Red Hat OpenShift GitOps 1.12

Argo Rollouts

Using Argo Rollouts for progressive delivery

Last Updated: 2024-06-21

Red Hat OpenShift GitOps 1.12 Argo Rollouts

Using Argo Rollouts for progressive delivery

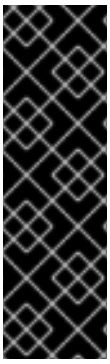
Legal Notice

Abstract

This document provides instructions for using Argo Rollouts to encapsulate all the required definitions for a declarative rollout strategy. It also discusses how to manage and automate progressive delivery of deployments as part of the GitOps workflow.

Table of Contents

CHAPTER 1. USING ARGO ROLLOUTS FOR PROGRESSIVE DEPLOYMENT DELIVERY



IMPORTANT

Argo Rollouts 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 .

Progressive delivery is the process of releasing product updates in a controlled and gradual manner. Progressive delivery reduces the risk of a release by exposing the new version of a product update only to a subset of users initially. The process involves continuously observing and analyzing this new version to verify whether its behavior matches the requirements and expectations set. The verifications continue as the process gradually exposes the product update to a broader and wider audience.

OpenShift Container Platform provides some progressive delivery capability by using routes to split traffic between different services, but this typically requires manual intervention and management.

With Argo Rollouts, you can use automation and metric analysis to support progressive deployment delivery and drive the automated rollout or rollback of a new version of an application. Argo Rollouts provide advanced deployment capabilities and enable integration with ingress controllers and service meshes. You can use Argo Rollouts to manage multiple replica sets that represent different versions of the deployed application. Depending on your deployment strategy, you can handle traffic to these versions during an update by optimizing their existing traffic shaping abilities and gradually shifting traffic to the new version. You can combine Argo Rollouts with a metric provider like Prometheus to do metric-based and policy-driven rollouts and rollbacks based on the parameters set.

1.1. PREREQUISITES

- You have access to the cluster with **cluster-admin** privileges.
- You have access to the OpenShift Container Platform web console.
- Red Hat OpenShift GitOps 1.9.0 or a newer version is installed on your cluster.

1.2. BENEFITS OF ARGO ROLLOUTS

Managing and coordinating advanced deployment strategies in traditional infrastructure often involves long maintenance windows. Automation with tools like OpenShift Container Platform and Red Hat OpenShift GitOps can reduce these windows, but setting up these strategies can still be challenging. With Argo Rollouts, you simplify this process by allowing application teams to define their rollout strategy declaratively. Teams no longer need to define multiple deployments and services or create automation for traffic shaping and integration of tests. Using Argo Rollouts, you can encapsulate all the required definitions for a declarative rollout strategy, automate and manage the process.

Using Argo Rollouts as a default workload in Red Hat OpenShift GitOps provides the following benefits:

- Automated progressive delivery as part of the GitOps workflow

- Advanced deployment capabilities
- Optimize the existing advanced deployment strategies such as blue-green or canary
- Zero downtime updates for deployments
- Fine-grained, weighted traffic shifting
- Able to test without any new traffic hitting the production environment
- Automated rollbacks and promotions
- Manual judgment
- Customizable metric queries and analysis of business key performance indicators (KPIs)
- Integration with ingress controller and Red Hat OpenShift Service Mesh for advanced traffic routing
- Integration with metric providers for deployment strategy analysis
- Usage of multiple providers

With Argo Rollouts, users can more easily adopt progressive delivery in end-user environments. This provides structure and guidelines without requiring teams to learn about traffic managers and complex infrastructure. With automated rollouts, the Red Hat OpenShift GitOps Operator provides security to your end-user environments and helps manage the resources, cost, and time effectively. Existing users who use Argo CD with security and automated deployments get feedback early in the process and avoid problems that impact them.

1.3. ABOUT ROLLOUTMANAGER CUSTOM RESOURCES AND SPECIFICATION

To use Argo Rollouts, you must install Red Hat OpenShift GitOps Operator on the cluster, and then create and submit a **RolloutManager** custom resource (CR) to the Operator in the namespace of your choice. You can scope the **RolloutManager** CR for single or multiple namespaces. The Operator creates an **argo-rollouts** instance with the following namespace-scoped supporting resources:

- Argo Rollouts controller
- Argo Rollouts metrics service
- Argo Rollouts service account
- Argo Rollouts roles
- Argo Rollouts role bindings
- Argo Rollouts secret

You can specify the command arguments, environment variables, a custom image name, and so on for the Argo Rollouts controller resource in the spec of the **RolloutsManager** CR. The **RolloutManager** CR spec defines the desired state of Argo Rollouts.

Example: RolloutManager CR

```
apiVersion: argoproj.io/v1alpha1
kind: RolloutManager
metadata:
  name: argo-rollout
  labels:
    example: basic
spec: {}
```

1.3.1. Argo Rollouts controller

With the Argo Rollouts controller resource, you can manage the progressive application delivery in your namespace. The Argo Rollouts controller resource monitors the cluster for events, and reacts whenever there is a change in any resource related to Argo Rollouts. The controller reads all the rollout details and brings the cluster to the same state as described in the rollout definition.

1.4. CREATING A ROLLOUTMANAGER CUSTOM RESOURCE

To manage progressive delivery of deployments by using Argo Rollouts in Red Hat OpenShift GitOps, you must create and configure a **RolloutManager** custom resource (CR) in the namespace of your choice. By default, any new **argo-rollouts** instance has permission to manage resources only in the namespace where it is deployed, but you can use Argo Rollouts in multiple namespaces as required.

Prerequisites

- Red Hat OpenShift GitOps 1.9.0 or a newer version is installed on your cluster.

Procedure

1. Log in to the OpenShift Container Platform web console as a cluster administrator.
2. In the **Administrator** perspective, click **Operators → Installed Operators**.
3. Create or select the project where you want to create and configure a **RolloutManager** custom resource (CR) from the **Project** drop-down menu.
4. Select **OpenShift GitOps Operator** from the installed operators.
5. In the **Details** tab, under the **Provided APIs** section, click **Create instance** in the **RolloutManager** pane.
6. On the **Create RolloutManager** page, select the **YAML view** and use the default YAML or edit it according to your requirements:

   Example: RolloutManager CR

   ```
   apiVersion: argoproj.io/v1alpha1
   kind: RolloutManager
   metadata:
     name: argo-rollout
     labels:
       example: basic
   spec: {}
   ```

7. Click **Create**.

8. In the **RolloutManager** tab, under the **RolloutManagers** section, verify that the **Status** field of the RolloutManager instance shows as **Phase: Available**.

9. In the left navigation pane, verify the creation of the namespace-scoped supporting resources:

    - Click **Workloads → Deployments** to verify that the **argo-rollouts** deployment is available with the **Status** showing as **1 of 1 pods** running.

    - Click **Workloads → Secrets** to verify that the **argo-rollouts-notification-secret** secret is available.

    - Click **Networking → Services** to verify that the **argo-rollouts-metrics** service is available.

    - Click **User Management → Roles** to verify that the **argo-rollouts** role and **argo-rollouts-aggregate-to-admin**, **argo-rollouts-aggregate-to-edit**, and **argo-rollouts-aggregate-to-view** cluster roles are available.

    - Click **User Management → RoleBindings** to verify that the **argo-rollouts** role binding is available.

1.5. DELETING A ROLLOUTMANAGER CUSTOM RESOURCE

Uninstalling the Red Hat OpenShift GitOps Operator does not remove the resources that were created during installation. You must manually delete the **RolloutManager** custom resource (CR) before you uninstall the Red Hat OpenShift GitOps Operator.

Prerequisites

- Red Hat OpenShift GitOps 1.9.0 or a newer version is installed on your cluster.
- A **RolloutManager** CR exists in your namespace.

Procedure

1. Log in to the OpenShift Container Platform web console as a cluster administrator.
2. In the **Administrator** perspective, click **Operators → Installed Operators**.
3. Click the **Project** drop-down menu and select the project that contains the **RolloutManager** CR.
4. Select **OpenShift GitOps Operator** from the installed operators.
5. Click the **RolloutManager** tab to find RolloutManager instances under the **RolloutManagers** section.
6. Click the instance.
7. Click **Actions → Delete RolloutManager** from the drop-down menu, and click **Delete** to confirm in the dialog box.
8. In the **RolloutManager** tab, under the **RolloutManagers** section, verify that the RolloutManager instance is not available anymore.
9. In the left navigation pane, verify the deletion of the namespace-scoped supporting resources:

    - Click **Workloads → Deployments** to verify that the **argo-rollouts** deployment is deleted.

- Click **Workloads → Secrets** to verify that the **argo-rollouts-notification-secret** secret is deleted.
- Click **Networking → Services** to verify that the **argo-rollouts-metrics** service is deleted.
- Click **User Management → Roles** to verify that the **argo-rollouts** role and **argo-rollouts-aggregate-to-admin**, **argo-rollouts-aggregate-to-edit**, and **argo-rollouts-aggregate-to-view** cluster roles are deleted.
- Click **User Management → RoleBindings** to verify that the **argo-rollouts** role binding is deleted.

1.6. ADDITIONAL RESOURCES

- Installing Red Hat OpenShift GitOps
- Uninstalling Red Hat OpenShift GitOps
- Canary deployments
- Blue-green deployments
- **RolloutManager** Custom Resource specification
- Blue-green and canary deployments with Argo Rollouts
- Argo Rollouts tech preview limitations