Red Hat OpenShift GitOps 1.13

Argo CD instance

Installing and deploying Argo CD instances, enabling notifications with an Argo CD instance, and configuring the NotificationsConfiguration CR.

Last Updated: 2024-07-22

Red Hat OpenShift GitOps 1.13 Argo CD instance

Installing and deploying Argo CD instances, enabling notifications with an Argo CD instance, and configuring the NotificationsConfiguration CR.

Legal Notice

Abstract

This document provides instructions for installing and deploying Argo CD instances to manage cluster configurations or deploy applications. It also discusses about how to enable notifications for an Argo CD instance and configure the NotificationsConfiguration custom resource (CR).

Table of Contents

CHAPTER 1. SETTING UP AN ARGO CD INSTANCE 3
1.1. INSTALLING A USER-DEFINED ARGO CD INSTANCE 3
1.2. CONFIGURING COMMON CLUSTER ROLES BY SPECIFYING USER-DEFINED CLUSTER ROLES FOR NAMESPACE-SCOPED INSTANCES 5
1.3. ENABLING REPLICAS FOR ARGO CD SERVER AND REPO SERVER 6
1.4. DEPLOYING RESOURCES TO A DIFFERENT NAMESPACE 7
1.5. CUSTOMIZING THE ARGO CD CONSOLE LINK 7

CHAPTER 2. ARGO CD CUSTOM RESOURCE AND COMPONENT PROPERTIES 9
2.1. ARGO CD CUSTOM RESOURCE PROPERTIES 9
2.2. REPO SERVER PROPERTIES 21
2.3. ENABLING NOTIFICATIONS WITH AN ARGO CD INSTANCE 22
2.4. NOTIFICATIONSCONFIGURATION CUSTOM RESOURCE PROPERTIES 23
2.4.1. Configuring the NotificationsConfiguration CR by using the web console 26
2.4.2. Configuring the NotificationsConfiguration CR by using the CLI 27
2.5. ADDITIONAL RESOURCES 27

CHAPTER 1. SETTING UP AN ARGO CD INSTANCE

By default, Red Hat OpenShift GitOps installs an instance of Argo CD in the **openshift-gitops** namespace with additional permissions for managing certain cluster-scoped resources. This default Argo CD instance is also called as the default cluster-scoped instance.

NOTE

For GitOps version 1.13 and later, the route TLS termination is set as default to the **reencrypt** mode for both the default and user-defined Argo CD instances. TLS connections to the Argo CD instances now receive the default ingress certificate that is set in OpenShift Container Platform, instead of the self-signed Argo CD certificate.

You can modify the route TLS termination policy by configuring the **.spec.server.route.tls** field of the Argo CD CR.

To manage cluster configurations or deploy applications, you can install and deploy a new user-defined Argo CD instance. By default, any new user-defined instance has permissions to manage resources only in the namespace where it is deployed.

1.1. INSTALLING A USER-DEFINED ARGO CD INSTANCE

To manage cluster configurations or deploy applications, you can install and deploy a new user-defined Argo CD instance.

Prerequisites

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

Procedure

1. Log in to the OpenShift Container Platform web console.
2. In the **Administrator** perspective of the web console, click **Operators → Installed Operators**.
3. Create or select the project where you want to install the user-defined Argo CD instance from the **Project** list.
4. Select **Red Hat OpenShift GitOps** from the installed Operators list and click the **Argo CD** tab.
5. Click **Create ArgoCD** to configure the parameters:
    a. Enter the **Name** of the instance. By default, the **Name** is set to **example**.
    b. Create an external OS Route to access Argo CD server. Click **Server → Route** and check **Enabled**.

TIP

You can alternatively configure YAML to create an external OS Route as shown in the following example:

Example Argo CD with external OS route created

```
apiVersion: argoproj.io/v1beta1
kind: ArgoCD
metadata:
  name: example
  namespace: openshift-gitops
spec:
  server:
    route:
      enabled: true
```

c. Optional: Modify the route TLS termination policy by configuring the **.spec.server.route.tls** field of the Argo CD CR.

6. Click **Create**.

7. Go to **Networking → Routes → <instance_name>-server** in the project where the user-defined Argo CD instance is installed.

8. On the **Details** tab, click the Argo CD web UI link under **Route details → Location**. The Argo CD web UI opens in a separate browser window.

9. Optional: To log in with your OpenShift Container Platform credentials, ensure you are a user of the **cluster-admins** group and then select the **LOG IN VIA OPENSHIFT** option in the Argo CD user interface.

NOTE

To be a user of the **cluster-admins** group, use the **oc adm groups new cluster-admins <user>** command, where **<user>** is the default cluster role that you can bind to users and groups cluster-wide or locally.

10. Obtain the password for the user-defined Argo CD instance:

a. Use the navigation panel to go to the **Workloads → Secrets** page.

b. Use the **Project** list and select the namespace where the user-defined Argo CD instance is created.

c. Select the **<argo_CD_instance_name>-cluster** instance to display the password.

d. On the **Details** tab, copy the password under **Data → admin.password**.

11. Use **admin** as the **Username** and the copied password as the **Password** to log in to the Argo CD UI in the new window.

Additional resources

- Configuring the route TLS termination

- Argo CD custom resource properties
- Specification for TLS termination configuration
- Using an Argo CD instance to manage cluster-scoped resources
- Disabling the creation of the default cluster roles for the cluster-scoped instance

1.2. CONFIGURING COMMON CLUSTER ROLES BY SPECIFYING USER-DEFINED CLUSTER ROLES FOR NAMESPACE-SCOPED INSTANCES

As a cluster administrator, when you give an Argo CD access to a namespace by using the **argocd.argoproj.io/managed-by** label, the Argo CD assumes **namespace-admin** privileges. The Red Hat OpenShift GitOps Operator then automatically creates role bindings for all managed namespaces of the following GitOps control plane components:

- Argo CD Application Controller
- Argo CD server
- Argo CD ApplicationSet Controller

When you provide namespaces to non-administrator users, for example, development teams, they can use the **namespace-admin** privileges to modify objects such as network policies. Installing an Argo CD instance in these namespaces gives the development teams **admin** privileges and indirectly elevates their assigned privileges. These roles are highly privileged and can delete all resources. As a preventive action, you can define a specific set of reduced permissions to meet your security requirements by configuring common cluster roles for all managed namespaces in the role bindings that the Operator creates for the Argo CD Application Controller and Argo CD server components.

To configure common cluster roles for all managed namespaces, you can specify user-defined cluster roles for the **CONTROLLER_CLUSTER_ROLE** and **SERVER_CLUSTER_ROLE** environment variables in the Operator's **Subscription** object YAML file. As a result, instead of creating the default **admin** role, the Operator uses the existing user-defined cluster roles and creates role bindings for all managed namespaces.

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.

Procedure

1. In the **Administrator** perspective, navigate to **Administration → CustomResourceDefinitions**.
2. Find the **Subscription** CRD and click to open it.
3. Select the **Instances** tab and click the **openshift-gitops-operator** subscription.
4. Select the **YAML** tab and make your customization:
    - Specify the user-defined cluster roles for the **CONTROLLER_CLUSTER_ROLE** and **SERVER_CLUSTER_ROLE** environment variables:

Example Subscription

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: openshift-gitops-operator
  namespace: openshift-gitops-operator
spec:
  config:
    env:
    - name: CONTROLLER_CLUSTER_ROLE
      value: gitops-controller-role 1
    - name: SERVER_CLUSTER_ROLE
      value: gitops-server-role 2
```

1. The name of the environment variable for the Argo CD Application Controller component.
2. The name of the environment variable for the Argo CD server component.

TIP

Alternatively, you can inject the preceding environment variables directly into the Operator's **Deployment** object YAML file.

Additional resources

- Customizing permissions by creating user-defined cluster roles for cluster-scoped instances

1.3. ENABLING REPLICAS FOR ARGO CD SERVER AND REPO SERVER

Argo CD-server and Argo CD-repo-server workloads are stateless. To better distribute your workloads among pods, you can increase the number of Argo CD-server and Argo CD-repo-server replicas. However, if a horizontal autoscaler is enabled on the Argo CD-server, it overrides the number of replicas you set.

Procedure

- Set the **replicas** parameters for the **repo** and **server** spec to the number of replicas you want to run:

Example Argo CD custom resource

```
apiVersion: argoproj.io/v1beta1
kind: ArgoCD
metadata:
  name: example-argocd
  labels:
    example: repo
spec:
  repo:
    replicas: <number_of_replicas>
  server:
```

```
replicas: <number_of_replicas>
route:
  enabled: true
  path: /
  tls:
    insecureEdgeTerminationPolicy: Redirect
    termination: passthrough
  wildcardPolicy: None
```

1.4. DEPLOYING RESOURCES TO A DIFFERENT NAMESPACE

To allow Argo CD to manage resources in other namespaces apart from where it is installed, configure the target namespace with a **argocd.argoproj.io/managed-by** label.

Procedure

- Configure the namespace:

```
$ oc label namespace <namespace> \
argocd.argoproj.io/managed-by=<namespace> 1
```

1 The namespace where Argo CD is installed.

1.5. CUSTOMIZING THE ARGO CD CONSOLE LINK

In a multi-tenant cluster, users might have to deal with multiple instances of Argo CD. For example, after installing an Argo CD instance in your namespace, you might find a different Argo CD instance attached to the Argo CD console link, instead of your own Argo CD instance, in the Console Application Launcher.

You can customize the Argo CD console link by setting the **DISABLE_DEFAULT_ARGOCD_CONSOLELINK** environment variable:

- When you set **DISABLE_DEFAULT_ARGOCD_CONSOLELINK** to **true**, the Argo CD console link is permanently deleted.
- When you set **DISABLE_DEFAULT_ARGOCD_CONSOLELINK** to **false** or use the default value, the Argo CD console link is temporarily deleted and visible again when the Argo CD route is reconciled.

Prerequisites

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

Procedure

1. In the **Administrator** perspective, navigate to **Administration → CustomResourceDefinitions**.
2. Find the **Subscription** CRD and click to open it.
3. Select the **Instances** tab and click the **openshift-gitops-operator** subscription.

4. Select the **YAML** tab and make your customization:

   - To enable or disable the Argo CD console link, edit the value of **DISABLE_DEFAULT_ARGOCD_CONSOLELINK** as needed:

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: openshift-gitops-operator
spec:
  config:
    env:
    - name: DISABLE_DEFAULT_ARGOCD_CONSOLELINK
      value: 'true'
```

CHAPTER 2. ARGO CD CUSTOM RESOURCE AND COMPONENT PROPERTIES

The **ArgoCD** custom resource is a Kubernetes Custom Resource (CRD) that describes the desired state for a given Argo CD cluster that allows you to configure the components which make up an Argo CD cluster.

2.1. ARGO CD CUSTOM RESOURCE PROPERTIES

The Argo CD Custom Resource consists of the following properties:

Name	Description	Default	Properties
applicationInstanceLabelKey	The **metadata.label** key name where Argo CD injects the app name as a tracking label.	**app.kubernetes.io/instance**	
applicationSet	**applicationSet** controller configuration options.	***<object>***	• **<image>** - The container image for the **applicationSet** controller. This overrides the **ARGOCD_APPLICATIONSET_IMAGE** environment variable. • **<version>** - The tag to use with the **applicationSet** container image. • **<resources>** - The container compute resources. • **<logLevel>** - The log level used by the Argo CD Application Controller component. Valid options are **debug**, **info**, **error**, and **warn**. • **<logFormat>** - The log format used by

Name	Description	Default	Properties
			the Argo CD Application Controller component. Valid options are **text** or **json**. • **<parallelismLimit>** - The kubectl parallelism limit to set for the controller (**--kubectl-parallelism-limit** flag). • **sourceNamespaces** - The list of non-control plane namespaces for creating and managing Argo CD **ApplicationSet** resources in target namespaces. • **scmProviders** - The list of URLs of the allowed Source Code Manager (SCM) Providers.
configManagementPlugins	Add a configuration management plugin.	***<empty>***	
controller	Argo CD Application Controller options.	***<object>***	• **<processors.operation>** - The number of operation processors. • **<processors.status>** - The number of status processors. • **<resources>** - The container compute resources. • **<logLevel>** - The log level used by the Argo CD

Name	Description	Default	Properties
			Application controller component. Valid options are **debug**, **info**, **error**, and **warn**. • **<appSync>** - AppSync is used to control the sync frequency of Argo CD applications. • **<sharding.enabled>** - Enable sharding on the Argo CD Application Controller component. Use this property to manage a large number of clusters and relieve memory pressure on the controller component. • **<sharding.replicas>** - The number of replicas that are used to support sharding of the Argo CD Application Controller. • **<sharding.dynamicScalingEnabled>** - Enable dynamic scaling of the Argo CD Application Controller component. Use this property to allow the Operator to scale the number of replicas based on the number of clusters the controller component is managing

Name	Description	Default	Properties
			presently. Setting this property to **true** overrides the configuration of the **sharding.enabled** and **sharding.replicas** properties. • **<sharding.minShards>** - The minimum number of Argo CD Application Controller replicas. • **<sharding.maxShards>** - The maximum number of Argo CD Application Controller replicas. • **<sharding.clusterPerShard>** - The number of clusters that need to be managed by each shard. When the replica count reaches the **maxShards**, the shards manage more than one cluster. • **<env>** - Environment to set for the application controller workloads. • **sourceNamespaces** - The list of non-control plane namespaces for creating and managing Argo CD **Application** resources in target namespaces.

Name	Description	Default	Properties
disableAdmin	Disables the built-in admin user.	**false**	
gaTrackingID	Use a Google Analytics tracking ID.	***<empty>***	
gaAnonymizeUsers	Enable hashed usernames sent to google analytics.	**false**	
ha	High availablity options.	***<object>***	• **<enabled>** - Toggle high availability support globally for Argo CD. • **<redisProxyImage>** - The Redis HAProxy container image. This overrides the **ARGOCD_REDIS_HA_PROXY_IMAGE** environment variable. • **<redisProxyVersion>** - The tag to use for the Redis HAProxy container image.
helpChatURL	URL for getting chat help (this is typically your Slack channel for support).	**https://mycorp.slack.com/argo-cd**	
helpChatText	The text that appears in a text box for getting chat help.	**Chat now!**	
image	The container image for all Argo CD components. This overrides the **ARGOCD_IMAGE** environment variable.	**argoproj/argocd**	

Name	Description	Default	Properties
ingress	Ingress configuration options.	***<object>***	
initialRepositories	Initial Git repositories to configure Argo CD to use upon creation of the cluster.	***<empty>***	
notifications	Notifications controller configuration options.	***<object>***	• **<enabled>** - The toggle to start the notifications-controller. • **<image>** - The container image for all Argo CD components. This overrides the **ARGOCD_IMAGE** environment variable. • **<version>** - The tag to use with the Notifications container image. • **<resources>** - The container compute resources. • **<logLevel>** - The log level used by the Argo CD Application Controller component. Valid options are **debug**, **info**, **error**, and **warn**.
repositoryCredentials	Git repository credential templates to configure Argo CD to use upon creation of the cluster.	***<empty>***	

Name	Description	Default	Properties
initialSSHKnownHosts	Initial SSH Known Hosts for Argo CD to use upon creation of the cluster.	***<default_Argo_CD_Known_Hosts>***	
kustomizeBuildOptions	The build options and parameters to use with **kustomize build**.	***<empty>***	
oidcConfig	The OIDC configuration as an alternative to Dex.	***<empty>***	
nodePlacement	Add the **nodeSelector** and the **tolerations**.	***<empty>***	
prometheus	Prometheus configuration options.	***<object>***	• **<enabled>** - Toggle Prometheus support globally for Argo CD. • **<host>** - The hostname to use for Ingress or Route resources. • **<ingress>** - Toggles Ingress for Prometheus. • **<route>** - Route configuration options. • **<size>** - The replica count for the Prometheus **StatefulSet**.

Name	Description	Default	Properties
rbac	RBAC configuration options.	***<object>***	• **<defaultPolicy>** - The **policy.default** property in the **argocd-rbac-cm** config map. The name of the default role which Argo CD falls back to when authorizing API requests. • **<policy>** - The **policy.csv** property in the **argocd-rbac-cm** config map. CSV data containing user-defined RBAC policies and role definitions. • **<scopes>** - The scopes property in the **argocd-rbac-cm** config map. Controls which OIDC scopes to examine during RBAC enforcement, in addition to sub scope.

Name	Description	Default	Properties
redis	Redis configuration options.	***<object>***	• **<autotls>** - Use the provider to create the Redis server's TLS certificate. Only the **openshift** value is currently available. • **<disableTLSVerification>** - Defines whether the Redis server should be accessed using strict TLS validation. • **<image>** - The container image for Redis. This overrides the **ARGOCD_REDIS_IMAGE** environment variable. • **<resources>** - The container compute resources. • **<version>** - The tag to use with the Redis container image.
resourceHealthChecks	Customize resource health check behavior.	***<empty>***	
resourceIgnoreDifferences	Customize resource ignore difference behavior.	***<empty>***	
resourceActions	Customize resource action behavior.	***<empty>***	
resourceExclusions	Completely ignore entire classes of resource group.	***<empty>***	

Name	Description	Default	Properties
resourceInclusions	The configuration to identify which resource group/kinds are applied.	***<empty>***	
server	Argo CD Server configuration options.	***<object>***	• **<autoscale>** - Server autoscale configuration options. • **<extraCommandArgs>** - List of arguments added to the existing arguments set by the Operator. • **<grpc>** - gRPC configuration options. • **<host>** - The hostname used for Ingress or Route resources. • **<ingress>** - Ingress configuration for the Argo CD server component. • **<insecure>** - Toggles the insecure flag for Argo CD server. • **<resources>** - The container compute resources. • **<replicas>** - The number of replicas for the Argo CD server. Must be greater than or

Name	Description	Default	Properties
			equal to **0**. If **autoscale** is enabled, **replicas** is ignored. • **<route>** - Route configuration options. • **<service.Type>** - The **serviceType** used for the service resource. • **<logLevel>** - The log level to be used by the Argo CD Server component. Valid options are **debug**, **info**, **error**, and **warn**. • **<logFormat>** - The log format used by the Argo CD Application Controller component. Valid options are **text** or **json**. • **<env>** - Environment to set for the server workloads.

Name	Description	Default	Properties
sso	Single Sign-on options.	***<object>***	• **<keycloak>** - Configuration options for Keycloak SSO provider. • **<dex>** - Configuration options for Dex SSO provider. • **<provider>** - The name of the provider used to configure Single Sign-on. Currently, the supported options are Dex and Keycloak.
statusBadgeEnabled	Enable application status badge.	**true**	
tls	TLS configuration options.	***<object>***	• **<ca.configMapName>** - The name of the **ConfigMap** which contains the CA certificate. • **<ca.secretName>** - The name of the secret which contains the CA certificate and key. • **<initialCerts>** - Initial set of certificates in the **argocd-tls-certs-cm** config map for connecting Git repositories through HTTPS.
usersAnonymousEnabled	Enable anonymous user access.	**true**	

Name	Description	Default	Properties
version	The tag to use with the container image for all Argo CD components.	Latest Argo CD version	
banner	Add a UI banner message.	***<object>***	• **<banner.content>** - The banner message content (required if a banner is displayed). • **<banner.url>** - The banner message link URL (optional).

2.2. REPO SERVER PROPERTIES

The following properties are available for configuring the Repo server component:

Name	Default	Description
resources	***<empty>***	The container compute resources.
mountsatoken	**false**	Defines whether the **serviceaccount** token should be mounted to the repo-server pod.
serviceaccount	**""**	The name of the **serviceaccount** to use with the repo-server pod.
verifytls	**false**	Defines whether to enforce strict TLS checking on all components when communicating with repo server.
autotls	**""**	Provider to use for setting up TLS for the repo-server's gRPC TLS certificate. Currently, only the **openshift** value is acceptable.

Name	Default	Description
image	**argoproj/argocd**	The container image for Argo CD Repo server. This overrides the **ARGOCD_REPOSERVER_IMAGE** environment variable.
version	same as **.spec.Version**	The tag to use with the Argo CD Repo server.
logLevel	**info**	The log level used by the Argo CD Repo server. Valid options are **debug**, **info**, **error**, and **warn**.
logFormat	**text**	The log format to be used by the Argo CD Repo server. Valid options are **text** or **json**.
execTimeout	**180**	Execution timeout in seconds for rendering tools (for example Helm or Kustomize).
env	***<empty>***	Environment to set for the repository server workloads.
replicas	***<empty>***	The number of replicas for the Argo CD Repo server. Must be greater than or equal to **0**.

2.3. ENABLING NOTIFICATIONS WITH AN ARGO CD INSTANCE

Argo CD notifications allow you to send notifications to external services when events occur in your Argo CD instance. For example, you can send notifications to Slack or email when a sync operation fails. By default, notifications are disabled in Argo CD instances.

Prerequisites

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

Procedure

To enable notifications for an Argo CD instance using the OpenShift Container Platform web console, complete the following steps:

1. Navigate to the **Operators → Installed Operators** page.
2. From the list of **Installed Operators**, select the Red Hat OpenShift GitOps Operator, and then click on the **ArgoCD** tab.

3. Select the Argo CD instance name you want to enable notifications. For example, **openshift-gitops**.

4. Click on the **YAML** tab, and then edit and set the **spec.notifications.enabled** parameter to **true**:

   Example

```
apiVersion: argoproj.io/v1beta1
kind: ArgoCD
metadata:
  name: openshift-gitops
spec:
  notifications:
    enabled: true
#....
```

5. Click **Save**.

TIP

Alternatively, you can enable notifications by using the **oc patch** command in the Openshift CLI. For example:

```
oc patch argocd openshift-gitops -n openshift-gitops --type merge --patch '{"spec": {"notifications": {"enabled": true}}}'
```

Additional resources

- Argo CD notifications controller.

2.4. NOTIFICATIONSCONFIGURATION CUSTOM RESOURCE PROPERTIES

The **NotificationsConfiguration** resource is a Kubernetes custom resource (CR) that manages notifications in a Kubernetes cluster. In Red Hat OpenShift GitOps, you can add templates, triggers, services, and subscription resources to an Argo CD **Notifications** config map by using the **NotificationsConfiguration** CR.

When you create a cluster in Red Hat OpenShift GitOps with notifications enabled, a **NotificationsConfiguration** CR is created by default with the name **default-notifications-configuration**.

Any change made in the existing configuration of the **NotificationsConfiguration** CR is replicated in the Argo CD **Notifications** config map. For example, if the user adds trigger configuration in the **NotificationsConfiguration** resource, this configuration is read, processed, and updated in the Argo CD **Notifications** config map.

IMPORTANT

Any configuration changes must be updated in the **default-notifications-configuration** CR. Custom resources created by the users for **NotificationsConfiguration** resource are not supported.

Any modification to the Argo CD **argocd-notifications-cm** config map is overridden by the changes made in the **NotificationsConfiguration** CR.

Table 2.1. **NotificationsConfiguration** custom resource properties

Properties	Default	Description
Templates	***<empty>***	Templates are used to generate the notification template message.
Triggers	***<empty>***	Triggers are used to define the condition when a notification is sent to the user and the list of templates required to generate the message.
Services	***<empty>***	Services are used to deliver a message.
Subscriptions	***<empty>***	Subscriptions contain centrally-managed global application subscriptions.

The following examples define how to add templates, triggers, services, and subscription resources to the Argo CD **argocd-notification-cm** config map by using the **default-notifications-configuration** custom resource.

Example for templates

```
apiVersion: argoproj.io/v1alpha1
kind: NotificationsConfiguration
metadata:
 name: default-notifications-configuration (1)
spec:
 templates:
  template.my-custom-template: | (2)
    message: |
     Application details: {{.context.argocdUrl}}/applications/{{.app.metadata.name}}.
```

(1) Default name of the **NotificationsConfiguration** CR in a cluster.

(2) An example custom template configuration for the **NotificationsConfiguration** CR.

Example for triggers

-

```
apiVersion: argoproj.io/v1alpha1
kind: NotificationsConfiguration
metadata:
 name: default-notifications-configuration (1)
spec:
 triggers:
  trigger.on-sync-status-unknown: | (2)
   - when: app.status.sync.status == 'Unknown'
   send: [my-custom-template]
```

1. Default name of the **NotificationsConfiguration** CR in a cluster.
2. An example custom trigger configuration for the **NotificationsConfiguration** CR.

Example for services

```
apiVersion: argoproj.io/v1alpha1
kind: NotificationsConfiguration
metadata:
 name: default-notifications-configuration (1)
spec:
 Services:
  service.slack: |
   token: $slack-token (2)
   username: <override-username> # optional username
   icon: <override-icon> # optional icon for the message (supports both emoji and url notation)
```

1. Default name of the **NotificationsConfiguration** CR in a cluster.
2. An example custom service configuration for the **NotificationsConfiguration** CR.

Example for subscriptions

```
apiVersion: argoproj.io/v1alpha1
kind: NotificationsConfiguration
metadata:
 name: default-notifications-configuration (1)
spec:
 Subscriptions: |
  subscriptions: | (2)
   # subscription for on-sync-status-unknown trigger notifications
   - recipients:
   - slack:test2
   - email:test@gmail.com
   triggers:
   - on-sync-status-unknown
   # subscription restricted to applications with matching labels only
   - recipients:
   - slack:test3
   selector: test=true
```

```
    triggers:
    - on-sync-status-unknown
icon: <override-icon> # optional icon for the message (supports both emoji and url notation)
```

1. Default name of the **NotificationsConfiguration** CR in a cluster.
2. An example custom subscription configuration for the **NotificationsConfiguration** CR.

You can configure the **NotificationsConfiguration** CR by using the OpenShift Container Platform web console or the CLI (**oc**).

2.4.1. Configuring the NotificationsConfiguration CR by using the web console

You can configure the **NotificationsConfiguration** custom resource (CR) by using the web console.

Prerequisites

- You have access to an OpenShift Container Platform cluster with **cluster-admin** privileges and are logged into the web console.
- You have installed the Red Hat OpenShift GitOps Operator on your cluster.
- You have enabled notifications for the Argo CD instance. For more information, see "Enabling notifications with an Argo CD instance".

Procedure

1. In the **Administrator** perspective of the OpenShift Container Platform web console, expand **Operators → Installed Operators**.
2. From the list of **Installed Operators**, select the Red Hat OpenShift GitOps Operator, and then click on the **NotificationsConfiguration** tab.
3. On the **NotificationsConfigurations** page, click **default-notifications-configuration**.
4. On the **default-notifications-configuration** page, click **YAML** and add the configuration for any supported resources such as **templates**, **triggers**, **services**, and **subscriptions**. For example, under **templates** in the code, add the following sample configuration:

   Example template configuration

   ```
   template.my-custom-template: |
     message: |
     Application details: {{.context.argocdUrl}}/applications/{{.app.metadata.name}}.
   ```

5. Click **Save**.
6. Verify that the configuration changes made in the **NotificationsConfiguration** CR are reflected in the **argocd-notifications-cm** config map:
   a. Go to **Workloads → ConfigMaps**.
   b. Click **argocd-notifications-cm** and select the **YAML** tab.

c. Scroll through the page in the **YAML** tab to verify the sample configuration added for the supported resources.

2.4.2. Configuring the NotificationsConfiguration CR by using the CLI

You can configure the **NotificationsConfiguration** custom resource (CR) by using the CLI (**oc**).

Prerequisites

- You have access to an OpenShift Container Platform cluster with **cluster-admin** privileges.
- You have installed the Red Hat OpenShift GitOps Operator on your cluster.
- You have enabled notifications for the Argo CD instance. For more information, see "Enabling notifications with an Argo CD instance".

Procedure

1. Edit the default **NotificationsConfiguration** CR in the cluster by running the following command:

```
$ oc edit notificationsconfiguration default-notifications-configuration -n <namespace>
```

where:

default-notifications-configuration
Specifies the name of the default **NotificationsConfiguration** CR.

<namespace>
Specifies the name of the namespace.

2. Under the **templates** section of the CR, add a configuration similar to the following example:

Example template configuration

```
template.my-custom-template: |
  message: |
  Application details: {{.context.argocdUrl}}/applications/{{.app.metadata.name}}.
```

3. Verify the contents of the **argocd-notifications-cm** config map by running the following command:

```
$ oc edit cm argocd-notifications-cm -n <namespace>
```

The changes made in the existing configuration of the **NotificationsConfiguration** CR are reflected in the **argocd-notifications-cm** config map.

2.5. ADDITIONAL RESOURCES

- Installing a user-defined Argo CD instance