド	EC2、AWS
クラウド	その他多数
Insights	Insights
Galaxy	galaxy.ansible.com、console.redhat.com

マシン	SSH
Galaxy	オンプレミス Automation Hub

11.3. 内容検証

Automation controller は GNU Privacy Guard (GPG) を使用してコンテンツを検証します。

詳細は、GNU プライバシーハンドブック を参照してください。

11.4. 認証情報タイプの使用開始

ナビゲーションパネルから、**Administration → Credential Types** を選択します。カスタム認証情報タイプが作成されていない場合は、**Credential Types** を追加するように求められます。

認証情報タイプが作成されていると、このページには、既存および利用可能な認証情報タイプのリストが表示されます。

認証情報タイプに関する詳細情報を表示するには、認証情報の名前または編集アイコン をクリックします。

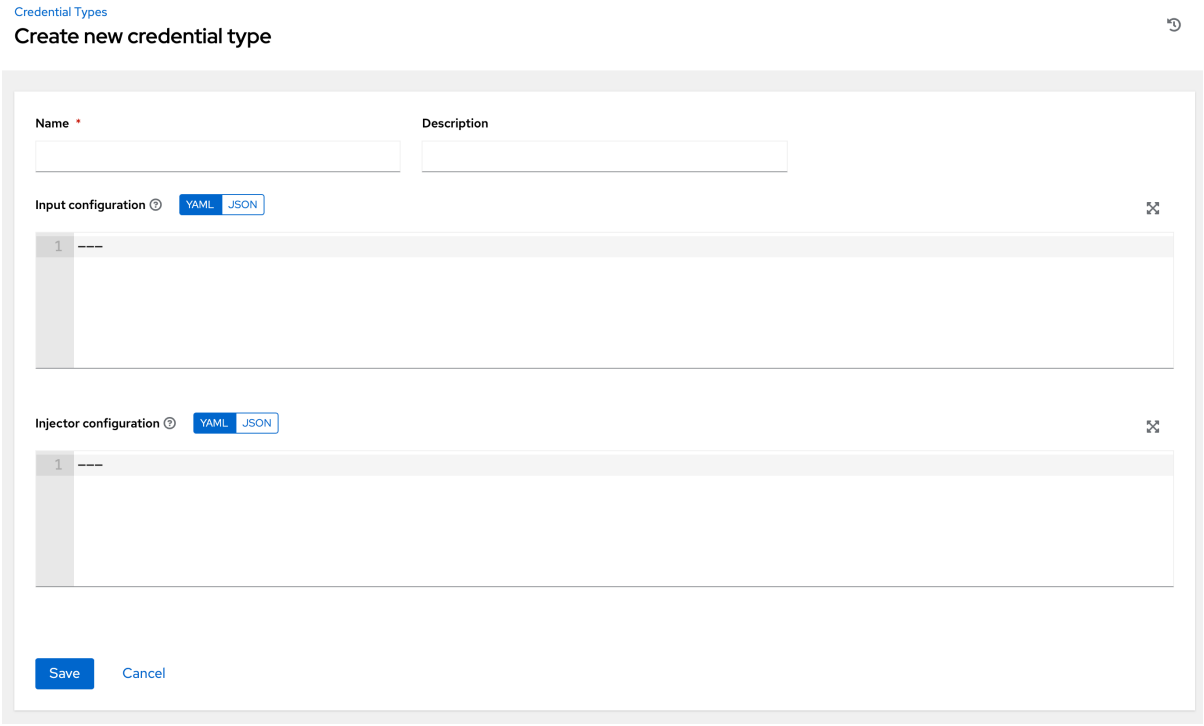
各認証情報タイプでは、該当する場合、**Input Configuration** フィールドと **Injector Configuration** フィールドに独自の固有の設定が表示されます。YAML 形式と JSON 形式の両方が設定フィールドでサポートされています。

11.5. 新しい認証情報タイプの作成

新規の認証情報タイプを作成するには、以下を実行します。

手順

1. **Credential Types** ビューで、**Add** をクリックします。



2. **Name** および **Description** フィールドに詳細を入力します。

注記

カスタムの認証タイプには無効であるため、認証タイプを新規作成する際には、**INPUT** 名、**INJECTOR** 名および ID に、**ANSIBLE_** で始まる予約済変数名を使用しないでください。

3. **Input Configuration** フィールドで、対象のタイプの順序付きフィールドのセットを定義する入力スキーマを指定します。形式は YAML または JSON です。

YAML

```
fields:
  - type: string
    id: username
    label: Username
  - type: string
    id: password
    label: Password
    secret: true
required:
  - username
  - password
```

YAML ページ でさらに YAML の例をご覧ください。

JSON

```
{
"fields": [
  {
  "type": "string",
  "id": "username",
```

```
  "label": "Username"
  },
  {
  "secret": true,
  "type": "string",
  "id": "password",
  "label": "Password"
   }
  ],
 "required": ["username", "password"]
}
```

その他の JSON の例については、JSON Web サイト をご覧ください。

JSON 形式の次の設定は、各フィールドとその使用方法を示しています。

```
{
  "fields": [{
    "id": "api_token",    # required - a unique name used to reference the field value

    "label": "API Token", # required - a unique label for the field

    "help_text": "User-facing short text describing the field.",

    "type": ("string" | "boolean")   # defaults to 'string'

    "choices": ["A", "B", "C"]   # (only applicable to `type=string`)

    "format": "ssh_private_key"  # optional, can be used to enforce data format validity
                                 for SSH private key data (only applicable to `type=string`)

    "secret": true,       # if true, the field value will be encrypted

    "multiline": false    # if true, the field should be rendered as multi-line for input entry
                          # (only applicable to `type=string`)
},{
    # field 2...
},{
    # field 3...
}],

"required": ["api_token"]   # optional; one or more fields can be marked as required
},
```

type=string の場合、フィールドではオプションで複数の選択オプションから指定できます。

```
{
  "fields": [{
      "id": "api_token",    # required - a unique name used to reference the field value
      "label": "API Token", # required - a unique label for the field
      "type": "string",
      "choices": ["A", "B", "C"]
  }]
},
```

4. **インジェクター設定** フィールドに、認証情報タイプが挿入できる値を指定する環境変数または追加変数を入力します。形式は YAML または JSON に指定できます (前の手順の例を参照)。JSON 形式の次の設定は、各フィールドとその使用方法を示しています。

```
{
  "file": {
      "template": "[mycloud]\ntoken={{ api_token }}"
  },
  "env": {
      "THIRD_PARTY_CLOUD_API_TOKEN": "{{ api_token }}"
  },
  "extra_vars": {
      "some_extra_var": "{{ username }}:{{ password }}"
  }
}
```

認証情報タイプは、**.ini** ファイルまたは証明書/キーデータをサポートする一時ファイルも生成します。

```
{
  "file": {
      "template": "[mycloud]\ntoken={{ api_token }}"
  },
  "env": {
      "MY_CLOUD_INI_FILE": "{{ tower.filename }}"
  }
}
```

この例では、Automation Controller が次の内容を含む一時ファイルを書き込みます。

```
[mycloud]\ntoken=SOME_TOKEN_VALUE
```

生成されるファイルへの絶対ファイルパスは、**MY_CLOUD_INI_FILE** という名前の環境変数に保存されます。

以下は、カスタム認証情報テンプレートで多数のファイルを参照する例です。

Inputs

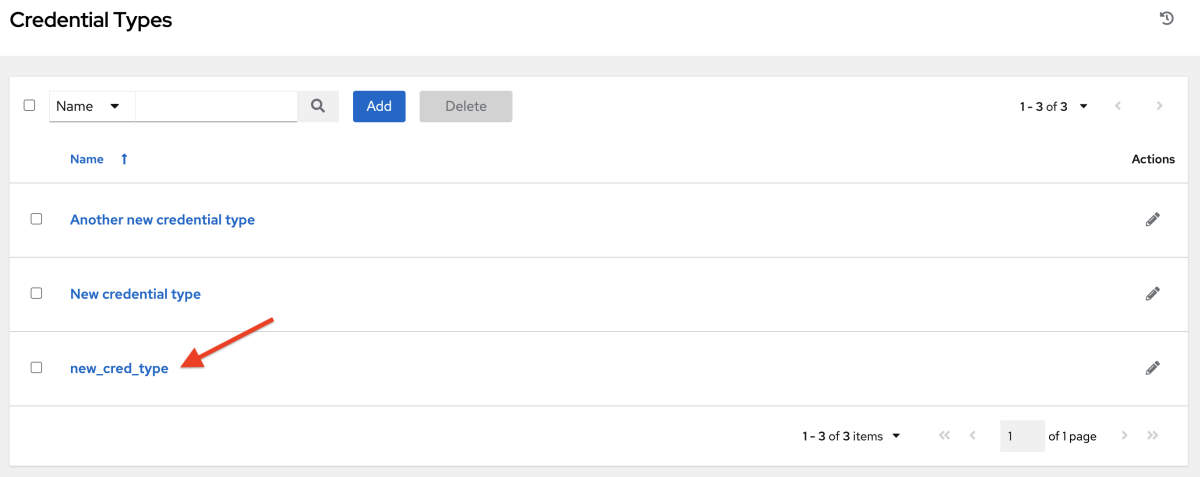
```
{
  "fields": [{
    "id": "cert",
    "label": "Certificate",
    "type": "string"
  },{
    "id": "key",
    "label": "Key",
    "type": "string"
  }]
}
```

Injectors

```
{
```

```
  "file": {
    "template.cert_file": "[mycert]\n{{ cert }}",
    "template.key_file": "[mykey]\n{{ key }}"
},
"env": {
    "MY_CERT_INI_FILE": "{{ tower.filename.cert_file }}",
    "MY_KEY_INI_FILE": "{{ tower.filename.key_file }}"
}
}
```

5. **Save** をクリックします。
新しく作成した認証情報タイプが、認証情報タイプのリストに表示されます。



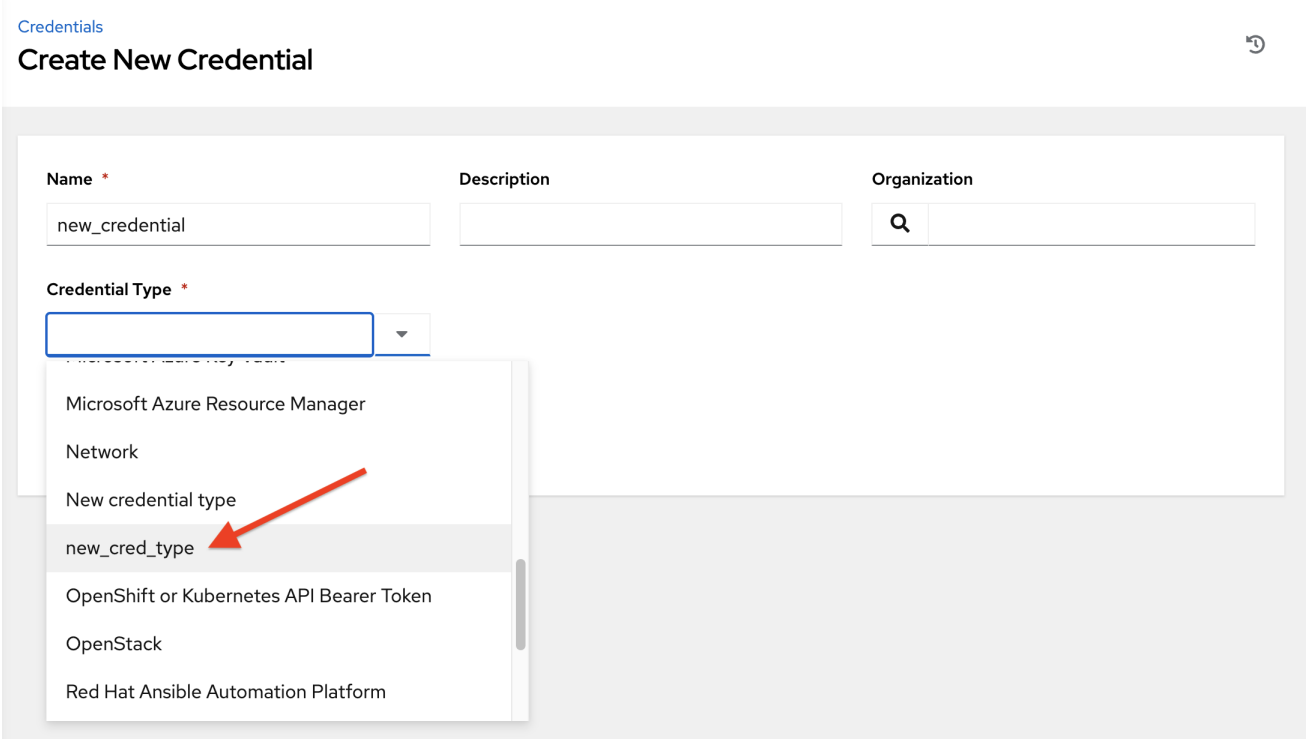
6. 編集アイコン をクリックして、認証情報タイプのオプションを変更します。

注記

Edit 画面では、詳細の変更や、認証情報の削除ができます。**Delete** オプションが無効になっている場合は、その認証情報タイプが認証情報によって使用されています。認証情報タイプを削除する前に、それを使用しているすべての認証情報からその認証情報タイプを削除する必要があります。

検証

- 新規の認証情報の作成時に、新規に作成された認証情報タイプを **Credential Type** 選択ウィンドウから選択できることを確認します。



関連情報

新しい認証情報を作成する方法については、認証情報の作成 を参照してください。

第12章 シークレット管理システム

ユーザーとシステム管理者は、マシンとクラウドの認証情報をアップロードして、代わりに自動化でマシンや外部サービスにアクセスできるようにします。デフォルトでは、SSH パスワード、SSH 秘密鍵、クラウドサービスの API トークンなどの機密認証情報の値は、暗号化された後にデータベースに保存されます。

認証情報プラグインによって裏付けられた外部認証情報を使用すると、認証情報フィールド (パスワードや SSH 秘密鍵など) を、Automation controller に直接提供する代わりに、**シークレット管理システム**に保存されている値にマップできます。

Automation controller は、以下の統合を含むシークレット管理システムを提供します。

- AWS Secrets Manager Lookup
- Centrify Vault Credential Provider Lookup
- **CyberArk Central Credential Provider**Lookup (CCP)
- CyberArk Conjur Secrets Manager Lookup
- HashiCorp Vault **Key-Value** Store (KV)
- HashiCorp Vault SSH Secrets Engine
- Microsoft Azure **Key Management System** (KMS)
- Thycotic DevOps Secrets Vault
- Thycotic Secret Server

これらの外部シークレット値は、それらを必要とする Playbook を実行する前にフェッチされます。

関連情報

ユーザーインターフェイスでシークレット管理システムの認証情報を指定する方法の詳細は、認証情報を参照してください。

12.1. シークレットルックアップの設定とリンク

サードパーティーシステムからシークレットを取得する場合、認証情報フィールドを外部システムにリンクすることになります。認証情報フィールドを外部システムに保存されている値にリンクするには、そのシステムに対応する外部認証情報を選択し、必要な値を検索するための **メタデータ** を指定します。メタデータ入力フィールドは、ソース認証情報の外部認証情報タイプ定義の一部です。

Automation Controller は、開発者、インテグレーター、システム管理者、パワーユーザー向けの認証情報プラグインインターフェイスを備えています。このインターフェイスから、新しい外部認証情報タイプを追加して、他のシークレット管理システムをサポートするように拡張できます。

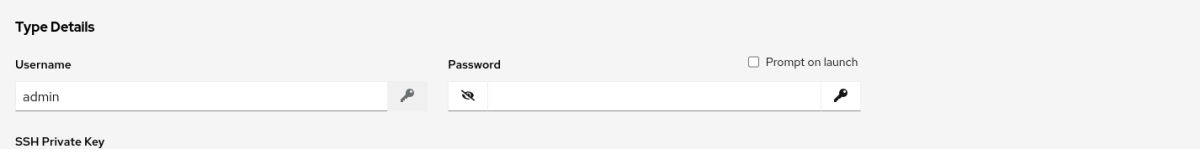
Automation controller で、サポート対象の各サードパーティーのシークレット管理システムを設定して使用するには、次の手順を実行します。

手順

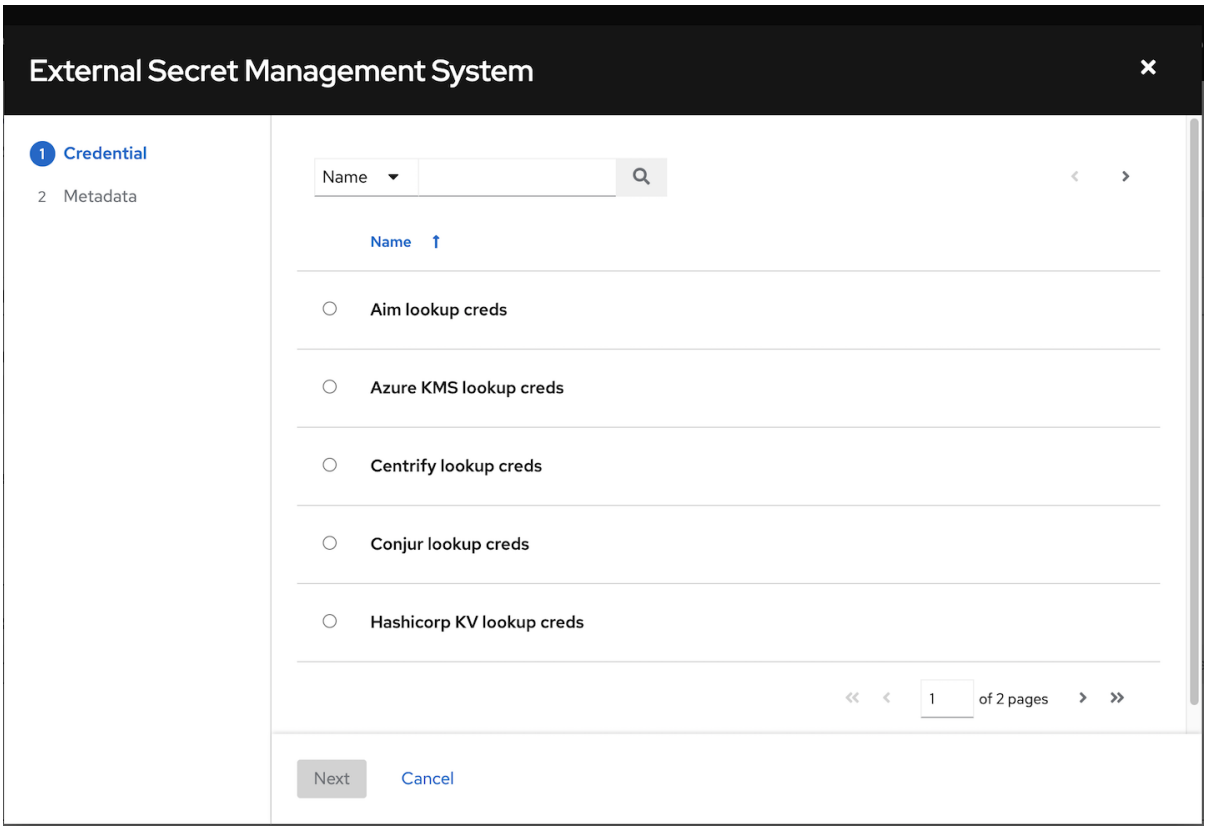
1. シークレット管理システムで認証するための外部認証情報を作成します。少なくとも、外部認証情報の名前を指定し、**Credential Type** フィールドで次のいずれかを選択します。

- AWS Secrets Manager Lookup
- Centrify Vault Credential Provider Lookup
- CyberArk Central Credential Provider (CCP) Lookup
- CyberArk Conjur Secrets Manager Lookup
- HashiCorp Vault Secret Lookup
- HashiCorp Vault Signed SSH
- Microsoft Azure Key Vault
- Thycotic DevOps Secrets Vault
- Thycotic Secret Server
  この例では、**デモ認証情報** がターゲット認証情報です。

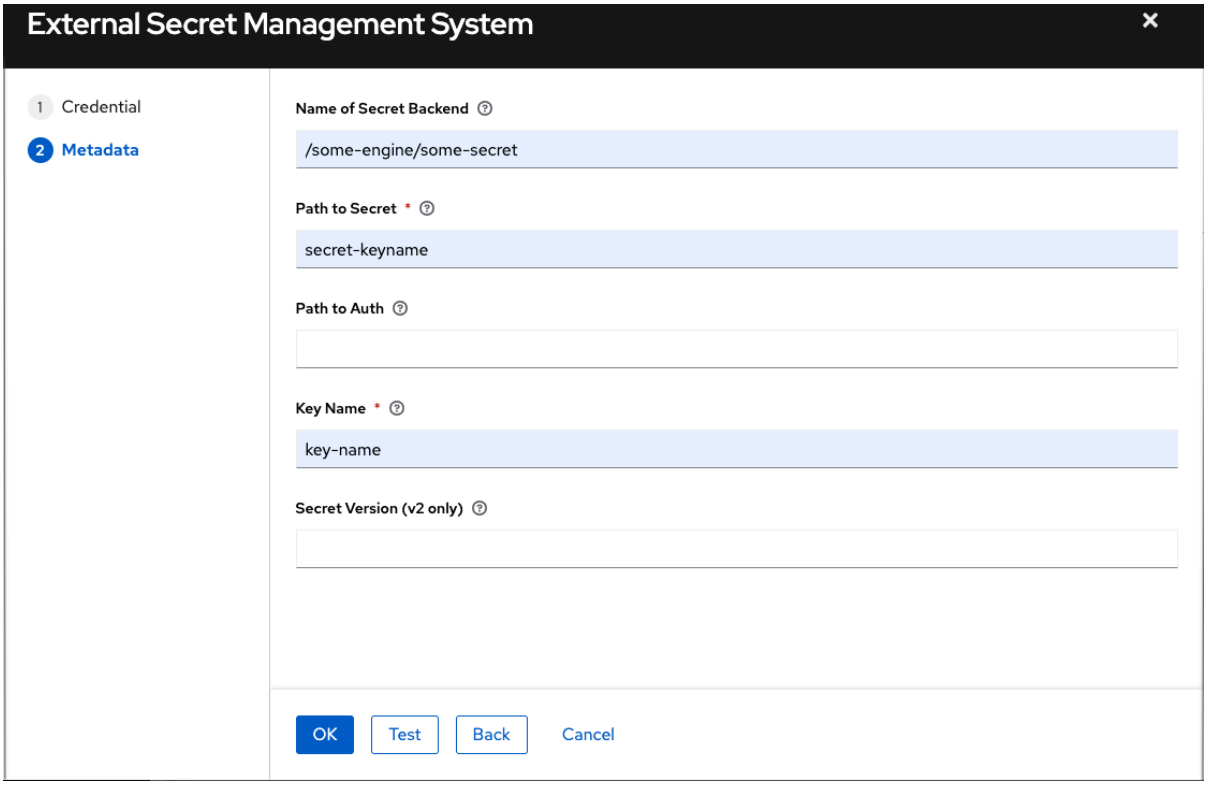
2. 外部認証情報にリンクする **Type Details** エリアに続くフィールドについては、入力フィールドの鍵アイコン をクリックし、1つ以上の入力フィールドを、外部認証情報と、外部システム内のシークレットを検索するためのメタデータにリンクします。



3. シークレット情報を取得するために使用する入力ソースを選択します。



4. リンクする認証情報を選択し、**Next** をクリックします。これにより、入力ソースの **Metadata** タブが表示されます。この例は、HashiVault Secret Lookup のメタデータプロンプトを示しています。メタデータは、選択した入力ソースに固有です。
詳細は、認証情報入力ソースのメタデータ の表を参照してください。



5. **Test** をクリックして、シークレット管理システムへの接続を確認します。ルックアップに失敗した場合は、次のようなエラーメッセージが表示されます。

6. **OK** をクリックします。ターゲット認証情報の **Details** 画面に戻ります。
7. ステップ 3 から始めてこれらのステップを繰り返し、ターゲット認証情報の残りの入力フィールドを完了します。この方法で情報をリンクすることで、Automation controller はユーザー名、パスワード、キー、証明書、トークンなどの機密情報をサードパーティーの管理システムから取得し、ターゲットの認証情報フォームの残りのフィールドにそのデータを入力します。
8. 必要に応じて、機密情報の取得方法としてリンクを使用しないフィールドには、情報を手動で入力します。各フィールドの詳細は、適切な 認証情報のタイプ を参照してください。
9. **Save** をクリックします。

関連情報

詳細は、Credential プラグイン の開発ドキュメントを参照してください。

12.1.1. 認証情報入力ソースのメタデータ

入力ソースの **Metadata** タブに必要な情報です。

AWS Secrets Manager Lookup

メタデータ	説明
AWS Secrets Manager Region (必須)	シークレットマネージャーが配置されているリージョン。
AWS Secret Name (必須)	AWS アクセスキーによって生成された AWS シークレット名を指定します。

Centrify Vault Credential Provider Lookup

メタデータ	説明
アカウント名 (必須)	Centrify Vault に関連付けられたシステムアカウントまたはドメインの名前。
システム名	Centrify ポータルが使用する名前を指定します。

CyberArk Central Credential Provider Lookup

メタデータ	説明
オブジェクトクエリー (必須)	オブジェクトの検索クエリー
オブジェクトクエリーフォーマット (必須)	特定のシークレット名には **Exact** を選択し、動的に生成された名前を持つシークレットには **Regexp** を選択します。
オブジェクトのプロパティー	返すプロパティーの名前を指定します。たとえば、デフォルトの **Content** 以外の **UserName** または **Address** です。
理由	オブジェクトのポリシーごとに必要な場合は、CyberArk は理由のログを記録するので、シークレットをチェックアウトする理由を指定します。

CyberArk Conjur Secrets Lookup

メタデータ	説明
シークレット識別子	シークレットの識別子
シークレットのバージョン	必要に応じてシークレットのバージョンを指定します。しない場合には、空白にしておくと、最新バージョンが使用されます。

HashiVault Secret Lookup

メタデータ	説明

メタデータ	説明
シークレットバックエンドの名前	使用する KV バックエンドの名前を指定します。代わりに **Path to Secret** フィールドの最初のパスセグメントを使用するには、空白のままにします。
シークレットへのパス (必須)	**/path/username** など、機密情報が保存されるパスを指定します。
キー名 (必須)	シークレット情報を検索するキーの名前を指定します。
シークレットのバージョン (V2 のみ)	必要に応じてバージョンを指定します。しない場合には、空白にしておくと、最新バージョンが使用されます。

HashiCorp Signed SSH

メタデータ	説明
未署名の公開鍵 (必須)	署名する証明書の公開鍵を指定します。これは、ターゲットホストの認証キーファイルに存在する必要があります。
シークレットへのパス (必須)	**/path/username** など、機密情報が保存されるパスを指定します。
ロール名 (必須)	ロールは、Hashi vault に保存される SSH 設定とパラメーターのコレクションです。通常、たとえば、異なる特権やタイムアウトを持つロールを指定できます。したがって、たとえば、ルートに対して署名された証明書の取得を許可されたロールと、その他の権限の低いロールを指定できます。
有効なプリンシパル	格納されているキーの証明書を承認するために Vault に要求する、デフォルト以外のユーザー (複数可) を指定します。Hashi Vault には、署名するデフォルトのユーザー (ec2-user など) があります。

Microsoft Azure KMS

メタデータ	説明
シークレット名 (必須)	Microsoft Azure の Key Vault アプリで参照されるシークレットの名前。
シークレットのバージョン	必要に応じてシークレットのバージョンを指定します。しない場合には、空白にしておくと、最新バージョンが使用されます。

Thycotic DevOps Secrets Vault

メタデータ	説明
シークレットパス (必須)	機密情報が保存されるパスを指定します (例:/path/username)。

Thycotic Secret Server

メタデータ	説明
シークレット ID (必須)	シークレットの識別子
シークレットフィールド	シークレットから使用するフィールドを指定します。

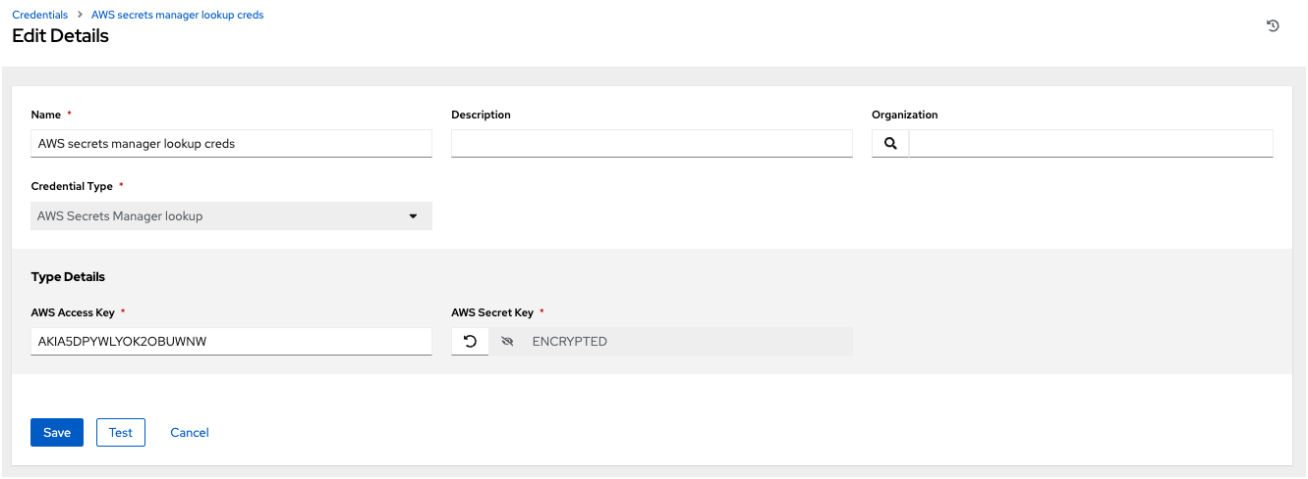
12.1.2. AWS Secrets Manager Lookup

このプラグインにより、Amazon Web Services を認証情報入力ソースとして使用し、Amazon Web Services Secrets Manager からシークレットを取得できるようになります。AWS Secrets Manager は、Microsoft Azure Key Vault と同様のサービスを提供しており、AWS コレクションでそのためのルックアッププラグインが提供されています。

Credential Type に AWS Secrets Manager Lookup を選択した場合は、ルックアップを設定するために次のメタデータを指定します。

- **AWS Access Key** (必須): AWS キー管理システムとの通信に使用するアクセスキーを入力します。
- **AWS Secret Key** (必須): AWS IAM コンソールで取得したシークレットを入力します。

以下は、設定された AWS Secret Manager 認証情報の例です。



12.1.3. Centrify Vault Credential Provider Lookup

この統合が機能するには、Centrify Vault Web サービスを実行してシークレットを保存する必要があります。**Credential Type** に **Centrify Vault Credential Provider Lookup**を選択した場合は、次のメタデータを指定してルックアップを設定します。

- **Centrify Tenant URL** (必須): Centrify のシークレット管理システムとの通信に使用する URL を指定します。
- **Centrify API User** (必須): ユーザー名を指定します。
- **Centrify API Password** (必須): パスワードを指定します。
- **OAuth2 Application ID**: OAuth2 クライアントに関連付けられている特定の識別子を指定します。

- **OAuth2 Scope**: OAuth2 クライアントのスコープを指定します。

12.1.4. CyberArk Central Credential Provider (CCP) Lookup

この統合が機能するようするには、CyberArk Central Credential Provider Web サービスが稼働中で、シークレットを保存する必要があります。**Credential Type** に **CyberArk Central Credential Provider Lookup** を選択した場合は、次のメタデータを指定して検索を設定します。

- **CyberArk CCP URL** (必須): CyberArk CCP のシークレット管理システムとの通信に使用する URL を指定します。http や https などの URL スキームを含める必要があります。
- オプション: **Web Service ID**: Web サービスの識別子を指定します。これを空白のままにすると、デフォルトで AIMWebService になります。
- **Application ID** (必須): CyberArk AIM サービスで提供される識別子を指定します。
- **Client Key**: クライアントキーを貼り付けます (CyberArk から提供されている場合)。
- **Client Certificate**: 証明書を貼り付けるときに **BEGIN CERTIFICATE** 行と **END CERTIFICATE** 行を含めます (CyberArk から提供されている場合)。
- **Verify SSL Certificates**: このオプションは、URL が HTTPS を使用している場合にのみ使用できます。サーバーの SSL/TLS 証明書が有効で信頼できるかどうかを検証するには、このオプションをオンにします。内部 CA またはプライベート CA を使用する環境では、このオプションをオフのままにして検証を無効にしてください。

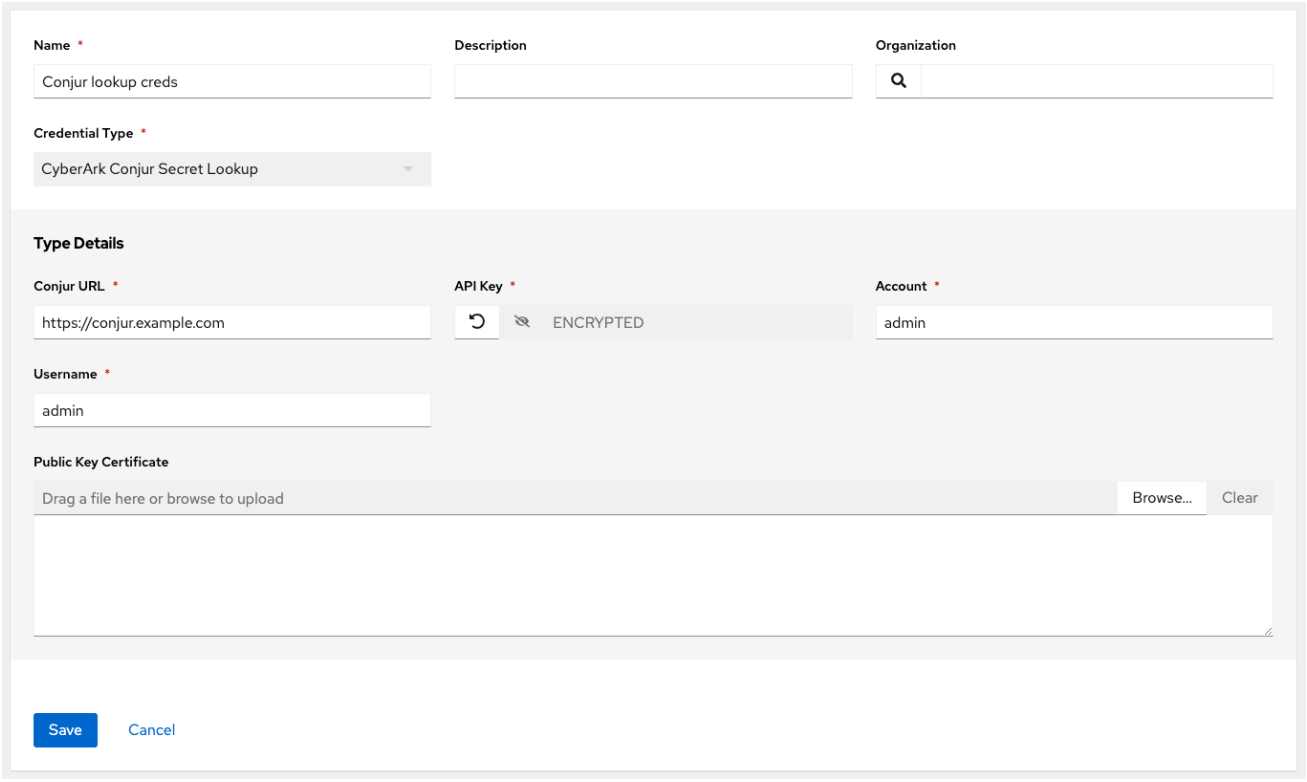
12.1.5. CyberArk Conjur Secrets Manager Lookup

Conjur Cloud テナントをターゲットとして使用できる場合、CyberArk Conjur Secrets Lookup 外部管理システム認証情報プラグインを設定します。

Credential Type に **CyberArk Conjur Secrets Manager Lookup**を選択した場合は、次のメタデータを指定してルックアップを設定します。

- **Conjur URL** (必須): CyberArk Conjur のシークレット管理システムとの通信に使用する URL を指定します。これには、http や https などの URL スキームが含まれている必要があります。
- **API キー** (必須): Conjur の管理者から受け取ったキーを指定します。
- **アカウント** (必須): 組織のアカウント名
- **ユーザー名** (必須): このサービス用に認証するユーザー
- **公開鍵鍵証明書**: CyberArk から提供されている場合、公開鍵を貼り付けるときに **BEGIN CERTIFICATE** および **END CERTIFICATE** の行を含めます。

以下は、設定された CyberArk Conjur 認証情報の例です。



12.1.6. HashiCorp Vault Secret Lookup

Credential Type に **HashiCorp Vault Secret Lookup**を選択した場合は、次のメタデータを指定してルックアップを設定します。

- **Server URL** (必須): HashiCorp Vault のシークレット管理システムとの通信に使用する URL を指定します。
- **Token**: HashiCorp のサーバーの認証に使用するアクセストークンを指定します。
- **CA Certificate**: HashiCorp のサーバーの検証に使用される CA 証明書を指定します。
- **Approle Role_ID**: 認証に Appprole を使用している場合は、ID を指定します。
- **Approle Secret_ID**: Approle 認証に対応するシークレット ID を指定します。
- **Client Certificate**: Hashicorp Vault が必要とする中間証明書など、TLS 認証方法を使用する場合に、PEM でエンコードされたクライアント証明書を指定します。
- **Client Certificate Key**: TLS 認証方法を使用する場合に、PEM でエンコードされた証明書の秘密鍵を指定します。
- **TLS Authentication Role**: TLS 認証方法の使用時にクライアント証明書に対応する Hashicorp Vault でロールまたは証明書名を指定します。指定しないと、Hashicorp Vault は証明書を自動的に一致しようとします。
- **Namespace name**: 名前空間名を指定します(Hashicorp Vault エンタープライズのみ)。
- **Kubernetes role**: Kubernetes 認証の使用時にロール名を指定します。
- **Username**: このサービスの認証に使用されるユーザーのユーザー名を入力します。
- **Password**: このサービスの認証に使用するユーザーに関連付けられたパスワードを入力します。

- **Path to Auth**: **/approle** のデフォルトパス以外のパスを指定します。
- **API Version** (必須): 静的ルックアップの場合は v1 を選択し、バージョンを含むルックアップの場合は v2 を選択します。

LDAP 認証では、HashiCorp の Vault UI で LDAP を設定し、ユーザーに追加するポリシーを設定する必要があります。cubbyhole は、デフォルトのシークレットマウントの名前です。適切なパーミッションがある場合は、他のマウントを作成し、キー値をそれらに書き込むことができます。

ルックアップをテストするには、Hashicorp Vault ルックアップを使用する別の認証情報を作成します。

関連情報

LDAP 認証方式およびそのフィールドの詳細は、LDAP 認証メソッドの Vault ドキュメント を参照してください。

Approle 認証メソッドおよびそのフィールドの詳細は、Vault documentation for AppRole auth method を参照してください。

userpass 認証メソッドおよびそのフィールドの詳細は、Vault documentation for userpass auth method を参照してください。

Kubernetes の auth メソッドおよびそのフィールドの詳細は、Vault documentation for Kubernetes auth method を参照してください。

TLS 証明書認証メソッドおよびそのフィールドの詳細は、Vault documentation for TLS certificates auth method を参照してください。

12.1.7. HashiCorp Vault Signed SSH

Credential Type に **HashiCorp Vault Signed SSH**を選択した場合は、次のメタデータを指定してルックアップを設定します。

- **Server URL** (必須): HashiCorp 署名の SSH のシークレット管理システムとの通信に使用する URL を指定します。
- **Token**: HashiCorp のサーバーの認証に使用するアクセストークンを指定します。
- **CA Certificate**: HashiCorp のサーバーの検証に使用される CA 証明書を指定します。
- **Approle Role_ID**: Approle 認証の ID を指定します。
- **Approle Secret_ID**: Approle 認証に対応するシークレット ID を指定します。
- **Client Certificate**: Hashicorp Vault が必要とする中間証明書など、TLS 認証方法を使用する場合に、PEM でエンコードされたクライアント証明書を指定します。
- **Client Certificate Key**: TLS 認証方法を使用する場合に、PEM でエンコードされた証明書の秘密鍵を指定します。
- **TLS Authentication Role**: TLS 認証方法の使用時にクライアント証明書に対応する Hashicorp Vault でロールまたは証明書名を指定します。指定しないと、Hashicorp Vault は証明書を自動的に一致しようとします。
- **Namespace name**: 名前空間名を指定します(Hashicorp Vault エンタープライズのみ)。

- **Kubernetes role**: Kubernetes 認証の使用時にロール名を指定します。
- **Username**: このサービスの認証に使用されるユーザーのユーザー名を入力します。
- **Password**: このサービスの認証に使用するユーザーに関連付けられたパスワードを入力します。
- **Path to Auth**: **/approle** のデフォルトパス以外のパスを指定します。

関連情報

Approle 認証メソッドおよびそのフィールドの詳細は、Vault documentation for AppRole auth method を参照してください。

Kubernetes の authtication メソッドおよびそのフィールドの詳細は、Kubernetes 認証メソッドの Vault ドキュメント を参照してください。

TLS 証明書認証メソッドおよびそのフィールドの詳細は、Vault documentation for TLS certificates auth method を参照してください。

12.1.8. Microsoft Azure Key Vault

Credential Type に **Microsoft Azure Key Vault** を選択した場合は、次のメタデータを指定してルックアップを設定します。

- **Vault URL (DNS Name)**(必須): Microsoft Azure の鍵管理システムとの通信に使用する URL を指定します。
- **Client ID** (必須): Microsoft Azure Active Directory によって取得された識別子を入力します。
- **Client Secret** (必須): Microsoft Azure Active Directory によって取得されたシークレットを入力します。
- **テナント ID** (必須): Azure サブスクリプション内の Microsoft Azure Active Directory インスタンスに関連付けられている一意の識別子を指定します。
- **Cloud Environment** 適用するクラウド環境を選択します。

12.1.9. Thycotic DevOps Secrets Vault

Credential Type に **Thycotic DevOps Secrets Vault** を選択した場合は、次のメタデータを指定してルックアップを設定します。

- **Tenant** (必須): Thycotic のシークレット管理システムとの通信に使用する URL を指定します
- **Top-level Domain (TLD)**: 統合するシークレット Vault に関連付けられたトップレベルドメインの指定 (com、edu、org など) を提供します。
- **Client ID** (必須): Thycotic のシークレット管理システムによって取得された識別子を入力します。
- **Client Secret** (必須): Thycotic のシークレット管理システムによって取得されたシークレットを入力します。

12.1.10. Thycotic Secret Server

Credential Type に **Thycotic Secrets Server** を選択した場合は、次のメタデータを指定してルックアップを設定します。

- **Secret Server URL** (必須): Thycotic Secrets Server 管理システムとの通信に使用する URL を指定します。
- **Username** (必須): このサービスの認証済みユーザーを指定します。
- **Password** (必須): ユーザーに関連付けられたパスワードを入力します。

第13章 アプリケーション

ServiceNow や Jenkins などの外部アプリケーション用のトークンベースの認証を作成および設定します。トークンベースの認証を使用すると、外部アプリケーションを Automation Controller と簡単に統合できます。

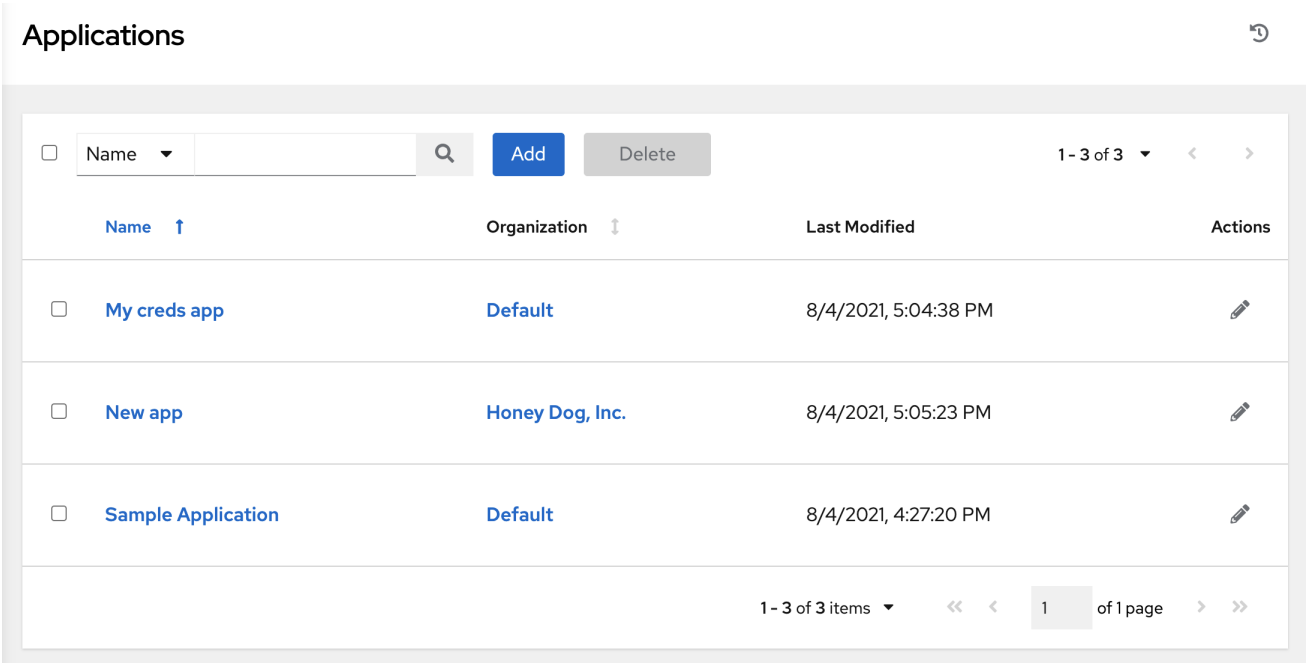
OAuth 2 を使用すると、ログイン情報を公開せずにトークンを使用してアプリケーションとデータを共有できます。このトークンは、読み取り専用に設定できます。

統合する外部アプリケーションを表すアプリケーションを作成し、それ使用してアプリケーションがユーザーの代わりに使用するトークンを作成できます。

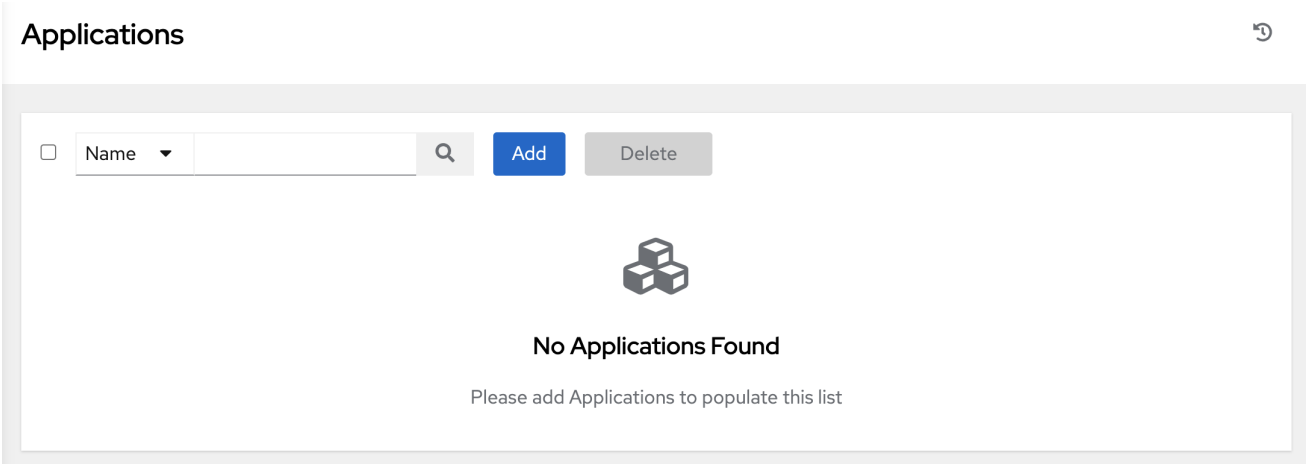
これらのトークンをアプリケーションリソースに関連付けると、特定のアプリケーションに対して発行されたすべてのトークンを管理できるようになります。**アプリケーション** の下でトークンの発行を分離して、システム内のすべてのトークンを取り消すことなく、そのアプリケーションをベースにするトークンをすべて取り消すことができます。

13.1. アプリケーションの使用開始

ナビゲーションパネルから、**Administration → Applications** を選択します。**Applications** ページには、現在 Automation controller によって管理されているすべての利用可能なアプリケーションの検索可能なリストが表示され、**名前** で並べ替えることができます。



アプリケーションが存在しない場合は、アプリケーションを追加するように求められます。



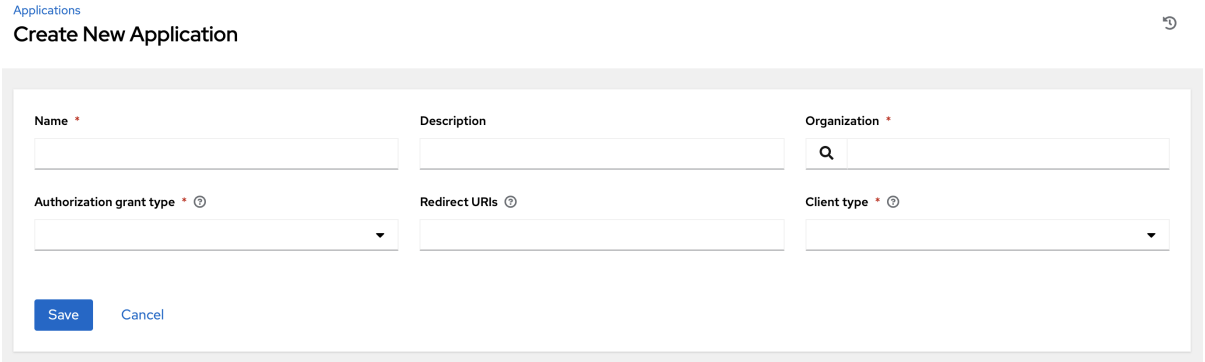
13.2. 新規アプリケーションの作成

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、認証コード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は、推奨されます。

- 外部アプリケーションはユーザーの認証情報を使用してユーザーの代わりに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ため、それらのトークンを簡単に管理できます。たとえば、その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します。

手順

1. ナビゲーションパネルから、**Administration → Applications** を選択します。
2. **Add** をクリックします。**Create New Application** ページが開きます。



3. 以下の詳細を入力します。
   - **Name** (必須): 作成するアプリケーションの名前を入力します。
   - オプション: **Description**: アプリケーションの簡単な説明を入力します。
   - **Organization** (必須): このアプリケーションに関連のある組織を指定します。
   - **Authorization Grant Type** (必須): ユーザーがこのアプリケーションのトークンを取得するために使用する付与タイプを 1 つ選択します。詳細は、**Automation controller 管理ガイド** のアプリケーションセクションの アプリケーション関数 を参照してください。

- **Redirect URI**: 許可される URI のリストをスペースで区切って指定します。これは、付与タイプを **Authorization code** に指定した場合に必須です。
- **Client Type** (必須): クライアントデバイスのセキュリティーレベルを選択します。

4. **Save** をクリックするか、**Cancel** をクリックして変更を破棄します。
クライアント ID がウィンドウに表示されます。

13.2.1. トークンの追加

Tokens を選択すると、アプリケーションにアクセスするためのトークンを持つユーザーのリストが表示されます。

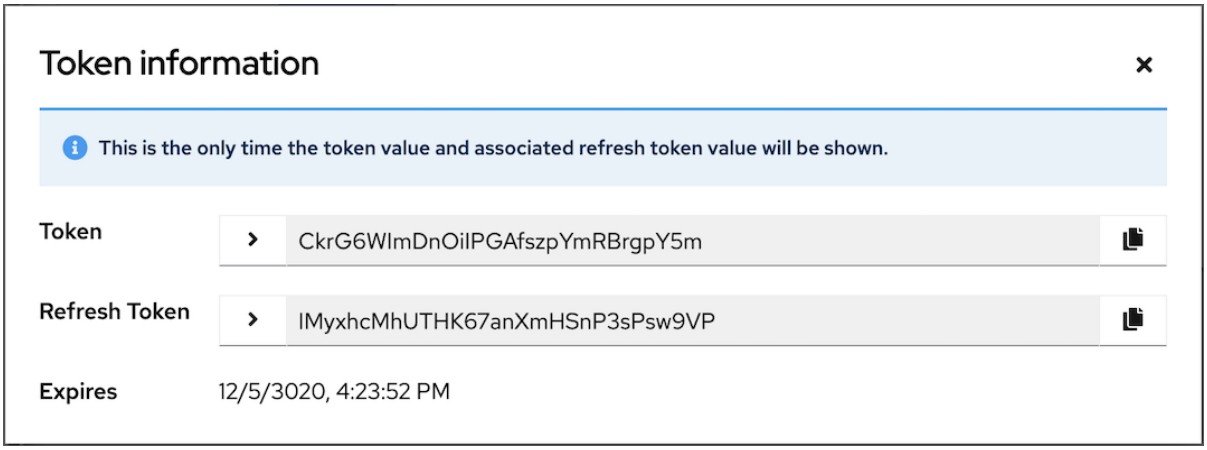
ユーザーの認証トークンを設定します。トークンを関連付けるアプリケーションと、トークンのアクセスレベルを選択できます。

注記

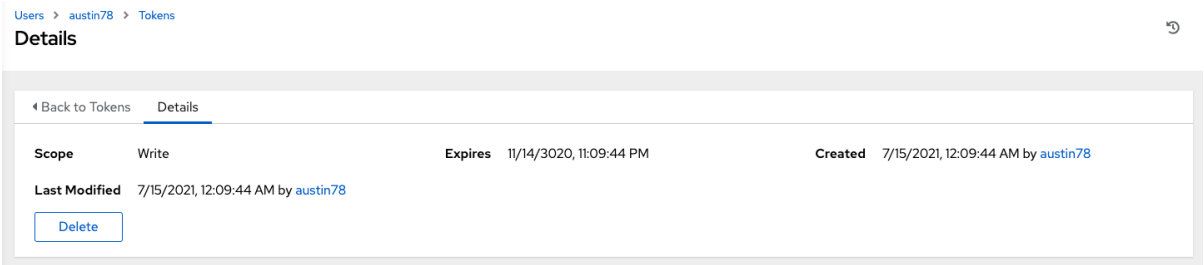
自身のユーザーに対してだけ OAuth 2 トークンを、API または UI から作成できます。つまり、トークンの設定や表示の際にアクセスできるのは、自分のユーザープロファイルのみです。

手順

1. ナビゲーションパネルから、**Access → Users** を選択します。
2. OAuth 2 トークンを設定するユーザーを選択します。
3. ユーザーのプロファイルで **Tokens** タブを選択します。
トークンがない場合には、**Tokens** 画面で追加するようにプロンプトが表示されます。
4. **Add** をクリックして **Create Token** ウィンドウを開きます。
5. 以下の詳細を入力します。
   - **アプリケーション**: トークンを関連付けるアプリケーションの名前を入力します。または、 アイコンをクリックして検索することもできます。これにより、別のウィンドウが開き、利用可能なオプションから選択できるようになります。リストが膨大な場合は、検索バーを使用して名前でフィルタリングします。どのアプリケーションにもリンクされていない Personal Access Token (PAT) を作成する場合は、このフィールドを空白のままにしておきます。
   - オプション: **説明**: トークンの短い説明を入力します。
   - **範囲** (必須): 対象のトークンのアクセスレベルを指定します。
6. **Save** をクリックするか、**Cancel** をクリックして変更を破棄します。
トークンを保存すると、ユーザーに対して新しく作成されたトークンが、トークン情報と有効期限とともに表示されます。

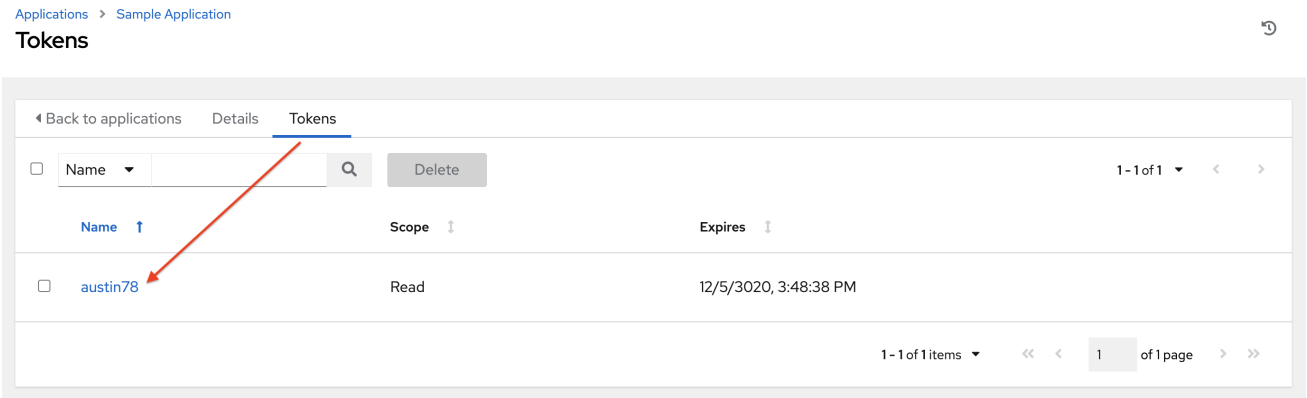


7. トークンが関連付けられているアプリケーションとトークンの有効期限を表示するには、トークンリストビューに移動します。



検証

アプリケーションに適切なトークンを持つユーザーが表示されていることを確認するには、アプリケーションウィンドウの **Tokens** タブを開きます。



関連情報

システム管理者の方で他のユーザーのトークンを作成または削除する必要がある場合は、**Automation controller 管理ガイド** の トークンとセッションの管理 セクションにある取り消しコマンドと作成コマンドを参照してください。

第14章 実行環境

従来の仮想環境とは異なり、実行環境は、システムレベルの依存関係とコレクションベースのコンテンツを組み込むことを可能にするコンテナーイメージです。各実行環境では、ジョブを実行するためのカスタマイズされたイメージを使用できます。また、ジョブの実行時に必要なものだけが含まれます。

14.1. 実行環境の構築

Ansible コンテンツがデフォルトの環境ではなくカスタム仮想環境に依存している場合は、追加の手順を実行する必要があります。各ノードにパッケージをインストールし、ホストシステムにインストールされている他のソフトウェアと適切に連携させ、パッケージの同期を維持する必要があります。

このプロセスを簡素化するために、Ansible コントロールノード として機能するコンテナーイメージをビルドできます。これらのコンテナーイメージは自動化実行環境と呼ばれ、ansible-builder を使用して作成できます。Ansible-runner は上記のようなイメージを利用できるようになります。

14.1.1. ansible-builder のインストール

イメージをビルドするには、Podman または Docker と **ansible-builder** Python パッケージがインストールされている必要があります。

--container-runtime オプションは、使用する Podman または Docker 実行可能ファイルに対応している必要があります。

詳細は、Ansible Builder のクイックスタート または 実行環境の作成と使用 を参照してください。

14.1.2. 実行環境に必要なコンテンツ

実行環境の作成には Ansible-builder を使用します。

実行環境には以下が含まれている必要があります。

- Ansible
- Ansible Runner
- Ansible コレクション
- Python とシステムの依存関係:
  - コレクション内のモジュールまたはプラグイン
  - ansible-base のコンテンツ
  - カスタムのユーザーニーズ

新しい実行環境を構築するには、コレクション、Python 要件、システムレベルのパッケージなど、実行環境に含めるコンテンツを指定した定義が必要です。定義は .yml ファイルである必要があります。

実行環境への移行によって生成された出力の内容には、ファイルにパイプしたり、この定義ファイルに貼り付けたりできる必要なデータの一部が含まれています。

関連情報

詳細は、Migrate legacy venvs to execution environments を参照してください。仮想環境から移行しなかった場合は、実行環境の設定リファレンス で説明されている必要なデータを含む定義ファイルを作成できます。

コレクション開発者は、適切なメタデータを提供することで、コンテンツの要件を宣言できます。

詳細は、依存関係 を参照してください。

14.1.3. イメージをビルドするための YAML ファイルの例

ansible-builder build コマンドは、実行環境定義を入力として受け取ります。実行環境イメージのビルドに必要なビルドコンテキストを出力し、そのイメージをビルドします。このイメージは、ビルドコンテキストを使用して他の場所で再ビルドでき、同じ結果が得られます。デフォルトでは、ビルダーは現在のディレクトリーで **execution-environment.yml** という名前のファイルを検索します。

次の **execution-environment.yml** ファイルの例は、開始点として使用できます。

```
---
version: 3
dependencies:
  galaxy: requirements.yml
```

required.yml の内容:

```
---
collections:
  - name: awx.awx
```

上記のファイルを使用して実行環境を構築し、次のコマンドを実行します。

```
ansible-builder build
...
STEP 7: COMMIT my-awx-ee
--> 09c930f5f6a
09c930f5f6ac329b7ddb321b144a029dbbfcc83bdfc77103968b7f6cdfc7bea2
Complete! The build context can be found at: context
```

すぐに使用できるコンテナーイメージが生成されるだけでなく、ビルドコンテキストが保持されます。このイメージは、**docker build** や **podman build** などの任意のツールを使用して、別の時間または場所で再ビルドできます。

関連情報

ansible-builder build コマンドの詳細は、Ansible の CLI Usage ドキュメントを参照してください。

14.1.4. 実行環境のマウントオプション

証明書の追加の方法として、実行環境の再構築が 1 つ挙げられますが、ホストから証明書を継承する方がより便利な解決策となります。仮想マシンベースのインストールの場合、Automation Controller はジョブの実行時に実行環境にシステムトラストストアを自動的にマウントします。

Job Settings ページの Paths to expose to isolated jobsフィールドで実行環境のマウントオプションとマウントパスをカスタマイズできます。このフィールドでは、Podman スタイルのボリュームマウント構文がサポートされています。

関連情報

詳細は、Podman ドキュメント を参照してください。

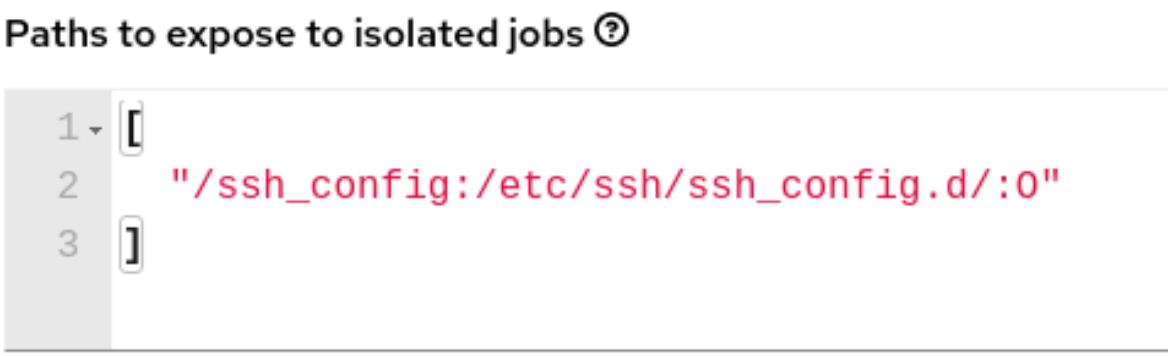
14.1.4.1. 実行環境マウントオプションのトラブルシューティング

実行環境のカスタマイズにより実行環境イメージに **/etc/ssh/*** ファイルが追加された場合、SSH エラーが発生する場合があります。たとえば、**/etc/ssh/ssh_config.d:/etc/ssh/ssh_config.d:O** パスを公開すると、コンテナーをマウントできるようになりますが、所有権のアクセス許可は正しくマップされません。

このエラーが発生した場合、または古いバージョンの Automation Controller からアップグレードした場合は、次の手順を使用してください。

手順

1. マウントされたボリュームのコンテナー所有権を **root** に変更します。
2. ナビゲーションパネルから、**Job Settings** を選択します。
3. 現在の例を使用して、**Paths to expose to isolated jobs**でパスを公開します。



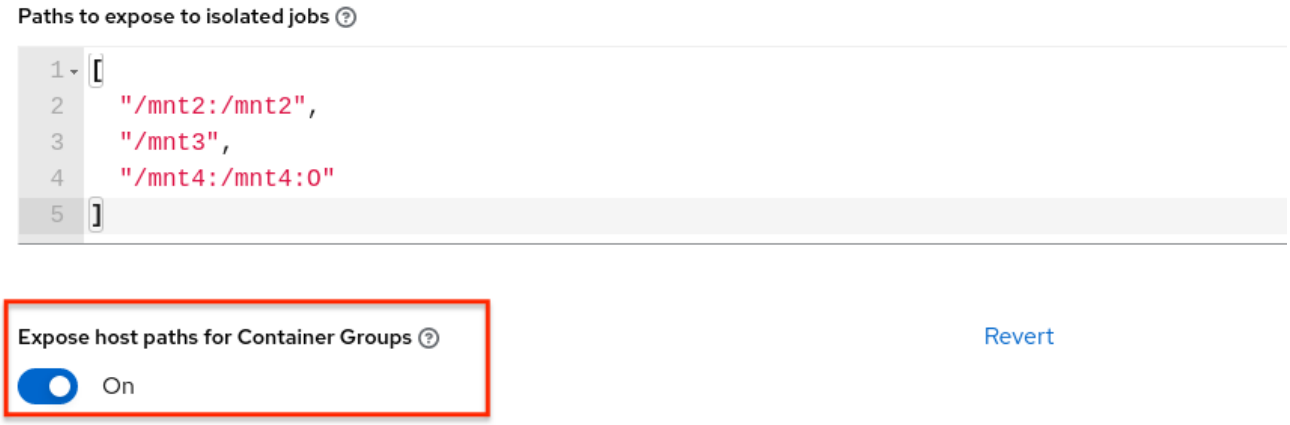
注記

:O オプションはディレクトリーに対してのみサポートされます。特にシステムパスを指定する場合は、できるだけ詳細に指定してください。**/etc** または **/usr** を直接マウントすると、トラブルシューティングが困難になる影響があります。

これにより、次の例と同様のコマンドを実行するように Podman に通知されます。このコマンドにより、設定がマウントされ、**ssh** コマンドが期待どおりに機能するようになります。

```
podman run -v /ssh_config:/etc/ssh/ssh_config.d/:O ...
```

OpenShift または Kubernetes コンテナー内の分離されたパスを HostPath として公開するには、以下の設定を想定してください。



有効にするには **Expose host paths for Container Groups**を **On** に設定します。

Playbook が実行されると、結果として得られる Pod 仕様は次の例のようになります。**volumeMounts** セクションと **Volumes** セクションの詳細に注目してください。

```
apiVersion: v1
kind: Pod
spec:
  containers:
  - image: registry.redhat.io/ansible-automation-platform-22/ee-minimal-rhel8
    args:
      - ansible-runner
      - worker
      - --private-data-dir=/runner
    volumeMounts:
      - mountPath: /mnt2
        name: volume-0
        readOnly: true
      - mountPath: /mnt3
        name: volume-1
        readOnly: true
      - mountPath: /mnt4
        name: volume-2
        readOnly: true
  volumes:
    - hostPath:
        path: /mnt2
        type: ""
      name: volume-0
    - hostPath:
        path: /mnt3
        type: ""
      name: volume-1
    - hostPath:
        path: /mnt4
        type: ""
      name: volume-2
```

14.1.4.2. 実行ノード内のディレクトリーを実行環境コンテナーにマウントする

Ansible Automation Platform 2.1.2 では、**O** および **z** オプションのみが使用可能でした。Ansible Automation Platform 2.2 以降では、**rw** などの追加のオプションが利用可能になりました。これは、NFS ストレージを使用する場合に便利です。

手順

1. ナビゲーションパネルから、**Settings → Job Settings** を選択します。
2. **Paths to expose to isolated jobs**を編集します。
   - 実行ノードまたはハイブリッドノードからコンテナーにボリュームをマウントするパスのリストを入力します。
   - 1 行に 1 つのパスを入力します。
   - サポートされている形式は、**HOST-DIR[:CONTAINER-DIR[:OPTIONS]** です。許可されるパスは、**z**、**O**、**ro**、および **rw** です。

     例

     ```
     [
       "/var/lib/awx/.ssh:/root/.ssh:O"
     ]
     ```

   - **rw** オプションの場合、SELinux ラベルを正しく設定します。たとえば、**/foo** ディレクトリーをマウントするには、次のコマンドを実行します。

     ```
     sudo su
     ```

     ```
     mkdir /foo
     ```

     ```
     chmod 777 /foo
     ```

     ```
     semanage fcontext -a -t container_file_t "/foo(/.*)?"
     ```

     ```
     restorecon -vvFR /foo
     ```

awx ユーザーには、少なくともこのディレクトリーでの読み取りまたは書き込みの許可が必要です。ここでは権限を **777** に設定します。

関連情報

ボリュームのマウントの詳細は、Podman ドキュメントの podman-run (1)の --volume オプション を参照してください。

14.2. ジョブテンプレートへの実行環境の追加

前提条件

- 実行環境の構築 で説明されているように、実行環境は ansible-builder を使用して作成されている必要があります。実行環境が作成されたら、その環境を使用してジョブを実行できます。Automation controller UI を使用して、ジョブテンプレートで使用する実行環境を指定します。

- 実行環境がグローバルに使用できるように指定されているか、組織に関連付けられているかに応じて、ジョブで実行環境を使用するには、適切なレベルの管理者権限が必要になります。組織に関連付けられた実行環境では、組織管理者がそれらの実行環境でジョブを実行できる必要があります。

- 認証情報が割り当てられた実行環境を使用するジョブまたはジョブテンプレートを実行する前に、認証情報にユーザー名、ホスト、およびパスワードが含まれていることを確認してください。

手順

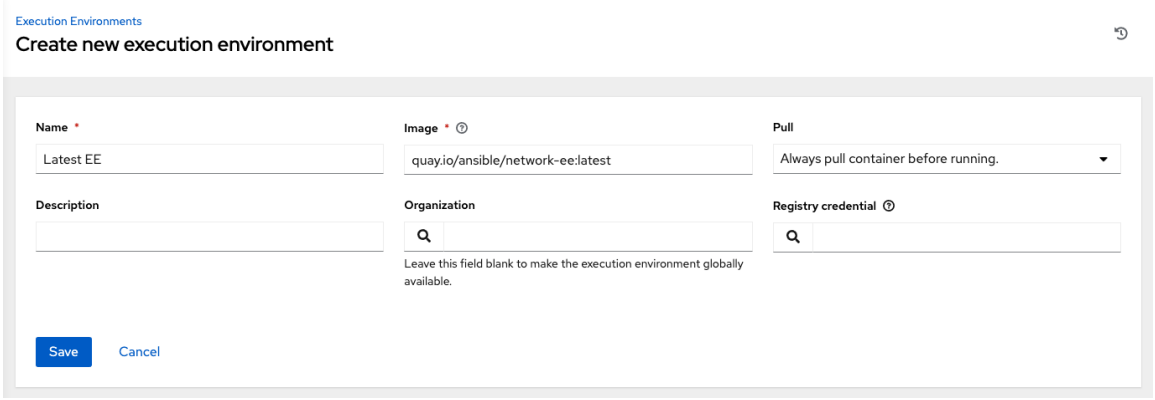
1. ナビゲーションパネルから、**Administration → Execution Environments** を選択します。
2. **Add** をクリックして実行環境を追加します。
3. 以下のフィールドに該当する詳細を入力します。
    - **Name** (必須): 実行環境の名前を入力します。
    - **Image** (必須): イメージ名を入力します。イメージ名には、完全な場所 (リポジトリー)、レジストリー、イメージ名、および **quay.io/ansible/awx-ee:latestrepo/project/image-name:tag** のサンプル形式のバージョンタグが必要です。
    - オプション: **Pull**: ジョブを実行するときのプルのタイプを選択します。
        - **Always pull container before running**: コンテナーの最新のイメージファイルをプルします。
        - **Only pull the image if not present before running**何も指定されていない場合のみ、最新のイメージをプルします。
        - **Never pull container before running** コンテナーイメージの最新バージョンをプルしないでください。

        注記

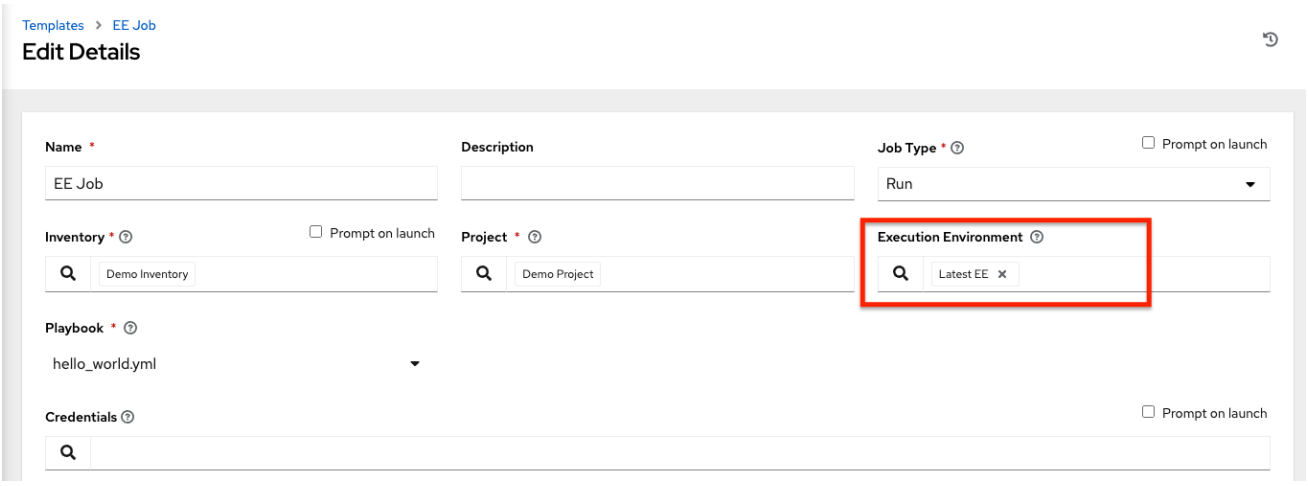
        プルのタイプを設定しなかった場合、値はデフォルトで **Only pull the image if not present before running** になります。

    - オプション: **説明**:
    - オプション: **組織**: この実行環境を使用する組織を具体的に割り当てます。実行環境を複数の組織間で使用できるようにするには、このフィールドを空白のままにします。
    - **Registry credential**: イメージに保護されたコンテナーレジストリーがある場合は、それにアクセスするための認証情報を提供します。

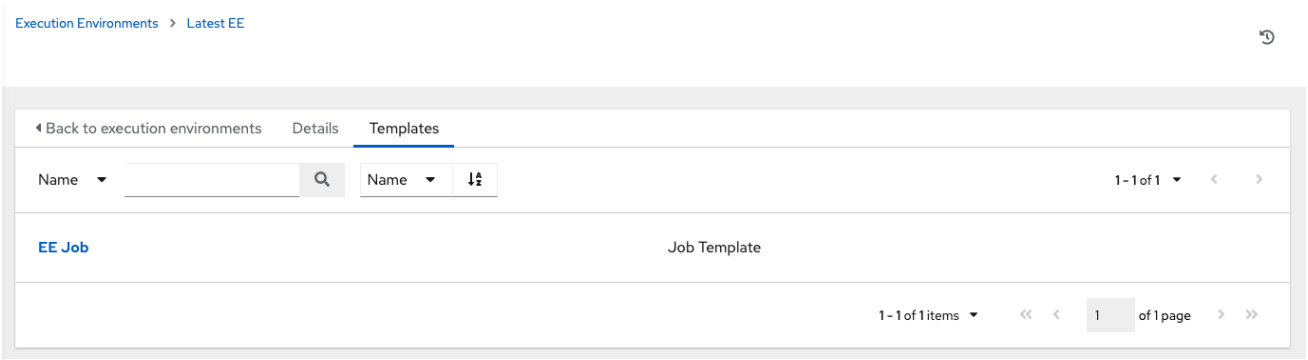


4. **Save** をクリックします。
   新しく追加した実行環境をジョブテンプレートで使用する準備ができました。

5. ジョブテンプレートに実行環境を追加するには、次の例に示すように、ジョブテンプレートの **Execution Environment** フィールドに実行環境を指定します。



実行環境をジョブテンプレートに追加すると、それらのテンプレートは実行環境の **Templates** タブにリストされます。



第15章 実行環境の設定リファレンス

このセクションには、実行環境の定義に関連する参考情報が含まれています。実行環境のコンテンツは YAML ファイルで定義します。デフォルトでは、このファイルの名前は **execution_environment.yml** です。このファイルは、Ansible Builder にビルド指示ファイル (Podman の場合は Containerfile、Docker の場合は docker) を作成し、コンテナーイメージのコンテキストをビルドする方法を指示します。

注記

Ansible Builder 3.x の定義スキーマはここに記載されています。古いバージョンの Ansible Builder を実行している場合は、古いスキーマバージョンが必要です。詳細は、こちら のドキュメントの古いバージョンを参照してください。Red Hat はバージョン 3 を使用することを推奨します。バージョン 3 では、過去のバージョンより設定可能なオプションや機能がはるかに多くなっています。

15.1. 実行環境定義の例

実行環境のイメージをビルドするには、定義ファイルを作成する必要があります。ファイルは YAML 形式です。

定義ファイルで Ansible Builder のバージョンを指定する必要があります。デフォルトのバージョンは 1 です。

次の定義ファイルは Ansible Builder バージョン 3 を使用しています。

```
version: 3
build_arg_defaults:
  ANSIBLE_GALAXY_CLI_COLLECTION_OPTS: '--pre'
dependencies:
  galaxy: requirements.yml
  python:
    - six
    - psutil
  system: bindep.txt
images:
  base_image:
    name: registry.redhat.io/ansible-automation-platform-24/ee-minimal-rhel8:latest
additional_build_files:
    - src: files/ansible.cfg
      dest: configs
additional_build_steps:
  prepend_galaxy:
    - ADD _build/configs/ansible.cfg /home/runner/.ansible.cfg
  prepend_final: |
    RUN whoami
    RUN cat /etc/os-release
  append_final:
    - RUN echo This is a post-install command!
    - RUN ls -la /etc
```

15.2. 設定オプション

定義ファイルで次の設定 YAML キーを使用します。

Ansible Builder 3.x 実行環境定義ファイルは、次の 7 つの最上位セクションを受け入れます。

- additional_build_files
- additional_build_steps
- build_arg_defaults
- dependencies
- images
  - image verification
- options
- version

15.2.1. additional_build_files

ビルドファイルは、ビルドコンテキストディレクトリーに追加する内容を指定します。これらは、任意のビルド段階で **Additional_build_steps** によって参照またはコピーできます。

形式はディクショナリー値のリストで、各リスト項目に **src** および **dest** のキーと値を含めます。

各リスト項目は、次の必須キーを含むディクショナリーである必要があります。

src	ビルドコンテキストディレクトリーにコピーするソースファイルを指定します。 これは、**/home/user/.ansible.cfg** などの絶対パス、またはファイルに対する相対パスで指定できます。相対パスは、1 つ以上のファイルに一致する glob 式にすることができます (**files/*.cfg** など)。絶対パスには正規表現を含めることができない点に注意してください。**src** がディレクトリーの場合、そのディレクトリーの内容全体が **dest** にコピーされます。
dest	ソースファイル (例: **files/configs**) が含まれるビルドコンテキストディレクトリーの **_build** サブディレクトリーの下にあるサブディレクトリーパスを指定します。 この値を絶対パスで指定することや、パス内に **..** を追加できません。このディレクトリーが存在しない場合は、自動的に作成されます。  **注記** **ansible.cfg** ファイルを使用してプライベートアカウントのトークンとその他の設定を Automation Hub サーバーに渡す場合、ここに設定ファイルのパスを文字列としてリストすると、ビルド引数としてビルドの初期フェーズに含めることができます。

15.2.2. additional_build_steps

ビルドステップは、任意のビルドフェーズのカスタムビルドコマンドを指定するものです。これらのコマンドは、コンテナーランタイムのビルド指示ファイル (Containerfile や Dockerfile など) に直接挿入されます。コマンドは、コンテナー化ツールで必要なルールに準拠する必要があります。

ビルドステップは、イメージ作成プロセスのどの段階の前後にも追加できます。たとえば、依存関係をインストールする前に **git** をインストールする必要がある場合は、ベースビルド段階の最後にビルドステップを追加できます。

有効なキーは次のとおりです。それぞれが複数行の文字列または文字列のリストをサポートします。

append_base	基本イメージの構築後に挿入するコマンド。
append_builder	ビルダーイメージの構築後に挿入するコマンド。
append_final	最終イメージの構築後に挿入するコマンド。
append_galaxy	Galaxy イメージの構築後に挿入するコマンド。
prepend_base	基本イメージを構築する前に挿入するコマンド。
prepend_builder	ビルダーイメージを構築する前に挿入するコマンド。
prepend_final	最終イメージを構築する前に挿入するコマンド。
prepend_galaxy	Galaxy イメージを構築する前に挿入するコマンド。

15.2.3. build_arg_defaults

これにより、ビルド引数のデフォルト値がディクショナリーとして指定されます。

これは、**--build-arg** CLI フラグの代わりに使用する方法です。

Ansible Builder は次のビルド引数を使用します。

ANSIBLE_GALAXY_CLI_COLLECTION_OPTS	**-pre** フラグやその他のフラグを渡して、プレリリースコレクションのインストールを有効にできます。
ANSIBLE_GALAXY_CLI_ROLE_OPTS	使用すると、**--no-deps** などのフラグをロールのインストールに渡すことができます。
PKGMGR_PRESERVE_CACHE	これは、イメージのビルドプロセス中にパッケージマネージャーのキャッシュがクリアされる頻度を制御します。 この値が設定されていない場合 (デフォルト)、キャッシュは頻繁にクリアされます。値が **always** の場合、キャッシュはクリアされません。それ以外の値を指定すると、最終ビルド段階でシステムの依存関係がインストールされた後にのみキャッシュが強制的にクリアされます。

Ansible Builder は、**build_arg_defaults** 内で指定された値をビルド指示ファイルにハードコーディングするため、コンテナーのビルドを手動で実行した場合でも値は保持されます。

定義と CLI **build-arg** フラグを使用したコマンドラインで同じ変数を指定すると、CLI の値によって定義の値がオーバーライドされます。

15.2.4. 依存関係

ansible-core、**ansible-runner**、Python パッケージ、システムパッケージ、コレクションなど、最終イメージにインストールする依存関係を指定します。Ansible Builder は、ユーザーがインストールする Ansible コレクションの依存関係を自動的にインストールします。

一般に、標準の構文を使用してパッケージのバージョンを制限できます。**dnf**、**pip**、**ansible-galaxy**、またはその他のパッケージ管理ユーティリティーに渡すのと同じ構文を使用します。パッケージまたはコレクションを別のファイルで定義し、定義ファイルの **dependencies** セクションで上記のファイルを参照することもできます。

有効なキーは次のとおりです。

ansible_core	インストールされる **ansible-core** Python パッケージのバージョン。
	
	この値は、キー (**package_pip**) が 1 つ含まれるディクショナリーです。**package_pip** 値は、インストールのために pip に直接渡され、pip がサポートする任意の形式に指定できます。以下は、値の例です。
	
	ansible_core:
	  package_pip: ansible-core
	ansible_core:
	  package_pip: ansible-core==2.14.3
	ansible_core:
	  package_pip: https://github.com/example_user/ansible/archive/refs/heads/ansible.tar.gz

ansible_runner	インストールする Ansible Runner Python パッケージのバージョン。
	
	この値は、キー (**package_pip**) が 1 つ含まれるディクショナリーです。**package_pip** 値は、インストールのために pip に直接渡され、pip がサポートする任意の形式に指定できます。以下は、値の例です。
	
	ansible_runner:
	  package_pip: ansible-runner
	ansible_runner:
	  package_pip: ansible-runner==2.3.2
	ansible_runner:
	  package_pip: https://github.com/example_user/ansible-runner/archive/refs/heads/ansible-runner.tar.gz

galaxy	Ansible Galaxy からインストールするコレクション。 これは、ファイル名、ディクショナリー、または Ansible Galaxy **required.yml** ファイルの複数行の文字列表現です。要件ファイル形式の詳細は、Galaxy ユーザーガイド を参照してください。

python	Python のインストール要件。 これには、ファイル名または要件のリストなどを使用できます。Ansible Builder は、**requirements-parser** ライブラリーを使用して、すべてのコレクションのすべての Python 要件ファイルを 1 つのファイルに結合します。 このライブラリーは、他のファイルへの参照を含む複雑な構文をサポートします。多数のコレクションに同じ **パッケージ名** が必要な場合、Ansible Builder はそれらを 1 つのエントリーにまとめ、制約を結合します。 Ansible Builder は、コレクションにパッケージが依存関係としてリストされている場合でも、Python 依存関係の結合ファイル内の一部のパッケージを除外します。これには、テストパッケージと、Ansible 自体を提供するパッケージが含まれます。完全なリストは、**src/ansible_builder/_target_scripts/introspect.py** の **EXCLUDE_REQUIREMENTS** で確認できます。 このような除外されるパッケージ名のいずれかを含める必要がある場合は、**introspect** コマンドの **--user-pip** オプションを使用して、ユーザー要件ファイルにそのパッケージ名をリストします。 この方法で提供されたパッケージは、除外された Python パッケージのリストに対して処理されません。
python_interpreter	dnf によってインストールされる Python システムパッケージ名 (**package_system**) または使用される Python インタープリターへのパス (**python_path)** を定義するディクショナリー。
システム	インストールされるシステムパッケージ (bindep 形式)。ファイル名または要件のリストを指定できます。 bindingep の詳細は、OpenDev のドキュメント を参照してください。 システムパッケージの場合、**bindep** 形式を使用してクロスプラットフォーム要件を指定すると、実行環境が使用するパッケージ管理システムによってシステムパッケージをインストールできます。コレクションでは、**platform:rpm** の要件を指定する必要があります。Ansible Builder は、複数のコレクションのシステムパッケージエントリーを 1 つのファイルに結合します。プロファイルのない要件 (ランタイム要件) のみがイメージにインストールされます。互いに重複している多数のコレクションのエントリーを、結合ファイルに統合できます。

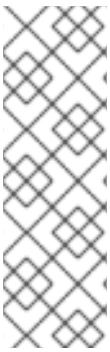
次の例では、さまざまな依存関係を含むファイル名を使用しています。

```
dependencies:
  python: requirements.txt
  system: bindep.txt
  galaxy: requirements.yml
  ansible_core:
     package_pip: ansible-core==2.14.2
  ansible_runner:
     package_pip: ansible-runner==2.3.1
```

```
  python_interpreter:
    package_system: "python310"
    python_path: "/usr/bin/python3.10"
```

この例ではインライン値を使用します。

```
dependencies:
  python:
    - pywinrm
  system:
    - iputils [platform:rpm]
  galaxy:
    collections:
      - name: community.windows
      - name: ansible.utils
        version: 2.10.1
  ansible_core:
    package_pip: ansible-core==2.14.2
  ansible_runner:
    package_pip: ansible-runner==2.3.1
  python_interpreter:
    package_system: "python310"
    python_path: "/usr/bin/python3.10"
```



注記

これらの依存関係ファイル (**requirements.txt**、**bindep.txt**、**requirements.yml**) のいずれかがコレクションの **build_ignore** に含まれている場合、ビルドが失敗します。

コレクションのメンテナーは、**introspect** コマンドを使用して、期待される要件を ansible-builder が認識していることを確認できます。

```
ansible-builder introspect --sanitize ~/.ansible/collections/
```

--sanitize オプションは、すべてのコレクション要件を確認し、重複を削除します。また、通常は除外される Python 要件も削除されます (**Python** の依存関係を参照)。

-v3 オプションを使用して **イントロスペクト** を行い、除外されている要件に関するログメッセージを確認します。

15.2.5. images

使用するベースイメージを指定します。少なくとも、ベースイメージのソース、イメージ、およびタグを指定する必要があります。基本イメージはオペレーティングシステムを提供し、いくつかのパッケージも提供することもできます。標準の **host/namespace/container:tag** 構文を使用してイメージを指定します。代わりに Podman または Docker のショートカット構文を使用することもできますが、完全な定義の方が信頼性と移植性が高くなります。

このセクションの有効なキーは次のとおりです。

base_image	実行環境の親イメージを定義するディクショナリー。 使用するコンテナーイメージには **name** キーを指定する必要があります。イメージがリポジトリー内でミラーリングされているが、イメージの元の署名キーで署名されている場合は、**signature_original_name** キーを使用します。

15.2.6. Image verification

podman コンテナーランタイムを使用している場合は、署名されたコンテナーイメージを検証できます。

Container-policy CLI オプションを設定して、コンテナーイメージの署名検証のための Podman **policy.json** ファイルに関連してこのデータがどのように使用されるかを制御します。

- **ignore_all** ポリシー: 署名検証が実行されないビルドの **context directory <context>** に、**policy.json** ファイルを生成します。
- **システム** ポリシー: 署名の検証は、標準のシステムの場所にある既存の **policy.json** ファイルを使用して実行されます。**ansible-builder** は、これらのファイル内のコンテンツに対する責任を負わず、ユーザーがコンテンツを完全に制御できます。
- **Signature_required** ポリシー: **ansible-builder** は、コンテナーイメージ定義で、ビルド中にイメージを検証するために使用される、ビルドの **context directory <context>** t内に **policy.json** ファイルを生成します。

15.2.7. options

Ansible Builder のランタイム機能に影響を与える可能性のあるキーワードまたはオプションのディクショナリー。

このセクションの有効なキーは次のとおりです。

- context_init: コンテナーの **ENTRYPOINT** および **CMD** ディレクティブ (および関連する動作) のカスタマイズを可能にするキーを含むディクショナリー。これらの動作のカスタマイズは高度なタスクであり、デバッグが困難な障害が発生する可能性があります。下記のデフォルト値は、複数の密接な動作を制御するものです。そのため、いずれかの値をオーバーライドすると、このディクショナリー内の残りのデフォルトがすべてスキップされます。
  有効なキーは次のとおりです。
  - **cmd** : **CMD** Containerfile ディレクティブのリテラル値。デフォルト値は **["bash"]** です。
  - **entrypoint**: **ENTRYPOINT** Containerfile ディレクティブのリテラル値。デフォルトのエントリーポイントの動作は、サブプロセスへのシグナルの伝播を処理するだけでなく、実行時に、コンテナーユーザーが有効で書き込み可能なホームディレクトリーが含まれる適切な環境が含まれます。このディレクトリーは、**/etc/passwd** で表現されており、**HOME** 環境変数と一致するように設定されています。デフォルトのエントリーポイントスクリプトは、ユーザーのランタイム環境を適切に調整できない場合に、**stderr** に警告を発行できます。この動作は無視されるか、致命的なエラーに格上げされる可能性があります。詳細は、**エントリーポイント** ターゲットスクリプトのソースを参照してください。
    デフォルト値は **"/opt/builder/bin/entrypoint"、"dumb-init"** です。

    - package_pip : エントリーポイントのサポートのために pip を使用してインストールするパッケージ。このパッケージは、最終的なビルドイメージにインストールされます。
      デフォルト値は **Dam-init==1.2.5** です。

- package_manager_path: 使用するパッケージマネージャー (dnf または microdnf) へのパスを含む文字列。デフォルトは **/usr/bin/dnf** です。この値は、**dependency** で指定されている場合、**assemble** スクリプトによるビルドフェーズ中に Python インタープリターをインストールするために使用されます。

- Skip_ansible_check: このブール値は、Ansible および Ansible Runner のインストールのチェックを最終イメージで実行するかどうかを制御します。
  このチェックを実行しない場合は、この値を **True** に設定します。

  デフォルトは **False** です。

- Relax_passwd_permissions : このブール値は、**ルート** グループ (GID 0) に、最終コンテナーイメージ内の **/etc/passwd** への書き込みパーミッションを明示的に付与するかどうかを制御します。デフォルトのエントリーポイントスクリプトは、完全に機能する POSIX ユーザー環境とホームディレクトリーを確保するために、動的に作成されたユーザーを使用して一部のコンテナーランタイムで **/etc/passwd** の更新を試みることができます。この機能を無効にすると、有効で書き込み可能なホームディレクトリーを持つユーザーを **/etc/passwd** にリストする必要があるソフトウェア機能 (たとえば、ansible-core の **async** や **~username** シェル拡張など) が失敗する可能性があります。
  デフォルトは **True** です。

- workdir: 最終コンテナーイメージで開始される新しいプロセスのデフォルトとして設定されている現在の作業ディレクトリー。一部のコンテナーランタイムは、**root** (GID 0) グループ内で動的に作成されたユーザーの **HOME** としてもこの値を使用します。この値を指定すると、ディレクトリーがまだ存在しない場合はディレクトリーが作成され、**root** グループの所有権が設定され、**rwx** グループのアクセス許可が再帰的に適用されます。
  デフォルト値は **/runner** です。

- user: 最終的なコンテナーイメージのデフォルトユーザーとして使用するユーザー名または UID を設定します。
  デフォルト値は **1000** です。

オプションの例:

```
options:
  container_init:
    package_pip: dumb-init>=1.2.5
    entrypoint: '["dumb-init"]'
    cmd: '["csh"]'
  package_manager_path: /usr/bin/microdnf
  relax_password_permissions: false
  skip_ansible_check: true
  workdir: /myworkdir
  user: bob
```

15.2.8. version

実行環境定義ファイルのスキーマバージョンを設定する整数値。

デフォルトは **1** です。

Ansible Builder 3.x を使用している場合、値は **3** である必要があります。

15.3. AWX のデフォルトの実行環境

test/data/pytz の例では、定義に **awx.awx** コレクションが必要です。ルックアッププラグイン **awx.awx.tower_schedule_rrule** が動作するには、PyPI **pytz** と別のライブラリーが必要です。**test/data/pytz/execution-environment.yml** ファイルが **ansible-builder build** コマンドに指定されている場合、イメージ内にコレクションがインストールされ、コレクション内の **requirements.txt** ファイルが読み取られて、**pytz** がイメージにインストールされます。。

これらの変数をプライベートデータディレクトリー内の **env/settings** ファイル内に配置して、生成されたイメージを **ansible-runner** プロジェクト内で使用できます。

```
---
container_image: image-name
process_isolation_executable: podman # or docker
process_isolation: true
```

awx.awx コレクションは、デフォルトの AWX に含まれるコンテンツのサブセットです。

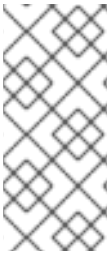
詳細は、awx-ee repository を参照してください。

第16章 プロジェクト

プロジェクトとは、Automation Controller にある Ansible Playbook の論理的コレクションのことです。Playbook と Playbook ディレクトリーはさまざまな方法で管理できます。

- Automation Controller サーバー上にあるプロジェクトのベースパスの下にこれらを手動で配置します。
- Playbook を Automation Controller でサポートされているソースコード管理 (SCM) システムに配置します。これらには、Git、Subversion、Mercurial、Red Hat Insights が含まれます。

Red Hat Insights プロジェクトの作成の詳細は、Insights 修復の設定 を参照してください。



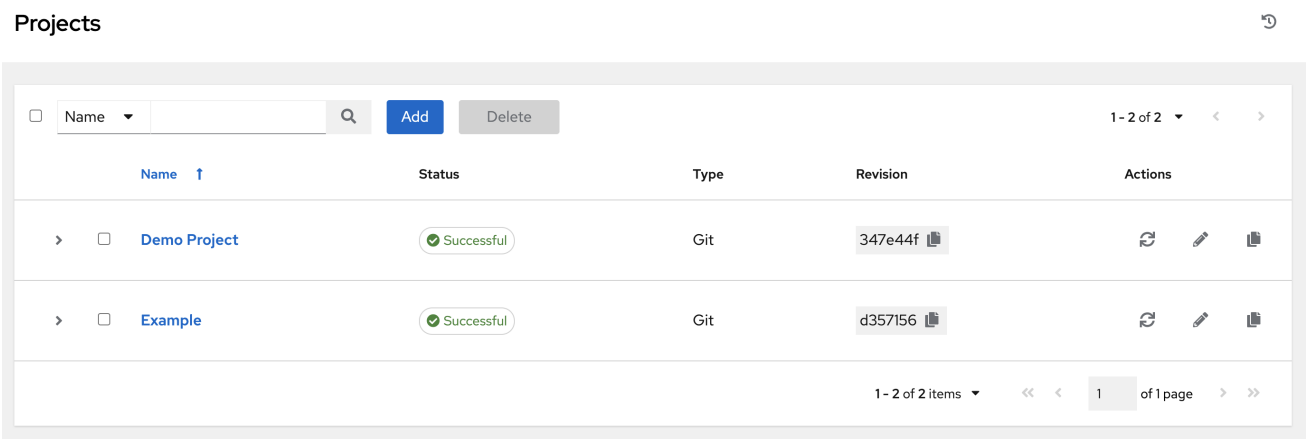
注記

プロジェクトのベースパスは **/var/lib/awx/projects** です。ただし、これはシステム管理者が変更できます。これは **/etc/tower/conf.d/custom.py** で設定されます。

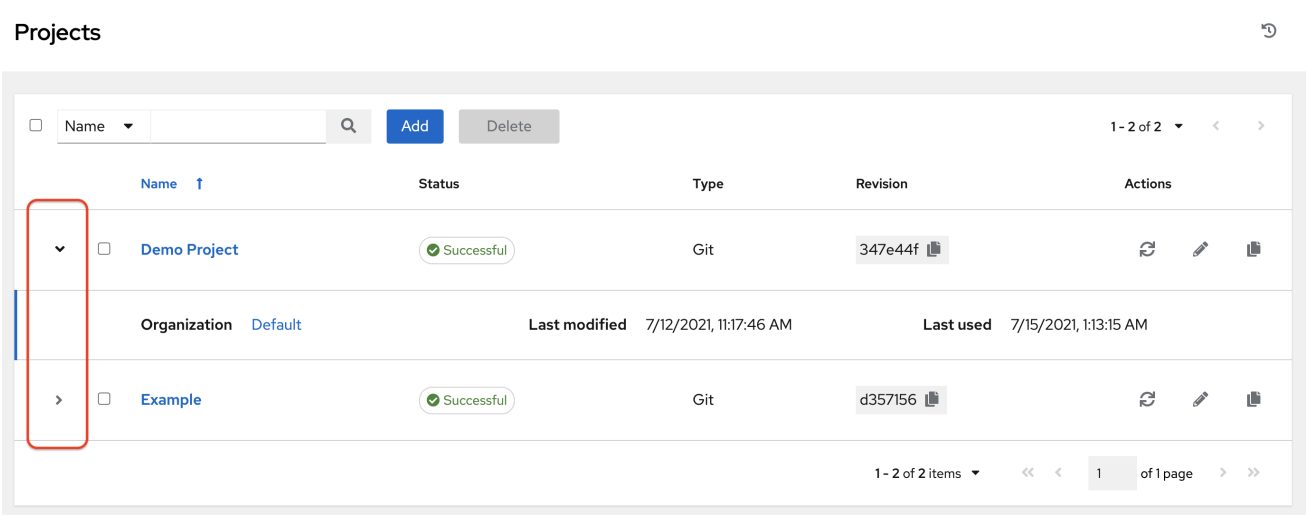
設定を誤るとインストールが無効になる可能性があるため、このファイルを編集する場合は注意してください。

プロジェクトページには、現在利用可能なプロジェクトのリストが表示されます。

Automation controller は、最初に作業できる **デモプロジェクト** を提供します。



デフォルトビューは、折りたたまれており (**Compact**)、プロジェクト名とステータスが表示されていますが、各エントリーの横にある > を使用して展開すると、詳細情報を表示できます。



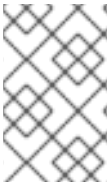
リストされているプロジェクトごとに、各プロジェクトの横にあるアイコンを使用して、最新の SCM リビジョン を取得し、プロジェクトの属性を編集 して、コピー します。

プロジェクトは、関連ジョブの実行中に更新できます。

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。

ステータス はプロジェクトの状態を示し、以下のいずれかになる可能性があります (特定のステータスタイプでビューをフィルターできることにも留意してください)。

- **Pending** - ソースコントロールの更新は作成されましたが、まだキューに入れられていない、または開始されていません。ジョブ (ソースコントロールの更新だけでなく) は、システムで実行できる状態になるまで保留状態になります。準備ができていない理由としては次のことが考えられます。
  - 現在実行中の依存関係があるため、完了するまで待つ必要があります。
  - 設定されている場所で実行するのに十分な容量がありません。
- **Waiting** - ソースコントロールの更新はキューに入れられており、実行を待機中です。
- **Running** - ソースコントロールの更新が進行中です。
- **成功** - このプロジェクトの最後のソースコントロールの更新が成功しました。
- **Failed** - このプロジェクトの最後のソースコントロールの更新が失敗しました。
- **Error** - 最後のソースコントロール更新ジョブは実行にまったく失敗しました。
- **取り消し** - このプロジェクトの最後のソースコントロールの更新が取り消されました。
- **未更新** - このプロジェクトはソースコントロール用に設定されていますが、更新されていません。
- **OK** - プロジェクトはソースコントロール用に設定されておらず、正しく配置されています。
- **Missing** - **/var/lib/awx/projects** のプロジェクトベースパスにプロジェクトがありません。これは、手動またはソースコントロール管理プロジェクトに適用されます。



注記

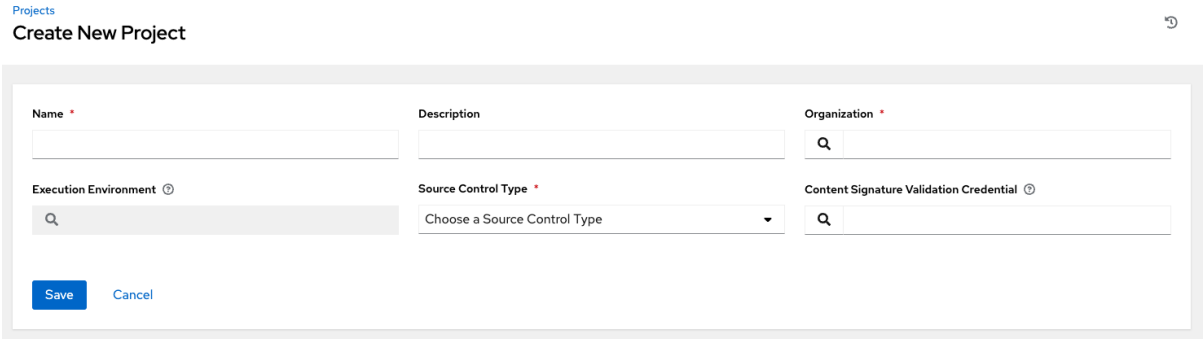
認証情報タイプが **手動** のプロジェクトでは、SCM タイプの認証情報として再設定されない限りソースコントロールベースのアクションを更新したり、スケジュールしたりすることはできません。

16.1. 新しいプロジェクトの追加

UI の **Resources** セクションで、Automation Controller のプロジェクトを管理できます。

手順

1. ナビゲーションパネルから、**Resources** → **Projects** を選択します。
2. **Projects** ページで **Add** をクリックして **Create Project** ウィンドウを起動します。



3. 以下の必須フィールドに適切な情報を入力します。
   - **Name** (必須)
   - オプション: **Description**
   - **Organization** (必須): プロジェクトには少なくとも 1 つの組織が必要です。ここで組織を 1 つ選択してプロジェクトを作成します。プロジェクトの作成時に、組織を追加できます。
   - オプション: **Execution Environment**: このプロジェクトを実行するための実行環境の名前を入力するか、既存の環境のリストから検索します。詳細は、**Red Hat Ansible Automation Platform アップグレードおよび移行ガイド** の 実行環境への移行 を参照してください。
   - **Source Control Type** (必須): このプロジェクトに関連付けられた SCM タイプをメニューから選択します。選択したタイプに応じて、次のセクションのオプションが使用可能になります。詳細は、Playbook の手動管理 または ソースコントロールを使用した Playbook の管理 を参照してください。
   - オプション: **Content Signature Validation Credential** このフィールドを使用して、コンテンツ検証を有効にします。プロジェクトの同期中にコンテンツの署名を検証するために使用する GPG キーを指定します。コンテンツが改ざんされている場合、ジョブは実行されません。詳細は、プロジェクトの署名と検証 を参照してください。
4. **Save** をクリックします。

関連情報

以下に、プロジェクトの調達方法を説明します。

- Playbook の手動管理
- ソースコントロールの使用による Playbook の管理
  - SCM タイプ - Git および Subversion
  - SCM タイプ - Red Hat Insights
  - SCM タイプ - リモートアーカイブ

16.1.1. Playbook の手動管理

手順

- Playbook を保管する 1 つ以上のディレクトリーをプロジェクトのベースパス (例: **/var/lib/awx/projects/**) に作成します。
- Playbook ファイルを作成し、これを Playbook ディレクトリーにコピーします。
- Playbook ディレクトリーおよびファイルが、サービスを実行するのと同じ UNIX ユーザーおよびグループで所有されていることを確認します。
- パーミッションが Playbook ディレクトリーおよびファイルについて適切であることを確認します。

トラブルシューティング

- ベースプロジェクトパスに Ansible Playbook ディレクトリーを追加していない場合は、エラーメッセージが表示されます。次のいずれかの方法を選択し、このエラーをトラブルシューティングしてください。
  - 適切な Playbook ディレクトリーを作成し、SCM (スペルを入力してください*) から Playbook をチェックアウトします。
  - Playbook を適切な Playbook ディレクトリーにコピーします。

16.1.2. ソースコントロールを使用した Playbook の管理

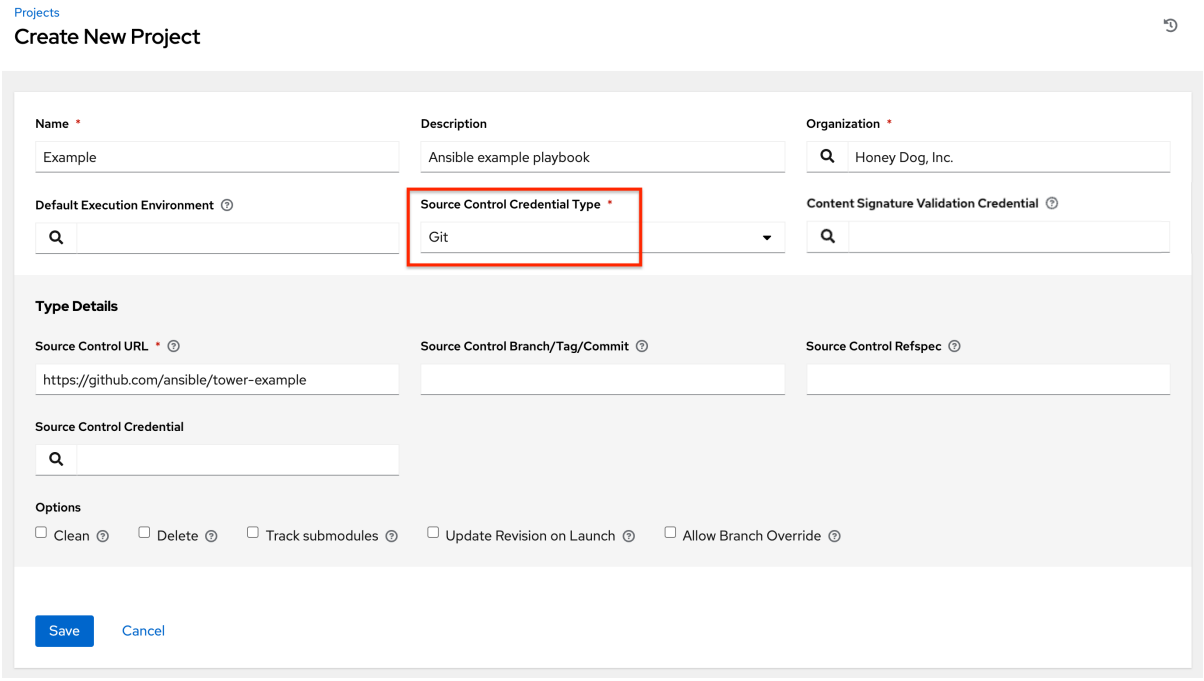
ソースコントロールを使用して Playbook を管理する場合は、次のいずれかの方法を選択します。

- SCM タイプ - Git と Subversion を使用するための Playbook の設定
- SCM タイプ - Red Hat Insights を使用するための Playbook の設定
- SCM タイプ - リモートアーカイブを使用するための Playbook の設定

16.1.2.1. SCM タイプ - Git と Subversion を使用するための Playbook の設定

手順

1. プロジェクトの **Details** タブで、**SCM Type** メニューから適切なオプション (Git または Subversion) を選択します。



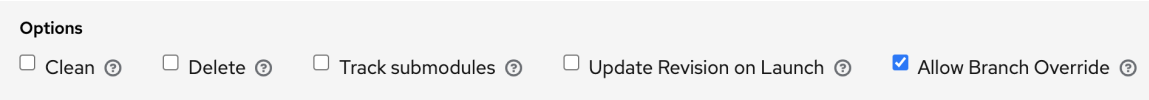
2. 以下のフィールドに該当する詳細を入力します。

   - **SCM URL** - ツールヒントの例を参照してください。
   - オプション: **SCM Branch/Tag/Commit**: チェックアウトするソースコントロール (Git または Subversion) からの SCM ブランチ、タグ、コミットハッシュ、任意の参照、またはリビジョン番号 (該当する場合) を入力します。次のフィールドにカスタム refspec も指定しない限り、一部のコミットハッシュと参照は使用できない場合があります。空白のままにした場合、デフォルトは **HEAD** です。これは、このプロジェクトで最後にチェックアウトされたブランチ、タグ、またはコミットのことです。
   - **SCM Refspec** - このフィールドは git ソースコントロール専用のオプションであり、git の知識があり、問題なく使用できる上級ユーザーのみが、リモートリポジトリーからダウンロードする参照を指定する必要があります。詳細は、ジョブブランチのオーバーライド を参照してください。
   - **Source Control Credential** - 認証が必要な場合は、適切なソースコントロール認証情報を選択します。

3. オプション: **SCM Update Options** - 該当する場合、起動動作を選択します。

   - **Clean** - 更新を実行する前にローカルの変更を削除します。
   - **削除** - 更新を実行する前に、ローカルリポジトリー全体を削除します。リポジトリーのサイズにより、更新の完了までに必要な時間が大幅に長くなる可能性があります。
   - **SCM Update Options** - 最新のコミットを追跡します。ツールチップに、詳細情報があります。
   - **Update Revision on Launch** - プロジェクトのリビジョンをリモートソースコントロールの現在のリビジョンに更新し、Galaxy または コレクションのサポート のロールディレクトリーをキャッシュします。Automation controller は、ローカルリビジョンが一致し、ロールとコレクションが最終更新で最新であることを確認します。さらに、プロジェクトが同期できるよりも早くジョブが生成された場合にジョブのオーバーフローを回避するために、これを選択すると、以前のプロジェクトの同期を指定した秒数キャッシュするようにキャッシュタイムアウトを設定できます。

- **Allow Branch Override** - このプロジェクトを使用するジョブテンプレートまたはインベントリーソースが、プロジェクト以外の指定された SCM ブランチまたはリビジョンで起動できるようにします。詳細は、ジョブブランチのオーバーライド を参照してください。



4. **Save** をクリックしてプロジェクトを保存します。

ヒント

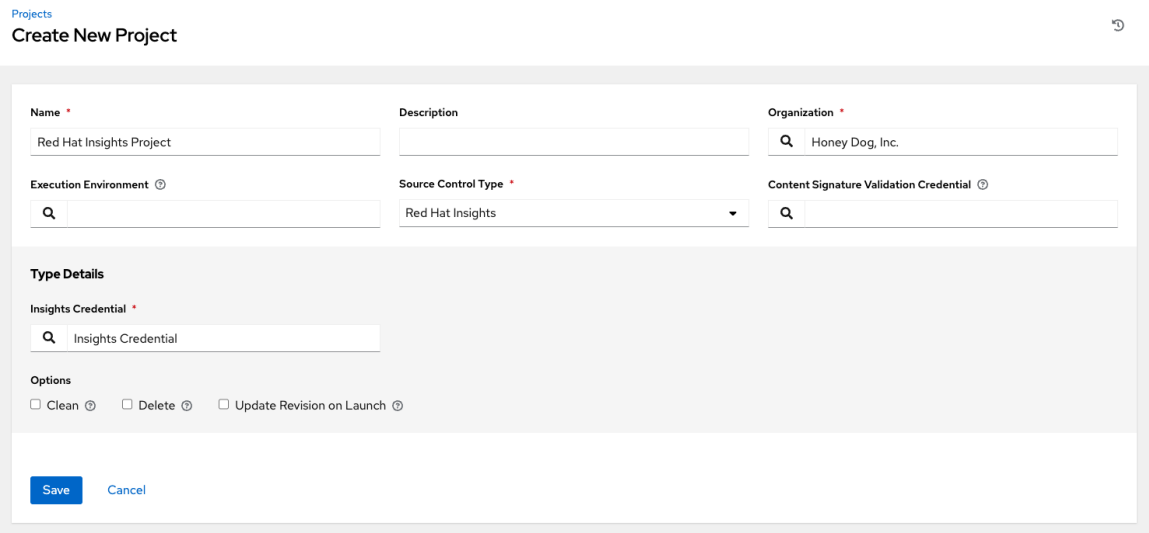
GitHub リンクを使用すると、Playbook を簡単に使用できます。開始するには、ここ で入手可能な **helloworld.yml** ファイルを使用してください。

このリンクから、Automation Controller ユーザーガイド の手順に従って手動で作成した Playbook と非常によく似た Playbook を入手できます。これを使用してもシステムが変更されたり、問題が発生することはありません。

16.1.2.2. SCM タイプ - Red Hat Insights を使用するための Playbook の設定

手順

1. プロジェクトの **Details** ページで、**SCM Type** メニューから **Red Hat Insights** を選択します。
2. Red Hat Insights では、認証に認証情報が必要です。Red Hat Insights で使用できるように、**Credential** フィールドから適切な認証情報を選択します。
3. オプション: **SCM Update Options** フィールドで、該当する場合、起動動作を選択します。
   - **Clean** - 更新を実行する前にローカルの変更を削除します。
   - **削除** - 更新を実行する前に、ローカルリポジトリー全体を削除します。リポジトリーのサイズにより、更新の完了までに必要な時間が大幅に長くなる可能性があります。
   - **Update Revision on Launch**- プロジェクトのリビジョンをリモートソースコントロールの現在のリビジョンに更新し、Ansible Galaxy サポート または コレクションサポート からロールディレクトリーをキャッシュします。Automation controller は、ローカルリビジョンが一致し、ロールとコレクションが最新であることを保証します。さらに、プロジェクトが同期できるよりも早くジョブが生成された場合に、これを選択すると、以前のプロジェクトの同期を指定した秒数キャッシュするようにキャッシュタイムアウトを設定できます。



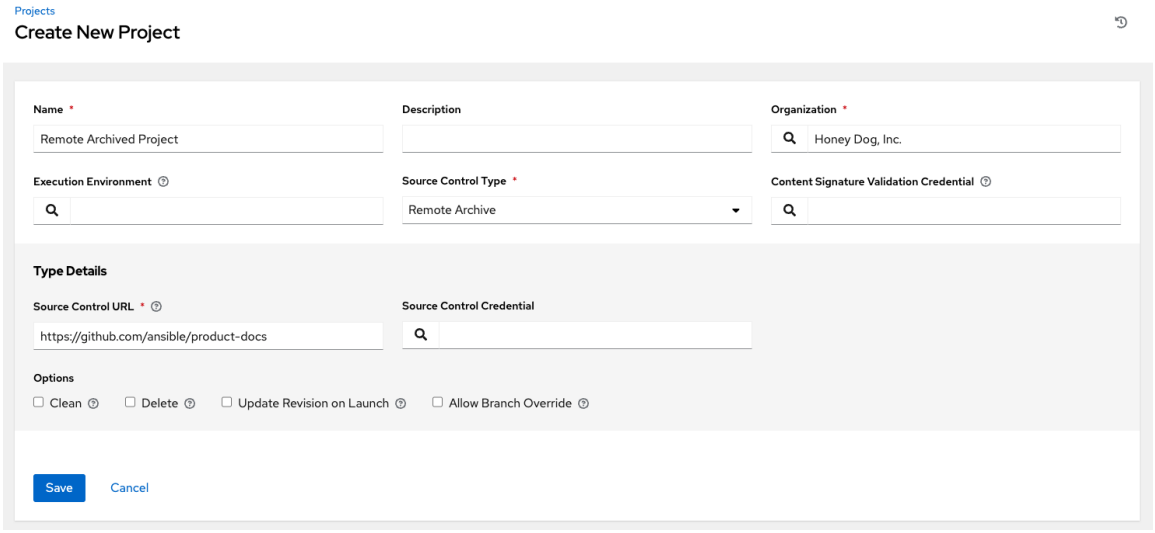
4. **Save** をクリックします。

16.1.2.3. SCM タイプ - リモートアーカイブを使用するための Playbook の設定

リモートアーカイブを使用する Playbook により、バージョン付けされたアーティファクトを生成するビルドプロセスまたはリリースに基づいてプロジェクトを提供することができます。これには、単一のアーカイブ内のそのプロジェクトのすべての要件が含まれます。

手順

1. プロジェクトの **Details** ページで、**SCM Type** メニューから **Remote Archive** を選択します。
2. 以下のフィールドに該当する詳細を入力します。
   - **SCM URL** - GitHub Release、Artifactory に保存されているビルドアーティファクトなど、リモートアーカイブへの URL を要求し、それをプロジェクトパスにデプロイメントして使用します。
   - **SCM Credential** - 認証が必要な場合には、適切な SCM 認証情報を選択します。
3. オプション: **SCM Update Options** フィールドで、該当する場合は起動動作を選択します。
   - **Clean** - 更新を実行する前にローカルの変更を削除します。
   - **削除** - 更新を実行する前に、ローカルリポジトリー全体を削除します。リポジトリーのサイズにより、更新の完了までに必要な時間が大幅に長くなる可能性があります。
   - **Update Revision on Launch** - 推奨されません。このオプションは、プロジェクトのリビジョンをリモートソースコントロールの現在のリビジョンに更新し、Ansible Galaxy サポート または コレクションサポート からのロールディレクトリーをキャッシュします。
   - **Allow Branch Override** - 推奨されません。このオプションを使用すると、このプロジェクトを使用するジョブテンプレートが、指定された SCM ブランチまたはプロジェクト以外のリビジョンで起動できるようになります。



注記

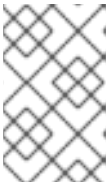
この SCM タイプは、アーティファクトの変更のない概念をサポートすることが意図されているため、(少なくてもロールに対して) Galaxy 統合を無効にすることが推奨されます。

4. **Save** をクリックします。

16.2. ソースコントロールからのプロジェクトの更新

手順

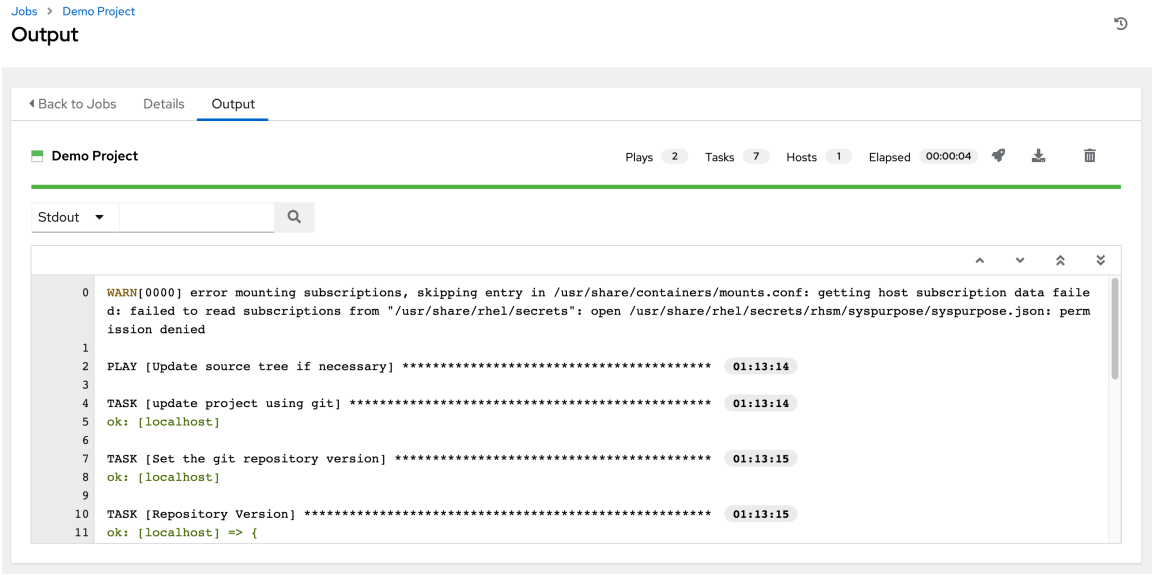
1. ナビゲーションパネルから、**Resources** → **Projects** を選択します。
2. 更新するプロジェクトの横にある同期アイコン をクリックします。



注記

ソースコントロールを使用するためのプロジェクト設定を追加すると、設定したソースコントロールからプロジェクトの詳細を取得する同期がすぐに開始します。

- 更新プロセスの詳細は、**Status** 列の下にあるプロジェクトのステータスをクリックしてください。**Jobs** セクションの **Output** タブが表示されます。



16.3. 権限の使用

プロジェクトに割り当てられており、プロジェクト、インベントリー、ジョブテンプレート、およびその他の要素の読み取り、変更、および管理を行う機能を提供する権限のセット (ロールベースのアクセス制御) は、特権とも呼ばれます。

プロジェクトの権限にアクセスするには、**Projects** ページの **Access** タブを選択します。この画面には、現在このプロジェクトに対する権限を持っているユーザーのリストが表示されます。

このリストは、**Username**、**First Name**、または **Last Name** で並べ替えおよび検索できます。

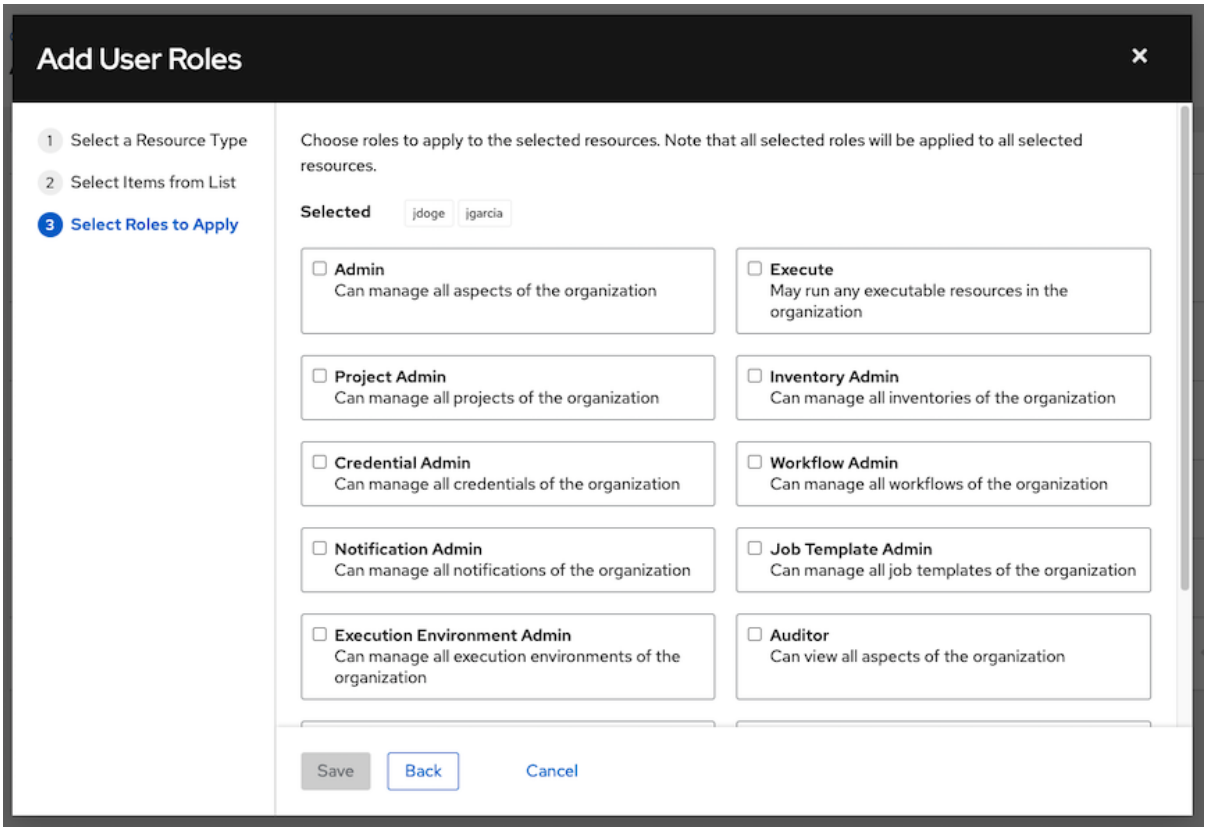
16.3.1. プロジェクト権限の追加

ユーザーとチームがプロジェクトにアクセスするために必要な権限を管理します。

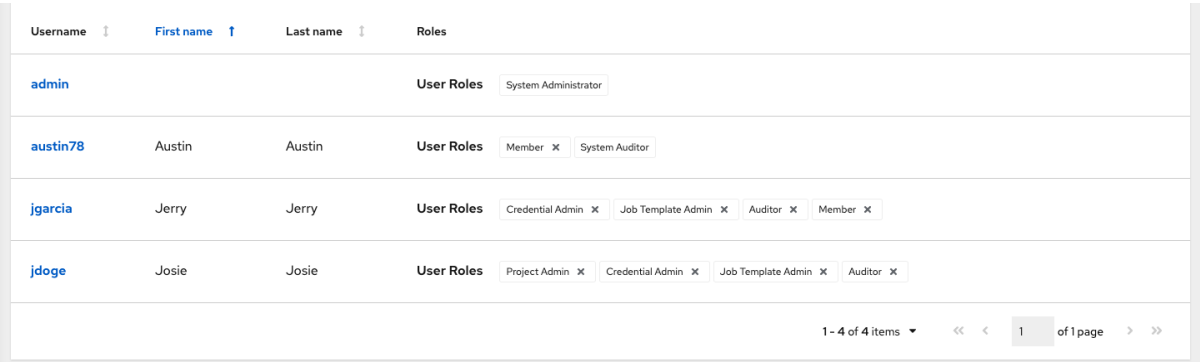
手順

1. ナビゲーションパネルから、**Resources** → **Projects** ページを選択します。

2. 更新するプロジェクトを選択し、Access タブをクリックします。
3. **Add** をクリックします。
4. 追加するユーザーまたはチームを選択し、**Next** をクリックします。
5. 名前の横にあるチェックボックスをクリックしてリストからユーザーまたはチームを 1 つ以上選択し、メンバーとして追加します。
6. **Next** をクリックします。
7. 選択したユーザーまたはチームに付与するロールを選択します。ロールの完全なリストを確認するには、必ず下にスクロールしてください。リソースが異なれば、利用可能なオプションも異なります。



8. **Save** をクリックして、選択したユーザーまたはチームにロールを適用し、メンバーとして追加します。各ユーザーおよびチームに割り当てられた更新済みのロールが表示されます。



16.3.2. プロジェクトからの権限の削除

特定のユーザーのロールを削除します。

手順

1. ナビゲーションパネルから、**Resources → Projects** を選択します。
2. 更新するプロジェクトを選択し、**Access** タブをクリックします。
3. **Roles** 列のユーザーロールの横にある ✖ アイコンをクリックします。
4. 確認ウィンドウで **Delete** をクリックして、関連付けの解除を確認します。

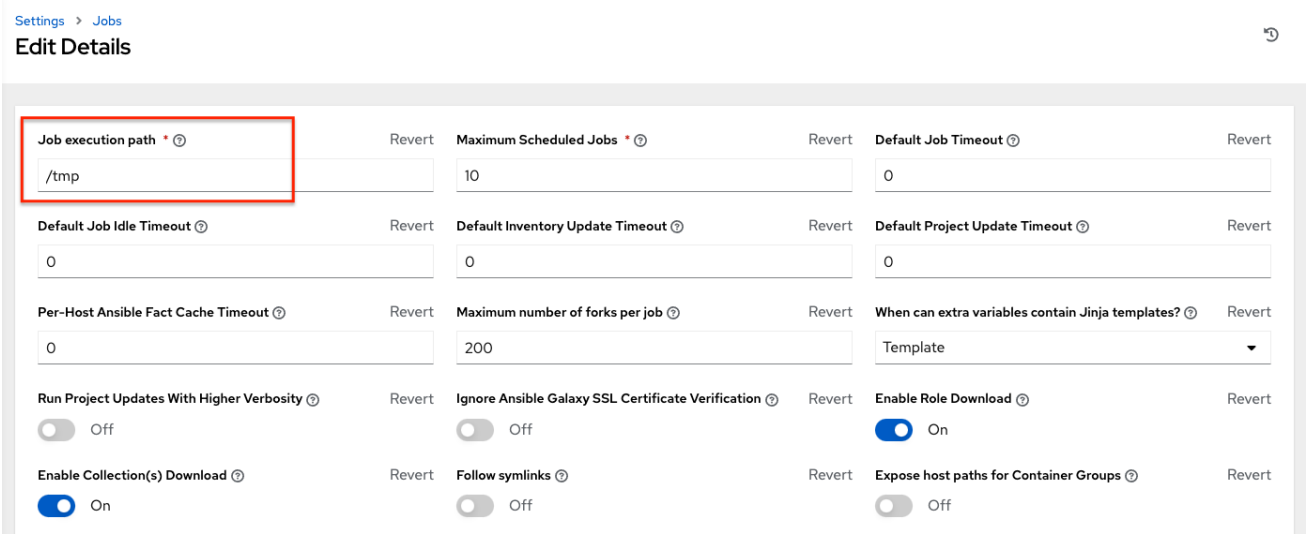
16.4. ANSIBLE GALAXY サポート

Automation Controller は、プロジェクトの更新の最後に、**<project-top-level-directory>/roles/requirements.yml** にある **roles** ディレクトリー内の **requirements.yml** ファイルを検索します。

このファイルが見つかると、次のコマンドが自動的に実行されます。

```
ansible-galaxy role install -r roles/requirements.yml -p <project-specific cache location>/requirements_roles -vvv
```

このファイルを使用すると、Ansible Galaxy ロール、または独自のプロジェクトと組み合わせてチェックアウトできる他のリポジトリー内のロールを参照できるようになります。Ansible Galaxy アクセスの追加により、この結果を得るために git サブモジュールを作成する必要がなくなります。SCM プロジェクトがロールまたはコレクションとともにプライベートジョブ環境に取り込まれ、そこから実行されることを考慮すると、デフォルトで **/tmp** 内にプロジェクトに固有の **<private job directory>** が作成されます。ただし、**Settings** ウィンドウの **Jobs Settings** タブでお使いの環境をもとに、別の **Job Execution Path** を指定できます。



キャッシュディレクトリーは、グローバルプロジェクトフォルダー内のサブディレクトリーです。コンテンツはキャッシュの場所から **<job private directory>/requirements_roles** にコピーできます。

デフォルトでは、Automation controller には、SCM プロジェクトの **roles/requirements.yml** ファイルからロールを動的にダウンロードできるようにするシステム全体の設定があります。**Settings** メニューの **Jobs settings** 画面で、**Enable Role Download** トグルボタンを **Off** に切り替えることで、この設定をオフにできます。

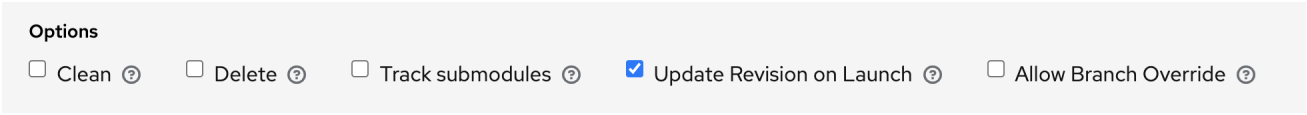
プロジェクトの同期が実行されるたびに、Automation controller は、プロジェクトソースおよび Galaxy または Collections のロールがプロジェクトに対して古くなっているかどうかを判断します。プロジェクトを更新すると、更新に含まれるロールがダウンロードされます。

アップストリームのロールに加えられた変更をジョブで取得する必要がある場合は、プロジェクトを更新すると取得されます。ロールの変更は、新しいコミットが **provision-role** のソースコントロールにプッシュされたことを意味します。

この変更をジョブで有効にするために、**Playbook** リポジトリーに新しいコミットをプッシュする必要はありません。プロジェクトの更新は行う必要があります。更新すると、ロールがローカルキャッシュにダウンロードされます。

たとえば、ソースコントロールに 2 つの git リポジトリーがあるとします。1 つ目は **playbooks** で、Automation Controller のプロジェクトがこの URL を参照します。2 つ目は **provision-role** で、**playbooks** git リポジトリー内の **roles/requirements.yml** ファイルによって参照されます。

ジョブは、すべてのジョブの実行前に最新のロールをダウンロードします。パフォーマンス上の理由から、ロールとコレクションはローカルにキャッシュされます。各ジョブの実行前にアップストリームロールを再ダウンロードするには、プロジェクトの **SCM Update Options** で **Update Revision on Launch** を選択する必要があります。

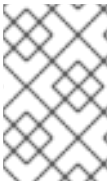


更新は同期よりもはるかに前のプロセスで行われるため、これによりエラーと詳細がより速く、より論理的な場所で詳細が表示されます。

requirements.yml ファイルの構文の詳細と例については、Ansible ドキュメントの ロール要件セクション を参照してください。

特に公開する必要があるディレクトリーがある場合は、**Settings** 画面の **Jobs** セクションの **Paths to Expose to Isolated Jobs** で指定できます。設定ファイル内の次のエントリーを更新することもできます。

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

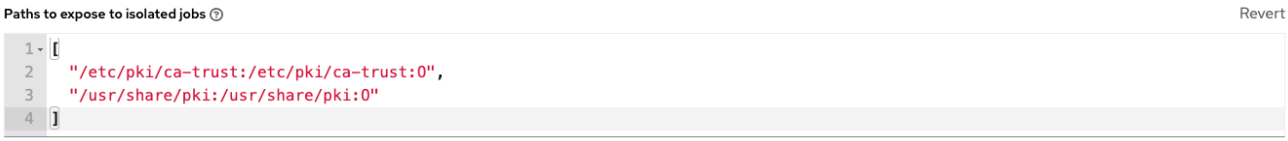


注記

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

設定ファイルに変更を加えた場合には、変更の保存後に **automation-controller-service restart** コマンドを使用してサービスを再起動するようにしてください。

UI では、**Jobs settings** ウィンドウでこれらの設定を設定できます。

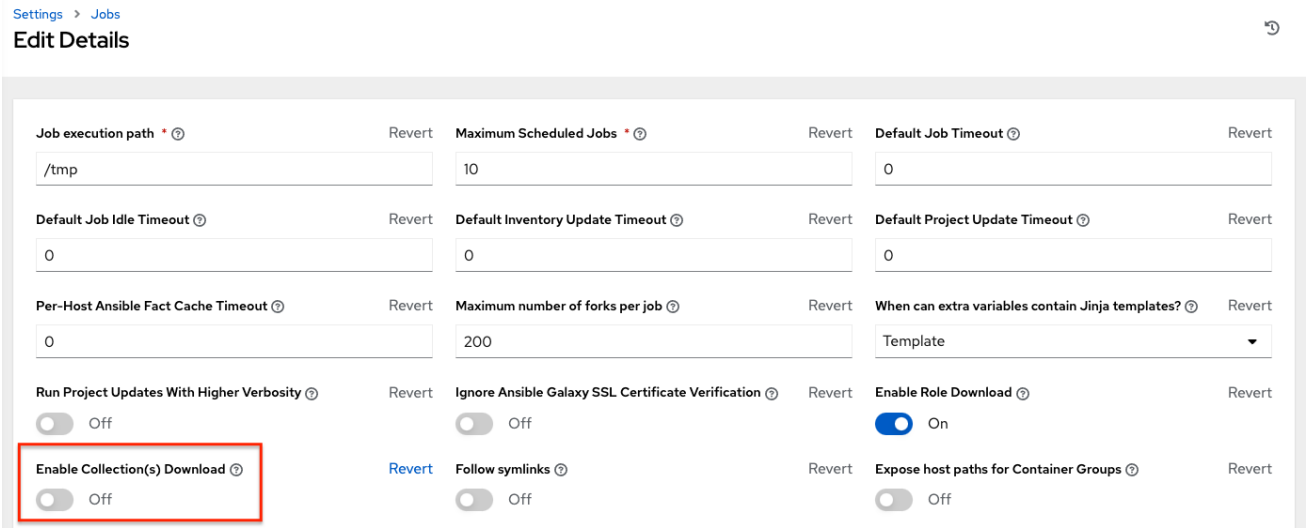


16.5. コレクションのサポート

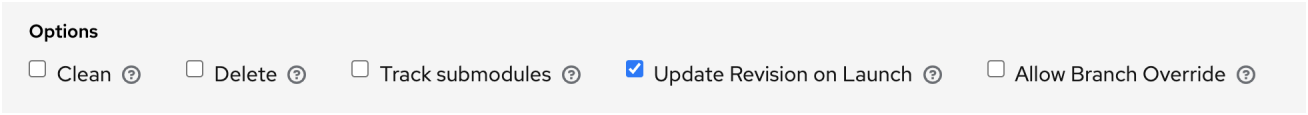
Automation controller は、ジョブ実行でプロジェクト固有の Ansible コレクション をサポートします。**collections/requirements.yml** で SCM のコレクション要件ファイルを指定すると、Automation Controller はジョブの実行前にプロジェクトを暗黙的に同期するときに、そのファイルにコレクション

をインストールします。

Automation controller には、SCM プロジェクトの **collections/requirements.yml** ファイルからコレクションを動的にダウンロードできるようにするシステム全体の設定があります。**Settings** メニューの **Jobs settings** タブで、**Enable Collections Download** トグルボタンを **Off** に切り替えることで、この設定をオフにできます。



ロールとコレクションはパフォーマンス上の理由からローカルにキャッシュされるため、以下を確実に行うためには、プロジェクトの SCM 更新オプションで **Update Revision on Launch** を選択する必要があります。



16.5.1. Automation Hub でのコレクションの使用

Automation Controller でコレクションコンテンツのデフォルトソースとして Automation Hub を使用するには、Automation Hub UI で API トークンを作成する必要があります。その後、このトークンを Automation Controller で指定します。

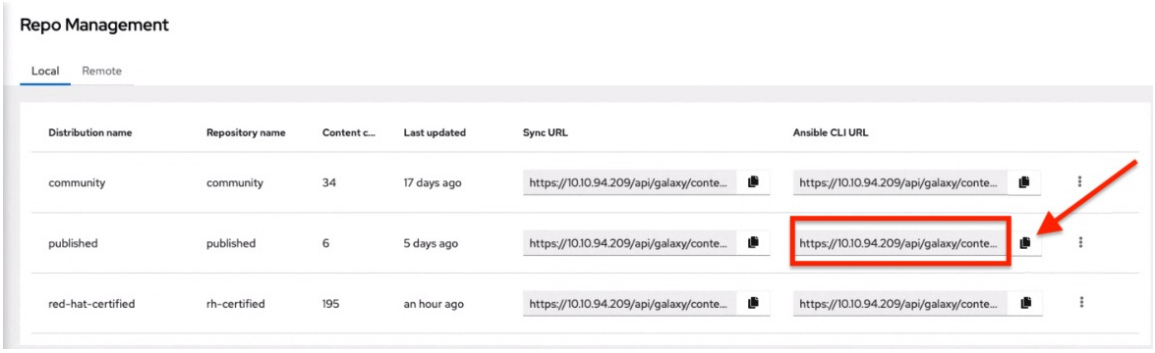
Private Automation Hub または Automation Hub に接続するには、次の手順を使用します。指定する URL のみが異なります。

手順

1. https://console.redhat.com/ansible/automation-hub/token にアクセスします。
2. **Load token** をクリックします。
3. コピー アイコンをクリックして、API トークンをクリップボードにコピーします。
4. 次のいずれかの方法を選択して認証情報を作成します。
    a. Automation Hub を使用するには、コピーしたトークンを使用し、トークンページの **Server URL** および **SSO URL** フィールドに表示されている URL を指す Automation Hub 認証情報を作成します。
        - **Galaxy Server URL** = **https://console.redhat.com/api/automation-hub/**

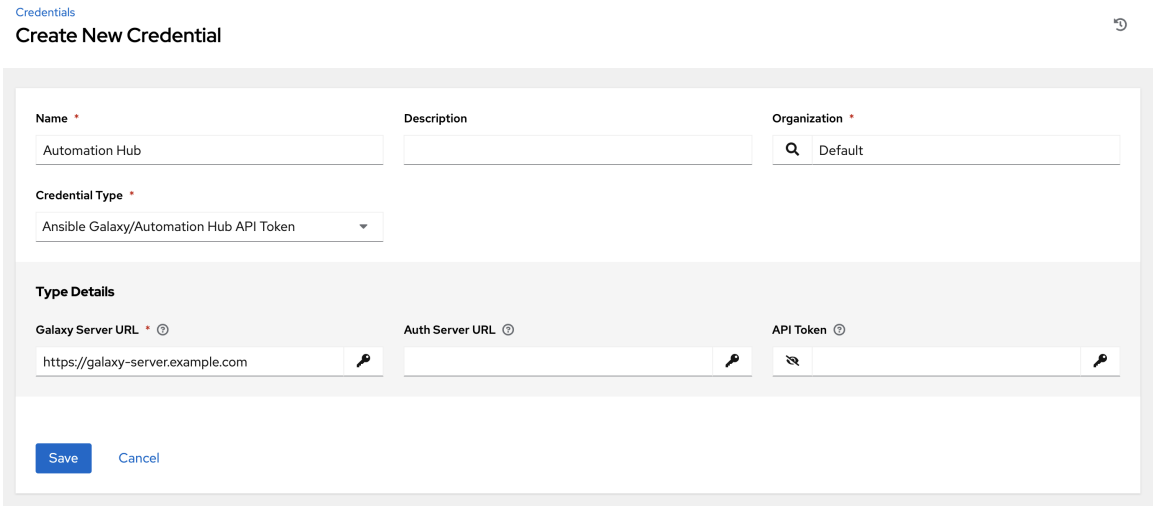
- AUTH SEVER URL = https://sso.redhat.com/auth/realms/redhat-external/protocol/openid-connect/token

b. Private Automation Hub を使用するには、Private Automation Hub の **Repo Management** ダッシュボードから取得したトークンを使用し、次に示すように公開されたリポジトリー URL を指す Automation Hub 認証情報を作成します。



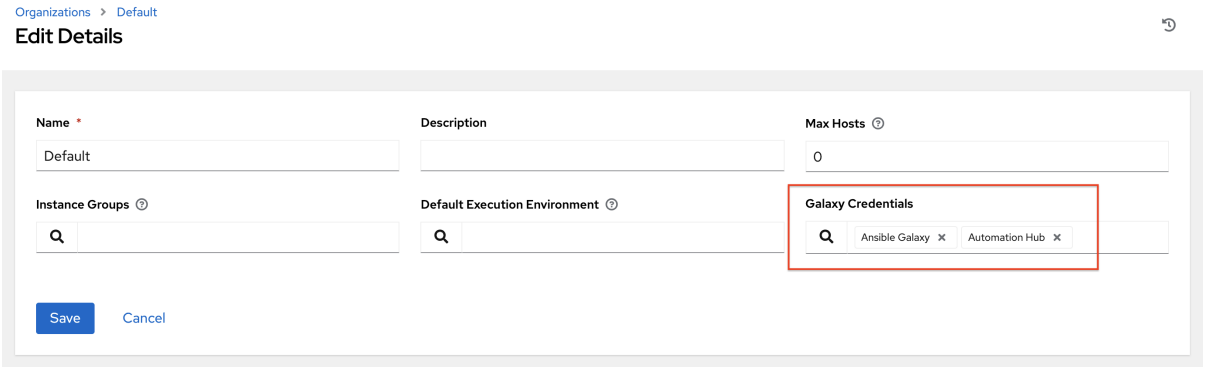
さまざまな名前空間またはコレクションを含む各種リポジトリーを作成できます。Automation Hub のリポジトリーごとに、異なる認証情報を作成する必要があります。

UI からの **/https://$<hub_url>/api/galaxy/content/<repo you want to pull from>** の形式の **Ansible CLI URL** を **Create Credential** フォームの **Galaxy Server URL** フィールドにコピーします。



UI 固有の手順については、Automation Hub の Red Hat 認定、検証済み、Ansible Galaxy コンテンツ を参照してください。

5. コンテンツを同期する組織に移動し、新しい認証情報を組織に追加します。これにより、コンテンツを使用する認証情報またはリポジトリーに各組織を関連付けることができます。



例

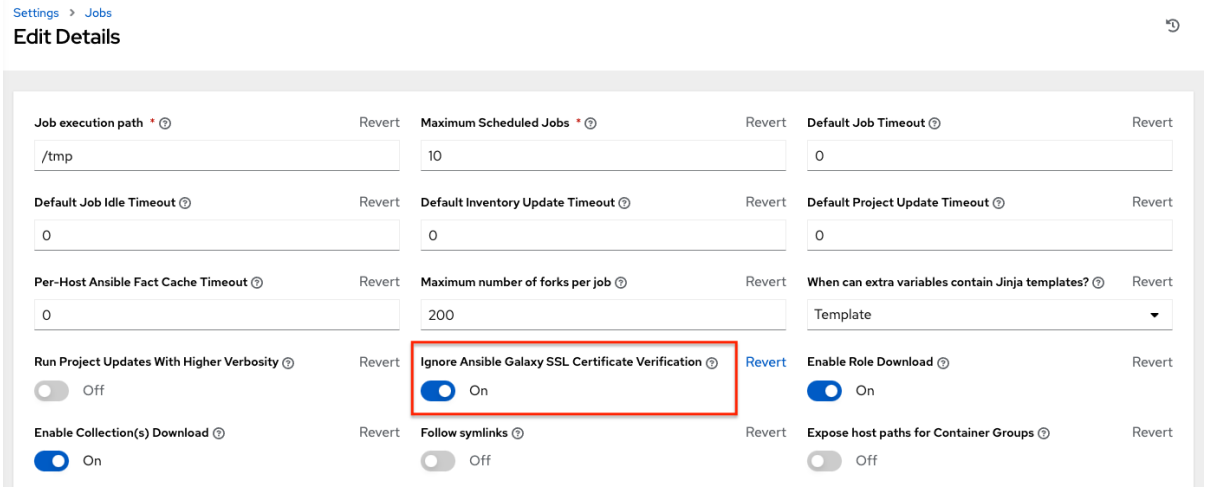
次の 2 つのリポジトリーがあるとします。

- Prod: **Namespace 1** と **Namespace 2** があり、それぞれにコレクション **A** と **B** (**namespace1.collectionA:v2.0.0** と **namespace2.collectionB:v2.0.0**) があります。
- Stage: **Namespace 1** があり、そこにコレクション **A** (**namespace1.collectionA:v1.5.0**) だけがあります。**Prod** と **Stage** のリポジトリー URL があります。
  それぞれに対して認証情報を作成できます。

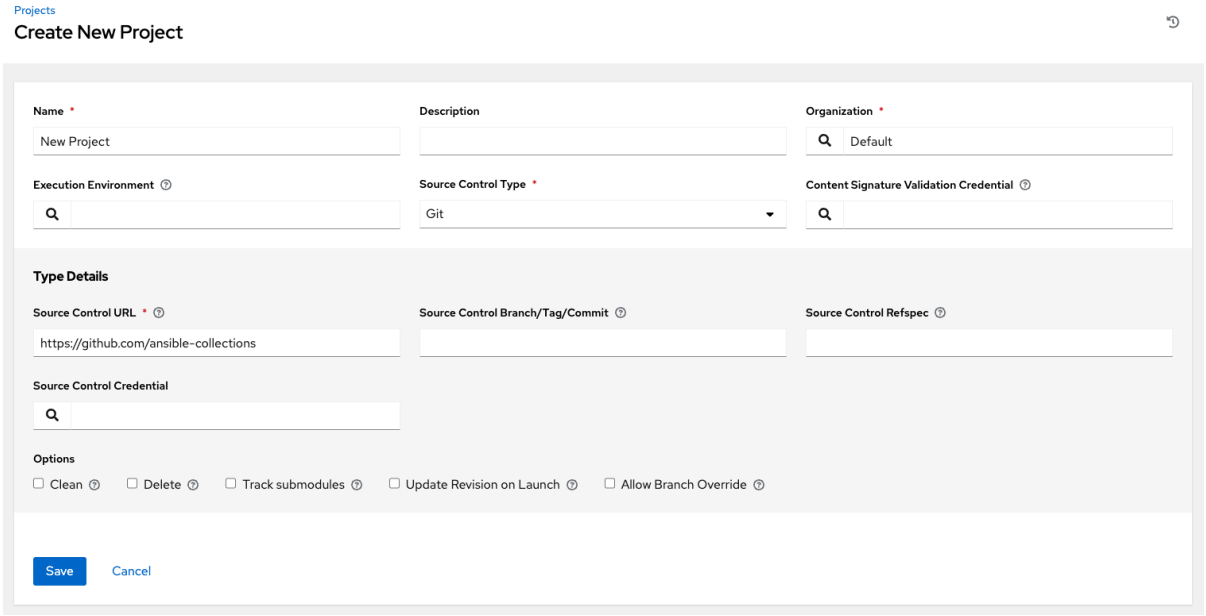
  その後、さまざまな組織に各種レベルのアクセスを割り当てることができます。たとえば、**Developers** 組織を作成して両方のリポジトリーへのアクセス権を付与し、Operations 組織には **Prod** リポジトリーへのアクセス権だけを付与することができます。

  UI 固有の手順については、 Private Automation Hub でのコンテナーリポジトリーのユーザーアクセスの設定 を参照してください。

6. Automation Hub に自己署名証明書がある場合は、トグルを使用して **Ansible Galaxy SSL 証明書検証を無視する** 設定を有効にします。署名付き証明書を使用する Automation Hub の場合は、代わりにトグルを使用して無効にします。これはグローバル設定です。



7. ソースリポジトリーが **collections/requirements.yml** ファイルにある要件ファイルで必要なコレクションを指定するプロジェクトを作成します。使用する構文の詳細は、Ansible ドキュメントの Using Ansible collections を参照してください。



8. Projects リストビューで、同期アイコン をクリックしてこのプロジェクトを更新します。Automation controller は、**collections/requirements.yml** ファイルから Galaxy コレクションをフェッチし、変更されたものとして報告します。このプロジェクトを使用するすべてのジョブテンプレートにコレクションがインストールされます。

注記

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、/tmp ファイルの領域がはるかに多く消費されます。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。

関連情報

コレクションの詳細は、Using Collections を参照してください。

インストールを直接自動化するのに使用できるこれらの公式コレクションの 1 つを Red Hat がどのように公開しているかについては、AWX Ansible Collection のドキュメントを参照してください。

第17章 プロジェクトの署名と検証

プロジェクトの署名と検証を使用すると、プロジェクトディレクトリー内のファイルに署名し、そのコンテンツが何らかの方法で変更されていないか、またはファイルがプロジェクトに予期せず追加または削除されていないかどうかを検証できます。これを行うには、署名用の秘密鍵と、検証用の対応する公開鍵が必要です。

プロジェクトのメンテナーがコンテンツの署名を行い場合でサポートされている手法は、**ansible-sign** ユーティリティーを使用して、付属の **コマンドラインインターフェイス** (CLI) を使用することです。

この CLI の目的は、**GNU Privacy Guard** (GPG) などの暗号化テクノロジーを簡単に使用して、プロジェクト内のファイルがいかなる方法でも改ざんされていないと検証することです。現在、サポートされている署名と検証の手段は GPG のみです。

Automation controller は、署名されたコンテンツを検証するために使用されます。一致する公開鍵が署名されたプロジェクトに関連付けられた後、Automation controller は、署名中に含まれたファイルが変更されていないこと、およびファイルが予期せず追加または削除されていないことを検証します。署名が無効であるかファイルが変更されている場合、プロジェクトは更新に失敗し、プロジェクトを使用するジョブは起動されません。プロジェクトの検証ステータスにより、安全で改ざんされていないコンテンツのみがジョブで実行できることが保証されます。

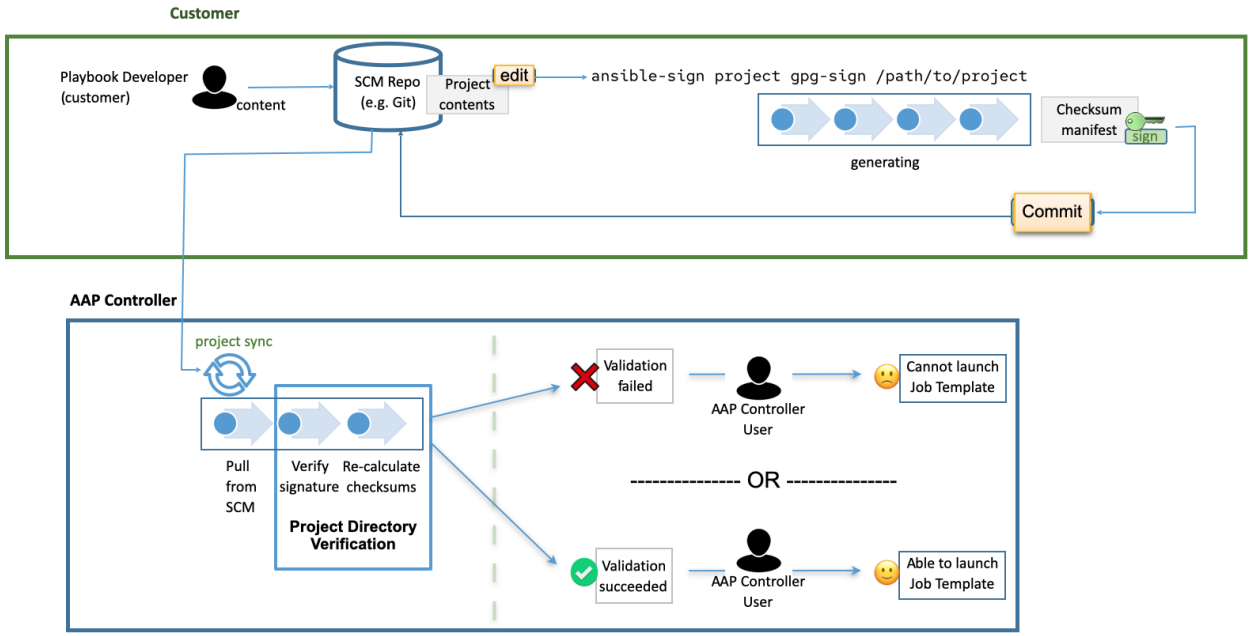
リポジトリーが署名と検証用にすでに設定されている場合、プロジェクト変更用の通常のワークフローは次のようになります。

1. プロジェクトリポジトリーがすでに設定されており、ファイルに変更を加えるとします。
2. 変更を加えて、次のコマンドを実行します。

   ```
   ansible-sign project gpg-sign /path/to/project
   ```

   このコマンドは、チェックサムマニフェストを更新し、署名します。
3. 変更、更新されたチェックサムマニフェスト、および署名をリポジトリーにコミットします。
4. プロジェクトを同期すると、Automation controller は新しい変更を取り込み、Automation controller 内のプロジェクトに関連付けられた公開鍵が、チェックサムマニフェストの署名に使用された秘密鍵と一致することを確認します (これにより、チェックサムマニフェスト自体の改ざんが防止されます)。マニフェスト内の各ファイルのチェックサムを計算し、チェックサムが一致していること (つまり、ファイルが変更されていないこと) を確認します。また、すべてのファイルが考慮されていることも保証されます。

ファイルは **MANIFEST.in** ファイルに含めるか、MANIFEST.in ファイルから除外する必要があります。このファイルの詳細は、プロジェクトの署名 を参照してください。ファイルが予期せず追加または削除された場合、検証は失敗します。



17.1. 前提条件

- RHEL ノードを適切にサブスクライブしている。
  - **baseos** および **appstream** リポジトリーを持つ RHEL サブスクリプションが有効になっている。
  - Red Hat Ansible Automation Platform サブスクリプションと適切なチャネルが有効になっている。

    ansible-automation-platform-2.4-for-rhel-8-x86_64-rpms for RHEL 8
    ansible-automation-platform-2.4-for-rhel-9-x86_64-rpms for RHEL 9

- コンテンツ署名用に有効な GPG 公開鍵または秘密鍵のペアがある。詳細は、GPG キーペアの作成方法 を参照してください。
  GPG キーの詳細は、GnuPG のドキュメント を参照してください。

  次のコマンドを使用して、デフォルトの GnuPG キーリングに有効な GPG キーペアがあることを確認します。

  ```
  gpg --list-secret-keys
  ```

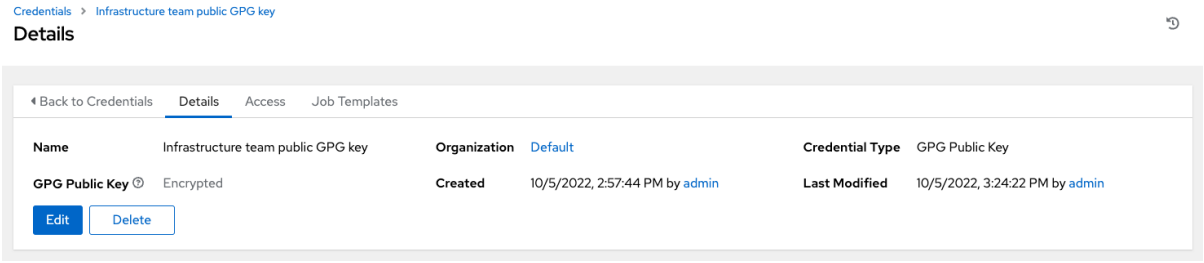
  このコマンドで何も出力されないか、 **trustdb was created** という 1 行の出力が生成される場合は、デフォルトのキーリングに秘密鍵がありません。この場合、続行する前に、How to create GPG keypairs を参照して、新しいキーペアを作成する方法を確認してください。他の出力が生成される場合は、有効な秘密鍵があり、**ansible-sign** を使用する準備ができています。

17.2. AUTOMATION CONTROLLER への GPG キーの追加

Automation controller でのコンテンツの署名と検証に GPG キーを使用するには、CLI で次のコマンドを実行して GPG キーを追加します。

```
$ gpg --list-keys
$ gpg --export --armour <key fingerprint> > my_public_key.asc
```

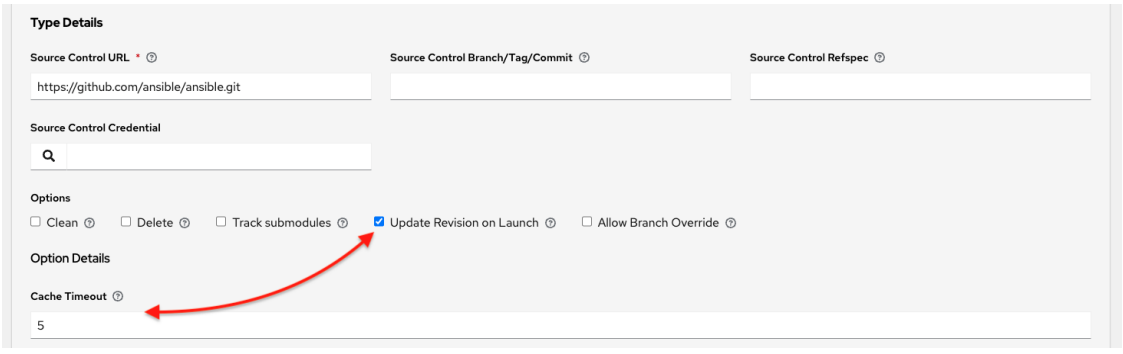
1. ナビゲーションパネルから **Credentials** を選択します。
2. **Add** をクリックします。
3. 新しい認証情報にわかりやすい名前を付けます (例: Infrastructure team public GPG key など)。
4. **Credential Type** フィールドで、**GPG Public Key** を選択します。
5. **Browse** をクリックして、公開鍵ファイル (**my_public_key.asc** など) を見つけて選択します。
6. **Save** をクリックします。



この認証情報は **projects <ug_projects_add>** で選択できるようになり、今後のプロジェクトの同期時にコンテンツの検証が自動的に行われます。

注記

プロジェクトキャッシュ SCM タイムアウトを使用して、Automation controller が署名されたコンテンツを再検証する頻度を制御します。プロジェクトが (そのプロジェクトを使用するように設定されたジョブテンプレートの) 起動時に更新されるように設定されている場合は、キャッシュタイムアウト設定を有効にして、最後の更新から **N** 秒経過後に更新するように設定できます。検証の実行頻度が高すぎる場合は、プロジェクトの **Option Details** ペインの **Cache Timeout** フィールドで時間を指定して、プロジェクトの更新頻度を遅くすることができます。



17.3. ANSIBLE-SIGN CLI ユーティリティーのインストール

ansible-sign ユーティリティーを使用して、ユーザーが署名し、プロジェクトが署名されているかどうかを確認するためのオプションを提供します。

手順

1. 次のコマンドを実行して **ansible-sign** をインストールします。

```
$ dnf install ansible-sign
```

2. 次のコマンドを使用して、**ansible-sign** が正常にインストールされたことを確認します。

```
$ ansible-sign --version
```

次のような出力は **ansible-sign** が正常にインストールされたことを示します。

```
ansible-sign 0.1
```

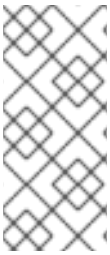
17.4. プロジェクトの署名

プロジェクトの署名には、Ansible プロジェクトディレクトリーが関係します。プロジェクトのディレクトリー構造の詳細は、Ansible ドキュメントの Sample Ansible setup を参照してください。

次のサンプルプロジェクトは、Playbook ディレクトリーの下にインベントリーファイルと、小規模な playbook が 2 つ含まれる非常に単純な構造となっています。

```
$ cd sample-project/
$ tree -a .
.
├── inventory
└── playbooks
    └── get_uptime.yml
    └── hello.yml

    1 directory, 3 files
```



注記

使用されるコマンドは、作業ディレクトリーがプロジェクトのルートであることを前提としています。**ansible-sign project** コマンドは、プロジェクトのルートディレクトリーを最後の引数として受け取ります。

. を使用して現在の作業ディレクトリーを示します。

ansible-sign は、プロジェクトに含まれるセキュリティーで保護された全ファイルのチェックサム (SHA256) を取得して、チェックサムマニフェストファイルにコンパイルし、そのマニフェストファイルに署名することでコンテンツを改ざんから保護します。

コンテンツに署名するには、保護するファイルを **ansible-sign** に指示する **MANIFEST.in** ファイルをプロジェクトのルートディレクトリーに作成します。

内部的には、**ansible-sign** は、Python の distlib ライブラリーの **distlib.manifest** モジュールを使用するため、**MANIFEST.in** はこのライブラリーが指定する構文に従う必要があります。MANIFEST.in ファイルディレクティブの説明は、Python Packaging User Guide を参照してください。

サンプルプロジェクトには 2 つのディレクティブが含まれており、最終的に次の **MANIFEST.in** ファイルになります。

```
include inventory
recursive-include playbooks *.yml
```

このファイルを配置したら、チェックサムマニフェストファイルを生成し、署名します。これらの手順は両方とも、**ansible-sign** コマンドで 1 つで実行できます。

```
$ ansible-sign project gpg-sign .
```

実行が成功すると、次のような出力が表示されます。

```
[OK   ] GPG signing successful!
[NOTE ] Checksum manifest: ./.ansible-sign/sha256sum.txt
[NOTE ] GPG summary: signature created
```

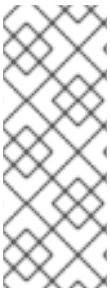
これでプロジェクトは署名されました。

gpg-sign サブコマンドは **project** サブコマンドの下にあることに注意してください。

プロジェクトのコンテンツに署名する場合、すべてのコマンドは **ansible-sign project** で始まります。

すべての **ansible-sign project** コマンドは、プロジェクトのルートディレクトリー **.** を最後の引数として受け取ります。

ansible-sign は、デフォルトのキーリングを使用し、最初に検出された利用可能な秘密鍵を探して、プロジェクトに署名します。**--fingerprint** オプションを使用して特定の秘密鍵を指定したり、**--gnupg-home** オプションを使用して完全に独立した GPG ホームディレクトリーを指定したりすることもできます。



注記

デスクトップ環境を使用している場合、GnuPG は秘密鍵のパスフレーズの入力を自動的に要求します。

この機能が動作しない場合、またはデスクトップ環境なしで作業している場合 (SSH など)、**gpg-sign** の後に **-p --prompt-passphrase** フラグを使用できます。これにより、代わりに **ansible-sign** がパスワードの入力を求めるようになります。

.ansible-sign ディレクトリーがプロジェクトディレクトリーに作成されたことに注意してください。このディレクトリーには、チェックサムマニフェストとそのマニフェストの分離された GPG 署名が含まれています。

```
$ tree -a .
.
├── .ansible-sign
│   ├── sha256sum.txt
│   └── sha256sum.txt.sig
├── inventory
├── MANIFEST.in
└── playbooks
    ├── get_uptime.yml
    └── hello.yml
```

17.5. プロジェクトの検証

署名された Ansible プロジェクトが変更されていないことを確認するには、**ansible-sign** を使用して、署名が有効かどうか、またファイルのチェックサムがチェックサムマニフェストに記載されている内容と一致しているかどうかを確認できます。**ansible-sign project gpg-verify** コマンドを使用すると、これらの条件の両方を自動的に検証できます。

```
$ ansible-sign project gpg-verify .
[OK   ] GPG signature verification succeeded.
[OK   ] Checksum validation succeeded.
```

注記

デフォルトでは、**ansible-sign** はデフォルトの GPG キーリングを使用して、一致する公開鍵を探します。**--keyring** オプションを使用してキーリングファイルを指定することも、**--gnugpg-home** オプションを使用して別の GPG ホームを指定することもできます。

何らかの理由で検証が失敗した場合は、原因のデバッグに役立つ情報が表示されます。使用したコマンドの **ansible-sign** の直後にグローバルの **--debug** フラグを渡すことで、追加の冗長性を有効にできます。

注記

GPG 認証情報がプロジェクトで使用されると、今後のプロジェクトの同期時にコンテンツの検証が自動的に行われます。

17.6. 署名の自動化

OpenShift や Jenkins など、信頼性の高い **継続的インテグレーション** (CI) 環境では、署名プロセスを自動化できます。

たとえば、GPG 秘密鍵を選択した CI プラットフォームにシークレットとして保存して CI 環境の GnuPG にインポートできます。その後、通常の CI 環境内で署名ワークフローを実行できます。

GPG を使用してプロジェクトに署名する場合、環境変数 **ANSIBLE_SIGN_GPG_PASSPHRASE** を署名鍵のパスフレーズに設定できます。これは、CI パイプラインで注入してマスクしたり、保護したりできます。

シナリオに応じて、**ansible-sign** は、署名中と検証中の両方で異なる終了コードを返します。CI 環境は障害に基づいて異なる動作を行うことができるため、CI と自動化のコンテキストでも役立ちます。たとえば、一部のエラーについてはアラートを送信できますが、他のエラーについては通知せずに失敗します。

以下は、**ansible-sign** で使用されている現在の終了コードであり、安定しているとみなされます。

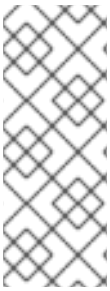
終了コード	概説	シナリオ例
0	正常に接続できる場合	• 署名は成功しました • 検証は成功しました

終了コード	概説	シナリオ例
1	一般的な障害	● 検証中にチェックサムマニフェストファイルに構文エラーが含まれていました ● 検証中に署名ファイルが存在しませんでした ● 署名中に **MANIFEST.in** が存在しませんでした
2	チェックサム検証の失敗	● 検証中に計算されたチェックサムハッシュが、署名されたチェックサムマニフェストに含まれていたものと異なりました (例: プロジェクトファイルが変更されたにもかかわらず、署名プロセスが再完了なかったなど)。
3	署名検証の失敗	● 署名者の公開鍵がユーザーの GPG キーリングにありませんでした ● 間違った GnuPG ホームディレクトリーまたはキーリングファイルが指定されました ● 署名付きチェックサムマニフェストファイルが何らかの方法で変更されました
4	署名プロセスの失敗	● 署名者の秘密鍵が GPG キーリングにありませんでした ● 間違った GnuPG ホームディレクトリーまたはキーリングファイルが指定されました

第18章 インベントリー

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

グループは、Automation controller にホスト名を入力して手動で取得することも、サポートされているクラウドプロバイダーの 1 つから取得することもできます。

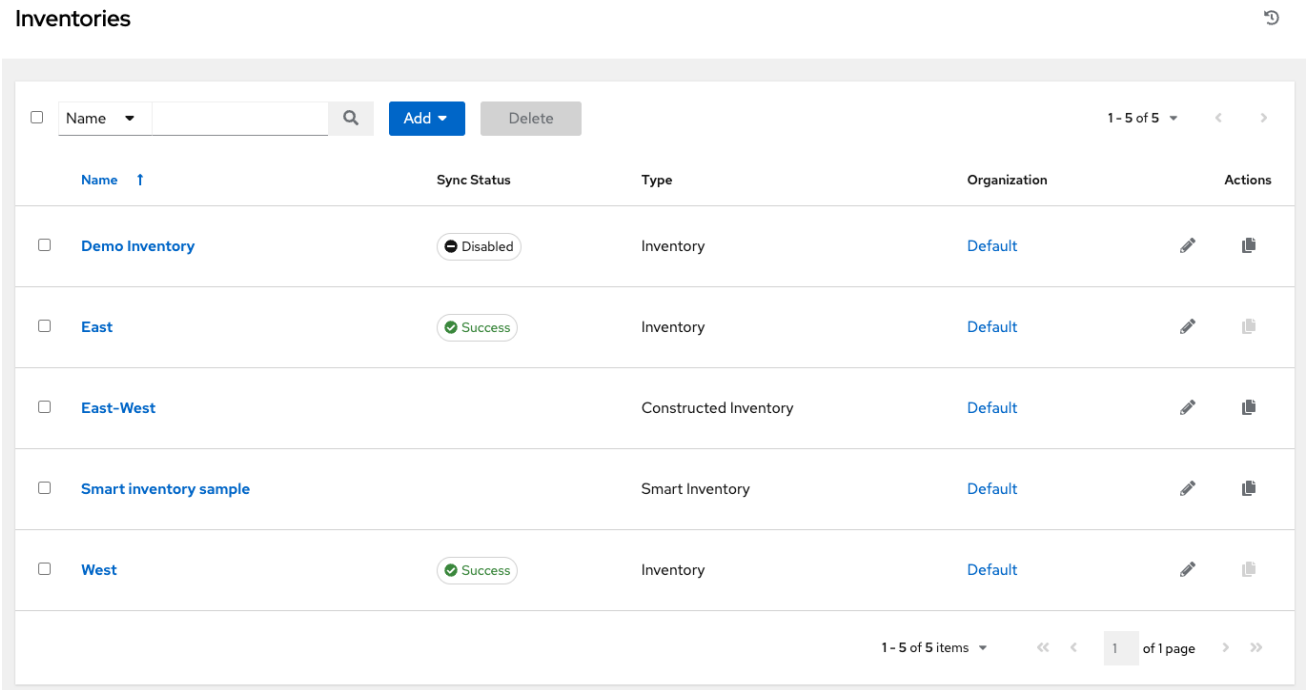


注記

カスタム動的インベントリースクリプト、または Automation controller でまだネイティブにサポートされていないクラウドプロバイダーがある場合は、それを Automation controller にインポートすることもできます。

詳細は、**Automation controller 管理ガイド** の インベントリーファイルのインポート を参照してください。

ナビゲーションパネルから、**Resources → Inventories** を選択します。**Inventories** ウィンドウには、現在使用可能なインベントリーのリストが表示されます。インベントリーリストは、**Name**、**Type**、または **Organization** で並べ替えることができます。



Inventory details ページには次の内容が含まれます。

- **Name**: インベントリー名。
- **Status**

ステータスは以下のとおりです。

- **Success**: インベントリーソースの同期が正常に完了した時

- **Disabled**: インベントリーに追加されたインベントリーソースがない
- **Error**: インベントリーソースの同期がエラーを出して完了する場合
  - **Type**: 標準インベントリー、スマートインベントリー、構築インベントリーのいずれであるかを識別します。
  - **Organization**: インベントリーが属する組織。
  - **Actions**: 以下のアクションを選択したインベントリーについて実行できます。
- **Edit** ( ): 選択したインベントリーのプロパティーを編集します。
- **コピー** : 新しいインベントリーの作成用のテンプレートとして、既存のインベントリーのコピーを作成します。

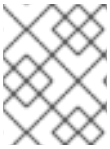
インベントリー名をクリックすると、選択したインベントリーの **Details** ページが表示され、インベントリーのグループとホストが表示されます。

18.1. スマートインベントリー

スマートインベントリーは、保存された検索によって定義されたホストのコレクションであり、標準インベントリーと同様に表示でき、ジョブの実行で簡単に使用できます。組織管理者は組織内のインベントリーに対する管理者パーミッションを持っており、スマートインベントリーを作成できます。

スマートインベントリーは、**KIND=smart** で識別されます。

検索で使用されているのと同じ方法を使用して、スマートインベントリーを定義できます。**InventorySource** はインベントリーに直接関連付けられます。



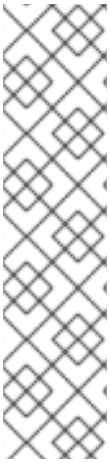
注記

スマートインベントリーは非推奨となり、今後のリリースでは削除される予定です。機能拡張や置き換えのために構築されたインベントリーへの移行を検討してください。

Inventory モデルには、デフォルトで空白にされる以下の新規フィールドがありますが、スマートインベントリーについては以下のように設定されます。

- スマートインベントリーの場合、**Inventory** は **smart** に設定される
- スマートインベントリーの場合、**host_filter** が設定され、**kind** は **smart** に設定される

ホスト モデルには、ホストが関連付けられているすべてのスマートインベントリーのセットを識別する関連エンドポイント、**smart_inventories** があります。メンバーシップテーブルは、スマートインベントリーに対してジョブが実行されるたびに更新されます。



注記

メンバーシップをさらに頻繁に更新するには、**AWX_REBUILD_SMART_MEMBERSHIP** ファイルベースの設定を **True** に変更します。(デフォルトは False です)。これにより、次のイベントが発生した場合にメンバーシップが更新されます。

- 新規ホストが追加される
- 既存ホストが変更される (更新または削除される)
- 新規スマートインベントリーが追加される
- 既存スマートインベントリーが変更される (更新または削除される)

編集できなくてもインベントリーを表示できます。

- ホストおよびグループの名前はインベントリーソースの同期の結果として作成されます。
- グループのレコードは編集したり、移動したりすることはできません。

通常のインベントリーの場合のように、スマートインベントリーホストエンドポイント (**/inventories/N/hosts/**) からホストを作成できません。スマートインベントリーの管理者には、名前、説明、変数などのフィールドを編集するパーミッションと、削除するパーミッションがありますが、スマートインベントリーに含まれるホスト (別のインベントリーでプライマリーメンバーシップを持つホスト) が左右されるため、**host_filter** を変更するパーミッションはありません。

host_filter は、スマートインベントリーの組織に含まれるインベントリーのホストにのみ適用されます。

host_filter を変更するには、インベントリーの組織の組織管理者である必要があります。組織管理者は、組織内のすべてのインベントリーに対する暗黙的な管理者アクセス権を持っているため、組織管理者がまだ所有していないパーミッションが継承されるわけではありません。

スマートインベントリーの管理者は、他のユーザー (組織の管理者ではないユーザー) に、スマートインベントリーに対する使用やアドホックなどのアクセス許可を付与できます。これらは、他の標準インベントリーと同様に、ロールで指定されたアクションを許可します。ただし、これはホスト (別のインベントリーに存在する) に特別なパーミッションを付与するものではありません。ホストへの直接読み取りパーミッションは割り当てられず、**/#/hosts/** の下にある追加のホストを表示することも許可されませんが、スマートインベントリーホストリストの下にあるホストは表示できます。

状況によっては、以下のものを変更できます。

- インベントリーソースを使用してインベントリーに手動で作成した新しいホスト。
- インベントリーソースの同期の結果として作成されたグループ。
- ホストとグループの変数は、ローカルシステム管理者であっても変更できません。

スマートインベントリーに関連付けられたホストは表示される際に明示されます。スマートインベントリーの結果に同じホスト名を持つ複数のホストが含まれる場合、一致するホストの 1 つのみがスマートインベントリーの一部として組み込まれ、ホスト ID 別に並べ替えられます。

18.1.1. スマートホストフィルター

検索フィルターを使用して、インベントリーのホストを入力できます。この機能はファクト検索機能を使用します。

Automation controller は、ジョブテンプレート別に **use_fact_cache=True** が設定されるたびに、ジョブテンプレートを実行中に Ansible Playbook で生成されたファクトをデータベースに保存します。新しいファクトはホスト別となっており、既存のファクトとマージされます。これらの保存されたファクトは、**GET** クエリーパラメーター **host_filter** を使用して **/api/v2/hosts** エンドポイントでホストをフィルター処理するために使用できます。

以下に例を示します。

```
/api/v2/hosts?host_filter=ansible_facts__ansible_processor_vcpus=8
```

host_filter パラメーターでは次のことが許可されます。

- () でのグループ化
- ブール値および演算子の使用:
  - __ 関連付けられたフィールドの関連フィールドを参照します。
  - __ JSON キーパスでキーを分離するために ansible_facts で使用されます。
  - `[] パスの指定で json アレイを表すために使用されます。
  - **""** 値にスペースが必要な場合は、その値に使用されます。
- "標準的" な Django クエリーを **host_filter** に組み込むことができる

例 :

```
/api/v2/hosts/?host_filter=name=localhost
/api/v2/hosts/?host_filter=ansible_facts__ansible_date_time__weekday_number="3"
/api/v2/hosts/?host_filter=ansible_facts__ansible_processor[]="GenuineIntel"
/api/v2/hosts/?host_filter=ansible_facts__ansible_lo__ipv6[]__scope="host"
/api/v2/hosts/?host_filter=ansible_facts__ansible_processor_vcpus=8
/api/v2/hosts/?host_filter=ansible_facts__ansible_env__PYTHONUNBUFFERED="true"
/api/v2/hosts/?host_filter=(name=localhost or name=database) and (groups__name=east or
groups__name="west coast") and ansible_facts__an
```

host_filter は、**ホスト名**、**グループ名**、および **Ansible ファクト** 別に検索できます。

グループ検索の形式は次のとおりです。

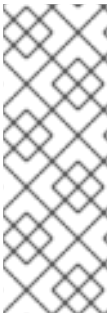
```
groups.name:groupA
```

ファクト検索の形式は次のとおりです。

```
ansible_facts.ansible_fips:false
```

また、ホスト名とホストの説明で設定されるスマート検索の検索を実行することもできます。

```
host_filter=name=my_host
```



注記

host_filter の検索語が文字列型の場合、数値 (例: **2.66**) または JSON キーワード (例: **null**、**true**、または **false**) の値を有効にするには、値の前後に二重引用符を追加して、コントローラーが検索語を二重引用符で囲み、非文字列として解析されないようにします。

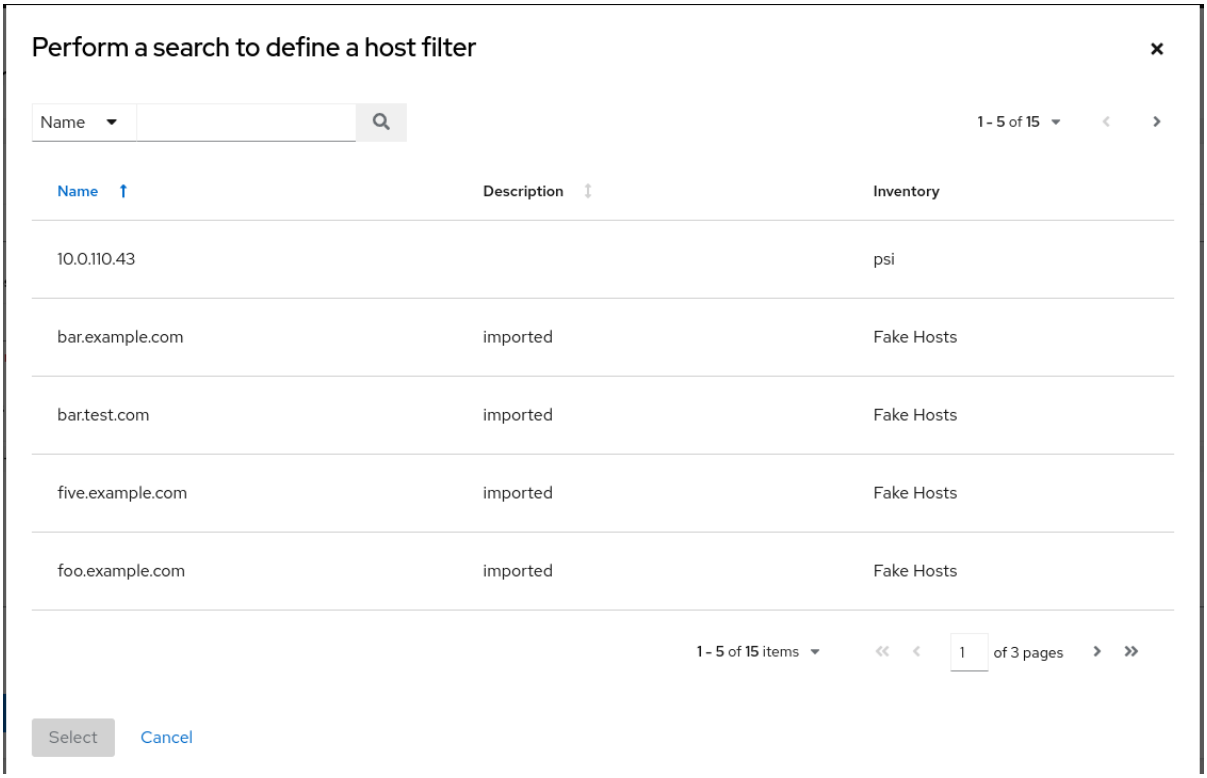
```
host_filter=ansible_facts__packages__dnsmasq[]__version="2.66"
```

18.1.2. ansible_facts を使用したホストフィルターの定義

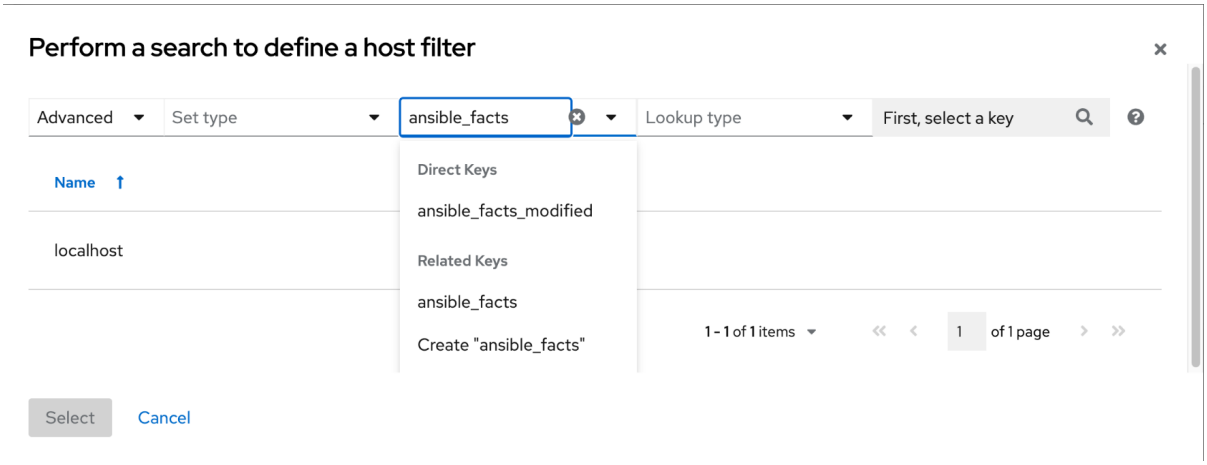
スマートインベントリーの作成時に **ansible_facts** を使用してホストフィルターを定義するには、次の手順を実行します。

手順

1. ナビゲーションパネルから、**Resources → Inventories** を選択します。
2. **Add** をクリックし、メニューから **Add Smart Inventory** を選択します。
3. **Create new smart inventory** ページで、 **Smart host filter** フィールドのアイコンをクリックします。これにより、このインベントリーのホストをフィルタリングするためのウィンドウが開きます。



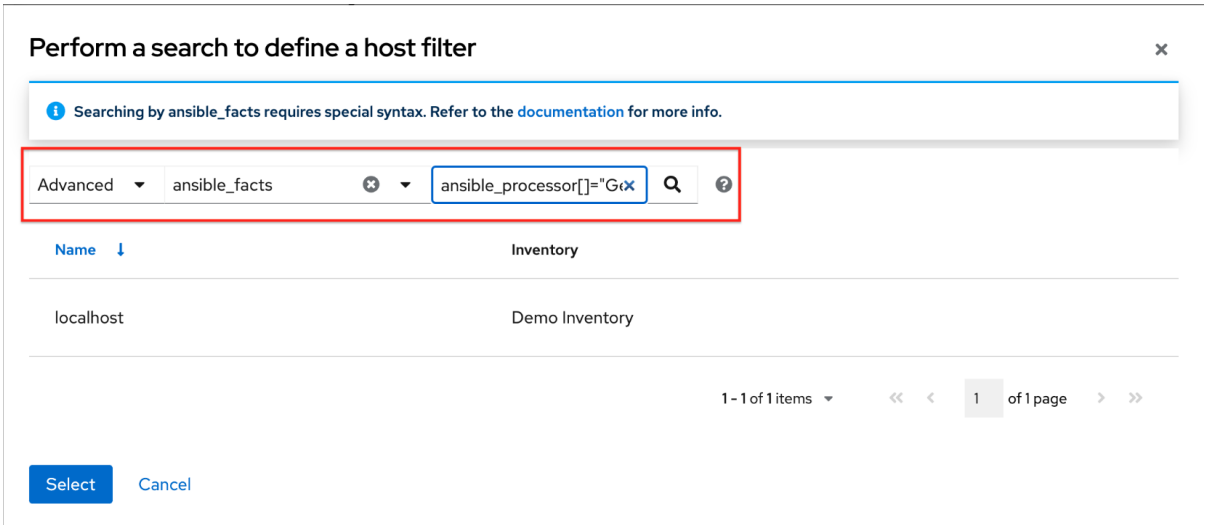
4. 検索メニューで、検索基準を **Name** から **Advanced** に変更し、**Key** フィールドから **ansible_facts** を選択します。



次の Ansible fact を追加する場合:

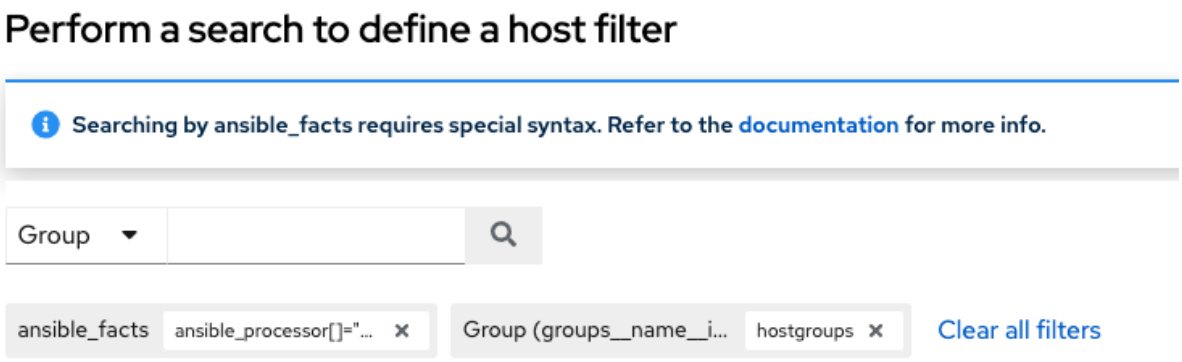
```
/api/v2/hosts/?host_filter=ansible_facts__ansible_processor[]="GenuineIntel"
```

検索フィールドに **ansible_processor="OriginalIntel"** と入力し (値の前に余分なスペースや __ はなし)、**Enter** をクリックします。



指定した Ansible fact の検索条件が表示されます。

5. **Select** をクリックして、**Smart host filter** フィールドに追加します。
6. **Save** をクリックします。
7. 新規スマートインベントリーの **Details** タブが開き、**Smart host filter** フィールドに指定した Ansible ファクトが表示されます。
8. **Details** ビューで、**Edit** をクリックして **Smart host filter** フィールドを編集することや、既存フィルターの削除、すべての既存フィルターの消去、新しいフィルターの追加が可能です。



18.2. 構築されたインベントリー

入力インベントリーのリストから新しいインベントリー (構築されたインベントリーと呼ばれる) を作成できます。

構築されたインベントリーには、入力インベントリー内のホストとグループのコピーが含まれており、ジョブが複数のインベントリーにわたるサーバーのグループを対象にできます。グループとホスト変数をインベントリーのコンテンツに追加したり、ホストをフィルタリングして構築されたインベントリーのサイズを制限したりできます。

構築されたインベントリーは、ansible.builtin.constructed インベントリー モデルを使用します。

構築されたインベントリーは、主に次のような要素が含まれます。

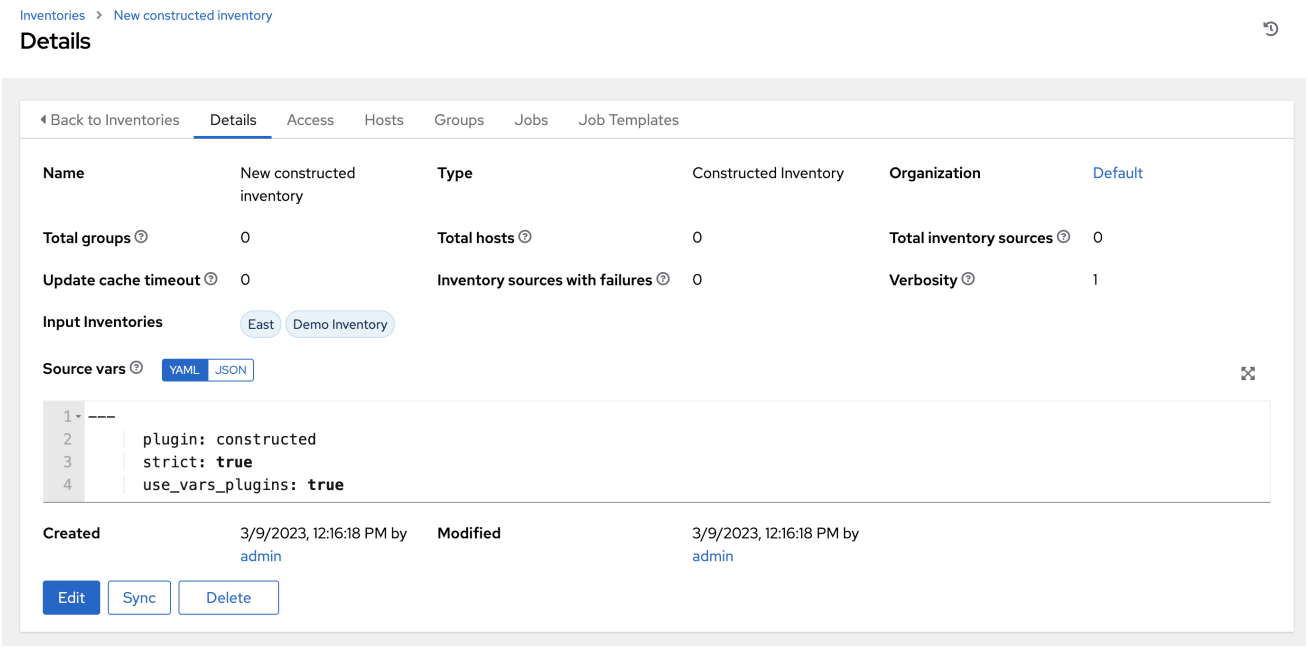
- 通常の Ansible hostvars 名前空間が利用可能である
- グループがある

構築されたインベントリーは、**source_vars** と **limit** を入力として受け取り、その **input_inventories** をグループを備えた新しいインベントリーに変換します。グループ (既存または構築された) を **limit** フィールドで参照して、生成されるホストの数を減らすことができます。

次のホストプロパティーに基づいてグループを構築できます。

- RHEL のメジャーバージョンまたはマイナーバージョン
- Windows ホスト
- 特定のリージョンでタグ付けされたクラウドベースのインスタンス
- その他

以下は、構築されたインベントリー詳細ビューの例です。



後続のセクションで説明する例は、入力インベントリーの構造別に整理されています。

18.2.1. グループ名と変数のフィルタリング

グループと変数の組み合わせに対してフィルタリングできます。たとえば、グループ変数値に一致し、ホスト変数値にも一致するホストをフィルタリングできます。

このフィルターを実行するには 2 つの方法があります。

- 2 つのグループを定義します。1 つのグループはグループ変数と一致し、もう 1 つのグループはホスト変数値と一致します。**limit** パターンを使用して、両方のグループに含まれるホストを返します。これは、推奨の手法です。
- 1 つのグループを定義します。定義には、グループ変数とホスト変数が特定の値に一致する必要があるという条件を含めます。**limit** パターンを使用して、新しいグループ内のすべてのホストを返します。

以下に例を示します。

次のインベントリーファイルは 4 つのホストを定義し、グループ変数とホスト変数を設定します。これは product グループと sustaining グループを定義し、2 つのホストをシャットダウン状態に設定します。

目標は、シャットダウンされた運用ホストのみを返すフィルターを作成することです。

```
[account_1234]
host1
host2 state=shutdown

[account_4321]
host3
host4 state=shutdown

[account_1234:vars]
account_alias=product_dev
```

```
[account_4321:vars]
account_alias=sustaining
```

ここでの目標は、**product_dev** と同等の **account_alias** 変数を持つグループ内に存在し、シャットダウンされているホストのみを返すことです。これには 2 つのアプローチがあり、どちらも YAML 形式で示されています。最初に提案する内容が推奨されます。

1. **2 つのグループを構築して交差部分に限定します。**
   source_vars:

   ```
   plugin: constructed
   strict: true
   groups:
     is_shutdown: state | default("running") == "shutdown"
     product_dev: account_alias == "product_dev"
   ```

   limit: **is_shutdown:&product_dev**

   この構築されたインベントリーの入力は、両方のカテゴリーのグループを作成し、**limit** ホストパターン) を使用して、これら 2 つのグループの交差部分にあるホストのみを返します。これは、Patterns:targeting hosts and groups に文書化されています。

   変数が定義されている場合と定義されていない場合 (ホストに応じて)、デフォルトを指定できます。たとえば、定義されていない場合に対象の値がわかっている場合は、| **default("running")** を使用します。これは、デバッグのヒント で説明されているように、デバッグに役立ちます。

2. **1 つのグループを構築してグループに限定します。**
   source_vars:

   ```
   plugin: constructed
   strict: true
   groups:
     shutdown_in_product_dev: state | default("running") == "shutdown" and account_alias ==
   "product_dev"
   ```

   limit: **shutdown_in_product_dev**

   この入力により、両方の基準に一致するホストのみを含む 1 つのグループが作成されます。この場合、制限はグループ名そのものとなり、**host2** が返されます。前述の方法と同じです。

18.2.2. デバッグのヒント

テンプレートの問題をデバッグできるように、**strict** パラメーターを **true** に設定することが重要です。テンプレートのレンダリングに失敗すると、その構築されたインベントリーに関連するインベントリーの更新でエラーが発生します。

エラーが発生した場合は、詳細レベルを上げてさらなる情報を取得します。

Ansible の Jinja2 テンプレートでは、| **default("running")** などでデフォルトを指定します。こうすることで、**strict: true** を設定した場合にテンプレートで発生するエラーを回避できます。

strict: false を設定して、テンプレートがエラーを生成できるようにすることもできます。その結果、ホストはそのグループに含まれなくなります。ただし、これを行うと、今後、テンプレートが継続して複雑化した場合、問題のデバッグが困難になります。

テンプレートが予期したインベントリーコンテンツを生成しない場合は、テンプレートで想定されている機能についてデバッグする必要がある場合があります。たとえば、**groups** グループに複雑なフィルター (**shutdown_in_product_dev** など) があるものの、結果として構築されたインベントリーにホストが含まれていない場合は、デバッグに役立つ **compose** パラメーターを使用します。

以下に例を示します。

```
source_vars:

plugin: constructed
strict: true
groups:
  shutdown_in_product_dev: state | default("running") == "shutdown" and account_alias ==
"product_dev"
compose:
  resolved_state: state | default("running")
  is_in_product_dev: account_alias == "product_dev"

limit: ``
```

limit を空白にして実行すると、すべてのホストが返されます。これを使用して、特定のホスト上の特定の変数を検査し、**groups** 内のどこに問題があるかの見解を得ることができます。

18.2.3. ネストされたグループ

ネストされたグループは、一方が他方の子である 2 つのグループで設定されます。次の例では、子グループの内部に別のホストがあり、親グループには変数が定義されています。

親グループの変数は、Ansible Core が動作する方法により、playbook として対象の名前空間で利用でき、フィルタリングに使用できます。

次のインベントリーファイルの例、**nested.yml** は YAML 形式です。

```
all:
  children:
    groupA:
      vars:
        filter_var: filter_val
      children:
        groupB:
          hosts:
            host1: {}
    ungrouped:
      hosts:
        host2: {}
```

host1 は **groupB** に属しているため、**groupA** にも属します。

ネストされたグループ名でのフィルタリング

ネストされたグループ名をフィルターするには、次の YAML 形式を使用します。

```
`source_vars`:

plugin: constructed

`limit`: `groupA`
```

ネストされたグループプロパティーでフィルタリングする

ホストが間接的にそのグループ二所属する場合でも、グループ変数でフィルタリングするには、次の YAML 形式を使用します。

インベントリーの内容では、**host2** はどのグループにも属していないため、想定として変数 **filter_var** は定義されない点に注意してください。**strict: true** を使用しているため、その変数を持たないホストが定義されるようにデフォルト値を使用します。これを使用すると、**host2** はエラーを生成するのではなく、式から **false** を返します。**host1** は、そのグループから変数を継承し、返されます。

```
source_vars:

plugin: constructed
strict: true
groups:
  filter_var_is_filter_val: filter_var | default("") == "filter_val"

limit: filter_var_is_filter_val
```

18.2.4. Ansible ファクト

Ansible fact を使用してインベントリーを作成するには、**Gather_facts: true** という設定のインベントリーに対して Playbook を実行する必要があります。ファクトはシステムごとに異なります。次の例は、すべての既知のシナリオに対処することを目的としたものではありません。

18.2.4.1. 環境変数のフィルタリング

次の例には、YAML 形式を使用した環境変数のフィルタリングが含まれます。

```
source_vars:

plugin: constructed
strict: true
groups:
  hosts_using_xterm: ansible_env.TERM == "xterm"

limit: hosts_using_xterm
```

18.2.4.2. プロセッサーの種類ごとにホストをフィルタリングする

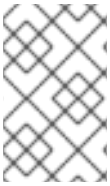
次の例では、YAML 形式を使用してプロセッサータイプ (Intel) ごとにホストをフィルタリングします。

```
source_vars:

plugin: constructed
strict: true
groups:
```

```
  intel_hosts: "GenuineIntel" in ansible_processor

limit: intel_hosts
```



注記

構築されたインベントリー内のホストは、元のインベントリーホストを参照しているため、ライセンスの割り当てにはカウントされません。さらに、元のインベントリーで無効になっているホストは、構築されたインベントリーには含まれません。

ansible-inventory を使用してインベントリー更新を実行すると、構築されたインベントリーのコンテンツが作成されます。

これは、常にジョブ実行前の起動時に更新されるように設定されていますが、この操作に時間がかかる場合に備え、キャッシュのタイムアウトを選択することもできます。

構築されたインベントリーを作成するとき、API で、このインベントリーに関連付けられているインベントリーソースが必ず 1 つとなるようにします。インベントリー更新にはすべて、関連するインベントリーソースがあり、構築されたインベントリーに必要なフィールド (**source_vars** および **limit**) は、インベントリーソースモデルにすでに存在します。

18.3. INVENTORY プラグイン

インベントリーの更新では、動的に生成された YAML ファイルが使用され、それぞれのインベントリープラグインによって解析されます。Automation controllerv4.4 では、次のインベントリーソースのインベントリーソース **source_vars** を使用して、インベントリープラグイン設定を Automation controller に直接提供できます。

- Amazon Web Services EC2
- Google Compute Engine
- Microsoft Azure Resource Manager
- VMware vCenter
- Red Hat Satellite 6
- Red Hat Insights
- OpenStack
- Red Hat Virtualization
- Red Hat Ansible Automation Platform

インベントリーソース用に新しく作成された設定には、デフォルトのプラグイン設定値が含まれています。新しく作成したインベントリーソースを従来のソースの出力と一致させるには、そのソースに特定の設定値のセットを適用する必要があります。下位互換性を確保するために、Automation controller はこれらの各ソースにテンプレートを使用して、強制的にインベントリープラグインの出力をレガシー形式にします。

ソースとそれぞれのテンプレートの詳細は、サポートされているインベントリープラグインテンプレート を参照してください。

最上位キーとして **plugin: foo.bar.baz** を含む **source_vars** は、実行時に **InventorySource** ソースに基づいて完全修飾インベントリープラグイン名に置き換えられます。たとえば、**InventorySource** に ec2 が選択されている場合、実行時にプラグインは **amazon.aws.aws_ec2** に設定されます。

18.4. 新規インベントリーの追加

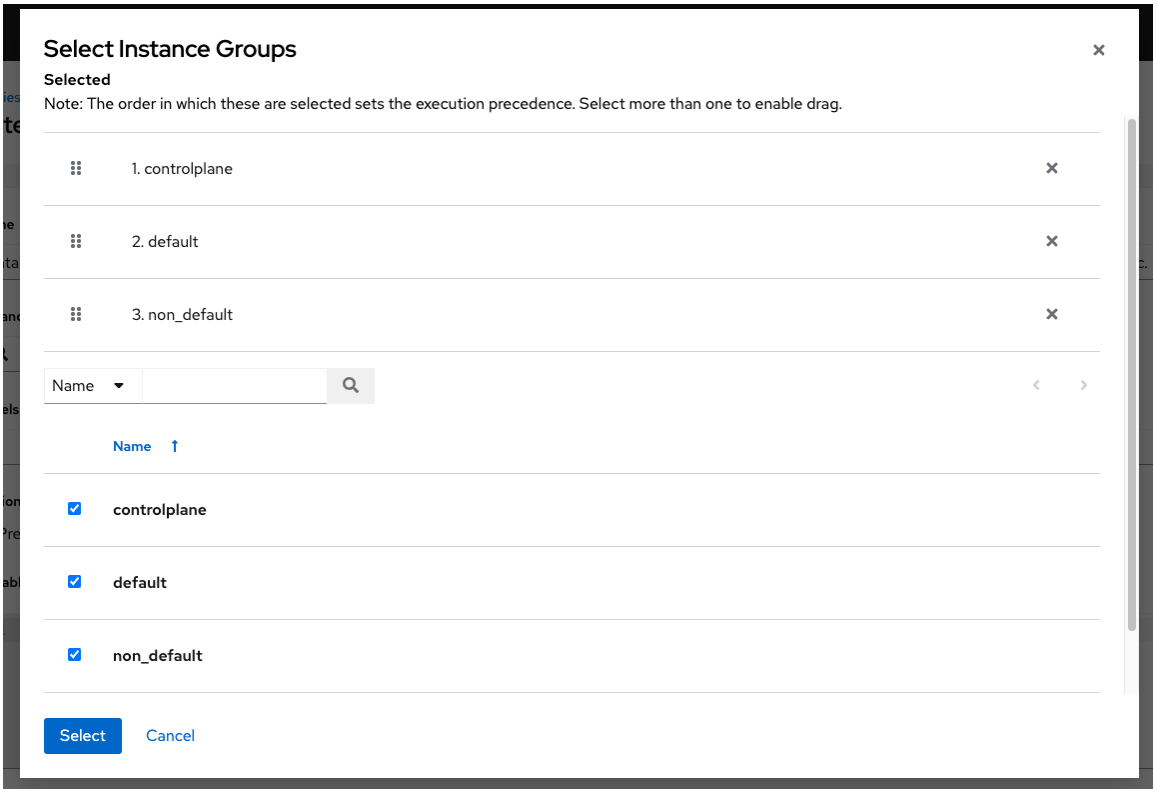
新しいインベントリーの追加には、次のコンポーネントが含まれます。

- インベントリーへのパーミッションの追加
- インベントリーへのグループの追加
- ホストの追加
- ソースの追加
- 完了したジョブの表示

インベントリーを作成するには、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、**Resources → Inventories** を選択します。**Inventories** ウィンドウには、現在使用可能なインベントリーのリストが表示されます。
2. **追加** をクリックし、作成するインベントリーのタイプを選択します。
3. 以下のフィールドに該当する詳細を入力します。
   - **Name**: このインベントリーに適した名前を入力します。
   - オプション: **Description**: 任意の説明を入力します (オプション)。
   - **Organization**: 必須。利用可能な組織の中から選択します。
   - スマートインベントリー (**Smart Host Filter**) にのみ適用: 🔍 アイコンをクリックして別のウィンドウを開き、このインベントリーのホストをフィルタリングします。これらのオプションは、選択した組織に基づいています。
     フィルターは、タグがそれらの名前を含む特定のホストをフィルタリングするために使用されるという点でタグに似ています。したがって、**Smart Host Filter** フィールドに値を設定するには、ホスト自体ではなく、必要なホストを含むタグを指定します。**Search** フィールドにタグを入力し、**Enter** をクリックします。フィルターでは大文字と小文字が区別されます。詳細は、スマートホストフィルター を参照してください。
   - **Instance Groups**: 🔍 アイコンをクリックすると、別のウィンドウが開きます。このインベントリーを実行するインスタンスグループを選択します。リストが膨大な場合は、検索を使用してオプションを絞り込みます。複数のインスタンスグループを選択し、実行する順序で並べ替えることができます。



- オプション: **Labels**: このインベントリーを説明するラベルを指定するため、インベントリーとジョブのグループ化とフィルタリングに使用できます。

- 構築されたインベントリーにのみ適用されます。**Input inventories**: この構築されたインベントリーに含めるソースインベントリーを指定します。 アイコンをクリックして、利用可能なインベントリーから選択します。入力インベントリーの空のグループは、構築されたインベントリーにコピーされます。

- オプション:(構築されたインベントリーにのみ適用可能): **キャッシュされたタイムアウト (秒)**: キャッシュプラグインデータのタイムアウト時間を設定します。

- 構築されたインベントリーにのみ適用されます。**Verbosity**: 構築されたインベントリーに関連付けられたインベントリーソースに関連する Playbook の実行時に Ansible が生成する出力のレベルを制御します。冗長性を標準からさまざまな冗長またはデバッグ設定まで選択します。これは詳細レポートビューにのみ表示されます。

  - 詳細ログには、すべてのコマンドの出力が含まれます。

  - デバッグログは非常に詳細であり、特定のサポートインスタンスで役立つ SSH 操作に関する情報が含まれています。ほとんどのユーザーはデバッグモードの出力を確認する必要はありません。

- 構築されたインベントリーにのみ適用されます。**Limit**: 構築されたインベントリーに関連付けられたインベントリーソースに対して返されるホストの数を制限します。グループ名を limit フィールドに貼り付けて、そのグループ内のホストのみを含めることができます。詳細は、**Source vars** 設定を参照してください。

- 標準インベントリーにのみ適用されます。**オプション**: **Prevent Instance Group Fallback** オプションをオンにして、**Instance Groups** フィールドにリストされているインスタンスグループのみがジョブを実行できるようにします。チェックを外した場合、実行プールにある使用可能なインスタンスはすべて **Automation controller 管理ガイド** の ジョブの実行場所の制御 で説明されている階層に基づいて使用されます。 アイコンをクリックすると追加情報が表示されます。

注記

API を介してスマートインベントリーの **Prevent_instance_group_fallback** オプションを設定します。

- Variables (構築されたインベントリーの Source vars):
  - Variables このインベントリー内のすべてのホストに適用する変数の定義と値。JSON または YAML 構文を使用して変数を入力します。ラジオボタンを使用して、その 2 つを切り替えます。
  - 構築されたインベントリーの Source vars は、特にデータの **groups** キーの下にグループを作成します。Jinja2 テンプレート構文を受け入れ、すべてのホストに対してレンダリングし、**true** または **false** の評価を行い、結果が **true** の場合はそのホストを (エントリーのキーから) グループに含めます。これは、そのグループ名を limit フィールドに貼り付けて、そのグループ内のホストのみを含めることができるため、特に便利です。スマートホストフィルター の例 1 を参照してください。

4. **Save** をクリックします。

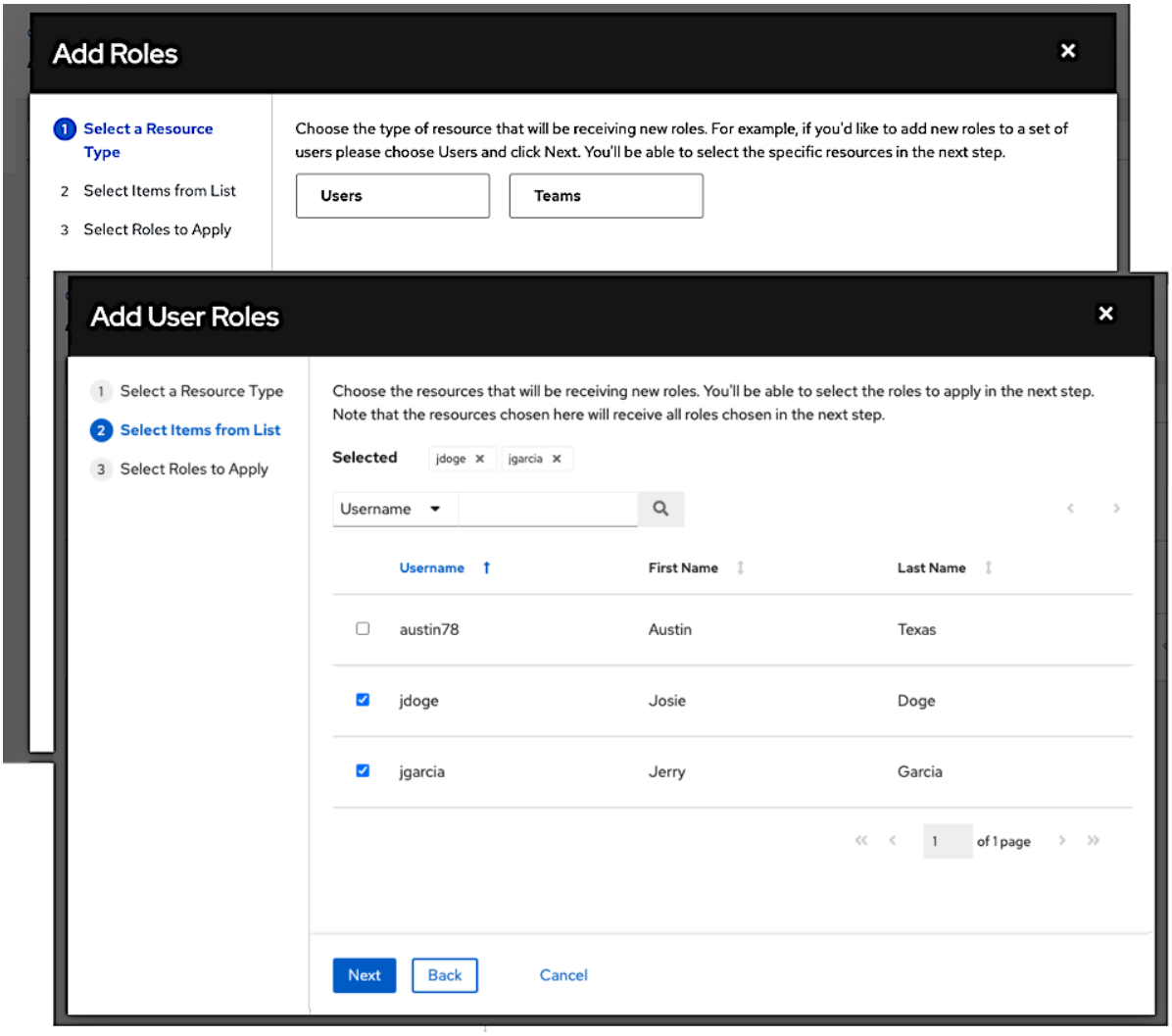
新規インベントリーが保存された後に、パーミッション、グループ、ホスト、およびソースの設定や、完了したジョブの表示に進みます (インベントリーのタイプに応じて適用可能な場合)。

18.4.1. インベントリーへのパーミッションの追加

インベントリーにパーミッションを追加するには、次の手順を使用します。

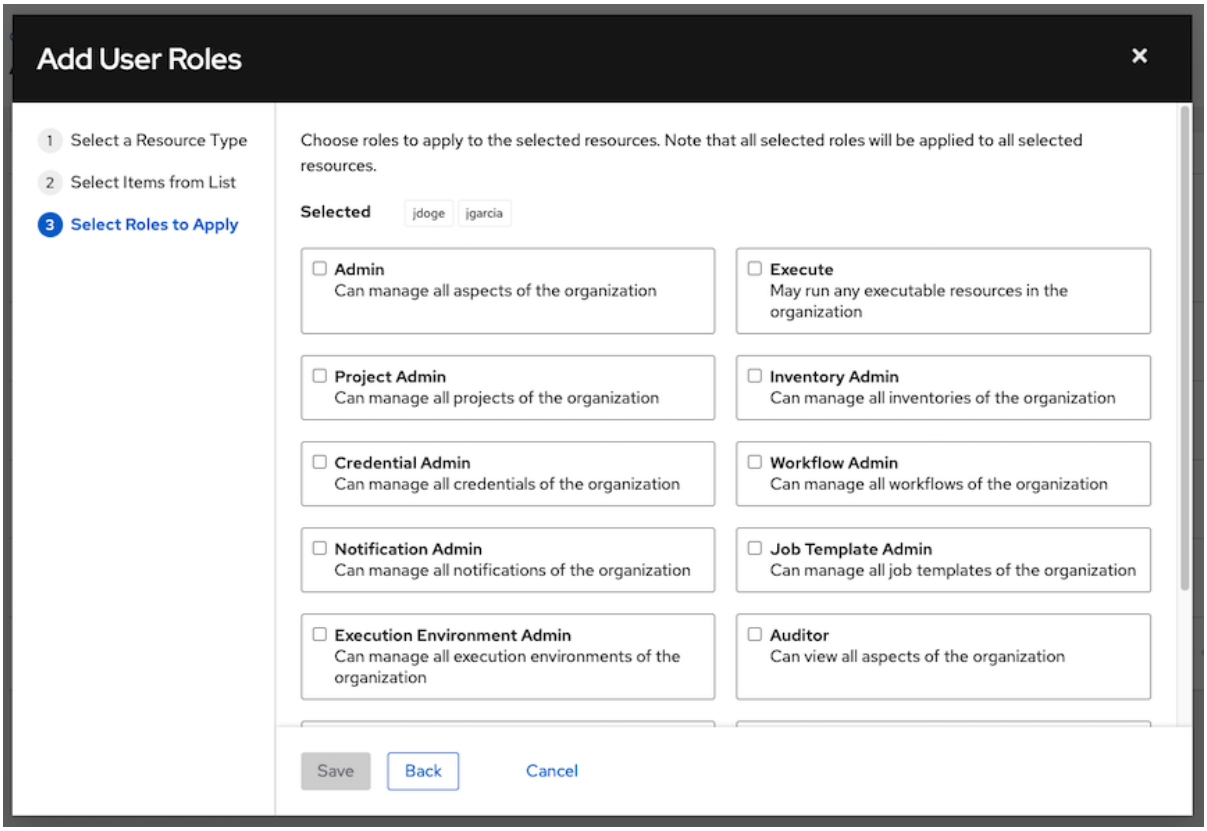
手順

1. ナビゲーションパネルから、Resources → Inventories を選択します。
2. テンプレートを選択し、Access tab で **Add** をクリックします。
3. 追加するユーザーまたはチームを選択し、**Next** をクリックします。
4. 名前の横にあるチェックボックスを選択して、リストからユーザーまたはチームを 1 つ以上メンバーとして追加します。
5. **Next** をクリックします。



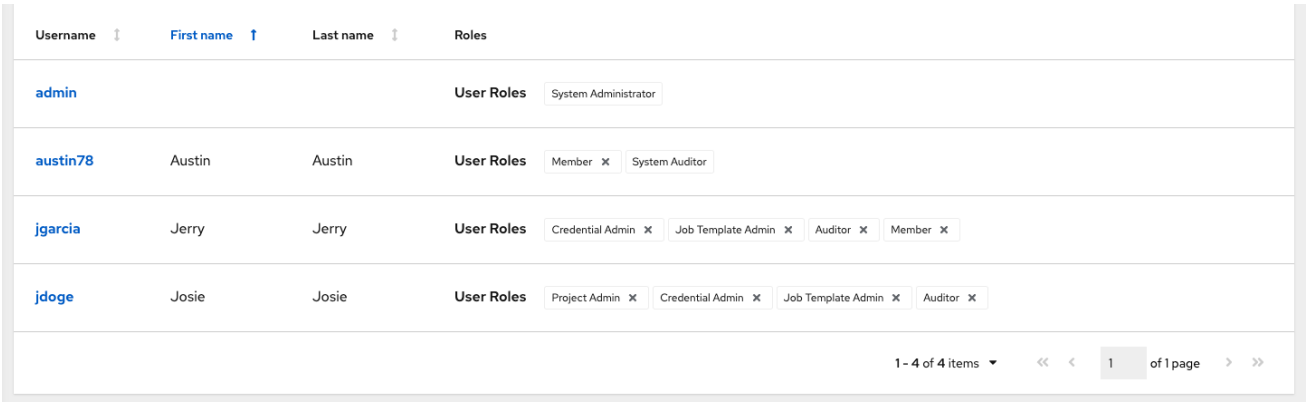
この例では、追加するユーザーが 2 つ選択されています。

6. 選択したユーザーまたはチームに付与するロールを選択します。下にスクロールして、ロールの完全なリストを表示します。リソースが異なれば、利用可能なオプションも異なります。



7. **Save** をクリックして、選択したユーザーまたはチームにロールを適用し、メンバーとして追加します。

ユーザー/チームの追加ウィンドウが閉じ、各ユーザーやチームに割り当てられた更新済みのロールが表示されます。



パーミッションの削除

- 特定のユーザーのロールを削除するには、リソースの横にある ✖ アイコンをクリックします。

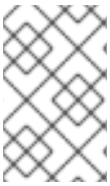
これにより、確認ウィンドウが起動し、関連付けの解除を確認するよう求められます。

18.4.2. インベントリーへのグループの追加

インベントリーはグループに分割されており、グループにはホストと他のグループを含めることができます。グループは標準インベントリーにのみ適用できます。スマートインベントリーを使用して直接設定できません。標準インベントリーで使用されるホストを介して既存のグループを関連付けることができます。

標準インベントリーでは次のアクションを使用できます。

- 新規グループの作成
- 新規ホストの作成
- 選択したインベントリーでのコマンドの実行
- インベントリープロパティーの編集
- グループおよびホストのアクティビティーストリームの表示
- インベントリーの構築に関するヘルプの取得



注記

インベントリーソースはグループに関連付けられません。生成されたグループはトップレベルであり、引き続き子グループを持つことができます。これらの生成されたグループはいずれも、ホストを持つことができます。

インベントリーの新しいグループを作成するには、次の手順を使用します。

手順

1. グループを追加するインベントリー名を選択します。
2. インベントリーの **Details** ページで、**Groups** を選択します。
3. **Add** をクリックして **Create Group** ウィンドウを開きます。
4. 適切な詳細を入力します。
   - **Name**: 必須
   - オプション: **Description**: 必要に応じて説明を入力します。
   - **Variables**: このグループ内のすべてのホストに適用される定義と値を入力します。JSON または YAML 構文を使用して変数を入力します。ラジオボタンを使用して、その 2 つを切り替えます。
5. **Save** をクリックします。
6. テンプレートにグループを追加すると、**Group details** ページが表示されます。

18.4.2.1. グループ内でのグループの追加

グループ内でグループを追加するには、次の手順を使用します。

手順

1. テンプレートにグループを追加すると、**Group details** ページが表示されます。
2. **Related Groups** タブを選択します。
3. **Add** をクリックします。
4. 設定にすでに存在するグループを追加するか、新しいグループを作成するかを選択します。

5. 新規グループを作成する場合、該当する詳細情報を必須およびオプションフィールドに入力します。

    - **Name** (必須):
    - オプション: **Description**: 必要に応じて説明を入力します。
    - **Variables**: このグループ内のすべてのホストに適用される定義と値を入力します。JSON または YAML 構文を使用して変数を入力します。ラジオボタンを使用して、その 2 つを切り替えます。

6. **Save** をクリックします。
7. **Create Group** ウィンドウが閉じ、新規に作成されたグループが、元のグループに関連付けられたグループのリストのエントリーとして表示されます。

既存グループを追加する選択をした場合、選択可能なグループが別個の選択ウィンドウに表示されます。

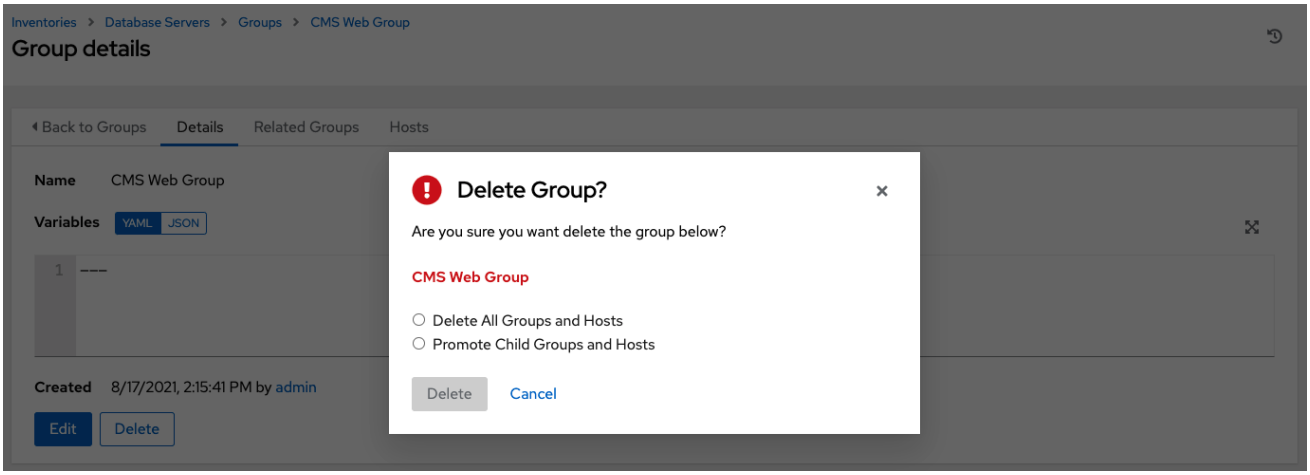
グループを選択すると、そのグループに関連付けられているグループのリストに表示されます。

- サブグループの下に追加のグループとホストを設定するには、グループのリストからサブグループの名前をクリックし、このセクションに記載されている手順を繰り返します。

18.4.2.2. インベントリーグループの表示または編集

グループリストビューにはすべてのインベントリーグループが表示されます。また、ルートグループのみを表示するようにフィルタリングすることもできます。インベントリーグループは、別のグループのサブセットではない場合、ルートグループとみなされます。

Automation controller は子グループやホストなどの依存関係を検索するため、依存関係を気にせずにサブグループを削除できます。存在する場合は、ルートグループとそのすべてのサブグループおよびホストを削除するかどうかを選択する確認ウィンドウが表示されます。または、サブグループをプロモートして、ホストとともに最上位のインベントリーグループになります。



18.4.3. インベントリーへのホストの追加

インベントリーだけでなく、グループやグループ内のグループに対してもホストを設定できます。

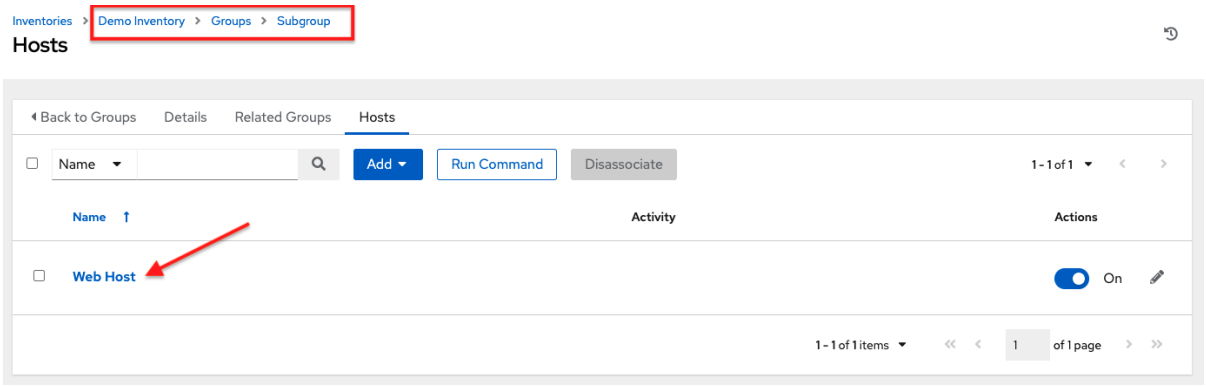
ホストを追加するには、次の手順を使用します。

手順

1. グループを追加するインベントリー名を選択します。
2. インベントリーの **Details** ページで、**Hosts** を選択します。
3. **Add** をクリックします。
4. 設定にすでに存在するホストを追加するか、新しいホストを作成するかを選択します。
5. 新しいホストを作成する場合は、トグルを **On** に設定して、ジョブの実行中にこのホストを含めます。
6. 適切な詳細を入力します。
   - **Host Name** (必須):
   - オプション: **Description**: 必要に応じて説明を入力します。
   - **Variables**: 次の例のように、このグループ内のすべてのホストに適用される定義と値を入力します。

     ```
     {
       ansible_user : <username to ssh into>
       ansible_ssh_pass : <password for the username>
       ansible_become_pass: <password for becoming the root>
     }
     ```

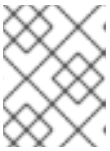
     JSON または YAML 構文を使用して変数を入力します。ラジオボタンを使用して、その 2 つを切り替えます。
7. **Save** をクリックします。
8. **Create Host** ウィンドウが閉じ、新規に作成されたホストが、元のグループに関連付けられたホストのリストのエントリーとして表示されます。



既存ホストを追加する選択をした場合、選択可能なホストが別個の選択ウィンドウに表示されます。

ホストを選択すると、グループに関連付けられているホストのリストに表示されます。

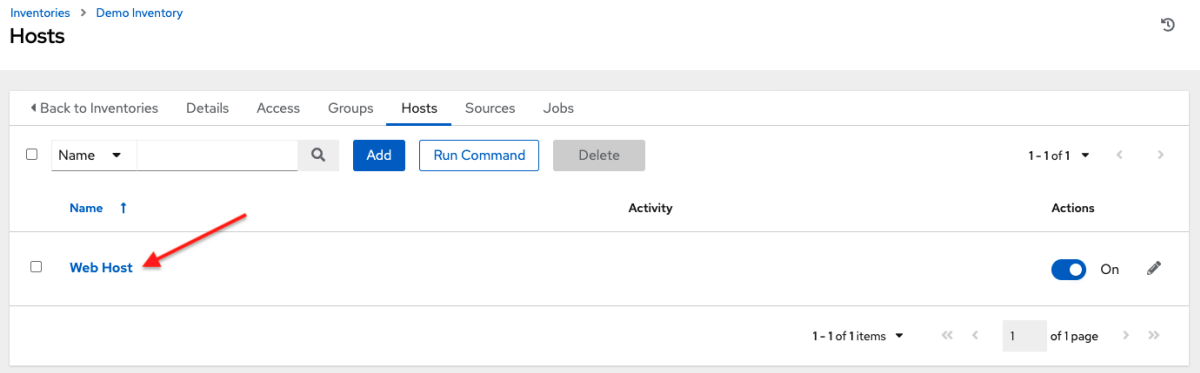
9. この画面でホストを選択し、✖ アイコンをクリックしてホストの関連付けを解除します。



注記

この画面からアドホックコマンドを実行することもできます。詳細は、アドホックコマンドの実行を参照してください。

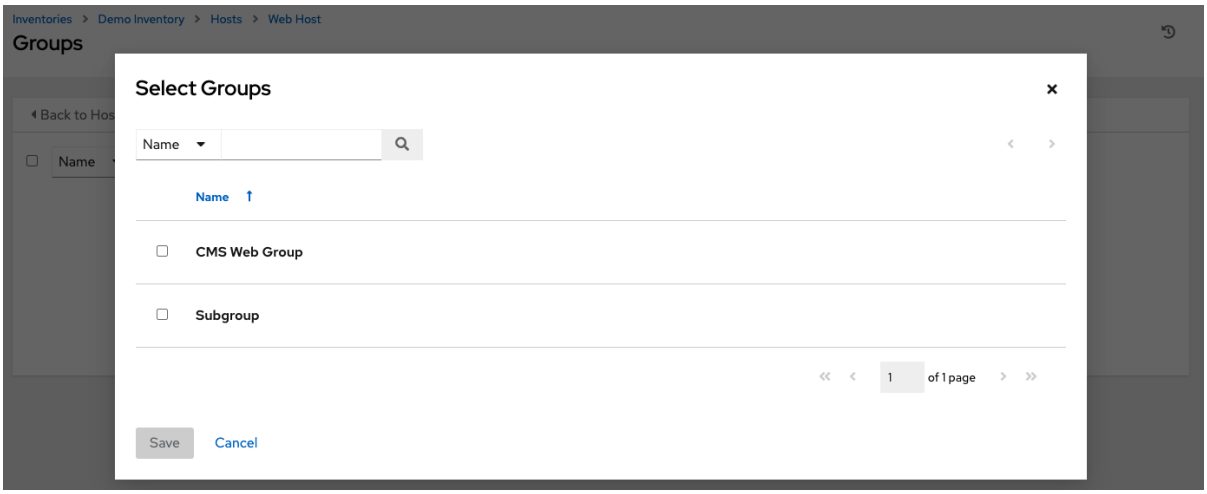
10. ホストの追加グループを設定するには、ホストのリストからホストの名前をクリックします。



これにより、選択したホストの **Details** タブが開きます。

11. **Groups** を選択して、ホストのグループを設定します。

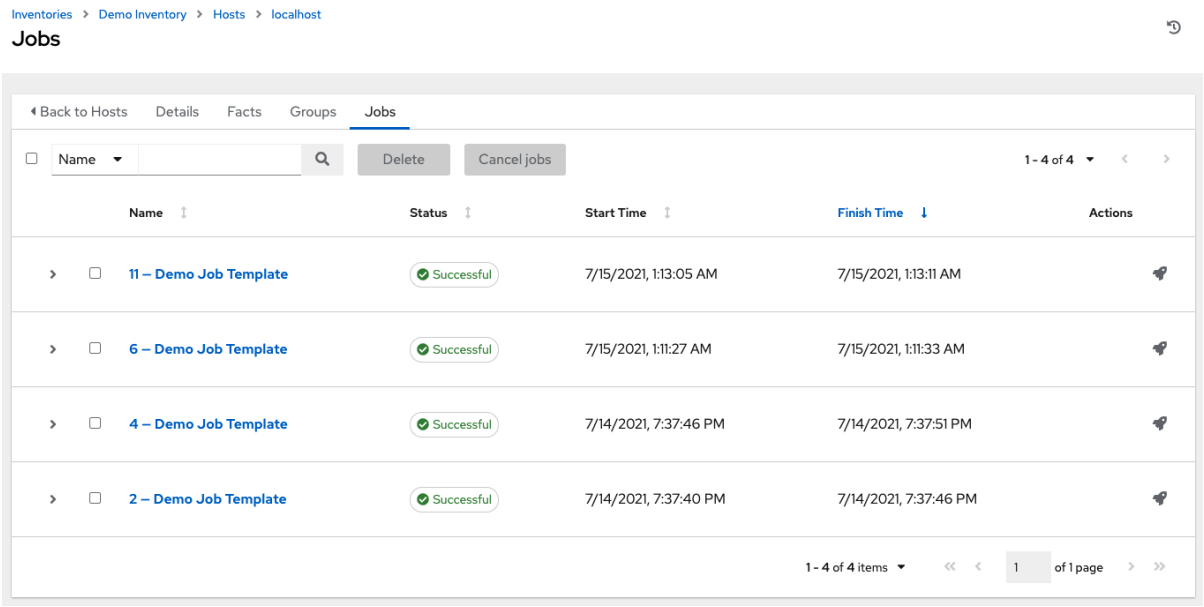
12. **Add** をクリックして、ホストを既存のグループに関連付けます。選択可能なグループが別個の選択ウィンドウに表示されます。



13. ホストに関連付けるグループを選択し、**Save** をクリックします。
グループが関連付けられている場合、そのグループはホストに関連付けられているグループのリストに表示されます。

14. ホストを使用してジョブを実行した場合は、ホストの **Completed Jobs** タブでそのようなジョブの詳細を表示できます。

15. **Expanded** をクリックして、各ジョブの詳細を表示します。



注記

API に新しく追加されたエンドポイント **/api/v2/bulk/host_create** を使用して、ホストを一括作成できます。このエンドポイントは JSON を受け入れ、ターゲットインベントリーとインベントリーに追加するホストのリストを指定できます。これらのホストはインベントリー内で一意である必要があります。すべてのホストが追加されるか、操作が完了できなかった理由を示したエラーが返されます。**OPTIONS** リクエストを使用して、関連するスキーマを返します。

詳細は、**Automation controller API ガイド**の バルクエンドポイント を参照してください。

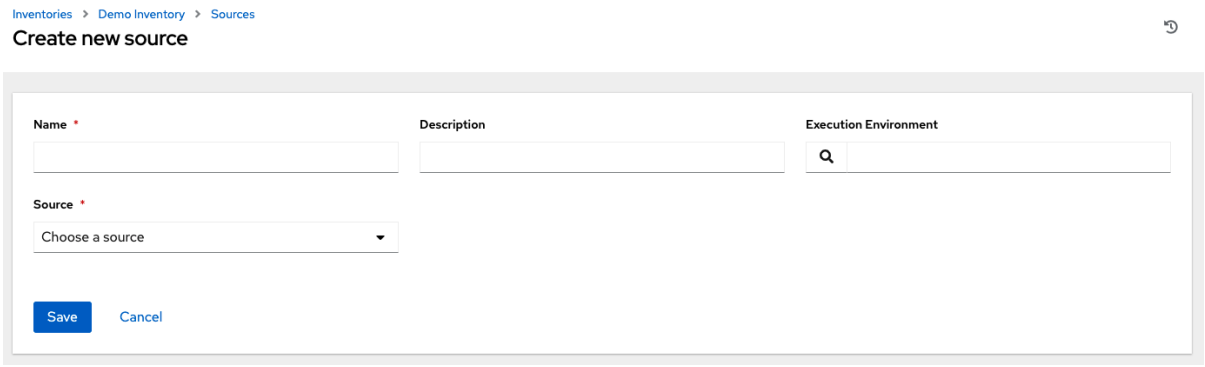
18.4.4. ソースの追加

インベントリーソースはグループに関連付けられません。生成されたグループはトップレベルであり、引き続き子グループを持つことができます。これらの生成されたグループはいずれも、ホストを持つことができます。インベントリーへのソースの追加は、標準インベントリーにのみ適用されます。スマートインベントリーは、関連付けられている標準インベントリーからソースを継承します。

インベントリーのソースを設定するには、次の手順を使用します。

手順

1. ソースを追加するインベントリー名を選択します。
2. インベントリーの **Details** ページで、**Sources** を選択します。
3. **Add** をクリックします。これにより、**Create Source** ウィンドウが開きます。



4. 適切な詳細を入力します。
   - **Name** (必須):
   - オプション: **Description**: 必要に応じて説明を入力します。
   - オプション: **Execution Environment**: 🔍 アイコンを選択するか、インベントリーのインポートを実行する実行環境の名前を入力します。実行環境の構築の詳細は、 実行環境 を参照してください。
   - **Source**: Inventory のソースを選択します。ソースの詳細と適切な情報の提供については、インベントリーソース を参照してください。
5. 選択した インベントリーソース の情報がすべて揃ったら、オプションで、冗長性、ホストフィルター、変数などの他の共通パラメーターを指定できます。
6. **Verbosity** メニューを使用して、インベントリーソースの更新ジョブの出力レベルを選択します。
7. **Host Filter** フィールドを使用して、automation controller. にインポートされる一致するホスト名のみを指定します。
8. **Enabled Variable** フィールドで、Automation controller がホスト変数のディクショナリーから有効な状態を取得することを指定します。ドット表記を使用して有効な変数を 'foo.bar' として指定できます。この場合、ルックアップは **from_dict.get('foo', {}).get('bar', default)** と同等のネストされたディクショナリーを検索します。
9. **Enabled Variable** フィールドでホスト変数のディクショナリーを指定した場合は、インポート時に有効にする値を指定できます。たとえば、次のホスト変数の **enabled_var='status.power_state'** および **'enabled_value='powered_on'** の場合、ホストは **enabled** とマークされます。

```
{
"status": {
"power_state": "powered_on",
"created": "2020-08-04T18:13:04+00:00",
"healthy": true
},
"name": "foobar",
"ip_address": "192.168.2.1"
}
```

power_state が **powered_on** 以外の値の場合、Automation controller へのインポート時にホストは無効になります。キーが見つからない場合、ホストは有効になっています。

10. すべてのクラウドインベントリーソースには、以下の更新オプションがあります。

    - **Overwrite**: チェックされている場合は、外部ソースにこれまで存在していたが現在は削除されているホストとグループが Automation controller インベントリーから削除されます。
      インベントリーソースの管理対象ではなかったホストとグループは、次に手動で作成されたグループにプロモートされます。または、これらをプロモートする手動で作成されたグループがない場合は、インベントリーのすべてのデフォルトグループに残されます。
      チェックが付けられていない場合、外部ソースにないローカルの子ホストおよびグループは、インベントリーの更新プロセスによって処理されないままになります。

    - **Overwrite Variables**: チェックが付けられている場合、子グループとホストのすべての変数が削除され、外部ソースで見つかった変数に置き換えられます。
      チェックが付けられていない場合は、ローカル変数と外部ソースにあるものを組み合わせるマージが実行されます。

    - **Update on Launch**: このインベントリーを使用してジョブを実行するたびに、ジョブタスクの実行前に選択したソースからのインベントリーが更新されます。
      インベントリーの同期よりも早くジョブが生成された場合のジョブのオーバーフローを回避するために、こちらを選択すると、以前のインベントリー同期を一定の秒数キャッシュするための **Cache Timeout** を設定できます。

      Update on Launch 設定は、プロジェクトとインベントリーの依存関係システムを参照しており、2 つのジョブの同時実行を特に除外するものではありません。

      キャッシュタイムアウトが指定されている場合、2 番目のジョブの依存関係が作成され、最初のジョブが生成したプロジェクトとインベントリーの更新が使用されます。

      その後、両方のジョブは、そのプロジェクトまたはインベントリーの更新が完了するのを待ってから続行します。ジョブテンプレートが異なり、システムにその機能がある場合は、両方を同時に開始および実行できます。動的インベントリーソースで Automation controller のプロビジョニングコールバック機能を使用する場合は、インベントリーグループに対して **起動時の更新** を設定する必要があります。

      起動時の更新 が設定されているプロジェクトを使用するインベントリーソースを同期すると、インベントリーの更新が開始される前に、(キャッシュタイムアウトルールに従って) プロジェクトが自動的に更新される可能性があります。

      テンプレートが使用するのと同じプロジェクトから取得するインベントリーを使うように、ジョブテンプレートを作成できます。このような場合、プロジェクトが更新されてからインベントリーが更新されます (更新がまだ進行中でない場合、またはキャッシュタイムアウトがまだ期限切れになっていない場合)。

11. 入力内容と選択内容を確認します。これにより、スケジュールや通知などの追加の詳細を設定できます。

12. このインベントリーソースに関連付けられているスケジュールを設定するには、**Schedules** タブをクリックします。

    - スケジュールがすでに設定されている場合は、スケジュール設定を確認、編集、有効化または無効化します。

    - スケジュールが設定されていない場合、スケジュールの設定の詳細は、スケジュール を参照してください。

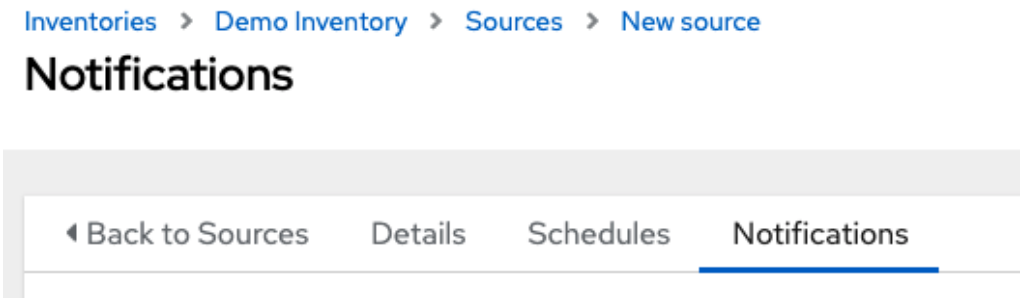
18.4.5. ソースの通知の設定

ソースの通知を設定するには、次の手順を使用します。

1. インベントリーの **Details** ページで、**Notifications** を選択します。

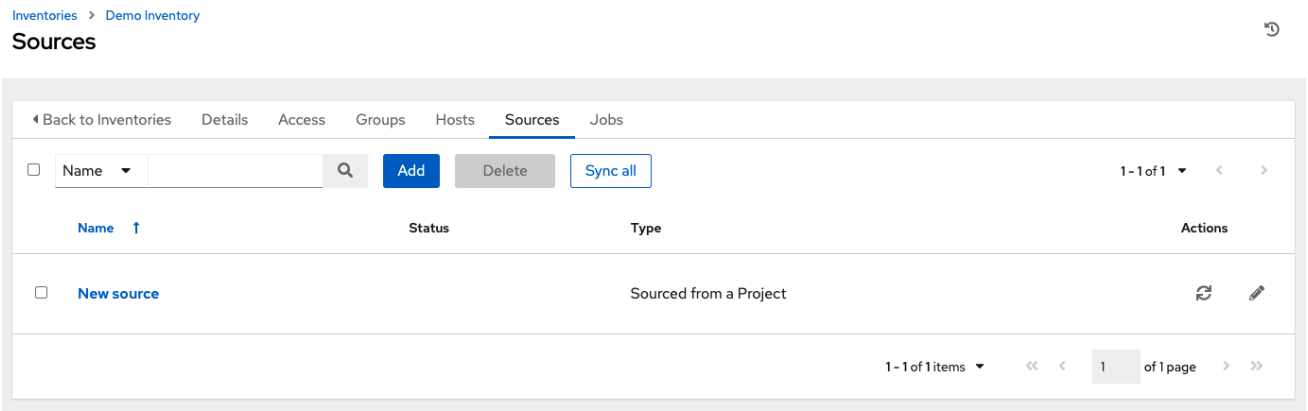
注記

Notifications タブは、新しく作成したソースを保存した場合にのみ表示されます。



2. 通知がすでに設定されている場合は、トグルを使用して、特定のソースで使用する通知を有効または無効にします。詳細は、通知の有効化と無効化 を参照してください。
3. 通知が設定されていない場合、詳細は 通知 を参照してください。
4. 入力内容と選択内容を確認します。
5. **Save** をクリックします。

ソースが定義されると、インベントリーに関連付けられたソースのリストに表示されます。**Sources** タブから、単一のソースに対して同期を実行することも、すべてのソースを一度に同期することもできます。同期プロセスのスケジュール設定など、追加アクションを実行したり、ソースを編集または削除したりすることもできます。



18.4.5.1. インベントリーソース

ホストを入力できるインベントリータイプに一致するソースを選択します。

- プロジェクトからの取得
- Amazon Web Services EC2
- Google Compute Engine
- Microsoft Azure Resource Manager

- VMware vCenter
- Red Hat Satellite 6
- Red Hat Insights
- OpenStack
- Red Hat Virtualization
- Red Hat Ansible Automation Platform

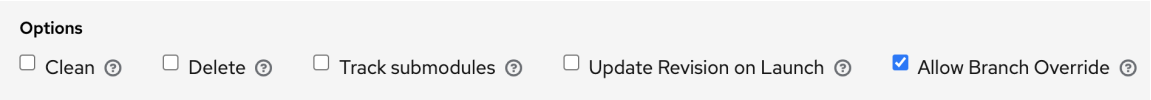
18.4.5.1.1. プロジェクトからの取得

イベントリーをプロジェクトから取得する場合は、紐づけられているプロジェクトの SCM タイプを使用します。たとえば、プロジェクトのソースが GitHub からのものである場合、インベントリーでは同じソースが使用されます。

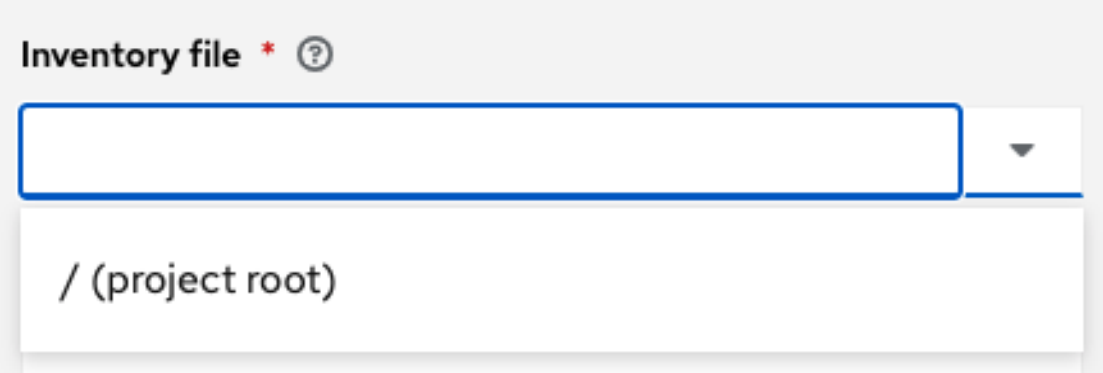
プロジェクトソースのインベントリーを設定するには、次の手順を使用します。

手順

1. **Create new source** ページで、**Source** を選択します。
2. メニューから、**Sourced from a Project** を選択します。
3. ソースの作成ウィンドウは追加フィールドと共にデプロイメントされます。以下の詳細を入力します。
   - オプション: **Source Control Branch/Tag/Commit** チェックアウトするソースコントロール (Git または Subversion) からの SCM ブランチ、タグ、コミットハッシュ、任意の参照、またはリビジョン番号 (該当する場合) を入力します。
     このフィールドは、ソースプロジェクトで **Allow Branch Override** オプションがチェックされている場合にのみ表示されます。詳細は、SCM タイプ - Git および Subversion を参照してください。

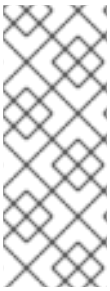
     

     次のフィールドにカスタム refspec も指定しない限り、一部のコミットハッシュと参照は使用できない場合があります。空白のままにした場合、デフォルトは HEAD です。これは、このプロジェクトで最後にチェックアウトされたブランチ/タグ/コミットです。
   - **認証情報**: このソースに使用する認証情報を指定します。
   - **プロジェクト** (必須): デフォルトのプロジェクトが事前に入力されます。それ以外の場合は、このインベントリーがソースとして使用しているプロジェクトを指定します。 アイコンをクリックしてプロジェクトのリストから選択します。リストが膨大な場合は、検索を使用してオプションを絞り込みます。
   - **インベントリーファイル** (必須): 取得したプロジェクトに関連付けられたインベントリーファイルを選択します。まだ入力されていない場合は、メニュー内のテキストフィールドに入力して、無関係なファイルタイプをフィルタリングできます。フラットファイルインベントリーに加えて、ディレクトリーまたはインベントリースクリプトを参照することもできます。



4. オプション: Adding a source で説明されているように、詳細度、ホストフィルター、有効な変数または値、および更新オプションを指定できます。

5. オプション: カスタムインベントリースクリプトに渡すために、**環境変数** フィールドで環境変数を設定できます。インベントリースクリプトをソースコントロールに配置し、プロジェクトから実行することもできます。詳細は、**Automation controller 管理ガイド** の インベントリーファイルのインポート を参照してください。



注記

SCM からカスタムインベントリースクリプトを実行している場合は、アップストリームのソースコントロールでスクリプトの実行ビット (**chmod +x**) を設定していることを確認してください。

そうしないと、Automation controller は実行時に **[Errno 13] Permission denied** エラーが発生します。

18.4.5.1.2. Amazon Web Services EC2

次の手順を使用して、AWS EC2 ソースのインベントリーを設定します。

手順

1. **Create new source** ページで、**Source** を選択します。

2. メニューから **Amazon EC2** を選択します。

3. **Create Source** ウィンドウは追加フィールドと共にデプロイメントされます。以下の詳細を入力します。

   - オプション: **認証情報**: 既存の AWS 認証情報から選択します (詳細は、 認証情報 を参照してください)。
     Automation controller が IAM ロールが割り当てられた EC2 インスタンス上で実行されている場合、認証情報は省略でき、代わりにインスタンスメタデータのセキュリティー認証情報が使用されます。IAM ロールの使用の詳細は、
     IAM_Roles_for_Amazon_EC2_documentation_at_Amazon を参照してください。

4. オプション: Adding a source で説明されているように、詳細度、ホストフィルター、有効な変数または値、および更新オプションを指定できます。

5. **Source Variables** フィールドを使用して、**aws_ec2** インベントリープラグインで使用される変数をオーバーライドします。JSON または YAML 構文を使用して変数を入力します。ラジオボタンを使用して、その 2 つを切り替えます。これらの変数の詳細は、aws インベントリープラグインのドキュメント を参照してください。

注記

include_filters のみを使用する場合、AWS プラグインは常にすべてのホストを返します。これを正しく使用するには、**フィルター** に **or** の最初の条件を設定し、残りの **OR** 条件を **include_filters** のリストに基づいて構築する必要があります。

18.4.5.1.3. Google Compute Engine

Google が提供するインベントリーを設定するには、次の手順を使用します。

手順

1. **Create new source** ページで、**Source** を選択します。
2. メニューから **Google Compute Engine** を選択します。
3. **Create Source** ウィンドウが展開され、必要な **Credential** フィールドが表示されます。既存の GCE 認証情報から選択します。詳細は、Credentials を参照してください。
4. オプション: Adding a source で説明されているように、詳細度、ホストフィルター、有効な変数または値、および更新オプションを指定できます。
5. **Source Variables** フィールドを使用して、**gcp_compute** インベントリープラグインで使用される変数をオーバーライドします。JSON または YAML 構文を使用して変数を入力します。ラジオボタンを使用して、その 2 つを切り替えます。これらの変数の詳細は、gcp_compute インベントリープラグインのドキュメント を参照してください。

18.4.5.1.4. Microsoft Azure Resource Manager

次の手順を使用して、Azure Resource Manager ソースのインベントリーを設定します。

手順

1. **Create new source** ページで、**Source** を選択します。
2. メニューから **Microsoft Azure Resource Manager** を選択します。
3. **Create Source** ウィンドウが展開され、必要な **Credential** フィールドが表示されます。既存の Azure 認証情報から選択します。詳細は、Credentials を参照してください。
4. オプション: Adding a source で説明されているように、詳細度、ホストフィルター、有効な変数または値、および更新オプションを指定できます。
5. **Source Variables** フィールドを使用して、**azure_rm** インベントリープラグインで使用される変数をオーバーライドします。JSON または YAML 構文を使用して変数を入力します。ラジオボタンを使用して、その 2 つを切り替えます。これらの変数の詳細は、azure_rm インベントリープラグインのドキュメント を参照してください。

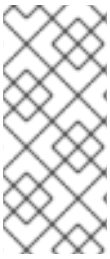
18.4.5.1.5. VMware vCenter

VMWare ソースのインベントリーを設定するには、次の手順を使用します。

手順

1. **Create new source** ページで、**Source** を選択します。

2. メニューから **VMware vCenter** を選択します。
3. **Create Source** ウィンドウが展開され、必要な **Credential** フィールドが表示されます。既存の VMware 認証情報から選択します。詳細は、Credentials を参照してください。
4. オプション: Adding a source で説明されているように、詳細度、ホストフィルター、有効な変数または値、および更新オプションを指定できます。
5. **Source Variables** フィールドを使用して、**vmware_inventory** インベントリープラグインで使用される変数をオーバーライドします。JSON または YAML 構文を使用して変数を入力します。ラジオボタンを使用して、その 2 つを切り替えます。これらの変数の詳細は、vmware_inventory インベントリープラグイン を参照してください。



注記

VMware プロパティーは小文字から camelCase に変更されました。Automation controller はトップレベルのキーのエイリアスを提供しますが、ネストされたプロパティーの小文字キーは廃止されました。サポート対象の有効なプロパティーのリストについては、VMware 動的インベントリープラグインでの仮想マシン属性の使用 を参照してください。

18.4.5.1.6. Red Hat Satellite 6

Red Hat Satellite ソースのインベントリーを設定するには、次の手順を使用します。

手順

1. **Create new source** ページで、**Source** を選択します。
2. メニューから **Red Hat Satellite** を選択します。
3. **Create Source** ウィンドウが展開され、必要な **Credential** フィールドが表示されます。既存の Satellite 認証情報から選択します。詳細は、Credentials を参照してください。
4. オプション: Adding a source で説明されているように、詳細度、ホストフィルター、有効な変数または値、および更新オプションを指定できます。
5. **Source Variables** フィールドを使用して、**foreman** インベントリーソースで使用されるパラメーターを指定します。JSON または YAML 構文を使用して変数を入力します。ラジオボタンを使用して、その 2 つを切り替えます。これらの変数の詳細は、Ansible ドキュメントの Foreman インベントリーソース を参照してください。

automation controller インベントリーで Satellite から関連グループが含まれない場合には、インベントリーソースでこれらの変数を定義する必要がある場合があります。詳細は、Red Hat Satellite 6 を参照してください。

"no foreman.id" variable(s) when syncing the inventory というメッセージが表示された場合は、Red Hat カスタマーポータル (https://access.redhat.com/solutions/5826451) で解決策を参照してください。記事全文にアクセスするには、必ず顧客の認証情報でログインしてください。

18.4.5.1.7. Red Hat Insights

Red Hat Insights ソースのインベントリーを設定するには、次の手順を使用します。

手順

1. **Create new source** ページで、**Source** を選択します。
2. メニューから **Red Hat Insights** を選択します。
3. **Create Source** ウィンドウが展開され、必要な **Credential** フィールドが表示されます。既存の GCE 認証情報から選択します。詳細は、認証情報 を参照してください。
4. オプション: Adding a source で説明されているように、詳細度、ホストフィルター、有効な変数または値、および更新オプションを指定できます。
5. **Source Variables** フィールドを使用して、**gcp_compute** インベントリープラグインで使用される変数をオーバーライドします。JSON または YAML 構文を使用して変数を入力します。ラジオボタンを使用して、その 2 つを切り替えます。これらの変数の詳細は、Insights インベントリープラグイン を参照してください。

18.4.5.1.8. OpenStack

OpenStack ソースのインベントリーを設定するには、次の手順を使用します。

手順

1. **Create new source** ページで、**Source** を選択します。
2. メニューから **Openstack** を選択します。
3. **Create Source** ウィンドウが展開され、必要な **Credential** フィールドが表示されます。既存の GCE 認証情報から選択します。詳細は、認証情報 を参照してください。
4. オプション: Adding a source で説明されているように、詳細度、ホストフィルター、有効な変数または値、および更新オプションを指定できます。
5. **Source Variables** フィールドを使用して、**gcp_compute** インベントリープラグインで使用される変数をオーバーライドします。JSON または YAML 構文を使用して変数を入力します。ラジオボタンを使用して、その 2 つを切り替えます。これらの変数の詳細は、openstack インベントリープラグイン を参照してください。

18.4.5.1.9. Red Hat Virtualization

Red Hat Virtualization ソースのインベントリーを設定するには、次の手順を使用します。

手順

1. **Create new source** ページで、**Source** を選択します。
2. メニューから **Red Hat Virtualization** を選択します。
3. **Create Source** ウィンドウが展開され、必要な **Credential** フィールドが表示されます。既存の GCE 認証情報から選択します。詳細は、Credentials を参照してください。
4. オプション: Adding a source で説明されているように、詳細度、ホストフィルター、有効な変数または値、および更新オプションを指定できます。
5. **Source Variables** フィールドを使用して、**gcp_compute** インベントリープラグインで使用される変数をオーバーライドします。JSON または YAML 構文を使用して変数を入力します。ラジオボタンを使用して、その 2 つを切り替えます。これらの変数の詳細は、ovirt インベントリープラグイン を参照してください。

注記

Red Hat Virtualization (ovirt) インベントリーソースリクエストはデフォルトで安全です。このデフォルト設定を変更するには、**source_variables** で **ovirt_insecure** のキーを **true** に設定します。これは、**/api/v2/inventory_sources/N/** エンドポイントにあるインベントリーソースの API 詳細からのみ使用できます。

18.4.5.1.10. Red Hat Ansible Automation Platform

Automation controller をソースとするインベントリーを設定するには、次の手順を使用します。

手順

1. **Create new source** ページで、**Source** を選択します。
2. メニューから **Red Hat Ansible Automation Platform** を選択します。
3. **Create Source** ウィンドウが展開され、必要な **Credential** フィールドが表示されます。既存の GCE 認証情報から選択します。詳細は、認証情報 を参照してください。
4. オプション: Adding a source で説明されているように、詳細度、ホストフィルター、有効な変数または値、および更新オプションを指定できます。
5. **Source Variables** フィールドを使用して、**gcp_compute** インベントリープラグインで使用される変数をオーバーライドします。JSON または YAML 構文を使用して変数を入力します。ラジオボタンを使用して、その 2 つを切り替えます。これらの変数の詳細は、コントローラーインベントリープラグイン を参照してください。これには、Red Hat カスタマーのログインが必要です。

18.4.5.2. 以前のインベントリースクリプトのエクスポート

カスタムインベントリースクリプト API が削除されたにもかかわらず、スクリプトは引き続きデータベースに保存されます。このセクションで説明するコマンドを使用すると、後でソースコントロールにチェックインするのに適した形式でデータベースからスクリプトを回復できます。

以下のコマンドを使用します。

```
$ awx-manage export_custom_scripts --filename=my_scripts.tar

Dump of old custom inventory scripts at my_scripts.tar
```

出力を活用します。

```
$ mkdir my_scripts
$ tar -xf my_scripts.tar -C my_scripts
```

スクリプトの名前の形式は、**<pk>_ <name>** です。これは、プロジェクトフォルダーに使用される命名スキームです。

```
$ ls my_scripts
10inventory_script_rawhook _19 _30inventory_script_listenhospital _11inventory_script_upperorder
_1inventory_script_commercialinternet45 _4inventory_script_whitestring
_12inventory_script_eastplant _22inventory_script_pinexchange
_5inventory_script_literaturepossession _13inventory_script_governmentculture
_23inventory_script_brainluck _6inventory_script_opportunitytelephone
```

```
_14inventory_script_bottomguess _25inventory_script_buyerleague _7inventory_script_letjury
_15inventory_script_wallisland _26inventory_script_lifesport _8random_inventory_script
16inventory_script_wallisland _27inventory_script_exchangesomewhere _9random_inventory_script
_17inventory_script_bidstory          _28inventory_script_boxchild _18p
_29__inventory_script_wearstress
```

各ファイルにはスクリプトが含まれています。スクリプトは **bash**/**python**/**ruby**/**more** であるため、拡張子は含まれません。これらはすべて直接実行可能です。スクリプトを実行すると、インベントリーデータがダンプされます。

```
$ ./my_scripts/11__inventory_script_upperorder
{"group\ud801\udcb0\uc20e\u7b0e\ud81c\udfeb\ub12b\ub4d0\u9ac6\ud81e\udf07\u6ff9\uc17b":
{"hosts":
["host_\ud821\udcad\u68b6\u7a51\u93b4\u69cf\uc3c2\ud81f\uddbe\ud820\udc92\u3143\u62c7",
"host_\u6057\u3985\u1f60\ufefb\u1b22\ubd2d\ua90c\ud81a\udc69\u1344\u9d15",
"host_\u78a0\ud820\udef3\u925e\u69da\ua549\ud80c\ude7e\ud81e\udc91\ud808\uddd1\u57d6\ud801\
ude57",
"host_\ud83a\udc2d\ud7f7\ua18a\u779a\ud800\udf8b\u7903\ud820\udead\u4154\ud808\ude15\u9711",

"host_\u18a1\u9d6f\u08ac\u74c2\u54e2\u740e\u5f02\ud81d\uddee\ufbd6\u4506"], "vars":
{"ansible_host": "127.0.0.1", "ansible_connection":
"local"}}}
```

ansible-inventory を使用して機能を確認できます。これにより、同じデータが得られますが、再フォーマットされます。

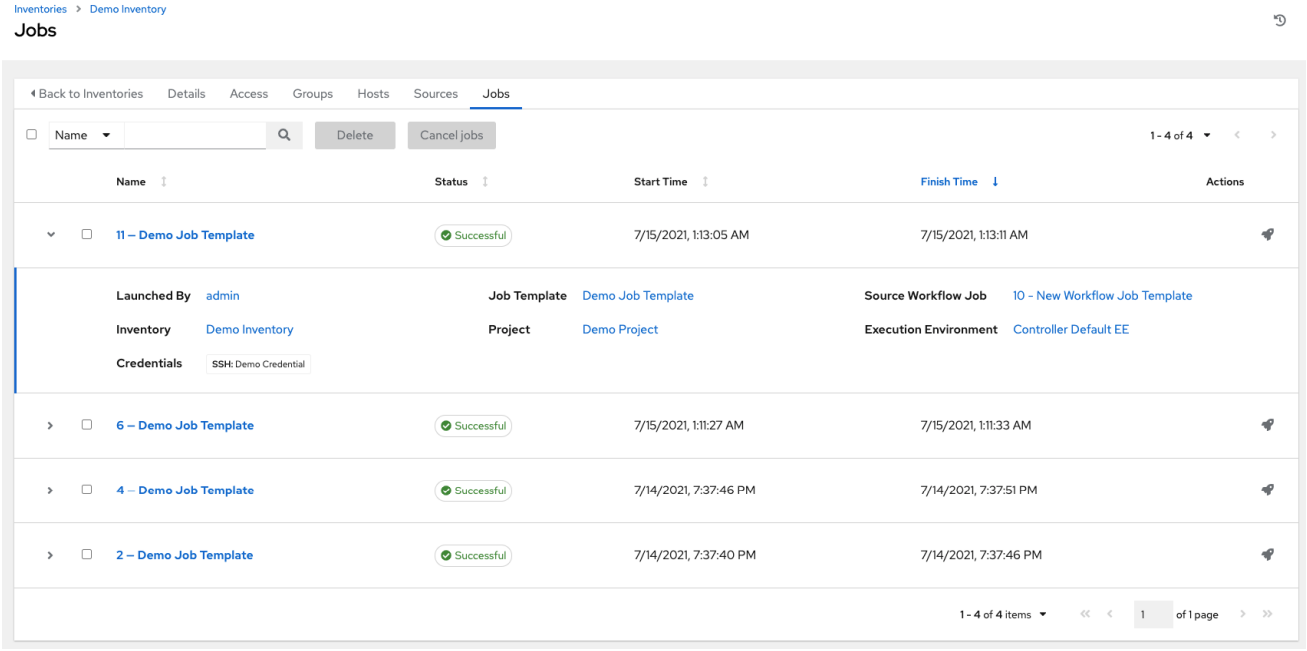
```
$ ansible-inventory -i ./my_scripts/_11__inventory_script_upperorder --list --export
```

前述の例では、**my_scripts** に **cd** で移動して **git init** コマンドを発行し、必要なスクリプトを追加してソースコントロールにプッシュし、ユーザーインターフェイスで SCM インベントリーソースを作成できます。

カスタムインベントリースクリプトの同期または使用の詳細は、**Automation controller 管理ガイド** の インベントリーファイルのインポート を参照してください。

18.5. 完了したジョブの表示

インベントリーがジョブの実行に使用されている場合、このジョブの詳細は、インベントリーの **Completed Jobs** タブで確認できます。各ジョブの詳細を表示するには、**Expanded** をクリックします。



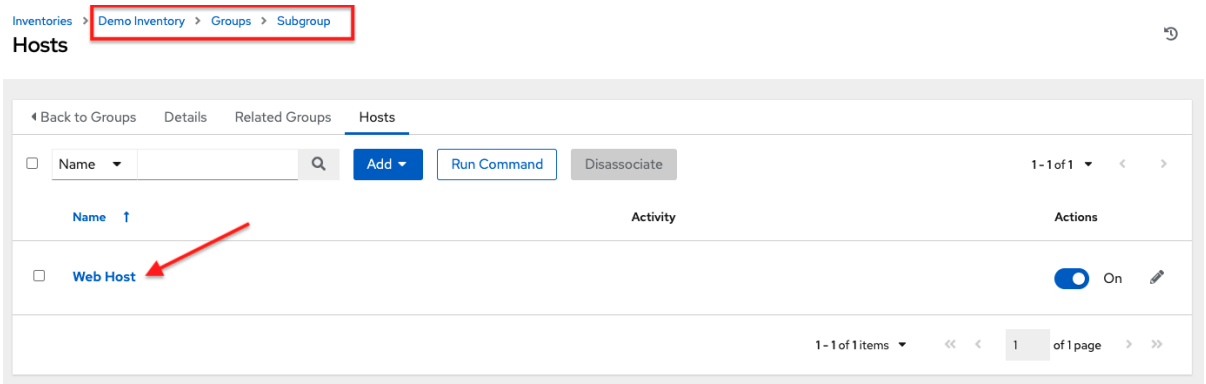
18.6. アドホックコマンドの実行

アドホック とは、オーケストレーション言語 (/usr/bin/ansible-playbook) ではなく、/usr/bin/ansible を使用して Ansible でクイックコマンドを実行することを指します。アドホックコマンドの例としては、インフラストラクチャー内の 50 台のマシンを再起動することが考えられます。アドホックで可能なことは、Playbook に記述して実現できます。Playbook では、他の多くの操作を結合することもできます。

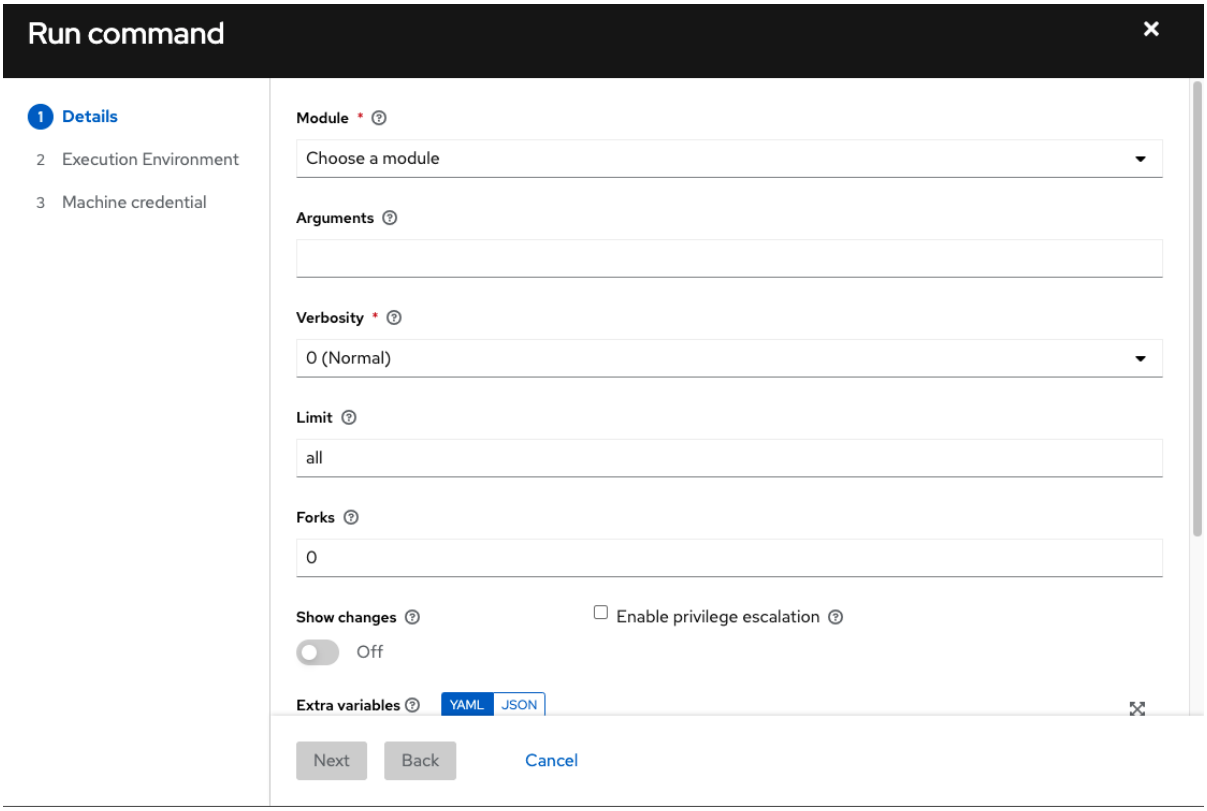
アドホックコマンドを実行するには、次の手順を使用します。

手順

1. ホストまたはグループのリストからインベントリーソースを選択します。インベントリーソースは、単一のグループまたはホスト、選択した複数のホスト、またはグループを指定できます。



2. **Run Command** をクリックします。コマンドの実行ウィンドウが開きます。



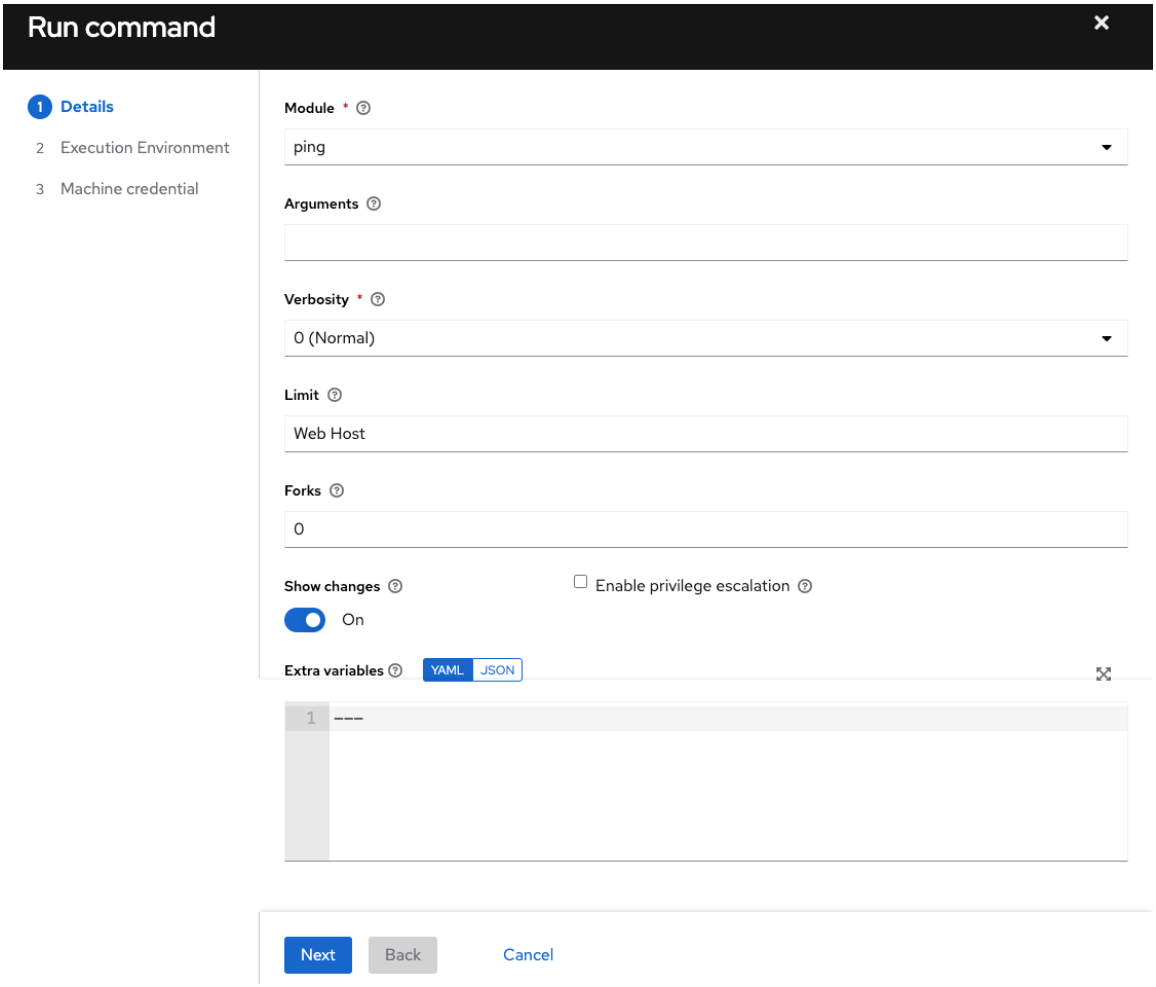
3. 以下の情報を入力します。

- **モジュール**: コマンドの実行での使用をサポートしているモジュールのいずれかを選択します。

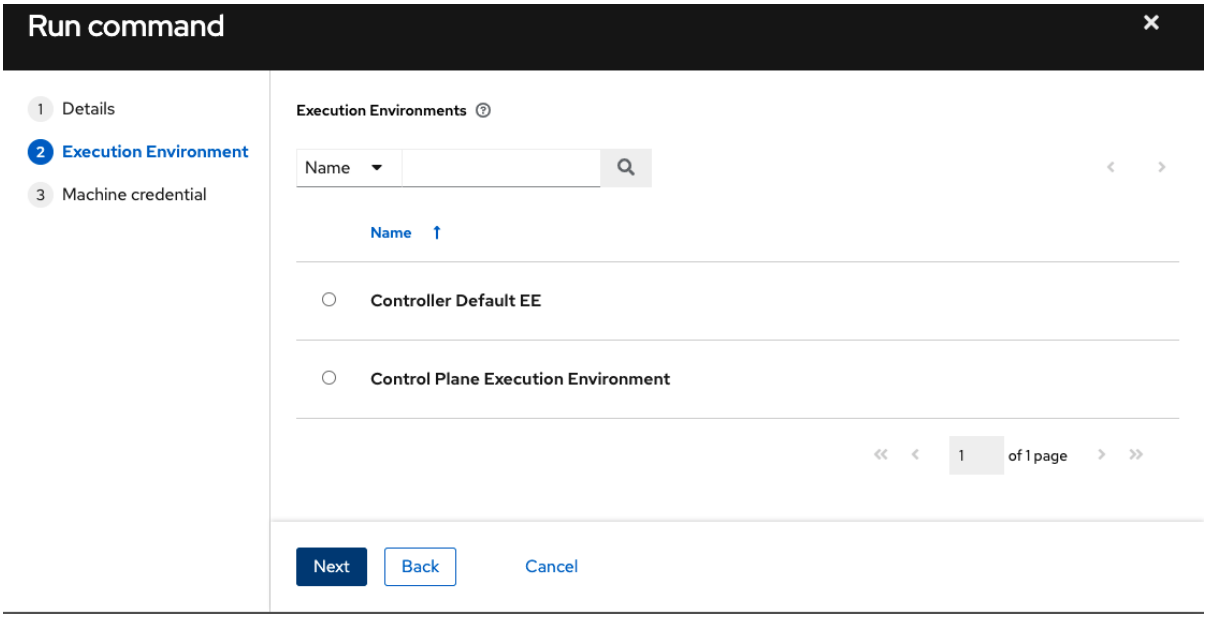
command	apt_repository	mount	win_service
shell	apt_rpm	ping	win_updates
yum	service	selinux	win_group
apt	group	setup	win_user
apt_key	user	win_ping	win_user

- **Arguments**: 選択したモジュールで使用する引数を指定します。
- **Limit**: インベントリー内のターゲットホストに使用される制限を入力します。インベントリー内のすべてのホストをターゲットにするには、**all** または * を入力するか、フィールドを空白のままにします。これには、起動ボタンをクリックする前に、先ほどのビューで選択した内容が自動的に入力されます。
- **Machine Credential**: リモートホストにアクセスしてコマンドを実行するときに使用する認証情報を選択します。Ansible がリモートホストにログインするために必要なユーザー名と SSH 鍵またはパスワードを含む認証情報を選択します。
- **Verbosity**: 標準出力の詳細レベルを選択します。
- **Forks**: 必要な場合には、コマンドの実行中に使用する並列または同時プロセスの数を選択します。

- **Show Changes**: 標準出力での Ansible の変更の表示を有効にする場合に選択します。デフォルトは OFF です。
- **Enable Privilege Escalation**: 有効にすると、Playbook は管理者権限で実行されます。これは、**ansible** コマンドで **--become** オプションを指定することと同じです。
- **追加の変数**: このインベントリーの実行時に適用される追加のコマンドライン変数を提供します。JSON または YAML 構文を使用して変数を入力します。ラジオボタンを使用して、その 2 つを切り替えます。

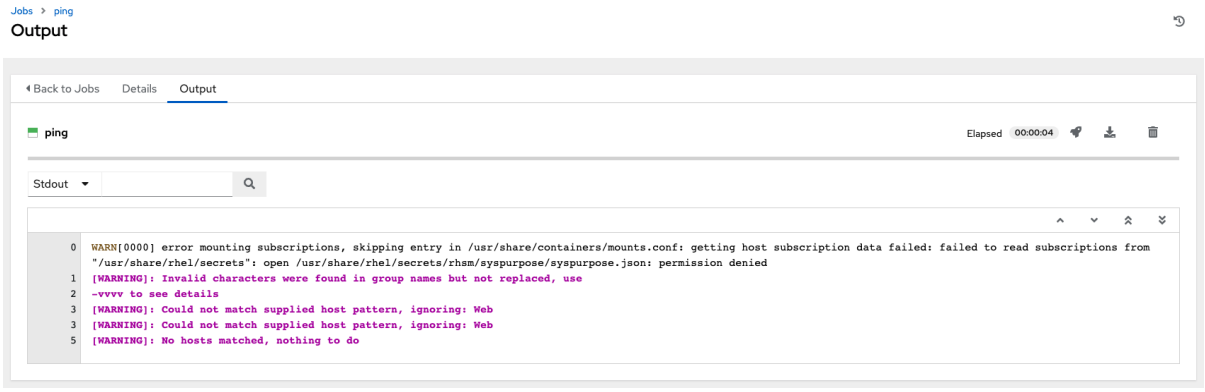


4. **Next** をクリックして、アドホックコマンドを実行する実行環境を選択します。



5. **Next** をクリックして、使用する認証情報を選択します。

6. **Launch** をクリックします。結果は、モジュールのジョブウィンドウの **Output** タブに表示されます。



第19章 サポートされているインベントリープラグインテンプレート

4.x にアップグレードすると、既存の設定は、下位互換性のあるインベントリー出力を生成する新しい形式に移行されます。以下のテンプレートを使用すると、インベントリーを新しいスタイルのインベントリープラグイン出力に移行するのに役に立ちます。

- Amazon Web Services EC2
- Google Compute Engine
- Microsoft Azure Resource Manager
- VMware vCenter
- Red Hat Satellite 6
- OpenStack
- Red Hat Virtualization
- Red Hat Ansible Automation Platform

19.1. AMAZON WEB SERVICES EC2

```
compose:
  ansible_host: public_ip_address
  ec2_account_id: owner_id
  ec2_ami_launch_index: ami_launch_index | string
  ec2_architecture: architecture
  ec2_block_devices: dict(block_device_mappings | map(attribute='device_name') | list |
zip(block_device_mappings | map(attribute='ebs.volume_id') | list))
  ec2_client_token: client_token
  ec2_dns_name: public_dns_name
  ec2_ebs_optimized: ebs_optimized
  ec2_eventsSet: events | default("")
  ec2_group_name: placement.group_name
  ec2_hypervisor: hypervisor
  ec2_id: instance_id
  ec2_image_id: image_id
  ec2_instance_profile: iam_instance_profile | default("")
  ec2_instance_type: instance_type
  ec2_ip_address: public_ip_address
  ec2_kernel: kernel_id | default("")
  ec2_key_name: key_name
  ec2_launch_time: launch_time | regex_replace(" ", "T") | regex_replace("(\+)(\d\d):(\d)(\d)$",
".\g<2>\g<3>Z")
  ec2_monitored: monitoring.state in ['enabled', 'pending']
  ec2_monitoring_state: monitoring.state
  ec2_persistent: persistent | default(false)
  ec2_placement: placement.availability_zone
  ec2_platform: platform | default("")
  ec2_private_dns_name: private_dns_name
  ec2_private_ip_address: private_ip_address
  ec2_public_dns_name: public_dns_name
```

```
  ec2_ramdisk: ramdisk_id | default("")
  ec2_reason: state_transition_reason
  ec2_region: placement.region
  ec2_requester_id: requester_id | default("")
  ec2_root_device_name: root_device_name
  ec2_root_device_type: root_device_type
  ec2_security_group_ids: security_groups | map(attribute='group_id') | list |  join(',')
  ec2_security_group_names: security_groups | map(attribute='group_name') | list |  join(',')
  ec2_sourceDestCheck: source_dest_check | default(false) | lower | string
  ec2_spot_instance_request_id: spot_instance_request_id | default("")
  ec2_state: state.name
  ec2_state_code: state.code
  ec2_state_reason: state_reason.message if state_reason is defined else ""
  ec2_subnet_id: subnet_id | default("")
  ec2_tag_Name: tags.Name
  ec2_virtualization_type: virtualization_type
  ec2_vpc_id: vpc_id | default("")
filters:
  instance-state-name:
  - running
groups:
  ec2: true
hostnames:
  - network-interface.addresses.association.public-ip
  - dns-name
  - private-dns-name
keyed_groups:
  - key: image_id | regex_replace("[^A-Za-z0-9\_]", "_")
    parent_group: images
    prefix: ''
    separator: ''
  - key: placement.availability_zone
    parent_group: zones
    prefix: ''
    separator: ''
  - key: ec2_account_id | regex_replace("[^A-Za-z0-9\_]", "_")
    parent_group: accounts
    prefix: ''
    separator: ''
  - key: ec2_state | regex_replace("[^A-Za-z0-9\_]", "_")
    parent_group: instance_states
    prefix: instance_state
  - key: platform | default("undefined") | regex_replace("[^A-Za-z0-9\_]", "_")
    parent_group: platforms
    prefix: platform
  - key: instance_type | regex_replace("[^A-Za-z0-9\_]", "_")
    parent_group: types
    prefix: type
  - key: key_name | regex_replace("[^A-Za-z0-9\_]", "_")
    parent_group: keys
    prefix: key
  - key: placement.region
    parent_group: regions
    prefix: ''
    separator: ''
  - key: security_groups | map(attribute="group_name") | map("regex_replace", "[^A-Za-z0-9\_]", "_") |
```

```
list
    parent_group: security_groups
    prefix: security_group
  - key: dict(tags.keys() | map("regex_replace", "[^A-Za-z0-9\_]", "_") | list | zip(tags.values()
      | map("regex_replace", "[^A-Za-z0-9\_]", "_") | list))
    parent_group: tags
    prefix: tag
  - key: tags.keys() | map("regex_replace", "[^A-Za-z0-9\_]", "_") | list
    parent_group: tags
    prefix: tag
  - key: vpc_id | regex_replace("[^A-Za-z0-9\_]", "_")
    parent_group: vpcs
    prefix: vpc_id
  - key: placement.availability_zone
    parent_group: '{{ placement.region }}'
    prefix: ''
    separator: ''
plugin: amazon.aws.aws_ec2
use_contrib_script_compatible_sanitization: true
```

19.2. GOOGLE COMPUTE ENGINE

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

```
- key: zone
  prefix: ''
  separator: ''
- key: gce_tags
  prefix: tag
- key: status | lower
  prefix: status
- key: image
  prefix: ''
  separator: ''
plugin: google.cloud.gcp_compute
retrieve_image_info: true
use_contrib_script_compatible_sanitization: true
```

19.3. MICROSOFT AZURE RESOURCE MANAGER

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

19.4. VMWARE VCENTER

```
compose:
  ansible_host: guest.ipAddress
  ansible_ssh_host: guest.ipAddress
  ansible_uuid: 99999999 | random | to_uuid
  availablefield: availableField
  configissue: configIssue
  configstatus: configStatus
  customvalue: customValue
  effectiverole: effectiveRole
  guestheartbeatstatus: guestHeartbeatStatus
  layoutex: layoutEx
  overallstatus: overallStatus
  parentvapp: parentVApp
  recenttask: recentTask
  resourcepool: resourcePool
  rootsnapshot: rootSnapshot
  triggeredalarmstate: triggeredAlarmState
filters:
- runtime.powerState == "poweredOn"
keyed_groups:
- key: config.guestId
  prefix: ''
  separator: ''
- key: '"templates" if config.template else "guests"'
  prefix: ''
  separator: ''
plugin: community.vmware.vmware_vm_inventory
properties:
- availableField
- configIssue
- configStatus
- customValue
- datastore
- effectiveRole
- guestHeartbeatStatus
- layout
- layoutEx
- name
- network
- overallStatus
- parentVApp
- permission
- recentTask
- resourcePool
- rootSnapshot
- snapshot
- triggeredAlarmState
- value
- capability
- config
- guest
- runtime
- storage
- summary
strict: false
with_nested_properties: true
```

19.5. RED HAT SATELLITE 6

```
group_prefix: foreman_
keyed_groups:
- key: foreman['environment_name'] | lower | regex_replace(' ', '') | regex_replace('[^A-Za-z0-9_]', '_') |
regex_replace('none', '')
  prefix: foreman_environment_
  separator: ''
- key: foreman['location_name'] | lower | regex_replace(' ', '') | regex_replace('[^A-Za-z0-9_]', '_')
  prefix: foreman_location_
  separator: ''
- key: foreman['organization_name'] | lower | regex_replace(' ', '') | regex_replace('[^A-Za-z0-9_]', '_')
  prefix: foreman_organization_
  separator: ''
- key: foreman['content_facet_attributes']['lifecycle_environment_name'] | lower | regex_replace(' ', '') |
regex_replace('[^A-Za-z0-9_]', '_')
  prefix: foreman_lifecycle_environment_
  separator: ''
- key: foreman['content_facet_attributes']['content_view_name'] | lower | regex_replace(' ', '') |
regex_replace('[^A-Za-z0-9_]', '_')
  prefix: foreman_content_view_
  separator: ''
legacy_hostvars: true
plugin: theforeman.foreman.foreman
validate_certs: false
want_facts: true
want_hostcollections: false
want_params: true
```

19.6. OPENSTACK

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

19.7. RED HAT VIRTUALIZATION

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

```
- fqdn
ovirt_insecure: false
plugin: ovirt.ovirt.ovirt
```

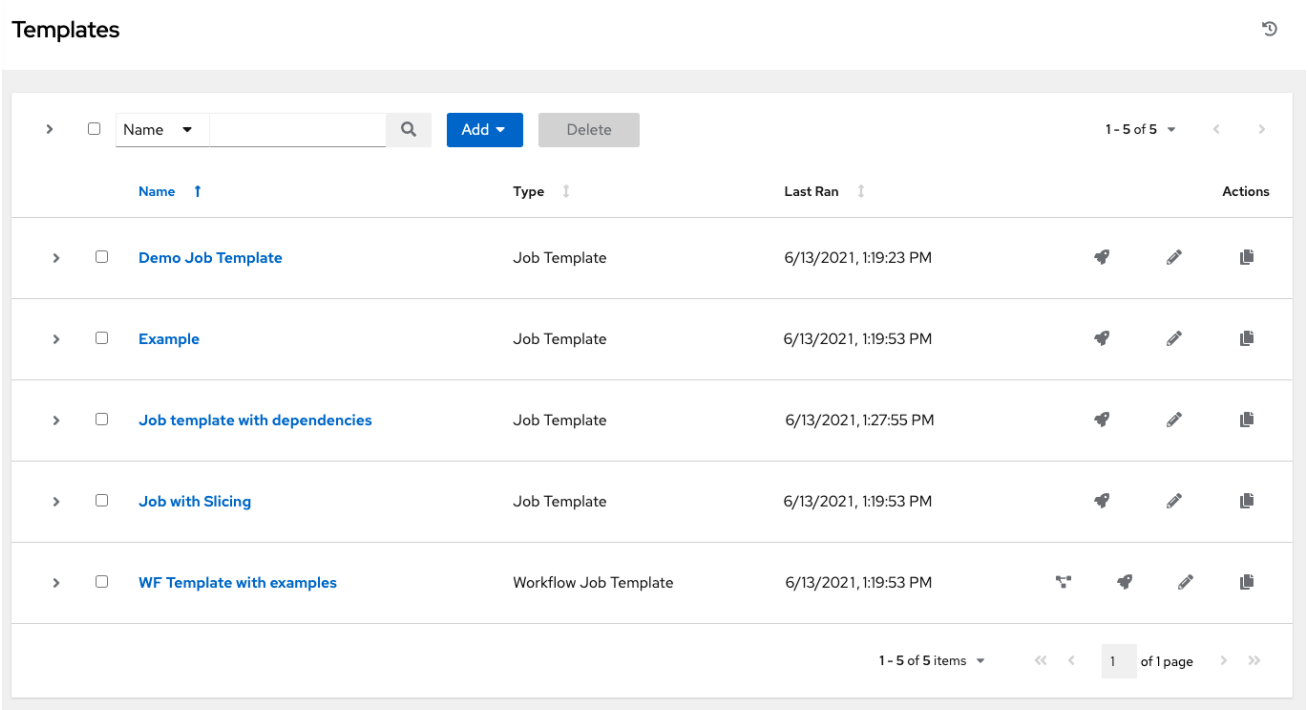
19.8. RED HAT ANSIBLE AUTOMATION PLATFORM

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

第20章 ジョブテンプレート

ジョブテンプレートは、Ansible ジョブを実行するための定義とパラメーターのセットです。ジョブテンプレートは、同じジョブを何度も実行する場合に役立ちます。また、Ansible Playbook コンテンツの再利用とチーム間のコラボレーションも促進します。

Templates リストビューには、現在利用可能なジョブテンプレートが表示されます。デフォルトのビューは折りたたまれており (コンパクト)、テンプレート名、テンプレートタイプ、およびそのテンプレートを使用して実行された最後のジョブのタイムスタンプが表示されます。各エントリーの横にある矢印 アイコンをクリックして、展開し、詳細情報を表示します。このリストは名前のアルファベット順に並べ替えられていますが、他の条件で並べ替えたり、テンプレートのさまざまなフィールドや属性で検索したりできます。



この画面から、ワークフローのジョブテンプレートを起動して ( )、編集し ( )、コピー ( ) します。

注記

ジョブテンプレートを使用して、ワークフローテンプレートを構築できます。それらの隣にある **Workflow Visualizer** アイコンを表示するテンプレートは、ワークフローテンプレートです。アイコンをクリックすると、ワークフローをグラフィカルに構築できます。ジョブテンプレートの多くのパラメーターで、**Prompt on Launch** を選択できます。これはワークフローレベルで変更でき、ジョブテンプレートレベルで割り当てられた値には影響しません。手順は、ワークフロービジュアライザー セクションを参照してください。

20.1. ジョブテンプレートの作成

手順

1. **Templates** リストビューで、**Add → Add job template** の順にクリックします。
2. 次のフィールドに適切な詳細を入力します。

注記

フィールドで **Prompt on launch** を表示するチェックボックスがオンになっている場合、ジョブを起動すると、起動時にそのフィールドの値の入力を求めるプロンプトが表示されます。ほとんどのプロンプト値は、ジョブテンプレートに設定されている値をオーバーライドします。次の表に例外を示します。

フィールド	オプション	Prompt on Launch
Name	ジョブの名前を入力します。	該当なし
Description	必要に応じて、任意の説明を入力します (オプション)。	該当なし
Job Type	ジョブタイプを選択します。 ● Run: 起動時に Playbook を開始し、選択したホストで Ansible タスクを実行します。 ● Check: Playbook の "ドライラン" を実行します。実際に変更を加えずに、変更内容を報告します。チェックモードをサポートしていないタスクは無視され、変更予定の内容は報告されません。 ジョブタイプの詳細は、Ansible ドキュメントの Playbook セクションを参照してください。	はい
Inventory	ログインしているユーザーが使用できるインベントリーから、このジョブテンプレートで使用するインベントリーを選択します。 システム管理者は、ジョブテンプレート内の特定のインベントリーを使用できるように、ユーザーまたはチームのパーミッションを割り当てる必要があります。	必須です。 インベントリーのプロンプトは、後続のプロンプトウィンドウに独自のステップとして表示されます。
Project	ログインしているユーザーが利用できるプロジェクトから、このジョブテンプレートで使用するプロジェクトを選択します。	該当なし

フィールド	オプション	Prompt on Launch
SCM branch	このフィールドは、ブランチの上書きを許可するプロジェクトを選択した場合にのみ表示されます。ジョブの実行で使用する上書きブランチを指定します。空白のままにすると、プロジェクトから指定された SCM ブランチ (またはコミットハッシュまたはタグ) が使用されます。 詳細は、ジョブブランチのオーバーライド を参照してください。	はい
Execution Environment	このジョブの実行に使用するコンテナーイメージを選択します。実行環境を選択する前にプロジェクトを選択する必要があります。	必須です。 実行環境のプロンプトは、後続のプロンプトウィンドウに独自のステップとして表示されます。
Playbook	使用可能な Playbook から、このジョブテンプレートで起動する Playbook を選択します。このフィールドには、選択したプロジェクトのプロジェクトベースパスにある Playbook の名前が自動的に入力されます。あるいは、Playbook がリストされていない場合は、その Playbook での実行に使用するファイル (foo.yml など) の名前など、Playbook の名前を入力することもできます。無効なファイル名を入力すると、テンプレートにエラーが表示されるか、ジョブが失敗します。	該当なし

フィールド	オプション	Prompt on Launch
Credentials	 アイコンを選択すると、別のウィンドウが開きます。 利用可能なオプションから、このジョブテンプレートで使用する認証情報を選択します。 リストが膨大な場合は、ドロップダウンメニューリストを使用して認証情報タイプでフィルタリングします。一部の認証情報タイプは、特定のジョブテンプレートに適用されないため、リストされていません。	● 選択した場合、デフォルトの認証情報が含まれるジョブテンプレートが起動したときに、別の認証情報を指定すると、デフォルトの認証情報が置き換えられます (認証情報が同じタイプの場合)。以下はこのメッセージの例です。 **Job Template default credentials must be replaced with one of the same type.Please select a credential for the following types in order to proceed: Machine.** ● あるいは、必要に応じて認証情報をさらに追加することもできます。 ● 認証情報のプロンプトは、後続のプロンプトウィンドウに独自のステップとして表示されます。

フィールド	オプション	Prompt on Launch
Labels	● 必要に応じて、このジョブテンプレートを説明するラベル (**dev** や **test** など) を指定します。 ● ラベルを使用して、表示内のジョブテンプレートと完了したジョブをグループ化およびフィルタリングします。 ● ラベルはジョブテンプレートの追加時に作成されます。ラベルは、ジョブテンプレートで指定されたプロジェクトを使用して、単一の組織に関連付けられます。組織のメンバーは、編集パーミッション (管理者ロールなど) を持っている場合、ジョブテンプレートにラベルを作成できます。 ● ジョブテンプレートを保存すると、Expanded ビューの **Job Templates** の概要にラベルが表示されます。 ● ラベルの横にある ✖ を選択して削除します。ラベルが削除されると、そのラベルはその特定のジョブまたはジョブテンプレートとの関連付けが解除されますが、ラベルを参照する他のジョブには関連付けられたままになります。 ● ジョブは、起動時にジョブテンプレートからラベルを継承します。ジョブテンプレートからラベルを削除すると、ジョブからも削除されます。	● 選択すると、デフォルト値が指定されている場合でも、必要に応じて起動時に追加のラベルを指定するようにプロンプトが表示されます。 ● 既存のラベルは削除されません。つまり、✖ を選択すると、、新たに追加したラベルだけ削除され、気温のデフォルトラベルは削除されません。

フィールド	オプション	Prompt on Launch
Variables	● 追加のコマンドライン変数を Playbook に渡します。これは、ansible-playbook の -e または --extra-vars コマンドラインパラメーターで、これについては、Ansible Tower ドキュメント (Defining variables at runtime) に説明されています。 ● YAML または JSON を使用してキーまたは値のペアを提供します。これらの変数には優先順位の最大値があり、他の場所で指定された他の変数をオーバーライドします。**git_branch: production release_version: 1.5** は、値の例です。	必須です。 スケジュールで **extra_vars** を指定できるようにするには、ジョブテンプレートの変数に対して **Prompt on launch** を選択するか、ジョブテンプレートで Survey を有効にする必要があります。回答された Survey の質問は **extra_vars** になります。
Forks	Playbook の実行中に使用する並列または同時プロセスの数です。値 0 は、**/etc/ansible/ansible.cfg** でオーバーライドされない限り、Ansible のデフォルト設定 (5 つの並列プロセス) を使用します。	はい

フィールド	オプション	Prompt on Launch
Limit	Playbook が管理または影響を与えるホストのリストをさらに制限するためのホストのパターンを指定します。複数のパターンをコロン (:) で区切ることができます。コア Ansible と同じです。 ● a:b は a または b のグループに含まれるという意味です。 ● a:b:&c は、a または b に含まれるが、c には必ず含まれなければならないという意味です。 ● a:!b は a にはあるが、b には絶対にないという意味です。 詳細は、Ansible ドキュメントの Patterns: targeting hosts and groups を参照してください。	はい 選択されていない場合、ジョブテンプレートがインベントリー内のすべてのノードに対して実行されるか、または **Limit** フィールドで事前定義されたノードに対してのみ実行されます。ワークフローの一部として実行する場合は、代わりにワークフロージョブテンプレートの制限が使用されます。
Verbosity	Playbook の実行時に Ansible が生成する出力レベルを制御します。Normal やさまざまな Verbose または Debug 設定から詳細度を選択します。これは **details** レポートビューにのみ表示されます。詳細ログには、すべてのコマンドの出力が含まれます。デバッグログは非常に詳細で、一部のサポート事例に役立つ SSH 操作に関する情報が含まれています。 詳細の値が **5** の場合、automation controller はジョブの実行時にブロックを実行し、これによりジョブが終了したことを示すレポートが遅延するため (レポートが作成されている場合でも)、ブラウザータブがロックアップする可能性があります。	はい

フィールド	オプション	Prompt on Launch
Job Slicing	このジョブテンプレートを実行するスライスの数を指定します。各スライスは、インベントリーの一部に対して同じタスクを実行します。ジョブスライスの詳細は、ジョブスライス を参照してください。	はい
Timeout	ジョブがキャンセルされるまでの実行時間 (秒単位) を指定できます。タイムアウト値を設定するには、次の点を考慮してください。 ● 設定にはグローバルタイムアウトが定義されており、デフォルトは 0 で、タイムアウトがないことを示します。 ● ジョブテンプレートの負のタイムアウト (<0) は、ジョブのタイムアウトはありません。 ● ジョブテンプレートのタイムアウトが 0 の場合、ジョブはデフォルトでグローバルタイムアウトになります (デフォルトではタイムアウトなし)。 ● 正のタイムアウトは、そのジョブテンプレートのタイムアウトを設定します。	はい
Show Changes	Ansible タスクによって加えられた変更を確認できます。	はい

フィールド	オプション	Prompt on Launch
Instance Groups	このジョブテンプレートに関連付ける Instance および Container Groups を選択します。リストが膨大な場合は、  アイコンを使用してオプションを絞り込みます。ジョブテンプレートのインスタンスグループは、ジョブのスケジュール基準に影響します。ジョブランタイムの動作 と ジョブの実行場所の制御 でルールを確認してください。システム管理者は、ジョブテンプレート内のインスタンスグループを使用できるように、ユーザーまたはチームのパーミッションを割り当てる必要があります。コンテナーグループを使用するには管理者権限が必要です。	• 必須です。 選択すると、ジョブの優先インスタンスグループが優先順に提供されます。最初のグループの容量が不足している場合は、容量のあるグループが見つかるまでリスト内の後続のグループが検討され、見つかった時点でそのグループが選択されてジョブが実行されます。 • インスタンスグループの入力を求めるプロンプトが表示された場合、入力した内容が通常のインスタンスグループ階層を置き換え、組織とインベントリーのすべてのインスタンスグループをオーバーライドします。 • インスタンスグループのプロンプトは、後続のプロンプトウィンドウに独自のステップとして表示されます。
Job Tags	**Create** メニューを入力して選択し、Playbook のどの部分を実行するかを指定します。詳細と例は、Ansible ドキュメントの Tags を参照してください。	はい
Skip Tags	**Create** メニューを入力して選択し、スキップする Playbook の特定のタスクまたは部分を指定します。詳細と例は、Ansible ドキュメントの Tags を参照してください。	はい

3. 必要に応じて、このテンプレートを起動するための次の **Options** を指定します。

   - **Privilege Escalation**: オンにすると、この Playbook を管理者として実行できるようになります。これは、**--become** オプションを **ansible-playbook** コマンドに渡すことと同じです。

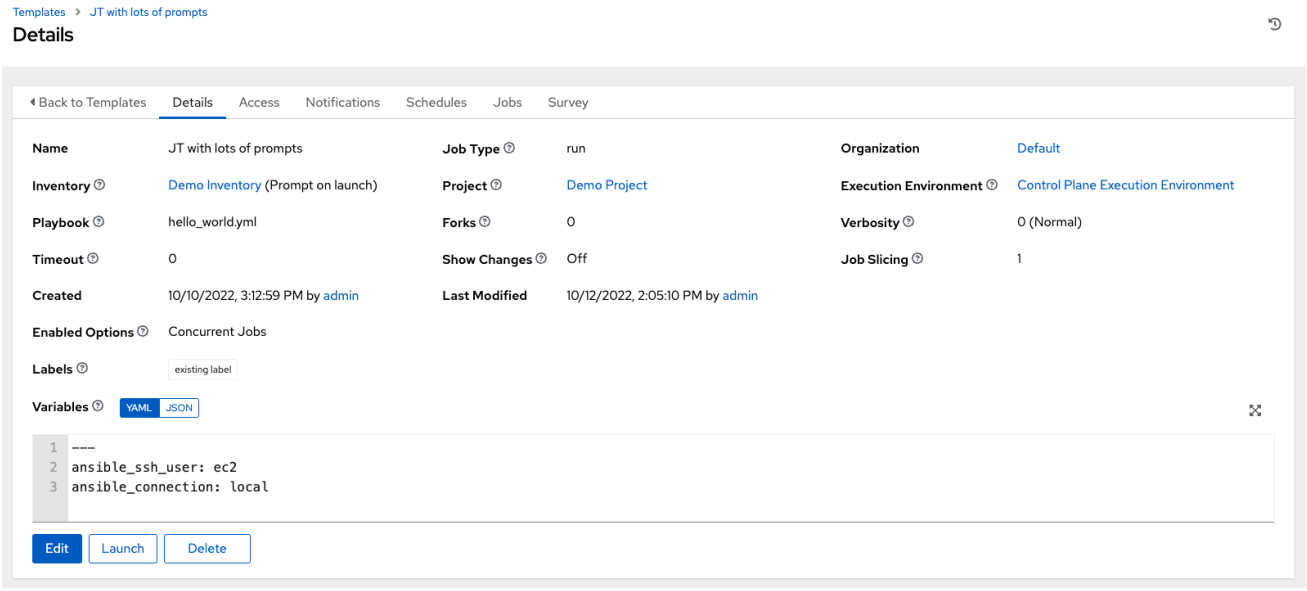
   - **Provisioning Callbacks**: オンにすると、ホストが REST API 経由で Automation Controller にコールバックし、このジョブテンプレートからジョブを起動できるようになります。詳細は、プロビジョニングコールバック を参照してください。

- **Webhook の有効化**: チェックすると、ジョブテンプレートの起動に使用される事前定義された SCM システム Web サービスとのインターフェイス機能が有効になります。GitHub と GitLab がサポートされている SCM システムです。
    - Webhook を有効にすると、他のフィールドが表示され、以下の追加情報の入力を求められます。

    - **Webhook Service**: Webhook からリッスンするサービスを選択します
    - **Webhook URL**: POST 要求を送信する Webhook サービスの URL が自動的に入力されます。
    - **Webhook キー**: Webhook サービスが automation controller に送信するペイロードに署名する際に使用するための、生成された共有シークレット。Automation controller がこのサービスから Webhook を受け入れるようにするには、Webhook サービスの設定で指定する必要があります。
    - **Webhook Credential** : オプションで、Webhook サービスにステータス更新を送信するために使用する認証情報として GitHub または GitLab パーソナルアクセストークン (PAT) を指定します。選択する前に、認証情報が存在している必要があります。認証情報を作成するには、認証情報タイプ を参照してください。
    - Webhook の設定の関連情報については、Webhook の使用 を参照してください。
- **Concurrent Jobs**: オンにすると、キュー内のジョブが相互に依存していない場合に同時に実行されるようになります。ジョブスライスを同時に実行する場合は、このボックスをオンにします。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。
- **Enable Fact Storage** オンにすると、ジョブの実行に関連するインベントリー内のすべてのホストについて収集されたファクトを Automation Controller が保存します。
- **Prevent Instance Group Fallback** このオプションをオンにすると、**Instance Groups** フィールドにリストされているインスタンスグループのみがジョブを実行できるようになります。オフにすると、ジョブの実行場所の制御 で説明されている階層に基づいて、実行プール内の使用可能なすべてのインスタンスが使用されます。

4. ジョブテンプレートの詳細の設定が完了したら、**Save** をクリックします。

テンプレートを保存してもジョブテンプレートページは終了せず、**Job Template Details** タブに進みます。テンプレートを保存した後、**Launch** をクリックしてジョブを起動するか、**Edit** をクリックしてテンプレートの属性 (パーミッション、通知など) を追加または変更したり、完了したジョブを表示したり、Survey を追加したりできます (ジョブタイプがスキャンでない場合))。起動する前にまずテンプレートを保存する必要があります。保存しないと、**Launch** が無効のままになります。



検証

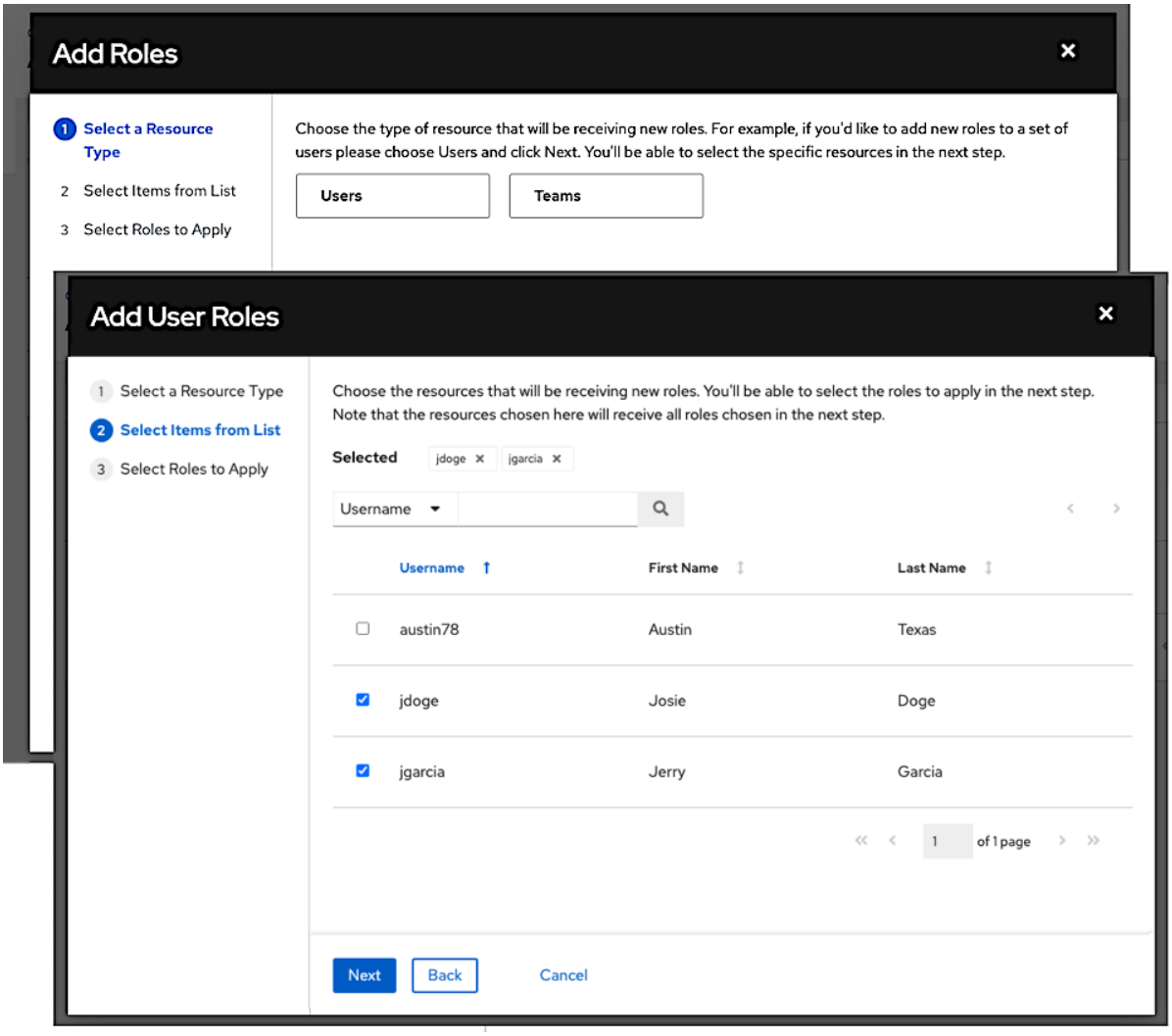
1. ナビゲーションパネルから、**Resources** → **Templates** を選択します。
2. 新しく作成したテンプレートが **Templates** リストビューに表示されることを確認します。

20.2. テンプレートへのパーミッションの追加

次の手順に従って、チームのパーミッションを追加します。

手順

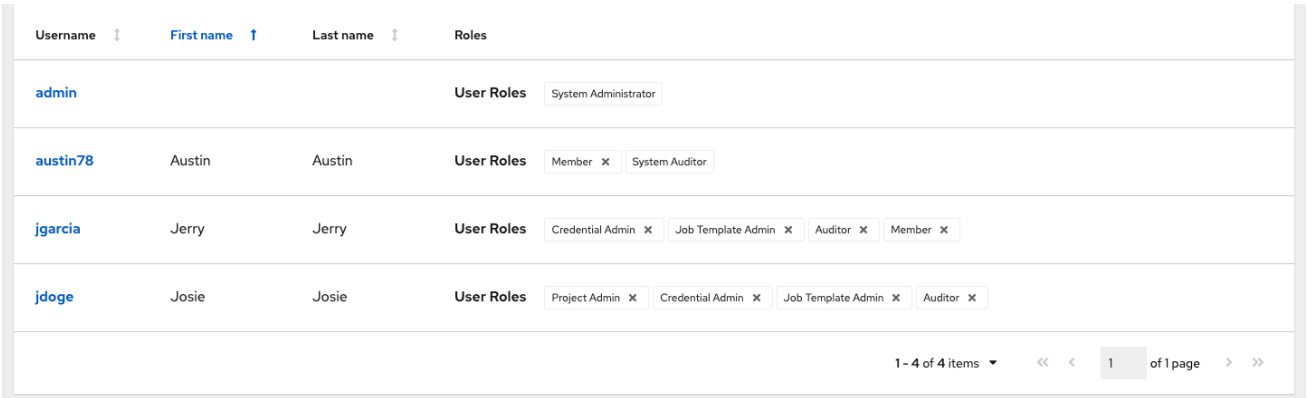
1. ナビゲーションパネルから、**Resources** → **Templates** を選択します。
2. テンプレートを選択し、**Access** tab で **Add** をクリックします。
3. **Users** または **Teams** を選択し、**次へ** をクリックします。
4. リストから 1 つまたは複数のユーザーを選択します。これには、メンバーとして追加するユーザーの隣にあるチェックボックスをクリックして、**Next** をクリックします。
   次の例は、ユーザーが 2 つ追加対象として選択されていることを示しています。



5. ユーザーまたはチームに割り当てるロールを選択します。ロールの完全なリストを確認するには、必ず下にスクロールしてください。各リソースには、使用可能なさまざまなオプションがあります。

6. **Save** をクリックして、選択したユーザーまたはチームにロールを適用し、メンバーとして追加します。

ユーザーとチームを追加するウィンドウが閉じて、各ユーザーとチームに割り当てられた更新されたロールが表示されます。



特定のユーザーのロールを削除するには、リソースの横にある ✖ アイコンをクリックします。

これにより確認ダイアログが起動し、関連付けの解除を確定するように求められます。

20.3. ジョブテンプレートの削除

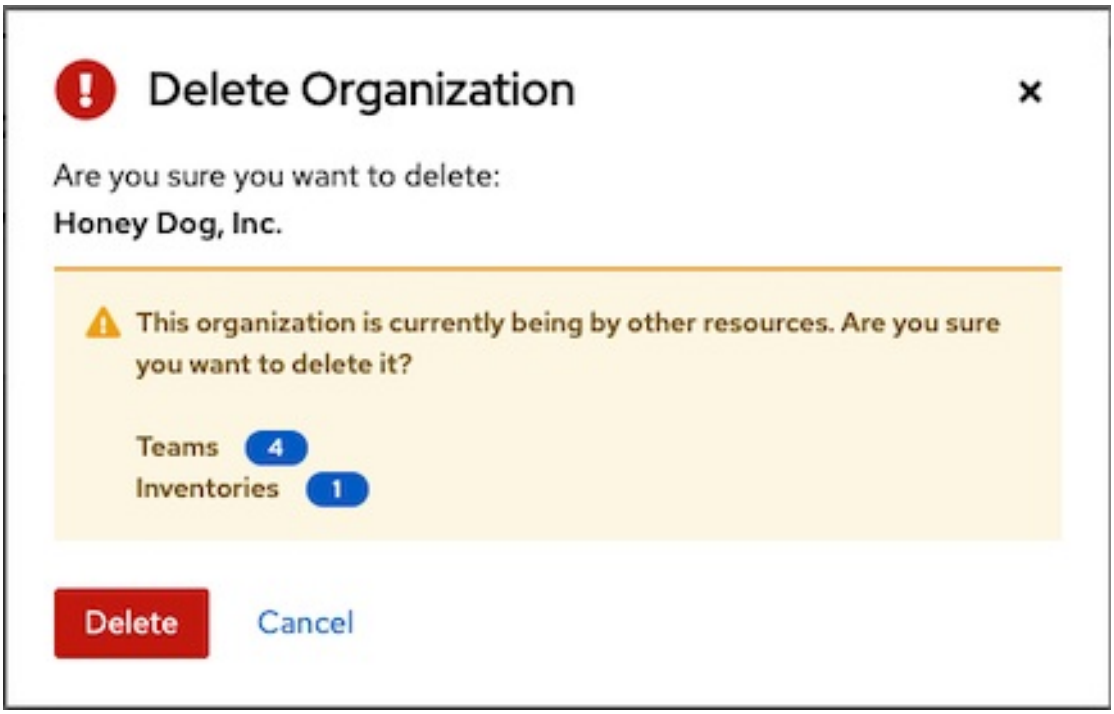
ジョブテンプレートを削除する前に、そのジョブテンプレートがワークフロージョブテンプレートで使用されていないことを確認してください。

手順

1. 次のいずれかの方法を使用して、ジョブテンプレートを削除します。
   - 1 つ以上のジョブテンプレートの横にあるチェックボックスを選択し、**Delete** をクリックします。
   - 目的のジョブテンプレートをクリックし、Details ページで **Delete** をクリックします。

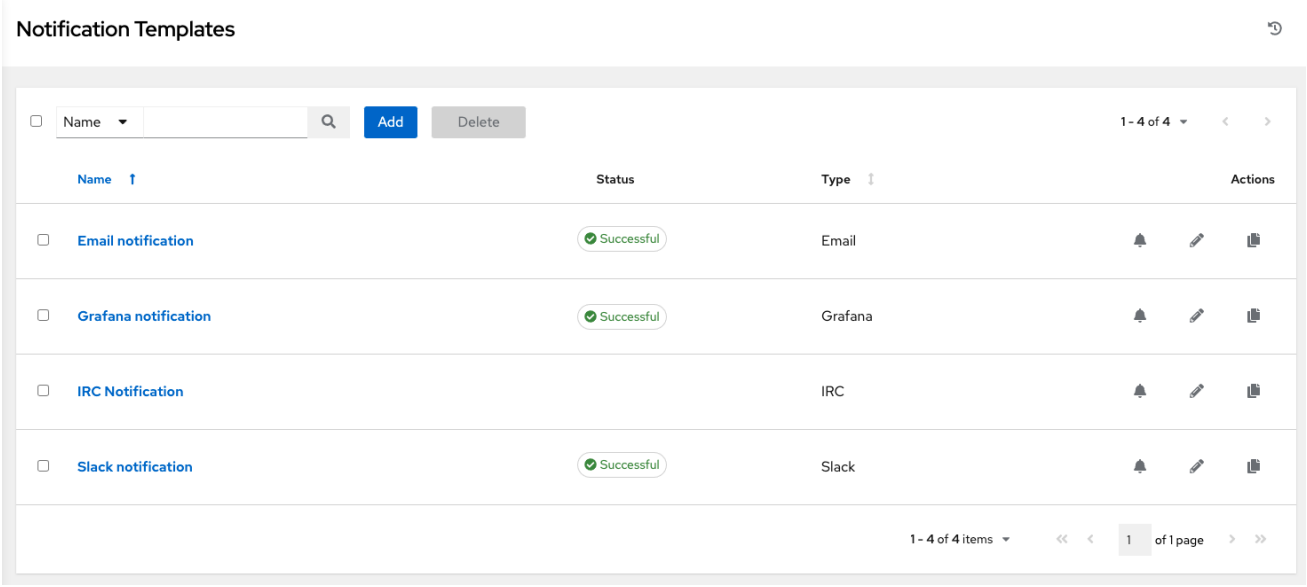
注記

他の作業項目で使用されている項目を削除すると、削除の影響を受ける項目がリストされ、削除の確認を求めるメッセージが表示されます。一部の画面には、無効な項目または以前に削除された項目が含まれており、実行に失敗します。以下はそのメッセージの例です。



20.4. 通知の使用

ナビゲーションパネルから、**Administration → Notifications** を選択します。これにより、設定した通知統合とそのステータス (実行されている場合) を確認できます。



トグルを使用して、特定のテンプレートで使用する通知を有効または無効にします。詳細は、通知の有効化と無効化 を参照してください。

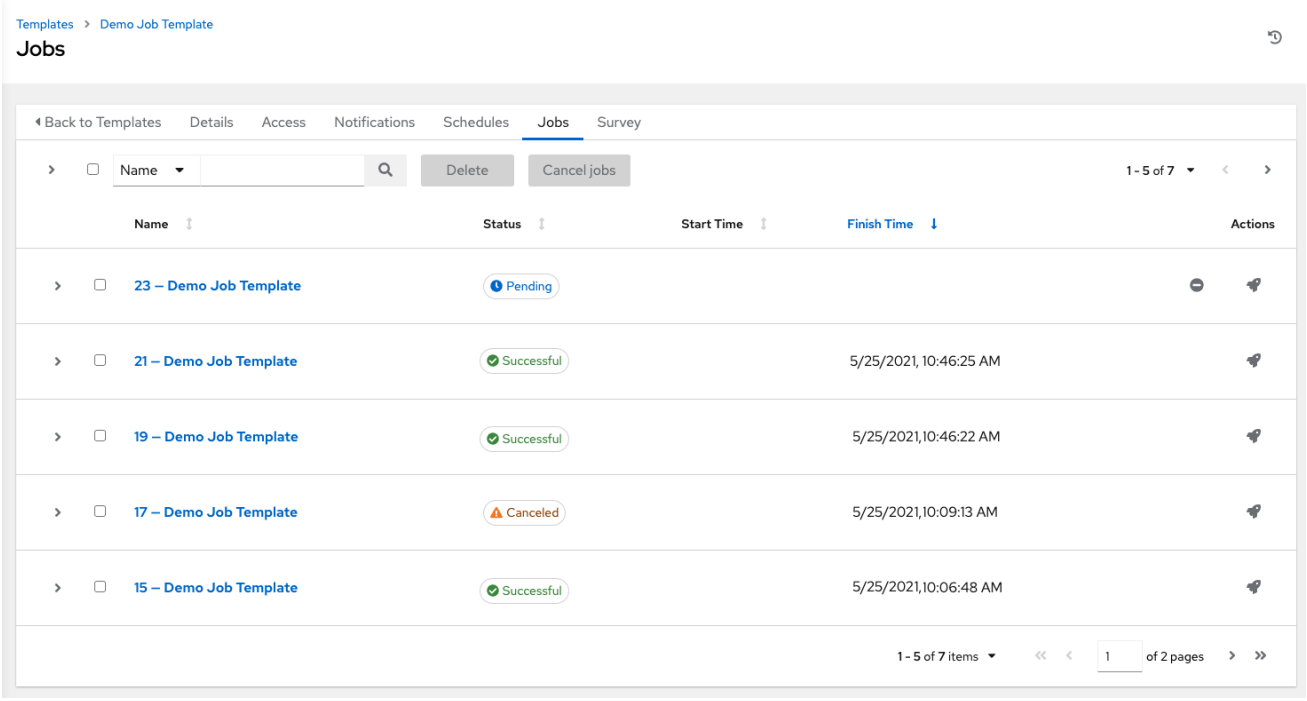
通知が設定されていない場合は、**Add** をクリックして新しい通知を作成します。さまざまな通知タイプと拡張メッセージングの設定の詳細は、通知タイプ を参照してください。

20.5. 完了したジョブの表示

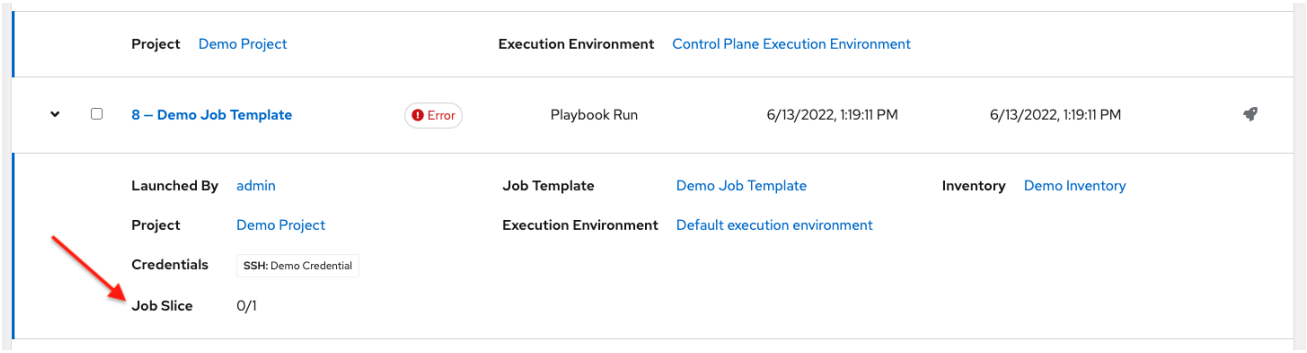
Jobs タブには、実行されたジョブテンプレートのリストが表示されます。各ジョブの横にあるデプロイメントアイコンをクリックすると、次の詳細が表示されます。

- ステータス
- ID および名前
- ジョブの種類
- 開始時間および終了時間
- ジョブの開始者、使用したテンプレート、インベントリー、プロジェクト、認証情報

これらの基準のいずれかを使用して、完了したジョブのリストをフィルタリングできます。

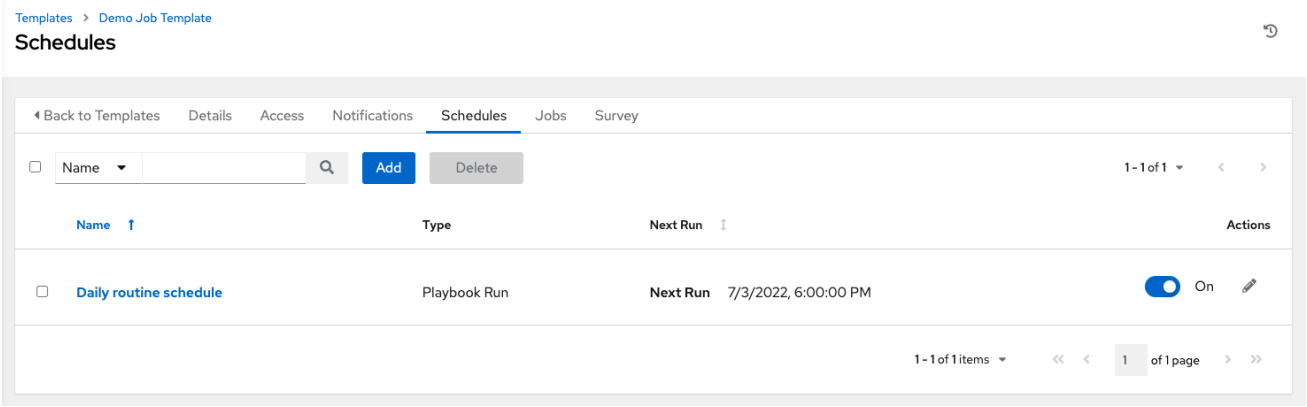


このリストに表示スライスされたジョブは適宜、実行したスライスジョブ数でラベルが付けられます。



20.6. ジョブテンプレートのスケジュール設定

Schedules タブから特定のジョブテンプレートのスケジュールにアクセスします。



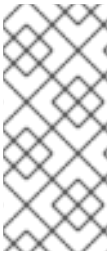
手順

- ジョブテンプレートをスケジュールするには、**Schedules** タブを選択し、適切な方法を選択します。

- スケジュールがすでに設定されている場合は、スケジュール設定を確認、編集、有効化または無効化します。
- スケジュールが設定されていない場合の詳細は スケジュール を参照してください。

Credentials フィールド で**Prompt on Launch** を選択し、ジョブテンプレートのスケジュール情報を作成または編集すると、スケジュールフォームに **Prompt** オプションが表示されます。

Prompt ダイアログでデフォルトのマシン認証情報を削除するには、保存する前に別のマシン認証情報に置き換える必要があります。



注記

スケジュールに **extra_vars** を設定するには、ジョブテンプレートで **Variables** の **Prompt on Launch** を選択するか、ジョブテンプレートで Survey を設定して有効にする必要があります。

回答された Survey の質問は **extra_vars** になります。

20.7. ジョブテンプレートの SURVEY

Run または **Check** のジョブタイプでは、**Job Template** の作成画面または編集画面で Survey を設定する方法が提供されます。Survey では、**Prompt for Extra Variables** と同様に、Playbook に追加変数を設定しますが、ユーザーにわかりやすいほうほうで質問と応答を設定します。Survey では、ユーザー入力の検証も可能です。**Survey** タブを選択して、Survey を作成します。

例

Survey はさまざまな状況で使用できます。たとえば、オペレーションでは、開発者が事前に Ansible の知識がなくても実行可能な「Push to Stage」ボタンを導入したいと考えています。このタスクを起動すると、どのタグを解放する必要がありますか? などの質問に対する回答を求めることができます。

多項選択式の質問など、多種の質問を尋ねることができます。

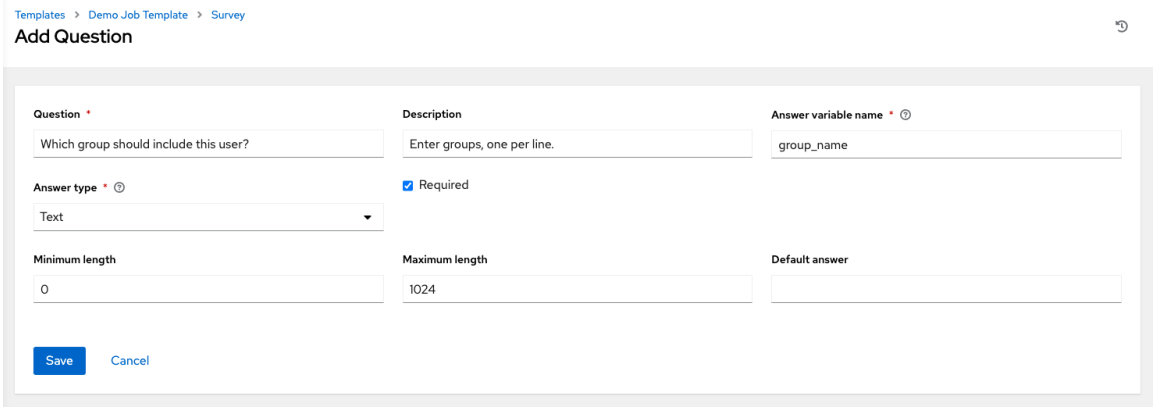
20.7.1. Survey の作成

手順

1. **Survey** タブで **Add** をクリックします。
2. Survey は、任意の数の質問を含めて構成できます。質問ごとに、次の情報を入力します。
   - **質問**: ユーザーに尋ねる質問。
   - オプション: **説明**: ユーザーに尋ねる内容の説明。
   - **回答変数名**: ユーザーの回答を保存する Ansible 変数名。これは、Playbook で使用される変数です。変数名にはスペースを含めることはできません。
   - **回答タイプ**: 以下の質問のタイプから選択します。
     - **テキスト**: 単一行のテキスト。この回答の最小長と最大長 (文字数) を設定できます。
     - **Textarea**: 複数行のテキストフィールド。この回答の最小長と最大長 (文字数) を設定できます。

- **パスワード**: 実際のパスワードが扱われるのと同様に、応答は機密情報として扱われます。この回答の最小長と最大長 (文字数) を設定できます。
- **複数の選択肢 (単一選択)**: 一度に 1 つだけ選択できるオプションのリスト。**Multiple Choice Options** フィールドにオプションを 1 行に 1 つずつ入力します。
- **多項選択法 (複数選択)**: 一度に任意の数を選択できるオプションのリスト。**Multiple Choice Options** フィールドにオプションを 1 行に 1 つずつ入力します。
- **整数**: 整数値。この回答の最小長と最大長 (文字数) を設定できます。
- **浮動**: 10 進数。この回答の最小長と最大長 (文字数) を設定できます。

- **必須**: この質問に対する回答がユーザーから求められているかどうかを示します。
- **最小長と最大長**: 回答が特定の長さである必要があるかどうかを指定します。
- **デフォルトの回答**: 質問に対するデフォルトの回答。この値は、インターフェイスに事前に入力されており、ユーザーが回答を指定供しない場合に使用されます。

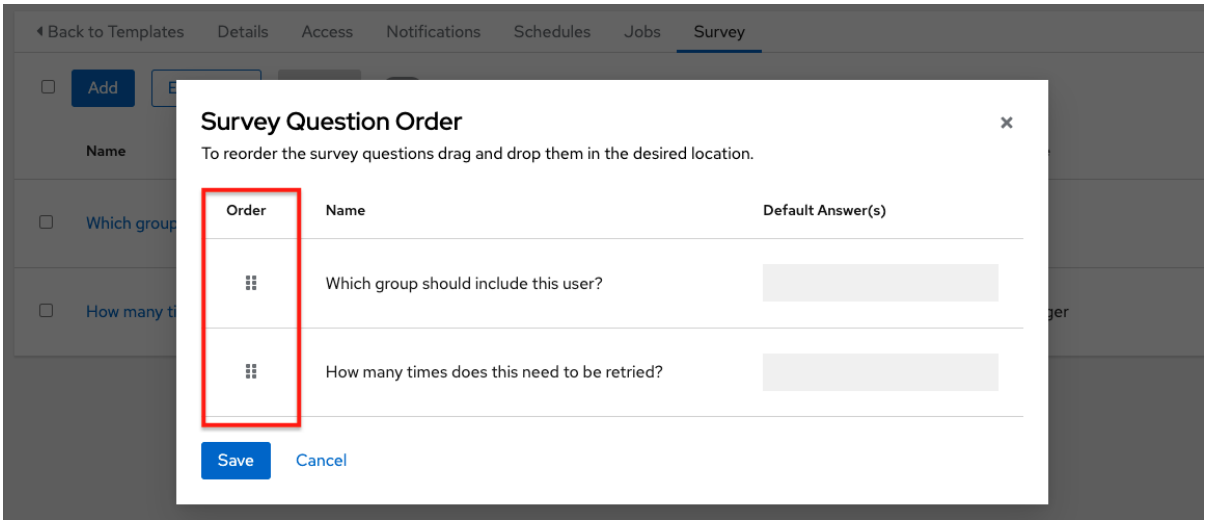


3. 質問情報を入力したら、**Save** をクリックして質問を追加します。

   Survey の質問が **Survey** リストに表示されます。質問がある場合は、 をクリックして編集してください。

   各質問の横にあるボックスをオンにし、**Delete** をクリックして質問を削除するか、メニューバーの切り替えオプションを使用して Survey プロンプトを有効または無効にします。

   Survey の質問が複数ある場合は、グリッドアイコンをクリックしてドラッグし、**Edit Order** をクリックして質問の順序を並べ替えます。



4. さらに質問を追加するには、**Add** をクリックします。

20.7.2. オプションの Survey の質問

Survey の質問に対する **必須** の設定は、対話するユーザーにとって回答がオプションかどうかを決定します。

オプションの Survey 変数を **extra_vars** で Playbook に渡すこともできます。

- テキスト以外の変数 (入力タイプ) がオプションとマークされ、入力されていない場合、Survey の **extra_var** は Playbook に渡されません。
- テキスト入力またはテキスト領域の入力がオプションとしてマークされ、入力されていない場合で、最小の **length > 0** が設定されている場合、Survey の **extra_var** は Playbook に渡されません。
- テキスト入力またはテキスト領域の入力がオプションとしてマークされ、入力されていない場合で、最小の **length === 0** が設定されている場合、Survey の **extra_var** は、値が空のストリング ("") に設定された状態で Playbook に渡されます。

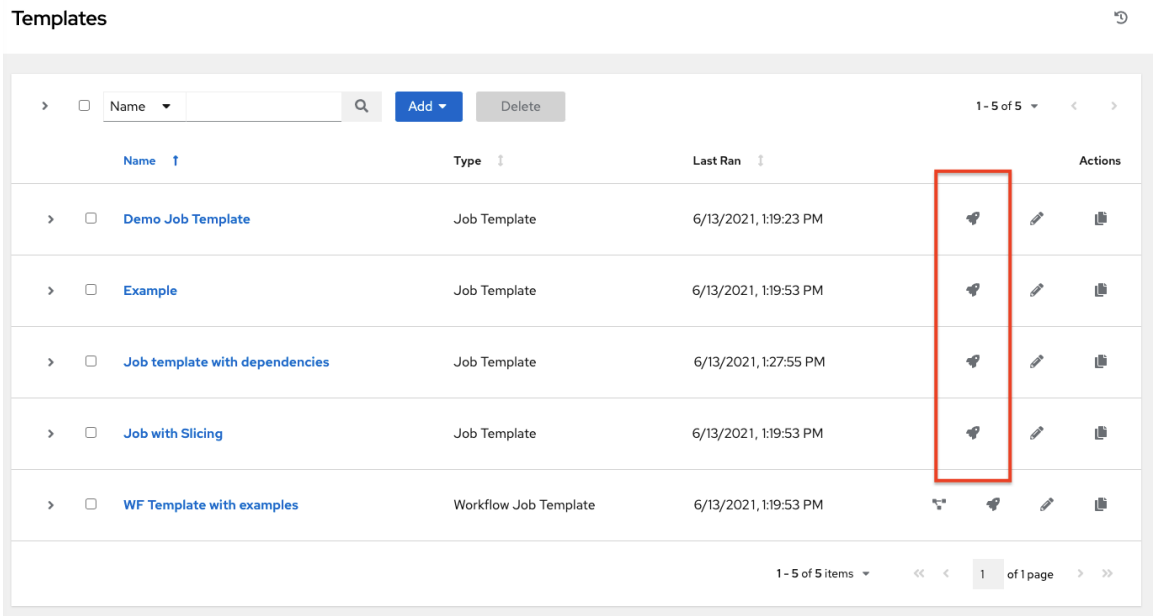
20.8. ジョブテンプレートの起動

Automation Controller の利点は、ボタンを押すだけで Ansible Playbook をデプロイできることです。コマンドラインで常に Ansible Playbook に渡すすべてのパラメーターを保存するようにテンプレートを設定できます。テンプレートは、Playbook に加えて、インベントリー、認証情報、追加の変数、およびコマンドラインで指定できるすべてのオプションと設定を渡します。

デプロイメントが簡単になると、毎回同じ方法で Playbook を実行し、責任を委任できるため、一貫性が高まります。

手順

- 次のいずれかの方法を使用して、ジョブテンプレートを起動します。
  - ナビゲーションパネルで、ジョブテンプレートの横にある **Templates → Launch** を選択します。



- 起動するジョブテンプレートのジョブテンプレートの **Details** ビューで、**Launch** をクリックします。

ジョブを実行するには追加情報が必要になる場合があります。起動時に次のデータをリクエストできます。

- 設定された認証情報
- すべてのパラメーターに対して **Prompt on Launch** オプションが選択されています
- **Ask** に設定されたパスワードまたはパスフレーズ
- Survey (ジョブテンプレート用に設定されている場合)
- 追加変数 (ジョブテンプレートで必要な場合)

注記

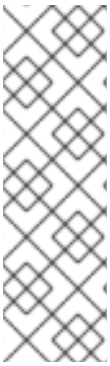
ジョブにユーザー指定の値がある場合、再起動時にそれらの値が考慮されます。ユーザーが値を指定しなかった場合、ジョブはジョブテンプレートのデフォルト値を使用します。ジョブはそのままでは再起動されません。ユーザープロンプトがジョブテンプレートに再適用されて再起動されます。

1 つのタブで値を指定した場合は、前のタブに戻り、次のタブに進むと、残りのタブで値を再指定する必要があります。プロンプトが表示される順序でタブに入力してください。

Automation controller を起動すると、Web ブラウザーが **Jobs** タブの下にあるこのジョブの **Job Status** ページに自動的にリダイレクトされます。

リストビューから最新のジョブを再起動して、指定したインベントリー内のすべてのホストまたは失敗したホストのみで再実行できます。詳細は、ジョブ セクションを参照してください。

スライスジョブが実行中の場合は、ジョブリストはワークフローとジョブスライス以外に、個別の詳細が確認できるリンクを表示します。



注記

API に新しく追加されたエンドポイント **/api/v2/bulk/job_launch** を使用して、ジョブを一括起動できます。このエンドポイントは JSON を受け入れ、起動する統合ジョブテンプレート (ジョブテンプレートやプロジェクト更新など) のリストを指定できます。ユーザーには、すべてのジョブを起動するための適切なパーミッションが割り当てられている必要があります。すべてのジョブが起動されなかった場合は、操作が完了できなかった理由を示すエラーが返されます。**OPTIONS** リクエストを使用して、関連するスキーマを返します。詳細は、Automation controller API ガイドのリファレンスセクションの バルクエンドポイント を参照してください。

20.9. ジョブテンプレートのコピー

ジョブテンプレートをコピーしても、関連するスケジュール、通知、またはパーミッションはコピーされません。スケジュールと通知は、ジョブテンプレートのコピーを作成するユーザーまたは管理者によって再作成される必要があります。ジョブテンプレートをコピーするユーザーには管理者パーミッションが付与されますが、ジョブテンプレートにはパーミッションが割り当てられません (コピーされません)。

手順

1. ナビゲーションパネルから、**Resources → Templates** を選択します。
2. コピーするテンプレートに関連付けられている アイコンをクリックします。
   - コピー元のテンプレート名とタイムスタンプが付いた新しいテンプレートがテンプレートのリストに表示されます。
3. クリックして新しいテンプレートを開き、**Edit*** をクリックします。
4. **Name** フィールドの内容を新規の名前に置き換え、他のフィールドのエントリーを指定または変更してこのページを完了します。
5. **Save** をクリックします。

20.10. スキャンジョブテンプレート

Automation controller3.2 以降、スキャンジョブはサポートされなくなりました。このシステム追跡機能は、ファクトを履歴データとして取得して保存する方法として使用されました。ファクトは、ファクトキャッシュを通じてコントローラーに保存されるようになりました。詳細は、ファクトキャッシュ を参照してください。

Automation controller 3.2 より前のシステムで使用されるジョブテンプレートスキャンジョブは、通常のジョブテンプレートと同様に、run を入力するように変換されます。インベントリーや認証情報などの関連リソースを保持します。デフォルトでは、関連プロジェクトを持たないジョブテンプレートスキャンジョブには特別な Playbook が割り当てられます。独自のスキャン Playbook を使用してプロジェクトを指定することもできます。awx-facts-playbooks を指すプロジェクトが組織ごとに作成され、ジョブテンプレートが Playbook: https://github.com/ansible/tower-fact-modules/blob/master/scan_facts.yml に設定されます。

20.10.1. ファクトスキャン Playbook

スキャンジョブ Playbook **scan_facts.yml** には、パッケージ、サービス、ファイルという 3 つの **ファクトスキャンモジュール** の呼び出しと、Ansible の標準ファクト収集機能が含まれています。**scan_facts.yml** Playbook ファイルは次のようになります。

```
- hosts: all
  vars:
    scan_use_checksum: false
    scan_use_recursive: false
  tasks:
    - scan_packages:
    - scan_services:
    - scan_files:
        paths: '{{ scan_file_paths }}'
        get_checksum: '{{ scan_use_checksum }}'
        recursive: '{{ scan_use_recursive }}'
      when: scan_file_paths is defined
```

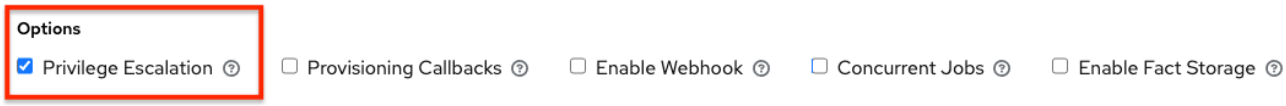
scan_files ファクトモジュールは、スキャンジョブテンプレートの **extra_vars** で渡されるパラメーターを受け入れる唯一のモジュールです。

scan_file_paths: /**tmp**/**scan_use_checksum**: true **scan_use_recursive**: true

- **scan_file_paths** パラメーターには複数の設定がある場合があります (/**tmp**/ または /**var**/**log** など)。

- **scan_use_checksum** および **scan_use_recursive** パラメーターを false に設定するか、省略することもできます。省略も、設定の誤りと同等です。

スキャンジョブテンプレートでは、**become** を有効にし、**become** が可能である **認証情報** を使用する必要があります。オプションメニューから **Privilege Escalation** をオンにすることで、**become** を有効にできます。



20.10.2. scan_facts.yml でサポートされるオペレーションシステム

ファクトキャッシュを使用して **scan_facts.yml** Playbook を使用する場合は、次のサポート対象のオペレーティングシステムのいずれかを使用していることを確認してください。

- Red Hat Enterprise Linux 5、6、7、8、9
- Ubunto 23.04 (Ubunto は非推奨となり、今後のリリースで削除されます)
- OEL 6 および 7
- SLES 11 および 12
- Debian 6、7、8、9、10、11、12
- Fedora 22、23、24
- Amazon Linux 2023.1.20230912
- Windows Server 2008 以降

これらのオペレーティングシステムの一部では、Python を実行するために、またはスキャンモジュールが依存する **python-apt** などの Python パッケージにアクセスするために初期設定が必要です。

20.10.3. スキャン前の設定

以下は、スキャンジョブを実行できるように特定のディストリビューションを設定する Playbook の例です。

```
Bootstrap Ubuntu (16.04)
---
- name: Get Ubuntu 16, and on ready
 hosts: all
 sudo: yes
 gather_facts: no
 tasks:
 - name: install python-simplejson
   raw: sudo apt-get -y update
   raw: sudo apt-get -y install python-simplejson
   raw: sudo apt-get install python-apt

Bootstrap Fedora (23, 24)
---
- name: Get Fedora ready
 hosts: all
```

```
sudo: yes
gather_facts: no
tasks:
- name: install python-simplejson
  raw: sudo dnf -y update
  raw: sudo dnf -y install python-simplejson
  raw: sudo dnf -y install rpm-python
```

20.10.4. カスタムファクトスキャン

カスタムファクトスキャンの Playbook は ファクトスキャン Playbook セクションの例に似ています。たとえば、カスタム **scan_foo** Ansible fact モジュールのみを使用する Playbook は次のようになります。

```
scan_foo.py:
def main():
    module = AnsibleModule(
        argument_spec = dict())


    foo = [
      {
        "hello": "world"
      },
      {
        "foo": "bar"
      }
    ]
    results = dict(ansible_facts=dict(foo=foo))
    module.exit_json(**results)


main()
```

カスタムファクトモジュールを使用するには、それがスキャンジョブテンプレートで使用される Ansible プロジェクトの **/library/** サブディレクトリーに存在することを確認してください。このファクトスキャンモジュールは、ハードコードされたファクトのセットを返します。

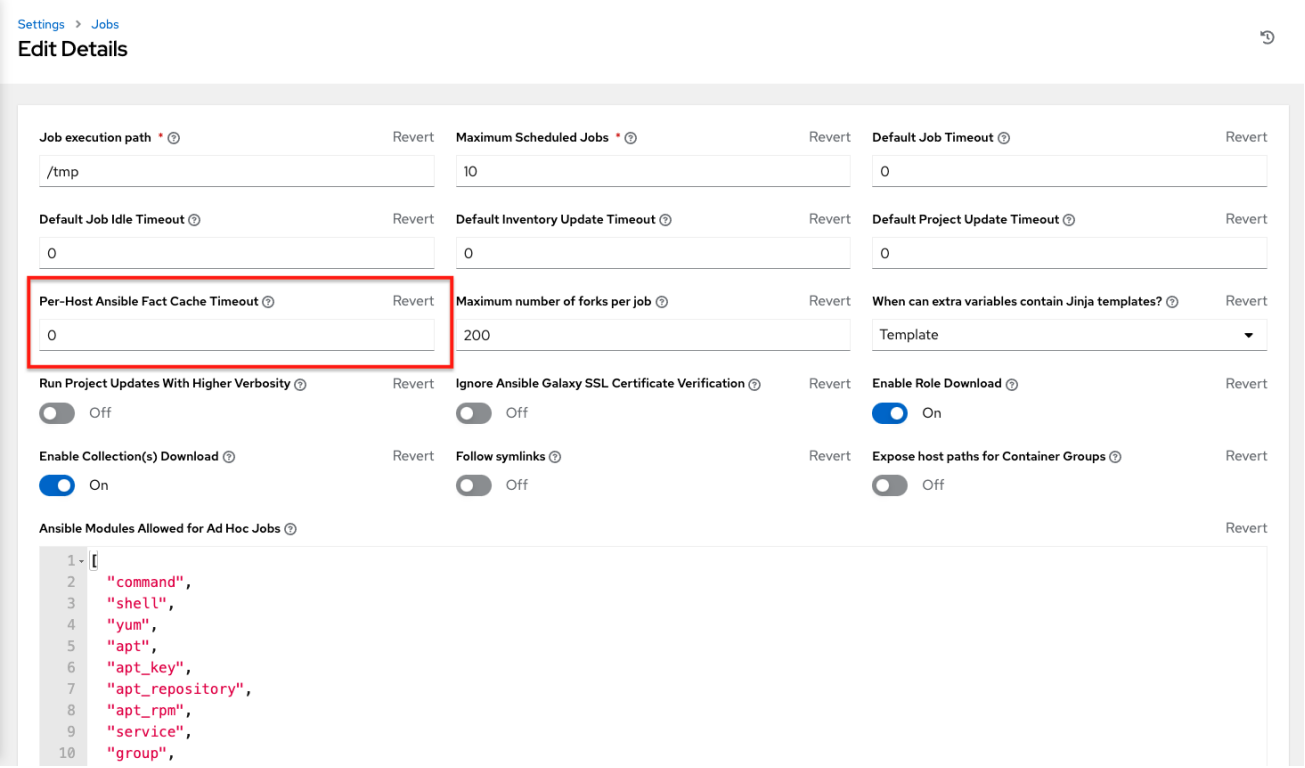
```
[
  {
    "hello": "world"
  },
  {
    "foo": "bar"
  }
]
```

詳細は、Ansible ドキュメントの モジュールの開発 セクションを参照してください。

20.10.5. ファクトキャッシング

Automation controller は、Ansible Fact Cache プラグインを通じてホストごとにファクトを保存および取得できます。この動作はジョブテンプレートごとに設定できます。ファクトキャッシングはデフォルトでオフになっていますが、こちらを有効にして、実行中のジョブに関連するインベントリーの全ホス

トに対するファクトリクエストに対応できます。これにより、ホストファクトのインベントリー全体にアクセスしながら、**--limit** を指定してジョブテンプレートを使用できるようになります。プラグインがホストごとに強制するグローバルタイムアウト設定は、**Jobs** 設定メニューから指定できます (秒単位)。



ファクトキャッシュ (**use_fact_cache=True**) を使用するジョブを起動すると、各ホストの **ansible_facts** はすべてコントローラーによってジョブのインベントリーに保存されます。

Automation controller に同梱されている Ansible Fact Cache プラグインは、ファクトキャッシュが有効になっている (**use_fact_cache=True**) ジョブで有効になります。

ファクトキャッシュが有効になっているジョブ (**use_fact_cache=True**) が実行されると、Automation controller はインベントリー内のホストの全レコードを復元します。ホストごとに現在保存されているファクトよりも更新時刻が新しいレコードはデータベース内で更新されます。

新しいファクトと変更されたファクトは、Automation controller のログ機能を通じて記録されます。具体的には、**system_tracking namespace** またはロガーに対してです。ロギングペイロードには次のフィールドが含まれます。

- **host_name**
- **inventory_id**
- **ansible_facts**

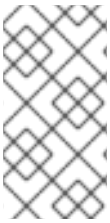
ansible facts は、Automation Controller インベントリー (**inventory_id**) に含まれる **host_name** の全 Ansible ファクトのディクショナリーです。

注記

ホスト名にスラッシュ (/) が含まれている場合、ファクトキャッシュはそのホストでは機能しません。100 台のホストを含むインベントリーがあり、1 台のホストの名前に / が含まれている場合、残りの 99 台のホストは引き続きファクトを収集します。

20.10.6. ファクトキャッシングの利点

ファクトキャッシュにより、ファクト収集の実行にかかる時間を節約できます。1000 個のホストとフォークに対して実行するジョブに Playbook がある場合、これらの全ホストからファクトを収集するのに 10 分費やすことができます。ただし、ジョブを定期的に実行する場合、最初の実行でこれらのファクトがキャッシュされ、次の実行でデータベースから取得されます。こうすることで、スマートインベントリーを含む大規模なインベントリーに対するジョブの実行時間が短縮されます。



注記

ファクトキャッシュの適用に ansible.cfg ファイルを変更しないでください。カスタムファクトキャッシュは、コントローラーのファクトキャッシュ機能と競合する可能性があります。Automation controller に付属のファクトキャッシュモジュールを使用する必要があります。

キャッシュされたファクトのジョブでの使用は、ジョブテンプレートウィンドウの **Options** フィールドでこれを有効にして選択できます。

ファクトを消去するには、Ansible **clear_facts** メタタスク を実行します。以下は、Ansible **clear_facts** メタタスクを使用する Playbook の例です。

```
- hosts: all
  gather_facts: false
  tasks:
    - name: Clear gathered facts from all currently targeted hosts
      meta: clear_facts
```

ファクトキャッシュの API エンドポイントは次の場所にあります。

http://<controller server name>/api/v2/hosts/x/ansible_facts

20.11. クラウドインベントリーでのクラウド認証情報の使用

クラウド認証情報は、クラウドインベントリーを同期するときに使用できます。また、ジョブテンプレートに関連付けて、Playbook で使用するためにランタイム環境に含めることもできます。次のクラウド認証情報がサポートされています。

- Openstack
- Amazon Web Services
- Google
- Azure
- VMware

20.11.1. OpenStack

次のサンプル Playbook は、**nova_compute** Ansible OpenStack クラウドモジュールを呼び出し、認証情報を必要とします。

- **auth_url**
- **username**
- **password**
- **project name**

これらのフィールドは、環境変数 **OS_CLIENT_CONFIG_FILE** を通じて Playbook で利用できるようになります。これは、クラウド認証情報の内容に基づいてコントローラーによって書き込まれた YAML ファイルを指します。次のサンプル Playbook は、YAML ファイルを Ansible 変数空間にロードします。

- OS_CLIENT_CONFIG_FILE の例:

```
clouds:
  devstack:
    auth:
      auth_url: http://devstack.yoursite.com:5000/v2.0/
      username: admin
      password: your_password_here
      project_name: demo
```

- Playbook の例:

```
- hosts: all
  gather_facts: false
  vars:
    config_file: "{{ lookup('env', 'OS_CLIENT_CONFIG_FILE') }}"
    nova_tenant_name: demo
    nova_image_name: "cirros-0.3.2-x86_64-uec"
    nova_instance_name: autobot
    nova_instance_state: 'present'
    nova_flavor_name: m1.nano


    nova_group:
      group_name: antarctica
      instance_name: deceptacon
      instance_count: 3
  tasks:
    - debug: msg="{{ config_file }}"
    - stat: path="{{ config_file }}"
      register: st
    - include_vars: "{{ config_file }}"
      when: st.stat.exists and st.stat.isreg


    - name: "Print out clouds variable"
      debug: msg="{{ clouds|default('No clouds found') }}"


    - name: "Setting nova instance state to: {{ nova_instance_state }}"
```

```
local_action:
  module: nova_compute
  login_username: "{{ clouds.devstack.auth.username }}"
  login_password: "{{ clouds.devstack.auth.password }}"
```

20.11.2. Amazon Web Services

Amazon Web Services (AWS) クラウド認証情報は Playbook の実行時に以下の環境変数として公開されます (ジョブテンプレートで、設定に必要なクラウド認証情報を選択します):

- **AWS_ACCESS_KEY_ID**
- **AWS-SECRET_ACCESS_KEY**

各 AWS モジュールは、**aws_access_key_id** または **aws_secret_access_key** モジュールオプションを設定することなく、コントローラーでの実行時にこれらの認証情報を暗黙的に使用します。

20.11.3. Google

Google クラウド認証情報は、Playbook の実行時に以下の環境変数として公開されます (ジョブテンプレートで、設定に必要なクラウド認証情報を選択します)。

- **GCE_EMAIL**
- **GCE_PROJECT**
- **GCE_CREDENTIALS_FILE_PATH**

各 Google モジュールは、**service_account_email**、**project_id**、または **pem_file** モジュールオプションを設定することなく、コントローラーでの実行時にこれらの認証情報を暗黙的に使用します。

20.11.4. Azure

Azure クラウド認証情報は、Playbook の実行時に以下の環境変数として公開されます (ジョブテンプレートで、設定に必要なクラウド認証情報を選択します)。

- **AZURE_SUBSCRIPTION_ID**
- **AZURE_CERT_PATH**

各 Azure モジュールは、**subscription_id** または **Management_cert_path** モジュールオプションを設定することなく、コントローラー経での実行時にこれらの認証情報を暗黙的に使用します。

20.11.5. VMware

VMware クラウド認証情報は、Playbook の実行時に以下の環境変数として公開されます (ジョブテンプレートで、設定に必要なクラウド認証情報を選択します)。

- **VMWARE_USER**
- **VMWARE_PASSWORD**
- **VMWARE_HOST**

次のサンプル Playbook は、これらの認証情報の使用法を示しています。

```
- vsphere_guest:
    vcenter_hostname: "{{ lookup('env', 'VMWARE_HOST') }}"
    username: "{{ lookup('env', 'VMWARE_USER') }}"
    password: "{{ lookup('env', 'VMWARE_PASSWORD') }}"
    guest: newvm001
    from_template: yes
    template_src: linuxTemplate
    cluster: MainCluster
    resource_pool: "/Resources"
    vm_extra_config:
      folder: MyFolder
```

20.12. プロビジョニングコールバック

プロビジョニングコールバックは、automation controller の機能で、ユーザーが automation controller コンソールからホストを管理するジョブを起動するのを待たずに、ホストが自身に対して実行する Playbook を開始できるようにします。

プロビジョニングコールバックは、呼び出し側ホストで Playbook を実行するためにのみ使用され、クラウドバースティングを目的としています。クラウドバースティングは、コンピューティング需要の急増時にパブリッククラウドにバーストすることで、プライベートクラウドがパブリッククラウドリソースにアクセスできるようにするクラウドコンピューティング設定です。

例

別のホストに対してジョブを実行しないように、認証キーの送信などの設定用にクライアントからサーバーへの通信が必要な新しいインスタンス。これにより、次のものが自動的に設定されます。

- 別のシステム (AWS 自動スケーリング、キックスタートやプレシードなどの OS プロビジョニングシステムなど) によってプロビジョニングされた後のシステム。
- Automation controller API を直接呼び出さずに、プログラムでジョブを起動します。

起動したジョブテンプレートは、プロビジョニングを要求しているホストに対してのみ実行されます。

通常、firstboot タイプのスクリプトまたは **cron** で、アクセスします。

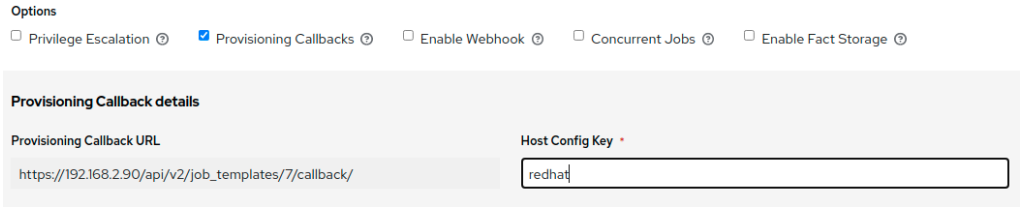
20.12.1. プロビジョニングコールバックの有効化

手順

- コールバックを有効にするには、ジョブテンプレートの **Provisioning Callbacks** チェックボックスをオンにします。これにより、ジョブテンプレートの **プロビジョニングコールバック URL** が表示されます。

注記

automation controller のプロビジョニングコールバック機能を動的インベントリーと共に使用しようとする場合、**Update on Launch** がジョブテンプレートでインベントリーグループに対して設定されている必要があります。



URL を持つ外部ホストが設定を要求できないように、コールバックにはホスト設定キーも必要です。ホスト設定キーのカスタム値を指定します。ホストキーを複数のホスト間で再利用して、このジョブテンプレートを複数のホストに適用できます。設定要求が可能なホストを制御する場合は、このキーをいつでも変更できます。

REST を使用して手動でコールバックするには以下を実行します。

手順

1. UI で、 https://<CONTROLLER_SERVER_NAME>/api/v2/job_templates/7/callback/ の形式のコールバック URL を確認します。
   - このサンプル URL の '7' は、automation controller のジョブテンプレート ID です。
2. ホストからのリクエストが POST であることを確認してください。以下は、**curl** を使用した例です (すべて 1 行にあります)。

```
curl -k -f -i -H 'Content-Type:application/json' -XPOST -d '{"host_config_key": "redhat"}' \
        https://<CONTROLLER_SERVER_NAME>/api/v2/job_templates/7/callback/
```

3. コールバックを成功させるには、要求元のホストがインベントリーに定義されていることを確認してください。

トラブルシューティング

Automation controller が、定義されたインベントリーのいずれかで名前または IP アドレスでホストを見つけることができない場合、リクエストは拒否されます。この方法でジョブテンプレートを実行する場合は、それ自体に対して Playbook の実行を開始するホストがインベントリーに存在することを確認してください。ホストがインベントリーにない場合、ジョブテンプレートは失敗し、**No Hosts Matched** のタイプエラーメッセージが表示されます。

ホストがインベントリーに存在せず、インベントリーグループに対して **Update on Launch** が設定されている場合、Automation controller はコールバックを実行する前にクラウドベースのインベントリーソースを更新しようとします。

検証

リクエストが成功すると、**Jobs** タブにエントリーが表示され、そこで結果と履歴を確認できます。
REST を使用してコールバックにアクセスできますが、コールバックを使用する推奨方法は、Automation controller に同梱されているサンプルスクリプトの 1 つを使用することです。

- **/usr/share/awx/request_tower_configuration.sh** (Linux/UNIX)

- **/usr/share/awx/request_tower_configuration.ps1** (Windows)

これらの使用法は、次に示すように **-h** フラグを渡すことによってファイルのソースコードに記述されます。

```
./request_tower_configuration.sh -h
Usage: ./request_tower_configuration.sh <options>


Request server configuration from Ansible Tower.


OPTIONS:
 -h      Show this message
 -s      Controller server (e.g. https://ac.example.com) (required)
 -k      Allow insecure SSL connections and transfers
 -c      Host config key (required)
 -t      Job template ID (required)
 -e      Extra variables
```

このスクリプトはコマンドを再試行できるため、単純な **CURL** リクエストよりも強力なコールバックの使用方法です。スクリプトは 1 分に 1 回、最大 10 分間再試行します。

注記

これはスクリプトの例です。200 以外のエラーコードは再試行が必要な一時的なエラーではない可能性があるため、障害シナリオの検出時に、より動的な動作が必要な場合は、このスクリプトを編集します。

Automation controller の動的インベントリーでコールバックを使用できます。たとえば、サポートされているクラウドプロバイダーの 1 つからクラウドインベントリーを取得する場合などです。このような場合は、**Update On Launch** の設定とともに、クラウドの API エンドポイントのハンマリングを回避するために、インベントリーソースのインベントリーキャッシュタイムアウトを必ず設定してください。**request_tower_configuration.sh** スクリプトは 1 分に 1 回、最大 10 分間ポーリングするため、インベントリー (インベントリーソース自体で設定) のキャッシュ無効化にかける推奨の時間値は 1 ~ 2 分になります。

cron ジョブから **request_tower_configuration.sh** スクリプトを実行することは推奨できませんが、推奨される cron 間隔は 30 分です。多くのユーザーは主に、オンラインになったタイミングで最新の設定にブートストラップされたベースイメージを有効にするという使用をしているので、繰り返しの設定は、automation controller をスケジューリングすることで処理できます。最初の起動時に実行するのがベストプラクティスです。最初の起動スクリプトは、init スクリプトで通常、自身でそのまま削除するため、**request_tower_configuration.sh** スクリプトのコピーを呼び出す init スクリプトを設定して、自動スケーリングイメージに変換します。

20.12.2. 追加変数をプロビジョニングコールバックに渡す方法

通常のジョブテンプレートと同じ方法で、プロビジョニングコールバックで **extra_vars** を渡すことができます。**extra_vars** を渡すには、送信されるデータは、アプリケーションまたは JSON のコンテンツタイプとして POST のボディーに含める必要があります。

手順

- 次のいずれかの方法を使用して、追加の変数を渡します。

- 独自の **extra_vars** を追加する場合は、例として次の JSON 形式を使用します。

```
'{"extra_vars": {"variable1":"value1","variable2":"value2",...}}'
```

- **curl** を使用して追加の変数をジョブテンプレート呼び出しに渡します。

```
root@localhost:~$ curl -f -H 'Content-Type: application/json' -XPOST \
-d '{"host_config_key": "redhat", "extra_vars": "{\"foo\": \"bar\"}"}' \
https://<CONTROLLER_SERVER_NAME>/api/v2/job_templates/7/callback
```

詳細は、**Automation controller 管理ガイド** の Curl を使用したジョブの起動 を参照してください。

20.13. 追加変数

survey 変数を渡すと、Automation controller 内の追加変数 (**extra_vars**) として渡されます。ただし、(Survey の場合と同様に) 追加の変数をジョブテンプレートに渡すと、インベントリーおよびプロジェクトから渡される他の変数がオーバーライドされる可能性があります。

デフォルトでは、**extra_vars** は **!unsafe** とマークされます。ただし、ジョブテンプレートの Extra Variables でユーザーが指定したものは、そのようにマークされません。これらは、ジョブテンプレートを追加または編集する権限を持つユーザーのみが追加できるため、信頼できるものです。たとえば、ネストされた変数は、プロンプトとして入力されたときに展開されません。Jinja の括弧は文字列として扱われるためです。安全でない変数の詳細は、Unsafe or raw strings を参照してください。

注記

ジョブ起動 API に渡される **extra_vars** は、以下のいずれかが該当する場合のみ有効です。

- それらは有効な survey の変数に対応する。
- **ask_variables_on_launch** が **True** に設定されている。

例

debug = true のインベントリーの変数が定義されています。この変数 **debug = true** は、ジョブテンプレート Survey でオーバーライドされる可能性があります。

渡す変数がオーバーライドされないようにするには、Survey で変数を再定義して変数が含まれていることを確認します。追加の変数は、インベントリー、グループ、およびホストのレベルで定義できます。

ALLOW_JINJA_IN_EXTRA_VARS パラメーターを指定する場合は、**Automation Controller 管理ガイド** の コントローラーのヒントと裏技 セクションを参照して、Controller UI の **Jobs Settings** 画面でパラメーターを設定してください。

ジョブテンプレートの追加変数ディクショナリーは Survey 変数にマージされます。

以下は、YAML および JSON 形式の **extra_vars** の簡略化された例です。

- YAML 形式の設定:

```
launch_to_orbit: true
satellites:
```

```
  - sputnik
  - explorer
  - satcom
```

- JSON 形式の設定:

```
{
  "launch_to_orbit": true,
  "satellites": ["sputnik", "explorer", "satcom"]
}
```

次の表は、automation controller での変数の順序に関する動作 (階層) を、Ansible と比較したものです。

表20.1 Automation Controller 変数の優先順位の階層 (最後にリストされたものが優先されます)

Ansible	Automation Controller
ロールのデフォルト	ロールのデフォルト
動的インベントリー変数	動的インベントリー変数
インベントリー変数	Automation Controller インベントリー変数
インベントリーの **group_vars**	Automation Controller グループ変数
インベントリーの **host_vars**	Automation Controller ホスト変数
Playbook の **group_vars**	Playbook の **group_vars**
Playbook の **host_vars**	Playbook の **host_vars**
ホストのファクト	ホストのファクト
登録された変数	登録された変数
ファクトの設定	ファクトの設定
プレイ変数	プレイ変数
プレイの **vars_prompt**	(サポート対象外)
プレイの **vars_files**	プレイの **vars_files**
ロールと include 変数	ロールと include 変数
ブロック変数	ブロック変数
タスク変数	タスク変数

Ansible	Automation Controller
追加変数	ジョブテンプレートの追加変数
	ジョブテンプレートの Survey (デフォルト)
	ジョブ起動の追加変数

20.13.1. ジョブテンプレートの起動

ジョブを手動で再起動する代わりに、**launch_type** を **relaunch** に設定することで再起動を指定します。再起動の動作は、**extra_vars** を継承しないという点で、起動の動作とは異なります。

ジョブの再起動は継承ロジックを使用しません。再起動されるジョブに対して計算されたものと同じ **extra_vars** を使用します。

例

extra_vars を指定せずにジョブテンプレートを起動すると、**j1** というジョブが作成されます。次に、ジョブテンプレートを編集し、**extra_vars** を追加します (**{ "hello": "world" }"** の追加など)。

j1 を再起動すると **j2** が作成されますが、継承するロジックがなく、**j1** には **extra_vars** がないため、**j2** には extra_vars が指定されません。

j1 の作成後に追加した **extra_vars** を使用してジョブテンプレートを起動すると、作成された再起動ジョブ (**j3**) には extra_vars が含まれます。**j3** を再起動すると、**extra_vars** も含まれる **j4** が作成されます。

第21章 ジョブスライス

スライスされたジョブは、分散ジョブの概念を指します。分散されたジョブは、大多数のホストでジョブを実行する場合に使用し、インベントリーのサブセット上で、複数の Ansible Playbook それぞれを実行することで、クラスターで並行してスケジューリングができます。

デフォルトでは、Ansible は単一のコントロールインスタンスからジョブを実行します。クロスホストオーケストレーションを必要としないジョブの場合、ジョブスライスは、クラスター内の複数のノードに作業を分散する Automation Controller の機能を利用します。

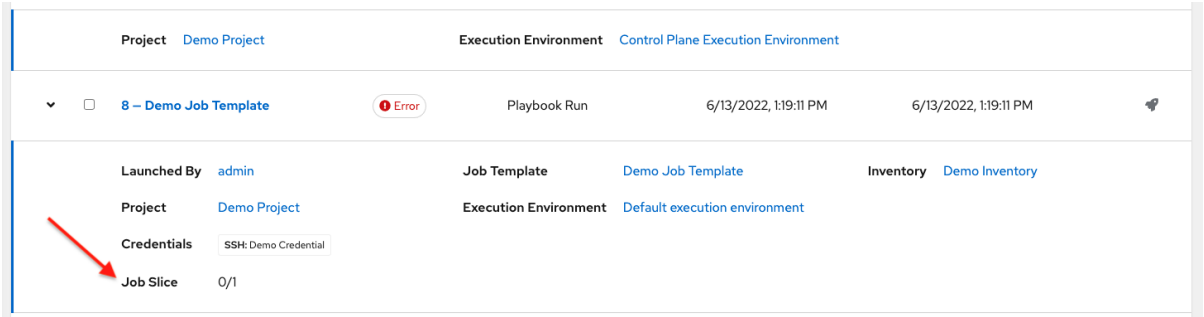
ジョブスライスは、ジョブテンプレートフィールド **job_slice_count** を追加することで機能します。これは、Ansible 実行をスライスするジョブの数を指定します。この数値が **1** より大きい場合、Automation controller はジョブではなくジョブテンプレートからワークフローを生成します。インベントリーはスライスジョブ間で均等に分散されます。その後、ワークフロージョブが開始され、通常のワークフローであるかのように処理が進みます。

ジョブを起動すると、API はジョブリソース (job_slice_count = 1 の場合) またはワークフロージョブリソースのいずれかを返します。対応するユーザーインターフェイス (UI) は適切な画面にリダイレクトされ、実行のステータスが表示されます。

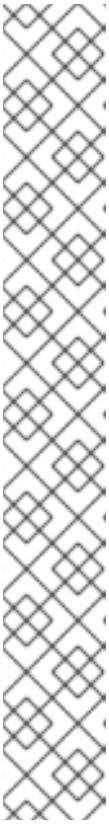
21.1. ジョブスライスの留意事項

ジョブスライスの設定時には以下を考慮してください。

- スライスされたジョブはワークフロージョブを作成し、さらにワークフロージョブがジョブを作成します。
- ジョブスライスはジョブテンプレート、インベントリー、スライス数で設定されること。
- スライスされたジョブが実行されると、各インベントリーがいくつかのスライスサイズのチャンクに分割されます。次に、適切なインベントリーの各チャンクで ansible-playbook 実行のジョブをキューに入れます。ansible-playbook に入力されるインベントリーは、対象となる特定のスライスにあるホストのみを含む元のインベントリーの短縮バージョンです。**Jobs** リストに表示される完了したスライスジョブには、実行されたスライスジョブの数とともにラベルが付けられます。



- スライスされたジョブでのスケジュールの動作は、通常通りです (フォーク数、容量に合わせたキューへの追加、インベントリーマッピングをもとにしたインスタンスグループへの割り当て)。



注記

ジョブスライスは、ジョブの実行を水平方向にスケーリングすることを目的としています。ジョブテンプレートでジョブスライスを有効にすると、処理対象のインベントリーが起動時に設定されたスライスの数に分割され、スライスごとにジョブが開始されます。

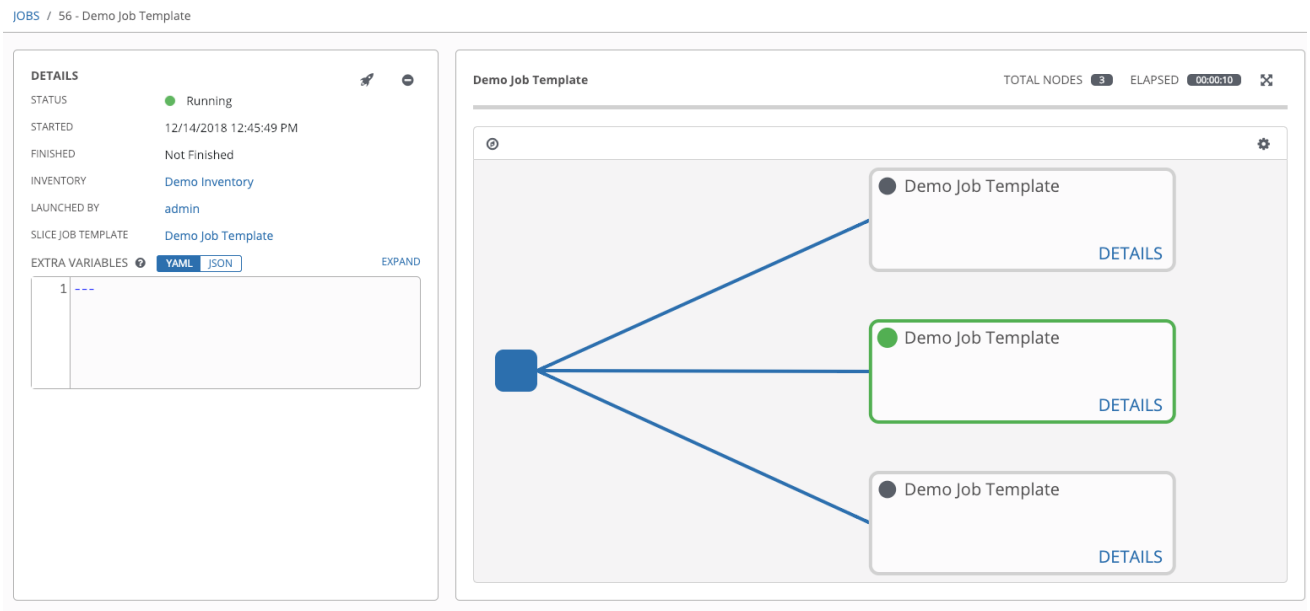
通常、スライスの数は、コントローラーノードの数と同じかそれ以下になります。ジョブスライスの数を極端に多く (例: 数千) 設定できますが、パフォーマンスが低下する可能性があります。これは、ジョブスケジューラーが、スライスされたジョブとなる数千のワークフローノードを同時にスケジュールするように設計されていないためです。

- プロンプトまたは追加の変数を含むスライスジョブテンプレートは、標準ジョブテンプレートと同じように動作し、結果として得られるワークフロージョブ内のスライスジョブのセット全体にすべての変数と制限を適用します。ただし、スライスされたジョブに制限が課された場合には、この制限が原因で、スライスにホストが割り当てられず、これらのスライスが失敗するので、ジョブ全体が失敗してしまいます。
- 分散されたジョブのジョブスライスのステータスは、ワークフロージョブと同じ方法で計算されます。サブジョブでの失敗した内容が未処理の場合は、失敗となります。

- (個別ホストへの変更適用などではなく) ホスト全体でオーケストレーションする予定のジョブは、スライスジョブとして設定しないでください。
- スライスジョブとして設定されたジョブは失敗する可能性があり、automation controller は、スライスジョブとして実行して失敗する Playbook の検出や説明は行いません。

21.2. ジョブスライスの実行動作

ジョブがスライスされると、どのノードでも実行できます。システムの容量が不十分な場合、一部のシステムが異なる時間に実行される可能性があります。スライスジョブの実行中の場合は、ジョブの詳細には、現在実行中のワークフローとジョブスライス、およびそれらの詳細を個別に表示するリンクが表示されます。



デフォルトでは、通常、ジョブテンプレートは同時に実行するように設定されていません (API で **allow_simultaneous** をオンにするか、UI で **Enable Concurrent Jobs** に指定する必要があります)。スライスはこの動作をオーバーライドし、その設定が解除されている場合でも、**allow_simultaneous** を暗黙的に指定します。これを指定する方法と、ジョブテンプレート設定のジョブスライスの数は、ジョブ テンプレートを参照してください。

Job templates のセクションでは、ユーザーインターフェイスの以下の操作を実行する方法が追加で提供されています。

- スライスジョブが複数あるジョブテンプレートでワークフロージョブを起動する
- スライスジョブのテンプレート軌道跡にワークフロー全体または個別ジョブをキャンセルする
- スライスジョブの実行が完了した後にワークフロー全体または個別ジョブを再起動する
- ジョブテンプレートの起動後にワークフローとスライスジョブに関する情報を表示する
- スライスジョブ作成後に特定のスライスジョブを検索する

21.3. ジョブスライスの検索

スライスジョブの検索を簡素化するには、検索機能を使用して検索フィルターを適用します。

- ジョブリストに適用してスライスジョブだけを表示する
- ジョブリストに適用してジョブスライスの親ワークフロージョブのみを表示する
- ジョブテンプレートリストに適用してスライスジョブを生成するジョブテンプレートのみを表示する

手順

- 次のいずれかの方法を使用して、スライスジョブを検索します。
  - ジョブリストのスライスジョブのみを表示するには、他の多くの場合と同様に、タイプ (ここではジョブ) または **unified_jobs** でフィルタリングできます。

    ```
    /api/v2/jobs/?job_slice_count__gt=1
    ```

  - ジョブスライスの親ワークフロージョブのみを表示する方法:

    ```
    /api/v2/workflow_jobs/?job_template__isnull=false
    ```

  - スライスジョブを生成するジョブテンプレートのみを表示する方法:

    ```
    /api/v2/job_templates/?job_slice_count__gt=1
    ```

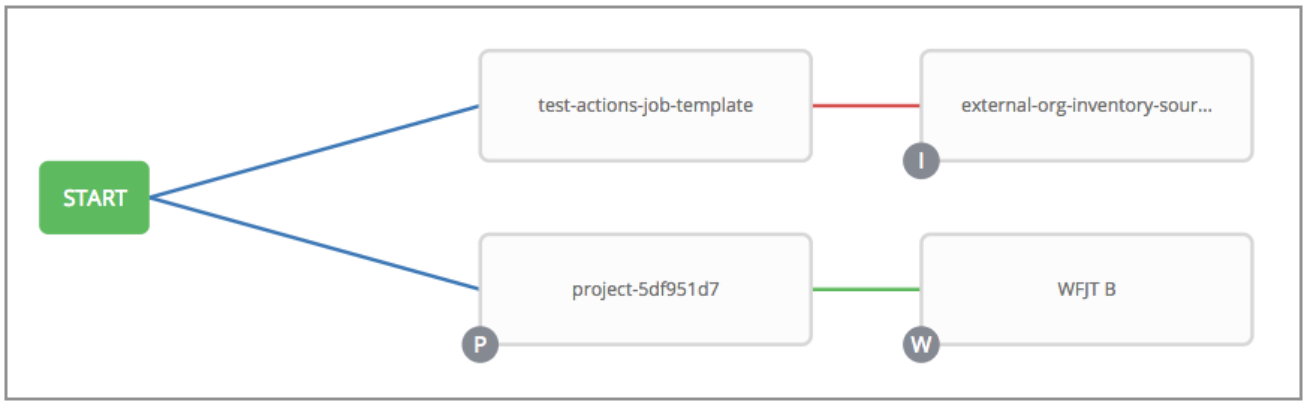
第22章 AUTOMATION CONTROLLER のワークフロー

ワークフローを使用すると、インベントリー、Playbook、またはパーミッションを共有する場合と共有しない場合がある、一連の異なるジョブテンプレート (またはワークフローテンプレート) を設定できます。

ワークフローには、ジョブテンプレートと同様に、**admin** パーミッションと **execute** パーミッションがあります。ワークフローは、1 つの単位としてリリースプロセスに含まれる全ジョブセットをトラッキングするタスクを実行します。

ジョブまたはワークフローテンプレートは、ノードと呼ばれるグラフのような構造を使用して相互にリンクされます。これらのノードは、ジョブ、プロジェクト同期、またはインベントリー同期です。テンプレートは、異なるワークフローの一部にすることも、同じワークフロー内で複数回使用することもできます。ワークフローを起動すると、グラフ構造のコピーがワークフロージョブに保存されます。

次の例は、3 つすべてを含むワークフローと、ワークフロージョブテンプレートを示しています。



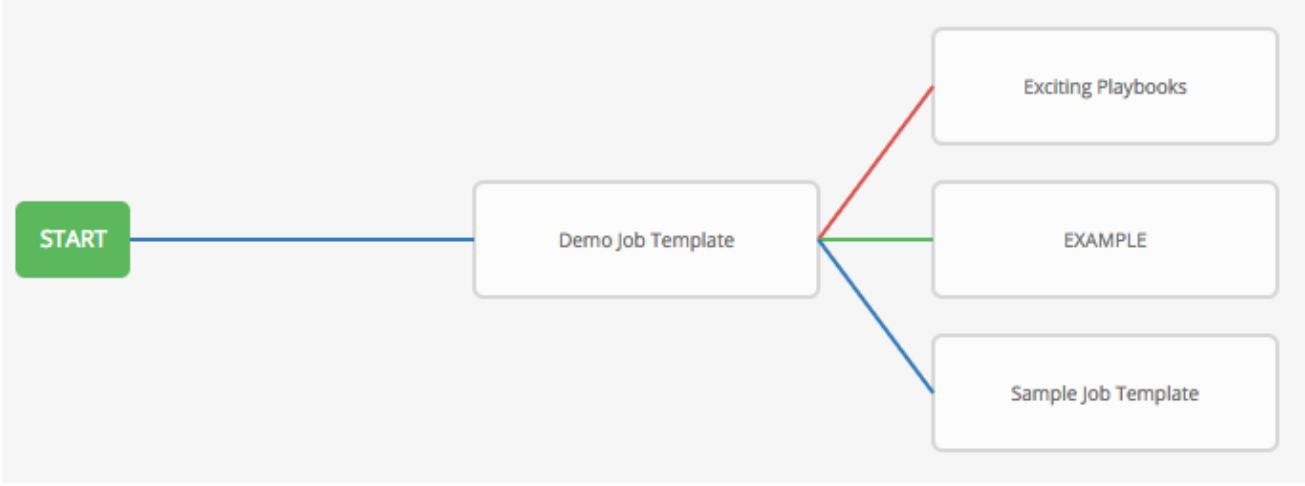
ワークフローが実行されると、ノード内のリンクされたテンプレートからジョブが生成されます。プロンプトフィールド (job_type、job_tags、skip_tags、limit) を持つジョブテンプレートにリンクしているノードには、それらのフィールドを含めることができ、起動時にプロンプトは表示されません。認証情報またはインベントリーの入力を求めるジョブテンプレート (デフォルトなし) は、ワークフローに含めることができません。

22.1. ワークフローのシナリオおよび留意事項

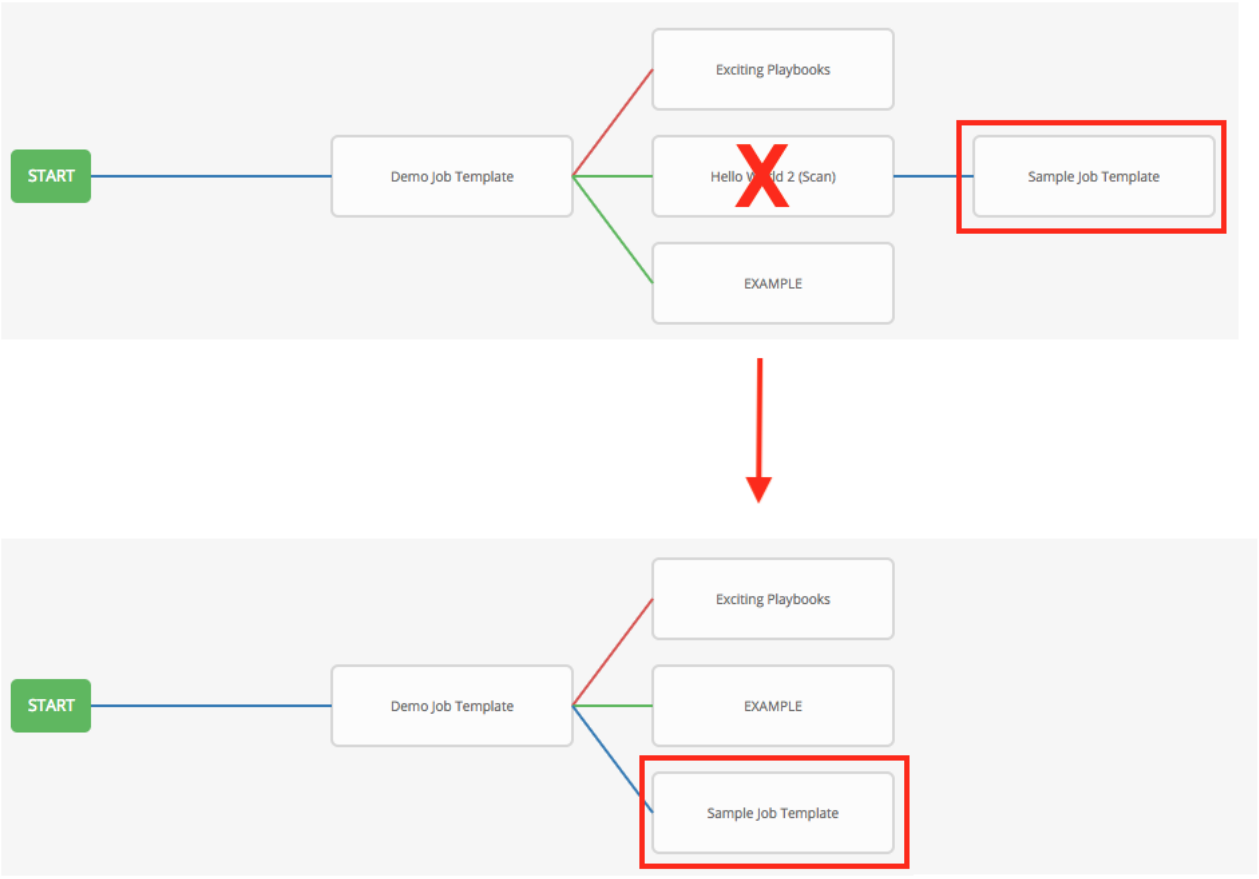
ワークフローを構築するときは、次の点を考慮してください。

- ルートノードは、デフォルトで **ALWAYS** に設定されており、編集できません。

- ノードは複数の親を持つことができ、子は Success、Failure、または Always のいずれかの状態にリンクできます。Always の場合、状態は Success でも Failure でもありません。状態は、ワークフロージョブテンプレートレベルではなく、ノードレベルで適用されます。ワークフロージョブは、キャンセルされるかエラーが発生しない限り、Success としてマークされます。



- 以下の例のように、ワークフロー内のジョブまたはワークフローテンプレートを削除した場合に、以前に削除済みのジョブやワークフローに連携されていたノードが自動的にアップストリームに接続され、エッジタイプが保持されます。

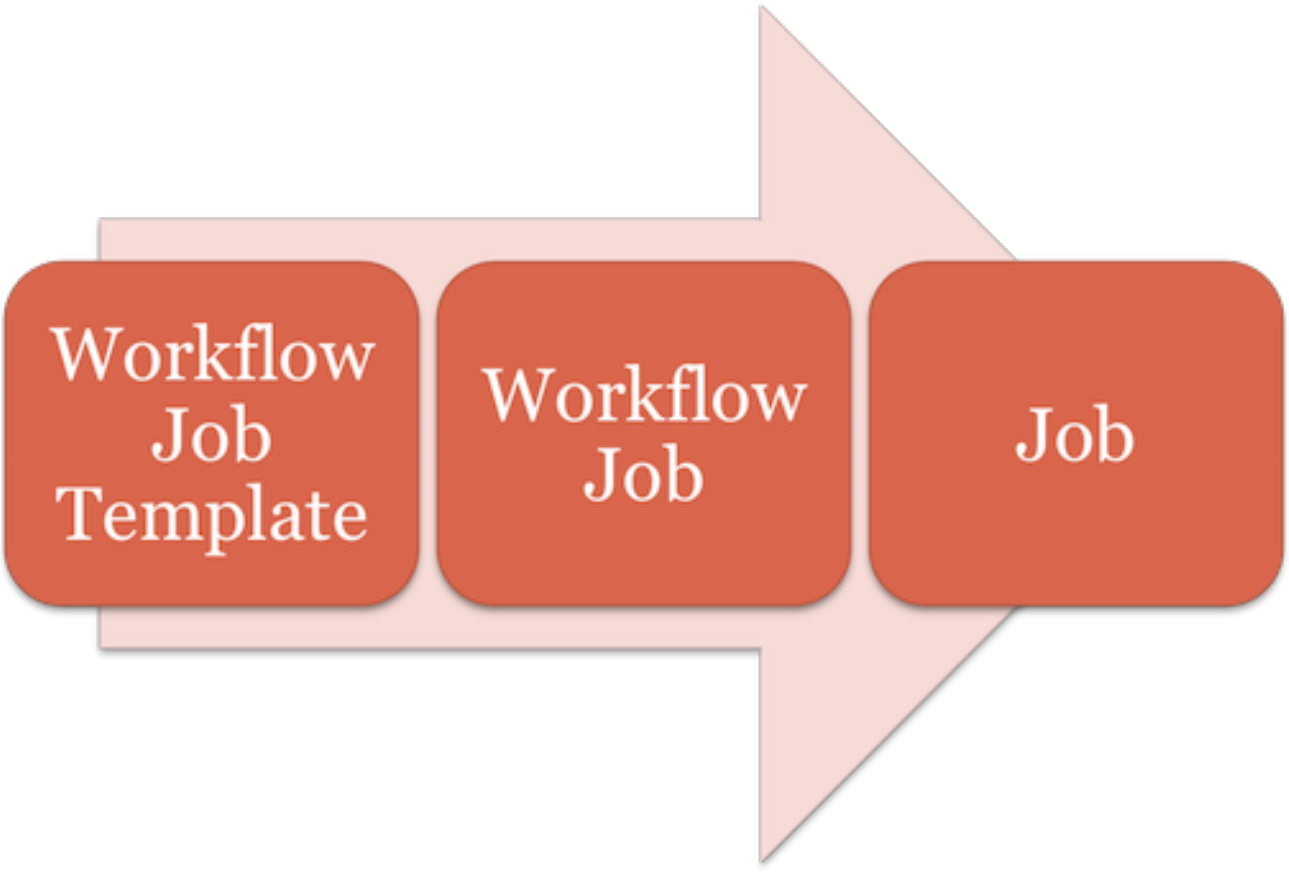


- 複数のジョブが 1 つに収束する収束ワークフローを実現できます。このシナリオでは、以下の例に示すように、次のジョブが実行される前に、いずれかのジョブまたはすべてのジョブを完了する必要があります。

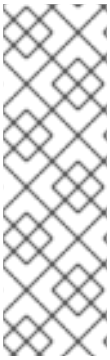
    - この例では、Automation controller は最初の 2 つのジョブテンプレートを並行して実行します。両方が指定どおりに終了し、成功すると、3 番目のダウンストリーム (コンバージェンスノード) がトリガーされます。

- インベントリーと Survey のプロンプトは、ワークフロージョブテンプレートに含まれるワークフローノードに適用されます。

- API から起動する場合、**get** コマンドを実行すると、警告のリストが表示され、欠落しているコンポーネントが強調表示されます。次の図は、ワークフロージョブテンプレートの基本的なワークフローを示しています。



- 複数のワークフローの同時起動や、起動のタイミングのスケジュールを設定できます。ジョブテンプレートと同様に、ジョブの完了時などの通知をワークフローに設定できます。



注記

ジョブスライスは、ジョブの実行を水平方向にスケーリングすることを目的としています。

ジョブテンプレートでジョブスライスを有効にすると、動作するインベントリーが起動時に設定されたスライスの数に分割されます。次に、スライスごとにジョブを開始します。

詳細は、ジョブスライス セクションを参照してください。

- 再帰的なワークフローを構築することができますが、automation controller がエラーを検出した場合には、ネスト化されたワークフローが実行しようとしたタイミングでワークフローは停止します。
- サブワークフローのジョブで収集されたアーティファクトは、ダウンストリームのノードには渡されません。
- インベントリーは、ワークレベルとして設定することもインベントリーの起動プロンプトとして設定することも可能です。
- 起動時に、**ask_inventory_on_launch=true** が指定されているワークフロー内のジョブテンプレートはすべて、ワークフローレベルのインベントリーを使用します。
- インベントリーを求めないジョブテンプレートでは、ワークフローインベントリーを無視し、独自のインベントリーに対して実行されます。
- ワークフローがインベントリーを求める場合には、スケジュールおよび他のワークフローノードでそのインベントリーを提供することができます。
- ワークフロー収束シナリオでは、**set_stats** データは、定義なしの状態でマージされるため、一意のキーを設定する必要があります。

22.2. ワークフローの追加変数

ワークフローは、Survey を使用して、ワークフロー内の Playbook で使用される **extra_vars** と呼ばれる変数を指定します。Survey 変数は、ワークフロージョブテンプレートで定義された **extra_vars** と結合され、ワークフロージョブ **extra_vars** に保存されます。ワークフロージョブ内の **extra_vars** は、ワークフロー内でジョブを生成するときにジョブテンプレート変数と結合されます。

ワークフローは、3 つの追加変数を除き、ジョブテンプレートと同じ変数の優先順位の動作 (階層) を使用します。ジョブテンプレートの追加変数セクションの Automation controller 変数の優先順位階層 を参照してください。追加の 3 つの変数には次のものが含まれます。

- ワークフロージョブテンプレートの追加変数
- ワークフロージョブテンプレート survey (デフォルト)
- ワークフロージョブ起動の追加変数

ワークフローに含まれるワークフローは同じ変数の優先順位に従い、特にプロンプトが要求された場合、または Survey の一部として定義された場合にのみ、変数を継承します。

ワークフローの **extra_vars** のほかにも、ワークフローの一部として実行されるジョブおよびワークフローは、ワークフローの親ジョブに含まれるアーティファクトディクショナリーの変数を継承できます (また、ブランチの祖先のアップストリームと組み合わされます)。これらは、**set_stats** Ansible モ

ジュール で定義できます。

Playbook で **set_stats** モジュールを使用すると、別のジョブがダウンストリームで使用できるように結果を生成できます。

例

統合実行の成功または失敗についてユーザーに通知します。この例では、ワークフロー内で組み合わせてアーティファクトの受け渡しを実行できる 2 つの Playbook があります。

- nvoke_set_stats.yml: ワークフローの最初の Playbook

```
---
- hosts: localhost
  tasks:
    - name: "Artifact integration test results to the web"
      local_action: 'shell curl -F "file=@integration_results.txt" https://file.io'
      register: result


    - name: "Artifact URL of test results to Workflows"
      set_stats:
        data:
          integration_results_url:  "{{ (result.stdout|from_json).link }}"
```

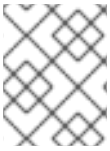
- use_set_stats.yml: ワークフローの 2 番目の Playbook

```
---
- hosts: localhost
  tasks:
    - name: "Get test results from the web"
      uri:
        url: "{{ integration_results_url }}"
        return_content: true
      register: results


    - name: "Output test results"
      debug:
        msg: "{{ results.content }}"
```

set_stats モジュールはこのワークフローを以下のように処理します。

1. 統合結果の内容は Web 上にアップロードされます。
2. 次に **invoke_set_stats*** Playbook により、**set_stats** が起動すると、アップロードされた **integration_results.txt** の URL が作成され、Ansible 変数 integration_results_url に渡されます。
3. ワークフローの 2 番目の Playbook は、Ansible の追加変数 integration_results_url を使用します。これは、uri モジュールを使用して Web を呼び出し、前のジョブテンプレートジョブによってアップロードされたファイルのコンテンツを取得します。次に、取得したファイルの内容を出力します。



注記

アーティファクトを機能させるには、**set_stats** モジュールの **per_host = False`** というデフォルト設定をそのまま使用してください。

22.3. ワークフローの状態

ワークフロージョブには以下の状態が設定されます (失敗状態は含まれなし)。

- 待機中
- 実行中
- 成功 (終了)
- 取り消し
- エラー
- 失敗

ワークフローのスキームでは、ジョブを取り消すとブランチが取り消され、ワークフロージョブを取り消すとワークフロー全体が取り消されます。

22.4. ロールベースのアクセス制御

ワークフロージョブテンプレートを編集および削除するには、管理者のロールが必要です。ワークフロージョブテンプレートを作成するには、組織管理者またはシステム管理者である必要があります。ただし、パーミッションのないジョブテンプレートを含むワークフロージョブテンプレートは、実行できます。システム管理者は空のワークフローを作成し、下位レベルのユーザーに **admin_role** を付与し、その後さらに多くのアクセスを委譲してグラフを構築できます。ジョブテンプレートをワークフロージョブテンプレートに追加するには、ジョブテンプレートへの **実行** アクセス権が必要です。

ユーザーに付与されているパーミッションに応じて、複製コピーの作成やワークフローの再起動など、他のタスクを実行することもできます。再起動またはコピーを作成する前に、ワークフローで使用されるすべてのリソース (ジョブテンプレートなど) に対するアクセス権限が必要です。

詳細は、ロールベースのアクセス制御 を参照してください。

このセクションで説明するタスクの実行の詳細は、管理ガイド を参照してください。

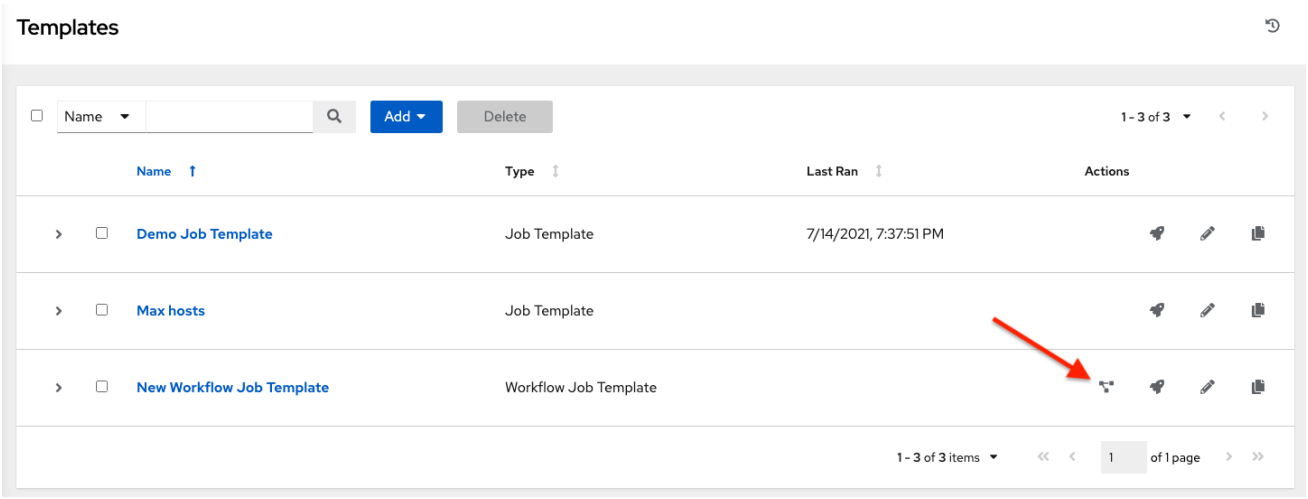
第23章 ワークフロージョブテンプレート

ワークフロージョブテンプレートは、一連の異種リソースをリンクして、リリースプロセスに含まれていたジョブ全体を1つの単位として追跡します。これらのリソースには以下が含まれます。

- ジョブテンプレート
- ワークフロージョブテンプレート
- プロジェクト同期
- インベントリーソース同期

Templates リストビューには、現在使用可能なワークフローテンプレートとジョブテンプレートが表示されます。デフォルトのビューは折りたたまれており (Compact)、テンプレート名、テンプレートの種類、およびそのテンプレートを使用して実行されたジョブのステータスが表示されます。各エントリーの横にある矢印をクリックすると、展開して詳細情報を表示できます。このリストは名前のアルファベット順に並べ替えられていますが、他の条件で並べ替えたり、テンプレートのさまざまなフィールドや属性で検索したりできます。この画面から、ワークフローのジョブテンプレートを起動して ( )、編集し ( )、コピー ( ) します。

ワークフローテンプレートにのみ、ワークフローエディターにアクセスするためのショートカットとしてワークフロービジュアライザーアイコン ( ) があります。

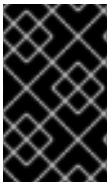


注記

ワークフローテンプレートは、別のワークフローテンプレートの設定要素として使用できます。ワークフローテンプレートでいくつかの設定を指定することで、**Prompt on Launch** を有効にできます。これらの設定は、ワークフロージョブテンプレートレベルで編集できます。これらは、個々のワークフローテンプレートレベルで割り当てられた値には影響しません。詳しい手順は、ワークフロービジュアライザー セクションを参照してください。

23.1. ワークフローテンプレートの作成

新しいワークフロージョブテンプレートを作成するには、次の手順を実行します。

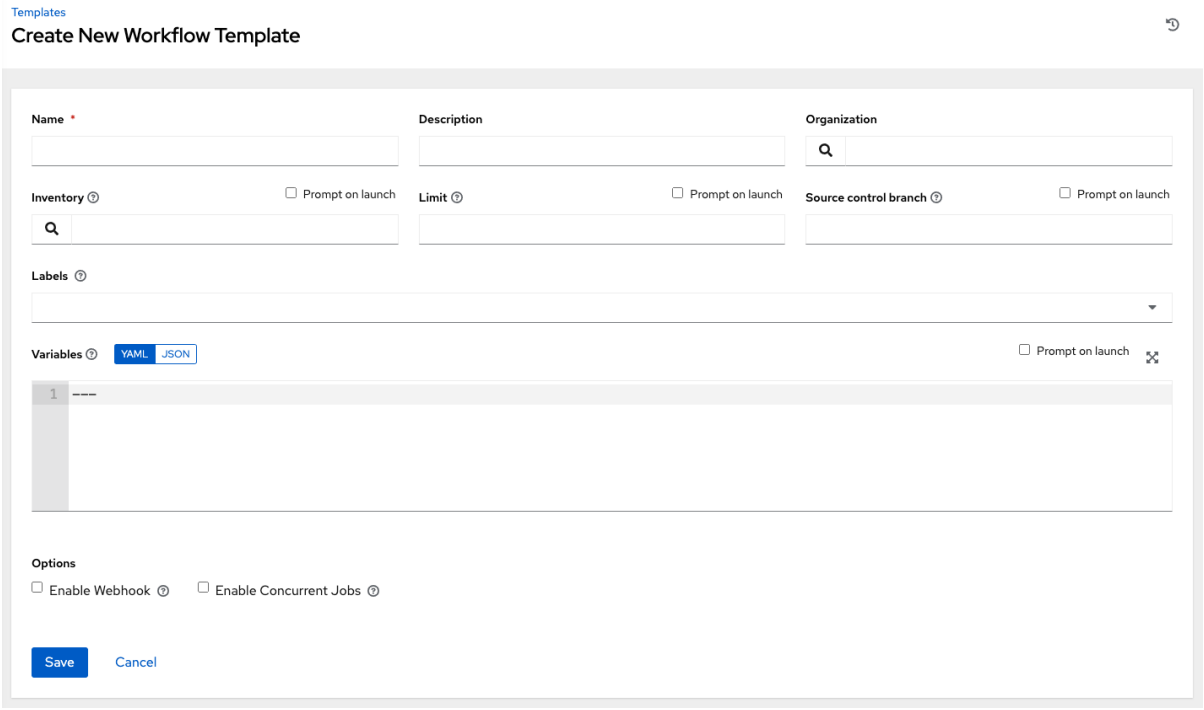


重要

ワークフローテンプレートに制限を設定する場合、**Prompt on launch** をオンにしないと、制限がジョブテンプレートに渡されません。実行している Playbook に制限が必須である場合、これにより Playbook が失敗する可能性があります。

手順

1. **Templates** リストビューで、**Add → Add workflow template** をクリックします。



2. 次のフィールドに適切な詳細を入力します。

注記

フィールドで **Prompt on launch** チェックボックスがオンになっている場合、ワークフローテンプレートを起動するか、別のワークフローテンプレート内でワークフローテンプレートを使用すると、そのフィールドの値の入力を求めるプロンプトが表示されます。ほとんどのプロンプト値は、ジョブテンプレートに設定されている値をオーバーライドします。次の表に例外を示します。

フィールド	オプション	Prompt on Launch
Name	ジョブの名前を入力します。	該当なし
Description	必要に応じて、任意の説明を入力します (オプション)。	該当なし
Organization	ログインしているユーザーが使用できる組織から、このテンプレートで使用する組織を選択します。	該当なし

フィールド	オプション	Prompt on Launch
Inventory	必要に応じて、ログインしたユーザーが使用できるインベントリーから、このテンプレートで使用するインベントリーを選択します。	はい
Limit	Playbook が管理または影響を与えるホストのリストをさらに制限するためのホストのパターンを指定します。複数のパターンをコロン (:) で区切ることができます。コア Ansible と同じです。 ● a:b は a または b のグループに含まれるという意味です。 ● a:b:&c は、a または b に含まれるが、c には必ず含まれなければならないという意味です。 ● a:!b は a にはあるが、b には絶対にないという意味です。 詳細は、Ansible ドキュメントの Patterns: targeting hosts and groups を参照してください。	はい 選択すると、デフォルト値が指定されている場合でも、起動時に制限を選択するように求めます。
ソースコントロールのブランチ	ワークフローのブランチを選択します。このブランチは、ブランチを要求するすべてのワークフロージョブテンプレートノードに適用されます。	はい

フィールド	オプション	Prompt on Launch
Labels	● 必要に応じて、このワークフロージョブテンプレートを説明するラベル (**dev** や **test** など) を指定します。ラベルを使用して、表示内のワークフロージョブテンプレートと完了したジョブをグループ化およびフィルタリングします。 ● ラベルは、ワークフローテンプレートの追加時に作成されます。ラベルは、ワークフローテンプレートで提供されるプロジェクトを使用して、単一の組織に関連付けられます。組織のメンバーは、編集パーミッション (管理者ロールなど) を持っている場合、ジョブテンプレートにラベルを作成できます。 ● ジョブテンプレートを保存すると、ワークフロージョブテンプレートの **Details** ビューにラベルが表示されます。 ● ラベルはワークフローテンプレートにのみ適用され、ワークフローで使用されるジョブテンプレートノードには適用されません。 ● ラベルの横にある✖を選択して削除します。ラベルが削除されると、そのラベルはその特定のジョブまたはジョブテンプレートとの関連付けが解除されますが、ラベルを参照する他のジョブには関連付けられたままになります。	はい 選択すると、デフォルト値が指定されている場合でも、必要に応じて起動時に追加のラベルを指定するように求められます。- 既存のラベルは削除できません。✖を選択すると、新たに追加されたラベルだけが削除され、既存のデフォルトラベルは削除されません。

フィールド	オプション	Prompt on Launch
Variables	• 追加のコマンドライン変数を Playbook に渡します。 これは、Ansible ドキュメントの Controlling how Ansible behaves: precedence rules に記載されている ansible-playbook の "-e" または "--extra-vars" コマンドラインパラメーターです。- YAML または JSON を使用してキーまたは値のペアを指定します。これらの変数には優先順位の最大値があり、他の場所で指定された他の変数をオーバーライドします。**git_branch: production release_version: 1.5** は、値の例です。	はい スケジュールで extra_vars を指定できるようにするには、ジョブテンプレートの変数に対して **Prompt on launch** を選択するか、ジョブテンプレートで Survey を有効にする必要があります。回答された Survey の質問は **extra_vars** になります。追加変数の詳細は、追加変数 を参照してください。
Job tags	**Create** ドロップダウンを入力して選択し、Playbook のどの部分を実行するかを指定します。詳細と例は、Ansible ドキュメントの Tags を参照してください。	はい
Skip Tags	**Create** ドロップダウンを入力して選択し、スキップする Playbook の特定のタスクまたは部分を指定します。詳細と例は、Ansible ドキュメントの Tags を参照してください。	はい

3. 必要に応じて、このテンプレートを起動するための次の **Options** を指定します。

- ワークフロージョブテンプレートを起動するために使用する定義済みの SCM システム Web サービスと接続する機能をオンにするには、**Enable Webhooks** をオンにします。GitHub と GitLab がサポートされている SCM システムです。
    - Webhook を有効にすると、他のフィールドが表示され、以下の追加情報の入力を求められます。
        - **Webhook Service**: Webhook からリッスンするサービスを選択します
        - **Webhook Credential** : オプションで、Webhook サービスにステータス更新を送信するために使用する認証情報として GitHub または GitLab パーソナルアクセストークン (PAT) を指定します。詳細は、認証情報タイプ を参照して作成してください。

- **Save** をクリックすると、追加のフィールドが入力され、ワークフロービジュアライザーが自動的に開きます。
  - **Webhook URL**: POST 要求を送信する Webhook サービスの URL が自動的に入力されます。
  - **Webhook キー**: Webhook サービスが automation controller に送信するペイロードに署名する際に使用するための、生成された共有シークレット。このサービスからの Webhook が Automation controller で受け入れられるように、Webhook サービスの設定で指定する必要があります。Webhook の設定の関連情報については、Webhook の使用 を参照してください。
    Enable Concurrent Jobsにチェックを入れて、このワークフローの同時実行を可能にします。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。

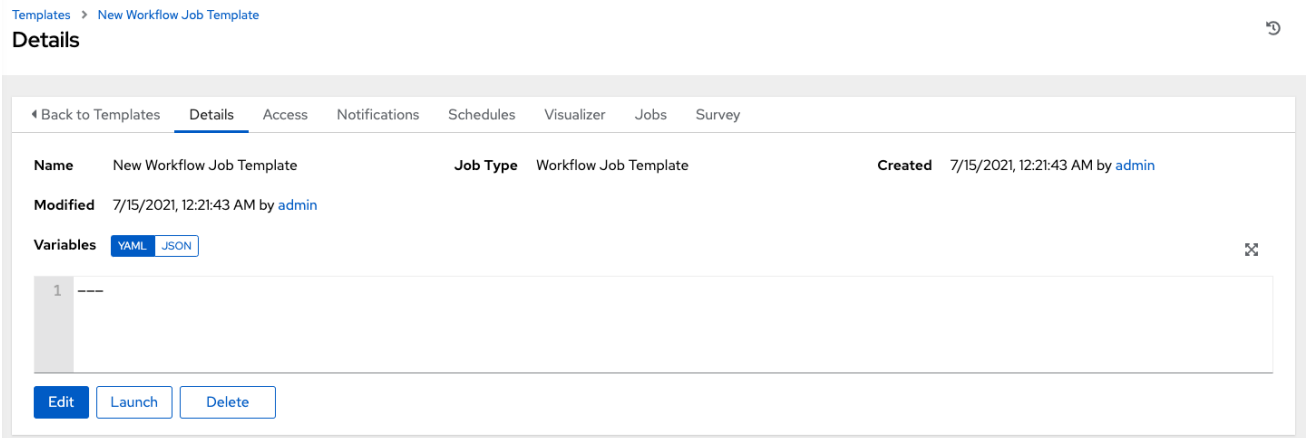
4. ワークフローテンプレートの設定が完了したら、**Save** をクリックします。

テンプレートを保存すると、ワークフローテンプレートページが終了し、ワークフロービジュアライザーが開き、ワークフローを構築できるようになります。詳細は、ワークフロービジュアライザー セクションを参照してください。それ以外の場合は、次のいずれかの方法を選択します。

- ワークフロービジュアライザーを閉じて、新しく保存したテンプレートの **Details** タブに戻ります。そこで次のタスクを実行できます。
  - パーミッション、通知、スケジュール、Survey を確認、編集、追加する
  - 完了したジョブの表示
  - ワークフローテンプレートを構築する
- **Launch** をクリックしてワークフローを起動します。

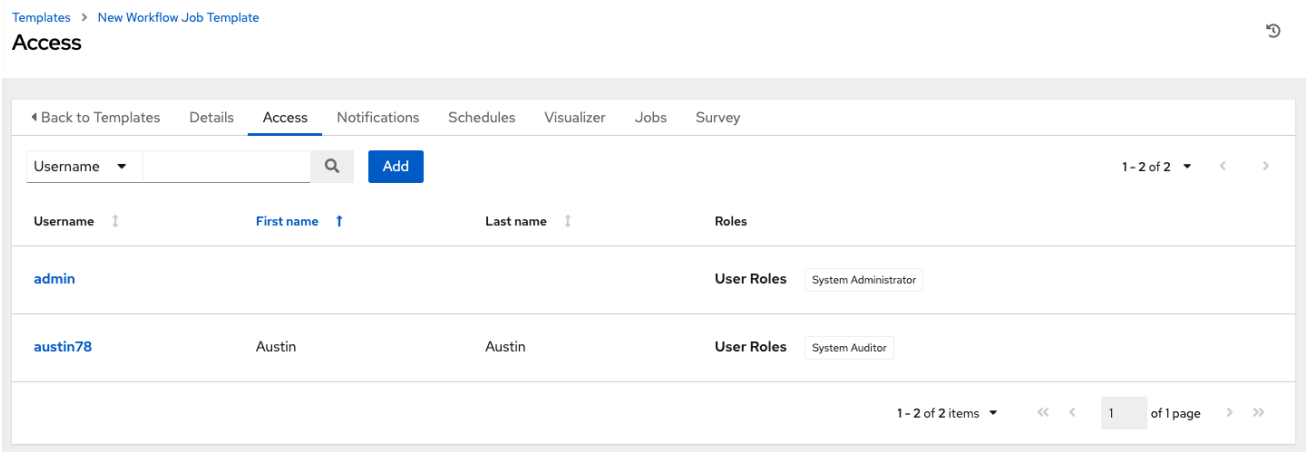
注記

起動する前にテンプレートを保存しないと、**Launch** は無効のままになります。**Notifications** タブは、テンプレートを保存した後にのみ表示されます。



23.2. 権限の使用

Access タブをクリックして、ユーザーおよびチームメンバーに関連付けられたパーミッションを確認、付与、編集、削除します。



Add をクリックして、プロンプトに従ってこのワークフローテンプレートに新しいパーミッションを作成し、それに応じてパーミッションを割り当てます。

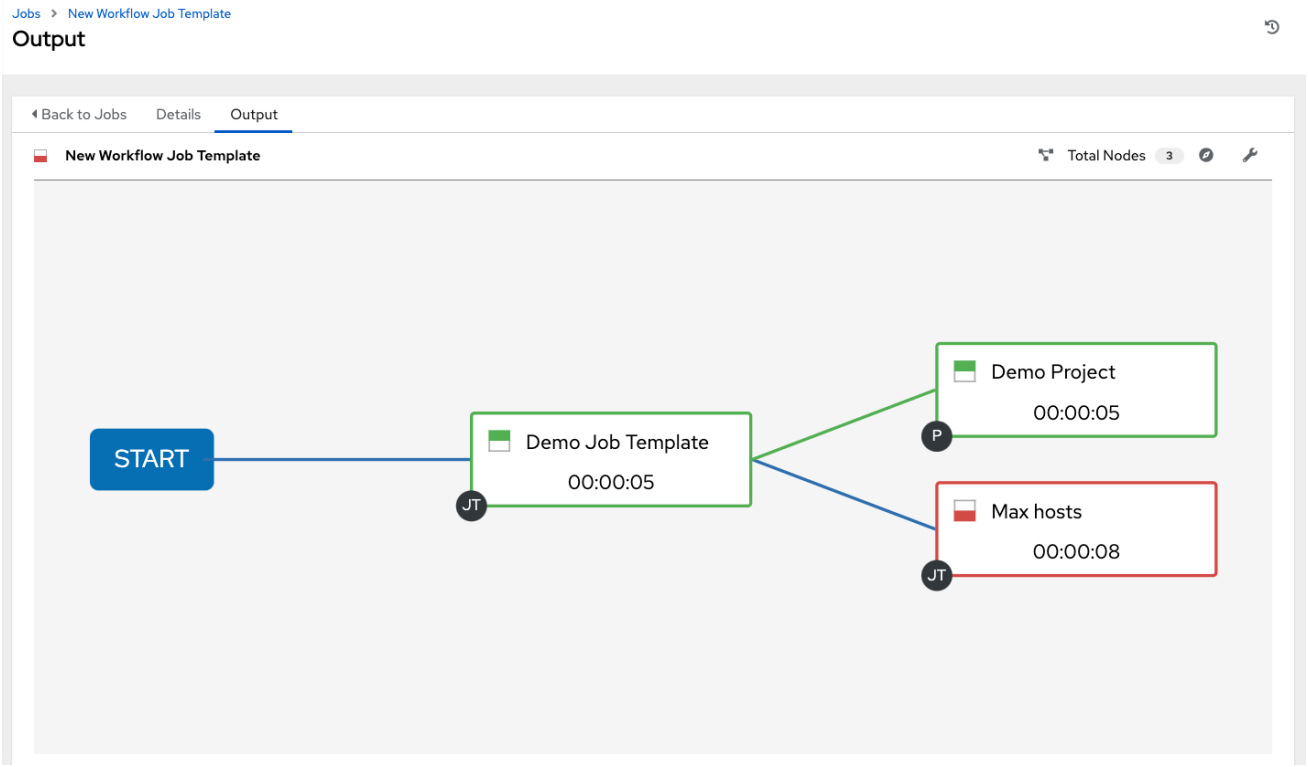
23.3. 通知の使用

ワークフロージョブテンプレートでの通知の操作の詳細は、通知の操作 を参照してください。

23.4. 完了したワークフロージョブの表示

Jobs タブには、実行されたジョブテンプレートのリストが表示されます。各ジョブの横にある アイコンをクリックして、各ジョブの詳細を表示します。

このビューから、ジョブ ID、ワークフロージョブの名前をクリックすると、その内容が図で表示されます。次の例は、ワークフロージョブのジョブの詳細を示しています。



ノードには、識別しやすいようにラベルが付けられています。詳細は、ワークフロービジュアライザー セクションの凡例を参照してください。

23.5. ワークフロージョブテンプレートのスケジュール設定

Schedules タブを選択して、特定のワークフロージョブテンプレートのスケジュールにアクセスします。

ワークフロージョブテンプレートの実行のスケジュール設定の詳細は、ジョブテンプレートのスケジュール設定 セクションを参照してください。

ネストされたワークフローで使用されるワークフロージョブテンプレートに Survey が含まれている場合、またはインベントリーオプションで **Prompt on Launch** が選択されている場合、スケジュールフォームの **Save** および **Cancel** オプションの横に **PROMPT** オプションが表示されます。**Prompt** をクリックすると、オプションの **INVENTORY** ステップが表示され、インベントリーを指定または削除するか、変更せずにこのステップをスキップできます。

23.6. ワークフロージョブテンプレートでの SURVEY

Run または **Check** のジョブタイプを含むワークフローでは、ワークフロージョブテンプレートの作成画面または編集画面で Survey を設定する方法が提供されます。

ワークフロージョブテンプレートでの Survey の作成方法やオプションの Survey の質問など、ジョブ Survey の詳細は、ジョブテンプレートでのSurvey セクションを参照してください。

23.7. ワークフロービジュアライザー

ワークフロービジュアライザーは、ジョブテンプレート、ワークフロー、プロジェクトの同期、インベントリーの同期を図を使用して連結し、ワークフローのテンプレートを構築します。ワークフローテンプレートを構築する前に、親ノード、子ノード、兄弟ノードのさまざまなシナリオに関連する考慮事項は ワークフロー セクションを参照してください。

23.7.1. ワークフローの構築

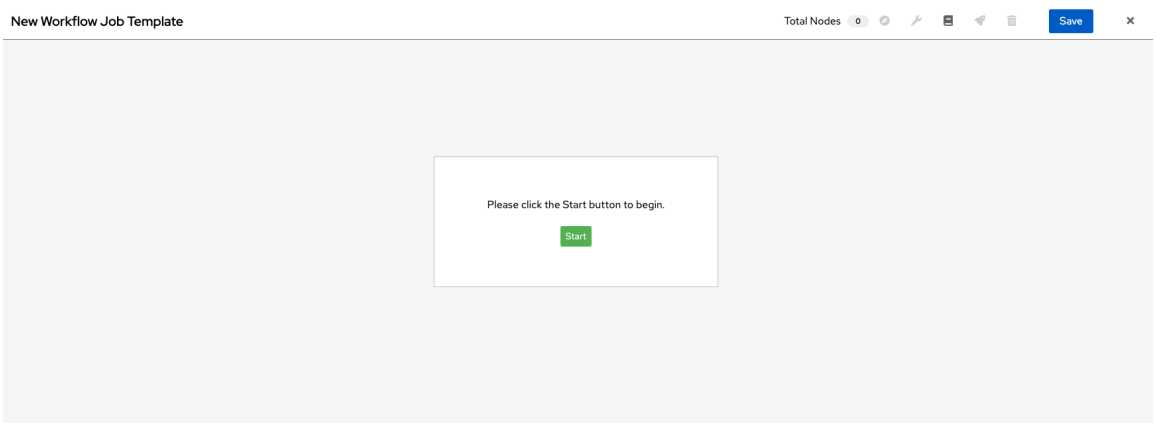
次のノードタイプの 2 つ以上を任意に組み合わせて設定して、ワークフローを構築できます。

- テンプレート (ジョブテンプレートまたはワークフロージョブテンプレート)
- プロジェクトの同期
- インベントリー同期
- 承認

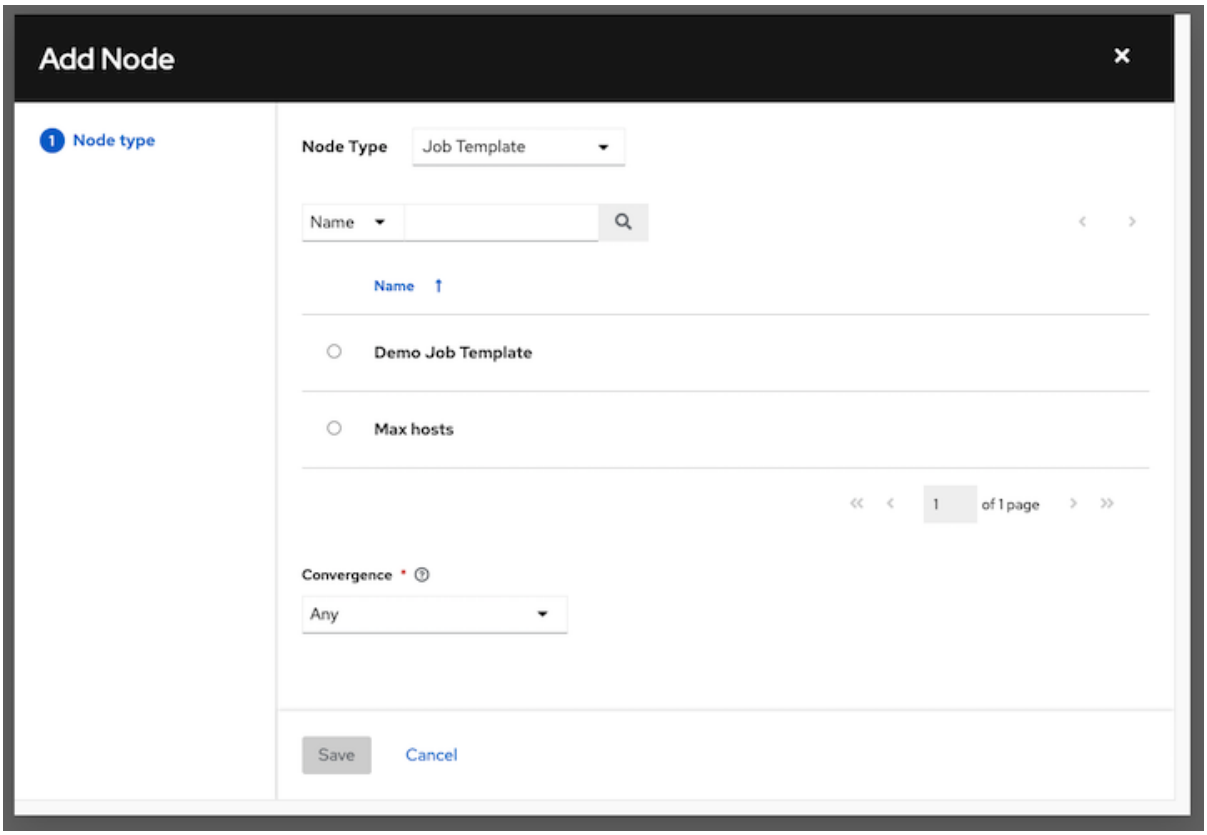
各ノードは長方形で表され、関係や関連するエッジタイプは線 (またはリンク) で表され、それぞれを連結します。

手順

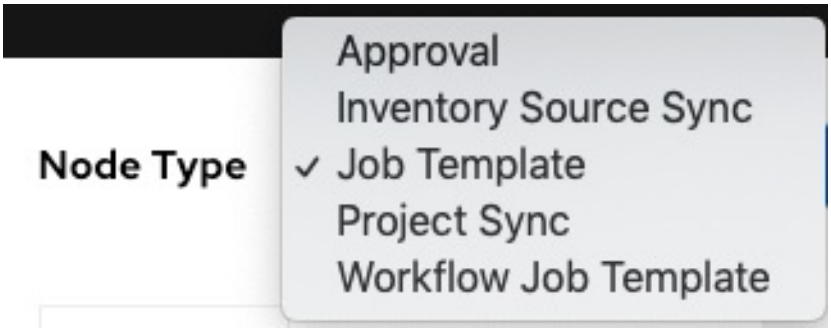
1. ワークフロービジュアライザーを起動するには、次のいずれかの方法を使用します。
   a. ナビゲーションパネルから、**Resources → Templates** を選択します。
      i. ワークフローテンプレートを選択し、**Details** タブで **Edit** をクリックします。
      ii. **Visualizer** タブを選択します。
   b. **Templates** リストビューで、 アイコンをクリックします。



2. **Start** をクリックして、ワークフローに追加するノードのリストを表示します。



3. **Node Type** リストから、追加するノードのタイプを選択します。



- **承認** ノードを選択した場合の詳細は 承認ノード を参照してください。
  ノードを選択すると、そのノードに関連する有効なオプションが表示されます。

注記

ワークフローグラフにデータを入力するときに、デフォルトのインベントリーがないジョブテンプレートを選択すると、親ワークフローのインベントリーが使用されます。ジョブテンプレートでは認証情報は必須ではありませんが、パスワードを必要とする認証情報が含まれている場合、その認証情報がプロンプトの認証情報に置き換えられない限り、ワークフローのジョブテンプレートを選択できません。

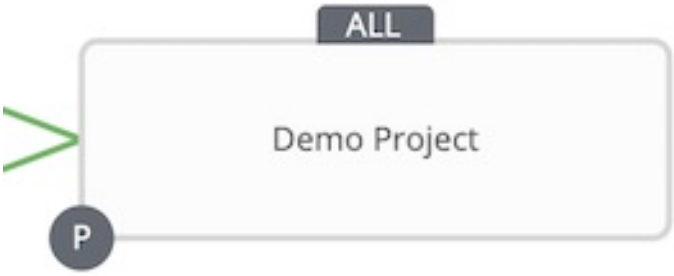
4. ノードタイプを選択すると、ワークフローの構築が開始され、選択したノードに対して実行するアクションのタイプを指定する必要があります。このアクションはエッジタイプとも呼ばれます。

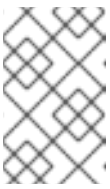
5. ノードが root ノードの場合には、エッジタイプはデフォルトで **Always** になり、編集できません。後続のノードでは、次のシナリオ (エッジタイプ) のいずれかを選択して、それぞれに適用できます。

   - **Always**: 成功または失敗にかかわらず、実行を継続します。
   - **On Success**: 正常に完了したら、次のテンプレートを実行します。
   - **On Failure**: 失敗後、別のテンプレートを実行します。

6. **Convergence** フィールドから、ノードが収束ノードである場合のノードの動作を選択します。

   - **Any** はデフォルトの動作で、次の収束ノードをトリガーする前に、任意のノードが指定どおりに完了できます。1 つの親のステータスがこれらの実行条件のいずれかを満たしている限り、**すべて** の子ノードが実行されます。**任意** のノードではすべてのノードが完了させる必要がありますが、想定した結果で完了させる必要があるのは 1 つのノードのみです。
   - 次のノードを収束してトリガーする前に、すべてのノードが指定どおりに完了させるようにするには、**All** を選択します。**all*** ノードの目的は、子ノードが実行できるように、親ノードすべてを想定の結果で完了させることです。ワークフローは、子ノードを実行するためにすべての親が期待どおりに動作することを確認します。それ以外の場合は、子ノードは実行されません。
     選択すると、グラフィカルビューでノードに **ALL** というラベルが付けられます。



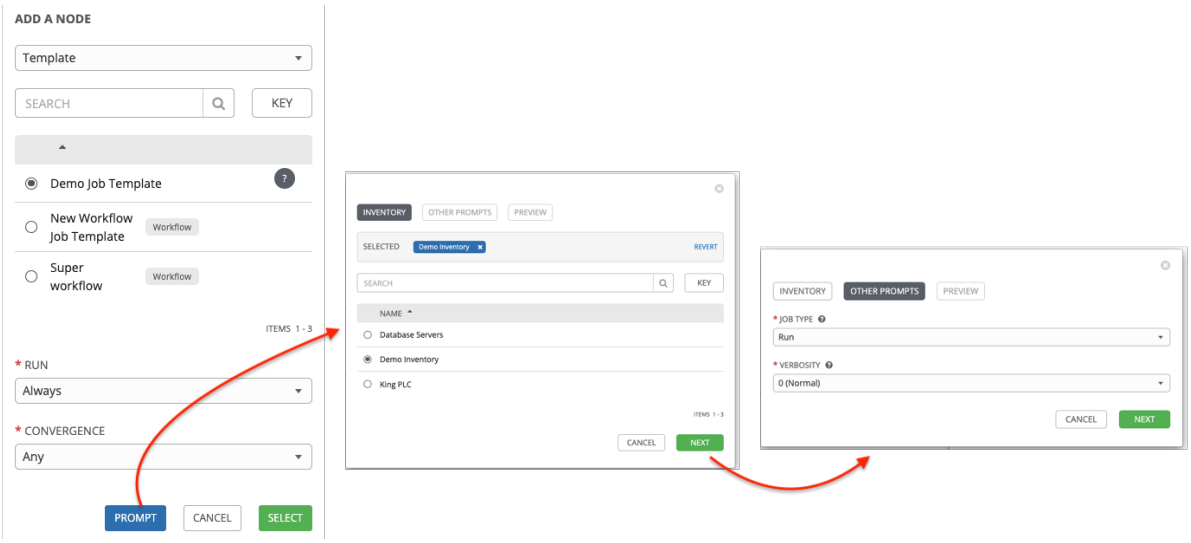


注記

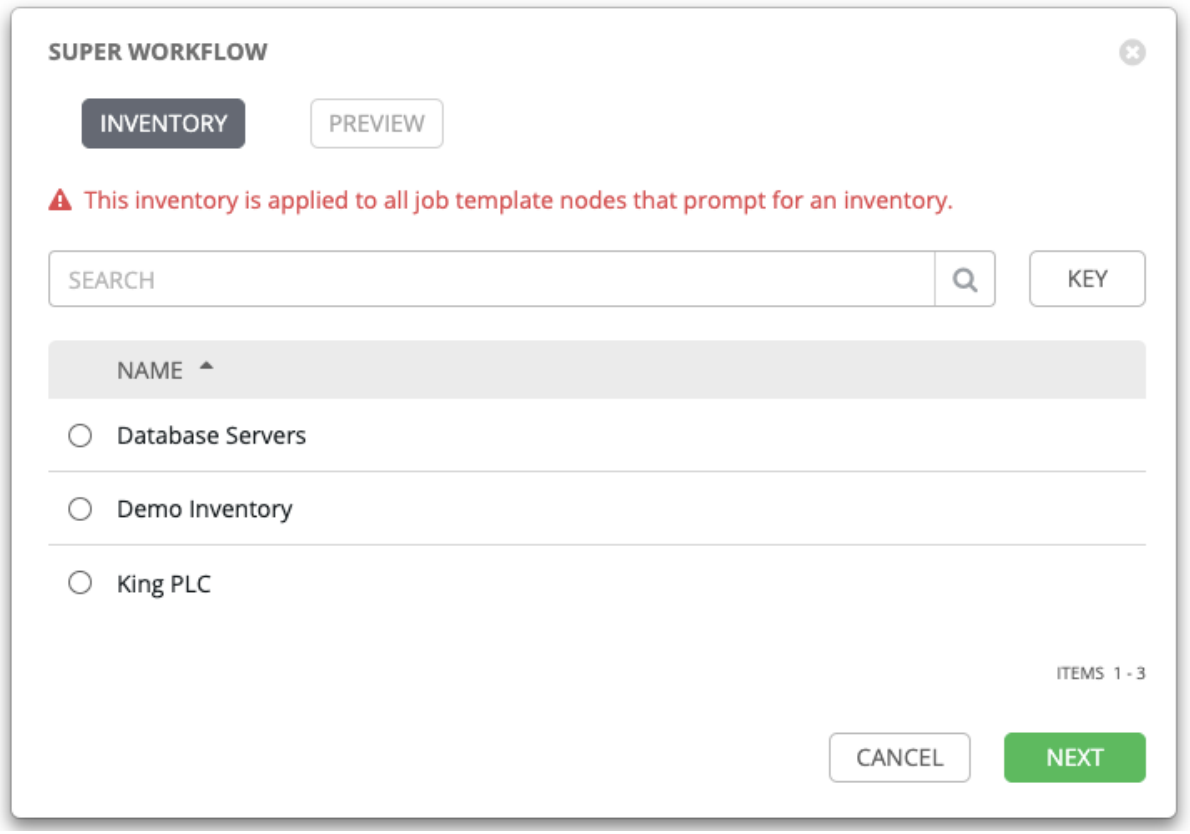
ノードがルートノードであるか、またはノードが収束していないノードである場合、Convergence ルールの設定は適用されません。その動作は、それをトリガーするアクションによって決定されるためです。

7. ワークフローで使用されるジョブテンプレートのパラメーターのいずれかで **Prompt on Launch** が選択されている場合、**Prompt** オプションが表示され、ノードレベルでそれらの値を変更できるようになります。ウィザードを使用して各タブの値を変更し、**Preview** タブで

Confirm をクリックします。



ワークフローで使用されるワークフローテンプレートのインベントリーオプションとして **Prompt on Launch** が選択されている場合は、ウィザードを使用してプロンプトでインベントリーを提供します。親ワークフローに独自のインベントリーがある場合、ここで提供されるインベントリーは上書きされます。



注記

詳細情報の入力を求める必須フィールドがあるが、デフォルトがないワークフロージョブテンプレートの場合、ノードの作成時に **Select** オプションを有効にする前にそれらの値を指定する必要があります。

次の 2 つのケースでは、**PROMPT** オプションによって値が指定されるまで、**SELECT** オプションが無効になります。

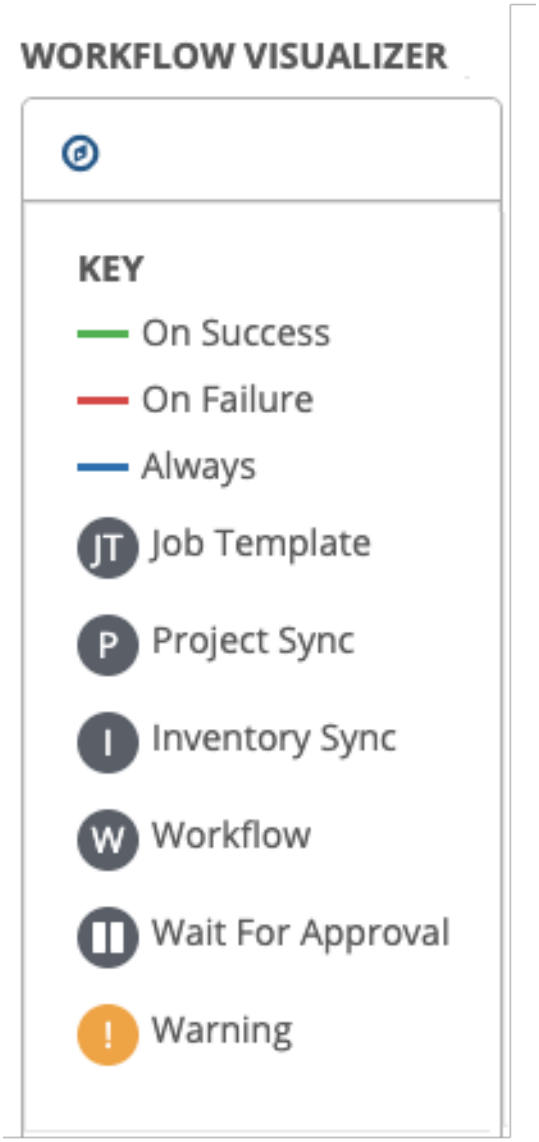
a. ワークフロージョブテンプレートで **Prompt on Launch** チェックボックスをオンにし、デフォルトを指定しない場合。

b. 必須であるがデフォルトの回答を提供しない Survey の質問を作成する場合。

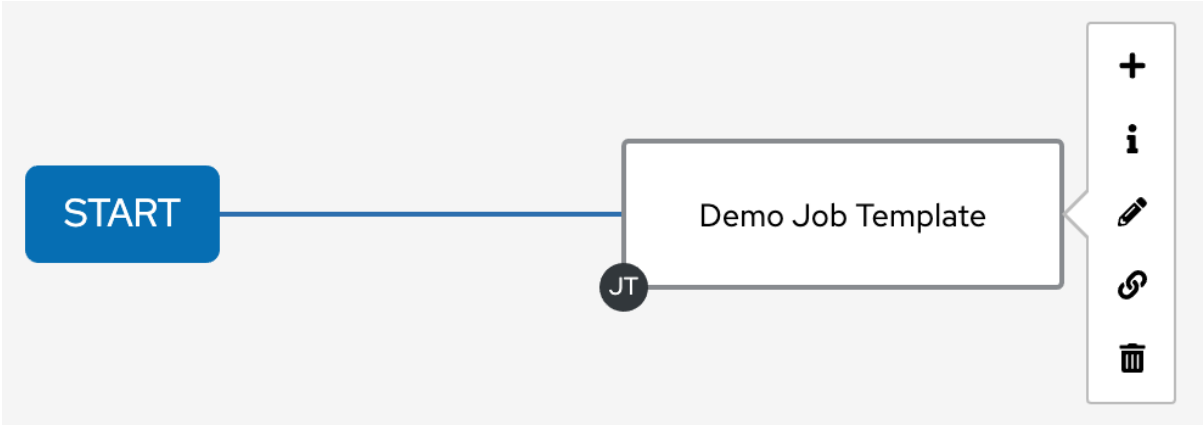
ただし、認証情報の場合はこの限りではありません。ノードの作成時にノードの起動に必要なものをすべて指定する必要があるため、ワークフローノードの作成時には、起動時にパスワードを必要とする認証情報は許可されません。ワークフロージョブテンプレートで認証情報の入力を求められた場合、Automation controller でパスワードを必要とする認証情報を選択できません。

そのノードでの変更を適用するには、プロンプトウィザードが閉じたときに **SELECT** をクリックする必要もあります。それ以外の場合、加えた変更はジョブテンプレートに設定された値に戻ります。

ノードが作成されると、そのジョブタイプのラベルが付けられます。各ワークフローノードに関連付けられたテンプレートは、進行中に選択された実行シナリオに基づいて実行されます。コンパス ( ) アイコンをクリックすると、各実行シナリオとそのジョブタイプの凡例が表示されます。

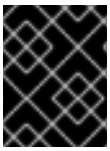


8. ノードの上にマウスを移動して、別のノードの追加、ノードに関する情報表示、ノードの詳細の編集、既存のリンクの編集、選択したノードの削除が可能です。



9. ノードを追加または編集したら、**SELECT** をクリックして変更を保存し、グラフィカルビューにレンダリングします。ワークフローを構築する可能な方法については、 ノードの構築シナリオ を参照してください。

10. ワークフロージョブテンプレートのビルドが完了したら、 **Save** をクリックしてワークフロージョブテンプレート全体を保存し、新しいワークフローテンプレートの詳細ページに戻ります。



重要

Close をクリックしても作業内容は保存されませんが、ワークフロービジュアライザー全体が閉じられるため、再度開始する必要があります。

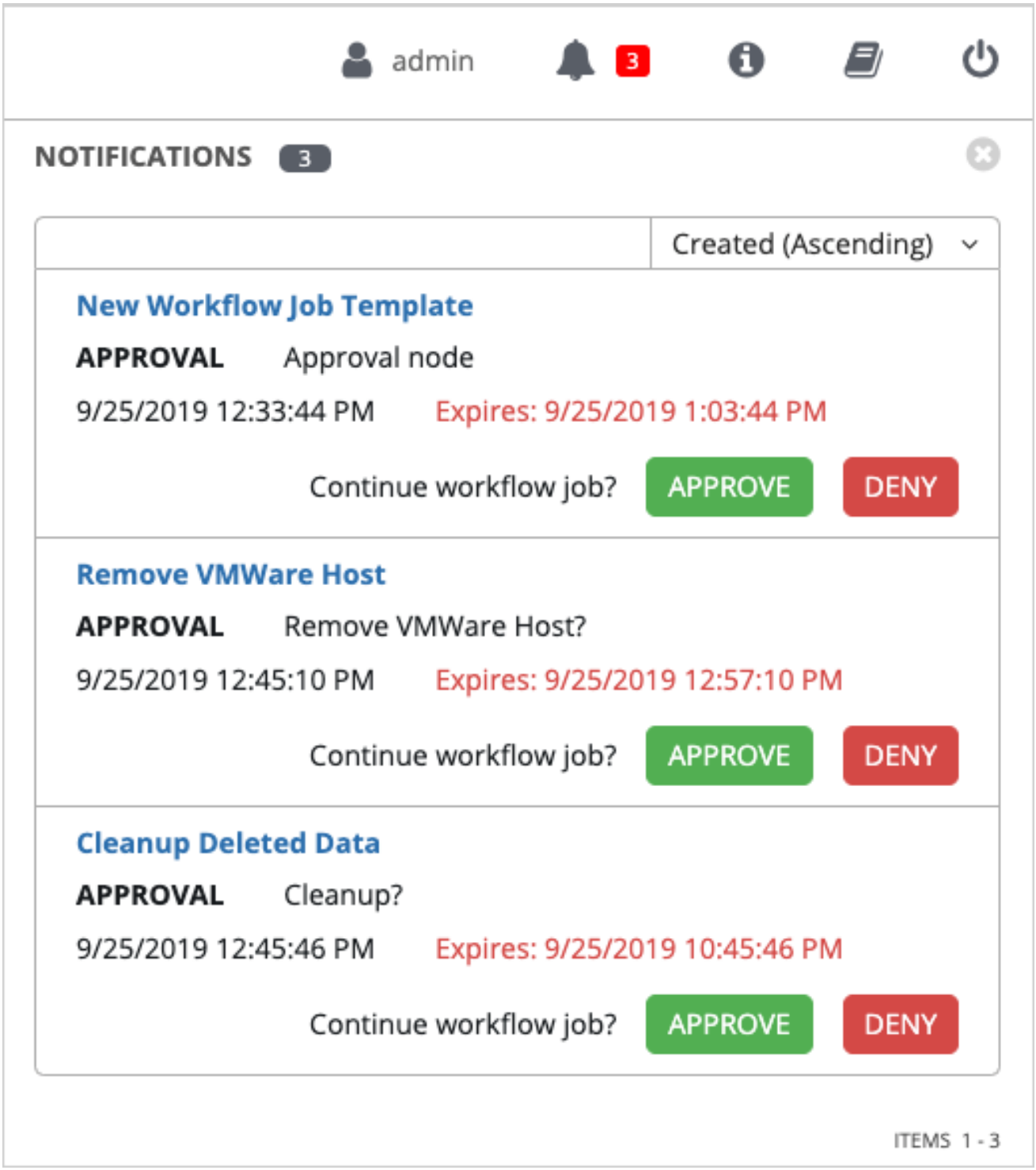
23.7.2. 承認ノード

Approval ノードを選択すると、ワークフローを進めるために介入が必要になります。これは、ワークフローで次の Playbook に進むことを承認できるように、Playbook 間でワークフローを一時停止する手段として機能します。これにより、ユーザーは指定された介入時間が与えられますが、別のトリガーを待たずにできるだけ早く続行できるようになります。

タイムアウトのデフォルトはなしですが、リクエストの有効期限が切れて自動的に拒否されるまでの時間を指定できます。承認ノードの情報を選択して指定すると、横に一時停止アイコンが付いた状態でグラフィカルビューに表示されます。

承認者は、次の基準を満たします。

- 承認ノードを含むワークフロージョブテンプレートを実行できるユーザー。
- 組織管理者以上の権限を持つユーザー (そのワークフロージョブテンプレートに関連付けられた組織に対して)。
- 特定のワークフロージョブテンプレート内で明示的に割り当てられた **Approve** パーミッションを持つユーザー。



保留中の承認ノードが指定された制限時間内に承認されない場合 (有効期限が割り当てられている場合)、または拒否された場合、それらのノードは "timed out" または "failed" としてマークされ、次の "on fail node" または "always node" に移動します。承認された場合は、"on success" パスが選択されます。すでに承認、拒否、またはタイムアウトしたノードに API で **POST** を実行しようとすると、このアクションが冗長であることがエラーメッセージで通知され、それ以上の手順は実行されません。

次の表は、承認ワークフローで許可されるさまざまなレベルのパーミッションを示しています。

SCOPE	ROLE	CREATE WORKFLOW APPROVAL	GRANT APPROVAL	VIEW WORKFLOW APPROVAL	APPROVE/DENY	VIEW WORKFLOW APPROVAL IN ACTIVITY STREAM
Organization	Organization Admin	Yes	Yes	Yes	Yes	Yes
Organization Workflow Job Template	Workflow Admin	Yes	Yes (*)	Yes	Yes	Yes
	Workflow Executor	No	No	Yes	No	Yes
	Workflow Approver	No	No	Yes	Yes	Yes
	Read on Workflow	No	No	Yes (**)	No	Yes (***)
System	System Admin	Yes	Yes	Yes	Yes	Yes
	System Auditor	No	No	Yes	No	Yes
Random user in Organization		No	No	No	No	No
Random user outside Organization		No	No	No	No	No

* Exception: A User with WF Admin permission at the organization level would not be able to grant approval.

** Exception: A User with Read on WF permission at the organization level would not be able to view WF approvals.

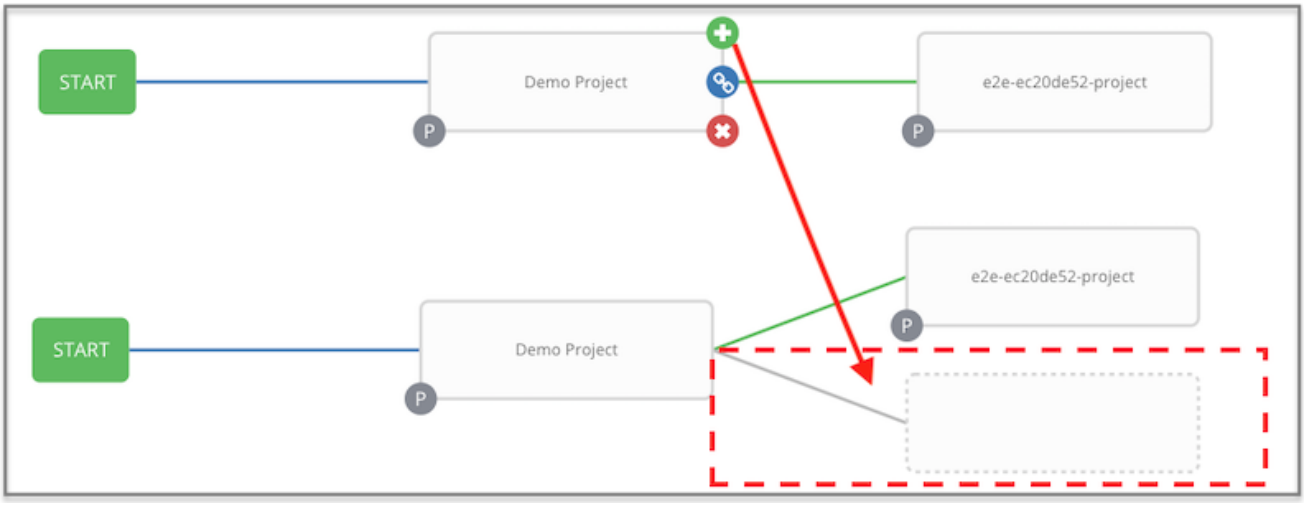
*** Exception: A User with Read on WF permission at the organization level would not be able to view approval jobs in the Activity Stream.

23.7.3. ノードの構築シナリオ

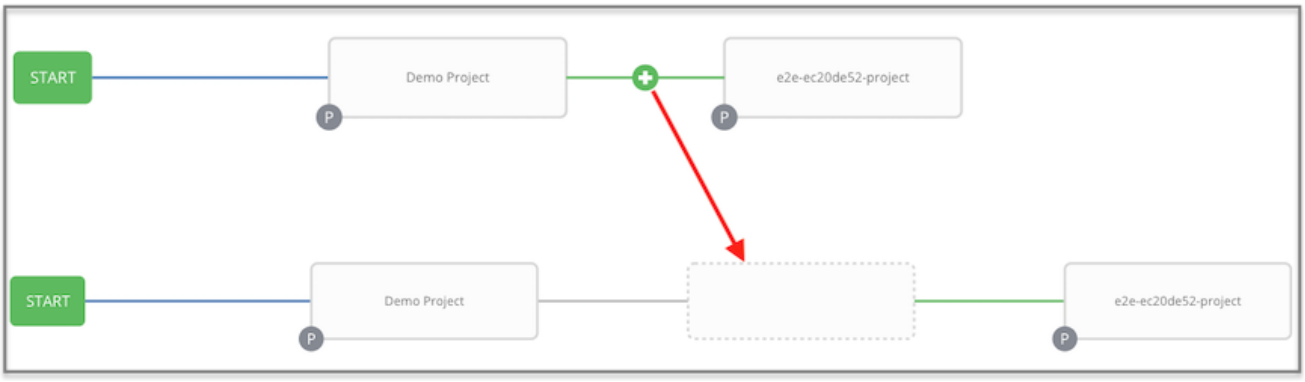
次のシナリオでノードを管理する方法を学習します。

手順

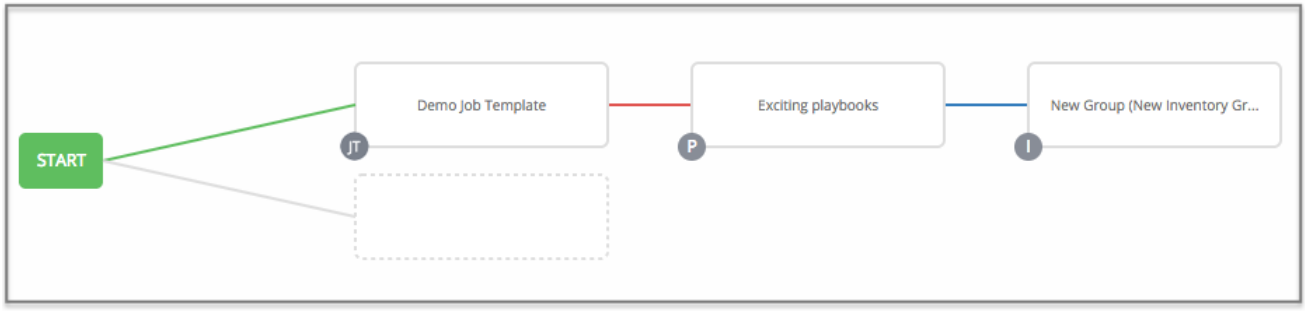
- 親ノードのアイコン (+) アイコンをクリック兄弟ノードを追加します。



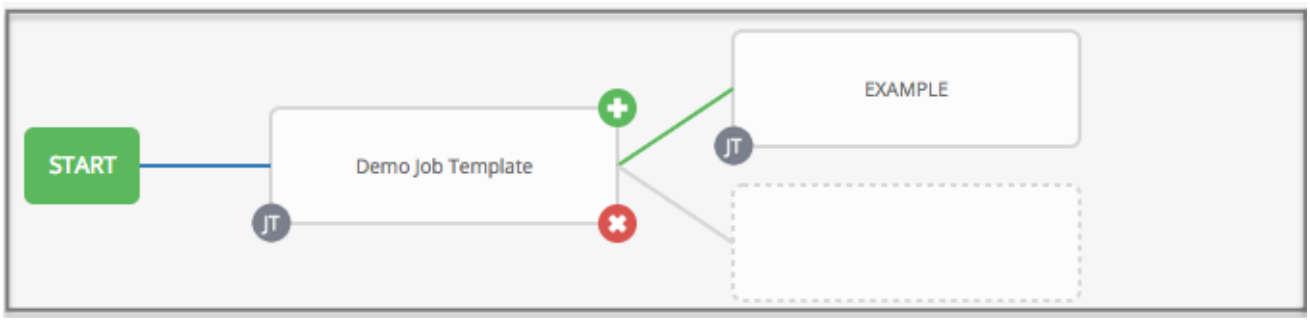
- 2 つのノードを結ぶ線の上にマウスを置き、プラス記号 (+) をクリックして、ノード間に別のノードを挿入します。プラス記号 (+) アイコンをクリックして、2 つのノードの間にノードを自動的に挿入します。



- もう一度 **START** をクリックして、分割シナリオを表すルートノードを追加します。



- 分割シナリオを作成する任意のノードで、分割シナリオが開始されるノードの上にマウスを置き、プラス記号 (+) アイコンをクリックします。これにより、同じ親ノードから複数のノードが追加され、兄弟ノードが作成されます。

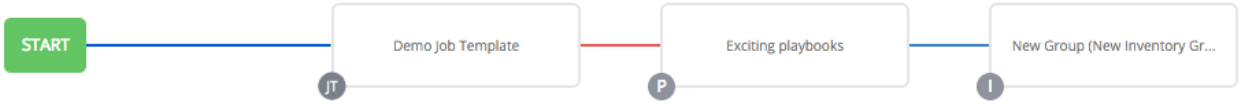


注記

新しいノードを追加する場合、**PROMPT** オプションはワークフローテンプレートにも適用されます。ワークフローテンプレートは、インベントリーと Survey を要求します。

- 次のいずれかの方法を使用して、最後に挿入したノードを元に戻すことができます。
  - 選択を行わずに別のノードをクリックします。
  - **Cancel** をクリックします。

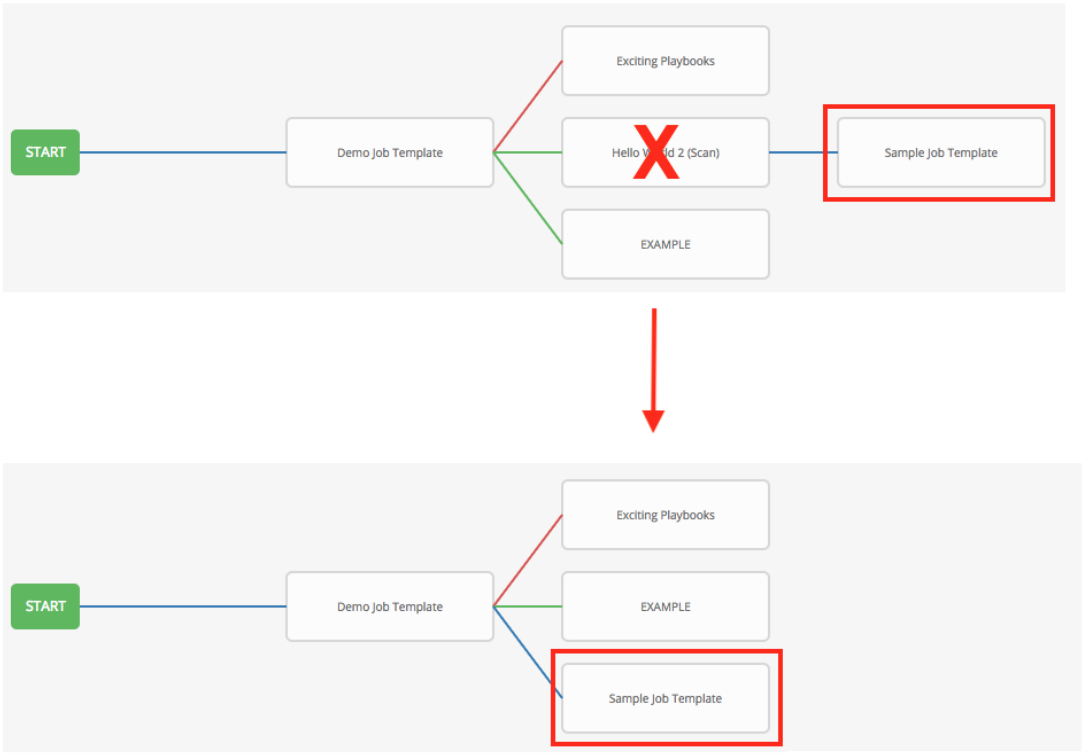
次のワークフロー例には、ジョブテンプレートによって開始される 3 種類のジョブすべてが含まれています。実行に失敗した場合は、同期ジョブを保護する必要があります。失敗したか成功したかに関係なく、インベントリー同期ジョブに進みます。



コンパス ( ) アイコンをクリックしてキーを参照し、グラフィック描写に関連付けられたシンボルと色の意味を識別します。

注記

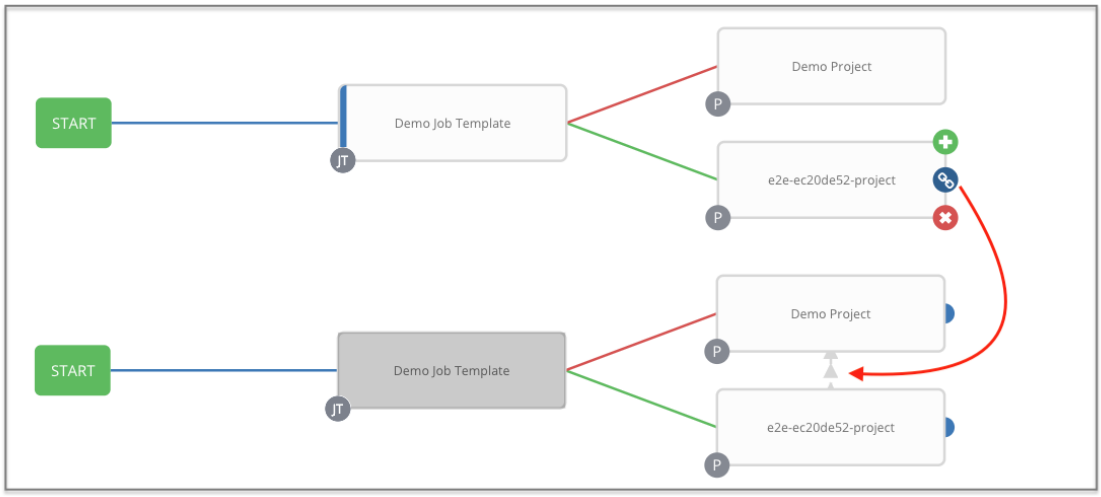
ワークフローで、兄弟ノードにさまざまなエッジタイプが指定されていて、後続のノードがアタッチされたノードを削除した場合には、アタッチされたノードが自動的に兄弟ノードセットと結合されて、そのエッジタイプを保持します。



23.7.4. ノードの編集

手順

- 次のいずれかの方法を使用してノードを編集します。
  - ノードを編集する場合は、編集するノードをクリックします。ペインには現在の選択内容が表示されます。変更を加えて **Select** をクリックして、変更をグラフィカルビューに適用します。
  - 既存のリンクのエッジタイプ (**success**、**Failure**、**always**) を編集するには、リンクをクリックします。ペインには現在の選択内容が表示されます。変更を加えて **Save** をクリックすると、変更がグラフィカルビューに適用されます。
  - 各ノードに表示されるリンク (🔗) アイコンを使用して、あるノードから別のノードに新しいリンクを追加します。これを行うと、リンクできるノードが強調表示されます。これらのオプションは点線で示されています。無効なオプションは、無効なボックス (ノード) で表現されます。こうすることで、無効なリンクが生成されないようにします。次の例では、リンク先の **e2e-ec20de52-project** の可能なオプションとして **デモプロジェクト** を矢印で示しています。



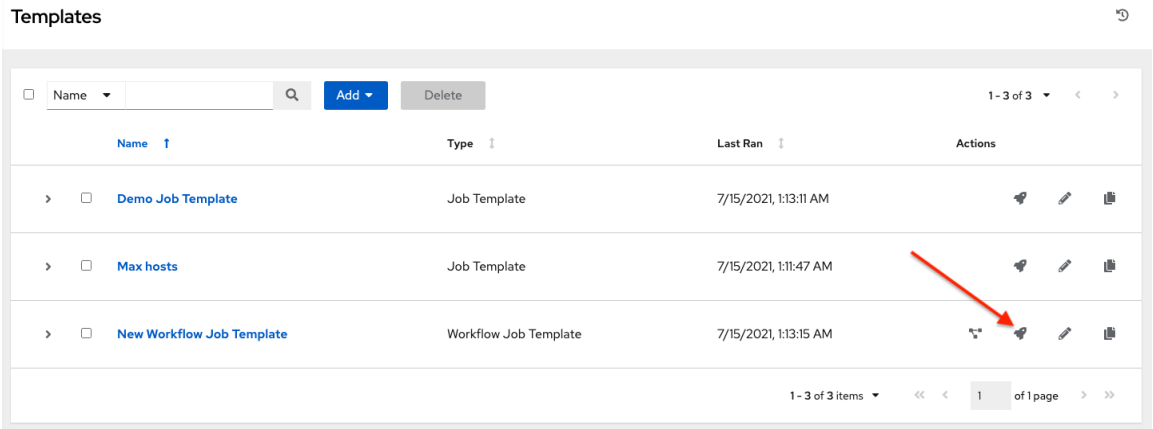
- リンクを削除するには、リンクをクリックして **UNLINK** をクリックします。このオプションは、ターゲットまたは子ノードに複数の親がある場合にのみペインに表示されます。すべてのノードは、少なくとも 1 つの他のノードに常にリンクされている必要があるため、古いリンクを削除する前に新しいリンクを作成する必要があります。

- 次のいずれかの方法を使用して、ワークフロー図のビューを編集します。
  - 設定アイコンをクリックして、ビューのズーム、パン、または位置変更を行います。
  - ワークフロー図をドラッグして画面上の位置を変更するか、マウスのスクロールを使用してズームします。

23.8. ワークフロージョブテンプレートの起動

手順

- 次のいずれかの方法を使用して、ワークフロージョブテンプレートを起動します。
  - ナビゲーションパネルで、ジョブテンプレートの横にある **Templates → Launch** を選択します。



  - 起動するワークフロージョブテンプレートの **Details** タブで **Launch** をクリックします。

ワークフロージョブテンプレートに追加された変数は、ワークフロージョブテンプレートおよび Survey に設定された追加の変数とともに、起動時に Automation controller に自動的に追加されます。

ワークフローの承認に関連するイベントは、承認リクエストがある場合は、その詳細情報とともにアクティビティーストリーム ( ) に表示されます。

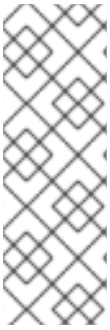
23.9. ワークフロージョブテンプレートのコピー

Automation controller を使用すると、ワークフロージョブテンプレートをコピーできます。ワークフロージョブテンプレートをコピーしても、関連するスケジュール、通知、またはパーミッションはコピーされません。スケジュールと通知は、ワークフローテンプレートのコピーを作成するユーザーまたはシステム管理者によって再作成される必要があります。ワークフローテンプレートをコピーするユーザーには管理者権限が付与されますが、ワークフローテンプレートには権限が割り当てられません (コピーされません)。

手順

1. 次のいずれかの方法を使用して、コピーするワークフロージョブテンプレートを開きます。
   - ナビゲーションパネルから、**Resources → Templates** を選択します。
   - ワークフロージョブテンプレートの **Details** ビューで、下にスクロールしてテンプレートのリストからアクセスします。
     - コピー ( ) アイコンをクリックします。
       新規テンプレートが、コピーしたテンプレートの名前とタイムスタンプが設定された状態で開きます。

2. コピーされたテンプレートを選択して **Name** フィールドの内容を新規の名前に置き換え、他のフィールドのエントリーを指定または変更してこのページを完了します。
3. **Save** をクリックします。



注記

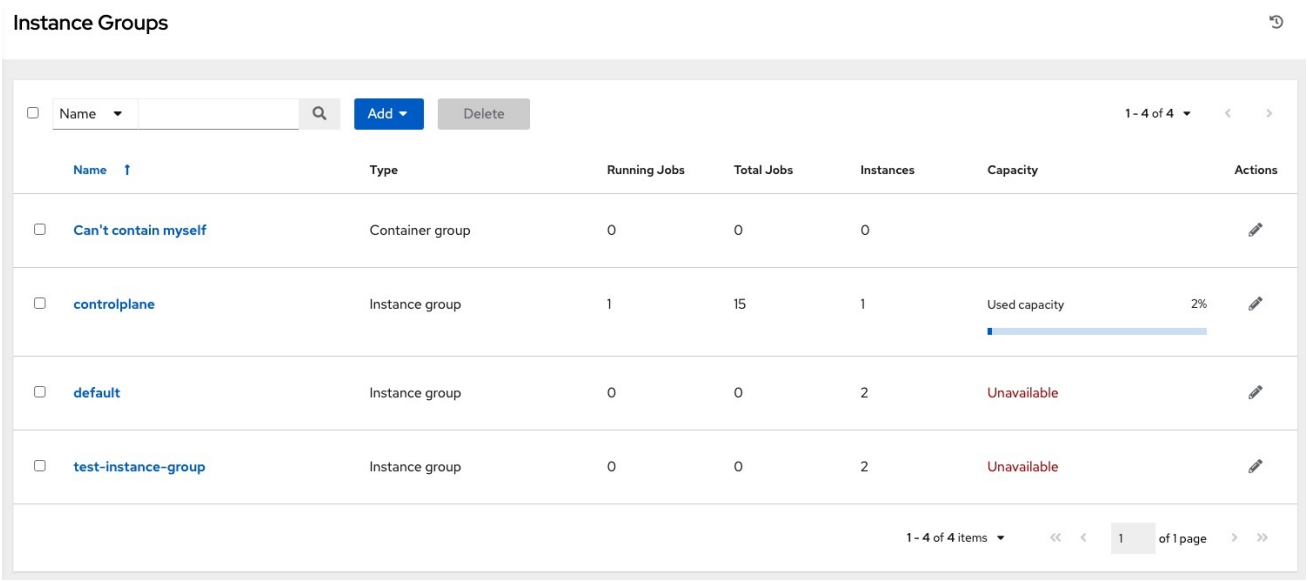
リソースに適切なレベルの権限が割り当てられていない関連リソースがある場合、そのリソースはコピーできません。たとえば、プロジェクトで、読み取り権限しか持たない現在のユーザーの認証情報を使用する場合などです。ただし、ワークフロージョブテンプレートの場合、そのノードのいずれかが未承認のジョブテンプレート、インベントリー、または認証情報を使用している場合でも、ワークフローテンプレートはコピーできます。しかし、コピーされたワークフロージョブテンプレートには、ワークフローテンプレートノードの対応するフィールドがありません。

23.10. ワークフロージョブテンプレートの追加変数

詳細は、追加変数 セクションを参照してください。

第24章 インスタンスグループの管理

インスタンスグループを使用すると、クラスター環境でインスタンスをグループ化できます。ポリシーは、インスタンスグループがどのように動作し、ジョブがどのように実行されるかを規定します。次のビューには、ポリシーアルゴリズムに基づいた容量レベルが表示されます。



関連情報

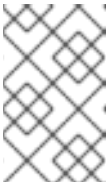
- インスタンスグループに関連付けられたポリシーまたはルールの詳細は、**Automation controller 管理ガイド** の インスタンスグループ セクションを参照してください。
- インスタンスグループをコンテナーに接続する方法の詳細は、コンテナーグループを 参照してください。

24.1. インスタンスグループの作成

新しいインスタンスグループを作成するには、次の手順を実行します。

手順

1. ナビゲーションパネルから **Administration → Instance Groups** を選択します。
2. **Add → Add instance group** の順にクリックします。
3. 以下のフィールドに該当する詳細を入力します。
   - **Name**: 名前は一意でなければならず、"controller" という名前に指定しないようにしてください。
   - **Policy instance minimum** 新規インスタンスがオンラインになると、このグループに自動的に最小限割り当てられるインスタンス数を入力します。
   - ***Policy instance percentage** スライダーを使用して、新規インスタンスがオンラインになると、このグループに自動的に最小限割り当てられるインスタンスの割合を選択します。



注記

新しいインスタンスグループを作成する場合、ポリシーインスタンスフィールドは必要ありません。値を指定しない場合、**Policy instance minimum** と **Policy instance percentage** は、デフォルトで 0 に設定されます。

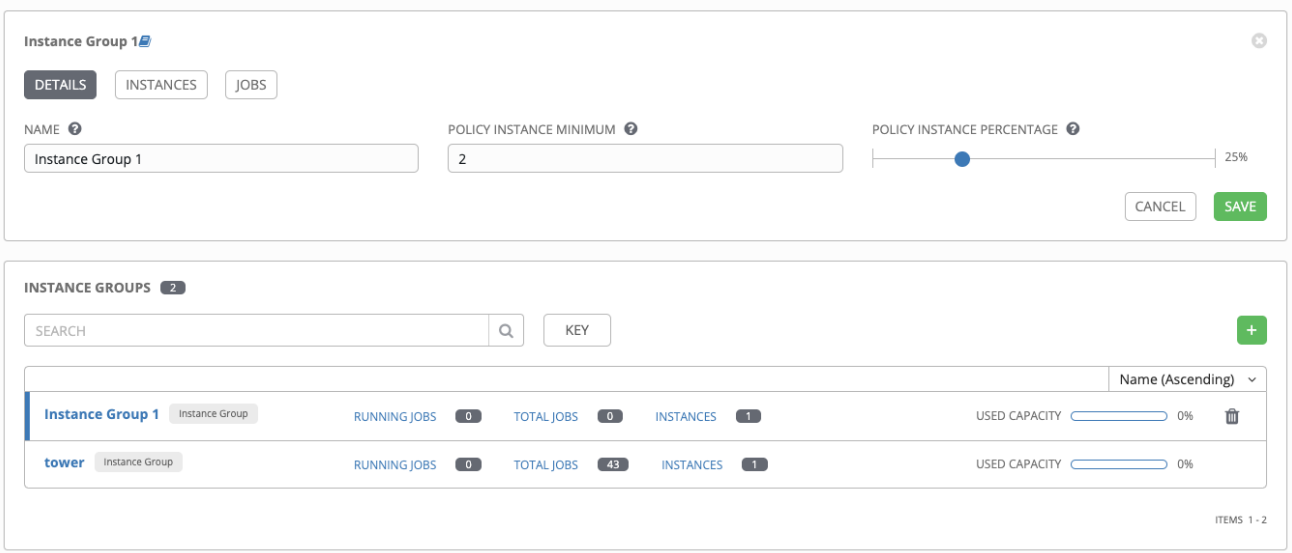
- **Max concurrent jobs**: 特定のジョブに対して実行できるフォークの最大数を指定します。
- **Max forks**: 特定のジョブに対して実行できる同時ジョブの最大数を指定します。

注記

Max concurrent jobs と **Max forks** のデフォルト値 0 は、制限がないことを示します。詳細は、**Automation controller 管理ガイド** の インスタンスグループの容量制限 を参照してください。

4. **Save** をクリックします。

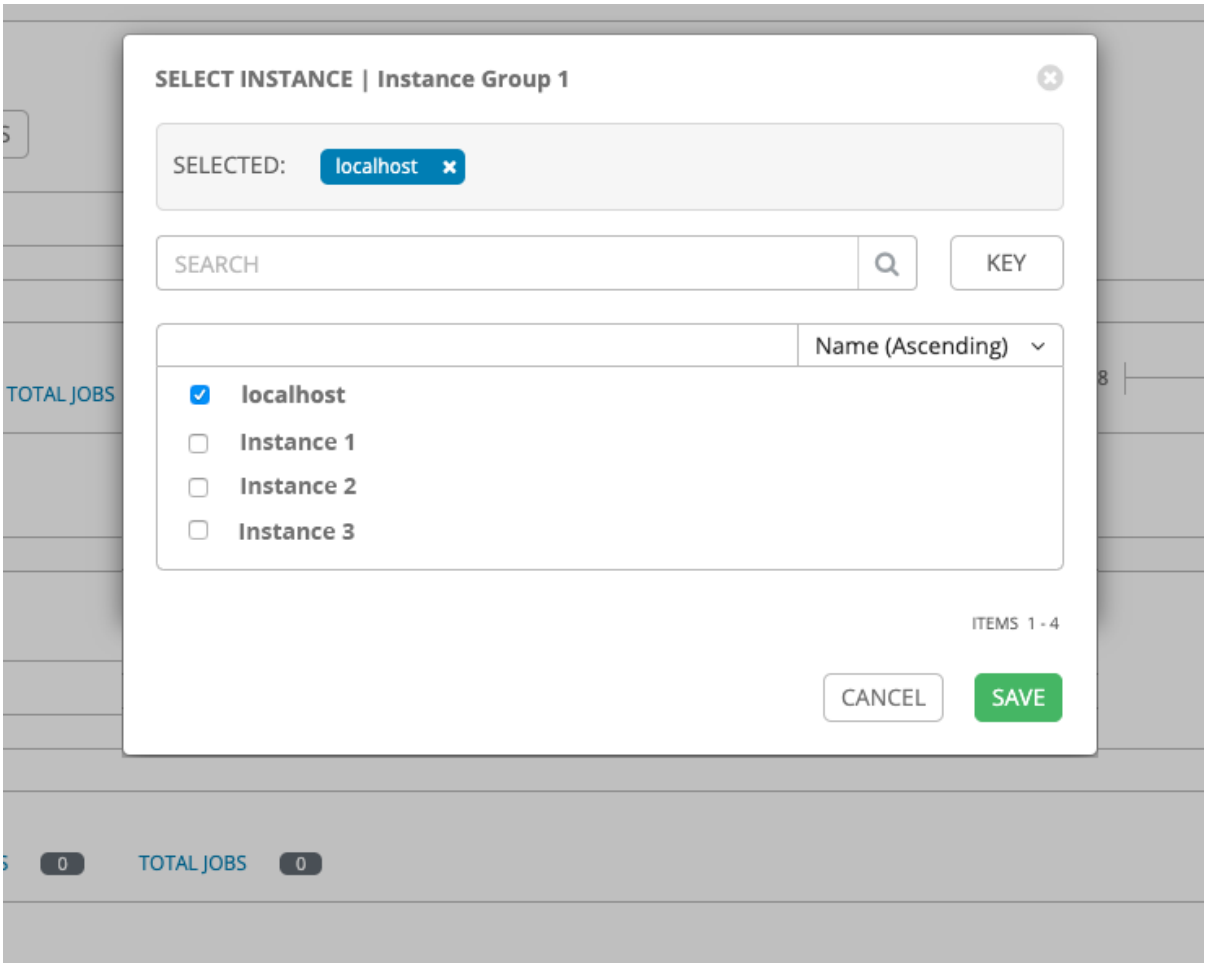
インスタンスグループが正常に作成されると、新しく作成されたインスタンスグループの **Details** タブが残り、インスタンスグループ情報を確認および編集できるようになります。これは、**Instance Groups** ビューから編集 ( ) アイコンをクリックしたときに開く画面と同じです。**Instances** を編集し、このインスタンスグループに関連付けられた **Jobs** を確認することもできます。



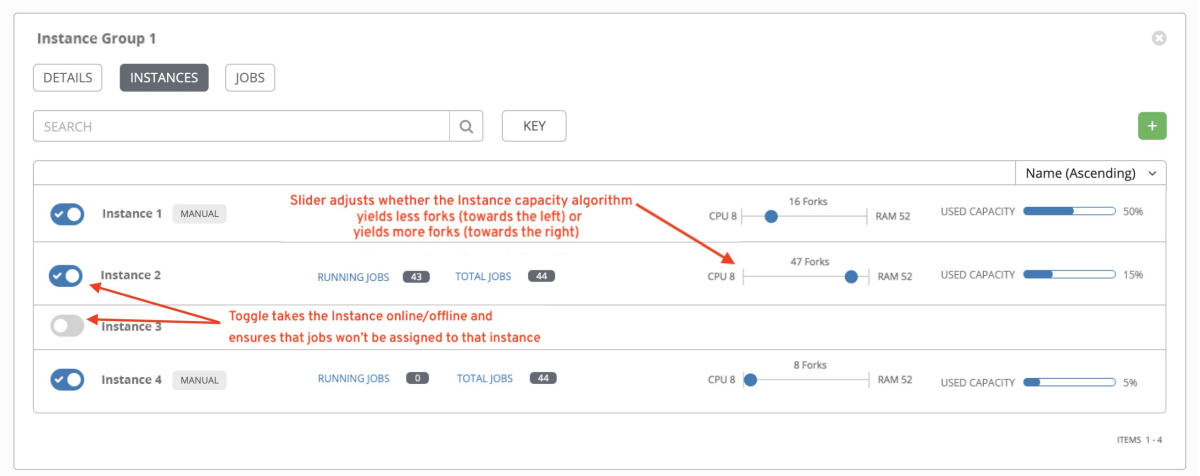
24.1.1. インスタンスグループへのインスタンスの関連付け

手順

1. **Instance Groups** ウィンドウの **Instance** タブを選択します。
2. **Associate** をクリックします。
3. リストから 1 つ以上の使用可能なインスタンスの横にあるチェックボックスをクリックして、インスタンスグループに関連付けるインスタンスを選択します。



4. 以下の例では、インスタンスグループに追加するインスタンスが、その容量に関する情報と合わせて表示されます。



24.1.2. インスタンスグループに関連付けられたジョブの表示

手順

1. **Instance Group** 画面の **Jobs** タブをクリックします。
2. ジョブの横にある矢印 > アイコンをクリックしてビューを展開し、各ジョブの詳細を表示します。
   各ジョブには次の詳細が表示されます。
   - ジョブステータス

- ID と名前
- ジョブの種類
- 開始時間および終了時間
- ジョブを開始したユーザー、および関連付けられた該当するリソース (テンプレート、インベントリー、プロジェクト、実行環境など)

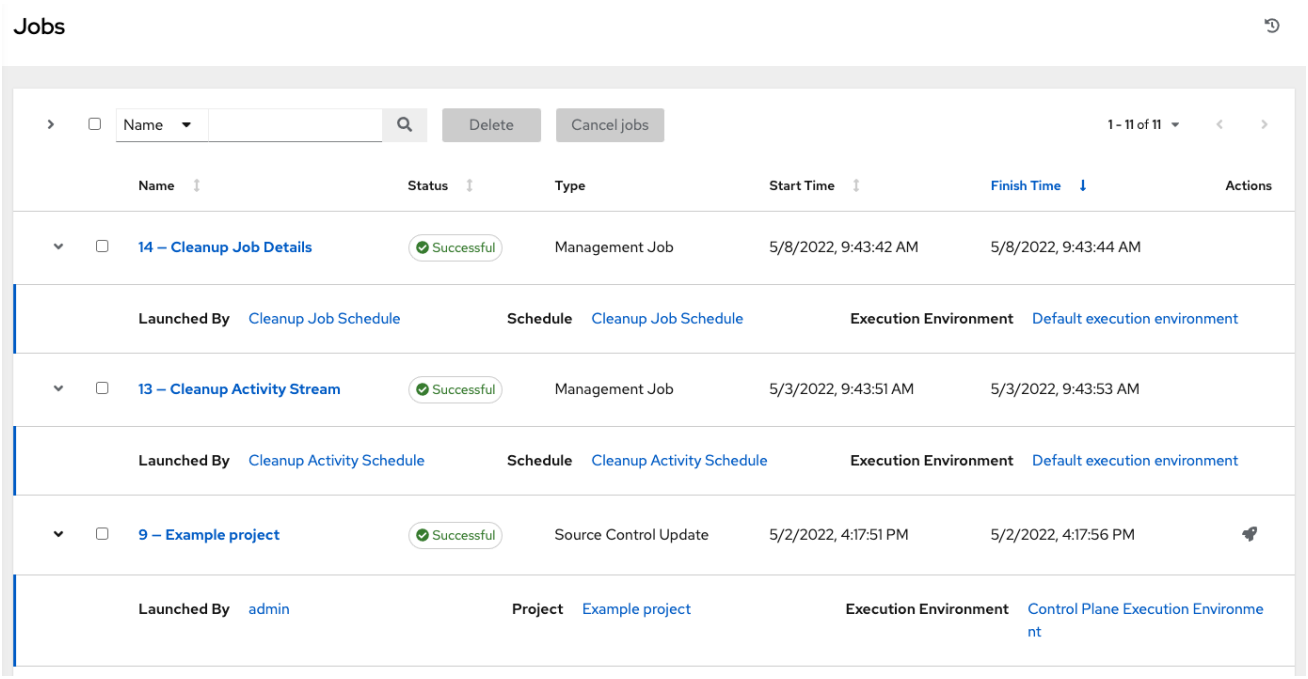
関連情報

インスタンスは、インスタンスグループポリシーに従って実行されます。詳細は、**Automation controller 管理ガイド** の インスタンスグループポリシー を参照してください。

第25章 AUTOMATION CONTROLLER のジョブ

ジョブは、Ansible Playbook をホストのインベントリーに対して起動する automation controller のインスタンスです。

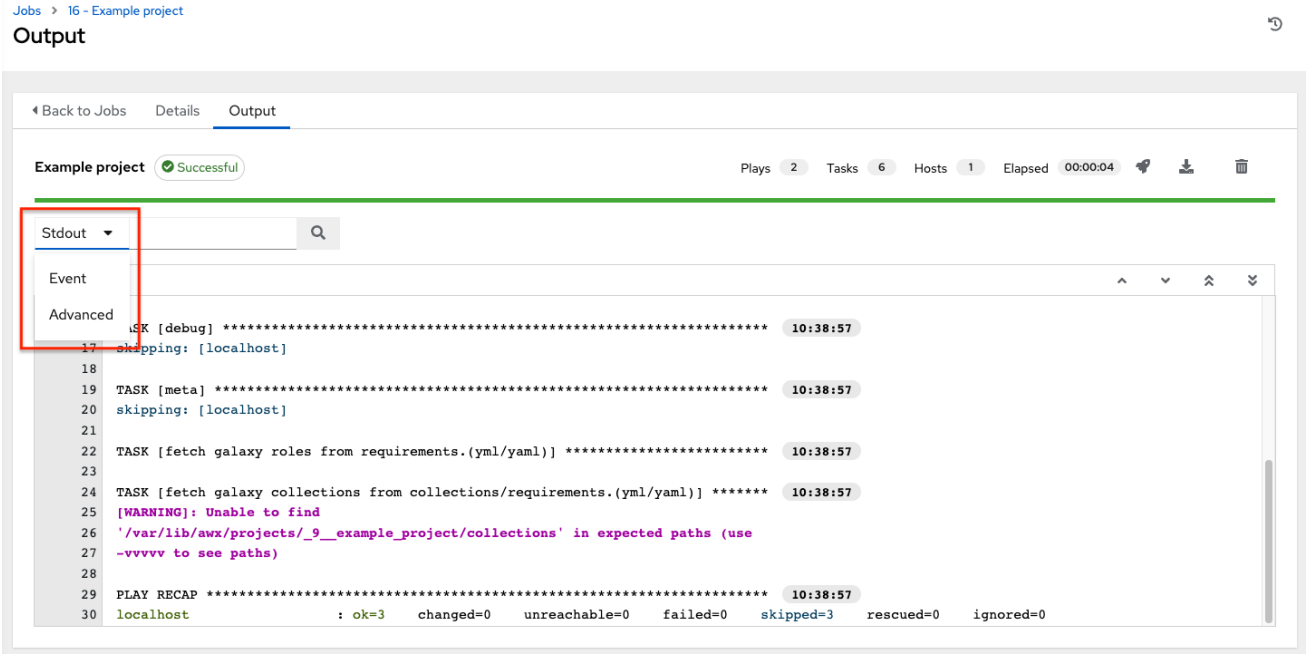
Jobs リストビューには、ジョブとそのステータスのリストが表示され、completed successfully、failed または active (running) ジョブとして表示されます。デフォルトのビューは折りたたまれて (コンパクト)、ジョブ名、ステータス、ジョブタイプ、開始時刻、終了時刻が表示されます。矢印 > アイコンをクリックして、展開し、詳細を表示します。このリストをさまざまな基準で並べ替えたり、検索を実行して目的のテンプレートをフィルタリングしたりできます。



この画面から、次のタスクを実行できます。

- 特定のジョブの詳細と標準出力を表示する
- ジョブを再起動する
- 選択したジョブを削除する

再起動操作は、Playbook の実行の再起動にのみ適用され、システムジョブ、プロジェクト/インベントリーの更新、ワークフロージョブなどには適用されません。ジョブが再起動されると、**Jobs Output** ビューが表示されます。任意のタイプのジョブを選択すると、そのジョブの **Jobs Output** ビューが表示され、さまざまな基準でジョブをフィルターできます。



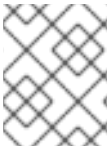
- **Stdout** オプションは、ジョブプロセスおよび出力を表示するデフォルトの表示です。
- **Event** オプションを使用すると、エラー、ホストの障害、ホストの再試行、スキップされた項目などの関心のあるイベントによってフィルタリングできます。フィルターには、必要な数のイベントを含めることができます。
- **Advanced** オプションは、条件の包含または除外、キーによる検索、またはルックアップタイプによる検索を組み合わせた詳細検索です。検索の使用方法の詳細は、検索 セクションを参照してください。

25.1. インベントリー同期ジョブ

インベントリー同期が実行されると、結果が **Output** タブに表示されます。使用すると、Ansible CLI に同じ情報が表示されます。これはデバッグに役立ちます。**ANSIBLE_DISPLAY_ARGS_TO_STDOUT** パラメーターは、すべての Playbook 実行に対して **False** に設定されます。このパラメーターは Ansible のデフォルトの動作と一致しており、特定の機密モジュールパラメーターが **stdout** に漏洩することを避けるために、**Job Detail** インターフェイスのタスクヘッダーにタスク引数が表示されません。以前の動作を復元するには、**AWX_TASK_ENV** 設定で **ANSIBLE_DISPLAY_ARGS_TO_STDOUT** を **True** に設定します。

詳細は、 ANSIBLE_DISPLAY_ARGS_TO_STDOUT を参照してください。

アイコンを使用して を再起動してジョブ出力のダウンロード 、ジョブの削除 を実行します。

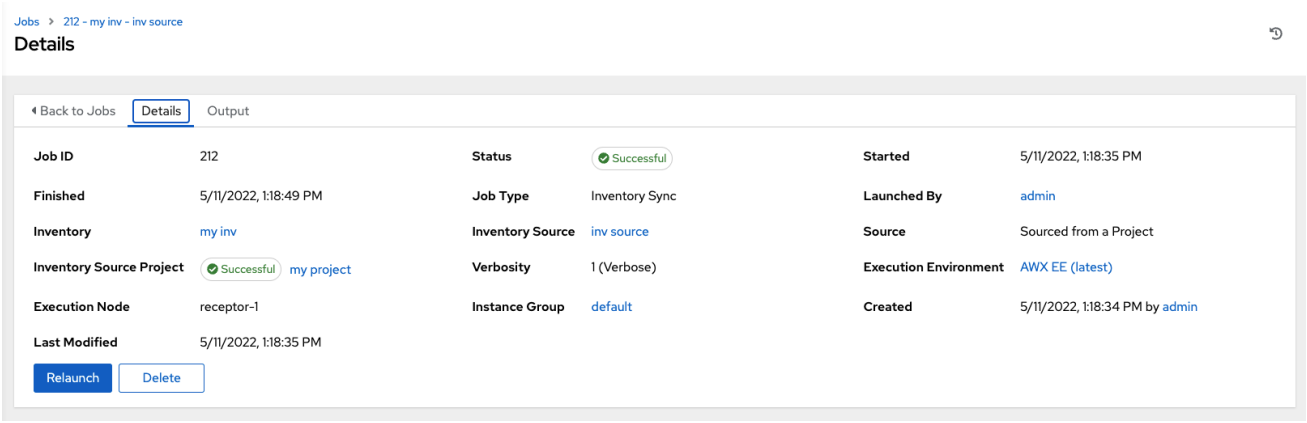


注記

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。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。

25.1.1. インベントリー同期の詳細

Details タブにアクセスして、ジョブの実行に関する詳細を表示します。



実行したジョブの次の詳細を表示できます。

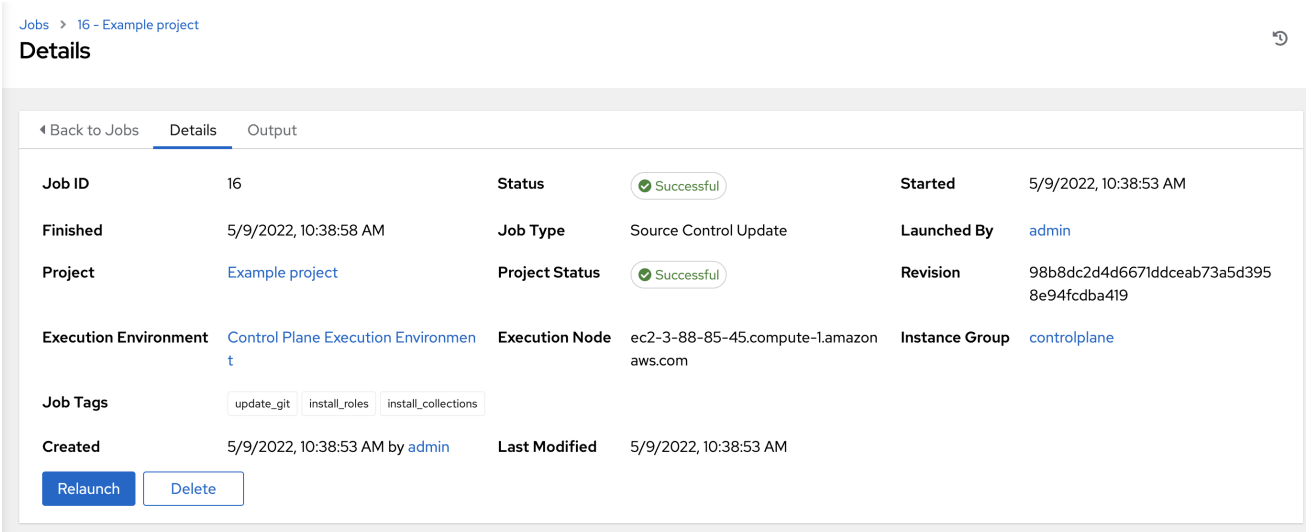
- **ステータス**: 次のいずれかになります。
  - **Pending**: インベントリー同期は作成されましたが、まだキューに登録されておらず、開始されていません。インベントリーソースの同期に限らず、すべてのジョブは、システムによる実行の準備が整うまで保留状態になります。インベントリーソース同期の準備ができていない理由には次のようなものがあります。
    - 現在実行中の依存関係 (次のステップを実行する前に、すべての依存関係が完了する必要があります)。
    - 設定された場所で実行するには容量が不十分です。
  - **Waiting**: インベントリーの同期はキューに入れられており、実行を待機中です。

    - **Running**: インベントリーの同期が進行中です。
    - **Successful**: インベントリー同期ジョブが成功しました。
    - **Failed**: インベントリーの同期ジョブが失敗しました。
- **Inventory**: 関連付けられたインベントリーグループの名前。
- **Source**: クラウドインベントリーのタイプ。
- **Inventory Source Project** このインベントリー同期ジョブのソースとして使用されるプロジェクト。
- **Execution Environment**: 使用される実行環境。
- ***Execution node**: ジョブの実行に使用されるノード。
- **Execution node**: このジョブで使用されるインスタンスグループの名前 (コントローラーはデフォルトのインスタンスグループ)。

これらの項目を選択すると、対応するジョブテンプレート、プロジェクト、およびその他のオブジェクトを表示できます。

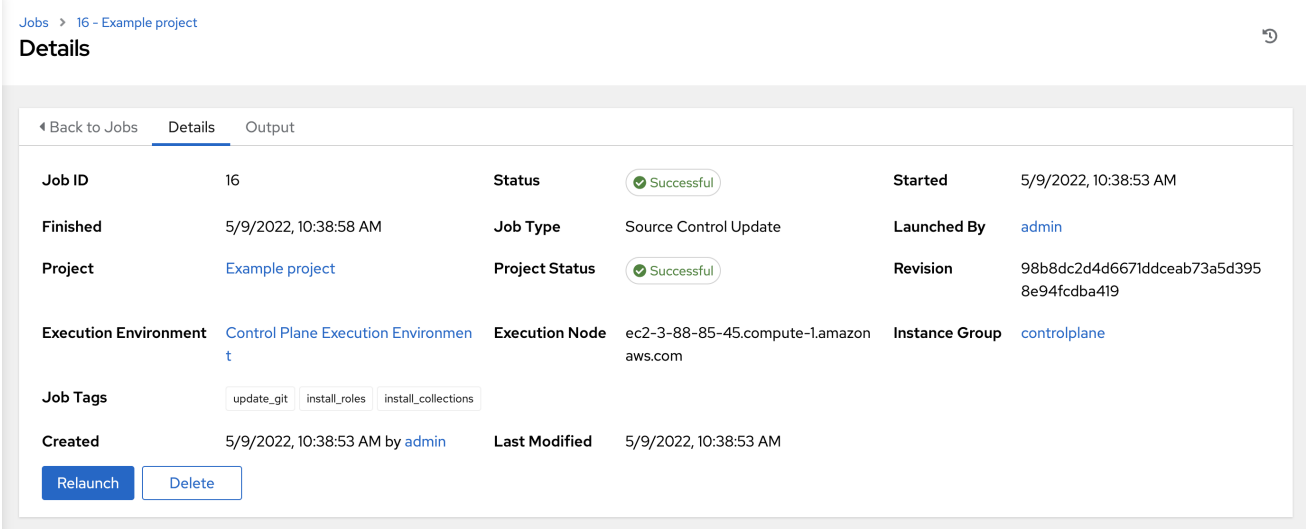
25.2. SCM インベントリージョブ

git などの SCM から取得したインベントリーが実行されると、結果が **Output** タブに表示されます。使用すると、Ansible CLI に同じ情報が表示されます。これはデバッグに役立ちます。ナビゲーションメニューのアイコンを使用して再起動 ( ) して、ジョブ出力のダウンロード ( )、またはジョブの削除 ( ) を実行します。



25.2.1. SCM インベントリーの詳細

ジョブの実行とそれに関連するプロジェクトの詳細を表示するには、**Access** タブを選択します。



実行したジョブの次の詳細を表示できます。

- **ステータス**: 次のいずれかになります。
  - **Pending**: SCM ジョブは作成されましたが、まだキューに登録されていないか、開始されていません。SCM ジョブに限らず、すべてのジョブは、システムで実行できる状態になるまで保留状態になります。SCM ジョブの準備ができていない理由としては、現在実行中の依存関係 (次のステップを実行する前にすべての依存関係が完了している必要がある) に含まれていないか、または設定されている場所で実行するのに十分な容量がないことが考えられます。
  - **Waiting**: SCM ジョブはキューに入れられており、実行を待機中です。
  - **Running**: SCM ジョブが進行中です。
  - **Running**: 直前の SCM ジョブが成功しました。
  - **Failed**: 直前の SCM ジョブが失敗しました。
- **Job Type**: SCM ジョブはソースコントロールの更新を表示します。
- **Project**: プロジェクトの名前。
- **Project Status**: 関連付けられたプロジェクトが正常に更新されたかどうかを示します。
- **Revision**: このジョブでソースとして使用されたプロジェクトのリビジョン番号を示します。
- **Execution Environment**: このジョブの実行に使用される実行環境を指定します。
- **Execution Node**: ジョブが実行されたノードを示します。
- **Instance Group**: 指定されている場合、ジョブが実行されたインスタンスグループを示します。
- **Job Tags**: タグは実行されたさまざまなジョブ操作を示します。

これらの項目を選択すると、対応するジョブテンプレート、プロジェクト、およびその他のオブジェクトを表示できます。

25.3. PLAYBOOK 実行ジョブ

Playbook が実行されると、結果が **Output** タブに表示されます。使用すると、Ansible CLI に同じ情報が表示されます。これはデバッグに役立ちます。

イベントの概要には、この Playbook の一部として実行される次のイベントが表示されます。

- この Playbook が実行された回数が **Plays** フィールドに表示されます。
- この Playbook に関連付けられたタスクの数は、**Tasks** フィールドに表示されます。
- この Playbook に関連付けられているホストの数が **Hosts** フィールドに表示されます。
- **Elapsed** フィールドで Playbook の実行を完了するのにかかった時間

イベントの横にあるアイコンを使用すると、再起動 ( )、ジョブ出力のダウンロード ( )、またはジョブの削除 ( ) を実行できます。

Output ビューのホストステータスバーのセクションにマウスをかざすと、そのステータスに関連付けられたホストの数が表示されます。

Playbook ジョブの出力には、**Jobs Templates** ページの **Jobs** タブからジョブを起動した後もアクセスできます。出力内の項目タスクをクリックして、ホストの詳細を表示します。

25.3.1. 検索

Search を使用して、特定のイベント、ホスト名、およびそれらのステータスを検索します。特定のステータスの特定のホストのみをフィルターするには、次の有効なステータスのいずれかを指定します。

ok

タスクは正常に完了しましたが、ホスト上で変更が実行されなかったことを示します。

changed

Playbook タスクが実行されました。Ansible タスクは冪等になるように作成する必要があるため、ホスト上で何も実行せずにタスクが正常に終了する場合があります。このような場合、タスクは **ok** を返しますが、**変更されません**。

failed

タスクは失敗しました。このホストに対するそれ以降の Playbook の実行は停止されました。

unreachable

ホストがネットワークから到達できないか、ホストに関連する別の致命的なエラーがあります。

skipped

ホストが目標状態に到達するために変更が必要なかったため、Playbook タスクはスキップされました。

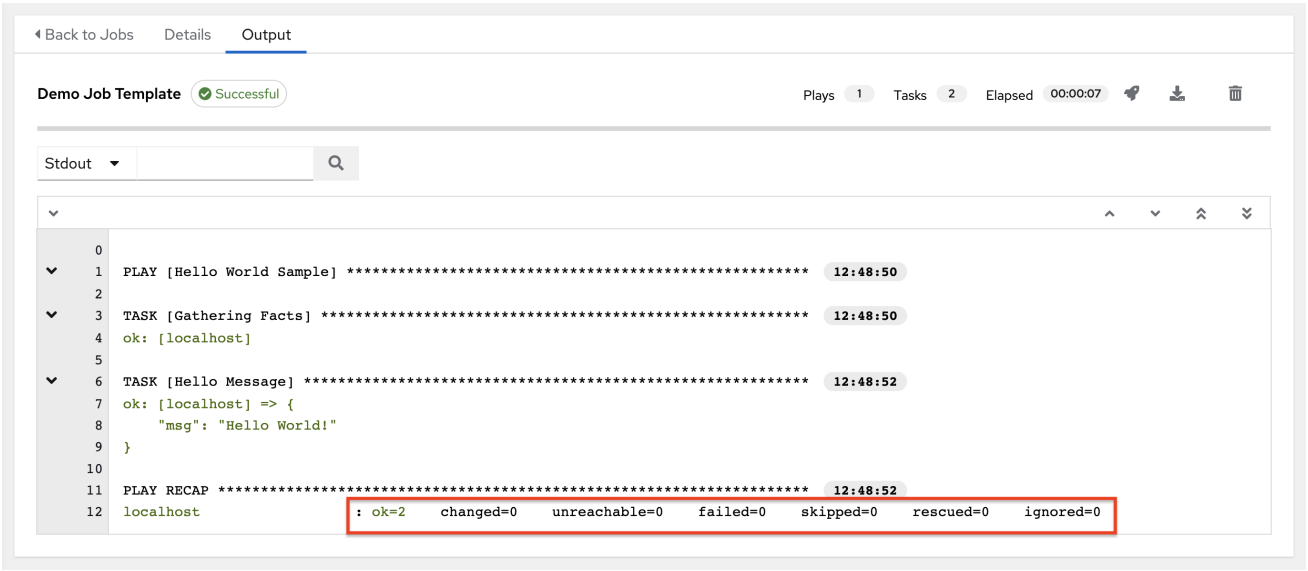
rescued

失敗したタスクが表示してから、レスキューセクションが実行されます。

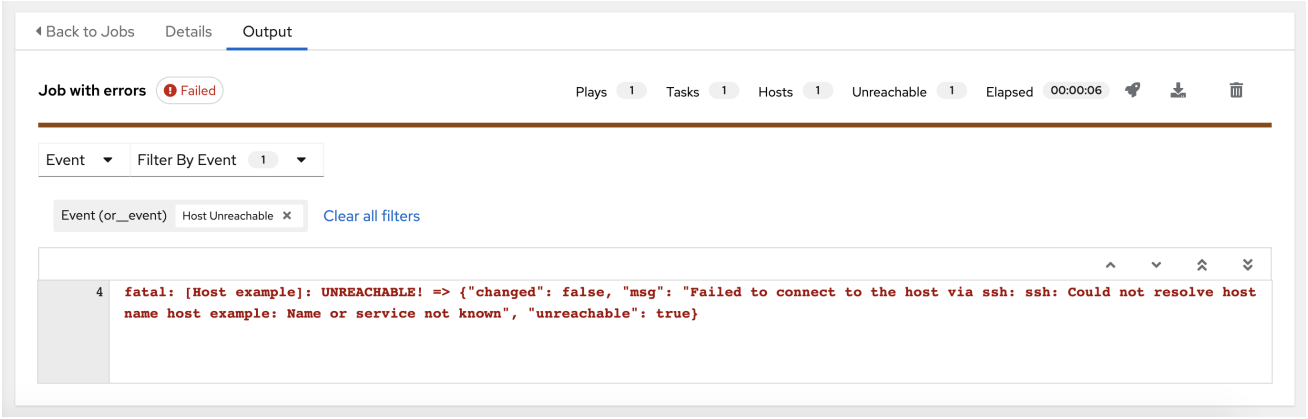
ignored

ignore_errors: yes が設定されていて、失敗したタスクを示しています。

これらのステータスは、各 **Stdout** ペインのホスト概要フィールドと呼ばれる統計のグループにも表示されます。



次の例は、到達不能なホストのみを使用した検索を示しています。

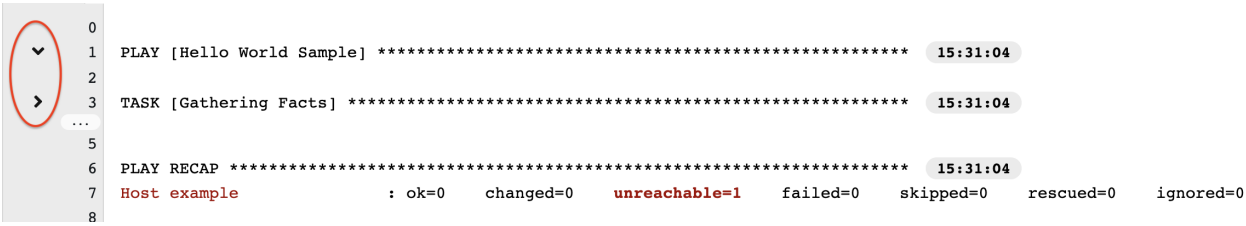


検索の使用方法の詳細は、検索 セクションを参照してください。

標準出力ビューには、特定のジョブで発生したイベントが表示されます。デフォルトでは、詳細が表示

されるようにすべての行がデプロイメントされます。すべて折りたたむ ( ) アイコンをクリックして、プレイとタスクのヘッダーのみを含むビューに切り替えます。プラス記号 ( ) アイコンをクリックすると、標準出力のすべての行が表示されます。

特定のプレイまたはタスクの横にある矢印アイコンをクリックすると、その詳細をすべて表示できます。横から下への矢印をクリックして、そのプレイまたはタスクに関連付けられた線をデプロイメントします。矢印をクリックして横の位置に戻り、線を折りたたんで非表示にします。



デプロイメントモードまたは折りたたみモードで詳細を表示する場合は、次の点に注意してください。

- 縮小表示されない行で、対応する行番号と開始時間を含む、表示されたそれぞれの行。
- プレイまたはタスクの完了後のプレイまたはタスクの開始時の展開/折りたたみアイコン。
- 特定のプレイまたはタスクのクエリーを実行する場合、それは完了したプロセスの最後に縮小表示されます。
- 場合によっては、出力が大きすぎて表示できない可能性があることを示すエラーメッセージが表示されます。これはイベントが 4000 を超える場合に発生します。エラーを回避するには、特定のイベントの検索とフィルターを使用します。

Stdout ペインでイベントの行をクリックすると、別のウィンドウに **Host Events** ウィンドウが表示されます。このウィンドウには、その特定のイベントの影響を受けたホストが表示されます。

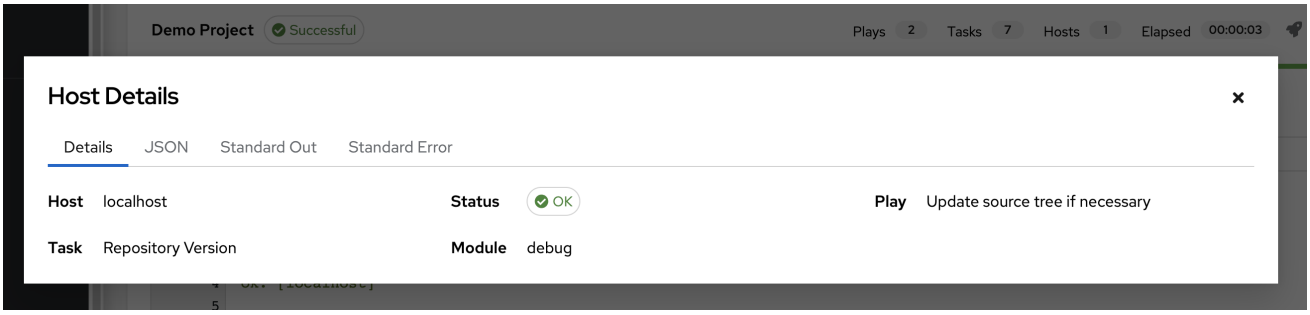
注記

Ansible Automation Platform の最新バージョンにアップグレードするには、すべての履歴 Playbook 出力とイベントを段階的に移行する必要があります。この移行プロセスは段階的に行われ、インストールの完了後にバックグラウンドで自動的に実行されます。非常に大量の履歴ジョブ出力 (数十 GB または数百 GB の出力) を含むインストールでは、移行が完了するまでジョブ出力が欠落する可能性があります。最新のデータが出力の先頭に表示され、その後に古いイベントが続きます。

25.3.2. ホストの詳細

Host Details ウィンドウには、選択したイベントの影響を受けるホストと、それに関連するプレイおよびタスクに関する次の情報が表示されます。

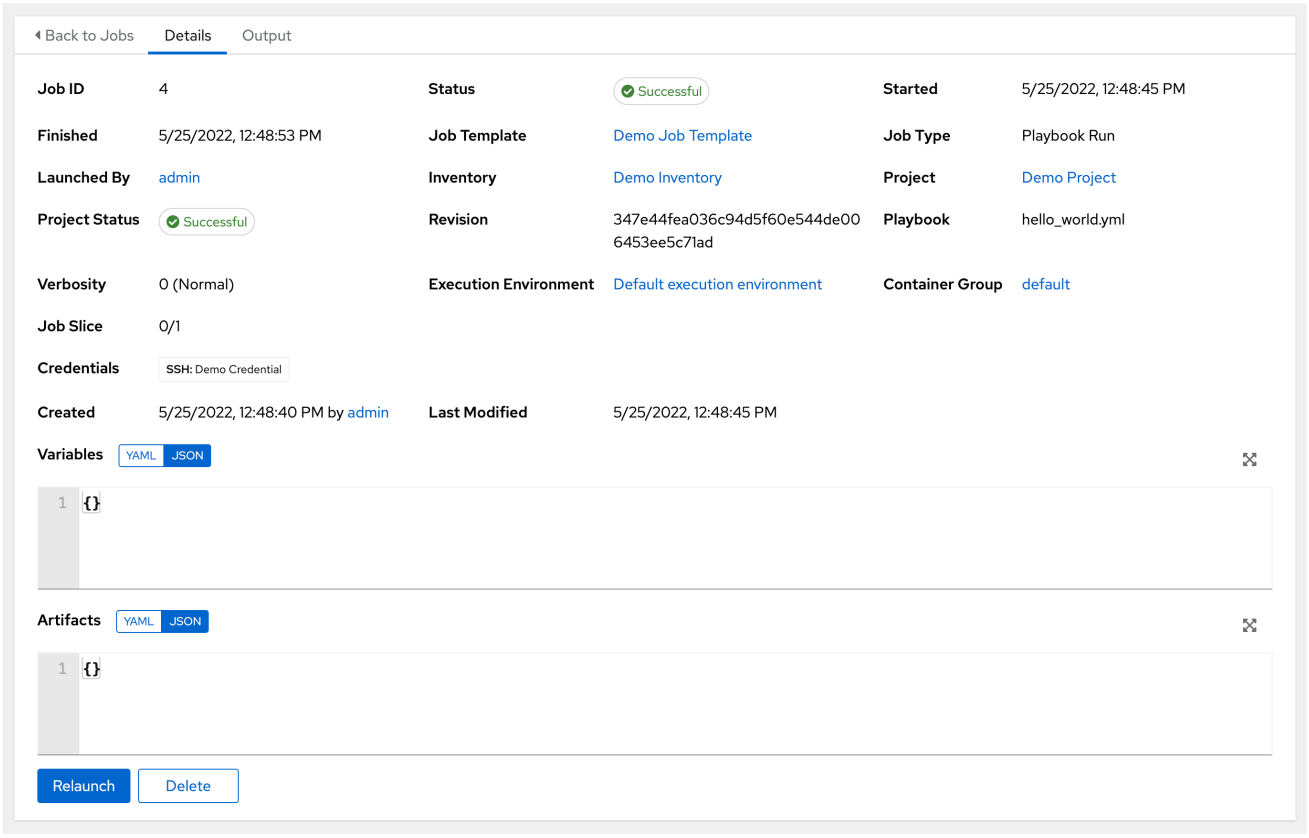
- **Host**
- **Status**
- **Play** フィールドでの実行タイプ
- **Task** のタイプ
- 該当する場合、Ansible Moduke タスク、およびモジュールのすべての引数



JSON 形式で結果を表示するには、**JSON** タブをクリックします。タスクの出力を表示するには、**Standard Out** をクリックします。出力からエラーを表示するには、**Standard Error** をクリックします。

25.3.3. Playbook 実行の出力:

Details タブにアクセスして、ジョブの実行に関する詳細を表示します。



実行したジョブの次の詳細を表示できます。

- **ステータス**: 次のいずれかになります。
  - **Pending**: Playbook の実行は作成されましたが、まだキューに入れられていないか、開始されていません。Playbook の実行だけでなく、すべてのジョブは、システムによって実行される準備ができるまで保留状態になります。Playbook の準備ができていない理由としては、現在実行中の依存関係 (次のステップを実行する前にすべての依存関係が完了している必要がある) に含まれていないか、または設定されている場所で実行するのに十分な容量がないことが考えられます。
  - **Waiting**: Playbook 実行はキューに入れられており、実行を待機中です。
  - **Running**: Playbook 実行が進行中です。

- **Successful**: 直前の Playbook 実行が成功しました。
- **Failed**: 直前の Playbook 実行が失敗しました。

- **Job Template**: ジョブが起動されるジョブテンプレートの名前。
- **Inventory**: このジョブを実行するために選択されたインベントリー。
- **Inventory**: 起動したジョブに関連付けられたプロジェクトの名前。
- **Project Status**: 起動したジョブに関連付けられたプロジェクトのステータス。
- **Playbook**: このジョブを起動するために使用される Playbook。
- **Execution Environment**: このジョブで使用される実行環境の名前。
- **Container Group**: このジョブで使用されるコンテナーグループの名前。
- **Credentials**: このジョブで使用される認証情報。
- **Extra Variables**: ジョブテンプレートの作成時に渡される追加変数がここに表示されます。

これらの項目のいずれかを選択すると、対応するジョブテンプレート、プロジェクト、およびその他のオブジェクトが表示されます。

25.4. AUTOMATION CONTROLLER の容量決定とジョブへの影響

Automation controller 容量システムは、インスタンスで利用可能なリソース量や、実行中のジョブのサイズ (*影響* と呼ぶ) をもとに、インスタンスでいくつのジョブを実行可能か判断します。これを判断するのに使用するアルゴリズムは、2 つのアイテムだけをベースとします。

- システムで利用可能なメモリー容量 (**mem_capacity**)
- システムで利用可能な処理容量 (**cpu_capacity**)

容量はインスタンスグループにも影響します。グループはインスタンスで設定されるため、インスタンスを複数のグループに割り当てることもできます。これは、1 つのインスタンスへの影響が他のグループの全体的な容量に影響を与える可能性があることを意味します。

インスタンス自体ではなく、インスタンスグループを、さまざまなレベルのジョブで使用するように割り当てることができます。詳細は、**Automation controller 管理ガイド** の クラスタリング を参照してください。

タスクマネージャーは、ジョブが実行されるグループを決定するためのグラフを準備するときに、まだ開始する準備ができていないジョブにインスタンスグループの容量をコミットします。

小規模な設定では、ジョブの実行に使用できるインスタンスが 1 つだけの場合、タスクマネージャーは、インスタンスの容量を超えた場合でも、そのジョブをインスタンス上で実行できるようにします。これにより、システムのプロビジョニング不足が原因でジョブが停止することがなくなります。

関連情報

- コンテナーグループの詳細は、 **Automation controller 管理ガイド** の コンテナーの容量制限 を参照してください。
- スライスされたジョブおよび容量への影響に関する情報は、ジョブスライスの実行動作 を参照してください。

25.4.1. 容量アルゴリズムに向けたリソースの判断

容量アルゴリズムは、システムが同時に実行できるフォークの数を決定します。これらのアルゴリズムは、Ansible が同時に通信できるシステムの数を制御します。Automation controller システムが実行するフォークの数を増やすと、より多くの作業を並行して実行できるため、ジョブをより高速に実行できます。ただし、これによりシステムの負荷が増加し、作業が遅くなる可能性があります。

デフォルトの **mem_capacity** を使用すると、システムがメモリー不足にならないようにしながら、処理リソースをオーバーコミットできます。作業のほとんどがプロセッサーに依存しない場合、このモードを選択するとフォークの数が最大になります。

25.4.1.1. メモリー対容量

mem_capacity は、フォークごとに必要なメモリー量に応じて計算されます。内部コンポーネントのオーバーヘッドを考慮すると、これはフォークあたり約 100MB になります。Ansible ジョブで使用できるメモリーの量を考慮する場合、容量アルゴリズムは、他のサービスの存在を考慮して 2GB のメモリーを予約します。このアルゴリズムの式は次のとおりです。

```
(mem - 2048) / mem_per_fork
```

以下に例を示します。

```
(4096 - 2048) / 100 == ~20
```

4GB のメモリーを搭載したシステムは、20 個のフォークを実行できます。**mem_per_fork** の値は、**SYSTEM_TASK_FORKS_MEM** の値を設定することによって制御されます。デフォルトは 100 です。

25.4.1.2. CPU 対容量

Ansible ワークロードは多くの場合、プロセッサーに依存します。このような場合、同時ワークロードを減らすことで、より多くのタスクをより速く実行できるようになり、それらのジョブの平均完了時間を短縮できます。

mem_capacity アルゴリズムがフォークごとに必要なメモリーの量を調整するのと同じように、**cpu_capacity** アルゴリズムはフォークごとに必要な処理リソースの量を調整します。このベースライン値は、コアあたり 4 つのフォークです。このアルゴリズムの式は次のとおりです。

```
cpus * fork_per_cpu
```

たとえば、4 コアシステムは次のようになります。

```
4 * 4 == 16
```

SYSTEM_TASK_FORKS_CPU の値 (デフォルトは 4) を設定することで、**fork_per_cpu** の値を制御できます。

25.4.2. ジョブが影響を与える容量

容量を選択する場合に、各ジョブタイプがどのように容量に影響を与えるかを理解することが重要です。

Ansible のデフォルトのフォーク値は 5 です。ただし、それよりも少ないシステムに対して実行するように Automation controller を設定すると、実際の同時実行値は低くなります。

ジョブが Automation controller で実行されると、Ansible の親プロセスを補うために、選択されたフォークの数が 1 つ増加します。

例

フォーク値が 5 の 5 つのシステムに対して Playbook を実行した場合、ジョブへの影響の観点から見た実際のフォーク値は 6 になります。

25.4.2.1. 自動コントローラーのジョブタイプの影響

ジョブとアドホックジョブは、前述のモデル、フォーク +1 に従います。ジョブテンプレートにフォーク値を設定した場合、ジョブの容量値は、指定されたフォーク値の最小値と、所有しているホストの数に 1 を加えた値になります。+1 は、親 Ansible プロセスを考慮するためのものです。

インスタンスの容量によって、どのジョブが特定のインスタンスに割り当てられるかが決まります。ジョブとアドホックコマンドは、フォーク値が大きいほど、より多くの容量を使用します。

以下を含むジョブタイプは、一定の影響を及ぼします。

- インベントリーの更新: 1
- プロジェクトの更新: 1
- システムジョブ: 5

注記

ジョブテンプレートにフォーク値を設定しない場合、ジョブは Ansible のデフォルトのフォーク値 5 を使用します。ただし、ジョブのホストの数が 5 つ未満の場合は、使用するホストの数が減ります。一般に、フォーク値をシステムの能力よりも高く設定すると、メモリー不足や CPU のオーバーコミットによって問題が発生する可能性があります。使用するジョブテンプレートのフォーク値はシステムに適合する必要があります。1,000 個のフォークを使用する Playbook があるが、個別にそれほど多くの容量を備えたシステムがない場合、システムのサイズが小さく、パフォーマンスまたはリソースの問題が発生するリスクがあります。

25.4.2.2. 正しい容量の選択

CPU 制限またはメモリー制限の範囲外の容量を選択する場合は、フォークの最小数または最大数のどちらかを選択することになります。前の例 では、CPU 容量では最大 16 個のフォークが許可され、メモリー容量では最大 20 個のフォークが許可されます。一部のシステムでは、これらの間の差が大きくなる可能性があるため、これら 2 つの間のバランスを取ることが必要な場合があります。

インスタンスフィールドの **Capacity_adjustment** を使用すると、考慮する量を選択できます。0.0 ~ 1.0 の値で表されます。値 1.0 に設定すると、最大値が使用されます。前の例にはメモリー容量が関係しているため、フォークの値を 20 として選択できます。値を 0.0 に設定すると、最小値が使用されます。値を 0.5 にすると 2 つのアルゴリズム間の 50/50 バランス、つまり 18 です。

```
16 + (20 - 16) * 0.5 = 18
```

手順

容量を表示または編集します。

1. **Instances Groups** リストビューから、目的のインスタンスを選択します。

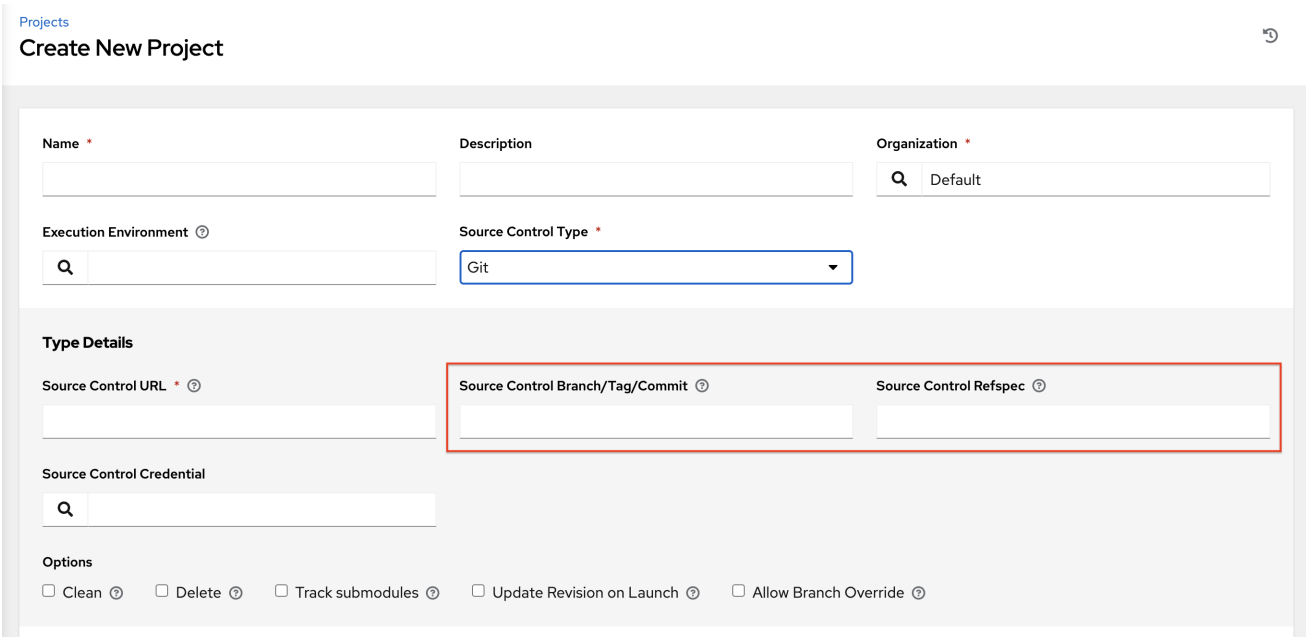
2. **Instances** タブを選択し、**Capacity Adjustment** スライダーを調整します。

注記

スライダーは、インスタンス容量アルゴリズムが生成するフォークの数が少ないか (左方向)、より多くのフォークが生成されているか (右方向) を調整します。

25.5. ジョブブランチの上書き

プロジェクトは、**scm_branch** フィールドでソースコントロールから使用するブランチ、タグ、または参照を指定します。これらは、**Type Details** フィールドで指定された値によって表されます。



ジョブの作成または編集時には、**Allow Branch Override** オプションがあります。このオプションをオンにすると、プロジェクト管理者は、そのプロジェクトを使用するジョブテンプレートにブランチの選択を委任でき、**use_role** プロジェクトのみが必要です。

25.5.1. ソースツリーのコピー動作

実行されるすべてのジョブには独自のプライベートデータディレクトリーがあります。このディレクトリーには、ジョブが実行されている特定の **scm_branch** のプロジェクトソースツリーのコピーが含まれています。ジョブはプロジェクトフォルダーに自由に変更を加え、実行中にその変更を利用できます。このフォルダーは一時的なフォルダーであり、ジョブの実行の終了時に削除されます。

Clean オプションをオンにすると、変更されたファイルは Automation controller のリポジトリーのローカルコピーから削除されます。対応の Ansible モジュールで、git または Subversion に関連する Force パラメーターを使用することで削除します。

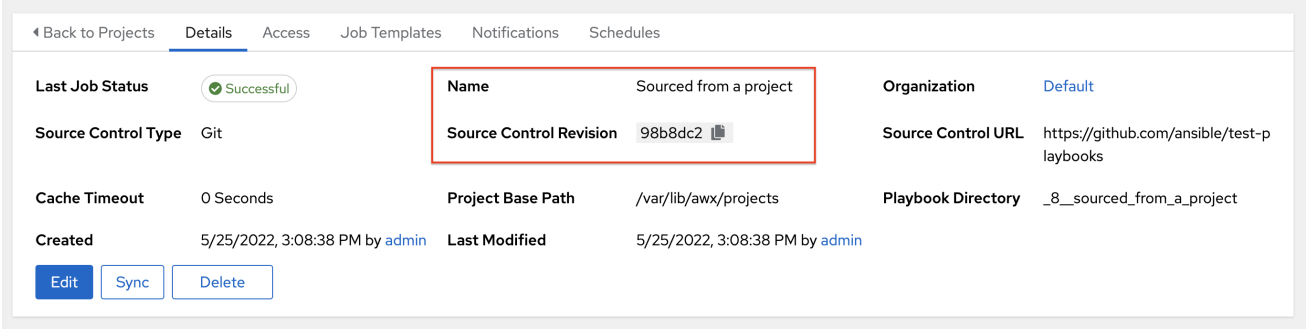
関連情報

詳細は、Ansible ドキュメントの パラメーター セクションを参照してください。

25.5.2. プロジェクトのリビジョンの動作

プロジェクトの更新中、デフォルトブランチ (プロジェクトの **SCM Branch** フィールドで指定) のリビジョンが更新時に保存されます。ジョブにデフォルト以外の **SCM Branch** (コミットハッシュやタグではない) を指定すると、ジョブが開始される直前に、最新のリビジョンがソースコントロールリモート

から取得されます。このリビジョンは、ジョブとそのプロジェクトの更新の **Source Control Revision** フィールドに表示されます。



その結果、デフォルト以外のブランチではオフラインジョブを実行できません。ジョブがソースコントロールからの静的バージョンを実行していることを確認するには、タグまたはコミットハッシュを使用します。プロジェクトの更新ではすべてのブランチが保存されるのではなく、プロジェクトのデフォルトブランチのみが保存されます。

SCM Branch フィールドは検証されていないため、プロジェクトを更新して有効であることを確認する必要があります。このフィールドが指定されたり、要求されたりした場合には、ジョブテンプレートの **Playbook** フィールドは検証されず、 ジョブテンプレートを起動して、必要とされる Playbook が存在するかを確認する必要があります。

25.5.3. Git Refspec

SCM Refspec フィールドは、更新がリモートからダウンロードする必要がある追加の参照を指定します。以下に例を示します。例としては次のようなものがあります。

- **refs/:refs/remotes/origin/**: これは、リモートのリモートを含むすべての参照を取得します。
- **refs/pull/:refs/remotes/origin/pull/** (GitHub 固有): これは、すべてのプルリクエストのすべての ref を取得します。
- **refs/pull/62/head:refs/remotes/origin/pull/62/head**: これは、1 つの GitHub プルリクエストの参照を取得します。

プロジェクトの規模が大きく、先ほどの 1 例目と 2 例目を使用する場合には、パフォーマンスへの影響を考慮する必要があります。

SCM Refspec パラメーターはプロジェクトブランチの可用性に影響を与え、他の方法では利用できない参照へのアクセスを可能にすることができます。前の例では、**SCM Branch** からプルリクエストを提供できますが、これは **SCM Refspec** フィールドがなければ不可能です。

Ansible git モジュールは、デフォルトで **refs/heads/** をフェッチします。これは、**SCM Refspec** が空白の場合、プロジェクトのブランチ、タグ、コミットハッシュを **SCM ブランチ** として使用できます。**SCM Refspec** フィールドに指定された値により、どの **SCM Branch** フィールドをオーバーライドとして使用できるかが左右されます。プロジェクトの更新 (タイプを問わず) は、追加の **git fetch** コマンドを実行して、その refspec をリモートからプルします。

例

最初または 2 番目の refspec サンプルを使用して、ブランチオーバーライドを有効にするプロジェクトを設定できます。これを、**SCM ブランチ** を求めるジョブテンプレートで使用します。その後、クライアントは新しいプルリクエストの作成時にジョブテンプレートを起動でき、**pull/N/head** ブランチを提供して、ジョブテンプレートは、提供された GitHub プルリクエスト参照に対して実行できます。

関連情報

詳細は、Ansible git モジュール を参照してください。

第26章 WEBHOOK の使用

Webhook を使用すると、Web 上のアプリケーション間で指定されたコマンドを実行できます。automation controller は現在、GitHub および GitLab との Webhook インテグレーションを提供しています。

次のサービスを使用して Webhook を設定します。

- GitHub Webhook の設定
- GitLab Webhook の設定
- ペイロード出力の表示

GitHub および GitLab の Webhook ポストステータスバック機能は、特定の CI イベント下でのみ動作するように設計されています。別の種類のイベントを受信すると、サービスログに次のようなメッセージが記録されます。

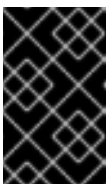
awx.main.models.mixins Webhook event did not have a status API endpoint associated, skipping.

26.1. GITHUB WEBHOOK の設定

automation controller には、トリガーされた Webhook イベントに基づいてジョブを実行する機能があります。ジョブのステータス情報 (pending, error, success) は、プルリクエストイベントの場合にのみ送信できます。Automation controller がジョブステータスを Webhook サービスにポストする必要がない場合は、直接ステップ 3 に進みます。

手順

1. Automation Controller で使用する **Personal Access Token** (PAT) を生成します。
   a. GitHub アカウントのプロファイル設定で、**Settings** を選択します。
   b. ナビゲーションパネルから、**<> Developer Settings** を選択します。
   c. **Developer Settings** ページで、**Personal access tokens** を選択します。
   d. **Personal access tokens** 画面で、**Generate a personal access token** をクリックします。
   e. プロンプトが表示されたら、GitHub アカウントのパスワードを入力して続行します。
   f. **Note** フィールドに、この PAT の用途に関する簡単な説明を入力します。
   g. **Select scopes** フィールドで、**repo:status**、**repo_deployment**、および **public_repo** の横のボックスをオンにします。自動 Webhook には、招待の場合を除き、リポジトリースコープへのアクセス権だけが必要です。詳細は、OAuth アプリのスコープのドキュメント を参照してください。
   h. **Generate Token** をクリックします。



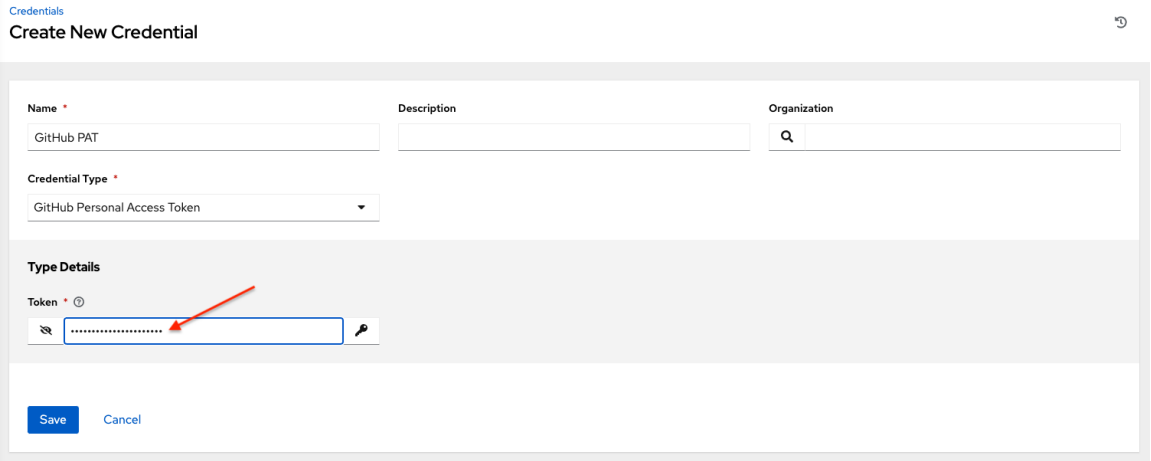
重要

トークンが生成されたら、ステップ 2 で必要となるため、必ず PAT をコピーしてください。GitHub でこのトークンに再度アクセスすることはできません。

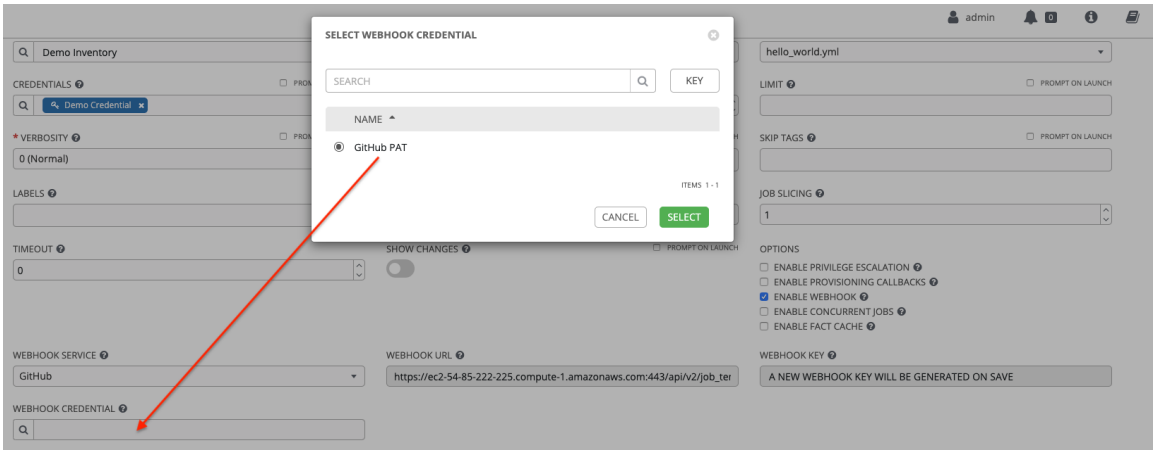
2. 必要に応じて、PAT を使用して、GitHub の認証情報を作成します。

   a. インスタンスに移動し、生成されたトークンを使用して GitHub PAT の新しい認証情報を作成 します。

   b. GitHub にポストバックするジョブテンプレートで使用するため、この認証情報の名前をメモしておきます。



   c. Webhook を有効にするジョブテンプレートに移動し、前の手順で作成した Webhook サービスと認証情報を選択します。



   d. **Save** をクリックします。ジョブテンプレートは GitHub にポストバックするように設定されています。

3. Webhook を設定する GitHub リポジトリーに移動し、**Settings** を選択します。

4. ナビゲーションパネルから、**Webhook → Add webhook** を選択します。

5. **Add webhook** ページを完了するには、ジョブテンプレートまたはワークフロージョブテンプレートで **Enable Webhook** オプションをオンにする必要があります。詳細は、 ジョブテンプレートの作成 と ワークフローテンプレートの作成 の両方のステップ 3 を参照してください。

6. 以下のフィールドに入力します。

   - **Payload URL**: **Webhook URL** の内容をジョブテンプレートからコピーし、ここに貼り付けます。結果は GitHub からこのアドレスに送信されます。
   - **Content type**: **application/json** に設定します。
   - **Secret**: ジョブテンプレートから **Webhook キー** の内容をコピーし、ここに貼り付けます。

- **Which events would you like to trigger this webhook?** Webhook をトリガーするイベントのタイプを選択します。このようなイベントがあれば、ジョブまたはワークフローがトリガーされます。ジョブのステータス (保留中、エラー、成功) を GitHub に送り返すには、**Let me select individual events** セクションで **Pull requests** を選択する必要があります。

- **Active**: これをチェックしたままにします。

7. **Add webhook** をクリックします。
8. Webhook が設定されると、編集または削除する機能とともに、リポジトリーでアクティブな Webhook のリストに表示されます。Webhook をクリックして、**Manage webhook** 画面に移動します。
9. スクロールして、Webhook に対して行われた配信試行と、それらが成功したか失敗したかを表示します。

関連情報

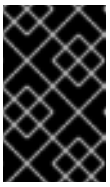
詳細は、Webhook のドキュメント を参照してください。

26.2. GITLAB WEBHOOK の設定

automation controller には、トリガーされた Webhook イベントに基づいてジョブを実行する機能があります。ジョブのステータス情報 (pending, error, success) は、プルリクエストイベントの場合にのみ送信できます。Automation controller がジョブステータスを Webhook サービスにポストする必要がない場合は、直接手順 3 に進みます。

手順

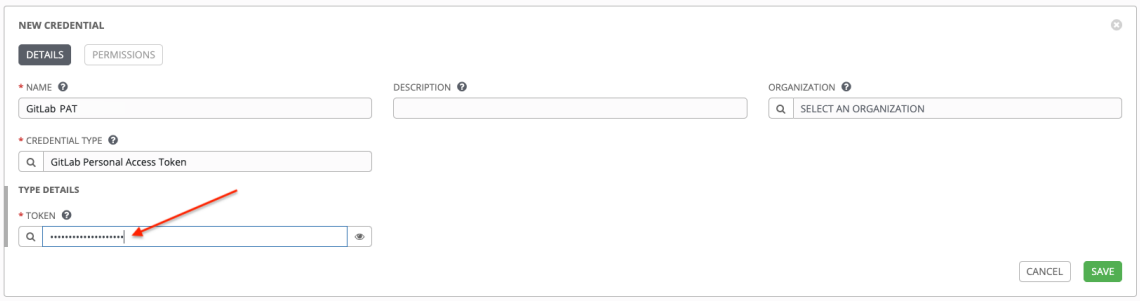
1. Automation Controller で使用する **Personal Access Token** (PAT) を生成します。
   a. GitLab のナビゲーションパネルから、アバターを選択して、**プロファイルを編集** します。
   b. ナビゲーションパネルから **Access tokens** を選択します。
   c. 以下のフィールドに入力します。
      - **Token name**: この PAT が何に使用されるかについての簡単な説明を入力します。
      - **Expiration date**: Webhook の有効期限を設定しない場合は、このフィールドをスキップします。
      - **Select scopes**: 統合に適用できるスコープを選択します。Automation controller の場合、必要な選択は **API** のみです。
   d. **Create personal access token** をクリックします。



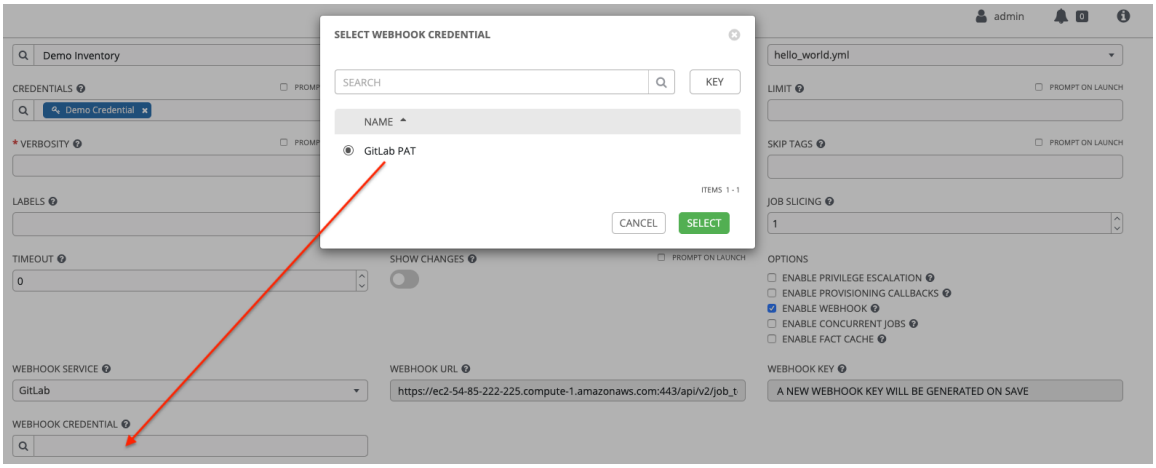
重要

トークンが生成されたら、ステップ 2 で必要となるため、必ず PAT をコピーしてください。GitLab ではこのトークンに再度アクセスすることはできません。

2. 必要に応じて、PAT を使用して、GitLab の認証情報を作成します。
   a. インスタンスに移動し、生成されたトークンを使用して GitLab PAT の新しい認証情報を作成 します。
   b. GitLab にポストバックするジョブテンプレートで使用するため、この認証情報の名前をメモしておきます。



   c. Webhook を有効にするジョブテンプレートに移動し、前の手順で作成した Webhook サービスと認証情報を選択します。



d. **Save** をクリックします。ジョブテンプレートは GitLab にポストバックするように設定されています。

3. Webhook を設定する GitLab リポジトリーに移動します。
4. ナビゲーションパネルから、**Settings** → **Integrations** を選択します。
5. **Add webhook** ページを完了するには、ジョブテンプレートまたはワークフロージョブテンプレートで **Enable Webhook** オプションをオンにする必要があります。詳細は、 ジョブテンプレートの作成 と ワークフローテンプレートの作成 の両方のステップ 3 を参照してください。
6. 以下のフィールドに入力します。
   - **URL**: **Webhook URL** の内容をジョブテンプレートからコピーし、ここに貼り付けます。結果は GitLab からこのアドレスに送信されます。
   - **Secret Token**: ジョブテンプレートから **Webhook Key** の内容をコピーし、ここに貼り付けます。
   - **Trigger**: Webhook をトリガーするイベントのタイプを選択します。このようなイベントがあれば、ジョブまたはワークフローがトリガーされます。ジョブのステータス (pending, error, success) を GitLab に送り返すには、**Trigger** セクションで **Merge request events** を選択する必要があります。
   - **SSL verification**: **Enable SSL verification** を選択したままにします。
7. **Add webhook** をクリックします。
8. Webhook が設定されると、リポジトリーの **Project Webhooks** リストに表示され、イベントのテスト、Webhook の編集または削除を行うことができます。Webhook イベントをテストすると、成功したか失敗したかの結果が各ページに表示されます。

関連情報

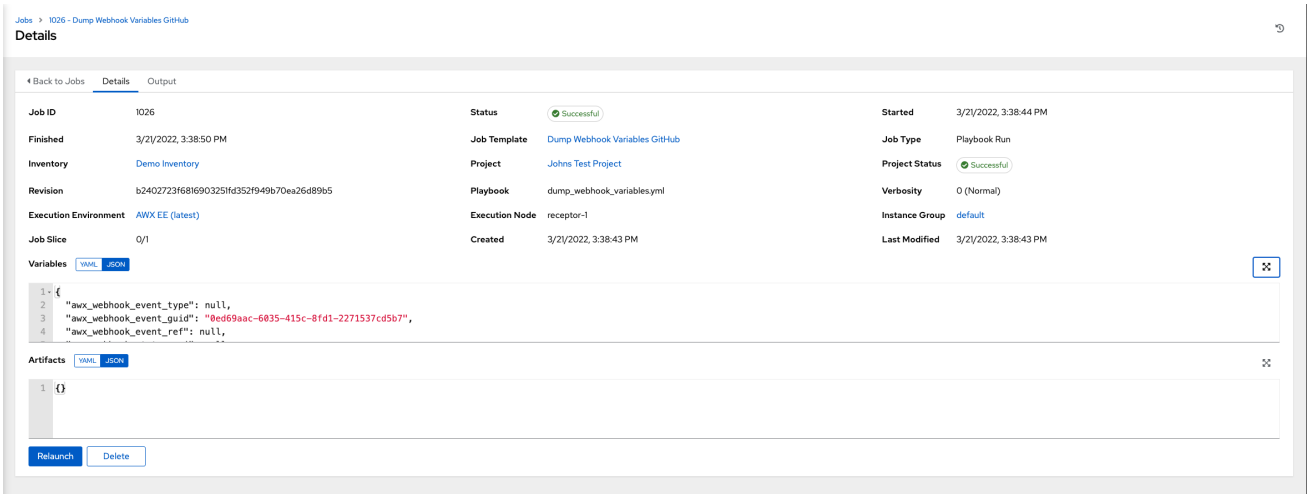
詳細は、Webhook を参照してください。

26.3. ペイロード出力の表示

追加の変数として公開されたペイロード全体を表示できます。

手順

1. ナビゲーションパネルから、**Views → Jobs** を選択します。
2. Webhook が有効になっているジョブテンプレートを選択します。
3. **Details** タブを選択します。
4. 次の例に示すように、**Extra Variables** フィールドで、**awx_webhook_payload** 変数からのペイロード出力を表示します。



第27章 通知

メール、Slack、Webhook などの 通知タイプ は、通知テンプレートのインスタンスであり、通知テンプレートで定義された名前、説明、設定を持ちます。

以下に、通知テンプレートの追加に必要な詳細の例を示します。

- メールの通知テンプレートには、ユーザー名、パスワード、サーバー、および受信者が必要です。
- Slack 通知テンプレートには、トークンとチャンネルのリストが必要です。
- Webhook 通知テンプレートには、URL とヘッダーが必要です。

ジョブが失敗すると、通知テンプレートで定義した設定を使用して通知が送信されます。

以下に、通知システムの一般的なフローを示します。

- API または UI を介して、**/api/v2/notification_templates エンドポイント** で **REST API** への通知テンプレートを作成します。
- 通知テンプレートは、サポートするさまざまなオブジェクト (ジョブテンプレートのすべてのバリアント、組織およびプロジェクト) のいずれかに対して、通知が必要な適切なトリガーレベル (開始、成功、またはエラー) に割り当てます。たとえば、ジョブテンプレート 1 が失敗したときにトリガーするように、特定の通知テンプレートを割り当てることができます。この場合、通知テンプレートを **を/api/v2/job_templates/n/notification_templates_error** API エンドポイントにあるジョブテンプレートに関連付けます。
- ジョブの開始時と終了時の通知を設定できます。ユーザーとチームは、任意のジョブに割り当てることができる独自の通知を定義することもできます。

27.1. 通知の階層

通知テンプレートは、次のような親オブジェクトで定義されたテンプレートを継承します。

- ジョブテンプレートは、ジョブテンプレート用に定義された通知テンプレートを使用します。さらに、ジョブテンプレートで使用されるプロジェクトや、ジョブテンプレートがリストされている組織から通知テンプレートを継承できます。
- プロジェクト更新は、プロジェクトで定義される通知テンプレートを使用し、それに関連付けられる組織から通知テンプレートを継承します。
- インベントリー更新は、その下にリスト表示される組織で定義される通知テンプレートを使用します。
- ad hoc コマンドは、インベントリーが関連付けられる組織に定義された通知テンプレートを使用します。

27.2. 通知ワークフロー

ジョブが成功または失敗すると、エラーまたは成功ハンドラーは、通知 セクションで定義された手順を使用して、関連する通知テンプレートのリストを取得します。

次に、ジョブに関する関連詳細を含む通知オブジェクトをそれぞれに作成し、宛先に送信します。これらには、メールアドレス、Slack チャネル、SMS 番号が含まれます。

これらの通知オブジェクトは、ジョブタイプ (ジョブ、インベントリー更新、プロジェクト更新) の関連リソースとして利用できるほか、**/api/v2/notifications** からも利用できます。また、関連リソースを調べて、通知テンプレートからどのような通知が送信されたかを確認することもできます。

通知が失敗しても、それに関連付けられているジョブに影響を与えたり、ジョブが失敗したりすることはありません。通知のステータスは、詳細エンドポイント **/api/v2/notifications/<n>** で確認できます。

27.3. 通知テンプレートの作成

通知テンプレートを作成するには、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 → Notifications** を選択します。
2. **Add** をクリックします。
3. 以下のフィールドに入力します。
    - **Name**: 通知の名前を入力します。
    - **Description**: 通知の説明を入力します。このフィールドは任意です。
    - **Organization**: 通知が属する組織を指定します。
    - **Type**: ドロップダウンメニューから通知のタイプを選択します。詳細は、通知の種類 セクションを参照してください。
4. **Save** をクリックします。

27.4. 通知タイプ

次の通知タイプが Automation controller でサポートされています。

- メール
- Grafana
- IRC
- Mattermost
- PagerDuty
- Rocket.Chat
- Slack
- Twilio
- Webhook
    - Webhook ペイロード

各通知タイプには、独自の設定と動作セマンティクスがあります。さまざまな方法でテストする必要がある場合があります。さらに、各タイプの通知を特定の詳細または通知をトリガーする一連の基準までカスタマイズできます。

関連情報

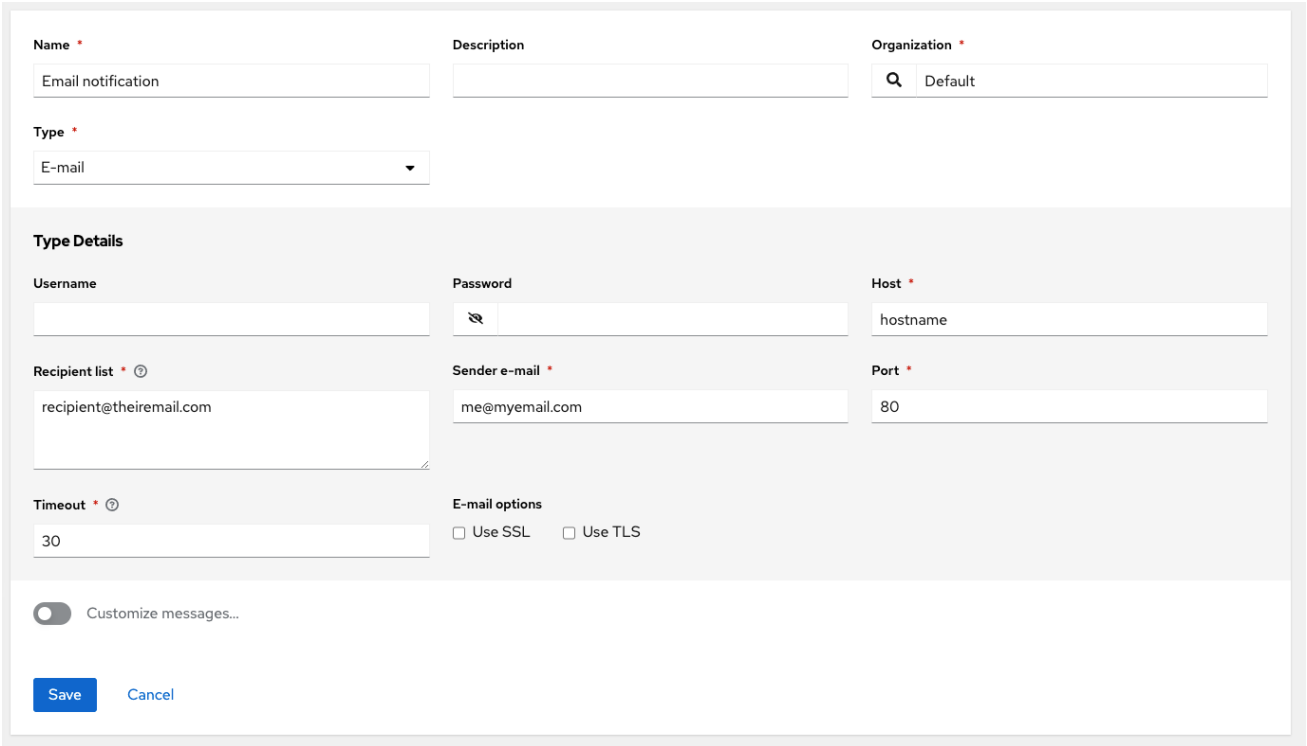
カスタム通知の設定の詳細は、カスタム通知の作成 を参照してください。次のセクションでは、各種類の通知について詳しく説明します。

27.4.1. メール

メール通知タイプは、さまざまな SMTP サーバーをサポートし、SSL/TLS 接続もサポートしています。

メール通知を設定するには、次の詳細を入力します。

- **Host**
- **Recipient list**
- **Sender e-mail**
- **Port**
- **Timeout** (秒): Automation controller がメールサーバーへの接続を試行して失敗するまでの時間を最大 120 秒まで指定できます。

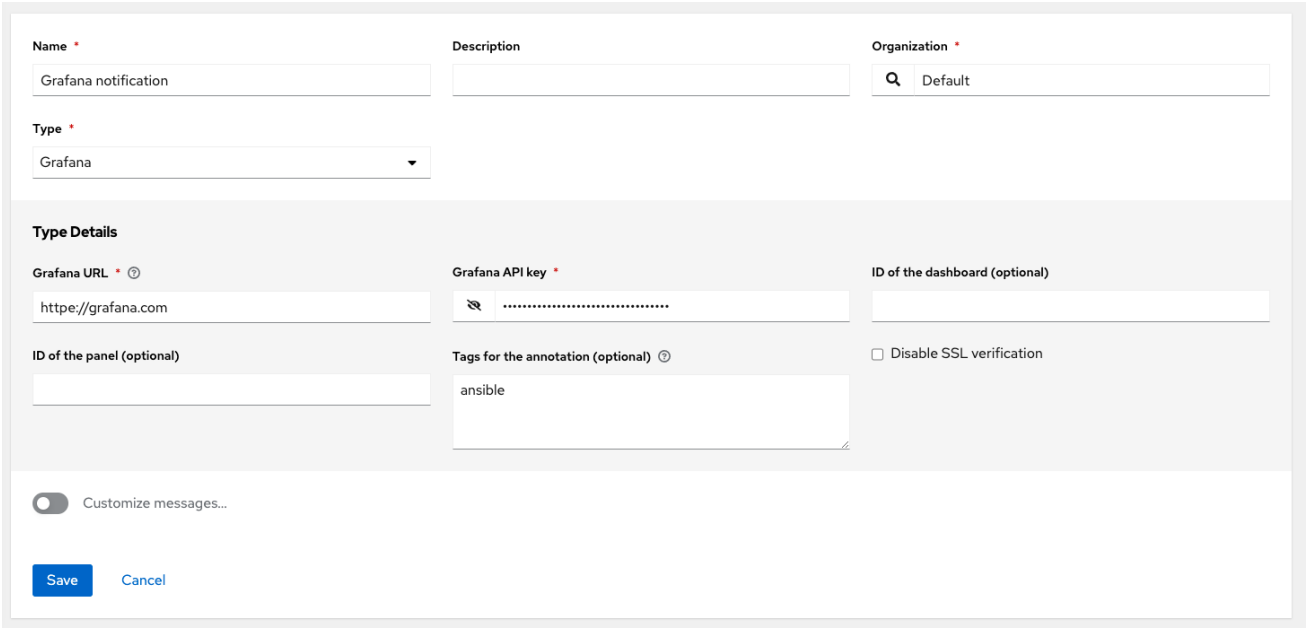


27.4.2. Grafana

Grafana を統合するには、まず Grafana システム で API キーを作成する必要があります。これは、Automation controller に与えられるトークンです。

Grafana 通知を設定するには、次の詳細を入力します。

- **Grafana URL**: Grafana API サービスの URL (例: http://yourcompany.grafana.com)。
- **Grafana API Key**: まず、Grafana システムで API キーを作成する必要があります。
- オプション: **ID of the dashboard**Grafana アカウントの API キーを作成したら、一意の ID を使用してダッシュボードを設定できます。
- オプション: **ID of the panel**パネルとグラフを Grafana インターフェイスに追加した場合に、この ID を指定できます。
- オプション: **Tags for the annotation** 設定している通知のイベントのタイプを識別するのに役立つキーワードを入力します。
- **Disable SSL verification**: SSL 検証はデフォルトでオンになっていますが、ターゲットの証明書の信頼性の検証をオフにできます。内部 CA またはプライベート CA を使用する環境の検証を無効にするには、このオプションを選択します。

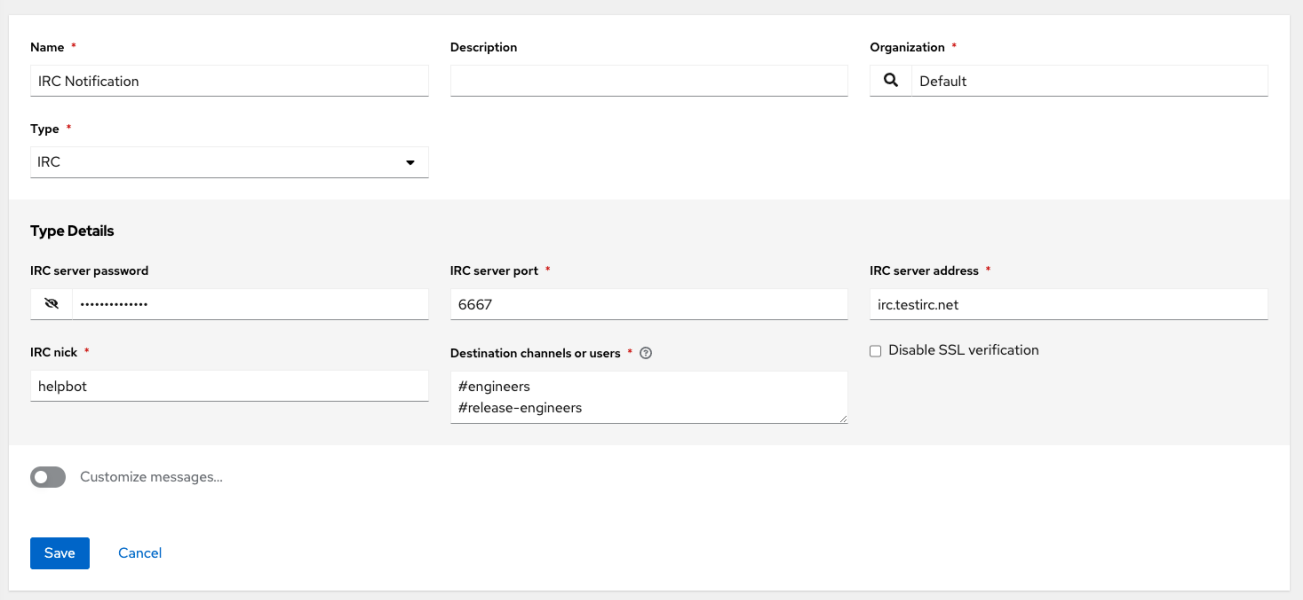


27.4.3. IRC

IRC 通知は、接続してチャネルまたは個々のユーザーにメッセージを配信し、その後切断する IRC ボットの形式をとります。通知ボットは SSL 認証もサポートしています。ボットは現在、Nickserv ID をサポートしていません。チャネルまたはユーザーが存在しないか、オンラインではない場合、通知に失敗します。障害シナリオは、接続専用に予約されています。

IRC 通知を設定するには、次の詳細を入力します。

- オプション: **IRC server password** IRC サーバーには、接続用のパスワードが必要な場合があります。サーバーでパスワードが必要でない場合は、空白のままにしておきます。**IRC server port**: IRC サーバーポート**IRC server address**: IRC サーバーのホスト名またはアドレス**IRC nick**: サーバー接続後のボットのニックネーム**Destination channels or users**: 通知の送信先となるユーザーまたはチャネルのリスト。
- オプション: **Disable SSL verification**: ボットが接続するときに SSL を使用するかどうかをチェックします。

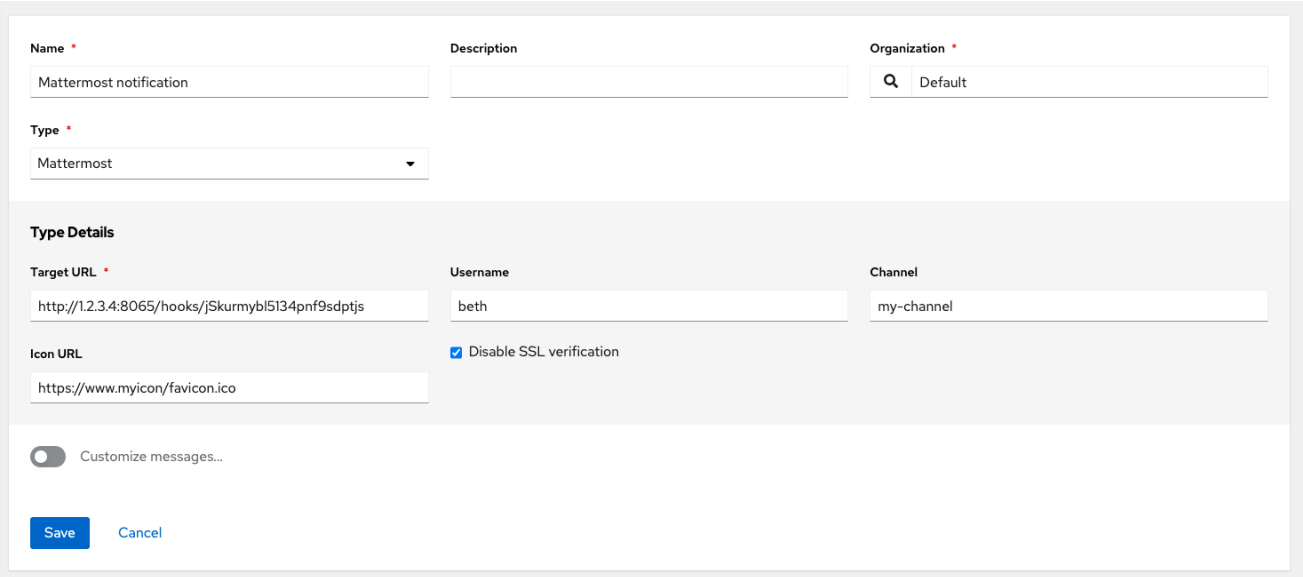


27.4.4. Mattermost

Mattermost 通知タイプは、Mattermost のメッセージングおよびコラボレーションワークスペースへのシンプルなインターフェイスを提供します。

Mattermost 通知を設定するには、次の詳細を入力します。

- ターゲット URL: Post を送信する先の完全な URL。
- オプション: **Username**: 通知のユーザー名を入力します。
- オプション: **Channel**: 通知のチャネルを入力します。
- **Icon URL**: この通知向けに表示するアイコンを指定します。
- **Disable SSL verification**: ターゲットの証明書の信頼性の検証をオフにします。内部 CA またはプライベート CA を使用する環境の検証を無効にするには、このオプションを選択します。

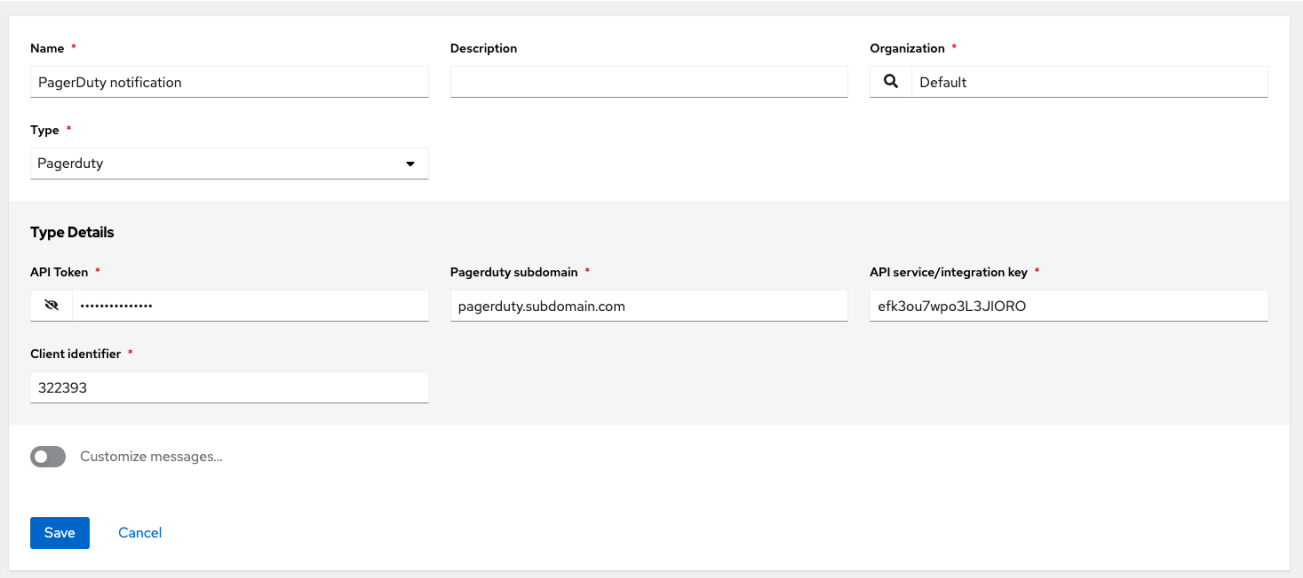


27.4.5. PagerDuty

PagerDuty を統合するには、まず PagerDuty システム で API キーを作成する必要があります。これは、Automation controller に与えられるトークンです。次に、Automation controller にも与えられる **Integration Key** を提供する **Service** を作成します。

PagerDuty 通知を設定するには、次の詳細を入力します。

- **API Token**: まず、PagerDuty システムで API キーを作成する必要があります。これは、Automation controller に与えられるトークンです。
- **PagerDuty subdomain**: PagerDuty アカウントにサインアップすると、通信するための固有のサブドメインを受け取ります。たとえば、testuser としてサインアップした場合、Web ダッシュボードは **testuser.pagerduty.com** にあり、API **testuser** を完全なドメインではなくサブドメインとして指定します。
- **API service/integration Key**: PagerDuty で作成した API サービス/統合キーを入力します。
- **Client identifier**: これは、API キーとサービスを使用しているサービスを識別するために、アラートの内容とともに PagerDuty サービスに送信されます。これは、複数の統合で同じ API キーとサービスを使用している場合に役立ちます。

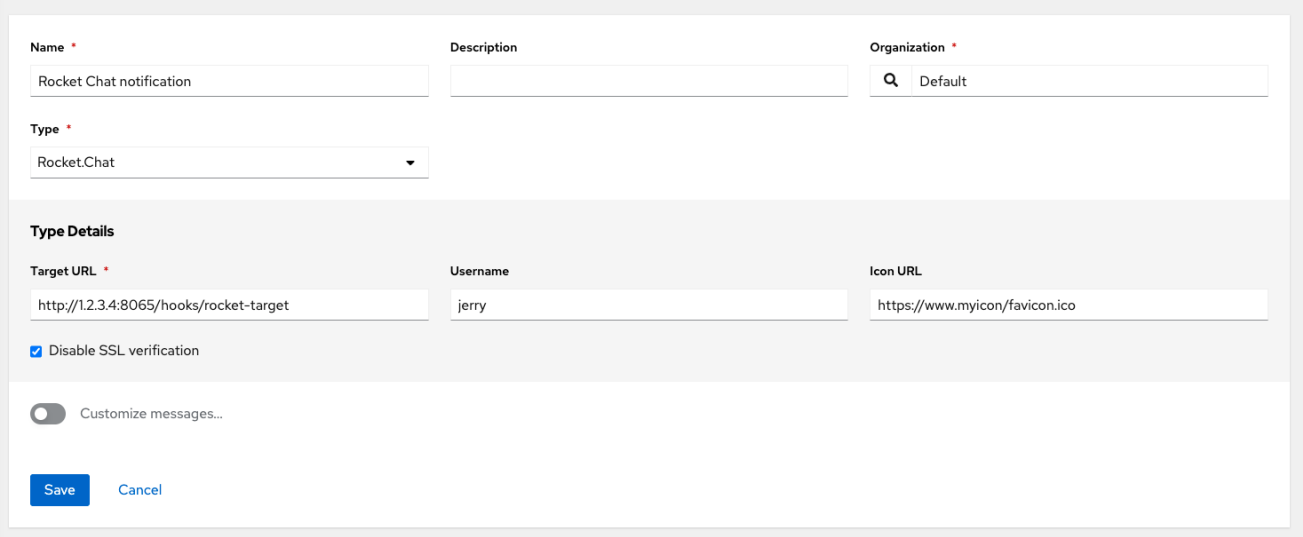


27.4.6. Rocket.Chat

Rocket.Chat 通知タイプは、Rocket.Chat のコラボレーションおよびコミュニケーションプラットフォームへのインターフェイスを提供します。

Rocket.Chat 通知を設定するには、次の詳細を入力します。

- **Target URL**: **POST** 先の完全な URL。
- オプション: **Username**: ユーザー名を入力します。
- オプション: **Icon URL**: この通知向けに表示するアイコンを指定します。
- **Disable SSL Verification**: ターゲットの証明書の信頼性の検証をオフにします。内部 CA またはプライベート CA を使用する環境の検証を無効にするには、このオプションを選択します。



27.4.7. Slack

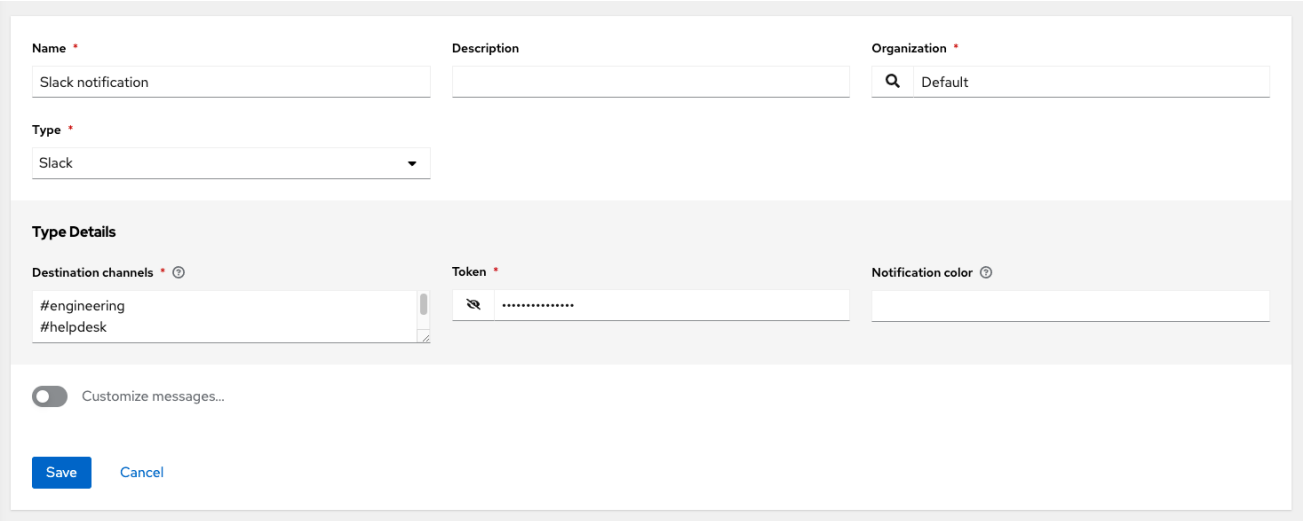
Slack は、チームの共同コミュニケーションおよびメッセージングツールです。

Slack 通知を設定するには、次の詳細を入力します。

- Slack アプリケーション。詳細は、作成方法に関する Slack ドキュメントの クイックスタート ページを参照してください。
- トークン。詳細は、現在のトークンタイプの ドキュメントページの レガシーボット とボットトークンの詳細を参照してください。

ボットまたはアプリを設定したら、次の手順を完了する必要があります。

1. **Apps** に移動します。
2. 新しく作成したアプリをクリックし、**Add features and functionality** に移動します。これにより、受信 Webhook、ボット、パーミッションを設定したり、**アプリをワークスペースにインストール** したりできます。



27.4.8. Twilio

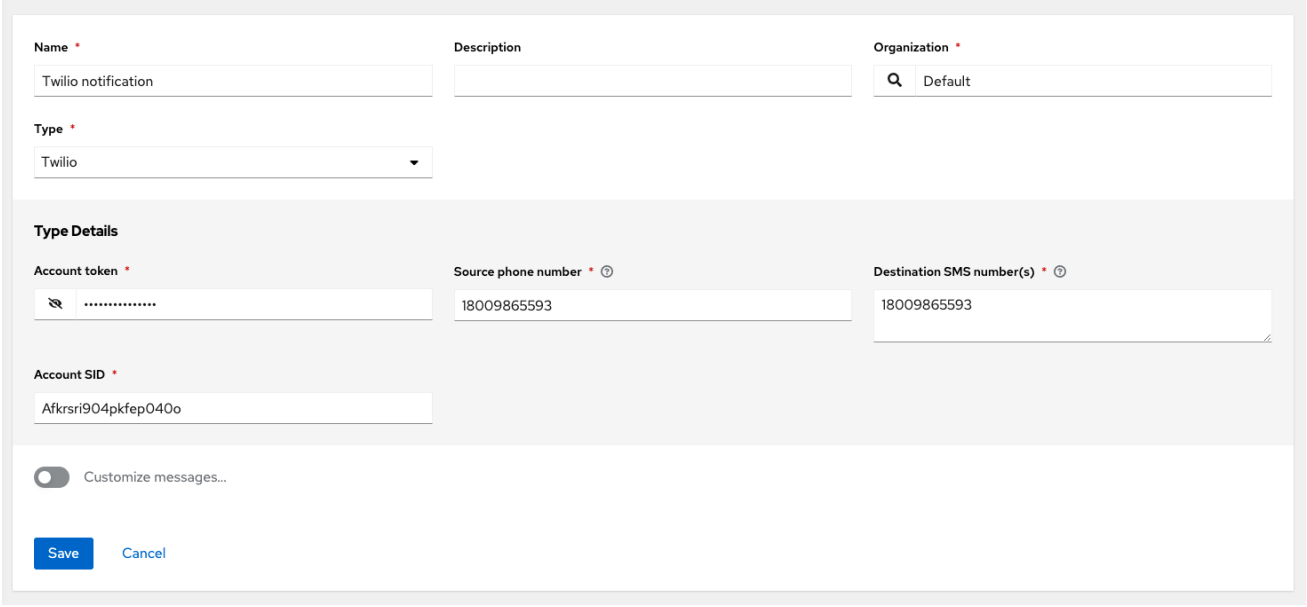
Twilio は音声と SMS の自動化サービスです。サインインしたら、メッセージの送信元となる電話番号を作成する必要があります。次に、**Programmable SMS** で **メッセージングサービス** を定義し、以前に作成した番号をそれに関連付けることができます。

この番号またはその他の情報を確認してからでないと、任意の番号に送信するときに使用できない可能性があります。**Messaging Service** には、ステータスコールバック URL は必要なく、受信メッセージを処理する機能も必要ありません。

個人 (またはサブ) アカウント設定の下に、API 認証情報があります。Twilio は 2 つの認証情報を使用して、API リクエストがどのアカウントから送信されているかを判断します。ユーザー名として機能する **Account SID** と、パスワードとして機能する **Auth Token**。

Twilio 通知を設定するには、次の詳細を入力します。

- **Account token**: アカウントトークンを入力します。
- **Source phone number**: メッセージングサービスに関連付けられた番号を +15556667777 の形式で入力します。
- **Destination SMS number(s)**: SMS を受信する番号のリストを入力します。10 桁の電話番号である必要があります。
- **Account SID**: アカウント SID を入力します。



27.4.9. Webhook

Webhook 通知タイプは、事前定義された Web サービスに **POST** を送信するためのシンプルなインターフェイスを提供します。Automation controller は、JSON 形式の関連詳細など、データペイロードを持つアプリケーションと JSON コンテンツタイプを使用して、このアドレスに **POST** を送信します。一部の Web サービス API は、HTTP リクエストが特定のフィールドを含む特定の形式であることを想定しています。

次のように Webhook 通知を設定します。

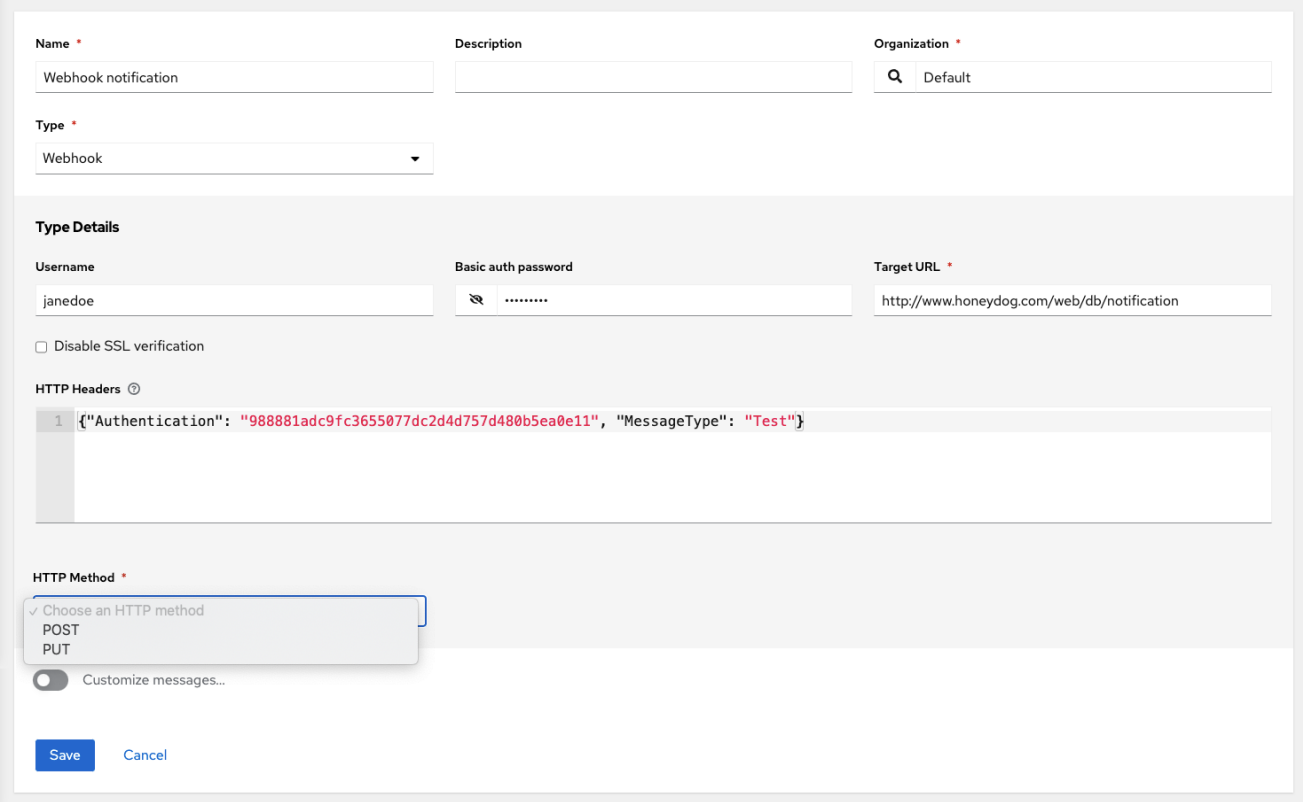
- HTTP メソッドの設定 (**POST** または **PUT** を使用)
- 送信要求の本文
- 認証の設定 (Basic 認証を使用)

Webhook 通知を設定するには、次の詳細を入力します。

- オプション: **Username**: ユーザー名を入力します。
- オプション: **Basic 認証パスワード**:
- **ターゲット URL**: Webhook 通知の **PUT** または **POST** の送信先となる完全な URL を入力します。
- **Disable SSL Verification**: SSL 検証はデフォルトでオンになっていますが、ターゲットの証明書の信頼性の検証をオフにできます。内部 CA またはプライベート CA を使用する環境の検証を無効にするには、このオプションを選択します。
- **HTTP ヘッダー**: キーと値が文字列である JSON 形式でヘッダーを入力します。以下に例を示します。

```
{"Authentication": "988881adc9fc3655077dc2d4d757d480b5ea0e11", "MessageType": "Test"}`.
```

- **HTTP メソッド**: Webhook のメソッドを選択します。
- **POST** : 新しいリソースを作成します。他のカテゴリーに該当しない演算用の汎用動詞としても機能します。Webhook サービスで **PUT** が必要であるとわかっている場合を除き、**POST** が必要になる可能性が高くなります。
- **PUT**: 特定のリソース (識別子によって) またはリソースのコレクションを更新します。リソース識別子が事前にわかっている場合は、**PUT** を使用して特定のリソースを作成することもできます。



27.4.9.1. Webhook ペイロード

次のデータは、Webhook エンドポイントの Automation controller によって送信されます。

```
job id
name
url
created_by
started
finished
status
traceback
inventory
project
playbook
credential
limit
extra_vars
hosts
http method
```

以下は、Automation controller から返される Webhook メッセージを介した **started** 通知の例です。

```
{"id": 38, "name": "Demo Job Template", "url": "https://host/#/jobs/playbook/38", "created_by":
"bianca", "started":
"2020-07-28T19:57:07.888193+00:00", "finished": null, "status": "running", "traceback": "", "inventory":
"Demo Inventory",
"project": "Demo Project", "playbook": "hello_world.yml", "credential": "Demo Credential", "limit": "",
"extra_vars": "{}",
"hosts": {}}POST / HTTP/1.1
```

次のデータは、Webhook エンドポイントの Automation controller によって **success/fail** ステータスとして返されます。

```
job id
name
url
created_by
started
finished
status
traceback
inventory
project
playbook
credential
limit
extra_vars
hosts
```

以下は、Webhook メッセージを通じて Automation controller によって返される **success/fail** 通知の例です。

```
{"id": 46, "name": "AWX-Collection-tests-awx_job_wait-long_running-XVFBGRSAvUUIrYKn", "url":
"https://host/#/jobs/playbook/46",
"created_by": "bianca", "started": "2020-07-28T20:43:36.966686+00:00", "finished": "2020-07-
28T20:43:44.936072+00:00", "status": "failed",
"traceback": "", "inventory": "Demo Inventory", "project": "AWX-Collection-tests-awx_job_wait-
long_running-JJSlglnwtsRJyQmw", "playbook":
```

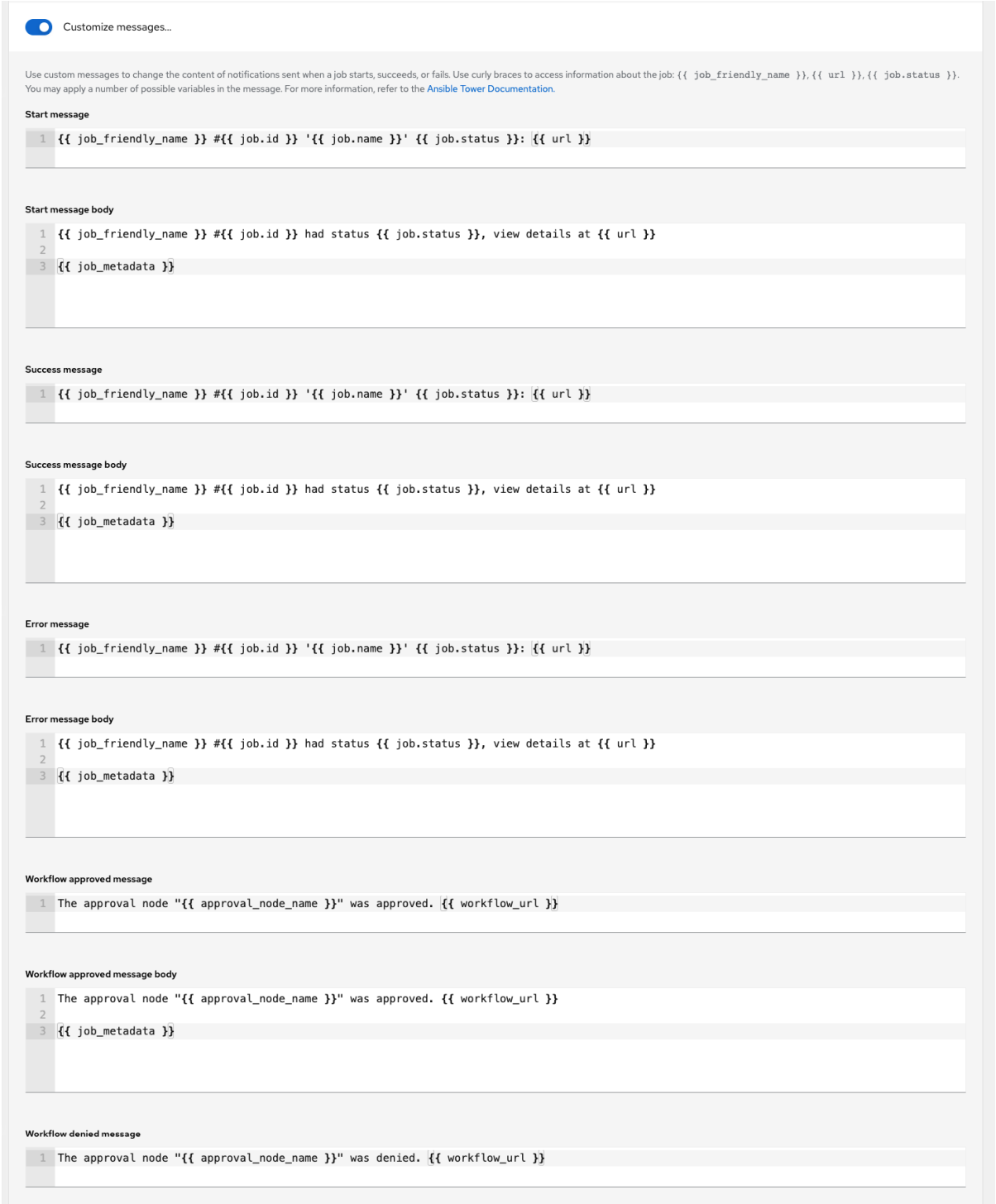
"fail.yml", "credential": null, "limit": "", "extra_vars": "{\"sleep_interval\": 300}", "hosts": {"localhost": {"failed": true, "changed": 0,
"dark": 0, "failures": 1, "ok": 1, "processed": 1, "skipped": 0, "rescued": 0, "ignored": 0}}}

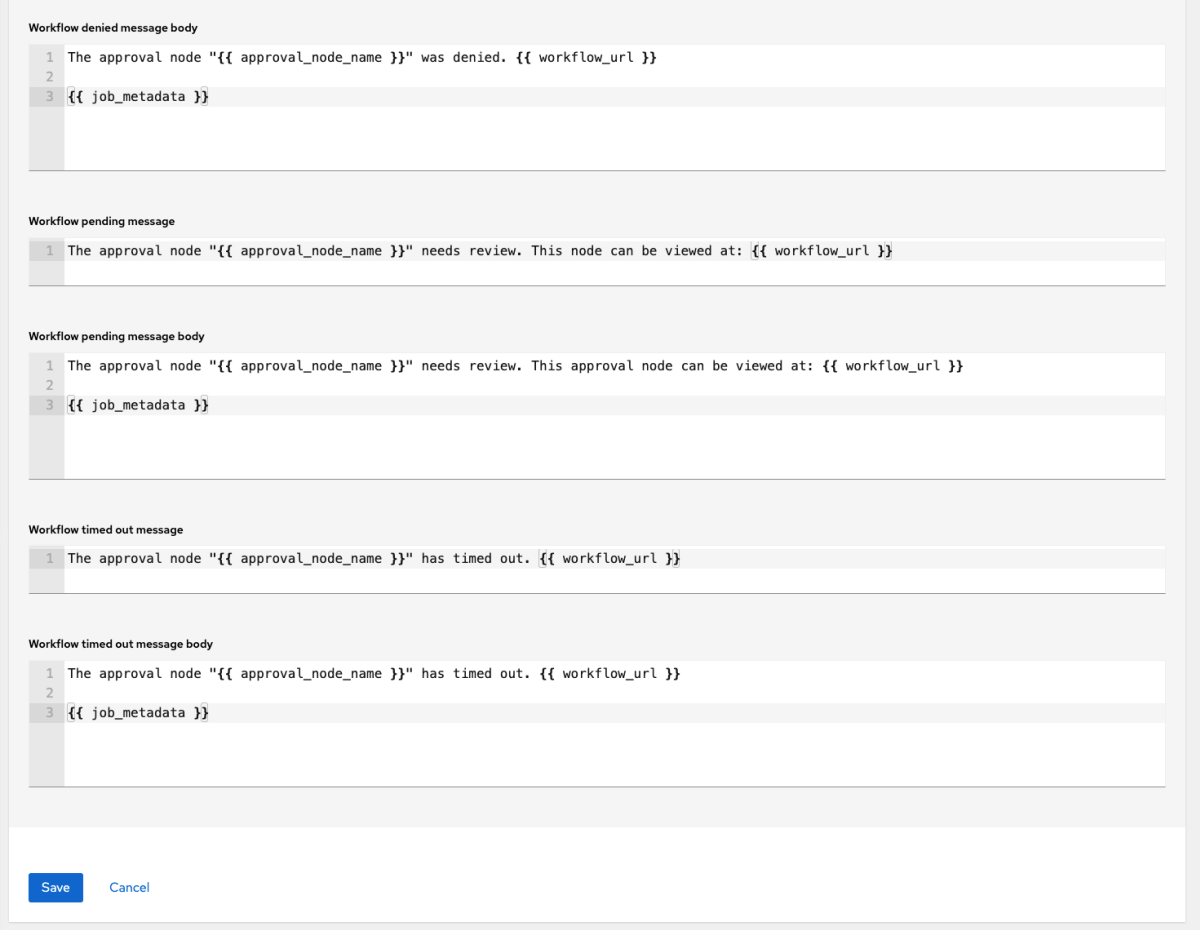
27.5. カスタム通知の作成

通知フォーム上の各 通知タイプ の テキストの内容をカスタマイズ できます。

手順

1. **Notification Templates** リストビューで **Add** をクリックします。
2. **Type** リストから通知タイプを選択します。
3. トグルを使用して **Customize messages** を有効にします。





4. さまざまなジョブイベントに、次のようなカスタムメッセージを提供できます。
    - **開始メッセージ**
    - **成功メッセージボディー**
    - **エラーメッセージ**
    - **ワークフロー承認メッセージ**
    - **ワークフロー拒否メッセージ**
    - **ワークフロー実行メッセージ**
    - **Workflow pending message**
    - **Workflow timed out message**

メッセージの形式は、設定している通知の種類によって異なります。たとえば、メールおよびPagerDuty 通知のメッセージは、本文と件名を備えた典型的なメールのように見えます。この場合、Automation controller はフィールドを **Message** および **Message Body** として表示します。他の通知タイプでは、イベントのタイプごとに **メッセージ** のみが必要です。

Message フィールドには、最上位の変数 (**id** または **name** など、属性と組み合わされている **job**) を含むテンプレートを使用して、事前に入力されます。テンプレートは中括弧で囲まれており、事前設定されたメッセージフィールドに示すように、Automation controller によって提供される固定フィールドセットから作成できます。

この事前入力フィールドは、イベントの通知の受信者に一般的に表示されるメッセージを提案します。必要に応じてジョブに独自の属性を追加することで、さまざまな基準でこれらのメッセージをカスタマイズできます。カスタム通知メッセージは、Jinja (Ansible Playbook で使用されるのと同じテンプレートエンジン) を使用してレンダリングされます。。

以下に概要を示すように、メッセージとメッセージ本文にはさまざまな種類のコンテンツがあります。

- メッセージは常に文字列のみであり、一行だけです。改行はサポートされていません。
- メッセージ本文は、ディクショナリーまたはテキストのブロックのいずれかです。
  - Webhook と PagerDuty のメッセージ本文はディクショナリー定義を使用します。これらのデフォルトのメッセージ本文は **{{ job_metadata }}** です。そのまま使用することも、独自のディクショナリーを提供することもできます。
  - メールのメッセージ本文には、テキストのブロックまたは複数行の文字列が使用されます。デフォルトのメッセージ本文は次のとおりです。

    > {{ job_friendly_name }} #{{ job.id }} had status {{ job.status }}, view details at {{ url }} {{ job_metadata }}

    このテキストは 、**{{ job_metadata }}** を残したまま編集することも、**{{ job_metadata }}** をドロップすることもできます。本文はテキストのブロックであるため、任意の文字列を使用できます。**{{ job_metadata }}** は、実行中のジョブを説明するフィールドを含むディクショナリーとしてレンダリングされます。いずれの場合も、**{{ job_metadata }}** には次のフィールドが含まれます。

    - **id**
    - **name**
    - **url**
    - **created_by**
    - **started**
    - **finished**
    - **status**
    - **トレースバック**

    注記

    {{ job_metadata }} 内の個々のフィールドをクエリーすることはできません。通知テンプレートで **{{ job_metadata }}** を使用すると、すべてのデータが返されます。

    結果のディクショナリーは次のようになります。

    > {"id": 18,

```
 "name": "Project - Space Procedures",
 "url": "https://host/#/jobs/project/18",
 "created_by": "admin",
 "started": "2019-10-26T00:20:45.139356+00:00",
 "finished": "2019-10-26T00:20:55.769713+00:00",
 "status": "successful",
 "traceback": ""
}
```


{{ job_metadata }} がジョブでレンダリングされる場合、次の追加フィールドが含まれます。

- **inventory**
- **project**
- **Playbook**
- **credential**
- **limit**
- **extra_vars**
- **hosts**
  結果のディクショナリーは次のようになります。


```
{"id": 12,
 "name": "JobTemplate - Launch Rockets",
 "url": "https://host/#/jobs/playbook/12",
 "created_by": "admin",
 "started": "2019-10-26T00:02:07.943774+00:00",
 "finished": null,
 "status": "running",
 "traceback": "",
 "inventory": "Inventory - Fleet",
 "project": "Project - Space Procedures",
 "playbook": "launch.yml",
 "credential": "Credential - Mission Control",
 "limit": "",
 "extra_vars": "{}",
 "hosts": {}
}
```


{{ job_metadata }} がワークフロージョブでレンダリングされる場合、次の追加フィールドが含まれます。

- **body** (これはワークフロージョブ内のノードを列挙し、各ノードに関連付けられたジョブの説明を含みます)
  結果のディクショナリーは次のようになります。


```
{"id": 14,
 "name": "Workflow Job Template - Launch Mars Mission",
 "url": "https://host/#/workflows/14",
 "created_by": "admin",
 "started": "2019-10-26T00:11:04.554468+00:00",
```

```
"finished": "2019-10-26T00:11:24.249899+00:00",
"status": "successful",
"traceback": "",
"body": "Workflow job summary:

        node #1 spawns job #15, \"Assemble Fleet JT\", which finished with status
successful.
        node #2 spawns job #16, \"Mission Start approval node\", which finished with
status successful.\n
        node #3 spawns job #17, \"Deploy Fleet\", which finished with status
successful."
}
```


無効な構文や使用できないフィールドの参照を使用する通知テンプレートを作成すると、エラーの内容を示すエラーメッセージが表示されます。通知のカスタムメッセージを削除すると、この場所にはデフォルトのメッセージが表示されます。

重要

カスタムメッセージを編集せずに通知テンプレートを保存した場合 (または編集してデフォルト値に戻した場合)、**Details** 画面にはデフォルトが表示されるはずで、カスタムメッセージテーブルは表示されません。いずれかの値を編集して保存すると、テーブル全体が **Details** 画面に表示されます。

関連情報

- 詳細は、 Jinja2 での変数の使用 を参照してください。
- Automation Controller には、正しいデータを取得してメッセージを表示する、有効な構文が必要です。サポート対象の属性のリストと適切な構文構造については、カスタム通知でサポートされる属性 セクションを参照してください。

27.6. 通知の有効化と無効化

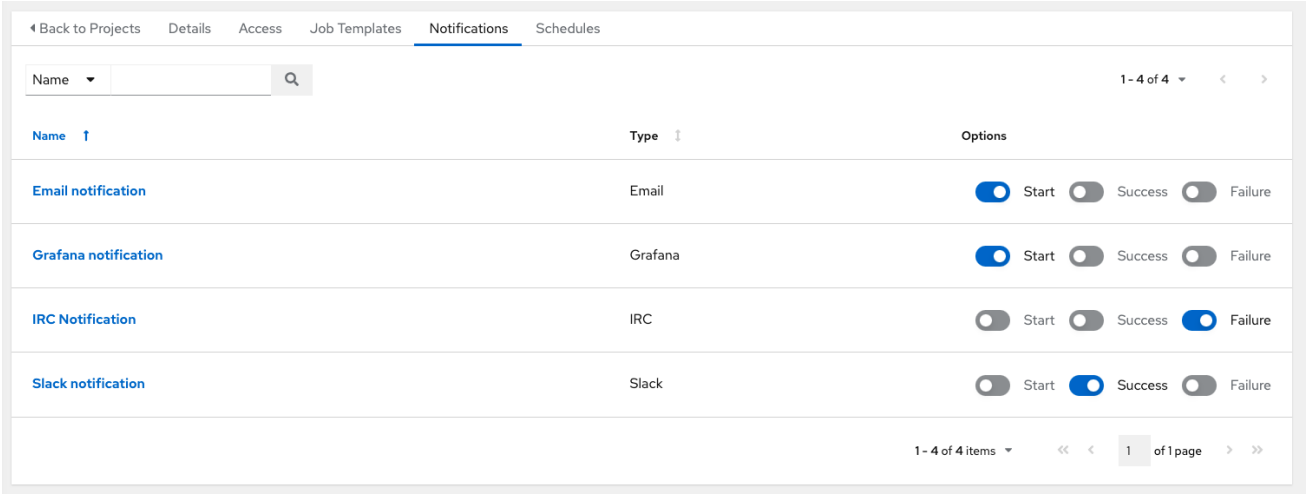
特定のジョブの開始時、およびジョブ実行終了時の成功または失敗を通知するように設定できます。次の動作に注意してください。

- ワークフロージョブテンプレートで開始時通知が有効になっており、そのワークフロー内のジョブテンプレートでも開始時通知が有効になっている場合、両方の通知を受け取ります。
- ワークフロージョブテンプレート内の多くのジョブテンプレートで通知を実行できます。
- スライスされたジョブテンプレートの開始時に通知を実行できるようにし、各スライスで通知が生成されます。
- ジョブの開始時に通知を実行できるようにし、その通知が削除された場合、ジョブテンプレートは引き続き実行されますが、エラーメッセージが表示されます。

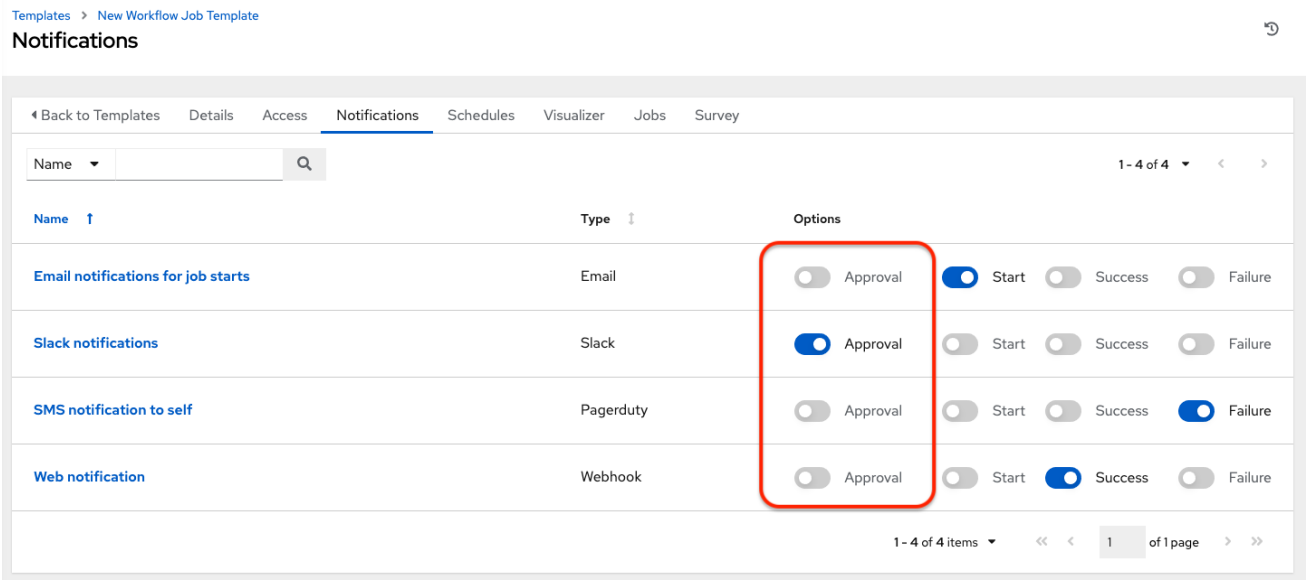
ジョブの開始、ジョブの成功、ジョブの失敗、またはこれらの任意の組み合わせの通知を、以下のリソースの **Notifications** タブから有効化できます。

- ジョブテンプレート
- ワークフローテンプレート

- プロジェクト (次の例を参照)
- インベントリーソース
- 組織



承認ノードがあるワークフローテンプレートの場合は、 **Start**、**Success**、および **Failure** に加えて、以下のように特定の承認関連のイベントを有効または無効にできます。

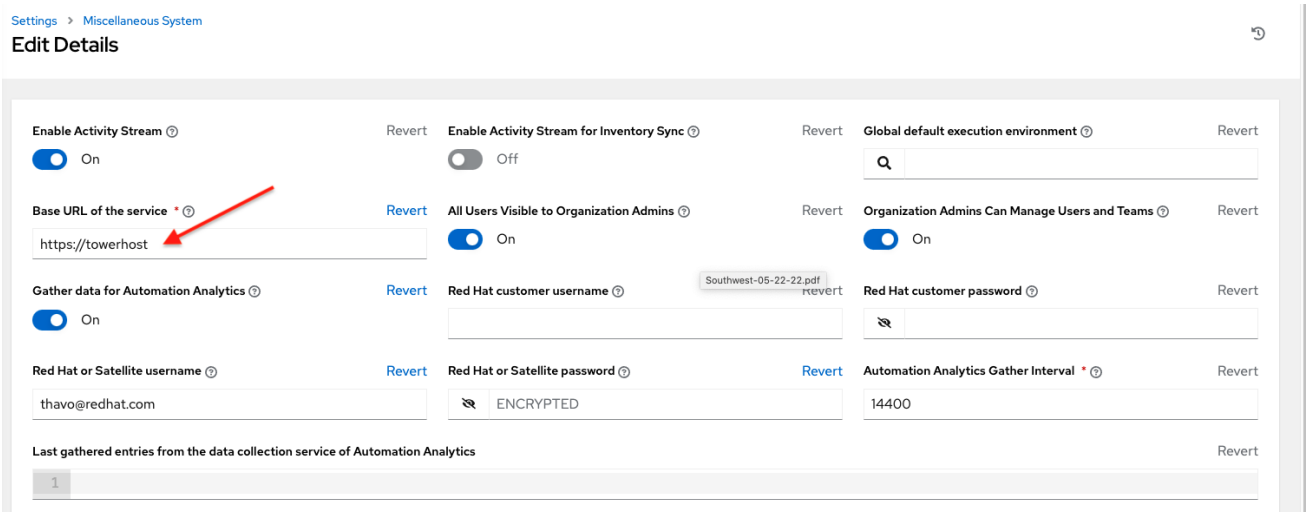


関連情報

これらのタイプのノードの操作の詳細は、承認ノード を参照してください。

27.7. 通知用のホスト名設定

System settings で、**Base URL of the service**フィールドのデフォルト値を、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。



Tower 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。

27.7.1. TOWER_URL_BASE のリセット

Automatioon Controller では、受信要求を確認して、受信要求をもとにサーバーアドレスを設定することで、ベース URL (**TOWER_URL_BASE**) の定義方法を決定します。

Automation controller は、最初にデータベースから設定値を取得します。設定値が見つからない場合は、設定ファイルの値が使用されます。Automation controller ホストの IP アドレスに移動してライセンスに対して Post を送信すると、Post が実行されたライセンスはデータベースの設定エントリーに書き込まれます。

間違ったアドレスを取得した場合は、次の手順を使用して **TOWER_URL_BASE** をリセットします。

手順

1. ナビゲーションパネルから、**Settings → Miscellaneous System settings** を選択します。
2. **Edit** をクリックします。
3. 通知に表示する DNS エントリーのアドレスを **Base URL of the service** フィールドに入力します。
4. **Settings → Subscription settings** でライセンスを再度追加します。

27.8. 通知 API

starting、**success**、または **error** エンドポイントを使用します。

```
/api/v2/organizations/N/notification_templates_started/
/api/v2/organizations/N/notification_templates_success/
/api/v2/organizations/N/notification_templates_error/
```

さらに、 **../../../N/notification_templates_started** エンドポイントには、以下に対する **GET** アクションと **POST** アクションがあります。

- 組織
- プロジェクト

- インベントリーソース
- ジョブテンプレート
- システムジョブテンプレート
- ワークフロージョブテンプレート

第28章 カスタム通知でサポートされている属性

サポートされているジョブ属性のリストと、通知のメッセージテキストを構築するための適切な構文について説明します。

サポートされているジョブ属性は次のとおりです。

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

- **name** - (文字列) このジョブの名前
- **Playbook** - (文字列) 実行された Playbook
- **scm_revision** - (文字列) このジョブに使用するプロジェクトからの SCM リビジョン (存在する場合)
- **skip_tags** - (文字列) 指定した場合は、このタグセットをスキップして Playbook を実行
- **start_at_task** - (文字列) 指定した場合は、この名前に一致するタスクで Playbook の実行を開始
- **started** - (日時) ジョブが開始するためにキューに入れられた日時
- **status** - (選択肢) **new**、**pending**、**waiting**、**running**、**success**、**failed**、**error**、**cancel**
- **timeout** - (整数) タスクが取り消されるまでの実行時間 (秒数)
- **type** - (選択肢) このジョブのデータ型。
- **url** - (文字列) このジョブの URL。
- **use_fact_cache** - (ブール値) ジョブに対して有効化されている場合、データベースへの Playbook 実行の最後に automation controller は Ansible ファクトキャッシュプラグインとして機能し、Ansibleで使用するファクトをキャッシュ。
- **verbosity** - (選択肢) 0 - 5 (正常 - WinRM デバッグに対応)
  - **host_status_counts** - (各ステータスに一意に割り当てられたホスト数)
    - **skipped**: (整数)
    - **ok**: (整数)
    - **failures**: (整数)
    - **failures**: (整数)
    - **dark**: (整数)
    - **processed**: (整数)
    - **rescued**: (整数)
    - **ignored** (整数)
    - **failed** (ブール値)
  - **summary_fields:**
    - **inventory**
      - **id** - (整数) インベントリーのデータベース ID。
      - **name** - (文字列) インベントリーの名前。
      - **description** - (文字列) インベントリーの説明 (任意)

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

■ **project**

- **id** - (int) プロジェクトのデータベース ID。
- **name** - (文字列) プロジェクトの名前。
- **description** - (文字列) プロジェクトの説明 (任意)
- **status** - (選択肢)
  new、**pending**、**waiting**、**running**、**successful**、**failed**、**error**、**canceled**、**never updated**、**ok**、または **missing** のいずれか。
- **scm_type** - (選択肢) (空の文字列)、**git**、**hg**、**svn**、**insights** のいずれか

■ **job_template**

- **id** - (int) ジョブテンプレートのデータベース ID。
- **description** - (文字列) プロジェクトの説明 (任意)
- **status** - (選択肢)
  new、**pending**、**waiting**、**running**、**successful**、**failed**、**error**、**canceled**、**never updated**、**ok**、または **missing** のいずれか。

■ **job_template**

- **id** - (int) ジョブテンプレートのデータベース ID。
- **name** - (文字列) ジョブテンプレートの名前。
- **description** - (文字列) ジョブテンプレートの説明 (任意)

- **unified_job_template**
  - **id** - (int) 統合ジョブテンプレートのデータベース ID。
  - **name** - (文字列) 統合ジョブテンプレートの名前。
  - **description** - (文字列) 統合ジョブテンプレートの説明 (任意)
  - **unified_job_type** - (選択肢) 統合ジョブタイプ (**job**、**workflow_job**、**project_update** など)。
- **instance_group**
  - **id** - (int) インスタンスグループのデータベース ID。
  - **name** - (文字列) インスタンスグループの名前。
- **created_by**
  - **id** - (int) 操作を開始したユーザーのデータベース ID。
  - **username** - (文字列) 操作を開始したユーザー名
  - **first_name** - (文字列) 名。
  - **last_name** - (文字列) 姓。
- **labels**
  - **count** - (int) ラベルの数。
  - **results** - ラベルを表すディクショナリーのリスト。例: {"id": 5, "name": "database jobs"}。

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

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

テンプレートに追加できる追加の変数は次のとおりです。

- **approval_node_name** - (文字列) 承認ノード名
- **approval_status** - (選択肢) **approved**、**denied**、および **timed_out** のいずれか
- **url** - (文字列) 通知が送信されるジョブの URL (**start**、**success**、**fail**、および **approval notifications** に適用)
- **workflow_url** - (文字列) 関連する承認ノードへの URL。これにより、通知受信者は関連するワークフロージョブページに移動して状況を確認できます。たとえば、**このノードは {{workflow_url }} で表示できます**。承認関連の通知の場合は、**url** と **workflow_url** の両方が同じです。
- **job_friendly_name** - (文字列) ジョブの分かりやすい名前

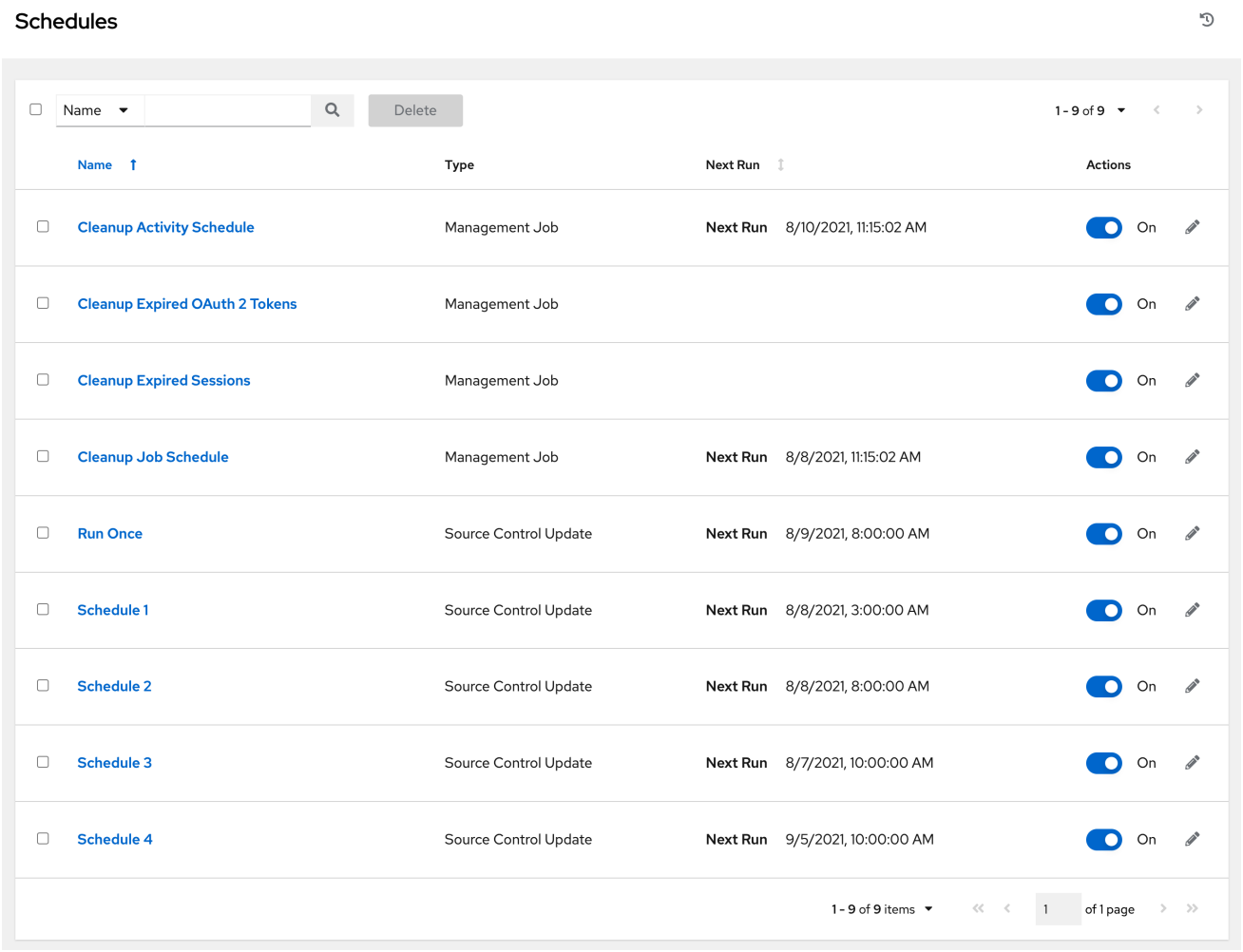
- **job_metadata** - (文字列) ジョブのメタデータを JSON 文字列として置き換えます。以下に例を示します。

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

第29章 スケジュール

ナビゲーションパネルから、**Views → Schedules** をクリックして、設定済みのスケジュールにアクセスします。スケジュールリストは、方向矢印を使用して各列の属性によって並べ替えることができます。名前、日付、またはスケジュールが実行される月で検索することもできます。

各スケジュールには、対応する **アクション** 列があり、スケジュール名の横にある **オン** または **オフ** の切り替えを使用して、そのスケジュールを有効または無効にするオプションがあります。編集 アイコンをクリックしてスケジュールを編集します。



テンプレート、プロジェクト、またはインベントリーソースを設定している場合は、**Schedules** タブをクリックして、これらのリソースのスケジュールを設定します。スケジュールを作成すると、スケジュールは次のようにリストされます。

名前

スケジュール名をクリックして詳細を開きます。

型

スケジュールがソースコントロールの更新に関連しているのか、システム管理されたジョブスケジュールに関連しているのかを識別します。

次回実行日時

次に予定されているタスクの実行

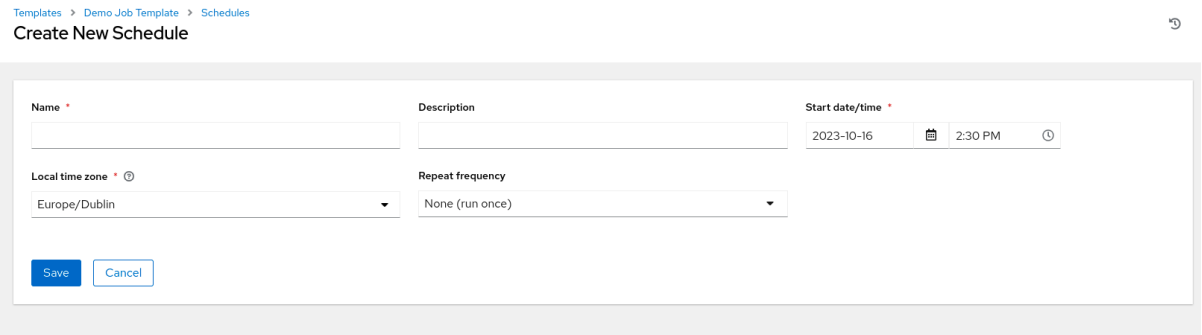
29.1. 新しいスケジュールの追加

スケジュールはテンプレート、プロジェクト、または Inventory ソースからのみ作成でき、メインの **Schedules** 画面から直接作成できません。

新規スケジュールを作成するには、以下を実行します。

手順

1. 設定しているリソースの **Schedules** タブをクリックします。これには、テンプレート、プロジェクト、またはインベントリーソースを指定できます。
2. **Add** をクリックします。これにより、**Create New Schedule** ウィンドウが開きます。



3. 以下のフィールドに該当する詳細を入力します。
   - **Name**: 名前を入力します。
   - オプション: **Description**: 説明を入力します。
   - **Start date/time**: スケジュールを開始する日時を入力します。
   - **Local time zone**: 入力する開始時刻は、このタイムゾーン内である必要があります。
   - **Repeat frequency**: 選択する頻度に応じて適切なスケジュールオプションが表示されます。スケジュールを設定すると **Schedule Details** が表示され、スケジュールの設定や、選択した **Local Time Zone** でのスケジュールの発生状況を確認することができます。

重要

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。スケジュールが保存されると、システムはローカルタイムゾーンベースの時刻を UTC に解決します。スケジュールが正しく作成されていることを確認するには、スケジュールを UTC 時間で設定します。

4. **Save** をクリックします。

オン または **オフ** の切り替えを使用して、アクティブなスケジュールを停止するか、停止したスケジュールをアクティブにします。

第30章 RED HAT ANSIBLE AUTOMATION PLATFORM 修復のための RED HAT INSIGHTS のセットアップ

Automation controller は Red Hat Insights との統合をサポートします。

ホストが Red Hat Insights に登録されると、脆弱性や既知の設定の競合がないか継続的にスキャンされます。特定された各問題には、Ansible Playbook の形式で修正を関連付けできます。

Red Hat Insights を使用している場合は、メンテナンス計画を作成して修正をグループ化し、問題を軽減するための Playbook を作成できます。Automation controller は、Red Hat Insights プロジェクトを通じてメンテナンス計画の Playbook を追跡します。

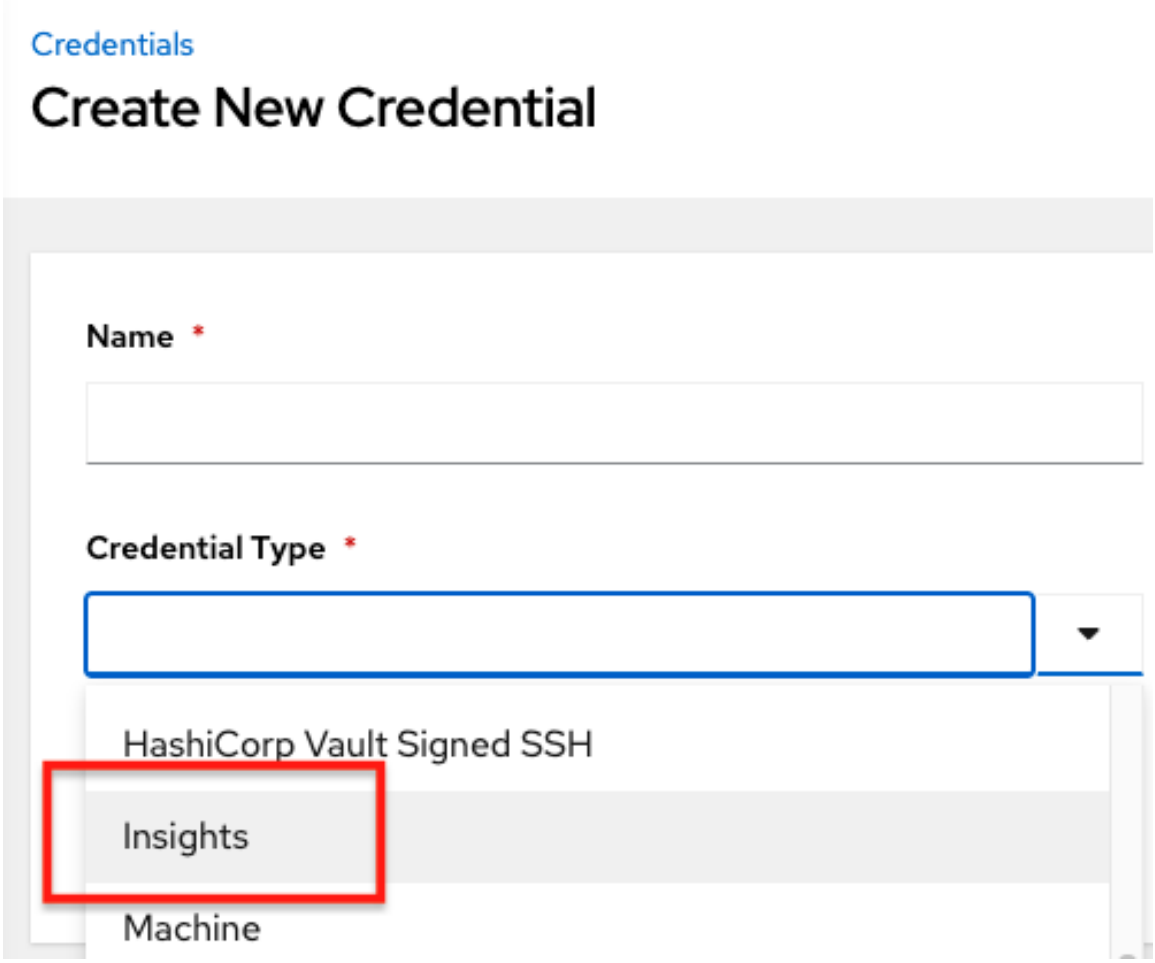
Basic 認証による Red Hat Insights への認証は特別な認証情報によって裏付けられており、最初に Automation controller で確立する必要があります。Red Hat Insights メンテナンスプランを実行するには、Red Hat Insights プロジェクトとインベントリーが必要です。

30.1. RED HAT INSIGHTS 認証情報の作成

Red Hat Insights で使用する新しい認証情報を作成するには、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 **Resources → Credentials** を選択します。
2. **Add** をクリックします。
3. 次のフィールドに適切な詳細を入力します。
   - **Name**: 認証情報の名前を入力します。
   - オプション: **Description**: 認証情報の説明を入力します。
   - オプション: **Organization**: 認証情報が関連付けられている組織の名前を入力するか、検索 アイコンをクリックして、**Select organization** ウィンドウから選択します。
   - **Credential Type**: **Insights** と入力するか、リストから選択します。



- **Username**: 有効な Red Hat Insights 認証情報を入力します。
- **Password**: 有効な Red Hat Insights 認証情報を入力します。Red Hat Insights 認証情報は、ユーザーの Red Hat カスタマーポータル アカウントのユーザー名とパスワードです。

4. **Save** をクリックします。

30.2. RED HAT INSIGHTS プロジェクトの作成

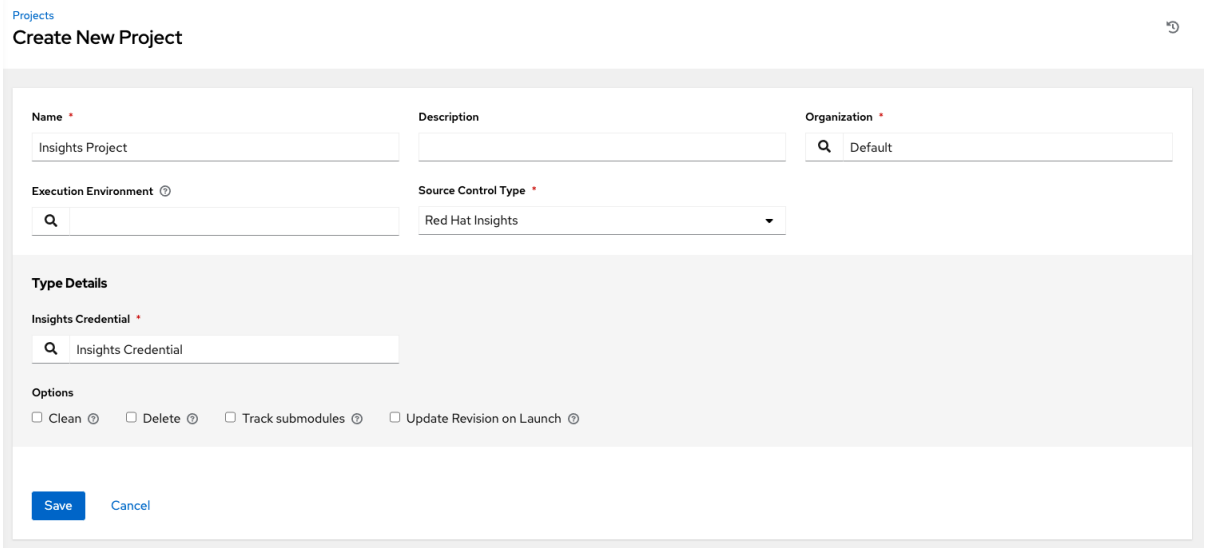
Red Hat Insights で使用する新しいプロジェクトを作成するには、次の手順を使用します。

手順

1. ナビゲーションパネルから、**Resources → Projects** を選択します。
2. **Add** をクリックします。
3. 次のフィールドに適切な詳細を入力します。次のフィールドには、特定の Red Hat Insights 関連エントリーが必要であることに注意してください。
   - **Name**: Red Hat Insights プロジェクトの名前を入力します。
   - オプション: **Description**: プロジェクトの説明を入力します。
   - **Organization**: 認証情報が関連付けられている組織の名前を入力するか、検索 ( ) アイコンをクリックし、**Select organization** ウィンドウから選択します。

- オプション: **Execution Environment**: このプロジェクトを使用するジョブで使用される実行環境。
- **Source Control Type**: **Red Hat Insights** を選択します。
- オプション: **Content Signature Validation Credential** コンテンツ署名を有効にして、プロジェクトの同期時にコンテンツが安全に保たれていることを確認します。
- **Insights Credential**: 以前に作成した Red Hat Insights 認証情報が事前に入力されます。そうでない場合は、認証情報を入力するか、検索アイコンをクリックし、**Select Insights Credential** ウィンドウから選択します。

4. **Options** フィールドからこのプロジェクトの更新オプションを選択し、必要に応じて追加の値を入力します。各オプションの詳細は、それぞれの横にあるツールヒントアイコンをクリックします。



5. **Save** をクリックします。

すべての SCM とプロジェクトの同期は、新しいプロジェクトを初めて保存するときに自動的に行われます。最新の Insights の内容で更新するには、プロジェクトのアクションの下にある更新アイコンをクリックして SCM ベースのプロジェクトを手動で更新します。

このプロセスでは、Red Hat Insights プロジェクトと Red Hat Insights アカウントソリューションが同期されます。プロジェクトの名前の横にあるステータスを示す丸は、同期が実行されたら更新される点に注目してください。

30.3. INSIGHTS インベントリーの作成

Insights Playbook には **hosts:** 行が含まれており、値は Red Hat Insights に提供されるホスト名で、Automation controller に提供されるホスト名とは異なる場合があります。

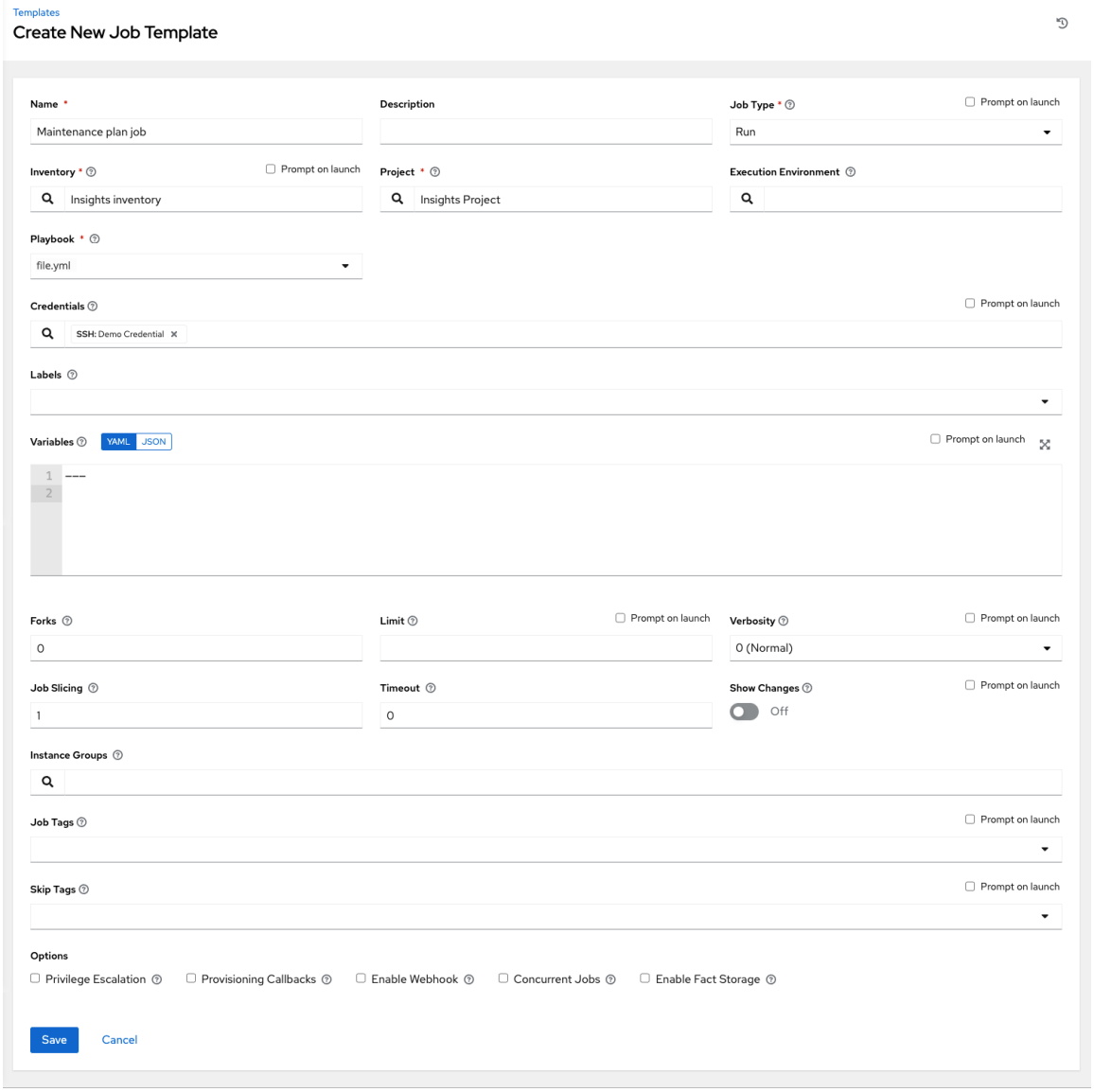
Red Hat Insights で使用する新しいインベントリーを作成するには、Insights 認証情報の作成 を参照してください。

30.4. RED HAT INSIGHTS インベントリーの修復

Red Hat Insights インベントリーを修復すると、Automation controller で 1 回クリックするだけで Red Hat Insights Playbook を実行できるようになります。これを行うには、Red Hat Insights 修復を実行するジョブテンプレートを作成します。

手順

1. ナビゲーションメニューから、**Resources → Templates** を選択します。
2. **Templates** リストビューで、**Add → Add job template** の順にクリックします。
3. 次のフィールドに適切な詳細を入力します。次のフィールドには、特定の Red Hat Insights 関連エントリーが必要であることに注意してください。
   - **Name**: メンテナンス計画の名前を入力します。
   - オプション: **Description**: ジョブテンプレートの説明を入力します。
   - **Job Type**: まだ入力されていない場合は、ジョブタイプリストから **Run** を選択します。
   - **Inventory**: 以前に作成した Red Hat Insights インベントリーを選択します。
   - **Project**: 以前に作成した Red Hat Insights プロジェクトを選択します。
   - オプション: **Execution Environment**: 実行に使用されるコンテナーイメージ。
   - **Playbook**: 実行するメンテナンスプランに関連付けられた Playbook を Playbook リストから選択します。
   - オプション: **Credentials**: このプロジェクトに使用する認証情報を入力するか、検索 ( ) アイコンをクリックし、ポップアップウィンドウから選択します。認証情報は Red Hat Insights 認証情報である必要はありません。
   - **Verbosity**: デフォルト設定をそのまま使用するか、リストから目的の詳細度を選択します。



4. **Save** をクリックします。

5. 起動 アイコンをクリックしてジョブテンプレートを起動します。

完了すると、ジョブの結果が **Job Details** ページに表示されます。

第31章 AUTOMATION CONTROLLER のベストプラクティス

以下では、Automation controller の使用に関するベストプラクティスについて説明します。

31.1. ソースコントロールの使用

Automation controller は、サーバーに直接保存された Playbook をサポートします。したがって、Playbook、ロール、および関連する詳細をソースコントロールに保存する必要があります。このようにして、インフラストラクチャーを自動化するルールを変更した時期と理由を説明する監査証跡を取得できます。さらに、インフラストラクチャーまたはチームの他の部分と Playbook を共有できるようになります。

31.2. ANSIBLE ファイルおよびディレクトリー構造

複数のプロジェクトで使用する共通のロールセットを作成している場合、これらのロールには、ソースコントロールサブモジュール、または **/opt** などの共通の場所を介してアクセスする必要があります。プロジェクトにおいて、他のプロジェクトからロールやコンテンツがインポートされるとの想定はしないでください。

詳細は、Ansible ドキュメントのリンク 一般的なヒント を参照してください。

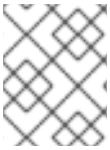
注記

- Automation controller では **vars_prompt** の質問を対話的に許可しないため、Playbook の **vars_prompt** 機能の使用は避けてください。**vars_prompt** の使用を避けられない場合は、Surveys 機能を参照してください。
- Automation controller では対話的に一時停止をキャンセルできないため、タイムアウトなしで Playbook の **pause** 機能を使用することは避けてください。どうしても **pause** の使用を避けられない場合は、タイムアウトを設定する必要があります。

ジョブは Playbook ディレクトリーを現在の作業ディレクトリーとして使用しますが、これに依存するのではなく **playbook_dir** 変数を使用するようにジョブをコーディングする必要があります。

31.3. 動的インベントリーソースの使用

インフラストラクチャー用に外部の信頼できるソースが設定されている場合は、それがクラウドプロバイダーであろうとローカル CMDB であろうと、インベントリーの同期プロセスを定義し、動的インベントリーのサポート (クラウドインベントリーソースを含む) を使用することが最も適切な方法になります。これにより、インベントリーの状態が常に最新の状態に保たれます。



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。

31.4. インベントリーの変数管理

group_vars/ と **host_vars/** を使用するのではなく、変数データをホストとグループの定義とともに保持します (インベントリーエディターを参照)。動的インベントリーソースを使用する場合、**Overwrite Variables** オプションが設定されていない限り、Automation controller は、上記の変数をデータベース

と同期できます。

31.5. 自動スケーリング

コールバック機能を使用して、新しく起動するインスタンスが自動スケーリングシナリオまたはプロビジョニング統合の設定を要求できるようにします。

31.6. 大規模なホスト数

実行の並列性を高めるには、ジョブテンプレートのフォークをより大きな値に設定します。Ansible のチューニングの詳細は、Ansible ブログ を参照してください。

31.7. 継続的インテグレーションまたは継続的デプロイ

Jenkins などの継続的インテグレーションシステムでジョブを生成するには、ジョブテンプレートに対して **CURL** リクエストを作成する必要があります。ジョブテンプレートへの認証情報では、特定のパスワードの入力を求めるプロンプトは使用しないでください。設定と使用の手順は、Ansible ドキュメントの インストール を参照してください。

第32章 セキュリティー

次のセクションでは、Automation controller がファイルシステムのセキュリティーをどのように処理し、制御できるようにするかについて説明します。

すべての Playbook は、**awx** ファイルシステムユーザーを通じて実行されます。ジョブを実行する場合、Automation controller は Linux コンテナーを使用してジョブを分離します。このように保護することで、ジョブは、ジョブテンプレートのプロジェクトディレクトリーにある Playbook、ロール、およびデータにのみアクセスできるようになります。

認証情報のセキュリティー上、ロックされた SSH 鍵をアップロードし、ロック解除パスワードを ask に設定できます。システムに SSH 認証情報または sudo パスワードをデータベースに保存させるのではなく、入力を求めるプロンプトを表示するように選択することもできます。

32.1. PLAYBOOK のアクセスおよび情報共有

Automation Controller は、自動化実行環境および Linux コンテナーを使用しているため、Playbook がプロジェクトディレクトリー外のファイルを読み取ることはできません。

デフォルトでは、コンテナー内の ansible-playbook プロセスに公開されるデータは、現在使用しているプロジェクトのみです。

これをジョブ設定でカスタマイズし、追加のディレクトリーをホストからコンテナーに公開できます。

32.1.1. 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必要がある場合は、Playbook の実行をカスタマイズする必要があります。ジョブの分離を設定するには、変数を設定します。

デフォルトでは、Automation controller はシステムの **tmp** ディレクトリー (デフォルトでは **/tmp**) をステージング領域として使用します。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"
```

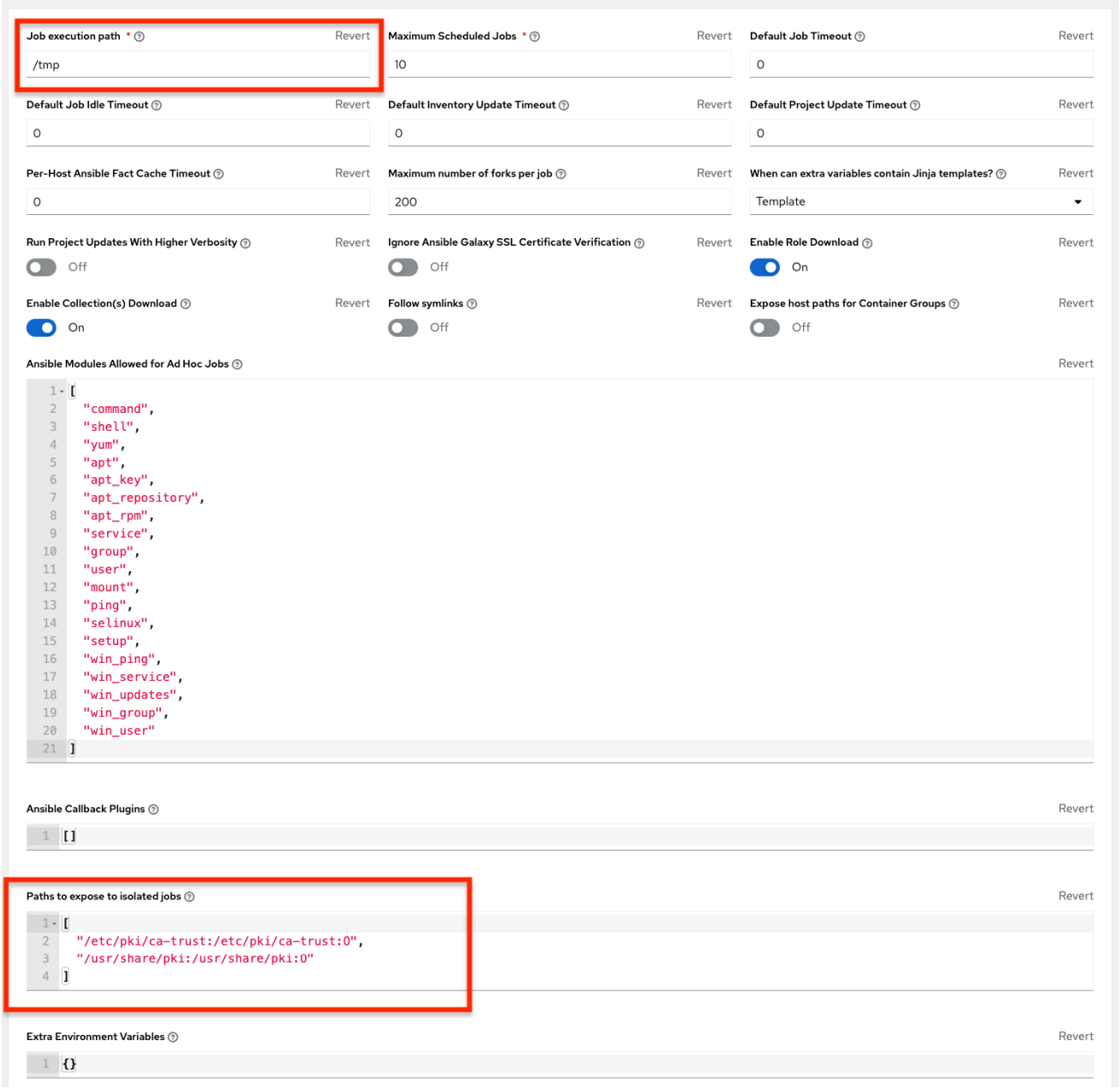
ホストから Playbook を実行するコンテナーに追加のディレクトリーを公開する場合は、**Jobs settings** ページの **Paths to Expose to Isolated Jobs**フィールド、または **/api/v2/settings/jobs** の REST API で以下を使用することで指定できます。

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

注記

Playbook で **AWX_ISOLATION_SHOW_PATHS** で定義されたキーまたは設定を使用する必要がある場合は、このファイルを **/var/lib/awx/.ssh** に追加します。

ここで説明するフィールドは、**Jobs settings** ページにあります。



32.2. ロールベースのアクセス制御

ロールベースのアクセス制御 (RBAC) は Automation controller に組み込まれており、管理者はサーバーインベントリー、組織などへのアクセスを委任できます。管理者はさまざまな認証情報の管理を一元化することもできます。管理を一元化することで、エンドユーザーが必要なシークレットをエンドユーザー自身に表示せずに利用できるようになります。RBAC を使用すると、Automation controller を有効にしてセキュリティーを強化し、管理を合理化できます。

RBAC は、ユーザーまたはチームにロールを付与する方法です。RBAC は、ロールの観点から考えることができます。ロールは、特定の機能が設定されている “オブジェクト” を、誰または何が参照、変更、または削除できるかを正確に定義したものです。

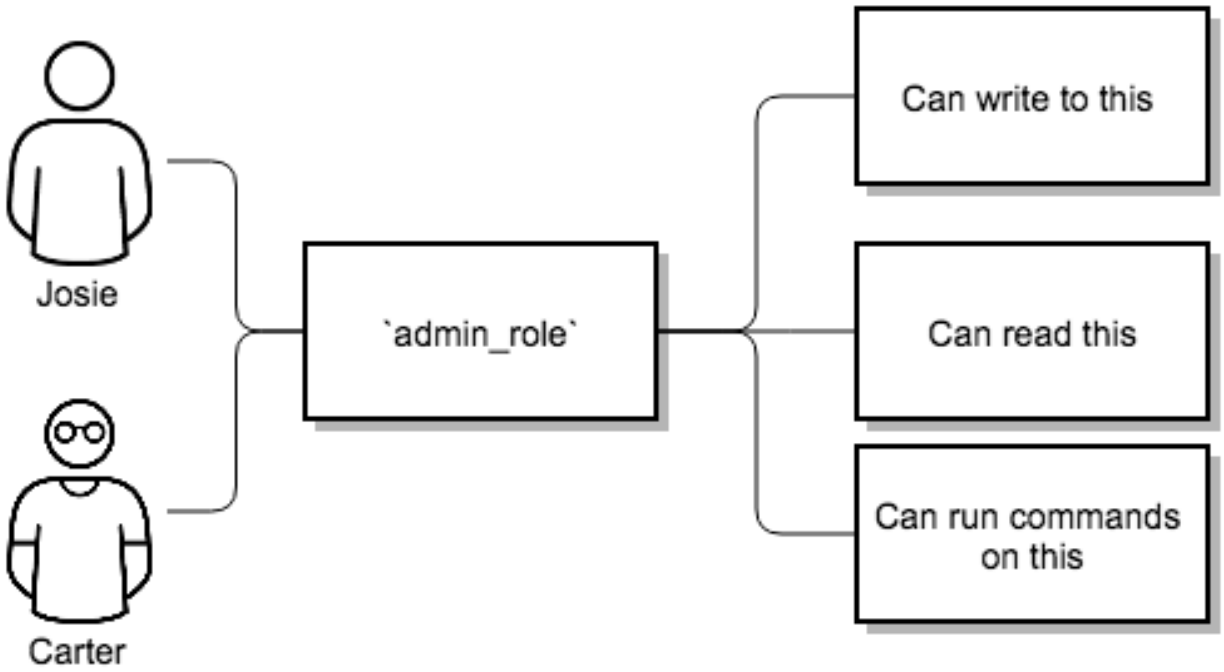
Automation controller の RBAC 設計のロール、リソース、およびユーザーの主な概念は次のとおりです。

- ユーザーはロールのメンバーになることができます。これにより、そのロールに関連付けられたリソース、または “子孫” ロールに関連付けられたリソースへの特定のアクセスが許可されます。

- ロールは機能の集合です。
- ユーザーには、割り当てられているロール、またはロール階層を通じて継承したロールを通じて、これらの機能および Automation Controller のリソースへのアクセスが許可されます。
- ロールは、機能のグループをユーザーのグループに関連付けます。機能はすべて、ロール内のメンバーシップから得られます。ユーザーは、割り当てられているロールを通じて、またはロール階層を通じて継承したロールを通じてのみ機能を受け取ります。ロールのすべてのメンバーは、そのロールに付与されたすべての機能を持ちます。組織内において、ロールは比較的一定ですが、ユーザーと機能は両方とも多数あり、短期間で変更されることがあります。
- ユーザーは多くのロールを持つことができます。

32.2.1. ロールの階層およびアクセスの継承

SomeCompany という名前の組織があり、Josie と Carter という 2 人の人に、その組織に関連付けられたすべての設定を管理できるアクセス権を付与する必要があるとします。これを行うには、両方のユーザーを組織の **admin_role** のメンバーに指定する必要があります。

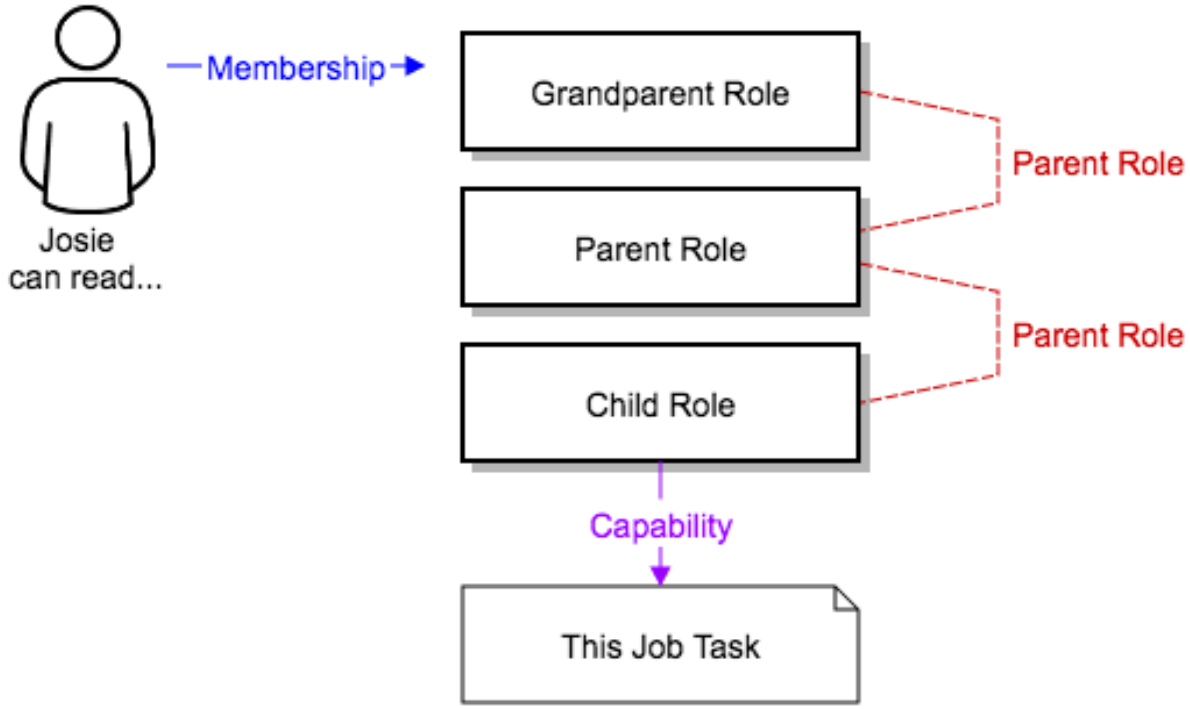


システムには多くのロールが存在することがよくあり、その一部には他のロールの機能をすべて含める必要があります。たとえば、組織管理者のアクセスできる内容をすべて、システム管理者に割り当て、プロジェクト管理者のアクセスできる内容をすべて組織管理者に割り当てる場合などです。

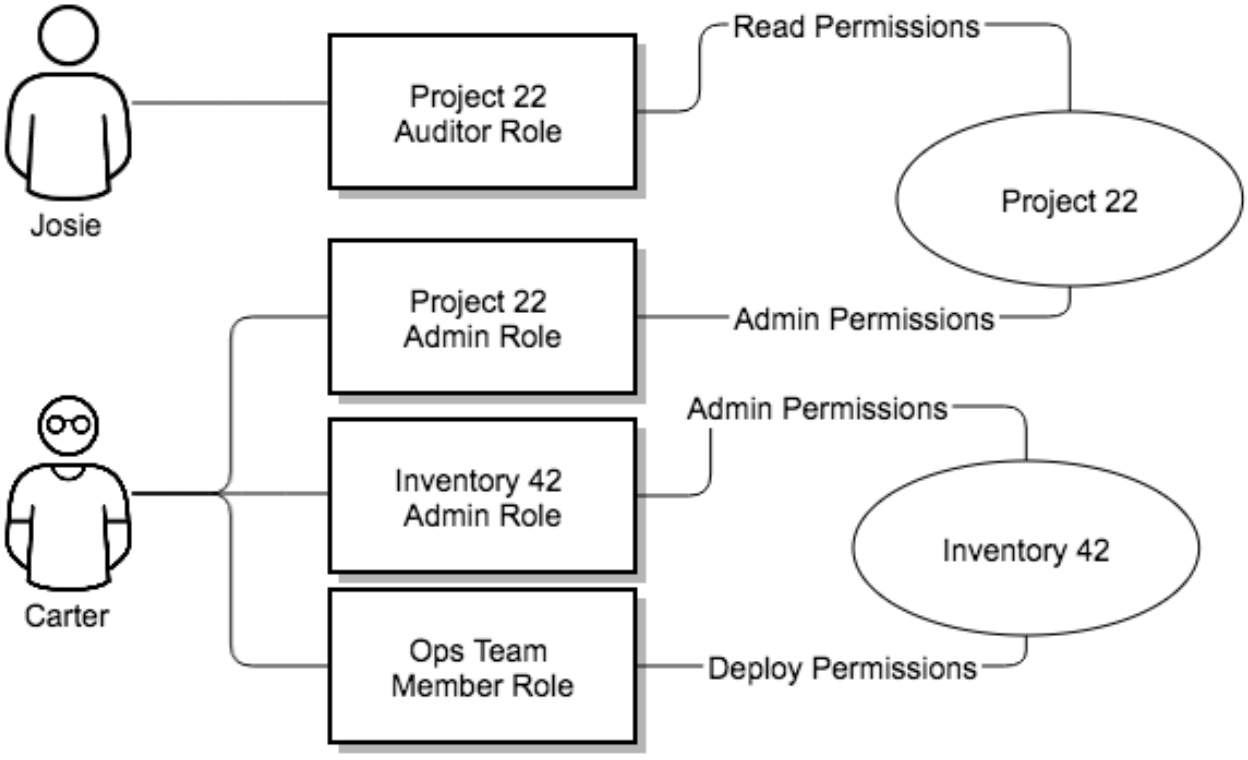
この概念はロールの階層と呼ばれています。

- 親ロールは、子ロールに付与するすべての機能を取得します。
- ロールのメンバーは、それらがメンバーであるロールと子ロールのすべての機能を自動的に取得します。

ロール階層は、ロールに「親ロール」を指定できるようにすることで表現します。ロールが持つ機能はすべて、親ロール (またはその親の親) に暗黙的に付与されます。



ロールには複数の親を持つことができ、機能はすべての親に暗黙的に付与されます。



また、RBAC を使用すると、ユーザーおよびユーザーのチームが特定のホストのセットに対して Playbook を実行することを明示的に許可できます。ユーザーとチームは、機能が付与されている Playbook とホストのセットのみに制限されます。Automation controller を使用すると、必要な数のユーザーとチームを作成またはインポートしたり、ユーザーとチームを手動で作成したり、LDAP または Active Directory からインポートしたりできます。

32.2.1.1. RBAC の使用

以下では、Automation controller の RBAC システムを環境に適用する方法について説明します。

32.2.1.1.1. ユーザーの編集

ユーザーを編集する場合、Automation controller システム管理者は、ユーザーを **System Administrator** (スーパーユーザーとも呼ばれる) または **System Auditor** として指定できます。

- システム管理者は、環境内のすべてのオブジェクトの全機能 (読み取り/書き込み/実行) を暗黙的に継承します。
- システム監査者は、環境内のすべてのオブジェクトの読み取り専用機能を暗黙的に継承します。

32.2.1.1.2. 組織の編集

組織の編集時に、システム管理者は以下のロールを指定できます。

- 組織管理者としての 1 ユーザー以上のユーザー
- 組織監査者としての 1 ユーザー以上のユーザー
- 組織メンバーとしての 1 ユーザー以上のユーザー (またはチーム)

組織のメンバーであるユーザー/チームは組織管理者を表示できます。

組織管理者であるユーザーは、組織内のすべてのオブジェクトのすべての機能を暗黙的に継承します。

組織監査者であるユーザーは、組織内のすべてのオブジェクトの読み取り専用機能を暗黙的に継承します。

32.2.1.1.3. 組織内のプロジェクト編集

管理者となっている組織のプロジェクトの編集時に、組織管理者および組織管理者は以下を指定できます。

- プロジェクト管理者である 1 つ以上のユーザー/チーム
- プロジェクトメンバーである 1 つ以上のユーザー/チーム
- その組織のメンバーであるユーザーおよびチームの中から、SCM からプロジェクトを更新できる 1 つ以上のユーザーまたはチーム。

プロジェクトのメンバーであるユーザーはプロジェクト管理者を表示できます。

プロジェクト管理者は、SCM からプロジェクトを更新する機能を暗黙的に継承します。

管理者は、ジョブテンプレートでプロジェクトを使用できる 1 以上のユーザー/チームを (プロジェクトのメンバーから) 指定することもできます。

32.2.1.1.4. 組織内でのインベントリーおよび認証情報の作成

認証情報の使用、読み取り、または書き込み用に付与されるアクセス権はすべて、ロールを使用して処理されます。ロールは、Automation Controller の RBAC システムを使用して、ownership、auditor、または usage のロールを付与します。

システム管理者および組織管理者は、管理機能に基づいて、組織内にインベントリーや認証情報を作成できます。

インベントリーまたは認証情報の編集のどちらの場合でも、システム管理者および組織管理者は、インベントリーまたは認証情報の使用機能を付与するために、(組織のメンバーから)1以上のユーザー/チームを指定できます。

システム管理者および組織管理者は、インベントリーを (動的または手動で) 更新する機能を持つ1人以上のユーザーまたはチームを (その組織のメンバーから) 指定できます。管理者は、インベントリーに対してアドホックコマンドを実行することもできます。

32.2.1.1.5. ジョブテンプレートの編集

システム管理者、組織管理者およびプロジェクト管理者は、管理機能に基づき、プロジェクト内でプロジェクトの新規ジョブテンプレートを作成したり、変更したりできます。

ジョブテンプレートの編集時に、管理者 (automation controller、組織、およびプロジェクト) は使用機能を持つ組織内のインベントリーおよび認証情報を選択したり、実行時に選択されるようにそれらのフィールドを空白にしたりすることができます。

さらに、そのジョブテンプレートの実行機能を持つ1つ以上のユーザーまたはチーム (そのプロジェクトのメンバーであるユーザーまたはチーム) を指定できます。実行機能は、ジョブテンプレートで指定されたインベントリーまたは認証情報に対してユーザーまたはチームに付与された明示的な機能に関係なく有効です。

32.2.1.1.6. ユーザービュー

ユーザーは以下を実行できます。

- それらのユーザーがメンバーとなっている組織またはプロジェクトの確認
- それらのユーザーに属する認証情報オブジェクトの作成
- 実行機能が付与されたジョブテンプレートの表示および実行

ユーザーに実行機能が付与されているジョブテンプレートでインベントリーまたは認証情報が指定されていない場合、ユーザーは実行時に、所有する組織または使用機能が付与されている組織内のインベントリーと認証情報の中から選択するよう求められます。

ジョブテンプレート管理者であるユーザーは、ジョブテンプレートを変更できます。ただし、ジョブテンプレートで使用するインベントリー、プロジェクト、Playbook、認証情報を変更するには、使用中または設定中のプロジェクトやインベントリーに対する使用ロールも必要になります。

32.2.1.2. ロール

認証情報の使用、読み取り、または書き込み用に付与されるすべてのアクセスはロールを通じて処理され、ロールはリソースに対して定義されます。

32.2.1.2.1. 組み込みロール

次の表に、RBAC システムのロールと、そのロールが Automation controller の特権に関してどのように定義されるかについての簡単な説明を示します。

システムロール	実行できる内容
システム管理者: システム全体のシングルトンロール (単一の管理者)	システムのすべての側面を管理します。
システム監査者: システム全体のシングルトンロール (単一の監査者)	システムのすべての側面を閲覧できます。
アドホックロール: インベントリー	インベントリーでアドホックコマンドを実行します。
管理者ロール: 組織、チーム、インベントリー、プロジェクト、ジョブテンプレート	定義された組織、チーム、インベントリー、プロジェクトまたはジョブテンプレートのすべての側面を管理します。
監査者ロール: すべて	定義された組織、プロジェクト、インベントリー、またはジョブテンプレートのすべての側面を表示します。
実行ロール: ジョブテンプレート	割り当てられたジョブテンプレートを実行します。
メンバーロール: 組織、チーム	ユーザーは定義された組織またはチームのメンバーです。
読み取りロール: 組織、チーム、インベントリー、プロジェクト、ジョブテンプレート	定義された組織、チーム、インベントリー、プロジェクトまたはジョブテンプレートのすべての側面を閲覧できます。
更新ロール: プロジェクト	設定されたソースコントロール管理システムからプロジェクトを更新します。
更新ロール: インベントリー	クラウドソース更新システムを使用してインベントリーを更新します。
所有者ロール: 認証情報	この認証情報のすべての側面を所有し、管理します。
ロールの使用 - 認証情報、インベントリー、プロジェクト、IG、CG	ジョブテンプレートで認証情報、インベントリー、プロジェクト、IG、または CG を使用します。

Singleton ロールは、システム全体のパーミッションを付与する特別なロールです。Automation controller は現在 2 つの組み込み Singletonロールを提供していますが、Singleton ロールを作成またはカスタマイズする機能は現時点ではサポートされていません。

32.2.1.3. 一般的なチームロール: "Personas"

Automation controller のサポート担当者は通常、Automation controller が利用可能であることを確認し、サポート性とユーザーの使いやすさのバランスを取りながら Automation controller を管理します。Automation controller サポート担当者は、多くの場合、ユーザーに **Organization Owner** または **Administrator** のロールを割り当て、ユーザーが新しい組織を作成したり、それぞれのアクセスに必要なチームのメンバーを追加したりできるようにします。これにより、サポート担当者の数を最小限に抑えられ、サービスの稼働時間を維持し、Automation controller を使用しているユーザーを支援することに重点が置かれます。

次の表で、Automation controller 組織で管理されるいくつかの一般的なロールを紹介します。

システムロール (組織向け)	一般的なユーザーロール	説明
Owner	チームリード - テクニカルリード	このユーザーは、組織内の他のユーザーのアクセスを制御できます。プロジェクト、インベントリー、ジョブテンプレートに、ユーザー固有のアクセスを追加、削除、付与できます。このタイプのユーザーは、組織のプロジェクト、テンプレート、インベントリー、チーム、認証情報のあらゆる機能を作成、削除、または変更することもできます。
Auditor	セキュリティーエンジニア - プロジェクトマネージャー	このアカウントは、組織のすべての側面を読み取り専用モードで表示できます。このロールは、コンプライアンスの確認および管理するユーザーに適切です。また、ジョブデータを管理したり、Automation controller から他のデータコレクターにジョブデータを送信したりするサービスアカウントにも適切でしょう。
Member - Team	その他のすべてのユーザー	デフォルトでは、組織メンバーとしてのこれらのユーザーは、組織のどの機能にもアクセスできません。これらのユーザーにアクセスを許可するには、それぞれの組織の所有者がそのユーザーをそれぞれのチームに追加し、組織のプロジェクト、インベントリー、およびジョブテンプレートの各コンポーネントに対する管理者、実行、使用、更新、およびアドホックのアクセス許可を付与する必要があります。
Member - Team "Owner"	パワーユーザー - 主任開発者	組織の所有者は、チームインターフェイスを通じて、プロジェクト、インベントリー、ジョブテンプレートなどの組織のコンポーネントに対して admin ロールが割り当てられます。これらのユーザーは、アクセス権が与えられたそれぞれのコンポーネントを変更および使用できます。
Member - Team "Execute"	開発者 - エンジニア	これは最も一般的なロールであり、組織メンバーがジョブテンプレートを実行し、特定のコンポーネントに対する読み取りパーミッションを付与できるようになります。このパーミッションはテンプレートに適用されます。
Member - Team "Use"	開発者 - エンジニア	このパーミッションは、組織の信用情報、インベントリー、プロジェクトに適用されます。このパーミッションにより、ユーザーはジョブテンプレート内の各コンポーネントを使用できるようになります。
Member - Team "Update"	開発者 - エンジニア	このパーミッションはプロジェクトに適用されます。ユーザーがプロジェクトで SCM 更新を実行できるようにします。

32.3. ロールの機能: 編集および作成

組織のリソースロール機能は、ワークフローなど、特定のリソースタイプに固有です。このようなロールのメンバーになると、通常、2 種類のパーミッションが提供されます。ユーザーに組織デフォルトの workflow admin 管理者ロールが付与されている場合、ユーザーには次のパーミッションが与えられます。

- このユーザーは、組織 "Default" に新しいワークフローを作成できます。
- このユーザーは "Default" 組織の全ワークフローを編集できます。

例外として、ジョブテンプレートが挙げられます。ロールを所有していても作成パーミッションがあるわけではありません。詳細は、ジョブテンプレート を参照してください。

32.3.1. リソースロールおよび組織のメンバーシップロールの独立性

リソース固有の組織ロールは、管理者およびメンバーの組織ロールから独立しています。デフォルト組織のワークフロー管理者ロールを持っていても、ユーザーは組織内の全ユーザーを表示できませんが、デフォルト組織のメンバーロールを持っている場合は表示できます。2 種類のロールは、互いに独立して委任されます。

32.3.1.1. ジョブテンプレートの編集に必要なパーミッション

ユーザーは、ジョブテンプレート管理者ロールのみを使用して、ジョブの実行に影響を与えないフィールド (機密でないフィールド) を編集できます。ただし、ジョブテンプレートでジョブの実行に影響を与えるフィールドを編集するには、ユーザーは以下のロールを持っている必要があります。

- ジョブテンプレートとコンテナーグループの **admin** ロール
- 関連プロジェクトの **use** ロール
- 関連するインベントリーの **use** ロール
- 関連するインスタンスグループの **use** ロール

"organization job template admin" ロールが導入されましたが、このロールを付与するだけでは、プロジェクト、インベントリー、インスタンスグループの **use** ロールまたは、ジョブテンプレートが使用するコンテナーグループへの admin ロールがない場合に、組織内のジョブテンプレートを編集できません。

(組織内の) ジョブテンプレートの **完全な** 制御をユーザーまたはチームに委任するには、そのチームまたはユーザーに次の 3 つの組織レベルのロールをすべて付与する必要があります。

- job template administrator
- project administrator
- inventory administrator

上記を割り当てると、ユーザー (またはこれらのロールを持つチームのメンバーであるすべてのユーザー) が、組織内のジョブテンプレートを変更する完全なアクセス権が割り当てられます。ジョブテンプレートが別の組織からのインベントリーまたはプロジェクトを使用する場合、これらの組織ロールを持つユーザーは、そのジョブテンプレートの変更権限はありません。パーミッション管理を明確にするため、異なる組織のプロジェクトやインベントリーを混合しないでください。

32.3.1.2. RBAC 権限

各ロールにはコンテンツオブジェクトが必要です。たとえば、組織管理者のロールには組織のコンテンツオブジェクトがあります。ロールを委任するには、ユーザーのパスワードをリセットできる一部の例外を除き、コンテンツオブジェクトに対する管理者パーミッションが必要です。

Parent は組織です。

Allow は新規パーミッションで明示的に許可される内容です。

Scope は、この新しいロールが作成される親リソースです。例: **Organization.project_create_role**。

リソースの作成者には、そのリソースの管理者ロールが付与されていると想定されます。リソースを作成しても、リソースの管理が行われるとは限らないインスタンスを明示的に呼び出します。

各管理者のタイプに関連付けられたルールは次のとおりです。

Project Admin

- 許可: プロジェクトの作成、読み取り、更新、削除
- 範囲: 組織
- ユーザーインターフェース: **Project Add Screen - Organizations**

Inventory Admin

- 親: 組織管理者
- 許可: インベントリーの作成、読み取り、更新、削除
- 範囲: 組織
- ユーザーインターフェース: **Inventory Add Screen - Organizations**

注記

Use ロールと同様に、プロジェクト管理者ロールとインベントリー管理者ロールをユーザーに割り当てると、ユーザーは組織のジョブテンプレート (ワークフローではなく) を作成できるようになります。

Credential Admin

- 親: 組織管理者
- 許可: 共有されている認証情報の作成、読み取り、更新、削除
- 範囲: 組織
- ユーザーインターフェース: **Credential Add Screen - Organizations**

Notification Admin

- 親: 組織管理者
- 許可: 通知の割り当て
- 範囲: 組織

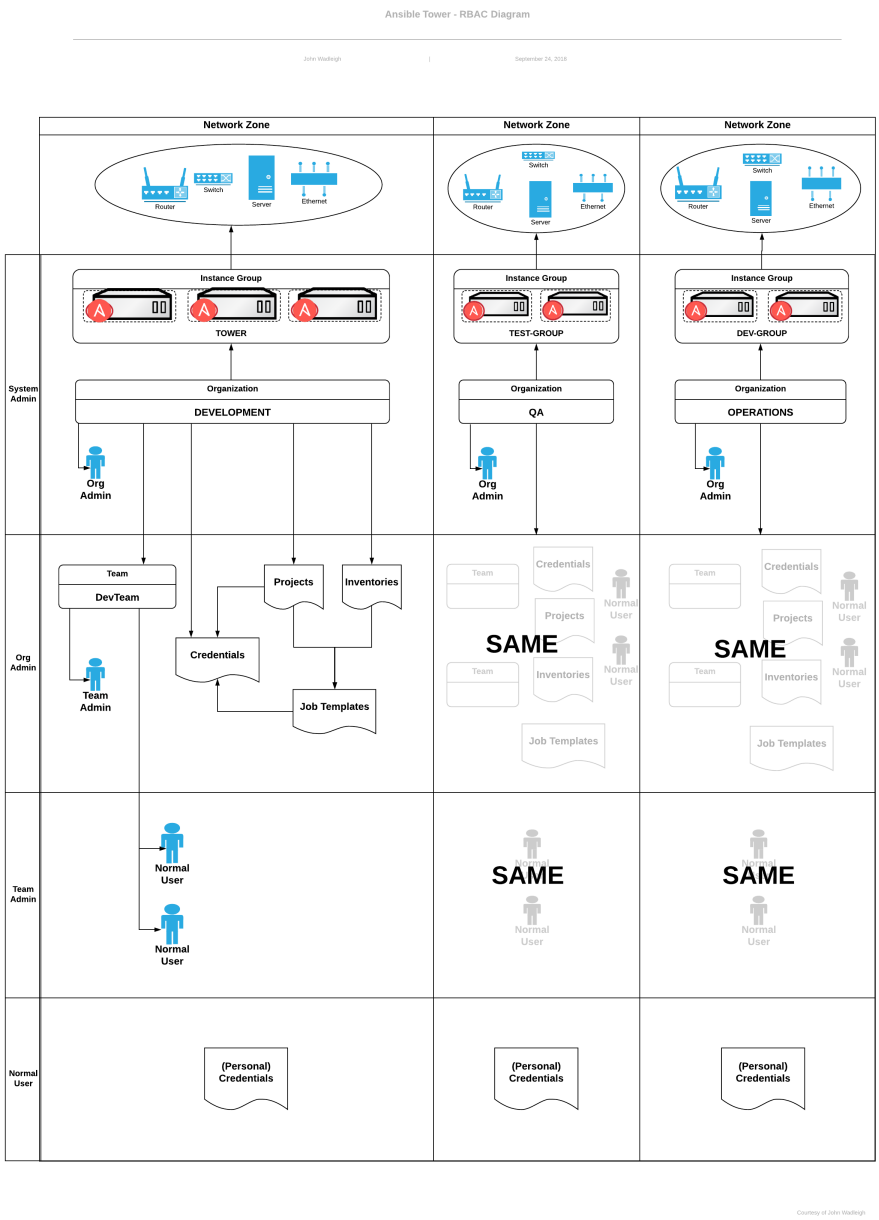
Workflow Admin

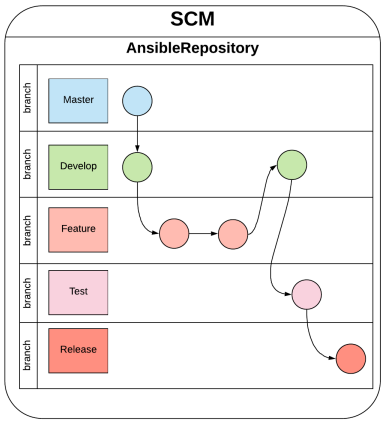
- 親: 組織管理者
- 許可: ワークフローの作成
- 範囲: 組織

Org Execute

- 親: 組織管理者
- 許可: ジョブテンプレートとワークフロージョブテンプレートの実行
- 範囲: 組織

以下は、組織とそのロール、それぞれがアクセスできるリソースを示すシナリオ例です。





第33章 用語集

アドホック
アドホック とは、オーケストレーション言語 (**/usr/bin/ansible-playbook**) ではなく、/usr/bin/ansible を使用して Ansible でクイックコマンドを実行することを指します。アドホックコマンドの例としては、インフラストラクチャー内の 50 台のマシンを再起動することが考えられます。アドホックで可能なことは、Playbook に記述して実現できます。Playbook では、他の多くの操作を結合することもできます。

Callback プラグイン
Ansible からの結果をインターセプトし、それに基づいて動作できるユーザー作成のコードを指します。GitHub プロジェクトのいくつかの例では、カスタムログを実行したり、メールを送信したり、効果音を再生したりしています。

コントロールグループ
cgroups としても知られるコントロールグループは、Linux カーネルの機能で、リソースをグループ化してプロセスを実行するために割り当てることができます。cgroups は、リソースをプロセスに割り当てるだけでなく、cgroup 内で実行されているすべてのプロセスによるリソースの使用状況をレポートすることもできます。

Check Mode (チェックモード)
--check オプションを使用して Ansible を実行することを指します。このオプションを使用すると、このフラグなしにこのコマンドを実行した場合に、加えられる可能性のある変更を出力するのみで、リモートシステムには変更が加えられません。これは、他のシステムのいわゆるドライランモードに似ています。ただし、これには予期しないコマンドの失敗やカスケード効果 (他のシステムの同様のモードにも当てはまります) は考慮されていません。チェックモードは、何が起こるかを把握するために使用しますが、適切なステージング環境の代わりにはなりません。

コンテナーグループ
コンテナーグループは、ジョブが実行される Kubernetes または OpenShift クラスターへの Pod のプロビジョニング設定を指定するインスタンスグループの一種です。これらの Pod はオンデマンドでプロビジョニングされ、Playbook の実行中のみ存在します。

Credentials
Automation controller がマシンに対してジョブを起動したり、インベントリーソースと同期したり、バージョン管理システムからプロジェクトコンテンツをインポートしたりするために使用できる認証の詳細です。詳細は、認証情報 を参照してください。

認証情報プラグイン
外部の認証情報タイプ、メタデータフィールド、シークレット管理システムとの対話に必要なコードの定義を含む Python コード

分散ジョブ
ジョブテンプレート、インベントリー、スライスサイズで設定されるジョブ。実行されると、分散されたジョブは、各インベントリーを複数のスライスサイズのチャンクにスライスし、これらをサイズの小さいジョブスライスの実行に使用します。

外部認証情報タイプ
シークレット管理システムとの認証に使用する管理認証情報タイプ

ファクト
ファクトとは、リモートノードに関して検出された情報です。これらは変数と同じように Playbook やテンプレートで使用できますが、ファクトは設定するのではなく推論されます。ファクトは、リモートノードで内部セットアップモジュールを実行することにより、プレイを実行するときに自動的に検出されます。セットアップモジュールを明示的に呼び出す必要はなく、ただ実行されるだけです。必要がな

い場合は、無効にして時間を節約できます。他の設定管理システムから切り替えるユーザーの利便性を考慮して、ファクトモジュールは、**ohai** ツールと **fater** ツールがインストールされている場合は、それぞれ Chef と Puppet のファクトライブラリーであるファクトも取り込みます。

Forks
Ansible と Automation controller はリモートノードと並行して通信します。並列処理のレベルは、ジョブテンプレートの作成または編集中に、**--forks** を渡すか、設定ファイルのデフォルトを編集するなど、いくつかの方法で設定できます。デフォルトは、非常に保守的な設定の 5 フォークですが、大量の RAM がある場合は、これを 50 などのより高い値に設定して並列処理を増やすことができます。

グループ
セットとして対応可能な Ansible 内の一連のホストを指します。それらの内の多くが単一インベントリーに存在する場合があります。

グループ変数
group_vars/ ファイルは、インベントリーファイルとともにディレクトリーに保存されるファイルで、オプションで各グループにちなんだファイル名が付けられます。このファイルは、特にデータ構造が複雑な場合などに、所定グループに提供される変数を配置できる便利な場所になります。これにより、これらの変数をインベントリーファイルまたは Playbook に埋め込む必要がなくなります。

ハンドラー
ハンドラーは Ansible Playbook の通常のタスクに似ています (**タスク** を参照) が、タスクに notify ディレクティブが含まれており、タスクで変更ありと示している場合にのみ実行されます。たとえば、設定ファイルが変更された場合、設定ファイルのテンプレート操作を参照するタスクがサービス再起動ハンドラーに通知する可能性があります。これは、サービスを再起動する必要がある場合にのみサービスをバウンスできることを意味します。ハンドラーはサービスの再起動以外にも使用できますが、最も一般的な用途はサービスの再起動です。

ホスト
Automation controller によって管理されるシステム。物理サーバー、仮想サーバー、クラウドベースのサーバー、またはその他のデバイス (通常はオペレーティングシステムインスタンス) が含まれる場合があります。ホストはインベントリーに含まれます。ノードと呼ばれることもあります。

ホスト指定子
Ansible の各プレイは、一連のタスク (システムのロール、目的、または順序を定義する) を一連のシステムにマップします。各プレイの host ディレクティブは、しばしばホスト指定子と呼ばれます。1 つのシステム、多数のシステム、1 つ以上のグループ、または 1 つのグループにあり明示的に別のグループにはない複数のホストを選択することができます。

インスタンスグループ
クラスター環境で使用するインスタンスを含むグループ。インスタンスグループは、ポリシーに基づいてインスタンスをグループ化する機能を提供します。

Inventory
ジョブの起動対象となるホストのコレクションです。

インベントリースクリプト
SQL データベース、CMDB ソリューション、LDAP などの外部リソースからホスト、ホストのグループメンバーシップ、および変数情報を検索するプログラム。この概念は Puppet (外部ノード分類子と呼ばれます) から採用されており、同様に機能します。

インベントリーソース
現在のインベントリーグループにマージする必要のあるクラウドまたは他のスクリプトについての情報のことです。グループ、ホストおよびグループやホストについての変数が自動的に設定されます。

ジョブ

Automation controller で起動される多くのバックグラウンドタスクの 1 つで、これは通常、Ansible Playbook の起動など、ジョブテンプレートのインスタンス化したものです。他の種類のジョブには、インベントリーのインポート、ソースコントロールからのプロジェクトの同期、管理上のクリーンアップアクションなどがあります。

ジョブの詳細
出力および成功/失敗を含む、特定ジョブの実行履歴のことです。

ジョブスライス
分散ジョブ を参照してください。

ジョブテンプレート
Ansible Playbook とその起動に必要なパラメーターのセットの組み合わせです。詳細は、ジョブテンプレート を参照してください。

JSON
JSON は、JavaScript オブジェクト構文に基づいて構造化データを表すためのテキストベースの形式です。Ansible と Automation controller は、リモートモジュールからの戻りデータに JSON を使用します。これにより、Python だけでなく任意の言語でモジュールを作成できるようになります。

Mesh
ノードで設定されるネットワークについて説明します。ノード間の通信は、TCP、UDP、Unix ソケットなどのプロトコルによってトランスポート層で確立されます。

ノード も参照してください。

メタデータ
認証後に外部システム内のシークレットを見つけるための情報。ユーザーは、外部認証情報をターゲット認証情報フィールドにリンクするときにこの情報を提供します。

ノード
ノードは、インスタンスデータベースモデルのエントリー、または **/api/v2/instances/** エンドポイントに対応し、クラスターまたはメッシュに参加するマシンです。統合ジョブ API は、**controller_node** と **execution_node** フィールドを報告します。実行ノードは、ジョブの実行先で、コントローラーノードはジョブとサーバー機能の間のインターフェイスとなります。

ノードタイプ	説明
Control	永続的なサービスを実行し、ジョブをハイブリッドノードと実行ノードに委任するノード。
Hybrid	永続的なサービスを実行し、ジョブを実行するノード。
Hop	メッシュ間の中継のみに使用されます。
Execution	コントロールノードから配信されたジョブを実行するノード (ユーザーの Ansible Automation から送信されたジョブ)

通知テンプレート
名前、説明および定義された設定を持つ通知タイプ (メール、Slack、Webhook など) のことを指します。

通知

メール、Slack、Webhook などの通知には、通知テンプレートで定義された名前、説明、設定があります。たとえば、ジョブが失敗すると、通知テンプレートで定義された設定を使用して通知が送信されます。

Notify (通知)
タスクが変更イベントを登録し、プレイの終了時に別のアクションを実行する必要があることをハンドラータスクに通知する行為。ハンドラーが複数のタスクから通知された場合でも、ハンドラーは 1 回だけ実行されます。ハンドラーは、通知された順序ではなく、リストされている順序で実行されます。

Organization
ユーザー、チーム、プロジェクト、インベントリーの論理的なコレクション。組織はオブジェクト階層の最上位レベルです。

Organization Administrator
組織内で新しいユーザーやプロジェクトを作成するなど、組織のメンバーシップと設定を変更する権限を持つユーザー。Organization Administsrator は、組織内の他のユーザーにパーミッションを付与することもできます。

パーミッション
プロジェクト、インベントリーや他のオブジェクトの読み取り、変更および管理を実行するためにユーザーおよびチームに割り当てられる権限のセットを指します。

プレイ
最小単位のプレイは、ホスト指定子で選択されるホストのセット (通常はグループで選択されますが、ホスト名 glob で選択されることもある) と、システムが実行するロールを定義するためにホストで実行されるタスク間のマッピングです。Playbook はプレイのリストです。Playbook には 1 つまたは複数のプレイが存在します。

Playbook
Ansible playbook。詳細は、Ansible Playbook を参照してください。

ポリシー
ポリシーは、インスタンスグループがどのように動作し、ジョブがどのように実行されるかを規定します。

プロジェクト
Automation Controller にある Ansible Playbook の論理的コレクションのことです。

ロール
ロールは、Ansible および Automation controller の組織単位です。ホストのグループ (または一連のグループ、ホストパターンなど) にロールを割り当てると、ホストが特定の動作を実装します。ロールには、変数値、タスク、ハンドラー、またはこれらの組み合わせの適用を含めることができます。ロールに関連付けられたファイル構造により、ロールは再配布可能な単位となり、Playbook 間または他のユーザーと動作を共有できるようになります。

シークレット管理システム
トークン、パスワード、証明書、暗号化キー、他の機密データをセキュアに保存して、これらのデータへのアクセスを制御するサーバーまたはサービス

スケジュール
ジョブが自動的に実行される日時のカレンダーのことを指します。

スライスされたジョブ
分散ジョブ を参照してください。

=== ソース認証情報

ターゲット認証情報のフィールドにリンクされる外部認証情報

Sudo
Ansible は root ログインを必要とせず、デーモンレスであるため、root レベルのデーモンも必要ありません (これは、機密性の高い環境ではセキュリティー上の問題となる可能性があります)。Ansible は、ログインして **sudo** コマンドでラップされた多くの操作を実行でき、パスワードなしの sudo とパスワードベースの sudo の両方で動作します。通常 **sudo** では機能しない一部の操作 (**scp** ファイル転送など) は、**sudo** モードでの実行中に Ansible の **copy**、**template**、および **fetch** モジュールを使用して実行できます。

スーパーユーザー
組織に関連付けられているかどうかに関係なく、システム内のオブジェクトを編集するパーミッションを持つサーバーの管理者。スーパーユーザーは組織や他のスーパーユーザーを作成できます。

Survey
ジョブテンプレート上で設定可能な、ジョブ起動時にジョブテンプレートによって尋ねられる質問のことを指します。

ターゲット認証情報
外部認証情報にリンクされている入力フィールドが含まれる外部以外の認証情報

チーム
ユーザー、プロジェクト、認証情報、パーミッションが関連付けられた組織の下位部門です。チームは、ロールベースのアクセス制御スキームを実装し、組織全体で責任を委譲する手段を提供します。

ユーザー
パーミッションおよび認証情報が関連付けられたオペレーターのことを指します。

Webhook
Webhook により、アプリケーション間の通信と情報共有が可能になります。これらは、SCM にプッシュされたコミットに応答し、ジョブテンプレートまたはワークフローテンプレートを起動するために使用されます。

ワークフロージョブテンプレート
単一ユニットとして実行するためにリンクされたジョブテンプレート、プロジェクトの同期、およびインベントリーの同期の任意の組み合わせで設定されるセットのことを指します。

YAML
設定ファイルの作成によく使用される人間が判読可能な言語。Ansible と Automation controller は、YAML を使用して、Playbook 設定言語と変数ファイルを定義します。YAML には最小限の構文があり、非常に簡潔で、一見して判断しやすくなっています。これは設定ファイルや人間にとって優れたデータ形式であるだけでなく、機械でも読み取り可能です。YAML は動的言語コミュニティーで人気があり、この形式には多くの言語でシリアル化に使用できるライブラリーが含まれています。たとえば、Python、Perl、または Ruby が含まれます。