Red Hat OpenShift Pipelines 1.14

OpenShift Pipeline の可観測性

OpenShift Pipeline の可観測性機能

Last Updated: 2024-03-27

Red Hat OpenShift Pipelines 1.14 OpenShift Pipeline の可観測性

OpenShift Pipeline の可観測性機能

法律上の通知

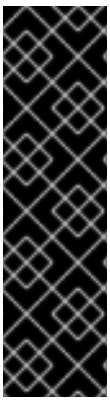
概要

このドキュメントでは、OpenShift Pipeline の可観測性機能について説明します。

目次

第1章 OPENSHIFT PIPELINES の可観測性のために TEKTON 結果を使用する

Tekton Results は、すべてのパイプライン実行とタスク実行の完全な情報をアーカイブするサービスです。必要に応じて **PipelineRun** リソースと **TaskRun** リソースをプルーニングし、Tekton Results API または **opc** コマンドラインユーティリティーを使用して、それらの YAML マニフェストとログ情報にアクセスできます。



重要

Tekton Results はテクノロジープレビューのみの機能です。テクノロジープレビュー機能は、Red Hat 製品のサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではないことがあります。Red Hat は、実稼働環境でこれらを使用することを推奨していません。テクノロジープレビュー機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

1.1. TEKTON RESULTS の概念

Tekton Results は、パイプラインの実行とタスクの実行を結果とレコードの形式でアーカイブします。

実行が完了した **PipelineRun** および **TaskRun** カスタムリソース (CR) ごとに、Tekton Results は **レコード** を作成します。

結果 には、1 つまたは複数のレコードを含めることができます。レコードは、常に 1 つの結果に含まれています。

結果はパイプライン実行に対応し、**PipelineRun** CR 自体のレコードと、パイプライン実行の一部として開始されたすべての **TaskRun** CR のレコードが含まれます。

タスク実行がパイプライン実行を使用せずに直接開始された場合、このタスク実行の結果が作成されます。この結果には、同じタスク実行のレコードが含まれます。

各結果には、**PipelineRun** または **TaskRun** CR が作成された namespace と CR の UUID が含まれる名前が付けられます。結果名の形式は **<namespace_name>/results/<parent_run_uuid>** です。この形式では、**<parent_run_uuid>** はパイプライン実行の UUUD、または直接開始されたタスク実行の UUUD です。

結果名の例

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

各レコードには、レコードを含む結果の名前と、レコードが対応する **PipelineRun** または **TaskRun** CR の UUID を含む名前が付いています。結果名の形式は **<namespace_name>/results/<parent_run_uuid>/results/<run_uuid>** です。

レコード名の例

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

レコードには、実行完了後に存在していた **TaskRun** または **PipelineRun** CR の完全な YAML マニフェストが含まれます。このマニフェストには、実行の仕様、実行に指定されたアノテーションのほか、完了時刻や実行が成功したかどうかなど、実行の結果に関する特定の情報が含まれます。

TaskRun または **PipelineRun** CR が存在している間は、次のコマンドを使用して YAML マニフェストを表示できます。

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

Tekton Results は、**TaskRun** または **PipelineRun** CR が削除された後もこのマニフェストを保存し、表示および検索できるようにします。

完了後に実行されるパイプラインの YAML マニフェストの例

```
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
```

Tekton Results は、パイプライン実行またはタスク実行の一部として実行されたすべてのツールのログ情報を含むログレコードも作成します。

すべての結果とレコードにその名前でアクセスできます。Common Expression Language (CEL) クエリーを使用して、YAML マニフェストなど、そのクエリーに含まれる情報によって結果とレコードを検索することもできます。

1.2. TEKTON RESULTS のインストールの準備

Tekton Results をインストールする前に、いくつかの準備手順を完了する必要があります。

1.2.1. SSL 証明書を使用したシークレットの準備

Tekton Results は、SSL 証明書を必要とする HTTPS プロトコルを使用した REST API を提供します。この証明書にシークレットを指定します。認証局 (CA) から提供された既存の証明書がある場合はその証明書を使用し、それ以外の場合は自己署名証明書を作成します。

前提条件

- **openssl** コマンドラインユーティリティーがインストールされている。

手順

1. CA から提供された証明書がない場合は、次のコマンドを入力して自己署名証明書を作成します。

```
$ 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"
```

tekton-results-api-service.openshift-pipelines.svc.cluster.local は、Tekton Results API に使用する予定のルートエンドポイントに置き換えます。

2. 次のコマンドを入力して、証明書からトランスポートセキュリティー層 (TLS) シークレットを作成します。

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

CA によって提供された既存の証明書を使用する場合は、**cert.pem** をこの証明書を含むファイルの名前に置き換えます。

1.2.2. データベース認証情報を使用したシークレットの準備

Tekton Results は、PostgreSQL データベースを使用してデータを保存します。Tekton Results とともに自動的にインストールされる PostgreSQL サーバー、またはデプロイメント内にすでに存在する外部 PostgreSQL サーバーのいずれかを使用するようにインストールを設定できます。どちらの場合も、データベースの認証情報にシークレットを指定します。

手順

次のいずれかの手順を実行します。

- 外部 PostgreSQL サーバーを使用する必要がない場合は、次のコマンドを入力して、**openshift-pipelines** namespace に **result** という名前のデータベースユーザーと任意のパスワードを使用してシークレットを作成します。

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

注記

このコマンドおよび後続のコマンドで、OpenShift Pipelines のカスタムターゲット namespace を設定した場合は、**openshift-pipelines** の代わりにこの namespace の名前を使用します。

- 外部 PostgreSQL データベースサーバーを使用して Tekton Results データを保存する場合は、次のコマンドを入力して、このサーバーの認証情報を使用してシークレットを作成します。

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

<user> は、Tekton Results が使用する必要がある PostgreSQL ユーザーのユーザー名に置き換えます。**<password>** は、同じアカウントのパスワードに置き換えます。

1.2.3. ログ情報用のストレージの準備

Tekton Results は、パイプラインの実行とタスクの実行に関連する情報をログに記録するために別のストレージを使用します。次のいずれかのタイプのストレージを設定できます。

- Red Hat OpenShift Pipelines クラスター上の Persistent Volume Claim (PVC)
- Google Cloud Storage
- S3 バケットストレージ

手順

以下の手順のいずれかを実行します。

- PVC を使用するには、次の手順を実行します。
  a. PVC について次の定義を含む **pvc.yaml** という名前のファイルを作成します。

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

  b. 次のコマンドを入力して定義を適用します。

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

- Google Cloud Storage を使用するには、次の手順を実行します。
  a. **gcloud** コマンドを使用して、アプリケーション認証情報ファイルを作成します。ファイルでアプリケーション認証情報を指定する手順については、Google Cloud ドキュメントの gcloud CLI を使用して提供されるユーザー認証情報 を参照してください。

  b. 次のコマンドを入力して、アプリケーション認証情報ファイルからシークレットを作成します。

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

  必要に応じて、アプリケーション認証情報ファイルのパスとファイル名を調整します。

- S3 バケットストレージを使用するには、次の手順を実行します。
  a. 次の内容を含む **s3_secret.yaml** という名前のファイルを作成します。

```
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) (1) S3 ストレージバケットの名前

(2) (2) S3 API エンドポイント URL

(3) S3 リージョン

(4) S3 アクセスキー ID

(5) S3 シークレットアクセスキー

b. 次のコマンドを入力して、ファイルからシークレットを作成します。

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

1.3. TEKTON RESULTS のインストール

Tekton Results をインストールするには、必要なリソースを提供し、**TektonResult** カスタムリソース (CR) を作成して適用する必要があります。**TektonResult** カスタムリソースを適用すると、OpenShift Pipelines Operator によって結果サービスがインストールされます。

前提条件

- Operator を使用して OpenShift Pipelines をインストールしている。
- SSL 証明書を使用してシークレットを準備している。
- ログ情報用のストレージを準備している。
- データベースの認証情報を使用してシークレットを準備している。

手順

1. 次の例に基づいて、**result.yaml** という名前のリソース定義ファイルを作成します。必要に応じて設定を調整できます。

```
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. 情報をログ記録するためのストレージの設定をこのファイルに追加します。

   - Persistent Volume Claim (PVC) を設定した場合は、次の行を追加して PVC の名前を指定します。

     ```
     logging_pvc_name: tekton-logs
     ```

   - Google Cloud Storage を設定した場合は、次の行を追加して、シークレット名、認証情報ファイル名、Google Cloud Storage バケットの名前を指定します。

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

     (1) シークレットの作成時に使用したアプリケーション認証情報ファイルの名前をパスなしで指定します。

     (2) Google Cloud Storage のバケットの名前を指定します。Tekton Chains は、このバケットを使用して、パイプライン実行とタスク実行のログ情報を保存します。

   - S3 バケットストレージを設定した場合は、次の行を追加して S3 シークレットの名前を指定します。

     ```
     secret_name: s3-credentials
     ```

3. オプション: 外部 PostgreSQL データベースサーバーを使用して Tekton Results 情報を保存する場合は、ファイルに次の行を追加します。

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

   (1) PostgreSQL サーバーのホスト名。

   (2) PostgreSQL サーバーのポート。

4. 次のコマンドを入力して、リソース定義を適用します。

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

5. 次のコマンドを入力して、Tekton Results サービス API のルートを公開します。

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

1.4. OPC コマンドラインユーティリティーを使用した TEKTON RESULTS のクエリー

opc コマンドラインユーティリティーを使用して、Tekton Results に結果とレコードを問い合わせることができます。**opc** コマンドラインユーティリティーをインストールするには、**tkn** コマンドラインユーティリティーのパッケージをインストールします。このパッケージのインストール手順については、tkn のインストール を参照してください。

レコードと結果の名前を使用して、その中に含まれるデータを取得できます。

Common Expression Language (CEL) クエリーを使用して結果とレコードを検索できます。これらの検索では、結果またはレコードの UUID が表示されます。提供された例を使用して、一般的な検索タイプのクエリーを作成できます。参照情報を使用して他のクエリーを作成することもできます。

1.4.1. Tekton Results をクエリーするための opc ユーティリティー環境の準備

Tekton Results をクエリーする前に、**opc** ユーティリティーの環境を準備する必要があります。

前提条件

- Tekton Results をインストールしている。
- **opc** ユーティリティーがインストールされている。

手順

1. 次のコマンドを入力して、**RESULTS_API** 環境変数を Tekton Results API へのルートに設定します。

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

2. 次のコマンドを入力して、Tekton Results API の認証トークンを作成します。

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

このコマンドが出力する文字列を保存します。

3. オプション: Tekton Results API による自動認証用に **~/.config/tkn/results.yaml** ファイルを作成します。ファイルには次の内容が含まれている必要があります。

```
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. Tekton Results API へのルート。**RESULTS_API** に設定したものと同じ値を使用します。
2. **oc create token** コマンドによって作成された認証トークン。このトークンを指定すると、**service_account** 設定がオーバーライドされ、**opc** はこのトークンを使用して認証します。
3. API エンドポイント用に設定した SSL 証明書を含むファイルの場所。
4. OpenShift Pipelines のカスタムターゲット namespace を設定した場合は、**openshift-pipelines** をこの namespace の名前に置き換えます。
5. 6 Tekton Results API で認証するためのサービスアカウントの名前。認証トークンを指定した場合は、**service_account** パラメーターを指定する必要はありません。

あるいは、~/**.config/tkn/results.yaml** ファイルを作成しない場合は、**--authtoken** オプションを使用して各 **opc** コマンドにトークンを渡すことができます。

1.4.2. 名前による結果とレコードのクエリー

名前を使用して、結果とレコードをリスト表示したり、クエリーを実行したりできます。

前提条件

- Tekton Results をインストールしている。
- **opc** ユーティリティーをインストールし、Tekton Results をクエリーするための環境を準備している。
- **jq** パッケージをインストールしている。

手順

1. namespace で作成されたパイプライン実行およびタスク実行に対応するすべての結果の名前をリストします。以下のコマンドを入力します。

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

コマンドの例

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

出力例

```
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. 次のコマンドを入力して、結果内のすべてのレコードの名前をリスト表示します。

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

コマンドの例

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

出力例

```
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. 次のコマンドを入力して、レコードからパイプライン実行またはタスク実行の YAML マニフェストを取得します。

```
$ 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))'
```

コマンドの例

```
$ 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. オプション: ログレコード名を使用して、レコードから実行されたタスクのログ情報を取得します。ログレコード名を取得するには、レコード名の **records** は、**logs** に置き換えます。以下のコマンドを入力します。

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

コマンドの例

```
$ 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. 結果の検索

Common Expression Language (CEL) クエリーを使用して結果を検索できます。たとえば、成功しなかったパイプライン実行の結果を見つけることができます。ただし、関連情報のほとんどは結果オブジェクトには含まれていません。名前、完了時間、その他のデータで検索するには、レコードを検索します。

前提条件

- Tekton Results をインストールしている。
- **opc** ユーティリティーをインストールし、Tekton Results をクエリーするための環境を準備している。

手順

- 次のコマンドを入力して、CEL クエリーを使用して結果を検索します。

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

<namespace_name> は、パイプラインの実行またはタスクの実行が作成された namespace に置き換えます。

表1.1 結果の CEL クエリーの例

目的	CEL クエリー
失敗したすべての実行の結果	**!(summary.status == SUCCESS)**
アノテーション **ann1** および **ann2** を含むすべてのパイプライン実行の結果	**summary.annotations.contains('ann1') && summary.annotations.contains('ann2') && summary.type=='PIPELINE_RUN'**

1.4.4. レコードの検索

Common Expression Language (CEL) クエリーを使用してレコードを検索できます。各レコードにはパイプライン実行またはタスク実行に関する完全な YAML 情報が含まれるため、さまざまな基準でレコードを検索できます。

前提条件

- Tekton Results をインストールしている。
- **opc** ユーティリティーをインストールし、Tekton Results をクエリーするための環境を準備している。

手順

- 次のコマンドを入力して、CEL クエリーを使用してレコードを検索します。

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

<namespace_name> は、パイプラインの実行またはタスクの実行が作成された namespace に置き換えます。または、次のコマンドを入力して、単一の結果内のレコードを検索します。

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

<result_name> は、結果の完全名に置き換えます。

表1.2 レコードの CEL クエリーの例

目的	CEL クエリー
失敗したすべてのタスク実行またはパイプライン実行のレコード	**!(data.status.conditions[0].status == 'True')**
TaskRun または **PipelineRun** カスタムリソース (CR) の名前が **run1** であったレコード	**data.metadata.name == 'run1'**
run1 という名前の **PipelineRun** CR で開始されたタスク実行すべてのレコード	**data_type == 'TASK_RUN' && data.metadata.labels['tekton.dev/pipelineRun'] == 'run1'**
Pipeline1 という名前の **パイプライン** CR から作成されたすべてのパイプライン実行とタスク実行のレコード	**data.metadata.labels['tekton.dev/pipeline'] == 'pipeline1'**
Pipeline1 という名前の **パイプライン** CR から作成されたすべてのパイプライン実行のレコード	**data.metadata.labels['tekton.dev/pipeline'] == 'pipeline1' && data_type == 'PIPELINE_RUN'**
TaskRun CR 名が **hello** で始まるすべてのタスク実行のレコード	**data.metadata.name.startsWith('hello') && data_type=='TASK_RUN'**
完了までに 5 分以上かかったすべてのパイプライン実行のレコード	**data.status.completionTime - data.status.startTime > duration('5m') && data_type == 'PIPELINE_RUN'**

目的	CEL クエリー
2023 年 10 月 7 日に完了したすべてのパイプライン実行とタスク実行のレコード	**data.status.completionTime.getDate() == 7 && data.status.completionTime.getMonth() == 10 && data.status.completionTime.getFullYear() == 2023**
3 つ以上のタスクを含むすべてのパイプライン実行のレコード	**size(data.status.pipelineSpec.tasks) >= 3 && data_type == 'PIPELINE_RUN'**
ann1 を含むアノテーションが付いたすべてのパイプライン実行のレコード	**data.metadata.annotations.contains('ann1') && data_type == 'PIPELINE_RUN'**
ann1 を含むアノテーションと **hello** で始まる **PipelineRun** CR の名前が付いたすべてのパイプライン実行のレコード	**data.metadata.annotations.contains('ann1') && data.metadata.name.startsWith('hello') && data_type == 'PIPELINE_RUN'**

1.4.5. 検索結果の参考情報

結果の共通表現言語 (CEL) クエリーでは次のフィールドを使用できます。

表1.3 結果の CEL クエリーで使用できるフィールド

CEL フィールド	Description
parent	**PipelineRun** または **TaskRun** カスタムリソース (CR) が作成された namespace。
uid	結果の一意識別子。
annotations	**PipelineRun** または **TaskRun** CR に追加されたアノテーション。
summary	結果の概要。
create_time	結果の作成時間。
update_time	結果の最終更新時刻。

summary.status フィールドを使用して、パイプラインの実行が成功したかどうかを判断できます。このフィールドには次の値を指定できます。

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

- **CANCELLED**

注記

このフィールドの値を指定する場合は、" や ' などの引用符を使用しないでください。

1.4.6. レコード検索の参考情報

レコードの Common Expression Language (CEL) クエリーでは次のフィールドを使用できます。

表1.4 レコードの CEL クエリーで使用できるフィールド

CEL フィールド	Description	値
name	レコード名	
data_type	レコードタイプ識別子	**tekton.dev/v1.TaskRun** または **TASK_RUNtekton.dev/v1.PipelineRun** または **PIPELINE_RUNresults.tekton.dev/v1alpha2.Log**
data	タスク実行またはパイプライン実行の YAML データ。ログレコードでは、このフィールドにはログ出力が含まれます。	

data フィールドにはタスク実行またはパイプライン実行の YAML データ全体が含まれるため、このデータのすべての要素を CEL クエリーで使用できます。たとえば、**data.status.completionTime** には、タスク実行またはパイプライン実行の完了時間が含まれます。

1.5. 関連情報

- Common Expression Language specification

第2章 OPENSHIFT LOGGING OPERATOR を使用したパイプラインログの表示

パイプライン実行、タスク実行、およびイベントリスナーによって生成されるログは、それぞれの Pod に保存されます。トラブルシューティングおよび監査に関するログの確認や分析は有用です。

ただし、Pod を無期限に保持すると、リソースを無駄に消費したり、namespace が不必要に分散されたりする可能性があります。

Pod の依存関係を削除して、パイプラインログを表示するには、OpenShift Elasticsearch Operator および OpenShift Logging Operator を使用できます。これらの Operator を使用すると、ログを含む Pod を削除した場合でも、Elasticsearch Kibana スタックを使用してパイプラインログを表示できます。

2.1. 前提条件

Kibana ダッシュボードでパイプラインログを表示しようとする前に、以下を確認してください。

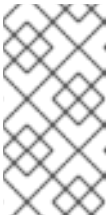
- クラスター管理者がこの手順を実行する。
- パイプライン実行およびタスク実行のログが利用可能である。
- OpenShift Elasticsearch Operator および OpenShift Logging Operator がインストールされている。

2.2. KIBANA でのパイプラインログの表示

Kibana Web コンソールでパイプラインログを表示するには、以下を実行します。

手順

1. クラスター管理者として OpenShift Container Platform Web コンソールにログインします。
2. メニューバーの右上にある **グリッド** アイコン → **Observability** → **Logging** をクリックします。Kibana Web コンソールが表示されます。
3. インデックスパターンを作成します。
    a. **Kibana** Web コンソールの左側のナビゲーションパネルで **Management** をクリックします。
    b. **Create index pattern** をクリックします。
    c. **ステップ 1/2: Define index pattern**→ **Index pattern** で、* のパターンを入力して **Next Step** をクリックします。
    d. **ステップ 2/2: Configure settings** → **Time filter field name** で、ドロップダウンメニューから **@timestamp** を選択し、**Create index pattern** をクリックします。
4. フィルターを追加します。
    a. **Kibana** Web コンソールの左側のナビゲーションパネルで **Discover** をクリックします。
    b. **Add a filter +**→ **Edit Query DSL** をクリックします。



注記

- 以下のフィルター例の例ごとに、クエリーを編集して **Save** をクリックします。
- フィルターは順次、適用されます。

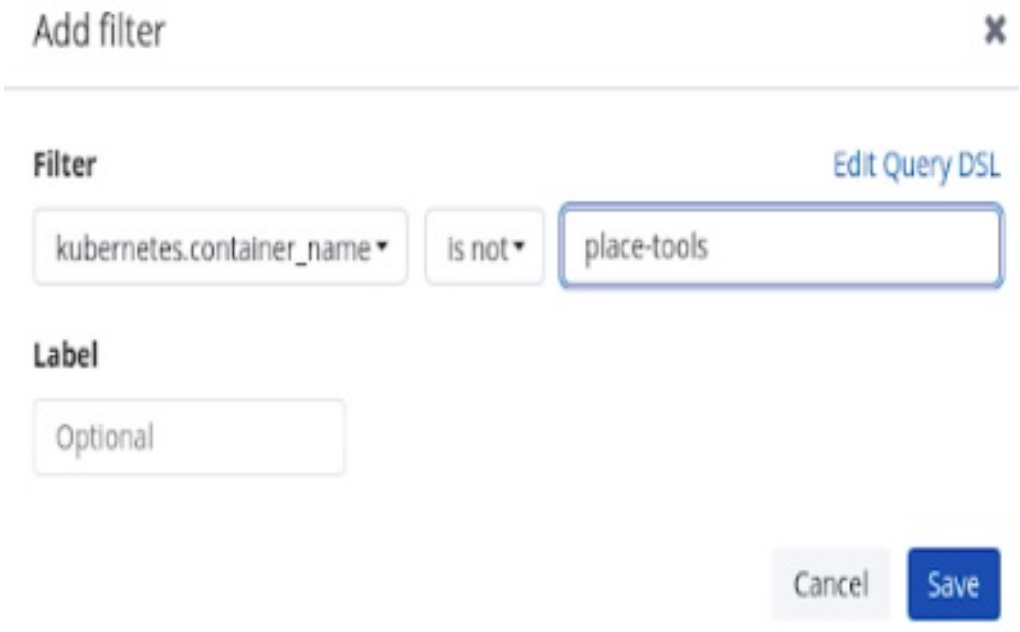
i. パイプラインに関連するコンテナーをフィルタリングします。

パイプラインコンテナーをフィルタリングするクエリーの例

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

ii. **place-tools** コンテナーではないすべてのコンテナーをフィルタリングします。クエリー DSL を編集する代わりに、グラフィカルドロップダウンメニューを使用する例として、以下の方法を考慮してください。

図2.1 ドロップダウンフィールドを使用したフィルタリングの例



iii. 強調表示できるように **pipelinerun** をラベルでフィルタリングします。

強調表示できるように pipelinerun をラベルでフィルタリングするクエリーの例

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

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

iv. 強調表示できるように **pipeline** をラベルでフィルタリングします。

強調表示できるように pipeline をラベルでフィルタリングするクエリーの例

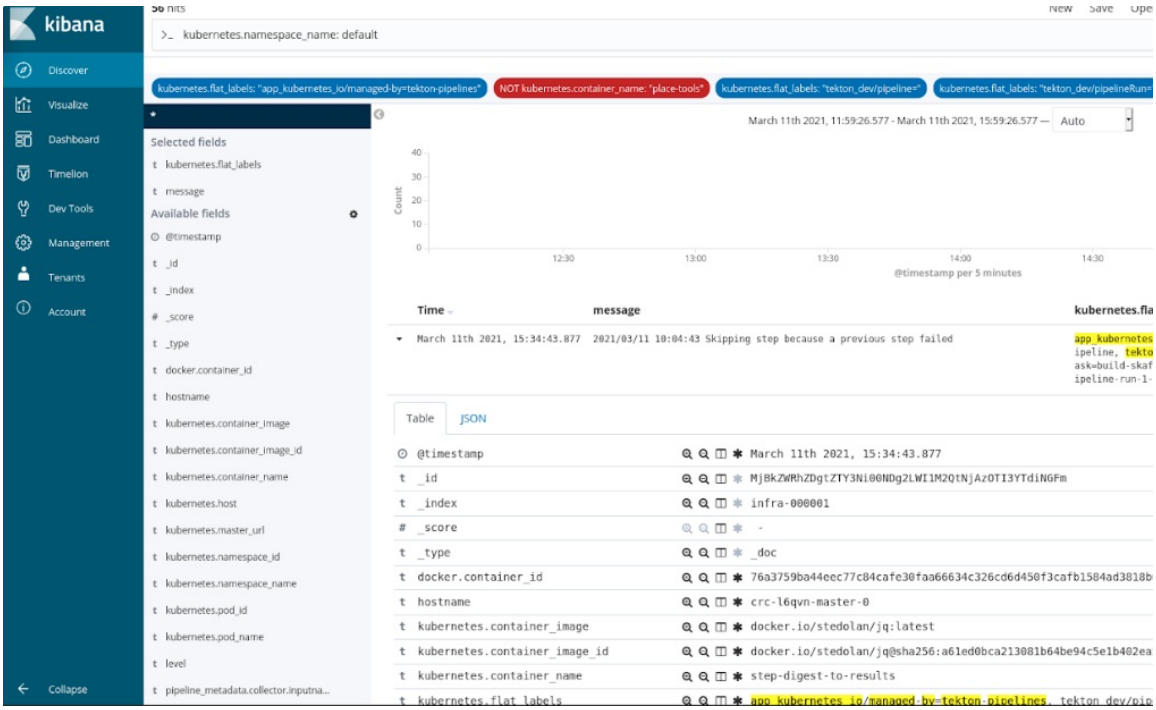
```
{
  "query": {
 "match": {
   "kubernetes.flat_labels": {
     "query": "tekton_dev/pipeline=",
     "type": "phrase"
   }
  }
   }
}
```

c. **Available fields** リストから以下のフィールドを選択します。

- **kubernetes.flat_labels**
- **message**
  選択したフィールドが **Selected fields** リストに表示されていることを確認します。

d. ログは **message** フィールドの下に表示されます。

図2.2 フィルタリングされたメッセージ



2.3. 関連情報

- OpenShift Logging のインストール
- リソースのログの表示
- Kibana を使用したクラスターログの表示