Red Hat Decision Manager 7.2

Ansible および Red Hat Decision Manager を使用した ONAP でのポリシーベースのデシジョンおよび制御

ガイド

Last Updated: 2023-02-21

Red Hat Decision Manager 7.2 Ansible および Red Hat Decision Manager を使用した ONAP でのポリシーベースのデシジョンおよび制御

ガイド

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律上の通知

概要

本書は、Open Network Architecture Platform (ONAP) アーキテクチャーで Ansible および Red Hat Decision Manager のテクノロジーを使用する方法を説明します。

目次

はじめに

本書は、Open Network Architecture Platform (ONAP) アーキテクチャーで Ansible および Red Hat Decision Manager のテクノロジーを使用する方法を説明します。ここでは、自動化およびルール管理システムを使用して包括的なネットワークポリシーエンジンを提供する方法を説明します。推奨されるポリシーエンジンでは、必要な自動化を行うことができ、人間が介入することなく、ネットワークやサービスの条件に事前対応できます。これは、クローズドループオートメーションとして知られています。ONAP はオープンソースのネットワークプロジェクトで、物理および仮想のネットワーク機能に、リアルタイムでポリシー駆動型のオーケストレーションおよび自動化のためのプラットフォームを提供します。ONAP では、サービスプロバイダーは新しいサービスを自動化にして、完全なライフサクル管理を迅速に行うことができます。

第1章 技術および定義

1.1. OPEN NETWORK AUTOMATION PLATFORM (ONAP)

Open Network Automation Platform (ONAP) は、物理ネットワーク機能と仮想ネットワーク機能のオーケストレーションおよび自動化機能を提供します。ONAP は、ソフトウェア定義ネットワークや、ネットワーク機能の仮想化の新しいパラダイムを追求することで、新しいサービスオファリングを実装するのに必要なコストや時間を削減します。ONAP には以下の機能が含まれています。

- リソースやその関係をモデル化してサービスを構築する
- ポリシーをもとにサービスのインスタンス化を自動化する
- 分析フレームワークを使用してサービスを監視する
- イベントやポリシーをもとに、サービスや基盤のリソースでライフサイクル管理アクションを自動化する

1.2. ポリシー

ポリシーは、条件、要件、制約、デシジョン、または指定、評価、維持、強制する必要のあるニーズとして定義できます。また、ポリシーは、より低いレベルや機能レベルで定義することも可能です。たとえば、機械で読み込み可能なルール、あるいは特定のタイミングで有効で選択された特定の条件に固有のトリガーと要求をもとに取るべきアクションを有効にするソフトウェア条件やアサーションなどが挙げられます。

以下の表では、ネットワークに影響を与えるポリシーをリストします。

表1.1 ネットワークに影響があるポリシー

ポリシー	説明
Virtual Network Function (VNF) の配置	アフィニティールールなど VNF の配置場所を統括するルール
データおよびフィード管理	収集するデータおよびそのタイミング、保持期間、問題のイベントを送信するタイミングと送信先
アクセス制御	どのデータに誰がアクセスできるか
条件とアクションのトリガー	どの条件がアクション可能かを決定し、その条件で何を実行するかを定義する
対話	変更管理と、障害およびパフォーマンス管理との間の対話を処理する方法

第2章 ルールエンジンおよび RED HAT DECISION MANAGER

ルールエンジンは、宣言形式を使用してビジネスロジックに基づいて意思決定を行うソフトウェア要素です。ルールエンジンには、非常に基本的 (単純なルールのサポート) なものや、非常に複雑なもの (洗練されたアルゴリズムをベースとする) があります。

Red Hat Decision Manager は、PHREAK アルゴリズム (有名な Rete アルゴリズムの進化版)、DMN コンプライアンスレベル 3 ランタイム、PMML のサポート、Complex Event Processing (CEP) エンジンに基づいたルールエンジンなど、柔軟性の高いデシジョン機能を提供します。

PHREAK の理由付けアルゴリズムのランタイムを使用すると、レイテンシーが低く、パフォーマンスの高いビジネスルール実行ができるだけでなく、単一のルール実行環境で、何十万ものルールにまで簡単にスケーリングできます。

軽量のエンジンであるため、埋め込みのシステムやマイクロサービスのアーキテクチャーなど、さまざまなアーキテクチャーやデプロイメントのトポロジーで、デシジョンサービスおよびランタイムとして Red Hat Decision Manager を使用できます。デシジョンエンジンと、複数のアプリケーションフレームワークやランタイムを組み合わせることができます。

Red Hat Decision Manager は、オープンな標準仕様 (DMN、XML、JSON、JAX-RS) および事実上の標準仕様 (Git、Maven、Eclipse、IntelliJ) で構築されています。以下の機能が含まれます。

- 意思決定を自動化する
- 洗練されたデシジョンロジックを容易に埋め込む
- 簡単かつ効率的に実装、管理、監査、変更ができるように、デシジョンロジックとプログラムコードを分離して、単純かつ宣言形式で、ビジネスに対応する用語で定義する
- ポリシーの構築で効率的に連携できるようにビジネスエキスパートでも開発者でも使いやすい UI を提供する

これらの機能を使用することで、ビジネスの俊敏性を向上し、一貫性をもたせて効率的にデシジョンを実行し、開発サイクルだけでなく、商品を市場に投入するまでの時間を短縮します。

第3章 自動化および ANSIBLE

Ansible は自動化エンジンで、ネットワーク機能、クラウドプロビジョニング、設定管理、アプリケーションのデプロイメント、イントラサービスのオーケストレーションなどの自動化のニーズを、完全に自動化します。

Ansible は、1 台ずつシステムを管理するのではなく、お使いの全システムがどのように反復作業を行うのかを記述して IT 環境をモデル化するため、複数階層のデプロイメント向けに設計されています。

Ansible は、エージェントや、追加のカスタムセキュリティーインフラストラクチャーを使用しないため、デプロイメントが簡単です。Ansible Playbook という形式の YAML 言語を使用します。この Playbook では、英語に似た言語で自動化ジョブを記述できます。

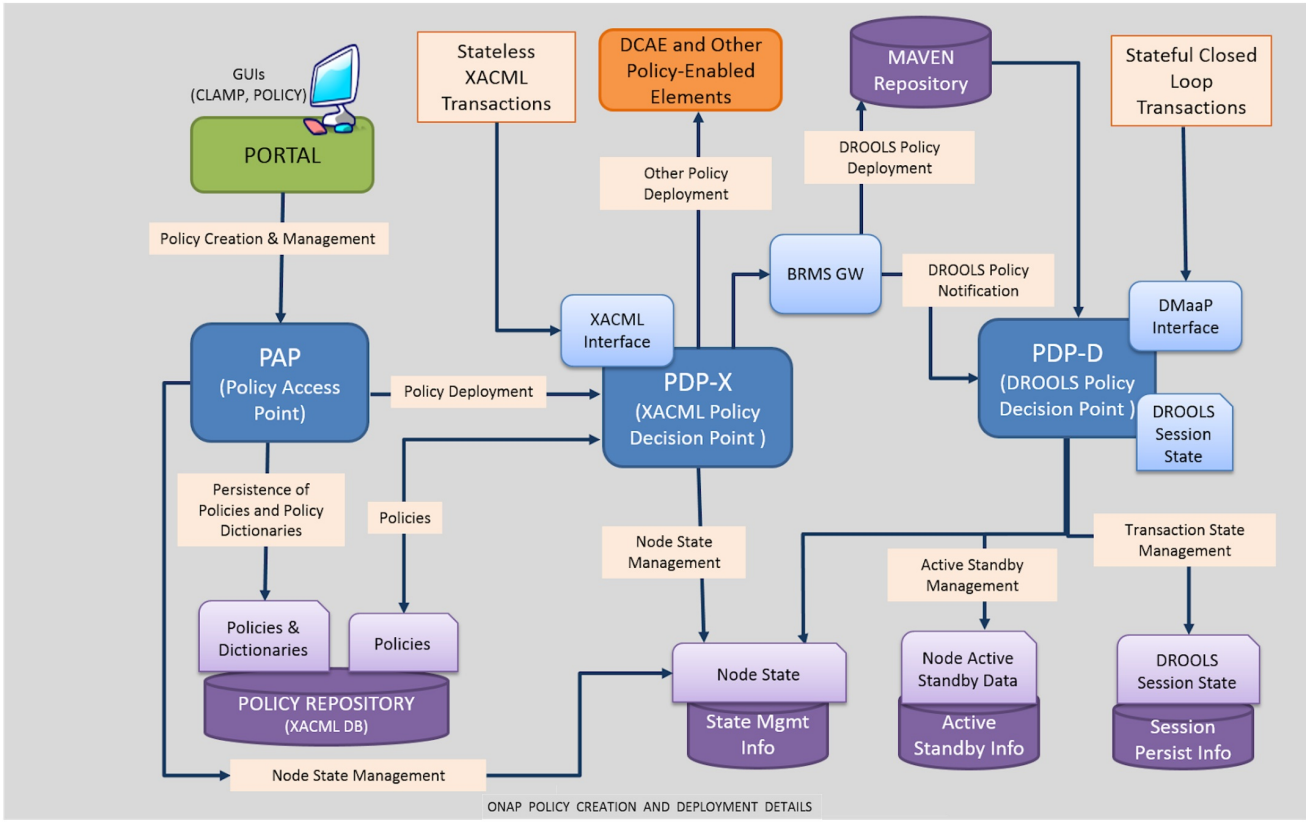
Ansible エンジンには、オプションの追加コンポーネントが 2 つ含まれています。

- Ansible Tower: Red Hat Ansible Tower では、IT 自動化のスケーリング、複雑なデプロイメントの管理、組織の生産性の向上が可能です。視覚的なダッシュボード、ロールベースのアクセス制御、ジョブのスケジュール、統合型の通知、グラフィカルなインベントリー管理を使用し、組織の IT ストラクチャーを集約して制御できます。Ansible Tower は、REST API や CLI を使用して既存のツールやプロセスに埋め込むこともできます。

- Ansible Networking アドオン: Ansible Engine が自動化ストラテジーに含まれる場合、Networking アドオンは、一般的なネットワークプラットフォーム用に社内で開発、テスト、およびリリースされる一部のネットワークモジュールへのサポートを提供します。ネットワークモジュールを使用すると、オペレーターおよびエンジニアは、異なるニーズや実装が含まれる可能性のある、混合またはハイブリッドのネットワーク環境をはじめとした環境で安心して使用できます。

第4章 RED HAT DECISION MANAGER でのポリシー

図4.1 ONAP ポリシー作成およびデプロイメント



ONAP ポリシーフレームワーク (別称 POLICY) は、以下のサブコンポーネントで設定されます。

- Policy Administration Point (PAP) では、ポリシー作成のインターフェイスを提供します。このインターフェイスは GUI を提供するポータルに組み込まれます。
- ポリシー定義点 (PDP: Policy Decision Point) は、特有のルール技術をもとにしています。これは、2 つのコンポーネントで設定されます。
  - PDP-X は XACML 技術をベースにしています。PDP-X は **ステートレス** で、PDP-X サーバーのリソースプールとしてデプロイできます。複数のサーバーをスケールアップして、容量 (水平スケーラビリティー) および可用性の両方を拡大できます。
  - PDP-D は Red Hat Decision Manager 技術をベースにしています。PDP-D は **ステートフル** で、Red Hat Decision Manager をネイティブかつステートフルに使用できます。トランザクションは、PDP-D が有効な限り、永続されます。PDP-D には以下の機能が含まれます。
    - 詳細コントロールループ
    - アクションをトリガーしてイベントを受信するためのさまざまな ONAP コンポーネントのインターフェイス
    - 正しいアクションで障害を処理し、ネットワーク全体のワークフローで状態を維持する機能
- ポリシー施行点 (PEP: Policy Enforcement Point) は、ポリシーが有効であるか、PDP などのポリシーが有効な要素からのコマンドに応答可能な ONAP サブシステムでランタイムポリシーを適用するポイントを指します。これらのサブシステムには以下が含まれます。

    - Master Services Orchestrator (MSO) は、非常に高いレベルでオーケストレーションを行い、インフラストラクチャー、ネットワーク、アプリケーションスコープのエンドツーエンドのビューを提供します。

    - Active and Available Inventory (AAI) は、アクティブ、利用可能、割り当て済みのリソースやサービス、およびその関係をリアルタイムビューで提供する ONAP サブシステムです。

    - Data Collection, Analytics, and Events (DCAE) サブシステムは、他の ONAP コンポーネントと連携して、管理対象の環境からパフォーマンス、使用率、設定に関する情報を収集します。さまざまな分析アプリケーションは、このデータを使用して、異常や重要なイベントを検出します。分析結果をもとに、ポリシー、MSO やコントローラーなど他の ONAP コンポーネントに公開するなど、アクションをトリガーします。

DCAE サブシステムの主な機能は以下のとおりです。

- 分析用にデータを収集/分析/変換/保存する。

- 分析デプロイメントのフレームワークを提供する。

これらの機能により、クローズドループは、さまざまな ONAP コンポーネントにより、ネットワーク内のイベントや他の条件に対応できるようになります。

- コントローラー: コントローラーは、サブドメイン固有のリソース (アプリケーションやネットワークなど) セットの状態を管理するのに使用します。コントローラーは、リソースの設定やインスタンス化を実行し、コントロールループアクション、移行、スケーリングなど、継続した管理で主要なエージェントです。ONAP は、以下のコントローラータイプを使用して、割り当てられたコントロールドメインに対応する実行環境でリソースを管理します。

    - ネットワークコントローラー (ネットワーク設定): ネットワークコントローラー (例: SDNC/ソフトウェア定義ネットワークコントローラー) は、ネットワーク設定のワークフローを実行し、MSO や AAI に結果のステータスを報告して、ネットワーク機能やサービスを管理し、制御します。コントロールネットワーク要素やサービスの例として、Transport Network Function、インフラストラクチャーネットワーク (リーフ、スパイン、仮想スイッチなど) および Wide-Area-Network (WAN) などが挙げられます。

    - アプリケーションコントローラー (アプリケーション): APPC などのアプリケーションコントローラーは、VFN のライフサイクル管理を行います。APPC HTTP API は、ライフサイクル管理コマンドをサポートするので、ユーザーが ONAP コンポーネントを使用して仮想アプリケーションやそのコンポーネントを管理できます。

これらのコントローラーはいずれも OpenDaylight Controller フレームワークをもとにしています。

ポリシーの初回作成後、または既存のポリシーの変更後には、ポリシー分散フレームワークを使用して、実際にポリシーが必要となる前に、リポジトリーからポリシー定義点やポリシー施行点などの使用地点にポリシーを送信します。

ポリシー適用の例として以下が挙げられます。

- DCAE データ収集機能を使用したデータの収集や保持のポリシールール

- 分析ポリシールール、異例または異常の状態の特定、DCAE 分析アプリケーションでこのような状況を検出したことを示すイベントの公開

- 関連の救済アクションやさらなる診断に関するポリシールールは、適切なコンポーネントにより、MSO、コントローラーまたは DCAE などコントロールループ内で実行されます。

- XACML および Red Hat Decision Manager などのポリシーエンジンは、ポリシーも適用し、結

果として他のコンポーネントをトリガーすることができます (例: ポリシーが指定する固有のアクションをコントローラーに実行させる)。さらに、ポリシー (ガードポリシー) によっては、強制的に、決定したアクションに対するチェックを行う場合もあります。

以下の手順では、簡単なポリシーフローを例示します。

1. 新しいデータフローは、ポリシー施行点 (PEP: Policy Enforcement Point) がインターセプトします。
2. PEP は、ポリシー定義点 (PDP: Policy Decision Point) に要求を転送します。
3. PDP は、設定されたポリシーに対する要求を評価します。
4. PDP は、デシジョン (Permit / Deny / Not Applicable / Indeterminate) に到達し、そのデシジョンを PEP に返します。

第5章 ANSIBLE を使用した自動化

Ansible では、**netconf** または **cli** などのプロトコルを使用して、さまざまな物理ネットワーク機能と仮想ネットワーク機能の自動化メカニズムを簡単に実装できます。Ansible は現在、APPC アーキテクチャーで ONAP を使用しており、VNF 管理フレームワークを提供するため、体系化された形で CLI のようなツールを使用できます。また、エージェントレスなため、ターゲットの VNF では追加のソフトウェアが必要ありません。このコンストラクトは、標準インターフェイスや **netconf** などのプロトコルをサポートするかどうかにかかわらず、一貫性を持って VNF を管理できます。アクションは、**Playbook** (または **複数の Playbook**) を構築して、SSH 経由で VNF 上の Ansible が Playbook を実行することで、VNF で実行できます。

APP-C の Ansible 拡張は、以下のアーキテクチャーで VNF を管理できます。

- Ansible 有向グラフ (DG): Ansible の有向グラフは、LCM アクションに合わせて、Ansible にある Playbook (および、Ansible では VNF アクションが Playbook にマッピングされるため APP-C アクション) を呼び出すのに使用可能な Ansible の汎用有向グラフです。

- APP-C Ansible アダプター: Ansible アダプターは、APP-C Karaf コンテナー内の OSGI バンドルで、Ansible サーバーと対話します。これは、2 つのアクションを実行する REST 呼び出しのセットです。これは最初に Playbook の実行リクエストの送信を行い、必要な場合は実行後に Playbook の結果を取得します (同期モードの場合)。

- APP-C/Ansible サーバーインターフェイス: Ansible ライブラリーは Python で記述されているため、 APP-C Karaf コンテナー内からネイティブで実行できません。代わりに、この設計では、Ansible Playbook を実行して、APP-C の要件に準拠する **REST** インターフェイスを公開できる Ansible サーバーを呼び出します。これらの要件は、準拠する Ansible サーバーによりサポートする必要があるサーバー API インターフェイスとして、文書にまとめられています。Ansible サーバーの実装はオープンのままで、サーバーがインターフェイスを使用している限り APP-C 操作への影響はありません。評価目的では、この APP-C/Ansible サーバーインターフェイスを実装する参照 Web サーバーが開発されており、コードは **appc-adapters/appc-ansible-adapter/appc-ansible-example-server** パスの App-C ONAP リポジトリーから入手できます。

アプリケーションコントローラーがイベントを受信したときのワークフロー例については、APPC Ansible Adapter のページを参照してください。

第6章 OPNFV コンポーネントを使用したクローズドループ

Barometer は、**collectd** の機能を開発して NFV デプロイメントに適したメトリックやイベントを収集する OPNFV のプロジェクトです。

Virtual function Event Stream (VES) は、VNF ヘルスやライフサイクルを管理する NFV Service Provider (SP) が使用できるように共通のモデルや形式を提供する ONAP プロジェクトです。このプロジェクトの目的は、共通のイベントストリーム形式や収集システムに移動することで、開発の工数を減らし、さまざまなソース (主に NFVI、VNF および PNF) からの計測データを自動化されたポリシーベースの管理システムに統合します。

Barometer VES プラグインは、**collectd** イベントを使用し、VES 形式で、事前設定された VES/DCAE コレクターに転送します。

イベントやメトリックを VES 形式で受け取る DCAE コレクターは、ONAP メッセージバスでイベントやメトリックを公開して、データベースに保存します。

第7章 参考資料

- ONAP Case Solution Architecture (ログインには Linux Foundation アカウントが必要)
- AT&T ECOMP Whitepaper
- ONAP Policy architecture
- ONAP Policy framework Project
- Controllers
- ONS F2F Casablanca planning (see policy)
- OPNFV Barometer project
- Barometer: Taking the pressure off of assurance and resource contention scenarios for NFVI
- Drools Community Documentation 5.3.0
- APPC Ansible Adapter
- APPC User Guide
- ONAP Project Specific Breakouts
- VES Home
- VNF Event Stream
- Network functions
- Cloud provisioning
- Configuration management
- Application deployment
- Intra-service orchestration
- PHREAK algorithm
- DMN v1.1 FEEL Compliance Level 3
- PMML

付録A バージョン情報

本書の最終更新日: 2021 年 11 月 15 日 (月)