Red Hat Advanced Cluster Management for Kubernetes 2.5

거버넌스

정책을 사용하여 클러스터 보안을 강화하는데 도움이 되는 거버넌스 정책 프레임워크에 대해 자세히 알아보십시오.

Last Updated: 2023-09-18

Red Hat Advanced Cluster Management for Kubernetes 2.5 거버넌스

정책을 사용하여 클러스터 보안을 강화하는데 도움이 되는 거버넌스 정책 프레임워크에 대해 자세히 알아보십시오.

법적 공지

초록

정책을 사용하여 클러스터 보안을 강화하는데 도움이 되는 거버넌스 정책 프레임워크에 대해 자세히 알아보십시오.

차례

1장. 위험 및 컴플라이언스

Red Hat Advanced Cluster Management for Kubernetes 구성 요소의 보안을 관리합니다. 정의된 정책 및 프로세스를 통해 클러스터를 관리하여 위험을 식별하고 최소화합니다. 정책을 사용하여 규칙을 정의하고 제어를 설정합니다.

사전 요구 **사항**: Red Hat Advanced Cluster Management for Kubernetes에 대한 인증 서비스 요구 사항을 구성해야 합니다. 자세한 내용은 액세스 제어를 참조하십시오.

클러스터 보안에 대한 자세한 내용은 다음 주제를 검토하십시오.

- 역할 기반 액세스 제어
- 인증 정보 관리 개요
- 인증서
- 거버넌스
  - 구성 정책의 템플릿 지원
  - 무결성 자동 보호 (기술 프리뷰)

1.1. 인증서

다양한 인증서가 Kubernetes용 Red Hat Advanced Cluster Management에서 생성되고 사용됩니다.

자체 인증서를 가져올 수 있습니다. 인증서에 대한 Kubernetes TLS 시크릿을 생성해야 합니다. 인증서를 생성한 후 Red Hat Advanced Cluster Management 설치 프로그램에서 생성한 특정 인증서를 교체할 수 있습니다.

필수 액세스: 클러스터 관리자

Red Hat Advanced Cluster Management에서 실행되는 서비스에 필요한 모든 인증서는 Red Hat Advanced Cluster Management를 설치하는 동안 생성됩니다. 인증서는 다음 구성 요소에서 생성하고 관리합니다.

- OpenShift 서비스 Serving certificates
- Red Hat Advanced Cluster Management webhook 컨트롤러
- Kubernetes Certificates API
- OpenShift 기본 수신

인증서 관리에 대해 자세히 알아보려면 계속 읽으십시오.

Red Hat Advanced Cluster Management hub 클러스터 인증서

- 관리 수신 인증서 교체
- OpenShift 기본 수신 인증서 교체
- 가시성 인증서
  - Bring Your Own (BYO) Observability 인증 기관 (CA) 인증서

- CA 인증서를 생성하는 OpenSSL 명령
- BYO 관찰 가능 CA 인증서와 관련된 시크릿 생성
- alertmanager 경로에 대한 인증서 교체

Red Hat Advanced Cluster Management 구성 요소 인증서

- 허브 클러스터 관리 인증서 나열
- refresh hub 클러스터 관리 인증서
- Red Hat Advanced Cluster Management Webhook 인증서 새로 고침

Red Hat Advanced Cluster Management 관리 인증서

- 채널 인증서
- 관리형 클러스터 인증서

타사 인증서

- 게이트키퍼 웹 후크 인증서 교체
- 무결성 방전 Webhook 인증서 교체 (기술 프리뷰)

참고: 사용자는 인증서 교체 및 업데이트를 담당합니다.

1.1.1. Red Hat Advanced Cluster Management hub 클러스터 인증서

1.1.1.1. 가시성 인증서

Red Hat Advanced Cluster Management가 설치되면 관찰 가능 구성 요소에서 관찰 가능 인증서를 생성 및 사용하여 허브 클러스터와 관리형 클러스터 간의 트래픽에 상호 TLS를 제공합니다. 관찰 가능 인증서와 관련된 Kubernetes 시크릿입니다.

open-cluster-management-observability 네임스페이스에는 다음 인증서가 포함됩니다.

- **observability-server-ca-certs**: 서버 측 인증서에 서명하기 위한 CA 인증서
- **observability-client-ca-certs**: 클라이언트 측 인증서에 서명하기 위한 CA 인증서
- **observability-server-certs**: **observability-observatorium-api** 배포에서 사용하는 서버 인증서
- **observability-grafana-certs**: **observability-rbac-query-proxy** 배포에서 사용하는 클라이언트 인증서

open-cluster-management-addon-observability 네임스페이스에는 관리 클러스터에서 다음 인증서가 포함됩니다.

- **observability-managed-cluster-certs**: hub 서버에 **observability-server-ca-certs** 와 동일한 서버 CA 인증서
- **observability-controller-open-cluster-management.io-observability-signer-client-cert**: **metrics-collector-deployment**에서 사용하는 클라이언트 인증서

CA 인증서는 5년 동안 유효하며 다른 인증서는 1년 동안 유효합니다. 모든 관찰 인증서는 만료 시 자동으로 새로 고쳐집니다.

다음 목록을 보고 인증서가 자동으로 갱신될 때의 영향을 파악합니다.

- 나머지 유효 시간이 73 일을 넘지 않으면 비 CA 인증서가 자동으로 갱신됩니다. 인증서가 갱신되면 관련 배포의 Pod가 자동 다시 시작하여 업데이트된 인증서를 사용합니다.
- 유효한 남은 시간이 1년을 넘지 않으면 CA 인증서가 자동으로 갱신됩니다. 인증서가 갱신되면 이전 CA는 삭제되지 않지만 업데이트된 CA와 공존합니다. 이전 인증서와 갱신된 인증서는 모두 관련 배포에서 사용되며 계속 작동합니다. 이전 CA 인증서는 만료될 때 삭제됩니다.
- 인증서가 갱신되면 hub 클러스터와 관리 클러스터 간의 트래픽이 중단되지 않습니다.

1.1.1.2. Bring Your Own (BYO) Observability 인증 기관 (CA) 인증서

Red Hat Advanced Cluster Management에서 생성한 기본 관찰성 CA 인증서를 사용하지 않으려면 관찰성을 활성화하기 전에 BYO Observability CA 인증서를 사용하도록 선택할 수 있습니다.

1.1.1.2.1. CA 인증서를 생성하는 OpenSSL 명령

관찰 기능에는 두 개의 CA 인증서가 필요합니다. 하나는 서버 측이고 다른 하나는 클라이언트 측용입니다.

- 다음 명령을 사용하여 CA RSA 개인 키를 생성합니다.

```
openssl genrsa -out serverCAKey.pem 2048

openssl genrsa -out clientCAKey.pem 2048
```

- 개인 키를 사용하여 자체 서명된 CA 인증서를 생성합니다. 다음 명령을 실행합니다.

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

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

1.1.1.2.2. BYO 관찰 가능 CA 인증서와 관련된 시크릿 생성

보안을 생성하려면 다음 단계를 완료합니다.

1. 인증서 및 개인 키를 사용하여 **observability-server-ca-certs** 시크릿을 생성합니다. 다음 명령을 실행합니다.

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

2. 인증서 및 개인 키를 사용하여 **observability-client-ca-certs** 시크릿을 생성합니다. 다음 명령을 실행합니다.

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

1.1.1.2.3. alertmanager 경로에 대한 인증서 교체

OpenShift 기본 수신 인증서를 사용하지 않으려면 alertmanager 경로를 업데이트하여 alertmanager 인증서를 교체할 수 있습니다. 다음 단계를 완료합니다.

1. 다음 명령을 사용하여 관찰 가능 인증서를 검사합니다.

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

2. 인증서의 **CN**(Common Name)을 **alertmanager** 로 변경합니다.

3. alertmanager 경로의 호스트 이름으로 **csr.cnf** 구성 파일의 SAN을 변경합니다.

4. **open-cluster-management-observability** 네임스페이스에 다음 두 개의 시크릿을 생성합니다. 다음 명령을 실행합니다.

```
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
```

자세한 내용은 인증서를 생성하는 OpenSSL 명령을 참조하십시오. alertmanager 경로에 대한 기본 자체 서명된 인증서를 복원하려면 관리 인그레스의 기본 자체 서명된 인증서 복원에서 **open-** cluster-management-observability 네임스페이스에서 두 시크릿을 삭제합니다.

1.1.2. Red Hat Advanced Cluster Management 구성 요소 인증서

1.1.2.1. 허브 클러스터 관리 인증서 나열

내부적으로 OpenShift Service ServingECDHE 서비스를 사용하는 허브 클러스터 관리 인증서 목록을 볼 수 있습니다. 다음 명령을 실행하여 인증서를 나열합니다.

```
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
```

참고: 관찰 기능이 활성화된 경우 인증서가 생성되는 추가 네임스페이스가 있습니다.

1.1.2.2. refresh hub 클러스터 관리 인증서

허브 클러스터 관리 인증서 섹션에서 **delete secret** 명령을 실행하여 hub 클러스터 관리 인증서를 새로 고칠 수 있습니다. 새로 고침해야 하는 인증서를 식별하는 경우 인증서와 연결된 보안을 삭제합니다. 예를 들어 다음 명령을 실행하여 시크릿을 삭제할 수 있습니다.

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

참고: 시크릿을 삭제한 후 새 시크릿이 생성됩니다. 그러나 새 인증서를 사용하기 시작할 수 있도록 보안을 사용하는 Pod를 수동으로 다시 시작해야 합니다.

1.1.2.3. Red Hat Advanced Cluster Management Webhook 인증서 새로 고침

Red Hat Advanced Cluster Management Webhook에서 사용하는 인증서인 OpenShift Container Platform 관리 인증서를 새로 고칠 수 있습니다.

Red Hat Advanced Cluster Management 웹 후크 인증서를 새로 고침하려면 다음 단계를 완료합니다.

1. 다음 명령을 실행하여 OpenShift Container Platform 관리 인증서와 연결된 보안을 삭제합니다.

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

참고: 일부 서비스에는 삭제해야 하는 시크릿이 없을 수 있습니다.

2. 다음 명령을 실행하여 OpenShift Container Platform 관리 인증서와 연결된 서비스를 다시 시작합니다.

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

중요: 여러 서비스의 복제본이 있으며 각 서비스를 다시 시작해야 합니다.

다음 표에서 인증서를 포함하는 Pod의 요약 목록과 Pod를 다시 시작하기 전에 보안을 삭제해야 하는지 여부를 확인합니다.

표 1.1. OpenShift Container Platform 관리 인증서가 포함된 Pod

서비스 이름	네임스페이스	Pod 이름 샘플	시크릿 이름(해당되는 경우)
channels-apps-open-cluster-management-webhook-svc	open-cluster-management	multicluster-operators-application-8c446664c-5lbfk	channels-apps-open-cluster-management-webhook-svc-ca
multicluster-operators-application-svc	open-cluster-management	multicluster-operators-application-8c446664c-5lbfk	multicluster-operators-application-svc-ca
cluster-manager-registration-webhook	open-cluster-management-hub	cluster-manager-registration-webhook-fb7b99c-d8wfc	registration-webhook-serving-cert
cluster-manager-work-webhook	open-cluster-management-hub	cluster-manager-work-webhook-89b8d7fc-f4pv8	work-webhook-serving-cert

1.1.3. Red Hat Advanced Cluster Management 관리 인증서

1.1.3.1. 채널 인증서

CA 인증서는 Red Hat Advanced Cluster Management 애플리케이션 관리의 일부인 Git 채널과 연결할 수 있습니다. 자세한 내용은 보안 HTTPS 연결에 사용자 정의 CA 인증서 사용을 참조하십시오.

Helm 채널을 사용하면 인증서 검증을 비활성화할 수 있습니다. 개발 환경에서 인증서 유효성 검사가 비활성화된 Helm 채널을 구성해야 합니다. 인증서 검증을 비활성화하면 보안 위험이 발생합니다.

1.1.3.2. 관리형 클러스터 인증서

인증서는 허브를 사용하여 관리되는 클러스터를 인증하는 데 사용됩니다. 따라서 이러한 인증서와 관련된 문제 해결 시나리오를 알고 있어야 합니다. 자세한 내용은 인증서 변경 후 오프라인으로 가져온 클러스터 문제 해결을 참조하십시오.

관리형 클러스터 인증서가 자동으로 새로 고쳐집니다.

1.1.4. 타사 인증서

1.1.4.1. 게이트키퍼 웹 후크 인증서 교체

게이트키퍼 웹 후크 인증서를 교체하려면 다음 단계를 완료합니다.

1. 다음 명령을 사용하여 인증서가 포함된 보안을 편집합니다.

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

2. **data** 섹션에서 **ca.crt**,**ca.key**, tls.crt' 및 **tls.key** 를 삭제합니다.

3. 다음 명령을 사용하여 **gatekeeper-controller-manager** Pod를 삭제하여 게이트키퍼 웹 후크 서비스를 다시 시작합니다.

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

게이트키퍼 웹 후크 인증서가 교체되었습니다.

1.1.4.2. 무결성 방전 Webhook 인증서 교체 (기술 프리뷰)

무결성 방패 웹 후크 인증서를 교체하려면 다음 단계를 완료합니다.

1. IntegrityShield 사용자 정의 리소스를 편집하고 **integrity- greld-operator-system** 네임스페이스를 **inScopeNamespaceSelector** 설정에서 제외된 네임스페이스 목록에 추가합니다. 다음 명령을 실행하여 리소스를 편집합니다.

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

2. 다음 명령을 실행하여 무결성 분리 인증서가 포함된 보안을 삭제합니다.

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

3. 시크릿이 다시 생성되도록 Operator를 삭제합니다. Operator Pod 이름이 시스템의 Pod 이름과 일치하는지 확인합니다. 다음 명령을 실행합니다.

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

4. 무결성 분리 서버 Pod를 삭제하여 다음 명령으로 새 인증서 사용을 시작합니다.

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

인증서 정책 컨트롤러를 사용하여 관리 클러스터에서 인증서 정책을 생성하고 관리합니다. 컨트롤러에 대한 자세한 내용은 정책 컨트롤러를 참조하십시오. 자세한 내용은 위험 및 규정 준수 페이지로 돌아갑니다.

1.2. 관리 수신 인증서 교체

OpenShift 기본 수신 인증서를 사용하지 않으려면 Kubernetes 경로를 업데이트하여 관리 수신 인증서를 교체할 수 있습니다.

- 관리 수신 인증서를 교체하기 위한 사전 요구 사항
- BYO(Bring Your Own) 수신 인증서 교체
- 관리 수신을 위한 기본 자체 서명 인증서 복원

1.2.1. 관리 수신 인증서를 교체하기 위한 사전 요구 사항

management-ingress 인증서 및 개인 키를 준비하고 준비합니다. 필요한 경우 OpenSSL을 사용하여 TLS 인증서를 생성할 수 있습니다. 인증서의**CN**(Common Name parameter)을 **manangement-ingress** 로 설정합니다. 인증서를 생성하는 경우 다음 설정을 포함합니다.

- 인증서 SAN(Subject Alternative Name) 목록에 Kubernetes용 Red Hat Advanced Cluster Management의 경로 이름을 도메인 이름으로 포함합니다.
  다음 명령을 실행하여 경로 이름을 수신합니다.

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

  다음과 같은 응답을 받을 수 있습니다.

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

1.2.1.1. 인증서를 생성하기 위한 설정 파일 예

다음 예제 설정 파일 및 OpenSSL 명령은 OpenSSL을 사용하여 TLS 인증서를 생성하는 방법에 대한 예제를 제공합니다. OpenSSL로 인증서를 생성하기 위한 구성 설정을 정의하는 다음 **csr.cnf** 구성 파일을 봅니다.

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

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

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

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

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

참고: DNS.1 이라는 레이블이 지정된 SAN을 관리 인그레스에 올바른 호스트 이름으로 업데이트하십시오.

1.2.1.2. 인증서를 생성하기 위한 OpenSSL 명령

다음 OpenSSL 명령을 이전 구성 파일과 함께 사용하여 필요한 TLS 인증서를 생성합니다.

1. CA(인증 기관) RSA 개인 키를 생성합니다.

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

2. CA 키를 사용하여 자체 서명된 CA 인증서를 생성합니다.

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

3. 인증서에 사용할 RSA 개인 키를 생성합니다.

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

4. 개인 키를 사용하여 인증서 서명 요청(CSR)을 생성합니다.

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

5. CA 인증서 및 키와 CSR을 사용하여 서명된 인증서를 생성합니다.

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

6. 인증서 내용을 검사합니다.

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

1.2.2. BYO(Bring Your Own) 수신 인증서 교체

BYO 수신 인증서를 교체하려면 다음 단계를 완료합니다.

1. 인증서 및 개인 키를 사용하여 **byo-ingress-tls** 시크릿을 생성합니다. 다음 명령을 실행합니다.

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

2. 다음 명령을 사용하여 보안이 올바른 네임스페이스에 생성되었는지 확인합니다.

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

3. 선택 사항: 다음 명령을 실행하여 CA 인증서가 포함된 보안을 생성합니다.

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

4. 서브스크립션을 재배포하려면 **management-ingress** 서브스크립션을 삭제합니다. 이전 단계에서 생성된 보안은 자동으로 사용됩니다. 다음 명령을 실행합니다.

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

5. 현재 인증서가 인증서이고 모든 콘솔 액세스 및 로그인 기능이 동일하게 유지되는지 확인합니다.

1.2.3. 관리 수신을 위한 기본 자체 서명 인증서 복원

1. 다음 명령을 사용하여 자체 인증서 보안 가져오기를 삭제합니다.

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

2. 서브스크립션을 재배포하려면 **management-ingress** 서브스크립션을 삭제합니다. 이전 단계에서 생성된 보안은 자동으로 사용됩니다. 다음 명령을 실행합니다.

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

3. 현재 인증서가 인증서이고 모든 콘솔 액세스 및 로그인 기능이 동일하게 유지되는지 확인합니다.

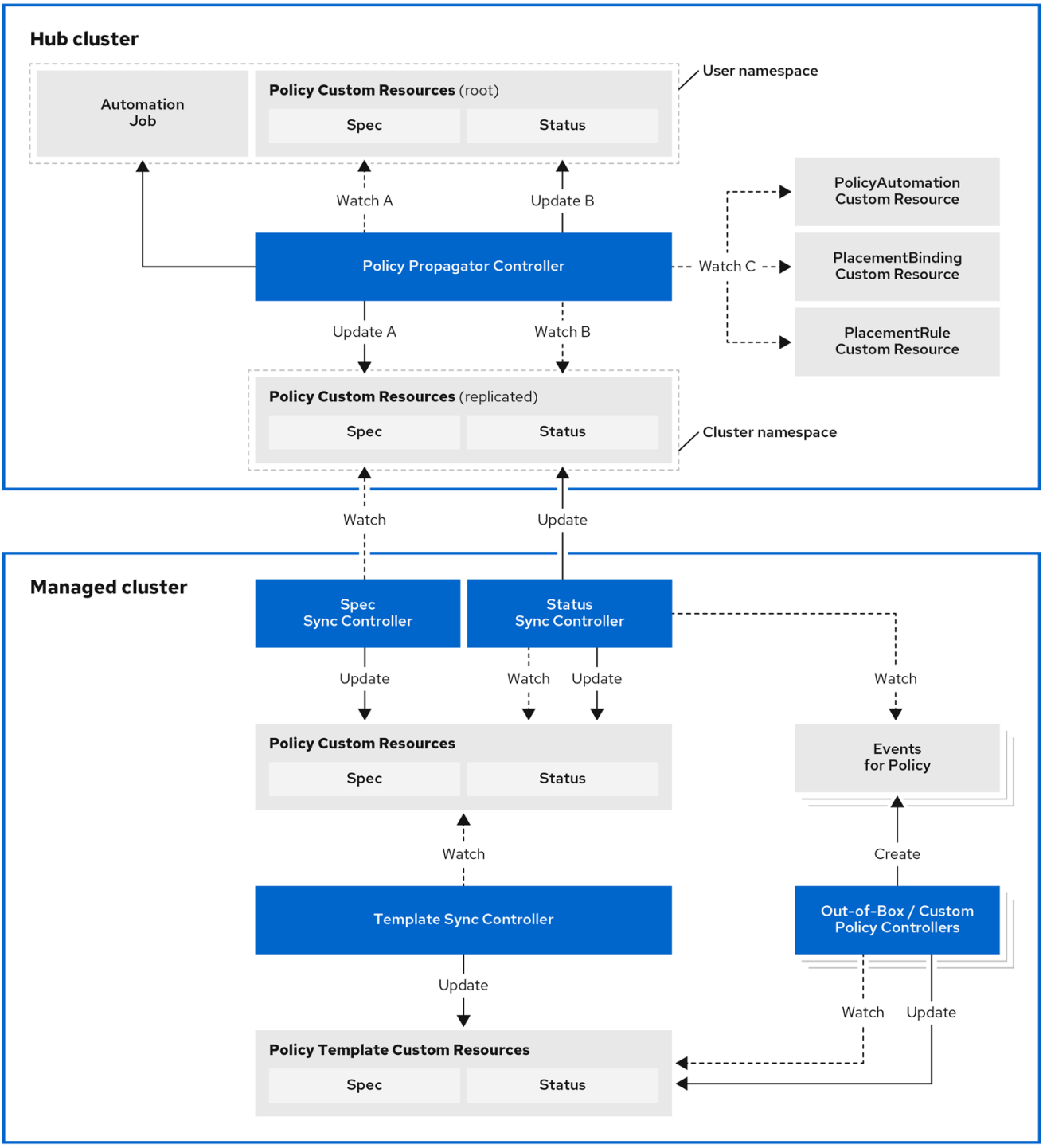
Red Hat Advanced Cluster Management에서 생성 및 관리하는 인증서에 대한 자세한 내용은 ScanSetting을 참조하십시오. ??? 클러스터 보안에 대한 자세한 내용은 위험 및 규정 준수 페이지로 돌아갑니다.

2장. 거버넌스

기업은 프라이빗, 멀티 및 하이브리드 클라우드에서 호스팅되는 워크로드에 대한 소프트웨어 엔지니어링, 보안 엔지니어링, 복원력, 보안 및 규정 준수를 위한 내부 표준을 충족해야 합니다. Red Hat Advanced Cluster Management for Kubernetes 거버넌스는 기업이 자체 보안 정책을 도입할 수 있는 확장 가능한 정책 프레임워크를 제공합니다.

2.1. 거버넌스 아키텍처

Red Hat Advanced Cluster Management for Kubernetes 거버넌스 라이프사이클을 사용하여 클러스터의 보안을 강화합니다. 제품 거버넌스 라이프사이클은 정의된 정책, 프로세스 및 절차를 기반으로 중앙 인터페이스 페이지에서 보안 및 규정 준수를 관리합니다. 거버넌스 아키텍처의 다음 다이어그램을 확인합니다.



186_RHACM_1221

거버넌스 아키텍처는 다음 구성 요소로 구성됩니다.

- **거버넌스 대시보드**: 정책 및 클러스터 위반을 포함하는 클라우드 거버넌스 및 위험 세부 정보에 대한 요약을 제공합니다.
  참고:
  - 정책을 관리 클러스터에 전파하는 경우 복제된 정책의 이름은 **namespaceName.policyName** 입니다. 정책을 생성할 때 오브젝트 이름의 Kubernetes 제한으로 인해 **namespaceName.policyName** 의 길이가 63자를 초과하지 않아야 합니다.
  - hub 클러스터에서 정책을 검색하는 경우 관리 클러스터에서 복제된 정책의 이름을 수신할 수도 있습니다. 예를 들어 **policy-dhaz-cert** 를 검색하는 경우 hub 클러스터의 다음 정책 이름이 **default.policy-dhaz-cert** 가 표시될 수 있습니다.

- **정책 기반 거버넌스 프레임워크:** 지리적 리전과 같은 클러스터와 관련된 속성을 기반으로 다양한 관리 클러스터에 정책 생성 및 배포를 지원합니다. 사전 정의된 정책의 예와 클러스터에 정책을 배포하는 방법에 대한 지침을 보려면**policy-collection** 리포지터리 를 참조하십시오. 사용자 정의 정책 컨트롤러 및 정책을 제공할 수도 있습니다. 정책이 위반되면 자동화를 실행하고 사용자가 선택하는 모든 작업을 수행하도록 구성할 수 있습니다. 자세한 내용은 Configuring Ansible Tower for governance 를 참조하십시오.
  policy_governance_info 지표를 사용하여 추세를 보고 정책 실패를 분석합니다. 자세한 내용은 Governance 메트릭 에서 참조하십시오.

- **정책 컨트롤러**: 관리 대상 클러스터에 대한 하나 이상의 정책을 지정된 제어에 대해 평가하고 위반에 대한 Kubernetes 이벤트를 생성합니다. 허브 클러스터로 위반이 전파됩니다. 설치에 포함된 정책 컨트롤러는 Kubernetes 구성, 인증서 및 IAM입니다. 사용자 정의 정책 컨트롤러를 생성할 수도 있습니다.

- **오픈 소스 커뮤니티**: Red Hat Advanced Cluster Management 정책 프레임워크를 기반으로 커뮤니티 기여를 지원합니다. 정책 컨트롤러 및 타사 정책은 **stolostron/policy-collection** 리포지토리의 일부입니다. GitOps를 사용하여 정책을 기여하고 배포하는 방법을 알아봅니다. 자세한 내용은 GitOps를 사용하여 정책 배포를 참조하십시오. 타사 정책을 Red Hat Advanced Cluster Management for Kubernetes와 통합하는 방법을 알아보십시오. 자세한 내용은 Integrate third-party policy controllers 를 참조하십시오.

Kubernetes 정책 프레임워크용 Red Hat Advanced Cluster Management 구조 및 Kubernetes 거버넌스용 Red Hat Advanced Cluster Management 사용 방법에 *대해* 알아보십시오.

- 정책 개요
- 정책 컨트롤러
- 지원되는 정책
- 보안 정책 관리
- hub 클러스터 보안

2.2. 정책 개요

Red Hat Advanced Cluster Management for Kubernetes 보안 정책 프레임워크를 사용하여 사용자 지정 정책 컨트롤러 및 기타 정책을 생성합니다. Kubernetes CRD(사용자 정의 리소스 정의) 인스턴스는 정책을 생성하는 데 사용됩니다. CRD에 대한 자세한 내용은 CustomResourceDefinitions를 사용하여 Kubernetes API 확장을 참조하십시오.

Kubernetes용 각 Red Hat Advanced Cluster Management 정책에는 하나 이상의 템플릿이 있을 수 있습니다. 정책 요소에 대한 자세한 내용은 이 페이지의 다음 *정책 YAML 테이블* 섹션을 참조하십시오.

정책에는 정책 문서가 적용되는 클러스터를 정의하는 *Placement Rule* 또는 Placement와 Red Hat Advanced Cluster Management for Kubernetes 정책을 배치 규칙에 바인딩하는 *PlacementBinding* 이 필요합니다. **PlacementRule** 을 정의하는 방법에 대한 자세한 내용은 애플리케이션 라이프사이클 설명서의 배치 규칙을 참조하십시오. 배치를 정의하는 방법에 대한 자세한 내용은 클러스터 라이프사이클 설명서의 배치 개요 를 참조하십시오.

중요:

- PlacementBinding 을 생성하여 PlacementRule 또는 Placement 와 연결해야 합니다.

  모범 사례: 배치 리소스를 사용할 때 CLI(명령줄 인터페이스)를 사용하여 정책을 업데이트합니다.

- 클러스터 네임스페이스를 제외하고 hub 클러스터의 모든 네임스페이스에 정책을 생성할 수 있습니다. 클러스터 네임스페이스에서 정책을 생성하면 Red Hat Advanced Cluster Management for Kubernetes에서 삭제됩니다.

- 각 클라이언트 및 공급자는 관리 클라우드 환경이 소프트웨어 엔지니어링, 보안 엔지니어링, 복원력, 보안 및 Kubernetes 클러스터에서 호스팅되는 워크로드에 대한 규정 준수의 내부 엔터프라이즈 보안 표준을 충족하도록 해야 합니다. 거버넌스 및 보안 기능을 사용하여 표준을 충족하기 위해 가시성을 확보하고 구성을 수정하십시오.

다음 섹션의 정책 구성 요소에 대해 자세히 알아보십시오.

- 정책 YAML 구조
- 정책 YAML 테이블
- 정책 샘플 파일
- YAML 샘플 파일 배치

2.2.1. 정책 YAML 구조

정책을 생성할 때 필수 매개변수 필드 및 값을 포함해야 합니다. 정책 컨트롤러에 따라 다른 선택적 필드 및 값을 포함해야 할 수 있습니다. 설명된 매개변수 필드의 다음 YAML 구조를 확인합니다.

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

```
spec:
  policy-templates:
    - objectDefinition:
        apiVersion:
        kind:
        metadata:
          name:
        spec:
  remediationAction:
  disabled:

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

2.2.2. 정책 YAML 테이블

필드	설명
apiVersion	필수 항목입니다. 값을 **policy.open-cluster-management.io/v1** 로 설정합니다.
kind	필수 항목입니다. 정책 유형을 나타내려면 값을 **Policy** 로 설정합니다.
metadata.name	필수 항목입니다. 정책 리소스를 식별하는 이름입니다.

필드	설명
metadata.annotations	선택 사항: 정책에서 검증하려는 표준 집합을 설명하는 보안 세부 정보 집합을 지정하는 데 사용됩니다. 여기에 설명된 모든 주석은 쉼표로 구분된 목록이 포함된 문자열로 표시됩니다. **참고**: 콘솔에서 정책 페이지에서 정책에 대해 정의한 표준 및 카테고리를 기반으로 *정책* 위반을 볼 수 있습니다.
annotations.policy.open-cluster-management.io/standards	정책이 관련된 보안 표준의 이름 또는 이름입니다. 예를 들어, National Quarkus of Standards and Technology (NIST) 및Pay Card Industry(Payment Card Industry) 가 있습니다.
annotations.policy.open-cluster-management.io/categories	보안 제어 카테고리는 하나 이상의 표준에 대한 특정 요구 사항을 나타냅니다. 예를 들어, 시스템 및 정보 무결성 카테고리는 귀하의 정책에ECDHE 및 PCI 표준에 따라 개인 정보를 보호하기 위한 데이터 전송 프로토콜이 포함되어 있음을 나타낼 수 있습니다.
annotations.policy.open-cluster-management.io/controls	확인 중인 보안 컨트롤의 이름입니다. 예를 들어 인증서 정책 컨트롤러입니다.
spec.policy-templates	필수 항목입니다. 관리형 클러스터에 적용할 하나 이상의 정책을 생성하는 데 사용됩니다.
spec.disabled	필수 항목입니다. 값을 **true** 또는 **false** 로 설정합니다. **disabled** 매개변수는 정책을 활성화하고 비활성화하는 기능을 제공합니다.
spec.remediationAction	선택 사항: 정책 수정을 지정합니다. 매개변수 값은 **강제 시행** 및 **정보 입니다**. 지정된 경우 정의된 **spec.remediationAction** 값이 **policy-templates** 섹션에서 하위 정책에 정의된 **remediationAction** 매개변수를 덮어씁니다. 예를 들어 **spec.remediationAction** value 섹션이 **강제 적용** 되도록 설정된 경우 **policy-templates** 섹션의 **remediationAction** 이 런타임 중에 **강제 적용** 되도록 설정됩니다. **중요**: 일부 정책은 적용 기능을 지원하지 않을 수 있습니다.

2.2.3. 정책 샘플 파일

```
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  name: policy-role
  annotations:
    policy.open-cluster-management.io/standards: NIST SP 800-53
    policy.open-cluster-management.io/categories: AC Access Control
    policy.open-cluster-management.io/controls: AC-3 Access Enforcement
```

```
spec:
  remediationAction: inform
  disabled: false
  policy-templates:
    - objectDefinition:
        apiVersion: policy.open-cluster-management.io/v1
        kind: ConfigurationPolicy
        metadata:
          name: policy-role-example
        spec:
          remediationAction: inform # the policy-template spec.remediationAction is overridden
by the preceding parameter value for spec.remediationAction.
          severity: high
          namespaceSelector:
            include: ["default"]
          object-templates:
            - complianceType: mustonlyhave # role definition should exact match
              objectDefinition:
                apiVersion: rbac.authorization.k8s.io/v1
                kind: Role
                metadata:
                  name: sample-role
                rules:
                  - apiGroups: ["extensions", "apps"]
                    resources: ["deployments"]
                    verbs: ["get", "list", "watch", "delete","patch"]
---
apiVersion: policy.open-cluster-management.io/v1
kind: PlacementBinding
metadata:
  name: binding-policy-role
placementRef:
  name: placement-policy-role
  kind: PlacementRule
  apiGroup: apps.open-cluster-management.io
subjects:
- name: policy-role
  kind: Policy
  apiGroup: policy.open-cluster-management.io
---
apiVersion: apps.open-cluster-management.io/v1
kind: PlacementRule
metadata:
  name: placement-policy-role
spec:
  clusterConditions:
  - status: "True"
    type: ManagedClusterConditionAvailable
  clusterSelector:
    matchExpressions:
      - {key: environment, operator: In, values: ["dev"]}
```

2.2.4. YAML 샘플 파일 배치

PlacementBinding 및 Placement 리소스를 이전 정책 예제와 결합하여 PlacementRule API 대신 클러스터 배치 API를 사용하여 정책을 배포할 수 있습니다.

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

정책을 만들고 업데이트하려면 보안 정책 관리를 참조하십시오. Red Hat Advanced Cluster Management 정책 컨트롤러를 활성화 및 업데이트하여 정책 준수를 검증할 수도 있습니다. 정책 컨트롤러를 참조하십시오.

자세한 정책 주제를 알아보려면 거버넌스를 참조하십시오.

2.3. 정책 컨트롤러

정책 컨트롤러는 클러스터가 정책과 호환되는지 여부를 모니터링하고 보고합니다. 즉시 사용 가능한 정책 템플릿을 사용하여 사전 정의된 정책 컨트롤러 및 정책을 적용하여 Red Hat Advanced Cluster Management for Kubernetes 정책 프레임워크를 사용합니다. 정책 컨트롤러는 Kubernetes CRD(사용자 정의 리소스 정의) 인스턴스입니다.

CRD에 대한 자세한 내용은 CustomResourceDefinitions를 사용하여 Kubernetes API 확장을 참조하십시오. 정책 컨트롤러는 정책 위반을 수정하여 클러스터 상태를 준수하도록 합니다.

제품 정책 프레임워크를 사용하여 사용자 지정 정책 및 정책 컨트롤러를 생성할 수 있습니다. 자세한 내용은 사용자 정의 정책 컨트롤러 생성(더 이상 사용되지 않음) 을 참조하십시오.

다음 주제를 참조하여 **Kubernetes** 정책 컨트롤러의 **Red Hat Advanced Cluster Management for Kubernetes**에 대해 자세히 알아보십시오.

- Kubernetes 구성 정책 컨트롤러
- 인증서 정책 컨트롤러
- IAM 정책 컨트롤러
- 정책 세트 컨트롤러

중요: 구성 정책 컨트롤러 정책만 적용 기능을 지원합니다. 정책 컨트롤러에서 시행 기능을 지원하지 않는 정책을 수동으로 수정해야 합니다.

정책 관리에 대한 자세한 내용은 관리 정책을 참조하십시오.

2.3.1. Kubernetes 구성 정책 컨트롤러

구성 정책 컨트롤러를 사용하여 **Kubernetes** 리소스를 구성하고 클러스터 전체에 보안 정책을 적용할 수 있습니다.

구성 정책 컨트롤러는 로컬 **Kubernetes API** 서버와 통신하여 클러스터에 있는 구성 목록을 가져옵니다. **CRD**에 대한 자세한 내용은 **CustomResourceDefinitions**를 사용하여 **Kubernetes API** 확장을 참조하십시오.

구성 정책 컨트롤러는 설치 중에 **hub** 클러스터에 생성됩니다. 구성 정책 컨트롤러는 적용 기능을 지원하고 다음 정책의 준수를 모니터링합니다.

- 메모리 사용량 정책

- 네임스페이스 정책
- 이미지 취약점 정책
- Pod 정책
- Pod 보안 정책
- 역할 정책
- 역할 바인딩 정책
- SCC(보안 콘텐츠 제약 조건) 정책
- ETCD 암호화 정책
- 컴플라이언스 Operator 정책
- 게이트키퍼 제약 조건 및 제약 조건 템플릿 통합

구성 정책의 수정 사항이 적용 되도록 설정되면 컨트롤러에서 대상 관리 클러스터에 복제 정책을 생성합니다. 구성 정책에서 템플릿을 사용할 수도 있습니다. 자세한 내용은 구성 정책의 템플릿 지원을 참조하십시오.

구성 정책 컨트롤러에 대한 자세한 내용을 보려면 계속 읽으십시오.

- 구성 정책 컨트롤러 YAML 구조
- 구성 정책 샘플

- 구성 정책 YAML 테이블

2.3.1.1. 구성 정책 컨트롤러 YAML 구조

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

2.3.1.2. 구성 정책 샘플

```
apiVersion: policy.open-cluster-management.io/v1
kind: ConfigurationPolicy
metadata:
  name: policy-config
spec:
  namespaceSelector:
    include: ["default"]
    exclude: []
  remediationAction: inform
    severity: low
    object-templates:
    - complianceType: musthave
      objectDefinition:
        apiVersion: v1
        kind: Pod
        metadata:
          name: pod
        spec:
          containers:
          - image: 'pod-image'
            name:
            ports:
           - containerPort: 80
```

```
evaluationInterval:
  compliant:
  noncompliant:
```

2.3.1.3. 구성 정책 YAML 테이블

표 2.1. 매개변수 테이블

필드	설명
apiVersion	필수 항목입니다. 값을 **policy.open-cluster-management.io/v1** 로 설정합니다.
kind	필수 항목입니다. 정책 유형을 나타내려면 **ConfigurationPolicy** 로 값을 설정합니다.
metadata.name	필수 항목입니다. 정책의 이름입니다.
spec	필수 항목입니다. 모니터링할 구성 정책과 이를 수정하는 방법에 대한 사양입니다.
spec.namespace	네임스페이스가 지정된 오브젝트 또는 리소스에 필요합니다. 정책이 적용되는 허브 클러스터의 네임스페이스입니다. 정책에 적용할 네임스페이스인 **include** 매개변수에 하나 이상의 네임스페이스를 입력합니다. **exclude** 매개변수는 정책을 명시적으로 적용하지 않으려는 네임스페이스를 지정합니다.
spec.remediationAction	필수 항목입니다. 정책 수정을 지정합니다. **정보**입력
spec.remediationAction.severity	필수 항목입니다. 정책을 준수하지 않는 경우 심각도를 지정합니다. 다음 매개변수 값을 사용합니다(**낮음**,**중간** 또는 **높음**).
spec.remediationAction.complianceType	필수 항목입니다. 관리 클러스터에 평가하거나 적용해야 하는 역할 및 기타 Kubernetes 오브젝트의 예상 동작을 나열하는 데 사용됩니다. 다음 동사를 매개변수 값으로 사용해야 합니다. **mustonlyhave**: 정확한 이름과 관련 필드가 있는 오브젝트가 있어야 함을 나타냅니다. **musthave**: 지정된 object-template과 동일한 이름의 오브젝트가 있어야 함을 나타냅니다. 템플릿의 다른 필드는 오브젝트에 존재하는 항목의 하위 집합입니다. **mustnothave**: 이름 또는 레이블이 같은 객체가 존재할 수 없으며 사양이나 규칙에 관계없이 삭제해야 함을 나타냅니다.

필드	설명
spec.evaluationInterval.compliant	선택 사항: 규정 준수 상태에 있을 때 정책을 평가하는 빈도를 정의하는 데 사용됩니다. 값은 시간 단위 접미사가 있는 일련의 숫자인 기간 형식이어야 합니다. 예를 들어 **12h30m5s** 는 12 시간, 30 분 및 5 초를 나타냅니다. 정책 **사양** 을 업데이트하지 않는 한 규정 준수 클러스터에서 정책을 다시 평가하지 않도록 **never** 로 설정할 수도 있습니다.
spec.evaluationInterval.noncompliant	선택 사항: 비호환 상태의 경우 정책을 평가하는 빈도를 정의하는 데 사용됩니다. **evaluationInterval.compliant** 매개변수와 유사하게 값은 시간 단위 접미사가 있는 숫자 시퀀스인 기간의 형식이어야 합니다. 정책 **사양** 을 업데이트하지 않는 한 정책이 비준수 클러스터에서 다시 평가되지 않도록 **never** 로 설정할 수도 있습니다.

NIST Special>-< 800-53 (Rev. 4)을 사용하고 CM-Configuration-Management 폴더의 Red Hat Advanced Cluster Management에서 지원하는 정책 샘플을 참조하십시오. hub 클러스터에 정책을 적용하는 방법에 대한 자세한 내용은 지원 정책을 참조하십시오.

정책 만들기 및 사용자 지정 방법에 대한 자세한 내용은 보안 정책 관리를 참조하십시오. 컨트롤러에 대한 자세한 내용은 정책 컨트롤러를 참조하십시오.

2.3.2. 인증서 정책 컨트롤러

인증서 정책 컨트롤러를 사용하면 만료에 가까운 인증서를 감지하고 너무 길거나 지정된 패턴과 일치하지 않는 DNS 이름이 포함된 시간 기간(시간)을 감지할 수 있습니다.

컨트롤러 정책에서 다음 매개변수를 업데이트하여 인증서 정책 컨트롤러를 구성하고 사용자 지정합니다.

- **minimumDuration**
- **minimumCADuration**
- **maximumDuration**

- maximumCADuration
- allowedSANPattern
- disallowedSANPattern

다음 시나리오 중 하나로 인해 정책을 준수하지 않을 수 있습니다.

- 인증서가 최소 시간 동안 만료되거나 최대 시간을 초과하는 경우
- DNS 이름이 지정된 패턴과 일치하지 않는 경우

인증서 정책 컨트롤러는 관리 클러스터에서 생성됩니다. 컨트롤러는 로컬 Kubernetes API 서버와 통신하여 인증서가 포함된 시크릿 목록을 가져오고 모든 호환 인증서를 결정합니다. CRD에 대한 자세한 내용은 CustomResourceDefinitions를 사용하여 Kubernetes API 확장을 참조하십시오.

인증서 정책 컨트롤러에서 시행 기능을 지원하지 않습니다.

2.3.2.1. 인증서 정책 컨트롤러 YAML 구조

인증서 정책의 다음 예제를 보고 YAML 테이블의 요소를 검토합니다.

```
apiVersion: policy.open-cluster-management.io/v1
kind: CertificatePolicy
metadata:
  name: certificate-policy-example
  namespace:
  labels: category=system-and-information-integrity
spec:
  namespaceSelector:
    include: ["default"]
  remediationAction:
  severity:
  minimumDuration:
  minimumCADuration:
  maximumDuration:
```

```
maximumCADuration:
allowedSANPattern:
disallowedSANPattern:
```

2.3.2.1.1. 인증서 정책 컨트롤러 YAML 테이블

표 2.2. 매개변수 테이블

필드	설명
apiVersion	필수 항목입니다. 값을 **policy.open-cluster-management.io/v1** 로 설정합니다.
kind	필수 항목입니다. 정책 유형을 나타내려면 값을 **CertificatePolicy** 로 설정합니다.
metadata.name	필수 항목입니다. 정책을 식별하는 이름입니다.
metadata.namespace	필수 항목입니다. 정책이 생성되는 관리형 클러스터 내의 네임스페이스입니다.
metadata.labels	선택 사항: 인증서 정책에서 **category=system-and-integrity** 레이블은 정책을 분류하고 인증서 정책을 쉽게 쿼리할 수 있습니다. 인증서 정책의 **category** 키에 대한 다른 값이 있는 경우 인증서 컨트롤러에서 값을 덮어씁니다.
spec	필수 항목입니다. 모니터링 및 새로 고칠 인증서의 사양입니다.
spec.namespaceSelector	필수 항목입니다. 정책을 적용하려는 관리형 클러스터 네임스페이스입니다. **Include** 및 **Exclude** 의 매개변수 값을 입력합니다. **참고**: • 여러 인증서 정책을 생성하여 동일한 관리 클러스터에 적용하는 경우 각 정책 **namespaceSelector** 에 다른 값을 할당해야 합니다. • 인증서 정책 컨트롤러의 **namespaceSelector** 가 네임스페이스와 일치하지 않으면 정책이 준수된 것으로 간주됩니다.
spec.remediationAction	필수 항목입니다. 정책 수정을 지정합니다. **알릴** 매개변수 값을 설정합니다. 인증서 정책 컨트롤러는 **정보 기능만** 지원합니다.
spec.severity	선택 사항: 정책을 준수하지 않는 경우 사용자에게 심각도를 알립니다. 다음 매개변수 값을 사용합니다(**낮음**,**중간** 또는 **높음**).

필드	설명
spec.minimumDuration	필수 항목입니다. 값을 지정하지 않으면 기본값은 **100h** 입니다. 이 매개변수는 인증서를 준수하지 않는 것으로 간주하기 전에 최소 기간(시간)을 지정합니다. 매개변수 값은 Golang의 시간 기간 형식을 사용합니다. 자세한 내용은 Golang Parse Duration 을 참조하십시오.
spec.minimumCADuration	선택 사항: 다른 인증서와 다른 값으로 만료될 수 있는 서명 인증서를 식별하는 값을 설정합니다. 매개변수 값을 지정하지 않으면 CA 인증서 만료는 **minimumDuration** 에 사용되는 값입니다. 자세한 내용은 Golang Parse Duration 을 참조하십시오.
spec.maximumDuration	선택 사항: 원하는 제한을 초과하는 기간으로 생성된 인증서를 식별하려면 값을 설정합니다. 이 매개변수는 Golang의 시간 기간 형식을 사용합니다. 자세한 내용은 Golang Parse Duration 을 참조하십시오.
spec.maximumCADuration	선택 사항: 정의된 제한을 초과하는 기간으로 생성된 서명 인증서를 식별하려면 값을 설정합니다. 이 매개변수는 Golang의 시간 기간 형식을 사용합니다. 자세한 내용은 Golang Parse Duration 을 참조하십시오.
spec.allowedSANPattern	선택 사항: 인증서에 정의한 모든 SAN 항목과 일치해야 하는 정규식입니다. 이 매개변수는 DNS 이름을 패턴에 대해 확인합니다. 자세한 내용은 Golang Regular Expression 구문 을 참조하십시오.
spec.disallowedSANPattern	선택 사항: 인증서에 정의한 SAN 항목과 일치하지 않아야 하는 정규식입니다. 이 매개변수는 DNS 이름을 패턴과 대조하여 확인합니다. **참고**: 와일드카드 인증서를 감지하려면 다음 SAN 패턴을 사용합니다. **허용하지 않는SANPattern: "[*]"** 자세한 내용은 Golang Regular Expression 구문 을 참조하십시오.

2.3.2.2. 인증서 정책 샘플

hub 클러스터에 인증서 정책 컨트롤러가 생성되면 관리 클러스터에 복제 정책이 생성됩니다. 인증서 정책 샘플을 보려면 policy-certificate.yaml 을 참조하십시오.

인증서 정책을 관리하는 방법에 대한 자세한 내용은 보안 정책 관리를 참조하십시오. 자세한 내용은 정책 컨트롤러를 참조하십시오.

2.3.3. IAM 정책 컨트롤러

IAM(Identity and Access Management) 정책 컨트롤러를 사용하여 호환되지 않는 IAM 정책에 대한 알림을 수신할 수 있습니다. 규정 준수 검사는 IAM 정책에서 구성한 매개변수를 기반으로 합니다.

IAM 정책 컨트롤러는 클러스터에서 특정 클러스터 역할(예: ClusterRole)이 있는 원하는 최대 사용자 수를 모니터링합니다. 모니터링할 기본 클러스터 역할은 cluster-admin 입니다. IAM 정책 컨트롤러는 로컬 Kubernetes API 서버와 통신합니다. 자세한 내용은 CustomResourceDefinitions를 사용하여 Kubernetes API 확장을 참조하십시오.

IAM 정책 컨트롤러는 관리 클러스터에서 실행됩니다. 자세한 내용은 다음 섹션을 참조하십시오.

- IAM 정책 YAML 구조
- IAM 정책 YAML 테이블
- IAM 정책 샘플

2.3.3.1. IAM 정책 YAML 구조

IAM 정책의 다음 예제를 보고 YAML 테이블의 매개변수를 검토합니다.

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

2.3.3.2. IAM 정책 YAML 테이블

설명은 다음 매개변수 표를 참조하십시오.

표 2.3. 매개변수 테이블

필드	설명
apiVersion	필수 항목입니다. 값을 **policy.open-cluster-management.io/v1** 로 설정합니다.
kind	필수 항목입니다. 정책 유형을 나타내려면 값을 **Policy** 로 설정합니다.
metadata.name	필수 항목입니다. 정책 리소스를 식별하는 이름입니다.
spec	필수 항목입니다. 정책에 대한 설정 세부 정보를 추가합니다.
spec.clusterRole	선택 사항: 모니터링할 클러스터 역할(예: **ClusterRole**)입니다. 지정하지 않는 경우 기본값은 **cluster-admin** 입니다.
spec.severity	선택 사항: 정책을 준수하지 않는 경우 사용자에게 심각도를 알립니다. 다음 매개변수 값을 사용합니다(**낮음**,**중간** 또는 **높음**).
spec.remediationAction	선택 사항: 정책 수정을 지정합니다. 정보 **.**
spec.ignoreClusterRoleBindings	선택 사항: 무시할 클러스터 역할 바인딩 이름을 나타내는 정규식(regex) 값 목록입니다. 이러한 정규식 값은 Go regexp 구문 을 따라야 합니다. 기본적으로 **system:** 로 시작하는 이름이 있는 모든 클러스터 역할 바인딩은 무시됩니다. 이를 더 엄격한 값으로 설정하는 것이 좋습니다. 클러스터 역할 바인딩 이름을 무시하지 않으려면 목록을 단일 값 **.^** 또는 일치하지 않는 다른 정규식으로 설정합니다.
spec.maxClusterRoleBindingUsers	필수 항목입니다. 정책을 준수하지 않는 것으로 간주하기 전에 사용할 수 있는 최대 IAM 역할 바인딩 수입니다.

2.3.3.3. IAM 정책 샘플

IAM 정책 샘플을 보려면 policy-limitclusteradmin.yaml 을 참조하십시오. 자세한 내용은 보안 정책 관리를 참조하십시오.

자세한 내용은 정책 컨트롤러를 참조하십시오.

2.3.4. 정책 세트 컨트롤러

정책 세트 컨트롤러는 동일한 네임스페이스에 정의된 정책으로 정책 상태 범위를 집계합니다. 동일한 네임스페이스에 있는 정책을 그룹화할 정책 세트(PolicySet)를 생성합니다. PolicySet의 모든 정책은 PolicySet 및 Placement 를 바인딩하기 위해 PlacementBinding 을 생성하여 선택한 클러스터에 함께 배치됩니다. 정책 세트가 hub 클러스터에 배포됩니다.

또한 정책이 여러 정책 세트의 일부인 경우 기존 및 새 배치 리소스는 정책에 남아 있습니다. 사용자가 정책 세트에서 정책을 제거하면 정책 세트에서 선택된 클러스터에 정책이 적용되지 않지만 배치는 그대로 유지됩니다. 정책 세트 컨트롤러는 정책 세트 배치를 포함하는 클러스터의 위반 사항만 확인합니다.

참고: Red Hat Advanced Cluster Management 강화 샘플 정책 세트에서는 클러스터 배치를 사용합니다. 클러스터 배치를 사용하는 경우 정책이 포함된 네임스페이스를 관리 클러스터 세트에 바인딩합니다. 클러스터 배치 사용에 대한 자세한 내용은 클러스터에 정책 배포를 참조하십시오.

다음 섹션에서 정책 집합 구조에 대해 자세히 알아보십시오.

- 정책 세트 컨트롤러 YAML 구조
- 정책 세트 컨트롤러 YAML 테이블
- 정책 세트 샘플

2.3.4.1. 정책 세트 YAML 구조

정책 세트는 다음 YAML 파일과 유사합니다.

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

---
apiVersion: policy.open-cluster-management.io/v1
kind: PlacementBinding
metadata:
  name: demo-policyset-pb
```

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

2.3.4.2. 정책 세트 테이블

설명은 다음 매개변수 표를 참조하십시오.

표 2.4. 매개변수 테이블

필드	설명
apiVersion	필수 항목입니다. 해당 값을 **policy.open-cluster-management.io/v1beta1** 로 설정합니다.
kind	필수 항목입니다. 정책 유형을 나타내려면 해당 값을 **PolicySet** 으로 설정합니다.
metadata.name	필수 항목입니다. 정책 리소스를 식별하는 이름입니다.
spec	필수 항목입니다. 정책에 대한 설정 세부 정보를 추가합니다.
spec.policies	선택 사항: 정책 세트에서 함께 그룹화할 정책 목록입니다.

2.3.4.3. 정책 세트 샘플

```
apiVersion: policy.open-cluster-management.io/v1beta1
kind: PolicySet
```

```
metadata:
  name: pci
  namespace: default
spec:
  description: Policies for PCI compliance
  policies:
  - policy-pod
  - policy-namespace
status:
  compliant: NonCompliant
  placement:
  - placementBinding: binding1
    placementRule: placement1
    policySet: policyset-ps
```

보안 *정책 관리 주제의 정책 세트 생성* 섹션을 참조하십시오. 또한 배포할 정책 생성기가 필요한 stable PolicySets, PolicySets-- Stable 을 확인합니다. 정책 생성기 설명서를 참조하십시오.

2.3.5. 사용자 정의 정책 컨트롤러 생성(더 이상 사용되지 않음)

사용자 정의 정책 컨트롤러를 작성, 적용, 보기 및 업데이트하는 방법을 학습합니다. 정책 컨트롤러의 YAML 파일을 생성하여 클러스터에 배포할 수 있습니다. 다음 섹션을 보고 정책 컨트롤러를 생성합니다.

2.3.5.1. 정책 컨트롤러 작성

governance-policy-framework 리포지토리에 있는 정책 컨트롤러 프레임워크를 사용합니다. 정책 컨트롤러를 생성하려면 다음 단계를 완료합니다.

1. 다음 명령을 실행하여 governance-policy-framework 리포지토리를 복제합니다.

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

2. 정책 스키마 정의를 업데이트하여 컨트롤러 정책을 사용자 지정합니다. 정책은 다음 내용과 유사합니다.

```
metadata:
  name: samplepolicies.policies.open-cluster-management.io
spec:
  group: policy.open-cluster-management.io
  names:
    kind: SamplePolicy
    listKind: SamplePolicyList
    plural: samplepolicies
    singular: samplepolicy
```

3. 정책 컨트롤러를 업데이트하여 **SamplePolicy** 유형을 확인합니다. 다음 명령을 실행합니다.

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

4. 다음 단계를 완료하여 정책 컨트롤러를 다시 컴파일하고 실행합니다.

   a. 클러스터에 로그인합니다.

   b. 사용자 아이콘을 선택한 다음 **Configure client** 를 클릭합니다.

   c. 구성 정보를 복사하여 명령줄에 붙여넣고 **Enter** 키를 누릅니다.

   d. 다음 명령을 실행하여 정책 **CRD**를 적용하고 컨트롤러를 시작합니다.

```
export GO111MODULE=on

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

export WATCH_NAMESPACE=<cluster_namespace_on_hub>

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

   컨트롤러가 실행됨을 나타내는 다음 출력이 표시될 수 있습니다.

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

   e. 정책을 생성하고 컨트롤러에서 정책을 검색하고 클러스터에 정책을 적용하는지 확인합니다. 다음 명령을 실행합니다.

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

정책이 적용되면 사용자 정의 컨트롤러에서 정책을 모니터링하고 탐지했음을 나타내는 메시지가 표시됩니다. 메시지는 다음 내용과 유사합니다.

```
{"level":"info","ts":1578503685.643426,"logger":"controller_samplepolicy","msg":"Reconciling SamplePolicy","Request.Namespace":"default","Request.Name":"example-samplepolicy"}
{"level":"info","ts":1578503685.855259,"logger":"controller_samplepolicy","msg":"Reconciling SamplePolicy","Request.Namespace":"default","Request.Name":"example-samplepolicy"}
Available policies in namespaces:
namespace = kube-public; policy = example-samplepolicy
namespace = default; policy = example-samplepolicy
namespace = kube-node-lease; policy = example-samplepolicy
```

5. 다음 명령을 실행하여 규정 준수 세부 정보에 대한 상태 필드를 확인합니다.

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

출력은 다음 내용과 유사합니다.

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

6. 정책 컨트롤러에 대한 새 규칙을 도입하도록 정책 규칙 및 정책 논리를 변경합니다. 다음 단계를 완료합니다.

   a. **SamplePolicySpec** 을 업데이트하여 **YAML** 파일에 새 필드를 추가합니다. 사양은 다음 내용과 유사할 수 있습니다.

```
spec:
```

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

b. samplepolicy_controller.go 에서 **SamplePolicySpec** 구조를 새 필드를 사용하여 업데이트합니다.

c. 정책 컨트롤러를 실행하도록 새 논리를 사용하여 **samplepolicy_controller.go** 파일에서 **PeriodicallyExecSamplePolicies** 함수를 업데이트합니다. **Periodically ExecSamplePolicies** 필드 의 예를 보려면 **stolostron/multicloud-operators-policy-controller** 를 참조하십시오.

d. 정책 컨트롤러를 다시 컴파일하고 실행합니다. 정책 컨트롤러참조

정책 컨트롤러가 작동합니다.

2.3.5.2. 클러스터에 컨트롤러 배포

사용자 지정 정책 컨트롤러를 클러스터에 배포하고 정책 컨트롤러를 *관리* 대시보드와 통합합니다. 다음 단계를 완료합니다.

1. 다음 명령을 실행하여 정책 컨트롤러 이미지를 빌드합니다.

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

2. 다음 명령을 실행하여 이미지를 선택한 리포지토리로 내보냅니다. 예를 들어 다음 명령을 실행하여 이미지를 **Docker Hub**로 푸시합니다.

```
docker login

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

3. **Red Hat Advanced Cluster Management for Kubernetes**에서 관리하는 클러스터를 가리키도록 **kubectl** 을 구성합니다.

4. 기본 제공 이미지 이름을 사용하도록 **Operator** 매니페스트를 교체하고 정책을 조사하도록 네임스페이스를 업데이트합니다. 네임스페이스는 클러스터 네임스페이스여야 합니다. 매니페스트는 다음 내용과 유사합니다.

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

5. 다음 명령을 실행하여 **RBAC** 역할을 업데이트합니다.

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

6. 클러스터에 정책 컨트롤러를 배포합니다.

   a. 다음 명령을 실행하여 클러스터의 서비스 계정을 설정합니다.

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

   b. 다음 명령을 실행하여 **Operator**에 대한 **RBAC**를 설정합니다.

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

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

   c. 정책 컨트롤러에 대한 **RBAC**를 설정합니다. 다음 명령을 실행합니다.

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

'

d. 다음 명령을 실행하여 **CRD**(사용자 정의 리소스 정의)를 설정합니다.

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

e. 다음 명령을 실행하여 **multicloud-operator-policy-controller** 를 배포합니다.

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

f. 다음 명령을 실행하여 컨트롤러가 작동하는지 확인합니다.

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

7. 컨트롤러에서 모니터링할 정책을 생성하여 정책 컨트롤러를 통합해야 합니다. 자세한 내용은 콘솔에서 클러스터 보안 정책 생성을 참조하십시오.

2.3.5.2.1. 컨트롤러 배포 스케일링

정책 컨트롤러 배포는 삭제 또는 제거를 지원하지 않습니다. 배포를 스케일링하여 배포가 적용되는 포드를 업데이트할 수 있습니다. 다음 단계를 완료합니다.

1. 관리형 클러스터에 로그인합니다.
2. 사용자 정의 정책 컨트롤러의 배포로 이동합니다.
3. 배포를 확장합니다. 배포를 **0**개의 **Pod**로 확장하면 정책 제어자 배포가 비활성화됩니다.

배포에 대한 자세한 내용은 **OpenShift Container Platform** 배포를 참조하십시오.

정책 컨트롤러가 클러스터에 배포 및 통합되었습니다. 제품 정책 컨트롤러를 확인합니다. 자세한 내용은 정책 컨트롤러를 참조하십시오.

2.4. 타사 정책 컨트롤러 통합

타사 정책을 통합하여 정책 템플릿 내에서 사용자 지정 주석을 생성하여 하나 이상의 규정 준수 표준, 제어 카테고리 및 제어를 지정합니다.

정책 수집/커뮤니티 의 타사 정책을 사용할 수도 있습니다.

다음 타사 정책을 통합하는 방법을 알아보십시오.

- 게이트키퍼 제약 조건 및 제약 조건 템플릿 통합
- 정책 생성기

2.4.1. 게이트키퍼 제약 조건 및 제약 조건 템플릿 통합

Gatekeeper는OPA(Open Policy Agent)를 사용하여 실행되는 CRD(사용자 정의 리소스 정의) 기반 정책을 적용하는 검증 웹 후크입니다. 게이트 키퍼는 게이트키퍼 운영자 정책을 사용하여 클러스터에 설치할 수 있습니다. 게이트키퍼 정책을 사용하여 Kubernetes 리소스 컴플라이언스를 평가할 수 있습니다. OPA를 정책 엔진으로 활용하고 정책 언어로 Rego를 사용할 수 있습니다.

게이트키퍼 정책은 Red Hat Advanced Cluster Management에서 Kubernetes 구성 정책으로 생성됩니다. 게이트키퍼 정책에는 제약 조건 템플릿(ConstraintTemplates) 및 제약 조건, 감사 템플릿, 승인 템플릿이 포함됩니다. 자세한 내용은 Gatekeeper 업스트림 리포지토리를 참조하십시오.

Red Hat Advanced Cluster Management는 Gatekeeper용 버전 3.3.0을 지원하며 Red Hat Advanced Cluster Management gatekeeper 정책에서 다음과 같은 제약 조건 템플릿을 적용합니다.

- ConstraintTemplates 및 제약 조건: policy-gatekeeper-k8srequiredlabels 정책을 사용하여 관리 클러스터에서 게이트키퍼 제약 조건 템플릿을 생성합니다.

```
apiVersion: policy.open-cluster-management.io/v1
kind: ConfigurationPolicy
metadata:
  name: policy-gatekeeper-k8srequiredlabels
spec:
  remediationAction: enforce # will be overridden by remediationAction in parent
policy
  severity: low
  object-templates:
    - complianceType: musthave
      objectDefinition:
```

```
      apiVersion: templates.gatekeeper.sh/v1beta1
      kind: ConstraintTemplate
      metadata:
        name: k8srequiredlabels
      spec:
        crd:
          spec:
            names:
              kind: K8sRequiredLabels
            validation:
              # Schema for the `parameters` field
              openAPIV3Schema:
                properties:
                  labels:
                    type: array
                    items: string
        targets:
          - target: admission.k8s.gatekeeper.sh
            rego: |
              package k8srequiredlabels
              violation[{"msg": msg, "details": {"missing_labels": missing}}] {
                provided := {label | input.review.object.metadata.labels[label]}
                required := {label | label := input.parameters.labels[_]}
                missing := required - provided
                count(missing) > 0
                msg := sprintf("you must provide labels: %v", [missing])
              }
    - complianceType: musthave
      objectDefinition:
        apiVersion: constraints.gatekeeper.sh/v1beta1
        kind: K8sRequiredLabels
        metadata:
          name: ns-must-have-gk
        spec:
          match:
            kinds:
              - apiGroups: [""]
                kinds: ["Namespace"]
            namespaces:
              - e2etestsuccess
              - e2etestfail
          parameters:
            labels: ["gatekeeper"]
```

- 감사 템플릿: **policy-gatekeeper-audit** 를 사용하여 기존의 잘못된 구성을 감지하기 위해 적용되는 게이트 키퍼 정책에 대해 정기적으로 기존 리소스를 확인하고 평가합니다.

```
apiVersion: policy.open-cluster-management.io/v1
kind: ConfigurationPolicy
metadata:
  name: policy-gatekeeper-audit
spec:
  remediationAction: inform # will be overridden by remediationAction in parent policy
  severity: low
```

```
  object-templates:
    - complianceType: musthave
      objectDefinition:
        apiVersion: constraints.gatekeeper.sh/v1beta1
        kind: K8sRequiredLabels
        metadata:
          name: ns-must-have-gk
        status:
          totalViolations: 0
```

- 승인 템플릿: **policy-gatekeeper-admission** 을 사용하여 게이트키퍼 승인 **Webhook**에 의해 생성되는 잘못된 설정을 확인합니다.

```
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
```

자세한 내용은 **policy-gatekeeper-sample.yaml** 을 참조하십시오.

다른 정책 관리에 대한 자세한 내용은 구성 정책 관리를 참조하십시오. 보안 프레임워크에 대한 자세한 내용은 관리에서 참조하십시오.

2.4.2. 정책 생성기

정책 생성기는 **Kustomize**를 사용하여 **Kubernetes** 정책용 **Red Hat Advanced Cluster Management for Kubernetes** 애플리케이션 라이프사이클 서브스크립션 워크플로의 일부로, **Kustomize**를 사용하여 **Kubernetes** 정책용 **Red Hat Advanced Cluster Management**를 생성합니다. 정책 생성기는 **Kubernetes** 매니페스트 **YAML** 파일에서 **Kubernetes** 정책용 **Red Hat Advanced Cluster Management**를 빌드하는 데 사용되는 **PolicyGenerator** 매니페스트 **YAML** 파일을 통해 제공됩

니다. 정책 생성기는 Kustomize 생성기 플러그인으로 구현됩니다. Kustomize에 대한 자세한 내용은 Kustomize 설명서 를 참조하십시오.

이 Red Hat Advanced Cluster Management 버전에 번들된 정책 생성기는 v1.8.0입니다.

2.4.2.1. 정책 생성기 기능

정책 생성기 및 Red Hat Advanced Cluster Management 애플리케이션 라이프사이클 서브스크립션 GitOps와의 통합은 Red Hat Advanced Cluster Management 정책을 통해 Kubernetes 리소스 오브젝트를 관리 OpenShift 클러스터 및 Kubernetes 클러스터에 배포할 수 있도록 단순화합니다. 특히 정책 생성기를 사용하여 다음 작업을 완료합니다.

- 모든 Kubernetes 매니페스트 파일을 Red Hat Advanced Cluster Management 구성 정책으로 변환합니다.
- 생성된 Red Hat Advanced Cluster Management 정책에 삽입되기 전에 입력 Kubernetes 매니페스트를 패치합니다.
- Kubernetes용 Red Hat Advanced Cluster Management를 통해 Gatekeeper 및 Kyverno 정책 위반에 대해 보고할 수 있는 추가 구성 정책을 생성합니다.
- hub 클러스터에서 정책 세트를 생성합니다. 자세한 내용은 정책 세트 컨트롤러를 참조하십시오.

자세한 내용은 에 대해 다음 주제를 확인하십시오.

- 정책 생성기 구성 구조
- Operator를 설치하기 위한 정책 생성
  - OpenShift GitOps를 설치하는 정책
  - Compliance Operator를 설치하는 정책

- OpenShift GitOps(ArgoCD)에 정책 생성기 설치
- 정책 생성기 구성 참조 테이블

2.4.2.2. 정책 생성기 구성 구조

정책 생성기는 **PolicyGenerator** 종류 및 **policy.open-cluster-management.io/v1 API** 버전의 매니페스트로 구성된 **Kustomize** 생성기 플러그인입니다.

플러그인을 사용하려면 **kustomization.yaml** 파일에 **generators** 섹션을 추가하여 시작합니다. 다음 예제를 확인합니다.

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

이전 예제에서 참조하는 **policy-generator-config.yaml** 파일은 생성할 정책의 지침이 포함된 **YAML** 파일입니다. 간단한 정책 생성기 구성 파일은 다음 예와 유사합니다.

```
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 은 정책에 포함할 **Kubernetes** 매니페스트 **YAML** 파일을 나타냅니다. 다음 예제를 확인합니다.

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

생성된 Policy 는 PlacementRule 및 PlacementBinding 과 함께 다음 예와 유사합니다.

```
apiVersion: apps.open-cluster-management.io/v1
kind: PlacementRule
metadata:
  name: placement-config-data
  namespace: policies
spec:
  clusterConditions:
  - status: "True"
    type: ManagedClusterConditionAvailable
  clusterSelector:
    matchExpressions: []
---
apiVersion: policy.open-cluster-management.io/v1
kind: PlacementBinding
metadata:
  name: binding-config-data
  namespace: policies
placementRef:
  apiGroup: apps.open-cluster-management.io
  kind: PlacementRule
  name: placement-config-data
subjects:
- apiGroup: policy.open-cluster-management.io
  kind: Policy
  name: config-data
---
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  annotations:
    policy.open-cluster-management.io/categories: CM Configuration Management
    policy.open-cluster-management.io/controls: CM-2 Baseline Configuration
    policy.open-cluster-management.io/standards: NIST SP 800-53
  name: config-data
  namespace: policies
spec:
  disabled: false
  policy-templates:
  - objectDefinition:
      apiVersion: policy.open-cluster-management.io/v1
      kind: ConfigurationPolicy
      metadata:
        name: config-data
      spec:
        object-templates:
        - complianceType: musthave
          objectDefinition:
            apiVersion: v1
            data:
              key1: value1
              key2: value2
            kind: ConfigMap
```

```
    metadata:
      name: my-config
      namespace: default
  remediationAction: inform
  severity: low
```

자세한 내용은 policy-generator-plugin 리포지터리를 참조하십시오.

2.4.2.3. Operator를 설치하기 위한 정책 생성

Red Hat Advanced Cluster Management 정책을 일반적으로 사용하는 것은 하나 이상의 관리형 OpenShift 클러스터에 Operator를 설치하는 것입니다. 다양한 설치 모드 및 필수 리소스의 다음 예제를 확인합니다.

2.4.2.3.1. OpenShift GitOps를 설치하는 정책

이 예제에서는 정책 생성기를 사용하여 OpenShift GitOps를 설치하는 정책을 생성하는 방법을 보여줍니다. OpenShift GitOps Operator는 *모든 네임스페이스 설치 모드*를 제공합니다. 먼저 다음 예제와 같이 openshift-gitops-subscription.yaml 이라는 서브스크립션 매니페스트 파일을 생성해야 합니다.

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

특정 버전의 Operator에 고정하려면 다음 매개변수 및 값을 추가할 수 있습니다 . spec.startingCSV: openshift-gitops-operator.v<version > . &lt ;version& gt;을 기본 버전으로 바꿉니다.

다음으로 policy-generator -config.yaml이라는 정책 생성기 구성 파일이 필요합니다. 다음 예는 모든 OpenShift 관리 클러스터에 OpenShift GitOps를 설치하는 단일 정책을 보여줍니다.

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

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

필요한 마지막 파일은 **kustomization.yaml** 파일입니다. **kustomization.yaml** 파일에는 다음 설정이 필요합니다.

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

생성된 정책은 다음 파일과 유사합니다.

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

```
    policy.open-cluster-management.io/standards: NIST SP 800-53
  name: install-openshift-gitops
  namespace: policies
spec:
  disabled: false
  policy-templates:
    - objectDefinition:
        apiVersion: policy.open-cluster-management.io/v1
        kind: ConfigurationPolicy
        metadata:
          name: install-openshift-gitops
        spec:
          object-templates:
            - complianceType: musthave
              objectDefinition:
                apiVersion: operators.coreos.com/v1alpha1
                kind: Subscription
                metadata:
                  name: openshift-gitops-operator
                  namespace: openshift-operators
                spec:
                  channel: stable
                  name: openshift-gitops-operator
                  source: redhat-operators
                  sourceNamespace: openshift-marketplace
          remediationAction: enforce
          severity: low
```

입력이 OpenShift Container Platform 설명서에서 제공되는 모든 정책은 정책 생성기에 의해 완전히 지원됩니다. OpenShift Container Platform 설명서에서 지원되는 YAML 입력의 다음 예제를 확인하십시오.

- 설치 후 클러스터 작업
- 감사 로그 정책 구성
- 타사 시스템으로 로그 전달 정보

자세한 내용은 OpenShift GitOps 및 Operator 설명서 이해를 참조하십시오.

2.4.2.3.2. Compliance Operator를 설치하는 정책

네임스페이스 된 설치 모드를 사용하는 Operator의 경우 Compliance Operator와 같이 OperatorGroup 매니페스트도 필요합니다. 이 예제에서는 Compliance Operator를 설치하는 생성된 정

책을 보여줍니다.

먼저 네임스페이스가 있는 YAML 파일, 서브스크립션 및 compliance-operator.yaml 이라는 OperatorGroup 매니페스트를 생성해야 합니다. 다음 예제에서는 이러한 매니페스트를 compliance-operator 네임스페이스에 설치합니다.

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

다음으로 policy-generator -config.yaml이라는 정책 생성기 구성 파일이 필요합니다. 다음 예제는 모든 OpenShift 관리 클러스터에 Compliance Operator를 설치하는 단일 정책을 보여줍니다.

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

필요한 마지막 파일은 **kustomization.yaml** 파일입니다. **kustomization.yaml** 파일에 다음 구성이 필요합니다.

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

결과적으로 생성된 정책은 다음 파일과 유사해야 합니다.

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

```
    kind: ConfigurationPolicy
    metadata:
      name: install-compliance-operator
    spec:
      object-templates:
        - complianceType: musthave
          objectDefinition:
            apiVersion: v1
            kind: Namespace
            metadata:
              name: openshift-compliance
        - complianceType: musthave
          objectDefinition:
            apiVersion: operators.coreos.com/v1alpha1
            kind: Subscription
            metadata:
              name: compliance-operator
              namespace: openshift-compliance
            spec:
              channel: release-0.1
              name: compliance-operator
              source: redhat-operators
              sourceNamespace: openshift-marketplace
        - complianceType: musthave
          objectDefinition:
            apiVersion: operators.coreos.com/v1
            kind: OperatorGroup
            metadata:
              name: compliance-operator
              namespace: openshift-compliance
            spec:
              targetNamespaces:
                - compliance-operator
      remediationAction: enforce
      severity: low
```

자세한 내용은 Compliance Operator 설명서 를 참조하십시오.

2.4.2.4. OpenShift GitOps(ArgoCD)에 정책 생성기 설치

ArgoCD 를 기반으로 하는 OpenShift GitOps를 사용하여 GitOps를 통해 정책 생성기를 생성할 수도 있습니다. 정책 생성기가 OpenShift GitOps 컨테이너 이미지에 사전 설치되지 않으므로 일부 사용자 지정이 필요합니다. 후속 조치를 위해 OpenShift GitOps Operator 가 Red Hat Advanced Cluster Management hub 클러스터에 설치되어 hub 클러스터에 로그인해야 합니다.

OpenShift GitOps가 Kustomize를 실행할 때 정책 생성기에 액세스할 수 있도록 Init Container는 Red Hat Advanced Cluster Management Application Subscription 컨테이너 이미지에서 Kustomize를 실행하는 OpenShift GitOps 컨테이너로 정책 생성기 바이너리를 복사해야 합니다. 자세한 내용은 Pod를 배포하기 전에 Init Container를 사용하여 작업을 수행합니다. 또한 Kustomize를 실행할 때 --

enable-alpha-plugins 플래그를 제공하도록 **OpenShift GitOps**를 구성해야 합니다. 다음 명령을 사용하여 **OpenShift GitOps argocd** 오브젝트 편집을 시작합니다.

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

그런 다음 다음 추가 **YAML** 콘텐츠를 포함하도록 **OpenShift GitOps argocd** 오브젝트를 수정합니다. 새로운 주요 버전의 **Red Hat Advanced Cluster Management**가 릴리스되고 정책 생성기를 최신 버전으로 업데이트하려면 **Init Container**에서 사용하는 **registry.redhat.io/rhacm2/multicluster-operators-subscription-rhel8** 이미지를 최신 태그로 업데이트해야 합니다. 다음 예제를 보고 **< version >**을 **2.5** 또는 원하는 **Red Hat Advanced Cluster Management** 버전으로 교체합니다.

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

OpenShift GitOps에서 정책 생성기를 사용할 수 있으므로 **Red Hat Advanced Cluster Management hub** 클러스터에서 정책을 생성할 수 있는 **OpenShift GitOps**에 액세스 권한이 부여되어야 합니다. 생성, 읽기, 업데이트, 삭제 정책 및 배치에 액세스할 수 있는 **openshift-gitops-policy-admin** 이라는 다음 **ClusterRole** 리소스를 생성합니다. **ClusterRole** 은 다음 예와 유사할 수 있습니다.

```
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
```

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

또한 **ClusterRoleBinding** 오브젝트를 생성하여 **OpenShift GitOps** 서비스 계정 액세스 권한을 **openshift-gitops-policy-admin ClusterRole** 에 부여합니다. **ClusterRoleBinding** 은 다음 리소스와 유사할 수 있습니다.

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

```
    namespace: openshift-gitops
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: openshift-gitops-policy-admin
```

2.4.2.5. 정책 생성기 구성 참조 테이블

네임스페이스 를 제외한 policyDefaults 섹션의 모든 필드는 정책별로 덮어쓸 수 있습니다.

필드	설명
apiVersion	필수 항목입니다. 값을 **policy.open-cluster-management.io/v1** 로 설정합니다.
complianceType	선택 사항: 매니페스트를 클러스터의 오브젝트와 비교할 때 정책 컨트롤러 동작을 결정합니다. 매개변수 값은 **musthave**,**mustonlyhave** 또는 **mustnothave** 입니다. 기본값은 **musthave** 입니다.
kind	필수 항목입니다. 정책 유형을 표시하려면 해당 값을 **PolicyGenerator** 로 설정합니다.
metadata	필수 항목입니다. 구성 파일을 고유하게 식별하는 데 사용됩니다.
metadata.name	필수 항목입니다. 정책 리소스를 식별하는 이름입니다.
placementBindingDefaults	필수 항목입니다. 생성기가 정의된 이름을 사용하여 고유한 **PlacementBinding** 이름을 생성할 수 있도록 **PlacementBinding** 에 여러 정책을 통합하는 데 사용됩니다.
placementBindingDefaults.name	선택 사항: 기본값을 사용하는 대신 사용할 명시적 배치 바인딩 이름을 설정하는 것이 좋습니다.
policyDefaults	필수 항목입니다. 여기에 나열된 기본값은 **네임스페이스** 를 제외한 정책 배열의 항목에 대해 재정의됩니다.
policyDefaults.categories	선택 사항: **policy.open-cluster-management.io/categories** 주석에 사용할 카테고리 배열입니다. 기본값은 **CM 구성 관리** 입니다.
policyDefaults.controls	선택 사항: **policy.open-cluster-management.io/controls** 주석에 사용할 컨트롤 배열입니다. 기본값은 **CM-2 기본 설정입니다**.

필드	설명
policyDefaults.consolidateManifests	선택 사항: 이렇게 하면 정책에서 래핑되는 모든 매니페스트에 대해 단일 구성 정책을 생성해야 하는지 확인합니다. **false** 로 설정하면 매니페스트당 구성 정책이 생성됩니다. 기본값은 **true**입니다.
policyDefaults.informGatekeeperPolicies	선택 사항: 정책이 위반된 게이트키퍼 정책 매니페스트를 참조하는 경우 Red Hat Advanced Cluster Management에서 정책 위반을 받으려면 추가 구성 정책을 생성해야 하는지 결정합니다. 기본값은 **true**입니다.
policyDefaults.informKyvernoPolicies	선택 사항: 정책이 Kyverno 정책 매니페스트를 참조하는 경우 Kyverno 정책을 위반할 때 Red Hat Advanced Cluster Management에서 정책 위반을 수신하기 위해 추가 구성 정책을 생성해야 하는지 결정합니다. 기본값은 **true**입니다.
policyDefaults.namespace	필수 항목입니다. 모든 정책의 네임스페이스입니다.
policyDefaults.placement	선택 사항: 정책에 대한 배치 구성입니다. 기본값은 모든 클러스터와 일치하는 배치 구성입니다.
placement.clusterSelectors	선택 사항: 다음 형식인 **key:value** 형식으로 클러스터 선택기를 정의하여 배치를 지정합니다. 기존 파일을 지정하려면 **placementRulePath** 를 참조하십시오.
placement.name	선택 사항: 동일한 클러스터 선택기를 포함하는 배치 규칙을 통합할 이름을 지정합니다.
placement.placementRulePath	선택 사항: 기존 배치 규칙을 재사용하려면 **kustomization.yaml** 파일을 기준으로 여기에 경로를 지정합니다. 이 배치 규칙은 기본적으로 모든 정책에서 사용됩니다. 새 **배치를** 생성하려면 **clusterSelector** 를 참조하십시오.
policyDefaults.remediationAction	선택 사항: 정책의 수정 메커니즘입니다. 매개변수 값은 **강제 시행** 및 **정보 입니다**. 기본값은 **inform** 입니다.
policyDefaults.severity	선택 사항: 정책 위반의 심각도입니다. 기본값은 **low** 입니다.
policyDefaults.standards	선택 사항: **policy.open-cluster-management.io/standards** 주석에 사용할 표준 배열입니다. 기본값은 **NIST SP 800-53** 입니다.
Policies	필수 항목입니다. 기본값 또는 **policyDefaults에 설정된 값에 대한 재정의와 함께 생성할 정책 목록입니다.**

필드	설명
policies[ ].manifests	필수 항목입니다. 정책에 포함할 Kubernetes 오브젝트 매니페스트 목록입니다.
policies[ ].name	필수 항목입니다. 생성할 정책의 이름입니다.
policies[ ].manifests[ ].complianceType	선택 사항: 매니페스트를 클러스터의 오브젝트와 비교할 때 정책 컨트롤러 동작을 결정합니다. 매개변수 값은 **musthave**,**mustonlyhave** 또는 **mustnothave** 입니다. 기본값은 **musthave** 입니다.
policies[ ].manifests[ ].path	필수 항목입니다. **kustomization.yaml** 파일과 관련된 단일 파일 또는 파일의 플랫 디렉터리 경로입니다.
policies[ ].manifests[ ].patches	선택 사항: 경로에 매니페스트에 적용할 Kustomize 패치입니다. 여러 매니페스트가 있는 경우 패치에는 **apiVersion**,**kind**,**metadata.name**, **metadata.namespace** (해당되는 경우) 필드가 필요하므로 Kustomize에서 패치가 적용되는 매니페스트를 확인할 수 있습니다. 단일 매니페스트가 있는 경우 **metadata.name** 및 **metadata.namespace** 필드가 패치될 수 있습니다.

2.5. 지원되는 정책

Red Hat Advanced Cluster Management for Kubernetes에서 정책을 생성하고 관리할 때 hub 클러스터에서 규칙, 프로세스 및 제어를 정의하는 방법을 알아보려면 지원되는 정책을 확인하십시오.

참고: 의 기존 정책을 복사하여 *Policy YAML* 에 붙여넣을 수 있습니다. 기존 정책을 붙여넣을 때 매개변수 필드의 값이 자동으로 입력됩니다. 검색 기능을 사용하여 정책 YAML 파일에서 콘텐츠를 검색할 수도 있습니다.

2.5.1. 즉시 사용 가능한 정책에 대한 지원 매트릭스

정책	Red Hat OpenShift Container Platform 3.11	Red Hat OpenShift Container Platform 4
메모리 사용량 정책	x	x
네임스페이스 정책	x	x
이미지 취약점 정책	x	x

정책	Red Hat OpenShift Container Platform 3.11	Red Hat OpenShift Container Platform 4
Pod 정책	x	x
Pod 보안 정책(더 이상 사용되지 않음)		
역할 정책	x	x
역할 바인딩 정책	x	x
SCC(보안 컨텍스트 제약 조건 정책)	x	x
ETCD 암호화 정책		x
게이트 키퍼 정책		x
컴플라이언스 Operator 정책		x
E8 검사 정책		x
OpenShift CIS 검사 정책		x
정책 세트		x
Kyverno add network		x
Kyverno add quota		x
Kyverno 동기화 시크릿		x

특정 정책이 적용되는 방법을 보려면 다음 정책 샘플을 확인합니다.

- 이미지 취약점 정책
- 메모리 사용량 정책
- 네임스페이스 정책

- Pod 정책
- Pod 보안 정책
- 역할 정책
- 역할 바인딩 정책
- 보안 컨텍스트 제약 조건 정책
- ETCD 암호화 정책
- 컴플라이언스 Operator 정책
- E8 검사 정책
- OpenShift CIS 검사 정책
- 정책 세트 컨트롤러
- Kyverno 네트워크 정책 추가
- Kyverno에서 할당량 정책 추가
- Kyverno 동기화 보안 정책

자세한 내용은 관리에서 참조하십시오.

2.5.2. 메모리 사용량 정책

Kubernetes 구성 정책 컨트롤러는 메모리 사용량 정책의 상태를 모니터링합니다. 메모리 사용 정책을 사용하여 메모리 및 컴퓨팅 사용량을 제한하거나 제한합니다. 자세한 내용은 Kubernetes 문서 *의 제한 범위*를 참조하십시오.

다음 섹션의 메모리 사용 정책 구조에 대해 자세히 알아보십시오.

- 메모리 사용량 정책 YAML 구조
- 메모리 사용량 정책 테이블
- 메모리 사용량 정책 샘플

2.5.2.1. 메모리 사용량 정책 YAML 구조

메모리 사용량 정책은 다음 YAML 파일과 유사합니다.

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

2.5.2.2. 메모리 사용량 정책 테이블

필드	설명
apiVersion	필수 항목입니다. 값을 **policy.open-cluster-management.io/v1** 로 설정합니다.
kind	필수 항목입니다. 정책 유형을 나타내려면 값을 **Policy** 로 설정합니다.
metadata.name	필수 항목입니다. 정책 리소스를 식별하는 이름입니다.
metadata.namespaces	선택 사항:
spec.namespace	필수 항목입니다. 정책이 적용되는 허브 클러스터 내의 네임스페이스입니다. 정책에 적용할 **네임스페이스** 인 .에 대한 매개변수 값을 입력합니다. **exclude** 매개변수는 정책을 명시적으로 적용하지 않으려는 네임스페이스를 지정합니다. **참고**: 정책 컨트롤러의 오브젝트 템플릿에 지정된 네임스페이스는 해당 상위 정책의 네임스페이스를 덮어씁니다.
remediationAction	선택 사항: 정책 수정을 지정합니다. 매개변수 값은 **강제 시행** 및 **정보 입니다**.
disabled	필수 항목입니다. 값을 **true** 또는 **false** 로 설정합니다. **disabled** 매개변수는 정책을 활성화하고 비활성화하는 기능을 제공합니다.
spec.complianceType	필수 항목입니다. 값을 **"musthave"**로 설정합니다.
spec.object-template	선택 사항: 관리 클러스터에 평가하거나 적용해야 하는 기타 Kubernetes 오브젝트를 나열하는 데 사용합니다.

2.5.2.3. 메모리 사용량 정책 샘플

정책 샘플을 보려면 policy-limitmemory.yaml 을 참조하십시오. 자세한 내용은 보안 정책 관리를 참조하십시오. 컨트롤러에서 모니터링하는 다른 구성 정책을 보려면 Kubernetes 구성 정책 컨트롤러를 참조하십시오.

2.5.3. 네임스페이스 정책

Kubernetes 구성 정책 컨트롤러는 네임스페이스 정책의 상태를 모니터링합니다. 네임스페이스 정책을 적용하여 네임스페이스에 대한 특정 규칙을 정의합니다.

다음 섹션에서 네임스페이스 정책 구조에 대한 자세한 내용을 확인하십시오.

- 네임스페이스 정책 YAML 구조
- 네임스페이스 정책 YAML 테이블
- 네임스페이스 정책 샘플

2.5.3.1. 네임스페이스 정책 YAML 구조

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

2.5.3.2. 네임스페이스 정책 YAML 테이블

필드	설명
apiVersion	필수 항목입니다. 값을 **policy.open-cluster-management.io/v1** 로 설정합니다.
kind	필수 항목입니다. 정책 유형을 나타내려면 값을 **Policy** 로 설정합니다.
metadata.name	필수 항목입니다. 정책 리소스를 식별하는 이름입니다.
metadata.namespaces	선택 사항:

필드	설명
spec.namespace	필수 항목입니다. 정책이 적용되는 허브 클러스터 내의 네임스페이스입니다. 정책에 적용할 **네임스페이스** 인 .에 대한 매개변수 값을 입력합니다. **exclude** 매개변수는 정책을 명시적으로 적용하지 않으려는 네임스페이스를 지정합니다. **참고**: 정책 컨트롤러의 오브젝트 템플릿에 지정된 네임스페이스는 해당 상위 정책의 네임스페이스를 덮어씁니다.
remediationAction	선택 사항: 정책 수정을 지정합니다. 매개변수 값은 **강제 시행** 및 **정보 입니다**.
disabled	필수 항목입니다. 값을 **true** 또는 **false** 로 설정합니다. **disabled** 매개변수는 정책을 활성화하고 비활성화하는 기능을 제공합니다.
spec.complianceType	필수 항목입니다. 값을 **"musthave"**로 설정합니다.
spec.object-template	선택 사항: 관리 클러스터에 평가하거나 적용해야 하는 기타 Kubernetes 오브젝트를 나열하는 데 사용합니다.

2.5.3.3. 네임스페이스 정책 샘플

정책 샘플을 보려면 policy-namespace.yaml 을 참조하십시오.

자세한 내용은 보안 정책 관리를 참조하십시오. 다른 구성 정책에 대한 자세한 내용은 Kubernetes 구성 정책 컨트롤러를 참조하십시오.

2.5.4. 이미지 취약점 정책

컨테이너 보안 Operator를 활용하여 컨테이너 이미지에 취약점이 있는지 감지하기 위해 이미지 취약점 정책을 적용합니다. 이 정책은 설치되지 않은 경우 관리 클러스터에 Container Security Operator를 설치합니다.

이미지 취약점 정책은 Kubernetes 구성 정책 컨트롤러에서 확인합니다. Security Operator에 대한 자세한 내용은 Quay 리포지토리 의 *Container Security Operator* 를 참조하십시오.

참고:

- 이미지 취약점 정책은 연결 해제된 설치 중에 작동하지 않습니다.
- 이미지 취약점 정책은 **IBM Power** 및 **IBM Z** 아키텍처에서 지원되지 않습니다. **Quay Container Security Operator** 를 사용합니다. **container-security-operator** 레지스트리에 **ppc64le** 또는 **s390x** 이미지가 없습니다.

자세한 내용은 다음 섹션을 참조하십시오.

- 이미지 취약점 정책 **YAML** 구조
- 이미지 취약점 정책 **YAML** 테이블
- 이미지 취약점 정책 샘플

2.5.4.1. 이미지 취약점 정책 YAML 구조

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

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

2.5.4.2. 이미지 취약점 정책 YAML 테이블

필드	설명
apiVersion	필수 항목입니다. 값을 **policy.open-cluster-management.io/v1** 로 설정합니다.
kind	필수 항목입니다. 정책 유형을 나타내려면 값을 **Policy** 로 설정합니다.
metadata.name	필수 항목입니다. 정책 리소스를 식별하는 이름입니다.
metadata.namespaces	선택 사항:
spec.namespace	필수 항목입니다. 정책이 적용되는 허브 클러스터 내의 네임스페이스입니다. 정책에 적용할 **네임스페이스** 인 .에 대한 매개변수 값을 입력합니다. **exclude** 매개변수는 정책을 명시적으로 적용하지 않으려는 네임스페이스를 지정합니다. **참고**: 정책 컨트롤러의 오브젝트 템플릿에 지정된 네임스페이스는 해당 상위 정책의 네임스페이스를 덮어씁니다.
remediationAction	선택 사항: 정책 수정을 지정합니다. 매개변수 값은 **강제 시행** 및 **정보 입니다**.
disabled	필수 항목입니다. 값을 **true** 또는 **false** 로 설정합니다. **disabled** 매개변수는 정책을 활성화하고 비활성화하는 기능을 제공합니다.
spec.complianceType	필수 항목입니다. 값을 **"musthave"**로 설정합니다.
spec.object-template	선택 사항: 관리 클러스터에 평가하거나 적용해야 하는 기타 Kubernetes 오브젝트를 나열하는 데 사용합니다.

2.5.4.3. 이미지 취약점 정책 샘플

policy-imagemanifestvuln.yaml 을 참조하십시오. 자세한 내용은 보안 정책 관리를 참조하십시오.

구성 컨트롤러에서 모니터링하는 다른 구성 정책을 보려면 Kubernetes 구성 정책 컨트롤러를 참조하십시오.

2.5.5. Pod 정책

Kubernetes 구성 정책 컨트롤러는 Pod 정책의 상태를 모니터링합니다. Pod 정책을 적용하여 Pod의 컨테이너 규칙을 정의합니다. 이 정보를 사용하려면 클러스터에 Pod가 있어야 합니다.

다음 섹션의 **Pod** 정책 구조에 대한 자세한 내용을 확인하십시오.

- Pod 정책 YAML 구조
- Pod 정책 테이블
- Pod 정책 샘플

2.5.5.1. Pod 정책 YAML 구조

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

2.5.5.2. Pod 정책 테이블

필드	설명
apiVersion	필수 항목입니다. 값을 **policy.open-cluster-management.io/v1** 로 설정합니다.
kind	필수 항목입니다. 정책 유형을 나타내려면 값을 **Policy** 로 설정합니다.

필드	설명
metadata.name	필수 항목입니다. 정책 리소스를 식별하는 이름입니다.
metadata.namespaces	선택 사항:
spec.namespace	필수 항목입니다. 정책이 적용되는 허브 클러스터 내의 네임스페이스입니다. 정책에 적용할 **네임스페이스** 인 .에 대한 매개변수 값을 입력합니다. **exclude** 매개변수는 정책을 명시적으로 적용하지 않으려는 네임스페이스를 지정합니다. **참고**: 정책 컨트롤러의 오브젝트 템플릿에 지정된 네임스페이스는 해당 상위 정책의 네임스페이스를 덮어씁니다.
remediationAction	선택 사항: 정책 수정을 지정합니다. 매개변수 값은 **강제 시행** 및 **정보 입니다**.
disabled	필수 항목입니다. 값을 **true** 또는 **false** 로 설정합니다. **disabled** 매개변수는 정책을 활성화하고 비활성화하는 기능을 제공합니다.
spec.complianceType	필수 항목입니다. 값을 **"musthave"**로 설정합니다.
spec.object-template	선택 사항: 관리 클러스터에 평가하거나 적용해야 하는 기타 Kubernetes 오브젝트를 나열하는 데 사용합니다.

2.5.5.3. Pod 정책 샘플

정책 샘플을 보려면 policy-pod.yaml 을 참조하십시오.

구성 컨트롤러에서 모니터링하는 다른 구성 정책을 보려면 Kubernetes 구성 정책 컨트롤러를 참조하십시오. 다른 정책을 관리하려면 구성 정책 관리를 참조하십시오.

2.5.6. Pod 보안 정책

Kubernetes 구성 정책 컨트롤러는 Pod 보안 정책의 상태를 모니터링합니다. Pod 보안 정책을 적용하여 Pod 및 컨테이너를 보호합니다. 자세한 내용은 Kubernetes 문서의 *Pod 보안 정책*을 참조하십시오.

다음 섹션의 Pod 보안 정책 구조에 대해 자세히 알아보십시오.

- Pod 보안 정책 YAML 구조

- Pod 보안 정책 테이블
- Pod 보안 정책 샘플

2.5.6.1. Pod 보안 정책 YAML 구조

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

2.5.6.2. Pod 보안 정책 테이블

필드	설명

필드	설명
apiVersion	필수 항목입니다. 값을 **policy.open-cluster-management.io/v1** 로 설정합니다.
kind	필수 항목입니다. 정책 유형을 나타내려면 값을 **Policy** 로 설정합니다.
metadata.name	필수 항목입니다. 정책 리소스를 식별하는 이름입니다.
metadata.namespaces	선택 사항:
spec.namespace	필수 항목입니다. 정책이 적용되는 허브 클러스터 내의 네임스페이스입니다. 정책에 적용할 **네임스페이스** 인 .에 대한 매개변수 값을 입력합니다. **exclude** 매개변수는 정책을 명시적으로 적용하지 않으려는 네임스페이스를 지정합니다. **참고**: 정책 컨트롤러의 오브젝트 템플릿에 지정된 네임스페이스는 해당 상위 정책의 네임스페이스를 덮어씁니다.
remediationAction	선택 사항: 정책 수정을 지정합니다. 매개변수 값은 **강제 시행** 및 **정보 입니다**.
disabled	필수 항목입니다. 값을 **true** 또는 **false** 로 설정합니다. **disabled** 매개변수는 정책을 활성화하고 비활성화하는 기능을 제공합니다.
spec.complianceType	필수 항목입니다. 값을 **"musthave"**로 설정합니다.
spec.object-template	선택 사항: 관리 클러스터에 평가하거나 적용해야 하는 기타 Kubernetes 오브젝트를 나열하는 데 사용합니다.

2.5.6.3. Pod 보안 정책 샘플

샘플 정책을 보려면 **policy-psp.yaml** 을 참조하십시오. 자세한 내용은 구성 정책 관리를 참조하십시오.

컨트롤러에서 모니터링하는 다른 구성 정책을 보려면 **Kubernetes** 구성 정책 컨트롤러를 참조하십시오.

2.5.7. 역할 정책

Kubernetes 구성 정책 컨트롤러는 역할 정책의 상태를 모니터링합니다. **object-template** 에서 역할을 정의하여 클러스터의 특정 역할에 대한 규칙과 권한을 설정합니다.

다음 섹션의 역할 정책 구조에 대해 자세히 알아보십시오.

- 역할 정책 YAML 구조
- 역할 정책 테이블
- 역할 정책 샘플

2.5.7.1. 역할 정책 YAML 구조

```
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  name: policy-role
  namespace:
  annotations:
    policy.open-cluster-management.io/standards: NIST-CSF
    policy.open-cluster-management.io/categories: PR.AC Identity Management Authentication
and Access Control
    policy.open-cluster-management.io/controls: PR.AC-4 Access Control
spec:
  remediationAction: inform
  disabled: false
  policy-templates:
    - objectDefinition:
        apiVersion: policy.open-cluster-management.io/v1
        kind: ConfigurationPolicy
        metadata:
          name: policy-role-example
        spec:
          remediationAction: inform # will be overridden by remediationAction in parent policy
          severity: high
          namespaceSelector:
            include: ["default"]
          object-templates:
            - complianceType: mustonlyhave # role definition should exact match
              objectDefinition:
                apiVersion: rbac.authorization.k8s.io/v1
                kind: Role
                metadata:
                  name: sample-role
                rules:
                  - apiGroups: ["extensions", "apps"]
                    resources: ["deployments"]
                    verbs: ["get", "list", "watch", "delete","patch"]
---
apiVersion: policy.open-cluster-management.io/v1
kind: PlacementBinding
```

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

        ...
```

2.5.7.2. 역할 정책 테이블

필드	설명
apiVersion	필수 항목입니다. 값을 **policy.open-cluster-management.io/v1** 로 설정합니다.
kind	필수 항목입니다. 정책 유형을 나타내려면 값을 **Policy** 로 설정합니다.
metadata.name	필수 항목입니다. 정책 리소스를 식별하는 이름입니다.
metadata.namespaces	선택 사항:
spec.namespace	필수 항목입니다. 정책이 적용되는 허브 클러스터 내의 네임스페이스입니다. 정책에 적용할 **네임스페이스** 인 .에 대한 매개변수 값을 입력합니다. **exclude** 매개변수는 정책을 명시적으로 적용하지 않으려는 네임스페이스를 지정합니다. **참고**: 정책 컨트롤러의 오브젝트 템플릿에 지정된 네임스페이스는 해당 상위 정책의 네임스페이스를 덮어씁니다.
remediationAction	선택 사항: 정책 수정을 지정합니다. 매개변수 값은 **강제 시행** 및 **정보 입니다**.

필드	설명
disabled	필수 항목입니다. 값을 **true** 또는 **false** 로 설정합니다. **disabled** 매개변수는 정책을 활성화하고 비활성화하는 기능을 제공합니다.
spec.complianceType	필수 항목입니다. 값을 **"musthave"**로 설정합니다.
spec.object-template	선택 사항: 관리 클러스터에 평가하거나 적용해야 하는 기타 Kubernetes 오브젝트를 나열하는 데 사용합니다.

2.5.7.3. 역할 정책 샘플

클러스터의 특정 역할에 대한 규칙과 권한을 설정하기 위해 역할 정책을 적용합니다. 역할에 대한 자세한 내용은 역할 기반 액세스 제어를 참조하십시오. 역할 정책의 샘플을 확인합니다. policy-role.yaml 을 참조하십시오.

역할 정책을 관리하는 방법에 대한 자세한 내용은 구성 정책 관리를 참조하십시오. 컨트롤러를 모니터링하는 다른 구성 정책을 보려면 Kubernetes 구성 정책 컨트롤러를 참조하십시오.

2.5.8. 역할 바인딩 정책

Kubernetes 구성 정책 컨트롤러는 역할 바인딩 정책의 상태를 모니터링합니다. 관리 클러스터의 네임스페이스에 정책을 바인딩하려면 역할 바인딩 정책을 적용합니다.

다음 섹션에서 네임스페이스 정책 구조에 대한 자세한 내용을 확인하십시오.

- 역할 바인딩 정책 YAML 구조
- 역할 바인딩 정책 테이블
- 역할 바인딩 정책 샘플

2.5.8.1. 역할 바인딩 정책 YAML 구조

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

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

2.5.8.2. 역할 바인딩 정책 테이블

필드	설명
apiVersion	필수 항목입니다. 값을 **policy.open-cluster-management.io/v1** 로 설정합니다.
kind	필수 항목입니다. 정책 유형을 나타내려면 값을 **Policy** 로 설정합니다.
metadata.name	필수 항목입니다. 정책 리소스를 식별하는 이름입니다.
metadata.namespaces	필수 항목입니다. 정책이 생성되는 관리형 클러스터 내의 네임스페이스입니다.
spec	필수 항목입니다. 규정 준수 위반을 식별하고 수정하는 방법에 대한 사양입니다.
metadata.name	필수 항목입니다. 정책 리소스를 식별하는 이름입니다.
metadata.namespaces	선택 사항:
spec.complianceType	필수 항목입니다. 값을 **"musthave"**로 설정합니다.

필드	설명
spec.namespace	필수 항목입니다. 정책을 적용하려는 관리형 클러스터 네임스페이스입니다. 정책에 적용할 **네임스페이스** 인 .에 대한 매개변수 값을 입력합니다. **exclude** 매개변수는 정책을 명시적으로 적용하지 않으려는 네임스페이스를 지정합니다. **참고**: 정책 컨트롤러의 오브젝트 템플릿에 지정된 네임스페이스는 해당 상위 정책의 네임스페이스를 덮어씁니다.
spec.remediationAction	필수 항목입니다. 정책 수정을 지정합니다. 매개변수 값은 **강제 시행** 및 **정보 입니다**.
spec.object-template	필수 항목입니다. 관리 클러스터에 평가하거나 적용해야 하는 기타 Kubernetes 오브젝트를 나열하는 데 사용합니다.

2.5.8.3. 역할 바인딩 정책 샘플

정책 샘플을 보려면 **policy-rolebinding.yaml** 을 참조하십시오. 다른 정책 관리에 대한 자세한 내용은 구성 정책 관리를 참조하십시오.

다른 구성 정책에 대한 자세한 내용은 **Kubernetes** 구성 정책 컨트롤러를 참조하십시오.

2.5.9. 보안 컨텍스트 제약 조건 정책

Kubernetes 구성 정책 컨트롤러는 **SCC**(보안 컨텍스트 제약 조건) 정책의 상태를 모니터링합니다. 정책에서 조건을 정의하여 **Pod**에 대한 권한을 제어하는 **SCC**(보안 컨텍스트 제약 조건) 정책을 적용합니다.

다음 섹션에서 **SCC** 정책에 대해 자세히 알아보십시오.

- **SCC** 정책 **YAML** 구조
- **SCC** 정책 테이블
- **SCC** 정책 샘플

2.5.9.1. SCC 정책 YAML 구조

```
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  name: policy-scc
  namespace: open-cluster-management-policies
spec:
  complianceType:
  remediationAction:
  namespaces:
    exclude:
    include:
  object-templates:
    - complianceType:
      objectDefinition:
        apiVersion:
        kind: SecurityContextConstraints # restricted scc
        metadata:
          annotations:
            kubernetes.io/description:
          name: sample-restricted-scc
        allowHostDirVolumePlugin:
        allowHostIPC:
        allowHostNetwork:
        allowHostPID:
        allowHostPorts:
        allowPrivilegeEscalation:
        allowPrivilegedContainer:
        allowedCapabilities:
        defaultAddCapabilities:
        fsGroup:
         type:
        groups:
        - system:
        priority:
        readOnlyRootFilesystem:
        requiredDropCapabilities:
        runAsUser:
          type:
        seLinuxContext:
          type:
        supplementalGroups:
          type:
        users:
        volumes:
```

2.5.9.2. SCC 정책 테이블

필드	설명
apiVersion	필수 항목입니다. 값을 **policy.open-cluster-management.io/v1** 로 설정합니다.

필드	설명
kind	필수 항목입니다. 정책 유형을 나타내려면 값을 **Policy** 로 설정합니다.
metadata.name	필수 항목입니다. 정책 리소스를 식별하는 이름입니다.
metadata.namespace	필수 항목입니다. 정책이 생성되는 관리형 클러스터 내의 네임스페이스입니다.
spec.complianceType	필수 항목입니다. 값을 **"musthave"**로 설정합니다.
spec.remediationAction	필수 항목입니다. 정책 수정을 지정합니다. 매개변수 값은 **강제 시행** 및 **정보 입니다**. **중요**: 일부 정책은 적용 기능을 지원하지 않을 수 있습니다.
spec.namespace	필수 항목입니다. 정책을 적용하려는 관리형 클러스터 네임스페이스입니다. 정책에 적용할 **네임스페이스** 인 .에 대한 매개변수 값을 입력합니다. **exclude** 매개변수는 정책을 명시적으로 적용하지 않으려는 네임스페이스를 지정합니다. **참고**: 정책 컨트롤러의 오브젝트 템플릿에 지정된 네임스페이스는 해당 상위 정책의 네임스페이스를 덮어씁니다.
spec.object-template	필수 항목입니다. 관리 클러스터에 평가하거나 적용해야 하는 기타 Kubernetes 오브젝트를 나열하는 데 사용합니다.

SCC 정책의 콘텐츠에 대한 자세한 내용은 OpenShift Container Platform 설명서에서 보안 컨텍스트 제약 조건 관리를 참조하십시오.

2.5.9.3. SCC 정책 샘플

정책에서 조건을 정의하여 Pod에 대한 권한을 제어하는 SCC(보안 컨텍스트 제약 조건) 정책을 적용합니다. 자세한 내용은 Managing Security Context Constraints (SCC) 에서 참조하십시오.

정책 샘플을 보려면 policy-scc.yaml 을 참조하십시오. 다른 정책 관리에 대한 자세한 내용은 구성 정책 관리를 참조하십시오.

다른 구성 정책에 대한 자세한 내용은 Kubernetes 구성 정책 컨트롤러를 참조하십시오.

2.5.10. ETCD 암호화 정책

etcd 암호화 정책을 적용하여 ETCD 데이터 저장소에서 중요한 데이터의 암호화를 감지하거나 활성화합니다. Kubernetes 구성 정책 컨트롤러는 etcd-encryption 정책의 상태를 모니터링합니다. 자세한 내용은 OpenShift Container Platform 설명서의 etcd 데이터 암호화 를 참조하십시오. 참고: ETCD 암호화 정책은 Red Hat OpenShift Container Platform 4 이상만 지원합니다.

다음 섹션에서 etcd-encryption 정책 구조에 대한 자세한 내용을 확인하십시오.

- ETCD 암호화 정책 YAML 구조
- ETCD 암호화 정책 테이블
- etcd 암호화 정책 샘플

2.5.10.1. ETCD 암호화 정책 YAML 구조

etcd-encryption 정책은 다음 YAML 파일과 유사합니다.

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

2.5.10.2. ETCD 암호화 정책 테이블

표 2.5. 매개변수 테이블

필드	설명
apiVersion	필수 항목입니다. 값을 **policy.open-cluster-management.io/v1** 로 설정합니다.
kind	필수 항목입니다. 정책 유형을 나타내는 값을 **Policy** 로 설정합니다(예: **ConfigurationPolicy**).
metadata.name	필수 항목입니다. 정책 리소스를 식별하는 이름입니다.
metadata.namespaces	선택 사항:
spec.namespace	필수 항목입니다. 정책이 적용되는 허브 클러스터 내의 네임스페이스입니다. 정책에 적용할 **네임스페이스** 인 .에 대한 매개변수 값을 입력합니다. **exclude** 매개변수는 정책을 명시적으로 적용하지 않으려는 네임스페이스를 지정합니다. **참고**: 정책 컨트롤러의 오브젝트 템플릿에 지정된 네임스페이스는 해당 상위 정책의 네임스페이스를 덮어씁니다.
remediationAction	선택 사항: 정책 수정을 지정합니다. 매개변수 값은 **강제 시행** 및 **정보 입니다**. **중요**: 일부 정책은 적용 기능을 지원하지 않을 수 있습니다.
disabled	필수 항목입니다. 값을 **true** 또는 **false** 로 설정합니다. **disabled** 매개변수는 정책을 활성화하고 비활성화하는 기능을 제공합니다.
spec.complianceType	필수 항목입니다. 값을 **"musthave"**로 설정합니다.
spec.object-template	선택 사항: 관리 클러스터에 평가하거나 적용해야 하는 기타 Kubernetes 오브젝트를 나열하는 데 사용합니다. OpenShift Container Platform 설명서의 etcd 데이터 암호화 를 참조하십시오.

2.5.10.3. etcd 암호화 정책 샘플

정책 샘플은 policy-etcdencryption.yaml 을 참조하십시오. 자세한 내용은 보안 정책 관리를 참조하십시오.

컨트롤러에서 모니터링하는 다른 구성 정책을 보려면 Kubernetes 구성 정책 컨트롤러를 참조하십시오.

2.5.11. 컴플라이언스 Operator 정책

Compliance Operator는 OpenSCAP을 실행하는 Operator이며 필요한 보안 벤치마크와 함께 Red

Hat OpenShift Container Platform 클러스터를 계속 준수할 수 있습니다. Compliance Operator 정책을 사용하여 관리 클러스터에 규정 준수 Operator를 설치할 수 있습니다.

규정 준수 Operator 정책은 Red Hat Advanced Cluster Management에서 Kubernetes 구성 정책으로 생성됩니다. OpenShift Container Platform 4.7 및 4.6은 규정 준수 Operator 정책을 지원합니다. 자세한 내용은 OpenShift Container Platform 설명서 의 Compliance Operator 이해를 참조하십시오.

참고: Compliance Operator 정책은 IBM Power 또는 IBM Z 아키텍처에서 지원되지 않는 OpenShift Container Platform Compliance Operator를 사용합니다. Compliance Operator에 대한 자세한 내용은 OpenShift Container Platform 설명서의 Compliance Operator 이해를 참조하십시오.

2.5.11.1. Compliance Operator 리소스

규정 준수 Operator 정책을 생성하면 다음 리소스가 생성됩니다.

- Operator 설치를 위한 컴플라이언스 Operator 네임스페이스(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
```

- 대상 네임스페이스를 지정하는 Operator 그룹(compliance-operator)입니다.

```
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
```

- 이름과 채널을 참조하는 서브스크립션(comp-operator-subscription)입니다. 서브스크립션은 다음을 지원하는 컨테이너로 프로필을 가져옵니다.

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

컴플라이언스 운영자 정책을 설치하면 compliance-operator,ocp4 및 rhcos4 의 포드가 생성됩니다. policy-compliance-operator-install.yaml 샘플을 참조하십시오.

규정 준수 Operator를 설치한 후 E8 검사 정책 및 OpenShift CIS 검사 정책을 생성하고 적용할 수도 있습니다. 자세한 내용은 E8 검사 정책 및 OpenShift CIS 스캔 정책을 참조하십시오.

규정 준수 운영자 정책 관리에 대한 자세한 내용은 보안 정책 관리를 참조하십시오. 구성 정책에 대한 자세한 내용은 Kubernetes 구성 정책 컨트롤러를 참조하십시오.

2.5.12. E8 검사 정책

Essential 8 (E8) 검사 정책은 마스터 및 작업자 노드에서 E8 보안 프로필을 준수하는지 확인하는 검사

를 배포합니다. E8 검사 정책을 적용하려면 규정 준수 Operator를 설치해야 합니다.

E8 검사 정책은 Red Hat Advanced Cluster Management에서 Kubernetes 구성 정책으로 생성됩니다. OpenShift Container Platform 4.7 및 4.6에서는 E8 검사 정책을 지원합니다. 자세한 내용은 OpenShift Container Platform 설명서 *의 Compliance Operator 이해*를 참조하십시오.

2.5.12.1. E8 검사 정책 리소스

E8 검사 정책을 생성하면 다음 리소스가 생성됩니다.

- 검사할 프로필을 식별하는 ScanSettingBinding 리소스(e8)입니다.

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

- status 필드를 확인하여 검사가 완료되었는지 확인하는 ComplianceSuite 리소스(compliance-suite-e8)입니다.

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

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

- ComplianceCheckResult CR(사용자 정의 리소스)을 확인하여 검사 모음의 결과를 보고하는 ComplianceCheckResult 리소스(compliance-suite-e8-results)입니다.

```
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
```

참고: 자동 수정이 지원됩니다. ScanSettingBinding 리소스를 생성하기 위해 적용 되도록 수정 작업을 설정합니다.

policy-compliance-operator-e8-scan.yaml 샘플을 참조하십시오. 자세한 내용은 보안 정책 관리를 참조하십시오. 참고: E8 정책이 삭제되면 대상 클러스터 또는 클러스터에서 제거됩니다.

2.5.13. OpenShift CIS 검사 정책

OpenShift CIS 검사 정책은 마스터 및 작업자 노드를 확인하여 OpenShift CIS 보안 벤치마크를 준수하는 검사를 배포합니다. OpenShift CIS 정책을 적용하려면 규정 준수 Operator를 설치해야 합니다.

OpenShift CIS 검사 정책은 Red Hat Advanced Cluster Management에서 Kubernetes 구성 정책으로 생성됩니다. OpenShift Container Platform 4.9, 4.7 및 4.6은 OpenShift CIS 검사 정책을 지원합니다. 자세한 내용은 OpenShift Container Platform 설명서 의 Compliance Operator 이해를 참조하십시오.

2.5.13.1. OpenShift CIS 리소스

OpenShift CIS 검사 정책을 생성하면 다음 리소스가 생성됩니다.

- 검사할 프로필을 식별하는 ScanSettingBinding 리소스(cis)입니다.

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

- status 필드를 확인하여 검사가 완료되었는지 확인하는 ComplianceSuite 리소스(compliance-suite-cis)

```
apiVersion: policy.open-cluster-management.io/v1
kind: ConfigurationPolicy
metadata:
  name: compliance-suite-cis
```

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

- ComplianceCheckResult CR(사용자 정의 리소스)을 확인하여 검사 모음의 결과를 보고하는 ComplianceCheckResult 리소스(compliance-suite-cis-results)입니다.

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

policy-compliance-operator-cis-scan.yaml 파일의 샘플을 참조하십시오. 정책 생성에 대한 자세한 내용은 보안 정책 관리를 참조하십시오.

2.5.14. Kyverno 네트워크 정책 추가

Kyverno add network 정책은 새 네임스페이스를 생성할 때 모든 트래픽을 거부하는 default-deny 라는 새 NetworkPolicy 리소스를 구성합니다. Kyverno 정책을 사용하려면 Kyverno 컨트롤러를 설치해야 합니다. 설치 정책은 policy-install-kyverno.yaml 을 참조하십시오.

다음 섹션에서 policy-kyverno-add-network-policy 정책 구조에 대한 자세한 내용을 확인하십시오.

- Kyverno는 네트워크 정책 YAML 구조 추가
- Kyverno 네트워크 정책 샘플 추가

2.5.14.1. Kyverno는 네트워크 정책 YAML 구조 추가

policy-kyverno-add-network-policy 정책은 다음 YAML 파일과 유사합니다.

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

2.5.14.2. Kyverno 네트워크 정책 샘플 추가

정책 샘플은 policy-kyverno-add-network-policy.yaml 을 참조하십시오. 정책 및 구성 정책 필드에 대한 자세한 내용은 정책 개요 문서 및 Kubernetes 구성 정책 컨트롤러를 참조하십시오.

2.5.15. Kyverno에서 할당량 정책 추가

Kyverno 추가 할당량 정책은 새 네임스페이스를 생성할 때 새 ResourceQuota 및 LimitRange 리소스를 구성합니다. Kyverno 정책을 사용하려면 Kyverno 컨트롤러를 설치해야 합니다. 설치 정책은 policy-install-kyverno.yaml 을 참조하십시오.

다음 섹션에서 policy-kyverno-add-quota 정책 구조에 대한 자세한 내용을 확인하십시오.

- Kyverno는 할당량 정책 YAML 구조 추가
- Kyverno에서 할당량 정책 샘플 추가

2.5.15.1. Kyverno는 할당량 정책 YAML 구조 추가

policy-kyverno-add-quota 정책은 다음 YAML 파일과 유사합니다.

```
apiVersion: kyverno.io/v1
kind: ClusterPolicy
metadata:
  name: add-ns-quota
  annotations:
    policies.kyverno.io/title: Add Quota
    policies.kyverno.io/category: Multi-Tenancy
    policies.kyverno.io/subject: ResourceQuota, LimitRange
    policies.kyverno.io/description: >-
      To better control the number of resources that can be created in a given
      Namespace and provide default resource consumption limits for Pods,
      ResourceQuota and LimitRange resources are recommended.
      This policy will generate ResourceQuota and LimitRange resources when
      a new Namespace is created.
spec:
  rules:
  - name: generate-resourcequota
    match:
      resources:
        kinds:
        - Namespace
    generate:
      apiVersion: v1
      kind: ResourceQuota
```

```
    name: default-resourcequota
    synchronize: true
    namespace: "{{request.object.metadata.name}}"
    data:
      spec:
        hard:
          requests.cpu: '4'
          requests.memory: '16Gi'
          limits.cpu: '4'
          limits.memory: '16Gi'
- name: generate-limitrange
  match:
    resources:
      kinds:
      - Namespace
  generate:
    apiVersion: v1
    kind: LimitRange
    name: default-limitrange
    synchronize: true
    namespace: "{{request.object.metadata.name}}"
    data:
      spec:
        limits:
        - default:
            cpu: 500m
            memory: 1Gi
          defaultRequest:
            cpu: 200m
            memory: 256Mi
          type: Container
```

2.5.15.2. Kyverno에서 할당량 정책 샘플 추가

정책 샘플은 **policy-kyverno-add-quota.yaml** 을 참조하십시오. 정책 및 구성 정책 필드에 대한 자세한 내용은 정책 개요 문서 및 **Kubernetes** 구성 정책 컨트롤러를 참조하십시오.

2.5.16. Kyverno 동기화 보안 정책

Kyverno 동기화 보안 정책은 기본 네임스페이스에 존재하는 **regcred** 라는 보안을 새 네임스페이스에 복사하고 변경 사항이 탐지될 때 시크릿을 업데이트합니다. **Kyverno** 정책을 사용하려면 **Kyverno** 컨트롤러를 설치해야 합니다. 설치 정책은 **policy-install-kyverno.yaml** 을 참조하십시오.

다음 섹션에서 **policy-kyverno-sync-secrets** 정책 구조에 대해 자세히 알아보십시오.

- **Kyverno** 동기화 보안 정책 **YAML** 구조

- Kyverno 동기화 보안 정책 샘플

2.5.16.1. Kyverno 동기화 보안 정책 YAML 구조

policy-kyverno-sync-secrets 정책은 다음 YAML 파일과 유사합니다.

```
apiVersion: kyverno.io/v1
kind: ClusterPolicy
metadata:
  name: sync-secrets
  annotations:
    policies.kyverno.io/title: Sync Secrets
    policies.kyverno.io/category: Sample
    policies.kyverno.io/subject: Secret
    policies.kyverno.io/description: >-
      Secrets like registry credentials often need to exist in multiple
      Namespaces so Pods there have access. Manually duplicating those Secrets
      is time consuming and error prone. This policy will copy a
      Secret called `regcred` which exists in the `default` Namespace to
      new Namespaces when they are created. It will also push updates to
      the copied Secrets should the source Secret be changed.
spec:
  rules:
  - name: sync-image-pull-secret
    match:
      resources:
        kinds:
        - Namespace
    generate:
      apiVersion: v1
      kind: Secret
      name: regcred
      namespace: "{{request.object.metadata.name}}"
      synchronize: true
      clone:
        namespace: default
        name: regcred
```

2.5.16.2. Kyverno 동기화 보안 정책 샘플

정책 샘플은 policy-kyverno-sync-secrets.yaml 을 참조하십시오. 정책 및 구성 정책 필드에 대한 자세한 내용은 정책 개요 문서 및 Kubernetes 구성 정책 컨트롤러를 참조하십시오.

2.6. 보안 정책 관리

거버넌스 대시보드를 사용하여 보안 정책 및 정책 위반을 생성, 보기 및 관리할 수 있습니다. CLI 및 콘솔에서 정책에 대한 YAML 파일을 생성할 수 있습니다.

2.6.1. 거버넌스 페이지

관리 페이지에 다음 탭이 표시됩니다.

- 개요

  개요 탭에서 다음 요약 카드를 확인하십시오. *정책 위반,정책 위반 ,클러스터,카테고리,제어* 및 *표준*.

- 정책 세트

  허브 클러스터 정책 세트를 만들고 관리합니다.

- **Policies**

  보안 정책을 생성하고 관리합니다. 정책 표에는 *이름,네임스페이스,상태* 관리,*정책 세트,클러스터 위반,소스,자동화* 및 *생성* 이라는 세부 정보가 나열됩니다.

  ***Actions** 아이콘을 선택하여 정책을 알리거나 적용하도록 수정을 편집, 활성화 또는 비활성화, 설정 또는 설정할 수 있습니다. 드롭다운 화살표를 선택하여 행을 확장하여 특정 정책의 카테고리 및 표준을 볼 수 있습니다.*

  *여러 정책을 선택하고 **Actions** 버튼을 클릭하여 대규모 작업을 완료합니다. 필터 버튼을 클릭하여 정책 테이블을 사용자 지정할 수도 있습니다.*

특정 정책에 대한 자동화가 구성된 경우 자동화를 선택하여 자세한 정보를 볼 수 있습니다. 자동화를 위한 일정 빈도 옵션에 대한 다음 설명을 확인합니다.

- *수동 실행 모드: 수동으로 이 자동화를 한 번 실행하도록 설정합니다. 자동화가 실행된 후 비활성화 됨으로 설정됩니다. 자동화의 **Manual run** 확인란을 선택합니다.*

- ***once mode:** 정책을 위반하면 자동화가 한 번 실행됩니다. 자동화가 실행된 후 비활성화 됨으로 설정됩니다. 자동화가 **disabled** 로 설정된 후 자동화를 수동으로 계속 실행해야 합니다. 한 번 모드를 실행하면 정책을 위반하는 클러스터 목록과 **target_cluster** 의 추가 변수가 자동으로*

제공됩니다. **Ansible Tower** 작업 템플릿에는 **EXTRA VARIABLES** (추가 변수) 섹션에 대해 **PROMPT ONECDHE**(시작 시 **PROMPT**)가 활성화되어 있어야 합니다.

- **Disabled:** 예약된 자동화가 비활성화 됨으로 설정되면 설정이 업데이트될 때까지 자동화가 실행되지 않습니다.

테이블 목록에서 정책을 선택하면 콘솔에서 다음 정보 탭이 표시됩니다.

- 세부 정보: 세부 정보 탭을 선택하여 정책 세부 정보와 배치 세부 정보를 확인합니다. 배치 테이블의 **Compliance** 열은 표시되는 클러스터의 규정 준수를 확인하는 링크를 제공합니다.

- 결과 탭을 선택하여 정책과 연결된 모든 클러스터의 테이블 목록을 확인합니다.

Message 열에서 **View details** 링크를 클릭하여 템플릿 세부 정보, 템플릿 **YAML** 및 관련 리소스를 확인합니다. 관련 리소스도 볼 수 있습니다. 기록 보기 링크를 클릭하여 위반 메시지와 마지막 보고서의 시간을 확인합니다.

다음 주제를 검토하여 보안 정책 생성 및 업데이트에 대해 자세히 알아보십시오.

- 보안 정책 관리
- 구성 정책 관리
- 게이트 키퍼 정책 관리
- 거버넌스를 위한 **Ansible Tower** 구성

자세한 내용은 관리에서 참조하십시오.

2.6.2. 거버넌스를 위한 Ansible Tower 구성

Red Hat Advanced Cluster Management for Kubernetes 거버넌스를 **Ansible Tower** 자동화와 통합하여 정책 위반 자동화를 생성할 수 있습니다. **Red Hat Advanced Cluster Management** 콘솔에서 자

동화를 구성할 수 있습니다.

- 사전 요구 사항
- 콘솔에서 정책 위반 자동화 생성
- CLI에서 정책 위반 자동화 생성

2.6.2.1. 사전 요구 사항

- **Red Hat OpenShift Container Platform 4.5** 이상
- **Ansible Tower** 버전 **3.7.3** 또는 이후 버전이 설치되어 있어야 합니다. 지원되는 최신 **Ansible Tower** 버전을 설치하는 것이 좋습니다. 자세한 내용은 **Red Hat Ansible Tower** 설명서를 참조하십시오.
- **Ansible Automation Platform Resource Operator**를 **hub** 클러스터에 설치하여 **Ansible** 작업을 거버넌스 프레임워크에 연결합니다. **AnsibleJob**을 사용하여 **Ansible Tower** 작업을 시작할 때 최상의 결과를 얻으려면 **Ansible Tower** 작업 템플릿이 멱등이어야 합니다. **Ansible Automation Platform Resource Operator**가 없는 경우 **Red Hat OpenShift Container Platform OperatorHub** 페이지에서 확인할 수 있습니다.

Ansible Tower 자동화 설치 및 구성에 대한 자세한 내용은 **Ansible** 작업 설정을참조하십시오.

2.6.2.2. 콘솔에서 정책 위반 자동화 생성

Red Hat Advanced Cluster Management hub 클러스터에 로그인한 후 탐색 메뉴에서 **Governance**를 선택한 다음 정책 탭을 클릭하여 정책 테이블을 확인합니다.

자동화 열에서 구성을 클릭하여 특정 정책에 대한 자동화 를 구성합니다. 정책 자동화 패널이 표시되면 자동화를 생성할 수 있습니다. **Ansible** 자격 증명 섹션에서 드롭다운 메뉴를 클릭하여 **Ansible** 자격 증명을 선택합니다. 인증 정보를 추가해야 하는 경우 인증 정보 관리 개요 를 참조하십시오.

참고: 이 인증 정보는 정책과 동일한 네임스페이스에 복사됩니다. 인증 정보는 생성된 **AnsibleJob** 리소스에서 자동화를 시작하는 데 사용됩니다. 콘솔의 인증 정보 섹션에서 **Ansible** 자격 증명 변경 사항이

자동으로 업데이트됩니다.

*인증 정보를 선택한 후 **Ansible** 작업 드롭다운 목록을 클릭하여 작업 템플릿을 선택합니다. **Extra variables** 섹션에서 **PolicyAutomation** 의 **extra_vars** 섹션에 있는 매개변수 값을 추가합니다. 자동화의 빈도를 선택합니다. 한 번 실행 모드를 선택하거나 자동화 비활성화 를 선택할 수 있습니다.*

- *수동 실행: 이 자동화를 수동으로 한 번 실행하도록 설정합니다. 자동화가 실행된 후 비활성화 됨으로 설정됩니다. 참고: 일정 빈도가 비활성화된 경우에만 수동 실행 모드를 선택할 수 있습니다.*
- *한 번 실행: 정책이 위반되면 자동화가 한 번 실행됩니다. 자동화가 실행된 후 비활성화 됨으로 설정됩니다. 자동화가 **disabled** 로 설정된 후 자동화를 수동으로 계속 실행해야 합니다. 한 번 모드를 실행하면 정책을 위반하는 클러스터 목록과 **target_cluster** 의 추가 변수가 자동으로 제공됩니다. **Ansible Tower** 작업 템플릿에는 **EXTRA VARIABLES (추가 변수)** 섹션에 대해 **PROMPT ON ECDHE(시작 시 확인)**가 활성화되어 있어야 합니다.*
- *자동 비활성화: 예약된 자동화가 비활성화 됨으로 설정되면 설정이 업데이트될 때까지 자동화가 실행되지 않습니다.*

***Submit** 을 선택하여 정책 위반 자동화를 저장합니다. **Ansible** 작업 세부 정보 측면 패널에서 작업 보기 링크를 선택하면 링크는 **Search** 페이지의 작업 템플릿으로 이동합니다. 자동화를 성공적으로 생성하면 **Automation** 열에 표시됩니다.*

참고: 관련 정책 자동화가 있는 정책을 삭제하면 정리의 일부로 정책 자동화가 자동으로 삭제됩니다.

콘솔에서 정책 위반 자동화가 생성됩니다.

2.6.2.3. CLI에서 정책 위반 자동화 생성

***CLI**에서 정책 위반 자동화를 구성하려면 다음 단계를 완료합니다.*

1. *터미널에서 **oc login** 명령을 사용하여 **Red Hat Advanced Cluster Management hub** 클러스터에 로그인합니다.*
2. *자동화를 추가할 정책을 찾아 생성합니다. 정책 이름과 네임스페이스를 확인합니다.*

3. 다음 샘플을 가이드로 사용하여 **PolicyAutomation** 리소스를 생성합니다.

```
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. 이전 샘플의 **Ansible** 작업 템플릿 이름은 **Policy Compliance Template** 입니다. 작업 템플릿 이름과 일치하도록 해당 값을 변경합니다.

5. **extra_vars** 섹션에서 **Ansible** 작업 템플릿에 전달하는 데 필요한 매개변수를 추가합니다.

6. 모드를 한 번 또는 비활성화 로 설정합니다. **once** 모드에서는 작업을 한 번 실행한 다음 모드가 **disabled** 로 설정됩니다.

   - **once mode:** 정책을 위반하면 자동화가 한 번 실행됩니다. 자동화가 실행된 후 비활성화 됨으로 설정됩니다. 자동화가 **disabled** 로 설정된 후 자동화를 수동으로 계속 실행해야 합니다. 한 번 모드를 실행하면 정책을 위반하는 클러스터 목록과 **target_cluster** 의 추가 변수가 자동으로 제공됩니다. **Ansible Tower** 작업 템플릿에는 **EXTRA VARIABLES** (추가 변수) 섹션에 대해 **PROMPT ON ECDHE**(시작 시 확인)가 활성화되어 있어야 합니다.

   - 자동 비활성화: 예약된 자동화가 비활성화 됨으로 설정되면 설정이 업데이트될 때까지 자동화가 실행되지 않습니다.

7. **policyRef** 를 정책 이름으로 설정합니다.

8. **Ansible Tower** 자격 증명이 포함된 이 **PolicyAutomation** 리소스와 동일한 네임스페이스에 시크릿을 생성합니다. 이전 예에서 시크릿 이름은 **ansible-tower** 입니다. 애플리케이션 라이프사이클의 샘플을 사용하여 시크릿을 생성하는 방법을 확인합니다.

9. *PolicyAutomation 리소스를 생성합니다.*

참고:

- *PolicyAutomation 리소스에 다음 주석을 추가하여 정책 자동화의 즉각적인 실행을 시작할 수 있습니다.*

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

- *정책이 once 모드이면 정책을 준수하지 않는 경우 자동화가 실행됩니다. target_clusters 라는 extra_vars 변수가 추가되고 값은 정책이 호환되지 않는 각 관리형 클러스터 이름의 배열입니다.*

2.6.3. GitOps를 사용하여 정책 배포

거버넌스 프레임워크를 사용하여 여러 관리 클러스터에 일련의 정책을 배포할 수 있습니다. 리포지토리의 정책에 기여하고 사용하여 오픈 소스 커뮤니티에 정책 수집 할 수 있습니다. 자세한 내용은 사용자 지정 정책 Contributing을 참조하십시오. 오픈 소스 커뮤니티의 안정 및 커뮤니티 폴더에 있는 각 정책은 NIST의 특수한 800-53 에 따라 추가로 구성됩니다.

GitOps를 사용하는 모범 사례를 계속 읽고 Git 리포지토리를 통해 정책 업데이트 및 생성을 자동화하고 추적합니다.

사전 요구 사항: 시작하기 전에 정책 수집 리포지토리를 분기해야 합니다.

- *로컬 리포지토리 사용자 정의*
- *로컬 리포지토리에 커밋*
- *클러스터에 정책 배포*
- *콘솔에서 GitOps 정책 배포 확인*

- *CLI에서 GitOps 정책 배포 확인*

2.6.3.1. 로컬 리포지토리 사용자 정의

안정적인 및 커뮤니티 정책을 단일 폴더에 통합하여 로컬 리포지토리를 사용자 정의합니다. 사용하지 않으려는 정책을 제거합니다. 로컬 리포지토리를 사용자 지정하려면 다음 단계를 완료합니다.

1. *리포지토리에 배포하려는 정책을 유지하기 위해 새 디렉터리를 생성합니다. **GitOps**의 기본 기본 분기의 로컬 정책 수집 리포지토리에 있는지 확인합니다. 다음 명령을 실행합니다.*

```
mkdir my-policies
```

2. *모든 안정 및 커뮤니티 정책을 **my-policies** 디렉터리에 복사합니다. 먼저 커뮤니티 정책부터 **stable** 폴더에 커뮤니티에서 사용할 수 있는 항목이 중복되는 경우부터 시작합니다. 다음 명령을 실행합니다.*

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

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

이제 단일 상위 디렉터리 구조에 모든 정책이 있으므로 포크의 정책을 편집할 수 있습니다.

팁:

- *사용하지 않으려는 정책을 제거하는 것이 좋습니다.*
- *다음 목록에서 정책 및 정책 정의에 대해 알아보십시오.*
  - *목적: 정책이 수행하는 작업을 파악합니다.*
  - *수정 작업: 정책은 규정 준수를 사용자에게만 알리거나 정책을 적용하고 변경합니까? **spec.remediationAction** 매개변수를 참조하십시오. 변경 사항이 적용되는 경우 기능적 기대치를 이해해야 합니다. 어떤 정책이 시행을 지원하는지 확인하십시오. 자세한 내용은 **Validate** 섹션을 참조하십시오.*

*참고: 정책에 대해 설정된 **spec.remediationAction** 은 개별 **spec.policy-***

templates 에 설정된 모든 수정 작업을 덮어씁니다.

- *배치: 정책이 배포되는 클러스터는 무엇입니까? 기본적으로 대부분의 정책은 environment: dev 레이블을 사용하여 클러스터를 대상으로 합니다. 일부 정책은 OpenShift Container Platform 클러스터 또는 다른 레이블을 대상으로 할 수 있습니다. 다른 클러스터를 포함하도록 라벨을 업데이트하거나 추가할 수 있습니다. 특정 값이 없는 경우 정책이 모든 클러스터에 적용됩니다. 하나의 클러스터 세트에 대해 한 가지 방식으로 구성된 정책을 사용하고 다른 클러스터 세트에 대해 다른 방식으로 구성하려는 경우 정책 복사본을 여러 개 생성하고 각 사본을 사용자 지정할 수도 있습니다.*

2.6.3.2. 로컬 리포지토리에 커밋

디렉터리에 대한 변경 사항을 충족한 후 클러스터에서 액세스할 수 있도록 변경 사항을 커밋 및 내보냅니다.

참고: 이 예제는 GitOps에서 정책을 사용하는 방법에 대한 기본을 표시하는 데 사용되므로 분기를 변경할 수 있는 다른 워크플로우가 있을 수 있습니다.

다음 단계를 완료합니다.

1. *터미널에서 git status 를 실행하여 이전에 생성한 디렉터리의 최근 변경 사항을 확인합니다. 다음 명령을 사용하여 커밋할 변경 사항 목록에 새 디렉터리를 추가합니다.*

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

2. *변경 사항을 커밋하고 메시지를 사용자 지정합니다. 다음 명령을 실행합니다.*

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

3. *GitOps에 사용되는 분기된 리포지토리 분기로 변경 사항을 내보냅니다. 다음 명령을 실행합니다.*

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

변경 사항이 커밋됩니다.

2.6.3.3. 클러스터에 정책 배포

변경 사항을 푸시한 후 **Kubernetes**용 **Red Hat Advanced Cluster Management**에 정책을 배포할 수 있습니다. 배포 후 **hub** 클러스터가 **Git** 리포지토리에 연결되어 있습니다. **Git** 리포지토리의 선택한 분기에 대한 추가 변경 사항은 클러스터에 반영됩니다.

참고: 기본적으로 **GitOps**와 함께 배포된 정책은 병합 조정 옵션을 사용합니다. 대신 교체 조정 옵션을 사용하려면 **apps.open-cluster-management.io/reconcile-option:** 주석을 **Subscription** 리소스에 추가합니다. 자세한 내용은 애플리케이션 라이프사이클 을 참조하십시오.

deploy.sh 스크립트는 허브 클러스터에 채널 및 서브스크립션 리소스를 생성합니다. 채널은 **Git** 리포지토리에 연결되고 서브스크립션은 채널을 통해 클러스터에 가져올 데이터를 지정합니다. 결과적으로 지정된 하위 디렉터리에 정의된 모든 정책이 허브에 생성됩니다. 서브스크립션을 통해 정책을 생성한 후 **Red Hat Advanced Cluster Management**는 정책을 분석하고 정의된 배치 규칙에 따라 정책이 적용되는 각 관리 대상 클러스터와 연결된 네임스페이스에 추가 정책 리소스를 생성합니다.

그런 다음 이 정책은 **hub** 클러스터의 해당 관리 클러스터 네임스페이스에서 관리 클러스터에 복사됩니다. 결과적으로 **Git** 리포지토리의 정책은 정책의 배치 규칙에 정의된 **clusterSelector** 와 일치하는 라벨이 있는 모든 관리 클러스터로 푸시됩니다.

다음 단계를 완료합니다.

1. **policy-collection** 디렉토리에서 다음 명령을 실행하여 디렉터리를 변경합니다.

```
cd deploy
```

2. 다음 명령을 사용하여 올바른 클러스터에 리소스를 생성하도록 **CLI(**명령줄 인터페이스)가 구성되어 있는지 확인합니다.

```
oc cluster-info
```

명령 출력은 **Red Hat Advanced Cluster Management**가 설치된 클러스터의 **API** 서버 세부 정보를 표시합니다. 올바른 **URL**이 표시되지 않으면 올바른 클러스터를 가리키도록 **CLI**를 구성합니다. 자세한 내용은 **OpenShift CLI** 사용을 참조하십시오.

3. 정책을 생성하여 액세스를 제어하고 정책을 구성합니다. 다음 명령을 실행합니다.

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

4. 다음 명령을 실행하여 클러스터에 정책을 배포합니다.

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

your-repository 를 **Git** 사용자 이름 또는 리포지토리 이름으로 바꿉니다.

참고: 참조를 위해 **deploy.sh** 스크립트의 전체 인수 목록은 다음 구문을 사용합니다.

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

각 인수에 대한 다음 설명을 확인합니다.

- **URL:** 기본 정책 수집 리포지토리에서 분기한 리포지토리의 **URL**입니다. 기본 **URL**은 **https://github.com/stolostron/policy-collection.git** 입니다.
- 분기: 을 가리키는 **Git** 리포지토리의 분기입니다. 기본 분기는 **main** 입니다.
- 하위 디렉터리 경로: 사용하려는 정책을 포함하도록 생성한 하위 디렉터리 경로입니다. 이전 샘플에서는 **my-policies** 하위 디렉터리를 사용했지만 시작할 폴더를 지정할 수도 있습니다. 예를 들어 **my-policies/AC-Access-Control** 을 사용할 수 있습니다. 기본 폴더는 **stable** 입니다.
- **namespace:** 허브 클러스터에서 리소스 및 정책이 생성되는 네임스페이스입니다. 이 명령은 **policy-namespace** 네임스페이스를 사용합니다. 기본 네임스페이스는 **policy** 입니다.
- 이름 접두사: 채널 및 서브스크립션 리소스의 접두사입니다. 기본값은 **demo-stable-policies** 입니다.

deploy.sh 스크립트를 실행하면 리포지토리에 액세스할 수 있는 모든 사용자가 분기에 대한 변경 사항을 커밋할 수 있으므로 클러스터의 기존 정책으로 변경 사항이 푸시됩니다.

참고: 서브스크립션을 사용하여 정책을 배포하려면 다음 단계를 완료하십시오.

1. open-cluster-management:subscription-admin ClusterRole을 서브스크립션을 생성한 사용자에게 바인딩합니다.

2. 서브스크립션에서 허용 목록을 사용하는 경우 다음 API 항목을 포함합니다.

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

2.6.3.4. 콘솔에서 GitOps 정책 배포 확인

콘솔에서 변경 사항이 정책에 적용되었는지 확인합니다. 콘솔에서 정책을 추가로 변경할 수도 있지만 서브스크립션 이 Git 리포지토리로 조정되면 변경 사항이 되돌아갑니다. 다음 단계를 완료합니다.

1. Red Hat Advanced Cluster Management 클러스터에 로그인합니다.

2. 탐색 메뉴에서 Governance 를 선택합니다.

3. 표에 배포한 정책을 찾습니다. GitOps를 사용하여 배포되는 정책에는 소스 열에 Git 레이블이 있습니다. 레이블을 클릭하여 Git 리포지토리의 세부 정보를 확인합니다.

2.6.3.4.1. CLI에서 GitOps 정책 배포 확인

다음 단계를 완료합니다.

1. 다음 정책 세부 정보를 확인합니다.

- *배포된 클러스터에 대한 특정 정책을 준수하거나 비준수하는 이유는 무엇입니까?*
- *올바른 클러스터에 정책이 적용됩니까?*
- *이 정책이 클러스터에 배포되지 않은 경우 이유는 무엇입니까?*

2. *생성하거나 수정한 **GitOps** 배포 정책을 식별합니다. **GitOps** 배포된 정책은 자동으로 적용되는 주석으로 식별할 수 있습니다. **GitOps** 배포 정책에 대한 주석은 다음 경로와 유사합니다.*

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

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

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

__GitOps__ 주석은 정책을 생성한 서브스크립션을 확인하는 데 유용합니다. 레이블을 기반으로 정책을 선택하는 런타임 쿼리를 작성할 수 있도록 정책에 고유한 레이블을 추가할 수도 있습니다.

예를 들어 다음 명령을 사용하여 정책에 라벨을 추가할 수 있습니다.

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

그런 다음 다음 명령을 사용하여 라벨이 있는 정책을 쿼리할 수 있습니다.

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

정책은 __GitOps__를 사용하여 배포됩니다.

2.6.4. 구성 정책의 템플릿 지원

구성 정책은 오브젝트 정의에 __Golang__ 텍스트 템플릿을 포함할 수 있도록 지원합니다. 이러한 템플릿은 해당 클러스터와 관련된 구성을 사용하여 __hub__ 클러스터 또는 대상 관리 클러스터에서 런타임 시 해결됩니다. 이를 통해 동적 콘텐츠로 구성 정책을 정의하고 대상 클러스터에 사용자 지정된 __Kubernetes__ 리소스에 알리거나 적용할 수 있습니다.

- 사전 요구 사항
- 템플릿 함수
- 구성 정책에서 허브 클러스터 템플릿 지원
  - 템플릿 처리
  - 재처리를 위한 특수 주석
  - 템플릿 처리 바이패스
  - hub 클러스터 및 관리형 클러스터 템플릿 비교

2.6.4.1. 사전 요구 사항

- 템플릿 구문은 Golang 템플릿 언어 사양을 준수해야 하며 확인된 템플릿에서 생성된 리소스 정의는 유효한 YAML이어야 합니다. 자세한 내용은 Package 템플릿에 대한 Golang 설명서를 참조하십시오. 템플릿 검증의 모든 오류는 정책 위반으로 인식됩니다. 사용자 지정 템플릿 함수를 사용하면 런타임 시 값이 교체됩니다.

2.6.4.2. 템플릿 함수

리소스별 및 일반 조회 템플릿 기능과 같은 템플릿 기능은 hub 클러스터에서 Kubernetes 리소스를 참조할 수 있습니다({{hub ... hub}} 구분 기호 사용), 또는 관리 클러스터({{ ... }} 구분 사용). 자세한 내용은 구성 정책의 허브 클러스터 템플릿 지원을 참조하십시오. 리소스별 기능은 편의를 위해 사용되며 리소스의 콘텐츠에 보다 쉽게 액세스할 수 있습니다. 일반 함수인 lookup 을 사용하는 경우 조회되는 리소스의 YAML 구조에 대해 잘 알고 있는 것이 가장 좋습니다. 이러한 함수 외에도 base64encode,base64decode,들여 쓰기,자동 들여 쓰기,ToInt,to Bool 등의 유틸리티 함수도 사용할 수 있습니다.

YAML 구문을 사용하여 템플릿을 준수하려면 따옴표 또는 블록 문자(| 또는 >)를 사용하여 정책 리소스에서 문자열로 설정해야 합니다. 이로 인해 확인된 템플릿 값도 문자열이 됩니다. 이를 재정의하려면 템플릿의 최종 기능으로 toInt 또는 toBool 을 사용하여 값을 각각 정수 또는 부울로 해석하도록 강제 적용하는 추가 처리를 시작합니다.

지원되는 사용자 지정 템플릿 함수 중 일부에 대한 설명 및 예를 보려면 계속 읽습니다.

- *fromSecret 함수*
- *fromConfigmap 함수*
- *fromClusterClaim 함수*
- *lookup 함수*
- *base64enc 기능*
- *base64dec 함수*
- *들여쓰기 함수*
- *autoindent 기능*
- *ToInt 기능*
- *toBool 기능*
- *기능 보호*

2.6.4.2.1. fromSecret 함수

fromSecret 함수는 시크릿에 지정된 데이터 키의 값을 반환합니다. 함수의 다음 구문을 확인합니다.

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

*이 기능을 사용하는 경우 **Kubernetes Secret** 리소스의 **namespace, name, data** 키를 입력합니다. 허브 클러스터 템플릿에서 함수를 사용할 때 정책에 사용되는 동일한 네임스페이스를 사용해야 합니다. 자세한 내용은 구성 정책의 허브 클러스터 템플릿 지원을 참조하십시오.*

*참고: **hub** 클러스터 템플릿과 함께 이 함수를 사용하면 보호 기능을 사용하여 출력이 자동으로 암호화됩니다.*

*Kubernetes Secret 리소스가 대상 클러스터에 없는 경우 정책 위반이 표시됩니다. 대상 클러스터에 데이터 키가 없으면 값이 빈 문자열이 됩니다. 대상 클러스터에 **Secret** 리소스를 적용하는 다음 구성 정책을 확인합니다. **PASSWORD** 데이터 키의 값은 대상 클러스터의 보안을 참조하는 템플릿입니다.*

```
apiVersion: policy.open-cluster-management.io/v1
kind: ConfigurationPolicy
metadata:
  name: demo-fromsecret
  namespace: test
spec:
  namespaceSelector:
    exclude:
    - kube-*
    include:
    - default
  object-templates:
  - complianceType: musthave
    objectDefinition:
      apiVersion: v1
      data:
        USER_NAME: YWRtaW4=
        PASSWORD: '{{ fromSecret "default" "localsecret" "PASSWORD" }}'
      kind: Secret
      metadata:
        name: demosecret
        namespace: test
      type: Opaque
  remediationAction: enforce
  severity: low
```

2.6.4.2.2. fromConfigmap 함수

fromConfigmap 함수는 ConfigMap에 지정된 데이터 키의 값을 반환합니다. 함수의 다음 구문을 확인합니다.

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

*이 기능을 사용하는 경우 **Kubernetes ConfigMap** 리소스의 네임스페이스, 이름 및 데이터 키를 입력합니다. 허브 클러스터 템플릿의 함수를 사용하여 정책에 사용되는 동일한 네임스페이스를 사용해야 합니*

다. 자세한 내용은 구성 정책의 허브 클러스터 템플릿 지원을 참조하십시오. **Kubernetes ConfigMap** 리소스가 대상 클러스터에 없는 경우 정책 위반이 표시됩니다. 대상 클러스터에 데이터 키가 없으면 값이 빈 문자열이 됩니다. 대상 관리 클러스터에 **Kubernetes** 리소스를 적용하는 다음 구성 정책을 확인합니다. **log-file** 데이터 키의 값은 **ConfigMap**에서 **log-file** 의 값을 검색하고, **default** 네임스페이스에서 **logs-config**, **log-level** 이 데이터 키 로그 수준으로 설정된 템플릿입니다.

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

2.6.4.2.3. fromClusterClaim 함수

fromClusterClaim 함수는 **ClusterClaim** 리소스의 **Spec.Value** 값을 반환합니다. 함수의 다음 구문을 확인합니다.

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

이 기능을 사용하는 경우 **Kubernetes ClusterClaim** 리소스의 이름을 입력합니다. **ClusterClaim** 리소스가 없는 경우 정책 위반이 표시됩니다. 대상 관리 클러스터에 **Kubernetes** 리소스를 적용하는 구성 정책의 다음 예제를 봅니다. 플랫폼 데이터 키의 값은 **platform.open-cluster-management.io** 클러스터 클레임의 가치를 검색하는 템플릿입니다. 마찬가지로 **ClusterClaim** 에서 제품 및 버전 값을 검색합니다.

```
apiVersion: policy.open-cluster-management.io/v1
kind: ConfigurationPolicy
metadata:
  name: demo-clusterclaims
  namespace: default
```

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

2.6.4.2.4. lookup 함수

lookup 함수는 **Kubernetes** 리소스를 **JSON** 호환 맵으로 반환합니다. 요청된 리소스가 없으면 빈 맵이 반환됩니다. 리소스가 없고 다른 템플릿 함수에 값이 제공되면 다음과 같은 **error: invalid value;** 예상 문자열 이 표시될 수 있습니다.

참고: 기본 템플릿 함수를 사용하므로 나중에 템플릿 함수에 올바른 유형이 제공됩니다. 오픈 소스 커뮤니티 기능 섹션을 참조하십시오.

함수의 다음 구문을 확인합니다.

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

이 기능을 사용하는 경우 **API** 버전, 종류, 네임스페이스, **Kubernetes** 리소스의 이름을 입력합니다. 허브 클러스터 템플릿 내에서 정책에 사용되는 동일한 네임스페이스를 사용해야 합니다. 자세한 내용은 구성 정책의 허브 클러스터 템플릿 지원을 참조하십시오. 대상 관리 클러스터에 **Kubernetes** 리소스를 적용하는 구성 정책의 다음 예제를 봅니다. **metrics-url** 데이터 키의 값은 **default** 네임스페이스에서 **v1/Service Kubernetes** 리소스 지표 를 검색하고 쿼리된 리소스에서 **Spec.ClusterIP** 의 값으로 설정된 템플릿입니다.

```
apiVersion: policy.open-cluster-management.io/v1
kind: ConfigurationPolicy
metadata:
  name: demo-lookup
  namespace: test-templates
```

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

2.6.4.2.5. base64enc 기능

base64enc 함수는 입력 데이터 문자열 의 **base64** 인코딩 값을 반환합니다. 함수의 다음 구문을 확인합니다.

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

이 함수를 사용하면 문자열 값을 입력합니다. **base64enc** 함수를 사용하는 구성 정책의 다음 예제를 확인합니다.

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

2.6.4.2.6. base64dec 함수

base64dec 함수는 입력 **enc-data** 문자열 의 **base64** 디코딩된 값을 반환합니다. 함수의 다음 구문을 확인합니다.

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

이 함수를 사용하면 문자열 값을 입력합니다. **base64dec** 함수를 사용하는 구성 정책의 다음 예제를 확인합니다.

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

2.6.4.2.7. 들여쓰기 함수

indent 함수는 **padded** 데이터 문자열 을 반환합니다. 함수의 다음 구문을 확인합니다.

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

이 함수를 사용하면 특정 수의 공백을 사용하여 데이터 문자열을 입력합니다. **indent** 함수를 사용하는 구성 정책의 다음 예제를 확인합니다.

```
apiVersion: policy.open-cluster-management.io/v1
kind: ConfigurationPolicy
metadata:
  name: demo-fromsecret
  namespace: test
spec:
  namespaceSelector:
    exclude:
```

```
    - kube-*
    include:
    - default
  object-templates:
  - complianceType: musthave
    objectDefinition:
    ...
    data:
      Ca-cert:  |
        {{ ( index ( lookup "v1" "Secret" "default" "mycert-tls"  ).data  "ca.pem"  ) |  base64dec |
indent 4  }}
```

2.6.4.2.8. autoindent 기능

autoindent 함수는 템플릿 전의 공백 수에 따라 선행 공백의 수를 자동으로 결정하는 들여 쓰기 함수처럼 작동합니다. **autoindent** 함수를 사용하는 구성 정책의 다음 예제를 확인합니다.

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

2.6.4.2.9. toInt 기능

toInt 함수는 입력 값의 정수 값을 캐스팅하고 반환합니다. 또한 템플릿의 마지막 기능인 경우 소스 컨텐츠를 추가로 처리합니다. 이는 값이 **YAML**에서 정수로 해석되도록 하기 위한 것입니다. 함수의 다음 구문을 확인합니다.

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

이 함수를 사용하면 정수로 캐스팅해야 하는 데이터를 입력합니다. **toInt** 함수를 사용하는 구성 정책의 다음 예제를 확인합니다.

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

2.6.4.2.10. toBool 기능

toBool 함수는 입력 문자열을 부울로 변환하고 부울을 반환합니다. 또한 템플릿의 마지막 기능인 경우 소스 컨텐츠를 추가로 처리합니다. 이는 값이 YAML에서 부울로 해석되도록 하기 위한 것입니다. 함수의 다음 구문을 확인합니다.

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

이 함수를 사용하는 경우 부울로 변환해야 하는 문자열 데이터를 입력합니다. toBool 함수를 사용하는 구성 정책의 다음 예제를 확인합니다.

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

2.6.4.2.11. 기능 보호

protect 기능을 사용하면 허브 클러스터 정책 템플릿의 문자열을 암호화할 수 있습니다. 정책을 평가할 때 관리 클러스터에서 자동으로 암호 해독됩니다. 보호 기능을 사용하는 구성 정책의 다음 예제를 확인합니다.

```
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}}
```

이전 YAML 예제에는 lookup 함수를 사용하도록 정의된 기존 hub 클러스터 정책 템플릿이 있습니다. 관리형 클러스터 네임스페이스의 복제 정책에서 값은 $ocm_encrypted:okrrBqt72ol+3WT/0vxel3vGa+wpL7ZxFMLvL204구문과 유사합니다.

사용된 각 암호화 알고리즘은 256비트 키를 사용하는 AES-CBC입니다. 각 암호화 키는 관리 클러스터별로 고유하며 30일마다 자동으로 순환됩니다.

이렇게 하면 암호 해독된 값이 관리 클러스터의 정책에 저장되지 않습니다.

즉시 순환하려면 hub 클러스터의 관리형 클러스터 네임스페이스의 policy-encryption-key Secret에서 policy.open-cluster-management.io/last-rotated 주석을 삭제합니다. 그런 다음 새 암호화 키를 사용하도록 정책이 다시 처리됩니다.

2.6.4.3. 구성 정책에서 허브 클러스터 템플릿 지원

Red Hat Advanced Cluster Management는 대상 클러스터에 동적으로 사용자 정의된 관리형 클러스터 템플릿 외에도 허브 클러스터 템플릿을 지원하여 hub 클러스터 값을 사용하여 구성 정책을 정의합니다. 이 조합을 사용하면 정책 정의의 각 대상 클러스터 또는 하드 코드 구성 값에 대해 별도의 정책을 생성해야 할 필요성이 줄어듭니다.

Hub 클러스터 템플릿은 Golang 텍스트 템플릿 사양을 기반으로 하며 {{hub ... hub}} 구분자는 구성 정책의 허브 클러스터 템플릿을 나타냅니다.

보안을 위해 허브 클러스터 템플릿의 리소스별 및 일반 조회 기능 모두 허브 클러스터 정책의 네임스페이스로 제한됩니다. 자세한 내용은 허브 및 관리형 클러스터 템플릿 비교 를 참조하십시오.

중요: 허브 클러스터 템플릿을 사용하여 시크릿 또는 기타 민감한 데이터를 전파하면 허브 클러스터의 관리 클러스터 네임 스페이스와 해당 정책이 배포되는 관리형 클러스터에 민감한 데이터가 있습니다. 정책에서 템플릿 콘텐츠가 확장되며 정책은 OpenShift Container Platform ETCD 암호화 지원에 의해 암호화되지 않습니다. 이 문제를 해결하려면 시크릿에서 값을 자동으로 암호화하거나 다른 값을 암호화하도록 보호하는 fromSecret 을 사용합니다.

2.6.4.3.1. 템플릿 처리

구성 정책 정의에는 허브 클러스터와 관리형 클러스터 템플릿이 모두 포함될 수 있습니다. Hub 클러스터 템플릿은 hub 클러스터에서 먼저 처리된 다음 확인된 허브 클러스터 템플릿이 있는 정책 정의가 대상 클러스터로 전파됩니다. 관리형 클러스터에서 ConfigurationPolicyController 는 정책 정의에서 모든 관리 클러스터 템플릿을 처리한 다음 완전히 확인된 오브젝트 정의를 적용하거나 확인합니다.

2.6.4.3.2. 재처리를 위한 특수 주석

정책은 생성 시 또는 업데이트 후에만 허브 클러스터에서 처리됩니다. 따라서 허브 클러스터 템플릿은 정책 생성 또는 업데이트 시 참조된 리소스의 데이터로만 확인됩니다. 참조된 리소스에 대한 변경 사항은 정책과 자동으로 동기화되지 않습니다.

특정 주석인 policy.open-cluster-management.io/trigger-update 를 사용하여 템플릿에서 참조하는 데이터의 변경 사항을 표시할 수 있습니다. 특수 주석 값을 변경하면 템플릿 처리가 시작되고, 참조된 리소스의 최신 콘텐츠가 관리 클러스터의 처리를 위한 전파자인 정책 정의에서 읽고 업데이트됩니다. 이 주석을 사용하는 일반적인 방법은 값을 한 번에 하나씩 늘리는 것입니다.

2.6.4.3.3. 템플릿 처리 바이패스

Red Hat Advanced Cluster Management에서 처리하지 않으려는 템플릿이 포함된 정책을 생성할 수 있습니다. 기본적으로 Red Hat Advanced Cluster Management는 모든 템플릿을 처리합니다.

hub 클러스터의 템플릿 처리를 바이패스하려면 {{ template content }} 를 {{ '{{ template content }}' }} 로 변경해야 합니다.

또는 정책의 Configuration Policy 섹션에 다음 주석을 추가할 수 있습니다.policy.open-cluster-management.io/disable-templates: "true". 이 주석이 포함된 경우 이전 해결 방법이 필요하지 않습니

*다. **ConfigurationPolicy** 에 대해 템플릿 처리를 바이패스합니다.*

허브 클러스터 및 관리형 클러스터 템플릿을 비교하려면 다음 표를 참조하십시오.

2.6.4.3.4. hub 클러스터 및 관리형 클러스터 템플릿 비교

표 2.6. 비교표

템플릿	hub cluster	관리형 클러스터
구문	Golang 텍스트 템플릿 사양	Golang 텍스트 템플릿 사양
구분 기호	{{Hub ... hub}}	{{ ... }}
context	런타임 시 정책이 전파되는 대상 클러스터의 이름으로 확인되는 **.ManagedClusterName** 변수를 사용할 수 있습니다.	컨텍스트 변수가 없음
액세스 제어	**Policy** 리소스와 동일한 네임스페이스에 있는 네임스페이스가 지정된 Kubernetes 오브젝트만 참조할 수 있습니다.	클러스터의 모든 리소스를 참조할 수 있습니다.
함수	Kubernetes 리소스 및 문자열 조작에 대한 동적 액세스를 지원하는 템플릿 함수 세트입니다. 자세한 내용은 템플릿 함수 를 참조하십시오. 조회 제한 사항은 액세스 제어 행을 참조하십시오. hub 클러스터의 **fromSecret** 템플릿 함수는 결과 값을 관리 클러스터 네임스페이스에 복제 정책의 암호화된 문자열로 저장합니다. 동일한 호출에서는 다음 구문을 사용할 수 있습니다. **{{hub " (lookup "v1" "Secret" "Secret" "default" "my-hub-secret").data.message | hub}}**	템플릿 기능 세트는 Kubernetes 리소스 및 문자열 조작에 대한 동적 액세스를 지원합니다. 자세한 내용은 템플릿 함수 를 참조하십시오.

템플릿	hub cluster	관리형 클러스터
함수 출력 스토리지	템플릿 함수의 출력은 관리 클러스터와 동기화되기 전에 hub 클러스터의 적용 가능한 각 관리 클러스터 네임스페이스의 **Policy** 리소스 오브젝트에 저장됩니다. 즉, 템플릿 함수의 중요한 결과는 hub 클러스터의 **Policy** 리소스 오브젝트에 대한 읽기 액세스 권한이 있는 모든 사용자가 읽을 수 있고, 관리형 클러스터에서 **ConfigurationPolicy** 리소스 오브젝트를 사용한 읽기 액세스 권한이 있습니다. 또한 etcd 암호화 가 활성화된 경우 **Policy** 및 **ConfigurationPolicy** 리소스 오브젝트가 암호화되지 않습니다. 중요한 출력을 반환하는 템플릿 함수(예: 시크릿에서)를 사용할 때는 이 점을 주의 깊게 고려하는 것이 가장 좋습니다.	템플릿 함수의 출력은 정책 관련 리소스 오브젝트에 저장되지 않습니다.
processing	처리는 복제된 정책을 클러스터에 전파하는 동안 hub 클러스터의 런타임 시 수행됩니다. 정책 내의 정책 및 허브 클러스터 템플릿은 템플릿을 만들거나 업데이트할 때만 hub 클러스터에서 처리됩니다.	처리는 관리되는 클러스터의 **ConfigurationPolicyController** 에서 수행됩니다. 정책은 정기적으로 처리되며, 이 정책은 참조된 리소스의 데이터로 확인된 오브젝트 정의를 자동으로 업데이트합니다.
처리 오류	허브 클러스터 템플릿의 오류는 정책이 적용되는 관리 클러스터의 위반으로 표시됩니다.	관리형 클러스터 템플릿의 오류는 위반이 발생한 특정 대상 클러스터에서 위반으로 표시됩니다.

2.6.5. 거버넌스 메트릭

정책 프레임워크는 정책 배포 및 규정 준수를 보여주는 지표를 노출합니다. hub 클러스터에서 policy_governance_info 지표를 사용하여 추세를 보고 정책 실패를 분석합니다.

2.6.5.1. 지표 개요

policy_governance_info 는 OpenShift Container Platform 모니터링에 의해 수집되며 일부 집계 데이터는 활성화된 경우 Red Hat Advanced Cluster Management Observability에 의해 수집됩니다.

참고: 관찰 기능이 활성화된 경우 Grafana Explore 페이지에서 메트릭 쿼리를 입력할 수 있습니다.

정책을 생성할 때 루트 정책을 생성합니다. 프레임워크는 관리 클러스터에 정책을 배포하기 위해 전파된 정책을 생성할 위치를 결정하기 위해 **PlacementRules** 및 **PlacementBindings** 뿐만 아니라 루트 정책을 감시합니다. 루트 정책과 전파 정책 모두에 대해 정책이 준수되는 경우 **0** 의 지표가 기록되고 **1** 이 호환되지 않는 경우 **1**이 기록됩니다.

policy_governance_info 메트릭은 다음 레이블을 사용합니다.

- **type:** 레이블 값은 **root** 이거나 전파됩니다.
- **Policy:** 연결된 **root** 정책의 이름입니다.
- **policy_namespace:** 루트 정책이 정의된 **hub** 클러스터의 네임스페이스입니다.
- **cluster_namespace:** 정책이 배포된 클러스터의 네임스페이스입니다.

이러한 레이블과 값을 사용하면 클러스터에서 발생하는 많은 상황을 추적하기 어려울 수 있는 쿼리를 사용할 수 있습니다.

참고: 메트릭이 필요하지 않으며 성능 또는 보안에 대한 우려 사항이 있는 경우 이 기능을 사용하지 않도록 설정할 수 있습니다. **propagator** 배포에서 **DISABLE_REPORT_METRICS** 환경 변수를 **true** 로 설정합니다. 또한 **policy_governance_info** 지표를 사용자 지정 지표로 **observability allowlist**에 추가할 수도 있습니다. 자세한 내용은 *사용자 정의 메트릭 추가* 를 참조하십시오.

2.6.6. 보안 정책 관리

지정된 보안 표준, 카테고리 및 제어를 기반으로 클러스터 컴플라이언스를 보고하고 검증할 수 있는 보안 정책을 생성합니다. **Red Hat Advanced Cluster Management for Kubernetes**에 대한 정책을 생성하려면 관리 클러스터에 **YAML** 파일을 생성해야 합니다.

참고: 의 기존 정책을 복사하여 **Policy YAML** 에 붙여넣을 수 있습니다. 기존 정책을 붙여넣을 때 매개변수 필드의 값이 자동으로 입력됩니다. 검색 기능을 사용하여 정책 **YAML** 파일에서 콘텐츠를 검색할 수도 있습니다.

다음 섹션을 확인합니다.

- 보안 정책 생성
  - 명령줄 인터페이스에서 보안 정책 생성
  - CLI에서 보안 정책 보기
  - 콘솔에서 클러스터 보안 정책 생성
  - 콘솔에서 보안 정책 보기
  - CLI에서 정책 세트 생성
  - 콘솔에서 정책 세트 생성
- 보안 정책 업데이트
  - 보안 정책 비활성화
- 보안 정책 삭제
  - 콘솔에서 정책 세트 삭제

2.6.6.1. 보안 정책 생성

CLI(명령줄 인터페이스) 또는 콘솔에서 보안 정책을 생성할 수 있습니다.

필수 액세스: 클러스터 관리자

중요: 특정 클러스터에 정책을 적용하려면 배치 규칙 및 배치 바인딩을 정의해야 합니다. **Cluster selector** 필드에 유효한 값을 입력하여 **PlacementRule** 및 **PlacementBinding** 을 정의합니다. 유효한 표

*현식은 **Kubernetes** 문서의 세트 기반 요구 사항을 지원하는 리소스를 참조하십시오. **Red Hat Advanced Cluster Management** 정책에 필요한 오브젝트 정의를 확인합니다.*

- ***PlacementRule:** 정책을 배포해야 하는 클러스터 선택기를 정의합니다.*
- ***PlacementBinding:** 배치를 배치 규칙에 바인딩합니다.*

*정책 개요에서 정책 **YAML** 파일에 대한 자세한 설명을 확인하십시오.*

2.6.6.1.1. 명령줄 인터페이스에서 보안 정책 생성

***CLI**(명령줄 인터페이스)에서 정책을 생성하려면 다음 단계를 완료합니다.*

1. *다음 명령을 실행하여 정책을 생성합니다.*

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

2. *정책이 사용하는 템플릿을 정의합니다. 템플릿을 정의할 템플릿 필드를 추가하여 **.yaml** 파일을 편집합니다. 정책은 다음 **YAML** 파일과 유사합니다.*

```
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  name: policy1
spec:
  remediationAction: "enforce" # or inform
  disabled: false # or true
  namespaces:
    include: ["default"]
    exclude: ["kube*"]
  policy-templates:
    - objectDefinition:
        apiVersion: policy.open-cluster-management.io/v1
        kind: ConfigurationPolicy
        metadata:
          namespace: kube-system # will be inferred
          name: operator
        spec:
          remediationAction: "inform"
          object-templates:
          - complianceType: "musthave" # at this level, it means the role must exist and
must have the following rules
            apiVersion: rbac.authorization.k8s.io/v1
```

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

3. *PlacementRule 을 정의합니다. PlacementRule 을 변경하여 clusterNames, 또는 clusterLabels 에서 정책을 적용해야 하는 클러스터를 지정해야 합니다. 배치 규칙 샘플 개요보기*

PlacementRule 은 다음 내용과 유사할 수 있습니다.

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

4. *정책 및 PlacementRule 을 바인딩하도록 PlacementBinding 을 정의합니다. PlacementBinding 은 다음 YAML 샘플과 유사할 수 있습니다.*

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

2.6.6.1.1.1. CLI에서 보안 정책 보기

CLI에서 보안 정책을 보려면 다음 단계를 완료합니다.

1. *다음 명령을 실행하여 특정 보안 정책에 대한 세부 정보를 확인합니다.*

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

2. *다음 명령을 실행하여 보안 정책에 대한 설명을 확인합니다.*

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

2.6.6.1.2. 콘솔에서 클러스터 보안 정책 생성

***Red Hat Advanced Cluster Management**에 로그인한 후 관리 페이지로 이동 하여 정책 생성을 클릭합니다.*

*콘솔에서 새 정책을 생성하면 **YAML** 파일도 **YAML** 편집기에서 생성됩니다. **YAML** 편집기를 보려면 **Create** 정책 양식의 시작 부분에 있는 토글을 선택하여 활성화합니다.*

***Create policy form**을 완료한 후 **Submit** 버튼을 선택합니다.*

***YAML** 파일은 다음 정책과 유사할 수 있습니다.*

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

```
    metadata:
      name: pod1
    spec:
      containers:
      - name: pod-name
        image: 'pod-image'
        ports:
        - containerPort: 80
  remediationAction: enforce
  disabled: false

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

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

정책 생성을 클릭합니다. 콘솔에서 보안 정책을 생성합니다.

2.6.6.1.2.1. 콘솔에서 보안 정책 보기

*콘솔에서 모든 보안 정책 및 해당 상태를 확인합니다. **Governance** 페이지로 이동하여 정책의 테이블 목록을 확인합니다. 참고: 정책 탭 또는 클러스터 위반 탭을 선택하여 정책의 표 목록을 필터링할 수 있습니다.*

자세한 내용을 보려면 정책 중 하나를 선택합니다. 세부 정보 , 클러스터 및 템플릿 탭이 표시됩니다. 클러스터 또는 정책 상태를 확인할 수 없는 경우 다음 메시지가 표시됩니다. 상태 없음.

2.6.6.1.3. CLI에서 정책 세트 생성

기본적으로 정책 세트는 정책 또는 배치 없이 생성됩니다. 정책 세트에 대한 배치를 생성하고 클러스터에 존재하는 정책이 하나 이상 있어야 합니다. 정책 세트를 생성하면 다양한 정책을 추가할 수 있습니다. 다음 명령을 실행하여 **CLI**에서 설정된 정책을 생성합니다.

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

2.6.6.1.4. 콘솔에서 정책 세트 생성

탐색 메뉴에서 **Governance** 를 선택합니다. 그런 다음 정책 설정 탭을 선택합니다. **Create policy set** 버튼을 선택하고 양식을 작성합니다. 정책 세트 세부 정보를 추가한 후 **Submit** 버튼을 선택합니다.

배포를 위해 정책 생성기가 필요한 안정된 정책 세트를 확인합니다. **PolicySets-- Stable**.

2.6.6.2. 보안 정책 업데이트

다음 섹션을 확인하여 보안 정책을 업데이트하는 방법을 알아봅니다.

2.6.6.2.1. CLI에서 설정된 정책에 정책 추가

다음 명령을 실행하여 정책 세트를 편집합니다. **kubectl edit policysets your-policyset-name**

정책 세트의 **policies** 섹션에 정책 이름을 추가합니다. **kubectl apply -f your-added-policy.yaml** 명령을 사용하여 다음 명령을 사용하여 정책 세트의 배치 섹션에 추가된 정책을 적용합니다. **PlacementBinding** 및 **PlacementRule** 이 생성됩니다. 참고: 배치 바인딩을 삭제하면 정책 세트에 따라 정책이 계속 배치됩니다.

2.6.6.2.2. 콘솔에서 설정된 정책에 정책 추가

Policy sets 탭을 선택하여 설정된 정책에 정책을 추가합니다. 작업 아이콘을 선택하고 편집 을 선택합니다. **Edit policy set form**이 나타납니다.

양식의 **Policies** 섹션으로 이동하여 정책 세트에 추가할 정책을 선택합니다.

2.6.6.2.3. 보안 정책 비활성화

정책은 기본적으로 활성화되어 있습니다. 콘솔에서 정책을 비활성화합니다.

Kubernetes 콘솔의 **Red Hat Advanced Cluster Management**에 로그인한 후 **Governance** 페이지로 이동하여 정책의 표 목록을 확인합니다.

Actions 아이콘 **> Disable policy** 를 선택합니다. 정책 비활성화 대화 상자가 나타납니다.

Disable policy 를 클릭합니다. 정책이 비활성화되어 있습니다.

2.6.6.3. 보안 정책 삭제

CLI 또는 콘솔에서 보안 정책을 삭제합니다.

- **CLI**에서 보안 정책을 삭제합니다.

  a. 다음 명령을 실행하여 보안 정책을 삭제합니다.

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

  정책이 삭제되면 대상 클러스터 또는 클러스터에서 제거됩니다. 다음 명령을 실행하여 정책이 제거되었는지 확인합니다. **kubectl get policy <securitypolicy-name> -n <open-cluster-management-namespace>**

- 콘솔에서 보안 정책을 삭제합니다.

  탐색 메뉴에서 **Governance** 를 클릭하여 정책의 테이블 목록을 확인합니다. 정책 위반 테이블에서 삭제할 정책의 **Actions** 아이콘을 클릭합니다.

  제거를 클릭합니다. 정책 제거 대화 상자에서 정책제거를 클릭합니다.

2.6.6.3.1. 콘솔에서 정책 세트 삭제

Policy sets 탭에서 정책 세트의 **Actions** 아이콘을 선택합니다. 삭제 를 클릭하면 정책 집합을 영구적으로 삭제하시겠습니까**?** 대화 상자가 나타납니다.

삭제 버튼을 클릭합니다.

다른 정책을 관리하려면 자세한 내용은 보안 정책 관리를 참조하십시오. 정책에 대한 자세한 내용은 감독을 참조하십시오.

2.6.7. 구성 정책 관리

구성 정책 생성, 적용, 보기 및 업데이트 방법을 학습합니다. 다음 리소스는 구성 정책입니다.

- 메모리 사용량 정책
- 네임스페이스 정책
- 이미지 취약점 정책
- **Pod** 정책
- **Pod** 보안 정책
- 역할 정책
- 역할 바인딩 정책
- **SCC**(보안 콘텐츠 제약 조건) 정책
- **ETCD** 암호화 정책
- 컴플라이언스 **Operator** 정책

필수 액세스: 관리자 또는 클러스터 관리자

- *구성 정책 생성*
  - *CLI에서 구성 정책 생성*
  - *CLI에서 구성 정책 보기*
  - *콘솔에서 구성 정책 생성*
  - *콘솔에서 구성 정책 보기*
- *구성 정책 업데이트*
  - *구성 정책 비활성화*
- *구성 정책 삭제*

2.6.7.1. 구성 정책 생성

CLI(명령줄 인터페이스) 또는 콘솔에서 구성 정책에 대한 YAML 파일을 생성할 수 있습니다. 다음 섹션을 보고 구성 정책을 생성합니다.

2.6.7.1.1. CLI에서 구성 정책 생성

(CLI)에서 구성 정책을 생성하려면 다음 단계를 완료합니다.

1. *구성 정책에 대한 **YAML** 파일을 생성합니다. 다음 명령을 실행합니다.*

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

구성 정책은 다음 정책과 유사할 수 있습니다.

```
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  name: policy-1
  namespace: kube-system
spec:
  namespaces:
    include: ["default", "kube-*"]
    exclude: ["kube-system"]
  remediationAction: inform
  disabled: false
  complianceType: musthave
  object-templates:
  ...
```

2. 다음 명령을 실행하여 정책을 적용합니다.

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

3. 다음 명령을 실행하여 정책을 확인하고 나열합니다.

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

구성 정책이 생성됩니다.

2.6.7.1.2. CLI에서 구성 정책 보기

CLI에서 구성 정책을 보려면 다음 단계를 완료합니다.

1. 다음 명령을 실행하여 특정 구성 정책에 대한 세부 정보를 확인합니다.

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

2. 다음 명령을 실행하여 구성 정책에 대한 설명을 확인합니다.

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

2.6.7.1.3. 콘솔에서 구성 정책 생성

콘솔에서 구성 정책을 생성하면 **YAML** 파일도 **YAML** 편집기에서 생성됩니다.

콘솔에서 클러스터에 로그인하고 탐색 메뉴에서 **Governance** 를 선택합니다.

정책 생성을 클릭합니다. 사양 매개변수에 대한 구성 정책 중 하나를 선택하여 생성할 정책을 지정합니다.

정책 양식을 작성하여 구성 정책 생성을 계속합니다. 다음 필드에 적절한 값을 입력하거나 선택합니다.

- 이름
- 사양
- 클러스터 선택기
- 수정 작업
- 표준
- 카테고리
- 제어

생성을 클릭합니다. 구성 정책이 생성됩니다.

2.6.7.1.4. 콘솔에서 구성 정책 보기

콘솔에서 모든 구성 정책 및 해당 상태를 확인합니다.

콘솔에서 클러스터에 로그인한 후 **Governance** 를 선택하여 정책의 테이블 목록을 확인합니다. 참고: 모든 정책 탭 또는 클러스터 위반 탭을 선택하여 정책의 표 목록을 필터링할 수 있습니다.

자세한 내용을 보려면 정책 중 하나를 선택합니다. 세부 정보 , 클러스터 및 템플릿 탭이 표시됩니다.

2.6.7.2. 구성 정책 업데이트

다음 섹션을 확인하여 구성 정책을 업데이트하는 방법을 알아봅니다.

2.6.7.2.1. 구성 정책 비활성화

구성 정책을 비활성화합니다. 앞서 언급한 지침과 유사하게 로그인하여 **Governance** 페이지로 이동합니다.

테이블 목록에서 구성 정책의 **Actions** 아이콘을 선택한 다음 **Disable** 를 클릭합니다. 정책 비활성화 대화 상자가 나타납니다.

Disable policy 를 클릭합니다.

구성 정책이 비활성화되어 있습니다.

2.6.7.3. 구성 정책 삭제

CLI 또는 콘솔에서 구성 정책을 삭제합니다.

- **CLI**에서 구성 정책을 삭제합니다.

  a. 다음 명령을 실행하여 구성 정책을 삭제합니다.

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

  정책이 삭제되면 대상 클러스터 또는 클러스터에서 제거됩니다.

b. 다음 명령을 실행하여 정책이 제거되었는지 확인합니다.

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

- 콘솔에서 구성 정책을 삭제합니다.

  탐색 메뉴에서 **Governance** 를 클릭하여 정책의 테이블 목록을 확인합니다.

  정책 위반 테이블에서 삭제할 정책의 **Actions** 아이콘을 클릭합니다. 그런 다음 제거를 클릭합니다. 정책 제거 대화 상자에서 정책 제거를 클릭합니다.

정책이 삭제됩니다.

CM-Configuration-Management 폴더의 **Red Hat Advanced Cluster Management**에서 지원하는 구성 정책 샘플을 참조하십시오.

또는 **Kubernetes** 구성 정책 컨트롤러를 참조하여 컨트롤러 에서 모니터링하는 다른 구성 정책을 볼 수 있습니다. 다른 정책을 관리하는 방법에 대한 자세한 내용은 보안 정책 관리를 참조하십시오.

2.6.8. 게이트 키퍼 운영자 정책 관리

게이트키퍼 운영자 정책을 사용하여 관리 클러스터에 게이트키퍼 운영자 및 게이트키퍼를 설치합니다. 다음 섹션에서 게이트 키퍼 운영자 정책을 생성, 보기 및 업데이트하는 방법을 알아봅니다.

필수 액세스: 클러스터 관리자

- 게이트 키퍼 운영자 정책을 사용하여 게이트 키퍼 설치
- 콘솔에서 게이트 키퍼 정책 생성
  - 게이트 키퍼 **Operator CR**

- 업그레이드 게이트키퍼 및 게이트키퍼 Operator
- 게이트 키퍼 운영자 정책 업데이트
  - 콘솔에서 게이트 키 운영자 정책 보기
  - 게이트키퍼 운영자 정책 비활성화
- 게이트 키퍼 Operator 정책 삭제
- 게이트 키퍼 정책, 게이트키퍼 및 게이트키퍼 운영자 정책 설치 제거

2.6.8.1. 게이트 키퍼 운영자 정책을 사용하여 게이트 키퍼 설치

거버넌스 프레임워크를 사용하여 게이트 키퍼 운영자를 설치합니다. 게이트키퍼 **Operator**는 **OpenShift Container Platform** 카탈로그에서 사용할 수 있습니다. 자세한 내용은 OpenShift Container Platform 설명서에서 클러스터에 **Operator** 추가 를 참조하십시오.

구성 정책 컨트롤러를 사용하여 게이트키퍼 운영자 정책을 설치합니다. 설치 중에 운영자 그룹과 구독은 게이트키퍼 운영자를 가져와 관리 클러스터에 설치합니다. 그런 다음 게이트 키퍼 운영자는 게이트 키퍼(**gatekeeper**)를 구성하기 위해 게이트키퍼 **CR**을 생성합니다. Gatekeeper operator CR 샘플을 확인합니다.

게이트키퍼 운영자 정책은 적용 작업을 지원하는 **Red Hat Advanced Cluster Management** 구성 정책 컨트롤러에서 모니터링합니다. 게이트키퍼 운영자 정책은 강제 적용이 설정된 경우 컨트롤러에서 자동으로 생성합니다.

2.6.8.2. 콘솔에서 게이트 키퍼 정책 생성

콘솔에서 게이트 키퍼 정책을 생성하여 게이트키퍼 정책을 설치합니다.

클러스터에 로그인한 후 **Governance** 페이지로 이동합니다.

정책 생성을 선택합니다. 양식을 작성할 때 사양 필드에서 GatekeeperOperator 를 선택합니다. 정책의 매개변수 값은 자동으로 채워지고 정책이 기본적으로 알림 으로 설정됩니다. 게이트키퍼를 설치하도록 적용하도록 수정 작업을 설정합니다. 샘플을 보려면 policy-gatekeeper-operator.yaml 을 참조하십시오.

+ 참고: Operator에서 기본값을 생성할 수 있습니다. 게이트키퍼 운영자 정책에 사용할 수 있는 선택적 매개변수에 대한 설명은 Gatekeeper Helm Chart 를 참조하십시오.

2.6.8.2.1. 게이트 키퍼 Operator CR

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

```
    operator: "Exists"
    effect: "NoSchedule"
  podAnnotations:
    some-annotation: "this is a test"
    other-annotation: "another test"
```

2.6.8.3. 업그레이드 게이트키퍼 및 게이트키퍼 Operator

게이트 키퍼 및 게이트키퍼 운영자의 버전을 업그레이드할 수 있습니다. 다음 단계를 완료합니다.

- 게이트 키퍼 운영자 정책을 사용하여 게이트 키퍼 운영자를 설치할 때 **installPlanApproval** 의 값을 확인합니다. **installPlanApproval** 이 자동으로 설정된 경우 **Operator** 업그레이드가 자동으로 수행됩니다. **installPlanApproval** 이 **Manual** 로 설정된 경우 각 클러스터에 대해 게이트 키퍼 **Operator**의 업그레이드를 수동으로 승인해야 합니다.

2.6.8.4. 게이트 키퍼 운영자 정책 업데이트

다음 섹션을 확인하여 게이트 키퍼 운영자 정책을 업데이트하는 방법을 알아봅니다.

2.6.8.4.1. 콘솔에서 게이트 키 운영자 정책 보기

게이트키퍼 운영자 정책 및 콘솔에서 해당 상태를 확인합니다.

콘솔에서 클러스터에 로그인한 후 **Governance** 를 클릭하여 정책의 테이블 목록을 확인합니다. 참고: 정책 탭 또는 클러스터 위반 탭을 선택하여 정책의 표 목록을 필터링할 수 있습니다.

자세한 내용을 보려면 **policy-gatekeeper-operator** 정책을 선택합니다. **Clusters** 탭을 선택하여 정책 위반을 확인합니다.

2.6.8.4.2. 게이트키퍼 운영자 정책 비활성화

게이트키퍼 운영자 정책을 비활성화합니다.

Kubernetes 콘솔의 **Red Hat Advanced Cluster Management**에 로그인한 후 **Governance** 페이지로 이동하여 정책의 표 목록을 확인합니다.

policy-gatekeeper-operator 정책의 **Actions** 아이콘을 선택한 다음 **Disable** 를 클릭합니다. 정책 비활성화 대화 상자가 나타납니다.

Disable policy 를 클릭합니다. **policy-gatekeeper-operator** 정책이 비활성화되어 있습니다.

2.6.8.5. 게이트 키퍼 Operator 정책 삭제

CLI 또는 콘솔에서 게이트키퍼 운영자 정책을 삭제합니다.

- **CLI**에서 게이트키퍼 운영자 정책을 삭제합니다.

  a. 다음 명령을 실행하여 게이트키퍼 운영자 정책을 삭제합니다.

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

  정책이 삭제되면 대상 클러스터 또는 클러스터에서 제거됩니다.

  b. 다음 명령을 실행하여 정책이 제거되었는지 확인합니다.

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

- 콘솔에서 게이트키퍼 운영자 정책 삭제:

  Governance 페이지로 이동하여 정책의 테이블 목록을 확인합니다.

  이전 콘솔 지침과 유사하게 **policy-gatekeeper-operator** 정책의 **Actions** 아이콘을 클릭합니다. 제거를 클릭하여 정책을 삭제합니다. 정책 제거 대화 상자에서 정책 제거를 클릭합니다.

게이트키퍼 운영자 정책이 삭제됩니다.

2.6.8.6. 게이트 키퍼 정책, 게이트키퍼 및 게이트키퍼 운영자 정책 설치 제거

게이트키퍼 정책, 게이트키퍼, 게이트키퍼 운영자 정책을 제거하려면 다음 단계를 완료합니다.

1.

관리형 클러스터에 적용되는 게이트키퍼 제약 조건 및 **Constraint Template** 을 제거합니다.

a. 게이트키퍼 운영자 정책을 편집합니다. 게이트키퍼 제약 조건 및 제약 조건을 생성하는 데 사용한 **ConfigurationPolicy** 템플릿을 찾습니다.

b. **ConfigurationPolicy** 템플릿의 **complianceType** 값을 **mustnothave** 로 변경합니다.

c. 정책을 저장하고 적용합니다.

2. 관리 클러스터에서 게이트 키퍼 인스턴스를 제거합니다.

a. 게이트키퍼 운영자 정책을 편집합니다. **Gatekeeper CR**(사용자 정의 리소스)을 생성하는 데 사용한 **ConfigurationPolicy** 템플릿을 찾습니다.

b. **ConfigurationPolicy** 템플릿의 **complianceType** 값을 **mustnothave** 로 변경합니다.

3. 관리되는 클러스터에 있는 **gatekeeper Operator**를 제거합니다.

a. 게이트키퍼 운영자 정책을 편집합니다. **Subscription CR**을 생성하는 데 사용한 **ConfigurationPolicy** 템플릿을 찾습니다.

b. **ConfigurationPolicy** 템플릿의 **complianceType** 값을 **mustnothave** 로 변경합니다.

게이트키퍼 정책, 게이트키퍼, 게이트키퍼 운영자 정책이 제거됩니다.

게이트키퍼에 대한 자세한 내용은 *게이트 키퍼 제약 조건 및 제약 조건 템플릿을* 통합합니다. 타사 정책을 제품과 통합하는 주제 목록은 *타사 정책 컨트롤러* 통합을 참조하십시오.

2.7. HUB 클러스터 보안

허브 클러스터 보안을 강화하여 **Kubernetes** 설치용 **Red Hat Advanced Cluster Management**를 보호합니다. 다음 단계를 완료합니다.

1. 보안 **Red Hat OpenShift Container Platform.** 자세한 내용은 **OpenShift Container Platform** 보안 및 컴플라이언스 를 참조하십시오.

2. **RBAC(**역할 기반 액세스 제어**) 설정.** 자세한 내용은 역할 기반 액세스 제어를 참조하십시오.

3. 인증서 사용자 지정, **certificate**을 참조하십시오. **???**

4. 클러스터 인증 정보 정의, 인증 정보 관리 개요를 참조하십시오.

5. 클러스터 보안을 강화하기 위해 사용 가능한 정책을 검토합니다. 지원되는 정책참조

2.8. 무결성 자동 보호 (기술 프리뷰)

무결성 방패는 리소스 생성 또는 업데이트에 대한 서명 확인을 시행하기 위한 무결성 제어를 지원하는 도구입니다. 무결성 중단은**OPA(Open Policy Agent)** 및 게이트키퍼를 지원하고, 요청에 서명이 있는지 확인하고, 정의된 제약 조건에 따라 인증되지 않은 요청을 차단합니다.

다음 무결성 방패 기능을 참조하십시오.

- 인증된 **Kubernetes** 매니페스트 배포만 지원합니다.
- 리소스가 허용 목록에 추가되지 않는 한 리소스 구성에서 **0drift**를 지원합니다.
- 승인 컨트롤러 강제와 같은 클러스터에서 모든 무결성 확인을 수행합니다.
- 인증되지 않은 **Kubernetes** 리소스가 클러스터에 배포되었는지 보고하기 위해 리소스를 지속적으로 모니터링합니다.
- **Kubernetes** 매니페스트 **YAML** 파일에 서명하기 위해 **X509, GPG** 및 **Sigstore** 서명이 지원됩니다. **Kubernetes** 무결성 분리는 **k8s-manifest-sigstore** 를 사용하여 **Sigstore** 서명을 지원합

니다.

2.8.1. 무결성 분리 아키텍처

무결성 방패는 **API**와 **Observer**의 두 가지 주요 구성 요소로 구성됩니다. 무결성 분리 **Operator**는 클러스터의 무결성 분리 구성 요소의 설치 및 관리를 지원합니다. 구성 요소에 대한 다음 설명을 확인합니다.

- 무결성 중단 **API** 는 **OPA** 또는 게이트키퍼에서 **Kubernetes** 리소스를 수신하고, 승인 요청에 포함된 리소스를 검증하고, 확인 결과를 **OPA** 또는 게이트 키퍼로 보냅니다. 무결성 방패 **API**는 **k8s-manifest-sigstore** 의 **verify-resource** 기능을 사용하여 **Kubernetes** 매니페스트 **YAML** 파일을 확인합니다. 무결성 분리 **API**는 **OPA** 또는 게이트 키퍼의 제약 조건 프레임 워크를 기반으로 하는 사용자 정의 리소스인 **ManifestingIntegrityConstraint** 에 따라 리소스를 검증합니다.

- 무결성 방위 **Observer** 는 **ManifestingIntegrityConstraint** 리소스에 따라 클러스터에서 **Kubernetes** 리소스를 지속적으로 확인하고 결과를 **ManifestIntegrityState** 라는 리소스로 내보냅니다. 무결성 배양 **Observer**는 또한 **k8s-manifest-sigstore** 를 사용하여 서명을 확인합니다.

2.8.2. 지원되는 버전

다음 제품 버전은 무결성 보호 기능을 지원합니다.

- **Red Hat OpenShift Container Platform 4.7.1 이상**
- **Kubernetes v1.19.7 이상**
- **gatekeeper-operator.v-.2.0**
- **gatekeeper v3.5**

자세한 내용은 무결성 방위 보호(기술 프리뷰) 를 참조하십시오.

2.8.3. 무결성 방위 보호 활성화 (기술 프리뷰)

Kubernetes 리소스의 무결성을 보호하도록 **Red Hat Advanced Cluster Management for**

Kubernetes 클러스터에서 무결성을 보호합니다.

2.8.3.1. 사전 요구 사항

Red Hat Advanced Cluster Management 관리형 클러스터에서 무결성 분리 보호를 사용하려면 다음과 같은 사전 요구 사항이 필요합니다.

- *oc 또는 kubectl 명령을 사용하기 위해 클러스터에 대한 클러스터 관리자 액세스 권한과 함께 하나 이상의 관리형 클러스터가 있는 Red Hat Advanced Cluster Management hub 클러스터를 설치합니다.*

- *무결성 분리를 설치합니다. 무결성 방패를 설치하기 전에 클러스터에 Open Policy Agent 또는 gatekeeper를 설치해야 합니다. 무결성 Operator를 설치하려면 다음 단계를 완료합니다.*

  a. *다음 명령을 실행하여 무결성 분리를 위해 네임스페이스에 무결성 분리 Operator를 설치합니다.*

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

  b. *다음 명령을 사용하여 무결성 사용자 정의 리소스를 설치합니다.*

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

  c. *무결성 분리를 위해서는 클러스터에서 보호해야 하는 리소스 서명과 서명에 대한 한 쌍의 키가 필요합니다. 서명 및 확인 키 쌍을 설정합니다.*

  - *다음 명령을 사용하여 새 GPG 키를 생성합니다.*

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

  - *다음 명령을 사용하여 새 GPG 공개 키를 파일에 내보냅니다.*

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

-

*yq 를 설치하여 **Red Hat Advanced Cluster Management** 정책에 서명하기 위한 스크립트를 실행합니다.*

- *무결성 보호 및 **Red Hat Advanced Cluster Management** 서명을 사용하려면 무결성 저장소에서 소스를 검색하고 커밋하는 작업이 포함됩니다. Git 을 설치해야 합니다.*

2.8.3.2. 무결성 보호 활성화

*다음 단계를 완료하여 **Red Hat Advanced Cluster Management** 관리 클러스터에서 무결성 분리를 활성화합니다.*

1. *무결성 분리를 위한 허브 클러스터에 네임스페이스를 생성합니다. 다음 명령을 실행합니다.*

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

2. ***Red Hat Advanced Cluster Management** 관리형 클러스터에 확인 키를 배포합니다. 서명 및 검증 키를 생성해야 합니다. 허브 클러스터에서 acm-verification-key-setup.sh 를 실행하여 확인 키를 설정합니다. 다음 명령을 실행합니다.*

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

확인 키를 제거하려면 다음 명령을 실행합니다.

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

3. ***hub** 클러스터에 **policy-integrity- weeld**라는 **Red Hat Advanced Cluster Management** 정책을 생성합니다.*

   a. *policy-collection리포지토리에서 **policy-integrity -weeld** 정책을 검색합니다. 리포지토리를 분기합니다.*

b. ***remediationAction** 매개변수 값을 업데이트하여 **Red Hat Advanced Cluster Management** 관리형 클러스터에 무결성 배전을 배포하도록 네임스페이스 를 구성합니다.*

c. ***signerConfig** 섹션을 업데이트하여 서명자 및 확인 키에 대한 이메일을 구성합니다.*

d. ***PlacementRule** 을 구성하여 무결성 분리를 배포해야 하는 **Red Hat Advanced Cluster Management** 관리 클러스터를 결정합니다.*

e. *다음 명령을 실행하여 **policy-integrity-weeld.yaml** 에 서명합니다.*

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

참고: 정책을 변경하고 다른 클러스터에 적용할 때마다 새 서명을 생성해야 합니다. 그렇지 않으면 변경 사항이 차단되어 적용되지 않습니다.

*자세한 내용은 **policy-integrity- weeld** 정책을 참조하십시오.*