Red Hat OpenShift Pipelines 1.18

Observability in OpenShift Pipelines

Observability features of OpenShift Pipelines

Last Updated: 2025-05-14

Red Hat OpenShift Pipelines 1.18 Observability in OpenShift Pipelines

Observability features of OpenShift Pipelines

Legal Notice

Abstract

This document provides information about observability features of OpenShift Pipelines.

Table of Contents

CHAPTER 1. USING TEKTON RESULTS FOR OPENSHIFT PIPELINES OBSERVABILITY

Tekton Results is a service that archives the complete information for every pipeline run and task run. You can prune the **PipelineRun** and **TaskRun** resources as necessary and use the Tekton Results API or the **opc** command line utility to access their YAML manifests as well as logging information.

1.1. TEKTON RESULTS CONCEPTS

Tekton Results archives pipeline runs and task runs in the form of results and records.

For every **PipelineRun** and **TaskRun** custom resource (CR) that completes running, Tekton Results creates a *record*.

A *result* can contain one or several records. A record is always a part of exactly one result.

A result corresponds to a pipeline run, and includes the records for the **PipelineRun** CR itself and for all the **TaskRun** CRs that were started as a part of the pipeline run.

If a task run was started directly, without the use of a pipeline run, a result is created for this task run. This result contains the record for the same task run.

Each result has a name that includes the namespace in which the **PipelineRun** or **TaskRun** CR was created and the UUID of the CR. The format for the result name is **<namespace_name>/results/<parent_run_uuid>**. In this format, **<parent_run_uuid>** is the UUUD of a pipeline run or else of a task run that was started directly.

Example result name

```
results-testing/results/04e2fbf2-8653-405f-bc42-a262bcf02bed
```

Each record has a name that includes name of the result that contains the record, as well as the UUID of the **PipelineRun** or **TaskRun** CR to which the record corresponds. The format for the result name is **<namespace_name>/results/<parent_run_uuid>/results/<run_uuid>**.

Example record name

```
results-testing/results/04e2fbf2-8653-405f-bc42-a262bcf02bed/records/e9c736db-5665-441f-922f-7c1d65c9d621
```

The record includes the full YAML manifest of the **TaskRun** or **PipelineRun** CR as it existed after the completion of the run. This manifest contains the specification of the run, any annotation specified for the run, as well as certain information about the results of the run, such as the time when it was completed and whether the run was successful.

While the **TaskRun** or **PipelineRun** CR exists, you can view the YAML manifest by using the following command:

```
$ oc get pipelinerun <cr_name> -o yaml
```

Tekton Results preserves this manifest after the **TaskRun** or **PipelineRun** CR is deleted and makes it available for viewing and searching.

Example YAML manifest of a pipeline run after its completion

```
kind: PipelineRun
spec:
  params:
    - name: message
      value: five
  timeouts:
    pipeline: 1h0m0s
  pipelineRef:
    name: echo-pipeline
  taskRunTemplate:
    serviceAccountName: pipeline
status:
  startTime: "2023-08-07T11:41:40Z"
  conditions:
    - type: Succeeded
      reason: Succeeded
      status: "True"
      message: 'Tasks Completed: 1 (Failed: 0, Cancelled 0), Skipped: 0'
      lastTransitionTime: "2023-08-07T11:41:49Z"
  pipelineSpec:
    tasks:
      - name: echo-task
        params:
          - name: message
            value: five
        taskRef:
          kind: Task
          name: echo-task-pipeline
    params:
      - name: message
        type: string
  completionTime: "2023-08-07T11:41:49Z"
  childReferences:
    - kind: TaskRun
      name: echo-pipeline-run-gmzrx-echo-task
      apiVersion: tekton.dev/v1
      pipelineTaskName: echo-task
metadata:
  uid: 62c3b02e-f12b-416c-9771-c02af518f6d4
  name: echo-pipeline-run-gmzrx
  labels:
    tekton.dev/pipeline: echo-pipeline
  namespace: releasetest-js5tt
  finalizers:
    - chains.tekton.dev/pipelinerun
  generation: 2
  annotations:
    results.tekton.dev/log: releasetest-js5tt/results/62c3b02e-f12b-416c-9771-
c02af518f6d4/logs/c1e49dd8-d641-383e-b708-e3a02b6a4378
    chains.tekton.dev/signed: "true"
    results.tekton.dev/record: releasetest-js5tt/results/62c3b02e-f12b-416c-9771-
c02af518f6d4/records/62c3b02e-f12b-416c-9771-c02af518f6d4
    results.tekton.dev/result: releasetest-js5tt/results/62c3b02e-f12b-416c-9771-c02af518f6d4
  generateName: echo-pipeline-run-
```

```
managedFields:
  - time: "2023-08-07T11:41:39Z"
    manager: kubectl-create
    fieldsV1:
      f:spec:
        .: {}
        f:params: {}
        f:pipelineRef:
          .: {}
          f:name: {}
      f:metadata:
        f:generateName: {}
    operation: Update
    apiVersion: tekton.dev/v1
    fieldsType: FieldsV1
  - time: "2023-08-07T11:41:40Z"
    manager: openshift-pipelines-controller
    fieldsV1:
      f:metadata:
        f:labels:
          .: {}
          f:tekton.dev/pipeline: {}
    operation: Update
    apiVersion: tekton.dev/v1
    fieldsType: FieldsV1
  - time: "2023-08-07T11:41:49Z"
    manager: openshift-pipelines-chains-controller
    fieldsV1:
      f:metadata:
        f:finalizers:
          .: {}
          v:"chains.tekton.dev/pipelinerun": {}
        f:annotations:
          .: {}
          f:chains.tekton.dev/signed: {}
    operation: Update
    apiVersion: tekton.dev/v1
    fieldsType: FieldsV1
  - time: "2023-08-07T11:41:49Z"
    manager: openshift-pipelines-controller
    fieldsV1:
      f:status:
        f:startTime: {}
        f:conditions: {}
        f:pipelineSpec:
          .: {}
          f:tasks: {}
          f:params: {}
        f:completionTime: {}
        f:childReferences: {}
    operation: Update
    apiVersion: tekton.dev/v1
    fieldsType: FieldsV1
    subresource: status
  - time: "2023-08-07T11:42:15Z"
    manager: openshift-pipelines-results-watcher
```

```
      fieldsV1:
        f:metadata:
          f:annotations:
            f:results.tekton.dev/log: {}
            f:results.tekton.dev/record: {}
            f:results.tekton.dev/result: {}
      operation: Update
      apiVersion: tekton.dev/v1
      fieldsType: FieldsV1
  resourceVersion: "126429"
  creationTimestamp: "2023-08-07T11:41:39Z"
  deletionTimestamp: "2023-08-07T11:42:23Z"
  deletionGracePeriodSeconds: 0
apiVersion: tekton.dev/v1
```

You can access every result and record by its name. You can also use Common Expression Language (CEL) queries to search for results and records by the information they contain, including the YAML manifest.

You can also configure Tekton Results to facilitate forwarding the logging information of all the tools that ran as a part of a pipeline or task to LokiStack. You can then query Tekton Results for logging information of the task run associated with a Tekton Results record.

1.2. CONFIGURING TEKTON RESULTS

After you install OpenShift Pipelines, Tekton Results is enabled by default.

However, if you want to store and access logging information for your pipeline runs and task runs, you must configure forwarding this information to LokiStack.

You can optionally complete additional configuration for Tekton Results.

1.2.1. Configuring LokiStack forwarding for logging information

If you want to use Tekton Results to query logging information for task runs, you must install LokiStack and OpenShift Logging on your OpenShift Container Platform cluster and configure forwarding of the logging information to LokiStack.

If you do not configure LokiStack forwarding for logging information, Tekton Results does not store this information or provide it from the command-line interface or API.

Prerequisites

- You installed the OpenShift CLI (**oc**) utility.
- You are logged in to your OpenShift Container Platform cluster as a cluster administrator user.

Procedure

To configure LokiStack forwarding, complete the following steps:

1. On your OpenShift Container Platform cluster, install LokiStack by using the Loki Operator and also install the OpenShift Logging Operator.

2. Create a **ClusterLogForwarder.yaml** manifest file for the **ClusterLogForwarder** custom resource (CR) with one of the following YAML manifests, depending on whether you installed OpenShift Logging version 6 or version 5:

YAML manifest for the ClusterLogForwarder CR if you installed OpenShift Logging version 6

```
apiVersion: observability.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: collector
  namespace: openshift-logging
spec:
  inputs:
  - application:
      selector:
        matchExpressions:
          - key: app.kubernetes.io/managed-by
            operator: In
            values: ["tekton-pipelines", "pipelinesascode.tekton.dev"]
    name: only-tekton
    type: application
  managementState: Managed
  outputs:
  - lokiStack:
      labelKeys:
        application:
          ignoreGlobal: true
          labelKeys:
          - log_type
          - kubernetes.namespace_name
          - openshift_cluster_id
      authentication:
        token:
          from: serviceAccount
      target:
        name: logging-loki
        namespace: openshift-logging
    name: default-lokistack
    tls:
      ca:
        configMapName: openshift-service-ca.crt
        key: service-ca.crt
    type: lokiStack
  pipelines:
  - inputRefs:
    - only-tekton
    name: default-logstore
    outputRefs:
    - default-lokistack
  serviceAccount:
    name: collector
```

YAML manifest for the ClusterLogForwarder CR if you installed OpenShift Logging version 5

```
apiVersion: "logging.openshift.io/v1"
kind: ClusterLogForwarder
metadata:
  name: instance
  namespace: openshift-logging
spec:
  inputs:
  - name: only-tekton
    application:
      selector:
        matchLabels:
          app.kubernetes.io/managed-by: tekton-pipelines
  pipelines:
    - name: enable-default-log-store
      inputRefs: [ only-tekton ]
      outputRefs: [ default ]
```

3. Create the **ClusterLogForwarder** CR in the **openshift-logging** namespace by entering the following command:

```
$ oc apply -n openshift-logging ClusterLogForwarder.yaml
```

4. Edit the **TektonConfig** custom resource (CR) by using the following command:

```
$ oc edit TektonConfig config
```

Make the following changes in the **result** spec:

```
apiVersion: operator.tekton.dev/v1alpha1
kind: TektonConfig
metadata:
  name: config
spec:
  result:
    loki_stack_name: logging-loki (1)
    loki_stack_namespace: openshift-logging (2)
```

(1) The name of the **LokiStack** CR, typically **logging-loki**.

(2) The name of the namespace where LokiStack is deployed, typically **openshift-logging**.

1.2.2. Configuring an external database server

Tekton Results uses a PostgreSQL database to store data. By default, the installation includes an internal PostgreSQL instance. You can configure the installation to use an external PostgreSQL server that already exists in your deployment.

Procedure

1. Create a secret with the credentials for connecting to your PostgreSQL server by entering the following command:

```
$ oc create secret generic tekton-results-postgres \
```

```
  --namespace=openshift-pipelines \
  --from-literal=POSTGRES_USER=<user> \
  --from-literal=POSTGRES_PASSWORD=<password>
```

2. Edit the **TektonConfig** custom resource (CR) by using the following command:

```
$ oc edit TektonConfig config
```

Make the following changes in the **result** spec:

```
apiVersion: operator.tekton.dev/v1alpha1
kind: TektonConfig
metadata:
  name: config
spec:
  result:
    is_external_db: true
    db_host: database.example.com (1)
    db_port: 5342 (2)
```

(1) Provide the host name of your PostgreSQL server.

(2) Provide the port number of your PostgreSQL server.

1.2.3. Configuring the retention policy for Tekton Results

By default, Tekton Results stores pipeline runs, task runs, events, and logs indefinitely. This leads to an unnecesary use of storage resources and can affect your database performance.

You can configure the retention policy for Tekton Results at the cluster level to remove older results and their associated records and logs.

Procedure

- Edit the **TektonConfig** custom resource (CR) by using the following command:

```
$ oc edit TektonConfig config
```

Make the following changes in the **result** spec:

```
apiVersion: operator.tekton.dev/v1alpha1
kind: TektonConfig
metadata:
  name: config
spec:
  result:
    options:
      configMaps:
        config-results-retention-policy:
          data:
            runAt: "3 5 * * 0" (1)
            maxRetention: "30" (2)
```

1. Specify, in cron format, when to run the pruning job in the database. This example runs the job at 5:03 AM every Sunday.
2. Specify how many days to keep the data in the database. This example retains the data for 30 days.

1.3. QUERYING TEKTON RESULTS USING THE OPC COMMAND LINE UTILITY

You can use the **opc** command line utility to query Tekton Results for results and records. To install the **opc** command line utility, install the package for the **tkn** command line utility. For instructions about installing this package, see Installing tkn.

You can use the names of records and results to retrieve the data in them.

You can search for results and records using Common Expression Language (CEL) queries. These searches display the UUIDs of the results or records. You can use the provided examples to create queries for common search types. You can also use reference information to create other queries.

1.3.1. Preparing the opc utility environment for querying Tekton Results

Before you can query Tekton Results, you must prepare the environment for the **opc** utility.

Prerequisites

- You installed the **opc** utility.

Procedure

1. Set the **RESULTS_API** environment variable to the route to the Tekton Results API by entering the following command:

```
$ export RESULTS_API=$(oc get route tekton-results-api-service -n openshift-pipelines --no-headers -o custom-columns=":spec.host"):443
```

2. Create an authentication token for the Tekton Results API by entering the following command:

```
$ oc create token <service_account>
```

   Save the string that this command outputs.

3. Optional: Create the **~/.config/tkn/results.yaml** file for automatic authentication with the Tekton Results API. The file must have the following contents:

```
address: <tekton_results_route> 1
token: <authentication_token> 2
ssl:
  roots_file_path: /home/example/cert.pem 3
  server_name_override: tekton-results-api-service.openshift-pipelines.svc.cluster.local 4
 service_account:
  namespace: service_acc_1 5
  name: service_acc_1 6
```

1. The route to the Tekton Results API. Use the same value as you set for **RESULTS_API**.
2. The authentication token that was created by the **oc create token** command. If you provide this token, it overrides the **service_account** setting and **opc** uses this token to authenticate.
3. The location of the file with the SSL certificate that you configured for the API endpoint.
4. If you configured a custom target namespace for OpenShift Pipelines, replace **openshift-pipelines** with the name of this namespace.
5. 6. The name of a service account for authenticating with the Tekton Results API. If you provided the authentication token, you do not need to provide the **service_account** parameters.

Alternatively, if you do not create the ~/**.config/tkn/results.yaml** file, you can pass the token to each **opc** command by using the **--authtoken** option.

1.3.2. Querying for results and records by name

You can list and query results and records using their names.

Prerequisites

- You installed the **opc** utility and prepared its environment to query Tekton Results.
- You installed the **jq** package.
- If you want to query logging information, you configured log forwarding to LokiStack.

Procedure

1. List the names of all results that correspond to pipeline runs and task runs created in a namespace. Enter the following command:

```
$ opc results list --addr ${RESULTS_API} <namespace_name>
```

Example command

```
$ opc results list --addr ${RESULTS_API} results-testing
```

Example output

```
Name                                                                    Start                         Update
results-testing/results/04e2fbf2-8653-405f-bc42-a262bcf02bed  2023-06-29 02:49:53 +0530 IST  2023-06-29 02:50:05 +0530 IST
results-testing/results/ad7eb937-90cc-4510-8380-defe51ad793f  2023-06-29 02:49:38 +0530 IST  2023-06-29 02:50:06 +0530 IST
results-testing/results/d064ce6e-d851-4b4e-8db4-7605a23671e4  2023-06-29 02:49:45 +0530 IST  2023-06-29 02:49:56 +0530 IST
```

2. List the names of all records in a result by entering the following command:

```
$ opc results records list --addr ${RESULTS_API} <result_name>
```

Example command

```
$ opc results records list --addr ${RESULTS_API} results-testing/results/04e2fbf2-8653-405f-bc42-a262bcf02bed
```

Example output

```
Name                                                                                          Type
Start                                   Update
results-testing/results/04e2fbf2-8653-405f-bc42-a262bcf02bed/records/e9c736db-5665-441f-922f-7c1d65c9d621  tekton.dev/v1.TaskRun             2023-06-29 02:49:53 +0530 IST
2023-06-29 02:49:57 +0530 IST
results-testing/results/04e2fbf2-8653-405f-bc42-a262bcf02bed/records/5de23a76-a12b-3a72-8a6a-4f15a3110a3e  results.tekton.dev/v1alpha2.Log        2023-06-29 02:49:57 +0530 IST           2023-06-29 02:49:57 +0530 IST
results-testing/results/04e2fbf2-8653-405f-bc42-a262bcf02bed/records/57ce92f9-9bf8-3a0a-aefb-dc20c3e2862d  results.tekton.dev/v1alpha2.Log        2023-06-29 02:50:05 +0530 IST
2023-06-29 02:50:05 +0530 IST
results-testing/results/04e2fbf2-8653-405f-bc42-a262bcf02bed/records/e9a0c21a-f826-42ab-a9d7-a03bcefed4fd  tekton.dev/v1.TaskRun             2023-06-29 02:49:57 +0530 IST
2023-06-29 02:50:05 +0530 IST
results-testing/results/04e2fbf2-8653-405f-bc42-a262bcf02bed/records/04e2fbf2-8653-405f-bc42-a262bcf02bed  tekton.dev/v1.PipelineRun          2023-06-29 02:49:53 +0530 IST
2023-06-29 02:50:05 +0530 IST
results-testing/results/04e2fbf2-8653-405f-bc42-a262bcf02bed/records/e6eea2f9-ec80-388c-9982-74a018a548e4  results.tekton.dev/v1alpha2.Log        2023-06-29 02:50:05 +0530 IST
2023-06-29 02:50:05 +0530 IST
```

3. Retrieve the YAML manifest for a pipeline run or task run from a record by entering the following command:

```
$ opc results records get --addr ${RESULTS_API} <record_name> \
  | jq -r .data.value | base64 -d | \
  xargs -0 python3 -c 'import sys, yaml, json; j=json.loads(sys.argv[1]);
print(yaml.safe_dump(j))'
```

Example command

```
$ opc results records get --addr ${RESULTS_API} \
  results-testing/results/04e2fbf2-8653-405f-bc42-a262bcf02bed/records/e9c736db-5665-441f-922f-7c1d65c9d621 | \
  jq -r .data.value | base64 -d | \
  xargs -0 python3 -c 'import sys, yaml, json; j=json.loads(sys.argv[1]);
print(yaml.safe_dump(j))'
```

4. Optional: Retrieve the logging information for a task run from a record using the log record name. To get the log record name, replace **records** with **logs** in the record name. Enter the following command:

```
$ opc results logs get --addr ${RESULTS_API} <log_record_name> | jq -r .data | base64 -d
```

Example command

```
$ opc results logs get --addr ${RESULTS_API} \
  results-testing/results/04e2fbf2-8653-405f-bc42-a262bcf02bed/logs/e9c736db-5665-441f-922f-7c1d65c9d621 | \
  jq -r .data | base64 -d
```

1.3.3. Searching for results

You can search for results using Common Expression Language (CEL) queries. For example, you can find results for pipeline runs that did not succeed. However, most of the relevant information is not contained in result objects; to search by the names, completion times, and other data, search for records.

Prerequisites

- You installed the **opc** utility and prepared its environment to query Tekton Results.

Procedure

- Search for results using a CEL query by entering the following command:

```
$ opc results list --addr ${RESULTS_API} --filter="<cel_query>" <namespace-name>
```

Replace **<namespace_name>** with the namespace in which the pipeline runs or task runs were created.

Table 1.1. Example CEL queries for results

Purpose	CEL query
The results of all runs that failed	**!(summary.status == SUCCESS)**
The results all pipeline runs that contained the annotations **ann1** and **ann2**	**summary.annotations.contains('ann1') && summary.annotations.contains('ann2') && summary.type=='PIPELINE_RUN'**

1.3.4. Searching for records

You can search for records using Common Expression Language (CEL) queries. As each record contains full YAML information for a pipeline run or task run, you can find records by many different criteria.

Prerequisites

- You installed the **opc** utility and prepared its environment to query Tekton Results.

Procedure

- Search for records using a CEL query by entering the following command:

```
$ opc results records list --addr ${RESULTS_API} --filter="<cel_query>"
<namespace_name>/result/-
```

-

Replace **<namespace_name>** with the namespace in which the pipeline runs or task runs were created. Alternatively, search for records within a single result by entering the following command:

```
$ opc results records list --addr ${RESULTS_API} --filter="<cel_query>" <result_name>
```

Replace **<result_name>** with the full name of the result.

Table 1.2. Example CEL queries for records

Purpose	CEL query
Records of all task runs or pipeline runs that failed	**!(data.status.conditions[0].status == 'True')**
Records where the name of the **TaskRun** or **PipelineRun** custom resource (CR) was **run1**	**data.metadata.name == 'run1'**
Records for all task runs that were started by the **PipelineRun** CR named **run1**	**data_type == 'TASK_RUN' && data.metadata.labels['tekton.dev/pipelineRun'] == 'run1'**
Records of all pipeline runs and task runs that were created from a **Pipeline** CR named **pipeline1**	**data.metadata.labels['tekton.dev/pipeline'] == 'pipeline1'**
Records of all pipeline runs that were created from a **Pipeline** CR named **pipeline1**	**data.metadata.labels['tekton.dev/pipeline'] == 'pipeline1' && data_type == 'PIPELINE_RUN'**
Records of all task runs where the **TaskRun** CR name stared with **hello**	**data.metadata.name.startsWith('hello') && data_type=='TASK_RUN'**
Records of all pipeline runs that took more than five minutes to complete	**data.status.completionTime - data.status.startTime > duration('5m') && data_type == 'PIPELINE_RUN'**
Records of all pipeline runs and task runs that completed on October 7, 2023	**data.status.completionTime.getDate() == 7 && data.status.completionTime.getMonth() == 10 && data.status.completionTime.getFullYear() == 2023**
Records of all pipeline runs that included three or more tasks	**size(data.status.pipelineSpec.tasks) >= 3 && data_type == 'PIPELINE_RUN'**
Records of all pipeline runs that had annotations containing **ann1**	**data.metadata.annotations.contains('ann1') && data_type == 'PIPELINE_RUN'**
Records of all pipeline runs that had annotations containing **ann1** and the name of the **PipelineRun** CR started with **hello**	**data.metadata.annotations.contains('ann1') && data.metadata.name.startsWith('hello') && data_type == 'PIPELINE_RUN'**

1.3.5. Reference information for searching results

You can use the following fields in Common Expression Language (CEL) queries for results:

Table 1.3. Fields available in CEL queries for results

CEL field	Description
parent	The namespace in which the **PipelineRun** or **TaskRun** custom resource (CR) was created.
uid	Unique identifier for the result.
annotations	Annotations added to the **PipelineRun** or **TaskRun** CR.
summary	The summary of the result.
create_time	The creation time of the result.
update_time	The last update time of the result.

You can use the **summary.status** field to determine whether the pipeline run was successful. This field can have the following values:

- **UNKNOWN**
- **SUCCESS**
- **FAILURE**
- **TIMEOUT**
- **CANCELLED**

NOTE

Do not use quote characters such as **"** or **'** to provide the value for this field.

1.3.6. Reference information for searching records

You can use the following fields in Common Expression Language (CEL) queries for records:

Table 1.4. Fields available in CEL queries for records

CEL field	Description	Values
name	Record name	

CEL field	Description	Values
data_type	Record type identifier	**tekton.dev/v1.TaskRun** or **TASK_RUNtekton.dev/v1.PipelineRun** or **PIPELINE_RUNresults.tekton.dev/v1alpha2.Log**
data	The YAML data for the task run or pipeline run. In log records, this field contains the logging output.	

Because the **data** field contains the entire YAML data for the task run or pipeline run, you can use all elements of this data in your CEL query. For example, **data.status.completionTime** contains the completion time of the task run or pipeline run.

1.4. ADDITIONAL RESOURCES

- Common Expression Language (CEL)