Red Hat Advanced Cluster Management for Kubernetes 2.10

GitOps

GitOps

Last Updated: 2024-07-01

Red Hat Advanced Cluster Management for Kubernetes 2.10 GitOps

GitOps

法律通告

摘要

了解如何使用集成的 GitOps 和 Argo CD。

目录

第 1 章 GITOPS 概述

Red Hat OpenShift Container Platform GitOps 和 Argo CD 与原始 Application Lifecycle *Channel* 和 *Subscription* 模型相比，与 Red Hat Advanced Cluster Management for Kubernetes 集成。

GitOps 与 Argo CD 开发集成处于活跃状态，以及为 Argo CD 提供功能增强和更新的大型社区。通过使用 OpenShift Container Platform GitOps Operator，您可以使用 Argo CD 开发中的最新改进，并从 GitOps Operator 订阅接收支持。

请参阅以下主题以了解更多有关与 OpenShift Container Platform GitOps 和 Argo CD 集成的 Red Hat Advanced Cluster Management for Kubernetes 的信息：

- GitOps 控制台
- 将受管集群注册到 OpenShift GitOps operator
- 为 GitOps 配置应用程序放置容限
- 使用 Push 和 Pull 模型部署 Argo CD
- 生成用于安装 GitOps Operator 的策略
- 使用 OpenShift Container Platform GitOps (Argo CD)管理策略定义

1.1. GITOPS 控制台

了解有关集成的 OpenShift Container Platform GitOps 控制台功能的更多信息。创建并查看应用程序，如 *ApplicationSet* 和 *Argo CD* 类型。**ApplicationSet** 代表从控制器生成的 Argo 应用程序。

- 要创建 ArgoCD **ApplicationSet**，您需要从 **Sync 策略**中启用 **Automatically sync when cluster state changes**。
- 对于带有 **kustomization** 控制器的 Flux，找到带有标签 **kustomize.toolkit.fluxcd.io/name=<app_name>** 的 Kubernetes 资源。
- 对于带有 **helm** 控制器的 Flux，找到带有标签 **helm.toolkit.fluxcd.io/name=<app_name>** 的 Kubernetes 资源。
- 您需要 GitOps 集群资源和 GitOps operator 来创建 **ApplicationSet**。如果没有满足这些先决条件，您将无法在控制台中看到 **Argo 服务器**选项来创建 **ApplicationSet**。

重要：可用的操作基于您分配的角色。了解 基于角色的访问控制文档中的访问要求。

- 点 **Launch resource in Search** 搜索相关资源。
- 使用 *Search* 根据每个资源的组件 **kind** 查找应用程序资源。要搜索资源，请使用以下值：

1.1.1. 查询 Argo CD 应用程序

当您搜索 Argo CD 应用程序时，您会被定向到 *Applications* 页面。完成以下步骤，从 *Search* 页面访问 Argo CD 应用程序：

1. 登录到您的 Red Hat Advanced Cluster Management hub 集群。
2. 在控制台标头中选择*搜索*图标。

3. 使用以下值过滤查询：**kind:application** 和 **apigroup:argoproj.io**。

4. 选择要查看的应用程序。*Application* 页面中显示应用的信息的概览。

有关搜索的更多信息，请参阅 控制台简介中的搜索。

1.2. 将受管集群注册到 OPENSHIFT GITOPS OPERATOR

要使用 Push 模型配置 GitOps，您可以将一个或多个 Red Hat Advanced Cluster Management for Kubernetes 受管集群的集合注册到 Red Hat OpenShift Container Platform GitOps operator 实例。注册后，您可以将应用程序部署到这些集群中。设置一个持续 GitOps 环境，以在开发、临时和生产环境中的集群间自动实现应用程序一致性。

1.2.1. 先决条件

1. 您需要在 Red Hat Advanced Cluster Management for Kubernetes 上安装 Red Hat OpenShift GitOps Operator。

2. 导入一个或多个受管集群。

1.2.2. 将受管集群注册到 GitOps

完成以下步骤，将受管集群注册到 GitOps：

1. 创建受管集群集绑定，并将受管集群添加到这些受管集群集绑定中。请参阅 **multicloud-integrations managedclusterset** 中的受管集群集示例。
如需更多信息，请参阅创建 *ManagedClusterSet* 文档。

2. 创建受管集群集绑定到部署 Red Hat OpenShift GitOps 的命名空间。有关将受管集群绑定到 **openshift-gitops** 命名空间的示例，请参阅 **multicloud-integrations** 受管集群集绑定示例。在 *Additional resources* 部分，请参阅 *创建 ManagedClusterSetBinding 资源* 以了解更多有关创建 **ManagedClusterSetBinding** 的信息。如需有关放置信息，请参阅从 ManagedClusterSets 中过滤 ManagedClusters。

3. 在受管集群集绑定中使用的命名空间中，创建一个 **Placement** 自定义资源来选择要注册到 OpenShift Container Platform GitOps operator 实例的一组受管集群。使用 **multicloud-integration** 放置示例作为模板。如需放置信息，请参阅 *使用 ManagedClusterSet with Placement*。
备注：

   - 只有 OpenShift Container Platform 集群注册到 Red Hat OpenShift Container Platform GitOps operator 实例，而不是其他 Kubernetes 集群。

   - 在一些不稳定的网络场景中，受管集群可能会临时处于 **unavailable** 或 **unreachable** 的状态。如需了解更多详细信息，请参阅 *为 Red Hat Advanced Cluster Management 和 OpenShift GitOps 配置放置容限*。

4. 创建 **GitOpsCluster** 自定义资源，将一组受管集群从放置决定注册到指定的 OpenShift GitOps 实例。这可让 OpenShift GitOps 实例将应用程序部署到任何 Red Hat Advanced Cluster Management 受管集群。使用 ***multicloud-integrations*** GitOps 集群示例。
注：引用的**放置**资源必须与 **GitOpsCluster** 资源位于同一个命名空间中。请参见以下示例：

```
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** 值是 **all-openshift-clusters**，并指定为在 **argoNamespace: openshift-gitops** 中安装的 GitOps 实例的目标集群。 **argoServer.cluster** 规格需要 **local-cluster** 值。

5. 保存您的更改。现在，您可以按照 GitOps 工作流来管理应用程序。

1.2.3. GitOps 令牌

当您通过放置和 **ManagedClusterSetBinding** 自定义资源与 GitOps operator 集成时，会在命名空间中创建一个令牌来访问 **ManagedCluster** 命名空间。GitOps 控制器需要把资源同步到受管集群。当用户获得一个管理员访问 GitOps 命名空间以执行应用程序生命周期操作时，用户也获得了对受管集群的此 secret 和 **admin** 级别的访问权限。

如果需要这样做，而不是将用户绑定到命名空间范围的 **admin** 角色，请使用更严格的自定义角色，以及与可创建和用于绑定用户的应用程序资源所需的权限。请参阅以下 **ClusterRole** 示例：

```
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. 其他资源

- 如需了解更多详细信息，请参阅为 GitOps 配置应用程序放置容限。
- 请参阅 **multicloud-integrations** 受管集群集 示例。
- 请参阅创建 *ManagedClusterSet*
- 请参阅 **multicloud-integration** 放置示例。

- 如需放置信息，请参阅放置概述。
- 请参阅 **multicloud-integrations** GitOps 集群 示例。
- 请参阅 **multicloud-integrations** 受管集群集绑定 示例。
- 如需更多信息，请参阅创建 *ManagedClusterSetBinding* 资源文档。
- 请参阅关于 GitOps 以了解更多信息。

1.3. 为 GITOPS 配置应用程序放置容限

Red Hat Advanced Cluster Management 为您提供了将应用程序部署到 Red Hat OpenShift GitOps 的受管集群。

在一些不稳定的网络场景中，受管集群可能会临时处于 **Unavailable** 状态。如果使用 **Placement** 资源来简化应用程序的部署，请为 **Placement** 资源添加以下容限，以继续包含不可用集群。以下示例显示了具有容限的 **Placement** 资源 ：

```
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. 使用 PUSH 和 PULL 模型部署 ARGO CD

使用 *Push 模型*，hub 集群上的 Argo CD 服务器会在受管集群中部署应用程序资源。对于 *Pull 模型*，应用程序资源由 *Propagation 控制器*传播到受管集群，使用 **manifestWork** 。

对于这两种模型，相同的 **ApplicationSet** CRD 用于将应用程序部署到受管集群。

需要的访问权限：集群管理员

- 先决条件
- 架构
- 创建 *ApplicationSet* 自定义资源
- *MulticlusterApplicationSetReport*

1.4.1. 先决条件

查看 Argo CD Pull 模型的以下先决条件：

重要：

- 如果您的 **openshift-gitops-ArgoCD-application-controller** 服务帐户 *没有* 作为集群管理员分配，GitOps 应用程序控制器可能无法部署资源。应用程序状态可能会发送类似以下错误的错误：

```
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"
```

- 在受管集群中安装 **OpenShift Gitops** Operator 后，您必须在同一受管集群上创建 **ClusterRoleBinding** 集群管理员特权。
- 要将 **ClusterRoleBinding** 集群管理员特权添加到受管集群，请参阅以下 YAML 示例：

```
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
```

- 如果您不是集群管理员，需要解决这个问题，请完成以下步骤：

1. 在部署 Argo CD 应用程序的每个受管集群中创建所有命名空间。
2. 将 **managed-by** 标签添加到每个命名空间。如果 Argo CD 应用程序部署到多个命名空间，则每个命名空间都应该由 Argo CD 管理。
   请参见以下带有 **managed-by** 标签的示例：

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

1. 您必须在存储库中声明应用程序的所有应用程序目的地命名空间，并在命名空间中包含 **managed-by** 标签。请参阅 *附加资源* 以了解如何声明命名空间。

请参阅以下要求来使用 Argo CD *Pull* 模型：

- GitOps Operator 必须安装到 hub 集群和 **openshift-gitops** 命名空间中的目标受管集群。
- 所需的 hub 集群 OpenShift Container Platform GitOps operator 必须是版本 1.9.0 或更高版本。
- 所需的受管集群 OpenShift Container Platform GitOps Operator 必须与 hub 集群相同。
- 您需要 *ApplicationSet 控制器* 为受管集群传播 Argo CD 应用程序模板。
- 每个受管集群都必须在 hub 集群上的 Argo CD 服务器命名空间中有一个集群 secret，而 ArgoCD 应用程序设置控制器需要该 secret 来为受管集群传播 Argo CD 应用程序模板。

要创建集群 secret，请创建一个 **gitOpsCluster** 资源，其中包含对 **placement** 资源的引用。**placement** 资源选择所有需要支持 Pull 模型的受管集群。当 GitOps 集群控制器协调时，它会在 Argo CD 服务器命名空间中为受管集群创建集群 secret。

1.4.2. 架构

对于 Push 和 Pull 模型，hub 集群上的 *Argo CD ApplicationSet 控制器*会协调为每个目标受管集群创建应用程序资源。有关这两种模型的架构，请查看以下信息：

1.4.2.1. 架构推送模型

- 使用 Push 模型时，OpenShift Container Platform GitOps 将资源*直接*从集中 hub 集群应用到受管集群。
- 在 hub 集群上运行的 Argo CD 应用程序与 GitHub 存储库通信，并将清单直接部署到受管集群。
- 推送模型实现仅包含 hub 集群上的 Argo CD 应用程序，其中包含受管集群的凭证。hub 集群上的 Argo CD 应用程序可将应用程序部署到受管集群。
- **重要：** 对于需要资源应用程序的大量受管集群，请考虑 OpenShift Container Platform GitOps 控制器内存和 CPU 用量的可能性。要优化资源管理，请参阅配置资源配额或请求。
- 默认情况下，Push 模型用于部署应用程序，除非您将 **apps.open-cluster-management.io/ocm-managed-cluster** 和 **apps.open-cluster-management.io/pull-to-ocm-managed-cluster** 注解添加到 **ApplicationSet** 的 template 部分。

1.4.2.2. 架构 **Pull** 模型

- 拉取模型可以通过减少 hub 集群中的控制器的压力与推送模型提供可扩展性，但需要更多请求和状态报告。
- 使用 Pull 模型时，OpenShift Container Platform GitOps *不会将*资源直接从集中 hub 集群应用到受管集群。Argo CD Application 从 hub 集群传播到受管集群。
- 拉取模型实现应用 OpenShift Cluster Manager 注册、放置和 **manifestWork** API，以便 hub 集群和受管集群之间可以使用安全通信频道来部署资源。
- 每个受管集群都单独与 GitHub 存储库通信，以便在本地部署资源清单，因此您必须在每个受管集群中安装和配置 GitOps operator。
- Argo CD 服务器必须在每个目标受管集群中运行。Argo CD 应用程序资源在受管集群中复制，然后由本地 Argo CD 服务器部署。受管集群中的分布式 Argo CD 应用程序使用 hub 集群上的单个 Argo CD **ApplicationSet** 资源创建。
- 受管集群由 **ocm-managed-cluster** 注解的值决定。
- 要成功实现 Pull 模型，Argo CD 应用程序控制器必须在 **ApplicationSet** 的 template 部分中*忽略*带有 **argocd.argoproj.io/skip-reconcile** 注解的 Push 模型应用程序资源。
- 对于 Pull 模型，受管集群中的 *Argo CD Application 控制器会* 协调以部署应用程序。
- hub 集群上的 Pull model *Resource sync 控制器* 会定期查询 OpenShift Cluster Manager 在每个受管集群上搜索 V2 组件，以检索每个 Argo CD 应用程序的资源列表和错误消息。
- hub 集群上的 *聚合控制器* 使用资源同步控制器的数据以及 **manifestWork** 的状态信息，从集群间创建和更新 **MulticlusterApplicationSetReport**。

- 部署的状态被收集回 hub 集群，但不会传输所有详细信息。定期提取其他状态更新以提供概述。状态反馈不是实时的，每个受管集群 GitOps operator 需要与 Git 存储库通信，这会导致多个请求。

1.4.3. 创建 *ApplicationSet* 自定义资源

Argo CD **ApplicationSet** 资源用于在用于获取受管集群列表的 generator 字段中使用带有 **放置资源** 的 Push 或 Pull 模型在受管集群中部署应用程序。

1. 对于 Pull 模型，将应用程序的目的地设置为默认本地 Kubernetes 服务器，如下例所示。应用程序由受管集群上的应用程序控制器在本地部署。

2. 添加覆盖默认 Push 模型所需的注解，如下例所示 **ApplicationSet** YAML，该 YAML 使用带有模板注解的 Pull 模型：

```
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 模型需要 **apps.open-cluster-management.io/ocm-managed-cluster**。
2. **argocd.argoproj.io/skip-reconcile** 需要忽略 Push 模型资源。
3. Pull 模型还需要 **apps.open-cluster-management.io/pull-to-ocm-managed-cluster: "true** "。

1.4.4. *MulticlusterApplicationSetReport*

- 对于 Pull 模型，**MulticlusterApplicationSetReport** 聚合受管集群中的应用程序状态。
- 该报告包括资源列表以及每个受管集群的应用程序的整体状态。
- 为每个 Argo CD ApplicationSet 资源创建一个单独的报告资源。该报告与 **ApplicationSet** 在同一命名空间中创建。
- 该报告包括以下项目：
  1. Argo CD 应用程序的资源列表
  2. 每个 Argo CD 应用程序的整体同步和健康状况
  3. 每个集群的出错信息，其中整个状态为 **sync** 或 **unhealthy**
  4. 受管集群的所有状态概述状态
- *资源同步控制器*和*聚合控制器*每 10 秒运行一次，以创建报告。
- 两个控制器以及 Propagation 控制器，在同一 **multicluster-integrations** pod 中的独立容器中运行，如下例所示：

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

以下是 **guestbook** 应用程序的 **MulticlusterApplicationSetReport** YAML 文件示例：

```
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"
```

注：如果资源无法部署，则资源不会包含在资源列表中。如需更多信息，请参阅错误消息。

1.4.5. 其他资源

- 请参阅 OpenShift Container Platform 文档中的使用集群配置配置 OpenShift 集群。
- 请参阅 OpenShift Container Platform 文档中的 设置 Argo CD 实例。

1.5. 使用 OPENSHIFT CONTAINER PLATFORM GITOPS (ARGO CD)管理策略定义

已弃用： PlacementRule

根据 Argo CD，您可以使用 OpenShift Container Platform GitOps 管理策略定义。要允许此工作流，您必须授予 OpenShift Container Platform GitOps 访问权限，以便在 Red Hat Advanced Cluster Management hub 集群中创建策略。完成以下步骤，为 OpenShift Container Platform GitOps 创建 **ClusterRole** 资源，可访问创建、读取、更新和删除策略和放置：

1. 从控制台创建 **ClusterRole**。您的 **ClusterRole** 可能类似以下示例：

```
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: openshift-gitops-policy-admin
rules:
  - verbs:
      - get
      - list
      - watch
      - create
      - update
      - patch
      - delete
    apiGroups:
      - policy.open-cluster-management.io
    resources:
      - policies
      - 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. 创建一个 **ClusterRoleBinding** 对象，为 OpenShift Container Platform GitOps 服务帐户授予 **openshift-gitops-policy-admin ClusterRole** 对象的访问权限。**ClusterRoleBinding** 可能类似以下示例：

```
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
```

当使用 OpenShift Container Platform GitOps 部署 Red Hat Advanced Cluster Management 策略定义时，会在每个受管集群命名空间中创建策略副本。这些副本称为复制策略。要防止 OpenShift Container Platform GitOps 重复删除此复制策略，或者显示 ArgoCD **应用程序** 没有同步，则 Red Hat Advanced Cluster Management 策略框架中会自动设置 **argocd.argoproj.io/compare-options: IgnoreExtraneous** 注解。

Argo CD 使用标签和注解来跟踪对象。要使复制策略不会在 Argo CD 中都显示，您可以在 Red Hat Advanced Cluster Management 策略定义中将 **spec.copyPolicyMetadata** 设置为 **false**，以禁用 Argo CD 跟踪标签和注解从复制到复制策略。

1.5.1. 将 Policy Generator 与 OpenShift Container Platform GitOps (ArgoCD) 集成

根据 Argo CD，您可以使用 OpenShift Container Platform GitOps 通过 GitOps 使用 Policy Generator 生成策略。因为 Policy Generator 没有预安装在 OpenShift Container Platform GitOps 容器镜像中，所以必须进行一些自定义。为了继续操作，必须在 Red Hat Advanced Cluster Management hub 集群中安装了 OpenShift Container Platform GitOps Operator，并确保登录到 hub 集群。

要使 OpenShift Container Platform GitOps 在运行 Kustomize 时有权访问 Policy Generator，需要将初始容器从 Red Hat Advanced Cluster Management Application Subscription 容器镜像复制到 OpenShift Container Platform GitOps 容器。另外，OpenShift Container Platform GitOps 必须配置为在运行 Kustomize 时提供 **--enable-alpha-plugins** 标志。完成以下步骤：

1. 使用以下命令开始编辑 OpenShift Container Platform GitOps **argocd** 对象：

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

2. 修改 OpenShift Container Platform GitOps **argocd** 对象，使其包含以下额外 YAML 内容。当发布新的 Red Hat Advanced Cluster Management 主版本且您要将 Policy Generator 更新至更新的版本时，您需要将 **registry.redhat.io/rhacm2/multicluster-operators-subscription-rhel9** 镜像更新到较新的标签。查看以下示例，将 < **version&gt**; 替换为 2.10 或您所需的 Red Hat Advanced Cluster Management 版本：

```
apiVersion: argoproj.io/v1beta1
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-tmp
        name: policy-generator
    volumeMounts:
    - mountPath: /etc/kustomize/plugin/policy.open-cluster-management.io/v1/policygenerator
      name: policy-generator
    volumes:
    - emptyDir: {}
      name: policy-generator
```

注： 另外，您可以创建一个包含 **ArgoCD** 清单和模板的 **ConfigurationPolicy** 资源，以匹配 **MulticlusterHub** 中设置的版本：

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

如果要在生成策略前在 Kustomize 目录中启用 Helm chart 处理，请在 **spec.repo.env** 字段中将环境变量 **POLICY_GEN_ENABLE_HELM** 设置为 **"true"** ：

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

3. 现在，OpenShift Container Platform GitOps 可以使用 Policy Generator，OpenShift Container Platform GitOps 必须被授予在 Red Hat Advanced Cluster Management hub 集群中创建策略的访问权限。创建名为 **openshift-gitops-policy-admin** 的 **ClusterRole** 资源，其具有创建、读取、更新和删除策略和放置的访问权限。请参阅 ealier **ClusterRole** 资源示例。

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

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

1.5.2. 其他资源

- 请参阅 Argo CD 文档。

1.6. 生成用于安装 GITOPS OPERATOR 的策略

Red Hat Advanced Cluster Management 策略的一个常见用途是在一个或多个受管 Red Hat OpenShift Container Platform 集群上安装 Operator。继续阅读以了解如何使用 Policy Generator 生成策略，并使用生成的策略安装 OpenShift Container Platform GitOps Operator：

1.6.1. 生成安装 OpenShift Container Platform GitOps 的策略

您可以使用 Policy Generator 生成安装 OpenShift Container Platform GitOps 的策略。OpenShift Container Platform GitOps operator 提供 *所有命名空间* 安装模式，您可以在以下示例中查看它们。创建名为 **openshift-gitops-subscription.yaml** 的 **Subscription** 清单文件，如下例所示：

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

```
  name: openshift-gitops-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
```

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

需要 **PolicyGenerator** 配置文件。使用名为 **policy-generator-config.yaml** 的配置文件来生成策略，以便在所有 OpenShift Container Platform 受管集群上安装 OpenShift Container Platform GitOps。请参见以下示例：

```
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
```

最后所需的文件是 **kustomization.yaml**，它需要以下配置：

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

生成的策略可能类似以下带有 **PlacementRule** 的文件（已弃用）：

```
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
```

支持 OpenShift Container Platform 文档中的从清单生成的策略。OpenShift Container Platform 文档中的任何配置指南都可以使用 Policy Generator 应用。

1.6.2. 使用带有 *OperatorGroup* 的策略依赖项

当使用 **OperatorGroup** 清单安装 Operator 时，**OperatorGroup** 必须在创建 **Subscription** 前存在于集群中。使用策略依赖项功能以及 Policy Generator，确保在执行 订阅 策略前 **OperatorGroup** 策略兼容。

通过按您想要的顺序列出清单来设置策略依赖项。例如，您可能想要首先创建命名空间策略，然后创建 **OperatorGroup**，最后再创建 **Subscription**。

启用 **policyDefaults.orderManifests** 参数，并在 Policy Generator 配置清单中禁用 **policyDefaults.consolidateManifests**，以自动设置清单之间的依赖项。

1.6.3. 其他资源

- 请参阅 生成安装 Compliance Operator 的策略。
- 如需了解更多详细信息，请参阅使用 GitOps 部署策略。
- 如需了解更多详细信息，请参阅了解 OpenShift GitOps 和 Operator 文档。
- 请参阅使用 CLI 将 Operator 添加到集群 - 从 OperatorHub 安装
- 如需了解更多详细信息，请参阅 Compliance Operator 文档。
- 查看*所有命名空间安装模式*。
- 请参阅 *namespaced* 安装模式。
- 请参阅在部署 pod 前使用初始容器执行任务。
- 请参阅 Argo CD。
- 查看以下 OpenShift Container Platform 支持的 YAML 输入示例：
  - 安装后集群任务
  - 配置审计日志策略
  - 关于将日志转发到第三方系统

1.7. 为 ARGO CD PUSH 模型创建自定义服务帐户

通过在 hub 集群中创建 **managedserviceaccount** 资源，在受管集群中创建服务帐户。使用 **clusterpermission** 资源为服务帐户授予特定权限。

创建用于 Argo CD push 模型的自定义服务帐户包括以下优点：

- Application Manager 附加组件在每个受管集群上运行。默认情况下，Argo CD 控制器使用服务帐户 Application Manager 将这些资源推送到受管集群。
- Application manager 服务帐户具有大量权限，因为应用程序订阅附加组件使用 Application manager 服务在受管集群中部署应用程序。如果您希望一组有限的权限，请不要使用 Application manager 服务帐户。
- 您可以指定您希望 Argo CD push 模型使用的不同服务帐户。当 Argo CD 控制器将资源从集中 hub 集群推送到受管集群时，您可以使用与默认应用程序管理器不同的服务帐户。通过使用不同的服务帐户，您可以控制为此服务帐户授予的权限。
- 服务帐户必须存在于受管集群上。为方便创建具有关联权限的服务帐户，请在集中式 hub 集群上使用 **managedserviceaccount** 资源和新的 **clusterpermission** 资源。

完成所有以下步骤后，您可以为受管服务帐户授予集群权限。使用集群权限，您的受管服务帐户具有在受管集群中部署应用程序资源所需的权限。完成以下步骤以：

1. 第 1.7.1 节 “创建受管服务帐户”
2. 第 1.7.2 节 “创建集群权限”
3. 第 1.7.3 节 “在 GitOpsCluster 资源中使用受管服务帐户”

4. 第 1.7.4 节 “创建 Argo CD 应用程序”

5. 第 1.7.5 节 “使用策略创建受管服务帐户和集群权限”

1.7.1. 创建受管服务帐户

hub 上的 **managedserviceaccount** 自定义资源提供了一种便捷的方法，可以在受管集群上创建 **serviceaccounts**。当在 hub 集群上的 < **managed_cluster** > 命名空间中创建 **managedserviceccount** 自定义资源时，会在受管集群上创建一个 **serviceccount**。

要创建受管服务帐户，请参阅启用 managedserviceaccount add-ons。

1.7.2. 创建集群权限

创建服务帐户时，它没有关联任何权限。要为新服务帐户授予权限，请使用 **clusterpermission** 资源。**集群permission** 资源在 hub 上的受管集群命名空间中创建。它提供了一种便捷的方式来创建角色，在受管集群中的集群角色资源，并通过 **rolebinding** 或 **clusterrolebinding** 资源将它们绑定到服务帐户。

1. 要为部署到 & **lt;managed cluster> 上的 mortgage 命名空间的示例 mortgage 应用程序授予 <managed-sa -sample&gt**; 服务帐户权限，请使用以下内容创建一个 YAML：

```
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. 将 YAML 文件保存到名为 **cluster-permission.yaml** 的文件中。

3. 运行 **oc apply -f cluster-permission.yaml**。

4. 示例 & **lt;clusterpermission&** gt; 在 mortgage 命名空间中创建一个名为 < **clusterpermission-msa-subject-sample** > 的角色。如果尚不存在，请创建一个命名空间 **mortgage**。

5. 查看在 < **managed cluster&gt**; 上创建的资源。

创建示例 < **clusterpermission&** gt; 后，会在示例受管集群中创建以下资源：

- 在 default 命名空间中，一个名为 < **clusterpermission-msa-subject-sample** > 的角色。
- 在 default 命名空间中一个名为 **<clusterpermission-msa-subject-sample** > 的 roleBinding，用于将角色绑定到受管服务帐户。
- mortgage 命名空间中的一个名为 **<clusterpermission-msa-subject-sample** > 的角色。
- mortgage 命名空间中的一个名为 & **lt;clusterpermission-msa-subject-sample** > 的 roleBinding，用于将角色绑定到受管服务帐户。
- 一个名为 < **clusterpermission-msa-subject-sample** > 的 clusterRole。
- 一个名为 < **clusterpermission-msa-subject-sample&gt**; 的 clusterRoleBinding，用于将 clusterRole 绑定到受管服务帐户。

1.7.3. 在 GitOpsCluster 资源中使用受管服务帐户

GitOpsCluster 资源使用放置将所选受管集群导入到 Argo CD 中，包括创建包含用于访问集群的信息的 Argo CD 集群 secret。默认情况下，Argo CD 集群 secret 使用 application manager 服务帐户来访问受管集群。

1. 要将 GitOpsCluster 资源更新为使用受管服务帐户，请使用受管服务帐户的名称添加 **managedServiceAccountRef** 属性。

2. 要创建 GitOpsCluster 自定义资源，请将以下 YAML 保存为 Gitops.YAML ：

```
---
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. 将 YAML 文件保存到名为 **gitops.yaml** 的文件中。

4. 运行 **oc apply -f gitops.yaml**。

5. 进入 **openshift-gitops** 命名空间，并验证是否有名为 <managed cluster- **managed-sa-sample-cluster-secret> 的新 Argo CD 集群 secret** ：

```
% 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. 创建 Argo CD 应用程序

使用推送模型从 Argo CD 控制台部署 Argo CD 应用程序。Argo CD 应用程序使用受管服务帐户 < **managed-sa-sample** > 部署。

1. 登录到 Argo CD 控制台。

2. 点 **Create a new application**。

3. 选择集群 URL。

4. 进入 Argo CD 应用程序，并确认它具有您传播到 < **managed cluster** > 的给定权限，如角色和集群角色。

1.7.5. 使用策略创建受管服务帐户和集群权限

> 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

当您将 **managedServiceAccount** 和 **clusterPermission** 资源应用到 hub 集群时，此策略的放置会绑定到本地集群。在 GitOpsCluster 资源放置中，将这些资源复制到所有受管集群的受管集群命名空间中。

使用策略创建 **managedServiceAccount** 和 **clusterPermission** 资源包括以下属性：

- 更新策略中的 **managedServiceAccount** 和 **clusterPermission** 对象模板会导致在所有受管集群中更新所有 **managedServiceAccount** 和 **clusterPermission** 资源。
- 直接更新到 **managedServiceAccount** 和 **clusterPermission** 资源会恢复到原始状态，因为策略强制执行它。
- 如果 GitOpsCluster 放置的放置决定更改，策略将管理受管集群命名空间中的资源的创建和删除。

1. 要为 YAML 创建策略以创建受管服务帐户和集群权限，请使用以下内容创建一个 YAML：

```
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. 将 YAML 文件保存到名为 **policy.yaml** 的文件中。
2. 运行 **oc apply -f policy.yaml**。
3. 在策略的对象模板中，它会迭代 GitOpsCluster 关联的放置决定，并应用以下 **managedServiceAccount** 和 **clusterPermission** 模板：