



Red Hat OpenShift GitOps 1.12

Declarative cluster configuration

Configuring an OpenShift cluster with cluster configurations by using OpenShift GitOps and creating and synchronizing applications in the default and code mode by using the GitOps CLI.

Red Hat OpenShift GitOps 1.12 Declarative cluster configuration

Configuring an OpenShift cluster with cluster configurations by using OpenShift GitOps and creating and synchronizing applications in the default and code mode by using the GitOps CLI.

Legal Notice

Copyright © 2024 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

This document provides instructions for configuring Argo CD to recursively sync the content of a Git directory with an application that contains custom configurations for your cluster. It also discusses about how to create and synchronize applications in the default and code mode by using the GitOps CLI.

Table of Contents

CHAPTER 1. CONFIGURING AN OPENSIFT CLUSTER BY DEPLOYING AN APPLICATION WITH CLUSTER CONFIGURATIONS	3
1.1. PREREQUISITES	3
1.2. USING AN ARGO CD INSTANCE TO MANAGE CLUSTER-SCOPED RESOURCES	3
1.3. DEFAULT PERMISSIONS OF AN ARGO CD INSTANCE	4
1.4. RUNNING THE ARGO CD INSTANCE AT THE CLUSTER-LEVEL	4
1.5. CREATING AN APPLICATION BY USING THE ARGO CD DASHBOARD	5
1.6. CREATING AN APPLICATION BY USING THE OC TOOL	7
1.7. CREATING AN APPLICATION IN THE DEFAULT MODE BY USING THE GITOPS CLI	7
1.8. CREATING AN APPLICATION IN CORE MODE BY USING THE GITOPS CLI	9
1.9. SYNCHRONIZING YOUR APPLICATION WITH YOUR GIT REPOSITORY	10
1.10. SYNCHRONIZING AN APPLICATION IN THE DEFAULT MODE BY USING THE GITOPS CLI	11
1.11. SYNCHRONIZING AN APPLICATION IN CORE MODE BY USING THE GITOPS CLI	12
1.12. IN-BUILT PERMISSIONS FOR CLUSTER CONFIGURATION	13
1.13. ADDING PERMISSIONS FOR CLUSTER CONFIGURATION	13
1.14. INSTALLING OLM OPERATORS USING RED HAT OPENSIFT GITOPS	15
1.14.1. Installing cluster-scoped Operators	15
1.14.2. Installing namespace-scoped Operators	15
1.15. ADDITIONAL RESOURCES	16
CHAPTER 2. SHARDING CLUSTERS ACROSS ARGO CD APPLICATION CONTROLLER REPLICAS	17
2.1. ENABLING THE ROUND-ROBIN SHARDING ALGORITHM	17
2.1.1. Enabling the round-robin sharding algorithm in the web console	17
2.1.2. Enabling the round-robin sharding algorithm by using the CLI	20
2.2. ENABLING DYNAMIC SCALING OF SHARDS OF THE ARGO CD APPLICATION CONTROLLER	22
2.2.1. Enabling dynamic scaling of shards in the web console	22
2.2.2. Enabling dynamic scaling of shards by using the CLI	24
2.2.3. Additional resources	25

CHAPTER 1. CONFIGURING AN OPENSIFT CLUSTER BY DEPLOYING AN APPLICATION WITH CLUSTER CONFIGURATIONS

With Red Hat OpenShift GitOps, you can configure Argo CD to recursively sync the content of a Git directory with an application that contains custom configurations for your cluster.

1.1. PREREQUISITES

- You have logged in to the OpenShift Container Platform cluster as an administrator.
- You have installed the Red Hat OpenShift GitOps Operator on your OpenShift Container Platform cluster.

1.2. USING AN ARGO CD INSTANCE TO MANAGE CLUSTER-SCOPED RESOURCES

To manage cluster-scoped resources, update the existing **Subscription** object for the Red Hat OpenShift GitOps Operator and add the namespace of the Argo CD instance to the **ARGOCD_CLUSTER_CONFIG_NAMESPACES** environment variable in the **spec** section.

Procedure

1. In the **Administrator** perspective of the web console, navigate to **Operators** → **Installed Operators** → **Red Hat OpenShift GitOps** → **Subscription**.
2. Click the **Actions** drop-down menu then click **Edit Subscription**.
3. On the **openshift-gitops-operator** Subscription details page, under the **YAML** tab, edit the **Subscription** YAML file by adding the namespace of the Argo CD instance to the **ARGOCD_CLUSTER_CONFIG_NAMESPACES** environment variable in the **spec** section:

```
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. To verify that the Argo instance is configured with a cluster role to manage cluster-scoped resources, perform the following steps:
 - a. Navigate to **User Management** → **Roles** and from the **Filter** drop-down menu select **Cluster-wide Roles**.
 - b. Search for the **argocd-application-controller** by using the **Search by name** field. The **Roles** page displays the created cluster role.

TIP

Alternatively, in the OpenShift CLI, run the following command:

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

The output **yes** verifies that the Argo instance is configured with a cluster role to manage cluster-scoped resources. Else, check your configurations and take necessary steps as required.

1.3. DEFAULT PERMISSIONS OF AN ARGO CD INSTANCE

By default Argo CD instance has the following permissions:

- Argo CD instance has the **admin** privileges to manage resources only in the namespace where it is deployed. For instance, an Argo CD instance deployed in the **foo** namespace has the **admin** privileges to manage resources only for that namespace.
- Argo CD has the following cluster-scoped permissions because Argo CD requires cluster-wide **read** privileges on resources to function appropriately:

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

NOTE

- You can edit the cluster roles used by the **argocd-server** and **argocd-application-controller** components where Argo CD is running such that the **write** privileges are limited to only the namespaces and resources that you wish Argo CD to manage.

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

1.4. RUNNING THE ARGO CD INSTANCE AT THE CLUSTER-LEVEL

The default Argo CD instance and the accompanying controllers, installed by the Red Hat OpenShift GitOps Operator, can now run on the infrastructure nodes of the cluster by setting a simple configuration toggle.

Procedure

1. Label the existing nodes:

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

2. Optional: If required, you can also apply taints and isolate the workloads on infrastructure nodes and prevent other workloads from scheduling on these nodes:

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

3. Add the **runOnInfra** toggle in the **GitOpsService** custom resource:

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

4. Optional: If taints have been added to the nodes, then add **tolerations** to the **GitOpsService** custom resource, for example:

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

5. Verify that the workloads in the **openshift-gitops** namespace are now scheduled on the infrastructure nodes by viewing **Pods** → **Pod details** for any pod in the console UI.



NOTE

Any **nodeSelectors** and **tolerations** manually added to the default Argo CD custom resource are overwritten by the toggle and **tolerations** in the **GitOpsService** custom resource.

Additional resources

- To learn more about taints and tolerations, see [Controlling pod placement using node taints](#).
- For more information on infrastructure machine sets, see [Creating infrastructure machine sets](#).


1.5. CREATING AN APPLICATION BY USING THE ARGO CD DASHBOARD

Argo CD provides a dashboard which allows you to create applications.

This sample workflow walks you through the process of configuring Argo CD to recursively sync the

content of the **cluster** directory to the **cluster-configs** application. The directory defines the OpenShift Container Platform web console cluster configurations that add a link to the **Red Hat**



Developer Blog - Kubernetes under the  menu in the web console, and defines a namespace **spring-petclinic** on the cluster.

Prerequisites

- You have logged in to the OpenShift Container Platform cluster as an administrator.
- You have installed the Red Hat OpenShift GitOps Operator on your OpenShift Container Platform cluster.
- You have logged in to Argo CD instance.

Procedure

1. In the Argo CD dashboard, click **NEW APP** to add a new Argo CD application.
2. For this workflow, create a **cluster-configs** application with the following configurations:

Application Name

cluster-configs

Project

default

Sync Policy

Manual

Repository URL

<https://github.com/redhat-developer/openshift-gitops-getting-started>

Revision

HEAD

Path

cluster

Destination

<https://kubernetes.default.svc>

Namespace

spring-petclinic

Directory Recurse

checked

3. Click **CREATE** to create your application.
4. Open the **Administrator** perspective of the web console and expand **Administration** → **Namespaces**.
5. Search for and select the namespace, then enter **argocd.argoproj.io/managed-by=openshift-gitops** in the **Label** field so that the Argo CD instance in the **openshift-gitops** namespace can manage your namespace.

1.6. CREATING AN APPLICATION BY USING THE `oc` TOOL

You can create Argo CD applications in your terminal by using the `oc` tool.

Prerequisites

- You have installed the Red Hat OpenShift GitOps Operator on your OpenShift Container Platform cluster.
- You have logged in to an Argo CD instance.

Procedure

1. Download [the sample application](#):

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

2. Create the application:

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

3. Run the `oc get` command to review the created application:

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

4. Add a label to the namespace your application is deployed in so that the Argo CD instance in the **openshift-gitops** namespace can manage it:

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

1.7. CREATING AN APPLICATION IN THE DEFAULT MODE BY USING THE GITOPS CLI

You can create applications in the default mode by using the GitOps **argocd** CLI.

This sample workflow walks you through the process of configuring Argo CD to recursively sync the content of the **cluster** directory to the **cluster-configs** application. The directory defines the OpenShift Container Platform cluster configurations and the **spring-petclinic** namespace on the cluster.

Prerequisites

- You have installed the Red Hat OpenShift GitOps Operator on your OpenShift Container Platform cluster.
- You have installed the OpenShift CLI (`oc`).
- You have installed the Red Hat OpenShift GitOps **argocd** CLI.
- You have logged in to Argo CD instance.

Procedure

1. Get the **admin** account password for the Argo CD server:

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

2. Get the Argo CD server URL:

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

3. Log in to the Argo CD server by using the **admin** account password and enclosing it in single quotes:



IMPORTANT

Enclosing the password in single quotes ensures that special characters, such as **\$**, are not misinterpreted by the shell. Always use single quotes to enclose the literal value of the password.

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

Example

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

4. Verify that you are able to run **argocd** commands in the default mode by listing all applications:

```
$ argocd app list
```

If the configuration is correct, then existing applications will be listed with the following header:

Sample output

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

5. Create an application in the default mode:

```
$ 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. Label the **spring-petclinic** destination namespace to be managed by the **openshift-gitops** Argo CD instance:

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

- List the available applications to confirm that the application is created successfully:

```
$ argocd app list
```

Even though the **cluster-configs** Argo CD application has the **Healthy** status, it is not automatically synced due to its **none** sync policy, causing it to remain in the **OutOfSync** status.

1.8. CREATING AN APPLICATION IN CORE MODE BY USING THE GITOPS CLI

You can create applications in **core** mode by using the GitOps **argocd** CLI.

This sample workflow walks you through the process of configuring Argo CD to recursively sync the content of the **cluster** directory to the **cluster-configs** application. The directory defines the OpenShift Container Platform cluster configurations and the **spring-petclinic** namespace on the cluster.

Prerequisites

- You have installed the Red Hat OpenShift GitOps Operator on your OpenShift Container Platform cluster.
- You have installed the OpenShift CLI (**oc**).
- You have installed the Red Hat OpenShift GitOps **argocd** CLI.

Procedure

- Log in to the OpenShift Container Platform cluster by using the **oc** CLI tool:

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

Example

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

- Check whether the context is set correctly in the **kubeconfig** file:

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

- Set the default namespace of the current context to **openshift-gitops**:

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

- Set the following environment variable to override the Argo CD component names:

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

- Verify that you are able to run **argocd** commands in **core** mode by listing all applications:

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

If the configuration is correct, then existing applications will be listed with the following header:

Sample output

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

6. Create an application in **core** mode:

```
$ 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. Label the **spring-petclinic** destination namespace to be managed by the **openshift-gitops** Argo CD instance:

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

8. List the available applications to confirm that the application is created successfully:

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


Even though the **cluster-configs** Argo CD application has the **Healthy** status, it is not automatically synced due to its **none** sync policy, causing it to remain in the **OutOfSync** status.

1.9. SYNCHRONIZING YOUR APPLICATION WITH YOUR GIT REPOSITORY

You can synchronize your application with your Git repository by modifying the synchronization policy for Argo CD. The policy modification automatically applies the changes in your cluster configurations from your Git repository to the cluster.

Procedure

1. In the Argo CD dashboard, notice that the **cluster-configs** Argo CD application has the statuses **Missing** and **OutOfSync**. Because the application was configured with a manual sync policy, Argo CD does not sync it automatically.
2. Click **SYNC** on the **cluster-configs** tile, review the changes, and then click **SYNCHRONIZE**. Argo CD will detect any changes in the Git repository automatically. If the configurations are changed, Argo CD will change the status of the **cluster-configs** to **OutOfSync**. You can modify the synchronization policy for Argo CD to automatically apply changes from your Git repository to the cluster.

3. Notice that the **cluster-configs** Argo CD application now has the statuses **Healthy** and **Synced**. Click the **cluster-configs** tile to check the details of the synchronized resources and their status on the cluster.
4. Navigate to the OpenShift Container Platform web console and click  to verify that a link to the **Red Hat Developer Blog - Kubernetes** is now present there.
5. Navigate to the **Project** page and search for the **spring-petclinic** namespace to verify that it has been added to the cluster.
Your cluster configurations have been successfully synchronized to the cluster.

1.10. SYNCHRONIZING AN APPLICATION IN THE DEFAULT MODE BY USING THE GITOPS CLI

You can synchronize applications in the default mode by using the GitOps **argocd** CLI.

This sample workflow walks you through the process of configuring Argo CD to recursively sync the content of the **cluster** directory to the **cluster-configs** application. The directory defines the OpenShift Container Platform cluster configurations and the **spring-petclinic** namespace on the cluster.

Prerequisites

- You have installed the Red Hat OpenShift GitOps Operator on your OpenShift Container Platform cluster.
- You have logged in to Argo CD instance.
- You have installed the OpenShift CLI (**oc**).
- You have installed the Red Hat OpenShift GitOps **argocd** CLI.

Procedure

1. Get the **admin** account password for the Argo CD server:

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

2. Get the Argo CD server URL:

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

3. Log in to the Argo CD server by using the **admin** account password and enclosing it in single quotes:



IMPORTANT

Enclosing the password in single quotes ensures that special characters, such as **\$**, are not misinterpreted by the shell. Always use single quotes to enclose the literal value of the password.

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

Example

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

4. Because the application is configured with the **none** sync policy, you must manually trigger the sync operation:

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

5. List the application to confirm that the application has the **Healthy** and **Synced** statuses:

```
$ argocd app list
```

1.11. SYNCHRONIZING AN APPLICATION IN CORE MODE BY USING THE GITOPS CLI

You can synchronize applications in **core** mode by using the GitOps **argocd** CLI.

This sample workflow walks you through the process of configuring Argo CD to recursively sync the content of the **cluster** directory to the **cluster-configs** application. The directory defines the OpenShift Container Platform cluster configurations and the **spring-petclinic** namespace on the cluster.

Prerequisites

- You have installed the Red Hat OpenShift GitOps Operator on your OpenShift Container Platform cluster.
- You have installed the OpenShift CLI (**oc**).
- You have installed the Red Hat OpenShift GitOps **argocd** CLI.

Procedure

1. Log in to the OpenShift Container Platform cluster by using the **oc** CLI tool:

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

Example

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

2. Check whether the context is set correctly in the **kubeconfig** file:

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

3. Set the default namespace of the current context to **openshift-gitops**:


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

- Set the following environment variable to override the Argo CD component names:

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

- Because the application is configured with the **none** sync policy, you must manually trigger the sync operation:

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

- List the application to confirm that the application has the **Healthy** and **Synced** statuses:

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

1.12. IN-BUILT PERMISSIONS FOR CLUSTER CONFIGURATION

By default, the Argo CD instance has permissions to manage specific cluster-scoped resources such as cluster Operators, optional OLM Operators and user management.



NOTE

Argo CD does not have cluster-admin permissions.

Permissions for the Argo CD instance:

Resources	Descriptions
Resource Groups	Configure the user or administrator
operators.coreos.com	Optional Operators managed by OLM
user.openshift.io , rbac.authorization.k8s.io	Groups, Users and their permissions
config.openshift.io	Control plane Operators managed by CVO used to configure cluster-wide build configuration, registry configuration and scheduler policies
storage.k8s.io	Storage
console.openshift.io	Console customization

1.13. ADDING PERMISSIONS FOR CLUSTER CONFIGURATION

You can grant permissions for an Argo CD instance to manage cluster configuration. Create a cluster role with additional permissions and then create a new cluster role binding to associate the cluster role with a service account.

Prerequisites

- You have access to an OpenShift Container Platform cluster with **cluster-admin** privileges and are logged in to the web console.
- You have installed the Red Hat OpenShift GitOps Operator on your OpenShift Container Platform cluster.

Procedure

1. In the web console, select **User Management** → **Roles** → **Create Role**. Use the following **ClusterRole** YAML template to add rules to specify the additional permissions.

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

2. Click **Create** to add the cluster role.
3. To create the cluster role binding, select **User Management** → **Role Bindings** → **Create Binding**.
4. Select **All Projects** from the **Project** drop-down.
5. Click **Create binding**.
6. Select **Binding type** as **Cluster-wide role binding (ClusterRoleBinding)**.
7. Enter a unique value for the **RoleBinding** name.
8. Select the newly created cluster role or an existing cluster role from the drop down list.
9. Select the **Subject** as **ServiceAccount** and then provide the **Subject namespace** and **name**.
 - a. **Subject namespace:** **openshift-gitops**
 - b. **Subject name:** **openshift-gitops-argocd-application-controller**
10. Click **Create**. The YAML file for the **ClusterRoleBinding** object is as follows:

```
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. INSTALLING OLM OPERATORS USING RED HAT OPENSIFT GITOPS

Red Hat OpenShift GitOps with cluster configurations manages specific cluster-scoped resources and takes care of installing cluster Operators or any namespace-scoped OLM Operators.

Consider a case where as a cluster administrator, you have to install an OLM Operator such as Tekton. You use the OpenShift Container Platform web console to manually install a Tekton Operator or the OpenShift CLI to manually install a Tekton subscription and Tekton Operator group on your cluster.

Red Hat OpenShift GitOps places your Kubernetes resources in your Git repository. As a cluster administrator, use Red Hat OpenShift GitOps to manage and automate the installation of other OLM Operators without any manual procedures. For example, after you place the Tekton subscription in your Git repository by using Red Hat OpenShift GitOps, the Red Hat OpenShift GitOps automatically takes this Tekton subscription from your Git repository and installs the Tekton Operator on your cluster.

1.14.1. Installing cluster-scoped Operators

Operator Lifecycle Manager (OLM) uses a default **global-operators** Operator group in the **openshift-operators** namespace for cluster-scoped Operators. Hence you do not have to manage the **OperatorGroup** resource in your Gitops repository. However, for namespace-scoped Operators, you must manage the **OperatorGroup** resource in that namespace.

To install cluster-scoped Operators, create and place the **Subscription** resource of the required Operator in your Git repository.

Example: Grafana Operator subscription

```

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. Installing namespace-scoped Operators

To install namespace-scoped Operators, create and place the **Subscription** and **OperatorGroup** resources of the required Operator in your Git repository.

Example: Ansible Automation Platform Resource Operator

```

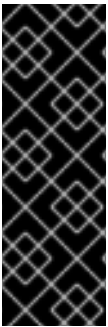
# ...
apiVersion: v1
kind: Namespace

```

```

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
# ...

```



IMPORTANT

When deploying multiple Operators using Red Hat OpenShift GitOps, you must create only a single Operator group in the corresponding namespace. If more than one Operator group exists in a single namespace, any CSV created in that namespace transition to a **failure** state with the **TooManyOperatorGroups** reason. After the number of Operator groups in their corresponding namespaces reaches one, all the previous **failure** state CSVs transition to **pending** state. You must manually approve the pending install plan to complete the Operator installation.

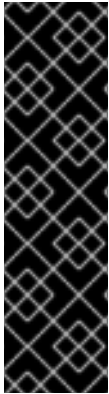
1.15. ADDITIONAL RESOURCES

- [Installing the GitOps CLI](#)
- [Basic GitOps argocd commands](#)

CHAPTER 2. SHARDING CLUSTERS ACROSS ARGO CD APPLICATION CONTROLLER REPLICAS

You can shard clusters across multiple Argo CD Application Controller replicas if the controller is managing too many clusters and uses too much memory.

2.1. ENABLING THE ROUND-ROBIN SHARDING ALGORITHM



IMPORTANT

The **round-robin** sharding algorithm is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see [Technology Preview Features Support Scope](#).

By default, the Argo CD Application Controller uses the non-uniform **legacy** hash-based sharding algorithm to assign clusters to shards. This can result in uneven cluster distribution. You can enable the **round-robin** sharding algorithm to achieve more equal cluster distribution across all shards.

Using the **round-robin** sharding algorithm in Red Hat OpenShift GitOps provides the following benefits:

- Ensure more balanced workload distribution
- Prevent shards from being overloaded or underutilized
- Optimize the efficiency of computing resources
- Reduce the risk of bottlenecks
- Improve overall performance and reliability of the Argo CD system

The introduction of alternative sharding algorithms allows for further customization based on specific use cases. You can select the algorithm that best aligns with your deployment needs, which results in greater flexibility and adaptability in diverse operational scenarios.

TIP

To leverage the benefits of alternative sharding algorithms in GitOps, it is crucial to enable sharding during deployment.

2.1.1. Enabling the round-robin sharding algorithm in the web console

You can enable the **round-robin** sharding algorithm by using the OpenShift Container Platform web console.

Prerequisites

- You have installed the Red Hat OpenShift GitOps Operator on your OpenShift Container Platform cluster.
- You have access to the OpenShift Container Platform web console.
- You have access to the cluster with **cluster-admin** privileges.

Procedure

1. In the **Administrator** perspective of the web console, go to **Operators** → **Installed Operators**.
2. Click **Red Hat OpenShift GitOps** from the installed operators and go to the **Argo CD** tab.
3. Click the Argo CD instance where you want to enable the **round-robin** sharding algorithm, for example, **openshift-gitops**.
4. Click the **YAML** tab and edit the YAML file as shown in the following example:

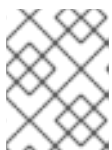
Example Argo CD instance with round-robin sharding algorithm enabled

```

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 Set the **sharding.enabled** parameter to **true** to enable sharding.
 - 2 Set the number of replicas to the wanted value, for example, **3**.
 - 3 Set the sharding algorithm to **round-robin**.
 - 4 Set the log level to **debug** so that you can verify to which shard each cluster is attached.
5. Click **Save**.
A success notification alert, **openshift-gitops has been updated to version <version>**, appears.



NOTE

If you edit the default **openshift-gitops** instance, the **Managed resource** dialog box is displayed. Click **Save** again to confirm the changes.

6. Verify that the sharding is enabled with **round-robin** as the sharding algorithm by performing the following steps:

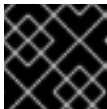
- a. Go to **Workloads** → **StatefulSets**.
- b. Select the namespace where you installed the Argo CD instance from the **Project** drop-down list.
- c. Click **<instance_name>-application-controller**, for example, **openshift-gitops-application-controller**, and go to the **Pods** tab.
- d. Observe the number of created application controller pods. It should correspond with the number of set replicas.
- e. Click on the controller pod you want to examine and go to the **Logs** tab to view the pod logs.

Example controller pod logs snippet

```
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" ❶
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"
```

- ❶ Look for the **"Using filter function: round-robin"** message.

- f. In the log **Search** field, search for **processed by shard** to verify that the cluster distribution across shards is even, as shown in the following example.



IMPORTANT

Ensure that you set the log level to **debug** to observe these logs.

Example controller pod logs snippet

```
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-wjfyw833n23 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" ❶
time="2023-12-13T09:05:34Z" level=debug msg="Cluster with id=068d8b26-6rhi-4w23-
jrf6-wjfyw833n23 will be processed by shard 1" ❷
time="2023-12-13T09:05:34Z" level=debug msg="Cluster with id=836d8b53-96k4-f68r-
8wq0-sh72j22kl90w will be processed by shard 2" ❸
```

- ❶ ❷ ❸ In this example, 3 clusters are attached consecutively to shard 0, shard 1, and shard 2.

**NOTE**

If the number of clusters "C" is a multiple of the number of shard replicas "R", then each shard must have the same number of assigned clusters "N", which is equal to "C" divided by "R". The previous example shows 3 clusters and 3 replicas; therefore, each shard has 1 cluster assigned.

2.1.2. Enabling the round-robin sharding algorithm by using the CLI

You can enable the **round-robin** sharding algorithm by using the command-line interface.

Prerequisites

- You have installed the Red Hat OpenShift GitOps Operator on your OpenShift Container Platform cluster.
- You have access to the cluster with **cluster-admin** privileges.

Procedure

1. Enable sharding and set the number of replicas to the wanted value by running the following command:

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

Example output

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

2. Configure the sharding algorithm to **round-robin** by running the following command:

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

Example output

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

3. Verify that the number of Argo CD Application Controller pods corresponds with the number of set replicas by running the following command:

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

Example output

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. Verify that the sharding is enabled with **round-robin** as the sharding algorithm by running the following command:

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

Example output snippet

```
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** Look for the **"Using filter function: round-robin"** message.

5. Verify that the cluster distribution across shards is even by performing the following steps:

- a. Set the log level to **debug** by running the following command:

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

Example output

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

- b. View the logs and search for **processed by shard** to observe to which shard each cluster is attached by running the following command:

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

Example output snippet

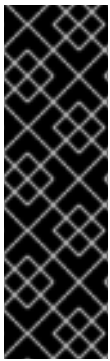
```
time="2023-12-13T09:05:34Z" level=debug msg="Cluster with id= will be processed by
shard 0" 1
time="2023-12-13T09:05:34Z" level=debug msg="Cluster with id=068d8b26-6rhi-4w23-
jrf6-wjjfyw833n23 will be processed by shard 1" 2
time="2023-12-13T09:05:34Z" level=debug msg="Cluster with id=836d8b53-96k4-f68r-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

- 1 2 3** In this example, 3 clusters are attached consecutively to shard 0, shard 1, and shard 2.

**NOTE**

If the number of clusters "C" is a multiple of the number of shard replicas "R", then each shard must have the same number of assigned clusters "N", which is equal to "C" divided by "R". The previous example shows 3 clusters and 3 replicas; therefore, each shard has 1 cluster assigned.

2.2. ENABLING DYNAMIC SCALING OF SHARDS OF THE ARGO CD APPLICATION CONTROLLER

**IMPORTANT**

Dynamic scaling of shards is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see [Technology Preview Features Support Scope](#).

By default, the Argo CD Application Controller assigns clusters to shards indefinitely. If you are using the **round-robin** sharding algorithm, this static assignment can result in uneven distribution of shards, particularly when replicas are added or removed. You can enable dynamic scaling of shards to automatically adjust the number of shards based on the number of clusters managed by the Argo CD Application Controller at a given time. This ensures that shards are well-balanced and optimizes the use of compute resources.

**NOTE**

After you enable dynamic scaling, you cannot manually modify the shard count. The system automatically adjusts the number of shards based on the number of clusters managed by the Argo CD Application Controller at a given time.

2.2.1. Enabling dynamic scaling of shards in the web console

You can enable dynamic scaling of shards by using the OpenShift Container Platform web console.

Prerequisites

- You have access to the cluster with **cluster-admin** privileges.
- You have access to the OpenShift Container Platform web console.
- You have installed the Red Hat OpenShift GitOps Operator on your OpenShift Container Platform cluster.

Procedure

1. In the **Administrator** perspective of the OpenShift Container Platform web console, go to **Operators** → **Installed Operators**.

- From the the list of **Installed Operators**, select the Red Hat OpenShift GitOps Operator, and then click the **ArgoCD** tab.
- Select the Argo CD instance name for which you want to enable dynamic scaling of shards, for example, **openshift-gitops**.
- Click the **YAML** tab, and then edit and configure the **spec.controller.sharding** properties as follows:

Example Argo CD YAML file with dynamic scaling enabled

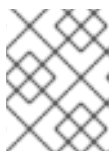
```

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

```

- Set **dynamicScalingEnabled** to **true** to enable dynamic scaling.
- Set **minShards** to the minimum number of shards that you want to have. The value must be set to **1** or greater.
- Set **maxShards** to the maximum number of shards that you want to have. The value must be greater than the value of **minShards**.
- Set **clustersPerShard** to the number of clusters that you want to have per shard. The value must be set to **1** or greater.

- Click **Save**.
A success notification alert, **openshift-gitops has been updated to version <version>**, appears.



NOTE

If you edit the default **openshift-gitops** instance, the **Managed resource** dialog box is displayed. Click **Save** again to confirm the changes.

Verification

Verify that sharding is enabled by checking the number of pods in the namespace:

- Go to **Workloads** → **StatefulSets**.
- Select the namespace where the Argo CD instance is deployed from the **Project** drop-down list, for example, **openshift-gitops**.

- Click the name of the **StatefulSet** object that has the name of the Argo CD instance, for example **openshift-gitops-application-controller**.
- Click the **Pods** tab, and then verify that the number of pods is equal to or greater than the value of **minShards** that you have set in the Argo CD **YAML** file.

2.2.2. Enabling dynamic scaling of shards by using the CLI

You can enable dynamic scaling of shards by using the OpenShift CLI (**oc**).

Prerequisites

- You have installed the Red Hat OpenShift GitOps Operator on your OpenShift Container Platform cluster.
- You have access to the cluster with **cluster-admin** privileges.

Procedure

- Log in to the cluster by using the **oc** tool as a user with **cluster-admin** privileges.
- Enable dynamic scaling by running the following command:

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

Example command

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

- 1** The example command enables dynamic scaling for the **openshift-gitops** Argo CD instance in the **openshift-gitops** namespace, and sets the minimum number of shards to **1**, the maximum number of shards to **3**, and the number of clusters per shard to **1**. The values of **minShard** and **clustersPerShard** must be set to **1** or greater. The value of **maxShard** must be equal to or greater than the value of **minShard**.

Example output

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

Verification

- Check the **spec.controller.sharding** properties of the Argo CD instance:

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

Example command

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

Example output when dynamic scaling of shards is enabled

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

- Optional: Verify that dynamic scaling is enabled by checking the configured **spec.controller.sharding** properties in the configuration **YAML** file of the Argo CD instance in the OpenShift Container Platform web console.
- Check the number of Argo CD Application Controller pods:

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

Example command

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

Example output

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

- The number of Argo CD Application Controller pods must be equal to or greater than the value of **minShard**.

2.2.3. Additional resources

- [Argo CD custom resource properties](#)
- [Automatically scaling pods with the horizontal pod autoscaler](#)