Red Hat Advanced Cluster Management for Kubernetes 2.10

ガバナンス

ガバナンス

Last Updated: 2024-05-24

Red Hat Advanced Cluster Management for Kubernetes 2.10 ガバナンス

ガバナンス

法律上の通知

Copyright © 2024 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at
http://creativecommons.org/licenses/by-sa/3.0/
. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

概要

ポリシーを使用してクラスターのセキュリティーを強化するのに役立つ、ガバナンスポリシーフレームワークについては、以下で確認してください。

目次

第1章 セキュリティーの概要

Red Hat Advanced Cluster Management for Kubernetes コンポーネントのセキュリティーを管理します。定義したポリシーおよびプロセスでクラスターを統制し、リスクを特定して最小限に抑えます。ポリシーを使用して、ルールの定義および制御の設定を行います。

前提条件: Red Hat Advanced Cluster Management for Kubernetes の認証サービス要件を設定する必要がある。詳細は、アクセス制御 を参照すること。

クラスターのセキュリティー保護に関する詳細は、以下のトピックを参照してください。

- 証明書の概要
- ガバナンス

1.1. 証明書の概要

さまざまな証明書を使用して、Red Hat Advanced Cluster Management for Kubernetes クラスターの信頼性を検証できます。さらに、独自の証明書を使用することもできます。証明書の管理について学習するには、読み続けてください。

- 証明書
- 独自の可観測性認証局 (CA) 証明書の導入
- 証明書の管理

1.2. 証明書

Red Hat Advanced Cluster Management で実行されるサービスに必要な証明書はすべて、Red Hat Advanced Cluster Management のインストール時に作成されます。以下の証明書のリストを確認してください。これらの証明書は、Red Hat OpenShift Container Platform の以下のコンポーネントによって作成および管理されます。

- OpenShift Service Serving Certificates
- Red Hat Advanced Cluster Management Webhook コントローラー
- Kubernetes Certificates API
- OpenShift デフォルト Ingress

必要なアクセス権限: クラスターの管理者

証明書の管理に関する詳細は、以下を参照してください。

- Red Hat Advanced Cluster Management ハブクラスター証明書
- Red Hat Advanced Cluster Management マネージド証明書

注記: ユーザーは証明書のローテーションおよび更新を行います。

1.2.1. Red Hat Advanced Cluster Management ハブクラスター証明書

OpenShift のデフォルトの Ingress 証明書は、技術的にはハブクラスター証明書です。Red Hat

Advanced Cluster Management をインストールすると、可観測性証明書が作成され、この証明書を可観測性コンポーネントが使用してハブクラスターとマネージドクラスターの間のトラフィックで相互 TLS を提供します。

- **open-cluster-management-observability** namespace には以下の証明書が含まれます。
  - **observability-server-ca-certs**: サーバー側の証明書に署名する CA 証明書が含まれます。
  - **observability-client-ca-certs**: クライアント側の証明書に署名する CA 証明書が含まれます。
  - **observability-server-certs**: **observability-observatorium-api** デプロイメントで使用されるサーバー証明書が含まれます。
  - **observability-grafana-certs**: **observability-rbac-query-proxy** デプロイメントで使用されるクライアント証明書が含まれます。
- **open-cluster-management-addon-observability** namespace には、マネージドクラスターに以下の証明書が含まれます。
  - **observability-managed-cluster-certs**: ハブサーバーの **observability-server-ca-certs** と同じサーバー CA 証明書が含まれます。
  - **observability-controller-open-cluster-management.io-observability-signer-client-cert**: **metrics-collector-deployment** が使用するクライアント証明書が含まれます。

CA 証明書は 5 年間、他の証明書は 1 年間有効です。可観測性の証明書はすべて、期限が切れると自動的に更新されます。以下のリストを表示し、証明書が自動更新される場合の影響について確認します。

- CA 以外の証明書は、有効期間の残りが 73 日以下になると自動的に更新されます。証明書が更新されると、更新された証明書を使用するように関連するデプロイメントの Pod は自動的に再起動されます。
- CA 証明書は、有効期間の残りが 1 年間未満になると自動的に更新されます。証明書を更新したら、古い CA は削除されませんが、更新された CA と共存します。以前の証明書と更新された証明書はいずれも関連するデプロイメントで使用され、引き続き機能します。以前 CA 証明書は有効期限が切れると削除されます。
- 証明書の更新時には、ハブクラスターとマネージドクラスターの間のトラフィックは中断されません。

次の Red Hat Advanced Cluster Management ハブクラスター証明書テーブルを表示します。

表1.1 Red Hat Advanced Cluster Management ハブクラスター証明書

Namespace	Secret 名	Pod Label	
open-cluster-management	channels-apps-open-cluster-management-webhook-svc-ca	app=multicluster-operators-channel	open-cluster-management
channels-apps-open-cluster-management-webhook-svc-signed-ca	app=multicluster-operators-channel	open-cluster-management	multicluster-operators-application-svc-ca

Namespace	Secret 名	Pod Label	
app=multicluster-operators-application	open-cluster-management	multicluster-operators-application-svc-signed-ca	app=multicluster-operators-application
open-cluster-management-hub	registration-webhook-serving-cert signer-secret	必須ではありません。	open-cluster-management-hub

1.2.2. Red Hat Advanced Cluster Management マネージド証明書

Red Hat Advanced Cluster Management で管理される証明書と関連するシークレットを含むコンポーネント Pod の要約リストについては、次の表を参照してください。

表1.2 Red Hat Advanced Cluster Management で管理される証明書を含む Pod

Namespace	シークレット名 (該当する場合)
open-cluster-management-agent-addon	cluster-proxy-open-cluster-management.io-proxy-agent-signer-client-cert
open-cluster-management-agent-addon	cluster-proxy-service-proxy-server-certificates

1.2.2.1. マネージドクラスター証明書

証明書を使用して、ハブクラスターでマネージドクラスターを認証できます。したがって、このような証明書に関連するトラブルシューティングシナリオを認識しておくことが重要です。

マネージドクラスター証明書は自動的に更新されます。

1.2.3. 関連情報

- 証明書ポリシーコントローラーを使用して、マネージドクラスターで証明書ポリシーを作成して管理します。詳細は 証明書ポリシーコントローラー を参照してください。
- SSL/TLS 証明書を使用してプライベートでホストされている Git サーバーにセキュアに接続する方法の詳細は、 セキュアな HTTPS 接続でのカスタム CA 証明書の使用 を参照してください。
- 詳細は OpenShift のサービス提供証明書 を参照してください。
- OpenShift Container Platform のデフォルトの Ingress はハブクラスター証明書です。詳細は、 デフォルトの Ingress 証明書の置き換え を参照してください。
- トピックは 証明書の概要 を参照してください。

1.2.4. 独自の可観測性認証局 (CA) 証明書の導入

Red Hat Advanced Cluster Management for Kubernetes をインストールすると、可観測性のための認証局 (CA) 証明書のみがデフォルトで提供されます。Red Hat Advanced Cluster Management によって生

成されたデフォルトの可観測性 CA 証明書を使用したくない場合は、可観測性を有効にする前に独自の可観測性 CA 証明書を使用することを選択できます。

1.2.4.1. OpenSSL コマンドを使用した CA 証明書の生成

可観測性には、サーバー側、クライアント側の 2 つの CA 証明書が必要です。

- 以下のコマンドを使用して、CA RSA 秘密鍵を生成します。

```
openssl genrsa -out serverCAKey.pem 2048
openssl genrsa -out clientCAKey.pem 2048
```

- 秘密鍵を使用して自己署名 CA 証明書を生成します。以下のコマンドを実行します。

```
openssl req -x509 -sha256 -new -nodes -key serverCAKey.pem -days 1825 -out
serverCACert.pem
openssl req -x509 -sha256 -new -nodes -key clientCAKey.pem -days 1825 -out
clientCACert.pem
```

1.2.4.2. BYO 可観測性 CA 証明書に関連付けられたシークレットの作成

シークレットを作成するには、以下の手順を実行します。

1. 証明書および秘密鍵を使用して **observability-server-ca-certs** シークレットを作成します。以下のコマンドを実行します。

```
oc -n open-cluster-management-observability create secret tls observability-server-ca-certs -
-cert ./serverCACert.pem --key ./serverCAKey.pem
```

2. 証明書および秘密鍵を使用して **observability-client-ca-certs** シークレットを作成します。以下のコマンドを実行します。

```
oc -n open-cluster-management-observability create secret tls observability-client-ca-certs --
cert ./clientCACert.pem --key ./clientCAKey.pem
```

1.2.4.3. 関連情報

- 証明書の管理 を参照してください。
- 証明書の概要 を参照してください。

1.2.5. 証明書の管理

証明書を更新、置換、ローテーション、およびリストする方法については、以下を参照してください。

- Red Hat Advanced Cluster Management Webhook 証明書の更新
- alertmanager ルートの証明書の置き換え
- gatekeeper Webhook 証明書のローテーション
- 証明書のローテーションの確認
- ハブクラスターで管理される証明書のリスト表示

1.2.5.1. Red Hat Advanced Cluster Management Webhook 証明書の更新

Red Hat Advanced Cluster Management で管理される証明書を更新できます。これは、Red Hat Advanced Cluster Management サービスによって作成および管理される証明書です。

Red Hat Advanced Cluster Management によって管理される証明書を更新するには、以下の手順を実行します。

1. 次のコマンドを実行して、Red Hat Advanced Cluster Management で管理される証明書に関連付けられたシークレットを削除します。

```
oc delete secret -n <namespace> <secret>
```
(1)

(1) **<namespace>** と **<secret>** を使用する値に置き換えます。

2. 次のコマンドを実行して、Red Hat Advanced Cluster Management のマネージド証明書に関連付けられたサービスを再起動します。

```
oc delete pod -n <namespace> -l <pod-label>
```
(1)

(1) **<namespace>** と **<pod-label>** を **Red Hat Advanced Cluster Management のマネージドクラスター証明書** テーブルの値に置き換えます。

注:pod-label が指定されていない場合は、再起動する必要があるサービスはありません。シークレットは自動的に再作成され、使用されます。

1.2.5.2. alertmanager ルートの証明書の置き換え

OpenShift のデフォルト Ingress 証明書を使用しない場合は、alertmanager ルートを更新して alertmanager 証明書を置き換えることができます。以下の手順を実行します。

1. 以下のコマンドで 可観測性証明書を検査します。

```
openssl x509  -noout -text -in ./observability.crt
```

2. 証明書のコモンネーム (**CN**) を **alertmanager** に変更します。

3. **csr.cnf** 設定ファイルの SAN は、alertmanager ルートのホスト名に変更します。

4. 次に **open-cluster-management-observability** namespace で以下の 2 つのシークレットを作成します。以下のコマンドを実行します。

```
oc -n open-cluster-management-observability create secret tls alertmanager-byo-ca --cert ./ca.crt --key ./ca.key

oc -n open-cluster-management-observability create secret tls alertmanager-byo-cert --cert ./ingress.crt --key ./ingress.key
```

1.2.5.3. gatekeeper Webhook 証明書のローテーション

gatekeeper Webhook 証明書をローテーションするには、次の手順を実行します。

1. 次のコマンドを使用して、証明書が含まれるシークレットを編集します。

```
oc edit secret -n openshift-gatekeeper-system gatekeeper-webhook-server-cert
```

2. **data** セクションの **ca.crt**、**ca.key**、**tls.crt** および **tls.key** の内容を削除します。
3. 次のコマンドで **gatekeeper-controller-manager** Pod を削除して、gatekeeper Webhook サービスを再起動します。

```
oc delete pod -n openshift-gatekeeper-system -l control-plane=controller-manager
```

gatekeeper Webhook 証明書がローテーションされます。

1.2.5.4. 証明書のローテーションの確認

次の手順を使用して、証明書がローテーションされていることを確認します。

1. 確認したいシークレットを特定します。
2. **tls.crt** キーをチェックして、証明書が使用可能であることを確認します。
3. 次のコマンドを使用して証明書情報を表示します。

```
oc get secret <your-secret-name> -n open-cluster-management -o jsonpath='{.data.tls\.crt}' | base64 -d | openssl x509 -text -noout
```

<your-secret-name> は、検証するシークレットの名前に置き換えます。必要に応じて、namespace と JSON パスも更新します。

4. 出力で **Validity** の詳細を確認します。次の **Validity** の例を参照してください。

```
Validity
        Not Before: Jul 13 15:17:50 2023 GMT (1)
        Not After : Jul 12 15:17:50 2024 GMT (2)
```

(1) **Not Before** の値は、証明書をローテーションした日時です。

(2) **Not After** 値は、証明書の有効期限を表します。

1.2.5.5. ハブクラスターで管理される証明書のリスト表示

OpenShift Service Serving Certificates サービスを内部で使用するハブクラスターの管理対象証明書の一覧を表示できます。以下のコマンドを実行して証明書一覧を表示します。

```
for ns in multicluster-engine open-cluster-management ; do echo "$ns:" ; oc get secret -n $ns -o custom-columns=Name:.metadata.name,Expiration:.metadata.annotations.service\\.beta\\.openshift\\.io/expiry | grep -v '<none>' ; echo ""; done
```

詳細については、**Additional resources** セクションの **OpenShift Service Serving Certificates** を参照してください。

注記: 可観測性が有効な場合は、証明書が作成される追加の namespace があります。

1.2.5.6. 関連情報

- OpenShift Service Serving Certificates
- 証明書の概要

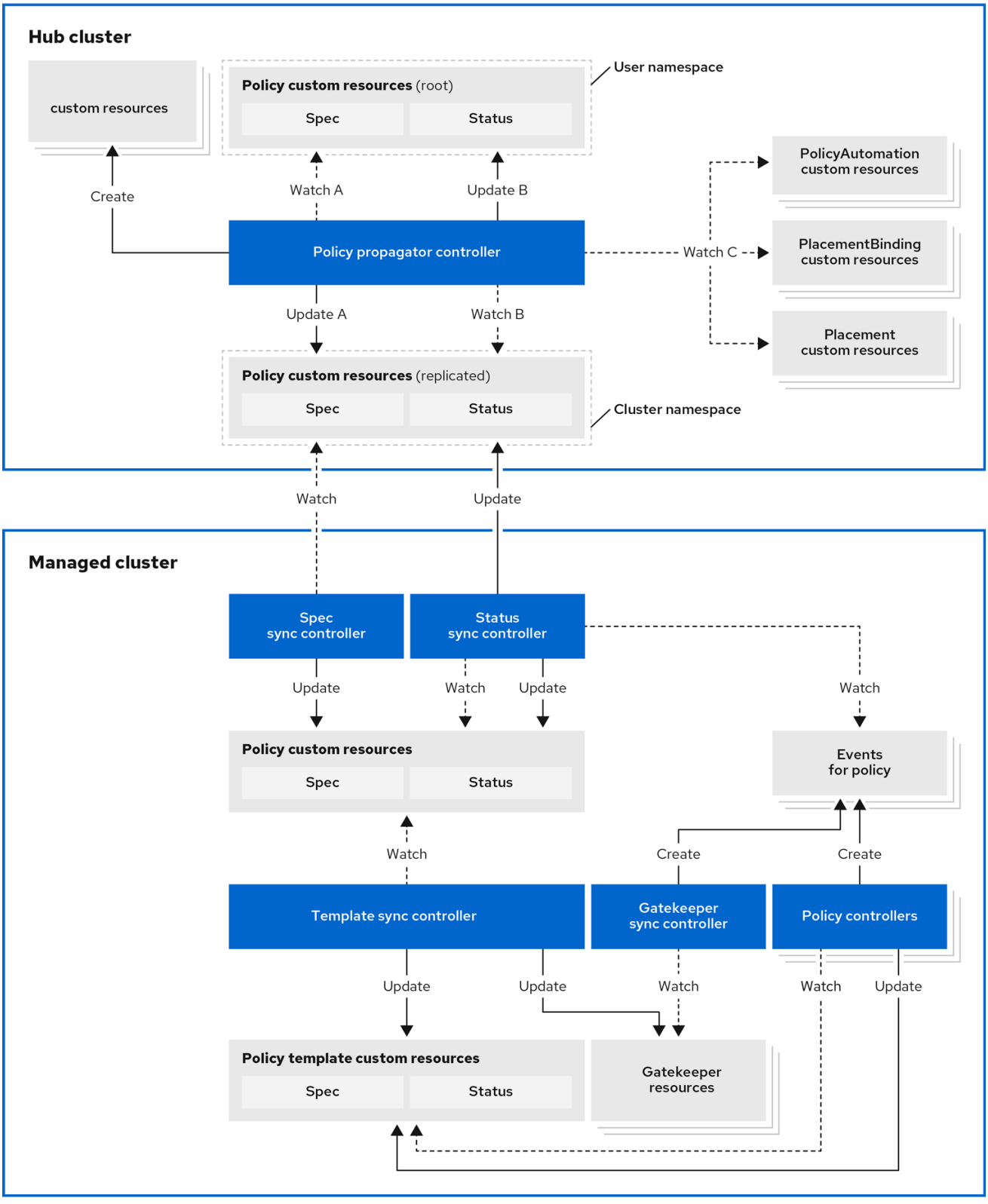
第2章 ガバナンス

企業が、プライベートクラウド、マルチクラウド、およびハイブリッドクラウドでホストされるワークロードについて、ソフトウェアエンジニアリング、セキュアなエンジニアリング、回復性、セキュリティー、規制準拠に関する内部標準を満たす必要があります。Red Hat Advanced Cluster Management for Kubernetes ガバナンスは、企業が独自のセキュリティーポリシーを導入するための拡張可能なポリシーフレームワークを提供します。Red Hat Advanced Cluster Management ガバナンスフレームワークの関連トピックを参照してください。

- ガバナンスアーキテクチャー
- ポリシーの概要
- ポリシーコントローラーの概要
- ポリシーコントローラーの高度な設定
- ポリシーコンプライアンス履歴 (テクノロジープレビュー)
- サポート対象のポリシー
- ポリシーの依存関係
- Governance ダッシュボードの管理
- ハブクラスターのセキュリティー保護
- サードパーティーポリシーコントローラーの統合

2.1. ガバナンスアーキテクチャー

Red Hat Advanced Cluster Management for Kubernetes ガバナンスライフサイクルを使用してクラスターのセキュリティーを強化します。製品ガバナンスのライフサイクルは、サポートポリシー、プロセス手順の使用をもとに、中央のインターフェイスページからセキュリティーおよびコンプライアンスを管理します。ガバナンスアーキテクチャーの以下の図を参照してください。



ガバナンスアーキテクチャーの図については、以下のコンポーネントの説明を参照してください。

- **ポリシープロパゲーターコントローラー:** Red Hat Advanced Cluster Management ハブクラスター上で実行され、ルートポリシーにバインドされた配置に基づいて、ハブ上のマネージドクラスター名前空間に複製されたポリシーを生成します。また、複製されたポリシーのコンプライアンスステータスをルートポリシーステータスに集約し、ルートポリシーにバインドされたポリシー自動化に基づいて自動化を開始します。

- **ガバナンスポリシーアドオンコントローラー:** Red Hat Advanced Cluster Management ハブクラスター上で実行され、マネージドクラスター上のポリシーコントローラーのインストールを管理します。

- **ガバナンスポリシーフレームワーク:** 前のイメージは、マネージドクラスター上で **governance-policy-framework** Pod として実行され、次のコントローラーを含むフレームワークを表しています。

  - **仕様同期コントローラー:** ハブクラスター上のマネージドクラスター名前空間内の複製されたポリシーを、マネージドクラスター上のマネージドクラスター名前空間に同期します。

  - **ステータス同期コントローラー:** ハブおよびマネージドクラスター上の複製されたポリシー内のポリシーコントローラーからのコンプライアンスイベントを記録します。ステータスには、現在のポリシーに関連する更新のみが含まれます。ポリシーが削除されて再作成された場合、過去のステータスは考慮されません。

  - **テンプレート同期コントローラー:** 複製されたポリシー **spec.policy-templates** エントリーの定義に基づいて、マネージドクラスター上のマネージドクラスター名前空間内のオブジェクトを作成、更新、および削除します。

  - **Gatekeeper 同期コントローラー:** Gatekeeper 制約監査結果を、対応する Red Hat Advanced Cluster Management ポリシーのコンプライアンスイベントとして記録します。

2.1.1. ガバナンスアーキテクチャーコンポーネント

ガバナンスアーキテクチャーには、以下のコンポーネントも含まれます。

- **ガバナンスダッシュボード**: ポリシーおよびクラスターの違反を含むクラウドガバナンスおよびリスクの詳細の概要を提供します。Red Hat Advanced Cluster Management for Kubernetes ポリシーフレームワークの構造、および Red Hat Advanced Cluster Management for Kubernetes Governance ダッシュボードの使用方法については、**Governance ダッシュボードの管理** セクションを参照してください。
  注記:

  - ポリシーがマネージドクラスターに伝播されると、最初にハブクラスターのクラスター namespace にレプリケートされ、**namespaceName.policyName** を使用して名前とラベルが付けられます。ポリシーを作成するときは、ラベル値の Kubernetes の長さ制限により、**namespaceName.policyName** の長さが 63 文字を超えないようにしてください。

  - ハブクラスターでポリシーを検索すると、マネージドクラスター namespace で複製されたポリシー名が返される場合もあります。たとえば、**default** namespace で **policy-dhaz-cert** を検索すると、ハブクラスターの次のポリシー名がマネージドクラスターの namespace にも表示される場合があります: **default.policy-dhaz-cert**。

- **ポリシーベースのガバナンスフレームワーク**: 地理的リージョンなどのクラスターに関連付けられた属性に基づいて、さまざまなマネージドクラスターへのポリシー作成およびデプロイメントをサポートします。**オープンソースコミュニティー** には、デフォルトのポリシーの例、およびクラスターにポリシーをデプロイするための手順があります。また、ポリシーに違反した場合は、ユーザーが選択したアクションを実行するように、自動化を設定できます。

- **オープンソースコミュニティー**: Red Hat Advanced Cluster Management ポリシーフレームワークの基盤を使用したコミュニティーの貢献をサポートします。ポリシーコントローラーとサードパーティーポリシーも **open-cluster-management/policy-collection** リポジトリー に含まれます。GitOps を使用して、ポリシーを投稿およびデプロイできます。

2.1.2. 関連情報

- ポリシーコントローラーの概要 を参照してください。
- GitOps を使用したポリシーのデプロイ を参照してください。

2.2. ポリシーの概要

ポリシーを作成および管理し、可視性を高め、標準に合わせて設定を修正するには、Red Hat Advanced Cluster Management for Kubernetes のセキュリティーポリシーフレームワークを使用します。ポリシーの作成には、Kubernetes カスタムリソース定義インスタンスが使用されます。ポリシーは、マネージドクラスターの namespace を除く、ハブクラスター上の任意の namespace に作成できます。マネージドクラスターの namespace にポリシーを作成すると、そのポリシーは Red Hat Advanced Cluster Management によって削除されます。Red Hat Advanced Cluster Management の各ポリシーは、1 つ以上のポリシーテンプレート定義にまとめることができます。ポリシー要素の詳細は、このページの **ポリシー YAML の表** のセクションを参照してください。

マネージドクラウド環境が、Kubernetes クラスターでホストされるワークロードの社内エンタープライズセキュリティー標準 (ソフトウェアエンジニアリング、セキュアエンジニアリング、回復力、セキュリティー、および規制コンプライアンスに関する標準) を満たしていることを確認するのは、お客様の責任です。

2.2.1. 前提条件

- 各ポリシーに、ポリシードキュメントの適用先のクラスターを定義する **Placement** リソースと、Red Hat Advanced Cluster Management for Kubernetes ポリシーをバインドする **PlacementBinding** リソースが必要です。**Placement** リソースの定義方法は、クラスターライフサイクルドキュメントの Placement の概要 を参照してください。
- マネージドクラスターにポリシーを伝播するには、**PlacementBinding** を作成して、ポリシーを **Placement** にバインドする必要があります。
- **Placement** リソースを使用するには、**ManagedClusterSetBinding** リソースを使用して、**ManagedClusterSet** リソースを **Placement** リソースの namespace にバインドする必要があります。詳細は、ManagedClusterSetBinding リソースの作成 を参照してください。
- コンソールからポリシーの **Placement** リソースを作成すると、配置の容認のステータスが **Placement** リソースに自動的に追加されます。詳細は、配置への容認の追加 参照してください。

ベストプラクティス: **Placement** リソースの使用時には、コマンドラインインターフェイス (CLI) を使用してポリシーの更新を行います。

以下のセクションでは、ポリシーコンポーネントについて説明します。

- ポリシー YAML の設定
- ポリシー YAML の表
- ポリシーサンプルファイル

2.2.2. ポリシー YAML の設定

ポリシーの作成時に、必須パラメーターフィールドと値を含める必要があります。ポリシーコントローラーによっては、他の任意のフィールドおよび値の追加が必要になる場合があります。ポリシーの YAML 設定を以下に示します。

```
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  name:
  annotations:
    policy.open-cluster-management.io/standards:
    policy.open-cluster-management.io/categories:
    policy.open-cluster-management.io/controls:
    policy.open-cluster-management.io/description:
spec:
  disabled:
  remediationAction:
  dependencies:
  - apiVersion: policy.open-cluster-management.io/v1
    compliance:
    kind: Policy
    name:
    namespace:
  policy-templates:
    - objectDefinition:
        apiVersion:
        kind:
        metadata:
          name:
        spec:
---
apiVersion: policy.open-cluster-management.io/v1
kind: PlacementBinding
metadata:
bindingOverrides:
  remediationAction:
subFilter:
  name:
placementRef:
  name:
  kind: Placement
  apiGroup: cluster.open-cluster-management.io
subjects:
- name:
  kind:
  apiGroup:
---
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name:
spec:
```

2.2.3. ポリシー YAML の表

ポリシーパラメーターの説明については、以下の表を参照してください。

表2.1パラメーターの表

フィールド	任意または必須	説明
apiVersion	必須	この値は **policy.open-cluster-management.io/v1** に設定します。
kind	必須	ポリシーのタイプを指定するには、値を **Policy** に設定します。
metadata.name	必須	ポリシーリソースを識別する名前。
metadata.annotations	任意	ポリシーが検証を試みる標準セットを記述する、一連のセキュリティー情報の指定に使用します。ここに記載されているすべてのアノテーションは、コンマ区切りリストを含む文字列として表示されます。 **注記**: コンソールの **ポリシー** ページで、ポリシー定義の標準およびカテゴリーに基づいてポリシー違反を表示できます。
bindingOverrides.remediationAction	任意	このパラメーターを **enforce** に設定すると、設定ポリシーに関連する **PlacementBinding** リソースの修復アクションをオーバーライドする方法が提供されます。デフォルト値は **null** です。
subFilter	任意	バインドされたポリシーのサブセットを選択するには、このパラメーターを **restriction** に設定します。デフォルト値は **null** です。
annotations.policy.open-cluster-management.io/standards	任意	ポリシーが関連するセキュリティー標準の名前。たとえば、アメリカ国立標準技術研究所 (NIST: National Institute of Standards and Technology) および Payment Card Industry (PCI) などがあります。

フィールド	任意または必須	説明
annotations.policy.open-cluster-management.io/categories	任意	セキュリティーコントロールカテゴリーは、1つ以上の標準に関する特定要件を表します。たとえば、システムおよび情報の整合性カテゴリーには、HIPAA および PCI 標準で必要とされているように、個人情報保護のデータ転送プロトコルが含まれる場合があります。
annotations.policy.open-cluster-management.io/controls	任意	チェックされるセキュリティー制御の名前。たとえば、アクセス制御やシステムと情報のインテグリティーなどです。
spec.disabled	必須	この値は **true** または **false** に設定します。**disabled** パラメーターを使用すると、ポリシーを有効または無効にすることができます。
spec.remediationAction	任意	ポリシーの修正を指定します。パラメーターの値は **enforce** および **inform** です。指定すると、定義した **spec.remediationAction** 値は、**policy-templates** セクションの子ポリシーに定義した **remediationAction** パラメーターより優先されます。たとえば、**spec.remediationAction** の値のセクションを **enforce** に設定すると、**policy-templates** の **remediationAction** はランタイム時に **enforce** に設定されます。
spec.copyPolicyMetadata	任意	ポリシーをマネージドクラスターに複製するときに、ポリシーのラベルとアノテーションをコピーするかどうかを指定します。**true** に設定すると、ポリシーのすべてのラベルとアノテーションが複製されたポリシーにコピーされます。**false** に設定すると、ポリシーフレームワーク固有のポリシーラベルとアノテーションのみが複製されたポリシーにコピーされます。

フィールド	任意または必須	説明
spec.dependencies	任意	コンプライアンスに関する特別な考慮事項を含む詳細な依存オブジェクトのリストを作成するために使用されます。
spec.policy-templates	必須	1つ以上のポリシーを作成し、マネージドクラスターに適用するのに使用します。
spec.policy-templates.extraDependencies	任意	ポリシーテンプレートの場合、これを使用して、コンプライアンスに関する特別な考慮事項を詳述した依存オブジェクトのリストを作成します。
spec.policy-templates.ignorePending	任意	依存関係の基準が検証されるまで、ポリシーテンプレートを準拠としてマークするために使用されます。 **重要:** 一部のポリシーの種類は、強制機能をサポートしていない場合があります。

2.2.4. ポリシーサンプルファイル

ロールの設定ポリシーである以下の YAML ファイルを確認します。

```
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  name: policy-role
  annotations:
    policy.open-cluster-management.io/standards: NIST SP 800-53
    policy.open-cluster-management.io/categories: AC Access Control
    policy.open-cluster-management.io/controls: AC-3 Access Enforcement
    policy.open-cluster-management.io/description:
spec:
  remediationAction: inform
  disabled: false
  policy-templates:
    - objectDefinition:
        apiVersion: policy.open-cluster-management.io/v1
        kind: ConfigurationPolicy
        metadata:
          name: policy-role-example
        spec:
          remediationAction: inform # the policy-template spec.remediationAction is overridden by the preceding parameter value for spec.remediationAction.
          severity: high
```

```
        namespaceSelector:
          include: ["default"]
        object-templates:
          - complianceType: mustonlyhave # role definition should exact match
            objectDefinition:
              apiVersion: rbac.authorization.k8s.io/v1
              kind: Role
              metadata:
                name: sample-role
              rules:
                - apiGroups: ["extensions", "apps"]
                  resources: ["deployments"]
                  verbs: ["get", "list", "watch", "delete","patch"]
---
apiVersion: policy.open-cluster-management.io/v1
kind: PlacementBinding
metadata:
  name: binding-policy-role
placementRef:
  name: placement-policy-role
  kind: Placement
  apiGroup: cluster.open-cluster-management.io
subjects:
- name: policy-role
  kind: Policy
  apiGroup: policy.open-cluster-management.io
---
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement-policy-role
spec:
  predicates:
  - requiredClusterSelector:
      labelSelector:
        matchExpressions:
        - {key: environment, operator: In, values: ["dev"]}
```

2.2.5. 関連情報

- ポリシーコントローラー を参照してください。
- ポリシーの作成および更新は、セキュリティーポリシーの管理 を参照してください。また、Red Hat Advanced Cluster Management ポリシーコントローラーを有効にして更新し、ポリシーのコンプライアンスを検証することもできます。
- ガバナンス ドキュメントに戻ります。

2.3. ポリシーコントローラーの概要

ポリシーコントローラーは、クラスターがポリシーに準拠しているかどうかを監視し、報告します。サポートされるポリシーテンプレートを使用して Red Hat Advanced Cluster Management for Kubernetes ポリシーフレームワークを使用し、これらのコントローラーによって管理されるポリシーを適用します。ポリシーコントローラーは Kubernetes のカスタムリソース定義インスタンスを管理します。

ポリシーコントローラーはポリシー違反をチェックし、コントローラーが強制機能をサポートしている場合、クラスターのステータスを準拠させることができます。Red Hat Advanced Cluster Management for Kubernetes の以下のポリシーコントローラーについては、次のトピックを参照してください。

- Kubernetes 設定ポリシーコントローラー
- 証明書ポリシーコントローラー
- IAM ポリシーコントローラー (非推奨)
- ポリシーセットコントローラー
- Operator ポリシーコントローラー (テクノロジープレビュー)

重要: 設定ポリシーコントローラーポリシーのみが **enforce** 機能をサポートします。ポリシーコントローラーが **enforce** 機能をサポートしないポリシーを手動で修正する必要があります。

2.3.1. Kubernetes 設定ポリシーコントローラー

設定ポリシーコントローラーを使用して、Kubernetes リソースを設定し、クラスター全体にセキュリティーポリシーを適用できます。設定ポリシーは、ハブクラスター上のポリシーの **policy-templates** フィールドで提供され、ガバナンスフレームワークによって選択されたマネージドクラスターに伝播されます。

Kubernetes オブジェクトは、設定ポリシーの **object-templates** 配列で (全体または一部で) 定義され、マネージドクラスター上のオブジェクトと比較するフィールドの設定ポリシーコントローラーを示します。設定ポリシーコントローラーは、ローカルの Kubernetes API サーバーと通信し、クラスターにある設定の一覧を取得します。

設定ポリシーコントローラーは、インストール時にマネージドクラスターに作成されます。設定ポリシーコントローラーは、設定ポリシーが準拠していない場合に修復するための **enforce** および **InformOnly** 機能をサポートします。

設定ポリシーの **remediationAction** が **enforce** に設定されている場合、コントローラーは指定された設定をターゲットのマネージドクラスターに適用します。

設定ポリシーの **remediationAction** が **InformOnly** に設定されている場合、親ポリシーの **remediationAction** が **enforce** に設定されている場合でも、親ポリシーは設定ポリシーを強制しません。

注記: 名前のないオブジェクトを指定する設定ポリシーは、**inform** のみにすることができます。

設定ポリシー内でテンプレート化された値を使用することもできます。詳細は、**Template processing** を参照してください。

ポリシーに追加したい既存の Kubernetes マニフェストがある場合、Policy Generator はこれを実現するための便利なツールです。

2.3.1.1. 設定ポリシーの例

```
apiVersion: policy.open-cluster-management.io/v1
kind: ConfigurationPolicy
metadata:
  name: policy-config
spec:
  namespaceSelector:
```

```
    include: ["default"]
    exclude: []
    matchExpressions: []
    matchLabels: {}
  remediationAction: inform
  severity: low
  evaluationInterval:
    compliant:
    noncompliant:
  object-templates:
  - complianceType: musthave
    objectDefinition:
      apiVersion: v1
      kind: Pod
      metadata:
        name: pod
      spec:
        containers:
        - image: pod-image
          name: pod-name
          ports:
          - containerPort: 80
  - complianceType: musthave
    objectDefinition:
    apiVersion: v1
    kind: ConfigMap
    metadata:
      name: myconfig
      namespace: default
      data:
      testData: hello
    spec:
...
```

2.3.1.2. 設定ポリシーの YAML の表

表2.2 パラメーターの表

フィールド	任意または必須	説明
apiVersion	必須	この値は **policy.open-cluster-management.io/v1** に設定します。
kind	必須	ポリシーのタイプを指定するには、値を **ConfigurationPolicy** に設定します。
metadata.name	必須	ポリシーの名前。

フィールド	任意または必須	説明
spec.namespaceSelector	namespace が指定されていない namespace 付きオブジェクトに必要	オブジェクトが適用されるマネージドクラスター内の namespace を決定します。**include** パラメーターと **exclude** パラメーターは、ファイルパス式を受け入れて、名前で namespace を含めたり除外したりします。**matchExpressions** および **matchLabels** パラメーターは、ラベルによって含める namespace を指定します。Kubernetes のラベルとセレクター のドキュメントを参照してください。結果のリストは、すべてのパラメーターからの結果の共通部分を使用してコンパイルされます。
spec.remediationAction	必須	ポリシーが準拠していない場合に実行するアクションを指定します。次のパラメーター値を使用します。**info**、**InformOnly**、または **enforce**。
spec.severity	必須	ポリシーがコンプライアンス違反の場合に重大度を指定します。次のパラメーター値を使用します: **low**、**medium**、**high**、または **critical**。
spec.evaluationInterval.compliant	任意	ポリシーが準拠状態にあるときに評価される頻度を定義するために使用されます。値は期間の形式に指定する必要があります。これは、時間単位接尾辞が付いた数字のシーケンスになります。たとえば、**12h30m5s** は 12 時間、30 分、および 5 秒を表します。ポリシー **spec** が更新されない限り、ポリシーが準拠クラスターで再評価されないように、**never** に設定することもできます。 デフォルトでは、**evaluationInterval.compliance** が設定されていないか、空の場合は、設定ポリシーの評価間隔の最小時間は約 10 秒です。設定ポリシーコントローラーがマネージドクラスターで飽和している場合、この感覚はさらに長くなる可能性があります。

フィールド	任意または必須	説明
spec.evaluationInterval.noncompliant	任意	ポリシーがコンプライアンス違反の状態にあるときに評価される頻度を定義するために使用します。**evaluationInterval.compliant** パラメーターと同様に、値は時間単位接尾辞が付いた数値のシーケンスにある期間の形式である必要があります。ポリシー **spec** が更新されない限り、ポリシーがコンプライアンス違反のクラスターで再評価されないように、**never** に設定することもできます。
spec.object-templates	任意	コントローラーがマネージドクラスター上のオブジェクトと比較するための Kubernetes オブジェクトの配列 (完全に定義されているか、フィールドのサブセットを含む)。**注: spec.object-templates** と **spec.object-templates-raw** は、オプションとしてリストされていますが、2 つのパラメーターフィールドのうち 1 つだけを設定する必要があります。

フィールド	任意または必須	説明
spec.object-templates-raw	任意	生の YAML 文字列を使用してオブジェクトテンプレートを設定するために使用されます。オブジェクトテンプレートの条件を指定します。ここでは、if-else ステートメントや **range** 関数などの高度な関数がサポートされる値です。たとえば、**object-template** 定義での重複を回避するために次の値を追加します。 **{{- if eq .metadata.name "policy-grc-your-meta-data-name" }} replicas: 2 {{- else }} replicas: 1 {{- end }}** **注: spec.object-templates** と **spec.object-templates-raw** は、オプションとしてリストされていますが、2 つのパラメーターフィールドのうち 1 つだけを設定する必要があります。

フィールド	任意または必須	説明
spec.object-templates[].complianceType	必須	マネージドクラスター上の Kubernetes オブジェクトの望ましい状態を定義するために使用されます。次のいずれかの動詞をパラメーター値として使用する必要があります。 **mustonlyhave**: **objectDefinition** で定義されている正確なフィールドと値を持つオブジェクトが存在する必要があることを示します。 **musthave**: **objectDefinition** で指定されたものと同じフィールドを持つオブジェクトが存在する必要があることを示します。**object-template** で指定されていないオブジェクト上の既存のフィールドは無視されます。通常、配列値は追加されます。パッチが適用される配列の例外は、既存のアイテムと一致する値を持つ **name** キーがアイテムに含まれている場合です。配列を置き換える場合は、**mustonlyhave** コンプライアンスタイプを使用して、完全に定義された **objectDefinition** を使用します。 **mustnothave**: **objectDefinition** で指定されたものと同じフィールドを持つオブジェクトが存在できないことを示します。
spec.object-templates[].metadataComplianceType	任意	マニフェストのメタデータセクションをクラスター上のオブジェクトと比較するときに、**spec.object-templates[].complianceType** をオーバーライドします (musthave、mustonlyhave)。デフォルトは、メタデータの **ComplianceType** をオーバーライドしないように設定されていません。

フィールド	任意または必須	説明
spec.object-templates[].recordDiff	任意	クラスター上のオブジェクトとポリシー内の **objectDefinition** との差異をログに記録するかどうか、および記録する場所を指定します。コントローラーログに差異を記録する場合は **Log** に設定し、差異を記録しない場合は **None** に設定します。デフォルトでは、このパラメーターは空であり、差異はログに記録されません。
spec.object-templates[].objectDefinition	必須	コントローラーがマネージドクラスター上のオブジェクトと比較するための Kubernetes オブジェクト (完全に定義されているか、フィールドのサブセットを含む)。
spec.pruneObjectBehavior	任意	マネージドクラスターからポリシーが削除されたときに、ポリシーに関連するリソースを消去するかどうかを決定します。

2.3.1.3. 関連情報

詳細については、以下のトピックを参照してください。

- ハブクラスターポリシーの詳細は、ポリシーの概要 を参照してください。
- NIST Special Publication 800-53 (Rev. 4) を使用するポリシーサンプルおよび、**CM-Configuration-Management** フォルダー の Red Hat Advanced Cluster Management がサポートするポリシーサンプルを参照してください。
- ポリシーがハブクラスターにどのように適用されるかについては、サポート対象のポリシー 参照してください。
- コントローラーの詳細は、ポリシーコントローラー を参照してください。
- ポリシーコントローラーの設定をカスタマイズします。ポリシーコントローラーの高度な設定 を参照してください。
- リソースのクリーンアップとその他のトピックについては、ポリシーによって作成されたリソースのクリーンアップ ドキュメントを参照してください。
- ポリシージェネレーター を参照してください。
- ポリシーを作成してカスタマイズする方法は、Governance ダッシュボードの管理 を参照してください。

- テンプレート処理 を参照してください。

2.3.2. 証明書ポリシーコントローラー

証明書ポリシーコントローラーは、有効期限が近い証明書、期間 (時間) が長すぎる証明書や、指定のパターンに一致しない DNS 名が含まれる証明書の検出に使用できます。ハブクラスターのポリシーの **policy-templates** フィールドに証明書ポリシーを追加できます。証明書ポリシーは、ガバナンスフレームワークを使用して、選択したマネージドクラスターに伝播されます。ハブクラスターポリシーの詳細は、ポリシーの概要 に関するドキュメントを参照してください。

証明書ポリシーコントローラーを設定してカスタマイズするには、コントローラーポリシーの以下のパラメーターを更新します。

- **minimumDuration**
- **minimumCADuration**
- **maximumDuration**
- **maximumCADuration**
- **allowedSANPattern**
- **disallowedSANPattern**

以下のシナリオのいずれかの場合は、ポリシーがコンプライアンス違反になる可能性があります。

- 証明書が、最小期間で指定されている期間以内、または最大期間で指定されている期間を超えて失効する場合
- DNS 名が指定のパターンと一致しない場合

証明書ポリシーコントローラーは、マネージドクラスターに作成されます。このコントローラーは、ローカルの Kubernetes API サーバーと通信して、証明書が含まれるシークレット一覧を取得して、コンプライアンス違反の証明書をすべて判別します。

証明書ポリシーコントローラーには、**enforce** 機能のサポートがありません。

注記: 証明書ポリシーコントローラーは、**tls.crt** キーのシークレットでのみ自動的に証明書を検索します。シークレットが別のキーの下に保存されている場合は、証明書ポリシーコントローラーに別のキーを検索するよう知らせるために、**certificate_key_name** という名前のラベルを、キーに設定された値とともに追加します。たとえば、**sensor-cert.pem** という名前のキーに保存されている証明書がシークレットに含まれている場合は、ラベル **certificate_key_name: sensor-cert.pem** をシークレットに追加します。

2.3.2.1. 証明書ポリシーコントローラーの YAML 設定

以下の証明書ポリシーの例を見て、YAML 表の要素を確認します。

```
apiVersion: policy.open-cluster-management.io/v1
kind: CertificatePolicy
metadata:
  name: certificate-policy-example
spec:
  namespaceSelector:
    include: ["default"]
```

```
  exclude: []
  matchExpressions: []
  matchLabels: {}
labelSelector:
  myLabelKey: myLabelValue
remediationAction:
severity:
minimumDuration:
minimumCADuration:
maximumDuration:
maximumCADuration:
allowedSANPattern:
disallowedSANPattern:
```

2.3.2.1.1. 証明書ポリシーコントローラーの YAML の表

表2.3 パラメーターの表

フィールド	任意または必須	説明
apiVersion	必須	この値は **policy.open-cluster-management.io/v1** に設定します。
kind	必須	この値を **CertificatePolicy** に設定してポリシーの種類を指定します。
metadata.name	必須	ポリシーを識別するための名前。
metadata.labels	任意	証明書ポリシーでは、**category=system-and-information-integrity** ラベルでポリシーを分類して、証明書ポリシーをスムーズにクエリーできるようにします。証明書ポリシーの **category** キーに別の値が指定されていると、この値は証明書コントローラーにより上書きされます。

フィールド	任意または必須	説明
spec.namespaceSelector	必須	シークレットがモニターされるマネージドクラスター内の namespace を決定します。**include** パラメーターと **exclude** パラメーターは、ファイルパス式を受け入れて、名前で namespace を含めたり除外したりします。**matchExpressions** および **matchLabels** パラメーターは、ラベルによって含まれる namespace を指定します。Kubernetes のラベルとセレクター のドキュメントを参照してください。結果のリストは、すべてのパラメーターからの結果の共通部分を使用してコンパイルされます。 **注記:**証明書ポリシーコントローラーの **namespaceSelector** がどの namespace にも一致しない場合は、ポリシーが準拠しているとみなされます。
spec.labelSelector	任意	オブジェクトの識別属性を指定します。Kubernetes のラベルとセレクター のドキュメントを参照してください。
spec.remediationAction	必須	ポリシーの修正を指定します。このパラメーター値に **inform** を設定します。証明書ポリシーコントローラーがサポートするのは **inform** 機能のみです。
spec.severity	任意	ポリシーがコンプライアンス違反の場合に重大度をユーザーに通知します。次のパラメーター値を使用します: **low**、**medium**、**high**、または **critical**。

フィールド	任意または必須	説明
spec.minimumDuration	必須	値の指定がない場合、デフォルト値は **100h** になります。このパラメーターで、証明書がコンプライアンス違反とみなされるまでの最小期間 (時間) を指定します。パラメーター値は Golang の期間形式を使用します。詳細は Golang Parse Duration を参照してください。
spec.minimumCADuration	任意	値を設定して、他の証明書とは異なる値で、まもなく有効期限が切れる可能性がある署名証明書を特定します。パラメーターの値が指定されていないと、CA 証明書の有効期限は **minimumDuration** で使用した値になります。詳細は Golang Parse Duration を参照してください。
spec.maximumDuration	任意	値を設定して、任意の制限期間を超えて作成された証明書を特定します。パラメーターは Golang の期間形式を使用します。詳細は Golang Parse Duration を参照してください。
spec.maximumCADuration	任意	値を設定して、定義した制限期間を超えて作成された署名証明書を特定します。パラメーターは Golang の期間形式を使用します。詳細は Golang Parse Duration を参照してください。
spec.allowedSANPattern	任意	証明書に定義した全 SAN エントリーと一致する必要がある正規表現。このパラメーターを使用して、パターンと DNS 名を照合します。詳細は Golang Regular Expression syntax を参照してください。

フィールド	任意または必須	説明
spec.disallowedSANPattern	任意	証明書で定義した SAN エントリーと一致してはいけない正規表現。このパラメーターを使用して、パターンと DNS 名を照合します。 **注記**: ワイルドカードの証明書を検出するには、**disallowedSANPattern: "[*]"** の SAN パターンを使用します。 詳細は Golang Regular Expression syntax を参照してください。

2.3.2.2. 証明書ポリシーの例

証明書ポリシーコントローラーがハブクラスターに作成されると、複製ポリシーがマネージドクラスターに作成されます。証明書ポリシーのサンプルを確認するには、**policy-certificate.yaml** を参照してください。

2.3.2.3. 関連情報

- 証明書ポリシーの管理方法の詳細は、セキュリティーポリシーの管理 を参照してください。
- 他のトピックについては、ポリシーコントローラーの概要 を参照してください。
- 証明書の概要 を参照してください。

2.3.3. IAM ポリシーコントローラー (非推奨)

IAM (ID and Access Management) ポリシーコントローラーを使用して、コンプライアンス違反の IAM ポリシーに関する通知を受信できます。IAM ポリシーで設定したパラメーターを基に、コンプライアンスチェックが行われます。IAM ポリシーは、ハブクラスターのポリシーの **policy-templates** フィールドで提供され、ガバナンスフレームワークによって選択されたマネージドクラスターに伝播されます。ハブクラスターポリシーの詳細は、ポリシー YAML 構造 のドキュメントを参照してください。

IAM ポリシーコントローラーは、クラスター内で特定のクラスターロール (**ClusterRole**) を割り当てたユーザーの必要な最大数を監視します。監視するデフォルトのクラスターロールは **cluster-admin** です。IAM ポリシーコントローラーは、ローカルの Kubernetes API サーバーと通信します。

IAM ポリシーコントローラーはマネージドクラスターで実行されます。詳細は、以下のセクションを参照してください。

- IAM ポリシー YAML の設定
- IAM ポリシー YAML の表
- IAM ポリシーの例

2.3.3.1. IAM ポリシー YAML の設定

以下の IAM ポリシーの例を見て、YAML 表のパラメーターを確認します。

```
apiVersion: policy.open-cluster-management.io/v1
kind: IamPolicy
metadata:
  name:
spec:
  clusterRole:
  severity:
  remediationAction:
  maxClusterRoleBindingUsers:
  ignoreClusterRoleBindings:
```

2.3.3.2. IAM ポリシー YAML の表

以下のパラメーター表で説明を確認してください。

表2.4 パラメーターの表

フィールド	任意または必須	説明
apiVersion	必須	この値は **policy.open-cluster-management.io/v1** に設定します。
kind	必須	ポリシーのタイプを指定するには、値を **Policy** に設定します。
metadata.name	必須	ポリシーリソースを識別する名前。
spec.clusterRole	任意	監視するクラスターロール (**ClusterRole**)指定されていない場合は、**cluster-admin** にデフォルト設定されます。
spec.severity	任意	ポリシーがコンプライアンス違反の場合に重大度をユーザーに通知します。次のパラメーター値を使用します: **low**、**medium**、**high**、または **critical**。
spec.remediationAction	任意	ポリシーの修正を指定します。**inform** と入力します。IAM ポリシーコントローラーは、**inform** 機能のみをサポートします。

フィールド	任意または必須	説明
spec.ignoreClusterRoleBindings	任意	無視するクラスターのロールバインディング名を指定する正規表現 (regex) 値の一覧。これらの正規表現の値は、Go regexp 構文 に従う必要があります。デフォルトでは、名前が **system:** で始まるすべてのクラスターロールバインディングは無視されます。これをより厳格な値に設定することが推奨されます。クラスターのロールバインディング名を無視するには、一覧を単一の値 **.^** か、一致しない他の正規表現に設定します。
spec.maxClusterRoleBindingUsers	必須	ポリシーが違反しているとみなされるまでに利用可能な IAM rolebinding の最大数。

2.3.3.3. IAM ポリシーの例

IAM ポリシーのサンプルを確認するには、**policy-limitclusteradmin.yaml** を参照してください。詳細は、セキュリティーポリシーの管理 を参照してください。他のトピックについては、ポリシーコントローラー を参照してください。

2.3.4. ポリシーセットコントローラー

ポリシーセットコントローラーは、同じ namespace で定義されるポリシーにスコープ指定されたポリシーのステータスを集約します。ポリシーセット (**PolicySet**) を作成して、同じ namespace にあるポリシーをグループ化します。**PolicySet** のすべてのポリシーは、**PolicySet** および **Placement** をバインドする **PlacementBinding** を作成して、選択したクラスターに配置されます。ポリシーセットがハブクラスターにデプロイされています。

また、ポリシーが複数のポリシーセットの一部である場合、既存および新規 **Placement** リソースはポリシーに残ります。ユーザーがポリシーセットからポリシーを削除すると、ポリシーはポリシーセットで選択したクラスターには適用されませんが、配置は残ります。ポリシーセットコントローラーは、ポリシーセット配置を含むクラスターの違反のみを確認します。

注記:

- Red Hat Advanced Cluster Management サンプルポリシーセットは、クラスター配置を使用します。クラスター配置を使用する場合は、ポリシーを含む namespace をマネージドクラスターセットにバインドします。クラスター配置の使用の詳細は、クラスターへのポリシーのデプロイ を参照してください。
- **Placement** リソースを使用するには、**ManagedClusterSetBinding** リソースを使用して、**ManagedClusterSet** リソースを **Placement** リソースの namespace にバインドする必要があります。詳細は、ManagedClusterSetBinding リソースの作成 を参照してください。

以下のセクションでは、ポリシーセットの設定について説明します。

- ポリシーセットコントローラー YAML の設定
- ポリシーセットコントローラー YAML の表
- ポリシーセットの例

2.3.4.1. ポリシーセット YAML の設定

ポリシーセットは、以下の YAML ファイルのようになります。

```
apiVersion: policy.open-cluster-management.io/v1beta1
kind: PolicySet
metadata:
  name: demo-policyset
spec:
  policies:
  - policy-demo

---
apiVersion: policy.open-cluster-management.io/v1
kind: PlacementBinding
metadata:
  name: demo-policyset-pb
placementRef:
  apiGroup: cluster.open-cluster-management.io
  kind: Placement
  name: demo-policyset-pr
subjects:
- apiGroup: policy.open-cluster-management.io
  kind: PolicySet
  name: demo-policyset
---
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: demo-policyset-pr
spec:
  predicates:
  - requiredClusterSelector:
      labelSelector:
        matchExpressions:
          - key: name
            operator: In
            values:
              - local-cluster
```

2.3.4.2. ポリシーセットの表

以下のパラメーター表で説明を確認してください。

表2.5 パラメーターの表

フィールド	任意または必須	説明
apiVersion	必須	この値は **policy.open-cluster-management.io/v1beta1** に設定します。
kind	必須	ポリシーのタイプを指定するには、値を **PolicySet** に設定します。
metadata.name	必須	ポリシーリソースを識別する名前。
spec	必須	ポリシーの設定詳細を追加します。
spec.policies	任意	ポリシーセットでグループ化するポリシーの一覧。

2.3.4.3. ポリシーセットの例

```
apiVersion: policy.open-cluster-management.io/v1beta1
kind: PolicySet
metadata:
  name: pci
  namespace: default
spec:
  description: Policies for PCI compliance
  policies:
  - policy-pod
  - policy-namespace
status:
  compliant: NonCompliant
  placement:
  - placementBinding: binding1
    placement: placement1
    policySet: policyset-ps
```

2.3.4.4. 関連情報

- Red Hat OpenShift Platform Plus policy set を参照してください。
- セキュリティーポリシーの管理 トピックの **Creating policy sets**セクションを参照してください。
- また、デプロイメント用のポリシージェネレーターである PolicySets-- Stable を必要とする安定したポリシー **PolicySets** も表示します。
- このトピックの最初の Policy set controller に戻ります。

2.3.5. Operator ポリシーコントローラー (テクノロジープレビュー)

Operator ポリシーコントローラーを使用すると、クラスター全体の Operator Lifecycle Manager (OLM) Operator を監視し、インストールできます。Operator ポリシーコントローラーは、Operator のさまざまな部分の健全性を監視し、Operator への更新を自動的に処理する方法を指定するために使用します。ガバナンスフレームワークを使用して、ハブクラスター上のポリシーの **policy-template** フィールドにポリシーを追加することで、Operator ポリシーをマネージドクラスターに配布することもできます。

2.3.5.1. 前提条件

- マネージドクラスターで OLM を使用可能にしている。これは、Red Hat OpenShift Container Platform ではデフォルトで有効になっています。
- **必要なアクセス権限:** クラスターの管理者

2.3.5.2. Operator ポリシー YAML の表

フィールド	任意または必須	説明
apiVersion	必須	この値は **policy.open-cluster-management.io/v1beta1** に設定します。
kind	必須	ポリシーのタイプを指定するには、値を **OperatorPolicy** に設定します。
metadata.name	必須	ポリシーリソースを識別する名前。
spec.remediationAction	必須	Operator ポリシーの **remediationAction** が **enforce** に設定されている場合、コントローラーはターゲットのマネージドクラスターにリソースを作成し、OLM と通信して Operator をインストールし、ポリシーで指定されたバージョンに基づいて更新を承認します。+ **remediationAction** が **inform** に設定されている場合、コントローラーはアップグレードが利用可能かどうかなど、Operator のステータスのみを報告します。

フィールド	任意または必須	説明
spec.operatorGroup	任意	デフォルトでは、**operatorGroup** フィールドが指定されていない場合、コントローラーは、サブスクリプションと同じ namespace に **AllNamespaces** タイプの OperatorGroup を生成します (サポートされている場合)。このリソースは、Operator ポリシーコントローラーによって生成されます。
spec.subscription	必須	Operator のササブスクリプションを作成するための設定を定義します。Operator のサブスクリプションを作成するには、次のフィールドに情報を追加する必要があります。 • **channel** • **name** • **namespace** • **source** • **sourceNamespace**
subscriptions.installPlanApproval	必須	**installPlanApproval** フィールドが **Manual** に設定され、**spec.versions** フィールドが空の場合、インストールプロセスを続行するには、関連する **InstallPlan** リソースの **.spec.approved** フィールドに手動でパッチを適用する必要があります。Operator の **InstallPlan** リソースは、コントローラーによってサブスクリプションが作成された後に生成され、目的の Operator の特定バージョンをインストールする旨を表します。

フィールド	任意または必須	説明
spec.versions	任意	Operator のどのバージョンがポリシーに準拠しているかを宣言します。フィールドが空の場合、クラスター上で実行されているすべてのバージョンが準拠しているとみなされます。フィールドが空でない場合、ポリシーに準拠しているとみなされるには、マネージドクラスターのバージョンがリスト内のいずれかのバージョンと一致している必要があります。ポリシーが **enforce** に設定され、リストが空でない場合、このフィールドにリストしたバージョンがクラスターのコントローラーによって承認されます。

2.3.5.3. 関連情報

- 詳細は、**OperatorPolicy** リソースを使用してオペレーターをインストールする を参照してください。
- 詳細は、Operator Lifecycle Manager (OLM) を参照してください。
- OLM に関する一般的な情報は、クラスターへの Operators の追加 ドキュメントを参照してください。

2.4. ポリシーコントローラーの高度な設定

ManagedClusterAddOn カスタムリソースを使用して、マネージドクラスターのポリシーコントローラー設定をカスタマイズできます。次の **ManagedClusterAddOn** は、ポリシーフレームワーク、Kubernetes 設定ポリシーコントローラー、証明書ポリシーコントローラー、および IAM ポリシーコントローラーを設定します。

必要なアクセス権限: クラスターの管理者

- ガバナンスフレームワークの同時実行性を設定する
- 設定ポリシーコントローラーの同時実行性を設定する
- API サーバーへのリクエストのレートを設定する
- デバッグログを設定する
- ガバナンスメトリクス
- 設定変更を確認する

2.4.1. ガバナンスフレームワークの同時実行性を設定する

各マネージドクラスターのガバナンスフレームワークの同時実行性を設定します。デフォルト値の **2** を変更するには、**policy-evaluation-concurrency** アノテーションを引用符で囲んだゼロ以外の整数で設

定します。次に、ハブクラスターのマネージドクラスター namespace で、**ManagedClusterAddOn** オブジェクト名の値を **governance-policy-framework** に設定します。

次の YAML の例を参照してください。ここでは、**cluster1** という名前のマネージドクラスターで、同時実行性が **2** に設定されています。

```
apiVersion: addon.open-cluster-management.io/v1alpha1
kind: ManagedClusterAddOn
metadata:
  name: governance-policy-framework
  namespace: cluster1
  annotations:
    policy-evaluation-concurrency: "2"
spec:
  installNamespace: open-cluster-management-agent-addon
```

client-qps および **client-burst** アノテーションを設定するには、**ManagedClusterAddOn** リソースを更新し、パラメーターを定義します。

次の YAML の例では、**cluster 1** というマネージドクラスター上で 1 秒あたりのクエリーが **30** に設定され、バーストが **45** に設定されています。

```
apiVersion: addon.open-cluster-management.io/v1alpha1
kind: ManagedClusterAddOn
metadata:
  name: governance-policy-framework
  namespace: cluster1
  annotations:
    client-qps: "30"
    client-burst: "45"
spec:
  installNamespace: open-cluster-management-agent-addon
```

2.4.2. 設定ポリシーコントローラーの同時実行性を設定する

マネージドクラスターごとに設定ポリシーコントローラーの並行処理性を設定して、同時に評価できる設定ポリシーの数を変更できます。デフォルト値の **2** を変更するには、**policy-evaluation-concurrency** アノテーションを引用符で囲んだゼロ以外の整数で設定します。次いで、ハブクラスターのマネージドクラスター namespace で、**ManagedClusterAddOn** オブジェクト名の値を **config-policy-controller** に設定します。

注記: 同時実行値を増やすと **config-policy-controller** Pod、Kubernetes API サーバー、および OpenShift API サーバー上の CPU とメモリーの使用率が増加します。

次の YAML の例を参照してください。ここでは、**cluster1** という名前のマネージドクラスターで同時実行性が **5** に設定されています。

```
apiVersion: addon.open-cluster-management.io/v1alpha1
kind: ManagedClusterAddOn
metadata:
  name: config-policy-controller
  namespace: cluster1
  annotations:
```

```
    policy-evaluation-concurrency: "5"
spec:
  installNamespace: open-cluster-management-agent-addon
```

2.4.3. API サーバーへのリクエストのレートを設定する

設定ポリシーコントローラーが各マネージドクラスター上で行う API サーバーへのリクエストの速度を設定します。レートが増加すると、設定ポリシーコントローラーの応答性が向上し、Kubernetes API サーバーと OpenShift API サーバーの CPU とメモリーの使用率も増加します。デフォルトでは、リクエストのレートは、**policy-evaluation-concurrency** 設定に応じて調整され、1 秒あたり **30** クエリー (QPS) に設定され、バースト値は **45** で、短期間でより多くのリクエストが発生することを表します。

client-qps および **client-burst** アノテーションを引用符で囲んだゼロ以外の整数で設定することで、レートとバーストを設定できます。ハブクラスターのマネージドクラスター namespace で、**ManagedClusterAddOn** オブジェクト名の値を **config-policy-controller** に設定できます。

次の YAML の例では、**cluster 1** というマネージドクラスター上で 1 秒あたりのクエリーが **20** に設定され、バーストが **100** に設定されています。

```
apiVersion: addon.open-cluster-management.io/v1alpha1
kind: ManagedClusterAddOn
metadata:
  name: config-policy-controller
  namespace: cluster1
  annotations:
    client-qps: "20"
    client-burst: "100"
spec:
  installNamespace: open-cluster-management-agent-addon
```

2.4.4. デバッグログを設定する

各ポリシーコントローラーのデバッグログを設定して収集するときに、ログレベルを調整できます。

注: デバッグログの量を減らすと、ログから表示される情報が少なくなります。

ポリシーコントローラーによって発行されるデバッグログを減らして、エラーのみのバグをログに表示するようにできます。デバッグログを減らすには、アノテーションで debug log の値を **-1** に設定します。それぞれの値が何を表すかを確認してください。

- **-1**: エラーログのみ
- **0**: 情報ログ
- **1**: デバッグログ
- **2**: 詳細なデバッグログ

Kubernetes 設定コントローラーの第 2 レベルのデバッグ情報を受け取るには、値が **2** の **log-level** アノテーションを **ManagedClusterAddOn** カスタムリソースに追加します。デフォルトでは、**log-level** は **0** に設定されています。つまり、情報メッセージを受信します。以下の例を参照してください。

```
apiVersion: addon.open-cluster-management.io/v1alpha1
kind: ManagedClusterAddOn
metadata:
```

```
  name: config-policy-controller
  namespace: cluster1
  annotations:
    log-level: "2"
spec:
  installNamespace: open-cluster-management-agent-addon
```

さらに、**ConfigurationPolicy** リソース内のそれぞれの **spec.object-template[]** で、パラメーター **RecordDiff** を **Log** に設定できます。**objectDefinition** とマネージドクラスター上のオブジェクトとの差異が、マネージドクラスター上の **config-policy-controller** Pod のログに記録されます。以下の例を参照してください。

recordDiff: Log に設定した **ConfigurationPolicy** リソース

```
apiVersion: policy.open-cluster-management.io/v1
kind: ConfigurationPolicy
metadata:
  name: my-config-policy
spec:
  object-templates:
  - complianceType: musthave
    recordDiff: Log
    objectDefinition:
      apiVersion: v1
      kind: ConfigMap
      metadata:
        name: my-configmap
      data:
        fieldToUpdate: "2"
```

クラスターの **ConfigMap** リソースに **fieldToUpdate: "1"** がリストされている場合、**config-policy-controller** Pod のログに次の情報を含む差異が表示されます。

```
Logging the diff:
--- default/my-configmap : existing
+++ default/my-configmap : updated
@@ -2,3 +2,3 @@
 data:
-  fieldToUpdate: "1"
+  fieldToUpdate: "2"
 kind: ConfigMap
```

重要: セキュアなオブジェクトの差異をログに記録しないでください。差異はプレーンテキストで記録されます。

2.4.5. ガバナンスメトリクス

ポリシーフレームワークは、ポリシーディストリビューションとコンプライアンスを表示するメトリックを公開します。ハブクラスターで **policy_governance_info** メトリックを使用してトレンドを表示し、ポリシーの失敗を分析します。メトリックの概要については、次のトピックを参照してください。

2.4.5.1. メトリック: policy_governance_info

OpenShift Container Platform モニタリングコンポーネントは **policy_governance_info** メトリックを収集します。可観測性を有効にすると、コンポーネントはいくつかの集約データを収集します。

注記: 可観測性を有効にする場合は、Grafana **Explore** ページからメトリクスのクエリーを入力します。ポリシーの作成時に、**root** ポリシーを作成します。フレームワークは、ルートポリシー、**Placement** リソース、および **PlacementBindings** リソースを監視して、**propagated** ポリシーを作成する場所の情報を取得し、ポリシーをマネージドクラスターに配布します。

ルートポリシーと伝播ポリシーにはいずれも、ポリシーが準拠している場合は **0** のメトリックが、コンプライアンス違反の場合は **1** が記録されます。

policy_governance_info メトリックは、以下のラベルを使用します。

- **Type**: ラベルの値は **root** または **propagate** を使用できます。
- **policy**: 関連付けられたルートポリシーの名前。
- **policy_namespace**: ルートポリシーが定義されているハブクラスター上の namespace。
- **cluster_namespace**: ポリシーの分散先のクラスターの namespace。

これらのラベルと値は、クラスターで発生している、追跡が困難なイベントを表示できるクエリーを有効にします。

注記: メトリクスが必要なく、パフォーマンスやセキュリティーに懸念がある場合は、メトリクスの収集を無効にできます。Propagator デプロイメントで **DISABLE_REPORT_METRICS** 環境変数を **true** に設定します。**policy_governance_info** メトリックを、可観測性の許可リストにカスタムメトリックとして追加することもできます。詳細は、カスタムメトリクスの追加 を参照してください。

2.4.5.2. メトリック: config_policies_evaluation_duration_seconds

config_policies_evaluation_duration_seconds ヒストグラムは、クラスターで評価する準備ができているすべての設定ポリシーを処理するのにかかる秒数を追跡します。次のメトリックを使用して、ヒストグラムをクエリーします。

- **config_policies_evaluation_duration_seconds_bucket**: バケットは累積的であり、次の可能なエントリーで秒を表します: 1、3、9、10.5、15、30、60、90、120、180、300、450、600、およびそれ以上。
- **config_policies_evaluation_duration_seconds_count**: すべてのイベントの数。
- **config_policies_evaluation_duration_seconds_sum**: すべての値の合計。

config_policies_evaluation_duration_seconds メトリックを使用して、頻繁に評価する必要がないリソース集約型ポリシーの **ConfigurationPolicy evaluationInterval** 設定を変更する必要があるかどうかを判断します。また、Kubernetes API サーバーでのリソース使用率が高くなる代わりに、同時実行数を増やすこともできます。詳細については、**同時実行の設定** セクションを参照してください。

設定ポリシーの評価に使用された時間に関する情報を取得するには、次の式のような Prometheus クエリーを実行します。

rate(config_policies_evaluation_duration_seconds_sum[10m])/rate (config_policies_evaluation_duration_seconds_count[10m]

open-cluster-management-agent-addon namespace のマネージドクラスターで実行されている **config-policy-controller** Pod がメトリックを計算します。デフォルトでは、**config-policy-controller** はメトリックを observability に送信しません。

2.4.6. 設定変更を確認する

コントローラーを使用して新しい設定を適用すると、**ManifestApplied** パラメーターが **ManagedClusterAddOn** で更新されます。その状態のタイムスタンプは、設定を正しく確認するのに役立ちます。たとえば、次のコマンドは、**local-cluster** の **cert-policy-controller** がいつ更新されたかを確認できます。

```
oc get -n local-cluster managedclusteraddon cert-policy-controller | grep -B4 'type: ManifestApplied'
```

次の出力が表示される場合があります。

```
- lastTransitionTime: "2023-01-26T15:42:22Z"
  message: manifests of addon are applied successfully
  reason: AddonManifestApplied
  status: "True"
  type: ManifestApplied
```

2.4.7. 関連情報

- Kubernetes 設定ポリシーコントローラー を参照してください。
- その他のトピックは、Governance トピックに戻ってください。
- このトピックの最初の ポリシーコントローラーの詳細設定 に戻ります。

2.5. ポリシーコンプライアンス履歴 (テクノロジープレビュー)

ポリシーコンプライアンス履歴 API は、Red Hat Advanced Cluster Management for Kubernetes のポリシーコンプライアンスイベントをクエリー可能な形式で長期間保存する場合に使用できる、オプションのテクニカルプレビュー機能です。この API を使用すると、**spec** フィールドなどの追加の詳細を取得して、ポリシーを監査およびトラブルシューティングすることができます。また、ポリシーが無効化されたりクラスターから削除されたりしたときに、コンプライアンスイベントを取得できます。ポリシーコンプライアンス履歴 API は、監査とトラブルシューティングに役立つ、ポリシーコンプライアンスイベントのコンマ区切り値 (CSV) スプレッドシートを生成することもできます。

ポリシーコンプライアンス履歴 API は、さらなる監査とトラブルシューティングのために、ポリシーコンプライアンスイベントのコンマ区切り値 (CSV) スプレッドシートを生成することもできます。

2.5.1. 前提条件

- ポリシーコンプライアンス履歴 API には、バージョン 13 以降の PostgreSQL サーバーが必要です。
  Red Hat がサポートする方式は、**registry.redhat.io/rhel9/postgresql-15** コンテナーイメージ、**registry.redhat.io/rhel8/postgresql-13** コンテナーイメージ、**postgresql-server** RPM、または **postgresql/server** モジュールの使用です。各方式のセットアップと設定については、該当する Red Hat 公式ドキュメントを確認してください。ポリシーコンプライアンス履歴 API は、あらゆる標準 PostgreSQL と互換性があり、Red Hat が公式にサポートする製品に限定されません。
- この PostgreSQL サーバーには、Red Hat Advanced Cluster Management ハブクラスターからアクセスできる必要があります。PostgreSQL サーバーがハブクラスターの外部で実行されている場合は、ハブクラスターが PostgreSQL サーバーのポート 5432 に接続できるように、ルーティングとファイアウォールの設定を確認してください。このポートは、PostgreSQL 設定で上書きされている場合、異なる値である場合があります。

2.5.2. コンプライアンス履歴 API の有効化

ポリシーコンプライアンスイベントを API に記録するようにマネージドクラスターを設定します。これは、すべてのクラスターまたはクラスターのサブセットで有効にできます。以下の手順を実行します。

1. PostgreSQL サーバーをクラスター管理者として設定します。Red Hat Advanced Cluster Management ハブクラスターに PostgreSQL をデプロイした場合は、**psql** コマンドを使用するために PostgreSQL ポートを一時的にポート転送します。以下のコマンドを実行します。

```
oc -n <PostgreSQL namespace> port-forward <PostgreSQL pod name> 5432:5432
```

2. 別のターミナルで、次のようなコマンドを使用して PostgreSQL サーバーにローカルで接続します。

```
psql 'postgres://postgres:@127.0.0.1:5432/postgres'
```

3. 次の SQL ステートメントを使用して、Red Hat Advanced Cluster Management ハブクラスターのユーザーとデータベースを作成します。

```
CREATE USER "rhacm-policy-compliance-history" WITH PASSWORD '<replace with password>';
CREATE DATABASE "rhacm-policy-compliance-history" WITH OWNER="rhacm-policy-compliance-history";
```

4. ポリシーコンプライアンス履歴 API にこのデータベースを使用するには、**governance-policy-database Secret** リソースを作成します。以下のコマンドを実行します。

```
oc -n open-cluster-management create secret generic governance-policy-database \ 1
  --from-literal="user=rhacm-policy-compliance-history" \
  --from-literal="password=rhacm-policy-compliance-history" \
  --from-literal="host=<replace with host name of the Postgres server>" \ 2
  --from-literal="dbname=ocm-compliance-history" \
 --from-literal="sslmode=verify-full" \
  --from-file="ca=<replace>" 3
```

1. Red Hat Advanced Cluster Management がインストールされている名前空間を追加します。デフォルトでは、Red Hat Advanced Cluster Management は **open-cluster-management** 名前空間にインストールされます。

2. PostgresQL サーバーのホスト名を追加します。PostgreSQL サーバーを Red Hat Advanced Cluster Management ハブクラスターにデプロイし、クラスターの外部に公開されていない場合は、ホスト値に **Service** オブジェクトを使用できます。形式は **<service name>.<namespace>.svc** です。このアプローチは、Red Hat Advanced Cluster Management ハブクラスターのネットワークポリシーに依存することに注意してください。

3. PostgreSQL サーバーの TLS 証明書に署名した証明機関の証明書ファイルを **ca** データフィールドに指定する必要があります。この値を指定しない場合は、それに応じて **sslmode** 値を変更する必要があります。ただし、これはデータベース接続のセキュリティーが低下するため、推奨しません。

5. Red Hat Advanced Cluster Management ハブクラスターの復元操作のために **Secret** リソースをバックアップするには、**cluster.open-cluster-management.io/backup** ラベルを追加します。以下のコマンドを実行します。

```
oc -n open-cluster-management label secret governance-policy-database cluster.open-cluster-management.io/backup=""
```

6. PostgreSQL 接続をさらにカスタマイズするには、**connectionURL** データフィールドを直接使用し、PostgreSQL 接続 URI の形式で値を指定します。パスワード内の特殊文字は URL エンコードする必要があります。1 つの方法として、Python を使用してパスワードの URL エンコード形式を生成する方法があります。たとえば、パスワードが **$uper<Secr&t%>** の場合、次の Python コマンドを実行して出力 **%24uper%3CSecr%26t%25%3E** を取得します。

```
python -c 'import urllib.parse; import sys; print(urllib.parse.quote(sys.argv[1]))' '$uper<Secr&t%>'
```

7. **governance-policy-database Secret** を作成した後、コマンドを実行してポリシーコンプライアンス履歴 API をテストします。OpenShift **Route** オブジェクトが同じ namespace に自動的に作成されます。Red Hat Advanced Cluster Management ハブクラスターのルートが信頼できる証明書を利用していない場合は、curl コマンドで **-k** フラグを指定して TLS 検証をスキップすることもできます。ただし、これは推奨されません。

```
curl -H "Authorization: Bearer $(oc whoami --show-token)" \
   "https://$(oc -n open-cluster-management get route governance-history-api -o jsonpath='{.spec.host}')/api/v1/compliance-events"
```

- 成功した場合、curl コマンドは次のメッセージのような値を返します。

```
{"data":[],"metadata":{"page":1,"pages":0,"per_page":20,"total":0}}
```

- 成功しなかった場合、curl コマンドは次の 2 つのメッセージのいずれかを返す可能性があります。

```
{"message":"The database is unavailable"}
```

```
{"message":"Internal Error"}
```

a. メッセージを受け取った場合は、次のコマンドを使用して、**open-cluster-management** 名前空間内の Kubernetes イベントを表示します。

```
oc -n open-cluster-management get events --field-selector reason=OCMComplianceEventsDBError
```

b. イベントから **governance-policy-propagator** ログを表示する指示を受け取った場合は、次のコマンドを実行します。

```
oc -n open-cluster-management logs -l name=governance-policy-propagator -f
```

c. ユーザー、パスワード、またはデータベースが正しく指定されていないことを示すエラーメッセージが表示される場合があります。次のメッセージの例を参照してください。

```
2024-03-05T12:17:14.500-0500 info compliance-events-api complianceeventsapi/complianceeventsapi_controller.go:261 The database connection failed: pq: password authentication failed for user "rhacm-policy-compliance-history"
```

d. 次のコマンドを使用して、**governance-policy-database Secret** リソースを正しい PostgreSQL 接続設定で更新します。

```
oc -n open-cluster-management edit secret governance-policy-database
```

2.5.3. コンプライアンス履歴 API URL を設定する

マネージドクラスターで機能を有効にするには、ポリシーコンプライアンス履歴 API URL を設定します。以下の手順を実行します。

1. 次のコマンドを使用して、ポリシーコンプライアンス履歴 API の外部 URL を取得します。

```
echo "https://$(oc -n open-cluster-management get route governance-history-api -o=jsonpath='{.spec.host}')"
```

出力は、次の情報のようになり、Red Hat Advanced Cluster Management ハブクラスターのドメイン名を含んでいます。

```
https://governance-history-api-open-cluster-management.apps.openshift.redhat.com
```

2. 次の例のような **AddOnDeploymentConfig** オブジェクトを作成します。

```
apiVersion: addon.open-cluster-management.io/v1alpha1
kind: AddOnDeploymentConfig
metadata:
  name: governance-policy-framework
  namespace: open-cluster-management
spec:
  customizedVariables:
    - name: complianceHistoryAPIURL
      value: <replace with URL from previous command>
```

- **value** パラメーター値は、コンプライアンス履歴の外部 URL に置き換えます。

2.5.3.1. すべてのマネージドクラスターで有効にする

すべてのマネージドクラスターでコンプライアンス履歴 API を有効にして、マネージドクラスターからのコンプライアンスイベントを記録します。以下の手順を実行します。

1. 次のコマンドを使用して、**governance-policy-framework ClusterManagementAddOn** オブジェクトが **AddOnDeploymentConfig** を使用するように設定します。

```
oc edit ClusterManagementAddOn governance-policy-framework
```

2. **spec.supportedConfigs** 配列を追加または更新します。リソースの設定は次のようになります。

```
  - group: addon.open-cluster-management.io
    resource: addondeploymentconfigs
    defaultConfig:
      name: governance-policy-framework
      namespace: open-cluster-management
```

2.5.3.2. 単一のマネージドクラスターを有効にする

単一のマネージドクラスターでコンプライアンス履歴 API を有効にして、マネージドクラスターからのコンプライアンスイベントを記録します。以下の手順を実行します。

1. マネージドクラスター名前空間で、**governance-policy-framework ManagedClusterAddOn** リソースを設定します。次のコマンドを使用して、Red Hat Advanced Cluster Management ハブクラスターから次のコマンドを実行します。

```
oc -n <manage-cluster-namespace> edit ManagedClusterAddOn governance-policy-framework
```

   - **<manage-cluster-namespace>** プレースホルダーは、有効にするマネージドクラスターの名前に置き換えます。

2. **spec.configs** 配列を追加または更新し、次の例のようなエントリーを含めます。

```
- group: addon.open-cluster-management.io
  resource: addondeploymentconfigs
  name: governance-policy-framework
  namespace: open-cluster-management
```

3. 設定を確認するには、マネージドクラスター上のデプロイメントで **--compliance-api-url** コンテナー引数が使用されていることを確認します。以下のコマンドを実行します。

```
oc -n open-cluster-management-agent-addon get deployment governance-policy-framework -o jsonpath='{.spec.template.spec.containers[1].args}'
```

   出力は次の例のような内容になります。

```
["--enable-lease=true","--hub-cluster-configfile=/var/run/klusterlet/kubeconfig","--leader-elect=false","--log-encoder=console","--log-level=0","--v=-1","--evaluation-concurrency=2","--client-max-qps=30","--client-burst=45","--disable-spec-sync=true","--cluster-namespace=local-cluster","--compliance-api-url=https://governance-history-api-open-cluster-management.apps.openshift.redhat.com"]
```

   新しいポリシーコンプライアンスイベントが、すべてポリシーコンプライアンス履歴 API に記録されます。

   a. 特定のマネージドクラスターのポリシーコンプライアンスイベントが記録されていない場合は、影響を受けるマネージドクラスターの **governance-policy-framework** ログを表示します。

```
oc -n open-cluster-management-agent-addon logs deployment/governance-policy-framework -f
```

   b. 次のメッセージに類似したログメッセージが表示されます。**message** 値が空の場合は、ポリシーコンプライアンス履歴 API の URL が正しくないか、ネットワーク通信の問題が発生しています。

```
024-03-05T19:28:38.063Z        info    policy-status-sync      statussync/policy_status_sync.go:750    Failed to record the compliance event with the compliance API. Will requeue.       {"statusCode": 503, "message": ""}
```

c. ポリシーコンプライアンス履歴 API URL が正しくない場合は、次のコマンドを使用してハブクラスターの URL を編集します。

```
oc -n open-cluster-management edit AddOnDeploymentConfig governance-policy-framework
```

注記: ネットワーク通信の問題が発生した場合は、ネットワークインフラストラクチャーに基づいて問題を診断する必要があります。

2.5.4. 関連情報

- ポリシーコンプライアンス履歴 API (テクノロジープレビュー) を参照してください。

2.6. サポート対象のポリシー

Red Hat Advanced Cluster Management for Kubernetes でポリシーの作成および管理時に、ハブクラスターでのルール、プロセス、制御の定義方法を説明するサポート対象のポリシーを確認します。

2.6.1. サンプル設定ポリシーの表

次のサンプル設定ポリシーを表示します。

表2.6 設定ポリシーの表のリスト

ポリシーのサンプル	説明
Namespace ポリシー	環境の分離と Namespace を使用した命名の一貫性を確保します。Kubernetes Namespace のドキュメント を参照してください。
Pod ポリシー	クラスターのワークロード設定を確認します。Kubernetes Pod のドキュメント を参照してください。
メモリー使用状況のポリシー	制限範囲を使用してワークロードリソースの使用を制限します。制限範囲のドキュメント を参照してください。
Pod セキュリティーポリシー (非推奨)	一貫したワークロードセキュリティーを確保します。Kubernetes Pod セキュリティーポリシーのドキュメント を参照してください。
ロールポリシー ロールバインディングポリシー	ロールとロールバインディングを使用して、ロールのアクセス権限とバインディングを管理します。Kubernetes RBAC のドキュメント を参照してください。
SCC (Security Context Constraints) ポリシー	Security Context Constraints を使用してワークロードのアクセス権限を管理します。OpenShift Container Platform ドキュメントの セキュリティーコンテキスト制約の管理ドキュメント を参照してください。

ポリシーのサンプル	説明
ETCD 暗号化ポリシー	etcd 暗号化でデータセキュリティーを確保します。OpenShift Container Platform ドキュメントの etcd データの暗号化 を参照してください。
Compliance Operator ポリシー	Compliance Operator をデプロイして、OpenSCAP を利用するクラスターのコンプライアンス状態をスキャンして適用します。OpenShift Container Platform ドキュメントの Compliance Operator について を参照してください。
Compliance Operator E8 のスキャン	Compliance Operator ポリシーを適用した後、Essential 8 (E8) スキャンをデプロイして、E8 セキュリティープロファイルへの準拠を確認します。OpenShift Container Platform ドキュメントの Compliance Operator について を参照してください。
Compliance Operator CIS のスキャン	Compliance Operator ポリシーを適用した後、Center for Internet Security (CIS) スキャンをデプロイメントして、CIS セキュリティープロファイルへの準拠を確認します。OpenShift Container Platform ドキュメントの Compliance Operator について を参照してください。
イメージ脆弱性ポリシー	Container Security Operator をデプロイし、クラスターで実行されている Pod で既知のイメージの脆弱性を検出します。Container Security Operator GitHub リポジトリーを参照してください。
Gatekeeper Operator の配置	Gatekeeper は、Open Policy Agent (OPA) ポリシーエンジンによって実行されるカスタムリソース定義ベースのポリシーを適用する受付 Webhook です。Gatekeeper のドキュメントを参照してください。
Gatekeeper のコンプライアンスポリシー	Gatekeeper をクラスターにデプロイした後、このサンプルの Gatekeeper ポリシーをデプロイして、クラスター上に作成された namespace が指定どおりにラベル付けされるようにします。

Red Hat OpenShift Container Platform 4.x は、Red Hat Advanced Cluster Management 設定ポリシーもサポートします。

ポリシーがどのように適用されるかを確認するには、ポリシーに関する以下のドキュメントを参照してください。

- イメージ脆弱性ポリシー
- メモリー使用状況のポリシー

- Namespace ポリシー
- Pod ポリシー
- Pod のセキュリティーポリシー
- ロールポリシー
- Role binding ポリシー
- SCC (Security Context Constraints) ポリシー
- ETCD 暗号化ポリシー
- Compliance Operator ポリシー
- E8 スキャンポリシー
- OpenShift CIS スキャンポリシー
- ポリシーセットコントローラー
- Red Hat OpenShift Platform Plus ポリシーセット

他のトピックについては、ガバナンス を参照してください。

2.6.2. メモリー使用状況のポリシー

Kubernetes 設定ポリシーコントローラーは、メモリー使用状況ポリシーのステータスを監視します。メモリー使用状況ポリシーを使用して、メモリーおよびコンピュートの使用量を制限または制約します。詳細は、Kubernetes ドキュメント の **Limit Ranges** を参照してください。

以下のセクションでは、メモリー使用状況ポリシーの設定について説明します。

- メモリー使用状況ポリシー YAML の設定
- メモリー使用状況のポリシーの表
- メモリー使用状況ポリシーの例

2.6.2.1. メモリー使用状況ポリシー YAML の設定

メモリー使用状況ポリシーは、以下の YAML ファイルのようになります。

```
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  name:
  namespace:
  annotations:
    policy.open-cluster-management.io/standards:
    policy.open-cluster-management.io/categories:
    policy.open-cluster-management.io/controls:
    policy.open-cluster-management.io/description:
spec:
  remediationAction:
```

```
disabled:
policy-templates:
  - objectDefinition:
      apiVersion: policy.open-cluster-management.io/v1
      kind: ConfigurationPolicy
      metadata:
        name:
      spec:
        remediationAction:
        severity:
        namespaceSelector:
          exclude:
          include:
          matchLabels:
          matchExpressions:
        object-templates:
          - complianceType: mustonlyhave
            objectDefinition:
              apiVersion: v1
              kind: LimitRange
              metadata:
                name:
              spec:
                limits:
                - default:
                    memory:
                  defaultRequest:
                    memory:
                  type:
      ...
```

2.6.2.2. メモリー使用状況のポリシーの表

表2.7 パラメーターの表

フィールド	任意または必須	説明
apiVersion	必須	この値は **policy.open-cluster-management.io/v1** に設定します。
kind	必須	ポリシーのタイプを指定するには、値を **Policy** に設定します。
metadata.name	必須	ポリシーリソースを識別する名前。
metadata.namespace	必須	ポリシーの namespace。

フィールド	任意または必須	説明
spec.remediationAction	任意	ポリシーの修正を指定します。パラメーターの値は **enforce** および **inform** です。この値はオプションです。**spec.policy-templates** で提供されるすべての値をオーバーライドするためです。
spec.disabled	必須	この値は **true** または **false** に設定します。**disabled** パラメーターを使用すると、ポリシーを有効または無効にすることができます。
spec.policy-templates[].objectDefinition	必須	マネージドクラスターに評価または適用する必要がある Kubernetes オブジェクトを含む設定ポリシーをリスト表示するために使用されます。

2.6.2.3. メモリー使用状況ポリシーの例

ポリシーのサンプルを確認するには、**policy-limitmemory.yaml** を参照してください。詳細については、セキュリティーポリシーの管理 を参照してください。ポリシーの概要 に関するドキュメントと Kubernetes 設定ポリシーコントローラー を参照して、コントローラーによって監視されるその他の設定ポリシーを確認してください。

2.6.3. Namespace ポリシー

Kubernetes 設定ポリシーコントローラーは、namespace ポリシーのステータスを監視します。Namespace ポリシーを適用し、namespace の特定のルールを定義します。

以下のセクションでは namespace ポリシーの設定について説明します。

- Namespace ポリシー YAML の設定
- Namespace ポリシーテーブル
- Namespace ポリシーの例

2.6.3.1. Namespace ポリシー YAML の設定

```
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  name:
  namespace:
  annotations:
```

```
    policy.open-cluster-management.io/standards:
    policy.open-cluster-management.io/categories:
    policy.open-cluster-management.io/controls:
    policy.open-cluster-management.io/description:
spec:
  remediationAction:
  disabled:
  policy-templates:
    - objectDefinition:
        apiVersion: policy.open-cluster-management.io/v1
        kind: ConfigurationPolicy
        metadata:
          name:
        spec:
          remediationAction:
          severity:
          object-templates:
            - complianceType:
              objectDefinition:
                kind: Namespace
                apiVersion: v1
                metadata:
                  name:
                ...
```

2.6.3.2. Namespace ポリシー YAML の表

フィールド	任意または必須	説明
apiVersion	必須	この値は **policy.open-cluster-management.io/v1** に設定します。
kind	必須	ポリシーのタイプを指定するには、値を **Policy** に設定します。
metadata.name	必須	ポリシーリソースを識別する名前。
metadata.namespace	必須	ポリシーの namespace。
spec.remediationAction	任意	ポリシーの修正を指定します。パラメーターの値は **enforce** および **inform** です。この値はオプションです。**spec.policy-templates** で提供されるすべての値をオーバーライドするためです。

フィールド	任意または必須	説明
spec.disabled	必須	この値は **true** または **false** に設定します。**disabled** パラメーターを使用すると、ポリシーを有効または無効にすることができます。
spec.policy-templates[].objectDefinition	必須	マネージドクラスターに評価または適用する必要がある Kubernetes オブジェクトを含む設定ポリシーをリスト表示するために使用されます。

2.6.3.3. Namespace ポリシーの例

ポリシーのサンプルを確認するには、**policy-namespace.yaml** を参照してください。

詳細については、セキュリティーポリシーの管理 を参照してください。その他の設定ポリシーについては、ポリシーの概要 に関するドキュメントと Kubernetes 設定ポリシーコントローラー を参照してください。

2.6.4. イメージ脆弱性ポリシー

イメージ脆弱性ポリシーを適用し、コンテナーセキュリティー Operator を利用してコンテナーイメージに脆弱性があるかどうかを検出します。このポリシーは、コンテナーセキュリティー Operator がインストールされていない場合は、これをマネージドクラスターにインストールします。

イメージ脆弱性ポリシーは、Kubernetes 設定ポリシーコントローラーがチェックします。セキュリティー Operator の詳細は、Quay リポジトリー の **コンテナーセキュリティー Operator** を参照してください。

注記:

- イメージ脆弱性ポリシーは、非接続インストール中は機能しません。
- イメージ脆弱性ポリシー は、IBM Power および IBM Z アーキテクチャーではサポートされません。これは Quay Container Security Operator に依存します。container-security-operator レジストリーには **ppc64le** または **s390x** のイメージがありません。

詳細は、以下のセクションを参照してください。

- イメージ脆弱性ポリシーの YAML 設定
- イメージ脆弱性ポリシーの例

2.6.4.1. イメージ脆弱性ポリシーの YAML 設定

コンテナーセキュリティー operator ポリシーを作成すると、次のポリシーが含まれます。

- 名前とチャネルを参照するサブスクリプション (**container-security-operator**) を作成するポリシー。この設定ポリシーには、リソースを作成するために **enforce** する

spec.remediationAction が設定されている必要があります。サブスクリプションは、サブスクリプションがサポートするプロファイルをコンテナーとしてプルします。以下の例を参照してください。

```
apiVersion: policy.open-cluster-management.io/v1
kind: ConfigurationPolicy
metadata:
  name: policy-imagemanifestvuln-example-sub
spec:
  remediationAction: enforce  # will be overridden by remediationAction in parent policy
  severity: high
  object-templates:
    - complianceType: musthave
      objectDefinition:
        apiVersion: operators.coreos.com/v1alpha1
        kind: Subscription
        metadata:
          name: container-security-operator
          namespace: openshift-operators
        spec:
          # channel: quay-v3.3 # specify a specific channel if desired
          installPlanApproval: Automatic
          name: container-security-operator
          source: redhat-operators
          sourceNamespace: openshift-marketplace
```

- コンテナーセキュリティー operator のインストールが成功したことを確認するために **ClusterServiceVersion** を監査するための **inform** 設定ポリシー。以下の例を参照してください。

```
apiVersion: policy.open-cluster-management.io/v1
kind: ConfigurationPolicy
metadata:
  name: policy-imagemanifestvuln-status
spec:
  remediationAction: inform  # will be overridden by remediationAction in parent policy
  severity: high
  object-templates:
    - complianceType: musthave
      objectDefinition:
        apiVersion: operators.coreos.com/v1alpha1
        kind: ClusterServiceVersion
        metadata:
          namespace: openshift-operators
        spec:
          displayName: Red Hat Quay Container Security Operator
        status:
          phase: Succeeded   # check the CSV status to determine if operator is running or not
```

- **ImageManifestVuln** オブジェクトがイメージの脆弱性スキャンによって作成されたかどうかを監査する **inform** 設定ポリシー。以下の例を参照してください。

```
apiVersion: policy.open-cluster-management.io/v1
kind: ConfigurationPolicy
metadata:
```

```
  name: policy-imagemanifestvuln-example-imv
spec:
  remediationAction: inform  # will be overridden by remediationAction in parent policy
  severity: high
  namespaceSelector:
    exclude: ["kube-*"]
    include: ["*"]
  object-templates:
    - complianceType: mustnothave # mustnothave any ImageManifestVuln object
      objectDefinition:
        apiVersion: secscan.quay.redhat.com/v1alpha1
        kind: ImageManifestVuln # checking for a Kind
```

2.6.4.2. イメージ脆弱性ポリシーの例

policy-imagemanifestvuln.yaml を参照してください。詳細は、セキュリティーポリシーの管理 を参照してください。設定コントローラーによって監視されるその他の設定ポリシーについては、Kubernetes 設定ポリシーコントローラー を参照してください。

2.6.5. Pod ポリシー

Kubernetes 設定ポリシーコントローラーは、ロールポリシーのステータスを監視します。Pod ポリシーを適用し、Pod のコンテナールールを定義します。この情報を使用するには、Pod がクラスターに存在している必要があります。

以下のセクションでは、Pod ポリシーの設定について説明します。

- Pod ポリシー YAML の設定
- Pod ポリシーの表
- Pod ポリシーの例

2.6.5.1. Pod ポリシー YAML の設定

```
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  name:
  namespace:
  annotations:
    policy.open-cluster-management.io/standards:
    policy.open-cluster-management.io/categories:
    policy.open-cluster-management.io/controls:
    policy.open-cluster-management.io/description:
spec:
  remediationAction:
  disabled:
  policy-templates:
    - objectDefinition:
        apiVersion: policy.open-cluster-management.io/v1
        kind: ConfigurationPolicy
        metadata:
          name:
        spec:
```

```
    remediationAction:
    severity:
    namespaceSelector:
      exclude:
      include:
      matchLabels:
      matchExpressions:
    object-templates:
      - complianceType:
        objectDefinition:
          apiVersion: v1
          kind: Pod
          metadata:
            name:
          spec:
            containers:
            - image:
              name:
          ...
```

2.6.5.2. Pod ポリシーの表

表2.8 パラメーターの表

フィールド	任意または必須	説明
apiVersion	必須	この値は **policy.open-cluster-management.io/v1** に設定します。
kind	必須	ポリシーのタイプを指定するには、値を **Policy** に設定します。
metadata.name	必須	ポリシーリソースを識別する名前。
metadata.namespace	必須	ポリシーの namespace。
spec.remediationAction	任意	ポリシーの修正を指定します。パラメーターの値は **enforce** および **inform** です。この値はオプションです。**spec.policy-templates** で提供されるすべての値をオーバーライドするためです。
spec.disabled	必須	この値は **true** または **false** に設定します。**disabled** パラメーターを使用すると、ポリシーを有効または無効にすることができます。

フィールド	任意または必須	説明
spec.policy-templates[].objectDefinition	必須	マネージドクラスターに評価または適用する必要がある Kubernetes オブジェクトを含む設定ポリシーをリスト表示するために使用されます。

2.6.5.3. Pod ポリシーの例

ポリシーのサンプルを確認するには、 **policy-pod.yaml** を参照してください。

設定コントローラーによって監視される他の設定ポリシーを表示するには、Kubernetes 設定ポリシーコントローラー を参照してください。また、ポリシーの YAML 構造と追加フィールドの完全な説明を確認するには、ポリシーの概要 ドキュメントを参照してください。他のポリシーを管理するには、設定ポリシーの管理 に関するドキュメントに戻ります。

2.6.6. Pod セキュリティーポリシー (非推奨)

Kubernetes 設定ポリシーコントローラーは、Pod セキュリティーポリシーのステータスを監視します。Pod のセキュリティーポリシーを適用して Pod およびコンテナーのセキュリティーを保護します。

以下のセクションでは、Pod セキュリティーポリシーの設定について説明します。

- Pod セキュリティーポリシー YAML の設定
- Pod セキュリティーポリシーの表
- Pod セキュリティーポリシーの例

2.6.6.1. Pod セキュリティーポリシー YAML の設定

```
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  name:
  namespace:
  annotations:
    policy.open-cluster-management.io/standards:
    policy.open-cluster-management.io/categories:
    policy.open-cluster-management.io/controls:
    policy.open-cluster-management.io/description:
spec:
  remediationAction:
  disabled:
  policy-templates:
    - objectDefinition:
        apiVersion: policy.open-cluster-management.io/v1
        kind: ConfigurationPolicy
        metadata:
          name:
        spec:
```

```
    remediationAction:
    severity:
    namespaceSelector:
      exclude:
      include:
      matchLabels:
      matchExpressions:
    object-templates:
      - complianceType:
        objectDefinition:
          apiVersion: policy/v1beta1
          kind: PodSecurityPolicy
          metadata:
            name:
            annotations:
              seccomp.security.alpha.kubernetes.io/allowedProfileNames:
          spec:
            privileged:
            allowPrivilegeEscalation:
            allowedCapabilities:
            volumes:
            hostNetwork:
            hostPorts:
            hostIPC:
            hostPID:
            runAsUser:
            seLinux:
            supplementalGroups:
            fsGroup:
          ...
```

2.6.6.2. Pod セキュリティーポリシーの表

表2.9 パラメーターの表

フィールド	任意または必須	説明
apiVersion	必須	この値は **policy.open-cluster-management.io/v1** に設定します。
kind	必須	ポリシーのタイプを指定するには、値を **Policy** に設定します。
metadata.name	必須	ポリシーリソースを識別する名前。
metadata.namespace	必須	ポリシーの namespace。

フィールド	任意または必須	説明
spec.remediationAction	任意	ポリシーの修正を指定します。パラメーターの値は **enforce** および **inform** です。この値はオプションです。**spec.policy-templates** で提供されるすべての値をオーバーライドするためです。
spec.disabled	必須	この値は **true** または **false** に設定します。**disabled** パラメーターを使用すると、ポリシーを有効または無効にすることができます。
spec.policy-templates[].objectDefinition	必須	マネージドクラスターに評価または適用する必要がある Kubernetes オブジェクトを含む設定ポリシーをリスト表示するために使用されます。

2.6.6.3. Pod セキュリティーポリシーの例

Pod セキュリティーポリシーのサポートは、OpenShift Container Platform 4.13 以降、および Kubernetes v1.25 以降から削除されました。**PodSecurityPolicy** リソースを適用すると、次の非準拠メッセージを受け取る場合があります。

> violation - couldn't find mapping resource with kind PodSecurityPolicy, please check if you have CRD deployed

- 非推奨の通知を含む詳細については、Kubernetes ドキュメント の **Pod セキュリティーポリシー** を参照してください。
- サンプルポリシーを確認するには、**policy-psp.yaml** を参照してください。詳細は、設定ポリシーの管理 を参照してください。
- ポリシーの YAML 構造の完全な説明については、ポリシーの概要 に関するドキュメントを参照してください。また、コントローラーによって監視されるその他の設定ポリシーを表示するには、Kubernetes 設定ポリシーコントローラー を参照してください。

2.6.7. ロールポリシー

Kubernetes 設定ポリシーコントローラーは、ロールポリシーのステータスを監視します。**object-template** にロールを定義して、クラスター内の特定ロールのルールおよびパーミッションを設定します。

以下のセクションでは、ロールポリシーの設定について説明します。

- ロールポリシー YAML の設定

- ロールポリシーの表
- ロールポリシーの例

2.6.7.1. ロールポリシー YAML の設定

```
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  name:
  namespace:
  annotations:
    policy.open-cluster-management.io/standards:
    policy.open-cluster-management.io/categories:
    policy.open-cluster-management.io/controls:
    policy.open-cluster-management.io/description:
spec:
  remediationAction:
  disabled:
  policy-templates:
    - objectDefinition:
        apiVersion: policy.open-cluster-management.io/v1
        kind: ConfigurationPolicy
        metadata:
          name:
        spec:
          remediationAction:
          severity:
          namespaceSelector:
            exclude:
            include:
            matchLabels:
            matchExpressions:
          object-templates:
            - complianceType:
              objectDefinition:
                apiVersion: rbac.authorization.k8s.io/v1
                kind: Role
                metadata:
                  name:
                rules:
                  - apiGroups:
                    resources:
                    verbs:
                ...
```

2.6.7.2. ロールポリシーの表

表2.10 パラメーターの表

フィールド	任意または必須	説明

フィールド	任意または必須	説明
apiVersion	必須	この値は **policy.open-cluster-management.io/v1** に設定します。
kind	必須	ポリシーのタイプを指定するには、値を **Policy** に設定します。
metadata.name	必須	ポリシーリソースを識別する名前。
metadata.namespace	必須	ポリシーの namespace。
spec.remediationAction	任意	ポリシーの修正を指定します。パラメーターの値は **enforce** および **inform** です。この値はオプションです。**spec.policy-templates** で提供されるすべての値をオーバーライドするためです。
spec.disabled	必須	この値は **true** または **false** に設定します。**disabled** パラメーターを使用すると、ポリシーを有効または無効にすることができます。
spec.policy-templates[].objectDefinition	必須	マネージドクラスターに評価または適用する必要がある Kubernetes オブジェクトを含む設定ポリシーをリスト表示するために使用されます。

2.6.7.3. ロールポリシーの例

ロールポリシーを適用して、クラスター内の特定のロールのルールおよびパーミッションを設定します。ロールの詳細は、ロールベースのアクセス制御 を参照してください。ロールポリシーのサンプルを確認するには **policy-role.yaml** を参照してください。

ロールポリシーの管理方法は、設定ポリシーの管理 を参照してください。コントローラーが監視する他の設定ポリシーについては、Kubernetes 設定ポリシーコントローラー を参照してください。

2.6.8. Role binding ポリシー

Kubernetes 設定ポリシーコントローラーは、role binding ポリシーのステータスを監視します。Role Binding ポリシーを適用し、ポリシーをマネージドクラスターの namespace にバインドします。

以下のセクションでは namespace ポリシーの設定について説明します。

- Role Binding ポリシー YAML の設定

- Role Binding ポリシーの表
- Role Binding ポリシーの例

2.6.8.1. Role Binding ポリシー YAML の設定

```
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  name:
  namespace:
  annotations:
    policy.open-cluster-management.io/standards:
    policy.open-cluster-management.io/categories:
    policy.open-cluster-management.io/controls:
    policy.open-cluster-management.io/description:
spec:
  remediationAction:
  disabled:
  policy-templates:
    - objectDefinition:
        apiVersion: policy.open-cluster-management.io/v1
        kind: ConfigurationPolicy
        metadata:
          name:
        spec:
          remediationAction:
          severity:
          namespaceSelector:
            exclude:
            include:
            matchLabels:
            matchExpressions:
          object-templates:
            - complianceType:
              objectDefinition:
                kind: RoleBinding # role binding must exist
                apiVersion: rbac.authorization.k8s.io/v1
                metadata:
                  name:
                subjects:
                - kind:
                  name:
                  apiGroup:
                roleRef:
                  kind:
                  name:
                  apiGroup:
                ...
```

2.6.8.2. Role Binding ポリシーの表

フィールド	任意または必須	説明
apiVersion	必須	この値は **policy.open-cluster-management.io/v1** に設定します。
kind	必須	ポリシーのタイプを指定するには、値を **Policy** に設定します。
metadata.name	必須	ポリシーリソースを識別する名前。
metadata.namespace	必須	ポリシーの namespace。
spec.remediationAction	任意	ポリシーの修正を指定します。パラメーターの値は **enforce** および **inform** です。この値はオプションです。**spec.policy-templates** で提供されるすべての値をオーバーライドするためです。
spec.disabled	必須	この値は **true** または **false** に設定します。**disabled** パラメーターを使用すると、ポリシーを有効または無効にすることができます。
spec.policy-templates[].objectDefinition	必須	マネージドクラスターに評価または適用する必要がある Kubernetes オブジェクトを含む設定ポリシーをリスト表示するために使用されます。

2.6.8.3. Role Binding ポリシーの例

ポリシーのサンプルを確認するには、 **policy-rolebinding.yaml** を参照してください。ポリシーの YAML 構造と追加フィールドの完全な説明については、ポリシーの概要 に関するドキュメントを参照してください。その他の設定ポリシーについては、Kubernetes 設定ポリシーコントローラー のドキュメントを参照してください。

2.6.9. SCC (Security Context Constraints) ポリシー

Kubernetes 設定ポリシーコントローラーは、SCC (Security Context Constraints) ポリシーのステータスを監視します。SCC (Security Context Constraints) ポリシーを適用し、ポリシーで条件を定義して Pod のパーミッションを制御します。

以下のセクションで、SCC ポリシーについての詳細を説明します。

- SCC ポリシー YAML の設定

- SCC ポリシーの表
- SCC ポリシーの例

2.6.9.1. SCC ポリシー YAML の設定

```
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  name:
  namespace:
  annotations:
    policy.open-cluster-management.io/standards:
    policy.open-cluster-management.io/categories:
    policy.open-cluster-management.io/controls:
    policy.open-cluster-management.io/description:
spec:
  remediationAction:
  disabled:
  policy-templates:
    - objectDefinition:
        apiVersion: policy.open-cluster-management.io/v1
        kind: ConfigurationPolicy
        metadata:
          name:
        spec:
          remediationAction:
          severity:
          namespaceSelector:
            exclude:
            include:
            matchLabels:
            matchExpressions:
          object-templates:
            - complianceType:
              objectDefinition:
                apiVersion: security.openshift.io/v1
                kind: SecurityContextConstraints
                metadata:
                  name:
                allowHostDirVolumePlugin:
                allowHostIPC:
                allowHostNetwork:
                allowHostPID:
                allowHostPorts:
                allowPrivilegeEscalation:
                allowPrivilegedContainer:
                fsGroup:
                readOnlyRootFilesystem:
                requiredDropCapabilities:
                runAsUser:
                seLinuxContext:
                supplementalGroups:
                users:
                volumes:
                ...
```

■

2.6.9.2. SCC ポリシーの表

フィールド	任意または必須	説明
apiVersion	必須	この値は **policy.open-cluster-management.io/v1** に設定します。
kind	必須	ポリシーのタイプを指定するには、値を **Policy** に設定します。
metadata.name	必須	ポリシーリソースを識別する名前。
metadata.namespace	必須	ポリシーの namespace。
spec.remediationAction	任意	ポリシーの修正を指定します。パラメーターの値は **enforce** および **inform** です。この値はオプションです。**spec.policy-templates** で提供されるすべての値をオーバーライドするためです。
spec.disabled	必須	この値は **true** または **false** に設定します。**disabled** パラメーターを使用すると、ポリシーを有効または無効にすることができます。
spec.policy-templates[].objectDefinition	必須	マネージドクラスターに評価または適用する必要がある Kubernetes オブジェクトを含む設定ポリシーをリスト表示するために使用されます。

SCC ポリシーの内容の説明は、OpenShift Container Platform ドキュメントの SCC (Security Context Constraints) の管理 を参照してください。

2.6.9.3. SCC ポリシーの例

SCC (Security Context Constraints) ポリシーを適用し、ポリシーで条件を定義して Pod のパーミッションを制御します。詳細は、SCC (Security Context Constraints) の管理 を参照してください。

ポリシーのサンプルを確認するには、 **policy-scc.yaml** を参照してください。ポリシーの YAML 構造と追加フィールドの完全な説明については、ポリシーの概要 に関するドキュメントを参照してください。その他の設定ポリシーについては、Kubernetes 設定ポリシーコントローラー のドキュメントを参照してください。

2.6.10. ETCD 暗号化ポリシー

etcd-encryption ポリシーを適用して、ETCD データストアで機密データを検出するか、機密データの暗号化を有効にします。Kubernetes 設定ポリシーコントローラーは、**etcd-encryption** ポリシーのステータスを監視します。詳細は、OpenShift Container Platform ドキュメントの etcd データの暗号化を参照してください。**注記**: ETCD 暗号化ポリシーは、Red Hat OpenShift Container Platform 4 以降のみをサポートします。

以下のセクションでは、 **etcd-encryption** ポリシーの設定について説明します。

- ETCD 暗号化ポリシーの YAML 設定
- ETCD 暗号化ポリシーの表
- ETCD 暗号化ポリシーの例

2.6.10.1. ETCD 暗号化ポリシーの YAML 設定

etcd-encryption ポリシーは、以下の YAML ファイルのようになります。

```
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  name:
  namespace:
  annotations:
    policy.open-cluster-management.io/standards:
    policy.open-cluster-management.io/categories:
    policy.open-cluster-management.io/controls:
    policy.open-cluster-management.io/description:
spec:
  remediationAction:
  disabled:
  policy-templates:
    - objectDefinition:
        apiVersion: policy.open-cluster-management.io/v1
        kind: ConfigurationPolicy
        metadata:
          name:
        spec:
          remediationAction:
          severity:
          object-templates:
            - complianceType:
              objectDefinition:
                apiVersion: config.openshift.io/v1
                kind: APIServer
                metadata:
                  name:
                spec:
                  encryption:
                ...
```

2.6.10.2. ETCD 暗号化ポリシーの表

表2.11 パラメーターの表

フィールド	任意または必須	説明
apiVersion	必須	この値は **policy.open-cluster-management.io/v1** に設定します。
kind	必須	ポリシーのタイプを指定するには、値を **Policy** に設定します。
metadata.name	必須	ポリシーリソースを識別する名前。
metadata.namespace	必須	ポリシーの namespace。
spec.remediationAction	任意	ポリシーの修正を指定します。パラメーターの値は **enforce** および **inform** です。この値はオプションです。**spec.policy-templates** で提供されるすべての値をオーバーライドするためです。
spec.disabled	必須	この値は **true** または **false** に設定します。**disabled** パラメーターを使用すると、ポリシーを有効または無効にすることができます。
spec.policy-templates[].objectDefinition	必須	マネージドクラスターに評価または適用する必要がある Kubernetes オブジェクトを含む設定ポリシーをリスト表示するために使用されます。

2.6.10.3. ETCD 暗号化ポリシーの例

ポリシーのサンプルについては、**policy-etcdencryption.yaml** を参照してください。ポリシーおよび設定ポリシーフィールドの詳細は、ポリシーの概要 ドキュメントと Kubernetes 設定ポリシーコントローラー を参照してください。

2.6.11. Compliance Operator ポリシー

Compliance Operator を使用すると、多数の技術実装の検査を自動化し、それらを業界標準、ベンチマーク、およびベースラインの特定の側面と比較できます。Compliance Operator は監査人ではありません。このようなさまざまな標準に対する準拠または認定を実現するには、Qualified Security Assessor (QSA)、Joint Authorization Board (JAB)、または業界で認められたその他の規制当局など、認定監査機関と協力して、環境を評価する必要があります。

Compliance Operator から生成される推奨事項は、そのような標準に関する一般に利用可能な情報と実践に基づいており、修復に役立つ可能性がありますが、実際のコンプライアンスはユーザーの責任となります。認定監査人と協力して標準への準拠を実現します。

最新の更新については、Compliance Operator リリースノート を参照してください。

2.6.11.1. Compliance Operator ポリシーの概要

Compliance Operator ポリシーを使用して、マネージドクラスターに Compliance Operator をインストールできます。Compliance Operator ポリシーは、Kubernetes 設定ポリシーとして Red Hat Advanced Cluster Management に作成されます。OpenShift Container Platform は、Compliance Operator ポリシーをサポートします。

注記: Compliance Operator ポリシー は、IBM Power または IBM Z アーキテクチャーではサポートされていない OpenShift Container Platform Compliance Operator に依存します。Compliance Operator の詳細は、OpenShift Container Platform ドキュメントの Compliance Operator について を参照してください。

2.6.11.2. Compliance Operator のリソース

Compliance Operator ポリシーを作成すると、次のリソースが作成されます。

- Operator インストール用の Compliance Operator namespace (**openshift-compliance**):

```
apiVersion: policy.open-cluster-management.io/v1
kind: ConfigurationPolicy
metadata:
  name: comp-operator-ns
spec:
  remediationAction: inform # will be overridden by remediationAction in parent policy
  severity: high
  object-templates:
    - complianceType: musthave
      objectDefinition:
        apiVersion: v1
        kind: Namespace
        metadata:
          name: openshift-compliance
```

- 対象の namespace を指定する Operator グループ (**compliance-operator**):

```
apiVersion: policy.open-cluster-management.io/v1
kind: ConfigurationPolicy
metadata:
  name: comp-operator-operator-group
spec:
  remediationAction: inform # will be overridden by remediationAction in parent policy
  severity: high
  object-templates:
    - complianceType: musthave
      objectDefinition:
        apiVersion: operators.coreos.com/v1
        kind: OperatorGroup
        metadata:
          name: compliance-operator
          namespace: openshift-compliance
        spec:
          targetNamespaces:
            - openshift-compliance
```

- 名前とチャネルを参照するためのサブスクリプション (**comp-operator-subscription**)。サブスクリプションは、サポートするプロファイルをコンテナーとしてプルします。以下のサンプルを参照してください。**4.x** は、現在のバージョンに置き換えます。

```
apiVersion: policy.open-cluster-management.io/v1
kind: ConfigurationPolicy
metadata:
  name: comp-operator-subscription
spec:
  remediationAction: inform  # will be overridden by remediationAction in parent policy
  severity: high
  object-templates:
    - complianceType: musthave
      objectDefinition:
        apiVersion: operators.coreos.com/v1alpha1
        kind: Subscription
        metadata:
          name: compliance-operator
          namespace: openshift-compliance
        spec:
          channel: "4.x"
          installPlanApproval: Automatic
          name: compliance-operator
          source: redhat-operators
          sourceNamespace: openshift-marketplace
```

Compliance Operator ポリシーをインストールすると、**compliance-operator**、**ocp4**、および **rhcos4** の Pod が作成されます。**policy-compliance-operator-install.yaml** のサンプルを参照してください。

2.6.11.3. 関連情報

- 詳細は、OpenShift Container Platform ドキュメントの Compliance Operator の管理 を参照してください。
- Compliance Operator をインストールした後に、E8 スキャンポリシーと OpenShift CIS スキャンポリシーを作成して適用することもできます。詳細は、E8 スキャンポリシー および OpenShift CIS スキャンポリシー を参照してください。
- Compliance Operator ポリシーの管理に関する詳細は、セキュリティポリシーの管理 を参照してください。設定ポリシーの他のトピックについては、Kubernetes 設定ポリシーコントローラー を参照してください。

2.6.12. E8 スキャンポリシー

Essential 8 (E8) スキャンポリシーは、マスターノードとワーカーノードが E8 セキュリティープロファイルに準拠しているかどうかを確認するスキャンをデプロイします。E8 スキャンポリシーを適用するには、Compliance Operator をインストールする必要があります。

E8 スキャンポリシーは、Kubernetes 設定ポリシーとして Red Hat Advanced Cluster Management に作成されます。OpenShift Container Platform は E8 スキャンポリシーをサポートします。詳細は、OpenShift Container Platform ドキュメントの Compliance Operator の管理 を参照してください。

2.6.12.1. E8 スキャンポリシーリソース

E8 スキャンポリシーを作成すると、次のリソースが作成されます。

- スキャンするプロファイルを特定する **ScanSettingBinding** リソース (**e8**):

```
apiVersion: policy.open-cluster-management.io/v1
kind: ConfigurationPolicy
metadata:
  name: compliance-suite-e8
spec:
  remediationAction: inform
  severity: high
  object-templates:
    - complianceType: musthave # this template checks if scan has completed by checking the
status field
      objectDefinition:
        apiVersion: compliance.openshift.io/v1alpha1
        kind: ScanSettingBinding
        metadata:
          name: e8
          namespace: openshift-compliance
        profiles:
        - apiGroup: compliance.openshift.io/v1alpha1
          kind: Profile
          name: ocp4-e8
        - apiGroup: compliance.openshift.io/v1alpha1
          kind: Profile
          name: rhcos4-e8
        settingsRef:
          apiGroup: compliance.openshift.io/v1alpha1
          kind: ScanSetting
          name: default
```

- **status** フィールドを確認してスキャンが完了したかどうかを確認する **ComplianceSuite** リソース (**compliance-suite-e8**):

```
apiVersion: policy.open-cluster-management.io/v1
kind: ConfigurationPolicy
metadata:
  name: compliance-suite-e8
spec:
  remediationAction: inform
  severity: high
  object-templates:
    - complianceType: musthave # this template checks if scan has completed by checking the
status field
      objectDefinition:
        apiVersion: compliance.openshift.io/v1alpha1
        kind: ComplianceSuite
        metadata:
          name: e8
          namespace: openshift-compliance
        status:
          phase: DONE
```

- **ComplianceCheckResult** カスタムリソース (CR) を確認してスキャンスイートの結果を報告する **ComplianceCheckResult** リソース (**compliance-suite-e8-results**):

```
apiVersion: policy.open-cluster-management.io/v1
kind: ConfigurationPolicy
metadata:
  name: compliance-suite-e8-results
spec:
  remediationAction: inform
  severity: high
  object-templates:
    - complianceType: mustnothave # this template reports the results for scan suite: e8 by
looking at ComplianceCheckResult CRs
      objectDefinition:
        apiVersion: compliance.openshift.io/v1alpha1
        kind: ComplianceCheckResult
        metadata:
          namespace: openshift-compliance
          labels:
            compliance.openshift.io/check-status: FAIL
            compliance.openshift.io/suite: e8
```

注記: 自動修復はサポート対象です。**ScanSettingBinding** リソースを作成するには修復アクションを **enforce** に設定します。

policy-compliance-operator-e8-scan.yaml の サンプルを参照してください。詳細は、セキュリティーポリシーの管理 を参照してください。**注記:** E8 ポリシーの削除後に、これはターゲットクラスターから削除されます。

2.6.13. OpenShift CIS スキャンポリシー

OpenShift CIS スキャンポリシーは、マスターとワーカーノードをチェックして、OpenShift CIS セキュリティーベンチマークに準拠しているかどうかを確認するスキャンをデプロイします。OpenShift CIS ポリシーを適用するには、Compliance Operator をインストールする必要があります。

OpenShift CIS ポリシーは、Kubernetes 設定ポリシーとして Red Hat Advanced Cluster Management に作成されます。OpenShift Container Platform では、OpenShift CIS スキャンポリシーがサポートされます。詳細は、OpenShift Container Platform ドキュメントの Compliance Operator について を参照してください。

2.6.13.1. OpenShift CIS リソース

OpenShift CIS スキャンポリシーを作成すると、次のリソースが作成されます。

- スキャンするプロファイルを特定する **ScanSettingBinding** リソース (**cis**):

```
apiVersion: policy.open-cluster-management.io/v1
kind: ConfigurationPolicy
metadata:
  name: compliance-cis-scan
spec:
  remediationAction: inform
  severity: high
  object-templates:
    - complianceType: musthave # this template creates ScanSettingBinding:cis
      objectDefinition:
        apiVersion: compliance.openshift.io/v1alpha1
        kind: ScanSettingBinding
```

```
      metadata:
        name: cis
        namespace: openshift-compliance
      profiles:
      - apiGroup: compliance.openshift.io/v1alpha1
        kind: Profile
        name: ocp4-cis
      - apiGroup: compliance.openshift.io/v1alpha1
        kind: Profile
        name: ocp4-cis-node
      settingsRef:
        apiGroup: compliance.openshift.io/v1alpha1
        kind: ScanSetting
        name: default
```

- **status** フィールドを確認してスキャンが完了したかどうかを確認する **ComplianceSuite** リソース (**compliance-suite-cis**):

```
apiVersion: policy.open-cluster-management.io/v1
kind: ConfigurationPolicy
metadata:
  name: compliance-suite-cis
spec:
  remediationAction: inform
  severity: high
  object-templates:
    - complianceType: musthave # this template checks if scan has completed by checking the status field
      objectDefinition:
        apiVersion: compliance.openshift.io/v1alpha1
        kind: ComplianceSuite
        metadata:
          name: cis
          namespace: openshift-compliance
        status:
          phase: DONE
```

- **ComplianceCheckResult** カスタムリソース (CR) を確認してスキャンスイートの結果を報告する **ComplianceCheckResult** リソース (**compliance-suite-cis-results**):

```
apiVersion: policy.open-cluster-management.io/v1
kind: ConfigurationPolicy
metadata:
  name: compliance-suite-cis-results
spec:
  remediationAction: inform
  severity: high
  object-templates:
    - complianceType: mustnothave # this template reports the results for scan suite: cis by looking at ComplianceCheckResult CRs
      objectDefinition:
        apiVersion: compliance.openshift.io/v1alpha1
        kind: ComplianceCheckResult
        metadata:
          namespace: openshift-compliance
```

```
    labels:
      compliance.openshift.io/check-status: FAIL
      compliance.openshift.io/suite: cis
```

policy-compliance-operator-cis-scan.yaml ファイルのサンプルを参照してください。ポリシーの作成に関する詳細は、セキュリティーポリシーの管理 を参照してください。

2.6.14. Red Hat OpenShift Platform Plus ポリシーセット

OpenShift Platform Plus ポリシーセット (**openshift-plus**) を設定して適用し、Red Hat OpenShift Platform Plus をインストールします。

OpenShift Platform Plus ポリシーセットには、デプロイされる 2 つの **PolicySets** が含まれます。OpenShift Plus ポリシーセットは、OpenShift Platform Plus 製品をインストールするために設定された複数のポリシーを適用します。Red Hat Advanced Cluster Security で保護されたクラスターサービスと Compliance Operator は、すべての OpenShift Container Platform マネージドクラスターにデプロイされます。

2.6.14.1. 前提条件

- Red Hat OpenShift Container Platform 4.13 以降を Amazon Web Services (AWS) 環境にインストールする。
- Red Hat Advanced Cluster Management for Kubernetes 2.7 以降をインストールする。
- Policy Generator KusTOMize プラグインをインストールする。詳細は、ポリシージェネレーター のドキュメントを参照してください。

2.6.14.2. OpenShift Platform Plus ポリシーセットのコンポーネント

ポリシーセットをハブクラスターに適用すると、次の OpenShift Platform Plus コンポーネントがインストールされます。

表2.12 コンポーネントテーブル

コンポーネント	ポリシー	説明
Red Hat Advanced Cluster Security	**policy-acs-central-ca-bundle**	中央サーバーを Red Hat Advanced Cluster Management for Kubernetes ハブクラスターおよびマネージドクラスターにインストールするために使用されるポリシー。
	policy-acs-central-status	Red Hat Advanced Cluster Security ステータスを受け取るためのデプロイメント。
	policy-acs-operator-central	Red Hat Advanced Cluster Security 中央 operator の設定

コンポーネント	ポリシー	説明
	policy-acs-sync-resources	Red Hat Advanced Cluster Security リソースが作成されていることを確認します。
OpenShift Container Platform	**policy-advanced-managed-cluster-status**	マネージドハブクラスター。マネージドクラスターのマネージャー。
Compliance Operator	**policy-compliance-operator-install**	Compliance operator のインストールに使用されるポリシー。
Red Hat Quay	**policy-config-quay**	Red Hat Quay の設定ポリシー。
	policy-install-quay	Red Hat Quay のインストールに使用されるポリシー。
	policy-quay-status	Red Hat Advanced Cluster Management ハブクラスターにインストールされます。
Red Hat Advanced Cluster Management	**policy-ocm-observability**	Red Hat Advanced Cluster Management 可観測性サービスをセットアップします。
Red Hat OpenShift Data Platform	**policy-odf**	Red Hat Advanced Cluster Management の可観測性と Quay によって使用される、ハブクラスターコンポーネント用の利用可能なストレージ。
	policy-odf-status	Red Hat OpenShift Data Platform のステータスを設定するために使用されるポリシー。

2.6.14.3. 関連情報

- ポリシーセットを使用した Red Hat OpenShift Platform Plus のインストール を参照してください。
- ポリシーセットコントローラー に戻ります。
- ポリシーセットに含まれるすべてのポリシーは 、**openshift-plus** ポリシーセットのサンプル を表示します。

2.7. GOVERNANCE ダッシュボードの管理

Governance ダッシュボードを使用してリソースを作成、表示、編集し、セキュリティーポリシーとポリシー違反を管理します。コマンドラインとコンソールからポリシーの YAML ファイルを作成できます。コンソールからの **ガバナンス** ダッシュボードの詳細については、引き続きお読みください。

2.7.1. ガバナンスページ

Governance ページの Overview、Policy sets、Policies には次のタブが表示されます。どの情報が表示されるかを確認するには、次の説明をお読みください。

- **概要**
  Overview タブで、Policy set violations、Policy violations、Clusters、Categories、Controls、および Standards タブから概要カードが表示されます。

- **ポリシーセット**
  ハブクラスターポリシーセットを作成および管理します。

- **ポリシー**

  - セキュリティーポリシーを作成および管理します。ポリシーの表には、ポリシーの次の詳細がリストされます。Name、Namespace、および Cluster violations が表示されます。

  - Actions アイコンを選択すると、修復を編集、有効化、無効化の設定をして、ポリシーの通知、有効化、または削除ができます。特定のポリシーのカテゴリーおよび標準を表示するには、ドロップダウン矢印を選択して行を展開します。

  - Manage column ダイアログボックスでテーブルの列の順序を変更します。Manage column アイコンを選択すると、ダイアログボックスが表示されます。列の順序を変更するには、Reorder アイコンを選択して列名を移動します。表に列を表示するには、その列名のチェックボックスをクリックし、Save ボタンを選択します。

  - 複数のポリシーを選択して Actions ボタンをクリックして、完全な一括処理を行います。Filter ボタンをクリックしてポリシーテーブルをカスタマイズすることもできます。表一覧でポリシーを選択すると、コンソールで、以下の情報タブが表示されます。

    - Details: Details タブを選択して、ポリシーの情報、配置の情報を表示します。Placement の表の **コンプライアンス** 列には、表示されるクラスターのコンプライアンスを確認するためのリンクがあります。

    - Results: Results タブを選択して、ポリシーに関連付けられた全クラスターの表リストを表示します。

- Message 列から View details リンクをクリックして、テンプレートの詳細、テンプレート YAML、および関連リソースを表示します。関連リソースを表示することもできます。View history リンクをクリックして、違反メッセージと最後のレポートの時間を表示します。

2.7.2. ガバナンスの自動化設定

特定のポリシーに設定済みの自動化がある場合は、自動化を選択して詳細を表示できます。自動化のスケジュール頻度オプションに関する以下の説明を参照してください。

- Manual Run: この自動化を手動で設定して 1 回実行します。自動化の実行後に、**disabled** に設定されます。**注記:** スケジュール頻度が無効になっている場合のみ Manual run モードを選択できます。

- Run once mode: ポリシーに違反すると、自動化が 1 回実行されます。自動化の実行後に、**disabled** に設定されます。自動化が **disabled** に設定された後は、引き続き自動化を手動で実行する必要があります。once mode を実行すると、**target_clusters** の追加変数にはポリシーに違反するクラスターのリストが自動的に指定されます。Ansible Automation Platform Job テンプレートでは、**EXTRA VARIABLES** セクション (**extra_vars** とも呼ばれる) に対して **PROMPT ON LAUNCH** が有効になっている必要があります。

- **Run everyEvent モード**: ポリシーに違反すると、自動化はマネージドクラスターごとに固有のポリシー違反が発生するたびに毎回実行されます。**DelayAfterRunSeconds** パラメーターを使用して、同じクラスターで自動化を再開できるようになるまでの最小秒数を設定します。ポリシーが遅延期間中に複数回違反され、違反状態のままである場合、自動化は遅延期間後に 1 回実行されます。デフォルトは 0 秒で、**everyEvent** モードにのみ適用されます。**everyEvent** モードを実行すると、**target_clusters** と Ansible Automation Platform Job テンプレートの追加変数は **once モード** と同じになります。

- **Disable automation**: スケジュールされた自動化が **disabled** に設定されると、設定が更新されるまで自動化は実行されません。

以下の変数は、Ansible Automation Platform ジョブの **extra_vars** で自動的に提供されます。

- **policy_name**: ハブクラスターで Ansible Automation Platform ジョブを開始する非準拠のルートポリシーの名前。

- **policy_namespace**: ルートポリシーの namespace。

- **hub_cluster**: **clusters DNS** オブジェクトの値によって決定されるハブクラスターの名前。

- **policy_sets**: このパラメーターには、ルートポリシーに関連付けられたすべてのポリシーセット名が含まれます。ポリシーがポリシーセット内にない場合、**policy_set** パラメーターは空です。

- **policy_violations**: このパラメーターには、非準拠のクラスター名のリストが含まれており、値は非準拠の各クラスターのポリシー **status** フィールドです。

2.7.3. 関連情報

セキュリティーポリシーの作成および更新の詳細は、以下のトピックを参照してください。

- セキュリティーポリシーの管理
- 設定ポリシーの管理
- gatekeeper ポリシーの管理
- ガバナンスのための Ansible Automation Platform の設定
- ガバナンス

2.7.4. ガバナンスのための Ansible Automation Platform の設定

Red Hat Advanced Cluster Management for Kubernetes ガバナンスを Red Hat Ansible Automation Platform と統合して、ポリシー違反の自動化を作成できます。Red Hat Advanced Cluster Management コンソールで、自動化を設定できます。

- 前提条件
- コンソールからのポリシー違反自動化の作成
- CLI からのポリシー違反自動化の作成

2.7.4.1. 前提条件

- Red Hat OpenShift Container Platform 4.13 以降

- Ansible Automation Platform バージョン 3.7.3 以降のバージョンがインストールされている。サポートされている最新バージョンの Ansible Automation Platform をインストールすることを推奨します。詳細については、Red Hat Ansible Automation Platform ドキュメント を参照してください。

- Operator Lifecycle Manager から Ansible Automation Platform Resource Operator をインストールしておく。**Update Channel** セクションで、**stable-2.x-cluster-scoped** を選択します。**All namespaces on the cluster (default)**インストールモードを選択します。
注記: Ansible Automation Platform ジョブテンプレートを実行する際は、べき等であることを確認してください。Ansible Automation Platform Resource Operator がない場合は、Red Hat OpenShift Container Platform **OperatorHub** ページから確認することができる。

Red Hat Ansible Automation Platform のインストールと設定の詳細については、Ansible タスクの設定 を参照してください。

2.7.4.2. コンソールからのポリシー違反自動化の作成

Red Hat Advanced Cluster Management ハブクラスターにログインし、ナビゲーションメニューから **Governance** を選択し、**Policies** タブをクリックしてポリシーテーブルを表示します。

Automation 列の **Configure** をクリックして、特定のポリシーの自動化を設定します。ポリシー自動化パネルが表示されたら、自動化を作成できます。**Ansible credential** セクションから、ドロップダウンメニューをクリックして Ansible 認証情報を選択します。認証情報を追加する必要がある場合は、認証情報の管理 を参照してください。

注記: この認証情報は、ポリシーと同じ namespace にコピーされます。自動化の開始用に作成された **AnsibleJob** リソースで、この認証情報を使用します。コンソールの **Credentials** セクションで Ansible 認証情報に加えられた変更は、自動的に更新されます。

認証情報を選択したら、Ansible ジョブドロップダウンリストをクリックしてジョブテンプレートを選択します。**Extra variables** セクションで、**PolicyAutomation** の **extra_vars** セクションからパラメーター値を追加します。自動化の頻度を選択します。**Run once mode**、**Run everyEvent mode**、または **Disable Automation** を選択できます。

Submit を選択して、ポリシー違反の自動化を保存します。Ansible ジョブの詳細パネルから **View Job** リンクを選択すると、このリンクから **Search** ページのジョブテンプレートが表示されます。自動化が正常に作成されると、**Automation** 列に表示されます。

注意: ポリシー自動化が関連付けられているポリシーを削除すると、ポリシー自動化はクリーンアップの一部として自動的に削除されます。

コンソールからポリシー違反の自動化が作成されました。

2.7.4.3. CLI からのポリシー違反自動化の作成

CLI からポリシー違反の自動化を設定するには、以下の手順を実行します。

1. ターミナルから、**oc login** コマンドを使用して Red Hat Advanced Cluster Management ハブクラスターに再度ログインします。

2. 自動化を追加するポリシーを検索するか、作成します。ポリシー名と namespace をメモします。

3. 以下のサンプルをガイドとして使用して、**Policy Automation** リソースを作成します。

```
apiVersion: policy.open-cluster-management.io/v1beta1
```

```
kind: PolicyAutomation
metadata:
  name: policyname-policy-automation
spec:
  automationDef:
    extra_vars:
      your_var: your_value
    name: Policy Compliance Template
    secret: ansible-tower
    type: AnsibleJob
  mode: disabled
  policyRef: policyname
```

4. 前のサンプルの Automation テンプレート名は **Policy Compliance Template** です。この値は、ジョブテンプレート名と一致するように変更してください。

5. **extra_vars** セクションで、Automation テンプレートに渡す必要があるパラメーターを追加します。

6. モードを **once**、**everyEvent**、または **disabled** に設定します。

7. **policyRef** は、ポリシーの名前に設定します。

8. Ansible Automation Platform 認証情報を含むこの **PolicyAutomation** リソースと同じ namespace にシークレットを作成します。上記の例では、シークレット名は **ansible-tower** です。アプリケーションライフサイクルからののサンプル を使用して、シークレットの作成方法を確認します。

9. **PolicyAutomation** リソースを作成します。
   注記:

   - 以下のアノテーションを **Policy Automation** リソースに追加することで、ポリシー自動化の即時実行を開始できます。

     ```
     metadata:
       annotations:
         policy.open-cluster-management.io/rerun: "true"
     ```

   - ポリシーが **once** モードの場合は、ポリシーがコンプライアンス違反があると自動化が実行されます。**target_clusters** という名前の **extra_vars** 変数が追加され、値はコンプライアンス違反のポリシーが含まれる、各マネージドクラスター名の配列です。

   - ポリシーが **everyEvent** モードであり、**DelayAfterRunSeconds** が定義された時間値を超えると、ポリシーは非準拠となり、ポリシー違反ごとに自動化が実行されます。

2.8. テンプレート処理の概要

設定ポリシーは、オブジェクト定義での Golang テキストテンプレートの追加をサポートします。これらのテンプレートは、そのクラスターに関連する設定を使用して、ハブクラスターまたはターゲットのマネージドクラスターでランタイム時に解決されます。これにより、動的コンテンツで設定ポリシーを定義でき、ターゲットクラスターに、カスタマイズされた Kubernetes リソースを通知したり、強制的に実行したりできます。

設定ポリシー定義には、ハブクラスターとマネージドクラスターのテンプレートの両方を含めることができます。ハブクラスターテンプレートは、先にハブクラスターで処理され、解決されたハブクラス

ターテンプレートを使用したポリシー定義がターゲットクラスターに伝播されます。マネージドクラスターでは、**ConfigurationPolicyController** はポリシー定義内のマネージドクラスターテンプレートを処理し、その後、完全に解決されたオブジェクト定義を有効にするか、検証します。

テンプレートは Golang テンプレート言語仕様に準拠し、解決されたテンプレートから生成されるリソース定義は有効な YAML である必要がある。詳細は、Golang ドキュメントの **Package templates** を参照。テンプレート検証のエラーは、ポリシー違反として認識される。カスタムのテンプレート関数を使用する場合、値はランタイム時に置き換えられる。

重要:

- ハブクラスターテンプレートを使用してシークレットや他の機密データを伝播する場合には、機密データはハブクラスターにあるマネージドクラスターの namespace か、そのポリシーが配布されているマネージドクラスター上に存在します。テンプレートの内容はポリシーで拡張され、 OpenShift Container Platform ETCD 暗号化サポートでは、ポリシーは暗号化されません。これに対処するには、**fromSecret** または **copySecretData を** 使用して、シークレットの値を自動的に暗号化するか、他の値を暗号化するための **protect** を使用します。

- 証明書などの複数行の文字列値を追加する場合は、改行を処理するために、テンプレートパイプラインの最後に常に **| toRawJson | toLiteral** 構文を追加します。たとえば、**Secret** リソースから証明書をコピーして **ConfigMap** リソースに含める場合、テンプレートパイプラインは次の構文のようになります。

  ```
  ca.crt: '{{ fromSecret "openshift-config" "ca-config-map-secret" "ca.crt"  | base64dec | toRawJson | toLiteral }}'
  ```

  toRawJson テンプレート関数は、YAML 構造に影響を与えないように改行をエスケープした入力値を JSON 文字列に変換します。**toLiteral** テンプレート関数は、出力から外側の単一引用符を削除します。たとえば、テンプレートが **key: '{{ 'hello\nworld' | toRawJson }}'** テンプレートパイプラインに対して処理される場合、出力は **key: '"hello\nworld"'** になります。**key: '{{ 'hello\nworld' | toRawJson | toLiteral }}'** テンプレートパイプラインの出力は、**key: "hello\nworld"** です。

ハブクラスターとマネージドクラスターのテンプレートの比較は、以下の表を参照してください。

2.8.1. ハブクラスターとマネージドクラスターテンプレートの比較

表2.13 比較表

テンプレート	ハブクラスター	マネージドクラスター
構文	Golang テキストテンプレートの仕様	Golang テキストテンプレートの仕様
デリミタ	{{hub ... hub}}	{{ ... }}

テンプレート	ハブクラスター	マネージドクラスター
コンテキスト	**.ManagedClusterName** 変数を使用できます。これはランタイム時に、ポリシーが伝播されるターゲットクラスターの名前に解決されます。**.ManagedClusterLabels** 変数も使用できます。これは、ポリシーが伝播されるマネージドクラスター上のラベルのキーと値のマップに解決されます。	コンテキスト変数はありません
アクセス制御	**Policy** リソースと同じ namespace に存在する namespace を使用した Kubernetes オブジェクトのみを参照できます。	クラスターの任意のリソースを参照できます。
関数	Kubernetes リソースおよび文字列操作への動的なアクセスをサポートするテンプレート関数のセット。詳細は、**Template functions** を参照してください。検索制限については、アクセス制御の行を参照してください。 ハブクラスターの **fromSecret** テンプレート機能は、結果の値をマネージドクラスターの namespace に複製されたポリシーで暗号化された文字列として保存します。 同等の呼び出しは、次の構文を使用する場合があります: **{{hub "(lookup "v1" "Secret" "default" "my-hub-secret").data.message | protect hub}}**	テンプレート関数セットは、Kubernetes リソースおよび文字列操作への動的なアクセスをサポートします。詳細は、**Template functions** を参照してください。

テンプレート	ハブクラスター	マネージドクラスター
関数出力ストレージ	テンプレート関数の出力は、マネージドクラスターに同期される前に、マネージドクラスターで適用可能な各マネージドクラスター namespace の **Policy** resource オブジェクトに保存されます。つまり、テンプレート関数からの結果は機密な内容であっても、ハブクラスター上の **Policy** リソースオブジェクトや、マネージドクラスター上の **ConfigurationPolicy** リソースオブジェクトへの読み取り権限がある全ユーザーによる読み取りが可能です。さらに、etcd 暗号化が有効になっている場合、**Policy** および **ConfigurationPolicy** リソースオブジェクトは暗号化されません。機密な情報の出力を返すテンプレート関数 (シークレットなど) を使用する場合には、この点を慎重に検討することが推奨されます。	テンプレート関数の出力は、ポリシー関連のリソースオブジェクトには保存されません。
処理	複製されたポリシーのクラスターへの伝播中に、ハブクラスターのランタイムで処理が発生します。ポリシーと、そのポリシー内にあるハブクラスターのテンプレートは、テンプレートの作成時または更新時にのみハブクラスターで処理されます。	処理は、マネージドクラスターの **ConfigurationPolicyController** で実行されます。ポリシーは定期的に処理され、参照されるリソースのデータを使用して解決されたオブジェクト定義を自動的に更新します。
エラーの処理	ハブクラスターテンプレートからのエラーは、ポリシーの適用先のマネージドクラスターの違反として表示されます。	マネージドクラスターテンプレートからのエラーは、違反が発生した特定のターゲットクラスターの違反として表示されます。

次のトピックを引き続きお読みください。

- テンプレート関数
- 設定ポリシーでの高度なテンプレート処理

2.8.2. テンプレート関数

リソース固有および汎用の **lookup** テンプレート関数など、テンプレート関数は、ハブクラスター (**{{hub ... hub}}** 区切り文字の使用) またはマネージドクラスター (**{{ ... }}** 区切り文字の使用) 上の Kubernetes リソースを参照するために用意されています。詳細は、**Template processing** を参照してください。リソース固有の関数は利便性があり、リソースの内容の使いやすさを高めます。より高度な

汎用関数 **lookup** を使用する場合は、検索されるリソースの YAML 構造をよく理解してください。これらの関数に加えて、**base64enc**、**base64dec**、**indent**、**autoindent**、**toInt**、**toBool** などのユーティリティー関数も使用できます。

YAML 構文でテンプレートに準拠するには、テンプレートは引用符またはブロック文字 (| または >) を使用して文字列としてポリシーリソースで設定する必要があります。これにより、解決済みのテンプレート値も文字列になります。これをオーバーライドするには、テンプレートの最後の関数として **toInt** または **toBool** を使用して、値をそれぞれ整数またはブール値として強制的に解釈するさらなる処理を開始します。サポート対象のカスタムテンプレート関数の説明と例について確認するには、以下を参照してください。

- fromSecret 関数
- fromConfigMap 関数
- fromClusterClaim 関数
- lookup 関数
- base64enc 関数
- base64dec 関数
- indent 関数
- autoindent 関数
- toInt 関数
- toBool 関数
- protect 関数
- toLiteral 関数
- copySecretData 関数
- copyConfigMapData 関数
- サポートされている Sprig オープンソース関数

2.8.2.1. fromSecret 関数

fromSecret 関数は、シークレット内にある指定のデータキーの値を返します。関数については、以下の構文を確認してください。

```
func fromSecret (ns string, secretName string, datakey string) (dataValue string, err error)
```

この関数を使用するには、Kubernetes **Secret** リソースの namespace、名前、およびデータキーを入力します。ハブクラスターテンプレートの関数を使用する場合は、ポリシーに使用されるのと同じ namespace を使用する必要があります。詳細は、**Template processing** を参照してください。

注意: この関数をハブクラスターテンプレートで使用すると、保護関数 を使用して出力が自動的に暗号化されます。

Kubernetes **Secret** がターゲットクラスターに存在しない場合は、ポリシー違反が表示されます。データキーがターゲットクラスターに存在しない場合は、値が空の文字列になります。以下で、ターゲット

クラスターで **Secret** リソースを有効にする設定ポリシーを確認します。**PASSWORD** データキーの値は、ターゲットクラスターのシークレットを参照するテンプレートを指します。

```
apiVersion: policy.open-cluster-management.io/v1
kind: ConfigurationPolicy
metadata:
  name: demo-fromsecret
  namespace: test
spec:
  namespaceSelector:
    exclude:
    - kube-*
    include:
    - default
  object-templates:
  - complianceType: musthave
    objectDefinition:
      apiVersion: v1
      data:
        USER_NAME: YWRtaW4=
        PASSWORD: '{{ fromSecret "default" "localsecret" "PASSWORD" }}'
      kind: Secret
      metadata:
        name: demosecret
        namespace: test
      type: Opaque
  remediationAction: enforce
  severity: low
```

重要: 証明書などの複数行の文字列値を追加する場合は、改行を処理するために、テンプレートパイプラインの最後に常に **| toRawJson | toLiteral** 構文を追加します。たとえば、**Secret** リソースから証明書をコピーして **ConfigMap** リソースに含める場合、テンプレートパイプラインは次の構文のようになります。

```
ca.crt: '{{ fromSecret "openshift-config" "ca-config-map-secret" "ca.crt"  | base64dec | toRawJson |
toLiteral }}'
```

toRawJson テンプレート関数は、YAML 構造に影響を与えないように改行をエスケープした入力値を JSON 文字列に変換します。**toLiteral** テンプレート関数は、出力から外側の単一引用符を削除します。たとえば、テンプレートが **key: '{{ 'hello\nworld' | toRawJson }}'** テンプレートパイプラインに対して処理される場合、出力は **key: "'hello\nworld"'** になります。**key: '{{ 'hello\nworld' | toRawJson | toLiteral }}'** テンプレートパイプラインの出力は、**key: "hello\nworld"** です。

2.8.2.2. fromConfigmap 関数

fromConfigMap 関数は、ConfigMap 内にある指定のデータキーの値を返します。関数については、以下の構文を確認してください。

```
func fromConfigMap (ns string, configmapName string, datakey string) (dataValue string, err Error)
```

この関数を使用するには、Kubernetes **ConfigMap** リソースの namespace、名前、およびデータキーを入力します。ハブクラスターテンプレートの関数を使用するポリシーに使用されるのと同じ namespace を使用する必要があります。詳細は、**Template processing** を参照してください。Kubernetes **ConfigMap** リソースまたはデータキーがターゲットクラスターに存在しない場合は、ポリシー違反が表示されます。データキーがターゲットクラスターに存在しない場合は、値が空の文字列に

なります。以下で、ターゲットのマネージドクラスターで Kubernetes リソースを有効にする設定ポリシーを表示します。**log-file** データキーの値は、ConfigMap から **log-file**、**default** namespace から **log-config** の値を取得するテンプレートであり、**log-level** はデータキーの **log-level** に設定されます。

```
apiVersion: policy.open-cluster-management.io/v1
kind: ConfigurationPolicy
metadata:
  name: demo-fromcm-lookup
  namespace: test-templates
spec:
  namespaceSelector:
    exclude:
    - kube-*
    include:
    - default
  object-templates:
  - complianceType: musthave
    objectDefinition:
      kind: ConfigMap
      apiVersion: v1
      metadata:
        name: demo-app-config
        namespace: test
      data:
        app-name: sampleApp
        app-description: "this is a sample app"
        log-file: '{{ fromConfigMap "default" "logs-config" "log-file" }}'
        log-level: '{{ fromConfigMap "default" "logs-config" "log-level" }}'
  remediationAction: enforce
  severity: low
```

2.8.2.3. fromClusterClaim 関数

fromClusterClaim 関数は、**ClusterClaim** リソースの **Spec.Value** の値を返します。関数については、以下の構文を確認してください。

```
func fromClusterClaim (clusterclaimName string) (dataValue string, err Error)
```

この関数を使用する場合は、Kubernetes **ClusterClaim** リソースの名前を入力します。**ClusterClaim** リソースが存在しない場合は、ポリシー違反が表示されます。以下で、ターゲットのマネージドクラスターで Kubernetes リソースを有効にする設定ポリシーの例を確認してください。**platform** データキーの値は、**platform.open-cluster-management.io** クラスター要求の値を取得するテンプレートです。同様に、 **product** と **version** の値は **ClusterClaim** から取得します。

```
apiVersion: policy.open-cluster-management.io/v1
kind: ConfigurationPolicy
metadata:
  name: demo-clusterclaims
  namespace: default
spec:
  namespaceSelector:
    exclude:
    - kube-*
    include:
```

```
    - default
  object-templates:
  - complianceType: musthave
    objectDefinition:
      kind: ConfigMap
      apiVersion: v1
      metadata:
        name: sample-app-config
        namespace: default
      data:
        # Configuration values can be set as key-value properties
        platform: '{{ fromClusterClaim "platform.open-cluster-management.io" }}'
        product: '{{ fromClusterClaim "product.open-cluster-management.io" }}'
        version: '{{ fromClusterClaim "version.openshift.io" }}'
  remediationAction: enforce
  severity: low
```

2.8.2.4. lookup 関数

lookup 関数は、JSON と互換性のあるマップとして Kubernetes リソースを返します。要求されたリソースが存在しない場合は、空のマップが返されます。リソースが存在せず、値が別のテンプレート関数に提供されている場合は、エラー **invalid value; expected string** が発生する可能性があります。

注記: default テンプレート関数を使用して、後のテンプレート関数に正しい型が提供されるようにします。**サポートされている Spig オープンソース関数** セクションを参照してください。

関数については、以下の構文を確認してください。

```
func lookup (apiversion string, kind string, namespace string, name string, labelselector ...string)
(value string, err Error)
```

この関数を使用する場合は、Kubernetes リソースの API バージョン、種類、namespace、名前、およびオプションのラベルセレクターを入力します。ハブクラスターテンプレート内のポリシーに使用されるものと同じ namespace を使用する必要があります。詳細は、**Template processing** を参照してください。ラベルセレクターの例については、**Additional resources** セクションにある **Kubernetes labels and selectors** ドキュメントへのリファレンスを参照してください。以下で、ターゲットのマネージドクラスターで Kubernetes リソースを有効にする設定ポリシーの例を確認してください。**metrics-url** データキーの値は、**default** namespace から **v1**/**Service** Kubernetes リソースの **metrics** を取得するテンプレートであり、クエリーされたリソースにある **Spec.ClusterIP** の値に設定されます。

```
apiVersion: policy.open-cluster-management.io/v1
kind: ConfigurationPolicy
metadata:
  name: demo-lookup
  namespace: test-templates
spec:
  namespaceSelector:
    exclude:
    - kube-*
    include:
    - default
  object-templates:
  - complianceType: musthave
    objectDefinition:
      kind: ConfigMap
```

```
      apiVersion: v1
      metadata:
        name: demo-app-config
        namespace: test
      data:
        # Configuration values can be set as key-value properties
        app-name: sampleApp
        app-description: "this is a sample app"
        metrics-url: |
          http://{{ (lookup "v1" "Service" "default" "metrics").spec.clusterIP }}:8080
  remediationAction: enforce
  severity: low
```

2.8.2.5. base64enc 関数

base64enc 関数は、入力 **データ文字列** を **base64** でエンコードされた値で返します。関数については、以下の構文を確認してください。

```
func base64enc (data string) (enc-data string)
```

この関数を使用する場合は、文字列値を入力します。以下で、**base64enc** 関数を使用する設定ポリシーの例を確認してください。

```
apiVersion: policy.open-cluster-management.io/v1
kind: ConfigurationPolicy
metadata:
  name: demo-fromsecret
  namespace: test
spec:
  namespaceSelector:
    exclude:
    - kube-*
    include:
    - default
  object-templates:
  - complianceType: musthave
    objectDefinition:
    ...
    data:
      USER_NAME: '{{ fromConfigMap "default" "myconfigmap" "admin-user" | base64enc }}'
```

2.8.2.6. base64dec 関数

base64dec 関数は、入力 **enc-data 文字列** を **base64** デコードされた値で返します。関数については、以下の構文を確認してください。

```
func base64dec (enc-data string) (data string)
```

この関数を使用する場合は、文字列値を入力します。以下で、**base64dec** 関数を使用する設定ポリシーの例を確認してください。

```
apiVersion: policy.open-cluster-management.io/v1
kind: ConfigurationPolicy
```

```
metadata:
  name: demo-fromsecret
  namespace: test
spec:
  namespaceSelector:
    exclude:
    - kube-*
    include:
    - default
  object-templates:
  - complianceType: musthave
    objectDefinition:
    ...
    data:
      app-name: |
        "{{ ( lookup "v1"  "Secret" "testns" "mytestsecret") .data.appname ) | base64dec }}"
```

2.8.2.7. indent 関数

indent 関数により、パディングされた **データ文字列** が返されます。関数については、以下の構文を確認してください。

```
func indent (spaces  int,  data string) (padded-data string)
```

この関数を使用する場合は、特定のスペース数でデータ文字列を入力します。以下で、**indent** 関数を使用する設定ポリシーの例を確認してください。

```
apiVersion: policy.open-cluster-management.io/v1
kind: ConfigurationPolicy
metadata:
  name: demo-fromsecret
  namespace: test
spec:
  namespaceSelector:
    exclude:
    - kube-*
    include:
    - default
  object-templates:
  - complianceType: musthave
    objectDefinition:
    ...
    data:
      Ca-cert:  |
        {{ ( index ( lookup "v1" "Secret" "default" "mycert-tls"  ).data  "ca.pem"  ) |  base64dec | indent 4
}}
```

2.8.2.8. autoindent 関数

autoindent 関数は、**indent** 関数のように機能し、テンプレートの前のスペース数に基づいて自動的に先頭のスペース数を決定します。以下で、**autoindent** 関数を使用する設定ポリシーの例を確認してください。

```
apiVersion: policy.open-cluster-management.io/v1
```

```
kind: ConfigurationPolicy
metadata:
  name: demo-fromsecret
  namespace: test
spec:
  namespaceSelector:
    exclude:
    - kube-*
    include:
    - default
  object-templates:
  - complianceType: musthave
    objectDefinition:
    ...
    data:
      Ca-cert:  |
        {{ ( index ( lookup "v1" "Secret" "default" "mycert-tls"  ).data  "ca.pem"  ) |  base64dec |
autoindent }}
```

2.8.2.9. toInt 関数

toInt 関数は入力値の整数値をキャストして返します。また、テンプレートの最後の関数である場合は、ソースコンテンツがさらに処理されます。これは、YAML で値が整数として解釈されるようにするためです。関数については、以下の構文を確認してください。

```
func toInt (input interface{}) (output int)
```

この関数を使用する場合は、整数としてキャストする必要があるデータを入力します。以下で、**toInt** 関数を使用する設定ポリシーの例を確認してください。

```
apiVersion: policy.open-cluster-management.io/v1
kind: ConfigurationPolicy
metadata:
  name: demo-template-function
  namespace: test
spec:
  namespaceSelector:
    exclude:
    - kube-*
    include:
    - default
  object-templates:
  - complianceType: musthave
    objectDefinition:
    ...
    spec:
      vlanid:  |
        {{ (fromConfigMap "site-config" "site1" "vlan")  | toInt }}
```

2.8.2.10. toBool 関数

toBool 関数は、入力文字列をブール値に変換し、ブール値を返します。また、テンプレートの最後の関数である場合は、ソースコンテンツがさらに処理されます。これは、YAML で値がブール値として解釈されるようにするためです。関数については、以下の構文を確認してください。

```
func toBool (input string) (output bool)
```

この関数を使用する場合は、ブール値に変換する必要のある文字列データを入力します。以下で、**toBool** 関数を使用する設定ポリシーの例を確認してください。

```
apiVersion: policy.open-cluster-management.io/v1
kind: ConfigurationPolicy
metadata:
  name: demo-template-function
  namespace: test
spec:
  namespaceSelector:
    exclude:
    - kube-*
    include:
    - default
  object-templates:
  - complianceType: musthave
    objectDefinition:
    ...
    spec:
      enabled: |
        {{ (fromConfigMap "site-config" "site1" "enabled")  | toBool }}
```

2.8.2.11. protect 関数

protect 機能により、ハブクラスターポリシーテンプレートで文字列を暗号化できます。これは、ポリシーの評価時にマネージドクラスターで自動的に復号化されます。以下で、**protect** 関数を使用する設定ポリシーの例を確認してください。

```
apiVersion: policy.open-cluster-management.io/v1
kind: ConfigurationPolicy
metadata:
  name: demo-template-function
  namespace: test
spec:
  namespaceSelector:
    exclude:
    - kube-*
    include:
    - default
  object-templates:
  - complianceType: musthave
    objectDefinition:
    ...
    spec:
      enabled: |
        {{hub (lookup "v1" "Secret" "default" "my-hub-secret").data.message | protect hub}}
```

前述の YAML の例では、**lookup** 関数を使用するために定義した既存のハブクラスターポリシーテンプレートがあります。マネージドクラスターの namespace に複製されたポリシーでは、値は **$ocm_encrypted:okrrBqt72ol+3WT/0vxel3vGa+wpLD7Z0ZxFMLvL204=** のようになります。

それぞれの暗号化アルゴリズムは、256 ビットキーを使用した AES-CBC です。各暗号化キーはマネージドクラスターごとに一意で、30 日ごとに自動的にローテーションされます。

これにより、復号化された値がマネージドクラスターのポリシーに保存されることはありません。

即時のローテーションを強制するには、ハブクラスターのマネージドクラスター namespace の **policy-encryption-key** Secret で **policy.open-cluster-management.io/last-rotated** アノテーションを削除します。その後、ポリシーが再処理され、新しい暗号化キーが使用されます。

2.8.2.12. toLiteral 関数

toLiteral 関数は、処理後にテンプレート文字列を囲む引用符をすべて削除します。この関数を使用して、JSON 文字列を ConfigMap フィールドからマニフェストの JSON 値に変換できます。次の関数を実行して、**key** パラメーター値から引用符を削除します。

```
key: '{{ "[\"10.10.10.10\", \"1.1.1.1\"]" | toLiteral }}'
```

toLiteral 関数を使用すると、次の更新が表示されます。

```
key: ["10.10.10.10", "1.1.1.1"]
```

2.8.2.13. copySecretData 関数

copySecretData 関数は、指定されたシークレットのすべての **data** コンテンツをコピーします。次の関数のサンプルをご覧ください。

```
complianceType: musthave
    objectDefinition:
      apiVersion: v1
      kind: Secret
      metadata:
        name: my-secret-copy
      data: '{{ copySecretData "default" "my-secret" }}'
```

注意: この関数をハブクラスターテンプレートで使用すると、保護関数 を使用して出力が自動的に暗号化されます。

2.8.2.14. copyConfigMapData 関数

copyConfigMapData 関数は、指定された ConfigMap のすべての **data** コンテンツをコピーします。次の関数のサンプルをご覧ください。

```
complianceType: musthave
    objectDefinition:
      apiVersion: v1
      kind: ConfigMap
      metadata:
        name: my-secret-copy
      data: '{{ copyConfigMapData "default" "my-configmap" }}'
```

2.8.2.15. サポートされている Sprig オープンソース関数

さらに、Red Hat Advanced Cluster Management は、**sprig** オープンソースプロジェクトに含まれる以下のテンプレート関数をサポートします。

表2.14 サポートされているコミュニティーの Sprig 関数の表

Sprig ライブラリー	関数
Cryptographic and security	**htpasswd**
Date	**date**, **mustToDate**, **now**, **toDate**
Default	**default**, **empty**, **fromJson**, **mustFromJson**, **ternary**, **toJson**, **toRawJson**
Dictionaries and dict	**dig**
Integer math	**add**, **mul**, **div**, **round**, **sub**
Integer slice	**until**, **untilStep**,
Lists	**append**, **concat**, **has**, **list**, **mustAppend**, **mustHas**, **mustPrepend**, **mustSlice**, **prepend**, **slice**
String functions	**cat**, **contains**, **hasPrefix**, **hasSuffix**, **join**, **lower**, **quote**, **replace**, **split**, **splitn**, **substr**, **trim**, **trimAll**, **trunc**, **upper**
Version comparison	**semver**, **semverCompare**

2.8.2.16. 関連情報

- 詳細は、Template processing を参照してください。
- 使用例は 設定ポリシーでの高度なテンプレート処理 を参照してください。
- ラベルセレクターの例は、Kubernetes のラベルとセレクター のドキュメントを参照してください。
- Golang ドキュメント - パッケージテンプレート を参照してください。
- 詳細は、Sprig 関数のドキュメント を参照してください。

2.8.3. 設定ポリシーでの高度なテンプレート処理

マネージドクラスターとハブクラスターの両方のテンプレートを使用すると、ターゲットクラスターごとに個別のポリシーを作成したり、ポリシー定義で設定値をハードコードしたりする必要性が軽減されます。セキュリティーを確保するには、ハブクラスターテンプレートのリソース固有および一般的なルックアップ機能の両方が、ハブクラスターのポリシーの namespace に制限されます。

重要: ハブクラスターテンプレートを使用して機密データやその他の機密データを伝播すると、ハブクラスター上のマネージドクラスターの namespace と、そのポリシーが配布されているマネージドクラ

スターで機密データが漏洩する原因になります。テンプレートの内容はポリシーで拡張され、OpenShift Container Platform ETCD 暗号化サポートでは、ポリシーは暗号化されません。これに対処するには、**fromSecret** または **copySecretData** を 使用して、シークレットの値を自動的に暗号化するか、他の値を暗号化するための **protect** を使用します。

高度なテンプレートの使用例については、引き続きお読みください。

- 再処理のための特別なアノテーション
- オブジェクトテンプレートの処理
- テンプレート処理のバイパス

2.8.3.1. 再処理のための特別なアノテーション

ハブクラスターテンプレートは、ポリシーの作成中、または参照されたリソースが更新されたときに、参照されたリソースのデータに解決されます。

更新を手動で開始する必要がある場合は、特別なアノテーション **policy.open-cluster-management.io/trigger-update** を使用して、テンプレートによって参照されるデータの変更を示します。特別なアノテーション値を変更すると、テンプレート処理が自動的に開始されます。さらに、参照されたリソースの最新のコンテンツが読み取られ、ポリシー定義で更新されます。ポリシー定義は、マネージドクラスターでの処理のために伝達されます。このアノテーションを使用する方法の 1 つは、値を毎回 1 ずつインクリメントすることです。

2.8.3.2. オブジェクトテンプレートの処理

YAML 文字列表現を使用してオブジェクトテンプレートを設定します。**object-template-raw** パラメーターは、if-else や **range** 関数などの高度なテンプレートのユースケースをサポートするオプションのパラメーターです。次の例は、**Sea Otter** と等しい **name** キーを持つ **default**の namespace 内の任意の ConfigMap に **species-category: mammal** ラベルを追加するように定義されています。

```
object-templates-raw: |
  {{- range (lookup "v1" "ConfigMap" "default" "").items }}
  {{- if eq .data.name "Sea Otter" }}
  - complianceType: musthave
    objectDefinition:
      kind: ConfigMap
      apiVersion: v1
      metadata:
        name: {{ .metadata.name }}
        namespace: {{ .metadata.namespace }}
        labels:
          species-category: mammal
  {{- end }}
  {{- end }}
```

注記: spec.object-templates と **spec.object-templates-raw** はオプションですが、2 つのパラメーターフィールドのいずれかを設定する必要があります。

高度なテンプレートを使用して、マネージドクラスターのインフラストラクチャー **MachineSet** オブジェクトを作成および設定する次のポリシーの例を参照してください。

```
apiVersion: policy.open-cluster-management.io/v1
kind: ConfigurationPolicy
```

```
metadata:
  name: create-infra-machineset
spec:
  remediationAction: enforce
  severity: low
  object-templates-raw: |
    {{- /* Specify the parameters needed to create the MachineSet  */ -}}
    {{- $machineset_role := "infra" }}
    {{- $region := "ap-southeast-1" }}
    {{- $zones := list "ap-southeast-1a" "ap-southeast-1b" "ap-southeast-1c" }}
    {{- $infrastructure_id := (lookup "config.openshift.io/v1" "Infrastructure" ""
"cluster").status.infrastructureName }}
    {{- $worker_ms := (index (lookup "machine.openshift.io/v1beta1" "MachineSet" "openshift-
machine-api" "").items 0) }}
    {{- /* Generate the MachineSet for each zone as specified  */ -}}
    {{- range $zone := $zones }}
    - complianceType: musthave
      objectDefinition:
        apiVersion: machine.openshift.io/v1beta1
        kind: MachineSet
        metadata:
          labels:
            machine.openshift.io/cluster-api-cluster: {{ $infrastructure_id }}
          name: {{ $infrastructure_id }}-{{ $machineset_role }}-{{ $zone }}
          namespace: openshift-machine-api
        spec:
          replicas: 1
          selector:
            matchLabels:
              machine.openshift.io/cluster-api-cluster: {{ $infrastructure_id }}
              machine.openshift.io/cluster-api-machineset: {{ $infrastructure_id }}-{{ $machineset_role }}-{{
$zone }}
          template:
            metadata:
              labels:
                machine.openshift.io/cluster-api-cluster: {{ $infrastructure_id }}
                machine.openshift.io/cluster-api-machine-role: {{ $machineset_role }}
                machine.openshift.io/cluster-api-machine-type: {{ $machineset_role }}
                machine.openshift.io/cluster-api-machineset: {{ $infrastructure_id }}-{{ $machineset_role }}-
{{ $zone }}
            spec:
              metadata:
                labels:
                  node-role.kubernetes.io/{{ $machineset_role }}: ""
              taints:
                - key: node-role.kubernetes.io/{{ $machineset_role }}
                  effect: NoSchedule
              providerSpec:
                value:
                  ami:
                    id: {{ $worker_ms.spec.template.spec.providerSpec.value.ami.id }}
                  apiVersion: awsproviderconfig.openshift.io/v1beta1
                  blockDevices:
                    - ebs:
                        encrypted: true
                        iops: 2000
```

```
                kmsKey:
                  arn: ''
                volumeSize: 500
                volumeType: io1
            credentialsSecret:
              name: aws-cloud-credentials
            deviceIndex: 0
            instanceType: {{ $worker_ms.spec.template.spec.providerSpec.value.instanceType }}
            iamInstanceProfile:
              id: {{ $infrastructure_id }}-worker-profile
            kind: AWSMachineProviderConfig
            placement:
              availabilityZone: {{ $zone }}
              region: {{ $region }}
            securityGroups:
              - filters:
                  - name: tag:Name
                    values:
                      - {{ $infrastructure_id }}-worker-sg
            subnet:
              filters:
                - name: tag:Name
                  values:
                    - {{ $infrastructure_id }}-private-{{ $zone }}
            tags:
              - name: kubernetes.io/cluster/{{ $infrastructure_id }}
                value: owned
            userDataSecret:
              name: worker-user-data
{{- end }}
```

2.8.3.3. テンプレート処理のバイパス

Red Hat AdvancedClusterManagement による処理を目的としていないテンプレートを含めて、ポリシーを作成する場合があります。デフォルトでは、Red Hat Advanced Cluster Management は全テンプレートを処理します。

ハブクラスターのテンプレート処理を省略するには、**{{ template content }}** を **{{ `{{ template content }}` }}** に変更する必要があります。

または、**Policy** の **ConfigurationPolicy** セクションに **policy.open-cluster-management.io/disable-templates: "true"** のアノテーションを追加します。このアノテーションを追加する場合に、1 つ前の回避策は必要ありません。**ConfigurationPolicy** のテンプレート処理はバイパスされます。

2.8.3.4. 関連情報

- 詳細は、テンプレート関数 を参照してください。
- テンプレート処理 に戻る
- 詳細は、Kubernetes 設定ポリシーコントローラー を参照してください。
- Red Hat OpenShift Container Platform の etcd 暗号化のドキュメント も参照してください。

2.9. セキュリティーポリシーの管理

セキュリティーポリシーを作成して、指定のセキュリティー標準、カテゴリー、制御をもとにクラスターのコンプライアンスを報告して検証します。

以下のセクションを参照してください。

- セキュリティーポリシーの作成
- セキュリティーポリシーの更新
- セキュリティーポリシーの削除
- ポリシーによって作成されたリソースのクリーンアップ

2.9.1. セキュリティーポリシーの作成

コマンドラインインターフェイス (CLI) またはコンソールからセキュリティーポリシーを作成できます。

必要なアクセス権限: クラスターの管理者

重要: * ポリシーを特定のクラスターに適用するには、配置および配置バインディングを定義する必要があります。**PlacementBinding** リソースは配置をバインドします。クラスターの **Label selector** フィールドに有効な値を入力して、**Placement** および **PlacementBinding** リソースを定義してください。* **Placement** リソースを使用するには、**ManagedClusterSetBinding** リソースを使用して、**ManagedClusterSet** リソースを **Placement** リソースの namespace にバインドする必要があります。詳細は、ManagedClusterSetBinding リソースの作成 を参照してください。

2.9.1.1. コマンドラインインターフェイスからのセキュリティーポリシーの作成

コマンドラインインターフェイス (CLI) からポリシーを作成するには、以下の手順を実行します。

1. 以下のコマンドを実行してポリシーを作成します。

```
oc create -f policy.yaml -n <policy-namespace>
```

2. ポリシーが使用するテンプレートを定義します。.yaml ファイルを編集し、**policy-templates** フィールドを追加してテンプレートを定義します。ポリシーは以下の YAML ファイルのようになります。

```
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  name: policy1
spec:
  remediationAction: "enforce" # or inform
  disabled: false # or true
  namespaceSelector:
    include:
    - "default"
    - "my-namespace"
  policy-templates:
    - objectDefinition:
        apiVersion: policy.open-cluster-management.io/v1
        kind: ConfigurationPolicy
        metadata:
```

```
        name: operator
        # namespace: # will be supplied by the controller via the namespaceSelector
      spec:
        remediationAction: "inform"
        object-templates:
        - complianceType: "musthave" # at this level, it means the role must exist and must
have the following rules
          apiVersion: rbac.authorization.k8s.io/v1
          kind: Role
          metadata:
            name: example
          objectDefinition:
            rules:
              - complianceType: "musthave" # at this level, it means if the role exists the rule is a
musthave
                apiGroups: ["extensions", "apps"]
                resources: ["deployments"]
                verbs: ["get", "list", "watch", "create", "delete","patch"]
```

3. **PlacementBinding** リソースを定義して、ポリシーを **Placement** リソースにバインドします。**PlacementBinding** リソースは、次の YAML の例のようになります。

```
apiVersion: policy.open-cluster-management.io/v1
kind: PlacementBinding
metadata:
  name: binding1
placementRef:
  name: placement1
  apiGroup: cluster.open-cluster-management.io
  kind: Placement
subjects:
- name: policy1
  apiGroup: policy.open-cluster-management.io
  kind: Policy
```

2.9.1.1.1. CLI からのセキュリティーポリシーの表示

以下の手順を実行して、CLI からセキュリティーポリシーを表示します。

1. 以下のコマンドを実行して、特定のセキュリティーポリシーの詳細を表示します。

```
oc get policies.policy.open-cluster-management.io <policy-name> -n <policy-namespace> -o
yaml
```

2. 以下のコマンドを実行して、セキュリティーポリシーの詳細を表示します。

```
oc describe policies.policy.open-cluster-management.io <policy-name> -n <policy-
namespace>
```

2.9.1.2. コンソールからのクラスターセキュリティーポリシーの作成

Red Hat Advanced Cluster Management にログインしたら、**Governance** ページに移動し、**Create policy** をクリックします。コンソールから新規ポリシーを作成すると、YAML エディターで YAML ファイルも作成されます。YAML エディターを表示するには、**Create policy** フォームの最初にトグルを選

択して有効にします。

1. **Create policy** フォームに入力し、**Submit** ボタンを選択します。YAML ファイルは以下のポリシーのようになります。

```
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  name: policy-pod
  annotations:
    policy.open-cluster-management.io/categories:
'SystemAndCommunicationsProtections,SystemAndInformationIntegrity'
    policy.open-cluster-management.io/controls: 'control example'
    policy.open-cluster-management.io/standards: 'NIST,HIPAA'
    policy.open-cluster-management.io/description:
spec:
  complianceType: musthave
  namespaces:
    exclude: ["kube*"]
    include: ["default"]
    pruneObjectBehavior: None
  object-templates:
  - complianceType: musthave
    objectDefinition:
      apiVersion: v1
      kind: Pod
      metadata:
        name: pod1
      spec:
        containers:
        - name: pod-name
          image: 'pod-image'
          ports:
          - containerPort: 80
  remediationAction: enforce
  disabled: false
```

次の **PlacementBinding** の例を参照してください。

```
apiVersion: policy.open-cluster-management.io/v1
kind: PlacementBinding
metadata:
  name: binding-pod
placementRef:
  name: placement-pod
  kind: Placement
  apiGroup: cluster.open-cluster-management.io
subjects:
- name: policy-pod
  kind: Policy
  apiGroup: policy.open-cluster-management.io
```

次の **Placement** の例を参照してください。

```
apiVersion: cluster.open-cluster-management.io/v1beta1
```

```
 kind: Placement
 metadata:
   name: placement-pod
spec:
  predicates:
  - requiredClusterSelector:
      labelSelector:
        matchLabels:
          cloud: "IBM"
```

2. **オプション:** ポリシーの説明を追加します。

3. **Create Policy** をクリックします。コンソールからセキュリティーポリシーが作成されました。

2.9.1.2.1. コンソールからのセキュリティーポリシーの表示

コンソールからセキュリティーポリシーとステータスを表示します。

1. **Governance** ページに移動し、ポリシー表の一覧を表示します。**注記**: ポリシー表の一覧をフィルタリングするには、**Policies** タブまたは **Cluster violations** タブを選択します。

2. 詳細を表示するポリシーを 1 つ選択します。**Details**、**Clusters**、および **Templates** タブが表示されます。クラスターまたはポリシーのステータスを判断できない場合は、**No status** メッセージが表示されます。

3. または、**Policies** タブを選択してポリシーのリストを表示します。ポリシーの行をデプロイメントすると、**Description**、**Standars**、**Controls**、および **Categories** の詳細が表示されます。

2.9.1.3. CLI からのポリシーセットの作成

デフォルトでは、ポリシーまたは配置のないポリシーセットが作成されます。ポリシーセットの配置を作成し、クラスターに存在するポリシーを 1 つ以上設定する必要があります。ポリシーセットを作成する場合は、多くのポリシーを追加できます。

以下のコマンドを実行して CLI からポリシーセットを作成します。

```
oc apply -f <policyset-filename>
```

2.9.1.4. コンソールからのポリシーセットの作成

1. ナビゲーションメニューから **Govern** を選択します。

2. **Policy sets** タブを選択します。

3. **Create policy set**ボタンを選択し、フォームを完了します。

4. ポリシーセットの詳細を追加し、**送信** ボタンを選択します。

ポリシーがポリシーテーブルからリスト表示されます。

2.9.2. セキュリティーポリシーの更新

セキュリティーポリシーを更新する方法を学びます。

2.9.2.1. CLI からのポリシーセットへのポリシーの追加

1. 次のコマンドを実行して、ポリシーセットを編集します。

```
oc edit policysets <your-policyset-name>
```

2. ポリシーセットの **policies** セクションのリストにポリシー名を追加します。
3. 次のコマンドを使用して、追加したポリシーをポリシーセットの配置セクションに適用します。

```
oc apply -f <your-added-policy.yaml>
```

PlacementBinding と **Placement** の両方が作成されます。

注記: 配置バインディングを削除すると、ポリシーはポリシーセットによって配置されます。

2.9.2.2. コンソールからのポリシーセットへのポリシーの追加

1. **Policy sets** タブを選択して、ポリシーセットにポリシーを追加します。
2. Actions アイコンを選択し、**Edit** を選択します。**Edit policy set** フォームが表示されます。
3. フォームの **Policies** セクションに移動し、ポリシーセットに追加するポリシーを選択します。

2.9.2.3. セキュリティーポリシーの無効化

デフォルトでは、ポリシーは有効です。コンソールからポリシーを無効にします。

Red Hat Advanced Cluster Management for Kubernetes コンソールにログインしたら、**Governance** ページに移動し、ポリシー表のリストを表示します。

Actions アイコン > **Disable policy** の順に選択します。**Disable Policy** ダイアログボックスが表示されます。

Disable policy をクリックします。ポリシーが無効化されました。

2.9.3. セキュリティーポリシーの削除

CLI またはコンソールからセキュリティーポリシーを削除します。

- CLI からセキュリティーポリシーを削除します。
  a. 以下のコマンドを実行してセキュリティーポリシーを削除します。

```
oc delete policies.policy.open-cluster-management.io <policy-name> -n <policy-namespace>
```

  ポリシーを削除すると、ターゲットクラスターから削除されます。次のコマンドを実行して、ポリシーが削除されたことを確認します: **oc getpolicy.policy.open-cluster-management.io <policy-name> -n <policy-namespace>**

- コンソールからセキュリティーポリシーを削除します。
  ナビゲーションメニューから **Governance** をクリックし、ポリシー表のリストを表示します。
  ポリシー違反表で、削除するポリシーの **Actions** アイコンをクリックします。

Remove をクリックします。Remove policy ダイアログボックスから、Remove policy をクリックします。

2.9.3.1. コンソールからのポリシーセットの削除

1. Policy sets タブから、ポリシーセットの Actions アイコンを選択します。Delete をクリックすると、Permanently delete Policyset? ダイアログボックスが表示されます。
2. Delete ボタンをクリックします。

2.9.4. ポリシーによって作成されたリソースのクリーンアップ

ポリシーによって作成されたリソースをクリーンアップするには、設定ポリシーで **pruneObjectBehavior** パラメーターを使用します。**pruneObjectBehavior** が設定されている場合、関連するオブジェクトは、関連する設定ポリシー (または親ポリシー) が削除された後にのみクリーンアップされます。

パラメーターに使用できる値について、次の説明を参照してください。

- **DeleteIfCreated**: ポリシーによって作成されたすべてのリソースをクリーンアップします。
- **DeleteAll**: ポリシーによって管理されるすべてのリソースをクリーンアップします。
- **None**: これはデフォルト値であり、関連するリソースが削除されない以前のリリースと同じ動作を維持します。

コマンドラインからポリシーを作成するときに、YAML ファイルに値を直接設定できます。

コンソールから、Policy templates ステップの Prune Object Behavior セクションで値を選択できます。

注記:

- Operator をインストールするポリシーに **pruneObjectBehavior** パラメーターが定義されている場合、Operator のアンインストールを完了するには、追加のクリーンアップが必要です。このクリーンアップの一環として、Operator の **ClusterServiceVersion** オブジェクトを削除する必要がある場合があります。
- マネージドクラスターで **config-policy-addon** リソースを無効にすると、**pruneObjbectBehavior** は無視されます。ポリシーの関連リソースを自動的にクリーンアップするには、アドオンを無効にする前に、マネージドクラスターからポリシーを削除する必要があります。

2.9.5. 関連情報

- ポリシー YAML ファイルに関する詳細は、ポリシーの概要 を参照してください。
- 有効な式については、Kubernetes ドキュメントの セットベースの要件をサポートするリソース を参照してください。
- 安定した **Policysets** を表示します。これには、デプロイメントに Policy Generator が必要です (PolicySets--Stable)。
- ポリシーに関する他のトピックについては、ガバナンス を参照してください。

2.9.6. 設定ポリシーの管理

設定ポリシーの作成、適用、表示、および更新について説明します。

必要なアクセス権限: 管理者およびクラスター管理者

- 設定ポリシーの作成
- 設定ポリシーの更新
- 設定ポリシーの削除

2.9.6.1. 設定ポリシーの作成

設定ポリシーの YAML ファイルは、コマンドラインインターフェイス (CLI) またはコンソールから作成できます。

既存の Kubernetes マニフェストがある場合は、ポリシージェネレーターを使用して、ポリシーにマニフェストを自動的に含めることを検討してください。ポリシージェネレーター ドキュメントを参照してください。設定ポリシーの作成は、以下のセクションを参照してください。

2.9.6.1.1. CLI からの設定ポリシーの作成

CLI から設定ポリシーを作成するには、以下の手順を実行します。

1. 設定ポリシーの YAML ファイルを作成します。以下のコマンドを実行します。

```
oc create -f configpolicy-1.yaml
```

設定ポリシーは以下のようになります。

```
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  name: policy-1
  namespace: my-policies
policy-templates:
- apiVersion: policy.open-cluster-management.io/v1
  kind: ConfigurationPolicy
  metadata:
    name: mustonlyhave-configuration
  spec:
    namespaceSelector:
      include: ["default"]
      exclude: ["kube-system"]
    remediationAction: inform
    disabled: false
    complianceType: mustonlyhave
    object-templates:
```

2. 以下のコマンドを実行してポリシーを適用します。

```
oc apply -f <policy-file-name>  --namespace=<namespace>
```

3. 以下のコマンドを実行してポリシーのリストを確認します。

```
oc get policies.policy.open-cluster-management.io --namespace=<namespace>
```

設定ポリシーが作成されました。

2.9.6.1.2. CLI からの設定ポリシーの表示

CLI から設定ポリシーを表示するには、以下の手順を実行します。

1. 以下のコマンドを実行して、特定の設定ポリシーの詳細を表示します。

```
oc get policies.policy.open-cluster-management.io <policy-name> -n <namespace> -o yaml
```

2. 以下のコマンドを実行して、設定ポリシーの詳細を表示します。

```
oc describe policies.policy.open-cluster-management.io <name> -n <namespace>
```

2.9.6.1.3. コンソールからの設定ポリシーの作成

コンソールから設定ポリシーを作成すると、YAML エディターで YAML ファイルも作成されます。

1. コンソールからクラスターにログインし、ナビゲーションメニューから **Governance** を選択します。
2. **Create policy** をクリックします。仕様パラメーターの設定ポリシーのいずれかを選択して、作成するポリシーを指定します。
3. ポリシーフォームを完了して、設定ポリシーの作成を続行します。以下のフィールドに適切な値を入力するか、選択します。
   - Name
   - Specifications
   - Cluster selector
   - Remediation action
   - Standards
   - Categories
   - Controls
4. **Create** をクリックします。設定ポリシーが作成されました。

2.9.6.1.4. コンソールからの設定ポリシーの表示

コンソールから設定ポリシーおよびそのステータスを表示します。

コンソールからクラスターにログインしたら、**Governance** を選択してポリシー表の一覧を表示します。**注記:** ポリシー表の一覧をフィルタリングするには、**All policies** タブまたは **Cluster violations** タブを選択します。

詳細を表示するポリシーを 1 つ選択します。**Details**、**Clusters**、および **Templates** タブが表示されます。

2.9.6.2. 設定ポリシーの更新

設定ポリシーの更新については、以下のセクションを参照してください。

2.9.6.2.1. 設定ポリシーの無効化

設定ポリシーを無効にします。前述の説明と同様に、ログインして **ガバナンス** ページに移動し、タスクを完了します。

1. 表リストから設定ポリシーの **Actions** アイコンを選択し、**Disable** をクリックします。**Disable Policy** ダイアログボックスが表示されます。
2. **Disable policy** をクリックします。

ポリシーは無効になっていますが、削除されていません。

2.9.6.3. 設定ポリシーの削除

CLI またはコンソールから設定ポリシーを削除します。

- 次の手順で、CLI から設定ポリシーを削除します。
  1. 次のコマンドを実行して、1 つまたは複数のターゲットクラスターからポリシーを削除します。

```
oc delete policies.policy.open-cluster-management.io <policy-name> -n <namespace>
```

  2. 以下のコマンドを実行して、ポリシーが削除されていることを確認します。

```
oc get policies.policy.open-cluster-management.io <policy-name> -n <namespace>
```

- 次の手順で、コンソールから設定ポリシーを削除します。
  1. ナビゲーションメニューから **Governance** をクリックし、ポリシー表のリストを表示します。
  2. ポリシー違反テーブルで削除するポリシーの **Actions** アイコンをクリックし、**Remove** をクリックします。
  3. **Remove policy** ダイアログボックスから、**Remove policy** をクリックします。

ポリシーが削除されました。

2.9.6.4. 関連情報

- CM-Configuration-Management フォルダーから RedHat Advanced Cluster Management でサポート対象の設定ポリシーのサンプルを参照してください。
- または、サンプル設定ポリシーの表 を参照して、コントローラーによってモニターされる他の設定ポリシーを確認することもできます。他のポリシーの管理については、セキュリティーポリシーの管理 を参照してください。

2.9.7. 非接続環境での Operator ポリシーの管理

インターネットに接続していない (切断) Red Hat OpenShift Container Platform クラスターに Red Hat

Advanced Cluster Management for Kubernetes ポリシーをデプロイしないといけない場合があります。デプロイするポリシーを使用して Operator Lifecycle Manager Operator をインストールするポリシーをデプロイする場合は、Operator カタログのミラーリング の手順に従う必要があります。

Operator イメージへのアクセスを検証するには、次の手順を実行します。

1. ポリシーで使用するために必要なパッケージが利用可能であることを検証するには、必要なパッケージが利用可能であることの確認 を参照してください。次のポリシーがデプロイされているマネージドクラスターで使用される各イメージレジストリーの可用性を検証する必要があります。
   - **container-security-operator**
   - **非推奨: gatekeeper-operator-product**
   - **compliance-operator**
2. ソースが利用可能であることを検証するには、イメージコンテンツソースポリシーの設定 を参照してください。イメージコンテンツソースポリシーは、切断されたマネージドクラスターのそれぞれに存在する必要があり、プロセスを簡素化するためにポリシーを使用してデプロイできます。次のイメージソースの場所の表を参照してください。

ガバナンスポリシーの種類	イメージソースの場所
コンテナーのセキュリティー	**registry.redhat.io/quay**
コンプライアンス	**registry.redhat.io/compliance**
ゲートキーパー	**registry.redhat.io/rhacm2**

2.9.8. ポリシーセットを使用した Red Hat OpenShift Platform Plus のインストール

Red Hat Openshift Platform Plus ポリシーセットを適用するためのガイダンスを読み続けてください。Red Hat OpenShift ポリシーセットを適用すると、Red Hat Advanced Cluster Security で保護されたクラスターサービスと Compliance Operator がすべての OpenShift Container Platform マネージドクラスターにデプロイされます。

2.9.8.1. 前提条件

ポリシーセットを適用する前に、次の手順を完了してください。

1. サブスクリプションをクラスターに適用できるようにするには、**policy-configure-subscription-admin-hub.yaml** ポリシーを適用し、修復アクションを **enforce** に設定する必要があります。次の YAML をコピーして、コンソールの YAML エディターに貼り付けます。

```
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  name: policy-configure-subscription-admin-hub
  annotations:
    policy.open-cluster-management.io/standards: NIST SP 800-53
    policy.open-cluster-management.io/categories: CM Configuration Management
    policy.open-cluster-management.io/controls: CM-2 Baseline Configuration
spec:
```

```
remediationAction: inform
disabled: false
policy-templates:
  - objectDefinition:
      apiVersion: policy.open-cluster-management.io/v1
      kind: ConfigurationPolicy
      metadata:
        name: policy-configure-subscription-admin-hub
      spec:
        remediationAction: inform
        severity: low
        object-templates:
          - complianceType: musthave
            objectDefinition:
              apiVersion: rbac.authorization.k8s.io/v1
              kind: ClusterRole
              metadata:
                name: open-cluster-management:subscription-admin
              rules:
              - apiGroups:
                - app.k8s.io
                resources:
                - applications
                verbs:
                - '*'
              - apiGroups:
                - apps.open-cluster-management.io
                resources:
                - '*'
                verbs:
                - '*'
              - apiGroups:
                - ""
                resources:
                - configmaps
                - secrets
                - namespaces
                verbs:
                - '*'
          - complianceType: musthave
            objectDefinition:
              apiVersion: rbac.authorization.k8s.io/v1
              kind: ClusterRoleBinding
              metadata:
                name: open-cluster-management:subscription-admin
              roleRef:
                apiGroup: rbac.authorization.k8s.io
                kind: ClusterRole
                name: open-cluster-management:subscription-admin
              subjects:
              - apiGroup: rbac.authorization.k8s.io
                kind: User
                name: kube:admin
              - apiGroup: rbac.authorization.k8s.io
                kind: User
                name: system:admin
```

```
---
apiVersion: policy.open-cluster-management.io/v1
kind: PlacementBinding
metadata:
  name: binding-policy-configure-subscription-admin-hub
placementRef:
  name: placement-policy-configure-subscription-admin-hub
  kind: Placement
  apiGroup: cluster.open-cluster-management.io
subjects:
- name: policy-configure-subscription-admin-hub
  kind: Policy
  apiGroup: policy.open-cluster-management.io
---
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement-policy-configure-subscription-admin-hub
spec:
  predicates:
  - requiredClusterSelector:
      labelSelector:
        matchExpressions:
        - {key: name, operator: In, values: ["local-cluster"]}
```

2. コマンドラインインターフェイスから以前の YAML を適用するには、以下のコマンドを実行します。

```
oc apply -f policy-configure-subscription-admin-hub.yaml
```

3. Policy Generator kusTOMize プラグインをインストールします。KusTOMize v4.5 以降を使用してください。Operator をインストールするためのポリシーの生成 を参照してください。

4. ポリシーは **policies** namespace にインストールされます。その nemaspace を **ClusterSet** にバインドする必要があります。たとえば、以下のサンプル YAML をコピーして適用し、namespace をデフォルトの **ClusterSet** にバインドします。

```
apiVersion: cluster.open-cluster-management.io/v1beta2
kind: ManagedClusterSetBinding
metadata:
    name: default
    namespace: policies
spec:
    clusterSet: default
```

5. 次のコマンドを実行して、コマンドラインインターフェイスから **ManagedClusterSetBinding** リソースを適用します。

```
oc apply -f managed-cluster.yaml
```

前提条件を満たしたら、ポリシーセットを適用できます。

2.9.8.2. Red Hat OpenShift Platform Plus ポリシーセットの適用

1. Red Hat OpenShift Plus の前提条件設定が含まれている **openshift-plus/policyGenerator.yaml** ファイルを使用します。**openshift-plus/policyGenerator.yaml** を参照してください。

2. **kusTOMize** コマンドを使用して、ポリシーをハブクラスターに適用します。

```
kustomize build --enable-alpha-plugins  | oc apply -f -
```

注記: インストールしたくない OpenShift Platform Plus のコンポーネントについては、**policyGenerator.yaml** ファイルを編集し、それらのコンポーネントのポリシーを削除またはコメントアウトします。

2.9.8.3. 関連情報

- ポリシーセットの概要は、Red Hat OpenShift Platform Plus policy set を参照してください。
- トピックの最初の ポリシーセットを使用した Red Hat OpenShift Platform Plus のインストール に戻ります。

2.9.9. OperatorPolicy リソースを使用してオペレーターをインストールする (テクノロジープレビュー)

マネージドクラスターに Operator Lifecycle Manager (OLM) マネージドオペレーターをインストールするには、**Policy** 定義で **OperatorPolicy** ポリシーテンプレートを使用します。

2.9.9.1. Quay をインストールするための OperatorPolicy リソースの作成

Red Hat Operator カタログを使用して **stable-3.11** チャネルに最新の Quay Operator をインストールする次の Operator ポリシーサンプルを参照してください。

```
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  name: install-quay
  namespace: open-cluster-management-global-set
spec:
  disabled: false
  policy-templates:
    - objectDefinition:
        apiVersion: policy.open-cluster-management.io/v1beta1
        kind: OperatorPolicy
        metadata:
          name: install-quay
        spec:
          remediationAction: enforce
          severity: critical
          complianceType: musthave
          subscription:
            channel: stable-3.11
            installPlanApproval: Automatic
            name: quay-operator
            source: redhat-operators
            sourceNamespace: openshift-marketplace
```

OperatorPolicy ポリシーテンプレートを追加すると、コントローラーを使用してクラスター上に **operatorGroup** オブジェクトおよび **subscription** オブジェクトが作成されます。その結果、インス

トールの残りの部分は OLM によって完了します。マネージドクラスター上の **OperatorPolicy** リソースの **.status.Conditions** フィールドと **.status.relatedObjects** フィールドに所有リソースの健全性を表示します。

Operator ポリシーのステータスを確認するには、マネージドクラスターで次のコマンドを実行します。

```
oc -n <managed cluster namespace> get operatorpolicy install-quay
```

2.9.9.2. 関連情報

Operator ポリシーコントローラー (テクノロジープレビュー) を参照してください。

2.10. ポリシーの依存関係

依存関係は、依存関係の基準が満たされたときにポリシーまたはポリシーテンプレートをアクティブ化するために使用できます。次のフィールドは、マネージドクラスター、**dependencies**、および **extraDependencies** でチェックされます。依存関係が満たされない場合、複製されたポリシーテンプレートのテンプレートステータスに詳細が表示されます。

必要なアクセス権: ポリシー管理者

次のポリシー依存関係の例を表示します。ここで、**ScanSettingBinding** は **upstream-compliance-operator** ポリシーがマネージドクラスターですでに準拠している場合にのみ作成されます。

```
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  annotations:
    policy.open-cluster-management.io/categories: CM Configuration Management
    policy.open-cluster-management.io/controls: CM-2 Baseline Configuration
    policy.open-cluster-management.io/standards: NIST SP 800-53
    policy.open-cluster-management.io/description:
  name: moderate-compliance-scan
  namespace: default
spec:
  dependencies:
  - apiVersion: policy.open-cluster-management.io/v1
    compliance: Compliant
    kind: Policy
    name: upstream-compliance-operator
    namespace: default
  disabled: false
  policy-templates:
  - objectDefinition:
      apiVersion: policy.open-cluster-management.io/v1
      kind: ConfigurationPolicy
      metadata:
        name: moderate-compliance-scan
      spec:
        object-templates:
        - complianceType: musthave
          objectDefinition:
            apiVersion: compliance.openshift.io/v1alpha1
            kind: ScanSettingBinding
```

```
    metadata:
      name: moderate
      namespace: openshift-compliance
    profiles:
    - apiGroup: compliance.openshift.io/v1alpha1
      kind: Profile
      name: ocp4-moderate
    - apiGroup: compliance.openshift.io/v1alpha1
      kind: Profile
      name: ocp4-moderate-node
    settingsRef:
      apiGroup: compliance.openshift.io/v1alpha1
      kind: ScanSetting
      name: default
  remediationAction: enforce
  severity: low
```

注記: 依存関係を使用して、別のクラスターのポリシーのステータスに基づいて、あるクラスターにポリシーを適用することはできません。

2.11. ハブクラスターのセキュリティー保護

ハブクラスターのセキュリティーを強化して、Red Hat Advanced Cluster Management for Kubernetes のインストールを保護します。以下の手順を実行します。

1. Red Hat OpenShift Container Platform のセキュリティーを確保します。詳細は、OpenShift Container Platform のセキュリティーおよびコンプライアンス を参照してください。
2. ロールベースアクセス制御 (RBAC) を設定します。詳細は、ロールベースのアクセス制御 を参照してください。
3. 証明書をカスタマイズします。(証明書 を参照)。
4. クラスターの認証情報を定義します。(認証情報の管理 を参照)。
5. クラスターのセキュリティー強化に利用できるポリシーを確認します。サポート対象のポリシー を参照してください。

第3章 GATEKEEPER OPERATOR

Gatekeeper は、Open Policy Agent (OPA) で実行されるカスタムリソース定義ベースのポリシーを強制できる監査機能を備えた検証 Webhook です。gatekeeper Operator ポリシーを使用して、クラスターに gatekeeper をインストールできます。gatekeeper 制約を使用して、Kubernetes リソースのコンプライアンスを評価できます。ポリシーエンジンとして OPA を活用し、ポリシー言語に Rego を使用できます。

前提条件: Gatekeeper をインストールし、クラスターに Gatekeeper ポリシーを適用するには、Red Hat Advanced Cluster Management for Kubernetes または Red Hat OpenShift Container Platform Plus サブスクリプションが必要です。Gatekeeper は、最新バージョンの Red Hat Advanced Cluster Management でサポートされるバージョン 3.11 を除く、OpenShift Container Platform のバージョンでのみサポートされます。

Gatekeeper Operator の使用方法の詳細は、以下を参照してください。

Gatekeeper の制約および制約テンプレートの統合 Gatekeeper Operator ポリシーの管理

3.1. GATEKEEPER の制約および制約テンプレートの統合

Gatekeeper ポリシーは、制約テンプレート (**ConstraintTemplates**) と制約を使用して記述されます。Red Hat Advanced Cluster Management ポリシーで gatekeepr 制約を使用する以下の YAML の例を表示します。

- **ConstraintTemplates** と制約: Red Hat Advanced Cluster Management ポリシーを使用して Gatekeeper 統合機能を使用し、Gatekeeper 制約のマルチクラスター分散とハブクラスターでの Gatekeeper 監査結果の集約を行います。次の例では、Gatekeeper **ConstraintTemplate** と制約 (**K8sRequiredLabels**) を定義して、**gatekeeper** ラベルがすべての namespace に設定されていることを確認します。

```
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  name: require-gatekeeper-labels-on-ns
spec:
  remediationAction: inform (1)
  disabled: false
  policy-templates:
    - objectDefinition:
        apiVersion: templates.gatekeeper.sh/v1beta1
        kind: ConstraintTemplate
        metadata:
          name: k8srequiredlabels
          annotations:
            policy.open-cluster-management.io/severity: low (2)
        spec:
          crd:
            spec:
              names:
                kind: K8sRequiredLabels
              validation:
                openAPIV3Schema:
                  properties:
                    labels:
                      type: array
```

```
                items: string
      targets:
        - target: admission.k8s.gatekeeper.sh
          rego: |
            package k8srequiredlabels
            violation[{"msg": msg, "details": {"missing_labels": missing}}] {
              provided := {label | input.review.object.metadata.labels[label]}
              required := {label | label := input.parameters.labels[_]}
              missing := required - provided
              count(missing) > 0
              msg := sprintf("you must provide labels: %v", [missing])
            }
  - objectDefinition:
      apiVersion: constraints.gatekeeper.sh/v1beta1
      kind: K8sRequiredLabels
      metadata:
        name: ns-must-have-gk
        annotations:
          policy.open-cluster-management.io/severity: low (3)
      spec:
        enforcementAction: dryrun
        match:
          kinds:
            - apiGroups: [""]
              kinds: ["Namespace"]
        parameters:
          labels: ["gatekeeper"]
```

(1) **remediationAction** が **inspired** に設定されているため、Gatekeeper 制約の **enforcementAction** フィールドは **warn** にオーバーライドされます。これは、**gatekeeper** ラベルが欠落している namespace の作成または更新を gatekeeper が検出し、警告することを意味します。ポリシー **remediationAction** が **enforce** に設定されている場合、Gatekeeper 制約の **enforceAction** フィールドは **deny** にオーバーライドされます。このコンテキストでは、この設定により、**gatekeeper** ラベルが欠落している namespace をユーザーが作成または更新できなくなります。

(2) (3) **オプション:** 各 Gatekeeper 制約または制約テンプレートに対して、**policy.open-cluster-management.io/severity** アノテーションの重大度値を設定します。有効な値は、他の Red Hat Advanced Cluster Management ポリシータイプと同じです (**low**、**middle**、**high**、または **crity**)。

以前のポリシーでは、次のポリシーステータスメッセージが表示される場合があります: **warn - you must provide labels: {"gatekeeper"} (on Namespace default); warn - you must provide labels: {"gatekeeper"} (on Namespace gatekeeper-system)**Gatekeeper 制約または **ConstraintTemplates** を含むポリシーが削除されると、制約および **ConstraintTemplates** もマネージドクラスターから削除されます。

コンソールから特定のマネージドクラスターの Gatekeeper 監査結果を表示するには、ポリシーテンプレートの **Results** ページに移動します。検索が有効になっている場合は、監査に失敗した Kubernetes オブジェクトの YAML を表示します。

注記:

- **Related resources** セクションは、監査結果が Gatekeeper バージョン 3.9 以降で生成された場合にのみ使用できます。

    - Gatekeeper の監査機能は、デフォルトでは 1 分ごとに実行されます。監査結果はハブクラスターに返送され、マネージドクラスターの Red Hat Advanced Cluster Management ポリシーステータスで表示されます。

- **policy-gatekeeper-admission**: Red Hat Advanced Cluster Management ポリシー内の **policy-gatekeeper-admission** 設定ポリシーを使用して、ゲートキーパーアドミッション Webhook によって拒否された Kubernetes API リクエストを確認します。

```
apiVersion: policy.open-cluster-management.io/v1
kind: ConfigurationPolicy
metadata:
  name: policy-gatekeeper-admission
spec:
  remediationAction: inform # will be overridden by remediationAction in parent policy
  severity: low
  object-templates:
    - complianceType: mustnothave
      objectDefinition:
        apiVersion: v1
        kind: Event
        metadata:
          namespace: openshift-gatekeeper-system # set it to the actual namespace where gatekeeper is running if different
          annotations:
            constraint_action: deny
            constraint_kind: K8sRequiredLabels
            constraint_name: ns-must-have-gk
            event_type: violation
```

3.1.1. 関連情報

- 詳細は、**policy-gatekeeper-operator.yaml** を参照してください。
- 詳細は、OPA ゲートキーパーとは を参照してください。
- 他のポリシーの管理に関する詳細は、設定ポリシーの管理 を参照してください。
- セキュリティーフレームワークに関する他のトピックについては、ガバナンス を参照してください。

3.2. GATEKEEPER OPERATOR ポリシーの管理

Gatekeeper operator ポリシーを使用して、Gatekeeper operator と Gatekeeper をマネージドクラスターにインストールします。次のセクションでは、Gatekeeper Operator ポリシーの作成、表示、および更新について説明します。

必要なアクセス権限: クラスターの管理者

- Gatekeeper operator ポリシーを使用した Gatekeeper のインストール
- コンソールからの gatekeeper ポリシーの作成
- gatekeeper および gatekeeper Operator のアップグレード
- gatekeeper Operator ポリシーの更新

- gatekeeper Operator ポリシーの削除
- gatekeeper ポリシー、gatekeeper、および gatekeeper Operator ポリシーのアンインストール

3.2.1. Gatekeeper operator ポリシーを使用した Gatekeeper のインストール

ガバナンスフレームワークを使用して gatekeeper Operator をインストールします。gatekeeper Operator は OpenShift Container Platform カタログで利用できます。詳細は、OpenShift Container Platform ドキュメント の **Operator のクラスターへの追加** を参照してください。

設定ポリシーコントローラーを使用して gatekeeper Operator ポリシーをインストールします。インストール時に、Operator グループおよびサブスクリプションは gatekeeper Operator をプルし、これをマネージドクラスターにインストールします。次に、gatekeeper Operator は gatekeeper custom resource を作成して gatekeeper を設定します。Gatekeeper operator のカスタムリソース のサンプルを表示します。

gatekeeper Operator ポリシーは、Red Hat Advanced Cluster Management 設定ポリシーコントローラーによって監視されます。ここでは、**enforce** 修復アクションがサポートされます。gatekeeper Operator ポリシーは、**enforce** に設定されるとコントローラーによって自動的に作成されます。

3.2.2. コンソールからの gatekeeper ポリシーの作成

コンソールからポリシーを作成するには、以下の手順を参照してください。

1. コンソールからポリシーを作成して、gatekeeper ポリシーをインストールします。あるいは、**Additional resources** セクションに移動してサンプル YAML を参照し、**policy-gatekeeper-operator.yaml** をデプロイします。
2. クラスターにログインしたら、**Governance** ページに移動します。
3. **Create policy** を選択します。
4. フォームを完了したら、**Specifications** フィールドから **Gatekeeper Operator** を選択します。ポリシーのパラメーター値が自動的に設定され、ポリシーはデフォルトで **inform** に設定されます。
5. gatekeeper をインストールするには、修復アクションを **enforce** に設定します。

注記: デフォルト値は Operator によって生成されます。

3.2.2.1. Gatekeeper operator のカスタムリソース

```
apiVersion: operator.gatekeeper.sh/v1alpha1
kind: Gatekeeper
metadata:
  name: gatekeeper
spec:
  audit:
    replicas: 1
      auditEventsInvolvedNamespace: Enabled (1)
    logLevel: DEBUG
    auditInterval: 10s
    constraintViolationLimit: 55
    auditFromCache: Enabled
    auditChunkSize: 66
```

```
    emitAuditEvents: Enabled
    resources:
      limits:
        cpu: 500m
        memory: 150Mi
      requests:
        cpu: 500m
        memory: 130Mi
  validatingWebhook: Enabled
  mutatingWebhook: Enabled
  webhook:
    replicas: 3
    emitAdmissionEvents: Enabled
    admissionEventsInvolvedNamespace: Enabled 2
    disabledBuiltins: - http.send
    operations: 3
     - "CREATE"
     - "UPDATE"
     - "CONNECT"
    failurePolicy: Fail
    resources:
      limits:
        cpu: 480m
        memory: 140Mi
      requests:
        cpu: 400m
        memory: 120Mi
  nodeSelector:
    region: "EMEA"
  affinity:
    podAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        - labelSelector:
            matchLabels:
              auditKey: "auditValue"
          topologyKey: topology.kubernetes.io/zone
  tolerations:
    - key: "Example"
      operator: "Exists"
      effect: "NoSchedule"
  podAnnotations:
    some-annotation: "this is a test"
    other-annotation: "another test"
```

+ <1> 作成する namespace 監査イベントを管理するには、**auditEventsInvolvedNamespace** パラメーターを有効にします。有効にすると、Gatekeeper コントローラーのデプロイメントに次の引数 **--audit-events-involved-namespace: true** 追加されます。<2> 作成する namespace アドミッションイベントを管理するには、**admissionEventsInvolvedNamespace** パラメーターを有効にします。有効にすると、Gatekeeper コントローラーのデプロイメントに引数 **--admission-events-involved-namespace: true** が追加されます。<3> 次の値を使用して、Webhook 操作を管理できます。**operations** パラメーターには、**"CREATE"**、**"UPDATE"**、**"CONNECT"**、および **"DELETE"** の値を使用します。

3.2.2.2. auditFromCache の同期情報の設定

Gatekeeper Operator は、**auditFromCache** 監査内のカスタムリソース定義の設定を公開しますが、これはデフォルトでは無効になっています。**AuditFromCache** を有効にする場合は、**config.gatekeeper.sh** で同期の詳細を設定する必要があります。gatekeeper ドキュメントの Configuring Audit を参照してください。

Gatekeeper Operator は、ユーザーがインストールした制約からリソースを収集し、それらのリソースを設定リソースに挿入します。リソースが存在しない場合、Operator は設定リソースを作成します。

1. 次の例に示すように、**Gatekeeper** リソースで **auditFromCache** を **Automatic** に設定します。

```
apiVersion: operator.gatekeeper.sh/v1alpha1
kind: Gatekeeper
metadata:
  name: gatekeeper
spec:
  audit: replicas: 2
    logLevel: DEBUG
    auditFromCache: Automatic
```

次の例のように、gatekeeper Operator が config ファイルに **syncOnlys** パラメーターセクションを追加することを確認してください。

```
apiVersion: config.gatekeeper.sh/v1alpha1
kind: Config
metadata:
 name: config
 namespace: "openshift-gatekeeper-system"
spec:
 sync:
   syncOnly:
   - group: ""
     version: "v1"
     kind: "Namespace"
   - group: ""
     version: "v1"
     kind: "Pod"
```

2. **sync** 設定の説明を取得し、ターミナルから以下のコマンドを実行します。

```
oc explain gatekeeper.spec.audit.auditFromCache
```

3.2.3. gatekeeper および gatekeeper Operator のアップグレード

gatekeeper および gatekeeper Operator のバージョンをアップグレードできます。gatekeeper Operator を gatekeeper Operator ポリシーを使用してインストールする場合は、**installPlanApproval** の値に注意してください。**installPlanApproval** が **Automatic** に設定されている場合は、Operator は自動的にアップグレードされます。

installPlanApproval が **Manual** に設定されている場合は、各クラスターの gatekeeper Operator のアップグレードを手動で承認する必要があります。

3.2.4. gatekeeper Operator ポリシーの更新

次のセクションを参照して、gatekeeper Operator ポリシーを更新する方法を確認してください。

3.2.4.1. コンソールからの gatekeeper Operator ポリシーの表示

コンソールから gatekeeper Operator ポリシーおよびそのステータスを表示します。

コンソールからクラスターにログインしたら、**Governance** をクリックし、ポリシー表の一覧を表示します。**注記**: ポリシー表の一覧をフィルタリングするには、**Policies** タブまたは **Cluster violations** タブを選択します。

詳細を表示するには、 **policy-gatekeeper-operator** ポリシーを選択します。**Clusters** タブを選択して、ポリシー違反を表示します。

3.2.4.2. gatekeeper Operator ポリシーの無効化

gatekeeper Operator ポリシーを無効にします。

Red Hat Advanced Cluster Management for Kubernetes コンソールにログインしたら、**Governance** ページに移動し、ポリシー表のリストを表示します。

policy-gatekeeper-operator ポリシーの **Actions** アイコンを選択し、**Disable** をクリックします。**Disable Policy** ダイアログボックスが表示されます。

Disable policy をクリックします。**policy-gatekeeper-operator** ポリシーが無効になりました。

3.2.5. gatekeeper Operator ポリシーの削除

CLI またはコンソールから gatekeeper Operator ポリシーを削除します。

- CLI から gatekeeper Operator ポリシーを削除します。
  a. 以下のコマンドを実行し、gatekeeper Operator ポリシーを削除します。

  ```
  oc delete policies.policy.open-cluster-management.io <policy-gatekeeper-operator-name> -n <namespace>
  ```

  ポリシーを削除すると、ターゲットクラスターから削除されます。

  b. 以下のコマンドを実行して、ポリシーが削除されていることを確認します。

  ```
  oc get policies.policy.open-cluster-management.io <policy-gatekeeper-operator-name> -n <namespace>
  ```

- コンソールから gatekeeper Operator ポリシーを削除します。
  Governance ページに移動し、ポリシー表の一覧を表示します。

  前のコンソールの手順と同様に、**policy-gatekeeper-operator** ポリシーの **Actions** アイコンをクリックします。**Remove** をクリックしてポリシーを削除します。**Remove policy** ダイアログボックスから、**Remove policy** をクリックします。

gatekeeper Operator ポリシーが削除されました。

3.2.6. gatekeeper ポリシー、gatekeeper、および gatekeeper Operator ポリシーのアンインストール

gatekeeper ポリシー、gatekeeper、および gatekeeper Operator ポリシーをアンインストールするには、以下の手順を実行します。

1. マネージドクラスターに適用される gatekeeper **Constraint** および **ConstraintTemplate** を削除します。

   a. gatekeeper Operator ポリシーを編集します。gatekeeper **Constraint** および **ConstraintTemplate** の作成に使用した **ConfigurationPolicy** テンプレートを見つけます。

   b. **ConfigurationPolicy** テンプレートの **complianceType** の値を **mustnothave** に変更します。

   c. ポリシーを保存して適用します。

2. マネージドクラスターから gatekeeper インスタンスを削除します。

   a. gatekeeper Operator ポリシーを編集します。gatekeeper カスタムリソースの作成に使用した **ConfigurationPolicy** テンプレートを見つけます。

   b. **ConfigurationPolicy** テンプレートの **complianceType** の値を **mustnothave** に変更します。

3. マネージドクラスターにある gatekeeper Operator を削除します。

   a. gatekeeper Operator ポリシーを編集します。サブスクリプション CR の作成に使用した **ConfigurationPolicy** テンプレートを見つけます。

   b. **ConfigurationPolicy** テンプレートの **complianceType** の値を **mustnothave** に変更します。

gatekeeper ポリシー、gatekeeper、および gatekeeper Operator ポリシーはアンインストールされました。

3.2.7. 関連情報

- gatekeeper の詳細は、gatekeeper 制約および制約テンプレートの統合 を参照してください。
- Policy Gatekeeper の例を参照してください。
- gatekeeper Operator ポリシーに使用できるオプションのパラメーターの説明は、Gatekeeper Helm Chart を参照してください。
- サードパーティーポリシーと製品の統合に関する詳細は、サードパーティーポリシーコントローラーの統合 を参照してください。

第4章 サードパーティーポリシーコントローラーの統合

サードパーティーポリシーを統合してポリシーテンプレート内にカスタムアノテーションを作成し、コンプライアンス標準、制御カテゴリー、制御を 1 つ以上指定します。

policy-collection/community からサードパーティーポリシーを使用することもできます。

以下のサードパーティーポリシーを統合する方法を説明します。

- Gatekeeper operator
- ポリシージェネレーター

4.1. ポリシージェネレーター

ポリシージェネレーターは、Kustomize を使用して Red Hat Advanced Cluster Management ポリシーを生成する Red Hat Advanced Cluster Management for Kubernetes アプリケーションライフサイクルサブスクリプション GitOps ワークフローの一部です。ポリシージェネレーターは、設定に使用される **PolicyGenerator** マニフェスト YAML ファイル提供の Kubernetes マニフェスト YAML ファイルから Red Hat Advanced Cluster Management ポリシーをビルドします。ポリシージェネレーターは、Kustomize ジェネレータープラグインとして実装されます。KusTOMize の詳細は、**KusTOMize のドキュメント** を参照してください。

詳細は、以下のセクションを参照してください。

- ポリシージェネレーター機能
- ポリシージェネレーターの設定構造
- ポリシージェネレーター設定の参照テーブル

4.1.1. ポリシージェネレーター機能

ポリシージェネレーターと Red Hat Advanced Cluster Management アプリケーションライフサイクルサブスクリプション GitOps ワークフローとの統合により、OpenShift Container Platform マネージドクラスターへの Kubernetes リソースオブジェクトの配布と、Red Hat Advanced Cluster Management ポリシーによる Kubernetes クラスターが簡素化されます。

ポリシージェネレーターを使用して、次のアクションを実行します。

- Kustomize ディレクトリーから作成されたマニフェストを含む、任意の Kubernetes マニフェストファイルを Red Hat Advanced Cluster Management 設定ポリシーに変換します。
- 生成された Red Hat Advanced Cluster Management ポリシーに挿入される前に、入力された Kubernetes マニフェストにパッチを適用します。
- Red Hat Advanced Cluster Management for Kubernetes で、Gatekeeper ポリシー違反について報告できるように追加の設定ポリシーを生成します。
- ハブクラスターでポリシーセットを生成します。

4.1.2. ポリシージェネレーターの設定構造

ポリシージェネレーターは、**PolicyGenerator** の種類および **policy.open-cluster-management.io/v1** API バージョンのマニフェストで設定される Kustomize ジェネレータープラグインです。

プラグインを使用するには、まず、**kustomization.yaml** ファイルに **generators** セクションを追加します。以下の例を参照してください。

```
generators:
  - policy-generator-config.yaml
```

直前の例で参照される **policy-generator-config.yaml** ファイルは、生成するポリシーの手順を含む YAML ファイルです。単純な **PolicyGenerator** 設定ファイルは以下の例のようになります。

```
apiVersion: policy.open-cluster-management.io/v1
kind: PolicyGenerator
metadata:
  name: config-data-policies
policyDefaults:
  namespace: policies
  policySets: []
policies:
  - name: config-data
    manifests:
      - path: configmap.yaml
```

configmap.yaml は、ポリシーに含まれる Kubernetes マニフェスト YAML ファイルを表します。また、Kustomize ディレクトリー、または複数の Kubernetes マニフェスト YAML ファイルを含むディレクトリーへのパスを設定できます。以下の例を参照してください。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: my-config
  namespace: default
data:
  key1: value1
  key2: value2
```

生成された **Policy**、**Placement** と **PlacementBinding** は以下の例のようになります。

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement-config-data
  namespace: policies
spec:
  predicates:
  - requiredClusterSelector:
      labelSelector:
        matchExpressions: []
---
apiVersion: policy.open-cluster-management.io/v1
kind: PlacementBinding
metadata:
  name: binding-config-data
  namespace: policies
placementRef:
  apiGroup: cluster.open-cluster-management.io
  kind: Placement
```

```
  name: placement-config-data
subjects:
- apiGroup: policy.open-cluster-management.io
  kind: Policy
  name: config-data
---
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  annotations:
    policy.open-cluster-management.io/categories: CM Configuration Management
    policy.open-cluster-management.io/controls: CM-2 Baseline Configuration
    policy.open-cluster-management.io/standards: NIST SP 800-53
    policy.open-cluster-management.io/description:
  name: config-data
  namespace: policies
spec:
  disabled: false
  policy-templates:
  - objectDefinition:
      apiVersion: policy.open-cluster-management.io/v1
      kind: ConfigurationPolicy
      metadata:
        name: config-data
      spec:
        object-templates:
        - complianceType: musthave
          objectDefinition:
            apiVersion: v1
            data:
              key1: value1
              key2: value2
            kind: ConfigMap
            metadata:
              name: my-config
              namespace: default
        remediationAction: inform
        severity: low
```

4.1.3. ポリシージェネレーター設定の参照テーブル

namespace を除く **policyDefaults** セクションのすべてのフィールドはポリシーごとにオーバーライドでき、**policySetDefaults** セクションのすべてのフィールドはポリシーセットごとにオーバーライドできることに注意してください。

表4.1 パラメーターの表

フィールド	任意または必須	説明
apiVersion	必須	この値は **policy.open-cluster-management.io/v1** に設定します。

フィールド	任意または必須	説明
kind	必須	ポリシーのタイプを指定するには、値を **PolicyGenerator** に設定します。
metadata.name	必須	ポリシーリソースを識別する名前。
placementBindingDefaults.name	任意	複数のポリシーが同じ配置を使用する場合、この名前を使用した結果、**PlacementBinding** の一意の名前が生成され、それを参照するポリシーの配列で配置バインドします。
policyDefaults	必須	**namespace** 以外の policies 配列のエントリーによって、ここでリスト表示されるデフォルト値は上書きされます。
policyDefaults.namespace	必須	すべてのポリシーの namespace。
policyDefaults.complianceType	任意	マニフェストとクラスターのオブジェクトを比較する場合のポリシーコントローラーの動作を決定します。使用できる値は、**musthave**、**mustonlyhave**、または **mustnothave** です。デフォルト値は **musthave** です。
policyDefaults.metadataComplianceType	任意	マニフェストメタデータセクションをクラスター上のオブジェクトと比較するときに、**complianceType** をオーバーライドします。使用できる値は **musthave** と **mustonlyhave** です。メタデータの **ComplianceType** のオーバーライドを避けるため、デフォルト値は空 (**{}**) です。

フィールド	任意または必須	説明
policyDefaults.categories	任意	**policy.open-cluster-management.io/categories** アノテーションで使用されるカテゴリーの配列。デフォルト値は **CM Configuration Management** です。
policyDefaults.controls	任意	**policy.open-cluster-management.io/controls** アノテーションで使用されるコントロールの配列。デフォルト値は **CM-2 Baseline Configuration** です。
policyDefaults.standards	任意	**policy.open-cluster-management.io/standards** アノテーションで使用する標準の配列。デフォルト値は **NIST SP 800-53** です。
policyDefaults.policyAnnotations	任意	ポリシーの **metadata.annotations** セクションに含まれるアノテーションです。ポリシーで指定されていない限り、すべてのポリシーに適用されます。デフォルト値は空です (**{}**)。
policyDefaults.configurationPolicyAnnotations	任意	生成された設定ポリシーに設定するアノテーションのキーと値のペアです。たとえば、**{"policy.open-cluster-management.io/disable-templates": "true"}** というパラメーターを定義することで、ポリシーテンプレートを無効にすることができます。デフォルト値は空です (**{}**)。
policyDefaults.copyPolicyMetadata	任意	すべてのポリシーのラベルとアノテーションをコピーし、レプリカポリシーに追加します。デフォルトでは **true** に設定されます。**false** に設定すると、ポリシーフレームワーク固有のポリシーラベルとアノテーションのみがレプリケートされたポリシーにコピーされます。

フィールド	任意または必須	説明
policyDefaults.severity	任意	ポリシー違反の重大度。デフォルト値は **low** です。
policyDefaults.disabled	任意	ポリシーが無効になっているかどうか、つまり、ポリシーが伝播されておらず、結果としてステータスがないことを意味します。ポリシーを有効にするデフォルト値は **false** です。
policyDefaults.remediationAction	任意	ポリシーの修復メカニズム。パラメーターの値は **enforce** および **inform** です。デフォルト値は **inform** です。
policyDefaults.namespaceSelector	namespace が指定されていない namespace 付きオブジェクトに必要	オブジェクトが適用されるマネージドクラスター内の namespace を決定します。**include** パラメーターと **exclude** パラメーターは、ファイルパス式を受け入れて、名前で namespace を含めたり除外したりします。**matchExpressions** および **matchLabels** パラメーターは、ラベルによって含める namespace を指定します。**Kubernetes のラベルとセレクター** のドキュメントを読んでください。結果のリストは、すべてのパラメーターからの結果の共通部分を使用してコンパイルされます。
policyDefaults.evaluationInterval	任意	特定のコンプライアンス状態にある場合にポリシーが評価される頻度を指定するには、パラメーター **compliant** および **noncompliant** を使用します。マネージドクラスターの CPU リソースが少ない場合、評価間隔を長くして Kubernetes API の CPU 使用率を減らすことができます。これらは期間を表す形式です。たとえば、**"1h25m3s"** は 1 時間 25 分 3 秒を表します。これらは、特定のコンプライアンス状態になった後にポリシーを評価しないように、"never" に設定することもできます。

フィールド	任意または必須	説明
policyDefaults.pruneObjectBehavior	任意	ポリシーが削除されたときに、ポリシーによって作成または監視されているオブジェクトを削除するかどうかを決定します。プルーニングは、ポリシーの修復アクションが **enforce** に設定されている場合にのみ行われます。値の例は、**DeleteIfCreated**、**DeleteAll**、または **None** です。デフォルト値は **None** です。
policyDefaults.recordDiff	任意	クラスター上のオブジェクトとポリシー内の **objectDefinition** との差異をログに記録するかどうか、および記録する場所を指定します。コントローラーログに差異を記録する場合は **Log** に設定し、差異を記録しない場合は **None** に設定します。デフォルトでは、このパラメーターは空であり、差異はログに記録されません。
policyDefaults.dependencies	任意	このポリシーが適用される前に、特定のコンプライアンス状態にある必要があるオブジェクトのリスト。**policyDefaults.orderPolicies** が **true** に設定されている場合は指定できません。
policyDefaults.dependencies[].name	必須	依存しているオブジェクトの名前。
policyDefaults.dependencies[].namespace	任意	依存しているオブジェクトの namespace。デフォルトは、ポリシージェネレーターに設定されたポリシーの namespace です。
policyDefaults.dependencies[].compliance	任意	オブジェクトが必要とするコンプライアンス状態。デフォルト値は **Compliant** です。
policyDefaults.dependencies[].kind	任意	オブジェクトの種類。デフォルトでは、種類は **Policy** に設定されています が、**ConfigurationPolicy** などのコンプライアンス状態を持つ他の種類にすることもできます。

フィールド	任意または必須	説明
policyDefaults.dependencies[].apiVersion	任意	オブジェクトの API バージョン。デフォルト値は **policy.open-cluster-management.io/v1** です。
policyDefaults.description	任意	作成するポリシーの説明。
policyDefaults.extraDependencies	任意	このポリシーが適用される前に、特定のコンプライアンス状態にある必要があるオブジェクトのリスト。定義した依存関係は、**dependencies** リストとは別に各ポリシーテンプレート (**ConfigurationPolicy** など) に追加されます。**policyDefaults.orderManifests** が **true** に設定されている場合は指定できません。
policyDefaults.extraDependencies[].name	必須	依存しているオブジェクトの名前。
policyDefaults.extraDependencies[].namespace	任意	依存しているオブジェクトの namespace。デフォルトでは、値はポリシージェネレーターに設定されたポリシーの namespace 間に設定されます。
policyDefaults.extraDependencies[].compliance	任意	オブジェクトが必要とするコンプライアンス状態。デフォルト値は **Compliant** です。
policyDefaults.extraDependencies[].kind	任意	オブジェクトの種類。デフォルト値は **Policy** ですが、**ConfigurationPolicy** など、コンプライアンス状態を持つ他の種類にすることもできます。
policyDefaults.extraDependencies[].apiVersion	任意	オブジェクトの API バージョン。デフォルト値は **policy.open-cluster-management.io/v1** です。
policyDefaults.ignorePending	任意	ポリシージェネレーターがその依存関係が目的の状態に達するのを待機しているときに、コンプライアンスステータスチェックをバイパスします。デフォルト値は **false** です。

フィールド	任意または必須	説明
policyDefaults.orderPolicies	任意	ポリシーの **dependencies** を自動的に生成して、ポリシーリストで定義した順序で適用されるようにします。デフォルトでは、値は **false** に設定されています。**policyDefaults.dependencies** と同時に指定することはできません。
policyDefaults.orderManifests	任意	ポリシーテンプレートに **extraDependencies** を自動的に生成して、そのポリシーのマニフェストリストで定義した順序で適用されるようにします。**policyDefaults.consolidateManifests** が **true** に設定されている場合は指定できません。**policyDefaults.extraDependencies** と同時に指定することはできません。
policyDefaults.consolidateManifests	任意	これは、ポリシーでラップされるすべてのマニフェストに対して設定ポリシーを 1 つ生成するかどうかを決定します。**false** に設定すると、マニフェストごとの設定ポリシーが生成されます。デフォルト値は **true** です。
policyDefaults.informGatekeeperPolicies (非推奨)	任意	Gatekeeper マニフェストを設定ポリシーで定義せずに直接使用するには、**informGatekeeperPolicies** を false に設定します。ポリシーが違反した Gatekeeper ポリシーマニフェストを参照している場合、Red Hat Advanced Cluster Management でポリシー違反を受け取るために追加の設定ポリシーが生成されます。デフォルト値は **true** です。
policyDefaults.informKyvernoPolicies	任意	このポリシーが Kyverno ポリシーマニフェストを参照すると、Kyverno ポリシーの違反時に Red Hat Advanced Cluster Management でポリシー違反を受け取るために、設定ポリシーを追加で生成するかどうかが決定されます。デフォルト値は **true** です。

フィールド	任意または必須	説明
policyDefaults.policyLabels	任意	ポリシーの **metadata.labels** セクションに含めるラベル。ポリシーで指定されていない限り、**policyLabels** パラメーターはすべてのポリシーに適用されます。
policyDefaults.policySets	任意	ポリシーが参加するポリシーセットの配列。ポリシーセットの詳細は、**policySets** セクションで定義できます。ポリシーがポリシーセットの一部である場合、配置バインディングはそのセットに対して生成されるため、ポリシーに対しては生成されません。**policies[].generatePlacementWhenInSet** または **policyDefaults.generatePlacementWhenInSet** を設定して、**policyDefaults.policySets** をオーバーライドします。
policyDefaults.generatePolicyPlacement	任意	ポリシーの配置マニフェストを生成します。デフォルトでは **true** に設定されます。**false** に設定すると、placement が指定されている場合でも、プレースメントマニフェストの生成はスキップされます。
policyDefaults.generatePlacementWhenInSet	任意	ポリシーがポリシーセットの一部である場合、デフォルトでは、ポリシーセットの配置が生成されるため、ジェネレーターはこのポリシーの配置を生成しません。ポリシーの配置とポリシーセットの配置の両方でポリシーをデプロイするには、**generatePlacementWhenInSet** を **true** に設定します。デフォルト値は **false** です。
policyDefaults.placement	任意	ポリシーの配置設定。このデフォルトは、すべてのクラスターに一致する配置設定になります。
policyDefaults.placement.name	任意	同じクラスターラベルセレクターを含む配置を統合するための名前を指定します。

フィールド	任意または必須	説明
policyDefaults.placement.labelSelector	任意	**key:value** を使用してクラスターラベルセレクターを定義するか、**matchExpressions**、**matchLabels**、またはその両方に適切な値を指定して、配置を指定します。既存のファイルを指定するには、**placementPath** を参照してください。
policyDefaults.placement.placementName	任意	クラスターにすでに存在する配置を使用するには、このパラメーターを定義します。**Placement** は作成されませんが、**PlacementBinding** はポリシーをこの **Placement** にバインドします。
policyDefaults.placement.placementPath	任意	既存の配置を再利用するには、**kustomization.yaml** ファイルの場所を基準とした相対パスを指定します。指定した場合には、デフォルトですべてのポリシーがこの配置を使用します。新しい **Placement** を生成するには、**labelSelector** 参照してください。
policyDefaults.placement.clusterSelector (非推奨)	任意	**PlacementRule** は非推奨になりました。代わりに **labelSelector** を使用して placement を生成します。**key:value** を使用してクラスターセレクターを定義するか、**matchExpressions**、**matchLabels**、またはその両方を適切な値で指定することにより、配置ルールを指定します。既存のファイルを指定する場合は、**placementRulePath** を参照してください。

フィールド	任意または必須	説明
policyDefaults.placement.placementRuleName (非推奨)	任意	**PlacementRule** は非推奨になりました。あるいは、**placementName** を使用して placement を指定します。クラスター上で既存の配置ルールを使用するには、このパラメーターの名前を指定します。**PlacementRule** は作成されませんが、**PlacementBinding** によってポリシーが既存の **PlacementRule** にバインドされます。
policyDefaults.placement.placementRulePath (非推奨)	任意	**PlacementRule** は非推奨になりました。あるいは、**placementPath** を使用して配置を指定します。既存の配置ルールを再利用するには、**kustomization.yaml** ファイルの場所に対する相対パスを指定します。指定した場合には、デフォルトですべてのポリシーがこの配置ルールを使用します。新しい **PlacementRule** を生成するには、**clusterSelector** を参照してください。
policySetDefaults	任意	ポリシーセットのデフォルト値。このパラメーターにリストされているデフォルト値は、**policySets** 配列のエントリーによってオーバーライドされます。
policySetDefaults.placement	任意	ポリシーの配置設定。このデフォルトは、すべてのクラスターに一致する配置設定になります。このフィールドの説明は、**policyDefaults.placement** を参照してください。
policySetDefaults.generatePolicySetPlacement	任意	ポリシーセットの配置マニフェストを生成します。デフォルトでは **true** に設定されます。**false** に設定すると、配置が指定されている場合でも、配置マニフェストの生成はスキップされます。

フィールド	任意または必須	説明
policies	必須	デフォルト値または **policyDefaults** で設定される値のいずれかを上書きする値と合わせて作成するポリシーのリスト。追加のフィールドと説明は、**policyDefaults** を参照してください。
policies.description	任意	作成するポリシーの説明。
policies[].name	必須	作成するポリシーの名前。
policies[].manifests	必須	デフォルト値、この **policies** 項目に設定された値、または **policyDefaults** に設定された値のいずれかへのオーバーライドとともに、ポリシーに含める Kubernetes オブジェクトマニフェストのリスト。追加のフィールドと説明については、**policyDefaults** を参照してくださ い。**consolidateManifests** が **true** に設定されている場合、**policies[].manifests** レベルでオーバーライドできるのは、**complianceType**、**metadataComplianceType**、および **recordDiff** のみです。
policies[].manifests[].path	必須	単一のファイル、ファイルのフラットディレクトリー、または **kustomization.yaml** ファイルに関連する Kustomize ディレクトリーへのパス。ディレクトリーが Kustomize ディレクトリーの場合、ジェネレーターは、ポリシーを生成する前にディレクトリーに対して Kustomize を実行します。Kustomize ディレクトリーの Helm チャートを処理する必要がある場合は、ポリシージェネレーターが実行されている環境で **POLICY_GEN_ENABLE_HELM** を **"true"** に設定して、ポリシージェネレーターの Helm を有効にします。

フィールド	任意または必須	説明
policies[].manifests[].patches	任意	パスのマニフェストに適用する Kustomize パッチのリスト。複数のマニフェストがある場合は、Kustomize がパッチの適用先のマニフェストを特定できるように、パッチに **apiVersion**、**kind**、**metadata.name**、および **metadata.namespace** (該当する場合) フィールドを設定する必要があります。マニフェストが 1 つの場合には、**metadata.name** および **metadata.namespace** フィールドにパッチを適用できます。
policies.policyLabels	任意	ポリシーの **metadata.labels** セクションに含めるラベル。ポリシーで指定されていない限り、**policyLabels** パラメーターはすべてのポリシーに適用されます。
policySets	任意	作成するポリシーセットのリストと、デフォルト値または **policySetDefaults** に設定されている値のいずれかへのオーバーライド。ポリシーセットにポリシーを含めるには、**policyDefaults.policySets**、**policies[].policySets**、または **policySets.policies** を使用します。追加のフィールドと説明は、**policySetDefaults** を参照してください。
policySets[].name	必須	作成するポリシーセットの名前です。
policySets[].description	任意	作成するポリシーセットの説明です。
policySets[].policies	任意	ポリシーセットに含まれるポリシーのリストです。**policyDefaults.policySets** または **policies[].policySets** も指定されている場合、リストはマージされます。

4.1.4. 関連情報

- GitOps Operator をインストールするためのポリシーの生成 を参照してください。
- 詳細は、ポリシーセットコントローラー を参照してください。
- 詳細は カスタマイズの適用 を参照してください。
- その他のトピックは、ガバナンスに関する ドキュメントを参照してください。
- **kustomization.yaml** ファイルの例を参照してください。
- Kubernetes のラベルとセレクター のドキュメントを参照してください。
- 詳細は、gatekeeper を参照してください。
- Kustomize ドキュメント を参照してください。
- サードパーティーポリシーコントローラーの統合 ドキュメントに戻ります。

4.2. COMPLIANCE OPERATOR をインストールするポリシーの生成

クラスターに Compliance Operator をインストールするポリシーを生成します。Compliance Operator などの **namespaced** インストールモードを使用する Operator の場合、**OperatorGroup** マニフェストも必要になります。

以下の手順を実行します。

1. **Namespace**、**Subscription**、および **compliance-operator.yaml** という **OperatorGroup** マニフェストを含む YAML ファイルを作成します。以下の例では、これらのマニフェストを **compliance-operator** namespace にインストールします。

```
apiVersion: v1
kind: Namespace
metadata:
  name: openshift-compliance
---
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: compliance-operator
  namespace: openshift-compliance
spec:
  targetNamespaces:
    - openshift-compliance
---
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: compliance-operator
  namespace: openshift-compliance
spec:
  channel: release-0.1
  name: compliance-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
```

2. **PolicyGenerator** 設定ファイルを作成します。すべての OpenShift Container Platform マネージドクラスターに Compliance Operator をインストールする以下の **PolicyGenerator** ポリシーの例を表示します。

```
apiVersion: policy.open-cluster-management.io/v1
kind: PolicyGenerator
metadata:
  name: install-compliance-operator
policyDefaults:
  namespace: policies
  placement:
    labelSelector:
      matchExpressions:
        - key: vendor
          operator: In
          values:
            - "OpenShift"
policies:
  - name: install-compliance-operator
    manifests:
      - path: compliance-operator.yaml
```

3. ポリシージェネレーターを **kustomization.yaml** ファイルに追加します。**generators** セクションの設定は次のようになります。

```
generators:
  - policy-generator-config.yaml
```

その結果、生成されたポリシーは次のファイルのようになります。

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement-install-compliance-operator
  namespace: policies
spec:
  predicates:
  - requiredClusterSelector:
      labelSelector:
        matchExpressions:
        - key: vendor
          operator: In
          values:
          - OpenShift
---
apiVersion: policy.open-cluster-management.io/v1
kind: PlacementBinding
metadata:
  name: binding-install-compliance-operator
  namespace: policies
placementRef:
  apiGroup: cluster.open-cluster-management.io
  kind: Placement
  name: placement-install-compliance-operator
subjects:
```

```
  - apiGroup: policy.open-cluster-management.io
    kind: Policy
    name: install-compliance-operator
---
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  annotations:
    policy.open-cluster-management.io/categories: CM Configuration Management
    policy.open-cluster-management.io/controls: CM-2 Baseline Configuration
    policy.open-cluster-management.io/standards: NIST SP 800-53
    policy.open-cluster-management.io/description:
  name: install-compliance-operator
  namespace: policies
spec:
  disabled: false
  policy-templates:
    - objectDefinition:
        apiVersion: policy.open-cluster-management.io/v1
        kind: ConfigurationPolicy
        metadata:
          name: install-compliance-operator
        spec:
          object-templates:
            - complianceType: musthave
              objectDefinition:
                apiVersion: v1
                kind: Namespace
                metadata:
                  name: openshift-compliance
            - complianceType: musthave
              objectDefinition:
                apiVersion: operators.coreos.com/v1alpha1
                kind: Subscription
                metadata:
                  name: compliance-operator
                  namespace: openshift-compliance
                spec:
                  channel: release-0.1
                  name: compliance-operator
                  source: redhat-operators
                  sourceNamespace: openshift-marketplace
            - complianceType: musthave
              objectDefinition:
                apiVersion: operators.coreos.com/v1
                kind: OperatorGroup
                metadata:
                  name: compliance-operator
                  namespace: openshift-compliance
                spec:
                  targetNamespaces:
                    - compliance-operator
          remediationAction: enforce
          severity: low
```

その結果、生成されたポリシーが表示されます。