Red Hat Advanced Cluster Management for Kubernetes 2.1

セキュリティー

セキュリティー

Last Updated: 2022-01-31

Red Hat Advanced Cluster Management for Kubernetes 2.1 セキュリティー

セキュリティー

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

Red Hat Advanced Cluster Management for Kubernetes のセキュリティーおよびガバナンス

目次

第1章 セキュリティー

Red Hat Advanced Cluster Management for Kubernetes コンポーネントのセキュリティーおよびロールベースのアクセス制御 (RBAC) 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、制御の設定を行います。

前提条件: Identity and Access Management (IAM) を識別するオンボードワークロードに合わせて Red Hat Advanced Cluster Management for Kubernetes の認証サービス要件を設定する必要があります。詳細は、OpenShift Container Platform ドキュメント の「**認証について**」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、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
- 証明書
- ガバナンスおよびリスク

1.1. ロールベースのアクセス制御

Red Hat Advanced Cluster Management for Kubernetes は、ロールベースのアクセス制御 (RBAC) をサポートします。ロールによって実行できるアクションが決まります。RBAC は、Red Hat OpenShift Container Platform と同様に Kubernetes の承認メカニズムに基づいています。RBAC の詳細は、OpenShift Container Platform ドキュメント の「**RBAC**の概要」を参照してください。

注記: ユーザーロールのアクセス権がない場合には、コンソールのアクションボタンが無効になります。

コンポーネントでサポートされる RBAC の詳細については、以下のセクションを参照してください。

- ロールの概要
- RBAC 実装
- クラスターライフサイクル RBAC
- アプリケーションライフサイクル RBAC
- ガバナンスライフサイクル RBAC
- 可観測性の RBAC

1.1.1. ロールの概要

クラスター別の製品リソースと、namespace がスコープの製品リソースがあります。アクセス制御に一貫性を持たせるため、クラスターのロールバインディングと、namespace のロールバインディングをユーザーに適用する必要があります。Red Hat Advanced Cluster Management for Kubernetes でサポートされている以下のロール定義の表を参照してください。

表1.1 ロール定義の表

ロール	定義

ロール	定義
cluster-admin	**cluster-admin** ロールへのクラスター全体のバインディングを持つユーザーは、すべてのアクセスを持つ OpenShift Container Platform スーパーユーザーです。
open-cluster-management:cluster-manager-admin	**cluster-manager-admin** ロールへのクラスター全体のバインディングを持つユーザーは、すべてのアクセスを持つ Red Hat Advanced Cluster Management for Kubernetes のスーパーユーザーです。
open-cluster-management:managed-cluster-x (admin)	**managed-cluster-x** ロールへのクラスターバインディングを持つユーザーには、**managedcluster "X"** リソースへの管理者アクセスが付与されます。
open-cluster-management:managed-cluster-x (viewer)	**managed-cluster-x** ロールへのクラスター全体のバインディングを持つユーザーには、**managedcluster "X"** リソースへのビューアクセスが付与されます。
open-cluster-management:subscription-admin	**subscription-admin** ロールを持つユーザーは、リソースを複数の namespace にデプロイする Git サブスクリプションを作成できます。リソースは、サブスクライブされた Git リポジトリーの Kubernetes リソース YAML ファイルで指定されます。**注記**:subscription-admin ユーザーがサブスクリプションを作成すると、リソースに指定された namespace に関係なく、すべてのリソースがサブスクリプションの namespace にデプロイされます。詳細は、「アプリケーションライフサイクル RBAC」のセクションを参照してください。
admin、edit、view	admin、edit、および view は OpenShift Container Platform のデフォルトロールです。これらのロールに対して namespace に限定されたバインディングが指定されているユーザーは、特定の namespace 内の **open-cluster-management** リソースにアクセスでき、同じロールに対してクラスター全体のバインディングが指定されている場合には、クラスター全体の **open-cluster-management** リソースすべてにアクセス権があります。

重要:

- ユーザーは OpenShift Container Platform からプロジェクトを作成できます。これにより、namespace の管理者ロールパーミッションが付与されます。
- ユーザーにクラスターへのロールアクセスがない場合には、クラスター名は表示されません。クラスター名は、**-** の記号で表示されます。

1.1.2. RBAC 実装

RBAC はコンソールレベルと API レベルで検証されます。コンソール内のアクションは、ユーザーのアクセスロールのパーミッションに基づいて有効化/無効化できます。製品の特定ライフサイクルの RBAC の詳細は、以下のセクションを参照してください。

1.1.2.1. クラスターライフサイクル RBAC

以下のクラスターライフサイクル RBAC 操作を確認してください。

全マネージドクラスターを作成して管理する方法:

- 以下のコマンドを入力して、クラスターロール **open-cluster-management:cluster-manager-admin** にバインドするクラスターロールを作成します。

  ```
  oc create clusterrolebinding <role-binding-name> --clusterrole=open-cluster-management:cluster-manager-admin
  ```

  このロールはスーパーユーザーであるため、すべてのリソースとアクションにアクセスできます。このロールを使用すると、クラスターレベルの **managedcluster** リソース、マネージドクラスターを管理するリソースの namespace、namespace 内のリソースを作成できます。このロールで、プロバイダー接続、マネージドクラスター作成に使用するベアメタルアセットにアクセスできます。

cluster-name という名前のマネージドクラスターを管理する方法:

- 以下のコマンドを入力して、クラスターロール **open-cluster-management:admin:<cluster-name>** にバインドするクラスターロールを作成します。

  ```
  oc create clusterrolebinding (role-binding-name) --clusterrole=open-cluster-management:admin:<cluster-name>
  ```

  このロールを使用すると、クラスターレベルの **managedcluster** リソースに読み取り/書き込みアクセスができるようになります。**managedcluster** はクラスターレベルのリソースで、namespace レベルのリソースではないので、このロールが必要です。

- 以下のコマンドを入力して、クラスターロール **admin** にバインドする namespace ロールを作成します。

  ```
  oc create rolebinding <role-binding-name> -n <cluster-name> --clusterrole=admin
  ```

  このロールにより、マネージドクラスターの namespace 内のリソースへの読み取りおよび書き込みアクセスが可能になります。

cluster-name という名前のマネージドクラスターを表示する方法:

- 以下のコマンドを入力して、クラスターロール **open-cluster-management:view:<cluster-name>** にバインドするクラスターロールを作成します。

  ```
  oc create clusterrolebinding <role-binding-name> --clusterrole=open-cluster-management:view:<cluster-name>
  ```

  このロールを使用すると、クラスターレベルの **managedcluster** リソースに読み取りアクセスができるようになります。**managedcluster** はクラスターレベルのリソースで、namespace レベルのリソースではないので、このロールが必要です。

- 以下のコマンドを入力して、クラスターロール **view** にバインドする namespace ロールを作成します。

```
oc create rolebinding <role-binding-name> -n <cluster-name> --clusterrole=view
```

このロールでは、マネージドクラスターの namespace 内にあるリソースに対して読み取り専用アクセスができるようになります。

ManagedClusterSet リソースの管理に 関する詳細は、「 **ManagedClusterSet** リソースの管理」を参照してください。

クラスターライフサイクルの以下のコンソールおよび API RBAC の表を表示します。

表1.2 クラスターライフサイクルのコンソール RBAC の表

アクション	管理	編集	表示
Clusters (クラスター)	read, update, delete	read, update	読み取り
プロバイダー接続	create, read, update, delete	create, read, update, delete	read
ベアメタルアセット	create, read, update, delete	read, update	read

表1.3 クラスターライフサイクルの API RBAC の表

API	管理	編集	表示
manageclusters.cluster.open-cluster-management.io	create, read, update, delete	read, update	read
baremetalassets.inventory.open-cluster-management.io	create, read, update, delete	read, update	read
klusterletaddonconfigs.agent.open-cluster-management.io	create, read, update, delete	read, update	read
managedclusteractions.action.open-cluster-management.io	create, read, update, delete	read, update	read
managedclusterviews.view.open-cluster-management.io	create, read, update, delete	read, update	read

API	管理	編集	表示
managedclusterinfos.internal.open-cluster-management.io	create, read, update, delete	read, update	read
manifestworks.work.open-cluster-management.io	create, read, update, delete	read, update	読み取り

1.1.2.2. アプリケーションライフサイクル RBAC

アプリケーションの作成時に、**サブスクリプション** namespace が作成され、設定マップが **サブスクリプション** namespace に作成されます。サブスクリプションを適用する場合は、サブスクリプションの管理者である必要があります。アプリケーションの管理の詳細は、「サブスクリプションの作成および管理」を参照してください。

アプリケーションライフサイクルタスクを実行するには、**admin** ロールを持つユーザーが、アプリケーションが作成される namespace および **managedcluster** namespace にアクセスできる必要があります。たとえば、namespace "N" 内でアプリケーションを作成するのに必要なアクセス権限は、namespace "N" の **admin** ロールに対する namespace スコープのバインディングです。

アプリケーションライフサイクルの以下のコンソールおよび API RBAC の表を表示します。

表1.4 アプリケーションライフサイクルのコンソール RBAC の表

アクション	管理	編集	表示
アプリケーション	create, read, update, delete	create, read, update, delete	read
チャネル	create, read, update, delete	create, read, update, delete	read
サブスクリプション	create, read, update, delete	create, read, update, delete	read
配置ルール	create, read, update, delete	create, read, update, delete	read

表1.5 アプリケーションライフサイクルの API RBAC の表

API	管理	編集	表示
applications.app.k8s.io	create, read, update, delete	create, read, update, delete	read
channels.apps.open-cluster-management.io	create, read, update, delete	create, read, update, delete	read

API	管理	編集	表示
deployables.apps.open-cluster-management.io	create, read, update, delete	create, read, update, delete	read
helmreleases.apps.open-cluster-management.io	create, read, update, delete	create, read, update, delete	read
placementrules.apps.open-cluster-management.io	create, read, update, delete	create, read, update, delete	read
subscriptions.apps.open-cluster-management.io	create, read, update, delete	create, read, update, delete	read
configmaps	create, read, update, delete	create, read, update, delete	read
secrets	create, read, update, delete	create, read, update, delete	read
namespace	create, read, update, delete	create, read, update, delete	read

1.1.2.3. ガバナンスライフサイクル RBAC

ユーザーは、ガバナンスライフサイクル操作を実行するには、ポリシーが作成される namespace および、ポリシーが適用される **managedcluster** namespace へのアクセス権が必要です。

以下の例を参照してください。

- namespace "N" でポリシーを表示するには、以下のロールが必要です。
  - namespace "N" の **view** ロールに対する namespace 限定のバインディング。
- namespace "N" でポリシーを作成し、これを **managedcluster** "X" に適用するには、以下のロールが必要です。
  - namespace "N" の **admin** ロールに対する namespace 限定のバインディング。
  - namespace "X" の **admin** ロールに対する namespace 限定のバインディング。

ガバナンスライフサイクルの以下のコンソールおよび API RBAC の表を表示します。

表1.6 ガバナンスライフサイクルのコンソール RBAC の表

アクション	管理	編集	表示
ポリシー	create, read, update, delete	read, update	read

アクション	管理	編集	表示
PlacementBindings	create, read, update, delete	read, update	read
PlacementRules	create, read, update, delete	read, update	read

表1.7 ガバナンスライフサイクルの API RBAC の表

API	管理	編集	表示
policies.policy.open-cluster-management.io	create, read, update, delete	read, update	read
placementbindings.policy.open-cluster-management.io	create, read, update, delete	read, update	read

1.1.2.4. 可観測性の RBAC

可観測性機能を使用するには、**cluster-admin** または **open-cluster-management:cluster-manager-admin** ロールを割り当てる必要があります。以下の可観測性機能のリストを参照してください。

- マネージドクラスターのメトリクスへのアクセス
- リソースの検索
- Visual Web ターミナルの使用 (マネージドクラスターにアクセスできる場合)

MultiClusterObservability カスタムリソースの作成、更新、削除には、以下を実行します。以下の RBAC の表を確認してください。

表1.8 可観測性の API RBAC の表

API	管理	編集	表示
multiclusterobservabilities.observability.open-cluster-management.io	作成、読み取り、更新、および削除	-	-

クラスターのセキュリティー保護に関する詳細の確認を続行するには、「セキュリティー」を参照してください。

1.2. 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、Red Hat Advanced Cluster Management 証明書マネージャー (**cert-manager**) により作成および管理されます。Red Hat Advanced Cluster Management ルート証明局 (CA) 証明書は、ハブクラスター namespace の Kubernetes Secret の **multicloud-ca-cert** 内に保存されます。証明書をクライアントトラストストアにインポートすることで、Red Hat Advanced Cluster Management Platform API にアクセスできます。

証明書を置き換えるには、以下のトピックを参照してください。

- ルート CA 証明書の置き換え
- 管理 Ingress 証明書の置き換え

1.2.1. 証明書のリスト

以下のコマンドを実行して、**cert-manager** を内部で使用する証明書の一覧を表示できます。

```
oc get certificates.certmanager.k8s.io -n open-cluster-management
```

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

1.2.2. 証明書の更新

「証明書のリスト」 セクションでコマンドを実行して、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
```

1.2.3. Red Hat Advanced Cluster Management for Kubernetes の証明書更新

Red Hat Advanced Cluster Management CA が発行するすべての証明書を更新できます。更新時に、各 **cert-manager** 証明書に関連付けられた Kubernetes Secret が削除されます。対象の証明書が使用されるように、サービスが自動的に再起動されます。以下のコマンドを実行します。

```
oc delete secret -n open-cluster-management $(oc get certificates.certmanager.k8s.io -n open-cluster-management -o wide | grep multicloud-ca-issuer | awk '{print $3}')
```

Red Hat OpenShift Container Platform 証明書は、Red Hat Advanced Cluster Management for Kubernetes の管理 Ingress に含まれていません。詳細は、「セキュリティーの既知の問題」を参照してください。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、セキュリティー ページに戻り、確認してください。

1.2.4. ルート CA 証明書の置き換え

ルート CA 証明書を置き換えることができます。

1.2.4.1. ルート CA 証明書の前提条件

Red Hat Advanced Cluster Management for Kubernetes クラスターが稼働していることを確認します。

以下のコマンドを実行して、既存の Red Hat Advanced Cluster Management for Kubernetes 証明書リソースをバックアップします。

```
oc get cert multicloud-ca-cert -n open-cluster-management -o yaml > multicloud-ca-cert-backup.yaml
```

1.2.4.2. OpenSSL を使用したルート CA 証明書の作成

OpenSSL でルート CA 証明書を作成するには、以下の手順を実行します。

1. 以下のコマンドを実行して、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。以下のコマンドを実行します。

   ```
   openssl req -x509 -new -nodes -key ca.key -days 400 -out ca.crt -config req.cnf
   ```

   req.cnf ファイルは以下のファイルのようになります。

   ```
   [ req ]              # Main settings
   default_bits = 4096       # 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.
   distinguished_name = dn   # Specifies the section that includes the distinguished name information.
   x509_extensions = v3_ca   # The extentions to add to the self signed cert

   [ 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 = www.redhat.com  # Common name.

   [ v3_ca ]          # x509v3 extensions
   basicConstraints=critical,CA:TRUE # Indicates whether the certificate is a CA certificate during the certificate chain verification process.
   ```

1.2.4.3. ルート CA 証明書の置き換え

1. 以下のコマンドを実行して CA 証明書で新規シークレットを作成します。

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

2. CA 発行者を編集して、独自の証明書を参照します。以下のコマンドを実行します。

```
oc edit issuer -n open-cluster-management multicloud-ca-issuer
```

3. **mulicloud-ca-cert** の文字列は **byo-ca-cert** に置き換えます。デプロイメントを保存し、エディターを終了します。

4. 管理 Ingress デプロイメントを編集して、Bring Your Own (BYO) の CA 証明書を参照します。以下のコマンドを実行します。

```
oc edit deployment management-ingress-435ab
```

5. **multicloud-ca-cert** の文字列は、**byo-ca-cert** に置き換えます。デプロイメントを保存し、エディターを終了します。

6. コンソールにログインしてカスタムの CA が使用中であることを確認し、使用中の証明書の詳細を表示します。

1.2.4.4. Cert-manager 証明書の更新

ルート CA を置き換えた後には、ルート CA で署名された全証明書を更新し、これらの証明書を使用するサービスを再起動する必要があります。Cert-manager は、ルート CA からのデフォルトの発行者を作成するので、**cert-manager** が発行した証明書および、デフォルトの ClusterIssuer が署名した証明書も、すべて更新する必要があります。

各 **cert-manager** 証明書に関連付けられた Kubernetes Secret を削除して、証明書を更新し、証明書を使用するサービスを再起動します。以下のコマンドを実行します。

```
oc delete secret -n open-cluster-management $(oc get cert -n open-cluster-management -o wide |
grep multicloud-ca-issuer | awk '{print $3}')
```

1.2.4.5. ルート CA 証明書の復元

ルート CA 証明書を復元するには、以下の手順を実行して CA 発行者を更新します。

1. CA 発行者を編集します。以下のコマンドを実行します。

```
oc edit issuer -n open-cluster-management multicloud-ca-issuer
```

2. エディターで **byo-ca-cert** の文字列を **multicloud-ca-cert** に置き換えます。発行者を保存し、エディターを終了します。

3. 管理 Ingress デプロイメントを編集して、元の CA 証明書を参照します。以下のコマンドを実行します。

```
oc edit deployment management-ingress-435ab
```

4. **byo-ca-cert** の文字列を **multicloud-ca-cert** に置き換えます。デプロイメントを保存し、エディターを終了します。

5. 独自の CA 証明書を削除します。以下のコマンドを実行します。

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

CA を使用するすべての **cert-manager** 証明書を更新します。詳細は前述のセクション「**cert-manager** 証明書の更新」を参照してください。

Red Hat Advanced Cluster Management for Kubernates 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。

1.2.5. 管理 Ingress 証明書の置き換え

管理 Ingress 証明書を置き換えることができます。OpenShift Container Platform のデフォルト Ingress 証明書を置き換える場合には、管理 Ingress を変更する必要があります。詳細は、「セキュリティーの既知の問題」の「**コンソールへのログイン時の 500 内部エラー**」を参照してください。

1.2.5.1. 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: Subject Alternative Name) の一覧に以下の IP アドレスおよびドメイン名を含めます。
  - 管理 ingress のサービス名: **management-ingress**。
  - 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
    ```

  - ローカルホストの IP アドレス (**127.0.0.1**) を追加します。
  - ローカルホストのエントリー (**localhost**) を追加します。

1.2.5.1.1. 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 での証明書生成の構成設定を定義します。

```
[ 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 = management-ingress
DNS.2 = multicloud-console.apps.grchub2.dev08.red-chesterfield.com
DNS.3 = localhost
DNS.4 = 127.0.0.1

[ 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.2** を必ず更新してください。

1.2.5.1.2. 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
```

1.2.5.2. 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。

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

2. 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。

```
kubectl get secret -n open-cluster-management | grep byo-ingress | grep tls
```

3. 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。

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

4. 管理 Ingress デプロイメントを編集します。デプロイメントの名前を指定します。以下のコマンドを実行します。

```
export MANAGEMENT_INGRESS=`oc get deployment -o custom-columns=:.metadata.name | grep management-ingress`

oc edit deployment $MANAGEMENT_INGRESS -n open-cluster-management
```

   - **multicloud-ca-cert** の文字列は、**byo-ca-cert** に置き換えます。
   - **$MANAGEMENT_INGRESS-tls-secret** の文字列は、**byo-ingress-tls-secret** に置き換えます。
   - デプロイメントを保存してエディターを終了します。管理 Ingress は自動的に再起動します。

5. 管理 Ingress Pod が再起動されたら、ブラウザーから Red Hat Advanced Cluster Management for Kubernetes コンソールに移動します。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。

1.2.5.3. 管理 Ingress のデフォルト自己署名証明書の復元

1. 管理 Ingress デプロイメントを編集します。**multicloud-ca-cert** の文字列は、**byo-ca-cert** に置き換えます。デプロイメントの名前を指定します。以下のコマンドを実行します。

```
export MANAGEMENT_INGRESS=`oc get deployment -o custom-columns=:.metadata.name | grep management-ingress`

oc edit deployment $MANAGEMENT_INGRESS -n open-cluster-management
```

   a. **byo-ca-cert** 文字列は、**multicloud-ca-cert** に置き換えます。

b. **byo-ingress-tls-secret** の文字列は、**$MANAGEMENT_INGRESS-tls-secret** に置き換えます。

c. デプロイメントを保存してエディターを終了します。管理 Ingress は自動的に再起動します。

2. すべての Pod が再起動されたら、ブラウザーから Red Hat Advanced Cluster Management for Kubernetes コンソールに移動します。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。

3. 以下のコマンドを実行して 独自の Ingress シークレットと ingress CA 証明書を削除します。

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

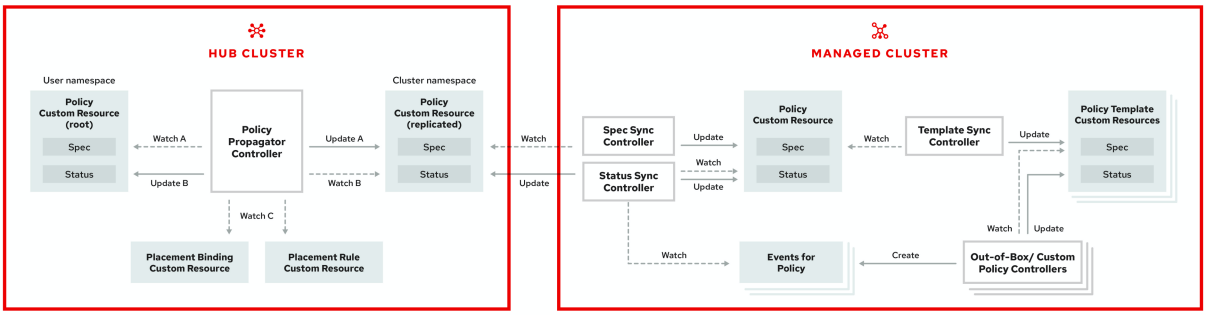
Red Hat Advanced Cluster Management for Kubernates で作成して管理する証明書の詳細は、「証明書」を参照してください。クラスターのセキュリティー保護に関する詳細は、セキュリティー ページに戻り、確認してください。

第2章 ガバナンスおよびリスク

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ースページからセキュリティーおよびコンプライアンスを管理します。ガバナンスアーキテクチャーの以下の図を参照してください。



ガバナンスアーキテクチャーは、以下のコンポーネントで構成されています。

- **ガバナンスおよびリスクダッシュボード**: 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ので、ポリシーの作成時は、**namespaceName.policyName** の長さが 63 文字未満となる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と

サードパーティーポリシーも **open-cluster-management/policy-collection** リポジトリーに含まれます。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構成、および Red Hat Advanced Cluster Management for Kubernetes のガバナンスおよびリスク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
- ポリシーサンプル
- セキュリティーポリシーの管理

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 CustomResourceDefinition (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** 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。

重要:

- **placementRule** を作成して、マネージドクラスターにポリシーを適用し、**placementRule** と **PlacementBinding** をバインドする必要があります。
- 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。

2.2.1. ポリシー YAML の構成

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を追加する必要がある場合があります。前述のパラメーターフィールドの YAML 構成は、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。

フィールド	説明
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。**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。
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: Center of Internet Security (CIS) および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、ポリシーを有効にしたり、無効にしたりできます。
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** と **inform** です。指定した場合には、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"]}
```

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、「 ガバナンスおよびリスク 」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 CustomResourceDefinition (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」を参照してください。

重要: 設定ポリシーコントローラーだけが **enforce** 機能をサポートしています。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。

Red Hat Advanced Cluster Management for Kubernetes の以下のポリシーコントローラーについては、次のトピックを参照してください。

- Kubernetes 設定ポリシーコントローラー
- 証明書ポリシーコントローラー
- IAM ポリシーコントローラー

ポリシー管理の他のトピックについては、「ガバナンスおよびリスク」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 nginx ポリシー
- Pod のセキュリティーポリシー
- ロールポリシー
- RoleBinding ポリシー

- SCC (Security Context Constraints) ポリシー
- ETCD 暗号化ポリシー

設定ポリシーの **remediationAction** が **enforce** に設定されている場合には、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。

2.3.1.1. 設定ポリシーコントローラーの YAML 構成

```
Name:         configuration-policy-example
Namespace:
Labels:
APIVersion:   policy.open-cluster-management.io/v1
Kind:         ConfigPolicy
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: ConfigPolicy
metadata:
  name: policy-config
spec:
  namespaceSelector:
    include: ["default"]
    exclude: []
  remediationAction: inform
    severity: low
    object-templates:
    - complianceType: musthave
      objectDefinition:
        apiVersion: v1
        kind: Pod
        metadata:
          name: nginx-pod
        spec:
          containers:
          - image: nginx:1.7.9
```

```
      name: nginx
      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 **ConfigPolicy** 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。
spec.namespaceSelector	必須。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** パラメーターでは、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。
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。
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リストするために使用されます。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。

ハブクラスターでポリシーを適用する方法を説明します。詳細は、「 ポリシーサンプル 」を参照してください。ポリシーを作成してカスタマイズする方法は、「セキュリティーポリシーの管理」を参照してください。

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には、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 構成

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
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。
netadata.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場合には、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には、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には、ポリシーは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は **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。

フィールド	説明
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は、デフォルト値は **100h** になります。このパラメーターで、証明書がコンプライアンス違反とみなされるまでの最小期間 (時間) を指定します。パラメーター値は Golang の期間形式を使用します。詳細は Golang Parse Duration を参照してください。
spec.minimumCADuration	任意。値を設定して、他の証明書とは異なる値でまもなく有効期限が切れる可能性がある署名証明書を特定します。パラメーターの値が指定されていない場合には、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。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。マネージドクラスターの証明書ポリシーは、以下のファイルのようになります。

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

```
metadata:
  name: certificate-policy-1
  namespace: kube-system
  label:
    category: "System-Integrity"
spec:
  namespaceSelector:
    include: ["default", "kube-*"]
    exclude: ["kube-system"]
  remediationAction: inform
  minimumDuration: 100h
  minimumCADuration: 200h
  maximumDuration: 2161h
  maximumCADuration: 43920h
  allowedSANPattern: "[[:alpha:]]"
  disallowedSANPattern: "[\\*]"
```

証明書ポリシーの管理方法の詳細は「証明書ポリシーの管理」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、クラスター内で許可されているクラスター管理者の数に関するコンプライアンスをチェックします。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。

2.3.3.1. IAM ポリシー YAML の構成

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:
  severity:
  namespaceSelector:
    include:
    exclude:
  remediationAction:
  maxClusterRoleBindingUsers:
```

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.namespaceSelector	必須。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** パラメーターでは、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、前述のパラメーター値の 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。**inform** と入力します。
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

```
apiVersion: policy.open-cluster-management.io/v1
kind: IamPolicy # limit clusteradminrole and report violation
metadata:
  name: {{name}}-example
spec:
  severity: medium
  namespaceSelector:
    include: ["*"]
    exclude: ["kube-*", "openshift-*"]
  remediationAction: inform # will be overridden by remediationAction in parent policy
  maxClusterRoleBindingUsers: 5
```

IAM ポリシーの管理方法の詳細は、「IAM 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. 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 1 つ以上を指定します。

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

2.3.5. カスタムポリシーコントローラーの作成

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。

2.3.5.1. ポリシーコントローラーの作成

multicloud-operators-policy-controller 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 **multicloud-operators-policy-controller** リポジトリーのクローンを作成します。

```
git clone git@github.com:open-cluster-management/multicloud-operators-policy-controller.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

operator-sdk run --local --verbose
```

-

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。メッセージは次のような内容になります。

```
{"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. 新しいフィールドを持つ samplepolicy_controller.go で、**SamplePolicySpec** 構造を更新します。

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、open-cluster-management/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。

2.3.5.2. 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およびリスク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。

```
operator-sdk build <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|open-cluster-management/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. PolicyController 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 CustomResourceDefinition (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。

2.3.5.2.1. 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場合、ポリシーコントロールのデプロイメントは無効になります。

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. ポリシーサンプル

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、うロセス、制御の定義方法を説明するポリシー例を確認します。

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。

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。

- Kubernetes 設定ポリシーコントローラーの例
- イメージ脆弱性ポリシーの例
- メモリー使用状況ポリシーの例
- Namespace ポリシーの例
- Pod nginx ポリシーの例
- Pod セキュリティーポリシーの例
- ロールポリシーの例
- RoleBinding ポリシーの例
- SCC (Security context constraints) ポリシーの例
- 証明書ポリシーの例
- IAM ポリシーの例
- gatekeeper ポリシーの例
- ETCD 暗号化ポリシーの例

他のトピックについては、「ガバナンスおよびリスク」を参照してください。

2.4.1. 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構成についてを説明します。

2.4.1.1. メモリー使用状況ポリシー YAML の構成

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:
          ...
```

2.4.1.2. 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** パラメーターでは、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。

フィールド	説明
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** と **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、ポリシーを有効にしたり、無効にしたりできます。
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のある他の Kubernetes オブジェクトを一覧表示するために使用します。

2.4.1.3. メモリー使用状況ポリシーの例

```
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  name: policy-limitrange
  namespace: mcm
spec:
  complianceType: musthave
  remediationAction: inform
  namespaces:
    exclude: ["kube-*"]
    include: ["default"]
  object-templates:
    - complianceType: musthave
      objectDefinition:
        apiVersion: v1
        kind: LimitRange # limit memory usage
        metadata:
          name: mem-limit-range
        spec:
          limits:
          - default:
              memory: 512Mi
            defaultRequest:
              memory: 256Mi
            type: Container
        ...
```

詳細は、「メモリー使用状況ポリシーの管理」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」のページを参照してください。

2.4.2. 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 ポリシーの構成について説明します。

2.4.2.1. Namespace ポリシー YAML の構成

```
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:
      ...
```

2.4.2.2. 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。
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** パラメーターでは、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。

フィールド	説明
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** と **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、ポリシーを有効にしたり、無効にしたりできます。
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のある他の Kubernetes オブジェクトを一覧表示するために使用します。

2.4.2.3. Namespace ポリシーの例

Namespace 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-namespace-1
  namespace: open-cluster-management
spec:
  complianceType: musthave
  remediationAction: inform
  namespaces:
    exclude: ["kube-*"]
    include: ["default"]
  object-templates:
    - complianceType: musthave
      objectDefinition:
        kind: Namespace # must have namespace 'prod'
        apiVersion: v1
        metadata:
          name: prod
    ...
```

Namespace ポリシーを管理します。詳細は、「namespace 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。

2.4.3. 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には、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。

2.4.3.1. イメージ脆弱性ポリシーの YAML 構成

```
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
```

2.4.3.2. 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** パラメーターでは、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** と **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、ポリシーを有効にしたり、無効にしたりできます。
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のある他の Kubernetes オブジェクトを一覧表示するために使用します。

2.4.3.3. イメージ脆弱性ポリシーの例

```
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: inform
  disabled: false
  policy-templates:
  - objectDefinition:
      apiVersion: policy.open-cluster-management.io/v1
      kind: ConfigurationPolicy
      metadata:
        name: policy-imagemanifestvulnpolicy-example-sub
      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: container-security-operator
              namespace: openshift-operators
            spec:
              channel: quay-v3.3
              installPlanApproval: Automatic
              name: container-security-operator
              source: redhat-operators
              sourceNamespace: openshift-marketplace
  - objectDefinition:
      apiVersion: policy.open-cluster-management.io/v1
      kind: ConfigurationPolicy
      metadata:
        name: policy-imagemanifestvulnpolicy-example-imv
      spec:
        remediationAction: inform # will be overridden by remediationAction in parent policy
        severity: high
        namespaceSelector:
          exclude: ["kube-*"]
          include: ["*"]
        object-templates:
          - complianceType: mustnothave # mustnothave any ImageManifestVuln object
            objectDefinition:
              apiVersion: secscan.quay.redhat.com/v1alpha1
              kind: ImageManifestVuln # checking for a kind
---
apiVersion: policy.open-cluster-management.io/v1
kind: PlacementBinding
metadata:
  name: binding-policy-imagemanifestvulnpolicy
  namespace: default
placementRef:
  name: placement-policy-imagemanifestvulnpolicy
  kind: PlacementRule
  apiGroup: apps.open-cluster-management.io
subjects:
- name: policy-imagemanifestvulnpolicy
  kind: Policy
  apiGroup: policy.open-cluster-management.io
---
apiVersion: apps.open-cluster-management.io/v1
kind: PlacementRule
metadata:
  name: placement-policy-imagemanifestvulnpolicy
  namespace: default
spec:
  clusterConditions:
  - status: "True"
    type: ManagedClusterConditionAvailable
  clusterSelector:
    matchExpressions:
      []  # selects all clusters if not specified
```

詳細は、「イメージ脆弱性ポリシーの管理」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。

2.4.4. Pod nginx ポリシー

Kubernetes 設定ポリシーコントローラーは、Pod nginx 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。Nginx Pod は、クラスター内に存在している必要があります。

2.4.4.1. Pod nginx ポリシー YAML 構成

```
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 # nginx pod must exist
        metadata:
          name:
        spec:
          containers:
          - image:
            name:
            ports:
            - containerPort:
    ...
```

2.4.4.2. Pod nginx 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。

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** パラメーターでは、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** と **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、ポリシーを有効にしたり、無効にしたりできます。
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のある他の Kubernetes オブジェクトを一覧表示するために使用します。

2.4.4.3. Pod nginx ポリシーの例

Pod nginx 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-pod
  namespace: open-cluster-management
spec:
  complianceType: musthave
  remediationAction: inform
  namespaces:
    exclude: ["kube-*"]
    include: ["default"]
  object-templates:
    - complianceType: musthave
      objectDefinition:
        apiVersion: v1
        kind: Pod # nginx pod must exist
        metadata:
          name: nginx-pod
        spec:
          containers:
```

```
        - image: nginx:1.7.9
          name: nginx
          ports:
          - containerPort: 80
    ...
```

Pod nginx ポリシーの管理方法の詳細は、「Pod nginx 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。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。

2.4.5. 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構成について説明します。

2.4.5.1. Pod セキュリティーポリシー YAML の構成

```
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:
    ...
```

2.4.5.2. 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** パラメーターでは、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** と **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、ポリシーを有効にしたり、無効にしたりできます。
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のある他の Kubernetes オブジェクトを一覧表示するために使用します。

2.4.5.3. Pod セキュリティーポリシーの例

Pod セキュリティーポリシーは、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-podsecuritypolicy
    namespace: open-cluster-management
  spec:
    complianceType: musthave
    remediationAction: inform
    namespaces:
      exclude: ["kube-*"]
      include: ["default"]
    object-templates:
      - complianceType: musthave
        objectDefinition:
          apiVersion: policy/v1beta1
          kind: PodSecurityPolicy # no privileged pods
          metadata:
            name: restricted-open-cluster-management
            annotations:
             seccomp.security.alpha.kubernetes.io/allowedProfileNames: '*'
          spec:
            privileged: false # no priviliedged pods
            allowPrivilegeEscalation: false
            allowedCapabilities:
            - '*'
            volumes:
            - '*'
            hostNetwork: true
            hostPorts:
            - min: 1000 # ports < 1000 are reserved
              max: 65535
            hostIPC: false
            hostPID: false
            runAsUser:
              rule: 'RunAsAny'
            seLinux:
              rule: 'RunAsAny'
            supplementalGroups:
              rule: 'RunAsAny'
            fsGroup:
              rule: 'RunAsAny'
      ...
```

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。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」のページを参照してください。

2.4.6. 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、ロールポリシーの構成について説明します。

2.4.6.1. ロールポリシー YAML の構成

```
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:
    []

      ...
```

2.4.6.2. 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。
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** パラメーターでは、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** と **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、ポリシーを有効にしたり、無効にしたりできます。
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のある他の Kubernetes オブジェクトを一覧表示するために使用します。

2.4.6.3. 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-role
  namespace: open-cluster-management
spec:
  complianceType: musthave
  remediationAction: inform
  namespaces:
    exclude: ["kube-*"]
    include: ["default"]
  role-templates:
    - apiVersion: open-cluster-management.io/v1/v1alpha1 # role must follow defined permissions
      metadata:
        namespace: "" # will be inferred
        name: operator-role-policy
      selector:
        matchLabels:
          dev: "true"
      complianceType: musthave # at this level, it means the role must exist with the rules that it must have the following
      rules:
        - complianceType: musthave # at this level, it means if the role exists the rule is a musthave
          policyRule:
            apiGroups: ["extensions", "apps"]
            resources: ["deployments"]
            verbs: ["get", "list", "watch", "create", "delete","patch"]
        - complianceType: "mustnothave" # at this level, it means if the role exists the rule is a mustnothave
          policyRule:
            apiGroups: ["core"]
            resources: ["secrets"]
            verbs: ["get", "list", "watch","delete", "create", "update", "patch"]
      ...
```

詳細は、「ロールポリシーの管理」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」のページを参照してください。Red Hat Advanced Cluster Management for Kubernates RBAC 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。

2.4.7. RoleBinding ポリシー

Kubernetes 設定ポリシーコントローラーは、rolebinding ポリシーのステータスを監視します。Rolebinding 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 ポリシーの構成について説明します。

2.4.7.1. RoleBinding ポリシー YAML の構成

```
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
      ...
```

2.4.7.2. RoleBinding 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。
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。

フィールド	説明
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** パラメーターでは、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** と **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.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のある他の Kubernetes オブジェクトを一覧表示するために使用します。

2.4.7.3. RoleBinding ポリシーの例

rolebinding 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-rolebinding
  namespace: open-cluster-management
spec:
  complianceType: musthave
  remediationAction: inform
  namespaces:
    exclude: ["kube-*"]
    include: ["default"]
  object-templates:
    - complianceType: musthave
      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: rbac.authorization.k8s.io
        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
    ...
```

rolebinding ポリシー の管理方法の詳細は、「rolebinding 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。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。

2.4.8. 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。

2.4.8.1. SCC ポリシー YAML の構成

```
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:
```

2.4.8.2. 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。
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** と **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** パラメーターでは、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のある他の Kubernetes オブジェクトを一覧表示するために使用します。

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) について」を参照してください。

2.4.8.3. 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。SCC 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-scc
    namespace: open-cluster-management
    annotations:
      policy.open-cluster-management.io/standards: NIST-CSF
      policy.open-cluster-management.io/categories: PR.PT Protective Technology
      policy.open-cluster-management.io/controls: PR.PT-3 Least Functionality
  spec:
    complianceType: musthave
    remediationAction: inform
    disabled: false
    namespaces:
      exclude: ["kube-*"]
      include: ["default"]
    object-templates:
      - complianceType: musthave
        objectDefinition:
          apiVersion: security.openshift.io/v1
          kind: SecurityContextConstraints # restricted scc
          metadata:
            annotations:
              kubernetes.io/description: restricted denies access to all host features and requires pods to
be run with a UID, and SELinux context that are allocated to the namespace.  This is the most
restrictive SCC and it is used by default for authenticated users.
            name: sample-restricted-scc
          allowHostDirVolumePlugin: false
          allowHostIPC: false
          allowHostNetwork: false
          allowHostPID: false
          allowHostPorts: false
          allowPrivilegeEscalation: true
          allowPrivilegedContainer: false
          allowedCapabilities: []
          defaultAddCapabilities: []
          fsGroup:
            type: MustRunAs
          groups:
          - system:authenticated
          priority: null
          readOnlyRootFilesystem: false
          requiredDropCapabilities:
          - KILL
          - MKNOD
          - SETUID
          - SETGID
          runAsUser:
            type: MustRunAsRange
          seLinuxContext:
            type: MustRunAs
          supplementalGroups:
            type: RunAsAny
          users: []
          volumes:
          - configMap
```

```
        - downwardAPI
        - emptyDir
        - persistentVolumeClaim
        - projected
        - secret
---
apiVersion: apps.open-cluster-management.io/v1
kind: PlacementBinding
metadata:
  name: binding-policy-scc
  namespace: open-cluster-management-policies
placementRef:
  name: placement-policy-scc
  kind: PlacementRule
  apiGroup: apps.open-cluster-management.io
subjects:
- name: policy-scc
  kind: Policy
  apiGroup: policy.mcm.ibm.com
---
apiVersion: apps.open-cluster-management.io/v1
kind: PlacementBinding
metadata:
  name: policy-scc-production-clusters
  namespace: open-cluster-management-policies
placementRef:
  name: production-clusters
  kind: PlacementRule
  apiGroup: apps.open-cluster-management.io
subjects:
- name: policy-scc
  kind: Policy
  apiGroup: policy.mcm.ibm.com
---
apiVersion: apps.open-cluster-management.io/v1
kind: PlacementRule
metadata:
  name: placement-policy-scc
  namespace: open-cluster-management-policies
spec:
  clusterConditions:
    - type: ManagedClusterConditionAvailable
      status: "True"
  clusterSelector:
    matchExpressions: []
```

SCC ポリシーの管理方法の詳細は、「Security Context Constraints 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。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。

2.4.9. 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 暗号化**」を参照してください。

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

2.4.9.1. ETCD 暗号化ポリシーの YAML 構成

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:
        ...
```

2.4.9.2. 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** パラメーターでは、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** と **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、ポリシーを有効にしたり、無効にしたりできます。
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のある他の Kubernetes オブジェクトを一覧表示するために使用します。詳細は、OpenShift Container Platform ドキュメント を参照してください。

2.4.9.3. etcd 暗号化ポリシーの例

```
apiVersion: policy.open-cluster-management.io/v1
kind: ConfigurationPolicy
metadata:
  name: policy-etcdencryption
  namespace: default
spec:
  complianceType: musthave
  remediationAction: inform
  namespaces:
    exclude: ["kube-*"]
    include: ["default"]
  object-templates:
    - complianceType: musthave
      objectDefinition:
        apiVersion: config.openshift.io/v1
        kind: APIServer
        metadata:
          name: cluster
        spec:
          encryption:
            type: aescbc
        ...
```

詳細は、「ETCD 暗号化ポリシーの管理」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」のページを参照してください。

2.4.10. 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 CustomResourceDefinition (CRD) ベースのポリシーを適用する検証用の Webhook です。gatekeeper をインストールして、Red Hat Advanced Cluster Management for Kubernetes と gatekeeper ポリシーを統合できます。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作成されます。gatekeeper ポリシーには、制約テンプレート (**ConstraintTemplates**) および制約、監査テンプレート、受付テンプレートが含まれます。詳細は、gatekeeper を参照してください。

前提条件:

- gatekeeper ポリシーコントローラーを使用するには、gatekeeper をマネージドクラスターにインストールする必要があります。詳細は、**open-policy-agent/gatekeeper リポジトリー** を参照してください。
- Kubernetes バージョン 1.14 以降

Red Hat Advanced Cluster Management では、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"]
```

- **audit** 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
```

- **admission** テンプレート: **policy-gatekeeper-admission** を使用して、gatekeeper admission Webhook によって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。

Red Hat Advanced Cluster Management gatekeeper Operator ポリシーを使用して gatekeeper をインストールし、Red Hat Advanced Cluster Management gatekeeper Operator オペレーターポリシーを作成する方法は、「 gatekeeper policy integration 」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、「ガバナンスおよびリスク」を参照してください。

2.5. 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、ガバナンスおよびリスクの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。

ガバナンスおよびリスク ページでは、カテゴリーや基準で違反をフィルタ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リングできます。

ポリシーの表では、**Policy name**、**Namespace**、**Remediation**、**Cluster violation**、**Standards**、**Categories** および **Controls** のポリシーの情報を表示します。**Actions** アイコンを選択すると、ポリシーの編集、無効化、通知、または削除が可能です。

表一覧でポリシーを選択すると、コンソールで、以下の情報タブが表示されます。

- **詳細**: **Details** タブを選択して、ポリシーの情報、配置の情報、_Policy テンプレートの表一覧を表示します。
- **Status**: **Status** タブを選択して、違反の表一覧を表示します。**Clusters** または **Templates** 別にビューをフィルタリングできます。ポリシーのコンプライアンスステータスを表示するには、**Status** タブを選択します。**View history** リンクをクリックして、違反メッセージの一覧を表示します。
- **YAML**: **YAML** タブを選択して、ポリシーを表示するか、エディターでポリシーを編集します。YAML の切り替えを選択してエディターを表示または非表示にします。

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

- イメージ脆弱性ポリシーの管理
- メモリー使用状況ポリシーの管理
- Namespace ポリシーの管理
- Pod nginx ポリシーの管理
- Pod セキュリティーポリシーの管理
- ロールポリシーの管理
- Rolebinding ポリシーの管理
- Security Context Constraints ポリシーの管理
- 証明書ポリシーの管理
- IAM ポリシーの管理
- ETCD 暗号化ポリシーの管理

他のトピックについては、「ガバナンスおよびリスク」を参照してください。

2.5.1. 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。

2.5.1.1. 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ース (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、PlacementPolicy および PlacementBinding を定義する必要があります。**Cluster binding** フィールドに値を入力して、PlacementPolicy と PlacementBinding を定義します。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 PlacementPolicy 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。

2.5.1.1.1. コマンドラインインターフェー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ース (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。「配置ルールの作成および管理」を参照してください。**PlacementRule** は以下の内容のようになります。

```
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.mcm.ibm.com
  kind: Policy
```

2.5.1.1.1.1. 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>
```

2.5.1.1.2. 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。

1. ナビゲーションメニューから **Govern risk** をクリックします。
2. ポリシーを作成するには、**Create policy** をクリックします。
3. 以下のパラメーターの値を入力または選択します。
   - Name (名前)
   - Specifications (仕様)
   - Cluster selector (クラスターセレクター)
   - Remediation action (修復アクション)
   - Standards (標準)
   - Categories (カテゴリー)
   - Controls (制御)

4. 以下で、Red Hat Advanced Cluster Management for Kubernetes セキュリティーポリシー定義の例を表示します。次に、ポリシーの YAML ファイルをコピーアンドペーストします。YAML ファイルは以下のポリシーのようになります。

```
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  name: policy-pod
  annotations:
    policy.open-cluster-management.io/categories:
'SystemAndCommunicationsProtections,SystemAndInformationIntegrity'
    policy.open-cluster-management.io/controls: 'control example'
    policy.open-cluster-management.io/standards: 'NIST,HIPAA'
spec:
  complianceType: musthave
  namespaces:
    exclude: ["kube*"]
    include: ["default"]
  object-templates:
  - complianceType: musthave
    objectDefinition:
      apiVersion: v1
      kind: Pod
      metadata:
        name: nginx1
      spec:
        containers:
        - name: nginx
          image: 'nginx:1.7.9'
          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.mcm.ibm.com

---
apiVersion: apps.open-cluster-management.io/v1
kind: PlacementRule
metadata:
  name: placement-pod
spec:
  clusterConditions:
    - type: ManagedClusterConditionAvailable
```

```
      status: "True"
  clusterLabels:
    matchLabels:
      cloud: "IBM"
```

5. **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。

2.5.1.1.2.1. 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できます。

1. コンソールから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、**Governance and risk** をクリックして、ポリシー表の一覧を表示します。
   注記: ポリシー表の一覧をフィルタリングするには、**Policies** タブまたは **Cluster violations** タブを選択します。
3. 詳細を表示するポリシーを 1 つ選択します。**Overview** タブ、**Status** タブ、および **YAML** タブが表示されます。

2.5.1.2. 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。

2.5.1.2.1. 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。ポリシーを無効にするには、以下の手順を実行します。

1. Red Hat Advanced Cluster Management for Kubernetes コンソールにログインします。
2. ナビゲーションメニューから **Govern risk** をクリックし、ポリシー表の一覧を表示します。
3. **Options** icon > **Disable policy** の順にクリックして、ポリシーを無効化します。**Disable Policy** ダイアログボックスが表示されます。
4. **Disable policy** をクリックします。

ポリシーが無効化されました。

2.5.1.2.2. 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。**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。

  a. ナビゲーションメニューから **Govern risk** をクリックし、ポリシー表の一覧を表示します。

  b. 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。

  c. **Remove** をクリックします。

  d. **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、「ガバナンスおよびリスク」を参照してください。

2.5.2. 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。

2.5.2.1. 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ース (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。

2.5.2.1.1. 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ポリシーのようになります。

   ```
   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。

2.5.2.1.1.1. 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>
```

2.5.2.1.2. 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. コンソールから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 **Governance and risk** をクリックします。
3. **Create policy** をクリックします。
4. 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。次に、以下のフィールドに適切な値を入力するか、または選択します。
   - Name (名前)
   - Specifications (仕様)
   - Cluster selector (クラスターセレクター)
   - Remediation action (修復アクション)
   - Standards (標準)
   - Categories (カテゴリー)
   - Controls (制御)
5. **Create** をクリックします。

2.5.2.1.2.1. 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できます。

1. コンソールから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 risk** をクリックし、ポリシー表の一覧を表示します。
   注記: ポリシー表の一覧をフィルタリングするには、**All policies** タブまたは **Cluster violations** タブを選択します。

3. 詳細を表示するポリシーを 1 つ選択します。**Overview** タブ、**Status** タブ、および **YAML** タブが表示されます。

2.5.2.2. 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。

2.5.2.2.1. 設定ポリシーの無効化

設定ポリシーを無効にするには、以下の手順を実行します。

1. Red Hat Advanced Cluster Management for Kubernetes コンソールにログインします。
2. ナビゲーションメニューから **Govern risk** をクリックし、ポリシー表の一覧を表示します。
3. **Actions** icon > **Disable** の順にクリックして、ポリシーを無効化します。**Disable Policy** ダイアログボックスが表示されます。
4. **Disable policy** をクリックします。

ポリシーが無効化されました。

2.5.2.3. 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 <mcm namespace>
  ```

  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。

  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 <mcm namespace>
  ```

- コンソールから設定ポリシーを削除します。
  a. ナビゲーションメニューから **Govern risk** をクリックし、ポリシー表の一覧を表示します。
  b. 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。
  c. **Remove** をクリックします。
  d. **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。

設定ポリシーの例を表示するには、「ポリシーサンプル」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。

2.5.3. イメージ脆弱性ポリシーの管理

設定ポリシーコントローラーは、イメージの脆弱性ポリシーのステータスを監視します。イメージ脆弱性ポリシーを適用すると、コンテナーに脆弱性があるかどうかを確認することができます。イメージ脆弱性ポリシーの作成、適用、表示、更新について説明します。

2.5.3.1. イメージ脆弱性ポリシーの作成

コマンドラインインターフェース (CLI) またはコンソールから、イメージ脆弱性ポリシーの YAML を作成できます。以下のセクションを参照して、イメージ脆弱性ポリシーを作成します。

2.5.3.1.1. CLI からのイメージ脆弱性ポリシーの作成

CLI からイメージ脆弱性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、イメージ脆弱性ポリシーの YAML ファイルを作成します。

```
kubectl create -f imagevulnpolicy-1.yaml
```

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 <imagevuln-policy-file-name>  --namespace=<namespace>
```

3. 以下のコマンドを実行してポリシーを表示して検証します。

```
kubectl get imagevulnpolicy --namespace=<namespace>
```

イメージ脆弱性ポリシーが作成されました。

2.5.3.1.1.1. CLI からのイメージ脆弱性ポリシーの表示

CLI からイメージ脆弱性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、特定のイメージ脆弱性ポリシーの詳細を表示します。

```
kubectl get imagevulnpolicy <policy-name> -n <namespace> -o yaml
```

2. 以下のコマンドを実行して、イメージ脆弱性ポリシーの詳細を表示します。

```
kubectl describe imagevulnpolicy <name> -n <namespace>
```

2.5.3.2. コンソールからのイメージ脆弱性ポリシーの作成

コンソールからイメージ脆弱性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。コンソールからイメージの脆弱性ポリシーを作成するには、以下の手順を実施します。

1. コンソールから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 **Governance and risk** をクリックします。
3. **Create policy** をクリックします。
4. **Specifications** フィールドから **ImageManifestVulnPolicy** を選択します。パラメーターの値は自動的に設定されます。値は編集できます。

5. **Create** をクリックします。

イメージ脆弱性ポリシーが作成されました。

2.5.3.3. コンソールからのイメージの脆弱性違反の表示

1. ナビゲーションメニューから **Govern risk** をクリックし、ポリシー表の一覧を表示します。
2. **policy-imagemanifestvulnpolicy** > **Status** を選択して、違反のあるクラスター場所を表示します。
   イメージの脆弱性違反は、以下のように表示されます。

   ```
   imagemanifestvulns exist and should be deleted:
   [sha256.7ac7819e1523911399b798309025935a9968b277d86d50e5255465d6592c0266] in
   namespace default;
   [sha256.4109631e69d1d562f014dd49d5166f1c18b4093f4f311275236b94b21c0041c0] in
   namespace calamari;
   [sha256.573e9e0a1198da4e29eb9a8d7757f7afb7ad085b0771bc6aa03ef96dedc5b743,
   sha256.a56d40244a544693ae18178a0be8af76602b89abe146a43613eaeac84a27494e,
   sha256.b25126b194016e84c04a64a0ad5094a90555d70b4761d38525e4aed21d372820] in
   namespace open-cluster-management-agent-addon;
   [sha256.64320fbf95d968fc6b9863581a92d373bc75f563a13ae1c727af37450579f61a] in
   namespace openshift-cluster-version
   ```

3. **Cluster** リンクをクリックして OpenShift Container Platform コンソールに移動します。
4. OpenShift Container Platform コンソールのナビゲーションメニューから **Administration** > **Custom Resource Definitions**の順にクリックします。
5. **imagemanifestvulns** > **Instances タブを選択して、imagemanifestvulns** インスタンスすべてを表示します。
6. 詳細を表示するエントリーを選択します。

2.5.3.4. イメージ脆弱性ポリシーの更新

イメージ脆弱性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。

2.5.3.4.1. イメージ脆弱性ポリシーの無効化

イメージ脆弱性ポリシーを無効にするには、以下の手順を実行します。

1. Red Hat Advanced Cluster Management for Kubernetes コンソールにログインします。
2. ナビゲーションメニューから **Govern risk** をクリックし、ポリシー表の一覧を表示します。
3. **Actions** icon > **Disable** の順にクリックして、ポリシーを無効化します。**Disable Policy** ダイアログボックスが表示されます。
4. **Disable policy** をクリックします。

ポリシーが無効化されました。

2.5.3.4.2. イメージ脆弱性ポリシーの削除

CLI またはコンソールからイメージ脆弱性ポリシーを削除します。

- CLI からイメージ脆弱性ポリシーを削除します。
  a. 以下のコマンドを実行して証明書ポリシーを削除します。

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

     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。

  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 <imagevulnpolicy-name> -n <mcm namespace>
     ```

- コンソールからイメージ脆弱性ポリシーを削除します。
  a. ナビゲーションメニューから **Govern risk** をクリックし、ポリシー表の一覧を表示します。

  b. 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。

  c. **Remove** をクリックします。

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

イメージの脆弱性ポリシーが削除されました。

イメージ脆弱性ポリシーの例を確認するには、「イメージ脆弱性ポリシー」ページから、**イメージ脆弱性ポリシーの例** を参照してください。Kubernetes 設定ポリシーコントローラーが監視する他の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。

2.5.4. メモリー使用状況ポリシーの管理

メモリー使用状況ポリシーを適用して、メモリーおよびコンピュート使用量を制限または制限します。以下のセクションでは、メモリー使用状況ポリシーの作成、適用、表示、および更新について説明します。

2.5.4.1. メモリー使用状況ポリシーの作成

メモリー使用状況ポリシーの YAML ファイルは、コマンドラインインターフェース (CLI) またはコンソールから作成できます。以下のセクションを参照して、メモリー使用状況ポリシーを作成します。

2.5.4.1.1. CLI からのメモリー使用状況ポリシーの作成

CLI からメモリー使用状況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、メモリー使用状況ポリシーの YAML ファイルを作成します。

   ```
   kubectl create -f memorypolicy-1.yaml
   ```

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 <memory-policy-file-name>  --namespace=<namespace>
   ```

3. 以下のコマンドを実行してポリシーを表示して検証します。

```
kubectl get memorypolicy --namespace=<namespace>
```

CLI からメモリー使用状況ポリシーが作成されました。

2.5.4.1.1.1. CLI からのポリシーの表示

CLI からメモリー使用状況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、特定のメモリー使用状況ポリシーの詳細を表示します。

```
kubectl get memorypolicy <policy-name> -n <namespace> -o yaml
```

2. 以下のコマンドを実行して、メモリー使用状況ポリシーの詳細を表示します。

```
kubectl describe memorypolicy <name> -n <namespace>
```

2.5.4.1.2. コンソールからのメモリー使用状況ポリシーの作成

コンソールからメモリー使用状況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。コンソールからメモリー使用状況ポリシーを作成するには、以下の手順を実行します。

1. Red Hat Advanced Cluster Management for Kubernetes コンソールにログインします。
2. ナビゲーションメニューから **Governance and risk** をクリックします。
3. **Create policy** をクリックします。
4. **Specifications** フィールドから **Limitrange** を選択します。パラメーターの値は自動的に設定されます。値は編集できます。
5. **Create** をクリックします。

2.5.4.1.2.1. コンソールからのメモリー使用状況ポリシーの表示

コンソールからメモリー使用状況ポリシーおよびそのステータスを表示できます。

1. コンソールから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 risk** をクリックし、ポリシー表の一覧を表示します。
   注記: ポリシー表の一覧をフィルタリングするには、**Policies** タブまたは **Cluster violations** タブを選択します。
3. 詳細を表示するポリシーを 1 つ選択します。
4. **Status** タブを選択して、ポリシー違反を表示します。

2.5.4.2. メモリー使用状況ポリシーの更新

メモリー使用状況ポリシーについては、以下のセクションを参照してください。

2.5.4.2.1. メモリー使用状況ポリシーの無効化

メモリー使用状況ポリシーを無効にするには、以下の手順を実行します。

1. Red Hat Advanced Cluster Management for Kubernetes コンソールにログインします。
2. ナビゲーションメニューから **Govern risk** をクリックし、ポリシー表の一覧を表示します。
3. **Actions** icon > **Disable** の順にクリックして、ポリシーを無効化します。**Disable Policy** ダイアログボックスが表示されます。
4. **Disable policy** をクリックします。

ポリシーが無効化されました。

2.5.4.2.2. メモリー使用状況ポリシーの削除

CLI またはコンソールからメモリー使用状況ポリシーを削除します。

- CLI からメモリー使用状況ポリシーを削除します。
  a. 以下のコマンドを実行してメモリー状況ポリシーを削除します。

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

  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。

  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 <memorypolicy-name> -n <mcm namespace>
  ```

- コンソールからメモリー使用状況ポリシーを削除します。
  a. ナビゲーションメニューから **Govern risk** をクリックし、ポリシー表の一覧を表示します。
  b. 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。
  c. **Remove** をクリックします。
  d. **Remove policy** ダイアログボックスから、**Remove policy** をクリックします。

メモリー使用状況ポリシーが削除されます。

メモリー使用状況ポリシーの例については、「メモリー使用状況ポリシー」ページの **メモリー使用状況ポリシーの例** 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。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。

2.5.5. Namespace ポリシーの管理

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、メモリー使用状況ポリシーの作成、適用、表示、および更新について説明します。

2.5.5.1. Namespace ポリシーの作成

コマンドラインインターフェース (CLI) またはコンソールから namespace ポリシーの YAML ファイルを作成できます。namespace ポリシーの作成には、以下のセクションを参照してください。

2.5.5.1.1. CLI からの namespace ポリシーの作成

CLI から namespace 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 namespace ポリシーの YAML ファイルを作成します。

```
kubectl create -f namespacepolicy-1.yaml
```

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 <namespace-policy-file-name>  --namespace=<namespace>
```

3. 以下のコマンドを実行してポリシーを表示して検証します。

```
kubectl get namespacepolicy --namespace=<namespace>
```

CLI からNamespace ポリシーが作成されました。

2.5.5.1.1.1. CLI からの namespace ポリシーの表示

CLI から namespace 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 namespace ポリシーの詳細を表示します。

```
kubectl get namespacepolicy <policy-name> -n <namespace> -o yaml
```

2. 以下のコマンドを実行して namespace ポリシーの詳細を表示します。

```
kubectl describe namespacepolicy <name> -n <namespace>
```

2.5.5.1.2. コンソールからの namespace ポリシーの作成

コンソールから namespace 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 namespace ポリシーを作成するには、以下の手順を実行します。

1. Red Hat Advanced Cluster Management for Kubernetes コンソールにログインします。
2. ナビゲーションメニューから **Governance and risk** をクリックします。
3. **Create policy** をクリックします。
4. **Specifications** フィールドから **Namespace** を選択します。パラメーターの値は自動的に設定されます。値は編集できます。
5. **Create** をクリックします。

2.5.5.1.2.1. コンソールからの namespace ポリシーの表示

コンソールから namespace ポリシーおよびそのステータスを表示できます。

1. コンソールから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、**Governance and risk** をクリックして、ポリシー表の一覧を表示します。

注記: ポリシー表の一覧をフィルタリングするには、**Policies** タブまたは **Cluster violations** タブを選択します。

3. 詳細を表示するポリシーを 1 つ選択します。
4. **Status** タブを選択して、ポリシー違反を表示します。

2.5.5.2. Namespace ポリシーの更新

namespace 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。

2.5.5.2.1. Namespace ポリシーの無効化

Namespace ポリシーを無効にするには、以下の手順を実行します。

1. Red Hat Advanced Cluster Management for Kubernetes コンソールにログインします。
2. ナビゲーションメニューから **Govern risk** をクリックし、ポリシー表の一覧を表示します。
3. **Actions** icon > **Disable** の順にクリックして、ポリシーを無効化します。**Disable Policy** ダイアログボックスが表示されます。
4. **Disable policy** をクリックします。

ポリシーが無効化されました。

2.5.5.2.2. Namespace ポリシーの削除

CLI またはコンソールから namespace ポリシーを削除します。

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

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

  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。

  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 <memorypolicy-name> -n <mcm namespace>
  ```

- コンソールから namespace ポリシーを削除します。
  a. ナビゲーションメニューから **Govern risk** をクリックし、ポリシー表の一覧を表示します。
  b. 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。
  c. **Remove** をクリックします。
  d. **Remove policy** ダイアログボックスから、**Remove policy** をクリックします。

Namespace ポリシーが削除されます。

namespace ポリシーの例については、「Namespace ポリシー」ページの **Namespace ポリシーの例** 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。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。

2.5.6. Pod nginx ポリシーの管理

Kubernetes 設定ポリシーコントローラーは、Pod nginx ポリシーのステータスを監視します。Pod nginx ポリシーを適用して、Pod のコンテナールールを定義します。Pod nginx ポリシーの作成、適用、表示、更新について説明します。

2.5.6.1. Pod nginx ポリシーの作成

コマンドラインインターフェース (CLI) またはコンソールから Pod nginx ポリシーの 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、Pod nginx ポリシーを作成します。

2.5.6.1.1. CLI からの Pod nginx ポリシーの作成

CLI から Pod nginx 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、Pod nginx ポリシーの YAML ファイルを作成します。

```
kubectl create -f podnginxpolicy-1.yaml
```

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 <podnginx-policy-file-name>  --namespace=<namespace>
```

3. 以下のコマンドを実行してポリシーを表示して検証します。

```
kubectl get podnginxpolicy --namespace=<namespace>
```

CLI からイメージ Pod nginx ポリシーが作成されました。

2.5.6.1.1.1. CLI からの nginx ポリシーの表示

CLI から Pod nginx 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 Pod nginx ポリシーの詳細を表示します。

```
kubectl get podnginxpolicy <policy-name> -n <namespace> -o yaml
```

2. 以下のコマンドを実行して、Pod nginx ポリシーの詳細を表示します。

```
kubectl describe podnginxpolicy <name> -n <namespace>
```

2.5.6.2. コンソールからの Pod nginx ポリシーの作成

コンソールから Pod nginx 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 Pod nginx ポリシーを作成するには、以下の手順を実行します。

1. Red Hat Advanced Cluster Management for Kubernetes コンソールにログインします。
2. ナビゲーションメニューから **Govern risk** をクリックします。

3. **Create policy** をクリックします。
4. **Specifications** フィールドから **Pod** を選択します。パラメーターの値は自動的に設定されます。値は編集できます。
5. **Create** をクリックします。

コンソールからの Pod nginx ポリシーの表示
コンソールから Pod nginx ポリシーおよびそのステータスを表示できます。

1. コンソールから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 risk** をクリックし、ポリシー表の一覧を表示します。
注記: ポリシー表の一覧をフィルタリングするには、**Policies** タブまたは **Cluster violations** タブを選択します。
3. 詳細を表示するポリシーを 1 つ選択します。
4. **Status** タブを選択して、ポリシー違反を表示します。

2.5.6.3. Pod nginx ポリシーの更新

Pod nginx 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。

2.5.6.3.1. Pod nginx ポリシーの無効化

Pod nginx ポリシーを無効にするには、以下の手順を実行します。

1. Red Hat Advanced Cluster Management for Kubernetes コンソールにログインします。
2. ナビゲーションメニューから **Govern risk** をクリックし、ポリシー表の一覧を表示します。
3. **Actions** icon > **Disable** の順にクリックして、ポリシーを無効化します。**Disable Policy** ダイアログボックスが表示されます。
4. **Disable policy** をクリックします。

ポリシーが無効化されました。

2.5.6.3.2. Pod nginx ポリシーの削除

CLI またはコンソールから Pod nginx ポリシーを削除します。

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

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

  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。
  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 <podnginxpolicy-name> -n <namespace>
  ```

- コンソールから Pod nginx ポリシーを削除します。

a. ナビゲーションメニューから **Govern risk** をクリックし、ポリシー表の一覧を表示します。

b. 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。

c. **Remove** をクリックします。

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

Pod nginx ポリシーが削除されました。

Pod nginx ポリシーの例については、「Pod nginx ポリシー」ページの **Pod nginx ポリシーの例** 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。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。

2.5.7. Pod セキュリティーポリシーの管理

Pod のセキュリティーポリシーを適用して Pod およびコンテナーのセキュリティーを保護します。以下のセクションでは、Pod セキュリティーの作成、適用、表示、更新について説明します。

2.5.7.1. Pod 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ース (CLI) またはコンソールから Pod セキュリティーポリシーの 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、Pod のセキュリティーポリシーを作成します。

2.5.7.1.1. CLI からの Pod セキュリティーポリシーの作成

CLI から Pod セキュリティーを作成するには、以下の手順を実行します。

1. 以下のコマンドを実行して、Pod セキュリティーポリシーの YAML ファイルを作成します。

```
kubectl create -f podsecuritypolicy-1.yaml
```

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 <podsecurity-policy-file-name>  --namespace=<namespace>
```

3. 以下のコマンドを実行してポリシーを表示して検証します。

```
kubectl get podsecuritypolicy --namespace=<namespace>
```

CLI から Pod セキュリティーポリシーが作成されました。

2.5.7.1.1.1. CLI からの Pod 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 Pod 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 Pod セキュリティーポリシーの詳細を表示します。

```
kubectl get podsecuritypolicy <policy-name> -n <namespace> -o yaml
```

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

```
kubectl describe podsecuritypolicy <name> -n <namespace>
```

2.5.7.1.2. コンソールからの Pod セキュリティーポリシーの作成

コンソールから Pod セキュリティー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 Pod セキュリティーポリシーを作成するには、以下の手順を実行します。

1. Red Hat Advanced Cluster Management for Kubernetes コンソールにログインします。
2. ナビゲーションメニューから **Govern risk** をクリックします。
3. **Create policy** をクリックします。
4. **Specifications** フィールドから **Podsecuritypolicy** を選択します。パラメーターの値は自動的に設定されます。値は編集できます。
5. **Create** をクリックします。

2.5.7.1.2.1. コンソールからの Pod セキュリティーポリシーの表示

コンソールから Pod セキュリティーポリシーおよびそのステータスを表示できます。

1. コンソールから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 risk** をクリックし、ポリシー表の一覧を表示します。
   注記: ポリシー表の一覧をフィルタリングするには、**Policies** タブまたは **Cluster violations** タブを選択します。
3. 詳細を表示するポリシーを 1 つ選択します。
4. **Status** タブを選択して、ポリシー違反を表示します。

2.5.7.2. Pod セキュリティーポリシーの更新

Pod のセキュリティー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。

2.5.7.2.1. Pod セキュリティーポリシーの無効化

Pod のセキュリティーポリシーを無効にするには、以下の手順を実行します。

1. Red Hat Advanced Cluster Management for Kubernetes コンソールにログインします。
2. ナビゲーションメニューから **Govern risk** をクリックし、ポリシー表の一覧を表示します。
3. **Actions** icon > **Disable** の順にクリックして、ポリシーを無効化します。**Disable Policy** ダイアログボックスが表示されます。
4. **Disable policy** をクリックします。

ポリシーが無効化されました。

2.5.7.2.2. Pod 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 Pod セキュリティーポリシーを削除します。

- CLI から Pod 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 Pod セキュリティーポリシーを削除します。

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

     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。

  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 <podsecurity-policy-name> -n <mcm namespace>
     ```

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

  b. 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。

  c. **Remove** をクリックします。

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

Pod セキュリティーポリシーが削除されました。

Pod セキュリティーポリシーの例については、「Pod セキュリティーポリシー」ページの **Pod セキュリティーポリシーの例** 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。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。

2.5.8. ロールポリシーの管理

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。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。以下のセクションでは、ロールポリシーの作成、適用、表示、および更新について説明します。

2.5.8.1. ロールポリシーの作成

コマンドラインインターフェース (CLI) またはコンソールからロールポリシーの 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、ロールポリシーを作成します。

2.5.8.1.1. CLI からのロールポリシーの作成

CLI からロールを作成するには、以下の手順を実行します。

1. 以下のコマンドを実行して、ロールポリシーの YAML ファイルを作成します。

   ```
   kubectl create -f rolepolicy-1.yaml
   ```

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 <role-policy-file-name>  --namespace=<namespace>
   ```

3. 以下のコマンドを実行してポリシーを表示して検証します。

```
kubectl get rolepolicy --namespace=<namespace>
```

CLI からロールポリシーが作成されました。

2.5.8.1.1.1. CLI からのロールポリシーの表示

CLI からロール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、特定のロールポリシーの詳細を表示します。

```
kubectl get rolepolicy <policy-name> -n <namespace> -o yaml
```

2. 以下のコマンドを実行して、ロールポリシーの詳細を表示します。

```
kubectl describe rolepolicy <name> -n <namespace>
```

2.5.8.1.2. コンソールからのロールポリシーの作成

コンソールからロール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。
コンソールからロールポリシーを作成するには、以下の手順を実行します。

1. Red Hat Advanced Cluster Management for Kubernetes コンソールにログインします。
2. ナビゲーションメニューから **Govern risk** をクリックします。
3. **Create policy** をクリックします。
4. **Specifications** フィールドから **Role** を選択します。パラメーターの値は自動的に設定されます。値は編集できます。
5. **Create** をクリックします。

2.5.8.1.2.1. コンソールからのロールポリシーの表示

コンソールからロールポリシーおよびそのステータスを表示できます。

1. コンソールから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 risk** をクリックし、ポリシー表の一覧を表示します。
   注記: ポリシー表の一覧をフィルタリングするには、**Policies** タブまたは **Cluster violations** タブを選択します。
3. 詳細を表示するポリシーを 1 つ選択します。
4. **Status** タブを選択して、ポリシー違反を表示します。

2.5.8.2. ロールポリシーの更新

以下のセクションでは、ロールポリシーの更新について説明します。

2.5.8.2.1. ロールポリシーの無効化

ロールポリシーを無効にするには、以下の手順を実行します。

1. Red Hat Advanced Cluster Management for Kubernetes コンソールにログインします。

2. ナビゲーションメニューから **Govern risk** をクリックし、ポリシー表の一覧を表示します。
3. **Actions** icon > **Disable** の順にクリックして、ポリシーを無効化します。**Disable Policy** ダイアログボックスが表示されます。
4. **Disable policy** をクリックします。

ポリシーが無効化されました。

2.5.8.2.2. ロールポリシーの削除

CLI またはコンソールからロールポリシーを削除します。

- CLI からロールポリシーを削除します。
  a. 以下のコマンドを実行してロールポリシーを削除します。

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

  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。

  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 <podsecurity-policy-name> -n <mcm namespace>
```

- コンソールからロールポリシーを削除します。
  a. ナビゲーションメニューから **Govern risk** をクリックし、ポリシー表の一覧を表示します。
  b. 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。
  c. **Remove** をクリックします。
  d. **Remove policy** ダイアログボックスから、**Remove policy** をクリックします。

ロールポリシーが削除されました。

ロールポリシーの例については、「**ロールポリシー」ページのロールポリシーの例** 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。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。

2.5.9. Rolebinding ポリシーの管理

ロールバインディングポリシーの作成、適用、表示、および更新について説明します。

2.5.9.1. Rolebinding ポリシーの作成

コマンドラインインターフェース (CLI) またはコンソールから rolebinding ポリシーの 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、rolebinding ポリシーを作成します。

2.5.9.1.1. CLI からの rolebinding ポリシーの作成

CLI から rolebinding ポリシーを作成するには、以下の手順を実行します。

1. rolebinding 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 rolebindingpolicy.yaml
```

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 <rolebinding-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 rolebindingpolicy --namespace=<namespace>
```

Rolebinding ポリシーが作成されました。

2.5.9.1.1.1. CLI からの rolebinding ポリシーの表示

CLI から rolebinding 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 rolebinding ポリシーの詳細を表示します。

```
kubectl get rolebindingpolicy <policy-name> -n <namespace> -o yaml
```

2. 以下のコマンドを実行して、rolebinding ポリシーの詳細を表示します。

```
kubectl describe rolebindingpolicy <name> -n <namespace>
```

2.5.9.1.2. コンソールからの rolebinding ポリシーの作成

コンソールから rolebinding 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 rolebinding ポリシーを作成するには、以下の手順を実行します。

1. コンソールから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 **Governance and risk** をクリックします。
3. **Create policy** をクリックします。
4. 以下のフィールドに適切な値を入力するか、または選択します。
   - Name (名前)
   - Specifications (仕様)
   - Cluster selector (クラスターセレクター)
   - Remediation action (修復アクション)
   - Standards (標準)
   - Categories (カテゴリー)
   - Controls (制御)
   - Disabled (無効)
5. **Create** をクリックします。

Rolebinding ポリシーが作成されました。

2.5.9.1.2.1. コンソールからの rolebinding ポリシーの表示

コンソールから rolebinding ポリシーおよびそのステータスを表示できます。

1. コンソールから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、**Governance and risk** をクリックして、ポリシー表の一覧を表示します。
   注記: ポリシー表の一覧をフィルタリングするには、**Policies** タブまたは **Cluster violations** タブを選択します。
3. 詳細を表示するポリシーを 1 つ選択します。
4. **Status** タブを選択して、rolebinding ポリシー違反を表示します。

2.5.9.2. Rolebinding ポリシーの更新

rolebinding 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。

2.5.9.2.1. Rolebinding ポリシーの無効化

Rolebinding ポリシーを無効にするには、以下の手順を実行します。

1. Red Hat Advanced Cluster Management for Kubernetes コンソールにログインします。
2. ナビゲーションメニューから **Govern risk** をクリックし、ポリシー表の一覧を表示します。
3. **Actions** icon > **Disable** の順にクリックして、ポリシーを無効化します。**Disable Policy** ダイアログボックスが表示されます。
4. **Disable policy** をクリックします。

ポリシーが無効化されました。

2.5.9.2.2. Rolebinding ポリシーの削除

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

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

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

  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。

  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 <podsecurity-policy-name> -n <namespace>
  ```

- コンソールから rolebinding ポリシーを削除します。
  a. ナビゲーションメニューから **Govern risk** をクリックし、ポリシー表の一覧を表示します。

b. 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。

c. **Remove** をクリックします。

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

Rolebinding ポリシーが削除されました。

rolebinding ポリシーの例については、「**Rolebinding ポリシー」ページの Rolebinding ポリシー**の例を参照してください。../security/rolebinding_policy.xml#rolebinding-policy-sample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。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。

2.5.10. Security Context Constraints ポリシーの管理

SCC (Security Context Constraints) ポリシーの作成、適用、表示、更新について説明します。

2.5.10.1. SCC ポリシーの作成

コマンドラインインターフェース (CLI) またはコンソールから SCC ポリシーの 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、SCC ポリシーを作成します。

2.5.10.1.1. CLI からの SCC ポリシーの作成

詳細は、OpenShift Container Platform ドキュメントの「SCC (Security Context Constraints) の作成」を参照してください。

2.5.10.1.1.1. CLI からの SCC ポリシーの表示

詳細は、Openshift Container Platform ドキュメントの「SCC の検査」を参照してください。

2.5.10.1.2. コンソールからの SCC ポリシーの作成

コンソールから SCC 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 SCC ポリシーを作成するには、以下の手順を実行します。

1. コンソールから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 **Governance and risk** をクリックします。
3. **Create policy** をクリックします。
4. 以下のフィールドに適切な値を入力するか、または選択します。
   - Name (名前)
   - Specifications (仕様)
   - Cluster selector (クラスターセレクター)
   - Remediation action (修復アクション)
   - Standards (標準)
   - Categories (カテゴリー)

- Controls (制御)
- Disabled (無効)

5. **Create** をクリックします。

SCC ポリシーが作成されました。

2.5.10.1.2.1. コンソールからの SCC ポリシーの表示

コンソールから SCC ポリシーおよびそのステータスを表示できます。

1. コンソールから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、**Governance and risk** をクリックして、ポリシー表の一覧を表示します。
   注記: ポリシー表の一覧をフィルタリングするには、**Policies** タブまたは **Cluster violations** タブを選択します。
3. 詳細を表示するポリシーを 1 つ選択します。
4. **Status** タブを選択して、SCC ポリシー違反を表示します。

2.5.10.2. SCC ポリシーの更新

SCC 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。

2.5.10.2.1. SCC ポリシーの無効化

SCC ポリシーを管理するには、以下の手順を実行します。

1. Red Hat Advanced Cluster Management for Kubernetes コンソールにログインします。
2. ナビゲーションメニューから **Govern risk** をクリックし、ポリシー表の一覧を表示します。
3. **Actions** icon > **Disable** の順にクリックして、ポリシーを無効化します。**Disable Policy** ダイアログボックスが表示されます。
4. **Disable policy** をクリックします。

ポリシーが無効化されました。

2.5.10.2.2. SCC ポリシーの削除

CLI またはコンソールから SCC ポリシーを削除します。

CLI からの SCC ポリシーの削除に関する詳細は、OpenShift Container Platform ドキュメントの「SCC の削除」を参照してください。

- コンソールから SCC ポリシーを削除します。
  a. ナビゲーションメニューから **Govern risk** をクリックし、ポリシー表の一覧を表示します。
  b. 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。

c. **Remove** をクリックします。

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

SCC ポリシーが削除されました。

SCC ポリシーの例については、「Security Context Constraints ポリシー」の「**Security context constraint ポリシーの例**」のセクション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。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。

2.5.11. 証明書ポリシーの管理

証明書ポリシーの作成、適用、表示、および更新について説明します。

2.5.11.1. 証明書ポリシーの作成

コマンドラインインターフェース (CLI) またはコンソールから証明書ポリシーの 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、証明書ポリシーを作成します。

2.5.11.1.1. CLI からの証明書ポリシーの作成

CLI から証明書ポリシーを作成するには、以下の手順を実行します。

1. 証明書ポリシーの 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 policy-1.yaml
```

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 <certificate-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 certificatepolicy --namespace=<namespace>
```

証明書ポリシーが作成されました。

2.5.11.1.1.1. CLI からの証明書ポリシーの表示

CLI から証明書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、特定の証明書ポリシーの詳細を表示します。

```
kubectl get certificatepolicy <policy-name> -n <namespace> -o yaml
```

2. 以下のコマンドを実行して、証明書ポリシーの詳細を表示します。

```
kubectl describe certificatepolicy <name> -n <namespace>
```

2.5.11.1.2. コンソールからの証明書ポリシーの作成

コンソールから証明書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。コンソールから証明書ポリシーを作成するには、以下の手順を実行します。

1. コンソールから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 **Governance and risk** をクリックします。
3. **Create policy** をクリックします。
4. **Specifications** パラメーターに **CertificatePolicy** を選択します。残りのパラメーターの値は、ポリシーを選択すると自動的に設定されます。値は編集できます。
5. **Create** をクリックします。

証明書ポリシーが作成されました。

2.5.11.1.2.1. コンソールからの証明書ポリシーの表示

コンソールから証明書ポリシーおよびそのステータスを表示できます。

1. コンソールから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 risk** をクリックし、ポリシー表の一覧を表示します。**注記**: ポリシー表の一覧をフィルタリングするには、**Policies** タブまたは **Cluster violations** タブを選択します。
3. 詳細を表示するポリシーを 1 つ選択します。**Details** タブ、**Status** タブ、および **YAML** タブが表示されます。
4. ポリシーのコンプライアンスステータスを表示するには、**Status** タブを選択します。**View history** リンクをクリックして、違反メッセージの一覧を表示します。

2.5.11.2. 証明書ポリシーの更新

2.5.11.2.1. 独自の証明書の使用

証明書ポリシーコントローラーを使用して、独自の証明書を監視できます。独自の証明書を監視するには、以下のいずれかの要件を満たす必要があります。

- 証明書の Kubernetes TLS Secret を作成する。
- 証明書を監視するための **certificate_key_name** ラベルを Kubernetes Secret に追加する。

以下のコマンドを実行して Kubernetes TLS Secret を作成し、独自の証明書を監視します。

```
kubectl -n <namespace> create secret tls <secret name> --cert=<path to certificate>/<certificate name> --key=<path to key>/<key name>
```

2.5.11.2.2. ラベルの Kubernetes Secret への追加

certificate_key_name ラベルを追加して、TLS Secret で **metadata** パラメーターを更新します。以下のコマンドを実行して **certificate_key_name** ラベルを追加します。

```
kubectl label secret my-certificate -n default certificate_key_name=cert
```

更新された TLS Secret は以下の内容のようになります。

```
apiVersion: policy.open-cluster-management.io/v1
kind: Secret
metadata:
  name: my-certificate
  namespace: default
  labels:
    certificate_key_name: cert
type: Opaque
data:
  cert: <Certificate Data>
  key: <Private Key Data>
```

注記: コンソールからラベルを追加する場合には、ラベルを TLS Secret YAML ファイルに手作業で追加する必要があります。

2.5.11.2.3. 証明書ポリシーの無効化

証明書ポリシーを作成すると、このポリシーはデフォルトで有効になっています。CLI またはコンソールから証明書ポリシーを無効にするには、以下の手順を実行します。

- コンソールから証明書ポリシーを無効にします。
  a. Red Hat Advanced Cluster Management for Kubernetes コンソールにログインします。
  b. ナビゲーションメニューから **Govern risk** をクリックし、ポリシー表の一覧を表示します。
  c. **Actions** icon > **Disable** の順にクリックして、ポリシーを無効化します。**Disable Policy** ダイアログボックスが表示されます。
  d. **Disable policy** をクリックします。

ポリシーが無効化されました。

2.5.11.2.4. 証明書ポリシーの削除

CLI またはコンソールから証明書ポリシーを削除します。

- CLI から証明書ポリシーを削除します。
  a. 以下のコマンドを実行して証明書ポリシーを削除します。

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

+ 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。

  a. 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 <mcm namespace>
```

- コンソールから証明書ポリシーを削除します。
  a. ナビゲーションメニューから **Govern risk** をクリックし、ポリシー表の一覧を表示します。

b. 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。

c. **Remove** をクリックします。

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

証明書ポリシーが削除されました。

証明書ポリシーの例については、「証明書 **ポリシー コントローラー」ページの 証明書ポリシー の例** を参照してください。他のポリシー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。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。

2.5.12. IAM ポリシーの管理

IAM ポリシーを適用し、マネージドクラスター内で許可されているクラスター管理者の数を確認します。以下のセクションでは、IAM 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。

2.5.12.1. IAM ポリシーの作成

コマンドラインインターフェース (CLI) またはコンソールから IAM ポリシーの YAML ファイルを作成できます。

2.5.12.1.1. CLI からの IAM ポリシーの作成

CLI から IAM ポリシーを作成するには、以下の手順を実行します。

1. IAM ポリシー定義を使用して 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 iam-policy-1.yaml
```

IAM 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: IamPolicy
metadata:
  name: iam-grc-policy
  label:
    category: "System-Integrity"
spec:
  namespaceSelector:
    include: ["default","kube-*"]
    exclude: ["kube-system"]
  remediationAction: inform
  disabled: false
  maxClusterRoleBindingUsers: 5
```

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 <iam-policy-file-name>  --namespace=<mcm_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 <iam-policy-file-name> --namespace=<mcm_namespace>
```

IAM ポリシーが作成されました。

2.5.12.1.1.1. CLI からの IAM ポリシーの表示

IAM 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 IAM ポリシーの詳細を表示します。

```
kubectl get iampolicy <policy-name> -n <namespace> -o yaml
```

2. 以下のコマンドを実行して、IAM ポリシーの詳細を表示します。

```
kubectl describe iampolicy <name> -n <namespace>
```

2.5.12.1.2. コンソールからの IAM ポリシーの作成

コンソールから IAM 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 IAM ポリシーを作成するには、以下の手順を実行します。

1. コンソールから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 risk** をクリックします。
3. **Create policy** をクリックします。
4. **Specifications** フィールドから **IamPolicy** を選択します。残りのパラメーターの値は、ポリシーを選択すると自動的に設定されます。値は編集できます。
5. **Create** をクリックします。

IAM ポリシーが作成されました。

2.5.12.1.2.1. コンソールからの IAM ポリシーの表示

コンソールから IAM ポリシーとそのステータスを表示できます。

1. コンソールから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 risk** をクリックし、ポリシー表の一覧を表示します。
   注記: ポリシー表の一覧をフィルタリングするには、**Policies** タブまたは **Cluster violations** タブを選択します。
3. 詳細を表示するポリシーを 1 つ選択します。
4. **Status** タブを選択して、IAM ポリシー違反を表示します。

2.5.12.2. IAM ポリシーの更新

IAM 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。

2.5.12.2.1. IAM ポリシーの無効化

IAM ポリシーを無効にするには、以下の手順を実行します。

1. Red Hat Advanced Cluster Management for Kubernetes コンソールにログインします。

2. ナビゲーションメニューから **Govern risk** をクリックし、ポリシー表の一覧を表示します。
3. **Actions** icon > **Disable** の順にクリックして、ポリシーを無効化します。**Disable Policy** ダイアログボックスが表示されます。
4. **Disable policy** をクリックします。

ポリシーが無効化されました。

2.5.12.2.2. IAM 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 IAM ポリシーを削除します。
  a. 以下のコマンドを実行し、IAM ポリシーを削除します。

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

  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。

  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 <iam-policy-name> -n <mcm namespace>
```

- コンソールから IAM ポリシーを削除します。
  a. ナビゲーションメニューから **Govern risk** をクリックし、ポリシー表の一覧を表示します。
  b. 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。
  c. **Remove** をクリックします。
  d. **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。

「IAM ポリシーコントローラー」ページの **IAM ポリシーの例** を確認してください。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。

2.5.13. ETCD 暗号化ポリシーの管理

暗号化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。以下のセクションでは、暗号化ポリシーの作成、適用、表示、および更新について説明します。

2.5.13.1. 暗号化ポリシーの作成

暗号化ポリシーの YAML ファイルは、コマンドラインインターフェース (CLI) またはコンソールから作成できます。暗号化ポリシーの作成は、以下のセクションを参照してください。

2.5.13.1.1. CLI からの暗号化ポリシーの作成

CLI から暗号化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、暗号化ポリシーの YAML ファイルを作成します。

```
kubectl create -f etcd-encryption-policy-1.yaml
```

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 <etcd-encryption-policy-file-name>  --namespace=<namespace>
```

3. 以下のコマンドを実行してポリシーを表示して検証します。

```
kubectl get etcd-encryption-policy --namespace=<namespace>
```

CLI から暗号化ポリシーが作成されました。

2.5.13.1.1.1. CLI からの暗号化ポリシーの表示

CLI から暗号化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、特定の暗号化ポリシーの詳細を表示します。

```
kubectl get etcd-encryption-policy <policy-name> -n <namespace> -o yaml
```

2. 以下のコマンドを実行して、暗号化ポリシーの詳細を表示します。

```
kubectl describe etcd-encryption-policy <name> -n <namespace>
```

2.5.13.1.2. コンソールからの暗号化ポリシーの作成

コンソールから暗号化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。コンソールから暗号化ポリシーを作成するには、以下の手順を実行します。

1. Red Hat Advanced Cluster Management for Kubernetes コンソールにログインします。
2. ナビゲーションメニューから **Govern risk** をクリックします。
3. **Create policy** をクリックします。
4. **Specifications** フィールドから **EtcdEncryption** を選択します。残りのパラメーターの値は、ポリシーを選択すると自動的に設定されます。値は編集できます。
5. **Create** をクリックします。

2.5.13.1.2.1. コンソールからの暗号化ポリシーの表示

コンソールから暗号化ポリシーおよびそのステータスを表示できます。

1. コンソールから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 risk** をクリックし、ポリシー表の一覧を表示します。
   注記: ポリシー表の一覧をフィルタリングするには、All policies タブまたは Cluster Violations タブを選択します。
3. 詳細を表示するポリシーを 1 つ選択します。**Overview** タブ、**Status** タブ、および **YAML** タブが表示されます。

2.5.13.2. 暗号化ポリシーの更新

暗号化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。

2.5.13.2.1. 暗号化ポリシーの無効化

暗号化ポリシーを無効にするには、以下の手順を実行します。

1. Red Hat Advanced Cluster Management for Kubernetes コンソールにログインします。
2. ナビゲーションメニューから **Govern risk** をクリックし、ポリシー表の一覧を表示します。
3. **Actions** icon > **Disable** の順にクリックして、ポリシーを無効化します。**Disable Policy** ダイアログボックスが表示されます。
4. **Disable policy** をクリックします。

ポリシーが無効化されました。

2.5.13.2.2. 暗号化ポリシーの削除

CLI またはコンソールから暗号化ポリシーを削除します。

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

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

  + 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。

  a. 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 <podsecurity-policy-name> -n <mcm namespace>
```

- コンソールから暗号化ポリシーを削除します。
  a. ナビゲーションメニューから **Govern risk** をクリックし、ポリシー表の一覧を表示します。
  b. 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。
  c. **Remove** をクリックします。
  d. **Remove policy** ダイアログボックスから、**Remove policy** をクリックします。

暗号化ポリシーが削除されました。

暗号化ポリシーの例を表示するには、ETCD 暗号化ポリシー ページで **ETCD 暗号化ポリシーの例** 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。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。

2.5.14. gatekeeper ポリシーの統合

gatekeeper ポリシーの作成、適用、表示、および更新について説明します。

必要なアクセス権限:: クラスターの管理者

前提条件: Gatekeeper をインストールしておく必要があります。詳細は、**open-policy-agent/gatekeeper リポジトリー** を参照してください。

2.5.14.1. gatekeeper ポリシーの作成

コマンドラインインターフェース (CLI) から gatekeeper ポリシーの YAML ファイルを作成できます。Red Hat Advanced Cluster Management for Kubernetes 設定ポリシーを使用して、ハブクラスターからマネージドクラスターに gatekeeper ポリシーを伝播します。以下のセクションを参照して、受付および監査シナリオの gatekeeper ポリシーを作成します。

2.5.14.1.1. 受付シナリオの gatekeeper ポリシーの作成

Red Hat Advanced Cluster Management 設定ポリシーを使用して gatekeeper ポリシーを作成し、gatekeeper 受付 Webhook が生成するイベントを検索します。

注記: Gatekeeper は、**emit-admission-events** を **true** に設定してデプロイする必要があります。

1. gatekeeper 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 policy-gatekeeper-admission.yaml
```

gatekeeper ポリシー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-gatekeeper
  namespace: default
  annotations:
    policy.open-cluster-management.io/standards:
    policy.open-cluster-management.io/categories:
    policy.open-cluster-management.io/controls:
spec:
  disabled: false
  policy-templates:
    - objectDefinition:
        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"]
            parameters:
              labels: ["gatekeeper"]
- objectDefinition:
    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: gatekeeper-system
              annotations:
                constraint_action: deny
                constraint_kind: K8sRequiredLabels
                constraint_name: ns-must-have-gk
                event_type: violation
```

2.5.14.1.2. 監査シナリオの gatekeeper ポリシーの作成

製品設定ポリシーを使用して gatekeeper ポリシーを作成して、その 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。Red Hat Advanced Cluster Management 設定ポリシーは、gatekeeper 制約の status フィールドで違反をチェックします。

1. gatekeeper 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 policy-gatekeeper-audit.yaml
```

gatekeeper ポリシー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-gatekeeper
  namespace: default
  annotations:
    policy.open-cluster-management.io/standards:
    policy.open-cluster-management.io/categories:
    policy.open-cluster-management.io/controls:
spec:
  disabled: false
  policy-templates:
    - objectDefinition:
        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
                  violations: []
```

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。