



Red Hat OpenShift GitOps 1.11

Observability

Using observability features to view Argo CD logs and monitor the performance and health of Argo CD and application resources

Red Hat OpenShift GitOps 1.11 Observability

Using observability features to view Argo CD logs and monitor the performance and health of Argo CD and application resources

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 details how to use OpenShift Logging with OpenShift GitOps and monitor the performance of Argo CD instances, application health status, and Argo CD custom resource workloads.

Table of Contents

CHAPTER 1. LOGGING	3
1.1. VIEWING ARGO CD LOGS	3
1.1.1. Storing and retrieving Argo CD logs	3
1.1.2. Additional resources	3
CHAPTER 2. MONITORING	5
2.1. MONITORING WITH GITOPS DASHBOARDS	5
2.1.1. Accessing GitOps monitoring dashboards	5
2.2. MONITORING ARGO CD INSTANCES	5
2.2.1. Prerequisites	6
2.2.2. Monitoring Argo CD health using Prometheus metrics	6
2.3. MONITORING THE GITOPS OPERATOR PERFORMANCE	6
2.3.1. Accessing the GitOps Operator metrics	7
2.3.2. Additional resources	8
2.4. MONITORING HEALTH INFORMATION FOR APPLICATION RESOURCES AND DEPLOYMENTS	8
2.4.1. Settings for environment labels and annotations	8
Environment labels	9
Environment annotations	9
2.4.2. Checking health information	10
2.5. MONITORING ARGO CD CUSTOM RESOURCE WORKLOADS	10
2.5.1. Prerequisites	11
2.5.2. Enabling Monitoring for Argo CD custom resource workloads	11
2.5.3. Disabling Monitoring for Argo CD custom resource workloads	12
2.5.4. Additional resources	13

CHAPTER 1. LOGGING

1.1. VIEWING ARGO CD LOGS

You can view the Argo CD logs with the logging subsystem for Red Hat OpenShift. The logging subsystem visualizes the logs on a Kibana dashboard. The OpenShift Logging Operator enables logging with Argo CD by default.


1.1.1. Storing and retrieving Argo CD logs

You can use the Kibana dashboard to store and retrieve Argo CD logs.

Prerequisites

- The Red Hat OpenShift GitOps Operator is installed in your cluster.
- The logging subsystem for Red Hat OpenShift is installed with default configuration in your cluster.

Procedure

1. In the OpenShift Container Platform web console, go to the  menu → **Observability** → **Logging** to view the Kibana dashboard.
2. Create an index pattern.
 - a. To display all the indices, define the index pattern as `*`, and click **Next step**.
 - b. Select `@timestamp` for **Time Filter field name**.
 - c. Click **Create index pattern**.
3. In the navigation panel of the Kibana dashboard, click the **Discover** tab.
4. Create a filter to retrieve logs for Argo CD. The following steps create a filter that retrieves logs for all the pods in the **openshift-gitops** namespace:
 - a. Click **Add a filter** \pm .
 - b. Select the `kubernetes.namespace_name` field.
 - c. Select the **is** operator.
 - d. Select the **openshift-gitops** value.
 - e. Click **Save**.
5. Optional: Add additional filters to narrow the search. For example, to retrieve logs for a particular pod, you can create another filter with **kubernetes.pod_name** as the field.
6. View the filtered Argo CD logs in the Kibana dashboard.

1.1.2. Additional resources

- [Installing the logging subsystem for Red Hat OpenShift using the web console](#)

CHAPTER 2. MONITORING

2.1. MONITORING WITH GITOPS DASHBOARDS

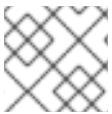
You can access a graphical view of GitOps instances with Red Hat OpenShift GitOps monitoring dashboards to observe the behavior and usage of each instance across the cluster.

There are three GitOps dashboards available:

- **GitOps Overview:** See an overview of all GitOps instances installed on the cluster, including the number of applications, health and sync status, application and sync activity.
- **GitOps Components:** View detailed information, such as CPU or memory, for application-controller, repo-server, server, and other GitOps components.
- **GitOps gRPC Services:** View metrics related to gRPC service activity between the various components in Red Hat OpenShift GitOps.

2.1.1. Accessing GitOps monitoring dashboards

The monitoring dashboards are deployed automatically by the Operator. You can access GitOps monitoring dashboards from the **Administrator** perspective of the OpenShift Container Platform web console.



NOTE

Disabling or changing the content of the dashboards is not supported.

Prerequisites

- You have access to the OpenShift Container Platform web console.
- The Red Hat OpenShift GitOps Operator is installed in the default namespace, **openshift-gitops-operator**.
- The cluster monitoring is enabled on the **openshift-gitops-operator** namespace.
- You have installed an Argo CD application in your defined namespace, for example, **openshift-gitops**.

Procedure

1. In the **Administrator** perspective of the web console, go to **Observe → Dashboards**.
2. From the **Dashboard** drop-down list, select the desired GitOps dashboard: **GitOps (Overview)**, **GitOps / Components**, or **GitOps / gRPC Services**
3. Optional: Choose a specific namespace, cluster, and interval from the **Namespace**, **Cluster**, and **Interval** drop-down lists.
4. View the desired GitOps metrics in the GitOps dashboard.

2.2. MONITORING ARGO CD INSTANCES

By default, the Red Hat OpenShift GitOps Operator automatically detects an installed Argo CD instance in your defined namespace, for example, **openshift-gitops**, and connects it to the monitoring stack of the cluster to provide alerts for out-of-sync applications.

2.2.1. 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 in your cluster.
- You have installed an Argo CD application in your defined namespace, for example, **openshift-gitops**.

2.2.2. Monitoring Argo CD health using Prometheus metrics

You can monitor the health status of an Argo CD application by running Prometheus metrics queries against it.

Procedure

1. In the **Developer** perspective of the web console, select the namespace where your Argo CD application is installed, and navigate to **Observe** → **Metrics**.
2. From the **Select query** drop-down list, select **Custom query**.
3. To check the health status of your Argo CD application, enter the Prometheus Query Language (PromQL) query similar to the following example in the **Expression** field:

Example

```
sum(argocd_app_info{dest_namespace=~"<your_defined_namespace>",health_status!=""})
by (health_status) 1
```

- 1** Replace the **<your_defined_namespace>** variable with the actual name of your defined namespace, for example **openshift-gitops**.

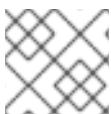
2.3. MONITORING THE GITOPS OPERATOR PERFORMANCE

The Red Hat OpenShift GitOps Operator emits metrics about its performance. With the OpenShift monitoring stack that picks up these metrics, you can monitor and analyze the Operator's performance. The Operator exposes the following metrics, which you can view by using the OpenShift Container Platform web console:

Table 2.1. GitOps Operator performance metrics

Metric name	Type	Description
-------------	------	-------------

Metric name	Type	Description
active_argocd_instances_total	Gauge	The total number of active Argo CD instances currently managed by the Operator across the cluster at a given time.
active_argocd_instances_by_phase	Gauge	The number of active Argo CD instances in a given phase, such as pending, or available.
active_argocd_instance_reconciliation_count	Counter	The total number of reconciliations that have occurred for an instance in a given namespace at any given time.
controller_runtime_reconcile_time_seconds_per_instance_bucket	Counter	The number of reconciliation cycles completed under given time durations for an instance. For example, controller_runtime_reconcile_time_seconds_per_instance_bucket{le="0.5"} shows the number of reconciliations that took under 0.5 seconds to complete for a given instance.
controller_runtime_reconcile_time_seconds_per_instance_count	Counter	The total number of reconciliation cycles observed for a given instance.
controller_runtime_reconcile_time_seconds_per_instance_sum	Counter	The total amount of time taken for the observed reconciliations for a given instance.



NOTE

Gauge is a value that can go up or down. Counter is a value that can only go up.

2.3.1. Accessing the GitOps Operator metrics

You can access the Operator metrics from the **Administrator** perspective of the OpenShift Container Platform web console to track the performance of the Operator.

Prerequisites

- You have access to the OpenShift Container Platform web console.
- The Red Hat OpenShift GitOps Operator is installed in the default **openshift-gitops-operator** namespace.
- The cluster monitoring is enabled on the **openshift-gitops-operator** namespace.

Procedure

1. In the **Administrator** perspective of the web console, go to **Observe** → **Metrics**.
2. Enter the metric in the **Expression** field. You can choose from the following metrics:
 - **active_argocd_instances_total**
 - **active_argocd_instances_by_phase**
 - **active_argocd_instance_reconciliation_count**
 - **controller_runtime_reconcile_time_seconds_per_instance_bucket**
 - **controller_runtime_reconcile_time_seconds_per_instance_count**
 - **controller_runtime_reconcile_time_seconds_per_instance_sum**
3. (Optional): Filter the metric by its properties. For example, filter the **active_argocd_instances_by_phase** metric by the **Available** phase:

Example

```
active_argocd_instances_by_phase{phase="Available"}
```

4. (Optional): Click **Add query** to enter multiple queries.
5. Click **Run queries** to enable and observe the GitOps Operator metrics.

2.3.2. Additional resources

- [Installing Red Hat OpenShift GitOps Operator in web console](#)

2.4. MONITORING HEALTH INFORMATION FOR APPLICATION RESOURCES AND DEPLOYMENTS

The Red Hat OpenShift GitOps **Environments** page in the **Developer** perspective of the OpenShift Container Platform web console shows a list of the successful deployments of the application environments, along with links to the revision for each deployment.

The **Application environments** page in the **Developer** perspective of the OpenShift Container Platform web console displays the health status of the application resources, such as routes, synchronization status, deployment configuration, and deployment history.

The environments pages in the **Developer** perspective of the OpenShift Container Platform web console are decoupled from the Red Hat OpenShift GitOps Application Manager command-line interface (CLI), **kam**. You do not have to use **kam** to generate Application Environment manifests for the environments to show up in the **Developer** perspective of the OpenShift Container Platform web console. You can use your own manifests, but the environments must still be represented by namespaces. In addition, specific labels and annotations are still needed.

2.4.1. Settings for environment labels and annotations

This section provides reference settings for environment labels and annotations required to display an environment application in the **Environments** page, in the **Developer** perspective of the OpenShift Container Platform web console.

Environment labels

The environment application manifest must contain **labels.openshift.gitops/environment** and **destination.namespace** fields. You must set identical values for the `<environment_name>` variable and the name of the environment application manifest.

Specification of the environment application manifest

```
spec:
  labels:
    openshift.gitops/environment: <environment_name>
  destination:
    namespace: <environment_name>
# ...
```

Example of an environment application manifest

```
apiVersion: argoproj.io/v1beta1
kind: Application
metadata:
  name: dev-env 1
  namespace: openshift-gitops
spec:
  labels:
    openshift.gitops/environment: dev-env
  destination:
    namespace: dev-env
# ...
```

- 1** The name of the environment application manifest. The value set is the same as the value of the `<environment_name>` variable.

Environment annotations

The environment namespace manifest must contain the **annotations.app.openshift.io/vcs-uri** and **annotations.app.openshift.io/vcs-ref** fields to specify the version controller code source of the application. You must set identical values for the `<environment_name>` variable and the name of the environment namespace manifest.

Specification of the environment namespace manifest

```
apiVersion: v1
kind: Namespace
metadata:
  annotations:
    app.openshift.io/vcs-uri: <application_source_url>
    app.openshift.io/vcs-ref: <branch_reference>
  name: <environment_name> 1
# ...
```

- 1 The name of the environment namespace manifest. The value set is the same as the value of the `<environment_name>` variable.

Example of an environment namespace manifest

```
apiVersion: v1
kind: Namespace
metadata:
  annotations:
    app.openshift.io/vcs-uri: https://example.com/<your_domain>/<your_gitops.git>
    app.openshift.io/vcs-ref: main
  labels:
    argocd.argoproj.io/managed-by: openshift-gitops
  name: dev-env
# ...
```

2.4.2. Checking health information

The Red Hat OpenShift GitOps Operator will install the GitOps backend service in the **openshift-gitops** namespace.

Prerequisites

- The Red Hat OpenShift GitOps Operator is installed from **OperatorHub**.
- Ensure that your applications are synchronized by Argo CD.

Procedure

1. Click **Environments** under the **Developer** perspective. The **Environments** page shows the list of applications along with their **Environment status**.
2. Hover over the icons under the **Environment status** column to see the synchronization status of all the environments.
3. Click the application name from the list to view the details of a specific application.
4. In the **Application environments** page, if the **Resources** section under the **Overview** tab displays icons, hover over the icons to get status details.
 - A broken heart indicates that resource issues have degraded the application's performance.
 - A yellow yield sign indicates that resource issues have delayed data about the application's health.
5. To view the deployment history of an application, click the **Deployment History** tab. The page includes details such as the **Last deployment**, **Description** (commit message), **Environment**, **Author**, and **Revision**.

2.5. MONITORING ARGO CD CUSTOM RESOURCE WORKLOADS

With Red Hat OpenShift GitOps, you can monitor the availability of Argo CD custom resource workloads for specific Argo CD instances. By monitoring Argo CD custom resource workloads, you have the latest information about the state of your Argo CD instances by enabling alerts for them. When the

component workload pods such as application-controller, repo-server, or server of the corresponding Argo CD instance are unable to come up for certain reasons and there is a drift between the number of ready replicas and the number of desired replicas for a certain period of time, the Operator then triggers the alerts.

You can enable and disable the setting for monitoring Argo CD custom resource workloads.

2.5.1. Prerequisites

- You have access to the cluster as a user with the **cluster-admin** role.
- Red Hat OpenShift GitOps is installed in your cluster.
- The monitoring stack is configured in your cluster in the **openshift-monitoring** project. In addition, the Argo CD instance is in a namespace that you can monitor through Prometheus.
- The **kube-state-metrics** service is running on your cluster.
- Optional: If you are enabling monitoring for an Argo CD instance already present in a user-defined project, ensure that the monitoring is [enabled for user-defined projects](#) in your cluster.



NOTE

If you want to enable monitoring for an Argo CD instance in a namespace that is not watched by the default **openshift-monitoring** stack, for example, any namespace that does not start with **openshift-***, then you must enable user workload monitoring in your cluster. This action enables the monitoring stack to pick up the created PrometheusRule.

2.5.2. Enabling Monitoring for Argo CD custom resource workloads

By default, the monitoring configuration for Argo CD custom resource workloads is set to **false**.

With Red Hat OpenShift GitOps, you can enable workload monitoring for specific Argo CD instances. As a result, the Operator creates a **PrometheusRule** object that contains alert rules for all the workloads managed by the specific Argo CD instances. These alert rules trigger the firing of an alert when the replica count of the corresponding component has drifted from the desired state for a certain amount of time. The Operator will not overwrite the changes made to the **PrometheusRule** object by the users.

Procedure

1. Set the **.spec.monitoring.enabled** field value to **true** on a given Argo CD instance:

Example Argo CD custom resource

```
apiVersion: argoproj.io/v1beta1
kind: ArgoCD
metadata:
  name: example-argocd
  labels:
    example: repo
spec:
  # ...
```

```

monitoring:
  enabled: true
# ...

```

2. Verify whether an alert rule is included in the PrometheusRule created by the Operator:

Example alert rule

```

apiVersion: monitoring.coreos.com/v1
kind: PrometheusRule
metadata:
  name: argocd-component-status-alert
  namespace: openshift-gitops
spec:
  groups:
  - name: ArgoCDComponentStatus
    rules:
    # ...
    - alert: ApplicationSetControllerNotReady 1
      annotations:
        message: >-
          applicationSet controller deployment for Argo CD instance in
          namespace "default" is not running
      expr: >-
        kube_statefulset_status_replicas{statefulset="openshift-gitops-application-controller
statefulset",
        namespace="openshift-gitops"} !=
        kube_statefulset_status_replicas_ready{statefulset="openshift-gitops-application-
controller statefulset",
        namespace="openshift-gitops"}
      for: 1m
      labels:
        severity: critical

```

- 1 Alert rule in the PrometheusRule that checks whether the workloads created by the Argo CD instances are running as expected.

2.5.3. Disabling Monitoring for Argo CD custom resource workloads

You can disable workload monitoring for specific Argo CD instances. Disabling workload monitoring deletes the created PrometheusRule.

Procedure

- Set the **.spec.monitoring.enabled** field value to **false** on a given Argo CD instance:

Example Argo CD custom resource

```

apiVersion: argoproj.io/v1beta1
kind: ArgoCD
metadata:
  name: example-argocd
labels:

```



```
example: repo
spec:
# ...
monitoring:
  enabled: false
# ...
```

2.5.4. Additional resources

- [Enabling monitoring for user-defined projects](#)