Deploy Strimzi in the Developer Sandbox

The developer sandbox does not include an AMQ-Streams / Strimzi operator, and as a user, you don’t have permission to deploy operators. This guide helps you deploy an all-in-one Kafka server and ui for your development activities.

Access the developer sandbox

You can access a free-to-use Openshift environment. Follow the instructions in this article: [How to access the Developer Sandbox for Red Hat OpenShift]

Deploy a Kafka server

Using the console

From Openshift’s Developer view, follow these steps:

1) Developer → +Add → Import Yaml

Unset

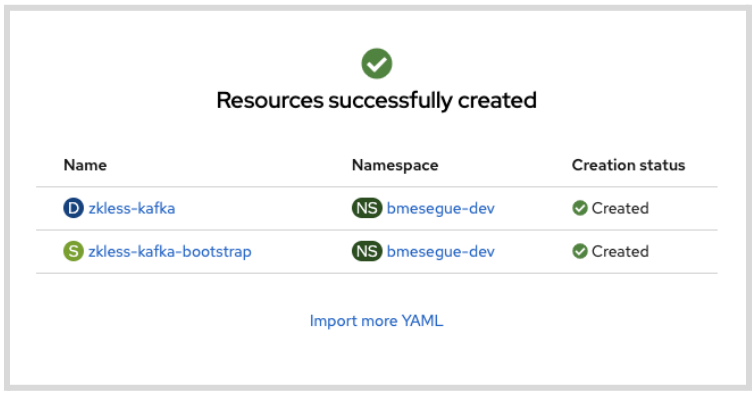
```
---
kind: Deployment
apiVersion: apps/v1
metadata:
  name: zkless-kafka
spec:
  replicas: 1
  selector:
    matchLabels:
      app: zkless-kafka
  template:
    metadata:
      labels:
        app: zkless-kafka
    spec:
      containers:
        - resources:
```

```
            limits:
              cpu: 250m
              memory: 512Mi
          terminationMessagePath: /dev/termination-log
          name: zkless-kafka
          command:
            - /bin/sh
            - '-c'
            - >-
              export CLUSTER_ID=$(bin/kafka-storage.sh random-uuid) &&
              bin/kafka-storage.sh format -t $CLUSTER_ID -c
              config/kraft/server.properties && bin/kafka-server-start.sh
              config/kraft/server.properties --override
              advertised.listeners=${KAFKA_ADVERTISED_LISTENERS}
          env:
            - name: LOG_DIR
              value: /tmp/logs
            - name: KAFKA_ADVERTISED_LISTENERS
              value: 'PLAINTEXT://zkless-kafka-bootstrap:9092'
          ports:
            - containerPort: 9092
              protocol: TCP
          imagePullPolicy: IfNotPresent
          terminationMessagePolicy: File
          image: 'registry.redhat.io/amq-streams/kafka-35-rhel8:2.5.0'
      restartPolicy: Always
      terminationGracePeriodSeconds: 30
      dnsPolicy: ClusterFirst
      securityContext: {}
      schedulerName: default-scheduler
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 25%
      maxSurge: 25%
  revisionHistoryLimit: 10
  progressDeadlineSeconds: 600
---
apiVersion: v1
kind: Service
metadata:
  name: zkless-kafka-bootstrap
spec:
  ports:
```

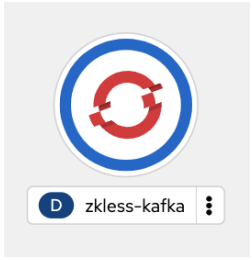
```
    - port: 9092
      protocol: TCP
      targetPort: 9092
  selector:
    app: zkless-kafka
  type: ClusterIP
```

2) Click Create

The screen will show:



You should see the server created in your Topology view:

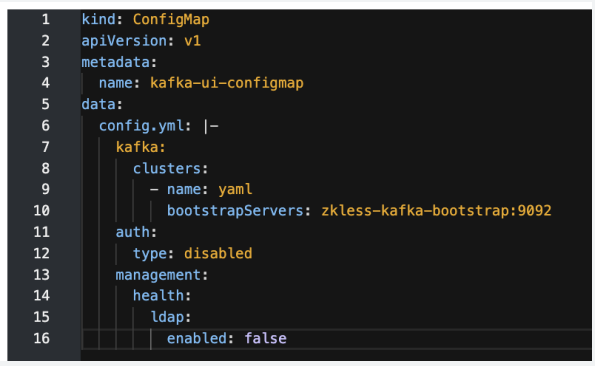


Deploy a Kafka UI

Create UI client configuration

Create a new ConfigMap with the following configuration:

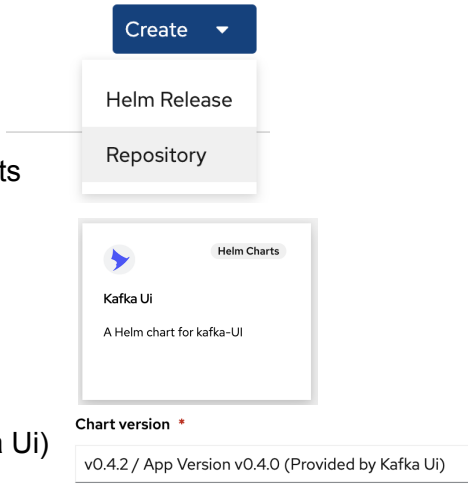
```
Unset
kind: ConfigMap
apiVersion: v1
metadata:
 name: kafka-ui-configmap
data:
 config.yml: |-
  kafka:
   clusters:
    - name: yaml
      bootstrapServers: zkless-kafka-bootstrap:9092
  auth:
   type: disabled
  management:
   health:
    ldap:
     enabled: false
```



Using the console

From Openshift's Developer view, follow these steps:

3) Developer → Helm → Create → Repository
    a) Name: kafka-ui
    b) URL: https://provectus.github.io/kafka-ui-charts
4) Click Create
5) Developer → Helm → Create → Helm Release
    a) Filter by Kafka
    b) Choose the "Kafka UI" chart
6) Click Create
7) Select the latest Chart version:
    - 0.7.5 / App Version v0.7.1 (Provided by Kafka Ui)
8) Configure YAML
    a) Delete the default YAML definition
    b) Use instead:



```
Unset
        yamlApplicationConfigConfigMap:
          keyName: config.yml
          name: kafka-ui-configmap
```

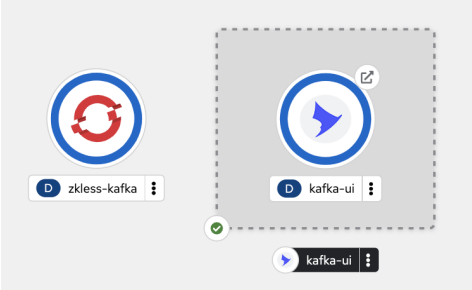
```
1  yamlApplicationConfigConfigMap:
2    keyName: config.yml
3    name: kafka-ui-configmap
```

9) Click Create

Create a Route

1) Administrator → Networking → Routes → Create
    a) Name: **kafka-ui**
    b) Service: **kafka-ui**
    c) Target Port: **80** → **http (TCP)**
2) Click Create

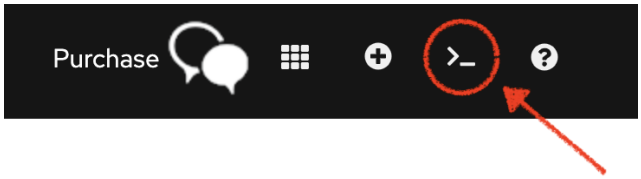
You should now see both the Kafka server and the Kafka UI from your Topology view:



Using CLI

Alternatively, you could use a terminal from where you can use the '**helm**' command.

1) Open a terminal



If the ConfigMap containing the client configuration hasn't been created yet, follow these steps:

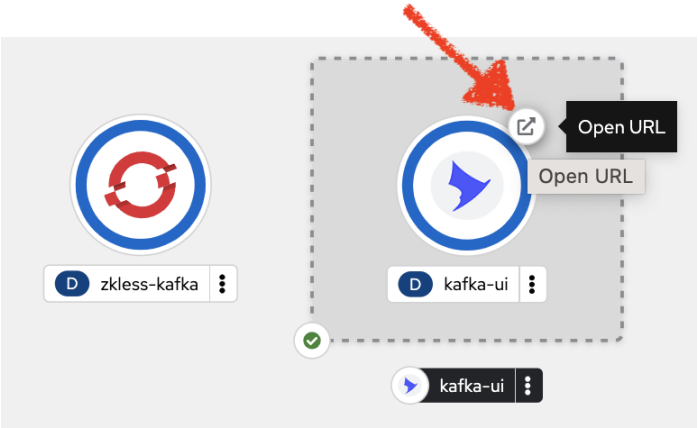
1) Create a ConfigMap (as described above)

Install the Kafka UI by executing the commands below:

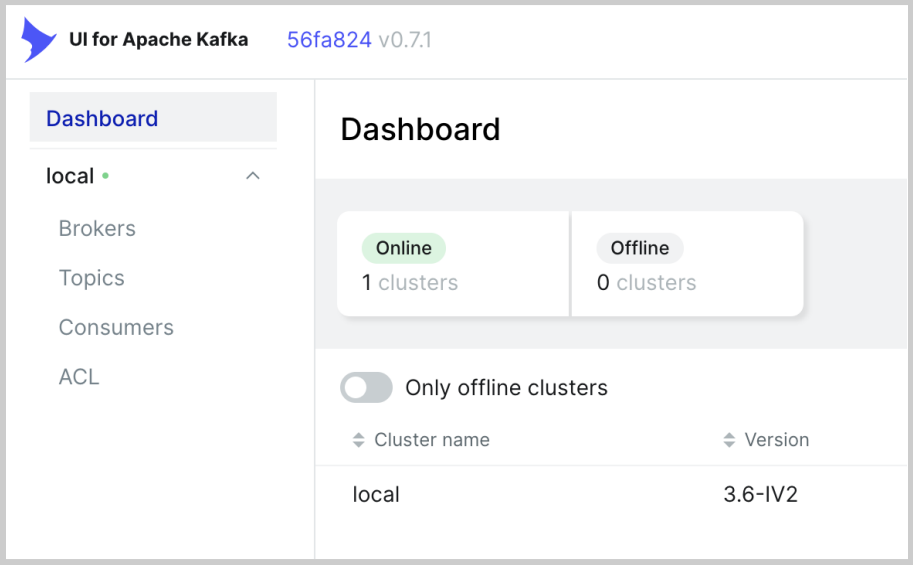
```
Unset
helm repo add kafka-ui https://provectus.github.io/kafka-ui-charts
helm install --set
yamlApplicationConfigConfigMap.name="kafka-ui-configmap",yamlApplicationConfigC
onfigMap.keyName="config.yml" kafka-ui kafka-ui/kafka-ui
```

Open the Kafka UI

From your Topology view, click on the route to open the Web UI to manage your Kafka instance:

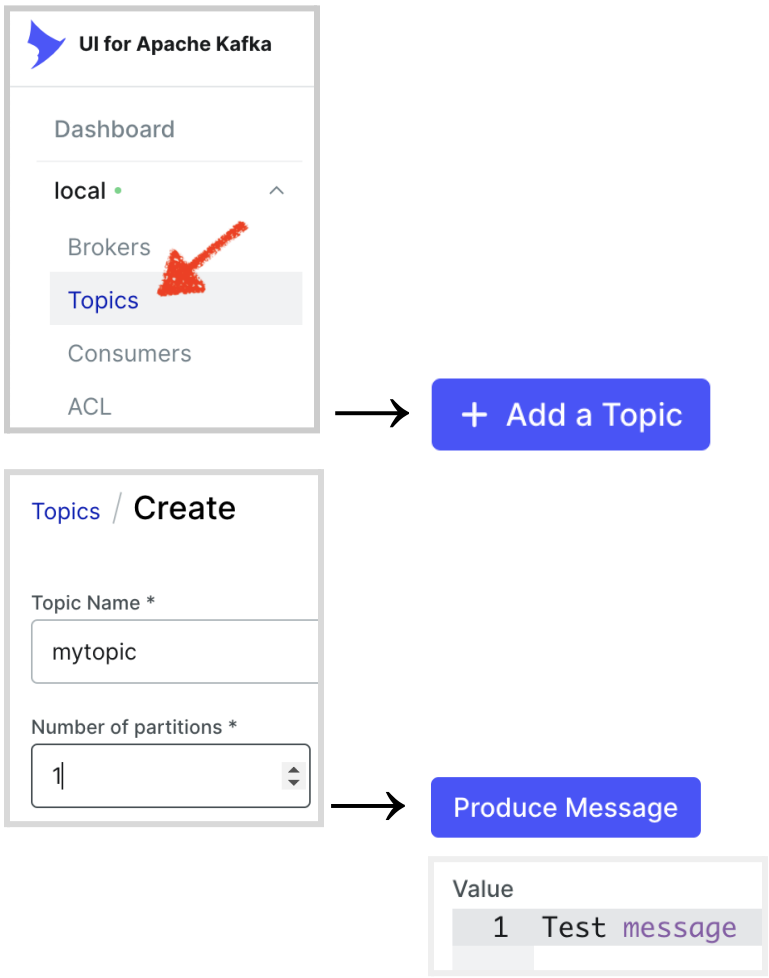


You should see the Dashboard as follows:



Test Kafka

From the Web UI, you can create a topic, and send a test message to validate all works as expected:



You should see your message listed in your browser: