Content

Installing Knative Serving using YAML files

Install the Knative Serving component

1. Install the required custom resources

   kubectl apply -f https://github.com/knative/serving/releases/download/knative-v1.20.0/serving-crds.yaml

```
PS D:\apps-data\docker> kubectl apply -f https://github.com/knative/serving/releas
es/download/knative-v1.20.1/serving-crds.yaml
Warning: unrecognized format "int64"
customresourcedefinition.apiextensions.k8s.io/certificates.networking.internal.kna
tive.dev created
Warning: unrecognized format "int32"
customresourcedefinition.apiextensions.k8s.io/configurations.serving.knative.dev c
reated
customresourcedefinition.apiextensions.k8s.io/clusterdomainclaims.networking.inter
nal.knative.dev created
customresourcedefinition.apiextensions.k8s.io/domainmappings.serving.knative.dev c
reated
customresourcedefinition.apiextensions.k8s.io/ingresses.networking.internal.knativ
e.dev created
customresourcedefinition.apiextensions.k8s.io/metrics.autoscaling.internal.knative
.dev created
customresourcedefinition.apiextensions.k8s.io/podautoscalers.autoscaling.internal.
knative.dev created
customresourcedefinition.apiextensions.k8s.io/revisions.serving.knative.dev create
d
customresourcedefinition.apiextensions.k8s.io/routes.serving.knative.dev created
customresourcedefinition.apiextensions.k8s.io/serverlessservices.networking.intern
al.knative.dev created
customresourcedefinition.apiextensions.k8s.io/services.serving.knative.dev created
customresourcedefinition.apiextensions.k8s.io/images.caching.internal.knative.dev
created
```

2. Install the core components of Knative Serving

   kubectl apply -f https://github.com/knative/serving/releases/download/knative-v1.20.0/serving-core.yaml

```
PS D:\istio-1.28.1> kubectl apply -f https://github.com/knative/serving/releases/down
load/knative-v1.20.0/serving-core.yaml
namespace/knative-serving created
role.rbac.authorization.k8s.io/knative-serving-activator created
clusterrole.rbac.authorization.k8s.io/knative-serving-activator-cluster created
clusterrole.rbac.authorization.k8s.io/knative-serving-aggregated-addressable-resolver
 created
clusterrole.rbac.authorization.k8s.io/knative-serving-addressable-resolver created
clusterrole.rbac.authorization.k8s.io/knative-serving-namespaced-admin created
clusterrole.rbac.authorization.k8s.io/knative-serving-namespaced-edit created
clusterrole.rbac.authorization.k8s.io/knative-serving-namespaced-view created
clusterrole.rbac.authorization.k8s.io/knative-serving-core created
clusterrole.rbac.authorization.k8s.io/knative-serving-podspecable-binding created
serviceaccount/controller created
clusterrole.rbac.authorization.k8s.io/knative-serving-admin created
clusterrolebinding.rbac.authorization.k8s.io/knative-serving-controller-admin created
clusterrolebinding.rbac.authorization.k8s.io/knative-serving-controller-addressable-r
esolver created
serviceaccount/activator created
rolebinding.rbac.authorization.k8s.io/knative-serving-activator created
clusterrolebinding.rbac.authorization.k8s.io/knative-serving-activator-cluster create
d
customresourcedefinition.apiextensions.k8s.io/images.caching.internal.knative.dev unc
hanged
certificate.networking.internal.knative.dev/routing-serving-certs created
customresourcedefinition.apiextensions.k8s.io/certificates.networking.internal.knativ
```

Install a networking layer

1. Install the Knative Kourier controller
   kubectl apply -f https://github.com/knative-extensions/net-kourier/releases/download/knative-v1.20.0/kourier.yaml

Problems that I met:

```
code-samples/serving/hello-world/helloworld-go/service.yaml
Unable to connect to the server: dial tcp 20.205.243.166:443: connectex: A connect
ion attempt failed because the connected party did not properly respond after a pe
riod of time, or established connection failed because connected host has failed t
o respond.
```

Solutions: Download the file locally https://github.com/knative-extensions/net-kourier/releases/download/knative-v1.20.0/kourier.yaml.

Then run the command *kubectl apply -f kourier.yaml*

```
PS D:\apps-data\docker> kubectl apply -f kourier.yaml
namespace/kourier-system created
configmap/kourier-bootstrap created
configmap/config-kourier created
serviceaccount/net-kourier created
clusterrole.rbac.authorization.k8s.io/net-kourier created
clusterrolebinding.rbac.authorization.k8s.io/net-kourier created
deployment.apps/net-kourier-controller created
service/net-kourier-controller created
deployment.apps/3scale-kourier-gateway created
service/kourier created
service/kourier-internal created
horizontalpodautoscaler.autoscaling/3scale-kourier-gateway created
poddisruptionbudget.policy/3scale-kourier-gateway-pdb created
```

2. Configure Knative Serving to use Kourier by default

```
@"
apiVersion: v1
kind: ConfigMap
metadata:
  name: config-network
  namespace: knative-serving
data:
  ingress-class: kourier.ingress.networking.knative.dev
"@ | kubectl apply -f –
```

```
PS D:\apps-data\docker> @"
>> apiVersion: v1
>> kind: ConfigMap
>> metadata:
>>   name: config-network
>>   namespace: knative-serving
>> data:
>>   ingress-class: kourier.ingress.networking.knative.dev
>> "@ | kubectl apply -f -
configmap/config-network_configured
```

3. Fetch the External IP address or CNAME

```
kubectl --namespace kourier-system get service kourier
```

```
PS D:\apps-data\docker> kubectl --namespace kourier-system get service kourier
NAME      TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)                      AGE
kourier   LoadBalancer   10.101.112.243   <pending>     80:31276/TCP,443:31537/TCP   10m
```

Verify the installation

```
kubectl get pods -n knative-serving
```

```
PS D:\apps-data\docker> kubectl get pods -n knative-serving
NAME                                    READY   STATUS    RESTARTS   AGE
activator-f56b94b44-f9flv               1/1     Running   0          37m
autoscaler-74d66ffcd-r446p              1/1     Running   0          37m
controller-5d68d6d797-6dplv             1/1     Running   0          37m
net-kourier-controller-7b7dd6479-l75z4  1/1     Running   0          11m
webhook-c47fc76d8-f5pj6                 1/1     Running   0          37m
```

Configure DNS

No DNS

1. Knative provides a Kubernetes Job called default-domain that configures Knative Serving to use sslip.io as the default DNS suffix

   kubectl apply -f https://github.com/knative/serving/releases/download/knative-v1.20.0/serving-default-domain.yaml

```
PS D:\apps-data\docker> kubectl apply -f https://github.com/knative/serving/releas
es/download/knative-v1.20.0/serving-default-domain.yaml
job.batch/default-domain created
service/default-domain-service created
```

2. Configure Knative to use a domain reachable from outside the cluster

```
@"
apiVersion: v1
kind: ConfigMap
metadata:
  name: config-domain
  namespace: knative-serving
data:
  example.com: ""
"@ | kubectl apply -f -
```

```
PS D:\apps-data\docker> @"
>> apiVersion: v1
>> kind: ConfigMap
>> metadata:
>>   name: config-domain
>>   namespace: knative-serving
>> data:
>>   example.com: ""
>> "@ | kubectl apply -f -
configmap/config-domain configured
```

```
@"
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: helloworld-go
  namespace: default
spec:
  template:
    spec:
      containers:
      - image: gcr.io/knative-samples/helloworld-go
        env:
        - name: TARGET
          value: "Go Sample v1"
"@ | kubectl apply -f -
```

```
PS D:\apps-data\docker> @"
>> apiVersion: serving.knative.dev/v1
>> kind: Service
>> metadata:
>>   name: helloworld-go
>>   namespace: default
>> spec:
>>   template:
>>     spec:
>>       containers:
>>       - image: gcr.io/knative-samples/helloworld-go
>>         env:
>>         - name: TARGET
>>           value: "Go Sample v1"
>> "@ | kubectl apply -f -
Warning: Kubernetes default value is insecure, Knative may default this to secure
in a future release: spec.template.spec.containers[0].securityContext.allowPrivile
geEscalation, spec.template.spec.containers[0].securityContext.capabilities, spec.
template.spec.containers[0].securityContext.runAsNonRoot, spec.template.spec.conta
iners[0].securityContext.seccompProfile
service.serving.knative.dev/helloworld-go created
```

3. Get the URL of the application

   kubectl get ksvc

```
PS D:\apps-data\docker> kubectl get ksvc
NAME            URL                                        LATESTCREATED         L
ATESTREADY           READY   REASON
helloworld-go   http://helloworld-go.default.example.com   helloworld-go-00001   h
elloworld-go-00001   True
```

4. Instruct curl to connect to the External IP or CNAME defined by the networking layer

```
PS D:\apps-data\docker> kubectl get svc -n kourier-system
NAME               TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)
            AGE
kourier            LoadBalancer   10.101.112.243   <pending>     80:31276/TCP,443:
31537/TCP   89m
kourier-internal   ClusterIP      10.109.126.209   <none>        80/TCP,443/TCP
            89m
```

```
PS D:\apps-data\docker> minikube ip
192.168.49.2
```

According to the above two figures, it can get http://192.168.49.2:31276

```
PS D:\apps-data\docker> curl.exe -H "Host: helloworld-go.default.example.com" http
://192.168.49.2:31276
curl: (28) Failed to connect to 192.168.49.2 port 31276 after 21027 ms: Could not
connect to server
```

Cause: The IP address 192.168.49.2 returned by minikube may not be reachable on the Windows host machine (especially with Docker driver/Hyper-V network isolation), so accessing 192.168.49.2:31276 will time out. Knative/Kourier is not broken; the problem is that the network entry point is not working.

Under the Docker driver:

- The minikube IP address is an internal/virtual network address within the cluster
- The Windows host machine may not be able to directly route to that network segment
- The NodePort port will also be inaccessible.

port-forward / minikube service --url essentially creates a "bridging entry point" on the host machine, bypassing network isolation

Solutions: port-forwardto the host

Open a new PowerShell window and keep it running.

kubectl -n kourier-system port-forward svc/kourier 8080:80

```
PS D:\> kubectl -n kourier-system port-forward svc/kourier 8080:80
>>
Forwarding from 127.0.0.1:8080 -> 8080
Forwarding from [::1]:8080 -> 8080
Handling connection for 8080
```

Another window:

curl.exe -H "Host: helloworld-go.default.example.com" http://127.0.0.1:8080

```
PS D:\apps-data\docker> curl.exe -H "Host: helloworld-go.default.example.com" http
://127.0.0.1:8080
Hello Go Sample v1!
```

Install optional Serving extensions

Install the components needed to support HPA-class autoscaling

kubectl apply -f serving-hpa.yaml

```
PS D:\apps-data\docker> kubectl apply -f serving-hpa.yaml
deployment.apps/autoscaler-hpa created
service/autoscaler-hpa created
```

Installing Knative Eventing using YAML files

1. Install the required custom resource definitions (CRDs)

   kubectl apply -f eventing-crds.yaml

```
customresourcedefinition.apiextensions.k8s.io/channels.messaging.knative.dev created
customresourcedefinition.apiextensions.k8s.io/containersources.sources.knative.dev created
customresourcedefinition.apiextensions.k8s.io/eventpolicies.eventing.knative.dev created
customresourcedefinition.apiextensions.k8s.io/eventtransforms.eventing.knative.dev created
customresourcedefinition.apiextensions.k8s.io/eventtypes.eventing.knative.dev created
customresourcedefinition.apiextensions.k8s.io/integrationsinks.sinks.knative.dev created
customresourcedefinition.apiextensions.k8s.io/integrationsources.sources.knative.dev created
customresourcedefinition.apiextensions.k8s.io/jobsinks.sinks.knative.dev created
customresourcedefinition.apiextensions.k8s.io/parallels.flows.knative.dev created
customresourcedefinition.apiextensions.k8s.io/pingsources.sources.knative.dev created
customresourcedefinition.apiextensions.k8s.io/requestreplies.eventing.knative.dev created
customresourcedefinition.apiextensions.k8s.io/sequences.flows.knative.dev created
customresourcedefinition.apiextensions.k8s.io/sinkbindings.sources.knative.dev created
customresourcedefinition.apiextensions.k8s.io/subscriptions.messaging.knative.dev created
customresourcedefinition.apiextensions.k8s.io/triggers.eventing.knative.dev created
```

2. Install the core components of Eventing

   kubectl apply -f eventing-core.yaml

```
PS D:\apps-data\docker> kubectl apply -f eventing-core.yaml
namespace/knative-eventing created
serviceaccount/eventing-controller created
clusterrolebinding.rbac.authorization.k8s.io/eventing-controller created
clusterrolebinding.rbac.authorization.k8s.io/eventing-controller-resolver created
clusterrolebinding.rbac.authorization.k8s.io/eventing-controller-source-observer created
clusterrolebinding.rbac.authorization.k8s.io/eventing-controller-sources-controller created
clusterrolebinding.rbac.authorization.k8s.io/eventing-controller-manipulator created
clusterrolebinding.rbac.authorization.k8s.io/eventing-controller-crossnamespace-subscriber created
serviceaccount/job-sink created
clusterrolebinding.rbac.authorization.k8s.io/knative-eventing-job-sink created
serviceaccount/pingsource-mt-adapter created
clusterrolebinding.rbac.authorization.k8s.io/knative-eventing-pingsource-mt-adapter created
serviceaccount/request-reply created
clusterrolebinding.rbac.authorization.k8s.io/knative-eventing-request-reply created
serviceaccount/eventing-webhook created
clusterrolebinding.rbac.authorization.k8s.io/eventing-webhook created
rolebinding.rbac.authorization.k8s.io/eventing-webhook created
clusterrolebinding.rbac.authorization.k8s.io/eventing-webhook-resolver created
clusterrolebinding.rbac.authorization.k8s.io/eventing-webhook-podspecable-binding created
configmap/config-br-default-channel created
configmap/config-br-defaults created
configmap/default-ch-webhook created
```

3. Verify the installation

   kubectl get pods -n knative-eventing

```
PS D:\apps-data\docker> kubectl get pods -n knative-eventing
NAME                                   READY   STATUS    RESTARTS   AGE
eventing-controller-5649bc7769-ptgrb   1/1     Running   0          84s
eventing-webhook-6d99b99cff-2pfs6      1/1     Running   0          83s
job-sink-5685879598-n6scs              1/1     Running   0          84s
request-reply-0                        1/1     Running   0          83s
```

Kafka start

Deployed according to https://strimzi.io/quickstarts/

Deploy Strimzi using installation files

kubectl create namespace kafka

kubectl create -f 'https://strimzi.io/install/latest?namespace=kafka' -n kafka

```
PS D:\apps-data\docker> kubectl create namespace kafka
namespace/kafka created
PS D:\apps-data\docker> kubectl create -f 'https://strimzi.io/install/latest?names
pace=kafka' -n kafka
customresourcedefinition.apiextensions.k8s.io/kafkabridges.kafka.strimzi.io create
d
rolebinding.rbac.authorization.k8s.io/strimzi-cluster-operator-entity-operator-del
egation created
customresourcedefinition.apiextensions.k8s.io/kafkamirrormaker2s.kafka.strimzi.io
created
customresourcedefinition.apiextensions.k8s.io/kafkatopics.kafka.strimzi.io created
clusterrole.rbac.authorization.k8s.io/strimzi-cluster-operator-namespaced created
customresourcedefinition.apiextensions.k8s.io/kafkas.kafka.strimzi.io created
clusterrolebinding.rbac.authorization.k8s.io/strimzi-cluster-operator-kafka-client
-delegation created
rolebinding.rbac.authorization.k8s.io/strimzi-cluster-operator-watched created
configmap/strimzi-cluster-operator created
deployment.apps/strimzi-cluster-operator created
```

kubectl get pod -n kafka –watch

```
PS D:\apps-data\docker> kubectl get pod -n kafka --watch
NAME                                        READY   STATUS    RESTARTS   AGE
strimzi-cluster-operator-8457d7566-nwdhc    1/1     Running   0          102s
```

Create a new Kafka custom resource to get a single node Apache Kafka cluster

kubectl apply -f https://strimzi.io/examples/latest/kafka/kafka-single-node.yaml -n kafka

```
PS D:\apps-data\docker> kubectl apply -f https://strimzi.io/examples/latest/kafka/
kafka-single-node.yaml -n kafka
kafkanodepool.kafka.strimzi.io/dual-role created
kafka.kafka.strimzi.io/my-cluster created
```

Wait while Kubernetes starts the required pods, services, and so on

kubectl wait kafka/my-cluster --for=condition=Ready --timeout=300s -n kafka

```
PS D:\apps-data\docker> kubectl wait kafka/my-cluster --for=condition=Ready --time
out=300s -n kafka
kafka.kafka.strimzi.io/my-cluster condition met
```

With the cluster running, run a simple producer to send messages to a Kafka topic

kubectl -n kafka run kafka-producer -ti --image=quay.io/strimzi/kafka:0.49.1-kafka-4.1.1 --rm=true --restart=Never -- bin/kafka-console-producer.sh --bootstrap-server my-cluster-kafka-bootstrap:9092 --topic my-topic

```
PS D:\apps-data\docker> kubectl -n kafka run kafka-producer -ti --image=quay.io/st
rimzi/kafka:0.49.1-kafka-4.1.1 --rm=true --restart=Never -- bin/kafka-console-prod
ucer.sh --bootstrap-server my-cluster-kafka-bootstrap:9092 --topic my-topic
All commands and output from this session will be recorded in container logs, incl
uding credentials and sensitive information passed through the command prompt.
If you don't see a command prompt, try pressing enter.
>
[2025-12-25 13:50:32,544] WARN [Producer clientId=console-producer] The metadata r
esponse from the cluster reported a recoverable issue with correlation id 6 : {my-
topic=UNKNOWN_TOPIC_OR_PARTITION} (org.apache.kafka.clients.NetworkClient)
>Hello Strimzi!
```

And to receive them in a different terminal

*kubectl -n kafka run kafka-consumer -ti --image=quay.io/strimzi/kafka:0.49.1-kafka-4.1.1 --rm=true --restart=Never -- bin/kafka-console-consumer.sh --bootstrap-server my-cluster-kafka-bootstrap:9092 --topic my-topic --**from**-beginning*

```
PS D:\apps-data\docker> kubectl -n kafka run kafka-consumer -ti --image=quay.io/st
rimzi/kafka:0.49.1-kafka-4.1.1 --rm=true --restart=Never -- bin/kafka-console-cons
umer.sh --bootstrap-server my-cluster-kafka-bootstrap:9092 --topic my-topic --from
-beginning
All commands and output from this session will be recorded in container logs, incl
uding credentials and sensitive information passed through the command prompt.
If you don't see a command prompt, try pressing enter.


Hello Strimzi!
```

Install a default Channel (messaging) layer

1. Install the Kafka controller

   kubectl apply -f eventing-kafka-controller.yaml

```
PS D:\apps-data\docker> kubectl apply -f eventing-kafka-controller.yaml
configmap/kafka-broker-config created
configmap/kafka-channel-config created
Warning: unrecognized format "int32"
Warning: unrecognized format "int64"
customresourcedefinition.apiextensions.k8s.io/kafkachannels.messaging.knative.dev
created
customresourcedefinition.apiextensions.k8s.io/consumers.internal.kafka.eventing.kn
ative.dev created
customresourcedefinition.apiextensions.k8s.io/consumergroups.internal.kafka.eventi
ng.knative.dev created
customresourcedefinition.apiextensions.k8s.io/kafkasinks.eventing.knative.dev crea
ted
customresourcedefinition.apiextensions.k8s.io/kafkasources.sources.knative.dev cre
ated
clusterrole.rbac.authorization.k8s.io/eventing-kafka-source-observer created
configmap/config-kafka-source-defaults created
configmap/config-kafka-autoscaler created
```

2. Install the KafkaChannel data plane

   kubectl apply -f eventing-kafka-channel.yaml

```
PS D:\apps-data\docker> kubectl apply -f eventing-kafka-channel.yaml
configmap/config-kafka-channel-data-plane created
clusterrole.rbac.authorization.k8s.io/knative-kafka-channel-data-plane created
serviceaccount/knative-kafka-channel-data-plane created
clusterrolebinding.rbac.authorization.k8s.io/knative-kafka-channel-data-plane crea
ted
statefulset.apps/kafka-channel-dispatcher created
deployment.apps/kafka-channel-receiver created
service/kafka-channel-ingress created
```

3. Upgrade from the previous version

   kubectl apply -f eventing-kafka-post-install.yaml

```
PS D:\apps-data\docker> kubectl apply -f eventing-kafka-post-install.yaml
clusterrole.rbac.authorization.k8s.io/knative-kafka-controller-post-install create
d
serviceaccount/knative-kafka-controller-post-install created
clusterrole.rbac.authorization.k8s.io/knative-kafka-storage-version-migrator creat
ed
serviceaccount/knative-kafka-storage-version-migrator created
clusterrolebinding.rbac.authorization.k8s.io/knative-kafka-storage-version-migrato
r created
clusterrolebinding.rbac.authorization.k8s.io/knative-kafka-controller-post-install
 created
job.batch/kafka-controller-post-install created
job.batch/knative-kafka-storage-version-migrator created
```

Install a Broker layer

kubectl apply -f eventing-kafka-broker.yaml

```
PS D:\apps-data\docker> kubectl apply -f eventing-kafka-broker.yaml
configmap/config-kafka-broker-data-plane created
clusterrole.rbac.authorization.k8s.io/knative-kafka-broker-data-plane created
serviceaccount/knative-kafka-broker-data-plane created
clusterrolebinding.rbac.authorization.k8s.io/knative-kafka-broker-data-plane creat
ed
statefulset.apps/kafka-broker-dispatcher created
deployment.apps/kafka-broker-receiver created
service/kafka-broker-ingress created
```

Install optional Eventing extensions: Kafka Sink

kubectl apply -f eventing-kafka-sink.yaml

```
PS D:\apps-data\docker> kubectl apply -f eventing-kafka-sink.yaml
configmap/config-kafka-sink-data-plane created
clusterrole.rbac.authorization.k8s.io/knative-kafka-sink-data-plane created
serviceaccount/knative-kafka-sink-data-plane created
clusterrolebinding.rbac.authorization.k8s.io/knative-kafka-sink-data-plane created
deployment.apps/kafka-sink-receiver created
service/kafka-sink-ingress created
```

Knative Functions

Install Knative functions

docker run --rm -it ghcr.io/knative/func/func create -l node -t http myfunc

```
PS D:\apps-data\docker> docker run --rm -it ghcr.io/knative/func/func create -l no
de -t http myfunc
Unable to find image 'ghcr.io/knative/func/func:latest' locally
latest: Pulling from knative/func/func
8a08932e790a: Pull complete
250c06f7c38e: Pull complete
1074353eec0d: Pull complete
Digest: sha256:6acd4ad306964e58de15e19aab3b83467bd8073f9ba634e63a581f9965dff09e
Status: Downloaded newer image for ghcr.io/knative/func/func:latest
Created node function in /myfunc
```

Download knative func from https://github.com/knative/func/releases/tag/knative-v1.20.1

```
PS D:\apps-data\docker> .\kn-func versior
v0.47.1
```

Create a function

.\kn-func create -l go hello

```
PS D:\apps-data\docker> .\kn-func create -l go hello
Created go function in D:\apps-data\docker\hello
```

Build a function

Start the local image repository registry (5001→5000).

docker run -d -p 5001:5000 --name registry registry:2

```
PS D:\apps-data\docker\hello> docker run -d -p 5001:5000 --name registry reg
istry:2
Unable to find image 'registry:2' locally
2: Pulling from library/registry
6d464ea18732: Pull complete
3493bf46cdec: Pull complete
44cf07d57ee4: Pull complete
8e82f80af0de: Pull complete
bbbdd6c6894b: Pull complete
Digest: sha256:a3d8aaa63ed8681a604f1dea0aa03f100d5895b6a58ace528858a7b332415
373
Status: Downloaded newer image for registry:2
95883fdc89d9f17b5013cf73e687f640eb99c67973c47fed3deabaae2bb1851e
```

..\kn-func run --build --registry localhost:5001

```
PS D:\apps-data\docker\hello> ..\kn-func run --build --registry localhost:50
01
Building function image
Still building
Still building
Yes, still building
Don't give up on me
Still building
This is taking a while
Still building
Still building
Yes, still building
Don't give up on me
Still building
This is taking a while
Still building
Still building
Yes, still building
Function built: localhost:5001/hello:latest
Initializing HTTP function
listening on http port 8080
Function running on 127.0.0.1:8080
```

Verify that my function has been successfully run by using the invoke command and observing the output

..\kn-func invoke

```
PS D:\apps-data\docker\hello> ..\kn-func invoke
"POST / HTTP/1.1\r\nHost: localhost:8080\r\nAccept-Encoding: gzip\r\nContent
-Length: 25\r\nContent-Type: application/json\r\nUser-Agent: Go-http-client/
1.1\r\n\r\n{\"message\":\"Hello World\"}"
```

The output of this command below the previous image is:

```
Received request
"POST / HTTP/1.1\r\nHost: localhost:8080\r\nAccept-Encoding: gzip\r\nContent
-Length: 25\r\nContent-Type: application/json\r\nUser-Agent: Go-http-client/
1.1\r\n\r\n{\"message\":\"Hello World\"}"
```

Build a function

..\kn func build

```
PS D:\apps-data\docker\hello> ..\kn-func build
A registry for function images is required. For example, 'docker.io/tigertea
m'.
X Sorry, your reply was invalid: registry required
? Registry for function images: localhost:5001
Note: building a function the first time will take longer than subsequent bu
ilds
Building function image
Still building
Still building
Yes, still building
Don't give up on me
Still building
This is taking a while
Still building
Still building
Function built: localhost:5001/hello:latest
```

Knative Serving

hello.yaml

```
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: hello
spec:
  template:
    spec:
      containers:
        - image: ghcr.io/knative/helloworld-go:latest
          ports:
            - containerPort: 8080
          env:
            - name: TARGET
              value: "World"
```

Deploy the Knative Service

```
kubectl apply -f hello.yaml
```

```
PS D:\apps-data\docker> kubectl apply -f hello.yaml
Warning: Kubernetes default value is insecure, Knative may default this to s
ecure in a future release: spec.template.spec.containers[0].securityContext.
allowPrivilegeEscalation, spec.template.spec.containers[0].securityContext.c
apabilities, spec.template.spec.containers[0].securityContext.runAsNonRoot,
spec.template.spec.containers[0].securityContext.seccompProfile
service.serving.knative.dev/hello created
```

Because kn wasn't downloaded, I used kubectl to perform autoscaling. During the process, I encountered the error "GET / HTTP/1.1\r\nHost: hello.default.example.com\r\nAccept: */*\r\nUser-Agent: curl/8.16.0\r\n\r\n". This was because port-forward wasn't enabled at the time. So, I created a new PowerShell window and entered `kubectl -n kourier-system port-forward svc/kourier 18080:80`.

```
PS D:\apps-data\docker> kubectl -n kourier-system port-forward svc/kourier 1
8080:80
Forwarding from 127.0.0.1:18080 -> 8080
Forwarding from [::1]:18080 -> 8080
Handling connection for 18080
```

Autoscaling

View a list of Knative Services

```
kubectl get ksvc
```

```
PS D:\apps-data\docker> kubectl get ksvc
NAME           URL                                        LATESTCREATED         L
ATESTREADY           READY   REASON
hello          http://hello.default.example.com           hello-00001           h
ello-00001           True
helloworld-go  http://helloworld-go.default.example.com   helloworld-go-00001   h
elloworld-go-00001   True
```

Access my Knative Service by opening the previous URL in my browser

$URL = kubectl get ksvc hello -o jsonpath="{.status.url}"

echo "Accessing URL $URL"

curl.exe -H "Host: hello.default.example.com" http://127.0.0.1:18080

```
PS D:\apps-data\docker> $URL = kubectl get ksvc hello -o jsonpath="{.status.
url}"
PS D:\apps-data\docker> echo "Accessing URL $URL"
Accessing URL http://hello.default.example.com
PS D:\apps-data\docker> curl.exe $URL
curl: (6) Could not resolve host: hello.default.example.com
PS D:\apps-data\docker> curl.exe -H "Host: hello.default.example.com" http:/
/127.0.0.1:8080
"GET / HTTP/1.1\r\nHost: hello.default.example.com\r\nAccept: */*\r\nUser-Ag
ent: curl/8.16.0\r\n\r\n"
PS D:\apps-data\docker> curl.exe -H "Host: hello.default.example.com" http:/
/127.0.0.1:18080
Hello World!
```

Watch the pods and see how they scale to zero after traffic stops going to the URL

kubectl get pod -l serving.knative.dev/service=hello -w

```
PS D:\apps-data\docker> kubectl get pod -l serving.knative.dev/service=hello -w
NAME                                     READY   STATUS    RESTARTS   AGE
hello-00001-deployment-dd585f869-6tfvq   2/2     Running   0          25s
hello-00001-deployment-dd585f869-6tfvq   2/2     Terminating   0          62s
hello-00001-deployment-dd585f869-6tfvq   2/2     Terminating   0          62s
hello-00001-deployment-dd585f869-6tfvq   1/2     Terminating   0          84s
hello-00001-deployment-dd585f869-6tfvq   0/2     Completed     0          89s
hello-00001-deployment-dd585f869-6tfvq   0/2     Completed     0          90s
hello-00001-deployment-dd585f869-6tfvq   0/2     Completed     0          90s
hello-00001-deployment-dd585f869-6tfvq   0/2     Completed     0          90s
```

Traffic splitting

hello.yaml

```
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: hello
spec:
  template:
    spec:
      containers:
        - image: ghcr.io/knative/helloworld-go:latest
          ports:
            - containerPort: 8080
          env:
            - name: TARGET
              value: "Knative"
```

Deploy the updated version of my Knative Service

kubectl apply -f .\yaml\hello.yaml

```
PS D:\apps-data\docker> kubectl apply -f .\yaml\hello.yaml
Warning: Kubernetes default value is insecure, Knative may default this to secure
in a future release: spec.template.spec.containers[0].securityContext.allowPrivile
geEscalation, spec.template.spec.containers[0].securityContext.capabilities, spec.
template.spec.containers[0].securityContext.runAsNonRoot, spec.template.spec.conta
iners[0].securityContext.seccompProfile
service.serving.knative.dev/hello configured
```

Access the Knative Service on the browser

$URL = kubectl get ksvc hello -o jsonpath="{.status.url}"

echo "Accessing URL $URL"

curl.exe -H "Host: hello.default.example.com" http://127.0.0.1:18080

```
PS D:\apps-data\docker> $URL = kubectl get ksvc hello -o jsonpath="{.status.url}"
PS D:\apps-data\docker> echo "Accessing URL $URL"
Accessing URL http://hello.default.example.com
PS D:\apps-data\docker> curl.exe -H "Host: hello.default.example.com" http://127.0
.0.1:18080
Hello Knative!
```

View a list of Revisions

kubectl get revisions

```
PS D:\apps-data\docker> kubectl get revisions
NAME                  CONFIG NAME     GENERATION   READY   REASON   ACTUAL REPLICA
S   DESIRED REPLICAS
hello-00001           hello           1            True             0
    0
hello-00002           hello           2            True             0
    0
helloworld-go-00001   helloworld-go   1            True             0
    0
```

Add to the bottom of the existing hello.yaml

```
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: hello
spec:
  template:
    spec:
      containers:
        - image: ghcr.io/knative/helloworld-go:latest
          ports:
            - containerPort: 8080
          env:
            - name: TARGET
              value: "Knative"
  traffic:
  - latestRevision: true
    percent: 50
```

- latestRevision: false

percent: 50

revisionName: hello-00001

Apply the YAML

kubectl apply -f .\yaml\hello.yaml

```
PS D:\apps-data\docker> kubectl apply -f .\yaml\hello.yaml
Warning: Kubernetes default value is insecure, Knative may default this to secure
in a future release: spec.template.spec.containers[0].securityContext.allowPrivile
geEscalation, spec.template.spec.containers[0].securityContext.capabilities, spec.
template.spec.containers[0].securityContext.runAsNonRoot, spec.template.spec.conta
iners[0].securityContext.seccompProfile
service.serving.knative.dev/hello configured
```

Verify the traffic split and list the Revisions

.\kn revisions list

```
PS D:\apps-data\docker> .\kn revisions list
NAME                  SERVICE         TRAFFIC   TAGS   GENERATION   AGE     CONDIT
IONS   READY   REASON
hello-00002           hello           50%              2            9m31s   3 OK /
 4     True
hello-00001           hello           50%              1            23m     3 OK /
 4     True
helloworld-go-00001   helloworld-go   100%             1            48m     3 OK /
 4     True
```

Access the Service URL from the terminal multiple times to see the traffic being split between the Revisions

curl.exe -H "Host: hello.default.example.com" http://127.0.0.1:18080

```
PS D:\apps-data\docker> curl.exe -H "Host: hello.default.example.com" http://127.0
.0.1:18080
Hello World!
PS D:\apps-data\docker> curl.exe -H "Host: hello.default.example.com" http://127.0
.0.1:18080
Hello Knative!
PS D:\apps-data\docker> curl.exe -H "Host: hello.default.example.com" http://127.0
.0.1:18080
Hello World!
PS D:\apps-data\docker> curl.exe -H "Host: hello.default.example.com" http://127.0
.0.1:18080
Hello World!
PS D:\apps-data\docker> curl.exe -H "Host: hello.default.example.com" http://127.0
.0.1:18080
Hello Knative!
```

Knative Eventing

Download from https://github.com/knative/eventing/releases/download/knative-v1.20.0/in-memory-channel.yaml

And apply it.

kubectl apply -f in-memory-channel.yaml

```
PS D:\apps-data\docker> kubectl apply -f in-memory-channel.yaml
serviceaccount/imc-controller created
clusterrolebinding.rbac.authorization.k8s.io/imc-controller created
rolebinding.rbac.authorization.k8s.io/imc-controller created
clusterrolebinding.rbac.authorization.k8s.io/imc-controller-resolver created
serviceaccount/imc-dispatcher created
clusterrolebinding.rbac.authorization.k8s.io/imc-dispatcher created
rolebinding.rbac.authorization.k8s.io/imc-dispatcher-tls-role-binding created
role.rbac.authorization.k8s.io/imc-dispatcher-tls-role created
configmap/config-imc-event-dispatcher created
deployment.apps/imc-controller created
service/inmemorychannel-webhook created
service/imc-dispatcher created
deployment.apps/imc-dispatcher created
Warning: unrecognized format "int32"
Warning: unrecognized format "int64"
customresourcedefinition.apiextensions.k8s.io/inmemorychannels.messaging.knative.d
ev created
clusterrole.rbac.authorization.k8s.io/imc-addressable-resolver created
```

.\kn broker create example-broker -n default

.\kn broker list -n default

```
PS D:\apps-data\docker> .\kn broker create example-broker -n default
Broker 'example-broker' successfully created in namespace 'default'.
PS D:\apps-data\docker> .\kn broker list -n default
NAME             URL   AGE   CONDITIONS   READY       REASON
example-broker         9s    0 OK / 0     <unknown>   <unknown>
```

At this point, it shows that there are no URLs under the broker, and Ready=unknown. Check using the following command.

kubectl get pods -n knative-eventing | findstr broker

kubectl describe broker example-broker -n default

The Broker specified is MTChannelBasedBroker, which also points to config-br-default-channel. However, there are no MT Broker controller/data plane components running in the cluster (knative-eventing only contains kafka-broker-receiver, not mt-broker-controller/broker-ingress/broker-filter). Therefore, it will never generate status/conditions and will always show <unknown>. Therefore, the mt-channel-broker component needs to be installed.

```
PS D:\apps-data\docker> kubectl get pods -n knative-eventing | findstr broker
kafka-broker-receiver-785787d5bc-k9t89    1/1     Running   0          130m
PS D:\apps-data\docker> kubectl describe broker example-broker -n default
Name:         example-broker
Namespace:    default
Labels:       <none>
Annotations:  eventing.knative.dev/broker.class: MTChannelBasedBroker
              eventing.knative.dev/creator: minikube-user
              eventing.knative.dev/lastModifier: minikube-user
API Version:  eventing.knative.dev/v1
Kind:         Broker
Metadata:
  Creation Timestamp:  2025-12-25T16:00:47Z
  Generation:          1
  Resource Version:    135512
  UID:                 edd825ac-5719-466b-bc55-bb8d2b88b73f
Spec:
  Config:
    API Version:  v1
    Kind:         ConfigMap
    Name:         config-br-default-channel
    Namespace:    knative-eventing
  Delivery:
    Backoff Delay:   PT0.2S
    Backoff Policy:  exponential
    Retry:           10
Events:              <none>
PS D:\apps-data\docker>
```

Download mt-channel-broker.yaml from https://github.com/knative/eventing/releases/download/knative-v1.20.0/mt-channel-broker.yaml and then apply it.

kubectl apply -f D:\apps-data\docker\mt-channel-broker.yaml

```
PS D:\apps-data\docker> kubectl apply -f D:\apps-data\docker\mt-channel-broker.yam
l
clusterrole.rbac.authorization.k8s.io/knative-eventing-mt-channel-broker-controlle
r created
clusterrole.rbac.authorization.k8s.io/knative-eventing-mt-broker-filter created
role.rbac.authorization.k8s.io/mt-broker-filter created
serviceaccount/mt-broker-filter created
clusterrole.rbac.authorization.k8s.io/knative-eventing-mt-broker-ingress created
role.rbac.authorization.k8s.io/mt-broker-ingress created
serviceaccount/mt-broker-ingress-oidc created
serviceaccount/mt-broker-ingress created
clusterrolebinding.rbac.authorization.k8s.io/eventing-mt-channel-broker-controller
 created
clusterrolebinding.rbac.authorization.k8s.io/knative-eventing-mt-broker-filter cre
ated
rolebinding.rbac.authorization.k8s.io/mt-broker-filter created
clusterrolebinding.rbac.authorization.k8s.io/knative-eventing-mt-broker-ingress cr
eated
rolebinding.rbac.authorization.k8s.io/mt-broker-ingress created
```

kubectl get pods -n knative-eventing | findstr -i "broker mt-broker ingress filter"

```
PS D:\apps-data\docker> kubectl get pods -n knative-eventing | findstr -i "broker
mt-broker ingress filter"
kafka-broker-receiver-785787d5bc-k9t89    1/1     Running   0          142m
mt-broker-controller-5d4f49dbcf-899st     1/1     Running   0          2m34s
mt-broker-filter-664b8fdcfd-qvgr8         1/1     Running   0          2m34s
mt-broker-ingress-75565b46bd-nk7pl        1/1     Running   0          2m34s
```

kubectl wait broker/example-broker -n default --for=condition=Ready=True --timeout=180s

.\kn broker list -n default

```
PS D:\apps-data\docker> kubectl wait broker/example-broker -n default --for=condit
ion=Ready=True --timeout=180s
broker.eventing.knative.dev/example-broker condition met
PS D:\apps-data\docker> .\kn broker list -n default
NAME             URL
                 AGE   CONDITIONS   READY   REASON
example-broker   http://broker-ingress.knative-eventing.svc.cluster.local/default/
example-broker   21m   7 OK / 7     True
```

Using a Knative Service as a source

cloudevents-player.yaml

```
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: cloudevents-player
spec:
  template:
    metadata:
      annotations:
        autoscaling.knative.dev/min-scale: "1"
    spec:
      containers:
        - image: quay.io/ruben/cloudevents-player:latest
```

Apply the YAML file

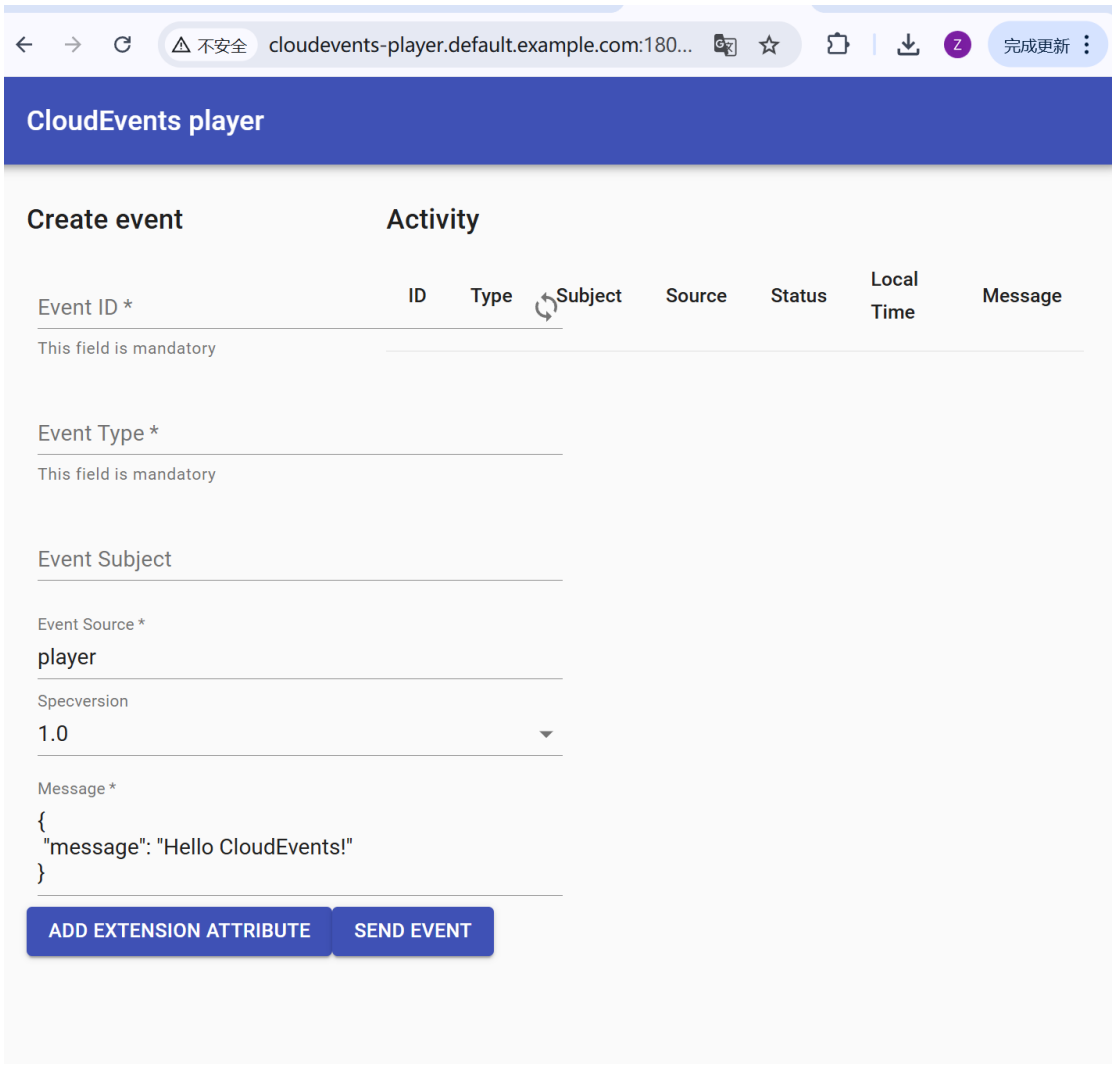
kubectl apply -f .\yaml\cloudevents-player.yaml

```
PS D:\apps-data\docker> kubectl apply -f .\yaml\cloudevents-player.yaml
Warning: Kubernetes default value is insecure, Knative may default this to secure
in a future release: spec.template.spec.containers[0].securityContext.allowPrivile
geEscalation, spec.template.spec.containers[0].securityContext.capabilities, spec.
template.spec.containers[0].securityContext.runAsNonRoot, spec.template.spec.conta
iners[0].securityContext.seccompProfile
service.serving.knative.dev/cloudevents-player created
```

```
PS D:\apps-data\docker> kubectl apply -f .\yaml\cloudevents-player-binding.yaml
sinkbinding.sources.knative.dev/ce-player-binding created
```

Open the following directory with administrator privileges:

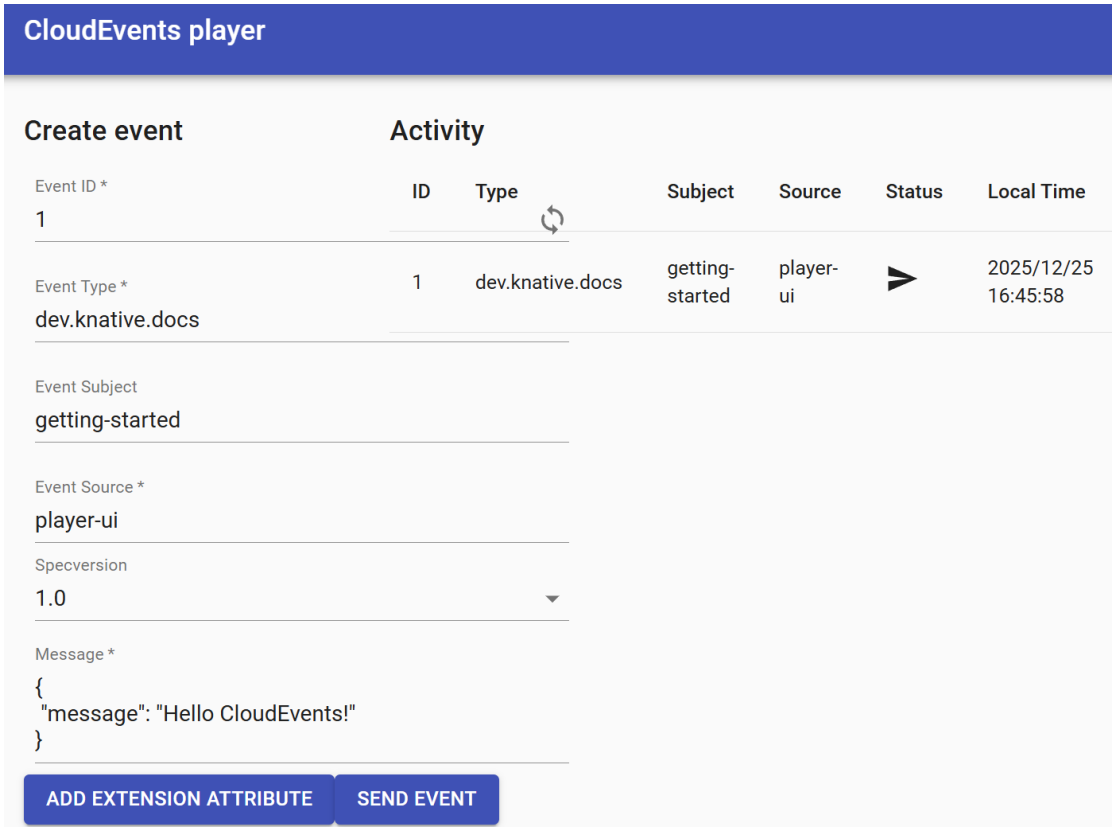
C:\Windows\System32\drivers\etc\hosts. Add the following line to the end: 127.0.0.1 cloudevents-player.default.example.com. Then open your browser and enter http://cloudevents-player.default.example.com:18080/



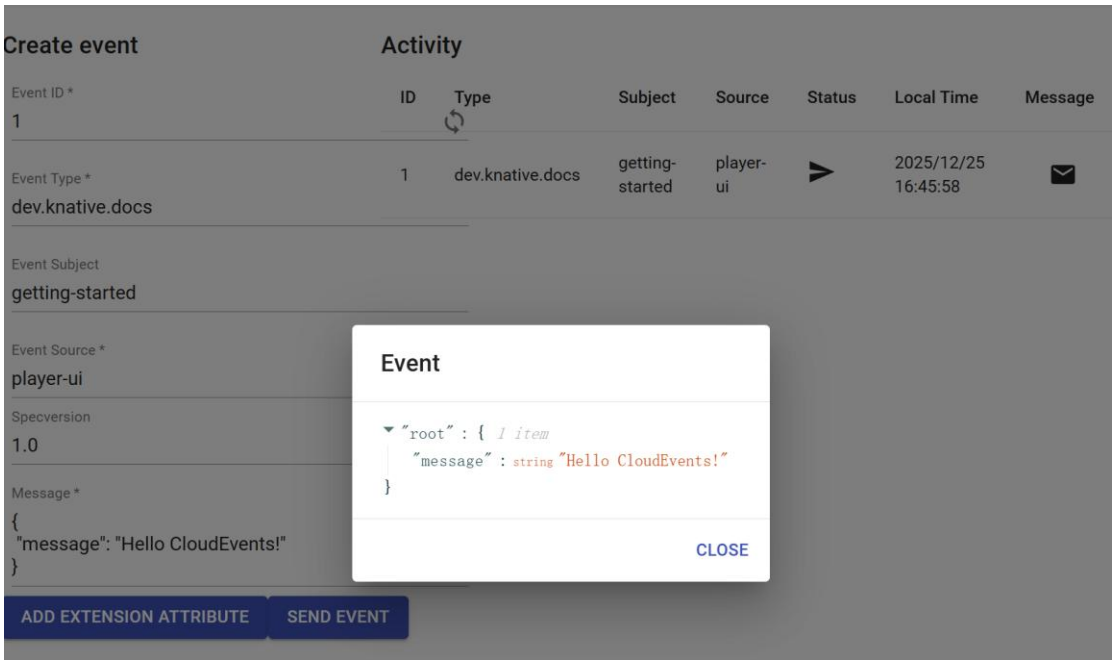
These fields mean:

Field	Description
Event ID	A unique ID. Click the loop icon to generate a new one.
Event Type	An event type.
Event Source	An event source.
Specversion	Demarcates which CloudEvents spec you're using (should always be 1.0).
Message	The data section of the CloudEvent, a payload which is carrying the data you care to be delive

Sending an event using the CloudEvents Player interface



Clicking the icon like envelop shows you the CloudEvent as the Broker sees it.



Send events using the command line

```
curl.exe -X POST http://127.0.0.1:18080/ `
  -H "Host: cloudevents-player.default.example.com" `
  -H "Ce-Id: 123456789" `
```

```
  -H "Ce-Specversion: 1.0" `
  -H "Ce-Type: some-type" `
  -H "Ce-Source: command-line" `
  -H "Content-Type: application/json" `
  -d "{""msg"":"" Hello CloudEvents! ""}"
```

```
PS D:\apps-data\docker> curl.exe -X POST http://127.0.0.1:18080/ `
>>   -H "Host: cloudevents-player.default.example.com" `
>>   -H "Ce-Id: 123456789" `
>>   -H "Ce-Specversion: 1.0" `
>>   -H "Ce-Type: some-type" `
>>   -H "Ce-Source: command-line" `
>>   -H "Content-Type: application/json" `
>>   -d "{""msg"":"" Hello CloudEvents! ""}"
PS D:\apps-data\docker> curl.exe cloudevents-player.default.example.com/messages
curl: (7) Failed to connect to cloudevents-player.default.example.com port 80 afte
r 2036 ms: Could not connect to server
```

Entering `kubectl logs -n default cloudevents-player-00002-deployment-cd76b487-tg8gl --tail=200` to view the logs of cloudevents-player reveals:

```
Caused by: com.fasterxml.jackson.core.JsonParseException: Unexpected character ('m
' (code 109)): was expecting double-quote to start field name
```

This usually happens when the data actually looks like this: {msg:Hello} (the field name isn't in quotes), or the data is stored as a string or incorrectly escaped, resulting in it not being standard JSON.

My use of -d "{""msg"":""Hello CloudEvents!""}" looks correct in PowerShell, but when actually passed, it might produce unexpected content (especially due to different quotes/escaping).

```
PS D:\apps-data\docker> curl.exe -i http://127.0.0.1:18080/ `
>>   -H "Host: cloudevents-player.default.example.com" `
>>   -H "Content-Type: application/json" `
>>   -H "Ce-Id: 999" `
>>   -H "Ce-Specversion: 1.0" `
>>   -H "Ce-Type: some-type" `
>>   -H "Ce-Source: command-line" `
>>   -d '{"msg":"Hello CloudEvents!"}'
HTTP/1.1 202 Accepted
content-length: 0
date: Thu, 25 Dec 2025 17:16:37 GMT
server: envoy
x-envoy-decorator-operation: cloudevents-player-00002-private.default.svc.cluster.
local:80/*
x-envoy-upstream-service-time: 37
```

Using Triggers and sinks

ce-trigger.yaml

```
apiVersion: eventing.knative.dev/v1
kind: Trigger
metadata:
  name: cloudevents-trigger
  annotations:
```

```
    knative-eventing-injection: enabled
spec:
  broker: example-broker
  subscriber:
    ref:
      apiVersion: serving.knative.dev/v1
      kind: Service
      name: cloudevents-player
```

Create the Trigger

kubectl apply -f ce-trigger.yaml

```
PS D:\apps-data\docker> kubectl apply -f .\yaml\ce-trigger.yaml
trigger.eventing.knative.dev/cloudevents-trigger created
```

Delete the existing Trigger

kn trigger delete cloudevents-trigger

Add a Trigger that listens for a certain CloudEvent Type

kn trigger create cloudevents-player-filter --sink cloudevents-player --broker example-broker --filter type=some-type

```
PS D:\apps-data\docker>  .\kn trigger delete cloudevents-trigger
Trigger 'cloudevents-trigger' deleted in namespace 'default'.
PS D:\apps-data\docker> .\kn trigger create cloudevents-player-filter --sink cloud
events-player  --broker example-broker --filter type=some-type
Trigger 'cloudevents-player-filter' successfully created in namespace 'default'.
```