Previous Doc. Number	Document Type **PUMA-Installation procedure**	Document ID **IS_1693337**
	GeodIS	Version **1.0**
Title **IP - Installation Procedure - Kafka Connect**		

Reason for Revision

	Approval Name	Date (Universal Time)	Reason for Signature
Approval	GRZESIUK Grzegorz Sanofi	24 Nov 2020 08:36:52	I am approving this document
	Effective Date:	24 Nov 2020	

INSTALLATION PROCEDURE

KAFKA CONNECT

	Name / Project Role	Signature	Date *(dd/mmm/yyyy)*
Author	Radoslaw Przewuski		See GeodIS
Approver*	Grzegorz Grzesiuk		See GeodIS

* *Signatures above indicate approvers agree with the content of the document and ensure it will be applicable within their domain of expertise.*

Distribution List

Names & Department	Rationale
ICOE Team	Information and Distirbution

Version History

Version	Date *(dd/mmm/yyyy)*	Modification
1.0	See GeodIS	Initial Version

Table of Contents

1. INTRODUCTION

The objective of the Installation Procedure is to provide detailed step by step instructions for installing Kafka Connect on OpenShift cluster.

2. PREREQUISITES

To be able to properly install Kafka Connect on OpenShift cluster, you would need to fulfill the following conditions:

- Have knowledge about the OpenShift environment, architecture and CLI
- Have **admin** access to **tibco-test** namespace on **e11** OpenShift cluster
- Have **oc** installed on your local machine
- Know the location and all the configuration data for Kafka you want to connect with

3. INSTALLATION ACTIVITIES

3.1. Configuration

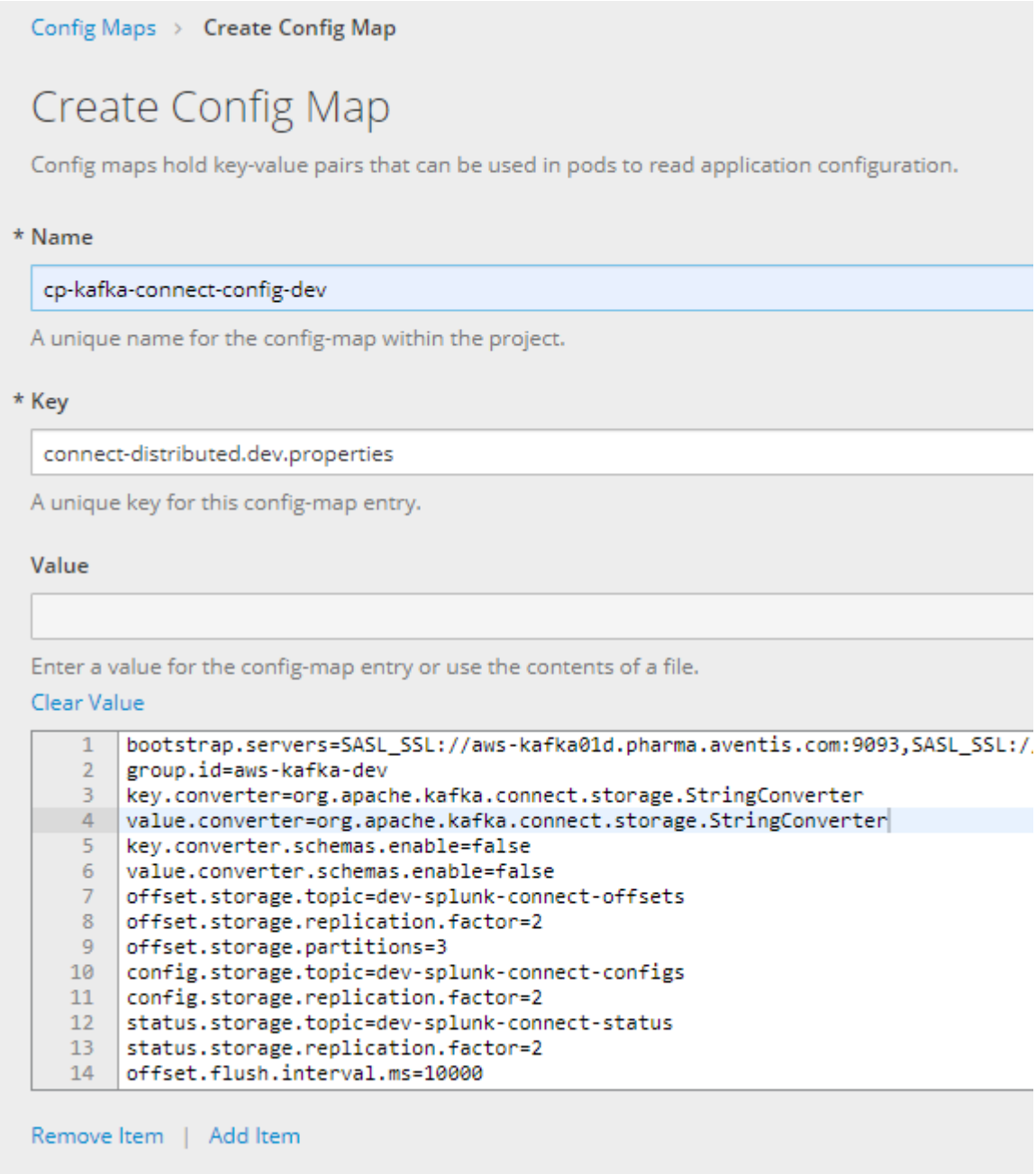
3.1.1. Create config map

To provide Kafka connect with your configuration, you will need to create a config map on OpenShift cluster. To do so, you will need to login to tibco-test namespace on e11 cluster and go to Resources -> Config Maps page.

Click on ‘Create Config Map’ button in the upper right corner and provide all the required data:

- Name – should fall into pattern ‘cp-kafka-connect-config-<<environment_name>>’
- Key – should fall into pattern ‘connect-distributed.<<environment_name>>.properties’
- Value – fill the text area below with correctly filled kafka connect configuration file

Filled data should look like this:



Click on ‘Create’ button afterwards.

3.1.2. Import Kafka certificate

If your Kafka uses SSL, you will need to import its certificate. To do so, you will need to login to OpenShift tibco-test cluster via oc CLI and invoke this command:

oc create cm cp-kafka-connect-<<envrionment_name>>-cert --from-file=<<jks_certificate>> -n tibco-test

3.1.3. Create kafka-client_jaas.conf config map

Create another config map (just like in 3.1.1.) for kafka-client_jaas.conf with provided login credentials

for Kafka and external system you want to connect with. File should look similar to the one below:

```
KafkaClient {
   org.apache.kafka.common.security.plain.PlainLoginModule required
   username="admin"
   password="password";
};

Client {
   org.apache.kafka.common.security.plain.PlainLoginModule required
   username="admin"
   password="password";
};
```

3.2. Deployment

3.2.1. Create deployment YAML

Fill up the deployment yaml from this template:

```yaml
apiVersion: apps.openshift.io/v1
kind: DeploymentConfig
metadata:
  labels:
    app: kafka-connect
  name: <<ENVIRONMENT>>-kafka-connect
  namespace: tibco-test
spec:
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    app: <<ENVIRONMENT>>-kafka-connect
    deploymentconfig: <<ENVIRONMENT>>-kafka-connect
  strategy:
    activeDeadlineSeconds: 21600
    resources: {}
    rollingParams:
      intervalSeconds: 1
      maxSurge: 25%
      maxUnavailable: 25%
      timeoutSeconds: 600
      updatePeriodSeconds: 1
    type: Rolling
  template:
    metadata:
      annotations:
```

```
        openshift.io/generated-by: OpenShiftWebConsole
      creationTimestamp: null
      labels:
        app: <<ENVIRONMENT>>-kafka-connect
        deploymentconfig: <<ENVIRONMENT>>-kafka-connect
    spec:
      containers:
        - args:
            - /etc/kafka-connect/config/connect-distributed.properties
          command:
            - /usr/bin/connect-distributed
          env:
            - name: KAFKA_ENV
              value: <<ENVIRONMENT>>
            - name: EXTRA_ARGS
              value: >-
                -Djava.security.auth.login.config=/etc/kafka-
connect/config/kafka-client_jaas.conf
                -Djavax.net.ssl.keyStore=/etc/kafka-connect/certs/aws-kafka-
<<ENVIRONMENT>>.jks
                -Djavax.net.ssl.trustStore=/etc/kafka-connect/certs/aws-kafka-
<<ENVIRONMENT>>.jks
                -Djavax.net.ssl.keyStorePassword=PgNfcw3z0ddcvoFT0SDI
                -Djavax.net.ssl.trustStorePassword=PgNfcw3z0ddcvoFT0SDI
          image: >-
            docker-registry.default.svc:5000/tibco-test/cp-kafka-connect-
imagestream
          imagePullPolicy: IfNotPresent
          name: <<ENVIRONMENT>>-kafka-connect
          ports:
            - containerPort: 8083
              protocol: TCP
            - containerPort: 9092
              protocol: TCP
          resources:
            limits:
              cpu: '1'
              memory: 2Gi
            requests:
              cpu: 1m
              memory: 1Gi
          securityContext:
            runAsGroup: 1000
            runAsUser: 1000
          terminationMessagePath: /dev/termination-log
```

```
          terminationMessagePolicy: File
          volumeMounts:
            - mountPath: /etc/kafka-connect/config/connect-
distributed.properties
              name: cp-kafka-connect-config
              subPath: connect-distributed.properties
            - mountPath: /etc/kafka-connect/config/kafka-client_jaas.conf
              name: cp-kafka-connect-config
              subPath: kafka-client_jaas.conf
            - mountPath: /etc/kafka-connect/certs/<<CERTIFICATE_NAME>>
              name: cp-kafka-connect-<<ENVIRONMENT>>-cert
              subPath: <<CERTIFICATE_NAME>>
            - mountPath: /etc/kafka-connect/log4j.properties
              name: connect-log4j
              subPath: log4j.properties
      dnsPolicy: ClusterFirst
      restartPolicy: Always
      schedulerName: default-scheduler
      securityContext: {}
      terminationGracePeriodSeconds: 30
      volumes:
        - configMap:
            defaultMode: 420
            items:
              - key: kafka-client_jaas.conf
                path: kafka-client_jaas.conf
              - key: connect-distributed.<<ENVIRONMENT>>.properties
                path: connect-distributed.properties
            name: cp-kafka-connect-config
          name: cp-kafka-connect-config
        - configMap:
            defaultMode: 420
            items:
              - key: <<CERTIFICATE_NAME>>
                path: <<CERTIFICATE_NAME>>
            name: cp-kafka-connect-<<ENVIRONMENT>>-cert
          name: cp-kafka-connect-<<ENVIRONMENT>>-cert
        - configMap:
            defaultMode: 420
            items:
              - key: connect-log4j.properties
                path: log4j.properties
            name: cp-kafka-connect-config
          name: connect-log4j
  test: false
```

```
  triggers:
    - type: ConfigChange
    - imageChangeParams:
        automatic: true
        containerNames:
          - <<ENVIRONMENT>>-kafka-connect
        from:
          kind: ImageStreamTag
          name: 'cp-kafka-connect-imagestream:latest'
          namespace: tibco-test
        lastTriggeredImage: >-
          docker-registry.default.svc:5000/tibco-test/cp-kafka-connect-
imagestream@sha256:d11051ab63d19ea8793437fcf89b58c74570712d521eb0ac09239686c38
ac0e9
      type: ImageChange
```

Each occurrence of <<ENVIRONMENT>> string should be changed to your environment name (e.g. ‘dev’, ‘prod’) and <<CERTIFICATE_NAME>> to the imported jks name.

After updating the template, go back to OpenShift and find ‘Add to Project’ button in the upper right corner. Click on it, choose ‘Import YAML / JSON’. New dialog box should appear, where you should paste your template and click ‘Create’

If everything went fine, you will be redirected to deployment page and kafka connect will be starting up.