Red Hat Advanced Cluster Management for Kubernetes 2.10

GitOps

GitOps

Last Updated: 2024-05-24

Red Hat Advanced Cluster Management for Kubernetes 2.10 GitOps

GitOps

法律上の通知

概要

統合された GitOps と Argo CD の使用方法の詳細は、こちらをお読みください。

目次

第1章 GITOPS の概要

Red Hat OpenShift Container Platform GitOps および Argo CD は、元のアプリケーションライフサイクル **Channel** および **Subscription** モデルと比較して高度な機能を備えた Red Hat Advanced Cluster Management for Kubernetes と統合されています。

Argo CD 開発と GitOps の統合が活発であり、Argo CD の機能拡張や更新に貢献する大規模なコミュニティーも活発です。OpenShift Container Platform GitOps Operator を利用すると、Argo CD 開発の最新の進歩を利用でき、GitOps Operator サブスクリプションからサポートを受けることができます。

Red Hat Advanced Cluster Management for Kubernetes と OpenShift Container Platform GitOps および Argo CD の統合の詳細は、以下のトピックを参照してください。

- GitOps コンソール
- マネージドクラスターの OpenShift GitOps オペレーターへの登録
- GitOps のアプリケーション配置許容範囲の設定
- プッシュアンドプルモデルを使用した Argo CD の導入
- GitOps Operator をインストールするためのポリシーの生成
- OpenShift Container Platform GitOps を使用したポリシー定義の管理 (Argo CD)

1.1. GITOPS コンソール

統合された OpenShift Container Platform GitOps コンソールの機能について詳しく説明します。**ApplicationSet** や **Argo CD** タイプなどのアプリケーションを作成および表示します。**ApplicationSet** は、このコントローラーから生成される Argo アプリケーションを表します。

- ArgoCD **ApplicationSet** を作成するには、**Sync policy** から **Automatically sync when cluster state changes** を有効にする必要があります。
- **kustomization** コントローラーを使用する Flux の場合は、**kustomize.toolkit.fluxcd.io/name=<app_name>** ラベルが付いた Kubernetes リソースを見つけます。
- **helm** コントローラーを使用する Flux の場合は、**helm.toolkit.fluxcd.io/name=<app_name>** ラベルが付いた Kubernetes リソースを見つけます。
- **ApplicationSet** を作成するには、GitOps クラスターリソースと GitOps Operator がインストールされている必要があります。これらの前提条件がないと、コンソールに **Argo サーバー** オプションが表示されず、**ApplicationSet** は作成されません。

重要: 利用可能なアクションは割り当てられたロールに基づきます。ロールベースのアクセス制御 のドキュメントで、アクセス要件について確認してください。

- **Launch resource in Search** をクリックし、関連リソースを検索します。
- **Search** を使用して、各リソースのコンポーネント **kind** 別にアプリケーションリソースを検索します。リソースの検索には、以下の値を使用します。

1.1.1. Argo CD アプリケーションのクエリー

Argo CD アプリケーションを検索すると、**Applications** ページに移動します。**Search** ページから Argo CD アプリケーションにアクセスするには、以下の手順を実行します。

1. Red Hat Advanced Cluster Management ハブクラスターにログインします。
2. コンソールヘッダーから **Search** アイコンを選択します。
3. **kind:application** および **apigroup:argoproj.io** の値でクエリーをフィルターします。
4. 表示するアプリケーションを選択します。**アプリケーション** ページでは、アプリケーションに関する情報の概要が表示されます。

検索の詳細は、コンソールでの検索の概要 を参照してください。

1.2. マネージドクラスターの OPENSHIFT GITOPS オペレーターへの登録

Push モデルで GitOps を設定するには、1 つ以上の Red Hat Advanced Cluster Management for Kubernetes マネージドクラスターのセットを Red Hat OpenShift Container Platform GitOps Operator のインスタンスに登録できます。登録後、アプリケーションをこれらのクラスターにデプロイできます。継続的な GitOps 環境を設定して、開発環境、ステージング、および実稼働環境のクラスター全体でアプリケーションの整合性を自動化します。

1.2.1. 前提条件

1. Red Hat Advanced Cluster Management for Kubernetes に Red Hat OpenShift GitOps Operator をインストールする必要がある。
2. 1 つ以上のマネージドクラスターをインポートする。

1.2.2. マネージドクラスターの GitOps への登録

マネージドクラスターを GitOps に登録するには、次の手順を実行します。

1. マネージドクラスターセットを作成し、マネージドクラスターをこれらのマネージドクラスターセットに追加します。**multicloud-integrations managedclusterset** のマネージドクラスターセットの例を参照してください。
   詳細は、ManagedClusterSet の作成 ドキュメントを参照してください。
2. Red Hat OpenShift GitOps がデプロイされている namespace にバインドするマネージドクラスターセットを作成します。マネージドクラスターを **openshift-gitops** namespace にバインドする例は、**multicloud-integrations** マネージドクラスターセットバインディングの例を参照してください。**ManagedClusterSetBinding** の作成に関する一般的な情報は、**追加リソース** セクションの **ManagedClusterSetBinding リソースの作成** を参照してください。配置の詳細は、ManagedCusterSets からの ManagedClusters のフィルタリング を参照してください。
3. マネージドクラスターセットバインディングで使用される namespace で、**Placement** カスタムリソースを作成し、OpenShift Container Platform GitOps Operator インスタンスに登録するマネージドクラスターのセットを選択します。**multicloud-integration** 配置例をテンプレートとして使用します。配置情報は、**配置での ManagedClusterSets の使用** を参照してください。
   注記:
   - 他の Kubernetes クラスターではなく、Red Hat OpenShift Container Platform GitOps Operator インスタンスに登録されるのは、OpenShift Container Platform クラスターのみです。
   - 一部の不安定なネットワークシナリオでは、マネージドクラスターが一時的に使用 **unavailable** または **unreachable** 状態になることがあります。詳細は、**Red Hat Advanced Cluster Management および OpenShift GitOps の配置許容範囲の設定** を参照してくださ

い。

4. **GitOpsCluster** カスタムリソースを作成し、配置決定から OpenShift GitOps の指定されたインスタンスにマネージドクラスターのセットを登録します。これにより、OpenShift GitOps インスタンスは、これらの Red Hat Advanced Cluster Management マネージドクラスターのいずれかにアプリケーションをデプロイできます。**multicloud-integrations** GitOps クラスターの例を使用します。
注記: 参照される **Placement** リソースは、**GitOpsCluster** リソースと同じ namespace に配置されている必要があります。以下の例を参照してください。

```
apiVersion: apps.open-cluster-management.io/v1beta1
kind: GitOpsCluster
metadata:
  name: gitops-cluster-sample
  namespace: dev
spec:
  argoServer:
    cluster: local-cluster
    argoNamespace: openshift-gitops
  placementRef:
    kind: Placement
    apiVersion: cluster.open-cluster-management.io/v1beta1
    name: all-openshift-clusters (1)
```

(1) **placementRef.name** の値は **all-openshift-clusters** で、**argoNamespace: openshift-gitops** にインストールされている GitOps インスタンスのターゲットクラスターとして指定されます。**argoServer.cluster** 仕様には **local-cluster** の値が必要です。

5. 変更を保存します。次に、GitOps ワークフローに従って、アプリケーションを管理できます。

1.2.3. GitOps トークン

配置および **ManagedClusterSetBinding** カスタムリソースを使用して GitOps namespace にバインドされているすべてのマネージドクラスターの GitOps Operator と統合すると、**ManagedCluster** にアクセスするためのトークンが含まれるシークレットがその namespace に作成されます。これは、GitOps コントローラーがリソースをマネージドクラスターと同期するために必要です。ユーザーに GitOps namespace への管理者アクセス権が付与され、アプリケーションのライフサイクル操作を実行すると、ユーザーはこのシークレットへのアクセス権と、マネージドクラスターへの **admin** レベルのアクセス権が付与されます。

これが望ましくない場合は、ユーザーをこの namespace の範囲の **admin** ロールにバインドする代わりに、ユーザーをバインドするために作成および使用できるアプリケーションリソースを操作するために必要なアクセス権限がある、より制限の厳しいカスタムロールを使用します。以下の **ClusterRole** の例を参照してください。

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: application-set-admin
rules:
- apiGroups:
  - argoproj.io
  resources:
  - applicationsets
```

```
  verbs:
  - get
  - list
  - watch
  - update
  - delete
  - deletecollection
  - patch
```

1.2.4. 関連情報

- 詳細は Red Hat Advanced Cluster Management および OpenShift GitOps の配置許容範囲の設定 を参照してください。
- **multicloud-integrations** マネージドクラスターセット の例を参照してください。
- **ManagedClusterSet の作成** を参照してください。
- **multicloud-integration** の配置 の例を参照してください。
- 配置情報は、Placement overview を参照してください。
- **multicloud-integrations** GitOps クラスターの 例を参照してください。
- **multicloud-integrations** マネージドクラスターセットバインディング の例を参照してください。
- 詳細は、ManagedClusterSetBinding リソースの作成 ドキュメントを参照してください。
- 詳細は、GitOps について を参照してください。

1.3. GITOPS のアプリケーション配置許容範囲の設定

Red Hat Advanced Cluster Management は、アプリケーションを Red Hat OpenShift GitOps にデプロイするマネージドクラスターを登録する方法を提供します。

一部の不安定なネットワークシナリオでは、マネージドクラスターが一時的に使用 **Unavailable** 状態になることがあります。アプリケーションのデプロイを容易にするために **Placement** リソースが使用されている場合は、使用できないクラスターを引き続き含めるために、**Placement** リソースに次の許容範囲を追加します。次の例は、許容範囲を含む **Placement** リソースを示しています。

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement
  namespace: ns1
spec:
  tolerations:
    - key: cluster.open-cluster-management.io/unreachable
      operator: Exists
    - key: cluster.open-cluster-management.io/unavailable
      operator: Exists
```

1.4. プッシュアンドプルモデルを使用した ARGO CD の導入

Push モデル 使用して、ハブクラスター上の Argo CD サーバーは、マネージドクラスターにアプリケーションリソースをデプロイします。**Pull モデル** の場合、アプリケーションリソースは **manifestWork** を使用して **Propagation controller** によってマネージドクラスターに伝播されます。

どちらのモデルでも、同じ **ApplicationSet** CRD を使用してアプリケーションをマネージドクラスターにデプロイします。

必要なアクセス権限: クラスターの管理者

- 前提条件
- アーキテクチャー
- ApplicationSet カスタムリソースの作成
- MulticlusterApplicationSetReport

1.4.1. 前提条件

Argo CD Pull モデルの次の前提条件を確認している。

重要:

- **openshift-gitops-ArgoCD-application-controller** サービスアカウントがクラスター管理者として割り当てられてい **ない** 場合、GitOps アプリケーションコントローラーはリソースをデプロイしない可能性があります。アプリケーションのステータスによって、次のようなエラーが送信される場合があります。

```
cannot create resource "services" in API group "" in the namespace
"mortgage",deployments.apps is forbidden: User
"system:serviceaccount:openshift-gitops:openshift-gitops-Argo CD-application-controller"
```

- 管理対象クラスターに **OpenShift Gitops** Operator をインストールした後、同じマネージドクラスターに **ClusterRoleBinding** クラスター管理者権限を作成する必要があります。
- 管理対象クラスターに **ClusterRoleBinding** クラスター管理者権限を追加するには、次の YAML の例を参照してください。

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

- クラスター管理者ではなく、この問題を解決する必要がある場合は、次の手順を実行してください。

  1. Argo CD アプリケーションがデプロイされる各マネージドクラスター上にすべての namespace を作成します。

2. **managed-by** ラベルを各 namespace に追加します。Argo CD アプリケーションが複数の namespace にデプロイされている場合、各 namespace は Argo CD によって管理される必要があります。
   managed-by ラベルを使用した次の例を参照してください。

```
apiVersion: v1
kind: Namespace
metadata:
  name: mortgage2
  labels:
    argocd.argoproj.io/managed-by: openshift-gitops
```

1. すべてのアプリケーション宛先 namespace をアプリケーションのリポジトリー内で宣言し、namespace に **managed** ラベルを含める必要があります。namespace を宣言する方法は、**Additional resources** を参照してください。

Argo CD **Pull** モデルを使用するには、次の要件を参照してください。

- GitOps Operator はハブクラスターに、ターゲットのマネージドクラスターを **openshift-gitops** namespace にインストールする必要があります。
- 必要なハブクラスター OpenShift Container Platform GitOps operator はバージョン 1.9.0 以降である必要があります。
- 必要なマネージドクラスター OpenShift Container Platform GitOps Operator はハブクラスターと同じバージョンである必要があります。
- マネージドクラスターの Argo CD アプリケーションテンプレートを伝播するには、**ApplicationSet コントローラー** が必要です。
- すべてのマネージドクラスターは、ハブクラスター上の Argo CD サーバー namespace にクラスターシークレットを持っている必要があります。これは、ArgoCD アプリケーションセットコントローラーがマネージドクラスターの Argo CD アプリケーションテンプレートを伝播するために必要です。
  クラスターシークレットを作成するには、**placement** リソースへの参照が含まれる **gitOpsCluster** リソースを作成します。**placement** リソースは、プルモデルをサポートする必要があるすべてのマネージドクラスターを選択します。GitOps クラスターコントローラーが調整すると、Argo CD サーバーの namespace にマネージドクラスターのクラスターシークレットが作成されます。

1.4.2. アーキテクチャー

Push および Pull モデルの両方で、ハブクラスター上の **Argo CD ApplicationSet コントローラー** が調整して、ターゲットのマネージドクラスターごとにアプリケーションリソースを作成します。両方のモデルのアーキテクチャーに関する以下の情報を参照してください。

1.4.2.1. アーキテクチャープッシュモデル

- Push モデルを使用すると、OpenShift Container Platform GitOps は、一元化されたハブクラスターからマネージドクラスターにリソースを **直接** 適用します。
- ハブクラスター上で実行されている Argo CD アプリケーションは、GitHub リポジトリーと通信し、マニフェストをマネージドクラスターに直接デプロイします。

- Push モデルの実装には、マネージドクラスターの認証情報を持つハブクラスター上の Argo CD アプリケーションのみが含まれます。ハブクラスター上の Argo CD アプリケーションは、アプリケーションをマネージドクラスターにデプロイできます。

- **重要:** リソースアプリケーションを必要とする多数のマネージドクラスターでは、OpenShift Container Platform GitOps コントローラーのメモリーと CPU 使用率に負荷がかかる可能性があることを検討してください。リソース管理を最適化するには、リソースクォータまたは要求の設定 を参照してください。

- デフォルトでは、**apps.open-cluster-management.io/ocm-managed-cluster** および **apps.open-cluster-management.io/pull-to-ocm-managed-cluster** を追加しない限り、アプリケーションのデプロイには Push モデルが使用されます。**ApplicationSet** のテンプレートセクションに cluster アノテーションを追加します。

1.4.2.2. アーキテクチャープルモデル

- プルモデルでは、ハブクラスター内のコントローラーのストレスが軽減されるため、プッシュモデルと比較してスケーラビリティーが軽減されますが、より多くのリクエストとステータスレポートが必要になります。

- プルモデルの場合、OpenShift Container Platform GitOps は、一元化されたハブクラスターからマネージドクラスターにリソースを直接適用 **しません**。Argo CD アプリケーションは、ハブクラスターからマネージドクラスターに伝播されます。

- Pull モデルの実装では、OpenShift Cluster Manager の登録、配置、**manifestWork** API が適用され、ハブクラスターがハブクラスターとマネージドクラスター間の安全な通信チャネルを使用してリソースをデプロイできるようになります。

- 各マネージドクラスターは個別に GitHub リポジトリーと通信してリソースマニフェストをローカルにデプロイするため、各マネージドクラスターに GitOps オペレーターをインストールして設定する必要があります。

- Argo CD サーバーは、各ターゲットのマネージドクラスター上で実行されている必要があります。Argo CD アプリケーションリソースはマネージドクラスター上に複製され、ローカルの Argo CD サーバーによってデプロイされます。マネージドクラスター上の分散 Argo CD アプリケーションは、ハブクラスター上の単一の Argo CD **ApplicationSet** リソースを使用して作成されます。

- マネージドクラスターは、**ocm-managed-cluster** アノテーションの値によって決定されます。

- Pull モデルを正常に実装するには、Argo CD アプリケーションコントローラーは、**ApplicationSet** のテンプレートセクションにある **argocd.argoproj.io/skip-reconcile** アノテーションを持つプッシュモデルアプリケーションリソースを **無視** する必要があります。

- Pull モデルの場合、マネージドクラスター上の **Argo CD アプリケーションコントローラー** が調整してアプリケーションをデプロイします。

- ハブクラスター上の Pull モデル **Resource sync controller** は、 各マネージドクラスター上の OpenShift Cluster Manager 検索 V2 コンポーネントに定期的にクエリーを実行し、各 Argo CD アプリケーションのリソースリストとエラーメッセージを取得します。

- ハブクラスターの **Aggregation controller** はリソース同期コントローラーからのデータと、**manifestWork** からのステータス情報を使用して、クラスター全体から **MulticlusterApplicationSetReport** を作成および更新します。

- デプロイメントのステータスはハブクラスターに収集されますが、すべての詳細情報が送信されるわけではありません。概要を提供するために、追加のステータス更新が定期的に収集され

ます。ステータスのフィードバックはリアルタイムではなく、各マネージドクラスターの GitOps operator は Git リポジトリーと通信する必要があるため、複数のリクエストが発生します。

1.4.3. ApplicationSet カスタムリソースの作成

Argo CD **ApplicationSet** リソース は、マネージドクラスターのリストを取得するために使用されるジェネレーターフィールド内の **placement** リソースを含むプッシュまたはプルモデルを使用して、マネージドクラスターにアプリケーションをデプロイするために使用されます。

1. プルモデルの場合、次の例に示すように、アプリケーションの宛先をデフォルトのローカル Kubernetes サーバーに設定します。アプリケーションは、マネージドクラスター上のアプリケーションコントローラーによってローカルにデプロイされます。

2. 以下の **ApplicationSet** YAML の例に示されるように、デフォルトの Push モデルを上書きするために必要なアノテーションを追加します。これは、テンプレートアノテーションで Pull モデルを使用します。

```
apiVersion: argoproj.io/v1alpha1
kind: `ApplicationSet`
metadata:
  name: guestbook-allclusters-app-set
  namespace: openshift-gitops
spec:
  generators:
  - clusterDecisionResource:
      configMapRef: ocm-placement-generator
      labelSelector:
        matchLabels:
          cluster.open-cluster-management.io/placement: aws-app-placement
      requeueAfterSeconds: 30
  template:
    metadata:
      annotations:
        apps.open-cluster-management.io/ocm-managed-cluster: '{{name}}' 1
        apps.open-cluster-management.io/ocm-managed-cluster-app-namespace: openshift-
gitops
        argocd.argoproj.io/skip-reconcile: "true" 2
      labels:
        apps.open-cluster-management.io/pull-to-ocm-managed-cluster: "true" 3
      name: '{{name}}-guestbook-app'
    spec:
      destination:
        namespace: guestbook
        server: https://kubernetes.default.svc
      project: default
      sources: [
      {
        repoURL: https://github.com/argoproj/argocd-example-apps.git
        targetRevision: main
        path: guestbook
         }
      ]
      syncPolicy:
```

```
    automated: {}
    syncOptions:
    - CreateNamespace=true
```

1. Pull モデルには **apps.open-cluster-management.io/ocm-managed-cluster** が必要です。
2. Push モデルリソースを無視するには、**argocd.argoproj.io/skip-reconcile** が必要です。
3. **apps.open-cluster-management.io/pull-to-ocm-managed-cluster: "true"** も Pull モデルに必要です。

1.4.4. MulticlusterApplicationSetReport

- Pull モデルの場合、**MulticlusterApplicationSetReport** はマネージドクラスター全体からアプリケーションステータスを集約します。
- レポートには、リソースのリストと、各マネージドクラスターからのアプリケーションの全体的なステータスが含まれます。
- Argo CD ApplicationSet リソースごとに個別のレポートリソースが作成されます。レポートは **ApplicationSet** と同じ namespace に作成されます。
- レポートには次の項目が含まれます。
  1. Argo CD アプリケーションのリソースのリスト
  2. 各 Argo CD アプリケーションの全体的な同期およびヘルスステータス
  3. 全体的なステータスが **out of sync**、または **unhealthy** 各クラスターのエラーメッセージ
  4. 管理対象クラスタのすべての状態を示すサマリーステータス
- **Resource sync controller** と **Aggregation controller** は両方とも 10 秒ごとに実行され、レポートを作成します。
- 次の出力例に示すように、2 つのコントローラーと Propagation コントローラーは、同じ **multicluster-integrations** Pod 内の別々のコンテナーで実行されます。

```
NAMESPACE                 NAME                                        READY   STATUS
open-cluster-management multicluster-integrations-7c46498d9-fqbq4  3/3     Running
```

以下は、**guestbook** アプリケーションの **MulticlusterApplicationSetReport** YAML ファイルの例です。

```
apiVersion: apps.open-cluster-management.io/v1alpha1
kind: MulticlusterApplicationSetReport
metadata:
  labels:
    apps.open-cluster-management.io/hosting-applicationset: openshift-gitops.guestbook-allclusters-
app-set
  name: guestbook-allclusters-app-set
  namespace: openshift-gitops
statuses:
  clusterConditions:
```

```
  - cluster: cluster1
    conditions:
    - message: 'Failed sync attempt: one or more objects failed to apply, reason: services is forbidden:
User "system:serviceaccount:openshift-gitops:openshift-gitops-Argo CD-application-controller" cannot
create resource "services" in API group "" in the namespace "guestbook",deployments.apps is
forbidden: User <name> cannot create resource "deployments" in API group "apps" in the
namespace "guestboo...'
      type: SyncError
    healthStatus: Missing
    syncStatus: OutOfSync
  - cluster: pcluster1
    healthStatus: Progressing
    syncStatus: Synced
  - cluster: pcluster2
    healthStatus: Progressing
    syncStatus: Synced
  summary:
    clusters: "3"
    healthy: "0"
    inProgress: "2"
    notHealthy: "3"
    notSynced: "1"
    synced: "2"
```

Note: リソースがデプロイに失敗した場合、リソースはリソース一覧に含まれません。詳細は、エラーメッセージを参照してください。

1.4.5. 関連情報

- OpenShift Container Platform ドキュメントの クラスター設定でアプリケーションをデプロイすることによる OpenShift クラスターの設定 を参照してください。
- OpenShift Container Platform ドキュメントの Argo CD インスタンスのセットアップ を参照してください。

1.5. OPENSHIFT CONTAINER PLATFORM GITOPS を使用したポリシー定義の管理 (ARGO CD)

非推奨: PlacementRule

Argo CD に基づく OpenShift Container Platform GitOps を使用して、ポリシー定義を管理できます。このワークフローを可能にするには、Red Hat Advanced Cluster Management ハブクラスターでポリシーを作成するための OpenShift Container Platform GitOps アクセス権を付与する必要があります。以下の手順を実行して、ポリシーと配置を作成、読み取り、更新、削除するためのアクセス権を持つ OpenShift Container Platform GitOps の **ClusterRole** リソースを作成してください。

1. コンソールから **ClusterRole** を作成します。**ClusterRole** は次の例のようになります。

```
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
      - policysets
      - 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
```

2. **ClusterRoleBinding** オブジェクトを作成して、OpenShift Container Platform GitOps サービスアカウントに **openshift-gitops-policy-admin ClusterRole** オブジェクトへのアクセスを許可します。**ClusterRoleBinding** は次の例のようになります。

```
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
```

–

Red Hat Advanced Cluster Management ポリシー定義が OpenShift Container Platform GitOps とともにデプロイされると、ポリシーのコピーが各マネージドクラスター namespace に作成されます。これらのコピーは、複製されたポリシーと呼ばれます。OpenShift Container Platform GitOps がこの複製されたポリシーを繰り返し削除したり、ArgoCD **Application** が同期していないことを示したりするのを防ぐために、**argocd.argoproj.io/compare-options: IgnoreExtraneous** アノテーションは、Red Hat Advanced Cluster Management ポリシーフレームワークによって、それぞれのレプリケーションされたポリシーに自動的に設定されます。

Argo CD がオブジェクトを追跡するために使用するラベルとアノテーションがあります。複製されたポリシーが Argo CD にまったく表示されないようにするには、Red Hat Advanced Cluster Management ポリシー定義で **spec.copyPolicyMetadata** を **false** に設定して、Argo CD の追跡ラベルとアノテーションが複製されたポリシーにコピーされないようにすることができます。

1.5.1. ポリシージェネレーターと OpenShift Container Platform GitOps (Argo CD) の統合

Argo CD に基づく OpenShift Container Platform GitOps を使用すると、GitOps を通じて Policy Generator を使用してポリシーを生成できます。Policy Generator は OpenShift Container Platform GitOps コンテナーイメージにプリインストールされていないため、いくつかのカスタマイズを行う必要があります。続行するには、OpenShift Container Platform GitOps Operator を Red Hat Advanced Cluster Management ハブクラスターにインストールし、ハブクラスターにログインする必要があります。

KusTOMize の実行時に OpenShift Container Platform GitOps が Policy Generator にアクセスできるようにするには、Red Hat Advanced Cluster Management アプリケーションのサブスクリプションコンテナーイメージから OpenShift Container Platform GitOps コンテナーに Policy Generator バイナリーをコピーするための Init コンテナーが必要です。さらに、OpenShift Container Platform GitOps は、KusTOMize の実行時に **--enable-alpha-plugins** フラグを提供するように設定する必要があります。以下の手順を実行します。

1. 次のコマンドを使用して、OpenShift Container Platform GitOps **argocd** オブジェクトの編集を開始します。

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

2. OpenShift Container Platform GitOps の **argocd** オブジェクトを変更して、次の追加の YAML コンテンツを含めます。Red Hat Advanced Cluster Management の新しいメジャーバージョンがリリースされ、ポリシージェネレーターを新しいバージョンに更新したい場合は、Init コンテナーで使用される **registry.redhat.io/rhacm2/multicluster-operators-subscription-rhel8** イメージをより新しいタグに更新する必要があります。以下の例を見て、**<version>** を 2.10 または目的の Red Hat Advanced Cluster Management バージョンに置き換えます。

```
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 /policy-generator/PolicyGenerator-not-fips-compliant /policy-generator-
tmp/PolicyGenerator
      command:
      - /bin/bash
      image: registry.redhat.io/rhacm2/multicluster-operators-subscription-rhel9:v<version>
      name: policy-generator-install
      volumeMounts:
      - mountPath: /policy-generator
        name: policy-generator-tmp
    volumeMounts:
    - mountPath: /etc/kustomize/plugin/policy.open-cluster-management.io/v1/policygenerator
      name: policy-generator
    volumes:
    - emptyDir: {}
      name: policy-generator
```

注記: または、**ArgoCD** マニフェストを含む **ConfigurationPolicy** リソースを作成し、**MultiClusterHub** で設定されたバージョンと一致するバージョンをテンプレート化することもできます。

```
image: '{{ (index (lookup "apps/v1" "Deployment" "open-cluster-management" "multicluster-
operators-hub-subscription").spec.template.spec.containers 0).image }}'
```

ポリシーを生成する前に Kustomize ディレクトリー内で Helm チャートの処理を有効にする場合は、**spec.repo.env** フィールドの環境変数 **POLICY_GEN_ENABLE_HELM** を **true** に設定します。

```
env:
- name: POLICY_GEN_ENABLE_HELM
  value: "true"
```

3. OpenShift Container Platform GitOps がポリシージェネレーターを使用できるようになったので、Red Hat Advanced Cluster Management ハブクラスターでポリシーを作成するためのアクセス権を OpenShift Container Platform GitOps に付与する必要があります。ポリシーとプレースメントを作成、読み取り、更新、および削除するためのアクセス権を持つ、**openshift-gitops-policy-admin** という名前の **ClusterRole** リソースを作成します。前述の **ClusterRole** リソースの例を参照してください。

4. **ClusterRoleBinding** オブジェクトを作成して、OpenShift Container Platform GitOps サービスアカウントに **openshift-gitops-policy-admin ClusterRole** へのアクセス権を付与します。**ClusterRoleBinding** は、次のようなリソースになる場合があります。

```
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
```

1.5.2. 関連情報

- Argo CD のドキュメントを参照してください。

1.6. GITOPS OPERATOR をインストールするためのポリシーの生成

Red Hat Advanced Cluster Management ポリシーの一般的な用途は、Operator を 1 つ以上の Red Hat OpenShift Container Platform マネージドクラスターにインストールすることです。ポリシージェネレーターを使用してポリシーを生成する方法、および生成されたポリシーを使用して OpenShift Container Platform GitOps Operator をインストールする方法は、引き続きお読みください。

1.6.1. OpenShift Container Platform GitOps をインストールするポリシーの生成

Policy Generator を使用して、OpenShift Container Platform GitOps をインストールするポリシーを生成できます。OpenShift Container Platform GitOps Operator は **all namespaces** インストールモードを提供しており、次の例で確認できます。次の例のように、**openshift-gitops-subscription.yaml** という **Subscription** マニフェストファイルを作成します。

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

Operator の特定のバージョンに固定するには、パラメーターと値 **spec.startingCSV: openshift-gitops-operator.v<version>** を追加します。**<version>** を希望のバージョンに置き換えます。

PolicyGenerator 設定ファイルが必要です。**policy-generator-config.yaml** という名前の設定ファイルを使用してポリシーを生成し、すべての OpenShift Container Platform マネージドクラスターに OpenShift GitOps をインストールします。以下の例を参照してください。

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

最後に必要なファイルは **kustomization.yaml** で、次の設定が必要です。

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

生成されたポリシーは、**PlacementRule** (非推奨) を含む次のファイルのようになります。

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

OpenShift Container Platform ドキュメントのマニフェストから生成されたポリシーがサポートされています。ポリシージェネレーターを使用して、OpenShift Container Platform ドキュメントの設定ガイダンスを適用できます。

1.6.2. OperatorGroups でのポリシー依存関係の使用

OperatorGroup マニフェストを使用して Operator をインストールする場合、**Subscription** が作成される前に、クラスターに **OperatorGroup** が存在している必要があります。ポリシージェネレーターとともにポリシー依存関係機能を使用して、**Subscription** ポリシーを実施する前に **OperatorGroup** ポリシーが準拠していることを確認します。

必要な順序でマニフェストを一覧表示して、ポリシーの依存関係を設定します。たとえば、namespace ポリシーを最初に作成し、次に **OperatorGroup** を作成し、最後に **Subscription** を作成することができます。

ポリシージェネレーター設定マニフェストで **policyDefaults.orderManifests** パラメーターを有効にし、**policyDefaults.consolidateManifests** を無効にして、マニフェスト間の依存関係を自動的に設定します。

1.6.3. 関連情報

- Compliance Operator をインストールするポリシーの生成 を参照してください。
- 詳細は、GitOps を使用したポリシーのデプロイ を参照してください。
- 詳細は、OpenShift GitOps の理解 および Operator ドキュメントを参照してください。
- クラスターへの Operator の追加 - CLI を使用した OperatorHub からのインストール を参照してください。
- 詳細は、Compliance Operator のドキュメント を参照してください。
- **すべての namespaces インストールモード** を参照してください。
- **namespaced インストールモード** を参照してください。
- Pod のデプロイ前の、Init コンテナーの使用によるタスクの実行 を参照してください。
- Argo CD を参照してください。

- OpenShift Container Platform がサポートする YAML 入力の次の例を表示します。
  - インストール後のクラスタータスク
  - 監査ログポリシーの設定
  - ログのサードパーティーシステムへの転送

1.7. ARGO CD プッシュモデル用のカスタマイズされたサービスアカウントの作成

ハブクラスター上に **managedserviceaccount** リソースを作成することにより、マネージドクラスター上にサービスアカウントを作成します。**clusterpermission** リソースを使用して、特定のパーミッションをサービスアカウントに付与します。

Argo CD プッシュモデルで使用するカスタマイズされたサービスアカウントを作成すると、次の利点があります。

- アプリケーションマネージャーアドオンは、各マネージドクラスター上で実行されます。デフォルトでは、Argo CD コントローラーはサービスアカウントアプリケーションマネージャーを使用して、これらのリソースをマネージドクラスターにプッシュします。
- アプリケーションサブスクリプションアドオンはアプリケーションマネージャーサービスを使用してマネージドクラスターにアプリケーションをデプロイするため、アプリケーションマネージャーサービスアカウントには大規模なアクセスパーミッションのセットがあります。制限された権限セットが必要な場合は、アプリケーションマネージャーサービスアカウントを使用しないでください。
- Argo CD プッシュモデルで使用する別のサービスアカウントを指定できます。Argo CD コントローラーが集中ハブクラスターからマネージドクラスターにリソースをプッシュする場合、デフォルトのアプリケーションマネージャーとは異なるサービスアカウントを使用できます。別のサービスアカウントを使用すると、このサービスアカウントに付与されるアクセス許可を制御できます。
- サービスアカウントはマネージドクラスター上に存在している必要があります。関連するアクセスパーミッションが割り当てられたサービスアカウントの作成を容易にするには、一元化されたハブクラスターで **managedserviceaccount** リソースと新しい **clusterpermission** リソースを使用します。

次の手順をすべて完了すると、マネージドサービスアカウントにクラスターのアクセスパーミッションを付与できます。クラスター権限があると、マネージドサービスアカウントには、マネージドクラスターにアプリケーションリソースをデプロイするために必要な権限が与えられます。以下の手順を実行します。

1. 「マネージドサービスアカウントの作成」
2. 「クラスターパーミッションの作成」
3. 「GitOpsCluster リソースでの管理サービスアカウントの使用」
4. 「Argo CD アプリケーションの作成」
5. 「ポリシーを使用したマネージドサービスアカウントおよびクラスターパーミッションの作成」

1.7.1. マネージドサービスアカウントの作成

ハブ上の **マネージドサービスアカウント** カスタムリソースを使用すると、マネージドクラスター上に **serviceaccounts** を作成する場合に便利です。**managedserviceccount** カスタムリソースがハブクラスターの **<managed_cluster>** namespace に作成されると、**serviceccount** がマネージドクラスターに作成されます。

マネージドサービスアカウントを作成するには、managedserviceaccount アドオンの有効化 を参照してください。

1.7.2. クラスターパーミッションの作成

サービスアカウントの作成時には、パーミッションはそのアカウントに関連付けられません。新規サービスアカウントにパーミッションを付与するには、**clusterpermission** リソースを使用します。**clusterpermission** リソースは、ハブのマネージドクラスター namespace に作成されます。このリソースを使用すると、ロール、マネージドクラスター上のクラスターロールリソースを作成し、**rolebinding** または **ClusterRoleBinding** リソースを使用して、サービスアカウントにバインドする場合に便利です。

1. **<managed-sa-sample>** サービスアカウントのアクセス許可を、**<managed cluster>** 上の mortgage namespace にデプロイされるサンプルの住宅ローンアプリケーションに付与するには、以下の内容を含む YAML を作成します。

```
apiVersion: rbac.open-cluster-management.io/v1alpha1
kind: ClusterPermission
metadata:
  name: <clusterpermission-msa-subject-sample>
  namespace: <managed cluster>
spec:
  roles:
  - namespace: default
    rules:
    - apiGroups: ["apps"]
      resources: ["deployments"]
      verbs: ["get", "list", "create", "update", "delete", "patch"]
    - apiGroups: [""]
      resources: ["configmaps", "secrets", "pods", "podtemplates", "persistentvolumeclaims",
"persistentvolumes"]
      verbs: ["get", "update", "list", "create", "delete", "patch"]
    - apiGroups: ["storage.k8s.io"]
      resources: ["*"]
      verbs: ["list"]
  - namespace: mortgage
    rules:
    - apiGroups: ["apps"]
      resources: ["deployments"]
      verbs: ["get", "list", "create", "update", "delete", "patch"]
    - apiGroups: [""]
      resources: ["configmaps", "secrets", "pods", "services", "namespace"]
      verbs: ["get", "update", "list", "create", "delete", "patch"]
  clusterRole:
    rules:
    - apiGroups: ["*"]
      resources: ["*"]
      verbs: ["get", "list"]
  roleBindings:
  - namespace: default
    roleRef:
```

```
    kind: Role
  subject:
    apiGroup: authentication.open-cluster-management.io
    kind: ManagedServiceAccount
    name: <managed-sa-sample>
- namespace: mortgage
  roleRef:
    kind: Role
  subject:
    apiGroup: authentication.open-cluster-management.io
    kind: ManagedServiceAccount
    name: <managed-sa-sample>
clusterRoleBinding:
  subject:
    apiGroup: authentication.open-cluster-management.io
    kind: ManagedServiceAccount
    name: <managed-sa-sample>
```

2. YAML ファイルを **cluster-permission.yaml** という名前のファイルとして保存します。

3. **oc apply -f cluster-permission.yaml** を実行します。

4. サンプル **<clusterpermission>** は、 mortgage namespace に **<clusterpermission-msa-subject-sample>** というロールを作成します。**mortgage** の namespace が存在しない場合は、この namespace を作成します。

5. **<managed cluster>** で作成されたリソースを確認します。

サンプル **<clusterpermission>** を作成すると、以下のリソースがサンプルのマネージドクラスターに作成されます。

- デフォルトの namespace 内の **<clusterpermission-msa-subject-sample>** という名前のロール が 1 つ。
- ロールをマネージドサービスアカウントにバインドするための default namespace で **<clusterpermission-msa-subject-sample>** という roleBinding が 1 つ。
- mortgage namespace に **<clusterpermission-msa-subject-sample>** といいうロールが 1 つ。
- ロールをマネージドサービスアカウントにバインドするための、mortgage namespace 内の **<clusterpermission-msa-subject-sample>** と呼ばれる roleBinding が 1 つ。
- **<clusterpermission-msa-subject-sample>** という clusterRole が 1 つ。
- clusterRole をマネージドサービスアカウントにバインドするための **<clusterpermission-msa-subject-sample>** という clusterRoleBinding が 1 つ。

1.7.3. GitOpsCluster リソースでの管理サービスアカウントの使用

GitOpsCluster リソースは、配置を使用して、選択したマネージドクラスターを Argo CD にインポートします。これには、クラスターへのアクセスに使用される情報を含む Argo CD クラスターシークレットの作成が含まれます。デフォルトでは、Argo CD クラスターシークレットは、アプリケーションマネージャーサービスアカウントを使用してマネージドクラスターにアクセスします。

1. マネージドサービスアカウントを使用するように GitOpsCluster リソースを更新するには、マネージドサービスアカウントの名前を指定して **manageServiceAccountRef** プロパティーを追加します。

2. GitOpsCluster カスタムリソースを作成するには、以下の YAML を Gitops.YAML として保存します。

```
---
apiVersion: apps.open-cluster-management.io/v1beta1
metadata:
  name: argo-acm-importer
  namespace: openshift-gitops
spec:
  managedServiceAccountRef: <managed-sa-sample>
  argoServer:
    cluster: notused
    argoNamespace: openshift-gitops
  placementRef:
    kind: Placement
    apiVersion: cluster.open-cluster-management.io/v1beta1
    name: all-openshift-clusters
    namespace: openshift-gitops
```

3. YAML ファイルを **gitops.yaml** という名前のファイルとして保存します。

4. **oc apply -f gitops.yaml** を実行します。

5. **openshift-gitops** namespace に移動し、**<managed cluster-managed-sa-sample-cluster-secret>** という名前の新しい Argo CD クラスターシークレットがあることを確認します。

```
% oc get secrets -n openshift-gitops <managed cluster-managed-sa-sample-cluster-secret>
NAME                                    TYPE     DATA   AGE
<managed cluster-managed-sa-sample-cluster-secret>   Opaque   3      4m2s
```

1.7.4. Argo CD アプリケーションの作成

プッシュモデルを使用して、Argo CD コンソールから Argo CD アプリケーションをデプロイします。Argo CD アプリケーションは、マネージドサービスアカウント **<managed-sa-sample>** でデプロイされます。

1. Argo CD コンソールにログインします。

2. **Create a new application** をクリックします。

3. クラスター URL を選択します。

4. Argo CD アプリケーションに移動し、**<managed cluster>** に伝播したロールやクラスターロールなど、指定のパーミッションがあることを確認します。

1.7.5. ポリシーを使用したマネージドサービスアカウントおよびクラスターパーミッションの作成

> When the GitOpsCluster resource is updated with the `managedServiceAccountRef`, each managed cluster in the placement of this GitOpsCluster needs to have the service account. If you have several managed clusters, it becomes tedious for you to create the managed service account and cluster

> permission for each managed cluster. You can simply this process by using a policy to create the managed service account and cluster permission for all your managed clusters

selectedServiceAccount リソースと **clusterPermission** リソースをハブクラスターに適用すると、このポリシーの配置はローカルクラスターにバインドされます。これらのリソースを、GitOpsCluster リソースの配置内のすべてのマネージドクラスターのマネージドクラスター namespace に複製します。

ポリシーを使用して **managedServiceAccount** および **clusterPermission** リソースを作成すると、次の属性が含まれます。

- ポリシー内の **managedServiceAccount** と **clusterPermission** オブジェクトテンプレートを更新すると、すべてのマネージドクラスターの **managedServiceAccount** および **clusterPermission** リソースがすべて更新されます。
- **managedServiceAccount** および **clusterPermission** リソースを直接更新すると、ポリシーが適用されるため、元の状態に戻されます。
- GitOpsCluster の配置に関する決定が変更された場合、ポリシーはマネージドクラスターの namespace 内のリソースの作成と削除を管理します。

1. マネージドサービスアカウントとクラスター権限を作成するための YAML のポリシーを作成するには、次の内容を含む YAML を作成します。

```
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  name: policy-gitops
  namespace: openshift-gitops
  annotations:
    policy.open-cluster-management.io/standards: NIST-CSF
    policy.open-cluster-management.io/categories: PR.PT Protective Technology
    policy.open-cluster-management.io/controls: PR.PT-3 Least Functionality
spec:
  remediationAction: enforce
  disabled: false
  policy-templates:

    - objectDefinition:
        apiVersion: policy.open-cluster-management.io/v1
        kind: ConfigurationPolicy
        metadata:
          name: policy-gitops-sub
        spec:
          pruneObjectBehavior: None
          remediationAction: enforce
          severity: low
          object-templates-raw: |
            {{ range $placedec := (lookup "cluster.open-cluster-management.io/v1beta1"
"PlacementDecision" "openshift-gitops" "" "cluster.open-cluster-
management.io/placement=aws-app-placement").items }}
            {{ range $clustdec := $placedec.status.decisions }}
            - complianceType: musthave
              objectDefinition:
                apiVersion: authentication.open-cluster-management.io/v1alpha1
                kind: ManagedServiceAccount
                metadata:
```

```
        name: <managed-sa-sample>
        namespace: {{ $clustdec.clusterName }}
      spec:
        rotation: {}
    - complianceType: musthave
      objectDefinition:
        apiVersion: rbac.open-cluster-management.io/v1alpha1
        kind: ClusterPermission
        metadata:
          name: <clusterpermission-msa-subject-sample>
          namespace: {{ $clustdec.clusterName }}
        spec:
          roles:
          - namespace: default
            rules:
            - apiGroups: ["apps"]
              resources: ["deployments"]
              verbs: ["get", "list", "create", "update", "delete"]
            - apiGroups: [""]
              resources: ["configmaps", "secrets", "pods", "podtemplates",
"persistentvolumeclaims", "persistentvolumes"]
              verbs: ["get", "update", "list", "create", "delete"]
            - apiGroups: ["storage.k8s.io"]
              resources: ["*"]
              verbs: ["list"]
          - namespace: mortgage
            rules:
            - apiGroups: ["apps"]
              resources: ["deployments"]
              verbs: ["get", "list", "create", "update", "delete"]
            - apiGroups: [""]
              resources: ["configmaps", "secrets", "pods", "services", "namespace"]
              verbs: ["get", "update", "list", "create", "delete"]
          clusterRole:
            rules:
            - apiGroups: ["*"]
              resources: ["*"]
              verbs: ["get", "list"]
          roleBindings:
          - namespace: default
            roleRef:
              kind: Role
            subject:
              apiGroup: authentication.open-cluster-management.io
              kind: ManagedServiceAccount
              name: <managed-sa-sample>
          - namespace: mortgage
            roleRef:
              kind: Role
            subject:
              apiGroup: authentication.open-cluster-management.io
              kind: ManagedServiceAccount
              name: <managed-sa-sample>
          clusterRoleBinding:
            subject:
              apiGroup: authentication.open-cluster-management.io
```

```
                kind: ManagedServiceAccount
                name: <managed-sa-sample>
        {{ end }}
        {{ end }}
---
apiVersion: policy.open-cluster-management.io/v1
kind: PlacementBinding
metadata:
  name: binding-policy-gitops
  namespace: openshift-gitops
placementRef:
  name: lc-app-placement
  kind: Placement
  apiGroup: cluster.open-cluster-management.io
subjects:
  - name: policy-gitops
    kind: Policy
    apiGroup: policy.open-cluster-management.io
---
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: lc-app-placement
  namespace: openshift-gitops
spec:
  numberOfClusters: 1
  predicates:
  - requiredClusterSelector:
      labelSelector:
        matchLabels:
          name: local-cluster
```

1. YAML ファイルを **policy.yaml** というファイルとして保存します。
2. **oc apply -f policy.yaml** を実行します。
3. ポリシーのオブジェクトテンプレートでは、GitOpsCluster に関連付けられた配置の決定を繰り返し処理し、次の **managedServiceAccount** および **clusterPermission** テンプレートを適用します。