Red Hat OpenShift GitOps 1.13

宣言型クラスター設定

OpenShift GitOps を使用してクラスター設定で OpenShift クラスターを設定し、GitOps CLI を使用してデフォルトモードとコードモードでアプリケーションを作成および同期する

Last Updated: 2024-08-22

Red Hat OpenShift GitOps 1.13 宣言型クラスター設定

OpenShift GitOps を使用してクラスター設定で OpenShift クラスターを設定し、GitOps CLI を使用してデフォルトモードとコードモードでアプリケーションを作成および同期する

法律上の通知

概要

このドキュメントでは、クラスターのカスタム設定を含むアプリケーションと Git ディレクトリーのコンテンツを再帰的に同期するように Argo CD を設定する手順を説明します。また、GitOps CLI を使用して、デフォルトモードとコードモードでアプリケーションを作成および同期する方法も説明します。

目次

第1章 クラスター設定を使用したアプリケーションのデプロイによる OPENSHIFT クラスターの設定

Red Hat OpenShift GitOps では、Argo CD を、クラスターのカスタム設定が含まれるアプリケーションと Git ディレクトリーの内容を再帰的に同期するように設定することができます。

1.1. 前提条件

- OpenShift Container Platform クラスターに管理者としてログインしている。
- Red Hat OpenShift GitOps Operator が OpenShift Container Platform クラスターにインストールされている。

1.2. ARGO CD インスタンスを使用してクラスタースコープのリソースの管理

クラスタースコープのリソースを管理するには、Red Hat OpenShift GitOps Operator の既存の **Subscription** オブジェクトを更新し、Argo CD インスタンスの名前空間を **spec** セクションの **ARGOCD_CLUSTER_CONFIG_NAMESPACES** 環境変数に追加します。

手順

1. Web コンソールの **Administrator** パースペクティブで、**Operators → Installed Operators → Red Hat OpenShift GitOps → Subscription** に移動します。
2. **Actions** リストをクリックし、**Edit Subscription** をクリックします。
3. **openshift-gitops-operator** サブスクリプションの詳細ページの **YAML** タブで、Argo CD インスタンスの namespace を仕様セクションの **ARGOCD_CLUSTER_CONFIG_NAMESPACES** 環境変数に追加して、**spec** セクションの **Subscription** YAML ファイルを編集します。

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

4. **Save**、**Reload** を順にクリックします。
5. Argo CD インスタンスがクラスタースコープのリソースを管理するクラスターロールで設定されていることを確認するには、次の手順を実行します。
   a. **User Management → Roles** に移動し、**Filter** リストから **Cluster-wide Roles** を選択します。
   b. **Search by name** フィールドを使用して、**argocd-application-controller** を検索します。**Roles** ページには、作成されたクラスターロールが表示されます。

ヒント

あるいは、OpenShift CLI で次のコマンドを実行します。

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

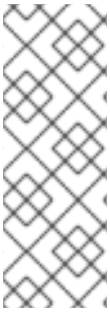
出力 **yes** は、Argo インスタンスがクラスタースコープのリソースを管理するクラスターロールで設定されていることを確認します。それ以外の場合は、設定を確認し、必要に応じて必要な手順を実行します。

1.3. ARGO CD インスタンスのデフォルトの権限

デフォルトでは、Argo CD インスタンスには次の権限があります。

- Argo CD インスタンスには、それがデプロイされている namespace 内のリソースのみを管理する **admin** 権限があります。たとえば、**foo** namespace にデプロイされた Argo CD インスタンスには、その namespace に対してのみリソースを管理する **admin** 権限があります。
- Argo CD が適切に機能するには、リソースに対するクラスター全体の **read** 権限が必要であるため、Argo CD には次のクラスタースコープのアクセス許可があります。

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



注記

- Argo CD が実行している **argocd-server** と **argocd-application-controller** コンポーネントで使用されるクラスターのロールを編集して、**write** 権限が Argo CD で管理したい namespace とリソースのみに制限できます。

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

1.4. クラスターレベルでの ARGO CD インスタンスの実行

Red Hat OpenShift GitOps Operator によってインストールされるデフォルトの Argo CD インスタンスおよび付随するコントローラーは、単純な設定の切り替えを設定して、クラスターのインフラストラクチャーノードで実行できるようになりました。

手順

1. 既存のノードにラベルを付けます。

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

2. オプション: 必要な場合は、テイントを適用し、インフラストラクチャーノードでワークロードを分離し、他のワークロードがそれらのノードでスケジュールされないようにすることもできます。

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

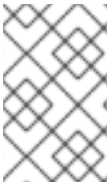
3. **GitOpsService** カスタムリソースに **runOnInfra** トグルを追加します。

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

4. オプション: テイントがノードに追加された場合は、**tolerations** を **GitOpsService** カスタムリソースに追加します。以下に例を示します。

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

5. コンソール UI の Pod を **Pods → Pod details** で表示して、**openshift-gitops** namespace のワークロードがインフラストラクチャーノードでスケジュールされていることを確認します。



注記

デフォルトの Argo CD カスタムリソースに手動で追加された **nodeSelectors** および **tolerations** は、**GitOpsService** カスタムリソースのトグルおよび **tolerations** によって上書きされます。

関連情報

- テイントと容認の詳細は、ノードテイントを使用した Pod 配置の制御 を参照してください。
- インフラストラクチャーマシンセットの詳細は、インフラストラクチャーマシンセットの作成を参照してください。

1.5. ARGO CD ダッシュボードを使用したアプリケーションの作成

Argo CD は、アプリケーションを作成できるダッシュボードを提供します。

このサンプルワークフローでは、**cluster** ディレクトリーの内容を **cluster-configs** アプリケーションに対して再帰的に同期するように Argo CD を設定するプロセスを説明します。このディレクトリーは、Web コンソールの  メニューに **Red Hat Developer Blog - Kubernetes**へのリンクを追加する OpenShift Container Platform Web コンソールクラスター設定を定義します。また、クラスターの namespace **spring-petclinic** を定義します。

前提条件

- OpenShift Container Platform クラスターに管理者としてログインしている。
- Red Hat OpenShift GitOps Operator が OpenShift Container Platform クラスターにインストールされている。
- Argo CD インスタンスにログインしている。

手順

1. Argo CD ダッシュボードで、**NEW APP** をクリックして新規の Argo CD アプリケーションを追加します。
2. このワークフローでは、以下の設定で **cluster-configs** アプリケーションを作成します。

   アプリケーション名
   cluster-configs
   プロジェクト
   default
   同期ポリシー
   Manual
   リポジトリー URL
   https://github.com/redhat-developer/openshift-gitops-getting-started
   リビジョン
   HEAD
   パス
   cluster
   宛先
   https://kubernetes.default.svc
   namespace
   spring-petclinic
   ディレクトリーの再帰処理
   checked

3. **CREATE** をクリックしてアプリケーションを作成します。
4. Web コンソールの **Administrator** パースペクティブを開き、**Administration** → **Namespaces** を展開します。
5. namespace を検索、選択してから **Label** フィールドに **argocd.argoproj.io/managed-by=openshift-gitops** を入力し、**openshift-gitops** namespace にある Argo CD インスタンスが namespace を管理できるようにします。

1.6. oc ツールを使用したアプリケーションの作成

oc ツールを使用して、ターミナルで Argo CD アプリケーションを作成できます。

前提条件

- Red Hat OpenShift GitOps Operator が OpenShift Container Platform クラスターにインストールされている。
- Argo CD インスタンスにログインしている。

手順

1. サンプルアプリケーション をダウンロードします。

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

2. アプリケーションを作成します。

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

3. **oc get** コマンドを実行して、作成されたアプリケーションを確認します。

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

4. アプリケーションがデプロイされている namespace にラベルを追加し、**openshift-gitops** namespace の Argo CD インスタンスが管理できるようにします。

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

1.7. GITOPS CLI を使用したデフォルトモードでのアプリケーションの作成

GitOps **argocd** CLI を使用して、default モードでアプリケーションを作成できます。

このサンプルワークフローでは、**cluster** ディレクトリーの内容を **cluster-configs** アプリケーションに対して再帰的に同期するように Argo CD を設定するプロセスを説明します。このディレクトリーは、OpenShift Container Platform クラスター設定とクラスター上の **spring-petclinic** namespace を定義します。

前提条件

- Red Hat OpenShift GitOps Operator が OpenShift Container Platform クラスターにインストールされている。
- OpenShift CLI (**oc**) がインストールされている。
- Red Hat OpenShift GitOps **argocd** CLI がインストールされている。
- Argo CD インスタンスにログインしている。

手順

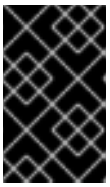
1. Argo CD サーバーの **admin** アカウントのパスワードを取得します。

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

2. Argo CD サーバーの URL を取得します。

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

3. **admin** アカウントのパスワードを使用して Argo CD サーバーにログインし、一重引用符で囲みます。



重要

パスワードを一重引用符で囲むと、**$** などの特殊文字がシェルによって誤って解釈されなくなります。パスワードのリテラル値を囲むには常に一重引用符を使用してください。

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

例

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

4. すべてのアプリケーションを表示して、**argocd** コマンドをデフォルトモードで実行できることを確認します。

```
$ argocd app list
```

設定が正しい場合は、既存のアプリケーションが次のヘッダーとともにリストされます。

出力例

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

5. デフォルトモードでアプリケーションを作成します。

```
$ 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. **openshif-gitops** Argo CD インスタンスが管理する **spring-petclinic** 宛先 namespace にラベルを付けます。

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

7. 使用可能なアプリケーションをリスト表示して、アプリケーションが正常に作成されたことを確認します。

```
$ argocd app list
```

cluster-configs の Argo CD アプリケーションは **Healthy** ステータスであっても、sync ポリシーが **none** であるため、自動的に同期されず、**OutOfSync** ステータスのままになります。

1.8. GITOPS CLI を使用したコアモードでのアプリケーションの作成

GitOps **argocd** CLI を使用して、**core** モードでアプリケーションを作成できます。

このサンプルワークフローでは、**cluster** ディレクトリーの内容を **cluster-configs** アプリケーションに対して再帰的に同期するように Argo CD を設定するプロセスを説明します。このディレクトリーは、OpenShift Container Platform クラスター設定とクラスター上の **spring-petclinic** namespace を定義します。

前提条件

- Red Hat OpenShift GitOps Operator が OpenShift Container Platform クラスターにインストールされている。
- OpenShift CLI (**oc**) がインストールされている。
- Red Hat OpenShift GitOps **argocd** CLI がインストールされている。

手順

1. **oc** CLI ツールを使用して OpenShift Container Platform クラスターにログインします。

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

例

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

2. コンテキストが **kubeconfig** ファイルで正しく設定されているかどうかを確認します。

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

3. 現在のコンテキストのデフォルトの namespace を **openshift-gitops** に設定します。

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

4. 次の環境変数を設定して、Argo CD コンポーネント名をオーバーライドします。

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

5. すべてのアプリケーションを一覧表示して、**core** モードで **argocd** コマンドを実行できることを確認します。

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

設定が正しい場合は、既存のアプリケーションが次のヘッダーとともにリストされます。

出力例

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

6. **core** モードでアプリケーションを作成します。

```
$ 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. **openshif-gitops** Argo CD インスタンスが管理する **spring-petclinic** 宛先 namespace にラベルを付けます。

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

8. 使用可能なアプリケーションをリスト表示して、アプリケーションが正常に作成されたことを確認します。

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

cluster-configs の Argo CD アプリケーションは **Healthy** ステータスであっても、sync ポリシーが **none** であるため、自動的に同期されず、**OutOfSync** ステータスのままになります。

1.9. アプリケーションの GIT リポジトリーとの同期

Argo CD の同期ポリシーを変更することで、アプリケーションを Git リポジトリーと同期できます。ポリシーを変更すると、クラスター設定の変更が Git リポジトリーからクラスターに自動的に適用されます。

手順

1. Argo CD ダッシュボードでは、**cluster-configs** Argo CD アプリケーションに **Missing** および **OutOfSync** のステータスがあることに注意してください。アプリケーションは手動の同期ポリシーで設定されているため、Argo CD はこれを自動的に同期しません。

2. **cluster-configs** タイルの **同期** をクリックし、変更を確認してから、**SYNCHRONIZE** をクリックします。Argo CD は Git リポジトリーの変更を自動的に検出します。設定が変更されると、Argo CD は **cluster-configs** のステータスを **OutOfSync** に変更します。Argo CD の同期ポリシーを変更し、Git リポジトリーからクラスターに変更を自動的に適用できるようにします。

3. **cluster-configs** Argo CD アプリケーションに **Healthy** および **Synced** のステータスがあることに注意してください。**cluster-configs** タイルをクリックし、クラスター上で同期されたリソースおよびそれらのステータスの詳細を確認します。

4. OpenShift Container Platform Web コンソールに移動し、  をクリックして **Red Hat Developer Blog - Kubernetes** へのリンクが表示されることを確認します。

5. **Project** ページに移動し、**spring-petclinic** namespace を検索し、これがクラスターに追加されていることを確認します。
クラスター設定がクラスターに正常に同期されます。

1.10. GITOPS CLI を使用したデフォルトモードでのアプリケーションの同期

GitOps **argocd** CLI を使用して、デフォルトモードでアプリケーションを同期できます。

このサンプルワークフローでは、**cluster** ディレクトリーの内容を **cluster-configs** アプリケーションに対して再帰的に同期するように Argo CD を設定するプロセスを説明します。このディレクトリーは、OpenShift Container Platform クラスター設定とクラスター上の **spring-petclinic** namespace を定義します。

前提条件

- Red Hat OpenShift GitOps Operator が OpenShift Container Platform クラスターにインストールされている。
- Argo CD インスタンスにログインしている。
- OpenShift CLI (**oc**) がインストールされている。
- Red Hat OpenShift GitOps **argocd** CLI がインストールされている。

手順

1. Argo CD サーバーの **admin** アカウントのパスワードを取得します。

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

2. Argo CD サーバーの URL を取得します。

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

3. **admin** アカウントのパスワードを使用して Argo CD サーバーにログインし、一重引用符で囲みます。

重要

パスワードを一重引用符で囲むと、**$** などの特殊文字がシェルによって誤って解釈されなくなります。パスワードのリテラル値を囲むには常に一重引用符を使用してください。

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

例

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

4. アプリケーションは **none** sync ポリシーを使用して設定されているため、同期操作を手動でトリガーする必要があります。

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

5. アプリケーションをリスト表示して、アプリケーションのステータスが **Healthy** および **Synced** であることを確認します。

```
$ argocd app list
```

1.11. GITOPS CLI を使用したコアモードでのアプリケーションの同期

GitOps **argocd** CLI を使用して、**core** モードでアプリケーションを同期できます。

このサンプルワークフローでは、**cluster** ディレクトリーの内容を **cluster-configs** アプリケーションに対して再帰的に同期するように Argo CD を設定するプロセスを説明します。このディレクトリーは、OpenShift Container Platform クラスター設定とクラスター上の **spring-petclinic** namespace を定義します。

前提条件

- Red Hat OpenShift GitOps Operator が OpenShift Container Platform クラスターにインストールされている。
- OpenShift CLI (**oc**) がインストールされている。
- Red Hat OpenShift GitOps **argocd** CLI がインストールされている。

手順

1. **oc** CLI ツールを使用して OpenShift Container Platform クラスターにログインします。

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

例

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

2. コンテキストが **kubeconfig** ファイルで正しく設定されているかどうかを確認します。

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

3. 現在のコンテキストのデフォルトの namespace を **openshift-gitops** に設定します。

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

4. 次の環境変数を設定して、Argo CD コンポーネント名をオーバーライドします。

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

5. アプリケーションは **none** sync ポリシーを使用して設定されているため、同期操作を手動でトリガーする必要があります。

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

6. アプリケーションをリスト表示して、アプリケーションのステータスが **Healthy** および **Synced** であることを確認します。

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

1.12. クラスター設定用の組み込みのアクセス許可

デフォルトでは、Argo CD インスタンスには、クラスター Operator、オプションの OLM オペレーター、およびユーザー管理など、特定のクラスタースコープのリソースを管理する権限があります。

注記

Argo CD にはクラスター管理者権限がありません。

Argo CD インスタンスのパーミッション:

リソース	説明
リソースグループ	ユーザーまたは管理者の設定
operators.coreos.com	OLM によって管理されるオプションの Operator
user.openshift.io , **rbac.authorization.k8s.io**	グループ、ユーザー、およびそれらの権限
config.openshift.io	クラスター全体のビルド設定、レジストリー設定、およびスケジューラーポリシーを設定するために使用される CVO によって管理されるコントロールプレーン Operator
storage.k8s.io	ストレージ
console.openshift.io	コンソールのカスタマイズ

1.13. クラスター設定のアクセス許可を追加する

Argo CD インスタンスにアクセス許可を付与して、クラスター設定を管理できます。追加のアクセス許可を持つクラスターロールを作成し、新しいクラスターロールバインディングを作成して、クラスターロールをサービスアカウントに関連付けます。

前提条件

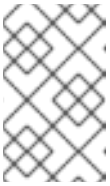
- **cluster-admin** 権限で OpenShift Container Platform クラスターにアクセスでき、Web コンソールにログインしている。
- Red Hat OpenShift GitOps Operator がクラスターにインストールされている。

手順

1. Web コンソールで、**User Management → Roles → Create Role** を選択します。以下の **ClusterRole** YAML テンプレートを使用してルールを追加し、追加のパーミッションを指定します。

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

2. **Create** をクリックしてクラスターロールを追加します。
3. クラスターのロールバインディングを作成するには、**User Management → Role Bindings → Create Binding** を選択します。
4. **Project** リストから **All Projects** を選択します。
5. **Create binding** をクリックします。
6. **Binding type** を **Cluster-wide role binding (ClusterRoleBinding)** として選択します。
7. **RoleBinding name** の一意の値を入力します。
8. ドロップダウンリストから、新しく作成したクラスターロールまたは既存のクラスターロールを選択します。
9. **Subject** を **ServiceAccount** として選択し、**サブジェクトの namespace** と **名前** を指定します。
   a. **サブジェクトの namespace**: **openshift-gitops**
   b. **サブジェクトの名前**: **openshift-gitops-argocd-application-controller**



注記

Subject name の値は、クラスターロールおよびクラスターロールバインディングを作成する GitOps コントロールプレーンのコンポーネントによって異なります。

10. **Create** をクリックします。**ClusterRoleBinding** オブジェクトの YAML ファイルは以下のとおりです。

```
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
```

関連情報

- クラススコープのインスタンスのユーザー定義のクラスターロールを作成してパーミッションをカスタマイズする

1.14. RED HAT OPENSHIFT GITOPS を使用した OLM OPERATOR のインストール

クラスター設定の Red Hat OpenShift GitOps は、特定のクラススコープのリソースを管理し、クラスター Operator または namespace スコープの OLM Operator のインストールを処理します。

クラスター管理者として、Tekton などの OLM Operator をインストールする必要がある場合を考えてみましょう。OpenShift Container Platform Web コンソールを使用して Tekton Operator を手動でインストールするか、OpenShift CLI を使用して Tekton サブスクリプションと Tekton Operator グループをクラスターに手動でインストールします。

Red Hat OpenShift GitOps は、Kubernetes リソースを Git リポジトリーに配置します。クラスター管理者は、Red Hat OpenShift GitOps を使用して、手動手順を行わずに他の OLM Operator のインストールを管理および自動化できます。たとえば、Red Hat OpenShift GitOps を使用して Tekton サブスクリプションを Git リポジトリーに配置すると、Red Hat OpenShift GitOps はこの Tekton サブスクリプションを Git リポジトリーから自動的に取得し、クラスターに Tekton Operator をインストールします。

1.14.1. クラススコープの Operator のインストール

Operator Lifecycle Manager (OLM) は、クラススコープの Operator の **openshift-operators** namespace 内のデフォルトの **global-operators** Operator グループを使用します。したがって、Gitops リポジトリーで **OperatorGroup** リソースを管理する必要はありません。ただし、namespace スコープの Operator の場合は、その namespace で **OperatorGroup** リソースを管理する必要があります。

クラススコープの Operator をインストールするには、必要な Operator の **Subscription** リソースを作成し、Git リポジトリーに配置します。

例: Grafana Operator サブスクリプション

```
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. namespace スコープの Operator のインストール

namespace スコープの Operator をインストールするには、必要な Operator の **Subscription** リソースと **OperatorGroup** リソースを作成して Git リポジトリーに配置します。

例: Ansible Automation Platform リソースオペレーター

```
# ...
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

Red Hat OpenShift GitOps を使用して複数の Operator をデプロイする場合は、対応する namespace に Operator グループを 1 つだけ作成する必要があります。1 つの namespace に複数の Operator グループが存在する場合、その namespace で作成された CSV はすべて、**TooManyOperatorGroups** の理由で **failure** 状態に移行します。対応する namespace 内の Operator グループの数が 1 に達すると、以前の **failure** 状態の CSV はすべて **pending** 状態に移行します。Operator のインストールを完了するには、保留中のインストールプランを手動で承認する必要があります。

1.15. 関連情報

- GitOps CLI のインストール

- 基本的な GitOps argocd コマンド

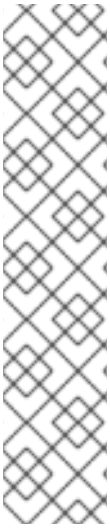
第2章 クラススコープのインスタンスのユーザー定義のクラスターロールを作成してパーミッションをカスタマイズする

デフォルトのクラススコープのインスタンスの場合、Red Hat OpenShift GitOps Operator は特定のクラススコープのリソース管理向けのパーミッションを追加で付与します。したがって、クラスター管理者として、Argo CD をクラススコープのインスタンスとしてデプロイすると、Operator は GitOps コントロールプレーンコンポーネントに対して追加のクラスターロールとクラスターロールバインディングを作成します。これらのクラスターロールとクラスターロールバインディングは、Argo CD がクラスターレベルで動作させるのに必要な追加の権限を提供します。

クラススコープのインスタンスに Operator によって付与されたすべての権限を付与せず、クラスター全体のリソースへの権限を追加または削除する場合は、まずクラススコープのインスタンスのデフォルトのクラスターロールの作成を無効にする必要があります。次に、次のクラススコープのインスタンスの権限をカスタマイズできます。

- デフォルトの ArgoCD インスタンス (デフォルトのクラススコープインスタンス)
- ユーザー定義のクラススコープ Argo CD インスタンス

このガイドでは、ユーザー定義のクラススコープ Argo CD インスタンスを作成し、クラスターのカスタム設定を含む定義済みの namespace に Argo CD アプリケーションをデプロイし、クラススコープインスタンスのデフォルトのクラスターロールの作成を無効にし、GitOps コントロールプレーンコンポーネントの新しいクラスターロールとクラスターロールバインディングを作成してユーザー定義のクラススコープインスタンスの権限をカスタマイズする手順について例を交えて説明します。



注記

開発者として、Argo CD アプリケーションを作成し、クラスター全体のリソースをデプロイする場合は、クラスター管理者がそれらに必要な権限を付与していることを確認してください。

それ以外の場合は、Argo CD の調整後に、アプリケーションの **Status** フィールドに次の例のような認証エラーメッセージが表示されます。

認証エラーメッセージの例

> persistentvolumes is forbidden: User "system:serviceaccount:gitops-demo:argocd-argocd-application-controller" cannot create resource "persistentvolumes" in API group "" at the cluster scope.

2.1. 前提条件

- OpenShift Container Platform クラスターに Red Hat OpenShift GitOps 1.13.0 以降のバージョンがインストールされている。
- OpenShift CLI (**oc**) がインストールされている。
- Red Hat OpenShift GitOps **argocd** CLI がインストールされている。
- クラススコープの Argo CD インスタンス を定義済みの namespace にインストールしている。(例: **spring-petclinic** namespace)。
- ユーザー定義のクラススコープインスタンスが、次のコンポーネントのクラスターロールとクラスターロールバインディングを使用して設定されていることを検証している。

- Argo CD Application Controller
- Argo CD サーバー
- Argo CD ApplicationSet Controller (ApplicationSet Controller 作成される)

- **spring-petclinic** namespace の **customclusterrole** パス を使用して **cluster-configs** Argo CD アプリケーション をデプロイし、**test-gitops-ns** namespace と **test-gitops-pv** 永続ボリュームリソースを作成している。

注記

cluster-configs Argo CD アプリケーションは、次のパラメーターが設定されたユーザー定義のクラススコープインスタンスによって管理される必要があります。

- **selfHeal** フィールドの値が **true** に設定されている
- **syncPolicy** フィールドの値が **automated** に設定されている
- Label フィールドが **app.kubernetes.io/part-of=argocd** の値に設定されている
- 定義した namespace 内の Argo CD インスタンスが namespace を管理できるように、Label フィールドが **argocd.argoproj.io/managed-by=<user_defined_namespace>** 値に設定されている
- **ラベル** フィールドが **app.kubernetes.io/name=<user_defined_argocd_instance>** 値に設定されている

2.2. クラススコープのインスタンスのデフォルトクラスターロール作成の無効化

必要に応じてクラスター全体のリソースへの権限を追加または削除するには、Argo CD カスタムリソース (CR) の YAML ファイルを編集して、クラススコープインスタンスのデフォルトのクラスターロールの作成を無効にする必要があります。

手順

1. Argo CD CR で、**.spec.defaultClusterScopedRoleDisabled** フィールドの値を **true** に設定します。

Argo CD CR の例

```
apiVersion: argoproj.io/v1beta1
kind: ArgoCD
metadata:
  name: example (1)
  namespace: spring-petclinic (2)
# ...
spec:
  defaultClusterScopedRoleDisabled: true (3)
# ...
```

1. クラススコープのインスタンスの名前。
2. クラススコープのインスタンスを実行する namespace。
3. クラススコープインスタンスのデフォルトのクラスターロールの作成を無効にするフラグ値。Operator がクラススコープインスタンスのデフォルトのクラスターロールとクラスターロールバインディングを再作成するようにする場合は、フィールド値を **false** に設定します。

出力例

```
argocd.argoproj.io/example configured
```

2. 次のコマンドを実行して、Red Hat OpenShift GitOps Operator が GitOps コントロールプレーンコンポーネントのデフォルトのクラスターロールとクラスターロールバインディングを削除したことを確認します。

```
$ oc get ClusterRoles/<argocd_name>-<argocd_namespace>-<control_plane_component>
```

```
$ oc get ClusterRoleBindings/<argocd_name>-<argocd_namespace>-
<control_plane_component>
```

出力例

```
No resources found
```

クラススコープのインスタンスのデフォルトのクラスターロールおよびクラスターロールバインディングは作成されません。クラスター管理者は、GitOps コントロールプレーンコンポーネントの新しいクラスターロールとクラスターロールバインディングを作成することで、クラススコープのインスタンスのアクセス許可を作成およびカスタマイズできるようになりました。

関連情報

- ユーザー定義の Argo CD インスタンスのインストール

2.3. クラススコープのインスタンスのパーミッションのカスタマイズ

クラスター管理者は、GitOps コントロールプレーンコンポーネントの新しいクラスターロールとクラスターロールバインディングを作成して、クラススコープのインスタンスの権限をカスタマイズする必要があります。

たとえば、以下の手順ではユーザー定義のクラススコープのインスタンスのみにフォーカスします。

手順

1. Web コンソールの **Administrator** パースペクティブを開き、**User Management → Roles → Create Role** に移動します。
2. 以下の **ClusterRole** YAML テンプレートを使用してルールを追加し、追加のパーミッションを指定します。

クラスターロール YAML テンプレートの例

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: example-spring-petclinic-argocd-application-controller 1
rules:
  - verbs:
      - get
      - list
      - watch
    apiGroups:
      - '*'
    resources:
      - '*'
  - verbs:
      - '*'
    apiGroups:
      - ''
    resources: 2
      - namespaces
      - persistentvolumes
```

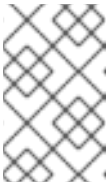
1 <argocd_name>-<argocd_namespace>-<control_plane_component> の命名規則に従ったクラスターロールの名前。

2 クラスターレベルでパーミッションを付与するリソース。

3. **Create** をクリックしてクラスターロールを追加します。
4. 次の手順を実行して、権限をカスタマイズするコントロールプレーンコンポーネントで使用されるサービスアカウントを見つけます。
   a. **Workloads → Pods** に移動します。
   b. **Project** リストから、ユーザー定義のクラススコープのインスタンスがインストールされているプロジェクトを選択します。
   c. コントロールプレーンコンポーネントの Pod をクリックし、**YAML** タブに移動します。
   d. **spec.ServiceAccount** フィールドを見つけ、サービスアカウントをメモします。
5. **User Management → RoleBindings → Create binding** に移動します。
6. **Create binding** をクリックします。
7. **Binding type** を **Cluster-wide role binding (ClusterRoleBinding)** として選択します。
8. <argocd_name>-<argocd_namespace>-<control_plane_component> 命名規則に従って、**RoleBinding name** の一意の値を入力します。
9. **Role name** のドロップダウンリストから新たに作成されたクラスターロールを選択します。
10. **Subject** を **ServiceAccount** として選択し、**サブジェクトの namespace** と **名前** を指定します。

a. **Subject namespace**: **spring-petclinic**

b. **Subject name**: **example-argocd-application-controller**



注記

Subject name については、設定する値が、権限をカスタマイズするコントロールプレーンコンポーネントの **spec.ServiceAccount** フィールドの値と同じであることを確認します。

11. **Create** をクリックします。
コントロールプレーンコンポーネントのサービスアカウントと namespace に必要な権限が作成されました。**ClusterRoleBinding** オブジェクトの YAML ファイルは以下の例のようになります。

クラスターロールバインディングの YAML ファイルの例

```
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: example-spring-petclinic-argocd-application-controller
subjects:
  - kind: ServiceAccount
    name: example-argocd-application-controller
    namespace: spring-petclinic
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: example-spring-petclinic-argocd-application-controller
```

2.4. 関連情報

- クラスター設定のアクセス許可を追加する
- namespace スコープのインスタンスにユーザー定義のクラスターロールを指定して、一般的なクラスターロールを設定する

第3章 ARGO CD アプリケーションコントローラーレプリカ間でのクラスターのシャーディング

コントローラーが管理するクラスターや使用するメモリーが過剰な場合は、複数の Argo CD アプリケーションコントローラーレプリカにわたってクラスターをシャーディングできます。

3.1. ラウンドロビンシャーディングアルゴリズムの有効化

重要

round-robin シャーディングアルゴリズムはテクノロジープレビューのみの機能です。テクノロジープレビュー機能は、Red Hat 製品のサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではないことがあります。Red Hat は、実稼働環境でこれらを使用することを推奨していません。テクノロジープレビューの機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

デフォルトでは、Argo CD アプリケーションコントローラーは、不均一な **legacy** のハッシュベースのシャーディングアルゴリズムを使用してクラスターをシャードに割り当てます。これにより、クラスターの分散が不均等になる可能性があります。**round-robin** シャーディングアルゴリズムを有効にして、すべてのシャードにわたってより均等なクラスター分散を実現できます。

Red Hat OpenShift GitOps で **round-robin** シャーディングアルゴリズムを使用すると、次の利点があります。

- よりバランスの取れたワークロード分散を確保する
- シャードが過負荷になったり、十分に活用されなかったりすることを回避する
- コンピューティングリソースの効率を最適化する
- ボトルネックのリスクを軽減する
- Argo CD システムの全体的なパフォーマンスと信頼性を向上する

代替シャーディングアルゴリズムの導入により、特定の使用例に基づいてさらにカスタマイズできるようになります。デプロイメントのニーズに最も適したアルゴリズムを選択できるため、さまざまな運用シナリオでの柔軟性と適応性が向上します。

ヒント

GitOps で代替シャーディングアルゴリズムの利点を活用するには、デプロイ中にシャーディングを有効にすることが重要です。

3.1.1. Web コンソールでの round-robin シャーディングアルゴリズムの有効化

OpenShift Container Platform Web コンソールを使用して **round-robin** シャーディングアルゴリズムを有効にできます。

前提条件

- Red Hat OpenShift GitOps Operator がクラスターにインストールされている。
- OpenShift Container Platform Web コンソールにアクセスできる。
- **cluster-admin** 権限でクラスターにアクセスできる。

手順

1. Web コンソールの **Administrator** パースペクティブで、**Operators → Installed Operators** に移動します。
2. インストールされている Operator から **Red Hat OpenShift GitOps** をクリックし、**Argo CD** タブに移動します。
3. **round-robin** シャーディングアルゴリズムを有効にする Argo CD インスタンス (例: **openshift-gitops)** をクリックします。
4. **YAML** タブをクリックし、以下の例のように YAML ファイルを編集します。

   ラウンドロビンシャーディングアルゴリズムが有効になっている Argo CD インスタンスの例

```
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. シャーディングを有効にするには、**sharding.enabled** パラメーターを **true** に設定します。
   2. レプリカの数を必要な値 (例: **3)** に設定します。
   3. シャーディングアルゴリズムを **round-robin** に設定します。
   4. 各クラスターがどのシャードに接続されているかを確認できるように、ログレベルを **debug** に設定します。

5. **Save** をクリックします。
   成功通知アラート **openshift-gitops has been updated to version <version>** が表示されます。

注記

デフォルトの **openshift-gitops** インスタンスを編集すると、**Managed resource** ダイアログボックスが表示されます。**Save** をもう一度クリックして、変更を確定します。

6. 次の手順を実行して、シャーディングアルゴリズムとして **round-robin** を使用する設定で、シャーディングが有効になっていることを確認します。

   a. **Workloads → StatefulSets** に移動します。

   b. Argo CD インスタンスをインストールしたネームスペースを **Project** ドロップダウンリストから選択します。

   c. **<instance_name>-application-controller** (例: **openshift-gitops-application-controller**) をクリックし、**Pod** タブに移動します。

   d. 作成されたアプリケーションコントローラー Pod の数を確認します。これは、セットレプリカの数に対応している必要があります。

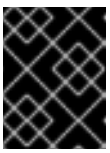
   e. 調べるコントローラー Pod をクリックし、**Logs** タブに移動して Pod ログを表示します。

   コントローラー Pod ログスニペットの例

   ```
   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) **"Using filter function: round-robin"** メッセージを探します。

   f. 次の例に示すように、ログの **Search** フィールドで **processed by shard** を検索して、シャード間でのクラスターの分布が均一であることを確認します。

   重要

   これらのログを確認するには、ログレベルを **debug** に設定していることを確認してください。

   コントローラー Pod ログスニペットの例

   ```
   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 この例では、3 つのクラスターがシャード 0、シャード 1、およびシャード 2 に連続して接続されています。

注記

クラスターの数 "C" がシャードレプリカの数 "R" の倍数である場合は、各シャードに同じ数のクラスター "N" が割り当てられている必要があります。これは、"C" を "R" で割ったものに相当します。前の例では、3 つのクラスターと 3 つのレプリカを示しています。したがって、各シャードには 1 つのクラスターが割り当てられます。

3.1.2. CLI を使用したラウンドロビンシャーディングアルゴリズムの有効化

コマンドラインインターフェイスを使用して、**round-robin** シャーディングアルゴリズムを有効にできます。

前提条件

- Red Hat OpenShift GitOps Operator がクラスターにインストールされている。
- **cluster-admin** 権限でクラスターにアクセスできる。

手順

1. 以下のコマンドを実行して、シャード化を有効にし、レプリカの数を必要な値に設定します。

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

出力例

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

2. 以下のコマンドを実行して、シャード化アルゴリズムを **round-robin** に設定します。

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

出力例

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

3. 次のコマンドを実行して、Argo CD アプリケーションコントローラー Pod の数がセットされたレプリカの数と一致していることを確認します。

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

出力例

```
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. 次のコマンドを実行して、シャーディングアルゴリズムとして **round-robin** を使用する設定で、シャーディングが有効になっていることを確認します。

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

出力の抜粋例

```
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) **"Using filter function: round-robin"** メッセージを探します。

5. 次の手順を実行して、シャード間でクラスターが均等に分散されていることを確認します。

   a. 次のコマンドを実行して、ログレベルを **debug** に設定します。

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

   出力例

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

   b. 次のコマンドを実行して、ログを表示し、**processed by shard** を検索して、各クラスターがどのシャードに接続されているかを確認します。

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

   出力の抜粋例

```
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-
```

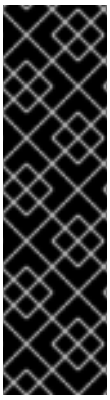
```
jrf6-wjjfyw833n23 will be processed by shard 1" 2
time="2023-12-13T09:05:34Z" level=debug msg="Cluster with id=836d8b53-96k4-f68r-
8wq0-sh72j22kl90w will be processed by shard 2" 3
```

1 2 3 この例では、3 つのクラスターがシャード 0、シャード 1、およびシャード 2 に連続して接続されています。

注記

クラスターの数 "C" がシャードレプリカの数 "R" の倍数である場合は、各シャードに同じ数のクラスター "N" が割り当てられている必要があります。これは、"C" を "R" で割ったものに相当します。前の例では、3 つのクラスターと 3 つのレプリカを示しています。したがって、各シャードには 1 つのクラスターが割り当てられます。

3.2. ARGO CD アプリケーションコントローラーのシャードの動的スケーリングを有効にする手順

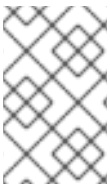


重要

シャードの動的スケーリングはテクノロジープレビューのみの機能です。テクノロジープレビュー機能は、Red Hat 製品のサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではないことがあります。Red Hat は、実稼働環境でこれらを使用することを推奨していません。テクノロジープレビューの機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

デフォルトでは、Argo CD アプリケーションコントローラーはクラスターをシャードに無期限に割り当てます。**round-robin** シャーディングアルゴリズムを使用している場合、この静的割り当てにより、特にレプリカが追加または削除されたときに、シャードが不均一に分散される可能性があります。シャードの動的なスケーリングを有効にして、特定の時点で Argo CD アプリケーションコントローラーが管理するクラスターの数に基づいてシャードの数を自動的に調整できます。これにより、シャードのバランスが確保され、コンピューティングリソースの使用が最適化されます。



注記

動的スケーリングを有効にした後は、シャード数を手動で変更できません。システムは、特定の時点で Argo CD アプリケーションコントローラーが管理するクラスターの数に基づいて、シャードの数を自動的に調整します。

3.2.1. Web コンソールでのシャードの動的スケーリングの有効化

OpenShift Container Platform Web コンソールを使用して、シャードの動的スケーリングを有効にできます。

前提条件

- **cluster-admin** 権限でクラスターにアクセスできる。

- OpenShift Container Platform Web コンソールにアクセスできる。
- Red Hat OpenShift GitOps Operator がクラスターにインストールされている。

手順

1. OpenShift Container Platform Web コンソールの **Administrator** パースペクティブで、**Operators → Installed Operators** に移動します。
2. **Installed Operators** のリストから Red Hat OpenShift GitOps Operator を選択し、 **ArgoCD** タブをクリックします。
3. シャードの動的スケーリングを有効にする Argo CD インスタンス名 (例: **openshift-gitops)** を選択します。
4. **YAML** タブをクリックし、以下のように **spec.controller.sharding** プロパティーを編集して設定します。

   動的なスケーリングを有効にした Argo CD YAML ファイルの例

```
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) 動的スケーリングを有効にするには、**dynamicScalingEnabled** を **true** に設定します。

   (2) **minShards** は、シャードの最小要件数に設定します。この値は **1** 以上に設定する必要があります。

   (3) **maxShards** を、シャードの最大要件数に設定します。値は **minShards** の値よりも大きくする必要があります。

   (4) **clustersPerShard** は、シャードごとに必要なクラスターの数に設定します。この値は **1** 以上に設定する必要があります。

5. **Save** をクリックします。
   成功通知アラート **openshift-gitops has been updated to version <version>** が表示されます。

注記

デフォルトの **openshift-gitops** インスタンスを編集すると、Managed resource ダイアログボックスが表示されます。**Save** をもう一度クリックして、変更を確定します。

検証

namespace の Pod 数をチェックして、シャード化が有効になっていることを確認します。

1. **Workloads → StatefulSets** に移動します。
2. Argo CD インスタンスがデプロイされている namespace を **Project** ドロップダウンリストから選択します (例: **openshift-gitops**)。
3. Argo CD インスタンスの名前を持つ **StatefulSet** オブジェクトの名前 (例: **openshift-gitops-apllication-controller)** をクリックします。
4. **Pod** タブをクリックし、Pod の数が Argo CD **YAML** ファイルで設定した **minShards** の値以上であることを確認します。

3.2.2. CLI を使用したシャードの動的スケーリングの有効化

OpenShift CLI (**oc**) を使用して、シャードの動的スケーリングを有効にできます。

前提条件

- Red Hat OpenShift GitOps Operator がクラスターにインストールされている。
- **cluster-admin** 権限でクラスターにアクセスできる。

手順

1. **oc** ツールを使用して、**cluster-admin** 権限を持つユーザーとしてクラスターにログインします。
2. 次のコマンドを実行して、動的スケーリングを有効にします。

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

コマンドの例

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

1 このサンプルコマンドは、**openshift-gitops** namespace の **openshift-gitops** Argo CD インスタンスの動的スケーリングを有効にし、シャードの最小数を **1** に、シャードの最大数を **3** に、およびシャードごとのクラスター数を **1** に設定します。**minShard** および **clustersPerShard** の値は、**1** 以上に設定する必要があります。**maxShard** の値は、**minShard** の値以下である必要があります。

出力例

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

検証

1. Argo CD インスタンスの **spec.controller.sharding** プロパティーを確認します。

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

コマンドの例

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

シャードの動的スケーリングが有効になっている場合の出力例

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

2. オプション: OpenShift Container Platform Web コンソールで Argo CD インスタンスの設定 **YAML** ファイルにある設定された **spec.controller.sharding** プロパティーをチェックして、動的スケーリングが有効になっていることを確認します。

3. Argo CD Application Controller Pod の数を確認します。

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

コマンドの例

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

出力例

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

1 Argo CD Application Controller Pod の数は、**minShard** の値以下である必要があります。

関連情報

- Argo CD カスタムリソースプロパティー
- Horizontal Pod Autoscaler での Pod の自動スケーリング