Ansible on Clouds 2.4

GCP Marketplace で入手可能な Ansible Automation Platform のガイド

GCP Marketplace で入手可能な Ansible Automation Platform のインストールおよび設定

Last Updated: 2024-03-14

Ansible on Clouds 2.4 GCP Marketplace で入手可能な Ansible Automation Platform のガイド

GCP Marketplace で入手可能な Ansible Automation Platform のインストールおよび設定

法律上の通知

概要

GCP Marketplace で入手可能な Red Hat Ansible Automation Platform に関心をお寄せいただきありがとうございます。Ansible Automation Platform は、Ansible を装備した環境に、制御、ナレッジ、委譲の機能を追加して、チームが複雑かつ複数層のデプロイメントを管理できるように支援する商用サービスです。本ガイドは、GCP Marketplace で入手可能な Ansible Automation Platform のインストールおよび使用について説明します。

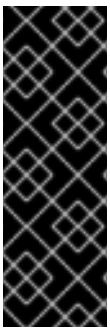
目次

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。



重要

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

第1章 概要

GCP Marketplace で入手可能な Ansible Automation Platform は、GCP Marketplace ポータルからデプロイできるサービスです。GCP Marketplace で入手可能な Ansible Automation Platform は、Ansible コンテンツコレクションのライブラリーにアクセスでき、主要な GCP サービスと統合されるため、インフラストラクチャーおよびアプリケーションのデプロイメント、設定、および管理の自動化をすぐに開始できます。

以下の Red Hat Ansible Automation Platform コンポーネントは、GCP Marketplace で入手可能な Ansible Automation Platform で提供されています。

- Automation Controller
- Ansible Automation Hub
- Private Automation Hub
- Ansible コンテンツコレクション
- 自動化実行環境
- Ansible コンテンツツール (Red Hat Ansible Automation Platform 用の Red Hat Insights へのアクセスなど)

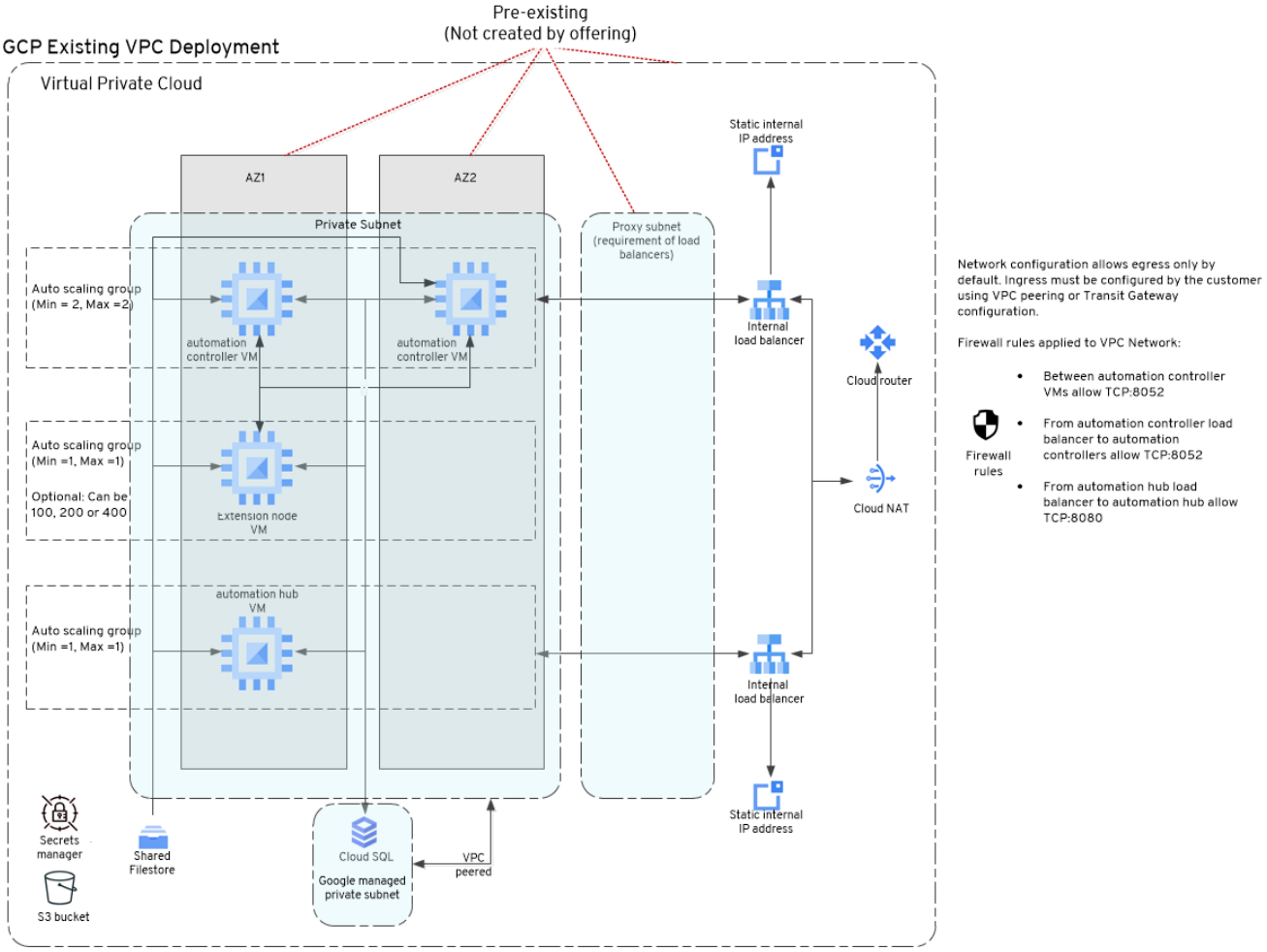
注記

現時点では、GCP Marketplace で入手可能な Ansible Automation Platform で自動化メッシュを利用できません。

1.1. アプリケーションのアーキテクチャー

GCP Marketplace で入手可能な Red Hat Ansible Automation Platform は、GCP アカウント内で実行されるインフラストラクチャーリソースにインストールされます。

この製品リストには 4 つの仮想マシンが含まれています。3 台の n2-standard-2 仮想マシンが、ソリューションの永続的なコンピューティングコンポーネントを設定します。さらに、一時的な e2-medium インスタンスが、Red Hat Ansible Automation Platform と Google Cloud デプロイメントワークロードの実行に、1 つ使用されます。この仮想マシンはデプロイメント中にのみ使用され、その後ソリューションから完全に削除されます。一時的な VM のインフラストラクチャーコストは、VM が存在する期間 (通常は 1 時間未満) の時間料金で請求されます。



GCP Marketplace の Ansible Automation Platform はプライベートになるように設計されており、デフォルトではパブリックアクセスは許可されていません。

これには、お客様が独自のネットワーク要件とセキュリティー慣行に従って、デプロイされた **内部ロードバランサー** (ILB) を自身で公開する必要があります。ILB を公開するのに考えられる方法には、VPC ピアリング、トランジットゲートウェイ、VPN、外部ロードバランサーなどが含まれます。

すべてのクラウドインフラストラクチャーコンポーネントは、**Virtual Private Cloud**(VPC) にデプロイされます。

お客様は、既存の VPC にデプロイするか、製品に新しい VPC をデプロイするかを選択できます。すべての仮想マシンインスタンスとクラウドインフラストラクチャーには、デフォルトでプライベート IP アドレス (割り当てはデプロイ時に指定した VPC とサブネットワークにより決定する) があります。

すべての内部トラフィックは、デプロイメント時に生成される自己署名証明書を使用して暗号化されます (外部トラフィックは、製品によってデプロイメントされる内部ロードバランサーに独自の証明書をデプロイメントすることによって暗号化することもできます)。

Ansible Automation Platform ソフトウェアは、デプロイされた仮想マシンインスタンス上でコンテナーとして実行します。

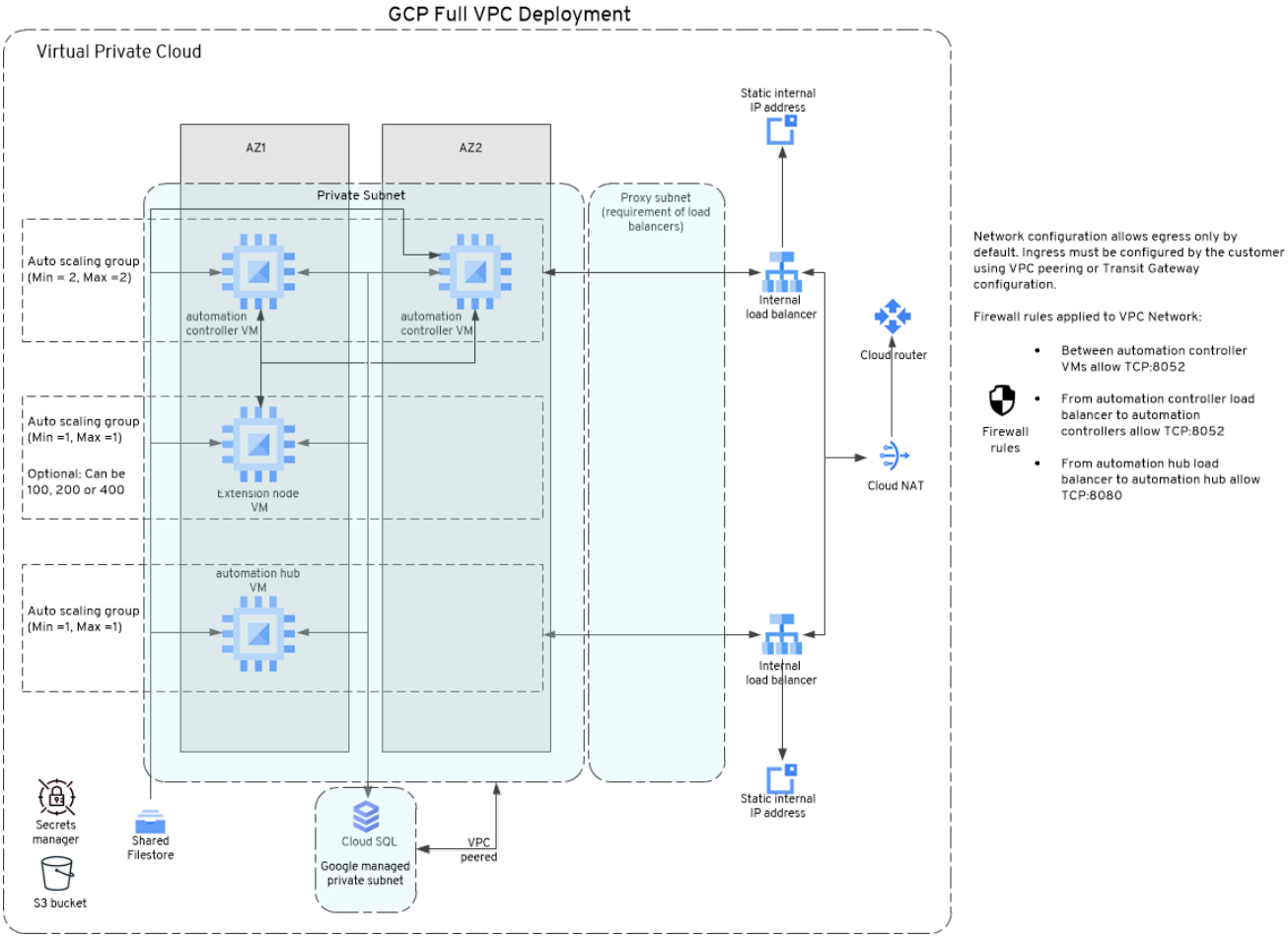
マネージドインスタンスグループ (MIG) は、仮想マシンインスタンスのライフサイクルを管理し、仮想マシンインスタンス上で実行している各サービスの健全性を監視します。健全性チェックが応答しない場合は、仮想マシンインスタンスを自動的にサイクルダウンし、新しい仮想マシンインスタンスに置き換えることで、Ansible Automation Platformサービスが稼働し続け、リクエストを処理できるようにします。

仮想マシンインスタンスは、カスタマイズされた **RedHat Enterprise Linux**(RHEL) Google Cloud

Machine Image をベースイメージとして実行します。この Google Cloud Machine Image には、Ansible Automation Platform (Automation Hub、Automation Controller、および Execution Node コンポーネント) を実行するために必要なすべてのコンテナーイメージとパッケージが事前に読み込まれています。

共有 Google File Store (GFS) ボリュームは、製品によりプロビジョニングされた各仮想マシンインスタンスにマウントされ、共通のファイルとリソースへの共有アクセスに使用されます。

Google Cloud SQL Service は、デプロイメント時に製品によりプロビジョニングされ、Automation Controller と Automation Hub の両方のデータベースが含まれます。



Foundation 製品には、Automation Controller コンポーネントと同じ仮想マシンインスタンス上で実行している 2 つの Execution Nodes が含まれています (これは、Ansible Automation Platform では Hybrid Node 設定と呼ばれます)。追加の Execution Node オファリングを購入して、自動化するためにライセンスが付与されている Ansible Automation Platform デプロイメントのスケール (マネージドノードの総数) を増やすことができます。Execution Node オファリングを既存の Ansible Automation Platform Foundation デプロイメントにデプロイする場合は、追加の Execution Node の仮想マシンインスタンスをデプロイして、Foundation デプロイメントの Automation Controller に自動的に接続し、自動化タスクの処理をすぐに開始できます。

Ansible Automation Platform コンポーネントは、仮想マシンインスタンス上で Podman コンテナーランタイムを使用してコンテナーとして実行します。Podman ランタイム設定は、稼働時間と可用性を確保するために **systemd** を使用するシステムサービスとして管理され、失敗したコンテナーは自動的に再起動します。

SELinux は仮想マシンインスタンスで有効になっており、コンテナーレベルまでサポートされています。

追加の運用自動化はこのオファリングにより提供され、registry.redhat.io からダウンロードできる個別の Docker コンテナーとして利用できます。これらの追加の運用自動化には、バックアップ、復元、アップグレードが含まれます。

RHEL OS のベースイメージ、Ansible Automation Platform コンテナー、または同梱されるパッケージで見つかった **Common Vulnerabilities and Exposures** (CVE) は、Ansible Automation Platform オファリングのアップグレード中に、ベースの RHEL Google Cloud Machine Image を、必要なすべてのソフトウェア、パッケージ、コンテナーを含む新しいバージョンに交換することで対応されます。

これは、付属のアップグレード自動化機能を使用して自動的に行われます。

お客様は、これらの運用自動化を利用して、独自の企業標準内で Ansible Automation Platform の運用準備を簡素化することができ、Ansible Automation Platform を管理するための自動化開発に時間を費やすのではなく、独自のインフラストラクチャーとアプリケーションを管理するための Ansible Automation の開発に集中できるようになります。

仮想マシンを再起動することはできません。代わりに、仮想マシンを置き換える必要があります。仮想マシンを置き換えると設定が失われるため、仮想マシン上で設定を行うことはできません。これはアップグレード時に常に発生します。

1.1.1. サービスの説明

サービス名	説明
Compute Engine	GCP 仮想マシンコンピュートプラットフォーム
Cloud SQL	GCP データベースサービス
Filestore	GCP ファイルストレージサービス
Virtual Private Cloud (VPC)	GCP ネットワーキングサービス
クラウドモニタリング	GCP メトリクスコレクター
クラウドロギング	GCP ログ管理サービス

第2章 インストール

2.1. 前提条件

Ansible Automation Platform をインストールして登録する前に、サービスの動作方法、データの保存方法、およびこれらのサービスを使用することによるプライバシーへの影響など、GCP についてよく理解しておく必要があります。また、**Google Cloud Platform** (GCP) でアカウントをセットアップする必要があります。

Google Cloud Platform の以下の点について実用的な知識を備えていることを前提としています。

- GCP Marketplace からのソリューションのデプロイ
- コンピューティングエンジンの **仮想マシン** (VM) インスタンス
- Cloud SQL for PostgreSQL
- Filestore
- GCP **Virtual Private Cloud**(VPC)
  - サブネット
  - ルートテーブル
  - ロードバランサー
- ネットワーク設計
  - ハブアンドスポークネットワーク設計
  - VPC ピアリング
  - CIDR (**Class Inter-Domain Routing**) ブロック
  - トランジットルーティング
- GCP クラウドモニタリング
  - GCP オペレーションエージェント
- SSH

Google Cloud Platform と用語の詳細は、GCP 製品ドキュメント を参照してください。

2.2. プロジェクトの作成

Ansible Automation Platform をインストールするには、アプリケーションをホストするプロジェクトを Google Cloud Platform アカウントに作成する必要があります (まだ作成していない場合)。GCP ドキュメントの プロジェクトの作成と管理 を参照してください。

2.2.1. 必要な API

Google Cloud Platform (GCP) プロジェクトでは、Ansible Automation Platform のインストールを完了するために、複数の API サービスに対するアクセス権が必要です。マーケットプレイスのデプロイ時に、プロセスは自動的に次の API を有効にしようとします。

必要に応じて、事前に API を有効にして、組織で許可されていることを確認できます。

API サービス	コンソールサービス名
Compute Engine API	**compute.googleapis.com**
Google Cloud API	**cloudapis.googleapis.com**
Identity and Access Management (IAM) API	**iam.googleapis.com**
Cloud SQL Admin API	**sql-component.googleapis.com**
Cloud Logging API	**logging.googleapis.com** モニタリングとロギング を参照してください
クラウドモニタリング API	**monitoring.googleapis.com** モニタリングとロギング を参照してください
Cloud Resource Manager API	**cloudresourcemanager.googleapis.com**
Cloud Identity-Aware Proxy API	**iap.googleapis.com**
Secret Manager API	**secretmanager.googleapis.com**
Service Networking API	**servicenetworking.googleapis.com**
Service Usage API	**serviceusage.googleapis.com**
OS Config API	**osconfig.googleapis.com**
Cloud Runtime Configuration API	**runtimeconfig.googleapis.com**
Cloud Filestore API	**file.googleapis.com**

2.2.2. サービスアカウントの作成

GCP Marketplace で入手可能な Ansible Automation Platform を設定するには、サービスアカウントが必要です。このアカウントは、アプリケーションのインストールと設定に使用され、Ansible Automation Platform 仮想マシンに関連付けられたままになります。必要なロールを持つ既存のサービスアカウントを使用するか、Ansible Automation Platform デプロイメントで必要なロールを持つアカウントを作成することができます。既存のアカウントを使用するか、事前に サービスアカウントを作成する 場合は、次のロールが必要です。

- エディター
- Logs Writer
- Cloud SQL Client

- Cloud SQL Instance User
- Secret Manager Secret Accessor
- コンピュートネットワーク管理者

必要なロールを既存のサービスアカウントに追加する手順は、単一のロールの付与 を参照してください。

Compute Network Aministrator ロールは、デプロイ時にアプリケーションを適切に設定するためにのみ必要です。インストールと設定が完了したら、このロールをサービスアカウントから削除できます。サービスアカウントは、Ansible Automation Platform 仮想マシンで設定されたままになります。

2.2.3. ポリシーおよび権限

Ansible Automation Platform デプロイメントおよび アプリケーションアーキテクチャー で説明されているリソースを正常に作成および管理するには、GCP アカウントに次の **Identity and Access Management** (IAM) パーミッションが必要です。

GCP Marketplace から Ansible Automation Platform をデプロイするには、GCP アカウントのライセンスも取得する必要があります。

IAM ポリシーでこれらのリソースのデプロイメントおよび管理が制限されている場合は、アプリケーションのデプロイに失敗する可能性があります。

アプリケーションには 2 つのデプロイメントオプションがあります。

- 新しい VPC を使用してデプロイメントを作成します。
- 既存の VPC を使用してデプロイメントを作成します。

どちらのオプションでも同じ最小限のパーミッションが必要です

> Cloud SQL Client
> Cloud SQL Instance User
> Compute Network Admin
> Editor
> Logs Writer
> Secret Manager Secret Accessor

2.3. アプリケーションのデプロイメント

Google Cloud Console でオファーを開始するには、マーケットプレイスに移動し、**Red Hat Ansible Automation Platform 2 - 最大 100 個の管理対象ノード** を検索します。このオファーを選択した後、**Launch** をクリックします。

この製品リストには 4 つの仮想マシンが含まれています。3 台の n2-standard-2 仮想マシンが、ソリューションの永続的なコンピューティングコンポーネントを設定します。さらに、一時的な e2-medium インスタンスが、Red Hat Ansible Automation Platform と Google Cloud デプロイメントワークロードの実行に、1 つ使用されます。この仮想マシンはデプロイメント中にのみ使用され、その後ソリューションから完全に削除されます。一時的な VM のインフラストラクチャーコストは、VM が存在する期間 (通常は 1 時間未満) の時間料金で請求されます。

重要

デプロイメント名の長さには、一時的ではありますが必要な制約が課されています。これは、Ansible Automation Platform デプロイメントを設定する内部コンポーネントの GCP の命名スキームによるものです。この命名スキームに基づくコンポーネント名は長くなりすぎ、他のサービスによって課せられた命名の制約に違反することが多く、デプロイメントエラーが発生します。

デプロイ名の長さと GCP プロジェクト名の長さを合わせた長さが 35 文字未満で、デプロイ名の長さが 30 文字未満である必要があります。

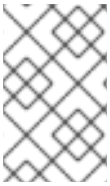
以下の計算は、プロジェクト内の Ansible Automation Platform デプロイメントの名前の最大長を見つけるのに役立ちます。

デプロイ名の長さ < = (最小 30~35) - 長さ (gcp プロジェクト名)

アプリケーションをデプロイするには 2 つの方法があります。

2.3.1. 新しい VPC を使用したアプリケーションのデプロイ

この手順では、新しい VPC ネットワークを作成し、作成した VPC にアプリケーションをデプロイします。



注記

新しい VPC を使用してアプリケーションをデプロイするプロセスは非推奨となり、この機能は今後のリリースで GCP Marketplace の Ansible Automation Platform から削除される予定です。

手順

1. Deployment ページで、**Confirm Service Usage API is enabled**チェックボックスの下にある **Service Usage API** リンクを選択します。
2. **API/Service Details** タブで、API が有効になっていることを確認してから、Deployment ページに戻ります。
3. **Confirm Service Usage API**のチェックボックスをオンにします。
4. **Service Account** を選択または作成します。詳細は、サービスアカウント を参照してください。
5. **Region** フィールドで、アプリケーションをデプロイするリージョンを選択します。
6. **Zone** フィールドで、Filestore をデプロイするゾーンを選択します。ゾーンは、選択した Region 内に存在する必要があります。
7. **Observability** セクションで、ロギングとメトリクスを Cloud Logging と Cloud Monitoring に送信できるようにします。これらのサービスを有効にする場合の金銭的なコストについては、Operations Suite の価格 を参照してください。この機能の設定に関する詳細は、モニタリングとロギング を参照してください。
8. **Network Selection** セクションで、**New network** を選択します。Networking セクションには、デプロイメントで使用されるすべてのネットワーク範囲のデフォルトが用意されています。これらの値を変更する場合は、ネットワークオプション を参照してください。

9. オプション: **Additional Labels** セクションに、デプロイの一部である GCP リソースに追加する追加のラベルのキーと値のペアを指定します。有効なキーと値は GCP ラベルの要件を満たしている必要があります。**キー** には、ハイフン、アンダースコア、小文字、数字のみが許可されます。**キー** は小文字で始まる必要があります。**値** には、ハイフン、アンダースコア、小文字、数字のみが許可されます。

10. **DEPLOY** をクリックします。

11. Deployment Manager に、実行中のデプロイメントが表示されます。

12. アプリケーションはプロビジョニングを開始します。インフラストラクチャーとアプリケーションが完全にプロビジョニングされるまで、しばらく時間がかかる場合があります。

注記

デプロイメントに関する警告が表示されます。

> This deployment has resources from the Runtime Configurator service, which is in Beta

この警告は想定どおりで、特に問題はありません。

デプロイメント後にネットワーク範囲を変更する場合は、現在のデプロイメントを削除してから、既存の VPC を使用したアプリケーションのデプロイ の手順に従ってください。

2.3.2. 既存の VPC を使用したアプリケーションのデプロイ

次の手順では、既存の VPC ネットワークを使用してアプリケーションをデプロイします。

手順

1. **Deployment** ページで、**Confirm Service Usage API is enabled**チェックボックスの下にある **Service Usage API** リンクを選択します。

2. **API/Service Details** タブで、API が有効になっていることを確認してから、**Deployment** ページに戻ります。

3. **Confirm Service Usage API**のチェックボックスをオンにします。

4. **Service Account** を選択または作成します。詳細は、サービスアカウント を参照してください。

5. **Region** フィールドで、アプリケーションをデプロイするリージョンを選択します。

6. **Zone** フィールドで、Filestore をデプロイするゾーンを選択します。ゾーンは、選択した Region 内に存在する必要があります。

7. **Observability** セクションで、ロギングとメトリクスを Cloud Logging と Cloud Monitoring に送信できるようにします。これらのサービスを有効にする場合の金銭的なコストについては、Operations Suite の価格 を参照してください。この機能の設定に関する詳細は、モニタリングとロギング を参照してください。

8. **Network Selection** セクションで、**Existing network** を選択します。

9. **Existing Network** セクションで、既存の VPC ネットワーク名、既存のサブネット名、および既存のプロキシーサブネット名を指定します。

注記

既存のプロキシーサブネットは、ロードバランシング用に予約されたプロキシー専用サブネットである **リージョン管理プロキシー** タイプである必要があります。

VPC ネットワーク内に **NAT ルーター** を作成するには、**クラウド NAT ルーター** を選択します。

10. **Networking** セクションには、デプロイメントで使用されるすべてのネットワーク範囲のデフォルトが用意されています。既存のネットワーク設定に基づいてこれらの値を指定します。これらの値を変更する場合は、ネットワークオプション を参照してください。

11. オプション: **Additional Labels** セクションに、デプロイの一部である GCP リソースに追加する追加のラベルのキーと値のペアを指定します。有効なキーと値は GCP ラベルの要件を満たしている必要があります。**キー** には、ハイフン、アンダースコア、小文字、数字のみが許可されます。**キー** は小文字で始まる必要があります。**値** には、ハイフン、アンダースコア、小文字、数字のみが許可されます。

12. **DEPLOY** をクリックします。

13. **Deployment Manager** に、実行中のデプロイメントが表示されます。

14. アプリケーションはプロビジョニングを開始します。インフラストラクチャーとアプリケーションが完全にプロビジョニングされるまで、しばらく時間がかかる場合があります。

注記

デプロイメントに関する警告が表示されます。

> This deployment has resources from the Runtime Configurator service, which is in Beta.

この警告は想定どおりで、特に問題はありません。

2.4. デプロイメント情報

Ansible Automation Platform をデプロイしたら、次の手順に従ってデプロイメントに関する情報を取得します。

2.4.1. 管理パスワードの取得

管理パスワードを取得するには、以下の手順を使用します。

手順

1. GCP UI で、メインメニューを選択します。
2. **Security** を選択します。**Security** が表示されない場合は、**View All Products** を選択します。
3. **Secret Manager** を選択します。

4. デプロイの名前でフィルター処理します。シークレット名の形式は、**<DeploymentName>-aap-admin** です。
5. デプロイのシークレット名をクリックします。
6. デプロイメントの行にある **More Actions** アイコン ⋮ をクリックします。
7. **View secret value** を選択します。管理パスワードが表示されます。

2.4.2. ロードバランサーアドレスの取得

Controller および Hub の IP アドレスを取得するには、次の手順を使用します。

手順

1. GCP UI で、メインメニューを選択します。
2. **Deployment Manager** を選択します。
3. **Deployments** を選択します。
4. デプロイメント名を選択します。
5. **View Details** を選択します。
6. 右ペインの **Deployment properties** で、**Layout** 行を見つけます。
7. **View** を選択します。**Outputs:** セクションで、**name** が controllerIp および hubIp である箇所の **finalValue** を確認します。

2.5. デプロイメント時のモニタリングとロギングのセットアップ

手順

1. GCP UI で、**Observability** に移動します。
2. **Connect Logging** および **Connect Metrics** のチェックボックスをオンにします。

注記

これらのチェックボックスは、基盤デプロイメントでのみ使用できます。

2.6. 拡張ノードのデプロイ

パブリックオファーまたはプライベートオファーから拡張ノードを購入して起動したら、拡張ノードを設定します。

手順

1. **Deployment name** フィールドに、アプリケーションのデプロイメント で説明されているように、十分に短い名前を入力します。
2. **Service Account** には、最大 100 のマネージドノードのデプロイメントを備えた Red Hat Ansible Automation Platform で使用されるサービスアカウントを選択します。

3. **Region** フィールドで、アプリケーションをデプロイするリージョンを選択します。
4. **Zone** フィールドで、基本オファーをデプロイするときに選択した **Region** のゾーンを選択します。このフィールドは、ネットワークリストをフィルタリングするためにのみ使用されます。
5. **Main Deployment Name** フィールドに、拡張ノードをデプロイする基盤デプロイメント名を入力します。

注記

Main Deployment Name は必須フィールドです。

6. **Networking** セクションで、デフォルトのオプションを展開します。
7. **Network** フィールドで、**-aap-net** で終わる既存の基盤ネットワークを選択します。
8. **Subnetwork** フィールドで、**-aap-subnet** で終わる既存の基盤サブネットワークを選択します。

重要

-aap-proxy-subnet で終わるサブネットは選択しないでください。

9. **Extend Ansible Automation Platform deployment in selected VPC** がチェックされていることを確認します。
10. **DEPLOY** をクリックします。
11. Deployment Manager に、実行中のデプロイメントが表示されます。

拡張ノードのプロビジョニングが開始されます。

インフラストラクチャーと拡張ノードが完全にプロビジョニングされるまで、しばらく時間がかかる場合があります。

第3章 拡張ノードのデプロイ

デフォルトでは、基盤デプロイメントは 2 つの Automation Controller ノードと 1 つの Automation Hub ノードでデプロイされます。実行ノードを追加して、GCP Marketplace から Ansible Automation Platform をスケールアウトできます。拡張ノードのオファーを組み合わせることで、Ansible Automation Platform デプロイメントをスケールアップおよびスケールアウトできます。

新しい拡張ノードをデプロイする前に、Ansible Automation Platform デプロイメントのバックアップを作成する必要があります。

この手順には、次のステップが含まれます。

1. 拡張ノードのオファータイプを決定します。
2. 拡張ノードを管理するための最小限の権限が満たされていることを確認します。
3. **ansible-on-clouds-ops** コンテナーイメージをプルします。
4. **ansible-on-clouds-ops** コンテナーを実行してデータファイルを生成します。
5. データファイルにデータを入力します。
6. **ansible-on-clouds-ops** コンテナーを実行して拡張ノードをデプロイします。

前提条件

- Linux または macOS システム (ansible-on-clouds-ops コンテナーイメージが実行される場所)
- Docker

3.1. オファーの種類の決定

以下の表は、オファーの種類と対応する GCP インスタンスタイプを一覧表示しています。ワークロードのニーズに応じて、拡張ノードにより適したオファーの種類を選択できます。

オファーのタイプ (ノード)	GCP インスタンスの種類
100	n2-standard-2
200	n2-standard-4
400	n2-standard-8

拡張ノードのオファーを組み合わせることで、Ansible Automation Platform デプロイメントをスケールアップできます。

3.2. IAM の最小権限

Ansible Automation Platform で拡張ノードを正常に作成および管理するには、GCP アカウントに次の **Identity and Access Management**(IAM) パーミッションが必要です。

GCP Marketplace から Ansible Automation Platform の拡張ノードオファリングをデプロイするには、GCP アカウントのライセンスも取得する必要があります。

```
Minimum Permissions -

Cloud SQL Client
Cloud SQL Instance User
Editor
Logs Writer
Secret Manager Secret Accessor
IAP-secured Tunnel User
```

3.3. ANSIBLE-ON-CLOUDS-OPS コンテナーイメージのプル

デプロイ対象のバージョンと同じタグを持つ Ansible on Clouds 運用コンテナーの Docker イメージをプルします。デプロイしたバージョンが不明な場合は、Ansible on Clouds デプロイメントの現在のバージョンを確認する方法の詳細について、コマンドジェネレーター と Playbook **gcp_get_aoc_version** を参照してください。

注記

Docker イメージをプルする前に、Docker を使用して registry.redhat.io にログインしていることを確認してください。以下のコマンドを使用して registry.redhat.io にログインします。

```
$ docker login registry.redhat.io
```

レジストリーのログインに関する詳細は、Registry Authentication を参照してください。

たとえば、基盤デプロイメントのバージョンが 2.4.20240215-00 の場合、拡張ノードを基盤デプロイメントにデプロイするには、タグが 2.4.20240215 の運用イメージをプルする必要があります。

以下のコマンドを使用します。

```
$ export IMAGE=registry.redhat.io/ansible-on-clouds/ansible-on-clouds-ops-rhel9:2.4.20240215
$ docker pull $IMAGE --platform=linux/amd64
```

注記

基盤のデプロイメントバージョンが 2.4.20240215-00 ではない場合は、Released versions ページの表で、**Ansible on Clouds バージョン** 列にある一致するデプロイメントバージョンを参照してください。**Ansible-on-clouds-ops コンテナーイメージ列の、IMAGE 環境変数の対応する運用イメージ** を見つけます。

3.4. ANSIBLE-ON-CLOUDS-OPS コンテナーを実行してデータファイルを生成する

次のコマンドで、必要なデータファイルを生成します。これらのコマンドは、ディレクトリーと空のデータテンプレートを作成します。このテンプレートは、データを入力して拡張ノードのデプロイメント時に使用するためのものです。

手順

1. 設定ファイルを保存するフォルダーを作成します。

```
$ mkdir command_generator_data
```

2. **command_generator_data** フォルダーに設定ファイルのテンプレートを追加します。

```
$ docker run --rm -v $(pwd)/command_generator_data:/data $IMAGE \
command_generator_vars gcp_add_extension_nodes \
--output-data-file /data/extra_vars.yml
```

3. これらのコマンドは、**command_generator_data/extra_vars.yml** テンプレートファイルを作成します。このテンプレートファイルは以下のようになります。

```
gcp_add_extension_nodes:
  cloud_credentials_path:
  deployment_name:
  extra_vars:
    gcp_compute_region:
    gcp_extension_node_subscription:
    gcp_instance_group_name:
    gcp_instance_template_name:
    gcp_offer_type:
```

3.5. データファイルにデータを取り込む

操作をトリガーする前に、データファイルを設定する必要があります。データファイルにリストされている変数は、次のように定義されます。

- **cloud_credentials_path** は、Google Cloud サービスアカウントの認証情報ファイルのパスです。これは、絶対パス名である必要があります。
- **deployment_name** は、拡張ノードを作成する AAP デプロイメントマネージャーデプロイメントの名前です。
- **gcp_instance_group_name** は、拡張ノード用に作成する GCP インスタンスグループの名前です。
- **gcp_instance_template_name** は、作成する GCP インスタンステンプレートの名前です。
- **gcp_offer_type** は、拡張ノードのオファーの種類です。これは 100、200、または 400 である必要があります。
- **gcp_compute_region** は、基盤デプロイメントがデプロイされる GCP リージョンです。これは、**Deployment Manager** で Deployments 設定を確認することで取得できます。
- **gcp_extension_node_subscription** は、拡張ノードのサブスクリプションが購入されているかどうかを確認するためのフラグです。**true** または **false** である必要があります。

3.6. 拡張ノードのデプロイ

手順

1. 拡張ノードをデプロイするために、コマンドジェネレーターを実行して CLI コマンドを生成します。

```
$ docker run --rm -v $(pwd)/command_generator_data:/data $IMAGE command_generator --
data-file /data/extra_vars.yml
```

次のコマンドが生成されます。

```
---------------------------------------------
Command to run playbook:

docker run --rm --env PLATFORM=GCP -v </path/to/gcp/service-
account.json>:/home/runner/.gcp/credentials:ro /
--env ANSIBLE_CONFIG=../gcp-ansible.cfg --env DEPLOYMENT_NAME=
<deployment_name> /
--env GENERATE_INVENTORY=true  $IMAGE
redhat.ansible_on_clouds.gcp_add_extension_nodes /
-e 'gcp_deployment_name=<deployment_name>
gcp_service_account_credentials_json_path=/home/runner/.gcp/credentials
gcp_compute_region=<region> gcp_instance_template_name=<instance_template_name> /
gcp_instance_group_name=<instance_group_name> gcp_offer_type=100
gcp_extension_node_subscription=True'
=============================================
```

2. 生成されたコマンドを実行して拡張ノードを追加します。

```
$ docker run --rm --env PLATFORM=GCP -v </path/to/gcp/service-
account.json>:/home/runner/.gcp/credentials:ro /
--env ANSIBLE_CONFIG=../gcp-ansible.cfg --env DEPLOYMENT_NAME=leena1 /
--env GENERATE_INVENTORY=true  $IMAGE
redhat.ansible_on_clouds.gcp_add_extension_nodes /
-e 'gcp_deployment_name=<deployment_name>
gcp_service_account_credentials_json_path=/home/runner/.gcp/credentials
gcp_compute_region=<region> gcp_instance_template_name=<instance_template_name> /
gcp_instance_group_name=<instance_group_name> gcp_offer_type=100
gcp_extension_node_subscription=True'
```

3. Playbook の実行が完了すると、出力は次のようになります。

```
TASK [redhat.ansible_on_clouds.standalone_gcp_add_extension_nodes :
[deploy_extension_nodes] Extension node created] ***
ok: [localhost] => {
    "msg": "Extension node is created for deployment test-ext1."
}

PLAY RECAP *********************************************************************
localhost                  : ok=39   changed=5    unreachable=0    failed=0    skipped=6
rescued=0    ignored=0
```

第4章 拡張ノードの削除

拡張ノードを削除する前に、Ansible Automation Platform デプロイメントのバックアップを作成する必要があります。

以下の手順に従って、AWS Marketplace 環境の Ansible Automation Platform から実行ノードを削除します。

前提条件

- Linux または MacOS システム (**ansible-on-clouds-ops** コンテナーイメージが実行される場所)
- Docker

手順

1. **ansible-on-clouds-ops** コンテナーイメージをプルします。
2. ansible-on-clouds-ops コンテナーを実行してデータファイルを生成します。
3. データファイルを更新します。
4. **ansible-on-clouds-ops** コンテナーを実行して、拡張ノードを削除します。

4.1. ANSIBLE-ON-CLOUDS-OPS コンテナーイメージのプル

基盤デプロイメントと同じタグを持つ Ansible on Clouds 運用コンテナーの Docker イメージをプルします。デプロイしたバージョンが不明な場合は、Ansible on Clouds デプロイメントの現在のバージョンを確認する方法の詳細について、コマンドジェネレーター と Playbook **gcp_get_aoc_version** を参照してください。

注記

Docker イメージをプルする前に、Docker を使用して registry.redhat.io にログインしていることを確認してください。以下のコマンドを使用して registry.redhat.io にログインします。

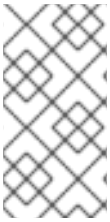
```
$ docker login registry.redhat.io
```

レジストリーのログインに関する詳細は、Registry Authentication を参照してください。

たとえば、基盤デプロイメントのバージョンが 2.4.20240215-00 の場合、拡張ノードを基盤デプロイメントにデプロイするには、タグが 2.4.20240215 の運用イメージをプルする必要があります。

以下のコマンドを使用します。

```
$ export IMAGE=registry.redhat.io/ansible-on-clouds/ansible-on-clouds-ops-rhel9:2.4.20240215
$ docker pull $IMAGE --platform=linux/amd64
```



注記

基盤のデプロイメントバージョンが 2.4.20240215-00 ではない場合は、Released versions ページの表で、**Ansible on Clouds バージョン** 列にある一致するデプロイメントバージョンを参照してください。**Ansible-on-clouds-ops コンテナーイメージ列の、IMAGE 環境変数の対応する運用イメージ** を見つけます。

4.2. ANSIBLE-ON-CLOUDS-OPS コンテナーを実行してデータファイルを生成する

次のコマンドで、必要なデータファイルを生成します。これらのコマンドは、ディレクトリーと空のデータテンプレートを作成します。このテンプレートは、データを入力してアップグレード時に使用するためのものです。

手順

1. 設定ファイルを保存するフォルダーを作成します。

```
$ mkdir command_generator_data
```

2. **$(pwd)/command_generator_data** フォルダーに設定ファイルのテンプレートを追加します。

```
$ docker run --rm -v $(pwd)/command_generator_data:/data $IMAGE \
command_generator_vars gcp_remove_extension_nodes \
--output-data-file /data/extra_vars.yml
```

これらのコマンドを実行すると、**command_generator_data/extra_vars.yml** テンプレートファイルが作成されます。このテンプレートファイルは以下のようになります。

```
gcp_add_extension_nodes:
  cloud_credentials_path:
  deployment_name:
  extra_vars:
    gcp_compute_region:
    gcp_instance_group_name:
    gcp_instance_template_name:
```

4.3. データファイルにデータを取り込む

操作をトリガーする前に、データファイルを設定する必要があります。データファイルにリストされている変数は、次のように定義されます。

- **cloud_credentials_path** は、Google Cloud サービスアカウントの認証情報ファイルのパスです。これは、絶対パス名である必要があります。
- **deployment_name** は、拡張ノードを作成するデプロイメントの名前です。
- **gcp_instance_group_name** は、拡張ノード用に作成する GCP インスタンスグループの名前です。
- **gcp_instance_template_name** は、作成する GCP インスタンステンプレートの名前です。

- **gcp_compute_region** は、基盤デプロイメントがデプロイされる GCP リージョンです。これは、Deployment Manager の Deployments 設定を確認することで取得できます。

4.4. ANSIBLE-ON-CLOUDS-OPS コンテナーを実行して拡張ノードを削除する

手順

1. 拡張ノードのセットを削除するために、コマンドジェネレーターを実行して CLI コマンドを生成します。

```
$ docker run --rm -v $(pwd)/command_generator_data:/data $IMAGE command_generator --data-file /data/extra_vars.yml
```

コマンドジェネレーターの出力には次のコマンドが含まれます。

```
d-----------------------------------------------
Command to run playbook:

docker run --rm --env PLATFORM=GCP -v </path/to/gcp/service-account.json>:/home/runner/.gcp/credentials:ro \
--env ANSIBLE_CONFIG=../gcp-ansible.cfg --env DEPLOYMENT_NAME=<deployment_name> \
--env GENERATE_INVENTORY=true $IMAGE redhat.ansible_on_clouds.gcp_remove_extension_nodes \
-e 'gcp_deployment_name=<deployment_name> gcp_service_account_credentials_json_path=/home/runner/.gcp/credentials gcp_compute_region=<region> gcp_instance_template_name=<instance_template_name> \
gcp_instance_group_name=<instance_group_name>'
===============================================
```

2. 提供されたコマンドを実行して、一連の拡張ノードを削除します。

```
$ docker run --rm --env PLATFORM=GCP -v </path/to/gcp/service-account.json>:/home/runner/.gcp/credentials:ro \
--env ANSIBLE_CONFIG=../gcp-ansible.cfg --env DEPLOYMENT_NAME=<deployment_name> \
--env GENERATE_INVENTORY=true  $IMAGE redhat.ansible_on_clouds.gcp_remove_extension_nodes \
-e 'gcp_deployment_name=<deployment_name> gcp_service_account_credentials_json_path=/home/runner/.gcp/credentials gcp_compute_region=<region> gcp_instance_template_name=<instance_template_name> \
gcp_instance_group_name=<instance_group_name>'
```

エラーがなければ、拡張ノードは正常に削除されます。

第5章 ネットワーキングおよびアプリケーションアクセス

GCP Marketplace で入手可能な Ansible Automation Platform をデプロイすると、分離された VPC にインストールされ、アクセスできなくなります。以下の手順では、GCP Marketplace で入手可能な Ansible Automation Platform で使用される VPC を既存の GCP ネットワークに接続する方法について説明します。接続時に、Ansible Automation Platform に接続する方法を決定する必要があります。

プライベートネットワークにアクセスできるように VPN、Google Cloud Interconnect、bastion サーバーなど、この接続を有効にする方法は多数あります。ロードバランサーなどの GCP サービスを使用して、パブリックインターネットアクセスのあるプラットフォームを公開することもできます。

組織が GCP 上でアプリケーションアクセスを設定する方法は、Red Hat のガイドラインと GCP Marketplace で入手可能な Ansible Automation Platform のサポート対象外です。詳細は、これらのトピックに関するガイドラインについて、仮想マシンへの安全な接続 を参照してください。

5.1. ネットワーキングオプション

Ansible Automation Platform from GCP Marketplace のネットワークトポロジーには、組織のネットワーク要件に合わせて変更できる設定可能なネットワークセグメントがいくつか含まれています。

デプロイメントにより、パブリックインターネットからアクセスできない、または既存の VPC ネットワークを使用してアクセスできない新しい VPC とサブネットが作成されます。ネットワークおよびアプリケーションアクセス で説明されているように、アプリケーションへのアクセスを提供する必要があります。

以下のネットワーク範囲を指定するときは、既存のネットワーク設定を考慮してください。各範囲がここで指定された他の範囲と重複しないこと、およびネットワーク内の既存の範囲と重複しないことを確認してください。各範囲は、プライベートネットワーククラス内に存在する必要があります。

アプリケーションサブネット範囲

オファリングによってデプロイされたカスタム VPC によって使用されるサブネット範囲を定義するネットワーク CIDR。最小 **/24** セグメントである必要があり、プライベートネットワーク範囲 (192.168 または 10.0) 内にある必要があります。

デフォルト: 192.168.240.0/24

Cloud SQL ピアリングネットワーク範囲

ネットワーク CIDR は、GCP CloudSQL ネットワークと、オファリングによってデプロイされたアプリケーションサブネットをピアリングするために使用されるネットワークセグメントを定義します。**/24** セグメントでなければなりません。**/24** セグメント範囲は、GCP CloudSQL ネットワークピアリング設定の要件です。

デフォルト: 192.168.241.0/24

Filestore ピアリングネットワーク範囲

Ansible Automation Platform from GCP Marketplace は、GCP Filestore サービスを使用して、デプロイの一部としてプロビジョニングされた複数の Automation Controller と Automation Hub VM の間で設定ファイルを共有します。このネットワーク CIDR 範囲は、GCP Filestore ネットワークとオファリングのカスタム VPC baseNetwork の間でピアリングするためにオファリングによって使用されるピアネットワークを定義します。最小の **/29** セグメントである必要があります。

デフォルト: 192.168.243.0/29

ロードバランサープロキシーのサブネット範囲

GCP Marketplace で入手可能な Ansible Automation Platform は、GCP のネイティブクラウド機能を使用してデプロイされ、スケーラブルで信頼性の高いインストールを提供します。GCP Marketplace で入手可能な Ansible Automation Platform デプロイメントトポロジーの一部として、2 つのロードバランサーが Automation Hub とAutomation Controller VM の前にデプロイされます。すべてのトラフィックはこれらのロードバランサーに向けられ、使用可能なバックエンド VM にプロキシーされます。このデプロイでは、GCP のネイティブロードバランシングサポートを利用して、お客様がロードバランサーに追加のポート (https) を追加して、リクエストをキャプチャーしてバックエンド VM に転送できるようにします。これにより、信頼性を高めるためにリクエストバランシングとセッショントラッキングも提供されます。ロードバランサーのデプロイの一環として、GCP では、GCP がバックエンド VM へのリクエストのリダイレクトをネイティブに処理する特別なプロキシーネットワークを作成する必要があります。この特別なプロキシーネットワークは、GCP のロードバランサーのプロキシーネットワーク要件以外の目的で、GCP Marketplace で入手可能な Ansible Automation Platform 内で使用されることはありません。**/24** セグメントが必要です。

デフォルト: 192.168.242.0/24

コントローラーの内部ロードバランサーの IP アドレス

これは、Automation Controller のロードバランサーに割り当てられた静的 IP アドレスです。このアドレスは、アプリケーションサブネット範囲セグメント内にある必要があります。

デフォルト: 192.168.240.20

ハブの内部ロードバランサーの IP アドレス

これは、Automation Hub ロードバランサーに割り当てられた静的 IP アドレスです。このアドレスは、アプリケーションサブネット範囲セグメント内にある必要があります。

デフォルト: 192.168.240.21

5.2. ネットワークピアリングオプション

プラットフォームにアクセスするための多くのネットワーク設定が可能ですが、次の設定は GCP Marketplace で入手可能な Ansible Automation Platform で動作することが検証されています。

注記

この内容が Google Cloud Platform のドキュメントと一致するように努めていますが、時間の経過とともに精度が低下する可能性があります。GCP のネットワーキングトピックに関する情報源として、GCP ドキュメント を使用します。

5.3. VPC ピアリング

Ansible Automation Platform がプライベート VPC に存在するリソース、または Google Cloud Platform とオンプレミスネットワーク間のトランジットルーティングが存在するリソースにアクセスするには、VPC ピアリング が必要です。VPC ピアリングにより、GCP インフラストラクチャー内のさまざまなネットワークを直接接続できるようになります。VPC は個別に相互に接続され、その間に他のルーティングホップはありません。VPC デプロイメントは、デフォルトでパブリックインターネットからのアクセスがなくなります。

これは、GCP Marketplace で入手可能な Ansible Automation Platform で使用される GCP アーキテクチャーのデプロイモデルでもあります。

これは単純なピアリングモデルであり、複数のネットワークを接続する場合に役立ちます。

複雑なピアリングを設定できますが、ルーティングは時間の経過とともに複雑になる可能性があります。

VPC ピアリングおよびルーティングが設定されている場合、接続された VPC サブネットの仮想マシンを介して Ansible Automation Platform にアクセスできるか、または GCP とローカルネットワーク間の転送ルーティング設定がある場合は直接、Ansible Automation Platform にアクセスできます。

2 つ以上のネットワークがピアリングされている場合、Google は自動化されたアクションを実行してルーティングを支援しますが、ルーティング可能な更新を実行して GCP インフラストラクチャー内のトラフィックフローを有効にすることもできます。

前提条件

VPC ピアリングを使用して VPC を接続する前に、VPC と GCP Marketplace で入手可能な Ansible Automation Platform のVPC アドレス空間との間でトラフィックをルーティングする予定のネットワーク間で、ネットワークアドレス空間が重複していないことを確認する必要があります。これが試行された場合、GCP ポータルはピアリングを防止する必要があります。

以下の手順で、Ansible Automation Platform との VPC ピアリングを設定します。

手順

1. GCP ポータルで、**VPC Network** に移動します。
2. **VPC menu** で、**VPC Network Peering** を選択します。
3. **Create peering connection** をクリックします。
4. **CONTINUE** をクリックします。
5. **Name** フィールドに、必要に応じてピアリング接続の名前を入力します。
6. **Your VPC Network** フィールドで、ピアリングする予定の最初の VPC を選択します。
7. **Peered VPC Network** フィールドで、ピアリングする 2 番目の VPC を選択します。これは、GCP Marketplace で入手可能な Ansible Automation Platform VPC である必要があります。
    - これら 2 つのネットワーク間のサブネットルートは、デフォルトで作成されます。したがって、Ansible Automation Platform へのアクセスは、ピアリングされた VPC に存在する VM から発生する可能性があります。ただし、ハブアンドスポークネットワークモデルなど、より複雑なルーティングがある場合は、他のルートを作成する必要があります。
    - パブリック IP を含む **Exchange custom routes** と **Exchange subnet routes with public IP** を慎重に選択します。Google は、各フィールドの機能について説明しています。選択内容によって、トラフィックが VPC を通過する方法が異なります。通常、これらのチェックボックスは、ルートテーブルエクスチェンジを介して新しくピアリングされたネットワークと他のネットワーク間のルーティングを設定し、ネットワークトラフィックが複数のネットワークを横断できるようにします (トランジットルーティング)。
8. **Create** をクリックします。

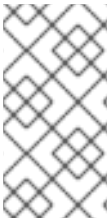
VPC ピアリングに関するルーティングテーブル、ファイアウォール、その他のネットワークコンポーネントの詳細は、GCP のドキュメント を参照してください。

5.4. 外部ロードバランサーの使用

ユーザーがインターネットに接続された任意のマシンからプラットフォームにアクセスできるようにする場合は、Ansible Automation Controller と Ansible Private Automation Hub をパブリックインターネットに公開することができます。

セキュリティー上の理由から、これはベストプラクティスとして推奨されません。

ただし、この実装はさまざまな GCP ツールで保護できます。このセクションでは、パブリックインターネットにリンクされているロードバランサーを接続する方法について説明しますが、Google のセキュリティー製品を使用してアクセスを強化する手順は含まれていません。Google Cloud Armor または同様の製品でパブリックエンドポイントを保護する場合にのみ、このアプローチを使用してください。

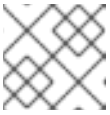


注記

GCP Marketplace で入手可能な Ansible Automation Platform は、2 つの **内部** アプリケーションロードバランサーと共にデプロイされます。これらのロードバランサーは、ローカル VPC 内でトラフィックを誘導し、パブリックロードバランサーを設定した場合でも、デプロイメントの中に含めた状態を保つようにする必要があります。

手順

1. 左上からメインメニューを選択します。
2. **Network Services** を選択します。**Network Services** が表示されない場合は、**View All Products** を選択します。
3. **Load Balancing** を選択します。
4. 上部のメニューで、**CREATE LOAD BALANCER** を選択します。
5. **Application Load Balancer (HTTP/S)** タイルを選択し、**START CONFIGURATION** をクリックします。
6. **From Internet to my VMs or serverless services** と **Global external Application Load Balancer** を選択します。
7. **CONTINUE** をクリックします。
8. ロードバランサーに名前を付けます (例: **<DeploymentName>-aap-<cntrlr/hub>-ext-lb)**。
9. 画面の左側で **Frontend** 設定が選択されていることを確認します。
10. **Frontend** 設定ページで、次のフィールドに入力します。
    - **Protocol**: HTTPS。
    - **Port**: 443。
    - **IP Address**: 既存の IP アドレスを選択するか、新しい IP アドレスを作成します。
    - **Certificate**: 独自の SSL 証明書を使用するか、Google 管理対象の証明書を使用できます。
    - **SSL Policy**: GCP のデフォルト、または別の設定済みポリシー。



注記

詳細は、SSL Certificates を参照してください。

11. **DONE** をクリックします。
12. 左側のメニューから **Backend configuration** を選択します。
13. **Backend services & backend buckets** ドロップダウンをクリックします。
14. **CREATE A BACKEND SERVICE** をクリックします。
    a. バックエンドサービスに名前を付けます (例: **<DeploymentName>-aap-<cntrlr/hub>-ext-lb-bknd-svc)**。
    b. **Backend type** を **Instance group** に設定します。
    c. **Protocol** を **HTTPS** に、**Named Port** を **https** に設定します。
    d. **Timeout** を **86400** に変更します。
    e. **Backends** セクションまで下にスクロールし、**New backend** フィールドで **Instance group** ドロップダウンを選択します。正しいインスタンスグループを選択します。
    Automation Controller ロードバランサーの場合、正しいインスタンスグループには接尾辞 **-aap-cntrlr-igm** が付いています。

    Automation Hub ロードバランサーの場合、正しいインスタンスグループには接尾辞 **-aap-hub-igm** が付いています。
    f. 表示されるダイアログボックスで、**USE EXISTING PORT NAME** を選択します。
    g. **Balancing mode** を **Rate** に設定します。
    h. **Maximum RPS** を 300 に設定します。
    i. **Cloud CDN** が表示されるまで下にスクロールします。**Cloud CDN** のチェックボックスをオフにします。
    j. **Health check** を示すテキストボックスが表示されるまで下にスクロールします。
    k. ドロップダウンメニューを選択し、**CREATE A HEALTH CHECK** をクリックします。
    l. ヘルスチェックの名前を入力します (例: **<DeploymentName>-aap-<cntrlr/hub>-ext-lb-hc**)。
    m. Automation Controller の場合は、次のヘルスチェック設定を使用します
        - **Health Check** ダイアログボックスで、**Protocol** を **HTTPS** に、**Port** を **8052** に設定します。
        - **Request** を **/api/v2/ping/** に設定します。
    n. Automation Hub の場合、次のヘルスチェック設定を使用します
        - **Health Check** ダイアログボックスで、**Protocol** を **HTTPS** に、**Port** を **8080** に設定します。
        - **Request** を **/api/galaxy/pulp/api/v3/status/** に設定します。

o. **Health criteria** セクションまで下にスクロールします。

p. **Check interval** フィールドに値 15 を入力します。

q. **Timeout** フィールドに値 10 を入力します。

r. **Healthy threshold** フィールドに値 2 を入力します。

s. **Unhealthy threshold** フィールドに値 10 を入力します。

t. **SAVE** をクリックします。
Backend services & backend buckets ウィンドウに戻ります。

u. **CREATE** をクリックします。
これにより、**Backend configuration** セクションに戻ります。

v. **OK** をクリックします。

w. **CREATE** をクリックして、Automation Controller または Automation Hub UI のロードバランサーを作成します。これが完了するまでに数分かかります。

15. ロードバランサーが作成されました。

16. SSL 証明書で使用したドメインの DNS レコードを、ロードバランサーの IP アドレスを指すように設定します。

この時点で、Ansible Automation Platform Automation Controller UI にアクセスできるはずです。管理パスワードを取得すると、ログインできます。

Private Automation Hub に対して同じプロセスを繰り返します。バックエンドの設定時にインスタンスグループ **-aap-hub-igm** を選択することを除いて、プロセスは同じです。

第6章 追加の設定

次の章では、GCP でのデプロイが完了した後に実行できる Ansible Automation Platform の設定手順について説明します。

6.1. デフォルトの管理者パスワードの変更

Ansible Automation Platform のデフォルトの管理者パスワードは、GCP Marketplace で入手可能な Ansible Automation Platform がデプロイされるときに無作為に生成されます。以下の手順に従って、Automation Controller と Automation Hub の両方の管理者パスワードを変更します。

手順

1. GCP Secrets Manager Console に移動します。
    a. **<deployment_name>-aap-admin** という名前の Ansible Automation Platform デプロイメントのシークレットを見つけて開きます。
    b. **NEW VERSION** を選択して、新しいバージョンを追加します。
    c. パスワードの秘密の値を入力します。
    d. **Disable all past versions** チェックボックスをオンにします。
    e. **ADD NEW VERSION** をクリックします。
2. 実行中の Ansible Automation Platform 仮想マシンインスタンスを変更して、新しい管理者パスワードを使用します。
    a. **GCP VM Instances** コンソールに移動します。
    b. Ansible Automation Platform デプロイメント用に Automation Controller 仮想マシンインスタンス 1 台と Automation Hub VM インスタンスを 1 台特定して削除します。
    c. Automation Controller と Automation Hub の Instance グループが新しい仮想インスタンスを作成するまで待ちます。
3. 新しい管理者パスワードは、新しい Automation Controller およびAutomation Hub 仮想マシンインスタンスが **Running** の状態に達したときに使用できます。

6.2. AUTOMATION CONTROLLER と AUTOMATION HUB 仮想マシンインスタンスの SSL/TLS 証明書とキーの置き換え

デフォルトでは、仮想マシンインスタンスは、有効期間が 10 年の自己署名 SSL/TLS 証明書で保護されます。証明書の有効期限が切れた場合、または仮想マシンインスタンスで独自の証明書を使用する場合は、SSL/TLS 証明書とキーを置き換える必要があります。

手順

1. GCP Secrets Manager Console に移動します。
    a. **<deployment_name>-pulp_cert** という名前の Ansible Automation Platform デプロイメントのシークレットを見つけて開きます。
    b. **NEW VERSION** を選択して、新しいバージョンを追加します。

c. 新しい SSL/TLS 証明書の値を入力します。

d. **Disable all past versions** チェックボックスをオンにします。

e. **ADD NEW VERSION** をクリックします。

2. GCP Secrets Manager Console に移動します。

a. **<deployment_name>-pulp_key** という名前の Ansible Automation Platform デプロイメントのシークレットを見つけて開きます。

b. **NEW VERSION** を選択して、新しいバージョンを追加します。

c. 新しい SSL/TLS キー値を入力します。

d. **Disable all past versions** チェックボックスをオンにします。

e. **ADD NEW VERSION** をクリックします。

3. 新しい SSL/TLS 証明書とキーを使用するように、実行中の Ansible Automation Platform 仮想マシンインスタンスを変更します。

a. **GCP VM Instances** コンソールに移動します。

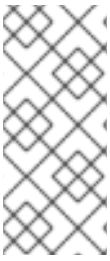
b. Ansible Automation Platform デプロイメントのすべての Automation Controller と Automation Hub 仮想マシンインスタンスを特定して削除します。

c. Automation Controller と Automation Hub の Instance グループが新しい仮想インスタンスを作成するまで待ちます。

4. 新しい Automation Controller と Automation Hub 仮想マシンインスタンスが **Running** のインスタンス状態になると、新しい証明書が使用されます。

6.3. SSL による内部通信の保護

GCP Marketplace で入手可能な Ansible Automation Platform は、ハブインスタンスとコントローラーインスタンスの前に 1 つずつ、2 つの **内部** アプリケーションロードバランサーと共にデプロイされます。これらの内部ロードバランサーは、デプロイメントの完了後に SSL 証明書を使用して設定する必要があります。



注記

これらの内部ロードバランサーを介してトラフィックを保護することは、前の手順で外部ロードバランサーを介してトラフィックを保護することとは異なります。このプロセスにより、トラフィックがプライベート GCP VPC にローカライズされている場合でも、HTTP トラフィックが確実に暗号化されます。Automation Controller と Automation Hub のロードバランサーのいずれの場合も、同じ手順を実行してください。

名前形式が **<DEPLOYMENT_NAME>-aap-<cntrlr/hub>-int-lb** である、Automation Controller と Automation Hub ロードバランサーの両方を変更します。

手順

1. 次のコマンドを使用して、Automation Controller または Automation Hub 証明書を生成します。

```
$ openssl req -x509 -nodes -newkey rsa:2048 -keyout key.pem -out cert.pem -sha256 -days
365
```

2. GCP コンソールで、Load Balancing ページに移動します。
3. 検索バーにデプロイメントの名前を入力して、ロードバランサーをフィルタリングします。
4. **<DEPLOYMENT_NAME>-aap-<cntrlr/hub>-int-lb** をクリックします。
5. **Edit** をクリックします。
6. **Frontend configuration** をクリックします。
7. **ADD FRONTEND IP AND PORT** をクリックします。次の値を使用します。
    a. **Protocol**: HTTPS (HTTP/2 を含む)。
    b. **Subnetwork**: 利用可能な aap-subnet を選択します。
    c. **Port**: 443
    d. **IP Address**: **<DEPLOYMENT_NAME>-aap<cntrlr/hub>-intl-lb-ip**
8. 証明書をすでに追加している場合は、それを選択します。
    a. 証明書を追加していない場合は、**CREATE A NEW CERTIFICATE** をクリックします。
    b. 証明書の名前を指定します。
    c. 以前に生成した証明書を使用して、**cert.pem** の内容をコピーし、**Certificate** の下に貼り付けます。
    d. 以前に生成した証明書キーを使用して、**key.pem** の内容をコピーし、**Private Key** の下に貼り付けます。
    e. **Create** をクリックします。
9. **Done** をクリックします。
10. オプション: HTTP フロントエンド設定を削除するには、以下を実行します。
    a. ロードバランサーインスタンスを開きます。
    b. **Frontend Configuration** をクリックします。UI の左側に設定が表示されます。
    c. 削除する設定までスクロールします。
    d. ごみ箱アイコンをクリックして、設定を削除します。
11. **Update** をクリックして、更新を確認します。

6.4. セキュリティーに関する考慮事項

Multi factor Authentication (MFA) を有効にできる ID プロバイダーを使用して Red Hat Single Sign-On を設定するには、こちら の手順に従ってエンタープライズ認証を Ansible Automation Platform に接続します。

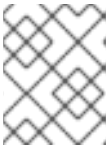
インフラストラクチャーサービスを保護することは、クラウドのデプロイメントにおいて重要なステップです。Ansible Automation Platform from GCP Marketplace のデプロイメントの一部として使用されるサービスについては、GCP ドキュメント の実装とセキュリティーに関する提案に従ってください。

第7章 コマンドジェネレーター

コマンドジェネレーターは、Ansible-on-clouds 操作 Playbook コレクションによって提供される操作 Playbook を起動するコマンドを生成するために使用されます。

このプロセスには 5 つのステップがあります。

1. **ansible-on-clouds-ops** コンテナーイメージをプルします。
2. 利用可能な Playbook を一覧表示します。
3. コマンドジェネレーターを使用して、データファイルと実行する次のコマンドを生成します。**command_generator_vars** および command_generator は docker コンテナーを使用して実装され、docker コマンドラインインターフェイスを使用して実行されます。
4. データファイルを入力し、以前に生成されたコマンドを実行します。このコマンドにより、すべてのパラメーターを含む最後のコマンドが生成されます。



注記

このステップが完了したら、生成されたコマンドを保存して、必要なときに Playbook を実行することができます。

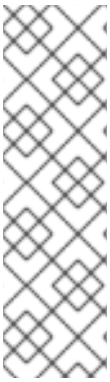
5. 最後のコマンドを実行します。

前提条件

- Docker
- GCP 認証情報ファイル
- Google Cloud へのインターネット接続

7.1. ANSIBLE-ON-CLOUDS-OPS コンテナーイメージのプル

デプロイメントと同じタグバージョンを持つ Clouds 運用コンテナーの Ansible の Docker イメージをプルします。デプロイしたバージョンが不明な場合、Ansible on Clouds デプロイメントの現在のバージョンを確認する方法の詳細は、コマンドジェネレーター と Playbook **gcp_get_aoc_version** を参照してください。



注記

Docker イメージをプルする前に、Docker を使用して registry.redhat.io にログインしていることを確認してください。以下のコマンドを使用して registry.redhat.io にログインします。

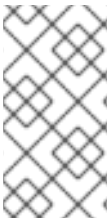
```
$ docker login registry.redhat.io
```

レジストリーのログインに関する詳細は、Registry Authentication を参照してください。

たとえば、基盤デプロイメントのバージョンが 2.4.20240215-00 の場合は、タグが 2.4.20240215 の運用イメージをプルする必要があります。

以下のコマンドを使用します。

```
$ export IMAGE=registry.redhat.io/ansible-on-clouds/ansible-on-clouds-ops-rhel9:2.4.20240215
$ docker pull $IMAGE --platform=linux/amd64
```



注記

基盤のデプロイメントバージョンが 2.4.20240215-00 ではない場合は、Released versions ページの表で、**Ansible on Clouds version** 列で一致するデプロイメントバージョンを確認し、IMAGE 環境変数の **Ansible-on-clouds-ops container image** 列で使用する対応する運用イメージを見つけます。

7.2. 利用可能な PLAYBOOK の一覧表示

手順

1. 詳細のない利用可能な Playbook の一覧については、以下のコマンドを使用します。

```
$ docker run --rm $IMAGE command_generator_vars | grep Playbook

The current version of the operational playbooks collection contains the following playbooks:

Playbook: gcp_aap_health_check
Playbook: gcp_add_extension_nodes
Playbook: gcp_add_labels
Playbook: gcp_backup_delete
Playbook: gcp_backup_deployment
Playbook: gcp_backup_list
Playbook: gcp_backups_delete
Playbook: gcp_check_aoc_version
Playbook: gcp_deployment_inventory
Playbook: gcp_get_aoc_version
Playbook: gcp_health_check
Playbook: gcp_list_deployments
Playbook: gcp_nodes_health_check
Playbook: gcp_remove_extension_nodes
Playbook: gcp_remove_labels
Playbook: gcp_restore_deployment
Playbook: gcp_setup_logging_monitoring
Playbook: gcp_upgrade
```

2. 利用可能なすべての Playbook の一覧を提供し、コマンドジェネレーターを使用するには、以下のコマンドを使用します。

```
$ docker run --rm $IMAGE command_generator_vars
```

これにより、Playbook の一覧と以下のようなコマンドが提供されます。

```
================================================
Playbook: gcp_upgrade
Description: Performs the upgrade of the Ansible Automation Platform from GCP
Marketplace components to the latest version.
------------------------------------------------
```

```
Performs the upgrade of the Ansible Automation Platform from GCP Marketplace
components to the latest version.

---------------------------------------------
Command generator template:

docker run --rm $IMAGE command_generator gcp_upgrade [--ansible-config
ansible_config_path>] \
-d <deployment_name> -c <cloud_credentials_path> --extra-vars 'gcp_compute_region=
<gcp_compute_region> gcp_compute_zone=<gcp_compute_zone> gcp_backup_taken=
<true|false>'
=============================================
```

7.3. データファイルの生成

手順

1. コマンドジェネレーターを実行します。

```
$ docker run --rm -v <local_directory_data_file>:/data $IMAGE command_generator_vars
<playbook_name> --output-data-file /data/<data-file>.yml
```

このコマンドの出力は、実行するコマンドとデータファイルテンプレートです。データファイルは **<local_data_file_directory>** にも保存されます。これは、データを入力するテンプレートです。

以下の例では、**gcp_backup_deployment** Playbook を使用します。

```
$ docker run --rm -v <local_data_file_directory>:/data $IMAGE command_generator_vars
gcp_backup_deployment \
--output-data-file /data/backup.yml
```

2. 以下の出力が生成されます。

```
=============================================
Playbook: gcp_backup_deployment
Description: This playbook is used to backup the AoC Self-managed GCP environment.
---------------------------------------------
This playbook is used to backup the AoC Self-managed GCP environment.
For more information regarding backup and restore, visit our official documentation -

---------------------------------------------
Command generator template:

docker run --rm -v /tmp:/data $IMAGE command_generator gcp_backup_deployment --data-
file /data/backup.yml

Data template:

gcp_backup_deployment:
  cloud_credentials_path:
  deployment_name:
  extra_vars:
    gcp_bucket_backup_name:
```

```
  gcp_compute_region:
  gcp_compute_zone:

==============================================
```

7.4. データファイルの入力

手順

- データファイルの生成 で生成されたデータファイルを編集します。
  パスを表す属性は絶対パスである必要があります。**command_generator** は、最後のコマンドでボリュームを自動的にマウントします。

  たとえば、**gcp_backup_deployment** Playbook の場合、ファイルは以下のようになります。

```
gcp_backup_deployment
  cloud_credentials_path: /path/to/credentials
  deployment_name: my-deployment
  extra_vars:
    cp_bucket_backup_name: my-bucket
    gcp_compute_region: us-east1
    gcp_compute_zone: us-east1-b
```

7.5. 生成されたコマンドの実行

手順

1. マウントされたボリュームが、データファイルが置かれるディレクトリーを指していることを確認します。
   gcp_backup_deployment Playbook の例の場合は、以下のとおりです。

```
$ docker run --rm -v /tmp:/data $IMAGE command_generator gcp_backup_deployment --
data-file /data/backup.yml
```

   これにより、以下の出力が生成されます。

```
Command to run playbook:

$ docker run --rm --env PLATFORM=GCP -v
/path/to/credentials:/home/runner/.gcp/credentials:ro \
--env ANSIBLE_CONFIG=../gcp-ansible.cfg  $IMAGE\
redhat.ansible_on_clouds.gcp_backup_deployment \
-e 'gcp_service_account_credentials_json_path=/home/runner/.gcp/credentials  \
gcp_deployment_name=my-deployment gcp_compute_region=us-east1
gcp_compute_zone=us-east1-b'
```

   この新しいコマンドには、Playbook の実行に必要なパラメーター、環境変数、およびマウントされたボリュームがあります。

2. 生成されたコマンドを実行します。このコマンドを保存して、必要に応じて後で再実行できます。

7.6. PLAYBOOK の使用

このドキュメントでは、すべてではありませんが、一部の Playbook を説明します。ここでは、Ansible on Clouds デプロイメントから情報を取得するか、情報を確認するために使用されるものを説明します。これらの Playbook はデプロイメントを変更しません。

gcp_aap_health_check

この Playbook は、Ansible アプリケーションが正常かどうかをチェックします。

```
$ docker run --rm $IMAGE command_generator_vars gcp_aap_health_check
```

これにより、以下の出力が生成されます。

```
================================================
Playbook: gcp_aap_health_check
Description: This playbook checks if the deployment is healthy using the Ansible health service.
------------------------------------------------
The health check consists of checking the Ansible Automation Platform from GCP Marketplace
environemnt to verify it is healthy.

------------------------------------------------
Command generator template:

docker run --rm $IMAGE command_generator gcp_aap_health_check [--ansible-config
ansible_config_path>] -d <deployment_name> -c <cloud_credentials_path> --extra-vars
'gcp_compute_region=<gcp_compute_region> gcp_compute_zone=<gcp_compute_zone>'
================================================
```

パラメーターを置き換えてこのコマンドを起動すると、起動する新しいコマンドが生成され、次のように出力されます。

```
...
PLAY RECAP *********************************************************************
localhost                  : ok=29   changed=1    unreachable=0    failed=0    skipped=0    rescued=0
ignored=0
```

failed がゼロに等しくない場合は、Ansible on Cloud デプロイメントに問題があることを示します。

gcp_add_labels

この Playbook は、デプロイメントにラベルを追加します。

```
$ docker run --rm $IMAGE command_generator_vars gcp_add_labels
```

これにより、以下の出力が生成されます。

```
================================================
Playbook: gcp_add_labels
Description: This playbook adds labels to the deployment.
------------------------------------------------
Add labels to the Ansible Automation Platform from GCP Marketplace deployment.

------------------------------------------------
Command generator template:
```

```
docker run --rm $IMAGE command_generator gcp_add_labels -d <deployment_name> -c
<cloud_credentials_path> --extra-vars 'gcp_compute_region=<gcp_compute_region>
gcp_compute_zone=<gcp_compute_zone> gcp_labels=<gcp_labels>'
================================================
```

パラメーター **gcp_labels** は、追加または更新する **key=value** ペアのコンマ区切りのリストです。たとえば、key1=value1,key2=value2 です。

パラメーターを置き換えてこのコマンドを起動すると、起動する新しいコマンドが生成され、次のように出力されます。

```
...
PLAY RECAP *********************************************************************
localhost                  : ok=22   changed=2    unreachable=0    failed=0    skipped=1    rescued=0
ignored=0
```

gcp_remove_labels

この Playbook は、デプロイメントからラベルを削除します。

```
$ docker run --rm $IMAGE command_generator_vars gcp_remove_labels
```

これにより、以下の出力が生成されます。

```
================================================
Playbook: gcp_remove_labels
Description: This playbook removes labels from the deployment.
------------------------------------------------
Remove labels from the Ansible Automation Platform from GCP Marketplace deployment.

------------------------------------------------
Command generator template:

docker run --rm $IMAGE command_generator gcp_remove_labels -d <deployment_name> -c
<cloud_credentials_path> --extra-vars 'gcp_compute_region=<gcp_compute_region>
gcp_compute_zone=<gcp_compute_zone> gcp_labels=<gcp_labels>'
================================================
```

パラメーター **gcp_labels** は、削除するキーのコンマ区切りのリストです。たとえば、key1,key2 です。

パラメーターを置き換えてこのコマンドを起動すると、起動する新しいコマンドが生成され、次のように出力されます。

```
...
PLAY RECAP *********************************************************************
localhost                  : ok=22   changed=2    unreachable=0    failed=0    skipped=1    rescued=0
ignored=0
```

gcp_check_aoc_version

この Playbook は、Ansible on Cloud のバージョンがコマンドジェネレーターコンテナーと同じかどうかを確認します。この確認は、Playbook が呼び出されるたびに行われます。

```
$ docker run --rm $IMAGE command_generator_vars gcp_check_aoc_version
```

これにより、以下の出力が生成されます。

```
================================================
Playbook: gcp_check_aoc_version
Description: Check the operational container version matches the Ansible on Clouds version.
------------------------------------------------
Check the operational container version matches the Ansible on Clouds version.

------------------------------------------------
Command generator template:

docker run --rm $IMAGE command_generator gcp_check_aoc_version [--ansible-config
ansible_config_path>] -c <cloud_credentials_path> -d <deployment_name>
================================================
```

パラメーターを置き換えてこのコマンドを起動すると、起動する新しいコマンドが生成され、次のように出力されます。

```
...
TASK [redhat.ansible_on_clouds.standalone_check_aoc_version : Verify operational playbook and
Ansible on Clouds deployment versions] ***
ok: [localhost] => {
    "changed": false,
    "msg": "This operation playbook version and the Ansible on Clouds deployment version are
identical: 2.4.20230606-00"
}

PLAY RECAP *********************************************************************
localhost                  : ok=8    changed=0    unreachable=0    failed=0    skipped=0    rescued=0
ignored=0
```

failed がゼロ以外の場合は、Ansible on Clouds デプロイメントのバージョンが **command_generator** コンテナーと一致せず、コマンドジェネレーターがそのデプロイメントを管理するには別のバージョンが必要であることを示します。

gcp_get_aoc_version

この Playbook は、Ansible on Clouds デプロイメントのバージョンを取得します。

```
$ docker run --rm $IMAGE command_generator_vars gcp_get_aoc_version
```

これにより、以下の出力が生成されます。

```
================================================
Playbook: gcp_get_aoc_version
Description: Get the current Ansible on Clouds version.
------------------------------------------------
Get the current Ansible on Clouds version.

------------------------------------------------
Command generator template:

docker run --rm $IMAGE command_generator gcp_get_aoc_version [--ansible-config
ansible_config_path>] -c <cloud_credentials_path> -d <deployment_name>
================================================
```

パラメーターを置き換えてこのコマンドを起動すると、起動する新しいコマンドが生成され、次のように出力されます。

```
...
TASK [Print version] ************************************************************
ok: [localhost] => {
    "msg": "The AOC version is 2.4.20230606-00"
}

PLAY RECAP **********************************************************************
localhost                  : ok=5    changed=0    unreachable=0    failed=0    skipped=0    rescued=0
ignored=0
```

gcp_health_check

この Playbook は、ノードと Ansible アプリケーションが正常かどうかをチェックします。

```
$ docker run --rm $IMAGE command_generator_vars gcp_health_check
```

これにより、以下の出力が生成されます。

```
================================================
Playbook: gcp_health_check
Description: This playbook checks if the Ansible Automation Platform from GCP Marketplace
deployment is healthy.
------------------------------------------------
The health check consists of checking the Ansible Automation Platform from GCP Marketplace heatlh
checks
and the health of the monitoring exporter.

------------------------------------------------
Command generator template:

docker run --rm $IMAGE command_generator gcp_health_check [--ansible-config
ansible_config_path>] -c <cloud_credentials_path> -d <deployment_name> --extra-vars
'gcp_compute_region=<gcp_compute_region> gcp_compute_zone=<gcp_compute_zone>'
================================================
```

パラメーターを置き換えてこのコマンドを起動すると、起動する新しいコマンドが生成され、次のように出力されます。

```
...
PLAY RECAP **********************************************************************
localhost                  : ok=47   changed=1    unreachable=0    failed=0    skipped=0    rescued=0
ignored=0
```

failed がゼロに等しくない場合は、ノードまたは Ansible on Cloud デプロイメントに問題があることを示します。

gcp_list_deployments

この Playbook にはデプロイメントがリストされています。リージョンとゾーンはオプションです。

```
$ docker run --rm $IMAGE command_generator_vars gcp_list_deployments
```

これにより、以下の出力が生成されます。

```
================================================
Playbook: gcp_list_deployments
Description: This playbook generates a list of available Ansible Automation Platform from GCP
Marketplace deployments.
------------------------------------------------
This playbook is used to generate a list of available Ansible Automation Platform from GCP
Marketplace deployments.

------------------------------------------------
Command generator template:

docker run --rm $IMAGE command_generator gcp_list_deployments -c <cloud_credentials_path> --
extra-vars '[gcp_compute_region=<gcp_compute_region>] [gcp_compute_zone=
<gcp_compute_zone>]'
================================================
```

パラメーターを置き換えてこのコマンドを起動すると、起動する新しいコマンドが生成され、次のように出力されます。

```
...
TASK [Show deployment list] ****************************************************
ok: [localhost] => {
    "msg": [
        "Deployment list: ['dep1', 'dep2', 'dep3']"
    ]
}

PLAY RECAP *********************************************************************
localhost                  : ok=7    changed=0    unreachable=0    failed=0    skipped=0    rescued=0
ignored=0
```

gcp_nodes_health_check

この Playbook は、ノードが正常かどうかをチェックします。

```
$ docker run --rm $IMAGE command_generator_vars gcp_nodes_health_check
```

これにより、以下の出力が生成されます。

```
================================================
Playbook: gcp_nodes_health_check
Description: This role runs a health check on a group of nodes in the Ansible Automation Platform
from GCP Marketplace deployment
------------------------------------------------
The playbook checks if the Ansible Automation Platform from GCP Marketplace monitoring exporter
is up and running.

------------------------------------------------
Command generator template:

docker run --rm $IMAGE command_generator gcp_nodes_health_check [--ansible-config
ansible_config_path>] -d <deployment_name> -c <cloud_credentials_path> --extra-vars
'check_monitoring=True'
================================================
```

パラメーターを置き換えてこのコマンドを起動すると、起動する新しいコマンドが生成され、次のように出力されます。

```
...
PLAY RECAP *********************************************************************
localhost                  : ok=47   changed=1    unreachable=0    failed=0    skipped=0    rescued=0
ignored=0
```

failed がゼロに等しくない場合は、デプロイメント内のノードに問題があることを示します。

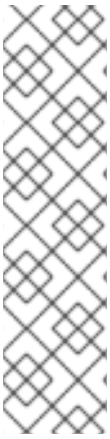
第8章 自動化ワークロード

GCP Marketplace のデフォルトの Ansible Automation Platform は、100 個のマネージドノードを自動化するように設計され、ライセンスが付与されています。

8.1. 自動化のパフォーマンス

このオファーで管理対象ノード割り当てを自動化する場合は、以下の想定条件があります。これらの基準の境界外での自動化はサポートされていませんが、自動化によっては機能する可能性があります。

メトリクス	しきい値
同時実行ジョブ	10
ジョブごとのフォーク	10



注記

GCP Marketplace で入手可能な Ansible Automation Platform は、3 つの n2-standard-2 インスタンスを使用して実行されます。そのうちの 2 つは Automation Controller を実行し、1 つは Automation Hub を実行します。Automation Controller インスタンスは、管理対象アクティブノード 100 個分の標準的なワークロードをまとめてサポートします。Red Hat はこれをテストし、それぞれ最大 10 個のフォークをサポートすることを証明しました。操作基準は、両方の Automation Controller ノードで 7 秒間隔で 2 つのメッセージを生成する出力集中型のチャットワークロードを使用して設定およびテストされました。I/O の負荷が高いワークロードは、これらの条件の境界内で機能しない可能性があり、拡張ノードを使用して、デプロイメントをスケーリングし、このような自動化をサポートする必要がある場合があります。

8.2. デプロイメントのスケーリング

GCP Marketplace で入手可能な Ansible Automation Platform では、サポート対象の管理対象ノードに最初に設定されている数よりも多く、デプロイメントをスケーリングする場合には、別売りの拡張ノードを使用して手動でスケーリングできます。

拡張ノードは、即時のスケーリング要件に応じてスケールアップまたはスケールアウトするためにデプロイできる追加のコンピュートインスタンスです。高度な並列自動化操作が必要な場合は、スケールアップするコンピュートシェイプを選択できます。時間の経過とともにより多くのノードを自動化する必要がある場合は、スケールアウトするコンピュートシェイプを選択できます。

拡張ノードは、GCP Marketplace で入手可能な Ansible Automation Platform の機能を拡張する方法としてサポートされています。

注記

Red Hat は、お客様の設計および実装を使用して拡張された環境をサポートしていません。

第9章 モニタリングおよびロギング

メトリクスを Google Cloud Platform モニタリングシステムに送信して、Google Cloud Platform UI で視覚化することができます。GCP Marketplace で入手可能な Ansible Automation Platform のメトリクスとロギングは、デフォルトで無効になっています。これらのメトリクスを GCP に送信するにはコストがかかるためです。詳細は、Cloud Monitoring と Cloud Logging をそれぞれ参照してください。

GCP のモニタリングとロギングは次のいずれかで設定できます。

- デプロイメント時 (デプロイメント時のモニタリングとロギングのセットアップ を参照)、または
- デプロイメント後

9.1. デプロイメント後のモニタリングとロギングのセットアップ

registry.redhat.com から入手可能な **gcp_setup_logging_monitoring** Playbook を使用して、デプロイメント後にロギングとモニタリングを開始または停止できます。

9.1.1. 必須のパーミッション

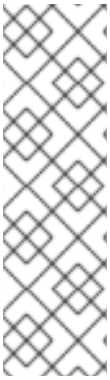
ロギングとモニタリングをセットアップするには、次の GCP IAM 権限が必要です。

```
required-roles:

Service Account User
Compute Instance Admin (v1)

required-permissions:

cloudsql.instances.connect
cloudsql.instances.get
cloudsql.instances.login
cloudsql.users.update
compute.addresses.get
compute.addresses.list
compute.instances.delete
compute.instances.get
compute.instances.list
compute.instances.setLabels
compute.zoneOperations.get
deploymentmanager.deployments.list
deploymentmanager.manifests.get
deploymentmanager.manifests.list
file.instances.get
file.instances.list
file.instances.update
file.operations.get
iap.tunnelInstances.accessViaIAP
logging.logEntries.create
monitoring.timeSeries.create
resourcemanager.projects.get
runtimeconfig.variables.create
runtimeconfig.variables.get
runtimeconfig.variables.list
```

```
runtimeconfig.variables.update
secretmanager.secrets.create
secretmanager.secrets.delete
secretmanager.secrets.get
secretmanager.versions.add
secretmanager.versions.get
secretmanager.versions.list
servicenetworking.operations.get
servicenetworking.services.addPeering
serviceusage.services.list
```

9.1.2. ansible-on-clouds-ops コンテナーイメージのプル

基盤デプロイメントのバージョンと一致する Ansible on Clouds 運用コンテナーの Docker イメージをプルします。デプロイしたバージョンが不明な場合は、Ansible on Clouds デプロイメントの現在のバージョンを確認する方法の詳細について、コマンドジェネレーター と Playbook **gcp_get_aoc_version** を参照してください。



注記

Docker イメージをプルする前に、Docker を使用して registry.redhat.io にログインしていることを確認してください。以下のコマンドを使用して registry.redhat.io にログインします。

```
$ docker login registry.redhat.io
```

レジストリーのログインに関する詳細は、Registry Authentication を参照してください。

たとえば、基盤デプロイメントのバージョンが 2.4.20240215-00 の場合は、タグが 2.4.20240215 の運用イメージをプルする必要があります。

以下のコマンドを使用します。

```
$ export IMAGE=registry.redhat.io/ansible-on-clouds/ansible-on-clouds-ops-rhel9:2.4.20240215
$ docker pull $IMAGE --platform=linux/amd64
```

注記

基盤のデプロイメントバージョンが 2.4.20240215-00 ではない場合は、Released versions ページの表で、**Ansible on Clouds バージョン** 列にある一致するデプロイメントバージョンを参照してください。**Ansible-on-clouds-ops コンテナーイメージ列の、IMAGE 環境変数の対応する運用イメージ** を見つけます。

9.1.3. ansible-on-clouds-ops コンテナーを実行してデータファイルを生成する

次のコマンドで、必要なデータファイルを生成します。これらのコマンドは、ディレクトリーと空のデータテンプレートを作成します。このテンプレートは、データが設定されると Playbook の生成に使用されます。

手順

1. 設定ファイルを保存するフォルダーを作成します。

```
$ mkdir command_generator_data
```

2. **command_generator_data** フォルダーに設定ファイルのテンプレートを追加します。

注記

Linux では、コマンドジェネレーターが作成したファイルまたはディレクトリーは、デフォルトで **root:root** の所有となります。ファイルとディレクトリーの所有権を変更するには、ファイルの作成後に **sudo chmod** コマンドを実行します。詳細は、コマンドジェネレーター - root が所有する Linux ファイル を参照してください。

```
$ docker run --rm -v $(pwd)/command_generator_data:/data $IMAGE \
command_generator_vars gcp_setup_logging_monitoring \
--output-data-file /data/logging-monitoring.yml
```

3. これらのコマンドを実行すると、**command_generator_data/logging-monitoring.yml** テンプレートファイルが作成されます。

注記

以下のファイルの例では、**ansible_config_path** はオプションです。

このテンプレートファイルは次のようになります。

```
gcp_setup_logging_monitoring:
  ansible_config_path:
  cloud_credentials_path:
  deployment_name:
  extra_vars:
    components:
    default_collector_interval:
    logging_enabled:
    monitoring_enabled:
```

9.1.4. データファイルの更新

パラメーターが必要ない場合は、そのパラメーターを設定ファイルから削除してください。

手順

- **command_generator_data/logging-monitoring.yml** ファイルを編集し、次のパラメーターを設定します。
- **ansible_config_path** は、**ansible-on-cloud オファリング** の標準設定としてデフォルトで使用されますが、環境に追加の要件がある場合は、独自の設定を指定できます。
- **cloud_credentials_path** は、認証情報への絶対パスです。これは、絶対パス名である必要があります。
- **deployment_name** は、デプロイメントの名前です。

- **components** (任意): セットアップを実行するコンポーネントのタイプ。デフォルトは [“controller”, “hub”] で、ロギングとモニタリングが Automation controller と Automation Hub の両方で有効になることを意味します。
- モニタリングを有効にするには、**monitoring_enabled** (任意) を **true** に設定し、無効にするには **false** に設定します。デフォルトは **false** です。
- ロギングを有効にするには、**logging_enabled** (オプション) を **true** に設定し、それ以外の場合は **false** に設定します。デフォルトは **false** です。
- **default_collector_interval** (任意) は、モニタリングデータを Google Cloud に送信する必要がある頻度です。デフォルト = 59 秒。

注記

このサービスの Google コストはその周期に依存するため、コレクター間隔の値が大きいほどコストは低くなります。

59 秒未満の値を設定しないでください。

注記

モニタリングとロギングが無効になっている場合、'default_collector_interval' の値は自動的に **0** に設定されます。

データファイルを入力すると、次のようになります。

次の値は例です。

注記

このセクションで説明されている任意のパラメーターは、以下のデータファイルの例では省略されています。Playbook は、データファイルから省略された任意のパラメーターのデフォルト値を使用します。任意のパラメーターのデフォルト値をオーバーライドする場合は、そのパラメーターをデータファイルに含めて値を割り当てる必要があります。

```
gcp_setup_logging_monitoring:
  cloud_credentials_path: ~/secrets/GCP-secrets.json
  deployment_name: AnsibleAutomationPlatform
  extra_vars:
```

9.1.5. Playbook の生成

Playbook を生成するには、コマンドジェネレーターを実行して CLI コマンドを生成します。

```
docker run --rm -v $(pwd)/command_generator_data:/data $IMAGE command_generator
gcp_setup_logging_monitoring \
--data-file /data/logging-monitoring.yml
```

次のコマンドが生成されます。

```
docker run --rm --env PLATFORM=GCP -v </path/to/gcp/service-
account.json>:/home/runner/.gcp/credentials:ro \
--env ANSIBLE_CONFIG=../gcp-ansible.cfg --env DEPLOYMENT_NAME=<deployment_name> --
env GENERATE_INVENTORY=true  \
$IMAGE redhat.ansible_on_clouds.gcp_setup_logging_monitoring -e 'gcp_deployment_name=
<deployment_name> \
gcp_service_account_credentials_json_path=/home/runner/.gcp/credentials  monitoring_enabled=
<monitoring_enabled> \
logging_enabled=<logging_enabled> default_collector_interval=<interval>'
```

指定されたコマンドを実行して、Playbook を実行します。

```
$ docker run --rm --env PLATFORM=GCP -v /path/to/credentials:/home/runner/.gcp/credentials:ro \
--env ANSIBLE_CONFIG=../gcp-ansible.cfg --env DEPLOYMENT_NAME=mu-deployment \
--env GENERATE_INVENTORY=true  $IMAGE
redhat.ansible_on_clouds.gcp_setup_logging_monitoring \
-e 'gcp_deployment_name=mu-deployment \
gcp_service_account_credentials_json_path=/home/runner/.gcp/credentials  components=
["hubs","controllers"]\
monitoring_enabled=True logging_enabled=True default_collector_interval=60s'
```

このプロセスでは、以下のような出力が表示されますが、時間がかかる場合があります。

```
TASK [redhat.ansible_on_clouds.setup_logging_monitoring : Update runtime variable
logging_enabled] ***
changed: [<user_name> -> localhost]

TASK [redhat.ansible_on_clouds.setup_logging_monitoring : Update runtime variable
monitoring_enabled] ***
changed: [<user_name> -> localhost]

PLAY RECAP *********************************************************************
<user_name>    : ok=20   changed=6    unreachable=0    failed=0    skipped=2    rescued=0
ignored=0
```

9.2. モニタリングとロギングのカスタマイズ

メトリクスは、Ansible、Podman、および Google Ops Agent によって提供されます。Google Ops Agent と Podman は Automation Controller と Automation Hub 仮想マシンインスタンスにインストールされますが、Ansible メトリックは Automation Hub インスタンスにのみインストールされます。

設定可能なプロセス (コレクター) が各 Automation Controller 仮想マシンインスタンスと Automation Hub 仮想マシンインスタンスで実行し、収集された Ansible メトリックと Podman メトリックを Google Cloud Platform Monitoring にエクスポートします。Google Ops Agent は Google Cloud ソリューションの一部であるため、独自の設定ファイルがあります。

Google Ops エージェントは、ロギングの設定も担当します。

サービス API の **monitoring.googleapis.com** と **logging.googleapis.com** は、モニタリング機能とロギング機能に対してそれぞれ有効にする必要があります。

Configuration

設定ファイルは、各 Automation Controller および Automation Hub が共有するディスク上にあります。ファイル **/aap/bootstrap/config_file_templates/<controller|hub>/monitoring.yml** を変更して、すべてのエクスポーターとエージェントを設定します。

9.2.1. Ansible と podman の設定

Automation Controller または Automation Hub のファイル **/aap/bootstrap/config_file_templates/<controller|hub>/monitoring.yaml** には、Ansible および Podman メトリックを収集して、GCP に送信するための設定が含まれています。

Automation Controller のデフォルト設定は次のようになります。

```
# This value will be set at deployment time.
# Set to zero if monitoringEnabled is false otherwise 59s
# The collection interval for each collector will be the minimum
# between the defaultCollectorInterval and all send Interval
# of a given collector
# NB: The awx exported should not run on controllers as
# it duplicates the number of records sent to GCP Monitoring

defaultCollectorInterval: $DEFAULT_COLLECTOR_INTERVAL
collectors:
- name: podman
 endpoint: http://localhost:9882/podman/metrics
 enabled: true

# list of metrics to exclude
# excludedMetrics:
# - podman_container_created_seconds

 metrics:
 - name: podman_container_exit_code
   # interval on which the metric must be pushed to gcp
   sendInterval: 59s
```

Automation Hub のデフォルト設定は次のようになります。

```
# This value will be set at deployment time.
# Set to zero if monitoringEnabled is false otherwise 59s
# The collection interval for each collector will be the minimum
# between the defaultCollectorInterval and all sendInterval
# of a given collector
# NB: The awx exporter should not run on controllers as
# it duplicates the number of records sent to GCP Monitoring

defaultCollectorInterval: 59s
collectors:
- name: awx
 userName: admin
 endpoint: http://<Controller_LB_IP>/api/v2/metrics/
 enabled: true
 metrics:
 - name: awx_inventories_total
   # interval on which the metric must be pushed to gcp
   sendInterval: 59s
```

```
- name: podman
 endpoint: http://localhost:9882/podman/metrics
 enabled: true

# list of metrics to exclude
#  excludedMetrics:
#  - podman_container_created_seconds

 metrics:
 - name: podman_container_exit_code
   # interval on which the metric must be pushed to gcp
   sendInterval: 59s
```

ここで、**collectors** は、コレクターごとに項目 (**awx** と podman) が 1 つ含まれる設定配列です。

awx コレクターには認証が必要なため、**userName** を **admin** に設定する必要があります。パスワードは secret-manager から取得されます。

エンドポイントは変更しないでください。

defaultCollectorInterval は、エクスポーターがメトリクスエンドポイントから情報を収集して Google Cloud Platform Monitoring に送信するデフォルトの間隔を指定します。

この値を **0** に設定するか、この属性を省略すると、すべてのコレクターが無効になります。

各コレクターは、**enabled** を **true** または **false** に設定することで個別に有効または無効にできます。

コレクターは、ファミリー別にグループ化された使用可能なすべての指標を返しますが、配列 excludeMetrics に名前を追加することで、Google Cloud Platform Monitoring に送信してはならないファミリーを **excludedMetrics** できます。

他のすべてのファミリーメトリクスについては、収集して Google Cloud Platform Monitoring に送信する間隔を指定できます。コレクター間隔は、すべてのファミリーメトリック間隔と **defaultCollectorInterval** の間の最小値です。これにより、Google Cloud Platform Monitoring に送信される指標のセットごとにコレクションが作成されます。

9.2.2. Google クラウドオペレーションエージェントの設定

設定ファイルの詳細は、こちら を参照してください。

設定ファイルは **/etc/google-cloud-ops-agent/config.yml** にあります。

これは、コンポーネントタイプに応じて、共有ディスク **/aap/bootstrap/config_file_templates/controller/gcp-ops-agent-config.yml** または **/aap/bootstrap/config_file_templates/hub/gcp-ops-agent-config.yml** へのシンボリックリンクです。

設定ファイルには、ops エージェントが収集する必要のある情報を指定するレシーバーが複数含まれています。

デプロイ時の Connect Logging と Connect Metrics の選択によって、ファイルに含まれるパイプラインが決まり、どのログと指標が収集されて GCP に送信されるかが決まります。

デプロイ後にさらにパイプラインを追加する必要がある場合は、それらを **/aap/bootstrap/config_file_templates/hub|controller/gcp-ops-agent-config.yml** に挿入できます。

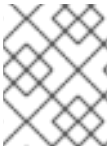
過去 10 分間に **gcp-ops-agent-config.yml** が変更された場合、crontab ジョブはエージェントを再起動します。エージェントは再起動後に設定を再読み込みします。

第10章 バックアップおよび復元

重要

- バックアップと同じ運用イメージバージョンを使用して、復元する必要があります。
- Ansible Automation Platform デプロイメントをバックアップおよび復元するには、既存の Ansible Automation Platform 管理者シークレットの名前と値を安全な場所に記録しておくことが重要です。
- Cloud SQL データベースインスタンスと filestore のバックアップを定期的に手動で作成し、デプロイメントを以前の動作状態にできるだけ近づけて復元できるようにすることも重要です。

Playbook のバックアップおよび復元は、Ansible Automation Platform from GCP Marketplace 基盤デプロイメントのバックアップと復元をサポートします。



注記

復元プロセスでは、新しい Ansible Automation Platform をデプロイし、filestore と SQL データベースインスタンスを指定のバックアップに復元します。

10.1. バックアッププロセス

バックアップは、データベースと共有ファイルシステムを保存することで環境をバックアップできます。保存された共有ファイルシステムを使用して、復元中に新しい環境が作成されます。新しい環境が整うと、プロセスによってデータベースが復元されます。

バックアップと復元のプロセスでは同じバージョンを使用する必要があります。以前のバージョンでバックアップを実行した場合は、そのバージョンの復元プロセスを使用する必要があります。その後、必要に応じてアップグレードを実行できます。

また、アップグレードの前にバックアップを作成する必要があります。詳細は、デプロイメントのアップグレード を参照してください。

バックアッププロセスでは、Cloud SQL データベースと filestore インスタンスを特定の時点でバックアップします。バックアップ Playbook では、アクティブな Ansible Automation Platform from GCP Marketplace 基盤デプロイメントが実行されている必要があります。

復元情報はバケットに保存されるため、プロジェクト内にバケットを作成する必要があります。

バケットには、同じデプロイメントまたは異なるデプロイメントからの複数のバックアップを追加できます。バックアップでは、<prefix>-<deployment_name>-<timestamp> という名前のディレクトリーと、<prefix>-<deployment_name>-<timestamp>.json という名前のファイルが生成されます。このディレクトリーには、awx データベースと pulp データベースのバックアップ、デプロイメント設定およびシークレットが含まれています。json ファイルには、復元手順の情報が含まれています。

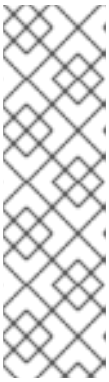
バックアップのリスト表示と削除を行うための Playbook が CLI を通じて提供されます。

以下の手順では、Ansible Automation Platform from GCP Marketplace デプロイメントをバックアップする方法について説明します。

10.1.1. ansible-on-clouds-ops コンテナーイメージのプル

手順

- 基盤デプロイメントと同じタグを持つ **ansible-on-clouds-ops** コンテナーの Docker イメージをプルします。デプロイしたバージョンが不明な場合は、Ansible on Clouds デプロイメントの現在のバージョンを確認する方法の詳細について、コマンドジェネレーター と Playbook **gcp_get_aoc_version** を参照してください。



注記

Docker イメージをプルする前に、Docker を使用して registry.redhat.io にログインしていることを確認してください。以下のコマンドを使用して registry.redhat.io にログインします。

```
$ docker login registry.redhat.io
```

レジストリーのログインに関する詳細は、Registry Authentication を参照してください。

```
$ export IMAGE=registry.redhat.io/ansible-on-clouds/ansible-on-clouds-ops-rhel9:2.4.20240215
$ docker pull $IMAGE --platform=linux/amd64
```

注記

基盤のデプロイメントバージョンが 2.4.20240215-00 ではない場合は、Released versions ページの表で、**Ansible on Clouds バージョン** 列にある一致するデプロイメントバージョンを参照してください。**Ansible-on-clouds-ops コンテナーイメージ列の、IMAGE 環境変数の対応する運用イメージ** を見つけます。

10.1.2. 必須のパーミッション

スタックのバックアップを作成するには、以下の GCP IAM 権限が必要です。

```
required-roles:

Service Account User
Compute Instance Admin (v1)
required-permissions:

compute.instances.list
deploymentmanager.deployments.get
deploymentmanager.manifests.get
deploymentmanager.manifests.list
deploymentmanager.resources.list
file.backups.create
file.operations.get
iap.tunnelInstances.accessViaIAP
storage.objects.create
storage.objects.list
```

10.1.3. 環境の設定

手順

- 設定ファイルを保存するフォルダーを作成します。

```
$ mkdir command_generator_data
```

10.1.4. バックアップ要件

Ansible on Clouds デプロイメントのバックアップを保存するバケットを作成する必要があります。バケットには、バックアップ情報、シークレット、データベースのバックアップが含まれています。

手順

1. Google Cloud コンソールで、**Cloud Storage→ Buckets** に移動します
2. プロジェクトを選択します。
3. **Create** をクリックします。
4. 名前を入力します。
5. 要件に最も適合するデータの場所を入力します。マルチリージョンおよびデュアルリージョンを使用すると、別のリージョンへの復元を実行できます。
6. 要件に最も適合するストレージクラスを入力します。
7. 要件に最も適合する制御アクセスを入力します。
8. 要件に最も適合するデータ保護を入力します。
9. **Create**をクリックします。

トラブルシューティング

バケット名が別のプロジェクトですでに使用されている場合は、エラーが発生します。

10.1.5. バックアップデータファイルの作成

手順

1. コマンドジェネレーター **command_generator_vars** を実行して、**backup.yml** を生成します。

注記

Linux では、コマンドジェネレーターが作成したファイルまたはディレクトリーは、デフォルトで **root:root** の所有となります。ファイルとディレクトリーの所有権を変更するには、ファイルの作成後に **sudo chmod** コマンドを実行します。詳細は、コマンドジェネレーター - root が所有する Linux ファイル を参照してください。

```
docker run --rm -v $(pwd)/command_generator_data/:/data $IMAGE
command_generator_vars gcp_backup_deployment --output-data-file /data/backup.yml
```

2. コマンドを実行すると、**$(pwd)/command_generator_data/backup.yml** テンプレートファイルが作成されます。このテンプレートファイルは以下のようになります。

```
gcp_backup_deployment:
  cloud_credentials_path:
  deployment_name:
  extra_vars:
    backup_prefix: aoc-backup
    gcp_bucket_backup_name:
    gcp_compute_region:
    gcp_compute_zone:
```

10.1.6. backup.yml ファイル内のパラメーター

バックアップを開始する前に、データファイルを設定する必要があります。次の変数は、データファイルにリストされているパラメーターです。

- **cloud_credentials_path** は、Google Cloud サービスアカウントの認証情報ファイルのパスです。これは、絶対パス名である必要があります。
- **deployment_name** は、バックアップを作成する AAP デプロイメントマネージャーのデプロイメント名です。
- **backup_prefix** は、バックアップ名に追加する接頭辞です (デフォルト: aoc-backup)。
- **gcp_bucket_backup_name** は、バックアップに使用するために以前に作成したバケットです。
- **gcp_compute_region** は、基盤デプロイメントがデプロイされている GCP リージョンです。これは、Deployment Manager の Deployments 設定を確認することで取得できます。
- **gcp_compute_zone** は、基盤デプロイメントがデプロイされる GCP ゾーンです。これは、Deployment Manager の Deployments 設定を確認することで取得できます。

10.1.7. バックアップ Playbook の実行

手順

1. バックアップを実行するには、コマンドジェネレーターを実行してバックアップコマンドを生成します。

```
docker run --rm -v $(pwd)/command_generator_data:/data $IMAGE command_generator
gcp_backup_deployment --data-file /data/backup.yml
```

その結果、次の出力が得られます。

```
------------------------------------------------
Command to run playbook:

docker run --rm --env PLATFORM=GCP -v  </path/to/gcp/service-
account.json>:/home/runner/.gcp/credentials:ro \
--env ANSIBLE_CONFIG=../gcp-ansible.cfg --env DEPLOYMENT_NAME=
<deployment_name --env GENERATE_INVENTORY=true  \
$IMAGE redhat.ansible_on_clouds.gcp_backup_deployment \
```

```
-e 'gcp_service_account_credentials_json_path=/home/runner/.gcp/credentials  \
gcp_deployment_name=<deployment_name> gcp_compute_region=<region>
gcp_compute_zone=<zone> \
gcp_bucket_backup_name=<bucket> backup_prefix=aoc-backup'
```

2. 提供されたバックアップコマンドを実行して、バックアップをトリガーします。

```
$ docker run --rm --env PLATFORM=GCP -v  </path/to/gcp/service-
account.json>:/home/runner/.gcp/credentials:ro \
--env ANSIBLE_CONFIG=../gcp-ansible.cfg --env DEPLOYMENT_NAME=
<deployment_name --env GENERATE_INVENTORY=true  \
$IMAGE redhat.ansible_on_clouds.gcp_backup_deployment \
-e 'gcp_service_account_credentials_json_path=/home/runner/.gcp/credentials  \
gcp_deployment_name=<deployment_name> gcp_compute_region=<region>
gcp_compute_zone=<zone> \
gcp_bucket_backup_name=<bucket> backup_prefix=aoc-backup'
```

3. Playbook の実行が完了すると、出力は次のようになります。

```
TASK [redhat.ansible_on_clouds.standalone_gcp_backup : [backup_deployment] Print the
variable required to restore deployment my-deployment] ***
ok: [localhost] => {
    "msg": [
        "AAP on GCP Backup successful. Please note below the bucket name and backup
name which are required for restore process.",
        "gcp_bucket_backup_name: my-bucket",
        "backup_name: aoc-backup-my-deployment-20230616T134002"
    ]
}

PLAY RECAP *********************************************************************
localhost                  : ok=38   changed=6    unreachable=0    failed=0    skipped=1
rescued=0    ignored=0
```

10.1.8. バックアップのリスト表示

この Playbook を使用すると、特定のバケット内の既存のバックアップをリスト表示できます。

手順

1. **command_generator_data** ディレクトリーに設定ファイルのテンプレートを追加します。

注記

Linux では、コマンドジェネレーターが作成したファイルまたはディレクトリーは、デフォルトで **root:root** の所有となります。ファイルとディレクトリーの所有権を変更するには、ファイルの作成後に **sudo chmod** コマンドを実行します。詳細は、テクニカルノート を参照してください。

```
docker run --rm -v $(pwd)/command_generator_data/:/data $IMAGE
command_generator_vars gcp_backup_list --output-data-file /data/backups_list.yml
```

2. コマンドを実行すると、**$(pwd)/command_generator_data/backups_list.yml** テンプレートファイルが作成されます。このテンプレートファイルは以下のようになります。

```
gcp_backup_list:
  cloud_credentials_path:
  extra_vars:
    gcp_bucket_backup_name:
```

3. バックアップを実行するには、コマンドジェネレーターを実行してバックアップコマンドを生成します。

```
docker run --rm -v $(pwd)/command_generator_data:/data $IMAGE command_generator
gcp_backup_list --data-file /data/backups_list.yml
```

その結果、次の出力が得られます。

```
------------------------------------------------
Command to run playbook:

docker run --rm --env PLATFORM=GCP -v </path/to/gcp/service-
account.json>:/home/runner/.gcp/credentials:ro \
--env ANSIBLE_CONFIG=../gcp-ansible.cfg  $IMAGE
redhat.ansible_on_clouds.gcp_backup_list \
-e 'gcp_service_account_credentials_json_path=/home/runner/.gcp/credentials
gcp_bucket_backup_name=<bucket>'
```

4. 提供されたバックアップコマンドを実行して、バックアップのリストをトリガーします。

5. Playbook の実行が完了すると、出力は次のようになります。

```
TASK [redhat.ansible_on_clouds.standalone_gcp_backup_list : [list_backup] Display list of
backups] ***
ok: [localhost] => {
    "msg": [
        "aoc-backup-deployment1-20230614T203926",
        "aoc-backup-deployment1-20230616T114134",
        "aoc-backup-deployment1-20230616T134002",
        "aoc-backup-deployment2-20230613T124127"
    ]
}

PLAY RECAP *********************************************************************
localhost                  : ok=11   changed=0    unreachable=0    failed=0    skipped=0
rescued=0    ignored=0
```

10.1.9. バックアップを削除する

バックアップを削除する 2 つの Playbook があります。

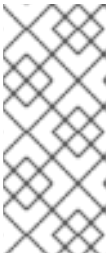
- バックアップを 1 つ削除する **gcp_backup_delete** Playbook を使用します。
- 複数のバックアップを一度に削除する **gcp_backups_delete** Playbook を使用します。

gcp_backups_delete は文字列の配列 ["backup1","backup2",...] を取りますが、 **gcp_backup_delete** は特定のバックアップの名前である 1 つの文字列のみ (backup1) を取ります。

gcp_backups_delete の使用方法は、このセクションで説明します。

手順

1. **command_generator_data** ディレクトリーに設定ファイルのテンプレートを追加します。



注記

Linux では、コマンドジェネレーターが作成したファイルまたはディレクトリーは、デフォルトで **root:root** の所有となります。ファイルとディレクトリーの所有権を変更するには、ファイルの作成後に **sudo chmod** コマンドを実行します。詳細は、コマンドジェネレーター - root が所有する Linux ファイル を参照してください。

```
docker run --rm -v $(pwd)/command_generator_data/:/data $IMAGE
command_generator_vars gcp_backups_delete --output-data-file /data/backups_delete.yml
```

2. コマンドを実行すると、**$(pwd)/command_generator_data/backups_delete.yml** テンプレートファイルが作成されます。このテンプレートファイルは以下のようになります。

```
gcp_backups_delete:
  cloud_credentials_path:
  extra_vars:
    backup_names:
    delete:
    gcp_bucket_backup_name:
```

backup_names パラメーターには、**["backup1","backup2"]** などの文字列の配列を指定する必要があります。正常に削除するには、**delete** パラメーターを **true** に設定する必要があります。

1. バックアップを削除するには、コマンドジェネレーターを実行して **gcp_backups_delete`** コマンドを生成します。

```
docker run --rm -v $(pwd)/command_generator_data:/data $IMAGE command_generator
gcp_backups_delete --data-file /data/backups_delete.yml
```

その結果、次の出力が得られます。

```
-----------------------------------------------
Command to run playbook:

docker run --rm --env PLATFORM=GCP -v </path/to/gcp/service-
account.json>:/home/runner/.gcp/credentials:ro \
--env ANSIBLE_CONFIG=../gcp-ansible.cfg  $IMAGE
redhat.ansible_on_clouds.gcp_backups_delete \
-e 'gcp_service_account_credentials_json_path=/home/runner/.gcp/credentials
gcp_bucket_backup_name=<bucket> \
backup_names=<backup_names> delete=True'
```

2. 提供されたバックアップコマンドを実行してバックアップを削除します。

3. Playbook の実行が完了すると、出力は次のようになります。

```
TASK [redhat.ansible_on_clouds.standalone_gcp_backup_delete : [delete_backup] Dry-run
message] ***
skipping: [localhost]

PLAY RECAP *********************************************************************
localhost                  : ok=23   changed=2    unreachable=0    failed=0    skipped=2
rescued=0    ignored=0
```

10.1.10. 失敗したバックアップ削除の修正

バックアップの削除に失敗した場合は、次のアクションを実行します。

手順

1. バックアップを含むバケットに移動します。
2. バックアップの名前を持つディレクトリーを見つけます。
3. バックアップディレクトリーを開きます。
4. バックアップの名前のディレクトリーを削除します。
5. 拡張子が **.json** のバックアップ名を持つファイルを削除します。
6. **Filestore → Backup** に移動します。
7. バックアップと同じ名前の Filestore バックアップを削除します。

10.2. 復元プロセス

復元プロセスでは、新しいデプロイメントが作成され、filestore と SQL データベースインスタンスが、指定されたバックアップに復元されます。

重要

- 復元には、バックアップに使用したものと同じ運用イメージバージョンを使用する必要があります。

以下の手順では、Ansible Automation Platform from GCP Marketplace デプロイメントを復元する方法を説明します。

10.2.1. ansible-on-clouds-ops コンテナーイメージのプル

手順

- バックアップが作成されたタグとタグが同じ **ansible-on-clouds-ops** コンテナーの docker イメージをプルします。バックアップされたバージョンが不明な場合は、バックアップ json ファイルでキー **liststore_properties.aoc_version** を確認できます。

注記

Docker イメージをプルする前に、Docker を使用して registry.redhat.io にログインしていることを確認してください。以下のコマンドを使用して registry.redhat.io にログインします。

```
$ docker login registry.redhat.io
```

レジストリーのログインに関する詳細は、Registry Authentication を参照してください。

```
$ export IMAGE=registry.redhat.io/ansible-on-clouds/ansible-on-clouds-ops-rhel9:2.4.20240215
$ docker pull $IMAGE --platform=linux/amd64
```

注記

バックアップバージョンが 2.4.20240215-00 ではない場合は、Released versions ページの表で、**Ansible on Clouds version** 列にある一致するバージョンを参照してください。**Ansible-on-clouds-ops コンテナーイメージ列の、IMAGE 環境変数の対応する運用イメージ** を見つけます。

10.2.2. 環境の設定

手順

- 設定ファイルを保存するフォルダーを作成します。

```
$ mkdir command_generator_data
```

10.2.3. 必須のパーミッション

スタックを復元するには、次の GCP IAM 権限が必要です。

```
required-roles:

Cloud SQL Client
Cloud SQL Instance User
Compute Instance Admin (v1)
Compute Network Admin
Editor
Logs Writer
Secret Manager Secret Accessor
Service Account User
required-permissions:

compute.instances.list
compute.networks.create
deploymentmanager.deployments.create
deploymentmanager.deployments.get
deploymentmanager.operations.get
file.instances.create
```

```
file.operations.get
iap.tunnelInstances.accessViaIAP
secretmanager.secrets.create
secretmanager.secrets.delete
secretmanager.secrets.get
secretmanager.secrets.update
secretmanager.versions.add
secretmanager.versions.list
storage.objects.get
storage.objects.list
```

10.2.4. restore.yml ファイルの生成

手順

1. コマンドジェネレーター **command_generator_vars** を実行して、**restore.yml** を生成します。

注記

Linux では、コマンドジェネレーターが作成したファイルまたはディレクトリーは、デフォルトで **root:root** の所有となります。ファイルとディレクトリーの所有権を変更するには、ファイルの作成後に **sudo chmod** コマンドを実行します。詳細は、テクニカルノート を参照してください。

```
docker run --rm -v $(pwd)/command_generator_data:/data $IMAGE
command_generator_vars gcp_restore_deployment --output-data-file /data/restore.yml
```

2. コマンドを実行すると、**$(pwd)/command_generator_data/restore.yml** テンプレートファイルが作成されます。このテンプレートファイルは以下のようになります。

```
==============================================
Playbook: gcp_restore_deployment
Description: This playbook is used to restore the Ansible Automation Platform from GCP
Marketplace environment from a backup.
----------------------------------------------
This playbook is used to restore the Ansible Automation Platform from GCP Marketplace
environment from a backup.
For more information regarding backup and restore, visit our official documentation -
https://access.redhat.com/documentation/ja-
jp/ansible_on_clouds/2.x/html/red_hat_ansible_automation_platform_from_gcp_marketplace_g
uide/index

----------------------------------------------
Command generator template:

docker run --rm -v <local_data_file_directory>:/data $IMAGE command_generator
gcp_restore_deployment --data-file /data/restore.yml
```

テンプレートは以下のようになります。

```
gcp_restore_deployment:
  cloud_credentials_path:
  deployment_name:
```

```
extra_vars:
  backup_name:
  gcp_bucket_backup_name:
  gcp_cloud_sql_peering_network:
  gcp_compute_region:
  gcp_compute_zone:
  gcp_controller_internal_ip_address:
  gcp_existing_vpc:
  gcp_filestore_ip_range:
  gcp_hub_internal_ip_address:
```

10.2.5. restore.yml ファイルのパラメーター

必ず新しい VPC ネットワークに復元する必要があります。

新しい VPC の場合

新しい VPC を使用して復元する場合は、以下のパラメーターを設定します。

- **gcp_existing_vpc** は、**false** に設定する必要があります。

以下のパラメーターを削除する必要があります。

- **gcp_filestore_ip_range**
- **gcp_cloud_sql_peering_network**
- **gcp_controller_internal_ip_address**
- **gcp_hub_internal_ip_address**

以下のパラメーターの値を指定します。

- **gcp_existing_vpc** は、**false** に設定する必要があります。
- **cloud_credentials_path** は、Google Cloud サービスアカウントの認証情報ファイルのパスです。
- **deployment_name** は、デプロイメントを復元する必要がある名前です。新しいデプロイメントはこの名前で作成されます。同じ名前のデプロイメントがすでに存在していると、その名前は使用できません。
- **backup_name** はバケット内のバックアップの名前です。この名前は、**gcp_backup_deployment** コマンドまたは **gcp_backup_list** コマンドの使用中に表示されます。
- **gcp_bucket_backup_name** は、バックアップに使用したバケットの名前です。
- **gcp_compute_region** は、バックアップが作成されたリージョンです。これは、Deployment Manager の Deployments 設定を確認することで取得できます。
- **gcp_compute_zone** は、バックアップが作成されたゾーンです。これは、Deployment Manager の Deployments 設定を確認することで取得できます。

既存の VPC の場合

既存の VPC を使用して復元する場合は、上記のパラメーターを設定する必要があります。

以下の追加パラメーターも設定する必要があります。

- **gcp_existing_vpc** は **true** に設定されます。
- **gcp_filestore_ip_range** は、VPC の空き ip/29 範囲に設定する必要があります。例: 192.168.245.0/29。192.168.243.0/29 は、Ansible Automation Platform from GCP Marketplace をデプロイするときに使用されるデフォルトであるため、使用しないでください。
- **gcp_cloud_sql_peering_network** は、空きの **/24** サブネットに設定する必要があります。192.168.241.0/24 は、元のデプロイメント時に使用されるため、使用しないでください。
- **gcp_controller_internal_ip_address** は、VPC ネットワーク内の空き IP に設定する必要があります。
- **gcp_hub_internal_ip_address** は、VPC ネットワーク内の空き IP に設定する必要があります。

10.2.6. 復元コマンドの実行

$(pwd)/command_generator_data/restore.yml が設定されている場合は、コマンドジェネレーターを使用して復元コマンドを作成できます。

手順

1. コマンドジェネレーターを実行します。

```
$ docker run --rm -v $(pwd)/command_generator_data:/data $IMAGE command_generator
gcp_restore_deployment --data-file /data/restore.yml
```

これにより、必要なすべてのボリューム、環境変数、パラメーターを含む新しいコマンドが生成されます。

生成されたコマンドは以下のようになります。

```
docker run --rm --env PLATFORM=GCP -v
<local_credential_file>:/home/runner/.gcp/credentials:ro \
--env ANSIBLE_CONFIG=../gcp-ansible.cfg --env DEPLOYMENT_NAME=
<deployment_name> --env GENERATE_INVENTORY=true -\
-env CHECK_GENERATED_INVENTORY=false  $IMAGE
redhat.ansible_on_clouds.gcp_restore_deployment \
-e 'gcp_service_account_credentials_json_path=/home/runner/.gcp/credentials  \
gcp_deployment_name=<deployment_name> gcp_compute_region=<region>
gcp_compute_zone=<zone> \
gcp_bucket_backup_name=<bucket> backup_name=<backup_name> gcp_existing_vpc=
<existing_vpc>'
```

2. 生成されたコマンドを実行します。

```
$ docker run --rm --env PLATFORM=GCP -v
<local_credential_file>:/home/runner/.gcp/credentials:ro \
--env ANSIBLE_CONFIG=../gcp-ansible.cfg  $IMAGE
redhat.ansible_on_clouds.gcp_restore_deployment \
-e 'gcp_service_account_credentials_json_path=/home/runner/.gcp/credentials  \
```

```
gcp_deployment_name=<former_deployment_name> gcp_restored_deployment_name=
<new_deployment_name> \
gcp_compute_region=<region> gcp_compute_zone=<zone> gcp_bucket_backup_name=
<bucket> gcp_existing_vpc=False'
```

3. Playbook が完了すると、出力は以下のようになります。

```
TASK [redhat.ansible_on_clouds.standalone_gcp_restore : Display internal IP addresses] ***
ok: [localhost] =>
  msg:
  - 'Hub        internal IP: 192.168.240.21'
  - 'Controller internal IP: 192.168.240.20'

PLAY RECAP *********************************************************************
localhost                  : ok=33   changed=8    unreachable=0    failed=0    skipped=6
rescued=0    ignored=2
```

10.2.7. 復元の失敗

復元中に次のようなメッセージが表示された場合は、復元を手動で行う必要があるため、サポートに連絡する必要があります。

```
TASK [redhat.ansible_on_clouds.standalone_gcp_restore : [restore_deployment] Restore awx db] *
fatal: [localhost -> dvernier-restore1-aap-cntrlr-x2c6]: FAILED!
```

第11章 アップグレード

既存の Ansible Automation Platform デプロイメントを新しいバージョンにアップグレードできます。アップグレードプロセスには、Automation Hub、Automation Controller、および拡張ノードのアップグレードが含まれます。アップグレードプロセスには、Ansible Automation Platform デプロイメントのインストールとほぼ同じだけ時間がかかります。アップグレードを実行する前にバックアップを作成する必要があります。

注記

GCP Marketplace で入手可能な Ansible Automation Platform は、順次アップグレードをサポートしています。アップグレードはすべて、アップグレード先のバージョンよりも最大で 1 つ前のメジャーバージョンである必要があります。たとえば、Ansible Automation Platform を 2.4.20240215 にアップグレードするには、バージョン 2.4.20231024 を使用している必要があります。

アップグレードを実行する前に、ロギングとモニタリングを無効にする必要があります。アップグレードを開始する前に、次の 手順 に従って現在のバージョンのモニタリングとロギングをオフに切り替えます。

アップグレードが完了したら、次の 手順 に従ってロギングとモニタリングを再度有効にすることができます。

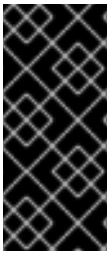
前提条件

- Docker をインストールしてアップグレード Playbook を実行できるようにする。
- Linux または macOS システム (**ansible-on-clouds-ops** コンテナーイメージが実行される場所)
- アップグレードプロセスでは、複数のボリュームをマウントする必要があるので、このプロセスに使用する新しいディレクトリーを準備する。

アップグレードプロセスは次の手順で行います。

1. Ansible Automation Platform スタックをバックアップします。
   - Ansible on Clouds のバックアップ手順 に従います。
2. Ansible Automation Platform のアップグレード
   a. 次の連続する ansible-on-clouds-ops バージョンコンテナーイメージをプルする
   b. 最低限必要な権限が満たされていることを確認する
   c. データファイルを生成する
   d. データファイルを更新する
   e. 運用コンテナーを実行して、Ansible Automation Platform のアップグレードを開始する
3. (任意) バックアップからスタックを復元します。
   - Ansible on Clouds の復元手順 に従います。

11.1. アップグレード前のバックアップ



重要

Ansible Automation Platform 環境のアップグレードを開始する前に、まず現在のバージョンで環境のバックアップを作成する必要があります。

以前のバージョンで Ansible Automation Platform 環境のバックアップを作成するには、Ansible on Clouds のバックアップ手順 に従ってください。

11.2. ANSIBLE AUTOMATION PLATFORM のアップグレード

11.2.1. ansible-on-clouds-ops コンテナーイメージのプル

手順

- アップグレード先のバージョンとタグが同じ Ansible on Clouds 運用コンテナーの Docker イメージをプルします。デプロイ済みのバージョンとアップグレード先のバージョンが不明な場合は、Ansible on Clouds デプロイメントの現在のバージョンとアップグレード先のバージョンを見つける方法の詳細は、Command Generator と Playbook **gcp_get_aoc_version** を参照してください。

注記

Docker イメージをプルする前に、Docker を使用して registry.redhat.io にログインしていることを確認してください。以下のコマンドを使用して registry.redhat.io にログインします。

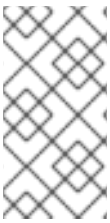
```
$ docker login registry.redhat.io
```

レジストリーのログインに関する詳細は、Registry Authentication を参照してください。

注記

Ansible on Clouds の運用イメージのタグが、アップグレード先のバージョンと一致している必要があります。たとえば、基盤デプロイメントのバージョンが 2.4.20231024 の場合は、タグが 2.4.20240215 の運用イメージをプルして、バージョン 2.4.20240215 にアップグレードします。

```
$ export IMAGE=registry.redhat.io/ansible-on-clouds/ansible-on-clouds-ops-
rhel9:2.4.20240215
$ docker pull $IMAGE --platform=linux/amd64
```



注記

基盤のデプロイメントバージョンが 2.4.20240215-00 ではない場合は、Released versions ページの表で、**Upgrade from version** 列にある一致するデプロイメントバージョンを参照してください。**Ansible-on-clouds-ops コンテナーイメージ列の、IMAGE 環境変数の対応する運用イメージ** を見つけます。

11.2.2. 必須のパーミッション

スタックをアップグレードするには、次の GCP IAM パーミッションが必要です。

```
required-roles:
  - Service Account User
  - Compute Instance Admin (v1)
required-permissions:
  - compute.healthChecks.update
  - compute.healthChecks.use
  - compute.healthChecks.useReadOnly
  - compute.regionBackendServices.update
  - iap.tunnelInstances.accessViaIAP
  - runtimeconfig.variables.get
  - secretmanager.locations.get
  - secretmanager.locations.list
  - secretmanager.secrets.create
  - secretmanager.secrets.delete
  - secretmanager.secrets.get
  - secretmanager.secrets.list
  - secretmanager.secrets.update
  - secretmanager.versions.access
  - secretmanager.versions.add
  - secretmanager.versions.disable
  - secretmanager.versions.enable
  - secretmanager.versions.get
  - secretmanager.versions.list
```

11.2.3. データファイルの生成

このセクションのコマンドは、ディレクトリーを作成し、そのディレクトリーに空のデータテンプレートを設定します。このテンプレートは、設定するとアップグレード中に使用されます。

手順

1. 次のコマンドを実行して、必要なデータファイルを生成します。

```
# Create a folder to hold the configuration files
$ mkdir command_generator_data
```

2. **command_generator_data** フォルダーに設定ファイルのテンプレートを追加します。

注記

Linux では、コマンドジェネレーターが作成したファイルまたはディレクトリーは、デフォルトで **root:root** の所有となります。ファイルとディレクトリーの所有権を変更するには、ファイルの作成後に **sudo chmod** コマンドを実行します。詳細は、コマンドジェネレーター - root が所有する Linux ファイル を参照してください。

```
$ docker run --rm -v $(pwd)/command_generator_data:/data $IMAGE \
command_generator_vars gcp_upgrade \
--output-data-file /data/extra_vars.yml
```

3. これらのコマンドを実行すると、**command_generator_data/extra_vars.yml** テンプレートファイルが作成されます。このテンプレートファイルは以下のようになります。

```
gcp_upgrade:
  ansible_config_path:
  cloud_credentials_path:
  deployment_name:
  extra_vars:
    gcp_backup_taken:
    gcp_compute_region:
    gcp_compute_zone:
```

11.2.4. データファイルの更新

アップグレードを開始する前に、データファイルを設定する必要があります。任意と記載されていない限り、各パラメーターは必須です。パラメーターが任意と指定されている場合は、そのキーと値の両方をデータファイルから削除できます。

- **ansible_config_path** (オプション) は、カスタムの **ansible_config** でオーバーライドする場合にのみ使用します。
- **cloud_credentials_path** は、GCP 認証情報ファイルへのパスです。
- **deployment_name** は、基盤デプロイメントの名前です。
  - これは、基盤をデプロイしたときに使用した名前と同じです。
- **gcp_backup_taken** は、このアップグレードを実行する前に、現在のデプロイメントの手動バックアップが最近作成されたことを確認するものです。新しいバックアップが作成されたことを確認するには、ここで **true** を使用します。
- **gcp_compute_region** は、基盤のデプロイメントをデプロイする時に指定した GCP リージョンです。基盤のデプロイ時にリージョンを指定していない場合は、ここでデフォルトのリージョン **us-east1** を使用します。
- **gcp_compute_zone** は、基盤のデプロイメントをデプロイする時に指定した GCP ゾーンです。基盤のデプロイ時にゾーンを指定していない場合は、ここでデフォルトの **us-east1-b** を使用します。

データファイルを入力すると、次のようになります。

次の値は例です。

```
gcp_upgrade:
  ansible_config_path:
  cloud_credentials_path: ~/secrets/GCP-secrets.json
  deployment_name: AnsibleAutomationPlatform
  extra_vars:
    gcp_backup_taken: true
    gcp_compute_region: us-east1
    gcp_compute_zone: us-east1-b
```

11.2.5. アップグレード Playbook の実行

注記

Ansible Automation Platform を 2.4.20240215 にアップグレードすると、内部ロードバランサーのプロトコルが更新されます。インストール後に追加のネットワーク設定が指定された場合は、接続を確保するために更新が必要になる場合もあります。詳細は、アップグレードに関する注意事項 を参照してください。

1. アップグレードを実行するために、コマンドジェネレーターを実行してアップグレード CLI コマンドを生成します。

```
$ docker run --rm -v $(pwd)/command_generator_data:/data $IMAGE command_generator --
data-file /data/extra_vars.yml
```

これにより、以下のコマンドが生成されます。

```
-----------------------------------------------
docker run --rm --env PLATFORM=GCP -v ~/secrets/GCP-
secrets.json:/home/runner/.gcp/credentials:ro
--env ANSIBLE_CONFIG=../gcp-ansible.cfg
--env DEPLOYMENT_NAME=AnsibleAutomationPlatform
--env GENERATE_INVENTORY=true  $IMAGE redhat.ansible_on_clouds.gcp_upgrade \
-e 'gcp_deployment_name=AnsibleAutomationPlatform \
gcp_service_account_credentials_json_path=/home/runner/.gcp/credentials \
gcp_compute_region=us-east1 gcp_compute_zone=us-east1-b gcp_backup_taken=True'
===============================================
```

2. 生成されたアップグレードコマンドを実行して、アップグレードをトリガーします。

```
$ docker run --rm --env PLATFORM=GCP -v ~/secrets/GCP-
secrets.json:/home/runner/.gcp/credentials:ro \
--env ANSIBLE_CONFIG=../gcp-ansible.cfg \
--env DEPLOYMENT_NAME=AnsibleAutomationPlatform \
--env GENERATE_INVENTORY=true  $IMAGE redhat.ansible_on_clouds.gcp_upgrade \
-e 'gcp_deployment_name=AnsibleAutomationPlatform \
gcp_service_account_credentials_json_path=/home/runner/.gcp/credentials \
gcp_compute_region=us-east1 gcp_compute_zone=us-east1-b gcp_backup_taken=True'
```

3. アップグレードが完了するまでに時間がかかることがありますが、システム上の拡張ノードの数によってはさらに時間がかかる場合があります。アップグレードが成功すると、以下のログが記録されます。

```
TASK [redhat.ansible_on_clouds.standalone_gcp_upgrade : [upgrade] Show GCP current
version] ***
ok: [localhost] => {
    "msg": "gcp_current_version: 2.4.20231024-00"
}
```

4. Ansible Automation Platform from GCP Marketplace デプロイメントが新しいバージョンにアップグレードされ、デプロイメント認証情報を使用して Red Hat Ansible Automation Platform Automation Controller および Automation Hub にログインできるようになりました。

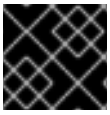
11.3. バックアップからの復元

以前のバージョンの Ansible Automation Platform 環境をアップグレード前の状態に復元する場合は、Ansible on Clouds 2.3 の復元手順 に従ってください。

第12章 アンインストール

この章では、Ansible Automation Platform をアンインストールする方法を説明します。

12.1. 拡張ノードの削除



重要

この操作を元に戻すことはできません。

拡張ノードは、GCP Deployment Mananger でデプロイメントとして作成されます。そのため、Ansible Automation Platform 拡張ノードを削除するプロセスは、メインの Ansible Automation Platform デプロイメントを削除するプロセスと同じです。

手順

1. メインメニューを選択します。
2. **Deployment Manager** を選択します。**Deployment Manager** が表示されない場合は、**View All Products** を選択します。
3. 削除する拡張ノードのチェックボックスをオンにします。

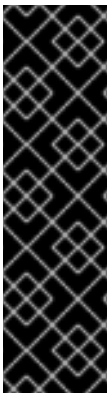
注記

拡張ノードとメインのデプロイメントは両方とも **Deployment Manager** にあります。メインのデプロイメントではなく、拡張ノードを必ず選択してください。拡張ノードの名前は、その作成者によって選択されたものです。

4. 上部のバーにある **Delete** をクリックします。この操作を元に戻すことはできません。

拡張ノードが削除されます。

12.2. ANSIBLE AUTOMATION PLATFORM のアンインストール



重要

この操作を元に戻すことはできません。

デプロイメント用に外部ロードバランサーが追加されている場合は、デプロイメントをアンインストールする前に削除する必要があります。詳細は、Load Balancers を参照してください。

拡張ノードがデプロイメントに追加されている場合は、デプロイメントをアンインストールする前に拡張ノードも削除する必要があります。拡張ノードの削除 の手順に従ってください。

手順

1. 左上からメインメニューを選択します。

2. **Deployment Manager** を選択します。**Deployment Manager** が表示されない場合は、**View All Products** を選択します。
3. 削除するデプロイメントの前にあるチェックボックスをオンにします。
4. 上部のバーにある **Delete** をクリックします。この操作を元に戻すことはできません。

削除が完了するまでに時間がかかる場合があります。

12.3. アップグレードリソースの削除

次の手順では、アップグレード後に残ったリソースを削除する方法を説明します。

注記

これらのアクションを元に戻すことはできません。

これらのアクションは、デプロイメントが削除されている場合にのみ実行する必要があります。

12.3.1. シークレットの削除

注記

この操作を元に戻すことはできません。

この操作は、デプロイメントを削除した場合にのみ実行してください。

完了した手順に応じて、検索できるシークレット名がいくつかあります。

- **<deployment_name>**-aap-secret-key
- **<deployment_name>**-aap-admin
- **<deployment_name>**-container_auth_private_key_pem
- **<deployment_name>**-container_auth_public_key_pem
- **<deployment_name>**-database_fields_symmetric_key
- **<deployment_name>**-pulp_cert
- **<deployment_name>**-pulp_key

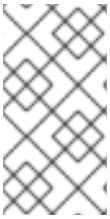
手順

1. 左上からメインメニューを選択します。
2. **Security** を選択します。**Security** が表示されない場合は、**View All Products** を選択します。
3. **Secret Manager** を選択します。
4. 上記のリスト内の名前を探して、デプロイメント名を検索します。
5. デプロイメントの行にある **More Actions** アイコン ⋮ をクリックします。

6. **Delete** を選択します。管理パスワードが表示されます。
7. シークレットの名前を入力します。
8. **Delete Secret** をクリックします。

12.4. バックアップリソースの削除

次の手順では、Ansible Automation Platform デプロイメントのバックアップを実行した後、アンインストールするときに残ったリソースを削除する方法を説明します。



注記

これらのアクションを元に戻すことはできません。

これらのアクションは、デプロイメントが削除されている場合にのみ実行する必要があります。

12.4.1. ファイルストアの削除

注記

この操作を元に戻すことはできません。

1. 手順
2. 左上からメインメニューを選択します。
3. **Filestore** を選択します。**Filestore** が表示されない場合は、**View All Products** を選択します。
4. **Backups** を選択します。
5. バックアップ名の形式は、**<DeploymentName>-filestore-<RandomSufix>** です。この名前を使用してフィルタリングします。
6. デプロイメントの行にある **More Actions** アイコン ⋮ をクリックします。
7. **Delete** を選択します。
8. 管理パスワードが表示されます。
9. シークレットの名前を入力します。
10. **Delete Backup** をクリックします。

12.4.2. ストレージバケットの削除

手順

1. 左上からメインメニューを選択します。
2. **Cloud Storage** を選択します。**Cloud Storage** が表示されない場合は、**View All Products** を選択します。

3. **Buckets** を選択します。
4. デプロイメント名を検索します。
5. デプロイメントの行にある **More Actions** アイコン ⋮ をクリックします。
6. **Delete** を選択します。
7. 管理パスワードが表示されます。
8. シークレットの名前を入力します。
9. **Delete Secret** をクリックします。

12.5. 残りのリソースの削除

次の手順では、Ansible Automation Platform デプロイメントをアンインストールするときに残ったリソースを削除する方法を説明します。

注記

これらのアクションを元に戻すことはできません。

これらのアクションは、デプロイメントが削除されている場合にのみ実行する必要があります。

12.5.1. サービスアカウントの削除

重要

デプロイメント用に新しいサービスアカウントが作成され、それが他の場所で使用されていない場合は、削除できます。

手順

1. 左上からメインメニューを選択します。
2. **IAM & Admin**を選択します。**IAM & Admin**が表示されない場合は、**View All Products** を選択します。
3. **Service Accounts**を選択します。
4. サービスアカウントの名前でフィルタリングします。
5. サービスアカウントの行にある **More Actions** アイコン ⋮ をクリックします。
6. **Delete** を選択します。
7. 管理パスワードが表示されます。
8. **Delete** をクリックします。

第13章 テクニカルノート

AGCP Marketplace で入手可能な Ansible Automation Platform は self-managed のデプロイメントです。以下は、GCP Marketplace デプロイメントの Ansible Automation Platform に関するテクニカルノートです。

13.1. アップグレード - ロギングとモニタリング

アップグレードを確実に成功させるには、アップグレードを実行する前にロギングとモニタリングを無効にする必要があります。アップグレードを開始する前に、次の 手順 に従って現在のバージョンのモニタリングおよびロギングをオフに切り替えます。アップグレードが完了したら、次の 手順 に従ってロギングとモニタリングを再度有効にできます。

13.2. コマンドジェネレーター - ROOT が所有する LINUX ファイル

Linux では、コマンドジェネレーターが作成したファイルまたはディレクトリーは、デフォルトで **root:root** の所有となります。ファイルとディレクトリーの所有権を変更するには、ファイルの作成後に **sudo chmod** コマンドを実行します。

```
# Change the owner of the command_generator_data directory recursively
$ sudo chown -R $USER:$USER command_generator_data/

# Check the permissions
$ ls -la command_generator_data/
```

コマンドジェネレーターは現在、デーモンモード で Docker CLI を使用することを想定しています。デフォルトの Docker インストールには、Discretionary access control (DAC) のユーザー名前空間マッピングがありません。したがって、**root** によりコンテナーから作成されたファイルは、そのファイルが共有ボリュームに置かれている場合、ホスト上の **root** が所有することになります。

User 名前空間を含む Linux 名前空間の詳細は、The 7 most used Linux namespaces を参照してください。

13.3. アップグレードの注意事項

Ansible Automation Platform を 2.4.20240215 にアップグレードすると、内部ロードバランサーのプロトコルが HTTP から HTTPs に更新されます。追加のネットワーク設定が指定された場合は、接続を確保するために更新することもできます。アップグレードが成功したら、追加されたネットワーク設定を再検証する必要があります。

13.4. ANSIBLE AUTOMATION PLATFORM CONTROLLER API

リクエストが通過するには、Controller の API エンドポイントの末尾にスラッシュが含まれている必要があります。末尾のスラッシュの自動リダイレクトは、GCP Marketplace の Ansible Automation Platform の現在のオファリングではサポートされていません。たとえば、**<controller_base_url>/api/v2/metrics** などのリクエストは、**<controller_base_url>/api/v2/metrics/** の通過中にタイムアウトになります。

13.5. 拡張ノードのメモを削除する

Ansible-on-Clouds 操作 Playbook を使用して拡張ノードを削除する場合は、正しいインスタンスグループ名とインスタンステンプレート名を指定していることを確認してください。インスタンスグループ名とインスタンステンプレート名が間違っていると、孤立リソースが発生します。

13.6. シークレットの更新

GCP シークレットマネージャーでシークレットを更新する場合は、最新のシークレットバージョンが有効になっていることを確認してください。たとえば、**<deployment-name>-aap-admin** シークレットに 2 つのシークレットバージョンがある場合は、シークレットバージョン 2 を有効にする必要があります。**<deployment-name>** は基盤デプロイメントの名前です。

13.7. 仮想マシンの制限事項

仮想マシンを再起動することはできません。代わりに、仮想マシンを置き換える必要があります。仮想マシンを置き換えると設定が失われるため、仮想マシン上で設定を行うことはできません。これはアップグレード時に常に発生します。

第14章 サポート

AGCP Marketplace で入手可能な Ansible Automation Platform は self-managed のデプロイメントです。アプリケーションを GCP インフラストラクチャーにデプロイする場合、お客様は GCP インフラストラクチャーのメンテナンス、OS のパッチ適用、Ansible Automation Platform のパッチ適用を行う責任があります。

Red Hat は、ソリューションの一部としてデプロイされたインフラストラクチャーリソースへの変更はサポートしません。ただし、Red Hat ネットワークのルートテーブルの設定などのプロセスを説明した文書がある場合や、アップグレードプロセスが文書にまとめられている場合を除きます。拡張ノードの外部にコンピューティングリソースを追加すると、GCP Marketplace デプロイメントの Ansible Automation Platform に対する Red Hat のサポートが無効になります。

GCP Marketplace から Ansible Automation Platform へのアップグレードは、自己インストールした Ansible Automation Platform とは異なる方法で実行します。**dnf** またはその他の手段を使用した仮想マシン上の個々のパッケージのアップグレードもサポートされていません。手順は、各バージョンが利用可能になり次第、このドキュメントのアップグレードセクションに、各バージョンの関連コンテンツとともに記載されます。

14.1. サポート対象のインフラストラクチャー設定の変更

Red Hat は、以下のオプションの変更をサポートしています。

- VPC ルート設定
- VPC ファイアウォールの設定
- VPC ロードバランサーの設定
- ブロックストレージ拡張
- DNS および **resolv.conf** ファイル

Red Hat の責任

Red Hat の責任範囲は以下のとおりです。

- Ansible Automation Platform の Premium サポート
- GCP Marketplace から Ansible Automation Platform をアップグレードするための手順とプロセス。
- Red Hat は、Ansible Automation Platform の現在のバージョンをサポートしています。バグ修正と CVE パッチには最新バージョンへのアップグレードが必要です。

お客様の責任

お客様の責任範囲は以下のとおりです。

- GCP インフラストラクチャー起動時間
- GCP インフラストラクチャーの変更 (ブロックストレージサイズの増加など)
- GCP ネットワークピアリングおよび設定
- Ansible Automation Platform のアップグレードの適用

    - オペレーティングシステムのアップグレードが含まれている
- GCP リソースのバックアップ

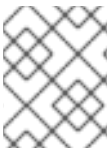
14.2. VM イメージパッケージ

GCP Marketplace で入手可能な Ansible Automation Platform は、Red Hat Ansible チームが作成した VM イメージに基づく仮想マシンを通じて提供されます。このイメージには、Google Cloud Platform で適切に機能し、Red Hat Ansible Automation Platform を実行するために、Red Hat と Google のパッケージが含まれています。このイメージには、次の Google パッケージとコンテナーが含まれています。

パッケージ名	説明
google-osconfig-agent	Google OS 設定エージェントは管理サービスを使用して、一貫した設定、つまり、仮想マシンインスタンスの望ましい状態とソフトウェアをデプロイ、クエリー、および維持します。
google-cloud-ops-agent	Ops エージェントは、Compute Engine インスタンスからテレメトリーを収集するための主要なエージェントです。ロギングとメトリクスを 1 つのエージェントに組み合わせた Ops エージェントは、ログに Fluent Bit を使用して高スループットのロギングをサポートし、OpenTelemetry Collector をメトリクスに使用します。
gce-proxy	Cloud SQL Authorization プロキシーは、仮想マシンサービスアカウントを使用して Ansible Automation Platform コンポーネントを Cloud SQL データベースに接続するために使用されます。
gcloud CLI	gcloud コマンドラインインターフェイスは、Ansible Automation Platform のインストールを設定するために使用されます。

付録A ANSIBLE ON CLOUDS 2.4 のリリースノート

このリリースには、GCP Marketplace から Ansible Automation Platform に実装された多数の拡張機能、追加、修正が含まれています。



注記

現時点では、Automation Mesh と Event Driven Automation は、セルフマネージド型 Ansible on Cloud オファリングではサポートされていません。

機能拡張

GCP Marketplace からの Ansible Automation Platform のリリースバージョン 2.4.20240215 には、次の拡張機能が含まれています。

- さまざまな CVE 修正

非推奨および削除された機能

以前のリリースで利用可能であった一部の機能が非推奨になるか、削除されました。非推奨の機能は依然として Ansible Automation Platform に含まれており、引き続きサポートされますが、この製品の今後のリリースで削除されるため、新規デプロイメントでの使用は推奨されません。以下は、Ansible Automation Platform 2.4 内で非推奨となって削除された主な機能のリストです。

- オンプレミスコンポーネントの Automation Services Catalog は、Ansible Automation Platform 2.4 以降から削除されます。
- Ansible Automation Platform 2.4 リリースでは、Ansible 2.9 の Execution Environment コンテナーイメージ (ee-29-rhel-8) は、デフォルトで Automation Controller 設定にロードされなくなりました。
- 新しい VPC を使用してアプリケーションをデプロイするプロセスは非推奨となり、この機能は今後のリリースで GCP Marketplace の Ansible Automation Platform から削除される予定です。

Ansible Automation Platform に関連するその他のリリースノート

- Red Hat Enterprise Linux 9 の最新機能を確認してください。
- 最新の Ansible Automation Platform 機能 の詳細を参照してください。