Red Hat Developer Hub 1.8

Audit logs in Red Hat Developer Hub

Tracking user activities, system events, and data changes with Red Hat Developer Hub audit logs

Last Updated: 2026-01-22

Red Hat Developer Hub 1.8 Audit logs in Red Hat Developer Hub

Tracking user activities, system events, and data changes with Red Hat Developer Hub audit logs

Legal Notice

Abstract

As a Red Hat Developer Hub (RHDH) administrator, you can track user activities, system events, and data changes with Developer Hub audit logs.

Table of Contents

CHAPTER 1. AUDIT LOGS IN RED HAT DEVELOPER HUB

Audit logs are a chronological set of records documenting the user activities, system events, and data changes that affect your Red Hat Developer Hub users, administrators, or components. Administrators can view Developer Hub audit logs in the OpenShift Container Platform web console to monitor scaffolder events, changes to the RBAC system, and changes to the Catalog database. Audit logs include the following information:

- Name of the audited event
- Actor that triggered the audited event, for example, terminal, port, IP address, or hostname
- Event metadata, for example, date, time
- Event status, for example, **success**, **failure**
- Severity levels, for example, **info**, **debug**, **warn**, **error**

You can use the information in the audit log to achieve the following goals:

Enhance security

Trace activities, including those initiated by automated systems and software templates, back to their source. Know when software templates are executed, and the details of application and component installations, updates, configuration changes, and removals.

Automate compliance

Use streamlined processes to view log data for specified points in time for auditing purposes or continuous compliance maintenance.

Debug issues

Use access records and activity details to fix issues with software templates or plugins.

NOTE

Audit logs are not forwarded to the internal log store by default because this does not provide secure storage. You are responsible for ensuring that the system to which you forward audit logs is compliant with your organizational and governmental regulations, and is properly secured.

1.1. CONFIGURING AUDIT LOGS FOR DEVELOPER HUB ON OPENSHIFT CONTAINER PLATFORM

Use the OpenShift Container Platform web console to configure the following OpenShift Container Platform logging components to use audit logging for Developer Hub:

Logging deployment

Configure the logging environment, including both the CPU and memory limits for each logging component.

Logging collector

Configure the **spec.collection** stanza in the **ClusterLogging** custom resource (CR) to use a supported modification to the log collector and collect logs from **STDOUT**.

Log forwarding

Send logs to specific endpoints inside and outside your OpenShift Container Platform cluster by specifying a combination of outputs and pipelines in a **ClusterLogForwarder** CR.

Additional resources

- Red Hat OpenShift Container Platform - Configuring your Logging deployment
- Red Hat OpenShift Container Platform - Configuring the logging collector
- Red Hat OpenShift Container Platform - Enabling JSON log forwarding
- Red Hat OpenShift Container Platform - Configuring log forwarding

1.2. FORWARDING RED HAT DEVELOPER HUB AUDIT LOGS TO SPLUNK

You can use the Red Hat OpenShift Logging (OpenShift Logging) Operator and a **ClusterLogForwarder** instance to capture the streamed audit logs from a Developer Hub instance and forward them to the HTTPS endpoint associated with your Splunk instance.

Prerequisites

- You have a cluster running on a supported OpenShift Container Platform version.
- You have an account with **cluster-admin** privileges.
- You have a Splunk Cloud account or Splunk Enterprise installation.

Procedure

1. Log in to your OpenShift Container Platform cluster.
2. Install the OpenShift Logging Operator in the **openshift-logging** namespace and switch to the namespace:

   Example command to switch to a namespace

   ```
   $ oc project openshift-logging
   ```

3. Create a **serviceAccount** named **log-collector** and bind the **collect-application-logs** role to the **serviceAccount** :

   Example command to create a serviceAccount

   ```
   $ oc create sa log-collector
   ```

   Example command to bind a role to a serviceAccount

   ```
   $ oc create clusterrolebinding log-collector --clusterrole=collect-application-logs --
   serviceaccount=openshift-logging:log-collector
   ```

4. Generate a **hecToken** in your Splunk instance.

5. Create a key/value secret in the **openshift-logging** namespace and verify the secret:

   Example command to create a key/value secret with hecToken

   ```
   $ oc -n openshift-logging create secret generic splunk-secret --from-literal=hecToken=
   <HEC_Token>
   ```

   Example command to verify a secret

   ```
   $ oc -n openshift-logging get secret/splunk-secret -o yaml
   ```

6. Create a basic `ClusterLogForwarder`resource YAML file as follows:

   Example `ClusterLogForwarder`resource YAML file

   ```
   apiVersion: logging.openshift.io/v1
   kind: ClusterLogForwarder
   metadata:
     name: instance
     namespace: openshift-logging
   ```

   For more information, see Creating a log forwarder.

7. Define the following **ClusterLogForwarder** configuration using OpenShift web console or OpenShift CLI:

   a. Specify the **log-collector** as **serviceAccount** in the YAML file:

      Example serviceAccount configuration

      ```
      serviceAccount:
        name: log-collector
      ```

   b. Configure **inputs** to specify the type and source of logs to forward. The following configuration enables the forwarder to capture logs from all applications in a provided namespace:

      Example inputs configuration

      ```
      inputs:
        - name: my-app-logs-input
          type: application
          application:
            includes:
              - namespace: my-rhdh-project
          containerLimit:
            maxRecordsPerSecond: 100
      ```

      For more information, see Forwarding application logs from specific pods .

   c. Configure outputs to specify where the captured logs are sent. In this step, focus on the **splunk** type. You can either use **tls.insecureSkipVerify** option if the Splunk endpoint uses self-signed TLS certificates (not recommended) or provide the certificate chain using a

Secret.

Example outputs configuration

```
outputs:
  - name: splunk-receiver-application
    type: splunk
    splunk:
      authentication:
        token:
          key: hecToken
          secretName: splunk-secret
      index: main
      url: 'https://my-splunk-instance-link'
      rateLimit:
        maxRecordsPerSecond: 250
```

For more information, see Forwarding logs to Splunk in OpenShift Container Platform documentation.

d. Optional: Filter logs to include only audit logs:

Example filters configuration

```
filters:
  - name: audit-logs-only
    type: drop
    drop:
      - test:
        - field: .message
          notMatches: isAuditEvent
```

For more information, see Filtering logs by content in OpenShift Container Platform documentation.

e. Configure pipelines to route logs from specific inputs to designated outputs. Use the names of the defined inputs and outputs to specify multiple **inputRefs** and **outputRefs** in each pipeline:

Example pipelines configuration

```
pipelines:
  - name: my-app-logs-pipeline
    detectMultilineErrors: true
    inputRefs:
      - my-app-logs-input
    outputRefs:
      - splunk-receiver-application
    filterRefs:
      - audit-logs-only
```

8. Run the following command to apply the **ClusterLogForwarder** configuration:

Example command to apply ClusterLogForwarder configuration

```
$ oc apply -f <ClusterLogForwarder-configuration.yaml>
```

9. Optional: To reduce the risk of log loss, configure your **ClusterLogForwarder** pods using the following options:

   a. Define the resource requests and limits for the log collector as follows:

   Example collector configuration

```
collector:
  resources:
    requests:
      cpu: 250m
      memory: 64Mi
      ephemeral-storage: 250Mi
    limits:
      cpu: 500m
      memory: 128Mi
      ephemeral-storage: 500Mi
```

   b. Define **tuning** options for log delivery, including **delivery**, **compression**, and **RetryDuration**. Tuning can be applied per output as needed.

   Example tuning configuration

```
tuning:
  delivery: AtLeastOnce (1)
  compression: none
  minRetryDuration: 1s
  maxRetryDuration: 10s
```

   (1) **AtLeastOnce** delivery mode means that if the log forwarder crashes or is restarted, any logs that were read before the crash but not sent to their destination are re-sent. It is possible that some logs are duplicated after a crash.

Verification

1. Confirm that logs are being forwarded to your Splunk instance by viewing them in the Splunk dashboard.
2. Troubleshoot any issues using OpenShift Container Platform and Splunk logs as needed.

1.3. VIEWING AUDIT LOGS IN DEVELOPER HUB

Administrators can view, search, filter, and manage the log data from the Red Hat OpenShift Container Platform web console. You can filter audit logs from other log types by using the **isAuditEvent** field.

Prerequisites

- You are logged in as an administrator in the OpenShift Container Platform web console.

Procedure

1. From the **Developer** perspective of the OpenShift Container Platform web console, click the **Topology** tab.
2. From the **Topology** view, click the pod that you want to view audit log data for.
3. From the pod panel, click the **Resources** tab.
4. From the **Pods** section of the **Resources** tab, click **View logs**.
5. From the **Logs** view, enter **isAuditEvent** into the **Search** field to filter audit logs from other log types. You can use the arrows to browse the logs containing the **isAuditEvent** field.

Additional resources

- About Logging in Red Hat OpenShift Container Platform