Red Hat Advanced Cluster Management for Kubernetes 2.4

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。

Last Updated: 2022-12-18

Red Hat Advanced Cluster Management for Kubernetes 2.4 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。

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

法律上の通知

概要

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

目次

第1章 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る。詳細は、アクセス制御 を参照すること。

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

- 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
  - 設定ポリシーでのテンプレートのサポート
  - 整合性シールド保護 (テクノロジープレビュー)

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: クラスター管理者またはチーム管理者

注記: ネイティブの Red Hat Advanced Cluster Management インストールでのみ、別の証明書に置き換えての使用がサポートされます。

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 OpenShift Service Serving Certificates サービスによって作成および管理されます。

OpenShift Service Serving 証明書をローテーションすることもできます。詳細は、OpenShift ドキュメントに従い、生成されたサービス証明書を手動でローテーションし、サービス CA 証明書を手動でローテーションしてください。ローテーションが完了したら、以下のコマンドで新しい証明書をすべてのサービスに適用します。

```
oc -n open-cluster-management delete pod -l chart=management-ingress
```

クラスター内の関連 Pod は自動的に再起動します。

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

- 管理 Ingress 証明書の置き換え

- 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

- ハブクラスターの管理対象証明書の一覧表示
- ハブクラスターの管理対象証明書の更新
- OpenShift Container Platform 管理対象証明書の更新

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

- チャネル証明書
- 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
- Integrity-shield Webhook 証明書のローテーション (テクノロジープレビュー)

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の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。

以下の一覧を表示し、証明書が自動更新される場合の影響について確認します。

- CA 以外の証明書は、有効期間の残りが 73 日以下になると自動的に更新されます。証明書が更新されると、更新された証明書を使用するように関連するデプロイメントの Pod は自動的に再起動されます。

- CA 証明書は、有効期間の残りが 1 年間未満になると自動的に更新されます。証明書を更新したら、古い CA は削除されませんが、更新された CA と共存します。以前の証明書と更新された証明書はいずれも関連するデプロイメントで使用され、引き続き機能します。以前 CA 証明書は有効期限が切れると削除されます。

- 証明書の更新時には、ハブクラスターとマネージドクラスターの間のトラフィックは中断されません。

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
```

詳細は、証明書を生成するための OpenSSL コマンド を参照してください。alertmanager ルートのデフォルト自己署名証明書を復元する場合は、管理 Ingress のデフォルトの自己署名証明書の復元 を参照して **open-cluster-management-observability** namespace にある 2 つのシークレットを削除します。

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

1.1.2.1. ハブクラスターの管理対象証明書の一覧表示

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

```
oc get secret -n open-cluster-management -o custom-columns=Name:.metadata.name,Expiration:.metadata.annotations.service\\.beta\\.openshift\\.io/expiry | grep -v '<none>'
```

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. ハブクラスターの管理対象証明書の更新

List hub cluster managed certificates セクションでコマンドを実行して、ハブクラスターの管理対象証明書を更新できます。更新する必要の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. OpenShift Container Platform 管理対象証明書の更新

Red Hat Advanced Cluster Management の Webhook とプロキシーサーバーで使用される、OpenShift Container Platform 管理対象証明書を更新します。

OpenShift Container Platform の管理対象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、削除する必要のあるシークレットが存在しない場合があります。

2. 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 管理対象証明書を含む 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	-
multicluster-operators-application-svc	open-cluster-management	multicluster-operators-application-8c446664c-5lbfk	-
multiclusterhub-operator-webhook	open-cluster-management	multiclusterhub-operator-bfd948595-mnhjc	-
ocm-webhook	open-cluster-management	ocm-webhook-679444669c-5cg76	ocm-webhook-secret
cluster-manager-registration-webhook	open-cluster-management-hub	cluster-manager-registration-webhook-fb7b99c-d8wfc	registration-webhook-serving-cert

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)
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 管理対象証明書

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、証明書の検証を無効にできます。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。証明書の検証を無効にすると、セキュリティー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。

1.1.4.2. Integrity-shield Webhook 証明書のローテーション (テクノロジープレビュー)

Integrity-shield 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. IntegrityShield カスタムリソースを編集して、**integrity-shield-operator-system** namespace を **inScopeNamespaceSelector** 設定の namespace 除外リストに追加します。以下のコマンドを実行してリソースを編集します。

```
oc edit integrityshield integrity-shield-server -n integrity-shield-operator-system
```

2. 次のコマンドを実行して、integrity-shield 証明書を含むシークレットを削除します。

```
oc delete secret -n integrity-shield-operator-system ishield-server-tls
```

3. シークレットが再作成されるように、Operator を削除します。Operator の Pod 名がシステムの Pod 名と一致していることを確認してください。以下のコマンドを実行します。

```
oc delete po -n integrity-shield-operator-system integrity-shield-operator-controller-manager-64549569f8-v4pz6
```

4. Integrity-shield サーバー Pod を削除して、次のコマンドで新しい証明書の使用を開始します。

```
oc delete po -n integrity-shield-operator-system integrity-shield-server-5fbdfbbbd4-bbfbz
```

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。

1.2. 管理 INGRESS 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、Red Hat Advanced Cluster Management for Kubernetes ルートを更新して、管理 Ingress 証明書を置き換えることができます。

- 管理 Ingress 証明書を置き換えるための前提条件
- 独自の Ingress 証明書の置き換え
- 管理 Ingress のデフォルト自己署名証明書の復元

1.2.1. 管理 Ingress 証明書を置き換えるための前提条件

management-ingress 証明書と秘密鍵を作成して準備します。必要に応じて、OpenSSL で TLS 証明書を生成できます。証明書のコモンネームパラメーター (**CN**) を **manangement-ingress** に設定します。証明書を生成する場合は、以下の設定を追加します。

- 証明書のサブジェクトの別名 (SAN) リストのドメイン名として Red Hat Advanced Cluster Management for Kubernetes のルート名を含めます。
  以下のコマンドを実行してルート名を取得します。

```
oc get route -n open-cluster-management
```

  以下の応答が返される場合があります。

```
multicloud-console.apps.grchub2.dev08.red-chesterfield.com
```

1.2.1.1. 証明書を生成する設定ファイルの例

以下の設定ファイルおよび OpenSSL コマンドの例では、OpenSSL を使用して TLS 証明書を生成する方法を示しています。以下の **csr.cnf** 設定ファイルを確認してください。このファイルは、OpenSSL での証明書生成の設定設定を定義します。

```
[ req ]             # Main settings
default_bits = 2048       # Default key size in bits.
prompt = no               # Disables prompting for certificate values so the configuration file values are used.
default_md = sha256       # Specifies the digest algorithm.
req_extensions = req_ext  # Specifies the configuration file section that includes any extensions.
distinguished_name = dn   # Specifies the section that includes the distinguished name information.

[ dn ]              # Distinguished name settings
C = US                    # Country
ST = North Carolina             # State or province
L = Raleigh               # Locality
O = Red Hat Open Shift     # Organization
OU = Red Hat Advanced Container Management        # Organizational unit
CN = management-ingress  # Common name.

[ req_ext ]         # Extensions
subjectAltName = @alt_names # Subject alternative names

[ alt_names ]       # Subject alternative names
DNS.1 = multicloud-console.apps.grchub2.dev08.red-chesterfield.com

[ v3_ext ]          # x509v3 extensions
authorityKeyIdentifier=keyid,issuer:always  # Specifies the public key that corresponds to the private key that is used to sign a certificate.
basicConstraints=CA:FALSE                   # Indicates whether the certificate is a CA certificate during the certificate chain verification process.
#keyUsage=keyEncipherment,dataEncipherment   # Defines the purpose of the key that is contained in the certificate.
extendedKeyUsage=serverAuth                 # Defines the purposes for which the public key can be used.
subjectAltName=@alt_names                   # Identifies the subject alternative names for the identify that is bound to the public key by the CA.
```

注記: 管理 Ingress の正しいホスト名を使用して SAN ラベルが付いた **DNS.1** を必ず更新してください。

1.2.1.2. 証明書生成の OpenSSL コマンド

以下の OpenSSL コマンドは、上記の設定ファイルと合わせて使用して、必要な TLS 証明書を生成します。

1. 認証局 (CA) RSA 秘密鍵を生成します。

```
openssl genrsa -out ca.key 4096
```

2. CA キーを使用して自己署名の CA 証明書を生成します。

```
openssl req -x509 -new -nodes -key ca.key -subj "/C=US/ST=North Carolina/L=Raleigh/O=Red Hat OpenShift" -days 400 -out ca.crt
```

3. 証明書の RSA 秘密鍵を生成します。

```
openssl genrsa -out ingress.key 4096
```

4. 秘密鍵を使用して証明書署名要求 (CSR) を生成します。

```
openssl req -new -key ingress.key -out ingress.csr -config csr.cnf
```

5. CA 証明書、キー、および CSR を使用して署名済み証明書を生成します。

```
openssl x509 -req -in ingress.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out ingress.crt -
sha256 -days 300 -extensions v3_ext -extfile csr.cnf
```

6. 証明書の内容を調べます。

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

1.2.2. 独自の Ingress 証明書の置き換え

独自の Ingress 証明書を置き換えるには、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 **byo-ingress-tls** 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 create secret tls byo-ingress-tls-secret --cert ./ingress.crt --
key ./ingress.key
```

2. 以下のコマンドでシークレットが正しい namespace に作成されていることを確認します。

```
oc get secret -n open-cluster-management | grep -e byo-ingress-tls-secret -e byo-ca-cert
```

3. 任意: 以下のコマンドを実行して、CA 証明書を含むシークレットを作成します。

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

4. サブスクリプションを再デプロイするには、**management-ingress** サブスクリプションを削除します。前の手順で作成したシークレットが自動的に使用されます。以下のコマンドを実行します。

```
oc delete subscription management-ingress-sub -n open-cluster-management
```

5. 現在の証明書が、指定した証明書になっており、すべてのコンソールアクセスとログイン機能がそのまま維持されていることを確認します。

1.2.3. 管理 Ingress のデフォルト自己署名証明書の復元

1. 次のコマンドで、独自の証明書のシークレットを削除します。

```
oc delete secret byo-ca-cert byo-ingress-tls-secret -n open-cluster-management
```

2. サブスクリプションを再デプロイするには、**management-ingress** サブスクリプションを削除します。前の手順で作成したシークレットが自動的に使用されます。以下のコマンドを実行します。

-

```
oc delete subscription management-ingress-sub -n open-cluster-management
```

3. 現在の証明書が、指定した証明書になっており、すべてのコンソールアクセスとログイン機能がそのまま維持されていることを確認します。

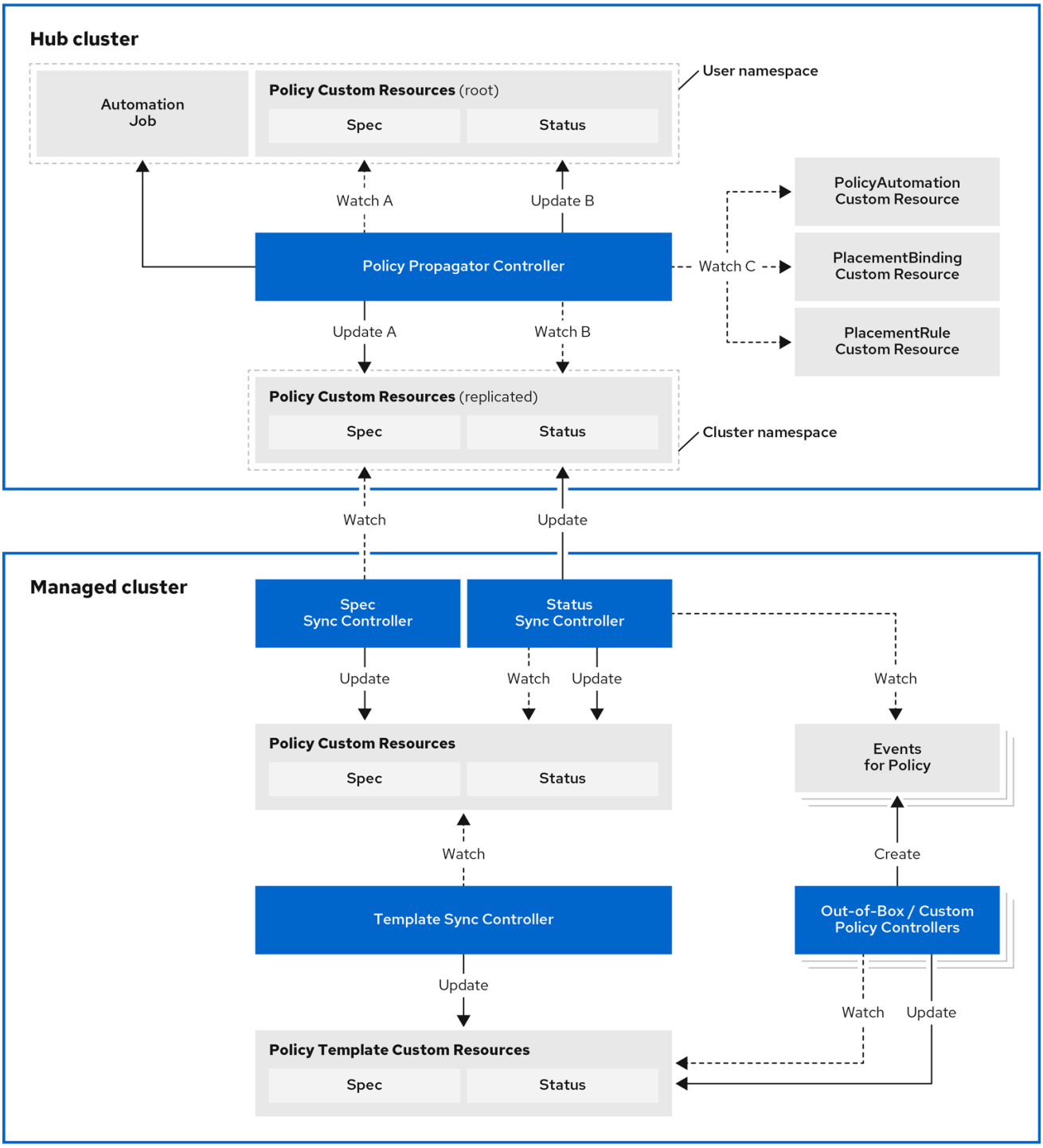
Red Hat Advanced Cluster Management で作成して管理する証明書の詳細は、証明書 を参照してください。クラスターのセキュリティー保護に関する詳細は、リスクおよびコンプライアンス ページに戻り、確認してください。

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、セキュリティー、規制準拠に関する内部標準を満たす必要があります。Red Hat Advanced Cluster Management for Kubernetes ガバナンスは、企業が独自のセキュリティーポリシーを導入するための拡張可能なポリシーフレームワークを提供します。

2.1. ガバナンスアーキテクチャー

Red Hat Advanced Cluster Management for Kubernetes ガバナンスライフサイクルを使用してクラスターのセキュリティーを強化します。製品ガバナンスのライフサイクルは、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ーおよびコンプライアンスを管理します。ガバナンスアーキテクチャーの以下の図を参照してください。



186_RHACM_1221

ガバナンスアーキテクチャーは、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。
  注記:
  - ポリシーがマネージドクラスターに伝播されると、複製されたポリシーには **namespaceName.policyName** という名前が付けられます。Kubernetes にはオブジェクト名の制限があるため、ポリシーの作成時は、**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、**policy-dhaz-cert** を検索すると、ハブクラスターから以下のポリシー名 (**default.policy-dhaz-cert**) が表示される場合があります。

- **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。事前定義済みの例や、クラスターへのポリシーのデプロイ方法を確認するには、**policy-collection** リポジトリー を参照してください。カスタムポリシーコントローラーおよびポリシーも指定できます。ポリシーに違反した場合は、ユーザーが選択するアクションを実行するように自動化を設定できます。詳細は、Ansible Tower でのガバナンスの設定 を参照してください。
  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。詳細は、ガバナンスのメトリクス を参照してください。

- **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ます。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。カスタムのポリシーコントローラーも作成できます。

- **オープンソースコミュニティー**: Red Hat Advanced Cluster Management ポリシーフレームワークの基盤を使ったコミュニティーの貢献をサポートします。ポリシーコントローラーおよびサードパーティーのポリシーも、**stolostron/policy-collection** リポジトリーに含まれます。GitOps を使用してポリシーを提供し、デプロイする方法を説明します。詳細は、GitOps を使用したポリシーのデプロイ を参照してください。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。詳細は、サードパーティーポリシーコントローラーの統合 を参照してください。

Red Hat Advanced Cluster Management for Kubernetes ポリシーフレームワークの設定、および Red Hat Advanced Cluster Management for Kubernetes の **ガバナンス** ダッシュボードの使用方法について説明します。

- ポリシーの概要
- ポリシーコントローラー
- サポート対象のポリシー
- セキュリティーポリシーの管理
- ハブクラスターのセキュリティー保護

2.2. ポリシーの概要

Red Hat Advanced Cluster Management for Kubernetes セキュリティーポリシーフレームワークを使用して、カスタムポリシーコントローラーおよびその他のポリシーを作成します。ポリシー作成には、Kubernetes カスタムリソース定義 (CRD) インスタンスを使用します。CRD の詳細は、Extend the Kubernetes API with CustomResourceDefinitions を参照してください。

各 Red Hat Advanced Cluster Management for Kubernetes ポリシーには、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、このページの以下の **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:

- **PlacementBinding** を作成して **PlacementRule** または **Placement** に関連付ける必要があります。
  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。
- ハブクラスターの namespae (クラスター namespace を除く) でポリシーを作成できます。クラスター namespace でポリシーを作成する場合には、Red Hat Advanced Cluster Management for Kubernetes により削除されます。
- 各クライアントおよびプロバイダーは、管理対象のクラウド環境で、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、セキュリティー、規制準拠に関する内部エンタープライズセキュリティー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ー機能を使用して、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:
  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

フィールド	説明
apiVersion	必須。この値は **policy.open-cluster-management.io/v1** に設定します。
kind	必須。ポリシーのタイプを指定するには、値を **Policy** に設定します。
metadata.name	必須。ポリシーリソースを識別する名前。
metadata.annotations	任意。ポリシーが検証を試みる標準セットを記述する、一連のセキュリティー情報の指定に使用します。ここに記載されているすべてのアノテーションは、コンマ区切りリストを含む文字列として表示されます。**注記**: コンソールの **ポリシー** ページで、ポリシー定義の標準およびカテゴリーに基づいてポリシー違反を表示できます。
annotations.policy.open-cluster-management.io/standards	ポリシーが関連するセキュリティー標準の名前。たとえば、アメリカ国立標準技術研究所 (NIST: National Institute of Standards and Technology) および Payment Card Industry (PCI) などがあります。

フィールド	説明
annotations.policy.open-cluster-management.io/categories	セキュリティーコントロールカテゴリーは、1つ以上の標準に関する特定要件を表します。たとえば、システムおよび情報の整合性カテゴリーには、HIPAA および PCI 標準で必要とされているように、個人情報保護のデータ転送プロトコルが含まれる場合があります。
annotations.policy.open-cluster-management.io/controls	チェックされるセキュリティー制御の名前。例: 証明書ポリシーコントローラー
spec.policy-templates	必須。1つ以上のポリシーを作成し、マネージドクラスターに適用するのに使用します。
spec.disabled	必須。この値は **true** または **false** に設定します。**disabled** パラメーターを使用すると、ポリシーを有効または無効にすることができます。
spec.remediationAction	任意。ポリシーの修正を指定します。パラメーターの値は **enforce** および **inform** です。指定すると、定義した **spec.remediationAction** 値は、**policy-templates** セクションから子ポリシーに定義した **remediationAction** パラメーターより優先されます。たとえば、**spec.remediationAction** の値のセクションを **enforce** に設定すると、**policy-templates** の **remediationAction** はランタイム時に **enforce** に設定されます。**重要**: ポリシーによっては、enforce 機能をサポートしない場合があります。

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:
          exclude: ["kube-*"]
          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て、**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
---
apiVersion: cluster.open-cluster-management.io/v1alpha1
kind: Placement
metadata:
  name: placement-policy-role
spec:
  predicates:
  - requiredClusterSelector:
      labelSelector:
        matchExpressions:
          - {key: environment, operator: In, values: ["dev"]}
```

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

他のポリシー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. 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。未設定のポリシーテンプレートを使用して事前定義のポリシーコントローラーおよびポリシーを適用し、Red Hat Advanced Cluster Management for Kubernetes ポリシーフレームワークを使用します。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) インスタンスです。

CRD の詳細は、Extend the Kubernetes API with CustomResourceDefinitions を参照してください。ポリシーコントローラーは、ポリシー違反を修正し、クラスターのステータスを準拠させます。

製品ポリシーフレームワークを使用して、カスタムポリシーおよびポリシーコントローラーを作成できます。詳細は、カスタムポリシーコントローラーの作成 (非推奨) を参照してください。

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

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ーポリシーを適用できます。

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。CRD の詳細は、Extend the Kubernetes API with CustomResourceDefinitions を参照してください。

設定ポリシーコントローラーは、インストール時にハブ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、**enforce** 機能をサポートし、以下のポリシーのコンプライアンスを監視します。

- メモリー使用状況のポリシー
- Namespace ポリシー
- イメージ脆弱性ポリシー
- 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
- gatekeeper 制約および制約テンプレートの統合

設定ポリシーの **remediationAction** が **enforce** に設定されている場合、コントローラーはターゲットのマネージドクラスターで複製ポリシーを作成します。設定ポリシーでテンプレート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。

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。

- 設定ポリシーコントローラーの YAML 設定
- 設定ポリシーの例
- 設定ポリシーの YAML の表

2.3.1.1. 設定ポリシーコントローラーの YAML 設定

```
Name:         configuration-policy-example
Namespace:
Labels:
APIVersion:   policy.open-cluster-management.io/v1
Kind:         ConfigurationPolicy
Metadata:
  Finalizers:
    finalizer.policy.open-cluster-management.io
Spec:
  Conditions:
    Ownership:
    NamespaceSelector:
      Exclude:
      Include:
```

```
    RemediationAction:
 Status:
   CompliancyDetails:
     Configuration-Policy-Example:
       Default:
       Kube - Public:
   Compliant:          Compliant
 Events:
```

2.3.1.2. 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: []
  remediationAction: inform
    severity: low
    object-templates:
    - complianceType: musthave
      objectDefinition:
        apiVersion: v1
        kind: Pod
        metadata:
          name: pod
        spec:
          containers:
          - image: 'pod-image'
            name:
            ports:
            - containerPort: 80
```

2.3.1.3. 設定ポリシーの YAML の表

表2.1パラメーターの表

フィールド	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	必須。監視する設定ポリシーと設定ポリシーの修正方法に関する仕様。

フィールド	説明
spec.namespace	namespace を使用したオブジェクトまたはリソースに必要です。ポリシーの適用先のハブクラスター内にある namespace。**include** パラメーターに、ポリシーを適用する namespace を最低でも 1 つ入力します。**exclude** パラメーターには、ポリシーを明示的に適用しない namespace を指定します。
spec.remediationAction	必須。ポリシーの修正を指定します。**inform** と入力します。
spec.remediationAction.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。パラメーター値 **low**、**medium**、または **high** を使用します。
spec.remediationAction.complianceType	必須。マネージドクラスターに評価または適用する必要のあるロールおよび他の Kubernetes オブジェクトの予想される動作を一覧表示するのに使用されます。以下の動詞をパラメーター値として使用する必要があります。 **mustonlyhave**: 正確な名前と関連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。 **musthave**: 指定された object-template と同じ名前を持つオブジェクトが存在することを示します。テンプレートの他のフィールドは、オブジェクトに存在するもののサブセットです。 **mustnothave**: 仕様またはルールに関係なく、同じ名前またはラベルを持つオブジェクトは存在できず、削除する必要があることを示します。

NIST Special Publication 800-53 (Rev. 4) を使用するポリシーサンプルおよび、**CM-Configuration-Management フォルダー** の Red Hat Advanced Cluster Management がサポートするポリシーサンプルを参照してください。ポリシーがハブクラスターにどのように適用されるかについては、サポート対象のポリシー 参照してください。

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。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。

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。CRD の詳細は、Extend the Kubernetes API with CustomResourceDefinitions を参照してください。

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。

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
  namespace:
  labels: category=system-and-information-integrity
spec:
  namespaceSelector:
    include: ["default"]
    exclude: ["kube-*"]
  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2.2 パラメーターの表

フィールド	説明

フィールド	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.namespace	必須。ポリシーが作成されるマネージドクラスター内の namespace。
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	必須。監視および更新する証明書の仕様。
spec.namespaceSelector	必須。ポリシーを適用するマネージドクラスターの namespace。**Include** および **Exclude** のパラメーター値を入力します。**注記**: 複数の証明書ポリシーを作成してそのポリシーを同じマネージドクラスターに適用する場合、各ポリシーの **namespaceSelector** には別の値を割り当てる必要があります。 • 証明書ポリシーコントローラーの **namespaceSelector** がどの namespace にも一致しない場合は、ポリシーが準拠しているとみなされます。
spec.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。パラメーター値 **low**、**medium**、または **high** を使用します。
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。

フィールド	説明
spec.minimumCADuration	任意。値を設定して、他の証明書とは異なる値で、まもなく有効期限が切れる可能性がある署名証明書を特定します。パラメーターの値が指定されていないと、CA 証明書の有効期限は **minimumDuration** で使用した値になります。詳細は Golang Parse Duration を参照してください。
spec.maximumDuration	任意。値を設定して、任意の制限期間を超えて作成された証明書を特定します。パラメーターは Golang の期間形式を使用します。詳細は Golang Parse Duration を参照してください。
spec.maximumCADuration	任意。値を設定して、定義した制限期間を超えて作成された署名証明書を特定します。パラメーターは Golang の期間形式を使用します。詳細は Golang Parse Duration を参照してください。
spec.allowedSANPattern	任意。証明書に定義した全 SAN エントリーと一致する必要がある正規表現。このパラメーターを使用して、パターンと DNS 名を照合します。詳細は Golang Regular Expression syntax を参照してください。
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、セキュリティーポリシーの管理 を参照してください。他のトピックについては、ポリシーコントローラー を参照してください。

2.3.3. 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。詳細

は、Extend the Kubernetes API with CustomResourceDefinitions を参照してください。

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2.3 パラメーターの表

フィールド	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	必須。ポリシーの設定詳細を追加します。
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。パラメーター値 **low**、**medium**、または **high** を使用します。

フィールド	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。
spec.ignoreClusterRoleBindings	任意。無視するクラスターのロールバインディング名を指定する正規表現 (regex) 値の一覧。これらの正規表現の値は、Go regexp 構文 に従う必要があります。デフォルトでは、名前が **system:** で始まるすべてのクラスターロールバインディングは無視されます。これをより厳格な値に設定することが推奨されます。クラスターのロールバインディング名を無視するには、一覧を単一の値 **.^** か、または一致しない他の正規表現に設定します。
spec.maxClusterRoleBindingUsers	必須。ポリシーが違反しているとみなされるまでに利用可能な IAM rolebinding の最大数。

2.3.3.3. IAM ポリシーの例

IAM ポリシーのサンプルを確認するには、**policy-limitclusteradmin.yaml** を参照してください。詳細は、セキュリティーポリシーの管理 を参照してください。

他のトピックについては、ポリシーコントローラー を参照してください。

2.3.4. カスタムポリシーコントローラーの作成 (非推奨)

カスタムポリシーコントローラーの作成、適用、表示、および更新について説明します。ポリシーコントローラーがクラスターにデプロイする YAML ファイルを作成できます。以下のセクションを参照して、ポリシーコントローラーを作成します。

2.3.4.1. ポリシーコントローラーの作成

governance-policy-framework リポジトリーにあるポリシーコントローラーフレームワークを使用します。ポリシーコントローラーを作成するには、以下の手順を完了します。

1. 以下のコマンドを実行して **governance-policy-framework** リポジトリーのクローンを作成します。

```
git clone git@github.com:stolostron/governance-policy-framework.git
```

2. ポリシースキーマ定義を更新してコントローラーポリシーをカスタマイズします。ポリシーは次のような内容になります。

```
metadata:
  name: samplepolicies.policies.open-cluster-management.io
spec:
  group: policy.open-cluster-management.io
  names:
    kind: SamplePolicy
```

```
    listKind: SamplePolicyList
    plural: samplepolicies
    singular: samplepolicy
```

3. **SamplePolicy** の種類を監視するようにポリシーコントローラーを更新します。以下のコマンドを実行します。

```
for file in $(find . -name "*.go" -type f); do  sed -i "" "s/SamplePolicy/g" $file; done
for file in $(find . -name "*.go" -type f); do  sed -i "" "s/samplepolicy-controller/samplepolicy-controller/g" $file; done
```

4. 以下の手順を実行してポリシーコントローラーを再コンパイルし、実行します。

   a. クラスターにログインします。

   b. ユーザーアイコンを選択し、**クライアントの設定** をクリックします。

   c. 設定情報をコマンドラインにコピーアンドペーストし、**Enter** を押します。

   d. 以下のコマンドを実行してポリシー CRD を適用し、コントローラーを起動します。

```
export GO111MODULE=on

kubectl apply -f deploy/crds/policy.open-cluster-management.io_samplepolicies_crd.yaml

export WATCH_NAMESPACE=<cluster_namespace_on_hub>

go run cmd/manager/main.go
```

   コントローラーが実行していることを示す以下の出力が表示される場合があります。

```
{“level”:”info”,”ts”:1578503280.511274,”logger”:”controller-runtime.manager”,”msg”:”starting metrics server”,”path”:”/metrics”}
{“level”:”info”,”ts”:1578503281.215883,”logger”:”controller-runtime.controller”,”msg”:”Starting Controller”,”controller”:”samplepolicy-controller”}
{“level”:”info”,”ts”:1578503281.3203468,”logger”:”controller-runtime.controller”,”msg”:”Starting workers”,”controller”:”samplepolicy-controller”,”worker count”:1}
Waiting for policies to be available for processing…
```

   e. ポリシーを作成し、コントローラーがポリシーを取得し、そのポリシーをクラスターに適用していることを確認します。以下のコマンドを実行します。

```
kubectl apply -f deploy/crds/policy.open-cluster-management.io_samplepolicies_crd.yaml
```

   ポリシーが適用されると、カスタムコントローラーによってポリシーが監視され、検出されることを示すメッセージが表示されます。メッセージは以下の内容のようになります。

```
{"level":"info","ts":1578503685.643426,"logger":"controller_samplepolicy","msg":"Reconciling SamplePolicy","Request.Namespace":"default","Request.Name":"example-samplepolicy"}
{"level":"info","ts":1578503685.855259,"logger":"controller_samplepolicy","msg":"Reconciling SamplePolicy","Request.Namespace":"default","Request.Name":"example-samplepolicy"}
Available policies in namespaces:
```

```
namespace = kube-public; policy = example-samplepolicy
namespace = default; policy = example-samplepolicy
namespace = kube-node-lease; policy = example-samplepolicy
```

5. 以下のコマンドを実行して、**status** フィールドでコンプライアンスの詳細を確認します。

```
kubectl describe SamplePolicy example-samplepolicy -n default
```

出力は次のような内容になります。

```
status:
  compliancyDetails:
    example-samplepolicy:
      cluster-wide:
      - 5 violations detected in namespace `cluster-wide`, there are 0 users violations
        and 5 groups violations
      default:
      - 0 violations detected in namespace `default`, there are 0 users violations
        and 0 groups violations
      kube-node-lease:
      - 0 violations detected in namespace `kube-node-lease`, there are 0 users violations
        and 0 groups violations
      kube-public:
      - 1 violations detected in namespace `kube-public`, there are 0 users violations
        and 1 groups violations
  compliant: NonCompliant
```

6. ポリシールールおよびポリシーロジックを変更して、ポリシーコントローラーの新規ルールを作成します。以下の手順を実行します。

   a. **SamplePolicySpec** を更新して、YAML ファイルに新規フィールドを追加します。仕様は次のような内容になります。

```
spec:
  description: SamplePolicySpec defines the desired state of SamplePolicy
  properties:
    labelSelector:
      additionalProperties:
        type: string
      type: object
    maxClusterRoleBindingGroups:
      type: integer
    maxClusterRoleBindingUsers:
      type: integer
    maxRoleBindingGroupsPerNamespace:
      type: integer
    maxRoleBindingUsersPerNamespace:
      type: integer
```

   b. samplepolicy_controller.go の **SamplePolicySpec** 構造を新しいフィールドで更新します。

   c. **samplepolicy_controller.go** ファイルの **PeriodicallyExecSamplePolicies** 関数を、ポリシーコントローラーを実行する新しいロジックで更新します。**PeriodicallyExecSamplePolicies** フィールドの例については、stolostron/multicloud-operators-policy-controller を参照してください。

d. ポリシーコントローラーを再コンパイルし、実行します。ポリシーコントローラーの作成を参照してください。

ポリシーコントローラーが機能します。

2.3.4.2. コントローラーのクラスターへのデプロイ

カスタムポリシーコントローラーをクラスターにデプロイし、ポリシーコントローラーと **ガバナンス** ダッシュボードを統合します。以下の手順を実行します。

1. 次のコマンドを実行して、ポリシーコントローラーイメージをビルドします。

```
make build
docker build . -f build/Dockerfile -t <username>/multicloud-operators-policy-controller:latest
```

2. 以下のコマンドを実行して、イメージを選択したリポジトリーにプッシュします。たとえば、以下のコマンドを実行してイメージを Docker Hub にプッシュします。

```
docker login

docker push <username>/multicloud-operators-policy-controller
```

3. **kubectl** を、Red Hat Advanced Cluster Management for Kubernetes が管理するクラスターを参照するように設定します。

4. Operator マニフェストを置き換えて、ビルトインイメージ名を使用し、ポリシーを監視するように namespace を更新します。namespace はクラスターの namespace である必要があります。マニフェストは次のような内容になります。

```
sed -i "" 's|stolostron/multicloud-operators-policy-controller|ycao/multicloud-operators-policy-controller|g' deploy/operator.yaml
sed -i "" 's|value: default|value: <namespace>|g' deploy/operator.yaml
```

5. 以下のコマンドを実行して RBAC ロールを更新します。

```
sed -i "" 's|samplepolicies|testpolicies|g' deploy/cluster_role.yaml
sed -i "" 's|namespace: default|namespace: <namespace>|g'
deploy/cluster_role_binding.yaml
```

6. ポリシーコントローラーをクラスターにデプロイします。

   a. 以下のコマンドを実行して、クラスターのサービスアカウントを設定します。

```
kubectl apply -f deploy/service_account.yaml -n <namespace>
```

   b. 次のコマンドを実行して、Operator の RBAC を設定します。

```
kubectl apply -f deploy/role.yaml -n <namespace>

kubectl apply -f deploy/role_binding.yaml -n <namespace>
```

   c. ポリシーコントローラーの RBAC を設定します。以下のコマンドを実行します。

```
kubectl apply -f deploy/cluster_role.yaml
kubectl apply -f deploy/cluster_role_binding.yaml
```

d. 以下のコマンドを実行してカスタムリソース定義 (CRD) を設定します。

```
kubectl apply -f deploy/crds/policies.open-cluster-
management.io_samplepolicies_crd.yaml
```

e. 以下のコマンドを実行して **multicloud-operator-policy-controller** をデプロイします。

```
kubectl apply -f deploy/operator.yaml -n <namespace>
```

f. 以下のコマンドを実行して、コントローラーが機能していることを確認します。

```
kubectl get pod -n <namespace>
```

7. コントローラーが監視する **policy-template** を作成してポリシーコントローラーを統合する必要があります。詳細は、コンソールからのクラスターセキュリティーポリシーの作成 を参照してください。

2.3.4.2.1. コントローラーデプロイメントのスケーリング

ポリシーコントローラーデプロイメントでは削除がサポートされていません。デプロイメントをスケーリングして、デプロイメントが適用される Pod を更新することができます。以下の手順を実行します。

1. ターゲットのマネージドクラスターにログインします。
2. カスタムポリシーコントローラーのデプロイメントに移動します。
3. デプロイメントでスケーリングします。デプロイメントをゼロ Pod にスケーリングすると、ポリシーコントロールのデプロイメントが無効になります。

デプロイメントの詳細は、OpenShift Container Platform デプロイメント を参照してください。

ポリシーコントローラーがクラスターにデプロイされ、クラスターに統合されています。製品のポリシーコントローラーを表示します。詳細は、ポリシーコントローラー を参照してください。

2.4. 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

2.4.1. 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。

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。

他のポリシーの管理に関する詳細は、設定ポリシーの管理 を参照してください。セキュリティーフレームワークに関する他のトピックについては、ガバナンス を参照してください。

2.4.2. 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 for Kubernetes 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 for Kubernetes 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 v1.4.1 です。

2.4.2.1. 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 ワークフローは、Red Hat Advanced Cluster Management ポリシーを使用した Kubernetes リソースオブジェクトの OpenShift マネージドクラスターおよび Kubernetes クラスターへの分散を単純化します。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。

- 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 および Kyverno 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。

2.4.2.2. ポリシージェネレーター設定の設定

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。単純なポリシージェネレーター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
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。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
```

詳細は **policy-generator-plugin** リポジトリーを参照してください。

2.4.2.3. 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使用方法としては、1 つ以上のマネージドクラスターの Operator をインストールすること が挙げられます。以下の各種インストールモードの例と必須リソースを確認してください。

2.4.2.3.1. OpenShift GitOps をインストールするためのポリシー

以下の例は、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する方法を示しています。OpenShift GitOps Operator は **全名前空間** のインストールモード を提供します。まず、以下の例のように **openshift-gitops-subscription.yaml** という名前の **サブスクリプションマニフェスト** ファイルを作成する必要があります。

```
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
```

オペレーター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>** を追加できます。**<version>** を希望のバージョンに置き換えます。

次に、**policy-generator-config.yaml** というポリシージェネレーター設定ファイルが必要です。以下の例は、すべての OpenShift マネージドクラスターに OpenShift GitOps をインストールする単一のポリシーを示しています。

```
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
```

最後に必要となるファイルは **kustomization.yaml** ファイルです。**kustomization.yaml** ファイルには、以下の設定が必要です。

```
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
```

詳細は、Understanding OpenShift GitOps と Operator ドキュメントを参照してください。

2.4.2.3.2. コンプライアンス Operator をインストールするためのポリシー

コンプライアンス Operator などの **名前空間** を使用したインストールモードを使用する Operator の場合に、**OperatorGroup** マニフェストも必要になります。以下の例は、コンプライアンス Operator をインストールする生成されたポリシーを示しています。

まず、**Namespace**、**Subscription**、および **OperatorGroup** マニフェストを含めて、**compliance-operator.yaml** という名前の YAML ファイルを作成する必要があります。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/v1alpha1
kind: Subscription
metadata:
```

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

次に、**policy-generator-config.yaml** というポリシージェネレーター設定ファイルが必要です。以下の例は、すべての OpenShift マネージドクラスターにコンプライアンス Operator をインストールする単一のポリシーを示しています。

```
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
```

最後に必要となるファイルは **kustomization.yaml** ファイルです。**kustomization.yaml** ファイルに以下の設定が必要です。

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

この設定を追加すると、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-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
```

詳細は、コンプライアンス Operator のドキュメント を参照してください。

2.4.2.4. 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。

フィールド	説明
apiVersion	必須。この値は **policy.open-cluster-management.io/v1** に設定します。
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。パラメーターの値は **musthave**、**mustonlyhave** または **mustnothave** です。デフォルト値は **musthave** です。
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	必須。設定ファイルを一意に識別するために使用されます。
metadata.name	必須。ポリシーリソースを識別する名前。
placementBindingDefaults	必須。**PlacementBinding** で複数のポリシーを統合し、ジェネレーターが、定義された名前を使用して **PlacementBinding** の一意名を作成できるようにします。
placementBindingDefaults.name	任意。デフォルト値を使用するのではなく、明示的な配置バインディング名を設定するのがベストプラクティスです。
policyDefaults	必須。**namespace** 以外の policies 配列のエントリーでは、ここで一覧表示されるデフォルト値は上書きされます。

フィールド	説明
policyDefaults.categories	任意。**policy.open-cluster-management.io/categories** アノテーションで使用されるカテゴリーの配列。デフォルト値は **CM Configuration Management** です。
policyDefaults.controls	任意。**policy.open-cluster-management.io/controls** アノテーションで使用されるコントロールの配列。デフォルト値は **CM-2 Baseline Configuration** です。
policyDefaults.consolidateManifests	任意。これは、ポリシーでラップされるすべてのマニフェストに対して設定ポリシーを 1 つ生成するかどうかを決定します。**false** に設定すると、マニフェストごとの設定ポリシーが生成されます。デフォルト値は **true** です。
policyDefaults.informGatekeeperPolicies	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が決定されます。デフォルト値は **true** です。
policyDefaults.informKyvernoPolicies	任意。このポリシーが、Kyverno ポリシーマニフェストを参照すると、Kyverno ポリシーの違反時に Red Hat Advanced Cluster Management でポリシー違反を受け取るために、設定ポリシーを追加で生成する必要があるかどうかが決定されます。デフォルト値は **true** です。
policyDefaults.namespace	必須。すべてのポリシーの namespace。
policyDefaults.placement	任意。ポリシーの配置設定。このデフォルトは、すべてのクラスターに一致する配置設定になります。
placement.clusterSelectors	任意。クラスターセレクターを **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場合は、**placementRulePath** を参照してください。
placement.name	任意。同じクラスターセレ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。
placement.placementRulePath	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。新規 **Placement** を生成するには **clusterSelectors** を参照してください。

フィールド	説明
policyDefaults.remediationAction	任意。ポリシーの修復メカニズム。パラメーターの値は **enforce** および **inform** です。デフォルト値は **inform** です。
policyDefaults.severity	任意。ポリシー違反の重大度。デフォルト値は **low** です。
policyDefaults.standards	任意。**policy.open-cluster-management.io/standards** アノテーションで使用する標準の配列。デフォルト値は **NIST SP 800-53** です。
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一覧。
policies[ ].manifests	必須。ポリシーに追加する Kubernetes オブジェクトマニフェストの一覧。
policies[ ].name	必須。作成するポリシーの名前。
policies[ ].manifests[ ].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。パラメーターの値は **musthave**、**mustonlyhave** または **mustnothave** です。デフォルト値は **musthave** です。
policies[ ].manifests[ ].path	必須。**kustomization.yaml** ファイルに対する単一ファイルまたはフラットディレクトリーへのパス。
policies[ ].manifests[ ].patches	任意。パスのマニフェストに適用する Kustomize パッチ。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。

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。

Note: **ポリシー YAML** に既存のポリシーをコピーアンドペーストします。パラメーターフィールドの値は、既存のポリシーを貼り付けると自動的に入力されます。検索機能で、ポリシー YAML ファイルの内容も検索できます。

2.5.1. 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

ポリシー	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

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

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。

2.5.2. メモリー使用状況のポリシー

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

以下のセクションでは、メモリー使用状況ポリシーの設定について説明します。

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

2.5.2.1. メモリー使用状況ポリシー YAML の設定

メモリー使用状況ポリシーは、以下の YAML ファイルのようになります。

```
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  name: policy-limitrange
  namespace:
spec:
  complianceType:
  remediationAction:
  namespaces:
    exclude:
    include:
  object-templates:
    - complianceType:
      objectDefinition:
        apiVersion:
        kind:
        metadata:
          name:
        spec:
          limits:
          - default:
              memory:
```

```
        defaultRequest:
          memory:
        type:
    ...
```

2.5.2.2. メモリー使用状況のポリシーの表

フィールド	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.namespaces	任意。
spec.namespace	必須。ポリシーの適用先のハブクラスター内にある namespace。**include** には、ポリシーを適用する namespace のパラメーター値を入力します。**exclude** パラメーターには、ポリシーを明示的に適用しない namespace を指定します。**注記**: ポリシーコントローラーのオブジェクトテンプレートで指定される namespace は、対応する親ポリシーの namespace を上書きします。
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。
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.complianceType	必須。値は **"musthave"** に設定します。
spec.object-template	任意。マネージドクラスターを評価するか、マネージドクラスターに適用する必要がある他の Kubernetes オブジェクトを一覧表示するのに使用します。

2.5.2.3. メモリー使用状況ポリシーの例

ポリシーのサンプルを確認するには、**policy-limitmemory.yaml** を参照してください。詳細は、Pod セキュリティーポリシーの管理 を参照してください。設定コントローラーが監視す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。

2.5.3. Namespace ポリシー

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

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

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

2.5.3.1. Namespace ポリシー YAML の設定

```
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  name: policy-namespace-1
  namespace:
spec:
  complianceType:
  remediationAction:
  namespaces:
    exclude:
    include:
  object-templates:
    - complianceType:
      objectDefinition:
        kind:
        apiVersion:
        metadata:
          name:
    ...
```

2.5.3.2. Namespace ポリシー YAML の表

フィールド	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.namespaces	任意。

フィールド	説明
spec.namespace	必須。ポリシーの適用先のハブクラスター内にある namespace。**include** には、ポリシーを適用する namespace のパラメーター値を入力します。**exclude** パラメーターには、ポリシーを明示的に適用しない namespace を指定します。**注記**: ポリシーコントローラーのオブジェクトテンプレートで指定される namespace は、対応する親ポリシーの namespace を上書きします。
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。
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.complianceType	必須。値は **"musthave"** に設定します。
spec.object-template	任意。マネージドクラスターを評価するか、マネージドクラスターに適用する必要がある他の Kubernetes オブジェクトを一覧表示するのに使用します。

2.5.3.3. Namespace ポリシーの例

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

詳細は、Pod セキュリティーポリシーの管理 を参照してください。他の設定ポリシーの詳細は、Kubernetes 設定ポリシーコントローラー を参照してください。

2.5.4. イメージ脆弱性ポリシー

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

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

注記:

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

詳細は、以下のセクションを参照してください。

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

2.5.4.1. イメージ脆弱性ポリシーの YAML 設定

```
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  name: policy-imagemanifestvulnpolicy
  namespace: default
  annotations:
    policy.open-cluster-management.io/standards: NIST-CSF
    policy.open-cluster-management.io/categories: DE.CM Security Continuous Monitoring
    policy.open-cluster-management.io/controls: DE.CM-8 Vulnerability Scans
spec:
  remediationAction:
  disabled:
  policy-templates:
  - objectDefinition:
      apiVersion: policy.open-cluster-management.io/v1
      kind: ConfigurationPolicy
      metadata:
        name:
      spec:
        remediationAction:
        severity: high
        object-templates:
          - complianceType:
            objectDefinition:
              apiVersion: operators.coreos.com/v1alpha1
              kind: Subscription
              metadata:
                name: container-security-operator
                namespace:
              spec:
                channel:
                installPlanApproval:
                name:
                source:
                sourceNamespace:
  - objectDefinition:
      apiVersion: policy.open-cluster-management.io/v1
      kind: ConfigurationPolicy
      metadata:
        name:
      spec:
        remediationAction:
        severity:
        namespaceSelector:
          exclude:
          include:
        object-templates:
          - complianceType:
```

```
        objectDefinition:
          apiVersion: secscan.quay.redhat.com/v1alpha1
          kind: ImageManifestVuln # checking for a kind
---
apiVersion: policy.open-cluster-management.io/v1
kind: PlacementBinding
metadata:
  name: binding-policy-imagemanifestvulnpolicy
  namespace: default
placementRef:
  name:
  kind:
  apiGroup:
subjects:
- name:
  kind:
  apiGroup:
---
apiVersion: apps.open-cluster-management.io/v1
kind: PlacementRule
metadata:
  name: placement-policy-imagemanifestvulnpolicy
  namespace: default
spec:
  clusterConditions:
  - status:
    type:
  clusterSelector:
    matchExpressions:
      [] # selects all clusters if not specified
```

2.5.4.2. イメージ脆弱性ポリシー YAML の表

フィールド	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.namespaces	任意。

フィールド	説明
spec.namespace	必須。ポリシーの適用先のハブクラスター内にある namespace。**include** には、ポリシーを適用する namespace のパラメーター値を入力します。**exclude** パラメーターには、ポリシーを明示的に適用しない namespace を指定します。**注記**: ポリシーコントローラーのオブジェクトテンプレートで指定される namespace は、対応する親ポリシーの namespace を上書きします。
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。
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.complianceType	必須。値は **"musthave"** に設定します。
spec.object-template	任意。マネージドクラスターを評価するか、マネージドクラスターに適用する必要がある他の Kubernetes オブジェクトを一覧表示するのに使用します。

2.5.4.3. 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。

2.5.5. Pod ポリシー

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

以下のセクションでは、Pod ポリシーの設定について説明します。

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

2.5.5.1. Pod ポリシー YAML の設定

```
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  name: policy-pod
  namespace:
spec:
  complianceType:
  remediationAction:
  namespaces:
    exclude:
    include:
  object-templates:
    - complianceType:
      objectDefinition:
        apiVersion:
        kind: Pod # pod must exist
        metadata:
          name:
        spec:
          containers:
          - image:
            name:
            ports:
            - containerPort:
    ...
```

2.5.5.2. Pod ポリシーの表

フィールド	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.namespaces	任意。
spec.namespace	必須。ポリシーの適用先のハブクラスター内にある namespace。**include** には、ポリシーを適用する namespace のパラメーター値を入力します。**exclude** パラメーターには、ポリシーを明示的に適用しない namespace を指定します。**注記**: ポリシーコントローラーのオブジェクトテンプレートで指定される namespace は、対応する親ポリシーの namespace を上書きします。
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。

フィールド	説明
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.complianceType	必須。値は **"musthave"** に設定します。
spec.object-template	任意。マネージドクラスターを評価するか、マネージドクラスターに適用する必要がある他の Kubernetes オブジェクトを一覧表示するのに使用します。

2.5.5.3. Pod ポリシーの例

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

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。他のポリシーの管理については、設定ポリシーの管理 を参照してください。

2.5.6. Pod のセキュリティーポリシー

Kubernetes 設定ポリシーコントローラーは、Pod セキュリティーポリシーのステータスを監視します。Pod のセキュリティーポリシーを適用して Pod およびコンテナーのセキュリティーを保護します。詳細は、Kubernetes ドキュメント の **Pod Security Policies** を参照してください。

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

2.5.6.1. Pod セキュリティーポリシー YAML の設定

```
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  name: policy-podsecuritypolicy
  namespace:
spec:
  complianceType:
  remediationAction:
  namespaces:
    exclude:
    include:
  object-templates:
    - complianceType:
      objectDefinition:
        apiVersion:
        kind: PodSecurityPolicy # no privileged pods
```

```
    metadata:
      name:
      annotations:
    spec:
      privileged:
      allowPrivilegeEscalation:
      allowedCapabilities:
      volumes:
      hostNetwork:
      hostPorts:
      hostIPC:
      hostPID:
      runAsUser:
        rule:
      seLinux:
        rule:
      supplementalGroups:
        rule:
      fsGroup:
        rule:
  ...
```

2.5.6.2. Pod セキュリティーポリシーの表

フィールド	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.namespaces	任意。
spec.namespace	必須。ポリシーの適用先のハブクラスター内にある namespace。**include** には、ポリシーを適用する namespace のパラメーター値を入力します。**exclude** パラメーターには、ポリシーを明示的に適用しない namespace を指定します。**注記**: ポリシーコントローラーのオブジェクトテンプレートで指定される namespace は、対応する親ポリシーの namespace を上書きします。
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。
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	説明
spec.complianceType	必須。値は **"musthave"** に設定します。
spec.object-template	任意。マネージドクラスターを評価するか、マネージドクラスターに適用する必要がある他の Kubernetes オブジェクトを一覧表示するのに使用します。

2.5.6.3. Pod セキュリティーポリシーの例

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。

設定コントローラーによって監視されるその他の設定ポリシーについては、Kubernetes 設定ポリシーコントローラー を参照してください。

2.5.7. ロールポリシー

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

以下のセクションでは、ロールポリシーの設定について説明します。

- ロールポリシー YAML の設定
- ロールポリシーの表
- ロールポリシーの例

2.5.7.1. ロールポリシー YAML の設定

```
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  name: policy-role
  namespace:
  annotations:
    policy.open-cluster-management.io/standards: NIST-CSF
    policy.open-cluster-management.io/categories: PR.AC Identity Management Authentication and
Access Control
    policy.open-cluster-management.io/controls: PR.AC-4 Access Control
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 # will be overridden by remediationAction in parent policy
        severity: high
        namespaceSelector:
          exclude: ["kube-*"]
          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
  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:
      []

        ...
```

2.5.7.2. ロールポリシーの表

フィールド	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	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.namespaces	任意。
spec.namespace	必須。ポリシーの適用先のハブクラスター内にある namespace。**include** には、ポリシーを適用する namespace のパラメーター値を入力します。**exclude** パラメーターには、ポリシーを明示的に適用しない namespace を指定します。**注記**: ポリシーコントローラーのオブジェクトテンプレートで指定される namespace は、対応する親ポリシーの namespace を上書きします。
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。
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.complianceType	必須。値は **"musthave"** に設定します。
spec.object-template	任意。マネージドクラスターを評価するか、マネージドクラスターに適用する必要がある他の Kubernetes オブジェクトを一覧表示するのに使用します。

2.5.7.3. ロールポリシーの例

ロールポリシーを適用して、クラスター内の特定のロールのルールおよびパーミッションを設定します。

ロールの詳細は、ロールベースのアクセス制御 を参照してください。ロールポリシーのサンプルを確認するには **policy-role.yaml** を参照してください。

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

2.5.8. Role binding ポリシー

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

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

- Role Binding ポリシー YAML の設定

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

2.5.8.1. Role Binding ポリシー YAML の設定

```
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  name:
  namespace:
spec:
  complianceType:
  remediationAction:
  namespaces:
    exclude:
    include:
  object-templates:
    - complianceType:
      objectDefinition:
        kind: RoleBinding # role binding must exist
        apiVersion: rbac.authorization.k8s.io/v1
        metadata:
          name: operate-pods-rolebinding
        subjects:
        - kind: User
          name: admin # Name is case sensitive
          apiGroup:
        roleRef:
          kind: Role #this must be Role or ClusterRole
          name: operator # this must match the name of the Role or ClusterRole you wish to bind to
          apiGroup: rbac.authorization.k8s.io
    ...
```

2.5.8.2. Role Binding ポリシーの表

フィールド	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ための名前。
metadata.namespaces	必須。ポリシーが作成されるマネージドクラスター内の namespace。
spec	必須。コンプライアンス違反を特定して修正する方法の仕様。

フィールド	説明
metadata.name	必須。ポリシーリソースを識別する名前。
metadata.namespaces	任意。
spec.complianceType	必須。値は **"musthave"** に設定します。
spec.namespace	必須。ポリシーを適用するマネージドクラスターの namespace。**include** には、ポリシーを適用する namespace のパラメーター値を入力します。**exclude** パラメーターには、ポリシーを明示的に適用しない namespace を指定します。**注記**: ポリシーコントローラーのオブジェクトテンプレートで指定される namespace は、対応する親ポリシーの namespace を上書きします。
spec.remediationAction	必須。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。
spec.object-template	必須。マネージドクラスターを評価するか、マネージドクラスターに適用する必要がある他の Kubernetes オブジェクトを一覧表示するのに使用します。

2.5.8.3. Role Binding ポリシーの例

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

他の設定ポリシーの詳細は、Kubernetes 設定ポリシーコントローラー を参照してください。

2.5.9. SCC (Security Context Constraints) ポリシー

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

以下のセクションで、SCC ポリシーについての詳細を説明します。

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

2.5.9.1. SCC ポリシー YAML の設定

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

```
  namespace: open-cluster-management-policies
spec:
  complianceType:
  remediationAction:
  namespaces:
    exclude:
    include:
  object-templates:
    - complianceType:
      objectDefinition:
        apiVersion:
        kind: SecurityContextConstraints # restricted scc
        metadata:
          annotations:
            kubernetes.io/description:
          name: sample-restricted-scc
        allowHostDirVolumePlugin:
        allowHostIPC:
        allowHostNetwork:
        allowHostPID:
        allowHostPorts:
        allowPrivilegeEscalation:
        allowPrivilegedContainer:
        allowedCapabilities:
        defaultAddCapabilities:
        fsGroup:
         type:
        groups:
        - system:
        priority:
        readOnlyRootFilesystem:
        requiredDropCapabilities:
        runAsUser:
          type:
        seLinuxContext:
          type:
        supplementalGroups:
          type:
        users:
        volumes:
```

2.5.9.2. SCC ポリシーの表

フィールド	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ための名前。

フィールド	説明
metadata.namespace	必須。ポリシーが作成されるマネージドクラスター内の namespace。
spec.complianceType	必須。値は **"musthave"** に設定します。
spec.remediationAction	必須。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。**重要**: ポリシーによっては、enforce 機能をサポートしない場合があります。
spec.namespace	必須。ポリシーを適用するマネージドクラスターの namespace。**include** には、ポリシーを適用する namespace のパラメーター値を入力します。**exclude** パラメーターには、ポリシーを明示的に適用しない namespace を指定します。**注記**: ポリシーコントローラーのオブジェクトテンプレートで指定される namespace は、対応する親ポリシーの namespace を上書きします。
spec.object-template	必須。マネージドクラスターを評価するか、マネージドクラスターに適用する必要がある他の Kubernetes オブジェクトを一覧表示するのに使用します。

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。

2.5.9.3. SCC ポリシーの例

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

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

他の設定ポリシーの詳細は、Kubernetes 設定ポリシーコントローラー を参照してください。

2.5.10. ETCD 暗号化ポリシー

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

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

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

- etcd 暗号化ポリシーの例

2.5.10.1. ETCD 暗号化ポリシーの YAML 設定

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

```
apiVersion: policy.open-cluster-management.io/v1
kind: ConfigurationPolicy
metadata:
  name: policy-etcdencryption
  namespace:
spec:
  complianceType:
  remediationAction:
  namespaces:
    exclude:
    include:
  object-templates:
    - complianceType:
      objectDefinition:
        apiVersion: config.openshift.io/v1
        kind: APIServer
        metadata:
          name: cluster
        spec:
          encryption:
            type:
        ...
```

2.5.10.2. ETCD 暗号化ポリシーの表

表2.4 パラメーターの表

フィールド	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 **Policy** に設定して **ConfigurationPolicy** などのポリシーの種類を指定します。
metadata.name	必須。ポリシーリソースを識別する名前。
metadata.namespaces	任意。

フィールド	説明
spec.namespace	必須。ポリシーの適用先のハブクラスター内にある namespace。**include** には、ポリシーを適用する namespace のパラメーター値を入力します。**exclude** パラメーターには、ポリシーを明示的に適用しない namespace を指定します。**注記**: ポリシーコントローラーのオブジェクトテンプレートで指定される namespace は、対応する親ポリシーの namespace を上書きします。
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。**重要**: ポリシーによっては、enforce 機能をサポートしない場合があります。
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.complianceType	必須。値は **"musthave"** に設定します。
spec.object-template	任意。マネージドクラスターを評価するか、マネージドクラスターに適用する必要がある他の Kubernetes オブジェクトを一覧表示するのに使用します。OpenShift Container Platform ドキュメントの etcd データの暗号化 を参照してください。

2.5.10.3. etcd 暗号化ポリシーの例

ポリシーのサンプルについては、**policy-etcdencryption.yaml** を参照してください。詳細は、セキュリティーポリシーの管理 を参照してください。

設定コントローラーによって監視されるその他の設定ポリシーについては、Kubernetes 設定ポリシーコントローラー を参照してください。

2.5.11. 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。

2.5.11.1. 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
```

- 名前とチャネルを参照するためのサブスクリプション (**comp-operator-subscription**)。サブスクリプションは、サポートするプロファイルをコンテナーとしてプルします。

```
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 ポリシー管理の詳細は、の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。

2.5.12. E8 スキャンポリシー

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

2.5.12.1. E8 スキャンポリシーリソース

E8 スキャンポリシーを作成すると、次のリソースが作成されます。

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

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

```
    namespace: openshift-compliance
  profiles:
  - apiGroup: compliance.openshift.io/v1alpha1
    kind: Profile
    name: ocp4-e8
  - apiGroup: compliance.openshift.io/v1alpha1
    kind: Profile
    name: rhcos4-e8
  settingsRef:
    apiGroup: compliance.openshift.io/v1alpha1
    kind: ScanSetting
    name: default
```

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

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

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

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

```
    labels:
      compliance.openshift.io/check-status: FAIL
      compliance.openshift.io/suite: e8
```

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

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

2.5.13. OpenShift CIS スキャンポリシー

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

2.5.13.1. OpenShift CIS リソース

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

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

```
apiVersion: policy.open-cluster-management.io/v1
kind: ConfigurationPolicy
metadata:
  name: compliance-cis-scan
spec:
  remediationAction: inform
  severity: high
  object-templates:
    - complianceType: musthave # this template creates ScanSettingBinding:cis
      objectDefinition:
        apiVersion: compliance.openshift.io/v1alpha1
        kind: ScanSettingBinding
        metadata:
          name: cis
          namespace: openshift-compliance
        profiles:
        - apiGroup: compliance.openshift.io/v1alpha1
          kind: Profile
          name: ocp4-cis
        - apiGroup: compliance.openshift.io/v1alpha1
          kind: Profile
          name: ocp4-cis-node
        settingsRef:
          apiGroup: compliance.openshift.io/v1alpha1
          kind: ScanSetting
          name: default
```

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

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

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

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

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

2.6. セキュリティーポリシーの管理

セキュリティーポリシー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. ガバナンスページのカスタマイズ

ガバナンス ページでは、カテゴリーや基準で違反をフィルターリングして**概要**ビューをカスタマイズしたり、概要ビューを折りたたみ表示数を減らしたりだけでなく、ポリシーの検索も可能です。ポリシーまたはクラスターの違反別に、違反の表のビューもフィルターリングできます。

以下のフィルターオプションを使用して、ビューのカスタマイズを継続します。

- 違反 (以下のオプションは、1つ以上のポリシーが基準を満たしている場合にのみ表示されます):
  - 違反なし
  - 違反
  - **-**
- ソース (以下のオプションは、1つ以上のポリシーが基準を満たしている場合にのみ表示されます):
  - Local
  - 外部
  - Git
- 修復 (以下のオプションは常に表示され、一括操作をサポートします)。
  - 通知
  - 実行
- ステータス (以下のオプションは常に表示され、一括操作をサポートします)。
  - 有効
  - 無効

ポリシーの表では、**Policy name**、**Namespace**、**Status**、**Remediation**、**Cluster violations**、**Source**、**Controls**、**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ーおよび標準を表示するには、ドロップダウン矢印を選択して行を展開します。

Automation の列の頻度フィールド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。
- **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一覧が自動的に指定されます。Ansible Tower ジョブテンプレートの **EXTRA VARIABLES** セクションで、**PROMPT ON LAUNCH** を有効にする必要があります。
- **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。

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。

- **Clusters**: **Clusters** タブを選択して、配置に関連付けられたすべてのクラスターの表一覧を表示します。**View details** リンクをクリックして、テンプレートの詳細と YAML 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、コンプライアンスのステータス、違反メッセージ、および最後のレポート時間を表示します。

- **Templates**: **Templates** タブを選択して、各テンプレートの配置に関連付けられたクラスターのテーブル一覧を表示します。**View history** リンクを選択すると、コンプライアンスのステータス、違反メッセージ、最後のレポートの時間、およびテンプレートの履歴を表示できます。

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

- セキュリティーポリシーの管理
- 設定ポリシーの管理
- gatekeeper ポリシーの管理
- Ansible Tower でのガバナンスの設定

他のトピックについては、ガバナンス を参照してください。

2.6.2. Ansible Tower でのガバナンスの設定

Red Hat Advanced Cluster Management for Kubernetes ガバナンスは、Ansible Tower の自動化と統合して、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の自動化の作成
- CLI からのポリシー違反の自動化の作成

2.6.2.1. 前提条件

- Red Hat OpenShift Container Platform 4.5 以降
- Ansible Tower バージョン 3.7.3 以降がインストールされている。Ansible Tower の最新のサポートバージョンをインストールすることがベストプラクティスです。詳細は、Red Hat AnsibleTower ドキュメント を参照。
- ハブクラスターに Ansible Automation Platform Resource Operator をインストールして、Ansible ジョブをガバナンスフレームワークに接続する。AnsibleJob を使用した Ansible Tower ジョブの実行時に最善の結果を得るには、実行時に Ansible Tower ジョブテンプレートが冪等でなければなりません。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。

Ansible Tower 自動化のインストールおよび設定に関する詳細は、Ansible タスクの設定 (テクノロジープレビュー) を参照してください。

2.6.2.2. コンソールからのポリシー違反の自動化の作成

Red Hat Advanced Cluster Management ハブクラスターにログインしたら、ナビゲーションメニューから **Governance** を選択します。

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。**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、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。

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

- **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。
- **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一覧が自動的に指定されます。Ansible Tower ジョブテンプレートの **EXTRA VARIABLES** セクションで、**PROMPT ON LAUNCH** を有効にする必要があります。
- **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。

Save を選択して、ポリシー違反の自動化を保存します。**History** タブから **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。

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。

2.6.2.3. CLI からのポリシー違反の自動化の作成

CLI からポリシー違反の自動化を設定するには、以下の手順を実行します。

1. ターミナルから、**oc login** コマンドを使用して Red Hat Advanced Cluster Management ハブクラスターに再度ログインします。
2. 自動化を追加するポリシーを検索するか、または作成します。ポリシー名と namespace をメモします。
3. 以下のサンプルをガイドとして使用して、**Policy Automation** リソースを作成します。

```
apiVersion: policy.open-cluster-management.io/v1beta1
kind: PolicyAutomation
metadata:
  name: policyname-policy-automation
spec:
  automationDef:
    extra_vars:
      your_var: your_value
    name: Policy Compliance Template
    secret: ansible-tower
```

```
  type: AnsibleJob
mode: disabled
policyRef: policyname
```

4. 先のサンプルの Ansible ジョブテンプレート名は **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、Ansible ジョブ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** または **disabled** のいずれかに設定します。**once** モードはジョブを 1 回実行し、モードを **disabled** に設定します。

   - **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一覧が自動的に指定されます。Ansible Tower ジョブテンプレートの **EXTRA VARIABLES** セクションで、**PROMPT ON LAUNCH** を有効にする必要があります。

   - **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。

7. **policyRef** は、ポリシーの名前に設定します。

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

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

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

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

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

2.6.3. 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。リポジトリー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ー (**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ーの各 **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 リポジトリー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** リポジトリーをフォークしてください。

- ローカルリポジトリーのカスタマイズ
- ローカルリポジトリーへのコミット
- 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

2.6.3.1. ローカルリポジトリーのカスタマイズ

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ー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ーをカスタマイズするには、以下の手順を実行します。

1. リポジトリー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** リポジトリー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ー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。
   - 以下の一覧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: ポリシーのロールを理解する。
     - 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。

2.6.3.2. ローカルリポジトリーへのコミット

ディレクトリーに行った変更に問題がなければ、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、コミットする変更一覧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リポジトリーのブランチに、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。

2.6.3.3. 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 リポジトリーに通知されます。Git リポジトリー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** 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。チャネルは Git リポジトリーに接続し、サブスクリプションは、チャ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 リポジトリーのポリシーは、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
```

2. 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。

3. 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
```

4. 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ー名に置き換えます。

注記: 参考までに、**deploy.sh** スクリプトの引数の全一覧では、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** リポジトリーからフォークしたリポジトリーへの 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 リポジトリーのブランチ。デフォルトのブランチは **main** です。
- サブディレクトリーパス: 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ーパス。上記のサンプルでは **my-policies** サブディレクトリーを使用しましたが、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 **policy-namespace** 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、リポジトリー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。

2.6.3.4. 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 リポジトリーと調整すると、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 リポジトリーの詳細を表示します。

2.6.3.4.1. 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 policy <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 policy -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。

2.6.4. 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
  - テンプレート処理のバイパス

2.6.4.1. 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。

2.6.4.2. テンプレート関数

リソース固有および一般的な **lookup** テンプレート関数などのテンプレート関数は、クラスター上の Kubernetes リソースを参照するために使用できます。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、**base64encode**、**base64decode**、**indent**、**autoindent**、**toInt**、**toBool** などのユーティリティー関数も利用できます。

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

2.6.4.2.1. 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。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
```

2.6.4.2.2. 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。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
```

2.6.4.2.3. 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) (value map[string]interface{}, err Error)
```

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
```

2.6.4.2.4. lookup 関数

lookup 関数は、JSON と互換性のあるマップとして Kubernetes リソースを返します。要求されたリソースが存在しない場合は、空のマップが返されることに注意してください。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)
```

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。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
```

2.6.4.2.5. base64enc 関数

base64enc 関数は、入力 **data string** 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)
```

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 }}'
```

2.6.4.2.6. base64dec 関数

base64dec 関数は、入力 **enc-data string** 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 }}"
```

2.6.4.2.7. indent 関数

indent 関数により、パディングされた **data string** 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)
```

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
}}
```

2.6.4.2.8. 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 }}
```

_

2.6.4.2.9. 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 }}
```

2.6.4.2.10. toBool 関数

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

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

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

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

```
- complianceType: musthave
  objectDefinition:
  ...
  spec:
    enabled: |
      {{ (fromConfigMap "site-config" "site1" "enabled")  | toBool }}
```

2.6.4.3. 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。

セキュリティーを確保するには、ハブクラスターテンプレートのリソース固有および一般的なルックアップ機能の両方が、ハブクラスターのポリシーの namespace に制限されます。

重要: ハブクラスターテンプレートを使用して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。

2.6.4.3.1. 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。

2.6.4.3.2. 再処理のための特別なアノテーション

ポリシーは、作成または更新時にのみハブクラスターで処理されます。そのため、ハブクラスターテンプレートは、ポリシーの作成または更新時に参照リソースのデータに対してのみ解決されます。参照リソースへの変更は、自動的にポリシーと同期されません。

テンプレートによって参照されるデータへの変更を示すために、特別なアノテーション **policy.open-cluster-management.io/trigger-update** を使用できます。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開始され、参照されるリソースの最新内容は、ポリシー定義に読み込まれて更新され、マネージクラスターので処理するように伝播するロールを果たします。このアノテーションの一般的な使用方法は、値を 1 回に 1 つずつ増やすことです。

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

表2.5 ハブクラスターとマネージドクラスターの比較表

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

テンプレート	ハブクラスター	マネージドクラスター
デリミタ	{{hub ... hub}}	{{ ... }}
関数	Kubernetes リソースおよび文字列操作への動的なアクセスをサポートするテンプレート関数のセット。詳細は、テンプレート関数 を参照してください。**注記:** **fromSecret** テンプレート関数は利用できません。	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。
関数出力ストレージ	テンプレート関数の出力は、マネージドクラスターに同期される前に、マネージドクラスターで適用可能な各マネージドクラスター namespace の **Policy** resource オブジェクトに保存されます。つまり、テンプレート関数からの結果は機密な内容であっても、ハブクラスター上の **Policy** リソースオブジェクトや、マネージドクラスター上の **ConfigurationPolicy** リソースオブジェクトへの読み取り権限がある全ユーザーによる読み取りが可能です。さらに、etcd 暗号化 が有効な場合には、**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。	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。
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
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。

テンプレート	ハブクラスター	マネージドクラスター
アクセス制御	**Policy** リソースと同じ namespace にある Kubernetes リソース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。
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。

2.6.4.3.3. テンプレート処理のバイパス

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

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

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

2.6.5. ガバナンスメトリクス

ポリシーフレームワークは、ポリシーディストリビューションとコンプライアンスを表示するメトリクスを公開します。ハブクラスターで **policy_governance_info** メトリクス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。

2.6.5.1. メトリックの概要

policy_governance_info は OpenShift Container Platform モニターリングが、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** ページからメトリクス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** のメトリクスが、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 メトリクスは、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ー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** メトリクスを、可観測性の許可リストにカスタムメトリクスとして追加することもできます。詳細は、カスタムメトリクスの追加 を参照してください。

2.6.6. セキュリティーポリシーの管理

セキュリティーポリシーを作成して、指定のセキュリティー標準、カテゴリー、制御をもとにクラスターのコンプライアンスを報告して検証します。Red Hat Advanced Cluster Management for Kubernetes のポリシーを作成するには、マネージドクラスターで YAML ファイルを作成する必要があります。

Note: **ポリシー YAML** に既存のポリシーをコピーアンドペーストします。パラメーターフィールドの値は、既存のポリシーを貼り付けると自動的に入力されます。検索機能で、ポリシー YAML ファイルの内容も検索できます。

以下のセクションを参照してください。

- セキュリティーポリシーの作成
  - 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
  - CLI からのセキュリティーポリシーの表示
  - コンソールからのクラスターセキュリティーポリシーの作成
  - コンソールからのセキュリティーポリシーの表示
- セキュリティーポリシーの更新
  - セキュリティーポリシーの無効化
- セキュリティーポリシーの削除

2.6.6.1. セキュリティーポリシーの作成

コマンドラインインターフェイス (CLI) またはコンソールからセキュリティーポリシーを作成できます。クラスター管理者のアクセス権限が必要です。

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のある **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 <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、**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
  namespaces:
    include: ["default"]
    exclude: ["kube*"]
  policy-templates:
    - objectDefinition:
        apiVersion: policy.open-cluster-management.io/v1
        kind: ConfigurationPolicy
        metadata:
          namespace: kube-system # will be inferred
          name: operator
        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。**PlacementRule** を変更して、**clusterNames** または **clusterLabels** で、ポリシーを適用する必要のあるクラスターを指定します。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"
  clusterLabels:
    matchLabels:
      cloud: IBM
```

4. **PlacementBinding** を定義して、ポリシーと **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 securitypolicy <policy-name> -n <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 securitypolicy <name> -n <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ーポリシーの作成

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。

Governance ページに移動し、**Create policy** をクリックします。

必要なポリシーを入力するか、または選択します。ポリシーを選択すると、パラメーターの値が自動的に入力されます。

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"]
  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:
    - type: ManagedClusterConditionAvailable
      status: "True"
  clusterLabels:
    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ーリングするには、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。

2.6.6.2. 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. 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、ポリシー表の一覧を表示します。

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ーポリシーを削除します。

- CLI からセキュリティーポリシーを削除します。

  a. 以下のコマンドを実行してセキュリティーポリシーを削除します。

  ```
  kubectl delete policy <securitypolicy-name> -n <open-cluster-management-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。**kubectl get policy <securitypolicy-name> -n <open-cluster-management-namespace>** 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。

- コンソールからセキュリティーポリシーを削除します。
  ナビゲーションメニューから Governance をクリックし、ポリシー表の一覧を表示します。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。

他のポリシーの管理については、セキュリティーポリシーの管理 を参照してください。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。

2.6.7. 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。なお、以下のリソースには設定ポリシーがあります。

- メモリー使用状況のポリシー
- Namespace ポリシー

- イメージ脆弱性ポリシー
- 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

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。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。

```
kubectl 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: kube-system
spec:
  namespaces:
    include: ["default", "kube-*"]
```

```
    exclude: ["kube-system"]
  remediationAction: inform
  disabled: false
  complianceType: musthave
  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。

```
kubectl 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一覧を確認します。

```
kubectl get policy --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。

```
kubectl get policy <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。

```
kubectl describe policy <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。

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。

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。

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 (名前)
- Specifications
- Cluster selector
- Remediation action
- Standards
- Categories
- Controls

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ーリングするには、**All policies** タブまたは **Cluster violations** タブを選択します。

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

2.6.7.2. 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、**ガバナンス** ページに移動します。

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。

Disable policy をクリックします。

設定ポリシーが無効になっています。

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。

- CLI から設定ポリシーを削除します。
  a. 以下のコマンドを実行して設定ポリシーを削除します。

  ```
  kubectl delete policy <policy-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 policy <policy-name> -n <namespace>
  ```

- コンソールから設定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、ポリシー表の一覧を表示します。

  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。次に、**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。

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、コントローラーが監視す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。他のポリシーの管理については、セキュリティーポリシーの管理 を参照してください。

2.6.8. 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 CR
- 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。

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 **GatekeeperOperator** 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。サンプルを表示するには、**policy-gatekeeper-operator.yaml** を参照してください。

+ **注記:** デフォルト値が Operator によって生成可能であることに留意してください。gatekeeper Operator ポリシーに使用できるオプションのパラメーターの説明については、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。以下の手順を実行します。

- 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ーリングするには、**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、ポリシー表の一覧を表示します。

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

2.7. ハブクラスターのセキュリティー保護

ハブクラスターセキュリティーを強化し、Red Hat Advanced Cluster Management for Kubernetes インストールのセキュリティーを保護します。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。

2.8. 整合性シールド保護 (テクノロジープレビュー)

整合性シールドは、整合性管理をサポートするツールで、リソースの作成または更新要求に対する署名検証を有効にします。整合性シールドは Open Policy Agent (OPA) および Gatekeeper をサポートし、要求に署名があるかどうかを検証して、定義した制約に従って不正な要求をブロックします。

以下の整合性シールド機能を参照してください。

- 承認された Kubernetes マニフェストのデプロイメントのみをサポートします。
- リソースが許可リストに追加されていない限り、リソース設定のゼロドリフトをサポートします。
- 受付コントローラーの実施など、クラスターで全整合性の検証を実行します。
- リソースを継続的に監視して、不正な Kubernetes リソースがクラスターにデプロイされているかどうかを報告します。
- Kubernetes マニフェスト YAML ファイルの署名には、X509、GPG、および Sigstore の署名がサポートされます。Kubernetes 整合性シールドは、**k8s-manifest-sigstore** を使用して署名した Sigstore をサポートします。

2.8.1. 整合性シールドアーキテクチャー

整合性シールドは、API と Observer の 2 つの主要なコンポーネントで設定されます。整合性シールド Operator は、クラスター上の整合性シールドコンポーネントのインストールおよび管理をサポートします。以下のコンポーネントの説明を確認してください。

- **整合性シールド API** は OPA または gatekeeper から Kubernetes リソースを受信し、受付要求に含まれるリソースを検証して検証結果を OPA または gatekeeper に送信します。整合性シールド API は **k8s-manifest-sigstore** の **verify-resource** 機能を使用して、Kubernetes マニフェスト YAML ファイルを検証します。整合性シールド API は、**ManifestingIntegrityConstraint** (OPA または gatekeeper の制約フレームワークをベースとするカスタムリソース) に従ってリソースを検証します。
- **整合性シールドオブザーバー** は、**ManifestingIntegrityConstraint** リソースに合わせてクラスター上の Kubernetes りそーすを継続的に検証し、**ManifestIntegrityState** と呼ばれるリソースに結果をエクスポートします。整合性シールドオブザーバーも **k8s-manifest-sigstore** を使用して署名を検証します。

2.8.2. サポート対象バージョン

以下の製品バージョンは、整合性シールドの保護をサポートします。

- Red Hat OpenShift Container Platform 4.7.1 以降

- Kubernetes v1.19.7 以降
- **gatekeeper-operator.v-.2.0**
- **gatekeeper v3.5**

詳細は Enable integrity shield protection (Technology preview) を参照してください。

2.8.3. 整合性シールド保護の有効化 (テクノロジープレビュー)

Red Hat Advanced Cluster Management for Kubernetes クラスターで整合性シールド保護を有効にして、Kubernetes リソースの整合性を保護します。

2.8.3.1. 前提条件

Red Hat Advanced Cluster Management マネージドクラスターで整合性シールド保護を有効にするには、以下の前提条件を満たす必要がある。

- マネージドクラスターが含まれる Red Hat Advanced Cluster Management ハブクラスターをインストールしており、そのクラスターに対して **oc** または **kubectl** コマンドを使用するためのクラスター管理者権限がある。
- 整合性シールドをインストールする。整合性シールドをインストールする前に、Open Policy Agent または gatekeeper をクラスターにインストールする必要がある。整合性シールド Operator をインストールするには、以下の手順を実行する。
  a. 以下のコマンドを実行して、整合性シールドの namespace に整合性シールド Operator をインストールする。

```
kubectl create -f https://raw.githubusercontent.com/open-cluster-management/integrity-shield/master/integrity-shield-operator/deploy/integrity-shield-operator-latest.yaml
```

  b. 以下のコマンドを使用して、整合性シールドカスタムリソースをインストールする。

```
kubectl create -f https://raw.githubusercontent.com/open-cluster-management/integrity-shield/master/integrity-shield-operator/config/samples/apis_v1_integrityshield.yaml -n integrity-shield-operator-system
```

  c. 整合性シールドには、クラスターで保護する必要があるリソースの署名および検証用の鍵のペアが必要である。署名と検証キーペアを設定する。
    - 以下のコマンドを使用して新規の GPG キーを生成する。

```
gpg --full-generate-key
```

    - 以下のコマンドを使用して、新しい GPG 公開鍵をファイルにエクスポートする。

```
gpg --export signer@enterprise.com > /tmp/pubring.gpg
```

- **yq** をインストールして、Red Hat Advanced Cluster Management ポリシーに署名するスクリプトを実行する。
- Integrity-shield 保護を有効にし、Red Hat Advanced Cluster Management に署名することで、**integrity-shield** リポジトリーからのソースの取得およびコミットが含まれる。Git をインストールする必要がある。

2.8.3.2. 整合性シールド保護の有効化

Red Hat Advanced Cluster Management マネージドクラスターで整合性シールドを有効にするには、以下の手順を実行します。

1. ハブクラスターに整合性シールド用の namespace を作成します。以下のコマンドを実行します。

```
oc create ns your-integrity-shield-ns
```

2. 検証キーを Red Hat Advanced Cluster Management マネージドクラスターにデプロイします。なお、署名キーおよび検証キーを作成する必要があります。ハブクラスターで **acm-verification-key-setup.sh** を実行して検証キーを設定します。以下のコマンドを実行します。

```
curl -s  https://raw.githubusercontent.com/stolostron/integrity-shield/master/scripts/ACM/acm-verification-key-setup.sh | bash -s \
        --namespace integrity-shield-operator-system  \
        --secret keyring-secret  \
        --path /tmp/pubring.gpg \
        --label environment=dev  |  oc apply -f -
```

検証キーを削除するには、以下のコマンドを実行します。

```
curl -s  https://raw.githubusercontent.com/stolostron/integrity-shield/master/scripts/ACM/acm-verification-key-setup.sh | bash -s - \
        --namespace integrity-shield-operator-system  \
        --secret keyring-secret  \
        --path /tmp/pubring.gpg \
        --label environment=dev  |  oc delete -f -
```

3. ハブクラスターに **policy-integrity-shield** という名前の Red Hat Advanced Cluster Management ポリシーを作成します。
   a. **policy-collection** リポジトリーから **policy-integrity-shield** ポリシーを取得します。リポジトリーをフォークしてください。
   b. **remediationAction** パラメーターの値を **inform** から **enforce** に更新して、Red Hat Advanced Cluster Management マネージドクラスターに整合性シールドをデプロイするように namespace を設定します。
   c. **signerConfig** セクションを更新して、署名および検証キーのメールを設定します。
   d. 整合性シールドをデプロイする Red Hat Advanced Cluster Management マネージドクラスターを決定する **PlacementRule** を設定します。
   e. 以下のコマンドを実行して、**policy-integrity-shield.yaml** を署名します。

```
curl -s  https://raw.githubusercontent.com/stolostron/integrity-shield/master/scripts/gpg-annotation-sign.sh | bash -s \
        signer@enterprise.com \
        policy-integrity-shield.yaml
```

注記: ポリシーを変更し、他のクラスターに適用する場合は、常に新規署名を作成する必要があります。そうでない場合は、変更はブロックされ、適用されません。

サンプルについては、**policy-integrity-shield** ポリシーを参照してください。