



Red Hat Advanced Cluster Management for Kubernetes 2.7

ガバナンス

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。

Red Hat Advanced Cluster Management for Kubernetes 2.7 ガバナンス

ポリシーを使用してクラスターのセキュリティ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。

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のセキュリティ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 リスクおよびコンプライアンス	3
1.1. 証明書	3
第2章 ガバナンス	9
2.1. ガバナンスアーキテクチャー	9
2.2. ポリシーの概要	11
2.3. ポリシーコントローラー	17
2.4. ポリシーコントローラーの高度な設定	31
2.5. サポート対象のポリシー	34
2.6. セキュリティポリシーの管理	60
2.7. ポリシーの依存関係	95
2.8. ハブクラスターのセキュリティ保護	97
2.9. サードパーティーポリシーコントローラーの統合	97

第1章 リスクおよびコンプライアンス

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ります。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。

- [ロールベースのアクセス制御](#)
- [認証情報の管理の概要](#)
- [証明書](#)
- [ガバナンス](#)

1.1. 証明書

さまざまな証明書が Red Hat Advanced Cluster Management for Kubernetes で作成され、使用されます。

独自に証明書を使用できます。証明書の Kubernetes TLS Secret を作成する必要があります。独自の証明書の作成後に、Red Hat Advanced Cluster Management インストーラーで作成した特定の証明書を置き換えることができます。

必要なアクセス権限: クラスターの管理者

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。証明書は以下のコンポーネントで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

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 ハブクラスターの証明書

- [OpenShift デフォルトの Ingress 証明書の置き換え](#)
- [可観測性の証明書](#)
 - [独自 \(BYO: Bring Your Own\) の可観測性認証局 \(CA\) 証明書の追加](#)
 - [CA 証明書を生成する OpenSSL コマンド](#)
 - [独自の CA 証明書に関連付けられたシークレットの作成](#)
 - [alertmanager ルートの証明書の置き換え](#)

Red Hat Advanced Cluster Management コンポーネントの証明書

- [ハブクラスターのマネージド証明書のリスト表示](#)
- [ハブクラスターの管理対象証明書の更新](#)
- [Red Hat Advanced Cluster Management Webhook 証明書の更新](#)

Red Hat Advanced Cluster Management の管理対象証明書

- [チャンネル証明書](#)
- [マネージドクラスター証明書](#)

サードパーティー証明書

- [gatekeeper Webhook 証明書のローテーション](#)

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。

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

1.1.1.1. 可観測性の証明書

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て、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間のトラフィックで相互 TLS を提供します。Kubernetes Secret が可観測性証明書に関連付けられます。

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間のトラフィックは中断されません。

1.1.1.2. 独自 (BYO: Bring Your Own) の可観測性認証局 (CA) 証明書の追加

Red Hat Advanced Cluster Management で生成されたデフォルトの可観測性 CA 証明書を使用しない場合は、可観測性を有効にする前に、独自の可観測性 CA 証明書を使用できます。

1.1.1.2.1. CA 証明書を生成する OpenSSL コマンド

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
```

1.1.1.2.2. 独自の 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
```

1.1.1.2.3. 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
```

1.1.2. Red Hat Advanced Cluster Management コンポーネントの証明書

1.1.2.1. ハブクラスタのマネージド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のマネージド証明書のリストを表示できます。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
```

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。

1.1.2.2. ハブクラスタの管理対象証明書の更新

[List hub cluster managed certificates](#) セクションで **delete secret** コマンドを実行して、ハブクラスタのマネージド証明書を更新できます。更新する必要がある証明書を特定したら、その証明書に関連するシークレットを削除します。たとえば、以下のコマンドを実行してシークレットを削除できます。

```
oc delete secret grc-0c925-grc-secrets -n open-cluster-management
```

注記: シークレットの削除すると、新規のシークレットが作成されます。ただし、新しい証明書の使用を開始するには、そのシークレットを使用する Pod を手動で再起動する必要があります。

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

Red Hat Advanced Cluster Management Webhook で使用される証明書である OpenShift Container Platform のマネージド証明書を更新できます。

Red Hat Advanced Cluster Management Webhook 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、OpenShift Container Platform の管理対象証明書に関連付けられているシークレットを削除します。

```
oc delete secret -n open-cluster-management ocm-webhook-secret
```

注記: サービスによっては、削除する必要があるシークレットが存在しない場合があります。

- 以下のコマンドを実行して、OpenShift Container Platform の管理対象証明書に関連付けられているサービスを再起動します。

```
oc delete po -n open-cluster-management ocm-webhook-679444669c-5cg76
```

重要: 多数のサービスのレプリカが存在するため、各サービスを再起動する必要があります。

証明書を含む Pod の概要を一覧にまとめた以下の表を参照して、Pod の再起動前にシークレットを削除する必要があるかどうかを確認します。

表1.1 OpenShift Container Platform マネージド証明書を含む Pod

サービス名	Namespace	サンプルの Pod 名	シークレット名 (該当する場合)
channels-apps-open-cluster-management-webhook-svc	open-cluster-management	multicluster-operators-application-8c446664c-5lbfk	channels-apps-open-cluster-management-webhook-svc-ca
multicluster-operators-application-svc	open-cluster-management	multicluster-operators-application-8c446664c-5lbfk	multicluster-operators-application-svc-ca
cluster-manager-registration-webhook	open-cluster-management-hub	cluster-manager-registration-webhook-fb7b99c-d8wfc	registration-webhook-serving-cert
cluster-manager-work-webhook	open-cluster-management-hub	cluster-manager-work-webhook-89b8d7fc-f4pv8	work-webhook-serving-cert

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

1.1.3.1. チャネル証明書

CA 証明書は、Red Hat Advanced ClusterManagement アプリケーション管理の一部である Git チャネルに関連付けることができます。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。

Helm チャネルを使用すると、証明書の検証が無効にできます。Helm チャネルで、証明書の検証が無効になっている場合は、Helm チャネルを開発環境で設定する必要があります。証明書の検証が無効になると、セキュリティリスクが発生します。

1.1.3.2. マネージドクラスター証明書

証明書は、ハブでマネージドクラスターを認証するのに使用されます。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。詳細は、[証明書を変更した後のインポート済みクラスターのオフラインでのトラブルシューティング](#) を参照してください。

マネージドクラスター証明書は自動的に更新されます。

1.1.4. サードパーティー証明書

1.1.4.1. 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 po -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。

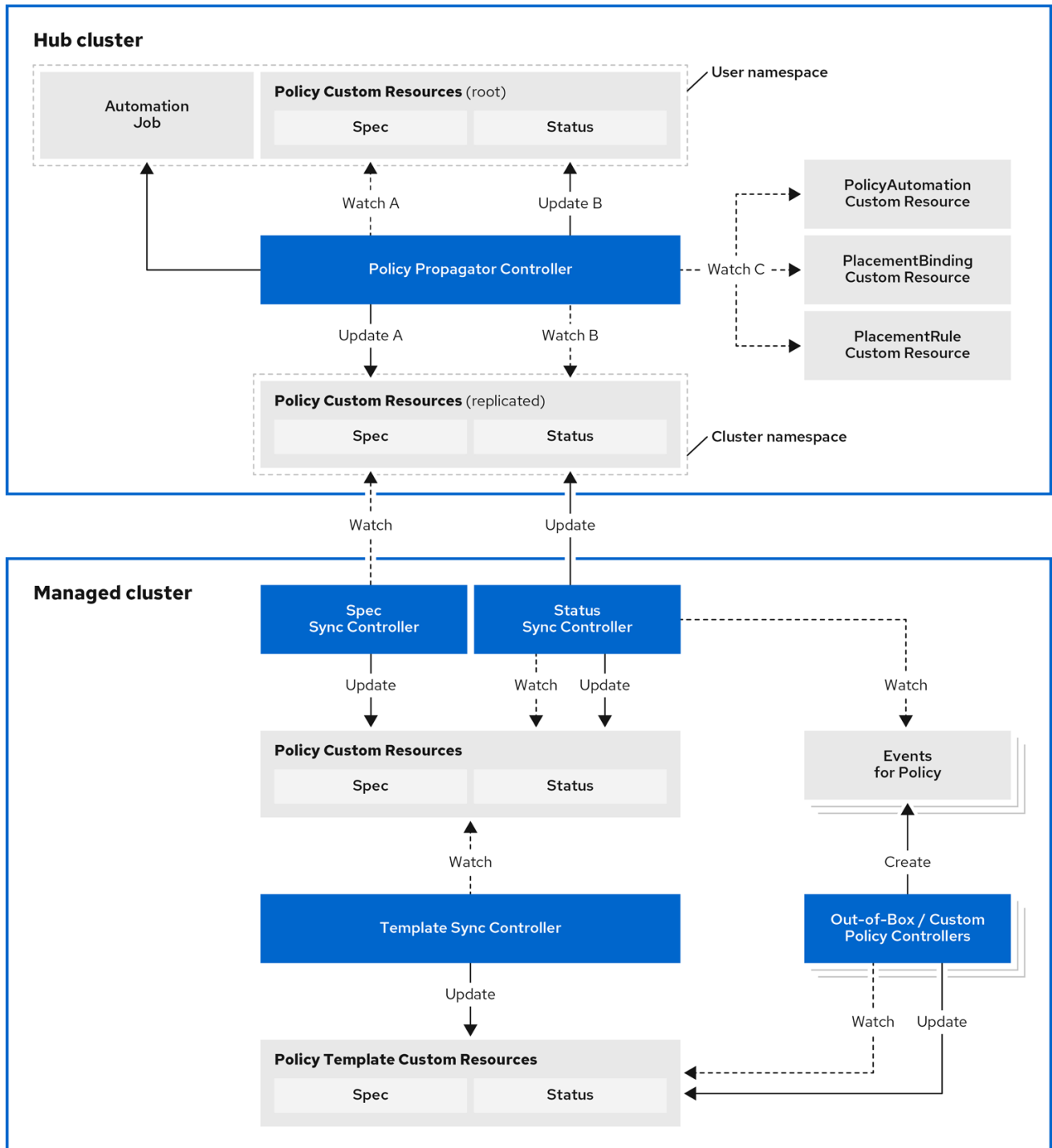
証明書ポリシーコントローラーを使用して、マネージドクラスターで証明書ポリシーを作成して管理します。コントローラーの詳細は、[ポリシーコントローラー](#) を参照してください。詳細は、[リスクおよびコンプライアンス](#) ページに戻り、確認してください。

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、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。

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を強化します。製品ガバナンスのライフサイクルは、定義されたポリシー、プロセス、および手順に基づいて、中央のインターフェイスページからセキュリティ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。



186_RHACM_I221

ガバナンスアーキテクチャは、以下のコンポーネントで設定されています。

- **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、**セキュリティーポリシーの管理** 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。
- **ポリシーコントローラー**: 指定した制御に対してマネージドクラスター上のポリシーを1つ以上評価し、違反の Kubernetes イベントを生成します。違反は、ハブクラスターに伝播されず。インストールに含まれるポリシーコントローラーは、Kubernetes 設定、証明書、および IAM です。詳細設定を使用してポリシーコントローラーをカスタマイズします。
- **オープンソースコミュニティ**: 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の貢献をサポートします。ポリシーコントローラーおよびサードパーティーのポリシーも、[stolostron/policy-collection リポジトリ](#)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。詳細については、[セキュリティポリシーの管理](#) セクションの [GitOps を使用したポリシーのデプロイ](#) を参照してください。Red Hat Advanced Cluster Management for Kubernetes とサードパーティーのポリシーの統合方法を説明します。

関連トピックを引き続きお読みください。

- [ポリシーの概要](#)
- [ポリシーコントローラー](#)
- [ポリシーコントローラーの高度な設定](#)
- [サポート対象のポリシー](#)
- [ポリシーの依存関係](#)
- [セキュリティポリシーの管理](#)
- [ハブクラスターのセキュリティ保護](#)
- [サードパーティーポリシーコントローラーの統合](#)

2.2. ポリシーの概要

Red Hat Advanced Cluster Management for Kubernetes セキュリティポリシーフレームワークを使用して、ポリシーを作成および管理します。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。

各 Red Hat Advanced Cluster Management ポリシーには、少なくとも1つ以上のテンプレートを含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。

このポリシーには、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 **PlacementRule** または **Placement** と、Red Hat Advanced Cluster Management for Kubernetes ポリシーを配置ルールにバインドする **PlacementBinding** が必要です。**PlacementRule** の定義方法に関する詳細は、アプリケーションライフサイクルドキュメントの [配置ルール](#) を参照してください。**Placement** 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 [配置の概要](#) を参照してください。

重要:

- 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 **PlacementRule** または **Placement** にバインドする必要があります。
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。
- 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。すべてのマネージドクラスターには、マネージドクラスターの名前と一致する namespace がハブクラスター上にあります。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 for Kubernetes によって削除されます。代わりに、**policy** などの namespace を使用してください。
- 各クライアントおよびプロバイダーは、マネージドのクラウド環境で、Kubernetes クラスターでホストされているワークロードの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、規制準拠に関する内部エンタープライズセキュリティ基準を満たしていることを確認します。ガバナンスおよびセキュリティ機能を使用して、標準を満たすように可視性を確保し、設定を調整します。

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](#)
- [Placement YAML のサンプルファイル](#)

2.2.1. 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。前述のパラメーターフィールドの YAML 設定は、以下を確認してください。

```
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:
spec:
  dependencies:
  - apiVersion: policy.open-cluster-management.io/v1
    compliance:
      kind: Policy
      name:
      namespace:
  policy-templates:
  - objectDefinition:
      apiVersion:
      kind:
      metadata:
```



```

    name:
    spec:
    remediationAction:
    disabled:
---
apiVersion: apps.open-cluster-management.io/v1
kind: PlacementBinding
metadata:
  name:
placementRef:
  name:
  kind:
  apiGroup:
subjects:
- name:
  kind:
  apiGroup:
---
apiVersion: apps.open-cluster-management.io/v1
kind: PlacementRule
metadata:
  name:
spec:
  clusterConditions:
  - type:
  clusterLabels:
  matchLabels:
  cloud:

```

2.2.2. ポリシー YAML の表

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。

フィールド	任意または必須	説明
metadata.annotations	任意	<p>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。</p> <p>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。</p>
annotations.policy.open-cluster-management.io/standards	任意	<p>ポリシーが関連するセキュリティ標準の名前。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。</p>
annotations.policy.open-cluster-management.io/categories	任意	<p>セキュリティ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。</p>
annotations.policy.open-cluster-management.io/controls	任意	<p>チェックされるセキュリティ制御の名前。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などです。</p>
spec.dependencies	任意	<p>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。</p>
spec.policy-templates	必須	<p>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。</p>

フィールド	任意または必須	説明
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。
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	オプション。	<p>ポリシーの修正を指定します。パラメータの値は 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。</p> <p>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。</p>

2.2.3. ポリシーサンプルファイル

```

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
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: PlacementRule
  apiGroup: apps.open-cluster-management.io
subjects:
  - name: policy-role
    kind: Policy
    apiGroup: policy.open-cluster-management.io
    ---
apiVersion: apps.open-cluster-management.io/v1
kind: PlacementRule
metadata:
  name: placement-policy-role
spec:
  clusterConditions:
    - status: "True"
      type: ManagedClusterConditionAvailable
  clusterSelector:
    matchExpressions:
      - {key: environment, operator: In, values: ["dev"]}

```

2.2.4. Placement YAML のサンプルファイル

PlacementBinding および **Placement** リソースは、以前のポリシー例と組み合わせ、**PlacementRule** API ではなく、クラスターの **Placement** API を使用してポリシーをデプロイできます。

-

```

---
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
---
//Depends on if governance would like to use v1beta1
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"]}

```

- [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

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。既製の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 (CRD) 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モニターし、コントローラーが強制機能をサポートしている場合、クラスターの状態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](#)
- [ポリシーセットコントローラー](#)

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。

ポリシー管理の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。

2.3.1. 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。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。

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 **enforce** 機能をサポートしています。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。**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、[設定ポリシーでのテンプレートのサポート](#) 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、[ポリシージェネレーター](#) はこれを実現するための便利なツールです。

設定ポリシーコントローラーについては、以下を参照してください。

- [設定ポリシーの例](#)
- [設定ポリシーの YAML の表](#)

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
object-templates-raw: |
- 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。
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 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。

フィールド	任意または必須	説明
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: inform 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	<p>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。</p> <p>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。</p>
spec.evaluationInterval.noncompliant	任意	<p>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。</p>

フィールド	任意または必須	説明
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うち少なくとも1つは設定する必要があります。
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。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うち少なくとも1つは設定する必要があります。
spec.object-templates-raw.spec	任意	オブジェクトテンプレートの条件を指定するために使用されます。If else ステートメントはサポートされている値です。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。 <pre> {{- if eq .metadata.name "policy-grc-your-meta-data-name" }} replicas: 2 {{- else }} replicas: 1 {{- end }} </pre>

フィールド	任意または必須	説明
spec.object-templates[].complianceType	必須	<p>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値として使用する必要があります。</p> <p>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。</p> <p>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。</p> <p>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。</p>
spec.object-templates[].metadataComplianceType	任意	<p>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。</p>

フィールド	任意または必須	説明
<code>spec.object-templates[].objectDefinition</code>	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)。
<code>spec.pruneObjectBehavior</code>	任意	管理対象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](#) を参照してください。
- 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、[セキュリティーポリシーの管理](#) を参照してください。
- [設定ポリシーでのテンプレートのサポート](#) 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 名が含まれている証明書の検出に使用できます。証明書ポリシーは、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。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。
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 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。

フィールド	任意または必須	説明
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 。
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。

フィールド	任意または必須	説明
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。
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。

証明書ポリシーの管理方法の詳細は、[セキュリティポリシーの管理](#) を参照してください。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。

2.3.3. IAM ポリシーコントローラー

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 。

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。 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。
spec.ignoreClusterRoleBindings	任意	無視するクラスタ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 regex 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 <code>^</code> 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、選択したクラスタに配置されます。ポリシーセットがハブクラスタ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の違反のみを確認します。

注意: Red Hat Advanced Cluster Management の強化サンプルポリシーセットは、クラスタ配置を使用します。クラスタ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セットにバインドします。クラスタ配置の使用の詳細については、[クラスタへのポリシーのデプロイ](#) 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: apps.open-cluster-management.io
  kind: PlacementRule
  name: demo-policyset-pr
subjects:
- apiGroup: policy.open-cluster-management.io
  kind: PolicySet
  name: demo-policyset

---
apiVersion: apps.open-cluster-management.io
kind: PlacementRule
metadata:
  name: demo-policyset-pr
spec:
  clusterConditions:pagewidth:
  - status: "True"
    type: ManagedClusterConditionAvailable
  clusterSelectors:
    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
  placementRule: placement1
  policySet: policyset-ps

```

[セキュリティポリシーの管理](#) 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。[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。

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 **ManagedClusterAddOns** は、ポリシーフレームワーク、**governance-policy-framework**、**config-policy-controller**、**cert-policy-controller**、および **iam-policy-controller** を設定します。

必要なアクセス権限: クラスターの管理者

2.4.1. 同時実行の設定

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、引用符で囲まれたゼロ以外の整数で **policy-evaluation-concurrency** アノテーションを設定します。ハブクラスターのマネージドクラスター namespace にある **config-policy-controller** という名前の **ManagedClusterAddOn** オブジェクトに値を設定できます。

注記: 並行処理性の値を高くすると、**config-policy-controller** Pod、Kubernetes API サーバー、および OpenShift API サーバーでの CPU とメモリーの使用率が増加します。

次の YAML の例では、**cluster1** というマネージドクラスターで並行処理性が **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
```

2.4.2. デバッグログの設定

ポリシーコントローラーごとにデバッグログを設定します。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
```

2.4.3. 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。

2.4.3.1. メトリック: **policy_governance_info**

policy_governance_info は OpenShift Container Platform モニタリングが、一部の集計データは Red Hat Advanced Cluster Management の可観測性が収集します (有効にされている場合)。

注記: 可観測性が有効になっている場合は、Grafana の **Explore** ページからメトリックのクエリーを入力できます。

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、**root** ポリシーと **PlacementRules** および **PlacementBindings** を監視して、**伝播** ポリシーを作成する場所を決定し、ポリシーをマネージドクラスターに分散します。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: メトリックが必要ではなく、パフォーマンスやセキュリティに関する懸念がある場合は、この機能を無効にすることができます。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、[カスタムメトリックスの追加](#) を参照してください。

2.4.3.2. 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ない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、次の可能なエンタ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。

2.4.4. 設定変更の確認

新しい設定がコントローラーによって適用されると、**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を使用して、設定が正しく検証されます。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
```

その他のトピックについては、[ガバナンス](#) ページに戻ります。

2.5. 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。

2.5.1. 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 セキュリティポリシー (非推奨)	一貫したワークロードセキュリティを確保します。Kubernetes Pod セキュリティポリシーのドキュメント を参照してください。
Role policy Role binding policy	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。

ポリシーのサンプル	説明
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 Security Context Constraints のドキュメントを参照してください。
ETCD 暗号化ポリシー	etcd 暗号化でデータセキュリティを確保します。Openshift etcd 暗号化のドキュメント を参照してください。
コンプライアンス 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 Compliance Operator のドキュメント 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 セキュリティプロファイルへの準拠を確認します。Openshift Compliance Operator のドキュメント 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 セキュリティプロファイルへの準拠を確認します。Openshift Compliance Operator 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	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 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。

2.5.2. 追加設定なしに使用可能なポリシーのサポートマトリックス

表2.7 サポート表

Policy	Red Hat OpenShift Container Platform 3.11	Red Hat OpenShift Container Platform 4
メモリー使用状況のポリシー	x	x
Namespace ポリシー	x	x
イメージ脆弱性ポリシー	x	x
Pod ポリシー	x	x
Pod セキュリティーポリシー (非推奨)		
ロールポリシー	x	x
Role binding ポリシー	x	x
SCC (Security Context Constraints) ポリシー	x	x
ETCD 暗号化ポリシー		x
gatekeeper ポリシー		x
コンプライアンス Operator ポリシー		x
E8 スキャンポリシー		x
OpenShift CIS スキャンポリシー		x
ポリシーセット		x

以下のポリシーサンプルを参照し、特定のポリシーの適用方法を確認します。

- [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](#)
- [コンプライアンス 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](#)

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。

2.5.3. 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タ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](#)

2.5.3.1. 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:
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:
...

```

2.5.3.2. メモリー使用状況のポリシーの表

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。

2.5.3.3. 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。

2.5.4. 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](#)

2.5.4.1. 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:
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:
    ...

```

2.5.4.2. 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。
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。

2.5.4.3. 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。

2.5.5. 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、コンテナセキュリティ Operator を利用してコンテナ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、コンテナセキュリティ 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 Operator の詳細は、[Quay リポジトリ](#) の [コンテナセキュリティ 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](#)

2.5.5.1. イメージ脆弱性ポリシーの YAML 設定

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

- 名前とチャン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

```

2.5.5.2. 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。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。

2.5.6. 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

2.5.6.1. 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:
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:
            ...

```

2.5.6.2. Pod 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
-------	---------	----

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。

2.5.6.3. 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。

2.5.7. 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 のセキュリティポリシーを適用して Pod およびコンテナのセキュリティを保護します。

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](#)

2.5.7.1. 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:
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:
...

```

2.5.7.2. Pod セキュリティーポリシーの表

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
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。

2.5.7.3. 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 4.12 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モニター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。

2.5.8. 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](#)

2.5.8.1. 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:
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:
  ...
---
apiVersion: policy.open-cluster-management.io/v1
kind: PlacementBinding
metadata:
  name: binding-policy-role
  namespace:
placementRef:
  name: placement-policy-role
  kind: PlacementRule
  apiGroup: apps.open-cluster-management.io
subjects:
  - name: policy-role
    kind: Policy
    apiGroup: policy.open-cluster-management.io
---
apiVersion: apps.open-cluster-management.io/v1
kind: PlacementRule
metadata:
  name: placement-policy-role
  namespace:
spec:
  clusterConditions:
    - type: ManagedClusterConditionAvailable
      status: "True"
  clusterSelector:
    matchExpressions:
      []
  ...

```

2.5.8.2. ロールポリシーの表

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。

フィールド	任意または必須	説明
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。

2.5.8.3. 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](#) ページを参照してください。

2.5.9. 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](#)

2.5.9.1. 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:
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:
    ...

```

2.5.9.2. 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。

フィールド	任意または必須	説明
<code>metadata.namespace</code>	必須	ポリシーの namespace。
<code>spec.remediationAction</code>	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。
<code>spec.disabled</code>	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。
<code>spec.policy-templates[].objectDefinition</code>	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。

2.5.9.3. 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。

2.5.10. 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](#)

2.5.10.1. 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:
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:
        ...

```

2.5.10.2. 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。

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 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。

2.5.10.3. 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。

2.5.11. 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](#)

2.5.11.1. 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:
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:
            ...
```

2.5.11.2. ETCD 暗号化ポリシーの表

表2.12 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 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。

2.5.11.3. 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。

2.5.12. コンプライアンス Operator ポリシー

コンプライアンス Operator は、OpenSCAP を実行する Operator で、Red Hat OpenShift Container Platform クラスターを必要なセキュリティーベンチマークに常に準拠させることができます。コンプライアンス Operator ポリシーを使用して、マネージドクラスターにコンプライアンス Operator をインストールできます。

コンプライアンス 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。コンプライアンス Operator ポリシーは、OpenShift Container Platform 4.6 および 4.7 でサポートされます。詳細は、OpenShift Container Platform ドキュメントの [コンプライアンス Operator について](#) を参照してください。

注記: [コンプライアンス 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 コンプライアンス Operator に依存します。コンプライアンス Operator の詳細は、OpenShift Container Platform ドキュメントの [コンプライアンス Operator について](#) を参照してください。

2.5.12.1. コンプライアンス Operator のリソース

コンプライアンス 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 インストール用のコンプライアンス 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
```

- 名前とチャン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としてプルします。

```
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: "4.7"
  installPlanApproval: Automatic
  name: compliance-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace

```

コンプライアンス 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。

コンプライアンス 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。

コンプライアンス 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。

2.5.13. 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、コンプライアンス 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。E8 スキャンポリシーは OpenShift Container Platform 4.6 および 4.7 でサポートされます。詳細は、[OpenShift Container Platform ドキュメント](#) の [コンプライアンス Operator について](#) を参照してください。

2.5.13.1. 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。

2.5.14. 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、コンプライアンス 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 CIS スキャンポリシーは OpenShift Container Platform 4.6 および 4.7、4.9 でサポートされます。詳細は、OpenShift Container Platform ドキュメントの [コンプライアンス Operator について](#) を参照してください。

2.5.14.1. 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、[セキュリティポリシーの管理](#) を参照してください。

2.6. セキュリティポリシーの管理

セキュリティポリシーおよびポリシー違反の作成、表示、および管理には、**ガバナンス** ダッシュボードを使用します。CLI およびコンソールからポリシーの YAML ファイルを作成できます。

2.6.1. ガバナンスページ

以下のタブが **Governance** ページに表示されます。

- **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を表示します。
- **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**
セキュリティポリシーを作成および管理します。ポリシーの表では、**Name**、**Namespace**、**Status**、**Remediation**、**Policy set**、**Cluster violations**、**Source**、**Automation** および **Created** のポリシーの詳細を表示します。

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。

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。

2.6.2. 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がない場合、**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。

セキュリティ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](#)

2.6.3. 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](#)

2.6.3.1. 前提条件

- Red Hat OpenShift Container Platform 4.5 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必要があります。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。

2.6.3.2. 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。

2.6.3.3. 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。

- 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。
- 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。
- 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: policynamespace-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。

2.6.4. GitOps を使用したポリシーのデプロイ

ガバナンスフレームワークを使用して、マネージドクラスター全体にポリシーセットをデプロイできます。リポジトリにポリシーを提供して使用することで、オープンソースコミュニティ (**policy-collection**) に追加できます。詳細は、[カスタムポリシーの取得](#) を参照してください。オープンソースコミュニティの各 **stable** および **community** フォルダーのポリシーは、[NIST Special Publication 800-53](#) に従ってさらに整理されています。

GitOps を使用して Git リポジトリ経由でポリシーの更新や作成を自動化して追跡する時のベストプラクティスを理解するにはこれ以降のセクションを確認してください。

前提条件: 開始する前に、**policy-collection** リポジトリをフォークしてください。

- [ローカルリポジトリのカスタマイズ](#)
- [ローカルリポジトリへのコミット](#)

- クラスターへのポリシーのデプロイ
- コンソールからの GitOps ポリシーデプロイメントの確認
- CLI からの GitOps ポリシーデプロイメントの確認

2.6.4.1. ローカルリポジトリのカスタマイズ

stable および **community** ポリシーを1つのフォルダーにまとめて、ローカルリポジトリをカスタマイズします。使用しないポリシーを削除します。ローカルリポジトリをカスタマイズするには、以下の手順を実行します。

1. リポジトリに新しいディレクトリーを作成し、デプロイするポリシーを保存します。GitOps のメインのデフォルトブランチに、ローカルの **policy-collection** リポジトリにあることを確認します。以下のコマンドを実行します。

```
mkdir my-policies
```

2. **stable** および **community** ポリシーのすべてを **my-policies** ディレクトリーにコピーします。**stable** フォルダーにコミュニティで利用可能なものが重複している場合があるため、**community** ポリシーから始めます。以下のコマンドを実行します。

```
cp -R community/* my-policies/
```

```
cp -R stable/* my-policies/
```

すべてのポリシーの構造は単一の親ディレクトリーとなっているため、フォークでポリシーを編集できます。

ヒント:

- 使用の予定がないポリシーを削除するのがベストプラクティスです。
- 以下のリストでポリシーおよびポリシーの定義について確認してください。
 - 目的: ポリシーのルールを理解する。
 - 修復アクション: ポリシーで、コンプライアンスの通知だけを行うのか、ポリシーを強制して、変更を加えるのか? **spec.remediationAction** パラメーターを参照してください。変更が適用される場合は、想定されている機能を理解するようにしてください。強制のサポートがあるポリシーを確認してください。詳細は、**Validate** セクションを参照してください。
注記: ポリシーに設定された **spec.remediationAction** は、個別の **spec.policy-templates** で設定される修復アクションを上書きします。
 - 配備: ポリシーのデプロイ先のクラスターは? デフォルトでは、ほとんどのポリシーは、**environment: dev** ラベルの付いたクラスターを対象にしています。ポリシーによっては、OpenShift Container Platform クラスターまたは別のラベルをターゲットにできます。追加のラベルを更新または追加して、他のクラスターを組み込むことができます。特定の値がない場合、ポリシーはすべてのクラスターに適用されます。また、ポリシーのコピーを複数作成し、クラスターセットごとに各ポリシーをカスタマイズして、別のクラスターセットには別の方法で設定することができます。

2.6.4.2. ローカルリポジトリへのコミット

ディレクトリーに行った変更の問題がなければ、変更を Git にコミットしてプッシュし、クラスターによるアクセスを可能にします。

注記: この例は、GitOps でポリシーを使用する基本的な方法を示しており、ブランチの変更を取得する場合には別のワークフローを使用する場合があります。

以下の手順を実行します。

1. ターミナルから **git status** を実行して、以前に作成したディレクトリーに最新の変更を確認します。以下のコマンドを使用して、コミットする変更リストに新しいディレクトリーを追加します。

```
git add my-policies/
```

2. 変更をコミットし、メッセージをカスタマイズします。以下のコマンドを実行します。

```
git commit -m "Policies to deploy to the hub cluster"
```

3. GitOps に使用するフォークしたりポジトリのブランチに、変更をプッシュします。以下のコマンドを実行します。

```
git push origin <your_default_branch>master
```

変更がコミットされます。

2.6.4.3. クラスターへのポリシーのデプロイ

変更をプッシュしたら、ポリシーを Red Hat Advanced Cluster Management for Kubernetes インストールにデプロイできます。デプロイメント後、ハブクラスターは Git リポジトリに通知されます。Git リポジトリの選択したブランチに追加された変更がクラスターに反映されます。

注記: デフォルトでは、GitOps でデプロイされるポリシーは **マージ** の調整オプションを使用します。代わりに **replace** 調整オプションを使用する場合は、[apps.open-cluster-management.io/reconcile-option: replace](https://apps.open-cluster-management.io/reconcile-option:replace) アノテーションを **Subscription** リソースに追加します。詳細は、[アプリケーションライフサイクル](#) を参照してください。

deploy.sh スクリプトは、ハブクラスターに **Channel** および **Subscription** リソースを作成します。チャンネルは Git リポジトリに接続し、サブスクリプションは、チャンネルを介してクラスターに配置するデータを指定します。その結果、指定のサブディレクトリーで定義された全ポリシーがハブに作成されます。サブスクリプションによりポリシーが作成されると、Red Hat Advanced Cluster Management はポリシーを分析し、定義した配置ルールに基づいて、ポリシーが適用される各マネージドクラスターに関連付けられた namespace に追加のポリシーリソースを作成します。

その後、ポリシーはハブクラスター上にある該当するマネージドクラスターの namespace からマネージドクラスターにコピーされます。そのため、Git リポジトリのポリシーは、ポリシーの配置ルールで定義される **clusterSelector** に一致するラベルが付いた全マネージドクラスターにプッシュされます。

以下の手順を実行します。

1. **policy-collection** フォルダーから、以下のコマンドを実行してディレクトリーを変更します。

```
cd deploy
```


- 以下のコマンドで、コマンドラインインターフェイス (CLI) が正しいクラスターでリソースを作成するように設定されていることを確認します。

```
oc cluster-info
```

コマンドの出力には、Red Hat Advanced Cluster Management がインストールされているクラスターの API サーバーの詳細が表示されます。正しい URL が表示されない場合は、CLI を正しいクラスターを参照するように設定します。詳細情報は、[OpenShift CLI の使用](#) セクションを参照してください。

- アクセス制御およびポリシー整理を行うポリシーの作成先の namespace を作成します。以下のコマンドを実行します。

```
oc create namespace policy-namespace
```

- 以下のコマンドを実行してクラスターにポリシーをデプロイします。

```
./deploy.sh -u https://github.com/<your-repository>/policy-collection -p my-policies -n policy-namespace
```

your-repository は、Git ユーザー名またはリポジトリ名に置き換えます。

注記: 参考までに、**deploy.sh** スクリプトの引数の全リストでは、以下の構文を使用します。

```
./deploy.sh [-u <url>] [-b <branch>] [-p <path/to/dir>] [-n <namespace>] [-a|--name <resource-name>]
```

引数については、以下のドキュメントを参照してください。

- URL: メインの **policy-collection** リポジトリからフォークしたリポジトリへの URL。デフォルトの URL は <https://github.com/stolostron/policy-collection.git> です。
- ブランチ: 参照する Git リポジトリのブランチ。デフォルトのブランチは **main** です。
- サブディレクトリパス: 使用するポリシーを含めるために作成したサブディレクトリパス。上記のサンプルでは **my-policies** サブディレクトリを使用しましたが、開始するフォルダーを指定することもできます。たとえば、**my-policies/AC-Access-Control** を使用できます。デフォルトのフォルダーは **stable** です。
- Namespace: リソースおよびポリシー作成先のハブクラスター上の namespace。これらの手順では、namespace **policy-namespace** を使用します。デフォルトの namespace は **policies** です。
- 名前のプレ接頭辞: **Channel** および **Subscription** リソースの接頭辞。デフォルトは **demo-stable-policies** です。

deploy.sh スクリプトの実行後に、リポジトリにアクセスできるユーザーはブランチに変更をコミットできます。これにより、クラスターの既存のポリシーに変更がプッシュされます。

注意: サブスクリプションを使用してポリシーをデプロイするには、次の手順を実行します。

- open-cluster-management:subscription-admin** ClusterRole をサブスクリプションを作成するユーザーにバインドします。
- サブスクリプションで許可リストを使用している場合は、次の API エントリーを含めます。

■

```

- apiVersion: policy.open-cluster-management.io/v1
  kinds:
    - "*"
- apiVersion: policy.open-cluster-management.io/v1beta1
  kinds:
    - "*"
- apiVersion: apps.open-cluster-management.io/v1
  kinds:
    - PlacementRule
- apiVersion: cluster.open-cluster-management.io/v1beta1
  kinds:
    - Placement

```

2.6.4.4. コンソールからの GitOps ポリシーデプロイメントの確認

変更がコンソールからポリシーに適用されていることを確認します。コンソールからポリシーをさらに変更することもできますが、**Subscription** と Git リポジトリと調整すると、これらの変更は元に戻されます。以下の手順を実行します。

1. Red Hat Advanced Cluster Management クラスターにログインします。
2. ナビゲーションメニューから **Govern** を選択します。
3. 表にデプロイされたポリシーを見つけます。GitOps を使用してデプロイしたポリシーには、**Source** 列に **Git** ラベルが付いています。ラベルをクリックして、Git リポジトリの詳細を表示します。

2.6.4.4.1. CLI からの GitOps ポリシーデプロイメントの確認

以下の手順を実行します。

1. 以下のポリシーの詳細を確認してください。
 - 配信先のクラスターで特定のポリシーが準拠している/準拠していないのはなぜか？
 - ポリシーが正しいクラスターに適用されているか？
 - このポリシーがクラスターに配布されていない場合は、なぜか？
2. 作成または変更した GitOps のデプロイポリシーを特定します。GitOps のデプロイポリシーは、自動適用されるアノテーションで特定できます。GitOps のデプロイポリシーのアノテーションは、以下のパスのようになります。

```
apps.open-cluster-management.io/hosting-deployable: policies/deploy-stable-policies-Policy-policy-role9
```

```
apps.open-cluster-management.io/hosting-subscription: policies/demo-policies
```

```
apps.open-cluster-management.io/sync-source: subgbk8s-policies/demo-policies
```

GitOps アノテーションは、ポリシーが作成されたサブスクリプションを確認するのに役立ちます。独自のラベルをポリシーに追加して、ラベルに基づいてポリシーを選択するランタイムクエリーを作成することもできます。

たとえば、次のコマンドを使用してポリシーにラベルを追加できます。


```
oc label policies.policy.open-cluster-management.io <policy-name> -n <policy-namespace>
<key>=<value>
```

続いて、以下のコマンドでラベルのあるポリシーをクエリーします。

```
oc get policies.policy.open-cluster-management.io -n <policy-namespace> -l <key>=<value>
```

ポリシーは GitOps を使用してデプロイされます。

2.6.5. 設定ポリシーでのテンプレートのサポート

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。

- [前提条件](#)
- [テンプレート関数](#)
- [設定ポリシーでのハブクラスターテンプレートのサポート](#)
 - [テンプレート処理](#)
 - [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](#)
 - [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](#)

2.6.5.1. 前提条件

- テンプレート構文は 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。

2.6.5.2. 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、[設定ポリシーでのハブクラスターテンプレートのサポート](#) 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** を使用する場合には、検索されるリソースの YAML 構造について理解しておくことが推奨されます。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。これを上書きするには、**toInt** または **toBool** をテンプレート内で最終関数として使用して、値を整数またはブール値として強制的に解釈する追加の処理を開始します。

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
- [オープンソースコミュニティ機能](#)

2.6.5.2.1. 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、[設定ポリシーでのハブクラスターテンプレートのサポート](#) 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

```

2.6.5.2.2. 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、[設定ポリシーでのハブクラスターテンプレートのサポート](#) 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

```

2.6.5.2.3. 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

```

2.6.5.2.4. 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。要求されたり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。オープンソースコミュニティ機能 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) (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、kind、namespace、および name 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、[設定ポリシーでのハブクラスターテンプレートのサポート](#) 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
```

2.6.5.2.5. 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 }}'

```

2.6.5.2.6. 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 }}'

```

2.6.5.2.7. 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
            }}
```

2.6.5.2.8. 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 }}
```

2.6.5.2.9. 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 }}
```

2.6.5.2.10. 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が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 }}
```


2.6.5.2.11. 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/0vxe13vGa+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。

2.6.5.2.12. 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"]
```

2.6.5.2.13. オープンソースコミュニティ機能

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。

- `cat`
- `contains`
- `default`
- `空`
- `fromJson`
- `hasPrefix`
- `hasSuffix`
- `join`
- `list`
- `lower`
- `mustFromJson`
- `quote`
- `replace`
- `semver`
- `semverCompare`
- `split`
- `splitn`
- `ternary`
- `trim`
- `until`
- `untilStep`
- `upper`

詳細は、[Sprig 関数のドキュメント](#) を参照してください。

2.6.5.3. 設定ポリシーでのハブクラスターテンプレートのサポート

Red Hat Advanced Cluster Management は、ターゲットクラスターに動的にカスタマイズされたマネージドクラスターテンプレートのほかに、ハブクラスターからの値を使用して設定ポリシーを定義するためのハブクラスターテンプレートもサポートします。この組み合わせにより、ポリシー定義の各ターゲットクラスターまたはハードコーディング設定値に個別のポリシーを作成する必要がなくなります。

ハブクラスターテンプレートは Golang テキストテンプレートの仕様をベースとしており、`{{hub ... hub}}` 区切り文字は設定ポリシーのハブクラスターテンプレートを示します。

セキュリティを確保するには、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。詳細は、[ハブおよびマネージドクラスターテンプレートの比較](#) をご覧ください。

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、シークレットからの値を自動的に暗号化する **fromSecret** を使用するか、他の値を暗号化するために **protect** します。

2.6.5.3.1. テンプレート処理

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。

2.6.5.3.2. 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。

2.6.5.4. 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。

2.6.5.4.1. 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。

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。

2.6.5.4.2. 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	<code>{{hub ... hub}}</code>	<code>{{ ... }}</code>
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。	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。

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	<p>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、テンプレート関数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。</p> <p>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。</p> <p>同等の呼び出しは、次の構文を使用する場合があります: <code>{{hub "(lookup "v1" "Secret" "default" "my-hub-secret").data.message protect hub}}</code></p>	<p>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、テンプレート関数を参照してください。</p>
関数出力ストレージ	<p>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 が有効な場合には、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。</p>	<p>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。</p>

テンプレート	ハブクラスター	マネージドクラスター
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。

2.6.6. セキュリティポリシーの管理

セキュリティポリシーを作成して、指定のセキュリティ標準、カテゴリ、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。

- [セキュリティポリシーの作成](#)
 - [コマンドラインインターフェイスからのセキュリティポリシーの作成](#)
 - [CLIからのセキュリティポリシーの表示](#)
 - [コンソールからのクラスターセキュリティポリシーの作成](#)
 - [コンソールからのセキュリティポリシーの表示](#)
 - [CLIからのポリシーセットの作成](#)
 - [コンソールからのポリシーセットの作成](#)
- [セキュリティポリシーの更新](#)
 - [セキュリティポリシーの無効化](#)
- [セキュリティポリシーの削除](#)
 - [コンソールからのポリシーセットの削除](#)
- [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](#)

2.6.6.1. セキュリティポリシーの作成

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、配置ルールおよび配置バインディングを定義する

必要があります。Cluster selector フィールドに値を入力して、PlacementRule と PlacementBinding を定義します。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。Red Hat Advanced Cluster Management for Kubernetes ポリシーに必要なオブジェクトの定義を表示します。

- **PlacementRule**: ポリシーをデプロイする必要がある **クラスターセレクター** を定義します。
- **PlacementBinding**: 配置を配置ルールにバインドします。

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。

2.6.6.1.1. 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。

```
kubectl 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. **PlacementRule** を定義します。 **clusterSelector** を調整して、 **PlacementRule** を変更し、ポリシーを適用する必要があるクラスターを指定してください。 [配置ルールの例の概要](#) を確認してください。

PlacementRule は以下のようになります。

```

apiVersion: apps.open-cluster-management.io/v1
kind: PlacementRule
metadata:
  name: placement1
spec:
  clusterConditions:
    - type: ManagedClusterConditionAvailable
      status: "True"
  clusterNames:
    - "cluster1"
    - "cluster2"
  - clusterSelector
    matchLabels:
      cloud: IBM

```

4. **PlacementBinding** を定義して、ポリシーを **PlacementRule** をバインドします。 **PlacementBinding** は以下の 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: apps.open-cluster-management.io
  kind: PlacementRule
subjects:
  - name: policy1
    apiGroup: policy.open-cluster-management.io
    kind: Policy

```

2.6.6.1.1.1. 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。

```

kubectl 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。

```

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

```


2.6.6.1.2. コンソールからのクラスターセキュリティポリシーの作成

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。

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

```

```

---
apiVersion: apps.open-cluster-management.io/v1
kind: PlacementRule
metadata:
  name: placement-pod
spec:
  clusterConditions: []
  clusterSelector:
    matchLabels:
      cloud: "IBM"

```

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。

2.6.6.1.2.1. 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およびそのステータスを表示します。**Governance** ページに移動し、ポリシー表の一覧を表示します。**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。

2.6.6.1.3. 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。

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

2.6.6.1.4. コンソールからのポリシーセットの作成

ナビゲーションメニューから **Govern** を選択します。次に、**Policy sets** タブを選択します。**Create policy set** ボタンを選択し、フォームを完了します。ポリシーセットの詳細情報を追加したら、**Submit** ボタンを選択します。

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** を表示します。

2.6.6.2. セキュリティーポリシーの更新

以下のセクションを参照して、セキュリティーポリシーを更新します。

2.6.6.2.1. CLI からのポリシーセットへのポリシーの追加

以下のコマンドを実行してポリシーセットを編集します (**kubectl edit policysets your-policyset-name**)。

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。**kubectl apply -f your-added-policy.yaml** のコマンドを使用して、ポリシーセットの配置セクションに、追加したポリシーを適用します。**PlacementBinding** および **PlacementRule** 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。

2.6.6.2.2. コンソールからのポリシーセットへのポリシーの追加

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。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。

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。

2.6.6.2.3. 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。

2.6.6.3. 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ポリシーを削除します。

- CLI からセキュリティ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ポリシーを削除します。

```
kubectl 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ていることを確認します: **kubectl get policies.policy.open-cluster-management.io <policy-name> -n <policy-namespace>**

- コンソールからセキュリティ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。

2.6.6.3.1. コンソールからのポリシーセットの削除

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。

Delete ボタンをクリックします。

他のポリシーの管理については、[セキュリティポリシーの管理](#) を参照してください。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。

2.6.6.4. 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。

- **DeletelfCreated**: 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。

CLI からポリシーを作成するときに、値を YAML で直接設定できます。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、**pruneObjectBehavior** 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。

2.6.7. 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](#)
 - [CLI からの設定ポリシーの作成](#)
 - [CLI からの設定ポリシーの表示](#)
 - [コンソールからの設定ポリシーの作成](#)
 - [コンソールからの設定ポリシーの表示](#)
- [設定ポリシーの更新](#)
 - [設定ポリシーの無効化](#)
- [設定ポリシーの削除](#)

2.6.7.1. 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。

2.6.7.1.1. 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。

2.6.7.1.2. 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>
```

2.6.7.1.3. 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。

2.6.7.1.4. 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** タブが表示されません。

2.6.7.2. 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。

2.6.7.2.1. 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。

2.6.7.3. 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。

[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、[セキュリティポリシーの管理](#) を参照してください。

2.6.8. gatekeeper Operator ポリシーの管理

gatekeeper Operator ポリシーを使用して、マネージドクラスターに gatekeeper Operator および gatekeeper をインストールします。以下のセクションでは、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 operator のカスタムリソース](#)
- [gatekeeper および gatekeeper Operator のアップグレード](#)
- [gatekeeper Operator ポリシーの更新](#)
 - [コンソールからの gatekeeper Operator ポリシーの表示](#)
 - [gatekeeper Operator ポリシーの無効化](#)
- [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](#)

2.6.8.1. 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 CR を作成して gatekeeper を設定します。[gatekeeper Operator CR](#) の例を表示します。

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。

2.6.8.2. コンソールからの gatekeeper ポリシーの作成

コンソールから gatekeeper ポリシーを作成して、インストールします。または、サンプル YAML を表示して、[policy-gatekeeper-operator.yaml](#) をデプロイすることもできます。

クラスターにログインしたら、**Governance** ページに移動します。

Create policy を選択します。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。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。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については、[Gatekeeper Helm Chart](#) を参照してください。

2.6.8.2.1. gatekeeper Operator CR

```
apiVersion: operator.gatekeeper.sh/v1alpha1
kind: Gatekeeper
metadata:
  name: gatekeeper
spec:
  audit:
    replicas: 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
  webhook:
    replicas: 2
    logLevel: ERROR
```



```
emitAdmissionEvents: Enabled
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"
```

2.6.8.3. 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。

2.6.8.4. 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。

2.6.8.4.1. 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。

2.6.8.4.2. 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。

2.6.8.5. 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。

```
kubectl 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。

```
kubectl 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。

2.6.8.6. 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 (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。
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。

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。サードパーティポリシーと製品の統合に関する詳細は、[サードパーティポリシーコントローラーの統合](#) を参照してください。

2.6.9. 切断された環境でのオペレーターポリシーの管理

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。デプロイメントするポリシーを使用して、Operator Lifecycle Manager オペレーターをインストールするポリシーをデプロイメントする場合は、[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。

次の手順を実行して、オペレーターイメージへのアクセスを検証します。

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ます。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
コンテナのセキュリティ	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

2.7. 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
  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。

2.8. 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。

2.9. 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 制約および制約テンプレートの統合](#)
- [ポリシージェネレーター](#)
 - [Operator をインストールするためのポリシーの生成](#)

2.9.1. gatekeeper 制約および制約テンプレートの統合

gatekeeper は、Open Policy Agent (OPA) で実行されるカスタムリソース定義 (CRD) ベースのポリシーを適用する検証用の 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 ポリシーを使用して、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、Gatekeeper ポリシーをクラスターに適用するには、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 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。gatekeeper ポリシーには、制約テンプレート (**ConstraintTemplates**) と **Constraints**、監査テンプレート、および受付テンプレートが含まれます。詳細は、[Gatekeeper upstream repository](#) を参照してください。

Red Hat Advanced Cluster Management では、Gatekeeper バージョン 3.3.0 をサポートし、Red Hat Advanced Cluster Management gatekeeper ポリシーで以下の制約テンプレートを適用します。

- **ConstraintTemplates** と制約: **policy-gatekeeper-k8srequiredlabels** を使用して、マネージドクラスターで gatekeeper 制約テンプレートを作成します。

```

apiVersion: policy.open-cluster-management.io/v1
kind: ConfigurationPolicy
metadata:
  name: policy-gatekeeper-k8srequiredlabels
spec:
  remediationAction: enforce # will be overridden by remediationAction in parent policy
  severity: low
  object-templates:
    - complianceType: musthave
      objectDefinition:
        apiVersion: templates.gatekeeper.sh/v1beta1
        kind: ConstraintTemplate
        metadata:
          name: k8srequiredlabels
        spec:
          crd:
            spec:
              names:
                kind: K8sRequiredLabels
              validation:
                # Schema for the `parameters` field
                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])
                    }
            }
          - complianceType: musthave
            objectDefinition:
              apiVersion: constraints.gatekeeper.sh/v1beta1
              kind: K8sRequiredLabels
              metadata:
                name: ns-must-have-gk
              spec:
                match:
                  kinds:
                    - apiGroups: [""]
                      kinds: ["Namespace"]
                namespaces:
                  - e2etestsuccess

```

```

- e2etestfail
parameters:
  labels: ["gatekeeper"]

```

- 監査テンプレート: **policy-gatekeeper-audit** を使用して、既存の設定ミスを検出するために適用された gatekeeper ポリシーに対して、既存のリソースを定期的に確認して評価します。

```

apiVersion: policy.open-cluster-management.io/v1
kind: ConfigurationPolicy
metadata:
  name: policy-gatekeeper-audit
spec:
  remediationAction: inform # will be overridden by remediationAction in parent policy
  severity: low
  object-templates:
    - complianceType: musthave
      objectDefinition:
        apiVersion: constraints.gatekeeper.sh/v1beta1
        kind: K8sRequiredLabels
        metadata:
          name: ns-must-have-gk
        status:
          totalViolations: 0

```

- 受付テンプレート: **policy-gatekeeper-admission** を使用して、gatekeeper 受付 Webhook により作成される設定ミスを確認します。

```

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

```

- 詳細は、[policy-gatekeeper-sample.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。

2.9.2. 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 ドキュメント](#) を参照してください。

このバージョンの Red Hat Advanced Cluster Management にバンドルされているポリシージェネレーターのバージョンは v1.10.0 です。詳細は、以下のトピック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](#)
- [関連リソース](#)

2.9.2.1. 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。特に、ポリシージェネレーターを使用して以下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。

2.9.2.2. 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が含まれる 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**、**PlacementRule** と **PlacementBinding** は以下の例のようになります。

```

apiVersion: apps.open-cluster-management.io/v1
kind: PlacementRule
metadata:
  name: placement-config-data
  namespace: policies
spec:
  clusterConditions:
    - status: "True"
      type: ManagedClusterConditionAvailable
  clusterSelector:
    matchExpressions: []
---
apiVersion: policy.open-cluster-management.io/v1
kind: PlacementBinding
metadata:
  name: binding-config-data
  namespace: policies
placementRef:
  apiGroup: apps.open-cluster-management.io
  kind: PlacementRule
  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
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

```

2.9.2.3. ポリシージェネレーター設定の参照テーブル

namespace 以外の **policyDefaults** セクションに含まれる全フィールドは、ポリシーごとに上書きされる可能性がある点に注意してください。

表2.14 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 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。

フィールド	任意または必須	説明
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。
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。

フィールド	任意または必須	説明
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.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。

フィールド	任意または必須	説明
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、名前が 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。
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。
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。

フィールド	任意または必須	説明
<code>policyDefaults.dependencies[].compliance</code>	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。
<code>policyDefaults.dependencies[].kind</code>	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。
<code>policyDefaults.dependencies[].apiVersion</code>	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。
<code>policyDefaults.extraDependencies</code>	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。
<code>policyDefaults.extraDependencies[].name</code>	必須	依存しているオブジェクトの名前。
<code>policyDefaults.extraDependencies[].namespace</code>	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。
<code>policyDefaults.extraDependencies[].compliance</code>	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。
<code>policyDefaults.extraDependencies[].kind</code>	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。

フィールド	任意または必須	説明
<code>policyDefaults.extraDependencies[].apiVersion</code>	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。
<code>policyDefaults.ignorePending</code>	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。
<code>policyDefaults.orderPolicies</code>	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。
<code>policyDefaults.orderManifests</code>	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。
<code>policyDefaults consolidateManifests</code>	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。
<code>policyDefaults.informGatekeeperPolicies</code>	任意	このポリシーが、違反した gatekeeper ポリシーマニフェストを参照すると、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必要があるかどうか決定されます。デフォルト値は true です。

フィールド	任意または必須	説明
<code>policyDefaults.informKyvernoPolicies</code>	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まります。デフォルト値は true です。
<code>policyDefaults.policySets</code>	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。
<code>policyDefaults.generatePlacementWhenInSet</code>	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。
<code>policyDefaults.placement</code>	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。
<code>policyDefaults.placement.name</code>	任意	同じクラスターセクターが含まれる配置ルール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
<code>policyDefaults.placement.placementName</code>	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。
<code>policyDefaults.placement.placementPath</code>	任意	既存の配置を再利用するには、 kustomization.yaml ファイルの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。新規 Placement を生成するには clusterSelectors を参照してください。
<code>policyDefaults.placement.clusterSelectors</code>	任意	クラスターセクターを key:value の形式で定義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。
<code>policyDefaults.placement.placementRuleName</code>	任意	クラスターにすでに存在する配置ルールを使用するには、ここでその名前を指定します。 PlacementRule は作成されませんが、 PlacementBinding はポリシーをこの PlacementRule にバインドします。
<code>policyDefaults.placement.placementRulePath</code>	任意	既存の配置ルールを再利用するには、 kustomization.yaml ファイルの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、 labelSelector を参照してください。
<code>policyDefaults.placement.labelSelector</code>	任意	クラスターセクターを key:value の形式で定義して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
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については、 policyDefaults を参照してください。
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 と metadataComplianceType 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。

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。
policySets	任意	作成するポリシーセットのリストです。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。
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。
policySets[].placement	任意	ポリシーセットの配置設定。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。配置のドキュメントについては、 policyDefaults.placement を参照してください。ただし、 policyDefaults.placement の設定はポリシーセットには適用されません。

2.9.2.4. 関連リソース

- [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。

- 詳細については、[policy-generator-plugin](#) リポジトリ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。
- [サードパーティ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。
- その他のトピックについては、[ガバナンス](#) ドキュメントを参照してください。
- 詳細については、[Kustomize の適用](#) 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。
- 詳細については、[ゲートキーパー](#) を参照してください。

2.9.3. Operator をインストールするためのポリシーの生成

Red Hat Advanced Cluster Management ポリシーの一般的な用途は、Operator を1つ以上の Red Hat OpenShift Container Platform マネージドクラスターにインストールすることです。ポリシージェネレーターを使用してポリシーを生成する方法、および生成されたポリシーを使用して OpenShift GitOps Operator をインストールする方法については、引き続きお読みください。

- [OpenShift GitOps をインストールするポリシーの生成](#)
- [Compliance Operator をインストールするポリシーの生成](#)
- [OperatorGroups でのポリシー依存関係の使用](#)
- [関連リソース](#)

2.9.3.1. OpenShift GitOps をインストールするポリシーの生成

ポリシージェネレーターを使用して、OpenShift GitOps をインストールするポリシーを生成できます。OpenShift GitOps Operator は次の例で使用されている `all namespaces` インストールモードを提供します。次の例のように、`openshift-gitops-subscription.yaml` という `Subscription` マニフェストファイルを作成します。

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: openshift-gitops-operator
  namespace: openshift-operators
spec:
  channel: stable
  name: openshift-gitops-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
```

Operator の特定のバージョンに固定するには、パラメーターと値 `spec.startingCSV: openshift-gitops-operator.v<version>` を追加します。<version> を希望のバージョンに置き換えます。

PolicyGenerator 設定ファイルが必要です。`policy-generator-config.yaml` という名前の設定ファイルを使用してポリシーを生成し、すべての OpenShift Container Platform マネージドクラスターに OpenShift GitOps をインストールします。以下の例を参照してください。

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

```

metadata:
  name: install-openshift-gitops
policyDefaults:
  namespace: policies
  placement:
    clusterSelectors:
      vendor: "OpenShift"
  remediationAction: enforce
policies:
- name: install-openshift-gitops
  manifests:
    - path: openshift-gitops-subscription.yaml

```

最後に必要なファイルは **kustomization.yaml** で、次の設定が必要です。

```

generators:
- policy-generator-config.yaml

```

生成されたポリシーは、以下のファイルのようになります。

```

apiVersion: apps.open-cluster-management.io/v1
kind: PlacementRule
metadata:
  name: placement-install-openshift-gitops
  namespace: policies
spec:
  clusterConditions:
    - status: "True"
      type: ManagedClusterConditionAvailable
  clusterSelector:
    matchExpressions:
      - key: vendor
        operator: In
        values:
          - OpenShift
---
apiVersion: policy.open-cluster-management.io/v1
kind: PlacementBinding
metadata:
  name: binding-install-openshift-gitops
  namespace: policies
placementRef:
  apiGroup: apps.open-cluster-management.io
  kind: PlacementRule
  name: placement-install-openshift-gitops
subjects:
- apiGroup: policy.open-cluster-management.io
  kind: Policy
  name: install-openshift-gitops
---
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
name: install-openshift-gitops
namespace: policies
spec:
  disabled: false
  policy-templates:
    - objectDefinition:
        apiVersion: policy.open-cluster-management.io/v1
        kind: ConfigurationPolicy
        metadata:
          name: install-openshift-gitops
        spec:
          object-templates:
            - complianceType: musthave
              objectDefinition:
                apiVersion: operators.coreos.com/v1alpha1
                kind: Subscription
                metadata:
                  name: openshift-gitops-operator
                  namespace: openshift-operators
                spec:
                  channel: stable
                  name: openshift-gitops-operator
                  source: redhat-operators
                  sourceNamespace: openshift-marketplace
              remediationAction: enforce
              severity: low

```

OpenShift Container Platform ドキュメントのマニフェストから生成されたポリシーがサポートされています。ポリシージェネレーターを使用して、OpenShift Container Platform ドキュメントの設定ガイドを適用できます。

2.9.3.2. Compliance Operator をインストールするポリシーの生成

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。

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:
    - compliance-operator
---

```

```

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

```

PolicyGenerator 設定ファイルが必要です。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:
    clusterSelectors:
      vendor: "OpenShift"
    remediationAction: enforce
policies:
- name: install-compliance-operator
  manifests:
    - path: compliance-operator.yaml

```

最後に必要なファイルは **kustomization.yaml** で、次の設定が必要です。

```

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: apps.open-cluster-management.io/v1
kind: PlacementRule
metadata:
  name: placement-install-compliance-operator
  namespace: policies
spec:
  clusterConditions:
    - status: "True"
      type: ManagedClusterConditionAvailable
  clusterSelector:
    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: apps.open-cluster-management.io
  kind: PlacementRule
  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
  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
```

2.9.3.3. OperatorGroups でのポリシー依存関係の使用

OperatorGroup マニフェストを使用して Operator をインストールする場合、**Subscription** が作成される前に、クラスターに **OperatorGroup** が存在する必要があります。ポリシージェネレーターとともにポリシー依存関係機能を使用して、**Subscription** ポリシーを実施する前に **OperatorGroup** ポリシーが準拠していることを確認します。

必要な順序でマニフェストを一覧表示して、ポリシーの依存関係を設定します。たとえば、namespace ポリシーを最初に作成し、次に **OperatorGroup** を作成し、最後に **Subscription** を作成することができます。

ポリシージェネレーター設定マニフェストで **policyDefaults.orderManifests** パラメーターを有効にし、**policyDefaults consolidateManifests** を無効にして、マニフェスト間の依存関係を自動的に設定します。

2.9.3.4. 関連リソース

- 詳細については、[GitOps を使用したポリシーのデプロイ](#) を参照してください。
- [サードパーティ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。
- 詳細については、[OpenShift GitOps の理解](#) および [Operator](#) ドキュメントを参照してください。
- [クラスターへの Operator の追加 - CLI を使用した OperatorHub からのインストール](#) を参照してください。
- 詳細については、[Compliance Operator ドキュメント](#) を参照してください。
- [all namespaces インストールモード](#) を参照してください。
- [namespaced インストールモード](#) を参照してください。
- [Init コンテナを使用して、Pod のデプロイ前に、タスクを実行する](#) を参照してください。
- [ArgoCD](#) を参照してください。
- 以下の YAML 入力の例は、OpenShift Container Platform ドキュメントでサポートされます。
 - [インストール後のクラスタータスク](#)
 - [監査ログポリシーの設定](#)
 - [ログのサードパーティシステムへの転送](#)

2.9.4. OpenShift GitOps (ArgoCD) を使用したポリシー定義の管理

ArgoCD に基づく OpenShift GitOps を使用して、ポリシー定義を管理することもできます。このワークフローを許可するには、Red Hat Advanced Cluster Management ハブクラスターでポリシーを作成するためのアクセス権を OpenShift GitOps に付与する必要があります。ポリシーと配置を作成、読み

取り、更新、および削除するためのアクセス権を持つ、**openshift-gitops-policy-admin** と呼ばれる以下の **ClusterRole** リソースを作成します。**ClusterRole** は次の例のようになります。

```
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: openshift-gitops-policy-admin
rules:
- verbs:
  - get
  - list
  - watch
  - create
  - update
  - patch
  - delete
  apiGroups:
  - policy.open-cluster-management.io
  resources:
  - policies
  - placementbindings
- verbs:
  - get
  - list
  - watch
  - create
  - update
  - patch
  - delete
  apiGroups:
  - apps.open-cluster-management.io
  resources:
  - placementrules
- verbs:
  - get
  - list
  - watch
  - create
  - update
  - patch
  - delete
  apiGroups:
  - cluster.open-cluster-management.io
  resources:
  - placements
  - placements/status
  - placementdecisions
  - placementdecisions/status
```

ClusterRoleBinding オブジェクトを作成して、OpenShift GitOps サービスアカウントに **openshift-gitops-policy-admin ClusterRole** オブジェクトへのアクセスを許可します。**ClusterRoleBinding** は次の例のようになります。

```
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
```

```

metadata:
  name: openshift-gitops-policy-admin
subjects:
  - kind: ServiceAccount
    name: openshift-gitops-argocd-application-controller
    namespace: openshift-gitops
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: openshift-gitops-policy-admin

```

Red Hat Advanced Cluster Management ポリシー定義が OpenShift GitOps でデプロイされると、ポリシーのコピーが各マネージドクラスター名前空間に作成されます。これらのコピーは、複製されたポリシーと呼ばれます。OpenShift GitOps がこの複製されたポリシーを繰り返し削除したり、ArgoCD **Application** が同期していないことを示したりするのを防ぐために、**argocd.argoproj.io/compare-options: ignoreExtraneous** アノテーションは、Red Hat Advanced Cluster Management ポリシーフレームワークによって、それぞれのレプリケーションされたポリシーに自動的に設定されます。

ArgoCD がオブジェクトを追跡するために使用するラベルとアノテーションがあります。複製されたポリシーが ArgoCD にまったく表示されないようにするには、Red Hat Advanced Cluster Management ポリシー定義で **spec.copyPolicyMetadata** を **false** に設定して、これらの ArgoCD 追跡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ないようにすることができます。

2.9.4.1. ポリシージェネレーターと OpenShift GitOps (ArgoCD) の統合

ArgoCD に基づく OpenShift GitOps を使用して、GitOps を介してポリシージェネレーターを使用してポリシーを生成することもできます。ポリシージェネレーターは OpenShift GitOps コンテナイメージにプリインストールされていないため、カスタマイズを行う必要があります。続行するには、OpenShift GitOps Operator を Red Hat Advanced Cluster Management ハブクラスターにインストールし、ハブクラスターにログインする必要があります。

Kustomize の実行時、OpenShift GitOps がポリシージェネレーターにアクセスできるようにするには、Red Hat Advanced Cluster Management Application Subscription コンテナイメージから OpenShift GitOps コンテナに Policy Generator バイナリーをコピーするための Init コンテナが必要です。さらに、Kustomize の実行時、**--enable-alpha-plugins** フラグを提供するように、OpenShift GitOps を設定する必要があります。以下のコマンドを使用して、OpenShift GitOps **argocd** オブジェクトの編集を開始します。

```
oc -n openshift-gitops edit argocd openshift-gitops
```

次に、OpenShift GitOps **argocd** オブジェクトを変更して、以下の追加の YAML コンテンツを含めます。Red Hat Advanced Cluster Management の新しいメジャーバージョンがリリースされ、ポリシージェネレーターを新しいバージョンに更新したい場合は、Init コンテナで使用される **registry.redhat.io/rhacm2/multicluster-operators-subscription-rhel8** イメージをより新しいタグに更新する必要があります。以下の例を見て、**<version>** を 2.7 または目的の Red Hat Advanced Cluster Management バージョンに置き換えます。

```

apiVersion: argoproj.io/v1alpha1
kind: ArgoCD
metadata:
  name: openshift-gitops
  namespace: openshift-gitops
spec:
  kustomizeBuildOptions: --enable-alpha-plugins
repo:

```

```

env:
- name: KUSTOMIZE_PLUGIN_HOME
  value: /etc/kustomize/plugin
initContainers:
- args:
  - -c
  - cp /etc/kustomize/plugin/policy.open-cluster-management.io/v1/policygenerator/PolicyGenerator
    /policy-generator/PolicyGenerator
  command:
  - /bin/bash
  image: registry.redhat.io/rhacm2/multicluster-operators-subscription-rhel8:v<version>
  name: policy-generator-install
  volumeMounts:
  - mountPath: /policy-generator
    name: policy-generator
volumeMounts:
- mountPath: /etc/kustomize/plugin/policy.open-cluster-management.io/v1/policygenerator
  name: policy-generator
volumes:
- emptyDir: {}
  name: policy-generator

```

OpenShift GitOps がポリシージェネレーターを使用できるようになったので、Red Hat Advanced Cluster Management ハブクラスターでポリシーを作成するためのアクセス権を OpenShift GitOps に付与する必要があります。ポリシーとプレースメントを作成、読み取り、更新、および削除するためのアクセス権を持つ、**openshift-gitops-policy-admin** という名前の **ClusterRole** リソースを作成します。前の **ClusterRole** の例を参照してください。

さらに、**ClusterRoleBinding** オブジェクトを作成して、OpenShift GitOps サービスアカウントに **openshift-gitops-policy-admin ClusterRole** へのアクセスを許可します。**ClusterRoleBinding** は、次のようなリソースになる場合があります。

```

kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: openshift-gitops-policy-admin
subjects:
- kind: ServiceAccount
  name: openshift-gitops-argocd-application-controller
  namespace: openshift-gitops
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: openshift-gitops-policy-admin

```

2.9.4.2. 関連リソース

- [OperatorGroups](#) でのポリシー依存関係の使用 を参照してください。
- [Operator](#) をインストールするためのポリシーの生成 に戻ります。