Declarative cluster configuration

Configuring an OpenShift cluster with cluster configurations by using OpenShift GitOps
Configuring an OpenShift cluster with cluster configurations by using OpenShift GitOps
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.
# Table of Contents

CHAPTER 1. CONFIGURING AN OPENSSHIFT 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 ARGOCD 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 .................................................. 6
1.7. SYNCHRONIZING YOUR APPLICATION WITH YOUR GIT REPOSITORY ....................... 7
1.8. IN-BUILT PERMISSIONS FOR CLUSTER CONFIGURATION ........................................... 7
1.9. ADDING PERMISSIONS FOR CLUSTER CONFIGURATION .............................................. 8
1.10. INSTALLING OLM OPERATORS USING RED HAT OPENShift GITOPS ....................... 9
    1.10.1. Installing cluster-scoped Operators .................................................................... 9
    1.10.2. Installing namespace-scoped Operators ............................................................ 10
CHAPTER 1. CONFIGURING AN OPENShift 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 in your cluster.
- You have logged into Argo CD instance.

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

To manage cluster-scoped resources, update the existing Subscription object for the `gitops-title` 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:

```yaml
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: openshift-gitops-operator
  namespace: openshift-operators
# ...
spec:
  config:
    env:
      - name: ARGOCDD_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:

```bash
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 ARGOCD 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:

  ```yaml
  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.

  ```bash
  $ 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.

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 navigate to Administration → Namespaces in the menu on the left.

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.

Procedure

1. Download the sample application:
2. Create the application:

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

   $ 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. 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.

### 1.8. 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.
NONE

Argo CD does not have cluster-admin permissions.

Permissions for the Argo CD instance:

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

1.9. 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.

Procedure

1. Log in to the OpenShift Container Platform web console as an admin.

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

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

3. Click Create to add the cluster role.

4. Now create the cluster role binding. In the web console, select User Management → Role Bindings → Create Binding.

5. Select All Projects from the Project drop-down.
6. Click Create binding.

7. Select Binding type as Cluster-wide role binding (ClusterRoleBinding).

8. Enter a unique value for the RoleBinding name.

9. Select the newly created cluster role or an existing cluster role from the drop down list.

10. Select the Subject as ServiceAccount and the provide the Subject namespace and name.
    a. Subject namespace: openshift-gitops
    b. Subject name: openshift-gitops-argocd-application-controller

11. Click Create. The YAML file for the ClusterRoleBinding object is as follows:

```yaml
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: admin
```

### 1.10. INSTALLING OLM OPERATORS USING RED HAT OPENShift 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.

#### 1.10.1. 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

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

1.10.2. 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

```yaml
# ...
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.