Red Hat OpenShift Pipelines 1.13

Observability in OpenShift Pipelines

Observability features of OpenShift Pipelines

Last Updated: 2023-12-08

Red Hat OpenShift Pipelines 1.13 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.

IMPORTANT

Tekton Results 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 .

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
  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/v1beta1
      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/v1beta1
    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/v1beta1
    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/v1beta1
    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/v1beta1
    fieldsType: FieldsV1
  resourceVersion: "126429"
  creationTimestamp: "2023-08-07T11:41:39Z"
  deletionTimestamp: "2023-08-07T11:42:23Z"
  deletionGracePeriodSeconds: 0
apiVersion: tekton.dev/v1beta1
```

Tekton Results also creates a log record that contains the logging information of all the tools that ran as a part of a pipeline run or task run.

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.

1.2. PREPARING TO INSTALL TEKTON RESULTS

You must complete several preparatory steps before installing Tekton Results.

1.2.1. Preparing a secret with an SSL certificate

Tekton Results provides a REST API using the HTTPS protocol, which requires an SSL certificate. Provide a secret with this certificate. If you have an existing certificate provided by a certificate authority (CA), use this certificate, otherwise create a self-signed certificate.

Prerequisites

- The **openssl** command-line utility is installed.

Procedure

1. If you do not have a certificate provided by a CA, create a self-signed certificate by entering the following command:

```
$ openssl req -x509 \
-newkey rsa:4096 \
-keyout key.pem \
```

```
-out cert.pem \
-days 365 \
-nodes \
-subj "/CN=tekton-results-api-service.openshift-pipelines.svc.cluster.local" \
-addext "subjectAltName = DNS:tekton-results-api-service.openshift-
pipelines.svc.cluster.local"
```

Replace **tekton-results-api-service.openshift-pipelines.svc.cluster.local** with the route endpoint that you plan to use for the Tekton Results API.

2. Create a transport security layer (TLS) secret from the certificate by entering the following command:

```
$ oc create secret tls -n openshift-pipelines tekton-results-tls --cert=cert.pem --key=key.pem
```

If you want to use an existing certificate provided by a CA, replace **cert.pem** with the name of the file containing this certificate.

1.2.2. Preparing a secret with the database credentials

Tekton Results uses a PostgreSQL database to store data. You can configure the installation to use either a PostgreSQL server that is automatically installed with Tekton Results or an external PostgreSQL server that already exists in your deployment. In both cases, provide a secret with the database credentials.

Procedure

Complete one of the following steps:

- If you do not need to use an external PostgreSQL server, create a secret with the database user named **result** and a random password in the **openshift-pipelines** namespace by entering the following command:

```
$ oc create secret generic tekton-results-postgres \
  --namespace=openshift-pipelines \
  --from-literal=POSTGRES_USER=result \
  --from-literal=POSTGRES_PASSWORD=$(openssl rand -base64 20)
```

NOTE

In this command and in subsequent commads, if you configured a custom target namespace for OpenShift Pipelines, use the name of this namespace instead of **openshift-pipelines**.

- If you want to use an external PostgreSQL database server to store Tekton Results data, create a secret with the credentials for this server by entering the following command:

```
$ oc create secret generic tekton-results-postgres \
  --namespace=openshift-pipelines \
  --from-literal=POSTGRES_USER=<user> \ 1
  --from-literal=POSTGRES_PASSWORD=<password> 2
```

Replace **<user>** with the username for the PostgreSQL user that Tekton Results must use. Replace **<password>** with the password for the same account.

1.2.3. Preparing storage for logging information

Tekton Results uses separate storage for logging information related to pipeline runs and task runs. You can configure any one of the following types of storage:

- Persistent volume claim (PVC) on your Red Hat OpenShift Pipelines cluster
- Google Cloud Storage
- S3 bucket storage

Procedure

Complete one of the following procedures:

- To use a PVC, complete the following steps:
  a. Create a file named **pvc.yaml** with the following definition for the PVC:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: tekton-logs
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

  b. Apply the definition by entering the following command:

```
$ oc apply -n openshift-pipelines -f pvc.yaml
```

- To use Google Cloud Storage, complete the following steps:
  a. Create an application credentials file by using the **gcloud** command. For instructions about providing application credentials in a file, see User credentials provided by using the gcloud CLI in the Google Cloud documentation.
  b. Create a secret from the application credentials file by entering the following command:

```
$ oc create secret generic gcs-credentials \
  --from-file=$HOME/.config/gcloud/application_default_credentials.json \
  -n openshift-pipelines
```

  Adjust the path and filename of the application credentials file as necessary.

- To use S3 bucket storage, complete the following steps:
  a. Create a file named **s3_secret.yaml** with the following content:

```
apiVersion: v1
kind: Secret
metadata:
  name: my_custom_secret
```

```
  namespace: tekton-pipelines
type: Opaque
stringData:
  S3_BUCKET_NAME: bucket1 (1)
  S3_ENDPOINT: https://example.localhost.com (2)
  S3_HOSTNAME_IMMUTABLE: "false"
  S3_REGION: region-1 (3)
  S3_ACCESS_KEY_ID: "1234" (4)
  S3_SECRET_ACCESS_KEY: secret_key (5)
  S3_MULTI_PART_SIZE: "5242880"
```

(1) The name of the S3 storage bucket

(2) The S3 API endpoint URL

(3) The S3 region

(4) The S3 access key ID

(5) The S3 secret access key

b. Create a secret from the file by entering the following command:

```
$ oc create secret generic s3-credentials \
  --from-file=s3_secret.yaml -n openshift-pipelines
```

1.3. INSTALLING TEKTON RESULTS

To install Tekton Results, you must provide the required resources and then create and apply a **TektonResult** custom resource (CR). The OpenShift Pipelines Operator installs the Results services when you apply the **TektonResult** custom resource.

Prerequisites

- You installed OpenShift Pipelines using the Operator.
- You prepared a secret with the SSL certificate.
- You prepared storage for the logging information.
- You prepared a secret with the database credentials.

Procedure

1. Create the resource definition file named **result.yaml** based on the following example. You can adjust the settings as necessary.

```
apiVersion: operator.tekton.dev/v1alpha1
kind: TektonResult
metadata:
  name: result
spec:
  targetNamespace: openshift-pipelines
```

```
logs_api: true
log_level: debug
db_port: 5432
db_host: tekton-results-postgres-service.openshift-pipelines.svc.cluster.local
logs_path: /logs
logs_type: File
logs_buffer_size: 32768
auth_disable: true
tls_hostname_override: tekton-results-api-service.openshift-pipelines.svc.cluster.local
db_enable_auto_migration: true
server_port: 8080
prometheus_port: 9090
```

2. Add configuration for the storage for logging information to this file:

   - If you configured a persistent volume claim (PVC), add the following line to provide the name of the PVC:

     ```
     logging_pvc_name: tekton-logs
     ```

   - If you configured Google Cloud Storage, add the following lines to provide the secret name, the credentials file name, and the name of the Google Cloud Storage bucket:

     ```
     gcs_creds_secret_name: gcs-credentials
     gcs_creds_secret_key: application_default_credentials.json (1)
     gcs_bucket_name: bucket-name (2)
     ```

     (1) Provide the name, without the path, of the application credentials file that you used when creating the secret.

     (2) Provide the name of a bucket in Google Cloud Storage. Tekton Chains uses this bucket to store logging information for pipeline runs and task runs.

   - If you configured S3 bucket storage, add the following line to provide the name of the S3 secret:

     ```
     secret_name: s3-credentials
     ```

3. Optional: If you want to use an external PostgreSQL database server to store Tekton Results information, add the following lines to the file:

   ```
   db_host: postgres.internal.example.com (1)
   db_port: 5432 (2)
   is_external_db: true
   ```

   (1) The host name for the PostgreSQL server.

   (2) The port for the PostgreSQL server.

4. Apply the resource definition by entering the following command:

   ```
   $ oc apply -n openshift-pipelines -f result.yaml
   ```

5. Expose the route for the Tekton Results service API by entering the following command:

```
$ oc create route -n openshift-pipelines \
  passthrough tekton-results-api-service \
  --service=tekton-results-api-service --port=8080
```

1.4. 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.

1.4.1. 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 Tekton Results.
- 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 sa <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.

1.4.2. Querying for results and records by name

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

Prerequisites

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

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/v1beta1.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/v1beta1.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/v1beta1.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
```

1.4.3. 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 Tekton Results.
- 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'**

1.4.4. 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 Tekton Results.
- 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**

Purpose	CEL query
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'**

1.4.5. 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.

1.4.6. 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	
data_type	Record type identifier	**tekton.dev/v1beta1.TaskRun** or **TASK_RUNtekton.dev/v1beta1.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.

1.5. ADDITIONAL RESOURCES

- Common Expression Language specification

CHAPTER 2. VIEWING PIPELINE LOGS USING THE OPENSHIFT LOGGING OPERATOR

The logs generated by pipeline runs, task runs, and event listeners are stored in their respective pods. It is useful to review and analyze logs for troubleshooting and audits.

However, retaining the pods indefinitely leads to unnecessary resource consumption and cluttered namespaces.

To eliminate any dependency on the pods for viewing pipeline logs, you can use the OpenShift Elasticsearch Operator and the OpenShift Logging Operator. These Operators help you to view pipeline logs by using the Elasticsearch Kibana stack, even after you have deleted the pods that contained the logs.

2.1. PREREQUISITES

Before trying to view pipeline logs in a Kibana dashboard, ensure the following:

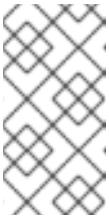
- The steps are performed by a cluster administrator.
- Logs for pipeline runs and task runs are available.
- The OpenShift Elasticsearch Operator and the OpenShift Logging Operator are installed.

2.2. VIEWING PIPELINE LOGS IN KIBANA

To view pipeline logs in the Kibana web console:

Procedure

1. Log in to OpenShift Container Platform web console as a cluster administrator.
2. In the top right of the menu bar, click the **grid** icon → **Observability** → **Logging**. The Kibana web console is displayed.
3. Create an index pattern:
    a. On the left navigation panel of the **Kibana** web console, click **Management**.
    b. Click **Create index pattern**.
    c. Under **Step 1 of 2: Define index pattern**→ **Index pattern**, enter a * pattern and click **Next Step**.
    d. Under **Step 2 of 2: Configure settings**→ **Time filter field name**, select **@timestamp** from the drop-down menu, and click **Create index pattern**.
4. Add a filter:
    a. On the left navigation panel of the **Kibana** web console, click **Discover**.
    b. Click **Add a filter +**→ **Edit Query DSL**.



NOTE

- For each of the example filters that follows, edit the query and click **Save**.
- The filters are applied one after another.

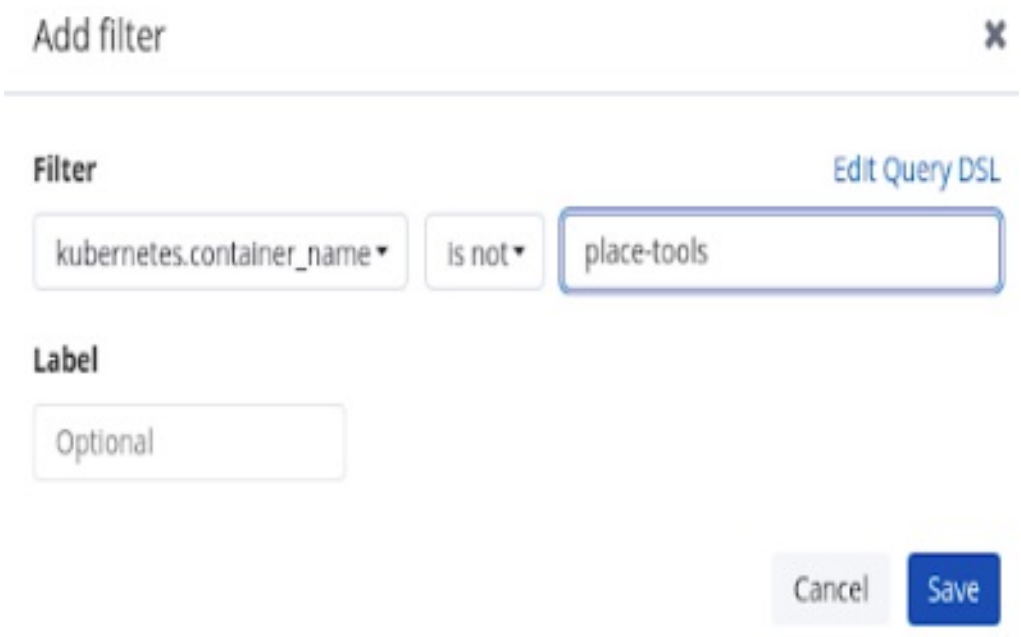
i. Filter the containers related to pipelines:

Example query to filter pipelines containers

```
{
  "query": {
 "match": {
   "kubernetes.flat_labels": {
     "query": "app_kubernetes_io/managed-by=tekton-pipelines",
     "type": "phrase"
   }
 }
  }
}
```

ii. Filter all containers that are not **place-tools** container. As an illustration of using the graphical drop-down menus instead of editing the query DSL, consider the following approach:

Figure 2.1. Example of filtering using the drop-down fields



iii. Filter **pipelinerun** in labels for highlighting:

Example query to filter pipelinerun in labels for highlighting

```
{
  "query": {
 "match": {
   "kubernetes.flat_labels": {
```

```
      "query": "tekton_dev/pipelineRun=",
      "type": "phrase"
    }
  }
   }
}
```

iv. Filter **pipeline** in labels for highlighting:

Example query to filter pipeline in labels for highlighting

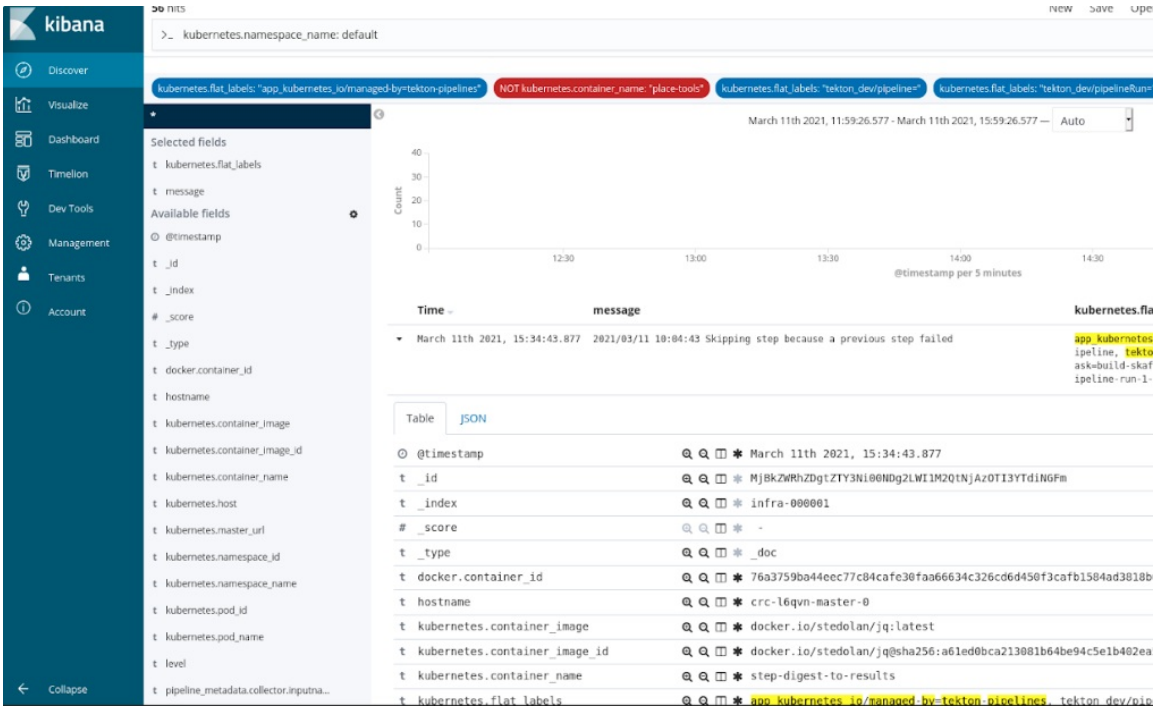
```
{
  "query": {
 "match": {
   "kubernetes.flat_labels": {
     "query": "tekton_dev/pipeline=",
     "type": "phrase"
   }
  }
   }
}
```

c. From the **Available fields** list, select the following fields:

- **kubernetes.flat_labels**
- **message**
  Ensure that the selected fields are displayed under the **Selected fields** list.

d. The logs are displayed under the **message** field.

Figure 2.2. Filtered messages



2.3. ADDITIONAL RESOURCES

- Installing OpenShift Logging
- Viewing logs for a resource
- Viewing cluster logs by using Kibana