Solution Assurance

How to set up IBM Event Streams with MongoDB on IBM Z

IBM

Contents

How to set up IBM Event Streams with MongoDB on IBM Z and IBM LinuxONE

This guide provides detailed information about how to set up IBM Event Streams by using IBM Cloud Pak for Integration. It describes a real-time scenario which shows how to transfer data between two databases (MongoDB) using Kafka Connect and Connectors.
In addition, it also explains the MongoDB setup on the OpenShift environment, using Helm Chart for POC purposes.

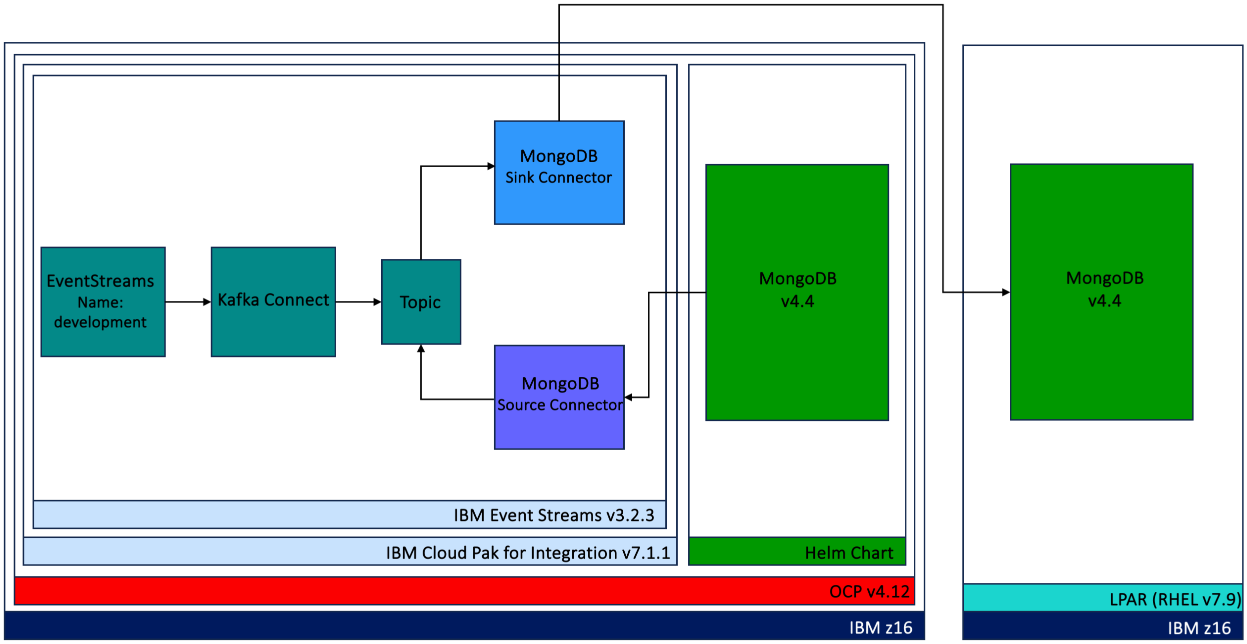
A PDF is also available here.

Chapter 1. Getting started

This guide explains how to build a proof-of-concept (PoC) use-case that uses IBM Event Streams as part of IBM Cloud Pak for Integration running on Red Hat OpenShift Container Platform (OpenShift) on IBM Z and IBM LinuxONE.

This solution updates a remote database whenever a change happens in the database that is running in OpenShift.

The final architectural setup for the PoC looks like this:



In addition, this guide shows you how set up PoC MongoDB to run on OpenShift that is installed on top of IBM Z and LinuxONE.

This guide is divided into these steps:

- The OpenShift environment prerequisites and resource planning
- Setting up IBM Cloud Pak for Integration
- Setting up MongoDB
- Setting up the solution including IBM Event Streams, Kafka Cluster, Kafka Connect, and Connectors

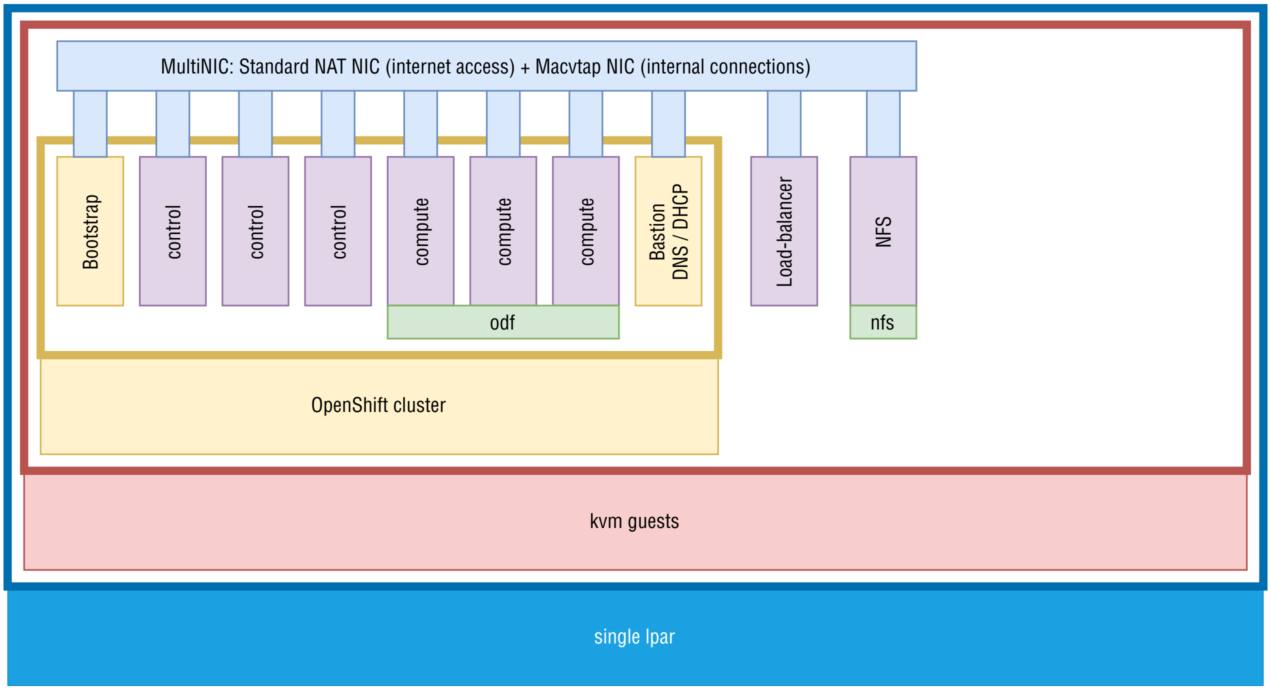
Chapter 2. Prerequisites and planning

This topic describes the architecture and resource planning required to run this scenario.

Make sure that your environment meets the requirements needed to install and run the PoC.

The prerequisites are a Red Hat OpenShift Container Platform (OpenShift) cluster with Red Hat OpenShift Data Foundation and enough resources (CPU, Memory, Storage) to run the PoC.

OpenShift architecture



Components:

- Bastion:
  It runs the DNS and DHCP server for the OpenShift cluster. It is used to run OC commands that manage OpenShift and perform other administrative tasks.
- Loadbalancer:
  It is used to run HAProxy and is able to connect to the OpenShift web interface.

For PoC purposes you can use an OpenShift setup running inside KVM guests, which are on top of a single LPAR, similar to the setup in the image above. For production environments, refer to the Reference Architecture Guide.

Optional: For more information about Red Hat Enterprise Linux (RHEL) and Red Hat High Availability (HA), refer to Use Red Hat HA to make OpenShift highly available.

Storage, compute, and memory

The OpenShift cluster control-nodes don't require any extra resources for the scenario to run successfully:

used by	vCPU (base)	Memory	Storage
control-node (3x)	8	16GiB	120GiB

Calculating the correct compute-node resources is more complicated because you need to consider all the components and the deployment itself.

Used by	vCPU (base)	Memory	Storage	Note
compute-node (no workload) (3x)	6	16GiB	120GiB	
ODF Operator & Deployment (3x)	10	24GiB	None	1TiB SCSI disk
CP4I Operator (1x)	0.2	1GiB	None	
CP4I Platform UI (1x)	2.2	4.6GiB	40GiB	
CP4I Event Streams Operator (1x)	0.2	1GiB	None	
CP4I Event Streams Deployment (1x)	2.8	6GiB	2GiB	
CP4I Cloud Pak foundational services Operator (1x)	0.1	0.2GiB	None	
MongoDB Helm Chart (1x)	0.1	1GiB	10GiB	

Note: The components marked with (3x) indicate that the resource amount must be applied to each compute-node (This assumes that the resources are evenly split across the compute-nodes). (1x) means that this is the total number of resources that the component requires.

Total amount of resource per compute-node (roughly):

Used by	vCPU	Memory	Storage
compute-node 1	~18	~30GiB	120GiB
compute-node 2	~18	~30GiB	120GiB
compute-node 3	~18	~30GiB	120GiB

- IBM Cloud Pak for Integration is modular and the amount of storage required depends on which modules you install. A table with how much storage each component needs as a minimum is found in Compute resources for development environments. (Make sure to select the correct version in the official documentation.)
- Red Hat OpenShift Data Foundation (ODF) is used as the storage backend of IBM Cloud Pak for Integration. To install ODF you can follow the Product Documentation for Red Hat OpenShift Data Foundation. For this PoC, three 1TiB SCSI disks were used (that are mirrored to each other by ODF). According to the required resources for IBM Cloud Pak for Integration you can use less storage. Consider having some spare storage so that you can install more components later on.
- The following references are used to calculate the resource values:
  - **Red Hat OpenShift Container Platform installation**: Preferred resource requirements for installing a cluster with RHEL KVM on IBM Z and IBM LinuxONE
  - **Red Hat OpenShift Data Foundation**: Resource requirements for IBM Z and LinuxONE infrastructure
  - **IBM Cloud Pak for Integration**: Compute resources for development environments
  - **IBM Event Streams**: Prerequisites

The user-provisioned infrastructure around OpenShift has a few more KVM guests that used few resources to work:

used by	**vCPU**	**Memory**	**Storage**
NFS-server	4	8GiB	100GiB
Bastion	4	8GiB	100GiB
Loadbalancer	4	8GiB	100GiB

- A NFS server is used for the OpenShift internal docker image registry and as storage for the MongoDB later on. To set up a NFS server, follow these NFS Server instructions.
- The Bastion runs a DNS-server, DHCP-server and is used to run cli admin commands against OpenShift.
- The load-balancer allows the access to the OpenShift cluster externally and is also used to run commands internally against OpenShift.

Chapter 3. Installing IBM Cloud Pak for Integration

Find out how to set up IBM Cloud Pak for Integration on Red Hat Openshift Container Platform 4.12 (OpenShift).

Adding the catalog source to OpenShift Cluster

To add the **IBM Cloud Pak for Integration operator** under the **OperatorHub**, you need to add the **Catalog Source** to your OpenShift cluster.

1. In the OpenShift cluster, create a dedicated project for the IBM Cloud Pak for Integration installation in the OpenShift cluster. For this use-case, the project name is `cloudpack4integration`.
2. Make sure you have **cluster administrator** role to perform the setup.
3. Copy the **Catalog Source** to a file called `catalog.yaml` on your bastion server:

```
apiVersion: operators.coreos.com/v1alpha1
kind: CatalogSource
metadata:
name: ibm-operator-catalog
namespace: openshift-marketplace
annotations:
   olm.catalogImageTemplate: "icr.io/cpopen/ibm-operator-catalog:v{kube_major_version}.
{kube_minor_version}"
spec:
displayName: IBM Operator Catalog
publisher: IBM
sourceType: grpc
image: icr.io/cpopen/ibm-operator-catalog:latest
updateStrategy:
   registryPoll:
      interval: 45m
```

4. Run this command:

```
oc apply -f catalog.yaml
```

Output:

```
catalogsource.operators.coreos.com/ibm-operator-catalog created
```

Note: To check the status of the `ibm-operator-catalog` pods, run this command and make sure `ibm-operator-catalog-xxxx` is running.

```
oc get pods -n openshift-marketplace
```

Output:

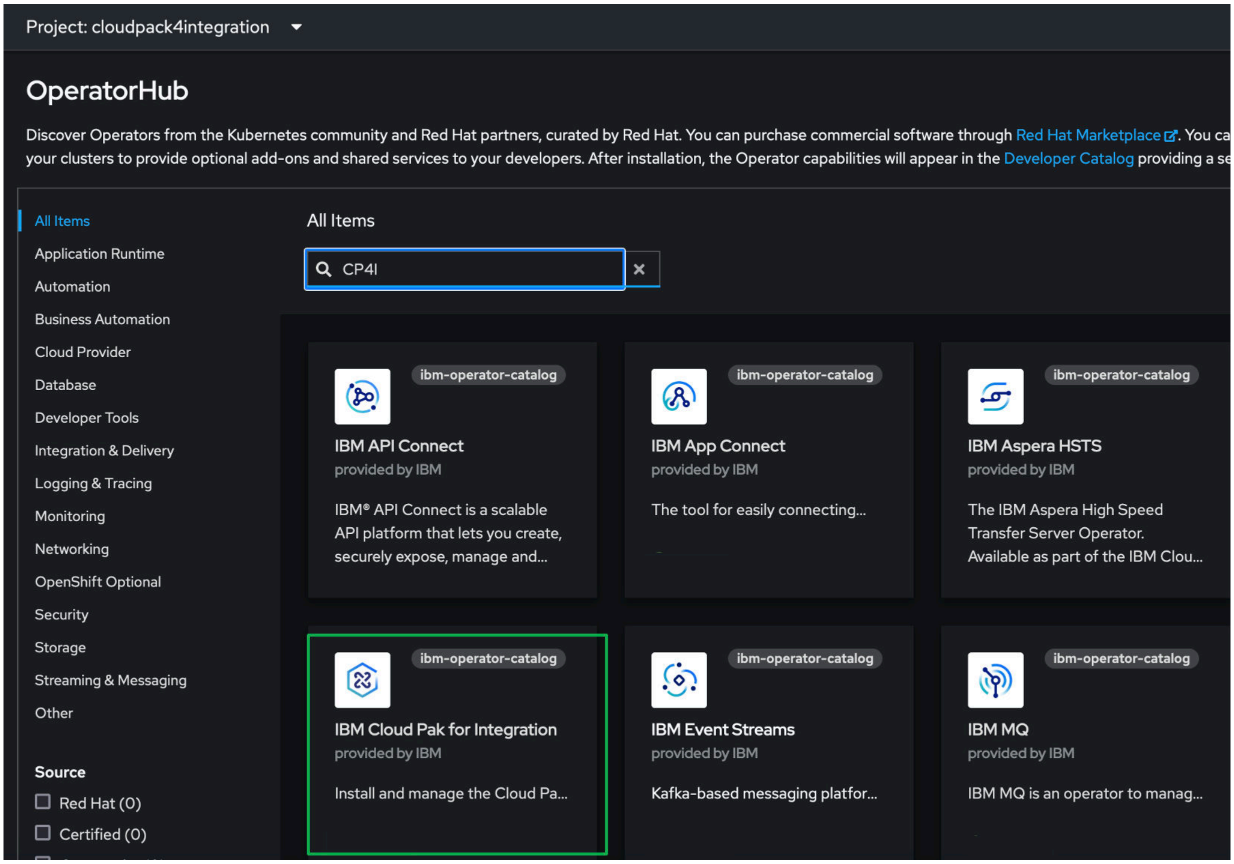
```
NAME                                                             READY   STATUS
RESTARTS   AGE
ibm-operator-catalog-xxxx                                        1/1     Running
0          61s
```

Note: For more information about adding Catalog Source, refer to Adding catalog sources to a cluster

Installing the Cloud Pak for Integration Operator

After the Catalog source has been successfully added, proceed with the following steps in the OpenShift web console.

1. Under **OperatorHub**, search for `CP4I` to find **IBM Cloud Pak for Integration**.



Project: cloudpack4integration

OperatorHub

Discover Operators from the Kubernetes community and Red Hat partners, curated by Red Hat. You can purchase commercial software through Red Hat Marketplace. You ca
your clusters to provide optional add-ons and shared services to your developers. After installation, the Operator capabilities will appear in the Developer Catalog providing a se

All Items

CP4I

ibm-operator-catalog
IBM API Connect
provided by IBM
IBM® API Connect is a scalable API platform that lets you create, securely expose, manage and...

ibm-operator-catalog
IBM App Connect
provided by IBM
The tool for easily connecting...

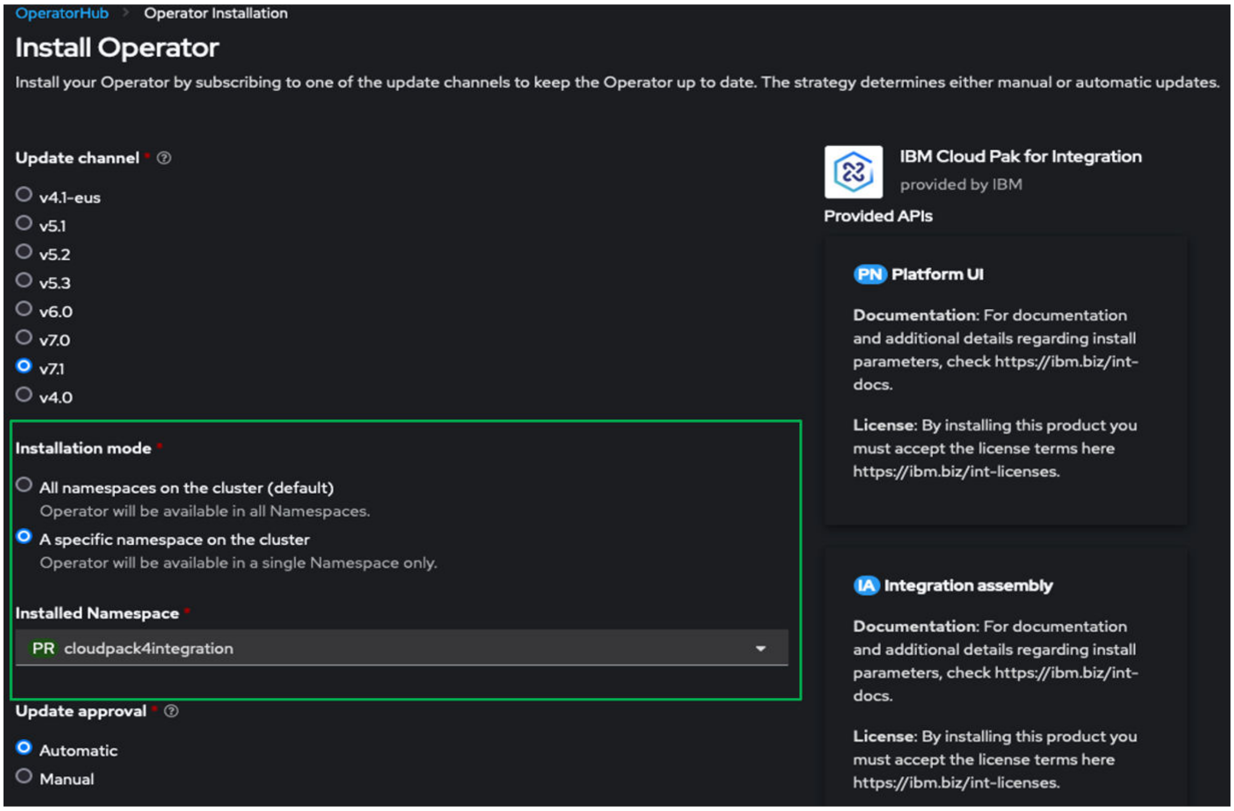
ibm-operator-catalog
IBM Aspera HSTS
provided by IBM
The IBM Aspera High Speed Transfer Server Operator. Available as part of the IBM Clou...

ibm-operator-catalog
IBM Cloud Pak for Integration
provided by IBM
Install and manage the Cloud Pa...

ibm-operator-catalog
IBM Event Streams
provided by IBM
Kafka-based messaging platfor...

ibm-operator-catalog
IBM MQ
provided by IBM
IBM MQ is an operator to manag...

2. Select **installation mode > A specific namespace on the cluster**. Under **Installed Namespace**, select `cloudpack4integration`, then proceed with the installation.



OperatorHub > Operator Installation

Install Operator

Install your Operator by subscribing to one of the update channels to keep the Operator up to date. The strategy determines either manual or automatic updates.

Update channel *
○ v4.1-eus
○ v5.1
○ v5.2
○ v5.3
○ v6.0
○ v7.0
● v7.1
○ v4.0

Installation mode *
○ All namespaces on the cluster (default)
Operator will be available in all Namespaces.
● A specific namespace on the cluster
Operator will be available in a single Namespace only.

Installed Namespace *
PR cloudpack4integration

Update approval *
● Automatic
○ Manual

IBM Cloud Pak for Integration
provided by IBM

Provided APIs

PN Platform UI

Documentation: For documentation and additional details regarding install parameters, check https://ibm.biz/int-docs.

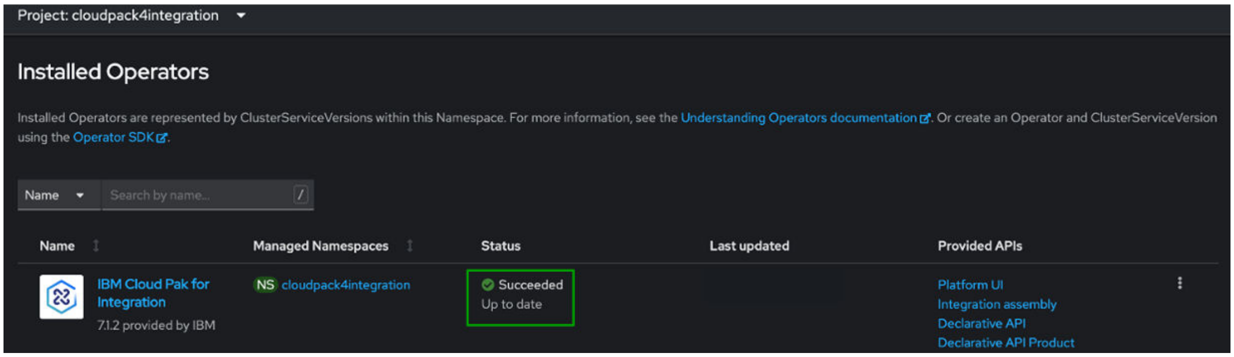
License: By installing this product you must accept the license terms here https://ibm.biz/int-licenses.

IA Integration assembly

Documentation: For documentation and additional details regarding install parameters, check https://ibm.biz/int-docs.

License: By installing this product you must accept the license terms here https://ibm.biz/int-licenses.

3. Verify the installation. `cloudpack4integration` has a status of `success` and is listed under the **Installed Operators**.



Project: cloudpack4integration

Installed Operators

Installed Operators are represented by ClusterServiceVersions within this Namespace. For more information, see the Understanding Operators documentation. Or create an Operator and ClusterServiceVersion using the Operator SDK.

Name	Managed Namespaces	Status	Last updated	Provided APIs
IBM Cloud Pak for Integration 7.1.2 provided by IBM	NS cloudpack4integration	Succeeded Up to date	[redacted]	Platform UI Integration assembly Declarative API Declarative API Product

Adding the entitlement key

The **IBM entitlement key** allows operators to pull software images automatically that are related to IBM Cloud Pak for Integration.

Follow this link to set up the entitlement key: Finding and applying your entitlement key by using the UI (online installation).

Installing the platform UI

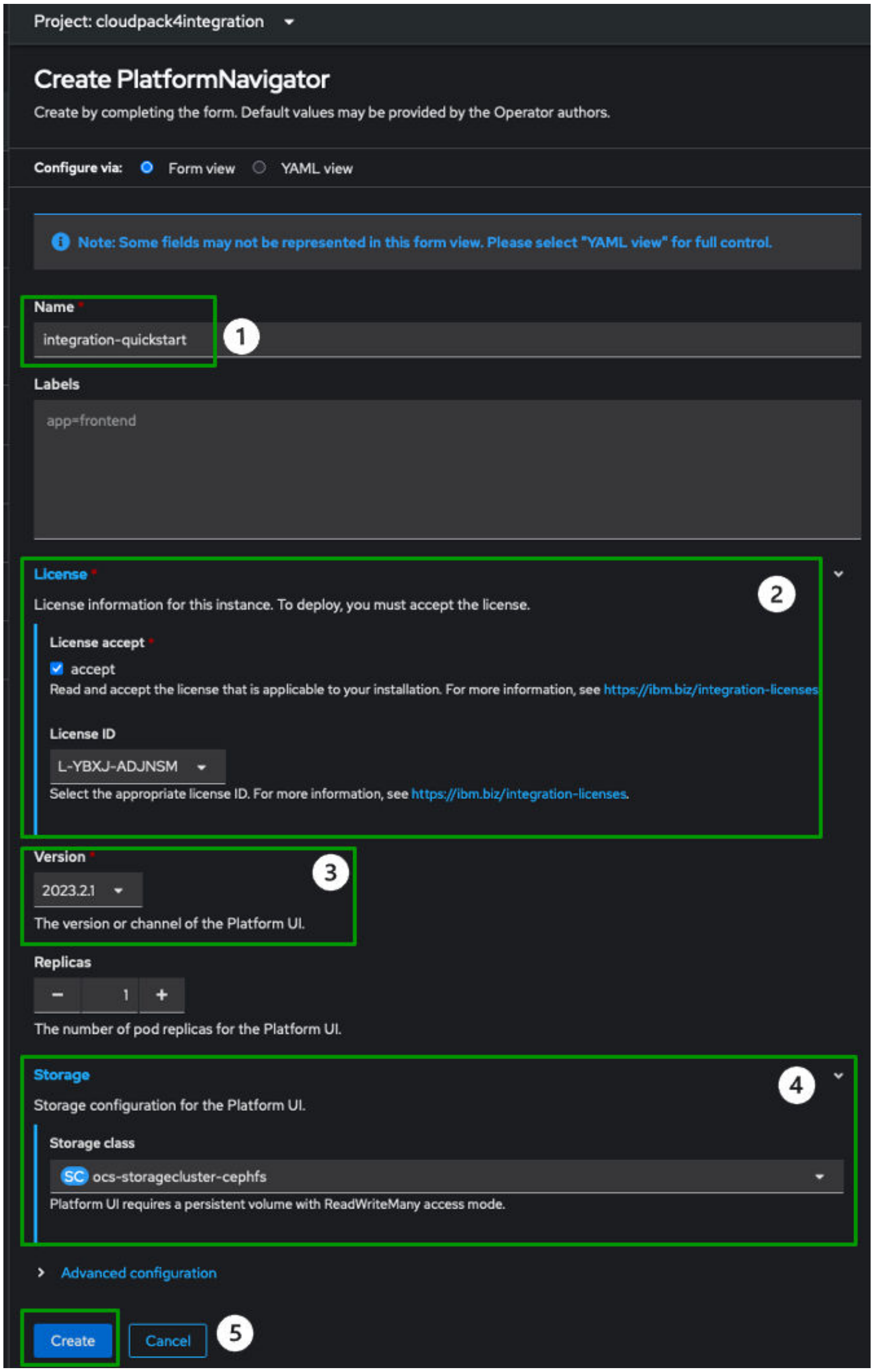
Note: As a prerequisite, make sure that the **entitlement key** is added to the namespace `cloudpack4integration`.

Note: Make sure you have switched into the project `cloudpack4integration`, from the drop-down menu in the upper left corner.

1. Select **Operators > Installed Operators > IBM Cloud Pak for Integration > Platform UI**.
2. Select **Create PlatformNavigator**. There are two methods to create the platform UI. One method uses **Form view** and the other method uses **Yaml view**. This setup uses **Form view**.

Form view

1. In the **Name:** field, enter the name `integration-quickstart`.
2. Click the arrow to expand the **Licence**. Check the licence checkbox. Select the default licence ID. For more information, refer to Licensing.
3. Select the **Version** or leave the default.
4. Expand **Storage** and select the storage class that supports ReadWriteMany(RWX). For more information, refer to Storage considerations.
5. Click **Create**.


Project: cloudpack4integration

Create PlatformNavigator

Create by completing the form. Default values may be provided by the Operator authors.

Configure via: ● Form view ○ YAML view

Note: Some fields may not be represented in this form view. Please select "YAML view" for full control.

Name *
integration-quickstart

Labels
app=frontend

License *
License information for this instance. To deploy, you must accept the license.

License accept *
☑ accept
Read and accept the license that is applicable to your installation. For more information, see https://ibm.biz/integration-licenses

License ID
L-YBXJ-ADJNSM
Select the appropriate license ID. For more information, see https://ibm.biz/integration-licenses.

Version *
2023.2.1
The version or channel of the Platform UI.

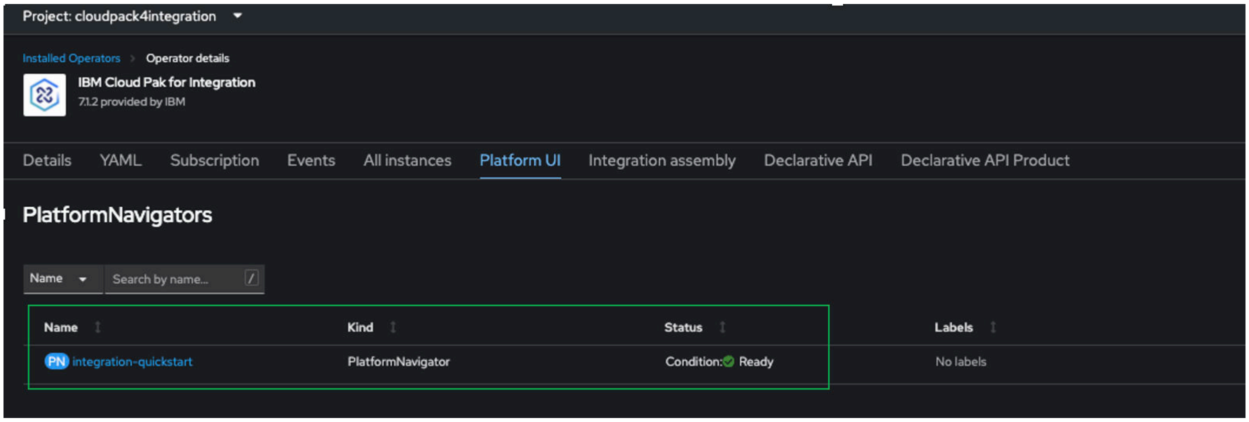
Replicas
1
The number of pod replicas for the Platform UI.

Storage
Storage configuration for the Platform UI.

Storage class
SC ocs-storagecluster-cephfs
Platform UI requires a persistent volume with ReadWriteMany access mode.

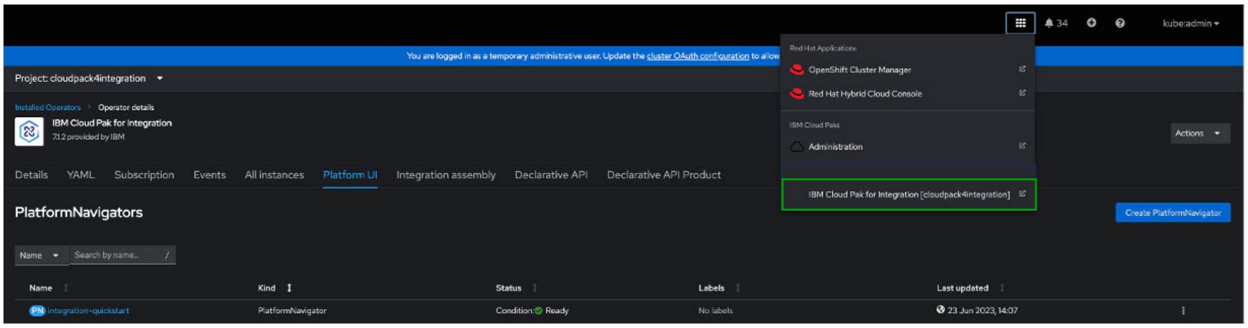
Verifying the setup

- Make sure the status of the **Platform UI** is in **Ready** state.



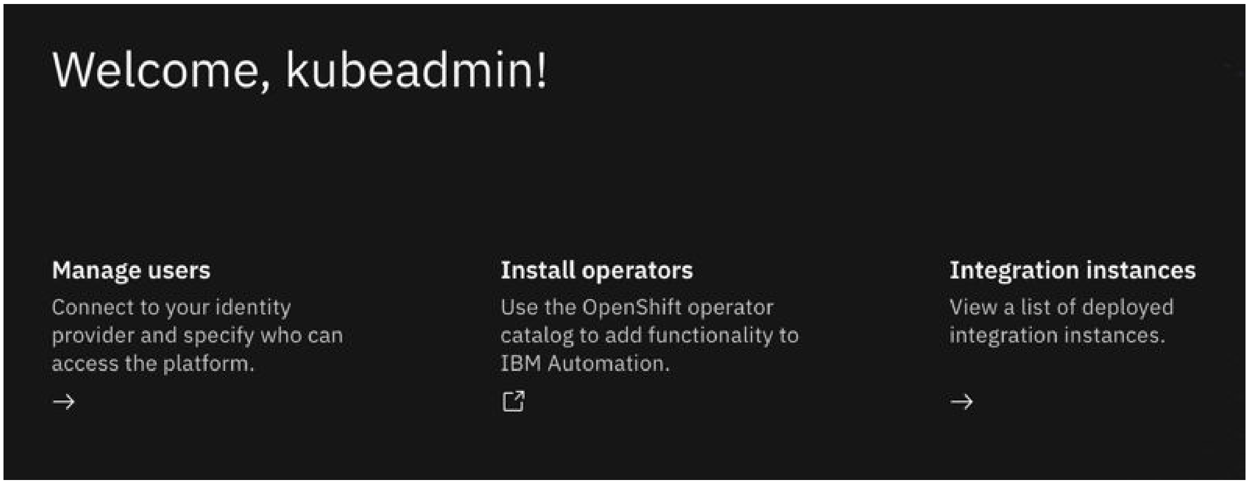
Name	Kind	Status	Labels
PN integration-quickstart	PlatformNavigator	Condition: Ready	No labels

- Select the **Cloud Pak for Integration UI** and login.


You are logged in as a temporary administrative user. Update the cluster OAuth configuration to allow

Name	Kind	Status	Labels	Last updated
PN integration-quickstart	PlatformNavigator	Condition: Ready	No labels	23 Jun 2023, 14:07

- Home Screen view:


Welcome, kubeadmin!

Manage users
Connect to your identity provider and specify who can access the platform.

Install operators
Use the OpenShift operator catalog to add functionality to IBM Automation.

Integration instances
View a list of deployed integration instances.

Chapter 4. MongoDB in Red Hat OpenShift Container Platform

This scenario uses an IBMz MongoDB EE Operator to test MongoDB in Red Hat OpenShift Container Platform (OpenShift).

To make MongoDB PoC run in OpenShift 4.11 and above, this Helm chart is modified: ibm-mongodb-enterprise-helm.

For OpenShift version below 4.11 (4.8, 4.9 and 4.10) an operator is available to try out MongoDB Enterprise Edition in OpenShift. See IBMz MongoDB EE Operator. As this operator was not available for 4.12 (at this time) a custom Helm Chart is used.

OpenShift prerequisites

An NFS server (NFSv4-only) is required to run these steps.

1. Provision storage for the MongoDB instance by creating a persistent volume. In this PoC use-case, NFS was used to make debugging easier. However, consider using OpenShift Data Foundation for reliability and performance in production environments. Make sure to adjust the value for the **NFS_SERVER_IP** variable in the following example:

```
NFS_SHARE_CAPACITY="20Gi"
NFS_SHARE_PATH="/nfs_share"
NFS_SERVER_IP="?.?.?.?"
STORAGE_CLASS_NAME="mongo-storage-class"

cat <(echo '{
   "apiVersion": "v1",
   "kind": "PersistentVolume",
   "metadata": {
     "name": "nfs4mongo-pv-0001",
     "annotations": {
       "pv.kubernetes.io/bound-by-controller": "yes"
     },
     "finalizers": [ "kubernetes.io/pv-protection" ]
   },
   "spec": {
     "capacity": {
       "storage": "${NFS_SHARE_CAPACITY}"
     },
     "accessModes": [ "ReadWriteMany" ],
     "nfs": {
       "path": "${NFS_SHARE_PATH}",
       "server": "${NFS_SERVER_IP}"
     },
     "persistentVolumeReclaimPolicy": "Delete",
     "storageClassName": "${STORAGE_CLASS_NAME}"
   }
}') > nfs-persistent-volume.json

oc create -f nfs-persistent-volume.json
```

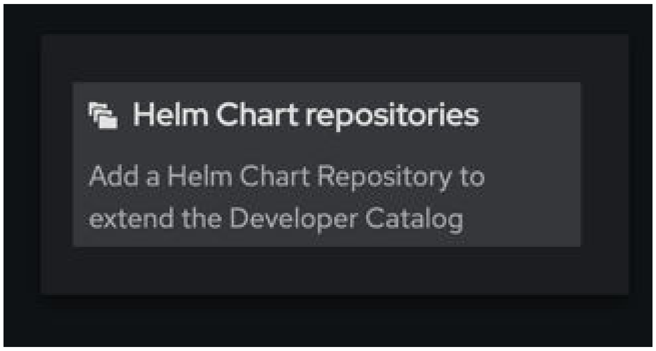
2. On the NFS server-side the user-id and group-id in the folder must match the user-id that the container runtime of mongoDB uses later (check the `mongo-db-helm-with-replset.yaml` and search for `runAsUser`). Run this command on the NFS server:

```
chown 1000740005:1000740005 /mongo_share
```

3. In the Developer View of OpenShift, add a new project. In this example, the project is called `mongodb-test-1`. Switch to the new project.

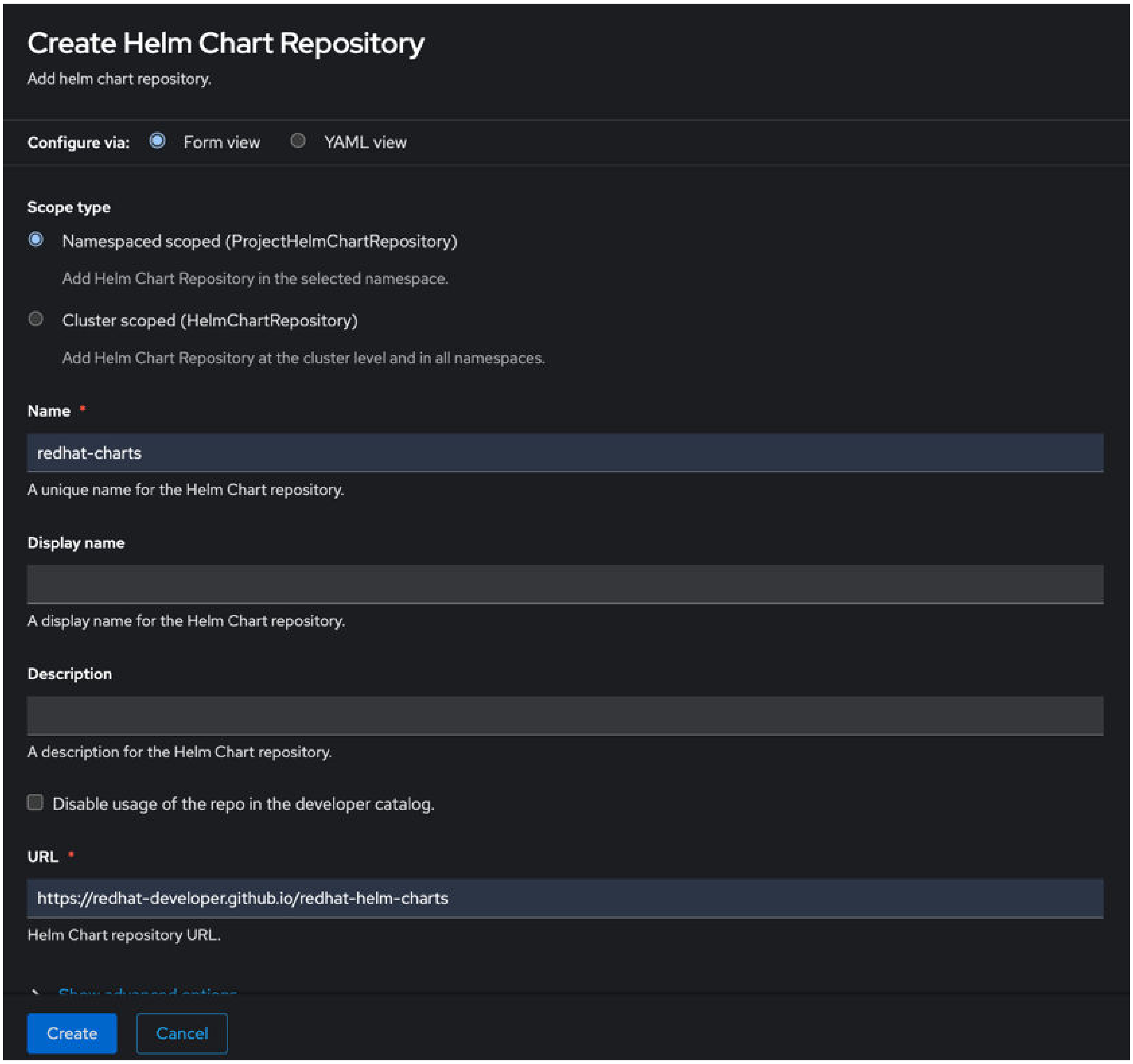
   Note: This project name/namespace is used in later steps. If you use a different project name, remember to change the name in the subsequent commands.

4. To display the Helm chart in the **Developer Catalog**, add the Red Hat Helm chart as a Helm chart repository by clicking **Helm Chart repositories > Create Helm Chart Repository** to . Set the **URL**, to point to `https://redhat-developer.github.io/redhat-helm-charts`.



Helm Chart repositories

Add a Helm Chart Repository to extend the Developer Catalog



Create Helm Chart Repository

Add helm chart repository.

Configure via: ● Form view ○ YAML view

Scope type

● Namespaced scoped (ProjectHelmChartRepository)

Add Helm Chart Repository in the selected namespace.

○ Cluster scoped (HelmChartRepository)

Add Helm Chart Repository at the cluster level and in all namespaces.

Name *

redhat-charts

A unique name for the Helm Chart repository.

Display name

A display name for the Helm Chart repository.

Description

A description for the Helm Chart repository.

□ Disable usage of the repo in the developer catalog.

URL *

https://redhat-developer.github.io/redhat-helm-charts

Helm Chart repository URL.

5. In **Developer Catalog**, click **+Add** and **Helm Charts**. Search for, then click **IBM Mongodb Enterprise Helm > Install Helm Chart**. Before you continue, edit the yaml file and replace the entire content with this yaml file:

```
affinity: {}
autoscaling:
  enabled: false
  maxReplicas: 100
  minReplicas: 1
  targetCPUUtilizationPercentage: 80
database:
  adminpassword: admin123
  adminuser: adminuser
  name_database: testdb
fullnameOverride: ''
global:
  license: true
  persistence:
    claims:
      accessMode: ReadWriteMany
      capacity: 10
      capacityUnit: Gi
      mountPath: /data/db/
      name: mongodb-pvc-0001
      storageClassName: mongo-storage-class
    securityContext:
      fsGroup: 0
      supplementalGroup: 0
image:
  pullPolicy: Always
  repository: quay.io/ibm/ibmz-mongodb-enterprise-database
  tag: "v4.4-rh7-s390x"
imagePullSecrets: []
ingress:
  annotations: {}
  controller: nginx
  host: mongotest.apps.ocp0.sa.boe
  tls: []
nameOverride: ''
nodeSelector: {}
podAnnotations: {}
podSecurityContext: {}
replicaCount: 1
resources: {}
securityContext:
  runAsNonRoot: true
  # Must be in a certain range:
  runAsUser: 1000740005
service:
  port: 27017
  type: ClusterIP
serviceAccount:
  annotations: {}
  create: true
  name: mongod
tolerations: []
```

6. Verify your installation.

   a. Go to the terminal of the spawned Pod, then run this command:

   ```
   mongo -u myUserAdmin -p password --authenticationDatabase admin
   ```

Setting up MongoDB as part of a replica set

The installation of MongoDB spawns a single instance of `mongod` and is not in a replica set. The MongoDB connector that is used later in the Kafka setup expects a replica set and does not work without it.

In this section, the public image is slightly modified to make the `mongod` instance part of a replica set. This replica set has just a single instance which thinks that the other instances are currently down and promotes itself to be the primary.

Related resources:

- Convert a Standalone to a Replica Set
- "The $changeStream stage is only supported on replica sets" error while using mongodb-source-connect

This procedure edits the init script that sets up `mongodb` to change the behaviour. You can also modify `/etc/mongod.conf` instead.

The following commands require that podman is installed on the bastion.

1. Download and run the mongodb docker image with an interactive shell:

```
podman run -it quay.io/ibm/ibmz-mongodb-enterprise-database:v4.4-rh7-s390x bash
```

2. Copy the `mongo_init.sh` to the local file system:

```
podman cp CONTAINER:/var/log/mongodb/mongo_init.sh ./mongo_init.sh
```

3. Edit the `mongo_init.sh` script:

   a. Change the line that starts the `mongod` daemon to: `mongod --replSet "rs0" -bind_ip_all`.

   Note: `--auth` was removed because it also caused issues.

4. Make the `mongo_init.sh` script executable and readable by anyone:

```
chmod 777 mongo_init.sh
```

5. Create a Dockerfile that uses the original image and replaces the `mongo_init.sh` script:

```
FROM quay.io/ibm/ibmz-mongodb-enterprise-database:v4.4-rh7-s390x

COPY mongo_init.sh /var/log/mongodb/mongo_init.sh
```

6. Build and tag the docker image:

```
# cd into the dir with the Dockerfile
podman build . -t custom-mongodb-1:latest
```

7. Push the image to the internal OpenShift image registry. When the `oc get route` command is not working check out: How to expose the registry.

```
# Get the route to the internal OpenShift image registry
HOST=$(oc get route default-route -n openshift-image-registry --template='{{ .spec.host }}')

# Login to the internal docker image registry
podman login -u admin -p $(oc whoami -t) ${HOST}

NAMESPACE="mongodb-test-1"
IMAGE_NAME="custom-mongodb-1:latest"
LOCAL_IMAGE_NAME="localhost/${IMAGE_NAME}"
REMOTE_IMAGE_NAME="${IMAGE_NAME}"
podman push ${LOCAL_IMAGE_NAME} ${HOST}/${NAMESPACE}/${REMOTE_IMAGE_NAME}
```

8. Give the `mongod` service account permission to pull the image from the internal image registry `mongodb-test-1` namespace:

```
oc policy add-role-to-user \
  system:image-puller system:serviceaccount:mongodb-test-1:mongod \
  --namespace=mongodb-test-1
```

9. Deploy the Helm chart that now uses the custom image you pushed to the internal image registry:

```
affinity: {}
autoscaling:
  enabled: false
  maxReplicas: 100
  minReplicas: 1
  targetCPUUtilizationPercentage: 80
database:
  adminpassword: admin123
  adminuser: adminuser
  name_database: testdb
fullnameOverride: ''
global:
  license: true
  persistence:
    claims:
      accessMode: ReadWriteMany
      capacity: 10
      capacityUnit: Gi
      mountPath: /data/db/
      name: mongodb-pvc-0001
      storageClassName: mongo-storage-class
    securityContext:
      fsGroup: 0
      supplementalGroup: 0
image:
  pullPolicy: Always
  # CUSTOM IMAGE (WITH REPLICA SET ENABLED):
  repository: default-route-openshift-image-registry.apps.ocp0.sa.boe/mongodb-test-1/custom-
mongodb-1
  tag: "latest"
imagePullSecrets: []
ingress:
  annotations: {}
  controller: nginx
  host: mongotest.apps.ocp0.sa.boe
  tls: []
nameOverride: ''
nodeSelector: {}
podAnnotations: {}
podSecurityContext: {}
replicaCount: 1
resources: {}
securityContext:
  runAsNonRoot: true
  # Bust be in a certain range:
  runAsUser: 1000740005
service:
  port: 27017
  type: ClusterIP
serviceAccount:
  annotations: {}
  create: true
  name: mongod
tolerations: []
```

Note: If you get an image pull error, delete the pod. OpenShift automatically creates a new pod and tries the pull process again.

10. Open a terminal for the mongodb pod. Log in and then initiate the replica set:

```
mongo -u myUserAdmin -p password --authenticationDatabase admin
rs.initiate()
```

Note: Make sure you wait until the mongodb transitioned from secondary to primary.

Setting up MongoDB on an LPAR

Follow the official documentation to Install MongoDB Enterprise Edition on Red Hat or CentOS. You can try it with a higher version than 4.4 but as this use-case uses version 4.4 on the OpenShift-side, version 4.4 is used on the LPAR-side for consistency.

Chapter 5. Setting up IBM Event Streams

Learn how to set up the solution including IBM Event Streams, Kafka Cluster, Kafka Connect, and Connectors.

There are different options available to create a Kafka environment with IBM Event Streams:

- Use the Operator details view of the IBM Event Streams operator create the required resources
- Create a new `Integration` instance of type `Kafka cluster` via the IBM Cloud Pak for Integration UI / Platform UI

After creating the `Kafka cluster/EventStreams` resource you can either:

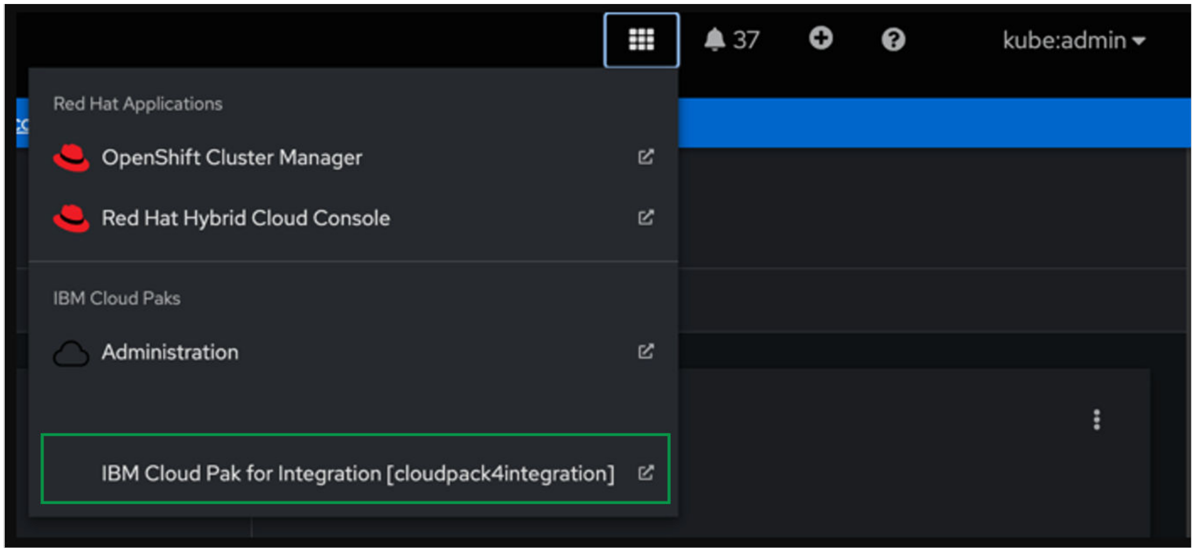
- Manage resources in the **IBM Event Streams** Operator overview
- Use the **IBM Event Stream** UI

In the following procedures both user interfaces are used to create and manage the Kafka cluster.

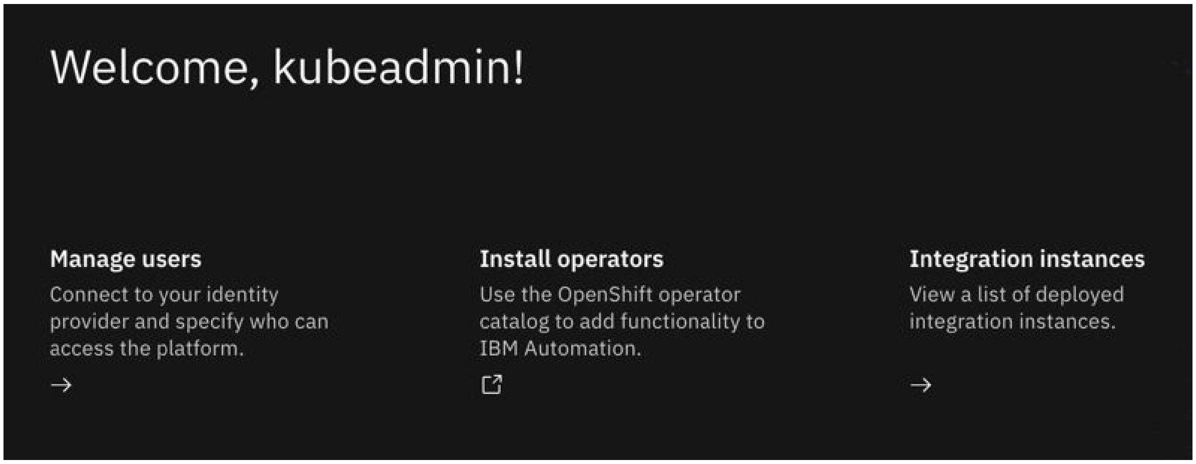
Setting up Kafka Connect

1. Create a Kafka cluster instance (also known as an Event Stream resource).

   a. Go to the **Cloud Pak for Integration UI**.


37
kube:admin
Red Hat Applications
OpenShift Cluster Manager
Red Hat Hybrid Cloud Console
IBM Cloud Paks
Administration
IBM Cloud Pak for Integration [cloudpack4integration]

   Click **Integration instances**:



Welcome, kubeadmin!

Manage users
Connect to your identity provider and specify who can access the platform.
→

Install operators
Use the OpenShift operator catalog to add functionality to IBM Automation.

Integration instances
View a list of deployed integration instances.
→

Click **Create an instance**, select **Event-Streams**, click **Next**, select **Development**, click **Next**, accept the license, click **Create**.

Note: Do not select the option: `minimal without security`, as this will lead to connection issues when following this guide.

b. ALTERNATIVE workflow:

i) As Administrator, go to **Operators > Installed Operators** and find `IBM Event Streams`. Some blue links are displayed in the **Provided APIs** column. Click the Event Stream link.

ii) Click **Create EventStreams**.

iii) Switch from **Form view** to **YAML view** and select the **Samples** tab in the right panel.

iv) Click **Development Sample > Try it**.

Note: The options presented under samples are similar to the options you get when you create the EventStreams resource via the Cloud Pak for Integration UI.

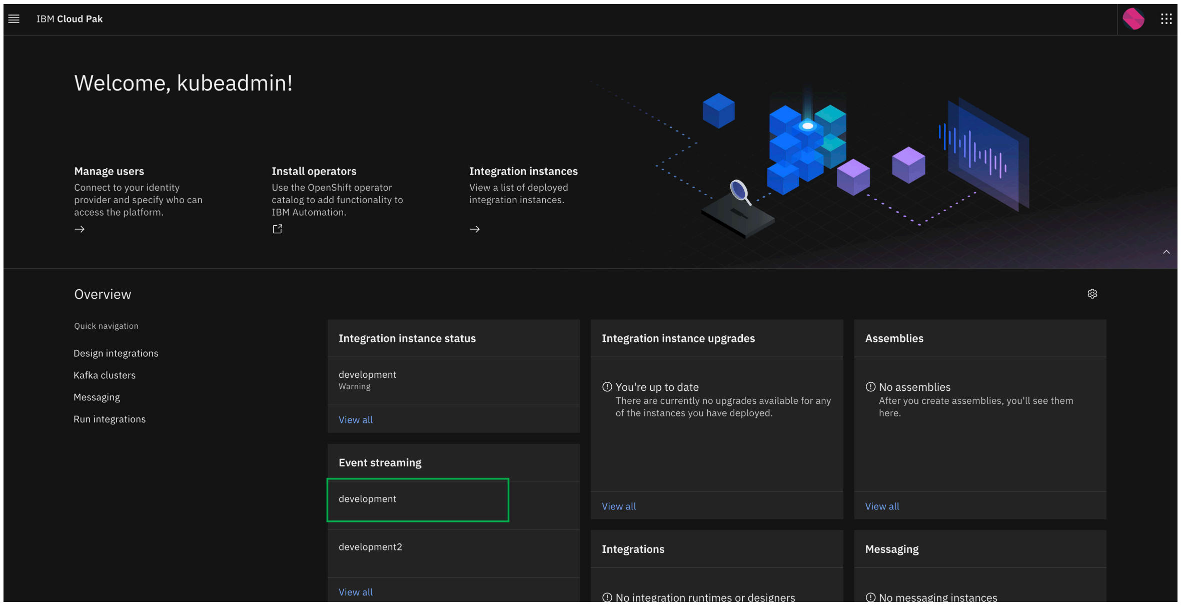
v) Press **Create**.

2. Create a KafkaConnect environment.

The KafkaConnect needs to be able to connect to MongoDB, so it must include the libraries for it to do so. To make this work a custom docker image is built that includes the required `.jar` files for the `MongoDbSourceConnector` and `MongoDbSinkConnector`.

The steps in this procedure require that you already have an internal image registry setup within OpenShift and that you can push/pull to it from the Bastion node.

a. The Dockerfile for the `Kafka Connect environment` can be downloaded from the IBM Event Streams UI Toolbox. To get there, open the IBM Event Streams UI:



Welcome, kubeadmin!

Manage users
Connect to your identity provider and specify who can access the platform.

Install operators
Use the OpenShift operator catalog to add functionality to IBM Automation.

Integration instances
View a list of deployed integration instances.

Overview

Integration instance status
development
Warning
View all

Integration instance upgrades
ⓘ You're up to date
There are currently no upgrades available for any of the instances you have deployed.
View all

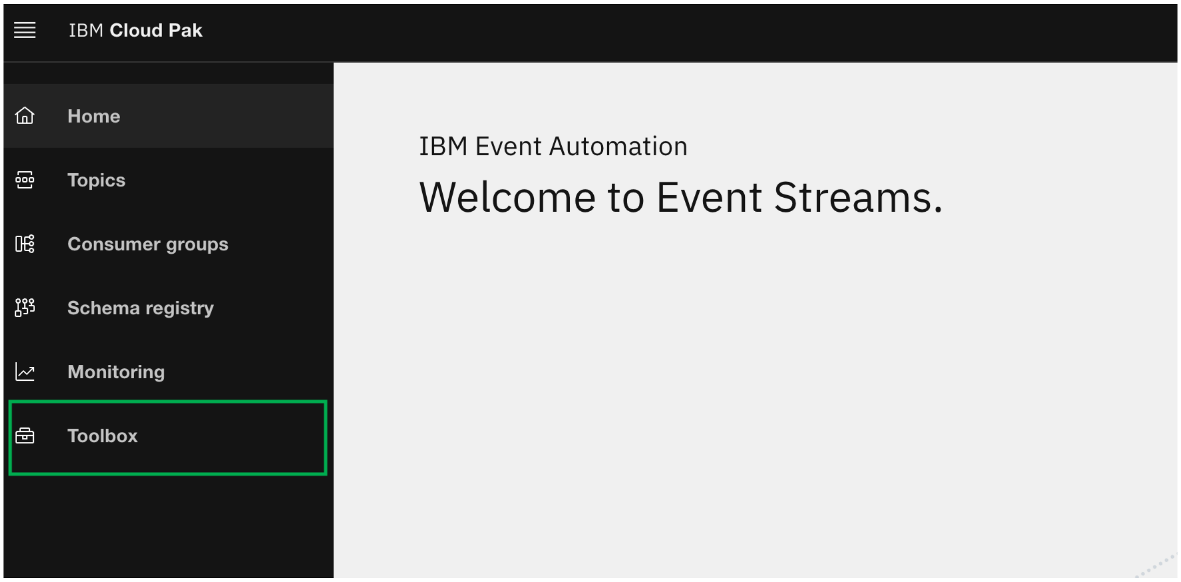
Assemblies
ⓘ No assemblies
After you create assemblies, you'll see them here.
View all

Event streaming
development
development2
View all

Integrations
ⓘ No integration runtimes or designers

Messaging
ⓘ No messaging instances

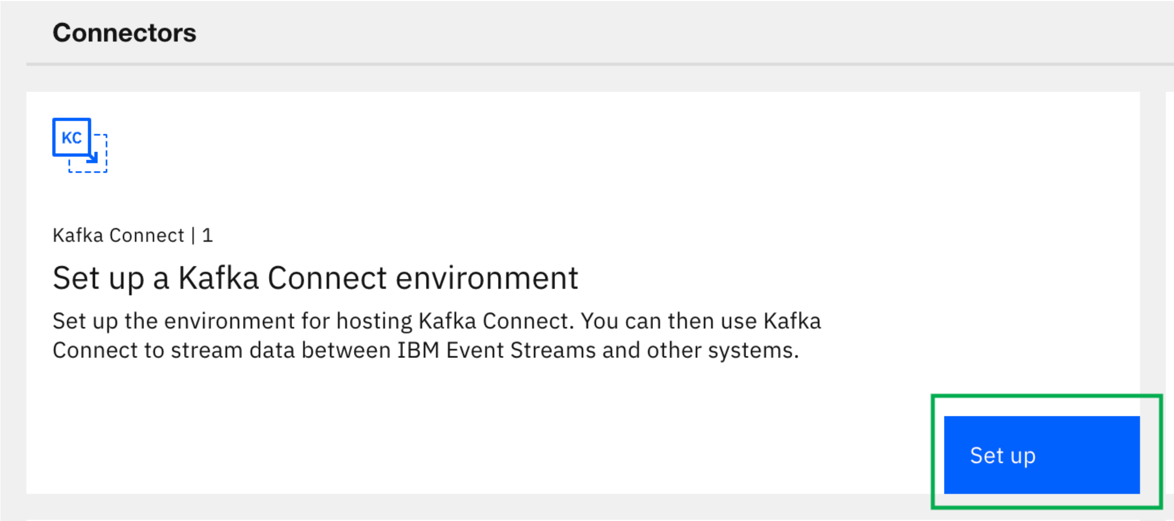
Go to Toolbox:



IBM Event Automation

Welcome to Event Streams.

Select Set up a Kafka Connect environment and proceed with the Set up:



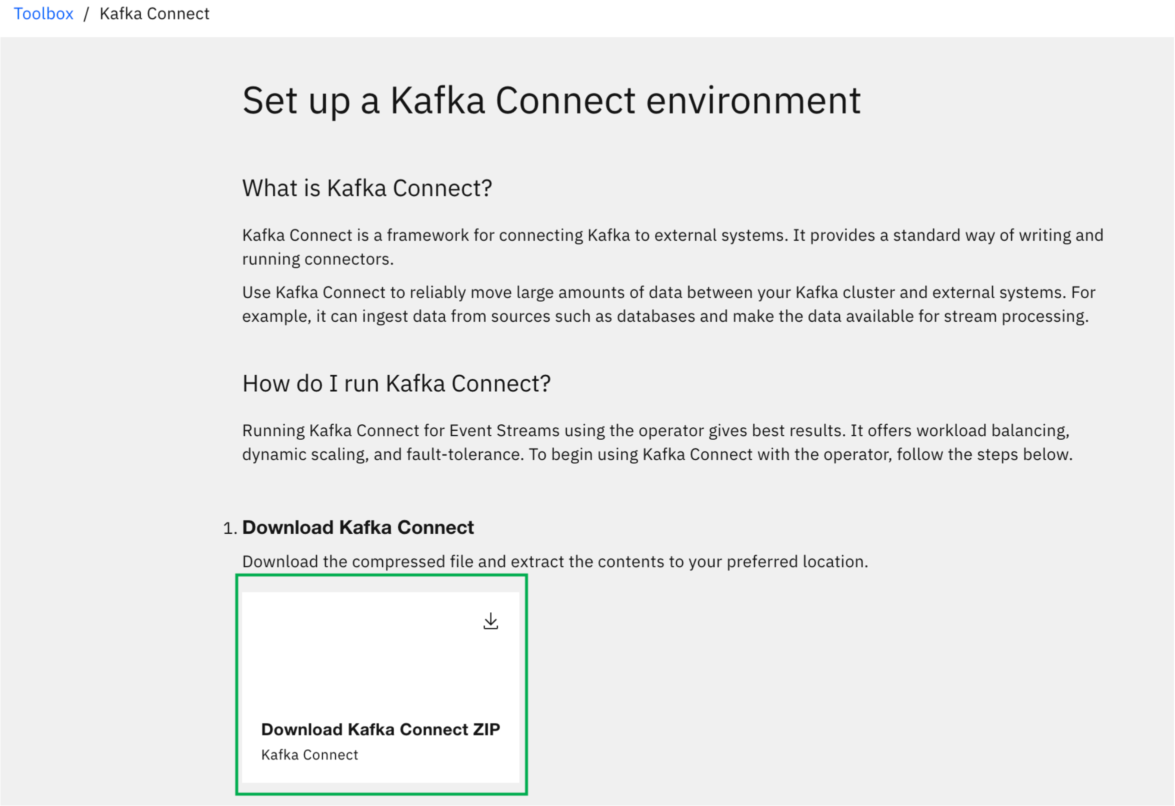
Connectors

Kafka Connect | 1

Set up a Kafka Connect environment

Set up the environment for hosting Kafka Connect. You can then use Kafka Connect to stream data between IBM Event Streams and other systems.

Download the Kafka Connect ZIP file, move it to the bastion and **unzip it to a folder called kafkaconnect**:



Toolbox / Kafka Connect

Set up a Kafka Connect environment

What is Kafka Connect?

Kafka Connect is a framework for connecting Kafka to external systems. It provides a standard way of writing and running connectors.

Use Kafka Connect to reliably move large amounts of data between your Kafka cluster and external systems. For example, it can ingest data from sources such as databases and make the data available for stream processing.

How do I run Kafka Connect?

Running Kafka Connect for Event Streams using the operator gives best results. It offers workload balancing, dynamic scaling, and fault-tolerance. To begin using Kafka Connect with the operator, follow the steps below.

1. **Download Kafka Connect**

Download the compressed file and extract the contents to your preferred location.

Download Kafka Connect ZIP

Kafka Connect

```
unzip kafkaconnect.zip -d kafkaconnect
```

b. Download the `mongo-kafka-connect-1.10.1-all.jar` file from org/mongodb/kafka/mongo-kafka-connect/1.10.1 to the bastion.

```
curl https://repo1.maven.org/maven2/org/mongodb/kafka/mongo-kafka-connect/1.10.1/mongo-
kafka-connect-1.10.1-all.jar --output mongo-kafka-connect-1.10.1-all.jar
```

c. Get the `mongo-kafka-connect-1.10.1-all.jar` file from org/mongodb/kafka/mongo-kafka-connect/1.10.1

Note: When a newer version is available you might want to try it.

Note: Check out the instructions about which jar file to download in Connector catalog. When we tried the jar file from here, KafkaConnect did not find the required library methods.

d. Copy the `mongo-kafka-connect-1.10.1-all.jar` file to the `my-plugins` folder of the unpacked zip archive:

```
cp mongo-kafka-connect-1.10.1-all.jar kafkaconnect/my-plugins/
```

e. As the Dockerfile pulls an image from `cp.icr.io` you need to make sure that you have access to this public container registry. You should be able to log in into the container registry with the entitlement key used during the Cloud Pak for Integration setup (see IBM Cloud Pak for Data instructions). Make sure that you are able to log in to the internal image registry of your OpenShift deployment.

i) Build and tag the image:

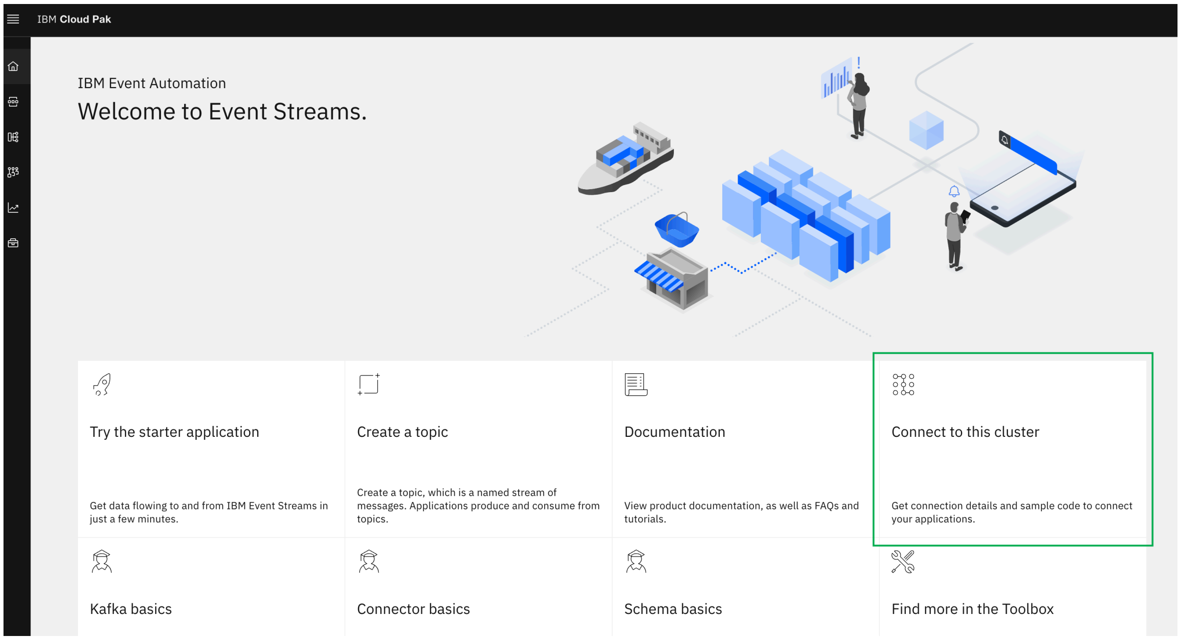
```
# authenticate
podman login cp.icr.io -u cp -p "${ENTITLENMENTKEY}"
```

```
# cd into the directory with the Dockerfile
podman build . -t my-connect-cluster-image-1
```

ii) Push the new image to the internal image registry. When the `oc get route` command is not working check out: How to expose the registry:

```
# Get the route to the internal OpenShift image registry
HOST=$(oc get route default-route -n openshift-image-registry --
template='{{ .spec.host }}')

# Login to the internal docker image registry
podman login -u admin -p $(oc whoami -t) ${HOST}

# Push the image
NAMESPACE="openshift"
IMAGE_NAME="my-connect-cluster-image-1:latest"
LOCAL_IMAGE_NAME="localhost/${IMAGE_NAME}"
REMOTE_IMAGE_NAME="${IMAGE_NAME}"
podman push ${LOCAL_IMAGE_NAME} ${HOST}/${NAMESPACE}/${REMOTE_IMAGE_NAME}
```

f. Generate credentials so that the KafkaConnect environment can connect to the Kafka cluster. In the `IBM Event Stream` UI: Click the tile `Connect to this cluster`.



IBM Event Automation

Welcome to Event Streams.

Try the starter application

Get data flowing to and from IBM Event Streams in just a few minutes.

Create a topic

Create a topic, which is a named stream of messages. Applications produce and consume from topics.

Documentation

View product documentation, as well as FAQs and tutorials.

Connect to this cluster

Get connection details and sample code to connect your applications.

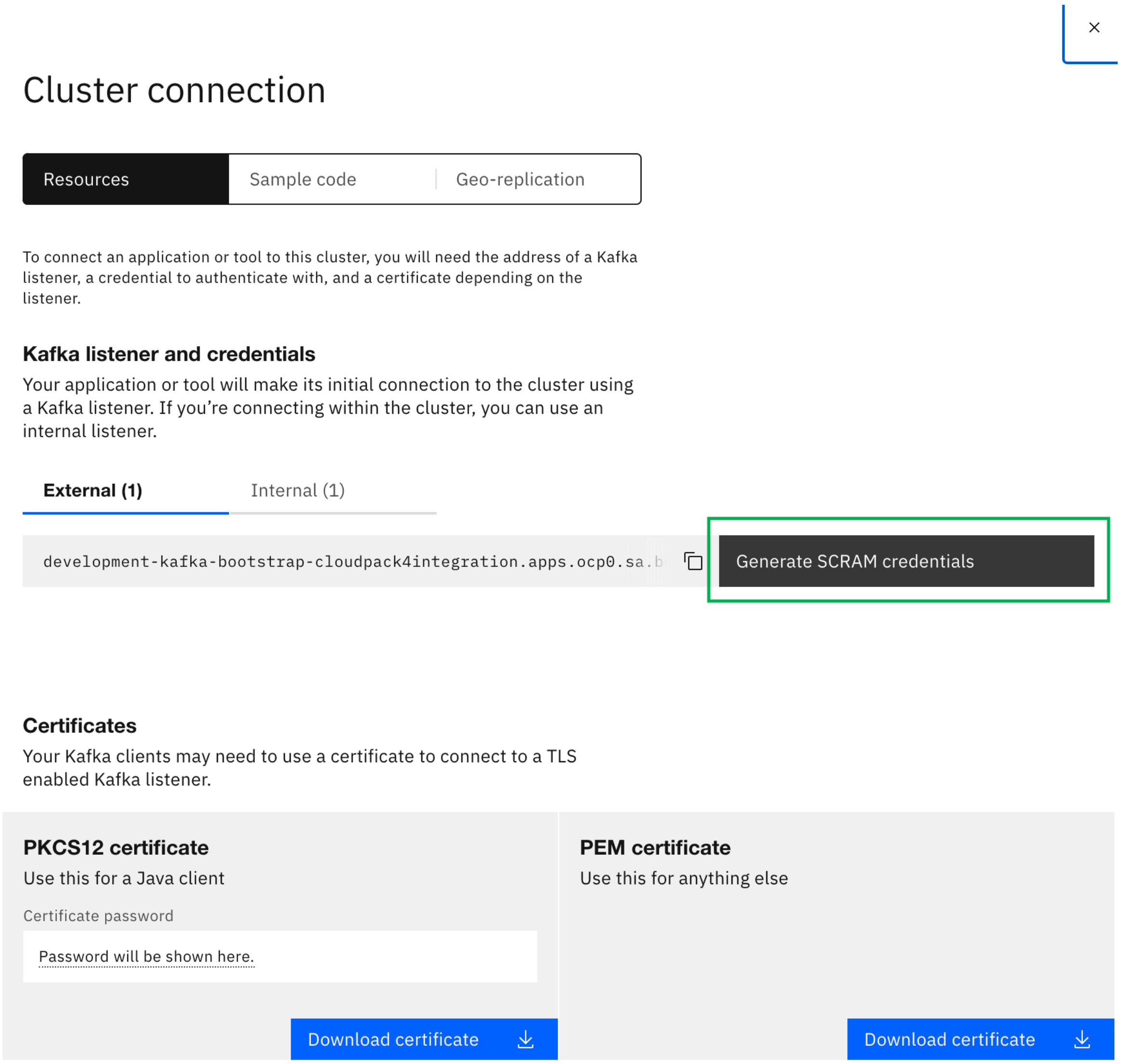
Kafka basics

Connector basics

Schema basics

Find more in the Toolbox

g. Then click `Generate SCRAM credentials`.



Cluster connection

To connect an application or tool to this cluster, you will need the address of a Kafka listener, a credential to authenticate with, and a certificate depending on the listener.

Kafka listener and credentials

Your application or tool will make its initial connection to the cluster using a Kafka listener. If you're connecting within the cluster, you can use an internal listener.

```
development-kafka-bootstrap-cloudpack4integration.apps.ocp0.sa.b
```

Certificates

Your Kafka clients may need to use a certificate to connect to a TLS enabled Kafka listener.

PKCS12 certificate

Use this for a Java client

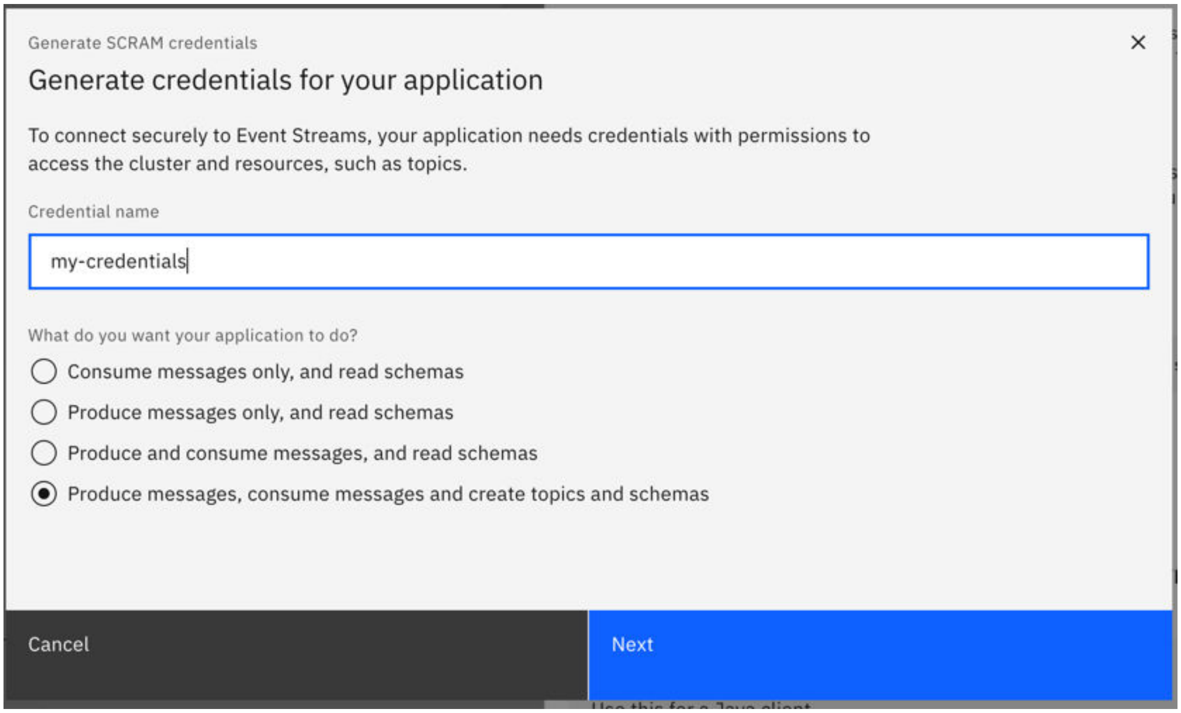
Certificate password

Password will be shown here.

PEM certificate

Use this for anything else

Select the most liberal options. The credential name used in the following is: `my-credentials`.



Generate SCRAM credentials

Generate credentials for your application

To connect securely to Event Streams, your application needs credentials with permissions to access the cluster and resources, such as topics.

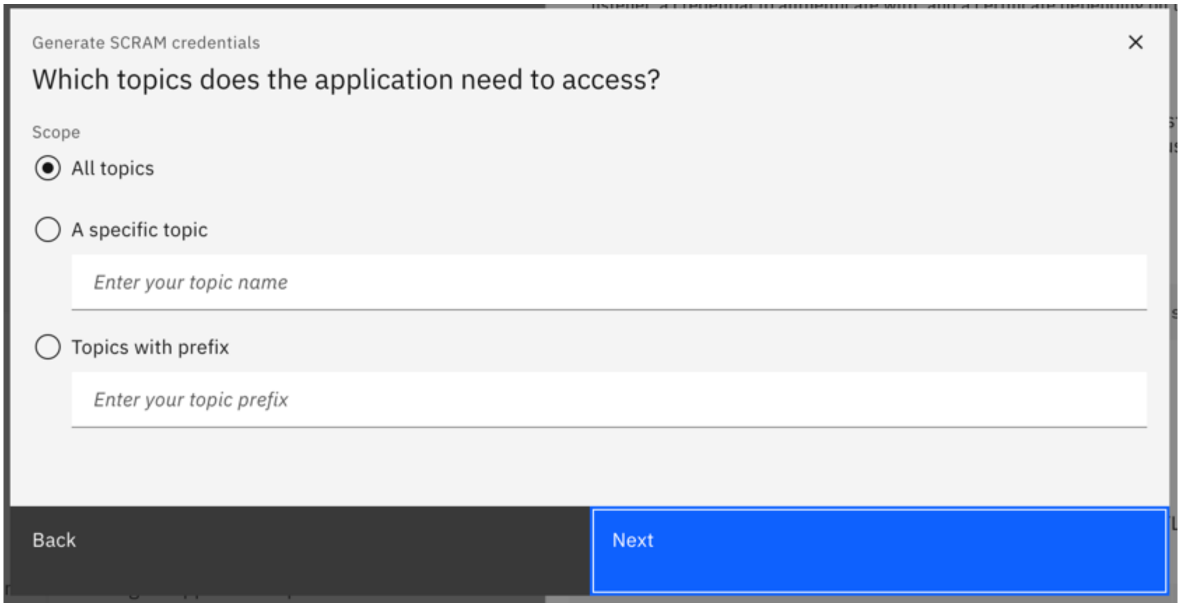
Credential name

my-credentials

What do you want your application to do?

○ Consume messages only, and read schemas
○ Produce messages only, and read schemas
○ Produce and consume messages, and read schemas
● Produce messages, consume messages and create topics and schemas

Select `All topics` and click next for the access.



Generate SCRAM credentials

Which topics does the application need to access?

Scope

● All topics

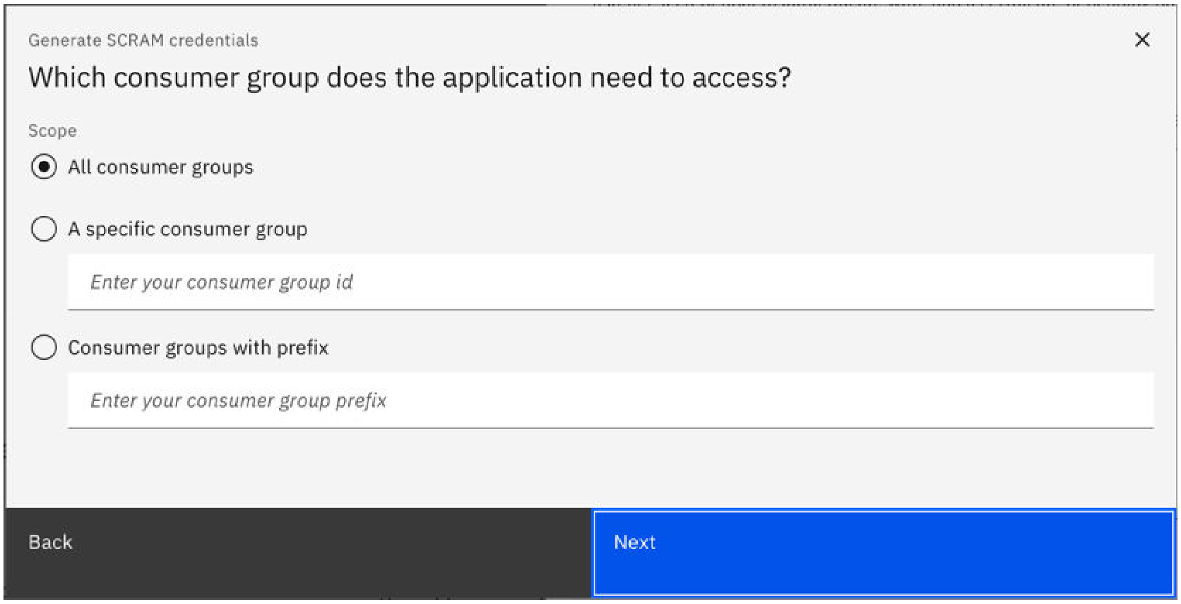
○ A specific topic

Enter your topic name

○ Topics with prefix

Enter your topic prefix

Select `All consumer group` and click next.


Generate SCRAM credentials

Which consumer group does the application need to access?

Scope

● All consumer groups

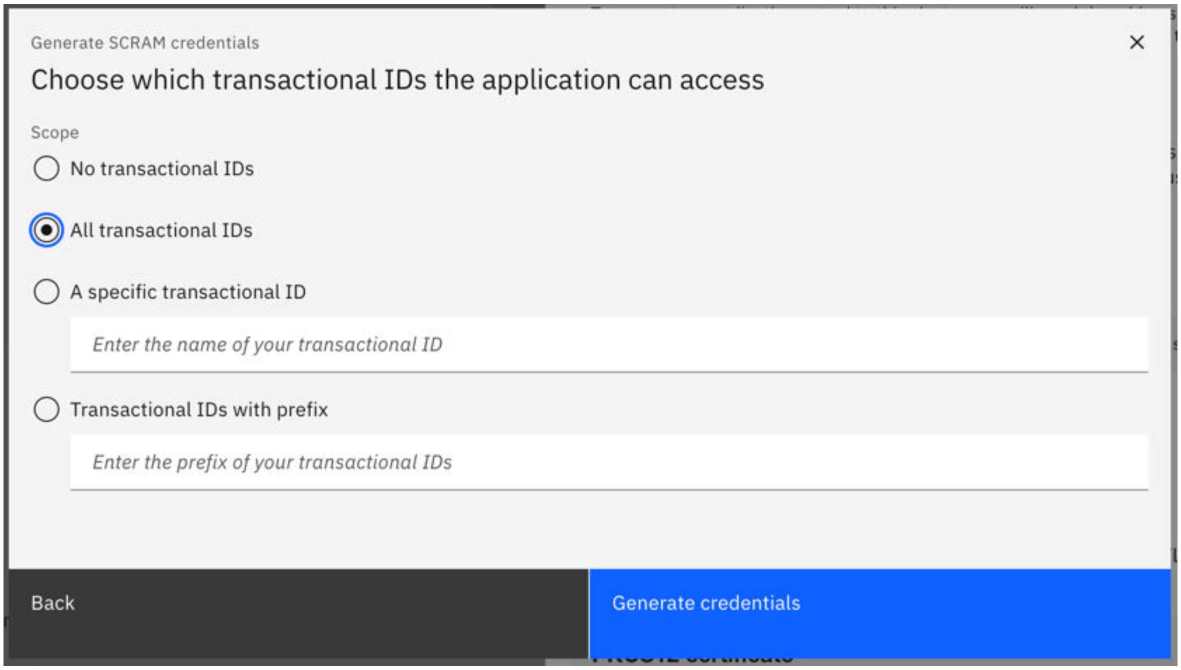
○ A specific consumer group

Enter your consumer group id

○ Consumer groups with prefix

Enter your consumer group prefix

Select `All transactional IDs` and click next.


Generate SCRAM credentials

Choose which transactional IDs the application can access

Scope

○ No transactional IDs

● All transactional IDs

○ A specific transactional ID

Enter the name of your transactional ID

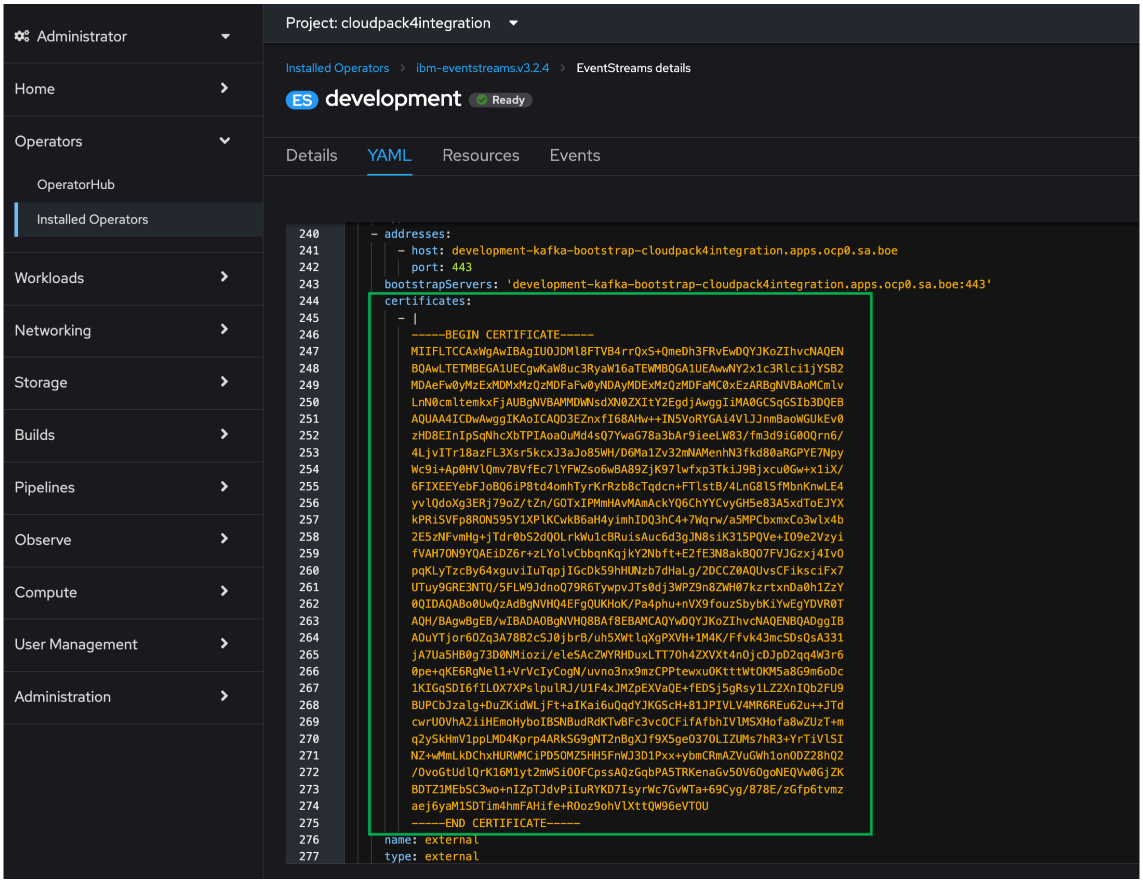
○ Transactional IDs with prefix

Enter the prefix of your transactional IDs

Note: This automatically creates a Kafka user.

h. The connection to the Kafka cluster is secured by TLS, so you need to trust the CA certificate used by the `Kafka cluster/Event Stream cluster`.

i) To find the certificate, go to **Installed Operators**, find **IBM Event Streams**, then click **Event Streams**. Click the Event Stream resource and go to the YAML view. The certificate is under **kafkaListeners**. Copy the certificate to a local plain-text file.



```
- addresses:
    - host: development-kafka-bootstrap-cloudpack4integration.apps.ocp0.sa.boe
      port: 443
  bootstrapServers: 'development-kafka-bootstrap-cloudpack4integration.apps.ocp0.sa.boe:443'
  certificates:
    - |
      -----BEGIN CERTIFICATE-----
      MIIFLTCCAxWgAwIBAgIUOJDMl8FTVB4rrQxS+QmeDh3FRvEwDQYJKoZIhvcNAQEN
      BQAwLTETMBEGA1UECgwKaW8uc3RyaW16aTEWMBQGA1UEAwwNY2x1c3Rlci1jYSB2
      MDAeFw0yMzExMDMxMzQzMDFaFw0yNDAyMDExMzQzMDFaMC0xEzARBgNVBAoMCmlv
      LnN0cmltemkxFjAUBgNVBAMMDWNsdXN0ZXItY2EgdjAwggIiMA0GCSqGSIb3DQEB
      AQUAA4ICDwAwggIKAoICAQD3EZnxfI68AHw++IN5VoRYGAi4VlJJnmBaoWGUkEv0
      zHD8EInIpSqNhcXbTPIAoaOuMd4sQ7YwaG78a3bAr9ieeLW83/fm3d9iG0OQrn6/
      4LjvITr18azFL3Xsr5kcxJ3aJo85WH/D6Ma1Zv32mNAMenhN3fkd80aRGPYE7Npy
      Wc9i+Ap0HVlQmv7BVfEc7lYFWZso6wBA89ZjK97lwfxp3TkiJ9Bjxcu0Gw+x1iX/
      6FIXEEYebFJoBQ6iP8td4omhTyrKrRzb8cTqdcn+FTlstB/4LnG8lSfMbnKnwLE4
      yvlQdoXg3ERj79oZ/tZn/GOTxIPMmHAvMAmAckYQ6ChYYCvyGH5e83A5xdToEJYX
      kPRiSVFp8RON595Y1XPlKCwkB6aH4yimhIDQ3hC4+7Wqrw/a5MPCbxmxCo3wlx4b
      2E5zNFvmHg+jTdr0bS2dQOLrkWu1cBRuisAuc6d3gJN8siK315PQVe+IO9e2Vzyi
      fVAH7ON9YQAEiDZ6r+zLYolvCbbqnKqjkY2Nbft+E2fE3N8akBQO7FVJGzxj4IvO
      pqKLyTzcBy64xguviIuTqpjIGcDk59hHUNzb7dHaLg/2DCCZ0AQUvsCFiksciFx7
      UTuy9GRE3NTQ/5FLW9JdnoQ79R6TywpvJTs0dj3WPZ9n8ZWH07kzrtxnDa0h1ZzY
      0QIDAQABo0UwQzAdBgNVHQ4EFgQUKHoK/Pa4phu+nVX9fouzSbybKiYwEgYDVR0T
      AQH/BAgwBgEB/wIBADAOBgNVHQ8BAf8EBAMCAQYwDQYJKoZIhvcNAQENBQADggIB
      AOuYTjor6OZq3A78B2cSJ0jbrB/uh5XWtlqXgPXVH+1M4K/Ffvk43mcSDsQsA331
      jA7Ua5HB0g73D0NMiozi/eleSAcZWYRHDuxLTT7Oh4ZXVXt4nOjcDJpD2qq4W3r6
      0pe+qKE6RgNel1+VrVcIyCogN/uvno3nx9mzCPPtewxuOKtttWtOKM5a8G9m6oDc
      1KIGqSDI6fILOX7XPslpulRJ/U1F4xJMZpEXVaQE+fEDSj5gRsy1LZ2XnIQb2FU9
      BUPCbJzalg+DuZKidWLjFt+aIKai6uQqdYJKGScH+81JPIVLV4MR6REu62u++JTd
      cwrUOVhA2iiHEmoHyboIBSNBudRdKTwBFc3vcOCFifAfbhIVlMSXHofa8wZUzT+m
      q2ySkHmV1ppLMD4Kprp4ARkSG9gNT2nBgXJf9X5geO37OLIZUMs7hR3+YrTiVlSI
      NZ+wMmLkDChxHURWMCiPD5OMZ5HH5FnWJ3D1Pxx+ybmCRmAZVuGWh1onODZ28hQ2
      /OvoGtUdlQrK16M1yt2mWSiOOFCpssAQzGqbPA5TRKenaGv5OV6OgoNEQVw0GjZK
      BDTZ1MEbSC3wo+nIZpTJdvPiIuRYKD7IsyrWc7GvWTa+69Cyg/878E/zGfp6tvmz
      aej6yaM1SDTim4hmFAHife+ROoz9ohVlXttQW96eVTOU
      -----END CERTIFICATE-----
  name: external
  type: external
```

ii) Create a new secret by clicking **Workloads**, **Secrets**. Create a `Key/value secret` then enter the following information:

- Name: `tls-cert-of-development-external`
- Key: `tls.crt`
- Value: DRAG AND DROP the plain-text file containing the certificate here.

iii) Click **Create**.

i. In the folder with the Dockerfile you also find a `kafka-connect.yaml` file. Make a backup of the file and edit it. Change it according to your environment. Compare your edits with the following file to match the details:

```
apiVersion: eventstreams.ibm.com/v1beta2
kind: KafkaConnect
metadata:
  name: my-kafka-connect-external-bootstrap
  annotations:
    eventstreams.ibm.com/use-connector-resources: "true"
spec:
  replicas: 1
  bootstrapServers: development-kafka-bootstrap-
cloudpack4integration.apps.ocp0.sa.boe:443
  image: default-route-openshift-image-registry.apps.ocp0.sa.boe/openshift/my-connect-
cluster-image-1:latest
  template:
    pod:
      imagePullSecrets: []
      metadata:
        annotations:
          eventstreams.production.type: CloudPakForIntegrationNonProduction
          productID: 2a79e49111f44ec3acd89608e56138f5
          productName: IBM Event Streams for Non Production
          productVersion: 11.2.1
          productMetric: VIRTUAL_PROCESSOR_CORE
          productChargedContainers: my-connect-cluster-connect
          cloudpakId: c8b82d189e7545f0892db9ef2731b90d
          cloudpakName: IBM Cloud Pak for Integration
          productCloudpakRatio: "2:1"
  config:
    group.id: connect-cluster
    offset.storage.topic: connect-cluster-offsets
    config.storage.topic: connect-cluster-configs
    status.storage.topic: connect-cluster-status
    config.storage.replication.factor: 3
    offset.storage.replication.factor: 3
    status.storage.replication.factor: 3
  tls:
    trustedCertificates:
      - secretName: tls-cert-of-development-external
        certificate: tls.crt
  authentication:
    type: scram-sha-512
    username: my-credentials
    passwordSecret:
      secretName: my-credentials
      password: password
```

j. Apply the edited yaml file with:

```
oc project cloudpack4integration
oc apply -f kafka-connect.yaml
```

Note: If the image couldn't be pulled because of permissions, refer to the official OpenShift documentation for information about how to allow the image pull.

3. To create the Source Connector that listens for changes, apply the `source-connector.yaml` with `oc apply -f source-connector.yaml`:

```
apiVersion: eventstreams.ibm.com/v1beta2
kind: KafkaConnector
metadata:
  name: my-source-connector
  labels:
    eventstreams.ibm.com/cluster: my-kafka-connect-external-bootstrap
spec:
  class: com.mongodb.kafka.connect.MongoSourceConnector
  tasksMax: 3
  config:
    connection.uri: mongodb://myUserAdmin:password@ibm-mongodb-enterprise-helm-
service.mongodb-test-1.svc.cluster.local:27017
    database: "warehouse"
    collection: "inventory"

    topic.prefix: "mongo"

    copy.existing: true

    key.converter: org.apache.kafka.connect.json.JsonConverter
    key.converter.schemas.enable: false

    value.converter: org.apache.kafka.connect.json.JsonConverter
    value.converter.schemas.enable: false

    publish.full.document.only: true

    pipeline: >
      [{"$match":{"operationType":{"$in":["insert","update","replace"]}}},{"$project":
{"_id":1,"fullDocument":1,"ns":1,"documentKey":1}}]
```

4. To create the Sink Connector that writes the changes to an external MongoDB instance, apply the `sink-connector.yaml` with `oc apply -f sink-connector.yaml`:

```
apiVersion: eventstreams.ibm.com/v1beta2
kind: KafkaConnector
metadata:
  name: my-sink-connector
  labels:
    eventstreams.ibm.com/cluster: my-kafka-connect-external-bootstrap
spec:
  class: com.mongodb.kafka.connect.MongoSinkConnector
  tasksMax: 3
  config:
    connection.uri: 'mongodb://mongodb.example.com:27017'
    database: "shop"
    collection: "inventory"

    # comma separated list
    topics: "mongo.warehouse.inventory"

    post.processor.chain:
com.mongodb.kafka.connect.sink.processor.DocumentIdAdder,com.mongodb.kafka.connect.sink.proce
ssor.KafkaMetaAdder
    key.converter: org.apache.kafka.connect.json.JsonConverter
    key.converter.schemas.enable: false
    value.converter: org.apache.kafka.connect.json.JsonConverter
    value.converter.schemas.enable: false
```

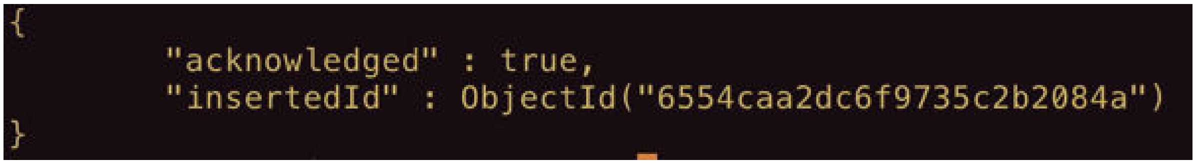
Note: If you don't have an external MongoDB instance you can also use the same MongoDB that was used in the SourceConnector to test if the connectors work properly. Change the connection.uri according to your need.

5. Verify that the connectors are working properly.

   a. Connect to the MongoDB database on the OpenShift side and run these commands to test if the connectors work:

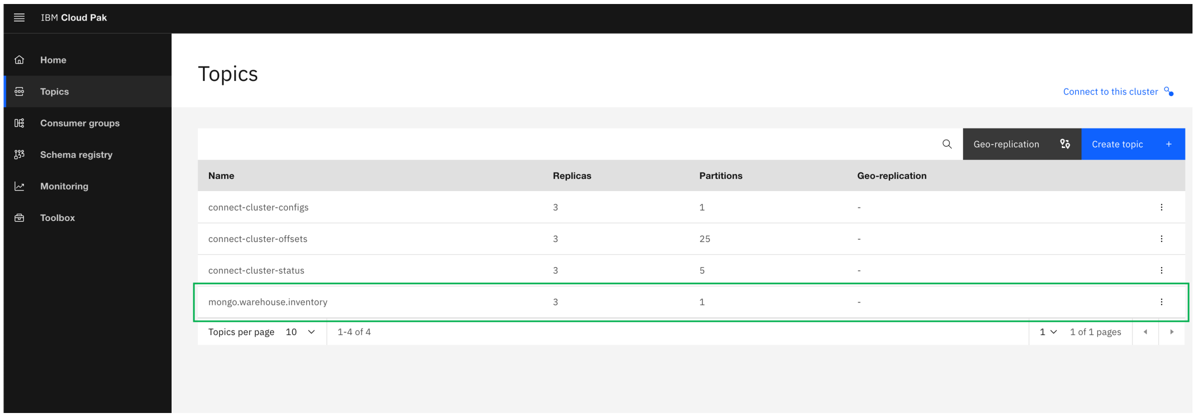
```
mongo -u myUserAdmin -p password --authenticationDatabase admin

use warehouse
db.inventory.insertOne( { name:"game-console-1"} )
db.inventory.insertOne( { name:"game-console-2"} )
```

After the insert command you should get an acknowledgement:



```
{
        "acknowledged" : true,
        "insertedId" : ObjectId("6554caa2dc6f9735c2b2084a")
}
```

The topics are created in the **IBM Event Streams** UI under **Topics**.

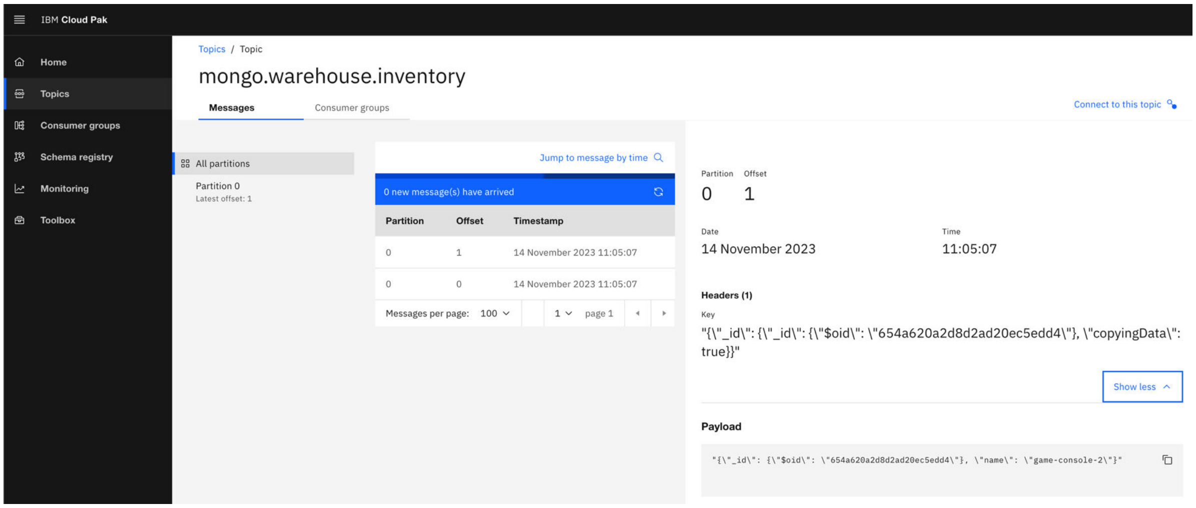


Topics

Name	Replicas	Partitions	Geo-replication
connect-cluster-configs	3	1	-
connect-cluster-offsets	3	25	-
connect-cluster-status	3	5	-
mongo.warehouse.inventory	3	1	-

Topics per page 10 1-4 of 4 1 1 of 1 pages

You should be able to see the messages from above when clicking on the topic:



mongo.warehouse.inventory

Partition 0
Latest offset: 1

0 new message(s) have arrived

Partition	Offset	Timestamp
0	1	14 November 2023 11:05:07
0	0	14 November 2023 11:05:07

Messages per page: 100 1 page 1

Partition 0 Offset 1

Date 14 November 2023 Time 11:05:07

Headers (1)

Key
"{\"_id\": {\"_id\": {\"$oid\": \"654a620a2d8d2ad20ec5edd4\"}, \"copyingData\": true}}"

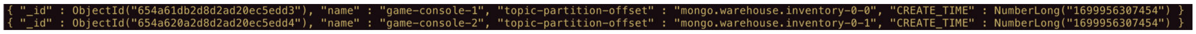
Payload

```
"{\"_id\": {\"$oid\": \"654a620a2d8d2ad20ec5edd4\"}, \"name\": \"game-console-2\"}"
```

b. On the remote MongoDB side, the entries are created:

```
mongo -u myUserAdmin -p password --authenticationDatabase admin

use shop
show collections
db.inventory.find()
```

The output should look similar to the following:



```
{ "_id" : ObjectId("654a61db2d8d2ad20ec5edd3"), "name" : "game-console-1", "topic-partition-offset" : "mongo.warehouse.inventory-0-0", "CREATE_TIME" : NumberLong("1699956307454") }
{ "_id" : ObjectId("654a620a2d8d2ad20ec5edd4"), "name" : "game-console-2", "topic-partition-offset" : "mongo.warehouse.inventory-0-1", "CREATE_TIME" : NumberLong("1699956307454") }
```

References

A list of sources and documents mentioned in this paper.

IBM

- NFS Server
- Storage considerations

IBM Event Streams

- Prerequisites
- Connector catalog

IBM Cloud Pak for Integration

- Compute resources for development environments
- Adding catalog sources to a cluster
- Finding and applying your entitlement key by using the UI (online installation)

Red Hat

- Reference Architecture Guide
- Use Red Hat HA to make OpenShift highly available
- Product Documentation for Red Hat OpenShift Data Foundation
- Preferred resource requirements for installing a cluster with RHEL KVM on IBM Z and IBM LinuxONE
- Resource requirements for IBM Z and LinuxONE infrastructure
- Red Hat support:
  - Components in General
  - Red Hat customer portal solutions

MongoDB

- ibm-mongodb-enterprise-helm
- IBMz MongoDB EE Operator
- Convert a Standalone to a Replica Set
- "The $changeStream stage is only supported on replica sets" error while using mongodb-source-connect
- Install MongoDB Enterprise Edition on Red Hat or CentOS
- org/mongodb/kafka/mongo-kafka-connect/1.10.1

Notices

References in this publication to IBM products, programs, or services do not imply that intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Subject to IBM's valid intellectual property or other legally protectable rights, any functionally equivalent product, program, or service may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user. IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
USA

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently IBM created programs and other programs (including this one) and (ii) the mutual use of the information, which has been exchanged, should contact:

IBM Deutschland Research & Development GmbH
Department 3282
Schönaicher Strasse 220
D-71032 Böblingen
Federal Republic of Germany
Attention: Information Request

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

Any pointers in this publication to websites are provided for convenience only and do not in any manner serve as an endorsement of these websites. Use of these materials is at your own risk.

Trademarks and service marks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Red Hat®, JBoss®, OpenShift®, Fedora®, Hibernate®, Ansible®, CloudForms®, RHCA®, RHCE®, RHCSA®, Ceph®, and Gluster® are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.