Red Hat Advanced Cluster Management for Kubernetes 2.10

监管

监管

Last Updated: 2024-07-01

Red Hat Advanced Cluster Management for Kubernetes 2.10 监管

监管

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 安全概述

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。

阅读以下主题以了解更多有关保护集群的信息：

- 证书简介
- 监管

1.1. 证书简介

您可以使用各种证书来验证 Red Hat Advanced Cluster Management for Kubernetes 集群的真实性。另外，您可以自带证书。继续阅读以了解证书管理的信息。

- 证书
- 使用您自己的可观察证书颁发机构 (CA) 证书
- 管理证书

1.2. 证书

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。查看以下证书列表，这些证书由 Red Hat OpenShift Container Platform 的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

需要的访问权限： 集群管理员

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
- Red Hat Advanced Cluster Management 管理的证书

注意 ：用户负责证书轮转和更新。

1.2.1. Red Hat Advanced Cluster Management hub 集群证书

OpenShift 默认入口证书技术是一个 hub 集群证书。安装 Red Hat Advanced Cluster Management 后，会创建可观察性证书，供可观察性组件用来在 hub 集群和受管集群之间的流量上提供 mutual TLS。

- **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。

查看以下 Red Hat Advanced Cluster Management hub 集群证书表：

表 1.1. Red Hat Advanced Cluster Management hub 集群证书

Namespace	Secret 名称	Pod 标签	
open-cluster-management	channels-apps-open-cluster-management-webhook-svc-ca	app=multicluster-operators-channel	open-cluster-management
channels-apps-open-cluster-management-webhook-svc-signed-ca	app=multicluster-operators-channel	open-cluster-management	multicluster-operators-application-svc-ca
app=multicluster-operators-application	open-cluster-management	multicluster-operators-application-svc-signed-ca	app=multicluster-operators-application
open-cluster-management-hub	registration-webhook-serving-cert signer-secret	不是必需的	open-cluster-management-hub

1.2.2. Red Hat Advanced Cluster Management 管理的证书

查看下表，了解包含 Red Hat Advanced Cluster Management 管理的证书和相关 secret 的组件 pod 的总结列表：

表 1.2. 包含 Red Hat Advanced Cluster Management 管理证书的 Pod

Namespace	Secret 名称（如果适用）
open-cluster-management-agent-addon	cluster-proxy-open-cluster-management.io-proxy-agent-signer-client-cert
open-cluster-management-agent-addon	cluster-proxy-service-proxy-server-certificates

1.2.2.1. 受管集群证书

您可以使用证书通过 hub 集群验证受管集群。因此，了解与这些证书关联的故障排除方案非常重要。

受管集群证书会自动刷新。

1.2.3. 其他资源

- 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。如需了解更多详细信息，请参阅证书策略控制器。
- 有关使用 SSL/TLS 证书安全连接到私有托管的 Git 服务器的更多详细信息，请参阅使用自定义 CA 证书进行安全 HTTPS 连接。
- 如需了解更多详细信息，请参阅 OpenShift Service Serving 证书。
- OpenShift Container Platform 默认入口是一个 hub 集群证书。如需了解更多详细信息，请参阅重新放置 默认入口证书。
- 有关主题，请参阅 证书简介。

1.2.4. 使用您自己的可观察证书颁发机构 (CA) 证书

安装 Red Hat Advanced Cluster Management for Kubernetes 时，默认只为可观察性提供证书颁发机构 (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自己的可观察 CA 证书。

1.2.4.1. 使用 OpenSSL 命令生成 CA 证书

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
```

1.2.4.2. 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
```

1.2.4.3. 其他资源

- 请参阅 管理证书。
- 返回 证书简介。

1.2.5. 管理证书

继续阅读以了解有关如何刷新、替换、轮转和列出证书的信息。

- 刷新 Red Hat Advanced Cluster Management Webhook 证书
- 替换 alertmanager 路由的证书
- 轮转 Gatekeeper Webhook 证书
- 验证证书轮转
- 列出 hub 集群受管证书

1.2.5.1. 刷新 Red Hat Advanced Cluster Management Webhook 证书

您可以刷新 Red Hat Advanced Cluster Management 受管证书，它们是由 Red Hat Advanced Cluster Management 服务创建和管理的证书。

完成以下步骤以刷新 Red Hat Advanced Cluster Management 管理的证书：

1. 运行以下命令，删除与 Red Hat Advanced Cluster Management 管理证书关联的 secret：

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

(1) 将 **<namespace>** 和 **<secret>** 替换为您要使用的值。

2. 运行以下命令，重启与 Red Hat Advanced Cluster Management 受管证书关联的服务：

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

1 将 **<namespace>** 和 **<pod-label>** 替换为 *Red Hat Advanced Cluster Management 受管集群证书*表中的值。

注： 如果没有指定 **pod-label**，则不会重启任何服务。secret 被重新创建并自动使用。

1.2.5.2. 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
```

1.2.5.3. 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 pod -n openshift-gatekeeper-system -l control-plane=controller-manager
```

gatekeeper Webhook 证书被轮转。

1.2.5.4. 验证证书轮转

按照以下流程验证您的证书是否已轮转：

1. 找到要检查的 secret。
2. 检查 **tls.crt** 密钥以验证证书是否可用。
3. 使用以下命令显示证书信息：

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

将 **<your-secret-name>** 替换为您要验证的 secret 的名称。如果需要，还要更新命名空间和 JSON 路径。

4. 检查输出中的 **Validity** 详情。查看以下 **Validity** 示例：

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

(1) **Not Before** 值是您轮转证书的日期和时间。

(2) **Not After** 值是证书过期的日期和时间。

1.2.5.5. 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
```

如需更多信息，请参阅*附加资源*部分中的 *OpenShift Service Serving 证书*部分。

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。

1.2.5.6. 其他资源

- OpenShift Service Serving 证书
- 证书简介

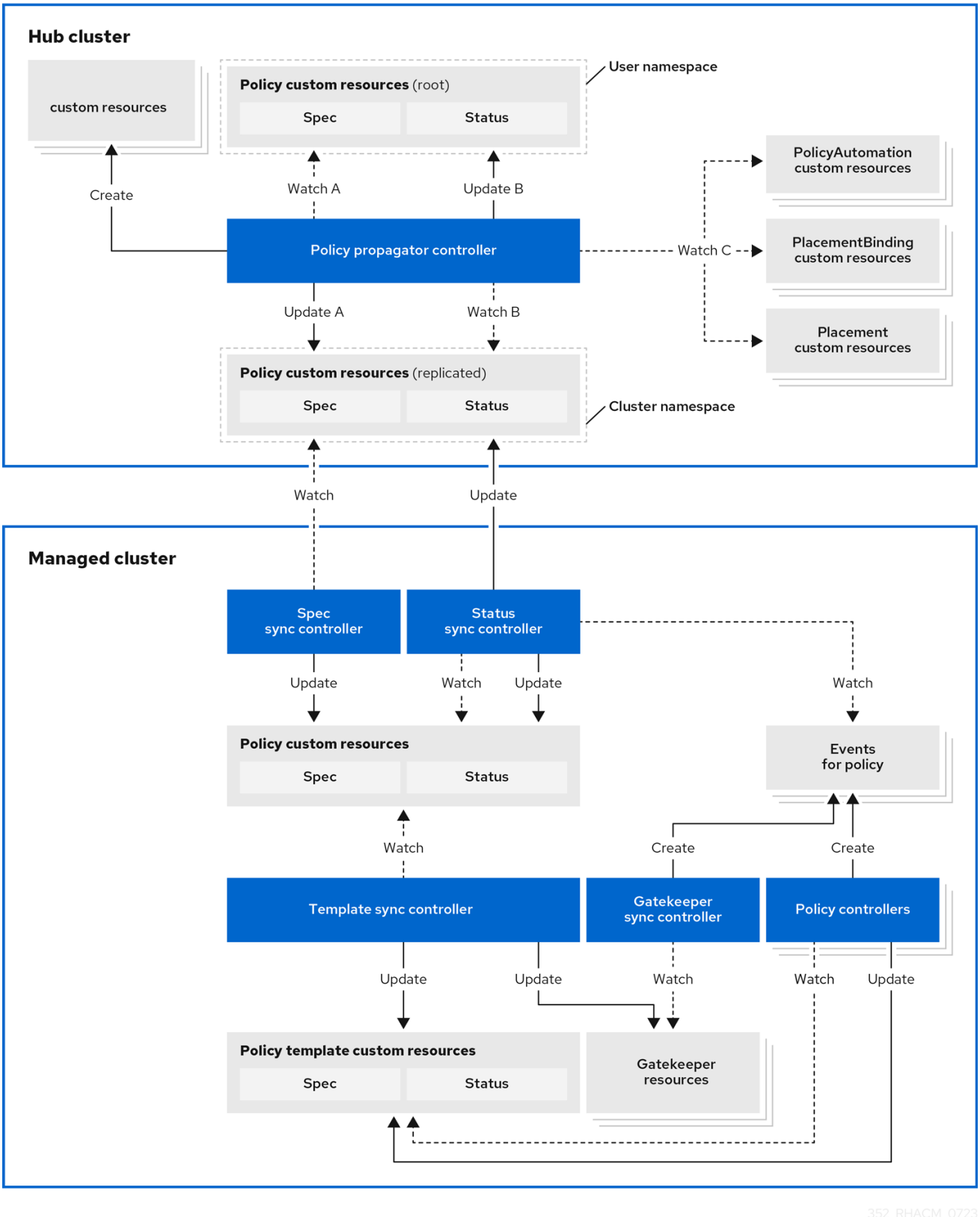
第 2 章 监管

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。继续阅读 Red Hat Advanced Cluster Management 监管框架的相关主题：

- 监管架构
- 策略概述
- 策略控制器简介
- 策略控制器高级配置
- 策略合规历史记录（技术预览）
- 支持的策略
- 策略依赖项
- 管理监管仪表板
- 保护 hub 集群
- 集成第三方策略控制器

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使用受支持的策略、流程和程序从中央接口页面管理安全和合规性。参阅以下监管架构图：



查看以下监管架构图的组件描述：

- **策略传播器控制器：** 在 Red Hat Advanced Cluster Management hub 集群上运行，并根据绑定到 root 策略的放置在 hub 上的受管集群命名空间中生成复制策略。它还将来自复制策略的合规性状态聚合到根策略状态，并根据绑定到 root 策略的策略自动化启动自动化。

- **监管策略附加组件控制器：** 在 Red Hat Advanced Cluster Management hub 集群上运行，并在受管集群中管理策略控制器的安装。

- **监管策略框架 ：** 以前的镜像代表作为受管集群中的 **governance-policy-framework** pod 运行的框架，并包含以下控制器：
  - **spec sync controller：** 将 hub 集群上的受管集群命名空间中的复制策略同步到受管集群上的受管集群命名空间。
  - **Status sync controller ：** 从 hub 和受管集群的复制策略中的策略控制器记录合规事件。状态仅包含与当前策略相关的更新，如果策略被删除和重新创建，则不会考虑过去的状态。
  - **模板同步控制器：** 根据复制策略 **spec.policy-templates** 条目中的定义，创建、更新和删除受管集群命名空间中的对象。
  - **Gatekeeper sync controller ：** 在相应的 Red Hat Advanced Cluster Management 策略中记录 Gatekeeper 约束审计作为合规事件。

2.1.1. 监管架构组件

监管架构还包括以下组件：

- **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。请参阅 *管理监管仪表板* 部分，以了解 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 *Governance* 仪表板。
  备注：
  - 当策略传播到受管集群时，首先会复制到 hub 集群上的集群命名空间中，并使用 **namespaceName.policyName** 命名并标记。在创建策略时，请确保 **namespaceName.policyName** 的长度不超过 63 个字符，因为 Kubernetes 对标签值有限制。
  - 当您在 hub 集群中搜索策略时，您可能还会在受管集群命名空间中收到复制策略的名称。例如，如果您在 **default** 命名空间中搜索 **policy-dhaz-cert**，hub 集群中的以下策略名称可能也会出现在受管集群命名空间中： **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。以下是在 *开源社区* 中向集群部署策略的预定义策略和说明的示例。另外，当出现违反策略的情况时，可以将自动化配置为运行并采取用户选择的任何操作。
- **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 **open-cluster-management/policy-collection** 存储库的一部分。您可以使用 GitOps 贡献和部署策略。

2.1.2. 其他资源

- 请参阅策略控制器简介。
- 请参阅使用 GitOps 部署策略。

2.2. 策略概述

要创建和管理策略，请获得可见性和修复以满足标准的配置，请使用 Red Hat Advanced Cluster Management for Kubernetes 安全策略框架。Kubernetes 自定义资源定义实例用于创建策略。除了受管集群命名空间外，您可以在 hub 集群上的任意命名空间中创建策略。如果在受管集群命名空间中创建策略，它将由 Red Hat Advanced Cluster Management 删除。每个 Red Hat Advanced Cluster Management 策略都可以被组织到一个或多个策略模板定义中。有关策略元素的详情，请查看此页面的 *Policy YAML 表* 部分。

您需要确保受管云环境满足针对 Kubernetes 集群上托管的工作负载的软件工程、安全工程、弹性、安全性和法规合规性的内部企业安全标准。

2.2.1. 先决条件

- 每个策略都需要一个 **放置** 资源来定义策略文档应用到的集群，以及绑定 Red Hat Advanced Cluster Management for Kubernetes 策略的 **PlacementBinding** 资源。有关如何定义 **放置资源的更多信息，请参阅集群生命周期文档中的放置概述**。
- 您必须创建 **PlacementBinding** 将您的策略与一个 **Placement** 绑定，以便将策略应用到受管集群。
- 要使用 **放置资源**，您必须将 **ManagedClusterSet** 资源绑定到带有 **ManagedClusterSetBinding** 资源的 **Placement** 资源的命名空间。如需了解更多详细信息，请参阅创建 *ManagedClusterSetBinding* 资源。
- 从控制台为策略创建放置资源时，放置容限的状态会自动添加到 **放置资源** 中。**如需了解更多详细信息，请参阅 为放置添加容限。**

最佳实践 ：在使用 Placement 资源时，使用命令行界面(CLI) 来更新策略。

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**

2.2.2. 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。查看策略的以下 **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:
    policy.open-cluster-management.io/description:
```

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

2.2.3. 策略 YAML 表

查看以下策略参数描述表：

表 2.1. 参数表

字段	可选或必需的	描述
apiVersion	必填	将值设置为 **policy.open-cluster-management.io/v1**。
kind	必填	将值设为 **Policy** 以表示策略类型。

字段	可选或必需的	描述
metadata.name	必填	用于标识策略资源的名称。
metadata.annotations	选填	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。
bindingOverrides.remediationAction	选填	当此参数设置为 **enforce** 时，它为您提供了覆盖配置策略相关 **PlacementBinding** 资源的补救操作。默认值为 **null**。
subFilter	选填	将此参数设置为 **restriction**，以选择绑定策略的子集。默认值为 **null**。
annotations.policy.open-cluster-management.io/standards	选填	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	选填	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	选填	正在接受检查的安全控制名称。例如，访问控制或系统及信息完整性。
spec.disabled	必填	将值设为 **true** 或 **false**。**disabled** 参数提供启用和禁用策略的功能。

字段	可选或必需的	描述
spec.remediationAction	选填	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 **policy-templates** 部分的子策略中定义的 **remediationAction** 参数。例如，如果 **spec.remediationAction** 值被设置为 **enforce**，则 **policy-templates** 部分中的 **remediationAction** 会在运行时被设置为 **enforce**。
spec.copyPolicyMetadata	选填	指定在将策略复制到受管集群时是否应复制策略的标签和注解。如果设置为 **true**，策略的所有标签和注解都会复制到复制策略中。如果设置为 **false**，则只有策略框架特定的策略标签和注解被复制到复制策略中。
spec.dependencies	选填	用于创建与满足合规性的额外注意事项相关的依赖关系对象列表。
spec.policy-templates	必填	用于创建一个或多个应用到受管集群的策略。
spec.policy-templates.extraDependencies	选填	对于策略模板，这用于创建依赖项对象列表，以及满足合规性的额外注意事项。
spec.policy-templates.ignorePending	选填	用于在验证依赖项条件前将策略模板标记为合规。 **重要：**有些策略类型可能不支持 enforce 功能。

2.2.4. 策略示例文件

查看以下 **YAML 文件，它是角色的配置策略：**

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

```
    policy.open-cluster-management.io/standards: NIST SP 800-53
    policy.open-cluster-management.io/categories: AC Access Control
    policy.open-cluster-management.io/controls: AC-3 Access Enforcement
    policy.open-cluster-management.io/description:
spec:
  remediationAction: inform
  disabled: false
  policy-templates:
    - objectDefinition:
        apiVersion: policy.open-cluster-management.io/v1
        kind: ConfigurationPolicy
        metadata:
          name: policy-role-example
        spec:
          remediationAction: inform # the policy-template spec.remediationAction is overridden
by the preceding parameter value for spec.remediationAction.
          severity: high
          namespaceSelector:
            include: ["default"]
          object-templates:
            - complianceType: mustonlyhave # role definition should exact match
              objectDefinition:
                apiVersion: rbac.authorization.k8s.io/v1
                kind: Role
                metadata:
                  name: sample-role
                rules:
                  - apiGroups: ["extensions", "apps"]
                    resources: ["deployments"]
                    verbs: ["get", "list", "watch", "delete","patch"]
---
apiVersion: policy.open-cluster-management.io/v1
kind: PlacementBinding
metadata:
  name: binding-policy-role
placementRef:
  name: placement-policy-role
  kind: Placement
  apiGroup: cluster.open-cluster-management.io
subjects:
- name: policy-role
  kind: Policy
  apiGroup: policy.open-cluster-management.io
---
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement-policy-role
spec:
  predicates:
  - requiredClusterSelector:
      labelSelector:
        matchExpressions:
        - {key: environment, operator: In, values: ["dev"]}
```

2.2.5. 其他资源

- 请参阅策略控制器。
- 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。
- 返回 监管 文档。

2.3. 策略控制器简介

策略控制器监控并报告集群是否合规。使用支持的策略模板应用由这些控制器管理的策略，使用 Red Hat Advanced Cluster Management for Kubernetes 策略框架。策略控制器管理 Kubernetes 自定义资源定义实例。

策略控制器检查策略违反情况，如果控制器支持强制功能，可以使集群状态兼容。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（已弃用）
- 策略控制器
- Operator 策略控制器（技术预览）

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。

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。配置策略在 hub 集群上的策略的 policy-templates 字段中提供，并由监管框架传播到所选受管集群。

在配置策略中的 object-templates 数组中定义了 Kubernetes 对象（完整或部分），指示字段的配置策略控制器与受管集群上的对象进行比较。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。

配置策略控制器是在安装过程中在受管集群上创建的。配置策略控制器支持 enforce 和 InformOnly 功能，以便在配置策略不合规时修复。

当将配置策略的 remediationAction 设置为 enforce 时，控制器会将指定的配置应用到目标受管集群。

当配置策略的 remediationAction 设置为 InformOnly 时，父策略不会强制执行配置策略，即使父策略中的 remediationAction 被设置为 enforce。

注： 指定没有名称的对象的配置策略只能是 inform。

您还可以在配置策略中使用模板的值。如需更多信息，请参阅*模板处理*。

如果您想将现有的 Kubernetes 清单放入到一个策略，则策略生成器是完成此任务的有用工具。

2.3.1.1. 配置策略示例

```
apiVersion: policy.open-cluster-management.io/v1
kind: ConfigurationPolicy
metadata:
  name: policy-config
spec:
  namespaceSelector:
    include: ["default"]
    exclude: []
    matchExpressions: []
    matchLabels: {}
  remediationAction: inform
  severity: low
  evaluationInterval:
    compliant:
    noncompliant:
  object-templates:
```

```
  - complianceType: musthave
    objectDefinition:
      apiVersion: v1
      kind: Pod
      metadata:
        name: pod
      spec:
        containers:
        - image: pod-image
          name: pod-name
          ports:
          - containerPort: 80
  - complianceType: musthave
    objectDefinition:
    apiVersion: v1
    kind: ConfigMap
    metadata:
      name: myconfig
      namespace: default
      data:
      testData: hello
    spec:
...
```

2.3.1.2. 配置策略 YAML 标

表 2.2. 参数表

字段	可选或必需的	描述
apiVersion	必填	将值设置为 **policy.open-cluster-management.io/v1**。
kind	必填	将值设为 **ConfigurationPolicy** 以表示策略类型。
metadata.name	必填	策略的名称。
spec.namespaceSelector	没有指定命名空间的命名空间对象是必需的	决定对象要应用到的受管集群中的命名空间。**include** 和 **exclude** 参数接受根据名称包含和排除命名空间的文件路径表达式。**matchExpressions** 和 **matchLabels** 参数指定要包含的命名空间。请参阅 Kubernetes 标签和选择器文档。生成的列表通过使用来自所有参数的结果的交集进行编译。
spec.remediationAction	必填	指定当策略不合规时要执行的操作。使用以下参数值：**inform**、**InformOnly** 或 **enforce**。

字段	可选或必需的	描述
spec.severity	必填	当策略不合规时，指定严重性。使用以下参数值：**low**, **medium**, **high**, 或 **critical**。
spec.evaluationInterval.compliant	选填	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**。 默认情况下，当 **evaluationInterval.compliant** is set 或 empty 时，配置策略评估之间的最短时间大约为 10 秒。如果配置策略控制器在受管集群中饱和，则这可能会更长。
spec.evaluationInterval.noncompliant	选填	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**。
spec.object-templates	选填	用于控制器的 Kubernetes 对象数组（已定义或包含字段子集），以便与受管集群上的对象进行比较。**注：** 虽然 **spec.object-templates** 和 **spec.object-templates-raw** 列为可选，但必须设置两个参数字段中的一个。

字段	可选或必需的	描述
spec.object-templates-raw	选填	用于使用原始 YAML 字符串设置对象模板。指定对象模板的条件，其中支持 if-else 语句和 **range** 函数等高级功能。例如，添加以下值以避免在 **object-templates** 定义中出现重复： **{{- if eq .metadata.name "policy-grc-your-meta-data-name" }} replicas: 2 {{- else }} replicas: 1 {{- end }}** **注：** 虽然 **spec.object-templates** 和 **spec.object-templates-raw** 列为可选，但必须设置两个参数字段中的一个。
spec.object-templates[].complianceType	必填	在受管集群中定义 Kubernetes 对象的所需状态。您必须使用以下动词之一作为参数值： **mustonlyhave** ：代表必须存在带有在 **objectDefinition** 中定义的特定字段和值。 **musthave** ：代表必须存在名称与 **objectDefinition** 中指定的相同字段的对象。在 **object-template** 中指定的任何现有字段都会被忽略。通常，会附加数组值。要修补数组的一个例外是，当项目包含一个与现有项匹配的 **name** 值时。如果要替换数组，请使用 **mustonlyhave** 合规类型来完全定义 **objectDefinition**。 **mustnothave** ：代表一个具有与 **objectDefinition** 中指定的字段相同的对象不能存在。
spec.object-templates[].metadataComplianceType	选填	当将清单的 metadata 部分与集群中的对象比较时覆盖 **spec.object-templates[].complianceType** ("musthave", "mustonlyhave")。默认没有设置，这不会为原始覆盖 **complianceType**。

字段	可选或必需的	描述
spec.object-templates[].recordDiff	选填	指定在策略中记录对象与 **objectDefinition** 之间的区别。设置为 **Log** 以记录控制器日志中的区别或 **None** 来记录区别。默认情况下，此参数为空，不记录区别。
spec.object-templates[].objectDefinition	必填	用于控制器的 Kubernetes 对象数组（完整定义或包含字段子集），以便与受管集群上的对象进行比较。
spec.pruneObjectBehavior	选填	决定在从受管集群中删除策略时是否清理与策略相关的资源。

2.3.1.3. 其他资源

如需更多信息，请参阅以下主题：

- 有关 hub 集群策略的详情，请参阅策略概述。
- 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。
- 有关控制器的详情，请参阅策略控制器。
- 自定义策略控制器配置。请参阅策略控制器高级配置。
- 参阅清理由策略创建的资源文档中的有关清理资源及其他相关的内容。
- 请参阅策略生成器。

- 了解如何创建和自定义策略，请参阅管理 监管仪表板。
- 请参阅模板处理。

2.3.2. 证书策略控制器

您可以使用证书策略控制器来检测即将过期的证书、持续时间（小时）或包含无法与指定模式匹配的 DNS 名称。您可以将证书策略添加到 hub 集群上的策略的 policy-templates 字段中，它使用监管框架传播到所选受管集群。有关 hub 集群策略的详情，请参阅策略概述文档。

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。

证书策略控制器不支持 enforce 功能。

注： 证书策略控制器只自动在 tls.crt 密钥中的 secret 中查找证书。如果 secret 存储在不同的密钥下，请添加名为 certificate_key_name 的标签，并将值设为键，以便证书策略控制器知道查看不同的密钥。例如，如果 secret 包含存储在名为 sensor-cert.pem 的键中的证书，请将以下标签添加到 secret: certificate_key_name: sensor-cert.pem。

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

2.3.2.1.1. 证书策略控制器 YAML 表

表 2.3. 参数表

字段	可选或必需的	描述
apiVersion	必填	将值设置为 policy.open-cluster-management.io/v1。
kind	必填	将值设为 CertificatePolicy 以表示策略类型。

字段	可选或必需的	描述
metadata.name	必填	用于标识策略的名称。
metadata.labels	选填	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.namespaceSelector	必填	决定监控 secret 的受管集群中的命名空间。**include** 和 **exclude** 参数接受根据名称包含和排除命名空间的文件路径表达式。**matchExpressions** 和 **matchLabels** 参数指定要包含在标签中的命名空间。请参阅 Kubernetes 标签和选择器文档。生成的列表通过使用来自所有参数的结果的交集进行编译。 **注:** 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.labelSelector	选填	指定识别对象属性。请参阅 Kubernetes 标签和选择器文档。
spec.remediationAction	必填	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	选填	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**, **high**, 或 **critical**。
spec.minimumDuration	必填	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	选填	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。

字段	可选或必需的	描述
spec.maximumDuration	选填	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	选填	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	选填	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。
spec.disallowedSANPattern	选填	正则表达式，不能与证书中定义的任何 SAN 条目匹配。这个参数会根据特征检查 DNS 名称。 **注**：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。

2.3.2.3. 其他资源

- **请参阅管理安全策略以了解更多详细信息。**
- **如需了解更多主题，请参阅策略控制器简介。**
- **返回 证书简介。**

2.3.3. IAM 策略控制器（已弃用）

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 策略在 hub 集群上的策略的 policy-templates 字段中提供，并由监管框架传播到所选受管集群。有关 hub 集群策略的详情，请参阅策略 YAML 结构文档。

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。

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 2.4. 参数表

字段	可选或必需的	描述
apiVersion	必填	将值设置为 **policy.open-cluster-management.io/v1**。
kind	必填	将值设为 **Policy** 以表示策略类型。
metadata.name	必填	用于标识策略资源的名称。
spec.clusterRole	选填	要监控的集群角色（如 **ClusterRole**）。如果没有指定，则默认为 **cluster-admin**。
spec.severity	选填	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**, **high**, 或 **critical**。
spec.remediationAction	选填	指定您的策略的修复。输入 **inform**。IAM 策略控制器只支持 **inform** 功能。
spec.ignoreClusterRoleBindings	选填	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	必填	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。

备注：

- Red Hat Advanced Cluster Management 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。

- 要使用 放置资源，ManagedClusterSet 资源必须绑定到带有 ManagedClusterSetBinding 资源的 Placement 资源的命名空间。如需了解更多详细信息，请参阅创建 *ManagedClusterSetBinding* 资源。

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

2.3.4.2. 策略设置表

查看以下参数表以获详细信息：

表 2.5. 参数表

字段	可选或必需的	描述
apiVersion	必填	将值设为 **policy.open-cluster-management.io/v1beta1**。
kind	必填	将值设为 **PolicySet** 以表示策略类型。
metadata.name	必填	用于标识策略资源的名称。
spec	必填	添加策略的配置详情。
spec.policies	选填	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
    placement: placement1
    policySet: policyset-ps
```

2.3.4.4. 其他资源

- 请参阅 Red Hat OpenShift Platform Plus 策略集。
- 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 Policy Generator 用于部署，PolicySets-Stable。
- 返回到此主题的开头，策略设置控制器。

2.3.5. Operator 策略控制器（技术预览）

Operator 策略控制器允许您监控和安装集群中的 Operator Lifecycle Manager (OLM) Operator。使用 Operator 策略控制器来监控 Operator 的各种部分的健康状况，并指定如何自动处理 Operator 更新。您还可以使用监管框架将 operator 策略分发到受管集群，并将策略添加到 hub 集群上的策略的 policy-templates 字段中。

2.3.5.1. 先决条件

- OLM 必须在受管集群中可用。这在 Red Hat OpenShift Container Platform 上默认启用。
- 需要的访问权限：集群管理员

2.3.5.2. Operator 策略 YAML 表

字段	可选或必需的	描述
apiVersion	必填	将值设为 **policy.open-cluster-management.io/v1beta1**。
kind	必填	将值设为 **OperatorPolicy** 以表示策略类型。
metadata.name	必填	用于标识策略资源的名称。
spec.remediationAction	必填	如果将 operator 策略的 **remediationAction** 设置为 **enforce**，则控制器会在目标受管集群上创建资源，以与 OLM 通信以根据策略中指定的版本安装 Operator 并根据策略中指定的版本批准更新。+ 如果 **remediationAction** 设置为 **inform**，控制器只报告 Operator 的状态，包括任何升级可用。
spec.operatorGroup	选填	默认情况下，如果没有指定 **operatorGroup** 字段，控制器会在与订阅相同的命名空间中生成一个 **AllNamespaces** 类型 OperatorGroup。此资源由 Operator 策略控制器生成。
spec.subscription	必填	定义用于创建 operator 订阅的配置。您必须在以下字段中添加信息来创建 operator 订阅： • **channel** • **name** • **namespace** • **source** • **sourceNamespace**
subscriptions.installPlanApproval	必填	如果 **installPlanApproval** 字段被设置为 **Manual**，并且 **spec.versions** 字段为空，则必须在相关 **InstallPlan** 资源中手动修补 **.spec.approved** 字段才能继续安装过程。Operator 的 **InstallPlan** 资源是在控制器创建订阅后生成的，表示安装所需 Operator 的特定版本的意图。

字段	可选或必需的	描述
spec.versions	选填	声明 Operator 的兼容版本。如果字段为空，则集群中运行的任何版本都被视为合规。如果字段不为空，则受管集群上的版本必须与要合规的策略列表中的一个版本匹配。如果策略被设置为 **enforce** 且列表不为空，则集群中控制器批准此处列出的版本。

2.3.5.3. 其他资源

- 如需了解更多详细信息，请参阅使用 OperatorPolicy 资源 安装 Operator。
- 如需了解更多详细信息，请参阅 Operator Lifecycle Manager (OLM)。
- 有关 OLM 的常规信息，请参阅将 Operator 添加到集群 文档。

2.4. 策略控制器高级配置

您可以使用 ManagedClusterAddOn 自定义资源自定义受管集群上的策略控制器配置。以下 ManagedClusterAddOns 配置策略框架、Kubernetes 配置策略控制器、证书策略控制器和 IAM 策略控制器。

需要的访问权限：集群管理员

- 配置监管框架的并发性
- 配置配置策略控制器的并发性
- 配置对 API 服务器的请求率
- 配置调试日志

- 监管指标
- 验证配置更改

2.4.1. 配置监管框架的并发性

为每个受管集群配置监管框架的并发性。要更改默认值 2，请在 quotation 标记中使用非零整数设置 policy-evaluation-concurrency 注解。然后，在 hub 集群的受管集群命名空间中将 ManagedClusterAddOn 对象名称设置为 governance-policy-framework。

请参阅以下 YAML 示例，其中 concurrency 设置为名为 cluster1 的受管集群：

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

要设置 client-qps 和 client-burst 注解，请更新 ManagedClusterAddOn 资源并定义参数。

请参阅以下 YAML 示例，其中每秒查询被设置为 30，在名为 cluster1 的受管集群中将 burst 设置为 45：

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

2.4.2. 配置配置策略控制器的并发性

您可以为每个受管集群配置配置策略控制器的并发性，以更改它可以同时评估的配置策略。要更改默

认值 2，请在 quotation 标记中使用非零整数设置 policy-evaluation-concurrency 注解。然后，在 hub 集群的受管集群命名空间中将 ManagedClusterAddOn 对象名称的值设置为 config-policy-controller。

注：增加并发值会增加 config-policy-controller pod、Kubernetes API 服务器和 OpenShift API 服务器上的 CPU 和内存使用率。

请参阅以下 YAML 示例，其中名为 cluster1 的受管集群中并发设置为 5 ：

```
apiVersion: addon.open-cluster-management.io/v1alpha1
kind: ManagedClusterAddOn
metadata:
  name: config-policy-controller
  namespace: cluster1
  annotations:
    policy-evaluation-concurrency: "5"
spec:
  installNamespace: open-cluster-management-agent-addon
```

2.4.3. 配置对 API 服务器的请求率

配置配置策略控制器在每个受管集群上进行的 API 服务器的请求率。速率提高了配置策略控制器的响应，这也会增加 Kubernetes API 服务器和 OpenShift API 服务器的 CPU 和内存使用率。默认情况下，请求的速度会根据 policy-evaluation-concurrency 设置进行扩展，并被设置为每秒 30 个查询(QPS)，带有 45 burst 值，代表短时间内的请求数。

您可以通过在引号内将 client-qps 和 client-burst 注解设置为非零整数来配置速率和突发。您可以在 hub 集群的受管集群命名空间中将 ManagedClusterAddOn 对象名称上的值设置为 config-policy-controller。

请参阅以下 YAML 示例，其中每秒查询被设置为 20，在名为 cluster1 的受管集群中将 burst 设置为 100 ：

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

2.4.4. 配置调试日志

当您为每个策略控制器配置和收集调试日志时，您可以调整日志级别。

注： 缩减调试日志的卷意味着日志中显示较少的信息。

您可以减少策略控制器提供的调试日志，以便在日志中显示仅限错误的错误。要减少 debug 日志，请在注解中将 debug 日志值设置为 -1。查看每个值代表什么：

- -1: error logs only
- 0 ：信息性日志
- 1： 调试日志
- 2: 详细调试日志

要接收 Kubernetes 配置控制器的第二个调试信息级别，请将值为 2 的 log-level 注解添加到 ManagedClusterAddOn 自定义资源中。默认情况下，日志级别被设置为 0，这意味着您会收到信息性的消息。查看以下示例：

```
apiVersion: addon.open-cluster-management.io/v1alpha1
kind: ManagedClusterAddOn
metadata:
  name: config-policy-controller
  namespace: cluster1
  annotations:
    log-level: "2"
spec:
  installNamespace: open-cluster-management-agent-addon
```

另外，对于 ConfigurationPolicy 资源中的每个 spec.object-template[]，您可以将 recordDiff 参数设置为 Log。objectDefinition 和受管集群上的对象之间的区别记录在受管集群上的 config-policy-controller pod 中。查看以下示例：

此带有 recordDiff 的 ConfigurationPolicy 资源：Log:

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

如果集群列表上的 ConfigMap 资源 ToUpdate: "1"，则 diff 会出现在 config-policy-controller pod 日志中，其中包含以下信息：

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

重要： 避免记录安全对象的区别。差别以纯文本记录。

2.4.5. 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。有关指标概述，请参阅以下主题：

2.4.5.1. Metric: policy_governance_info

OpenShift Container Platform 监控组件收集 policy_governance_info 指标。如果启用可观察性，则组件会收集一些聚合数据。

注： 如果启用可观察性，请从 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、放置 资源和 PlacementBindings 资源，以获取有关创建 *传播* 策略的信息，以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 集群上的命名空间。
- 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： 如果您不需要指标，并且对性能或安全性有任何关注，您可以禁用指标集合。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。

2.4.5.2. Metric: *config_policies_evaluation_duration_seconds*

config_policies_evaluation_duration_seconds 直方图跟踪集群中准备好评估的所有配置策略所需的秒数。使用以下指标查询直方图：

- config_policies_evaluation_duration_seconds_bucket ：存储桶是累积的，以秒为单位的以下可能的值：1, 3, 9, 10.5, 15, 30, 60, 90, 90, 120, 180, 300, 450, 600 等。
- config_policies_evaluation_duration_seconds_count: 所有事件的计数。
- config_policies_evaluation_duration_seconds_sum: 所有值的总和。

使用 config_policies_evaluation_duration_seconds 指标来确定 ConfigurationPolicy evaluationInterval 设置是否需要为不需要频繁评估的资源密集型策略更改。您还可以以 Kubernetes

API 服务器的资源使用率更高的成本增加并发性。如需了解更多详细信息，请参阅*配置并发* 部分。

要接收有关评估配置策略的时间的信息，请执行类似以下表达式的 Prometheus 查询：

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

open-cluster-management-agent-addon 命名空间中的受管集群上运行的 config-policy-controller pod 会计算指标。config-policy-controller 默认不会将指标发送到可观察性。

2.4.6. 验证配置更改

当使用控制器应用新配置时，在 ManagedClusterAddOn 中会更新 ManifestApplied 参数。该条件时间戳有助于正确验证配置。例如，这个命令可在 local-cluster 中的 cert-policy-controller 已更新时验证：

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

您可能会收到以下输出：

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

2.4.7. 其他资源

- 请参阅 Kubernetes 配置策略控制器
- 返回监管主题以获取更多主题。
- 返回到此主题的开头，策略控制器高级配置。

2.5. 策略合规历史记录（技术预览）

如果您希望以可查询的格式进行 Red Hat Advanced Cluster Management for Kubernetes 策略合规事件的长期存储，策略合规历史记录 API 是一个可选的技术预览功能。您可以使用 API 获取其他详情，如 spec 字段审核并排除您的策略，并在策略从集群中删除或从集群中删除时获取合规事件。策略合规历史记录 API 也可以生成以逗号分隔的策略合规事件表，以帮助您进行审计和故障排除。

策略合规历史记录 API 也可以生成以逗号分隔的值(CSV)电子表格，以进行进一步审核和故障排除。

2.5.1. 先决条件

- 策略合规历史记录 API 需要版本 13 或更高版本的 PostgreSQL 服务器。

  有些红帽支持的选项包括使用 registry.redhat.io/rhel9/postgresql-15 容器镜像、registry.redhat.io/rhel8/postgresql-13 容器镜像、postgresql-server RPM 或 postgresql/server 模块。有关设置和配置您选择的路径，请查看相关的官方红帽文档。策略合规历史记录 API 与任何标准 PostgreSQL 兼容，不仅限于官方红帽支持的产品。

- 此 PostgreSQL 服务器必须可从 Red Hat Advanced Cluster Management hub 集群访问。如果 PostgreSQL 服务器在 hub 集群外部运行，请确保路由和防火墙配置允许 hub 集群连接到 PostgreSQL 服务器的端口 5432。如果在 PostgreSQL 配置中被覆盖，此端口可能是不同的值。

2.5.2. 启用合规性历史记录 API

配置受管集群，将策略合规事件记录到 API。您可以在所有集群或集群子集中启用它。完成以下步骤：

1. 将 PostgreSQL 服务器配置为集群管理员。如果您在 Red Hat Advanced Cluster Management hub 集群中部署了 PostgreSQL，则临时端口转发 PostgreSQL 端口以使用 psql 命令。运行以下命令：

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

2. 在不同的终端中，在本地连接到 PostgreSQL 服务器，使用以下命令：

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

3. 使用以下 SQL 语句为您的 Red Hat Advanced Cluster Management hub 集群创建用户和数据库：

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

4. 创建 governance-policy-database Secret 资源，将此数据库用于策略合规历史记录 API。运行以下命令：

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

1 添加安装 Red Hat Advanced Cluster Management 的命名空间。默认情况下，Red Hat Advanced Cluster Management 安装在 open-cluster-management 命名空间中。

2 添加 PostgresQL 服务器的主机名。如果您在 Red Hat Advanced Cluster Management hub 集群中部署了 PostgreSQL 服务器，且没有在集群外公开，您可以将 Service 对象用于主机值。格式为 < service name>.<namespace>.svc。请注意，此方法取决于 Red Hat Advanced Cluster Management hub 集群的网络策略。

3 您必须在为 PostgreSQL 服务器的 TLS 证书签名的 ca data 字段中指定证书颁发机构证书文件。如果没有提供这个值，您必须相应地更改 *sslmode* 值，但不推荐这样做，因为它降低了数据库连接的安全性。

5. 添加 cluster.open-cluster-management.io/backup 标签，以备份 Red Hat Advanced Cluster Management hub 集群恢复操作的 Secret 资源。运行以下命令：

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

6. 对于 PostgreSQL 连接的更多自定义，请直接使用 connectionURL data 字段，并以 PostgreSQL 连接 URI 格式提供值。密码中的特殊字符必须采用 URL 编码。一个选项是使用 Python 来生成密码的 URL 编码格式。例如，如果密码是 $uper<Secr&t% >，请运行以下 Python 命令获取输出 %24uper%3CSecr%26t%25%3E ：

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

7. 在创建 governance-policy-database Secret 后，运行 命令以测试策略合规历史记录 API。OpenShift Route 对象会自动在同一命名空间中创建。如果 Red Hat Advanced Cluster Management hub 集群上的路由没有使用可信证书，您可以选择在 curl 命令中提供 -k 标志来跳过 TLS 验证，但不建议这样做：

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

- 如果成功，curl 命令会返回类似如下的值：

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

- 如果它不成功，则 curl 命令可能会返回这两个消息之一：

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

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

a. 如果您收到消息，使用以下命令查看 open-cluster-management 命名空间中的 Kubernetes 事件：

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

b. 如果您从事件收到说明来查看 governance-policy-propagator 日志，请运行以下命令：

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

c. 您可能会收到一条错误消息，指示用户、密码或数据库被错误指定。请参见以下消息示例：

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

d. 使用以下命令更新 governance-policy-database Secret 资源，使其包含正确的 PostgreSQL 连接设置：

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

2.5.3. 设置合规性历史记录 API URL

设置策略合规历史记录 API URL，以在受管集群上启用该功能。完成以下步骤：

1. 使用以下命令检索策略合规历史记录 API 的外部 URL：

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

输出可能类似以下信息，以及 Red Hat Advanced Cluster Management hub 集群的域名：

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

2. 创建一个类似以下示例的 AddOnDeploymentConfig 对象：

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

- 将 value 参数值替换为您的合规性历史记录外部 URL。

2.5.3.1. 在所有受管集群中启用

在所有受管集群中启用合规历史记录 API，以记录来自受管集群的合规事件。完成以下步骤：

1. 将 governance-policy-framework ClusterManagementAddOn 对象配置为使用以下命令使用 AddOnDeploymentConfig ：

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

2. 添加或更新 spec.supportedConfigs 数组。您的资源可能有以下配置：

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

2.5.3.2. 启用单个受管集群

在单一受管集群中启用合规历史记录 API，以记录来自受管集群的合规事件。完成以下步骤：

1. 在受管集群命名空间中配置 governance-policy-framework ManagedClusterAddOn 资源。使用以下命令，从 Red Hat Advanced Cluster Management hub 集群中运行以下命令：

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

- 将 & lt;manage-cluster-namespace > 占位符替换为您要启用的受管集群名称。

2. 添加或更新 spec.configs 数组，使其具有类似以下示例的条目：

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

3. 要验证配置，请确认受管集群上的部署是否使用 --compliance-api-url 容器参数。运行以下命令：

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

输出可能类似以下：

```
["--enable-lease=true","--hub-cluster-configfile=/var/run/klusterlet/kubeconfig","--leader-elect=false","--log-encoder=console","--log-level=0","--v=-1","--evaluation-concurrency=2","--client-max-qps=30","--client-burst=45","--disable-spec-sync=true","--cluster-
```

```
namespace=local-cluster","--compliance-api-url=https://governance-history-api-open-cluster-management.apps.openshift.redhat.com"]
```

任何新的策略合规事件都会记录在策略合规历史记录 API 中。

a. 如果没有为特定受管集群记录策略合规事件，请查看受影响受管集群中的 governance-policy-framework 日志：

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

b. 此时会显示类似以下消息的日志消息。如果 消息 值为空，策略合规历史记录 API URL 不正确，或者存在网络通信问题：

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

c. 如果策略合规历史记录 API URL 不正确，使用以下命令编辑 hub 集群上的 URL：

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

注： 如果遇到网络通信问题，您必须根据网络基础架构诊断问题。

2.5.4. 其他资源

- 请参阅策略合规历史记录 API （技术预览）。

2.6. 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。

2.6.1. 配置策略示例策略表

查看以下示例配置策略：

表 2.6. 配置策略表列表

策略示例	描述
命名空间策略	确保环境与命名空间一致。请参阅 Kubernetes 命名空间文档。
Pod 策略	确保集群工作负载配置。请参阅 Kubernetes Pod 文档。
内存用量策略	使用限制范围限制工作负载资源使用情况。请参阅限制范围文档。
Pod 安全策略（已弃用）	确保工作负载安全性一致。请参阅 Kubernetes Pod 安全策略文档。
角色策略 角色绑定策略	使用角色和角色绑定管理角色绑定。请参阅 Kubernetes RBAC 文档。
安全内容约束 (SCC) 策略	使用安全性上下文约束管理工作负载权限。请参阅 OpenShift Container Platform 文档中的管理安全性上下文约束文档。
ETCD 加密策略	确保通过 etcd 加密的数据安全性。请参阅 OpenShift Container Platform 文档中的加密 etcd 数据。
Compliance operator 策略	部署 Compliance Operator，以利用 OpenSCAP 扫描并强制实施集群的合规性状态。请参阅 OpenShift Container Platform 文档中的了解 Compliance Operator 部分。
Compliance operator E8 扫描	应用 Compliance operator 策略后，部署 Essential 8 (E8) 扫描来检查 E8 安全配置集的合规性。请参阅 OpenShift Container Platform 文档中的了解 Compliance Operator 部分。
Compliance operator CIS 扫描	应用 Compliance operator 策略后，部署互联网安全中心 (CIS) 扫描，以检查与 CIS 安全配置集的合规性。请参阅 OpenShift Container Platform 文档中的了解 Compliance Operator 部分。
镜像漏洞策略	部署 Container Security Operator，并检测集群中运行的 pod 中的已知镜像漏洞。请参阅 Container Security Operator GitHub 仓库。
Gatekeeper operator 部署	Gatekeeper 是一个准入 Webhook，它强制执行基于自定义资源定义的策略，由 Open Policy Agent (OPA)策略引擎运行。请参阅 Gatekeeper 文档。

策略示例	描述
Gatekeeper 合规策略	将 Gatekeeper 部署到集群后，部署此示例 Gatekeeper 策略以确保在集群中创建的命名空间标记为指定。
Red Hat OpenShift Platform Plus 策略集	Red Hat OpenShift Platform Plus 是混合云套件，可安全地为多个基础架构构建、部署、运行和管理应用程序。您可以使用通过 Red Hat Advanced Cluster Management 应用程序提供的 **PolicySets** 将 Red Hat OpenShift Platform Plus 部署到受管集群。有关 OpenShift Platform Plus 的详情，请查看 OpenShift Platform Plus 文档。

Red Hat OpenShift Container Platform 4.x 还支持 Red Hat Advanced Cluster Management 配置策略。

查看以下策略文档以了解如何应用策略：

- **命名空间策略**
- **Pod 策略**
- **内存用量策略**
- **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
- 镜像漏洞策略
- Red Hat OpenShift Platform Plus 策略集

更多主题，请参阅监管。

2.6.2. 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
- 命名空间策略表
- 命名空间策略示例

2.6.2.1. 命名空间策略 YAML 结构

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

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

2.6.2.2. 命名空间策略 YAML 表

字段	可选或必需的	描述
apiVersion	必填	将值设置为 **policy.open-cluster-management.io/v1**。
kind	必填	将值设为 **Policy** 以表示策略类型。
metadata.name	必填	用于标识策略资源的名称。
metadata.namespace	必填	策略的命名空间。
spec.remediationAction	选填	指定您的策略的修复。参数值是 **enforce** 和 **inform**。这个值是可选的，因为它会覆盖 **spec.policy-templates** 中提供的任何值。
spec.disabled	必填	将值设为 **true** 或 **false**。**disabled** 参数提供启用和禁用策略的功能。
spec.policy-templates[].objectDefinition	必填	用于列出包含必须接受评估或应用到受管集群的 Kubernetes 对象的配置策略。

2.6.2.3. 命名空间策略示例

请参阅 policy-namespace.yaml 以查看策略示例。

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

2.6.3. 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

2.6.3.1. Pod 策略 YAML 结构

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

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

2.6.3.2. Pod 策略表

表 2.7. 参数表

字段	可选或必需的	描述
apiVersion	必填	将值设置为 **policy.open-cluster-management.io/v1**。
kind	必填	将值设为 **Policy** 以表示策略类型。
metadata.name	必填	用于标识策略资源的名称。
metadata.namespace	必填	策略的命名空间。
spec.remediationAction	选填	指定您的策略的修复。参数值是 **enforce** 和 **inform**。此值是可选的，因为该值会覆盖 **spec.policy-templates** 中提供的任何值。
spec.disabled	必填	将值设为 **true** 或 **false**。**disabled** 参数提供启用和禁用策略的功能。
spec.policy-templates[].objectDefinition	必填	用于列出包含必须接受评估或应用到受管集群的 Kubernetes 对象的配置策略。

2.6.3.3. 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，并查看 Policy 概述文档，以查看策略 YAML 结构和其他字段的完整描述。返回到管理配置策略文档，以管理其他策略。

2.6.4. 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

2.6.4.1. 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:
  namespace:
  annotations:
    policy.open-cluster-management.io/standards:
    policy.open-cluster-management.io/categories:
    policy.open-cluster-management.io/controls:
    policy.open-cluster-management.io/description:
spec:
  remediationAction:
  disabled:
  policy-templates:
    - objectDefinition:
```

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

2.6.4.2. 内存用量策略表

表 2.8. 参数表

字段	可选或必需的	描述
apiVersion	必填	将值设置为 **policy.open-cluster-management.io/v1**。
kind	必填	将值设为 **Policy** 以表示策略类型。
metadata.name	必填	用于标识策略资源的名称。
metadata.namespace	必填	策略的命名空间。
spec.remediationAction	选填	指定您的策略的修复。参数值是 **enforce** 和 **inform**。此值是可选的，因为该值会覆盖 **spec.policy-templates** 中提供的任何值。

字段	可选或必需的	描述
spec.disabled	必填	将值设为 **true** 或 **false**。**disabled** 参数提供启用和禁用策略的功能。
spec.policy-templates[].objectDefinition	必填	用于列出包含必须接受评估或应用到受管集群的 Kubernetes 对象的配置策略。

2.6.4.3. 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。请参阅策略概述文档，以及 Kubernetes 配置策略控制器，以查看控制器监控的其他配置策略。

2.6.5. Pod 安全策略（已弃用）

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。

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

2.6.5.1. Pod 安全策略 YAML 结构

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

```
spec:
  remediationAction:
  disabled:
  policy-templates:
    - objectDefinition:
        apiVersion: policy.open-cluster-management.io/v1
        kind: ConfigurationPolicy
        metadata:
          name:
        spec:
          remediationAction:
          severity:
          namespaceSelector:
            exclude:
            include:
            matchLabels:
            matchExpressions:
          object-templates:
            - complianceType:
              objectDefinition:
                apiVersion: policy/v1beta1
                kind: PodSecurityPolicy
                metadata:
                  name:
                  annotations:
                    seccomp.security.alpha.kubernetes.io/allowedProfileNames:
                spec:
                  privileged:
                  allowPrivilegeEscalation:
                  allowedCapabilities:
                  volumes:
                  hostNetwork:
                  hostPorts:
                  hostIPC:
                  hostPID:
                  runAsUser:
                  seLinux:
                  supplementalGroups:
                  fsGroup:
                ...
```

2.6.5.2. Pod 安全策略表

表 2.9. 参数表

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

字段	可选或必需的	描述
metadata.namespace	必填	策略的命名空间。
spec.remediationAction	选填	指定您的策略的修复。参数值是 **enforce** 和 **inform**。此值是可选的，因为该值会覆盖 **spec.policy-templates** 中提供的任何值。
spec.disabled	必填	将值设为 **true** 或 **false**。**disabled** 参数提供启用和禁用策略的功能。
spec.policy-templates[].objectDefinition	必填	用于列出包含必须接受评估或应用到受管集群的 Kubernetes 对象的配置策略。

2.6.5.3. Pod 安全策略示例

对 **Pod 安全策略的支持已从 OpenShift Container Platform 4.13 及更新的版本中删除，并从 Kubernetes v1.25 及之后的版本中删除。如果应用 PodSecurityPolicy 资源，您可能会收到以下不合规的信息：**

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

- **有关包括弃用通知的更多信息，请参阅 Kubernetes 文档中的 *Pod 安全策略*。**
- **请参阅 policy-psp.yaml 以查看示例策略。如需更多信息，请参阅管理配置策略。**
- **如需了解策略 YAML 结构的完整描述，请参阅策略概述文档，以及 Kubernetes 配置策略控制器，以查看控制器监控的其他配置策略。???**

2.6.6. 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

2.6.6.1. 角色策略 YAML 结构

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

```
            resources:
            verbs:
        ...
```

2.6.6.2. 角色策略表

表 2.10. 参数表

字段	可选或必需的	描述
apiVersion	必填	将值设置为 **policy.open-cluster-management.io/v1**。
kind	必填	将值设为 **Policy** 以表示策略类型。
metadata.name	必填	用于标识策略资源的名称。
metadata.namespace	必填	策略的命名空间。
spec.remediationAction	选填	指定您的策略的修复。参数值是 **enforce** 和 **inform**。此值是可选的，因为该值会覆盖 **spec.policy-templates** 中提供的任何值。
spec.disabled	必填	将值设为 **true** 或 **false**。**disabled** 参数提供启用和禁用策略的功能。
spec.policy-templates[].objectDefinition	必填	用于列出包含必须接受评估或应用到受管集群的 Kubernetes 对象的配置策略。

2.6.6.3. 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。

2.6.7. 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

2.6.7.1. 角色绑定策略 YAML 结构

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

```
    name:
    apiGroup:
  roleRef:
    kind:
    name:
    apiGroup:
  ...
```

2.6.7.2. 角色绑定策略表

字段	可选或必需的	描述
apiVersion	必填	将值设置为 **policy.open-cluster-management.io/v1**。
kind	必填	将值设为 **Policy** 以表示策略类型。
metadata.name	必填	用于标识策略资源的名称。
metadata.namespace	必填	策略的命名空间。
spec.remediationAction	选填	指定您的策略的修复。参数值是 **enforce** 和 **inform**。这个值是可选的，因为它会覆盖 **spec.policy-templates** 中提供的任何值。
spec.disabled	必填	将值设为 **true** 或 **false**。**disabled** 参数提供启用和禁用策略的功能。
spec.policy-templates[].objectDefinition	必填	用于列出包含必须接受评估或应用到受管集群的 Kubernetes 对象的配置策略。

2.6.7.3. 角色绑定策略示例

请参阅 policy-rolebinding.yaml 查看策略示例。有关策略 YAML 结构和其他字段的完整描述，请参阅策略概述文档。请参阅 Kubernetes 配置策略控制器 文档，以了解其他配置策略。

2.6.8. 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

2.6.8.1. SCC 策略 YAML 结构

```
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  name:
  namespace:
  annotations:
    policy.open-cluster-management.io/standards:
    policy.open-cluster-management.io/categories:
    policy.open-cluster-management.io/controls:
    policy.open-cluster-management.io/description:
spec:
  remediationAction:
  disabled:
  policy-templates:
    - objectDefinition:
        apiVersion: policy.open-cluster-management.io/v1
        kind: ConfigurationPolicy
        metadata:
          name:
        spec:
          remediationAction:
          severity:
          namespaceSelector:
            exclude:
            include:
            matchLabels:
            matchExpressions:
          object-templates:
            - complianceType:
              objectDefinition:
                apiVersion: security.openshift.io/v1
                kind: SecurityContextConstraints
                metadata:
                  name:
                allowHostDirVolumePlugin:
                allowHostIPC:
                allowHostNetwork:
                allowHostPID:
```

```
    allowHostPorts:
    allowPrivilegeEscalation:
    allowPrivilegedContainer:
    fsGroup:
    readOnlyRootFilesystem:
    requiredDropCapabilities:
    runAsUser:
    seLinuxContext:
    supplementalGroups:
    users:
    volumes:
    ...
```

2.6.8.2. SCC 策略表

字段	可选或必需的	描述
apiVersion	必填	将值设置为 **policy.open-cluster-management.io/v1**。
kind	必填	将值设为 **Policy** 以表示策略类型。
metadata.name	必填	用于标识策略资源的名称。
metadata.namespace	必填	策略的命名空间。
spec.remediationAction	选填	指定您的策略的修复。参数值是 **enforce** 和 **inform**。这个值是可选的，因为它会覆盖 **spec.policy-templates** 中提供的任何值。
spec.disabled	必填	将值设为 **true** 或 **false**。**disabled** 参数提供启用和禁用策略的功能。
spec.policy-templates[].objectDefinition	必填	用于列出包含必须接受评估或应用到受管集群的 Kubernetes 对象的配置策略。

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。

2.6.8.3. 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。有关策略 YAML 结构和其他字段的完整描述，请参阅策略概述文档。请参阅 Kubernetes 配置策略控制器 文档，以了解其他配置策略。

2.6.9. 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

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

```
    metadata:
      name:
    spec:
      remediationAction:
      severity:
      object-templates:
        - complianceType:
          objectDefinition:
            apiVersion: config.openshift.io/v1
            kind: APIServer
            metadata:
              name:
            spec:
              encryption:
            ...
```

2.6.9.2. ETCD 加密策略表

表 2.11. 参数表

字段	可选或必需的	描述
apiVersion	必填	将值设置为 **policy.open-cluster-management.io/v1**。
kind	必填	将值设为 **Policy** 以表示策略类型。
metadata.name	必填	用于标识策略资源的名称。
metadata.namespace	必填	策略的命名空间。
spec.remediationAction	选填	指定您的策略的修复。参数值是 **enforce** 和 **inform**。这个值是可选的，因为它会覆盖 **spec.policy-templates** 中提供的任何值。
spec.disabled	必填	将值设为 **true** 或 **false**。**disabled** 参数提供启用和禁用策略的功能。
spec.policy-templates[].objectDefinition	必填	用于列出包含必须接受评估或应用到受管集群的 Kubernetes 对象的配置策略。

2.6.9.3. ETCD 加密策略示例

如需策略示例，请参阅 **policy-etcdencryption.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。

2.6.10. Compliance Operator 策略

您可以使用 Compliance Operator 自动检查许多技术实施，并将其与行业标准、基准和基准的某些方面进行比较。Compliance Operator 不是一个审核员(auditor)。要符合这些各种标准，您需要与授权的审核员（如限定安全评估器(QSA)、联合授权局(JAB)或其他行业认可的规范机构）合作来评估您的环境。

来自 Compliance Operator 生成的建议基于有关此类标准的一般信息和实践，并可能帮助您进行补救，但实际的合规性是您的责任。与授权的审核员合作来实现符合标准的合规性。

有关最新更新，请参阅 Compliance Operator 发行注记。

2.6.10.1. Compliance Operator 策略概述

您可以使用 Compliance Operator 策略在受管集群上安装 Compliance Operator。Compliance operator 策略在 Red Hat Advanced Cluster Management 中作为 Kubernetes 配置策略创建。OpenShift Container Platform 支持 Compliance operator 策略。

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 部分。

2.6.10.2. 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容器，它支持。请参见以下示例，使用当前版本替换 4.x ：

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

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。

2.6.10.3. 其他资源

- 如需更多信息，请参阅 OpenShift Container Platform 文档中的管理 Compliance Operator 部分以了解更多详细信息。
- 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信息，请参阅 Kubernetes 配置策略控制器。

2.6.11. 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 支持 E8 扫描策略。如需更多信息，请参阅 OpenShift Container Platform 文档中的管理 Compliance Operator 部分以了解更多详细信息。

2.6.11.1. 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移除。

2.6.12. 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 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。

2.6.12.1. 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。

2.6.13. 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
- 镜像漏洞策略示例

2.6.13.1. 镜像漏洞策略 YAML 结构

在创建容器安全 Operator 策略时，它会涉及以下策略：

-

创建订阅的策略 (container-security-operator) 来引用名称和频道。此配置策略必须将 spec.remediationAction 设置为 enforce 来创建资源。订阅会拉取配置集，作为订阅支持的容器。查看以下示例：

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

- 一个 inform 配置策略来审核 ClusterServiceVersion，以确保容器安全 Operator 安装成功。查看以下示例：

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

- 一个 inform 配置策略，用于审核镜像漏洞扫描创建的任何 ImageManifestVuln 对象。查看

以下示例：

```
apiVersion: policy.open-cluster-management.io/v1
kind: ConfigurationPolicy
metadata:
 name: policy-imagemanifestvuln-example-imv
spec:
 remediationAction: inform  # will be overridden by remediationAction in parent policy
 severity: high
 namespaceSelector:
  exclude: ["kube-*"]
  include: ["*"]
 object-templates:
  - complianceType: mustnothave # mustnothave any ImageManifestVuln object
    objectDefinition:
      apiVersion: secscan.quay.redhat.com/v1alpha1
      kind: ImageManifestVuln # checking for a Kind
```

2.6.13.2. 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。

2.6.14. Red Hat OpenShift Platform Plus 策略集

配置并应用 OpenShift Platform Plus 策略集 (openshift-plus) 来安装 Red Hat OpenShift Platform Plus。

OpenShift Platform Plus 策略集包含两个部署的 PolicySets。OpenShift Plus 策略集应用设置为安装 OpenShift Platform Plus 产品的多个策略。Red Hat Advanced Cluster Security 安全集群服务和 Compliance Operator 部署到所有 OpenShift Container Platform 受管集群。

2.6.14.1. 先决条件

- 在 Amazon Web Services (AWS)环境中安装 Red Hat OpenShift Container Platform 4.13 或更高版本。
- 安装 Red Hat Advanced Cluster Management for Kubernetes 2.7 或更高版本。
- 安装策略生成器 Kustomize 插件。如需更多信息，请参阅策略生成器文档。

2.6.14.2. OpenShift Platform Plus 策略设置组件

当您将策略设置为 hub 集群时，会安装以下 OpenShift Platform Plus 组件：

表 2.12. 组件表

组件	策略	描述
Red Hat Advanced Cluster Security	**policy-acs-central-ca-bundle**	用于将中央服务器安装到 Red Hat Advanced Cluster Management for Kubernetes hub 集群和受管集群的策略。
	policy-acs-central-status	用于接收 Red Hat Advanced Cluster Security 状态的部署。
	policy-acs-operator-central	配置 Red Hat Advanced Cluster Security central operator。
	policy-acs-sync-resources	用于验证 Red Hat Advanced Cluster Security 资源是否已创建的策略。
OpenShift Container Platform	**policy-advanced-managed-cluster-status**	受管 hub 集群。受管集群的管理器。
Compliance operator	**policy-compliance-operator-install**	用于安装 Compliance operator 的策略。
Red Hat Quay	**policy-config-quay**	Red Hat Quay 的配置策略。
	policy-install-quay	用于安装 Red Hat Quay 的策略。
	policy-quay-status	安装到 Red Hat Advanced Cluster Management hub 集群中。
Red Hat Advanced Cluster Management	**policy-ocm-observability**	设置 Red Hat Advanced Cluster Management observability 服务。
Red Hat OpenShift Data Platform	**policy-odf**	Red Hat Advanced Cluster Management observability 和 Quay 使用的 hub 集群组件的可用存储。
	policy-odf-status	用于配置 Red Hat OpenShift Data Platform 状态的策略。

2.6.14.3. 其他资源

- 请参阅使用策略集安装 Red Hat OpenShift Platform Plus。
- 返回到 Policy set controller。
- 查看策略集中包含的所有策略的 openshift-plus 策略示例。

2.7. 管理监管仪表板

使用 *监管* 仪表板创建、查看和编辑资源，管理您的安全策略和策略违反情况。您可以使用命令行和控制台为您的策略创建 YAML 文件。从控制台继续读取 *监管* 仪表板的详细信息。

2.7.1. 监管页面

Governance page *Overview*、*Policy set* and *Policies* 中显示以下标签页。阅读以下描述以了解显示哪些信息：

- 概述

  以下概述卡显示在 *Overview* 选项卡中： *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：显示 *Name*、*Namespace* 和 *Cluster violations*。
  - 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。
  -

在 *Manage 列中* 重新排序您的表列。为要显示的对话框选择 *Manage 列* 图标。要重新排序列，选择 *Reorder* 图标并移动列名称。对于您要出现在表中的列，点特定列名称的复选框，然后选择 Save 按钮。

- 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。

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。

2.7.2. 监管自动化配置

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*：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：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 Automation Platform Job 模板需要为 EXTRA VARIABLES 段（也称为 extra_vars）启用 PROMPT ON LAUNCH。
- *运行 everyEvent 模式* ：违反策略时，每个受管集群的唯一策略都会运行自动化。使用 DelayAfterRunSeconds 参数在同一集群中重启自动化前设置最小秒数。如果策略在延迟期内违反多次，并保持在违反的状态，则自动化会在延迟期后运行一次。默认值为 0 秒，它仅适用于 everyEvent 模式。当您运行 everyEvent 模式时， target_clusters 和 Ansible Automation Platform 作业模板的额外变量与 *once mode* 相同。

- *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。

Ansible Automation Platform 作业的 extra_vars 中会自动提供以下变量：

- policy_name：在 hub 集群上启动 Ansible Automation Platform 作业的不合规根策略的名称。
- policy_namespace: root 策略的命名空间。
- hub_cluster ：hub 集群的名称，它由集群 DNS 对象中的值决定。
- policy_sets ：此参数包含根策略的所有关联的策略集名称。如果策略不在策略集中，policy_set 参数为空。
- policy_violations ：此参数包含不合规集群名称的列表，值是每个不合规集群的策略 status 字段。

2.7.3. 其他资源

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 Automation Platform
- 监管

2.7.4. 为监管配置 Ansible Automation Platform

Red Hat Advanced Cluster Management for Kubernetes 监管可与 Red Hat Ansible Automation Platform 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

2.7.4.1. 先决条件

- Red Hat OpenShift Container Platform 4.13 或更高版本
- 已安装 Ansible Automation Platform 版本 3.7.3 或更高版本。最佳实践是安装最新支持的 Ansible Automation Platform 版本。如需了解更多详细信息，请参阅 Red Hat Ansible Automation Platform 文档。
- 从 Operator Lifecycle Manager 安装 Ansible Automation Platform Resource Operator。在 *Update Channel* 部分中，选择 stable-2.x-cluster-scoped。选择 All namespaces on the cluster (default) 安装模式。

  注： 在运行 Ansible Automation Platform 作业模板时，请确保 Ansible Automation Platform 作业模板是幂等的。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 Red Hat Ansible Automation Platform 的更多信息，请参阅设置 Ansible 任务。

2.7.4.2. 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。如果您需要添加凭证，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*、*Run everyEvent mode* 或 *Disable automation*。

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。

2.7.4.3. 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 Automation 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 Automation 模板的任何参数。

6. 将模式设置为 once、everyEvent 或 disabled。

7. 将 policyRef 设置为您的策略的名称。

8. 在与包含 Ansible Automation Platform 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。

   - 当策略处于 *everyEvent* 模式且 DelayAfterRunSeconds 超过定义的时间值时，策略不合规，且自动化会针对每个策略违反。

2.8. 模板处理简介

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。

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。

模板必须符合 Golang 模板语言规格，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。

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。要解决这个问题，请使用 fromSecret 或 copySecretData，它会自动加密 secret 中的值，或 protect 加密其他值。

- 当您添加多行字符串值时，如证书，始终在模板管道末尾添加 | toRawJson | toLiteral 语法，以处理换行符。例如，如果您要从 Secret 资源复制证书并将其包含在 ConfigMap 资源中，则模板管道可能类似以下示例：

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

  toRawJson 模板函数将输入值转换为带有转义的新行的 JSON 字符串，不会影响 YAML 结构。toLiteral 模板函数从输出中删除外部单引号。例如，当为 密钥处理模板时： '{{ 'hello\nworld' | toRawJson }}' 模板管道，输出为 key: '"hello\nworld"'。密钥的输出： '{{ 'hello\nworld' | toRawJson | toLiteral }}' 模板管道是 key: "hello\nworld".

如需 hub 集群和受管集群模板的比较，请参阅下表：

2.8.1. hub 集群和受管集群模板的比较

表 2.13. 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。**.ManagedClusterLabels** 变量也可用，它解析为传播策略的受管集群中标签的键和值映射。	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。

继续阅读以下主题：

- 模板功能
- 配置策略中的高级模板处理

2.8.2. 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*模板处理*。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，它更为高级，请熟悉正在查找的资源的 YAML 结构。除了这些功能外，还提供 base64enc、 base64enc、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* 功能
- *fromConfigMap* function
- *fromClusterClaim* 功能
- *lookup* 功能
- *base64enc* 功能

- *base64dec* 功能
- *indent* 功能
- *autoindent* 功能
- *toInt* 功能
- *toBool* 功能
- *保护功能*
- *toLiteral* 功能
- *copySecretData* function
- *copyConfigMapData* function
- 支持的 Sprig 开源功能

2.8.2.1. *fromSecret* 功能

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*模板处理*。

注： 当您将此功能与 hub 集群模板搭配使用时，输出会使用 protect 功能自动加密。

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
```

重要： 当您添加多行字符串值时，如证书，始终在模板管道末尾添加 | toRawJson | toLiteral 语法，以处理换行符。例如，如果您要从 Secret 资源复制证书并将其包含在 ConfigMap 资源中，则模板管道可能类似以下示例：

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

toRawJson 模板函数将输入值转换为带有转义的新行的 JSON 字符串，不会影响 YAML 结构。toLiteral 模板函数从输出中删除外部单引号。例如，当为 密钥处理模板时： '{{ 'hello\nworld' | toRawJson }}' 模板管道，输出为 key: '"hello\nworld"'。密钥的输出： '{{ 'hello\nworld' | toRawJson | toLiteral }}' 模板管道是 key: "hello\nworld".

2.8.2.2. *fromConfigmap* 功能

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*模板处理*。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
```

2.8.2.3. *fromClusterClaim* 功能

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) (dataValue string, 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
```

2.8.2.4. *lookup* 功能

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 *支持的 Sprig 开源功能*部分。

查看该功能的以下语法：

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

使用此功能时，输入 Kubernetes 资源的 API 版本、类型、命名空间、名称和可选标签选择器。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 *模板处理*。有关标签选择器示例，请参阅 *Kubernetes 标签和选择器* 文档的 *额外资源* 部分。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
```

2.8.2.5. *base64enc* 功能

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 }}'
```

-

2.8.2.6. *base64dec* 功能

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 }}"
```

2.8.2.7. *indent* 功能

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 }}
```

2.8.2.8. *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 }}
```

2.8.2.9. *toInt* 功能

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 }}
```

2.8.2.10. *toBool* 功能

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 }}
```

2.8.2.11. *protect* 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:okrrBqt72ol+3WT/0vxel3vGa+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。

2.8.2.12. *toLiteral* 功能

toLiteral 函数会在处理模板字符串后删除任何引号。您可以使用此功能将 JSON 字符串从 ConfigMap 字段转换为清单中的 JSON 值。运行以下功能从 key 参数值中删除引号：

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

使用 toLiteral 功能后，会显示以下更新：

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

2.8.2.13. *copySecretData* function

copySecretData 功能复制指定 secret 的所有数据内容。查看以下函数示例：

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

注： 当您将此功能与 hub 集群模板搭配使用时，输出会使用 protect 功能自动加密。

2.8.2.14. *copyConfigMapData* function

copyConfigMapData 功能复制指定 ConfigMap 的所有 data 内容。查看以下函数示例：

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

2.8.2.15. 支持的 Sprig 开源功能

另外，Red Hat Advanced Cluster Management 还支持 sprig 开源项目中包含的以下模板功能：

表 2.14. 支持的社区 Sprig 功能表

sprig 库	Functions
加密和安全性	**htpasswd**
Date	**date**,**mustToDate**,**now**,**toDate**
default	**Default**,**empty**,**fromJson**,**mustFromJson**,**ternary**,**toJson**,**toRawJson**
字典和字典	**dig**
整数数	**添加**,**mul**,**div**,**round**,**sub**

sprig 库	Functions
整数片段	**直到，直到Step，**
列表	**附加,concat,has,list,mustAppend,mustHas,mustPrepend**, mustPrepend,**mustSlice,prepend,slice**
字符串函数	**cat,contains,hasPrefix,hasSuffix,join**, join ,**lower,replace,split,splitn,substr,trim,trimAll,trunc,upper**
版本比较	**semver**, **semverCompare**

2.8.2.16. 其他资源

- 如需了解更多详细信息，请参阅 模板处理。
- 有关用例，请参阅配置策略中的高级模板处理。
- 有关标签选择器示例，请参阅 Kubernetes 标签和选择器 文档。
- 请参阅 Golang 文档 - 软件包模板。
- 如需了解更多详细信息，请参阅 Sprig Function 文档。

2.8.3. 配置策略中的高级模板处理

使用受管集群和 hub 集群模板来减少在策略定义中为每个目标集群或硬代码配置值创建单独的策略的需求。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。

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。要解决这个问题，请使用 fromSecret 或 copySecretData，它会自动加密 secret 中的值，或 protect 加密其他值。

继续阅读高级模板用例：

- 重新处理的特殊注解
- 对象模板处理
- 绕过模板处理

2.8.3.1. 重新处理的特殊注解

hub 集群模板会在策略创建过程中或更新引用的资源时解析到引用资源中的数据。

如果您需要手动启动更新，请使用特殊注解 policy.open-cluster-management.io/trigger-update 来指示模板引用的数据的更改。对特殊注解值的任何更改都会自动启动模板处理。另外，引用资源的最新内容会在传播以在受管集群上处理的策略定义中读取和更新。使用此注解的方法是每次递增值。

2.8.3.2. 对象模板处理

使用 YAML 字符串表示设置对象模板。object-template-raw 参数是一个可选参数，它支持高级模板用例，如 if-else 和 range 功能。以下示例定义了将 species-category: mammal 标签添加到 default 命名空间中的任何 ConfigMap 中，其 name 键等于 Sea Otter ：

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

注：虽然 spec.object-templates 和 spec.object-templates-raw 是可选的，但必须设置两个参数字段中的一个。

查看以下策略示例，它使用高级模板为您的受管集群创建和配置基础架构 MachineSet 对象。

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

```
            - key: node-role.kubernetes.io/{{ $machineset_role }}
              effect: NoSchedule
            providerSpec:
              value:
                ami:
                  id: {{ $worker_ms.spec.template.spec.providerSpec.value.ami.id }}
                apiVersion: awsproviderconfig.openshift.io/v1beta1
                blockDevices:
                  - ebs:
                      encrypted: true
                      iops: 2000
                      kmsKey:
                        arn: ''
                      volumeSize: 500
                      volumeType: io1
                credentialsSecret:
                  name: aws-cloud-credentials
                deviceIndex: 0
                instanceType: {{ $worker_ms.spec.template.spec.providerSpec.value.instanceType
}}
                iamInstanceProfile:
                  id: {{ $infrastructure_id }}-worker-profile
                kind: AWSMachineProviderConfig
                placement:
                  availabilityZone: {{ $zone }}
                  region: {{ $region }}
                securityGroups:
                  - filters:
                      - name: tag:Name
                        values:
                          - {{ $infrastructure_id }}-worker-sg
                subnet:
                  filters:
                    - name: tag:Name
                      values:
                        - {{ $infrastructure_id }}-private-{{ $zone }}
                tags:
                  - name: kubernetes.io/cluster/{{ $infrastructure_id }}
                    value: owned
                userDataSecret:
                  name: worker-user-data
    {{- end }}
```

2.8.3.3. 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。

2.8.3.4. 其他资源

- 如需了解更多详细信息，请参阅模板功能。
- 返回到模板处理。
- 如需了解更多详细信息，请参阅 Kubernetes 配置策略控制器。
- 另请参阅 Red Hat OpenShift Container Platform etcd 加密文档。

2.9. 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。

查看以下部分：

- 创建安全策略
- 更新安全策略
- 删除安全策略
- 清理由策略创建的资源

2.9.1. 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：* 您必须定义一个放置和放置绑定，才能将策略应用到特定集群。PlacementBinding 资源绑定放置。为 cluster *Label selector* 字段输入有效的值来定义 Placement 和 PlacementBinding 资源。为了使用 Placement 资源，ManagedClusterSet 资源必须绑定到带有 ManagedClusterSetBinding 资源的 Placement 资源的命名空间。如需了解更多详细信息，请参阅创建 *ManagedClusterSetBinding* 资源。

2.9.1.1. 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：

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

2. 定义策略使用的模板。通过添加 policy-templates 字段来定义模板来编辑 YAML 文件。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
  namespaceSelector:
    include:
    - "default"
    - "my-namespace"
  policy-templates:
    - objectDefinition:
        apiVersion: policy.open-cluster-management.io/v1
        kind: ConfigurationPolicy
        metadata:
          name: operator
          # namespace: # will be supplied by the controller via the namespaceSelector
        spec:
          remediationAction: "inform"
          object-templates:
          - complianceType: "musthave" # at this level, it means the role must exist and
must have the following rules
            apiVersion: rbac.authorization.k8s.io/v1
            kind: Role
            metadata:
              name: example
```

```
        objectDefinition:
          rules:
            - complianceType: "musthave" # at this level, it means if the role exists the
rule is a musthave
              apiGroups: ["extensions", "apps"]
              resources: ["deployments"]
              verbs: ["get", "list", "watch", "create", "delete","patch"]
```

3. 定义一个 PlacementBinding 资源，将您的策略绑定到 Placement 资源。您的 PlacementBinding 资源可能类似以下 YAML 示例：

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

2.9.1.1.1. 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：

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

2. 运行以下命令，查看您的安全策略的描述：

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

2.9.1.2. 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。

1. 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'
    policy.open-cluster-management.io/description:
spec:
  complianceType: musthave
  namespaces:
    exclude: ["kube*"]
    include: ["default"]
    pruneObjectBehavior: None
  object-templates:
  - complianceType: musthave
    objectDefinition:
      apiVersion: v1
      kind: Pod
      metadata:
        name: pod1
      spec:
        containers:
        - name: pod-name
          image: 'pod-image'
          ports:
          - containerPort: 80
  remediationAction: enforce
  disabled: false
```

请参阅以下 PlacementBinding 示例：

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

请参阅以下 Placement 示例：

```
apiVersion: cluster.open-cluster-management.io/v1beta1
 kind: Placement
 metadata:
   name: placement-pod
spec:
  predicates:
  - requiredClusterSelector:
      labelSelector:
        matchLabels:
          cloud: "IBM"
```

2. 可选： 为您的策略添加描述。

3. 点击 Create Policy。从控制台创建了安全策略。

2.9.1.2.1. 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。

1. 进入 *Governance* 页面，以查看您的策略的表列表。注： 您可以选择 *Policies* 标签页或 *Cluster violations* 选项卡来过滤策略列表。

2. 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。

3. 或者，选择 *Policies* 选项卡来查看策略列表。展开一个策略行，以查看 *Description*,*Standards*,*Controls*, 和 *Categories* 详情。

2.9.1.3. 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：

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

2.9.1.4. 从控制台创建策略集

1. 在导航菜单中选择 **Governance**。
2. 选择 ***Policy set*** 选项卡。
3. 选择 **Create policy set** 按钮并完成表单。
4. 添加您的策略集的详细信息，然后选择 **Submit** 按钮。

您的策略从策略表中列出。

2.9.2. 更新安全策略

了解如何更新安全策略。

2.9.2.1. 通过 CLI 将策略添加到策略集中

1. 运行以下命令来编辑您的策略集：

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

2. 将策略名称添加到策略集的 **policies** 部分的列表中。
3. 使用以下命令在策略集的 **placement** 部分中应用添加的策略：

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

PlacementBinding 和 **Placement** 都创建。

注： 如果您删除放置绑定，策略仍会由策略集放置。

2.9.2.2. 从控制台在策略集中添加策略

1. 选择 ***Policy set*** 选项卡，在策略集中添加一个策略。
2. 选择 **Actions** 图标并选择 **Edit**。此时会出现 ***Edit policy set*** 表单。
3. 进入到表单的 ***Policies*** 部分，以选择要添加到策略集的策略。

2.9.2.3. 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。

2.9.3. 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：

```
oc delete policies.policy.open-cluster-management.io <policy-name> -n <policy-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：oc get policies.policy.open-cluster-management.io <policy-name> -n <policy-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。

2.9.3.1. 从控制台创建策略集

1. 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。

2. 点击 Delete 按钮。

2.9.4. 清理由策略创建的资源

在配置策略中使用 pruneObjectBehavior 参数来清理策略创建的资源。当设置 pruneObjectBehavior 时，仅在删除与其关联的配置策略（或父策略）后，才会清理相关的对象。

查看可用于参数的值的以下描述：

- DeleteIfCreated ：清理策略创建的所有资源。
- DeleteAll ：清理策略管理的所有资源。
- None：这是默认值，维护之前版本中的相同行为，但没有删除相关资源。

您可以在命令行中创建策略时，直接在 YAML 文件中设置值。

在控制台中，您可以选择 *Policy templates* 步骤的 *Prune Object Behavior* 部分中的值。

备注：

- 如果安装 Operator 的策略定义了 pruneObjectBehavior 参数，则需要额外的清理来完成 Operator 卸载。您可能需要在这个清理过程中删除 operator ClusterServiceVersion 对象。
- 当您在受管集群中禁用 config-policy-addon 资源时，会忽略 pruneObjbectBehavior。要在策略上自动清理相关资源，您必须在禁用附加组件前从受管集群中删除策略。

2.9.5. 其他资源

- 在策略概述中查看策略 YAML 文件的更多描述。
- 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 stable Policysets，它需要 Policy Generator 用于部署，PolicySets-table。
- 有关策略的更多主题，请参阅监管。

2.9.6. 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。

需要的访问权限 ： 管理员或集群管理员

- 创建配置策略
- 更新配置策略

- 删除配置策略

2.9.6.1. 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。

如果您有一个现有的 Kubernetes 清单，请考虑使用 Policy Generator 在策略中自动包含清单。请参阅策略生成器文档。查看以下部分以创建配置策略：

2.9.6.1.1. 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：

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

2. 运行以下命令来应用策略：

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

3. 运行以下命令，验证并列出策略：

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

您的配置策略已创建。

2.9.6.1.2. 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：

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

2. 运行以下命令，查看您的配置策略的描述：

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

2.9.6.1.3. 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。

1. 从控制台登录到集群，然后从导航菜单中选择 Governance。

2. 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。

3. 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**

4. **点 Create。您的配置策略已创建。**

2.9.6.1.4. 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。

从控制台登录到集群后，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。

2.9.6.2. 更新配置策略

查看以下部分以了解如何更新配置策略。

2.9.6.2.1. 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 *Governance* 页面来完成任务。

1. **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。**

2. 点击 **Disable policy**。

策略被禁用，但不被删除。

2.9.6.3. 删除配置策略

通过 CLI 或控制台删除配置策略。

- 使用以下步骤通过 CLI 删除配置策略：

1. 运行以下命令以从目标集群或集群中删除策略：

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

2. 运行以下命令验证您的策略是否已移除：

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

- 使用以下步骤从控制台删除配置策略：

1. 在导航菜单中点 **Governance** 来查看您的策略的表列表。
2. 在策略违反表中点击您要删除的策略的 **Actions** 图标，然后点 **Remove**。
3. 在 ***Remove policy*** 对话框中点击 **Remove policy**。

您的策略已删除。

2.9.6.4. 其他资源

- 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，请参阅 示例配置策略表，以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。

2.9.7. 在断开连接的环境中管理 Operator 策略

您可能需要在没有连接到互联网（断开连接）的 Red Hat OpenShift Container Platform 集群上部署 Red Hat Advanced Cluster Management for Kubernetes 策略。如果您使用您部署的策略来部署安装 Operator Lifecycle Manager Operator 的策略，您必须按照以下步骤 镜像 Operator 目录。

完成以下步骤以验证对 Operator 镜像的访问：

1. 请参阅验证所需软件包可用来验证您与策略一起使用的软件包是否可用。您必须验证由以下策略部署到的任何受管集群使用的每个镜像 registry 的可用性：
   - container-security-operator
   - 已弃用： gatekeeper-operator-product
   - compliance-operator
2. 请参阅配置镜像内容源策略以验证源是否可用。镜像内容源策略必须存在于每个断开连接的受管集群中，并可使用策略进行部署以简化流程。请参阅以下镜像源位置表：

监管策略类型	镜像源位置
容器安全性	registry.redhat.io/quay
Compliance	registry.redhat.io/compliance
Gatekeeper	registry.redhat.io/rhacm2

2.9.8. 使用策略集合安装 Red Hat OpenShift Platform Plus

继续阅读以获取应用 Red Hat Openshift Platform Plus 策略集的指导。应用 Red Hat OpenShift 策略集时，Red Hat Advanced Cluster Security secured 集群服务和 Compliance Operator 部署到所有

OpenShift Container Platform 受管集群。

2.9.8.1. 先决条件

应用策略集前完成以下步骤：

1. 要允许将订阅应用到集群，您必须应用 policy-configure-subscription-admin-hub.yaml 策略，并将补救操作设置为 enforce。将以下 YAML 复制并粘贴到控制台的 YAML 编辑器中：

```
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  name: policy-configure-subscription-admin-hub
  annotations:
    policy.open-cluster-management.io/standards: NIST SP 800-53
    policy.open-cluster-management.io/categories: CM Configuration Management
    policy.open-cluster-management.io/controls: CM-2 Baseline Configuration
spec:
  remediationAction: inform
  disabled: false
  policy-templates:
    - objectDefinition:
        apiVersion: policy.open-cluster-management.io/v1
        kind: ConfigurationPolicy
        metadata:
          name: policy-configure-subscription-admin-hub
        spec:
          remediationAction: inform
          severity: low
          object-templates:
            - complianceType: musthave
              objectDefinition:
                apiVersion: rbac.authorization.k8s.io/v1
                kind: ClusterRole
                metadata:
                  name: open-cluster-management:subscription-admin
                rules:
                - apiGroups:
                  - app.k8s.io
                  resources:
                  - applications
                  verbs:
                  - '*'
                - apiGroups:
                  - apps.open-cluster-management.io
                  resources:
                  - '*'
                  verbs:
                  - '*'
                - apiGroups:
                  - ''''
```

```
              resources:
              - configmaps
              - secrets
              - namespaces
              verbs:
              - '*'
          - complianceType: musthave
            objectDefinition:
              apiVersion: rbac.authorization.k8s.io/v1
              kind: ClusterRoleBinding
              metadata:
                name: open-cluster-management:subscription-admin
              roleRef:
                apiGroup: rbac.authorization.k8s.io
                kind: ClusterRole
                name: open-cluster-management:subscription-admin
              subjects:
              - apiGroup: rbac.authorization.k8s.io
                kind: User
                name: kube:admin
              - apiGroup: rbac.authorization.k8s.io
                kind: User
                name: system:admin
---
apiVersion: policy.open-cluster-management.io/v1
kind: PlacementBinding
metadata:
  name: binding-policy-configure-subscription-admin-hub
placementRef:
  name: placement-policy-configure-subscription-admin-hub
  kind: Placement
  apiGroup: cluster.open-cluster-management.io
subjects:
- name: policy-configure-subscription-admin-hub
  kind: Policy
  apiGroup: policy.open-cluster-management.io
---
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement-policy-configure-subscription-admin-hub
spec:
  predicates:
  - requiredClusterSelector:
      labelSelector:
        matchExpressions:
        - {key: name, operator: In, values: ["local-cluster"]}
```

2. 要从命令行界面应用前面的 YAML，请运行以下命令：

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

3.

安装 Policy Generator kustomize 插件。使用 Kustomize v4.5 或更高版本。请参阅 生成策略以安装 Operator。

4. 策略安装到 policies 命名空间。您必须将该命名空间绑定到 ClusterSet。例如，复制并应用以下示例 YAML 将命名空间绑定到默认的 ClusterSet ：

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

5. 运行以下命令以从命令行界面应用 ManagedClusterSetBinding 资源：

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

满足先决条件后，您可以应用策略集。

2.9.8.2. 应用 Red Hat OpenShift Platform Plus 策略集

1. 使用 openshift-plus/policyGenerator.yaml 文件，其中包含 Red Hat OpenShift Plus 的先决条件配置。请参阅 openshift-plus/policyGenerator.yaml。

2. 使用 kustomize 命令将策略应用到您的 hub 集群：

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

注： 对于您不想安装的 OpenShift Platform Plus 的任何组件，请编辑 policyGenerator.yaml 文件并删除或注释掉这些组件的策略。

2.9.8.3. 其他资源

- 如需了解策略集的概述，请参阅 Red Hat OpenShift Platform Plus 策略集。
- 返回到主题的开头，使用策略集安装 Red Hat OpenShift Platform Plus

2.9.9. 使用 *OperatorPolicy* 资源安装 Operator （技术预览）

要在受管集群中安装 Operator Lifecycle Manager (OLM)受管 Operator，请在 Policy 定义中使用 OperatorPolicy 策略模板。

2.9.9.1. 创建 *OperatorPolicy* 资源以安装 Quay

请参阅以下 operator 策略示例，它使用 Red Hat operator 目录在 stable-3.11 频道中安装最新的 Quay Operator：

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

添加 OperatorPolicy 策略模板后，使用控制器在集群中创建 operatorGroup 和 subscription 对象。因此，OLM 会完成其余的安装。您可以查看受管集群上 OperatorPolicy 资源的 .status.Conditions 和 .status.relatedObjects 字段中拥有的资源的健康状态。

要验证 Operator 策略状态，请在受管集群中运行以下命令：

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

2.9.9.2. 其他资源

请参阅 Operator 策略控制器（技术预览）

2.10. 策略依赖项

在满足依赖项条件时，依赖项可用于激活策略或策略模板。在受管集群中的以下字段会被检查：dependencies 和 extraDependencies。没有满足依赖项时，复制策略模板的模板状态会显示更多详细信息。

需要的访问权限：策略管理员

查看以下策略依赖项示例，只有 upstream-compliance-operator 策略已在受管集群中合规时才会创建 ScanSettingBinding ：

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

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

注： 依赖项不能用于根据另一个集群中的策略状态应用到一个集群。

2.11. 保护 HUB 集群

通过增强 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

第 3 章 GATEKEEPER OPERATOR

Gatekeeper 是一个验证 webhook，它带有审计功能，可以强制执行基于自定义资源定义的策略，该策略使用 Open Policy Agent (OPA) 运行。您可以使用 Gatekeeper operator 策略在集群中安装 Gatekeeper。Gatekeeper 约束可用于评估 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。

先决条件：安装 Gatekeeper 并将 Gatekeeper 策略应用到集群时，需要 Red Hat Advanced Cluster Management for Kubernetes 或 Red Hat OpenShift Container Platform Plus 订阅。只有在最新版本的 Red Hat Advanced Cluster Management 支持的 OpenShift Container Platform 版本（版本 3.11 除外）上支持 gatekeeper。

继续阅读以了解更多有关使用 Gatekeeper operator 的信息：

集成 Gatekeeper 约束和约束模板管理 Gatekeeper operator 策略

3.1. 集成 GATEKEEPER 约束和约束模板

Gatekeeper 策略通过使用约束模板(ConstraintTemplates)和约束编写。查看 Red Hat Advanced Cluster Management 策略中使用 Gatekeeper 约束的以下 YAML 示例：

- ConstraintTemplates 和约束：使用 Red Hat Advanced Cluster Management 策略在 hub 集群上进行多集群发布 Gatekeeper 约束和 Gatekeeper 审计结果聚合。以下示例定义了一个 Gatekeeper ConstraintTemplate 和 constraint (K8sRequiredLabels)，以确保在所有命名空间中设置了 gatekeeper 标签：

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

```
        spec:
          names:
            kind: K8sRequiredLabels
          validation:
            openAPIV3Schema:
              properties:
                labels:
                  type: array
                  items: string
      targets:
        - target: admission.k8s.gatekeeper.sh
          rego: |
            package k8srequiredlabels
            violation[{"msg": msg, "details": {"missing_labels": missing}}] {
              provided := {label | input.review.object.metadata.labels[label]}
              required := {label | label := input.parameters.labels[_]}
              missing := required - provided
              count(missing) > 0
              msg := sprintf("you must provide labels: %v", [missing])
            }
  - objectDefinition:
      apiVersion: constraints.gatekeeper.sh/v1beta1
      kind: K8sRequiredLabels
      metadata:
        name: ns-must-have-gk
        annotations:
          policy.open-cluster-management.io/severity: low (3)
      spec:
        enforcementAction: dryrun
        match:
          kinds:
            - apiGroups: [""]
              kinds: ["Namespace"]
        parameters:
          labels: ["gatekeeper"]
```

(1)

因为 remediationAction 被设置为 inform，所以 Gatekeeper 约束的 enforcementAction 字段会被覆盖来 warn。这意味着 Gatekeeper 会检测并警告您创建或更新缺少 gatekeeper 标签的命名空间。如果策略 remediationAction 设置为 enforce，则 Gatekeeper 约束 enforcementAction 字段会被覆盖为 deny。在这种情况下，此配置可防止任何用户创建或更新缺少 gatekeeper 标签的命名空间。

(2) (3)

可选：为每个 Gatekeeper 约束或约束模板设置 policy.open-cluster-management.io/severity 注解的严重性值。有效值与其他 Red Hat Advanced Cluster Management 策略类型相同：low, medium, high, 或 critical。

使用前面的策略，您可能会收到以下策略状态消息：warn - you must provide labels: {"gatekeeper"} (on Namespace default); warn - you must provide labels: {"gatekeeper"}

(on Namespace gatekeeper-system)。当包含 Gatekeeper 约束或 ConstraintTemplates 被删除后，限制和 ConstraintTemplates 也会从受管集群中删除。

要从控制台查看特定受管集群的 Gatekeeper 审计结果，请进入策略模板 *结果* 页面。如果启用了搜索，请查看失败的审计的 Kubernetes 对象的 YAML。

备注：

- 只有在 Gatekeeper 版本 3.9 或更高版本生成的审计结果时，*相关资源* 部分才可用。
- Gatekeeper 审计功能默认每分钟运行一次。审计结果会发回到 hub 集群，以便在受管集群的 Red Hat Advanced Cluster Management 策略状态中查看。

- policy-gatekeeper-admission ：在 Red Hat Advanced Cluster Management 策略中使用 policy-gatekeeper-admission 配置策略来检查 gatekeeper admission webhook 拒绝的 Kubernetes API 请求：

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

3.1.1. 其他资源

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

- 如需了解更多详细信息，请参阅 OPA Gatekeeper 是什么。
- 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。

3.2. 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 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
- 升级 Gatekeeper 和 Gatekeeper operator
- 更新 Gatekeeper operator 策略
- 删除 Gatekeeper operator 策略
- 卸载 Gatekeeper 策略、Gatekeeper 和 Gatekeeper operator 策略

3.2.1. 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 文档中的将 Operator* 添加到集群部分。

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 自定义资源来配置 Gatekeeper。查看 Gatekeeper operator 自定义资源示例。

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。

3.2.2. 从控制台创建 Gatekeeper 策略

请参阅以下流程来从控制台创建策略：

1. 通过从控制台创建策略来安装 Gatekeeper 策略。另外，您可以进入 *Additional resources* 部分，以引用示例 YAML 以部署 policy-gatekeeper-operator.yaml。
2. 登录到集群后，进入 *Governance* 页面。
3. 选择 **Create policy**。
4. 在完成表单时，从 *Specifications* 字段中选择 Gatekeeper Operator。策略的参数值会自动填充，策略默认设置为 **inform**。
5. 将补救操作设置为 **enforce** 来安装 gatekeeper。

注： 默认值由 Operator 生成。

3.2.2.1. Gatekeeper operator 自定义资源

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

```
    constraintViolationLimit: 55
    auditFromCache: Enabled
    auditChunkSize: 66
    emitAuditEvents: Enabled
    resources:
      limits:
        cpu: 500m
        memory: 150Mi
      requests:
        cpu: 500m
        memory: 130Mi
  validatingWebhook: Enabled
  mutatingWebhook: Enabled
  webhook:
    replicas: 3
    emitAdmissionEvents: Enabled
    admissionEventsInvolvedNamespace: Enabled (2)
    disabledBuiltins: - http.send
    operations: (3)
     - "CREATE"
     - "UPDATE"
     - "CONNECT"
    failurePolicy: Fail
    resources:
      limits:
        cpu: 480m
        memory: 140Mi
      requests:
        cpu: 400m
        memory: 120Mi
  nodeSelector:
    region: "EMEA"
  affinity:
    podAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        - labelSelector:
            matchLabels:
              auditKey: "auditValue"
          topologyKey: topology.kubernetes.io/zone
  tolerations:
    - key: "Example"
      operator: "Exists"
      effect: "NoSchedule"
  podAnnotations:
    some-annotation: "this is a test"
    other-annotation: "another test"
```

+ <1> 启用 auditEventsInvolvedNamespace 参数，以管理要创建的命名空间审计事件。因此，以下参数会添加到 Gatekeeper 控制器部署中，--audit-events-involved-namespace: true. <2> 启用 admissionEventsInvolvedNamespace 参数来管理要创建的命名空间准入事件。因此，以下参数被添加到 Gatekeeper 控制器部署中，--admission-events-involved-namespace: true.<3> 您可以使用以下值来管理 webhook 操作。对 operations 参数使用以下值： "CREATE", "UPDATE", "CONNECT", 和 "DELETE "。

3.2.2.2. 为同步详情配置 *auditFromCache*

Gatekeeper operator 在 auditFromCache 审计中的自定义资源定义中公开设置，该设置默认为禁用。如果启用了 auditFromCache，则需要为同步详情设置 config.gatekeeper.sh。请参阅 Gatekeeper 文档中的配置审计 部分。

Gatekeeper operator 从用户安装的限制中收集资源，然后将这些资源插入到配置中。如果资源不存在，Operator 会创建配置资源。

1. 在 Gatekeeper 资源中将 auditFromCache 设置为 Automatic，如下例所示：

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

请参阅 Gatekeeper operator 在以下示例的配置文件中添加 syncOnlys 参数部分：

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

2. 在终端中运行以下命令，获取 同步 设置的说明：

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

3.2.3. 升级 Gatekeeper 和 Gatekeeper operator

您可以升级 Gatekeeper 和 Gatekeeper operator 的版本。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，您必须手动批准每个集群的 Gatekeeper Operator 升级。

3.2.4. 更新 Gatekeeper operator 策略

查看以下部分以了解如何更新 Gatekeeper operator 策略。

3.2.4.1. 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： 您可以选择 ***Policies*** 标签页或 ***Cluster violations*** 选项卡来过滤策略列表。

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。

3.2.4.2. 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。

3.2.5. 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：

  ```
  oc delete policies.policy.open-cluster-management.io <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：

  ```
  oc get policies.policy.open-cluster-management.io <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 策略已删除。

3.2.6. 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 自定义资源的 ConfigurationPolicy 模板。

b. 将 ConfigurationPolicy 模板的 complianceType 的值改为 mustnothave。

3. 删除受管集群上的 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。

3.2.7. 其他资源

- 如需有关 gatekeeper 的详细信息，请参阅集成 gatekeeper 约束和约束模板。
- 请参阅 Policy Gatekeeper 示例。
- 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。
- 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。

第 4 章 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 policy-collection/community 中的第三方策略。

了解如何集成以下第三方策略：

- Gatekeeper operator
- 策略生成器

4.1. 策略生成器

Policy Generator 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，它使用 Kustomize 生成 Red Hat Advanced Cluster Management 策略。策略生成器从 Kubernetes 清单 YAML 文件构建 Red Hat Advanced Cluster Management 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，请阅读 *Kustomize 文档*。

如需更多信息，请参阅以下部分：

- 策略生成器功能
- 策略生成器配置结构
- 策略生成器配置参考表

4.1.1. 策略生成器功能

Policy Generator 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 Container Platform 集群，并通过 Red Hat Advanced Cluster Management 策略简化 Kubernetes 集群的发布。

使用 Policy Generator 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，包括从 Kustomize 目录创建的清单。
- 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，以便您可以通过 Red Hat Advanced Cluster Management for Kubernetes 报告 Gatekeeper 策略违反情况。
- 在 hub 集群上生成策略集。

4.1.2. 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。

要使用插件，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 PolicyGenerator 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。另外，您可以设置 Kustomize 目录的路径，或者设置具有多个 Kubernetes 清单 YAML 文件的目录。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 Placement 和 PlacementBinding 可能类似以下示例：

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement-config-data
  namespace: policies
spec:
  predicates:
  - requiredClusterSelector:
      labelSelector:
        matchExpressions: []
---
apiVersion: policy.open-cluster-management.io/v1
kind: PlacementBinding
metadata:
  name: binding-config-data
  namespace: policies
placementRef:
  apiGroup: cluster.open-cluster-management.io
  kind: Placement
  name: placement-config-data
subjects:
- apiGroup: policy.open-cluster-management.io
  kind: Policy
  name: config-data
---
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  annotations:
    policy.open-cluster-management.io/categories: CM Configuration Management
    policy.open-cluster-management.io/controls: CM-2 Baseline Configuration
    policy.open-cluster-management.io/standards: NIST SP 800-53
    policy.open-cluster-management.io/description:
  name: config-data
  namespace: policies
spec:
  disabled: false
  policy-templates:
```

```
- objectDefinition:
    apiVersion: policy.open-cluster-management.io/v1
    kind: ConfigurationPolicy
    metadata:
      name: config-data
    spec:
      object-templates:
      - complianceType: musthave
        objectDefinition:
          apiVersion: v1
          data:
            key1: value1
            key2: value2
          kind: ConfigMap
          metadata:
            name: my-config
            namespace: default
      remediationAction: inform
      severity: low
```

4.1.3. 策略生成器配置参考表

请注意，每个策略都可以覆盖 policyDefaults 部分中除 namespace 以外的所有字段，每个策略集都可以覆盖 policySetDefaults 部分中的所有字段。

表 4.1. 参数表

字段	可选或必需的	描述
apiVersion	必填	将值设置为 **policy.open-cluster-management.io/v1**。
kind	必填	将值设为 **PolicyGenerator** 以表示策略类型。
metadata.name	必填	用于标识策略资源的名称。
placementBindingDefaults.name	选填	如果多个策略使用相同的放置，则使用此名称为生成的 **PlacementBinding** 生成唯一名称，将放置与引用它的策略数组绑定。
policyDefaults	必填	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.namespace	必填	所有策略的命名空间。

字段	可选或必需的	描述
policyDefaults.complianceType	选填	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。您可以使用的值是 **musthave**、**musthave** 或 **mustnothave**。默认值为 **musthave**。
policyDefaults.metadataComplianceType	选填	在将清单元数据部分与集群中的对象进行比较时，会覆盖 **complianceType**。您可以使用的值是 **musthave** 和 **mustonlyhave**。默认值为空 (**{}**)，以避免覆盖元数据的 **complianceType**。
policyDefaults.categories	选填	**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	选填	**policy.open-cluster-management.io/controls** 注解中使用的控制数组。默认值为 **CM-2 Baseline Configuration**。
policyDefaults.standards	选填	**policy.open-cluster-management.io/standards** 注解中使用的一组标准。默认值为 **NIST SP 800-53**。
policyDefaults.policyAnnotations	选填	策略在 **metadata.annotations** 部分中包含的注解。除非在策略中指定的，否则会适用于所有策略。默认值为空 (**{}**)。
policyDefaults.configurationPolicyAnnotations	选填	用于对生成的配置策略设置的注解的键值对。例如，您可以通过定义以下参数来禁用策略模板：**{"policy.open-cluster-management.io/disable-templates": "true"}**。默认值为空 (**{}**)。

字段	可选或必需的	描述
policyDefaults.copyPolicyMetadata	选填	复制所有策略的标签和注解，并将它们添加到副本策略中。默认设置为 **true**。如果设置为 **false**，则只有策略框架特定的策略标签和注解复制到复制策略中。
policyDefaults.severity	选填	策略违反的严重性。默认值为 **low**。
policyDefaults.disabled	选填	策略是否被禁用，代表不会传播它，因此结果为没有状态。默认值为 **false**，这会启用策略。
policyDefaults.remediationAction	选填	策略的补救机制。参数值是 **enforce** 和 **inform**。默认值是 **inform**。
policyDefaults.namespaceSelector	没有指定命名空间的命名空间对象是必需的	决定对象要应用到的受管集群中的命名空间。**include** 和 **exclude** 参数接受根据名称包含和排除命名空间的文件路径表达式。**matchExpressions** 和 **matchLabels** 参数指定要包含的命名空间。请参阅 *Kubernetes 标签和选择器*文档。生成的列表通过使用来自所有参数的结果的交集进行编译。
policyDefaults.evaluationInterval	选填	使用 **compliant** 和 **noncompliant** 参数指定特定合规状态时要评估的策略的频率。当受管集群具有较低的 CPU 资源时，可以增加评估间隔，以减少 Kubernetes API 上的 CPU 使用量。这是持续时间的格式。例如，**"1h25m3s"** 代表 1 小时，25 分钟和 3 秒。它们也可以设置为 "never"，以避免在策略成为特定合规状态后评估策略。
policyDefaults.pruneObjectBehavior	选填	决定在删除策略时是否应删除由策略创建或监控的对象。只有策略的补救操作被设置为 **enforce** 时才会进行修剪。示例值为 **DeleteIfCreated**、**DeleteAll** 或 **None**。默认值为 **None**。

字段	可选或必需的	描述
policyDefaults.recordDiff	选填	指定在策略中记录对象与 **objectDefinition** 之间的区别。设置为 **Log** 以记录控制器日志中的区别或 **None** 来记录区别。默认情况下，此参数为空，不记录区别。
policyDefaults.dependencies	选填	应用此策略前必须处于特定合规性状态的对象列表。当 **policyDefaults.orderPolicies** 设置为 **true** 时，无法指定。
policyDefaults.dependencies[].name	必填	要依赖的对象名称。
policyDefaults.dependencies[].namespace	选填	要依赖的对象的命名空间。默认为 Policy Generator 设置的策略命名空间。
policyDefaults.dependencies[].compliance	选填	对象需要处于的合规性状态。默认值为 **Compliant**。
policyDefaults.dependencies[].kind	选填	对象的种类。默认情况下，kind 设置为 **Policy**，但也可以是具有合规状态的其他类型，如 **ConfigurationPolicy**。
policyDefaults.dependencies[].apiVersion	选填	对象的 API 版本。默认值为 **policy.open-cluster-management.io/v1**。
policyDefaults.description	选填	您要创建的策略的描述。
policyDefaults.extraDependencies	选填	应用此策略前必须处于特定合规性状态的对象列表。您定义的依赖项会独立于 **dependencies** 列表添加到每个策略模板中（如 **ConfigurationPolicy**）。当 **policyDefaults.orderManifests** 被设置为 **true** 时，无法指定。
policyDefaults.extraDependencies[].name	必填	要依赖的对象名称。
policyDefaults.extraDependencies[].namespace	选填	要依赖的对象的命名空间。默认情况下，值设为为 Policy Generator 设置的策略命名空间。

字段	可选或必需的	描述
policyDefaults.extraDependencies[].compliance	选填	对象需要处于的合规性状态。默认值为 **Compliant**。
policyDefaults.extraDependencies[].kind	选填	对象的种类。默认值为 **Policy**，但也可以是具有合规状态的其他类型，如 **ConfigurationPolicy**。
policyDefaults.extraDependencies[].apiVersion	选填	对象的 API 版本。默认值为 **policy.open-cluster-management.io/v1**。
policyDefaults.ignorePending	选填	当 Policy Generator 等待其依赖项达到其所需状态时，绕过合规性状态检查。默认值为 **false**。
policyDefaults.orderPolicies	选填	自动生成有关策略的**依赖项**，以便它们按照您在策略列表中定义的顺序应用。默认情况下，值设为 **false**。无法与 **policyDefaults.dependencies** 同时指定。
policyDefaults.orderManifests	选填	在策略模板上自动生成 **extraDependencies**，以便它们按照您在该策略的清单列表中定义的顺序应用。当 **policyDefaults.consolidateManifests** 被设置为 **true** 时，无法指定。无法与 **policyDefaults.extraDependencies** 同时指定。
policyDefaults.consolidateManifests	选填	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 （已弃用）	选填	将 **informGatekeeperPolicies** 设置为 false 以直接使用 Gatekeeper 清单，而无需在配置策略中定义。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**。

字段	可选或必需的	描述
policyDefaults.informKyvernoPolicies	选填	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.policyLabels	选填	策略在 **metadata.labels** 部分中包含的标签。**policyLabels** 参数适用于所有策略，除非在策略中指定。
policyDefaults.policySets	选填	策略加入的策略集合的数组。策略设置详情可在 **policySets** 部分中定义。当一个策略是策略集的一部分时，对这个策略的放置绑定没有保证，因为会为集合生成一个。设置 **policies[].generatePlacementWhenInSet** 或 **policyDefaults.generatePlacementWhenInSet** 来覆盖 **policyDefaults.policySets**。
policyDefaults.generatePolicyPlacement	选填	为策略生成放置清单。默认设置为 **true**。当设置为 **false** 时，会跳过放置清单生成，即使指定了放置也是如此。
policyDefaults.generatePlacementWhenInSet	选填	当一个策略是策略集的一部分时，默认情况下，生成器不会为这个策略生成放置，因为会为策略集生成一个。将 **generatePlacementWhenInSet** 设置为 **true**，以使用策略放置和策略设置放置部署策略。默认值为 **false**。
policyDefaults.placement	选填	策略的放置配置。这默认为与所有集群匹配的放置配置。
policyDefaults.placement.name	选填	指定一个名称来整合包含相同集群标签选择器的放置。

字段	可选或必需的	描述
policyDefaults.placement.labelSelector	选填	使用 **key:value** 定义集群标签选择器，或为 **matchExpressions**、**matchLabels** 或两者都提供适当的值来指定放置。请参阅 ***placementPath*** 指定现有文件。
policyDefaults.placement.placementName	选填	定义此参数以使用集群中已存在的放置。没有创建 **Placement**，而是一个 **PlacementBinding** 将策略绑定到这个 **Placement**。
policyDefaults.placement.placementPath	选填	要重复使用现有放置，请指定相对于 **kustomization.yaml** 文件的位置的路径。如果提供，则默认所有策略使用此放置。请参阅 ***labelSelector*** 以生成新的 **Placement**。
policyDefaults.placement.clusterSelector (Deprecated)	选填	**PlacementRule** 已被弃用。使用 **labelSelector** 来生成放置。使用 **key:value** 或提供 **matchExpressions**、**matchLabels** 或使用适当的值来定义集群选择器来指定放置规则。请参阅 **placementRulePath** 以指定现有文件。
policyDefaults.placement.placementRuleName (Deprecated)	选填	**PlacementRule** 已被弃用。或者，使用 **placementName** 指定放置。要使用集群中的现有放置规则，请指定此参数的名称。**PlacementRule** 不会被创建，但 **PlacementBinding** 将策略绑定到现有的 **PlacementRule**。
policyDefaults.placement.placementRulePath (Deprecated)	选填	**PlacementRule** 已被弃用。或者，使用 **placementPath** 指定放置。要重复使用现有放置规则，请指定相对于 **kustomization.yaml** 文件的位置的路径。如果提供，则默认所有策略都使用此放置规则。请参阅 ***clusterSelector*** 以生成新的 **PlacementRule**。
policySetDefaults	选填	策略设置的默认值。为此参数列出的任何默认值都会被 **policySets** 数组中的条目覆盖。

字段	可选或必需的	描述
policySetDefaults.placement	选填	策略的放置配置。这默认为与所有集群匹配的放置配置。有关此字段的描述，请参阅 ***policyDefaults.placement***。
policySetDefaults.generatePolicySetPlacement	选填	为策略集合生成放置清单。默认设置为 **true**。当设置为 **false** 时，会跳过放置清单生成，即使指定了放置也是如此。
policies	必填	要创建的策略列表以及覆盖默认值或 **policyDefaults** 中设置的值。如需了解更多字段和描述，请参阅 ***policyDefaults***。
policies.description	选填	您要创建的策略的描述。
policies[].name	必填	要创建的策略的名称。
policies[].manifests	必填	策略中包含的 Kubernetes 对象清单列表，以及覆盖默认值、此**策略**项中设置的值或 **policyDefaults** 中设置的值。如需了解更多字段和描述，请参阅 **policyDefaults**。当将 **Manifests** 设置为 **true** 时，只有 **complianceType**、**metadataComplianceType** 和 **recordDiff** 可以在 **policies[].manifests** 级别被覆盖。
policies[].manifests[].path	必填	相对于 **kustomization.yaml** 文件，到单个文件的路径、平面文件目录或 Kustomize 目录。如果该目录是一个 Kustomize 目录，则生成器会在生成策略前针对目录运行 Kustomize。如果需要为 Kustomize 目录处理 Helm chart，请在运行策略生成器的环境中将 **POLICY_GEN_ENABLE_HELM** 设置为 **"true"**，以便为策略生成器启用 Helm。

字段	可选或必需的	描述
policies[].manifests[].patches	选填	应用到路径上清单的 Kustomize 补丁列表。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。
policies.policyLabels	选填	策略在 **metadata.labels** 部分中包含的标签。**policyLabels** 参数适用于所有策略，除非在策略中指定。
policySets	选填	要创建的策略集合列表，以及覆盖默认值或 **policySetDefaults** 中设置的值。要在策略集中包含策略，请使用 **policyDefaults.policySets**, **policies[].policySets**, 或 **policySets.policies**。如需了解更多字段和描述，请参阅 ***policySetDefaults***。
policySets[].name	必填	要创建的策略的名称。
policySets[].description	选填	要创建的策略集的描述。
policySets[].policies	选填	包括在策略集中的策略列表。如果还指定了 **policyDefaults.policySets** 或 **policies[].policySets**，则列表会被合并。

4.1.4. 其他资源

- 读取 生成策略以安装 GitOps Operator。
- 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，请参阅应用 Kustomize。

- 如需了解更多主题，请参阅监管文档。
- 请参阅 kustomization.yaml 文件示例。
- 请参阅 Kubernetes 标签和选择器 文档。
- 如需更多详细信息，请参阅 Gatekeeper。
- 请参阅 Kustomize 文档。
- 返回集成第三方策略控制器文档。

4.2. 生成安装 COMPLIANCE OPERATOR 的策略

生成将 Compliance Operator 安装到集群中的策略。对于使用 *namespaced* 安装模式的 Operator，如 Compliance Operator，还需要 OperatorGroup 清单。

完成以下步骤：

1. 创建一个 YAML 文件，其中包含 Namespace、Subscription 和名为 compliance-operator.yaml 的 OperatorGroup 清单。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/v1
kind: OperatorGroup
metadata:
  name: compliance-operator
  namespace: openshift-compliance
spec:
  targetNamespaces:
    - openshift-compliance
---
apiVersion: operators.coreos.com/v1alpha1
```

```
kind: Subscription
metadata:
  name: compliance-operator
  namespace: openshift-compliance
spec:
  channel: release-0.1
  name: compliance-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
```

2. 创建 PolicyGenerator 配置文件。查看以下 PolicyGenerator 策略示例，该示例在所有 OpenShift Container Platform 受管集群上安装 Compliance Operator：

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

3. 将策略生成器添加到 kustomization.yaml 文件中。generators 部分可能类似以下配置：

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

因此，生成的策略类似于以下文件：

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement-install-compliance-operator
  namespace: policies
spec:
  predicates:
  - requiredClusterSelector:
      labelSelector:
        matchExpressions:
        - key: vendor
```

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

```
  - complianceType: musthave
    objectDefinition:
      apiVersion: operators.coreos.com/v1
      kind: OperatorGroup
      metadata:
        name: compliance-operator
        namespace: openshift-compliance
      spec:
        targetNamespaces:
          - compliance-operator
remediationAction: enforce
severity: low
```

因此，会显示生成的策略。