Red Hat OpenShift GitOps 1.12

声明性集群配置

使用 OpenShift GitOps 配置 OpenShift 集群，并使用 GitOps CLI 在默认和代码模式中创建并同步应用程序。

Last Updated: 2024-05-15

Red Hat OpenShift GitOps 1.12 声明性集群配置

使用 OpenShift GitOps 配置 OpenShift 集群，并使用 GitOps CLI 在默认和代码模式中创建并同步应用程序。

法律通告

摘要

本文档提供有关将 Argo CD 配置为将 Git 目录的内容递归与包含集群的自定义配置的应用程序同步的说明。它还讨论了如何使用 GitOps CLI 在默认和代码模式中创建和同步应用程序。

目录

第 1 章 通过部署带有集群配置的应用程序来配置 OPENSHIFT 集群

使用 Red Hat OpenShift GitOps，您可以将 Argo CD 配置为将 Git 目录的内容与包含集群自定义配置的应用程序递归同步。

1.1. 先决条件

- 以管理员身份登录到 OpenShift Container Platform 集群。
- 您已在 OpenShift Container Platform 集群中安装了 Red Hat OpenShift GitOps Operator。

1.2. 使用 ARGO CD 实例管理集群范围的资源

要管理集群范围的资源，请更新 Red Hat OpenShift GitOps Operator 的现有 **Subscription** 对象，并将 Argo CD 实例的命名空间添加到 **spec** 部分中的 **ARGOCD_CLUSTER_CONFIG_NAMESPACES** 环境变量中。

流程

1. 在 Web 控制台的 **Administrator** 视角中，进入到 **Operators → Installed Operators → Red Hat OpenShift GitOps → Subscription**。
2. 点 **Actions** 下拉菜单，然后点 **Edit Subscription**。
3. 在 **openshift-gitops-operator** 订阅详情页面的 **YAML** 选项卡下，通过将 Argo CD 实例的命名空间添加到 **spec** 部分中的 **ARGOCD_CLUSTER_CONFIG_NAMESPACES** 环境变量来编辑 **Subscription** YAML 文件：

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: openshift-gitops-operator
  namespace: openshift-operators
# ...
spec:
  config:
    env:
    - name: ARGOCD_CLUSTER_CONFIG_NAMESPACES
      value: openshift-gitops, <list of namespaces of cluster-scoped Argo CD instances>
# ...
```

4. 要验证 Argo 实例是否已配置有集群角色来管理集群范围的资源，请执行以下步骤：
   a. 进入到 **User Management → Roles**，然后从 **Filter** 下拉菜单中选择 **Cluster-wide Roles**。
   b. 使用 **Search by name** 字段搜索 **argocd-application-controller**。
   Roles 页面显示创建的集群角色。

提示

或者，在 OpenShift CLI 中运行以下命令：

```
oc auth can-i create oauth -n openshift-gitops --as system:serviceaccount:openshift-gitops:openshift-gitops-argocd-application-controller
```

输出 **yes** 代表 Argo 实例配置了集群角色来管理集群范围的资源。否则，检查您的配置并根据需要执行必要的步骤。

1.3. ARGO CD 实例的默认权限

默认情况下，Argo CD 实例具有以下权限：

- Argo CD 实例具有 **admin** 权限，以便仅管理部署它的命名空间中的资源。例如，在 **foo** 命名空间中部署的 Argo CD 实例具有 **admin** 权限，仅管理该命名空间的资源。
- Argo CD 具有以下集群范围的权限，因为 Argo CD 需要对资源进行集群范围的**读取**权限才能正常工作：

```
- verbs:
  - get
  - list
  - watch
  apiGroups:
  - '*'
  resources:
  - '*'
- verbs:
  - get
  - list
  nonResourceURLs:
  - '*'
```

注意

- 您可以编辑运行 Argo CD 的 **argocd-server** 和 **argocd-application-controller** 组件使用的集群角色，以便**写入**权限仅限于您希望 Argo CD 管理的命名空间和资源。

```
$ oc edit clusterrole argocd-server
$ oc edit clusterrole argocd-application-controller
```

1.4. 在集群级别运行 ARGO CD 实例

默认 Argo CD 实例和附带的控制器（由 Red Hat OpenShift GitOps Operator 安装）现在可以通过设置一个简单的配置切换在集群的基础架构节点上运行。

流程

1. 标记现有节点：

```
$ oc label node <node-name> node-role.kubernetes.io/infra=""
```

2. 可选：如果需要，您还可以在基础架构节点上应用污点并隔离工作负载，并防止其他工作负载在这些节点上调度：

```
$ oc adm taint nodes -l node-role.kubernetes.io/infra \
infra=reserved:NoSchedule infra=reserved:NoExecute
```

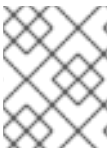
3. 在 **GitOpsService** 自定义资源中添加 **runOnInfra** 切换：

```
apiVersion: pipelines.openshift.io/v1alpha1
kind: GitopsService
metadata:
  name: cluster
spec:
  runOnInfra: true
```

4. 可选：如果将污点添加到节点，则在 **GitOpsService** 自定义资源中添加 **容限**，例如：

```
spec:
  runOnInfra: true
  tolerations:
  - effect: NoSchedule
    key: infra
    value: reserved
  - effect: NoExecute
    key: infra
    value: reserved
```

5. 通过在控制台 UI 中查看 **Pods → Pod details**，验证 **openshift-gitops** 命名空间中的工作负载现在已调度到基础架构节点上。



注意

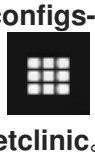
任何手工添加到默认 Argo CD 自定义资源中的 **nodeSelectors** 和 **tolerations**，都会被 **GitOpsService** 自定义资源的 **tolerations** 覆盖。

其他资源

- 要了解更多有关污点和容限的信息，请参阅使用节点污点控制 pod 放置。
- 如需有关基础架构机器集的更多信息，请参阅 创建基础架构机器集。

1.5. 使用 ARGO CD 仪表板创建应用程序

Argo CD 提供了一个仪表板，供您创建应用程序。

此示例工作流逐步指导您完成将 Argo CD 配置为递归将 **cluster** 目录中的内容同步到 **cluster-configs-** 应用程序。目录定义了 OpenShift Container Platform Web 控制台集群配置，在 web 控制台中的  菜单下向 **Red Hat Developer Blog - Kubernetes**添加链接，并在集群中定义命名空间 **spring-petclinic**。

先决条件

- 以管理员身份登录到 OpenShift Container Platform 集群。

- 您已在 OpenShift Container Platform 集群中安装了 Red Hat OpenShift GitOps Operator。
- 已登陆到 Argo CD 实例。

流程

1. 在 Argo CD 控制面板中，单击 **NEW APP** 以添加新 Argo CD 应用。
2. 对于此工作流，使用以下配置创建一个 **cluster-configs** 应用程序：

   应用程序名称
   cluster-configs
   project
   default
   同步策略
   Manual（手动）
   仓库 URL
   https://github.com/redhat-developer/openshift-gitops-getting-started
   修订
   HEAD
   路径
   cluster
   目的地
   https://kubernetes.default.svc
   命名空间
   spring-petclinic
   Directory Recurse
   checked

3. 单击 **CREATE** 以创建应用程序。
4. 打开 Web 控制台的 **Administrator** 视角，再展开 **Administration → Namespaces**。
5. 搜索并选择命名空间，然后在 **Label** 字段中输入 **argocd.argoproj.io/managed-by=openshift-gitops**，以便 **openshift-gitops** 命名空间中的 Argo CD 实例可以管理您的命名空间。

1.6. 使用 oc 工具创建应用程序

您可以使用 **oc** 工具在终端中创建 Argo CD 应用程序。

先决条件

- 您已在 OpenShift Container Platform 集群中安装了 Red Hat OpenShift GitOps Operator。
- 已登陆到 Argo CD 实例。

流程

1. 下载 示例应用程序：

   -

```
$ git clone git@github.com:redhat-developer/openshift-gitops-getting-started.git
```

2. 创建应用程序：

```
$ oc create -f openshift-gitops-getting-started/argo/app.yaml
```

3. 运行 **oc get** 命令以查看所创建的应用程序：

```
$ oc get application -n openshift-gitops
```

4. 在部署应用程序的命名空间中添加标签，以便 **openshift-gitops** 命名空间中的 Argo CD 实例可以管理它：

```
$ oc label namespace spring-petclinic argocd.argoproj.io/managed-by=openshift-gitops
```

1.7. 使用 GITOPS CLI 在默认模式中创建应用程序

您可以使用 GitOps **argocd** CLI 在默认模式中创建应用程序。

此示例工作流逐步指导您完成将 Argo CD 配置为递归将 **cluster** 目录中的内容同步到 **cluster-configs-** 应用程序。目录定义 OpenShift Container Platform 集群配置和集群中的 **spring-petclinic** 命名空间。

先决条件

- 您已在 OpenShift Container Platform 集群中安装了 Red Hat OpenShift GitOps Operator。
- 已安装 OpenShift CLI(**oc**)。
- 已安装 Red Hat OpenShift GitOps **argocd** CLI。
- 已登陆到 Argo CD 实例。

流程

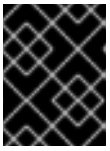
1. 获取 Argo CD 服务器的 **admin** 帐户密码：

```
$ ADMIN_PASSWD=$(oc get secret openshift-gitops-cluster -n openshift-gitops -o
jsonpath='{.data.admin\.password}' | base64 -d)
```

2. 获取 Argo CD 服务器 URL：

```
$ SERVER_URL=$(oc get routes openshift-gitops-server -n openshift-gitops -o
jsonpath='{.status.ingress[0].host}')
```

3. 使用 **admin** 帐户密码并用单引号括起来来登录到 Argo CD 服务器：



重要

用单引号括起密码可确保 shell 不会误解特殊字符，如 **$**。始终使用单引号括起密码的字面值。

```
$ argocd login --username admin --password ${ADMIN_PASSWD} ${SERVER_URL}
```

Example

```
$ argocd login --username admin --password '<password>' openshift-gitops.openshift-gitops.apps-crc.testing
```

4. 通过列出所有应用程序，验证您是否可以在默认模式下运行 **argocd** 命令：

```
$ argocd app list
```

如果配置正确，则现有应用程序将使用以下标头列出：

输出示例

```
NAME CLUSTER NAMESPACE  PROJECT  STATUS  HEALTH  SYNCPOLICY  CONDITIONS  REPO PATH TARGET
```

5. 以默认模式创建应用程序：

```
$ argocd app create app-cluster-configs \
  --repo https://github.com/redhat-developer/openshift-gitops-getting-started.git \
  --path cluster \
  --revision main \
  --dest-server  https://kubernetes.default.svc \
  --dest-namespace spring-petclinic \
  --directory-recurse \
  --sync-policy none \
  --sync-option Prune=true \
  --sync-option CreateNamespace=true
```

6. 标记 **spring-petclinic** 目标命名空间，使其由 **openshif-gitops** Argo CD 实例管理：

```
$ oc label ns spring-petclinic "argocd.argoproj.io/managed-by=openshift-gitops"
```

7. 列出可用的应用程序，以确认应用程序是否已成功创建：

```
$ argocd app list
```

虽然 **cluster-configs** Argo CD 应用程序具有 **Healthy** 状态，但它不会 **因为没有** 同步策略而自动同步，从而导致它处于 **OutOfSync** 状态。

1.8. 使用 GITOPS CLI 以核心模式创建应用程序

您可以使用 GitOps **argocd** CLI 在 **core** 模式中创建应用程序。

此示例工作流逐步指导您完成将 Argo CD 配置为递归将 **cluster** 目录中的内容同步到 **cluster-configs-** 应用程序。目录定义 OpenShift Container Platform 集群配置和集群中的 **spring-petclinic** 命名空间。

先决条件

- 您已在 OpenShift Container Platform 集群中安装了 Red Hat OpenShift GitOps Operator。
- 已安装 OpenShift CLI(**oc**)。

- 已安装 Red Hat OpenShift GitOps **argocd** CLI。

流程

1. 使用 **oc** CLI 工具登录到 OpenShift Container Platform 集群：

```
$ oc login -u <username> -p <password> <server_url>
```

Example

```
$ oc login -u kubeadmin -p '<password>' https://api.crc.testing:6443
```

2. 检查 **kubeconfig** 文件中是否正确设置了上下文：

```
$ oc config current-context
```

3. 将当前上下文的默认命名空间设置为 **openshift-gitops** ：

```
$ oc config set-context --current --namespace openshift-gitops
```

4. 设置以下环境变量来覆盖 Argo CD 组件名称：

```
$ export ARGOCD_REPO_SERVER_NAME=openshift-gitops-repo-server
```

5. 通过列出所有应用程序，验证您是否可以在 **core** 模式下运行 **argocd** 命令：

```
$ argocd app list --core
```

如果配置正确，则现有应用程序将使用以下标头列出：

输出示例

```
NAME CLUSTER NAMESPACE  PROJECT  STATUS  HEALTH   SYNCPOLICY
CONDITIONS  REPO PATH TARGET
```

6. 以 **core** 模式创建应用程序：

```
$ argocd app create app-cluster-configs --core \
   --repo https://github.com/redhat-developer/openshift-gitops-getting-started.git \
   --path cluster \
   --revision main \
   --dest-server  https://kubernetes.default.svc \
   --dest-namespace spring-petclinic \
   --directory-recurse \
   --sync-policy none \
   --sync-option Prune=true \
   --sync-option CreateNamespace=true
```

7. 标记 **spring-petclinic** 目标命名空间，使其由 **openshif-gitops** Argo CD 实例管理：

```
$ oc label ns spring-petclinic "argocd.argoproj.io/managed-by=openshift-gitops"
```

8. 列出可用的应用程序，以确认应用程序是否已成功创建：

```
$ argocd app list --core
```

虽然 **cluster-configs** Argo CD 应用程序具有 **Healthy** 状态，但它不会 **因为没有** 同步策略而自动同步，从而导致它处于 **OutOfSync** 状态。

1.9. 将应用程序与 GIT 存储库同步

您可以通过修改 Argo CD 的同步策略，将应用程序与 Git 存储库同步。策略修改会自动将集群配置的更改从 Git 存储库应用到集群。

流程

1. 在 Argo CD 仪表板中，**cluster-configs** Argo CD 应用程序的状态为 **Missing** 和 **OutOfSync**。因为应用程序配置了手动同步策略，所以 Argo CD 不会自动同步。

2. 点 **cluster-configs** 标题上的 **SYNC**，查看更改，然后点 **SYNCHRONIZE**。Argo CD 将自动检测 Git 存储库中的任何更改。如果更改了配置，Argo CD 会将 **cluster-configs** 的状态改为 **OutOfSync**。您可以修改 Argo CD 的同步策略，以自动将 Git 存储库中的更改应用到集群。

3. 现在，**cluster-configs** Argo CD 应用程序的状态为 **Healthy** 和 **Synced**。点 **cluster-configs** 标题检查同步资源的详情及其在集群中的状态。

4. 进入到 OpenShift Container Platform Web 控制台并点击  以验证 **Red Hat Developer Blog - Kubernetes** 的链接现在是否存在。

5. 导航到 **Project** 页面并搜索 **spring-petclinic** 命名空间，以验证它是否已添加到集群中。集群配置已成功与集群同步。

1.10. 使用 GITOPS CLI 以默认模式同步应用程序

您可以使用 GitOps **argocd** CLI 在默认模式下同步应用程序。

此示例工作流逐步指导您完成将 Argo CD 配置为递归将 **cluster** 目录中的内容同步到 **cluster-configs-** 应用程序。目录定义 OpenShift Container Platform 集群配置和集群中的 **spring-petclinic** 命名空间。

先决条件

- 您已在 OpenShift Container Platform 集群中安装了 Red Hat OpenShift GitOps Operator。
- 已登陆到 Argo CD 实例。
- 已安装 OpenShift CLI(**oc**)。
- 已安装 Red Hat OpenShift GitOps **argocd** CLI。

流程

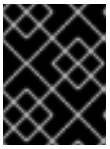
1. 获取 Argo CD 服务器的 **admin** 帐户密码：

```
$ ADMIN_PASSWD=$(oc get secret openshift-gitops-cluster -n openshift-gitops -o
jsonpath='{.data.admin\.password}' | base64 -d)
```

2. 获取 Argo CD 服务器 URL：

```
$ SERVER_URL=$(oc get routes openshift-gitops-server -n openshift-gitops -o
jsonpath='{.status.ingress[0].host}')
```

3. 使用 **admin** 帐户密码并用单引号括起来来登录到 Argo CD 服务器：



重要

用单引号括起密码可确保 shell 不会误解特殊字符，如 **$**。始终使用单引号括起密码的字面值。

```
$ argocd login --username admin --password ${ADMIN_PASSWD} ${SERVER_URL}
```

Example

```
$ argocd login --username admin --password '<password>' openshift-gitops.openshift-
gitops.apps-crc.testing
```

4. 因为应用程序配置了 **none** 同步策略，所以您必须手动触发同步操作：

```
$ argocd app sync openshift-gitops/app-cluster-configs
```

5. 列出应用程序，以确认应用程序具有 **Healthy** 和 **Synced** 状态：

```
$ argocd app list
```

1.11. 使用 GITOPS CLI 以核心模式同步应用程序

您可以使用 GitOps **argocd** CLI 在 **core** 模式中同步应用程序。

此示例工作流逐步指导您完成将 Argo CD 配置为递归将 **cluster** 目录中的内容同步到 **cluster-configs-** 应用程序。目录定义 OpenShift Container Platform 集群配置和集群中的 **spring-petclinic** 命名空间。

先决条件

- 您已在 OpenShift Container Platform 集群中安装了 Red Hat OpenShift GitOps Operator。
- 已安装 OpenShift CLI(**oc**)。
- 已安装 Red Hat OpenShift GitOps **argocd** CLI。

流程

1. 使用 **oc** CLI 工具登录到 OpenShift Container Platform 集群：

```
$ oc login -u <username> -p <password> <server_url>
```

Example

```
$ oc login -u kubeadmin -p '<password>' https://api.crc.testing:6443
```

2. 检查 **kubeconfig** 文件中是否正确设置了上下文：

```
$ oc config current-context
```

3. 将当前上下文的默认命名空间设置为 **openshift-gitops** ：

```
$ oc config set-context --current --namespace openshift-gitops
```

4. 设置以下环境变量来覆盖 Argo CD 组件名称：

```
$ export ARGOCD_REPO_SERVER_NAME=openshift-gitops-repo-server
```

5. 因为应用程序配置了 **none** 同步策略，所以您必须手动触发同步操作：

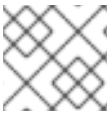
```
$ argocd app sync --core openshift-gitops/app-cluster-configs
```

6. 列出应用程序，以确认应用程序具有 **Healthy** 和 **Synced** 状态：

```
$ argocd app list --core
```

1.12. 集群配置的内置权限

默认情况下，Argo CD 实例具有管理特定集群范围资源的权限，如集群 Operator、可选 OLM Operator 和用户管理。



注意

Argo CD 没有 cluster-admin 权限。

Argo CD 实例的权限：

Resources	描述
资源组	配置用户或管理员
operators.coreos.com	由 OLM 管理的可选 Operator
user.openshift.io , **rbac.authorization.k8s.io**	组、用户及其权限
config.openshift.io	由 CVO 管理的 control plane Operator，用于配置集群范围的构建配置、registry 配置和调度程序策略
storage.k8s.io	Storage
console.openshift.io	控制台自定义

1.13. 为集群配置添加权限

您可以授予 Argo CD 实例的权限来管理集群配置。创建具有额外权限的集群角色，然后创建新的集群角色绑定以将集群角色与服务帐户关联。

先决条件

- 您可以使用 **cluster-admin** 权限访问 OpenShift Container Platform 集群，并登录到 web 控制台。
- 您已在 OpenShift Container Platform 集群中安装了 Red Hat OpenShift GitOps Operator。

流程

1. 在 Web 控制台中，选择 **User Management → Roles → Create Role**。使用以下 **ClusterRole** YAML 模板来添加规则来指定额外权限。

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: secrets-cluster-role
rules:
- apiGroups: [""]
  resources: ["secrets"]
  verbs: ["*"]
```

2. 点 **Create** 添加集群角色。
3. 要创建集群角色绑定，请选择 **User Management → Role Bindings → Create Binding**。
4. 从 **Project** 下拉菜单中选择 **All Projects**。
5. 点 **Create binding**。
6. 将 **Binding type** 选择为 **Cluster-wide role binding(ClusterRoleBinding)**。
7. 为 **RoleBinding 名称** 输入一个唯一值。
8. 从下拉列表中选择新创建的集群角色或现有集群角色。
9. 选择 **Subject** 作为 **ServiceAccount**，并提供 **Subject 命名空间**和**名称**。
   a. **主题命名空间**:**openshift-gitops**
   b. **主题名称**:**openshift-gitops-argocd-application-controller**
10. 点 **Create**。**ClusterRoleBinding** 对象的 YAML 文件如下：

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

```
apiGroup: rbac.authorization.k8s.io
kind: ClusterRole
name: secrets-cluster-role
```

1.14. 使用 RED HAT OPENSHIFT GITOPS 安装 OLM OPERATOR

带有集群配置的 Red Hat OpenShift GitOps 管理特定集群范围的资源，并负责安装集群 Operator 或任何命名空间范围的 OLM Operator。

考虑作为集群管理员的情况，您必须安装 OLM Operator，如 Tekton。您可以使用 OpenShift Container Platform Web 控制台手动安装 Tekton Operator 或 OpenShift CLI，在集群中手动安装 Tekton 订阅和 Tekton Operator 组。

Red Hat OpenShift GitOps 将 Kubernetes 资源放置在 Git 存储库中。作为集群管理员，使用 Red Hat OpenShift GitOps 管理和自动化其他 OLM Operator 安装，而无需手动步骤。例如，在使用 Red Hat OpenShift GitOps 将 Tekton 订阅放在 Git 仓库后，Red Hat OpenShift GitOps 会自动从 Git 仓库获取此 Tekton 订阅，并在集群中安装 Tekton Operator。

1.14.1. 安装集群范围的 Operator

Operator Lifecycle Manager (OLM) 为集群范围的 Operator 使用 **openshift-operators** 命名空间中的默认 **global-operators** Operator 组。因此，您不必在 Gitops 仓库中管理 **OperatorGroup** 资源。但是，对于命名空间范围的 Operator，您必须管理该命名空间中的 **OperatorGroup** 资源。

要安装集群范围的 Operator，请在 Git 仓库中创建并放置所需 Operator 的 **Subscription** 资源。

示例：Grafana Operator 订阅

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: grafana
spec:
  channel: v4
  installPlanApproval: Automatic
  name: grafana-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
```

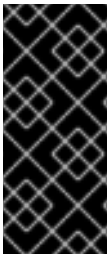
1.14.2. 安装 namepace-scoped Operator

要安装命名空间范围的 Operator，请在 Git 仓库中创建并放置所需 Operator 的 **Subscription** 和 **OperatorGroup** 资源。

示例：Ansible Automation Platform Resource Operator

```
# ...
apiVersion: v1
kind: Namespace
metadata:
  labels:
    openshift.io/cluster-monitoring: "true"
  name: ansible-automation-platform
# ...
```

```
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: ansible-automation-platform-operator
  namespace: ansible-automation-platform
spec:
  targetNamespaces:
    - ansible-automation-platform
# ...
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: ansible-automation-platform
  namespace: ansible-automation-platform
spec:
  channel: patch-me
  installPlanApproval: Automatic
  name: ansible-automation-platform-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
# ...
```



重要

当使用 Red Hat OpenShift GitOps 部署多个 Operator 时，您必须在对应的命名空间中创建单个 Operator 组。如果一个命名空间中存在多个 Operator 组，则在该命名空间中创建的任何 CSV 都会变为带有 **TooManyOperatorGroups** 原因的 **failure** 状态。在相应命名空间中的 Operator 组数量达到一个后，所有以前有 **failure** 状态的 CSV 都会过渡到 **pending** 状态。您需要手动批准待处理的安装计划以完成 Operator 安装。

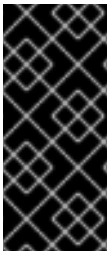
1.15. 其他资源

- 安装 GitOps CLI
- 基本 GitOps argocd 命令

第 2 章 在 ARGO CD APPLICATION CONTROLLER 副本间分片集群

如果控制器管理太多集群并使用太多内存，您可以在多个 Argo CD Application Controller 副本间分片集群。

2.1. 启用循环分片算法



重要

循环 分片算法只是一个技术预览功能。技术预览功能不受红帽产品服务等级协议（SLA）支持，且功能可能并不完整。红帽不推荐在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

有关红帽技术预览功能支持范围的更多信息，请参阅技术预览功能支持范围。

默认情况下，Argo CD Application Controller 使用非统一 **旧的** 基于哈希的分片算法将集群分配给分片。这可能导致集群分布不均匀。您可以启用 **循环** 分片算法，以便在所有分片间实现更相同的集群分布。

在 Red Hat OpenShift GitOps 中使用 **round-robin** 分片算法有以下优点：

- 确保更多平衡的工作负载分布
- 防止分片超载或使用率不足
- 优化计算资源的效率
- 降低瓶颈风险
- 提高 Argo CD 系统的整体性能和可靠性

引入其他分片算法允许根据特定用例进一步自定义。您可以选择最适合与部署需求匹配的算法，从而提高灵活性和适应性在不同操作场景中。

提示

要利用 GitOps 中替代分片算法的好处，在部署期间启用分片至关重要。

2.1.1. 在 web 控制台中启用循环分片算法

您可以使用 OpenShift Container Platform Web 控制台启用 **循环** 分片算法。

先决条件

- 您已在 OpenShift Container Platform 集群中安装了 Red Hat OpenShift GitOps Operator。
- 访问 OpenShift Container Platform web 控制台。
- 您可以使用 **cluster-admin** 权限访问集群。

流程

1. 在 Web 控制台的 **Administrator** 视角中，进入 **Operators → Installed Operators**。

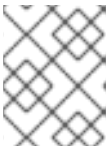
2. 从已安装的 Operator 点 **Red Hat OpenShift GitOps**，再进入 **Argo CD** 选项卡。
3. 点击您要启用 **循环** 分片算法的 Argo CD 实例，如 **openshift-gitops**。
4. 点击 **YAML** 选项卡并编辑 YAML 文件，如下例所示：

启用循环分片算法的 Argo CD 实例示例

```
apiVersion: argoproj.io/v1beta1
kind: ArgoCD
metadata:
  name: openshift-gitops
  namespace: openshift-gitops
spec:
  controller:
    sharding:
      enabled: true 1
      replicas: 3 2
    env: 3
      - name: ARGOCD_CONTROLLER_SHARDING_ALGORITHM
        value: round-robin
    logLevel: debug 4
```

1. 将 **shard.enabled** 参数设置为 **true** 以启用分片。
2. 将副本数量设置为所需的值，例如 **3**。
3. 将分片算法设置为 **round-robin**。
4. 将日志级别设置为 **debug**，以便您可以验证每个集群附加的分片。

5. 点击 **Save**。
此时会出现一个成功通知警报 **openshift-gitops 更新至 <version>** 版本。



注意

如果您编辑了默认的 **openshift-gitops** 实例，则会显示 **Managed resource** 对话框。再次单击 **Save** 以确认更改。

6. 通过执行以下步骤验证分片是否以 **round-robin** 作为分片算法启用：
   a. 进入 **Workloads → StatefulSets**。
   b. 从 **Project** 下拉列表中选择安装 Argo CD 实例的命名空间。
   c. 点 **<instance_name>-application-controller**，如 **openshift-gitops-application-controller**，然后进入 **Pods** 选项卡。
   d. 观察创建的应用控制器 pod 的数量。它应该与集合副本数对应。
   e. 点您要检查的控制器 pod，并进入 **Logs** 选项卡来查看 pod 日志。

控制器 pod 日志片断示例

```
time="2023-12-13T09:05:34Z" level=info msg="ArgoCD Application Controller is starting"
built="2023-12-01T19:21:49Z" commit=a3vd5c3df52943a6fff6c0rg181fth3248976299
namespace=openshift-gitops version=v2.9.2+c5ea5c4
time="2023-12-13T09:05:34Z" level=info msg="Processing clusters from shard 1"
time="2023-12-13T09:05:34Z" level=info msg="Using filter function:  round-robin" (1)
time="2023-12-13T09:05:34Z" level=info msg="Using filter function:  round-robin"
time="2023-12-13T09:05:34Z" level=info msg="appResyncPeriod=3m0s,
appHardResyncPeriod=0s"
```

(1) 查找 **"Using filter function: round-robin"** 信息。

f. 在日志搜索字段中，搜索 **分片处理** 以验证跨分片的集群分布是否正常，如下例所示。

重要

确保将日志级别设置为 debug 以观察这些日志。

控制器 pod 日志片断示例

```
time="2023-12-13T09:05:34Z" level=debug msg="ClustersList has 3 items"
time="2023-12-13T09:05:34Z" level=debug msg="Adding cluster with id= and
name=in-cluster to cluster's map"
time="2023-12-13T09:05:34Z" level=debug msg="Adding cluster with id=068d8b26-
6rhi-4w23-jrf6-wjjfyw833n23 and name=in-cluster2 to cluster's map"
time="2023-12-13T09:05:34Z" level=debug msg="Adding cluster with id=836d8b53-
96k4-f68r-8wq0-sh72j22kl90w and name=in-cluster3 to cluster's map"
time="2023-12-13T09:05:34Z" level=debug msg="Cluster with id= will be
processed by shard 0" (1)
time="2023-12-13T09:05:34Z" level=debug msg="Cluster with id=068d8b26-6rhi-
4w23-jrf6-wjjfyw833n23 will be processed by shard 1" (2)
time="2023-12-13T09:05:34Z" level=debug msg="Cluster with id=836d8b53-96k4-
f68r-8wq0-sh72j22kl90w will be processed by shard 2" (3)
```

(1) (2) (3) **在本例中，3 个集群会持续附加到分片 0、分片 1和 shard 2。**

注意

如果集群"C"数量是分片副本"R"的数量，则每个分片必须具有相同数量的分配集群 "N"，这等同于"C"除以 "R"。以上示例显示了 3 个集群和 3 个副本，因此每个分片都分配有 1 个集群。

2.1.2. 使用 CLI 启用循环分片算法

您可以使用命令行界面启用 循环 分片算法。

先决条件

- **您已在 OpenShift Container Platform 集群中安装了 Red Hat OpenShift GitOps Operator。**
- **您可以使用 cluster-admin 权限访问集群。**

流程

1. 运行以下命令，启用分片并将副本数设置为所需的值：

```
$ oc patch argocd <argocd_instance> -n <namespace> --patch='{"spec":{"controller":
{"sharding":{"enabled":true,"replicas":<value>}}}}' --type=merge
```

输出示例

```
argocd.argoproj.io/<argocd_instance> patched
```

2. 运行以下命令，将分片算法配置为 循环 ：

```
$ oc patch argocd <argocd_instance> -n <namespace> --patch='{"spec":{"controller":
{"env":[{"name":"ARGOCD_CONTROLLER_SHARDING_ALGORITHM","value":"round-
robin"}]}}}' --type=merge
```

输出示例

```
argocd.argoproj.io/<argocd_instance> patched
```

3. 运行以下命令，验证 Argo CD Application Controller pod 的数量是否与设置副本数对应：

```
$ oc get pods -l app.kubernetes.io/name=<argocd_instance>-application-controller -n
<namespace>
```

输出示例

```
NAME                                  READY   STATUS    RESTARTS   AGE
<argocd_instance>-application-controller-0   1/1     Running   0          11s
<argocd_instance>-application-controller-1   1/1     Running   0          32s
<argocd_instance>-application-controller-2   1/1     Running   0          22s
```

4. 运行以下命令，使用 round-robin 作为分片算法验证分片是否已启用：

```
$ oc logs <argocd_application_controller_pod> -n <namespace>
```

输出片断示例

```
time="2023-12-13T09:05:34Z" level=info msg="ArgoCD Application Controller is
starting" built="2023-12-01T19:21:49Z"
commit=a3vd5c3df52943a6fff6c0rg181fth3248976299 namespace=<namespace>
version=v2.9.2+c5ea5c4
time="2023-12-13T09:05:34Z" level=info msg="Processing clusters from shard 1"
time="2023-12-13T09:05:34Z" level=info msg="Using filter function:  round-robin" (1)
time="2023-12-13T09:05:34Z" level=info msg="Using filter function:  round-robin"
time="2023-12-13T09:05:34Z" level=info msg="appResyncPeriod=3m0s,
appHardResyncPeriod=0s"
```

(1) 查找 "Using filter function: round-robin" 信息。

5. 通过执行以下步骤验证跨分片的集群分布是否：

   a. 运行以下命令，将日志级别设置为 debug ：

   ```
   $ oc patch argocd <argocd_instance> -n <namespace> --patch='{"spec":
   {"controller":{"logLevel":"debug"}}}' --type=merge
   ```

   输出示例

   ```
   argocd.argoproj.io/<argocd_instance> patched
   ```

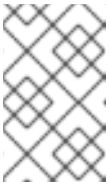
   b. 运行以下命令，查看由分片处理 的日志，以观察每个集群附加到的分片 ：

   ```
   $ oc logs <argocd_application_controller_pod> -n <namespace> | grep
   "processed by shard"
   ```

   输出片断示例

   ```
   time="2023-12-13T09:05:34Z" level=debug msg="Cluster with id= will be
   processed by shard 0" 1
   time="2023-12-13T09:05:34Z" level=debug msg="Cluster with id=068d8b26-6rhi-
   4w23-jrf6-wjjfyw833n23 will be processed by shard 1" 2
   time="2023-12-13T09:05:34Z" level=debug msg="Cluster with id=836d8b53-96k4-
   f68r-8wq0-sh72j22kl90w will be processed by shard 2" 3
   ```

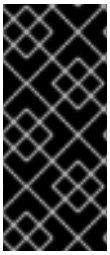
1 2 3 在本例中，3 个集群会持续附加到分片 0、分片 1和 shard 2。



注意

如果集群"C"数量是分片副本"R"的数量，则每个分片必须具有相同数量的分配集群 "N"，这等同于"C"除以 "R"。以上示例显示了 3 个集群和 3 个副本，因此每个分片都分配有 1 个集群。

2.2. 启用 ARGO CD APPLICATION CONTROLLER 的动态扩展



重要

分片的动态扩展只是一个技术预览功能。技术预览功能不受红帽产品服务等级协议（SLA）支持，且功能可能并不完整。红帽不推荐在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

有关红帽技术预览功能支持范围的更多信息，请参阅技术预览功能支持范围。

默认情况下，Argo CD Application Controller 无限分配集群到分片。如果您使用 循环 分片算法，这个静态分配可能会导致分片分布不均匀，特别是在添加或删除副本时。您可以启用动态扩展分片，根据给定时间由 Argo CD Application Controller 管理的集群数量自动调整分片数量。这样可确保分片良好平衡，并优化计算资源的使用。

注意

启用动态扩展后，您无法手动修改分片计数。系统会根据给定时间由 Argo CD Application Controller 管理的集群数量自动调整分片数量。

2.2.1. 在 web 控制台中启用动态扩展分片

您可以使用 OpenShift Container Platform Web 控制台启用动态扩展分片。

先决条件

- 您可以使用 cluster-admin 权限访问集群。
- 访问 OpenShift Container Platform web 控制台。
- 您已在 OpenShift Container Platform 集群中安装了 Red Hat OpenShift GitOps Operator。

流程

1. 在 OpenShift Container Platform Web 控制台的 Administator 视角中，进入 Operators → Installed Operators。
2. 在 Installed Operators 列表中，选择 Red Hat OpenShift GitOps Operator，然后点ArgoCD 选项卡。
3. 选择您要为其启用动态扩展分片的 Argo CD 实例名称，如 openshift-gitops。
4. 点 YAML 选项卡，然后编辑并配置 spec.controller.sharding 属性，如下所示：

   启用动态扩展的 Argo CD YAML 文件示例

```
apiVersion: argoproj.io/v1beta1
kind: ArgoCD
metadata:
  name: openshift-gitops
  namespace: openshift-gitops
spec:
  controller:
    sharding:
      dynamicScalingEnabled: true (1)
      minShards: 1 (2)
      maxShards: 3 (3)
      clustersPerShard: 1 (4)
```

   (1) 将 dynamicScalingEnabled 设置为 true 以启用动态扩展。

   (2) 将 minShards 设置为您要具有的最小分片数量。该值必须设置为 1 或更高。

   (3) 将 maxShards 设置为您要具有的最大分片数。该值必须大于 minShards 的值。

   (4) 将 clusterPerShard 设置为您要为每个分片具有的集群数量。该值必须设置为 1 或更高。

5. 点击 Save。
   此时会出现一个成功通知警报 openshift-gitops 更新至 <version> 版本。

注意

如果您编辑了默认的 openshift-gitops 实例，则会显示 Managed resource 对话框。再次单击 Save 以确认更改。

验证

通过检查命名空间中的 pod 数量来验证分片是否已启用：

1. **进入 Workloads → StatefulSets。**
2. **从 Project 下拉列表中选择部署 Argo CD 实例的命名空间，如openshift-gitops。**
3. **点 Argo CD 实例名称的 StatefulSet 对象的名称，如 openshift-gitops-apllication-controller。**
4. **点 Pods 选项卡，然后验证 pod 的数量等于或大于您在 Argo CDYAML 文件中设置的 minShards 值。**

2.2.2. 使用 CLI 启用动态扩展分片

您可以使用 OpenShift CLI (oc)启用动态扩展分片。

先决条件

- **您已在 OpenShift Container Platform 集群中安装了 Red Hat OpenShift GitOps Operator。**
- **您可以使用 cluster-admin 权限访问集群。**

流程

1. **以具有 cluster-admin 权限的用户身份登录 oc 工具。**
2. **运行以下命令来启用动态扩展：**

```
$ oc patch argocd <argocd_instance> -n <namespace> --type=merge --patch='{"spec":
{"controller":{"sharding":{"dynamicScalingEnabled":true,"minShards":
<value>,"maxShards":<value>,"clustersPerShard":<value>}}}}'
```

示例命令

```
$ oc patch argocd openshift-gitops -n openshift-gitops --type=merge --patch='{"spec":
{"controller":{"sharding":
{"dynamicScalingEnabled":true,"minShards":1,"maxShards":3,"clustersPerShard":1}}}
}' (1)
```

(1) **示例命令为 openshift-gitops 命名空间中的 openshift-gitops Argo CD 实例启用动态扩展，并将最小分片数量设置为 1，将分片的最大数量设置为 3，以及每个分片的集群数量 1。minShard 和 clusterPerShard 的值必须设置为 1 或更高。maxShard 的值必须等于或大于 minShard 的值。**

输出示例

```
argocd.argoproj.io/openshift-gitops patched
```

验证

1. 检查 Argo CD 实例的 **spec.controller.sharding** 属性：

```
$ oc get argocd <argocd_instance> -n <namespace> -o
jsonpath='{.spec.controller.sharding}'
```

示例命令

```
$ oc get argocd openshift-gitops -n openshift-gitops -o
jsonpath='{.spec.controller.sharding}'
```

启用分片动态扩展时的输出示例

```
{"dynamicScalingEnabled":true,"minShards":1,"maxShards":3,"clustersPerShard":1}
```

2. **可选：通过检查 OpenShift Container Platform web 控制台中的 Argo CD 实例的配置 YAML 文件中的 spec.controller.sharding 属性来验证是否启用了动态扩展。**

3. 检查 Argo CD Application Controller pod 的数量：

```
$ oc get pods -n <namespace> -l app.kubernetes.io/name=<argocd_instance>-
application-controller
```

示例命令

```
$ oc get pods -n openshift-gitops -l app.kubernetes.io/name=openshift-gitops-
application-controller
```

输出示例

```
NAME                                  READY   STATUS    RESTARTS   AGE
openshift-gitops-application-controller-0     1/1     Running   0          2m  1
```

1 Argo CD Application Controller pod 的数量必须等于或大于 minShard 的值。

2.2.3. 其他资源

- Argo CD 自定义资源属性
- 使用 pod 横向自动扩展自动扩展 pod