Red Hat Advanced Cluster Management for Kubernetes 2.5

监管

了解更多有关监管策略框架的信息，这有助于使用策略强化集群安全性。

Last Updated: 2023-07-14

Red Hat Advanced Cluster Management for Kubernetes 2.5 监管

了解更多有关监管策略框架的信息，这有助于使用策略强化集群安全性。

法律通告

摘要

了解更多有关监管策略框架的信息，这有助于使用策略强化集群安全性。

目录

第 1 章 风险和合规性

管理 Red Hat Advanced Cluster Management for Kubernetes 组件的安全性。使用定义的策略和流程监管集群，以识别并最大程度降低风险。使用策略来定义规则和设置控制。

先决条件：您必须为 Red Hat Advanced Cluster Management for Kubernetes 配置身份验证服务要求。如需更多信息，请参阅 访问控制。

查看以下主题以了解有关保护集群的更多信息：

- 基于角色的访问控制
- 管理凭证概述
- 证书
- 监管
  - 支持配置策略中的模板
  - 完整性 shield 保护（技术预览）

1.1. 证书

在整个 Red Hat Advanced Cluster Management for Kubernetes 范围内会创建和使用各种证书。

您可以自带证书。为您的证书创建 Kubernetes TLS Secret。创建证书后，您可以替换由 Red Hat Advanced Cluster Management 安装程序创建的某些证书。

需要的访问权限： 集群管理员

在 Red Hat Advanced Cluster Management 中运行的服务所需的所有证书都是在 Red Hat Advanced Cluster Management 安装过程中创建的。证书由以下组件创建和管理：

- OpenShift Service Serving 证书
- Red Hat Advanced Cluster Management Webhook 控制器
- Kubernetes 证书 API
- OpenShift 默认入口

继续阅读以了解有关证书管理的更多信息：

Red Hat Advanced Cluster Management hub 集群证书

- 替换管理入口证书
- 替换 OpenShift 默认入口证书
- Observability 证书
  - 使用您自己的 (BYO) 可观察证书认证机构 (CA) 证书
  - 用于生成 CA 证书的 OpenSSL 命令
  - 创建与 BYO observability CA 证书关联的 secret

注意 ：用户负责证书轮转和更新。

1.1.1. Red Hat Advanced Cluster Management hub 集群证书

1.1.1.1. Observability 证书

安装 Red Hat Advanced Cluster Management 后，会创建可观察性证书并供可观察组件使用，以便在 hub 集群和受管集群间的网络流量上提供 mutual TLS。与可观察证书关联的 Kubernetes secret。

open-cluster-management-observability 命名空间包含以下证书：

- **Observability-server-ca-certs**：获取 CA 证书来签署服务器端证书
- **Observability-client-ca-certs**：获取 CA 证书来签署客户端的证书
- **Observability-server-certs**：获取由 **observability-observatorium-api 部署使用的服务器证书**
- **Observability-grafana-certs**：获取 **observability-rbac-query-proxy** 部署使用的客户端证书

open-cluster-management-addon-observability 命名空间在受管集群中包含以下证书：

- **Observability-managed-cluster-certs** ：与 hub 服务器中的 **observability-server-ca-certs** 相同的服务器 CA 证书
- **observability-controller-open-cluster-management.io-observability-signer-client-cert**：由被 **metrics-collector-deployment** 使用的客户证书

CA 证书的有效期为五年，其他证书的有效期为一年。所有可观察证书会在过期后自动刷新。

查看以下列表以了解证书自动更新时的影响：

- 当剩余的有效时间不超过 73 天时，非 CA 证书将自动续订。续订证书后，相关部署中的 Pod 会自动重启以使用更新的证书。

- 当剩余有效时间不超过一年时，CA 证书会被自动续订。证书被续订后，旧的 CA 不会被删除，而是与更新的 CA 共存。相关部署同时使用旧和更新的证书，并继续工作。旧的 CA 证书在过期时会被删除。
- 当证书被续订时，hub 集群和受管集群之间的流量不会中断。

1.1.1.2. 使用您自己的 (BYO) 可观察证书认证机构 (CA) 证书

如果您不想使用 Red Hat Advanced Cluster Management 生成的默认可观察性 CA 证书，您可以在启用可观察性前选择使用 BYO observability CA 证书。

1.1.1.2.1. 用于生成 CA 证书的 OpenSSL 命令

Observability 需要两个 CA 证书；一个用于服务器端，另一个用于客户端。

- 使用以下命令生成您的 CA RSA 私钥：

```
openssl genrsa -out serverCAKey.pem 2048

openssl genrsa -out clientCAKey.pem 2048
```

- 使用私钥生成自签名 CA 证书。运行以下命令：

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

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

1.1.1.2.2. 创建与 BYO observability CA 证书关联的 secret

完成以下步骤以创建 secret：

1. 使用您的证书和私钥创建 **observability-server-ca-certs** secret。运行以下命令：

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

2. 使用您的证书和私钥创建 **observability-client-ca-certs** secret。运行以下命令：

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

1.1.1.2.3. 替换 alertmanager 路由的证书

如果您不想使用 OpenShift 默认入口证书，您可以通过更新 alertmanager 路由来替换 alertmanager 证书。完成以下步骤：

1. 使用以下命令检查可观察证书：

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

2. 将证书上的通用名称 (**CN**) 更改为 **alertmanager**。

3. 使用 alertmanager 路由的主机名更改 **csr.cnf** 配置文件中的 SAN。

4. 在 **open-cluster-management-observability** 命名空间中创建以下两个 secret。运行以下命令：

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

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

如需更多信息，请参阅 OpenSSL 命令以生成证书。如果要恢复 alertmanager 路由的默认自签名证书，请参阅为管理 ingress 恢复默认自签名证书以删除 **open-cluster-management-observability** 命名空间中的两个 secret。

1.1.2. Red Hat Advanced Cluster Management 组件证书

1.1.2.1. 列出 hub 集群受管证书

您可以查看在内部使用 OpenShift Service Serving 证书服务的 hub 集群受管证书列表。运行以下命令列出证书：

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

注：如果启用了可观察性，则还需要额外的命名空间来创建证书。

1.1.2.2. 刷新 hub 集群受管证书

您可以通过运行在 列出 hub 集群受管证书 部分介绍的 **delete secret** 命令来刷新 hub 集群受管证书。当您标识需要刷新的证书时，删除与该证书关联的 secret。例如，您可以运行以下命令删除 secret:

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

注：删除 secret 后会创建一个新 secret。但是，您必须手动重启使用该 secret 的 pod，以便它们能够开始使用新证书。

1.1.2.3. 刷新 Red Hat Advanced Cluster Management Webhook 证书

您可以刷新 OpenShift Container Platform 受管证书，它们是 Red Hat Advanced Cluster Management Webhook 使用的证书。

完成以下步骤以刷新 Red Hat Advanced Cluster Management Webhook 证书：

1. 运行以下命令，删除与 OpenShift Container Platform 受管证书关联的 secret：

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

注： 有些服务可能没有需要删除的 secret。

2. 运行以下命令，重启与 OpenShift Container Platform 受管证书关联的服务：

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

重要 ：许多服务都有副本；每个服务都需要重启。

查看下表，了解包含证书的 pod 概述列表，以及是否需要在重启 pod 前删除 secret 的信息：

表 1.1. 包含 OpenShift Container Platform 受管证书的 Pod

服务名称	Namespace	pod 名称示例	Secret 名称（如果适用）
channels-apps-open-cluster-management-webhook-svc	open-cluster-management	multicluster-operators-application-8c446664c-5lbfk	channels-apps-open-cluster-management-webhook-svc-ca
multicluster-operators-application-svc	open-cluster-management	multicluster-operators-application-8c446664c-5lbfk	multicluster-operators-application-svc-ca
cluster-manager-registration-webhook	open-cluster-management-hub	cluster-manager-registration-webhook-fb7b99c-d8wfc	registration-webhook-serving-cert
cluster-manager-work-webhook	open-cluster-management-hub	cluster-manager-work-webhook-89b8d7fc-f4pv8	work-webhook-serving-cert

1.1.3. Red Hat Advanced Cluster Management 管理的证书

1.1.3.1. 频道证书

CA 证书可与作为 Red Hat Advanced Cluster Management 应用程序管理一部分的 Git 频道关联。如需了解更多详细信息，请参阅使用自定义 CA 证书进行安全 HTTPS 连接。

Helm 频道允许您禁用证书验证。禁用证书验证的 Helm 频道，必须在开发环境中配置。禁用证书验证会带来安全隐患。

1.1.3.2. 受管集群证书

证书用于使用 hub 验证受管集群。因此，了解与这些证书关联的故障排除方案非常重要。如需了解更多详细信息，请参阅证书更改后离线清理的集群。

受管集群证书会自动刷新。

1.1.4. 第三方证书

1.1.4.1. 轮转 gatekeeper Webhook 证书

完成以下步骤以轮转 gatekeeper Webhook 证书：

1. 使用以下命令编辑包含证书的 secret：

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

2. 删除 **data** 部分中的以下内容： **ca.crt**、**ca.key**、tls.crt' 和 **tls.key**。
3. 使用以下命令删除 **gatekeeper-controller-manager** pod 来重启 gatekeeper Webhook 服务：

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

gatekeeper Webhook 证书被轮转。

1.1.4.2. 轮转完整性 shield Webhook 证书（技术预览）

完成以下步骤以轮转完整性 shield Webhook 证书：

1. 编辑 IntegrityShield 自定义资源，把 **integrity-shield-operator-system** 命名空间添加到 **inScopeNamespaceSelector** 设置中的命名空间排除列表中。运行以下命令编辑资源：

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

2. 运行以下命令，删除包含 shield 证书的 secret:

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

3. 删除 Operator，以便重新创建 secret。确保 Operator pod 名称与系统中的 pod 名称匹配。运行以下命令：

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

4. 删除完整 shield 服务器 pod，以使用以下命令开始使用新证书：

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

在受管集群中使用证书策略控制器来创建和管理证书策略。请参阅策略控制器以了解更多有关控制器的信息。返回到风险和合规页面以了解更多信息。

1.2. 替换管理入口证书

如果您不想使用 OpenShift 默认入口证书，您可以通过更新 Red Hat Advanced Cluster Management for Kubernetes 路由来替换管理入口证书。

- 替换管理入口证书的先决条件
- 替换自带 (BYO) 入口证书
- 恢复管理入口的默认自签名证书

1.2.1. 替换管理入口证书的先决条件

将您的 **management-ingress** 证书和私钥准备妥当。如果需要，您可以使用 OpenSSL 生成 TLS 证书。将证书上的通用名称参数 **CN** 设置为 **manangement-ingress**。如果您要生成证书，请加入以下设置：

- 在您的证书主题备用名称（SAN）列表中包括 Red Hat Advanced Cluster Management for Kubernetes 的路由名称作为域名。
  运行以下命令来接收路由名称：

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

  您可能会收到以下响应：

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

1.2.1.1. 用于生成证书的示例配置文件

以下示例配置文件和 OpenSSL 命令提供了有关如何使用 OpenSSL 生成 TLS 证书的示例。查看以下 **csr.cnf** 配置文件，该文件定义了用来使用 OpenSSL 生成证书的配置设置。

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

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

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

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

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

备注：请必使用您的管理入口的正确主机名更新标记的 SAN，即 **DNS.1**。

1.2.1.2. 用于生成证书的 OpenSSL 命令

以下 OpenSSL 命令与上述配置文件一同用来生成所需的 TLS 证书。

1. 生成您的证书颁发机构 (CA) RSA 私钥：

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

2. 使用您的 CA 密钥生成自签名 CA 证书：

```
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. 为您的证书生成 RSA 私钥：

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

4. 使用私钥生成证书签名请求 (CSR)：

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

5. 使用您的 CA 证书和密钥及 CSR 生成签名证书：

```
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. 检查证书内容：

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

1.2.2. 替换自带 (BYO) 入口证书

完成以下步骤以替换您的 BYO 入口证书：

1. 使用您的证书和私钥创建 **byo-ingress-tls** secret。运行以下命令：

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

2. 使用以下命令验证是否在正确的命名空间中创建了 secret：

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

3. 可选：运行以下命令，创建包含 CA 证书的 secret：

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

4. 删除 **management-ingress** 订阅以重新部署订阅。前面步骤中创建的 secret 会被自动使用。运行以下命令：

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

5. 验证当前证书是您的证书，并且所有控制台访问和登录功能保持不变。

1.2.3. 恢复管理入口的默认自签名证书

1. 使用以下命令删除您自己的证书 secret：

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

2. 删除 **management-ingress** 订阅以重新部署订阅。前面步骤中创建的 secret 会被自动使用。运行以下命令：

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

3. 验证当前证书是您的证书，并且所有控制台访问和登录功能保持不变。

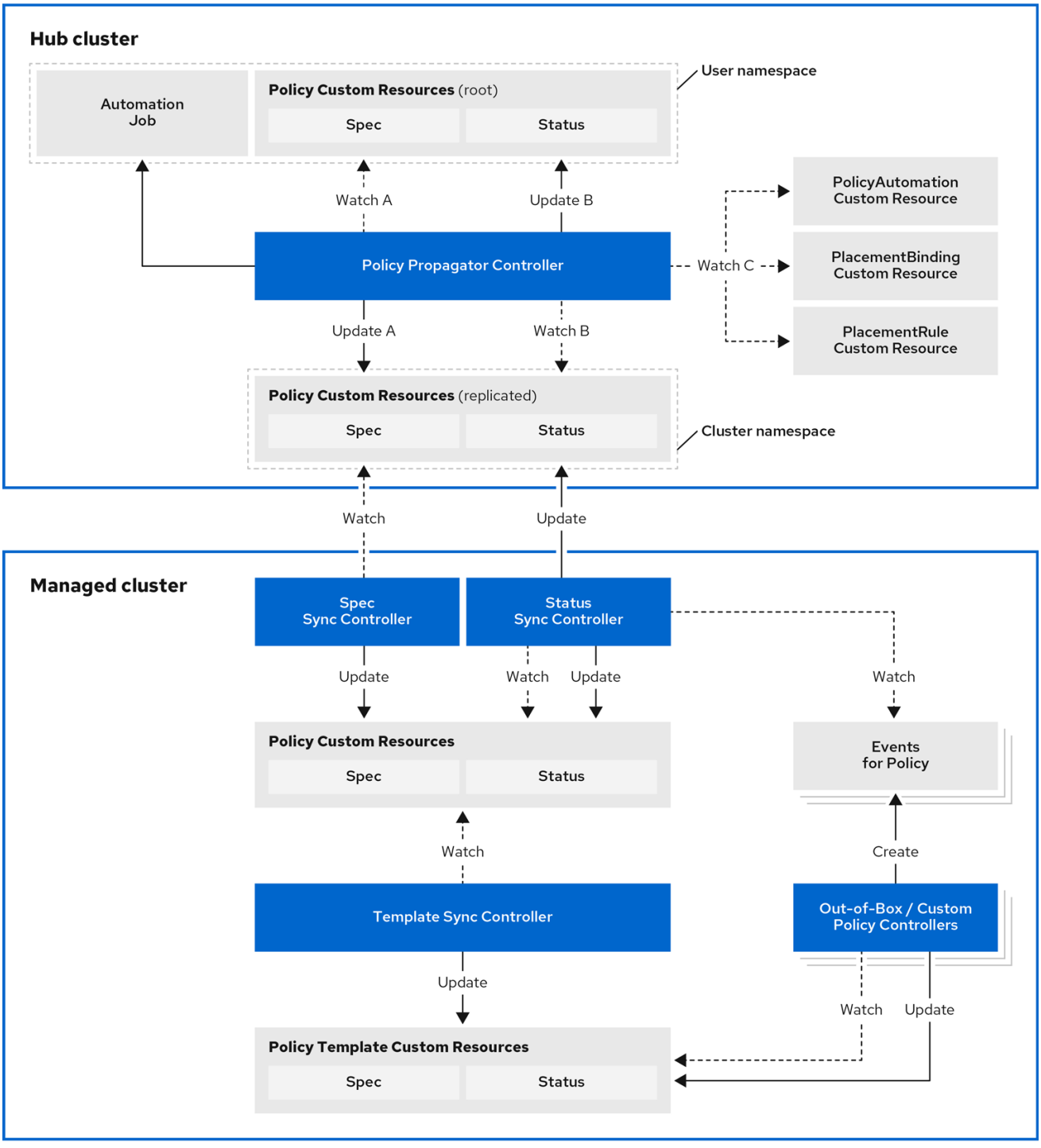
如需有关由 Red Hat Advanced Cluster Management 创建和管理的证书的更多信息，请参阅证书。返回到风险和合规页面以了解更多有关保护集群的信息。

第 2 章 监管

对于在私有云、多云和混合云环境中部署的工作负载，企业必须满足内部对软件工程、安全工程、弹性、安全性以及规范标准的要求。Red Hat Advanced Cluster Management for Kubernetes 监管功能为企业引进自己的安全策略提供了一个可扩展的策略框架。

2.1. 监管架构

使用 Red Hat Advanced Cluster Management for Kubernetes 监管声明周期来增强集群的安全性。产品监管生命周期基于定义的策略、流程和程序，以便可以通过一个中央接口页面来管理安全性和合规性。参阅以下监管架构图：



监管架构由以下组件组成：

- **监管仪表板**：提供云监管和风险详情的概述信息，其中包括策略和集群违反情况。
  备注：
  - 当策略传播到受管集群时，复制策略会命名为 **namespaceName.policyName**。创建策略时，请确保 **namespaceName.policyName** 的长度不得超过 63 个字符，因为 Kubernetes 对对象名称有限制。
  - 当您在 hub 集群中搜索策略时，您可能还会在受管集群上收到复制策略的名称。例如，如果搜索 **policy-dhaz-cert**，则可能出现来自 hub 集群中的以下策略名称：**default.policy-dhaz-cert**。
- **基于策略的监管框架：** 支持根据与集群关联的属性（如一个地区）支持策略创建和部署到各种受管集群。请参阅 **policy-collection** 存储库，以查看预定义的策略示例，以及向集群部署策略的说明。您还可以贡献自定义策略控制器和策略。违反策略时，可将自动化配置为运行并采取用户选择的任何操作。如需更多信息，请参阅配置 Ansible Tower 以了解监管。
  使用 **policy_governance_info** 指标查看趋势并分析任何策略故障。如需了解更多详细信息，请参阅监管指标。
- **策略控制器**：根据您的指定控制在受管集群中评估一个或多个策略，并为违反行为生成 Kubernetes 事件。违反行为会被传播到 hub 集群中。安装中包含的策略控制器如下：Kubernetes 配置、证书和 IAM。您还可以创建自定义策略控制器。
- **开源社区**：在 Red Hat Advanced Cluster Management 策略框架的基础上支持社区贡献。策略控制器和第三方策略也是 **stostron/policy-collection** 存储库的一部分。了解如何使用 GitOps 贡献和部署策略。如需更多信息，请参阅使用 GitOps 部署策略。了解如何将第三方策略与 Red Hat Advanced Cluster Management for Kubernetes 集成。如需更多信息，请参阅集成第三方策略控制器。

了解 Red Hat Advanced Cluster Management for Kubernetes 策略的结构，以及如何使用 Red Hat Advanced Cluster Management for Kubernetes 监管仪表板。

- 策略概述
- 策略控制器
- 支持的策略
- 管理安全策略
- 保护 hub 集群

2.2. 策略概述

使用 Red Hat Advanced Cluster Management for Kubernetes 安全策略框架，以创建自定义策略控制器和其他策略。Kubernetes 自定义资源定义（CRD）实例用于创建策略。有关 CRD 的更多信息，请参阅使用 CustomResourceDefinitions 扩展 Kubernetes API。

每个 Red Hat Advanced Cluster Management for Kubernetes 策略可以至少有一个或多个模板。有关策略元素的更多详情，请参阅本页面的以下 *策略 YAML 表* 部分。

策略需要一个 *PlacementRule* 或 *Placement*，用于定义策略文档应用到的集群，以及将 Red Hat Advanced Cluster Management for Kubernetes 策略绑定到放置规则的 *PlacementBinding*。有关如何定义 **PlacementRule** 的更多信息，请参阅应用程序生命周期文档中的放置规则。有关如何定义 **放置** 的更多信息，请参阅集群生命周期文档中的 放置概述。

重要：

- 您必须创建 **PlacementBinding**，并将它与 **PlacementRule** 或 **Placement** 关联。**最佳实践** ：在使用 **Placement** 资源时，使用命令行界面(CLI) 来更新策略。
- 除集群命名空间外，您可在 hub 集群上的任意命名空间中创建策略。如果在集群命名空间中创建策略，则 Red Hat Advanced Cluster Management for Kubernetes 会将其删除。
- 每个客户端和供应商负责确保其受管云环境满足适用于 Kubernetes 集群上托管的工作负载的内部软件工程、安全工程、弹性、安全性以及合规性企业安全标准。使用监管和安全功能来提高可见性并对配置进行修复，以满足标准。

在以下部分了解更多有关策略组件的详细信息：

- 策略 YAML 结构
- 策略 YAML 表
- 策略示例文件
- 放置 YAML 示例文件

2.2.1. 策略 YAML 结构

创建策略时，必须包含所需的参数字段和值。根据您的策略控制器，您可能需要包含其他可选字段和值。查看解释的参数字段的以下 YAML 结构：

```
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. 策略 YAML 表

字段	描述
apiVersion	必需。将值设置为 **policy.open-cluster-management.io/v1**。
kind	必需。将值设为 **Policy** 以表示策略类型。
metadata.name	必需。用于标识策略资源的名称。
metadata.annotations	可选。用于指定一组描述策略试图验证的标准集合的安全详情。这里介绍的所有注解都以逗号分隔的字符串表示。**注**：您可以在控制台中根据您在*策略*页面上为策略定义的标准和类别查看策略违反。
annotations.policy.open-cluster-management.io/standards	与策略相关的安全标准的名称。例如，美国国家标准与技术研究院 (NIST) 和支付卡行业 (PCI)。
annotations.policy.open-cluster-management.io/categories	安全控制类别是针对一个或多个标准的具体要求。例如，系统和信息完整性类别可能表明您的策略包含一个数据传输协议来保护个人信息，符合 HIPAA 和 PCI 标准的要求。
annotations.policy.open-cluster-management.io/controls	正在接受检查的安全控制名称。例如，证书策略控制器。
spec.policy-templates	必需。用于创建一个或多个应用到受管集群的策略。
spec.disabled	必需。将值设为 **true** 或 **false**。**disabled** 参数提供启用和禁用策略的功能。

字段	描述
spec.remediationAction	可选。指定您的策略的修复。参数值是 **enforce** 和 **inform**。如果指定，定义的 **spec.remediationAction** 值会覆盖子策略中定义的 **remediationAction** 参数，从 **policy-templates** 部分。例如，如果将 **spec.remediationAction** 值部分设定为 **enforce**，那么 **policy-templates** 部分中的 **remediationAction** 会在运行时被设置为 **enforce**。**重要**：有些策略可能不支持 enforce 功能。

2.2.3. 策略示例文件

```
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  name: policy-role
  annotations:
    policy.open-cluster-management.io/standards: NIST SP 800-53
    policy.open-cluster-management.io/categories: AC Access Control
    policy.open-cluster-management.io/controls: AC-3 Access Enforcement
spec:
  remediationAction: inform
  disabled: false
  policy-templates:
    - objectDefinition:
        apiVersion: policy.open-cluster-management.io/v1
        kind: ConfigurationPolicy
        metadata:
          name: policy-role-example
        spec:
          remediationAction: inform # the policy-template spec.remediationAction is overridden by the preceding parameter value for spec.remediationAction.
          severity: high
          namespaceSelector:
            include: ["default"]
          object-templates:
            - complianceType: mustonlyhave # role definition should exact match
              objectDefinition:
                apiVersion: rbac.authorization.k8s.io/v1
                kind: Role
                metadata:
                  name: sample-role
                rules:
                  - apiGroups: ["extensions", "apps"]
                    resources: ["deployments"]
                    verbs: ["get", "list", "watch", "delete","patch"]
---
apiVersion: policy.open-cluster-management.io/v1
kind: PlacementBinding
metadata:
  name: binding-policy-role
```

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

2.2.4. 放置 YAML 示例文件

PlacementBinding 和 **Placement** 资源可以与上一策略示例结合使用，以使用集群 **Placement** API 而非 **PlacementRule** API 部署策略。

```
---
apiVersion: policy.open-cluster-management.io/v1
kind: PlacementBinding
metadata:
  name: binding-policy-role
placementRef:
  name: placement-policy-role
  kind: Placement
  apiGroup: cluster.open-cluster-management.io
subjects:
- name: policy-role
  kind: Policy
  apiGroup: policy.open-cluster-management.io
---
//Depends on if governance would like to use v1beta1
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement-policy-role
spec:
  predicates:
  - requiredClusterSelector:
      labelSelector:
        matchExpressions:
          - {key: environment, operator: In, values: ["dev"]}
```

请参阅管理安全策略以创建和更新策略。您还可以启用和更新 Red Hat Advanced Cluster Management 策略控制器，以验证您的策略合规性。请参阅策略控制器。

要了解更多策略主题，请参阅监管。

2.3. 策略控制器

策略控制器监控并报告集群是否合规。通过开箱即用的策略模板应用预定义策略控制器和策略，以使用 Red Hat Advanced Cluster Management for Kubernetes 策略框架。策略控制器是 Kubernetes 自定义资源定义（CRD）实例。

有关 CRD 的更多信息，请参阅使用 CustomResourceDefinitions 扩展 Kubernetes API。策略控制器会修复策略违反情况，以使集群状态兼容。

您可以使用产品策略框架创建自定义策略和策略控制器。如需更多信息，请参阅 创建自定义策略控制器（已弃用）。

查看以下主题以了解有关以下 Red Hat Advanced Cluster Management for Kubernetes 策略控制器的更多信息：

- Kubernetes 配置策略控制器
- 证书策略控制器
- IAM 策略控制器
- 策略控制器

重要 ：只有配置策略控制器策略支持 **enforce** 功能。当策略控制器不支持 **enforce** 功能时，您必须手动修复策略。

有关管理您的策略的更多主题，请参阅监管。

2.3.1. Kubernetes 配置策略控制器

配置策略控制器可用于配置任何 Kubernetes 资源，并在集群中应用安全策略。

配置策略控制器与本地 Kubernetes API 服务器通信，以获取集群中的配置列表。有关 CRD 的更多信息，请参阅使用 CustomResourceDefinitions 扩展 Kubernetes API。

配置策略控制器是在安装过程中在 hub 集群上创建的。配置策略控制器支持 **enforce** 功能并监控以下策略的合规性：

- 内存用量策略
- 命名空间策略
- 镜像漏洞策略
- Pod 策略
- Pod 安全策略
- 角色策略
- 角色绑定策略
- 安全内容约束 (SCC) 策略
- ETCD 加密策略

- Compliance operator 策略
- 集成 gatekeeper 约束和约束模板

当将配置策略的 **remediationAction** 设置为 **enforce** 时，控制器会在目标受管集群上创建副本策略。您还可以在配置策略中使用模板。如需更多信息，请参阅配置策略中的模板支持。

继续读取以了解更多有关配置策略控制器的信息：

- 配置策略控制器 YAML 结构
- 配置策略示例
- 配置策略 YAML 标

2.3.1.1. 配置策略控制器 YAML 结构

```
Name:         configuration-policy-example
Namespace:
Labels:
APIVersion:   policy.open-cluster-management.io/v1
Kind:         ConfigurationPolicy
Metadata:
  Finalizers:
    finalizer.policy.open-cluster-management.io
Spec:
  Conditions:
    Ownership:
    NamespaceSelector:
      Exclude:
      Include:
    RemediationAction:
 Status:
   CompliancyDetails:
     Configuration-Policy-Example:
       Default:
       Kube - Public:
   Compliant:          Compliant
 Events:
```

2.3.1.2. 配置策略示例

```
apiVersion: policy.open-cluster-management.io/v1
kind: ConfigurationPolicy
metadata:
  name: policy-config
spec:
  namespaceSelector:
    include: ["default"]
    exclude: []
  remediationAction: inform
    severity: low
    object-templates:
    - complianceType: musthave
      objectDefinition:
```

```
      apiVersion: v1
      kind: Pod
      metadata:
        name: pod
      spec:
        containers:
        - image: 'pod-image'
          name:
          ports:
          - containerPort: 80
  evaluationInterval:
    compliant:
    noncompliant:
```

2.3.1.3. 配置策略 YAML 标

表 2.1. 参数表

字段	描述
apiVersion	必需。将值设置为 **policy.open-cluster-management.io/v1**。
kind	必需。将值设为 **ConfigurationPolicy** 以表示策略类型。
metadata.name	必需。策略的名称。
spec	必需。有关监控哪些配置策略以及如何进行修复的规格。
spec.namespace	命名空间对象或资源是必需的。策略应用到的 hub 集群中的命名空间。至少为 **include** 参数输入一个命名空间，这是您要将策略应用到的命名空间。**exclude** 参数指定您明确不希望将策略应用到的命名空间。
spec.remediationAction	必需。指定您的策略的修复。输入 **inform**
spec.remediationAction.severity	必需。当策略不合规时，指定严重性。使用以下参数值： **low**、**medium** 或 **high**。

字段	描述
spec.remediationAction.complianceType	必需。用于列出必须被评估或应用到受管集群的角色的预期行为和任何 Kubernetes 对象。您必须使用以下操作动词作为参数值： **mustonlyhave**：必须存在带有相同的名称和相关项的对象。 **musthave**：必须存在名称与指定 object-template 的名称相同的对象。模板中的其它字段是对象中存在的子集。 **mustnothave**：具有相同名称或标签的对象不能存在，需要删除，而不管规格或规则是什么。
spec.evaluationInterval.compliant	可选。用于定义策略处于合规状态时的评估频率。该值的格式必须是一个持续时间，它是带有时间单元后缀的数字序列。例如：**12h30m5s** 代表 12 小时、30 分钟和 5 秒。另外，也可以将其设定为 **never**，以便策略不会在合规集群中重新评估，除非更新了策略 **spec**。
spec.evaluationInterval.noncompliant	可选。用于定义策略处于不合规状态时的频率。与 **evaluationInterval.compliant** 参数类似，值必须采用持续时间格式，后者是时间后缀的序列。另外，也可以将其设定为 **never**，以便策略不会在不合规的集群中重新评估，除非更新了策略 **spec**。

请参阅 **CM-Configuration-Management** 目录获取使用NIST Special Publication 800-53 (Rev. 4) 的，被 Red Hat Advanced Cluster Management 支持的策略示例。了解策略如何应用到您的 hub 集群，请参阅支持的策略以了解更多详细信息。

了解如何创建和自定义策略，请参阅管理安全策略。有关控制器的详情，请参阅策略控制器。

2.3.2. 证书策略控制器

证书策略控制器可以用来检测快要到期的证书，并检测时间太长（小时）或包含无法与指定模式匹配的 DNS 名称。

通过更新控制器策略中的以下参数来配置和自定义证书策略控制器：

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

由于以下情况之一，您的策略可能会变得不合规：

- 当证书过期的时间少于最短持续时间，或超过最长时间时。
- 当 DNS 名称与指定模式匹配时。

证书策略控制器是在受管集群上创建的。控制器与本地 Kubernetes API 服务器通信，以获取包含证书的 secret 列表，并确定所有不合规的证书。有关 CRD 的更多信息，请参阅使用 CustomResourceDefinitions 扩展 Kubernetes API。

证书策略控制器不支持 **enforce** 功能。

2.3.2.1. 证书策略控制器 YAML 结构

查看证书策略的以下示例，并查看 YAML 表中的元素：

```
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"]
  remediationAction:
  severity:
  minimumDuration:
  minimumCADuration:
  maximumDuration:
  maximumCADuration:
  allowedSANPattern:
  disallowedSANPattern:
```

2.3.2.1.1. 证书策略控制器 YAML 表

表 2.2. 参数表

字段	描述
apiVersion	必需。将值设置为 **policy.open-cluster-management.io/v1**。
kind	必需。将值设为 **CertificatePolicy** 以表示策略类型。
metadata.name	必需。用于标识策略的名称。
metadata.namespace	必需。创建了策略的受管集群内命名空间。

字段	描述
metadata.labels	可选。在证书策略中，**category=system-and-information-integrity** 标签将对策略进行分类，并促进查询证书策略。如果您的证书策略中 **category** 键的值不同，该值会被证书控制器覆盖。
spec	必需。有关要监控和刷新哪些证书的规格。
spec.namespaceSelector	必需。要将策略应用到的受管集群命名空间。输入 **Include** 和 **Exclude** 的参数值。**备注**: • 当您创建多个证书策略并将其应用到同一受管集群时，必须为每个策略 **namespaceSelector** 分配一个不同的值。 • 如果证书策略控制器的 **namespaceSelector** 与任何命名空间不匹配，则该策略被视为合规。
spec.remediationAction	必需。指定您的策略的修复。将参数值设置为 **inform**。证书策略控制器只支持 **inform** 功能。
spec.severity	可选。当策略不合规时，会告知用户的严重性。使用以下参数值： **low**、**medium** 或 **high**。
spec.minimumDuration	必需。如果没有指定值，则默认为 **100h**。参数指定证书被视为不合规前的最短持续时间（以小时为单位）。参数值使用 Golang 的持续时间格式。如需更多信息，请参阅 Golang 解析持续时间。
spec.minimumCADuration	可选。设定一个与其他证书不同的值来标识可能很快过期的签名证书。如果没有指定参数值，则 CA 证书过期时间作为 **minimumDuration** 的值。如需更多信息，请参阅 Golang 解析持续时间。
spec.maximumDuration	可选。指定一个值来标识创建的时间超过您所期望的限制值的证书。参数使用 Golang 的持续时间格式。如需更多信息，请参阅 Golang 解析持续时间。
spec.maximumCADuration	可选。创建一个值来标识创建的时间超过您定义的限制值的签名的证书。参数使用 Golang 的持续时间格式。如需更多信息，请参阅 Golang 解析持续时间。
spec.allowedSANPattern	可选。正则表达式，必须与您证书中定义的每个 SAN 条目匹配。这个参数会根据特征检查 DNS 名称。如需更多信息，请参阅 Golang 郑则表达式语法。

字段	描述
spec.disallowedSANPattern	可选。正则表达式，不能与证书中定义的任何 SAN 条目匹配。这个参数根据模式检查 DNS 名称。 **注意** ：要检测通配符证书，请使用以下 SAN 模式： **disallowedSANPattern: "[*]"** 如需更多信息，请参阅 Golang 郑则表达式语法。

2.3.2.2. 证书策略示例

当在 hub 集群上创建证书策略控制器时，会在受管集群上创建复制策略。请参阅 **policy-certificate.yaml** 查看证书策略示例。

请参阅管理安全策略以了解更多详细信息。如需更多主题，请参阅策略控制器。

2.3.3. IAM 策略控制器

Identity and Access Management (IAM) 策略控制器可以用来接收有关不合规的 IAM 策略的通知。合规性检查是基于您在 IAM 策略中配置的参数。

IAM 策略控制器监控集群中具有特定集群角色（例如 **ClusterRole**）所需的最大数量用户数。要监控的默认集群角色是 **cluster-admin**。IAM 策略控制器与本地 Kubernetes API 服务器通信。如需更多信息，请参阅使用 CustomResourceDefinitions 扩展 Kubernetes API 。

IAM 策略控制器在您的受管集群上运行。查看以下部分以了解更多信息：

- IAM 策略 YAML 结构
- IAM 策略 YAML 表
- IAM 策略示例

2.3.3.1. IAM 策略 YAML 结构

查看 IAM 策略的以下示例，并查看 YAML 表中的参数：

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

2.3.3.2. IAM 策略 YAML 表

查看以下参数表以获详细信息：

表 2.3. 参数表

字段	描述
apiVersion	必需。将值设置为 **policy.open-cluster-management.io/v1**。
kind	必需。将值设为 **Policy** 以表示策略类型。
metadata.name	必需。用于标识策略资源的名称。
spec	必需。添加策略的配置详情。
spec.clusterRole	可选。要监控的集群角色（如 **ClusterRole**）。如果没有指定，则默认为 **cluster-admin**。
spec.severity	可选。当策略不合规时，会告知用户的严重性。使用以下参数值： **low**、**medium** 或 **high**。
spec.remediationAction	可选。指定您的策略的修复。输入 **inform**。
spec.ignoreClusterRoleBindings	可选。正则表达式(regex)值列表，指示要忽略的集群角色绑定名称。这些正则表达式值必须遵循 Go regexp 语法。默认情况下，所有具有以 **system:** 开头的名称的集群角色绑定都将被忽略。建议将其设置为更严格的值。要不忽略任何集群角色绑定名称，请将列表设置为单个值 **.^** 或一些永不匹配的其他正则表达式。
spec.maxClusterRoleBindingUsers	必需。在策略被视为不合规前可用的 IAM rolebinding 的最大数量。

2.3.3.3. IAM 策略示例

请参阅 **policy-limitclusteradmin.yaml** 查看 IAM 策略示例。如需更多信息，请参阅管理安全策略。

如需更多主题，请参阅策略控制器。

2.3.4. 策略控制器

策略控制器将策略状态范围聚合到同一命名空间中定义的策略。创建策略集合(**PolicySet**)，以对同一命名空间中的策略进行分组。**PolicySet** 中的所有策略都放在选定集群中，方法是创建一个 **PlacementBinding** 来绑定 **PolicySet** 和 **Placement**。策略集已部署到 hub 集群。

另外，当策略是多个策略集的一部分时，现有和新的 **Placement** 资源保留在策略中。当用户从策略集合中删除策略时，策略不会应用到在策略集合中选择的集群，但放置会保留。策略控制器只检查包括策略设置放置的集群中的违反情况。

注： Red Hat Advanced Cluster Management 强化示例策略集使用集群放置。如果使用集群放置，请将包含策略的命名空间绑定到受管集群集。有关使用集群放置的详情，请参阅在集群中部署策略。

在以下部分了解更多有关策略设置结构的详细信息：

- 策略控制器控制器 YAML 结构
- 策略控制器控制器 YAML 表
- 策略示例

2.3.4.1. 策略设置 YAML 结构

您的策略集可能类似以下 YAML 文件：

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

---
apiVersion: policy.open-cluster-management.io/v1
kind: PlacementBinding
metadata:
  name: demo-policyset-pb
placementRef:
  apiGroup: apps.open-cluster-management.io
  kind: PlacementRule
  name: demo-policyset-pr
subjects:
- apiGroup: policy.open-cluster-management.io
  kind: PolicySet
  name: demo-policyset
---
apiVersion: apps.open-cluster-management.io
kind: PlacementRule
metadata:
  name: demo-policyset-pr
spec:
  clusterConditions:pagewidth:
  - status: "True"
    type: ManagedCLusterConditionAvailable
  clusterSelectors:
    matchExpressions:
      - key: name
        operator: In
        values:
          - local-cluster
```

2.3.4.2. 策略设置表

查看以下参数表以获详细信息：

表 2.4. 参数表

字段	描述
apiVersion	必需。将值设为 **policy.open-cluster-management.io/v1beta1**。
kind	必需。将值设为 **PolicySet** 以表示策略类型。
metadata.name	必需。用于标识策略资源的名称。
spec	必需。添加策略的配置详情。
spec.policies	可选。要在策略集合中分组的策略列表。

2.3.4.3. 策略示例

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

请参阅管理安全策略部分中的*创建策略*部分。另外，查看 stable **PolicySets**，它要求策略生成器用于部署，PolicySets- Stable。请参阅策略生成器文档。

2.3.5. 创建自定义策略控制器（已弃用）

了解如何编写、应用、查看和更新您的自定义策略控制器。您可以为策略控制器创建 YAML 文件，以部署到集群中。查看以下部分以创建策略控制器：

2.3.5.1. 编写一个策略控制器

使用位于 **Govern-policy-framework** 存储库中的策略控制器框架。完成以下步骤来创建策略控制器：

1. 运行以下命令克隆 **governance-policy-framework** 存储库：

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

2. 通过更新策略模式定义自定义控制器策略。您的策略可能类似以下内容：

   ```
   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. 更新策略控制器以监视 **SamplePolicy** kind。运行以下命令：

```
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. 通过完成以下步骤重新编译并运行策略控制器：

   a. 登录到您的集群。

   b. 选择用户图标，然后点击 **Configure client**。

   c. 将配置信息复制并粘贴到您的命令行中，然后按 **Enter** 键。

   d. 运行以下命令以应用您的策略 CRD 并启动控制器：

```
export GO111MODULE=on

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

export WATCH_NAMESPACE=<cluster_namespace_on_hub>

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

   您可能会收到以下输出来表示控制器在运行：

```
{“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. 创建策略，验证控制器是否已检索策略，并在集群中应用策略。运行以下命令：

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

   应用策略时，会出现一条消息来指示您的自定义控制器监控并检测到策略。这个信息可能类似以下内容：

```
{"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. 运行以下命令，检查 **status** 字段中的合规详情：

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

您的输出可能类似以下内容：

```
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. 更改策略规则和策略逻辑，为您的策略控制器引入新规则。完成以下步骤：

   a. 通过更新 **SamplePolicySpec** 在 YAML 文件中添加新字段 。您的规格应该和以下类似：

```
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. 使用新字段在 samplepolicy_controller.go 中更新 **SamplePolicySpec** 的数据结构。

   c. 使用新的路径更新 **samplepolicy_controller.go** 文件中的 **PeriodicallyExecSamplePolicies** 函数来运行策略控制器。查看 **PeriodicallyExecSamplePolicies** 字段的示例，请参阅 stolostron/multicloud-operators-policy-controller。

d. 重新编译并运行策略控制器。请参阅编写策略控制器

您的策略控制器可以正常工作。

2.3.5.2. 将控制器部署到集群中

将自定义策略控制器部署到集群，并将策略控制器与*监管*仪表板集成。完成以下步骤：

1. 运行以下命令构建策略控制器镜像：

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

2. 运行以下命令将镜像推送到您选择的仓库中。例如，运行以下命令将镜像 push 到 Docker Hub:

```
docker login

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

3. 配置 **kubectl** 以指向由 Red Hat Advanced Cluster Management for Kubernetes 管理的集群。

4. 替换 operator 清单以使用内置镜像名称，并更新命名空间以监视策略。命名空间必须是集群的命名空间。您的清单可能类似以下内容：

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

5. 运行以下命令更新 RBAC 角色：

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

6. 将策略控制器部署到集群中：

   a. 运行以下命令为集群设置服务帐户：

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

   b. 运行以下命令来为 operator 创建 RBAC：

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

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

   c. 为您的策略控制器设置 RBAC。运行以下命令：

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

   d. 运行以下命令来设置自定义资源定义（CRD）：

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

e. 运行以下命令来部署 **multicloud-operator-policy-controller**:

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

f. 运行以下命令验证控制器是否正常工作：

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

7. 您必须通过为控制器创建一个 **策略模板** 来集成您的策略控制器。如需更多信息，请参阅通过控制台创建集群安全策略。

2.3.5.2.1. 扩展控制器部署

策略控制器部署不支持删除。您可以扩展部署，以更新部署应用到哪些 pod。完成以下步骤：

1. 登录到受管集群。
2. 导航到自定义策略控制器的部署。
3. 扩展部署。当将部署扩展到零个 pod 时，策略控制程序部署将被禁用。

如需有关部署的更多信息，请参阅 OpenShift Container Platform 部署。

您的策略控制器已部署并在集群中集成。如需更多信息，请参阅策略控制器。

2.4. 集成第三方策略控制器

集成第三方策略，在策略模板中创建自定义注解，以指定一个或多个合规标准、控制类别和控制。

您还可以使用来自 policy-collection/community 中的第三方策略。

了解如何集成以下第三方策略：

- 集成 gatekeeper 约束和约束模板
- 策略生成器

2.4.1. 集成 gatekeeper 约束和约束模板

Gatekeeper 是一个验证 webhook，它强制执行基于自定义资源定义（CRD）的策略，该策略与 Open Policy Agent（OPA）一起运行。您可以使用 gatekeeper operator 策略在集群中安装 gatekeeper。Gatekeeper 策略可用于评估 Kubernetes 资源合规性。您可以使用 OPA 作为策略引擎，并使用 Rego 作为策略语言。

gatekeeper 策略在 Red Hat Advanced Cluster Management 中作为 Kubernetes 配置策略创建。Gatekeeper 策略包括约束模板（**ConstraintTemplates**）和 **Constraints**、审计模板和准入模板。如需更多信息，请参阅 Gatekeeper 上游存储库。

Red Hat Advanced Cluster Management 支持 Gatekeeper 版本 3.3.0，并在 Red Hat Advanced Cluster Management gatekeeper 策略中应用以下约束模板：

- **ConstraintTemplates** 和约束： 使用 **policy-gatekeeper-k8srequiredlabels** 策略在受管集群上创建 gatekeeper 约束模板。

```
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 模板： 使用 **policy-gatekeeper-audit** 定期检查并评估为检测现有错误配置而强制执行的门管理器策略的现有资源。

```
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 模板： 使用 **policy-gatekeeper-admission** 检查由 gatekeeper admission webhook 创建的错误配置：

```
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
```

如需了解更多详细信息，请参阅 **policy-gatekeeper-sample.yaml**。

有关管理其他策略的更多信息，请参阅管理配置策略。有关安全框架的更多主题，请参阅监管。

2.4.2. 策略生成器

策略生成器是 Red Hat Advanced Cluster Management for Kubernetes 应用程序生命周期订阅 GitOps 工作流的一部分，该工作流使用 Kustomize 生成 Red Hat Advanced Cluster Management for Kubernetes 策略。策略生成器从 Kubernetes 清单 YAML 文件构建 Red Hat Advanced Cluster Management for Kubernetes 策略，该文件通过用于配置它的 **PolicyGenerator** 清单 YAML 文件提供。策略生成器作为 Kustomize 生成器插件实施。有关 Kustomize 的更多信息，请参阅 Kustomize 文档。

此 Red Hat Advanced Cluster Management 版本捆绑的策略生成器版本是 v1.8.0。

2.4.2.1. 策略生成器功能

策略生成器及其与 Red Hat Advanced Cluster Management 应用程序生命周期 订阅 GitOps 工作流的集成简化了 Kubernetes 资源对象的分发到受管 OpenShift 集群，并通过 Red Hat Advanced Cluster Management 策略简化了 Kubernetes 集群的分发。特别是，使用策略生成器完成以下操作：

- 将任何 Kubernetes 清单文件转换为 Red Hat Advanced Cluster Management 配置策略。
- 在将输入 Kubernetes 清单插入到生成的 Red Hat Advanced Cluster Management 策略前对其进行补丁。
- 生成额外的配置策略，以便能够通过 Red Hat Advanced Cluster Management for Kubernetes 报告 Gatekeeper 和 Kyverno 策略违反情况。
- 在 hub 集群上生成策略集。如需了解更多详细信息，请参阅策略设置控制器。

如需更多信息，请查看以下主题：

- 策略生成器配置结构
- 生成用于安装 Operator 的策略
  - 安装 OpenShift GitOps 的策略
  - 安装 Compliance Operator 的策略
- 在 OpenShift GitOps (ArgoCD) 上安装策略生成器
- 策略生成器配置参考表

2.4.2.2. 策略生成器配置结构

策略生成器是一个 Kustomize 生成器插件，它配置了一个 **PolicyGenerator** kind 和 **policy.open-cluster-management.io/v1** API 版本的清单。

要使用该插件，请首先在 **kustomization.yaml** 文件中添加一个 **generators** 部分。查看以下示例：

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

上例中引用的 **policy-generator-config.yaml** 文件是一个 YAML 文件，其中包含要生成的策略的说明。简单的策略生成器配置文件可能类似以下示例：

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

configmap.yaml 代表要包含在策略中的 Kubernetes 清单 YAML 文件。查看以下示例：

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

生成的 **Policy** 以及生成的 **PlacementRule** 和 **PlacementBinding** 可能类似以下示例：

```
apiVersion: apps.open-cluster-management.io/v1
kind: PlacementRule
metadata:
  name: placement-config-data
  namespace: policies
spec:
  clusterConditions:
  - status: "True"
    type: ManagedClusterConditionAvailable
  clusterSelector:
    matchExpressions: []
---
apiVersion: policy.open-cluster-management.io/v1
kind: PlacementBinding
metadata:
  name: binding-config-data
  namespace: policies
placementRef:
  apiGroup: apps.open-cluster-management.io
  kind: PlacementRule
  name: placement-config-data
subjects:
- apiGroup: policy.open-cluster-management.io
  kind: Policy
  name: config-data
---
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  annotations:
    policy.open-cluster-management.io/categories: CM Configuration Management
    policy.open-cluster-management.io/controls: CM-2 Baseline Configuration
    policy.open-cluster-management.io/standards: NIST SP 800-53
  name: config-data
  namespace: policies
spec:
  disabled: false
  policy-templates:
  - objectDefinition:
      apiVersion: policy.open-cluster-management.io/v1
      kind: ConfigurationPolicy
      metadata:
```

```
    name: config-data
  spec:
    object-templates:
    - complianceType: musthave
      objectDefinition:
        apiVersion: v1
        data:
          key1: value1
          key2: value2
        kind: ConfigMap
        metadata:
          name: my-config
          namespace: default
    remediationAction: inform
    severity: low
```

如需了解更多详细信息，请参阅 **policy-generator-plugin** 存储库。

2.4.2.3. 生成用于安装 Operator 的策略

Red Hat Advanced Cluster Management 策略的一个常见用途是在一个或多个受管 OpenShift 集群上安装 Operator。查看以下不同安装模式和所需资源的示例。

2.4.2.3.1. 安装 OpenShift GitOps 的策略

本例演示了如何生成一个策略，以使用策略生成器安装 OpenShift GitOps。OpenShift GitOps 操作器 *提供所有命名空间安装模式*。首先，需要按照以下示例创建名为 **openshift-gitops-subscription.yaml** 的订阅清单文件。

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

要固定到 Operator 的特定版本，您可以添加以下参数和值： **spec.startingCSV: openshift-gitops-operator.v<version>**。将 **<version>** 替换为您的首选版本。

接下来，需要名为 **policy-generator-config.yaml** 的策略生成器配置文件。以下示例显示了一个策略，它将在所有 OpenShift 受管集群上安装 OpenShift GitOps：

```
apiVersion: policy.open-cluster-management.io/v1
kind: PolicyGenerator
metadata:
  name: install-openshift-gitops
policyDefaults:
  namespace: policies
  placement:
    clusterSelectors:
      vendor: "OpenShift"
```

```
  remediationAction: enforce
policies:
  - name: install-openshift-gitops
    manifests:
      - path: openshift-gitops-subscription.yaml
```

所需的最后一个文件是 **kustomization.yaml** 文件。**kustomization.yaml** 文件需要以下配置：

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

生成的策略可能类似以下文件：

```
apiVersion: apps.open-cluster-management.io/v1
kind: PlacementRule
metadata:
  name: placement-install-openshift-gitops
  namespace: policies
spec:
  clusterConditions:
    - status: "True"
      type: ManagedClusterConditionAvailable
  clusterSelector:
    matchExpressions:
      - key: vendor
        operator: In
        values:
          - OpenShift
---
apiVersion: policy.open-cluster-management.io/v1
kind: PlacementBinding
metadata:
  name: binding-install-openshift-gitops
  namespace: policies
placementRef:
  apiGroup: apps.open-cluster-management.io
  kind: PlacementRule
  name: placement-install-openshift-gitops
subjects:
  - apiGroup: policy.open-cluster-management.io
    kind: Policy
    name: install-openshift-gitops
---
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  annotations:
    policy.open-cluster-management.io/categories: CM Configuration Management
    policy.open-cluster-management.io/controls: CM-2 Baseline Configuration
    policy.open-cluster-management.io/standards: NIST SP 800-53
  name: install-openshift-gitops
  namespace: policies
spec:
  disabled: false
  policy-templates:
```

```
    - objectDefinition:
        apiVersion: policy.open-cluster-management.io/v1
        kind: ConfigurationPolicy
        metadata:
          name: install-openshift-gitops
        spec:
          object-templates:
            - complianceType: musthave
              objectDefinition:
                apiVersion: operators.coreos.com/v1alpha1
                kind: Subscription
                metadata:
                  name: openshift-gitops-operator
                  namespace: openshift-operators
                spec:
                  channel: stable
                  name: openshift-gitops-operator
                  source: redhat-operators
                  sourceNamespace: openshift-marketplace
          remediationAction: enforce
          severity: low
```

所有输入来自 OpenShift Container Platform 文档中的策略，并由策略生成器生成。查看 OpenShift Container Platform 文档中的以下 YAML 输入示例：

- 安装后集群任务
- 配置审计日志策略
- 关于将日志转发到第三方系统

如需了解更多详细信息，请参阅 了解 OpenShift GitOps 和 Operator 文档。

2.4.2.3.2. 安装 Compliance Operator 的策略

对于使用 *命名空间安装模式的* Operator，如 Compliance Operator，还需要 **OperatorGroup** 清单。本例演示了安装 Compliance Operator 的生成的策略。

首先，必须创建一个带有 **Namespace**、**Subscription** 和名为 **compliance-operator.yaml** 的 **OperatorGroup** 清单的 YAML 文件。以下示例将这些清单安装到 **compliance-operator** 命名空间中：

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

```
---
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: compliance-operator
  namespace: openshift-compliance
spec:
  targetNamespaces:
    - compliance-operator
```

接下来，需要名为 **policy-generator-config.yaml** 的策略生成器配置文件。以下示例显示了在一个 OpenShift 受管集群上安装 Compliance Operator 的单一策略：

```
apiVersion: policy.open-cluster-management.io/v1
kind: PolicyGenerator
metadata:
  name: install-compliance-operator
policyDefaults:
  namespace: policies
  placement:
    clusterSelectors:
      vendor: "OpenShift"
  remediationAction: enforce
policies:
  - name: install-compliance-operator
    manifests:
      - path: compliance-operator.yaml
```

所需的最后一个文件是 **kustomization.yaml** 文件。**kustomization.yaml** 文件中需要以下配置：

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

因此，生成的策略应类似以下文件：

```
apiVersion: apps.open-cluster-management.io/v1
kind: PlacementRule
metadata:
  name: placement-install-compliance-operator
  namespace: policies
spec:
  clusterConditions:
    - status: "True"
      type: ManagedClusterConditionAvailable
  clusterSelector:
    matchExpressions:
      - key: vendor
        operator: In
        values:
          - OpenShift
---
apiVersion: policy.open-cluster-management.io/v1
kind: PlacementBinding
metadata:
  name: binding-install-compliance-operator
```

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

```
        - compliance-operator
    remediationAction: enforce
    severity: low
```

如需了解更多详细信息，请参阅 Compliance Operator 文档。

2.4.2.4. 在 OpenShift GitOps (ArgoCD) 上安装策略生成器

基于 ArgoCD 的 OpenShift GitOps 也可用于通过 GitOps 使用策略生成器生成策略。由于 OpenShift GitOps 容器镜像中没有预安装策略生成器，因此需要进行一些自定义。为了继续操作，预计在 Red Hat Advanced Cluster Management hub 集群中安装了 OpenShift GitOps Operator，并确保登录到 hub 集群。

为了使 OpenShift GitOps 在运行 Kustomize 时可以访问策略生成器，需要初始容器将 Red Hat Advanced Cluster Management Application Subscription 容器镜像中的策略生成器二进制文件复制到 OpenShift GitOps 容器，该运行 Kustomize。如需了解更多详细信息，请参阅在部署 pod 前使用初始容器执行任务。另外，OpenShift GitOps 必须被配置为在运行 Kustomize 时提供 **--enable-alpha-plugins** 标志。使用以下命令开始编辑 OpenShift GitOps **argocd** 对象：

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

然后，修改 OpenShift GitOps **argocd** 对象使其包含以下额外的 YAML 内容。当发布新的 Red Hat Advanced Cluster Management 主版本且您要将策略生成器更新至更新的版本时，您需要更新 **registry.redhat.io/rhacm2/multicluster-operators-subscription-rhel8** 镜像，供 Init 容器用于较新的标签。查看以下示例，将 **<version>** 替换为 **2.5** 或所需的 Red Hat Advanced Cluster Management 版本：

```
apiVersion: argoproj.io/v1alpha1
kind: ArgoCD
metadata:
  name: openshift-gitops
  namespace: openshift-gitops
spec:
  kustomizeBuildOptions: --enable-alpha-plugins
  repo:
    env:
    - name: KUSTOMIZE_PLUGIN_HOME
      value: /etc/kustomize/plugin
    initContainers:
    - args:
      - -c
      - cp /etc/kustomize/plugin/policy.open-cluster-management.io/v1/policygenerator/PolicyGenerator
        /policy-generator/PolicyGenerator
      command:
      - /bin/bash
      image: registry.redhat.io/rhacm2/multicluster-operators-subscription-rhel8:v<version>
      name: policy-generator-install
      volumeMounts:
      - mountPath: /policy-generator
        name: policy-generator
    volumeMounts:
    - mountPath: /etc/kustomize/plugin/policy.open-cluster-management.io/v1/policygenerator
      name: policy-generator
    volumes:
    - emptyDir: {}
      name: policy-generator
```

现在，OpenShift GitOps 可以使用策略生成器，OpenShift GitOps 必须被授予在 Red Hat Advanced Cluster Management hub 集群中创建策略的访问权限。创建以下 **ClusterRole** 资源，名为 **openshift-gitops-policy-admin**，它有权创建、读取、更新和删除策略和放置。您的 **ClusterRole** 可能类似以下示例：

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

另外，创建一个 **ClusterRoleBinding** 对象来授予 OpenShift GitOps 服务帐户对 **openshift-gitops-policy-admin ClusterRole** 的访问权限。**ClusterRoleBinding** 可能类似以下资源：

```
kind: ClusterRoleBinding
```

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

2.4.2.5. 策略生成器配置参考表

请注意，每个策略可覆盖 **policyDefaults** 部分中除 **namespace** 以外的所有字段。

字段	描述
apiVersion	必需。将值设置为 **policy.open-cluster-management.io/v1**。
complianceType	可选。决定在将清单与集群上对象进行比较时的策略控制器行为。参数值为 **musthave**, **mustonlyhave**, 或 **mustnothave**.默认值为 **musthave**。
kind	必需。将值设为 **PolicyGenerator** 以表示策略类型。
metadata	必需。用于唯一标识配置文件。
metadata.name	必需。用于标识策略资源的名称。
placementBindingDefaults	必需。用于整合 **PlacementBinding** 中的多个策略，以便生成器能够使用定义的名称创建唯一的 **PlacementBinding** 名称。
placementBindingDefaults.name	可选。最佳实践是设置要使用的显式放置绑定名称，而不使用默认值。
policyDefaults	必需。对于 policies 数组中的条目，这里列出的任何默认值都会被覆盖，但 **namespace** 除外。
policyDefaults.categories	可选。**policy.open-cluster-management.io/categories** 注解中使用的类别数组。默认值为 **CM 配置管理**。
policyDefaults.controls	可选。**policy.open-cluster-management.io/controls** 注解中使用的控制数组。默认值为 **CM-2 Baseline Configuration**。

字段	描述
policyDefaults.consolidateManifests	可选。这决定了是否为策略中包含的所有清单生成单一配置策略。如果设置为 **false**，则为每个清单生成配置策略。默认值为 **true**。
policyDefaults.informGatekeeperPolicies	可选。当策略引用违反 gatekeeper 策略清单时，这决定了是否应生成额外的配置策略以便在 Red Hat Advanced Cluster Management 中接收策略违反情况。默认值为 **true**。
policyDefaults.informKyvernoPolicies	可选。当策略引用 Kyverno 策略清单时，这决定了在 Kyverno 策略被违反时，是否应生成额外的配置策略来接收 Red Hat Advanced Cluster Management 中的策略违反情况。默认值为 **true**。
policyDefaults.namespace	必需。所有策略的命名空间。
policyDefaults.placement	可选。策略的放置配置。这默认为与所有集群匹配的放置配置。
placement.clusterSelectors	可选。通过以以下格式 **key:value** 定义集群选择器来指定放置。请参阅 **placementRulePath** 以指定现有文件。
placement.name	可选。指定一个名称来整合包含相同集群选择器的放置规则。
placement.placementRulePath	可选。要重复使用现有放置规则，请指定相对于 **kustomization.yaml** 文件的路径。如果提供，则默认所有策略都使用此放置规则。请参阅 **clusterSelectors** 以生成新 **放置**。
policyDefaults.remediationAction	可选。策略的补救机制。参数值是 **enforce** 和 **inform**。默认值是 **inform**。
policyDefaults.severity	可选。策略违反的严重性。默认值为 **low**。
policyDefaults.standards	可选。**policy.open-cluster-management.io/standards** 注解中使用的一组标准。默认值为 **NIST SP 800-53**。
policies	必需。要创建的策略列表以及覆盖默认值或 **policyDefaults** 中设置的值。
policies[ ].manifests	必需。策略中包含的 Kubernetes 对象清单列表。
policies[ ].name	必需。要创建的策略的名称。

字段	描述
policies[ ].manifests[ ].complianceType	可选。决定在将清单与集群上对象进行比较时的策略控制器行为。参数值为 **musthave**, **mustonlyhave**, 或 **mustnothave**.默认值为 **musthave**。
policies[ ].manifests[ ].path	必需。与 **kustomization.yaml** 文件相关的单个文件或平面文件目录的路径。
policies[ ].manifests[ ].patches	可选。应用到路径上清单的 Kustomize 补丁程序。如果有多个清单，补丁需要设置 **apiVersion**、**kind**、**metadata.name** 和 **metadata.namespace** （如果适用）字段，以便 Kustomize 可以识别补丁应用到的清单。如果只有一个清单，则可以修补 **metadata.name** 和 **metadata.namespace** 字段。

2.5. 支持的策略

查看支持的策略示例，了解如何在 Red Hat Advanced Cluster Management for Kubernetes 中创建和管理策略时如何在 hub 集群上定义规则、流程和控制。

注：您可以将现有策略复制到 *Policy YAML* 中。当您粘贴现有策略时，会自动输入参数字段的值。您还可以使用搜索功能搜索策略 YAML 文件中的内容。

2.5.1. 开箱即用策略的支持列表

策略	Red Hat OpenShift Container Platform 3.11	Red Hat OpenShift Container Platform 4
内存用量策略	x	x
命名空间策略	x	x
镜像漏洞策略	x	x
Pod 策略	x	x
Pod 安全策略（已弃用）		
角色策略	x	x
角色绑定策略	x	x
安全性上下文约束策略(SCC)	x	x
ETCD 加密策略		x

策略	Red Hat OpenShift Container Platform 3.11	Red Hat OpenShift Container Platform 4
Gatekeeper 策略		x
Compliance operator 策略		x
E8 扫描策略		x
OpenShift CIS 扫描策略		x
策略设置		x
Kyverno 添加网络		x
Kyverno 添加配额		x
Kyverno 同步 secret		x

查看以下策略示例以查看如何应用特定策略：

- 镜像漏洞策略
- 内存用量策略
- 命名空间策略
- Pod 策略
- Pod 安全策略
- 角色策略
- 角色绑定策略
- 安全性上下文约束策略
- ETCD 加密策略
- Compliance operator 策略
- E8 扫描策略
- OpenShift CIS 扫描策略
- 策略控制器
- Kyverno 添加网络策略
- Kyverno 添加配额策略
- Kyverno sync secret 策略

更多主题，请参阅监管。

2.5.2. 内存用量策略

Kubernetes 配置策略控制器负责监控内存用量策略的状态。使用内存用量策略来限制或约束您的内存和计算用量。如需更多信息，请参阅 Kubernetes 文档中的*限制范围*。

在以下部分了解更多有关内存用量策略结构的详细信息：

- 内存用量策略 YAML 结构
- 内存用量策略表
- 内存用量策略示例

2.5.2.1. 内存用量策略 YAML 结构

您的内存用量策略可能类似以下 YAML 文件：

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

2.5.2.2. 内存用量策略表

字段	描述
apiVersion	必需。将值设置为 **policy.open-cluster-management.io/v1**。

字段	描述
kind	必需。将值设为 **Policy** 以表示策略类型。
metadata.name	必需。用于标识策略资源的名称。
metadata.namespaces	可选。
spec.namespace	必需。策略应用到的节点集群内命名空间。输入 **include** 的参数值，这是您要将策略应用到的命名空间。**exclude** 参数指定您明确不希望将策略应用到的命名空间。**注**：在策略控制器的对象模板中指定的命名空间会覆盖对应父策略中的命名空间。
remediationAction	可选。指定您的策略的修复。参数值是 **enforce** 和 **inform**。
disabled	必需。将值设为 **true** 或 **false**。**disabled** 参数提供启用和禁用策略的功能。
spec.complianceType	必需。将值设为 **"musthave"**
spec.object-template	可选。用于列出必须接受评估或应用到受管集群的任何其他 Kubernetes 对象。

2.5.2.3. 内存用量策略示例

请参阅 **policy-limitmemory.yaml** 查看策略示例。如需了解更多详细信息，请参阅管理安全策略。请参阅 Kubernetes 配置策略控制器以查看控制器监控的其他配置策略。

2.5.3. 命名空间策略

Kubernetes 配置策略控制器负责监控命名空间策略的状态。应用命名空间策略来为您的命名空间定义特定规则。

在以下部分了解更多有关命名空间策略结构的详细信息：

- 命名空间策略 YAML 结构
- 命名空间策略 YAML 表
- 命名空间策略示例

2.5.3.1. 命名空间策略 YAML 结构

```
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  name: policy-namespace-1
  namespace:
spec:
```

```
complianceType:
remediationAction:
namespaces:
  exclude:
  include:
object-templates:
  - complianceType:
    objectDefinition:
      kind:
      apiVersion:
      metadata:
        name:
  ...
```

2.5.3.2. 命名空间策略 YAML 表

字段	描述
apiVersion	必需。将值设置为 **policy.open-cluster-management.io/v1**。
kind	必需。将值设为 **Policy** 以表示策略类型。
metadata.name	必需。用于标识策略资源的名称。
metadata.namespaces	可选。
spec.namespace	必需。策略应用到的节点集群内命名空间。输入 **include** 的参数值，这是您要将策略应用到的命名空间。**exclude** 参数指定您明确不希望将策略应用到的命名空间。**注**：在策略控制器的对象模板中指定的命名空间会覆盖对应父策略中的命名空间。
remediationAction	可选。指定您的策略的修复。参数值是 **enforce** 和 **inform**。
disabled	必需。将值设为 **true** 或 **false**。**disabled** 参数提供启用和禁用策略的功能。
spec.complianceType	必需。将值设为 **"musthave"**
spec.object-template	可选。用于列出必须接受评估或应用到受管集群的任何其他 Kubernetes 对象。

2.5.3.3. 命名空间策略示例

请参阅 **policy-namespace.yaml** 查看策略示例。

如需了解更多详细信息，请参阅管理安全策略。请参阅 Kubernetes 配置策略控制器以了解其他配置策略。

2.5.4. 镜像漏洞策略

应用镜像漏洞策略，以利用 Container Security Operator 来检测容器镜像是否有漏洞。如果没有安装 Container Security Operator，该策略会在受管集群上安装它。

镜像漏洞策略由 Kubernetes 配置策略控制器负责检查。有关 Security Operator 的更多信息，请参阅 Quay 存储库中的 *Container Security Operator*。

备注：

- 镜像漏洞策略在断开连接的安装过程中无法正常工作。
- IBM Power 和 IBM Z 架构不支持 镜像漏洞策略。它依赖于 Quay Container Security Operator。container-security-operator registry 中没有 **ppc64le** 或 **s390x** 镜像。

查看以下部分以了解更多信息：

- 镜像漏洞策略 YAML 结构
- 镜像漏洞策略 YAML 表
- 镜像漏洞策略示例

2.5.4.1. 镜像漏洞策略 YAML 结构

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

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

2.5.4.2. 镜像漏洞策略 YAML 表

字段	描述
apiVersion	必需。将值设置为 **policy.open-cluster-management.io/v1**。

字段	描述
kind	必需。将值设为 **Policy** 以表示策略类型。
metadata.name	必需。用于标识策略资源的名称。
metadata.namespaces	可选。
spec.namespace	必需。策略应用到的节点集群内命名空间。输入 **include** 的参数值，这是您要将策略应用到的命名空间。**exclude** 参数指定您明确不希望将策略应用到的命名空间。**注**：在策略控制器的对象模板中指定的命名空间会覆盖对应父策略中的命名空间。
remediationAction	可选。指定您的策略的修复。参数值是 **enforce** 和 **inform**。
disabled	必需。将值设为 **true** 或 **false**。**disabled** 参数提供启用和禁用策略的功能。
spec.complianceType	必需。将值设为 **"musthave"**
spec.object-template	可选。用于列出必须接受评估或应用到受管集群的任何其他 Kubernetes 对象。

2.5.4.3. 镜像漏洞策略示例

请参阅 **policy-imagemanifestvuln.yaml**。如需更多信息，请参阅管理安全策略。

请参阅 Kubernetes 配置策略控制器，以查看配置控制器监控的其他配置策略。

2.5.5. Pod 策略

Kubernetes 配置策略控制器负责监控 Pod 策略的状态。应用 Pod 策略来为 Pod 定义容器规则。集群中必须存在 pod 才能使用此信息。

在以下部分了解更多有关 pod 策略结构的详细信息：

- Pod 策略 YAML 结构
- Pod 策略表
- Pod 策略示例

2.5.5.1. Pod 策略 YAML 结构

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

```
spec:
  complianceType:
  remediationAction:
  namespaces:
    exclude:
    include:
  object-templates:
    - complianceType:
      objectDefinition:
        apiVersion:
        kind: Pod # pod must exist
        metadata:
          name:
        spec:
          containers:
          - image:
            name:
            ports:
            - containerPort:
    ...
```

2.5.5.2. Pod 策略表

字段	描述
apiVersion	必需。将值设置为 **policy.open-cluster-management.io/v1**。
kind	必需。将值设为 **Policy** 以表示策略类型。
metadata.name	必需。用于标识策略资源的名称。
metadata.namespaces	可选。
spec.namespace	必需。策略应用到的节点集群内命名空间。输入 **include** 的参数值，这是您要将策略应用到的命名空间。**exclude** 参数指定您明确不希望将策略应用到的命名空间。**注**：在策略控制器的对象模板中指定的命名空间会覆盖对应父策略中的命名空间。
remediationAction	可选。指定您的策略的修复。参数值是 **enforce** 和 **inform**。
disabled	必需。将值设为 **true** 或 **false**。**disabled** 参数提供启用和禁用策略的功能。
spec.complianceType	必需。将值设为 **"musthave"**
spec.object-template	可选。用于列出必须接受评估或应用到受管集群的任何其他 Kubernetes 对象。

2.5.5.3. Pod 策略示例

请参阅 **policy-pod.yaml** 查看策略示例。

请参阅 Kubernetes 配置策略控制器，以查看配置控制器监控的其他配置策略。请参阅管理配置策略以管理其他策略。

2.5.6. Pod 安全策略

Kubernetes 配置策略控制器负责监控 Pod 安全策略的状态。应用 Pod 安全策略来保护 Pod 和容器。如需更多信息，请参阅 Kubernetes 文档中的 *Pod 安全策略*。

在以下部分了解更多有关 Pod 安全策略结构的详细信息：

- Pod 安全策略 YAML 结构
- Pod 安全策略表
- Pod 安全策略示例

2.5.6.1. Pod 安全策略 YAML 结构

```
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  name: policy-podsecuritypolicy
  namespace:
spec:
  complianceType:
  remediationAction:
  namespaces:
    exclude:
    include:
  object-templates:
    - complianceType:
      objectDefinition:
        apiVersion:
        kind: PodSecurityPolicy # no privileged pods
        metadata:
          name:
          annotations:
        spec:
          privileged:
          allowPrivilegeEscalation:
          allowedCapabilities:
          volumes:
          hostNetwork:
          hostPorts:
          hostIPC:
          hostPID:
          runAsUser:
            rule:
          seLinux:
            rule:
          supplementalGroups:
            rule:
```

```
      fsGroup:
        rule:
...
```

2.5.6.2. Pod 安全策略表

字段	描述
apiVersion	必需。将值设置为 **policy.open-cluster-management.io/v1**。
kind	必需。将值设为 **Policy** 以表示策略类型。
metadata.name	必需。用于标识策略资源的名称。
metadata.namespaces	可选。
spec.namespace	必需。策略应用到的节点集群内命名空间。输入 **include** 的参数值，这是您要将策略应用到的命名空间。**exclude** 参数指定您明确不希望将策略应用到的命名空间。**注**：在策略控制器的对象模板中指定的命名空间会覆盖对应父策略中的命名空间。
remediationAction	可选。指定您的策略的修复。参数值是 **enforce** 和 **inform**。
disabled	必需。将值设为 **true** 或 **false**。**disabled** 参数提供启用和禁用策略的功能。
spec.complianceType	必需。将值设为 **"musthave"**
spec.object-template	可选。用于列出必须接受评估或应用到受管集群的任何其他 Kubernetes 对象。

2.5.6.3. Pod 安全策略示例

请参阅 **policy-psp.yaml** 查看示例策略。如需更多信息，请参阅管理配置策略。

请参阅 Kubernetes 配置策略控制器以查看控制器监控的其他配置策略。

2.5.7. 角色策略

Kubernetes 配置策略控制器负责监控角色策略的状态。在 **object-template** 中定义角色来为集群中的特定角色设置规则和权限。

在以下部分了解更多有关角色策略结构的详细信息：

- 角色策略 YAML 结构
- 角色策略表

- 角色策略示例

2.5.7.1. 角色策略 YAML 结构

```
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:
            include: ["default"]
          object-templates:
            - complianceType: mustonlyhave # role definition should exact match
              objectDefinition:
                apiVersion: rbac.authorization.k8s.io/v1
                kind: Role
                metadata:
                  name: sample-role
                rules:
                  - apiGroups: ["extensions", "apps"]
                    resources: ["deployments"]
                    verbs: ["get", "list", "watch", "delete","patch"]
---
apiVersion: policy.open-cluster-management.io/v1
kind: PlacementBinding
metadata:
  name: binding-policy-role
  namespace:
placementRef:
  name: placement-policy-role
  kind: PlacementRule
  apiGroup: apps.open-cluster-management.io
subjects:
- name: policy-role
  kind: Policy
  apiGroup: policy.open-cluster-management.io
---
apiVersion: apps.open-cluster-management.io/v1
```

```
kind: PlacementRule
metadata:
  name: placement-policy-role
  namespace:
spec:
  clusterConditions:
    - type: ManagedClusterConditionAvailable
      status: "True"
  clusterSelector:
    matchExpressions:
      []

        ...
```

2.5.7.2. 角色策略表

字段	描述
apiVersion	必需。将值设置为 **policy.open-cluster-management.io/v1**。
kind	必需。将值设为 **Policy** 以表示策略类型。
metadata.name	必需。用于标识策略资源的名称。
metadata.namespaces	可选。
spec.namespace	必需。策略应用到的节点集群内命名空间。输入 **include** 的参数值，这是您要将策略应用到的命名空间。**exclude** 参数指定您明确不希望将策略应用到的命名空间。**注**：在策略控制器的对象模板中指定的命名空间会覆盖对应父策略中的命名空间。
remediationAction	可选。指定您的策略的修复。参数值是 **enforce** 和 **inform**。
disabled	必需。将值设为 **true** 或 **false**。**disabled** 参数提供启用和禁用策略的功能。
spec.complianceType	必需。将值设为 **"musthave"**
spec.object-template	可选。用于列出必须接受评估或应用到受管集群的任何其他 Kubernetes 对象。

2.5.7.3. 角色策略示例

应用角色策略来为集群中的特定角色设置规则和权限。如需有关角色的更多信息，请参阅基于角色的访问控制。查看角色策略示例，请参阅 **policy-role.yaml**。

要了解如何管理角色策略，请参阅管理配置策略以了解更多信息。请参阅 Kubernetes 配置策略控制器，以查看监控控制器的其他配置策略。

2.5.8. 角色绑定策略

Kubernetes 配置策略控制器负责监控角色绑定策略的状态。应用角色绑定策略，将策略绑定到受管集群中的命名空间。

在以下部分了解更多有关命名空间策略结构的详细信息：

- 角色绑定策略 YAML 结构
- 角色绑定策略表
- 角色绑定策略示例

2.5.8.1. 角色绑定策略 YAML 结构

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

2.5.8.2. 角色绑定策略表

字段	描述
apiVersion	必需。将值设置为 **policy.open-cluster-management.io/v1**。
kind	必需。将值设为 **Policy** 以表示策略类型。
metadata.name	必需。用于标识策略资源的名称。

字段	描述
metadata.namespaces	必需。创建了策略的受管集群内命名空间。
spec	必需。有关如何标识和修复合规违反情况的规格。
metadata.name	必需。用于标识策略资源的名称。
metadata.namespaces	可选。
spec.complianceType	必需。将值设为 **"musthave"**
spec.namespace	必需。要将策略应用到的受管集群命名空间。输入 **include** 的参数值，这是您要将策略应用到的命名空间。**exclude** 参数指定您明确不希望将策略应用到的命名空间。**注**：在策略控制器的对象模板中指定的命名空间会覆盖对应父策略中的命名空间。
spec.remediationAction	必需。指定您的策略的修复。参数值是 **enforce** 和 **inform**。
spec.object-template	必需。用于列出必须接受评估或应用到受管集群的任何其他 Kubernetes 对象。

2.5.8.3. 角色绑定策略示例

请参阅 **policy-rolebinding.yaml** 查看策略示例。有关管理其他策略的更多信息，请参阅管理配置策略。

请参阅 Kubernetes 配置策略控制器以了解其他配置策略。

2.5.9. 安全性上下文约束策略

Kubernetes 配置策略控制器负责监控安全性上下文约束 (SCC) 策略的状态。应用安全性上下文约束 (SCC) 策略，通过在策略中定义条件来控制 Pod 的权限。

在以下部分了解更多有关 SCC 策略的详细信息：

- SCC 策略 YAML 结构
- SCC 策略表
- SCC 策略示例

2.5.9.1. SCC 策略 YAML 结构

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

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

2.5.9.2. SCC 策略表

字段	描述
apiVersion	必需。将值设置为 **policy.open-cluster-management.io/v1**。
kind	必需。将值设为 **Policy** 以表示策略类型。
metadata.name	必需。用于标识策略资源的名称。
metadata.namespace	必需。创建了策略的受管集群内命名空间。

字段	描述
spec.complianceType	必需。将值设为 **"musthave"**
spec.remediationAction	必需。指定您的策略的修复。参数值是 **enforce** 和 **inform**。**重要**：有些策略可能不支持 enforce 功能。
spec.namespace	必需。要将策略应用到的受管集群命名空间。输入 **include** 的参数值，这是您要将策略应用到的命名空间。**exclude** 参数指定您明确不希望将策略应用到的命名空间。**注**：在策略控制器的对象模板中指定的命名空间会覆盖对应父策略中的命名空间。
spec.object-template	必需。用于列出必须接受评估或应用到受管集群的任何其他 Kubernetes 对象。

有关 SCC 策略内容的解释，请参阅 OpenShift Container Platform 文档中的管理安全性上下文约束。

2.5.9.3. SCC 策略示例

应用安全性上下文约束 (SCC) 策略，通过在策略中定义条件来控制 Pod 的权限。如需更多信息，请参阅管理安全性上下文约束（SCC）。

请参阅 **policy-scc.yaml** 查看策略示例。有关管理其他策略的更多信息，请参阅管理配置策略。

请参阅 Kubernetes 配置策略控制器以了解其他配置策略。

2.5.10. ETCD 加密策略

应用 **etcd-encryption** 策略，在 ETCD 数据存储中检测或启用敏感数据的加密。Kubernetes 配置策略控制器负责监控 **etcd-encryption** 策略的状态。如需更多信息，请参阅 OpenShift Container Platform 文档中的加密 etcd 数据。**注** ：ETCD 加密策略只支持 Red Hat OpenShift Container Platform 4 及更新的版本。

在以下部分了解更多有关 **etcd-encryption** 策略结构的详细信息：

- ETCD 加密策略 YAML 结构
- ETCD 加密策略表
- etcd 加密策略示例

2.5.10.1. ETCD 加密策略 YAML 结构

etcd-encryption 策略可能类似以下 YAML 文件：

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

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

2.5.10.2. ETCD 加密策略表

表 2.5. 参数表

字段	描述
apiVersion	必需。将值设置为 **policy.open-cluster-management.io/v1**。
kind	必需。将值设为 **Policy** 以表示策略类型，例如 **ConfigurationPolicy**。
metadata.name	必需。用于标识策略资源的名称。
metadata.namespaces	可选。
spec.namespace	必需。策略应用到的节点集群内命名空间。输入 **include** 的参数值，这是您要将策略应用到的命名空间。**exclude** 参数指定您明确不希望将策略应用到的命名空间。**注**：在策略控制器的对象模板中指定的命名空间会覆盖对应父策略中的命名空间。
remediationAction	可选。指定您的策略的修复。参数值是 **enforce** 和 **inform**。**重要**：有些策略可能不支持 enforce 功能。
disabled	必需。将值设为 **true** 或 **false**。**disabled** 参数提供启用和禁用策略的功能。
spec.complianceType	必需。将值设为 **"musthave"**
spec.object-template	可选。用于列出必须接受评估或应用到受管集群的任何其他 Kubernetes 对象。请参阅 OpenShift Container Platform 文档中的加密 etcd 数据。

2.5.10.3. etcd 加密策略示例

如需策略示例，请参阅 **policy-etcdencryption.yaml**。如需更多信息，请参阅管理安全策略。

请参阅 Kubernetes 配置策略控制器以查看控制器监控的其他配置策略。

2.5.11. Compliance operator 策略

Compliance Operator 是一个运行 OpenSCAP 的 operator，可让您使 Red Hat OpenShift Container Platform 集群与您需要的安全基准兼容。您可以使用合规 operator 策略在受管集群上安装合规 Operator。

compliance operator 策略在 Red Hat Advanced Cluster Management 中作为 Kubernetes 配置策略创建。OpenShift Container Platform 4.6 和 4.7 支持合规性 Operator 策略。如需更多信息，请参阅 OpenShift Container Platform 文档中的 了解 Compliance Operator 部分以了解更多详细信息。

注： Compliance operator 策略依赖于 OpenShift Container Platform Compliance Operator，它不受 IBM Power 或 IBM Z 架构的支持。如需有关 Compliance Operator 的更多信息，请参阅 OpenShift Container Platform 文档中的了解 Compliance Operator。

2.5.11.1. Compliance operator 资源

创建合规 Operator 策略时，会创建以下资源：

- Operator 安装的合规性 operator 命名空间（**openshift- compliance**）：

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

- 用于指定目标命名空间的 operator 组（**compliance-operator**）：

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

```
    metadata:
      name: compliance-operator
      namespace: openshift-compliance
    spec:
      targetNamespaces:
        - openshift-compliance
```

- 用于引用名称和频道的订阅（**comp-operator-subscription**）。订阅会拉取配置集作为一个容器，它支持：

```
apiVersion: policy.open-cluster-management.io/v1
kind: ConfigurationPolicy
metadata:
  name: comp-operator-subscription
spec:
  remediationAction: inform  # will be overridden by remediationAction in parent policy
  severity: high
  object-templates:
    - complianceType: musthave
      objectDefinition:
        apiVersion: operators.coreos.com/v1alpha1
        kind: Subscription
        metadata:
          name: compliance-operator
          namespace: openshift-compliance
        spec:
          channel: "4.7"
          installPlanApproval: Automatic
          name: compliance-operator
          source: redhat-operators
          sourceNamespace: openshift-marketplace
```

安装 compliance operator 策略后，会创建以下 pod：**compliance-operator**、**ocp4** 和 **rhcos4**。请参阅 **policy-compliance-operator-install.yaml** 示例。

安装 compliance Operator 后，您还可以创建并应用 E8 扫描策略和 OpenShift CIS 扫描策略。如需更多信息，请参阅 E8 扫描策略和 OpenShift CIS 扫描策略。

要了解如何管理合规 Operator 策略，请参阅管理安全策略以了解更多详细信息。有关配置策略的更多主题，请参阅 Kubernetes 配置策略控制器。

2.5.12. E8 扫描策略

一个 Essential 8（E8）扫描策略会部署一个扫描，检查 master 和 worker 节点是否满足 E8 安全配置集。您必须安装 Compliance Operator 以应用 E8 扫描策略。

E8 扫描策略在 Red Hat Advanced Cluster Management 中作为 Kubernetes 配置策略创建。OpenShift Container Platform 4.7 和 4.6 支持 E8 扫描策略。如需更多信息，请参阅 OpenShift Container Platform 文档中的 *了解 Compliance Operator* 部分以了解更多详细信息。

2.5.12.1. E8 扫描策略资源

当您创建 E8 扫描策略时，会创建以下资源：

- **ScanSettingBinding** 资源 (**e8**) 用于识别要扫描的配置集：

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

- 一个 **ComplianceSuite** 资源 (**compliance-suite-e8**)，用于通过检查 **status** 字段来验证扫描是否已完成：

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

- 一个 **ComplianceCheckResult** 资源 (**compliance-suite-e8-results**)，它通过检查 **ComplianceCheckResult** 自定义资源 (CR) 来报告扫描套件的结果：

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

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

注： 支持自动补救。将补救操作设置为 **enforce** 以创建 **ScanSettingBinding** 资源。

请参阅 **policy-compliance-operator-e8-scan.yaml** 示例。如需更多信息，请参阅管理安全策略。**注：** 删除 E8 策略后，它会从目标集群或集群中删除。

2.5.13. OpenShift CIS 扫描策略

OpenShift CIS 扫描策略会部署一个扫描来检查 master 和 worker 节点是否与 OpenShift CIS 安全基准相符。您必须安装 Compliance operator 以应用 OpenShift CIS 策略。

OpenShift CIS 扫描策略在 Red Hat Advanced Cluster Management 中作为 Kubernetes 配置策略创建。OpenShift Container Platform 4.9、4.7 和 4.6 支持 OpenShift CIS 扫描策略。如需更多信息，请参阅 OpenShift Container Platform 文档中的 了解 Compliance Operator 部分以了解更多详细信息。

2.5.13.1. OpenShift CIS 资源

创建 OpenShift CIS 扫描策略时，会创建以下资源：

- 用于识别要扫描的配置集的 **ScanSettingBinding** 资源 (**cis**)：

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

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

- 一个 **ComplianceSuite** 资源 (**compliance-suite-cis**)，用于通过检查 **status** 字段来验证扫描是否已完成：

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

- 一个 **ComplianceCheckResult** 资源 (**compliance-suite-cis-results**)，它通过检查 **ComplianceCheckResult** 自定义资源 (CR) 来报告扫描套件的结果：

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

请参阅 **policy-compliance-operator-cis-scan.yaml** 文件示例。有关创建策略的更多信息，请参阅管理安全策略。

2.5.14. Kyverno 添加网络策略

Kyverno 添加网络策略配置一个名为 **default-deny** 的新 **NetworkPolicy** 资源，它拒绝您在创建新命名空间时的所有流量。您必须安装 Kyverno 控制器才能使用 Kyverno 策略。有关安装策略，请参阅 **policy-install-kyverno.yaml**。

在以下部分了解更多有关 **policy-kyverno-add-network-policy** 策略结构的详细信息：

- Kyverno 添加网络策略 YAML 结构
- Kyverno 添加网络策略示例

2.5.14.1. Kyverno 添加网络策略 YAML 结构

您的 **policy-kyverno-add-network-policy** 策略可能类似以下 YAML 文件：

```
apiVersion: kyverno.io/v1
kind: ClusterPolicy
metadata:
  name: add-networkpolicy
  annotations:
    policies.kyverno.io/title: Add Network Policy
    policies.kyverno.io/category: Multi-Tenancy
    policies.kyverno.io/subject: NetworkPolicy
    policies.kyverno.io/description: >-
      By default, Kubernetes allows communications across all Pods within a cluster.
      The NetworkPolicy resource and a CNI plug-in that supports NetworkPolicy must be used to
restrict
      communications. A default NetworkPolicy should be configured for each Namespace to
      default deny all ingress and egress traffic to the Pods in the Namespace. Application
      teams can then configure additional NetworkPolicy resources to allow desired traffic
      to application Pods from select sources. This policy will create a new NetworkPolicy resource
      named `default-deny` which will deny all traffic anytime a new Namespace is created.
spec:
  rules:
  - name: default-deny
    match:
      resources:
        kinds:
        - Namespace
    generate:
      apiVersion: networking.k8s.io/v1
      kind: NetworkPolicy
      name: default-deny
      namespace: "{{request.object.metadata.name}}"
      synchronize: true
      data:
        spec:
          # select all pods in the namespace
          podSelector: {}
          # deny all traffic
          policyTypes:
          - Ingress
          - Egress
```

2.5.14.2. Kyverno 添加网络策略示例

如需策略示例，请参阅 **policy-kyverno-add-network-policy.yaml**。请参阅策略概述文档 和 Kubernetes 配置策略控制器，以查看策略和配置策略字段的更多详情。

2.5.15. Kyverno 添加配额策略

Kyverno 添加配额策略会在创建新命名空间时配置新的 **ResourceQuota** 和 **LimitRange** 资源。您必须安装 Kyverno 控制器才能使用 Kyverno 策略。有关安装策略，请参阅 **policy-install-kyverno.yaml**。

在以下部分了解更多有关 **policy-kyverno-add-quota** 策略结构的详细信息：

- Kyverno 添加配额策略 YAML 结构
- Kyverno 添加配额策略示例

2.5.15.1. Kyverno 添加配额策略 YAML 结构

您的 **policy-kyverno-add-quota** 策略可能类似以下 YAML 文件：

```
apiVersion: kyverno.io/v1
kind: ClusterPolicy
metadata:
  name: add-ns-quota
  annotations:
    policies.kyverno.io/title: Add Quota
    policies.kyverno.io/category: Multi-Tenancy
    policies.kyverno.io/subject: ResourceQuota, LimitRange
    policies.kyverno.io/description: >-
      To better control the number of resources that can be created in a given
      Namespace and provide default resource consumption limits for Pods,
      ResourceQuota and LimitRange resources are recommended.
      This policy will generate ResourceQuota and LimitRange resources when
      a new Namespace is created.
spec:
  rules:
  - name: generate-resourcequota
    match:
      resources:
        kinds:
        - Namespace
    generate:
      apiVersion: v1
      kind: ResourceQuota
      name: default-resourcequota
      synchronize: true
      namespace: "{{request.object.metadata.name}}"
      data:
        spec:
          hard:
            requests.cpu: '4'
            requests.memory: '16Gi'
            limits.cpu: '4'
            limits.memory: '16Gi'
  - name: generate-limitrange
    match:
```

```
    resources:
      kinds:
      - Namespace
  generate:
    apiVersion: v1
    kind: LimitRange
    name: default-limitrange
    synchronize: true
    namespace: "{{request.object.metadata.name}}"
    data:
      spec:
        limits:
        - default:
            cpu: 500m
            memory: 1Gi
          defaultRequest:
            cpu: 200m
            memory: 256Mi
          type: Container
```

2.5.15.2. Kyverno 添加配额策略示例

如需策略示例，请参阅 **policy-kyverno-add-quota.yaml**。请参阅策略概述文档 和 Kubernetes 配置策略控制器，以查看策略和配置策略字段的更多详情。

2.5.16. Kyverno sync secret 策略

Kyverno sync secret 策略会将名为 **regcred** 的 secret（存在于 **default** 命名空间中） 复制到您在检测到更改时创建和更新 secret 的新命名空间中。您必须安装 Kyverno 控制器才能使用 Kyverno 策略。有关安装策略，请参阅 **policy-install-kyverno.yaml**。

在以下部分了解更多有关 **policy-kyverno-sync-secrets** 策略结构的详细信息：

- Kyverno 同步 secret 策略 YAML 结构
- Kyverno 同步 secret 策略示例

2.5.16.1. Kyverno 同步 secret 策略 YAML 结构

您的 **policy-kyverno-sync-secrets** 策略可能类似以下 YAML 文件：

```
apiVersion: kyverno.io/v1
kind: ClusterPolicy
metadata:
  name: sync-secrets
  annotations:
    policies.kyverno.io/title: Sync Secrets
    policies.kyverno.io/category: Sample
    policies.kyverno.io/subject: Secret
    policies.kyverno.io/description: >-
      Secrets like registry credentials often need to exist in multiple
      Namespaces so Pods there have access. Manually duplicating those Secrets
      is time consuming and error prone. This policy will copy a
      Secret called `regcred` which exists in the `default` Namespace to
      new Namespaces when they are created. It will also push updates to
```

```
      the copied Secrets should the source Secret be changed.
spec:
  rules:
  - name: sync-image-pull-secret
    match:
      resources:
        kinds:
        - Namespace
    generate:
      apiVersion: v1
      kind: Secret
      name: regcred
      namespace: "{{request.object.metadata.name}}"
      synchronize: true
      clone:
        namespace: default
        name: regcred
```

2.5.16.2. Kyverno 同步 secret 策略策略示例

如需策略示例，请参阅 **policy-kyverno-sync-secrets.yaml**。请参阅策略概述文档 和 Kubernetes 配置策略控制器，以查看策略和配置策略字段的更多详情。

2.6. 管理安全策略

使用*监管*仪表板创建、查看和管理安全策略和策略违反情况。您可以通过 CLI 和控制台为您的策略创建 YAML 文件。

2.6.1. 监管页面

Governance 页面中显示以下标签页：

- **概述**
  在 *Overview* 选项卡中查看以下概述卡： *Policy set violations*,*Policy violations*,*Clusters*,*Categories*, *Controls*, 和 *Standards*。

- **策略集合**
  创建和管理 hub 集群策略集。

- **策略（policy）**
  创建和管理安全策略。策略列表列出了以下策略详情：*Name*, *Namespace*, *Status*, *Remediation*, *Policy set*, *Cluster violations*, *Source*, *Automation* 和 *Created*。

  您可以编辑、启用或禁用，将补救设置为 inform 或 enforce，或通过选择 **Actions** 图标来删除策略。您可以通过选择要展开行的下拉箭头来查看特定策略的类别和标准。

  选择多个策略并点击 **Actions** 按钮来完成批量操作。您还可以点 **Filter** 按钮自定义您的策略表。

如果为特定策略配置了自动化，您可以选择自动化来查看更多详情。查看以下自动化调度频率选项的描述：

- *Manual run* 模式：手动将此自动化设置为运行一次。在自动化运行后，它将设置为 **disabled**。为自动化选择 *Manual run* 复选框。

- *Once mode*: 违反策略时，自动化将运行一次。在自动化运行后，它将设置为 **disabled**。在将自

动化设置为**禁用**后，您必须继续手动运行自动化。当使用 *once mode* 时，**target_clusters** 的额外变量会自动提供违反策略的集群列表。Ansible Tower 作业模板必须为 **EXTRA VARIABLES** 部分启用 **PROMPT ON LAUNCH**。

- *Disable*：当调度的自动化被设置为**禁用**时，在更新设置前不会运行自动化。

当您在表列表中选择一个策略时，控制台中会显示以下信息标签页：

- *Details* ：选择 *Details* 选项卡来查看策略详情和放置详情。在 *Placement* 表中，*Compliance* 列提供了查看所显示集群的合规性的链接。

- *Results* ：选择 *Results* 选项卡来查看与放置关联的所有集群的表列表。
从 *Message* 列中，点 **View details** 链接来查看模板详情、模板 YAML 和相关资源。您还可以查看相关的资源。点 **View history** 链接查看违反消息以及最后一次报告的时间。

查看以下主题以了解更多有关创建和更新您的安全策略的信息。

- 管理安全策略
- 管理配置策略
- 管理 gatekeeper 策略
- 为监管配置 Ansible Tower

更多主题，请参阅监管。

2.6.2. 为监管配置 Ansible Tower

Red Hat Advanced Cluster Management for Kubernetes 监管可与 Ansible Tower 自动化集成，以创建策略违反自动化。您可以从 Red Hat Advanced Cluster Management 控制台配置自动化。

- 先决条件
- 从控制台创建策略违反自动化
- 通过 CLI 创建策略违反自动化

2.6.2.1. 先决条件

- Red Hat OpenShift Container Platform 4.5 或更高版本
- 已安装 Ansible Tower 版本 3.7.3 或更高版本。安装最新版本的 Ansible Tower 是最佳实践方案。如需了解更多详细信息，请参阅 Red Hat Ansible Tower 文档 。
- 将 Ansible Automation Platform Resource Operator 安装到 hub 集群，将 Ansible 作业连接到管理框架。为了获得最佳结果，在使用 AnsibleJob 启动 Ansible Tower 作业时，Ansible Tower 作业模板在运行时应该是等价的。如果没有 Ansible Automation Platform Resource Operator，您可以在 Red Hat OpenShift Container Platform *OperatorHub* 页面中找到它。

有关安装和配置 Ansible Tower 自动化的更多信息，请参阅设置 Ansible 任务。

2.6.2.2. 从控制台创建策略违反自动化

登录到 Red Hat Advanced Cluster Management hub 集群后，从导航菜单中选择 **Governance**，然后点 *Policies* 选项卡来查看策略表。

单击 *Automation* 列中的 **Configure**，为特定策略配置自动化。当策略自动化面板出现时，您可以创建自动化。在 *Ansible credential* 部分中，单击下拉菜单来选择 Ansible 凭据。如果需要添加凭证，请参阅 管理凭证概述。

注：此凭证复制到与策略相同的命名空间中。该凭据供创建用于启动自动化的 **AnsibleJob** 资源使用。控制台的 *Credentials* 部分中的 Ansible 凭据更改会被自动更新。

选择了凭据后，单击 Ansible 作业下拉列表来选择作业模板。在 *Extra variables* 部分，添加来自 **PolicyAutomation** 的 **extra_vars** 部分中的参数值。选择自动化的频率。您可以选择 *Run once mode*，或*Disable automation*。

- *Manual run*：手动将此自动化设置为运行一次。在自动化运行后，它将设置为 **disabled**。**注意**：您只能在禁用调度频率时选择 *Manual run* 模式。
- *Run once mode*：违反策略时，自动化将运行一次。在自动化运行后，它将设置为 **disabled**。在将自动化设置为**禁用**后，您必须继续手动运行自动化。当使用 *once mode* 时，**target_clusters** 的额外变量会自动提供违反策略的集群列表。Ansible Tower 作业模板必须为 **EXTRA VARIABLES** 部分启用了 **PROMPT ON LAUNCH**。
- *Disable automation*：当调度的自动化被设置为**禁用**时，在更新设置前不会运行自动化。

选择 **Submit** 保存您的策略违反自动化。当您从 Ansible 作业详情侧面板选择 *View Job* 链接时，链接会将您定向到 *Search* 页面上的作业模板。成功创建自动化后，它会显示在 *Automation* 列中。

注：当您删除具有关联策略自动化的策略时，策略自动化会在清理过程中自动删除。

您的策略违反自动化是从控制台创建的。

2.6.2.3. 通过 CLI 创建策略违反自动化

完成以下步骤，通过 CLI 配置策略违反自动化：

1. 在终端中，使用 **oc login** 命令登录到 Red Hat Advanced Cluster Management hub 集群。
2. 查找或创建您要向其添加自动化的策略。请注意策略名称和命名空间。
3. 使用以下示例创建 **PolicyAutomation** 资源，作为指南：

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

4. 上例中的 Ansible 作业模板名称是 **Policy Compliance Template**。更改该值，使其与您的任务模板名称匹配。
5. 在 **extra_vars** 部分中，添加您需要传递给 Ansible 作业模板的任何参数。

6. 将模式设置为 **once** 或 **disabled**。**once** 模式只运行作业一次，然后模式会被设置为 **disabled**。
   - *once mode* ：违反策略时，自动化将运行一次。在自动化运行后，它将设置为 **disabled**。在将自动化设置为**禁用**后，您必须继续手动运行自动化。当使用 *once mode* 时，**target_clusters** 的额外变量会自动提供违反策略的集群列表。Ansible Tower 作业模板必须为 **EXTRA VARIABLES** 部分启用了 **PROMPT ON LAUNCH**。
   - *Disable automation* ：当调度的自动化被设置为**禁用**时，在更新设置前不会运行自动化。
7. 将 **policyRef** 设置为您的策略的名称。
8. 在与包含 Ansible Tower 凭据的 **PolicyAutomation** 资源相同的命名空间中创建一个 secret。在上例中，secret 名称为 **ansible-tower**。使用应用程序生命周期中的示例来查看如何创建 secret。
9. 创建 **PolicyAutomation** 资源。
   备注:
   - 可以通过在 **PolicyAutomation** 资源中添加以下注解来立即运行策略自动化：

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

   - 当策略为 **once** 模式时，当策略不合规时自动化将运行。添加名为 **target_clusters** 的 **extra_vars** 变量，值是每个受管集群名称的数组，其中的策略不合规。

2.6.3. 使用 GitOps 部署策略

您可以使用监管框架在一组受管集群中部署一组策略。您可以通过添加到开源社区的 **policy-collection**，贡献并使用软件仓库中的策略。如需更多信息，请参阅贡献自定义策略。来自开源社区的、位于 **stable** 和 **community** 目录中的策略会进一步根据 NIST Special Publication 800-53 进行组织。

继续阅读以了解使用 GitOps 通过 Git 存储库自动化和跟踪策略更新和创建的最佳做法。

先决条件：开始之前，请务必 fork **policy-collection** 存储库。

- 自定义本地存储库
- 提交到您的本地存储库
- 在集群中部署策略
- 从控制台验证 GitOps 策略部署
- 通过 CLI 验证 GitOps 策略部署

2.6.3.1. 自定义本地存储库

通过将 **stable** 和 **community** 策略整合到单个文件夹中，自定义您的本地存储库。删除您不想使用的策略。完成以下步骤以自定义您的本地存储库：

1. 在存储库中创建一个新目录，以存放您要部署的策略。确保您位于 GitOps 的主要默认分支的本地 **policy-collection** 存储库中。运行以下命令：

   ```
   mkdir my-policies
   ```

2. 将所有 **stable** 和 **community** 策略复制到您的 **my-policies** 目录中。首先，如果 **stable** 文件夹包含 **community** 中可用内容的副本，则首先从社区策略开始。运行以下命令：

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

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

现在，您在一个父目录结构中已有所有策略，您可以在 fork 中编辑策略。

提示：

- 最佳做法是删除您不打算使用的策略。
- 从以下列表中了解策略和策略定义：
  - 用途：了解策略的作用。
  - 补救操作：该策略是否仅告知您合规，还是强制执行策略并进行更改？请参阅 **spec.remediationAction** 参数。如果强制进行更改，请确保您了解功能预期。记得检查哪些策略支持执行。如需更多信息，请参阅 *Validate* 部分。
    注：策略的 **spec.remediationAction** 设置覆盖单个 **spec.policy-templates** 中设置的任何补救操作。
  - placement:策略部署到哪些群集？默认情况下，大多数策略都以带有 **environment: dev** 标签的集群为目标。有些策略可能针对 OpenShift Container Platform 集群或其他标签。您可以更新或添加附加标签来包括其他集群。如果没有特定值，策略会应用到所有集群。您还可以创建策略的多个副本并为每个副本自定义，如果您想要使用一组集群配置的一种策略，并为另一组集群配置另一种方式。

2.6.3.2. 提交到您的本地存储库

在您对目录所做的更改满意后，提交您的更改并将其推送到 Git，以便集群可以访问它们。

注：本示例用于显示如何将策略与 GitOps 搭配使用的基础知识，因此您可能具有不同的工作流来对分支进行更改。

完成以下步骤：

1. 在终端中运行 **git status**，以查看您之前创建的目录中的最近更改。将您的新目录添加到要使用以下命令提交的更改列表中：

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

2. 提交更改并自定义您的消息。运行以下命令：

```
git commit -m “Policies to deploy to the hub cluster”
```

3. 将更改推送到用于 GitOps 的已分叉存储库的分支。运行以下命令：

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

您的更改已提交。

2.6.3.3. 在集群中部署策略

在推送更改后，您可以将策略部署到 Red Hat Advanced Cluster Management for Kubernetes 安装中。在部署后，您的 hub 集群连接到 Git 存储库。对 Git 存储库所选分支的任何其他更改都会反映在集群中。

注: 默认情况下，通过 GitOps 部署的策略使用 **merge** reconcile 选项。如果要使用 **replace** 协调选项，将 **apps.open-cluster-management.io/reconcile-option: replace** 注解添加到 **Subscription** 资源。如需了解更多详细信息，请参阅应用程序生命周期。

deploy.sh 脚本在 hub 集群中创建 **Channel** 和 **Subscription** 资源。频道连接到 Git 仓库，订阅指定要通过频道传递给集群的数据。因此，在您的 hub 上会创建指定子目录中定义的所有策略。在订阅创建策略后，Red Hat Advanced Cluster Management 会根据定义的放置规则分析策略，并在与策略应用到的每个受管集群关联的命名空间中创建额外的策略资源。

该策略随后从 hub 集群上对应的受管集群命名空间中复制到受管集群。因此，Git 仓库中的策略推送到具有与策略放置规则中定义的 **clusterSelector** 匹配的所有受管集群。

完成以下步骤：

1. 在 **policy-collection** 文件夹中运行以下命令来更改目录：

   ```
   cd deploy
   ```

2. 请确定将命令行界面（CLI）配置为使用以下命令在正确的集群中创建资源：

   ```
   oc cluster-info
   ```

   命令的输出显示集群的 API 服务器详情，其中安装了 Red Hat Advanced Cluster Management。如果没有显示正确的 URL，请将 CLI 配置为指向正确的集群。如需更多信息，请参阅使用 OpenShift CLI。

3. 创建一个命名空间，创建您的策略以控制访问和组织策略。运行以下命令：

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

4. 运行以下命令将策略部署到集群中：

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

   将 **your-repository** 替换为您的 Git 用户名或存储库名称。

   注：deploy.sh 脚本的完整参数列表使用以下语法：

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

   查看每个参数的以下解释：

   - URL：从主 **policy-collection** 存储库派生的存储库的 URL。默认 URL 为 **https://github.com/stolostron/policy-collection.git**。
   - 分支：要指向的 Git 存储库的分支。默认分支为 **main**。
   - 子目录路径：为包含要使用的策略而创建的子目录路径。在前面的示例中，我们使用了 **my-policies** 子目录，但您也可以指定您想要从哪个文件夹开始。例如，您可以使用 **my-policies/AC-Access-Control**。默认文件夹为 **stable**。

- Namespace：在 hub 集群中创建资源和策略的命名空间。这些说明使用 **policy-namespace** 命名空间。默认命名空间是 **policies**。
- 名称前缀： **Channel** 和 **Subscription** 资源的前缀。默认为 **demo-stable-policies**。

运行 **deploy.sh** 脚本后，有权访问存储库的任何用户都可以提交更改到分支，该分支会将更改推送到集群上执行策略。

注：要使用订阅部署策略，请完成以下步骤：

1. 将 **open-cluster-management:subscription-admin** ClusterRole 绑定到创建订阅的用户。
2. 如果您在订阅中使用允许列表，请包含以下 API 条目：

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

2.6.3.4. 从控制台验证 GitOps 策略部署

从控制台验证您的更改是否已应用于您的策略。您还可以从控制台对策略进行更多更改，但当 **订阅** 与 Git 存储库协调时，会恢复更改。完成以下步骤：

1. 登录您的 Red Hat Advanced Cluster Management 集群。
2. 在导航菜单中选择 **Governance**。
3. 查找您在表中部署的策略。使用 GitOps 部署的策略在 *Source* 列中具有 *Git* 标签。单击该标签，以查看 Git 存储库的详细信息。

2.6.3.4.1. 通过 CLI 验证 GitOps 策略部署

完成以下步骤：

1. 检查以下策略详情：
   - 为什么一个特定的策论在所分发的集群中合规或不合规？
   - 策略是否应用到正确的集群？
   - 如果此策略没有分发到任何集群，为什么？
2. 识别您创建或修改的 GitOps 部署策略。GitOps 部署的策略可以通过自动应用的注解来标识。GitOps 部署的策略的注解类似以下路径：

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

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

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

GitOps 注解可用于查看哪一订阅创建了该策略。您还可以在策略中添加自己的标签，以便可以编写基于标签选择策略的运行时查询。

例如，您可以使用以下命令在策略中添加标签：

```
oc label policy <policy-name> -n <policy-namespace> <key>=<value>
```

然后，您可以使用以下命令查询带有标签的策略：

```
oc get policy -n <policy-namespace> -l <key>=<value>
```

使用 GitOps 部署了您的策略。

2.6.4. 支持配置策略中的模板

配置策略支持将 Golang 文本模板包含在对象定义中。这些模板在 hub 集群或目标受管集群的运行时使用与该集群相关的配置解决。这可让您使用动态内容定义配置策略，并通知或强制实施为目标集群自定义的 Kubernetes 资源。

- 前提条件
- 模板功能
- 在配置策略中支持 hub 集群模板
  - 模板处理
  - 重新处理的特殊注解
  - 绕过模板处理
  - hub 集群和受管集群模板的比较

2.6.4.1. 前提条件

- 模板语法必须符合 Golang 模板语言规范，并且从解析的模板生成的资源定义必须是有效的 YAML。如需更多与 Package 模板相关的信息，请参阅 Golang 文档。模板验证中的任何错误都将识别为策略违反情况。当您使用自定义模板功能时，这些值会在运行时被替换。

2.6.4.2. 模板功能

模板功能（如特定于资源和通用的 **lookup** 模板功能）可用于引用 hub 集群上的 Kubernetes 资源（使用 **{{hub ... hub}}** 分隔符）或受管集群（使用 **{{ ... }}** 分隔符）。如需了解更多详细信息，请参阅配置策略中的 hub 集群模板 支持。特定于资源的功能用于方便使用，并使资源内容更易于访问。如果您使用通用的函数 **lookup**，则最好熟悉正在查找的资源的 YAML 结构。除了这些功能外，还可使用 **base64encode**、**base64decode**、**indent**、**autoindent**、**toInt**、**toBool** 等实用程序功能。

要将模板符合 YAML 语法，必须使用引号或块字符（| 或 >）在策略资源中以字符串的形式设置模板。这会导致解析的模板值也是字符串。要覆盖此功能，请考虑使用 **toInt** 或 **toBool** 作为模板中的最终功能，以启动进一步处理，强制将值解释为整数或布尔值。

继续阅读以查看支持的一些自定义模板功能的描述和示例：

- fromSecret function
- fromConfigmap function
- fromClusterClaim function
- lookup function
- base64enc function
- base64dec function
- indent function
- autoindent 功能
- toInt function
- toBool function
- 保护功能

2.6.4.2.1. fromSecret function

fromSecret 功能返回 secret 中给定 data 键的值。查看该功能的以下语法：

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

使用此功能时，请输入 Kubernetes **Secret** 资源的命名空间、名称和数据键。在 hub 集群模板中使用函数时，您必须使用用于策略的同一命名空间。如需了解更多详细信息，请参阅配置策略中的 hub 集群模板支持。

注意 ：当将此功能与 hub 集群模板搭配使用时，输出会自动使用 保护功能 加密。

如果目标集群上不存在 Kubernetes **Secret** 资源，则会出现策略违反的情况。如果目标集群上不存在 data 键，则该值将变为空字符串。查看在目标集群上强制执行 **Secret** 资源的以下配置策略。**PASSWORD** data 键的值是引用目标集群上 secret 的模板：

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

```
      USER_NAME: YWRtaW4=
      PASSWORD: '{{ fromSecret "default" "localsecret" "PASSWORD" }}'
    kind: Secret
    metadata:
      name: demosecret
      namespace: test
    type: Opaque
  remediationAction: enforce
  severity: low
```

2.6.4.2.2. fromConfigmap function

fromConfigmap 功能返回 ConfigMap 中给定 data 键的值。查看该功能的以下语法：

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

使用此功能时，请输入 Kubernetes **ConfigMap** 资源的命名空间、名称和数据键。您必须使用 hub 集群模板中的功能用于策略的同一命名空间。如需了解更多详细信息，请参阅配置策略中的 hub 集群模板 支持。如果目标集群上不存在 Kubernetes **ConfigMap** 资源，则会出现策略违反的情况。如果目标集群上不存在 data 键，则该值将变为空字符串。查看在目标受管集群中强制执行 Kubernetes 资源的以下配置策略。**log-file** data 键的值是一个模板，它从 ConfigMap 获得 **log-file** 的值，从 **default** 命名空间获得 **log-config** ， **log-level** 被设置为 data 键 **log-level**。

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

2.6.4.2.3. fromClusterClaim function

fromClusterClaim 功能返回 **ClusterClaim** 资源中的 **Spec.Value** 的值。查看该功能的以下语法：

```
func fromClusterClaim (clusterclaimName string) (value map[string]interface{}, err Error)
```

使用此功能时，输入 Kubernetes **ClusterClaim** 资源的名称。如果 **ClusterClaim** 资源不存在，您会收到策略违反情况。查看在目标受管集群上强制执行 Kubernetes 资源的配置策略示例。**platform** 数据键的值是一个模板，它检索 **platform.open-cluster-management.io** 集群声明的值。同样，它从 **ClusterClaim** 获取**产品**和**版本**的值：

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

2.6.4.2.4. lookup function

lookup 功能将 Kubernetes 资源作为 JSON 兼容映射返回。如果请求的资源不存在，则返回空映射。如果资源不存在，并且值提供给另一个模板功能，您可能会得到以下错误： **invalid value; expected string**。

注： 使用**默认** 模板功能，因此为后续模板功能提供了正确的类型。请参阅 *开源社区功能* 部分。

查看该功能的以下语法：

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

使用功能时，输入 Kubernetes 资源的 API 版本、类型、命名空间和名称。您必须在 hub 集群模板中使用与策略相同的命名空间。如需了解更多详细信息，请参阅配置策略中的 hub 集群模板 支持。查看在目标受管集群上强制执行 Kubernetes 资源的配置策略示例。**metrics-url** 数据键的值是一个模板，它从 **default** 命名空间中获取 **v1**/**Service** Kubernetes **metrics** 资源，并设置为查询的资源中的 **Spec.ClusterIP** 的值：

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

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

2.6.4.2.5. base64enc function

base64enc 功能返回以 **base64** 编码的输入 **data string** 值。查看该功能的以下语法：

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

使用这个功能时，输入字符串值。查看以下使用 **base64enc** 功能的配置策略示例：

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

2.6.4.2.6. base64dec function

base64dec 功能返回一个以 **base64** 解码的输入的 **enc-data string** 值。查看该功能的以下语法：

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

使用这个功能时，输入字符串值。查看以下使用 **base64enc** 功能的配置策略示例：

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

2.6.4.2.7. indent function

indent 功能会返回经过 padded 的 **data string**。查看该功能的以下语法：

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

使用这个功能时，输入带有特定空格数的数据字符串。查看以下使用 **indent** 功能的配置策略示例：

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

2.6.4.2.8. autoindent 功能

autoindent 函数的作用类似于 **indent** 函数，它根据模板前面的空格数自动决定前导空格的数量。查看以下使用 **autoindent** 函数的配置策略示例：

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

2.6.4.2.9. toInt function

toInt 函数处理并返回输入值的整数值。另外，如果这是模板中的最后一个功能，也会进一步处理源内容。这是为了确保该值由 YAML 解释为整数。查看该功能的以下语法：

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

使用这个功能时，输入需要转换为整数的数据。查看以下使用 **toInt** 功能的配置策略示例：

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

2.6.4.2.10. toBool function

toBool 函数将输入字符串转换为布尔值，并返回布尔值。另外，如果这是模板中的最后一个功能，也会进一步处理源内容。这是为了确保该值被 YAML 解释为布尔值。查看该功能的以下语法：

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

使用此功能时，请输入需要转换为布尔值的字符串数据。查看以下使用 **toBool** 函数的配置策略示例：

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

2.6.4.2.11. 保护功能

通过 **protect** 功能，您可以在 hub 集群策略模板中对字符串进行加密。评估策略时，它将在受管集群上自动解密。查看以下使用 **protect** 功能的配置策略示例：

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

在前面的 YAML 示例中，定义了使用 **lookup** 功能的现有 hub 集群策略模板。在受管集群命名空间中的复制策略上，值可能类似以下语法
：**$ocm_encrypted:okrrBqt72oI+3WT/0vxeI3vGa+wpLD7Z0ZxFMLvL204=**

使用的每个加密算法是 256 位密钥的 AES-CBC。对于每个受管集群，每个加密密钥都需要是唯一的，每 30 天自动轮转。

这样可确保您的解密的值永不会存储在受管集群的策略中。

要强制立即轮转，在 hub 集群上删除 **policy-encryption-key** Secret 上的 **policy.open-cluster-management.io/last-rotated** 注解。然后，会重新处理策略以使用新的加密密钥。

2.6.4.3. 在配置策略中支持 hub 集群模板

除了动态地自定义目标集群的受管集群模板外，Red Hat Advanced Cluster Management 还支持 hub 集群模板，以使用 hub 集群中的值定义配置策略。这个组合减少了在每个目标集群或策略定义中创建硬编码配置值的需求。

hub 集群模板基于 Golang 文本模板规格，**{{hub ... hub}}** 分隔符表示配置策略中的 hub 集群模板。

为安全起见，hub 集群模板中的特定于资源和通用查询功能都仅限于 hub 集群上策略的命名空间。查看 hub 和受管集群模板的比较以了解更多详情。

重要： 如果您使用 hub 集群模板传播 secret 或其他敏感数据，则敏感数据存在于 hub 集群上的受管集群命名空间和发布该策略的受管集群中。模板内容在策略中扩展，策略不会由 OpenShift Container Platform ETCD 加密支持加密。要解决此问题，请使用 **fromSecret**，它会自动从 Secret 中加密值，或使用 **protect** 加密其他值。

2.6.4.3.1. 模板处理

配置策略定义可以包含 hub 集群和受管集群模板。hub 集群模板首先在 hub 集群中处理，然后将带有已解析 hub 集群模板的策略定义传播到目标集群。在受管集群中，**ConfigurationPolicyController** 处理策略定义中的任何受管集群模板，然后强制执行或验证完全解析的对象定义。

2.6.4.3.2. 重新处理的特殊注解

只有在创建或更新后，才会在 hub 集群中处理策略。因此，hub 集群模板仅在策略创建或更新时解析到引用的资源中的数据。对引用资源的任何更改都不会自动与策略同步。

可以使用特殊的注解 **policy.open-cluster-management.io/trigger-update** 来指示模板引用数据的更改。对特殊注解值的任何更改都会启动模板处理，引用资源的最新内容都会在作为在受管主机上处理的传播器的策略定义中读取和更新。使用此注解的一种典型方法是每次递增值。

2.6.4.3.3. 绕过模板处理

您可能会创建一个策略，其中包含不是由 Red Hat Advanced Cluster Management 处理的模板。默认情况下，Red Hat Advanced Cluster Management 会处理所有模板。

要绕过 hub 集群的模板处理，必须将 **{{ template content }}** 改为 **{{ `{{ template content }}` }}**。

另外，您还可以在 **Policy** 的 **ConfigurationPolicy** 部分添加以下注解：**policy.open-cluster-management.io/disable-templates: "true"**。当包含此注解时，则不需要以前的临时解决方案。为 **ConfigurationPolicy** 绕过模板处理。

如需 hub 集群和受管集群模板的比较，请参阅下表：

2.6.4.3.4. hub 集群和受管集群模板的比较

表 2.6. 比较表

模板	hub 集群	受管集群（managed cluster）
Syntax	golang 文本模板规格	golang 文本模板规格
Delimiter	{{hub ... hub}}	{{ ... }}
Context	**.ManagedClusterName** 变量在运行时解析为传播策略的目标集群的名称。	没有上下文变量
Access control	您只能引用与 **Policy** 资源相同的命名空间中的命名空间 Kubernetes 对象。	您可以引用集群中的任何资源。
Functions	组模板功能，支持对 Kubernetes 资源和字符串操作的动态访问。如需更多信息，请参阅模板功能。有关查询限制，请参阅 Access control 行。 hub 集群上的 **fromSecret** 模板功能将生成的值作为加密字符串存储在受管集群命名空间中。 等效的调用可能使用以下语法：**{{hub "(lookup "v1" "Secret" "default" "my-hub-secret").data.message | protect hub}}**	组模板功能，支持对 Kubernetes 资源和字符串操作的动态访问。如需更多信息，请参阅模板功能。
Function output storage	在与受管集群同步前，模板功能的输出存储在 hub 集群上每个适用的受管集群命名空间中的 **Policy** 资源对象中。这意味着，对 hub 集群上的 **Policy** 资源对象具有读取访问权限的模板功能的任何敏感结果，以及受管集群中的 **ConfigurationPolicy** 资源对象的读取访问权限。另外，如果启用了 etcd 加密，则 **Policy** 和 **ConfigurationPolicy** 资源对象不会被加密。使用返回敏感输出的模板功能（如从机密中）时，最好仔细考虑这一点。	模板功能的输出不存储在策略相关的资源对象中。
Processing	在 hub 集群的运行时，处理会在复制策略传播到集群的过程中进行。只有在创建或更新模板时，策略中的策略和 hub 集群模板才会在 hub 集群中处理。	处理发生在受管集群上的 **ConfigurationPolicyController** 中。策略定期处理，利用所引用资源中的数据自动更新解析的对象定义。

模板	hub 集群	受管集群（managed cluster）
Processing errors	hub 集群模板中的错误显示为策略应用到的受管集群中的违反情况。	受管集群模板中的错误会在发生违反情况的特定目标集群中以违反的形式显示。

2.6.5. 监管指标

策略框架公开指标来显示策略分布和合规性。在 hub 集群中使用 **policy_governance_info** 指标来查看趋势并分析任何策略失败。

2.6.5.1. 指标概述

OpenShift Container Platform 监控会收集 **policy_governance_info**，如果已启用，Red Hat Advanced Cluster Management observability 会收集一些聚合数据。

注 ：如果启用了可观察性，您可以从 Grafana *Explore* 页面输入对指标的查询。

创建策略时，您要创建一个 *root* 策略。框架监视根策略以及 **PlacementRules** 和 **PlacementBindings**，以确定在哪里创建 *propagated* 策略，以便将策略分发到受管集群。对于根和传播策略，如果策略合规，则指标会记录 **0**，如果策略不合规，则记录 **1**。

policy_governance_info 指标使用以下标签：

- **type** ：标签值为 **root** 或 **propagated**。
- **policy** ：关联的根策略的名称。
- **policy_namespace** ：定义根策略的 hub 集群上命名空间。
- **cluster_namespace** ：分发策略的集群的命名空间。

这些标签和值启用查询来显示集群中可能发生的许多事情，这些情况可能很难跟踪。

注 ：如果不需要指标数据，且对性能或安全性有任何顾虑，则可以禁用此功能。在传播器部署中，将 **DISABLE_REPORT_METRICS** 环境变量设置为 **true**。您还可以将 **policy_governance_info** 指标添加到 observability allowlist 作为自定义指标。如需了解更多详细信息，请参阅 添加自定义指标。

2.6.6. 管理安全策略

创建一个安全策略，根据您指定的安全标准、类别和控制，报告并验证您的集群合规性。要为 Red Hat Advanced Cluster Management for Kubernetes 创建策略，您必须在受管集群上创建 YAML 文件。

注：您可以将现有策略复制到 *Policy YAML* 中。当您粘贴现有策略时，会自动输入参数字段的值。您还可以使用搜索功能搜索策略 YAML 文件中的内容。

查看以下部分：

- 创建安全策略
  - 使用命令行界面创建安全策略
  - 通过 CLI 查看您的安全策略
  - 从控制台创建集群安全策略

- 从控制台查看您的安全策略
- 通过 CLI 创建策略设置
- 从控制台创建策略集

- 更新安全策略
  - 禁用安全策略
- 删除安全策略
  - 从控制台创建策略集

2.6.6.1. 创建安全策略

您可以使用命令行界面 (CLI) 或者从控制台创建安全策略。

需要的访问权限： 集群管理员

重要 ：您必须定义放置规则和放置绑定，才能将策略应用到特定集群。为 *Cluster selector* 字段输入有效的值，以定义 **PlacementRule** 和 **PlacementBinding**。如需有效表达式，请参阅 Kubernetes 文档中的支持基于集合的要求的资源。查看 Red Hat Advanced Cluster Management 策略所需的对象定义：

- *PlacementRule*：定义必须部署策略的*集群选择器*。
- *PlacementBinding* ：将放置绑定到放置规则。

在策略概述中查看策略 YAML 文件的更多描述。

2.6.6.1.1. 使用命令行界面创建安全策略

完成以下步骤，使用命令行界面 (CLI) 创建策略：

1. 运行以下命令来创建策略：

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

2. 定义策略使用的模板。要编辑 **.yaml** 文件，请添加 **templates** 字段来定义模板。您的策略可能类似以下 YAML 文件：

```
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. 定义 **PlacementRule**。务必更改 **PlacementRule**，以通过 **clusterNames** 或 **clusterLabels** 指定需要应用策略的集群。查看放置规则示例概述
   您 **PlacementRule** 可能类似如下内容：

```
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. 定义一个 **PlacementBinding** 来绑定您的策略和 **PlacementRule**。您 **PlacementBinding** 可能类似以下 YAML 示例：

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

2.6.6.1.1.1. 通过 CLI 查看您的安全策略

完成以下步骤，通过 CLI 查看您的安全策略：

1. 运行以下命令，查看具体安全策略的详情：

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

2. 运行以下命令，查看您的安全策略的描述：

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

2.6.6.1.2. 从控制台创建集群安全策略

登录到 Red Hat Advanced Cluster Management 后，进入 *Governance* 页面并点 **Create policy**。

从控制台创建新策略时，也会在 YAML 编辑器中创建 YAML 文件。要查看 YAML 编辑器，请在 *Create policy* 表单的开头选择切换来启用它。

完成 *Create policy* 表单，然后选择 **提交**按钮。

您的 YAML 文件可能类似以下策略：

```
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  name: policy-pod
  annotations:
    policy.open-cluster-management.io/categories:
'SystemAndCommunicationsProtections,SystemAndInformationIntegrity'
    policy.open-cluster-management.io/controls: 'control example'
    policy.open-cluster-management.io/standards: 'NIST,HIPAA'
spec:
  complianceType: musthave
  namespaces:
    exclude: ["kube*"]
    include: ["default"]
  object-templates:
  - complianceType: musthave
    objectDefinition:
      apiVersion: v1
      kind: Pod
      metadata:
        name: pod1
      spec:
        containers:
        - name: pod-name
          image: 'pod-image'
          ports:
          - containerPort: 80
  remediationAction: enforce
  disabled: false

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

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

---
apiVersion: apps.open-cluster-management.io/v1
kind: PlacementRule
metadata:
  name: placement-pod
spec:
  clusterConditions:
    - type: ManagedClusterConditionAvailable
      status: "True"
  clusterLabels:
    matchLabels:
      cloud: "IBM"
```

点击 **Create Policy**。从控制台创建了安全策略。

2.6.6.1.2.1. 从控制台查看您的安全策略

在控制台中查看任何安全策略及其状态。进入 *Governance* 页面，以查看您的策略的表列表。**注**：您可以通过选择 *Policies* 标签页或 *Cluster violations* 标签页来过滤策略列表。

选择一个策略来查看更多详情。此时会显示 *Details*、*Clusters* 和 *Templates* 标签页。当无法决定集群或策略状态时，会显示以下信息： **No status**。

2.6.6.1.3. 通过 CLI 创建策略设置

默认情况下，策略集是在没有策略或放置的情况下创建的。您必须为策略集合创建放置，并至少有一个策略存在于集群中。在创建策略集时，您可以添加多个策略。运行以下命令，通过 CLI 创建策略集：

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

2.6.6.1.4. 从控制台创建策略集

在导航菜单中选择 **Governance**。然后选择 *Policy set* 选项卡。选择 **Create policy set**按钮并完成表单。添加策略集详情后，选择 **提交** 按钮。

查看 stable **Policyets**，这需要部署策略生成器，PolicySets-table。

2.6.6.2. 更新安全策略

查看以下部分以了解如何更新安全策略。

2.6.6.2.1. 通过 CLI 将策略添加到策略集中

运行以下命令来编辑您的策略设置：**kubectl edit policysets your-policyset-name**

将策略名称添加到策略集的 **policies** 部分的列表中。使用以下命令在策略集的放置部分中应用添加的策略： **kubectl apply -f your-added-policy.yaml**。已创建一个 **PlacementBinding** 和 **PlacementRule**。**注意** ：如果您删除放置绑定，策略仍然由策略集放置。

2.6.6.2.2. 从控制台在策略集中添加策略

选择 ***Policy set*** 选项卡在策略集中添加一个策略。选择 Actions 图标并选择 **Edit**。此时会出现 *Edit policy set* 表单。

进入到表单的 *Policies* 部分，以选择要添加到策略集的策略。

2.6.6.2.3. 禁用安全策略

您的策略默认是启用的。从控制台禁用您的策略。

登录到 Red Hat Advanced Cluster Management for Kubernetes 控制台后，进入 *Governance* 页面来查看您的策略的表列表。

选择 **Actions** 图标 > **Disable policy**。此时会出现 *Disable Policy* 对话框。

点击 **Disable policy**。您的策略已禁用。

2.6.6.3. 删除安全策略

通过 CLI 或控制台删除安全策略。

- 通过 CLI 删除安全策略：
  a. 运行以下命令来删除安全策略：

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

  删除策略后，它会从一个或多个目标集群中移除。运行以下命令验证您的策略是否已移除： **kubectl get policy <securitypolicy-name> -n <open-cluster-management-namespace>**

- 从控制台删除安全策略：
  在导航菜单中点 **Governance** 来查看您的策略的表列表。在策略违反表中点击您要删除的策略的 **Actions** 图标。

  点击 **Remove**。在 *Remove policy* 对话框中点击 **Remove policy**

2.6.6.3.1. 从控制台创建策略集

在 *Policy set* 选项卡中，选择策略集的 **Actions** 图标。当您单击 **Delete** 时，会出现 *Permanently delete Policyset?* 对话框。

点击 **Delete** 按钮。

要管理其他策略，请参阅管理安全策略以了解更多信息。有关策略的更多主题，请参阅监管。

2.6.7. 管理配置策略

了解如何创建、应用、查看和更新您的配置策略。提醒，以下资源是配置策略：

- 内存用量策略
- 命名空间策略
- 镜像漏洞策略
- Pod 策略
- Pod 安全策略
- 角色策略
- 角色绑定策略
- 安全内容约束 (SCC) 策略
- ETCD 加密策略
- Compliance operator 策略

需要的访问权限 ： 管理员或集群管理员

- 创建配置策略
  - 通过 CLI 创建配置策略
  - 通过 CLI 查看您的配置策略
  - 从控制台创建配置策略
  - 从控制台查看您的配置策略
- 更新配置策略
  - 禁用配置策略
- 删除配置策略

2.6.7.1. 创建配置策略

您可以使用命令行界面 (CLI) 或者从控制台为配置策略创建 YAML 文件。查看以下部分以创建配置策略：

2.6.7.1.1. 通过 CLI 创建配置策略

完成以下步骤，通过 CLI 创建配置策略：

1. 为您的配置策略创建一个 YAML 文件。运行以下命令：

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

您的配置策略可能类似以下策略：

```
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. 运行以下命令来应用策略：

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

3. 运行以下命令，验证并列出策略：

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

您的配置策略已创建。

2.6.7.1.2. 通过 CLI 查看您的配置策略

完成以下步骤，通过 CLI 查看您的配置策略：

1. 运行以下命令，查看具体配置策略的详情：

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

2. 运行以下命令，查看您的配置策略的描述：

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

2.6.7.1.3. 从控制台创建配置策略

从控制台创建配置策略时，也会在 YAML 编辑器中创建 YAML 文件。

从控制台登录到集群，然后从导航菜单中选择 **Governance**。

点击 **Create policy**。通过为规格参数选择一个配置策略来指定您要创建的策略。

通过完成策略表单继续配置策略创建。为以下字段输入或选择适当的值：

- Name
- Specifications
- Cluster selector
- Remediation action
- Standards
- Categories

- Controls

点击 **Create**。您的配置策略已创建。

2.6.7.1.4. 从控制台查看您的配置策略

在控制台中查看任何配置策略及其状态。

在控制台中登录到集群后，选择 **Governance** 来查看您的策略的表列表。**注**：您可以选择 *All policies* 选项卡或者 *Cluster violations* 选项卡来过滤您的策略表列表。

选择一个策略来查看更多详情。此时会显示 *Details*、*Clusters* 和 *Templates* 标签页。

2.6.7.2. 更新配置策略

查看以下部分以了解如何更新配置策略。

2.6.7.2.1. 禁用配置策略

禁用您的配置策略。与前面提到的说明类似，登录并进入到 *Governance* 页面。

从表列表中选择配置策略的 **Actions** 图标，然后点 **Disable**。此时会出现 *Disable Policy* 对话框。

点击 **Disable policy**。

您的配置策略已禁用。

2.6.7.3. 删除配置策略

通过 CLI 或控制台删除配置策略。

- 通过 CLI 删除配置策略：
  a. 运行以下命令来删除配置策略：

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

  删除策略后，它会从一个或多个目标集群中移除。

  b. 运行以下命令验证您的策略是否已移除：

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

- 通过控制台删除配置策略：
  在导航菜单中点 **Governance** 来查看您的策略的表列表。

  在策略违反表中点击您要删除的策略的 **Actions** 图标。然后点**删除**。在 *Remove policy* 对话框中点击 **Remove policy**。

您的策略已删除。

如需被 Red Hat Advanced Cluster Management 支持的配置策略示例，查看 CM-Configuration-Management 文件夹。

或者，参阅 Kubernetes 配置策略控制器来查看控制器监控的其他配置策略。有关管理其他策略的详情，请参阅管理安全策略。

2.6.8. 管理 gatekeeper Operator 策略

使用 gatekeeper operator 策略在受管集群上安装 gatekeeper operator 和 gatekeeper。在以下部分中了解如何创建、查看和更新您的 gatekeeper Operator 策略。

需要的访问权限： 集群管理员

- 使用 gatekeeper operator 策略安装 gatekeeper
- 从控制台创建 gatekeeper 策略
  - Gatekeeper operator CR
- 升级 gatekeeper 和 gatekeeper operator
- 更新 gatekeeper operator 策略
  - 从控制台查看 gatekeeper operator 策略
  - 禁用 gatekeeper operator 策略
- 删除 gatekeeper operator 策略
- 卸载 gatekeeper 策略、gatekeeper 和 gatekeeper operator 策略

2.6.8.1. 使用 gatekeeper operator 策略安装 gatekeeper

使用监管框架来安装 gatekeeper Operator。OpenShift Container Platform 目录中提供了 gatekeeper operator。如需更多信息，请参阅 OpenShift Container Platform 文档 中的 *Adding Operators to a cluster* 部分。

使用配置策略控制器来安装 gatekeeper operator 策略。在安装过程中，Operator 组和订阅会拉取 gatekeeper operator 将其安装到受管集群中。然后，gatekeeper operator 会创建一个 gatekeeper CR 来配置 gatekeeper。查看 Gatekeeper operator CR 示例。

在支持 **enforce（强制）** 补救操作的情况下，gatekeeper operator 策略由 Red Hat Advanced Cluster Management 配置策略控制器监控。当设置为 **enforce** 时，控制器会自动创建 Gatekeeper Operator 策略。

2.6.8.2. 从控制台创建 gatekeeper 策略

通过从控制台创建 gatekeeper 策略来安装 gatekeeper 策略。

登录到集群后，进入 *Governance* 页面。

选择 **Create policy**。在填写表单时，从 *Specifications* 项中选择 **GatekeeperOperator**。策略的参数值会自动填充，策略默认设置为 **inform**。将补救操作设置为 **enforce** 来安装 gatekeeper。请参阅 **policy-gatekeeper-operator.yaml** 查看示例。

+ **注：** 考虑可由 Operator 生成默认值。如需了解可用于 gatekeeper operator 策略的可选参数的说明，请参阅 Gatekeeper Helm Chart。

2.6.8.2.1. Gatekeeper operator CR

-

```
apiVersion: operator.gatekeeper.sh/v1alpha1
kind: Gatekeeper
metadata:
  name: gatekeeper
spec:
  audit:
    replicas: 1
    logLevel: DEBUG
    auditInterval: 10s
    constraintViolationLimit: 55
    auditFromCache: Enabled
    auditChunkSize: 66
    emitAuditEvents: Enabled
    resources:
      limits:
        cpu: 500m
        memory: 150Mi
      requests:
        cpu: 500m
        memory: 130Mi
  validatingWebhook: Enabled
  webhook:
    replicas: 2
    logLevel: ERROR
    emitAdmissionEvents: Enabled
    failurePolicy: Fail
    resources:
      limits:
        cpu: 480m
        memory: 140Mi
      requests:
        cpu: 400m
        memory: 120Mi
  nodeSelector:
    region: "EMEA"
  affinity:
    podAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        - labelSelector:
            matchLabels:
              auditKey: "auditValue"
          topologyKey: topology.kubernetes.io/zone
  tolerations:
    - key: "Example"
      operator: "Exists"
      effect: "NoSchedule"
  podAnnotations:
    some-annotation: "this is a test"
    other-annotation: "another test"
```

2.6.8.3. 升级 gatekeeper 和 gatekeeper operator

您可以升级 gatekeeper 和 gatekeeper operator 的版本。完成以下步骤：

- 当使用 gatekeeper operator 策略安装 gatekeeper operator 时，请注意 **installPlanApproval** 的

值。当 **installPlanApproval** 设置为 **Automatic** 时，Operator 会自动升级。当 **installPlanApproval** 设置为 **Manual** 时，您必须为每个集群手动批准 gatekeeper operator 的升级。

2.6.8.4. 更新 gatekeeper operator 策略

查看以下部分以了解如何更新 gatekeeper operator 策略。

2.6.8.4.1. 从控制台查看 gatekeeper operator 策略

在控制台中查看您的 gatekeeper operator 策略及其状态。

从控制台登录到集群后，点 **Governance** 来查看您的策略的表列表。**注**：您可以通过选择 *Policies* 标签页或 *Cluster violations* 标签页来过滤策略列表。

选择 **policy-gatekeeper-operator** 策略来查看更多详细信息。选择 *Clusters* 选项卡来查看策略违反情况。

2.6.8.4.2. 禁用 gatekeeper operator 策略

禁用 gatekeeper operator 策略。

登录到 Red Hat Advanced Cluster Management for Kubernetes 控制台后，进入 *Governance* 页面来查看您的策略的表列表。

选择 **policy-gatekeeper-operator** 策略的 **Actions** 图标，然后点 **Disable**。此时会出现 *Disable Policy* 对话框。

点击 **Disable policy**。您的 **policy-gatekeeper-operator** 策略已被禁用。

2.6.8.5. 删除 gatekeeper operator 策略

通过 CLI 或控制台删除 gatekeeper operator 策略。

- 通过 CLI 删除 gatekeeper operator 策略：
  a. 运行以下命令来删除 gatekeeper operator 策略：

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

     删除策略后，它会从一个或多个目标集群中移除。

  b. 运行以下命令验证您的策略是否已移除：

     ```
     kubectl get policy <policy-gatekeeper-operator-name> -n <namespace>
     ```

- 从控制台删除 gatekeeper operator 策略：
  进入 *Governance* 页面，以查看您的策略的表列表。

  与之前的控制台指令类似，点 **policy-gatekeeper-operator** 策略的 **Actions** 图标。点 **Remove** 以删除该策略。在 *Remove policy* 对话框中点击 **Remove policy**。

您的 gatekeeper operator 策略被删除。

2.6.8.6. 卸载 gatekeeper 策略、gatekeeper 和 gatekeeper operator 策略

完成以下步骤以卸载 gatekeeper 策略、gatekeeper 和 gatekeeper operator 策略：

1. 删除应用到受管集群的 gatekeeper **Constraint** 和 **ConstraintTemplate**：
   a. 编辑 gatekeeper operator 策略。找到您用来创建 gatekeeper **Constraint** 和 **ConstraintTemplate** 的 **ConfigurationPolicy** 模板 。
   b. 将 **ConfigurationPolicy** 模板的 **complianceType** 的值改为 **mustnothave**。
   c. 保存并应用策略。
2. 从受管集群中删除 gatekeeper 实例：
   a. 编辑 gatekeeper operator 策略。找到用于创建 Gatekeeper 自定义资源（CR）的 **ConfigurationPolicy** 模板。
   b. 将 **ConfigurationPolicy** 模板的 **complianceType** 的值改为 **mustnothave**。
3. 删除受管集群中的 gatekeeper Operator：
   a. 编辑 gatekeeper operator 策略。找到用于创建 Subscription CR 的 **ConfigurationPolicy** 模板。
   b. 将 **ConfigurationPolicy** 模板的 **complianceType** 的值改为 **mustnothave**。

Gatekeeper 策略、gatekeeper 和 gatekeeper operator 策略会被卸载。

如需有关 gatekeeper 的详细信息，请参阅集成 gatekeeper 约束和约束模板。有关将第三方策略与产品集成的主题列表，请参阅集成第三方策略控制器。

2.7. 保护 HUB 集群

通过强化 hub 集群安全性来保护 Red Hat Advanced Cluster Management for Kubernetes 安装。完成以下步骤：

1. 保护 Red Hat OpenShift Container Platform。如需更多信息，请参阅 OpenShift Container Platform 安全性和合规性。
2. 设置基于角色的访问控制(RBAC)。如需更多信息，请参阅基于角色的访问控制。
3. 自定义证书，请参阅证书。
4. 定义集群凭证，请参阅管理凭证概述
5. 查看可帮助您强化集群安全性的策略。请参阅支持的策略

2.8. 完整性 SHIELD 保护（技术预览）

完整性屏蔽是一种工具，它有助于为创建或更新资源的任何请求执行签名验证。完整性截图支持开放策略代理(OPA)和 Gatekeeper，验证请求是否有签名，并根据定义的约束阻止任何未授权的请求。

请参阅以下完整性盾牌功能：

- 仅支持部署授权的 Kubernetes 清单。
- 在资源配置中支持零偏移，除非资源已添加到 allowlist 中。

- 在集群上执行所有完整性验证，如强制准入控制器。
- 如果集群中部署了未经授权的 Kubernetes 资源，则持续监控资源以报告。
- X509、GPG 和 Sigstore 签名支持为 Kubernetes 清单 YAML 文件签名。Kubernetes 完整性 shield 使用 **k8s-manifest-sigstore** 支持 Sigstore 签名。

2.8.1. 完整性 shield 架构

完整性屏蔽由两个主要组件组成，即 API 和 Observer。完整性 shield operator 支持在集群上安装和管理完整的 shield 组件。查看以下组件描述：

- *完整性 shield API* 从 OPA 或 gatekeeper 接收 Kubernetes 资源，验证准入请求中包含的资源，并将验证结果发送到 OPA 或 gatekeeper。shield API 在内部使用 **k8s-manifest-sigstore** 的 **verify-resource** 功能来验证 Kubernetes 清单 YAML 文件。完整性 shield API 根据 **ManifestingIntegrityConstraint** （是基于 OPA 或 gatekeeper 的约束框架的自定义资源）验证资源。
- *完整性 shield Observer* 根据 **ManifestingIntegrityConstraint** 资源持续验证集群中的 Kubernetes 资源，并将结果导出到名为 **ManifestIntegrityState** 的资源。完整性 shield Observer 还使用 **k8s-manifest-sigstore** 来验证签名。

2.8.2. 支持的版本

以下产品版本支持完整性 shield 保护：

- Red Hat OpenShift Container Platform 4.7.1 及更新的版本
- Kubernetes v1.19.7 及更新的版本
- **gatekeeper-operator.v-.2.0**
- **gatekeeper v3.5**

如需了解更多详细信息，请参阅 启用完整性 shield 保护（技术预览）。

2.8.3. 启用完整性 shield 保护（技术预览）

在 Red Hat Advanced Cluster Management for Kubernetes 集群中启用完整性 shield 保护 Kubernetes 资源的完整性。

2.8.3.1. 先决条件

在 Red Hat Advanced Cluster Management 受管集群中启用完整性 shield 保护需要以下先决条件：

- 安装具有一个或多个受管集群的 Red Hat Advanced Cluster Management hub 集群，以及集群管理员对集群的访问权限，以使用 **oc** 或 **kubectl** 命令。
- 安装完整性 shield.在安装完整性 shield 前，您必须在集群中安装 Open Policy Agent 或 gatekeeper。完成以下步骤以安装完整性 shield operator:

  a. 运行以下命令，在命名空间中安装完整性 shield operator 以获得完整性 shield：

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

b. 使用以下命令安装完整性 shield 自定义资源：

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

c. 完整性屏蔽需要一对密钥来签名和验证集群中需要保护的资源签名。设置签名和验证密钥对：

- 使用以下命令生成一个新的 GPG 密钥：

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

- 使用以下命令将您的新 GPG 公钥导出到文件中：

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

- 安装 **yq** 以运行用于签署 Red Hat Advanced Cluster Management 策略的脚本。
- 启用完整性 shield 保护和签名 Red Hat Advanced Cluster Management 包括从 **integrity-shield** 仓库检索和提交源。您必须安装 Git。

2.8.3.2. 启用完整性 shield 保护

通过完成以下步骤，在 Red Hat Advanced Cluster Management 受管集群中启用完整性 shield：

1. 为完整性 shield 在 hub 集群上创建一个命名空间。运行以下命令：

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

2. 在 Red Hat Advanced Cluster Management 受管集群中部署验证密钥。提醒，您必须创建签名和验证密钥。在 hub 集群上运行 **acm-verification-key-setup.sh** 以设置验证密钥。运行以下命令：

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

要删除验证密钥，请运行以下命令：

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

3. 在 hub 集群中创建一个名为 **policy-integrity-shield** 的 Red Hat Advanced Cluster Management 策略。

a. 从 **policy-collection** 存储库获取 **policy-integrity-shield** 策略。确定 fork 了 存储库。

b. 通过更新 **remediationAction** 参数值（从 **inform** 到 **enforce** ），将命名空间配置为在 Red Hat Advanced Cluster Management 受管集群上部署完整性 shield。

c. 通过更新 **signerConfig** 部分，为签名者和验证密钥配置电子邮件。

d. 配置 **PlacementRule** 决定了 Red Hat Advanced Cluster Management 受管集群应该部署到哪些集群。

e. 运行以下命令，签署 **policy-integrity-shield.yaml** ：

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

注 ：每当您更改策略并应用到其他集群时，您必须创建新的签名。否则，更改会被阻止且不应用。

如需示例，请参阅 **policy-integrity-shield** 策略。