



# Red Hat OpenShift

## Creating an API server source

- [Creating an API server source by using the web console](#)
- [Creating an API server source by using the Knative CLI](#)
  - [Knative CLI sink flag](#)
- [Creating an API server source by using YAML files](#)

The API server source is an event source that can be used to connect an event sink, such as a Knative service, to the Kubernetes API server. The API server source watches for Kubernetes events and forwards them to the Knative Eventing broker.

## Creating an API server source by using the web console

After Knative Eventing is installed on your cluster, you can create an API server source by using the web console. Using the OpenShift Container Platform web console provides a streamlined and intuitive user interface to create an event source.

### Prerequisites

- You have logged in to the OpenShift Container Platform web console.
- The OpenShift Serverless Operator and Knative Eventing are installed on the cluster.
- You have created a project or have access to a project with the appropriate roles and permissions to create applications and other workloads in OpenShift Container Platform.
- You have installed the OpenShift CLI ( `oc` ).



### *Procedure*

If you want to re-use an existing service account, you can modify your existing `ServiceAccount` resource to include the required permissions instead of creating a new resource.

- Create a service account, role, and role binding for the event source as a YAML file:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: events-sa
  namespace: default (1)

---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: event-watcher
  namespace: default (1)
rules:
  - apiGroups:
      - ""
    resources:
      - events
    verbs:
      - get
      - list
      - watch

---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: k8s-ra-event-watcher
  namespace: default (1)
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: event-watcher
subjects:
  - kind: ServiceAccount
    name: events-sa
    namespace: default (1)
```

**1** Change this namespace to the namespace that you have selected for installing the event source.

- Apply the YAML file:

```
$ oc apply -f <filename>
```

- In the **Developer** perspective, navigate to **+Add → Event Source**. The **Event Sources** page is displayed.
- Optional: If you have multiple providers for your event sources, select the required provider from the **Providers** list to filter the available event sources from the provider.
- Select **ApiServerSource** and then click **Create Event Source**. The **Create Event Source** page is displayed.
- Configure the **ApiServerSource** settings by using the **Form view** or **YAML view**:



You can switch between the **Form view** and **YAML view**. The data is persisted when switching between the views.

- Enter **v1** as the **APIVERSION** and **Event** as the **KIND**.
- Select the **Service Account Name** for the service account that you created.
- In the **Target** section, select your event sink. This can be either a **Resource** or a **URI**:
  - Select **Resource** to use a channel, broker, or service as an event sink for the event source.
  - Select **URI** to specify a Uniform Resource Identifier (URI) where the events are routed to.
- Click **Create**.

## Verification

- After you have created the API server source, check that it is connected to the event sink by viewing it in the **Topology** view.

The screenshot shows the OpenShift Serverless console interface. On the left, there's a 'Display Options' dropdown. The main area displays a topology diagram with a 'REV' event source (labeled 'event-...-vn85s') connected to a 'KSV' sink (labeled 'event-...-ay-api'), which is then connected to the 'testevents' API server source (labeled 'AS testevents'). The right-hand panel shows details for 'testevents', including its Knative Service URI and a list of pods.



If a URI sink is used, you can modify the URI by right-clicking on **URI sink** → **Edit URI**.

## Deleting the API server source

- Navigate to the **Topology** view.
- Right-click the API server source and select **Delete ApiServerSource**.

The screenshot shows the OpenShift Serverless console interface. The left sidebar has a 'Developer' dropdown and a list of navigation options: '+Add', 'Topology', 'Monitoring', 'Search', 'Builds', 'Helm', 'Project', 'Config Maps', and 'Secrets'. The main area displays a topology diagram with a 'REV' event source (labeled 'hellow...-wcrsq') connected to a 'KSV' sink (labeled 'helloworld-go'), which is then connected to the 'api-server' API server source (labeled 'AS api-server'). A right-click context menu is open over the 'api-server' source, with the option 'Delete ApiServerSource' highlighted.

## Creating an API server source by using the Knative CLI

You can use the `kn source apiserver create` command to create an API server source by using the `kn` CLI. Using the `kn` CLI to create an API server source provides a more streamlined and intuitive user interface than modifying YAML files directly.

### Prerequisites

- The OpenShift Serverless Operator and Knative Eventing are installed on the cluster.
- You have created a project or have access to a project with the appropriate roles and permissions to create applications and other workloads in OpenShift Container Platform.
- You have installed the OpenShift CLI ( `oc` ).
- You have installed the Knative ( `kn` ) CLI.

#### *Procedure*



If you want to re-use an existing service account, you can modify your existing `ServiceAccount` resource to include the required permissions instead of creating a new resource.

- Create a service account, role, and role binding for the event source as a YAML file:

```

apiVersion: v1
kind: ServiceAccount
metadata:
  name: events-sa
  namespace: default (1)

---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: event-watcher
  namespace: default (1)
rules:
  - apiGroups:
      - ""
    resources:
      - events
    verbs:
      - get
      - list
      - watch

---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: k8s-ra-event-watcher
  namespace: default (1)
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: event-watcher
subjects:
  - kind: ServiceAccount
    name: events-sa
    namespace: default (1)

```

- 1 Change this namespace to the namespace that you have selected for installing the event source.

- Apply the YAML file:

```
$ oc apply -f <filename>
```

- Create an API server source that has an event sink. In the following example, the sink is a broker:

```
$ kn source apiserver create <event_source_name> --sink broker:  
<broker_name> --resource "event:v1" --service-account  
<service_account_name> --mode Resource
```

- To check that the API server source is set up correctly, create a Knative service that dumps incoming messages to its log:

```
$ kn service create event-display --image quay.io/openshift-  
knative/showcase
```

- If you used a broker as an event sink, create a trigger to filter events from the default broker to the service:

```
$ kn trigger create <trigger_name> --sink ksvc:event-display
```

- Create events by launching a pod in the default namespace:

```
$ oc create deployment event-origin --image quay.io/openshift-  
knative/showcase
```

- Check that the controller is mapped correctly by inspecting the output generated by the following command:

```
$ kn source apiserver describe <source_name>
```

## Example output



```

Name:          mysource
Namespace:     default
Annotations:   sources.knative.dev/creator=developer,
sources.knative.dev/lastModifier=developer
Age:          3m
ServiceAccountName: events-sa
Mode:         Resource
Sink:
  Name:        default
  Namespace:   default
  Kind:        Broker (eventing.knative.dev/v1)
Resources:
  Kind:        event (v1)
  Controller:  false
Conditions:
  OK TYPE                AGE REASON
  ++ Ready              3m
  ++ Deployed           3m
  ++ SinkProvided       3m
  ++ SufficientPermissions 3m
  ++ EventTypesProvided 3m

```

## Verification

To verify that the Kubernetes events were sent to Knative, look at the event-display logs or use web browser to see the events.

- To view the events in a web browser, open the link returned by the following command:

```
$ kn service describe event-display -o url
```



# Welcome to Serverless, Cloud-Native world!

## What can I do from here?

Invoke a hello endpoint: [/hello](#).

It will send CloudEvent to `K_SINK = http://localhost:31111`

## Collected CloudEvents (1)

id	source	application/json
Jiechu5w	Kubernetes	<pre>{   "apiVersion": "v1",   "involvedObject": {     "apiVersion": "v1",     "fieldPath": "spec.containers(hello-node)",     "kind": "Pod",     "name": "hello-node",     "namespace": "default"   },   "kind": "Event",   "message": "Started container",   "metadata": {     "name": "hello-node.159d7608e3a35572c",     "namespace": "default"   },   "reason": "Started" }</pre>
type	time	
dev.knative.apiserver.resource.update	less than a minute	

This app captures CloudEvents on `POST /events` endpoint. Newer are listed first.

## Application

Group: `com.redhat.openshift`

Artifact: `knative-showcase`

Version: `v0.7.0-4-g23d460f`

Platform: `Quarkus/2.13.7.Final-redhat-00003 Java/17.0.7`

Powered by:



QUARKUS



This application has been written with React & Quarkus to showcase Knative.

## Figure 1. Example browser page

- Alternatively, to see the logs in the terminal, view the event-display logs for the pods by entering the following command:

```
$ oc logs $(oc get pod -o name | grep event-display) -c user-container
```

## Example output

```

cloudevents.Event
Validation: valid
Context Attributes,
  specversion: 1.0
  type: dev.knative.apiserver.resource.update
  datacontenttype: application/json
  ...
Data,
  {
    "apiVersion": "v1",
    "involvedObject": {
      "apiVersion": "v1",
      "fieldPath": "spec.containers{event-origin}",
      "kind": "Pod",
      "name": "event-origin",
      "namespace": "default",
      .....
    },
    "kind": "Event",
    "message": "Started container",
    "metadata": {
      "name": "event-origin.159d7608e3a3572c",
      "namespace": "default",
      ....
    },
    "reason": "Started",
    ...
  }

```

## Deleting the API server source

- Delete the trigger:

```
$ kn trigger delete <trigger_name>
```

- Delete the event source:

```
$ kn source apiserver delete <source_name>
```

- Delete the service account, cluster role, and cluster binding:

```
$ oc delete -f authentication.yaml
```

## Knative CLI sink flag

When you create an event source by using the Knative ( `kn` ) CLI, you can specify a sink where events are sent to from that resource by using the `--sink` flag. The sink can be any addressable or callable resource that can receive incoming events from other resources.

The following example creates a sink binding that uses a service, `http://event-display.svc.cluster.local`, as the sink:

### Example command using the sink flag

```
$ kn source binding create bind-heartbeat \
  --namespace sinkbinding-example \
  --subject "Job:batch/v1:app=heartbeat-cron" \
  --sink http://event-display.svc.cluster.local \ (1)
  --ce-override "sink=bound"
```

- 1 `svc` in `http://event-display.svc.cluster.local` determines that the sink is a Knative service. Other default sink prefixes include `channel`, and `broker`.

## Creating an API server source by using YAML files

Creating Knative resources by using YAML files uses a declarative API, which enables you to describe event sources declaratively and in a reproducible manner. To create an API server source by using YAML, you must create a YAML file that defines an `ApiServerSource` object, then apply it by using the `oc apply` command.

### Prerequisites

- The OpenShift Serverless Operator and Knative Eventing are installed on the cluster.
- You have created a project or have access to a project with the appropriate roles and permissions to create applications and other workloads in OpenShift Container Platform.
- You have created the `default` broker in the same namespace as the one defined in the API server source YAML file.
- Install the OpenShift CLI ( `oc` ).



### *Procedure*

If you want to re-use an existing service account, you can modify your existing `ServiceAccount` resource to include the required permissions instead of creating a new resource.

- Create a service account, role, and role binding for the event source as a YAML file:

```

apiVersion: v1
kind: ServiceAccount
metadata:
  name: events-sa
  namespace: default (1)

---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: event-watcher
  namespace: default (1)
rules:
  - apiGroups:
      - ""
    resources:
      - events
    verbs:
      - get
      - list
      - watch

---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: k8s-ra-event-watcher
  namespace: default (1)
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: event-watcher
subjects:
  - kind: ServiceAccount
    name: events-sa
    namespace: default (1)

```

**1** Change this namespace to the namespace that you have selected for installing the event source.

- Apply the YAML file:

```
$ oc apply -f <filename>
```

- Create an API server source as a YAML file:

```
apiVersion: sources.knative.dev/v1alpha1
kind: ApiServerSource
metadata:
  name: testevents
spec:
  serviceAccountName: events-sa
  mode: Resource
  resources:
    - apiVersion: v1
      kind: Event
  sink:
    ref:
      apiVersion: eventing.knative.dev/v1
      kind: Broker
      name: default
```

- Apply the `ApiServerSource` YAML file:

```
$ oc apply -f <filename>
```

- To check that the API server source is set up correctly, create a Knative service as a YAML file that dumps incoming messages to its log:

```
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: event-display
  namespace: default
spec:
  template:
    spec:
      containers:
        - image: quay.io/openshift-knative/showcase
```

- Apply the `Service` YAML file:

```
$ oc apply -f <filename>
```

- Create a `Trigger` object as a YAML file that filters events from the default broker to the service created in the previous step:

```
apiVersion: eventing.knative.dev/v1
kind: Trigger
metadata:
  name: event-display-trigger
  namespace: default
spec:
  broker: default
  subscriber:
    ref:
      apiVersion: serving.knative.dev/v1
      kind: Service
      name: event-display
```

- Apply the Trigger YAML file:

```
$ oc apply -f <filename>
```

- Create events by launching a pod in the default namespace:

```
$ oc create deployment event-origin --image=quay.io/openshift-knative/showcase
```

- Check that the controller is mapped correctly, by entering the following command and inspecting the output:

```
$ oc get apiserversource.sources.knative.dev testevents -o yaml
```

## Example output




```
apiVersion: sources.knative.dev/v1alpha1
kind: ApiServerSource
metadata:
  annotations:
  creationTimestamp: "2020-04-07T17:24:54Z"
  generation: 1
  name: testevents
  namespace: default
  resourceVersion: "62868"
  selfLink:
/apis/sources.knative.dev/v1alpha1/namespaces/default/apiserver
sources/testevents2
  uid: 1603d863-bb06-4d1c-b371-f580b4db99fa
spec:
  mode: Resource
  resources:
  - apiVersion: v1
    controller: false
    controllerSelector:
      apiVersion: ""
      kind: ""
      name: ""
      uid: ""
    kind: Event
    labelSelector: {}
  serviceAccountName: events-sa
  sink:
    ref:
      apiVersion: eventing.knative.dev/v1
      kind: Broker
      name: default
```

## Verification

To verify that the Kubernetes events were sent to Knative, you can look at the event-display logs or use web browser to see the events.

- To view the events in a web browser, open the link returned by the following command:

```
$ oc get ksvc event-display -o jsonpath='{.status.url}'
```


Welcome to Serverless, Cloud-Native world!

### What can I do from here?

Invoke a hello endpoint: </hello>.

💡 It will send CloudEvent to `K_SINK = http://localhost:31111`

### Collected CloudEvents (1)



id	source	application/json
<span style="color: red;">Jiechu5w</span>	Kubernetes	<pre>{   "apiVersion": "v1",   "involvedObject": {     "apiVersion": "v1",     "fieldPath": "spec.containers(hello-node)",     "kind": "Pod",     "name": "hello-node",     "namespace": "default"   },   "kind": "Event",   "message": "Started container",   "metadata": {     "name": "hello-node.159d7608e3a35572c",     "namespace": "default"   },   "reason": "Started" }</pre>
type	time	
dev.knative.apiserver.resource.update	less than a minute	

💡 This app captures CloudEvents on `POST /events` endpoint. Newer are listed first.

### Application

Group: com.redhat.openshift  
 Artifact: knative-showcase  
 Version: v0.7.0-4-g23d460f  
 Platform: Quarkus/2.13.7.Final-redhat-00003 Java/17.0.7

### Powered by:

This application has been written with React & Quarkus to showcase Knative.

**Figure 2. Example browser page**

- To see the logs in the terminal, view the event-display logs for the pods by entering the following command:

```
$ oc logs $(oc get pod -o name | grep event-display) -c user-container
```

### Example output

```

cloudevents.Event
Validation: valid
Context Attributes,
  specversion: 1.0
  type: dev.knative.apiserver.resource.update
  datacontenttype: application/json
  ...
Data,
  {
    "apiVersion": "v1",
    "involvedObject": {
      "apiVersion": "v1",
      "fieldPath": "spec.containers{event-origin}",
      "kind": "Pod",
      "name": "event-origin",
      "namespace": "default",
      .....
    },
    "kind": "Event",
    "message": "Started container",
    "metadata": {
      "name": "event-origin.159d7608e3a3572c",
      "namespace": "default",
      ....
    },
    "reason": "Started",
    ...
  }

```

## Deleting the API server source

- Delete the trigger:

```
$ oc delete -f trigger.yaml
```

- Delete the event source:

```
$ oc delete -f k8s-events.yaml
```

- Delete the service account, cluster role, and cluster binding:

```
$ oc delete -f authentication.yaml
```



Copyright © 2024 Red Hat, Inc.